

APLIKASI TEKNOLOGI SELULER Dengan **Android**



YUSUF PRIMA SANTOSO TEGUH

Dr. Joseph Teguh Santoso, M.Kom.

APLIKASI TEKNOLOGI SELULER dengan Android

Penulis :

Dr. Joseph Teguh Santoso, S.Kom., M.Kom

ISBN : 9 786235 734491

Editor :

Muhammad Sholikan, M.Kom

Penyunting :

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :

Irdha Yunianto, S.Ds., M.Kom

Penebit :

Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :

Jl. Majapahit no 605 Semarang

Telp. (024) 6723456

Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :

Universitas STEKOM

Jl. Majapahit no 605 Semarang

Telp. (024) 6723456

Fax. 024-6710144

Email : info@stekom.ac.id

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa ijin tertulis dari penerbit

KATA PENGANTAR

Puji syukur penulis panjatkan atas kehadiran Tuhan karena buku yang berjudul *“Aplikasi Teknologi Seluler Dengan Android”* dapat terselesaikan. Aplikasi seluler telah merambah secara dramatis di berbagai bidang dan telah mengubah kehidupan banyak orang. Mengembangkan aplikasi seluler yang efektif telah menjadi masalah penting bagi perusahaan saat ini untuk menyebarkan layanan atau produksi mereka, dan membangun hubungan langsung dengan pelanggan. Aplikasi mobile merupakan pilihan untuk mendapatkan perhatian penuh dari penggunanya, yang membedakan dari desktop yang memiliki penawaran multitask. Jumlah unduhan dan eksekusi aplikasi seluler telah meningkat pesat selama bertahun-tahun dengan perkembangan pesat teknologi seluler. Membedakan dari aplikasi seluler lainnya telah menjadi isu penting bagi perusahaan untuk meningkatkan nilai aplikasi seluler. Oleh karena itu, menemukan cara yang efektif untuk memaksimalkan kinerja aplikasi seluler menciptakan permintaan mendesak bagi praktisi aplikasi seluler kontemporer. Menggunakan teknik lanjutan dari aplikasi seluler dianggap sebagai pendekatan yang efektif untuk membuat aplikasi seluler menonjol dalam sekelompok pilihan aplikasi.

Buku ini berfokus pada pengenalan teknik lanjutan aplikasi seluler dan upaya untuk pembelajaran mahasiswa dalam keterampilan menggunakan pendekatan canggih tersebut tentang pengembangan aplikasi seluler praktis. Pendekatan yang terlibat dalam buku ini membahas pencapaian utama terbaru dari teknologi seluler dan jaringan nirkabel. Mahasiswa dapat memperoleh pengetahuan tentang cakupan luas aplikasi seluler dalam kerangka kerja Android. Tujuan instruksional adalah untuk berhasil menyebarkan metode pengembangan aplikasi seluler baru dan memungkinkan penemuan pengetahuan di lapangan. Mahasiswa akan memperoleh teknik seluler yang diperbarui dan keterampilan pengembangan aplikasi yang sesuai dengan studi tingkat pascasarjana setelah memahami isi buku ini. Instruksi akan mencakup beberapa bidang, termasuk algoritma canggih, sistem tertanam, arsitektur aplikasi seluler baru, dan paradigma komputasi awan seluler.

Ada dua konsentrasi utama dalam buku ini, yaitu pengembangan aplikasi seluler dan algoritma. Kedua konsentrasi ini dicakup oleh tiga bagian, yang mewakili tiga dimensi utama dalam domain pengembangan aplikasi seluler saat ini. Bagian 1 Keterampilan desain dan pengembangan aplikasi seluler. Ini termasuk Bab 2 hingga 4. Dalam Bab 2, kami menawarkan permulaan cepat di Android dari memperkenalkan Java hingga menjalankan aplikasi Android di telepon nyata. Bab 3 memberikan gambaran umum tentang konsep dan desain utama dalam aplikasi seluler Android. Terakhir, Bab 4 memperkenalkan keterampilan grafis 2D dan desain UI serta multimedia dalam aplikasi seluler Android.

Bagian 2 dalam buku ini mencakup tentang pengoptimalan aplikasi seluler tingkat lanjut. Bab 5 sampai 8 fokus pada aspek ini. Kami memberikan gambaran umum tentang sistem tertanam seluler dan arsitekturnya di Bab 5. Bab 6 memperkenalkan teknik penyimpanan data di Android. Selain itu, kami membahas pengetahuan optimasi seluler

dengan pemrograman dinamis di Bab 7. Akhirnya, presentasi optimasi seluler dengan penjadwalan loop diberikan di Bab 8.

Bagian ke 3 Teknik aplikasi seluler dalam teknologi yang sedang berkembang. Hal ini dibahas dalam Bab 9 dan Bab 10. Dalam Bab 9, kita membahas teknik komputasi awan mobile dalam penyebaran aplikasi mobile. Selain itu, kami menawarkan teknik lanjutan lainnya yang digunakan dalam data besar di Bab 10, yang berfokus pada penyimpanan data besar seluler.

Pembelajaran dalam buku ini secara keseluruhan adalah untuk memungkinkan pembaca mengetahui pendekatan pengembangan aplikasi seluler Android menggunakan teknik canggih untuk mencapai kinerja yang ditingkatkan. Teknologi baru yang berdampak pada aplikasi seluler juga dipertimbangkan. Target konsentrasi instruksional memfasilitasi kesadaran mahasiswa tentang pengetahuan dengan menggabungkan representasi pengetahuan dengan latihan praktis. Akhir kata semoga buku ini berguna bagi para pembaca.

Semarang, Mei 2022

Penulis

Dr. Joseph Teguh Santoso, S.Kom, M.Kom

DAFTAR ISI

Halaman judul	i
Kata Pengantar	iii
Daftar isi	v
BAGIAN I KETERAMPILAN DESAIN DAN PENGEMBANGAN APLIKASI SELULER	1
BAB 1 IKHTISAR APLIKASI SELULER DAN ANTARMUKA SELULER	1
1.1 Sistem Seluler	1
1.2 Antarmuka Dan Aplikasi Seluler	3
1.2.1 Pengoptimalan Dalam Sistem Seluler	4
1.2.2 Sistem Tertanam Seluler	5
1.3 Awan Seluler	5
1.3.1 Aplikasi Big Data Dalam Sistem Seluler	6
1.3.2 Keamanan Data Dan Perlindungan Privasi Dalam Sistem Seluler	7
1.3.3 Konsep Aplikasi Seluler	9
1.3.4 Pengenalan Singkat Android Dan Kerangkanya	9
1.4 Latihan	14
BAB 2 MULAI CEPAT DI ANDROID	16
2.1 Menginstal Java	16
2.2 Menginstal Pengembangan Terintegrasi Lingkungan	16
2.3 Menginstal Android Sdk	17
2.4 Menciptakan Aplikasi Android	18
2.5 Perangkat Virtual Android	19
2.6 Latihan	24
BAB 3 PENGENALAN KONSEP KUNCI ANDROID	26
3.1 Komponen Aplikasi	26
3.1.1 Kegiatan	26
3.1.2 Layanan	26
3.1.3 Penyedia Konten	27
3.1.4 Tujuan	27
3.2 Sumber Daya Aplikasi	28
3.3 Aplikasi Terbaik	30
3.3.1 Elemen	31
3.3.2 Atribut	31
3.3.3 Mendeklarasikan Nama Kelas	31
3.3.4 Beberapa Nilai	31
3.3.5 Nilai Sumber Daya	32
3.3.6 Nilai Sengatan	32
3.4 Latihan	33

3.4.1 Latihan Dasar	33
3.4.2 Latihan Lanjutan	34
BAB 4 GRAFIK DAN MULTIMEDIA 2D DI ANDROID	35
4.1 Pengenalan Teknik Grafik 2D	35
4.1.1 Warna	35
4.1.2 Cat	37
4.1.3 Jalur	37
4.1.4 Kanvas	38
4.1.5 Dapat Digambar	39
4.1.6 Pemilih Tombol	42
4.2 Desain Ui Lanjutan	43
4.2.1 Beberapa Layar	43
4.2.2 Bilah Tindakan	45
4.2.3 Tampilan Kustom	46
4.3 Tinjauan Multimedia Di Android	47
4.3.1 Memahami Mediaplayer Kelas	47
4.3.2 Siklus Hidup Status Mediaplayer	48
4.4 Implementasi Audio Di Android	48
4.5 Mengeksekusi Video Di Android	50
4.6 Latihan	53
4.6.1 Latihan Dasar	53
4.6.2 Latihan Lanjutan	54
BAGIAN II PENGOPTIMALAN APLIKASI SELULER TINGKAT LANJUT	56
BAB 5 ARSITEKTUR SISTEM EMBEDDED SELULER	56
5.1 Sistem Tertanam	56
5.2 Algoritma Penjadwalan	58
5.2.1 Konsep Dasar	58
5.2.2 Algoritma Penjadwalan First-Come, First-Served	59
5.2.3 Algoritma Penjadwalan Singkat-Pekerjaan-Pertama	60
5.2.4 Multiprosesor	62
5.2.5 Algoritma Penjadwalan Asap Dan Alap	66
5.3 Teknologi Memori	70
5.4 Sistem Termasuk Seluler	71
5.4.1 Sistem Tertanam Di Perangkat Seluler	71
5.4.2 Sistem Tertanam Di Android	72
5.4.3 Manajemen Daya Android	73
5.4.4 Sistem Tertanam Di Aplikasi Seluler	74
5.5 Mekanisme Pesan Dan Komunikasi	75
5.5.1 Mekanisme Pesan	75
5.5.2 Mekanisme Komunikasi	76
5.6 Latihan	78

BAB 6 PENYIMPANAN DATA DAN OPERASI SQLITE	80
6.1 Data Lokal	80
6.1.1 Penyimpanan Internal Dan Eksternal	80
6.1.2 Menyimpan File Di Penyimpanan Internal	81
6.1.3 Menyimpan File Di Penyimpanan Eksternal	83
6.1.4 Menghapus File	85
6.1.5 Query Space	86
6.2 Database Sqlite	87
6.2.1 Struktur Tabel	87
6.2.2 Operasi Crud	88
6.2.3 Penggunaan Teknik Sqlite	93
6.3 Penyedia Konten	94
6.4 Latihan	96
6.4.1 Latihan Dasar	96
6.4.2 Latihan Tingkat Lanjut	97
BAB 7 PENGOPTIMALAN SELULER DENGAN PEMROGRAMAN DINAMIS	98
7.1 Pengenalan Sistem Embedded Heterogen Dan Pemrograman Dinamis	98
7.2 Model Waktu Tetap	99
7.2.1 Penugasan Heterogen	99
7.2.2 Meminimalkan Biaya Dengan Penjadwalan	101
7.3 Model Waktu Probabilistik	107
7.3.1 Pengenalan Model Waktu Probabilistik	107
7.3.2 Solusi Untuk Tugas Heterogen Soal	110
7.3.3 Membuat Tabel D	111
7.3.4 Contoh Pembuatan Tabel D	113
7.4 Masalah Waktu Polinomial Nondeterministik	119
7.5 Latihan	119
7.5.1 Pertanyaan Mendasar	119
7.5.2 Pertanyaan Praktis	120
7.6 Daftar Istilah	122
BAB 8 PENGOPTIMALAN SELULER DENGAN PENJADWALAN LOOP	123
8.1 Pendahuluan	123
8.2 Teknik Dan Model Grafik Dasar	124
8.2.1 Grafik Aliran Data Dalam Penjadwalan Loop	124
8.2.2 Pengaturan Waktu Dan Pembukaan	126
8.3 Optimasi Waktu Dasar	127
8.4 Optimasi Waktu Dan Daya Dengan Penjadwalan Loop	130
8.4.1 Grafik Aliran Data Probabilistik	131
8.4.2 Penjadwalan Loop Dan Komputasi Paralel	132
8.5 Kesimpulan	138
8.6 Latihan	138

8.6.1	Pertanyaan Mendasar	138
8.6.2	Pertanyaan Praktis	139
8.7	Daftar Istilah	141
	BAGIAN III TEKNIK APLIKASI SELULER DALAM TEKNOLOGI YANG BERKEMBANG	143
	BAB 9 KOMPUTASI AWAN SELULER DALAM APLIKASI SELULER	143
9.1	Pendahuluan	143
9.2	Konsep Komputasi Cloud Seluler	144
9.2.1	Struktur Teknologi Mobile Cloud Computing	144
9.2.2	Perbedaan Antara Cloud Computing Dan Mobile Cloud	145
9.2.3	Komputasi Seluler	146
9.2.4	Jaringan Nirkabel	147
9.3	Teknik Utama Mobile Cloud Computing	152
9.3.1	Virtualisasi	152
9.3.2	Model Pemrograman Paralel	154
9.3.3	Penyimpanan Terdistribusi Massal	155
9.4	Arsitektur Komputasi Cloud Mobile	156
9.5	Latihan	157
9.5.1	Pertanyaan Mendasar	157
9.5.2	Pertanyaan Praktis	157
9.6	Daftar Istilah	158
	BAB 10 SINKRONISASI DATA ANDROID DALAM DATA BESAR	161
10.1	Gambaran Umum Data Besar	161
10.1.1	Memahami Tipe Data	161
10.1.2	Mengkategorikan Model Big Data	162
10.1.3	Tantangan Saat Ini Dalam Big Data	163
10.2	Pemrosesan Data Besar	164
10.2.1	Pembelajaran Mesin	164
10.3	Penyimpanan Data Besar Seluler	170
10.3.1	Pengenalan Dan Konsep Dasar	170
10.3.2	Arsitektur Memori Heterogen	171
10.3.3	Alokasi Data Pemrograman Dinamis Multidimensi	173
10.4	Masalah Keamanan Dan Privasi	180
10.5	Deduplikasi Data	183
10.6	Latihan	184
10.6.1	Pertanyaan Mendasar	184
10.6.2	Pertanyaan Praktis	184
	Daftar Pustaka	186

BAGIAN I

KETERAMPILAN DESAIN DAN PENGEMBANGAN APLIKASI SELULER

BAB 1

IKHTISAR APLIKASI SELULER DAN ANTARMUKA SELULER

Sistem Seluler Dan Aplikasi Seluler adalah dua aspek mendasar dalam pengembangan aplikasi seluler Android. Dalam bab ini, kami memperkenalkan ikhtisar sistem seluler dan aplikasi seluler, yang meliputi:

1. Pengenalan sistem seluler.
2. Antarmuka seluler dan aplikasi dalam sistem seluler.
3. Optimalisasi pada sistem mobile.
4. Sistem tertanam seluler.
5. Komputasi awan seluler.
6. Data besar dalam sistem seluler.
7. Keamanan data dan perlindungan privasi di sistem seluler.
8. Aplikasi seluler.
9. Pengenalan android.

1.1 SISTEM SELULER

Sistem seluler mencakup perangkat seluler, sistem operasi seluler, jaringan nirkabel, aplikasi seluler, dan platform aplikasi. Perangkat mobile tidak hanya terdiri dari smartphone tetapi juga komputer genggam lainnya, seperti tablet dan Personal Digital Assistant (PDA). Perangkat seluler memiliki sistem operasi seluler dan dapat menjalankan berbagai jenis aplikasi. Bagian terpenting dari perangkat seluler adalah *Central Processing Unit* (CPU), memori, dan penyimpanan, yang serupa dengan desktop tetapi kinerjanya lebih lemah daripada perangkat lokal. Sebagian besar perangkat seluler juga dapat dilengkapi dengan kemampuan Wi-Fi, Bluetooth, dan Sistem Pemosisian Global (GPS), dan dapat terhubung ke Internet, perangkat berkemampuan Bluetooth lainnya, dan sistem navigasi satelit. Sementara itu, sebuah perangkat bergerak dapat dilengkapi dengan beberapa kemampuan interaksi manusia-komputer, seperti kamera, mikrofon, sistem audio, dan beberapa sensor.

Semua jenis perangkat seluler berjalan di berbagai Sistem Operasi (OS) seluler, juga disebut OS seluler, seperti iOS dari Apple Inc., Android dari Google Inc., Windows Phone dari Microsoft, Blackberry dari BlackBerry, Firefox OS dari Mozilla, dan Sailfish OS dari Jolla. Perangkat seluler sebenarnya menjalankan dua sistem operasi seluler. Selain sistem operasi seluler yang dapat dilihat pengguna akhir, perangkat seluler juga menjalankan sistem operasi kecil yang mengelola segala sesuatu yang berhubungan dengan radio. Karena ketergantungan waktu yang tinggi, sistem ini merupakan sistem operasi real-time berpemilik tingkat rendah. Namun, sistem tingkat rendah ini rentan terhadap keamanan jika beberapa stasiun pangkalan

berbahaya memperoleh kontrol tingkat tinggi atas perangkat seluler. Kami akan membahas masalah keamanan di perangkat seluler nanti.

Perangkat seluler dapat terhubung ke Internet melalui jaringan nirkabel. Ada dua jaringan nirkabel populer untuk perangkat seluler: jaringan seluler dan Wi-Fi. Jaringan seluler khusus untuk transceiver portabel. Jaringan seluler dilayani oleh setidaknya satu transceiver lokasi tetap, yang disebut situs sel atau stasiun pangkalan, seperti yang ditunjukkan pada Gambar 1.1. Setiap perangkat seluler menggunakan rangkaian frekuensi yang berbeda dari perangkat yang berdekatan, yang berarti perangkat seluler harus terhubung ke base station sebelum mengakses Internet. Demikian pula, ketika perangkat seluler yang menggunakan jaringan seluler ingin menghubungkan perangkat seluler lain, perangkat tersebut harus terhubung ke beberapa BTS sebelum berkomunikasi dengan perangkat target melalui BTS.



Gambar 1.1 Struktur jaringan seluler.



Gambar 1.2 Logo Wi-Fi.

Wi-Fi adalah teknologi nirkabel area lokal, yang memungkinkan perangkat seluler berpartisipasi dalam jaringan komputer menggunakan pita radio 2,4 GHz dan 5 GHz. Gambar 1.2 mewakili dua logo umum Wi-Fi. Perangkat seluler dapat terhubung ke Internet melalui titik akses jaringan nirkabel. Jangkauan titik akses yang valid terbatas, dan intensitas sinyal menurun seiring dengan bertambahnya jarak. Wi-Fi memungkinkan penyebaran Jaringan Area Lokal (LAN) yang lebih murah, terutama untuk ruang di mana kabel tidak dapat dijalankan. Enkripsi Akses Terlindungi Wi-Fi (WPA2) dianggap sebagai pendekatan yang aman dengan memberikan frasa sandi yang kuat. Sinyal Wi-Fi menempati lima saluran di pita 2,4 GHz. Setiap dua nomor saluran berbeda lima atau lebih. Banyak perangkat konsumen yang lebih baru mendukung standar 802.11ac 2 terbaru, yang menggunakan 5 GHz dan mampu melakukan throughput WLAN multistasiun setidaknya 1 gigabit per detik.

1. Hz adalah satuan frekuensi dalam Sistem Satuan Internasional dan didefinisikan sebagai satu siklus per detik. Satu gigahertz (GHz) mewakili 10⁹ Hz.
2. IEEE 802.11ac telah disetujui pada Januari 2014 oleh IEEE Standards Association.

Aplikasi seluler adalah program yang dirancang untuk berjalan di ponsel cerdas, komputer tablet, dan perangkat seluler lainnya. Aplikasi seluler muncul pada tahun 2008 dan dioperasikan oleh pemilik sistem operasi seluler. Saat ini, platform distribusi digital yang paling populer untuk aplikasi seluler adalah App Store, Google Play, Windows Phone Store, dan BlackBerry App World, seperti yang ditunjukkan pada Gambar 1.3. Platform ini masing-masing dikembangkan oleh Apple Inc., Google, Microsoft, dan BlackBerry Ltd., dan menyediakan aplikasi yang berbeda, yang hanya dapat digunakan pada sistem operasi mereka sendiri.



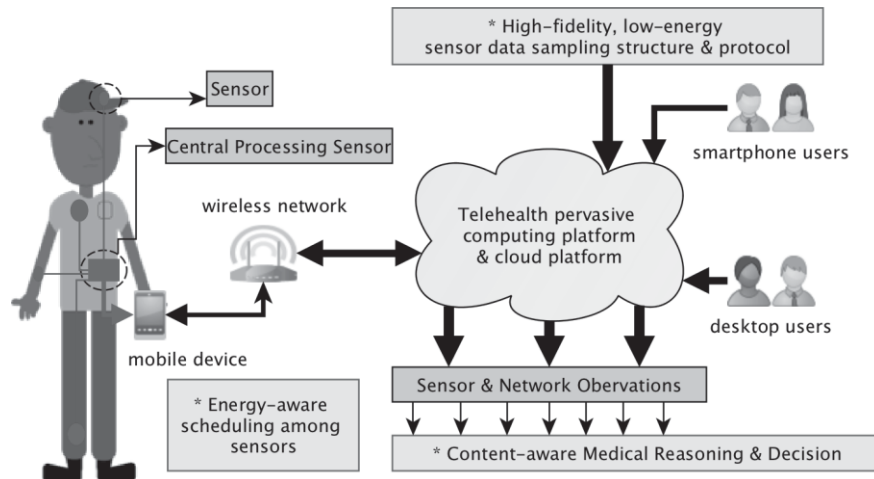
Gambar 1.3 Empat platform yang mendominasi untuk aplikasi seluler.

1.2 ANTARMUKA DAN APLIKASI SELULER

Perangkat seluler, sampai batas tertentu, jauh lebih kuat daripada desktop. Mereka sangat pribadi, selalu aktif, selalu dengan pengguna, biasanya terhubung, dan dapat dialamatkan secara langsung. Selain itu, mereka merangkak dengan sensor yang kuat dengan berbagai fungsi yang mendeteksi lokasi, akselerasi, orientasi, gerakan, kedekatan, dan kondisi sekitarnya. Portabilitas perangkat seluler yang dikombinasikan dengan sensor yang kuat membuat antarmuka seluler sangat berharga untuk menggunakan perangkat seluler.

Antarmuka Pengguna (UI) adalah tampilan dan nuansa sistem di layar, termasuk cara kerjanya, skema warnanya, dan cara sistem merespons operasi pengguna. Interaksi tidak hanya mencakup operasi aktif pengguna, tetapi juga pasif. Operasi pasif pengguna mencakup lokasi, pergerakan, dan informasi lain pengguna yang tidak memerlukan operasi aktif pengguna. Kami akan mengambil telehealth sebagai contoh antarmuka seluler. Telehealth adalah penyampaian layanan dan informasi terkait kesehatan melalui teknologi telekomunikasi.

Kita dapat memisahkan sistem telehealth menjadi beberapa mode: store-and-forward, real-time, pemantauan pasien jarak jauh, dan konsultasi elektronik, seperti yang ditunjukkan pada Gambar 1.4. Setiap mode menyelesaikan tugasnya masing-masing dan mencapai seluruh proses pengumpulan data dari pengguna, mengirimkan data ini ke organisasi medis atau klinis, penalaran dan keputusan medis, dan mengirim kembali ke pengguna. Pada langkah pertama, pengamatan kehidupan sehari-hari dan data klinis ditangkap dan disimpan di perangkat seluler. Semua sensor yang mengumpulkan dan merekam data adalah perangkat medis heterogen dengan fitur biaya dan waktu yang berbeda. Kemudian perangkat seluler mentransmisikan informasi ini ke platform komputasi pervasif Telehealth dan platform cloud melalui jaringan nirkabel.



Gambar 1.4 Struktur sistem telehealth.

Akibatnya, tantangan utama termasuk menemukan pendekatan pengumpulan data dari pengguna dengan menggunakan sensor dan sensor penjadwalan untuk mencapai tujuan energi-sadar. Proses transmisi data merupakan bagian dari sistem real-time. Berbeda dengan sistem real-time normal, transmisi data dalam telehealth berada dalam kondisi nirkabel. Serupa dengan langkah pertama, ada berbagai jalur jaringan dengan persyaratan biaya dan waktu yang berbeda, yang menghasilkan tantangan besar untuk keamanan dan integritas data.

Lebih lanjut, penalaran dan keputusan medis yang sadar konteks merupakan isu penting lainnya dalam sistem telehealth. Konteks dapat merujuk pada karakteristik dunia nyata, seperti suhu, waktu, atau lokasi. Menggabungkan dengan informasi pribadi pengguna, pertimbangan medis dan keputusan berfokus pada masalah analitik data, penambahan, dan pembuatan profil. Sebagai kesimpulan, semua tantangan yang disebutkan di atas dapat diringkas sebagai masalah umum: bagaimana meminimalkan total biaya telehealth heterogen sambil menyelesaikan seluruh diagnosis dalam batasan waktu tertentu.

1.2.1 Pengoptimalan dalam Sistem Seluler

Semua perangkat seluler saat ini adalah perangkat bertenaga baterai. Tingginya penggunaan perangkat seluler membuat mereka sulit untuk terus mengisi daya seperti desktop, sehingga peningkatan masa pakai baterai pada perangkat seluler semakin mendapat perhatian. Selain beberapa operasi hemat energi oleh pengguna, ada beberapa penelitian yang berfokus pada optimasi pada sistem mobile. Masalah optimasi, sampai batas tertentu, adalah tradeo antara beberapa kendala. Sebelum berbicara tentang pengoptimalan, mari kita bahas beberapa kendala dalam sistem seluler.

Kendala pertama dan terpenting adalah energi. Yang kedua adalah performa. Yang ketiga adalah kecepatan jaringan ke Internet. Yang keempat adalah sumber daya perangkat seluler. Kendala tersebut saling terkait dan saling membatasi satu sama lain. Misalkan dalam situasi ekstrem, seseorang mematikan perangkat selulernya. Dalam situasi ini, masa pakai baterai dapat bertahan hampir tak terbatas tanpa mempertimbangkan pengosongan baterai sendiri. Namun, perangkat seluler dalam

situasi itu tidak berguna, dan tidak ada yang membeli perangkat seluler hanya untuk hiasan. Jelas bahwa semakin banyak fungsi yang digunakan pengguna, semakin banyak energi yang dikonsumsi perangkat. Demikian pula, kinerja terkait dengan kecepatan jaringan sementara dibatasi oleh energi dan sumber daya. Untuk mengatasi masalah ini, banyak peneliti mengusulkan berbagai algoritma dan kerangka kerja optimasi.

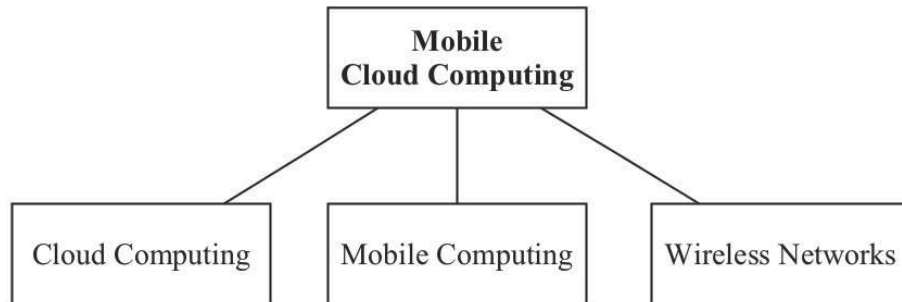
1.2.2 Sistem Tertanam Seluler

Sistem tertanam adalah sistem komputer dengan fungsi khusus, yang tertanam sebagai bagian dari perangkat lengkap termasuk perangkat keras dan bagian mekanis. Sistem tertanam mendorong revolusi informasi dengan penyebarannya. Sistem kecil ini dapat ditemukan di mana-mana, mulai dari elektronik komersial, seperti ponsel, kamera, sistem pemantauan kesehatan portabel, pengontrol mobil, robot, dan perangkat keamanan cerdas, hingga infrastruktur penting, seperti jaringan telekomunikasi, jaringan listrik, lembaga keuangan, dan pembangkit nuklir. Sistem tertanam yang semakin rumit membutuhkan otomatisasi desain yang ekstensif dan alat pengoptimalan. Sintesis tingkat arsitektur dengan pembuatan kode merupakan tahap penting untuk menghasilkan sistem tertanam yang memenuhi persyaratan ketat, seperti waktu, area, keandalan, dan konsumsi daya, sambil menjaga biaya produk tetap rendah dan siklus pengembangan singkat.

Perangkat seluler adalah sistem tertanam yang khas, yang mencakup prosesor seluler, penyimpanan, memori, grafik, sensor, kamera, baterai, dan chip lainnya untuk berbagai fungsi. Perangkat mobile adalah sintesis tingkat tinggi untuk sistem tertanam real-time menggunakan unit fungsional heterogen (FUs). Sebuah unit fungsional adalah bagian dari sistem tertanam, dan melakukan operasi dan perhitungan untuk tugas-tugas. Akibatnya, sangat penting untuk memilih jenis FU terbaik untuk berbagai tugas.

1.3 AWAN SELULER

Sumber daya yang terbatas adalah karakteristik penting lain dari perangkat mobile. Dengan perkembangan komputasi awan, komputasi awan mobile telah diperkenalkan ke publik. Komputasi awan seluler, seperti yang ditunjukkan pada Gambar 1.5, adalah kombinasi komputasi awan, komputasi seluler, dan jaringan nirkabel untuk menghadirkan sumber daya komputasi yang kaya ke sistem seluler. Secara umum, perangkat seluler dengan sumber daya terbatas dapat memanfaatkan sumber daya komputasi dari berbagai sumber daya cloud untuk meningkatkan kemampuan komputasi itu sendiri. Ada beberapa tantangan dalam komputasi awan seluler, seperti memindahkan proses komputasi dari perangkat seluler ke awan, latensi jaringan, pemrosesan konteks, manajemen energi, keamanan, dan privasi.



Gambar 1.5 Struktur utama komputasi awan bergerak.

Saat ini, beberapa penelitian dan pengembangan membahas pemuatan kode eksekusi, konektivitas tanpa batas, dan latensi jaringan; namun, upaya masih kurang di bidang lain.

Arsitektur. Arsitektur untuk lingkungan komputasi awan seluler yang heterogen sangat penting untuk melepaskan kekuatan komputasi seluler menuju komputasi di mana-mana tanpa batas.

Transmisi sadar energi. Memuat kode eksekutif ke cloud dapat sangat mengurangi beban dan waktu perangkat seluler lokal, tetapi meningkatkan transmisi antara perangkat seluler dan cloud. Protokol transmisi harus dirancang dengan hati-hati untuk menghemat energi.

Komputasi kontekstual. Komputasi yang sadar konteks dan sadar sosial adalah sifat yang tidak terpisahkan dari perangkat seluler. Bagaimana mencapai visi komputasi mobile di antara jaringan terkonvergensi heterogen di antara perangkat mobile merupakan kebutuhan penting.

Migrasi Mesin Virtual (VM) langsung. Mesin virtual adalah emulasi dari sistem komputer tertentu. Pemuatan sumber daya eksekutif melibatkan enkapsulasi aplikasi seluler dalam instans VM, dan migrasi di cloud adalah tugas yang menantang.

Keamanan dan Privasi. Karena kurangnya kepercayaan di cloud, banyak pengguna yang peduli dengan keamanan dan privasi informasi mereka. Sangat penting untuk meningkatkan keamanan dan privasi komputasi awan seluler.

1.3.1 Aplikasi Big Data dalam Sistem Seluler

Data besar atau Big Data adalah istilah yang mencakup semua untuk setiap kumpulan kumpulan data yang begitu besar atau kompleks sehingga menjadi sulit untuk memprosesnya menggunakan aplikasi pemrosesan data tradisional. Kumpulan data tumbuh dalam ukuran sebagian karena semakin banyak dikumpulkan oleh perangkat seluler. Ada 4,6 miliar pelanggan telepon seluler di seluruh dunia dan antara 1 miliar hingga 2 miliar orang mengakses Internet. Dengan miliaran perangkat seluler di dunia saat ini, komputasi seluler menjadi platform komputasi universal dunia. Perangkat seluler ini menghasilkan data dalam jumlah besar setiap hari. Maraknya big data menuntut kita untuk dapat mengakses sumber daya data kapan saja dan di mana saja tentang segala hal sehari-hari. Selain itu, jenis data ini sangat berharga dan menguntungkan jika digunakan dengan baik.

Namun, beberapa tantangan harus diatasi untuk memungkinkan analitik data besar. Lebih khusus lagi, alih-alih dibatasi untuk satu komputer, aplikasi di mana-mana harus dapat dijalankan pada ekosistem perangkat jaringan, yang masing-masing dapat

bergabung atau meninggalkan ruang bersama di mana-mana setiap saat. Selain itu, ada tugas analitik yang terlalu mahal secara komputasi untuk dilakukan pada ekosistem perangkat seluler. Juga, bagaimana kita dapat memanfaatkan kemampuan spesifik dari setiap perangkat, termasuk berbagai ukuran tampilan, modalitas input, dan sumber daya komputasi?

1.3.2 Keamanan Data dan Perlindungan Privasi di Sistem Seluler

Karena universalitas dan kekhususan sistem seluler ke sistem desktop, keamanan dalam sistem seluler jauh lebih rumit dan penting daripada di sistem desktop. Keamanan dalam sistem mobile dapat dipisahkan menjadi beberapa bagian. Ancaman pertama adalah malware (virus). Mobile malware adalah perangkat lunak berbahaya yang menargetkan perangkat seluler dan mengakibatkan runtuhnya sistem dan kehilangan atau kebocoran informasi. Menurut Laporan Ancaman McAfee Labs Juni 2014, malware mobile baru telah meningkat selama lima kuartal berturut-turut, dengan total pertumbuhan malware mobile sebesar 167 persen dalam beberapa tahun terakhir. Ancaman keamanan juga berkembang dengan 200 ancaman baru setiap menit. Selain 2,4 juta sampel baru malware seluler, 2013 juga membawa 1 juta sampel unik ransomware baru, 5,7 juta binari baru yang ditandatangani berbahaya, dan 2,2 juta sampel baru yang terkait dengan serangan Master Boot Record (MBR). Dua insentif yang paling sering adalah menyaring informasi pengguna dan panggilan atau SMS premium. Selain itu, ada beberapa insentif lain, seperti mengirim spam iklan, hal baru dan hiburan, dan meningkatkan kredensial pengguna.

Masalah penelitian lainnya adalah kerangka keamanan atau pendekatan untuk mendeteksi malware seluler. Ada beberapa pendekatan untuk memantau perangkat seluler dan mendeteksi malware seluler. Solusi berbasis tanda tangan adalah pendekatan yang digunakan untuk mendeteksi serangan, tetapi gagal total dalam mendeteksi penjahat dunia maya canggih yang menargetkan organisasi tertentu dengan eksploitasi yang disesuaikan untuk para korban tersebut. Dari perspektif proses, ketika datang untuk memvalidasi ancaman dan analisis akar penyebab selanjutnya, responden tingkat pertama harus mengirim semua data yang terlihat seperti kode berbahaya ke reverse engineer. Proses ini sering menyebabkan penundaan, karena tim malware ini biasanya kebanjiran. Sementara itu, dengan perkembangan teknologi, representasi yang efisien dari perilaku malware menggunakan pengamatan kunci sering mengungkapkan niat jahat bahkan ketika setiap tindakan saja mungkin tampak tidak berbahaya. Urutan logis dari tindakan aplikasi sering kali dari waktu ke waktu. Berdasarkan ide ini, peneliti menyajikan berbagai pendekatan untuk memantau dan mendeteksi perilaku jahat menggunakan analisis statis pada aliran data.

Masalah keamanan berikutnya adalah perilaku pengumpulan data yang berlebihan di aplikasi seluler. Sistem operasi ponsel saat ini hanya memberikan izin kasar yang menentukan apakah suatu aplikasi dapat mengakses informasi pribadi, sambil memberikan sedikit wawasan tentang ruang lingkup informasi pribadi yang digunakan. Sementara itu, hanya sedikit pengguna yang mengetahui informasi izin

selama penginstalan. Selain itu, beberapa pengguna memilih untuk berhenti menginstal atau mencopot pemasangan aplikasi ketika sistem memperingatkan mereka dan meminta izin, meskipun mereka tahu itu mungkin membawa beberapa masalah keamanan tersembunyi. Misalnya, kami mengambil data lokasi dan menganalisis status saat ini dan mendiskusikan risiko yang disebabkan oleh pengumpulan yang berlebihan.

Data lokasi merupakan data yang paling sering digunakan di smartphone. Ini dapat digunakan di aplikasi yang fungsi utamanya meliputi peta, pengaturan foto, rekomendasi belanja dan restoran, serta cuaca. Dari laporan Appthority, 50% aplikasi gratis iOS teratas dan 24% aplikasi berbayar iOS teratas melacak lokasi pengguna. Meskipun pengguna diperingatkan setiap kali aplikasi bermaksud untuk menangkap lokasi mereka, mereka biasanya memilih untuk mengizinkan izin untuk fungsi yang ditawarkan oleh aplikasi tersebut. Aplikasi yang mengumpulkan data lokasi secara berlebihan dapat dipisahkan menjadi dua jenis utama: layanan lokasi sebagai fungsi utama dan layanan lokasi sebagai fungsi tambahan. Jenis aplikasi pertama biasanya meminta izin kepada pengguna untuk informasi lokasi mereka, sedangkan jenis aplikasi lainnya dapat mengumpulkan informasi lokasi pengguna tanpa memperhatikan pengguna. Risiko pertama dan paling langsung adalah masalah keamanan fisik. Jejak pengguna mudah terekspos ke mereka yang memiliki data lokasi akurat dan real-time pengguna. Kebiasaan dan kebiasaan pengguna mudah disimpulkan dengan menggunakan metode penambangan data sederhana.

Selain itu, pemecahan masalah pengumpulan data juga merupakan masalah penelitian di aplikasi seluler. PiOS, disajikan oleh M. Egele et al., untuk mendeteksi kebocoran privasi di aplikasi iOS, menggunakan analisis statis untuk mendeteksi aliran data sensitif untuk mencapai tujuan mendeteksi kebocoran privasi dalam aplikasi di iOS. Berbagi tujuan yang sama dengan PiOS, TaintDroid, adalah taint dinamis di seluruh sistem yang melacak berbagai sumber data sensitif. Strategi utama TaintDroid adalah analisis real-time dengan memanfaatkan lingkungan eksekusi virtual Android. Model aman lainnya melalui validasi otomatis menggunakan infrastruktur cloud komoditas untuk meniru ponsel cerdas guna melacak arus dan tindakan informasi secara dinamis.

Model ini secara otomatis mendeteksi perilaku berbahaya dan penyalahgunaan data sensitif melalui analisis lebih lanjut dari grafik ketergantungan berdasarkan arus informasi yang dilacak dan tindakan. Pendekatan atau teknik yang disebutkan di atas hanya berfokus pada pemantauan dan pendeteksian aplikasi. Prasyaratnya adalah aplikasi sudah mendapatkan izin dari pengguna. Namun, solusi ini hanya menyediakan metode pemantauan dan pendeteksian perilaku pengumpulan data yang berlebihan. Pendekatan ini menyerahkan operasi perbaikan kepada pengguna, seperti menonaktifkan izin aplikasi atau mencopot pemasangan aplikasi tersebut. Pengguna harus secara manual menonaktifkan izin aplikasi ini yang mengumpulkan data pengguna secara berlebihan atau mencopot pemasangannya. Lebih jauh lagi, menjalankan pendekatan atau alat ini menambah konsumsi energi, yang sangat berharga untuk ponsel pintar dengan sumber daya terbatas. Akibatnya, metode aktif

untuk menghindari perilaku pengumpulan data di aplikasi seluler merupakan tantangan penting yang perlu dipecahkan.

1.3.3 Konsep Aplikasi Seluler

Aplikasi seluler awalnya dikembangkan untuk menawarkan produktivitas umum dan pencarian informasi, termasuk email, kalender, kontak, dan informasi cuaca. Namun, dengan peningkatan pesat kebutuhan publik, aplikasi seluler berkembang menjadi banyak kategori lain, seperti game, musik, keuangan, dan berita. Banyak orang membedakan aplikasi dari aplikasi dalam perspektif bentuk perangkat. Mereka mengira aplikasi digunakan di desktop atau laptop, sedangkan aplikasi digunakan di ponsel atau tablet. Namun demikian, pandangan sederhana ini terlalu sempit dan tidak lagi menjadi konsensus, karena aplikasi dapat digunakan di desktop, dan sebaliknya, aplikasi dapat berjalan di ponsel. Di Gartner Portals, Content and Collaboration Summit 2013, banyak pakar dan pengembang berpartisipasi dalam diskusi meja bundar berjudul “Mengapa Aplikasi bukan Aplikasi”. Mereka mengusulkan bahwa perbedaan antara aplikasi dan aplikasi bukan tentang mekanisme pengiriman dan mencapai konsensus bahwa:

App = perangkat lunak yang dirancang untuk satu tujuan dan melakukan satu fungsi.

Aplikasi = perangkat lunak yang dirancang untuk melakukan berbagai fungsi.

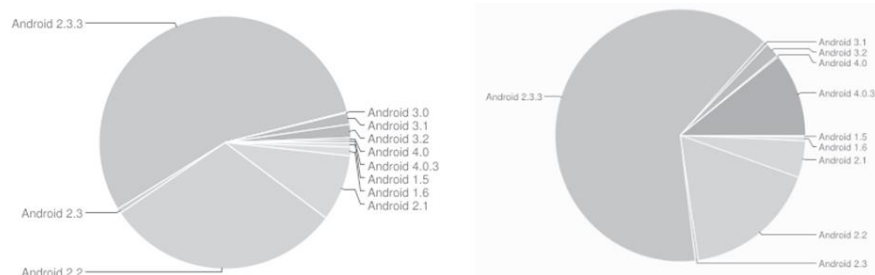
Dari sudut pandang pengguna, mereka tidak peduli apakah itu aplikasi atau aplikasi menurut definisi, dan mereka hanya ingin menyelesaikan tugas mereka dengan mudah. Sementara itu, dari sudut pandang pengembang, pertanyaan yang harus mereka jawab bukanlah apakah mereka harus membangun aplikasi atau aplikasi, tetapi bagaimana mereka dapat menggabungkan yang terbaik dari keduanya menjadi sesuatu yang disukai pengguna.

1.3.4 Pengenalan Singkat Android dan Kerangkanya

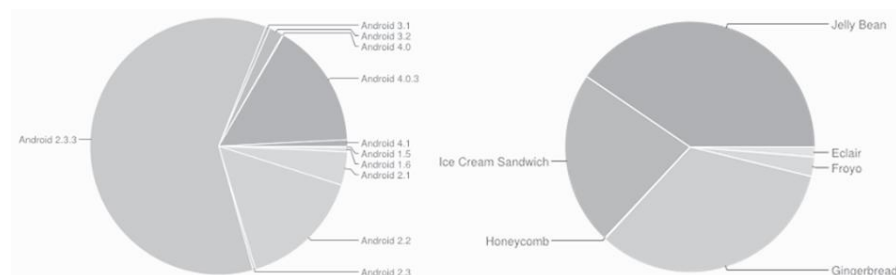
1.3.4.1 Sejarah Singkat Android

Android didirikan di Palo Alto, California, pada Oktober 2003 oleh Andy Rubin, Rich Miner, Nick Sears, dan Chris White dalam upaya mengembangkan perangkat seluler yang lebih cerdas yang lebih mengetahui lokasi dan preferensi pemiliknya. Kemudian ke Google mengakuisisi Android Inc. dan karyawan kunci, termasuk Rubin, Miner, dan White, pada 17 Agustus 2005. Di Google, tim yang dipimpin oleh Rubin, mengembangkan platform perangkat seluler yang didukung oleh kernel Linux. Google telah menyusun serangkaian komponen perangkat keras dan mitra perangkat lunak dan memberi isyarat kepada operator bahwa itu terbuka untuk berbagai tingkat kerjasama di pihak mereka. Pada tanggal 5 November 2007, Open Handset Alliance meluncurkan dirinya dengan tujuan untuk mengembangkan standar terbuka untuk perangkat mobile. Aliansi ini mencakup perusahaan teknologi, seperti Google, produsen perangkat seperti HTC, operator nirkabel seperti T-Mobile, dan pembuat chipset seperti Qualcomm. Kemudian, pada 22 Oktober 2008, smartphone pertama yang tersedia secara komersial yang menjalankan Android keluar dengan nama fantasi: HTC Dream. Sejak 2008, Android telah

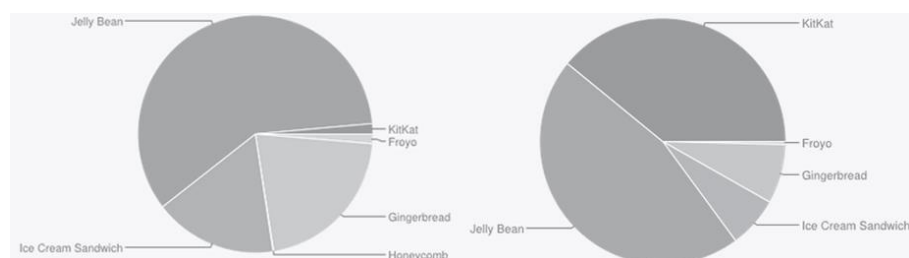
melihat banyak pembaruan yang secara bertahap meningkatkan sistem operasi, menambahkan fitur baru dan memperbaiki bug di rilis sebelumnya. Ada beberapa milestone Android SDK, seperti Android SDK 2.0 (Eclair) di tahun 2009, Android SDK 3.0 (Honeycomb) hanya untuk tablet di tahun 2011, Android SDK 4.0 (Ice Cream Sandwich) di tahun 2011, Android 4.1 hingga 4.3 (Jelly Bean) pada tahun 2012, Android SDK 4.4 (KitKat) pada tahun 2013, dan Android SDK 5.0 (Lollipop) pada tahun 2014.



Gambar 1.6 Distribusi perangkat Android pada bulan Januari dan Juli 2012.



Gambar 1.7 Distribusi perangkat Android pada bulan Agustus 2012 dan Agustus 2013.



Gambar 1.8 Distribusi perangkat Android pada Januari 2014 dan Januari 2015

1.3.4.2 Distribusi Perangkat Android

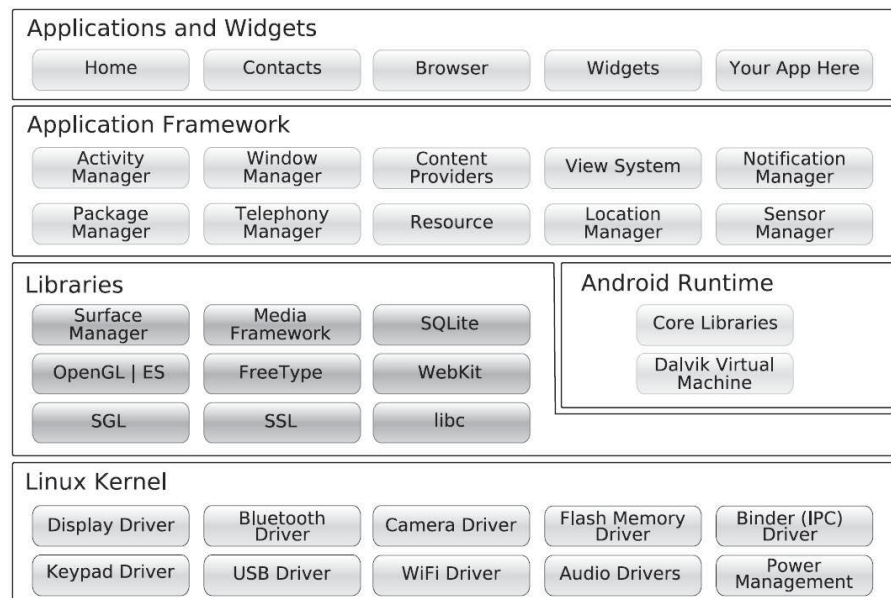
Gambar 1.6 menunjukkan distribusi perangkat Android pada tahun 2012. Kita dapat melihat bahwa Android 2.3.3 dan 2.2 mendominasi lebih dari setengah pasar. Meskipun demikian, pada paruh kedua tahun 2012, Android 4.0.3 menjadi semakin populer. Pada bulan Agustus 2013, Android 4.0 dan 4.1, masing-masing bernama Ice Cream Sandwich dan Jelly Bean, melampaui Android 2.0s dan mendominasi pasar Android, seperti yang ditunjukkan pada Gambar 1.7. Pada Januari 2014, Android 4.1 hingga 4.3 masih mendominasi pasar Android. Namun, setelah satu tahun, Android 4.4, bernama KitKat,

dengan cepat menduduki 39,1% dari seluruh pasar, seperti yang ditunjukkan pada Gambar 1.8.

1.3.4.3 Android SDK

Android SDK adalah open-source dan banyak digunakan, yang menjadikannya pilihan terbaik untuk pengajaran dan pembelajaran pengembangan seluler. Android adalah kumpulan perangkat lunak untuk perangkat seluler, dan mencakup sistem operasi seluler, middleware, dan beberapa aplikasi utama. Seperti ditunjukkan pada Gambar 1.9, ada kernel Linux, perpustakaan, kerangka aplikasi, dan aplikasi dan widget, dari bawah ke atas. Kami akan memperkenalkan mereka satu per satu.

Kernel Linux digunakan untuk menyediakan beberapa layanan sistem inti, seperti keamanan, manajemen memori, manajemen proses, manajemen daya, dan driver perangkat keras. Layanan ini tidak dapat dipanggil oleh program Android secara langsung dan transparan bagi pengguna. Lapisan berikutnya di atas kernel adalah pustaka asli, yang semuanya ditulis dalam C atau C++. Pustaka ini dikompilasi untuk arsitektur perangkat keras tertentu yang digunakan oleh perangkat seluler. Mereka bertanggung jawab untuk menangani penyimpanan data terstruktur, grafik, audio, video, dan jaringan, yang hanya dapat dipanggil oleh program tingkat yang lebih tinggi. Sementara itu, waktu proses Android juga berada di atas kernel, termasuk mesin virtual Dalvik dan perpustakaan inti Java.



Gambar 1.9 Arsitektur sistem Android.

Apa itu Dalvik? Dalvik adalah proses mesin virtual dalam sistem operasi Google Android, yang secara khusus mengeksekusi aplikasi yang ditulis untuk Android. Program ditulis dalam Java dan dikompilasi ke bytecode untuk mesin virtual Java, yang kemudian diterjemahkan ke bytecode Dalvik dan disimpan dalam file .dex dan .odex. Format executable

Dalvik yang ringkas dirancang untuk sistem dengan sumber daya terbatas. Lapisan kerangka aplikasi menyediakan blok bangunan tingkat tinggi yang digunakan untuk membuat aplikasi. Itu sudah diinstal sebelumnya dengan Android, tetapi dapat diperpanjang dengan komponennya sendiri sesuai kebutuhan. Kami akan memperkenalkan beberapa blok bangunan dasar dan penting Android.

Aktivitas. Aktivitas adalah layar antarmuka pengguna. Aktivitas tunggal mendefinisikan satu layar dengan antarmuka pengguna, dan mendefinisikan metode siklus hidup sederhana seperti `onCreate`, `onResume`, dan `onPause` untuk menangani interupsi. Selanjutnya, aplikasi dapat mendefinisikan satu atau lebih aktivitas untuk menangani fase program yang berbeda.

Intent. Intent adalah mekanisme untuk menggambarkan tindakan tertentu, seperti "pilih foto", atau "telepon ke rumah". Di Android, semuanya berjalan melalui maksud, dan pengembang, memiliki banyak peluang untuk mengganti atau menggunakan kembali komponen. Intent bisa implisit atau eksplisit. Maksud eksplisit bisa untuk memanggil layar lain saat tombol ditekan pada Aktivitas dalam konteks. Intent implisit adalah saat Anda membuat intent dan menyerahkannya ke sistem untuk menanganinya.

Layanan. Layanan adalah tugas yang berjalan di latar belakang tanpa interaksi langsung pengguna. Faktanya, ia melakukan sebagian besar pemrosesan untuk suatu aplikasi. Pengembang dapat membuat sub-kelas Layanan kelas untuk menulis layanan kustom mereka sendiri.

Penyedia konten. Penyedia Konten adalah kumpulan data yang dibungkus dalam Antarmuka Pemrograman Aplikasi (API) khusus untuk membaca dan menulisnya. Ini adalah cara terbaik untuk berbagi data global antar aplikasi. Penyedia konten menyediakan antarmuka tunggal yang seragam untuk konten dan data dan menyediakan antarmuka yang konsisten untuk mengambil/menyimpan data melalui model RESTful yang mendukung operasi buat, baca, perbarui, dan hapus (CRUD).

Emulator Android, seperti yang ditunjukkan pada Gambar 1.10, disebut Android Virtual Device (AVD), penting untuk menguji aplikasi Android tetapi bukan pengganti perangkat nyata. AVD memiliki resolusi yang dapat dikonfigurasi, RAM, kartu SD, skin, dan perangkat keras lainnya. Jika Anda telah menginstal Android SDK, AVD Manager dapat mengizinkan Anda membuat AVD yang menargetkan level API Android apa pun.

Emulator Android memiliki fungsi dasar berikut:

- Keyboard komputer host berfungsi sebagai keyboard perangkat.
- Mouse host bertindak sebagai jari.
- Menghubungkan ke Internet menggunakan koneksi Internet host.
- Tombol: Beranda, Menu, Kembali, Pencarian, Volume naik dan turun.
- Ctrl-F11 beralih lanskap ke potret.
- Alt-Enter beralih mode layar penuh.

Namun, emulator memiliki beberapa keterbatasan. Mereka tidak mendukung untuk:

- Menempatkan atau menerima panggilan telepon yang sebenarnya.
- Koneksi USB dan Bluetooth.

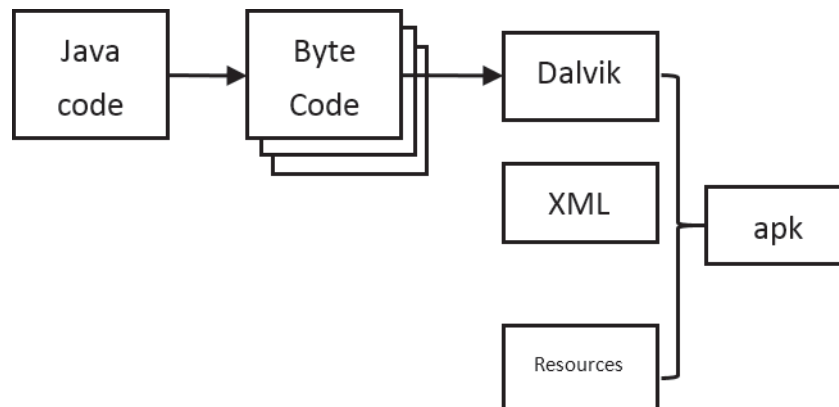
- Pengambilan kamera atau video sebagai input.
- Headphone yang terpasang ke perangkat.
- Menentukan status terhubung.
- Menentukan tingkat pengisian daya baterai dan status pengisian daya AC.
- Menentukan memasukkan atau mengeluarkan kartu SD. Kartu SD adalah kartu memori nonvolatile yang digunakan secara luas di perangkat portabel.
- Simulasi akselerometer.



Gambar 1.10 Emulator Android.

Kemudian kami akan memperkenalkan proses pembuatan aplikasi Android. Pada Gambar 1.11, aplikasi android ditulis dalam Java dan menghasilkan file .java. Kemudian kompiler javac .java membaca file sumber dan mengubah kode java menjadi kode byte. Kemudian Dalvik bertanggung jawab untuk menangani kode byte yang digabungkan dengan kode byte lain untuk file .class lainnya, dan menghasilkan class.dex. Terakhir, class.dex, resources, dan AndroidManifest.xml bekerja sama dan menghasilkan file .apk, yang merupakan aplikasi Android yang dapat dijalankan.

Setiap aplikasi Android harus memiliki file AndroidManifest.xml di direktori root-nya. Manifes menyajikan informasi penting tentang aplikasi ke sistem Android, informasi yang harus dimiliki sistem sebelum dapat menjalankan kode aplikasi apa pun. File AndroidManifest.xml menamai paket Java untuk aplikasi dan menjelaskan komponen aplikasi, termasuk aktivitas, layanan, penerima siaran, dan penyedia konten yang terdiri dari aplikasi. File juga menamai kelas yang mengimplementasikan setiap komponen dan mempublikasikan kemampuannya. Deklarasi ini membuat sistem Android mengetahui apa saja komponen tersebut dan dalam kondisi apa mereka dapat diluncurkan.



Gambar 1.11 Proses pembuatan aplikasi Android.

Selanjutnya, file `AndroidManifest.xml` menentukan, proses mana yang akan menghosting komponen aplikasi, dan mendeklarasikan izin mana yang harus dimiliki aplikasi untuk mengakses bagian API yang dilindungi dan berinteraksi dengan aplikasi lain. File juga mendeklarasikan izin yang harus dimiliki orang lain untuk berinteraksi dengan komponen aplikasi dan mendaftarkan kelas instrumentasi yang menyediakan profil dan informasi lain saat aplikasi berjalan. Deklarasi ini hadir dalam manifest hanya saat aplikasi diterbitkan. Ini mendeklarasikan level minimum Android API yang diperlukan aplikasi, dan mencantumkan pustaka yang harus ditautkan ke aplikasi. Dalam bab berikutnya, kita akan membahas arsitektur sistem tertanam mobile.

1.4 LATIHAN

Dasar

1. Komponen apa yang termasuk dalam sistem seluler?
2. Apakah perangkat seluler hanya berarti smartphone?
3. Berapa banyak sistem operasi seluler yang berjalan di perangkat seluler? Apakah mereka?
4. Apa dasar dari jaringan seluler?
5. Bagaimana perangkat seluler terhubung ke Internet di bawah lingkungan jaringan seluler?
6. Apa itu Wi-Fi?
7. Bagaimana perangkat seluler terhubung ke Internet di bawah lingkungan jaringan Wi-Fi?
8. Apa hubungan antara jarak dan intensitas sinyal di bawah lingkungan jaringan Wi-Fi?
9. Dapatkah perangkat Android menjalankan aplikasi iOS?
10. Apa perbedaan antara antarmuka perangkat seluler dan desktop?
11. Apa itu telehealth?
12. Apa tantangan terbesar dalam langkah pengumpulan dan transmisi data dalam sistem telehealth?
13. Apa yang dimaksud dengan masalah optimasi? (pertanyaan opsional)
14. Berapa banyak kendala yang ada dalam sistem mobile? Apakah mereka?

15. Apa itu sistem tertanam?
16. Apa itu sistem tertanam seluler?
17. Apa manfaat terbesar mobile cloud untuk sistem mobile?
18. Apa itu big data dalam sistem seluler?
19. Apa itu malware di perangkat seluler?
20. Apa perilaku pengumpulan data di aplikasi seluler?
21. Apa itu Android?
22. Apa itu Open Handset Alliance?
23. Kapan Android 2.0 dirilis?
24. Apa nama perangkat Android pertama?
25. Versi Android mana yang paling banyak digunakan pada Januari 2014?
26. Apa saja fungsi utama yang disediakan Kernel Linux di Android?
27. Apa itu Aktivitas di Android?
28. Apa itu Niat?
29. Apa itu Intent implisit?
30. Apa itu Intent eksplisit?
31. Apa itu Layanan?
32. Apa yang dilakukan penyedia Konten?
33. Bagaimana proses pembuatan aplikasi Android dari kode Java ke file .apk?

Lanjutan

34. Mengapa konsumsi energi lebih penting untuk perangkat seluler daripada desktop?
35. Apa perbedaan antara aplikasi dan aplikasi?
36. Apa itu Dalvik? Apa perbedaan antara Dalvik dan Java Virtual Machine (JVM) tradisional?

BAB 2

MULAI CEPAT DI ANDROID

Sebelum kita melompat ke dunia android, mari kita lihat ulasan singkat tentang instalasi Android, pembuatan proyek, dan eksekusi aplikasi. Perkenalkan proses penginstalan Android dan pembuatan proyek Android dalam bab ini. Isi utama meliputi:

1. Menginstal Java
2. Memasang lingkungan pengembangan terintegrasi
3. Memasang Android SDK
4. Membuat proyek aplikasi Android
5. Membuat Perangkat Virtual Android
6. Menjalankan aplikasi Android di emulator
7. Menjalankan aplikasi Android di ponsel asli

2.1 MENGINSTAL JAVA

Android Software Development Kit (SDK) dapat bekerja pada sistem operasi apa pun, seperti Windows, Linux, dan Mac OS X. Sebelum memulai penginstalan Android dan program pengkodean, kita perlu menginstal Java. Semua alat pengembangan Android memerlukan Java, dan program akan menggunakan bahasa Java. Dari versi terbaru situs Pengembang Android, kami menyarankan Java 7 atau 8 adalah pilihan terbaik. Kami merekomendasikan untuk mendapatkan Java runtime environment (JRE) 8 dari <http://java.com/en/download/manual.jsp>. Untuk pengguna Windows, ada dua macam versi yang ditawarkan, yaitu 32-bit dan 64-bit. Anda dapat memilih unduhan 32-bit untuk digunakan dengan browser 32-bit, dan memilih unduhan 64-bit untuk digunakan dengan browser 64-bit. Untuk pengguna Mac OS, hanya ada satu pilihan, yaitu membutuhkan Mac OS X versi 10.7.3 ke atas. Untuk pengguna Linux, ada empat pilihan, dan pengguna dapat mengunduh salah satunya berdasarkan sistem operasi pengguna. Tidak cukup hanya memiliki JRE, dan Anda memerlukan kit pengembangan lengkap. Sebaiknya unduh Java Development Kit (JDK) 8 dari <http://www.Oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>. Untuk memverifikasi bahwa Anda memiliki versi yang benar, buka jendela shell atau terminal Anda dan ketik "java ?version". Hasilnya harus seperti yang ditunjukkan pada Gambar 2.1.

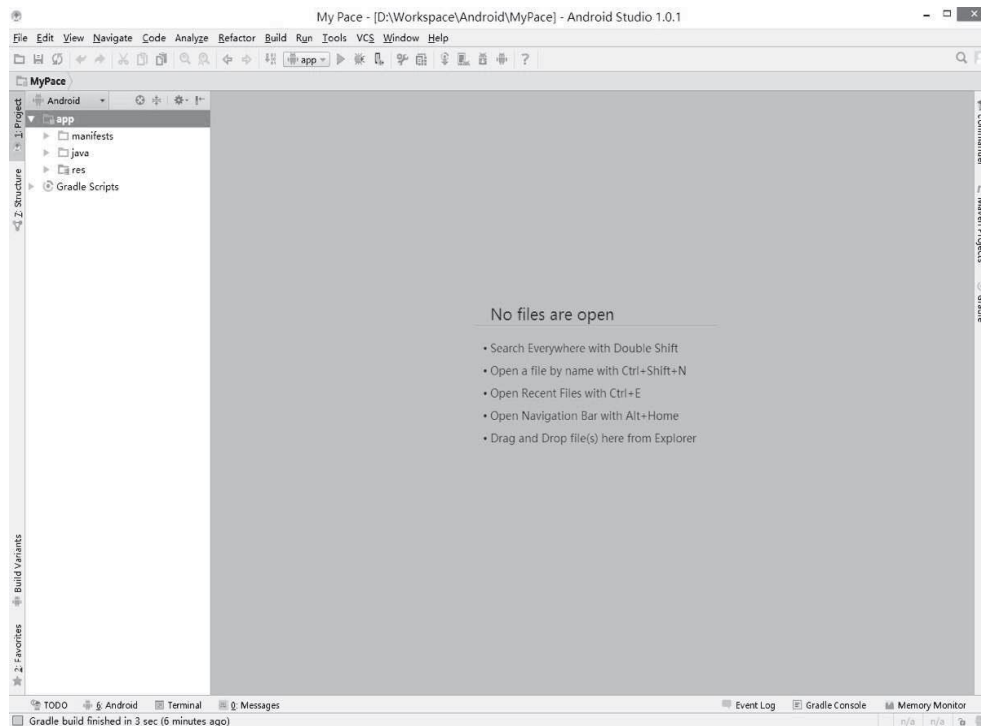
```
C:\>java -version
java version "1.8.0_25"
Java(TM) SE Runtime Environment (build 1.8.0_25-b18)
Java HotSpot(TM) 64-Bit Server VM (build 25.25-b02, mixed mode)
```

Gambar 2.1 Verifikasi versi Java.

2.2 MENGINSTAL LINGKUNGAN PEMBANGUNAN TERPADU

Lingkungan pengembangan Java direkomendasikan untuk membuat pemrograman Android lebih mudah. Ada banyak Integrate Development Environments (IDE) opsional, tetapi

kami hanya memperkenalkan yang paling banyak digunakan, yaitu Google Android Studio. Android Studio adalah IDE resmi untuk pengembangan aplikasi Android. Anda dapat mengunduhnya dari <http://developer.android.com/sdk/index.html>. Setelah mengunduh dan menginstal Android Studio, Anda dapat melihat gambar layar yang serupa, seperti yang ditunjukkan pada 2.2, saat Anda membukanya.

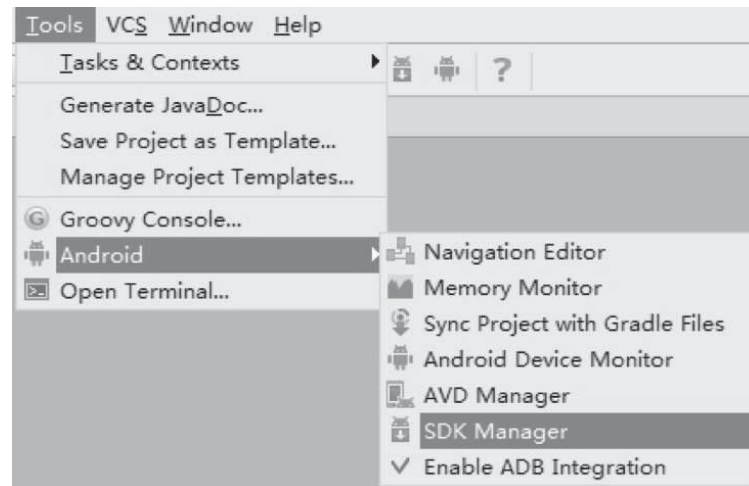


Gambar 2.2 Antarmuka kosong Android Studio.

2.3 MENGINSTAL ANDROID SDK

SDK Android mencakup seperangkat alat pengembangan yang komprehensif. Alat-alat ini termasuk debugger, perpustakaan, emulator handset, dokumentasi, kode sampel, dan tutorial. Menggunakan IDE yang diinstal, Android SDK dapat diunduh dan diinstal dengan nyaman. Di Android Studio, di bagian atas layar, pilih menu Tools, lalu Android, lalu SDK Manager (Tools → Android → SDK Manager), seperti yang ditunjukkan pada Gambar 2.3. Kemudian kita bisa melihat antarmuka Android SDK Manager, seperti Gambar 2.4.

Instal Android SDK Tools, Android SDK Platform-tools, setidaknya satu Android SDK Build-tools, dan setidaknya satu Android API, seperti yang ditunjukkan pada Gambar 2.5. API adalah seperangkat rutinitas, protokol, dan alat untuk membangun aplikasi perangkat lunak. Android 5.0.1 (API 21) adalah versi terbaru dari Android SDK. Kami menyarankan untuk menginstal Documentation for Android SDK, SDK Platform, ARM EABI v7a System Image, dan Google API. Dokumentasi untuk Android SDK dapat membantu memecahkan masalah pemrograman. Citra sistem ARM EABI v7a adalah citra sistem operasi seluler virtual yang berjalan pada perangkat virtual.

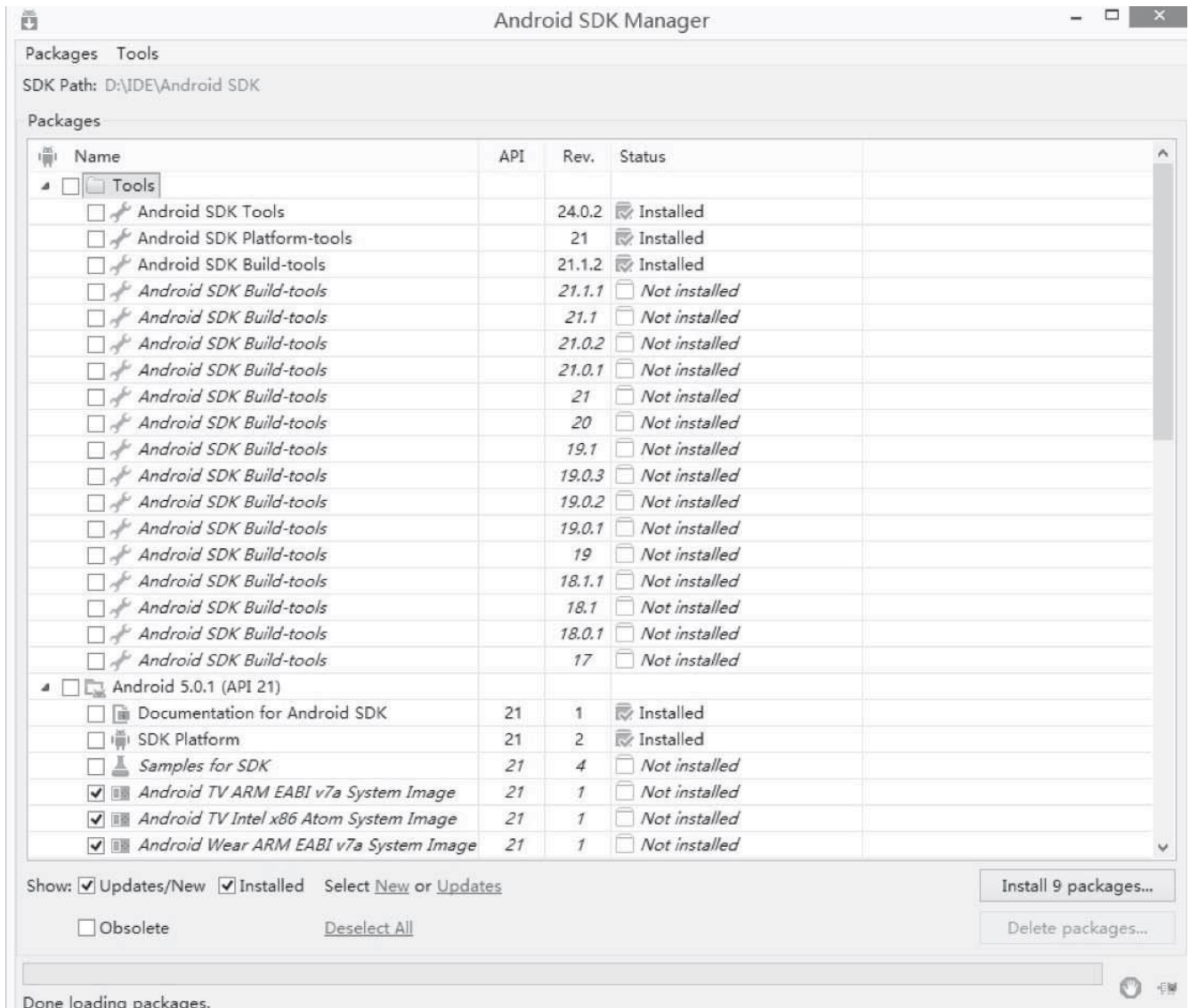


Gambar 2.3 Android SDK Manager di Android Studio.

2.4 MENCIPTAKAN APLIKASI ANDROID

Setelah menginstal Android SDK, kita dapat membuat Aplikasi Android pertama kita. Di pojok kiri atas Android Studio, pilih File, lalu New Project (File → New Project). Anda akan melihat dialog "Buat Proyek Baru". Pada langkah pertama membuat aplikasi Android baru, ketik nama aplikasi, seperti "Aplikasi Saya", seperti yang ditunjukkan pada Gambar 2.5. Anda dapat mengetikkan domain perusahaan, seperti "my.android.example.com". Selanjutnya, Anda dapat memilih direktori untuk menyimpan proyek Android Anda. Pada langkah kedua membuat aplikasi Android baru, Anda dapat memilih jenis perangkat yang menjalankan aplikasi Anda. Anda dapat memilih lebih dari satu perangkat, seperti ponsel dan tablet, TV, dan Wear. Pada aplikasi Android ini, pilih saja "Phone and Tablet", seperti terlihat pada Gambar 2.6.

Pada langkah ketiga membuat aplikasi Android baru, Anda dapat menambahkan aktivitas ke aplikasi Android Anda, dan Anda memiliki banyak pilihan, seperti aktivitas kosong, aktivitas kosong dengan fragmen, aktivitas layar penuh, aktivitas peta Google, aktivitas login, aktivitas laci navigasi, aktivitas pengaturan, dan aktivitas tab, seperti yang ditunjukkan pada Gambar 2.7. Di Android Studio versi terbaru, fragmen diintegrasikan ke dalam aktivitas.

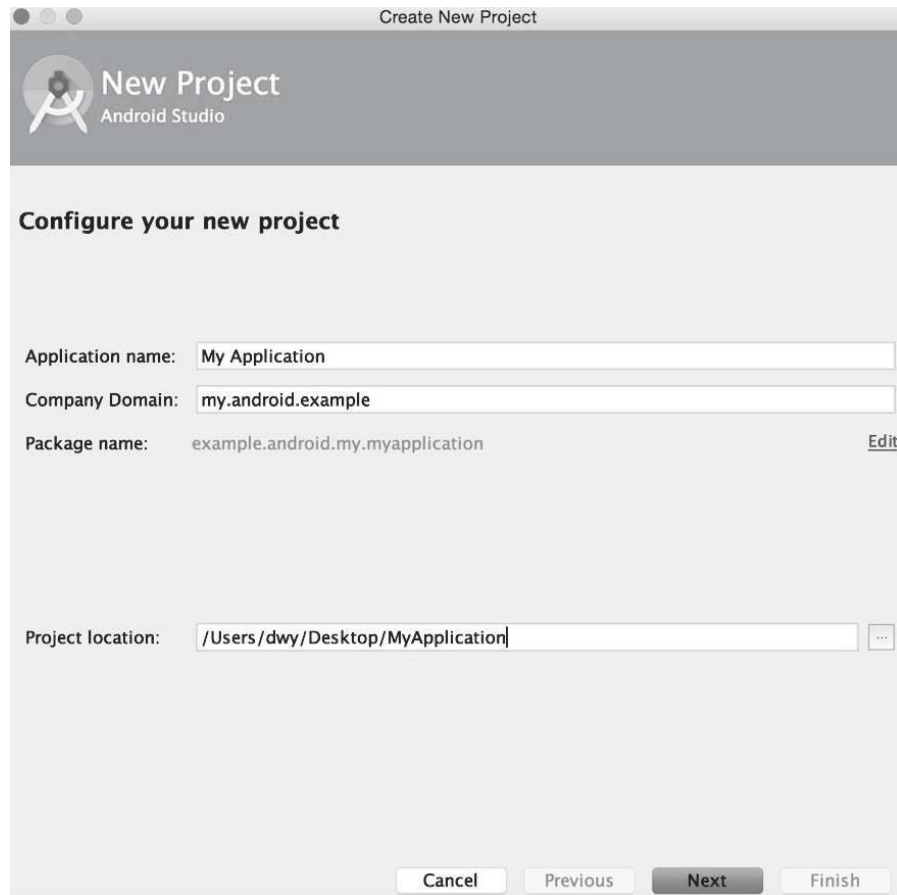


Gambar 2.4 Detail Android SDK Manager di Android Studio.

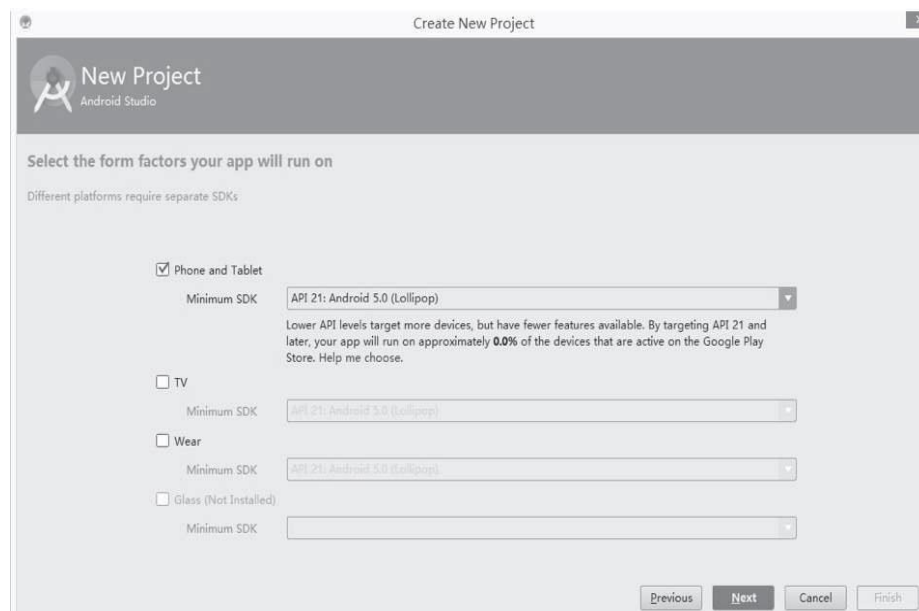
Pada langkah terakhir membuat aplikasi Android baru, Anda dapat mengubah nama aktivitas yang ditambahkan pada langkah ketiga, seperti yang ditunjukkan pada Gambar 2.8. Kemudian klik "finish", antarmuka Android Studio akan seperti Gambar 2.9.

2.5 PERANGKAT VIRTUAL ANDROID

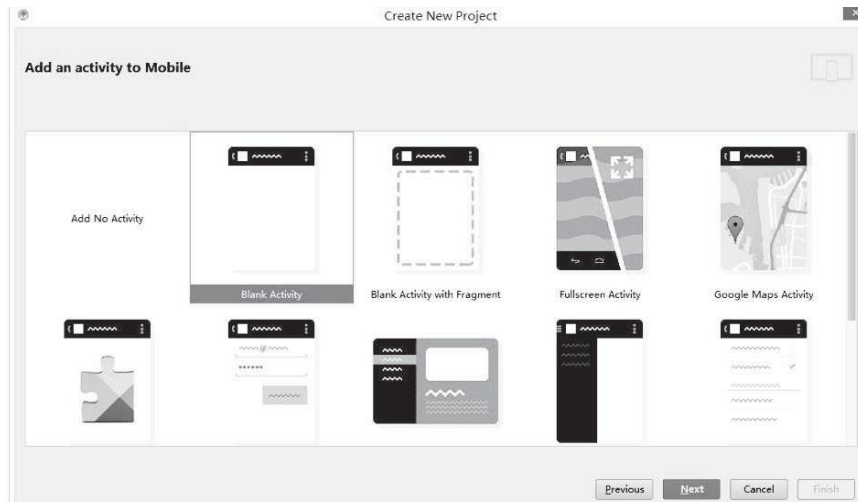
Setelah membuat aplikasi Android pertama, kita perlu membuat Android Virtual Device (AVD) untuk menjalankannya. Pertama, di bagian atas antarmuka, pilih Alat, lalu Android, lalu AVD Manager (Alat → Android → Manajer AVD). AVD Manager mirip dengan Gambar 2.10. Klik "Buat perangkat virtual"; antarmuka akan mirip dengan Gambar. 2.11. Pilih Telepon dalam daftar kategori, dan Nexus S sebagai perangkat. Kemudian klik "Selanjutnya." Pada langkah kedua membuat AVD di Android Studio, Anda dapat memilih versi Android SDK yang ingin Anda gunakan, seperti yang ditunjukkan pada Gambar 2.12. Kemudian klik "Selanjutnya."



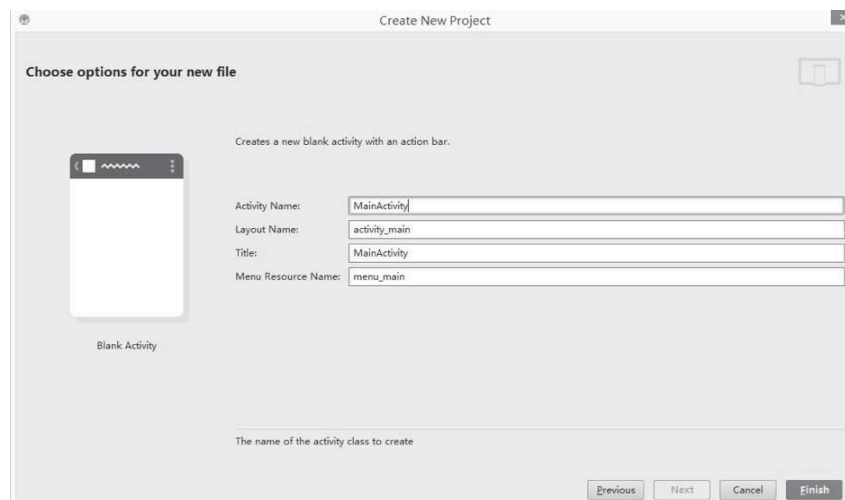
Gambar 2.5 Langkah Pertama membuat Aplikasi Android di Android Studio.



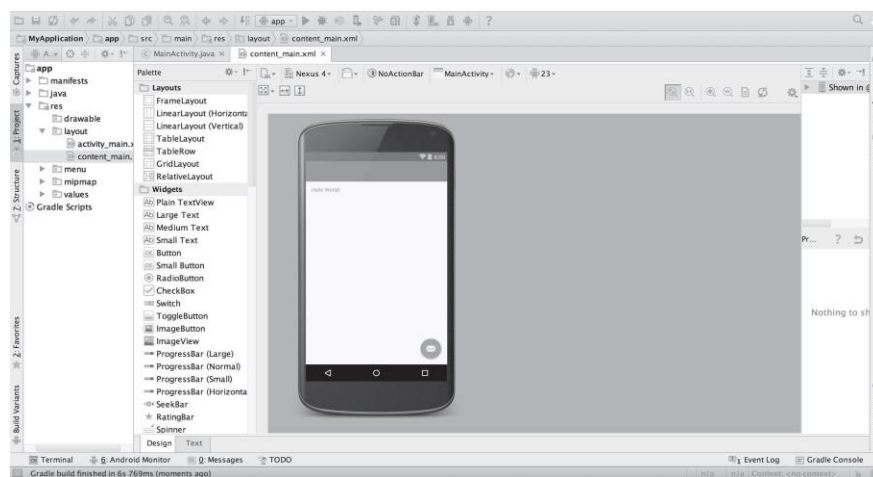
Gambar 2.6 Langkah pertama membuat aplikasi Android di Android Studio.



Gambar 2.7 Langkah kedua membuat aplikasi Android di Android Studio.



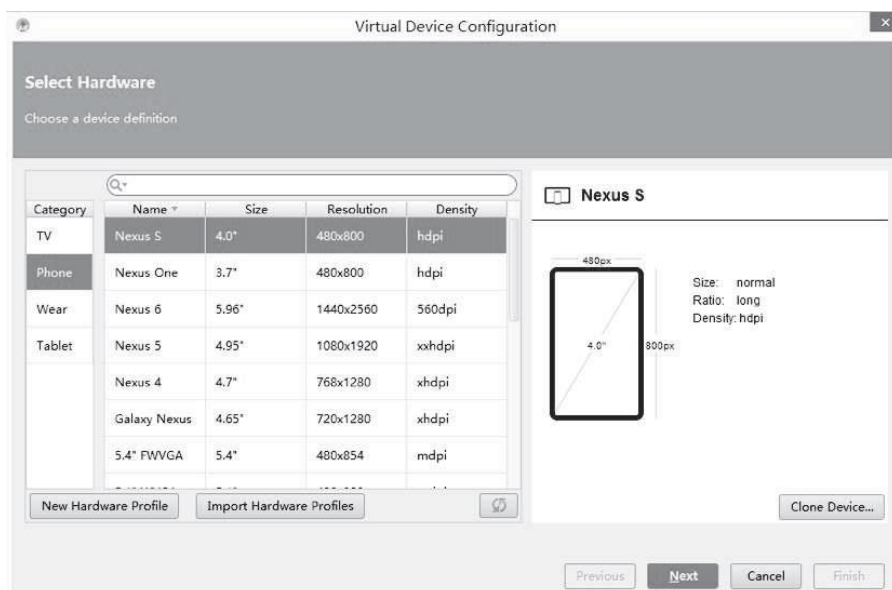
Gambar 2.8 Langkah terakhir membuat aplikasi Android di Android Studio.



Gambar 2.9 Antarmuka Android Studio dengan proyek Android baru.

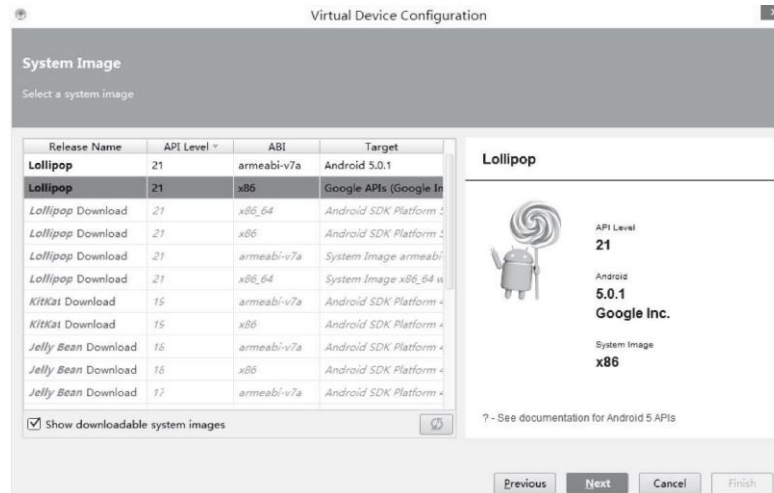


Gambar 2.10 Pengelola Perangkat Virtual Android di Android Studio.

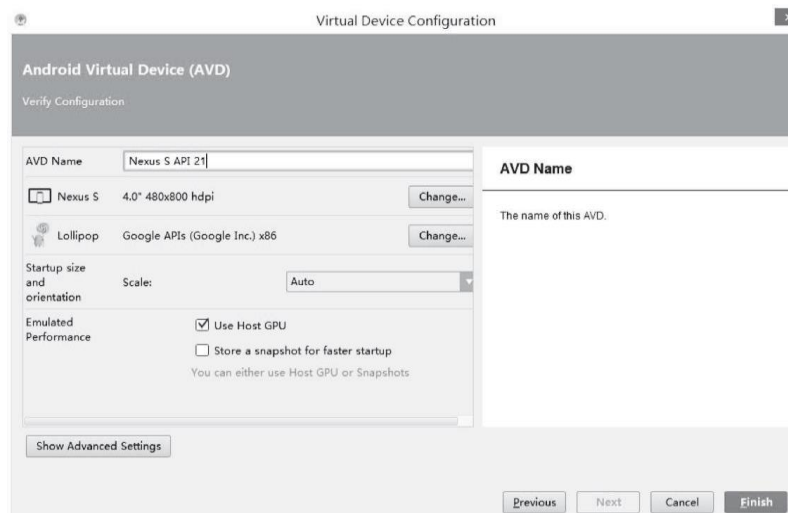


Gambar 2.11 Langkah Pertama Membuat Perangkat Virtual Android di Android Studio -1.

Pada langkah terakhir membuat AVD di Android Studio, Anda dapat mengubah nama AVD yang ingin Anda buat, seperti yang ditunjukkan pada Gambar 2.13. Kemudian klik "Selesai". AVD dibuat, seperti yang ditunjukkan pada Gambar 2.14. Kemudian klik panah hijau di sisi kanan untuk memulai perangkat virtual ini.



Gambar 2.12 Langkah Kedua Membuat Android Virtual Device di Android Studio -2.



Gambar 2.13 Langkah Terakhir Membuat Android Virtual Device di Android Studio.



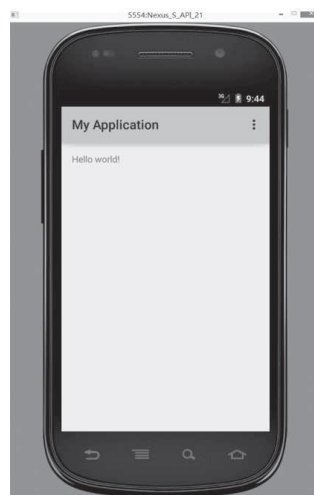
Gambar 2.14 Perangkat Virtual Baru di AVD Manager di Android Studio.

Setelah menunggu beberapa saat, perangkat virtual dimulai, seperti yang ditunjukkan pada Gambar 2.15. Kemudian jalankan aplikasi Android Anda di perangkat virtual ini. Pilih

aplikasi yang dibuat sebelumnya, lalu klik panah hijau untuk menjalankannya. Akhirnya, aplikasi Android berjalan pada perangkat virtual, seperti yang ditunjukkan pada Gambar 2.16.



Gambar 2.15 Perangkat Virtual Android di Android Studio



Gambar 2.16 Aplikasi Android yang Berjalan pada Virtual Device Android di Android Studio

2.6 LATIHAN

1. Sistem operasi apa yang dapat digunakan Android?
2. JDK edisi terendah apa yang dibutuhkan untuk mendukung Android?
3. Bagaimana cara memverifikasi versi Java di komputer Anda?
4. Apa yang dibangun Android?
5. Mengapa Android menggunakan Linux?
6. Bisakah program Anda melakukan panggilan Linux secara langsung?
7. Dalam bahasa apa perpustakaan asli ditulis?
8. Jenis musik apa yang bisa dimainkan Android?
9. Mengapa Android menggunakan SQLite?
10. Apa yang ada di atas kernel Android?
11. Jenis file apa yang dijalankan oleh Dalvik Virtual Machine?

12. Apa perbedaan antara file .dex dan .class dan . standar?
 1. file .jar?
13. Apakah Android menggunakan library Java Standard Edition atau Java Mobile Edition?
14. Lapisan mana yang berada di atas library dan runtime asli?
15. Apa fungsi utama dari Activity Manager?
16. Apa fungsi utama Content Provider?
17. Lapisan manakah yang tertinggi dalam arsitektur Android?
18. Apa itu aplikasi di Android?
19. Apa itu Widget?
20. Apakah Widget dioperasikan hanya dalam kotak kecil pada aplikasi layar Beranda?
21. Aplikasi mana yang sudah dikemas dalam ponsel Android baru?
22. Bisakah banyak aplikasi berjalan dan terlihat sekaligus di berbagai jendela di Android?
23. Aplikasi mana yang pertama kali digunakan ketika pengguna menghidupkan ponselnya?
24. Di mana program dan layar direkam?
25. Bisakah aplikasi tetap hidup meskipun prosesnya telah dimatikan?
26. Status apa yang bisa menjadi program Android?
27. Metode mana yang dipanggil saat aktivitas pertama kali dijalankan?
28. Metode mana yang menunjukkan bahwa aktivitas akan ditampilkan kepada pengguna?
29. Metode mana yang dipanggil ketika suatu aktivitas tidak lagi terlihat oleh pengguna dan tidak diperlukan untuk sementara waktu?
30. Apakah fungsi onDestroy() akan dipanggil jika memori sempit?
31. Metode mana yang dipanggil untuk mengizinkan aktivitas menyimpan status per-instance?
32. Apa itu kegiatan?
33. Apa itu niat?
34. Apa itu layanan?
35. Apa itu penyedia konten?
36. Apa itu sumber daya?
37. Apa fungsi utama dari kelas R?
38. Bagaimana Anda mengakses operasi kritis tertentu?

BAB 3

PENGENALAN KONSEP KUNCI ANDROID

Memahami konsep utama android adalah persyaratan dasar untuk mendesain aplikasi seluler Android. Dalam bab ini, kami memperkenalkan beberapa konsep dasar Android, termasuk komponen aplikasi, sumber daya aplikasi, dan manifes aplikasi. Siswa akan mampu menjawab pertanyaan-pertanyaan berikut setelah membaca bab ini.

1. Apa itu aktivitas di Android?
2. Bisakah kita langsung menyimpan file sumber daya di dalam direktori res/?
3. Apa itu APP MAINFEST?

3.1 KOMPONEN APLIKASI

Komponen aplikasi adalah blok pembangun penting dari aplikasi Android. Setiap komponen adalah titik berbeda yang melaluinya sistem dapat memasuki aplikasi Anda. Tidak semua komponen merupakan titik masuk aktual bagi pengguna, dan beberapa saling bergantung satu sama lain, tetapi masing-masing komponen ada sebagai entitasnya sendiri dan memainkan peran khusus. Masing-masing adalah blok penyusun unik yang membantu menentukan perilaku keseluruhan aplikasi Anda. Subbagian berikut mewakili empat jenis komponen aplikasi, yang mencakup aktivitas, layanan, penyedia konten, dan maksud.

3.1.1 Kegiatan

Aktivitas mewakili satu layar dengan antarmuka pengguna. Misalnya, aplikasi email mungkin memiliki satu aktivitas yang menampilkan daftar email baru, aktivitas lain untuk menulis email, dan aktivitas lain untuk membaca email. Meskipun aktivitas bekerja sama untuk membentuk pengalaman pengguna yang kohesif dalam aplikasi email, masing-masing aktivitas tidak bergantung pada yang lain. Dengan demikian, aplikasi yang berbeda dapat memulai salah satu dari aktivitas ini (jika aplikasi email mengizinkannya). Misalnya, aplikasi kamera dapat memulai aktivitas di aplikasi email yang membuat email baru, agar pengguna dapat berbagi gambar. Anda dapat menemukan informasi lebih lanjut tentang aktivitas di

<https://developer.android.com/guide/components/activities.html>.

3.1.2 Layanan

Layanan adalah komponen yang berjalan di latar belakang untuk melakukan operasi yang berjalan lama atau untuk melakukan pekerjaan untuk proses jarak jauh. Sebuah layanan tidak menyediakan antarmuka pengguna. Misalnya, layanan mungkin memutar musik di latar belakang saat pengguna berada di aplikasi yang berbeda, atau mungkin mengambil data melalui jaringan tanpa memblokir interaksi pengguna dengan suatu aktivitas. Komponen lain, seperti aktivitas, dapat memulai layanan dan membiarkannya berjalan atau mengikatnya untuk berinteraksi dengannya. Anda dapat menemukan informasi lebih lanjut tentang penyedia konten di

<https://developer.android.com/reference/android/app/Service.html>.

3.1.3 Penyedia Konten

Penyedia konten mengelola kumpulan data aplikasi bersama. Anda dapat menyimpan data dalam sistem file, database SQLite, di web, atau lokasi penyimpanan persisten lainnya yang dapat diakses aplikasi Anda. Melalui penyedia konten, aplikasi lain dapat melakukan kueri atau bahkan mengubah data (jika penyedia konten mengizinkannya). Misalnya, sistem Android menyediakan penyedia konten yang mengelola informasi kontak pengguna. Dengan demikian, aplikasi apa pun dengan izin yang sesuai dapat meminta bagian dari penyedia konten (seperti `ContactsContract.Data`) untuk membaca dan menulis informasi tentang orang tertentu. Penyedia konten juga berguna untuk membaca dan menulis data yang bersifat pribadi untuk aplikasi Anda dan tidak dibagikan. Misalnya, aplikasi sampel Note Pad menggunakan penyedia konten untuk menyimpan catatan. Anda dapat menemukan informasi lebih lanjut tentang penyedia konten di:

<https://developer.android.com/reference/android/content/ContentProvider.html>

3.1.4 Tujuan

Intent adalah mekanisme untuk mendeskripsikan tindakan tertentu, seperti “pilih foto”, “telepon ke rumah”, atau “buka pintu ruang pod”. Di Android, hampir semuanya berjalan melalui maksud, jadi Anda memiliki banyak peluang untuk mengganti atau menggunakan kembali komponen. Misalnya, ada maksud untuk "mengirim email." Jika aplikasi Anda perlu mengirim email, Anda bisa memanggil maksud itu. Atau, jika Anda sedang menulis aplikasi email baru, Anda bisa mendaftarkan aktivitas untuk menangani maksud tersebut dan mengganti program email standar. Saat berikutnya seseorang mencoba mengirim email, mereka akan mendapatkan opsi untuk menggunakan program Anda, bukan yang standar. Anda dapat menemukan informasi selengkapnya tentang maksud di :

<https://developer.android.com/guide/components/intents-filters.html>.

Aspek unik dari desain sistem Android adalah aplikasi apa pun dapat memulai komponen aplikasi lain. Misalnya, jika Anda ingin pengguna mengambil foto dengan kamera perangkat, mungkin ada aplikasi lain yang melakukannya dan aplikasi Anda dapat menggunakannya, alih-alih mengembangkan aktivitas untuk mengambil foto sendiri. Anda tidak perlu memasukkan atau bahkan menautkan ke kode dari aplikasi kamera. Sebagai gantinya, Anda cukup memulai aktivitas di aplikasi kamera yang mengambil foto. Setelah selesai, foto tersebut bahkan dikembalikan ke aplikasi Anda sehingga Anda dapat menggunakannya. Bagi pengguna, seperti halnya kamera sebenarnya adalah bagian dari aplikasi Anda.

Saat sistem memulai komponen, sistem akan memulai proses untuk aplikasi tersebut (jika belum berjalan) dan membuat instance kelas yang diperlukan untuk komponen tersebut. Misalnya, jika aplikasi Anda memulai aktivitas di aplikasi kamera yang mengambil foto, aktivitas tersebut berjalan dalam proses yang termasuk dalam aplikasi kamera, bukan dalam proses aplikasi Anda. Oleh karena itu, tidak seperti aplikasi pada kebanyakan sistem lain, aplikasi Android tidak memiliki satu titik masuk (misalnya, tidak ada fungsi `main()`). Karena sistem menjalankan setiap aplikasi dalam proses terpisah dengan izin file yang membatasi

akses ke aplikasi lain, aplikasi Anda tidak dapat secara langsung mengaktifkan komponen dari aplikasi lain. Sistem Android, bagaimanapun, bisa. Jadi, untuk mengaktifkan komponen di aplikasi lain, Anda harus mengirimkan pesan ke sistem yang menetapkan maksud Anda untuk memulai komponen tertentu. Sistem kemudian mengaktifkan komponen untuk Anda.

Intent dapat digunakan untuk mengaktifkan aktivitas dan layanan, tetapi penyedia konten diaktifkan saat ditargetkan oleh permintaan dari ContentResolver. Ada metode terpisah untuk mengaktifkan setiap jenis komponen:

1. Anda dapat memulai aktivitas (atau memberinya sesuatu yang baru untuk dilakukan) dengan meneruskan Intent ke `startActivity()` atau `startActivityForResult()` (bila Anda ingin aktivitas mengembalikan hasil).
2. Anda dapat memulai layanan (atau memberikan instruksi baru ke layanan yang sedang berjalan) dengan meneruskan Intent ke `startService()`. Atau Anda dapat mengikat ke layanan dengan meneruskan Intent ke `bindService()`.
3. Anda dapat melakukan kueri ke penyedia konten dengan memanggil `query()` pada ContentResolver.

Gambar 3.1 adalah `AndroidManifest.xml` default yang dihasilkan oleh Integrated Development Environment (IDE) setelah kita membuat aplikasi Android kosong. `MainActivity` adalah satu-satunya aktivitas dalam proyek ini, dan merupakan subkelas dari `Activity`, seperti yang ditampilkan "kelas `MainActivity` memperluas `Activity`". Di `MainActivity`, kita perlu mengimplementasikan metode callback yang dipanggil sistem saat aktivitas bertransisi di antara berbagai status siklus hidupnya. Metode `onCreate()` sangat diperlukan, dan akan dipanggil saat `MainActivity` dibuat. Dalam implementasi metode `onCreate()`, kita harus menginisialisasi komponen penting dari aktivitas ini. Kita harus memanggil `setContentView()` untuk menentukan tata letak antarmuka pengguna aktivitas ini. Meskipun proses ini diimplementasikan oleh IDE, kita perlu memiliki pengetahuan ini.

3.2 SUMBER DAYA APLIKASI

Anda harus selalu mengeksternalisasi sumber daya, seperti gambar dan string, dari kode aplikasi Anda, sehingga Anda dapat memeliharanya secara independen. Anda harus menempatkan setiap jenis sumber daya dalam subdirektori tertentu dari direktori `res/` proyek Anda, seperti yang ditunjukkan pada Gambar 3.2. `drawable/` berisi file bitmap, seperti `png`, `jpg` dan `gif`. `layout/` berisi file XML yang mendefinisikan tata letak antarmuka pengguna. `menu/` berisi file XML yang mendefinisikan menu aplikasi. `values/` berisi file XML yang berisi nilai sederhana, seperti string, integer, dan warna. Selain yang ditunjukkan pada Gambar 3.1, kita dapat menambahkan beberapa file sumber daya lain ke dalam direktori `res/`, seperti file `animator/`, `raw/`, dan `xml/`.

```

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

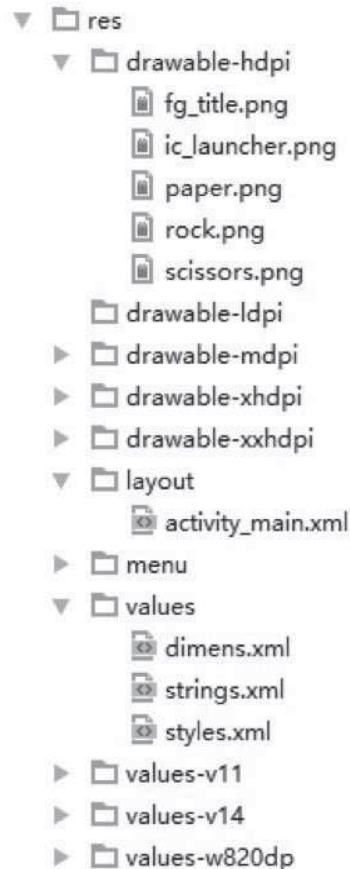
        return super.onOptionsItemSelected(item);
    }
}

```

Gambar 3.1 Default MainActivity.java.

File animator berisi animasi properti Android. Sistem animasi properti adalah kerangka kerja yang kuat yang memungkinkan Anda menganimasikan hampir semua hal. Anda dapat menentukan animasi untuk mengubah properti objek apa pun dari waktu ke waktu, terlepas dari apakah itu menarik ke layar atau tidak. Anda dapat menemukan informasi lebih lanjut tentang animasi properti di <http://developer.android.com/guide/topics/graphics/prop-animation.html>. File mentah menyimpan file apa pun dalam bentuk mentahnya. Anda harus memanggil `Resource.openRawResource()` untuk membuka sumber daya ini dengan `InputStream` mentah. File XML berisi file XML arbitrer yang dapat dibaca saat runtime dengan memanggil `Resource.getXML()`. Berbagai file konfigurasi XML harus disimpan di sini, seperti konfigurasi yang dapat dicari.

PETUNJUK: Jangan pernah menyimpan file sumber daya secara langsung di dalam direktori `res/`, karena akan menyebabkan kesalahan kompilator.



Gambar 3.2 Hirarki file untuk proyek sederhana.

3.3 APLIKASI UTAMA FEST

File manifes sangat diperlukan di setiap aplikasi Android. File manifes menyajikan informasi penting tentang aplikasi Anda ke sistem Android, informasi yang harus dimiliki sistem sebelum dapat menjalankan kode aplikasi apa pun. Antara lain, manifes melakukan hal berikut:

1. Ini menamai paket Java untuk aplikasi tersebut. Nama paket berfungsi sebagai pengidentifikasi unik untuk aplikasi.
2. Ini menjelaskan komponen aplikasi, aktivitas, layanan, penerima siaran, dan penyedia konten yang terdiri dari aplikasi. Ini menamai kelas yang mengimplementasikan setiap komponen dan memublikasikan kemampuannya (misalnya, pesan Intent mana yang dapat mereka tangani). Deklarasi ini memungkinkan sistem Android mengetahui apa saja komponen tersebut dan dalam kondisi apa mereka dapat diluncurkan.
3. Ini menentukan proses mana yang akan menampung komponen aplikasi.
4. Ini mendeklarasikan izin mana yang harus dimiliki aplikasi untuk mengakses bagian yang dilindungi dari Application Programming Interface (API) dan berinteraksi dengan aplikasi lain.
5. Ini juga menyatakan izin yang harus dimiliki orang lain untuk berinteraksi dengan komponen aplikasi.

6. Ini mencantumkan kelas Instrumentasi yang menyediakan profil dan informasi lain saat aplikasi sedang berjalan. Deklarasi ini hadir dalam manifes hanya saat aplikasi sedang dikembangkan dan diuji; mereka dihapus sebelum aplikasi diterbitkan.
7. Ini mendeklarasikan level minimum Android API yang dibutuhkan aplikasi.
8. Ini mencantumkan pustaka yang harus ditautkan ke aplikasi.

3.3.1 Elemen

Hanya elemen `<manifest>` dan `<application>` yang diperlukan, masing-masing harus ada dan hanya dapat muncul sekali. Sebagian besar yang lain dapat, terjadi berkali-kali atau tidak sama sekali, meskipun setidaknya beberapa dari mereka harus ada agar manifes mencapai sesuatu yang berarti. Jika suatu elemen mengandung apa pun, itu mengandung elemen lain. Semua nilai ditetapkan melalui atribut, bukan sebagai data karakter dalam suatu elemen. Elemen pada level yang sama umumnya tidak berurutan. Sebagai contoh, Elemen `<aktivitas>`, `<penyedia>`, dan `<layanan>` dapat digabungkan dalam urutan apa pun. (Elemen `<activity-alias>` adalah pengecualian untuk aturan ini: Elemen ini harus mengikuti `<aktivitas>` yang merupakan alias untuknya.)

3.3.2 Atribut

Dalam arti formal, semua atribut adalah opsional. Namun, ada beberapa yang harus ditentukan untuk elemen untuk mencapai tujuannya. Gunakan dokumentasi sebagai panduan. Untuk atribut yang benar-benar opsional, ia menyebutkan nilai default atau menyatakan apa yang terjadi tanpa adanya spesifikasi. Kecuali untuk beberapa atribut dari elemen `<manifest>` root, semua nama atribut dimulai dengan awalan `android:`, misalnya, `android:alwaysRetainTaskState`. Karena awalan bersifat universal, dokumentasi umumnya menghilangkannya saat mengacu pada atribut berdasarkan nama.

3.3.3 Mendeklarasikan Nama Kelas

Banyak elemen yang sesuai dengan objek Java, termasuk elemen untuk aplikasi itu sendiri (elemen `<application>`) dan komponen utamanya, aktivitas (`<aktivitas>`), layanan (`<layanan>`), penerima siaran (`<receiver>`), dan penyedia konten (`<penyedia>`). Jika Anda mendefinisikan subclass, seperti yang hampir selalu Anda lakukan untuk kelas komponen (Activity, Service, BroadcastReceiver, dan Content-Provider), subclass dideklarasikan melalui atribut `name`. Nama harus menyertakan penunjukan paket lengkap. Namun, sebagai singkatan, jika karakter pertama dari string adalah titik, string tersebut ditambahkan ke nama paket aplikasi (seperti yang ditentukan oleh atribut paket elemen `<manifest>`). Saat memulai komponen, Android membuat turunan dari subclass bernama. Jika subclass tidak ditentukan, itu membuat turunan dari kelas dasar.

3.3.4 Beberapa Nilai

Jika lebih dari satu nilai dapat ditentukan, elemen tersebut hampir selalu diulang, daripada mendaftar beberapa nilai dalam satu elemen.

3.3.5 Nilai Sumber Daya

Beberapa atribut memiliki nilai yang dapat ditampilkan kepada pengguna, misalnya, label dan ikon untuk suatu aktivitas. Nilai atribut ini harus dilokalkan dan oleh karena itu ditetapkan dari sumber daya atau tema. Nama paket dapat dihilangkan jika sumber daya berada dalam paket yang sama dengan aplikasi, tipe adalah jenis sumber daya, seperti "string" atau "dapat digambar", dan nama adalah nama yang mengidentifikasi sumber daya tertentu.

3.3.6 Nilai Sengatan

Jika nilai atribut berupa string, garis miring terbalik ganda ('\\') harus digunakan untuk menghindari karakter, misalnya, '\\n' untuk baris baru atau '\\uxxxx' untuk karakter Unicode.



```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.csis.pace.edu.mypace" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="My Pace "
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="My Pace " >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Gambar 3.3 Default AndroidManifest.xml

Gambar 3.3 adalah file AndroidManifest.xml default yang dihasilkan oleh IDE setelah aplikasi Android kosong. Pada baris pembuatan ketiga, "com.example.csis.pace.edu.mypace," adalah nama paket proyek, dan sama persis dengan baris pertama MainActivity.java. Banyak elemen di dalam <application> dan </application> sesuai dengan objek Java, termasuk aktivitas (<activity>), layanan (<service>), penerima siaran (<receiver>), dan penyedia konten (<provider>). Dalam proyek kami, kami hanya membuat aktivitas, jadi, hanya ada satu <aktivitas> di file AnroidManifest.xml. Dalam <aktivitas> ini, android:name=".MainActivity" menunjukkan nama aktivitas. <intent-filter> menentukan tipe intent yang dapat ditanggapi oleh aktivitas, layanan, atau penerima siaran. Filter maksud mendeklarasikan kemampuan komponen induknya, apa yang dapat dilakukan aktivitas atau layanan, dan jenis siaran apa yang dapat ditangani oleh penerima. Ini membuka

komponen untuk menerima maksud dari jenis yang diiklankan, sambil menyaring yang tidak berarti untuk komponen.

Saat menambahkan tindakan ke filter maksud. Sebuah elemen `<intent-filter>` harus berisi satu atau lebih elemen `<action>`. Jika tidak mengandung apa pun, tidak ada objek Intent yang akan melewati filter. Beberapa tindakan standar didefinisikan dalam kelas Intent sebagai konstanta `ACTION_string`. Untuk menetapkan salah satu tindakan ini ke atribut ini, tambahkan "android.intent.action." ke string yang mengikuti `ACTION_`. Dalam proyek kami, gunakan "android.intent.action.MAIN" untuk `ACTION_MAIN`. `<category>` digunakan untuk menambahkan nama kategori ke filter maksud. Kategori standar didefinisikan dalam kelas Intent sebagai Konstanta `CATEGORY_name`. Nama yang ditetapkan di sini dapat diturunkan dari konstanta tersebut dengan awalan "android.intent.category." dengan nama yang mengikuti `CATEGORY_`. Dalam proyek kami, nilai string adalah "android.intent.category.LAUNCHER" untuk `CATEGORY_LAUNCHER`.

3.4 LATIHAN

3.4.1 Latihan Dasar

1. Apa yang dimaksud dengan prosedural?
2. Apa itu desain deklaratif?
3. Apa yang dilakukan Android untuk mengatasi kesenjangan antara prosedural dan deklaratif?
4. Apa yang dilakukan metode `onCreate()`?
5. Apa yang dilakukan metode `setContentView()`?
6. Apa itu `R.layout.main`?
7. Apa yang dilakukan `main.xml`?
8. Bagaimana cara kerja kelas R?
9. Bagaimana Android menangani XML?
10. Apa itu tata letak?
11. Apa yang dilakukan `FrameLayout`?
12. Apa yang dilakukan `LinearLayout`?
13. Apa yang dilakukan `RelativeLayout`?
14. Apa yang dilakukan `TableLayout`?
15. Bagaimana Anda mengubah emulator ke mode lansekap?
16. Apa hal pertama yang memulai aktivitas di Android?
17. Apa yang dimaksud dengan niat publik?
18. Apa yang dimaksud dengan maksud pribadi?
19. Apa itu tema?
20. Jenis menu apa yang didukung Android?
21. Apa yang dilakukan kelas Log?

3.4.2 Latihan Tingkat Lanjut

1. Setelah membaca buku teks tentang antarmuka pengguna, harap bicarakan pemahaman Anda tentang antarmuka pengguna.
2. Berikan beberapa contoh untuk menjelaskan apa yang ramah penggunaan menurut Anda.

BAB 4

GRAFIK 2 D DAN MULTIMEDIA DI ANDROID

GRAFIS 2D DAN DESAIN UI adalah dua aspek penting dalam desain Antarmuka Pengguna (UI). Dalam bab ini, kami akan memperkenalkan grafik 2-D dan beberapa teknik desain UI tingkat lanjut. Teknik utama grafik 2-D meliputi Color, Paint, Path, Canvas, Drawable, dan Button Selector. Siswa juga akan belajar cara membuat beberapa layar, bilah tindakan, dan tampilan kustom di UI. Selain itu, multimedia pada sistem Android adalah fungsi yang meningkatkan kemampuan adopsi aplikasi seluler Anda. Dalam bab ini, kami akan memperkenalkan multimedia di Android dan cara menambahkan multimedia ke aplikasi Android kami. Tiga aspek utama dalam multimedia meliputi Media, Audio, dan Video.

4.1 PENGENALAN TEKNIK GRAFIS 2 D

Android mengimplementasikan fungsi 2-D lengkap dalam satu paket, bernama `android.graphics`. Paket ini menyediakan berbagai macam alat grafis, seperti kanvas, filter warna, titik, garis, dan persegi panjang. Kita dapat menggunakan alat grafis ini untuk menggambar layar secara langsung. Kami akan memperkenalkan beberapa pengetahuan dasar secara rinci. Pertama-tama, kami membuat proyek aplikasi Android baru bernama `ColorTester`.

4.1.1 Warna

Warna direpresentasikan sebagai bilangan bulat yang dikemas, terdiri dari 4 byte: Alfa, Merah, Hijau, dan Biru. Alpha adalah ukuran transparansi, dari nilai 0 sampai nilai 255. Nilai 0 menunjukkan warna benar-benar transparan. Nilai 255 menunjukkan warna benar-benar buram. Selain alpha, setiap komponen berkisar antara 0 dan 255, dengan 0 berarti tidak ada kontribusi untuk komponen tersebut, dan 255 berarti kontribusi 100%.

Kita dapat membuat warna ungu setengah buram seperti: `int`

```
color1 = Color.argb(127, 255, 0, 255);
```

Atau dalam file sumber daya XML, seperti:

```
<color name="half_op_purple">#7fff00ff</color>
```

Warna dalam file sumber daya XML Android harus diformulasikan sebagai "#" + 6 atau 8 bit angka Heksadesimal.

Selanjutnya, Android menawarkan beberapa warna dasar sebagai konstanta, seperti yang ditunjukkan pada Gambar 4.1. Kita dapat menggunakannya secara langsung, seperti:

```
int color2 = Color.Black;
```

Di Android Studio, kita dapat melihat pratinjau warna yang kita buat dalam file XML, seperti yang ditunjukkan pada Gambar 4.2. Ada beberapa kotak kecil dengan warna yang dibuat dalam garis yang sama. Kita dapat melihat bahwa #ffffff berwarna putih buram, dan #ff000000 berwarna hitam buram.

BLUE	int
GRAY	int
LTGRAY	int
parseColor (String colorString)	int
HSVToColor (float[] hsv)	int
HSVToColor (int alpha, float[] hsv)	int
alpha (int color)	int
BLACK	int
blue (int color)	int
CYAN	int
DKGRAY	int
GREEN	int
green (int color)	int
MAGENTA	int
RED	int
red (int color)	int
rgb (int red, int green, int blue)	int
TRANSPARENT	int
WHITE	int
YELLOW	int

Gambar 4.1 Warna sebagai konstanta yang disediakan oleh Android.

Kita dapat menggunakan warna ini, dibuat di colors.xml oleh "color/color_name". Misalnya, android:background="color/my_color". Setelah kita mendefinisikan beberapa warna dalam file XML, kita dapat merujuknya dengan namanya, seperti yang kita lakukan untuk string, atau kita dapat menggunakannya dalam kode Java seperti:

```
int color3 = getResources().getColor(R.color.my_color); or
int color3 = R.color.text_color
```

Metode getResources() mengembalikan kelas ResourceManager untuk aktivitas saat ini, dan getColor() meminta manajer mencari warna yang diberikan ID sumber daya.



Gambar 4.2 Pratinjau warna dalam file XML di Android Studio.

4.1.2 Cat

Kelas Paint menyimpan informasi gaya dan warna pada geometri gambar, teks, dan bitmap. Sebelum kita menggambar sesuatu di layar, kita bisa mengatur warna ke Paint melalui metode setColor().

```
Paint cPaint = new Paint();
cPaint.setColor(Color.LTGRAY);
Paint tPaint = new Paint();
tPaint.setColor(Color.BLUE);
tPaint.setTextSize((float) 20.0);
```

Gambar 4.3 Kelas Paint di Android.

Seperti yang ditunjukkan pada Gambar 4.3, kita membuat dua Paint, yaitu cPaint untuk menggambar lingkaran dan tPaint untuk menggambar teks. Kami mengatur warna lingkaran sebagai abu-abu muda dan warna teks sebagai biru. Selain warna, kita juga bisa mengatur atribut lain ke kelas Paint, seperti TextSize.

4.1.3 Jalur

Kelas Path merangkum beberapa jalur geometris kontur, seperti garis, persegi panjang, lingkaran, dan kurva. Gambar 4.4 adalah contoh yang mendefinisikan jalur melingkar dan jalur persegi panjang. Baris kedua mendefinisikan sebuah lingkaran, yang pusatnya berada pada posisi x=300, y=200, dengan radius 150 piksel. Garis keempat merupakan persegi panjang yang titik kiri atas berada pada posisi x=150, y=400, dan titik kanan bawah pada posisi x=400, y=650. Path.Direction.CW menunjukkan bahwa bentuk akan digambar searah jarum jam. Arah lainnya adalah CCW, yang menunjukkan berlawanan arah jarum jam.

```
Path path = new Path();
path.addCircle(300, 200, 150, Path.Direction.CW);

Path path2 = new Path();
path2.addRect(150, 400, 400, 650, Path.Direction.CW);
```

Gambar 4.4 Buat dua objek Path dan tambahkan detailnya.

4.1.4 Kanvas

Untuk menggambar sesuatu, kita perlu menyiapkan empat komponen dasar, termasuk Bitmap untuk menampung piksel, Canvas untuk menampung panggilan undian, primitif menggambar, dan Paint. Bitmap adalah tempat untuk menggambar sesuatu, dan Canvas digunakan untuk menahan panggilan "draw". Gambar primitif dapat berupa Rect, Circle, Path, Text, dan Bitmap. Di Android, layar tampilan diambil oleh Aktivitas, yang menghosting View, yang pada gilirannya menghosting Canvas. Kita bisa menggambar di kanvas dengan mengganti metode View.onDraw(). Objek Canvas adalah satu-satunya parameter untuk metode onDraw(). Kami membuat aktivitas baru, yang berisi tampilan yang disebut GraphicsView, tetapi bukan layout.xml, seperti yang ditunjukkan pada Gambar 4.8. Pada Gambar 4.5, kita mengomentari kode asli dan menyetel tampilan konten aktivitas ini ke beberapa layout.xml, dan menyetelnya ke beberapa tampilan baru yang kita buat, yaitu GraphicsView.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // setContentView(R.layout.activity_main);
    setContentView(new GraphicsView(this));
}

static public class GraphicsView extends View {
    public GraphicsView(Context context){
        super(context);
    }

    @Override
    protected void onDraw(Canvas canvas){...}
}

```

Gambar 4.5 Aktivitas baru yang contentView-nya adalah tampilan yang dibuat sendiri tetapi bukan layout.xml.

Mari kita tinjau dua metode mendesain aplikasi Android. Ada dua metode untuk merancang aplikasi Android, yaitu prosedural dan deklaratif. "setContentView(R.layout.activity_main)" adalah contoh khas dari deklaratif, yang dijelaskan semua objek dalam aktivitas menggunakan file XML. "setContentView(new GraphicsView(this))" adalah contoh khas dari prosedural, yang berarti menulis kode Java untuk membuat dan memanipulasi semua objek antarmuka pengguna. Kelas baru ini, GraphicsView, memperluas tampilan kelas. Metode onDraw() adalah over-rider dan digunakan untuk mengimplementasikan fungsi menggambar. Gambar 4.6 menunjukkan detail metode onDraw(). Kami menggunakan Paint dengan warna berbeda untuk menggambar Path pada View melalui pemanggilan metode onDraw(Canvas). Sementara itu, kita memiliki pilihan lain untuk membuat Canvas, seperti yang ditunjukkan pada Gambar 4.8. Pada Gambar 4.8, kita membuat Bitmap yaitu persegi yang ukurannya 100*100 dan akan digunakan sebagai argumen Canvas.

Kemudian kita dapat menggunakan kanvas ini sama seperti yang ditawarkan dalam metode `onDraw()`.

4.1.5 Dapat Digambar

`Android.graphics.drawable` menyediakan kelas untuk mengelola berbagai elemen visual, yang dimaksudkan untuk ditampilkan, seperti bitmap dan gradien. Kita bisa menggabungkan `drawable` dengan grafik lain, atau kita bisa menggunakannya di widget UI, seperti latar belakang untuk sebuah tombol. Android menawarkan jenis `drawable` berikut:

Bitmap: File grafik bitmap (.png, .jpg, atau .gif).

Nine-Patch: File PNG dengan area yang dapat diregangkan untuk memungkinkan perubahan ukuran gambar berdasarkan konten (.9.png).

Layer: Sumber Daya Dapat Digambar yang mengelola larik dari Sumber Daya Dapat Digambar lainnya. Ini digambar dalam urutan array, sehingga elemen dengan indeks terbesar digambar di atas.

```
@Override
protected void onDraw(Canvas canvas){

    String QUOTE = "PACE UNIVERSITY CSIS DEPT.";

    Paint cPaint = new Paint();
    cPaint.setColor(Color.LTGRAY);
    Paint tPaint = new Paint();
    tPaint.setColor(Color.BLUE);
    tPaint.setTextSize((float) 20.0);

    Path path = new Path();
    path.addCircle(300, 200, 150, Path.Direction.CW);

    Path path2 = new Path();
    path2.addRect(150, 400, 400, 650, Path.Direction.CW);

    canvas.drawPath(path, cPaint);
    canvas.drawTextOnPath(QUOTE, path, 0, 20, tPaint);

    canvas.drawPath(path2, tPaint);
}
```

Gambar 4.6 Metode “`onDraw()`” yang menggambar lingkaran dan persegi panjang.



Gambar 4.7 Hasil running GraphicsView.

```
//Creating a new Canvas object
Bitmap bitmap = Bitmap.createBitmap(100,100,Bitmap.Config.ARGB_8888);
Canvas canvas = new Canvas(bitmap);
```

Gambar 4.8 Gunakan Bitmap untuk membuat Canvas baru.

Status: File XML yang mereferensikan grafik bitmap yang berbeda untuk status yang berbeda (misalnya, untuk menggunakan gambar yang berbeda saat tombol ditekan).

Level: File XML yang mendefinisikan sumber daya dapat digambar yang mengelola sejumlah Sumber Daya Dapat Digambar alternatif, masing-masing diberi nilai numerik maksimum. Membuat LevelListDrawable.

Transition: File XML yang mendefinisikan sumber daya dapat digambar yang dapat memudahkan silang antara dua sumber daya yang dapat digambar.

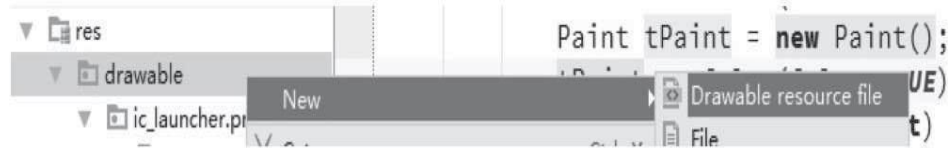
Inset Drawable: File XML yang mendefinisikan drawable yang menyisipkan drawable lain dengan jarak tertentu. Ini berguna saat Tampilan membutuhkan sumber daya dapat digambar di latar belakang yang lebih kecil dari batas sebenarnya Tampilan.

Clip: File XML yang mendefinisikan drawable yang memotong Drawable lain berdasarkan nilai level saat ini dari Drawable.

Skala: File XML yang mendefinisikan sumber daya dapat digambar yang mengubah ukuran Sumber Daya Dapat Digambar lain berdasarkan nilai levelnya saat ini.

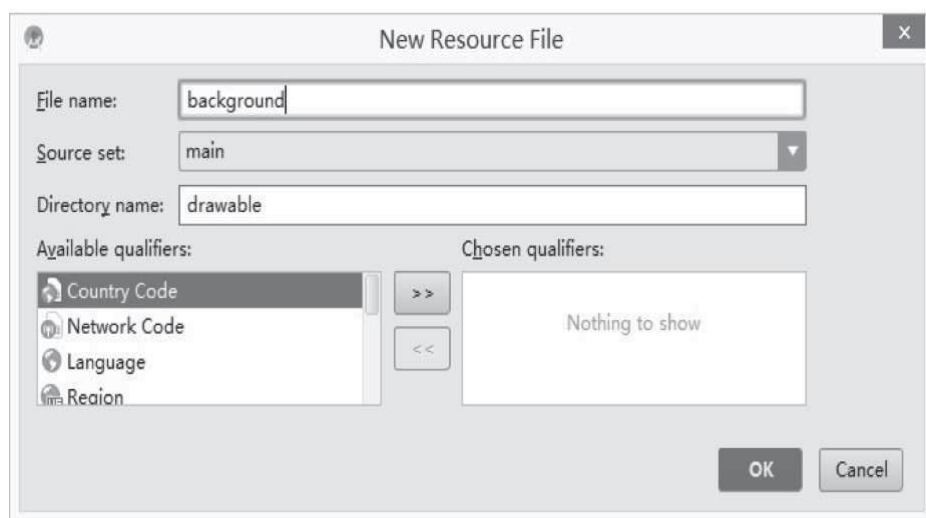
Bentuk: File XML yang mendefinisikan bentuk geometris, termasuk warna dan gradien.

Sumber daya yang dapat digambar adalah konsep umum untuk grafik yang dapat digambar ke layar dan dapat diambil dengan Antarmuka Pemrograman Aplikasi (API). Sekarang kita akan menambahkan latar belakang gradien ke ColorTester kita. Kami membuat file sumber daya yang dapat digambar di res\drawable, dan kemudian membuat Bentuk di dalam file background.xml, seperti yang ditunjukkan pada Gambar 4.9 dan Gambar 4.10.



Gambar 4.9 Langkah pertama membuat file sumber daya Drawable baru.

Seperti ditunjukkan pada Gambar 4.11, kita mendefinisikan gradien dari warna awal hingga warna akhir. Sudut menunjukkan arah gradien, dan itu harus diekstraksi dikalikan 45. Jika sudut = 0, urutannya dari kiri ke kanan. Jika sudut = 90, urutannya dari bawah ke atas. Jika sudut = 180, urutannya dari kanan ke kiri. Jika sudut = 270, urutannya dari atas ke bawah.



Gambar 4.10 Langkah kedua membuat file sumber daya Drawable baru.



Gambar 4.11 Bentuk Dapat Digambar.

Kemudian tambahkan satu atribut ke dalam RelativeLayout di activity_main.xml sebagai "android:background:@drawable/background". Hasil running ditunjukkan pada Gambar 4.12. Selain gradien, ada beberapa atribut umum lainnya yang dapat ditambahkan ke dalam bentuk, termasuk stroke, sudut, dan padding. Kami menambahkannya ke dalam bentuk background.xml, dan mengatur warna goresan menjadi merah, lebar tanda hubung adalah 10dp, dll. Atribut dan hasil lari ditunjukkan pada Gambar 4.13. Dari Gambar 4.13, kita dapat melihat bahwa latar belakang dicoret dengan garis merah, dan setiap sudut memiliki tepi yang bulat.



Gambar 4.12 Latar belakang gradien.



Gambar 4.13 Stroke, Sudut, dan Sumber Daya Dapat Digambar Padding

4.1.6 Pemilih Tombol

Kami ingin mengatur warna yang berbeda untuk tombol ketika mereka berada di negara bagian yang berbeda. Kami mengatur warna default tombol sebagai ungu muda, dan warna saat ditekan adalah oranye muda. Seperti yang kita perkenalkan di bagian sebelumnya, kita perlu membuat file sumber daya yang dapat digambar untuk mengimplementasikan fungsi ini. Jadi, kita membuat file sumber daya baru yang dapat digambar bernama "button_selection", dan di antara <selector> dan </selector> tambahkan dua item. Yang pertama adalah keadaan ditekan, yang menunjukkan bahwa tombol ditekan, seperti yang ditunjukkan pada Gambar 4.14. Yang kedua adalah status default, seperti yang ditunjukkan pada Gambar 4.15.

```

<item android:state_pressed="true" >
  <shape>
    <gradient
      android:startColor="#ffc2b7"
      android:endColor="#FFFFFF"
      android:type="radial"
      android:gradientRadius="50" />
    <stroke
      android:width="2dp"
      android:color="#dcdcdc"
      android:dashWidth="5dp"
      android:dashGap="3dp" />
    <corners
      android:radius="2dp" />
    <padding
      android:left="10dp"
      android:top="10dp"
      android:right="10dp"
      android:bottom="10dp" />
  </shape>
</item>

```

Gambar 4.14 Status default tombol.

Kemudian, kami menambahkan satu atribut ke ketiga tombol sebagai berikut: `android:background="@drawable/button_selector"`. Hasil running ditunjukkan pada Gambar 4.16.

4.2 DESAIN UI LANJUTAN

Android menyediakan kerangka kerja yang fleksibel untuk desain UI yang memungkinkan aplikasi menampilkan tata letak yang berbeda untuk perangkat yang berbeda, membuat widget UI kustom, dan mengontrol aspek UI sistem di luar jendela aplikasi.

4.2.1 Beberapa Layar

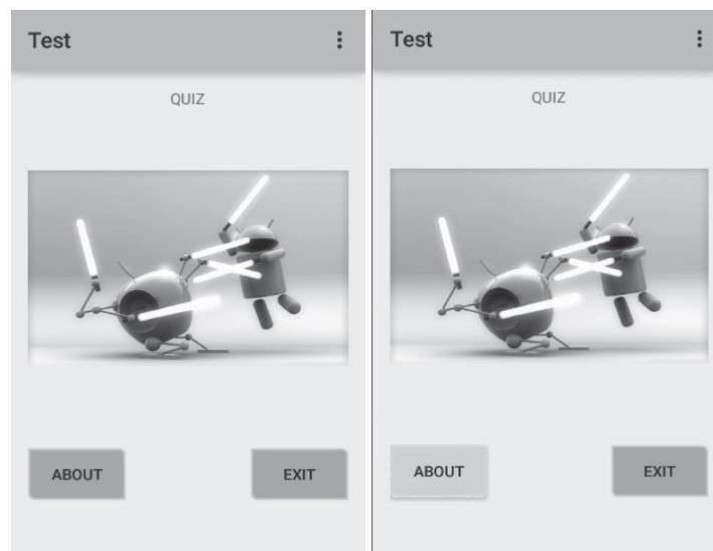
Tujuan dari bagian ini adalah untuk membangun UI, yang cukup fleksibel untuk dipasang dengan sempurna di layar mana pun dan untuk membuat pola interaksi berbeda yang dioptimalkan untuk ukuran layar berbeda. Untuk memastikan bahwa tata letak Anda fleksibel dan beradaptasi dengan ukuran layar yang berbeda, Anda harus menggunakan `"wrap_content"` dan `"match_parent"` untuk lebar dan tinggi beberapa komponen tampilan. Jika Anda menggunakan `"wrap_content"`, lebar atau tinggi tampilan disetel ke ukuran minimum yang diperlukan agar sesuai dengan konten dalam tampilan itu, sementara `"match_parent"` (juga dikenal sebagai `"fill_parent"` sebelum API level 8) membuat komponen diperluas ke cocok dengan ukuran tampilan induknya.

Anda dapat membuat tata letak yang cukup rumit menggunakan instance `LinearLayout` dan kombinasi ukuran `"wrap_content"` dan `"match_parent"`. Namun, `LinearLayout` tidak memungkinkan Anda mengontrol secara tepat hubungan spasial tampilan turunan; tampilan dalam `LinearLayout` cukup berbaris berdampingan. Jika Anda memerlukan tampilan anak untuk diorientasikan dalam variasi selain garis lurus, solusi yang lebih baik adalah sering menggunakan `RelativeLayout`, yang

memungkinkan Anda untuk menentukan tata letak dalam kaitannya dengan hubungan khusus antar komponen. Misalnya, Anda dapat menyelaraskan satu tampilan anak di sisi kiri dan tampilan lain di sisi kanan layar.

```
<item android:state_focused="false">
  <shape>
    <solid android:color="#8f0000f0"/>
    <stroke
      android:width="2dp"
      android:color="#fad3cf" />
    <corners
      android:topRightRadius="5dp"
      android:bottomLeftRadius="5dp"
      android:topLeftRadius="0dp"
      android:bottomRightRadius="0dp"
    />
    <padding
      android:left="10dp"
      android:top="10dp"
      android:right="10dp"
      android:bottom="10dp" />
  </shape>
</item>
```

Gambar 4.15 Status tombol yang ditekan.



Gambar 4.16 Hasil running dari button selector.

Mendukung ukuran layar yang berbeda biasanya berarti bahwa sumber daya gambar Anda juga harus mampu beradaptasi dengan ukuran yang berbeda. Misalnya, latar belakang tombol harus sesuai dengan bentuk tombol mana pun yang diterapkan. Jika Anda menggunakan gambar sederhana pada komponen yang dapat mengubah ukuran, Anda akan segera melihat bahwa hasilnya agak kurang mengesankan, karena runtime akan meregangkan atau mengecilkan gambar Anda secara seragam. Solusinya adalah menggunakan bitmap sembilan tambalan, yang merupakan file PNG yang

diformat secara khusus yang menunjukkan area mana yang dapat dan tidak dapat diregangkan.

Oleh karena itu, ketika merancang bitmap yang akan digunakan pada komponen dengan ukuran variabel, selalu gunakan sembilan patch. Untuk mengonversi bitmap menjadi sembilan tambalan, Anda dapat memulai dengan gambar biasa. Kemudian jalankan melalui utilitas draw9patch dari Software Development Kit (SDK) (yang terletak di direktori alat/), di mana Anda dapat menandai area yang harus direntangkan dengan menggambar piksel di sepanjang batas kiri dan atas. Anda juga dapat menandai area yang harus menampung konten dengan menggambar piksel di sepanjang batas kanan dan bawah. Prosesnya ditunjukkan dari Gambar 4.17 sampai Gambar 4.18.



Gambar 4.17 Gambar asli (.png).

Piksel hitam berada di sepanjang perbatasan. Yang di batas atas dan kiri menunjukkan tempat di mana gambar dapat diregangkan, dan yang di batas kanan dan bawah menunjukkan di mana konten harus ditempatkan.



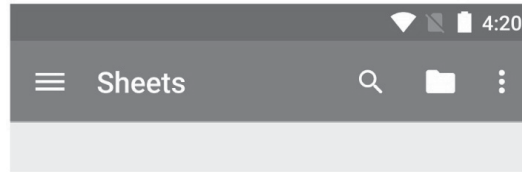
Gambar 4.18 Gambar sembilan tambalan (.9.png).



Gambar 4.19 Gambar sembilan tambalan yang digunakan dalam berbagai ukuran.

4.2.2 Bilah Tindakan

Bilah tindakan, juga disebut bilah aplikasi, adalah salah satu elemen desain terpenting dalam aktivitas. Ini menyediakan struktur visual dan elemen interaktif yang akrab bagi pengguna. Sebuah bar tindakan khas ditunjukkan pada Gambar. 4.20.



Gambar 4.20 Bilah tindakan tipikal.

Bilah tindakan memiliki beberapa fungsi utama yang tercantum sebagai berikut:

1. Ruang khusus untuk memberikan identitas pada aplikasi dan menunjukkan lokasi virtual pengguna di aplikasi.
2. Akses ke tindakan penting dengan cara yang dapat diprediksi, seperti pencarian.
3. Dukungan untuk navigasi dan pengalihan tampilan (dengan tab atau daftar drop-down).

Dalam bentuknya yang paling dasar, bilah tindakan menampilkan judul aktivitas di satu sisi dan menu limpahan di sisi lain. Mulai Android 3.0 (API level 11), semua aktivitas yang menggunakan tema default memiliki ActionBar sebagai bilah aplikasi. Namun, fitur bilah aplikasi secara bertahap telah ditambahkan ke ActionBar asli melalui berbagai rilis Android. Akibatnya, ActionBar asli berperilaku berbeda tergantung pada versi sistem Android yang mungkin digunakan perangkat. Sebaliknya, fitur terbaru ditambahkan ke versi pustaka dukungan Toolbar, dan tersedia di perangkat apa pun yang dapat menggunakan pustaka dukungan.

Karena alasan ini, Anda harus menggunakan kelas Toolbar pustaka dukungan untuk mengimplementasikan bilah aplikasi aktivitas Anda. Menggunakan bilah alat pustaka dukungan membantu memastikan bahwa aplikasi Anda akan memiliki perilaku yang konsisten di berbagai perangkat. Misalnya, widget Toolbar memberikan pengalaman desain material pada perangkat yang menjalankan Android 2.1 (API level 7) atau lebih baru, tetapi bilah tindakan asli tidak mendukung desain material kecuali perangkat menjalankan Android 5.0 (API level 21) atau lebih baru.

4.2.3 Tampilan Kustom

Android memiliki sekumpulan besar kelas tampilan untuk berinteraksi dengan pengguna dan menampilkan berbagai jenis data. Namun terkadang kami memiliki beberapa persyaratan unik yang tidak tercakup oleh tampilan bawaan. Untuk menjadi kelas yang dirancang dengan baik, tampilan kustom harus:

1. sesuai dengan standar Android;
2. memberikan atribut yang dapat disesuaikan gayanya yang berfungsi dengan tata letak XML Android;
3. mengirim acara aksesibilitas;
4. kompatibel dengan beberapa platform Android.

Semua kelas tampilan yang ditentukan dalam kerangka kerja Android memperluas Tampilan. Tampilan kustom juga dapat memperluas Tampilan secara langsung, atau kita dapat memperluas beberapa subkelas tampilan yang ada, seperti Tombol. Kemudian kita perlu mendefinisikan beberapa atribut untuk tampilan kustom.

Untuk menentukan atribut khusus, tambahkan `<declare-styleable>` sumber daya ke proyek kami. Merupakan kebiasaan untuk menempatkan sumber daya ini ke dalam file `res/values/attrs.xml`. Setelah tampilan dibuat dari tata letak XML, semua atribut dalam tag XML dibaca dari bundel sumber daya dan diteruskan ke konstruktor tampilan sebagai `AttributeSet`. Kemudian kita akan meneruskan `AttributeSet` ke `getStyledAttributes()`. Metode ini melewati kembali sebuah array `TypedArray` dari nilai-nilai yang telah di-dereference dan ditata.

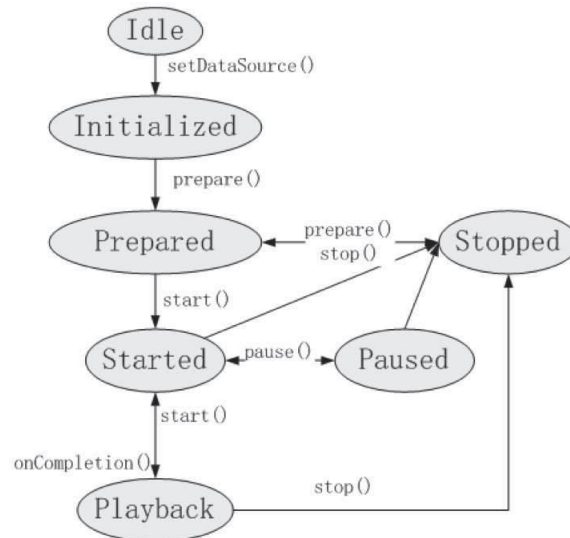
Kemudian kita perlu menambahkan properti dan acara ke tampilan kustom. Untuk memberikan perilaku dinamis, kita perlu mengekspos pasangan pengambil dan penyetal properti untuk setiap atribut khusus, misalnya, menampilkan teks dan gambar. Setelah membuat dan memulai tampilan kustom, kita beralih ke bagian terpenting dari tampilan kustom, yaitu tampilannya. Selanjutnya, langkah terpenting dalam menggambar tampilan kustom adalah mengganti metode `onDraw()`. Parameter ke `onDraw()` adalah objek `Canvas` yang dapat digunakan tampilan untuk menggambar dirinya sendiri. Kelas `Canvas` mendefinisikan metode untuk menggambar teks, garis, bitmap, dan banyak grafik primitif lainnya. Anda dapat menggunakan metode ini di `onDraw()` untuk membuat UI kustom Anda.

Menggambar UI hanyalah satu bagian dari membuat tampilan kustom. Anda juga perlu membuat tampilan Anda merespons masukan pengguna dengan cara yang sangat mirip dengan tindakan dunia nyata yang Anda tiru. Kita perlu membuat tampilan interaktif, termasuk gerakan input, gerakan yang masuk akal secara fisik, dan membuat transaksi menjadi lancar.

4.3 TINJAUAN MULTIMEDIA DI ANDROID

4.3.1 Memahami Kelas MediaPlayer

Android mendukung output audio dan video melalui kelas `MediaPlayer` dalam paket `android.media`. `android.media` digunakan untuk mengelola berbagai antarmuka media. `Media API` digunakan untuk memutar dan merekam file media, termasuk audio dan video. Kelas `MediaPlayer` dapat digunakan untuk mengontrol pemutaran file dan streaming audio/video. Kontrol file audio/video dan streaming diatur sebagai mesin negara, seperti ditunjukkan pada Gambar 4.21.



Gambar 4.21 Diagram Status MediaPlayer.

4.3.2 Siklus Hidup Status MediaPlayer

Siklus hidup dan status objek MediaPlayer didorong oleh operasi kontrol pemutaran yang didukung. Metode `setDataSource()` dipanggil untuk mentransfer objek MediaPlayer dari status idle ke status inialisasi. Objek MediaPlayer harus memasuki status yang disiapkan terlebih dahulu sebelum dimulai dan diputar ulang. MediaPlayer dapat memasuki status siap dengan memanggil metode persiapan() atau persiapanAsync(). Metode `prepare()` mentransfer objek ke status siap setelah pemanggilan metode dikembalikan. Metode `prepareAsync()` pertama-tama mentransfer objek ke status persiapan setelah panggilan kembali sementara mesin pemutar internal terus bekerja untuk menyelesaikan sisa pekerjaan persiapan. Metode `start()` harus dipanggil untuk memulai pemutaran. Objek MediaPlayer berada dalam status mulai, setelah `start()` kembali. Memanggil `start()` tidak berpengaruh pada objek MediaPlayer yang sudah dalam status mulai.

4.4 IMPLEMENTASI AUDIO DI ANDROID

Untuk mempelajari cara memutar audio, kami membuat proyek baru bernama "MediaTester" dan mempertahankan konfigurasi default lainnya. Kemudian kita salin satu lagu dari direktori lokal ke direktori "MediaTester\app\src\main\res\raw". Perhatikan bahwa kita perlu memastikan bahwa format file dapat dikenali oleh Android. Untungnya, Android mendukung hampir semua jenis format file audio. Namun, jika Android tidak mendukung format file audio Anda, coba ubah ke format umum. Pertama, buat dua tombol untuk menampilkan fungsi "mulai" dan "jeda". Seperti yang diperkenalkan pada bab sebelumnya, kita membuat dua tombol di `activity_main.xml`, seperti yang ditunjukkan pada Gambar 4.22. Sementara itu, kita perlu menambahkan dua string dalam file `strings.xml` sebagai:

```

< stringname = "start_button" > Start < /string >
< stringname = "pause_button" > Pause < /string >

```



```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/start_button"
    android:id="@+id/button_start"
    android:layout_below="@+id/textView"
    android:layout_alignParentStart="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/pause_button"
    android:id="@+id/button_pause"
    android:layout_below="@+id/button_start"
    android:layout_alignParentStart="true" />

```

Gambar 4.22 Membuat dua tombol di activity_main.xml.

Kemudian lompat ke file MainActivity.java dan tambahkan objek MediaPlayer baru ke dalam kelas MainActivity sebagai:

```
private MediaPlayer mp
```

Kemudian kita memodifikasi MainActivity untuk mengimplementasikan OnClickListener, seperti yang diperkenalkan di bab sebelumnya, dan membuat metode onClick(View v) untuk mengimplementasikan fungsi dari kedua tombol ini. Kemudian atur OnClickListener ke dua tombol ini, dan sekarang MainActivity.java ditampilkan seperti Gambar 4.23.

```

public class MainActivity extends Activity implements View.OnClickListener {

    private MediaPlayer mp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        findViewById(R.id.button_1).setOnClickListener(this);
        findViewById(R.id.button_2).setOnClickListener(this);
    }

    public void onClick(View v)
    {...}
}

```

Gambar 4.23 Objek MediaPlayer dan OnClickListener.

Kemudian pada metode onClick(), kita mengimplementasikan fungsi pada kedua tombol tersebut, yaitu memulai lagu, menjeda, dan melanjutkannya. Sebelum memulai lagu, kita perlu membuat sumber daya ke objek MediaPlayer. Kemudian kita perlu memberi tahu komputer audio mana yang ingin kita putar. Kita dapat menggunakan ID integer sumber daya audio untuk mengidentifikasi sumber daya audio. Dalam contoh kita, kita menggunakan resId = R.raw.test, kemudian kita memanggil metode start() untuk memutar musik. Sebelum menggunakan jeda lagu, kita perlu menilai apakah lagu itu sedang diputar. Jika sedang diputar, kami memanggil metode pause() untuk menghentikannya; jika tidak, kami memanggil metode start() untuk melanjutkannya. Kode tersebut ditunjukkan pada Gambar 4.24. Hasil running

ditunjukkan pada Gambar 4.25. Saat kita mengklik tombol “MULAI”, Android memainkan lagu yang sebelumnya kita masukkan ke dalam file mentah. Ketika kita mengklik tombol “PAUSE”, Android akan menjeda lagu jika sedang diputar atau melanjutkannya jika dijeda.

4.5 MENGEKSEKUSI VIDEO DI ANDROID

Kelas MediaPlayer bekerja dengan video seperti halnya dengan audio. Namun, kita perlu membuat permukaan untuk memutar video, dan permukaannya adalah kelas VideoView. Kelas VideoView dapat memuat gambar dari berbagai sumber dan mengambil alih komputasi pengukurannya dari video. Selain itu, ia menyediakan berbagai opsi tampilan, seperti penskalaan.

PETUNJUK: VideoView tidak mempertahankan status penuhnya saat masuk ke latar belakang. Itu tidak mengembalikan status pemutaran saat ini, posisi, trek yang dipilih, atau trek subtitle apa pun.

Kami akan menambahkan sesuatu tentang video ke dalam proyek MediaTester.

```
public void onClick(View v) {
    int resId;
    switch (v.getId()) {
        case R.id.button_start:
            resId = R.raw.test;
            if (mp != null) {
                mp.release();
            }
            mp = MediaPlayer.create(this, resId);
            mp.start();
            break;
        case R.id.button_pause:
            if (mp.isPlaying()) {
                mp.pause();
            } else {
                mp.start();
            }
            break;
    }
}
```

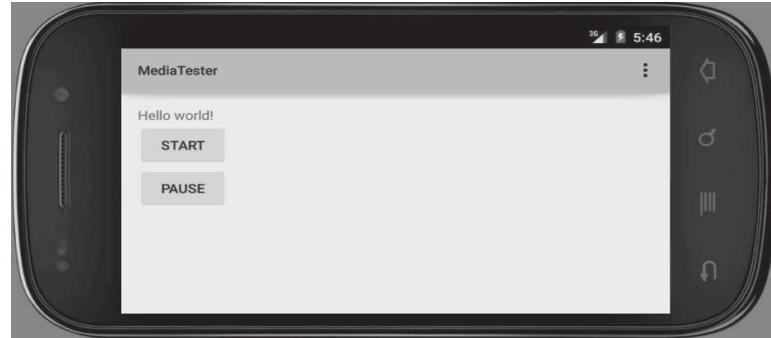
Gambar 4.24 Implementasi fungsi dua tombol pada metode onClick().

Pertama, kami membuat VideoView baru di bawah tombol jeda di activity_main.xml sebagai berikut:

```
<VideoView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button_pause"
    android:layout_gravity="center"/>
```

Kemudian, melompat ke file Java, kita membuat objek View bernama video dan menghubungkannya ke VideoView sebagai berikut:

```
VideoView video = (VideoView) findViewById(R.id.video);
```



Gambar 4.25 Hasil running dari MediaTester.

Kemudian kita perlu mengatur jalur untuk mengidentifikasi lokasi video. Namun, Android Virtual Device (AVD) tidak dapat mengenali jalur lokal di komputer kita. Android menawarkan beberapa opsi untuk menyimpan data aplikasi persisten sebagai berikut:

Preferensi Bersama Kelas `SharedPreferences` menyediakan kerangka kerja umum yang memungkinkan Anda untuk menyimpan dan mengambil pasangan kunci-nilai persisten dari tipe data primitif.

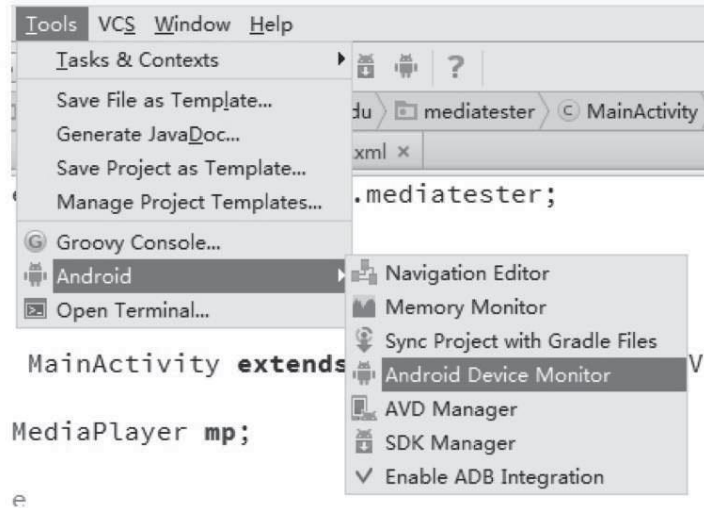
Penyimpanan Internal Kami dapat menyimpan file langsung di penyimpanan internal perangkat. File yang disimpan ke penyimpanan internal bersifat pribadi secara default.

Penyimpanan Eksternal Perangkat Android mendukung penyimpanan eksternal bersama untuk menyimpan file. Penyimpanan eksternal dapat berupa media penyimpanan yang dapat dilepas, seperti kartu SD, atau penyimpanan internal.

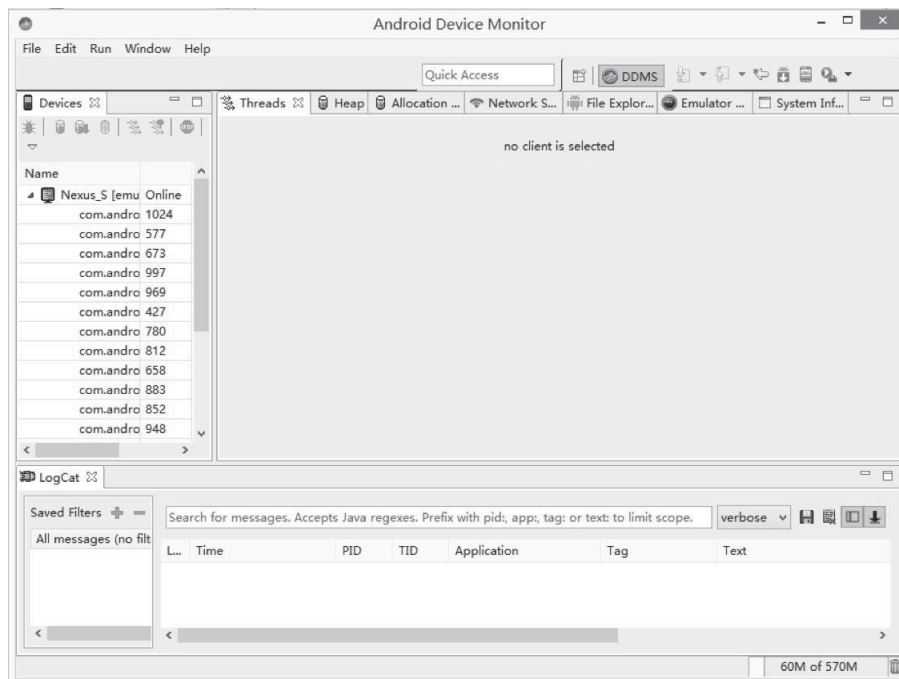
Database SQLite Kita dapat menggunakan SQLite di Android untuk membuat database yang dapat diakses dengan nama ke kelas mana pun di aplikasi.

Koneksi Jaringan Kami dapat menggunakan jaringan untuk menyimpan dan mengambil data dalam layanan kami sendiri.

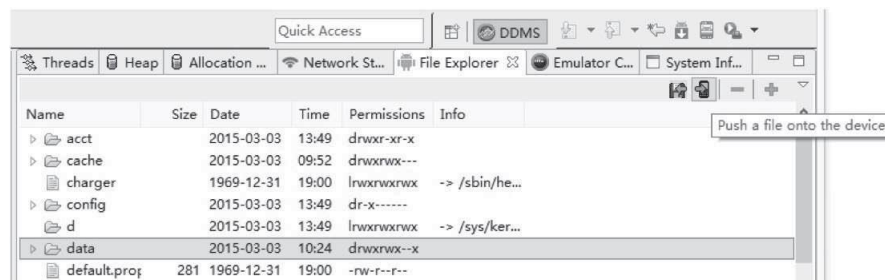
Sebelum kita memutar video menggunakan `VideoView`, kita perlu mengatur jalur untuk menemukan video, dan jalur ini harus berada di dalam AVD itu sendiri. Pertama-tama, jalankan AVD, dan masuk ke Dalvik Debug Monitor Service (DDMS) setelah AVD berjalan. Di Android Studio, pilih Tools, lalu Android, lalu klik Android Device Monitor (Tools → Android → Android Device Monitor), seperti terlihat pada Gambar 4.26. Monitor Perangkat Android akan mirip dengan Gambar 4.27. Kemudian pilih AVD yang baru saja kita jalankan, lalu pada tab "File Explorer", kita dapat melihat banyak folder dan file yang terdaftar. Temukan folder "data" dan klik "Push a file to the device" di kanan atas antarmuka, seperti yang ditunjukkan pada Gambar 4.28. Kemudian pilih dan dorong file video lokal ke perangkat.



Gambar 4.26 Monitor perangkat Android.



Gambar 4.27 Monitor perangkat Android.



Gambar 4.28 Dorong sebuah file ke perangkat.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    //audio part
    findViewById(R.id.button_start).setOnClickListener(this);
    findViewById(R.id.button_pause).setOnClickListener(this);

    //video part
    //Before we play a video, we need to push the video resource
    //into AVD
    VideoView video = (VideoView) findViewById(R.id.video);
    video.setVideoPath("/data/samplevideo.3gp");
    video.start();
}

```

Gambar 4.29 Mengatur jalur video dan memulai video.

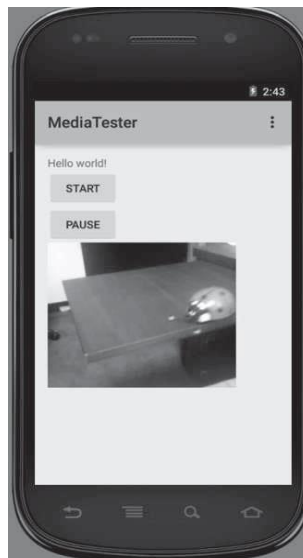
DDMS digunakan untuk mengoperasikan AVD, bukan aplikasi Android. Jika kami telah mendorong beberapa video ke perangkat sebelumnya, kami tidak perlu mendorongnya lagi di Proyek Android. Kemudian tambahkan dua metode ke metode onCreate() untuk mengatur jalur Video dan memutarinya sebagai berikut:

```

video.setVideoPath("/data/samplevideo.3gp");
video.start();

```

Kemudian metode onCreate() saat ini dapat merujuk pada Gambar 4.29. Pada akhirnya, hasil running ditunjukkan pada Gambar 4.30.



Gambar 4.30 Hasil running dari MediaTester.

4.6 LATIHAN

Bagian 1: Grafik 2-D

4.6.1 Latihan Dasar

1. Paket mana yang disediakan oleh Android dengan pustaka grafis 2D asli yang lengkap?
2. Bagaimana Android merepresentasikan warna?
3. Tipe data mana yang merupakan turunan dari kelas Warna?
4. Apa artinya alfa 0?

5. Apa itu Cat? Apa itu Canvas?
6. Apa parameter metode `onDraw()`?
7. Apa isi kelas `Path`?
8. Kelas mana yang dapat menawarkan beberapa efek mewah ke jalur?
9. Untuk apa kelas `Drawable` digunakan?
10. Di mana `Drawables` sering didefinisikan?
11. Di mana Anda mendaftarkan suatu kegiatan?
12. Bagaimana Anda mendapatkan ukuran tampilan yang sebenarnya?
13. Metode apa yang akan dipanggil oleh Android ketika ada bagian dari tampilan yang perlu diperbarui?
14. Apa perbedaan utama antara Android dan iOS saat menangani input?
15. Metode mana yang berisi operasi tentang `trackball`?
16. Bagaimana Anda menggunakan fungsi menggambar dalam metode `onDraw()`?
17. Metode mana yang menangani input keyboard?
18. Apa yang akan terjadi jika panggilan ke `invalidate()` tanpa parameter?
19. Apa itu papan tombol?

Bagian 2: Multimedia

1. Bagaimana Android mendukung keluaran suara dan musik?
2. Direktori mana yang menyimpan file suara?
3. Apa yang dilakukan metode `setVolumeControlStream()`?
4. Apa yang kita sebut metode `release()` di dalam metode `onKeyDown()`?
5. Apa yang akan terjadi jika kita lupa melepaskan sumber daya yang terkait dengan `MediaPlayer` lama?
6. Bagaimana cara kerja `MediaPlayer` Android?
7. Apa perbedaan antara `MediaPlayer` Android yang diputar di video dan audio?
8. Apa yang dilakukan metode `setVideoPath()`?
9. Bagaimana Anda membuat video untuk mengambil alih seluruh layar termasuk bilah judul dan bilah status?
10. Mengapa video diputar ulang saat memutar layar?
11. Apa yang dapat dilakukan metode `onRetainNonConfigurationInstance()` dalam suatu aktivitas?
12. Apa yang akan terjadi jika Anda menggunakan layanan untuk aplikasi musik ketika itu berakhir?
13. Apa hubungan antara metode `onPause()` dan metode `onResume()`?

4.6.2 Latihan Lanjutan

Bagian 1: Grafik 2-D

1. Gunakan Eclipse untuk memilih warna yang paling Anda sukai dan posting nilai ARGB (Red Green Blue Alpha).
2. Jelaskan proses pewarnaan di Android.

Bagian 2: Multimedia

1. Tolong jelaskan perbedaan antara menggunakan beberapa musik sebagai aktivitas dan menggunakannya sebagai layanan.

BAGIAN II

PENGOPTIMALAN APLIKASI SELULER TINGKAT LANJUT

BAB 5

ARSITEKTUR SISTEM EMBEDDED SELULER

Perangkat seluler adalah bagian tak terpisahkan dari sistem seluler, dan semua chip yang digunakan dalam perangkat seluler adalah sistem tertanam. Sistem tertanam dengan berbagai fungsi ini dikendalikan oleh sistem operasi seluler dan berkolaborasi satu sama lain untuk menyelesaikan setiap tugas yang diminta aplikasi seluler.

Dalam bab ini, kami memperkenalkan arsitektur sistem tertanam seluler, termasuk:

- Tinjauan sistem tertanam
- Aplikasi sistem tertanam.
- Teknologi prosesor dalam sistem tertanam
- Konsep dasar dalam teknologi prosesor dalam sistem tertanam
- Algoritma penjadwalan dalam teknologi prosesor dalam sistem tertanam
- Teknologi memori dalam sistem tertanam
- Sistem tertanam di perangkat seluler
- Sistem tertanam di Android

5.1 SISTEM TERTANAM (EMBEDDED)

Ikhtisar Sistem Tertanam

Sistem tertanam adalah segala sesuatu yang menggunakan mikroprosesor tetapi bukan komputer tujuan umum. Sistem tertanam adalah sistem komputer dengan fungsi khusus, yang tertanam sebagai bagian dari perangkat lengkap, termasuk perangkat keras dan bagian mekanis. Sistem kecil ini dapat ditemukan di mana-mana, mulai dari elektronik komersial, seperti ponsel, kamera, sistem pemantauan kesehatan portabel, pengontrol mobil, robot, dan perangkat keamanan pintar, hingga infrastruktur kritis, seperti jaringan telekomunikasi, jaringan tenaga listrik, lembaga keuangan, dan pembangkit nuklir. Sistem tertanam yang semakin rumit memerlukan otomatisasi desain yang ekstensif dan alat pengoptimalan.

Sistem tertanam modern sering didasarkan pada mikrokontroler, seperti Central Processing Unit (CPU) dengan memori terintegrasi atau antarmuka perifer, tetapi mikroprosesor biasa, yang menggunakan chip eksternal untuk memori, dan sirkuit antarmuka perifer masih umum. Sistem tertanam biasanya digunakan dalam sistem telekomunikasi, elektronik konsumen, sistem transportasi, dan peralatan medis.

Sistem Telekomunikasi

Sistem telekomunikasi menggunakan banyak sistem tertanam, mulai dari sakelar telepon hingga telepon seluler.

Elektronik Konsumen

Elektronik konsumen termasuk personal digital assistant (PDA), seperti pemutar audio, ponsel, konsol videogame, kamera digital, pemutar video, dan printer. Sistem tertanam digunakan untuk memberikan fleksibilitas, efisiensi, dan fitur.

Otomatisasi Rumah

Perangkat tertanam digunakan untuk merasakan dan mengontrol otomatisasi di rumah menggunakan jaringan kabel dan nirkabel. Perangkat tertanam dapat digunakan untuk mengontrol lampu, iklim, keamanan, audio/visual, dan pengawasan.

Sistem Transportasi

Sistem tertanam semakin banyak digunakan dari penerbangan ke mobil dalam sistem transportasi. Pesawat baru berisi avionik canggih, seperti penerima Inertial Guidance Systems (IGS) dan Global Positioning System (GPS) yang juga memiliki persyaratan keselamatan yang cukup besar. Berbagai motor listrik menggunakan pengontrol motor listrik. Mobil, kendaraan listrik, dan kendaraan hibrida semakin banyak menggunakan sistem tertanam untuk memaksimalkan efisiensi dan mengurangi polusi.

Peralatan medis

Peralatan medis menggunakan sistem tertanam untuk pemantauan tanda-tanda vital, stetoskop elektronik untuk memperkuat suara, dan berbagai pencitraan medis untuk inspeksi internal non-invasif. Sistem tertanam dalam peralatan medis sering kali ditenagai oleh komputer industri.

Selain penggunaan yang disebutkan di atas, sistem tertanam juga banyak digunakan dalam jenis teknologi baru, yaitu jaringan sensor nirkabel (WSN). WSN terdiri dari sensor otonom yang terdistribusi secara spasial untuk memantau kondisi fisik atau lingkungan. Parameter yang biasanya dipantau adalah suhu, kelembaban, tekanan, arah dan kecepatan angin, intensitas penerangan, intensitas getaran, intensitas suara, tegangan saluran listrik, konsentrasi kimia, tingkat polutan, dan fungsi vital tubuh. WSN memungkinkan orang dan perusahaan untuk mengukur banyak sekali hal di dunia fisik dan bertindak berdasarkan informasi ini di bawah bantuan sistem Wi-Fi tertanam. Selanjutnya, sensor nirkabel jaringan, menggunakan teknologi optimasi sistem tertanam, sepenuhnya mandiri dan biasanya akan kehabisan sumber baterai selama bertahun-tahun sebelum baterai perlu diganti atau diisi.

Sistem tertanam juga dapat didefinisikan sebagai komputer yang dibeli sebagai bagian dari beberapa peralatan lain. Mereka selalu memiliki perangkat lunak khusus di dalamnya, dan perangkat lunak tersebut dapat disesuaikan untuk pengguna. Seringkali tidak ada "keyboard" dan tampilan terbatas atau tidak ada tampilan tujuan umum dalam sistem tertanam. Sistem tertanam penting karena tiga jenis alasan:

Alasan rekayasa. Setiap perangkat yang perlu dikendalikan dapat dikendalikan oleh mikroprosesor. Dalam banyak situasi, tidak mungkin atau tidak perlu bagi perangkat untuk menjadi komputer yang lengkap. Terminal POS (Point of Sale) McDonald hanya bertugas mencatat pembelian, menghitung dan menunjukkan harga, mengumpulkan uang, memberikan kembalian, dan mencetak kwitansi. Fungsi semacam ini sederhana, sehingga terminal POS memiliki sedikit sumber daya penghitungan. Terminal POS tidak perlu lengkap seperti komputer pada umumnya, karena benar-benar mubazir.

Alasan pasar. Pasar komputer tujuan umum bernilai miliaran dolar; sementara itu pasar sistem tertanam juga bernilai miliaran dolar. Meskipun harga sistem tertanam mungkin jauh lebih rendah daripada komputer tujuan umum, jumlah sistem tertanam jauh lebih besar daripada komputer tujuan umum. Pada tahun 2009, sekitar 200 sistem tertanam digunakan di setiap mobil baru. Ada lebih dari 80 juta komputer pribadi terjual setiap tahun. Sementara lebih dari 3 miliar CPU tertanam terjual setiap tahun. Selain itu, pasar komputer pribadi sebagian besar sudah jenuh, tetapi pasar tertanam masih terus berkembang.

Alasan pedagogis. Perancang sistem tertanam sering kali perlu mengetahui perangkat keras, perangkat lunak, dan beberapa kombinasi jaringan, teori kontrol, dan pemrosesan sinyal. Hal ini membuat metode pengajaran merancang sistem tertanam berbeda dari metode untuk merancang sistem tujuan umum.

Pada bagian ini, kami memperkenalkan gambaran umum dari sistem tertanam, menganalisis penggunaannya, menjelaskan pentingnya mereka, membuat daftar beberapa aplikasi nyata dari sistem tertanam, dan memberikan desain tingkat tinggi dari sistem tertanam. Kami memperkenalkan pengetahuan yang lebih dalam setelah memperkenalkan desain sistem tertanam. Hal pertama dan terpenting adalah penjadwalan.

5.2 ALGORITMA PENJADWALAN

5.2.1 Konsep Dasar

Pertama-tama, beberapa konsep dasar harus diperkenalkan dan dijelaskan. Penjadwalan adalah pusat desain sistem operasi. Keberhasilan penjadwalan CPU tergantung pada dua eksekusi. Yang pertama adalah eksekusi proses yang terdiri dari siklus eksekusi CPU dan menunggu Input/Output (I/O). Yang kedua adalah eksekusi proses, yang dimulai dengan pemrosesan CPU, diikuti oleh pemrosesan I/O, kemudian diikuti oleh pemrosesan CPU yang lain, kemudian pemrosesan I/O lainnya, dan seterusnya. Siklus Pemrosesan I/O CPU adalah konsep dasar teknologi prosesor. Waktu pemrosesan adalah waktu aktual yang diperlukan untuk menyelesaikan beberapa pekerjaan.

Penjadwal CPU memilih dari antara proses dalam memori yang siap dieksekusi, dan mengalokasikan CPU ke salah satunya. Keputusan penjadwalan CPU terjadi ketika suatu proses beralih dari status berjalan ke status menunggu; beralih dari menjalankan ke status siap; beralih dari menunggu ke siap dan mengakhiri.

Selain CPU scheduler, dispatcher juga merupakan konsep dasar dan penting dalam teknologi prosesor. Modul dispatcher memberikan kontrol CPU ke proses yang dipilih oleh penjadwal jangka pendek, dan ini melibatkan: beralih konteks, beralih ke mode pengguna, dan melompat ke lokasi yang tepat di program pengguna untuk memulai kembali program itu. Sebagian besar operator memiliki latensi pengiriman, yaitu waktu yang dibutuhkan operator untuk menghentikan satu proses dan memulai proses lainnya. Kemudian kita membahas beberapa kriteria penjadwalan.

Utilisasi CPU. Utilisasi CPU mengacu pada penggunaan sumber daya pemrosesan komputer, atau jumlah pekerjaan yang ditangani oleh CPU, dan digunakan untuk

mengukur kinerja sistem. Utilisasi CPU yang sebenarnya bervariasi tergantung pada jumlah dan jenis tugas komputasi yang dikelola. Tujuan pertama dari teknologi prosesor adalah meningkatkan penggunaan CPU dengan menjaga agar CPU tetap sesibuk mungkin.

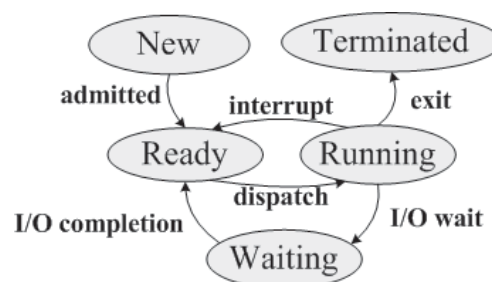
Throughput. Throughput berarti jumlah proses yang menyelesaikan eksekusi mereka per unit waktu.

Waktu penyelesaian. Waktu penyelesaian berarti jumlah waktu untuk mengeksekusi proses tertentu, dan dapat dihitung sebagai jumlah waktu menunggu untuk masuk ke memori, menunggu dalam antrian siap, dan mengeksekusi pada CPU dan I/O.

Waktu menunggu. Waktu tunggu berarti jumlah waktu suatu proses telah menunggu dalam antrian siap.

Waktu merespon. Waktu respons berarti jumlah waktu yang diperlukan sejak permintaan diajukan hingga respons pertama dihasilkan.

Waktu penyelesaian. Waktu penyelesaian satu pekerjaan berarti jumlah waktu yang dibutuhkan untuk menyelesaikannya, jika tidak pernah mendahului, disela, atau dihentikan.



Gambar 5.1 Diagram status proses.

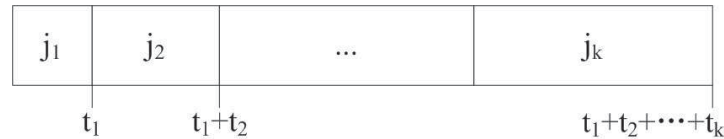
Seperti ditunjukkan pada Gambar. 5.1, proses memiliki lima jenis keadaan. Pada keadaan baru, proses berada pada tahap penciptaan. Pada status siap, proses memiliki semua sumber daya yang tersedia untuk dijalankan, tetapi CPU saat ini tidak mengerjakan instruksi proses ini. Pada status berjalan, CPU mengerjakan instruksi proses ini. Pada keadaan menunggu, proses tidak dapat berjalan saat ini, karena menunggu beberapa sumber daya tersedia atau beberapa peristiwa terjadi. Pada keadaan terminasi, proses telah selesai.

5.2.2 Algoritma Penjadwalan First Come, First Served

Indikator penting yang dapat diukur dari prosesor adalah waktu penyelesaian rata-rata pekerjaan. Gambar 5.2 merupakan contoh jadwal untuk k pekerjaan. Seperti yang ditunjukkan pada gambar, ada k pekerjaan, ditandai sebagai j_k , yang harus diselesaikan dalam prosesor. Pekerjaan pertama j_1 membutuhkan t_1 unit waktu sehingga pekerjaan j_1 dapat diselesaikan pada waktu t_1 . Pekerjaan kedua j_2 dimulai setelah pekerjaan pertama j_1 selesai, dan lama waktu yang dibutuhkan adalah t_2 . Oleh karena itu, pekerjaan kedua j_2 dapat diselesaikan pada waktu $t_1 + t_2$. Ulangi prosedur ini sampai pekerjaan terakhir j_k selesai.

Total waktu penyelesaian:

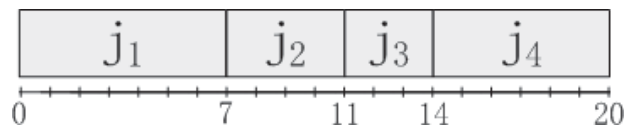
$$\begin{aligned} A &= t_1 + (t_1 + t_2) + (t_1 + t_2 + t_3) + \dots + (t_1 + t_2 + t_3 + \dots + t_k) \\ &= k * t_1 + (k - 1) * t_2 + (k - 2) * t_3 + \dots + t_k \end{aligned} \quad (5.1)$$



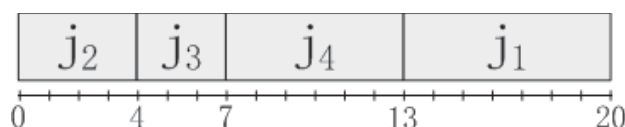
Gambar 5.2 Jadwal untuk k pekerjaan.

Salah satu algoritma penjadwalan yang paling sederhana adalah First Come, First Served (FCFS). Kebijakan FCFS banyak digunakan dalam kehidupan sehari-hari. Misalnya, ini adalah kebijakan standar untuk pemrosesan sebagian besar antrian, di mana orang menunggu layanan yang tidak diatur sebelumnya atau direncanakan sebelumnya. Di bidang teknologi prosesor, itu berarti pekerjaan ditangani sesuai pesanan.

Misalnya ada empat job, j_1 , j_2 , j_3 , dan j_4 , dengan waktu proses yang berbeda, yaitu masing-masing 7, 4, 3, dan 6. Pekerjaan ini tiba dalam urutan: j_1 , j_2 , j_3 , j_4 . Dalam kebijakan FCFS, mereka ditangani dengan urutan j_1 , j_2 , j_3 , j_4 , seperti yang ditunjukkan pada Gambar 5.3. Waktu tunggu untuk j_1 adalah 0, untuk j_2 adalah 7, untuk j_3 adalah 11, dan untuk j_4 adalah 14. Waktu tunggu rata-rata adalah $(0+7+11+14)/4 = 8$. Waktu tunggu rata-rata adalah $[7 + (7+4) + (7+4+3) + (7+4+3+6)] / 4 = 13$. Misalkan pekerjaan tiba dalam urutan j_2 , j_3 , j_4 , j_1 ; hasil yang dihasilkan dengan menggunakan FCFS ditunjukkan pada Gambar 5.4. Waktu tunggu untuk j_1 adalah 13, untuk j_2 adalah 0, untuk j_3 adalah 4, dan untuk j_4 adalah 7. Waktu tunggu rata-rata adalah $(13+0+4+7)/4 = 6$. Waktu rata-rata penyelesaian adalah $[4 + (4+3) + (4+3+6) + (4+3+6+7)] / 4 = 11$. Waktu tunggu rata-rata dan waktu penyelesaian rata-rata penjadwalan ini lebih kecil dari yang sebelumnya.



Gambar 5.3 Contoh penjadwalan FCFS.



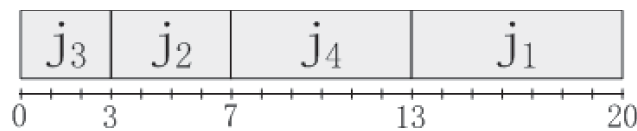
Gambar 5.4 Hasil FCFS lain jika mengubah urutan kedatangan.

5.2.3 Algoritma Penjadwalan Pekerjaan Pertama yang Dipendekkan

Kemudian kami akan memperkenalkan kebijakan penjadwalan lainnya, yaitu Shortest Job First (SJF). SJF adalah kebijakan penjadwalan yang memilih proses

menunggu dengan waktu eksekusi terkecil untuk dieksekusi terlebih dahulu. SJF menguntungkan karena kesederhanaannya, dan meminimalkan waktu penyelesaian rata-rata. Setiap proses harus menunggu sampai eksekusinya selesai.

Menggunakan contoh yang disebutkan di Bagian 2.2, sambil mengabaikan waktu kedatangannya, pertama-tama kita mengurutkan pekerjaan ini berdasarkan waktu pemrosesannya, sebagai j_3, j_2, j_4, j_1 . Hasil penjadwalan SJF ditunjukkan pada Gambar 5.5. Waktu tunggu untuk j_1 adalah 13, j_2 adalah 3, j_3 adalah 0, dan j_4 adalah 7. Waktu tunggu rata-rata adalah $(13+3+0+7)/4 = 5,75$. Waktu penyelesaian untuk j_1 adalah $(13+7)$, j_2 adalah $(3+4)$, j_3 adalah $(0+3)$, j_4 adalah $(7+6)$. Waktu penyelesaian rata-rata adalah $(20+7+3+13)/4 = 10,75$. Penjadwalan ini memiliki rata-rata waktu tunggu dan waktu penyelesaian rata-rata yang lebih rendah dibandingkan dua jadwal sebelumnya.



Gambar 5.5 Contoh penjadwalan SJF.

Teorema: Penjadwalan SJF memiliki total waktu penyelesaian terendah dengan prosesor tunggal.

Bukti dengan kontradiksi: Dengan asumsi bahwa ada serangkaian pekerjaan yang diurutkan berdasarkan waktu penyelesaiannya dari pendek ke panjang, seperti $j_1, j_2, j_3, \dots, j_i, j_{i+1}, \dots, j_k$, yang juga berarti waktu penyelesaiannya dapat diurutkan sebagai $t_1 < t_2 < t_3 < \dots < t_i < t_{i+1} < \dots < t_k$. Menggunakan algoritma penjadwalan SJF, hasilnya sama persis dengan orde $j_1, j_2, j_3, \dots, j_i, j_{i+1}, \dots, j_k$. Kemudian kita misalkan ada orde A lain yang memiliki total waktu penyelesaian lebih rendah dari yang dihasilkan oleh SJF, $j_1, j_2, j_3, \dots, j_{i+1}, j_i, \dots, j_k$. Berdasarkan Persamaan 5.1, total waktu penyelesaian adalah $T = k * t_1 + (k - 1) * t_2 + (k - 2) * t_3 + \dots + (k - i + 1) * t_i + (k - i) * t_{i+1} + \dots + t_k$. Jadi, kita bisa mendapatkan total waktu penyelesaian kedua pesanan. Yang SJF adalah $T_s = k * t_1 + (k - 1) * t_2 + (k - 2) * t_3 + \dots + (k - i + 1) * t_i + (k - i) * t_{i+1} + \dots + t_k$. Yang A adalah $T_a = k * t_1 + (k - 1) * t_2 + (k - 2) * t_3 + \dots + (k - i + 1) * t_{i+1} + (k - i) * t_i + \dots + t_k$. Dari kondisi pengandaian, $T_s < T_a$.

$$T_s > T_a;$$

$$k * t_1 + (k - 1) * t_2 + (k - 2) * t_3 + \dots + (k - i + 1) * t_i + (k - i) * t_{i+1} + \dots + t_k$$

$$> k * t_1 + (k - 1) * t_2 + (k - 2) * t_3 + \dots + (k - i + 1) * t_{i+1} + (k - i) * t_i + \dots + t_k;$$

$$(k - i + 1) * t_i + (k - i) * t_{i+1} > (k - i + 1) * t_{i+1} + (k - i) * t_i;$$

$$t_i > t_{i+1}.$$

Namun, $t_i > t_{i+1}$ bertentangan dengan $t_i < t_{i+1}$, dalam kondisi asumsi. Akibatnya, A tidak ada, yang berarti tidak ada solusi yang memiliki total waktu penyelesaian lebih rendah dari penjadwalan SJF. Pada akhirnya, kita dapat menyimpulkan bahwa penjadwalan SJF memiliki waktu tunggu rata-rata terendah dengan prosesor tunggal. Namun, apakah SJF masih optimal dengan banyak prosesor?

5.2.4 Multiprosesor

Setelah membahas prosesor tunggal, kami akan memperluas topik menjadi multiprosesor. Ada sembilan pekerjaan dengan waktu penyelesaian yang berbeda dalam tiga prosesor, seperti yang ditunjukkan pada Gambar. 5.6, dan pertama-tama kami memberikan jadwal optimal menggunakan SJF. Waktu penyelesaian rata-rata adalah

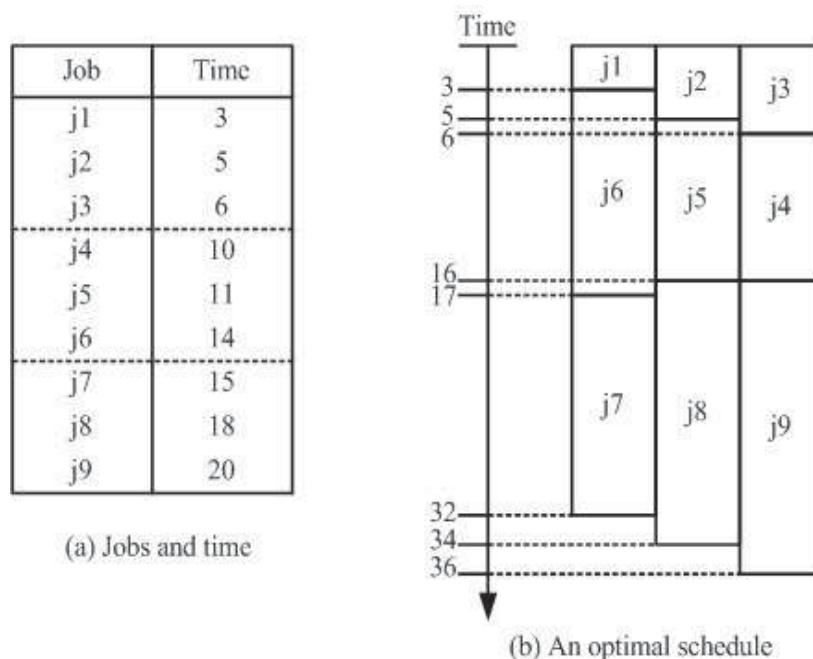
$$\{(3+5+6) + [(6+10)+(5+11)+(3+14)] + [(3+14+15)+(5+11+18) + (6+10+20)]\} / 9 = 18,33.$$

Ada jadwal optimal lain, seperti yang ditunjukkan pada Gambar. 5.7. Waktu penyelesaian rata-rata adalah

$$\{(3+5+6) + [(5+10)+(3+11)+(6+14)] + [(5+10+15)+(6+14+18) + (3+11+20)]\} / 9 = 18,33.$$

Dalam multiprosesor, ada tiga teorema:

1. **Teorema 5.1** Penjadwalan SJF memiliki rata-rata waktu tunggu dan waktu penyelesaian yang optimal dalam multiprosesor.
2. **Teorema 5.2** Dengan rata-rata waktu tunggu yang sama, terdapat lebih dari satu jadwal dengan waktu penyelesaian akhir yang bervariasi.
3. **Teorema 5.3** Algoritma untuk mencari waktu penyelesaian akhir yang optimal adalah NP-Hard.



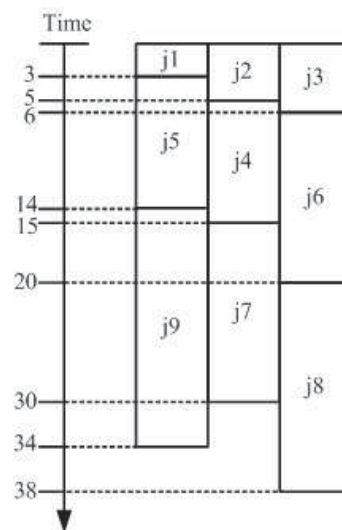
Gambar 5.6 Jadwal SJF untuk menyelesaikan sembilan pekerjaan dalam tiga prosesor.

Dengan asumsi bahwa waktu pemrosesan j_1 hingga j_{3k} adalah t_1 hingga t_{3k} , masing-masing, waktu penyelesaian rata-rata dalam tiga prosesor dihitung sebagai Persamaan 5.2: Waktu penyelesaian rata-rata adalah

$$\begin{aligned} & \{(t_1 + t_2 + t_3) + (t_1 + t_2 + t_3 + t_4 + t_5 + t_6) + \dots + (t_1 + t_2 + \dots + t_{3k})\} / 3k \\ & = \{k(t_1 + t_2 + t_3) + (k-1)(t_4 + t_5 + t_6) + \dots + (t_{3k-2} + t_{3k-1} + t_{3k})\} / 3k. \end{aligned} \quad (5.2)$$

Kemudian kami menetapkan bahwa $T_1 = t_1 + t_2 + t_3$, $T_2 = t_4 + t_5 + t_6, \dots$,

$T_k = t_{3k-2} + t_{3k-1} + t_{3k}$. Total waktu penyelesaian dalam tiga prosesor dapat dirumuskan sebagai $kT_1 + (k-1)T_2 + \dots + T_k$. Akhirnya, kita bisa merumuskan masalah ini ke dalam satu prosesor. Pada akhirnya, kita dapat menggunakan metode yang sama seperti pada Bagian 2.3 untuk membuktikan bahwa jadwal SJF memiliki waktu penyelesaian rata-rata yang optimal dalam multiprosesor. Dari Persamaan 5.2, kita dapat melihat bahwa urutan rinci dari j_1, j_2, j_3 tidak mempengaruhi waktu tunggu rata-rata dari seluruh jadwal. Akibatnya, dua jadwal pada Gambar 5.6 dan Gambar 5.7 memiliki waktu tunggu rata-rata yang sama.



Gambar 5.7 Jadwal lain untuk menyelesaikan sembilan pekerjaan di tiga prosesor.

Namun waktu penyelesaian pekerjaan terakhir kedua jadwal ini berbeda, yaitu 36 dan 38. Jika ada kendala waktu kurang dari 38, jadwal kedua tidak sesuai, sedangkan jadwal pertama dapat dipilih. Selain itu, masih banyak jadwal lain yang rata-rata waktu tunggunya sama dengan kedua jadwal tersebut, karena mengubah urutan $j_{3i+1}, j_{3i+2}, j_{3i}$ tidak mengubah rata-rata waktu tunggu. Namun demikian, waktu penyelesaian pekerjaan terakhir berbeda-beda, dan cara menemukan jadwal optimal yang memiliki waktu paling singkat saat pekerjaan terakhir diselesaikan terlalu sulit untuk diselesaikan dengan algoritma normal. Masalah ini adalah masalah khas NP-Hard, dan kita akan membahas masalah ini dan bagaimana menyelesaikannya di bab-bab selanjutnya.

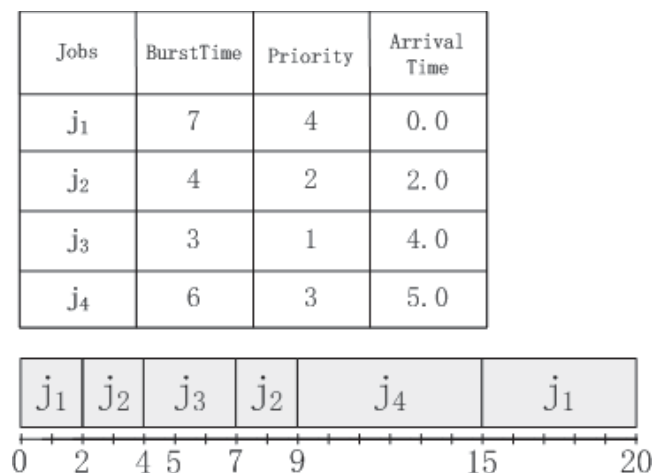
Algoritma Penjadwalan Prioritas

Algoritma penjadwalan selanjutnya adalah algoritma Priority Scheduling. Dalam penjadwalan prioritas, nomor prioritas, yang dapat berupa bilangan bulat, diasosiasikan dengan setiap proses. CPU dialokasikan ke pekerjaan dengan prioritas tertinggi, dan bilangan bulat terkecil mewakili prioritas tertinggi. Penjadwalan prioritas dapat digunakan dalam skema preemptive dan nonpreemptive. Penjadwalan SJF adalah penjadwalan prioritas, di mana prioritas adalah perkiraan waktu pemrosesan CPU berikutnya. Berikut ini adalah contoh yang diberikan tentang implementasi penjadwalan prioritas dalam skema preemptive, seperti yang ditunjukkan pada

Gambar 5.8. Prioritas setiap pekerjaan berbanding terbalik dengan waktu pemrosesannya. Akibatnya, hasil menggunakan algoritma penjadwalan prioritas sama dengan hasil dari penjadwalan SJF.

Penjadwalan prioritas memiliki batasan potensial yang berasal dari kelaparan proses. Process Starvation adalah proses yang membutuhkan waktu penyelesaian yang lama, sedangkan proses yang membutuhkan waktu penyelesaian yang lebih pendek terus ditambahkan. Skema "Penuaan" digunakan untuk mengatasi masalah ini. Seiring berjalannya waktu, prioritas proses meningkat. Kerugian lain adalah bahwa total waktu eksekusi suatu pekerjaan harus diketahui sebelum eksekusi. Meskipun tidak mungkin untuk memprediksi waktu eksekusi secara tepat, beberapa metode dapat digunakan untuk memperkirakan waktu eksekusi untuk suatu pekerjaan, seperti rata-rata tertimbang dari waktu eksekusi sebelumnya.

Terakhir, kami akan memperkenalkan penjadwalan Round Robin (RR). Dalam penjadwalan RR, setiap pekerjaan mendapat unit kecil waktu CPU, yang disebut imequantum, biasanya 10 - 100 milidetik. Setelah waktu ini berlalu, pekerjaan didahulukan dan ditambahkan ke akhir antrian siap. Jika ada n pekerjaan dalam antrian siap dan kuantum waktunya adalah q , maka setiap pekerjaan mendapat $1/n$ waktu CPU dalam potongan paling banyak q unit waktu sekaligus. Tidak ada pekerjaan yang menunggu lebih dari $(n - 1)$ unit waktu. Jika q besar, penjadwalan RR akan menjadi penjadwalan FCFS. Namun demikian, jika q kecil, overhead mungkin terlalu tinggi karena terlalu sering beralih konteks.



Gambar 5.8 Contoh penjadwalan prioritas.

Sebenarnya ada dua macam skema penjadwalan yaitu non-preemptive dan preemptive.

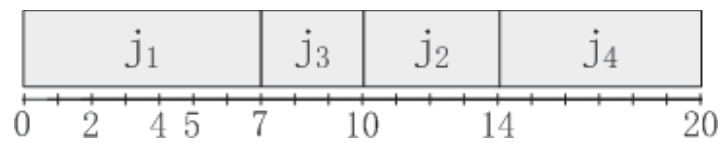
Nonpreemptive.

Penjadwalan nonpreemptive berarti bahwa setelah CPU dialokasikan ke suatu proses, proses tersebut menyimpan sumber daya CPU sampai ia melepaskan CPU baik dengan menghentikan atau beralih ke status menunggu.

Preemptive.

Dalam skema preemptive, pekerjaan baru dapat mendahului sumber daya CPU, jika panjang pemrosesan CPU-nya kurang dari waktu yang tersisa dari pekerjaan yang dijalankan saat ini. Skema ini dikenal sebagai *Shortest-Remaining-Time-First* (SRTF).

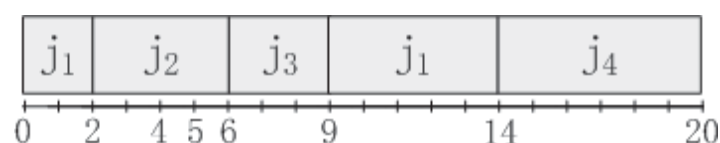
Dalam ilmu komputer, preemption adalah tindakan menghentikan sementara pekerjaan yang sedang dilakukan oleh komputer. Biasanya dilakukan oleh pekerjaan istimewa pada sistem yang memungkinkan interupsi. Gambar 5.5 menunjukkan penjadwalan SJF dalam situasi ketika semua pekerjaan tiba pada waktu yang sama, tetapi situasi akan menjadi rumit ketika mempertimbangkan waktu kedatangan yang berbeda, terutama dalam skema preemptive.



Gambar 5.9 Contoh solusi SJF nonpreemptive.

Masih mengambil contoh yang disebutkan dalam Bagian 5.2.2, tambahkan waktu kedatangannya, j_1 tiba pada waktu 0.0; j_2 tiba pada waktu 2.0; j_3 tiba pada waktu 4.0; j_4 tiba pada waktu 5.0. Penjadwalan SJF dalam skema nonpreemptive ditunjukkan pada Gambar. 5.9. Pada saat 0, j_1 tiba, dan tidak ada pekerjaan lain yang bersaing dengannya, jadi j_1 ada dalam daftar yang sedang berjalan. Pada saat 2, 4, dan 5, j_2 , j_3 , dan j_4 masing-masing tiba.

Namun, mereka tidak dapat menginterupsi j_1 dan mengambil sumber daya yang digunakan j_1 , jadi mereka semua ada dalam daftar tunggu. Pada waktu 7, j_1 selesai, dan sekarang ada tiga pekerjaan dalam daftar tunggu. Di antara ketiga pekerjaan ini, j_3 membutuhkan waktu pemrosesan tersingkat, sehingga mendapatkan sumber daya dan berubah menjadi daftar yang berjalan. Pada waktu 10, j_3 selesai, dan sekarang ada dua pekerjaan dalam daftar tunggu, yaitu j_2 dan j_4 . Karena j_2 membutuhkan waktu pemrosesan yang lebih pendek daripada j_4 , j_2 mendapatkan sumber daya dan berubah menjadi daftar yang sedang berjalan. Pada waktu 14, j_2 selesai, dan sekarang hanya ada satu pekerjaan dalam daftar tunggu, yaitu j_4 . Jadi j_4 mendapatkan sumber daya dan berubah menjadi daftar yang sedang berjalan. Akhirnya, j_4 selesai pada waktu 20. Dalam penjadwalan ini, waktu tunggu untuk j_1 adalah 0, j_2 adalah (10-2), j_3 adalah (7-4), dan j_4 adalah (14-5). Waktu tunggu rata-rata $(0+8+3+9)/4 = 5$. Waktu penyelesaian j_1 adalah 7, j_2 (14-2), j_3 (10-4), dan j_4 (20-5). Waktu penyelesaian rata-rata adalah $(7+12+6+15)/4 = 10$.



Gambar 5.10 Contoh solusi SJF preemptive.

Penjadwalan SJF dalam skema preemptive ditunjukkan pada Gambar. 5.10. Pada saat 0, j_1 tiba, dan tidak ada pekerjaan lain yang bersaing dengannya, jadi j_1 ada dalam daftar yang sedang berjalan. Pada waktu 2, j_2 tiba, dan j_2 memiliki waktu pemrosesan yang lebih pendek dari j_1 , sehingga mendahului j_1 . j_1 masuk ke daftar tunggu, sementara j_2 di daftar berjalan. Pada waktu 4, j_3 tiba. j_3 membutuhkan 3 waktu untuk diselesaikan, sedangkan j_2 membutuhkan 2 waktu. Jadi j_3 tidak dapat mendahului j_2 dan tetap dalam daftar tunggu. Pada tahap saat ini, j_1 dan j_3 keduanya dalam daftar tunggu.

Selanjutnya, pada waktu 5, j_4 tiba, tetapi memiliki waktu pemrosesan lebih lama dari j_2 , sehingga tidak dapat mendahului j_2 . j_4 bergabung dalam daftar tunggu. Pada waktu 6, j_2 selesai, dan sekarang ada tiga pekerjaan dalam daftar tunggu. Diantaranya, j_3 membutuhkan waktu pemrosesan terpendek, jadi j_3 mendapatkan sumber daya, sementara yang lain masih menunggu. Pada waktu 9, j_3 selesai, dan sekarang ada dua pekerjaan dalam daftar tunggu. Karena j_1 membutuhkan waktu pemrosesan yang lebih pendek, yaitu 5, daripada j_4 , yaitu 6. j_1 mendapatkan sumber daya dan berubah menjadi daftar yang berjalan. Pada waktu 14, j_1 selesai, dan sekarang hanya ada satu pekerjaan di daftar tunggu, yaitu j_4 . Akibatnya, j_4 mendapatkan sumber daya dan akhirnya selesai pada waktu 20. Dalam penjadwalan ini, waktu tunggu untuk j_1 adalah 9-2, j_2 adalah (0), j_3 adalah (6-4), dan j_4 adalah (14- 5). waktu tunggu rata-rata adalah $(7+0+2+9)/4 = 4,5$. Waktu penyelesaian untuk j_1 adalah 14, j_2 adalah (6-2), j_3 adalah (9-6), dan j_4 adalah (20-5). Waktu penyelesaian rata-rata adalah $(14+4+3+15)/4 = 9$.

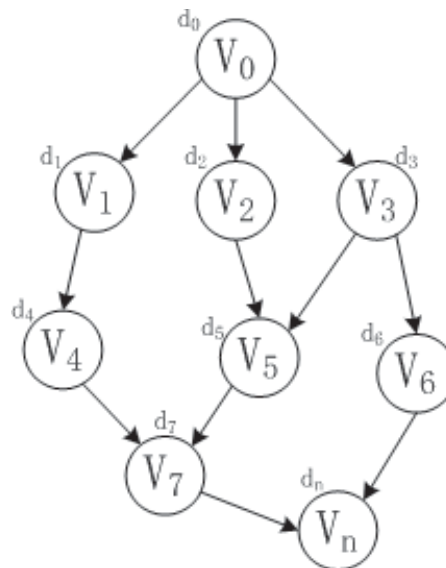
5.2.5 Algoritma Penjadwalan ASAP dan ALAP

Pertama, kami akan memperkenalkan Directed Acyclic Graphs (DAG) untuk memodelkan masalah penjadwalan tentang penundaan pada prosesor. DAG adalah graf berarah tanpa siklus berarah. Ini dibentuk oleh kumpulan simpul dan tepi berarah, setiap tepi menghubungkan satu simpul ke simpul lainnya. Tidak ada cara untuk memulai di beberapa simpul dan mengikuti urutan tepi yang akhirnya loop kembali ke simpul ini. Kami membuat DAG dengan node sumber dan node sink, seperti yang ditunjukkan pada Gambar. 5.11. Node sumber adalah V_0 , dan node sink adalah V_n . Garis solid mengacu pada penundaan eksekusi antar node. Garis putus-putus berarti tidak ada penundaan eksekusi antar node. Misalnya, baik node sumber maupun node sink tidak memiliki waktu eksekusi.

Selain itu, siswa perlu memahami dua konsep sebelum memperkenalkan algoritma, termasuk Pendahulu dan Penerus. Pendahulu mengacu pada simpul yang perlu diselesaikan sebelum simpul saat ini. Misalnya, pada Gambar 5.11, v_2 dan v_3 adalah pendahulu dari v_5 .

Penerus mengacu pada simpul yang menggantikan simpul saat ini. Pada Gambar 5.11, v_4 adalah penerus v_1 . Seperti ditunjukkan pada Gambar 5.11, kita mendefinisikan $V = \{v_0, v_1, \dots, v_n\}$ di mana v_0 dan v_n adalah node semu yang

menunjukkan node sumber dan node sink, masing-masing. $D = \{d_0, d_1, \dots, d_n\}$ di mana d_i menunjukkan penundaan eksekusi v_i ;

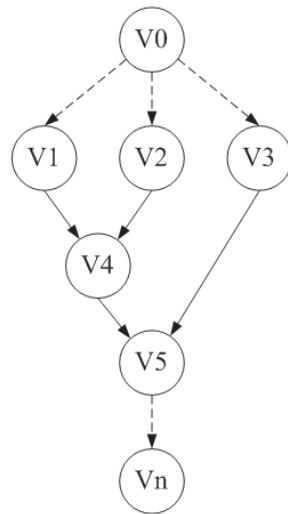


Gambar 5.11 Contoh graf asiklik berarah.

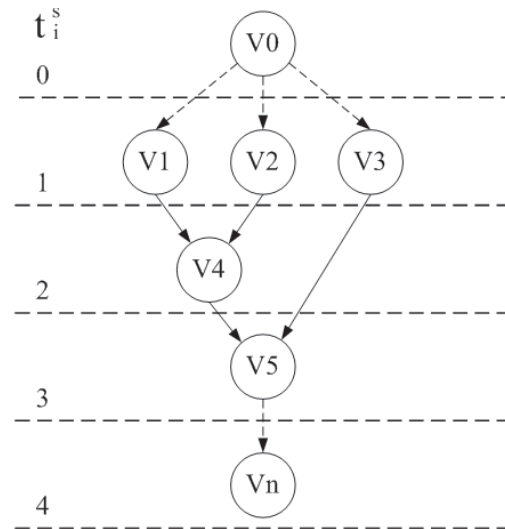
Kemudian kami menggunakan algoritma pengurutan topologi untuk menghasilkan urutan hukum, yaitu penjadwalan untuk uniprosesor. Penyortiran topologi dari graf asiklik berarah adalah pengurutan linier dari simpul-simpulnya, sehingga untuk setiap tepi berarah $\{u, v\}$ dari simpul u ke simpul v , u datang sebelum v dalam pengurutannya. Pertama, mencari daftar node yang derajat masuknya = 0, yang berarti mereka tidak memiliki tepi masuk, memasukkannya ke dalam himpunan S , dan menghapusnya dari V . Kemudian mulai loop yang terus menghapus node tanpa tepi masuk sampai V kosong. Outputnya adalah hasil sortir topologi dan penjadwalan untuk uniprosesor. Mengacu pada Gambar 5.11, kita bisa mendapatkan tiga hasil: $\{v_0, v_1, v_4, v_7, v_n\}$, $\{v_0, v_2, v_5, v_7, v_n\}$, dan $\{v_0, v_3, v_6, v_n\}$.

Untuk menghilangkan latensi, kami menetapkan nilai d_i dan menyederhanakan masalah. Kami menetapkan d_1, d_2, d_3, d_4 , dan d_5 sebagai 1. Kami menggunakan dua algoritma penjadwalan, yaitu Algoritma Penjadwalan As-Soon-As-Possible (ASAP) dan As-Late-As-Possible (ALAP).

ASAP



(a) A DAG $G(V, E, D)$ with $d_1=d_2=d_3=d_4=d_5=1, d_0=d_n=0$



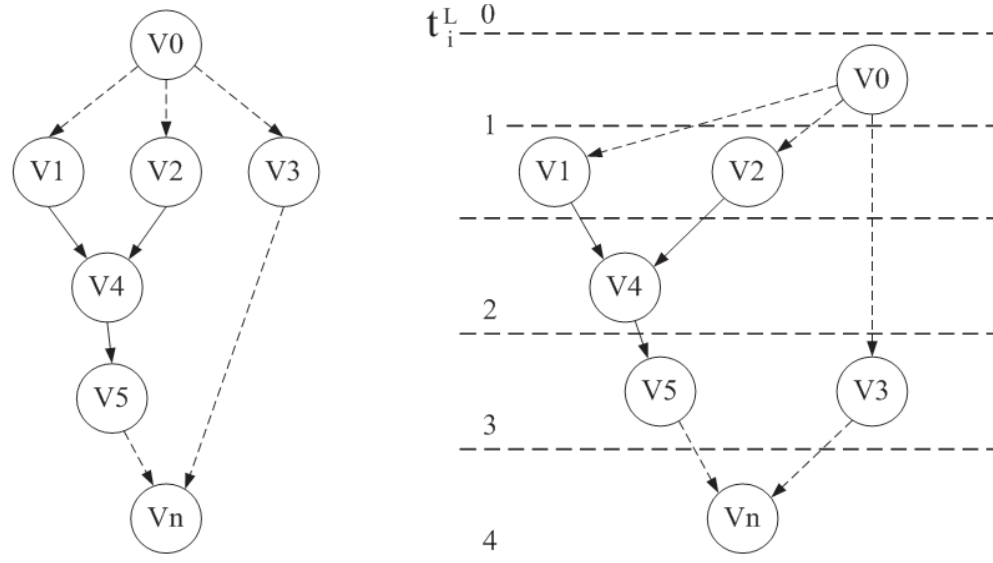
(b) The schedule generated by ASAP

Gambar 5.12 ASAP sederhana untuk penjadwalan latensi minimum.

Seperti ditunjukkan pada Gambar 5.12, pertama, himpunan $t_s = 1$, dan v_0 tidak memiliki pendahulu, dan d_0 adalah 0. Dengan demikian, v_0 memiliki latensi yang sama dengan penerusnya, v_1 , v_2 , dan v_3 . Pada langkah ini, v_0 dijadwalkan. Kemudian karena v_1 pendahulunya v_0 dijadwalkan, dapat dipilih pada waktu latensi 1. Operasi yang sama dapat diimplementasikan dengan v_2 dan v_3 pada unit waktu latensi pertama. Pada langkah ini, v_1 , v_2 , dan v_3 dijadwalkan. Kemudian v_4 dapat dipilih pada latensi 2, karena pendahulunya, v_1 dan v_2 , dijadwalkan. Namun, v_5 tidak dapat dipilih pada latensi 2, karena salah satu pendahulunya, v_4 , tidak dijadwalkan sebelum latensi 2. Kemudian setelah v_4 dijadwalkan, v_5 dapat dipilih pada latensi 3, karena pendahulunya, v_3 dan v_4 , dijadwalkan. Akhirnya, v_n dipilih pada 4 latensi, karena pendahulunya, v_5 dijadwalkan. Dalam ASAP untuk algoritma penjadwalan latensi minimum:

1. Langkah 1: jadwalkan v_0 dengan mengatur $t_0^s = 1$. Langkah ini untuk meluncurkan perhitungan algoritma.
2. Langkah 2: pilih node v_i yang pendahulunya semuanya dijadwalkan. Proses ini akan diulang sampai node sink V_n dipilih.
3. Langkah 3: jadwalkan v_i dengan mengatur $t_i^s = \max_{j:v_j \rightarrow v_i \in E} t_j^s + d_j$. Persamaan mewakili status node saat ini pada unit waktu yang tepat. Ini mewakili waktu latensi pada simpul saat ini yang menjumlahkan waktu latensi maksimum dari simpul pendahulunya.
4. Langkah 4: ulangi Langkah 2 hingga v_n dijadwalkan.

ALAP



(a) A DAG $G(V, E, D)$ with $d_1=d_2=d_3=d_4=d_5=1, d_0=d_n=0$

(b) The schedule generated by ALAP when latency-constraint 3

Gambar 5.13 Penjadwalan ALAP untuk penjadwalan latency-constraint.

Seperti ditunjukkan pada Gambar. 5.13, pertama, jadwalkan node tombol v_n pada latensi waktu $3+1$, dan atur $t_n^L = 4$. Pada langkah ini, v_n dijadwalkan. Kemudian v_3 dan v_5 dapat dipilih pada latensi 3 waktu, karena penggantinya, v_n , dijadwalkan. Pada langkah ini, v_3 dan v_5 dijadwalkan. Maka v_4 bisa menjadi dipilih pada latensi 2 waktu, karena penggantinya, v_5 , dijadwalkan. Pada langkah ini, v_4 dijadwalkan. Dalam latensi waktu ini, meskipun v_0 adalah pendahulu dari v_3 , ia tidak dapat dipilih pada latensi waktu ke-2, karena penerus v_0 lainnya, v_1 dan v_2 , tidak dijadwalkan. Kemudian v_1 dan v_2 dapat dipilih pada latensi 1 waktu, karena penerusnya, v_4 , dijadwalkan. Pada langkah ini, v_1 dan v_2 dijadwalkan. Akhirnya, v_0 dapat dipilih pada 1 waktu latency, karena penerusnya, v_1 , v_2 , dan v_3 , dijadwalkan, dan d_0 adalah 0.

Dalam ALAP untuk algoritma penjadwalan latency-constraint (λ):

- Langkah 1: jadwalkan v_n dengan mengatur $t_n^L = \lambda + 1$. Langkah ini berarti node terjadwal pertama adalah v_n .
- Langkah 2: pilih node v_i yang semua penerusnya dijadwalkan. Artinya node yang dipilih harus node yang penerusnya harus dijadwalkan. Proses ini akan diulang sampai node sumber v_0 dipilih.
- Langkah 3: jadwalkan v_i dengan mengatur $t_i^L = \min_{j:v_j \rightarrow v_i \in E} t_j^L + d_j$. persamaan mewakili status node saat ini pada unit waktu yang tepat. Dia menyatakan bahwa waktu latensi pada simpul saat ini mengurangi jumlah waktu latensi minimum dari kendala latensi simpul tenggelam.
- Langkah 4: ulangi Langkah 2 hingga v_0 dijadwalkan. Gambar 5.13 menunjukkan penjadwalan ALAP untuk penjadwalan latency-constraint.

Membandingkan penjadwalan ASAP dan ALAP seperti yang ditunjukkan pada Gambar. 5.12 dan Gambar. 5.13, kita dapat menemukan bahwa v_3 dapat diselesaikan

pada beberapa latensi waktu. Itu dapat diselesaikan pada 1 waktu latency sesegera mungkin, dan 3 waktu latency selambat mungkin. Pada bagian ini, kami memperkenalkan beberapa konsep dasar, seperti penggunaan CPU, waktu tunggu, waktu respons, dan waktu penyelesaian. Kemudian kami memperkenalkan beberapa algoritma penjadwalan, termasuk *First-Come, First Server*, *Shortest-Job-First*, penjadwalan prioritas, *Round Robin*, *As-Soon-As-Possible*, dan *As-Late-As-Possible*. Pada bagian selanjutnya, kami memperkenalkan teknologi prosesor tentang algoritma penjadwalan dalam prosesor tunggal dan multi-prosesor.

5.3 TEKNOLOGI MEMORI

Memori adalah salah satu teknologi yang paling cepat berkembang dalam sistem tertanam selama dekade terakhir. Tidak peduli seberapa cepat prosesor dapat berjalan, ada satu fakta yang tidak berubah sehingga setiap sistem tertanam membutuhkan memori untuk menyimpan data. Selanjutnya, dengan perkembangan prosesor yang pesat, semakin banyak data yang bolak-balik antara prosesor dan memori. Bandwidth memori, yang merupakan kecepatan memori, menjadi kendala utama yang mempengaruhi kinerja sistem.

Saat membangun sistem tertanam, perancang harus mempertimbangkan kinerja keseluruhan memori dalam sistem. Ada dua metrik utama untuk kinerja memori: kemampuan menulis dan permanensi penyimpanan. Menulis dalam memori bisa bermacam-macam dalam teknologi memori yang berbeda. Beberapa jenis memori, seperti Random-Access Memory (RAM), memerlukan perangkat atau teknik khusus untuk menulis. Perangkat RAM memungkinkan item data untuk dibaca dan ditulis dalam jumlah waktu yang kira-kira sama terlepas dari urutan item data yang diakses. Dua bentuk utama dari RAM modern adalah Static RAM (SRAM) dan Dynamic RAM (DRAM). Dalam SRAM, sedikit data disimpan menggunakan keadaan enam sel memori transistor. Bentuk RAM ini lebih mahal untuk diproduksi, tetapi umumnya lebih cepat dan membutuhkan daya yang lebih sedikit daripada DRAM dan, di komputer modern, sering digunakan sebagai memori cache untuk CPU. DRAM menyimpan sedikit data menggunakan pasangan transistor dan kapasitor, yang bersama-sama membentuk sel memori DRAM. Kapasitor menahan muatan tinggi atau rendah (masing-masing 1 atau 0), dan transistor bertindak sebagai sakelar yang memungkinkan sirkuit kontrol pada chip membaca status muatan kapasitor atau mengubahnya. Karena bentuk memori ini lebih murah untuk diproduksi daripada RAM statis, ini adalah bentuk utama dari memori komputer yang digunakan di komputer modern.

Pada teknologi memori kelas atas, kita dapat memilih memori yang dapat digunakan prosesor untuk menulis dalam waktu singkat. Ada beberapa jenis memori yang dapat diakses dengan mengatur jalur alamat, atau bit data, atau jalur kontrol secara tepat. Di tengah kisaran teknologi memori, beberapa memori tertulis yang lambat dapat dipilih. Di bagian bawah adalah jenis memori yang membutuhkan peralatan khusus untuk menulis. Selain kemampuan menulis, kita juga perlu mempertimbangkan permanensi penyimpanan. Berapa lama memori dapat menampung bit-bit tertulis itu sendiri dapat memiliki dampak utama pada keandalan sistem.

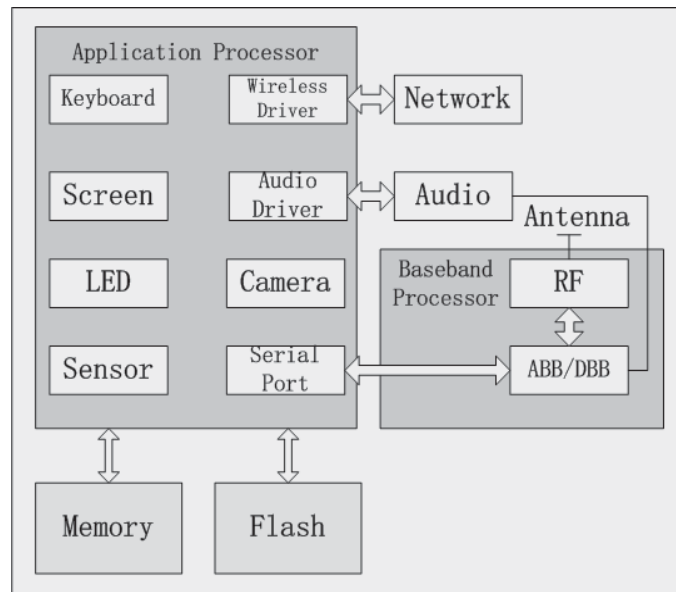
Dalam aspek keabadian penyimpanan, ada dua jenis teknologi memori: nonvolatile dan volatile. Perbedaan utama adalah bahwa memori non-volatil dapat menyimpan bit-bit tertulis setelah daya tidak lagi disuplai, tetapi volatil tidak bisa. Memori nonvolatil biasanya digunakan untuk tugas penyimpanan sekunder, atau penyimpanan persisten jangka panjang. Sementara itu, bentuk penyimpanan utama yang paling banyak digunakan saat ini adalah memori volatil. Saat komputer dimatikan, apa pun yang terkandung dalam memori yang mudah menguap akan hilang. Teknologi memori tingkat lanjut perlu dipasang ke sistem operasi. Pemrograman dinamis adalah opsi untuk optimasi memori heterogen, yang akan dibahas di Bab 7.

5.4 SISTEM TERMASUK SELULER

5.4.1 Sistem Tertanam di Perangkat Seluler

Perangkat seluler adalah sistem tertanam yang khas, yang dibentuk oleh sekelompok komponen elektronik, seperti prosesor seluler, penyimpanan, memori, grafik, sensor, kamera, baterai, dan chip lainnya. Mengintegrasikan bagian-bagian elektronik ini untuk mencapai berbagai fungsi yang diinginkan untuk tujuan yang berbeda. Di bagian ini, kita akan menggunakan smartphone untuk mewakili contoh sistem tertanam seluler. Ponsel pintar adalah salah satu perangkat seluler yang paling banyak diadopsi dalam kehidupan masyarakat kontemporer. Saat ini, struktur perangkat keras sebagian besar ponsel cerdas adalah kerangka kerja dua prosesor. Kedua prosesor tersebut adalah prosesor aplikasi dan prosesor baseband, yang ditunjukkan pada Gambar 5.14. Prosesor Aplikasi bertugas menjalankan sistem operasi seluler dan berbagai jenis aplikasi seluler. Ini adalah salah satu yang mengontrol seluruh sistem. Sebagian besar fungsi yang disediakan oleh chip, seperti keyboard, layar, kamera, dan sensor, dikendalikan oleh prosesor aplikasi.

Sementara itu, Prosesor Baseband bertanggung jawab atas komunikasi nirkabel. Komunikasi nirkabel ini bukanlah jaringan seluler atau Wi-Fi, melainkan jaringan telepon dengan Radio Frequency (RF). Frekuensi radio adalah tingkat osilasi, yang sesuai dengan frekuensi gelombang radio, dan arus bolak-balik yang membawa sinyal radio. Modul frekuensi radio digunakan untuk mengirim sinyal ke jaringan telepon. Ada dua modul dasar lain pada prosesor pita dasar, yaitu Digital Baseband (DBB) dan Analog Baseband (ABB). Mereka memodulasi dan mendemodulasi sinyal suara dan sinyal digital, mengkodekan dan mendekode saluran komunikasi, dan mengontrol modem nirkabel (modulator-demodulator). Prosesor aplikasi berkomunikasi dengan prosesor pita dasar melalui port serial, USB, dan lain-lain.



Gambar 5.14 Struktur perangkat keras smartphone.

5.4.2 Sistem Tertanam di Android

Setelah memperkenalkan struktur perangkat keras smartphone, kami akan mengambil Android sebagai contoh untuk menjelaskan Kernel di dalam Android dan menunjukkan cara kerja Kernel. Seperti yang dibahas dalam Bab 1, Android didasarkan pada Kernel Linux, dan Kernel Linux adalah lapisan abstrak antara perangkat keras dan perangkat lunak. Fungsi dasar Android disediakan oleh layanan sistem inti Kernel Linux, seperti manajemen file, manajemen memori, manajemen proses, tumpukan jaringan, dan driver. Kernel Linux juga menyediakan driver untuk mendukung semua perangkat keras yang terkait dengan sistem tertanam seluler. Seperti terlihat pada Gambar 5.15, terdapat driver tampilan, driver keyboard, driver audio, manajemen daya, driver Wi-Fi, driver kamera, dan driver sensor lainnya. Kami akan membuat daftar beberapa dari mereka dan menjelaskan apa yang mereka lakukan.

Pengemudi Tampilan. Ini didasarkan pada driver framebuffer di Linux. Framebuffer menawarkan mekanisme yang memungkinkan aplikasi untuk secara langsung mengontrol perubahan layar.

Pengemudi Papan Ketik. Ini adalah driver untuk tombol pada perangkat seluler, seperti tombol Home, tombol Menu, tombol Return, dan tombol Power.

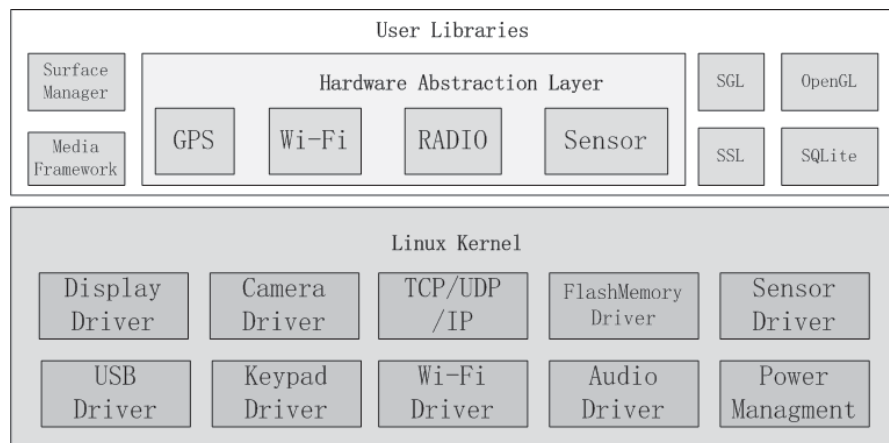
Pengemudi Wi-Fi. Ini adalah driver untuk koneksi Wi-Fi berdasarkan IEEE 802.11.

Pengemudi Sensor. Sebagian besar perangkat Android memiliki sensor bawaan yang mengukur gerakan, orientasi, dan berbagai kondisi lingkungan. Sensor ini mampu menyediakan data mentah dengan presisi dan akurasi tinggi, dan berguna jika Anda ingin memantau pergerakan atau pemosisian perangkat tiga dimensi, atau Anda ingin memantau perubahan di lingkungan sekitar di dekat perangkat.

Misalnya, sebuah game mungkin melacak pembacaan dari sensor gravitasi perangkat untuk menyimpulkan gerakan dan gerakan pengguna yang kompleks, seperti kemiringan, goyang, rotasi, atau ayunan. Demikian juga, aplikasi cuaca mungkin menggunakan sensor suhu dan sensor kelembaban perangkat untuk menghitung dan

melaporkan titik embun, atau aplikasi perjalanan mungkin menggunakan sensor medan geomagnetik dan akselerometer untuk melaporkan bantalan kompas.

Di atas Kernel Linux terdapat lapisan abstraksi perangkat keras, yang menyediakan cara mudah bagi aplikasi untuk menemukan perangkat keras pada sistem. "Abstrak" dari lapisan abstraksi perangkat keras tidak berarti operasi nyata dari perangkat keras, dan operasi perangkat keras masih dicapai oleh driver. Namun, antarmuka yang ditawarkan oleh lapisan abstraksi perangkat keras memudahkan pengembang untuk "menggunakan" perangkat keras. Lapisan Abstraksi Perangkat Keras Lapisan abstraksi perangkat keras adalah subsistem perangkat lunak untuk sistem operasi berbasis UNIX yang menyediakan abstraksi perangkat keras. Tujuan dari lapisan abstraksi perangkat keras adalah untuk memungkinkan aplikasi menemukan dan menggunakan perangkat keras dari sistem host melalui Antarmuka Pemrograman Aplikasi (API) yang sederhana, portabel, dan abstrak, terlepas dari jenis perangkat keras yang mendasarinya.

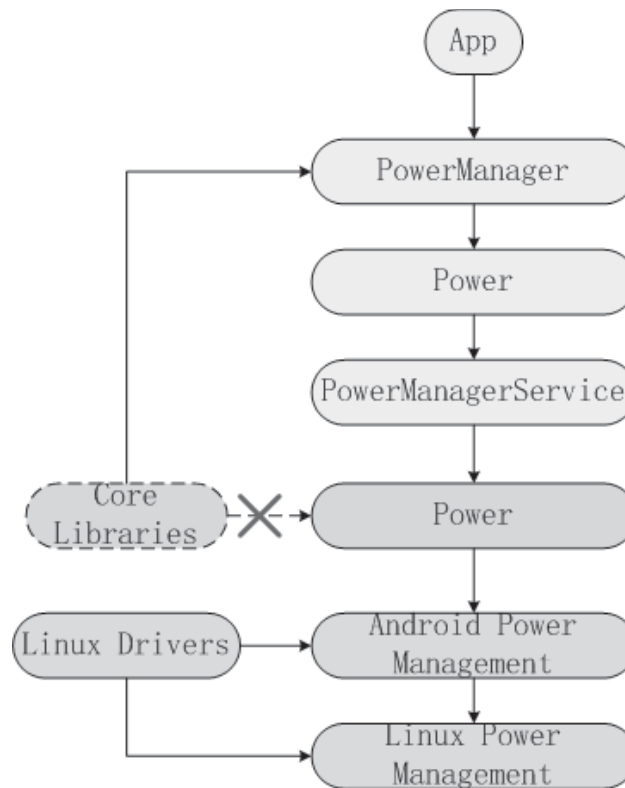


Gambar 5.15 Kernel Linux Android.

5.4.3 Manajemen Daya Android

Android mendukung manajemen dayanya sendiri (di atas manajemen daya Linux standar) yang dirancang dengan premis bahwa CPU tidak boleh mengonsumsi daya jika tidak ada aplikasi atau layanan yang memerlukan daya. Seperti yang ditunjukkan pada Gambar 5.16, Android mengharuskan aplikasi dan layanan meminta sumber daya CPU dengan penguncian saat aktif melalui kerangka aplikasi Android dan pustaka Linux asli. Jika tidak ada penguncian layar aktif, Android akan mematikan CPU. Kunci bangun digunakan oleh aplikasi dan layanan untuk meminta sumber daya CPU. Manajemen daya menggunakan penguncian bangun dan mekanisme waktu habis untuk mengubah status daya sistem, sehingga konsumsi daya sistem berkurang. Saat ini, Android hanya mendukung layar, keyboard, lampu latar tombol, dan kecerahan layar. Seperti yang ditunjukkan pada Gambar 5.17, ketika aplikasi pengguna memperoleh penguncian layar saat aktif penuh atau peristiwa aktivitas sentuh layar/keyboard terjadi, mesin akan memasuki status "bangun". Jika waktu habis atau tombol daya ditekan, mesin akan memasuki status "pemberitahuan". Jika kunci

bangun sebagian diperoleh, itu akan tetap dalam "pemberitahuan". Jika semua kunci sebagian dilepaskan, mesin akan masuk ke mode "tidur".



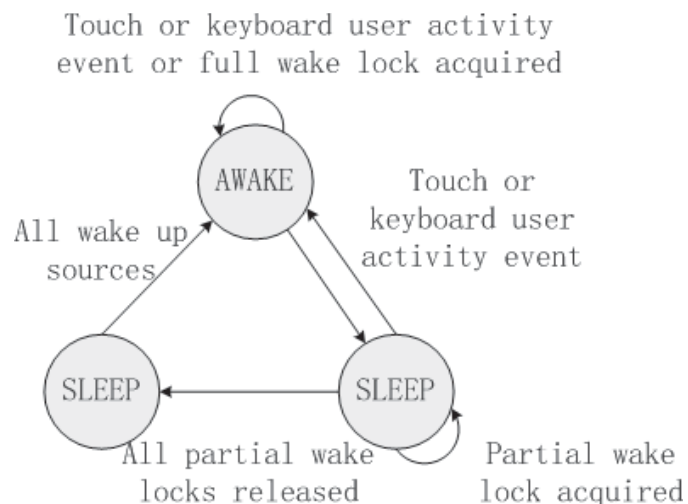
Gambar 5.16 Manajemen daya Android.

5.4.4 Sistem Tertanam di Aplikasi Seluler

Sistem tertanam seluler berada di bawah lapisan sistem operasi seluler. Sistem tertanam seluler tidak dapat langsung digunakan oleh aplikasi seluler, dan hanya dapat digunakan melalui sistem operasi seluler, seperti iOS dan Android. Kami akan mengambil Android sebagai contoh. Android sudah menjadi sistem operasi tertanam, dan akhirnya berasal dari Linux tertanam. Platform perangkat keras utama untuk Android adalah arsitektur Acorn RISC Machine (ARM). ARM adalah keluarga arsitektur set instruksi untuk prosesor komputer berdasarkan arsitektur komputasi set instruksi yang dikurangi. Pendekatan yang didasarkan pada pengurangan set instruksi mengurangi biaya, panas, dan konsumsi daya. Pengurangan tersebut adalah sifat yang diinginkan untuk perangkat yang ringan, portabel, bertenaga baterai, dan sistem tertanam lainnya. Perangkat Android menggabungkan banyak komponen perangkat keras opsional, termasuk kamera, GPS, sensor orientasi, kontrol permainan khusus, akselerometer, giroskop, barometer, magnetometer, sensor jarak, termometer, dan layar sentuh.

Kita dapat menggunakan Android Software Development Kit (SDK) untuk mengembangkan aplikasi seluler kita sendiri, dan melalui metode yang telah diterapkan untuk menggunakan sistem tertanam di dalam perangkat seluler. Misalnya, pengembang hanya dapat menggunakan kamera perangkat seluler melalui metode

pemanggilan yang dienkapsulasi dalam Android SDK. Metode desain ini membuat proses pengembangan aplikasi seluler jauh lebih sederhana daripada metode lama. Pengembang tidak perlu menghabiskan waktu merancang interaksi dengan sistem tertanam di dalam perangkat Android, dan mereka hanya perlu mengetahui fungsi apa yang dapat disediakan oleh Android SDK. Kami akan memperkenalkan lebih banyak pengetahuan tentang Android SDK dan mengembangkan teknologi di bab berikutnya.



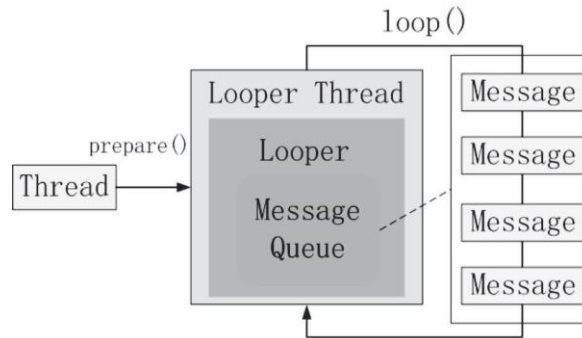
Gambar 5.17 Mesin status-terbatas dari manajemen daya Android.

5.5 MEKANISME PESAN DAN KOMUNIKASI

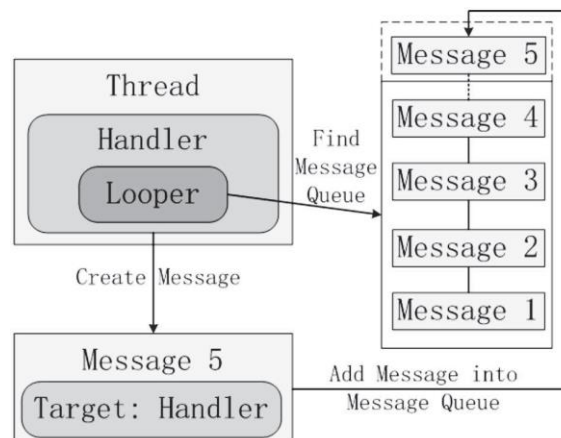
Pada bagian ini, kami akan memperkenalkan dua mekanisme yang digunakan di Android, termasuk mekanisme pesan dan komunikasi.

5.5.1 Mekanisme Pesan

Android menyediakan mekanisme pesan dalam tiga kelas inti: `Looper`, `Handler`, dan `Message`. Mirip dengan beberapa sistem operasi lain, ada Antrian Pesan di Android. Namun, Antrian Pesan ini dikemas dalam kelas `Looper`. Kelas ini terutama digunakan untuk menjalankan loop pesan untuk sebuah utas. Utas secara default tidak memiliki loop pesan yang terkait dengannya. Kita bisa memanggil metode `prepare()` di utas yang menjalankan loop, dan kemudian memanggil `loop()` untuk memproses antrian pesan. Fungsi utama dari metode `prepare()` adalah mendefinisikan objek `Looper` sebagai objek `ThreadLocal`. Setelah memanggil metode `loop()`, utas `Looper` mulai bekerja, dan terus memproses pesan pertama dalam antrian pesan. Mekanisme kerja metode `prepare()` dan `loop()`, ditunjukkan pada Gambar 5.18.



Gambar 5.18 metode prepare() dan loop().



Gambar 5.19 Gunakan handler untuk menambahkan pesan ke Antrian Pesan.

Selanjutnya, kami akan memperkenalkan cara menambahkan pesan ke dalam antrian pesan. Di Android, Handler memungkinkan kita untuk mengirim dan memproses Message dan objek Runnable yang terkait dengan Message-Queue thread. Setiap instance Handler dikaitkan dengan satu utas dan antrian pesan utas itu. Handler memiliki dua fungsi utama: (1) untuk menjadwalkan pesan dan dapat dijalankan untuk dieksekusi sebagai beberapa titik di masa depan, dan (2) untuk mengantrekan tindakan yang akan dilakukan pada utas yang berbeda dari milik kita. Gambar 5.19 menunjukkan proses penambahan pesan ke dalam antrian pesan menggunakan Handler. Pertama, Handler membuat pesan. Kemudian, temukan antrian pesan terkait berdasarkan looper. Kemudian tambahkan pesan baru ke akhir antrian pesan. Kelas pesan di Android mendefinisikan pesan yang berisi deskripsi dan objek data arbitrer yang dapat dikirim ke Handler. Meskipun konstruktor Message bersifat publik, cara terbaik untuk mendapatkan salah satunya adalah dengan memanggil metode memperoleh() atau memperolehMessage() metode Handler untuk menghemat biaya sumber daya.

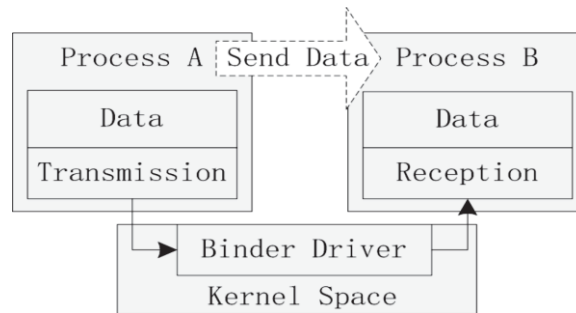
5.5.2 Mekanisme Komunikasi

Android menyediakan model komponen unit proses. Semua operasi Android pada akhirnya dinyatakan sebagai proses Linux. Android berjalan berdasarkan Kernel Linux, dan memori, proses, dan manajemen file dikendalikan oleh Kernel Linux. Layanan sistem diisolasi oleh proses Linux untuk perlindungan. Untuk mendukung

perangkat seluler, semua fungsi sistem default Android disediakan sebagai proses server. Sementara itu, fungsi-fungsi yang diwujudkan oleh aplikasi termasuk dalam proses aplikasi. Di Android, proses server dan proses aplikasi diimplementasikan oleh kelas Binder. Binder adalah bagian terpenting di Android, dan merupakan bagian inti dari mekanisme Remote Procedure Call (RPC) yang ringan. Kita dapat memperoleh langsung dari Binder untuk mengimplementasikan protokol RPC kustom kita sendiri atau sekadar membuat instance objek Binder mentah secara langsung untuk menggunakan token yang dapat dibagikan ke seluruh proses. RPC adalah bentuk Inter Process Communication (IPC), yang merupakan seperangkat teknik untuk pertukaran data di antara beberapa utas dalam satu atau lebih proses.

Android Interface Definition Language (AIDL) memungkinkan kita untuk mendefinisikan antarmuka pemrograman yang disetujui oleh klien dan layanan untuk berkomunikasi satu sama lain menggunakan IPC. Biasanya, satu proses tidak dapat mengakses memori proses lainnya. Proses perlu menguraikan objeknya menjadi primitif agar sistem operasi dapat memahami dan mengonfigurasi objek melintasi batas itu. Semua fungsi sistem Android disediakan sebagai proses server, yang membuat metode komunikasi yang dioptimalkan antar proses menjadi sangat penting. Binder mengacu pada memori Kernel yang dibagi di antara semua proses untuk meminimalkan overhead yang disebabkan oleh salinan memori. Selanjutnya, kerangka RPC yang disediakan oleh Binder ditulis dalam C++, yang lebih efisien daripada Java.

Dalam kerangka RPC, ruang kernel adalah tempat di mana semua proses dapat berbagi dan membiarkan setiap proses merujuk ke alamat memori. Di ruang Kernel, Driver Binder diimplementasikan untuk menggunakan ruang kernel untuk mengubah alamat memori yang telah dipetakan oleh setiap proses dengan alamat memori dari ruang kernel untuk referensi. Driver Binder mendukung panggilan sistem Input/Output Control (ioctl) dan operasi file, termasuk open, map, release, dan poll. Dalam ilmu komputer, ioctl adalah panggilan sistem untuk operasi input/output khusus perangkat dan operasi lain yang tidak dapat diekspresikan oleh panggilan sistem biasa. Gambar 5.20 menunjukkan contoh proses transmisi data dari proses A ke proses B. Hal pertama yang harus dilakukan proses A adalah membuka modul kernel Binder, dan modul ini menggunakan deskriptor untuk mengidentifikasi inisiator dan penerima IPC Binder. Setelah mendefinisikan transmisi (proses A) dan penerimaan (proses B) dari operasi ini, proses A mentransmisikan data ke Driver Pengikat terlebih dahulu. Kemudian, Driver Binder mengubah alamat memori data untuk memungkinkan proses B mengaksesnya.



Gambar 5.20 Mengirim data melalui Binder Driver dalam kerangka RPC.

5.6 LATIHAN

1. Apa itu sistem tertanam?
2. Bagaimana sistem tertanam digunakan dalam peralatan medis?
3. Apa utilisasi CPU?
4. Berapa waktu tunggu rata-rata dari beberapa pekerjaan prosesor?
5. Apa itu penjadwalan First-Come, First-Served?
6. Apa itu Shortest-Job-First?
7. Apa perbedaan antara skema nonpreemptive dan preemptive?
8. Apa metrik utama untuk kinerja memori?
9. Apa itu RAM?
10. Apa kendala untuk optimasi dalam sistem tertanam?
11. Mengapa perangkat seluler merupakan sistem tertanam?
12. Apa prosesor aplikasi di perangkat seluler?
13. Apa prosesor baseband di perangkat seluler?
14. Apa itu lapisan abstraksi perangkat keras di Android?
15. Apa nama CPU iPhone 6?
16. Apa nama CPU HTC One?
17. Bagaimana cara pengembang menggunakan fungsi yang disediakan oleh sistem tertanam di dalam perangkat seluler di aplikasinya sendiri?

Lanjutan

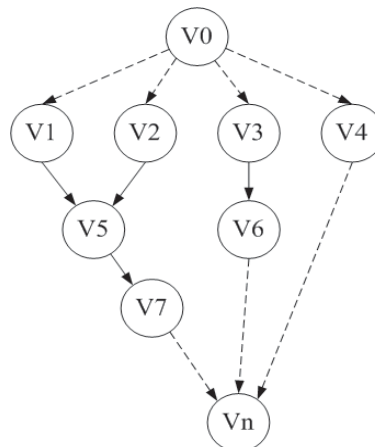
1. Bagaimana Anda membuktikan bahwa penjadwalan SJF memiliki waktu tunggu rata-rata terendah dari skema nonpreemptive dengan prosesor tunggal?
2. Apakah SJF optimal di beberapa prosesor? Mengingat waktu rata-rata dan total waktu penyelesaian.
3. Pertimbangkan bahwa ada enam pekerjaan dengan waktu penyelesaian dan waktu kedatangan yang berbeda, seperti yang ditunjukkan pada Tabel 5.1. Silakan gunakan algoritma penjadwalan FCFS untuk menjadwalkan pekerjaan ini. Anda dapat menggambar grafik untuk menjawab pertanyaan ini.
4. Mengikuti masalah di atas, silakan gunakan algoritma penjadwalan SJF untuk menjadwalkan pekerjaan ini. Hitung waktu tunggu setiap pekerjaan, waktu tunggu rata-rata, dan total waktu penyelesaian.

5. Mengikuti masalah di atas, anggap kita memiliki 3 prosesor, kemudian gunakan algoritma penjadwalan SJF untuk menjadwalkan pekerjaan ini. Hitung waktu tunggu setiap pekerjaan, waktu tunggu rata-rata, dan total waktu penyelesaian.

Tabel 5.1 Tabel penjadwalan untuk Pertanyaan Pertama

Pekerjaan	Jam kedatangan	Waktu pengerjaan
j1	0.0	7
j2	2.0	4
j3	3.0	1
j4	5.0	4
j5	6.0	5
j6	8.0	3

6. Seperti yang ditunjukkan pada Gambar 5.21, gunakan algoritma ASPA dan ALAP untuk menganalisisnya, dan gambar grafik untuk menjelaskannya.



A DAG $G=(V, E, D)$ and
 $d1=d2=d3=d4=d5=d6=d7=1$; $d0=dn=0$

Gambar 5.21 Penjadwalan DAG untuk Pertanyaan Kedua

BAB 6

PENYIMPANAN DATA DAN OPERASI SQLITE

Penyimpanan data pada perangkat seluler merupakan masalah besar selama eksekusi aplikasi seluler. Di bab terakhir, kami memperkenalkan multimedia di Android, dan menerapkan grafik 2D untuk meningkatkan mini-game Android kami. Namun, jika pengguna terganggu oleh beberapa aplikasi lain saat dia bermain game, semua data akan hilang. Kita perlu mempertimbangkan cara menyimpan data dari aplikasi di perangkat seluler untuk digunakan nanti.

Dalam bab ini, kami memperkenalkan teknik tentang menyimpan data di Android. Kita dapat menyimpan data menggunakan beberapa teknik berbeda bergantung pada ukuran, struktur, dan masa pakai data, dan apakah data tersebut akan dibagikan dengan aplikasi lain. Kami akan memperkenalkan tiga metode untuk menyimpan data lokal, termasuk preferensi Application Programming Interface (API), bundel status instance, dan file memori flash. Kemudian kami memperkenalkan SQLite di Android.

6.1 DATA LOKAL

Kami dapat menyimpan data menggunakan beberapa teknik berbeda tergantung pada ukuran, struktur, dan masa pakai data, dan apakah data tersebut akan dibagikan dengan aplikasi lain. Kami memperkenalkan tiga metode untuk menyimpan data lokal, termasuk API preferensi, bundel status instance, dan file memori flash.

6.1.1 Penyimpanan Internal dan Eksternal

Android menggunakan sistem file yang mirip dengan sistem file berbasis hard drive. Di bab sebelumnya, kami secara singkat memperkenalkan beberapa opsi untuk menyimpan data di Android. Sekarang kami memperkenalkan Penyimpanan Internal dan Penyimpanan Eksternal secara rinci. Semua perangkat Android memiliki dua area penyimpanan file: penyimpanan "internal" dan "eksternal". Pada hari-hari awal Android, sebagian besar perangkat menawarkan built-in, memori nonvolatile, yang disebut penyimpanan internal, dan media penyimpanan yang dapat dilepas, seperti kartu SD, sebagai penyimpanan eksternal. Selalu ada dua ruang penyimpanan. Tabel 6.1 mencantumkan perbedaan dalam dua jenis ruang penyimpanan ini.

Tabel 6.1 Perbedaan antara Penyimpanan Internal dan Eksternal.

Penyimpanan internal	Penyimpanan luar
Selalu tersedia	Tidak selalu tersedia
File hanya dapat diakses oleh aplikasi secara default	File dapat dibaca di luar, mereka dapat dibaca dunia

File akan dihapus saat mencopot pemasangan aplikasi	Saat mencopot pemasangan aplikasi, sistem menghapus file aplikasi dari sini hanya jika file tersebut disimpan dalam direktori dari <code>getExternalFilesDir()</code> .
---	---

Seperti yang ditunjukkan pada Tabel 6.1, ada tiga perbedaan utama antara penyimpanan internal dan penyimpanan eksternal. Pertama, penyimpanan internal adalah perangkat seluler bawaan; dengan demikian, data dan file yang disimpan di penyimpanan internal selalu tersedia. Sementara itu, penyimpanan eksternal dapat dilepas, seperti kartu SD dan penyimpanan USB. File dan data yang disimpan di penyimpanan eksternal tidak tersedia jika penyimpanan eksternal dihapus dari perangkat. Kedua, file dan data hanya dapat diakses oleh aplikasi yang menghasilkan file atau data ini sebelumnya secara default. Namun, file dan data yang disimpan di penyimpanan eksternal dapat dibaca oleh dunia, yang berarti file dan data ini dapat digunakan oleh semua aplikasi.

Ketiga, file dan data akan dihapus saat pengguna menghapus aplikasi, tetapi file dan data tidak akan dihapus kecuali jika disimpan dalam direktori dari metode `getExternalFilesDir()`. Kesimpulannya, penyimpanan internal lebih baik ketika kita ingin menyimpan file dan data agar tidak dapat diakses oleh pengguna atau aplikasi lain. Penyimpanan eksternal adalah tempat yang lebih baik untuk file dan data yang tidak memerlukan pembatasan akses dan untuk file dan data yang dimaksudkan untuk dibagikan dengan aplikasi lain.

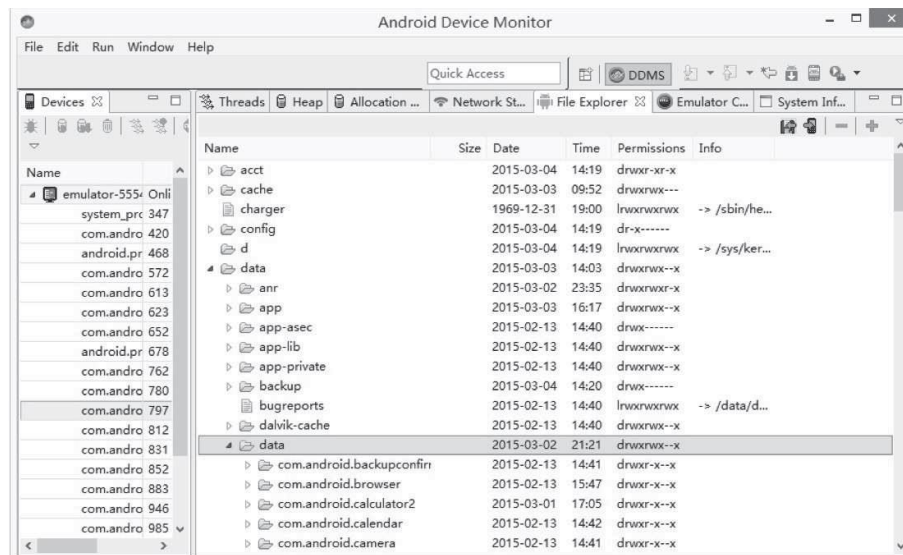
6.1.2 Menyimpan File di Penyimpanan Internal

Android menjalankan Kernel Linux di "level rendah", dan ada sistem file nyata yang terpasang di sana dengan direktori root dan semuanya. Sebenarnya, hal utama yang dapat diakses oleh aplikasi adalah direktori private paket yang dibuat pada waktu penginstalan, (`/data/data/nama paket`), seperti yang ditunjukkan pada Gambar 6.1. Saat menyimpan file ke penyimpanan internal, kita bisa mendapatkan direktori sebagai File dengan memanggil metode `getFilesDir()` atau `getCacheDir()`. Metode `getFilesDir()` mengembalikan File yang mewakili direktori internal untuk aplikasi. Metode `getCacheDir()` mengembalikan File yang mewakili direktori internal untuk file cache sementara aplikasi.

PETUNJUK: Ingatlah untuk menghapus file cache setelah tidak lagi diperlukan dan terapkan batas ukuran yang wajar untuk jumlah memori yang digunakan setiap saat. File cache akan dihapus tanpa peringatan jika sistem membutuhkan lebih banyak kapasitas penyimpanan. Kita dapat menggunakan konstruktor `File()` untuk membuat file baru di salah satu direktori ini, melalui File yang disediakan oleh salah satu metode di atas. Sebagai contoh:

```
File file = new File(context.getFilesDir(), filename);
```

Kelas Konteks menawarkan beberapa metode pembantu untuk memungkinkan kita membaca dan menulis data di sana. Kami mencantumkan beberapa metode yang paling banyak digunakan: *openFileInput()*, untuk membuka file pribadi untuk dibaca. *openFileOutput()*, untuk membuka file pribadi untuk ditulis. *fileList()*, untuk mendapatkan daftar semua file di area pribadi aplikasi. *deleteFile()*, untuk menghapus file pribadi.



Gambar 6.1 Direktori `"/data/data/"` di monitor perangkat Android.

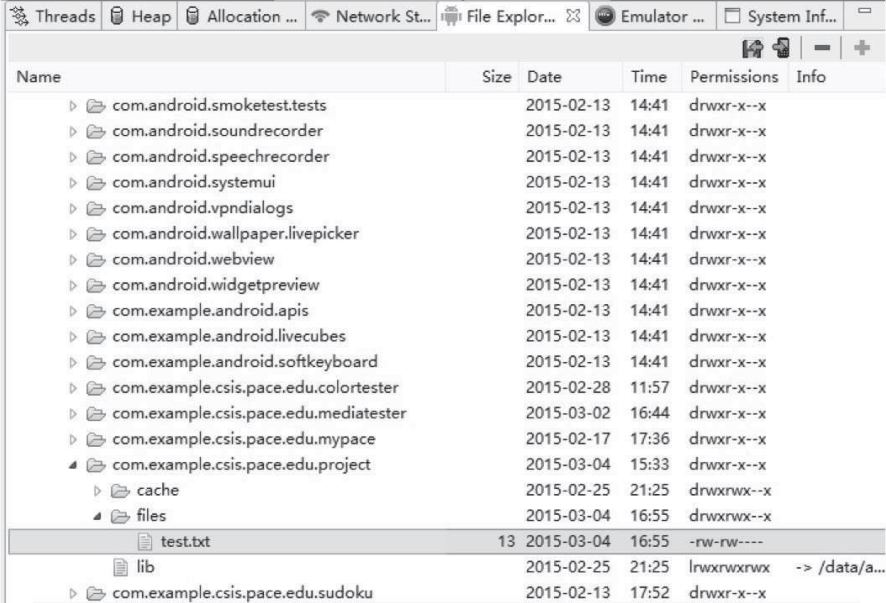
Misalnya, kita dapat memanggil *openFileOutput()* untuk mendapatkan *FileOutput-Stream* yang menulis ke dalam file di direktori internal. Metode *openFileOutput()* membuka file pribadi yang terkait dengan paket aplikasi Context ini untuk menulis, dan membuat file jika tidak ada.

```
FileOutputStream outputStream;
try {
    outputStream = openFileOutput("test.txt", Context.MODE_PRIVATE);
    outputStream.write("something new".getBytes());
    outputStream.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

Setelah menjalankan kode Android di atas, kita dapat menemukan bahwa sistem menulis "sesuatu yang baru" ke dalam "test.txt" pada direktori `"/data/data/(nama paket)/file"`, mirip dengan Gambar 6.2. Saat kita membuat proyek aplikasi Android baru dan menggunakan Android Virtual Devices (AVD) untuk menjalankannya, AVD ini secara otomatis membuat folder dengan nama yang sama dengan nama paket aplikasi di direktori `"/data/data"`.

Selanjutnya, "Context.MODE_PRIVATE" menunjukkan bahwa folder ini tidak pernah dapat diakses oleh aplikasi lain. Kita dapat menggunakan bendera lain untuk membiarkan data ini dibagikan dengan aplikasi lain. Menggunakan `MODE_WORLD_READABLE` dan `MODE_WORLD_WRITEABLE` dapat membuat data dapat dibaca dan ditulis ke aplikasi lain. Memori internal terbatas, jadi kami

menyarankan untuk menjaga ukuran data apa pun tetap rendah, dan dengan hati-hati menangani kesalahan dan pengecualian Input/Output (I/O) saat menulis jika ruang habis. Kemudian kami memperkenalkan tempat lain untuk menyimpan data.



Name	Size	Date	Time	Permissions	Info
com.android.smoketest.tests		2015-02-13	14:41	drwxr-x--x	
com.android.soundrecorder		2015-02-13	14:41	drwxr-x--x	
com.android.speechrecorder		2015-02-13	14:41	drwxr-x--x	
com.android.systemui		2015-02-13	14:41	drwxr-x--x	
com.android.vpndialogs		2015-02-13	14:41	drwxr-x--x	
com.android.wallpaper.livepicker		2015-02-13	14:41	drwxr-x--x	
com.android.webview		2015-02-13	14:41	drwxr-x--x	
com.android.widgetpreview		2015-02-13	14:41	drwxr-x--x	
com.example.android.apis		2015-02-13	14:41	drwxr-x--x	
com.example.android.livecubes		2015-02-13	14:41	drwxr-x--x	
com.example.android.softkeyboard		2015-02-13	14:41	drwxr-x--x	
com.example.csis.pace.edu.colortester		2015-02-28	11:57	drwxr-x--x	
com.example.csis.pace.edu.mediatester		2015-03-02	16:44	drwxr-x--x	
com.example.csis.pace.edu.mypace		2015-02-17	17:36	drwxr-x--x	
com.example.csis.pace.edu.project		2015-03-04	15:33	drwxr-x--x	
cache		2015-02-25	21:25	drwxrwx--x	
files		2015-03-04	16:55	drwxrwx--x	
test.txt	13	2015-03-04	16:55	-rw-rw----	
lib		2015-02-25	21:25	lrwxrwxrwx	-> /data/a...
com.example.csis.pace.edu.sudoku		2015-02-13	17:52	drwxr-x--x	

Gambar 6.2 Contoh direktori file baru.

6.1.3 Menyimpan File di Penyimpanan Eksternal

Saat ini, semua aplikasi memiliki kemampuan untuk membaca penyimpanan eksternal tanpa izin khusus, tetapi dari dokumen resmi Android, dikatakan bahwa ini akan berubah dalam rilis mendatang. Untuk memastikan bahwa aplikasi kami terus berfungsi, kami harus mendeklarasikan izin "baca" sekarang.

```
<manifest>
<user-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
</manifest>
```

Untuk menulis ke penyimpanan eksternal, kita harus mendapatkan izin terlebih dahulu.

```
<manifest>
<user-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
</manifest>
```

Berbeda dengan penyimpanan internal, penyimpanan eksternal mungkin tidak tersedia untuk beberapa waktu, seperti file dan data di penyimpanan eksternal mungkin ditransfer ke komputer, atau kartu SD dikeluarkan dari smartphone. Akibatnya, kita harus memverifikasi apakah penyimpanan eksternal tersedia sebelum mengaksesnya. Metode `getExternalStorageState()` dipanggil untuk menanyakan status penyimpanan eksternal. Jika status yang dikembalikan sama dengan `MEDIA_MOUNTED`, maka kita dapat membaca dan menulis file dan data di penyimpanan eksternal. Gambar 6.3 adalah contoh untuk memeriksa status

penyimpanan eksternal untuk pembacaan. Ada dua syarat yang harus dipenuhi, yaitu apakah penyimpanan eksternal tersedia dan dapat dibaca.

```
//Check the external storage state for read
public boolean isExternalStorageReadable(){
    String state = Environment.getExternalStorageState();
    if(Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
        return true;
    }
    return false;
}
```

Gambar 6.3 Memeriksa status penyimpanan eksternal untuk membaca.

Gambar 6.4 adalah contoh untuk memeriksa status penyimpanan eksternal untuk membaca dan menulis file di penyimpanan eksternal. Di Android, penyimpanan eksternal dapat dimodifikasi oleh pengguna dan aplikasi, dan ada dua kategori file: *File publik*. File publik harus tersedia untuk aplikasi lain dan pengguna. Saat pengguna mencopot pemasangan aplikasi, file-file ini harus tetap tersedia bagi pengguna. *File pribadi*. File pribadi yang secara sah dimiliki oleh beberapa aplikasi tertentu dan harus dihapus saat aplikasi tersebut dicopot pemasangannya. Meskipun file-file ini secara teknis dapat diakses oleh pengguna dan aplikasi lain karena berada di penyimpanan eksternal, file-file ini secara realistis tidak memberikan nilai kepada pengguna di luar aplikasi.

```
//Check external storage state for reading and writing
public boolean isExternalStorageWritable(){
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)){
        return true;
    }
    return false;
}
```

Gambar 6.4 Memeriksa status penyimpanan eksternal untuk membaca dan menulis

Kita bisa menggunakan metode `getExternalStoragePublicDirectory()` untuk menyimpan file publik di penyimpanan eksternal. Metode ini mendapatkan `File` yang mewakili direktori yang sesuai pada penyimpanan eksternal, dan dibutuhkan argumen yang menentukan jenis file yang akan disimpan. Misalnya, kami ingin mendapatkan direktori yang menyimpan gambar publik pengguna.

```
//get the directory for the public directory
public File getAlbumStorageDir(String name){
    File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), name);
    return file;
}
```

Gambar 6.5 Mendapatkan direktori file publik.











Kemudian kita dapat memanggil metode `getExternalFilesDir()` untuk menyimpan file dan data dalam folder pribadi. Serupa dengan metode

`getExternalStoragePublicDirectory()`, metode ini juga mendapatkan File yang mewakili direktori pada penyimpanan eksternal. Namun, semua direktori yang dibuat dengan cara ini ditambahkan ke direktori induk yang merangkum semua file penyimpanan eksternal aplikasi, yang akan dihapus saat aplikasi dicopot pemasangannya. Metode `getExternalFilesDir()` secara otomatis membuat direktori tertentu yang akan dihapus saat aplikasi dicopot pemasangannya. Jika file dan data tetap tersedia setelah aplikasi dihapus, kita harus menggunakan metode `getExternalStoragePublicDirectory()`.

Kedua metode ini harus menggunakan nama direktori yang disediakan oleh konstanta API, seperti `DIRECTORY_PICTURES` dalam dua contoh di atas. Nama direktori ini memastikan bahwa file dan data diperlakukan dengan benar oleh sistem Android. Selain `DIRECTORY_PICTURES`, ada beberapa konstanta API lainnya, seperti yang ditunjukkan pada Gambar 6.7.

```
//get the directory for the private directory
public File getAlbumStorageDir(Context context, String name){
    File file = new File(context.getExternalFilesDir(
        Environment.DIRECTORY_PICTURES), name);
    return file;
}
```

Gambar 6.6 Mendapatkan direktori file pribadi.

 DIRECTORY_PICTURES	String
 DIRECTORY_ALARMS	String
 DIRECTORY_DCIM	String
 DIRECTORY_DOCUMENTS	String
 DIRECTORY_DOWNLOADS	String
 DIRECTORY_MOVIES	String
 DIRECTORY_MUSIC	String
 DIRECTORY_NOTIFICATIONS	String
 DIRECTORY_PODCASTS	String
 DIRECTORY_RINGTONES	String

Gambar 6.7 Nama direktori yang disediakan oleh konstanta API.

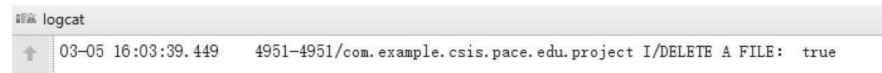
6.1.4 Menghapus File

Seperti disebutkan dalam bab pertama, perangkat seluler memiliki sumber daya yang terbatas; oleh karena itu, sangat penting untuk menghapus file yang tidak lagi diperlukan. Cara yang paling mudah dan banyak digunakan untuk menghapus file adalah dengan memiliki panggilan referensi file yang dibuka `delete()` pada dirinya sendiri. Metode `delete()` mengembalikan `true` jika file dihapus, dan mengembalikan `false` sebaliknya. Jika kita ingin menghapus sebuah file di penyimpanan internal, kita bisa mendapatkan direktori tersebut dengan menggunakan metode `getFilesDir()`, kemudian membuat File baru yang merupakan file yang akan dihapus. Gambar 6.8 menunjukkan proses penghapusan File pada penyimpanan internal dan hasil logcat yang menunjukkan bahwa file berhasil dihapus.

Jika kita ingin menghapus sebuah file pada penyimpanan eksternal, kita dapat menggunakan metode `getExternalFilesDir()` dan `getExternalStoragePublicDirectory()`

untuk mendapatkan direktori file tersebut, kemudian membuat File baru yang merupakan file yang akan dihapus. Namun, sebelum kita menghapus beberapa file, kita perlu memastikan bahwa kita memiliki izin untuk menghapusnya. Masuk ke AndroidManifest.xml dan periksa apakah kami memiliki izin WRITE_EXTERNAL_STORAGE. Jika tidak, tambahkan izin seperti yang diperkenalkan di Bagian 6.1.3

```
public void deleteTest(){
    //delete a File on internal storage via its name
    File dir = getFilesDir();
    File file = new File(dir, "test.txt");
    Boolean bool = file.delete();
    Log.i("DELETE A FILE", bool.toString());
}
```



Gambar 6.8 Menghapus File dari Penyimpanan Internal

Langkah pertama menghapus file di direktori publik dan privat mirip dengan proses mendapatkan direktori dari direktori publik dan privat. Gambar 6.9 adalah contoh penghapusan file album di direktori publik.

```
public void deleteAlbumStorage(String name){
    File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), name);
    file.delete();
}
```

Gambar 6.9 Menghapus File Album di Direktori Publik dengan Nama.

6.1.5 Query Space

Sebelum kita menyimpan beberapa file atau data ke dalam perangkat, kita dapat memeriksa apakah cukup ruang yang tersedia untuk menghindari menyebabkan IOException. Kita dapat menggunakan getFreeSpace() untuk mendapatkan jumlah ruang yang tersedia dan getTotalSpace() untuk mendapatkan jumlah total ruang dalam volume penyimpanan. Kedua metode ini mengembalikan angka dalam byte dan mengembalikan 0 jika jalurnya tidak ada. Misalnya, Gambar 6.10 menunjukkan fungsi mendapatkan ruang yang tersedia saat ini dan total ruang di penyimpanan internal. Hasilnya ditunjukkan pada Gambar 6.11.

```
public void getSpace(){
    File dir = getFilesDir();
    String free = "" + dir.getFreeSpace();
    String total = "" + dir.getTotalSpace();
    Log.i("Free Space", free);
    Log.i("Total Space", total);
}
```

Gambar 6.10 Mendapatkan ruang yang tersedia dan total saat ini di penyimpanan internal.

```

logcat
03-05 20:32:57.547 901-901/com.example.csis.pace.edu.project I/Free Space: 390037504
03-05 20:32:57.548 901-901/com.example.csis.pace.edu.project I/Total Space: 567640064

```

Gambar 6.11 Hasil `getSpace()`.

Faktanya, sistem Android tidak menjamin bahwa kita dapat menyimpan byte sebanyak jumlah yang diperoleh dengan metode `getFreeSpace()`, karena sistem membutuhkan ruang untuk menangani operasi lain. Jika jumlah yang diperoleh dengan metode `getFreeSpace()` lebih sedikit dari ruang yang kita butuhkan, atau jika ada lebih dari 10% ruang yang tersedia, maka file tersebut aman untuk disimpan. Jika tidak, proses penyimpanan mungkin tidak berhasil. Jika kita tidak tahu persis berapa banyak ruang yang kita perlukan untuk menyimpan file, kita dapat mencoba menulis file, dan kemudian menangkap `IOException` jika terjadi. Dalam banyak situasi, kita tidak dapat mengetahui ukuran file yang tepat sebelumnya, seperti mengekstrak file dari format terkompresi.

6.2 DATABASE SQLITE

SQLite adalah database SQL open-source yang menyimpan data ke file teks di perangkat. Ini mendukung semua database relasional, jadi kita tidak perlu membuat koneksi apa pun untuk menggunakannya, seperti Java Database Connectivity (JDBC) dan Open Database Connectivity (ODBC). Database SQLite tertanam di Android, dan API untuk menggunakan database di Android tersedia dalam paket `android.database.sqlite`.

Tabel 6.2 Struktur Tabel Kontak

Field	Type	Key
ID	Int	Primary
Name	Text	
Phone_number	Text	

6.2.1 Struktur Tabel

Salah satu prinsip utama database SQL adalah skema, deklarasi formal tentang bagaimana database diatur. Skema tercermin dalam pernyataan SQL yang Anda gunakan untuk membuat database Anda. Kami akan membuat tabel sederhana, yang disebut kontak, untuk mengilustrasikan teknik SQLite di Android. Struktur tabel ditunjukkan pada Tabel 6.2. ID adalah kunci utama, dan tipenya adalah int, sedangkan nama dan nomor_telepon adalah atribut biasa yang tipenya adalah Teks. Sebelum kita menggunakan teknik SQLite untuk mengimplementasikan fungsi `create`, `update`, `delete`, dan `read`, kita perlu membuat kelas individual, bernama `Contact`, dengan `getter` dan `setter`, sebagai berikut.

6.2.2 Operasi CRUD

Kemudian kita perlu membuat kelas kita sendiri untuk menangani semua operasi database CRUD (membuat, membaca, memperbarui, dan menghapus).

Mirip dengan proses kelas Kontak, kami membuat kelas baru, bernama DatabaseHandler, dan membuatnya memperluas SQLiteOpenHelper.

```
public class DatabaseHandler extends SQLiteOpenHelper
```

Setelah memperluas kelas DatabaseHandler dari SQLiteOpenHelper, kita perlu mengganti dua metode, yaitu onCreate() dan onUpgrade(). Metode onCreate() adalah tempat kita perlu menulis pernyataan tabel. Itu dipanggil ketika database dibuat. Metode onUpgrade() dipanggil saat database diupgrade, seperti memodifikasi struktur tabel, menambahkan batasan ke database, dll. Sebelum mengesampingkan kedua metode ini, kita perlu membuat dan mendefinisikan beberapa atribut pribadi, seperti yang ditunjukkan pada Gambar 6.13. Kami mendefinisikan nama database, nama tabel, dan nama kolom. DatabaseHandler(Konteks konteks) adalah konstruktornya.

Kemudian kami mengganti metode onCreate() dan onUpgrade(). Dalam metode onCreate(), kita membuat dan mendefinisikan kalimat sql untuk membuat tabel, dan menggunakan metode execSQL() untuk mengimplementasikannya. Dalam metode onUpgrade(), kami menghapus tabel Kontak dan membuatnya kembali. Kode dari kedua metode ini ditunjukkan pada Gambar 6.14. Kalimat SQL untuk membuat tabel adalah: CREATE TABLE table_name (attribute1 TYPE PRIMARY KEY, attribute2 TYPE, . . .).


```

public class Contact {
    int _id;
    String _name;
    String _phone_number;

    public Contact(){}

    public Contact(int id, String name, String _phone_number){
        this._id = id;
        this._name = name;
        this._phone_number = _phone_number;
    }

    public Contact(String name, String _phone_number){
        this._name = name;
        this._phone_number = _phone_number;
    }

    // getting ID
    public int getID(){
        return this._id;
    }

    // setting id
    public void setID(int id){
        this._id = id;
    }

    // getting name
    public String getName(){
        return this._name;
    }

    // setting name
    public void setName(String name){
        this._name = name;
    }

    // getting phone number
    public String getPhoneNumber(){
        return this._phone_number;
    }

    // setting phone number
    public void setPhoneNumber(String phone_number){
        this._phone_number = phone_number;
    }
}

```

Gambar 6.12 Contoh kontak.

```

// All Static variables
// Database Version
private static final int DATABASE_VERSION = 1;

// Database Name
private static final String DATABASE_NAME = "contactsManager";

// Contacts table name
private static final String TABLE_CONTACTS = "contacts";

// Contacts Table Columns names
private static final String KEY_ID = "id";
private static final String KEY_NAME = "name";
private static final String KEY_PH_NO = "phone_number";

public DatabaseHandler(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

```

Gambar 6.13 Beberapa atribut pribadi di Kelas DatabaseHandler.

```

@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
        + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
        + KEY_PH_NO + " TEXT" + ")";
    db.execSQL(CREATE_CONTACTS_TABLE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // Drop older table if existed
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);

    // Create tables again
    onCreate(db);
}

```

Gambar 6.14 metode onCreate() dan onUpgrade().

Sekarang kita perlu mengimplementasikan operasi CRUD database. Gambar 6.15 adalah gambaran umum dari metode yang kami gunakan. Kami menerapkan fungsi berikut: menambahkan kontak baru, mendapatkan satu kontak, mendapatkan semua kontak, memperbarui satu kontak, dan menghapus satu kontak.

```

//CRUD
//Add new contact
public void addContact(Contact contact){...}

//Get single contact
public Contact getContact(int id){...}

//Get all contacts
public List<Contact> getAllContacts(){...}

//Update single contact
public int updateContact(Contact contact){...}

//Delete single contact
public void deleteContact(Contact contact){...}

```

Gambar 6.15 Ikhtisar operasi CRUD.

Create Operation: addContact()

Parameter metode addContact() adalah objek Kontak, yang akan dibuat dan ditambahkan ke dalam database. Kita perlu membangun parameter Content- Values menggunakan objek Contact. Kelas ContentValues mirip dengan kelas Hashtable, dan bertanggung jawab untuk menyimpan pasangan nilai kunci. Implementasi rinci dari metode addContact() ditunjukkan pada Gambar 6.16. Setelah kita memasukkan data ke dalam database, kita perlu menutup koneksi database.

```

public void addContact(Contact contact){
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact.getName());
    values.put(KEY_PH_NO, contact.getPhoneNumber());

    db.insert(TABLE_CONTACTS, null, values);
    db.close();
}

```

Gambar 6.16 Metode addContact().

Read Operations: getContact() and getAllContacts()

Kami menggunakan metode getContact() untuk membaca satu kontak dan metode getAllContacts() untuk membaca semua kontak dari database. Parameter metode getContact() adalah ID, dan kita tidak memerlukan parameter apa pun. Implementasi getContact() ditunjukkan pada Gambar 6.17, dan implementasi getAllContacts() ditunjukkan pada Gambar 6.18. Dalam dua metode ini, kami menggunakan cursor untuk mengarahkan satu baris dari hasil yang diambil oleh kueri.

```

public Contact getContact(int id){
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(TABLE_CONTACTS, new String[]{KEY_ID,KEY_ID,KEY_PH_NO},
        KEY_ID + "=?", new String[] { String.valueOf(id) }, null, null, null, null);
    if(cursor != null){
        cursor.moveToFirst();
    }
    Contact contact = new Contact(Integer.parseInt(cursor.getString(0)),
        cursor.getString(1), cursor.getString(2));

    return contact;
}

```

Gambar 6.17 Metode getContact().

```

public List<Contact> getAllContacts(){
    List<Contact> contactList = new ArrayList<Contact>();

    String sql_select = "SELECT * FROM "+ TABLE_CONTACTS;

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(sql_select, null);

    if(cursor.moveToFirst()){
        while(cursor.moveToNext()){
            Contact contact = new Contact();
            contact.setID(Integer.parseInt(cursor.getString(0)));
            contact.setName(cursor.getString(1));
            contact.setPhoneNumber(cursor.getString(2));
            // Adding contact to list
            contactList.add(contact);
        }
    }
    return contactList;
}

```

Gambar 6.18 Metode getAllContacts().

Update Operation: updateContact()

Kami menggunakan updateContact() untuk memperbarui kontak tunggal dalam database. Parameter dari metode ini adalah objek Kontak. Mirip dengan addContact(), kami menggunakan ContentValues untuk merangkum sebagai objek kontak, dan kemudian mengganti kontak lama dengan yang baru dienkapsulasi melalui ID. Implementasi metode updateContact() ditunjukkan pada Gambar 6.19.

```

public int updateContact(Contact contact){
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact.getName());
    values.put(KEY_PH_NO, contact.getPhoneNumber());

    return db.update(TABLE_CONTACTS, values, KEY_ID + "=?",
        new String[] {String.valueOf(contact.getID())});
}

```

Gambar 6.19 Metode updateContact().

Delete Operation: deleteContact()

Kami menggunakan metode deleteContact() untuk menghapus satu kontak dari database menggunakan parameter objek Kontak. Implementasi metode deleteContact() ditunjukkan pada Gambar 6.20.

```
public void deleteContact(Contact contact){
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(TABLE_CONTACTS, KEY_ID + "=?",
        new String[]{String.valueOf(contact.getID())});
    db.close();
}
```

Gambar 6.20 Metode deleteContact().

6.2.3 Penggunaan Teknik SQLite

Setelah kita membuat kelas Kontak dan kelas DatabaseHandler, kita akan mengenalkan cara menggunakannya di aplikasi Android. Dalam metode onCreate() suatu aktivitas, kita membuat objek DatabaseHandler sebagai berikut: DatabaseHandler db = new DatabaseHandler(this).

```
Log.d("Insert", "Inserting ...");
db.addContact(new Contact("AAA", "1233211232"));
Log.d("Insert AAA", "Success!");
db.addContact(new Contact("BBB", "9143493981"));
Log.d("Insert BBB", "Success!");
```

```
10-27 17:23:32.200    2287-2287/? D/Insert: Inserting ...
10-27 17:23:32.230    2287-2287/? D/Insert AAA: Success!
10-27 17:23:32.240    2287-2287/? D/Insert BBB: Success!
```

Gambar 6.21 Hasil running tentang addContact() dan getAllContacts().

```
Log.d("Reading: ", "Reading all contacts..");
contacts = db.getAllContacts();
for (Contact cn : contacts) {
    String log = "Id: " + cn.getID() + " ,Name: " + cn.getName() + " ,Phone: " +
cn.getPhoneNumber();
    // Writing Contacts to log
    Log.d("Name: ", log);
}
```

```
Reading:      Reading all contacts..
Name:        Id: 2 ,Name: AAA ,Phone: 1233211232
Name:        Id: 3 ,Name: BBB ,Phone: 9143493981
```

Gambar 6.22 Hasil running tentang getAllContacts().

Kemudian kita uji penggunaan metode addContact() sebagai berikut. Untuk mempermudah melacak hasilnya, kami menggunakan Log.d() untuk mencetak beberapa informasi catatan. Hasil running ditunjukkan pada Gambar 6.21. Kemudian kita uji penggunaan getAllContacts() sebagai berikut. Kami menggunakan loop foreach

dalam implementasi kami dan mencetak setiap catatan kontak yang diambil dari database. Hasil running ditunjukkan pada Gambar 6.22.

Untuk menghapus satu kontak dari database, implementasinya mirip dengan yang sebelumnya. Kami menggunakan loop untuk menghapus semua kontak dalam database dan mencetaknya sebelum dihapus. Selanjutnya, kita menggunakan `getAllContacts()` untuk memeriksa ulang fungsi metode `deleteContact()`, dan hasilnya ditunjukkan pada Gambar 6.23. Setelah “Membaca semua kontak..”, tidak ada catatan yang dicetak, yang menunjukkan bahwa kami telah menghapus semua kontak di database. Terakhir, kami menguji metode `updateContact()`. Dari log yang dicetak sebelumnya, kita tahu bahwa ID kontak AAA adalah 2. Kami berencana untuk memperbarui kontak ini dari (“AAA”, “1233211232”) menjadi (“CCC”, “3213213121”). Sebelum memperbarui kontak ini, kami mencetak beberapa informasi untuk membantu kami menemukan informasi yang berguna di logcat. Hasil running ditunjukkan pada Gambar 6.24.

```
Log.d("deleting:", "Deleting all contacts..");
for (Contact c : contacts){
    String log = "Id: " + c.getID() + " ,Name: " + c.getName() + " ,Phone: " +
c.getPhoneNumber();
    Log.d("Delete:", log);
    db.deleteContact(c);
}
deleting:      Deleting all contacts..
Delete:       Id: 2 ,Name: AAA ,Phone: 1233211232
Delete:       Id: 3 ,Name: BBB ,Phone: 9143493981
Reading:      Reading all contacts..
HostConnection::get() New Host Connection established 0
0, tid 3546
```

Gambar 6.23 Hasil Menjalankan `deleteContact()`

```
Log.d("Update", "AAA->CCC");
Contact contact1 = db.getContact(2);
Log.d("Old Contact", contact1.getName());
contact1.setName("CCC");
contact1.setPhoneNumber("3213213121");
Log.d("New Contact", contact1.getName());
db.updateContact(contact1);
Update       AAA->CCC
Old Contact  AAA
New Contact  CCC
Reading:     Reading all contacts..
Name:       Id: 2 ,Name: CCC ,Phone: 3213213121
Name:       Id: 3 ,Name: BBB ,Phone: 9143493981
```

Gambar 6.24 Hasil running dari `updateContact()`.

6.3 PENYEDIA KONTEN

Jika kita perlu berbagi data di antara beberapa aplikasi, kita dapat menggunakan penyedia konten untuk mencapai tujuan ini. Penyedia konten adalah salah satu blok

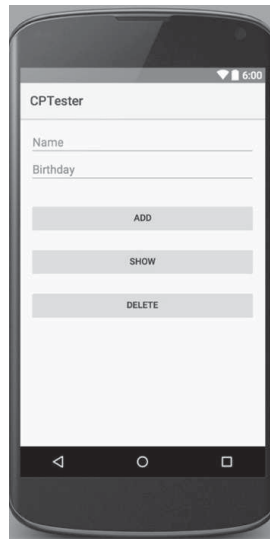
pembangun utama aplikasi Android, yang menyediakan konten untuk aplikasi. Mereka merangkum data dan menyediakannya ke aplikasi melalui antarmuka `ContentResolver` tunggal. Saat permintaan dibuat melalui `ContentResolver`, sistem memeriksa otoritas Uniform Resource Identifier (URI) yang diberikan dan meneruskan permintaan ke penyedia konten yang terdaftar pada otoritas. Penyedia konten dapat menginterpretasikan sisa URI sesuai keinginannya. Kelas `UriMatcher` berguna untuk menguraikan URI.

Metode utama yang perlu diterapkan adalah:

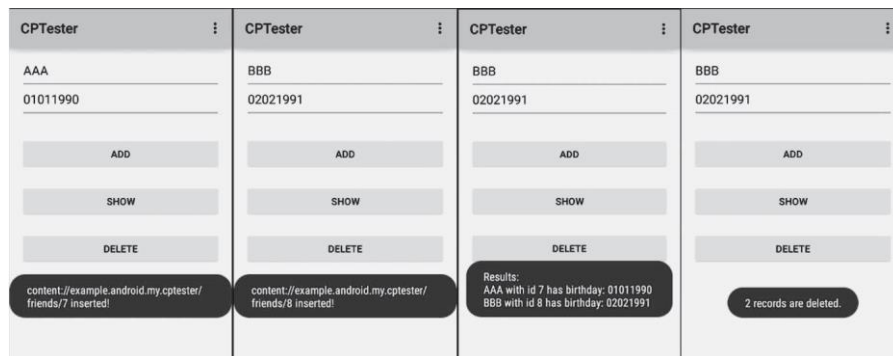
1. `onCreate()`, yang dipanggil untuk menginisialisasi penyedia
2. `query(Uri, String[], String, String[], String)`, yang mengembalikan data ke pemanggil
3. `insert(Uri, ContentValues)`, yang menyisipkan data baru ke dalam penyedia konten
4. `update(Uri, ContentValues, String, String[])`, yang mengupdate data yang ada di content provider
5. `delete(Uri, String, String[])`, yang menghapus data dari penyedia konten
6. `getType(Uri)`, yang mengembalikan tipe data MIME di penyedia konten

Kami akan menggunakan contoh sederhana untuk menunjukkan cara menggunakan penyedia konten. Setelah membuat aplikasi Android kosong biasa, buat kelas Java normal, bernama `BirthProvider`, memperluas `ContentProvider`. Hal pertama yang perlu kita lakukan adalah mengganti metode `onCreate()`, `query()`, `insert()`, `update()`, `delete()`, dan `getType()`. Semua penyedia konten harus mengimplementasikan antarmuka yang sama untuk penggunaan lebih lanjut. Kemudian kita akan menggunakan kelas `UriMatcher` untuk memetakan URI konten dengan pola tertentu, yang akan membantu kita memilih tindakan yang diinginkan untuk URI konten yang masuk. Data di penyedia konten mirip dengan satu database. Setiap kolom memiliki nilai tipe numerik yang unik, bernama `_ID`, yang digunakan untuk mencari record tertentu. Hasil kembali dari kueri penyedia konten adalah `Cursor`. Setiap penyedia konten harus memiliki URI publik yang unik, yang digunakan untuk menentukan kumpulan data yang mengarah ke sana. Satu penyedia konten dapat memiliki beberapa kumpulan data (seperti beberapa tabel), dan dalam situasi seperti ini, beberapa URI diperlukan agar sesuai dengan setiap kumpulan data. URI ini harus dimulai dengan "konten: //". Kode `BirthProvider.java` disediakan di Lampiran.

Dari awal hingga di sini, kami menyelesaikan bagian implementasi dari aplikasi Android penyedia konten kami. Hal berikutnya yang perlu kita lakukan adalah antarmuka pengguna. Kami menambahkan dua `EditTexts` ke dalam tata letak untuk memungkinkan pengguna mengisi nama dan tanggal lahir, dan tiga tombol untuk menambah, menampilkan, dan menghapus catatan yang sesuai dengan tiga metode dalam aktivitas masing-masing. Gambar 6.25 menunjukkan antarmuka pengguna. Hasil running ditunjukkan pada Gambar 6.26.



Gambar 6.25 Antarmuka pengguna CPTester.



Gambar 6.26 Hasil running CPTester.

6.4 LATIHAN

6.4.1 Latihan Dasar

1. Aspek data apa yang dipertimbangkan untuk disimpan?
2. Apa yang dilakukan PreferenceActivity?
3. Apa yang dapat dilakukan API preferensi untuk lebih dari sekadar opsi?
4. Metode mana yang akan ditimpa saat memulihkan dari game yang disimpan?
5. Apa yang dilakukan tampilan Android normal dengan membidik orientasi layar?
6. Apakah status instance permanen?
7. Di mana status instance disimpan?
8. Untuk apa keadaan instans digunakan?
9. Apakah ada sistem file nyata di Android?
10. Di mana metode rutin I/O file Java yang biasa di Android?
11. Apa metode umum untuk membaca dan menulis data yang disediakan oleh kelas Konteks?
12. Apa yang dilakukan metode deleteFile()?
13. Apa yang dilakukan metode fileList()?
14. Apa yang dilakukan openFileInput()?
15. Apa yang dilakukan operFileOutput()?
16. Apa itu kartu SD? Bisakah kartu SD digunakan untuk kode?

17. Bagaimana Anda mendapatkan direktori pada sistem file eksternal?

6.4.2 Latihan Tingkat Lanjut

1. Ada berapa macam teknik yang dapat digunakan untuk menyimpan data di Android? Apakah mereka?
2. Apa perbedaan antara menyimpan data di perangkat dan di kartu SD?

BAB 7

PENGOPTIMALAN SELULER DENGAN PEMROGRAMAN DINAMIS

Optimasi dengan menggunakan heterogeneous computings adalah salah satu metode penting untuk meningkatkan kinerja. Sistem tertanam memiliki banyak kendala, seperti waktu, keandalan, dan konsumsi energi. Menyeimbangkan kendala ini merupakan mekanisme yang efisien untuk meningkatkan kinerja sistem tertanam. Oleh karena itu, memanfaatkan komputasi heterogen dalam sistem tertanam seluler merupakan masalah optimasi. Pemrograman dinamis merupakan pendekatan penting untuk mengoptimalkan sistem tertanam, dan ini telah digunakan secara luas di berbagai industri dan domain seluler. Bab ini berfokus pada pengenalan mekanisme terbaru dalam mengadopsi pemrograman dinamis dalam sistem tertanam. Skema yang diwakili diberi nama Heterogeneous Embedded Systems (HES) yang dapat digunakan untuk memungkinkan sistem tertanam untuk menyelesaikan pekerjaan dengan biaya sumber daya paling sedikit di bawah batasan waktu tertentu. Dua model sistem tertanam heterogen diperkenalkan dalam bab ini. Isi utama bab ini meliputi:

1. Pemrograman dinamis
2. Sistem tertanam heterogen
3. Model waktu tetap dari sistem tertanam heterogen
4. Model waktu probabilistik sistem tertanam heterogen

Bab ini bermaksud untuk menginstruksikan siswa untuk belajar yang penting mekanisme yang digunakan dalam domain pengoptimalan seluler saat ini, yang HES. Ini adalah mekanisme penting untuk meningkatkan kinerja memori heterogen. Sistem ini didasarkan pada algoritma pemrograman dinamis yang memanfaatkan dan implementasi yang sesuai akan dibahas dalam bab ini. Beberapa contoh dan studi kasus akan diberikan dalam bab ini untuk membantu siswa memahami lebih jauh skema pemrograman dinamis. Sepanjang bab ini, siswa harus dapat menjawab pertanyaan-pertanyaan berikut:

1. Apa itu HES?
2. Apa itu pemrograman dinamis?
3. Apa skema utama pengoptimalan HES?
4. Berapa tahapan utama optimasi HES? Apakah mereka?
5. Bagaimana Anda dapat mengoptimalkan Model Waktu Tetap dari HES?
6. Bagaimana cara mengoptimalkan Model Waktu Probabilistik dari HES?
7. Mengapa masalah optimasi HES dapat dianggap sebagai masalah polinomial semu?
8. Apa yang dimaksud dengan masalah waktu polinomial non-deterministik?

7.1 PENGENALAN SISTEM EMBEDDED HETEROGEN DAN PEMROGRAMAN DINAMIS

Sistem tertanam modern memiliki berbagai fitur, seperti desain tujuan khusus, ketergantungan pada perangkat lain, dan memori yang hanya dapat dibaca. Sistem tertanam mungkin memiliki semua atau beberapa fitur ini, yang bergantung pada tuntutan praktis. Heterogeneous Embedded System (HES) adalah metode penyebaran beberapa unit fungsional dalam sistem tertanam untuk mencapai kinerja tingkat yang lebih tinggi. Dalam perspektif

real-time, heterogenitas adalah salah satu karakteristik yang signifikan. Istilah Heterogenitas mengacu pada beberapa unit operasi dasar, seperti memori, prosesor, dan unit pemrosesan grafis. Mengatasi masalah waktu eksekutif, ada dua jenis HES. Jenis pertama adalah Fixed Time Model (FTM), yang merepresentasikan model yang memiliki nilai tetap untuk setiap waktu eksekusi tugas. Jenis lainnya adalah Probabilistic Time Model (PTM) yang memiliki nilai tidak tetap untuk setiap waktu eksekusi tugas. Nilai tidak tetap didefinisikan sebagai variabel acak di PTM.

Selain itu, biaya proses dapat dikurangi ketika unit fungsional yang heterogen dapat berkolaborasi secara efisien satu sama lain. Dalam ilmu komputer, Unit Fungsional adalah komponen tingkat rendah atau sekelompok komponen yang digunakan untuk tujuan tertentu, seperti fungsi aritmatika, penugasan, operasi logika, atau untuk membandingkan operasi. Beberapa perangkat keras dapat menjadi pembawa unit fungsional, seperti memori, register, atau unit penyimpanan. Manfaat unit fungsional terkait dengan berbagai aspek, seperti energi, keandalan sistem, atau penggunaan infrastruktur. Memahami model ini dapat membantu siswa untuk lebih memahami konsep sistem tertanam dan solusi berbasis heterogenitasnya. Solusi yang efisien biasanya terdiri dari dua fase, termasuk fase Penugasan Heterogen dan fase Meminimalkan Biaya dengan Penjadwalan. Bagian berikut mewakili rincian dari dua fase ini.

Mekanisme penting dari optimasi sistem tertanam heterogen menerapkan pemrograman dinamis. Pemrograman Dinamis adalah pendekatan pemecahan masalah yang rumit dengan membagi masalah menjadi sub-masalah yang berukuran lebih kecil. Pendekatan pemrograman dinamis biasanya digunakan untuk mengoptimalkan sistem atau proses melalui meminimalkan biaya sumber daya di bawah atau sejumlah kendala yang diinginkan. Dalam HES, pemrograman dinamis terutama menawarkan batasan waktu di mana sumber daya komputasi diminimalkan, seperti biaya energi dan risiko keandalan. Konten berikut mencakup FTM HES di Bagian 7.2 dan PTM HES di Bagian 7.3.

7.2 MODEL WAKTU TETAP

Fixed Time Model (FTM) adalah model yang terdiri dari sejumlah node yang semuanya memiliki nilai tetap dengan waktu eksekusi tertentu. Seperti disebutkan dalam Bagian 7.1, optimasi FTM dibentuk oleh dua langkah: fase Penugasan Heterogen dan fase Meminimalkan Biaya dengan Penjadwalan. Tugas Heterogen adalah proses membuat tabel yang mewakili informasi penting mengenai optimasi, seperti unit fungsional dan waktu eksekutif. Tahap kedua, Meminimalkan Biaya dengan Penjadwalan, adalah dengan menggunakan pemrograman dinamis untuk menjadwalkan mode kerja untuk mencapai biaya minimum. Isi berikut akan memberikan pengenalan rinci tentang dua fase ini.

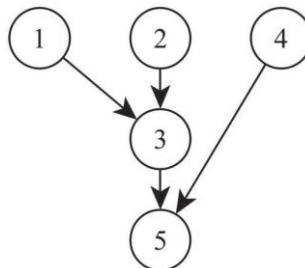
7.2.1 Tugas Heterogen

Bagian ini berfokus pada menjelaskan apa itu Tugas Heterogen dan metode apa yang dapat menyelesaikan pekerjaan. Pada fase pertama, tugas Heterogeneous Assignment adalah menghasilkan tabel dengan informasi jenis proses dan biaya yang sesuai. Biaya dapat dalam berbagai jenis atau cara, seperti energi, waktu, atau probabilitas. Gambar 7.1 merupakan contoh grafik aliran data untuk unit fungsional

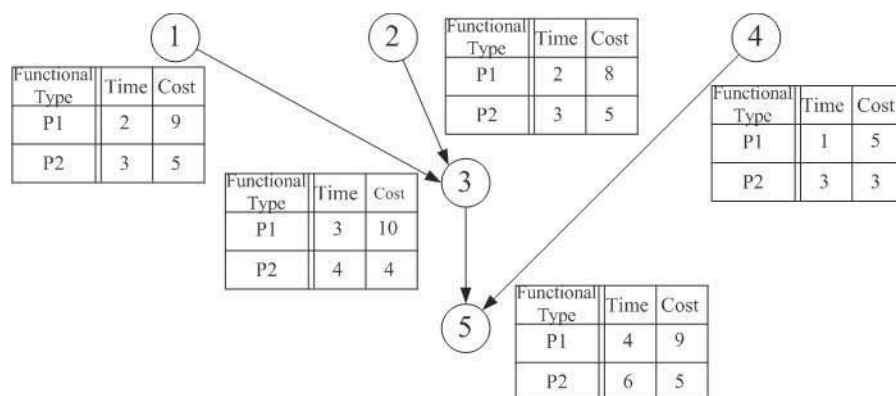
yang digunakan untuk meminimalkan biaya. Data Flow Graph (DFG) adalah pendekatan menggunakan diagram grafik untuk mewakili dependensi data atau unit fungsional antara operasi. Dalam contoh yang diberikan, alur kerja DFG memiliki berbagai node yang menuntut waktu eksekutif dan konsumsi energi yang berbeda saat menggunakan berbagai jenis unit fungsional.

Seperti yang ditampilkan pada Gambar. 7.1, lingkaran dengan nomor kode mewakili node. Untuk menyelesaikan setiap node, mungkin ada berbagai tipe fungsional dengan permintaan konsumsi yang berbeda. Setiap tipe fungsional dapat menyelesaikan tugas tetapi kinerjanya dapat bervariasi. Misalnya, sebuah node dapat berjalan lebih lambat tetapi dengan energi yang lebih sedikit pada satu jenis unit fungsional daripada yang lain.

Gambar 7.2 memberikan contoh DFG dengan dua jenis unit fungsional, yang diasosiasikan dengan Gambar 7.1. Gambar tersebut mengilustrasikan skenario dengan asumsi bahwa dua unit fungsional dipilih dari perpustakaan unit fungsional, yaitu P1 dan P2. Opsi trade-off adalah biaya waktu dan energi, yang ditunjukkan pada gambar. Lima tabel dalam gambar mewakili tipe fungsional dan waktu pencapaian yang sesuai serta konsumsi energi. P mengacu pada jalur dengan menggunakan satu jenis unit fungsional. Nomor yang dilampirkan ke P mewakili tipe fungsional yang tepat, dan parameter yang relevan diberikan dalam kolom berikut. Misalnya, P1 pada simpul 1 membutuhkan biaya energi 9 unit untuk menyelesaikan tugas dalam waktu 2 unit.



Gambar 7.1 Contoh grafik aliran data dengan satuan fungsional.



Gambar 7.2 Contoh grafik aliran data dengan dua jenis unit fungsional.

Menyebarkan unit fungsional yang berbeda pada node dapat menghasilkan kinerja yang berbeda di bawah batasan waktu yang berbeda. Optimasi dapat dilakukan jika kombinasi pemilihan unit fungsional dapat menyebabkan konsumsi energi yang lebih sedikit dalam batasan waktu tertentu. Tujuan ini berarti bahwa output mengenai biaya yang diminimalkan untuk setiap node yang selaras dengan unit fungsional yang sesuai akan dihasilkan. Kendala adalah variabel yang akan menentukan jenis unit fungsional. Mempertimbangkan konfigurasi batasan waktu, unit fungsional dapat dipilih untuk mencapai biaya terendah.

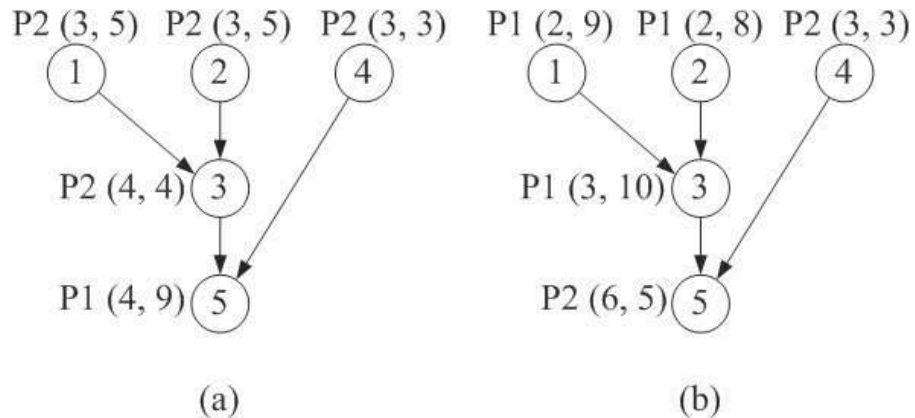
Tabel 7.1 Node dengan Unit Fungsional yang Berbeda.

Nodes	P1		P2	
	T_1	C_1	T_2	C_2
1	2	9	3	5
2	2	8	3	5
3	3	10	4	4
4	1	5	3	3
5	4	9	6	5

Menyelaraskan dengan Gambar 7.1, semua pilihan dapat disintesis ke dalam tabel, yang ditunjukkan pada Tabel 7.1. Dalam tabel, " T_i " mewakili waktu eksekutif dan " C_i " berarti biaya eksekutif untuk setiap unit fungsional tipe P_i . Di antara T dan C , " i " dapat berupa 1 atau 2. Setelah menyelesaikan Tugas Heterogen, menggunakan pendekatan yang diperkenalkan di Bagian 7.2.2 dapat mencapai solusi optimal. Gambar 7.3 mewakili dua cara penjadwalan dengan penugasan yang berbeda di bawah batasan waktu 11. Gambar 7.3 (a) memberikan solusi optimal dengan total biaya 26. Gambar 7.3 (b) juga dapat menyelesaikan pekerjaan dalam batasan waktu 11, tetapi total biaya 35.

7.2.2 Meminimalkan Biaya dengan Penjadwalan

Untuk mencapai tujuan yang disebutkan dalam Bagian 7.2.1, fase kedua diperlukan untuk menemukan pendekatan dengan memilih unit fungsional yang tepat di setiap node dan mendapatkan penjumlahan biaya minimum. Oleh karena itu, fase ini membuat jadwal yang bertujuan untuk menggunakan sumber daya minimum dengan menggunakan hasil yang diperoleh dari fase sebelumnya. Metode untuk mencapai tujuan ini adalah memanfaatkan Minimum Resource Scheduling Algorithm (MRSA). Hasil dari fase ini mengkonfigurasi jumlah dan jenis unit fungsional yang akan digunakan untuk meminimalkan total biaya.



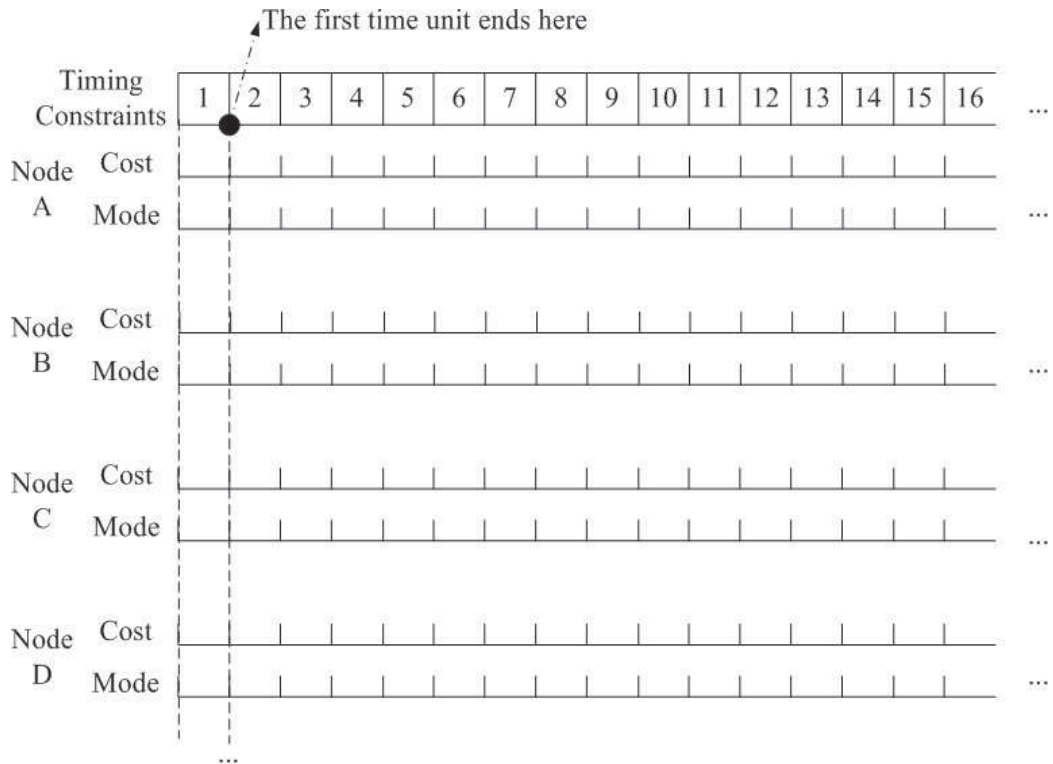
Gambar 7.3 (a) Tugas 1 dengan biaya 26. (b) Tugas 2 dengan biaya 35.

Perhitungan berbasis bentuk ditunjukkan pada Gambar 7.4 untuk membantu siswa memahami MRSA. Menurut struktur yang ditampilkan pada gambar, operasi yang terjadi pada setiap node diselaraskan dengan batasan waktu. Baris pertama mewakili batasan waktu yang dapat menentukan jalur optimal untuk hasil akhir. Bintik hitam pada baris pertama memberikan contoh batasan waktu, yang berarti unit waktu pertama berakhir dengan garis putus-putus yang melekat pada titik hitam. Baris berikut dikelompokkan berdasarkan node yang terdiri dari biaya yang sesuai dan mode kerja. Mode kerja mengacu pada pemilihan unit fungsional. Baris biaya menampilkan jumlah yang tepat dari biaya pada waktu tertentu dengan eksekusi unit fungsional tertentu. Bagian berikut menjelaskan pendekatan penggunaan formulir yang diberikan pada Gambar 7.4.

Contoh pemanfaatan formulir MRSA diberikan, yang terkait dengan contoh yang diberikan di Bagian 7.2.1. Untuk keperluan penjelasan, contoh tersebut merupakan skenario yang memiliki jalur sederhana dengan empat node, yaitu A, B, C, dan D. Setiap node memiliki dua mode kerja yang membutuhkan waktu eksekutif dan biaya energi yang berbeda. Dalam asumsi yang diberikan, mode kerja membutuhkan waktu eksekutif yang lebih lama saat memotong biaya. Misalnya, node A memiliki dua mode kerja, A1 dan A2. A1 dapat menyelesaikan pekerjaan dalam 2 satuan waktu, yang lebih pendek dari A2. Namun, A1 membutuhkan lebih banyak energi daripada A2. Tabel 7.2 mewakili informasi rinci tentang empat node.

Node A

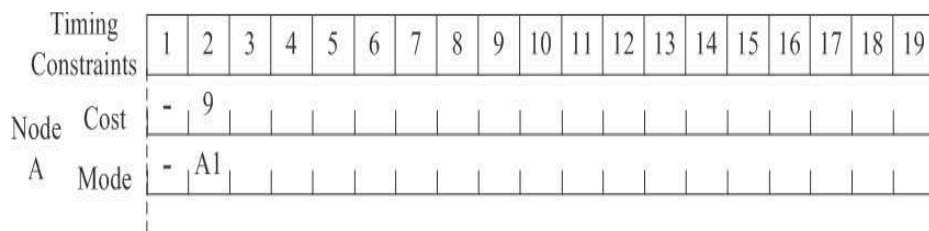
Berdasarkan Tabel 7.2, jumlah waktu maksimum dari jalur $A \rightarrow B \rightarrow C \rightarrow D$ adalah $\max(A, B, C, D) = 19$. Pertama-tama, kita perlu mencari node biaya minimum untuk Node A untuk setiap kendala waktu yang diberikan J . Dimulai dengan batasan waktu, tidak ada node yang dapat melakukan instruksi pada unit waktu pertama. Beralih ke unit waktu kedua, kita dapat menemukan node, Node A1, dapat menyelesaikan pekerjaan dengan 9 unit konsumsi energi. Formulir pada Gambar 7.4 dapat diperbarui untuk Gambar 7.5.



Gambar 7.4 Formulir perhitungan untuk algoritma penjadwalan sumber daya minimum.

Tabel 7.2 Node dengan Dua Mode Kerja Berbeda (Waktu dan Biaya Eksekutif Berbeda)

Node	Waktu	Biaya
A1	2	9
A2	3	5
B1	3	10
B2	4	4
C1	4	9
C2	6	5
D1	2	6
D2	6	2



Gambar 7.5 Pembaruan pertama dari formulir MRSA.

Selanjutnya, pada kendala waktu ketiga, mode kerja A2 dapat menyelesaikan pekerjaan dengan 5 unit biaya, yang lebih kecil dari biaya dengan menggunakan A1, 9

unit biaya. Secara konsisten, A2 memiliki biaya terendah di semua batasan waktu lainnya. Gambar 7.6 merepresentasikan tabel yang diperbarui yang menggunakan mode kerja A2 ketika batasan waktu lebih dari 2.

Timing Constraints	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Node Cost	-	9	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
A Mode	-	A1	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2

Gambar 7.6 Pembaruan kedua dari formulir MRSA.

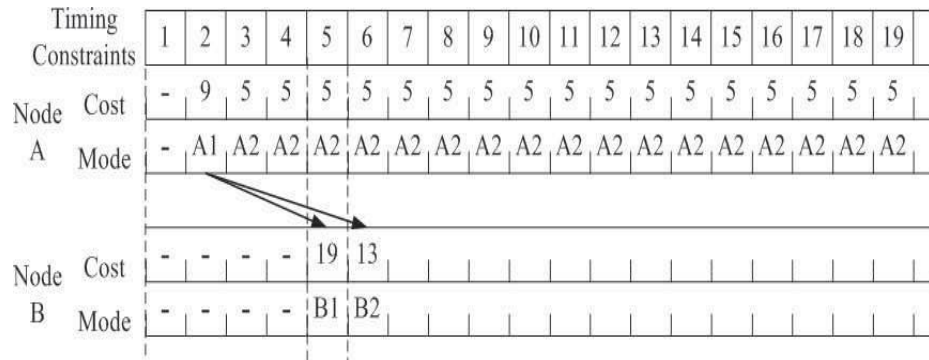
Jalur A → B

Langkah selanjutnya adalah mencari biaya minimum jalur A → B dalam batasan waktu J . Waktu eksekutif terpendek untuk Node B adalah 3 unit waktu yang ditawarkan oleh mode kerja B1, dan B1 hanya dapat dimulai setelah Node A selesai. Jalur paling awal A → B adalah A1 → B1, dan B1 dimulai pada unit waktu kelima. Ini juga berarti bahwa tidak ada jalur untuk batasan waktu kurang dari 5 unit waktu. Gambar 7.7 merepresentasikan jalur pertama dari A ke B.

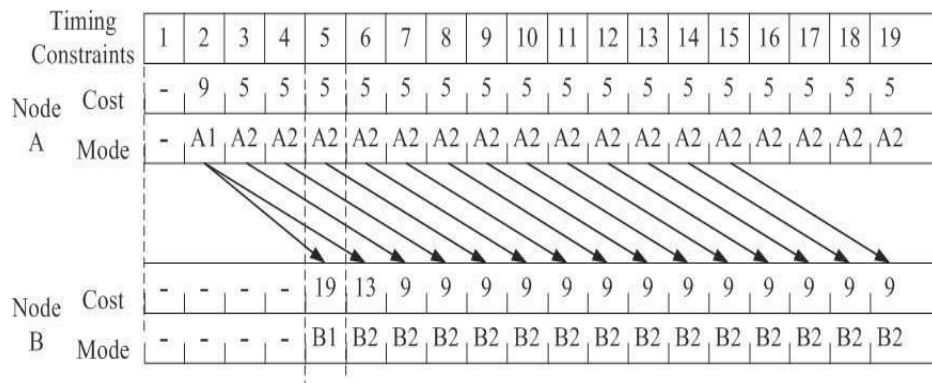
Timing Constraints	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Node Cost	-	9	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
A Mode	-	A1	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2	A2
Node Cost	-	-	-	-	19														
B Mode	-	-	-	-	B1														

Gambar 7.7 Jalur pertama yang mungkin dari A ke B di bawah batasan waktu 5.

Ketika batasan waktu menjadi 6, jalur A → B bisa menjadi A1 → B2. Biaya A1 → B2 adalah 13 yang diturunkan dari (9+4), yang lebih kecil dari jalur A1 → B1 yang bernilai 19. Gambar 7.8 menunjukkan dua jalur A → B di bawah dua batasan waktu, 5 dan 6. Ketika batasan waktu menjadi 7, jalur A → B dapat berupa A2 → B2 dengan biaya 9 diturunkan dari (5+4). Pemilihan ini menggunakan dua mode kerja A2 dan B2 yang membutuhkan lebih sedikit energi tetapi membutuhkan lebih banyak waktu eksekutif. Oleh karena itu, jalur terbaik A → B adalah A2 → B2 dari batasan waktu 7. Gambar 7.9 menunjukkan semua pilihan optimal dari batasan waktu 5.



Gambar 7.8 Lintasan dari A ke B di bawah batasan waktu 6.



Gambar 7.9 Jalur optimal untuk A → B di bawah batasan waktu yang berbeda.

Jalur A → B → C

Langkah selanjutnya adalah menghitung biaya minimum jalur A → B → C di bawah batasan waktu. Mirip dengan jalur A → B, tidak ada jalur untuk batasan waktu yang lebih pendek dari 9 unit waktu karena batasan waktu minimum adalah 4. Gambar 7.10 mengilustrasikan tabel MRSA yang diperbarui dengan jalur optimal potensial B → C.

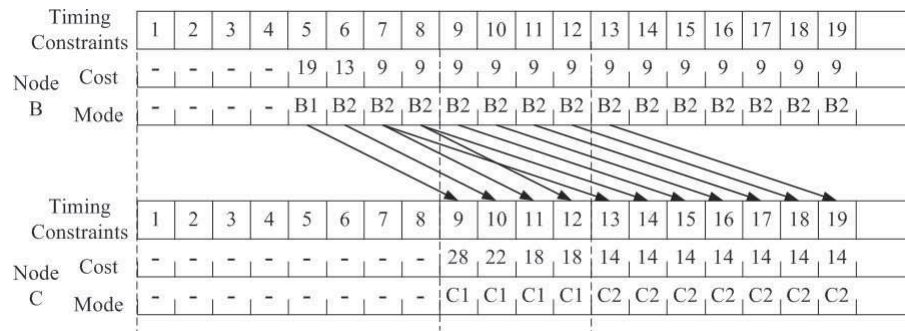
Di bawah batasan waktu 9, jalur B1 → C1 adalah jalur optimal, dengan biaya 28 dari (19+9).

Di bawah batasan waktu 10, jalur B2 → C1 adalah jalur optimal, dengan biaya 22 dari (13+9).

Di bawah batasan waktu 11, jalur B2 → C1 adalah jalur optimal, dengan biaya 18 dari (9+9).

Di bawah batasan waktu 12, jalur B2 → C1 dan jalur B2 → C2 biaya 18.

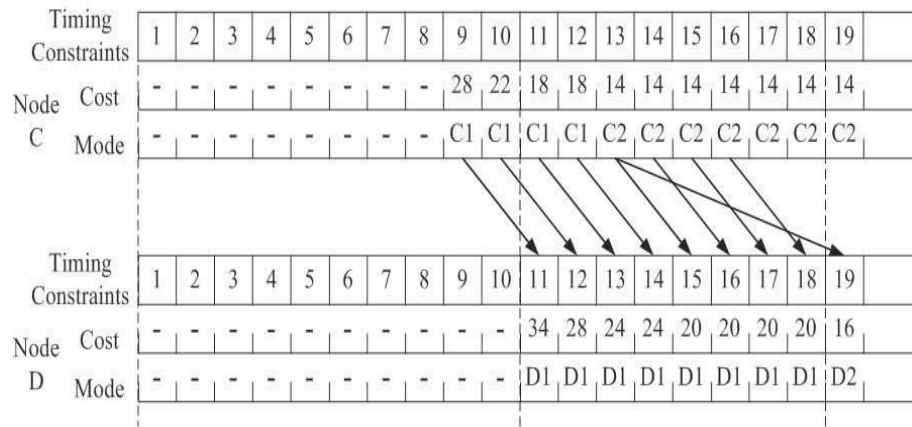
Di bawah batasan waktu 13, jalur B2 → C2 adalah jalur optimal, dengan biaya 14 dari (9+5). Selain itu, jalur B2 → C2 adalah jalur optimal untuk B → C jika batasan waktu lebih panjang dari 13 unit waktu.



Gambar 7.10 Jalur optimal untuk B → C di bawah batasan waktu yang berbeda.

Jalur A → B → C → D

Langkah terakhir adalah menghitung biaya minimum jalur A → B → C → D di bawah batasan waktu. Waktu mulai paling awal untuk D adalah pada batasan waktu ke-11. Gambar 7.10 merepresentasikan jalur C → D.

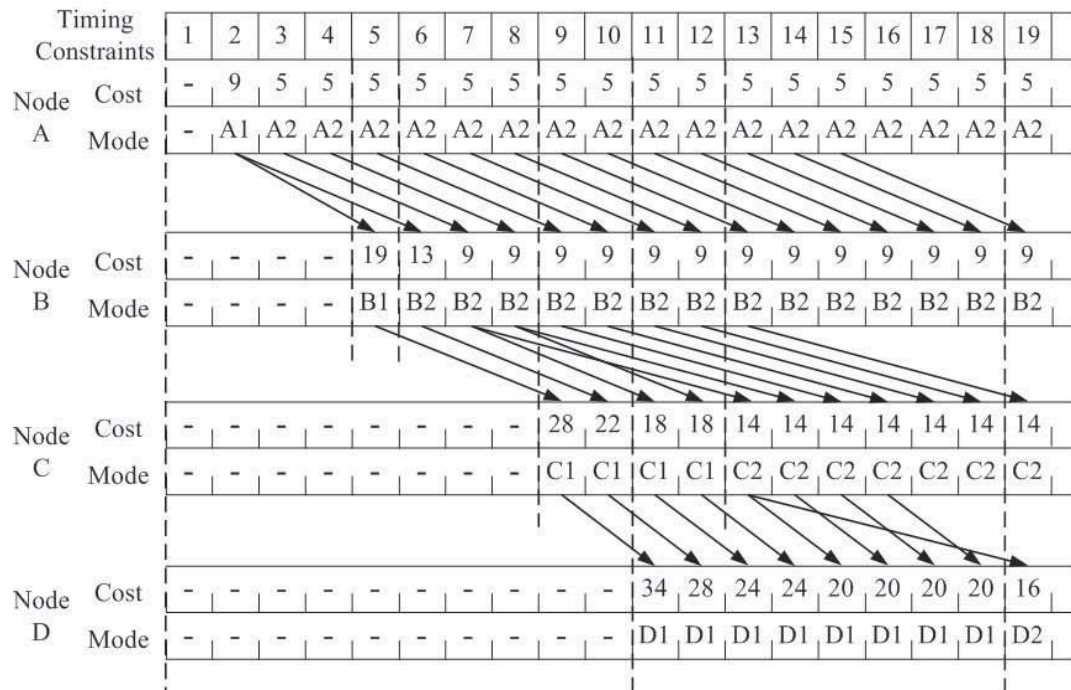


Gambar 7.11 Jalur optimal untuk C → D di bawah batasan waktu yang berbeda

Di bawah batasan waktu 11, jalur C1 → D1 adalah jalur optimal, dengan biaya 34 dari (28+6).
 Di bawah batasan waktu 12, jalur C1 → D1 adalah jalur optimal, dengan biaya 28 dari (22+6).
 Di bawah batasan waktu 13, jalur C1 → D1 adalah jalur optimal, dengan biaya 24 dari (18+6). Jalur optimal yang sama C1 → D1 juga berada di bawah batasan waktu 14.
 Di bawah batasan waktu 15 dan 16, jalur C2 → D1 adalah jalur optimal, dengan biaya 20 dari (14+6).
 Di bawah batasan waktu 17 dan 18, jalur C1 → D2 adalah jalur optimal, dengan biaya 20 dari (18+2). Biaya yang sama juga dapat dicapai dengan jalur C2 → D1, yaitu 20 dari (14+6).

Akhirnya, jalur C1 → D2 adalah jalur optimal, dengan biaya 16 dari (14+2). Seperti yang telah dibahas sebelumnya, 19 unit waktu adalah batasan waktu terpanjang dalam kasus ini. Gambar 7.12 mewakili semua jalur optimal untuk A → B → C → D di bawah batasan waktu yang berbeda. Menurut gambar, waktu penyelesaian

paling awal adalah 11 unit waktu. Rentang batasan waktu untuk semua mode kerja adalah antara 11 dan 19 unit waktu. Ringkasnya, MRSA adalah pendekatan yang efisien untuk memecahkan Masalah Penugasan Heterogen jika DFG adalah jalur sederhana atau pohon.



Gambar 7.12 Jalur optimal untuk A → B → C → D di bawah batasan waktu yang berbeda

Kompleksitas masalah terkait dengan nilai waktu eksekutif maksimum node sehingga masalah dapat dianggap sebagai masalah polinomial semu. Metode ini efisien bila nilainya kecil atau dapat dinormalisasi menjadi kecil. Pada bagian berikutnya, kami memperkenalkan Model Waktu Probabilistik Sistem Tertanam Heterogen.

7.3 MODEL WAKTU PROBABILISTIK

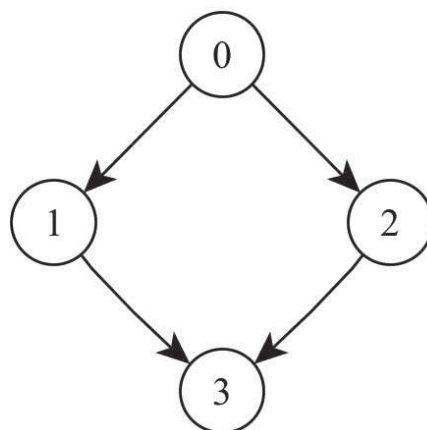
7.3.1 Pengenalan Model Waktu Probabilistik

Berdasarkan algoritma yang diberikan di Bagian 7.2, model lanjutan diperkenalkan di bagian ini, yaitu Model Waktu Probabilistik. Bagian ini memperkenalkan algoritma optimasi yang beroperasi di lingkungan probabilistik untuk memecahkan masalah Heterogeneous Assignment with Probability (HAP). Model Waktu Probabilistik adalah model yang terdiri dari sejumlah node yang dibatasi oleh waktu eksekutif bersyarat yang terkait dengan beberapa input atau penggunaan perangkat keras. Dalam program HAP, waktu eksekutif tugas dimodelkan sebagai variabel acak. Unit fungsional dalam Sistem Heterogen memiliki biaya yang berbeda dalam berbagai cara, seperti ukuran dan keandalan. Setiap unit fungsional dikaitkan dengan jenis biaya, seperti konsumsi waktu dan biaya energi, dalam sistem heterogen.

Secara umum, kinerja unit fungsional memiliki hubungan positif dengan biaya tipikal. Kinerja setiap unit fungsional dikaitkan dengan satu atau beberapa jenis biaya. Dengan probabilitas keyakinan P , kami dapat menjamin bahwa total waktu eksekusi DFG kurang dari atau sama dengan batasan waktu dengan probabilitas yang lebih besar atau sama dengan P .

Dalam skenario praktis, banyak tugas yang diproses pada aplikasi tertanam dibatasi oleh waktu eksekutif yang tetap. Tugas biasanya terdiri dari instruksi bersyarat, serta operasi dengan berbagai waktu eksekusi untuk input yang berbeda. Pendekatan yang ada tidak efisien untuk memecahkan masalah dengan ketidakpastian. Solusi saat ini adalah membuat asumsi baik untuk kasus terburuk atau kasus rata-rata konsumsi waktu yang dibutuhkan oleh tugas. Namun, metode ini menghasilkan implementasi sistem tertanam yang tidak efisien dalam merancang sistem waktu nyata yang lembut.

Misalnya, unit berkinerja lebih tinggi mungkin lebih mahal dalam penggunaan energi tetapi dapat dieksekusi lebih cepat. Metode menetapkan tipe unit fungsional yang sesuai ke dalam simpul DFG adalah langkah penting untuk meminimalkan total biaya. Ini membutuhkan batasan waktu yang dipenuhi oleh Guaranteed Confidence Probability (GCP). GCP adalah pernyataan kemungkinan yang dapat menjamin tugas yang ditargetkan dapat diselesaikan dengan batasan waktu yang diberikan. Dengan mendefinisikan probabilitas keyakinan P , dijamin bahwa total waktu eksekutif DFG sama dengan atau kurang dari batasan waktu dengan nilai yang lebih besar. Sebuah contoh diberikan di bagian ini untuk membantu siswa memahami sepenuhnya solusi untuk masalah HAP. Sebuah asumsi dibuat untuk menggambarkan skenario mengatasi masalah, yang memilih unit fungsional dari perpustakaan unit fungsional yang menyediakan dua jenis unit fungsional. R1 dan R2 mewakili dua tipe unit fungsional ini. Gambar 7.13 menunjukkan Probabilistic Data Flow Graph (PDFG) contoh yang menunjukkan pohon dengan empat node.



Gambar 7.13 Sebuah grafik aliran data probabilistik yang diberikan.

Tabel 7.4 merepresentasikan fungsi distribusi kumulatif waktu dan biaya pada setiap node dengan menggunakan tipe unit fungsional yang berbeda. T mengacu pada waktu eksekusi, C berarti biaya, dan P mewakili probabilitas. Setiap node dapat

memilih salah satu dari dua tipe unit fungsional dan mengeksekusi setiap tipe unit fungsional dengan waktu eksekutif probabilistik. Ini menyiratkan bahwa waktu eksekusi (T) dari setiap unit fungsional harus berupa variabel acak. Misalnya node 1 dapat dieksekusi dalam 1 unit waktu dengan probabilitas 0,9 (90%) atau dieksekusi dalam 3 unit waktu dengan probabilitas 0,1 (10%) saat mengimplementasikan R1. Ini juga berarti bahwa eksekusi node 1 dapat diselesaikan dalam 3 unit waktu dengan probabilitas 100%.

Tabel 7.3 Node dengan Waktu, Probabilitas, dan Biaya dengan Menggunakan Unit Fungsional yang Berbeda.

Node	R1			R2		
	T_1	P_1	C_1	T_2	P_2	C_2
0	1	0.8	9	2	0.8	5
	2	0.2	9	3	0.2	5
1	1	0.9	10	2	0.7	4
	3	0.1	10	4	0.3	4
2	1	0.9	9	2	0.8	5
	4	0.1	9	6	0.2	5
3	1	0.2	8	3	0.4	2
	2	0.8	8	6	0.6	2

Tabel 7.4 merepresentasikan fungsi distribusi kumulatif waktu dan biaya pada setiap node dengan menggunakan tipe unit fungsional yang berbeda. Tabel tersebut menggunakan notasi yang mirip dengan Tabel 7.3, seperti T dan C . P mewakili probabilitas kumulatif. Setiap node dapat memilih salah satu dari dua tipe unit fungsional dan mengeksekusi setiap tipe unit fungsional dengan waktu eksekutif probabilistik. Ini menyiratkan bahwa waktu eksekusi (T) dari setiap unit fungsional harus berupa variabel acak. Misalnya, node 1 dapat dieksekusi dalam 1 unit waktu, dengan probabilitas 0,9, atau dieksekusi dalam 3 unit waktu, dengan probabilitas kumulatif 1,0 ketika menerapkan R1. Berdasarkan nilai yang diberikan pada tabel, waktu penyelesaian terpendek adalah 4 kali unit, dan waktu penyelesaian terlama adalah 19 kali unit.

Efisiensi solusi yang diperkenalkan pada bagian ini telah dibuktikan pada penelitian sebelumnya. Algoritme dapat memecahkan masalah Hard Heterogeneous Assignment (HA) untuk situasi waktu nyata yang sulit dan lunak. Selain itu, algoritme mampu memberikan lebih banyak pilihan dengan Probabilitas Keyakinan Terjamin dan biaya total yang lebih rendah, dan beberapa pilihan merupakan solusi yang efisien untuk beberapa batasan waktu tertentu.

Tabel 7.4 Fungsi Distribusi Kumulatif Waktu dan Biaya pada Setiap Node dengan Menggunakan Jenis Unit Fungsional yang Berbeda

Nodes	R1			R2		
	T_1	P_1	C_1	T_2	P_2	C_2
0	1	0.8	9	2	0.8	5
	2	1.0	9	3	1.0	5
1	1	0.9	10	2	0.7	4
	3	1.0	10	4	1.0	4
2	1	0.9	9	2	0.8	5
	4	1.0	9	6	1.0	5
3	1	0.2	8	3	0.4	2
	2	1.0	8	6	1.0	2

Pengembangan penjadwalan heterogen probabilistik dan kerangka penugasan dapat secara sederhana diterapkan untuk memecahkan masalah penskalaan tegangan dinamis dengan menggunakan Model Grafik Probabilistik. Salah satu pendekatan yang ada adalah menemukan tingkat tegangan yang tepat untuk setiap node yang dapat dikaitkan dengan waktu eksekusi yang berbeda dan konsumsi energi dengan probabilitas. Solusi kami menggunakan penjadwalan loop dengan arsitektur multicore yang menawarkan kinerja tingkat lanjut. Solusinya menganggap setiap level tegangan sebagai tipe unit fungsional, dengan mendefinisikan konsumsi energi yang diharapkan sebagai biayanya. Menggunakan kerangka kerja HAP dapat mengubah masalah ini menjadi kasus khusus dari masalah HAP. Berikut ini adalah contoh penjelasan implementasi kerangka HAP.

7.3.2 Solusi untuk Masalah Penugasan Heterogen

Bagian ini bertujuan untuk memecahkan masalah HAP dengan data yang disediakan oleh Tabel 7.4. Ada dua fase untuk memecahkan masalah HAP, termasuk Membangkitkan Tabel B dan Meminimalkan Biaya dengan Penjadwalan D Tabel. Tabel B adalah tabulasi yang dapat mewakili biaya dan probabilitas penyelesaian simpul di bawah batasan waktu yang berbeda. Tabel 7.5 menampilkan tabel B yang sejajar dengan Tabel 7.4.

Dalam tabel B, tipe unit fungsional direpresentasikan dengan cara (P_i , C_i). " P_i " mengacu pada probabilitas, dan " C_i " adalah biaya pada node di bawah satu kendala waktu tertentu. Seperti ditunjukkan pada Tabel 7.5, baris pertama menunjukkan batasan waktu. Dalam hal ini, batasan waktu terpanjang untuk setiap node adalah 6 unit waktu. Di bawah setiap batasan waktu, sebuah node dapat memiliki satu atau dua mode kerja dengan menggunakan unit fungsional yang berbeda. Misalnya, di bawah batasan waktu 2, Node 0 memiliki dua mode, yaitu (0.8, 5) dan (1.0, 9). Ini berarti kedua mode kerja dapat menyelesaikan node di bawah batasan waktu 2 tetapi kinerjanya bervariasi. Modus pertama, (0,8, 5), memiliki tingkat probabilitas 80% dan biaya 5 unit sumber daya. Modus lainnya, (1,0, 9), memiliki tingkat probabilitas yang

lebih tinggi yaitu 100%, tetapi biayanya lebih tinggi daripada modus pertama. Selain itu, di bawah batasan waktu 1, Node 2 hanya memiliki satu mode, (0.9, 9), yang berarti hanya ada satu opsi mode di bawah batasan waktu ini untuk Node 2.

Tabel 7.5 Contoh Tabel B (Berkaitan dengan Tabel 7.4). TC: Batasan Waktu.

TC	1	2	3	4	5	6
Node 0	(0.8, 9)	(0.8, 5)	(1.0, 5)			
		(1.0, 9)				
Node 1	(0.9, 10)	(0.9, 10)	(1.0, 10)	(1.0, 4)		
		(0.7, 4)				
Node 2	(0.9, 9)	(0.8, 5)	(0.8, 5)	(0.8, 5)	(0.8, 5)	(1.0, 5)
		(0.9, 9)	(0.9, 9)	(1.0, 9)	(1.0, 9)	
Node 3	(0.2, 8)	(1.0, 8)	(0.4, 2)	(0.4, 2)	(0.4, 2)	(1.0, 2)
			(1.0, 8)	(1.0, 8)	(1.0, 8)	

7.3.3 Membuat Tabel D

Tabel D adalah tabulasi yang mewakili solusi optimal dengan probabilitas dan biaya di bawah batasan waktu yang berbeda, yang dihasilkan dengan menggunakan pemrograman dinamis. Metode pembuatan tabel D menggunakan data dari tabel B dan memilih jalur tergantung pada kriteria yang diminta. Tabel 7.6 memberikan contoh tabel D, dengan nilai yang diturunkan dari Tabel 7.5. Pada tabel, V_k mewakili node, $X(k, 1)$, yang sejajar dengan Tabel 7.5. Misalnya, V_1 mewakili Node 0 dan V_2 mewakili Node 1. Di bawah setiap batasan waktu yang ditunjukkan di kolom, semua potensi kombinasi probabilitas dan biaya perlu dicantumkan, yang berkorelasi dengan V_k baris.

Tabel 7.6 Contoh Tabel D (Terkait dengan Tabel 7.4 dan 7.5). TC (Batasan Waktu)

TC	1	2	3	4	5	6	...
V1	(0.8, 9)	(1.0, 9)	(1.0, 5)				...
		(0.8, 5)					...
V2		(0.72, 19)	(0.56, 13)	(0.7, 13)	(0.8, 13)	(1.0, 13)	...
			(0.9, 19)	(0.56, 9)	(1.0, 19)	(0.8, 9)	...
			(0.72, 15)	(0.9, 15)			...
...

Metode penghitungan nilai untuk setiap sel dalam tabel D diberikan dalam Definisi 1. Dalam definisi, $(P_{(i,j)}^k, C_{(i,j)}^k)$ mengacu pada pasangan (Probabilitas, Biaya). i mewakili jalur mode kerja. j mewakili eksekusi waktu yang tepat. k mewakili node yang terlibat dalam operasi. Simbol \odot adalah singkatan dari Operator yang menggabungkan dua pasang (Probabilitas, Biaya).

Definisi 1

$$(P_{(i,j)}^k, C_{(i,j)}^k) \odot (P_{(i,j)}^{k+1}, C_{(i,j)}^{k+1}) = (P_{(i,j)}^k \times P_{(i,j)}^{k+1}, C_{(i,j)}^k + C_{(i,j)}^{k+1})$$

Menggunakan operator \odot dapat menghasilkan pasangan baru (P', C') diikuti dengan Algoritma 1. Misalnya, dengan asumsi dua pasang H_1 dan H_2 , $(P_{(i,j)}^1, C_{(i,j)}^1)$ dan $(P_{(i,j)}^2, C_{(i,j)}^2)$, pasangan baru (P', C') dapat dibangkitkan dari $P' = P_{(i,j)}^1 \times P_{(i,j)}^2$ dan $C' = C_{(i,j)}^1 + C_{(i,j)}^2$. Proses operasi ini dilambangkan dengan $H_1 \odot H_2$. Prinsip operasi mengikuti aturan:

1. Total biaya diperoleh dengan menjumlahkan semua biaya node.
2. Probabilitas melekat pada total biaya berasal dari mengalikan probabilitas semua node.

Algoritma yang mewakili aturan-aturan ini diberikan dalam Algoritma 1. Operasi yang terjadi pada operator mengikuti proses dan metode yang diberikan oleh algoritma.

Algoritma 1 Algoritma operator untuk menggabungkan pasangan

Membutuhkan: Daftar paris $(P_{(i,j)}^k, C_{(i,j)}^k)$

Pastikan: Buat daftar bebas-pasangan redundan (P, C)

1. Mengurutkan daftar yang diberikan dengan membandingkan $P(i,j)$ dalam urutan menaik dan $(P_{(i,j)}^k \leq C_{(i,j)}^{(k+1)})$
2. FOR setiap dua pasangan tetangga $(P_{(i,j)}^k, C_{(i,j)}^k)$ dan $(P_{(i,j)}^{(k+1)}, C_{(i,j)}^{(k+1)})$
3. IF peluang pasangan bertetangga sama, $(P_{(i,j)}^k = P_{(i,j)}^{(k+1)})$
4. IF total biaya dengan probabilitas lebih rendah, $C_{(i,j)}^k$, tidak kurang daripada total biaya pasangan tetangga dengan probabilitas lebih tinggi, $C_{(i,j)}^{(k+1)}$, $C_{(i,j)}^k \geq C_{(i,j)}^{(k+1)}$
5. lepaskan pasangan $(P_{(i,j)}^k, C_{(i,j)}^k)$
6. ELSE
7. lepaskan pasangan $(P_{(i,j)}^k, C_{(i,j)}^k)$
8. ELSE IF
9. ELSE /* pasangan lain dengan probabilitas berbeda dari pasangan bertetangga */
10. IF total biaya dalam pasangan probabilitas yang lebih rendah tidak kurang dari total biaya dalam pasangan probabilitas yang lebih tinggi, $C_{(i,j)}^k \geq C_{(i,j)}^{(k+1)}$
11. lepaskan pasangan $C_{(i,j)}^k \geq C_{(i,j)}^{(k+1)}$
12. END IF
13. END FOR

Bagian berikut memberikan contoh rinci yang menjelaskan proses menghasilkan Tabel D langkah demi langkah. Contoh ilustrasi menggunakan nilai yang sama seperti Tabel B yang ditunjukkan pada Tabel 7.5. Setelah Tabel D dibuat, kami menggunakan

pemrograman dinamis untuk mencari solusi optimal ke belakang. Karena biaya dipetakan dalam Tabel D, peningkatan eksekusi adalah linier.

7.3.4 Contoh Pembuatan Tabel D

Menyelaraskan dengan Algoritma 1, Tabel D dapat dibuat dengan langkah-langkah berikut. Dalam instruksi berikut, simbol \odot juga digunakan untuk mewakili operator yang menggabungkan dan membandingkan probabilitas dan biaya. Berdasarkan pada data yang diberikan pada Tabel 7.5, ada tiga langkah untuk menghasilkan daftar pasangan, termasuk **Jalur V1 → V2, Jalur V2 → V3, dan Jalur V3 → V4.**

Membuat Tabel D untuk Jalur V1 → V2

Pada awal konstruksi Tabel D, baris node pertama V0 diisi langsung oleh data dari Tabel 7.5, mengacu pada Tabel 7.6. Setelah kolom V0 terisi, jalur V0 → V1 akan dihasilkan di bawah batasan waktu yang berbeda. Pemilihan pasangan tetangga didasarkan pada batasan waktu yang diberikan. Jumlah waktu penyelesaian dari dua pasangan tidak boleh lebih dari batasan waktu. Di bawah batasan waktu 2, hanya ada satu kemungkinan dalam kasus ini:

$$TC\ 1+1: (0.8, 9) \odot (0.9, 10) = (0.8 \times 0.9, 9+10) = (0.72, 19)$$

Bagian ini menggunakan format “Nomor TC a +Nomor b” untuk mewakili pasangan. Nomor a mewakili biaya waktu dari node sebelumnya. Angka b mewakili biaya waktu dari node berikutnya. Nilai dari “Angka a + Angka b” sama dengan batasan waktu yang sesuai. Rincian probabilitas di bawah berbagai batasan waktu diberikan sebagai berikut:

Di bawah **batasan waktu 3**, probabilitas berikut meliputi:

$$TC\ 1+2: (0.8, 9) \odot (0.9, 10) = (0.8 \times 0.9, 9+10) = (0.72, 19)$$

$$TC\ 1+2: (0.8, 9) \odot (0.7, 4) = (0.8 \times 0.7, 9+4) = (0.56, 13)$$

$$TC\ 2+1: (0.8, 5) \odot (0.9, 10) = (0.8 \times 0.9, 5+10) = (0.72, 15)$$

$$TC\ 2+1: (1.0, 9) \odot (0.9, 10) = (1.0 \times 0.9, 9+10) = (0.9, 19)$$

Operasi di atas mewakili proses aritmatika dengan detail. Menurut Definisi 1, (0,72, 19) dihilangkan, karena pasangan (0,72, 15) harganya lebih murah dengan probabilitas yang sama. Pasangan lain dipilih untuk daftar pasangan, termasuk (0,56, 13), (0,9, 19), dan (0,72, 15). Representasi berikut akan menyembunyikan proses aritmatika dan langsung menampilkan hasilnya.

Di bawah **batasan waktu 4**, probabilitas dihasilkan sebagai berikut:

$$TC\ 1+3: (0.8, 9) \odot (1.0, 10) = (0.8, 19)$$

$$TC\ 2+2: (0.8, 5) \odot (0.9, 10) = (0.72, 15)$$

$$TC\ 2+2: (0.8, 5) \odot (0.7, 4) = (0.56, 9)$$

$$TC\ 2+2: (1.0, 9) \odot (0.9, 10) = (0.9, 19)$$

$$TC\ 2+2: (1.0, 9) \odot (0.7, 4) = (0.7, 13)$$

$$TC\ 3+1: (1.0, 5) \odot (0.9, 10) = (0.9, 15)$$

Menurut Definisi 1, pasangan berikut dihilangkan, termasuk (0,8, 19), (0,9, 19), dan (0,72, 15). Pasangan yang tersisa termasuk (0,7, 13), (0,56, 9), dan (0,9, 15).

Di bawah **batasan waktu 5**, probabilitas dihasilkan sebagai berikut:

$$TC\ 1+4: (0.8, 9) \odot (1.0, 4) = (0.8, 13)$$

$$TC\ 2+3: (0.8, 5) \odot (1.0, 10) = (0.8, 15)$$

$$TC\ 2+3: (1.0, 9) \odot (1.0, 10) = (1.0, 19)$$

$$TC\ 3+2: (1.0, 5) \odot (0.9, 10) = (0.9, 15)$$

$$TC\ 3+2: (1.0, 5) \odot (0.7, 4) = (0.7, 9)$$

Menurut Definisi 1, pasangan (0.8, 15) dihilangkan dan pasangan yang tersisa termasuk (0.8, 13), (1.0, 19), (0.9, 15), dan (0.7, 9).

Di bawah **batasan waktu 6**, probabilitasnya meliputi:

$$TC\ 2+4: (0.8, 5) \odot (1.0, 4) = (0.8, 9)$$

$$TC\ 2+4: (1.0, 9) \odot (1.0, 4) = (1.0, 13)$$

$$TC\ 3+3: (1.0, 5) \odot (1.0, 10) = (1.0, 15)$$

Menurut Definisi 1, pasangan (1.0, 15) dihilangkan. Pasangan yang diperoleh termasuk (0.8, 9) dan (1.0, 13).

Di bawah **batasan waktu 7**, hanya ada satu kemungkinan untuk jalur $V1 \rightarrow V2$:

$$TC\ 3+4: (1.0, 5) \odot (1.0, 4) = (1.0, 9)$$

Tabel D yang mempertimbangkan jalur $V1 \rightarrow V2$ dihasilkan seperti yang ditunjukkan pada Tabel 7.7.

Membuat Tabel D untuk Jalur $V2 \rightarrow V3$

Memfaatkan pendekatan yang sama seperti yang menghasilkan Jalur $V0 \rightarrow V1$ dapat membuat daftar probabilitas berikut.

Tabel 7.7 D Tabel Pengalamatan Jalur $V1 \rightarrow V2$

T	(P, C)	(P, C)	(P, C)	(P, C)
2	(0.72, 19)			
3	(0.56, 13)	(0.72, 15)	(0.9, 19)	
4	(0.56, 9)	(0.7, 13)	(0.9, 15)	
5	(0.7, 9)	(0.8, 13)	(0.9, 15)	(1.0, 19)
6	(0.8, 9)	(1.0, 13)		
7	(1.0, 9)			

Membuat Tabel D untuk Jalur $V3 \rightarrow V4$

Membuat Tabel D untuk jalur $V3 \rightarrow V4$ menggunakan metode yang sama untuk membuat jalur $V2 \rightarrow V3$ dari Tabel D. Dalam hal ini, prosedur tidak akan selesai sampai mencapai batasan waktu terlama, yaitu 19.

1. Di bawah batasan waktu 4:

$$\text{Hanya ada satu jalur, } TC\ 3+1: (0.65, 28) \odot (0.2, 8) = (0.1296, 36)$$

2. Di bawah batasan waktu 5:

$$TC\ 3+2: (0.648, 28) \odot (1.0, 8) = (0.648, 36)$$

$$TC\ 4+1: (0,504, 22) \odot (0,2, 8) = (0,1, 30)$$

$$TC\ 4+1: (0,648, 24) \odot (0,2, 8) = (0,13, 32)$$

$$TC\ 4+1: (0,81, 28) \odot (0,2, 8) = (0,16, 36)$$

Pasangan yang dipilih termasuk (0.648)

3. Di bawah batasan waktu 6:

$$TC\ 3+3: (0,648, 28) \odot (0,4, 2) = (0,2592, 30)$$

$$TC\ 3+3: (0,648, 28) \odot (1,0, 8) = (0,648, 36)$$

$$TC\ 4+2: (0,504, 22) \odot (1,0, 8) = (0,504, 30)$$

$$TC\ 4+2: (0,648, 24) \odot (1,0, 8) = (0,648, 32)$$

$$TC\ 4+2: (0,81, 28) \odot (1,0, 8) = (0,81, 36)$$

$$TC\ 5+1: (0,504, 18) \odot (0,2, 8) = (0,1152, 26)$$

$$TC\ 5+1: (0,576, 20) \odot (0,2, 8) = (0,126, 28)$$

$$TC\ 5+1: (0,63, 22) \odot (0,2, 8) = (0,126, 30)$$

$$TC\ 5+1: (0,81, 24) \odot (0,2, 8) = (0,162, 32)$$

Pasangan yang dipilih meliputi (0.1152, 28), (0.1008, 26), (0.504, 30), (0,81, 36), dan (0,648, 32).

(0,126, 30) dan (0,2592, 30) dihapus, karena mereka memiliki lebih sedikit probabilitas dari pasangan (0,504, 30). (0,162, 32) dihilangkan, karena memiliki kemungkinan lebih kecil dari pasangan (0,648, 32). (0,648, 36) dihapus, karena harganya lebih dari (0,648, 32).

4. Di bawah batasan waktu 7:

$$TC\ 3+4: (0.648, 28) \odot$$

$$(0.4, 2) = (0.2592, 30)$$

$$TC\ 3+4: (0.648, 28) \odot$$

$$(1.0, 8) = (0.648, 36)$$

$$TC\ 4+3: (0.504, 22) \odot$$

$$(0.4, 2) = (0.2016, 24)$$

$$TC\ 4+3: (0.504, 22) \odot$$

$$(1.0, 8) = (0.504, 30)$$

$$TC\ 4+3: (0.648, 24) \odot$$

$$(0.4, 2) = (0.2592, 26)$$

$$TC\ 4+3: (0.648, 24) \odot$$

$$(1.0, 8) = (0.648, 32)$$

$$TC\ 4+3: (0.81, 28) \odot (0.4,$$

$$2) = (0.324, 30)$$

$$TC\ 4+3: (0.81, 28) \odot (1.0,$$

$$8) = (0.81, 36)$$

$$TC\ 5+2: (0.504, 18) \odot$$

$$(1.0, 8) = (0.504, 26)$$

$$TC\ 5+2: (0.578, 20) \odot$$

$$(1.0, 8) = (0.576, 28)$$

$$TC\ 5+2: (0.63, 22) \odot (1.0,$$

$$8) = (0.63, 30)$$

$$TC\ 5+2: (0.81, 24) \odot (1.0,$$

$$8) = (0.81, 32)$$

$$TC\ 6+1: (0.448, 14) \odot$$

$$(0.2, 8) = (0.0896, 22)$$

$$TC\ 6+1: (0.63, 18) \odot (0.2,$$

$$8) = (0.126, 26)$$

$$TC\ 6+1: (0.72, 20) \odot (0.2,$$

$$8) = (0.144, 28)$$

$$TC\ 6+1: (0.81, 24) \odot (0.2,$$

$$8) = (0.162, 32)$$

$$TC\ 6+1: (0.9, 28) \odot (0.2,$$

$$8) = (0.18, 36)$$

Pasangan yang dipilih meliputi (0,2016, 24), (0,576, 28), (0,63, 30), (0,504, 26), (0,81, 32), dan (0,0896, 22).

Pair (0,2592, 30), (0,504, 30), dan (0,324, 30) dihilangkan, karena semuanya memiliki probabilitas yang lebih kecil daripada pair (0,63, 30). (0,81, 36) dihapus, karena biayanya lebih dari (0,81, 32). (0,126, 32) dihilangkan, karena memiliki probabilitas lebih rendah dari (0,81, 32). (0,648, 36) dan (0,18, 36) dihilangkan, karena memiliki probabilitas lebih rendah daripada pasangan (0,81, 36), yang telah dihilangkan. (0,648, 36) dihilangkan, karena memiliki probabilitas yang lebih rendah tetapi tingkat biaya yang lebih tinggi daripada (0,81, 32). Pair (0,2592, 26) dan (0,126, 26) dihilangkan, karena keduanya memiliki probabilitas yang lebih rendah daripada (0,504, 26). (0,144, 28) dihilangkan, karena memiliki probabilitas lebih rendah dari (0,576, 28).

5. Di bawah batasan waktu 8:

TC 3+5: (0.648, 28) ⊙ (0.4, 2) = (0.2592, 30)	TC 5+3: (0.63, 22) ⊙ (0.4, 2) = (0.252, 24)
TC 3+5: (0.648, 28) ⊙ (1.0, 8) = (0.648, 36)	TC 5+3: (0.63, 22) ⊙ (1.0, 8) = (0.63, 30)
TC 4+4: (0.504, 22) ⊙ (0.4, 2) = (0.2016, 24)	TC 5+3: (0.81, 24) ⊙ (0.4, 2) = (0.324, 26)
TC 4+4: (0.504, 22) ⊙ (1.0, 8) = (0.504, 30)	TC 5+3: (0.81, 24) ⊙ (1.0, 8) = (0.81, 32)
TC 4+4: (0.648, 24) ⊙ (0.4, 2) = (0.2592, 26)	TC 6+2: (0.448, 14) ⊙ (1.0, 8) = (0.448, 22)
TC 4+4: (0.648, 24) ⊙ (1.0, 8) = (0.648, 32)	TC 6+2: (0.63, 18) ⊙ (1.0, 8) = (0.63, 26)
TC 4+4: (0.81, 28) ⊙ (0.4, 2) = (0.324, 30)	TC 6+2: (0.72, 20) ⊙ (1.0, 8) = (0.72, 28)
TC 4+4: (0.81, 28) ⊙ (1.0, 8) = (0.81, 36)	TC 6+2: (0.81, 24) ⊙ (1.0, 8) = (0.81, 32)
TC 5+3: (0.504, 18) ⊙ (0.4, 2) = (0.2016, 20)	TC 6+2: (0.9, 28) ⊙ (1.0, 8) = (0.9, 36)
TC 5+3: (0.504, 18) ⊙ (1.0, 8) = (0.504, 26)	TC 7+1: (0.56, 14) ⊙ (0.2, 8) = (0.112, 22)
TC 5+3: (0.576, 20) ⊙ (0.4, 2) = (0.2304, 22)	TC 7+1: (0.72, 18) ⊙ (0.2, 8) = (0.144, 26)
TC 5+3: (0.576, 20) ⊙ (1.0, 8) = (0.576, 28)	TC 7+1: (0.9, 22) ⊙ (0.2, 8) = (0.18, 30)

Pasangan yang dipilih antara lain (0,81, 32), (0,2016, 20), (0,448, 22), (0,72, 28), (0,9, 36), dan (0,63, 26).

Pasangan (0,648, 36) dan (0,81, 36) dihilangkan karena keduanya memiliki probabilitas lebih rendah dari pasangan (0,9, 36). (0,648, 32) dihilangkan, karena probabilitasnya lebih rendah daripada (0,81, 32). Pasangan (0,2592, 30), (0,504, 30), (0,324, 30), (0,63, 30), dan (0,18, 30) dihilangkan, karena memiliki probabilitas lebih rendah tetapi biaya lebih tinggi daripada pasangan (0,72, 28). (0,576, 28) dihilangkan, karena memiliki probabilitas lebih rendah dari (0,72, 28). Pair (0,2304, 22) dan (0,112, 22) dihilangkan, karena memiliki probabilitas yang lebih rendah daripada (0,448, 22). Pair (0,2592, 26), (0,504, 26), (0,324, 26), dan (0,144, 26) dihilangkan, karena memiliki probabilitas yang

lebih rendah daripada (0,63, 26). Pair (0.2016, 24) dan (0.252, 24) dihilangkan, karena memiliki probabilitas lebih rendah tetapi biaya lebih tinggi daripada pair (0,448, 22).

6. Gunakan metode perhitungan yang sama untuk mendapatkan hasil untuk batasan waktu lainnya, hingga batasan waktu 16.
7. Di bawah batasan waktu 16:

$$\text{TC } 10+6: (0.8, 14) \odot (1.0, 2) = (0.8, 16)$$

$$\text{TC } 10+6: (0.9, 18) \odot (1.0, 2) = (0.9, 20)$$

$$\text{TC } 10+6: (1.0, 22) \odot (1.0, 2) = (1.0, 24)$$

$$\text{TC } 11+5: (0.8, 14) \odot (0.4, 2) = (0.32, 16)$$

$$\text{TC } 11+5: (0.8, 14) \odot (1.0, 8) = (0.8, 22)$$

$$\text{TC } 11+5: (1.0, 18) \odot (0.4, 2) = (0.4, 20)$$

$$\text{TC } 11+5: (1.0, 18) \odot (1.0, 8) = (1.0, 26)$$

$$\text{TC } 12+4: (0.8, 14) \odot (0.4, 2) = (0.32, 16)$$

$$\text{TC } 12+4: (0.8, 14) \odot (1.0, 8) = (0.8, 22)$$

$$\text{TC } 12+4: (1.0, 18) \odot (0.4, 2) = (0.4, 20)$$

$$\text{TC } 12+4: (1.0, 18) \odot (1.0, 8) = (1.0, 26)$$

$$\text{TC } 13+3: (1.0, 14) \odot (0.4, 2) = (0.4, 16)$$

$$\text{TC } 13+3: (1.0, 14) \odot (1.0, 8) = (1.0, 22)$$

Pasangan yang dipilih meliputi (0.9, 20), (0.8, 16), dan (1.0, 22).

(1.0, 24) dan (1.0, 26) dihilangkan, karena memiliki biaya yang lebih tinggi daripada (1.0, 22). (0.8, 22) dan (0.2, 22) dihilangkan, karena mereka memiliki probabilitas lebih rendah dari (1.0, 22). (0,4, 20) dihilangkan, karena mereka memiliki probabilitas yang lebih rendah daripada (0,9, 20). (0,32, 16) dihilangkan, karena mereka memiliki probabilitas yang lebih rendah daripada (0,8, 16).

8. Di bawah batasan waktu 17:

$$\text{TC } 11+6: (0.8, 14) \odot (1.0, 2) = (0.8, 16)$$

$$\text{TC } 11+6: (1.0, 18) \odot (1.0, 2) = (1.0, 20)$$

$$\text{TC } 12+5: (0.8, 14) \odot (0.4, 2) = (0.32, 16)$$

$$\text{TC } 12+5: (0.8, 14) \odot (1.0, 8) = (0.8, 22)$$

$$\text{TC } 12+5: (1.0, 18) \odot (0.4, 2) = (0.4, 20)$$

$$\text{TC } 12+5: (1.0, 18) \odot (1.0, 8) = (1.0, 26)$$

$$\text{TC } 13+4: (1.0, 14) \odot (0.4, 2) = (0.4, 16)$$

$$\text{TC } 13+4: (1.0, 14) \odot (1.0, 8) = (1.0, 22)$$

Pasangan yang dipilih termasuk (1.0, 20) dan (0.8, 16).

Pasangan (1.0, 26) dan (1.0, 22) dihapus, karena memiliki biaya yang lebih tinggi daripada (1.0, 20). (0,4, 20) dihapus, karena memiliki probabilitas lebih rendah daripada (1,0, 20). (0.8, 22) dihilangkan, karena memiliki biaya yang lebih tinggi dari (0.8, 16). (0.2, 22) dihapus, karena memiliki probabilitas lebih rendah dari (0.8, 22), yang telah dihapus. (0,32, 16) dan (0,4, 16) dihilangkan, karena mereka memiliki probabilitas yang lebih rendah daripada (0,8, 16).

9. Di bawah batasan waktu 18:

$$TC_{12+6}: (0.8, 14) \odot (1.0, 2) = (0.8, 16)$$

$$TC_{12+6}: (1.0, 18) \odot (1.0, 2) = (1.0, 20)$$

$$TC_{13+5}: (1.0, 14) \odot (0.4, 2) = (0.4, 16)$$

$$TC_{13+5}: (1.0, 14) \odot (1.0, 8) = (1.0, 22)$$

Pasangan yang dipilih termasuk $(1.0, 20)$ dan $(0.8, 16)$.

$(1.0, 22)$ dihapus karena memiliki biaya lebih tinggi dari $(1.0, 20)$. $(0.4, 16)$ dihapus karena memiliki probabilitas lebih rendah dari $(0.8, 16)$.

10. Di bawah batasan waktu 19:

$$TC_{13+6}: (1.0, 14) \odot (1.0, 2) = (1.0, 16)$$

Hanya ada satu pasangan di bawah batasan waktu ini. Tabel 7.9 merepresentasikan hasil akhir Tabel D untuk jalur $V3 \rightarrow V4$.

Tabel 7.9 Tabel D Menampilkan Jalur $V3 \rightarrow V4$

T	(P, C)	(P, C)	(P, C)	(P, C)	(P, C)	(P, C)
4	(0.1296, 36)					
5	(0.648, 36)	(0.1008, 30)	(0.1296, 32)			
6	(0.1152, 28)	(0.1008, 26)	(0.504, 30)	(0.81, 36)	(0.648, 32)	
7	(0.2016, 24)	(0.576, 28)	(0.63, 30)	(0.504, 26)	(0.81, 32)	(0.0896, 22)
8	(0.81, 32)	(0.2016, 20)	(0.448, 22)	(0.72, 28)	(0.9, 36)	(0.63, 26)
9	(0.1792, 16)	(0.252, 20)	(0.56, 22)	(0.9, 30)	(0.72, 26)	
10	(0.224, 16)	(0.288, 20)	(0.64, 22)	(0.9, 26)		
11	(0.504, 20)	(0.9, 26)	(0.256, 16)	(1.0, 36)	(0.8, 22)	
12	(0.448, 16)	(0.63, 20)	(0.8, 22)	(0.9, 26)	(1.0, 30)	
13	(0.56, 16)	(0.9, 24)	(0.72, 20)	(1.0, 26)	(0.8, 22)	
14	(0.64, 16)	(0.9, 20)				
15	(0.8, 16)	(0.9, 20)	(1.0, 22)			
16	(0.9, 20)	(0.8, 16)	(1.0, 22)			
17	(1.0, 20)	(0.8, 16)				
18	(1.0, 20)	(0.8, 16)				
19	(1.0, 16)					

Berdasarkan tabel, batasan waktu terpendek adalah 4 unit waktu, dan batasan terpanjang adalah 19. Jalur $V1 \rightarrow V2 \rightarrow V3 \rightarrow V4$ dapat ditemukan dengan hasil akhir. Misalnya, pada batasan waktu 9, ada lima jalur, yaitu $(0.1792, 16)$, $(0.252, 20)$, $(0.56, 22)$, $(0.9, 30)$, dan $(0.72, 26)$. Jalur mencapai pasangan $(0.252, 20)$ dapat ditemukan dengan langkah-langkah berikut:

1. $(0.63, 18) \odot (0.4, 2) = (0.252, 20)$ saat $V3 \rightarrow V4$
2. $(0.7, 9) \odot (0.9, 9) = (0.63, 18)$ saat $V2 \rightarrow V3$
3. $(1.0, 5) \odot (0.7, 4) = (0.7, 9)$ saat $V1 \rightarrow V2$

4. Oleh karena itu, jalur $V1 \rightarrow V2 \rightarrow V3 \rightarrow V4$ berakhir pada pasangan (0.252, 20) dan diturunkan dari (1.0, 5) (0.7, 4) \rightarrow (0.7, 9) (0.9, 9) \rightarrow (0.63, 18) (0.4, 2) \rightarrow (0.252, 20).

Singkatnya, PTM HES memiliki keunggulan dalam memecahkan masalah HA. Total biaya sistem dapat dikurangi dengan menjadwalkan probabilitas kepercayaan yang dijamin di bawah berbagai batasan waktu. Bagian berikutnya secara singkat memperkenalkan konsep Masalah Waktu Polinomial Non-deterministik, yang menjelaskan jenis masalah dari bab ini.

7.4 MASALAH WAKTU POLINOMIAL NONDETERMINISTIK

Nondeterministic Polynomial-Time Problems (NP-Problem) adalah sekumpulan masalah keputusan yang tidak dapat diselesaikan dengan algoritma waktu polinomial. Masalah Keputusan adalah menentukan apakah ada rute yang biayanya tidak lebih dari K untuk K tertentu. Salah satu contoh Masalah NP adalah Travelling Salesman Problem (TSP), yaitu mencari cara termurah untuk mengunjungi semua kota dan kembali ke titik awal ketika diberikan kumpulan kota dan biaya perjalanan antara masing-masing pasangan.

Teorema Cook

Masalah Propositional Satisfiability Problem (SAT) adalah salah satu masalah tersulit dalam NP, yang juga disingkat sebagai masalah Satisfiability. Masalahnya biasanya membahas masalah penilaian yang menentukan apakah ada penetapan nilai yang benar untuk satu set kalimat dalam logika proposisional. Logika mengacu pada nilai penugasan ke Bahasa Formal yang merupakan pendekatan simbolisasi menggunakan string simbol untuk mewakili serangkaian batasan, aturan, atau struktur. Teorema Cook menyatakan bahwa semua masalah di NP dapat "direduksi" menjadi masalah SAT dengan menggunakan rumus Boolean F yang diberikan untuk menentukan apakah ada penugasan untuk setiap x_i , sehingga $F = \text{TRUE}$.

Soal A direduksi menjadi Soal B, dan A lebih kecil atau lebih kecil dari B ($A < B$). Pemecahan algoritma B dapat digunakan untuk menyelesaikan A. Jadi Soal B lebih sulit atau tidak sederhana dari Soal A. Contoh 1. Soal A: temukan media barisan. Soal B: mengurutkan barisan, dan A lebih kecil atau lebih kecil dari B ($A < B$). B ada di NP, dan $\text{SAT} < B \rightarrow$. Dari teorema Cook, kita tahu bahwa SAT paling sulit di NP, maka B agak setara dengan SAT. C ada di NP, dan $B < C$, dll. Banyak masalah seperti SAT dan B; mereka disebut masalah NP-lengkap. Perhatikan bahwa "pengurangan" ini harus dilakukan dalam waktu polinomial. Jika ada solusi waktu polinomial untuk hanya satu NP yang lengkap, $\text{NP} = \text{P}$. Bagaimana membuktikan masalah B adalah NP lengkap: Tunjukkan bahwa B ada di NP. Temukan NP menyelesaikan masalah A, dan Anda dapat membuat pengurangan waktu polinomial (transformasi): $A < B$.

7.5 LATIHAN

7.5.1 Pertanyaan Mendasar

1. Apa konsep Sistem Tertanam Heterogen?
2. Apa saja fitur utama Sistem Tertanam Heterogen?
3. Model dasar apa yang dimiliki Sistem Tertanam Heterogen?

4. Apa itu pemrograman dinamis dan bagaimana kontribusi pemrograman dinamis pada Sistem Tertanam Heterogen?
5. Apa saja ciri-ciri Unit Fungsional? Peran apa yang dilakukan? Unit Fungsional bermain di Sistem Tertanam Heterogen?
6. Apa definisi dari Model Waktu Tetap dari Sistem Tertanam Heterogen? Apa arti istilah Waktu Tetap dalam konsep ini?
7. Mengapa kita perlu menyelesaikan Tugas Heterogen untuk mengoptimalkan Model Waktu Tetap?
8. Apa yang dimaksud dengan Grafik Aliran Data? Dalam situasi apa kita perlu menggunakan teknik ini?
9. Apa tujuan dari Penjadwalan Sumber Daya Minimum setelah menyelesaikan Tugas Heterogen dalam Model Waktu Tetap?
10. Apa konsep dari Tugas Heterogen dengan masalah Probabilitas?
11. Apa yang dimaksud dengan Probabilitas Keyakinan Terjamin?
12. Apa yang dimaksud dengan Grafik Aliran Data Probabilistik? Apa perbedaan antara Grafik Aliran Data Probabilistik dan Grafik Aliran Data?
13. Apa yang dimaksud dengan Tabel B dalam Tugas Heterogen dengan masalah Probabilitas?
14. Apa yang dimaksud dengan Tabel D dalam Tugas Heterogen dengan masalah Probabilitas? Apa hubungan antara Tabel B dan Tabel D?
15. Apa yang dimaksud dengan masalah Waktu Polinomial Nondeterministik? Berikan contoh.
16. Bagaimana pendapat Anda tentang Masalah Kepuasan Proposisional?

7.5.2 Pertanyaan Praktis

1. Latihan untuk Model Waktu Tetap

Tabel 7.10 merepresentasikan 5 node A, B, C, D, dan E dalam sebuah linked list. Setiap node memiliki dua mode kerja yang berbeda, yang semuanya memiliki waktu penyelesaian dan biaya sumber daya yang berbeda. Misalnya, node A memiliki mode A1 dan mode A2. Misi Anda meliputi:

- (a) Anda perlu menghitung total biaya minimum sambil memenuhi batasan waktu dengan menggunakan pemrograman dinamis.
- (b) Tunjukkan dan jelaskan setiap langkah.
- (c) Apa penugasan mode untuk semua node? Berikan detail semua jalur.

Tabel 7.10 Node dengan Waktu dan Biaya Penyelesaian

Node	Waktu	Biaya
A1	1	6
A2	2	4
B1	3	8
B2	5	5

C1	4	9
C2	6	7
D1	2	9
D2	5	4
E1	3	7
E2	7	4

2. Pertanyaan Tambahan untuk Model Waktu Tetap

Rancanglah sebuah eksperimen atau skenario praktik yang dapat menggunakan hasil soal Latihan Model Waktu Tetap. Lakukan perbandingan antara algoritme ini dan algoritme yang diajarkan di bab lain, seperti Bab Arsitektur Sistem Tertanam Seluler.

3. Latihan untuk Model Waktu Probabilistik

Ditunjukkan pada Tabel 7.11, ada 4 node 0, 1, 2, dan 3 dalam sebuah linked list. Setiap node memiliki dua mode kerja yang berbeda, R1 dan R2. Tabel menampilkan probabilitas, waktu, dan biaya setiap node. Berikut adalah misi Anda:

- Ubah Tabel 7.11 menjadi tabel yang menunjukkan fungsi distribusi kumulatif dan biaya pada setiap node dengan menggunakan tipe unit fungsional yang berbeda. (Lihat Tabel 7.4.)
- Buatlah Tabel B. (Lihat Bagian 7.3.2.)
- Buatlah Tabel D. (Lihat Bagian 7.3.3 dan Bagian 7.3.4.)
- Hitung total biaya minimum sambil memenuhi batasan waktu dengan semua probabilitas yang dijamin dengan menggunakan program dinamis.
- Tunjukkan dan jelaskan setiap langkah.
- Untuk skenario hard real-time dengan batasan waktu 19, apa penugasan mode untuk semua node untuk mendapatkan biaya minimum?

Tabel 7.11 Node dengan Waktu, Probabilitas, dan Biaya

Node	R1			R2		
	T_1	P_1	C_1	T_2	P_2	C_2
0	1	0.7	8	2	0.7	3
	3	0.3	8	4	0.3	3
1	1	0.8	9	2	0.6	4
	2	0.2	9	3	0.4	4
2	1	0.9	7	3	0.2	5
	4	0.1	7	5	0.8	5
3	1	0.4	10	2	0.6	6
	3	0.6	10	5	0.4	6

7.6 DAFTAR ISTILAH

Pemrograman Dinamis

adalah pendekatan untuk memecahkan masalah yang rumit dengan membaginya menjadi sub-masalah yang lebih kecil.

Sistem Tertanam Heterogen (HESs)

adalah metode penyebaran beberapa unit fungsional dalam sistem tertanam untuk mencapai kinerja tingkat yang lebih tinggi.

Heterogenitas

pada beberapa unit operasi dasar, seperti memori, prosesor, dan unit pemrosesan grafis.

Unit fungsional

adalah komponen tingkat rendah atau sekelompok komponen yang digunakan untuk tujuan tertentu, seperti fungsi aritmatika, penugasan, operasi logika, atau operasi untuk membandingkan.

Model Waktu Tetap (FTM)

adalah model yang terdiri dari sejumlah node yang semuanya memiliki nilai tetap dengan waktu eksekusi tertentu.

Model Waktu Probabilistik

adalah model yang terdiri dari sejumlah node yang dibatasi oleh waktu eksekutif bersyarat yang terkait dengan beberapa input atau penggunaan perangkat keras.

Tabel B

adalah tabulasi yang dapat mewakili biaya dan probabilitas penyelesaian simpul di bawah batasan waktu yang berbeda.

Tabel D

adalah tabulasi yang mewakili solusi optimal dengan probabilitas dan biaya di bawah batasan waktu yang berbeda, yang dihasilkan dengan menggunakan pemrograman dinamis.

Masalah Waktu Polinomial Nondeterministik (Masalah NP)

adalah sekumpulan masalah keputusan yang tidak dapat diselesaikan dengan algoritma polinomial-waktu.

Masalah Keputusan

adalah untuk menentukan apakah ada rute yang biayanya tidak lebih dari K untuk K tertentu.

Teorema Cook

nyatakan bahwa semua masalah dalam NP dapat "direduksi" menjadi masalah SAT dengan menggunakan rumus Boolean F yang diberikan untuk menentukan apakah ada penugasan untuk setiap x_i , sehingga $F=TRUE$.

BAB 8

PENGOPTIMALAN SELULER DENGAN PENJADWALAN LOOP

Optimasi Loops adalah teknik penting dan efektif untuk mencapai kinerja yang lebih baik tanpa mengganti atau meningkatkan perangkat keras. Bab ini memperkenalkan pendekatan lain untuk mengoptimalkan sistem tertanam seluler. Dalam beberapa skenario praktis, kinerja sistem tertanam seluler dapat diukur dengan dua parameter, yaitu waktu dan daya. Salah satu skema optimal adalah menyelesaikan tugas di bawah batasan waktu dengan konsumsi energi tingkat yang lebih rendah. Untuk mencapai tujuan ini, bab ini memberikan siswa metode yang menggunakan teknik Penjadwalan Loop untuk meningkatkan efisiensi kerja dan kemampuan manajemen daya, karena menyelesaikan loop adalah salah satu bagian aplikasi seluler yang paling memakan waktu dan energi. Mengoptimalkan Loop adalah teknik penting dan efektif untuk mencapai kinerja yang lebih baik dengan menggunakan infrastruktur seluler yang ada.

Isi utama bab ini meliputi:

1. Konsep yang terkait dengan optimasi penjadwalan loop
2. Kategori penjadwalan
3. Metode optimasi penjadwalan loop
4. Perbedaan antara penugasan dan rotasi dengan batasan waktu
5. Menggunakan teknik Probability Data Flow Graph (PDFG) untuk optimasi penjadwalan loop

8.1 PENDAHULUAN

Bab ini bertujuan untuk menginstruksikan siswa untuk mendapatkan pengetahuan tentang sistem tertanam mobile menggunakan penjadwalan loop dan teknik komputasi paralel. Siswa harus memahami prinsip operasi dari optimasi waktu dan memahami bagaimana menerapkan skema yang diperkenalkan dalam bab ini. Konsep penting harus dipahami sepenuhnya dari membaca bab ini, seperti Probabilistic Data-Flow Graph (PDFG), Retiming, Rotation Scheduling, dan Parallel Computing. Setelah membaca bab ini, siswa perlu memahami skenario memanfaatkan dan menerapkan teknik yang disebutkan. Sepanjang bab ini, siswa harus dapat menjawab pertanyaan-pertanyaan berikut:

1. Apa perbedaan antara Grafik Aliran Data Probabilistik dan Grafik Aliran Data? Mana yang lebih efektif dalam konteks dunia nyata?
2. Kapan dan mengapa Penjadwalan Loop dan Komputasi Paralel dapat mencapai optimalisasi waktu dan daya dalam sistem tertanam seluler?
3. Apa yang dimaksud dengan Retiming, Unfolding, dan Rotation? Bagaimana Anda menggunakan kedua teknik ini dalam optimalisasi waktu dan tenaga?
4. Apa langkah-langkah kunci dalam mengimplementasikan Optimasi Penjadwalan Loop?

8.2 TEKNIK DAN MODEL GRAFIK DASAR

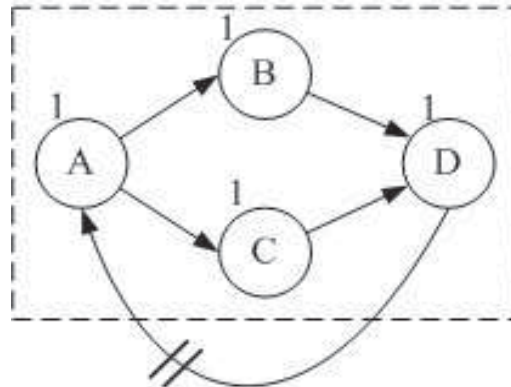
8.2.1 Grafik Aliran Data dalam Penjadwalan Loop

Dalam bab-bab sebelumnya, kami telah memperkenalkan pendekatan Data-Flow Graph (DFG) yang menggunakan diagram grafik untuk mewakili dependensi data atau unit fungsional antar operasi. Dalam prakteknya, terdapat berbagai macam metode untuk memodelkan aplikasi dengan sequencing graph. Menggunakan metodologi berbasis grafis dapat mempertimbangkan setiap aplikasi urutan kegiatan, fungsi, dan jadwal tugas. Untuk tujuan optimasi, berbagai fase dalam aplikasi dapat disintesis sebagai rantai subproses dengan menggunakan metode teori-grafik. Bab ini terutama menggunakan teknik yang diturunkan dari metode DFG untuk mewakili penjadwalan loop. Definisi loop DFG diberikan oleh Definisi 2. Notasi utama yang digunakan dalam bab ini diberikan pada Tabel 8.1.

Tabel 8.1 Notasi Utama yang Digunakan dalam Bab Ini

G	Grafik Aliran Data (DFG)
G_r	DFG mundur
G_f	Grafik DFG yang tidak dilipat
V	Satu set node $-A_1, A_2, \dots, A_n$, A_n mengacu pada node
E	Satu set tepi
d	Sebuah fungsi yang mewakili penundaan yang melekat pada tepi
d_r	Penundaan waktu ulang
t	Waktu komputasi
$t(v)$	Unit waktu komputasi dari setiap node
$r(v)$	Unit waktu tunda setiap node
$D(p)$	Total waktu komputasi jalur p
$T(p)$	Jumlah tundaan total dari jalur p
$B(G)$	Batas iterasi dari graf aliran data G
$T(I)$	Mengacu pada penjumlahan semua waktu komputasi dalam satu siklus.
$D(I)$	Mengacu pada penjumlahan semua waktu tunda dalam satu siklus.
I	Mengacu pada siklus yang terdiri dari perhitungan dan penundaan.
P	Periode iterasi
r	Fungsi pengaturan waktu
f	Faktor terbuka
S	Jumlah titik dalam ruang desain
c	Sebuah periode siklus

Definisi 2 $G = \langle V, E, d, t \rangle$ merepresentasikan graf berarah dengan ciri pembobotan simpul dan pembobotan sisi; dimana $V = -A_1, A_2, \dots, A_n$ mewakili satu set node; $E_i = A_j \rightarrow A_k$, mengacu pada edge E_i yang dimulai dari A_j dan menuju A_k ; $t(v)$ mengacu pada waktu komputasi dari simpul v .



Gambar 8.1 Contoh Data-Flow Graph (DFG).

Gambar 8.1 merupakan contoh DFG di mana $V = \{A, B, C, D\}$. Simbol “ \rightarrow ” digunakan untuk menggambarkan sebuah edge dari satu node ke node lainnya. Dari contoh, $A \rightarrow C$ mengacu pada tepi dari A ke C. Pada Gambar 8.1, ada lima tepi yang ditunjukkan pada garis panah, termasuk $A \rightarrow C$, $A \rightarrow B$, $B \rightarrow D$, $C \rightarrow D$, dan $D \rightarrow A$. Tepi “ $D \rightarrow A$ ” ditandai dengan dua garis pendek, yang berarti bahwa tepi memiliki dua penundaan. Tepi lainnya tidak memiliki penundaan, yang ditunjukkan oleh garis panah normal tanpa menandai garis pendek. Waktu komputasi setiap node, $t(v)$, biasanya dinyatakan sebagai angka nilai di samping node. Berdasarkan Gambar 8.1, waktu komputasi tiap node adalah satu satuan waktu. Berdasarkan pemahaman skema DFG, kami memperkenalkan konsep iterasi dalam penjadwalan loop.

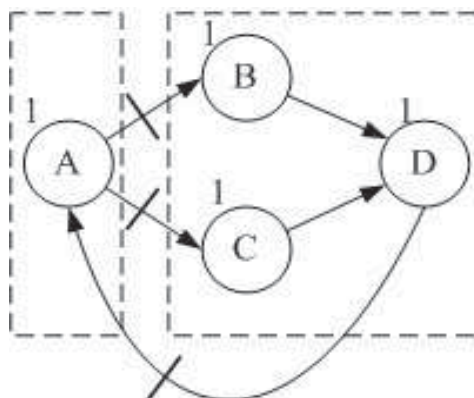
Iterasi mewakili pengulangan proses yang setiap node di V dieksekusi tepat satu kali. Iterasi dapat dikaitkan dengan jadwal statis. Jadwal statis berarti node harus mengikuti urutan prioritas yang ditentukan oleh DFG yang diberikan. Tepi berbobot digunakan untuk mewakili dependensi antar iterasi. Misalnya, seperti yang ditunjukkan pada Gambar. 8.1, sebuah iterasi j memiliki dua simpul D dan A yang dihubungkan oleh tepi e . Keterlambatan dari D ke A adalah $d(e)$. Artinya, komputasi simpul v pada iterasi j berhubungan dengan eksekusi simpul n pada iterasi $j + d(e)$. Tepi lain dalam gambar tidak memiliki penundaan, yang berarti dependensi data yang melekat pada tepi tersebut berada dalam iterasi yang sama.

Selain itu, periode siklus mengacu pada total waktu komputasi dari jalur tunda nol terpanjang dalam DFG. Kami menggunakan $c(G)$ untuk mewakili periode siklus, menggunakan $D(p)$ untuk mewakili total waktu komputasi, dan menggunakan $T(p)$ untuk mewakili jumlah penundaan total dari jalur, “ p .” Sebagai contoh, $c(G)$ dari DFG pada Gambar 8.1 adalah 3 satuan waktu, jika kita mengasumsikan bahwa waktu komputasi adalah satu satuan waktu pada setiap node. Ekspresi matematis dari $c(G)$ adalah: $c(G) = \max_{p \in G} \{D(p) + T(p)\}$. Tujuan pengenalan iterasi dan periode siklus adalah untuk merancang algoritma penjadwalan loop yang dioptimalkan. Optimasi dapat dicapai dengan mengoptimalkan proses iterasi dan meminimalkan biaya waktu periode siklus. Dalam prakteknya, DFG yang diberikan terdiri dari sejumlah siklus yang memiliki konsumsi waktu yang berbeda. Prinsip optimasi adalah meminimalkan waktu eksekusi semua tugas dalam satu iterasi. Dua

pendekatan penting untuk meminimalkan total waktu eksekusi termasuk retiming dan berlangsung.

8.2.2 Pengaturan Waktu dan Pembukaan

Retiming adalah pendekatan optimasi yang mengubah DFG menjadi waktu polinomial dan meminimalkan periode siklus dengan mendistribusikan kembali register atau penundaan dalam rangkaian. Metode utama dari retiming adalah untuk mengatur ulang atau merestrukturisasi posisi logis dari register atau kait di sirkuit. Menggunakan pendekatan berbasis grafik dapat secara efektif mendistribusikan penundaan antara tepi. Misalnya, Gambar 8.2 menunjukkan grafik retime yang diturunkan dari Gambar 8.1. Seperti ditunjukkan pada Gambar 8.2, sebuah retiming dapat dianggap sebagai fungsi retiming r , yang mendistribusikan kembali node dari DFG sebelumnya dari Gambar. 8.1. DFG baru direpresentasikan sebagai $G_r = \langle V, E, dr, t \rangle$, dengan penundaan retime “ dr .” Dalam hal ini, waktu tunda simpul A adalah 1 satuan waktu. Node B, C, dan D tidak memiliki waktu tunda. Retiming menghasilkan periode baru dari siklus $c(G_r)$ yang berubah menjadi 2 dari 3. Mengingat dependensi, fungsi retiming bervariasi, yang tergantung pada situasi independen.



Gambar 8.2 Contoh Retimed Data-Flow Graph (DFG)

Kita dapat mengabstraksi ekspresi matematis dari deskripsi retiming yang disebutkan di atas. Gambar 8.3 mengilustrasikan situasi umum dari formulasi retiming. Gambar tersebut menyatakan bahwa ada $r(v)$, unit waktu tunda yang mengikuti tepi dari v ke w , $v \rightarrow w$, yang darinya dikurangi tepi dari u ke v , $u \rightarrow v$. Semua simpul u , v , dan w berada di dalam DFG G . Berdasarkan representasi pada Gambar 8.3, pewaktuan ulang dapat dirumuskan dengan $dr(e) = d(e) + r(u) - r(v)$, yang juga diberikan dalam Persamaan (8.1). Rumusan tersebut menjelaskan satuan waktu tunda untuk setiap sisi $u \rightarrow v$.

$$dr(e) = d(e) + r(u) - r(v) \tag{8.1}$$



Gambar 8.3 Formulasi Retiming.

Unfolding adalah teknik lain yang dapat mengoptimalkan penjadwalan loop dengan meningkatkan periode siklus rata-rata dari jadwal statis. Prinsip dasar dari pembukaan adalah untuk membuka gulungan dan meratakan DFG asli beberapa kali untuk mendapatkan sejumlah salinan dari set node asli dan menggunakan Paralelisme Tingkat Instruksi (ILP) untuk mengoptimalkan Periode Iterasi. ILP adalah paradigma yang diterapkan untuk mengkaji kemampuan operasi simultan dalam program komputasi atau aplikasi. Periode iterasi mengacu pada waktu komputasi rata-rata dari sebuah iterasi untuk DFG.

Pendekatan ini juga dapat diterjemahkan ke dalam ekspresi matematika. Siswa dapat merujuk pada deskripsi berikut untuk lebih memahami konsep berlangsung. Seperti yang telah kita diskusikan, langkah pertama dari pembukaan adalah membuka lipatan graf asli, DFG G , f kali. Grafik yang tidak dilipat direpresentasikan sebagai Gf , dan f ditandai sebagai faktor yang tidak terbuka. Setiap Gf memiliki f salinan dari himpunan simpul asli, V . Salinan f yang digandakan membuat jadwal berisi f iterasi dari DFG asli. Operasi ini menyediakan proposisi untuk menggunakan pendekatan ILP untuk meningkatkan periode iterasi antar iterasi. Periode iterasi P dapat dirumuskan dengan $P=c(Gf)/f$. Berdasarkan pendekatan yang berlangsung, konsep Iteration Bound diperkenalkan untuk tujuan perhitungan penjadwalan. Sebuah Iteration Bound berarti rasio maksimum waktu komputasi terhadap waktu tunda untuk semua siklus dalam DFG. Prasyaratnya adalah harus ada setidaknya satu loop di DFG. Rasio tersebut dapat diperoleh dengan menghitung proporsi waktu komputasi total dengan total waktu tunda.

Sebagai contoh, kami memberikan contoh dengan menggunakan DFG retiming yang ditunjukkan pada Gambar 8.2. Batas iterasi DFG ini adalah 11 dari $3/2$. Dalam perhitungan, 3 diperoleh dari total waktu komputasi yang merupakan penjumlahan dari B, C, dan D. 2 diperoleh dari 2 penundaan yang melekat pada B dan C. Untuk mencapai optimasi penjadwalan loop, faktor pembukaan minimum dengan fungsi retiming perlu ditemukan. Oleh karena itu, memahami konsep ini sangat penting bagi siswa untuk mengeksplorasi lebih lanjut optimasi waktu dengan menggunakan retiming dan berlangsung. Definisi dari iterasi terikat diberikan oleh Definisi 3.

Definisi 3 Batas iterasi DFG adalah $B(G)$ yang merupakan rasio maksimum waktu komputasi terhadap waktu tunda untuk semua siklus dalam DFG. Persamaan $B(G) = \max_l \frac{T(l)}{D(l)}$ mengacu pada siklus yang terdiri dari komputasi dan penundaan. $T(l)$ mengacu pada penjumlahan semua waktu komputasi dalam satu siklus. $D(l)$ mengacu pada penjumlahan semua waktu tunda dalam satu siklus.

8.3 OPTIMASI WAKTU DASAR

Mengingat kondisi pembatasan sumber daya, ada dua jenis masalah optimasi penjadwalan. Jenis pertama dari masalah penjadwalan adalah meningkatkan penjadwalan loop dengan kendala sumber daya; jenis lainnya adalah optimasi tanpa kendala sumber daya.

Penentuan jenis optimasi tergantung pada apakah ada operasi paralel yang dijalankan oleh perangkat keras bergaya paralel. Misalnya, beberapa prosesor yang dirancang oleh arsitektur Very Long Instruction Word (VLIW) dapat secara bersamaan mendukung hingga delapan operasi, yang menyiratkan bahwa kendala sumber daya perlu dipertimbangkan. Bab ini memperkenalkan pendekatan Rotation Scheduling Algorithm (RSA), yang merupakan algoritma penjadwalan polinomial-waktu yang mempertimbangkan kendala sumber daya. Algoritma ini menggunakan pendekatan retiming pada DFGs dan menghasilkan jadwal loop yang optimal dalam waktu polinomial dengan iterasi. Rotation Scheduling adalah skema penjadwalan yang digunakan untuk mengoptimalkan kinerja jadwal loop dengan beberapa kendala sumber daya. Menggunakan teknik penjadwalan rotasi dapat menawarkan pendekatan untuk memadatkan iterasi loop dan menghasilkan panjang jadwal minimum dalam waktu polinomial. Metode pelaksanaan penjadwalan rotasi diberikan dalam Bagian 8.4.

Pendekatan yang diperkenalkan pada bagian ini adalah algoritma optimal tingkat lanjut yang memiliki kinerja lebih baik daripada kebanyakan algoritma lainnya. Misalnya, penelitian sebelumnya telah membuktikan bahwa penjadwalan instruksi yang paralel dengan perpipaan perangkat keras dapat mengoptimalkan loop secara efisien. Dalam ilmu komputer, jenis penjadwalan instruksi ini juga dikenal sebagai Modulo Scheduling atau Software Pipelining. Namun, telah ditentukan bahwa algoritma penjadwalan rotasi lebih unggul daripada penjadwalan modulo untuk aplikasi pemrosesan sinyal digital dengan mengoptimalkan panjang jadwal.

Gambar 8.4 memberikan contoh penugasan untuk sistem dual-prosesor. Kolom kiri mewakili waktu eksekusi, daftar dari 1 hingga 13 unit waktu. Lima tugas dilampirkan ke prosesor yang sesuai dengan konsumsi waktu. Misalnya, tugas 1 membutuhkan 3 unit waktu, dan tugas 2 dan 3 adalah tugas yang berhasil. Menurut Gambar. 8.4, waktu pencapaian adalah 13 unit waktu. Selain itu, menggunakan rotasi dapat mengurangi waktu penyelesaian siklus. Gambar 8.5 merupakan contoh rotasi dengan menggunakan sistem yang sama seperti yang ditunjukkan pada Gambar 8.4. Panjang siklus diturunkan menjadi 10 unit waktu dan prosesor 2 untuk menjalankan lebih banyak tugas, termasuk tugas 1 dan 2. Perbandingan antara Gambar 8.4 dan Gambar 8.5 menjelaskan keuntungan yang jelas dari penggunaan rotasi untuk meningkatkan efisiensi kerja untuk DFG yang diberikan.

Meminimalkan Faktor Pembuka Salah satu bagian yang menantang dari algoritma penjadwalan rotasi adalah sulitnya menemukan solusi yang layak dalam ruang desain yang besar dalam waktu singkat. Mengatasi target optimasi, Ruang Desain diasosiasikan dengan retiming, faktor pembukaan, dan jumlah poin. Jumlah titik diwakili oleh S . Kami mendefinisikan nilai Ruang Desain = $r \times f \times S$. Tujuan dari optimasi adalah menghasilkan set minimum faktor yang tidak terbuka untuk menurunkan batasan periode iterasi dengan retiming.

	Processor 1	Processor 2
1	1	
2		
3		
4	3	2
5		
6	4	
7		
8		
9		
10	5	
11		
12		
13		

Gambar 8.4 Contoh template penugasan.

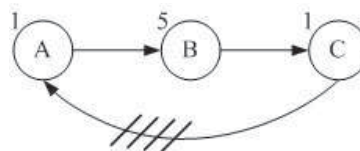
Metode menghasilkan periode siklus minimum mengikuti beberapa langkah:

1. Konfirmasi bahwa ada jadwal statis hukum dengan retiming untuk DFG.
2. Cari tahu nilai variabel dalam DFG yang diberikan, termasuk himpunan node "V", himpunan edge "E", delay "d", dan waktu komputasi "t". Definisikan "c" sebagai periode siklus.
3. Periode siklus minimum dapat diperoleh saat mencocokkan dua kondisi. Pertama, nilai yang diperoleh dengan membagi periode siklus dengan faktor bukaan tidak lebih kecil dari batas iterasi DFG. Kedua, periode siklus tidak kurang dari waktu komputasi terbesar pada node.
4. Plafon produk dari faktor bukaan dan batas iterasi DFG.
5. Bandingkan waktu komputasi terbesar dari semua node dengan hasil yang dihasilkan pada langkah (4). Nilai yang lebih besar adalah periode siklus minimum.

Gambar 8.6 merupakan contoh menghasilkan periode siklus minimum. Berdasarkan gambar tersebut, terdapat tiga node dalam DFG, yaitu A, B, dan C. Waktu komputasi setiap node bervariasi, dengan $t(A)=1$, $t(B)=5$, dan $t(C)=1$, yang juga diberikan dalam gambar. Tepi $A \rightarrow B$ dan $B \rightarrow C$ tidak memiliki waktu tunda. Tepi $C \rightarrow A$ memiliki empat unit waktu tunda, yang ditandai dengan empat garis pendek. Periode siklus dari DFG yang diberikan adalah 7 satuan waktu yang diturunkan dari $1 + 5 + 1$.

	Processor 1	Processor 2
1	3	2
2		
3	4	1
4		
5		
6		
7	5	
8		
9		
10		
11		
12		
13		

Gambar 8.5 Contoh template rotasi.



Gambar 8.6 Contoh periode siklus minimum menggunakan Teorema 8.1.

Menurut Definisi 3, batas iterasi adalah $7/4$, karena total satuan waktu komputasi adalah 7 dan total satuan waktu tunda adalah 4. fakta yang terungkap sama dengan 2, periode siklus terbaik yang layak dengan retiming harus 5, yang berasal dari pilihan nilai yang lebih besar dari 5 dan 4. Selain itu, 5 adalah waktu komputasi terpanjang dari node. 4 adalah langit-langit dari $2 \times (7/4)$, yang merupakan produk dari faktor yang tidak terbuka dan batas iterasi dari grafik aliran data. Periode siklus ini dianggap sebagai periode siklus minimum untuk DFG yang diberikan. Hasil ini juga menunjukkan bahwa tidak ada jadwal dengan periode siklus kurang dari 5 ketika faktor penyingkapan adalah 2.

Langkah-langkah di atas dapat diterjemahkan ke dalam ekspresi matematika, yang diberikan dalam Teorema 8.1.

Teorema 8.1 (Periode Siklus Minimum $c_{\min}(G_f)$) Ada skedul statis legal dengan pewaktuan ulang, misalkan $G = \langle V, E, d, t \rangle$, $f \in \mathbb{Z}^+$, $c \in \mathbb{R}$, jika $c/f \geq B(G)$ dan $c \geq \max_v t(v)$, $\forall v \in V$, lalu $c_{\min}(G_f) = \max(\max_v t(v), \lceil f \cdot B(G) \rceil)$

8.4 OPTIMASI WAKTU DAN DAYA DENGAN PENJADWALAN LOOP

Memiliki pengetahuan tentang waktu ulang dan pembukaan adalah dasar untuk mempelajari metode optimasi tingkat lanjut dalam penjadwalan loop. Bagian ini berfokus pada pengenalan algoritme optimal untuk optimasi waktu dan daya dengan penjadwalan loop.

8.4.1 Grafik Aliran Data Probabilistik

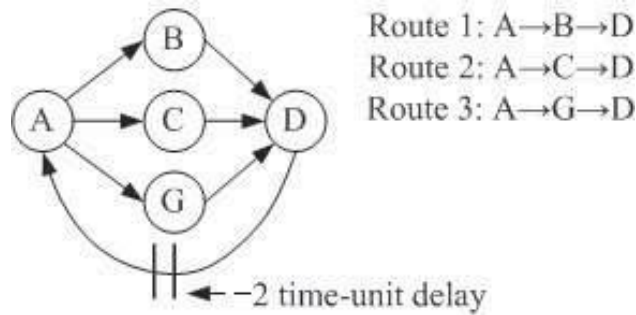
Teknik DFG adalah pendekatan tradisional untuk mensintesis sistem tertanam seluler. Namun, dalam implementasi dunia nyata, perilaku dan kompleksitas sistem seringkali memerlukan DFG tingkat yang lebih tinggi. DFG normal memiliki kesulitan dalam menangani ketidakpastian sistem. Sebagian besar solusi saat ini adalah pendekatan Berbasis Kasus Asumsi, yang didasarkan pada asumsi waktu komputasi pada kasus terburuk atau kasus tingkat rata-rata. Namun demikian, pendekatan ini tidak dapat digunakan untuk banyak situasi praktis karena waktu komputasi dan komunikasi dapat menjadi variabel acak. Rentang kesalahan yang tidak diperbolehkan yang disebabkan oleh variabel acak dapat secara dramatis menurunkan akurasi dan akhirnya menghasilkan jadwal yang tidak efisien.

Probabilistic Data-Flow Graph (PDFG) diperkenalkan untuk mengoptimalkan sistem yang memiliki tugas dengan waktu komputasi yang tidak pasti, seperti sistem antarmuka, sistem fuzzy, atau sistem kecerdasan buatan. Tujuan utama dari PDFG adalah untuk membuat perilaku sistem yang kompleks dapat dimengerti. Untuk mencapai tujuan ini, grafik yang dioptimalkan dapat diperoleh dengan menggunakan pendekatan probabilistik. Bagian penting dari pendekatan ini adalah menghasilkan periode siklus yang mendekati status optimal dengan tingkat kepercayaan yang tinggi.

Menurut penelitian sebelumnya, menggunakan solusi berbasis PDFG lebih unggul daripada pendekatan berbasis kasus asumsi. Peningkatan panjang jadwal menggunakan retiming probabilistik baru dapat mencapai hingga 40% lebih pendek daripada berbasis kasus yang lebih buruk dan hingga 30% lebih pendek dari pendekatan berbasis kasus rata-rata dalam keadaan yang sama. Efisiensi penerapan PDFG memiliki keuntungan besar. Untuk merancang algoritma optimasi, model probabilistik perlu dibuat untuk menjaga hasil dalam rentang kesalahan yang diijinkan. Model probabilistik yang efektif harus berisi semua kemungkinan nilai dari variabel kejadian.

Gambar 8.7 memberikan contoh yang mewakili perilaku sistem yang kompleks. Informasi perilaku yang sesuai ditampilkan oleh Tabel 8.2. Ada lima node dalam grafik, dan setiap node memiliki atribut yang berbeda ketika memilih rute yang berbeda. Tersedia tiga rute, termasuk $A \rightarrow B \rightarrow D$, $A \rightarrow C \rightarrow D$, dan $A \rightarrow G \rightarrow D$, yang direpresentasikan sebagai R1, R2, dan R3 pada Tabel 8.2. Variabel atribut yang dihitung dalam grafik ini meliputi waktu, probabilitas, dan konsumsi energi, direpresentasikan sebagai T , P , dan E pada Tabel 8.2.

Sebagai contoh, node A bekerja secara berbeda ketika mengimplementasikan rute yang berbeda. Saat memilih rute 1, R1, node A memiliki dua mode kerja. Modus pertama membutuhkan waktu komputasi 1 unit waktu pada probabilitas 70% tingkat pencapaian dengan 10 unit konsumsi energi. Mode lainnya membutuhkan waktu komputasi 3 unit waktu dengan probabilitas 100% tingkat pencapaian dengan konsumsi energi yang sama dengan mode kerja pertama. Probabilitas 100% berasal dari jumlah 30% dan 70%, $(0,3+0,7)$.



Gambar 8.7 Contoh DFG probabilistik sesuai dengan Tabel 8.2.

Tabel 8.2 Tabel untuk PDFG dengan Waktu, Probabilitas, dan Konsumsi Energi dari Nodenya Di Bawah Rute yang Berbeda

Node	R ₁			R ₂			R ₃		
	T	P	E	T	P	E	T	P	E
A	1	0.7	10	2	0.8	5	4	0.8	1
	3	0.3		5	0.2		6	0.2	
B	2	1.0	7	3	1.0	5	5	1.0	1
C	1	1.0	9	2	1.0	7	3	1.0	3
G	1	0.8	7	4	0.8	3	5	0.6	2
	2	0.2		5	0.2		6	0.4	
D	1	0.8	9	3	0.9	4	5	0.5	1
	3	0.2		4	0.1		6	0.5	

Selain itu, node A memiliki kinerja yang berbeda saat melewati rute 2, mengacu pada R2. Di R2, ada juga dua mode kerja. Mode kerja pertama dapat menyelesaikan tugas dalam 2 unit waktu dengan probabilitas 80% dan membutuhkan 5 unit biaya energi. Modus kerja lainnya dapat menyelesaikan tugas dengan 5 unit waktu pada probabilitas 100% dengan 5 unit biaya energi. Akhirnya, rute terakhir R3 menawarkan pilihan lain untuk node A. Mode kerja pertama di R3 membutuhkan 4 unit waktu untuk menyelesaikan tugas. Probabilitasnya adalah 80% dan biaya energi adalah 1. Modus kerja kedua di R3 membutuhkan 6 unit waktu untuk menyelesaikan tugas, dengan probabilitas 100% dan 1 unit biaya energi.

Node-node lainnya yang direpresentasikan dalam Tabel 8.2 dapat dioperasikan sama seperti prosedur yang dijelaskan di atas. Untuk mengoptimalkan penjadwalan loop, solusi optimal yang menggabungkan manfaat dari setiap rute perlu ditemukan, meskipun itu adalah tugas yang menantang dalam praktiknya. Bagian berikut dapat memandu siswa untuk merancang solusi menggunakan keterampilan optimasi penjadwalan loop.

8.4.2 Penjadwalan Loop dan Komputasi Paralel

Pendekatan berbasis grafik biasanya diperoleh dari program loop dalam praktek. Optimalisasi penjadwalan loop biasanya dikombinasikan dengan penerapan Komputasi Paralel. Komputasi paralel adalah paradigma untuk komputasi yang secara

bersamaan mengeksekusi beberapa perhitungan. Prinsip operasi komputasi paralel adalah membagi tugas berukuran lebih besar menjadi subtugas berukuran lebih kecil dan menjalankan tugas yang dibagi secara bersamaan. Implementasi komputasi paralel memungkinkan sebuah loop untuk menyelesaikan beberapa proses dalam satu siklus. Memanfaatkan teknik retiming dalam penjadwalan loop menghasilkan solusi optimal, yang menggunakan keunggulan komputasi paralel untuk memaksimalkan kinerja infrastruktur yang ada.

Parallel Computing System (PCS) adalah komputer atau sistem yang menerapkan beberapa prosesor untuk menyelesaikan beberapa proses paralel. Teknik warisan sistem multiprosesor hanya memungkinkan setiap prosesor untuk melampirkan paket prosesornya sendiri. Teknologi multiprocessing saat ini memungkinkan beberapa prosesor logis untuk bekerja dalam satu paket, yang dijalankan oleh prosesor multicore. Teknologi yang muncul telah memperkenalkan berbagai jenis PCS. Salah satu metode populer untuk mengkategorikan PCS adalah mengklasifikasikan PCS berdasarkan prosesor dan memori. Contoh PCS modern adalah multiprosesor Raw yang dikembangkan oleh MIT, multiprosesor Trips yang dikembangkan oleh University of Texas, dan Power5 yang dikembangkan oleh IBM.

Berikut ini adalah contoh program loop bernama W. Gambar 8.8 mewakili DFG yang sesuai untuk program W. Asumsikan bahwa ada empat prosesor yang tersedia untuk menjalankan program W, yang melibatkan prosesor P1, P2, P3, dan P4. Bagian ini memberikan solusi optimal yang dapat memperoleh total waktu eksekusi minimum dengan menggunakan retiming. Langkah pertama adalah menggambar DFG menurut program yang diberikan W. Ekspresi program W mengacu pada berikut ini.

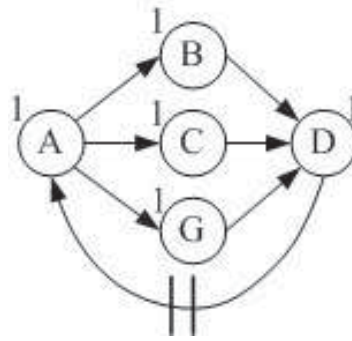
Program yang diberikan W:

```

For (i ∈ N)
  {A[i]=D[i-2]+12;
  B[i]=A[i]×4;
  C[i]=A[i]-10;
  G[i]=A[i]+2;
  D[i]=B[i]+C[i]+G[i];}

```

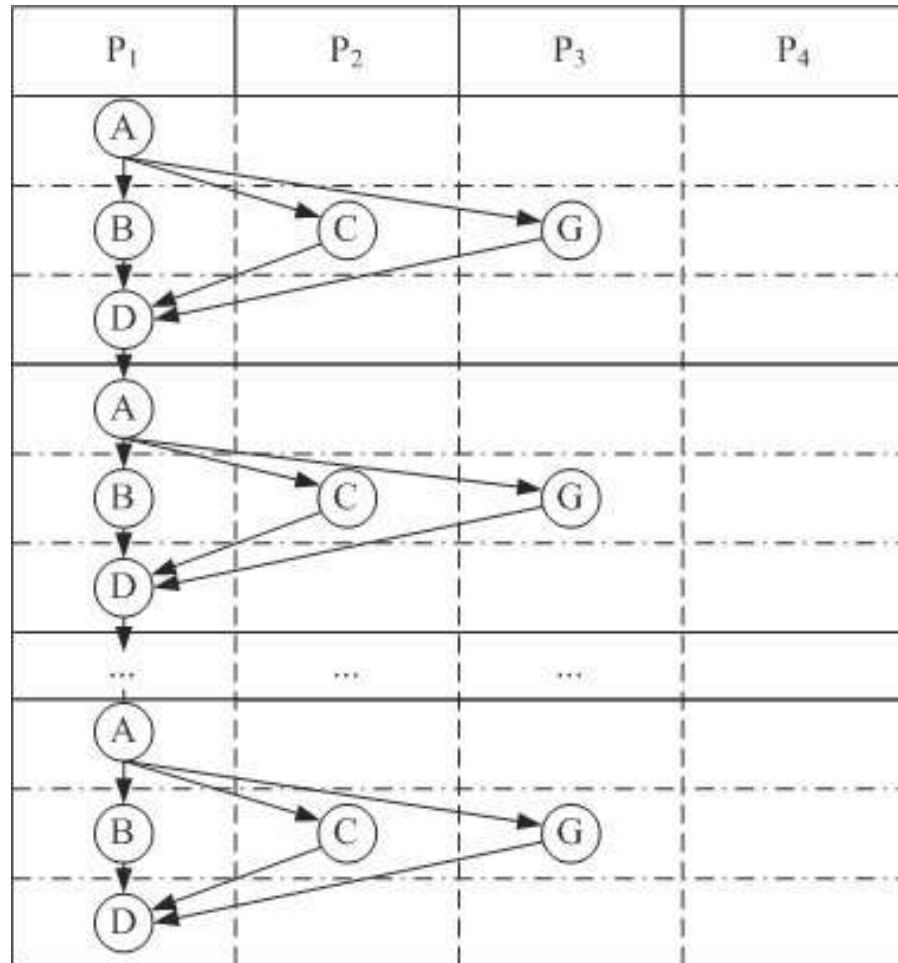
Menurut program yang diberikan W, tepi $D \rightarrow A$ memiliki dua unit waktu tunda karena " $A[i] = D[i-2]+12$ ". Dalam ekspresi ini, " $i-2$ ", berarti penundaan dua unit waktu dari A ke D. Tepi lainnya tidak memiliki penundaan unit waktu dalam kasus ini. Ekspresi juga menggambarkan hubungan antara node dan arah tepi. Misalnya, " $B[i]=A[i]×4$ " berarti tepi $A \rightarrow B$. Ekspresi " $D[i]=B[i]+C[i]+G[i]$ " mengacu pada tiga sisi yang menunjuk pada simpul D, termasuk $B \rightarrow D$, $C \rightarrow D$, dan $G \rightarrow D$. Menyisir semua ekspresi dapat memperoleh loop yang dapat ditunjukkan dengan jelas oleh DFG.



Gambar 8.8 Grafik aliran data probabilistik untuk program W.

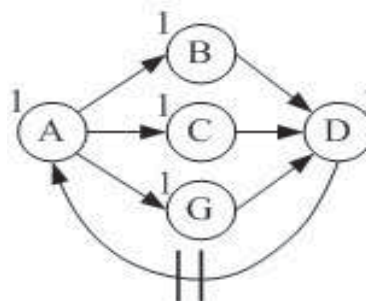
Seperti ditunjukkan pada Gambar 8.8, sebuah DFG digambar berdasarkan program W, di mana $V = \{A, B, C, G, D\}$. $u \rightarrow v$ merupakan edge dari node u ke node v . Pada gambar, edge set terdiri dari tujuh edge, yaitu $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow G$, $B \rightarrow D$, $C \rightarrow D$, $G \rightarrow D$, dan $D \rightarrow A$. Seperti yang telah kita diskusikan, tepi $D \rightarrow A$ memiliki dua unit waktu tunda, yang diwakili oleh garis panah yang ditandai oleh dua garis pendek. Satu garis pendek mengacu pada penundaan satu unit waktu. Tepi lainnya tidak memiliki waktu tunda, yang diwakili oleh garis panah tanpa garis pendek. Waktu komputasi pada setiap node adalah 1, juga ditandai di samping setiap node. Mengikuti DFG ini mengarah ke konsumsi waktu iterasi saat ini, yang perlu melalui semua node di set V tepat satu kali.

Gambar 8.9 merepresentasikan iterasi dari skedul statis yang diturunkan dari Gambar 8.8. Dalam hal ini, tiga prosesor sedang digunakan. Garis panah menggambarkan urutan node, dan panah menunjuk pada node berikutnya. Misalnya, garis panah antara node A dan B berarti node B dapat dieksekusi setelah node A. Menggunakan jadwal statis ini membutuhkan tiga siklus eksekusi untuk menyelesaikan setiap loop. Oleh karena itu, jika program dijalankan 100 kali, jumlah siklus eksekusi total akan menjadi 300, dari 3×100 .

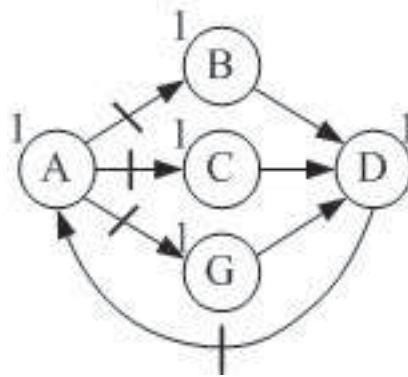


Gambar 8.9 Iterasi jadwal statis untuk grafik aliran data yang diberikan.

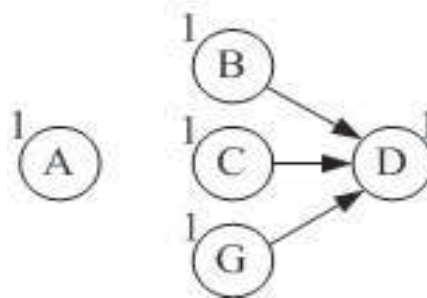
Untuk mengoptimalkan penjadwalan loop, loop yang lebih pendek dapat dicapai dengan mengurangi jumlah siklus eksekusi. Untuk mencapai tujuan ini, diperlukan retiming dan rotasi. Prosedur optimasi untuk PDFG menggunakan retiming dan rotasi ditunjukkan pada Gambar 8.10 –8.13. Langkah-langkah utama masing-masing diwakili dalam Gambar 8.10, 8.11, 8.12, dan 8.13.



Gambar 8.10 Grafik aliran data probabilistik untuk program W .



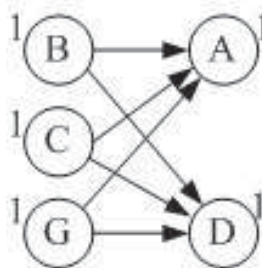
Gambar 8.11 Grafik aliran data probabilistik setelah retiming.



Gambar 8.12 Grafik aliran data probabilistik penjadwalan statis sebelum pengelompokan ulang.

1. PDFG Asli

Gambar 8.10 mengilustrasikan PDFG asli untuk program yang diberikan W, yang sama seperti Gambar 8.8. Pada tahap ini, PDFG mengikuti jadwal statis dan belum ada optimasi. Tepi $D \rightarrow A$ memiliki dua unit waktu tunda.



Gambar 8.13 Prosedur optimasi grafik aliran data probabilistik dengan retiming.

siklus eksekusi diperlukan untuk menyelesaikan satu loop sehingga panjang jadwal loop adalah 3. PDFG mengikuti program yang diberikan W yang ditunjukkan sebagai berikut.

Untuk ($i \in \mathbb{N}$)
 $\{A[i] = D[i-2] + 12;$
 $B[i] = A[i] \times 4;$
 $C[i] = A[i] - 10;$
 $G[i] = A[i] + 2;$

$$D[i]=B[i]+C[i]+G[i];\}$$

2. Retime

Selanjutnya, Gambar 8.11 mewakili PDFG yang menyatakan status setelah retime. Penundaan di sirkuit didistribusikan kembali untuk tujuan meminimalkan periode siklus. Kami mendistribusikan ulang penundaan dari tepi $D \rightarrow A$ ke tepi mulai dari A, termasuk tepi $A \rightarrow B$, $A \rightarrow C$, dan $A \rightarrow G$, karena mempertimbangkan dependensi node. Penundaan yang didistribusikan kembali pada ketiga tepi ini adalah sama, penundaan 1 unit waktu, yang ditunjukkan pada Gambar 8.11. Pada tahap ini, masih ada tiga siklus eksekusi dalam loop. Mengacu pada Persamaan (8.1) di Bagian 8.2.1, penundaan pada setiap node setelah retiming juga dapat diperoleh dengan perhitungan berikut.

$$\text{Menghitung } dr \text{ untuk } A \rightarrow B, dr(e) = d(e) + r(u) - r(v) = 0 + 1 - 0 = 1$$

$$\text{Menghitung } dr \text{ untuk } A \rightarrow C, dr(e) = d(e) + r(u) - r(v) = 0 + 1 - 0 = 1$$

$$\text{Menghitung } dr \text{ untuk } A \rightarrow G, dr(e) = d(e) + r(u) - r(v) = 0 + 1 - 0 = 1$$

Tubuh loop dapat direpresentasikan sebagai berikut:

```
{B[i]=A[i]×4;
C[i]=A[i]-10;
G[i]=A[i]+2;
D[i]=B[i]+C[i]+G[i];
A[i+1]=D[i-1]+12; }
```

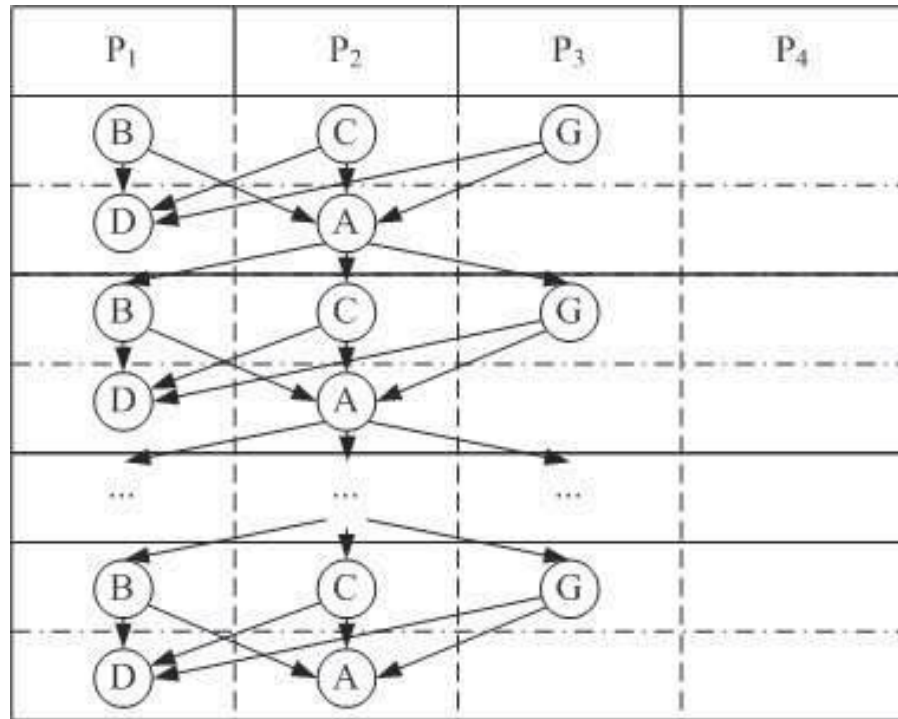
3. Rotasi

Gambar 8.12 memberikan status sebelum mengelompokkan ulang loop. Node A dipisahkan dari loop, dan node lain mengikuti jadwal loop statis. Langkah (2) menunjukkan bahwa node A memiliki delay satu unit waktu untuk ketiga node berikutnya, B, C, dan G. Karena ketergantungan node, node A dapat diputar sehingga dapat menjadi node penerus dari node B, C, dan G. Gambar 8.13 menunjukkan PDFG setelah rotasi. Seperti yang ditunjukkan pada gambar, PDFG yang dikelompokkan ulang hanya memiliki dua siklus eksekusi, yang berarti panjang jadwal perulangan adalah 2. Program berikut adalah kode dari perulangan ekuivalen setelah pengelompokan ulang badan perulangan. Tubuh loop dapat direpresentasikan sebagai berikut:

```
B[N]=A[N]×4;
C[N]=A[N]-10;
G[N]=A[N]+2;
D[N]=B[N]+C[N]+G[N];
```

Gambar 8.14 mengilustrasikan iterasi PDFG yang dioptimalkan. Solusi ini juga membutuhkan tiga prosesor. Dibandingkan dengan solusi yang ditunjukkan oleh Gambar 8.9, jadwal loop yang dioptimalkan memiliki siklus eksekusi yang lebih pendek. Panjang jadwal loop telah dikurangi dari 3 menjadi 2. Jika program

dijalankan 100 kali, jumlah siklus eksekusi total untuk loop setelah pengoptimalan akan menjadi 200, dari 2×100 . Efisiensi kerja telah ditingkatkan setidaknya 30% pada kasus ini.



Gambar 8.14 Iterasi dari grafik aliran data probabilistik setelah retime.

8.5 KESIMPULAN

Bab ini berfokus pada pengoptimalan penjadwalan loop untuk aplikasi seluler. Pentingnya memahami skema yang diperkenalkan dalam bab ini adalah untuk mengatasi dua aspek yang memakan sumber daya, yaitu, waktu komputasi dan konsumsi energi, dengan mengoptimalkan penjadwalan loop. Setelah mempelajari bab ini, siswa harus memahami konsep kunci optimasi loop dan mampu merancang loop optimal untuk kinerja tinggi aplikasi seluler.

8.6 LATIHAN

8.6.1 Pertanyaan Mendasar

1. Apa yang dimaksud dengan Graf Asiklik Berarah? Lalu, apa itu Grafik Aliran Data?
2. Apa yang dimaksud dengan Jadwal Statis?
3. Jelaskan metode penentuan Periode Siklus.
4. Jelaskan konsep retiming dan kapan teknik retiming dapat mengoptimalkan jadwal loop.
5. Apa yang dimaksud dengan Unfolding, dan peran apa yang dimainkan dalam proses optimasi penjadwalan loop? Apa faktor yang tidak terungkap?
6. Jelaskan pemahaman Anda tentang Iterasi.
7. Apa konsep Periode Iterasi?

8. Apa konsep dari Iteration Bound? Coba bandingkan periode iterasi dengan iterasi terikat dan jelaskan perbedaan antara kedua konsep ini.
9. Apa itu penjadwalan statis? Untuk apa kita perlu menggunakan penjadwalan statis?
10. Jelaskan prinsip operasi dasar menggunakan retiming untuk mengoptimalkan penjadwalan loop.
11. Apa yang dimaksud dengan Grafik Aliran Data Probabilistik (PDFG)? Mengapa kita membutuhkan PDFG untuk menyelesaikan beberapa masalah dalam praktik?
12. Dua parameter optimasi penting dipertimbangkan dalam bab ini. Apakah mereka? Gunakan beberapa kalimat sederhana untuk menjelaskan pentingnya kedua parameter tersebut.
13. Apa Pengertian Arsitektur Paralel?
14. Apa Pengertian Parallel Computing?
15. Gunakan kalimat Anda sendiri untuk menjelaskan perbedaan antara arsitektur paralel dan komputasi paralel.
16. Gunakan beberapa kalimat sederhana untuk menggambarkan hubungan antara penjadwalan loop dan komputasi paralel dalam pengoptimalan seluler.

8.6.2 Pertanyaan Praktis

Pertanyaan-pertanyaan berikut memberikan siswa kesempatan untuk mempraktekkan teknik optimasi yang dipelajari dalam bab ini. Dua bagian penting yang tercakup dalam pertanyaan adalah menggambar grafik aliran data dan menggunakan retiming.

1. Latihan Menggambar Grafik Aliran Data

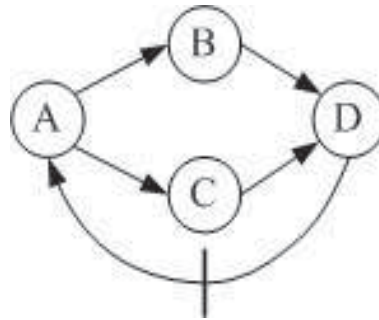
Mengacu pada program yang diberikan berikut, gambarlah grafik aliran data. Waktu tunda dapat diwakili dengan menandai garis pendek di tepi. Satu garis pendek mengacu pada satu unit waktu tunda.

Program yang diberikan:

```
Untuk (i= 1; i<N; i++) {
A[i]=D[i-3]*3;
B[i]=A[i]-8;
C[i]=A[i]+1;
D[i]=B[i]+C[i];
E[i]=C[i]*D[i];}
```

2. Praktek Teknik Retiming (a)

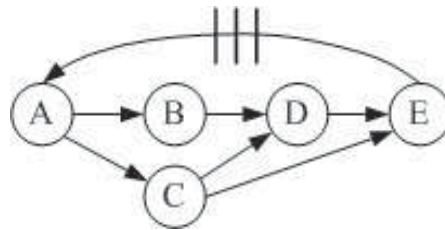
Gambar 8.15 memberikan PDFG sederhana yang memiliki empat node, termasuk node A, B, C, dan D. Asumsikan waktu komputasi pada semua node adalah 1 unit waktu. Tiga prosesor tersedia untuk pengoptimalan Anda. Ada satu waktu penundaan unit di tepi $D \rightarrow A$. Misi Anda adalah menggunakan teknik yang diperkenalkan dalam bab ini untuk mengoptimalkan PDFG ini dan menggambar grafik yang dikelompokkan kembali. Untuk menjelaskan prosedur analisis, siswa diminta untuk menggambar peta penjadwalan langkah demi langkah.



Gambar 8.15 Gambar untuk pertanyaan praktis 2.

3. Praktek Teknik Retiming (b)

Gambar 8.16 memberikan PDFG yang lebih rumit daripada Soal Praktikum 2). Ada lima node dalam PDFG ini, yaitu node A, B, C, D, dan E. Asumsikan waktu komputasi pada semua node adalah 1 unit waktu. Empat prosesor tersedia untuk pengoptimalan Anda. Ada tiga unit waktu tunda di tepi $E \rightarrow A$. Misi Anda adalah menggunakan teknik yang diperkenalkan dalam bab ini untuk mengoptimalkan PDFG ini dan menggambar grafik yang dikelompokkan kembali. Untuk menjelaskan prosedur analisis, siswa diminta untuk menggambar peta penjadwalan langkah demi langkah.



Gambar 8.16 Gambar untuk pertanyaan praktis 3.

4. Praktik Optimasi Penjadwalan Loop

Tugas 1: Diberikan program W di bawah ini, harap gambarkan Grafik Aliran Data (DFG) yang sesuai.

```
Untuk (i= 1 sampai N)
  {A[i]=D[i-2]+6;
  B[i]=A[i]*2;
  C[i]=A[i]-7;
  G[i]=A[i]+1 D[i]=B[i]+C[i]+G[i];}
```

Tugas 2: Kami memiliki empat prosesor, P1, P2, P3, dan P4, tersedia. Misi pertama Anda adalah mencari tahu jumlah siklus eksekusi total tanpa retime setelah menjalankan program W 100 kali.

Misi kedua Anda adalah menghitung jumlah siklus eksekusi total dengan menggunakan retiming setelah menjalankan program W 100 kali. Coba bandingkan dua hasil yang dihasilkan dari kedua pendekatan tersebut. Untuk menjelaskan prosedur analisis, siswa diminta untuk menggambar peta penjadwalan langkah demi langkah.

8.7 DAFTAR ISTILAH

Graf Asiklik Berarah

adalah pendekatan yang menggunakan diagram grafik untuk mewakili ketergantungan data atau unit fungsional antar operasi.

Jadwal Statis

node harus mengikuti urutan prioritas yang ditentukan oleh DFG yang diberikan.

Periode Siklus

mengacu pada total waktu komputasi dari jalur zero-delay terpanjang di DFG yang diberikan.

Pengunduran waktu

adalah pendekatan optimasi yang mengubah DFG menjadi waktu polinomial dan meminimalkan periode siklus dengan mendistribusikan ulang register atau penundaan dalam rangkaian.

Terungkap

adalah teknik yang dapat mengoptimalkan penjadwalan loop dengan meningkatkan periode siklus rata-rata dari jadwal statis.

Paraleisme Tingkat Instruksi

adalah paradigma yang diterapkan untuk menguji kemampuan operasi simultan dalam program komputasi atau aplikasi.

Pengulangan

mewakili pengulangan proses yang memiliki setiap node dalam satu set node dieksekusi tepat satu kali.

Periode Iterasi

mengacu pada waktu komputasi rata-rata dari suatu iterasi.

Terikat Iterasi

berarti rasio maksimum waktu komputasi terhadap waktu tunda untuk semua siklus dalam DFG.

Jadwal Statis

berarti node harus mengikuti urutan prioritas yang ditentukan oleh grafik aliran data yang diberikan.

Faktor Pembukaan

angka yang menunjukkan berapa kali DFG asli dibuka.

Algoritma Penjadwalan Rotasi

adalah algoritma penjadwalan polinomial-waktu yang mempertimbangkan kendala sumber daya.

Grafik Aliran Data Probabilistik

adalah DFG tingkat lanjut yang digunakan untuk mengoptimalkan sistem yang memiliki tugas dengan waktu komputasi yang tidak pasti.

Arsitektur Paralel

mengacu pada jenis arsitektur komputasi yang mendukung beberapa operasi simultan.

Komputasi Paralel

adalah paradigma komputasi yang secara bersamaan mengeksekusi beberapa perhitungan dengan menggunakan beberapa prosesor.

Sistem Komputasi Paralel

adalah komputer atau sistem yang menerapkan beberapa prosesor untuk menyelesaikan beberapa proses paralel.

BAGIAN III

TEKNIK APLIKASI SELULER DALAM TEKNOLOGI BERKEMBANG

BAB 9

KOMPUTASI CLOUD SELULER DALAM PENERAPAN APLIKASI SELULER

Mobile Cloud Computing telah menjadi topik hangat di bidang aplikasi mobile Android. Bab ini bertujuan untuk memberikan siswa pandangan panorama tentang komputasi awan seluler dan implementasinya dalam pengembangan aplikasi seluler. Sejumlah konsep dan teknik kunci yang terkait dengan komputasi awan bergerak diperkenalkan dalam bab ini. Sepanjang bab ini, siswa harus dapat memahami teknik penting yang digunakan di cloud seluler dan arsitektur terapan umum. Pengalaman belajar ini bertujuan untuk memungkinkan siswa membangun koneksi antara cloud seluler dan keterampilan pengembangan aplikasi seluler. Tujuan utama mempelajari bab ini adalah untuk memfasilitasi siswa untuk memperoleh kemampuan mengembangkan aplikasi seluler melalui pendekatan berbasis cloud. Isi utama bab ini meliputi:

1. Konsep yang terkait dengan komputasi awan seluler
2. Arsitektur komputasi awan seluler
3. Struktur teknologi cloud seluler
4. Teknologi utama di cloud seluler

9.1 PENDAHULUAN

Bab ini menginstruksikan siswa tentang komputasi awan seluler untuk membantu siswa dalam menjembatani awan seluler dengan keterampilan pengembangan yang dipelajari di bab-bab sebelumnya. Bab ini berfokus pada pengenalan tampilan holistik cloud seluler dengan detail untuk setiap dimensi teknis. Struktur teknologi mobile cloud terdiri dari tiga aspek, mobile computing, mobile internet, dan cloud computing. Teknik penting yang diterapkan di setiap aspek adalah konten instruksional yang penting dalam bab ini, dan diselaraskan dengan pengembangan aplikasi seluler dan teknik pengoptimalan. Tujuan mempelajari bab ini ada dua. Pertama, mahasiswa harus memiliki kognisi yang baik tentang implementasi mobile cloud dengan mempersepsikan berbagai pilihan teknologi. Kedua, mahasiswa perlu memiliki kemampuan memahami dan menggunakan arsitektur mobile cloud computing dalam skenario dunia nyata. Sepanjang bab ini, siswa harus dapat menjawab pertanyaan-pertanyaan berikut:

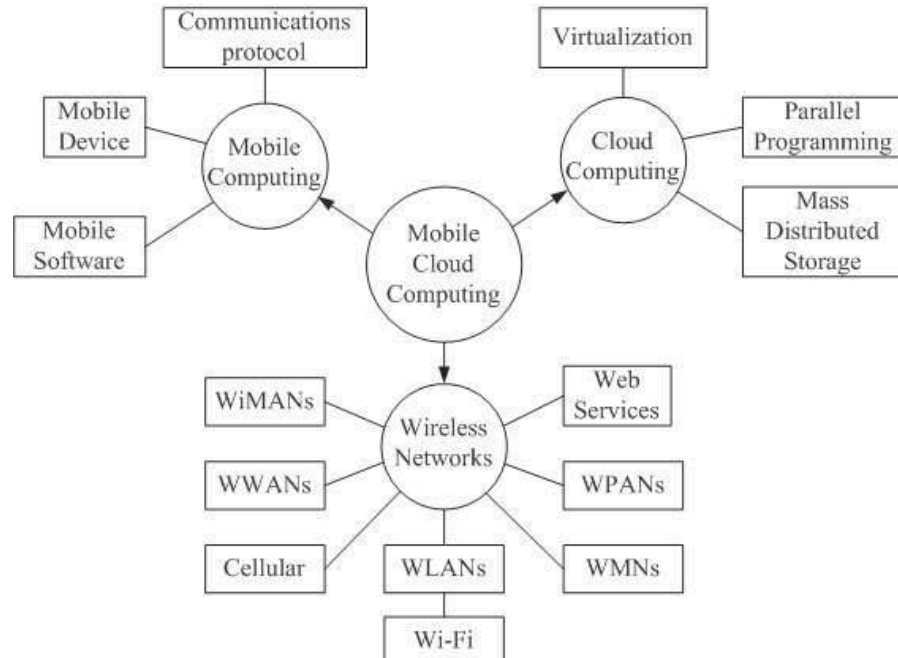
1. Apa perbedaan utama antara komputasi awan seluler dan komputasi awan?
2. Apa saja komponen utama dari struktur teknologi mobile cloud? Hubungan apa yang dimiliki komponen-komponen ini?
3. Apa saja teknologi utama yang diterapkan dalam komputasi awan? Fitur apa yang mereka miliki?
4. Apa arsitektur cloud seluler dan bagaimana cara kerjanya?

9.2 KONSEP KOMPUTASI CLOUD SELULER

Mobile Cloud Computing (MCC) adalah paradigma teknologi yang menggunakan pendekatan berbasis cloud untuk menyediakan pengguna seluler yang menggunakan berbagai layanan dengan teknologi dan perangkat seluler. Pendekatan ini memberikan layanan berbasis cloud kepada pengguna akhir seluler dengan memanfaatkan manfaat perangkat seluler dan memperluas cakupan layanan. Ada dua migrasi beban kerja utama yang memanfaatkan keunggulan server jarak jauh, termasuk pemrosesan data dan penyimpanan data. Skema untuk mencapai migrasi operasi adalah dengan memindahkan pemrosesan dan penyimpanan data di lokasi ke server fisik jarak jauh. Nilai dari penerapan solusi berbasis MCC adalah untuk meningkatkan kemampuan perangkat seluler dengan mengalihkan beban kerja ke server cloud. Perhitungan aplikasi seluler dapat diselesaikan pada server jarak jauh, yang menyediakan aplikasi seluler dengan pendekatan untuk menghasilkan kinerja tingkat yang lebih tinggi.

9.2.1 Struktur Teknologi Mobile Cloud Computing

Aplikasi MCC diterapkan untuk menyediakan layanan berbasis cloud seluler yang diwakili oleh perangkat seluler. Representasi layanan memerlukan dukungan dari beberapa aspek teknologi. Teknologi utama melibatkan tiga dimensi, yaitu komputasi bergerak, jaringan nirkabel, dan komputasi awan. Sebagian besar layanan PKS saat ini dapat dianggap sebagai kombinasi dari ketiga sudut ini.



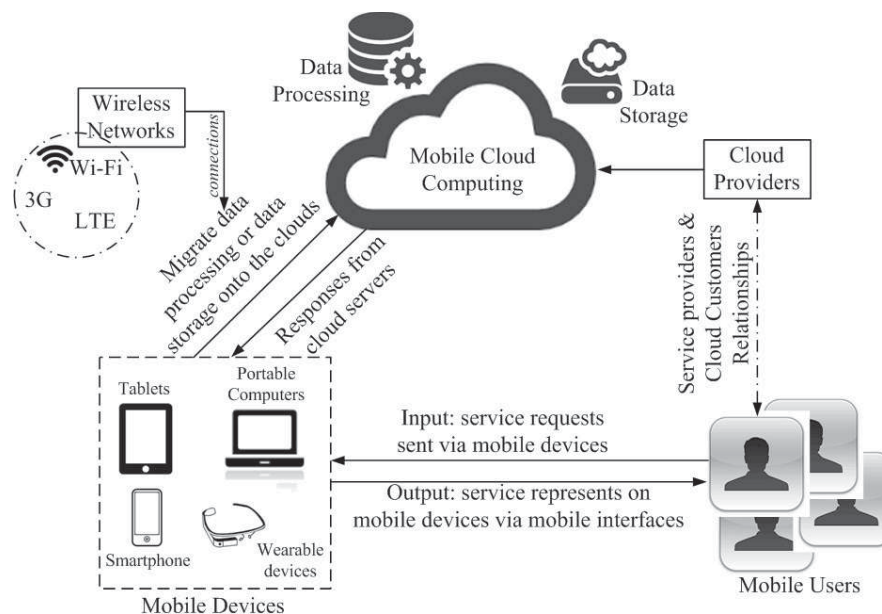
Gambar 9.1 Struktur teknis komputasi awan bergerak.

Gambar 9.1 menunjukkan struktur teknis PKS dengan deskripsi singkat tentang teknik yang sesuai di setiap dimensi. Gambar 9.2 menunjukkan diagram alir layanan untuk memanfaatkan aplikasi berbasis PKS yang berjalan pada perangkat seluler. Tiga komponen penting dalam diagram adalah komputasi seluler, jaringan nirkabel, dan server cloud. Komunikasi antara perangkat seluler dan server cloud dilakukan oleh

berbagai jaringan nirkabel. Presentasi layanan diberikan pada perangkat seluler di mana pengguna seluler dapat mengakses layanan. Ketiga bagian ini adalah tiga teknik kunci untuk menghasilkan solusi berbasis PKS.

9.2.2 Perbedaan antara Cloud Computing dan Mobile Cloud

Dalam kebanyakan situasi, komputasi awan berbagi sebagian besar pendekatan model layanan dan penyebaran dengan PKS. Konsep Cloud Computing didefinisikan sebagai paradigma komputasi yang menggunakan teknologi berbasis Web untuk menyediakan pengguna dengan layanan scalable on-demand dengan berbagi atau menawarkan sumber daya komputasi. Mengingat sarana penyampaian layanan, masih ada dua perbedaan antara komputasi awan dan PKS, meskipun PKS awalnya berasal dari komputasi awan.



Gambar 9.2 Diagram pemanfaatan komputasi awan seluler.

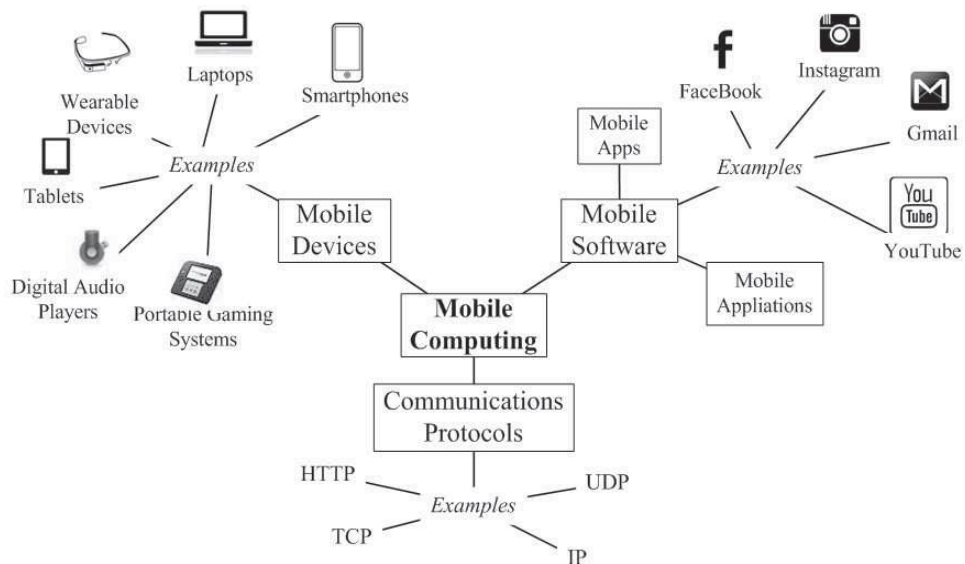
Pertama, model MCC mungkin memiliki fokus yang berbeda dari komputasi awan umum. Baik komputasi awan maupun solusi MCC dapat dilakukan pada jaringan kabel dan nirkabel. Namun, layanan MCC menonjolkan peran komunikasi nirkabel dan interkoneksi daripada memvirtualisasikan sumber daya komputasi jarak jauh, yang ditekankan oleh komputasi awan. Dibandingkan dengan komputasi awan, jaringan nirkabel merupakan komponen mendasar dalam PKS. Solusi berbasis MCC harus dirancang, dikembangkan, dan diimplementasikan dengan mempertimbangkan penggunaan dalam lingkungan operasi nirkabel. Memanfaatkan teknologi seluler memungkinkan MCC menjadi pendorong komunikasi seluler yang cerdas.

Kedua, penawaran layanan MCC adalah layanan berorientasi mobilitas yang menekankan penggunaan perangkat seluler, antarmuka, dan platform. Ini membutuhkan solusi berbasis MCC tidak hanya melakukan fungsionalitas yang dapat dilakukan oleh solusi lokal, tetapi juga memperkenalkan ekstensi berbasis mobilitas, seperti komunikasi waktu nyata, sinkronisasi data nirkabel, dan fitur layar sentuh.

Ekstensi ini dianggap nilai penting untuk menghasilkan layanan PKS. Tidak seperti PKS, layanan komputasi awan menyebarkan penawaran layanan mereka di domain dengan jangkauan yang lebih luas.

9.2.3 Komputasi Seluler

Istilah Komputasi Seluler adalah sekelompok teknologi untuk membangun konektivitas dan komunikasi antara perangkat seluler di Web. Teknologi mencakup setidaknya tiga aspek: protokol komunikasi, perangkat seluler, dan perangkat lunak seluler. Sebagian besar komunikasi data membutuhkan operasi assortatif yang menggabungkan ketiga teknologi ini. Gambar 9.3 merepresentasikan struktur konseptual dari aspek-aspek kunci dalam komputasi seluler. Sejumlah contoh di setiap aspek juga diberikan dalam gambar.



Gambar 9.3 Tiga aspek utama komputasi seluler dan contohnya.

Protokol Komunikasi adalah seperangkat aturan yang memastikan semua perangkat seluler yang terlibat dalam komunikasi dapat saling memahami saat data dikirimkan. Aturan dalam protokol komunikasi perlu mengatur transmisi data berdasarkan tujuan yang ditentukan. Misalnya, inti protokol Internet dasar adalah Transmission Control Protocol (TCP) dan Internet Protocol (IP), yang sering disebut Model TCP/IP. Model protokol ini bertujuan untuk menyediakan komputasi yang dikomunikasikan dengan transmisi data yang andal dan aman. Sebagai gantinya, protokol umum lainnya, Connectionless User Datagram Protocol (UDP), terutama digunakan untuk komunikasi cepat dengan persyaratan keamanan tingkat rendah. Aspek berikutnya dari komputasi mobile adalah perangkat mobile. Perangkat Seluler mengacu pada seperangkat perangkat portabel atau perangkat yang dapat dipakai dengan kemampuan komunikasi nirkabel, tempat aplikasi dapat dijalankan. Perangkat seluler yang umum termasuk laptop, tablet, ponsel pintar, perangkat yang dapat dikenakan, pemutar audio digital, dan sistem permainan portabel. Mengatasi masalah mobilitas, perangkat seluler juga dianggap sebagai platform di mana pengguna akhir dapat memperoleh

layanan berbasis Internet atau mengakses database online tanpa batasan jaringan kabel.

Selanjutnya, Perangkat Lunak Seluler mengacu pada aplikasi atau aplikasi yang berjalan di perangkat seluler yang digunakan untuk mewakili layanan seluler. Dalam disiplin ilmu komputer, konsep aplikasi berbeda dengan aplikasi. Kedua konsep tersebut digunakan untuk menggambarkan program komputer dari representasi layanan yang berjalan pada perangkat seluler. Namun, ada perbedaan besar antara aplikasi dan aplikasi. Aplikasi berjalan langsung di Sistem Operasi (OS) tetapi aplikasi berjalan dalam kerangka kerja, seperti Android. Misalnya, program yang berjalan di Android adalah aplikasi Android. Menggunakan pendekatan pengembangan aplikasi dalam kerangka kerja dapat sangat mengurangi waktu pengembangan aplikasi.

Ada beberapa pendekatan bagi pengguna seluler untuk memperoleh aplikasi atau aplikasi seluler. Cara yang umum adalah mengunduh perangkat lunak seluler dari Application Distribution Platforms (AppDP). Secara umum, AppDP adalah antarmuka untuk menyediakan pembelian dan pengunduhan perangkat lunak, yang dimiliki dan dioperasikan oleh penyedia OS. Misalnya, AppDP khas saat ini termasuk App Store, Amazon Appstore, Google Play, Windows Store, Windows Phone Store, dan Samsung Apps Store. Beberapa contoh aplikasi seluler populer adalah Facebook, Google Maps, Gmail, Instagram, dan YouTube.

9.2.4 Jaringan Nirkabel

Internet Seluler, juga dikenal sebagai Teknologi Jaringan Nirkabel, adalah seperangkat teknologi jaringan canggih yang mengaktifkan konektivitas antar komunikator melalui jaringan nirkabel yang didukung oleh perangkat lunak seluler. Komunikator di Internet seluler adalah sejumlah Node Jaringan, yang mengacu pada beberapa jenis lokasi pemrosesan yang terjadi di berbagai infrastruktur digital. Node jaringan ini dapat dilihat sebagai beberapa titik penghubung dan komunikatif dengan tujuan atau fungsi yang berbeda. Semua aktivitas yang terjadi di node jaringan harus mengikuti aturan jaringan yang ditentukan oleh protokol yang digunakan. Intinya, node jaringan itu sendiri dapat mengenali tindakan selanjutnya dengan kemampuan pemrogramannya. Jenis dasar dari node jaringan termasuk koneksi dan node titik akhir, yang menunjukkan dua gerakan mendasar selama proses transmisi data.

Selain itu, ada berbagai jenis jaringan nirkabel untuk tujuan penggunaan yang berbeda. Saat ini, jenis jaringan nirkabel umum termasuk Wireless Personal Area Networks (WPANs), Wireless Local Area Networks (WLANs), Wireless Mesh Networks (WMNs), Wireless Metropolitan Area Networks (WiMANs), Wireless Wide Area Networks (WWANs), dan Jaringan Seluler (Seluler). Tabel 9.1 mengevaluasi perbandingan antara jenis jaringan nirkabel utama saat ini. Secara umum, jaringan nirkabel bergerak masih menghadapi keterbatasan bandwidth ketika cakupan cakupan layanan semakin besar dan tingkat mobilitas semakin tinggi. Melayani cakupan area berukuran lebih kecil adalah pendekatan yang efisien bagi pengguna ponsel untuk mendapatkan komunikasi yang lebih cepat dengan menggunakan titik akses lokal yang terhubung ke Internet.

Karena keterbatasan ruang, bagian ini terutama memperkenalkan tiga jenis jaringan yang diadopsi secara luas di industri saat ini, termasuk WLAN, WWAN, dan Seluler.

Catatan: Topologi Wireless Mesh adalah jenis jaringan nirkabel yang menghubungkan semua perangkat seluler.

Jaringan Area Lokal Nirkabel (WLAN)

Wireless Local Area Networks (WLANs), juga disingkat Wireless LAN atau WLAN, adalah metode penyebaran nirkabel untuk penggunaan area kecil lokal melalui koneksi radio frekuensi tinggi. Akses koneksi Internet dapat berupa jaringan kabel yang ada, seperti koneksi Ethernet, atau sumber nirkabel lainnya. Jangkauan cakupan biasanya tergantung pada kemampuan peralatan. Kecepatan komunikasi bervariasi dari 1 hingga 600 Mbps karena kinerja dan standar produk yang berbeda. Saat ini jenis jaringan nirkabel ini merupakan salah satu jenis yang paling populer digunakan dalam praktek karena banyak kelebihanannya.

Pertama, pembentukan WLAN sederhana untuk sebagian besar pengguna dan penyedia layanan. WLAN terdiri dari sejumlah stasiun nirkabel. Stasiun Nirkabel di WLAN mengacu pada peralatan apa pun yang terhubung ke media nirkabel dalam jaringan nirkabel. Pada dasarnya ada dua jenis stasiun nirkabel: Stasiun Titik Akses Nirkabel (WAPS) dan Stasiun Titik Klien Nirkabel (WCPS). WAPS memainkan peran komunikator yang menerima dan mengirimkan sinyal radio frekuensi tinggi untuk perangkat yang terlibat dalam komunikasi nirkabel. Router nirkabel adalah contoh WAPS, yang sering digunakan untuk mendistribusikan sinyal radio dengan menghubungkan dengan jaringan kabel atau nirkabel. WCPS berarti perangkat bergerak atau semua perangkat tetap dengan antarmuka jaringan nirkabel, yang menggunakan layanan jaringan nirkabel. Misalnya, ponsel, laptop portabel, dan desktop yang terhubung nirkabel adalah semua stasiun klien dalam WLAN.

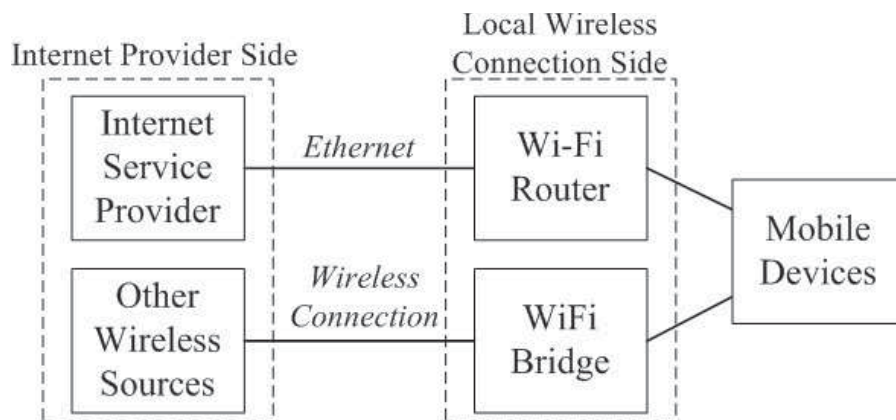
Kedua, WLAN memiliki tingkat fleksibilitas yang tinggi sehingga dapat digunakan untuk memenuhi berbagai tuntutan. Beberapa metode komunikasi dapat didukung oleh WLAN, yang meliputi peer-to-peer dan point-to-point. Point-to-Point adalah metode komunikasi yang mendukung koneksi nirkabel antara node jaringan atau titik akhir.

Tabel 9.1 Perbandingan Berbagai Jenis Jaringan Nirkabel Saat Ini. T (Tipe), P (Kinerja), CC (Kemampuan Cakupan)

T	P	Aplikasi	CC
WPAN	Sedang	Tujuan penggunaan pribadi	Kecil
WLAN	Tinggi	Penggunaan komunitas atau organisasi; menghubungkan perangkat seluler melalui titik akses yang terhubung ke Internet (WiFi)	Dalam suatu wilayah
WMN	Tinggi	Menggunakan node radio dalam topologi mesh untuk	Global

		menghubungkan klien mesh yang merupakan perangkat seluler	
WiMAN	Tinggi	Mencakup area yang lebih luas dari WLAN; layanan nirkabel ditawarkan dan dioperasikan oleh organisasi di sebagian besar situasi	Di dalam kota
WWAN	Rendah	Akses seluler ke jaringan nirkabel	Global
seluler	Rendah	Akses seluler ke jaringan nirkabel dengan menggunakan jaringan radio terdistribusi gaya sel	Global

Misalnya, laptop dapat memiliki koneksi nirkabel ke printer dengan menggunakan metode komunikasi point-to-point. Selanjutnya, metode jaringan Peer-to-Peer (P2P) adalah pendekatan membangun jaringan dengan menghubungkan node jaringan yang merupakan klien jaringan dan server. Dalam kebanyakan situasi, jaringan P2P dibuat dalam sistem Terdesentralisasi. Sistem Jaringan Terdesentralisasi adalah model jaringan terdistribusi di mana setiap node jaringan hanya bekerja pada operasi lokal dan memiliki tanggung jawab yang sama terhadap jaringan. Selain itu, Sistem Jaringan Terpusat adalah model jaringan yang terdiri dari node master dan node titik akhir, dan cocok untuk sekelompok kecil pengguna yang bekerja bersama dalam ruang terbatas. Aplikasi khas WLAN adalah Wireless Fidelity (Wi-Fi), yang telah menjadi cara populer untuk memanfaatkan WLAN. Bagian berikut menjelaskan perbedaan utama antara dua konsep yang mirip: WLAN dan Wi-Fi. Bagian ini juga memberikan peta konseptual untuk membangun Wi-Fi untuk membantu siswa lebih memahami penempatan teknologi ini. Gambar 9.4 menunjukkan struktur jaringan Wi-Fi. Menurut representasi gambar, Wi-Fi dapat memperoleh sumber nirkabel dari sumber Internet kabel dan nirkabel.



Gambar 9.4 Struktur jaringan Wi-Fi.

Perbandingan antara Jaringan Area Lokal Nirkabel dan Fidelity Nirkabel

Arti WLAN berbeda dari Wi-Fi, meskipun kedua konsep ini sering merujuk pada teknologi yang sama dalam kehidupan kita sehari-hari. Dalam perspektif teknik, mereka adalah dua konsep yang berbeda dalam subjek ilmu komputer. Siswa harus memahami perbedaan utama antara kedua definisi ini. Perbedaan utama antara WLAN dan Wi-Fi adalah bahwa Wi-Fi hanya satu jenis WLAN. Pengguna WLAN dapat memiliki akses ke jaringan nirkabel melalui koneksi radio. Wi-Fi mengacu pada sekelompok produk yang mengikuti keluarga protokol nirkabel 802.11 di WLAN.

Jaringan Area Luas Nirkabel

Wireless Wide Area Networks (WWANs), juga dikenal sebagai Wirelss WAN atau Wireless broadband, adalah jaringan nirkabel besar penggunaan geografis di mana sejumlah besar sel mengirimkan sinyal radio ke perangkat mobile dan on-premise. Model jaringan nirkabel ini memiliki kinerja yang efisien dalam mentransmisikan sinyal ke perangkat yang bergerak cepat dibandingkan dengan WLAN yang hanya dapat melayani peralatan yang bergerak lambat dalam situs jangkauan terbatas.

Teknologi utama WWAN termasuk Global System for Mobile Communications (GSM), Code Division Multiple Access (CDMA), dan Worldwide Interoperability for Microwave Access (WiMAX). GSM dan CDMA adalah dua teknologi seluler lama, dan keduanya memanfaatkan Akses Paket Berkecepatan Tinggi (HSPA) dan Evolution-Data Optimized (EV-DO) untuk mengirimkan data.

GSM adalah standar global untuk mengoperasikan komunikasi bergerak, yang dikembangkan oleh European Telecommunications Standards Institute (ETSI). Jaringan GSM dasar terdiri dari beberapa subsistem, termasuk Base Station Subsystem (BSS), Network Switching Subsystem (NSS), Operation Support Subsystem (OSS), dan Mobile Station. Stasiun bergerak adalah peralatan pengguna yang berkomunikasi dengan BSS yang bertanggung jawab atas transceiver dan pengontrol sinyal. NSS terhubung dengan BSS melalui antarmuka yang ditentukan di mana komunikasi dapat dilakukan dan dialihkan di antara jaringan yang berbeda. Baik NSS dan BSS didukung oleh OSS. Dalam prakteknya, beberapa subsistem tambahan ditambahkan ke dalam struktur GSM ini sesuai dengan kebutuhan database atau komunikasi, seperti Equipment Identity Register dan Chargeback Center.

Selanjutnya, CDMA adalah pendekatan lain untuk komunikasi seluler yang dapat mengirimkan beberapa sinyal digital dengan menghubungkan beberapa terminal melalui media transmisi yang sama. CDMA adalah implementasi dari Channel-Access Schema yang memungkinkan aliran data atau sinyal radio yang berbeda melewati atau berbagi saluran komunikasi yang sama. Metode ini juga didefinisikan sebagai Multiplexing. Teknologi utama dari pendekatan ini adalah menggunakan Spread-Spectrum yang memperluas kapasitas bandwidth dengan memvariasikan frekuensi sinyal yang ditransmisikan. Proses operasional mencakup tiga langkah: multiplexing sinyal input, membawa sinyal multiplexed melalui link data rate yang tinggi, dan demultiplexing sinyal data sebagai data output. Akses beberapa saluran memfasilitasi CDMA untuk menjalankan kinerja yang lebih baik daripada GSM.

Terakhir, WiMAX merupakan alternatif pengganti GSM dan CDMA atau peningkatan kapasitas jaringan yang sudah ada. Penyebaran WiMAX mirip dengan Wi-Fi, tetapi menawarkan cakupan yang lebih besar dan mendukung lebih banyak pengguna seluler. Kecepatan transmisi data hingga 1 gigabit/detik, yang didasarkan pada standar IEEE 802.16. WiMAX mendukung beberapa topologi Radio Access Network (RAN) dan WiMAX tipikal memiliki tiga bagian: Mobile System, Access Service Network (ASN), dan Connectivity Service Network (CSN). Sistem seluler adalah platform bagi pengguna untuk mengakses jaringan, yang mirip dengan sistem seluler di GSM. ASN terdiri dari beberapa BSS dan gateway ASN di mana sinyal radio dapat mengakses jaringan. CSN adalah komponen interkoneksi yang menghubungkan semua IP dan menyelaraskan semua jaringan inti IP yang terpasang Tabel 9.2 menampilkan perbandingan antara WiMAX dan Wi-Fi dalam lima perspektif, yaitu standar, cakupan, bandwidth, mobilitas, dan pengguna.

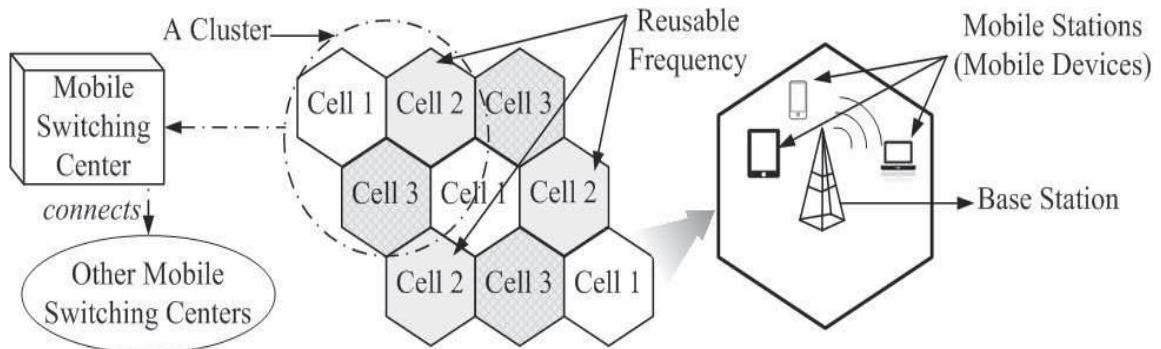
Jaringan Seluler

Jaringan Seluler mengacu pada jenis jaringan seluler yang terdiri dari sejumlah besar stasiun pangkalan radio. Setiap base station mencakup ruang lingkup khusus area yang didefinisikan sebagai "sel". Sel-sel yang bersebelahan tidak menggunakan frekuensi yang sama untuk menghindari interferensi. Jaringan seluler adalah pendekatan efisien yang mendukung penggunaan kembali frekuensi yang sama dengan beberapa teknik, seperti sel terdistribusi, pengkodean sinyal sel, atau teknik panggilan switching. Jaringan seluler dasar terdiri dari stasiun bergerak, stasiun pangkalan, dan Pusat Pengalihan Seluler (MSC). Setiap base station adalah media yang secara simultan interkoneksi dengan semua mobile station dalam cakupan cakupannya dan membawa transmisi data ke MSC. MSC bertanggung jawab untuk merutekan data dengan menerima lalu lintas dan mengalihkan panggilan antar jaringan seluler. Terkadang, MSC juga memainkan peran manajemen yang mengatur dan mengontrol koneksi nirkabel dan sel atau cluster. Cluster adalah sekelompok sel yang berdampingan. Gambar 9.5 mengilustrasikan contoh struktur jaringan seluler.

Tabel 9.2 Perbandingan antara WiMAX dan Wi-Fi

	WiMAX	Wifi
Standar	IEEE 802.16	Keluarga standar IEEE 802.11
liputan	Area yang luas (jangkauan hingga 40 mil per antena WiMAX)	Area kecil (biasanya hingga beberapa ratus kaki)
Bandwidth	rentang bandwidth yang dapat disesuaikan dari 1,25 hingga 20 MHz	20 atau 25 MHz

Mobilitas	Tersedia (teknologi dirancang untuk pengguna seluler)	Tidak ada (pengguna yang bergerak lambat di ruang terbatas)
Pengguna	Skalabilitas besar, dari satu hingga ratusan pengguna tergantung pada permintaan	Jumlah pengguna terbatas tergantung pada perangkat



Gambar 9.5 Contoh struktur jaringan seluler.

Seperti yang ditunjukkan pada gambar, sekelompok sel mewakili jaringan seluler. Sel dengan warna dan gaya yang sama berarti frekuensi yang digunakan sama. Sel-sel yang berdekatan menggunakan frekuensi yang berbeda untuk menghindari interferensi satu sama lain. Misalnya, pada Gambar 9.5, tidak ada Sel 2 yang merupakan sel tetangga, yang menunjukkan bahwa semua Sel 2 memanfaatkan frekuensi yang sama. Frekuensi yang sama yang digunakan dalam jaringan seluler disebut Frekuensi yang Dapat Digunakan Kembali. Selain itu, sekelompok sel terhubung dengan MSC yang terhubung ke MSC lain.

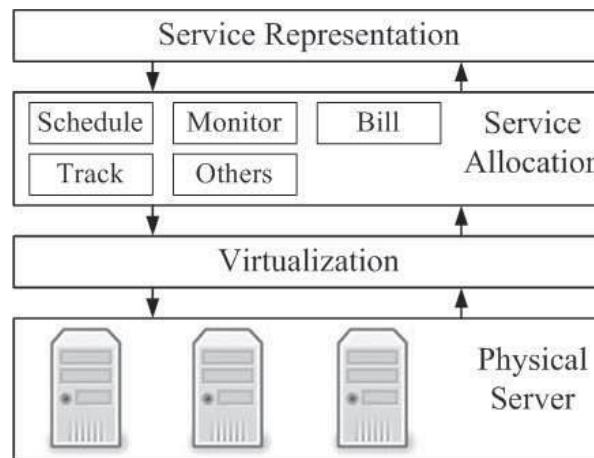
9.3 TEKNIK UTAMA MOBILE CLOUD COMPUTING

Bagian ini memperkenalkan teknik utama yang diterapkan di PKS dari berbagai pandangan. Tiga skema utama dalam komputasi awan adalah Virtualisasi, Model Pemrograman Paralel, dan Penyimpanan Terdistribusi Massal. Ketiga teknologi ini menangani tiga aspek berbeda dalam memberikan layanan MCC, yaitu, penyampaian layanan dan komunikasi antara perangkat seluler dan server jarak jauh, pemrosesan data sisi jarak jauh, dan penyimpanan data jarak jauh.

9.3.1 Virtualisasi

Virtualisasi adalah mekanisme yang memvirtualisasikan sumber daya komputasi objek untuk mewakilinya dalam cara berbasis layanan oleh berbagai tingkat layanan. Mesin Virtual (VM) adalah pendekatan berbasis perangkat lunak yang digunakan secara luas dalam solusi cloud saat ini. Pengguna akhir dapat memperoleh layanan cloud melalui eksekusi virtualisasi yang diwakili oleh infrastruktur, platform, dan perangkat lunak. Tiga keuntungan utama penerapan VM adalah memangkas biaya, menghemat energi,

dan memfasilitasi pemeliharaan. Sebagian besar layanan cloud dapat dikirimkan ke pelanggan cloud dengan menggunakan VM.



Gambar 9.6 Contoh penggunaan mesin virtual untuk memberikan layanan cloud.

VM adalah teknik emulasi yang digunakan untuk mendistribusikan dan memvirtualisasikan beberapa sumber daya komputasi jarak jauh dan menyajikan akuisisi komputasi kepada pengguna akhir dalam mode layanan. Dalam komputasi awan, pengguna dapat menjalankan solusi berbasis cloud untuk memperoleh kinerja yang sama dengan solusi di tempat, karena layanan komputasi berjalan secara virtual di situs lokal. Implementasi VM di MCC juga efektif, karena pemrosesan data dan penyimpanan data yang berjalan di server jarak jauh direpresentasikan ke pelanggan cloud dengan mode penyajian rangsangan. Saat ini, sebagian besar sumber daya komputasi jarak jauh dapat berubah menjadi layanan cloud dengan mengadopsi teknologi VM, dari aplikasi hingga sistem operasi.

Gambar 9.6 merupakan contoh penggunaan VM untuk membawa layanan cloud ke pelanggan cloud. VM melakukan fungsi yang sama seperti server fisik jarak jauh melalui koneksi internet nirkabel atau kabel. Permintaan layanan dialokasikan ke berbagai VM tergantung pada jenis permintaan, ketika pengguna cloud mencoba mendapatkan layanan cloud. Permintaan layanan akan dikirim ke mesin fisik melalui VM di mana respons layanan akan diteruskan. Pengguna cloud dapat memperoleh respons layanan mereka setelah respons mencapai lapisan representasi layanan, yang biasanya merupakan antarmuka pengguna yang diikuti oleh API.

Selanjutnya, VM biasanya dianggap sebagai skema aman untuk melindungi informasi pengguna karena platform berbasis virtualisasi memiliki keadaan operasi independen. Sebagian besar VM diisolasi dari bagian sistem lainnya dan dijalankan sebagai aplikasi yang berjalan di sistem operasi. Kerentanan utama penerapan VM adalah bahwa serangan ke hypervisor lapisan bawah bersifat dinamis. Perubahan ancaman yang konsisten menghasilkan kesulitan keamanan yang berkelanjutan. Selain itu, berpindah VM antara server fisik yang berbeda tidak aman, karena sistem operasi yang disimulasikan tidak menghosting firewall untuk memisahkan VM. Singkatnya, solusi cloud berbasis VM adalah mekanisme yang populer di industri cloud saat ini. Ini adalah

pendekatan yang aman kecuali hypervisor lapisan bawah diserang dan dikendalikan oleh penyerang. Kekhawatiran lain dalam menggunakan solusi berbasis VM melibatkan penggunaan dan keandalan virtualisasi yang berlebihan, yang terkait dengan strategi dan kebijakan TI pengguna. Secara teknis, VM menjadi salah satu pendekatan umum untuk menghadirkan layanan pelanggan cloud dengan memanfaatkan teknologi simulasi.

9.3.2 Model Pemrograman Paralel

Parallel Programming Model (PPM) adalah teknologi yang digunakan untuk menyelesaikan tugas bersamaan berdasarkan platform berbasis cloud. Skema ini diadopsi secara luas dalam pemrosesan data paralel yang berjalan di server jarak jauh. Tujuan utama penggunaan PPM adalah menghasilkan program paralel yang dapat dikompilasi dan dioperasikan secara efisien. PPM adalah mekanisme penting bagi pengembang dan penyedia cloud seluler yang biasanya diminta untuk menawarkan layanan cepat melalui jaringan nirkabel. Ini juga merupakan pendekatan untuk berhasil menyelesaikan transmisi data berukuran besar. Prinsip dari PPM adalah membagi suatu masalah atau proses yang kompleks menjadi beberapa sub masalah atau sub proses. Dua teknik tipikal yang diterapkan dalam PPM meliputi Interacting Processes dan Decomposing Problems.

Proses Berinteraksi, juga dikenal sebagai Interaksi Proses, adalah pendekatan yang digunakan untuk meningkatkan efisiensi operasional dengan berinteraksi proses paralel. Bagian penting dari metode ini adalah untuk menemukan proporsi interaktif antara proses paralel. Dua cara umum untuk mengatasi masalah ini adalah berbagi kenangan dan menyampaikan pesan antar program.

Memori Bersama adalah metode proses interaksi yang mendukung pengiriman data antar program berdasarkan ruang alamat global bersama di mana semua program paralel dapat membaca dan menulis data secara asinkron. Ini berarti memungkinkan program untuk berjalan pada satu atau beberapa prosesor baik di situs lokal atau melalui server terdistribusi jarak jauh. Cara lain untuk mencapai interaksi proses adalah dengan menyampaikan pesan antar program. Message Passing adalah istilah yang menggambarkan solusi pertukaran data antar proses, yang dicapai dengan komunikasi antara program paralel. Message in message passing adalah jenis konten komunikasi antara program paralel, yang dapat dihasilkan ke dalam beberapa bentuk, seperti fungsi, sinyal, dan beberapa jenis paket data. Dibandingkan dengan memori bersama, penyampaian pesan memiliki tingkat fleksibilitas yang lebih tinggi, karena mendukung komunikasi sinkron dan asinkron.

Selain itu, teknik kedua dari PPM adalah penguraian masalah. Istilah Dekomposisi Masalah mengacu pada metodologi yang merumuskan program paralel. Dua metode umum yang digunakan dalam dekomposisi masalah adalah paralelisme fungsi dan paralelisme data.

Fungsi Paralelisme, juga dikenal sebagai Tugas atau Kontrol Paralelisme, adalah metode komputasi paralel antara beberapa prosesor yang mendistribusikan proses atau benang antara node komputasi paralel. Pendekatan ini sering digunakan dalam

sistem multiprosesor. Kode semu berikut memberikan contoh paralelisme fungsi. Selain itu, Paralelisme Data adalah pendekatan lain dari dekomposisi masalah yang mendistribusikan data antara node komputasi paralel alih-alih mendistribusikan proses. Sebagai pembeda dari paralelisme fungsi, paralelisme data berfokus pada penetapan data di antara multiprosesor berdasarkan beberapa pernyataan bersyarat.

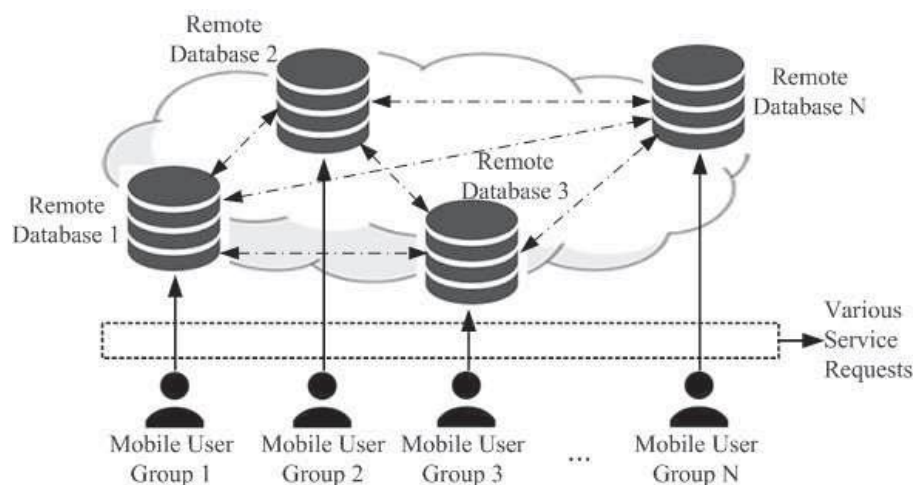
```

if (Processor = "A"){
do function "1";
else if (Processor = "B"){
do function "2";
}
}

```

9.3.3 Penyimpanan Terdistribusi Massal

Mass Distributed Storage (MDS) adalah teknologi baru yang menggabungkan dua teknik, penyimpanan massal cloud dan penyimpanan terdistribusi cloud untuk menerapkan beberapa server penyimpanan serta meningkatkan keandalan data dan kredibilitas infrastruktur. Pendekatan ini tidak hanya dapat mendukung penyimpanan data massal, tetapi juga mencegah data dari berbagai ancaman fisik, seperti bencana situs, karena data disimpan secara terdistribusi. Jaringan nirkabel memungkinkan situs penyimpanan terdistribusi untuk saling terhubung dan berkomunikasi satu sama lain. Memanfaatkan solusi seluler dapat mencapai solusi aman bagi pengguna cloud seluler untuk menyimpan data dalam jumlah besar. Cloud Mass Storage mengacu pada skema yang menyimpan data berukuran besar pengguna di server jarak jauh dan memastikan data dapat dibaca di berbagai antarmuka. Konsep ini berasal dari Mass Storage yang biasanya berarti solusi untuk penggunaan di tempat. Selain itu, Cloud Distributed Storage adalah sarana yang menyimpan data pada beberapa basis data atau node jarak jauh.



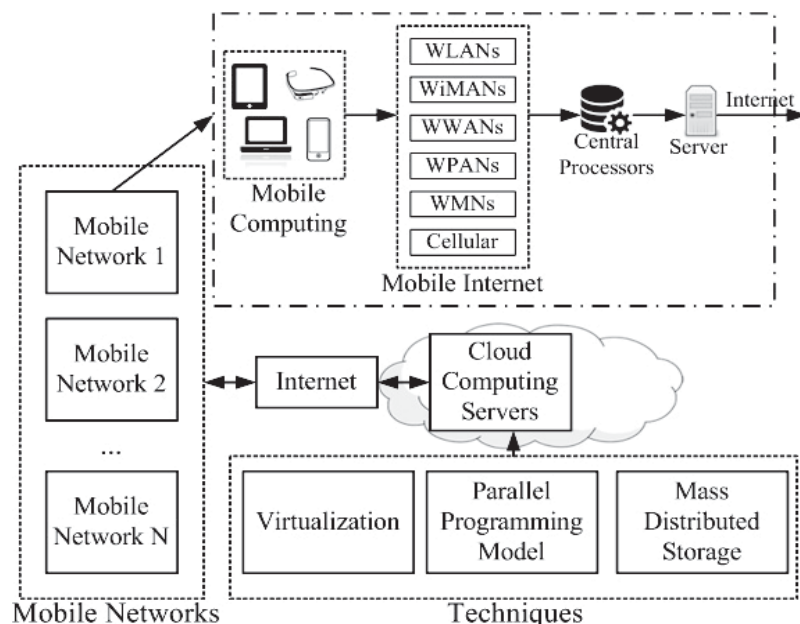
Gambar 9.7 Struktur mengadopsi penyimpanan terdistribusi massal.

Gambar 9.7 menampilkan struktur adopsi teknologi MDS di PKS. Gambar tersebut menjelaskan skenario implementasi yang menampilkan beberapa grup pengguna ponsel yang meminta layanan fungsi tertentu serta beberapa layanan bersama dengan

grup pengguna ponsel lainnya. Di sisi server jarak jauh, beberapa database terdistribusi dikerahkan, dan masing-masing memiliki fokus penawaran layanan tertentu. Pengguna dapat mengakses data target mereka dengan cara yang cepat dan memperoleh data jenis layanan lain melalui interkoneksi dalam jaringan database terhubung yang terdistribusi.

9.4 ARSITEKTUR KOMPUTASI CLOUD SELULER

Arsitektur Komputasi Awan Seluler adalah struktur perancangan, pengembangan, dan implementasi solusi komputasi awan bergerak. Arsitektur terdiri dari seperangkat teknologi yang diperkenalkan dan dievaluasi dalam bab ini. Menyebarkan solusi berbasis MCC memerlukan tiga aspek pendukung penting, termasuk komputasi seluler, Internet seluler, dan komputasi awan. Setiap aspek didukung oleh sejumlah teknik inti. Gambar 9.8 mengilustrasikan arsitektur MCC, yang menunjukkan pendekatan penerapan solusi MCC.



Gambar 9.8 Arsitektur komputasi awan bergerak.

Menurut penjelasan gambar tersebut, pengguna mobile cloud memperoleh layanan mobile cloud dengan menggunakan akses ke komputasi mobile. Perangkat seluler dan protokol komunikasi Internet adalah dua prasyarat. Saat ini, pengguna ponsel memiliki sejumlah pilihan jaringan nirkabel yang dapat digunakan untuk berbagai kebutuhan dan peralatan bergerak. Setelah permintaan layanan pengguna diterima, permintaan akan melalui prosesor pusat dan server jaringan seluler untuk mengalokasikan layanan dan terhubung ke server cloud. Tiga teknologi kunci mendukung implementasi komputasi awan, yang melibatkan virtualisasi, model pemrograman paralel, dan penyimpanan terdistribusi massal. Respons layanan akan dikirim kembali ke pengguna ponsel ketika mesin fisik yang berjalan di server jauh menyelesaikan pemrosesan data. Pengguna cloud seluler memperoleh layanan melalui mesin virtual yang menyediakan representasi layanan kepada pengguna cloud.

9.5 LATIHAN

9.5.1 Pertanyaan Mendasar

1. Apa konsep Mobile Cloud Computing (MCC)?
2. Apa dua migrasi beban kerja utama dari penerapan komputasi awan seluler?
3. Jelaskan secara singkat perbedaan antara komputasi awan dan komputasi awan seluler.
4. Sebutkan tiga aspek teknis utama komputasi awan seluler.
5. Apa yang dimaksud dengan Komputasi Seluler? Apa itu protokol komunikasi?
6. Apakah aplikasi sama dengan aplikasi? Jelaskan jawabanmu.
7. Berikan penjelasan dari istilah-istilah berikut: Jaringan Nirkabel, Komunikator, dan Node Jaringan. Jelaskan secara singkat hubungan antara istilah-istilah ini.
8. Apa saja jenis utama jaringan nirkabel?
9. Apakah LAN Nirkabel berbeda dari Wi-Fi? Berikan alasan Anda untuk menjelaskan tanggapan Anda.
10. Apa konsep jaringan seluler? Apa saja komponen utama jaringan seluler? Jelaskan pengertian cluster dalam jaringan selular.
11. Jelaskan secara singkat pengoperasian jaringan seluler.
12. Apa teknik utama untuk solusi cloud seluler saat ini? Daftar mereka, dan berikan deskripsi singkat untuk setiap teknik.
13. Apa itu Mesin Virtual? Apa prinsip operasi mesin virtual?
14. Coba ilustrasikan proses implementasi mesin virtual di cloud seluler.
15. Apakah aman menggunakan mesin virtual? Silakan gunakan beberapa kalimat untuk menjelaskan pendapat Anda.
16. Apa Konsep Pemrograman Paralel? Jelaskan alasan mengapa penyedia cloud perlu memanfaatkan teknologi ini.
17. Apa konsep Mass Distributed Storage? Jelaskan kesempatan ketika penyimpanan terdistribusi massal diperlukan dalam komputasi awan seluler.

9.5.2 Pertanyaan Praktis

1. Bayangkan diri Anda sebagai Chief Information Officer (CIO) di sebuah perusahaan yang fokus pada bisnis perdagangan internasional. Saat ini, perusahaan Anda bermaksud untuk membangun jaringan nirkabel lokal untuk lokasi kantor baru yang terbuka. Proyek ini bertujuan agar setiap karyawan yang bekerja di kantor dapat mengakses Internet dengan jaringan nirkabel. Setidaknya ada tujuh karyawan yang akan bekerja di kantor secara rutin. Sementara itu, beberapa akses tambahan juga diperlukan, karena beberapa pengunjung sering meminta akses Wi-Fi, seperti mitra bisnis, tamu, atau klien. Inilah misi Anda. Anda perlu menulis rencana teknis singkat dan melaporkannya ke CIO Anda. Rencana teknis perlu menjelaskan jenis jaringan nirkabel yang akan digunakan dan alasannya. Perbandingan antara jenis yang berbeda lebih disukai. Siswa harus menggunakan empat sampai lima kalimat untuk secara singkat mewakili poin-poin kunci untuk rencana tersebut.

2. Diskusikan perbedaan antara arsitektur komputasi awan dan arsitektur komputasi awan bergerak. Anda dapat memikirkan pertanyaan ini dalam perspektif teknis, seperti apakah arsitektur menggunakan teknologi serupa. Coba gunakan empat sampai lima kalimat untuk menjelaskan pemahaman Anda.

9.6 DAFTAR ISTILAH

Komputasi Awan Seluler

adalah paradigma teknologi yang menggunakan pendekatan berbasis cloud untuk menyediakan berbagai layanan bagi pengguna seluler dengan teknologi dan perangkat seluler.

Komputasi awan

adalah paradigma komputasi yang menggunakan teknologi berbasis Web untuk menyediakan layanan on-demand yang dapat diskalakan kepada pengguna dengan berbagi atau menawarkan sumber daya komputasi.

Komputasi Seluler

adalah konsep teknis yang menggambarkan seperangkat teknologi untuk menjembatani tautan dan komunikasi antar perangkat seluler.

Perangkat Lunak Seluler

mewakili sekelompok aplikasi yang dirancang untuk tujuan penggunaan seluler yang berjalan pada perangkat seluler untuk representasi layanan.

Aplikasi Seluler

adalah sekumpulan program yang diimplementasikan pada perangkat seluler yang biasanya dijalankan dalam suatu kerangka kerja, seperti Android.

Protokol komunikasi

mengacu pada seperangkat aturan yang memastikan semua perangkat seluler yang terlibat dalam komunikasi dapat saling memahami saat data dikirimkan.

Platform Distribusi Aplikasi

adalah antarmuka untuk menyediakan pembelian dan unduhan perangkat lunak, yang dimiliki dan dioperasikan oleh penyedia OS.

Internet Seluler

adalah seperangkat teknologi jaringan canggih yang mengaktifkan interkoneksi antara komunikator melalui jaringan nirkabel yang didukung oleh perangkat lunak seluler.

Node Jaringan

mengacu pada beberapa jenis lokasi pemrosesan yang terjadi di berbagai infrastruktur digital.

Jaringan Area Lokal Nirkabel

adalah metode penyebaran nirkabel untuk tujuan penggunaan area jangkauan kecil melalui koneksi radio frekuensi tinggi.

Stasiun Nirkabel di WLANs

mengacu pada peralatan apa pun yang terhubung ke media nirkabel dalam jaringan nirkabel.

Poin ke poin

adalah metode komunikasi yang mendukung koneksi nirkabel antara node jaringan atau titik akhir.

Peer-to-Peer (P2P)

adalah pendekatan membangun jaringan dengan menghubungkan node jaringan yang merupakan klien jaringan dan server.

Sistem Jaringan Terdesentralisasi

adalah model jaringan terdistribusi di mana setiap node jaringan hanya bekerja pada operasi lokal dan memiliki tanggung jawab yang sama terhadap jaringan.

Sistem Jaringan Terpusat

adalah model jaringan yang terdiri dari node master dan node titik akhir, yang cocok untuk sekelompok kecil pengguna yang bekerja bersama dalam ruang terbatas.

Jaringan Area Luas Nirkabel (WWANS)

juga dikenal sebagai Wirelss WAN atau Wireless broadband, adalah jaringan nirkabel penggunaan geografis yang besar di mana sejumlah besar sel mengirimkan sinyal radio ke perangkat seluler dan perangkat lokal.

Sistem Global untuk Komunikasi Seluler (GSM)

adalah standar global untuk mengoperasikan komunikasi seluler, yang dikembangkan oleh European Telecommunications Standards Institute.

Akses Ganda Divisi Kode (CDMA)

adalah pendekatan lain untuk komunikasi seluler yang dapat mengirimkan beberapa sinyal digital dengan menghubungkan beberapa terminal melalui media transmisi yang sama.

Skema Akses Saluran

adalah mekanisme yang memungkinkan aliran data atau sinyal radio yang berbeda melewati atau berbagi saluran komunikasi yang sama.

Menyebarkan spektrum

adalah pendekatan yang memperluas kapasitas bandwidth dengan memvariasikan frekuensi sinyal yang ditransmisikan.

Virtualisasi

adalah mekanisme yang memvirtualisasikan sumber daya komputasi objek untuk merepresentasikannya dalam cara berbasis layanan oleh berbagai tingkat layanan.

Mesin virtual

adalah teknik meniru yang digunakan untuk mendistribusikan dan memvirtualisasikan beberapa sumber daya komputasi jarak jauh dan menyajikan akuisisi komputasi kepada pengguna akhir dalam mode layanan.

Model Pemrograman Paralel (PPM)

adalah teknologi yang digunakan untuk menyelesaikan tugas bersamaan berdasarkan platform berbasis cloud.

Proses Berinteraksi

juga dikenal sebagai Interaksi Proses, adalah pendekatan yang digunakan untuk meningkatkan efisiensi operasi dengan berinteraksi proses paralel.

Berbagi memori

adalah metode proses interaksi yang mendukung pengiriman data antar program berdasarkan ruang alamat global bersama di mana semua program paralel dapat membaca dan menulis data secara asinkron.

Melewati Pesan

adalah istilah yang menggambarkan solusi pertukaran data antar proses, yang dicapai dengan mengirimkan pesan.

Pesan (dalam Melewati Pesan)

adalah jenis konten komunikasi antara program paralel, yang dapat dibangkitkan ke dalam beberapa bentuk, seperti fungsi, sinyal, dan beberapa jenis paket data.

Dekomposisi Masalah

mengacu pada metodologi yang merumuskan program paralel.

Fungsi Paralelisme

juga dikenal sebagai Task atau Control Parallelism, adalah metode komputasi paralel antara beberapa prosesor yang mendistribusikan proses atau utas antara node komputasi paralel.

Penyimpanan Terdistribusi Massal (MDS)

adalah teknologi baru yang menggabungkan dua teknik, penyimpanan massal dan penyimpanan terdistribusi, untuk menerapkan beberapa server penyimpanan serta meningkatkan keandalan data dan kredibilitas infrastruktur.

Penyimpanan Massal Cloud

mengacu pada skema yang menyimpan data berukuran besar pengguna di server jarak jauh dan memastikan data dapat dibaca di berbagai antarmuka.

Penyimpanan Terdistribusi Cloud

adalah sarana yang menyimpan data pada beberapa basis data jarak jauh atau node.

Arsitektur Komputasi Cloud Seluler

adalah struktur merancang, mengembangkan, dan mengimplementasikan solusi komputasi awan seluler.

BAB 10

SINKRONISASI DATA YANG EFISIEN

PADA PERANGKAT SELULER DALAM BIG DATA

Penyimpanan data besar dalam sistem tertanam seluler merupakan masalah penting dalam meningkatkan kinerja aplikasi seluler. Pada bab sebelumnya, kami memperkenalkan komputasi awan seluler. Tidak diragukan lagi bahwa kita berada di era data besar. Big data adalah istilah populer yang digunakan untuk menggambarkan data masif yang dihasilkan dan tumbuh dengan urutan eksponensial. Data besar membawa banyak manfaat bagi kami, tetapi juga membawa banyak tantangan. Karena jumlah data, semua metode tradisional untuk memproses data dalam database relasional jauh lebih efisien daripada sebelumnya. Selain itu, dengan popularitas perangkat seluler yang cepat, masalah data besar menjadi lebih rumit. Dalam bab ini, kami akan memperkenalkan penyimpanan data besar seluler, dan konten utama bab ini meliputi:

1. Ikhtisar data besar.
2. Data besar seluler.
3. Pemrosesan dan analisis data besar.
4. Penyimpanan data besar seluler.
5. Masalah keamanan dan privasi data besar seluler.
6. Deduplikasi Data

10.1 IKHTISAR DATA BESAR

10.1.1 Memahami Tipe Data

Data besar mencakup data terstruktur, tidak terstruktur, dan semi terstruktur. Data terstruktur adalah data yang berada di bidang tetap dalam file rekaman, dan data struktur sering dikelola oleh Structured Query Language (SQL), yang merupakan bahasa pemrograman yang dibuat untuk mengelola dan meminta data dalam sistem manajemen basis data relasional, seperti seperti SQL Server, MySQL, Oracle Rdb, dan SQLite.

Data tidak terstruktur adalah semua hal yang tidak dapat diklasifikasikan dan dimasukkan ke dalam bentuk yang rapi, seperti gambar, video, audio, data instrumen streaming, halaman web, file pdf, email, dan entri blog. Data tidak terstruktur terutama menunjukkan informasi tidak terstruktur, yang biasanya teks-berat. Meskipun informasi tidak terstruktur mungkin berisi beberapa data yang terbentuk, seperti angka, tanggal, dan fakta, itu tidak dapat diproses dengan mudah sebagai informasi terstruktur. Ketidakteraturan dan ambiguitas data tidak terstruktur membuatnya sangat sulit untuk diproses menggunakan program tradisional, dibandingkan dengan data terstruktur dalam database relasional.

Data semi terstruktur adalah bentuk data terstruktur yang tidak sesuai dengan struktur formal model data yang terkait dengan database relasional atau bentuk tabel data

lainnya, tetapi tetap mengandung tag atau penanda lain untuk memisahkan elemen semantik dan menegaskan hierarki. catatan dan bidang dalam data. Dua tipe utama dari data semistruktur adalah Extensible Markup Language (XML) dan JavaScript Object Notation (JSON). Data besar sama pentingnya bagi bisnis, masyarakat, komunitas industri, dan akademisi seperti halnya Internet. Lebih banyak data mengarah pada analisis yang lebih akurat, yang mengarah pada pengambilan keputusan yang lebih percaya diri untuk efisiensi operasional yang lebih besar, pengurangan biaya, dan pengurangan risiko. Namun, sebelum menikmati manfaat yang dibawa oleh data besar, kita perlu memecahkan beberapa masalah teknik.

Tantangan terbesar dari data besar adalah jumlah data yang sangat besar. Selanjutnya jumlahnya masih meningkat pesat. Dunia Komputer menyatakan bahwa informasi yang tidak terstruktur mungkin mencakup lebih dari 70% - 80% dari semua data dalam suatu organisasi.

PETUNJUK: Zettabyte adalah kelipatan dari unit byte untuk informasi digital, dan simbolnya adalah ZB. Zebibyte (ZiB) adalah unit terkait dengan zettabyte. $1\text{ZB} = 1000^7\text{byte}$; $1\text{ZiB} = 240\text{GB} = 270\text{B}$.

10.1.2 Mengkategorikan Model Big Data

Big data memiliki beberapa karakteristik sebagai berikut:

- Volume mengacu pada sejumlah besar data yang dihasilkan setiap detik. Dalam kehidupan sehari-hari, kita memproduksi dan berbagi data setiap detik, seperti email, pesan, foto, video, dan audio. Di Facebook, kami mengirim 10 miliar pesan per hari, mengklik tombol "suka" 4,5 miliar kali, dan mengunggah 350 juta gambar baru setiap hari. Terlebih lagi, dengan pesatnya perkembangan perangkat seluler dan Internet of Things (IoT), server data sensor besar juga membanjiri.
- Varietas mengacu pada jenis data. Di masa lalu, kami berfokus pada data terstruktur yang masuk dengan rapi ke dalam database relasional. Seperti disebutkan di atas, data tidak terstruktur tidak dapat diproses dengan metode tradisional. Selain itu, 80% dari data saat ini tidak terstruktur.
- Teknologi data besar dapat membantu kita memanfaatkan berbagai jenis data untuk beberapa persyaratan tertentu.
- Velocity mengacu pada kecepatan di mana data baru dihasilkan dan kecepatan di mana data diproses dan ditransmisikan untuk memenuhi tuntutan. Misalnya, kecepatan pemeriksaan transaksi kartu kredit harus dalam milidetik. Teknologi big data memungkinkan kita untuk menganalisis data saat sedang dibuat tanpa memasukkannya ke dalam database.
- Variabilitas mengacu pada inkonsistensi yang dapat ditunjukkan oleh data pada suatu waktu, sehingga menghambat proses untuk dapat menangani dan mengelola data secara efektif.
- Veracity mengacu pada kekacauan atau kepercayaan data. Akurasi analisis tergantung pada kebenaran sumber data. Dengan banyak bentuk data besar, kualitas, dan akurasi kurang dapat dikendalikan dari sebelumnya, tetapi data besar

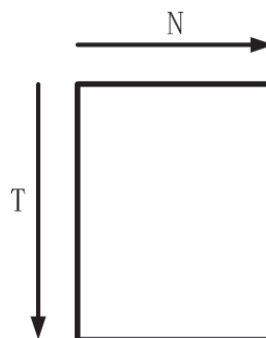
dan teknologi analitik dapat membantu kami bekerja dengan jenis data ini. Selain itu, volume dapat digunakan untuk menutupi kekurangan kualitas atau akurasi.

- Kompleksitas mengacu pada kesulitan pengelolaan data. Pengelolaan data menjadi proses yang sangat kompleks, terutama ketika volume data yang besar berasal dari berbagai sumber. Untuk menyembunyikan informasi pada data ini, mereka perlu ditautkan dan dikorelasikan, yang membentuk “kompleksitas” data besar.

Kita dapat membentuk data sebagai tabel dengan N kolom dan T baris, seperti yang ditunjukkan pada Gambar 10.1. N menunjukkan dimensi setiap vektor, dan T menunjukkan jumlah sampel pelatihan. Berdasarkan tabel ini, big data adalah dengan T besar, atau N besar, atau keduanya. Data besar dapat kita pisahkan menjadi tiga model, yaitu T besar dengan N kecil, T tak hingga dengan N kecil, dan T kecil dengan N besar.

10.1.3 Tantangan Saat Ini dalam Big Data

Menghadapi data yang begitu besar, setiap metode tradisional jauh lebih tidak efisien dibandingkan sebelumnya. Tantangan big data meliputi analisis, pengambilan, kurasi, pencarian, berbagi, penyimpanan, transfer, visualisasi, dan privasi. Kumpulan data tumbuh dalam ukuran sebagian karena semakin banyak dikumpulkan oleh perangkat seluler. Ada 4,6 miliar langganan telepon seluler di seluruh dunia dan antara 1 miliar hingga 2 miliar orang mengakses Internet. Selain ponsel, ribuan jenis sensor seluler menghasilkan dan menggunakan data dalam jumlah besar setiap detik, terutama di Smart City.



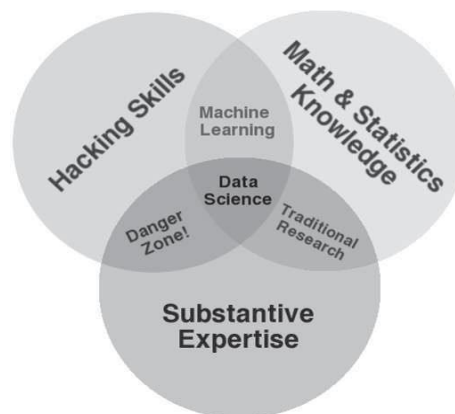
Gambar 10.1 Tabel pemodelan data besar.

Dengan miliaran perangkat seluler di dunia saat ini, data besar seluler menjadi tantangan paling signifikan yang harus dipecahkan. Maraknya big data menuntut kita untuk dapat mengakses sumber daya data kapan saja dan di mana saja tentang segala hal sehari-hari. Selain itu, jenis data ini sangat berharga dan menguntungkan jika digunakan dengan baik. Namun, agar data besar dapat dianalisis, beberapa tantangan harus diatasi. Lebih khusus lagi, alih-alih dibatasi untuk satu komputer, aplikasi di mana-mana harus dapat dijalankan pada ekosistem perangkat jaringan, yang masing-masing dapat bergabung atau meninggalkan ruang bersama di mana-mana setiap saat. Selain itu, ada tugas analitik yang terlalu mahal secara komputasi untuk dilakukan pada ekosistem perangkat seluler. Juga, bagaimana kita dapat memanfaatkan kemampuan

spesifik dari setiap perangkat, termasuk ukuran tampilan yang bervariasi, modalitas input, dan sumber daya komputasi? Dalam bab ini, kami terutama memperkenalkan penyimpanan data besar seluler.

10.2 PEMROSESAN DATA BESAR

Sebelum menyimpan big data, kita perlu melakukan preprocessing terhadap data tersebut. Metode yang berskala ke data besar sangat menarik dalam ilmu data, meskipun disiplin umumnya tidak dianggap terbatas pada data tersebut. Ilmu data adalah ekstraksi pengetahuan dari data. Ini menggunakan teknik dan teori yang diambil dari banyak bidang dalam bidang matematika, statistik, teori informasi, dan teknologi informasi yang luas, termasuk pemrosesan sinyal, model probabilitas, pembelajaran mesin, pembelajaran statistik, pemrograman komputer, rekayasa data, pengenalan pola dan pembelajaran. Visualisasi, analitik prediktif, pemodelan ketidakpastian, pergudangan data, kompresi data, dan komputasi kinerja tinggi. Ilmu data muncul untuk memenuhi tantangan pemrosesan data besar.



Gambar 10.2 Diagram venn ilmu data.

Gambar 10.2 adalah Diagram Venn Ilmu Data yang disajikan oleh Drew Conway. Warna utama data adalah keterampilan meretas, pengetahuan matematika dan statistik, dan keahlian substantif. Masing-masing keterampilan ini sangat berharga, tetapi ketika dikombinasikan dengan hanya satu keterampilan lainnya, itu adalah ilmu data terbaik, atau paling buruk benar-benar berbahaya. Karena peretas adalah salah satu disiplin teknologi informasi yang paling terampil, peretas membutuhkan pengetahuan luas tentang teknologi dan teknik TI. Dikombinasikan dengan pengetahuan matematika dan statistik, keterampilan meretas dapat digunakan untuk pembelajaran mesin. Namun, beberapa ahli substantif dapat menggunakan keterampilan peretasan untuk melakukan beberapa hal ilegal, dan itu adalah zona bahaya. Akhirnya, keterampilan meretas, pengetahuan matematika dan statistik, dan keahlian substantif membentuk ilmu data.

10.2.1 Pembelajaran Mesin

Perkembangan pembelajaran mesin, cabang kecerdasan buatan yang digunakan untuk mengungkap pola dalam data dari mana model prediktif dapat dikembangkan, telah meningkatkan pertumbuhan dan pentingnya ilmu data. Aplikasi seluler tradisional

tidak fokus pada pembelajaran mesin karena pembatasan yang disebabkan oleh keterbatasan kapasitas perangkat seluler. Namun, menggunakan komputasi awan seluler telah memungkinkan aplikasi seluler memanfaatkan teknik pembelajaran mesin. Beban kerja yang berat dapat dipindahkan ke sisi cloud di mana perangkat seluler dapat menghemat energi dan mencapai kinerja tinggi. Selain itu, pembelajaran mesin biasanya diklasifikasikan ke dalam tiga kategori besar: pembelajaran yang diawasi, pembelajaran tanpa pengawasan, dan pembelajaran semi-terawasi. Kami memperkenalkan ketiga jenis pembelajaran mesin ini secara rinci,

Pembelajaran dengan Pengawasan

Dalam pembelajaran yang diawasi, data pelatihan selalu dengan fitur dan label. Algoritme pembelajaran yang diawasi menganalisis data pelatihan dan menghasilkan fungsi yang disimpulkan. Tujuan pembelajaran terawasi adalah prediksi, peringkat, dan klasifikasi. Naive Bayes Naive Bayes secara mengejutkan banyak digunakan dalam pembelajaran terawasi. Pengklasifikasi Navie Bayes didasarkan pada penerapan teorema Bayes dengan asumsi independensi naif antara fitur.

$$\text{classify}(f_1, \dots, f_n) = \arg \max p(C = c) \prod_{i=1}^n p(F_i = f_i | C = c) \quad (10.1)$$

Kami biasanya memperkirakan $P(f_i|c)$ menggunakan perkiraan :

$$P(f_i|c) = (n_c + m_p)/(n + m) \quad (10.2)$$

Dimana: n = jumlah contoh pelatihan dimana $C = c$; n_c = jumlah contoh yang $C = c$ dan $F_i = f_i$; p = perkiraan apriori untuk $P(f_i|c)$, dan m = ukuran sampel yang setara. Misalnya, kita akan menggunakan metode Naive Bayes untuk menentukan apakah sebuah SUV domestik berwarna merah dicuri. Dalam Teorema Bayes, $P(A|B) = P(B|A) P(A)/P(B)$, dan jika semua kejadian saling bebas, pasti benar bahwa: $P(A, B, C|D) = P(A|D) P(B|D) P(C|D)$. Dalam contoh ini, kita dapat mengetahui bahwa: $P(\text{Yes}|\text{merah, domestik, SUV}) = P(\text{merah, domestik, SUV} | \text{Yes}) P(\text{Yes})/P(\text{merah, domestik, SUV})$, $P(\text{merah, dom, SUV} | \text{Yes}) = P(\text{merah}|\text{Yes}) * P(\text{dom}|\text{Yes}) * P(\text{SUV} | \text{Yes})$, dan $P(\text{merah, dom, SUV} | \text{Tidak}) = P(\text{merah} | \text{Tidak}) * P(\text{dom} | \text{Tidak}) P(\text{SUV} | \text{Tidak})$. Kami memiliki sepuluh contoh, seperti yang ditunjukkan pada Gambar 10.3, di mana kita bisa mendapatkan nilai n , n_c , p , dan m , seperti yang ditunjukkan pada Tabel 10.1. Perhatikan bahwa tidak ada contoh SUV Domestik Merah di kumpulan data kami.

Tabel 10.1 Nilai Dasar

	Yes	No
Red	$n = 5$ $n_c = 3$ $p = 0.5$ $m = 3$	$n = 5$ $n_c = 2$ $p = 0.5$ $m = 3$
SUV	$n = 5$ $n_c = 1$ $p = 0.5$	$n = 5$ $n_c = 3$ $p = 0.5$

	$m = 3$	$m = 3$
Domestic	$n = 5$ $n_c = 2$ $p = 0.5$ $m = 3$	$n = 5$ $n_c = 3$ $p = 0.5$ $m = 3$

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

Gambar 10.3 Contoh tabel untuk Bayes.

Dari Gambar 10.3, kita dapat menghitung bahwa:

$$P(\text{Red}|\text{Yes}) = (3 + 3 * 0.5)/(5 + 3) = 0.5625,$$

$$P(\text{Red}|\text{No}) = (2 + 3 * 0.5)/(5 + 3) = 0.4375,$$

$$P(\text{SUV}|\text{Yes}) = (1 + 3 * 0.5)/(5 + 3) = 0.3125,$$

$$P(\text{SUV}|\text{No}) = (3 + 3 * 0.5)/(5 + 3) = 0.5625,$$

$$P(\text{Domestic}|\text{Yes}) = (2 + 3 * 0.5)/(5 + 3) = 0.4375,$$

$$P(\text{Domestic}|\text{No}) = (3 + 3 * 0.5)/(5 + 3) = 0.5625.$$

Kemudian kita dapat menghitung peluang sebuah SUV domestik berwarna merah dicuri: $P(\text{Yes}) P(\text{Merah}|\text{Yes}) P(\text{SUV}|\text{Yes}) P(\text{Domestik}|\text{Yes}) = 0,5 * 0,5625 * 0,3125 * 0,4375 = 0,03845$, dan peluang sebuah SUV domestik berwarna merah tidak dicuri: $P(\text{Tidak}) P(\text{Merah}|\text{Tidak}) P(\text{SUV}|\text{Tidak}) P(\text{Domestik}|\text{Tidak}) = 0,5 * 0,4375 * 0,5625 * 0,5625 = 0,06921$. Karena $0,06921 > 0,03845$, SUV domestik merah tidak dicuri.

Pohon Keputusan

Pembelajaran pohon keputusan merupakan metode yang umum digunakan dalam data mining. Tujuannya adalah untuk membuat model yang memprediksi nilai variabel target berdasarkan beberapa variabel input. Sebuah pohon dapat "dipelajari" dengan membagi set sumber menjadi subset berdasarkan tes nilai atribut. Proses ini diulang pada setiap subset yang diturunkan secara rekursif yang disebut partisi rekursif. Rekursi selesai ketika subset pada sebuah node memiliki semua nilai yang sama dari variabel target, atau ketika pemisahan tidak lagi menambah nilai prediksi. Proses Top-Down Induction of Decision Trees (TDIDT) ini adalah contoh dari algoritma serakah,

dan sejauh ini merupakan strategi paling umum untuk mempelajari pohon keputusan dari data.

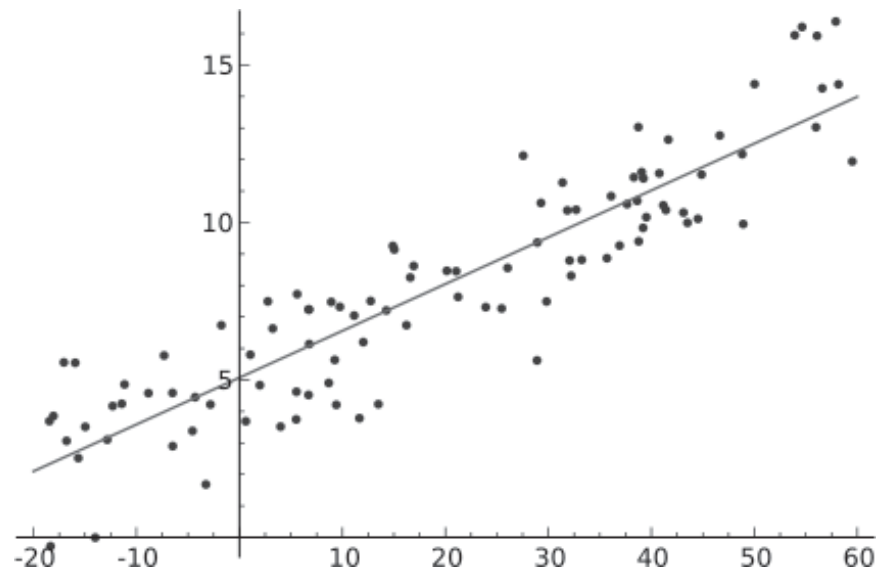
Regresi

Regresi linier adalah pendekatan untuk memodelkan hubungan antara variabel dependen skalar y dan satu atau lebih variabel penjelas yang dilambangkan X . Seperti yang ditunjukkan pada Gambar 10.4, dalam regresi linier, data dimodelkan menggunakan fungsi prediktor linier, dan model yang tidak diketahui parameter-nya diperkirakan dari data.

Hubungan statistik antara istilah kesalahan dan regresi memainkan peran penting dalam menentukan apakah prosedur estimasi memiliki sifat pengambilan sampel yang diinginkan, seperti tidak bias dan konsisten. Selanjutnya, pengaturan, atau distribusi probabilitas dari variabel prediktor x , memiliki pengaruh besar pada ketepatan estimasi. Pengambilan sampel dan desain eksperimen adalah subbidang statistik yang sangat berkembang yang memberikan panduan untuk mengumpulkan data sedemikian rupa untuk mencapai perkiraan yang tepat.

Pohon Keputusan yang Didorong Gradien

Boosting adalah meta-algoritma ansambel pembelajaran mesin untuk mengurangi bias terutama dan juga varians dalam pembelajaran yang diawasi. Gradient boosting merupakan teknik machine learning untuk masalah regresi, yang menghasilkan model prediksi berupa ensemble model prediksi lemah. Meskipun boosting tidak dibatasi secara algoritme, sebagian besar algoritma boosting terdiri dari mempelajari pengklasifikasi lemah secara iteratif sehubungan dengan distribusi dan menambahkannya ke pengklasifikasi kuat akhir. Ketika mereka ditambahkan, mereka biasanya dibobot dalam beberapa cara yang biasanya terkait dengan akurasi pelajar yang lemah. Setelah pelajar yang lemah ditambahkan, data dibobot ulang: contoh yang salah diklasifikasikan menambah berat badan, dan contoh yang diklasifikasikan dengan benar menurunkan berat badan. Dengan demikian, pembelajar yang lemah di masa depan lebih fokus pada contoh-contoh yang salah diklasifikasikan oleh pembelajar lemah sebelumnya.



Gambar 10.4 Analisis regresi.

Pembelajaran Tanpa Pengawasan

Tugas belajar tanpa pengawasan selalu dengan fitur tetapi tanpa label. Karena contoh yang diberikan kepada pelajar tidak berlabel, tidak ada kesalahan atau sinyal penghargaan untuk mengevaluasi solusi potensial. Ini membedakan pembelajaran tanpa pengawasan dari pembelajaran terawasi dan pembelajaran penguatan. Tujuan pembelajaran tanpa pengawasan adalah pengelompokan, rekayasa/penemuan fitur, dan pengurangan dimensi.

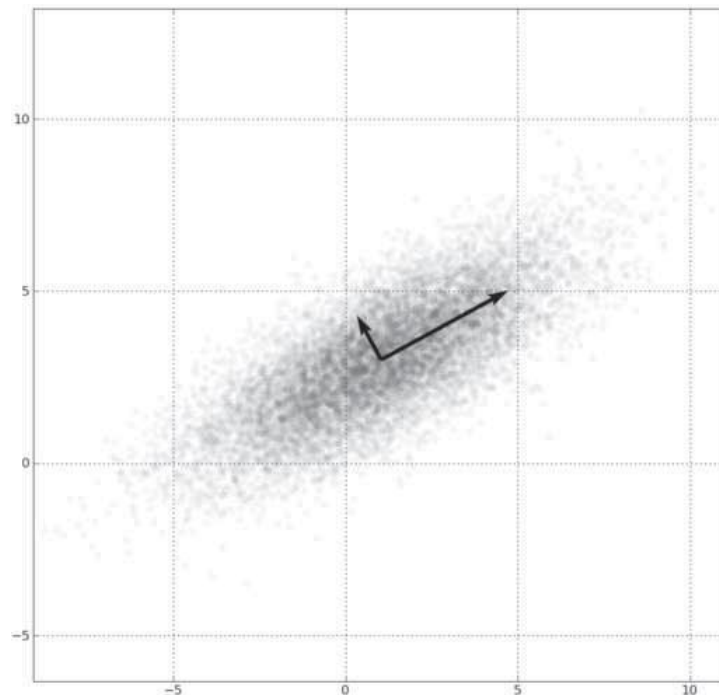
Analisis Komponen Prinsip

Analisis komponen utama (PCA) adalah prosedur statistik yang menggunakan transformasi ortogonal untuk mengubah sekumpulan pengamatan dari variabel yang mungkin berkorelasi menjadi sekumpulan nilai variabel yang tidak berkorelasi linier yang disebut komponen utama. PCA adalah analisis multivariat berbasis vektor eigen yang paling sederhana. Operasinya dapat dianggap sebagai mengungkapkan struktur internal data dengan cara yang paling baik menjelaskan varians dalam data. Jika dataset multivariat divisualisasikan sebagai satu set koordinat dalam ruang data dimensi tinggi, PCA dapat menyediakan pengguna dengan gambar dimensi yang lebih rendah, proyeksi atau "bayangan" dari objek ini bila dilihat dari sudut pandang yang paling informatif. Ini dilakukan dengan hanya menggunakan beberapa komponen utama pertama sehingga dimensi dari data yang ditransformasikan berkurang. Gambar 10.5 adalah PCA dari distribusi Gaussian multivariat yang berpusat pada (1,3) dengan standar deviasi 3 pada kira-kira (0,878, 0,478) arah dan 1 pada arah ortogonal. Vektor yang ditampilkan adalah vektor eigen dari matriks kovarians yang diskalakan oleh akar kuadrat dari nilai eigen yang sesuai, dan digeser sehingga ekornya berada di mean.

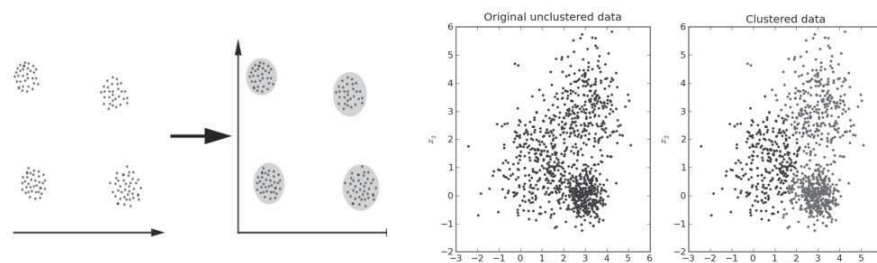
Kekelompokan

Clustering adalah tugas mengelompokkan satu set objek sedemikian rupa sehingga objek dalam kelompok yang sama lebih mirip satu sama lain daripada yang ada di kelompok lain. Ini adalah tugas utama penggalian data eksplorasi, dan teknik umum

untuk analisis data statistik, yang digunakan di banyak bidang, termasuk pembelajaran mesin, pengenalan pola, analisis gambar, pengambilan informasi, dan bioinformatika.



Gambar 10.5 PCA dari distribusi Gaussian multivariat.



Gambar 10.6 Dua contoh pengelompokan.

Dua gambar kiri dari Gambar 10.6 adalah kemajuan yang mudah dari pengelompokan, sedangkan dua gambar kanan adalah yang sulit. Selanjutnya, untuk Big Data, tantangannya akan jauh lebih rumit. Ada terlalu banyak contoh dan fitur dalam data besar, yang membuat pemrosesan pengelompokan menjadi sangat lambat. Selanjutnya, karena berbagai perangkat yang menghasilkan data, kita harus mempertimbangkan toleransi kesalahan perangkat keras untuk label data. Akibatnya, bagaimana memproses data besar secara efisien adalah tujuan utama dari data besar seluler, yang akan diperkenalkan di bagian selanjutnya.

Pembelajaran Semisupervised

Pembelajaran semi terawasi adalah kelas tugas dan teknik pembelajaran terawasi yang juga menggunakan data tak berlabel untuk pelatihan, biasanya sejumlah kecil data berlabel dengan sejumlah besar data tak berlabel. Pembelajaran semi-diawasi berada di antara pembelajaran tanpa pengawasan dan pembelajaran terawasi. Tugas belajar

semi-diawasi hanya dengan label positif. Data yang tidak berlabel dapat menghasilkan peningkatan yang cukup besar dalam akurasi pembelajaran, bila digunakan bersama dengan sejumlah kecil data berlabel.

10.3 PENYIMPANAN DATA BESAR SELULER

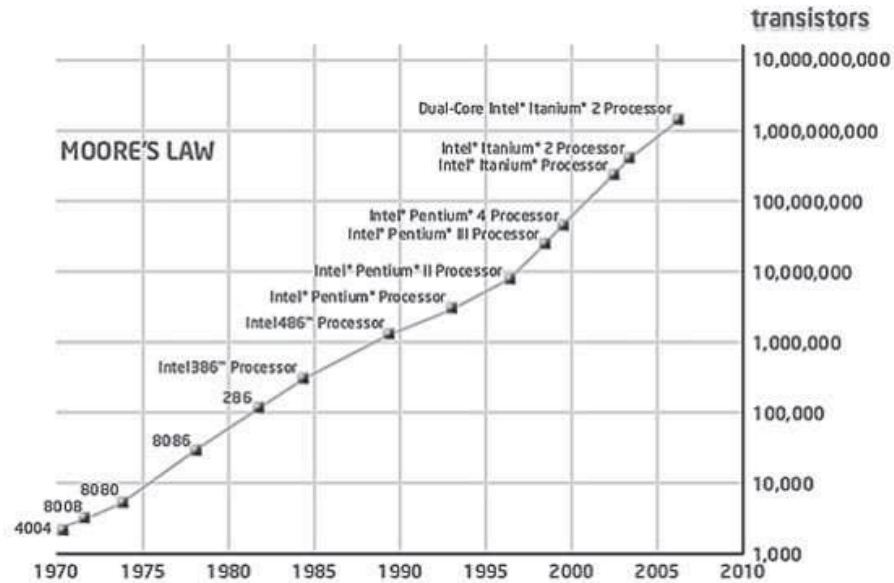
Setelah melakukan preprocessing big data, kita perlu menentukan bagaimana cara menyimpan dan menggunakan data tersebut. Biasanya, kami memiliki beberapa pilihan untuk menangani data ini, seperti menyimpan semuanya ke server, menyimpan sebagian ke server dan meninggalkan yang lain di perangkat, dan menyimpan semuanya ke perangkat. Karena keterbatasan waktu dan ruang, kami dapat menyimpan data terbaru di perangkat, dan menyimpan lainnya ke server jarak jauh. Di bagian ini, kami terutama berfokus pada penyimpanan data besar seluler.

10.3.1 Pengenalan dan Konsep Dasar

Gambar 10.7 adalah "Hukum Moore" yang terkenal, yang mengamati bahwa jumlah transistor dalam sirkuit terpadu padat berlipat ganda kira-kira setiap 18 bulan. Saat ini perkembangan Central Processing Unit (CPU) jauh lebih cepat dibandingkan dengan memori yang membentuk dinding memori. Kecepatan CPU meningkat pada tingkat tahunan sebesar 55%, sementara kecepatan memori hanya meningkat 10%. Latensi memori menjadi hambatan besar dalam kinerja komputer. Menawarkan lebih banyak cara untuk memenuhi kebutuhan memori menjadi semakin penting untuk meningkatkan kinerja dan mengurangi biaya.

Saat ini, sistem heterogen dengan memori hibrida menjadi tren tidak hanya untuk sistem tertanam, tetapi juga aplikasi cloud seluler. Sistem heterogen adalah platform yang memiliki beberapa elemen pemrosesan yang heterogen, seperti hierarki memori dan Input/Output (I/O). Ada dua jenis memori, yaitu memori volatil dan memori non-volatil. Memori volatil adalah memori tradisional termasuk Static Random-Access Memory (SRAM) dan Dynamic Random-Access Memory (DRAM). Sementara itu, memori non-volatile merupakan revolusi teknologi memori dan terdiri dari Phase Change Memory (PCM), Magnetoresistive RAM (MRAM), dan Flash Memory. Memori non-volatil adalah salah satu jenis memori komputer yang dapat memperoleh kembali informasi yang tersimpan meskipun komputer tidak dalam keadaan mati.

Tabel 10.2 menunjukkan perbedaan antara SRAM dan DRAM, dan Tabel 10.3 menunjukkan perbedaan antara MRAM, PCM, dan Memori Flash. SRAM memiliki beberapa kelebihan, seperti biaya baca dan tulis yang rendah, tetapi memiliki beberapa kekurangan, seperti daya bocor yang tinggi. Sementara itu, DRAM memiliki beberapa kelebihan, termasuk kepadatan tinggi dan harga rendah, tetapi memiliki beberapa kekurangan seperti biaya baca dan tulis yang tinggi. MRAM memiliki biaya baca yang rendah, kebocoran yang rendah, dan kepadatan yang tinggi, tetapi biaya penulisan yang tinggi, termasuk energi dan latensi. PCM memiliki kepadatan dan kapasitas memori yang tinggi, sementara itu memiliki biaya tulis yang tinggi dan daya tahan yang terbatas, dari 10^8 hingga 10^9 . Memori Flash memiliki daya tahan tinggi tetapi daya tahannya sangat terbatas, sekitar 10^5 .



Gambar 10.7 Hukum Moore. Diperoleh dari <http://archive.isgtw.org/images/mooreslaw.jpg>.

Tabel 10.2 Perbedaan antara SRAM dan DRAM

	Biaya R/W	Kepadatan	Harga
SRAM	Rendah	Rendah	Tinggi
DRAM	Tinggi	Tinggi	Rendah

Tabel 10.3 Perbedaan antara MRAM, PCM, dan Memori Flash

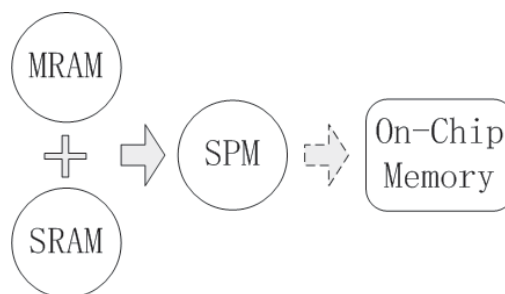
	Biaya R/W	Kepadatan	Kapasitas	Daya tahan	Daya tahan
MRAM	Baca Rendah,	Tinggi	Tinggi	Tinggi	Tak terbatas
PCM	Tulis Tinggi	Tinggi	Tinggi	Terbatas	Terbatas
Flash Memory	Tulis Tinggi	Tinggi	Tinggi	Tinggi	Sangat

10.3.2 Arsitektur Memori Heterogen

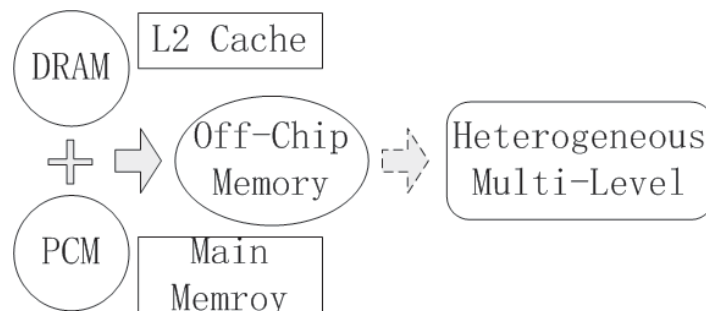
Untuk mencapai kinerja tinggi dan mengurangi biaya, kami dapat menggabungkan berbagai jenis memori, dan itu membentuk arsitektur memori yang heterogen. Seperti yang ditunjukkan pada Gambar 10.8, kita dapat menggabungkan MRAM dan SRAM dan membentuk Scratchpad Memory (SPM), yang dapat dianggap sebagai memori on-chip. SPM adalah memori internal berkecepatan tinggi yang digunakan untuk penyimpanan sementara perhitungan, data, dan pekerjaan lain yang sedang berlangsung. Ini dapat dianggap mirip dengan cache L1 dalam hal itu adalah memori lemari berikutnya ke unit logika aritmatika setelah register internal, dengan instruksi eksplisit untuk mengirimkan data ke dan dari memori utama. Selanjutnya, kita dapat menggabungkan DRAM dan PCM dan membentuk memori off-chip. Dalam arsitektur memori heterogen

ini, DRAM dianggap sebagai cache L2, sedangkan PCM dianggap sebagai memori utama.

Memori on-chip mengacu pada memori yang secara fisik ada pada mikrokontroler itu sendiri. Sedangkan memori off-chip mengacu pada memori eksternal di luar mikrokontroler. Memori on-chip dan memori off-chip merupakan memori multi-level yang heterogen. Gambar 10.10 adalah arsitektur untuk Chip Multiprocessor (CMP). CMP adalah komponen komputasi tunggal dengan dua atau lebih unit pemrosesan aktual yang independen, yang merupakan unit yang membaca dan mengeksekusi instruksi program. Bagian atas, dikelilingi oleh garis dashed, adalah memori on-chip, dan persegi kecil putus-putus adalah mikroprosesor.



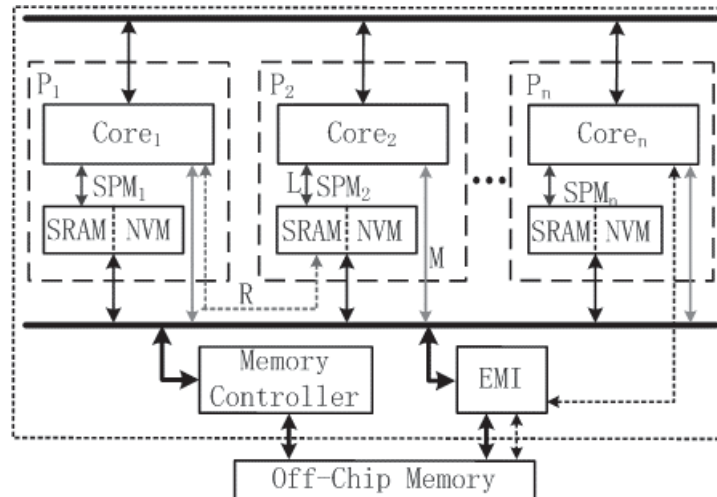
Gambar 10.8 Menggabungkan MRAM dan SRAM ke dalam memori on-chip.



Gambar 10.9 Menggabungkan DRAM dan PCM ke dalam memori off-chip.

Garis biru pada memori on-chip menunjukkan akses lokal. Garis merah menunjukkan akses jarak jauh, dan garis hijau menunjukkan akses memori utama. Dalam praktiknya, sistem CMP dengan SPM heterogen banyak digunakan dalam sistem tertanam, terutama di perangkat seluler. Dalam sistem CMP, menempatkan data yang sama ke dalam unit memori yang berbeda memiliki biaya yang berbeda, yang dapat didefinisikan sebagai daya dan latensi. Daya sebanding dengan latensi. Selanjutnya, ada berbagai macam tugas, termasuk granularity halus dan granularity kasar. Tugas yang berbeda menggunakan jenis data yang berbeda dan pada frekuensi yang berbeda, yang membuat pemrosesan dan penyimpanan data besar menjadi lebih rumit daripada data tradisional.

Untuk meningkatkan kinerja dan mengurangi biaya, kita perlu menentukan bagaimana mengalokasikan data ke unit memori yang berbeda.



Gambar 10.10 Arsitektur SPM heterogen untuk CMP.

10.3.3 Alokasi Data Pemrograman Dinamis Multi dimensi

Pada bagian ini, kami memperkenalkan arsitektur memori on-chip SPM hybrid yang menggabungkan SRAM dan MRAM untuk meningkatkan kinerja keseluruhan sistem memori, seperti yang disebutkan dalam Gambar 10.8. Masalah yang menantang adalah bahwa arsitektur SPM hybrid harus menyelesaikan cara mengurangi konsumsi energi, latensi akses memori, dan jumlah operasi tulis ke MRAM. Untuk memanfaatkan manfaat dari setiap jenis memori, kita harus mengalokasikan data secara strategis pada setiap modul memori sehingga total biaya akses memori dapat diminimalkan. Ingatlah bahwa SPM adalah perangkat lunak yang dapat dikontrol, yang berarti datum di dalamnya dapat dikelola oleh pemrogram atau kompilar.

Misalnya, ada tujuh buah data, yaitu A, B, C, D, E, F, dan G. Pada data tersebut, hanya G yang merupakan SRAM, sedangkan yang lainnya adalah memori utama. Berdasarkan referensi, kami mengasumsikan biaya akses memori dan biaya pemindahan ditunjukkan pada Tabel 10.4 untuk tujuan demonstrasi. Secara umum, kita dapat menggunakan CACTI (dikembangkan oleh HP) untuk memodelkan cache dan waktu akses memori, waktu siklus, area, kebocoran, dan daya dinamis. CACTI adalah cache terintegrasi dan waktu akses memori, waktu siklus, area, kebocoran, dan model daya dinamis. Pada Tabel 10.4, baris “Pindah” menunjukkan biaya pemindahan data dari satu memori ke memori lainnya. Misalnya, 7 mewakili biaya pemindahan data dari SRAM ke MRAM.

Tabel 10.4 Biaya untuk Operasi Memori yang Berbeda

Operation		SRAM	MRAM	Main
Read		2	4	80
Write		2	10	80
Move	SRAM	0	7	70
	MRAM	5	0	70
	Main	65	80	0

Tabel 10.5 Jumlah Akses Memori

Data	Reads	Writes
A	7	1
B	6	2
C	5	3
D	4	4
E	3	5
F	2	6
G	1	7

Biaya total keseluruhan sistem adalah penjumlahan dari biaya baca, tulis, dan pindah, dan dapat dirumuskan sebagai:

$$Total_{cost} = \sum_{i=0}^n (\#R_i \times \$R_i + \#W_i \times \$W_i + \$M_i). \quad (10.3)$$

Dalam Persamaan 10.3, #R menunjukkan jumlah operasi baca, dan #W menunjukkan jumlah operasi tulis. \$R menunjukkan biaya operasi baca, dan \$W menunjukkan biaya operasi tulis. \$M menunjukkan biaya pemindahan data. Misalnya, total biaya pengalokasian G ke SRAM adalah: $Cost_{GS} = 1 \cdot 2 + 7 \cdot 2 = 16$. Jumlah pembacaan G adalah 1, dan biaya pembacaan SRAM adalah 2; jadi, total biaya membaca G adalah $1 \cdot 2$. Demikian pula, total biaya penulisan G adalah $7 \cdot 2$. G menggunakan SRAM; dengan demikian, biaya pengalokasian G ke SRAM adalah 0; Total biaya pengalokasian A ke MRAM adalah: $Cost_{AM} = 7 \cdot 4 + 1 \cdot 10 + 80 = 118$. Jumlah pembacaan A adalah 7, dan biaya pembacaan MRAM adalah 4, maka biaya pembacaan A adalah $7 \cdot 4$. Sedangkan total biaya penulisan A adalah $1 \cdot 10$. A ada di memori utama; jadi, biaya pemindahan A dari memori utama ke MRAM adalah 80.

Tabel 10.6 Biaya Mengalokasikan Setiap Data ke Unit Memori yang Berbeda

Data	SRAM	MRAM	Main
A	81	118	640
B	81	124	640
C	81	130	640
D	81	136	640
E	81	142	640
F	81	148	640
G	16	81	710

Menggunakan Persamaan 10.3, kita dapat menghitung biaya pengalokasian data ke unit memori yang berbeda, seperti yang ditunjukkan pada Tabel 10.6. Kemudian kami menetapkan x dan y sebagai jumlah unit MRAM yang tidak tersedia dan unit SRAM yang tidak tersedia. Asumsikan bahwa ada empat blok MRAM dan dua blok SRAM. Gambar 10.11 menunjukkan semua biaya ketika $x = 4$, $y = 0, 1, 2$. Jika $x = 4$, $y = 2$, yang

berarti bahwa semua MRAM dan SRAM tidak dapat digunakan, semua data berada di memori utama:

Total biayanya adalah $4550 = 640 + 640 + 640 + 640 + 640 + 640 + 710$.

Jika $x = 4, y = 1$, artinya semua MRAM tidak dapat digunakan, tetapi satu SRAM dapat digunakan:

Jika kita mengalokasikan A ke SRAM, total biayanya adalah $3991 = 4550 - (640 - 81)$.

Jika kita mengalokasikan B, C, D, E, dan F ke SRAM, total biayanya adalah $3991 = 4550 - (640 - 81)$.

Jika kita mengalokasikan G ke SRAM, total biayanya adalah $3856 = 4550 - (710 - 16)$.

Jika $x = 4, y = 0$, yang berarti semua MRAM tidak dapat digunakan, tetapi dua SRAM dapat digunakan:

Jika mengalokasikan B ke SRAM, dan A sudah dialokasikan ke SRAM, total biayanya adalah $3432 = 3991 - (640 - 81)$.

Jika kita mengalokasikan G ke SRAM, total biayanya adalah $3297 = 3856 - (640 - 81)$.

x = 4							
y	A	B	C	D	E	F	G
0		3432	3432	3432	3432	3432	3297
1	3991	3991	3991	3991	3991	3991	3856
2	4550						

Gambar 10.11 Langkah pertama pemrograman dinamis.

Gambar 10.12 menunjukkan semua biaya ketika $x = 3, y = 0, 1, 2$. Jika $x = 3, y = 2$, yang berarti bahwa satu ruang MRAM dapat digunakan:

- Jika mengalokasikan A ke MRAM, total biayanya adalah: $4028 = 4550 - (640 - 118)$.
- Jika mengalokasikan B ke MRAM, total biayanya adalah: $4034 = 4550 - (640 - 124)$; Namun, biaya ini lebih besar daripada mengalokasikan A ke MRAM, jadi kami tetap mengalokasikan A ke MRAM, ketika total biaya: $4028 = 4550 - (640 - 118)$.
- Jika mengalokasikan C ke MRAM, total biayanya adalah: $4040 = 4550 - (640 - 130)$; Namun, biaya ini lebih besar daripada mengalokasikan A ke MRAM, jadi kami tetap mengalokasikan A ke MRAM, ketika total biaya: $4028 = 4550 - (640 - 118)$.
- Jika mengalokasikan D ke MRAM, total biayanya adalah: $4046 = 4550 - (640 - 136)$; Namun, biaya ini lebih besar daripada mengalokasikan A ke MRAM, jadi kami tetap mengalokasikan A ke MRAM, ketika total biaya: $4028 = 4550 - (640 - 118)$.
- Jika mengalokasikan E ke MRAM, total biayanya adalah: $4052 = 4550 - (640 - 142)$; Namun, biaya ini lebih besar daripada mengalokasikan A ke MRAM, jadi kami tetap mengalokasikan A ke MRAM, ketika total biaya: $4028 = 4550 - (640 - 118)$.

- Jika mengalokasikan F ke MRAM, total biayanya adalah: $4058 = 4550 - (640 - 148)$; Namun, biaya ini lebih besar daripada mengalokasikan A ke MRAM, jadi kami tetap mengalokasikan A ke MRAM, ketika total biaya: $4028 = 4550 - (640 - 118)$.
- Jika mengalokasikan G ke MRAM, total biayanya adalah: $3921 = 4550 - (710 - 81)$.

Jika $y = 1$, yang berarti bahwa satu ruang MRAM dan satu ruang SRAM dapat digunakan

- Asumsikan A menggunakan MRAM, jika B menggunakan SRAM, total biayanya adalah $3469 = 4028 - (640 - 81)$.
- Asumsikan A menggunakan MRAM, jika C menggunakan SRAM, total biayanya adalah $3469 = 4028 - (640 - 81)$.
- Asumsikan A menggunakan MRAM, jika D menggunakan SRAM, total biayanya adalah $3469 = 4028 - (640 - 81)$.
- Asumsikan A menggunakan MRAM, jika E menggunakan SRAM, total biayanya adalah $3469 = 4028 - (640 - 81)$.
- Asumsikan A menggunakan MRAM, jika F menggunakan SRAM, total biayanya adalah $3469 = 4028 - (640 - 81)$.
- Asumsikan A menggunakan MRAM, jika G menggunakan SRAM, total biayanya adalah $3334 = 4028 - (710 - 16)$. Asumsikan A menggunakan SRAM, jika G menggunakan MRAM, total biayanya adalah $3399 = 3921 - (640 - 118)$. $3334 < 3399$; dengan demikian, kami memilih 3334 sebagai total biaya terendah, yang menunjukkan bahwa A menggunakan MRAM, sedangkan G menggunakan SRAM.

Jika $y = 0$, yang berarti satu ruang MRAM dan dua ruang SRAM dapat digunakan:

- Dengan asumsi A menggunakan MRAM dan B menggunakan SRAM, jika C menggunakan SRAM, total biayanya adalah $2910 = 3469 - (640 - 81)$.
- Dengan asumsi A menggunakan MRAM dan B menggunakan SRAM, jika D menggunakan SRAM, total biayanya adalah $2910 = 3469 - (640 - 81)$.
- Dengan asumsi A menggunakan MRAM dan B menggunakan SRAM, jika E menggunakan SRAM, total biayanya adalah $2910 = 3469 - (640 - 81)$.
- Dengan asumsi A menggunakan MRAM dan B menggunakan SRAM, jika F menggunakan SRAM, total biayanya adalah $2910 = 3469 - (640 - 81)$.
- Dengan asumsi G menggunakan MRAM, dua jenis data lainnya menggunakan SRAM, dan total biayanya adalah: $2803 = 3921 - 2*(640-81)$. Dengan asumsi bahwa G menggunakan SRAM, dua jenis data lainnya menggunakan satu MRAM dan satu SRAM, dan total biayanya adalah: $2775 = 3469 - (710 - 16)$. Biaya strategi kedua lebih rendah dari yang pertama, sehingga kami memilih 2775 sebagai pilihan alternatif.

x = 3							
y	A	B	C	D	E	F	G
0	--	--	2910	2910	2910	2910	2775
1	--	3469	3469	3469	3469	3469	3334
2	4028	4028	4028	4028	4028	4028	3921

x = 4							
y	A	B	C	D	E	F	G
0	--	3432	3432	3432	3432	3432	3297
1	3991	3991	3991	3991	3991	3991	3856
2	4550						

Gambar 10.12 Langkah kedua pemrograman dinamis.

Gambar 10.13 menunjukkan semua biaya ketika $x = 2$, $y = 0, 1, 2$. Jika $x = 2$, $y = 2$, yang berarti bahwa dua ruang MRAM dapat digunakan:

- Asumsikan A menggunakan MRAM, jika mengalokasikan B ke MRAM, total biayanya adalah: $3512 = 4028 - (640 - 124)$.
- Asumsikan A menggunakan MRAM, jika mengalokasikan C ke MRAM, total biayanya adalah: $3518 = 4028 - (640 - 130)$. Asumsikan A menggunakan MRAM, jika mengalokasikan B ke MRAM, total biayanya adalah: $3512 = 4028 - (640 - 124)$. Hasilnya, kami memilih 3512 sebagai pilihan, yang berarti $A \rightarrow \text{MRAM}$, $B \rightarrow \text{MRAM}$. D, E, dan F sama dengan C; jadi, biaya semuanya adalah 3512.
- Asumsikan A menggunakan MRAM, jika mengalokasikan B ke MRAM, total biayanya adalah: $3512 = 4028 - (640 - 124)$. Dengan asumsi A menggunakan MRAM, dan mengalokasikan G ke MRAM, total biayanya adalah $3399 = 4028 - (710 - 81)$. Hasilnya, kami memilih 3399 sebagai pilihan, yang berarti $A \rightarrow \text{MRAM}$, $G \rightarrow \text{MRAM}$.

Jika $x = 2$, $y = 1$, yang berarti bahwa dua ruang MRAM dan satu ruang SRAM dapat digunakan:

- Dengan asumsi A menggunakan MRAM dan B menggunakan MRAM, jika mengalokasikan C ke SRAM, total biayanya adalah: $2953 = 3512 - (640 - 81)$. Strateginya adalah $A \rightarrow \text{MRAM}$, $B \rightarrow \text{MRAM}$, $C \rightarrow \text{SRAM}$.
- Dengan asumsi A menggunakan MRAM dan B menggunakan MRAM, jika mengalokasikan D ke SRAM, total biayanya adalah: $2953 = 3512 - (640 - 81)$. Strateginya adalah $A \rightarrow \text{MRAM}$, $B \rightarrow \text{MRAM}$, $D \rightarrow \text{SRAM}$.
- Dengan asumsi A menggunakan MRAM dan B menggunakan MRAM, jika mengalokasikan E ke SRAM, total biayanya adalah: $2953 = 3512 - (640 - 81)$. Strateginya adalah $A \rightarrow \text{MRAM}$, $B \rightarrow \text{MRAM}$, $E \rightarrow \text{SRAM}$.
- Dengan asumsi A menggunakan MRAM dan B menggunakan MRAM, jika mengalokasikan F ke SRAM, total biayanya adalah: $2953 = 3512 - (640 - 81)$. Strateginya adalah $A \rightarrow \text{MRAM}$, $B \rightarrow \text{MRAM}$, $F \rightarrow \text{SRAM}$.

- Asumsikan A menggunakan MRAM dan B menggunakan MRAM, jika mengalokasikan G ke SRAM, total biayanya adalah: $2818 = 3512 - (710 - 16)$. Dengan asumsi A menggunakan SRAM, B dan G menggunakan MRAM, total biayanya adalah $2846 = 4550 - (640 - 81) - (640 - 124) - (710 - 81)$. Hasilnya, kami memilih 2818 sebagai pilihan, yang berarti $A \rightarrow \text{MRAM}, B \rightarrow \text{MRAM}, G \rightarrow \text{SRAM}$.

Kemudian kita dapat menghitung biaya ketika $x = 0, 1$, dan semua hasilnya ditunjukkan pada Gambar 10.13. Kemajuan penghitungan serupa dengan yang sebelumnya di Bab 9.

Setelah menghitung semua hasil, kami dapat melacak kembali biaya terendah setiap tabel untuk setiap data dari strategi ketika $x = 0, y = 0$. Seperti yang ditunjukkan pada Gambar 10.14, kami mendapatkan jejak sebagai berikut:

G: $1245 = 1939 - (710 - 16)$, $G \rightarrow \text{SRAM}$

x = 0							
y	A	B	C	D	E	F	G
0	--	--	--	--	--	1380	1245
1	--	--	--	--	1939	1939	1804
2	--	--	--	2498	2498	2498	2373

x = 1							
y	A	B	C	D	E	F	G
0	--	--	--	--	1884	1884	1749
1	--	--	--	2443	2443	2443	2308
2	--	--	3002	3002	3002	3002	2883

x = 2							
y	A	B	C	D	E	F	G
0	--	--	--	2394	2394	2394	2259
1	--	--	2953	2953	2953	2953	2818
2	--	3512	3512	3512	3512	3512	3399

Gambar 10.13 Langkah ketiga pemrograman dinamis.

F: $1939 = 1939$, $F \rightarrow \text{Memori utama}$

E: $1939 = 2498 - (640 - 81)$, $E \rightarrow \text{SRAM}$

D: $2498 = 3002 - (640 - 136)$, $D \rightarrow \text{MRAM}$

C: $3002 = 3512 - (640 - 130)$, $C \rightarrow \text{MRAM}$

B: $3512 = 4028 - (640 - 124)$, $B \rightarrow \text{MRAM}$

A: $4028 = 4550 - (640 - 118)$, $A \rightarrow \text{MRAM}$

x = 0							
y	A	B	C	D	E	F	G
0	--	--	--	--	--	1380	1245
1	--	--	--	--	1939	1939	1804
2	--	--	--	2498	2498	2498	2373

x = 1							
y	A	B	C	D	E	F	G
0	--	--	--	--	1884	1884	1749
1	--	--	--	2443	2443	2443	2308
2	--	--	3002	3002	3002	3002	2883

x = 2							
y	A	B	C	D	E	F	G
0	--	--	--	2394	2394	2394	2259
1	--	--	2953	2953	2953	2953	2818
2	--	3512	3512	3512	3512	3512	3399

x = 3							
y	A	B	C	D	E	F	G
0	--	--	2910	2910	2910	2910	2775
1	--	3469	3469	3469	3469	3469	3334
2	4028	4028	4028	4028	4028	4028	3921

x = 4							
y	A	B	C	D	E	F	G
0	--	3432	3432	3432	3432	3432	3297
1	3991	3991	3991	3991	3991	3991	3856
2	4550	4550	4550	4550	4550	4550	4550

Gambar 10.14 Langkah keempat pemrograman dinamis.

Kita dapat merumuskan total biaya sebagai berikut:

$$\begin{aligned}
 & Cost[d_i, m_1, m_2, \dots, m_M] = \\
 & \begin{cases} \sum_{i=1}^N C(d_i, MM_i), & \text{SRAM and MRAM are full} \\ \infty, & \text{there is not enough space for data } i \end{cases} \\
 & \min \begin{pmatrix} Cost[d-1, m_1, m_2, \dots, m_M] \\ Cost[d_i, m_1 + 1, m_2, \dots, m_M] - (C(d_i, MM_i) - C(d_i, m_1)) \\ Cost[d_i, m_1, m_2 + 1, \dots, m_M] - (C(d_i, MM_i) - C(d_i, m_2)) \\ \dots \\ Cost[d_i, m_1, m_2, \dots, m_M + 1] - (C(d_i, MM_i) - C(d_i, m_M)) \end{pmatrix}, \text{ else}
 \end{aligned}
 \tag{10.4}$$

Tabel 10.7 menunjukkan hasil perbandingan pemrograman dinamis multidimensi dengan algoritma heuristik. Sangat mudah untuk membuktikan bahwa algoritma pemrograman dinamis multidimensi ini optimal, dan kompleksitas waktunya berada di bawah waktu polinomial, yaitu $O(N * \text{sizeof}(\text{SRAM}) * \text{sizeof}(\text{MRAM}))$. Namun, algoritma ini memiliki kelemahan, yaitu kompleksitas ruangnya terlalu tinggi, terutama untuk prosesor multi-core.

Tabel 10.7 Hasil Perbandingan dengan Algoritma Heuristik

Alokasi	Heuristik	Pemrograman dinamis multi-dimensi
SRAM	A,B	MISALNYA
MRAM	C,D,E,F	A,B,C,D
Utama	G	F
Biaya	1428	1245

10.4 MASALAH KEAMANAN DAN PRIVASI

Sekarang ada lebih dari enam miliar smartphone yang digunakan untuk menghasilkan data. Informasi tentang setiap teks, setiap pencarian, setiap panggilan telepon, setiap email, dan setiap gambar atau video yang diunggah atau disimpan pengguna. Setiap pengguna smartphone akan menghasilkan sekitar 60 gigabyte data setiap tahun. Kemudian kalikan dengan enam miliar perangkat, kami menghasilkan dan menyimpan lebih dari 335 exabyte informasi setiap tahun hanya dengan ponsel pintar. Tidak diragukan lagi, data ini adalah data besar, dan bermanfaat bagi pekerjaan, studi, dan kehidupan kita. Namun, data ini berisi informasi pribadi kami dan membawa potensi bahaya keamanan yang tidak terduga.

Data lokasi merupakan data yang paling sering digunakan di smartphone. Ini dapat digunakan di aplikasi yang fungsi utamanya adalah peta, pengorganisasian foto, layanan jejaring sosial (SNS), rekomendasi belanja dan restoran, serta cuaca. Seperti yang disebutkan di bab pertama, 50% aplikasi gratis iOS teratas dan 24% aplikasi berbayar iOS teratas untuk melacak lokasi pengguna. Meskipun pengguna diperingatkan setiap kali aplikasi bermaksud untuk menangkap lokasi, mereka biasanya memilih untuk mengizinkan izin untuk fungsi yang ditawarkan oleh aplikasi tersebut. Risiko pertama dan paling langsung adalah masalah keamanan fisik. Jejak pengguna mudah diekspos ke seseorang yang memiliki data lokasi real-time dan akurat mereka. Menggunakan beberapa metode penambangan data sederhana, kebiasaan dan kebiasaan pengguna mudah untuk disimpulkan. Penguntit, perampokan, bahkan pembunuhan lebih mungkin terjadi pada mereka. Risiko kedua adalah tentang kekhawatiran spionase perusahaan, pemerintah, dan militer.

Album juga banyak digunakan di smartphone. Pengguna tidak hanya mengambil foto untuk memori, tetapi juga untuk kenyamanan, seperti mengambil foto alih-alih menuliskannya dan mencetak rute yang ditemukan oleh Maps. Hasilnya, smartphone dengan kapasitas penyimpanan yang besar menyimpan lebih banyak gambar, termasuk foto kehidupan dan gambar informasi. Sementara itu, karena popularitas SNS, orang membentuk kebiasaan memposting foto yang menunjukkan apa yang mereka lakukan menggunakan aplikasi SNS. Hampir semua aplikasi SNS berhasil mendapatkan izin ke Album pengguna. Ada beberapa jenis aplikasi lain yang juga mengakses Album pengguna, seperti aplikasi penyimpanan cloud, aplikasi wallpaper, aplikasi album yang disesuaikan, dan aplikasi dekorasi gambar. Faktanya, pengguna menggunakan aplikasi ini untuk menangani hanya beberapa atau bagian dari foto mereka, tidak semuanya.

Namun, sistem operasi ponsel cerdas saat ini hanya memberikan otorisasi izin kasar, semua atau tidak sama sekali. Lebih jauh lagi, otorisasi izin semacam ini selalu operasi satu kali, dan hampir selalu muncul selama penggunaan pertama kali. Jika pengguna mengizinkan aplikasi untuk mengakses album mereka sekali, aplikasi ini akan memiliki izin ini selamanya. iOS memberi pengguna cara untuk melarikan diri, di mana pengguna dapat secara manual menonaktifkan izin aplikasi untuk foto mereka di Pengaturan Privasi. Lebih buruk lagi, di Android, pengguna tidak memiliki cara untuk menonaktifkan izin beberapa aplikasi kecuali aplikasi tersebut di-uninstall. Foto harian pengguna mengungkapkan kehidupan sehari-hari mereka. Pengumpulan foto yang berlebihan tidak hanya melanggar hak pengguna, seperti potret, tetapi juga dapat merusak reputasi pengguna. Selain itu, foto pengguna tentang

informasi jauh lebih berharga daripada foto harian. Dengan bantuan aplikasi pengumpulan data yang berlebihan, organisasi pihak ketiga dapat mengumpulkan beberapa foto dari ponsel cerdas pengguna dan menggunakannya untuk perdagangan. Perilaku semacam ini sama dengan mencuri aset dari pengguna. Misalnya, seorang desainer menumbuhkan ide baru dan menggambar konsep, lalu dia mengambil foto untuk direkam. Akan ada kerugian langsung baginya jika fotonya dikumpulkan oleh beberapa aplikasi dan dijual ke pesaingnya. Secara umum, sangat menyebalkan untuk menemukan foto diri sendiri yang ditampilkan di beberapa iklan atau di suatu tempat melebihi harapan seseorang.

Untuk menghubungi orang lain dengan lebih nyaman, pengguna bersedia membuat kontak baru, mengisi kembali kontak yang ada dengan alamat email, nomor telepon baru, dan komentar. Fungsionalitas buku alamat memang memberikan kemudahan bagi pengguna untuk berkomunikasi dan bekerja. Namun, pengumpulan buku alamat pengguna yang berlebihan membawa potensi bahaya keamanan yang serius. Buku alamat mencakup nama pengguna, alamat fisik, nomor telepon, alamat email, dan catatan lainnya. Mirip dengan foto, sistem operasi ponsel cerdas saat ini gagal memberikan otorisasi izin yang terperinci. Setelah aplikasi mendapatkan izin ke buku alamat pengguna, aplikasi dapat mengumpulkan semua informasi di buku alamat ini. Data tentang buku alamat biasanya ditangkap oleh aplikasi SNS, aplikasi game online, aplikasi komersial dan aplikasi yang memiliki fungsi "berbagi dengan teman Anda". Kontak pengguna sangat berharga. Data ini dapat digunakan oleh pengembang aplikasi untuk memperluas basis pelanggan mereka, dan digunakan oleh organisasi pihak ketiga untuk memasarkan aplikasi atau layanan seluler tambahan kepada mereka yang ada dalam daftar kontak.

Selain itu, pengumpulan data kontak yang berlebihan dapat membawa potensi spionase perusahaan. Jika pengguna mencolokkan ponsel cerdasnya ke desktop perusahaannya di tempat kerja, itu akan memberi mereka opsi untuk menyinkronkan dengan kontak dari perangkat lunak email, seperti Outlook, yang selalu menyertakan kontak pribadi dan perusahaan. Namun, aplikasi hanya menanyakan kepada pengguna apakah akan memberikan izin untuk mengakses buku alamat, meskipun kontak tersebut milik perusahaan. Itu sangat mengurangi keamanan data perusahaan.

Aplikasi kalender digunakan untuk mengatur jadwal pengguna, melacak acara, dan mengingatkan pengguna tentang acara yang akan datang. Pengguna menyimpan nama dan nomor telepon untuk peserta rapat, tanggal rapat, dan waktu serta lampiran dalam bagian komentar. Baik iOS dan Android memiliki aplikasi kalender yang terpasang di sistem operasi, dan tidak mungkin untuk menghapus instalasinya kecuali jailbreak. Aplikasi kalender ini dapat dengan mudah mendapatkan izin ke kalender pengguna karena fungsi utamanya adalah mengelola kalender pengguna. Akibatnya, pengguna biasanya tidak menolak izin aplikasi kalender ke kalender mereka. Selain itu, ada beberapa jenis aplikasi lain, yang ditawarkan di pasar yang juga menggunakan kalender sebagai fungsi tambahannya, seperti aplikasi gaya hidup, aplikasi perjalanan, dan aplikasi bisnis. Setelah mendapatkan izin ke kalender pengguna, semua aplikasi ini akan mengakses setiap baris informasi yang disimpan dalam kalender itu. Risiko utama pengumpulan data kalender adalah spionase perusahaan. Data kalender mencakup informasi tentang rapat dan acara pengguna. Informasi tentang rapat

adalah rahasia komersial. Ini akan menjadi kerugian besar bagi satu perusahaan jika informasi kalender seperti informasi panggilan masuk, kata sandi panggilan, orang-orang yang hadir, dan topik yang dibahas diungkapkan kepada para pesaingnya. Setidaknya, pengungkapan informasi tentang acara pribadi pengguna juga membawa pelecehan kepada pengguna, jika informasi ini dimiliki oleh organisasi pihak ketiga, seperti biro iklan.

Baik International Mobile Station Equipment Identity (IMEI) dan Unique Device Identifier (UDID) adalah ID unik dari ponsel seseorang, dan keduanya seperti cookie di situs web yang tidak dapat dihapus oleh pengguna. Meskipun Apple telah melarang pengembang iOS menggunakan UDID sebagai metode untuk melacak dan mengidentifikasi pengguna, aturan ini hanya diterapkan pada perangkat dengan versi iOS terbaru. Bahkan, tingkat adopsi iOS 8 baru naik sekitar 50 persen. Selain itu, Apple telah mendorong pengembang aplikasi untuk menggunakan metode baru identifikasi pengguna, dan untuk melacak perilaku pengguna berdasarkan aplikasi demi aplikasi. Seperti yang kita ketahui bersama, pentingnya Primary Key dalam database, IMEI/UDID sama pentingnya dengan Primary Key. Ini mengidentifikasi data di telepon dan membuat semua data dikategorikan berdasarkan perangkat. Itu membuat data lebih dekat ke dunia nyata dan lebih berharga untuk penambangan data. Perhatian utama dalam mengakses IMEI/UDID adalah bahwa perilaku pengguna dapat dikorelasikan di beberapa aplikasi dan dicocokkan dengan pengguna unik. Bahkan jika pengguna memiliki berbagai nama pengguna dan kata sandi untuk setiap aplikasi, datanya dapat dengan mudah diintegrasikan dengan ID unik perangkatnya. Selain itu, IMEI/UDID juga dapat digunakan untuk mencocokkan dengan data pengguna yang sebenarnya, seperti nama, kata sandi, lokasi, dan lainnya. Hal ini memungkinkan pengembang aplikasi dan jaringan iklan untuk membuat profil lengkap pengguna di beberapa aplikasi dan profil dan menggabungkannya dengan data lain yang dikumpulkan untuk tampilan pengguna yang mendalam.

Beberapa aplikasi SNS, seperti Facebook dan Twitter, dapat mengakses data lokasi, foto, buku alamat, dan bahkan UDID. Misalnya, foto yang diambil dan dibagikan oleh aplikasi SNS disematkan dengan lokasi, data, dan waktu, serta di mana dan kapan foto ini diambil. Perilaku mengambil dan mengklik untuk membagikan foto diselesaikan secara manual oleh pengguna, sementara mengunggah atau mengirim dilakukan secara otomatis oleh aplikasi. Lebih jauh lagi, pengunggahan atau pengiriman sebagian besar dijalankan di latar belakang, dan sebagian besar pengguna bahkan tidak akan secara aktif mengetahui foto mana yang sedang diunggah dan kapan. Lebih buruk lagi, foto dengan informasi lokasi tertanam yang diunggah oleh aplikasi yang berbeda dapat diberi label dengan IMEI/UDID yang sama, yang menunjukkan bahwa foto tersebut dikirim dari perangkat yang sama dan menghilangkan semua privasi pengguna. Kami membahas perilaku aplikasi dalam mengumpulkan data dalam dua aspek sebagai berikut.

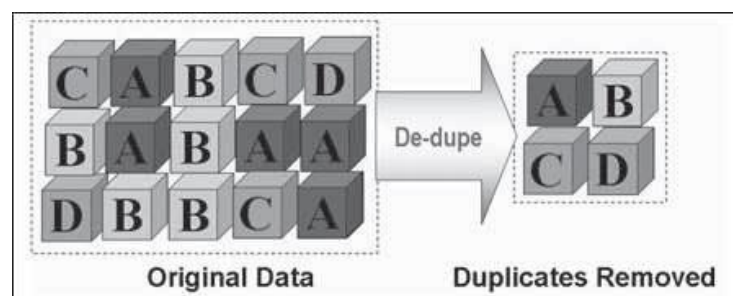
Karena lingkungan pengembangan yang terkunci dan audit manual, iOS tampaknya lebih aman daripada sistem operasi ponsel cerdas lainnya. Namun, kedua kendala ini terutama bekerja dalam memecahkan malware, tetapi tidak pada pengumpulan data. Dari laporan Appthority, 93% aplikasi iOS menunjukkan setidaknya satu perilaku berisiko, dibandingkan dengan 89% aplikasi Android. Akibatnya, pengumpulan data yang berlebihan oleh aplikasi

menjadi perhatian semua pengguna ponsel pintar. Hampir semua sistem operasi smartphone saat ini hanya menyediakan izin kasar untuk aplikasi. Pengguna hanya memiliki dua pilihan, ya atau tidak, ketika sebuah aplikasi meminta izin untuk mengakses data pengguna. Pengguna biasanya memilih ya untuk memberikan izin ke aplikasi untuk fungsi yang ditawarkannya. Namun, itu berarti aplikasi ini mendapat izin untuk mengakses setiap baris data ini. Izin semacam ini jelas terlalu berlebihan dan perlu dirinci. Namun, sangat rumit untuk mengetahui data apa di bumi yang diizinkan oleh pengguna untuk diakses oleh aplikasi. Sebagian besar sistem operasi ponsel cerdas saat ini tidak dapat memberikan izin terperinci. Untungnya, Apple pertama kali memberikan beberapa izin tentang data lokasi ke aplikasi di iOS 8, yaitu "saat digunakan" dan "selalu". Sayangnya, data lain di iOS atau semua data di non-iOS masih mengalami krisis karena terlalu banyak dikumpulkan. Lebih jauh lagi, memberikan banyak izin bukanlah solusi yang ramah pengguna, yang membebani pengguna dengan kompleksitas operasi yang lebih tinggi.

10.5 DEDUPLICASI DATA

Dengan peningkatan data besar yang tak terbendung, industri penyimpanan berlomba untuk menyediakan perangkat penyimpanan data dengan kepadatan yang lebih tinggi dan lebih tinggi dengan biaya yang lebih rendah, dan teknologi deduplikasi data menjadi semakin penting. Deduplikasi data adalah teknik untuk mengurangi jumlah ruang penyimpanan yang dibutuhkan, seperti yang ditunjukkan pada Gambar 10.15. Di sebagian besar organisasi, sistem file berisi salinan duplikat dari banyak bagian data untuk keamanan, ketersediaan, keandalan, dan kinerja. Namun, jumlah data meningkat begitu cepat sehingga menjadi penghambat dari total kinerja itu sendiri. Untungnya, teknik deduplikasi data dapat membantu menghilangkan salinan tambahan dengan petunjuk yang mengarah kembali ke salinan asli. Salah satu bentuk paling umum dari implementasi deduplikasi data bekerja dengan membandingkan potongan data untuk mendeteksi duplikat.

Memotong. Dalam beberapa sistem, potongan didefinisikan oleh batasan lapisan fisik. Sistem lain hanya membandingkan file lengkap, yang disebut penyimpanan instans tunggal. Metode chunking yang paling cerdas adalah blok geser. Dalam blok geser, sebuah jendela dilewatkan di sepanjang aliran file untuk mencari batas file internal yang lebih alami.



Gambar 10.15 Deduplikasi data.

Deduplikasi cadangan klien. Ini adalah proses di mana kalkulasi hash deduplikasi awalnya dibuat pada mesin sumber. File yang memiliki hash identik dengan file yang sudah ada di perangkat target tidak dikirim, perangkat target hanya membuat tautan internal yang sesuai untuk mereferensikan data yang diduplikasi. Keuntungannya adalah menghindari pengiriman data yang tidak perlu melalui jaringan, sehingga mengurangi beban lalu lintas.

Penyimpanan primer dan penyimpanan sekunder. Menurut definisi, sistem penyimpanan utama dirancang untuk kinerja yang optimal, daripada biaya serendah mungkin. Kriteria desain untuk sistem ini adalah untuk meningkatkan kinerja, dengan mengorbankan pertimbangan lain. Selain itu, sistem penyimpanan utama kurang toleran terhadap operasi apa pun yang dapat berdampak negatif pada kinerja. Sistem penyimpanan sekunder terutama berisi duplikat, atau salinan data sekunder. Salinan data ini biasanya tidak digunakan untuk operasi produksi aktual dan, sebagai hasilnya, lebih toleran terhadap beberapa penurunan kinerja sebagai ganti peningkatan efisiensi.

10.6 LATIHAN

10.6.1 Pertanyaan Mendasar

1. Apa itu data terstruktur?
2. Apa itu data tidak terstruktur?
3. Apa itu data semi terstruktur?
4. Tolong jelaskan Volume Big Data.
5. Apa Kebenaran Big Data?
6. Apa yang dimaksud dengan pembelajaran terawasi dalam pembelajaran mesin?
7. Dalam Teorema Bayes, bagaimana cara menghitung $P(A|B)$, jika $P(B|A)$, $P(A)$, dan $P(B)$ sudah diketahui?
8. Apa itu pohon keputusan?
9. Apa itu pembelajaran tanpa pengawasan?
10. Untuk apa Clustering digunakan dalam pembelajaran tanpa pengawasan?
11. Apa itu pembelajaran semi-diawasi?
12. Apa itu Hukum Moore?
13. Apa itu dinding memori?
14. Apa itu memori non-volatile?
15. Apa itu Memori Scratch Pad?
16. Apa itu sistem CMP?
17. Apakah data yang dihasilkan oleh perangkat seluler merupakan data besar?
18. Apa yang dimaksud dengan chunk dalam deduplikasi data?

10.6.2 Pertanyaan Praktis

1. Dalam Bagian 10.3, kami menghitung hasil dari $x = 4$ hingga $x = 2$. Harap hitung semua hasil ketika $x = 1$, $y = 0, 1, 2$. Biaya pengalokasian setiap data ke unit memori

yang berbeda ditunjukkan pada Tabel 10.6. Hasilnya ditunjukkan pada Gambar 10.14. Tolong beri rincian kemajuan penghitungan.

DAFTAR PUSTAKA

- Abdul Kadir, T.C Triwahyuni. (2013). Pengenalan Teknologi Informasi. Edisi II. Penerbit Andi, Yogyakarta.
- Bush, T. &. (2006). Leading and Managing People in Education. London: SAGE Publication.
- Fitri, N., & Budayawan, K. (2019). Rancang Bangun Aplikasi Media Pembelajaran Sistem Komputer Berbasis Mobile. Jurnal Vokasional Teknik Elektronika Dan Informatika, 7(3), 211-219.
- Hermawan S., Stephanus., (2011), Mudah membuat Aplikasi Android, Yogyakarta: Andi offset.
- Izzatul Mufidah, C. (2014). Pengembangan Modul Pembelajaran Pada Kompetensi Dasar Hubungan Masyarakat Kelas X APK 2 di SMKN 10 Surabaya. Jurnal Administrasi Perkantoran (JPAP), 2(2).
- Karimuribo, Esron Daniel dkk., 2017. A Smartphone App (AfyaData) for Innovative One Health Disease Surveillance from Community to National Levels in Africa: Intervention in Disease Surveillance. JMIR Public Health Surveill ; 3(4):e94.
- Kasman, Akhmad Dharma. (2013). Kolaborasi Dahsyat Android PHP & Mysql. Lokomedia, Yogyakarta.
- N. Safaat. (2015). Pemrograman Aplikasi Mobile Smartphone dan Tablet PC berbasis Android. Bandung: Informatika.
- Satyaputra, Alfa & Maulina Eva Aritonang. (2016). Let's Build Your Android Apps With Android Studio. Jakarta: PT Elex Media Komputindo.



APLIKASI TEKNOLOGI SELULER Dengan **Android**

Dr. Joseph Teguh Santoso, S.Kom, M.Kom

BIODATA PENULIS



Dr. Joseph Teguh Santoso, S.Kom, M.Kom adalah Rektor dari Universitas Sains & Teknologi Komputer (Universitas STEKOM) Semarang yang memiliki banyak pengalaman praktis dalam bidang *e-commerce* sejak Tahun 2002. Beliau mempunyai 3 (tiga) toko *Official Online Store* di China untuk merek Sepeda Raleigh, dengan omzet tahunan pada Tahun 2019 mencapai lebih dari Rp. 35 Milyar rupiah dan terus meningkat. Dr. Joseph T.S memiliki lisensi tunggal sepeda merek “Raleigh” untuk penjualan *Online* di seluruh China. Di samping itu beliau juga memiliki pabrik sepeda dan sepeda listrik merek “Fengjiu”, yaitu Pabrik Sepeda Listrik yang masih tergolong kecil di China. Pengalaman beliau malang melintang di dunia *online store* di China seperti Alibaba, Tmall, Taobao, JD, Aliexpress sangat membantu mahasiswa untuk memiliki pengalaman teknis dan praktis untuk membuka toko *online* bersama beliau.



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-5734-49-1 (PDF)



9 786235 734491

APLIKASI TEKNOLOGI SELULER Dengan **Android**

Dr. Joseph Teguh Santoso, S.Kom, M.Kom



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK

Jl. Majapahit No. 605 Semarang

Telp. (024) 6723456. Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id