

BUKU AJAR

MATA KULIAH ARSITEKTUR KOMPUTER

BAB 14

ARSITEKTUR PARALEL DAN

MULTIPROSESOR

Buku Ajar Arsitektur Komputer 2026
Berbasis Rencana Pembelajaran Semester (RPS)
IF221101 | 3 SKS

IDENTITAS BAB

Komponen	Keterangan
Mata Kuliah	Arsitektur Komputer
Kode	IF221101
Bobot	3 SKS
Bab	Bab 14
Judul	Arsitektur Paralel dan Multiprosesor
Fokus Pembelajaran	Memahami konsep paralelisme, organisasi multiprosesor, multicore, shared memory, distributed memory, sinkronisasi, dan evaluasi kinerja sistem paralel
Level Pembelajaran	Pemahaman konsep, analisis organisasi paralel, evaluasi speedup dan efisiensi, serta perancangan simulasi sederhana

CAPAIAN PEMBELAJARAN BAB

Setelah mempelajari bab ini, mahasiswa diharapkan mampu:

1. Menjelaskan alasan penggunaan pemrosesan paralel dalam sistem komputer.
2. Membedakan konsep concurrency dan parallelism.
3. Mengidentifikasi bentuk paralelisme pada tingkat instruksi, data, dan tugas.
4. Menjelaskan klasifikasi arsitektur paralel secara konseptual.
5. Membedakan organisasi shared memory dan distributed memory.
6. Menjelaskan konsep multicore dan multiprosesor.
7. Menjelaskan masalah komunikasi, sinkronisasi, dan race condition.
8. Menganalisis konsep speedup, efisiensi, dan scalability.
9. Menggunakan Hukum Amdahl untuk menganalisis batas percepatan teoritis.
10. Mengevaluasi trade-off antara jumlah prosesor dan bagian program yang dapat diparalelkan.
11. Merancang model sederhana pembagian pekerjaan paralel.

PETA KONSEP

ALUR KONSEP BAB 14

Kebutuhan Kinerja → Concurrency dan Parallelism → Tingkat Paralelisme → Flynn Taxonomy → Multicore dan Multiprosesor → Shared Memory → Distributed Memory → Interkoneksi → Sinkronisasi → Race Condition → Speedup → Efficiency → Scalability → Hukum Amdahl.

14.1 Pendahuluan

Peningkatan kinerja komputer tidak selalu dicapai dengan menaikkan frekuensi satu prosesor. Batas daya, panas, kompleksitas desain, dan karakteristik aplikasi mendorong penggunaan beberapa unit pemrosesan yang dapat bekerja secara bersamaan.

Arsitektur paralel memanfaatkan pembagian pekerjaan menjadi bagian-bagian yang dapat dikerjakan secara serentak. Namun, peningkatan jumlah prosesor tidak secara otomatis menghasilkan percepatan yang sebanding karena terdapat bagian program yang tetap bersifat serial, biaya komunikasi, dan kebutuhan sinkronisasi.

PERTANYAAN PEMANTIK

Jika sebuah komputer memiliki 8 core, apakah semua program pasti berjalan delapan kali lebih cepat? Bagian mana dari program yang dapat menjadi pembatas percepatan?

14.2 Concurrency dan Parallelism

Konsep	Penjelasan
Concurrency	Beberapa aktivitas mengalami kemajuan dalam periode waktu yang saling tumpang tindih secara konseptual.
Parallelism	Beberapa operasi atau pekerjaan benar-benar dieksekusi secara bersamaan menggunakan sumber daya pemrosesan yang tersedia.

Concurrency berkaitan dengan struktur dan pengelolaan beberapa aktivitas, sedangkan parallelism menekankan pelaksanaan serentak. Dalam sistem modern, keduanya dapat muncul bersamaan.

14.3 Mengapa Pemrosesan Paralel Diperlukan?

- Meningkatkan throughput untuk banyak pekerjaan.
- Mengurangi waktu penyelesaian untuk pekerjaan yang dapat dibagi.
- Menangani volume data dan komputasi yang besar.
- Mendukung aplikasi ilmiah, analitik, grafika, kecerdasan buatan, dan layanan berskala besar.
- Memanfaatkan beberapa core atau prosesor yang tersedia.

Efektivitas paralelisme sangat bergantung pada kemampuan masalah untuk dibagi serta biaya koordinasi antar bagian.

14.4 Tingkat Paralelisme

Tingkat	Contoh Konseptual
Instruction-Level Parallelism	Beberapa operasi instruksi dapat diproses dengan tumpang tindih atau bersamaan sesuai organisasi prosesor.
Data-Level Parallelism	Operasi yang sama diterapkan pada banyak elemen data.
Task-Level Parallelism	Tugas berbeda dikerjakan secara bersamaan.
Thread-Level Parallelism	Beberapa thread dieksekusi menggunakan sumber daya pemrosesan berbeda.

Satu aplikasi dapat memanfaatkan lebih dari satu tingkat paralelisme secara bersamaan.

14.5 Klasifikasi Flynn

Klasifikasi Flynn digunakan sebagai kerangka konseptual berdasarkan jumlah aliran instruksi dan aliran data.

Kategori	Instruksi	Data	Gagasan Utama
SISD	Single	Single	Satu aliran instruksi terhadap satu aliran data
SIMD	Single	Multiple	Satu instruksi diterapkan pada banyak elemen data
MISD	Multiple	Single	Kombinasi khusus dan jarang digunakan sebagai model umum
MIMD	Multiple	Multiple	Beberapa unit pemrosesan menjalankan instruksi berbeda terhadap data berbeda

CATATAN

Klasifikasi Flynn adalah alat konseptual. Sistem nyata dapat memiliki kombinasi karakteristik yang tidak selalu dapat dijelaskan oleh satu kategori sederhana.

14.6 Multicore dan Multiprosesor

Multicore mengintegrasikan beberapa core pemrosesan dalam satu chip atau paket sesuai rancangan perangkat keras. Multiprosesor secara lebih umum mengacu pada sistem yang memiliki lebih dari satu unit pemrosesan.

Aspek	Multicore	Multiprosesor
Organisasi umum	Beberapa core dalam rancangan prosesor	Beberapa prosesor atau unit pemrosesan dalam sistem
Komunikasi	Bergantung pada hierarki cache dan interkoneksi internal	Bergantung pada organisasi sistem dan interkoneksi
Tujuan	Meningkatkan kemampuan pemrosesan paralel	Mendukung pembagian pekerjaan antar unit pemrosesan

14.7 Shared Memory Architecture

Pada shared memory, beberapa unit pemrosesan dapat mengakses ruang memori bersama sesuai mekanisme yang disediakan sistem. Model ini memudahkan pertukaran data melalui struktur memori yang sama, tetapi membutuhkan pengelolaan akses bersama.

Kelebihan	Tantangan
Komunikasi melalui ruang data bersama	Kemungkinan race condition
Model pemrograman relatif intuitif	Kontensi akses memori

Cocok untuk berbagai sistem multicore	Kebutuhan sinkronisasi dan koherensi
---------------------------------------	--------------------------------------

14.8 Distributed Memory Architecture

Pada distributed memory, setiap unit pemrosesan atau node memiliki memori lokalnya sendiri. Pertukaran data dilakukan melalui mekanisme komunikasi antar node.

Kelebihan	Tantangan
Dapat diperluas ke banyak node	Program harus mengelola komunikasi data
Memori lokal mengurangi kebutuhan akses global tertentu	Latensi dan biaya komunikasi
Cocok untuk sistem terdistribusi dan klaster	Pembagian data lebih kompleks

14.9 Shared Memory vs Distributed Memory

Aspek	Shared Memory	Distributed Memory
Ruang data	Dapat diakses bersama	Terpisah pada node atau unit
Komunikasi	Melalui struktur memori bersama	Melalui pertukaran pesan atau mekanisme komunikasi
Sinkronisasi	Sangat penting untuk akses bersama	Penting pada koordinasi pertukaran data
Scalability	Dapat dibatasi kontensi dan interkoneksi	Dapat berkembang dengan penambahan node, tetapi komunikasi menjadi faktor penting

14.10 Interkoneksi Sistem Paralel

Unit pemrosesan memerlukan mekanisme untuk berkomunikasi dengan memori atau unit lain. Interkoneksi memengaruhi bandwidth, latensi, biaya, dan kemampuan sistem untuk berkembang.

- Bus bersama
- Interkoneksi berbasis switch
- Network-on-chip pada konteks tertentu
- Interkoneksi jaringan antar node pada sistem terdistribusi

Pemilihan interkoneksi merupakan bagian penting dari desain karena bottleneck komunikasi dapat mengurangi manfaat penambahan unit pemrosesan.

14.11 Sinkronisasi

Ketika beberapa proses atau thread mengakses data bersama, urutan operasi dapat memengaruhi hasil. Sinkronisasi digunakan untuk mengoordinasikan akses dan menjaga hubungan yang benar antar operasi.

Mekanisme Konseptual	Tujuan
Mutual Exclusion	Membatasi akses bersamaan ke critical section
Lock/Mutex	Memberikan mekanisme kepemilikan akses
Semaphore	Mengelola koordinasi berdasarkan nilai atau izin
Barrier	Menunggu sejumlah peserta mencapai titik tertentu sebelum melanjutkan
Atomic Operation	Melakukan operasi yang dipandang tidak dapat diinterupsi pada tingkat tertentu

14.12 Race Condition dan Critical Section

Race condition terjadi ketika hasil program bergantung pada urutan atau waktu relatif dari operasi yang berjalan bersamaan. Critical section adalah bagian kode yang mengakses sumber daya bersama sehingga perlu dikelola.

CONTOH KONSEPTUAL

Dua thread membaca nilai saldo yang sama, kemudian masing-masing melakukan pembaruan. Tanpa koordinasi, salah satu pembaruan dapat menimpa hasil pembaruan lainnya.

14.13 Koherensi Cache

Dalam sistem multicore dengan cache lokal atau privat, beberapa cache dapat menyimpan salinan data yang sama. Ketika satu core mengubah data, sistem perlu memiliki mekanisme agar core lain tidak menggunakan informasi yang tidak sesuai dengan keadaan yang diharapkan.

Masalah koherensi berbeda dari sekadar konsistensi algoritma aplikasi. Arsitektur perangkat keras dan perangkat lunak memiliki mekanisme serta aturan masing-masing untuk mengelola perilaku akses memori.

14.14 Speedup

Speedup menggambarkan perbandingan waktu penyelesaian antara implementasi referensi dan implementasi paralel.

RUMUS

$$\text{Speedup (S)} = T_{\text{serial}} / T_{\text{parallel}}$$

Jika pekerjaan serial selesai dalam 100 detik dan versi paralel selesai dalam 25 detik, speedup yang diperoleh adalah 4.

14.15 Efisiensi Paralel

Efisiensi digunakan untuk melihat seberapa efektif sejumlah P unit pemrosesan digunakan dalam memperoleh speedup.

RUMUS

$$\text{Efficiency (E)} = S / P$$

Jika speedup adalah 6 menggunakan 8 prosesor, efisiensi konseptualnya adalah $6/8 = 0,75$ atau 75%.

14.16 Hukum Amdahl

Hukum Amdahl menunjukkan bahwa bagian program yang tidak dapat diparalelkan akan membatasi percepatan maksimum. Jika P adalah proporsi yang dapat diparalelkan dan N adalah jumlah unit pemrosesan, model ideal sederhananya:

RUMUS

$$\text{Speedup maksimum} = 1 / ((1 - P) + P/N)$$

Misalnya 90% pekerjaan dapat diparalelkan dan tersedia 10 prosesor. Speedup ideal menurut model tersebut adalah $1 / (0,10 + 0,90/10) = \text{sekitar } 5,26$.

Model ini tidak memasukkan seluruh biaya nyata seperti komunikasi dan sinkronisasi sehingga hasilnya harus dipahami sebagai batas atau estimasi berdasarkan asumsi yang digunakan.

14.17 Scalability

Scalability menggambarkan kemampuan sistem untuk mempertahankan atau meningkatkan kinerja ketika sumber daya atau ukuran pekerjaan bertambah. Sistem yang scalable tidak hanya bergantung pada penambahan prosesor, tetapi juga pada kemampuan distribusi pekerjaan dan kapasitas komunikasi.

Faktor	Pengaruh terhadap Scalability
Bagian serial	Membatasi percepatan
Komunikasi	Menambah overhead dan latensi
Sinkronisasi	Dapat menyebabkan waktu tunggu
Load imbalance	Sebagian unit selesai lebih cepat dan menunggu unit lain
Kontensi memori	Akses bersama dapat menjadi bottleneck

14.18 Load Balancing

Load balancing bertujuan membagi pekerjaan sehingga unit pemrosesan memperoleh beban yang relatif sesuai. Pembagian yang tidak seimbang menyebabkan sebagian core menganggur ketika core lain masih bekerja.

Pendekatan Konseptual	Karakteristik
Static Scheduling	Pembagian dilakukan sebelum atau pada awal pekerjaan
Dynamic Scheduling	Pembagian dapat berubah sesuai kondisi pekerjaan

14.19 Granularity dan Overhead

Granularity menggambarkan ukuran relatif pekerjaan yang diberikan kepada satu unit pemrosesan. Pekerjaan yang terlalu kecil dapat menghasilkan overhead koordinasi yang tinggi, sedangkan pekerjaan terlalu besar dapat mengurangi kesempatan paralel.

ANALISIS

Desain paralel yang baik harus mempertimbangkan keseimbangan antara jumlah komputasi yang berguna dan biaya komunikasi atau sinkronisasi.

14.20 Studi Kasus 1: Perkalian Matriks

Perkalian matriks dapat dibagi berdasarkan baris, kolom, atau blok. Setiap unit pemrosesan dapat menghitung bagian hasil yang berbeda, kemudian hasil digabungkan sesuai struktur matriks.

Mahasiswa diminta menganalisis pembagian pekerjaan untuk matriks berukuran $N \times N$ dan mengidentifikasi kebutuhan komunikasi pada shared memory serta distributed memory.

14.21 Studi Kasus 2: Analisis Hukum Amdahl

Sebuah program memiliki 80% bagian yang dapat diparalelkan. Mahasiswa menghitung speedup teoritis menggunakan 2, 4, 8, dan 16 unit pemrosesan, kemudian menganalisis mengapa peningkatan jumlah unit menghasilkan tambahan percepatan yang semakin kecil.

Jumlah Prosesor	Model Perhitungan
2	$1 / (0,20 + 0,80/2)$
4	$1 / (0,20 + 0,80/4)$
8	$1 / (0,20 + 0,80/8)$
16	$1 / (0,20 + 0,80/16)$

14.22 Studi Kasus 3: Race Condition

Dua thread meningkatkan sebuah variabel bersama. Operasi peningkatan sebenarnya melibatkan pembacaan nilai, perhitungan nilai baru, dan penulisan kembali. Jika operasi tersebut saling tumpang tindih tanpa sinkronisasi, hasil akhir dapat tidak sesuai jumlah peningkatan yang diharapkan.

TUGAS ANALISIS

Buat dua urutan interleaving yang menghasilkan hasil berbeda. Selanjutnya tambahkan mekanisme critical section secara konseptual.

14.23 Aktivitas Pembelajaran Mahasiswa

Aktivitas 1: Identifikasi Paralelisme

Temukan contoh data-level, task-level, dan thread-level parallelism pada aplikasi modern.

Aktivitas 2: Flynn Taxonomy

Klasifikasikan beberapa skenario sistem berdasarkan SISD, SIMD, dan MIMD secara konseptual.

Aktivitas 3: Pembagian Tugas

Bagi sebuah pekerjaan besar ke 2, 4, dan 8 unit pemrosesan lalu analisis kemungkinan load imbalance.

Aktivitas 4: Hukum Amdahl

Hitung speedup teoritis untuk beberapa nilai proporsi paralel dan jumlah prosesor.

Aktivitas 5: Sinkronisasi

Buat simulasi race condition dan solusi menggunakan critical section.

Aktivitas 6: Shared vs Distributed

Bandingkan kebutuhan komunikasi pada dua organisasi memori untuk masalah yang sama.

Aktivitas 7: Scalability

Identifikasi bottleneck yang mungkin muncul ketika jumlah core terus ditambah.

14.24 Rangkuman

- Pemrosesan paralel membagi pekerjaan agar beberapa operasi dapat dijalankan secara bersamaan.
- Concurrency dan parallelism berkaitan tetapi memiliki fokus konsep yang berbeda.
- Paralelisme dapat muncul pada tingkat instruksi, data, tugas, dan thread.
- Klasifikasi Flynn mencakup SISD, SIMD, MISD, dan MIMD sebagai kerangka konseptual.
- Multicore dan multiprosesor menyediakan lebih dari satu sumber daya pemrosesan.
- Shared memory memungkinkan akses ke ruang data bersama tetapi membutuhkan sinkronisasi.
- Distributed memory menggunakan memori lokal dan membutuhkan komunikasi antar unit.
- Sinkronisasi diperlukan untuk mengelola akses terhadap data bersama.
- Race condition dapat menghasilkan keluaran yang bergantung pada urutan operasi.
- Koherensi cache menjadi isu penting ketika beberapa cache menyimpan salinan data yang sama.
- Speedup membandingkan waktu serial dan paralel.
- Efisiensi menunjukkan tingkat pemanfaatan unit pemrosesan berdasarkan speedup.
- Hukum Amdahl menunjukkan bahwa bagian serial membatasi percepatan maksimum.
- Scalability dipengaruhi oleh komunikasi, sinkronisasi, load imbalance, dan kontensi sumber daya.

14.25 Glosarium Bab

Istilah	Pengertian
Barrier	Titik sinkronisasi tempat peserta menunggu hingga kondisi

	tertentu terpenuhi.
Concurrency	Kemajuan beberapa aktivitas dalam periode waktu yang tumpang tindih.
Critical Section	Bagian kode yang mengakses sumber daya bersama.
Distributed Memory	Organisasi dengan memori lokal pada unit atau node yang berbeda.
Efficiency	Ukuran pemanfaatan unit pemrosesan berdasarkan speedup.
Flynn Taxonomy	Klasifikasi organisasi berdasarkan aliran instruksi dan data.
Load Balancing	Upaya membagi beban kerja antar unit pemrosesan.
Multicore	Organisasi yang memiliki beberapa core pemrosesan.
Parallelism	Pelaksanaan beberapa operasi secara bersamaan.
Race Condition	Kondisi ketika hasil dipengaruhi urutan atau waktu operasi bersamaan.
Scalability	Kemampuan mempertahankan atau meningkatkan kinerja saat sumber daya atau pekerjaan bertambah.
Shared Memory	Organisasi tempat beberapa unit dapat mengakses ruang memori bersama.
Speedup	Perbandingan waktu serial terhadap waktu paralel.
Synchronization	Koordinasi operasi antar proses atau thread.
Amdahl's Law	Model yang menunjukkan batas percepatan berdasarkan bagian serial dan paralel program.

14.26 Soal Latihan

A. Soal Pemahaman

- Jelaskan perbedaan concurrency dan parallelism.
- Mengapa peningkatan jumlah prosesor tidak selalu menghasilkan speedup linear?
- Jelaskan data-level dan task-level parallelism.
- Jelaskan empat kategori dalam klasifikasi Flynn.
- Bandingkan multicore dan multiprosesor.
- Bandingkan shared memory dan distributed memory.
- Mengapa sinkronisasi diperlukan?
- Jelaskan race condition dan critical section.
- Apa yang dimaksud speedup dan efficiency?
- Jelaskan gagasan utama Hukum Amdahl.

B. Soal Perhitungan dan Analisis

- Sebuah program membutuhkan 120 detik secara serial dan 30 detik secara paralel. Hitung speedup.

23. Jika speedup 6 diperoleh menggunakan 8 prosesor, hitung efficiency.
24. Gunakan Hukum Amdahl untuk menghitung speedup ideal ketika 95% program dapat diparalelkan menggunakan 4, 8, dan 32 prosesor.
25. Program memiliki 70% bagian paralel. Analisis batas speedup ketika jumlah prosesor mendekati sangat besar menurut model Amdahl.
26. Bandingkan dua pembagian beban kerja pada 4 core ketika salah satu core menerima 50% total pekerjaan.
27. Analisis pengaruh overhead komunikasi terhadap speedup teoritis.

C. Soal HOTS dan Studi Kasus

28. Rancang strategi paralelisasi untuk pemrosesan dataset berukuran besar dan jelaskan pembagian data, komunikasi, serta sinkronisasinya.
29. Sebuah aplikasi memiliki speedup rendah walaupun menggunakan banyak core. Analisis kemungkinan penyebab berdasarkan bagian serial, load imbalance, komunikasi, dan sinkronisasi.
30. Bandingkan rancangan shared memory dan distributed memory untuk aplikasi analisis citra berukuran besar.
31. Buat skenario race condition pada sistem pemesanan atau inventori dan rancang mekanisme sinkronisasi yang sesuai secara konseptual.
32. Rancang eksperimen untuk mengukur speedup dan efficiency pada jumlah worker 1, 2, 4, dan 8.

14.27 Proyek Mini

Judul proyek: Parallel Computing and Performance Simulator.

Mahasiswa membuat aplikasi, program, notebook, spreadsheet interaktif, atau simulator sederhana untuk menunjukkan dampak pembagian pekerjaan terhadap kinerja.

- Menyediakan parameter jumlah pekerjaan dan jumlah worker atau core.
- Menyediakan skenario serial dan paralel.
- Menghitung waktu konseptual, speedup, dan efficiency.
- Menyediakan parameter proporsi bagian serial dan paralel.
- Mengimplementasikan perhitungan Hukum Amdahl.
- Menyimulasikan load balancing dan load imbalance.
- Menampilkan perbandingan shared memory dan distributed memory secara konseptual.
- Menyediakan simulasi race condition atau critical section sederhana.
- Menyediakan minimal lima skenario eksperimen dan analisis hasil.

OUTPUT YANG DIHARAPKAN

Diagram arsitektur, desain pembagian pekerjaan, tabel hasil eksperimen, perhitungan speedup dan efficiency, visualisasi scalability, serta analisis trade-off desain paralel.

14.28 Daftar Pustaka

- Patterson, D. A., & Hennessy, J. L. Computer Organization and Design: The Hardware/Software Interface. Morgan Kaufmann.
- Hennessy, J. L., & Patterson, D. A. Computer Architecture: A Quantitative Approach. Morgan Kaufmann.
- Stallings, W. Computer Organization and Architecture: Designing for Performance. Pearson.
- Grama, A., Gupta, A., Karypis, G., & Kumar, V. Introduction to Parallel Computing. Addison-Wesley.

- Null, L., & Lobur, J. The Essentials of Computer Organization and Architecture. Jones & Bartlett Learning.
- Dokumen Rencana Pembelajaran Semester Mata Kuliah Arsitektur Komputer IF221101, Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Pembangunan Nasional Veteran Jawa Timur.

CATATAN PENYUSUNAN BAB

Bab ini mengembangkan pemahaman arsitektur komputer dari sistem pemrosesan tunggal menuju organisasi yang menggunakan beberapa unit pemrosesan. Konsep paralelisme, organisasi memori, interkoneksi, sinkronisasi, speedup, dan scalability menjadi dasar untuk memahami komputer multicore, server multiprosesor, dan sistem komputasi paralel modern.