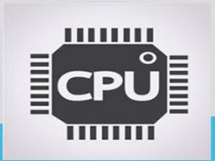
Arsitektur dan Organisasi Komputer
COM 60011

Bab #17 – Proses parallel
Bab #18 – Komputer Multicore

UNIVERSITAS BRAWIJAYA
UB FILKOM
FAKULTAS ILMU KOMPUTER

Bab 17
Proses parallel



Organisasi Prosesor Ganda

Instruksi tunggal, data tunggal (SISD) aliran

- Prosesor tunggal menjalankan aliran instruksi tunggal untuk beroperasi pada data yang disimpan dalam satu memori
- Uniprosesor termasuk dalam kategori ini

Beberapa instruksi, data tunggal (MISD) aliran

- Urutan data ditransmisikan ke sekumpulan prosesor, yang masing-masing menjalankan urutan instruksi yang berbeda
- Tidak diterapkan secara komersial

Instruksi tunggal, banyak data (SIMD) aliran

- Instruksi mesin tunggal mengontrol eksekusi simultan dari sejumlah elemen pemrosesan dengan basis lockstep
- Prosesor vektor dan array termasuk dalam kategori ini

Instruksi ganda, banyak data (MIMD) aliran

- Seperangkat prosesor secara bersamaan menjalankan urutan instruksi yang berbeda pada kumpulan data yang berbeda
- SMP, cluster, dan sistem NUMA cocok dengan kategori ini

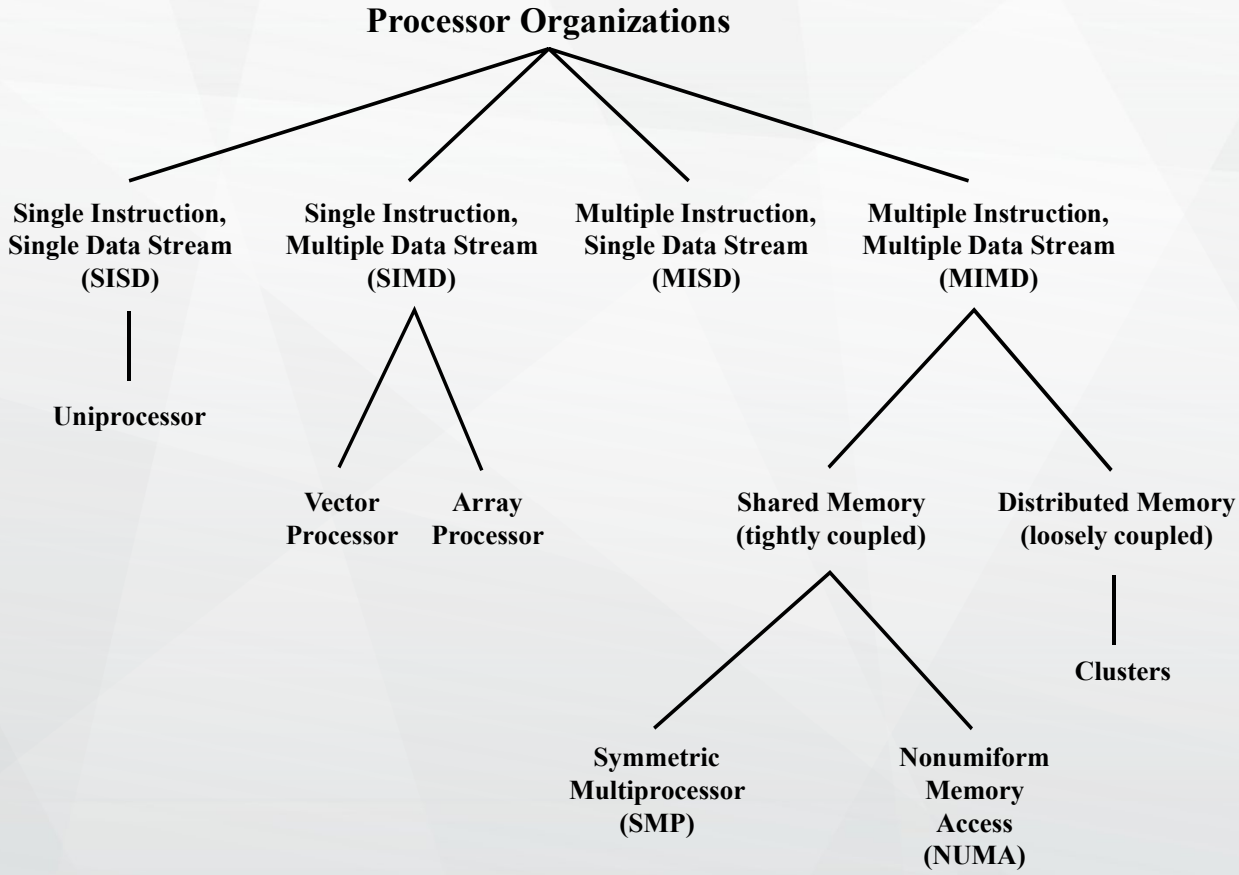
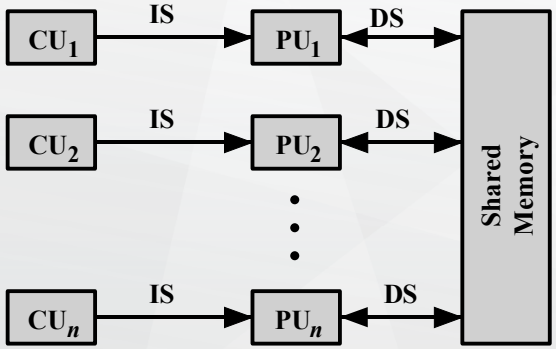
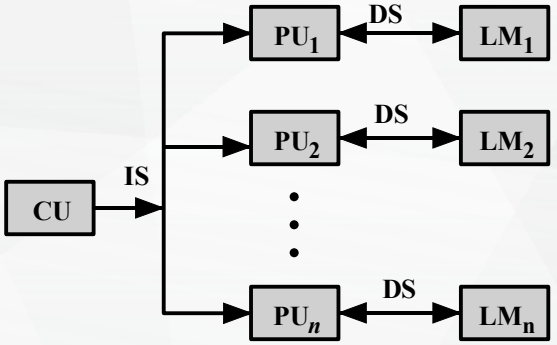


Figure 17.1 A Taxonomy of Parallel Processor Architectures

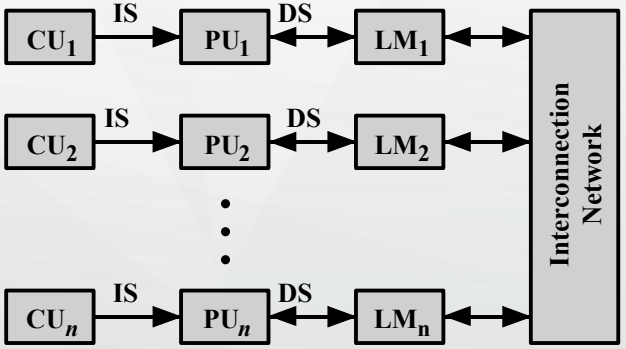
(a) SISD



(c) MIMD (with shared memory)



(b) SIMD (with distributed memory)



(d) MIMD (with distributed memory)

CU = control unit
IS = instruction stream
PU = processing unit
DS = data stream
MU = memory unit
LM = local memory

SISD = single instruction, single data stream
SIMD = single instruction, multiple data stream
MIMD = multiple instruction, multiple data stream

Figure 17.2 Alternative Computer Organizations

Simetris Multiprosesor (SMP)

Stand-alone computer dengan karakteristik berikut:

Two or more similar processors of comparable capacity	Processors share same memory and I/O facilities • Prosesor disambungkan oleh bus atau sambungan internal lainnya • Memory access time kurang lebih sama untuk setiap prosesor	All processors share access to I/O devices • Baik melalui saluran yang sama atau saluran yang berbeda memberikan jalur ke perangkat yang sama	Semua prosesor dapat melakukan fungsi yang samas (hence “symmetric”)	System controlled by integrated operating system • Menyediakan interaksi antara pemroses dan program mereka di tingkat pekerjaan, tugas, file, dan elemen data

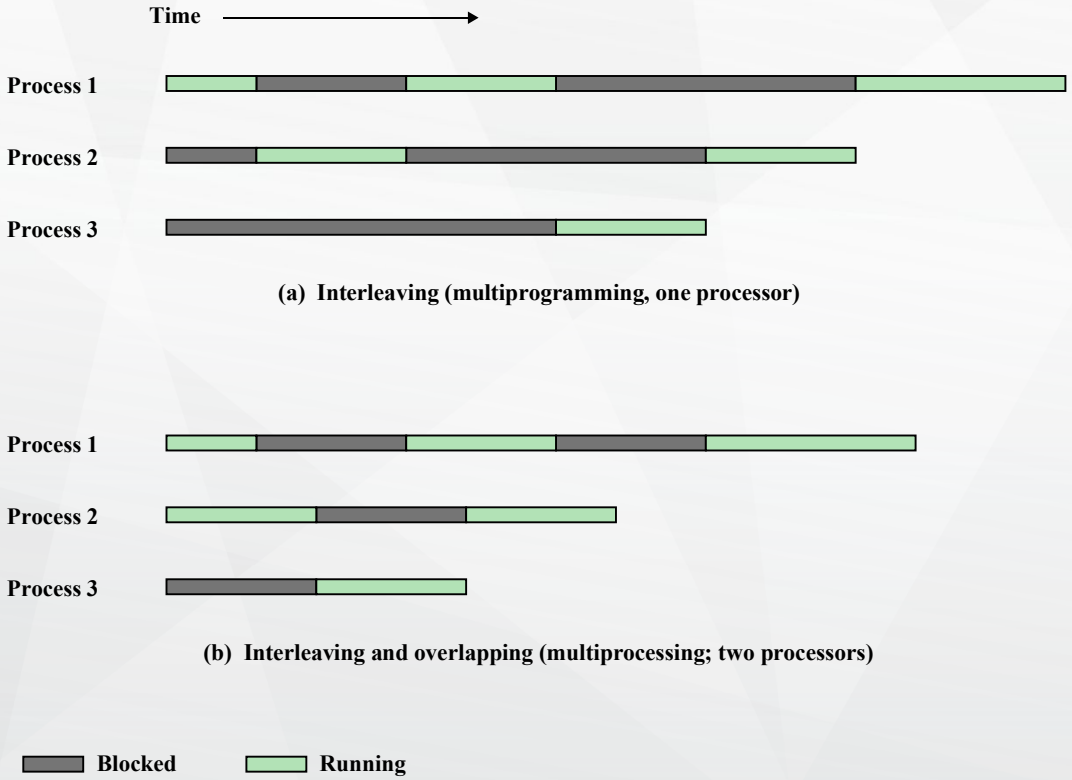


Figure 17.3 Multiprogramming and Multiprocessing

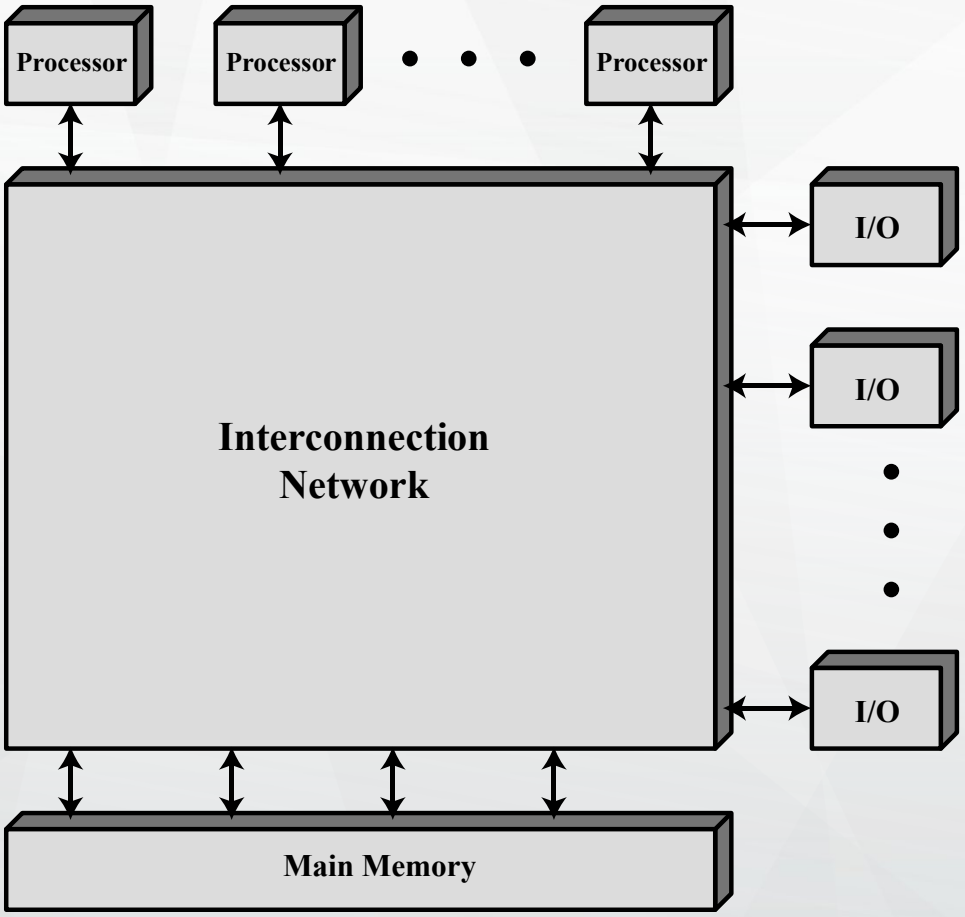


Figure 17.4 Generic Block Diagram of a Tightly Coupled Multiprocessor

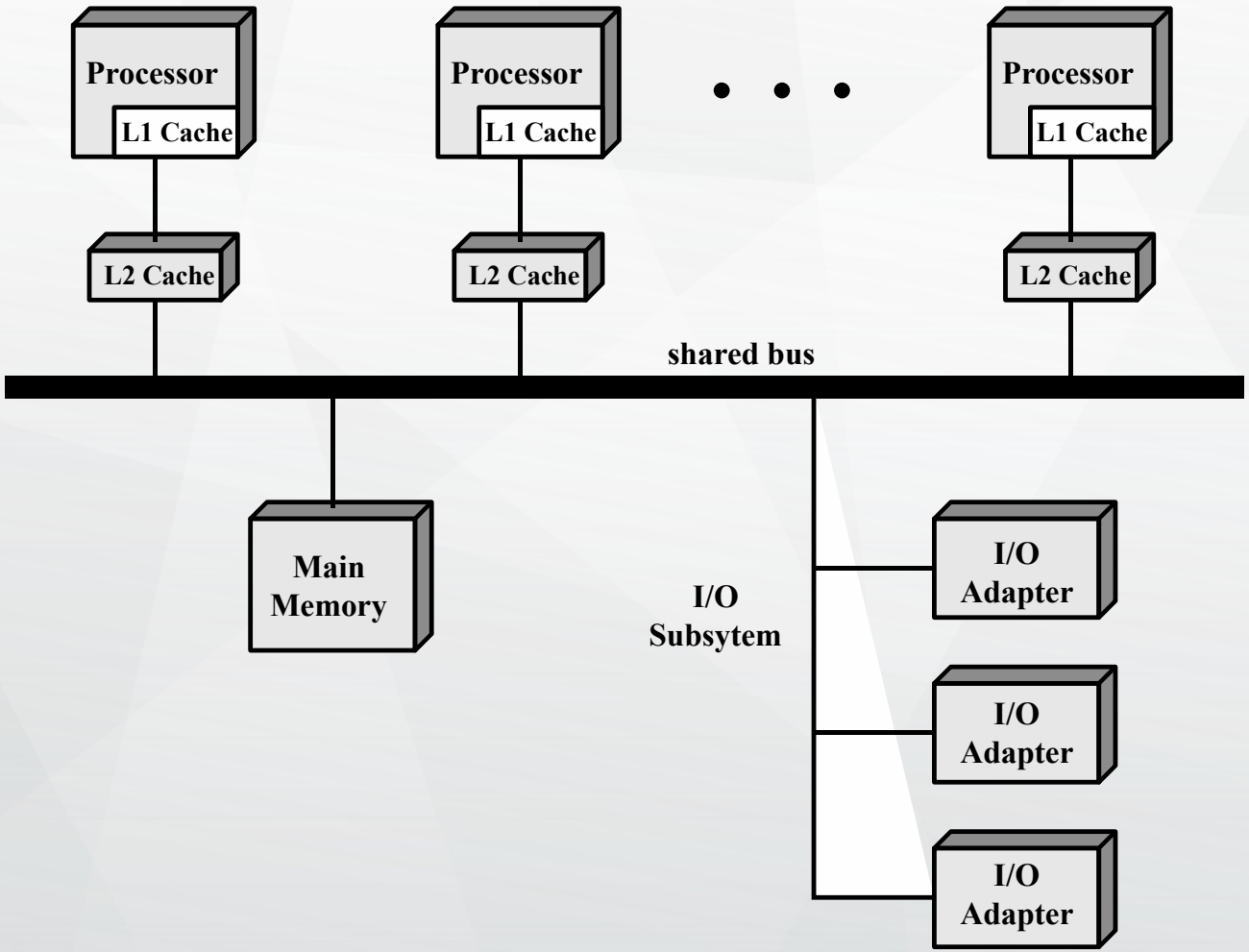


Figure 17.5 Symmetric Multiprocessor Organization

Organisasi bus memiliki beberapa fitur menarik:

Simplicity/Kesederhanaan

Pendekatan paling sederhana untuk organisasi multiprosesor

Flexibility/Fleksibilitas

Umumnya mudah untuk memperluas sistem dengan memasang lebih banyak prosesor ke bus

Reliability/Keandalan

Bus pada dasarnya adalah media pasif dan kegagalan perangkat yang terpasang tidak boleh menyebabkan kegagalan seluruh sistem

Kekurangan dari organisasi bus:

Kelemahan utama adalah kinerja

- Semua referensi memori melewati bus umum
- Performa terbatas dengan waktu siklus bus

Setiap prosesor harus memiliki memori cache

- Mengurangi jumlah akses bus

Mengarah ke masalah dengan *koherensi cache*

- Jika sebuah kata diubah dalam satu cache, kata tersebut dapat membuat kata tidak valid di cache lain
  - Untuk mencegah hal ini, prosesor lain harus diberitahu bahwa pembaruan telah terjadi
- Biasanya ditangani di perangkat keras daripada sistem operasi

Pertimbangan Desain Sistem Operasi Multiprosesor

Bersamaan secara bersamaan proses

- Rutinitas OS harus reentrant untuk memungkinkan beberapa prosesor menjalankan kode IS yang sama secara bersamaan
- Tabel OS dan struktur manajemen harus dikelola dengan baik untuk menghindari kebuntuan atau operasi yang tidak valid

Penjadwalan

- Prosesor apapun dapat melakukan penjadwalan sehingga konflik harus dihindari
- Penjadwal harus menetapkan proses yang siap untuk prosesor yang tersedia

Sinkronisasi

- Dengan beberapa proses aktif yang memiliki akses potensial ke ruang alamat bersama atau sumber daya I / O, kehati-hatian harus diberikan untuk menyediakan sinkronisasi yang efektif
- Sinkronisasi adalah fasilitas yang memberlakukan pengecualian timbal balik dan urutan acara

Penyimpanan pengelolaan

- Selain menangani semua masalah yang ditemukan pada mesin uniprocessor, OS perlu memanfaatkan paralelisme perangkat keras yang tersedia untuk mencapai kinerja terbaik.
- Mekanisme paging pada prosesor yang berbeda harus dikoordinasikan untuk menegakkan konsistensi saat beberapa prosesor berbagi halaman atau segmen dan untuk memutuskan penggantian halaman

Keandalan dan kesalahan toleransi

- OS harus memberikan degradasi yang baik saat menghadapi kegagalan prosesor
- Penjadwal dan bagian lain dari sistem operasi harus mengenali hilangnya prosesor dan merestrukturisasi yang sesuai

Koherensi Cache

Solusi Perangkat Lunak

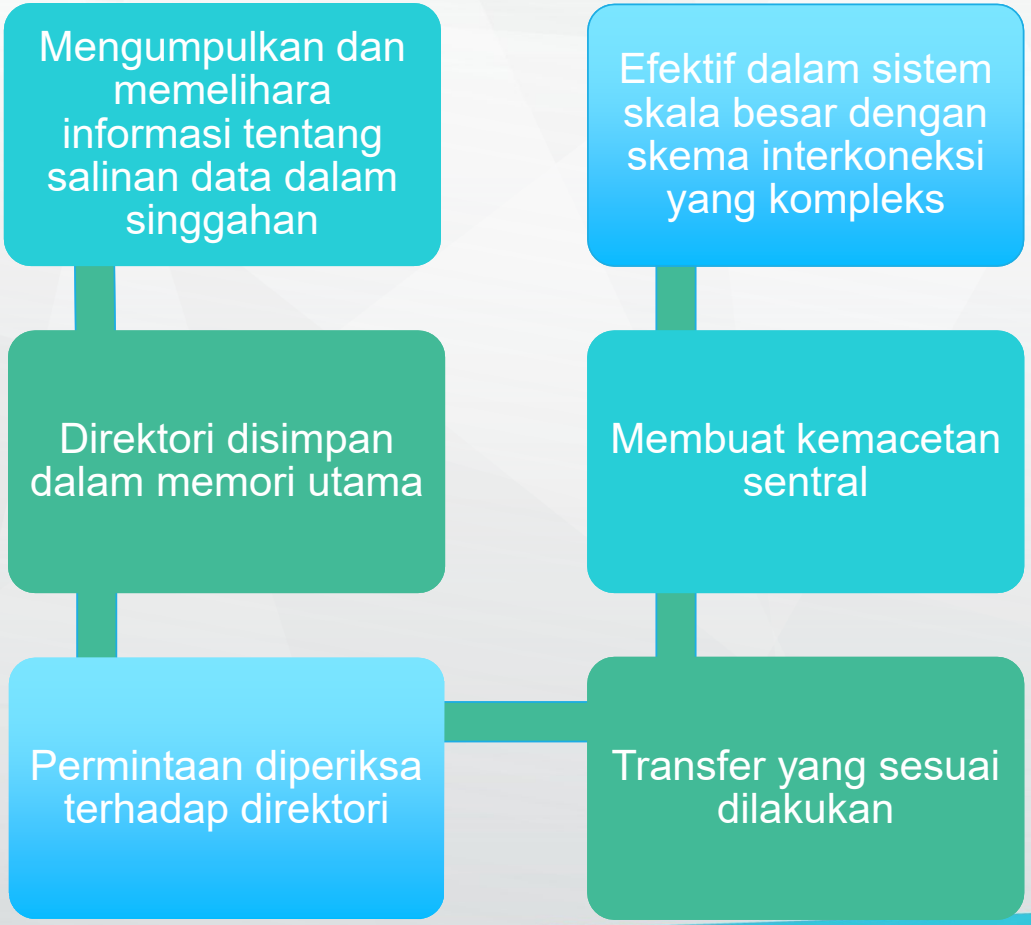
- **Mencoba menghindari kebutuhan akan sirkuit dan logika perangkat keras tambahan dengan mengandalkan kompiler dan sistem operasi untuk mengatasi masalah tersebut**
- **Menarik karena overhead untuk mendeteksi potensi masalah ditransfer dari waktu proses ke waktu kompilasi, dan kompleksitas desain ditransfer dari perangkat keras ke perangkat lunak**
  - Namun, pendekatan perangkat lunak waktu kompilasi umumnya harus membuat keputusan konservatif, yang mengarah pada pemanfaatan cache yang tidak efisien

Koherensi Cache

Solusi Berbasis Perangkat Keras

- Umumnya disebut sebagai protokol koherensi cache
- Solusi ini memberikan pengenalan dinamis pada saat menjalankan kondisi potensi ketidakkonsistenan
- Karena masalah hanya ditangani ketika benar-benar muncul, ada penggunaan cache yang lebih efektif, yang mengarah pada peningkatan kinerja melalui pendekatan perangkat lunak
- Pendekatan transparan bagi pemrogram dan kompiler, mengurangi beban pengembangan perangkat lunak
- Dapat dibagi menjadi dua kategori:
  - Protokol direktori
  - Protokol snoopy

Protokol Direktori



Protokol Snoopy

Mendistribusikan tanggung jawab untuk memelihara koherensi cache di antara semua pengontrol cache dalam multiprosesor

- Cache harus mengenali ketika sebuah baris yang dimilikinya dibagikan dengan cache lain
- Ketika pembaruan dilakukan pada baris cache bersama, itu harus diumumkan kepada orang lain cache dengan mekanisme siaran
- Setiap pengontrol cache dapat "mengintip" di jaringan untuk mengamati pemberitahuan siaran ini dan bereaksi sesuai dengannya

Cocok untuk multiprosesor berbasis bus karena bus bersama menyediakan sarana sederhana untuk penyiaran dan pengintaian

- Perhatian harus diberikan agar peningkatan lalu lintas bus yang diperlukan untuk penyiaran dan pengintaian tidak membatalkan keuntungan dari penggunaan cache lokal

Dua pendekatan dasar telah dieksplorasi:

- Tulis tidak valid
- Tulis pembaruan (atau tulis siaran)

Write Invalidate

- Banyak pembaca, tetapi hanya satu penulis dalam satu waktu
- Saat menulis diperlukan, semua cache lain dari baris tersebut menjadi tidak valid
- Prosesor penulisan kemudian memiliki akses eksklusif (murah) hingga lini Dibutuhkan oleh prosesor lain
- Paling banyak digunakan dalam sistem multiprosesor komersial seperti arsitektur x86
- Status setiap baris ditandai sebagai diubah, eksklusif, dibagikan, atau tidak valid
  - Untuk alasan inilah protokol write-invalidate dipanggil *MESI*

Write Update

Bisa beberapa pembaca dan penulis

Ketika prosesor ingin memperbarui baris bersama kata yang akan diperbarui didistribusikan ke semua orang lain dan cache yang berisi baris itu dapat memperbaruinya

Beberapa sistem menggunakan campuran adaptif dari mekanisme write-invalidate dan write-update

Protokol MESI

Untuk memberikan konsistensi cache pada SMP, cache data mendukung protokol yang dikenal sebagai MESI:

Modified /Diubah

Baris di cache telah diubah dan hanya tersedia di cache ini

Exclusive /Eksklusif

Baris di cache sama dengan baris di memori utama dan tidak ada di cache lain

Shared /Bersama

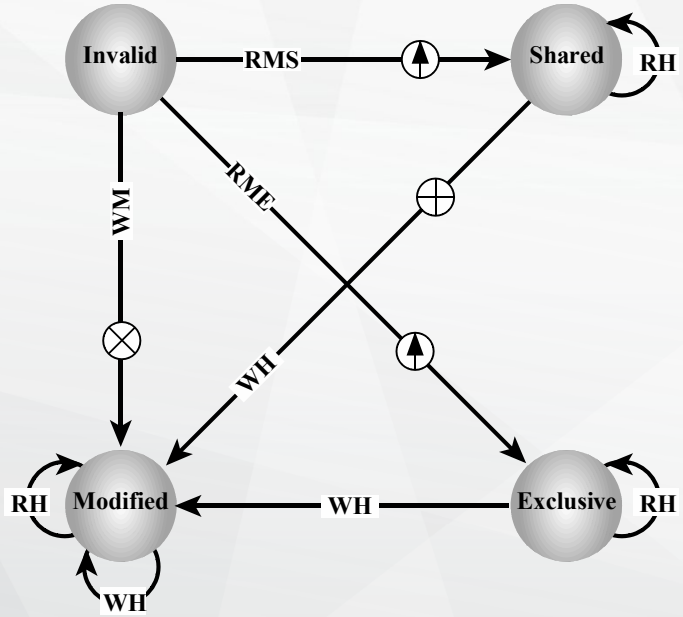
Baris di cache sama dengan baris di memori utama dan mungkin ada di cache lain

Invalid /Tidak valid

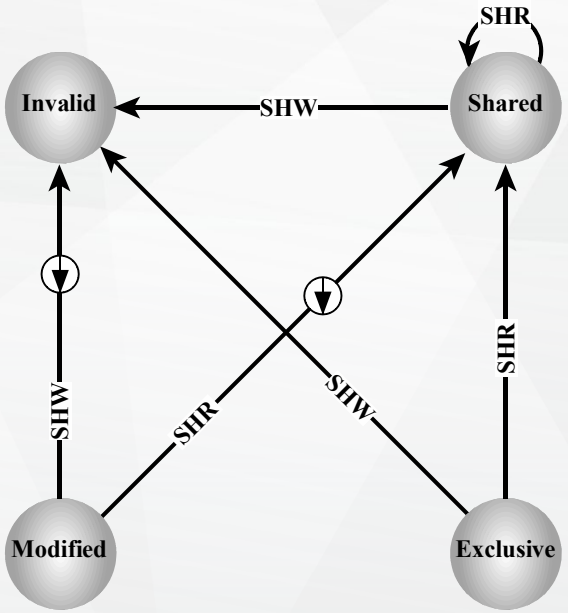
Baris di cache tidak berisi data yang valid

Tabel 17.1
MESI Status Baris Cache

	M **Modified**	**E** **Exclusive**	**S** **Shared**	**I** **Invalid**
This cache line valid?	Yes	Yes	Yes	No
The memory copy is…	out of date	valid	valid	—
Copies exist in other caches?	No	No	Maybe	Maybe
A write to this line…	does not go to bus	does not go to bus	goes to bus and updates cache	goes directly to bus



(a) Line in cache at initiating processor



(b) Line in snooping cache

RH	Read hit
RMS	Read miss, shared
RME	Read miss, exclusive
WH	Write hit
WM	Write miss
SHR	Snoop hit on read
SHW	Snoop hit on write or read-with-intent-to-modify

Dirty line copyback

Invalidate transaction

Read-with-intent-to-modify

Cache line fill

Figure 17.6 MESI State Transition Diagram

Multithreading dan Multiprosesor Chip

Kinerja prosesor dapat diukur dengan kecepatan menjalankan instruksi

Tingkat MIPS = f * IPC

- f = prosesor frekuensi clock, dalam MHz
- IPC = instruksi rata-rata per siklus

Tingkatkan kinerja dengan meningkatkan frekuensi jam dan menambah instruksi yang diselesaikan selama siklus

Multithreading

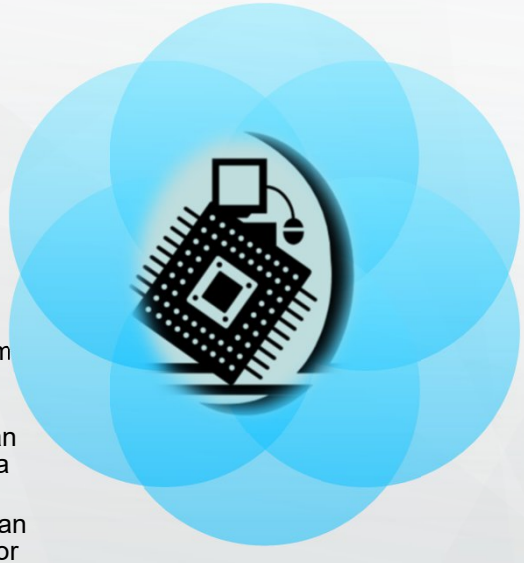
- Memungkinkan tingkat paralelisme instruksi yang tinggi tanpa meningkatkan kompleksitas sirkuit atau konsumsi daya
- Aliran instruksi dibagi menjadi beberapa aliran kecil, yang dikenal sebagai utas, yang dapat dijalankan secara paralel

Definisi Thread dan Proses

Thread in multithreaded processors may or may not be the same as the concept of software threads in a multiprogrammed operating system

Thread switch

- Tindakan beralih kontrol prosesor antar thread dalam proses yang sama
- Biasanya lebih murah daripada sakelar proses



Thread berkaitan dengan penjadwalan dan eksekusi, sedangkan proses berkaitan dengan penjadwalan / eksekusi dan kepemilikan sumber daya dan sumber daya

Thread:

- Unit kerja yang dapat dikirim dalam proses
- Termasuk konteks prosesor (yang mencakup penghitung program dan penunjuk tumpukan) dan area data untuk tumpukan
- Mengeksekusi secara berurutan dan dapat diinterupsi sehingga prosesor dapat beralih ke thread lain

Process:

- Contoh program yang berjalan di komputer
- Dua karakteristik utama:
  - Kepemilikan sumber daya
  - Penjadwalan/eksekusi

Process switch

- Operasi yang mengalihkan prosesor dari satu proses ke proses lain dengan menyimpan semua data kontrol proses, register, dan informasi lain untuk yang pertama dan menggantinya dengan informasi proses untuk yang kedua

Multithreading implisit dan eksplisit

- **Semua prosesor komersial dan yang paling eksperimental menggunakan eksplisit multithreading**
  - Bersamaan menjalankan instruksi dari utas eksplisit yang berbeda
  - Interleave instruksi dari thread berbeda pada pipeline bersama atau eksekusi paralel pada pipeline paralel
- **Multithreading implisit adalah eksekusi bersamaan dari beberapa utas yang diekstrak dari program sekuensial tunggal**
  - Utas implisit didefinisikan secara statis oleh kompiler atau secara dinamis oleh perangkat keras

Pendekatan Multithreading Eksplisit

Disisipkan

- Berbutir halus
- Prosesor berurusan dengan dua atau lebih konteks utas sekaligus
- Mengalihkan utas di setiap siklus jam
- Jika utas diblokir, itu akan dilewati

Serentak (SMT)

- Instruksi secara bersamaan dikeluarkan dari beberapa utas ke unit eksekusi superscalar prosesor

Diblokir

- Berbutir kasar
- Utas dijalankan sampai peristiwa menyebabkan penundaan
- Efektif pada prosesor berurutan
- Menghindari kemacetan pipa

Multiprocessing chip

- Prosesor direplikasi pada satu chip
- Setiap prosesor menangani utas terpisah
- Keuntungannya adalah area logika yang tersedia pada sebuah chip digunakan secara efektif

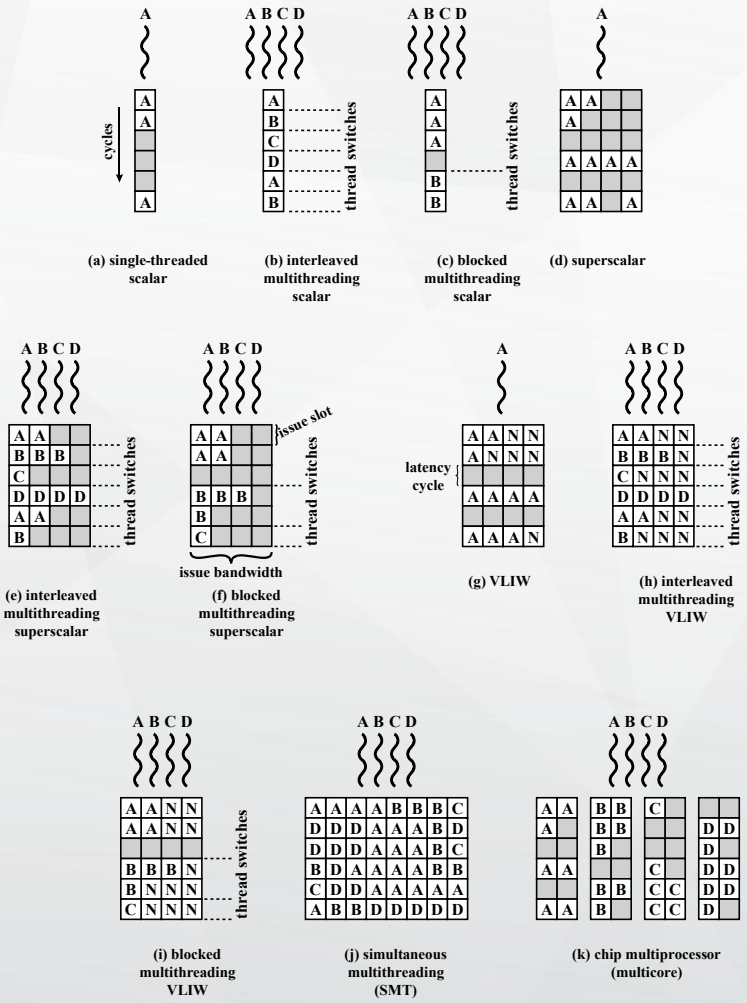
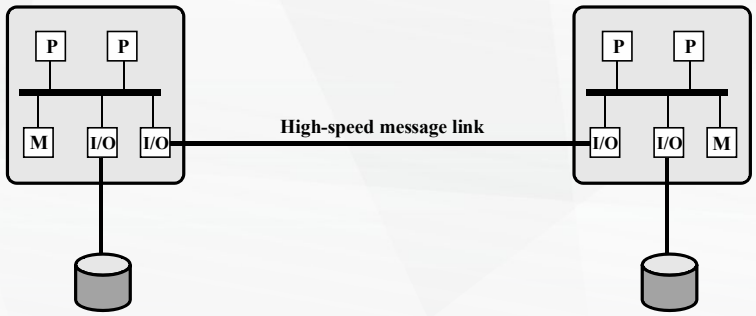


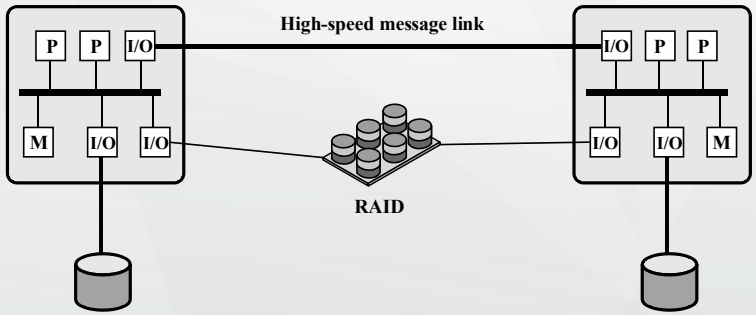
Figure 17.7 Approaches to Executing Multiple Threads

Kluster

- **Alternatif untuk SMP sebagai pendekatan untuk memberikan kinerja yang tinggi dan tinggi ketersediaan**
- **Sangat menarik untuk aplikasi server**
- **Didefinisikan sebagai:**
  - SEBUAH sekelompok keseluruhan yang saling berhubungan komputer bekerja bersama sebagai sumber daya komputasi terpadu yang dapat menciptakan ilusi menjadi satu mesin
  - (Syarat *seluruh komputer* berarti sistem yang dapat berjalan sendiri, selain dari cluster)
- **Setiap komputer dalam sebuah cluster disebut Sebuah simpul**
- **Manfaat:**
  - Skalabilitas mutlak
  - Skalabilitas tambahan
  - Ketersediaan tinggi
  - Harga / kinerja yang unggul



(a) Standby server with no shared disk



(b) Shared disk

Figure 17.8 Cluster Configurations

Tabel 17.2
Kekelompokan Metode: Manfaat dan Batasan

Clustering Method	Description	Benefits	Limitations
Passive Standby	A secondary server takes over in case of primary server failure.	Easy to implement.	High cost because the secondary server is unavailable for other processing tasks.
Active Secondary:	The secondary server is also used for processing tasks.	Reduced cost because secondary servers can be used for processing.	Increased complexity.
Separate Servers	Separate servers have their own disks. Data is continuously copied from primary to secondary server.	High availability.	High network and server overhead due to copying operations.
Servers Connected to Disks	Servers are cabled to the same disks, but each server owns its disks. If one server fails, its disks are taken over by the other server.	Reduced network and server overhead due to elimination of copying operations.	Usually requires disk mirroring or RAID technology to compensate for risk of disk failure.
Servers Share Disks	Multiple servers simultaneously share access to disks.	Low network and server overhead. Reduced risk of downtime caused by disk failure.	Requires lock manager software. Usually used with disk mirroring or RAID technology.

Pengoperasian Sistem Masalah Desain

Bagaimana kegagalan dikelola bergantung pada metode pengelompokan yang digunakan

Dua pendekatan:

- Cluster yang sangat tersedia
- Kesalahan cluster yang toleran

Failover

- Fungsi mengalihkan aplikasi dan sumber data dari sistem yang gagal ke sistem alternatif di cluster

Failback

- Pemulihan aplikasi dan sumber data ke sistem asli setelah diperbaiki

Penyeimbang beban

- Skalabilitas tambahan
- Secara otomatis menyertakan komputer baru dalam penjadwalan
- Kebutuhan Middleware untuk mengakui bahwa proses dapat beralih antar mesin

Paralelisasi Komputasi

Penggunaan kluster yang efektif memerlukan mengeksekusi perangkat lunak dari satu aplikasi secara paralel

Tiga pendekatan adalah:

Parallelizing complier

- Menentukan pada waktu kompilasi bagian mana dari aplikasi yang dapat dieksekusi secara paralel
- Ini kemudian dipisahkan untuk ditugaskan ke komputer yang berbeda dalam kluster

Parallelized application

- Aplikasi yang ditulis sejak awal untuk berjalan pada kluster dan menggunakan passing pesan untuk memindahkan data antar node kluster

Parametric computing

- Dapat digunakan jika esensi aplikasi adalah algoritma atau program yang harus dijalankan dalam jumlah besar, setiap kali dengan serangkaian kondisi awal atau parameter yang berbeda

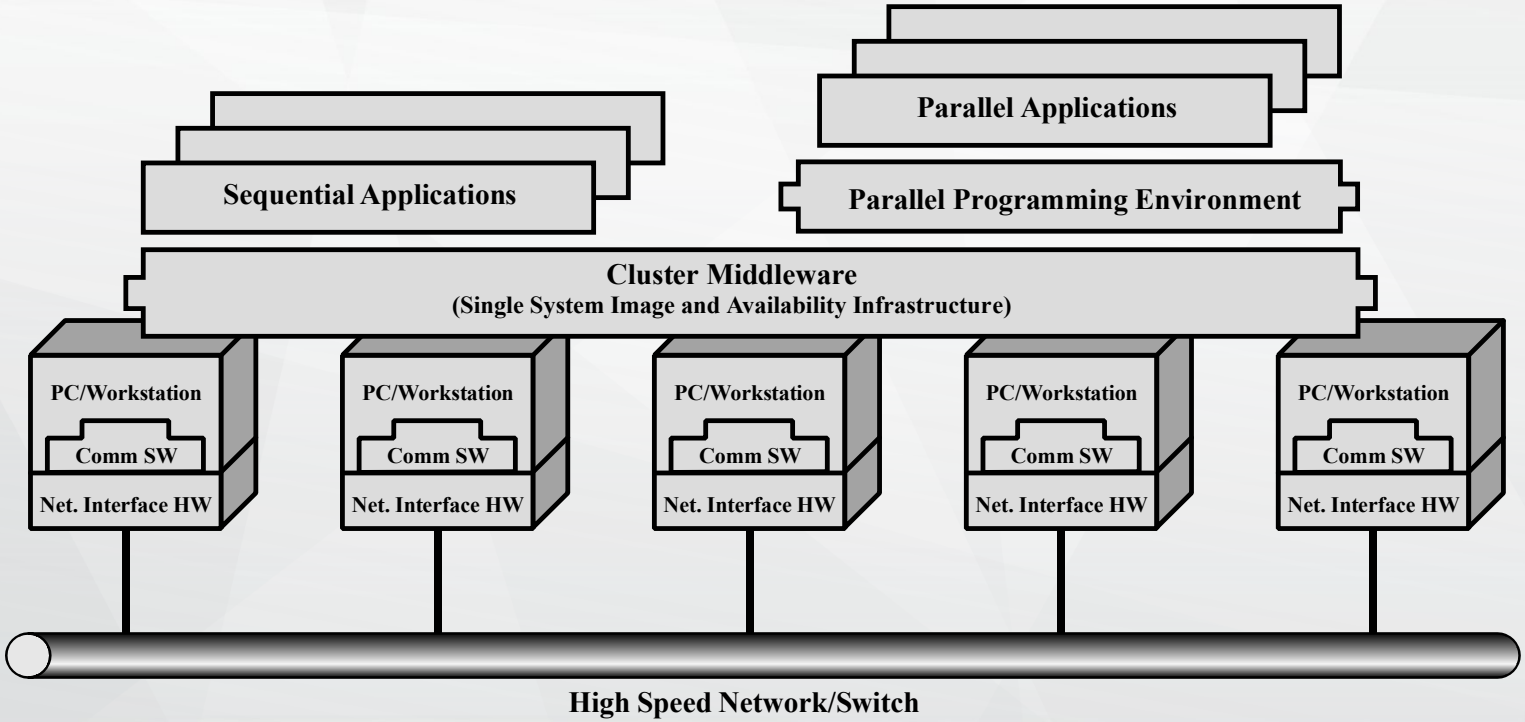


Figure 17.9 Cluster Computer Architecture

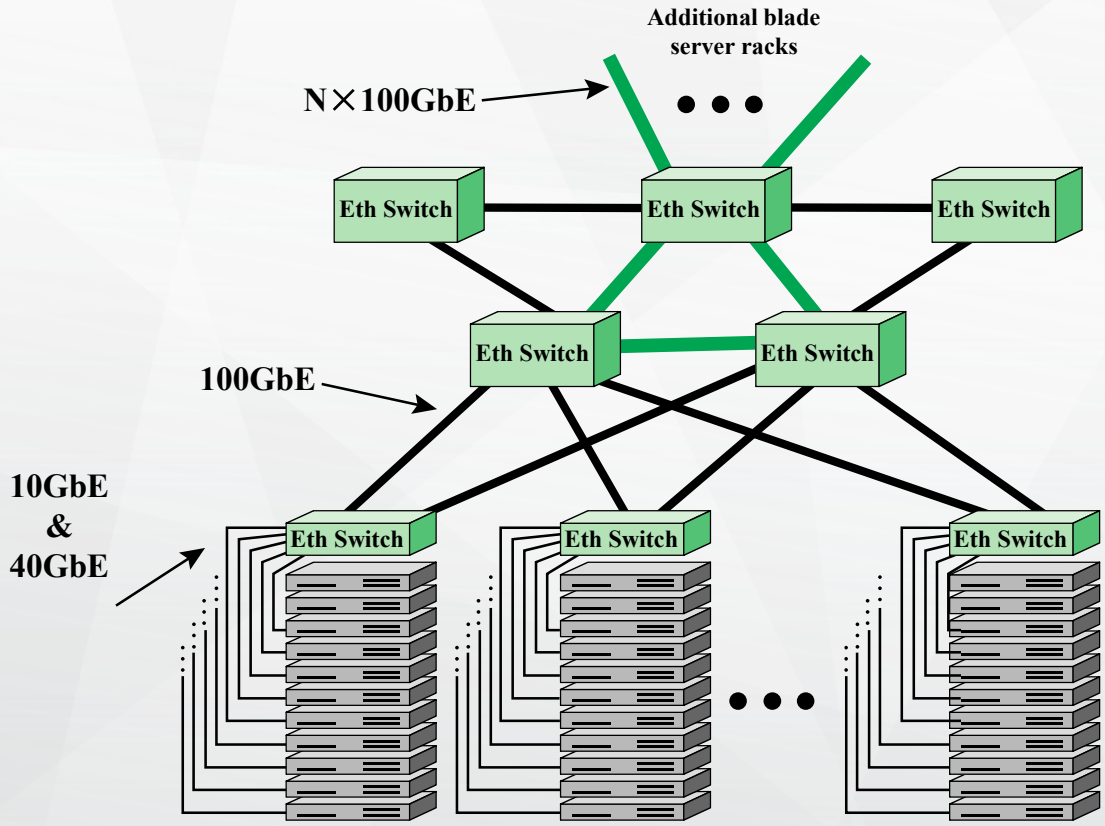
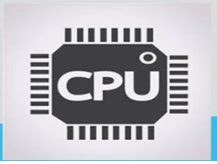


Figure 17.10 Example 100-Gbps Ethernet Configuration for Massive Blade Server Cloud Site

Cluster Dibandingkan dengan SMP

- Keduanya menyediakan konfigurasi dengan banyak prosesor untuk mendukung aplikasi permintaan tinggi
- Kedua solusi tersebut tersedia secara komersial

SMP	Kekelompokan
Lebih mudah untuk mengelola dan mengkonfigurasi	**Jauh lebih unggul dalam hal skalabilitas inkremental dan absolut**
Jauh lebih mirip dengan model prosesor tunggal asli yang hampir semua aplikasi ditulis	**Unggul dalam hal ketersediaan**
Lebih sedikit ruang fisik dan konsumsi daya yang lebih rendah	**Semua komponen sistem dapat dibuat sangat redundan**
Mapan dan stabil	



Akses Memori Nonuniform (NUMA)

Alternatif untuk SMP dan kekelompokan

Memori seragam akses (UMA)

- Semua prosesor memiliki akses ke semua bagian memori utama menggunakan beban dan penyimpanan
- Mengakses waktu untuk semua wilayah memori adalah sama
- Akses waktu ke memori untuk prosesor yang berbeda adalah sama

Tidak seragam Penyimpanan akses (NUMA)

- Semua prosesor memiliki akses ke semua bagian memori utama menggunakan beban dan penyimpanan
- Mengakses waktu prosesor berbeda tergantung pada wilayah mana dari memori utama sedang diakses
- Prosesor yang berbeda mengakses berbagai wilayah memori dengan kecepatan berbeda

NUMA koheren-cache (CC-NUMA)

- Sistem NUMA di mana koherensi cache dipertahankan di antara cache dari berbagai prosesor

Motivasi

SMP memiliki batas praktis terhadap jumlah prosesor yang dapat digunakan

- Batas lalu lintas bus hingga antara 16 dan 64 prosesor

Dalam kluster setiap node memiliki memori utama pribadinya sendiri

- Aplikasi tidak melihat memori global yang besar
- Koherensi dikelola oleh perangkat lunak daripada perangkat keras

NUMA mempertahankan rasa SMP sekaligus memberikan multiproses skala besar

Tujuan dengan NUMA adalah untuk mempertahankan memori sistem yang transparan sambil mengizinkan beberapa node multiprosesor, masing-masing dengan bus sendiri atau sistem interkoneksi internal

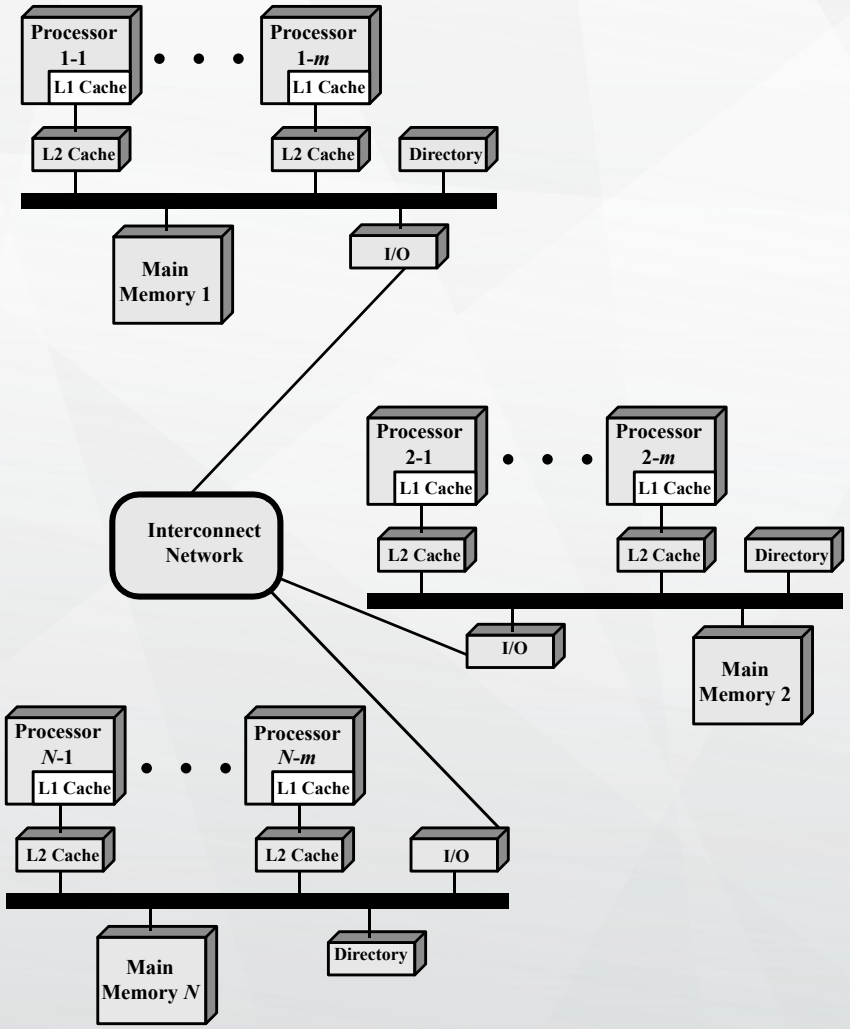


Figure 17.11 CC-NUMA Organization

NUMA Pro dan Kontra

- Keuntungan utama dari sistem CC-NUMA adalah dapat memberikan kinerja yang efektif pada tingkat paralelisme yang lebih tinggi daripada SMP tanpa memerlukan perubahan perangkat lunak
- Lalu lintas bus pada setiap node terbatas pada permintaan yang dapat ditangani bus
- Jika banyak dari akses memori ke node jauh, kinerja mulai rusak
- Tidak secara transparan terlihat seperti SMP
- Perubahan perangkat lunak akan diperlukan untuk memindahkan sistem operasi dan aplikasi dari SMP ke sistem CC-NUMA
- Peduli dengan ketersediaan

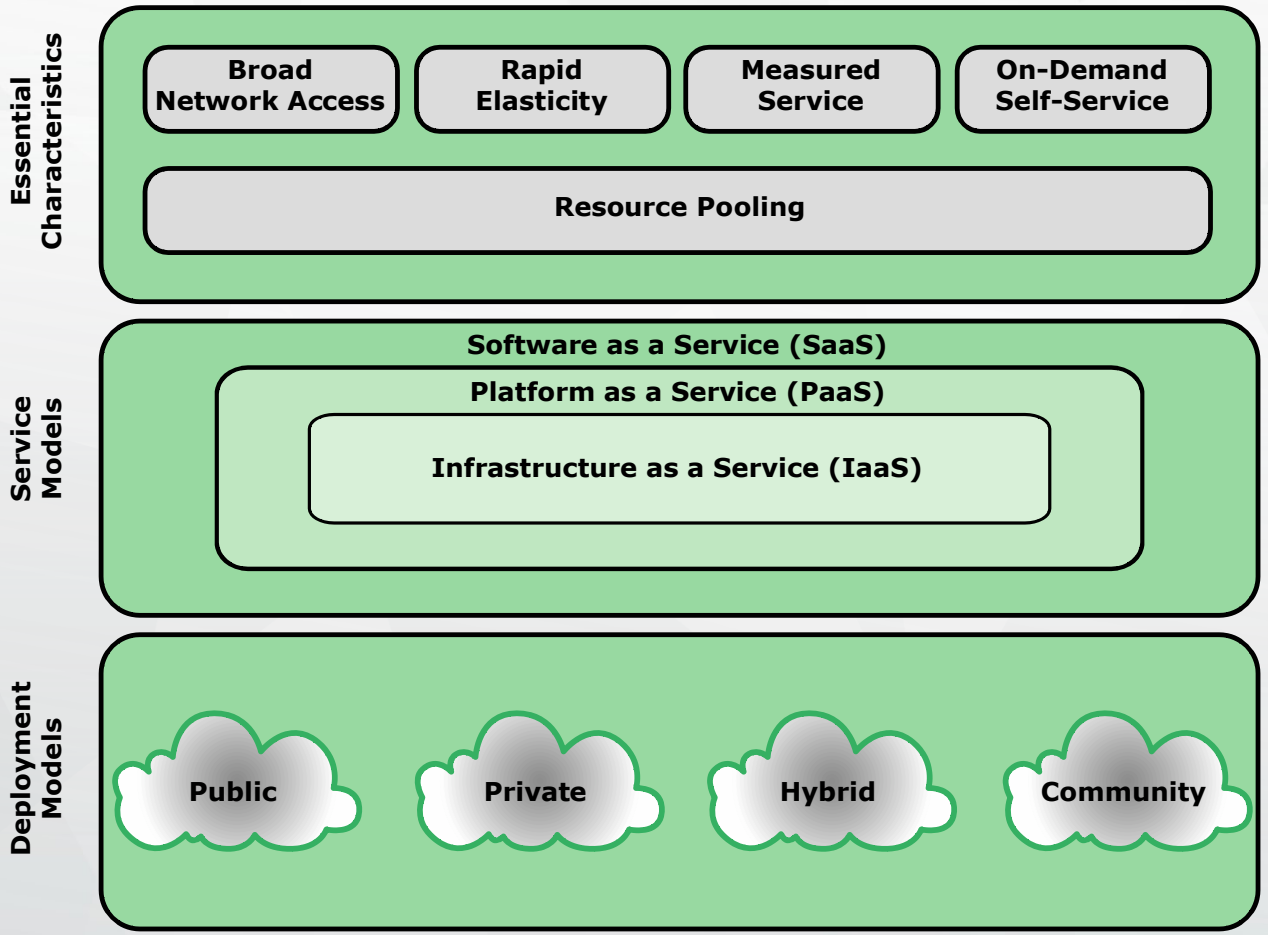


Figure 17.12 Cloud Computing Elements

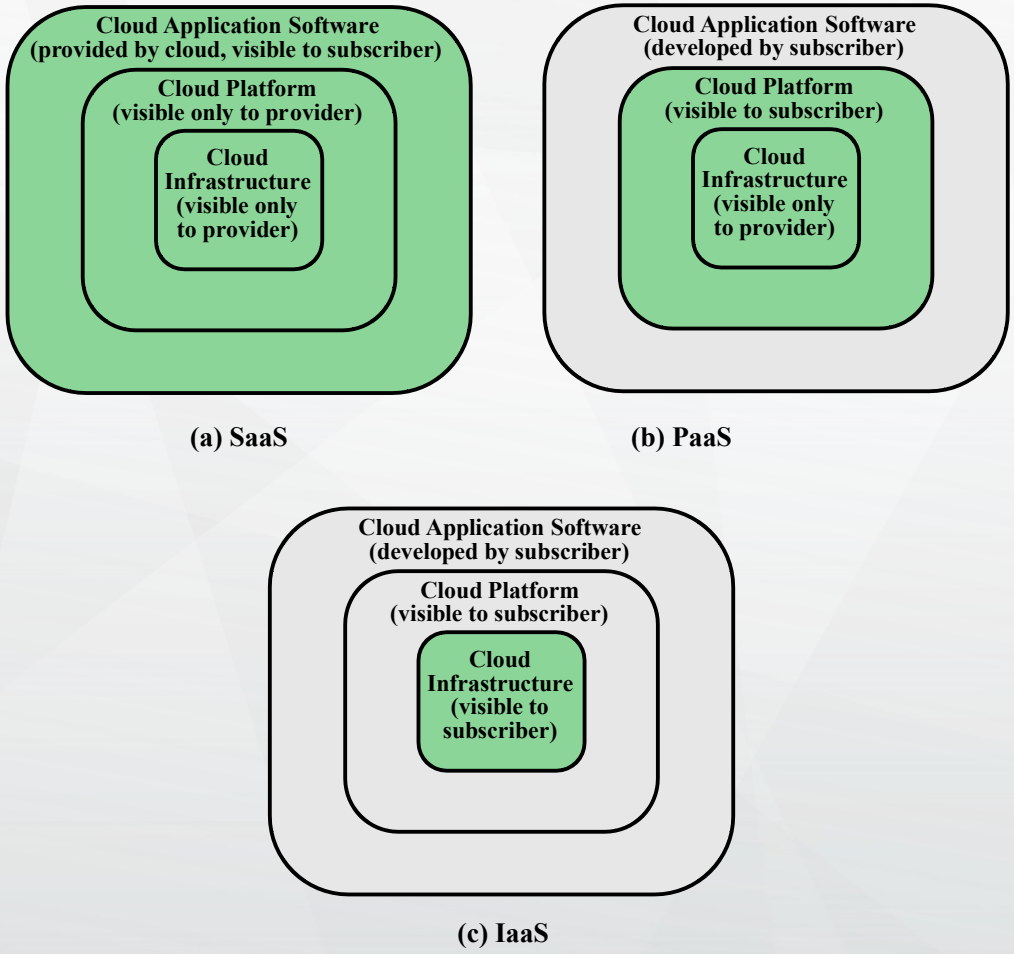


Figure 17.13 Cloud Service Models

Model Penerapan

Cloud publik

- Infrastruktur cloud tersedia untuk masyarakat umum atau grup industri besar dan dimiliki oleh organisasi yang menjual layanan cloud
- Keuntungan utama adalah biaya

Cloud pribadi

- Infrastruktur cloud yang diimplementasikan dalam lingkungan TI internal organisasi
- Motivasi utama untuk memilih cloud pribadi adalah keamanan

Cloud komunitas

- Seperti cloud pribadi, ini tidak terbuka untuk pelanggan mana pun
- Seperti cloud publik, sumber daya dibagi di antara sejumlah independen orgaizations

Awan hibrida

- Infrastruktur cloud adalah komposisi dari dua atau lebih cloud yang tetap merupakan entitas unik tetapi terikat bersama oleh teknologi standar atau kepemilikan yang memungkinkan portabilitas data dan aplikasi
- Informasi sensitif dapat ditempatkan di area pribadi cloud dan data yang kurang sensitif dapat memanfaatkan keuntungan biaya dari cloud publik

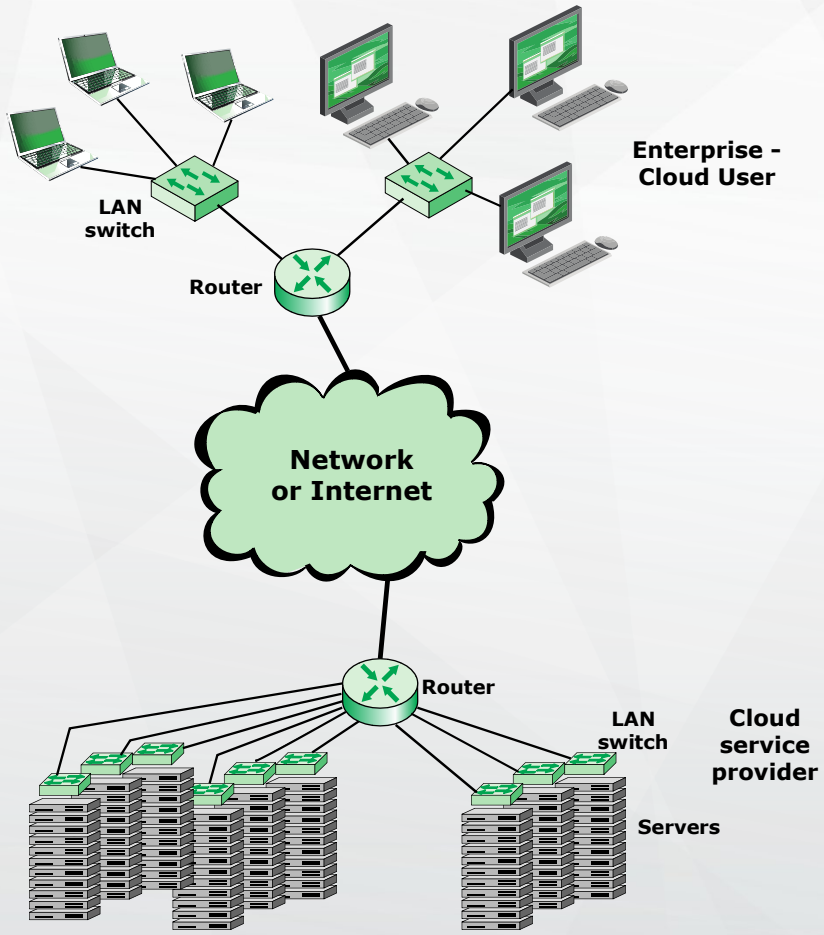


Figure 17.14 Cloud Computing Context

Arsitektur Referensi Komputasi Awan

NIST SP 500-292 menetapkan arsitektur referensi, yang dijelaskan sebagai:

"Itu Arsitektur referensi komputasi awan NIST berfokus pada persyaratan dari "apa yang "disediakan oleh layanan cloud, bukan" cara "merancang solusi dan implementasi. Referensi arsitektur dimaksudkan untuk memfasilitasi pemahaman tentang seluk-beluk operasional dalam komputasi awan. Ini tidak mewakili arsitektur sistem aspesifik
sistem komputasi awan; alih-alih ini adalah alat untuk mendeskripsikan, mendiskusikan, dan mengembangkan sistem khusus
Arsitektur menggunakan kerangka acuan umum. ”

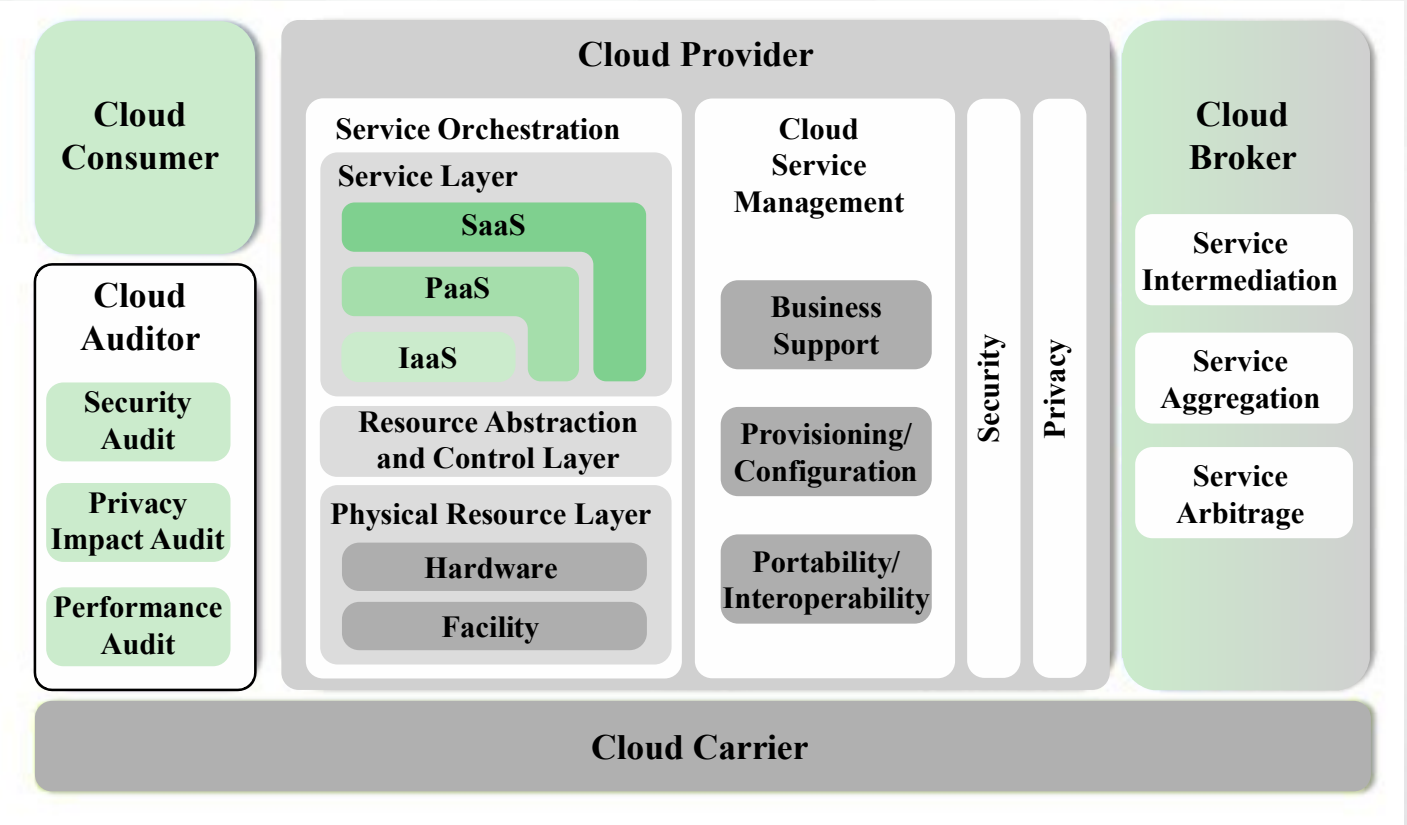


Figure 17.15 NIST Cloud Computing Reference Architecture

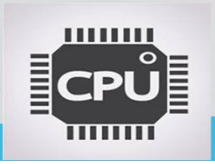
Ringkasan

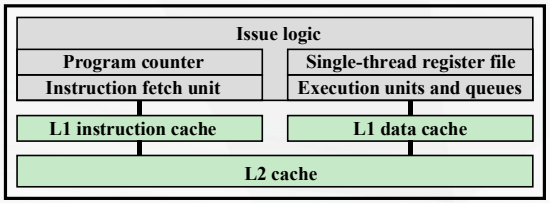
Bab 17

Paralel Pengolahan

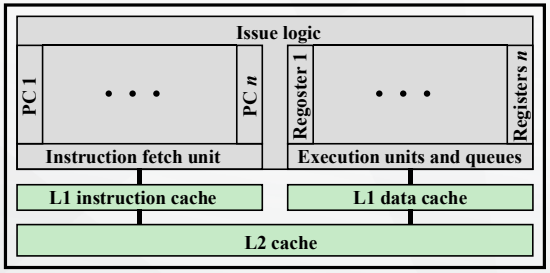
- Beberapa organisasi prosesor
  - Jenis sistem prosesor paralel
  - Organisasi paralel
- Multiprosesor simetris
  - Organisasi
  - Pertimbangan desain sistem operasi multiprosesor
- Koherensi cache dan protokol MESI
  - Solusi perangkat lunak
  - Solusi perangkat keras
  - Protokol MESI
- Multithreading dan multiprosesor chip
  - Multithreading implisit dan eksplisit
  - Pendekatan multithreading eksplisit
- Kluster
  - Konfigurasi cluster
  - Masalah desain sistem operasi
  - Arsitektur komputer cluster
  - Server blade
  - Cluster dibandingkan dengan SMP
- Akses memori tidak seragam
  - Motivasi
  - Organisasi
  - NUMA Pro dan kontra
- Komputasi awan
  - Elemen komputasi awan
  - Referensi komputasi awan Arsitektur

Bab 18
Komputer Multicore

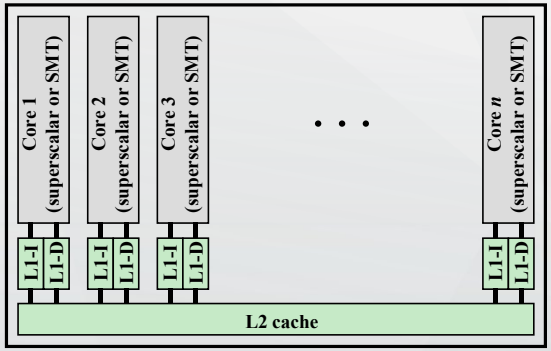




(a) Superscalar

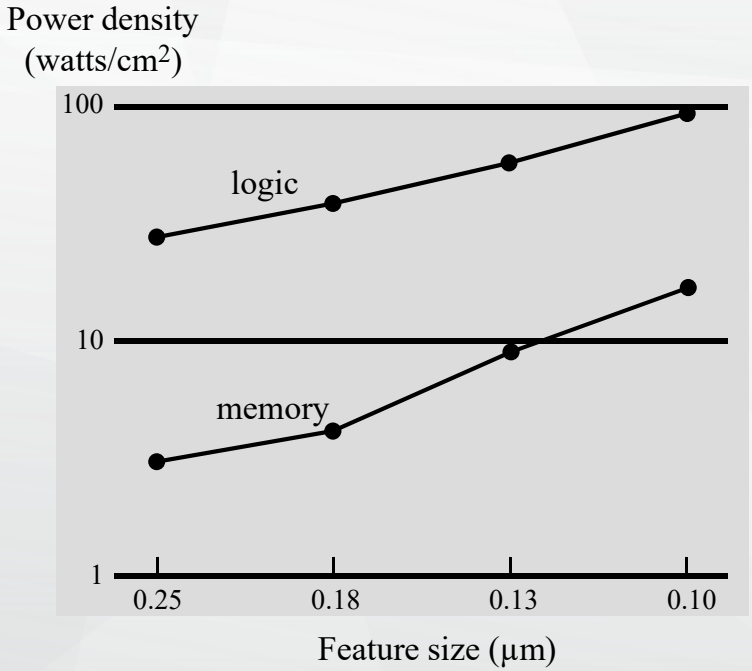


(b) Simultaneous multithr eading



(c) Multicore

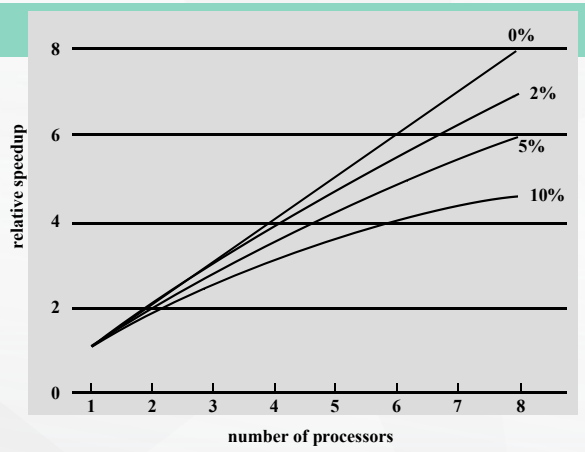
Figure 18.1 Alternative Chip Organizations



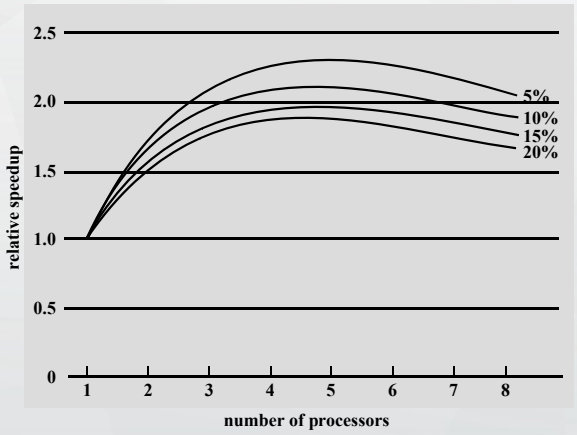
Kekuasaan

Penyimpanan

Figure 18.2 Power and Memory Considerations



(a) Speedup with 0%, 2%, 5%, and 10% sequential portions



(b) Speedup with overheads

Figure 18.3 Performance Effect of Multiple Cores

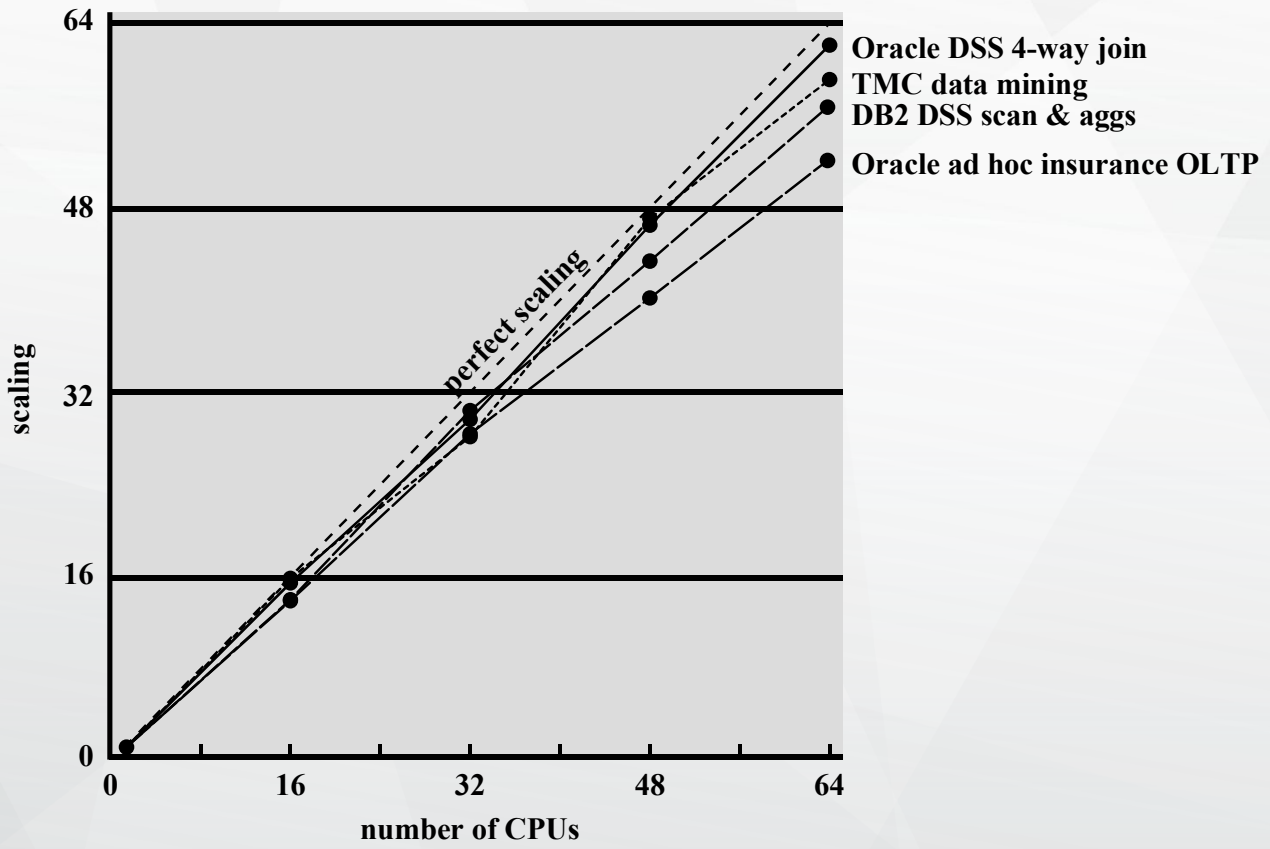


Figure 18.4 Scaling of Database Workloads on Multiple-Processor Hardware

Aplikasi Efektif untuk Prosesor Multicore

Multiasli berulir aplikasi

- Paralelisme tingkat benang
- Ditandai dengan memiliki sejumlah kecil proses yang sangat berulir

Multi-proses aplikasi

- Paralelisme tingkat proses
- Ditandai dengan adanya banyak proses single-threaded

Jawa aplikasi

- Gunakan threading dengan cara yang fundamental
- Jawa Mesin virtual adalah multi-proses berulir yang menyediakan penjadwalan dan manajemen memori untuk aplikasi Java

Multi-aplikasi instan

- Jika beberapa contoh aplikasi memerlukan beberapa tingkat isolasi, teknologi virtualisasi dapat digunakan untuk menyediakan masing-masing lingkungan yang terpisah dan aman

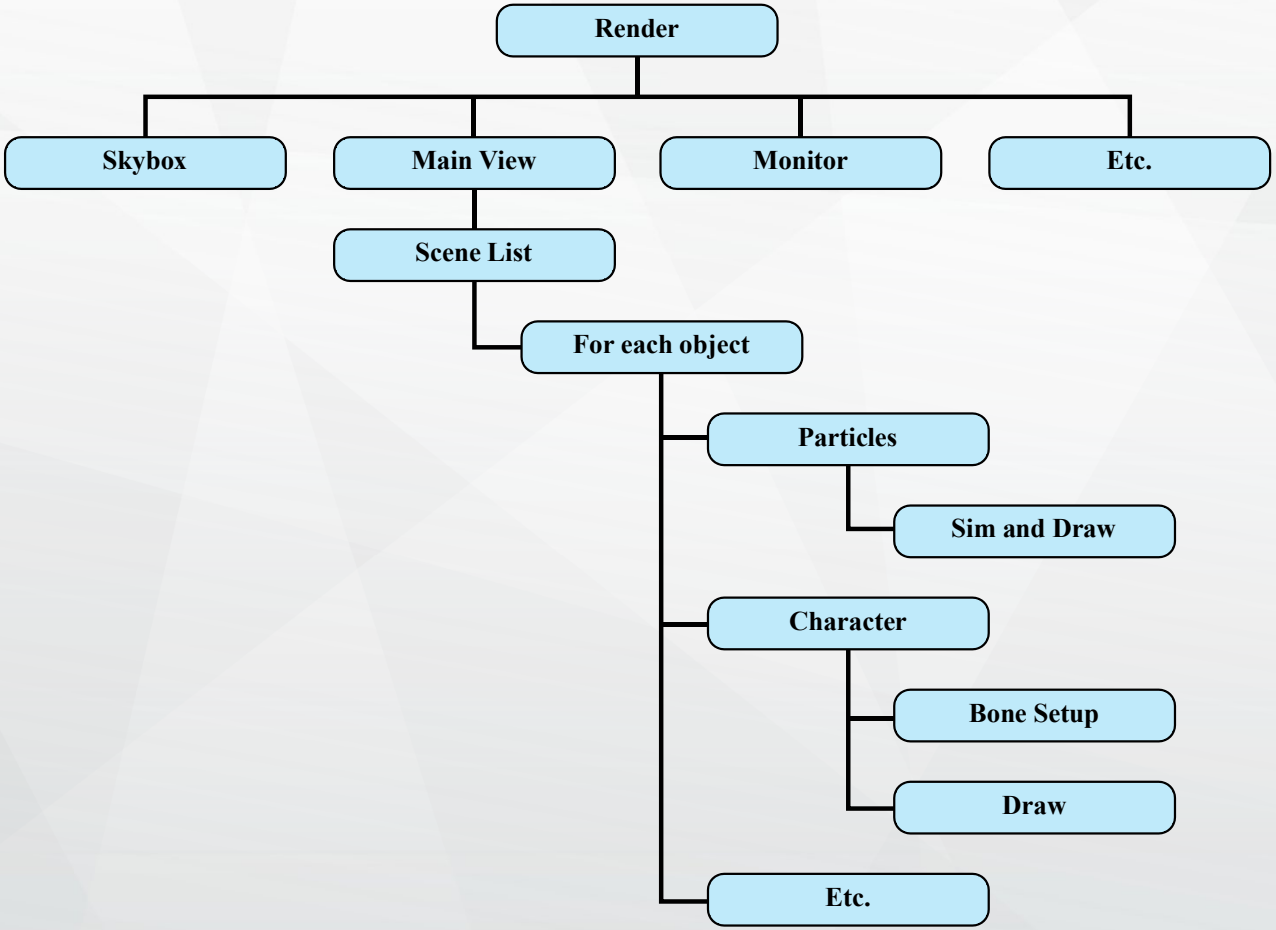


Figure 18.5 Hybrid Threading for Rendering Module

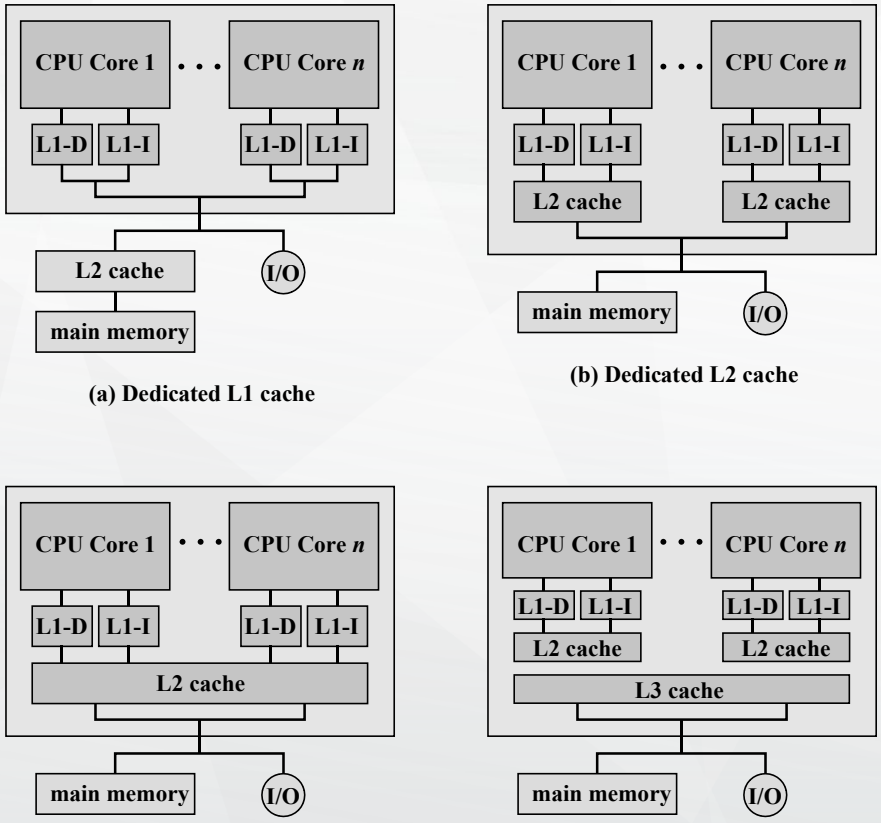


Figure 18.6 Multicore Organization Alternatives

Organisasi Multicore heterogen

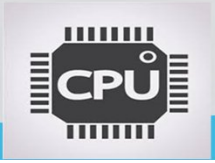
Mengacu pada chip prosesor yang mencakup lebih dari satu jenis inti

Tren yang paling menonjol adalah penggunaan CPU dan unit pemrosesan grafis (GPU) pada chip yang sama

- This mix however presents issues of coordination and correctness

GPU ditandai dengan kemampuan untuk mendukung ribuan tren eksekusi paralel

Dengan demikian, GPU sangat cocok dengan aplikasi yang memproses data vektor dan matriks dalam jumlah besar



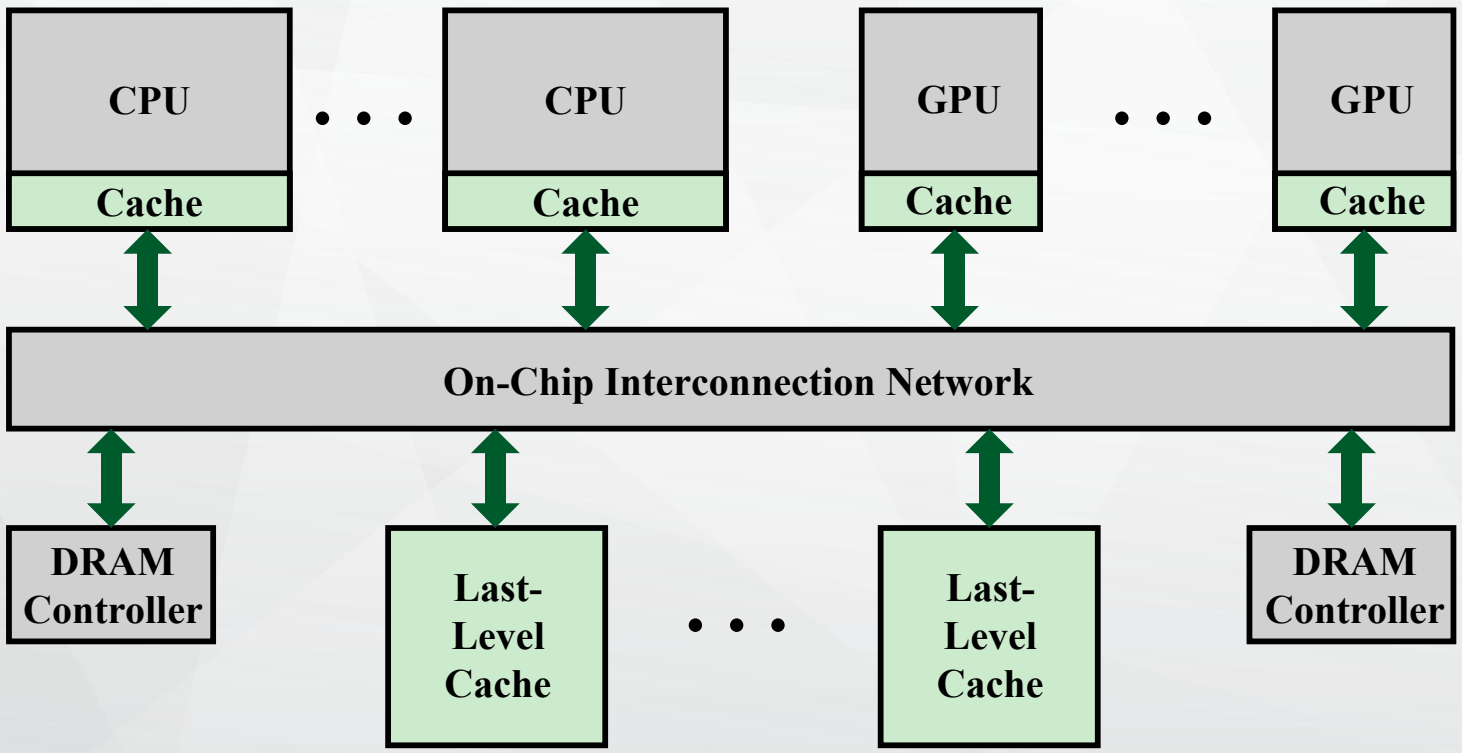


Figure 18.7 Heterogenous Multicore Chip Elements

Tabel 18.1

Pengoperasian Parameter AMD 5100K Heterogen Prosesor Multicore

	CPU	GPU
Clock frequency (GHz)	3.8	0.8
Cores	4	384
FLOPS/core	8	2
GFLOPS	121.6	614.4

FLOPS = operasi floating point per detik
FLOPS / core = jumlah operasi floating point paralel yang dapat dilakukan

Arsitektur Sistem Heterogen (HSA)

Fitur utama dari pendekatan HSA meliputi:

- Seluruh ruang memori virtual terlihat oleh CPU dan GPU
- Sistem memori virtual memasukkan halaman ke memori utama fisik sesuai kebutuhan
- Kebijakan memori yang koheren memastikan bahwa cache CPU dan GPU sama-sama melihat tampilan data yang terbaru
- Antarmuka pemrograman terpadu yang memungkinkan pengguna untuk memanfaatkan kemampuan paralel dari GPU dalam program yang juga mengandalkan eksekusi CPU

Tujuan keseluruhannya adalah untuk memungkinkan pemrogram menulis aplikasi yang memanfaatkan kekuatan serial CPU dan kekuatan pemrosesan paralel GPU secara mulus dengan koordinasi yang efisien di tingkat OS dan perangkat keras.

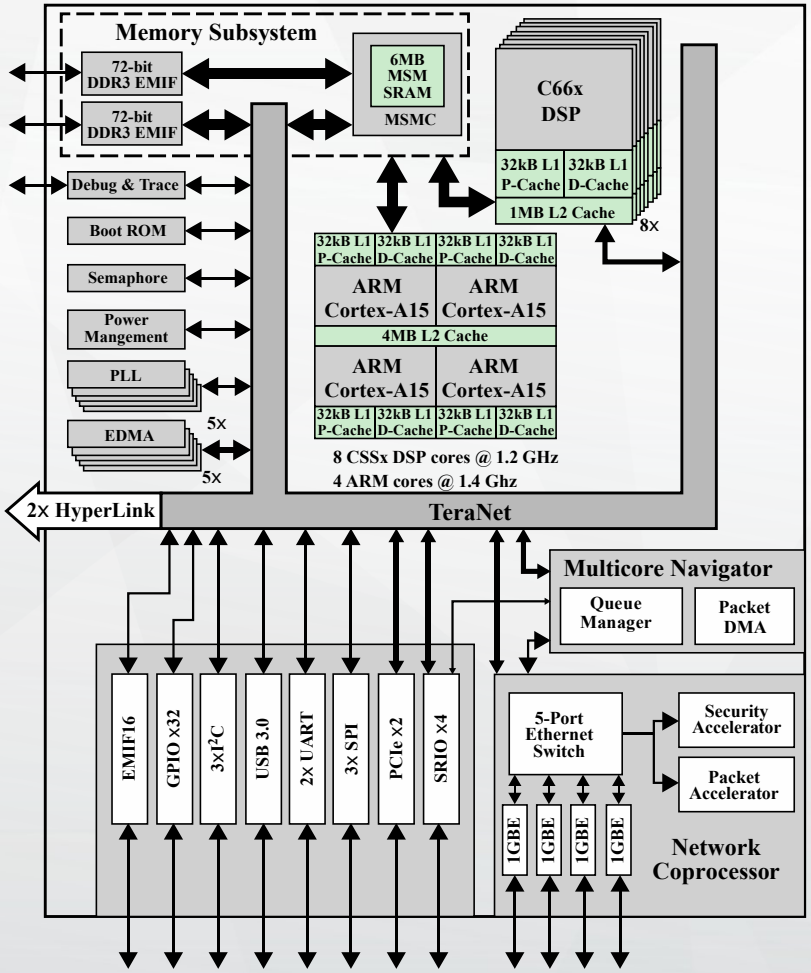


Figure 18.8 Texas Instruments 66AK2H12 Heterogenous Multicore Chip

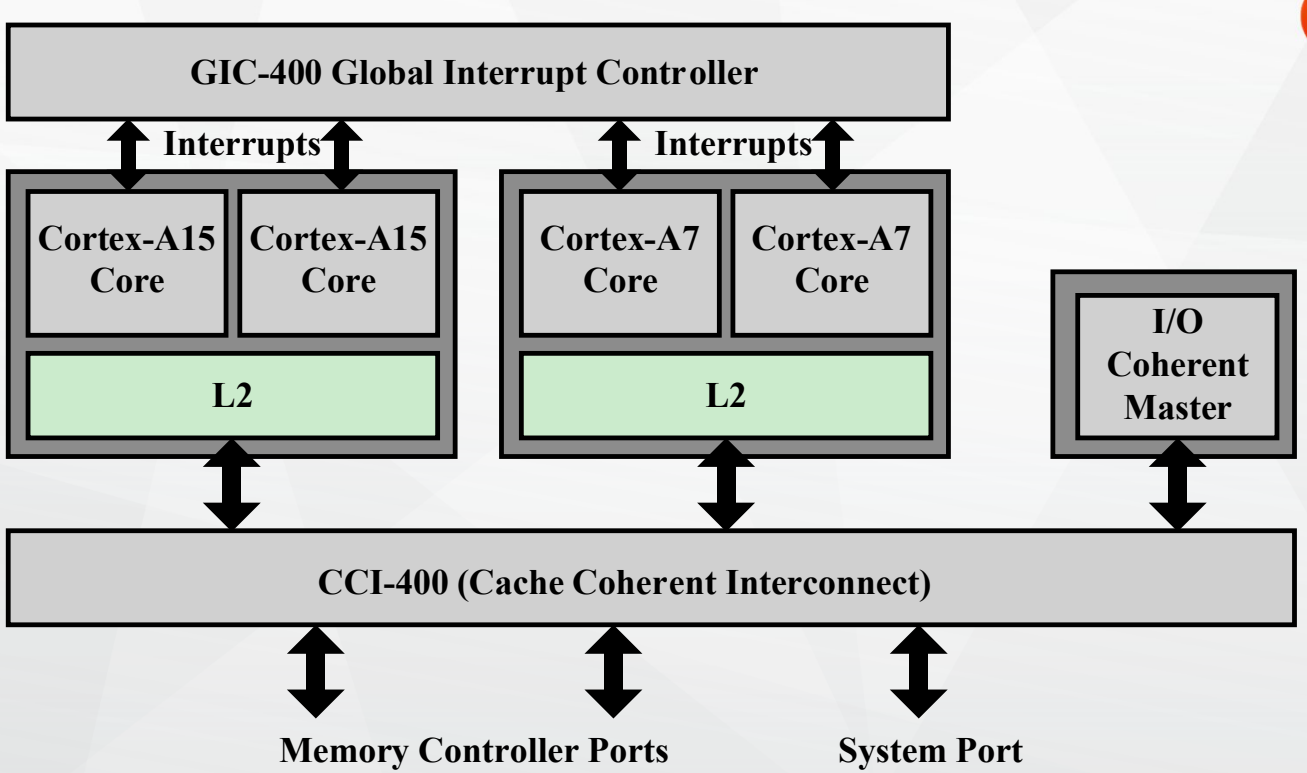
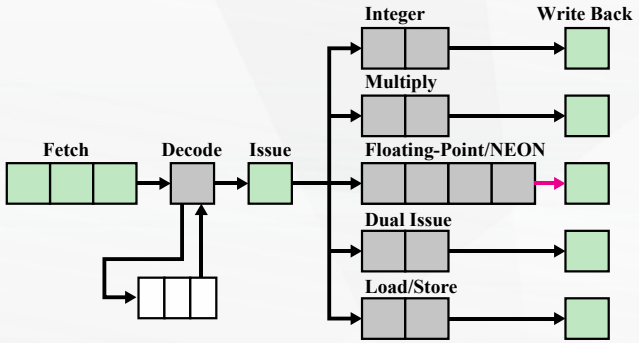
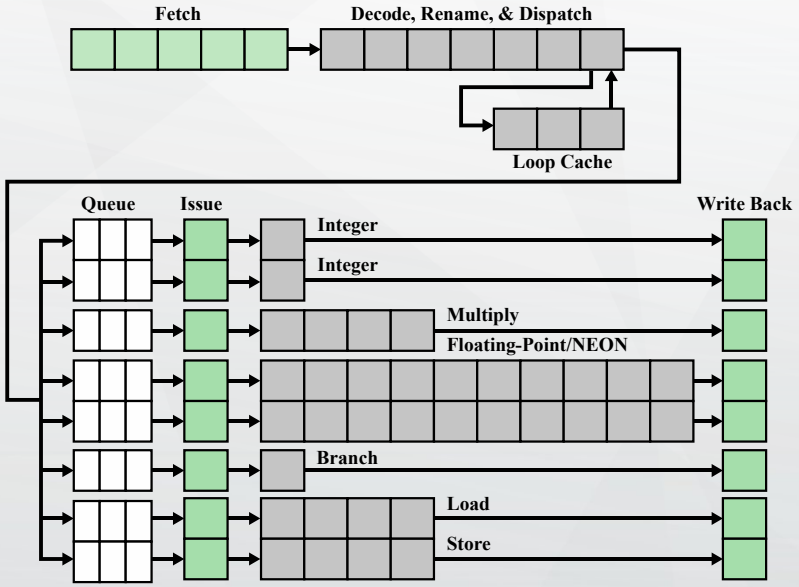


Figure 18.9 Big.Litte Chip Components



(a) Cortex A-7 Pipeline



(b) Cortex A-15 Pipeline

Figure 18.10 Cortex A-7 and A-15 Pipelines

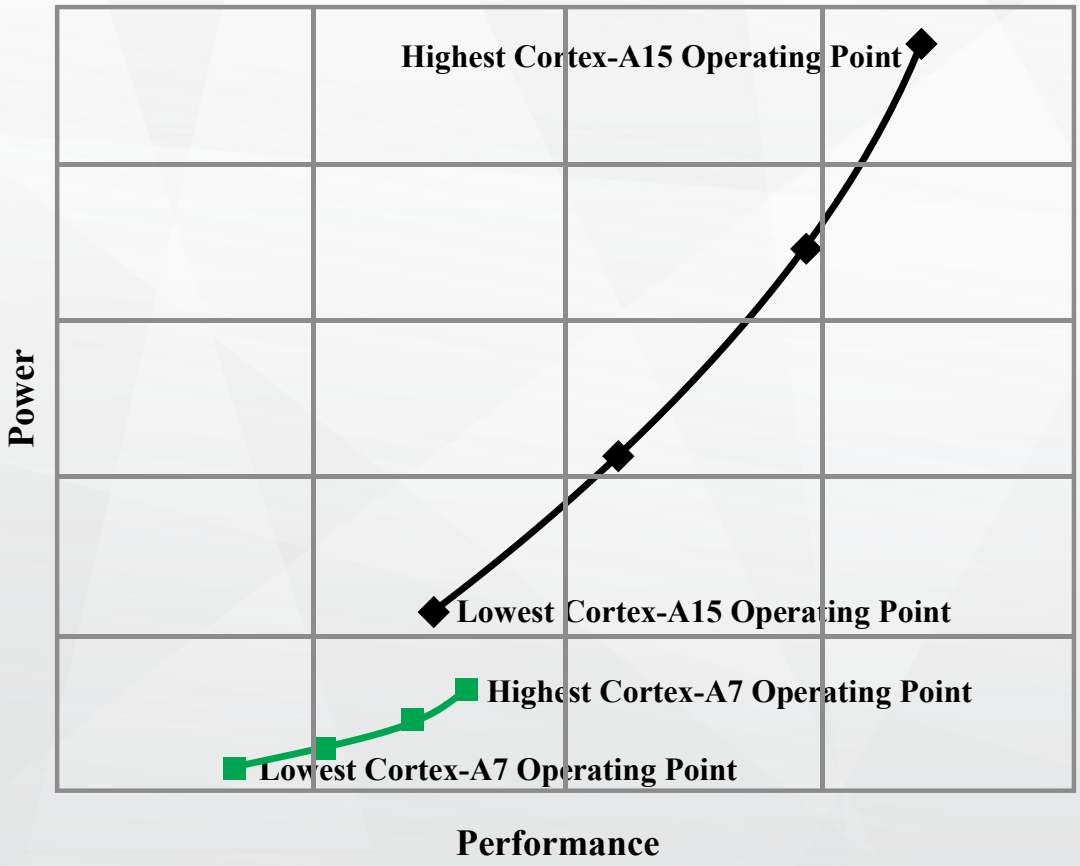


Figure 18.11 Cortex-A7 and A15 Performance Comparison

Koherensi Cache

Dapat diatasi dengan teknik berbasis software

- Beban perangkat lunak menghabiskan terlalu banyak sumber daya dalam file SoC chip

Jika ada beberapa cache, skema koherensi cache diperlukan untuk menghindari akses ke data yang tidak valid

Ada dua pendekatan utama untuk koherensi cache yang diimplementasikan perangkat keras

- Protokol direktori
- Protokol snoopy

ACE (Ekstensi Koherensi Antarmuka yang Dapat Diperluas Tingkat Lanjut)

- Kemampuan koherensi perangkat keras yang dikembangkan oleh ARM
- Dapat dikonfigurasi untuk diterapkan apakah direktori atau pendekatan snoopy
- Telah dirancang untuk mendukung berbagai master yang koheren dengan kemampuan berbeda
- Mendukung koherensi antara prosesor berbeda yang memungkinkan ARM besar. kecil teknologi
- Mendukung koherensi I / O untuk master yang tidak di-cache, mendukung master dengan ukuran baris cache yang berbeda, model status cache internal yang berbeda, dan master dengan cache tulis-balik atau tulis-tayang

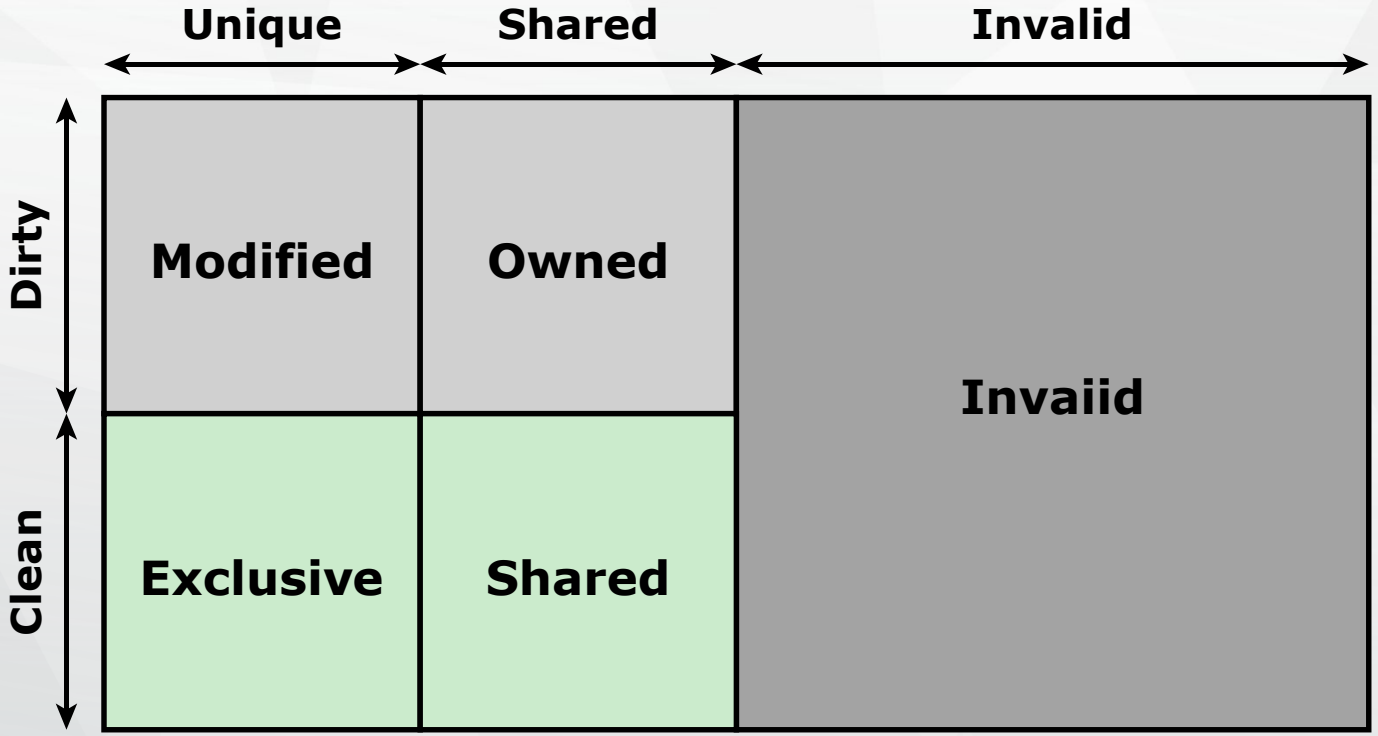


Figure 18.12 ARM ACE Cache Line States

Table 18.2 Comparison of States in Snoop Protocols

(a) MESI

	Modified	Exclusive	Shared	Invalid
Clean/Dirty	Dirty	Clean	Clean	N/A
Unique?	Yes	Yes	No	N/A
Can write?	Yes	Yes	No	N/A
Can forward?	Yes	Yes	Yes	N/A
Comments	Must write back to share or replace	Transitions to M on write	Shared implies clean, can forward	Cannot read

(b) MOESI

	Modified	Owned	Exclusive	Shared	Invalid
Clean/Dirty	Dirty	Dirty	Clean	Either	N/A
Unique?	Yes	Yes	Yes	No	N/A
Can write?	Yes	Yes	Yes	No	N/A
Can forward?	Yes	Yes	Yes	No	N/A
Comments	Can share without write back	Must write back to transition	Transitions to M on write	Shared, can be dirty or clean	Cannot read

(Tabel dapat ditemukan di halaman 676 di buku teks.)

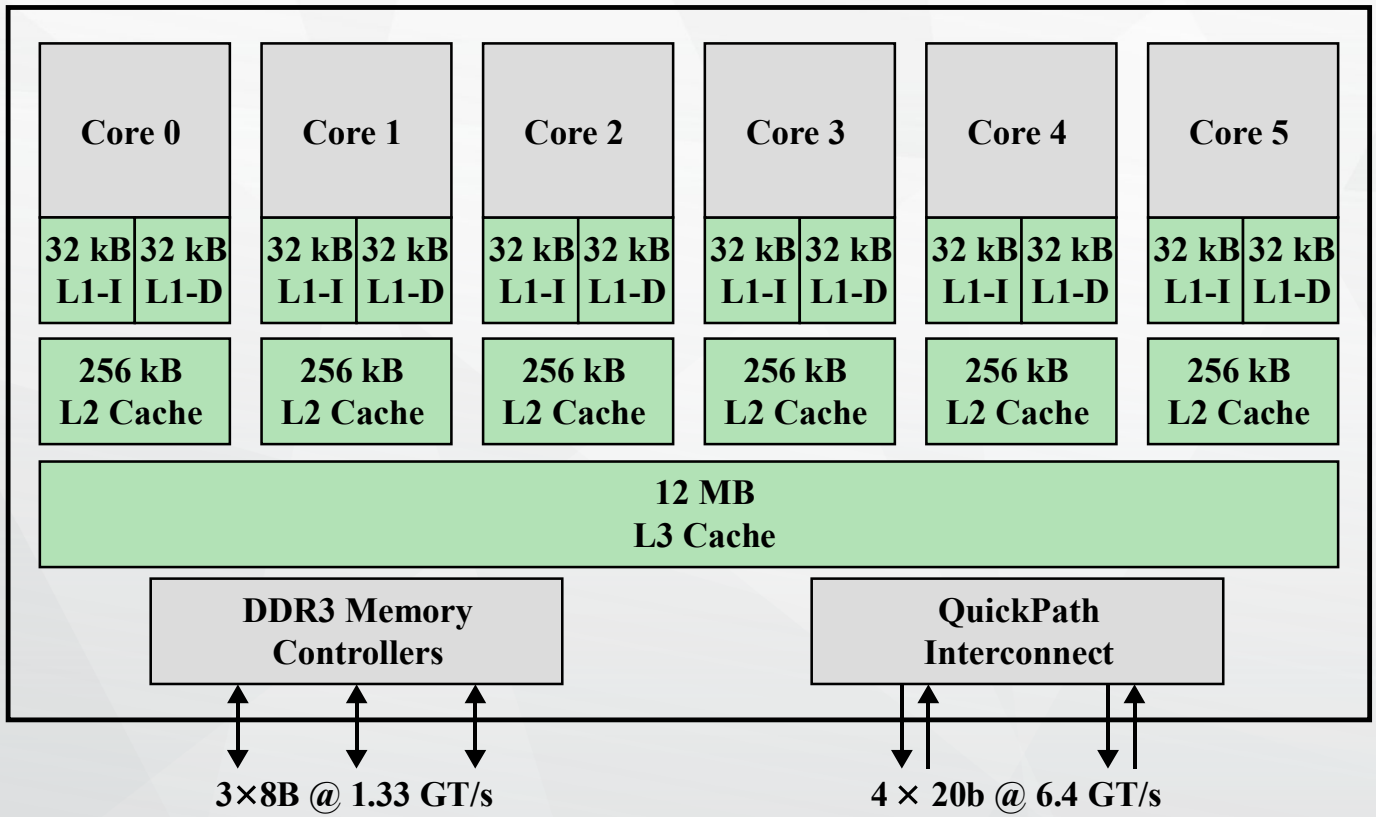


Figure 18.13 Intel Core i7-990X Block Diagram

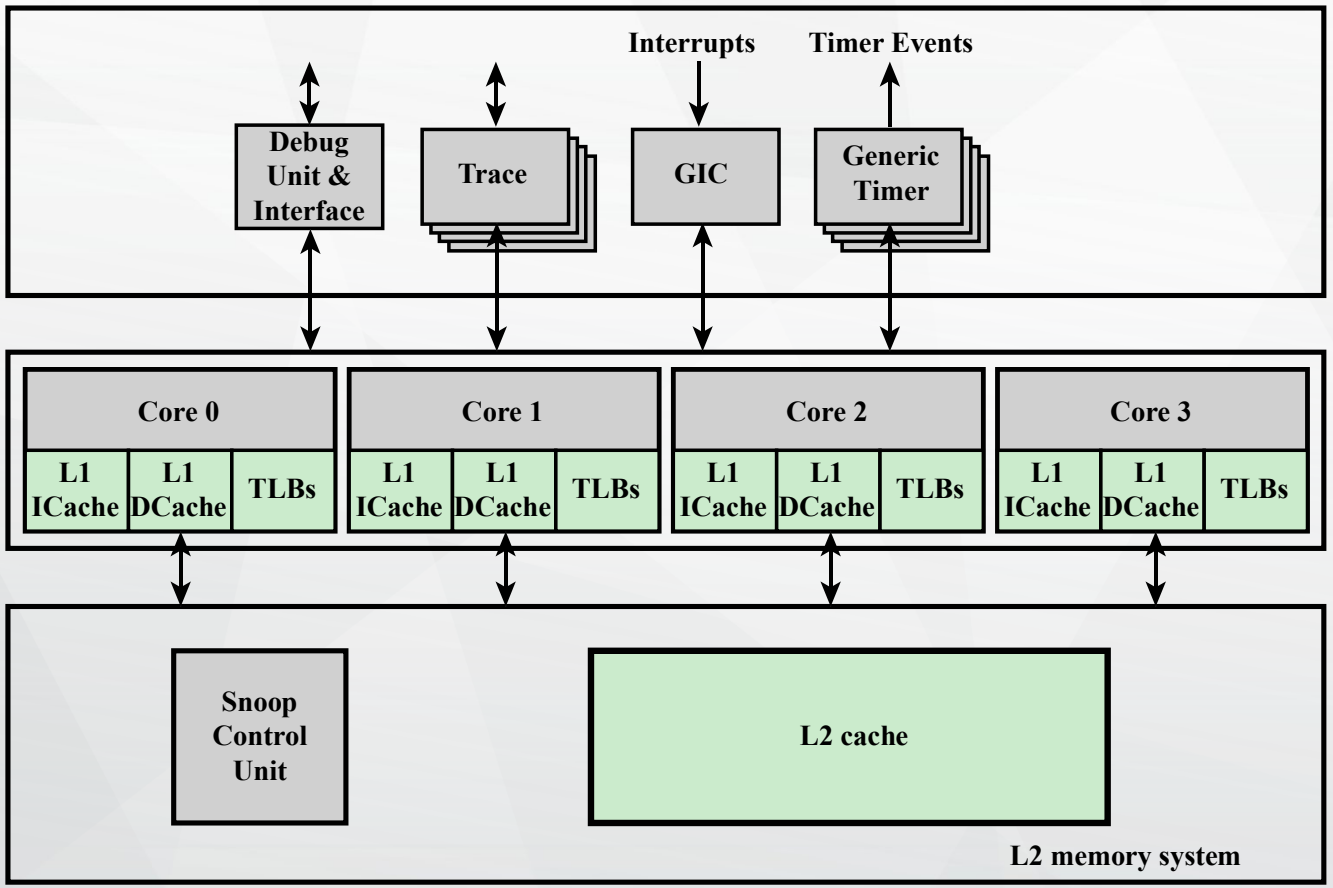


Figure 18.14 ARM Cortex-A15 MPCore Chip Block Diagram

Penanganan Interupsi

Generic interrupt controller (GIC) provides:

- Masking interupsi
- Prioritas interupsi
- Distribusi interupsi ke target inti A15
- Melacak status interupsi
- Generasi interupsi oleh perangkat lunak

GIC

- Apakah memori dipetakan
- Adalah unit fungsional tunggal yang ditempatkan dalam sistem bersama inti A15
- Hal ini memungkinkan jumlah interupsi yang didukung dalam sistem menjadi independen dari desain inti A15
- Diakses oleh core A15 menggunakan antarmuka privat melalui SCU

GIC

Dirancang untuk memenuhi dua persyaratan fungsional:

- Menyediakan sarana untuk merutekan permintaan interupsi ke satu CPU atau beberapa CPU sesuai kebutuhan
- Menyediakan sarana komunikasi interprosesor sehingga thread pada satu CPU dapat menyebabkan aktivitas dengan utas pada CPU lain

Dapat merutekan interupsi ke satu atau beberapa CPU dengan tiga cara berikut:

- Interupsi hanya dapat diarahkan ke prosesor tertentu
- An interrupt can be directed to a defined group of processors
- Interupsi dapat diarahkan ke semua prosesor

Interupsi dapat berupa:

Tidak aktif

Salah satu yang tidak ditegaskan, atau yang dalam lingkungan multiprosesing telah sepenuhnya diproses oleh CPU itu tetapi masih bisa Tertunda atau Aktif di beberapa CPU yang ditargetkan, dan mungkin belum dihapus di sumber interupsi

Menunggu keputusan

Salah satu yang telah ditetapkan, dan yang pemrosesannya belum dimulai pada CPU tersebut

Aktif

Salah satu yang telah dimulai pada CPU itu, tetapi pemrosesannya belum selesai

Dapat dicegah ketika interupsi baru dengan prioritas yang lebih tinggi mengganggu pemrosesan interupsi inti A15

Interupsi berasal dari sumber berikut:

Interupsi interprosesor (IPI)

Pengatur waktu pribadi dan / atau pengawas menyela

Garis FIQ lama

Interupsi perangkat keras

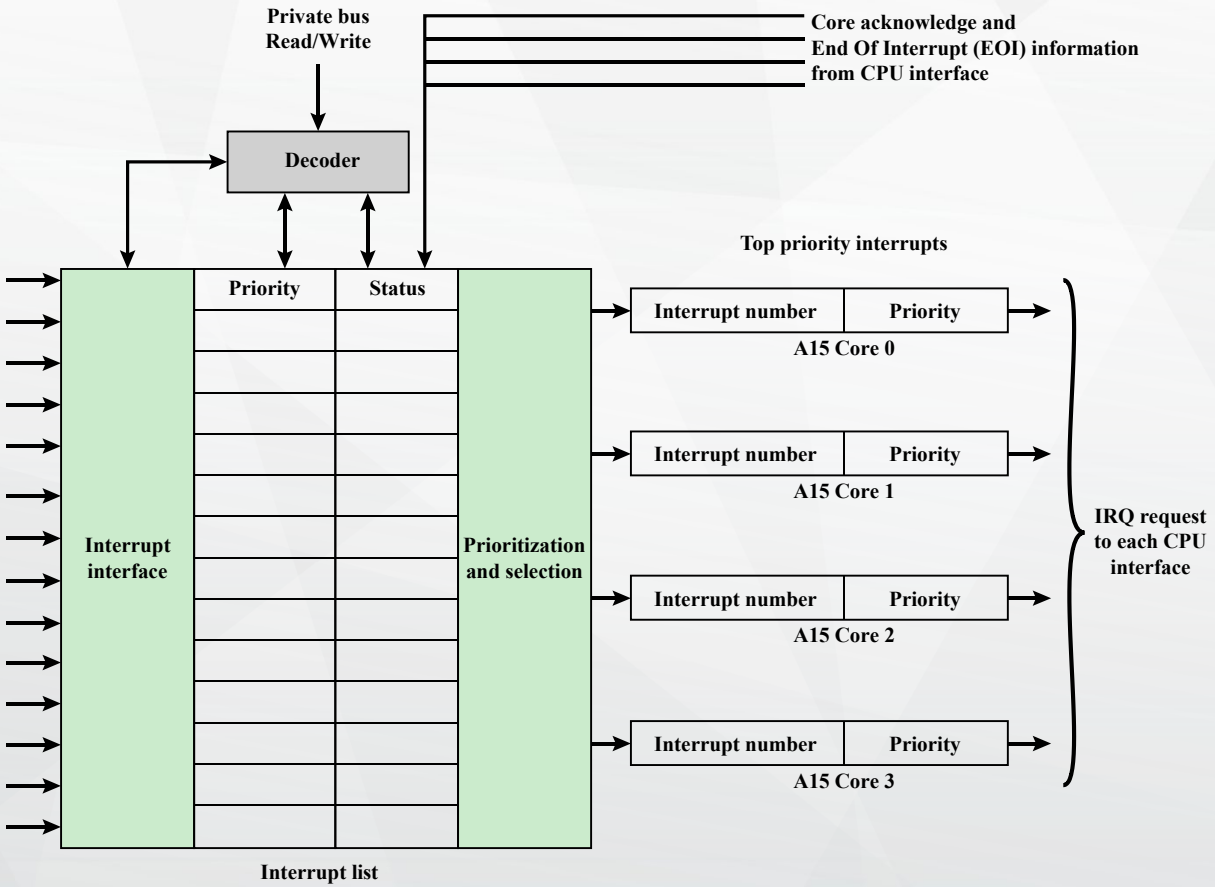


Figure 18.15 Interrupt Distributor Block Diagram

Koherensi Cache

Snoop Control Unit (SCU) menyelesaikan sebagian besar hambatan tradisional terkait akses ke data bersama dan batasan skalabilitas yang disebabkan oleh lalu lintas koherensi

Koherensi cache L1 skema didasarkan di protokol MESI

Langsung Intervensi Data (DDI)

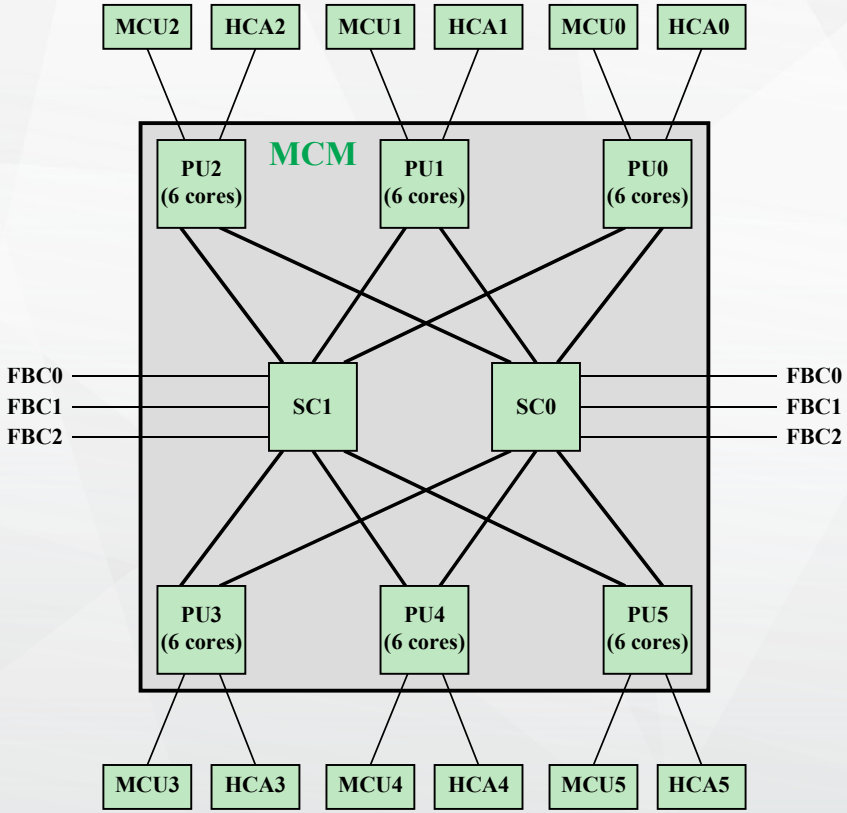
- Memungkinkan penyalinan bersih data antara cache L1 tanpa mengakses memori eksternal
- Mengurangi membaca setelah menulis dari L1 ke L2
- Dapat menyelesaikan ketinggalan L1 lokal dari terpencil L1 bukan L2

RAM tag duplikat

- Tag cache diimplementasikan sebagai blok RAM terpisah
- Panjangnya sama dengan jumlah baris dalam cache
- Duplikat yang digunakan oleh SCU untuk memeriksa ketersediaan data sebelum mengirim perintah koherensi
- Hanya kirim ke CPU yang harus memperbarui cache data yang koheren

Garis migrasi

- Memungkinkan memindahkan data kotor antara CPU tanpa menulis ke L2 dan membaca kembali dari memori eksternal



FBC = fabric book connectivity
HCA = host channel adapter
MCM = multichip module
MCU = memory control unit
PU = processor unit
SC = storage control

Figure 18.16 IBM zEC12 Processor Node Structure

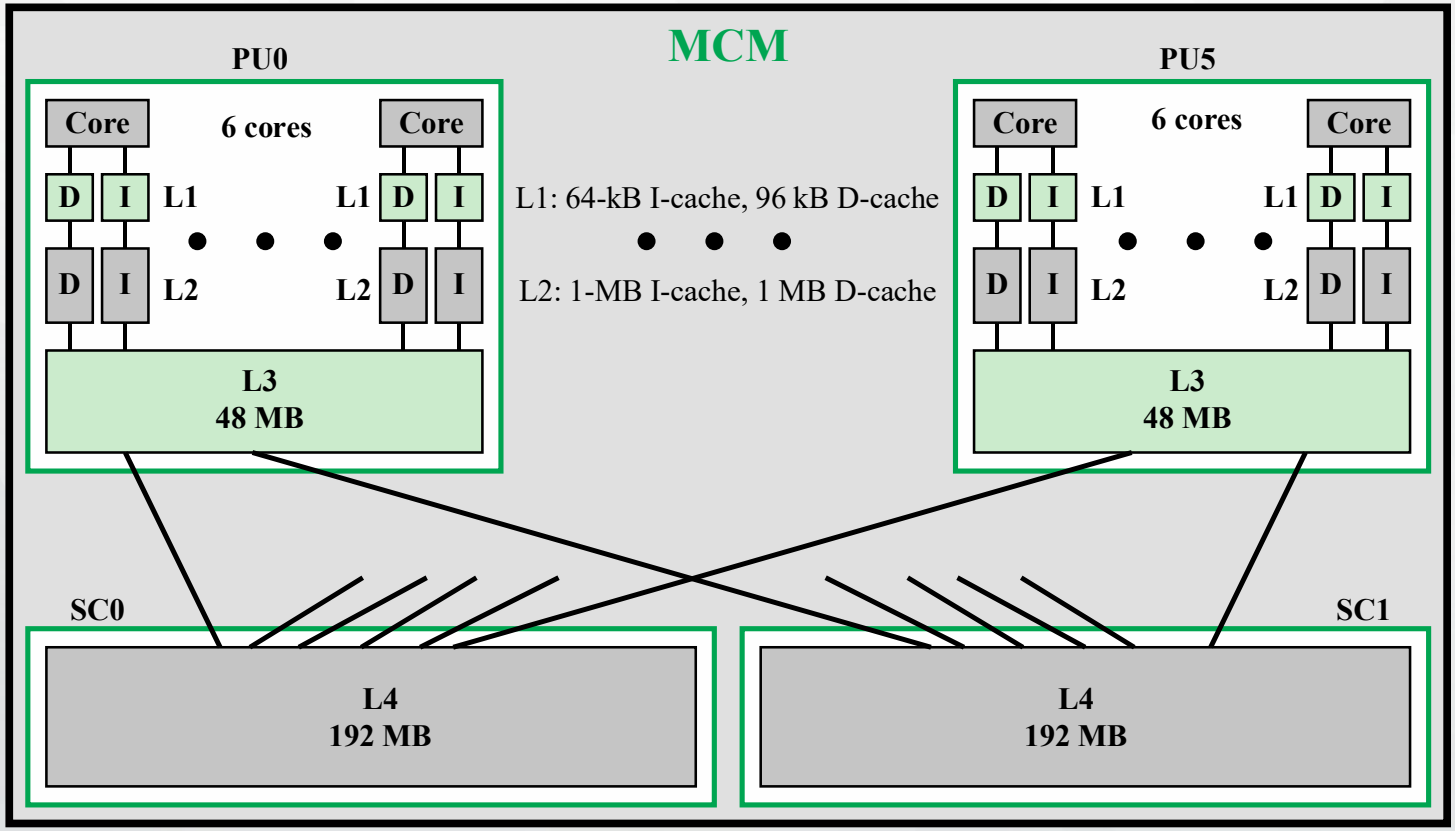


Figure 18.17 IBM zEC12 Cache Hierarchy

Ringkasan

Bab 18

Masalah kinerja perangkat keras

- Peningkatan paralelisme dan kompleksitas
- Konsumsi daya

Masalah kinerja perangkat lunak

- Perangkat lunak pada multicore
- Perangkat lunak permainan katup contoh

Intel Core i7-990X

IBM zEnterprise EC12 mainframe

- Organisasi
- Struktur cache

Multicore Komputer

Multicore organisasi

- Tingkat cache
- Multithreading simultan

Multi inti heterogen organisasi

- Arsitektur set instruksi yang berbeda
- Arsitektur set instruksi yang setara
- Koherensi cache dan model MOESI

ARM Cortex-A15 MPCore

- Penanganan interupsi
- Koherensi cache
- Koherensi cache L2