Parallel & Distributed Computing (WQD7008)

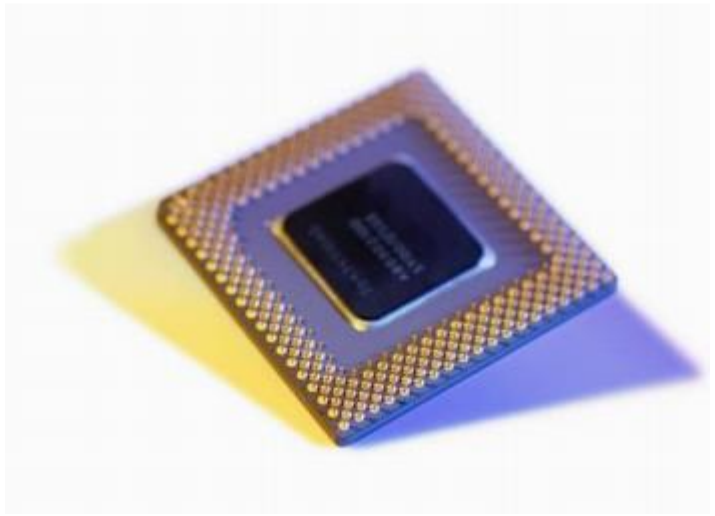
Week 2

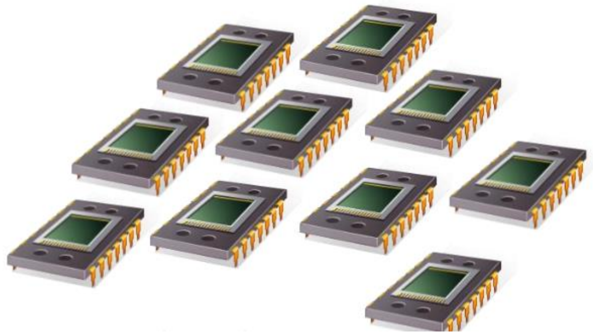
2019/2020 Semester 1

Dr. Hamid Tahaei

Parallel vs Distributed System

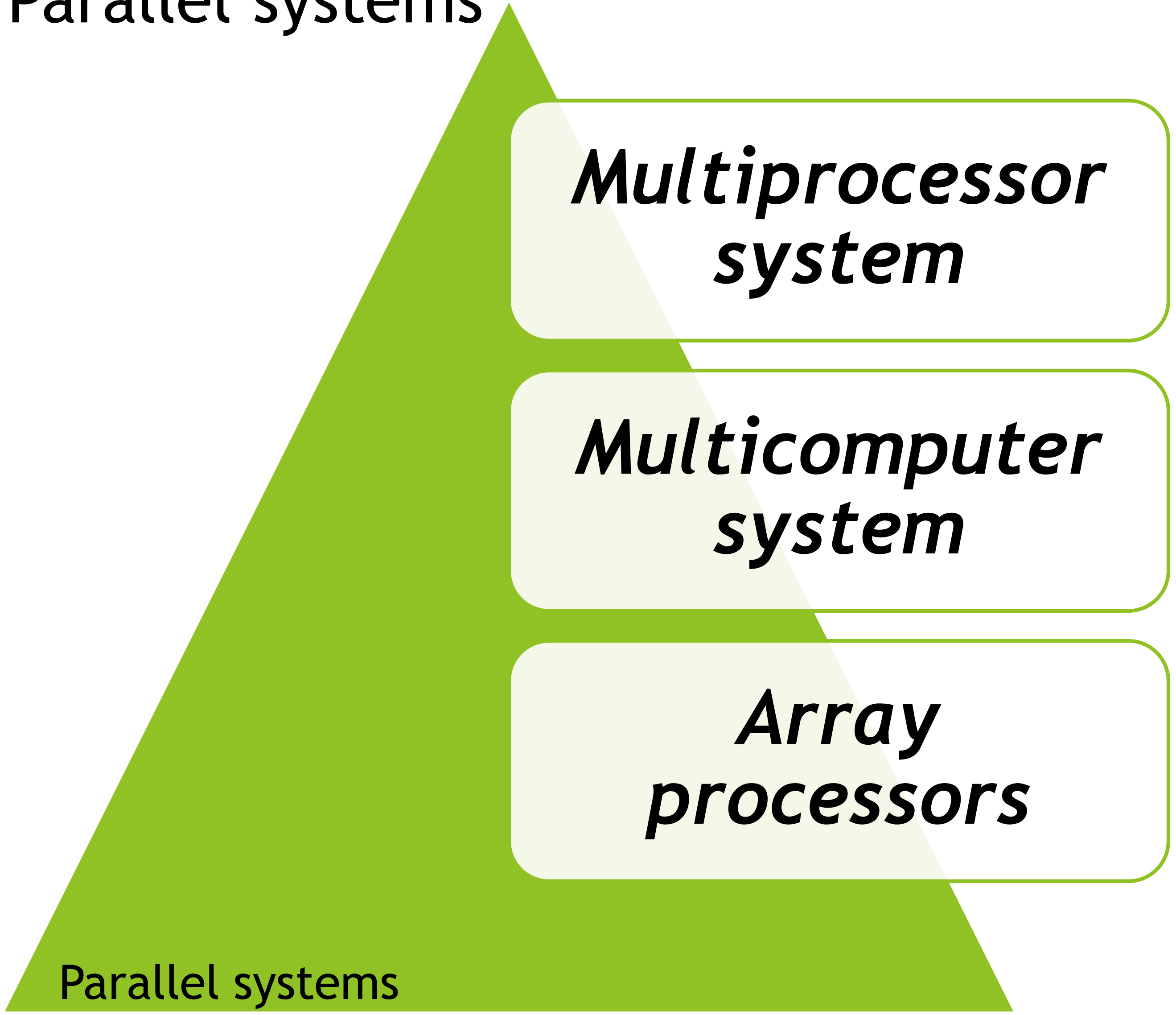
	Parallel Systems	Distributed Systems
Memory	**Tightly coupled shared** memory UMA, NUMA	**Distributed** memory Message passing, RPC, and/or used of distributed shared memory
Control	**Global clock** control SIMD, MIMD	**No global clock** control Synchronization algorithms needed
Processor interconnection	Order of **Tbps** Bus, mesh, tree, mesh of tree, and hypercube (-related) network	Order of **Gbps** Ethernet(bus), token ring and Star(ring), myrinet(switching network)
Operation System **Processor Type**	The **same** operating system, hardware and software are very tightly coupled. usually of **the same type**, and are housed within the same box/container with a shared memory.	**Heterogenous** CPU, **Heterogeneous** OS, **No share** memory, Geographically **separated**
Main focus	**Performance** Scientific computing	Performance(**cost and scalability**) **Reliability/availability** **Information/resource sharing**


CPU


Core

multiple CPU’s or cores on a single chip

models of Parallel systems

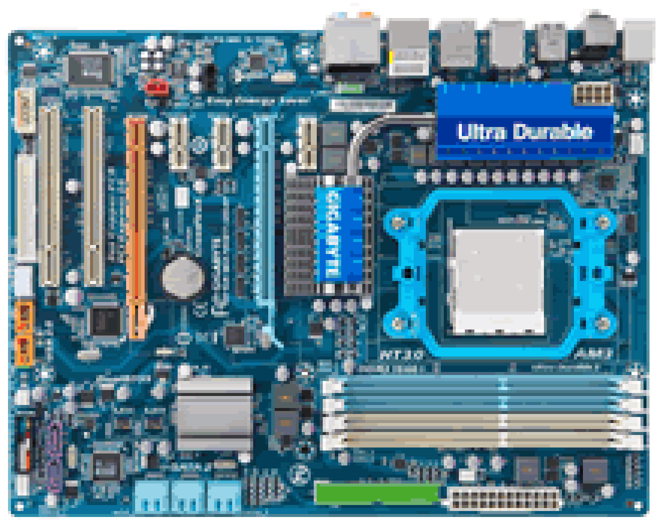


Parallel Systems Model and Characteristic

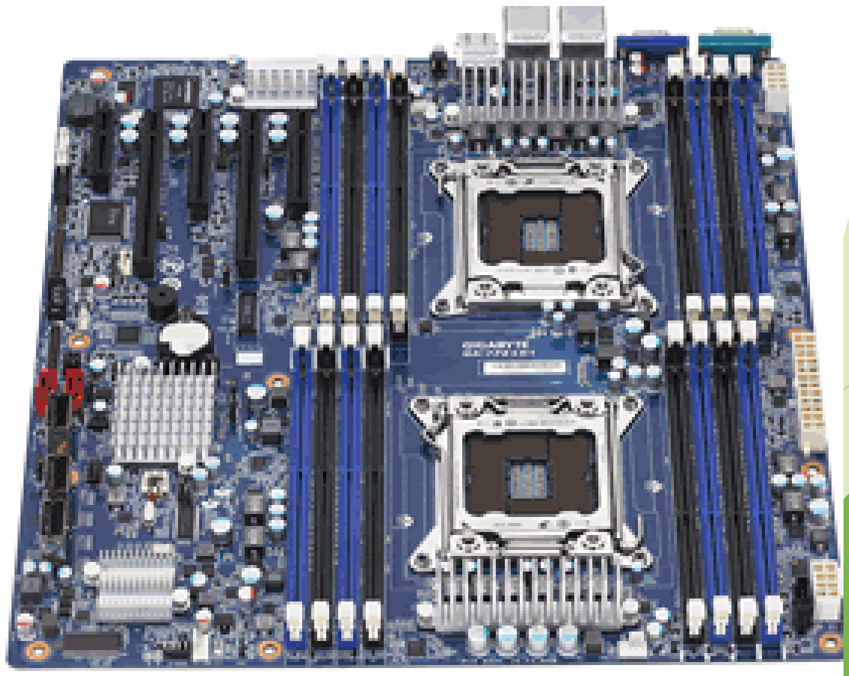
1. A ***multiprocessor system***
   - multiple processors have ***direct access** to shared memory* which forms a common address space.
   - Such processors usually **do not have a common clock**.
   - *usually* corresponds to a uniform memory access **(UMA)** architecture.
     - the access latency, i.e., waiting time, to complete an access to any memory location from any processor is the same.

Parallel Systems Model and Characteristic

- processors are in very close physical proximity and are connected by an interconnection network.
- Interprocess communication across processors is traditionally through read and write operations on the shared memory.
- although the use of message-passing primitives such as MPI, is also possible (using emulation on the shared memory).
- All the processors usually run **the same operating system**, and both the **hardware and software are very tightly coupled**.

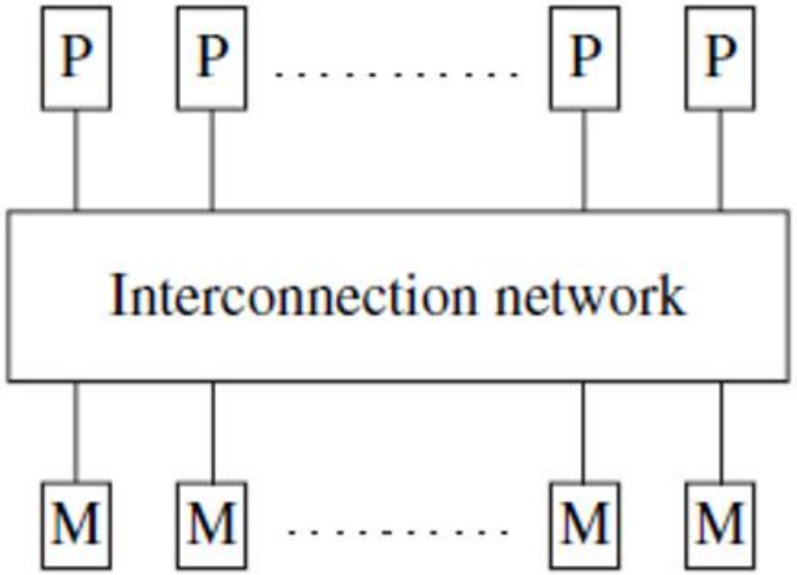

Single Processor System

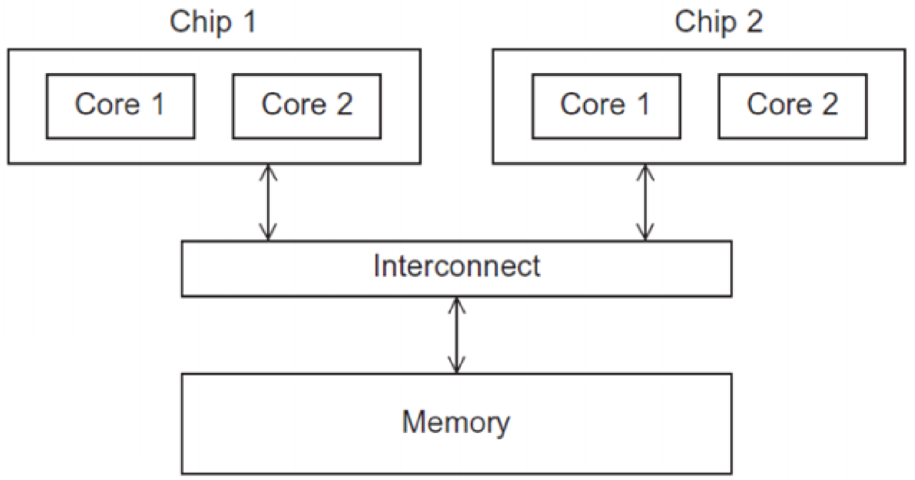
vs


Multi Processor System

A *multiprocessor system* – *(Continued)*

- The processors are usually of **the same type**.
- The processors housed within **the same box/container** with a shared memory.
- The interconnection network to access the memory may be a bus, although for greater efficiency, it is usually a *multistage switch* with a symmetric and regular design





UMA multicore system

Time to access all the memory locations will be the same for all the cores.

Figure 1. Uniform memory access (UMA) multiprocessor system. The processors may locally cache data from memory.

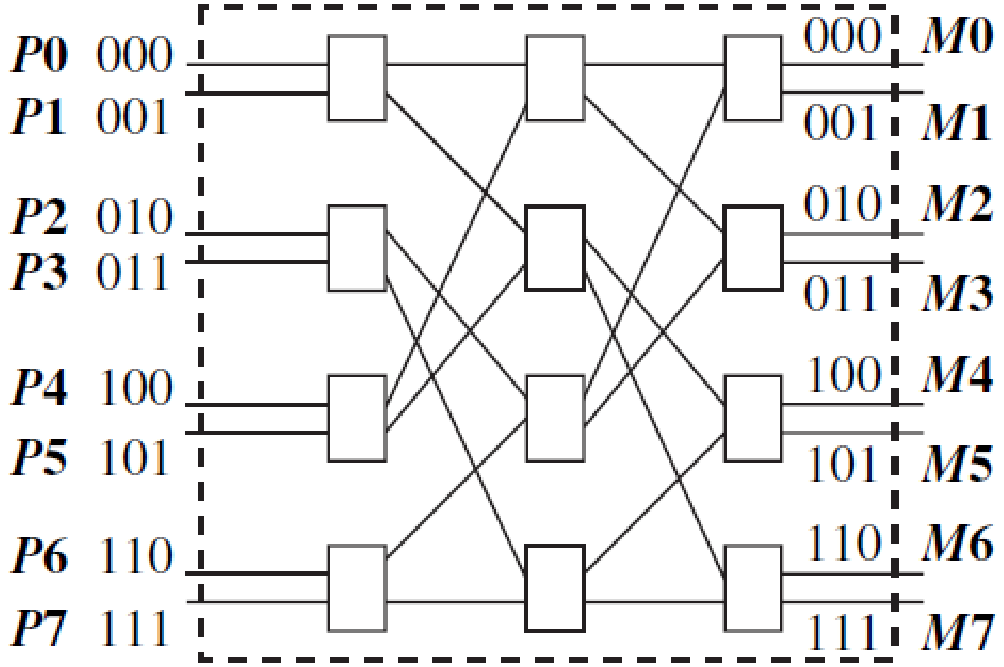
A *multiprocessor system* – *(Continued)*

- Two popular interconnection networks – the Omega network and the Butterfly network.
- Omega Network **Model:**
  - Each 2×2 switch allows data on either of the two input wires to be switched to the upper or the lower output wire. In a single step, however, only **one data unit can be sent on an output wire**. So if the data from both the input wires is to be routed to the same output wire in a single step, there is a collision. Various techniques such as buffering or more elaborate interconnection designs can address collisions.
- Omega **interconnection function:**
  - The Omega network which connects ***n*** processors to ***n*** memory units has:
  - switching elements = $n/2 log_2 n$ switching elements of size 2×2 arranged in $log_2 n$ stage.
- Between each pair of adjacent stages of the Omega network, a link exists between output ***i*** of a stage and the input ***j*** to the next stage according to the following *perfect shuffle* pattern which is a left-rotation operation on the binary representation of ***i*** to get ***j***.

A *multiprocessor system – (Continued)* Omega Network Model:

- The iterative function generation is as follows:

$$j = \begin{cases} 2i, & \text{for } 0 \le i \le n/2-1, \\ 2i+1-n, & \text{for } n/2 \le i \le n-1. \end{cases}$$



Interconnection networks for shared memory multiprocessor systems. Omega network for n = 8 processors P0–P7 and memory banks M0–M7.

Figure 2. 3-stage Omega network (n = 8, M = 4)

A *multiprocessor system* – *(Continued)*

- **Butterfly interconnection Model:**

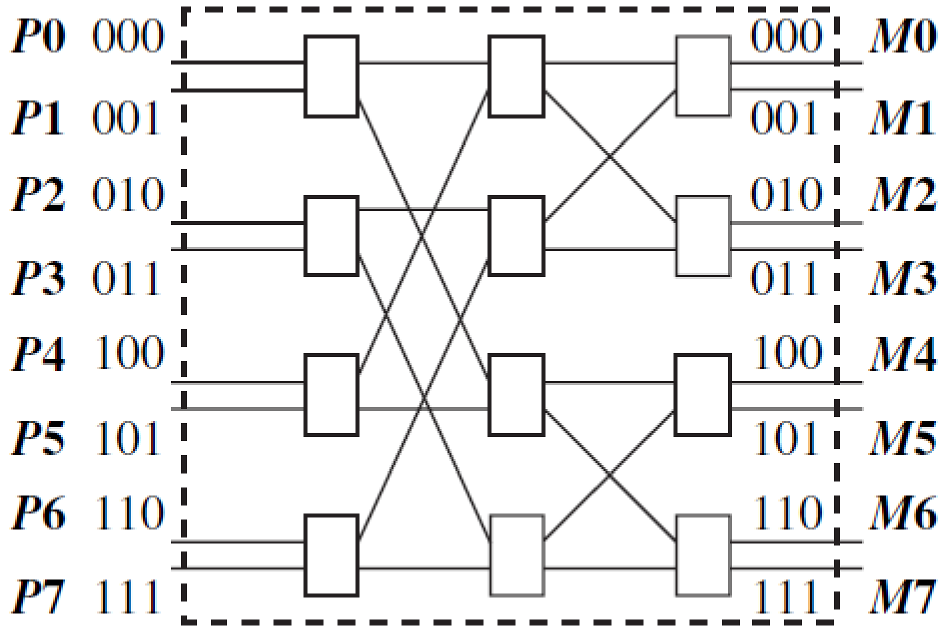
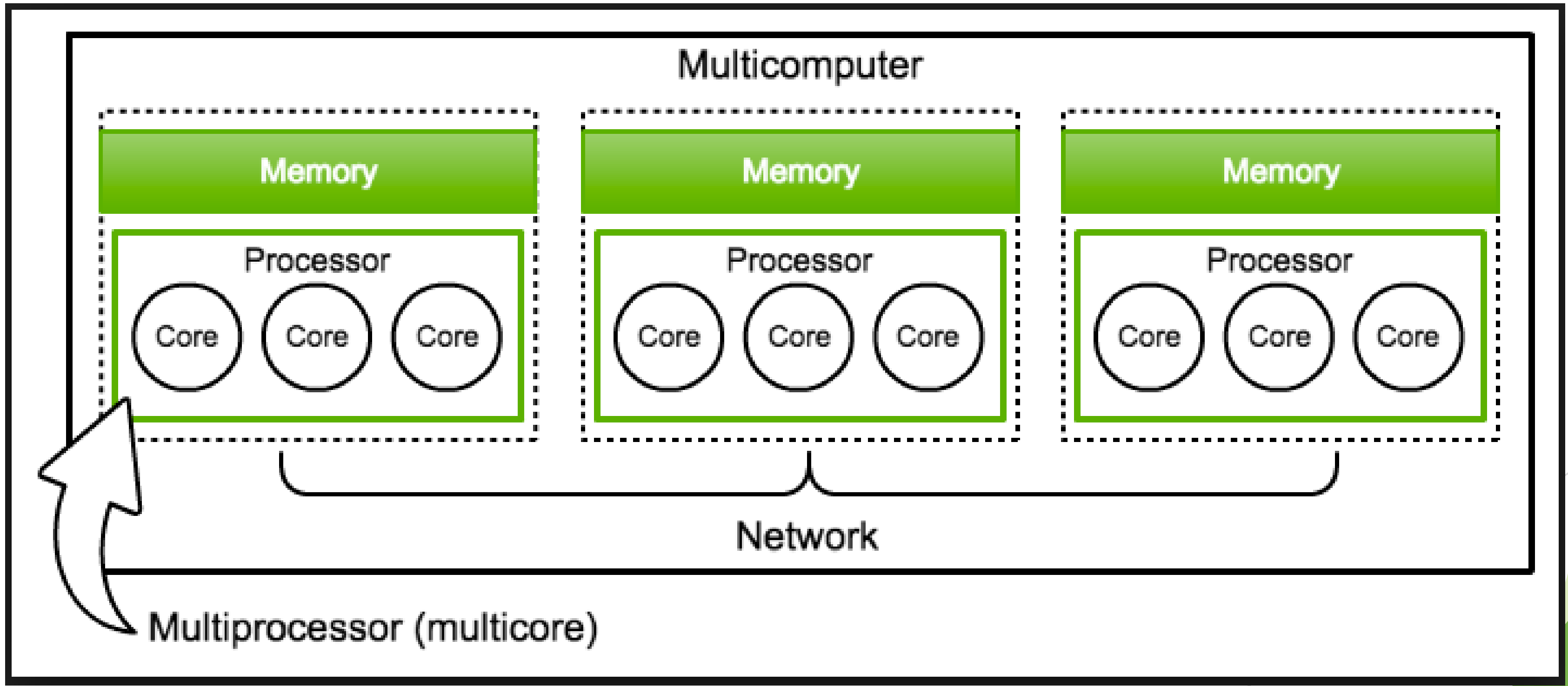


Figure 3: 3-stage Butterfly network (n = 8, M = 4)

2. A *multicomputer system*

- Made up of **several computers**.
- A multicomputer system **is a cluster of computers** that **operate as a singular computer**.
- Multiple processors ***do not have direct access*** *to shared memory*.
- The memory of the multiple processors **may or may not form a common address space**.
- The processors are in **close physical proximity** and are usually **very tightly coupled** (homogenous hardware and software), and connected by an **interconnection network**.



2. A *multicomputer system (Continued)*

- The processors communicate either via a **common address space** or via **message-passing**.
- A multicomputer system that has **a common address space** *usually* corresponds to a **non-uniform memory access (NUMA) architecture**.
- In NUMA, the latency to access various shared memory locations **from the different processors varies.**

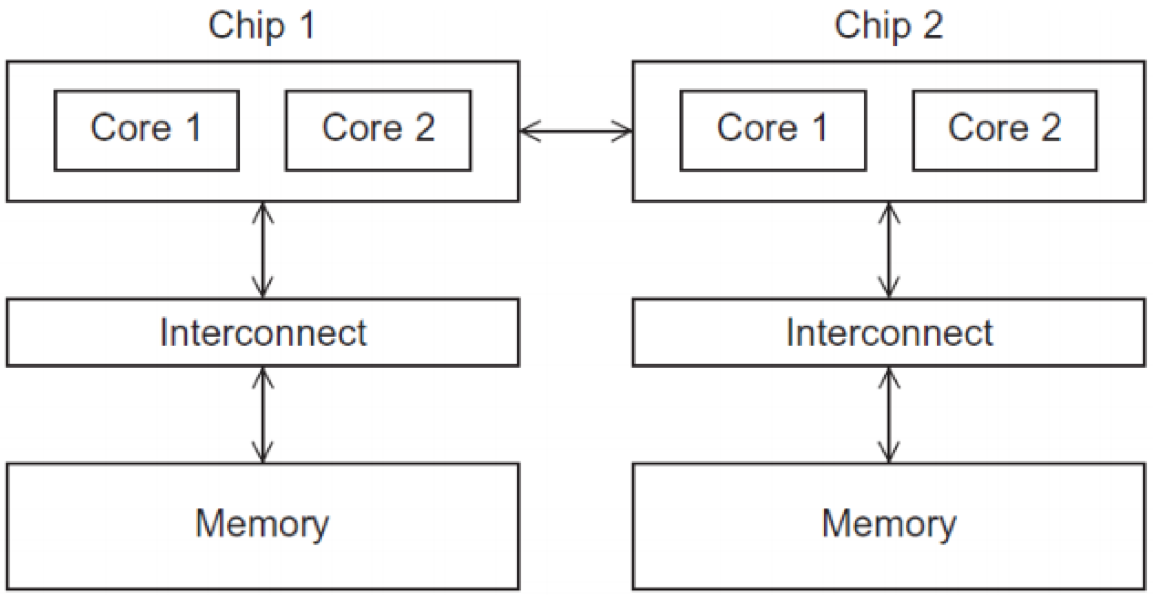


Figure 4. Uniform memory access (NUMA) multiprocessor system. The processors may locally cache data from memory.

A *multicomputer system* - *(Continued)*

- Examples of parallel multicomputers are: the NYU Ultracomputer and the Sequent shared memory machines, Connection machine and processors configured in regular and symmetrical topologies such as an array or mesh, ring, torus, cube, and hypercube (message-passing machines). The regular and symmetrical topologies have interesting mathematical properties that enable very easy routing and provide many rich features such as alternate routing.

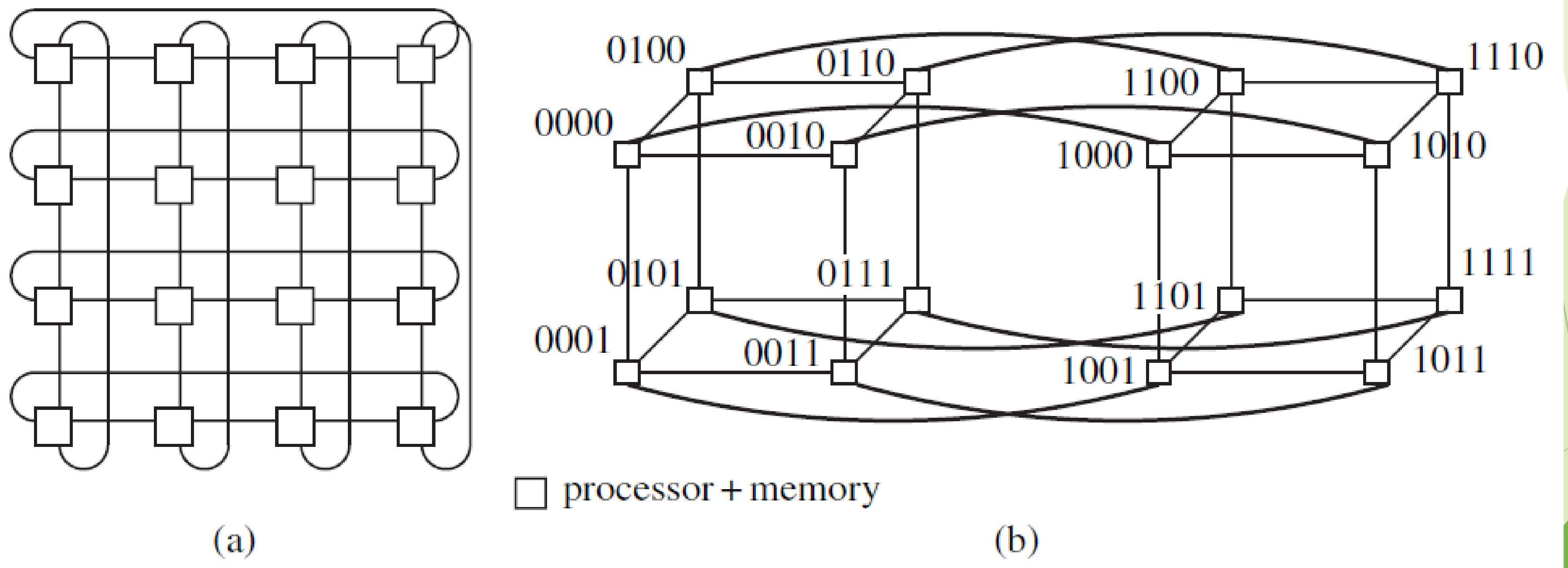


Figure 5: Some popular topologies for multicomputer shared-memory machines. (a) Wrap-around 2D-mesh, also known as torus. (b) Hypercube of dimension 4.

A *multicomputer system - (Continued)*

- In a wrap-around 4×4 mesh, for $k \times k$ mesh:
  - Number of processor: k^2
  - The maximum path length between any two processors: $2(k/2-1)$
  - Routing in the hypercube is done hop-by-hop.
  - At any hop, the message can be sent along any dimension corresponding to the bit position in which the current node's address and the destination address differ.
  - there are multiple routes between any pair of nodes, which provides fault tolerance as well as a congestion control mechanism.

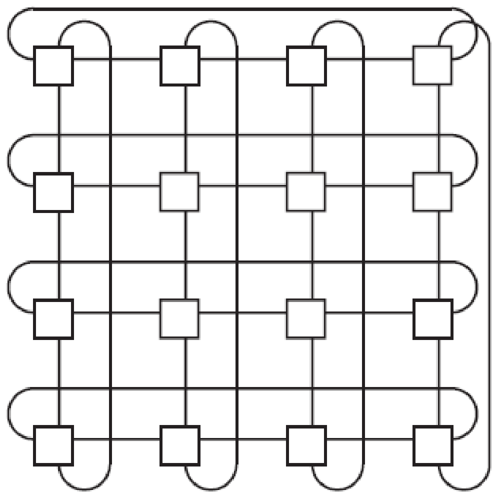


Figure 6: Wrap-around 4X4 mesh also known as torus, a popular topologies for multicomputer shared-memory machines.

A *multicomputer system* – *(Continued)*

- Four-dimensional hypercube: $k \times k$
  - A k-dimensional hypercube has 2^k processors and memory units.
  - The labels of any two adjacent nodes are identical except for the bit position corresponding to the dimension in which the two nodes differ.
  - the processors are labelled such that the shortest path between any two processors is the ***Hamming distance*** (defined as the number of bit positions in which the two equal sized bit strings differ) between the processor labels.

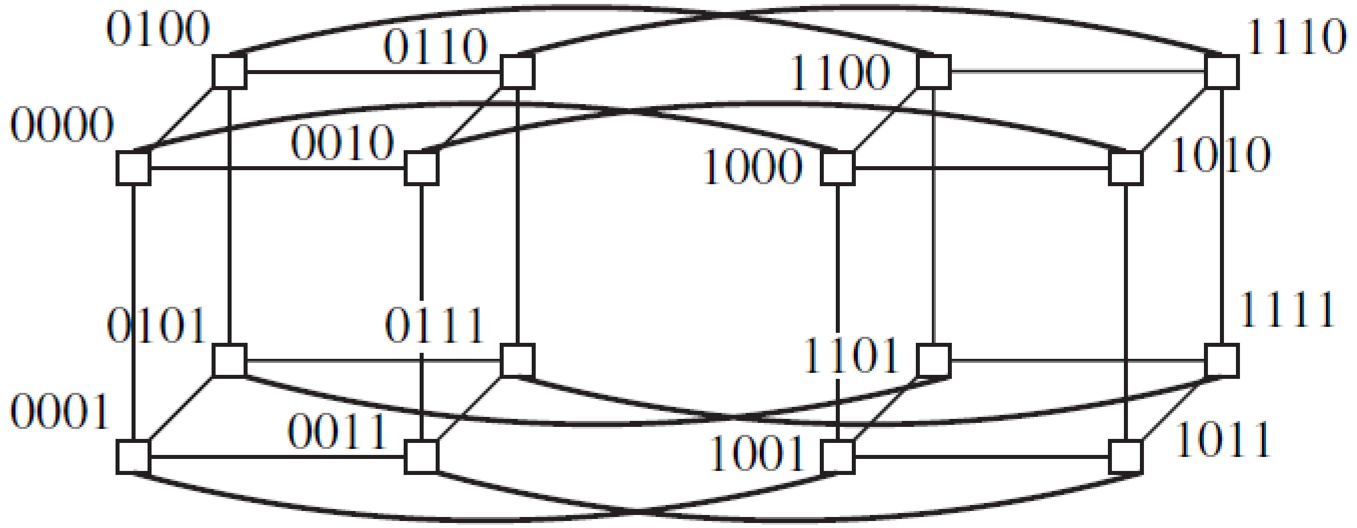
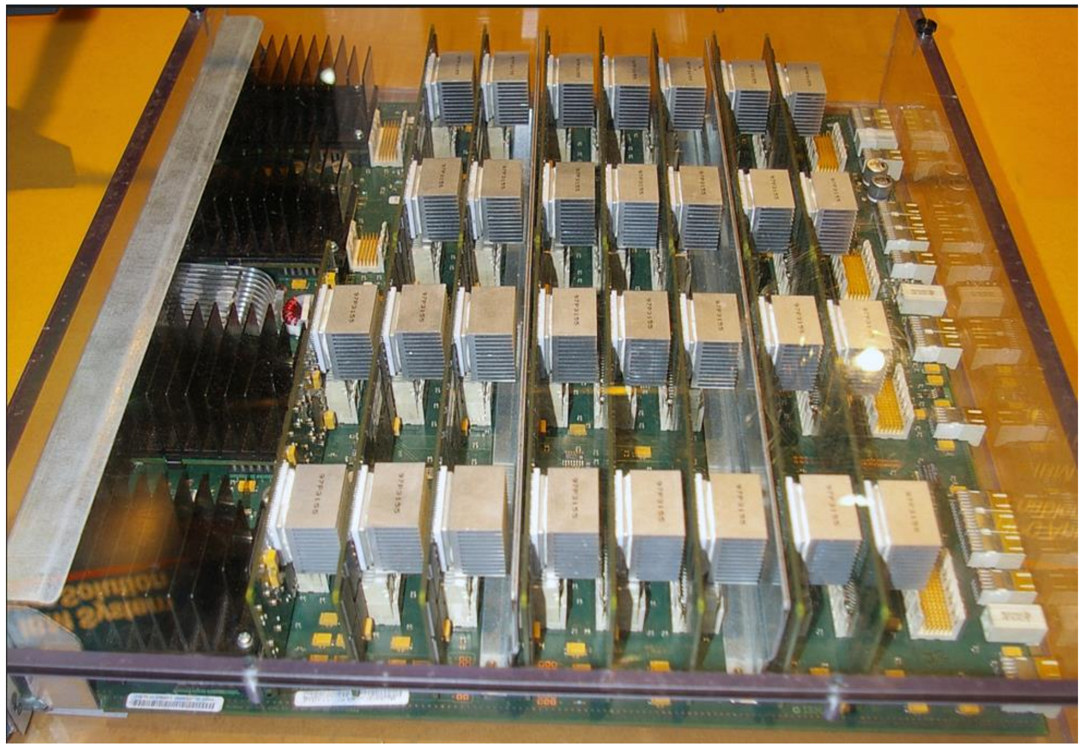
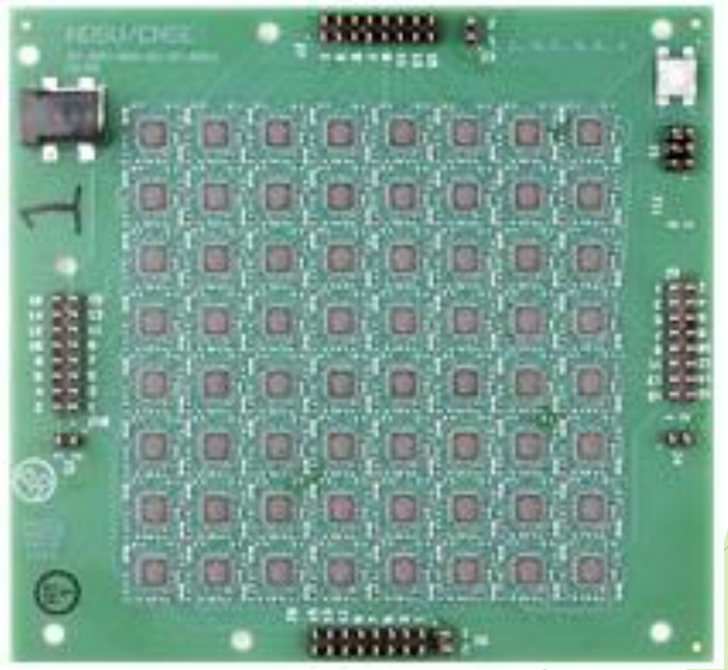


Figure 7: Hypercube of dimension 4 topologies for multicomputer shared-memory machines.

3. *Array processors*

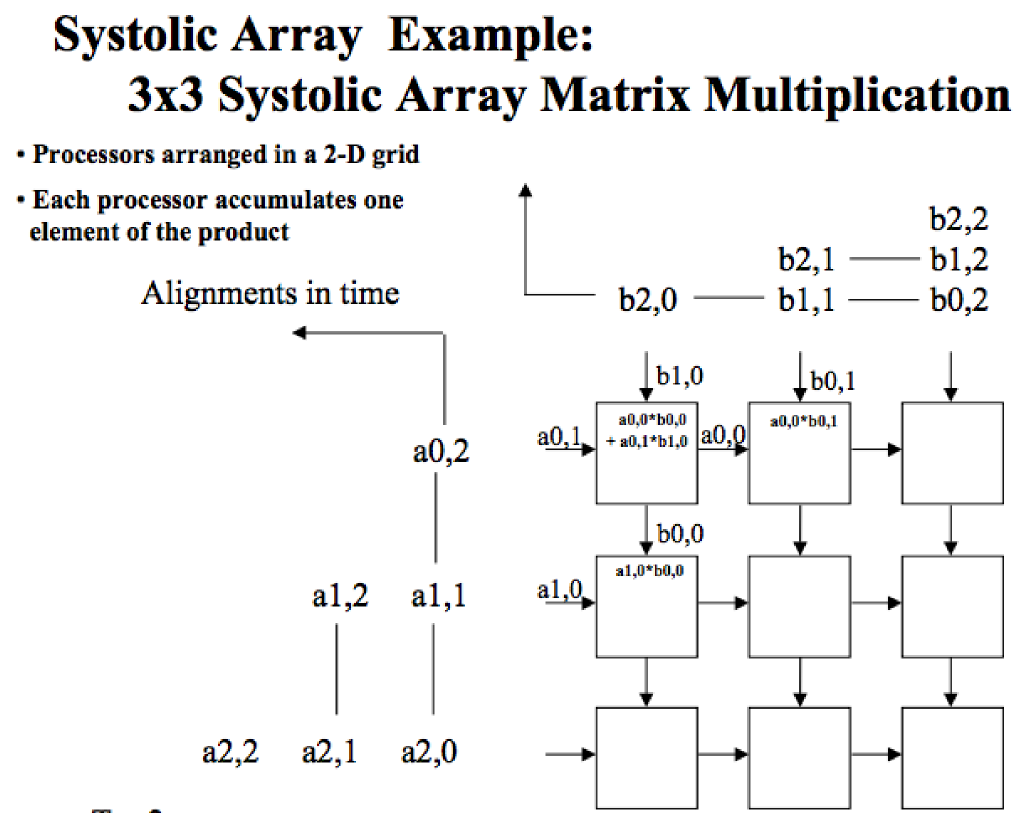
- belong to a class of parallel computers that are physically co-located, are very tightly coupled, and have a common system clock (but may not share memory and communicate by passing data using messages).





3. *Array processors*

- Array processors and systolic arrays that perform tightly synchronized processing and data exchange in lock-step for **applications such as Digital Signal Processing (DSP) and image processing** belong to this category. These applications usually involve a large number of iterations on the data. This class of parallel systems has a very niche market.



Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- **Processors arranged in a 2-D grid**
- **Each processor accumulates one element of the product**

Figure 8: 3x3 Systolic Array Matrix Complication

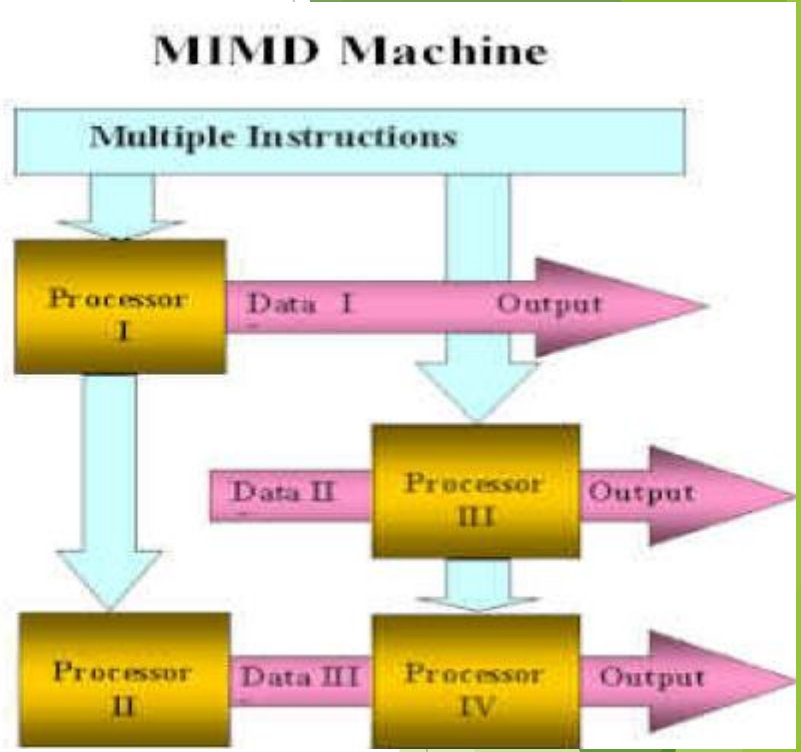
- As compared to UMA systems and array processors, NUMA and message-passing multicomputer systems are **less suitable when the degree of granularity** of accessing shared data and communication **is very fine**.

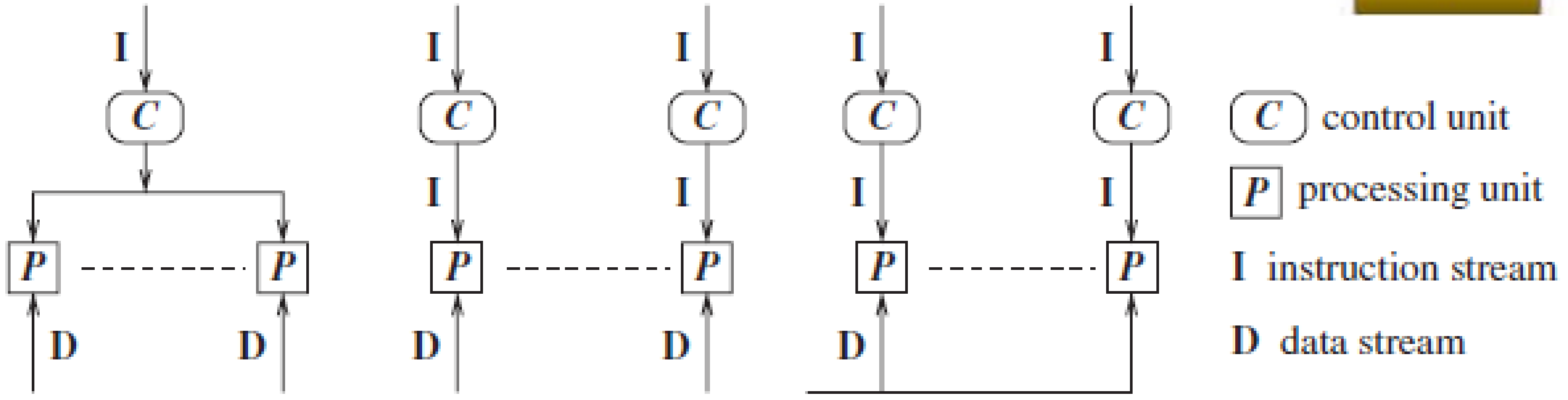
Let's see the difference between UMA and NUMA:

S.NO	UMA	NUMA
1.	UMA stands for Uniform Memory Access.	NUMA stands for Non-uniform Memory Access.
2.	In Uniform Memory Access, Single memory controller is used.	In Non-uniform Memory Access, Different memory controller is used.
3.	Uniform Memory Access is slower than non-uniform Memory Access.	Non-uniform Memory Access is faster than uniform Memory Access.
4.	Uniform Memory Access has limited bandwidth.	Non-uniform Memory Access has more bandwidth than uniform Memory Access.
5.	Uniform Memory Access is applicable for general purpose applications and time-sharing applications.	Non-uniform Memory Access is applicable for real-time applications and time-critical applications.
6.	In uniform Memory Access, memory access time is balanced or equal.	In non-uniform Memory Access, memory access time is not equal.
7.	There are 3 types of buses used in uniform Memory Access which are: Single, Multiple and Crossbar.	While in non-uniform Memory Access, There are 2 types of buses used which are: Tree and hierarchical.

Flynn’s taxonomy

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data stream (SIMD)
- Multiple instruction stream, multiple data stream (MIMD)
- Multiple instruction stream, single data stream (MISD)





(a) SIMD (b) MIMD (c) MISD

The big Question

Can you examine a situation and determine which processor is more suited for the situation?

Coupling, parallelism, concurrency, and granularity

Coupling

The degree of coupling among a set of modules, whether hardware or software, is measured in terms of the **interdependency and binding and/or homogeneity among the modules.**

- **Tightly coupled multiprocessors (with UMA shared memory):**

These may be either switch-based (e.g., NYU Ultracomputer, RP3) or bus-based (e.g., Sequent, Encore).

- **Tightly coupled multiprocessors (with NUMA shared memory or that communicate by message passing):**

Examples are the SGI Origin 2000 and the Sun Ultra HPC servers (that communicate via NUMA shared memory), and the hypercube and the torus (that communicate by message passing).

Coupling – (continued)

- **Loosely coupled multicomputers (without shared memory) physically collocated:**
  - These may be bus-based (e.g., NOW connected by a LAN or Myrinet card) or using a more general communication network, and **the processors may be heterogenous**. In such systems, processors **neither share memory nor have a common clock**, and hence **may be classified as distributed systems** – however, the processors are very close to one another, which **is characteristic of a parallel system**. As the communication latency may be significantly lower than in wide-area distributed systems, the solution approaches to various problems may be different for such systems than for wide-area distributed systems.
- **Loosely coupled multicomputers (without shared memory and without common clock) that are physically remote:**
  - These correspond to the conventional notion of distributed systems.

Coupling – (continued)

- **Parallelism or speedup of a program on a specific system:**

An measure of the relative speedup of a specific program, on a given machine. The speedup **depends on the number of processors** and the mapping of the code to the processors. It is expressed as the ratio of the time T1 with a single processor, to the time Tn with n processors.

- **Parallelism within a parallel/distributed program:**

An aggregate measure of **the percentage of time that all the processors are executing CPU instructions productively**, as opposed to waiting for communication (either via shared memory or message-passing) operations to complete.

- **Concurrency of a program:**

Is used in the context of distributed programs. The *parallelism/concurrency* in a parallel/distributed program can be measured by **the ratio of the number of local (non-communication and non-shared memory access) operations to the total number of operations**, including the communication or shared memory access operations.

Coupling – (continued)

- **Granularity of a program:**

The ratio of the amount of **computation to the amount of communication** within the parallel/distributed program is termed as *granularity*. If the degree of parallelism is **coarse-grained (fine-grained),** there are relatively many **more (fewer)** productive CPU instruction executions, compared to the number of times the processors communicate either via shared memory or message-passing and wait to get synchronized with the other processors.

- Programs with **fine-grained parallelism** are **best suited for tightly coupled systems**. These typically include SIMD and MISD architectures, tightly coupled MIMD multiprocessors (that have shared memory), and loosely coupled multicomputers (without shared memory) that are physically colocated.
- If programs with **fine-grained parallelism were run over loosely coupled** multiprocessors that are physically remote, the latency delays for the frequent communication over the WAN would **significantly degrade the overall throughput.** As a result, it follows that on such loosely coupled multicomputers, programs with a coarse-grained communication/message-passing granularity will incur substantially less overhead.

Coupling – (continued)

Operation System:

- The operating system running **on loosely coupled processors** (i.e., heterogenous and/or geographically distant processors), which are themselves **running loosely coupled software** (i.e., software that is heterogenous), is classified as **a *network operating system.*** (Windows NT, Windows Server, MAC OS, unix)
  - In this case, the application cannot run any significant distributed function that is not provided by the application layer of the network protocol stacks on the various processors.
- The operating system running on **loosely coupled processors**, which are **running tightly coupled software** (i.e., the middleware software on the processors is homogenous), is classified as a ***distributed operating system.*(**IRIX, DYNIX , Solaris Mach/OS)
- The operating system running on **tightly coupled processors**, which are themselves running **tightly coupled software**, is classified as a ***multiprocessor operating system.*** Such a parallel system can run sophisticated algorithms contained in the tightly coupled software.(Intel Nehalem, AMD Opteron, ARM Cortex, Oracle (Sun) UltraSparc)

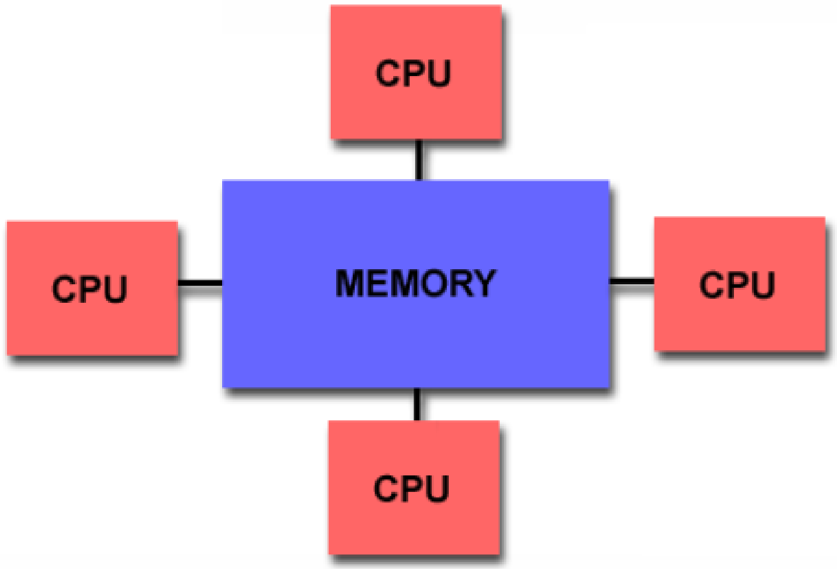
Shared Memory and Message Passing

Parallel computers can also be classified according to memory access.

- Shared memory computers.
- Message-passing (distributed memory) computers.
- Multi-processor computers
- Multi-computers
- Clusters
- Grids

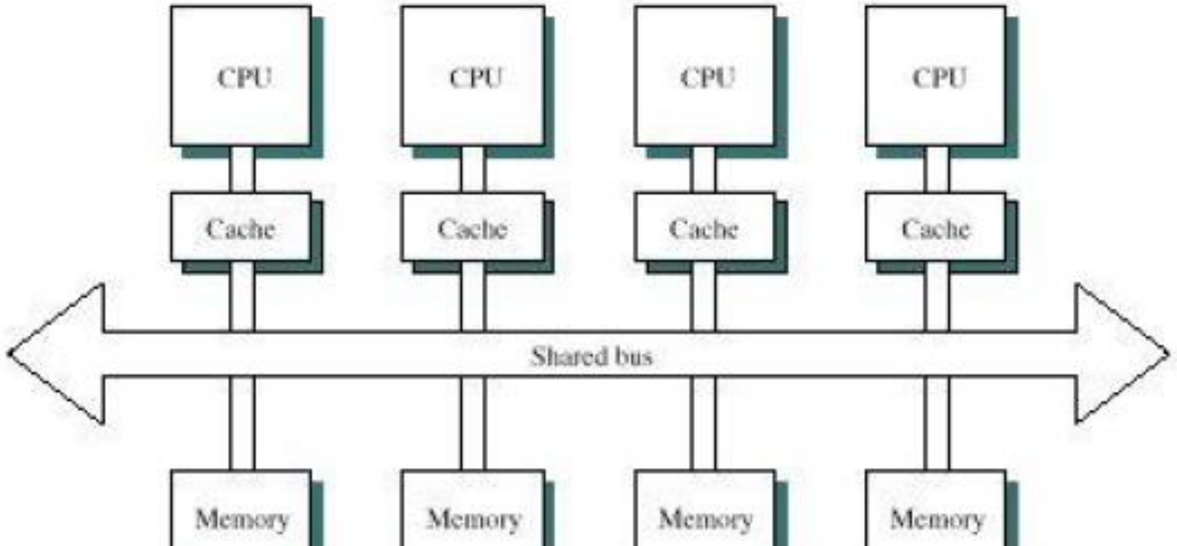
Shared Memory Computers

- All processors have **access to all memory as a global address space** Multiple processors can operate independently, **but share the same memory Resources**
- **Changes** in a memory location effected by one processor **are visible to all other processors**.
- Two classes of shared memory machines:
  - UMA and NUMA

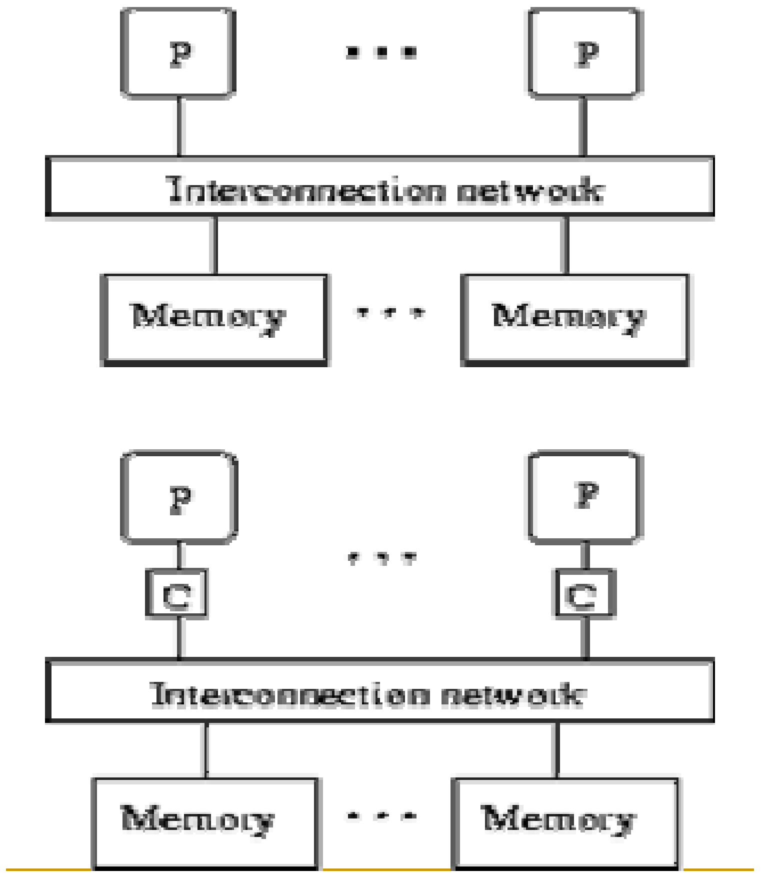


Uniform Memory Access (UMA)

- Most commonly represented today by Symmetric Multiprocessor (SMP) machines.
- Identical processors.
- Equal access and access time to memory.
- Sometimes called CC-UMA – Cache Coherent UMA.
- **Cache Coherence:**
  - If one processor updates a location in shared memory, all the other processors know about the update.
  - Cache coherence is accomplished at the hardware level.

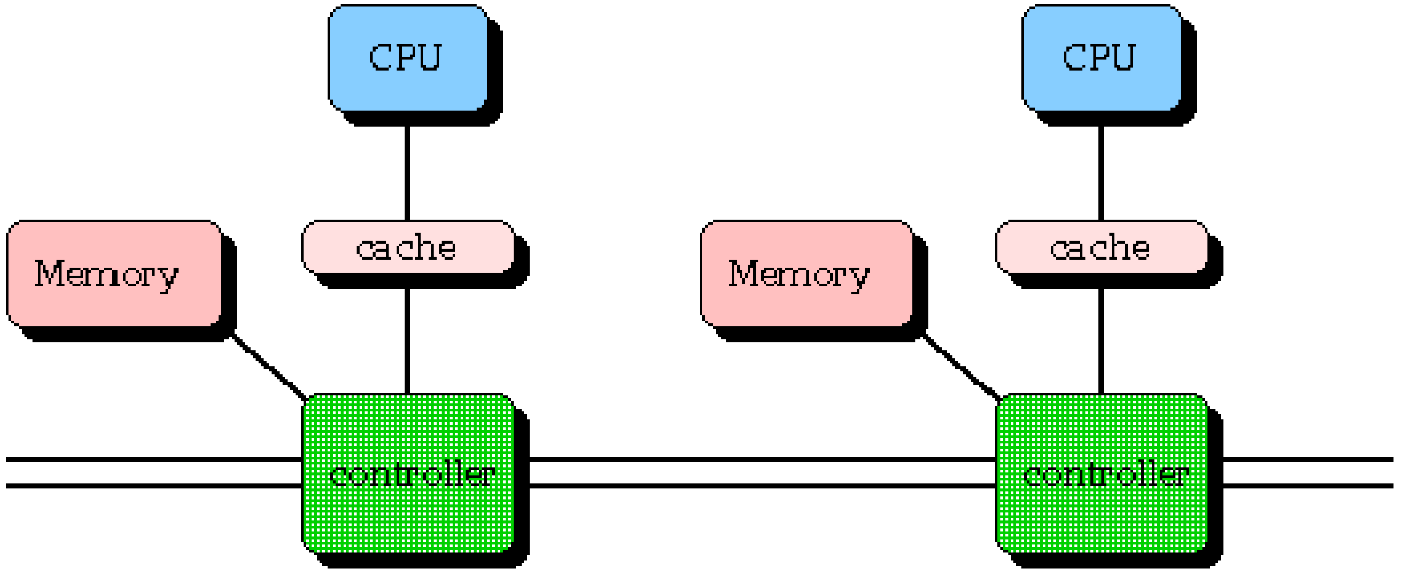


UMA –with and without caches



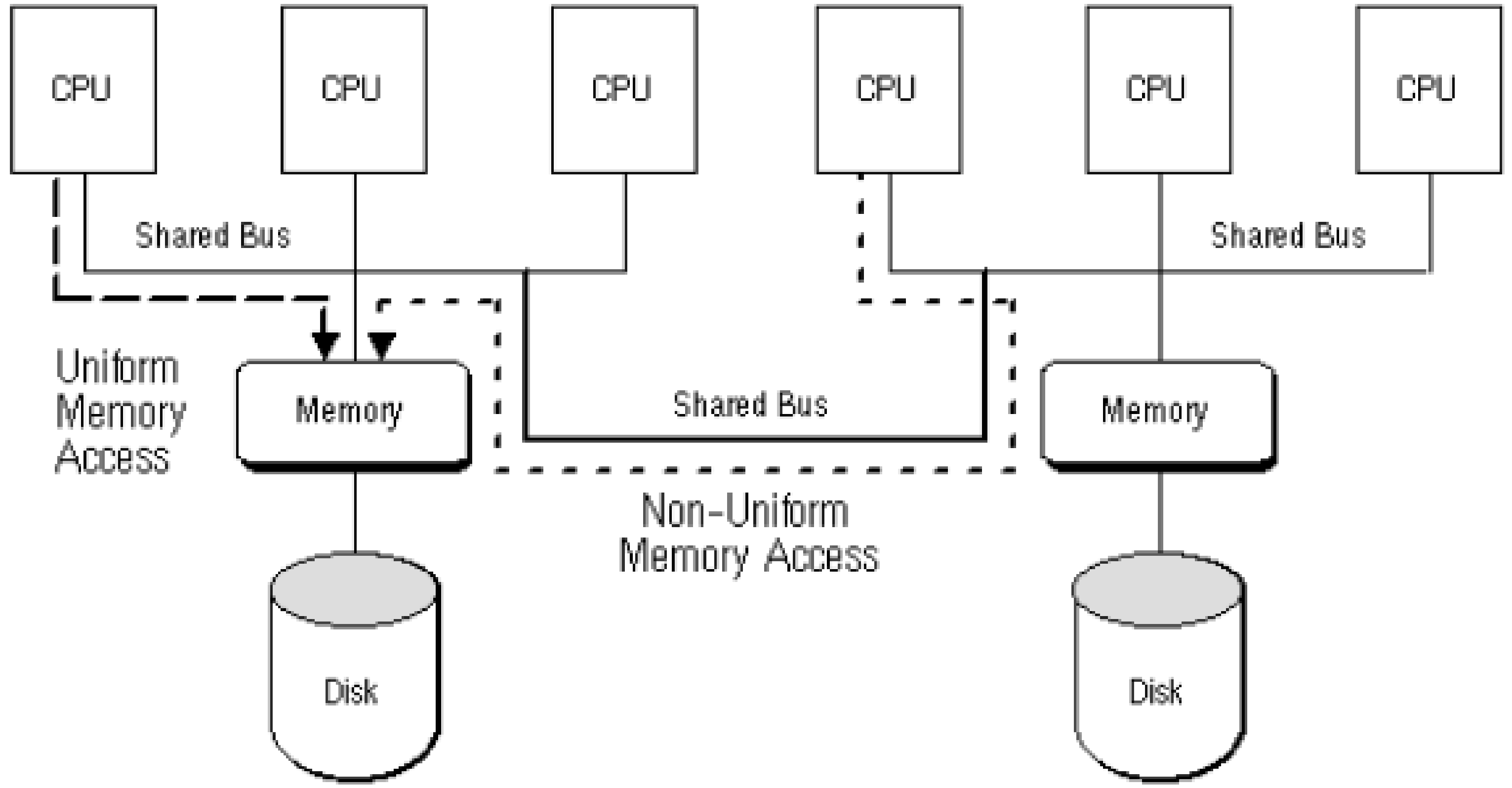
Nonuniform Memory Access (NUMA)

- Often made by physically linked two or more Symmetric Multiprocessor (SMP) .
- One SMP can directly access memory of another Symmetric Multiprocessor (SMP) .
- Not all processors have equal access time to all memories.
- Memory access across link is slower.
- If cache coherence is maintained, they may also be called CC-NUMA.



SMP (symmetric multiprocessing) is the processing of programs by multiple processors that share a common operating system and memory. In symmetric (or "tightly coupled") multiprocessing

Hybrid Machine (UMA & NUMA)



Advantages and Disadvantage of Shared Memory Machines

Advantage:

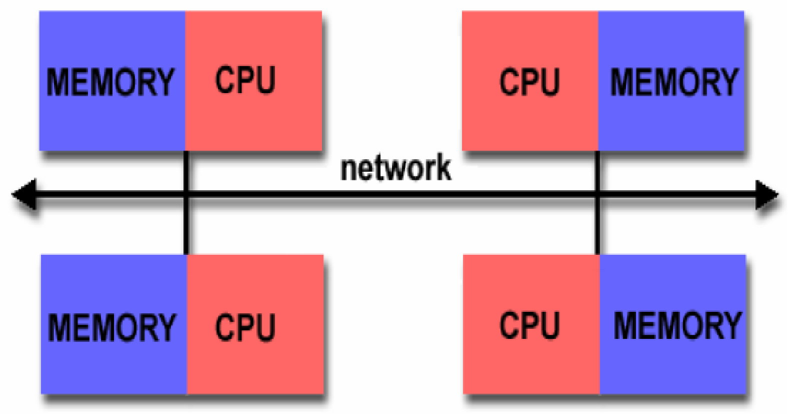
- Global address space **provides a user-friendly programming environment** to memory access.
- Data sharing between tasks is both **fast and uniform** due to proximity of memory to CPUs.

Disadvantage:

- **Lack of scalability between memory and CPUs**: Adding processors can geometrically increase traffic on the shared memory-CPU path and for cache coherence management.
- Programmer's responsibility for synchronization constructs (correct access to memory).
- **Expensive** to design shared memory computers **with increasing numbers of processors**

Distributed Memory Computers

- Processors have their **own local memory**.
- **It requires** a **communication network** to connect inter-processor memory.
- Memory addresses in one processor do not map to another processor – **no concept of global address space across all processors.**
- The concept of cache coherence does not apply
- Data are **exchanged** explicitly through **message passing.**
- Synchronization between tasks is the programmer's responsibility



Advantages and Disadvantage of Distributed Memory Machines

Advantage:

- Memory is **scalable** with the number of processors.
- Increase the number of processors, the size of memory increases proportionally.
- Each processor can rapidly access its own memory without interference and without the overhead incurred with trying to maintain cache coherence.
- Cost effectiveness: can use commodity, off-the shelf processors and networking.

Disadvantage:

- Difficult to program: Programmer has to handle data communication between Processors.
- Nonuniform memory access (NUMA) times
- It may be **difficult to map existing data structures**, based on global memory, to distributed memory organization

Architectural Design Tradeoffs

	Shared memory	Distributed memory
Programmability	easier	harder
Scalability	harder	easier

Distributed systems challenges from a system perspective

The following functions must be addressed when designing and building a distributed system:

- **Communication:** involves **designing appropriate mechanisms for communication among the processes in the network.** Some example mechanisms are: remote procedure call (RPC), remote object invocation (ROI), message-oriented communication versus stream-oriented communication.
- **Processes:** Some of the issues involved are: management of processes and threads at clients/servers; code migration; and the design of software and mobile agents.
- **Naming:** Devising easy to use and robust schemes for names, identifiers, and addresses is essential for locating resources and processes in a transparent and scalable manner. Naming in mobile systems provides additional challenges because naming cannot easily be tied to any static geographical topology.
- **Synchronization:** Mechanisms for synchronization or coordination among the processes are essential.
- **Data storage and access:** Schemes for data storage, and implicitly for accessing the data in a fast and scalable manner across the network are important for efficiency. Traditional issues such as file system design have to be reconsidered in the setting of a distributed system.

Distributed systems challenges from a system perspective (Continued)

- **Consistency and replication: To avoid bottlenecks, to provide fast access to data**, and to provide scalability, replication of data objects is highly desirable. This leads to issues of managing the replicas, and dealing with consistency among the replicas/caches in a distributed setting. A simple example issue is deciding the level of granularity (i.e., size) of data access.
- **Fault tolerance:** Fault tolerance requires **maintaining correct and efficient operation** in spite of any failures of links, nodes, and processes. Process resilience, reliable communication, distributed commit, checkpointing and recovery, agreement and consensus, failure detection, and self-stabilization are some of the mechanisms to provide fault-tolerance.
- **Security:**Distributed systems security involves various aspects of cryptography, secure channels, access control, key management – generation and distribution, authorization, and secure group management.

Distributed systems challenges from a system perspective (Continued)

- **Applications Programming Interface (API) and transparency:** The API for communication and other specialized services is important for **the ease of use and wider adoption of the distributed systems** services by non-technical users.

- **Scalability and modularity:** The algorithms, data (objects), and services **must be as distributed as possible**. Various techniques such as replication, caching and cache management, and asynchronous processing help to achieve scalability.

Technologies for Network-based Systems

- Multicore CPUs and Multithreading Technologies.
- GPU Computing to Exascale and Beyond.
- Memory, Storage, and Wide-Area Networking.
- Virtual Machines and Virtualization Middleware.
- Data Center Virtualization for Cloud Computing.

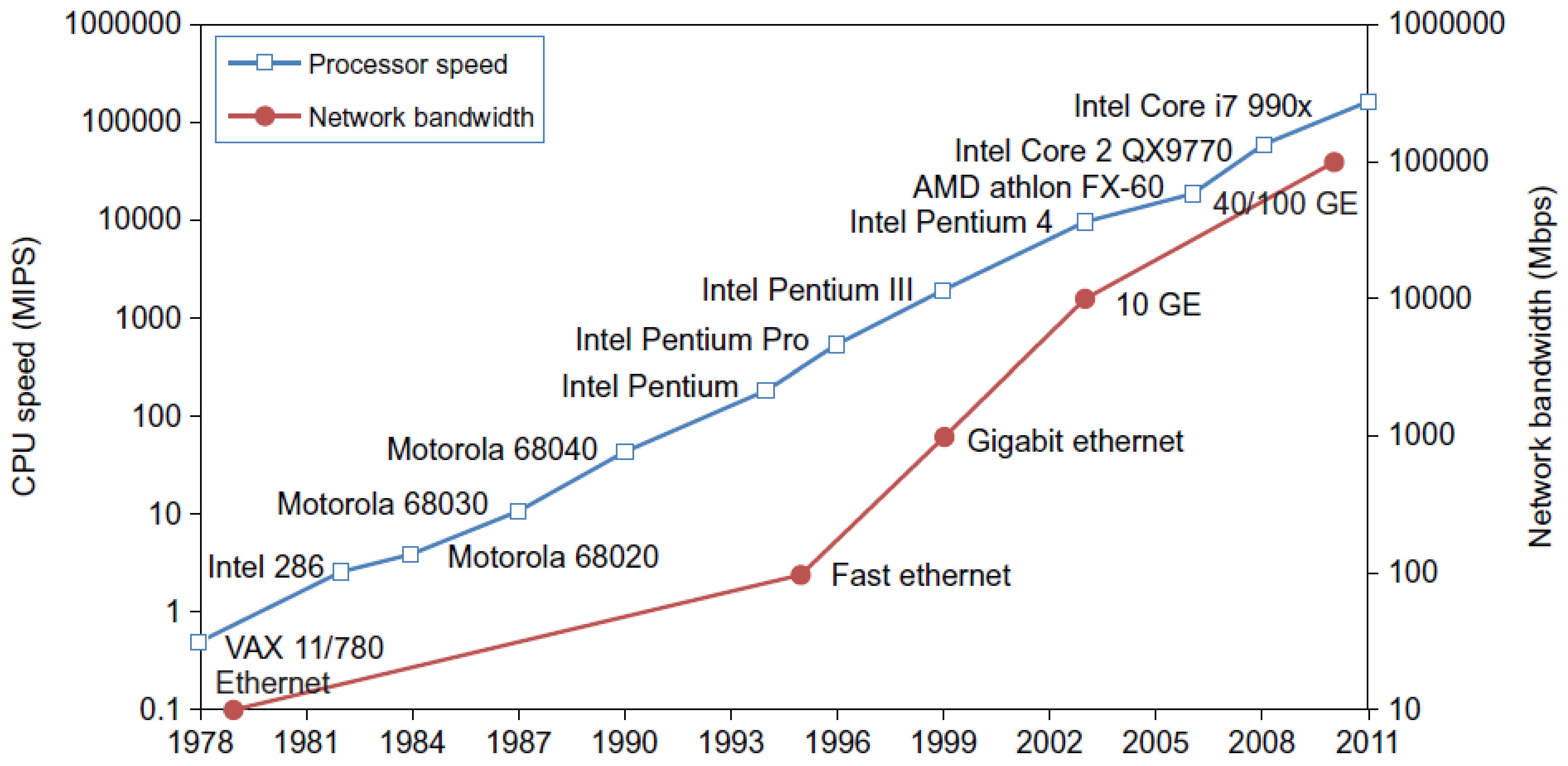
Technologies for Network-based Systems - (Continued)

Multicore CPUs and Multithreading Technologies.

- Considering the growth of component and network technologies over the past 30 years. They are crucial to the development of High Performance Computing (HPC) and High Throughput Computing (HTC) systems.
- processor speed is measured:
  - in millions of instructions per second (MIPS)
- network bandwidth is measured:
  - in megabits per second (Mbps) or gigabits per second (Gbps). The unit GE refers to 1 Gbps Ethernet bandwidth.

Multicore CPUs and Multithreading Technologies – (Continued)

- **Advances in CPU Processors:**

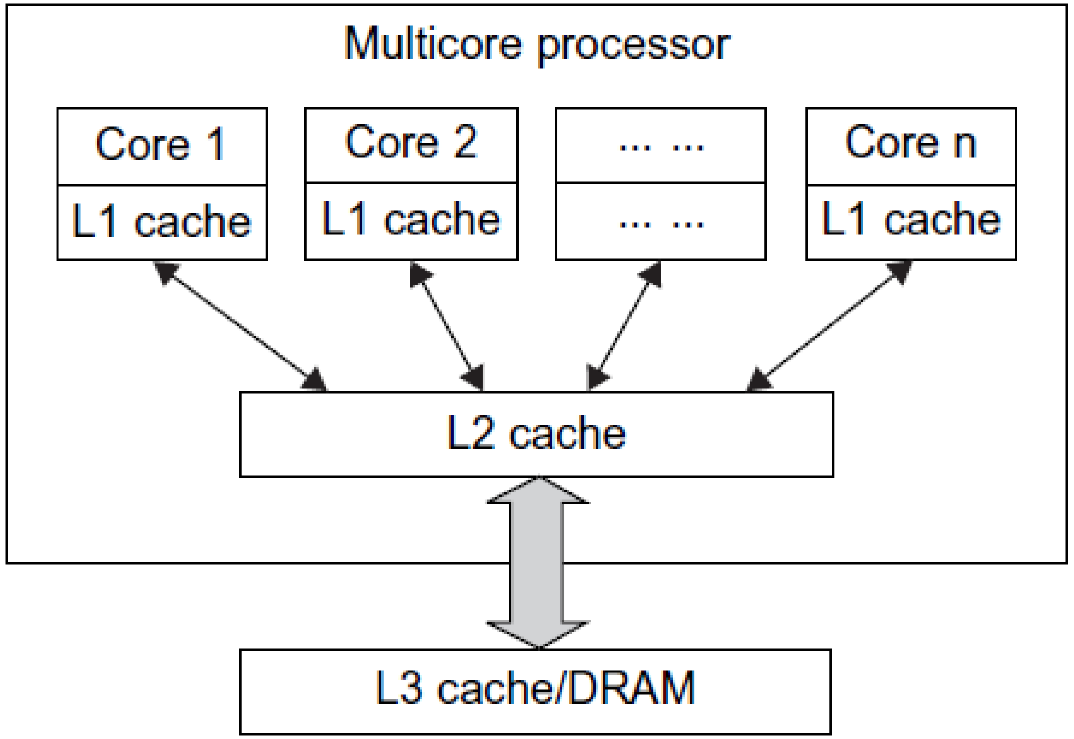


Multicore CPUs and Multithreading Technologies – (Continued)

- **Multicore CPU and Many-Core GPU Architectures:**
  - CPUs are general purpose integrated circuit / IC that are designed to perform a variety of operations. i.e, Watching movie, listening music , making spread-sheets , creating word documents, playing games etc.
  - On the other hand, GPUs are specific purpose circuits that are designed to perform specific tasks like rendering videos, playing high-end graphics games (which needs parallel processing).
  - The Data-level parallelism (DLP) and Task-level parallelism (TLP) are highly explored in graphics processing units (GPUs) that adopt a many-core architecture with hundreds to thousands of simple cores.

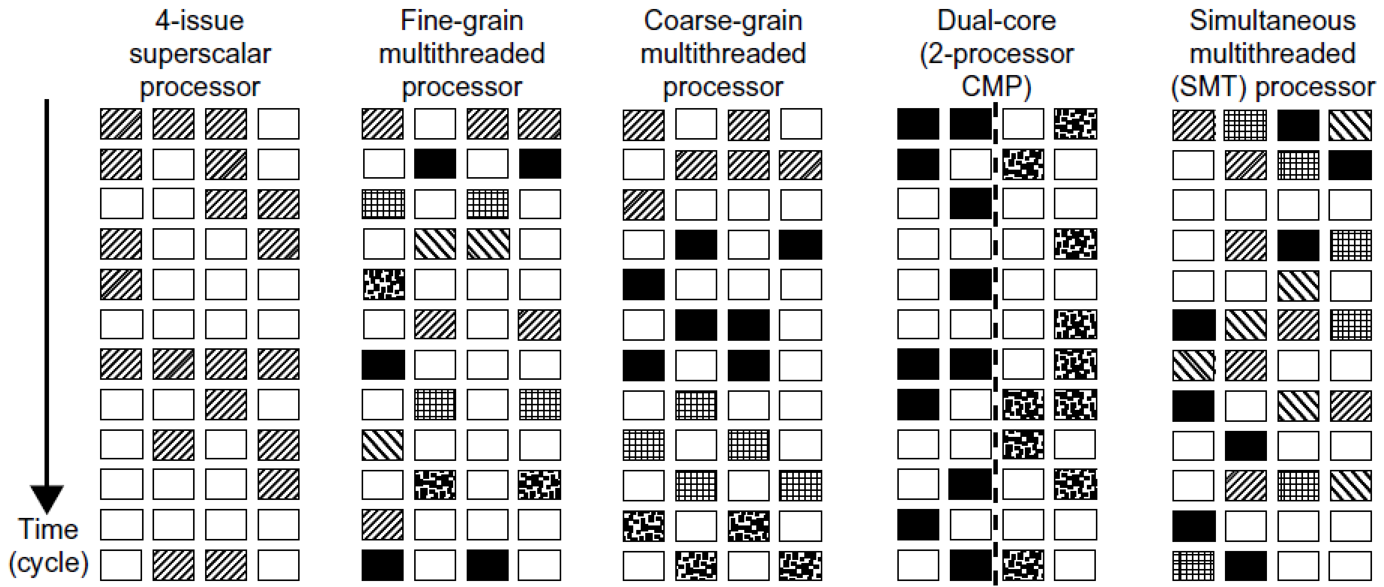
Multicore CPUs and Many-core GPU Architecture – (Continued)

- Multiple cores are housed in the same chip with an L2 cache that is shared by all cores.
- In the future, multiple chip multiprocessors (CMPs) could be built on the same CPU chip with even the L3 cache on the chip.
- Multicore CPUs may increase from the tens of cores to hundreds or more in the future. But the CPU has reached its limit in terms of exploiting massive DLP due to the memory wall problem (Memory access time did not improve much in the past). This has triggered the development of many-core GPUs with hundreds or more thin cores.



Multicore CPUs and Multithreading Technologies - (Continued)

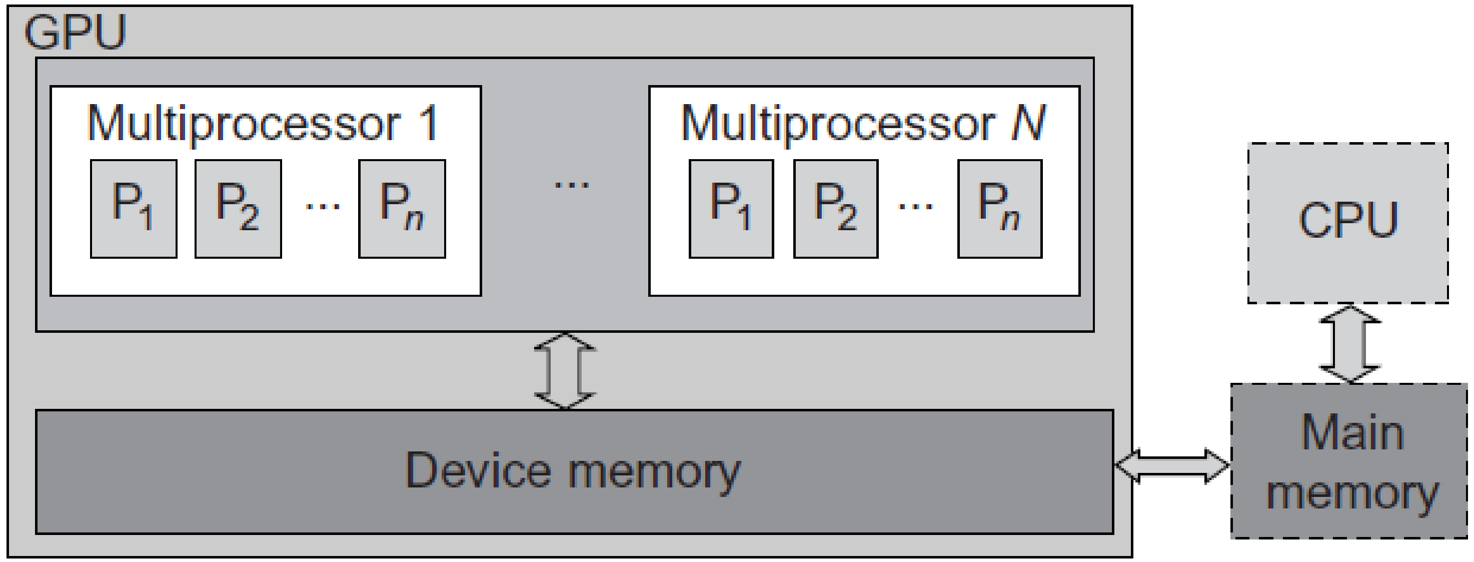
- **Multithreading Technology**
- five independent threads of instructions to four pipelined data paths (functional units) in each of the following five processor categories, from left to right:
  - a four-issue superscalar processor,
  - a fine-grain multithreaded processor,
  - a coarse-grain multithreaded processor,
  - a two-core CMP,
  - a simultaneous multithreaded (SMT) processor.


4-issue superscalar processor
Fine-grain multithreaded processor
Coarse-grain multithreaded processor
Dual-core (2-processor CMP)
Simultaneous multithreaded (SMT) processor
Time (cycle)

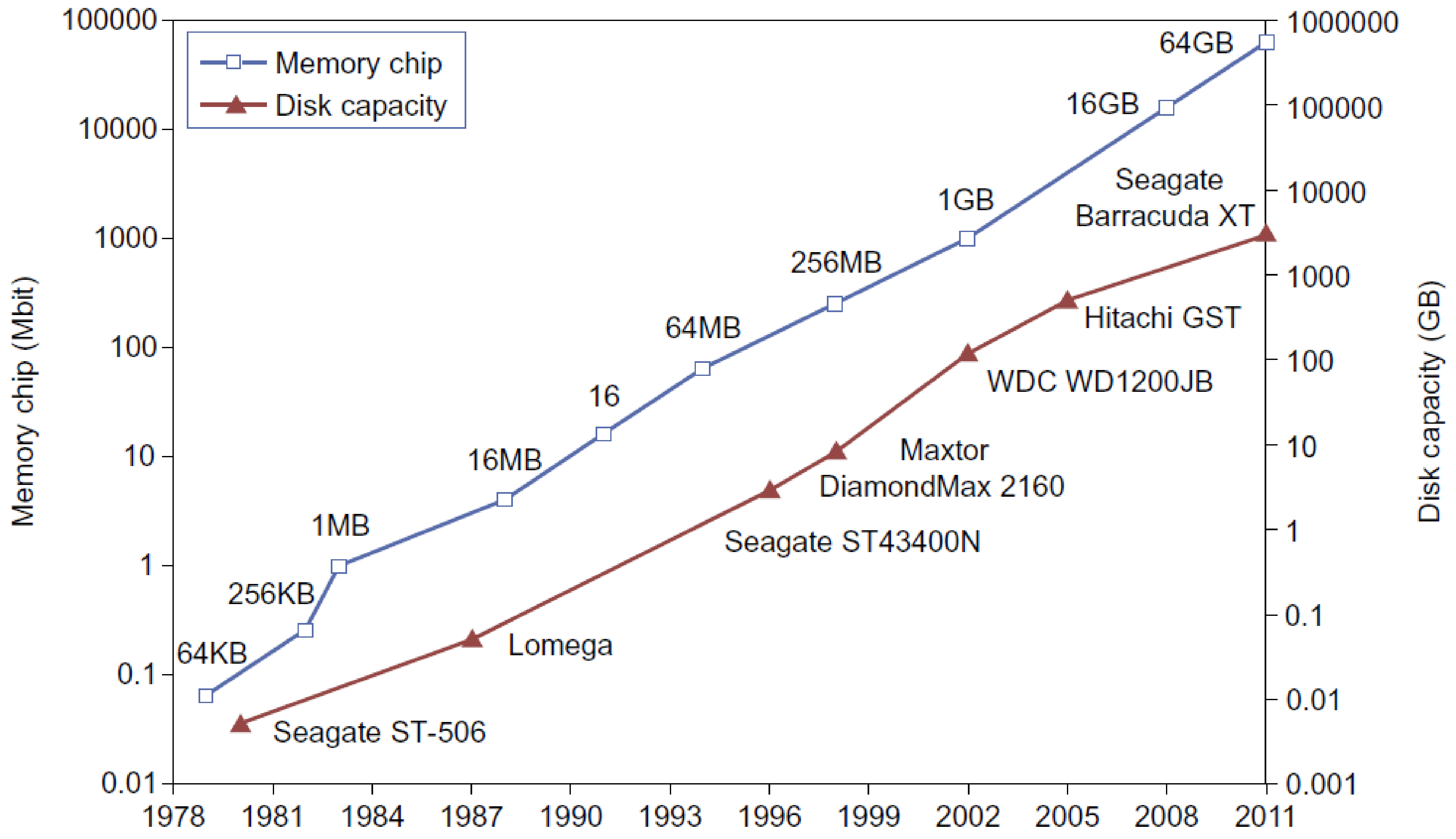
GPU Computing to Exascale and Beyond

- A GPU is a **graphics coprocessor** or accelerator mounted on a computer's graphics card or video card.
- It **offloads the CPU from tedious graphics tasks** in video editing applications.
- GPU chips can process **a minimum of 10 million polygons per second**, and are used in nearly every computer on the market today. Some GPU features were also integrated into certain CPUs.
- GPUs have a throughput architecture that exploits massive parallelism by executing many concurrent threads slowly, instead of executing a single long thread in a conventional microprocessor very quickly.
- parallel GPUs or GPU clusters have been garnering a lot of attention against the use of CPUs with limited parallelism. General-purpose computing on GPUs, known as GPGPUs, have appeared in the HPC field. NVIDIA's CUDA model was for HPC using GPGPUs.
- Modern GPUs are not restricted to accelerated graphics or video coding.
- They are used in HPC systems to power supercomputers with massive parallelism at multicore and multithreading levels.
- The CPU instructs the GPU to perform massive data processing.
- The bandwidth must be matched between the on-board main memory and the on-chip GPU memory.

Interconnection between GPU & CPU



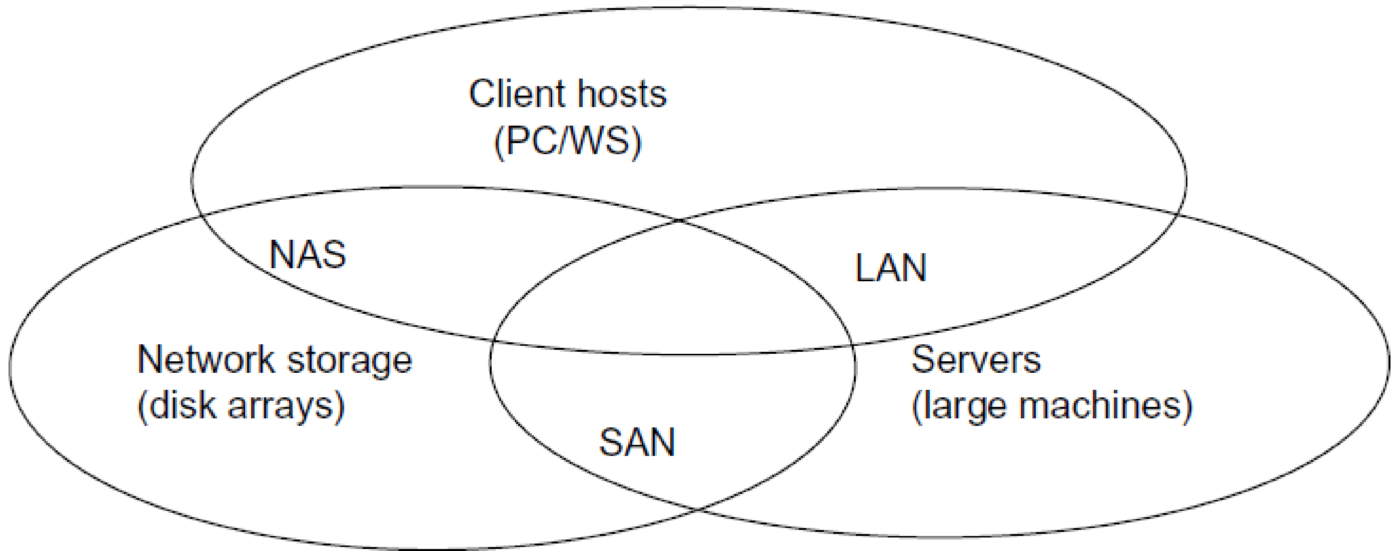
Memory, Storage, and Wide-Area Networking



Memory, Storage, and Wide-Area Networking (Continued)

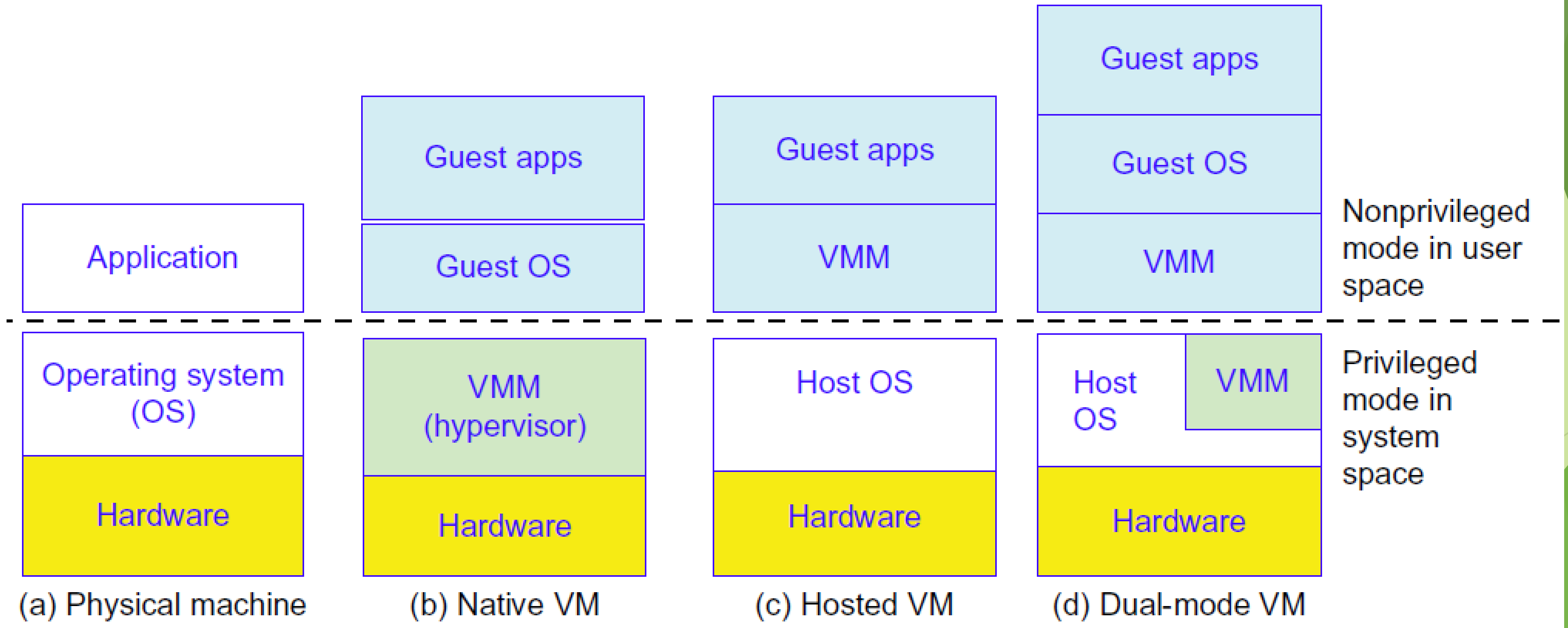
System-Area Interconnects

- Three types of networks often appear in a large cluster built with commercial network components.
  - LAN: typically is used to connect client hosts to big servers.
  - storage area network (SAN): connects servers to network storage such as disk arrays.
  - Network attached storage (NAS): connects client hosts directly to the disk arrays.
- If no large distributed storage is shared:
  - a small cluster could be built with a multiport Gigabit Ethernet switch plus copper cables to link the end machines.



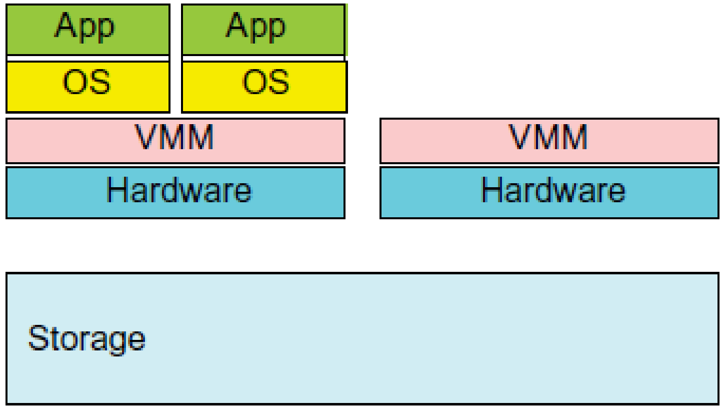
Virtual Machines and Virtualization Middleware

- Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.

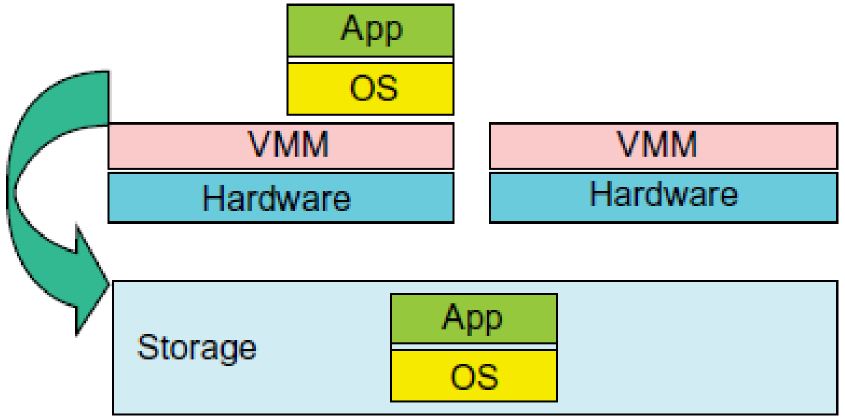


Virtual Machines and Virtualization Middleware (continued)

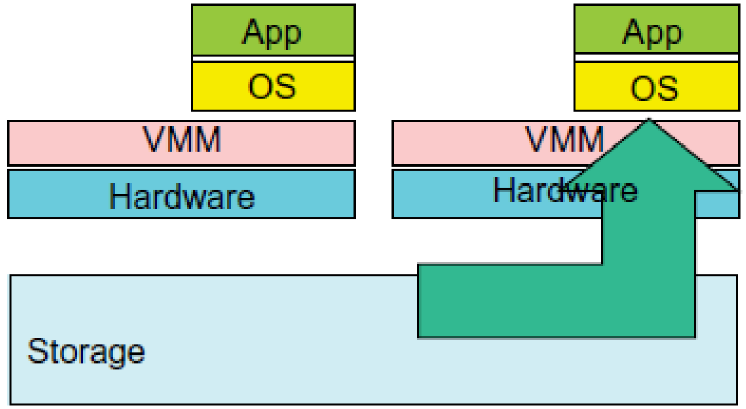
- (a) the VMs can be multiplexed between hardware machines
- (b) a VM can be suspended and stored in stable storage
- (c) a suspended VM can be resumed or provisioned to a new hardware platform
- (d) a VM can be migrated from one hardware platform to another



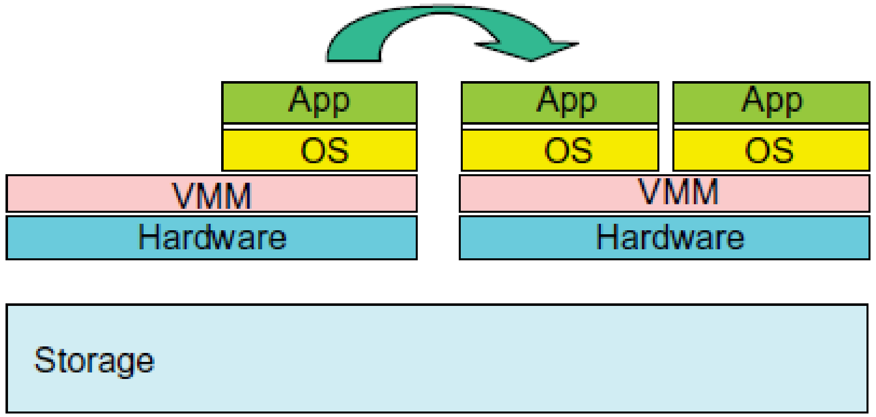
(a) Multiplexing



(b) Suspension (storage)



(c) Provision (resume)



(d) Life migration

Data Center Virtualization for Cloud Computing

- Low-cost terabyte disks and Gigabit Ethernet are used to build data centers.
- Datacenter design emphasizes the performance/price ratio over speed performance alone. In other words, storage and energy efficiency are more important than shear speed performance.
- A large datacenter may be built with thousands of servers.
- Smaller datacenters are typically built with hundreds of servers.
- High-end switches or routers may be too cost-prohibitive for building data centers. Thus, using high-bandwidth networks may not fit the economics of cloud computing.
- cloud computing is enabled by the convergence of technologies in four areas:
  - (1) hardware virtualization and multi-core chips,
  - (2) utility and grid computing,
  - (3) SOA, Web 2.0, and WS mashups, and
  - (4) atonomic computing and data center automation.

SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

- Distributed and cloud computing systems are built over a **large number of autonomous computer nodes**.
- These node machines are **interconnected by SANs, LANs, or WANs** in a hierarchical manner.
- With today's networking technology, **a few LAN switches can** easily connect **hundreds of machines** as a working cluster.
- A WAN can connect many local clusters to form a very large cluster of clusters. In this sense, one can build a massive system with millions of computers connected to edge networks.

SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING- (Continued)

Massive systems:

In terms of node number, these four system classes may involve hundreds, thousands, or even zillions of computers as participating nodes. These machines work collectively, cooperatively, or collaboratively at various levels. The table entries characterize these four system classes in various technical and application aspects.

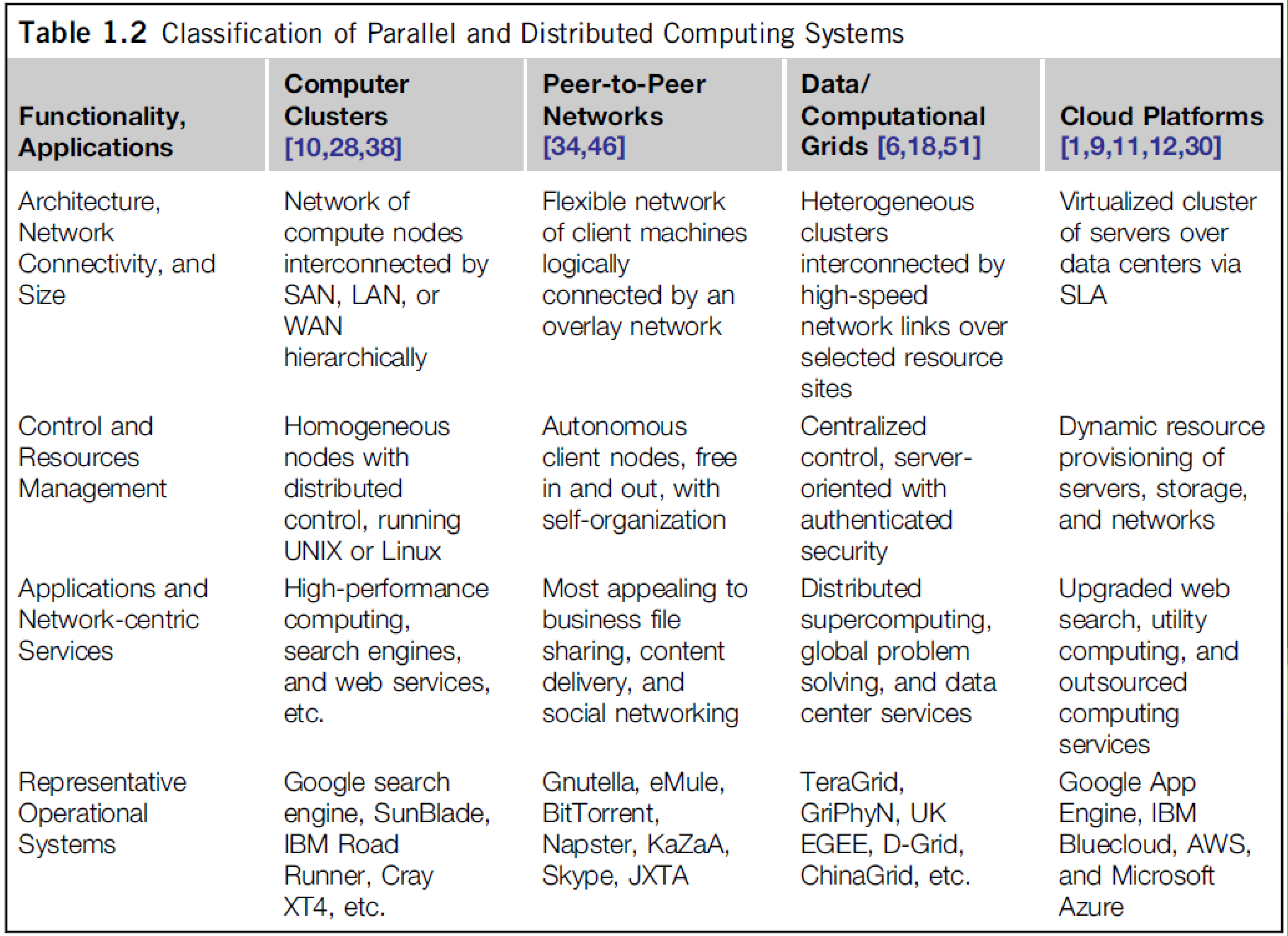
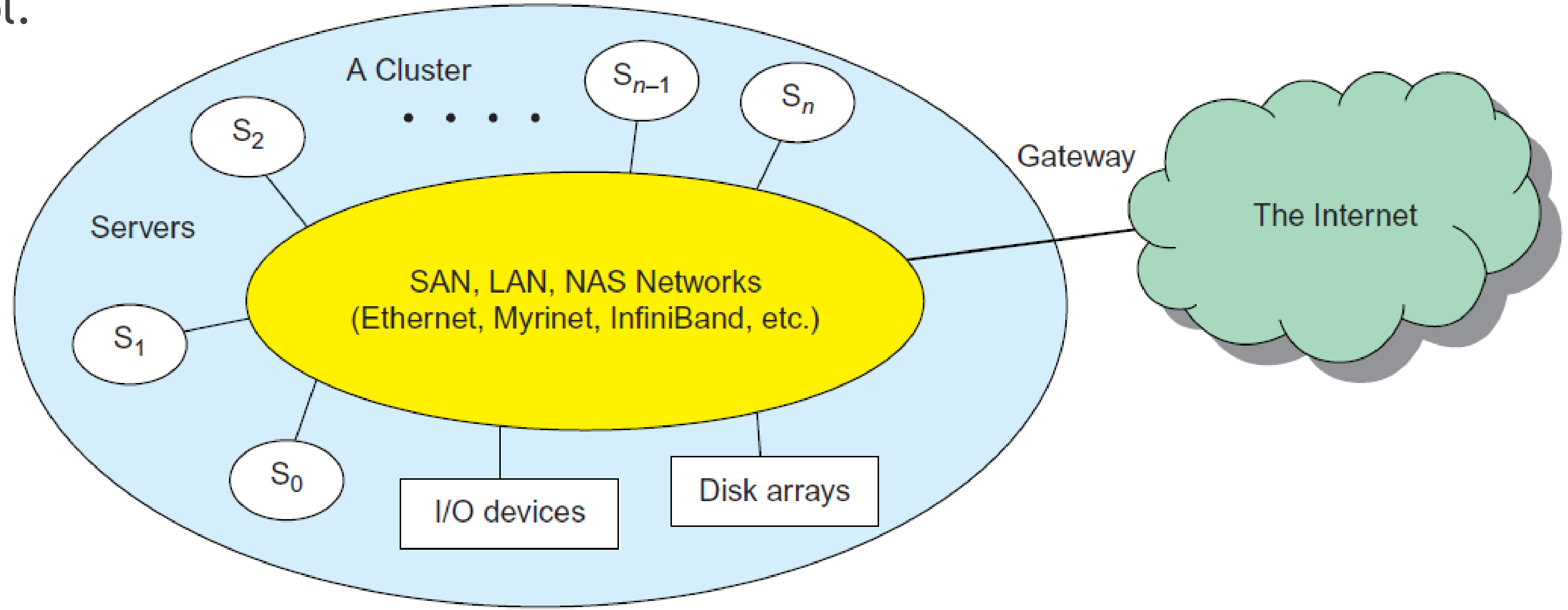


Table 1.2 Classification of Parallel and Distributed Computing Systems

Functionality, Applications	**Computer Clusters** [10,28,38]	**Peer-to-Peer Networks** [34,46]	**Data/ Computational Grids** [6,18,51]	**Cloud Platforms** [1,9,11,12,30]
Architecture, Network Connectivity, and Size	Network of compute nodes interconnected by SAN, LAN, or WAN hierarchically	Flexible network of client machines logically connected by an overlay network	Heterogeneous clusters interconnected by high-speed network links over selected resource sites	Virtualized cluster of servers over data centers via SLA
Control and Resources Management	Homogeneous nodes with distributed control, running UNIX or Linux	Autonomous client nodes, free in and out, with self-organization	Centralized control, server-oriented with authenticated security	Dynamic resource provisioning of servers, storage, and networks
Applications and Network-centric Services	High-performance computing, search engines, and web services, etc.	Most appealing to business file sharing, content delivery, and social networking	Distributed supercomputing, global problem solving, and data center services	Upgraded web search, utility computing, and outsourced computing services
Representative Operational Systems	Google search engine, SunBlade, IBM Road Runner, Cray XT4, etc.	Gnutella, eMule, BitTorrent, Napster, KaZaA, Skype, JXTA	TeraGrid, GriPhyN, UK EGEE, D-Grid, ChinaGrid, etc.	Google App Engine, IBM Bluecloud, AWS, and Microsoft Azure

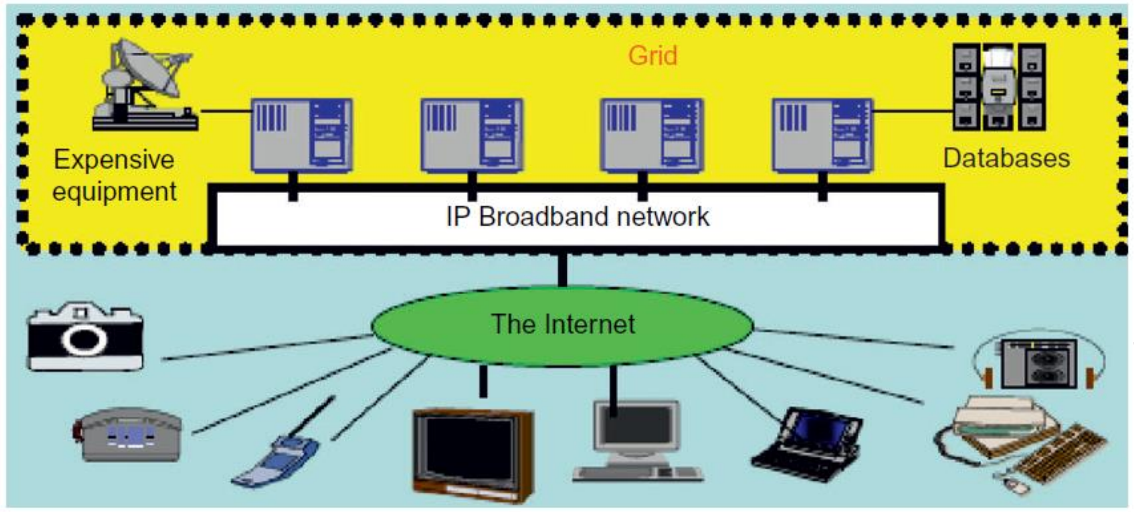
Cluster Architecture:

- Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes.
- The cluster is connected to the Internet via a virtual private network (VPN) gateway.
- The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources.
- Most clusters have loosely coupled node computers. All resources of a server node are managed by their own OS. Thus, most clusters have multiple system images as a result of having many autonomous nodes under different OS control.



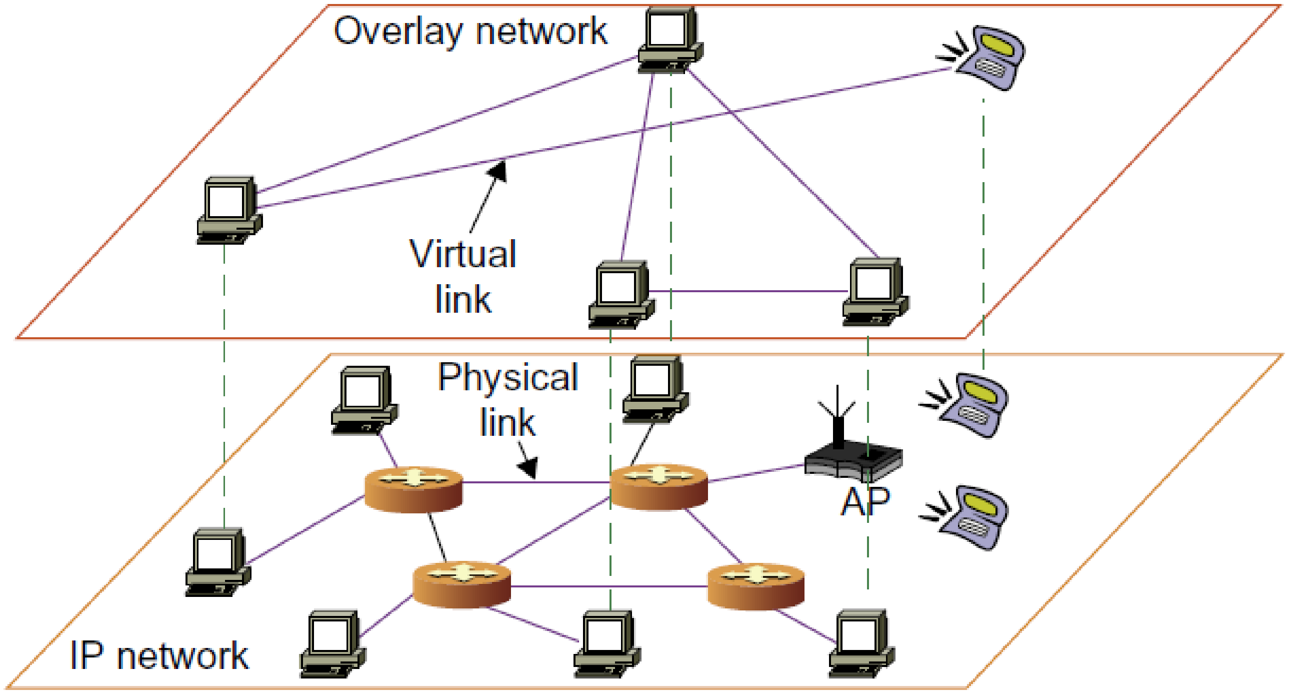
Grid Computing Infrastructures:

- Grid computing is envisioned to **allow close interaction among applications** running on distant computers simultaneously.
- a computing grid offers **an infrastructure that couples** computers, software/middleware, special instruments, and people and sensors together.
- The grid is often constructed across **LAN, WAN, or Internet backbone networks** at a regional, national, or global scale.
- Enterprises or organizations present grids as **integrated computing resources**. They can also be viewed as virtual platforms to support virtual organizations.
- The computers used in a grid are primarily **workstations, servers, clusters, and supercomputers.**
- Personal computers, laptops, can be used as access devices to a grid system.



Peer-to-Peer Network Families:

- The P2P architecture **offers** a distributed model of **networked systems**.
- Every node acts as **both a client and a server**, providing part of the system resources.
- Peer machines are simply client computers connected to the Internet.
- All client machines **act autonomously to join or leave** the system freely.
- No master-slave relationship exists among the peers.
- No central coordination or central database is needed.
- **No** peer machine has a **global view** of the entire P2P system.
- The system is self-organizing with distributed control.



Peer-to-Peer Network – (Contibued):

- Initially, the peers are totally unrelated. Each peer machine joins or leaves the P2P network voluntarily.
- Only the participating peers form the physical network at any time.
- Unlike the cluster or grid, a P2P network **does not use a dedicated interconnection network**. The physical network is simply an ad hoc network formed at various Internet domains randomly using the TCP/IP and NAI protocols.
- The physical network **varies in size and topology** dynamically due to the free membership in the P2P network.
- When a new peer joins the system, its peer ID is added as a node in the overlay network.
- When an existing peer leaves the system, its peer ID is removed from the overlay network automatically. Therefore, it is the P2P overlay network that characterizes the logical connectivity among the peers.

Peer-to-Peer Network – (Contibued):

P2P Computing Challenges:

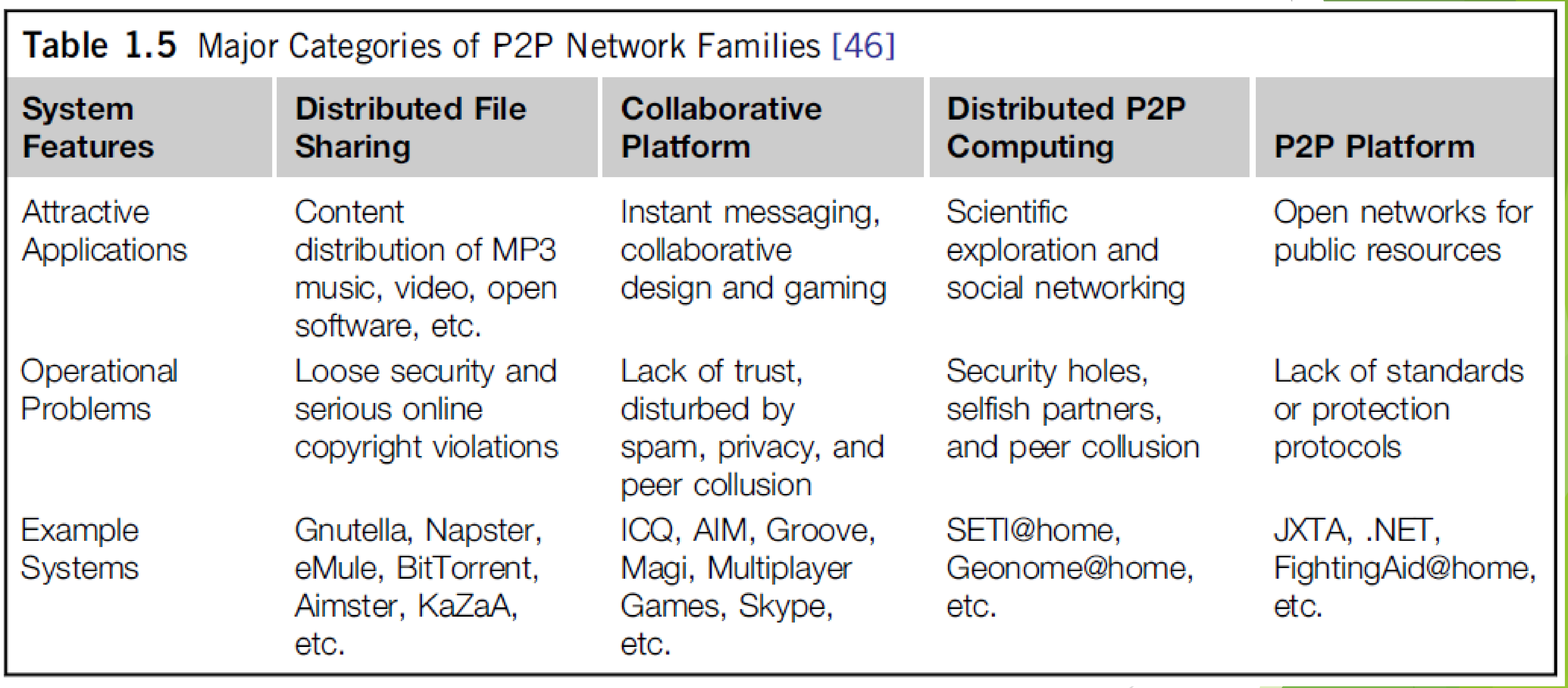


Table 1.5 Major Categories of P2P Network Families [46]

System Features	Distributed File Sharing	Collaborative Platform	Distributed P2P Computing	P2P Platform
Attractive Applications	Content distribution of MP3 music, video, open software, etc.	Instant messaging, collaborative design and gaming	Scientific exploration and social networking	Open networks for public resources
Operational Problems	Loose security and serious online copyright violations	Lack of trust, disturbed by spam, privacy, and peer collusion	Security holes, selfish partners, and peer collusion	Lack of standards or protection protocols
Example Systems	Gnutella, Napster, eMule, BitTorrent, Aimster, KaZaA, etc.	ICQ, AIM, Groove, Magi, Multiplayer Games, Skype, etc.	SETI@home, Geonome@home, etc.	JXTA, .NET, FightingAid@home, etc.

Cloud Computing over the Internet:

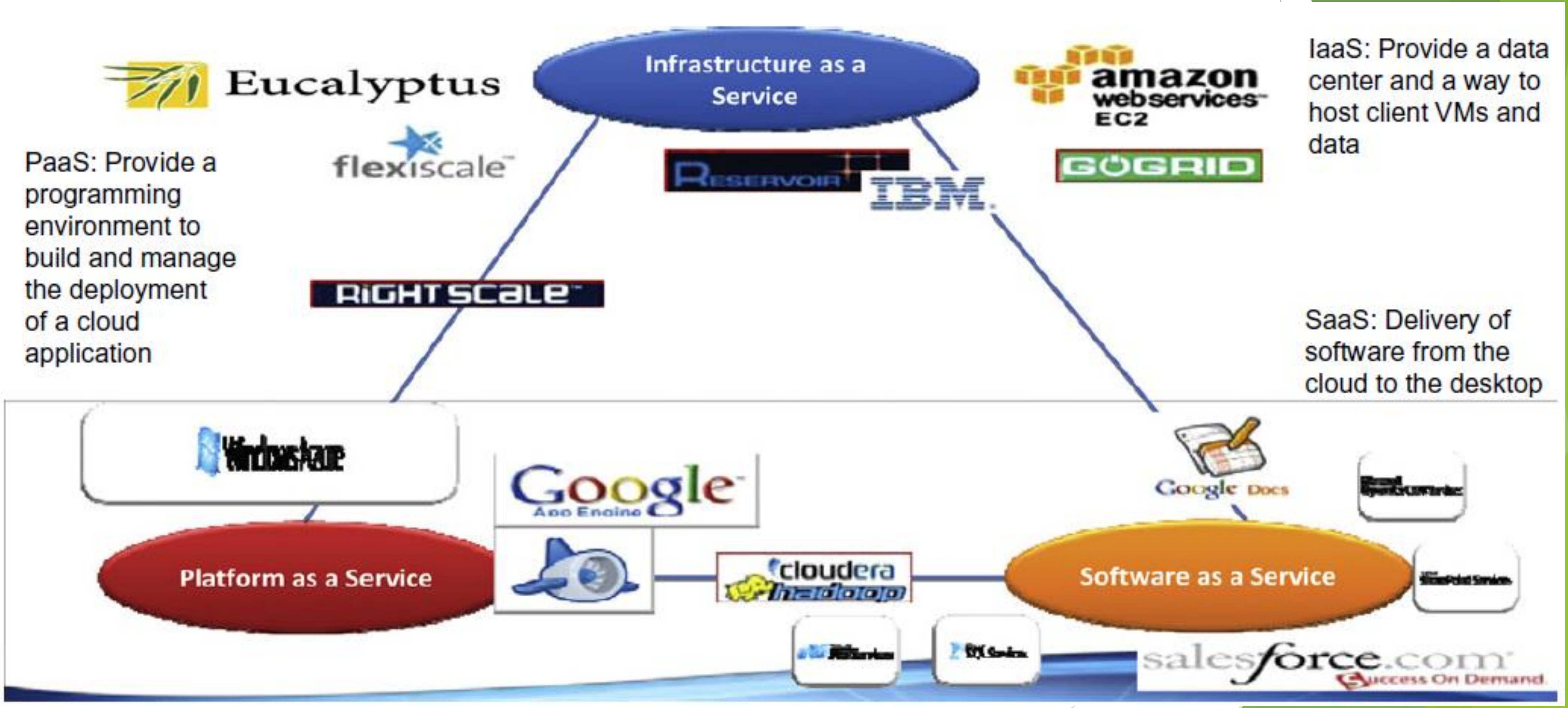
- Cloud computing has been defined differently by many users and designers.
- IBM, a major player in cloud computing, has defined it as follows:

  "A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications."

- Based on this definition, a cloud allows workloads to be deployed and scaled out quickly through rapid provisioning of virtual or physical machines.
- The cloud supports redundant, self-recovering, highly scalable programming models that allow workloads to recover from many unavoidable hardware/software failures. Finally, the cloud system should be able to monitor resource use in real time to enable rebalancing of allocations when needed.
- Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and data sets dynamically.
- The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers.
- Cloud computing leverages its low cost and simplicity to benefit both users and providers.

Cloud Computing over the Internet (Continued):

Three cloud service models: Infrastructure as a Service (**IaaS**), Platform as a Service (**PaaS**), Software as a Service (**SaaS**)


Infrastructure as a Service
Eucalyptus
amazon webservices EC2
IaaS: Provide a data center and a way to host client VMs and data
PaaS: Provide a programming environment to build and manage the deployment of a cloud application
flexiscale
RESERVOIR
IBM
GOGRID
RIGHTSCALE
SaaS: Delivery of software from the cloud to the desktop
Windows Azure
Google App Engine
Google Docs
Platform as a Service
cloudera hadoop
Software as a Service
salesforce.com Success On Demand