Parallel & Distributed Computing (WQD7008)

Week 4

Computer Cluster for Scale Computing

2019/2020 Semester 1

Dr. Hamid Tahaei

IBM
Blue Gene/P

Clustering

- A collection of interconnected stand-alone computers which can work together **collectively** and **cooperatively** as a single integrated computing resource pool.
- It explores massive parallelism at the job level, achieving high availability (HA) through standalone operations.
- Benefits of computer clusters and massively parallel processors (MPPs):
  - Scalable performance
  - HA
  - Fault tolerance
  - Modular growth
  - Use of commodity components.
- Of the Top 500 supercomputers reported in 2010, 85 percent were computer clusters or MPPs built with homogeneous nodes.
- Computer clusters have laid the foundation for today's supercomputers, computational grids, and Internet clouds built over data centers.
- A majority of the Top 500 supercomputers are used for HPC applications in science and engineering. Meanwhile, the use of high throughput computing (HTC) clusters of servers is growing rapidly in business and web services applications.

Classification of Clusters in terms of attributes:

- **Scalability**
  - Modular growth
- **Packaging**
  - Compact
  - Slack
- **Control**
  - Centralized
  - Decentralized
- **Homogeneity**
  - Homogeneous
  - Heterogeneous
- **Security**
  - Exposed
  - Enclosed
- **Dedicated versus Enterprise Clusters**
  - Dedicated
  - Enterprise

Objectives of Computer Clusters

Scalability

- modular growth.
- a nontrivial task.
- Can be limited by the number of factors.

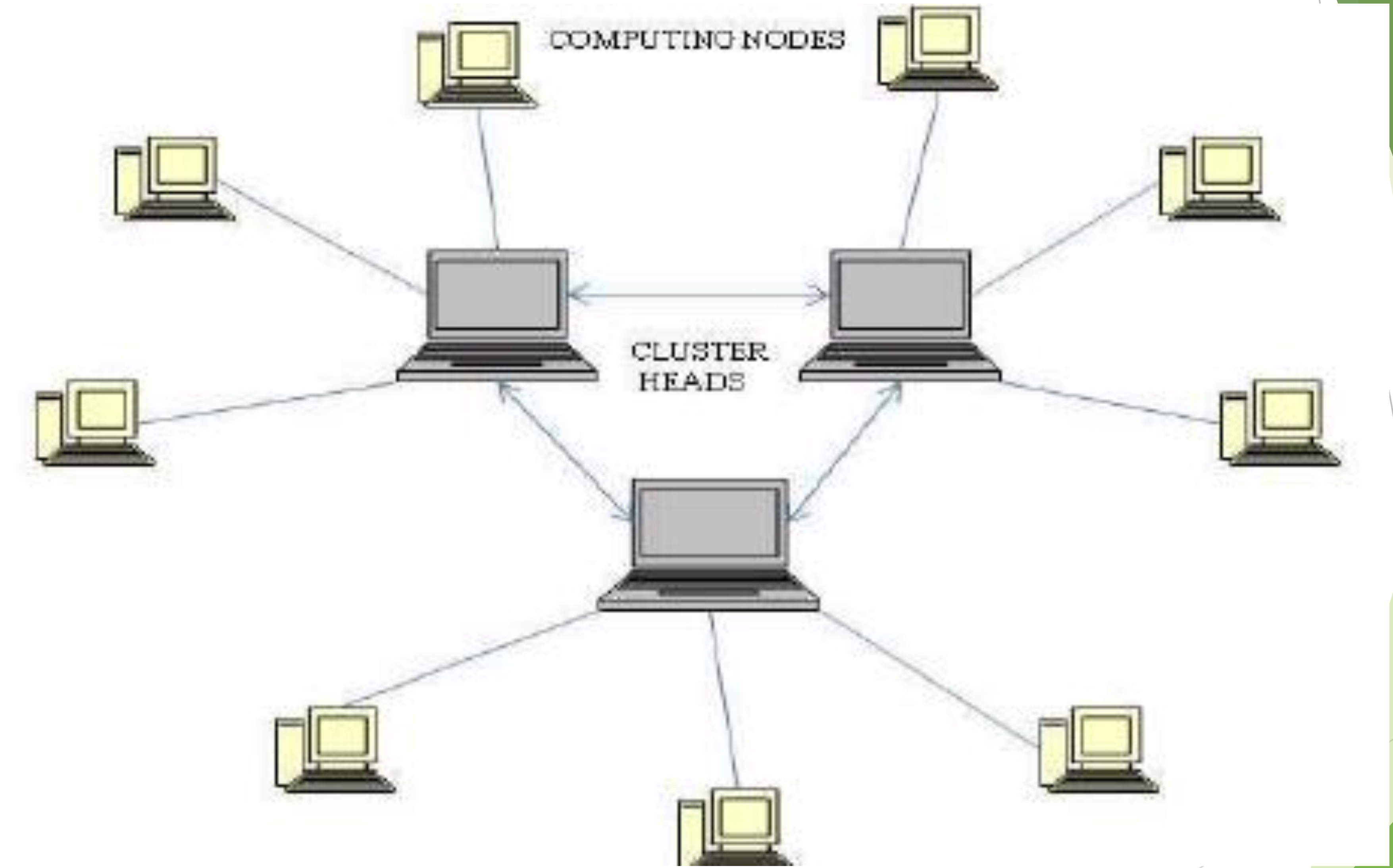
Objectives of Computer Clusters

Scalability

- Clustering of computers is based on the concept of modular growth.
- a nontrivial task.
- Can be limited by: multicore chip technology, cluster topology, packaging method, power consumption, cooling scheme applied, memory wall, disk I/O bottlenecks, and latency tolerance, among others.

Packaging

- Compact
  - The nodes are closely packaged in one or more racks sitting in a room, and the nodes are not attached to peripherals (monitors, keyboards, mice, etc.).
- Slack fashion
  - The nodes are attached to their usual peripherals (i.e., they are complete SMPs, workstations, and PCs), and they may be located in different rooms, different buildings, or even remote regions.


COMPUTING NODES
CLUSTER HEADS

Objectives of Computer Clusters (Continued)

Control

- Centralized.
  - All the nodes are owned, controlled, managed, and administered by a central operator.
  - A compact cluster normally has centralized control, while a slack cluster can be controlled either way.
- Decentralized.
  - Nodes have individual owners.
  - This lack of a single point of control makes system administration of such a cluster very difficult.
  - It also calls for special techniques for process scheduling, workload migration, checkpointing, accounting, and other similar tasks.

Homogeneity:

- A homogeneous cluster uses nodes from the same platform, that is, the same processor architecture and the same operating system; often, the nodes are from the same vendors.
  - A binary process image can migrate to another node and continue execution.
- Heterogeneous cluster uses nodes of different platforms. Interoperability is an important issue in heterogeneous clusters.
  - Process migration is often needed for load balancing or availability.
  - A binary process image will not be executable when the process migrates to a node of a different platform.

Objectives of Computer Clusters (Continued)

Security

- Exposed.
  - The communication paths among the nodes are exposed to the outside world. An outside machine can access the communication paths, and thus individual nodes, using standard protocols (e.g., TCP/IP).
  - Easy to implement, Disadvantage:
    - Being exposed, intracluster communication is not secure, unless the communication subsystem performs additional work to ensure privacy and security.
    - Outside communications may disrupt intracluster communications in an unpredictable fashion.
    - Standard communication protocols tend to have high overhead.
- Enclosed.
  - Intracluster communication is shielded from the outside world, which alleviates the aforementioned problems.
  - A disadvantage is that there is currently no standard for efficient, enclosed intracluster communication.
  - Most commercial or academic clusters realize fast communications through one-of-a-kind protocols.

Objectives of Computer Clusters (Continued)

Dedicated versus Enterprise Clusters

Dedicated.

- Is Typically installed in a deskside rack in a central computer room.
- It is homogeneously configured with the same type of computer nodes and managed by a single administrator group like a frontend host.
- are used as substitutes for traditional mainframes or supercomputers.
- is installed, used, and administered as a single machine.
- Users can log into the cluster to execute both interactive and batch jobs.
- Offers much enhanced throughput, as well as reduced response time.

Example of Jobs:

- Submit the batch job.

```
sbatch simple.slurm
qsub simple.pbs
```

- Interactive job.

```
srun --x11 -n 1 -c 2 --time=1:00:00 --pty /bin/bash
```

This will launch two cpus per task (-c 2) with a single task (-n 1) for 1 hour (--time=1:00:00), with graphical windows (--x11) ready. If you want to request, let's say, 100 hrs, you can do it as --time=4-04:00:00 in the the format: "days-hours:minutes:seconds“

```
srun -p gpufermi --nodelist=gpu009t --pty bash
```

t is requesting gpu009t. It is useful when we want to use a specific node only. You might have to wait until the node is free to use if someone else is running a job in that node.

Dedicated versus Enterprise Clusters (Continued)

Enterprise

- is mainly used to utilize idle resources in the nodes.
- Each node is usually a full-fledged SMP, workstation, or PC, with all the necessary peripherals attached.
- The nodes are typically geographically distributed, and are not necessarily in the same room or even in the same building.
- The nodes are individually owned by multiple owners.
- The cluster administrator has only limited control over the nodes, as a node can be turned off at any time by its owner.
- The owner’s “local” jobs have higher priority than enterprise jobs.
- The cluster is often configured with heterogeneous computer nodes.
- The nodes are often connected through a low-cost Ethernet network.
- Most data centers are structured with clusters of low-cost servers.
- Virtual clusters play a crucial role in upgrading data centers.

Fundamental Cluster Design Issues

- Scalable Performance
  - Cluster nodes, memory capacity, I/O bandwidth, etc.
  - leads to a proportional increase in performance.
- Single-System Image (SSI)
  - A set of workstations connected by an Ethernet network is not necessarily a cluster.
  - A cluster is a single system.
- Availability Support
  - Clusters can provide cost-effective HA capability with lots of redundancy in processors, memory, disks, I/O devices, networks, and operating system images.
- Cluster Job Management
  - Clusters try to achieve high system utilization from traditional workstations or PC nodes that are normally not highly utilized. Job management software is required to provide batching, load balancing, parallel processing, and other functionality.

Fundamental Cluster Design Issues (Continued)

- Internode Communication
  - Because of their higher node complexity, cluster nodes cannot be packaged as compactly as MPP nodes. The internode physical wire lengths are longer in a cluster than in an MPP. This is true even for centralized clusters. A long wire implies greater interconnect network latency. But more importantly, **longer wires have more problems in terms of reliability, clock skew, and cross talking**. These problems call for reliable and secure communication protocols, which increase overhead. Clusters often use commodity networks (e.g., Ethernet) with standard protocols such as TCP/IP.
- Fault Tolerance and Recovery
  - Clusters of machines can be designed to eliminate all single points of failure. Through **redundancy**, a cluster can tolerate faulty conditions up to a certain extent. **Heartbeat mechanisms** can be installed to monitor the running condition of all nodes. In case of a node failure, critical jobs running on the failing nodes can be saved by failing over to the surviving node machines. **Rollback recovery** schemes restore the computing results through **periodic checkpointing**.

Cluster Family Classification

Three classes of cluster based on the application demand:

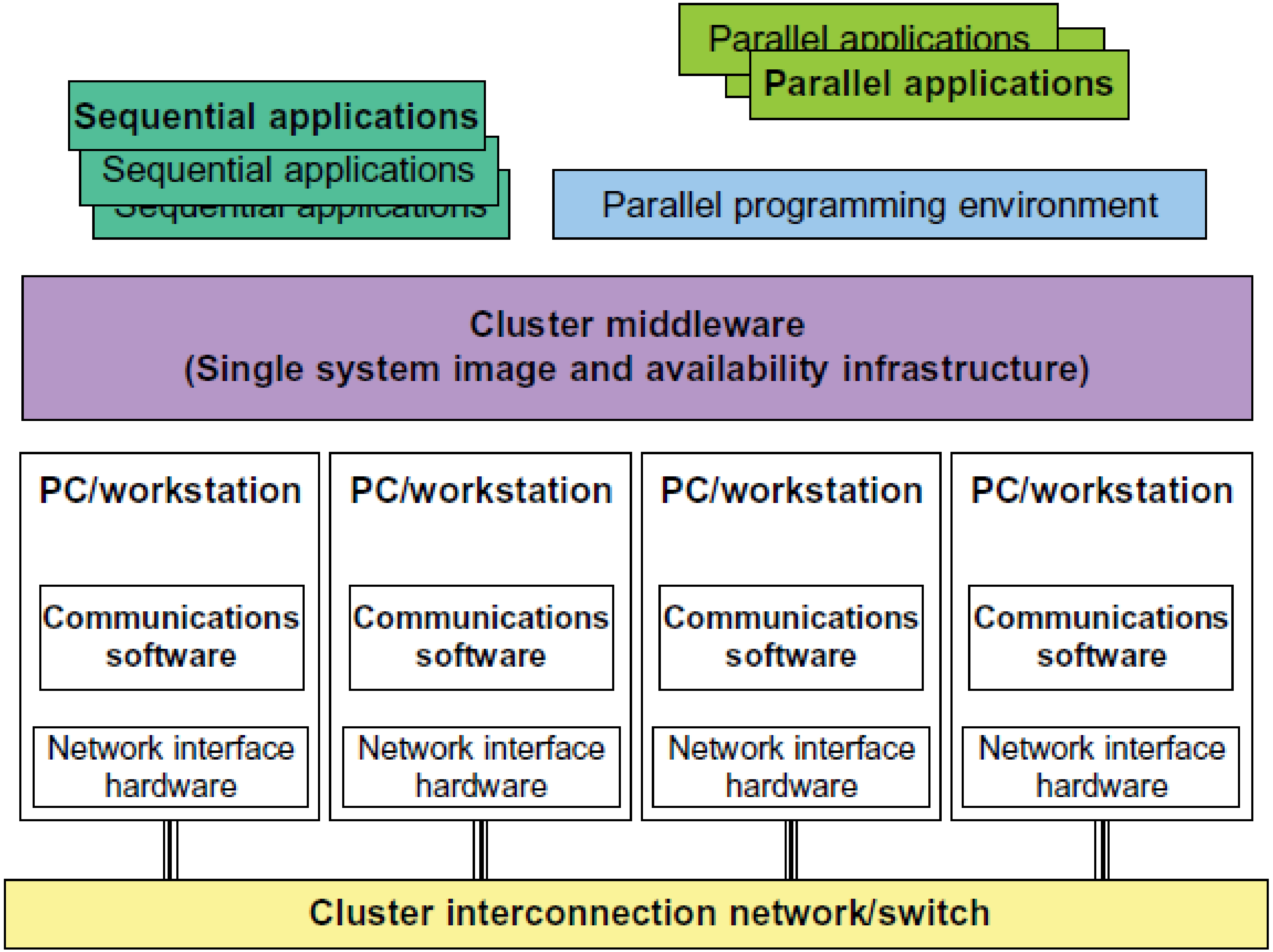
- Compute clusters
  - These are clusters designed mainly for collective computation over a single large job. A good example is a cluster dedicated to numerical simulation of weather conditions. The compute clusters do not handle many I/O operations, such as database services. When a single compute job requires frequent communication among the cluster nodes, the cluster must share a dedicated network, and thus the nodes are mostly homogeneous and tightly coupled. This type of clusters is also known as a **Beowulf cluster**. When the nodes require internode communication over a small number of heavy-duty nodes, they are essentially known as a computational grid. Tightly coupled compute clusters are designed for supercomputing applications. Compute clusters apply middleware such as a message-passing interface (MPI) or Parallel Virtual Machine (PVM) to port programs to a wide variety of clusters.
- High-Availability clusters (Active Passive)
  - HA (high-availability) clusters are designed to be fault-tolerant and achieve HA of services. HA clusters operate with many redundant nodes to sustain faults or failures. The simplest HA cluster has only two nodes that can fail over to each other. Of course, high redundancy provides higher availability. HA clusters should be designed to avoid all single points of failure. Many commercial HA clusters are available for various operating systems.

Cluster Family Classification (Continued)

- Load-balancing clusters (Active - Active)
  - These clusters shoot for higher resource utilization through load balancing among all participating nodes in the cluster. All nodes share the workload or function as a single virtual machine (VM). Requests initiated from the user are distributed to all node computers to form a cluster. This results in a balanced workload among different machines, and thus higher resource utilization or higher performance. Middleware is needed to achieve dynamic load balancing by job or process migration among all the cluster nodes.

A Basic Cluster Architecture

- The processing nodes are **commodity workstations**, PCs, or servers.
  - Should be easy to replace or upgrade with new generations of hardware.
- The node **operating systems** should be designed for multiuser, multitasking, and multithreaded applications.
- The nodes are interconnected by one or more fast **commodity networks**.
- These networks use standard communication protocols and operate at a speed that should be two orders of magnitude **faster than that of the current TCP/IP speed over Ethernet**.
- The network interface card is connected to the node's standard I/O bus (e.g., PCI).
- When the processor or the operating system is changed, only the **driver software** needs to change.
- A **middleware** is deployed to glue together all node platforms at the user space.
- An **SSI** layer provides a single entry point, a single file hierarchy, a single point of control, and a single job management system.


Parallel applications
Parallel applications
Sequential applications
Sequential applications
Sequential applications
Parallel programming environment
Cluster middleware
(Single system image and availability infrastructure)
PC/workstation
Communications software
Network interface hardware
PC/workstation
Communications software
Network interface hardware
PC/workstation
Communications software
Network interface hardware
PC/workstation
Communications software
Network interface hardware
Cluster interconnection network/switch

A Basic Cluster Architecture (Continued)

An idealized cluster is supported by three subsystems.

- conventional databases and online transaction processing (OLTP) monitors offer users a desktop environment in which to use the cluster.
- In addition to running sequential user programs, the cluster supports parallel programming based on standard languages and communication libraries using PVM, MPI, or OpenMP. The programming environment also includes tools for debugging, profiling, monitoring, and so forth.
- A user interface subsystem is needed to combine the advantages of the web interface and the Windows GUI. It should also provide user-friendly links to various programming environments, job management tools, hypertext, and search support so that users can easily get help in programming the computer cluster.

Resource Sharing in Clusters

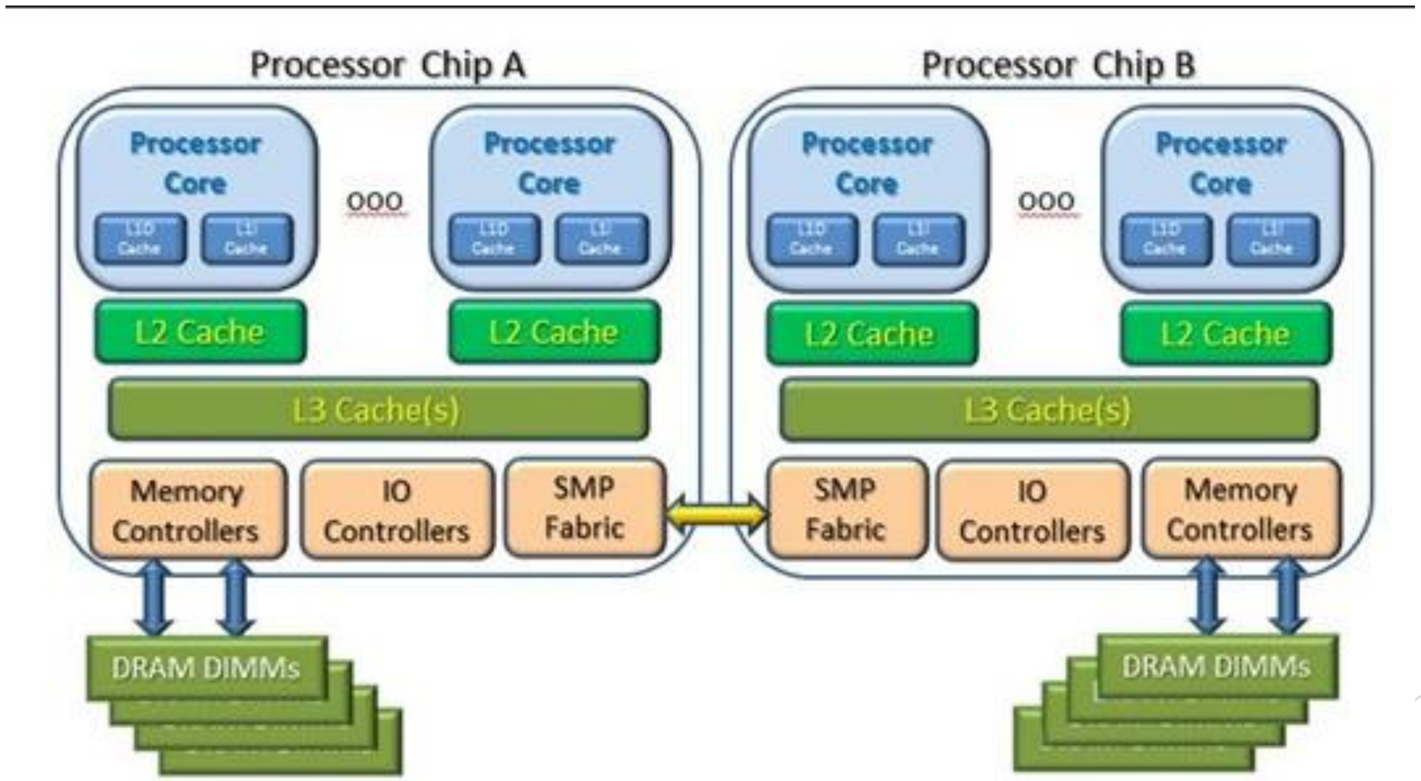
Some HA clusters use hardware redundancy for scalable performance.

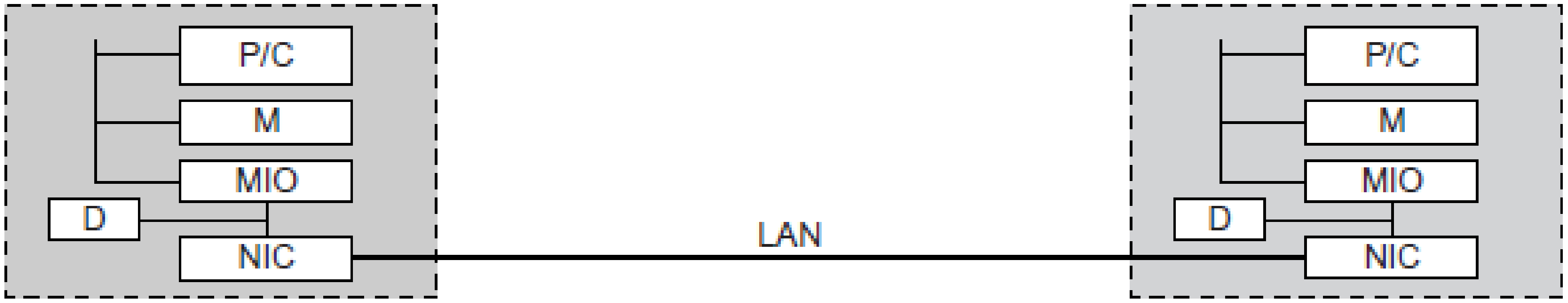
The nodes of a cluster can be connected in one of three ways.

- **The shared-nothing architecture:** is used in most clusters, where the nodes are connected through the I/O bus.
  - connects two or more autonomous computers via a LAN such as Ethernet.
- **The shared-disk architecture:** is in favor of small-scale availability clusters in business applications. When one node fails, the other node takes over.
  - most business clusters desire so that they can enable recovery support in case of node failure.
  - The shared disk can hold checkpoint files or critical system images to enhance cluster availability.
  - Without shared disks, check pointing, rollback recovery, failover, and failback are not possible in a cluster.

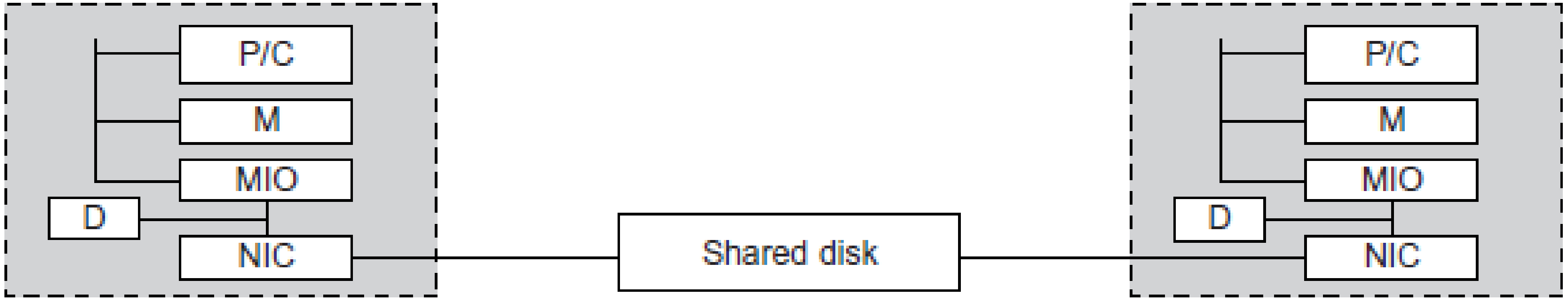
Resource Sharing in Clusters (Continued)

- **shared-memory cluster:** The nodes could be connected by a scalable coherence interface (SCI) ring, which is connected to the memory bus of each node through an NIC module. In the other two architectures, the interconnect is attached to the I/O bus. The memory bus operates at a higher frequency than the I/O bus.

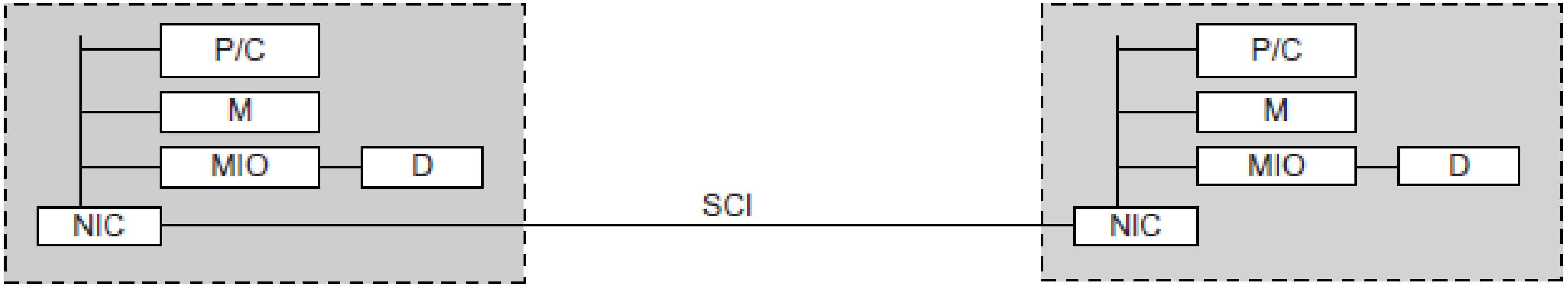




(a) Shared nothing



(b) Shared disk

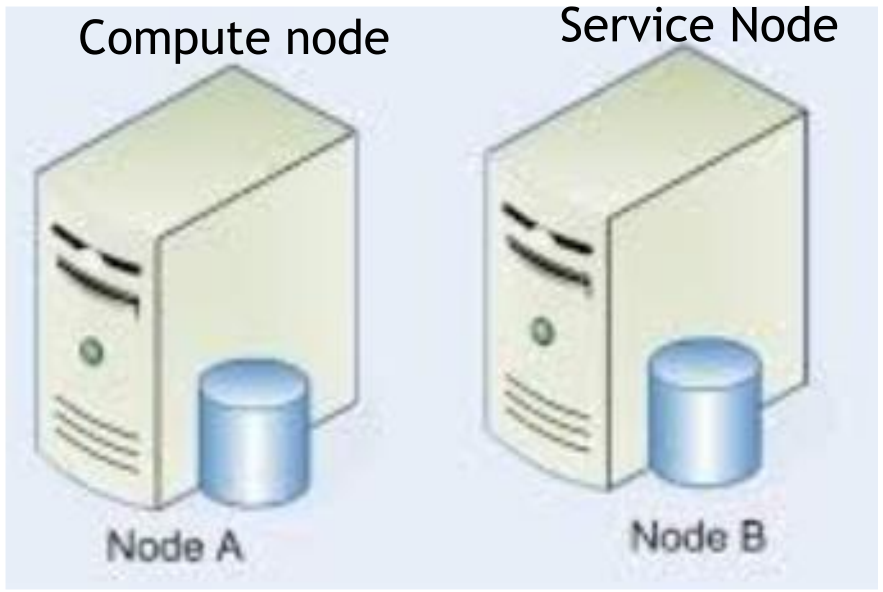


(c) Shared memory

Node Architectures and MPP Packaging

Providing Services:

- largescale searching or parallel floating-point computations
- High computation
- Homogeneous
- Hybrid



Providing Services:

- File access
- Handle I/O
- DNS
- Login
- Create
- Update
- Remove
- Scheduling
- …

Node Architectures and MPP Packaging

Cluster nodes for MPP are classified into two categories:

- compute nodes
  - appear in larger quantities mainly used for largescale searching or parallel floating-point computations. two example compute node architectures: homogeneous design and hybrid node design.
  - A homogeneous node design makes it easier to program and maintain the system.
    - For MPP clusters, the compute nodes dominate in system cost, because there may be 1,000 times more compute nodes than service nodes in a single large clustered system.
- service nodes
  - could be built with different processors mainly used to handle I/O, file access, and system monitoring.

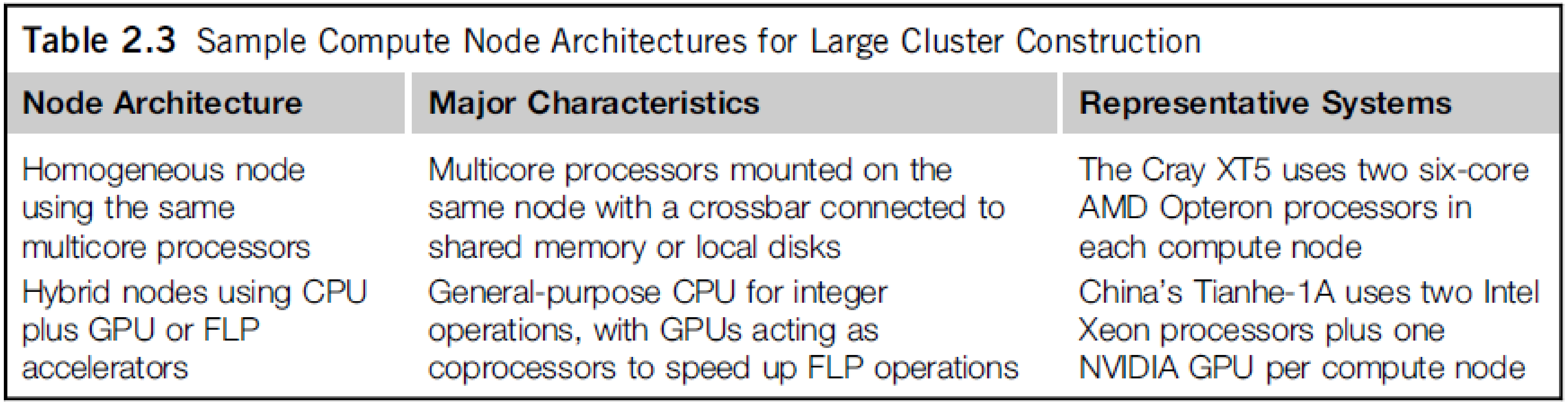


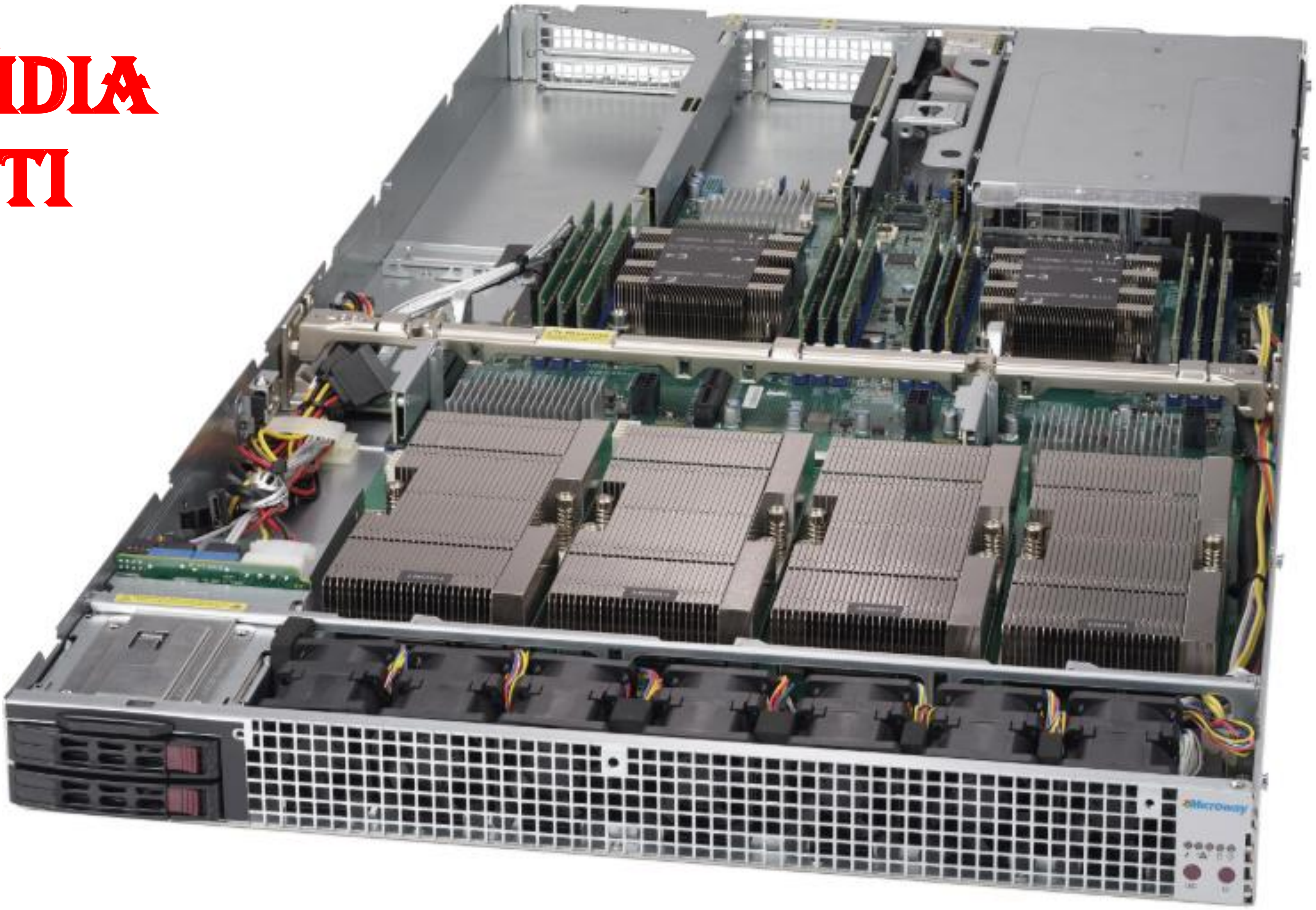
Table 2.3 Sample Compute Node Architectures for Large Cluster Construction

Node Architecture	Major Characteristics	Representative Systems
Homogeneous node using the same multicore processors	Multicore processors mounted on the same node with a crossbar connected to shared memory or local disks	The Cray XT5 uses two six-core AMD Opteron processors in each compute node
Hybrid nodes using CPU plus GPU or FLP accelerators	General-purpose CPU for integer operations, with GPUs acting as coprocessors to speed up FLP operations	China's Tianhe-1A uses two Intel Xeon processors plus one NVIDIA GPU per compute node

GPU Clusters for Massive Parallelism

NVIDIA
ATI

CUDA
OPENCL

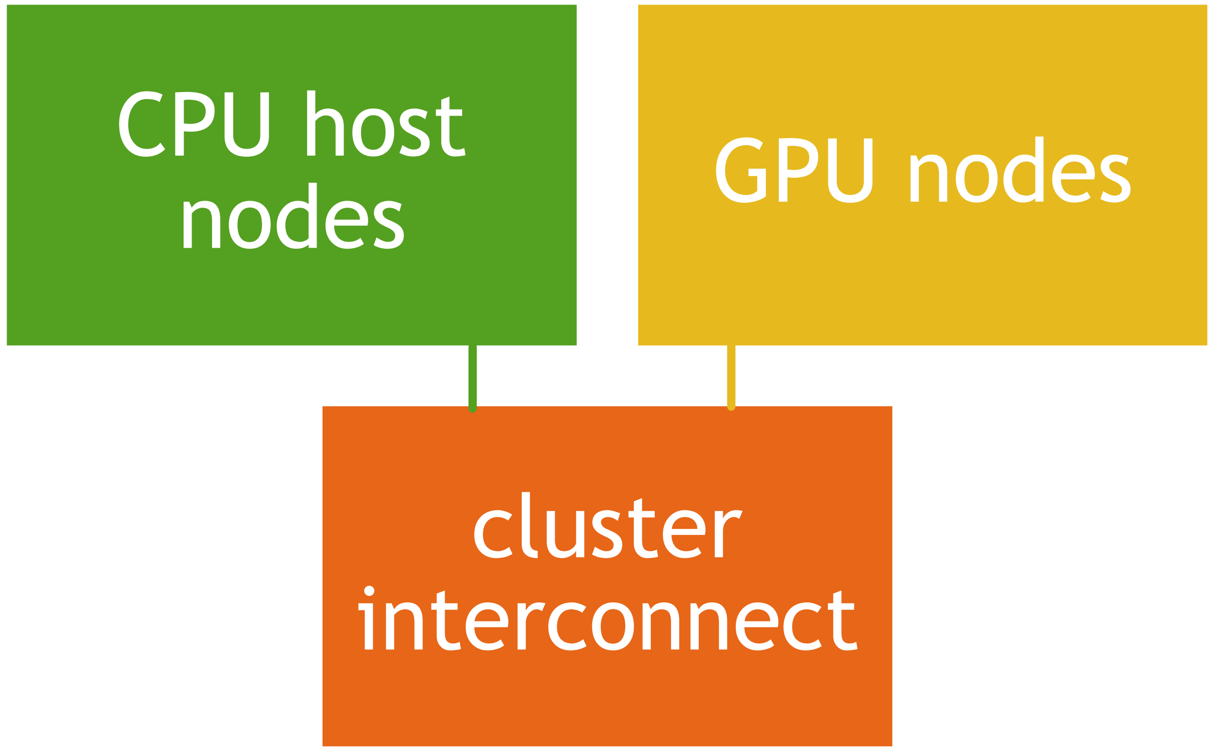


GPU Clusters for Massive Parallelism

- Commodity GPUs are becoming high-performance accelerators for data-parallel computing.
- GPU chips contain hundreds of processor cores per chip.
- Recent HPC-optimized GPUs contain up to 4 GB of on-board memory, and are capable of sustaining memory bandwidths exceeding 100 GB/second.
- GPU clusters are built with a large number of GPU chips.
- Most GPU clusters are structured with homogeneous GPUs of the same hardware class, make, and model.
- The software used in a GPU cluster includes the OS, GPU drivers, and clustering API such as an MPI.
- GPU clusters already are more cost-effective than traditional CPU clusters.
- GPU clusters result in not only a quantum jump in speed performance, but also significantly reduced space, power, and cooling demands.
- CUDA (Compute Unified Device Architecture) offers a parallel computing architecture developed by NVIDIA. CUDA is the computing engine in NVIDIA GPUs.
  - Has been used to accelerate non-graphical applications in computational biology, cryptography, and other fields by an order of magnitude or more.

GPU Clusters for Massive Parallelism (Continued)

A GPU cluster is often built as a heterogeneous system consisting of three major components:



GPU Clusters for Massive Parallelism (Continued)

- **CPU host nodes**
  - The host node controls program execution.
- **GPU nodes**
  - The GPU nodes are formed with general-purpose GPUs, known as GPGPUs, to carry out numerical calculations.
- **Cluster interconnect between them.**
  - The cluster interconnect handles inter-node communications.

- To guarantee the performance, multiple GPUs must be fully supplied with data streams over high-bandwidth network and memory. Host memory should be optimized to match with the on-chip cache bandwidths on the GPUs.

DESIGN PRINCIPLES OF COMPUTER CLUSTERS

Single-System Image Features.

- **Single system:** The entire cluster is viewed by users as one system that has multiple processors. The user could say, "Execute my application using five processors." This is different from a distributed system.
- **Single control:** Logically, an end user or system user utilizes services from one place with a single interface. For instance, a user submits batch jobs to one set of queues; a system administrator configures all the hardware and software components of the cluster from one control point.
- **Symmetry:** A user can use a cluster service from any node. In other words, all cluster services and functionalities are symmetric to all nodes and all users, except those protected by access rights.
- **Location-transparent:** The user is not aware of the where abouts of the physical device that eventually provides a service. For instance, the user can use a tape drive attached to any cluster node as though it were physically attached to the local node.

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)

The illusion of an SSI can be obtained at several layers.

- **Application software layer:** Two examples are parallel web servers and various parallel databases. The user sees an SSI through the application and is not even aware that he is using a cluster. This approach demands the modification of workstation or SMP applications for clusters.
- **Hardware or kernel layer:** Ideally, SSI should be provided by the operating system or by the hardware. Unfortunately, this is not a reality yet. Furthermore, it is extremely difficult to provide an SSI over heterogeneous clusters. With most hardware architectures and operating systems being proprietary, only the manufacturer can use this approach.
- **Middleware layer:** The most viable approach is to construct an SSI layer just above the OS kernel. This approach is promising because it is platform-independent and does not require application modification. Many cluster job management systems have already adopted this approach.

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)

Home node p

Local node p

Compute node

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)

From the view point of a process p cluster nodes can be classified into three types.

- **Home node of a process P:** is the node where P resided when it was created.
- **local node of a process p:** is the node where P currently resides.
- **All other nodes:** are remote nodes to P. Cluster nodes can be configured to suit different needs.

A host node: serves user logins through Telnet, rlogin, or even FTP and HTTP.

A compute node: is one that performs computational jobs.

An I/O node: is one that serves file I/O requests. If a cluster has large shared disks and tape units, they are normally physically attached to I/O nodes.

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)

Features:

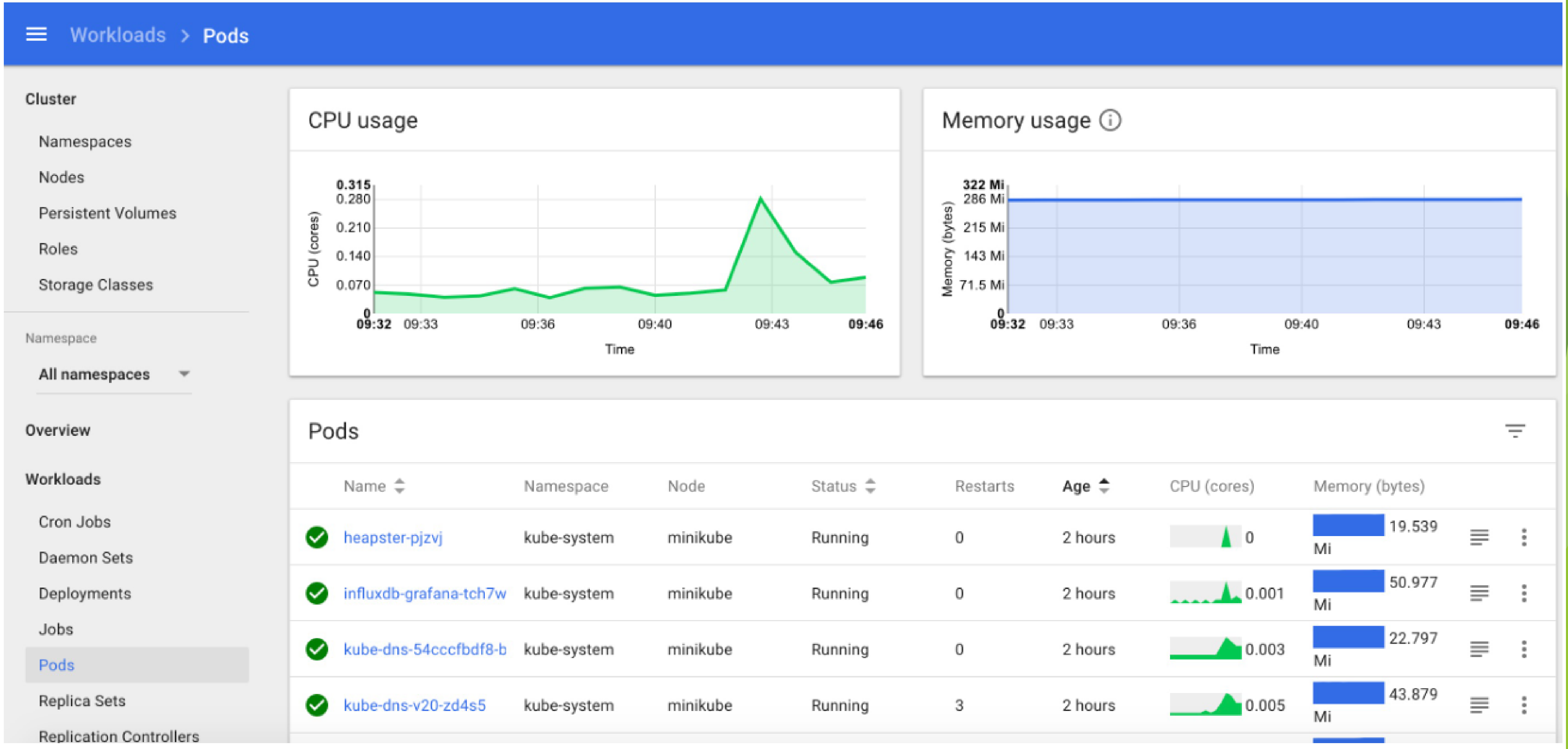
- **Single Entry Point:** enables users to log in (e.g., through Telnet, rlogin, or HTTP) to a cluster as one virtual host, although the cluster may have multiple physical host nodes to serve the login sessions.
  - **Issues:** Home directory, Authentication, Multiple connections, Host failure.
- **Single File Hierarchy:** the huge file system image that transparently integrates local and global disks and other file devices (e.g., tapes).
- **Visibility of Files:** means a process can use traditional UNIX system or library calls such as fopen, fread, and fwrite to access files.
- **Support of Single-File Hierarchy**
  - Single system: There is just one file hierarchy from the user's viewpoint.
  - Symmetry: A user can access the global storage (e.g., /scratch) using a cluster service from any node. In other words, all file services and functionalities are symmetric to all nodes and all users, except those protected by access rights.
  - Location-transparent: The user is not aware of the whereabouts of the physical device that eventually provides a service.

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)

Features (Continued):

- **Single I/O, Networking, and Memory Space:**
  - Single Networking
  - Single Point of Control
  - Single Memory Space
  - Single I/O Address Space
- Single job management system: All cluster jobs can be submitted from any node to a single job management system.
- Single user interface: The users use the cluster through a single graphical interface.
- Single process space: All user processes created on various nodes form a single process space and share a uniform process identification scheme.
- Middleware support for SSI clustering
- Management level
- Programming level: This level provides single file hierarchy and distributed shared memory.
- Implementation level: This level supports a single process space, checkpointing, process migration, and a single I/O space.

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)



Pods

	Name	Namespace	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
✓	heapster-pjzvj	kube-system	minikube	Running	0	2 hours	0	19.539 Mi
✓	influxdb-grafana-tch7w	kube-system	minikube	Running	0	2 hours	0.001	50.977 Mi
✓	kube-dns-54cccfbdf8-b	kube-system	minikube	Running	0	2 hours	0.003	22.797 Mi
✓	kube-dns-v20-zd4s5	kube-system	minikube	Running	3	2 hours	0.005	43.879 Mi

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)

High Availability through Redundancy:

- **Reliability:** measures how long a system can operate without a breakdown. Refer to as the mean time to failure (MTTF)
- **Availability:** indicates the percentage of time that a system is available to the user, that is, the percentage of system uptime.
- **Serviceability:** refers to how easy it is to service the system, including hardware and software maintenance, repair, upgrades, and so on. Refer to as the mean time to repair (MTTR)

- Availability and Failure Rate:

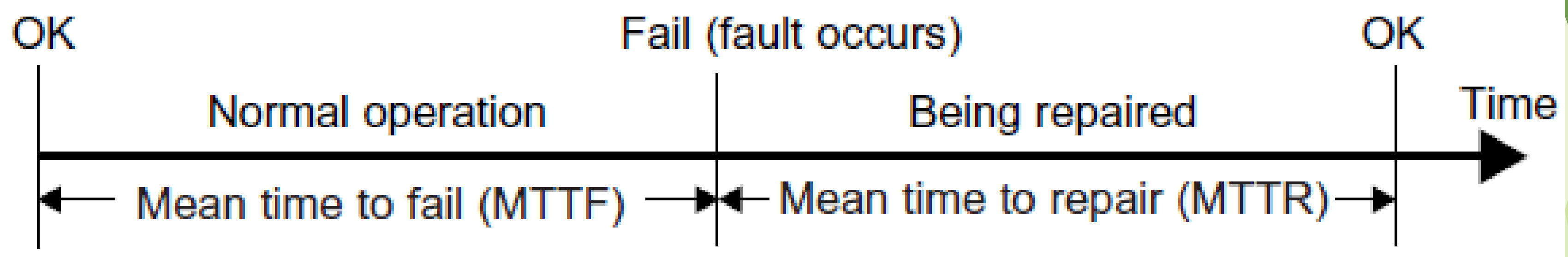
$$\mathit{Availability} = \mathrm{MTTF}/(\mathrm{MTTF} + \mathrm{MTTR})$$

- Planned versus Unplanned Failure
  - Unplanned failures: The system breaks, due to an operating system crash, hardware failure, and etc.
- Planned shutdowns: The system is not broken, but is periodically taken off normal operation for upgrades, reconfiguration, and maintenance.

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)

High Availability through Redundancy (Continued):

- Transient versus Permanent Failures



- Partial versus Total Failures
- Redundancy Techniques
- Isolated Redundancy
- N-Version Programming to Enhance Software Reliability

DESIGN PRINCIPLES OF COMPUTER CLUSTERS (Continued)

Fault-Tolerant Cluster Configurations

- Hot standby
- Active takeover
- Fault-tolerant

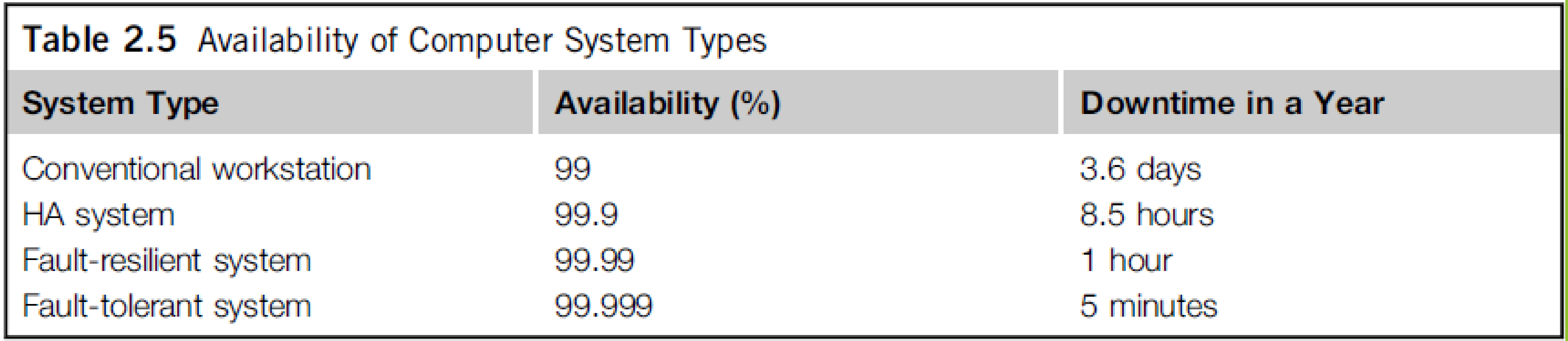


Table 2.5 Availability of Computer System Types

System Type	Availability (%)	Downtime in a Year
Conventional workstation	99	3.6 days
HA system	99.9	8.5 hours
Fault-resilient system	99.99	1 hour
Fault-tolerant system	99.999	5 minutes

CLUSTER JOB AND RESOURCE MANAGEMENT

Three schemes are used to share cluster nodes:

- **Dedicated mode:** only one job runs in the cluster at a time, and at most, one process of the job is assigned to a node at a time. The single job runs until completion before it releases the cluster to run other jobs. Note that even in the dedicated mode, some nodes may be reserved for system use and not be open to the user job. Other than that, all cluster resources are devoted to run a single job. This may lead to poor system utilization.
- **Static scheme:** fixes the number of nodes for a single job for its entire period. Static scheme may underutilize the cluster resource. It cannot handle the situation when the needed nodes become unavailable, such as when the workstation owner shuts down the machine.
- **Dynamic:** resource allows a job to acquire or release nodes during execution. However, it is much more difficult to implement, requiring cooperation between a running job and the Java Message Service (JMS). The jobs make asynchronous requests to the JMS to add/delete resources. The JMS needs to notify the job when resources become available. The synchrony means that a job should not be delayed (blocked) by the request/notification. Cooperation between jobs and the JMS requires modification of the programming languages/libraries. A primitive mechanism for such cooperation exists in PVM and MPI.

CLUSTER JOB AND RESOURCE MANAGEMENT (Continued)

Cluster Job Scheduling Methods (Continued)

- Space Sharing: multiple jobs can run on disjointed partitions (groups) of nodes simultaneously. At most, one process is assigned to a node at a time. Although a partition of nodes is dedicated to a job, the interconnect and the I/O subsystem may be shared by all jobs. Space sharing must solve the tiling problem and the large-job problem.
- Time Sharing: multiple user processes are assigned to the same node.
  - Independent scheduling: Using operating system of each node. However, performance of parallel jobs could be significantly degraded.
  - Gang scheduling: All processes are scheduled to run at the same time. However, Gang-scheduling skew happens.
- Competition with foreign (local) jobs: higher priority for local jobs and lower priority for cluster jobs. he cluster job can either stay in the workstation node or migrate to another idle node.

CLUSTER JOB AND RESOURCE MANAGEMENT (Continued)

Cluster Job Management Systems: also known as workload management, load sharing, or load management. A Job Management System (JMS) should have three parts:

- **A user server:** lets the user submit jobs to one or more queues, specify resource requirements for each job, delete a job from a queue, and inquire about the status of a job or a queue.
- **A job scheduler:** performs job scheduling and queuing according to job types, resource requirements, resource availability, and scheduling policies.
- **A resource manager:** allocates and monitors resources, enforces scheduling policies, and collects accounting information.

CLUSTER JOB AND RESOURCE MANAGEMENT (Continued)

Cluster Job Management Systems (JMS).

JMS Administration:

- should be able to dynamically reconfigure the cluster with minimal impact on the running jobs. The administrator’s prologue and epilogue scripts should be able to run before and after each job for security checking, accounting, and cleanup. Users should be able to cleanly kill their own jobs. The administrator or the JMS should be able to cleanly suspend or kill any job. Clean means that when a job is suspended or killed, all its processes must be included. Otherwise, some “orphan” processes are left in the system, which wastes cluster resources and may eventually render the system unusable.

CLUSTER JOB AND RESOURCE MANAGEMENT (Continued)

Cluster Job Management Systems (JMS) (Continued)

Cluster Job Types:

- Serial jobs: run on a single node.
- Parallel jobs: use multiple nodes.
- Interactive jobs: are those that require fast turnaround time, and their input/output is directed to a terminal. These jobs do not need large resources, and users expect them to execute immediately, not to wait in a queue.
- Batch jobs: normally need more resources, such as large memory space and long CPU time. But they do not need immediate responses. They are submitted to a job queue to be scheduled to run when the resource becomes available (e.g., during off hours).

CLUSTER JOB AND RESOURCE MANAGEMENT (Continued)

Cluster Job Management Systems (JMS) (Continued)

Migration Schemes:

- **Node availability:** This refers to node availability for job migration. The Berkeley NOW project has reported such opportunity does exist in university campus environment. Even during peak hours, 60 percent of workstations in a cluster are available at Berkeley campus.
- **Migration overhead:** What is the effect of the migration overhead? The migration time can significantly slow down a parallel job. It is important to reduce the migration overhead (e.g., by improving the communication subsystem) or to migrate only rarely. The slowdown is significantly reduced if a parallel job is run on a cluster of twice the size.

CLUSTER JOB AND RESOURCE MANAGEMENT (Continued)

Cluster Job Management Systems (JMS) (Continued)

Migration Schemes (Continued):

- **Recruitment threshold:** In the worst scenario, right after a process migrates to a node, the node is immediately claimed by its owner. Thus, the process has to migrate again, and the cycle continues. The recruitment threshold is the amount of time a workstation stays unused before the cluster considers it an idle node.