CHAPTER 2: Computer Clusters for Scalable Parallel Computing

Presented by Faramarz Safi (Ph.D.)
Islamic Azad University, Najafabad Branch

SUMMARY

- Clustering of computers enables scalable parallel and distributed computing in **both science and business applications**.
- This chapter is devoted to building cluster-structured **massively parallel processors**.
- We focus on the design principles and assessment of the hardware, software, middleware, and operating system support to achieve scalability, availability, programmability, single-system images, and fault tolerance in clusters.
- Only physical clusters are studied in this chapter.
- Virtual clusters will be studied in Chapters 3 and 4.

CLUSTERING FOR MASSIVE PARALLELISM

- A computer cluster is a collection of interconnected stand-alone computers which can work together collectively and cooperatively as a single integrated computing resource pool.
- Clustering explores massive parallelism at the job level and achieves high availability (HA) through stand-alone operations.
- The benefits of computer clusters and massively parallel processors (MPPs) include scalable performance, HA, fault tolerance, modular growth, and use of commodity components.
- These features can sustain the generation changes experienced in hardware, software, and network components.
- Cluster computing became popular in the mid-1990s as **traditional mainframes and vector supercomputers were proven to be less cost-effective** in many high-performance computing (HPC) applications.

Cluster Development Trends

Milestone Cluster Systems

- Clustering has been **a hot research challenge in computer architecture**. **Fast communication, job scheduling, SSI, and HA are active areas in cluster research**. Table 2.1 lists some milestone cluster research projects and commercial cluster products. Details of these old clusters can be found in [14].

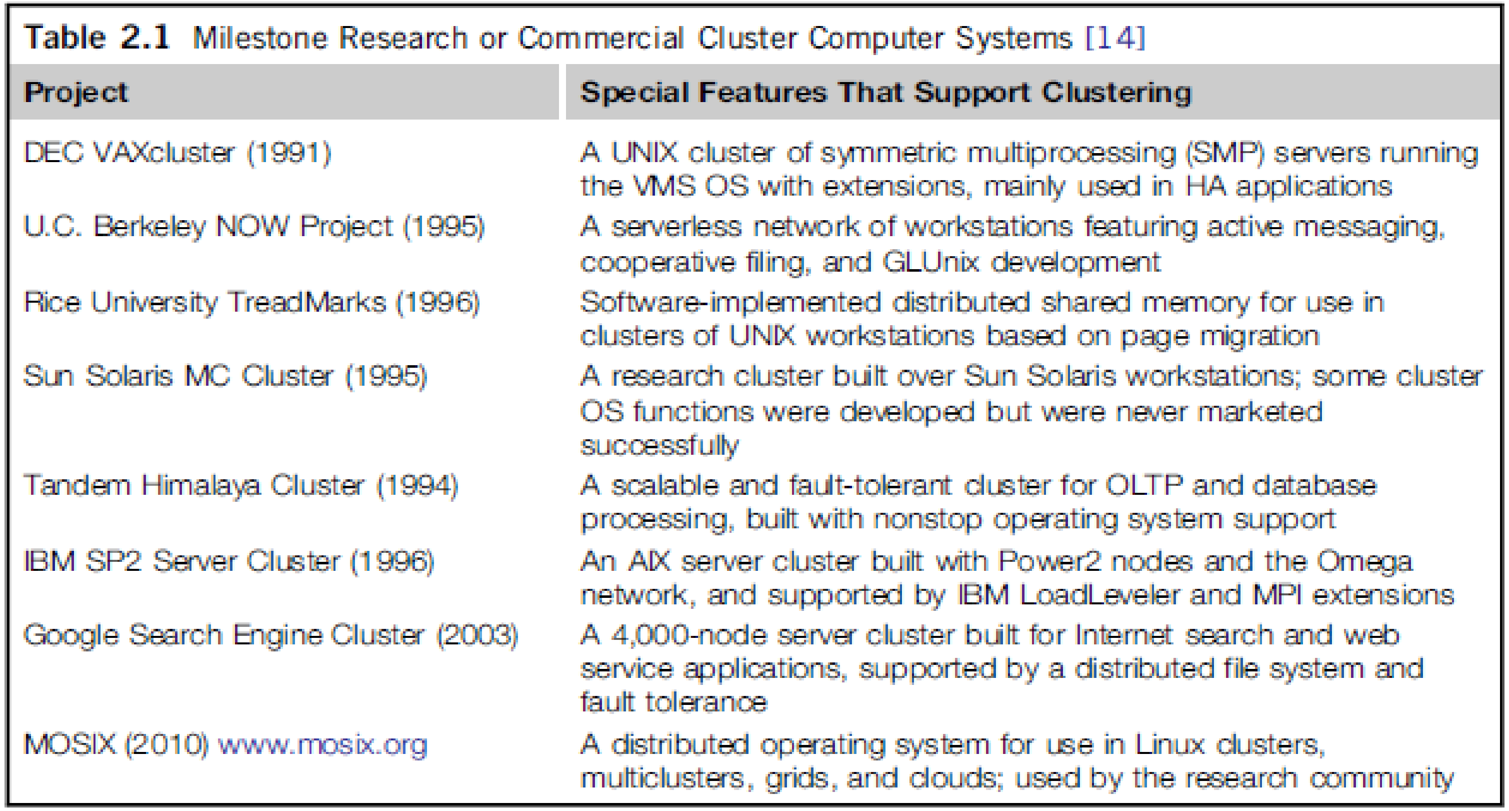

Table 2.1 Milestone Research or Commercial Cluster Computer Systems [14]

Project	Special Features That Support Clustering
DEC VAXcluster (1991)	A UNIX cluster of symmetric multiprocessing (SMP) servers running the VMS OS with extensions, mainly used in HA applications
U.C. Berkeley NOW Project (1995)	A serverless network of workstations featuring active messaging, cooperative filing, and GLUnix development
Rice University TreadMarks (1996)	Software-implemented distributed shared memory for use in clusters of UNIX workstations based on page migration
Sun Solaris MC Cluster (1995)	A research cluster built over Sun Solaris workstations; some cluster OS functions were developed but were never marketed successfully
Tandem Himalaya Cluster (1994)	A scalable and fault-tolerant cluster for OLTP and database processing, built with nonstop operating system support
IBM SP2 Server Cluster (1996)	An AIX server cluster built with Power2 nodes and the Omega network, and supported by IBM LoadLeveler and MPI extensions
Google Search Engine Cluster (2003)	A 4,000-node server cluster built for Internet search and web service applications, supported by a distributed file system and fault tolerance
MOSIX (2010) www.mosix.org	A distributed operating system for use in Linux clusters, multiclusters, grids, and clouds; used by the research community

2.1.2 Design Objectives of Computer Clusters

- Clusters have been classified in various ways in the literature. We classify clusters using **six orthogonal attributes: scalability, packaging, control, homogeneity, programmability, and security**.
- **Scalability:**
- Clustering of computers is based on the concept of modular growth.
- To scale a cluster from hundreds of uniprocessor nodes to a supercluster with 10,000 multicore nodes is a nontrivial task.
- The scalability could be limited by a number of factors, such as the multicore chip technology, cluster topology, packaging method, power consumption, and cooling scheme applied.
- The purpose is to achieve scalable performance constrained by the aforementioned factors.
- We have to also consider other limiting factors such as the memory wall, disk I/O bottlenecks, and latency tolerance, among others.

2.1.2 Design Objectives of Computer Clusters

- **Packaging:**
- Cluster nodes can be packaged in a **compact** or a **slack** fashion.
- In a **compact** cluster, 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.).
- In a **slack** cluster, 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.
- Packaging directly affects communication wire length, and thus the selection of interconnection technology used.
- While a compact cluster can utilize a high-bandwidth, low-latency communication network that is often proprietary, nodes of a slack cluster are normally connected through standard LANs or WANs.

2.1.2 Design Objectives of Computer Clusters

- **Control:**
- A cluster can be either controlled or managed in a **centralized** or **decentralized** fashion.
- A compact cluster normally has centralized control, while a slack cluster can be controlled either way.
- In a centralized cluster, all the nodes are owned, controlled, managed, and administered by a central operator.
- In a decentralized cluster, the nodes have individual owners. For instance, consider a cluster comprising an interconnected set of desktop workstations in a department, where each workstation is individually owned by an employee.
- The owner can reconfigure, upgrade, or even shut down the workstation at any time. 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, check pointing, accounting, and other similar tasks.

2.1.2 Design Objectives of Computer Clusters

- **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 **heterogeneous** cluster uses nodes of **different platforms**. Interoperability is an important issue in heterogeneous clusters.
- For instance, **process migration** is often needed for load balancing or availability. In a homogeneous cluster, a binary process image can migrate to another node and continue execution.
- This is not feasible in a heterogeneous cluster, as the binary code will not be executable when the process migrates to a node of a different platform.

2.1.2 Design Objectives of Computer Clusters

- **Security**
- **Intracluster communication** can be either **exposed** or **enclosed**.
- In an **exposed** cluster, 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).
- Such exposed clusters are easy to implement, but have several disadvantages: 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.
- In an **enclosed** cluster, 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.

2.1.2.6 Dedicated versus Enterprise Clusters

- **A dedicated cluster** 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.
- Dedicated clusters are used as substitutes for traditional mainframes or supercomputers.
- A dedicated cluster is installed, used, and administered as a single machine. Many users can log in to the cluster to execute both interactive and batch jobs.
- The cluster offers much enhanced throughput, as well as reduced response time.
- **An enterprise cluster** 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.

2.1.3 Fundamental Cluster Design Issues

- **Scalable Performance:** This refers to the fact that scaling of resources (cluster nodes, memory capacity, I/O bandwidth, etc.) leads to a proportional increase in performance. Of course, both **scale-up and scale-down capabilities** are needed, depending on application demand or cost-effectiveness considerations. Clustering is driven by scalability.
- **Single-System Image (SSI):** A set of workstations connected by an Ethernet network is not necessarily a cluster. **A cluster is a single system.** For example, suppose a workstation has a 300 Mflops/second processor, 512 MB of memory, and a 4 GB disk and can support 50 active users and 1,000 processes. **By clustering 100 such workstations, can we get a single system that is equivalent to one huge workstation**, or a mega-station, that has a 30 Gflops/second processor, 50 GB of memory, and a 400 GB disk and can support 5,000 active users and 100,000 processes? **SSI techniques are aimed at achieving this goal**.

2.1.3 Fundamental Cluster Design Issues

- **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**. However, to realize this potential, availability techniques are required.

- **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**. Special software tools are needed to manage multiple jobs simultaneously.

2.1.3 Fundamental Cluster Design Issues

- **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.

2.1.3 Fundamental Cluster Design Issues

- **Cluster Family Classification:**

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.

Beowulf cluster: 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.

Computational Grid: 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.

2.1.3 Fundamental Cluster Design Issues

Cluster Family Classification:

High-Availability clusters 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.

Load-balancing clusters: These clusters shoot for **higher resource utilization through load balancing** among all participating nodes in the cluster. 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**.

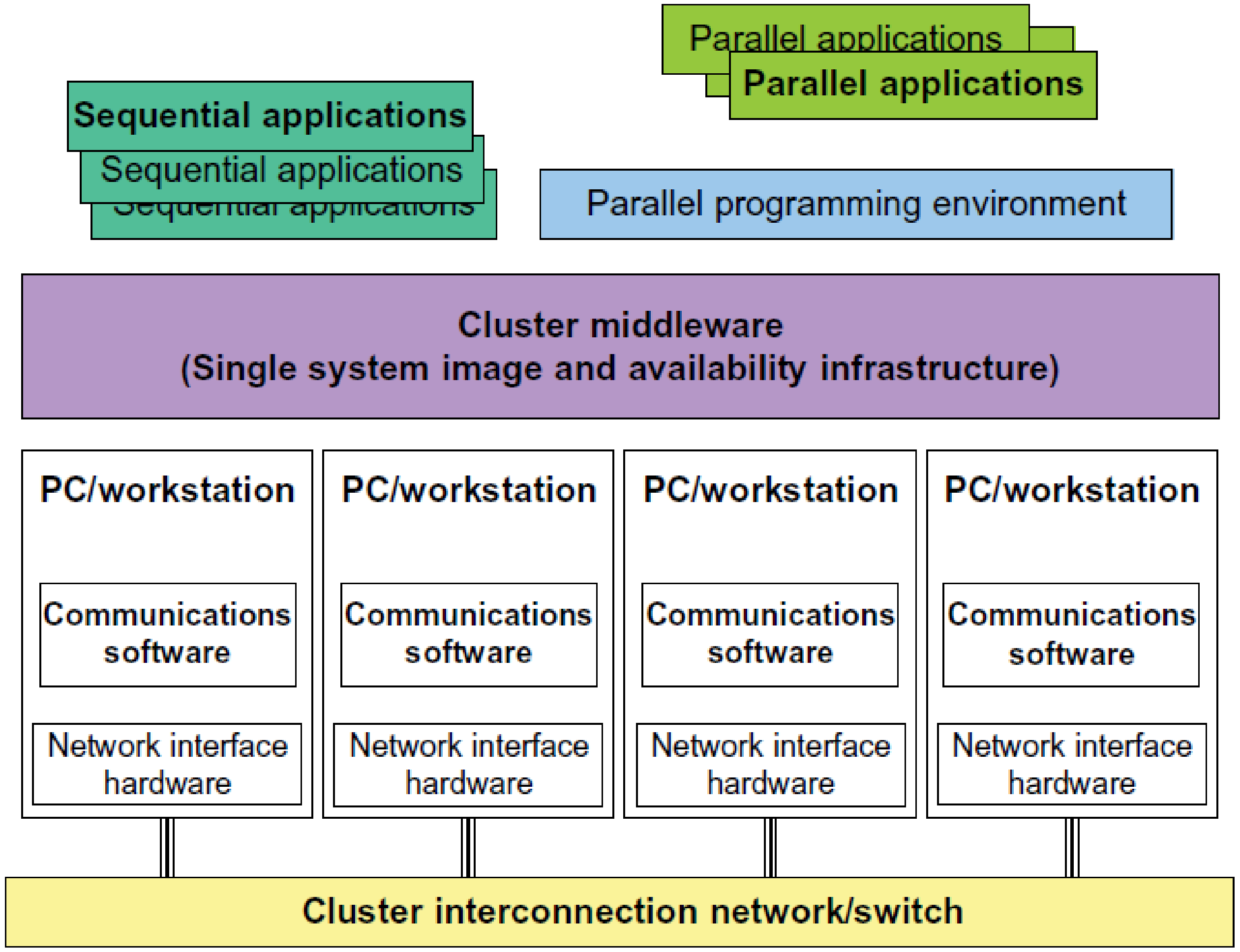

FIGURE 2.4

The architecture of a computer cluster built with commodity hardware, software, middleware, and network components supporting HA and SSI.

(*Courtesy of M. Baker and R. Buyya, reprinted with Permission [3]*)

2.2.1 Cluster Organization and Resource Sharing

2.2.1.1 A Basic Cluster Architecture

- Figure 2.4 shows the basic architecture of a computer cluster over PCs or workstations.
- The figure shows a simple cluster of computers built with commodity components and fully supported with desired SSI features and HA capability.
- The **processing nodes are commodity workstations, PCs, or servers**. These commodity nodes are 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).

2.2.1.1 A Basic Cluster Architecture

- When the processor or the operating system is changed, only the driver software needs to change.
- We desire to have a platform-independent cluster operating system, sitting on top of the node platforms. But such a cluster OS is not commercially available.
- Instead, we can **deploy some cluster middleware to glue together all node platforms at the user space**. An availability middleware offers HA services.
- An **SSI layer** provides a **single entry point**, a **single file hierarchy**, a **single point of control**, and a **single job management system**. **Single memory** may be realized with the help of the compiler or a runtime library.
- A **single process space** is not necessarily supported.

2.2.1.1 A Basic Cluster Architecture

- In general, an idealized cluster is supported by **three subsystems**:
- First, **conventional databases and 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.

2.2.1.2 Resource Sharing in Clusters

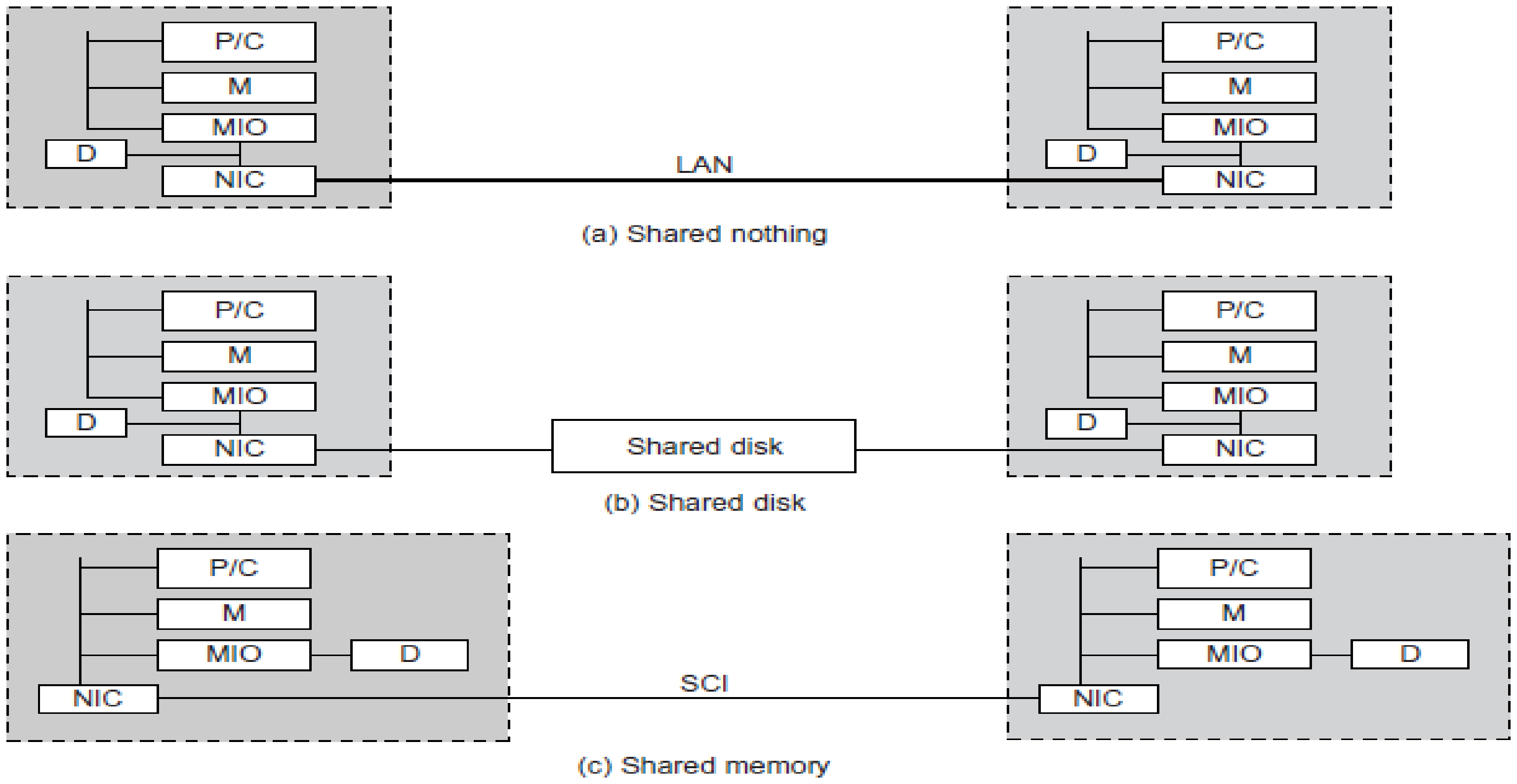

FIGURE 2.5

Three ways to connect cluster nodes (P/C: Processor and Cache; M: Memory; D: Disk; NIC: Network Interface Circuitry; MIO: Memory-I/O Bridge.)

(Courtesy of Hwang and Xu [14])

There is no widely accepted standard for the memory bus. But there are such standards for the I/O buses. One recent, popular standard is the PCI I/O bus standard. So, if you implement an NIC card to attach a faster Ethernet network to the PCI bus you can be assured that this card can be used in other systems that use PCI as the I/O bus.

2.2.1.2 Resource Sharing in Clusters

- The **nodes of a cluster can be connected in one of three ways**, as shown in Figure 2.5:

1. The **shared-nothing architecture** is used in most clusters, where the nodes are connected through the I/O bus. The shared-nothing configuration in Part (a) simply connects two or more autonomous computers via a LAN such as Ethernet.
2. The **shared-disk architecture** is in favor of small-scale availability clusters in business applications. When one node fails, the other node takes over. A shared-disk cluster is shown in Part (b). This is what 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.
3. The **shared-memory** cluster in Part (c) is much more difficult to realize. 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.

2.2.2 Node Architectures and MPP Packaging

- In building large-scale clusters or MPP systems, **cluster nodes are classified into two categories: 1) compute nodes, and 2) service nodes**.
- Compute nodes appear in larger quantities mainly **used for largescale searching or parallel floating-point computations**.
- Service nodes could be built with different processors **mainly used to handle I/O, file access, and system monitoring**.
- For MPP clusters, the compute nodes dominate in system cost, because we may have 1,000 times more compute nodes than service nodes in a single large clustered system.
- In the past, most MPPs are built with a homogeneous architecture by interconnecting a large number of the same compute nodes.
- In 2010, the Cray XT5 Jaguar system was built with 224,162 AMD Opteron processors with six cores each. The Tiahe-1A adopted a hybrid node design using two Xeon CPUs plus two AMD GPUs per each compute node.
- The GPU could be replaced by special floating-point accelerators. A homogeneous node design makes it easier to program and maintain the system.

2.2.2 Node Architectures and MPP Packaging

- Table 2.3 introduces two example compute node architectures: homogeneous design and hybrid node design.

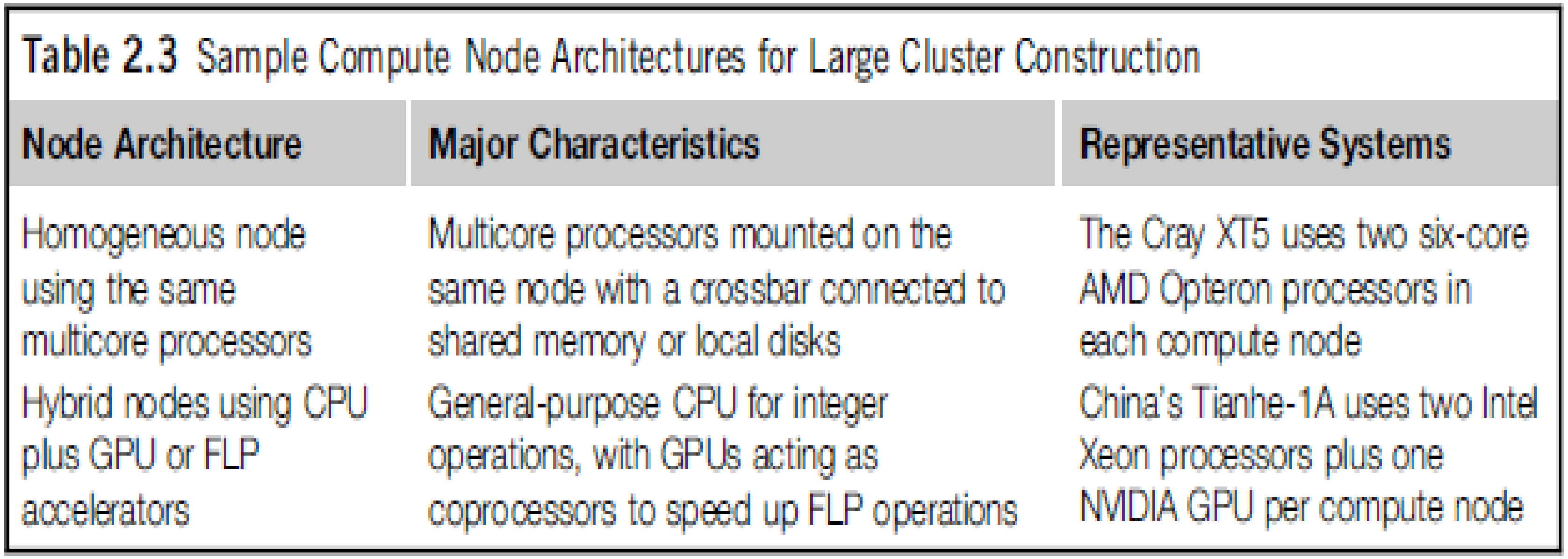

Table 2.3 Sample Compute Node Architectures for Large Cluster Construction

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

Modular Packaging of the IBM Blue Gene/L System

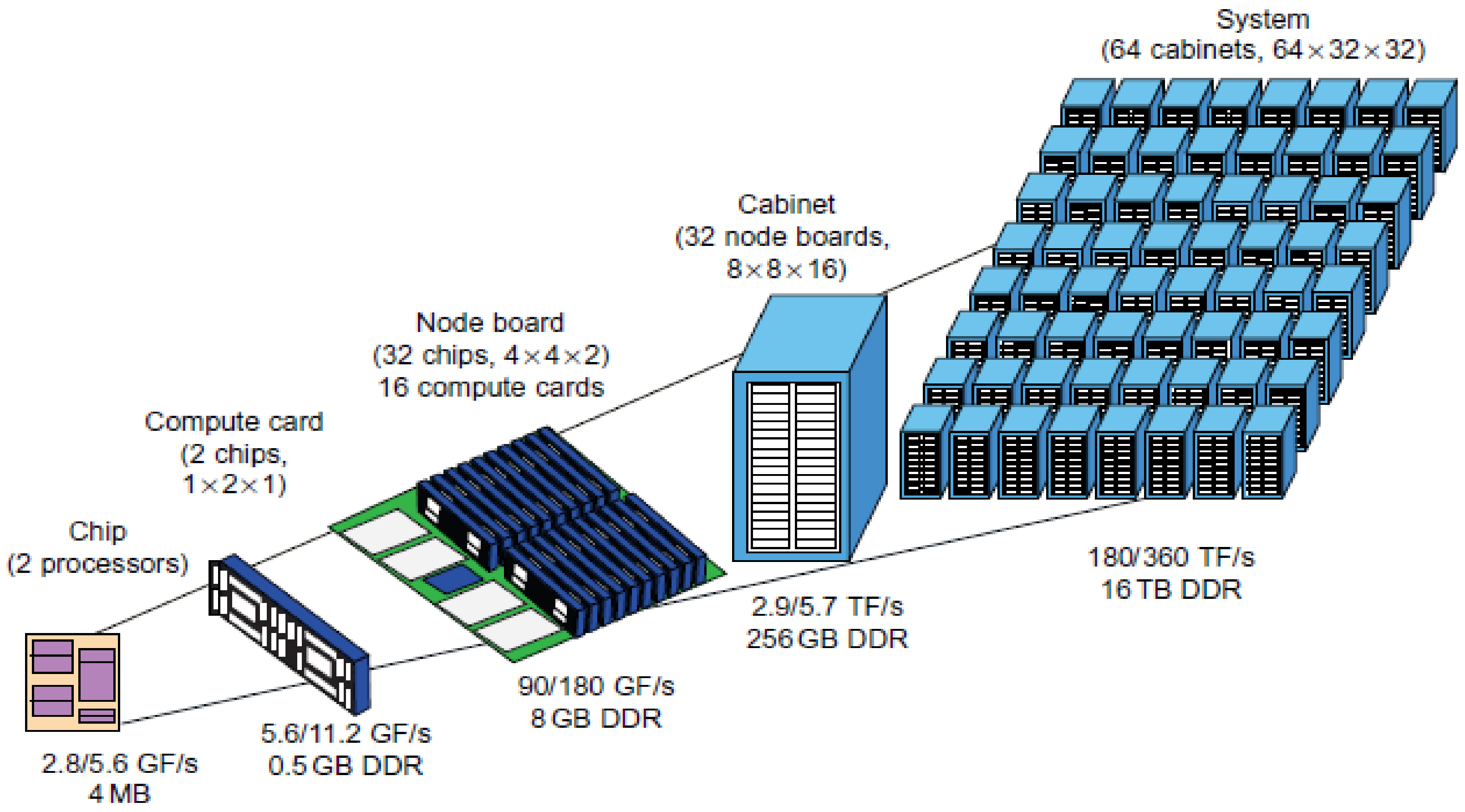

FIGURE 2.6

The IBM Blue Gene/L architecture built with modular components packaged hierarchically in five levels.

(Courtesy of N. Adiga, et al., IBM Corp., 2005 [1])

Modular Packaging of the IBM Blue Gene/L System

- The Blue Gene/L is a supercomputer jointly developed by IBM and Lawrence Livermore National Laboratory.
- The system became operational in 2005 with a 136 Tflops performance at the No. 1 position in the Top-500 list—toped the Japanese Earth Simulator.
- The system was upgraded to score a 478 Tflops speed in 2007. By examining the architecture of the Blue Gene series, we reveal the modular construction of a scalable MPP system as shown in Figure 2.6.
- With modular packaging, the Blue Gene/L system is constructed hierarchically from processor chips to 64 physical racks.
- This system was built with a total of 65,536 nodes with two PowerPC 449 FP2 processors per node.
- The 64 racks are interconnected by a huge 3D 64 x 32 x 32 torus network.

2.2.4 Hardware, Software, and Middleware Support

- Realistically, **SSI and HA features in a cluster must be supported by hardware, software, middleware, or OS extensions**.
- Any **change in hardware design and OS extensions** must be done by the manufacturer. The hardware and OS support could be cost-prohibitive to ordinary users.
- However, programming level is a big burden to cluster users. Therefore, the **middleware support at the application level** costs the least to implement.
- As an example, we show in Figure 2.10 the middleware, OS extensions, and hardware support needed to achieve HA in a typical Linux cluster system.
- Close to the user application end, **middleware packages are also needed at the cluster management level: one for fault management** to support failover and failback, to be discussed in Section 2.3.3.
- Another desired feature is to achieve HA using **failure detection and recovery and packet switching**. In the middle of Figure 2.10, we need to modify the Linux OS to support HA, and we need special drivers to support HA, I/O, and hardware devices.
- Toward the bottom, we need **special hardware to support hot-swapped devices and provide router interfaces**. We will discuss various supporting mechanisms in subsequent sections.

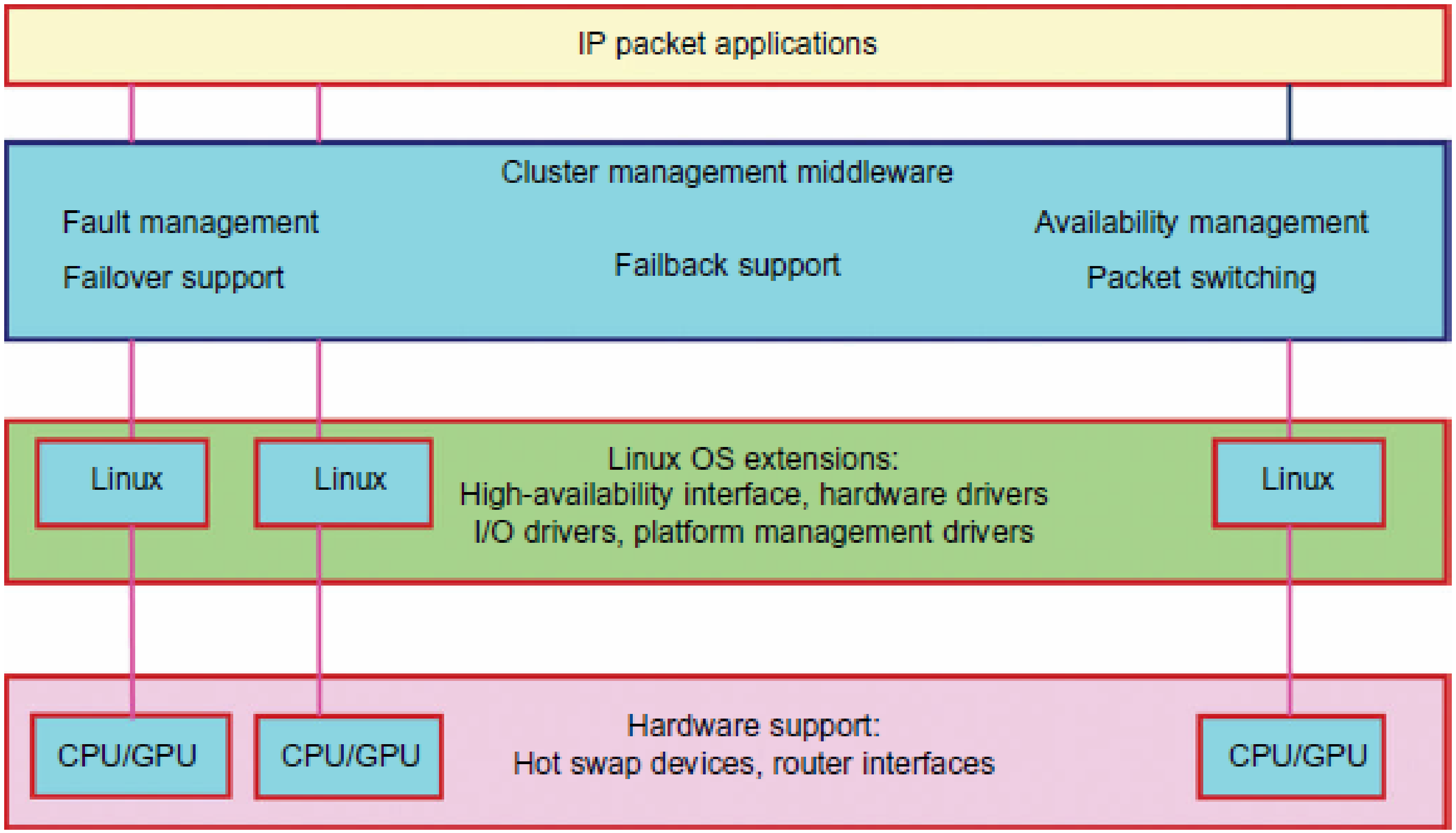

FIGURE 2.10

Middleware, Linux extensions, and hardware support for achieving massive parallelism and HA in a Linux cluster system built with CPUs and GPUs.

2.2.5 GPU Clusters for Massive Parallelism

- Commodity GPUs are becoming high-performance accelerators for data-parallel computing.
- Modern GPU chips contain hundreds of processor cores per chip. Based on a 2010 report [19], each GPU chip is capable of achieving up to 1 Tflops for single-precision (SP) arithmetic, and more than 80 Gflops for double-precision (DP) calculations.
- 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. GPU clusters have already demonstrated their capability to achieve Pflops performance in some of the Top 500 systems.
- 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.
- The high performance of a GPU cluster is attributed mainly to its massively parallel multicore architecture, high throughput in multithreaded floating-point arithmetic, and significantly reduced time in massive data movement using large on-chip cache memory.
- In other words, 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. A GPU cluster can operate with a reduced number of operating system images, compared with CPU-based clusters.
- These reductions in power, environment, and management complexity make GPU clusters very attractive for use in future HPC applications.

CUDA Parallel Programming Interfaces

- **CUDA (Compute Unified Device Architecture) offers a parallel computing architecture** developed by NVIDIA.
- **CUDA is the computing engine in NVIDIA GPUs**.
- This software is accessible to developers through standard programming languages. Programmers use C for CUDA C with NVIDIA extensions and certain restrictions.
- This CUDA C is compiled through a Path Scale Open64 C compiler for parallel execution on a large number of GPU cores. Example 2.4 shows the advantage of using CUDA C in parallel processing.
- CUDA architecture shares a range of computational interfaces with two competitors: the Khronos Group's Open Computing Language and Microsoft's DirectCompute. Third-party wrappers are also available for using Python, Perl, FORTRAN, Java, Ruby, Lua, MATLAB, and IDL.
- CUDA has been used to accelerate non-graphical applications in **computational biology, cryptography, and other fields** by an order of magnitude or more.
- A good example is the BOINC distributed computing client.

Trends in CUDA Usage

- Tesla and Fermi are two generations of CUDA architecture released by NVIDIA in 2007 and 2010, respectively.
- The CUDA version 3.2 is used for using a single GPU module in 2010.
- A newer CUDA version 4.0 will allow multiple GPUs to address use an unified virtual address space of shared memory.
- The next NVIDIA GPUs will be Kepler-designed to support C++. The Fermi has eight times the peak double-precision floating-point performance of the Tesla GPU (5.8 Gflops/W
- versus 0.7 Gflops/W).
- Currently, the Fermi has up to 512 CUDA cores on 3 billion transistors. Future applications of the CUDA GPUs and the Echelon system may include the following:
- The search for extraterrestrial intelligence (SETI@Home)
- Distributed calculations to predict the native conformation of proteins
- Medical analysis simulations based on CT and MRI scan images
- Physical simulations in fluid dynamics and environment statistics
- Accelerated 3D graphics, cryptography, compression, and interconversion of video file formats
- Building the single-chip cloud computer (SCC) through virtualization in many-core architecture.