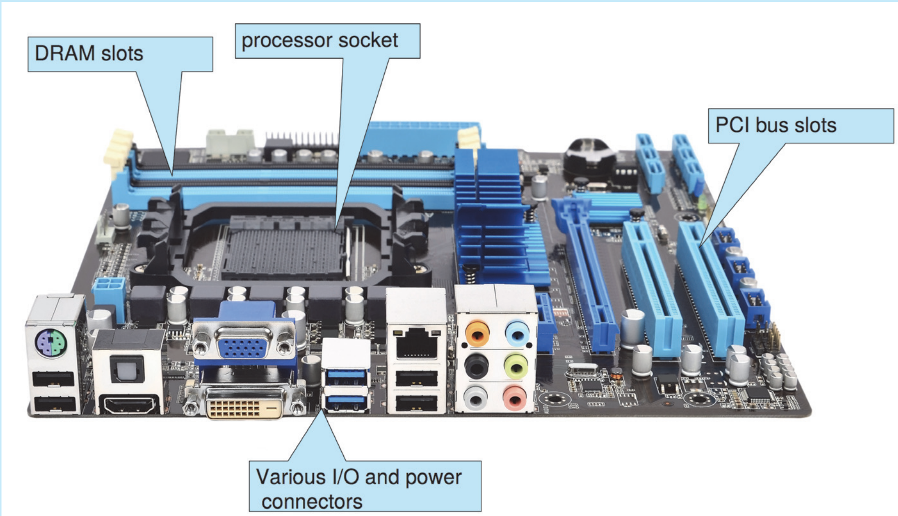
PC MOTHERBOARD

Consider the desktop PC motherboard with a processor socket shown below:



This board is a fully functioning computer, once its slots are populated. It consists of a processor socket containing a CPU, DRAM sockets, PCIe bus slots, and I/O connectors of various types. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.

server. In **symmetric clustering**, two or more hosts are running applications and are monitoring each other. This structure is obviously more efficient, as it uses all of the available hardware. However, it does require that more than one application be available to run.

Since a cluster consists of several computer systems connected via a network, clusters can also be used to provide **high-performance computing** environments. Such systems can supply significantly greater computational power than single-processor or even SMP systems because they can run an application concurrently on all computers in the cluster. The application must have been written specifically to take advantage of the cluster, however. This involves a technique known as **parallelization**, which divides a program into separate components that run in parallel on individual cores in a computer or computers in a cluster. Typically, these applications are designed so that once each computing node in the cluster has solved its portion of the problem, the results from all the nodes are combined into a final solution.

Other forms of clusters include parallel clusters and clustering over a wide-area network (WAN) (as described in Chapter 19). Parallel clusters allow multiple hosts to access the same data on shared storage. Because most oper-

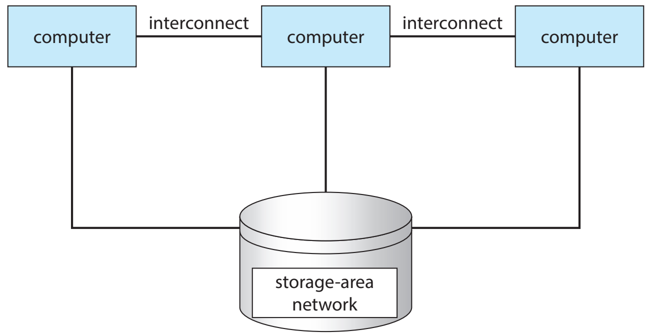


Figure 1.11 General structure of a clustered system.

ating systems lack support for simultaneous data access by multiple hosts, parallel clusters usually require the use of special versions of software and special releases of applications. For example, Oracle Real Application Cluster is a version of Oracle's database that has been designed to run on a parallel cluster. Each machine runs Oracle, and a layer of software tracks access to the shared disk. Each machine has full access to all data in the database. To provide this shared access, the system must also supply access control and locking to ensure that no conflicting operations occur. This function, commonly known as a **distributed lock manager** (**DLM**), is included in some cluster technology.

Cluster technology is changing rapidly. Some cluster products support thousands of systems in a cluster, as well as clustered nodes that are separated by miles. Many of these improvements are made possible by **storage-area networks** (**SANs**), as described in Section 11.7.4, which allow many systems to attach to a pool of storage. If the applications and their data are stored on the SAN, then the cluster software can assign the application to run on any host that is attached to the SAN. If the host fails, then any other host can take over. In a database cluster, dozens of hosts can share the same database, greatly increasing performance and reliability. Figure 1.11 depicts the general structure of a clustered system.

1.4 Operating-System Operations

Now that we have discussed basic information about computer-system organization and architecture, we are ready to talk about operating systems. An operating system provides the environment within which programs are executed. Internally, operating systems vary greatly, since they are organized along many different lines. There are, however, many commonalities, which we consider in this section.

For a computer to start running—for instance, when it is powered up or rebooted—it needs to have an initial program to run. As noted earlier, this initial program, or bootstrap program, tends to be simple. Typically, it is stored within the computer hardware in firmware. It initializes all aspects of the system, from CPU registers to device controllers to memory contents. The bootstrap program must know how to load the operating system and how to

HADOOP

Hadoop is an open-source software framework that is used for distributed processing of large data sets (known as **big data**) in a clustered system containing simple, low-cost hardware components. Hadoop is designed to scale from a single system to a cluster containing thousands of computing nodes. Tasks are assigned to a node in the cluster, and Hadoop arranges communication between nodes to manage parallel computations to process and coalesce results. Hadoop also detects and manages failures in nodes, providing an efficient and highly reliable distributed computing service.

Hadoop is organized around the following three components:

1. A distributed file system that manages data and files across distributed computing nodes.
2. The YARN ("Yet Another Resource Negotiator") framework, which manages resources within the cluster as well as scheduling tasks on nodes in the cluster.
3. The **MapReduce** system, which allows parallel processing of data across nodes in the cluster.

Hadoop is designed to run on Linux systems, and Hadoop applications can be written using several programming languages, including scripting languages such as PHP, Perl, and Python. Java is a popular choice for developing Hadoop applications, as Hadoop has several Java libraries that support MapReduce. More information on MapReduce and Hadoop can be found at https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html and https://hadoop.apache.org

start executing that system. To accomplish this goal, the bootstrap program must locate the operating-system kernel and load it into memory.

Once the kernel is loaded and executing, it can start providing services to the system and its users. Some services are provided outside of the kernel by system programs that are loaded into memory at boot time to become **system daemons**, which run the entire time the kernel is running. On Linux, the first system program is "`systemd`," and it starts many other daemons. Once this phase is complete, the system is fully booted, and the system waits for some event to occur.

If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system will sit quietly, waiting for something to happen. Events are almost always signaled by the occurrence of an interrupt. In Section 1.2.1 we described hardware interrupts. Another form of interrupt is a **trap** (or an **exception**), which is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed by executing a special operation called a **system call**.