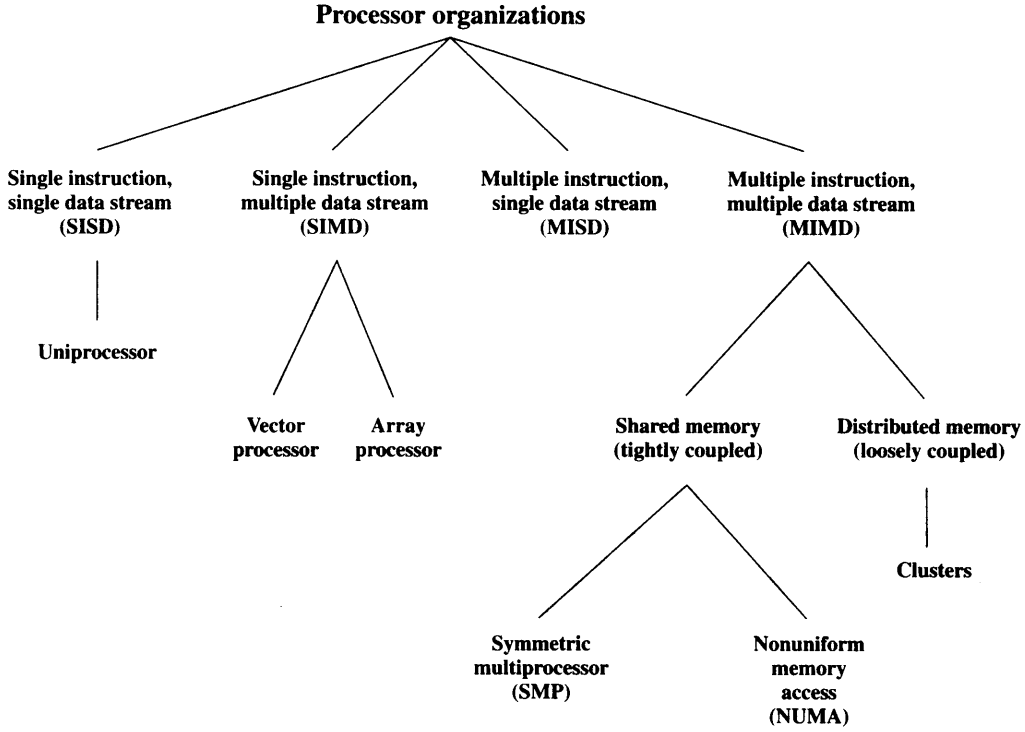


Figure 18.1 A Taxonomy of Parallel Processor Architectures

then each processor accesses programs and data stored in the shared memory, and processors communicate with each other via that memory. The most common form of such system is known as a **symmetric multiprocessor (SMP)**, which we examine in Section 18.2. In an SMP, multiple processors share a single memory or pool of memory by means of a shared bus or other interconnection mechanism; a distinguishing feature is that the memory access time to any region of memory is approximately the same for each processor. A more recent development is the **nonuniform memory access (NUMA)** organization, which is described in Section 18.5. As the name suggests, the memory access time to different regions of memory may differ for a NUMA processor.

A collection of independent uniprocessors or SMPs may be interconnected to form a **cluster**. Communication among the computers is either via fixed paths or via some network facility.

Parallel Organizations

Figure 18.2 illustrates the general organization of the taxonomy of Figure 18.1. Figure 18.2a shows the structure of an SISD. There is some sort of control unit (CU) that provides an instruction stream (IS) to a processing unit (PU). The processing unit operates on a single data stream (DS) from a memory unit (MU). With an SIMD, there is still a single control unit, now feeding a single instruction stream to multiple PUs. Each PU may have its own dedicated memory (illustrated in Figure 18.2b), or there may be

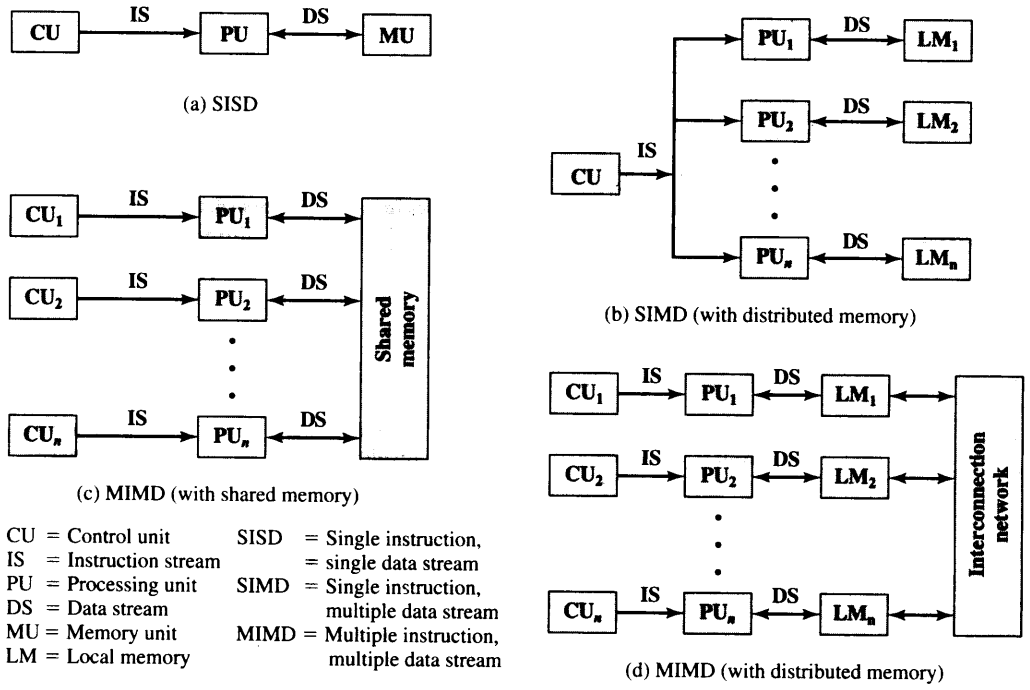


Figure 18.2 Alternative Computer Organizations

a shared memory. Finally, with the MIMD, there are multiple control units, each feeding a separate instruction stream to its own PU. The MIMD may be a shared-memory multiprocessor (Figure 18.2c) or a distributed-memory multicomputer (Figure 18.2d).

The design issues relating to SMPs, clusters, and NUMAs are complex, involving issues relating to physical organization, interconnection structures, interprocessor communication, operating system design, and application software techniques. Our concern here is primarily with organization, although we touch briefly on operating system design issues.

18.2 SYMMETRIC MULTIPROCESSORS

Until fairly recently, virtually all single-user personal computers and most workstations contained a single general-purpose microprocessor. As demands for performance increase and as the cost of microprocessors continues to drop, vendors have introduced systems with an SMP organization. The term *SMP* refers to a computer hardware architecture and also to the operating system behavior that reflects that architecture. An SMP can be defined as a standalone computer system with the following characteristics:

1. There are two or more similar processors of comparable capability.
2. These processors share the same main memory and I/O facilities and are interconnected by a bus or other internal connection scheme, such that memory access time is approximately the same for each processor.

3. All processors share access to I/O devices, either through the same channels or through different channels that provide paths to the same device.
4. All processors can perform the same functions (hence the term *symmetric*).
5. The system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels.

Points 1 to 4 should be self-explanatory. Point 5 illustrates one of the contrasts with a loosely coupled multiprocessing system, such as a cluster. In the latter, the physical unit of interaction is usually a message or complete file. In an SMP, individual data elements can constitute the level of interaction, and there can be a high degree of cooperation between processes.

The operating system of an SMP schedules processes or threads across all of the processors. An SMP organization has a number of potential advantages over a uniprocessor organization, including the following:

- **Performance:** If the work to be done by a computer can be organized so that some portions of the work can be done in parallel, then a system with multiple processors will yield greater performance than one with a single processor of the same type (Figure 18.3).

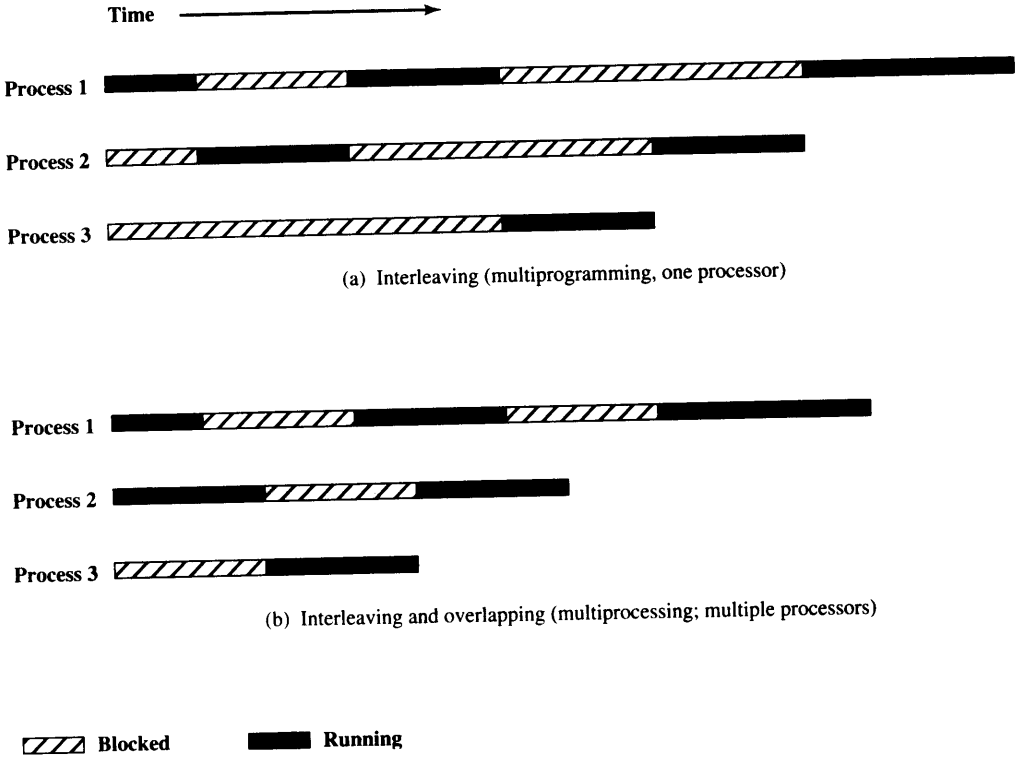


Figure 18.3 Multiprogramming and Multiprocessing

- **Time sharing:** When one module is controlling the bus, other modules are locked out and must, if necessary, suspend operation until bus access is achieved.

These uniprocessor features are directly usable in an SMP organization. In this latter case, there are now multiple processors as well as multiple I/O processors all attempting to gain access to one or more memory modules via the bus.

The bus organization has several attractive features:

- **Simplicity:** This is the simplest approach to multiprocessor organization. The physical interface and the addressing, arbitration, and time-sharing logic of each processor remain the same as in a single-processor system.
- **Flexibility:** It is generally easy to expand the system by attaching more processors to the bus.
- **Reliability:** The bus is essentially a passive medium, and the failure of any attached device should not cause failure of the whole system.

The main drawback to the bus organization is performance. All memory references pass through the common bus. Thus, the bus cycle time limits the speed of the system. To improve performance, it is desirable to equip each processor with a cache memory. This should reduce the number of bus accesses dramatically. Typically, workstation and PC SMPs have two levels of cache, with the L1 cache internal (same chip as the processor) and the L2 cache either internal or external. Some processors now employ a L3 cache as well.

The use of caches introduces some new design considerations. Because each local cache contains an image of a portion of memory, if a word is altered in one cache, it could conceivably invalidate a word in another cache. To prevent this, the other processors must be alerted that an update has taken place. This problem is known as the *cache coherence* problem and is typically addressed in hardware rather than by the operating system. We address this issue in Section 18.4.

Multiprocessor Operating System Design Considerations

An SMP operating system manages processor and other computer resources so that the user perceives a single operating system controlling system resources. In fact, such a configuration should appear as a single-processor multiprogramming system. In both the SMP and uniprocessor cases, multiple jobs or processes may be active at one time, and it is the responsibility of the operating system to schedule their execution and to allocate resources. A user may construct applications that use multiple processes or multiple threads within processes without regard to whether a single processor or multiple processors will be available. Thus a multiprocessor operating system must provide all the functionality of a multiprogramming system plus additional features to accommodate multiple processors. Among the key design issues are the following:

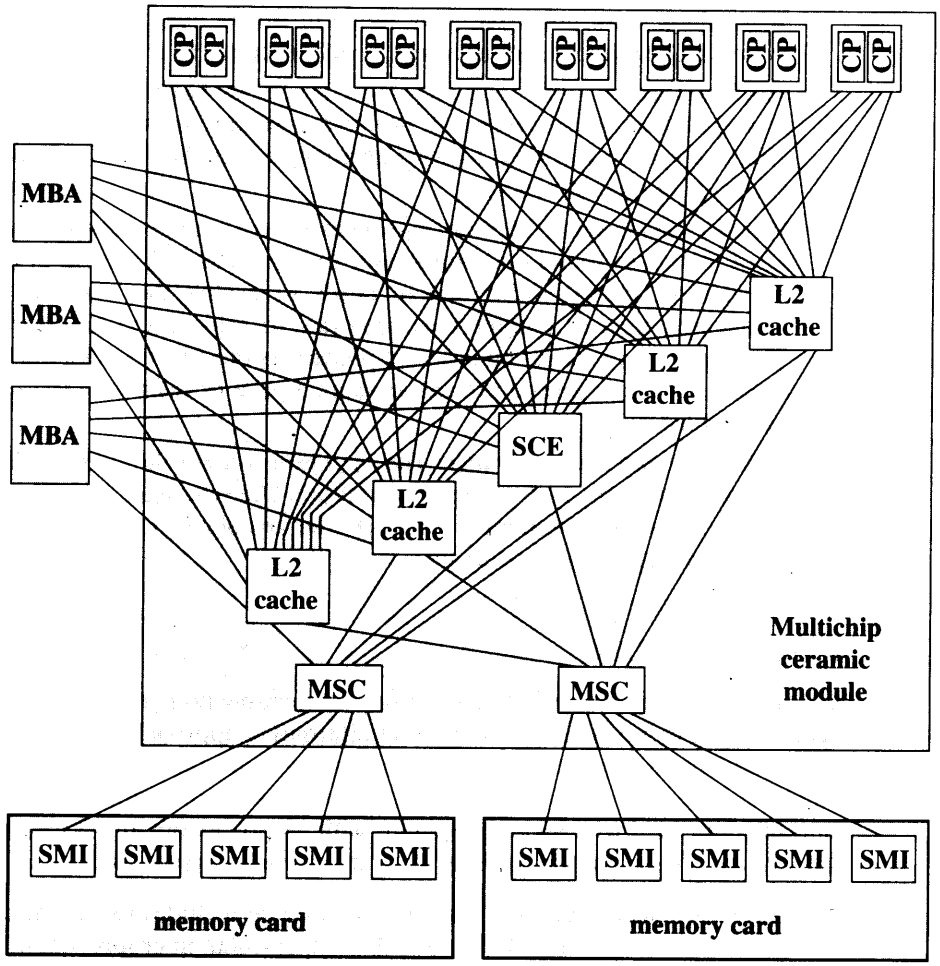
- **Simultaneous concurrent processes:** OS routines need to be reentrant to allow several processors to execute the same IS code simultaneously. With multiple processors executing the same or different parts of the OS, OS tables and management structures must be managed properly to avoid deadlock or invalid operations.

- **Scheduling:** Any processor may perform scheduling, so conflicts must be avoided. The scheduler must assign ready processes to available processors.
- **Synchronization:** With multiple active processes having potential access to shared address spaces or shared I/O resources, care must be taken to provide effective synchronization. Synchronization is a facility that enforces mutual exclusion and event ordering.
- **Memory management:** Memory management on a multiprocessor must deal with all of the issues found on uniprocessor machines, as is discussed in Chapter 8. In addition, the operating system needs to exploit the available hardware parallelism, such as multiported memories, to achieve the best performance. The paging mechanisms on different processors must be coordinated to enforce consistency when several processors share a page or segment and to decide on page replacement.
- **Reliability and fault tolerance:** The operating system should provide graceful degradation in the face of processor failure. The scheduler and other portions of the operating system must recognize the loss of a processor and restructure management tables accordingly.

A Mainframe SMP

Most PC and workstation SMPs use a bus interconnection strategy as depicted in Figure 18.5. It is instructive to look at an alternative approach, which is used for a recent implementation of the IBM zSeries mainframe family [SIEG04, MAK04], called the z990. This family of systems spans a range from a uniprocessor with one main memory card to a high-end system with 48 processors and 8 memory cards. The key components of the configuration are shown in Figure 18.6:

- **Dual-core processor chip:** Each processor chip includes two identical central processors (CPs). The CP is a CISC superscalar microprocessor, in which most of the instructions are hardwired and the rest are executed by vertical microcode. Each CP includes a 256-KB L1 instruction cache and a 256-KB L1 data cache.
- **L2 cache:** Each L2 cache contains 32 MB. The L2 caches are arranged in clusters of five, with each cluster supporting eight processor chips and providing access to the entire main memory space.
- **System control element (SCE):** The SCE arbitrates system communication, and has a central role in maintaining cache coherence.
- **Main store control (MSC):** The MSCs interconnect the L2 caches and the main memory.
- **Memory card:** Each card holds 32 GB of memory. The maximum configurable memory consists of 8 memory cards for a total of 256 GB. Memory cards interconnect to the MSC via synchronous memory interfaces (SMIs).
- **Memory bus adapter (MBA):** The MBA provides an interface to various types of I/O channels. Traffic to/from the channels goes directly to the L2 cache.



CP = central processor
MBA = memory bus adapter
MSC = main store control
SCE = system control element
SMI = synchronous memory interface

Figure 18.6 IBM z990 Multiprocessor Structure

The microprocessor in the z990 is relatively uncommon compared with other modern processors because, although it is superscalar, it executes instructions in strict architectural order. However, it makes up for this by having a shorter pipeline and much larger caches and TLBs compared with other processors, along with other performance-enhancing features.

The z990 system comprises one to four **books**. Each book is a pluggable unit containing up to 12 processors with up to 64 GB of memory, I/O adapters, and a system control element (SCE) that connects these other elements. The SCE within

each book contains a 32-MB L2 cache, which serves as the central coherency point for that particular book. Both the L2 cache and the main memory are accessible by a processor or I/O adapter within that book or any of the other three books in the system. The SCE and L2 cache chips also connect with corresponding elements on the other books in a ring configuration.

There are a several interesting features in the z990 SMP configuration, which we discuss in turn:

- Switched interconnection
- Shared L2 caches

Switched Interconnection A single shared bus is a common arrangement on SMPs for PCs and workstations (Figure 18.5). With this arrangement, the single bus becomes a bottleneck affecting the scalability (ability to scale to larger sizes) of the design. The z990 copes with this problem in two ways. First, main memory is split into multiple cards, each with its own storage controller that can handle memory accesses at high speeds. The average traffic load to main memory is cut, because of the independent paths to separate parts of memory. Each book includes two memory cards, for a total of eight cards across a maximum configuration. Second, the connection from processors (actually from L2 caches) to a single memory card is not in the form of a shared bus but rather point-to-point links. Each processor chip has a link to each of the L2 caches on the same book, and each L2 cache has a link, via the MSC, to each of the two memory cards on the same book.

Each L2 cache only connects to the two memory cards on the same book. The system controller provides links (not shown) to the other books in the configuration, so that all of main memory is accessible by all of the processors.

Point-to-point links rather than a bus also provides connections to I/O channels. Each L2 cache on a book connects to each of the MBAs for that book. The MBAs, in turn, connect to the I/O channels.

Shared L2 Caches In a typical two-level cache scheme for an SMP, each processor has a dedicated L1 cache and a dedicated L2 cache. In recent years, interest in the concept of a shared L2 cache has been growing. In an earlier version of its mainframe SMP, known as generation 3 (G3), IBM made use of dedicated L2 caches. In its later versions (G4, G5, and z900 series), a shared L2 cache is used. Two considerations dictated this change:

1. In moving from G3 to G4, IBM doubled the speed of the microprocessors. If the G3 organization were retained, a significant increase in bus traffic would occur. At the same time, it was desired to reuse as many G3 components as possible. Without a significant bus upgrade, the BSNs would become a bottleneck.
2. Analysis of typical mainframe workloads revealed a large degree of sharing of instructions and data among processors.

These considerations led the G4 design team to consider the use of one or more L2 caches, each of which was shared by multiple processors (each processor having a dedicated on-chip L1 cache). At first glance, sharing an L2 cache might

seem a bad idea. Access to memory from processors should be slower because the processors must now contend for access to a single L2 cache. However, if a sufficient amount of data is in fact shared by multiple processors, then a shared cache can increase throughput rather than retard it. Data that are shared and found in the shared cache are obtained more quickly than if they must be obtained over the bus.

18.3 CACHE COHERENCE AND THE MESI PROTOCOL

In contemporary multiprocessor systems, it is customary to have one or two levels of cache associated with each processor. This organization is essential to achieve reasonable performance. It does, however, create a problem known as the *cache coherence* problem. The essence of the problem is this: Multiple copies of the same data can exist in different caches simultaneously, and if processors are allowed to update their own copies freely, an inconsistent view of memory can result. In Chapter 4 we defined two common write policies:

- **Write back:** Write operations are usually made only to the cache. Main memory is only updated when the corresponding cache line is flushed from the cache.
- **Write through:** All write operations are made to main memory as well as to the cache, ensuring that main memory is always valid.

It is clear that a write-back policy can result in inconsistency. If two caches contain the same line, and the line is updated in one cache, the other cache will unknowingly have an invalid value. Subsequent reads to that invalid line produce invalid results. Even with the write-through policy, inconsistency can occur unless other caches monitor the memory traffic or receive some direct notification of the update.

In this section, we will briefly survey various approaches to the cache coherence problem and then focus on the approach that is most widely used: the MESI (modified/exclusive/shared/invalid) protocol. A version of this protocol is used on both the Pentium 4 and PowerPC implementations.

For any cache coherence protocol, the objective is to let recently used local variables get into the appropriate cache and stay there through numerous reads and write, while using the protocol to maintain consistency of shared variables that might be in multiple caches at the same time. Cache coherence approaches have generally been divided into software and hardware approaches. Some implementations adopt a strategy that involves both software and hardware elements. Nevertheless, the classification into software and hardware approaches is still instructive and is commonly used in surveying cache coherence strategies.

Software Solutions

Software cache coherence schemes attempt to avoid the need for additional hardware circuitry and logic by relying on the compiler and operating system to deal with the problem. Software approaches are attractive because the overhead of detecting potential problems is transferred from run time to compile time, and the design complexity is transferred from hardware to software. On the other hand,