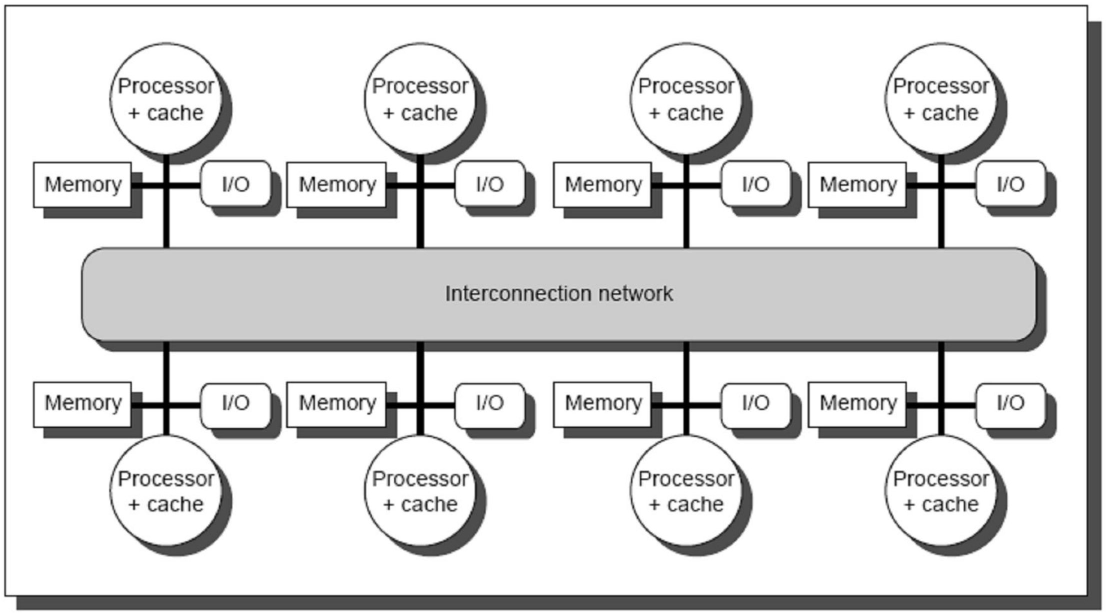
Distributed-memory multiprocessor

second group consists of multiprocessors with physically distributed memory. To support larger processor counts, memory must be distributed among the processors rather than centralized; otherwise the memory system would not be able to support the bandwidth demands of a larger number of processors without incurring excessively long access latency. With the rapid increase in processor performance and the associated increase in a processor's memory bandwidth requirements, the scale of multiprocessor for which distributed memory is preferred over a single, centralized memory continues to decrease in number (which is another reason not to use small and large scale). Of course, the larger number of processors raises the need for a high bandwidth interconnect. Both direct interconnection networks (i.e., switches) and indirect networks (typically multidimensional meshes) are used. Figure 6.2 shows what these multiprocessors look like.

Distributing the memory among the nodes has two major benefits. First, it is a cost-effective way to scale the memory bandwidth, if most of the accesses are to the local memory in the node. Second, it reduces the latency for accesses to the local memory. These two advantages make distributed memory attractive at smaller processor counts as processors get ever faster and require more memory bandwidth and lower memory latency. The key disadvantage for a distributed memory architecture is that communicating data between processors becomes somewhat more complex and has higher latency, at least when there is no contention, because the processors no longer share a single centralized memory. As we will see shortly, the use of distributed memory leads to two different paradigms for inter processor communication.



The basic architecture of a distributed-memory multiprocessor consists of individual nodes containing a processor, some memory, typically some I/O, and an interface to an interconnection network that connects all the nodes. Individual nodes may contain a small number of processors, which may be interconnected by a small bus or a different interconnection technology, which is less scalable than the global interconnection network.

Performance of Distributed Shared-Memory Multiprocessors

The performance of a directory-based multiprocessor depends on many of the same factors that influence the performance of bus-based multiprocessors (e.g., cache size, processor count, and block size), as well as the distribution of misses to various locations in the memory hierarchy. The location of a requested data item depends on both the initial allocation and the sharing patterns. We start by examining the basic cache performance of our scientific/technical workload and then look at the effect of different types of misses. Because the multiprocessor is larger and has longer latencies than our snooping- based multiprocessor, we begin with a slightly larger cache (128 KB) and a larger block size of 64 bytes.
In distributed memory architectures, the distribution of memory requests between local and remote is key to performance, because it affects both the consumption of global bandwidth and the latency seen by requests. Therefore, for the figures in this section we separate the cache misses into local and remote requests. In looking at the figures, keep in mind that, for these applications, most of the remote misses that arise are coherence misses, although some capacity misses can also be remote, and in some applications with poor data distribution, such misses can be significant

The miss rates with these cache sizes are not affected much by changes in processor count, with the exception of Ocean, where the miss rate rises at 64 processors. This rise results from two factors: an increase in mapping conflicts in the cache that occur when the grid becomes small, which
leads to a rise in local misses, and an increase in the number of the coherence misses, which are all remote.

The miss rates change as the cache size is increased, assuming a 64-processor execution and 64-byte blocks. These miss rates decrease at rates that we might expect, although the dampening effect caused by little or no reduction in coherence misses leads to a slower decrease in the remote misses than in the local misses. By the time we reach the largest cache size

shown, 512 KB, the remote miss rate is equal to or greater than the local miss rate. Larger caches would amplify this trend.
Because these applications have good spatial locality, increases in block size reduce the miss rate, even for large blocks, although the performance benefits for going to the largest blocks are small. Furthermore, most of the improvement in miss rate comes from a reduction in the local misses.