

Slides taken from

Parallel Computing Platforms

**Ananth Grama, Anshul Gupta,
George Karypis, and Vipin Kumar**

To accompany the text "Introduction to Parallel Computing",
Addison Wesley, 2003.

Limitations of Memory System Performance

- Memory system, and not processor speed, is often the bottleneck for many applications.
- Memory system performance is largely captured by two parameters, latency and bandwidth.
- Latency is the time from the issue of a memory request to the time the data is available at the processor.
- Bandwidth is the rate at which data can be pumped to the processor by the memory system.

Memory Latency: An Example

- Consider a processor operating at 1 GHz (1 ns clock) connected to a DRAM with a latency of 100 ns (no caches). Assume that the processor has two multiply-add units and is capable of executing four instructions in each cycle of 1 ns. The following observations follow:
 - The peak processor rating is 4 GFLOPS.
 - Since the memory latency is equal to 100 cycles and block size is one word, every time a memory request is made, the processor must wait 100 cycles before it can process the data.

Memory Latency: An Example

- On the above architecture, consider the problem of computing a dot-product of two vectors.
 - A dot-product computation performs one multiply-add on a single pair of vector elements, i.e., each floating point operation requires one data fetch.
 - It follows that the peak speed of this computation is limited to one floating point operation every 100 ns, or a speed of 10 MFLOPS, a very small fraction of the peak processor rating!

Improving Effective Memory Latency Using Caches

- Caches are small and fast memory elements between the processor and DRAM.
- This memory acts as a low-latency high-bandwidth storage.
- If a piece of data is repeatedly used, the effective latency of this memory system can be reduced by the cache.
- The fraction of data references satisfied by the cache is called the cache *hit ratio* of the computation on the system.
- Cache hit ratio achieved by a code on a memory system often determines its performance.

Explicitly Parallel Platforms

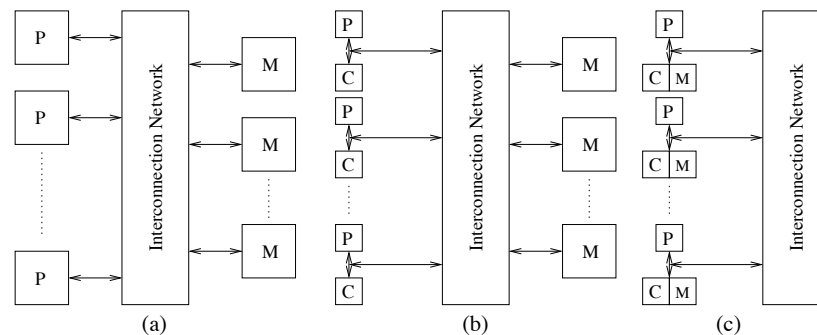
Communication Model of Parallel Platforms

- There are two primary forms of data exchange between parallel tasks – accessing a shared data space and exchanging messages.
- Platforms that provide a shared data space are called shared-address-space machines or multiprocessors.
- Platforms that support messaging are also called message passing platforms or multicomputers.

Shared-Address-Space Platforms

- Part (or all) of the memory is accessible to all processors.
- Processors interact by modifying data objects stored in this shared-address-space.
- If the time taken by a processor to access any memory word in the system global or local is identical, the platform is classified as a uniform memory access (UMA), else, a non-uniform memory access (NUMA) machine.

NUMA and UMA Shared-Address-Space Platforms



Typical shared-address-space architectures: (a) Uniform-memory access shared-address-space computer; (b) Uniform-memory-access shared-address-space computer with caches and memories; (c) Non-uniform-memory-access shared-address-space computer with local memory only.

NUMA and UMA Shared-Address-Space Platforms

- The distinction between NUMA and UMA platforms is important from the point of view of algorithm design. NUMA machines require locality from underlying algorithms for performance.
- Programming these platforms is easier since reads and writes are implicitly visible to other processors.
- However, read-write data to shared data must be coordinated (this will be discussed in greater detail when we talk about threads programming).
- Caches in such machines require coordinated access to multiple copies. This leads to the cache coherence problem.

Shared-Address-Space vs. Shared Memory Machines

- It is important to note the difference between the terms shared address space and shared memory.
- We refer to the former as a programming abstraction and to the latter as a physical machine attribute.
- It is possible to provide a shared address space using a physically distributed memory.

Message-Passing Platforms

- These platforms comprise of a set of processors and their own (exclusive) memory.
- Instances of such a view come naturally from clustered workstations and non-shared-address-space multicomputers.
- These platforms are programmed using (variants of) send and receive primitives.
- Libraries such as MPI provide such primitives.

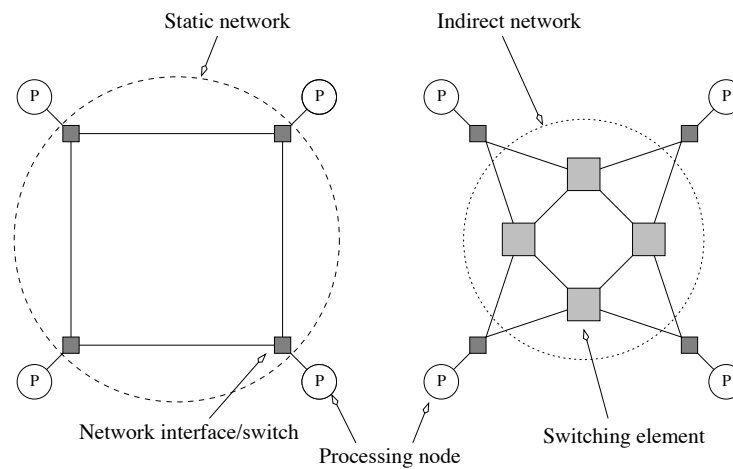
Message Passing vs. Shared Address Space Platforms

- Message passing requires little hardware support, other than a network.
- Shared address space platforms can easily emulate message passing. The reverse is more difficult to do (in an efficient manner).

Interconnection Networks for Parallel Computers

- Interconnection networks carry data between processors and to memory.
- Interconnects are made of switches and links (wires, fiber).
- Interconnects are classified as static or dynamic.
- Static networks consist of point-to-point communication links among processing nodes and are also referred to as *direct* networks.
- Dynamic networks are built using switches and communication links. Dynamic networks are also referred to as *indirect* networks.

Static and Dynamic Interconnection Networks



Classification of interconnection networks: (a) a static network; and (b) a dynamic network.

Design of parallel algorithms

Core content of this course:

- Take memory hierarchy into account (data locality)
- Distribute data over memories
- Distribute work over processors
- Introduce & analyse communication & synchronization

A first hands-on experience : do the exercise !