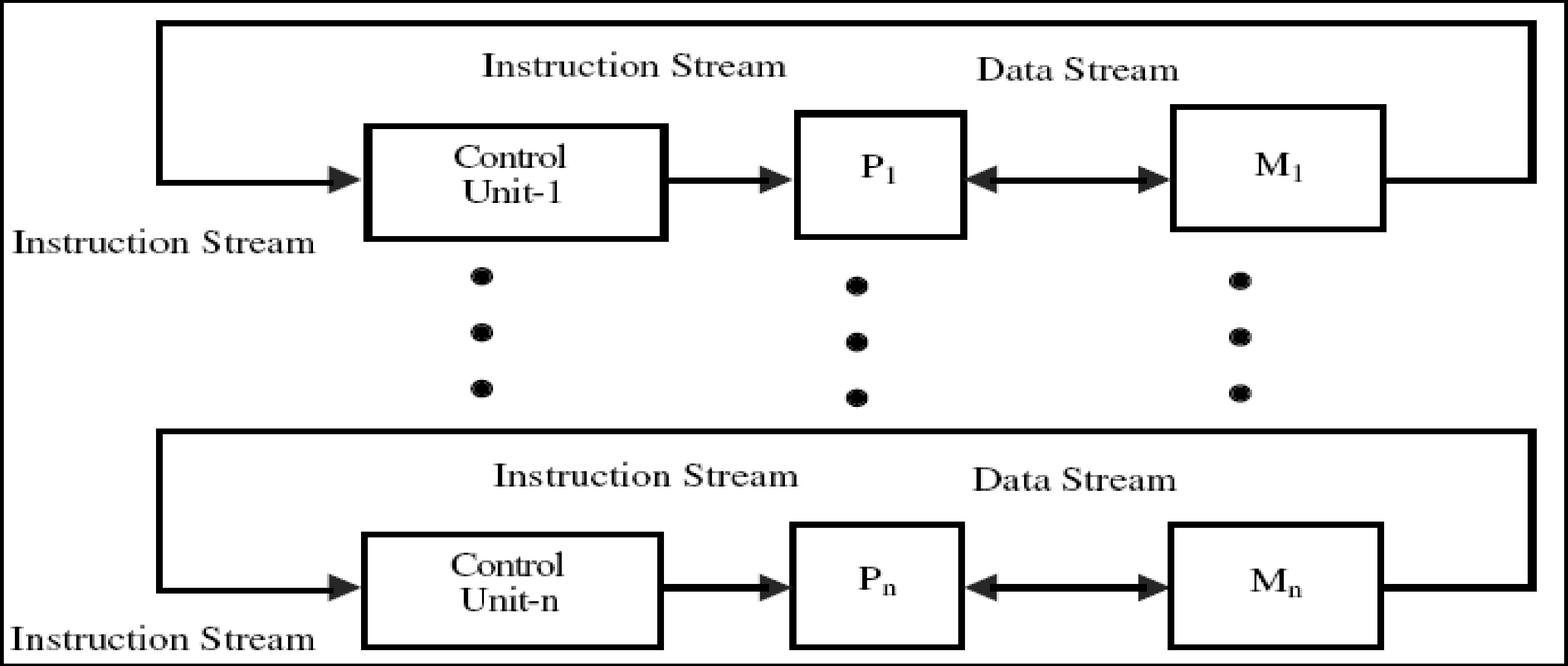
Thread Level Parallelism and Multiprocessors

4. *Multiple instruction streams, multiple data streams* (MIMD)

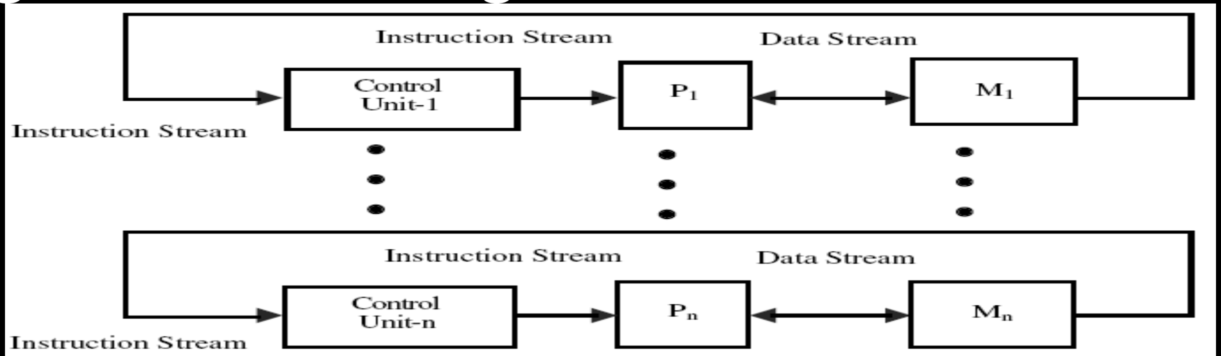


A Taxonomy of Parallel Architectures

4. *Multiple instruction streams, multiple data streams* (MIMD)—Each proc fetches its own instrns and operates on its own data.
MIMD is the archi of multiprocessors.

MIMDs offer flexibility. MIMDs can function as single-user multiprocessors focusing on high performance for one application.

With an MIMD, each processor executes a different process. It is also useful to be able to have multiple processors executing a single program and sharing the code.



A Taxonomy of Parallel Architectures (Contd..)

When multiple processes share code and data in this way, they are often called *threads*.

To take adv of an MIMD multiprocessor with *n* procs, we must usually have at least *n* threads or processes to execute.

Since the llelism in this situation is contained in the threads, it is called *thread-level parallelism*.

A Taxonomy of Parallel Architectures (Contd..)

MIMD multiprocessors fall into two classes,
 depending on the no. of processors, memory organization and IN strategy.
We refer multiprocessors by their mem org.
The first group, called ***as centralized shared-memory architectures,*** have at most a few dozen processors.

For multiprocessors with small processor counts, it is possible for the processors to share a single centralized memory and to interconnect the processors and memory by a bus.

A Taxonomy of Parallel Architectures (Contd..)

There is
-a single main memory that has a symmetric relationship to all processors and
a uniform access time from any processor,
these multiprocessors are often called *symmetric (shared-memory) multiprocessors* (*SMPs*), and
this style of architecture is sometimes called *UMA* for *uniform memory access*.
Below Fig shows how these multiprocessors look like.

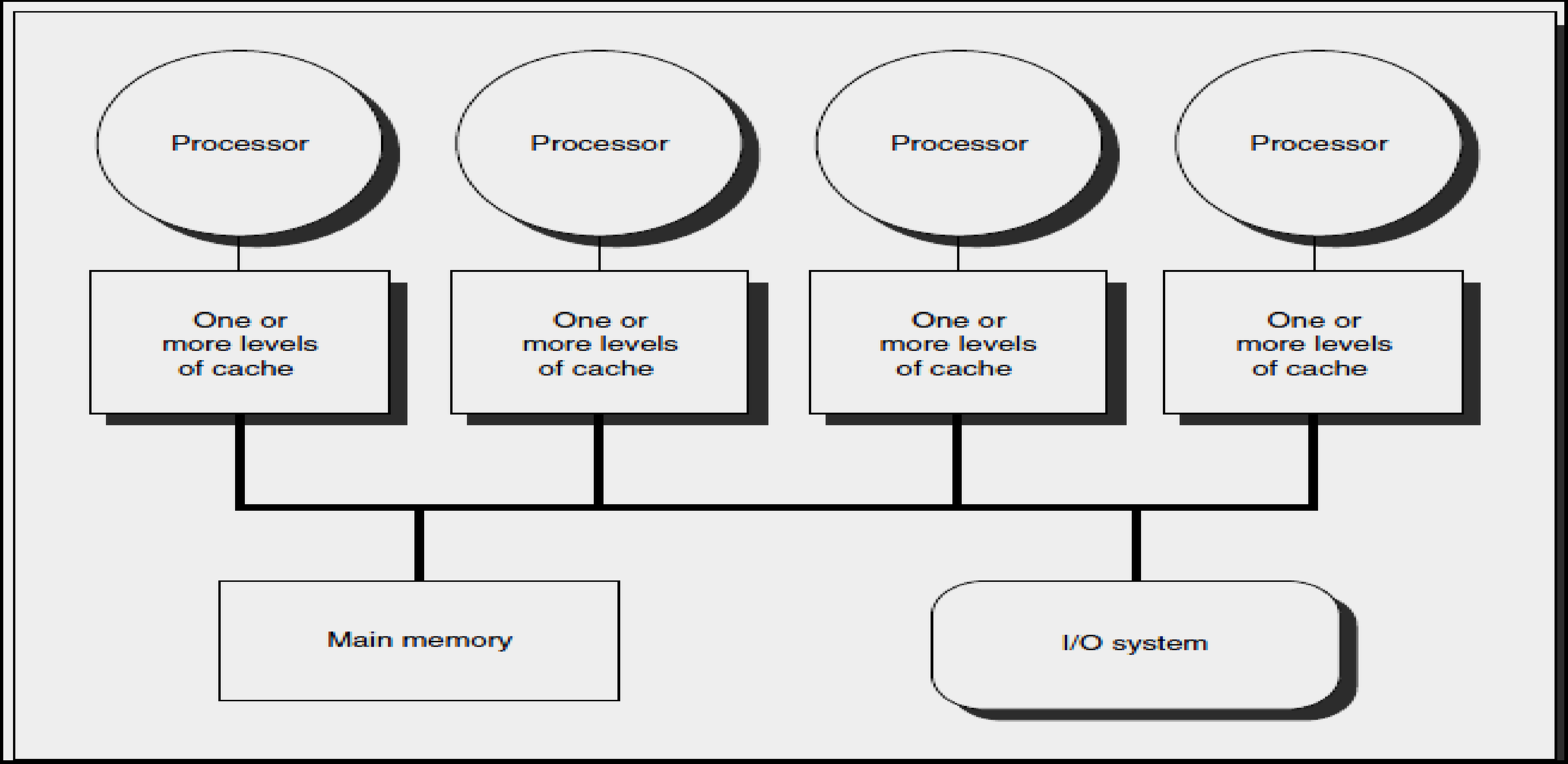

Processor
Processor
Processor
Processor
One or more levels of cache
One or more levels of cache
One or more levels of cache
One or more levels of cache
Main memory
I/O system

FIG. Basic structure of a centralized shared-memory multiprocessor.

Multiple processor-cache subsystems share the same physical memory, connected by a bus.

In larger designs, multiple buses, / a switch may be used, but uniform access time to all memory from all processors remains constant.

The centralised memory system would not be able to support the BW demands of a larger number of processors
i.e it causes excessively long access latency.

The 2nd group ..multiprocessors with physically distri mem. (**Distributed memory architecture).**

Here memory must be distri among the procs rather than centralized to support larger processor counts.

With the rapid increase in processor perfo & increase in the processor-memory BW requirements, distri memory is preferred over a single, centralized memory.
Of course, the larger number of processors raises the need for a high BW interconnect.
Below Fig shows this type of multiprocessors.

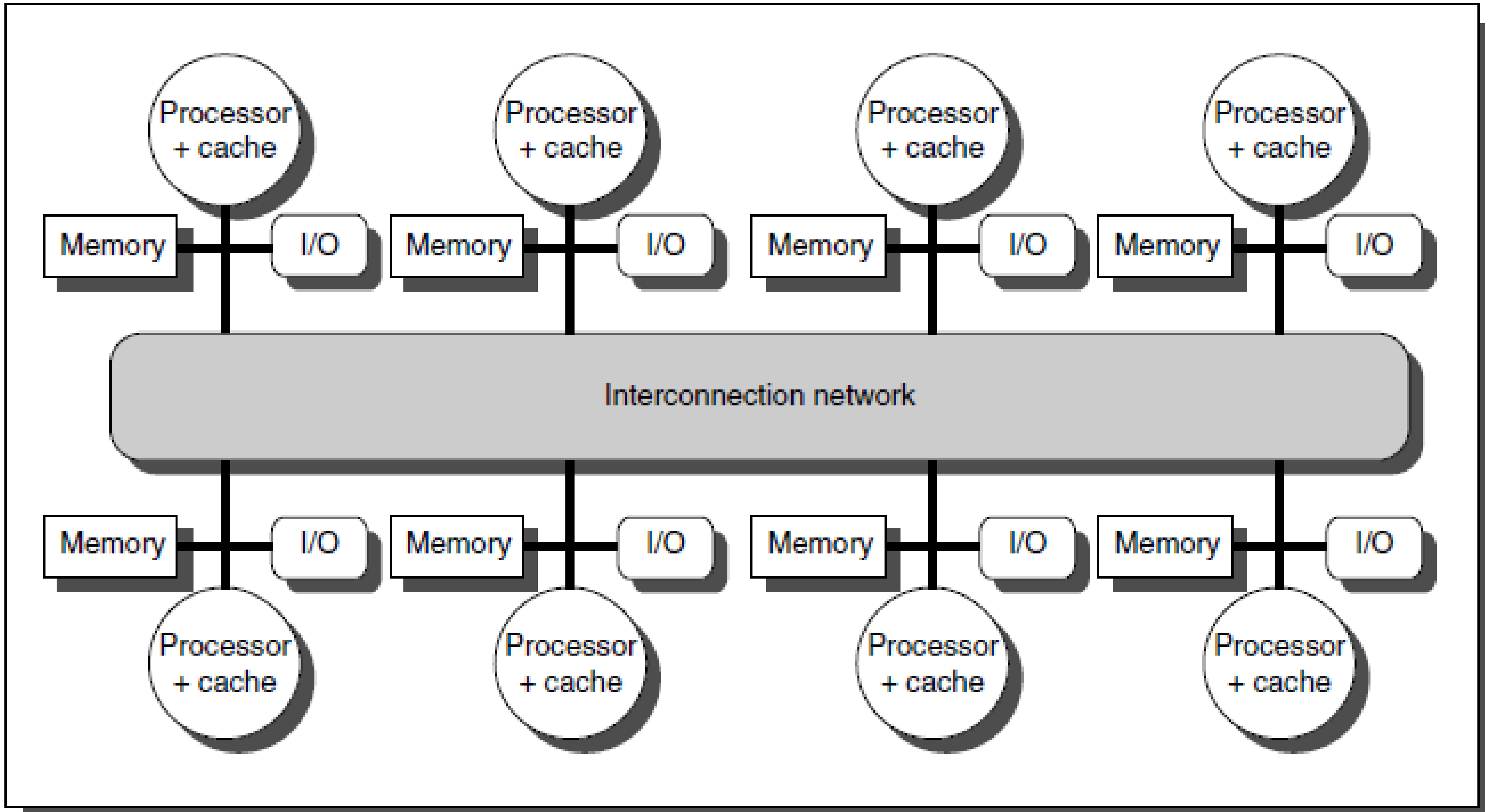
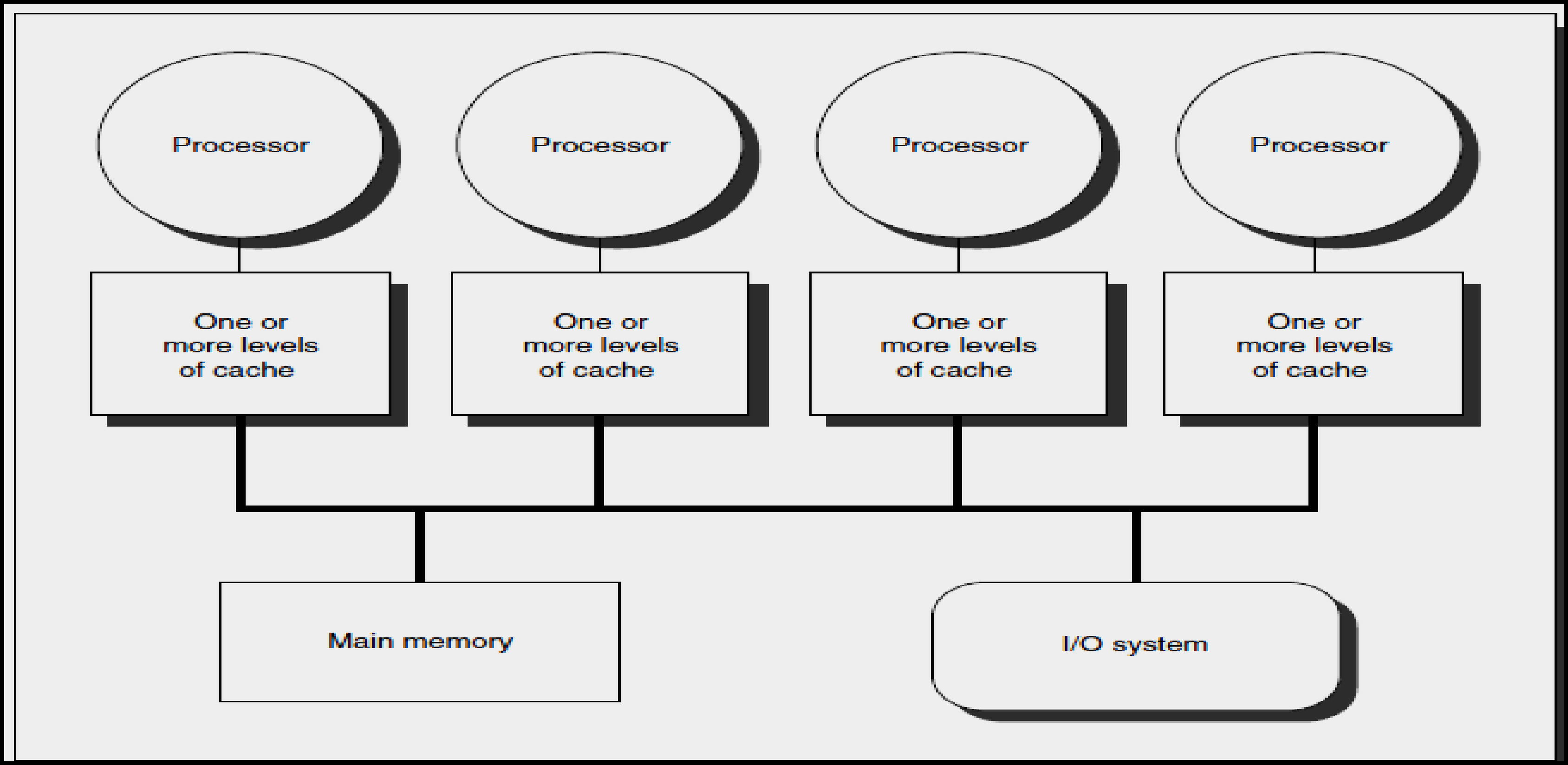


FIG. The basic architecture of a distri-mem multiprocessor. Consists of individual nodes containing a processor, memory, I/O, and an i/f to an IN connecting all the nodes.

Individual nodes may contain a small number of procs, which may be interconnected through bus or through different IN technology.

Distributing the mem among the nodes has two major benefits:
First, it is good to scale the mem BW, if most of the accesses are to the local mem in the node.
Second, it reduces the latency for accesses to the local memory.
With this it makes distri mem to be attractive at smaller proc counts as procs get ever faster access to mem and lower mem latency.
The disadv for a distri mem archi is that data communication between processors becomes more complex b’ze the procs no longer share a single centralized memory.

Symmetric Shared-Memory Architectures


Processor
Processor
Processor
Processor
One or more levels of cache
One or more levels of cache
One or more levels of cache
One or more levels of cache
Main memory
I/O system

Symmetric Shared-Memory Architectures:

Small-scale shared-memory machines usually support the caching of both shared and private data.

Private data is used by a single processor,

shared data is used by multiple processors

providing comm among the processors through reads and writes of the shared data.

When a private item is cached, its location is migrated to cache, and hence reducing the avg access time.

Since no other processor uses the private data, the pgm behavior is identical to that in a uniprocessor.

When shared data are cached, the shared value may have to be replicated in multiple caches.

Caching of shared data, however, introduces a new problem: Cache Coherence.

Centralized shared-memory multiprocessor

Distributed memory multiprocessor