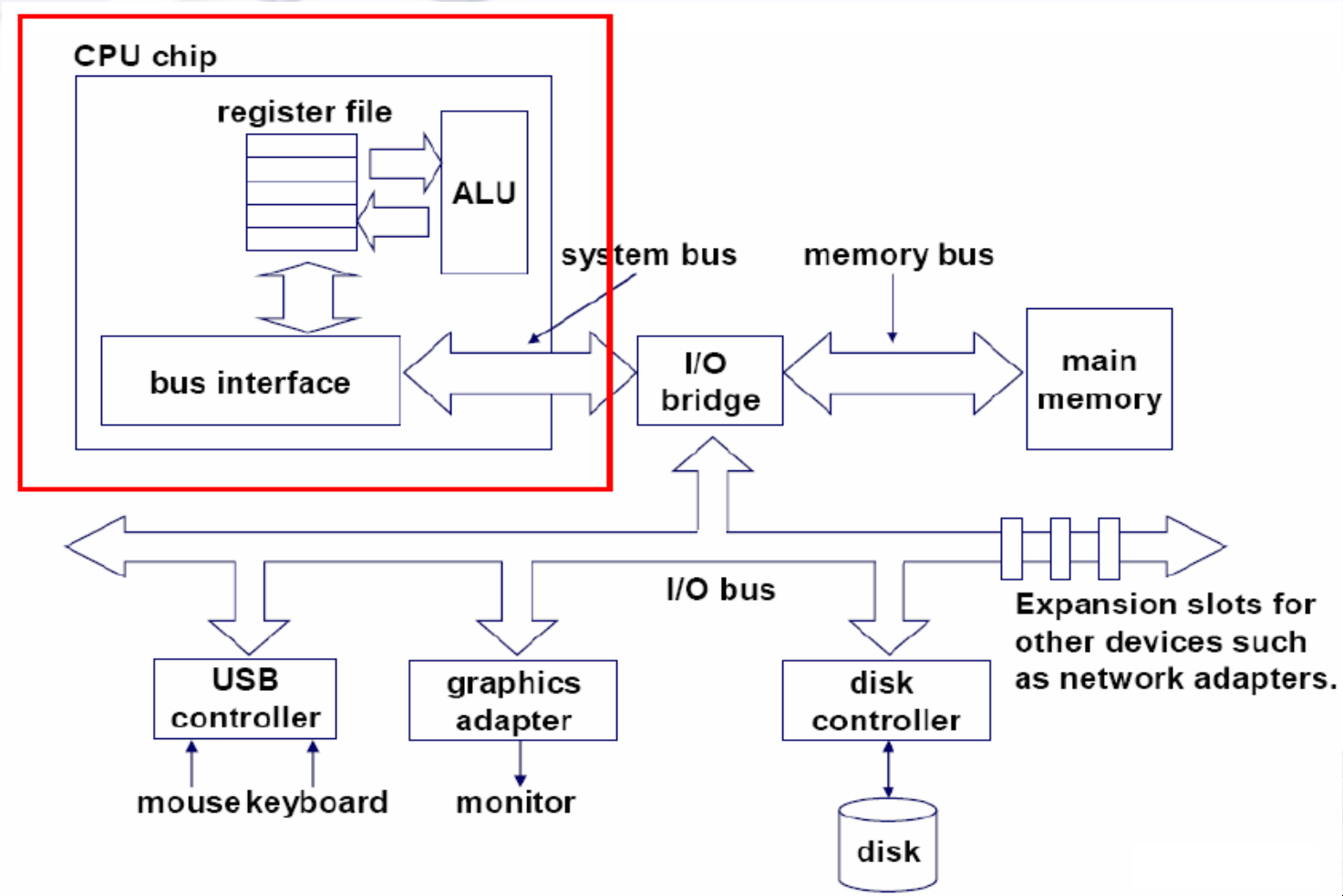
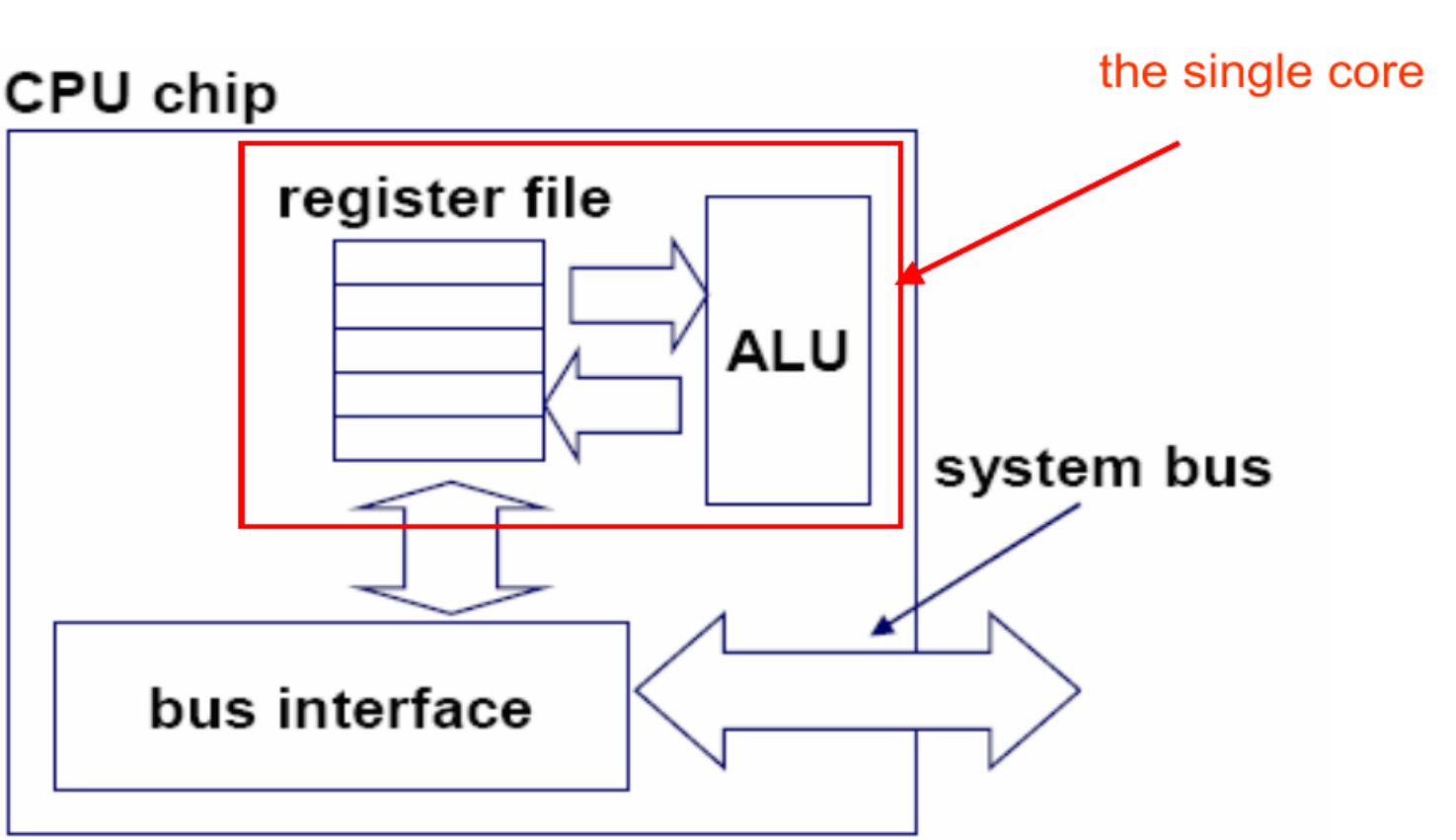
TLP: Thread Level Parallelism

Tassadaq Hussain
Riphah International University
Barcelona Supercomputing Center
Universitat Politècnica de Catalunya

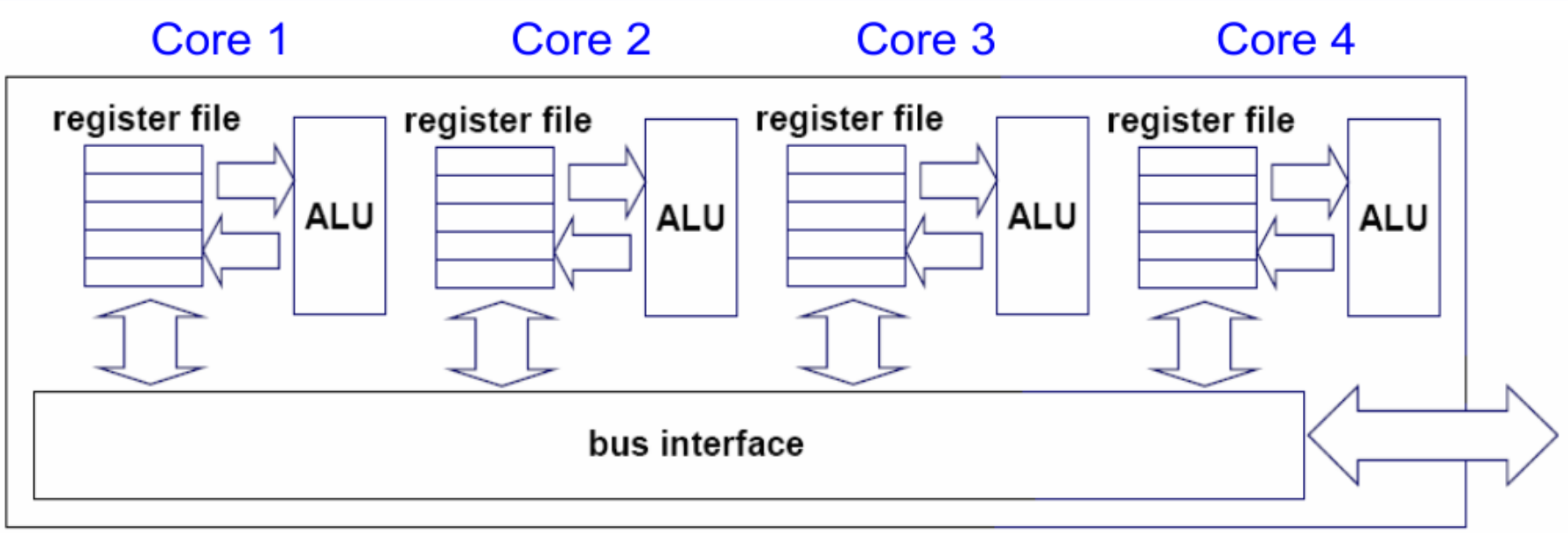
Uni Processor Architecture



Uni Core

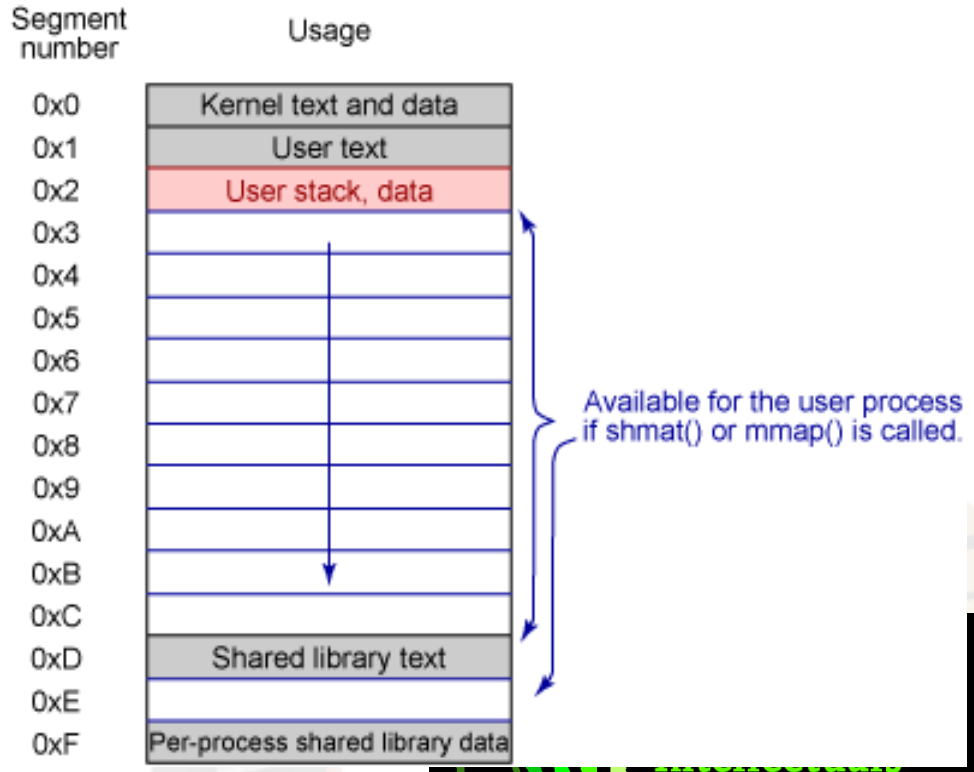


Multi-core Processor



Address Space

An address space is a range of valid addresses in memory that are available for a program or process. That is, it is the memory that a program or process can access. The memory can be either physical or virtual and is used for executing instructions and storing data.



Type of Architectures

Single instruction stream and single data stream - uniprocessor system

Single instruction stream and multiple data stream.

Each processing element has its own data memory but there is single instruction memory and a control processor.

Vector processor architectures belongs to this class.

Multiple instruction stream and single data stream.

There is no commercial multiprocessor of this type.

Each processor has its own instruction stream and data stream.

Each processor may be an off-the-shelf microprocessor.

MIMD: The Architecture of Choice

MIMD is **flexible** and with the help of appropriate **hardware** and **software**, it can be used as a **high performance computation platform** for different applications.

MIMD system can be manufactured using off-the-shelf components (microprocessors).

TLP: Thread Level Parallelism

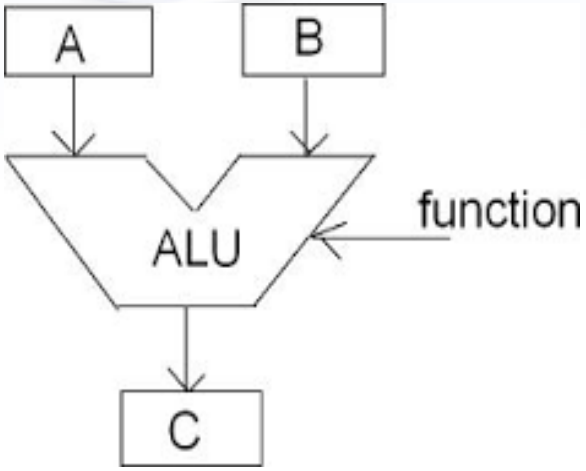
- A process is a segment of code that may be run independently;
- the state of the process contains all the information necessary to execute that program on a processor.
- each process is typically independent of other processes.
- A thread uses multiple processes which share code and data.

Thread Level Parallelism

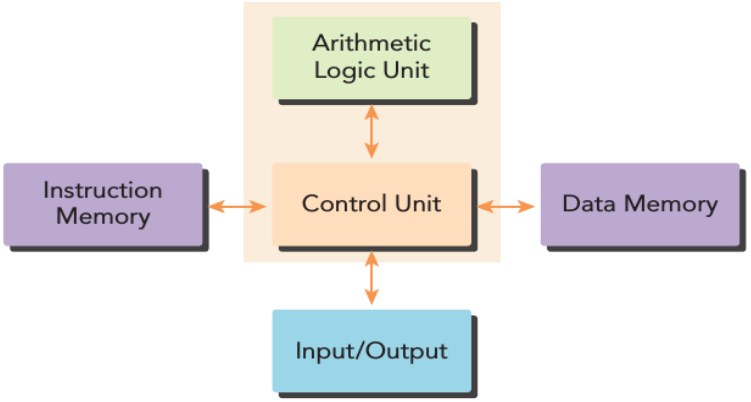
Each processor executes independent processes, or communicating processes or different threads of a process.

The threads or processes may be created by the programmer e.g. by fork() call or by the compiler e.g. parallel iterations of a loop.

Basic introduction of Microprocessor



Function	Inputs A	Input B	Output
0	A	B	A+B
1	A	B	A-B



Types of MIMD Architecture

Centralized shared-memory architectures or symmetric shared-memory multiprocrssors (SMP) or uniform memory access (UMA) architectures.

Distributed-memory multiprocessors.

SMP: Shared Memory Processor

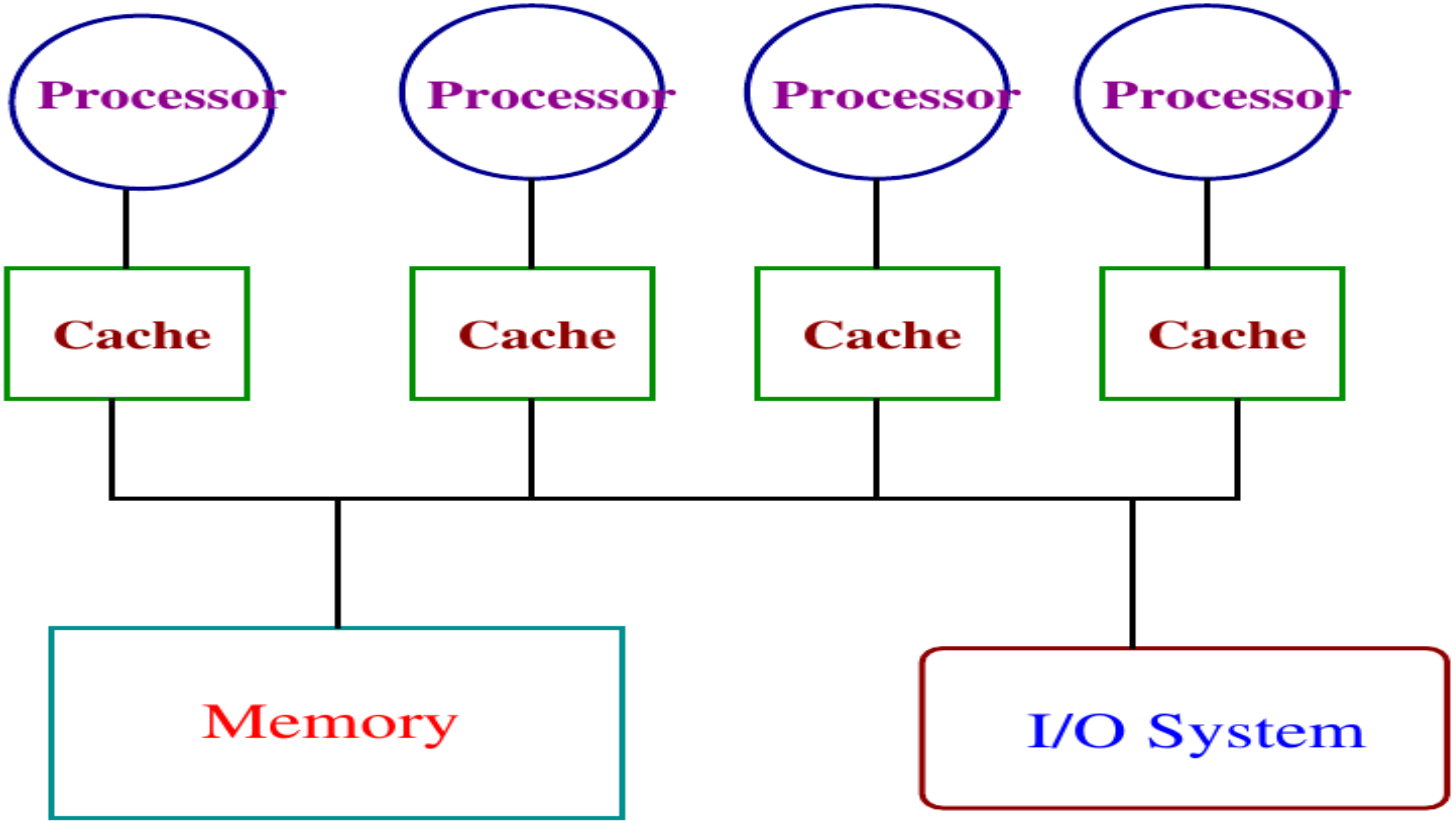
Small number of similar processors (at most a few dozen).

Each processor has a large cache.

A centralized memory (multiple banks) is shared through a memory bus.

Each memory location has identical access time from each processor.

SMP

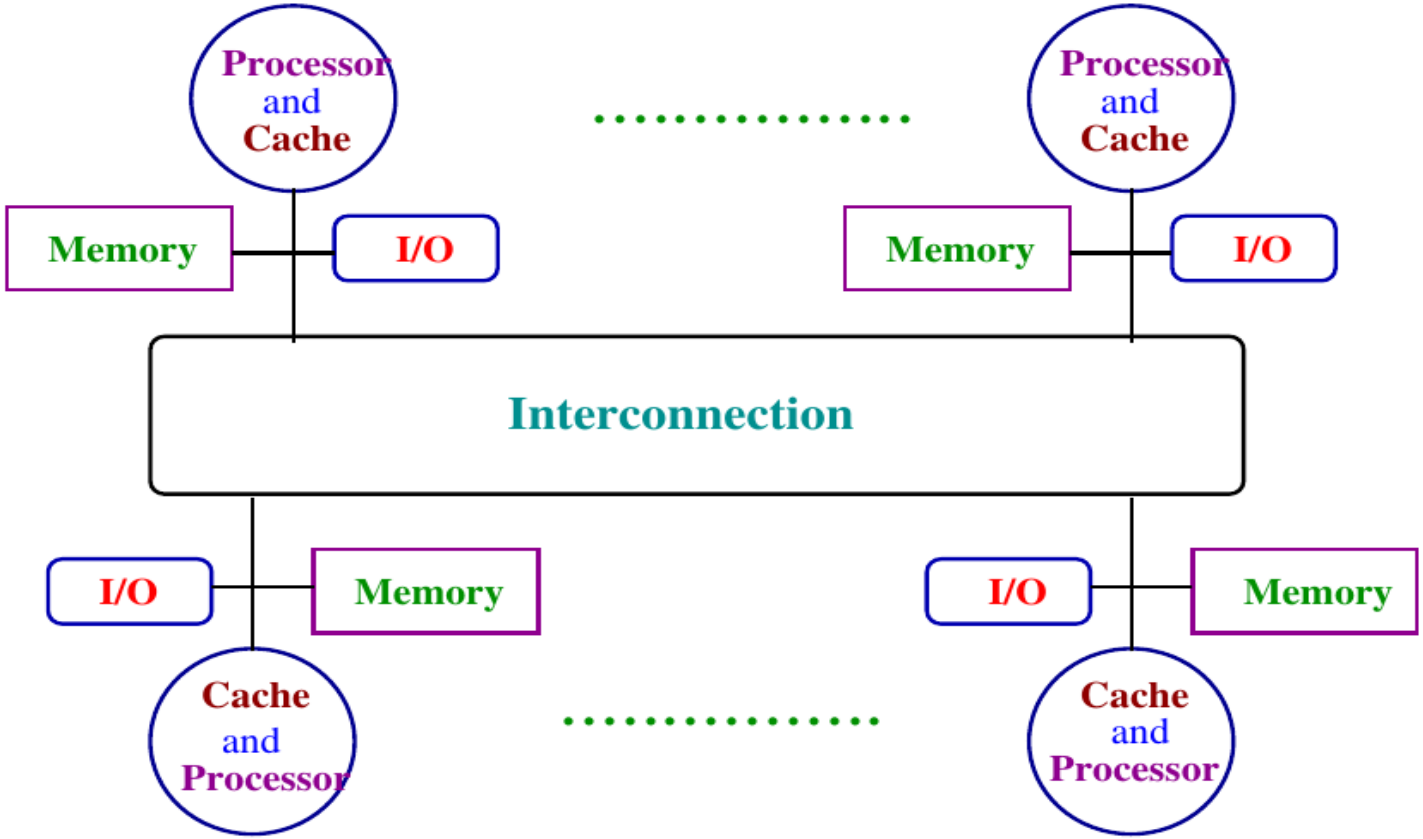


Distributed Memory

Larger processor count.

Memory is physically distributed among the processors for better bandwidth.

Connected through high-speed interconnection e.g. switches.



Distributed Memory

The bandwidth for the local memory is high and the latency is low.

But access to data present in the local memory of some other processor is complex and of high latency.

Memory Architecture

Large-scale multiprocessors have physically distributed memory with the processors.

There are essentially two different models of memory architectures and the corresponding models of communication.

Memory Arch..

- Memory is distributed with different processors to support higher bandwidth demand of larger number of processors.
- Any processor can access a location of physically distributed memory (with proper access permission).
- This is called distributed shared-memory architecture (DSM) also known as NUMA (nonuniform memory access).

Shared Memory Architecture

The same physical address on two processors refers to the same location in memory.

The communication is through the shared address space.

The other alternative is that every processor has its private address space.

Each processor is essentially a separate computer - this is called a multicomputer model or clusters.

The communication is by message-passing.

Such cluster of processors use standardized or customized interconnect for communication

Message Passing: Memory Sharing

If a processor wants to access (or process) some data in a remote memory, it sends a message (similar to remote procedure call (RPC)).

The destination processor receives it (polling or interrupt), performs the operation, and returns the result through a reply message.

The message passing is synchronous - the initiating processor after sending the requests waits for the reply.

Amdahl's Law

Amdahl's law gives the theoretical speedup in latency of the execution of a task at fixed workload that can be expected of a system whose resources are improved.

$$\text{Speedup} = \frac{1}{\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} + (1 - \text{Fraction}_{\text{enhanced}})}$$

Suppose you want to achieve a speedup of 80 with 100 processors. What fraction of the original computation can be sequential?

$$\text{Speedup} = \frac{1}{\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} + (1 - \text{Fraction}_{\text{enhanced}})}$$

$$80 = \frac{1}{\frac{\text{Fraction}_{\text{parallel}}}{100} + (1 - \text{Fraction}_{\text{parallel}})}$$

Suppose we have an application running on a 32 processor multiprocessor, which has a 200 ns time to handle reference to a remote memory. For this application, assume that all the references except those involving communication hit in the local memory hierarchy, which is slightly optimistic. Processors are stalled on a remote request, and the processor clock rate is 2 GHz. If the base CPI (assuming that all references hit in the cache) is 0.5, how much faster is the multiprocessor if there is no communication versus if 0.2% of the instructions involve a remote communication reference?

$$\text{CPI} = \text{Base CPI} + \text{Remote request rate} \times \text{Remote request cost}$$

$$\text{Remote request cost} = \frac{\text{Remote access cost}}{\text{Cycle time}}$$

Coherence

- Since we have private caches: How to keep the data consistent across caches?
- Each core should perceive the memory as a monolithic array, shared by all the cores.