Parallel Computers Chapter 1

(short form)

Parallel Computing

- Using more than one computer, or a computer with more than one processor, to solve a problem. Basic ideas around for 50 years!

Motives

- Usually faster computation - very simple idea - that *n* computers operating simultaneously can achieve the result *n* times faster - it will not be *n* times faster for various reasons.
- Other motives include: fault tolerance, larger amount of memory available, ...

Speedup Factor

$$S(p) = \frac{\text{Execution time using one processor (best sequential algorithm)}}{\text{Execution time using a multiprocessor with } p \text{ processors}} = \frac{t_s}{t_p}$$

where t_s is execution time on a single processor and t_p is execution time on a multiprocessor.

$S(p)$ gives increase in speed by using multiprocessor.

Use best sequential algorithm with single processor system. Underlying algorithm for parallel implementation might be (and is usually) different.

Speedup factor can also be cast in terms of computational steps:

$$S(p) = \frac{\text{Number of computational steps using one processor}}{\text{Number of parallel computational steps with } p \text{ processors}}$$

Can also extend time complexity to parallel computations.

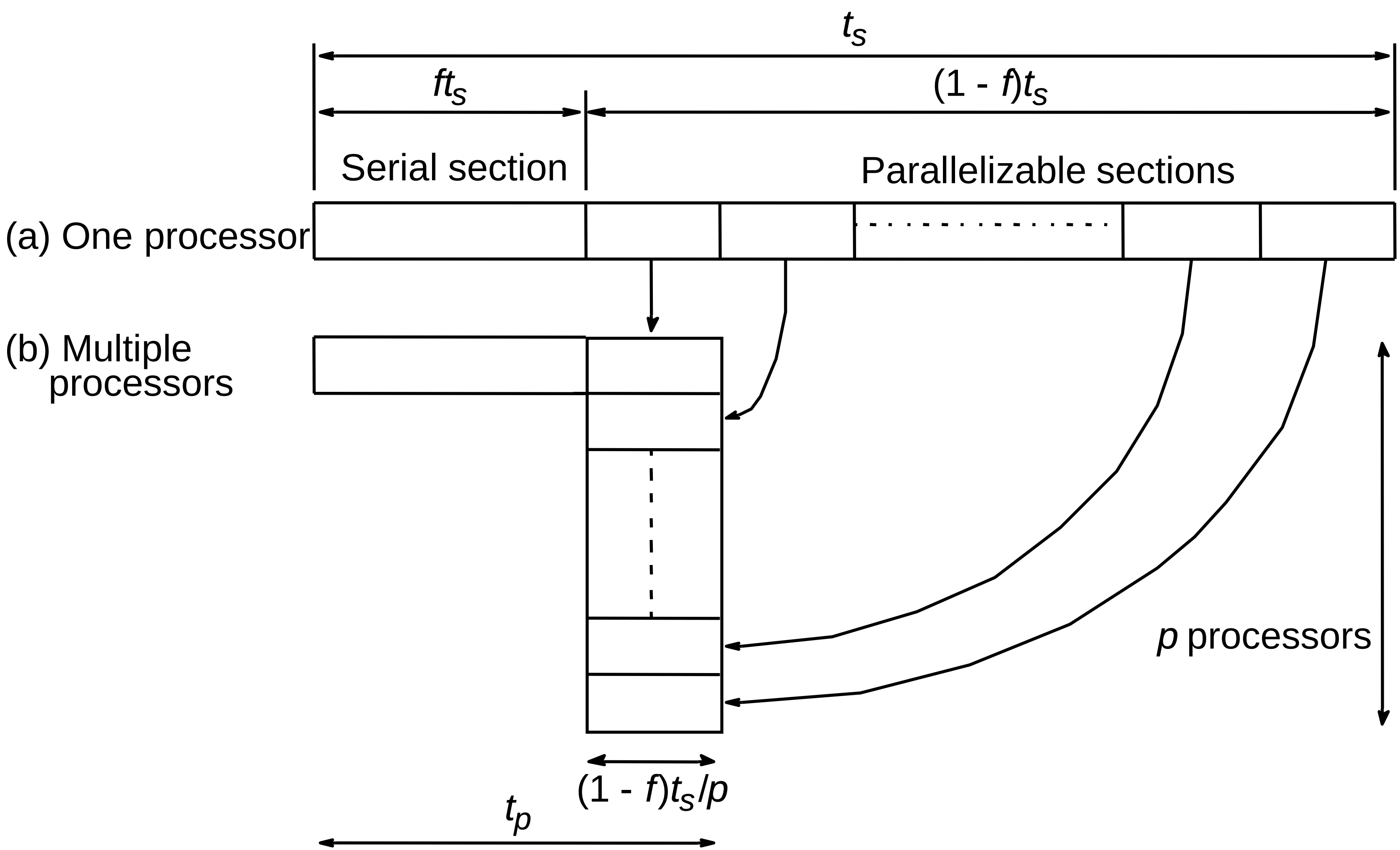
Maximum Speedup

Maximum speedup is usually *p* with *p* processors (linear speedup).

Possible to get superlinear speedup (greater than *p*) but usually a specific reason such as:

- Extra memory in multiprocessor system
- Nondeterministic algorithm

Maximum Speedup
Amdahl's law



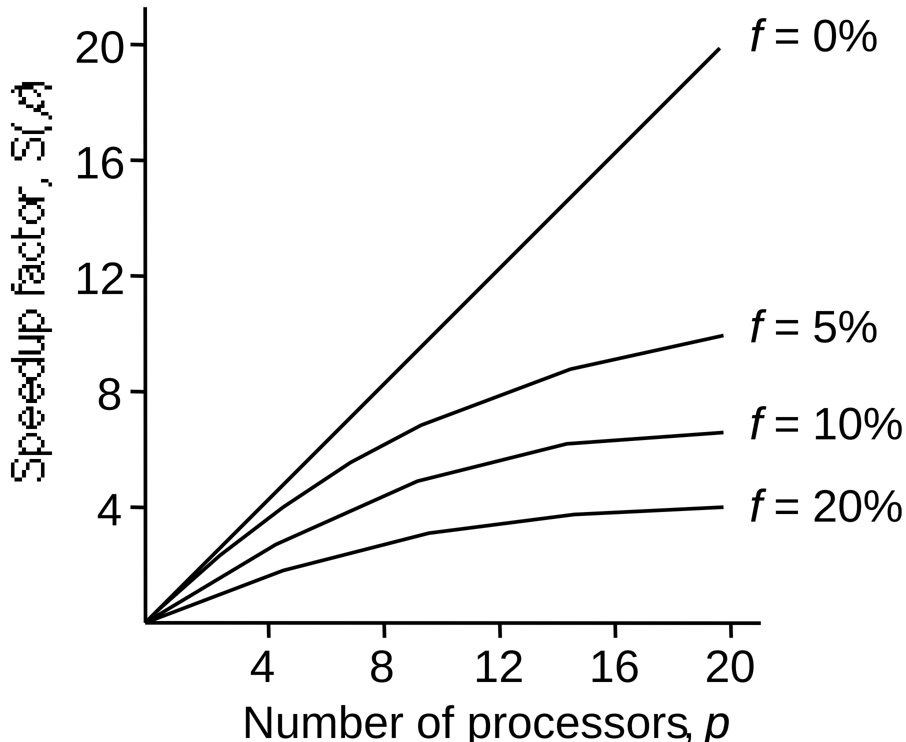
Speedup factor is given by:

$$S(p) = \frac{t_s}{ft_s + (1 - f)t_s/p} = \frac{p}{1 + (p - 1)f}$$

This equation is known as Amdahl's law

Speedup against number of processors

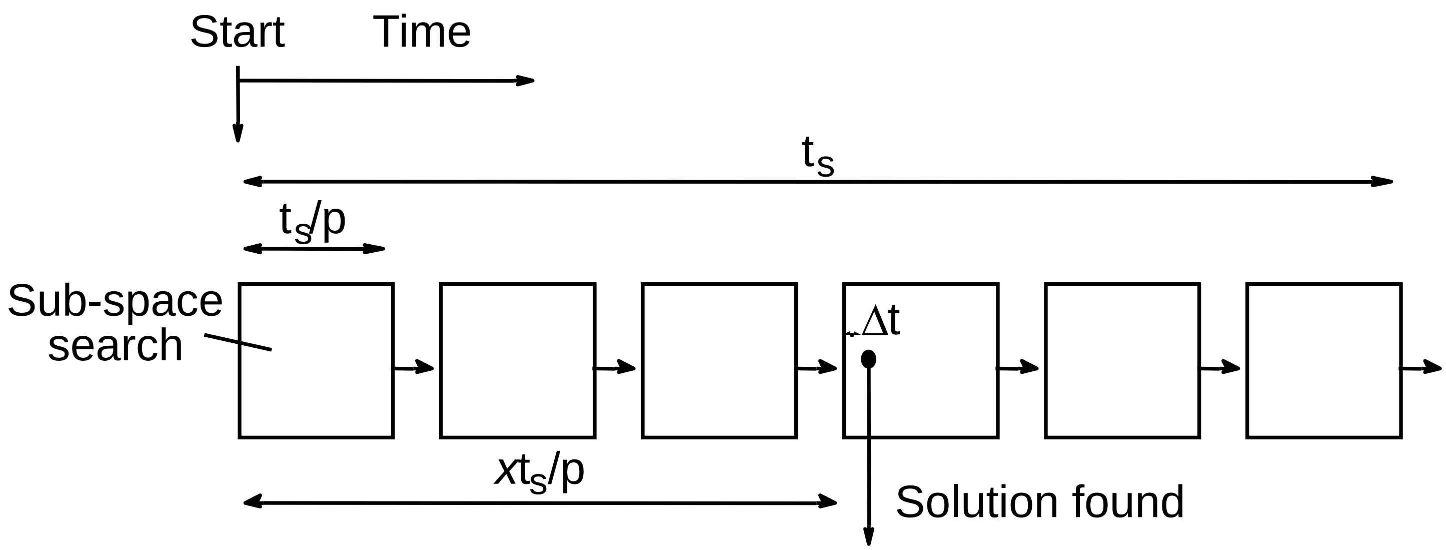
Even with infinite number of processors, maximum speedup limited to $1/f$.



Example with only 5% of computation being serial, maximum speedup is 20, irrespective of number of processors.

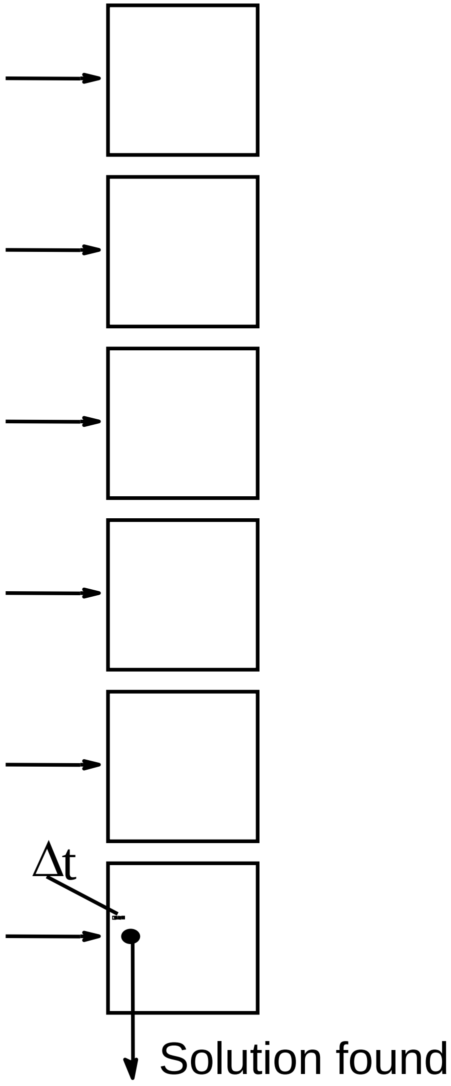
Superlinear Speedup example
- Searching

(a) Searching each sub-space sequentially



x indeterminate

(b) Searching each sub-space in parallel



Speed-up then given by

$$S(p) = \frac{\left(x \times \frac{t_s}{p}\right) + \Delta t}{\Delta t}$$

Worst case for sequential search when solution found in last sub-space search. Then parallel version offers greatest benefit, i.e.

$$S(p) = \frac{\left(\frac{p-1}{p}\right) \times t_s + \Delta t}{\Delta t} \rightarrow \infty$$

as Δt tends to zero

Least advantage for parallel version when solution found in first sub-space search of the sequential search, i.e.

$$S(p) = \frac{\Delta t}{\Delta t} = 1$$

Actual speed-up depends upon which subspace holds solution but could be extremely large.

Types of Parallel Computers

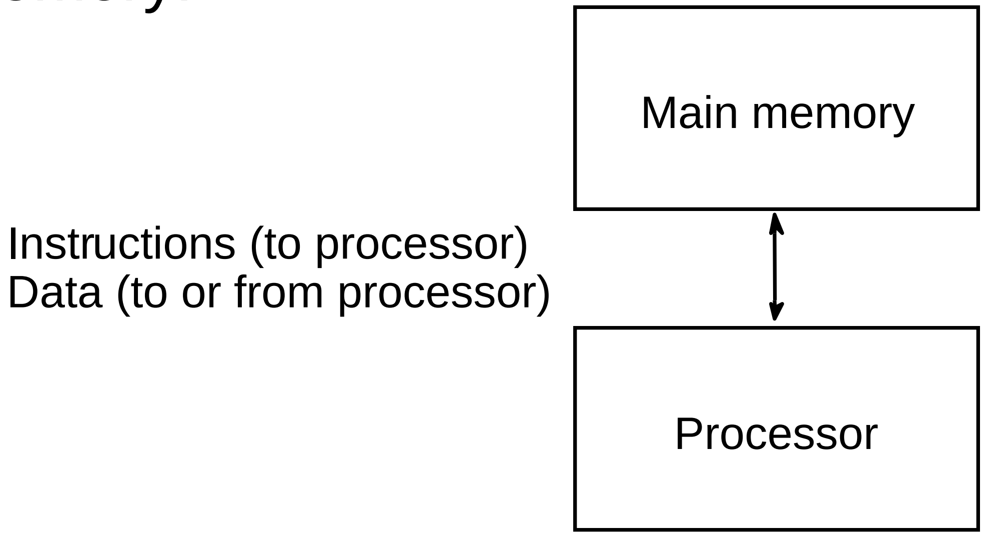
Two principal types:

- Shared memory multiprocessor (now often may be thought of as a single node in …
- Distributed memory multicomputer

Shared Memory Multiprocessor

Conventional Computer

Consists of a processor executing a program stored in a (main) memory:



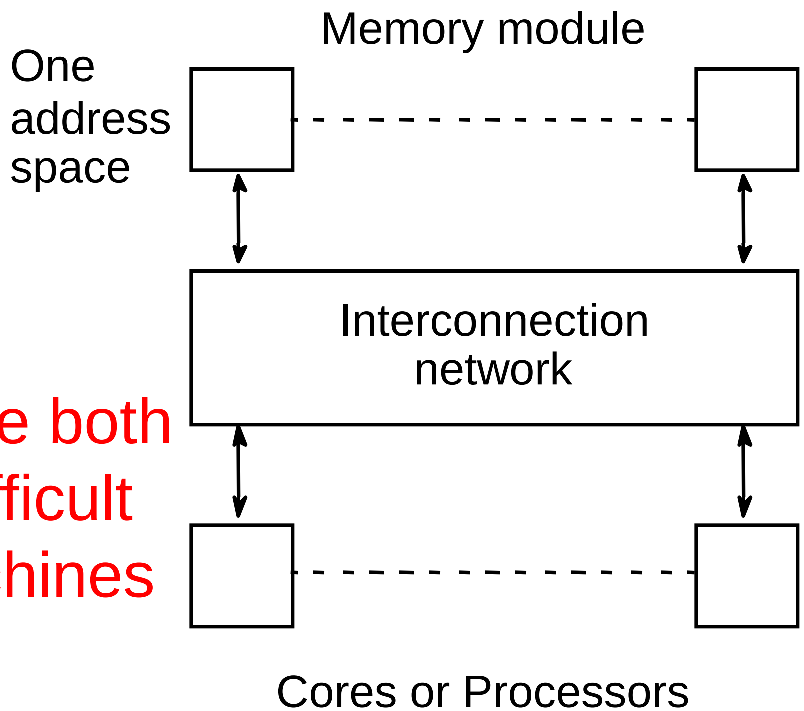
Each main memory location located by its address. Addresses start at 0 and extend to $2^b - 1$ when there are *b* bits (binary digits) in

Shared Memory Multiprocessor or Multicore System

Natural way to extend single processor model - have multiple processors connected to multiple memory modules, such that each processor can access any memory module :

Often this is now a single node in a large machine.

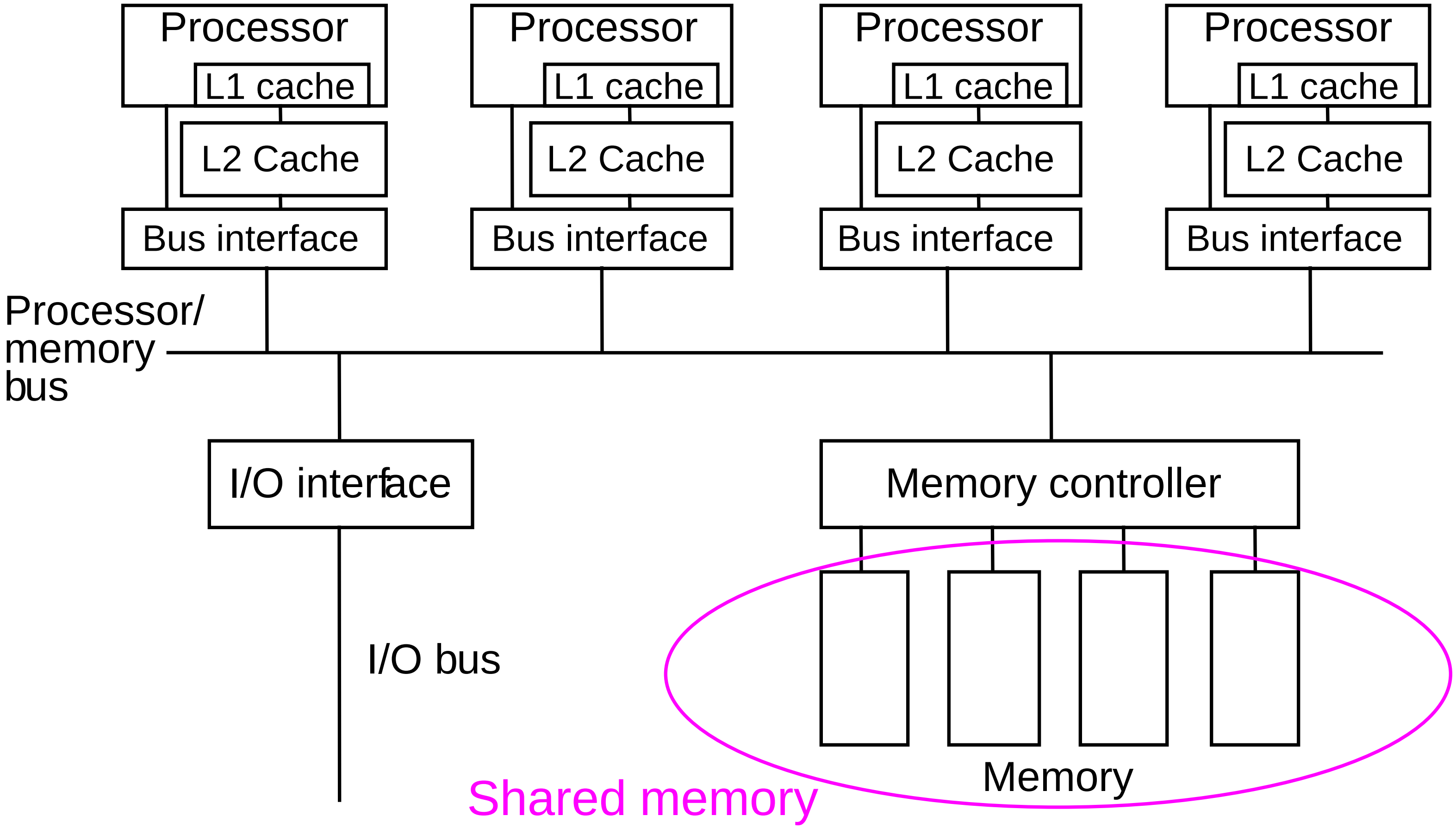
It is now otherwise both expensive and difficult to build such machines



Almost Every processor has more than one core

Old shared memory systems have a complex communications backplane connecting core on different boards

Old Quad Pentium Shared Memory Multiprocessor



Programming Shared Memory Multiprocessors

- Threads - programmer decomposes program into individual parallel sequences, (threads), each being able to access variables declared outside threads.

    Example Pthreads

- Sequential programming language with preprocessor compiler directives to declare shared variables and specify parallelism.

    Example OpenMP - industry standard - needs OpenMP compiler

- Sequential programming language with added syntax to declare shared variables and specify parallelism.

  Example UPC (Unified Parallel C) - needs a UPC compiler.

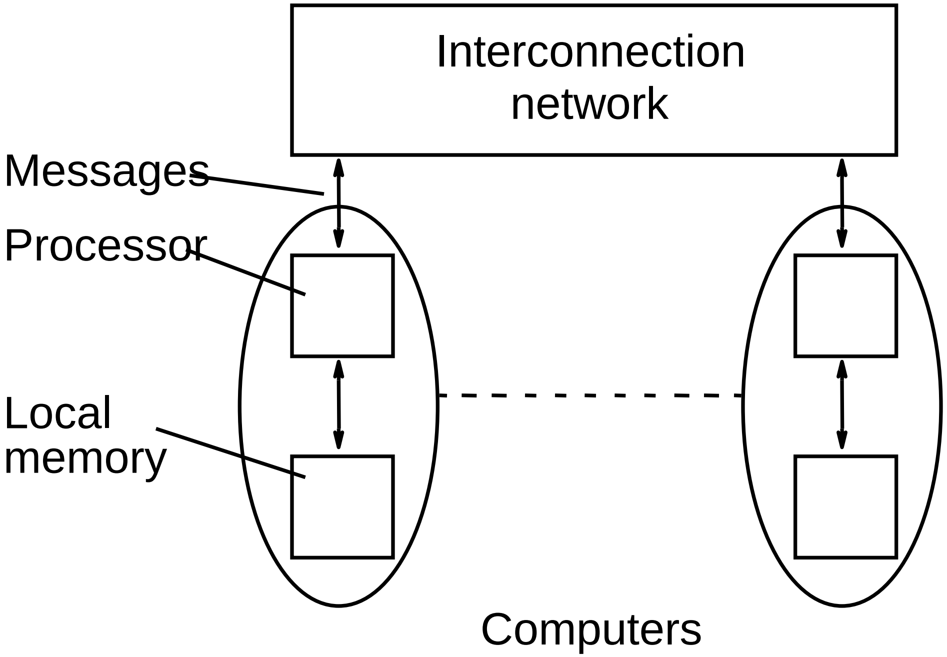
- Parallel programming language with syntax to express parallelism - compiler creates executable code for each processor (not now common)

- Sequential programming language and ask parallelizing compiler to convert it into parallel executable code. - also not now common

Message-Passing Multicomputer

Complete computers connected through an interconnection network:

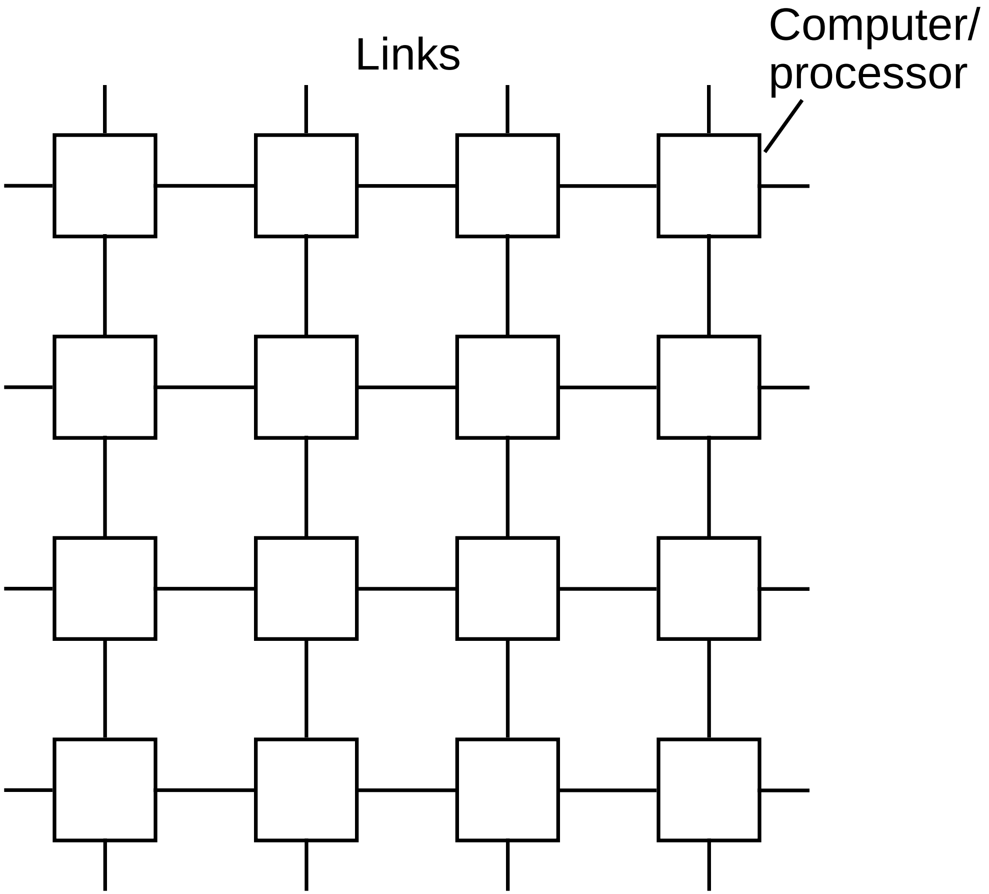
Often accelerators are now attached to these nodes



Interconnection Networks

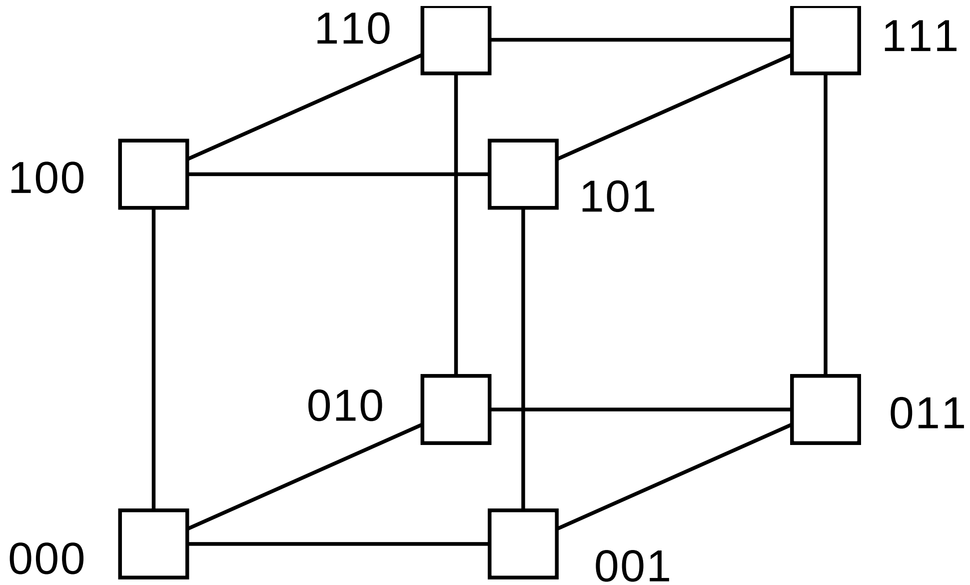
- Limited and exhaustive interconnections
- 2- and 3-dimensional meshes
- Hypercube (not now common)
- Using Switches:
  – Crossbar
  – Trees
  – Multistage interconnection networks

Two-dimensional array (mesh)

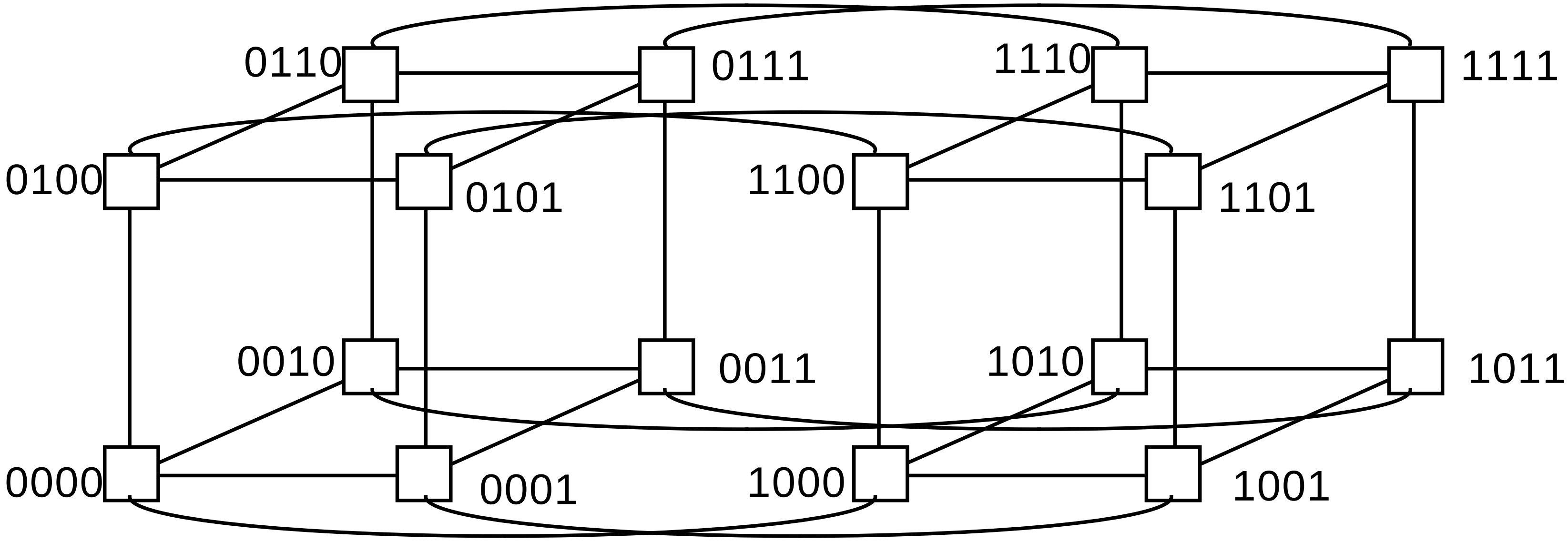


Also three-dimensional - used in some large high performance systems.

Three-dimensional hypercube

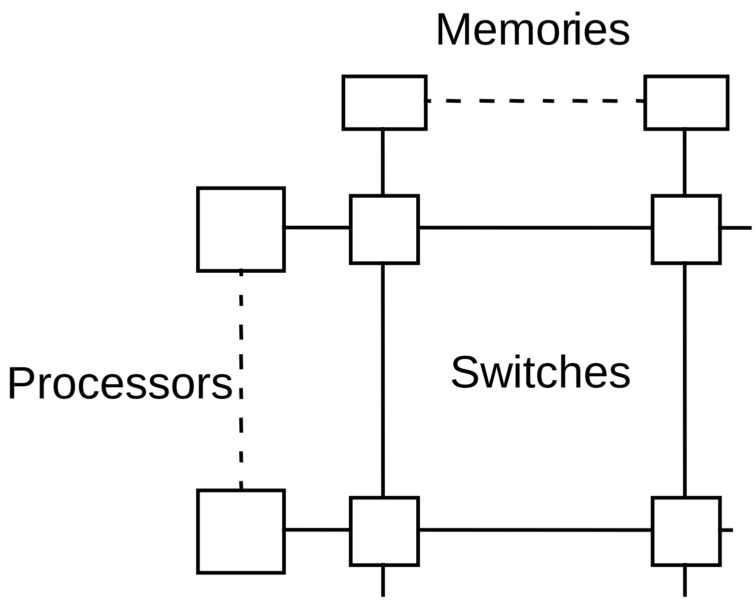


Four-dimensional hypercube

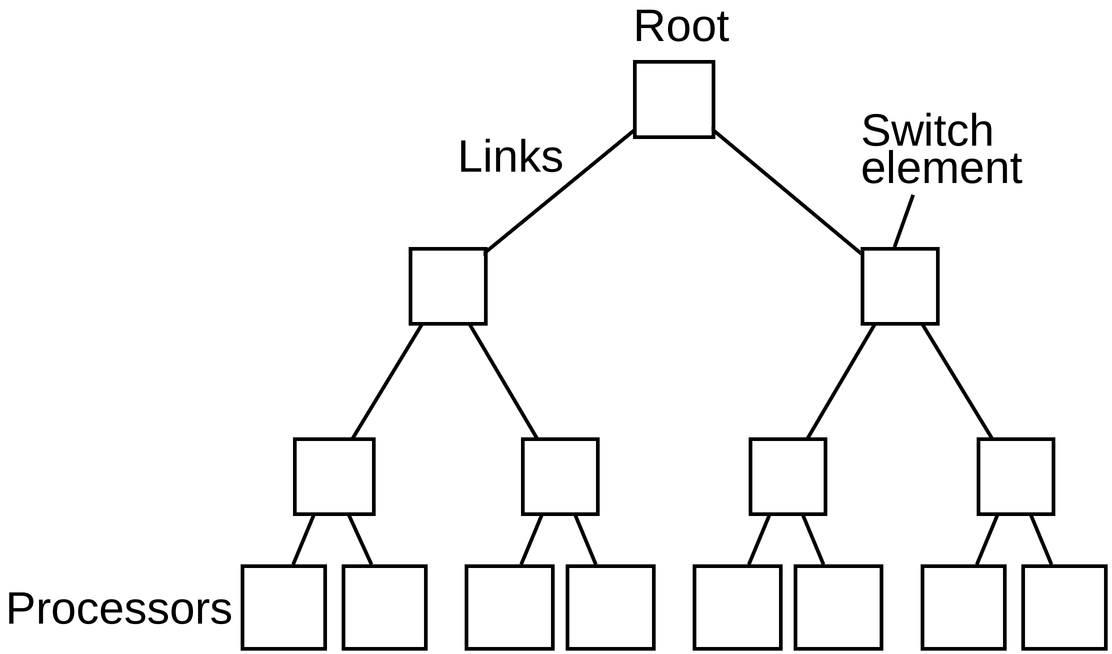


Hypercubes popular in 1980's - not now

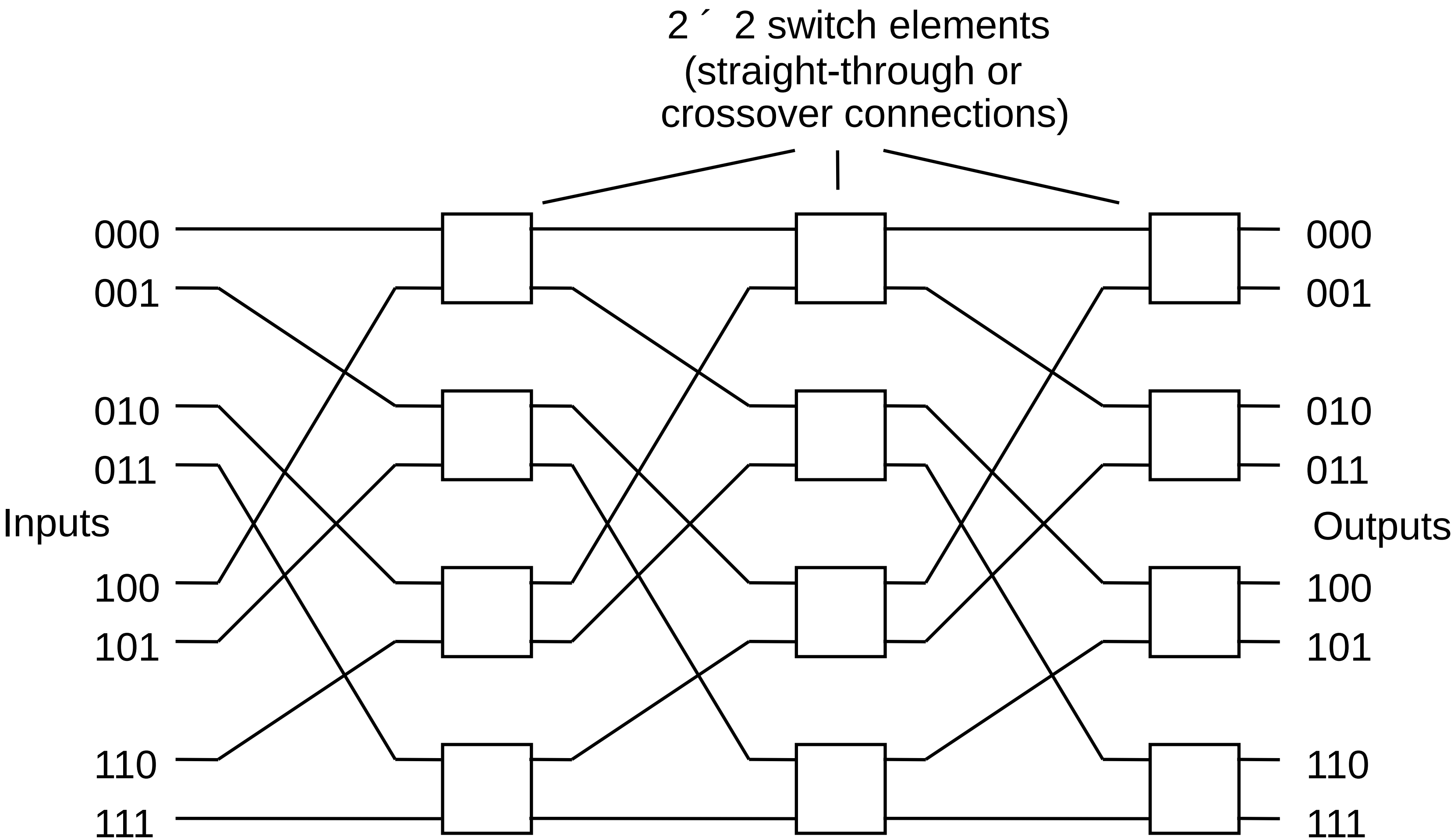
Crossbar switch



Tree



Multistage Interconnection Network
Example: Omega network



Distributed Shared Memory

Making main memory of group of interconnected computers look as though a single memory with single address space. Then can use shared memory programming techniques.

This is the approach used in Unified Parallel C