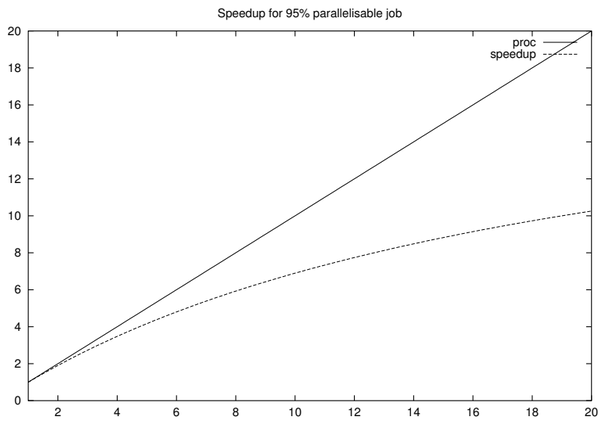
Parallel Computing

Used when jobs are to big for a single processor. A simple analysis can be done with Amdahl's Law.

$$S = \frac{t_{seq}}{t_{par}} = \frac{(1 - F_p)t_{seq} + F_p t_{seq}}{(1 - F_p)t_{seq} + \frac{F_p t_{seq}}{N_p}}$$

For a maximum speed up of $\frac{1}{(1-F_p)+\frac{F_p}{N_p}}$, where $F_p =$ amount of the job that is (perfectly) parallelisable and $N_p =$ number of processors.



Good news:

- Bigger problems often have large F_p.
- A lot of time spent in loops, which are often good candidates for parallelisation.

Bad news:

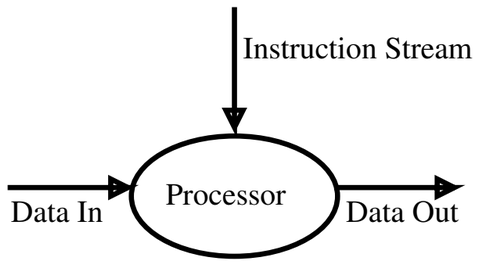
- Not always perfectly parallelisable.
- Overhead can be high (communication & synchronisation).
- Parallelising can be hard work for the programmer.

Types of Parallel Computer

Various attempts have been made to classify parallel computers:

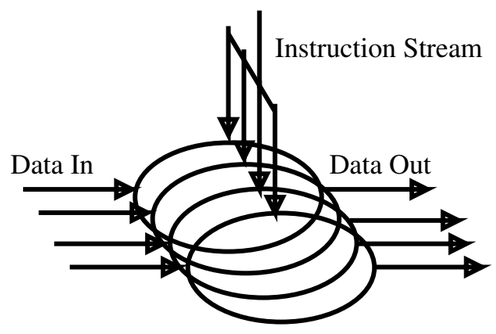
- SIMD, MIMD, MISD computers.
- Vector computers.
- Shared or Distributed memory.
- network topology used.

Single Instruction, Single Data



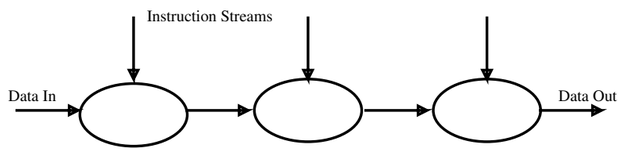
Conventional computer.

Single Instruction, Multiple Data



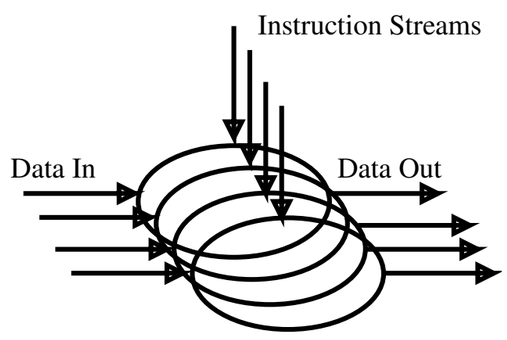
Eg. CM-2 (65535 proc) or MasPar (16384).

Multiple Instruction, Single Data



Uncommon, used in image/signal processing.

Multiple Instruction, Multiple Data



Eg. Network of Workstations, IBM SP, CM-5.

Vector Processor

Type of SIMD machine? Based on idea that operations like: $\vec{w} = \vec{u} + a\vec{v}$ and $a = \vec{u} \cdot \vec{v}$ are common.

Vector co-processor with vector registers (64-128 doubles per reg). Performs operation on whole vector register in one instruction.

Compiler replaces loops with vector ops.

for(i = 0 ; i $<$ n ; i++)

    w[i] = i[i] + a $*$ v[i];

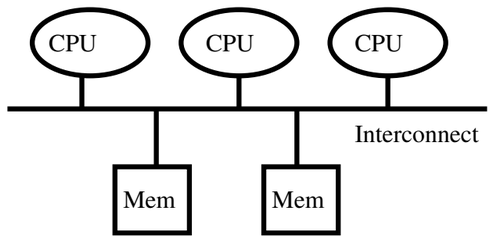
Vector computers need to get data quickly and will have to use interleaving or other techniques to get data fast enough.

Eg. Cray T90, NEC SX-4, Fujitsu VPP700, Hatachi S3800.

Shared/Distributed Memory

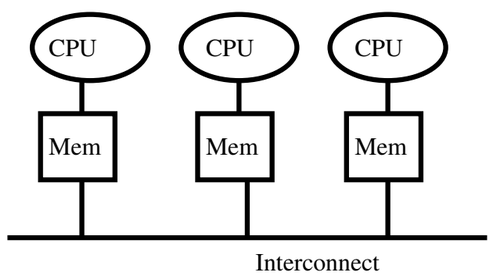
Are changes to data visable implicitly?

Shared Mem:



Easier to program, but CPU caches must be keep syncronised.

Distributed Mem:



Data must be passed explicitly.

SMP

A Symetric Multi-Processor computer has several CPUs connected to a single memory. Each CPU may have its own cache. All CPUs are equal in the eyes of the kernel.

Competition on the bus prevents huge numbers of processors but 64 processor machines are not uncommon (Sun & SGI).

Compilers are available which can automatically parallelise code for SMP machines. Threads also work well on SMP machines.

Locking of data is an important issue for the kernel and application on SMP machines.

NOW

A network of workstations is a Distributed Memory system.

IBM SP A collection of IBM 1 or n processor workstations connected by a special high speed switch.

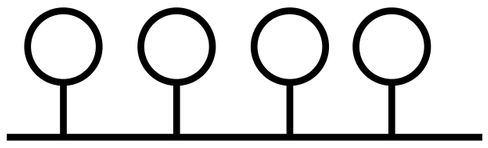
Linux Beowulf Patches for Linux to make a cluster of PCs seem more like one machine. Uses ethernet as interconnect.

distributed.net Many different computers around the world using the internet as interconnect.

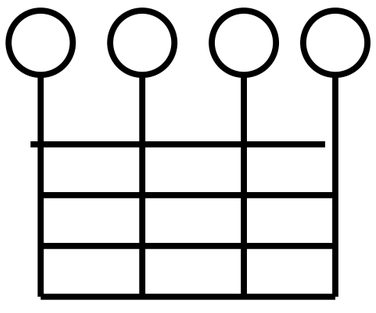
NOW is popular and (possibly) cheap high power computer system. Programming is more complicated than SMP.

Interconnect

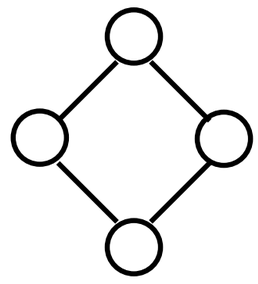
Bus One pair can communicate at a time. Complexity independant of N_p, between node bandwidth $1/N_p$.



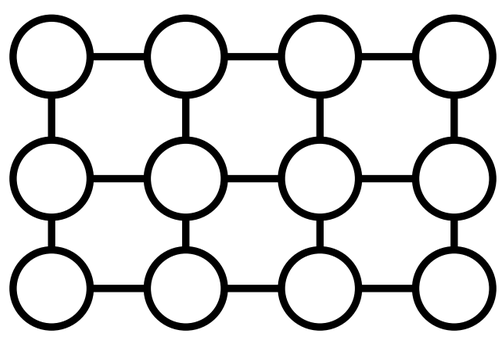
Cross-bar All processors linked with matrix of switches. Requires $O(p^2)$ switches, between node bandwidth of $1/N_p$.



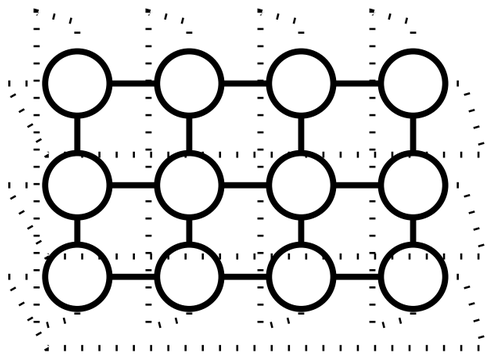
Ring Each processor connected to 2 nearest neighbours. Requires $p/2$ hops at worst.



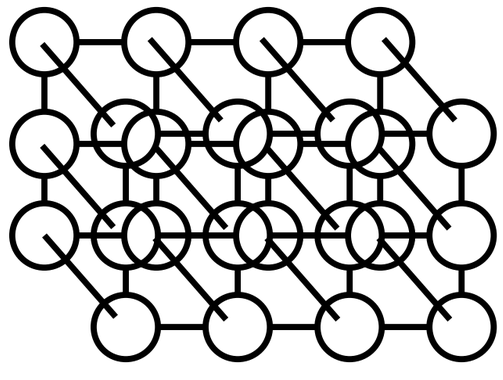
2-d mesh Each processor connected to 4 nieghbours. Requires about $\sqrt{(p)}$ hops at worst.



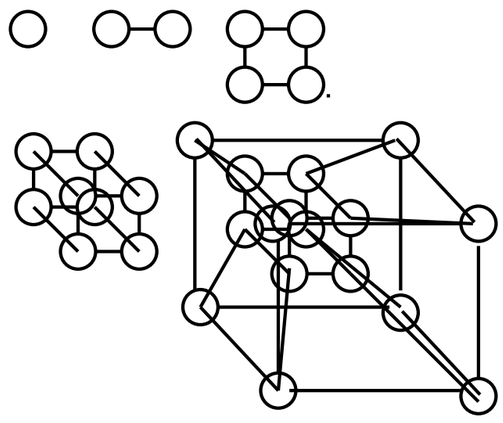
2-d torus Each processor connected to 4 nieghbours. Requires about $\sqrt(p)/2$ hops at worst.



3-d mesh/tours Each processor connected to 4 nieghbours. Requires about $\sqrt[3]{(p)}$ hops.



Cube/Hypercube 2^d processors each with d connections. Gets complicated quickly.



Other Some systems use networks of switches (Omega Netowrk - IBM SP) or irregular netowrks (distributed.net, NOW). Others again use combinations of the above ('hyper-cross bar', 3d mesh with crossbar along edges).

Assignment

Various libraries have been developed which make programming parallel computers easier. PVM and MPI are two such libraries. PVM (Parallel Virtual Machine) grew and MPI (Message Passing Interface) was designed. We will be using MPI.

Write a program calculates $\sum_{i=1}^{n} \frac{1}{n}$ in parallel. It should take n as an argument on the command line, eg: `recipsum 100000`. Try summing smallest to largest and see if you get a different answer.

MPI Docs:
`http://www.netlib.org/mpi`