Top500 Supercomputer list

- Tends to represent parallel computers, so distributed systems such as SETI@Home are neglected.
- Does not consider storage or I/O issues
- Both custom designed machines and commodity machines win positions in the list
- General trend towards commodity machines (COTS – Commodity-Off-The-Shelf).
- Connecting a large number of machines with relatively lower performance is more rewarding than connecting a small number of machines each with high performance
  - reference paper: “A note on the Zipf distribution of Top500 supercomputers” (download from my homepage)
- Performance doubles each year, better than Moore’s Law.
  - Moore’s Law: performance doubles approximately every 18 months
- Dominated by the United States

UK supercomputers in the latest list (announced on 11/2009)
- Edinburgh: 20
- ECMWF: 33 and 34 (last year 22)
- Southampton: 74
- AWE: 172 (last year: 77)
BlueGene/L (No. 4 in the Latest Top500 List)

- first supercomputer in the Blue Gene project
- architecture.
  - Individual PowerPC 440 processors at 700Mhz
  - Two processors reside in a single chip.
  - Two chips reside on a “compute card” with 512MB memory.
  - 16 of these compute cards are placed on a node board.
  - 32 node boards fit into one cabinet, and there are 64 cabinets.
  - 212992 CPUs with theoretical peak of 596.4 TFLOPS
  - Multiple network topologies available, which can be selected depending on the application.
- High density of processors in a small area:
  - comparatively slow processors - just lots of them!
  - Fast interconnects and low-latency.
Architecture Classifications

A taxonomy of parallel architectures: in 1972, Flynn categorised HPC architectures into four classes, based on how many instruction and data streams can be observed in the architecture.

They are:

- **SISD - Single Instruction, Single Data**
  - Instructions are operated sequentially on a single stream of data in a single memory.
  - Classic “Von Neumann” architecture.
  - Machines may still consist of multiple processors, operating on independent data - these can be considered as multiple SISD systems.

- **SIMD - Single Instruction, Multiple Data**
  - A single instruction stream (broadcast to all PE*s), acting on multiple data.
  - The most common form of this architecture class are Vector processors.
  - These can deliver results several times faster than scalar processors.

*PE = Processing Element*
Architecture Classifications

- **MISD - Multiple instruction, Single data**
  - No practical implementations of this architecture.

- **MIMD - Multiple instruction, Multiple data**
  - Independent instruction streams, acting on different (but related) data
  - Note the difference between multiple SISD and MIMD
Architecture Classifications

MIMD: MPP, Cluster, SMP and NUMA

SISD: Machine with a single scalar processor

SIMD: Machine with vector processors
Parallelism in single processor systems

Pipelines

- Performing more operations per clock cycle (Reduces the idle time of hardware components).
- Difficult to keep pipelines full (Good performance with independent instructions)
- Branch prediction helps

Vector architectures

- One master processor and multiple math co-processors
- Large memory bandwidth and low latency access.
- No cache because of above.
- Perform operations involving large matrices, commonly encountered in engineering areas
Multiprocessor Parallelism

Use multiple processors on the same program:

- Divide up a task between processors.
- Dividing up a data structure, each processor working on its own data
- Typically processors need to communicate.
  - Shared memory is one approach
  - Explicit messaging is increasingly common.
  - Distributed shared memory (virtual global address space)
- Load balancing is critical for maintaining good performance.
Granularity of Parallelism

- Defined as the size of the computations that are being performed in parallel

- Four types of parallelism (in order of granularity size)
  - Instruction-level parallelism (e.g. pipeline)
  - Thread-level parallelism (e.g. run a multi-thread java program)
  - Process-level parallelism (e.g. run an MPI job in a cluster)
  - Job-level parallelism (e.g. run a batch of independent jobs in a cluster)
Dependency and Parallelism

- Dependency: If event B must occur after event A, then B is dependent on A

- Two types of Dependency
  - Control dependency: waiting for the instruction which controls the execution flow to be completed
    - IF (X!=0) Then Y=1.0/X: Y has the control dependency on X!=0
  - Data dependency: dependency due to memory access
    - Flow dependency: A=X+Y; B=A+C;
    - Anti-dependency: B=A+C; A=X+Y;
    - Output dependency: A=2; X=A+1; A=5;
Identifying Dependency

→ Draw a Directed Acyclic Graph (DAG) to identify the dependency among a sequence of instructions

- **Anti-dependency**: a variable appears as a parent in a calculation and then as a child in a later calculation

- **Output dependency**: a variable appears as a child in a calculation and then as a child again in a later calculation

![Diagram](image)
High Performance Computing

Course Notes 2009-2010

Models of Parallel Programming
Models of Parallel Programming

Different approaches for programming on parallel and distributed computing systems include:

- Dedicated languages designed specifically for parallel computers
- Smart compilers, which automatically parallelise sequential codes
- Data parallelism: multiple processors run the same operation on different elements of a data structure
- Shared memory: processors share a common address space
- Message passing: the memory in each processor has its own address space
Specially Designed Language
Occam

→ Occam is a concurrent programming language

→ Occam is an executable implementation of Communicating Sequential Processes (CSP) theory
  - CSP: a mathematical theory for describing the interactions of tasks in a concurrent system
  - Can theoretically prove if the program written in Occam is correct

→ Occam is specially designed to make full use of the architecture characteristics of the computer system consisting of the transputer (transistor computer) chips, developed by INMOS
  - Transputer is the first microprocessor specially designed for parallel computing
  - A number of transputer chips are wired to form a complete computer system (no bus, RAM or OS)

→ In Occam, the processes communicate through channels
  - Channel can be regarded as the message passing mechanism within a computer
Occam

Sequential execution
SEQ
  x := x + 1
  y := x * x

Parallel execution:
PAR
  p()
  q()

Communication between processes:
ProcessA ! Channel_var
ProcessB ? Channel_var
Occam

- The Occam language was not popular
  - Poor portability
  - Transputer chip is very expensive

For more information:

Occam 2 reference manual

www.wotug.org/occam/documentation/oc21refman.pdf

Occam archive

http://vl.fmnet.info/occam/

Transputer

Dedicated languages

In general dedicated languages are going to do a better job
1. Designed with the hardware architecture
2. Structure of the language reflects the nature of parallelism

However
1. Niche languages are not generally popular
2. It’s hard to port existing code

Much better to modify and extend an existing language to include parallelism, because
1. Better audience
2. Only need to learn new constructs or API, not a new language
3. Porting is a lot easier