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 it’s 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

```
X = A + B
D = X * 17
A = B + C
X = C + E
```