Computing architectures Part 2

TMA4280—Introduction to Supercomputing

NTNU, IMF

January 16. 2017

Supercomputing

What is the motivation for Supercomputing?

Solve complex problems fast and accurately:

— efforts in modelling and simulation push sciences and engineering applications forward,

— computational requirements drive the development of new hardware and software.

$\rightarrow$ architectures and software models evolved with the algorithms.

Supercomputers at NTNU: History

Supercomputing center established in 1983.

Year	System	Processors	Type	GFLOPS
1986–1992	Cray X-MP	2	Vector	0.5
1992–1996	Cray Y-MP	4	Vector	1.3
1995–2003	Cray J90	8	Vector	1.6
1992–1999	Intel Paragon	56	MPP	5.0
1996–2003	Cray T3E	96	MPP	58
2000–2001	SGI O2	160	ccNUMA	100
2001–2008	SGI O3	898	ccNUMA	1000
2006–2011	IBM P5+	2976	Distributed SMP	23500
2012–	SGI Altix ICE X	23040	Distributed SMP	497230

$\rightarrow$ representative of the evolution of supercomputers

Evolution: system architecture
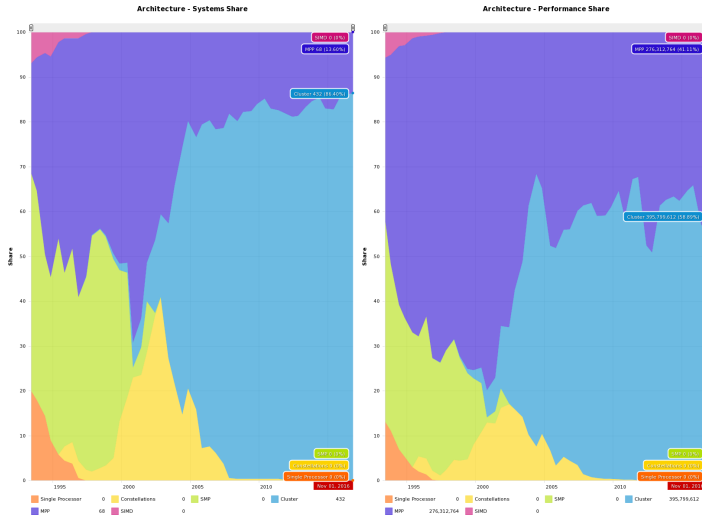
Comparison of the evolution w.r.t system and performance share.

Supercomputers

— 70's–80's: vector processors (CRAY-1 1976);
  one or a few expensive, custom-made chips.
— 80's–: MPP systems, Constellations;
  many processors; standard micro-processors but possibly proprietary
  interconnects.
— Current trend: multicore systems;
  heterogeneous computing.

$\rightarrow$ chosen solutions are a compromise between requirements and costs.

Challenges of parallel computing

1. Limited parallelism in programs:

Amdahl's law: speed-up on p processor w.r.t serial

$$\mathcal{S}_p = \frac{T_1}{T_p} = \frac{p}{f(1-p)+1}$$

with f fraction of time spent in serial sections of the code.

The fraction f:

— $\rightarrow 1$ for purely serial case

— $\rightarrow 0$ for idea parallel case

Linear strong scaling: $f = 0$

$$\mathcal{S}_p = \frac{T_1}{T_p} = p$$

ideal speed-up when solving a fixed problem on p processors.
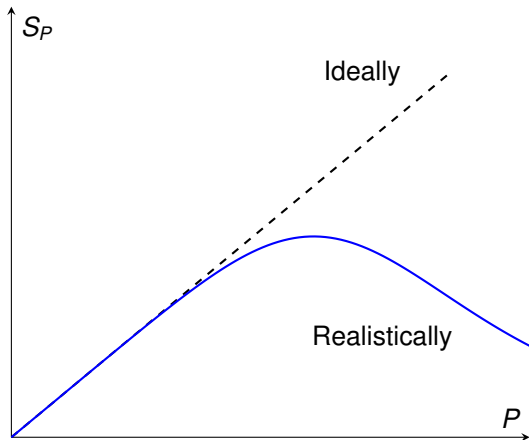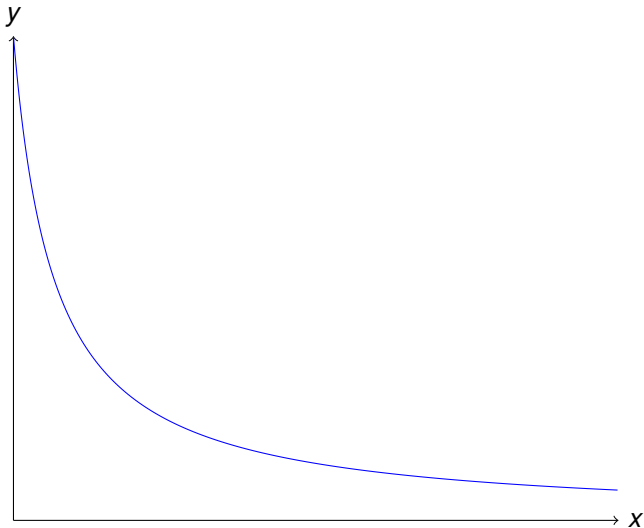
Challenges of parallel computing

Figure: Ideal speedup ($S_P = P$) and realistic speedup.

$\rightarrow$ strong scaling should be interpreted carefully! $\rightarrow$ weak scaling should also be considered.

Challenges of parallel computing

Example Amdhal function for a node on Vilje: 16 processors

$\rightarrow$ steep decrease in efficiency

Challenges of parallel computing

Amdahl's law: relative speed-up

$$S_p = \frac{1}{\frac{\bar{f}}{\bar{S}_p} + (1 - \bar{f})}$$

with $\bar{f}$ fraction of time spent in **parallel** sections of the code.

Example: How to reach 80 percent of theoretical with 100 processors?

$\rightarrow$ less then 0.25 should be spent in serial.

$\rightarrow$ crucial to (re-)think algorithms for parallelism.

Challenges of parallel computing

2. Communication cost if high

— data movement between the different level of memory hierarchy

— connections (bus, network) may exhibit hight latencies

Latencies can be addressed by:

1. architectures: caching data.
2. software: rewriting data structures to improve locality.

Single processor systems: memory hierarchy

CPU

Registers

Cache

Main memory (RAM)

Local disk

Distributed memory

Tape
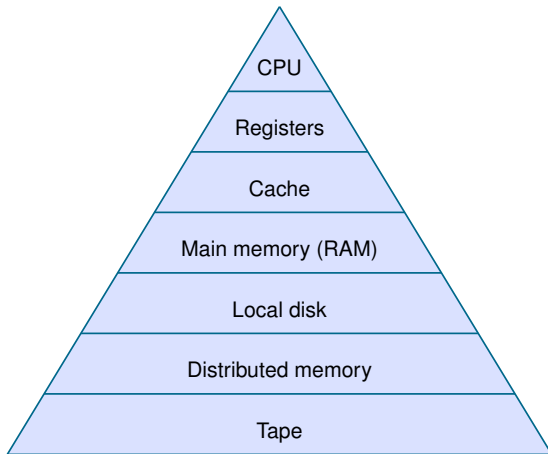
Flynn's Taxonomy

Table: Flynn's taxonomy:

	SD	MD
SI	Von Neumann (single-processor)	SIMD (vector processors)
MD		MIMD (multiprocessors)

grain-size: amount of computation performed by a parallel task.
$\rightarrow$ granularity influences the choice of a solution.

— SISD: ILP, memory hierarchy
— SIMD: DLP, fine-grained parallelism
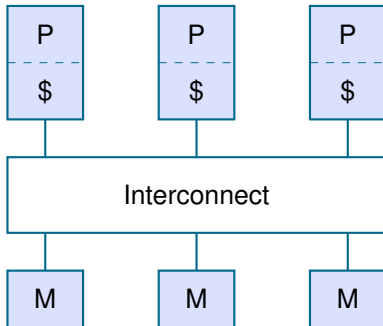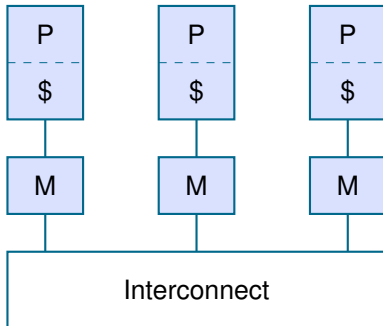— MIMD: TLP, coarse-grained parallelism

Multi-processor systems

Challenges:

— communication between processors
  (memory access, programming models);

— computational methods or algorithms;

— scalability (hardware and algorithms);

— large volumes of data (storage and visualization).
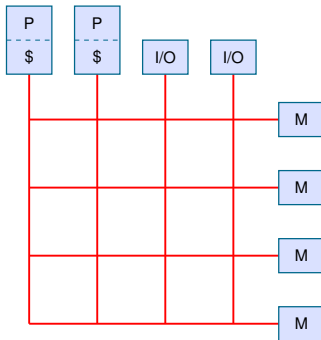
Shared memory access

Distributed memory access

```
┌─────┐   ┌─────┐   ┌─────┐
│  P  │   │  P  │   │  P  │
│ - - │   │ - - │   │ - - │
│  $  │   │  $  │   │  $  │
└─────┘   └─────┘   └─────┘
   │         │         │
┌─────┐   ┌─────┐   ┌─────┐
│  M  │   │  M  │   │  M  │
└─────┘   └─────┘   └─────┘
   │         │         │
┌───────────────────────────┐
│        Interconnect        │
└───────────────────────────┘
```

Shared memory: uniform access

This is called a *symmetric multi-processor*. Examples: bus-based, switch-based and crossbar organizations. Challenges: cache coherency and cost.
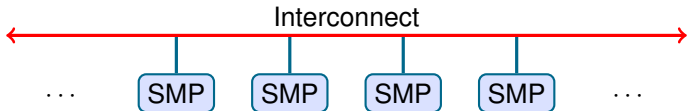
Shared memory: non-uniform access

This is called NUMA or ccNUMA (*cache-coherent non-uniform memory access*). Example: Several SMPs connected with a high-speed low-latency network. Each SMP has uniform memory access internally.
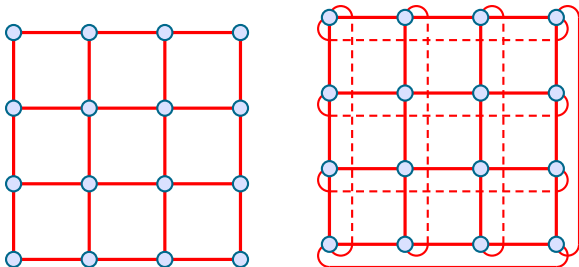
Distributed memory systems

Only the local address space is available to each processor. Data from other processors are only available through explicit message-passing.

Interconnect (message passing)
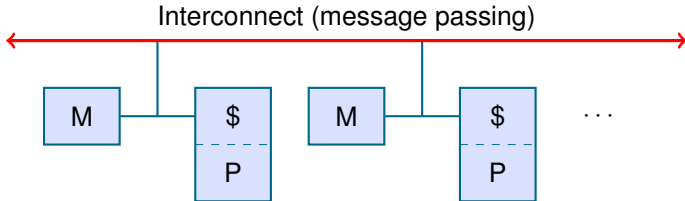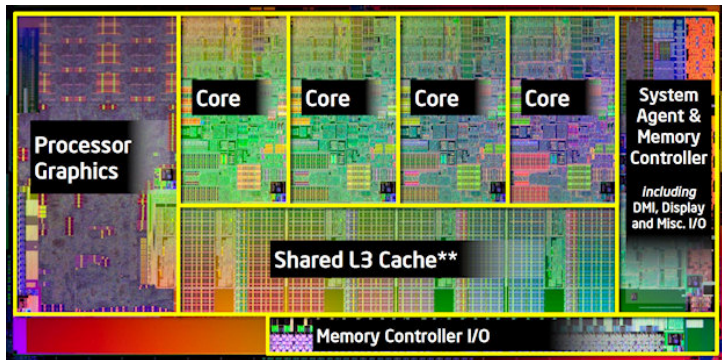
Network topology

Examples: 2D mesh or toroid. Vilje is an eight-dimensional hyper-cube (!).

The current supercomputer at NTNU

Based on the Intel Sandy Bridge microprocessor, an octa-core chip (image shows the quad-core version)

The current supercomputer at NTNU

Intel Sandy bridge E5-2670:

— An octa-core chip (8 physical processing cores)

— Caches and memory:
  - private L1 cache (32kB instruction+32kB data) 3 clocks;
  - private L2 cache (256kB) 8 clocks;
  - shared L3 cache (20MB) $\sim$ 30 clocks (could not find info);
  - main memory (32GB) $\sim$ 150 clocks (could not find info).

— FMA capable AVX unit, meaning 8 Flop per cycle, SSE 4.x.

— Simultaneous multi-threading (SMT): Intel calls this "hyperthreading": each processor core can handle two instruction streams at the same time. Problem: Shared SIMD units.

A node: two E5-2670 chips
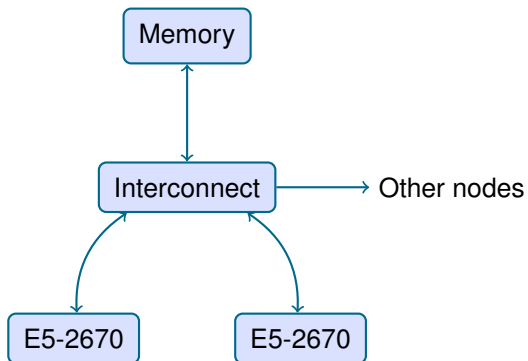
Key data

Vilje:

— 1440 nodes or 23040 physical cores;
— 16-core shared memory within a single node;
— distributed memory across nodes;
— 394 TB storage.
— 8.6GB/s aggregated bandwidth.

Programming models:

— shared memory programming model (OpenMP) within a node;
— message passing (MPI) across nodes;
— also possible: message passing within a single node;
— also possible: both models within the same program, hybrid.

Levels of parallelism: Single processor

Core
— pipelining
— superscalar execution
— vector processing (SIMD unit)
— branch prediction
— caching techniques
— multithreading
— prefetching
— …

Instruction-level parallelism, Concurrency, SIMD unit

Levels of parallelism: Multi-processor

Compute node
 — multiple cores on a chip
 — core sharing cache memory
 — affinity, locality groups
 — accelerators
 — . . .

Shared memory model (OpenMP)

Levels of parallelism: Distributed system

Cluster (system comprised of several compute nodes)

— network topologies

— optimized libraries

— communication patterns

— . . .

Distributed memory model (MPI)