1️⃣

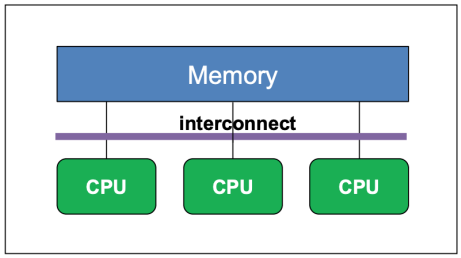
Introduction

Files	Introduction.pdf
Link	
Status	Done
Type	Lecture

Architectures

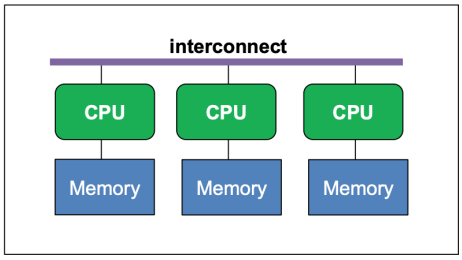
- Shared-Memory

  Programming via OpenMP and Java-Threads



- Distributed-Memory

  Programming via MPI



Definitions

- Latency: time it takes to send a message of size zero (set up the communication channel)
- Bandwidth: rate at which large messages are transferred
- Process: abstraction of a program in memory (there are no common variable, each has its own "address space")
- Thread: lightweight process meaning it has common variables with all other threads belonging to the same processes. But has its own stack.

> 📌 A thread can only be on one processor. A process can be on multiple processors by having multiple threads.

Measuring Parallelisation Effectiveness

- **Time using 1 CPU:** $T(1)$
- **Time using p CPUs:** $T(p)$
- **Speedup S:** $S(p) = \frac{T(1)}{T(p)}$
  - Measures how much fast the parallel computation is
- **Efficiency E:** $E(p) = \frac{S(p)}{p}$

- **Example:**
  - $T(1) = 6s,\ T(2) = 4s$
    - $\rightarrow S(2) = \frac{6}{4} = \frac{3}{2} = 1.5$
    - $\rightarrow E(2) = \frac{1.5}{2} = \frac{3}{4} = 0.75$
- **Ideal case: $T(p) = T(1)/p$**
  - $S(p) = p$
  - $E(p) = 1.0$

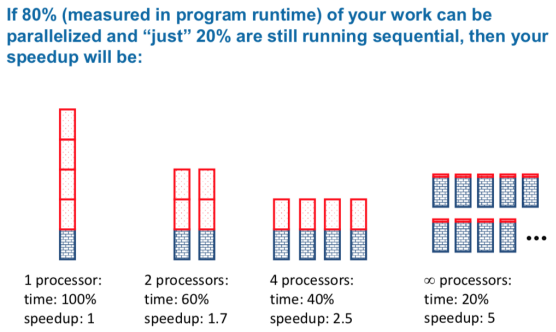
Amdahl's Law

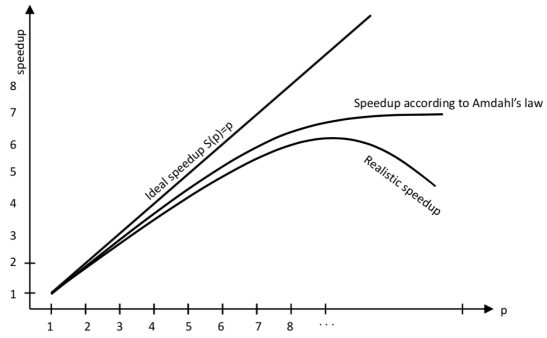
Describes the influence of the serial part on scalability. Does not take overhead into account.

Basically, because of the serial part even if there is an infinite number of cores, there is still a limit on speedup and efficiency because of the serial part.

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

where f is the serial part (0-1]





Concurrency & Granularity

Concurrency could be in either form of:

- Task Decomposition
- Data Decomposition

Comparing Granularity and Parallelisation Levels

Aa Parallelisation Level	Load Balancing Issues	Communicaton Issues
High	High	Low

Parallelisation Level	Load Balancing Issues	Communicaton Issues
Low	Low	High

Possible Issues

- Issues of serial programming
- Harder to debug than serial code
- Overhead (scheduling and such)
- Data Races/Race conditions (value updated in middle of another calculation)
- Deadlocks (threads waiting for resources of another, possibly in a cycle)
- Load balancing (thread done and waiting for others because the data was not balanced properly)
- Serialisation (if the threads are waiting for the results from another thread)
- Irreproducibility (different numerical results)

> "Efficient parallelisation is about minimising the overhead introduced by the parallelisation itself"