FARABI UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE



Lecture 1.
Parallel Hardware Architectures and Parallel Programming Models

Dr. Bazargul Matkerim, Professor-Researcher

Outline

1. **Parallel hardware architectures**

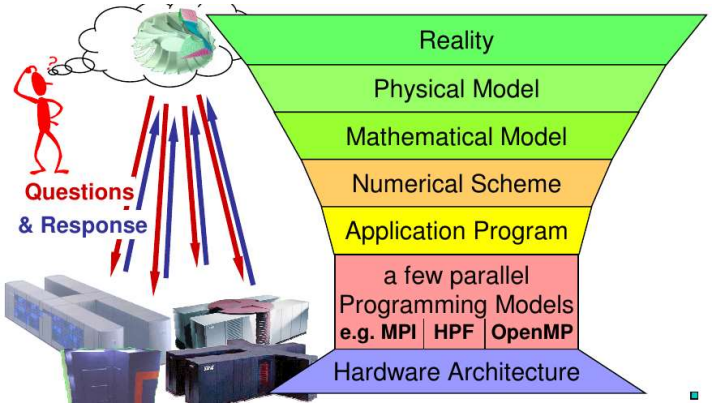
   concepts, shared and distributed memory, hybrid systems, interconnects

2. **Parallel programming models**

   parallelization strategies, OpenMP, HPF, MPI

3. **Which parallel programming model is the best for my application?**

   speedup, efficiency, Amdahl's law



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 15

Parallel Processing Concepts

Pipelining → **vector computing**

Functional parallelism → modern processor technology

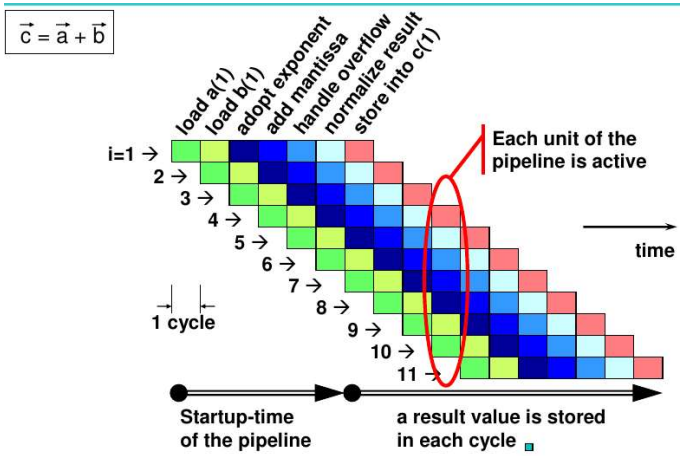
Combined instructions, e.g. multiply-add

Multithreading, array processing

Multiprocessors (strongly coupled) → **shared memory**

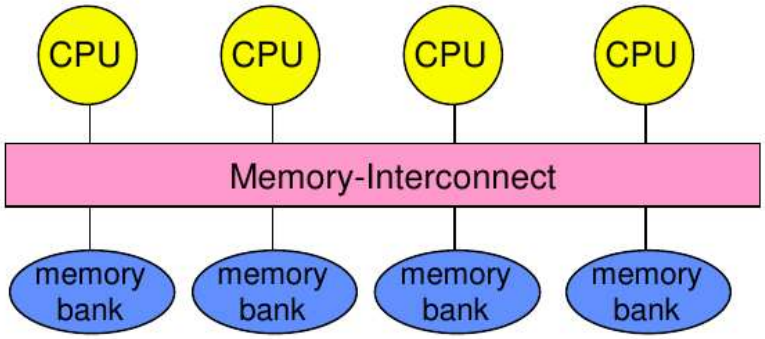
Multicomputers (weakly coupled) → **distributed memory**
both together: hybrid architectures

Memory access: cache, memory banks, pre-fetch →
MFLOP/s and **MB/s**



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 62

Multiprocessor – Shared Memory



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 6

All CPUs reach all memory banks with the **same speed**

Uniform Memory Access (UMA)

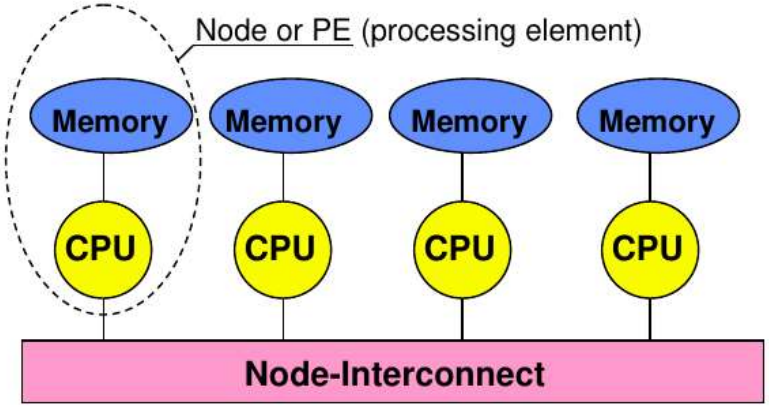
Symmetric Multi-Processing (SMP)

Network types:

- crossbar: independent access from each CPU
- bus: one CPU can block the others

Programming: threads, **OpenMP**

Multicomputer – Distributed Memory



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 7

Nodes (PE) are coupled by a node interconnect

Each CPU:

- fast access to its own memory
- slower access to other CPUs' memories

Non-Uniform Memory Access (NUMA)

Networks: bus, torus, crossbar

Programming: message passing, **MPI**

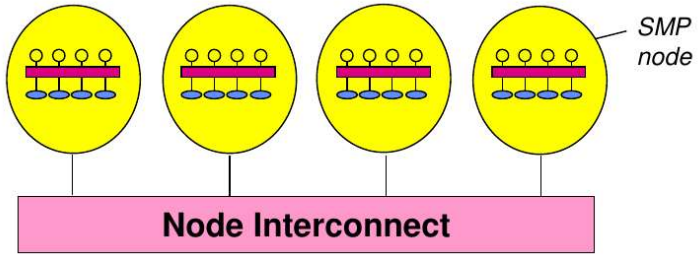
Hybrid Architectures

Most modern HPC systems are **clusters of SMP nodes**

SMP inside each node, **DMP** on the node interconnect

ccNUMA: looks like one big SMP, but is a cluster of small SMPs

Programming: **MPI + OpenMP**



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 8



Source: commons.wikimedia.org/wiki/File:IBM_Blue_Gene_P_supercomputer.jpg (CC BY-SA 2.0)

IBM Blue Gene/P: a cluster of SMP nodes

Interconnects

Bus based networks: **cheap**, but poor

Multi-link networks:

- ring with independent connections
- 2-D / 3-D torus: links to 4 or 6 neighbours
- fat tree: higher levels have more bandwidth

Ring, torus, fat tree: **scalable**

Crossbar (single level): high switch costs

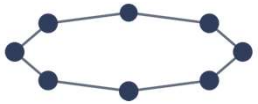
Full interconnect: **not scalable**

Number of links in a full interconnect:

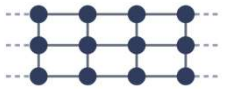
$$L = \frac{n(n-1)}{2}$$



Bus



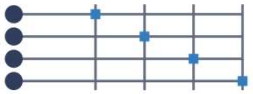
Ring



2-D torus



Fat tree



Crossbar



Full interconnect

Why Parallel Hardware Architectures?

The response of only one processor is **not just in time**

Moore's law: transistors on a chip double about every 18 months

the number of processors on a chip will grow

You own:

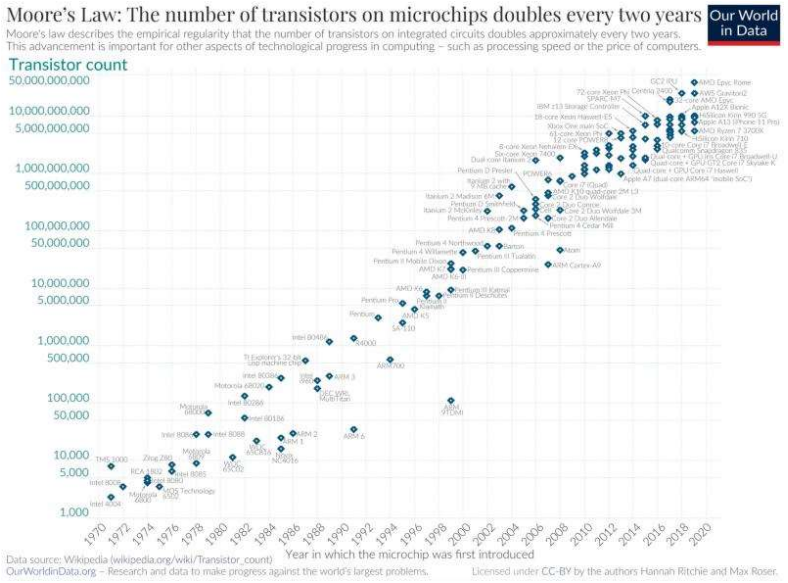
- a network of workstations (NOW)
- a Beowulf cluster of COTS PCs
- a dual- or quad-board PC

Huge applications with **huge memory needs**



A Beowulf cluster: ordinary PCs connected by a network

Source: commons.wikimedia.org (GPL)



Source: Our World in Data, commons.wikimedia.org (CC BY 4.0)

Parallelization Strategies

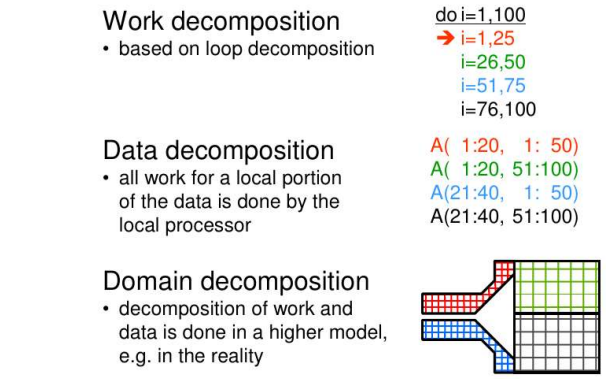
Two resources: **processor** and **memory**

Parallelization means:

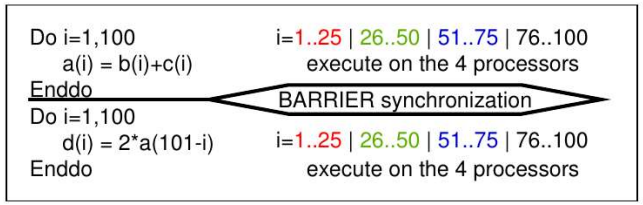
- distributing work to processors
- distributing data (if memory is distributed)
- synchronization of the distributed work
- communication of remote data

Synchronization may cause **idle time** and overhead

Programming models offer a combined method for all of this



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 17



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 18

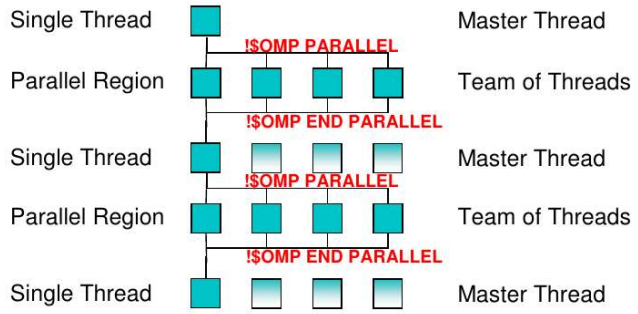
Major Programming Models

OpenMP: shared memory directives
- user defines work decomposition
- no data decomposition, implicit synchronization

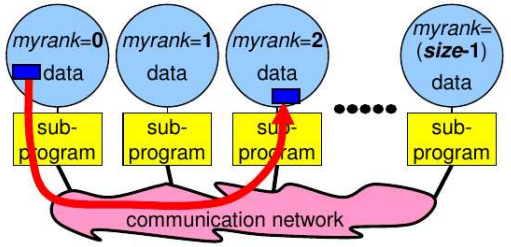
HPF: data parallelism
- user defines data decomposition
- communication is implicit

MPI: message passing
- user distributes work and data
- communication by MPI library routines

Hybrid systems: **MPI + OpenMP**



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 22



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 28

Speedup, Efficiency and Amdahl's Law

Definitions

$$S(p,N) = \frac{T(1,N)}{T(p,N)}$$

$$E(p,N) = \frac{S(p,N)}{p}$$

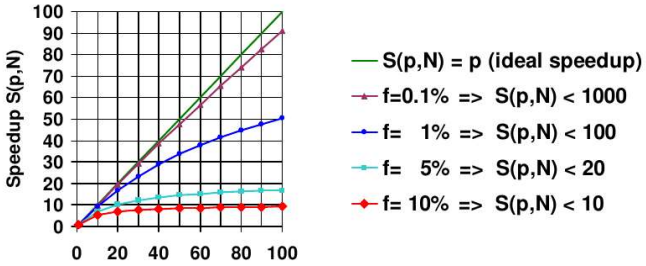
Amdahl's law

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

f — sequential part of the code

Example: f = 5 %, p = 100

$$S = \frac{1}{0.05 + \frac{0.95}{100}} \approx 16.8$$



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 45

Which Programming Model Is the Best?

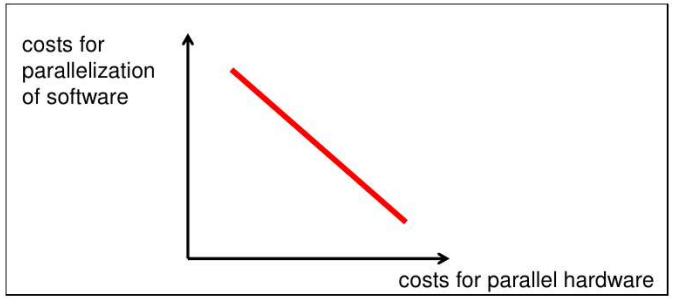
Available parallelization strategies?
    loop parallelism, domain decomposition

Which hardware architecture? (today and in the future)

How many working hours for parallelizing the code?

Shared memory: OpenMP, HPF, MPI
    expensive hardware, cheap programming effort

Distributed memory: MPI, HPF, shmem, Co-array Fortran
    cheap hardware, expensive programming effort



Source: R. Rabenseifner, HLRS, Lecture 1 material, slide 48