Basics of Parallel Systems and Programs

9th SimLab Course on Parallel Numerical Simulation

October 3—9, 2010, Belgrade, Serbia

Ralf-Peter Mundani
Technische Universität München

Stabilitätspakt für Südosteuropa
Gefördert durch Deutschland
Stability Pact for South Eastern Europe
Sponsored by Germany

IGSSE TUM
International Graduate School of Science and Engineering

Overview

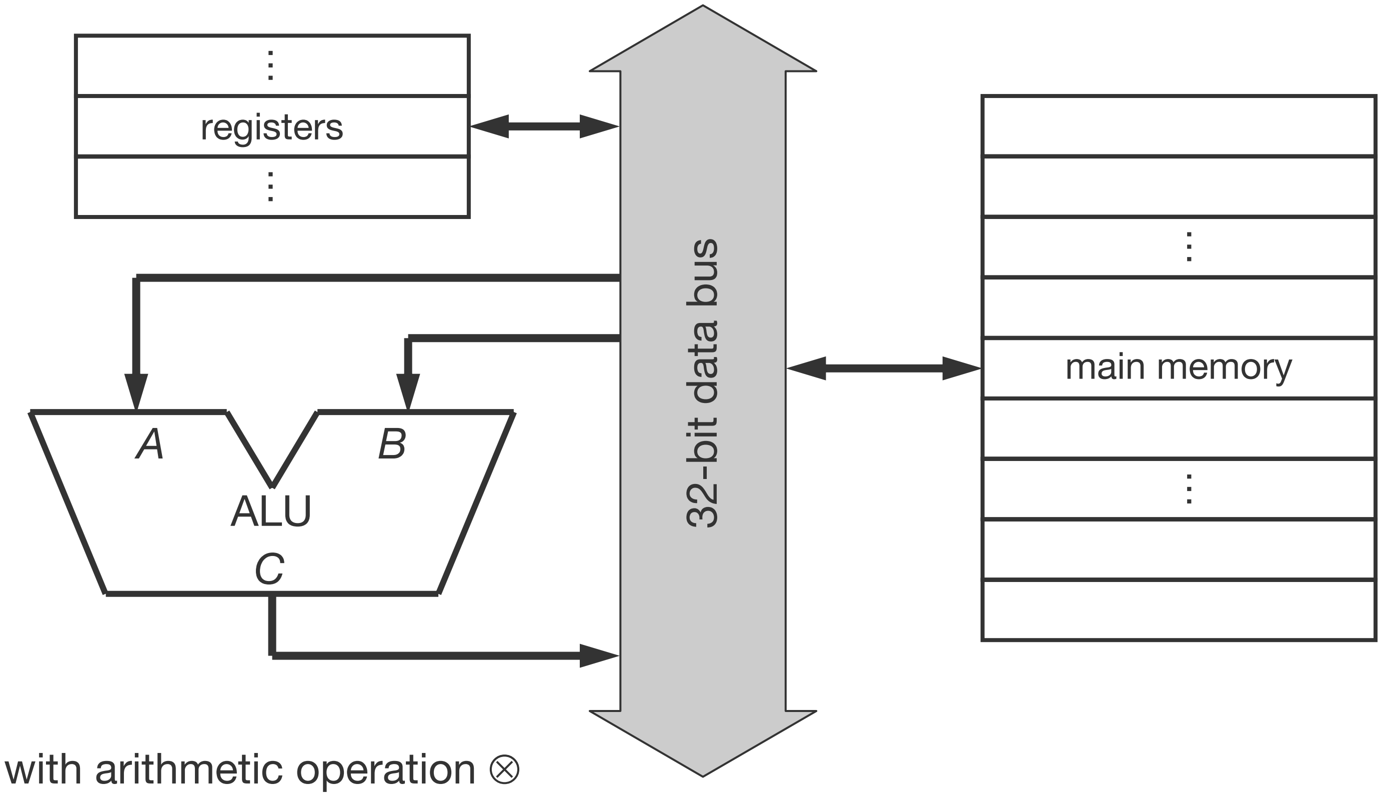
- hardware aspects
- supercomputers
- network topologies
- terms and definitions

Supercomputer: Turns CPU-bound problems into I/O-bound problems.

—Ken Batcher

Hardware Aspects

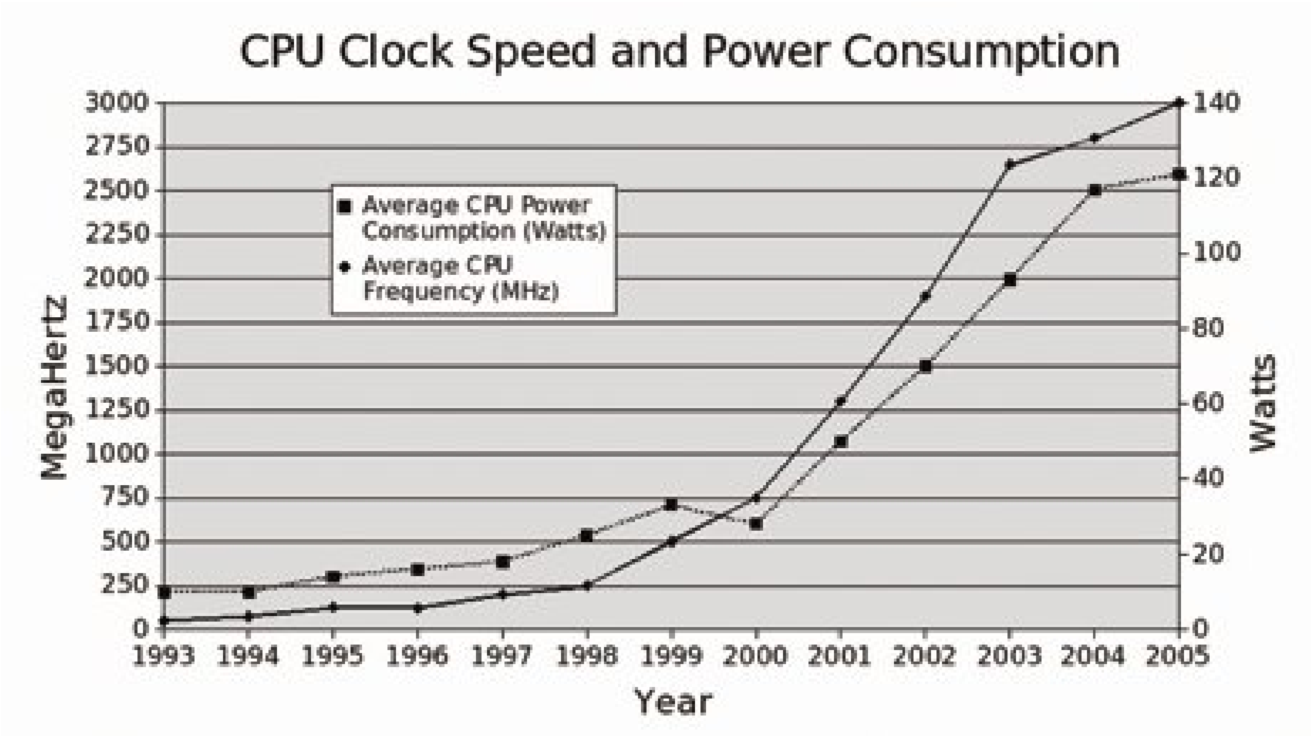
- reminder: arithmetic logical unit (ALU)
  - schematic layout of the (classical 32-bit) arithmetic logical unit



$C \leftarrow A \otimes B$ with arithmetic operation $\otimes$

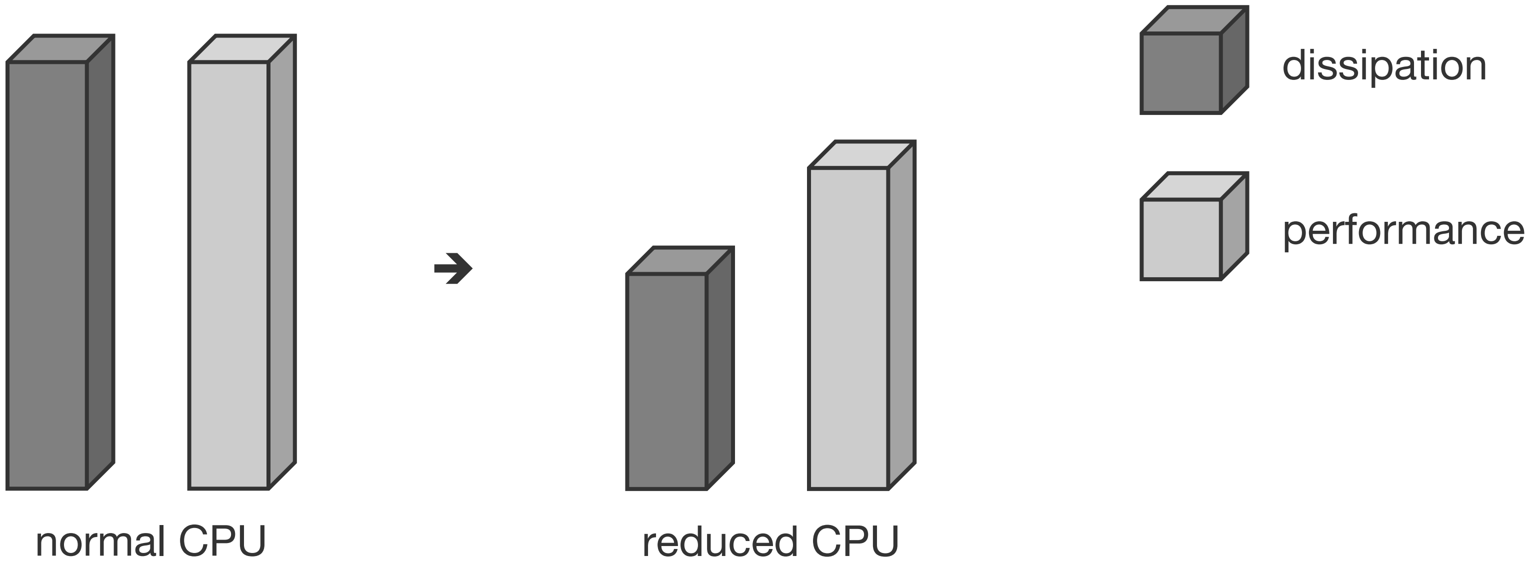
Hardware Aspects

- dual core, quad core, manycore, and multicore
  - observation: increasing frequency (and thus core voltage) over past years
  - problem: thermal power dissipation $P \sim f \cdot v^2$ (f: frequency; v: core voltage)



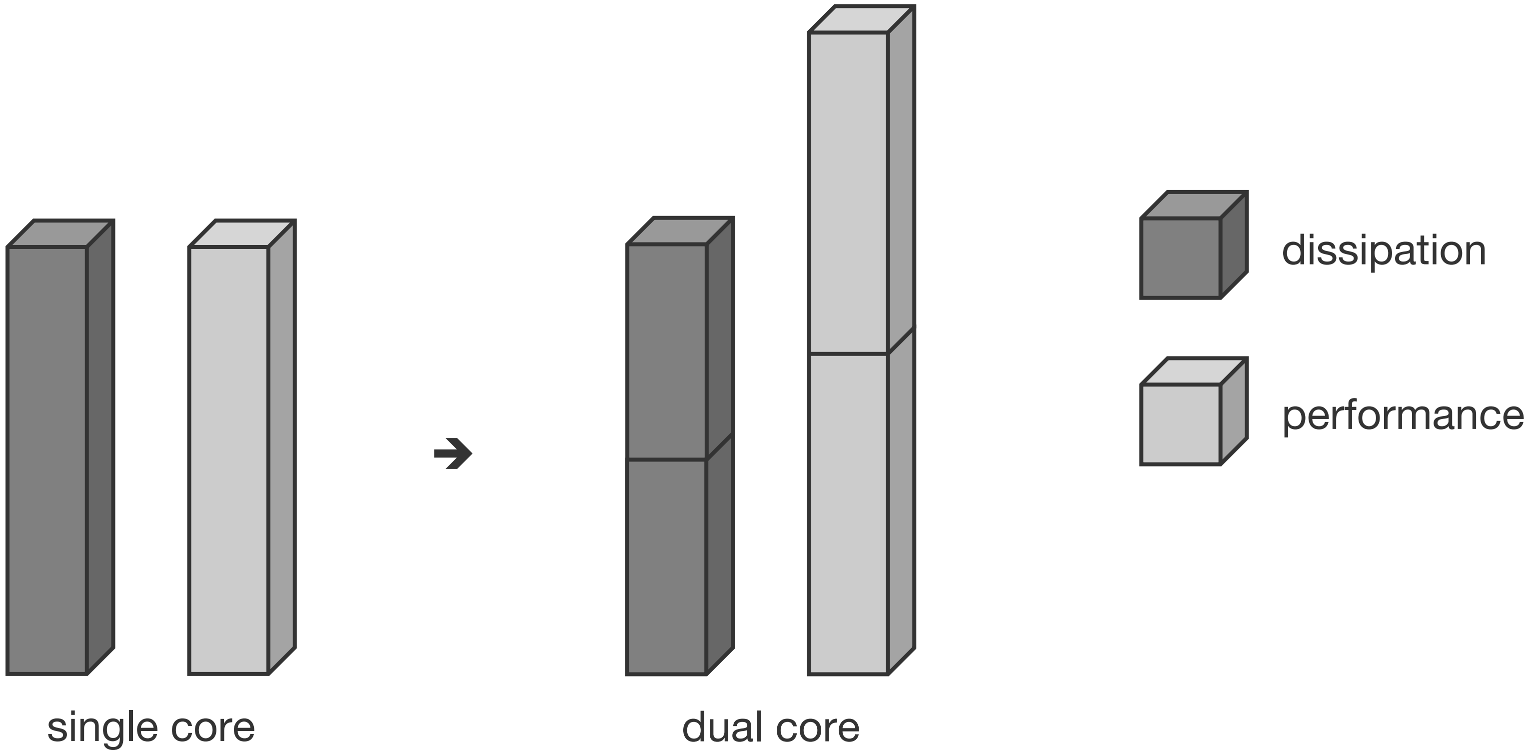
Hardware Aspects

- dual core, quad core, manycore, and multicore (cont'd)
  - 25% reduction in performance (i. e. core voltage) leads to approx. 50% reduction in dissipation



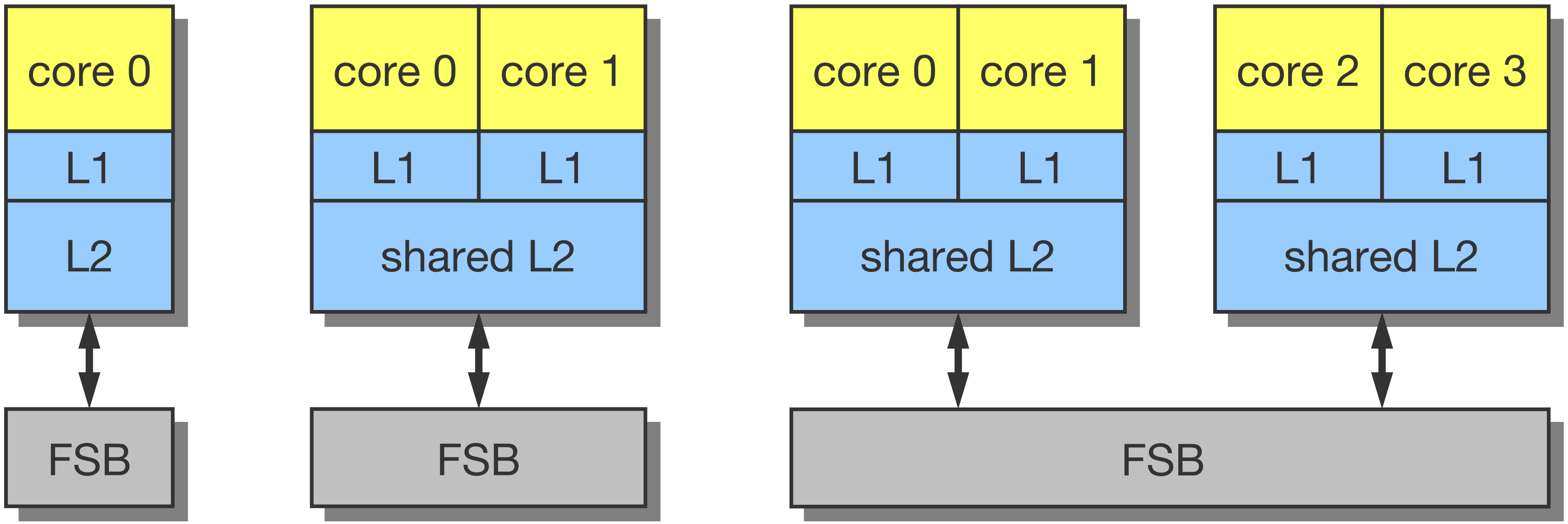
Hardware Aspects

- dual core, quad core, manycore, and multicore (cont'd)
  - idea: installation of two cores per die with same dissipation as single core system



Hardware Aspects

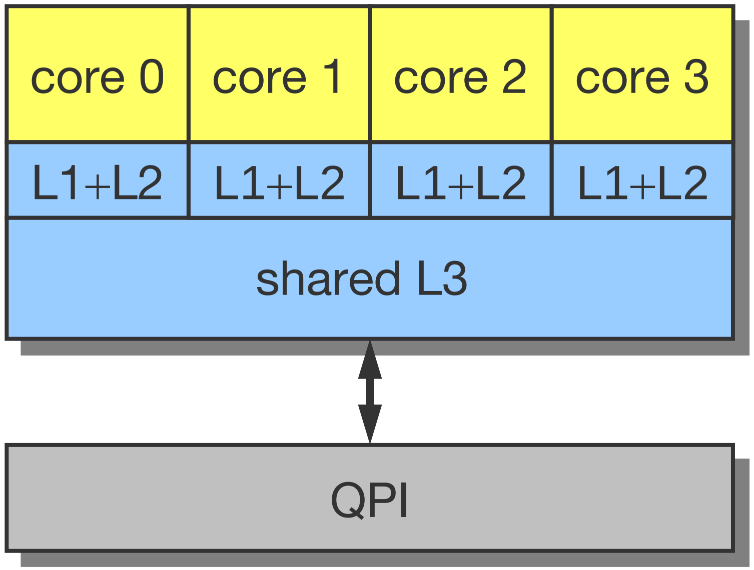
- dual core, quad core, manycore, and multicore (cont'd)
  - single vs. dual/quad

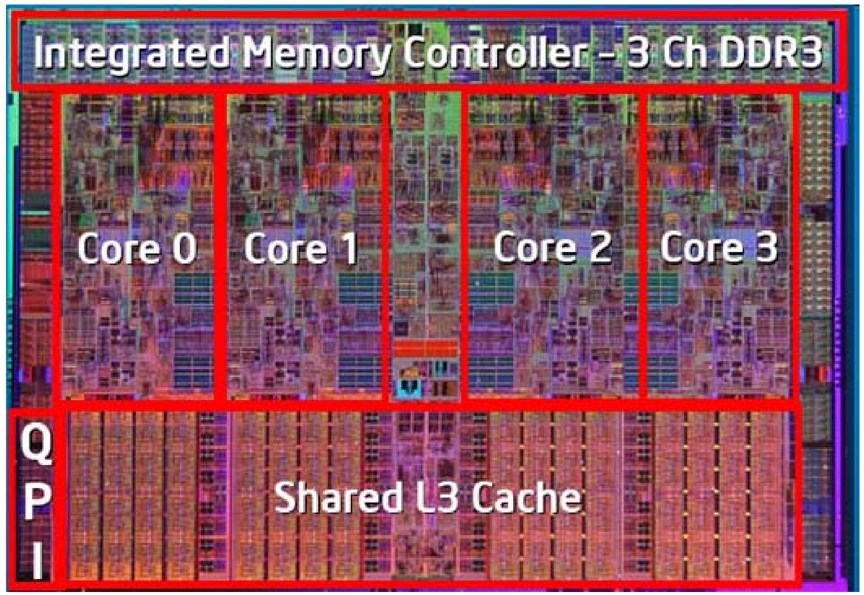


FSB: front side bus (i. e. connection to memory (via north bridge))

Hardware Aspects

- INTEL Nehalem Core i7



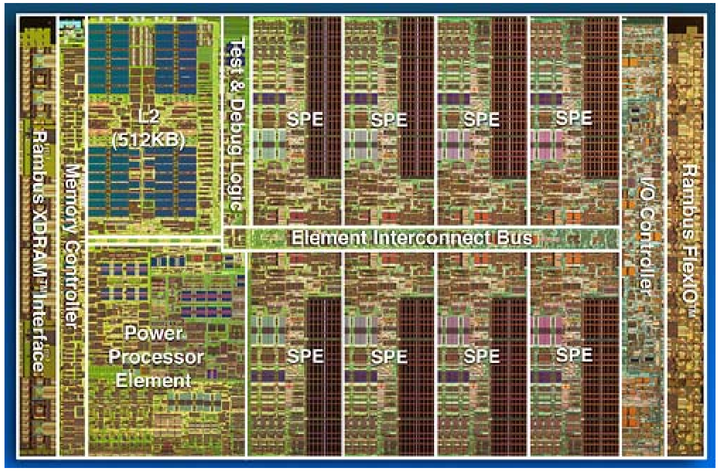


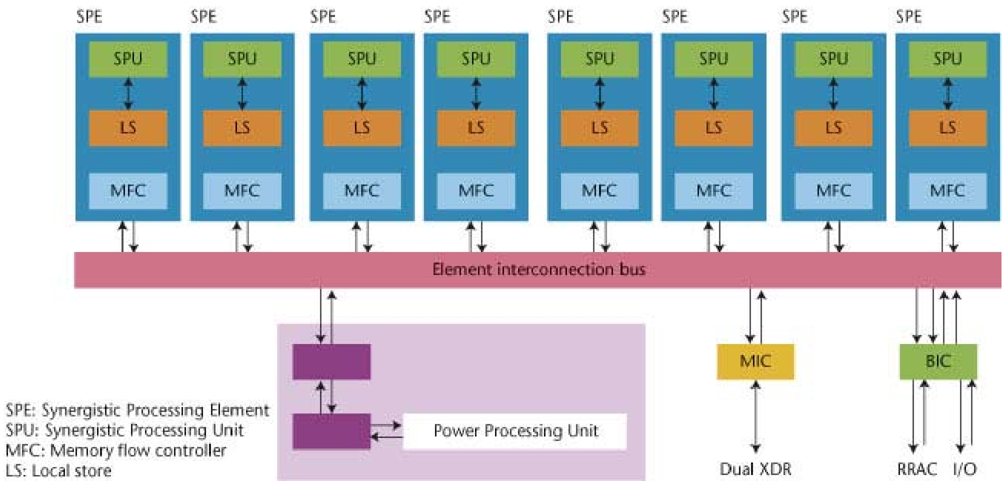
source: www.samrathacks.com

QPI: QuickPath Interconnect replaces FSB (QPI is a point-to-point interconnection – with a memory controller now on-die – in order to allow both reduced latency and higher bandwidth ➔ up to (theoretically) 25.6 GByte/s data transfer, i. e. 2× FSB)

Hardware Aspects

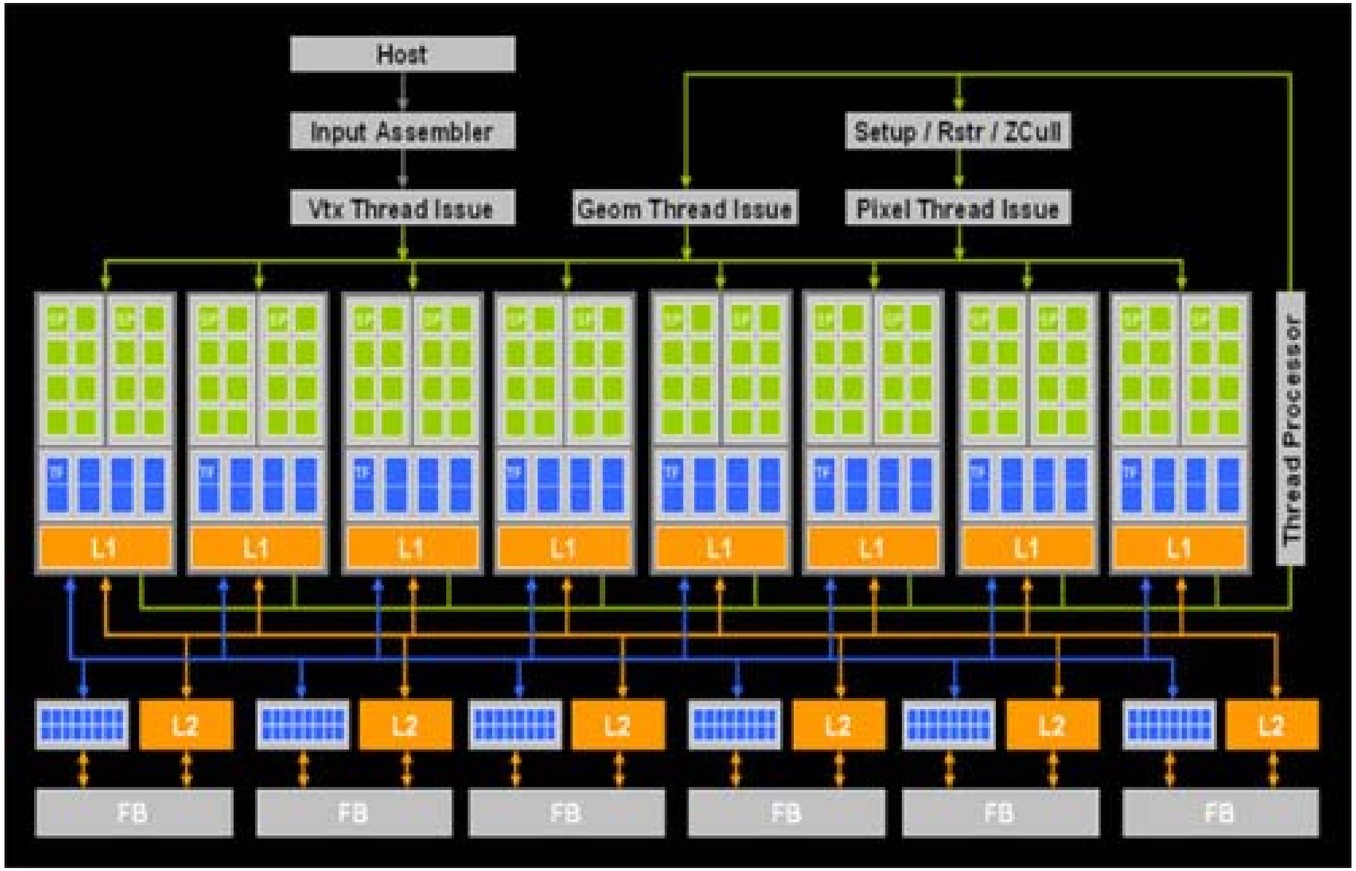
- Cell Broadband Engine Architecture
  - jointly developed by Sony, Toshiba, IBM (so called "STI alliance") starting in 2001 with a budget of approximate 400 M$
  - cell processor consists of
    - one main processor called *power processing element* (PPE)
    - 8 co-processors called *synergistic processing elements* (SPE)
    - a high-bandwidth bus connecting PPE, SPEs, and I/O elements





Hardware Aspects

- GPU computing
  - example: nVidia G80 kernel with 128 (8800 GTX) or 96 (8800 GTS) units



Hardware Aspects

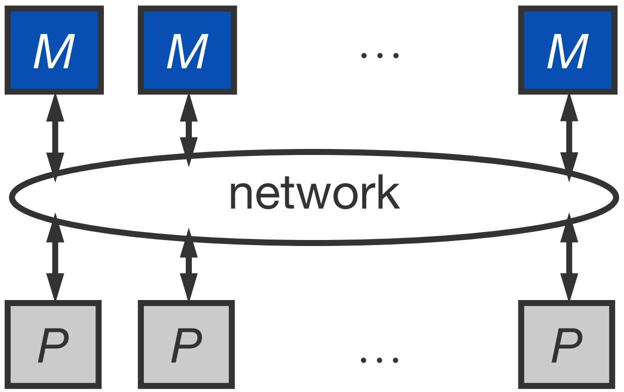
- dawning of manycore and multicore
  - **strong aspects**
    - shared memory ➔ simple parallelisation
    - increasing number of cores (> 100 proposed by INTEL)
    - also other HW available: GPU, e. g.
    - …
  - **weak aspects**
    - parallelisation necessary (TANSTAFL!)
    - memory limitations (parallel read/write access)
    - race conditions (algorithm dependent)
    - limited scalability

Overview

- motivation
- supercomputers
- network topologies
- terms and definitions

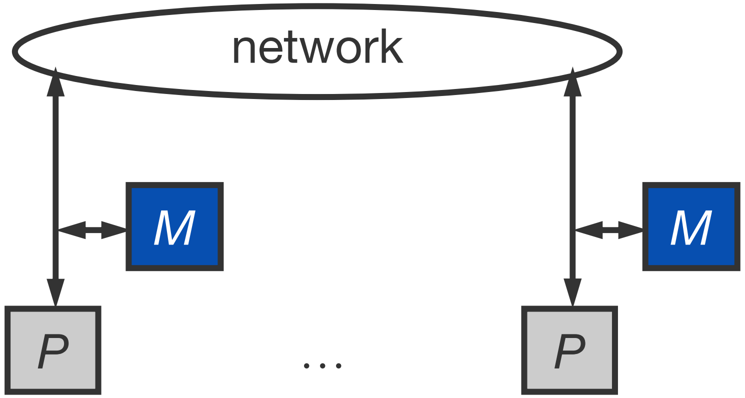
Supercomputers

- Processor coupling
  - uniform memory access (UMA)
    - each processor *P* has direct access via the network to each memory module *M* with same access times to all data
    - standard programming model can be used (i. e. no explicit send / receive of messages necessary)
    - communication and synchronisation via shared variables (inconsistencies (write conflicts, e. g.) have to prevented in general by the programmer)



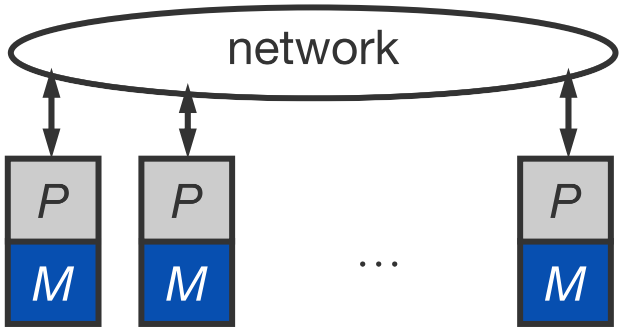
Supercomputers

- Processor coupling (cont'd)
  - non-uniform memory access (NUMA)
    - memory modules physically distributed among processors
    - shared address space, but access times depend on location of data (i. e. local addresses faster than remote addresses)
    - differences in access times are visible in the program
    - example: DSM / VSM, Cray T3E



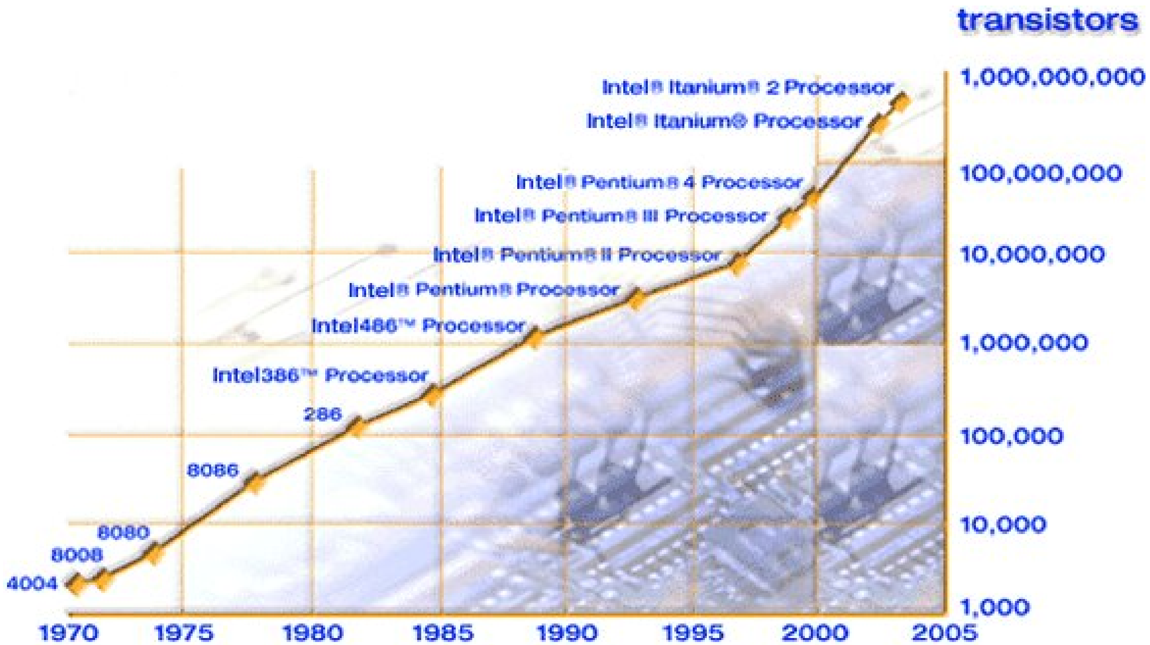
Supercomputers

- Processor coupling (cont'd)
  - no remote memory access (NORMA)
    - each processor has direct access to its local memory only
    - access to remote memory only via explicit message exchange (due to distributed address space) possible
    - synchronisation implicitly via the exchange of messages
    - performance improvement between memory and I/O due to parallel data transfer (Direct Memory Access, e. g.) possible
    - example: IBM SP2, ASCI Red / Blue / White



Supercomputers

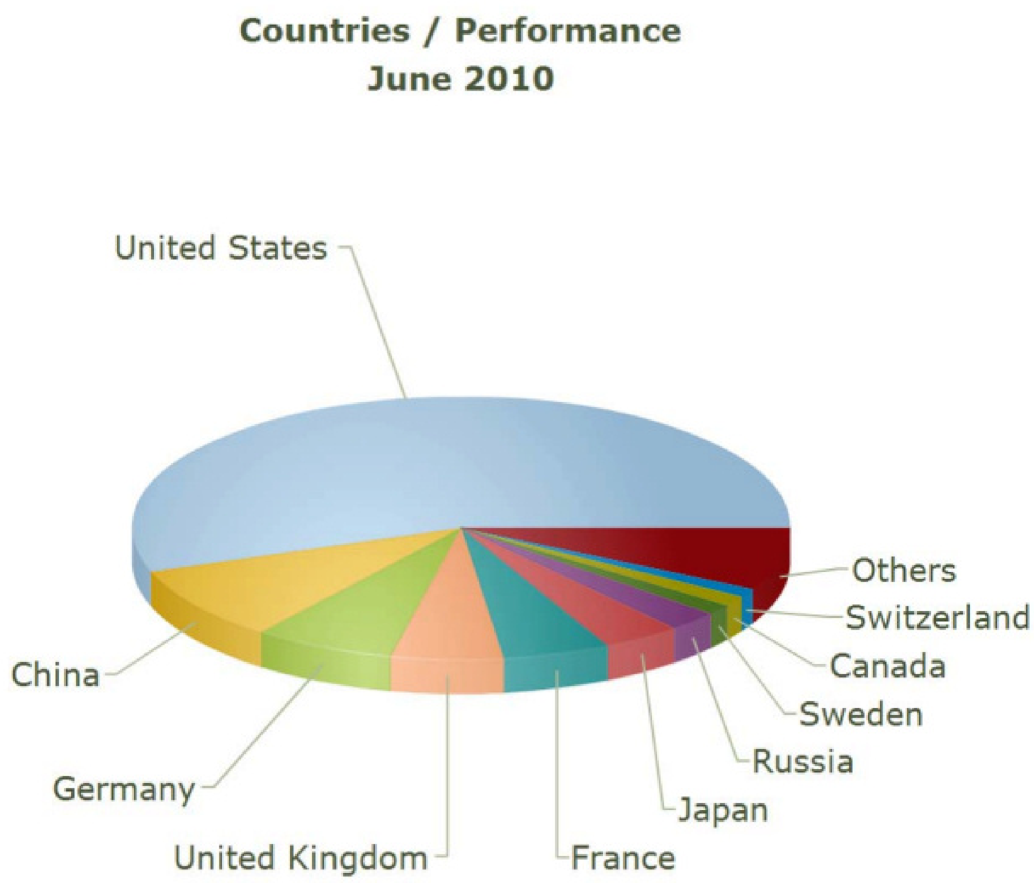
- MOORE's law
  - observation of Intel co-founder Gordon E. MOORE, describes important trend in history of computer hardware (1965)



number of transistors that can be placed on an integrated circuit is increasing exponentially, doubling approximately every two years

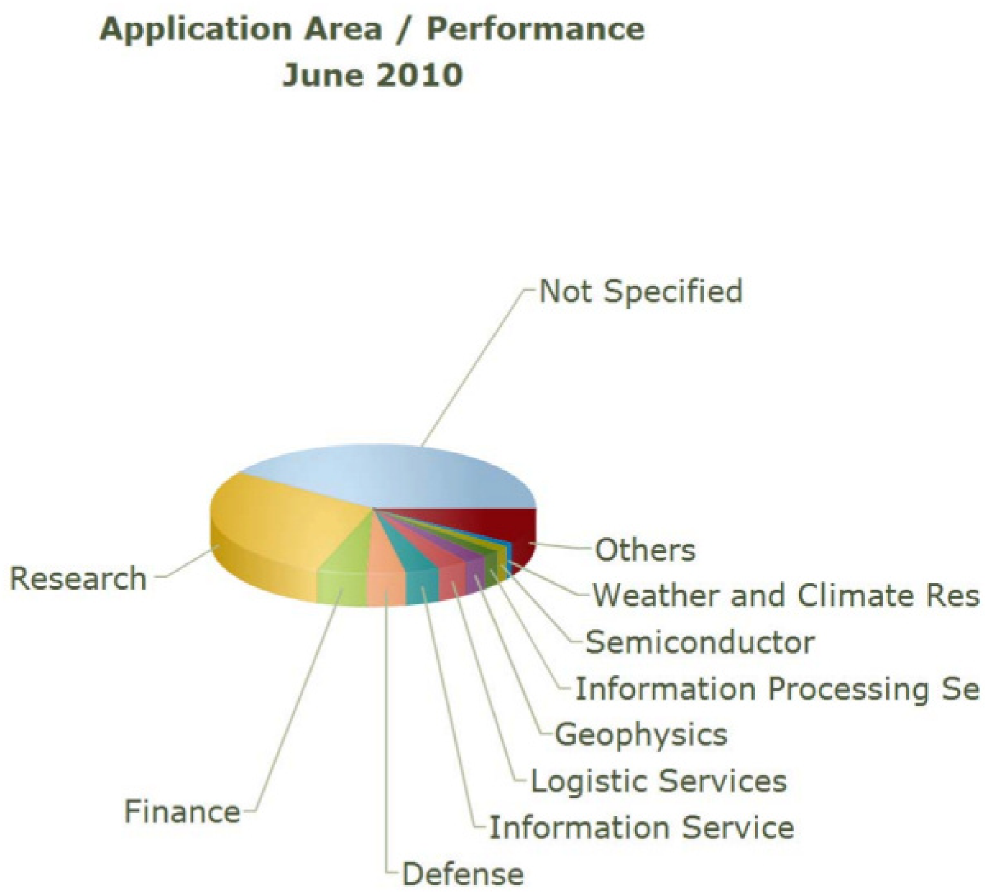
Supercomputers

- some numbers: Top500



Supercomputers

- some numbers: Top500 (cont'd)

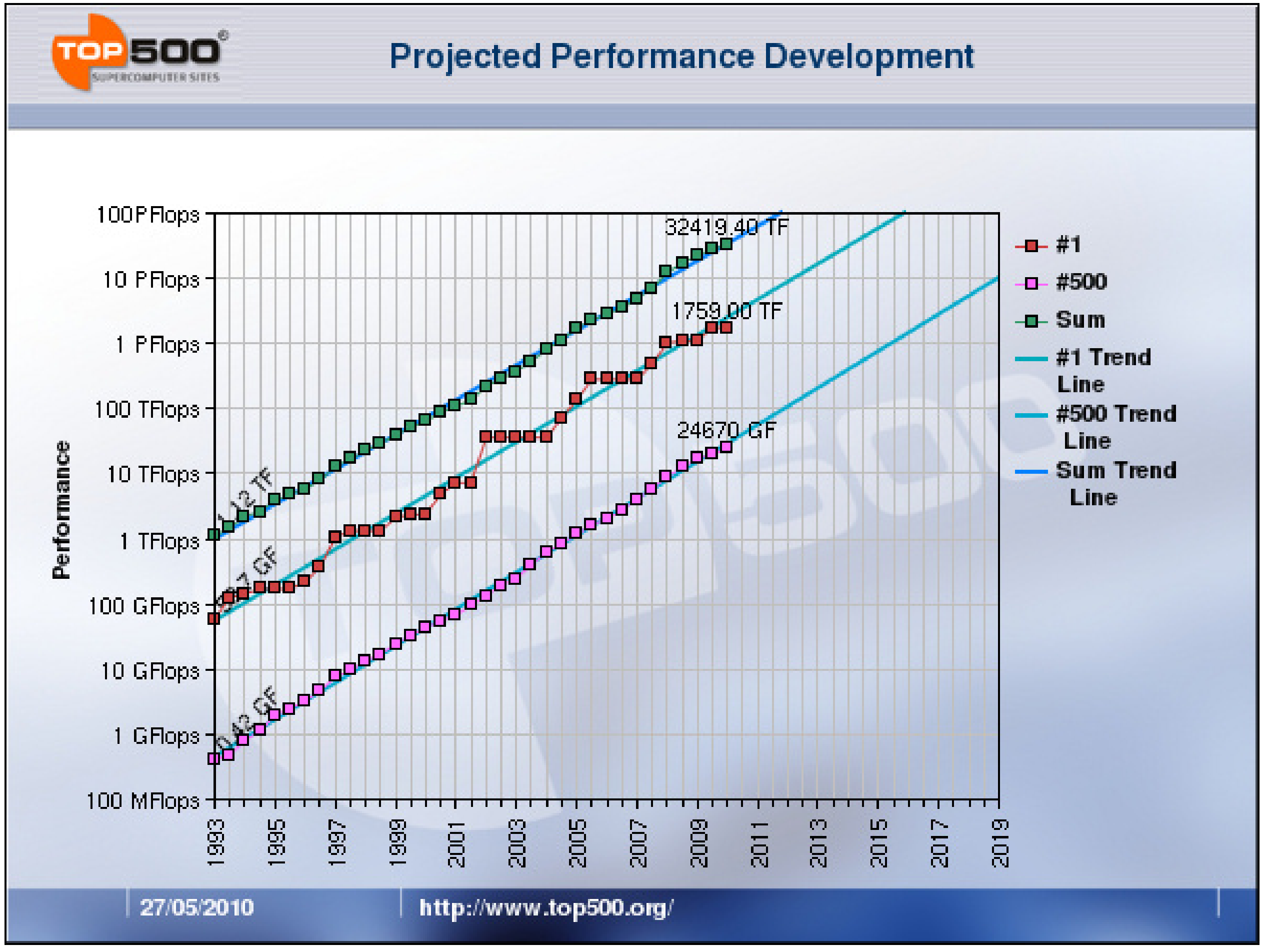


Supercomputers

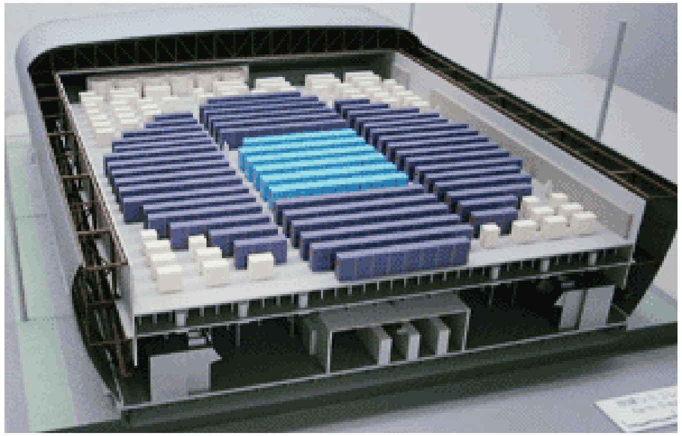
- some numbers: Top500 (cont'd)

Supercomputers

- some numbers: Top500 (cont'd)



Supercomputers



The Earth Simulator – world's #1 from 2002—04

- installed in 2002 in Yokohama, Japan
- ES-building (approx. 50m × 65m × 17m)
- based on NEC SX-6 architecture
- developed by three governmental agencies
- highly parallel vector supercomputer
- consists of 640 nodes (plus 2 control & 128 data switching)
  - 8 vector processors (8GFlops each)
  - 16GB shared memory

  ➔ 5120 processors (40.96TFlops peak performance) and 10TB memory; 35.86TFlops sustained performance (Linpack)

- nodes connected by 640×640 single stage crossbar (83,200 cables with a total extension of 2400km; 8TBps total bandwidth)
- further 700TB disc space and 1.60PB mass storage

Supercomputers

BlueGene/L – world’s #1 from 2004–08

- installed in 2005 at LLNL, CA, USA (beta-system in 2004 at IBM)
- cooperation of DoE, LLNL, and IBM
- massive parallel supercomputer
- consists of 65,536 nodes (plus 12 front-end and 1204 I/O nodes)
  - 2 PowerPC 440d processors (2.8GFlops each)
  - 512MB memory

  ➔ 131,072 processors (367.00TFlops peak performance) and 33.50TB memory; 280.60TFlops sustained performance (Linpack)

- nodes configured as 3D torus (32 × 32 × 64); global reduction tree for fast operations (global max / sum) in a few microseconds
- 1024Gbps link to global parallel file system
- further 806TB disc space; operating system SuSE SLES 9

Supercomputers

Roadrunner – world's #1 from 2008–09

- installed in 2008 at LANL, NM, USA
- installation costs about $120 million
- first "hybrid" supercomputer
  - dual-core Opteron
  - Cell Broadband Engine

➔ 129,600 cores (1456.70TFlops peak performance) and 98TB memory; 1144.00TFlops sustained performance (Linpack)

- standard processing (file system I/O, e. g.) handled by Opteron, while mathematically and CPU-intensive tasks are handled by Cell
- 2.35MW power consumption (➔ 437MFlops per Watt ☺)
- primarily usage: ensure safety and reliability of nation's nuclear weapons stockpile, real-time applications (cause & effect in capital markets, bone structures and tissues renderings as patients are being examined, e. g.)

Supercomputers

- Jaguar – world's #1 since 2009
  - installed in 2009 at ORNL, TN, USA
  - each compute node contains
    - two hex-core Opteron 2.6GHz (10.4GFlops)
    - 16GB memory

  ➔ 224,162 cores (2331.00TFlops peak performance); 1759.00TFlops sustained performance (Linpack)

Overview

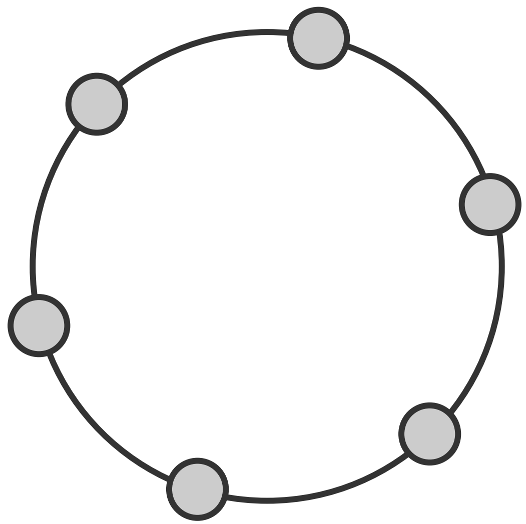
- motivation
- supercomputers
- network topologies
- terms and definitions

Static Network Topologies

- chain (linear array)
  - one-dimensional network
  - N nodes and N–1 edges
  - degree = 2
  - diameter = N–1
  - bisection width = 1
  - drawback: too slow for large N

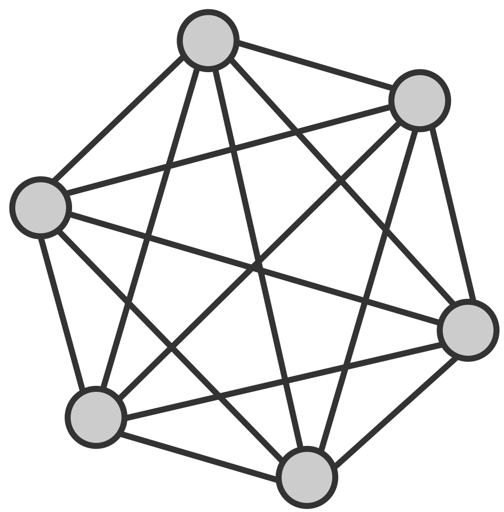
Static Network Topologies

- ring
  - two-dimensional network
  - N nodes and N edges
  - degree = 2
  - diameter = $\lfloor N/2 \rfloor$
  - bisection width = 2
  - drawback: too slow for large N
  - how about fault tolerance?



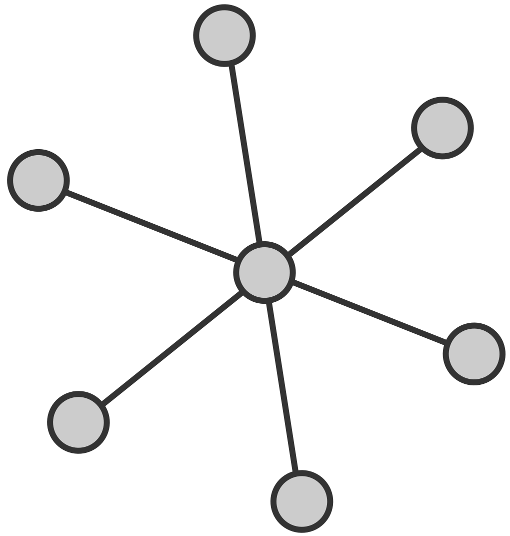
Static Network Topologies

- completely connected
  - two-dimensional network
  - N nodes and $N \cdot (N-1)/2$ edges
  - degree $= N-1$
  - diameter $= 1$
  - bisection width $= \lfloor N/2 \rfloor \cdot \lceil N/2 \rceil$
  - very high fault tolerance
  - drawback: too expensive for large N



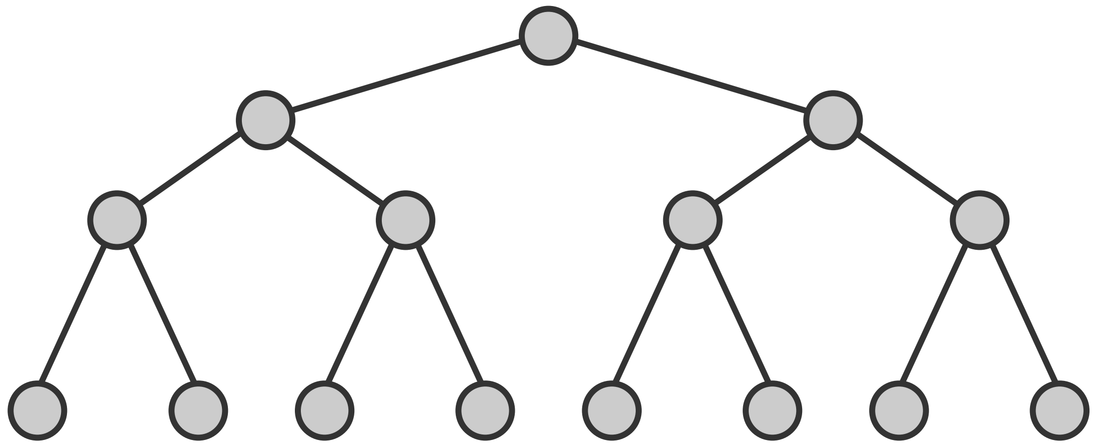
Static Network Topologies

- star
  - two-dimensional network
  - N nodes and $N-1$ edges
  - degree $= N-1$
  - diameter $= 2$
  - bisection width $= \lfloor N/2 \rfloor$
  - drawback: bottleneck in central node



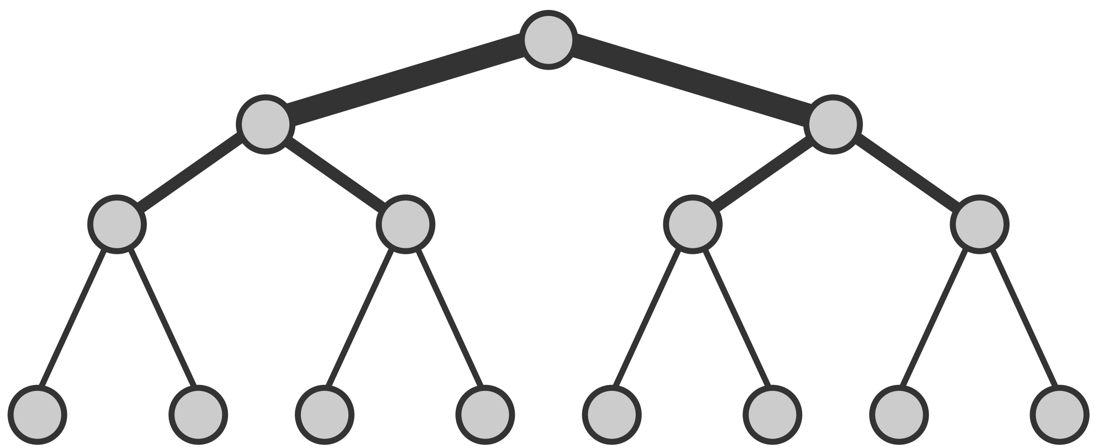
Static Network Topologies

- binary tree
  - two-dimensional network
  - N nodes and $N-1$ edges (tree height $h = \lceil \mathrm{ld}\, N \rceil$)
  - degree = 3
  - diameter = $2(h-1)$
  - bisection width = 1
  - drawback: bottleneck in direction of root (➔ blocking)



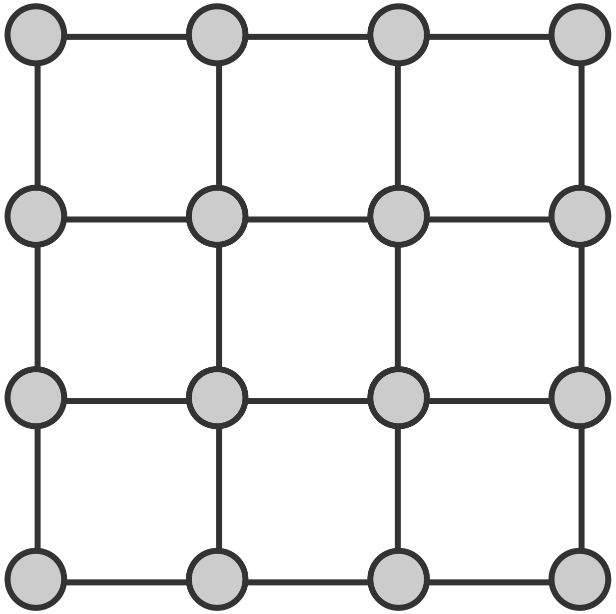
Static Network Topologies

- binary tree (cont'd)
  - solution to overcome the bottleneck ➔ fat tree
  - edges on level *m* get higher priority than edges on level $m+1$
  - capacity is doubled on each higher level
  - now, bisection width $= 2^{h-2}$
  - frequently used: HLRB II, e. g.



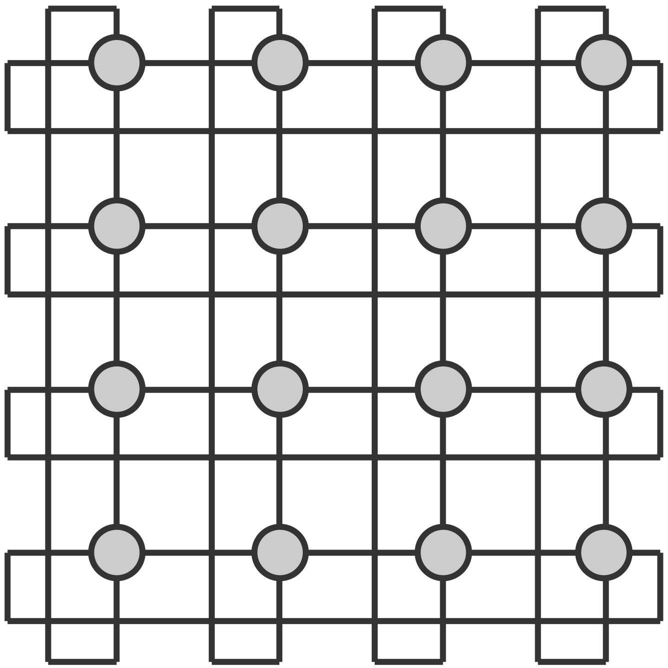
Static Network Topologies

- mesh / torus
  - k-dimensional network
  - N nodes and $k \cdot (N-r)$ edges ($r \times r$ mesh, $r = \sqrt[k]{N}$)
  - degree $= 2k$
  - diameter $= k \cdot (r-1)$
  - bisection width $= r^{k-1}$
  - high fault tolerance
  - drawback
    - large diameter
    - too expensive for $k > 3$



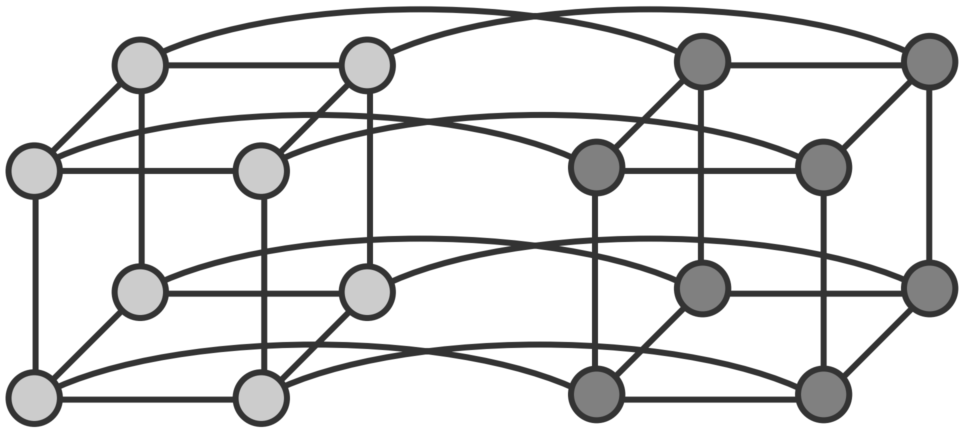
Static Network Topologies

- mesh / torus (cont'd)
  - k-dimensional mesh with cyclic connections in each dimension
  - N nodes and $k \cdot N$ edges ($r \times r$ mesh, $r = \sqrt[k]{N}$)
  - diameter $= k \cdot \lfloor r/2 \rfloor$
  - bisection width $= 2r^{k-1}$
  - frequently used: BlueGene/L, e. g.
  - drawback: too expensive for $k > 3$



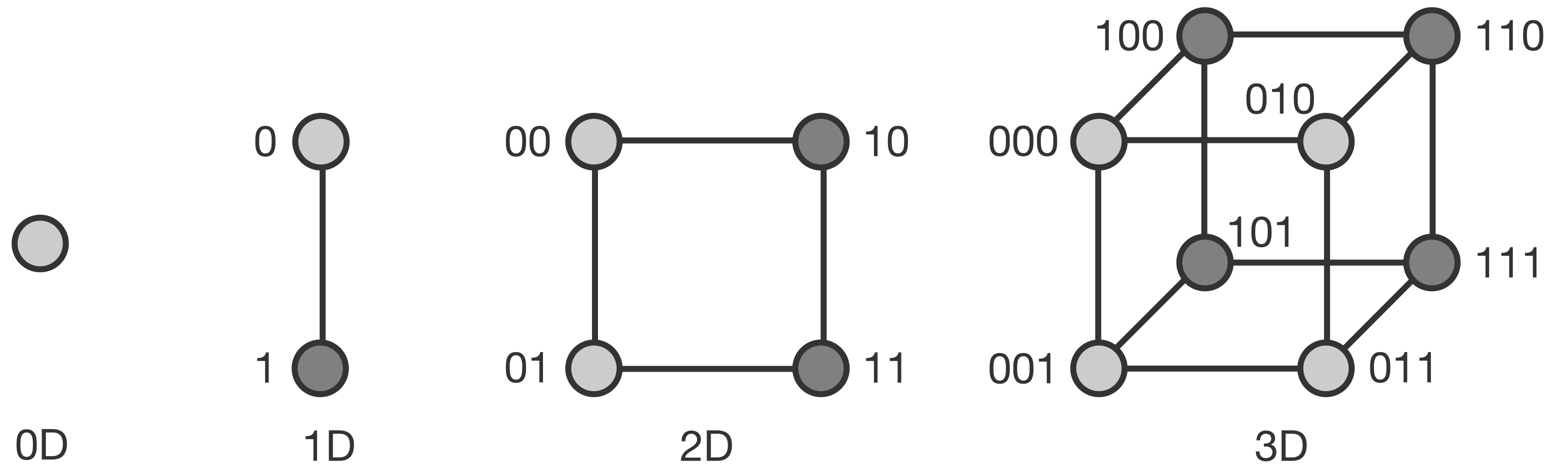
Static Network Topologies

- hypercube
  - k-dimensional network
  - 2^k nodes and $k \cdot 2^{k-1}$ edges
  - degree $= k$
  - diameter $= k$
  - bisection width $= 2^{k-1}$
  - drawback: scalability (only doubling of nodes allowed)



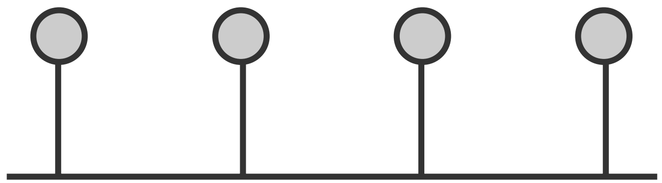
Static Network Topologies

- hypercube (cont'd)
  - principle design
    - construction of a *k*-dimensional hypercube via connection of the corresponding nodes of two *k*–1-dimensional hypercubes
    - inherent labelling via adding prefix "0" to one sub-cube and prefix "1" to the other sub-cube

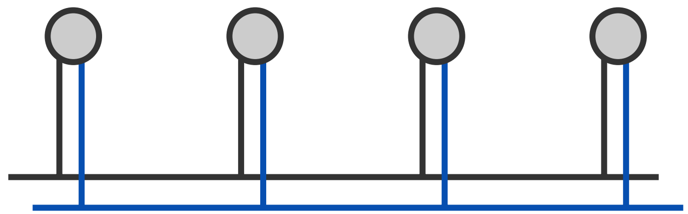


Dynamic Network Topologies

- bus
  - simple and cheap single stage network
  - shared usage from all connected nodes, thus, just one frame transfer at any point in time
  - frame transfer in one step (i. e. diameter = 1)
  - good extensibility, but bad scalability
  - fault tolerance only for multiple bus systems
  - example: Ethernet



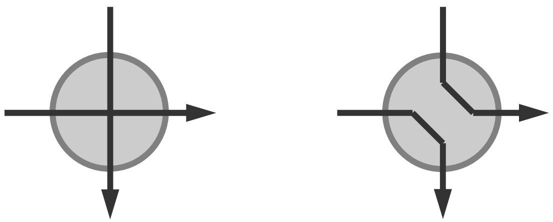
single bus



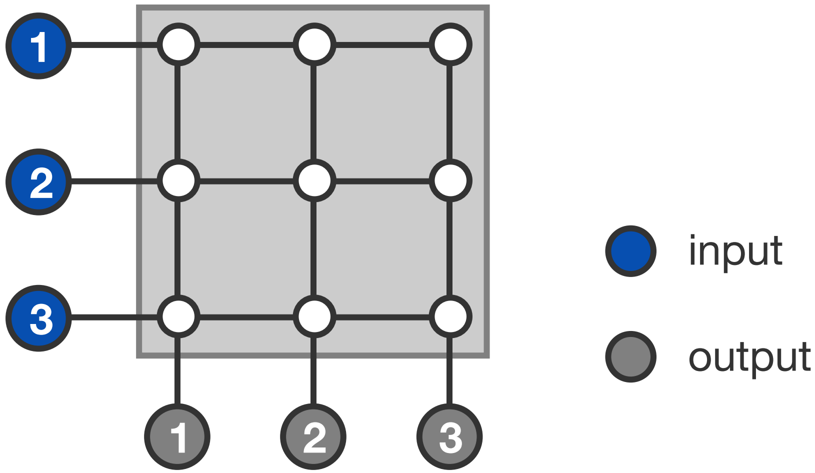
multiple bus (here dual)

Dynamic Network Topologies

- crossbar
  - completely connected network with all possible permutations of N inputs and N outputs (in general $N \times M$ inputs / outputs)
  - switch elements allow simultaneous communication between all possible disjoint pairs of inputs and outputs without blocking
  - very fast (diameter = 1), but expensive due to N^2 switch elements
  - used for processor—processor and processor—memory coupling
  - example: The Earth Simulator



switch element



Overview

- motivation
- supercomputers
- network topologies
- terms and definitions

Terms and Definitions

- quantitative performance evaluation
  - correlation of multi- and monoprocessor systems' performance
  - important: program that can be executed on both systems
  - definitions
    - $P(1)$: amount of unit operations of a program on the monoprocessor system
    - $P(p)$: amount of unit operations of a program on the multiprocessor systems with p processors
    - $T(1)$: execution time of a program on the monoprocessor system (measured in steps or clock cycles)
    - $T(p)$: execution time of a program on the multiprocessor system (measured in steps or clock cycles) with p processors

Terms and Definitions

- quantitative performance evaluation (cont'd)
  - simplifying preconditions
    - $T(1) = P(1)$
      - one operation to be executed in one step on the monoprocessor system
    - $T(p) \leq P(p)$
      - more than one operation to be executed in one step (for $p \geq 2$) on the multiprocessor system with p processors

Terms and Definitions

- quantitative performance evaluation (cont'd)
  - speed-up
    - $S(p)$ indicates the improvement in processing speed

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

    - in general, $1 \leq S(p) \leq p$
  - efficiency
    - $E(p)$ indicates the relative improvement in processing speed

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

    - improvement is normalised by the amount of processors p
    - in general, $1/p \leq E(p) \leq 1$

Terms and Definitions

- quantitative performance evaluation (cont’d)
  - speed-up and efficiency can be seen in two different ways
    - algorithm-independent
      - best known sequential algorithm for the monoprocessor system is compared to the respective parallel algorithm for the multiprocessor system

        ➔ absolute speed-up

        ➔ absolute efficiency
    - algorithm-dependent
      - parallel algorithm is treated as sequential one to measure the execution time on the monoprocessor system; “unfair” due to communication and synchronisation overhead

        ➔ relative speed-up

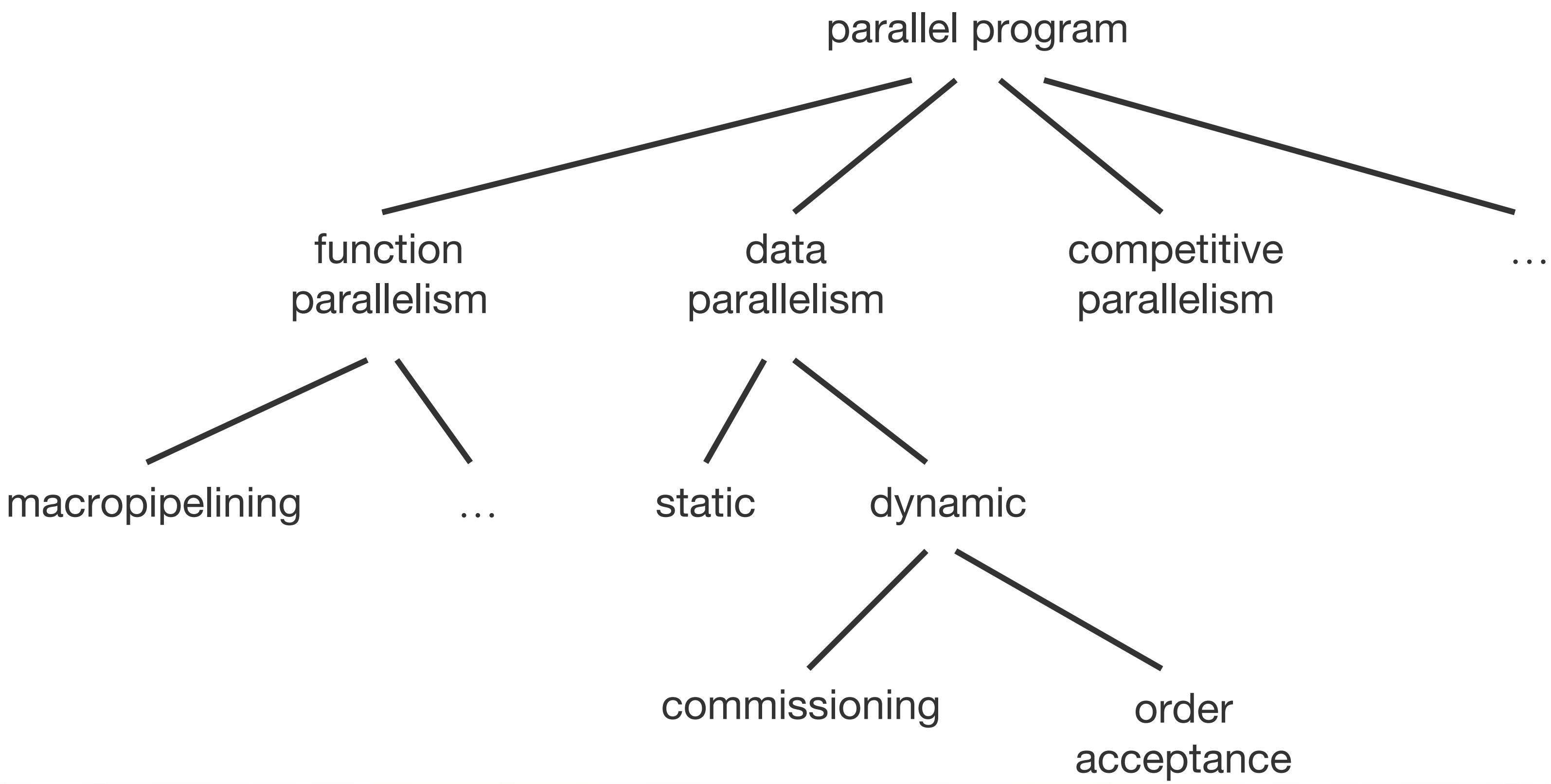
        ➔ relative efficiency

Terms and Definitions

- general design questions
  - several considerations have to be taken into account for writing a parallel program (either from scratch or based on an existing sequential program)
  - standard questions comprise
    - which part of the (sequential) program can be parallelised
    - what kind of structure to be used for parallelisation
    - which parallel programming model to be used
    - what kind of compiler to be used
    - what about load balancing strategies
    - what kind of architecture is the target machine
  - and, better education necessary (*“How many files…”*)

Terms and Definitions

- structures of parallel programs



Terms and Definitions

- dependence analysis
  - processes / (blocks of) instructions cannot be executed simultaneously if there exist dependencies between them
  - hence, a dependence analysis of a given algorithm is necessary
  - example

```
for_all_processes i ← 0 to N do
    a[i] ← i + 1
od
```

  - what about the following code

```
for_all_processes i ← 1 to N do
    x ← i − 2*i + i*i
    a[i] ← a[x]
od
```

  - as it is not always obvious, an algorithmic way of recognising dependencies (via the compiler, e. g.) would preferable

Terms and Definitions

- dependence analysis (cont'd)
  - BERNSTEIN (1966) established a set of conditions, sufficient for determining whether two processes can be executed in parallel
  - definitions
    - I_i (input): set of memory locations read by process P_i
    - O_i (output): set of memory locations written by process P_i
  - BERNSTEIN's conditions

$$I_1 \cap O_2 = \varnothing \qquad I_2 \cap O_1 = \varnothing \qquad O_1 \cap O_2 = \varnothing$$

  - example

$$P_1\!: a \leftarrow x + y \qquad\qquad P_2\!: b \leftarrow x + z$$

$I_1 = \{x, y\}$, $O_1 = \{a\}$, $I_2 = \{x, z\}$, $O_2 = \{b\}$ ➔ all conditions fulfilled

Terms and Definitions

- dependence analysis (cont'd)
  - further example

P_1: $a \leftarrow x + y$ P_2: $b \leftarrow a + b$

$I_1 = \{x, y\}$, $O_1 = \{a\}$, $I_2 = \{a, b\}$, $O_2 = \{b\}$ ➔ $\boldsymbol{I_2 \cap O_1 \neq \emptyset}$

  - BERNSTEIN's conditions help to identify instruction-level parallelism or coarser parallelism (loops, e. g.)
  - hence, sometimes dependencies within loops can be solved
  - example: two loops with dependencies – which to be solved

loop *A*:

```
for i ← 2 to 100 do
    a[i] ← a[i−1] + 4
od
```

loop *B*:

```
for i ← 2 to 100 do
    a[i] ← a[i−2] + 4
od
```

Terms and Definitions

- dependence analysis (cont'd)
  - expansion of loop *B*

```
a[2] ← a[0] + 4        a[3] ← a[1] + 4
a[4] ← a[2] + 4        a[5] ← a[3] + 4
a[6] ← a[4] + 4        a[7] ← a[5] + 4
```

  - hence, *a*[3] can only be computed after *a*[1], *a*[4] after *a*[2], …
    ➔ computation can be split into two independent loops

```
a[0] ← …                          a[1] ← …
for i ← 1 to 50 do                for i ← 1 to 50 do
    j ← 2*i                           j ← 2*i + 1
    a[j] ← a[j–2] + 4                 a[j] ← a[j–2] + 4
od                                od
```

  - many other techniques for recognising / creating parallelism exist

謝謝