



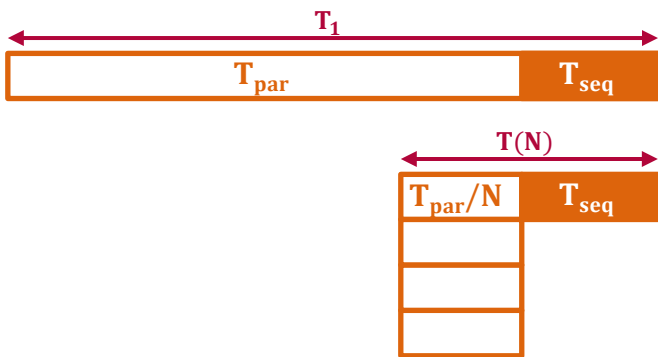
Parallel Programming and Heterogeneous Computing

Shared-Nothing Basics

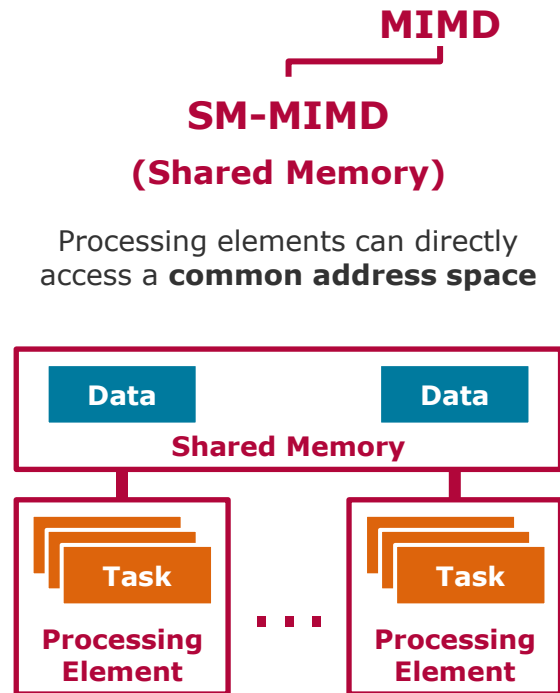
Max Plauth, Sven Köhler, Felix Eberhardt, Lukas Wenzel and Andreas Polze
Operating Systems and Middleware Group

Recap

Anatomy of a Workload / MIMD Hardware Taxonomy



$$T(N) = \frac{T_{par}}{N} + T_{seq}$$
$$= \left(\frac{P}{N} + (1 - P) \right) \cdot T_1$$



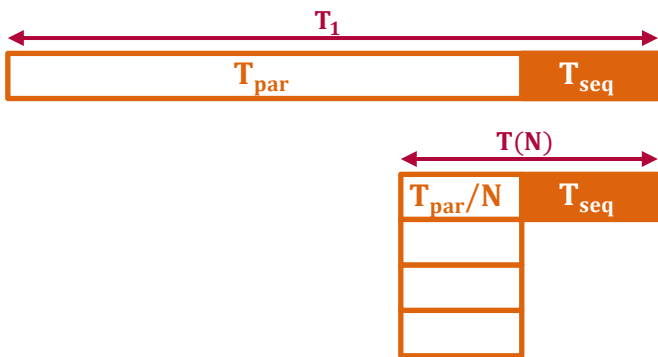
ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

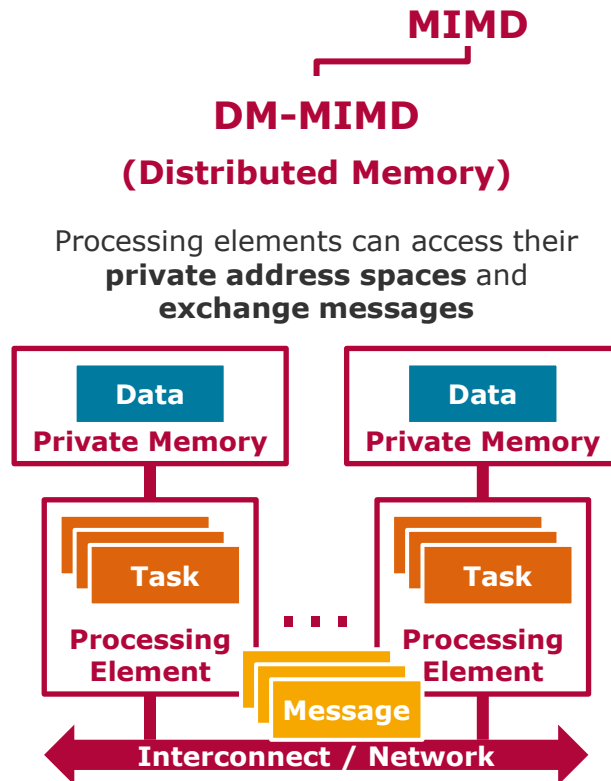
Chart 2

Recap

Anatomy of a Workload / MIMD Hardware Taxonomy



$$T(N) = \frac{T_{par}}{N} + T_{seq}$$
$$= \left(\frac{P}{N} + (1 - P) \right) \cdot T_1$$



ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

Chart 3

Parallel Machine Models

Early parallel machine models are abstractions of shared memory machines:

- Parallel Random Access Machine Model (**PRAM**)
 - Used in many variations in terms of memory access and execution modalities

Later models capture properties of distributed memory machines:

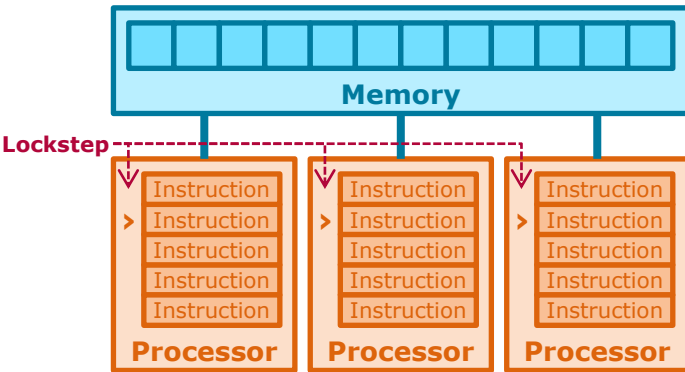
- Bulk Synchronous Parallel Model (**BSP**)
- **LogP** Model

Recent models focus on memory hierarchy:

- Universal Parallel Memory Hierarchy (UPMH)
- Memory LogP, Log_nP

Parallel Random Access Machine (PRAM)

Natural extension of the Random Access Machine (RAM) model:



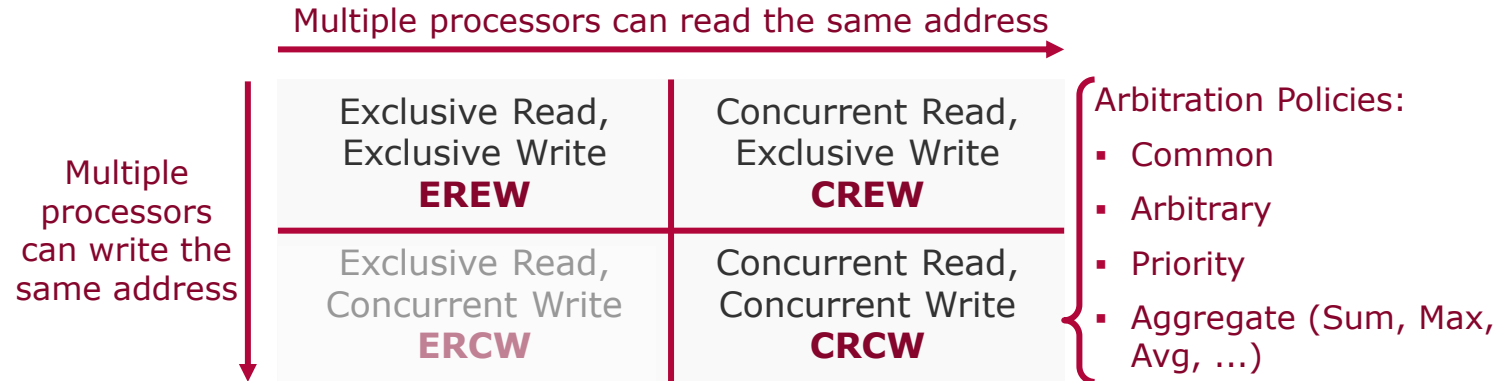
- **Arbitrary amount of memory**
- **Constant memory access latency:**
Processor can read or write a single memory cell per cycle.
- **Arbitrary number of processors**
- **Lockstep execution:**
Each processor executes any instruction in a single cycle of a shared clock.
- **No synchronization primitives:** not strictly required by algorithms because of lockstep execution

Parallel Random Access Machine (PRAM)

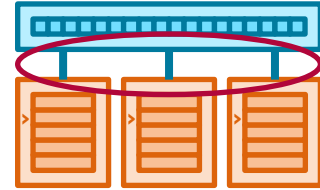
Memory Access Modalities

Multiple accesses to different addresses can always proceed in the same cycle.
 ~ Infinite memory bandwidth

Multiple accesses to the same address may cause varying behavior:



Algorithms accessing the same address from multiple processors in exclusive mode are considered incorrect!



Parallel Random Access Machine (PRAM)

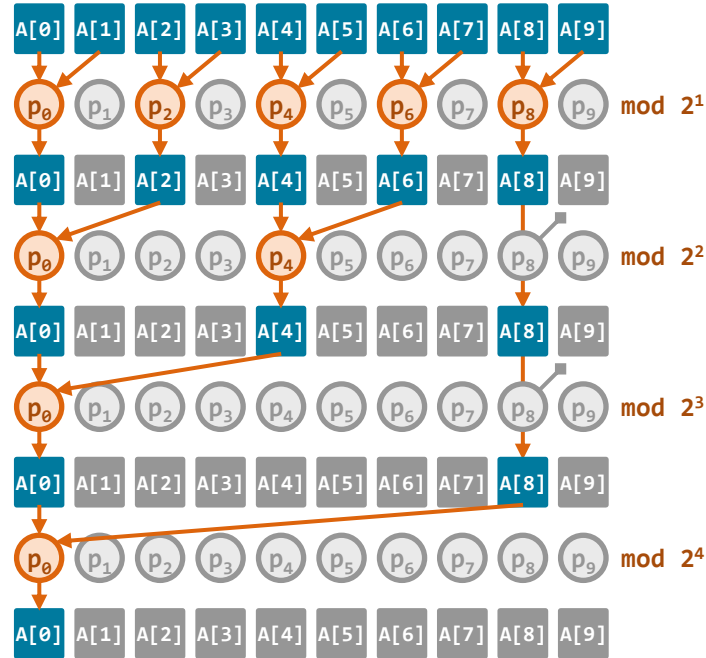
Example: Parallel Sum

- Sum elements in array $A[N]$ using a PRAM with N processors
- Time complexity $\mathcal{O}(\log_2 N)$
- Correctness relies on lockstep execution
 - APRAM variant discards the lockstep criterion
 - Would require a barrier after each addition

```

for l in 1 to ceil(log2(N)) {
  if ((p % 2^l) == 0 &&
      (p + 2^(l-1)) < N) {
    A[p] += A[p + 2^(l-1)];
  }
}
    
```

p - Processor ID between 0 and $N-1$



ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

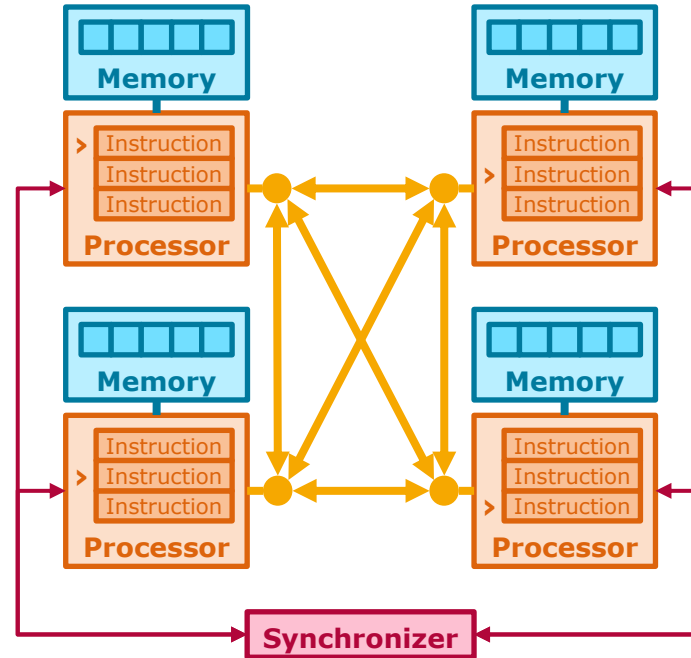
Chart 7

[Valiant1990] Bulk Synchronous Parallel Model (BSP)

- Intended to bridge the gap between computational and network models

Models a distributed system:

- Processors use **local memory** and **execute instructions asynchronously**
- Processors are connected through a **communication network**
- Processors share a common **synchronization mechanism**



ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

Chart 8

[Valiant1990] Bulk Synchronous Parallel Model (BSP)

Algorithms are divided into three repeating phases, forming multiple supersteps:

1. **Local Computation**
2. **Global Communication**
3. **Synchronization**

Superstep duration varies at runtime depending on computational and communication load.

Performance estimates using the following parameters:

Computation time: $t_W = \max\{w_i\}$

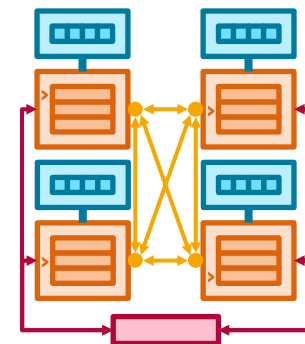
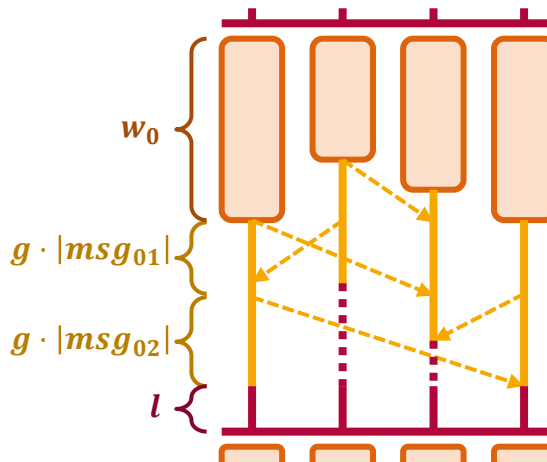
Communication time: $t_C = g \cdot m \cdot h$

$g \sim$ message bandwidth

$m = \max\{|msg_k|\} \sim$ message size

$h = \max\{\#in_i, \#out_i\} \sim$ communication pattern

Synchronization overhead: $t_S = l$



ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

Chart 9

[Valiant1990] Bulk Synchronous Parallel Model (BSP)

BSP exhibits many important characteristics of real distributed systems:

- **Communication is not free**

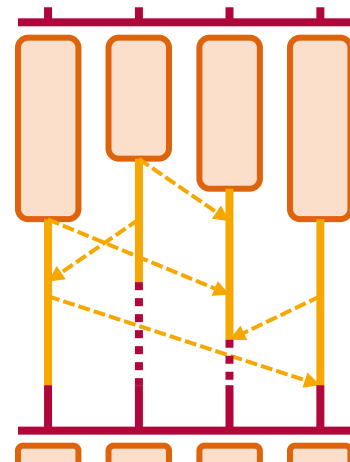
Interaction between distributed nodes comes at a cost t_c

- **Surface-to-Volume effect**

Excessive subdivision of workloads increases communication time t_c without sufficiently decreasing computation time t_w

- **Unbalanced work distribution**

Superstep period is determined by longest running individual computation $\max\{w_i\}$, idle time grows with $\max\{w_i\} - \min\{w_i\}$



**ParProg 2020 D1
Shared-Nothing
Basics**

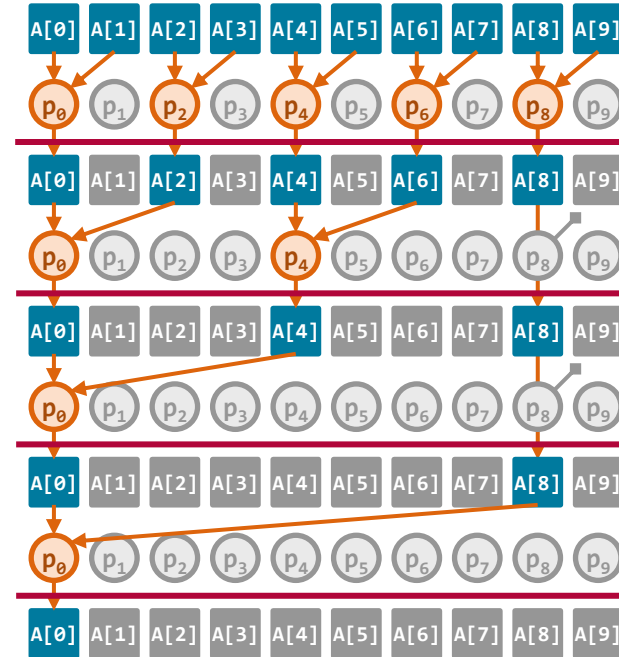
Lukas Wenzel

Chart 10

[Valiant1990] Bulk Synchronous Parallel Model (BSP)

Parallel Sum algorithm on BSP:

- Synchronization after every addition
- Excessive ratio between communication and computation



ParProg 2020 D1
Shared-Nothing
Basics

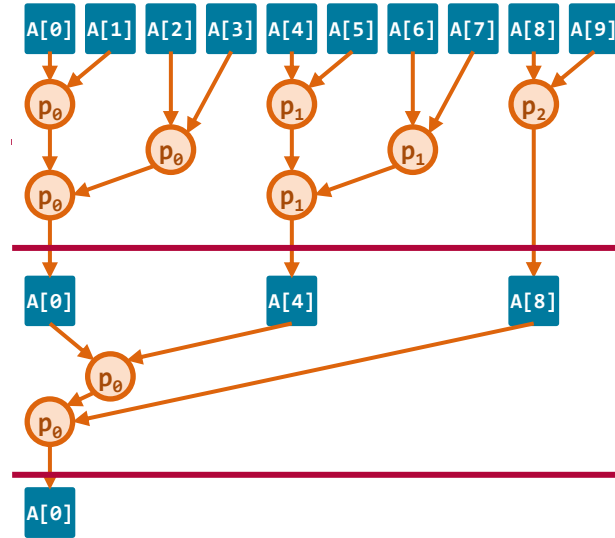
Lukas Wenzel

Chart 11.1

[Valiant1990] Bulk Synchronous Parallel Model (BSP)

Parallel Sum algorithm on BSP:

- Synchronization after every addition
- Excessive ratio between communication and computation
- **Perform additions in larger blocks using fewer processors**



[Culler1993] LogP Model

Similar to BSP architecture, but omits global synchronization in favor of individual synchronization.

- Processors use **local memory** and **execute instructions asynchronously**
- Processors **communicate and synchronize** through a **network**

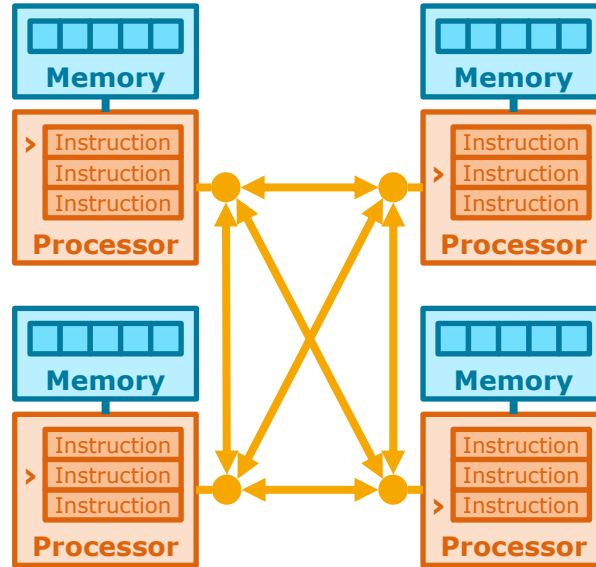
Parameters:

P – #processors

g – **gap** (time in cycles between messages from / to a single processor)

o – **overhead** (time in cycles for send / receive operation)

l – **latency** (time in cycles between transmission and reception of a message)



ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

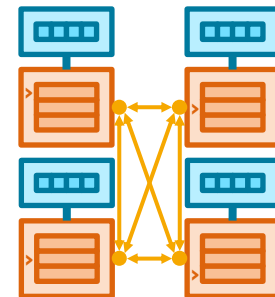
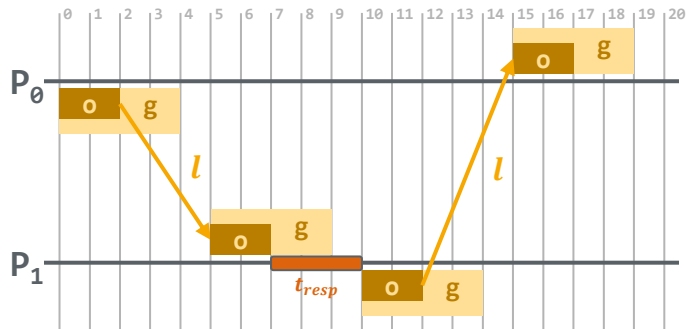
Chart 12

[Culler1993] LogP Model

LogP enables a fine-grained analysis of communication patterns.

Example: Request-Response sequence between two processors

- $P = 2$; $l = 3$; $g = 4$; $o = 2$; $t_{resp} = 3$
- $t_{total} = 2 \cdot l + 4 \cdot o + t_{resp} = 17$
- t_{total} is independent of g because processor bandwidth is not saturated by this workload



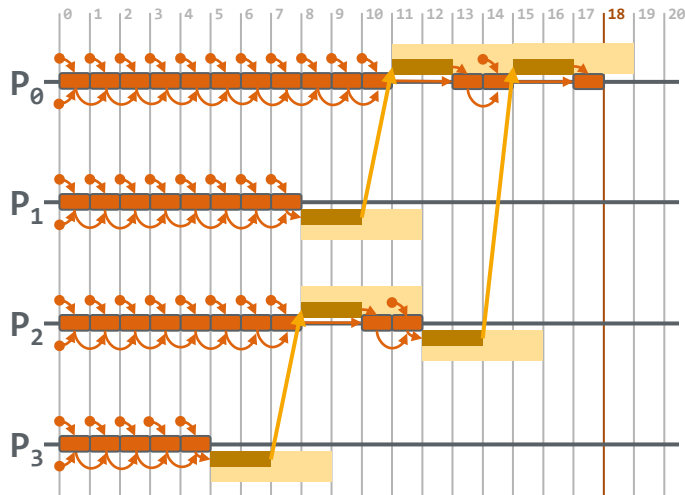
ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

Chart 13

Parallel Sum algorithm on LogP

- $P = 4$; $l = 1$; $g = 4$; $o = 2$; $t_{add} = 1$
- In **18 cycles**, the optimal algorithm on the given LogP parameterization can sum **38 values**
- Each processor performs local calculations for the longest possible time
- Find the latest cycle when each slave process must send results to its master, by tracing back communication times
- Each slave is associated with the master that has the latest message reception requirement



Parallel Machine Models

BSP and LogP allow abstract reasoning about parallel algorithms for DM-MIMD systems in general, without relying on characteristics of an actual system.

- Valuable for designing, analyzing and optimizing algorithms.

Optimizing a particular implementation of an algorithm usually benefits from knowledge of actual system characteristics.



Recap

DM-MIMD Hardware

Processing elements can access their **private address spaces** and **exchange messages**

Cluster: *Multiple independent machines connected through a network*

- **Compute** cluster: Speedup
- **Load Balancing** cluster: Throughput
- **High Availability** cluster: Dependability

All clusters are distributed systems, but only compute clusters intended for parallel workloads.

This lecture considers only compute clusters.

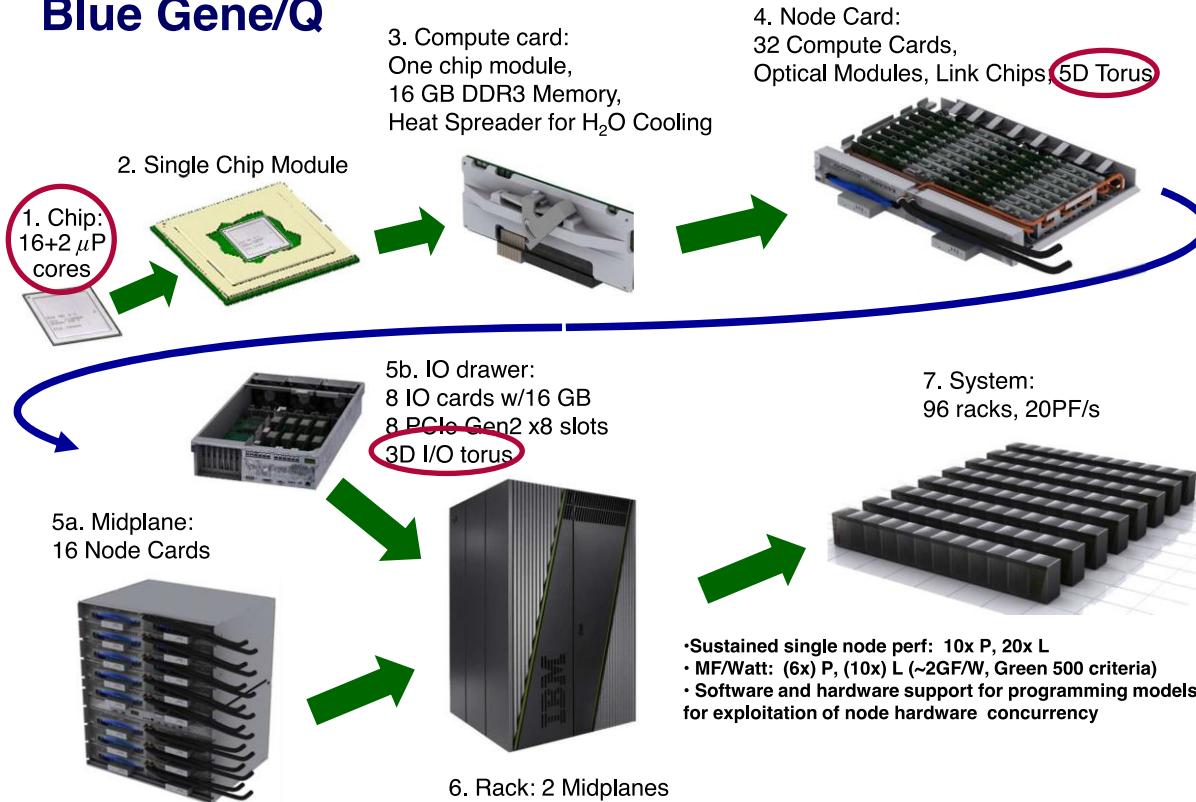
ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

Chart **16**

A Large Compute Cluster

Blue Gene/Q



- Sustained single node perf: 10x P, 20x L
- MF/Watt: (6x) P, (10x) L (~2GF/W, Green 500 criteria)
- Software and hardware support for programming models for exploitation of node hardware concurrency

ParProg 2020 D1 Shared-Nothing Basics

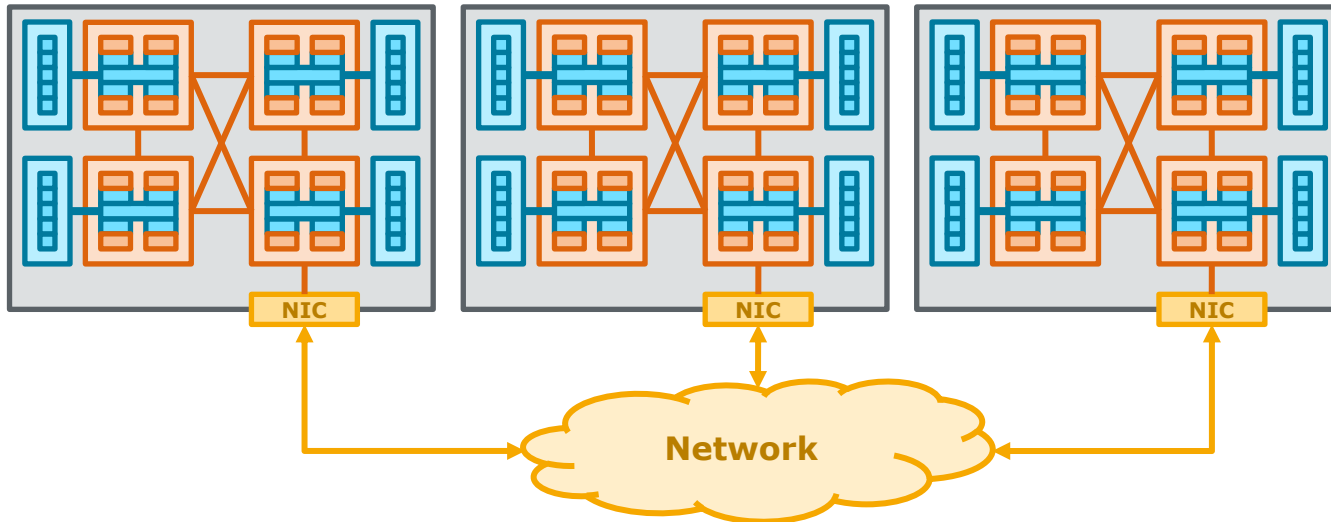
Lukas Wenzel

Chart 17

DM-MIMD Ssssystems

Nodes in a DM-MIMD system are usually SM-MIMD machines, to exploit multiple levels of scalability.

Node architecture has been discussed, but **what about network architecture?**



Network Components

Network Interface Controllers (NIC)

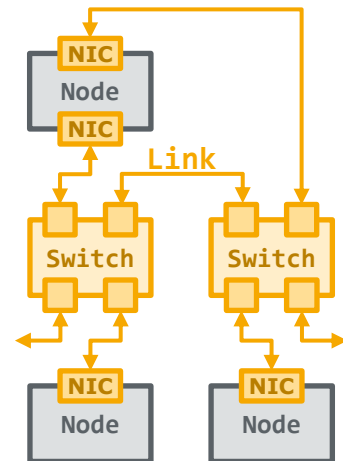
- = Peripheral devices attached to a node's IO subsystem, implement a network port
- Various IO-interconnect and DMA mechanisms
- May offer limited processing capability (e.g. for packet decoding, filtering, ...)

Switches

- = Independent components with multiple network ports, route messages between attached links

Links

- = physical media (e.g. optical fibers, copper wires, coaxial cables), connected in a specific topology



**ParProg 2020 D1
Shared-Nothing
Basics**

Lukas Wenzel

Chart 19

Excursion

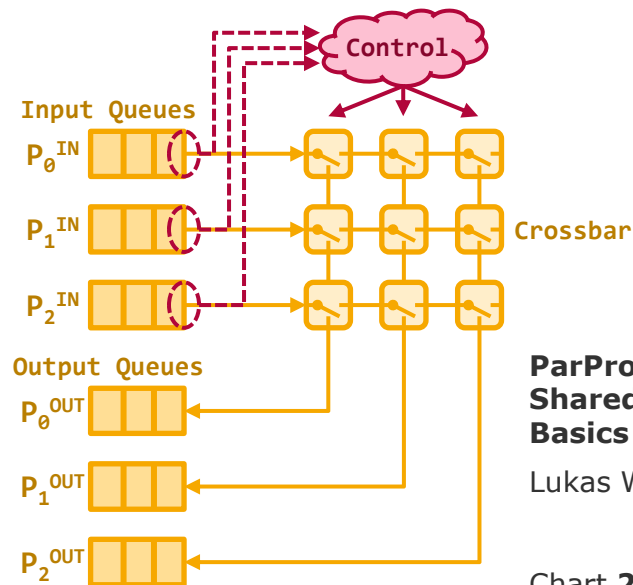
Network Switches

Forwarding packets between any pair of N ports requires implementation complexity $\mathcal{O}(N^2)$.

- Switches usually incorporate input and/or output queues as well as a crossbar between them

There are switch implementations in $\mathcal{O}(N \cdot \log(N))$.

- Often multilayer networks of (2,2)-switch primitives
- Can not connect any possible set of distinct port pairs at a time
- Sacrifice worst-case throughput for implementation efficiency



**ParProg 2020 D1
Shared-Nothing
Basics**

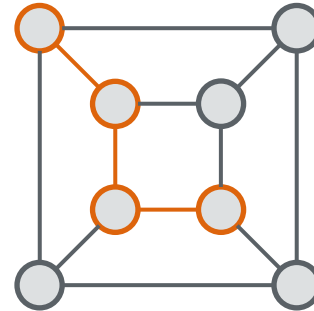
Lukas Wenzel

Chart 20

Network Topologies

Topologies are characterized by multiple metrics:

- **Diameter** \sim Latency
Maximum distance between any two nodes
- **Connectivity** \sim Resilience
Minimum number of removed edges to cause partition
- **Bisection Bandwidth** \sim Throughput
Transfer capacity across balanced network cuts
- **Cost** \sim Network complexity
Total number of edges
- **Degree** \sim Node complexity
Maximum number of edges per node
- **Link Bandwidth**



ParProg 2020 D1
Shared-Nothing
Basics

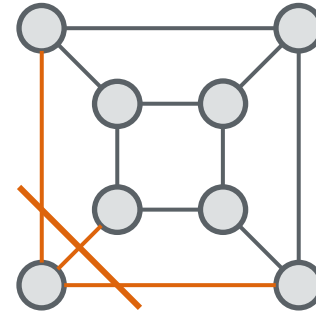
Lukas Wenzel

Chart 21.1

Network Topologies

Topologies are characterized by multiple metrics:

- **Diameter** \sim Latency
Maximum distance between any two nodes
- **Connectivity** \sim Resilience
Minimum number of removed edges to cause partition
- **Bisection Bandwidth** \sim Throughput
Transfer capacity across balanced network cuts
- **Cost** \sim Network complexity
Total number of edges
- **Degree** \sim Node complexity
Maximum number of edges per node
- **Link Bandwidth**



ParProg 2020 D1
Shared-Nothing
Basics

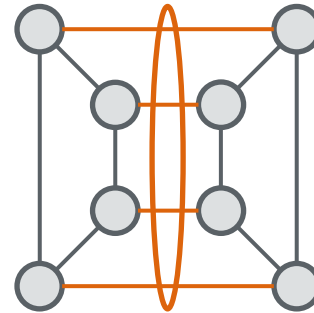
Lukas Wenzel

Chart 21.2

Network Topologies

Topologies are characterized by multiple metrics:

- **Diameter** \sim Latency
Maximum distance between any two nodes
- **Connectivity** \sim Resilience
Minimum number of removed edges to cause partition
- **Bisection Bandwidth** \sim Throughput
Transfer capacity across balanced network cuts
- **Cost** \sim Network complexity
Total number of edges
- **Degree** \sim Node complexity
Maximum number of edges per node
- **Link Bandwidth**



ParProg 2020 D1
Shared-Nothing
Basics

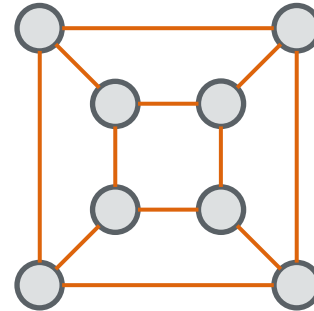
Lukas Wenzel

Chart 21.3

Network Topologies

Topologies are characterized by multiple metrics:

- **Diameter** \sim Latency
Maximum distance between any two nodes
- **Connectivity** \sim Resilience
Minimum number of removed edges to cause partition
- **Bisection Bandwidth** \sim Throughput
Transfer capacity across balanced network cuts
- **Cost** \sim Network complexity
Total number of edges
- **Degree** \sim Node complexity
Maximum number of edges per node
- **Link Bandwidth**



ParProg 2020 D1
Shared-Nothing
Basics

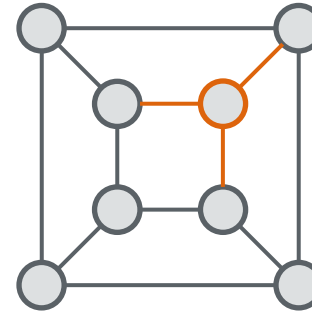
Lukas Wenzel

Chart 21.4

Network Topologies

Topologies are characterized by multiple metrics:

- **Diameter** \sim Latency
Maximum distance between any two nodes
- **Connectivity** \sim Resilience
Minimum number of removed edges to cause partition
- **Bisection Bandwidth** \sim Throughput
Transfer capacity across balanced network cuts
- **Cost** \sim Network complexity
Total number of edges
- **Degree** \sim Node complexity
Maximum number of edges per node
- **Link Bandwidth**



ParProg 2020 D1
Shared-Nothing
Basics

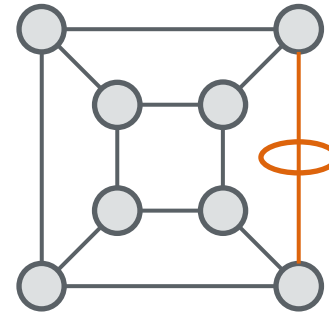
Lukas Wenzel

Chart 21.5

Network Topologies

Topologies are characterized by multiple metrics:

- **Diameter** \sim Latency
Maximum distance between any two nodes
- **Connectivity** \sim Resilience
Minimum number of removed edges to cause partition
- **Bisection Bandwidth** \sim Throughput
Transfer capacity across balanced network cuts
- **Cost** \sim Network complexity
Total number of edges
- **Degree** \sim Node complexity
Maximum number of edges per node
- **Link Bandwidth**



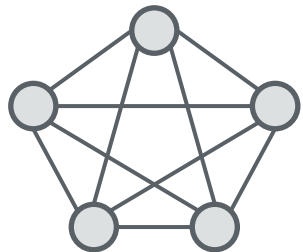
ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

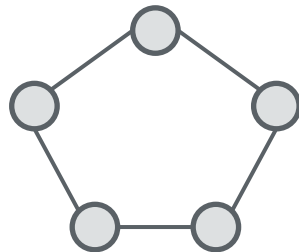
Chart 21.6

Network Topologies

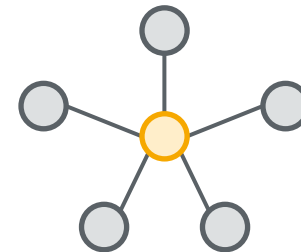
Fully Connected	
Diameter	1
Connectivity	$n - 1$
Cost	$\frac{n^2 - n}{2}$
Degree	$n - 1$



Ring	
Diameter	$\lfloor \frac{n}{2} \rfloor$
Connectivity	2
Cost	n
Degree	2



Star	
Diameter	2
Connectivity	1
Cost	n
Degree	1 n



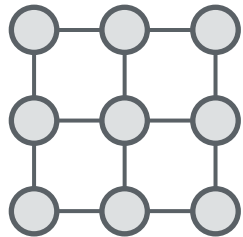
ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

Chart 22

Network Topologies

d-Mesh	
Diameter	$d \cdot (k - 1)$ $= d \cdot (\sqrt[d]{n} - 1)$
Connectivity	d
Cost	$d \cdot k^{d-1} \cdot (k - 1)$ $= d \cdot (n - n^{d-1/d})$
Degree	$2 \cdot d$

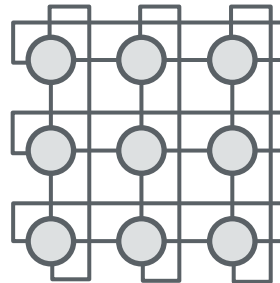


$$d = 2$$

$$k = 3$$

$$n = k^d = 9$$

d-Torus	
Diameter	$\left\lceil \frac{d \cdot (k - 1)}{2} \right\rceil$ $= \left\lceil \frac{d \cdot (\sqrt[d]{n} - 1)}{2} \right\rceil$
Connectivity	$2 \cdot d$
Cost	$d \cdot k^d = d \cdot n$
Degree	$2 \cdot d$



$$d = 2$$

$$k = 3$$

$$n = k^d = 9$$

**ParProg 2020 D1
Shared-Nothing
Basics**

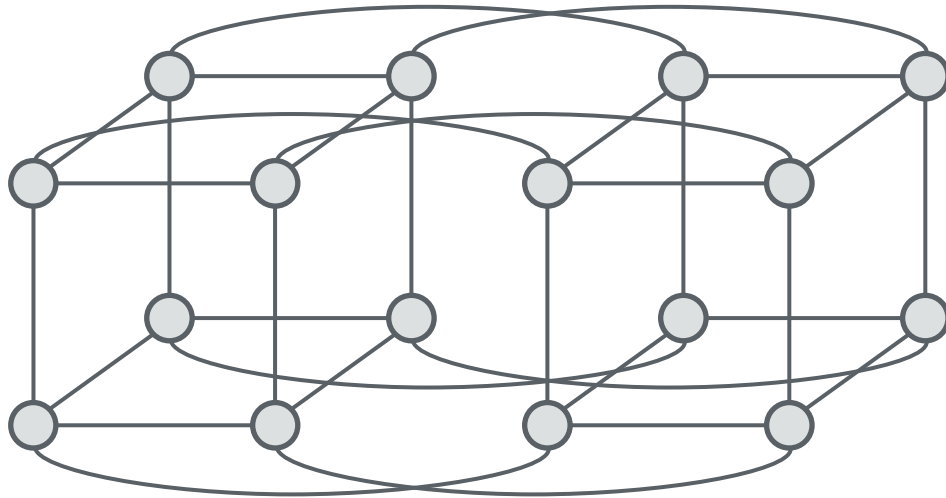
Lukas Wenzel

Chart 23

Network Topologies

Hypercubes

= **d-Mesh with $k = 2$**



e.g. 4D-Hypercube = 4-Mesh with $k=2$

ParProg 2020 D1
Shared-Nothing
Basics

Lukas Wenzel

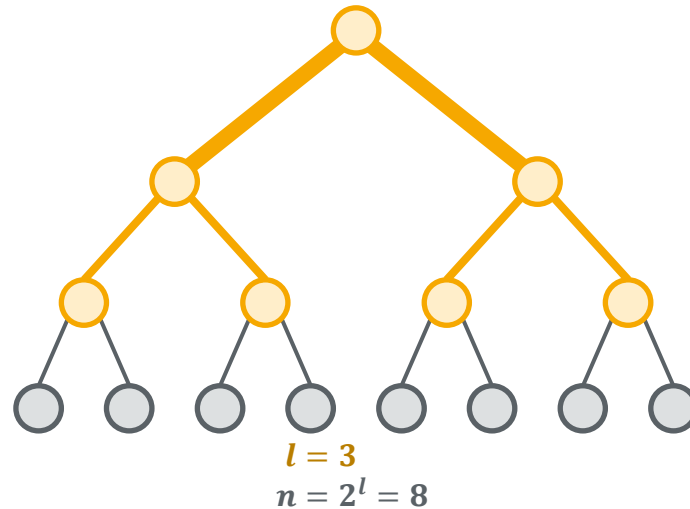
Chart 24

Network Topologies

Fat Tree of Depth l

= Binary l -level switch hierarchy,
 where uplink bandwidth equals sum of downlink bandwidths

Fat Tree	
Diameter	$2 \cdot l = 2 \cdot \log_2(n)$
Connectivity	1
Cost	$2^{l+1} - 2 = 2 \cdot n - 2$
Cost (Bandwidth adjusted)	$l \cdot 2^l = n \cdot \log_2(n)$
Degree	1 3



ParProg 2020 D1
 Shared-Nothing
 Basics

Lukas Wenzel

Chart 25

And now for a break and
a cup of Dian Hong Gushu.



*or beverage of your choice