Concepts and Models of Parallel and Data-centric Programming

BSP I

Lecture, Summer 2020

Dr. Christian Terboven <terboven@itc.rwth-aachen.de>

Outline

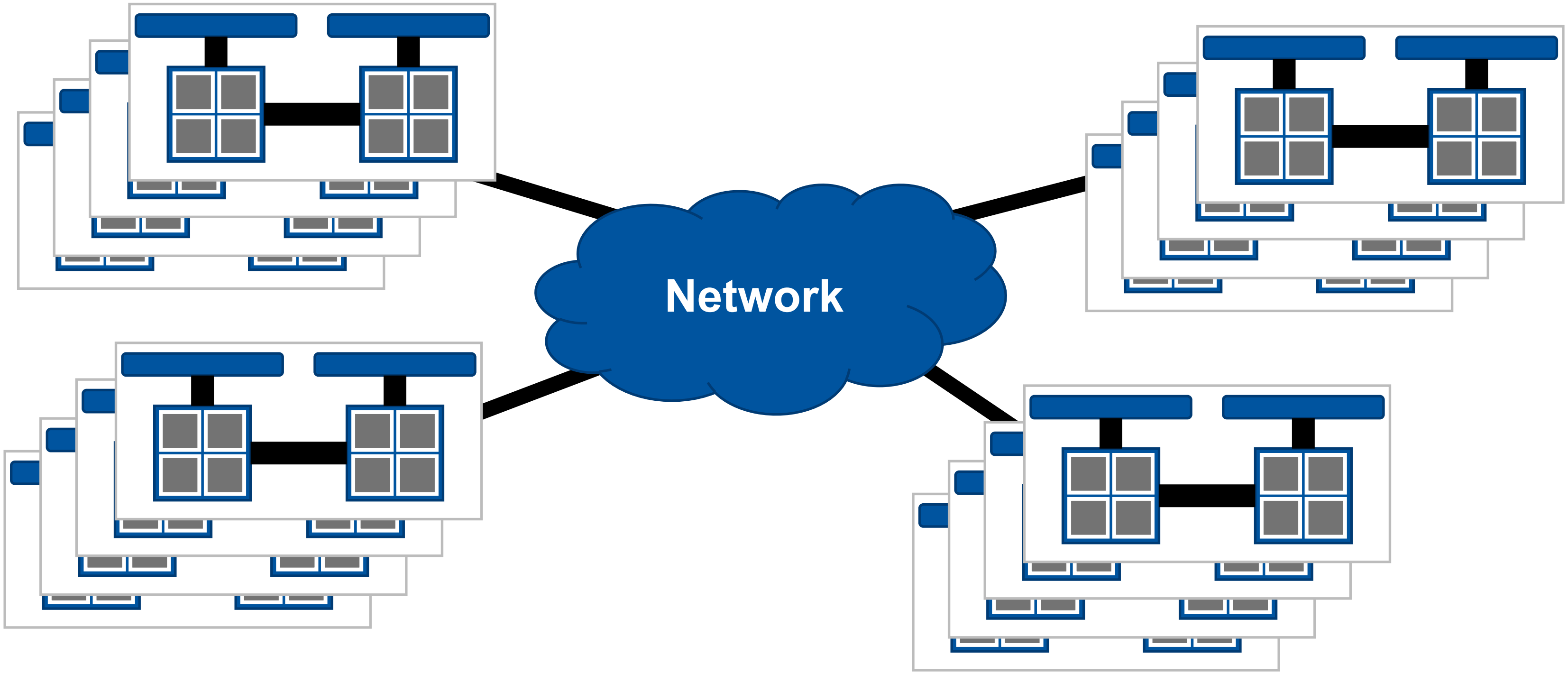
0. Organization
1. Foundations
2. Shared Memory
3. GPU Programming
4. **Bulk-Synchronous Parallelism**
5. Message Passing
6. Distributed Shared Memory
7. Parallel Algorithms
8. Parallel I/O
9. MapReduce
10. Apache Spark

a. Motivation
b. BSP Computer
c. BSP Programming Model
d. BSP Cost Model
e. Bulk Library

Motivation

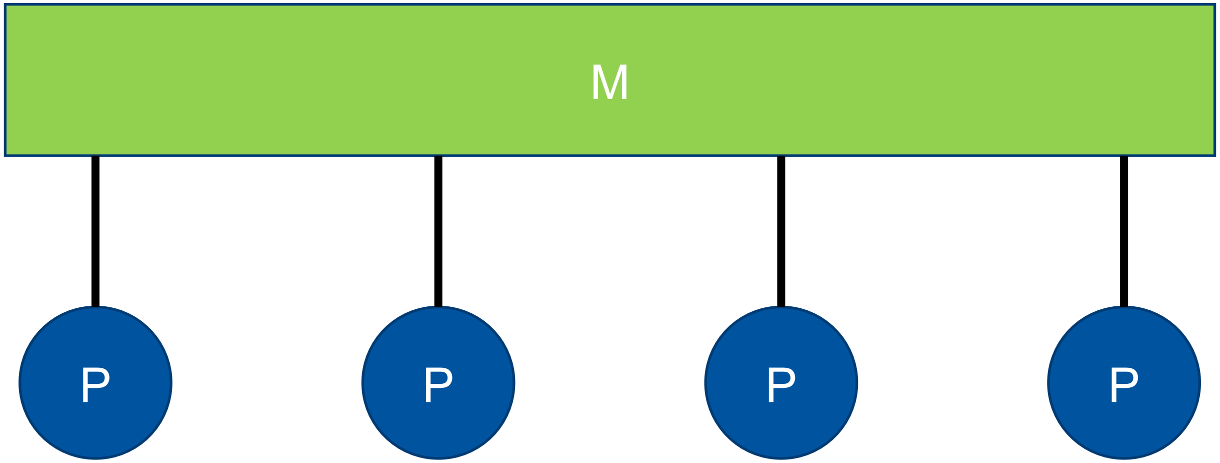
Parallel Architectures

- HPC market is at large dominated by distributed memory *multicomputers*: *clusters* and specialised *supercomputers*
- Nodes have no direct access to other nodes' memory and run a separate copy of the (possibly stripped down) OS



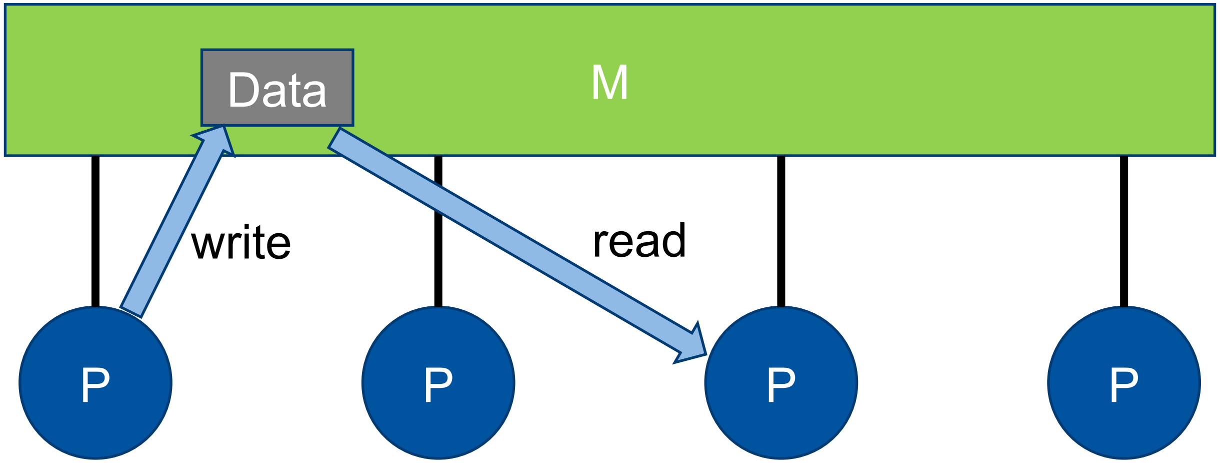
Parallel Architectures

- Shared Memory
  - All processing elements (P) have direct access to the main memory block (M)



Parallel Architectures

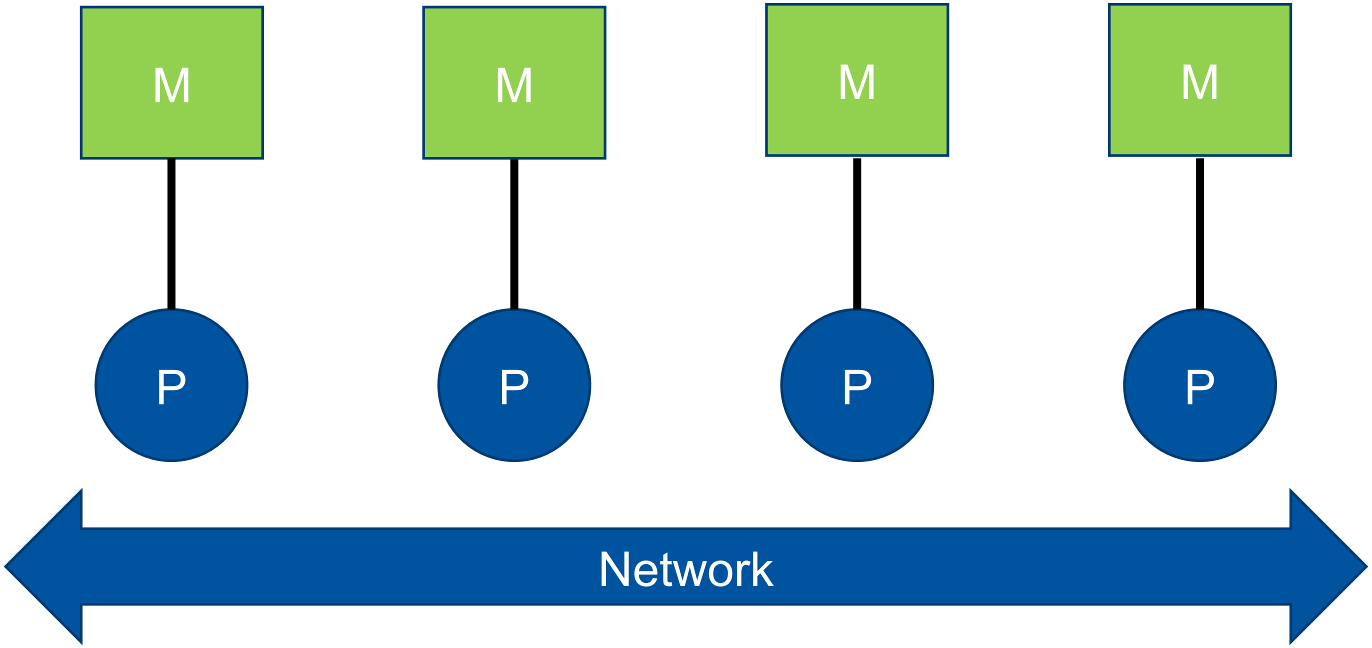
- Shared Memory
  - All processing elements (P) have direct access to the main memory block (M)



  - Data exchange is achieved through read/write operations on shared variables located in the global address space

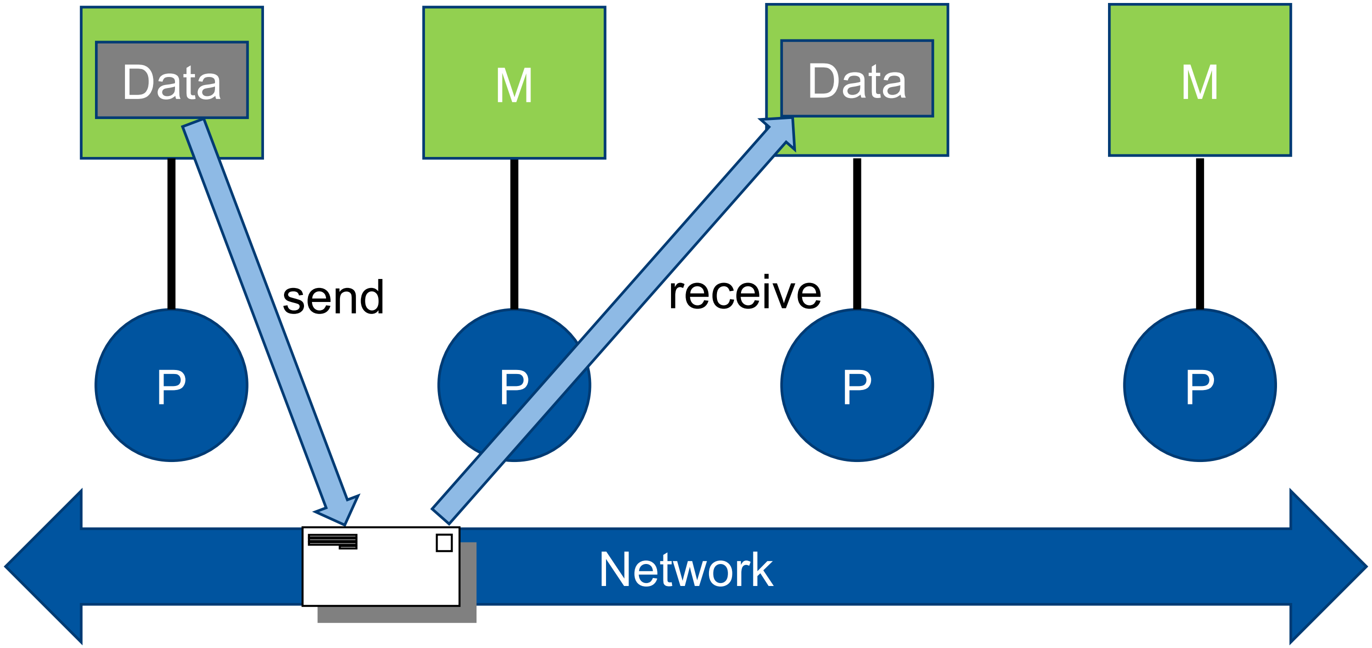
Parallel Architectures

- Distributed Memory
  - Each processing element (P) has its separate main memory block (M)



Parallel Architectures – Two-Sided Communication

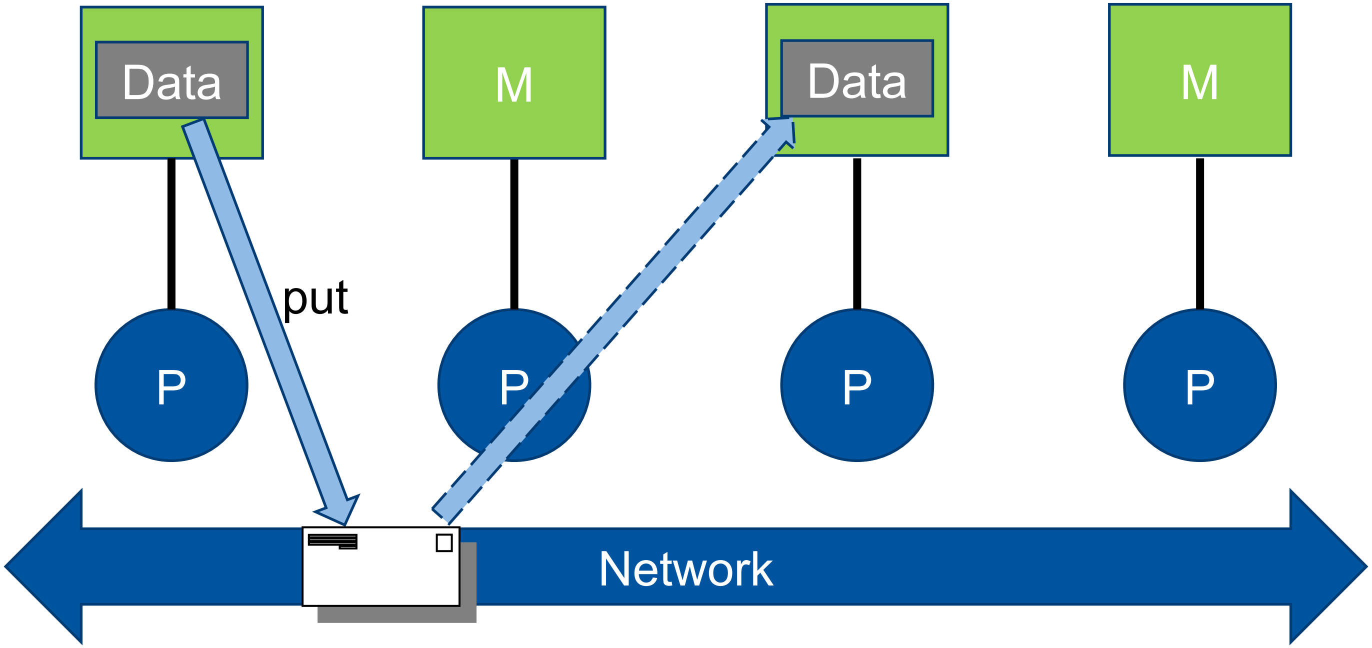
- Distributed Memory
  - Each processing element (P) has its separate main memory block (M)



  - Data exchange is achieved through **message** passing over the network
  - **Both** processes actively call send and receive function

Parallel Architectures – One-Sided Communication

- Distributed Memory
  - Each processing element (P) has its separate main memory block (M)



  - Alternative: Remote Direct Memory Access (RDMA)
  - Processes can **directly** access (put / get) mem. location of a remote process
    - Remote process does not have to call a matching function

Parallel Architectures

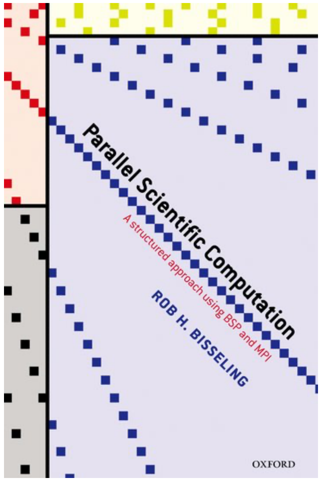
- Distributed Memory
  - Each processing element (P) has its separate main memory block (M)
  - Data exchange is achieved through message passing over the network
  - Message passing could be either explicit (MPI) or implicit (BSP, PGAS)
  - Programs typically implemented as a set of OS entities with own (virtual) address spaces – *processes*
  - No shared variables
    - Usually no data races – but sometimes the opposite: missing data

Writing Programs for Distributed Memory Machines

- How do we write programs for such an architecture?
- **Classical approach: MPI (Message Passing Interface)** → HPC lecture
  - Defines an interface for sending messages between processors
  - Point-to-Point communication, Collective communication
- **More abstract approach: BSP (Bulk-Synchronous Parallel) programs**
  - Defines a generic structure of a parallel program (so called “supersteps”)
  - Program resp. algorithm has to be adapted to the BSP programming model
  - Provides a cost model to reason about performance
  - In practice: Often implemented on top of MPI

Literature

- Original paper on BSP: Valiant, L.G.. *A Bridging Model for Parallel Computation*. Communications of the ACM 33, 8 (August 1990), 103-111
  - http://doi.acm.org/10.1145/79173.79181
- Parallel Algorithms Lecture (Rob Bisseling, Utrecht University)
  - http://www.staff.science.uu.nl/~bisse101/Education/PA/pa.html
- Rob H. Bisseling: *Parallel Scientific Computation – A Structured Approach using BSP and MPI*, 2004, Oxford University Press
- Buurlage, J. W., Bannink, T., Bisseling, R. H.. *Bulk: A Modern C++ Interface for Bulk-Synchronous Parallel Programs*. In EuroPar 2018 (pp. 519-532). Springer
  - https://doi.org/10.1007/978-3-319-96983-1_37

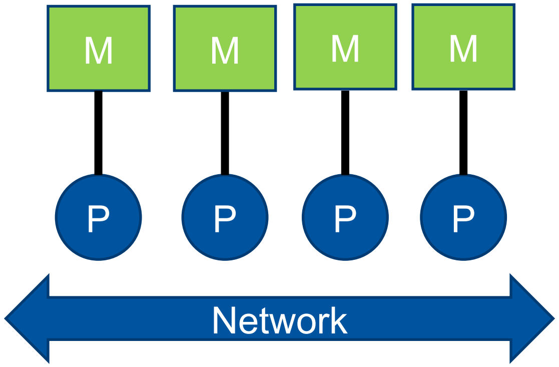

Parallel Scientific Computation
A structured approach using BSP and MPI
ROB H. BISSELING
OXFORD

BSP Computer

BSP Model Components

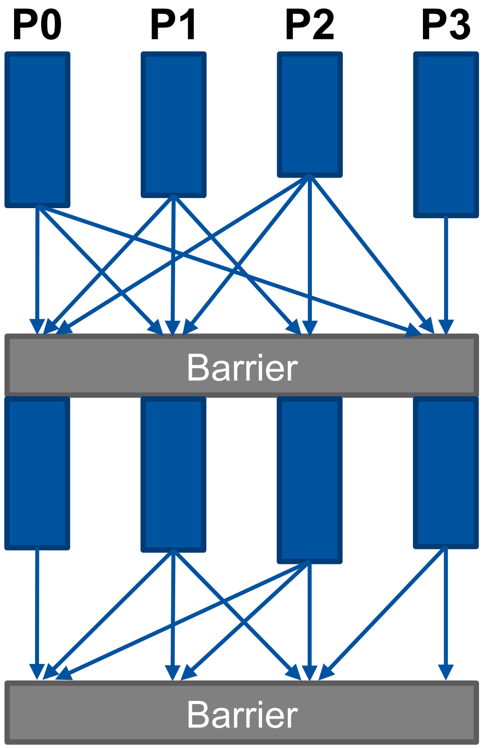
BSP Computer

(Distributed Memory Computer)

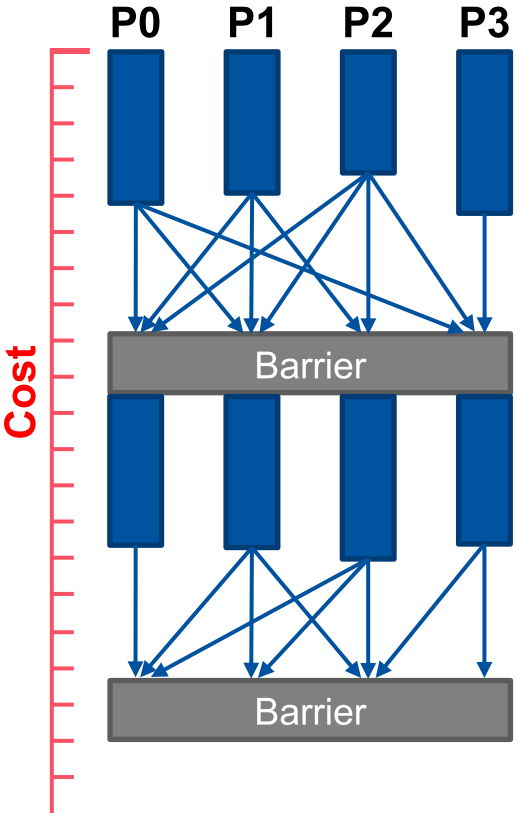


Programming Model

(Algorithmic Framework)

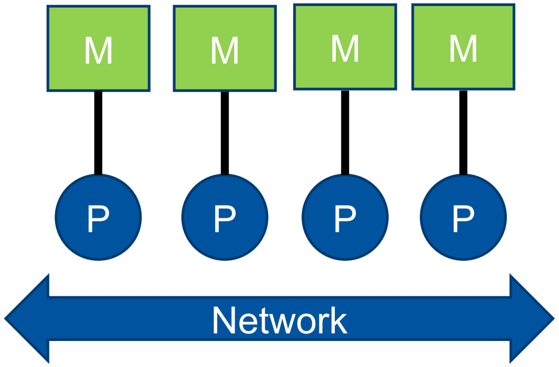


Cost Model



BSP Computer

- BSP computer is a distributed memory computer
  - Consists of processors with own memory
  - Local memory accesses are fast, remote memory accesses slower
- Communication network treated as black box
  - Algorithm designer does not have to care about network details
- BSP algorithms are portable
  - Run on many different parallel computers (even on shared memory computers)


M
M
M
M
P
P
P
P
Network