



Chapter 10

Parallel Algorithms

Algorithm Theory
WS 2018/19

Fabian Kuhn

Sequential Algorithms

Classical Algorithm Design:

- One machine/CPU/process/... doing a computation

RAM (Random Access Machine):

- Basic standard model
- Unit cost basic operations
- Unit cost access to all memory cells

Sequential Algorithm / Program:

- Sequence of operations
(executed one after the other)

Parallel and Distributed Algorithms

Today's computers/systems are not sequential:

- Even cell phones have several cores
- Future systems will be highly parallel on many levels
- This also requires appropriate algorithmic techniques

Goals, Scenarios, Challenges:

- Exploit parallelism to speed up computations
- Shared resources such as memory, bandwidth, ...
- Increase reliability by adding redundancy
- Solve tasks in inherently decentralized environments
- ...

- Many different forms
- Processors/computers/machines/... communicate and share data through
 - Shared memory or message passing
- Computation and communication can be
 - Synchronous or asynchronous
- Many possible **topologies** for message passing
- Depending on system, various **types of faults**

Algorithmic and theoretical challenges:

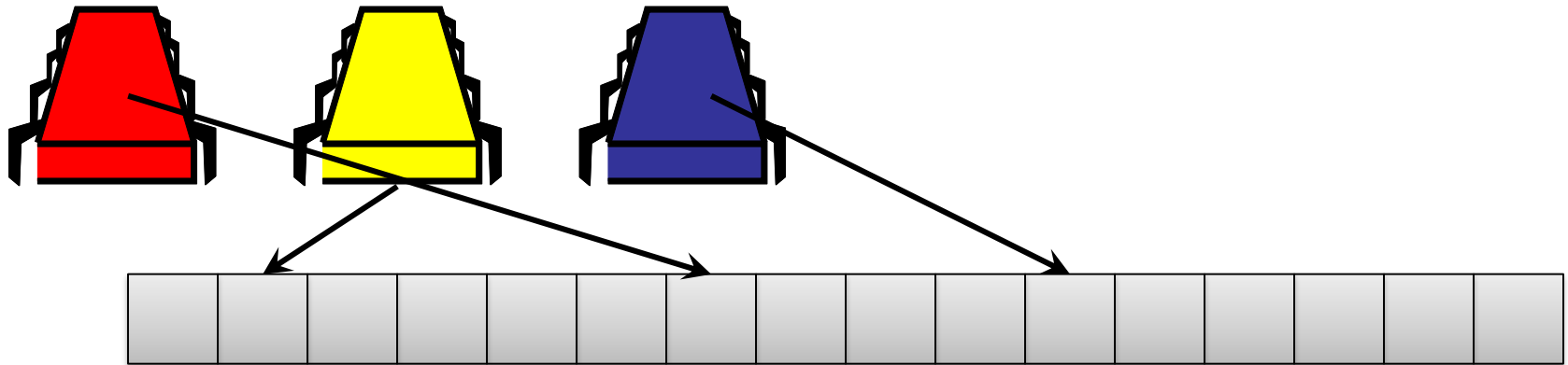
- How to parallelize computations
- Scheduling (which machine does what)
- Load balancing
- Fault tolerance
- Coordination / consistency
- Decentralized state
- Asynchrony
- Bounded bandwidth / properties of comm. channels
- ...

Models

- A large variety of models, e.g.:
- **PRAM** (Parallel Random Access Machine)
 - Classical model for parallel computations
- **Shared Memory**
 - Classical model to study coordination / agreement problems, distributed data structures, ...
- **Message Passing** (fully connected topology)
 - Closely related to shared memory models
- Message Passing in **Networks**
 - Decentralized computations, large parallel machines, comes in various flavors...
- Computations in large clusters of powerful individual machines: Massively Parallel Computations (MPC)

PRAM

- Parallel version of RAM model
- p processors, shared random access memory



- Basic operations / access to shared memory cost 1
- Processor operations are synchronized
- **Focus on parallelizing computation** rather than cost of communication, locality, faults, asynchrony, ...

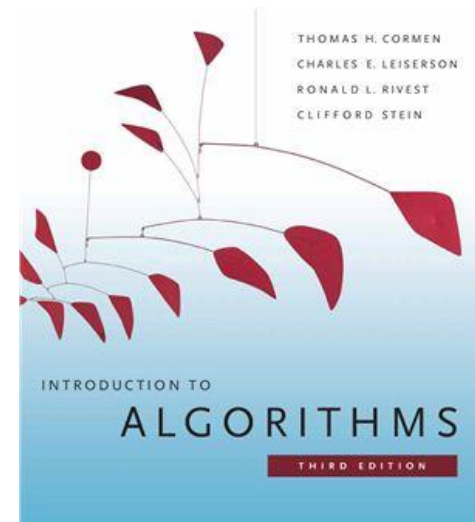
Other Parallel Models

- **Message passing:** Fully connected network, local memory and information exchange using messages
- **Dynamic Multithreaded Algorithms:** Simple parallel programming paradigm
 - E.g., used in Cormen, Leiserson, Rivest, Stein (CLRS)

```

FIB( $n$ )
1  if  $n < 2$ 
2    then return  $n$ 
3   $x \leftarrow$  spawn FIB( $n - 1$ )
4   $y \leftarrow$  spawn FIB( $n - 2$ )
5  sync
6  return ( $x + y$ )

```



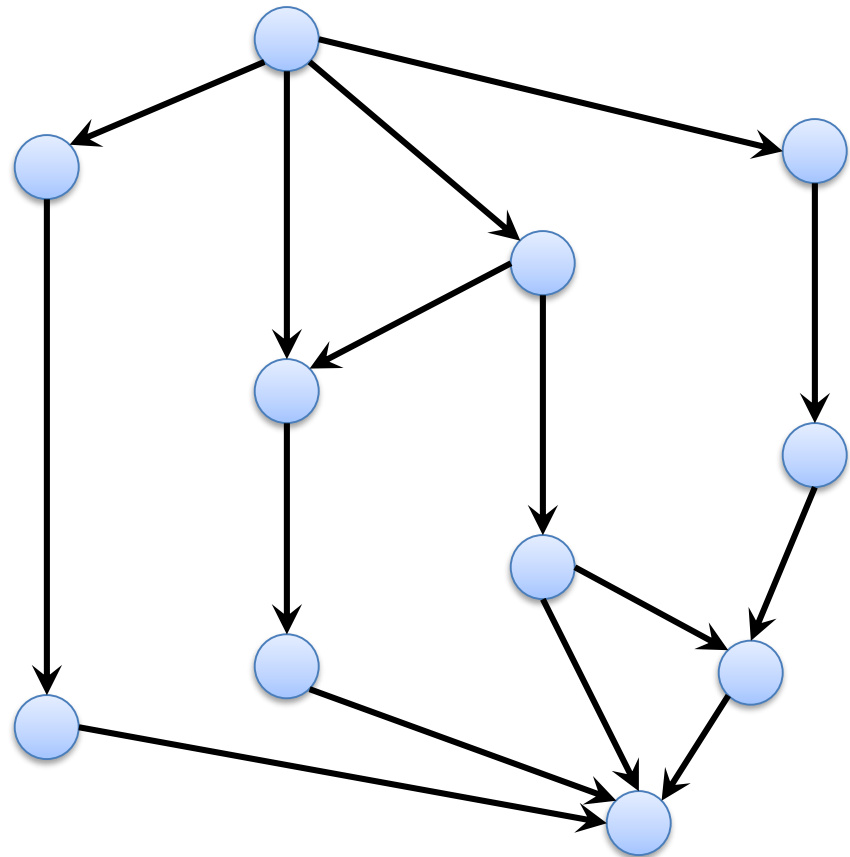
Sequential Computation:

- Sequence of operations



Parallel Computation:

- Directed Acyclic Graph (DAG)



Parallel Computations

T_p : time to perform comp. with p procs

- T_1 : **work** (total # operations)
 - Time when doing the computation sequentially
- T_∞ : **critical path / span**
 - Time when parallelizing as much as possible

• **Lower Bounds:**

$$T_p \geq \frac{T_1}{p}, \quad T_p \geq T_\infty$$

