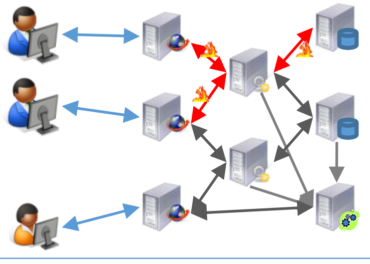


Verteilte Systeme/ Distributed Systems

Artur Andrzejak

1

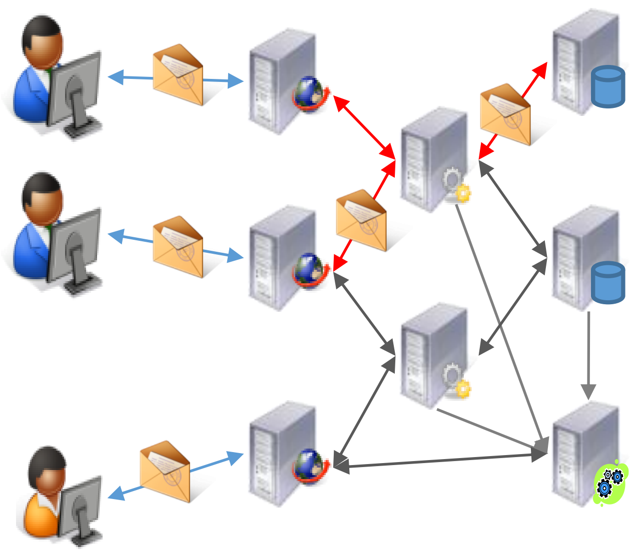
Motivation and Introduction

Distributed System (DS) - Definitions

- A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."

  Leslie Lamport

- [**Col**] A distributed system is one in which components..
  - are located at networked computers
  - communicate and coordinate <u>only by passing messages</u>



More Definitions

- A distributed computing system
  - consists of multiple autonomous processors
  - that do not share primary memory
  - but cooperate by sending messages over a communication network

Henri E. Bal

- Consequences
  - concurrency of actions
  - lack of a global clock
  - independent failures

- A lot of material in this lecture is devoted to dealing with these issues!

Common Challenges in Distributed Systems

- **Higher complexity of software**
  - Algorithms, implementation and debugging much more involved than in „serialized“ software
  - A BIG problem now, especially with multi-core CPUs
- **Dependency on the underlying network**
  - Differences in message transmission times
  - Non-deterministic phenomena, e.g. order of message arrival
- **Higher failure rate, lower reliability**
  - My internet connection at home (cable) malfunctions at least once a month
    - A generic laptop breaks down every 2-4 years

Other Challenges in Distributed Systems

1. **Security**
   - Privacy
   - Authentication
   - Availability
2. **Scalability**
   - e.g. IP addresses become scarce: in IPv4, we have only 2^{32} ~ $4*10^9$ IP addresses
   - Ongoing effort: from 32 to 128 bits in IPv6
3. **Heterogeneity of …**
   - network infrastructure
   - computer hard- and software (e.g., operating systems)
   - data representation in protocols
4. **Concurrency**
   - Avoidance of deadlocks and race conditions

Why are Distributed Systems Needed at all?

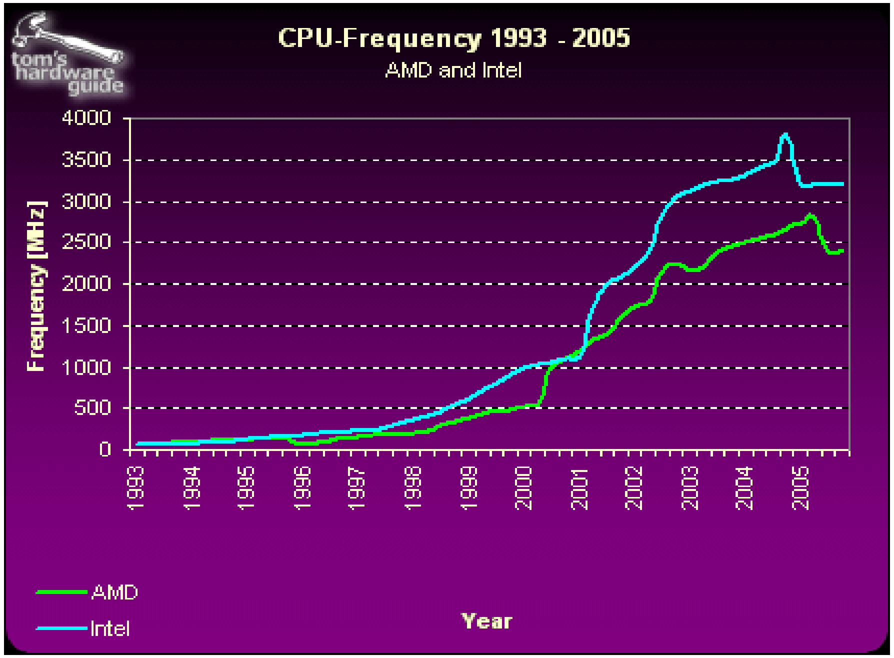
- [Col] “Sharing of resources is a main motivation for constructing distributed systems”
- „Sharing of resources“ has two reasons/motivations:
  - R1. Distributing computations over several cores/CPUs
  - R2. Information/data sharing and communication

- Reason 1 leads to distributed / parallel computing
  - Related to scientific computing / number crunching and non-consumer applications (i.e. science, simulations)
- Reason 2 is related to end-users/“business”
  - Most consumers buy computers for data/information exchange (email, WWW), not for “compute jobs”

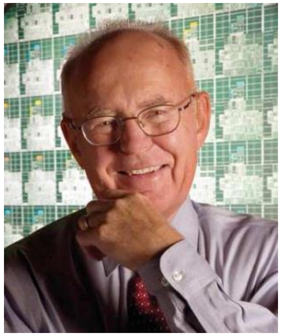
Reason 1: Why distributed/parallel computing?

- **Cumulative power**: distributed systems consisting of collections of microcomputers may have processing powers that no single computer will ever achieve
- **Economics**: collections of microprocessors offer a better price/performance ratio than mainframes
  - Why is Google using (hundred of thousands) of commodity-like PCs?
- **CPU architectures**: we experience a shift in CPU-architectures from “one core, higher frequency”, to “many cores, same frequency”

Reason 1: Moore‘s Law and CPU architectures



- Moore‘s Law: „*The density of transistors on a chip doubles every 18 months, for the same cost (1965)*“



- Since about 2006, performance of a <u>single core</u> does not increase at this rate
- But Moore’s Law still valid => more cores per chip
- Parallelization is needed to increase computing power

Reason 1: Supercomputers Become Massively Parallel

- Parallelization is the primary way to speed up computations
- Processor count in supercomputers increases each year
  - Speed of a rank X supercomputer doubles every 13 months
  - But # transistors ~ # cores per chip doubles only every 24 months
- This implies higher costs - mainly due to el. energy

	Cray X-MP/24	Cray Y-MP4/264	Cray T3D SC 192	Cray T3E	IBM 690 pSeries	SGI ICE
Year	1986	1992	1994	1997	2002	2008
GFlops/s	0.47	1.33	38	363	2662	125,000
# CPUs	2	4	192	512	**512**	**~2500**
Power (kW)				90	**160**	**600**

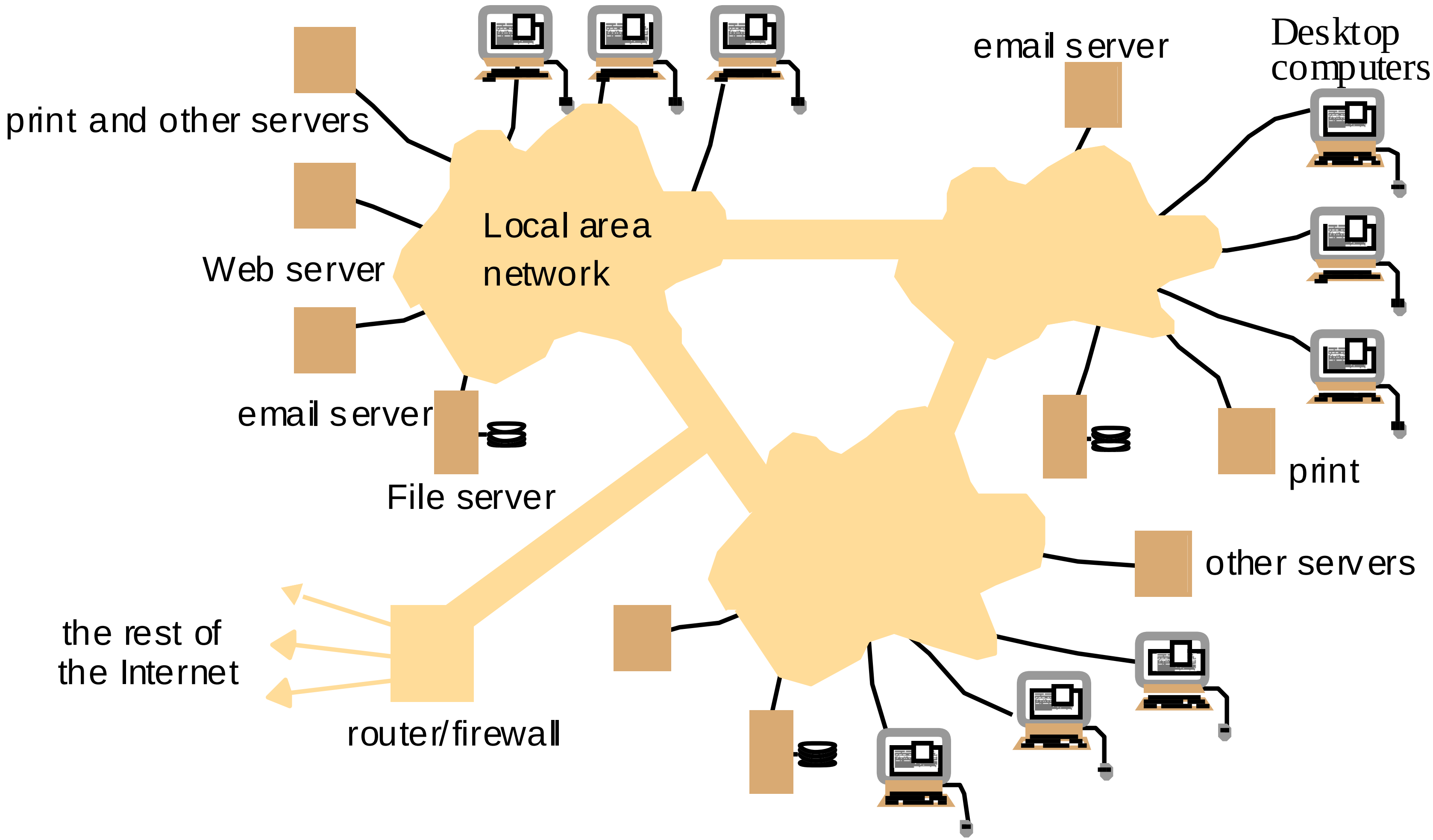
Supercomputers at Zuse Institute Berlin (ZIB)

Reason 2 for Distributed Systems

- Information/data sharing, and communication
- WWW as a main distributed application

Date	Computers	Web servers	Percentage
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12
2001, July	125,888,197	31,299,592	25
2003, January	171,638,297	35,424,956	20

Illustration of Reason 2: Communication and Sharing of Resources in Intranet



System Architectures

Taxonomy of Architectures

- Introduced 1972 by Michael Flynn

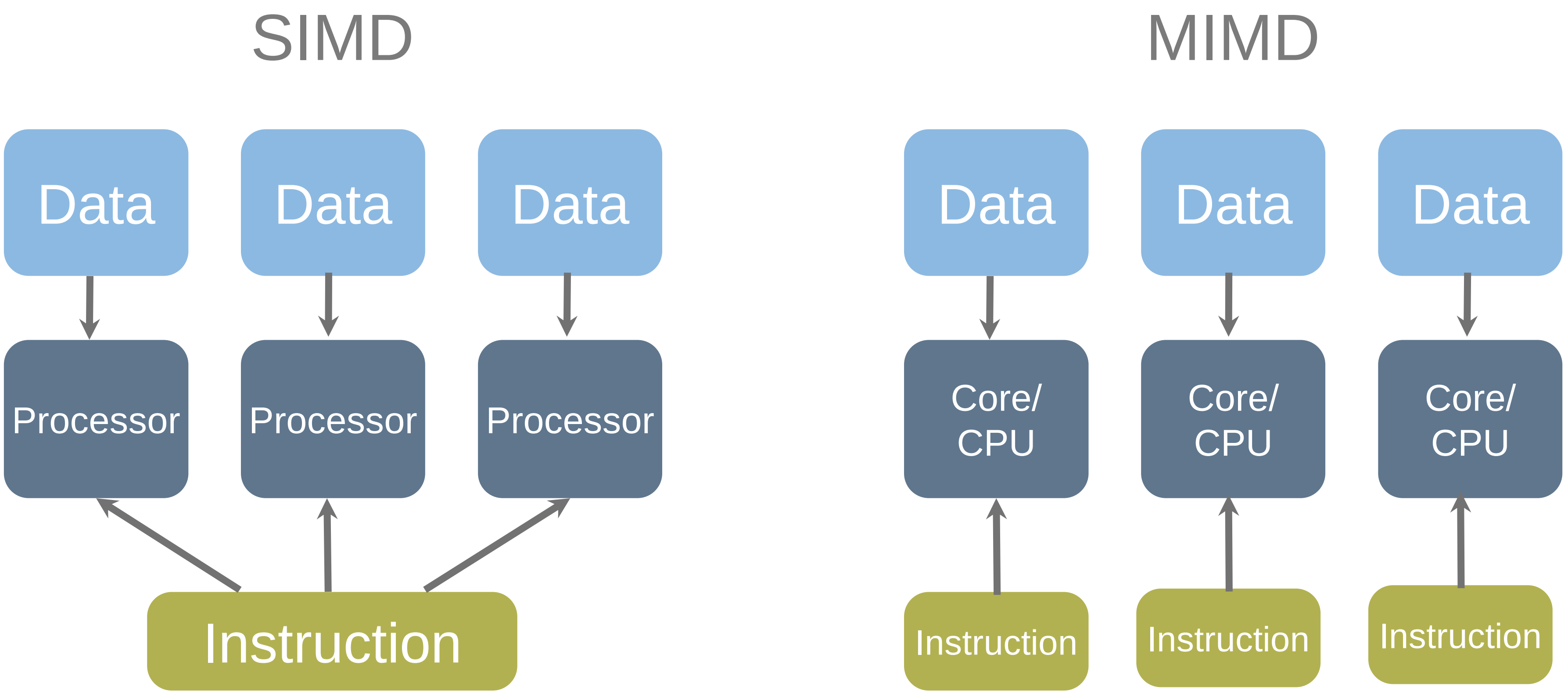
number of data streams ↓ / *number of instruction streams* →	single	multiple
single	**SISD** – single-threaded process	**MISD** – pipeline architecture (rare)
multiple	**SIMD** – vector processing	**MIMD** – multi-threaded programming

Michael Flynn's Taxonomy of Architectures

		number of instruction streams	
		single	multiple
number of data streams	single	**SISD** • Traditional one-core system	**MISD** • Generally not used and doesn't make sense, except to classifying redundant systems (tandem-computers)

	multiple	**SIMD**
		• Array (vector) processor, e.g.
		  • GPUs
		  • SSE3: Intel's Streaming SIMD Extensions
			MIMD
			• Multiple computers / cores, each with:
			  • program counter, program, data
			• Both parallel and distributed systems

Flynn's Taxonomy: SIMD vs. MIMD

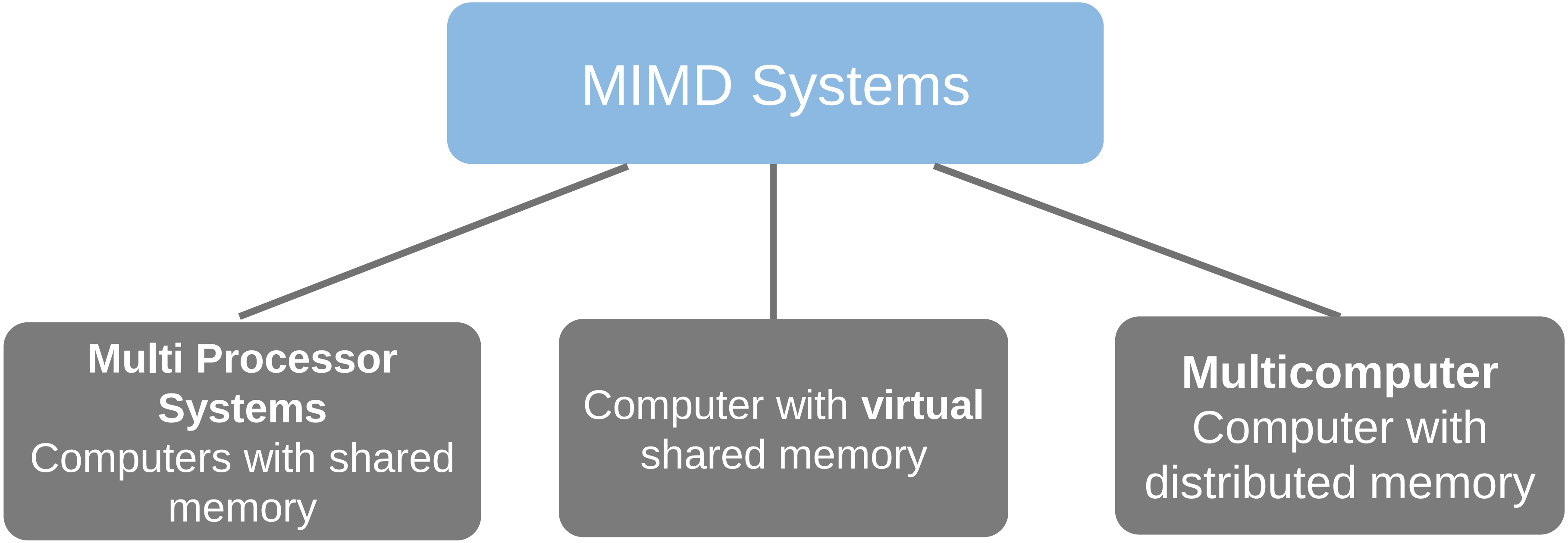


E.g. GPUs, MMX, SSE2, vector computers

E.g. power workstation, cluster, supercomputer

Memory Organization

- Most parallel computers have MIMD-architecture
- However, they can have various memory organization / architecture:



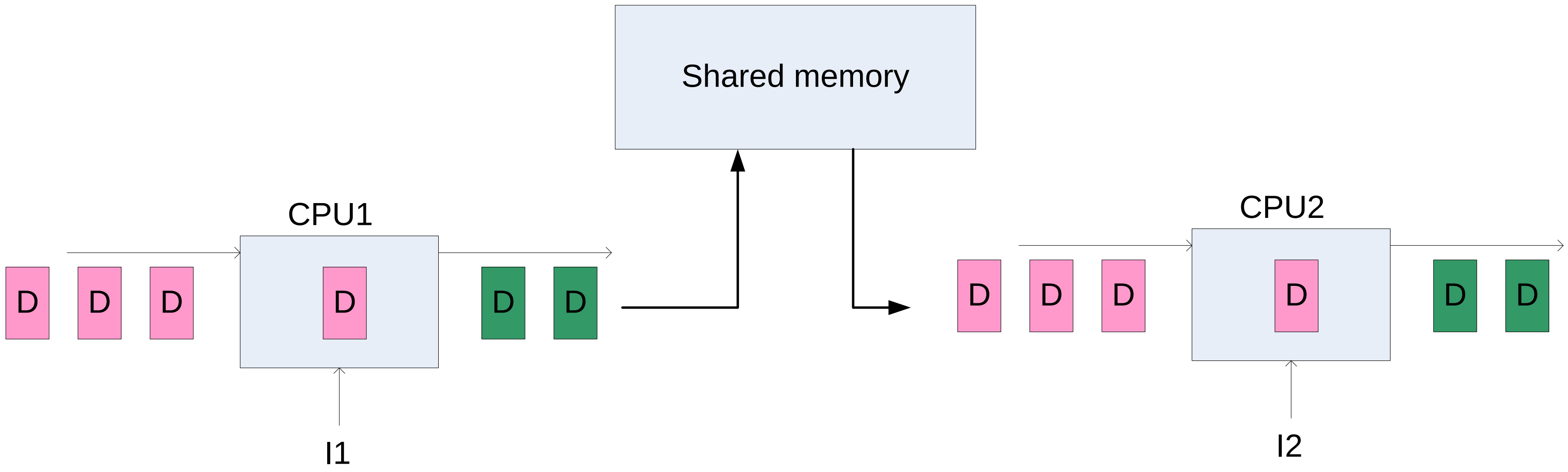
(siehe Rauber, Rünger: Parallele und verteilte Programmierung, 2000)

Parallel and Distributed Computing Systems

- **Parallel computing** systems
  - Multi- and many-core processors
  - **Multiprocessor** systems
    - Multiple CPUs in the same „box“
  - Vector processing of data (e.g. Cray-Systems)
- **Distributed computing** systems
  - **Multicomputers**: multiple CPUs across many computers (independent nodes)
  - Communication via (high-speed) network

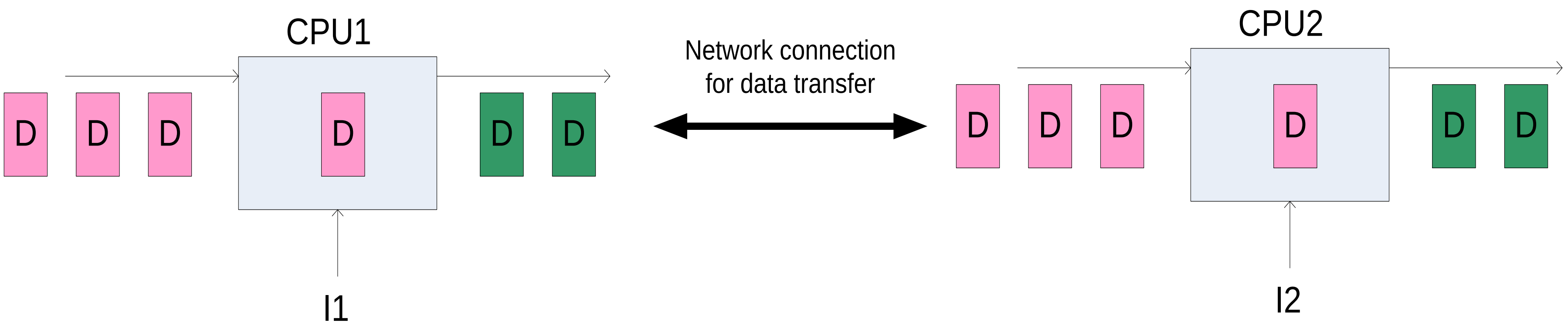
Parallel, but not Distributed Hardware

- **Multi processor machines** perform parallel processing via multiple CPUs communicating over the same **shared-memory** hardware/bus



Multicomputers: „Truly Distributed“ Hardware

- A **multicomputer** has following architecture:
  - It has multiple independent computers / nodes
  - Each node has its own memory
  - They communicate over a (private or public) network
    - Usually via message passing, e.g. MPI



Communication Styles 1: Hardware vs. Communication

- DS definitions say that it <u>does not share</u> primary memory and nodes communicate <u>only</u> by message passing
- In fact, these are two <u>independent</u> aspects
  - A. Memory organization: shared vs. not shared memory
  - B. Communication type: “shared mem” vs. “message passing”

	A1. “True” single bus/memory hardware (PCs, workstations etc.)	**A2. Physically distributed nodes (clusters, clouds, HPC systems, …)**
B1. Communication via shared-memory	“Natural”, but coding is error-prone (race conditions, deadlocks..)	Possible as “Distributed Shared Memory”; lot of research, controversial
B2. Communication via message passing	Used in “safe” OSs and modern prg. languages: Actors in Erlang, Scala, ..	“Natural” and widely used, e.g. MPI; can be more abstract, e.g. Java RMI

Communication Styles 2: Shared-Memory vs. Message Passing

- Both form of communication are used in distributed and non-distributed systems!
- **Distributed Shared-Memory**
  - A communication approach, in which each node of a distributed system has access to a large shared memory (in addition to own non-shared private memory)
- **Actor model** of concurrent programming
  - A model of concurrent processing where threads communicate via message passing and do not use locks / monitors; i.e. processors with physically shared memory still use message passing
  - Available in Erlang, Scala, Java libraries, some OSs..

Thank you.