TECHNISCHE UNIVERSITÄT DARMSTADT

Large-Scale Parallel Computing

Prof. Dr. Felix Wolf

INTRODUCTION

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

- Why parallel computing?
- Why parallel programming?
- Types of parallelism
- Developing parallel software
- Obstacles to parallelism
- Preview of course contents

Why (large-scale) parallel computing?

TECHNISCHE UNIVERSITÄT DARMSTADT

Problems that cannot be solved fast enough sequentially

- Example: simulation of physical phenomena

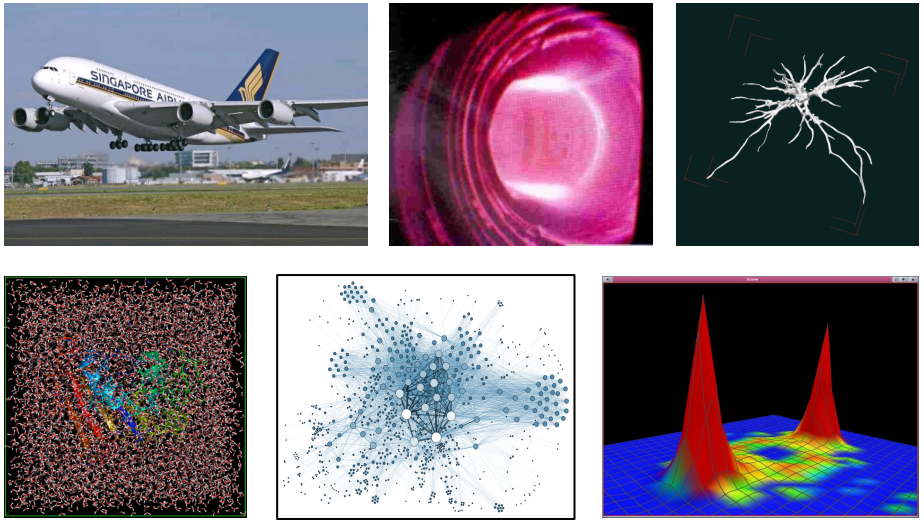
- Based on the idea that larger problems can be divided into subproblems to be solved concurrently
- Two dimensions
  - Problem size
  - Time to solution
- Sometimes real-time constraints (e.g., weather forecast)
- Usually requires parallel computer / multiprocessor

Hurricane Katrina

Further examples of compute-intensive application

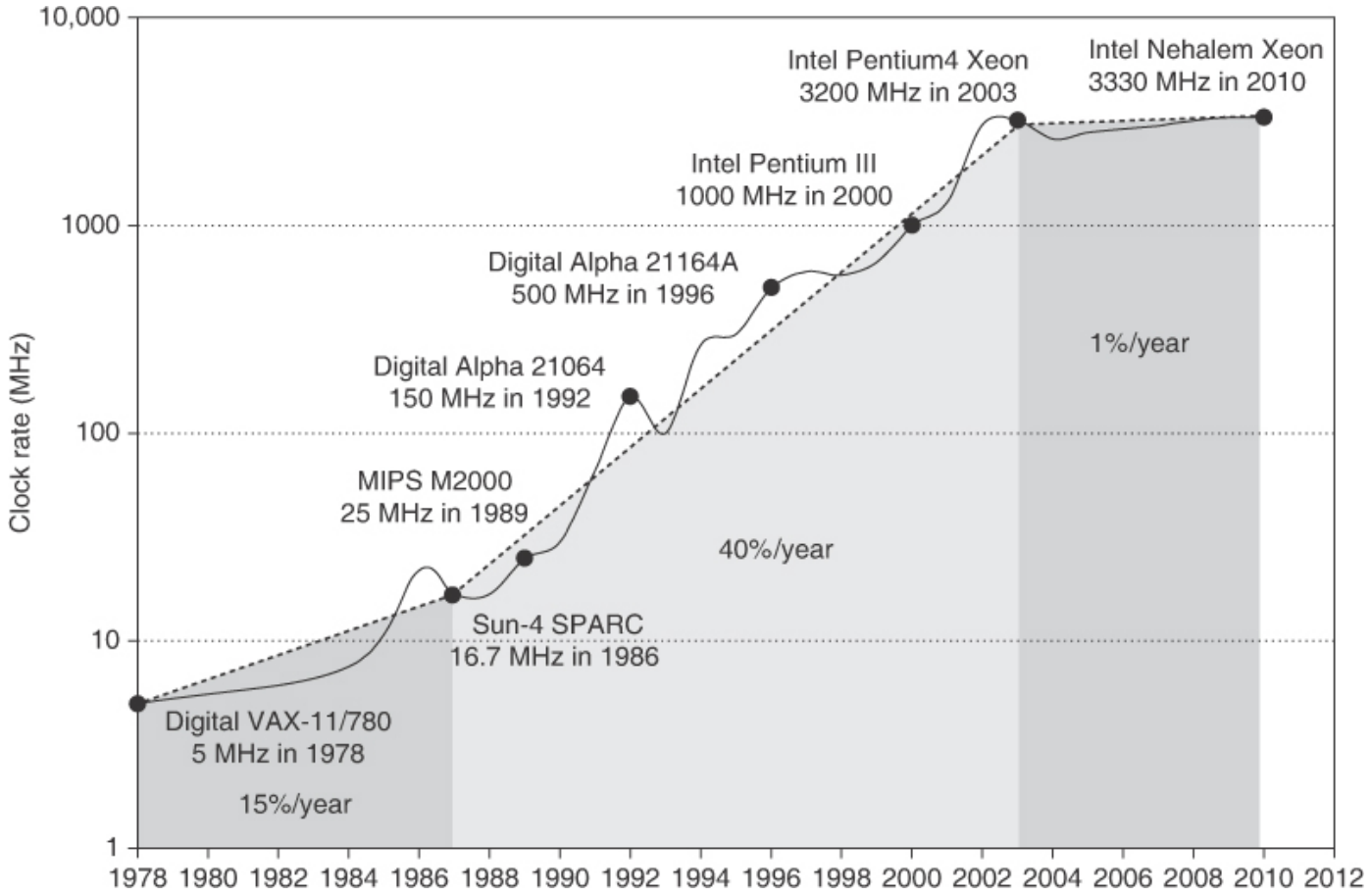
TECHNISCHE UNIVERSITÄT DARMSTADT

- Simulations
  - Natural sciences: molecular dynamics, materials science
  - Engineering: crash, aerodynamics, fluid dynamics, combustion
- BigData
  - Graph analysis
  - Sorting
- Multimedia
  - Stream processing
- Finance
  - Valuation of options and financial securities



Why can't we just create faster uni-processors?

TECHNISCHE UNIVERSITÄT DARMSTADT



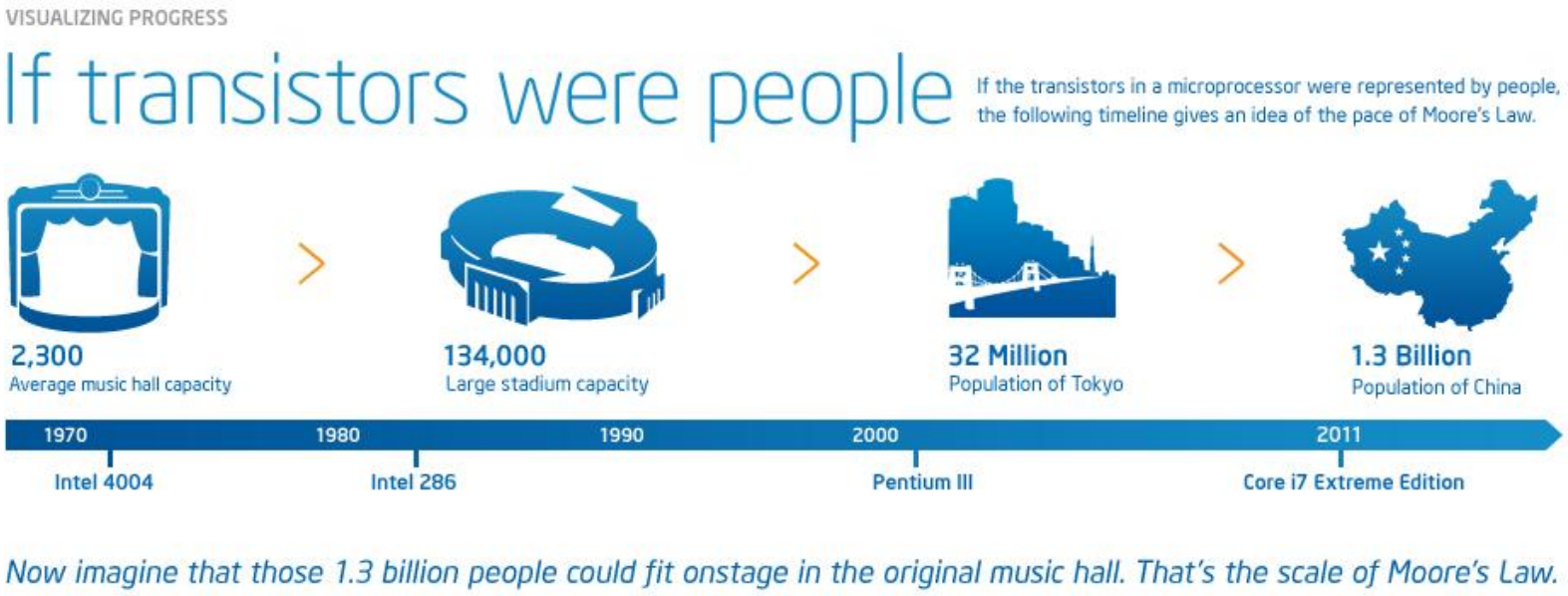
Source: Hennessy, Patterson: Computer Architecture, 5th edition, Morgan Kaufmann

Moore's law

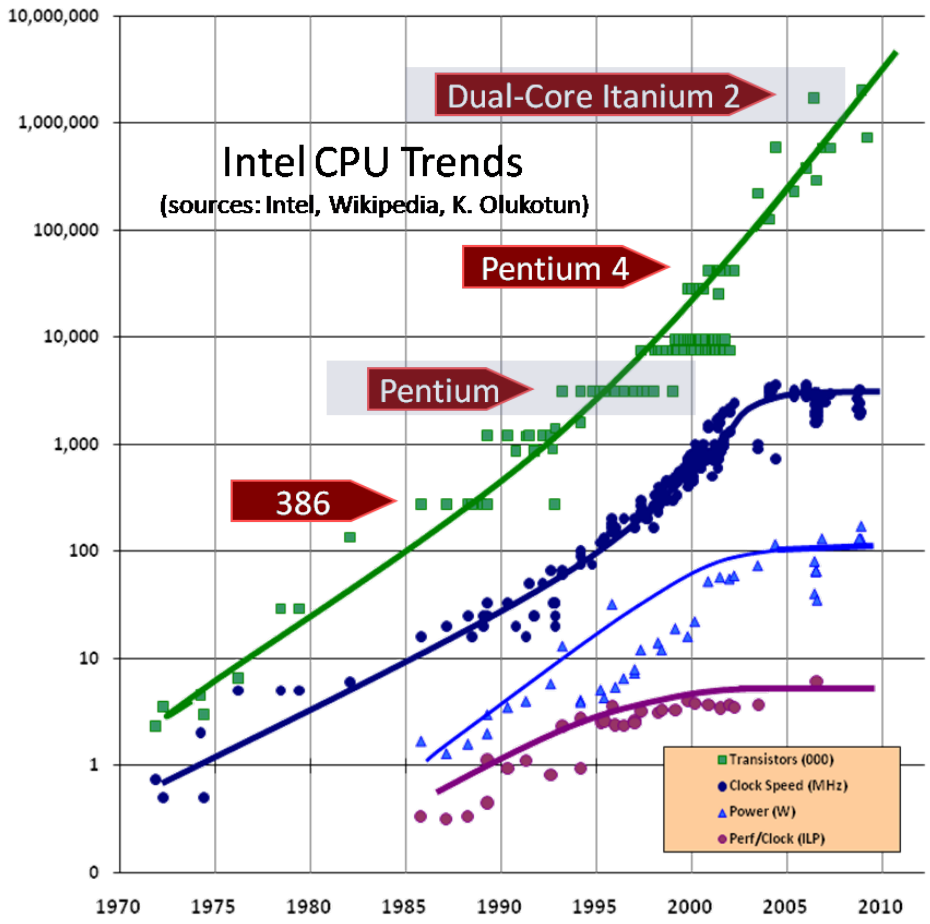
TECHNISCHE UNIVERSITÄT DARMSTADT

The number of transistors per chip doubles roughly every two years

Exponential growth of processor performance



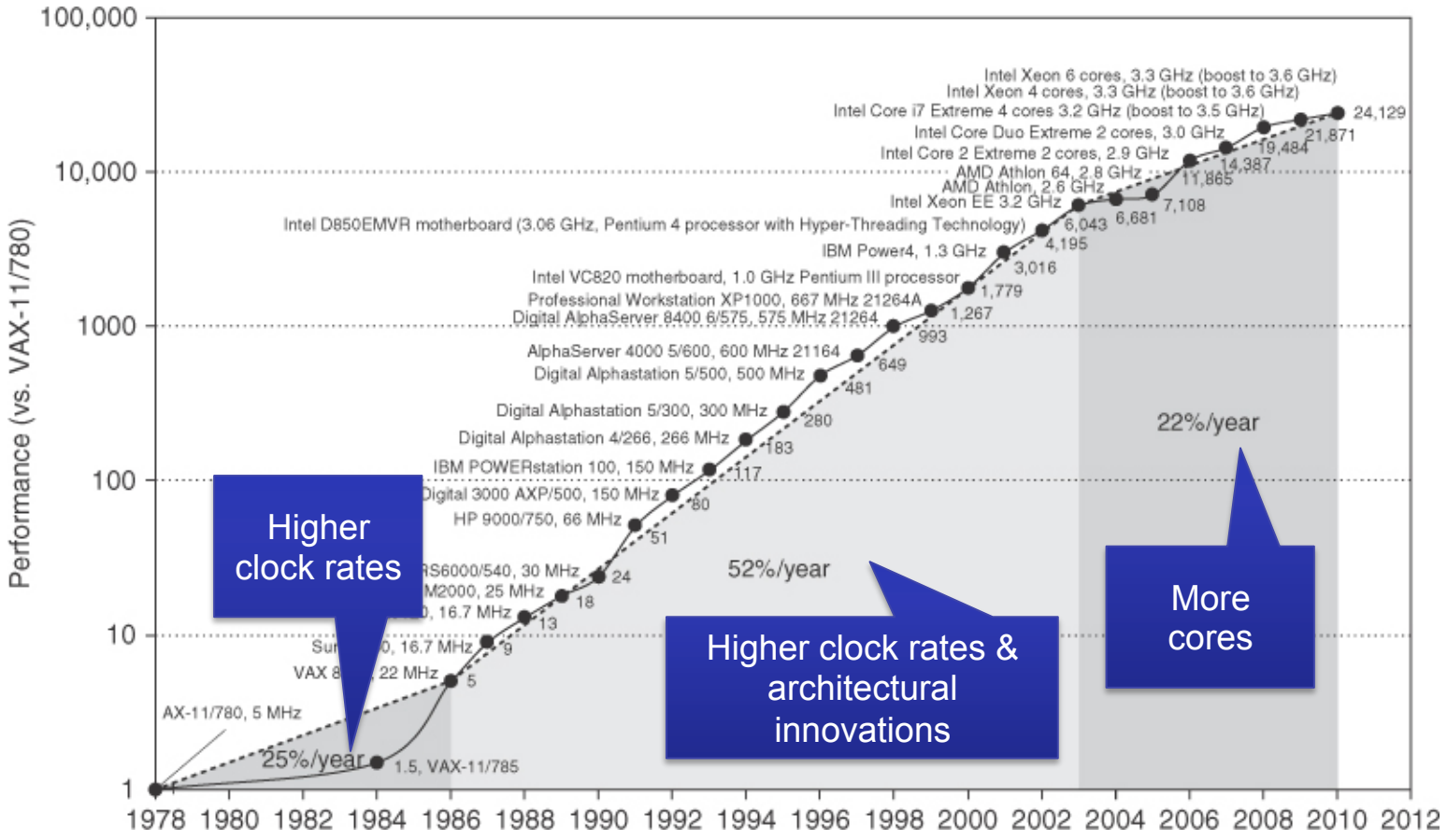
Moore's law (2)



- Reduction of feature size won't last forever
- May continue up to 5 nm
- End maybe ~ mid 2020s

Source: Herb Sutter: The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software, 2009.

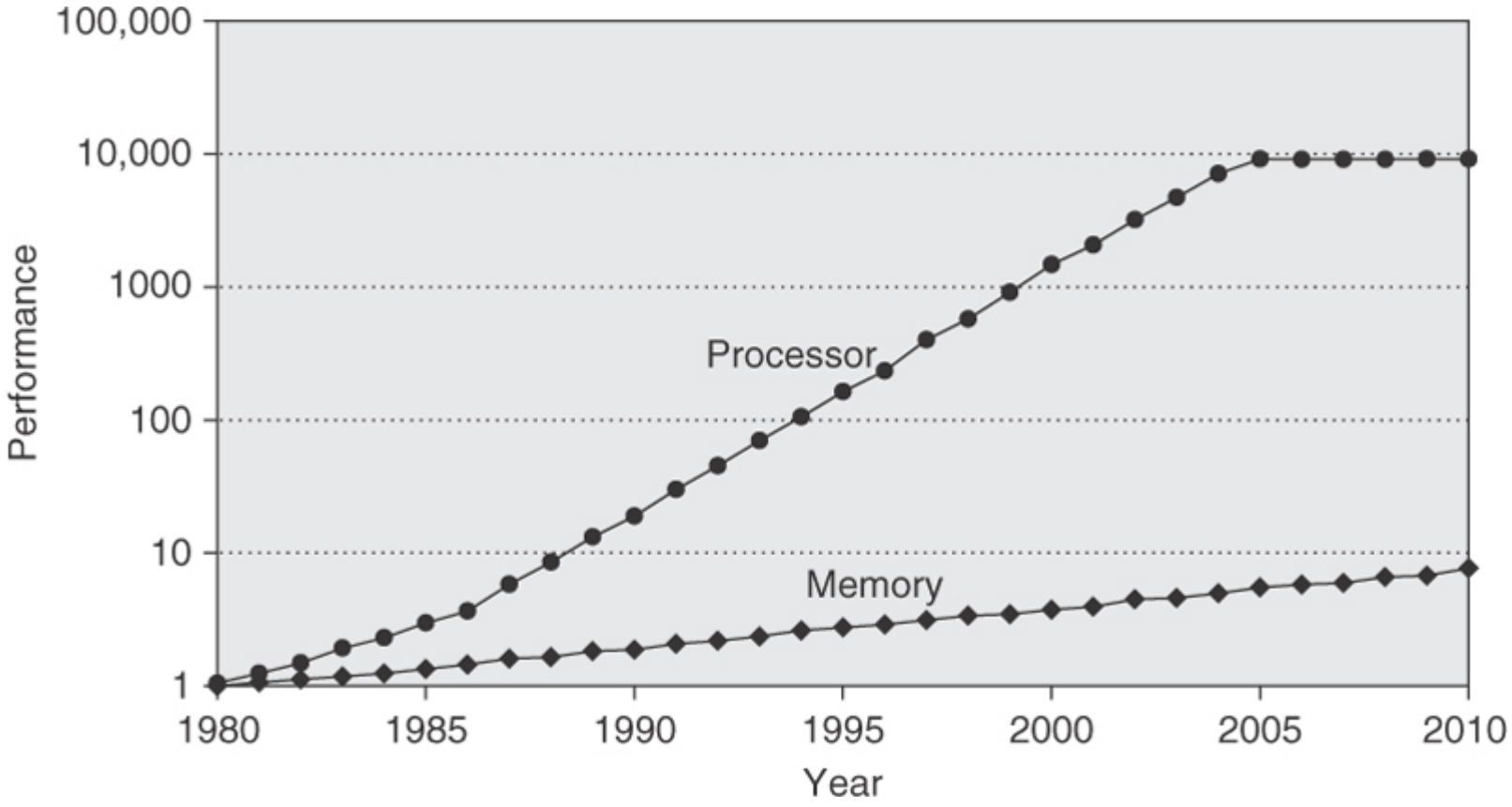
Growth in processor performance



Source: Hennessy, Patterson: Computer Architecture, 5th edition, Morgan Kaufmann

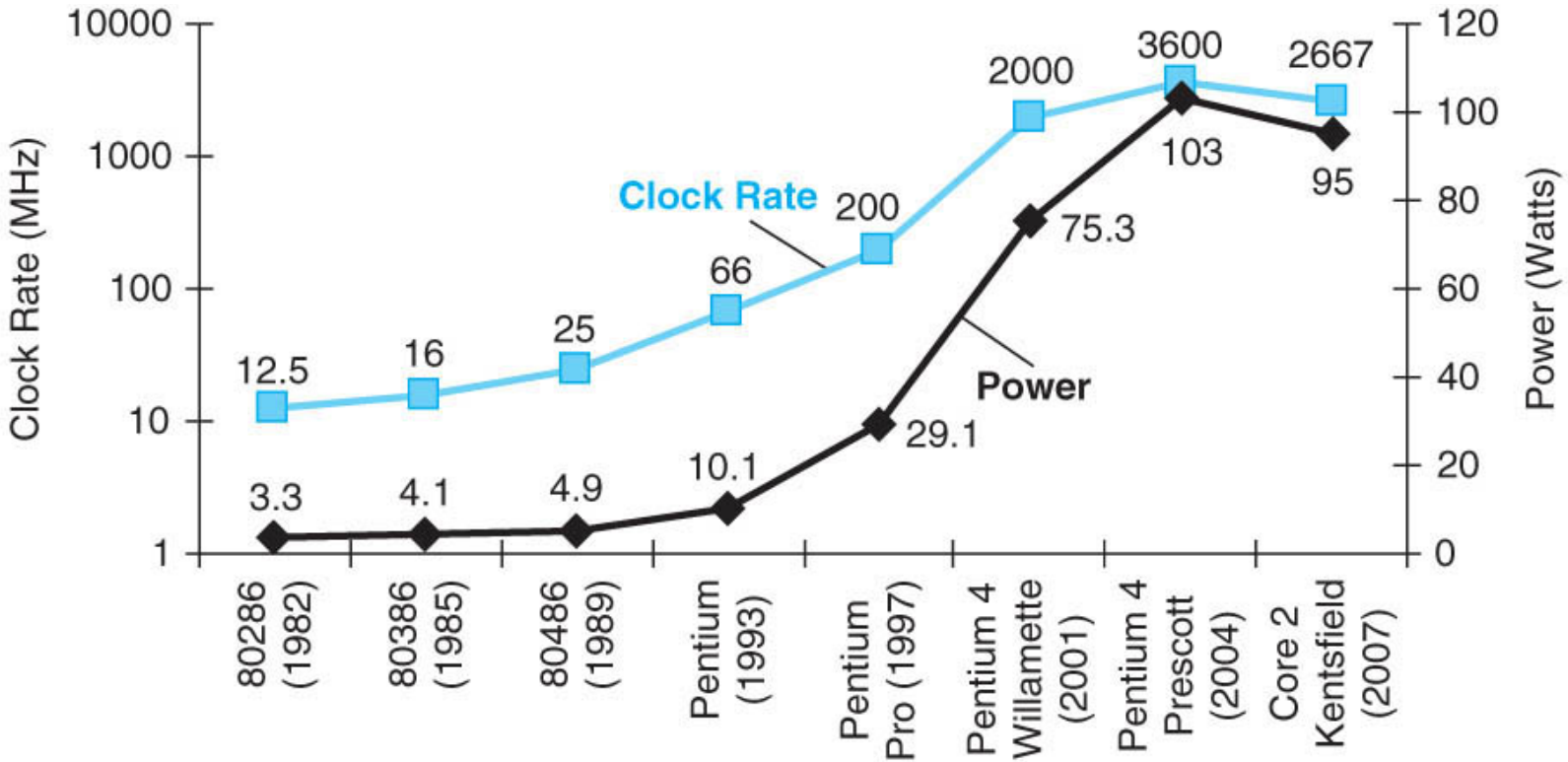
CPU and memory performance

TECHNISCHE UNIVERSITÄT DARMSTADT



Source: Hennessy, Patterson: Computer Architecture, 5th edition, Morgan Kaufmann

Clock rate vs. power



Source: Patterson, Hennessy: Computer Organization & Design, 4th edition, Morgan Kaufmann

Dennard scaling

TECHNISCHE UNIVERSITÄT DARMSTADT

- Why haven't clock speeds increased, even though transistors have continued to shrink?
- Dennard (1974) observed that voltage and current should be proportional to the linear dimensions of a transistor
  - Thus, as transistors shrank, so did necessary voltage and current; power is proportional to the area of the transistor

Courtesy of Bill Gropp

Dennard scaling

TECHNISCHE UNIVERSITÄT DARMSTADT

Dynamic power = $\alpha * CFV^2$

- α = percent time switched
- C = capacitance
- F = frequency
- V = voltage

Capacitance is related to area

- So, as the size of the transistors shrunk, and the voltage was reduced, circuits could operate at higher frequencies at the same power

Courtesy of Bill Gropp

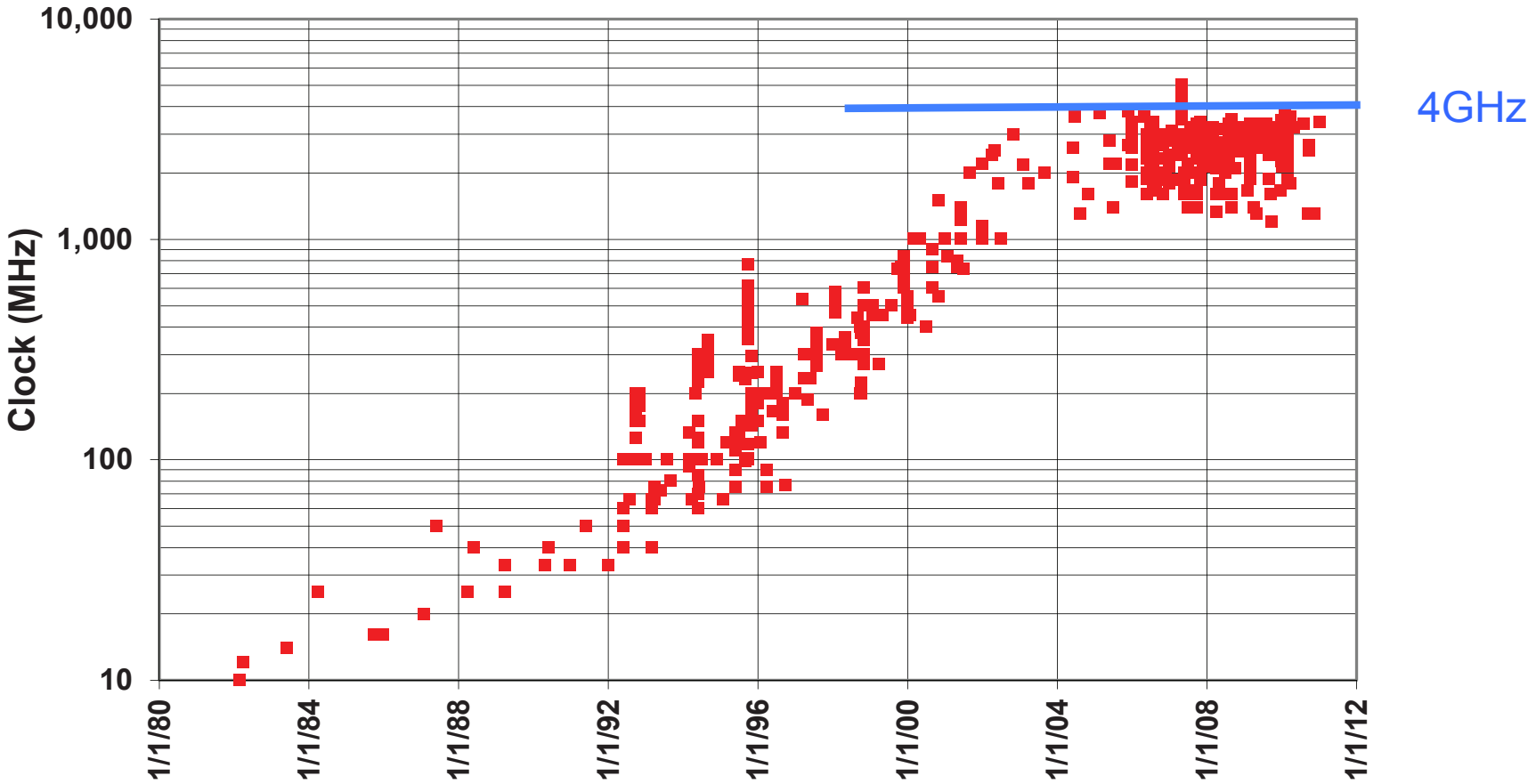
End of Dennard scaling

TECHNISCHE UNIVERSITÄT DARMSTADT

- Dennard scaling ignored the “leakage current” and “threshold voltage”, which establish a baseline of power per transistor
- As transistors get smaller, power density increases because these don’t scale with size
- These created a “Power Wall” that has limited practical processor frequency to around 4 GHz since 2006

Courtesy of Bill Gropp

Historical clock rates



Courtesy of Bill Gropp

Road maps

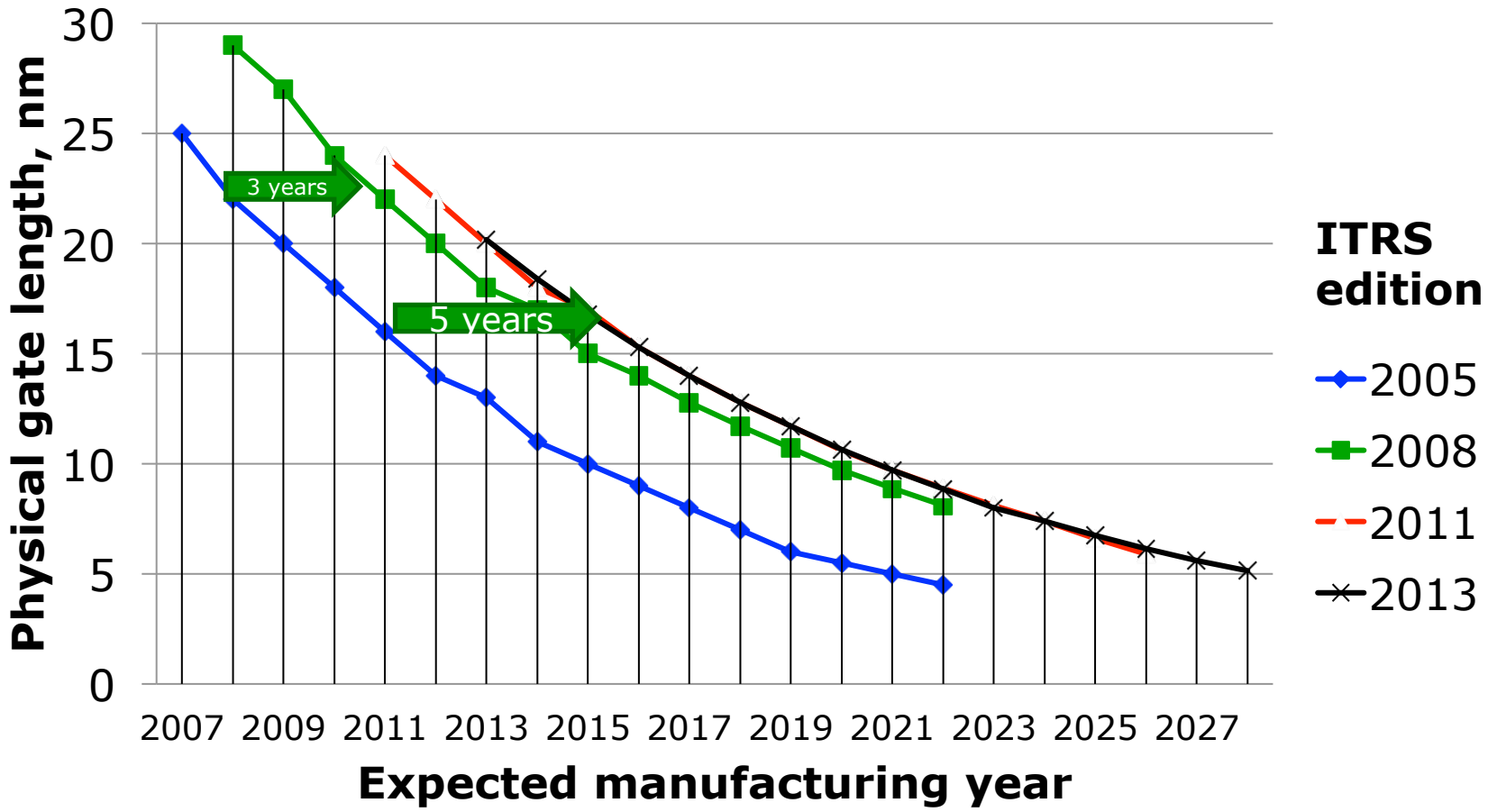
TECHNISCHE UNIVERSITÄT DARMSTADT

- The Semiconductor industry has produced a roadmap of future trends and requirements
- Semiconductor Industry Association (~1977, roadmaps from early ’90s)
- International Technology Roadmap for Semiconductors (~1998)
  - http://www.itrs.net/

Courtesy of Bill Gropp

ITRS projections for gate lengths (nm) for 2005, 2008 and 2011 editions

TECHNISCHE UNIVERSITÄT DARMSTADT



Courtesy of Bill Gropp

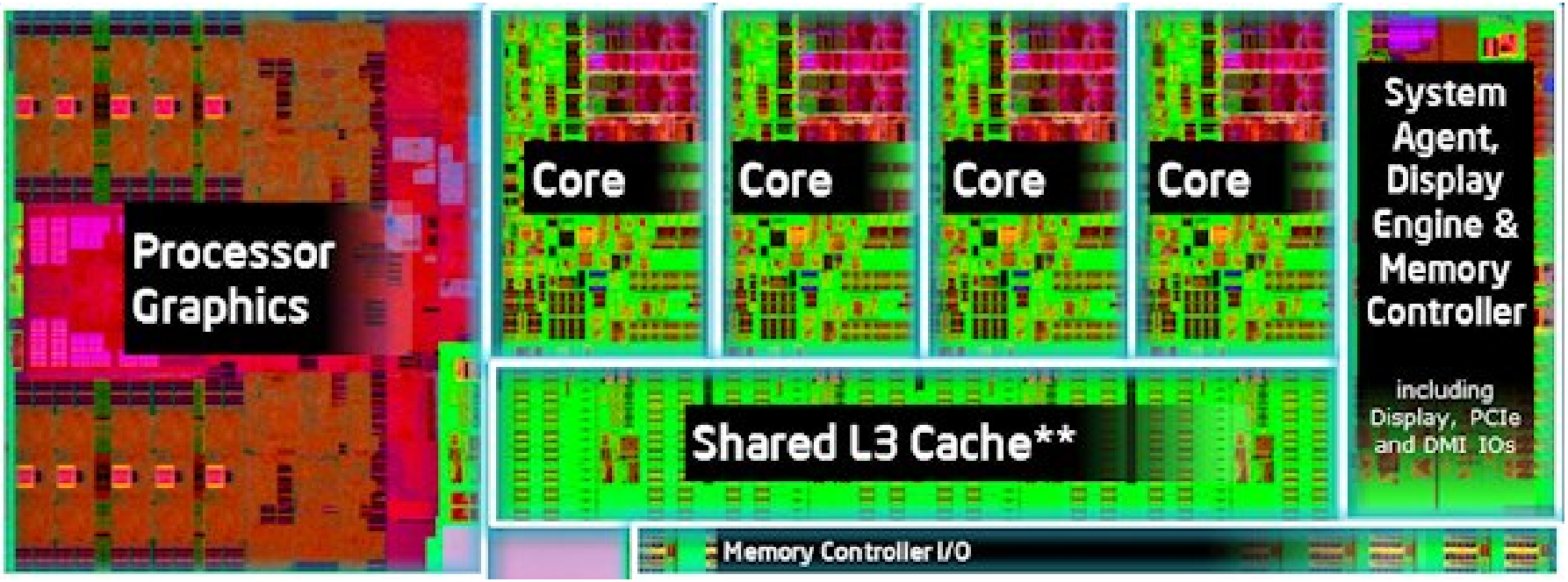
Multicore

TECHNISCHE UNIVERSITÄT DARMSTADT

- Since 2002, uni-processor performance improvement has dropped
  - Power dissipation
  - Little instruction-level parallelism left to exploit efficiently
  - Almost unchanged memory latency
- Further performance improvements by placing multiple processors on a single die (multi-core architecture)
  - Initially called on-chip or single-chip multiprocessing
  - Cores often share resources (e.g., L1, L2 cache, I/O bus)
  - Does not solve the memory wall problem (memory bandwidth)
- Leverages design investment by replicating it

Intel Haswell

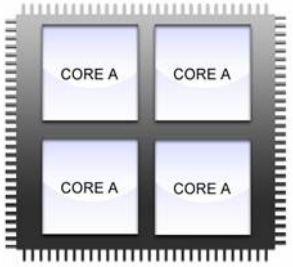
TECHNISCHE UNIVERSITÄT DARMSTADT



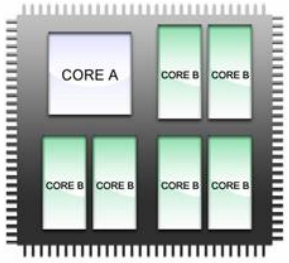
Manycore

TECHNISCHE UNIVERSITÄT DARMSTADT

- New version of Moore's law
  - The number of cores will double every two years
  - Recall that today's GPUs feature 100s and 1000s of cores
- Heterogeneity
  - Not all cores necessarily uniform
  - Cores for specific functions
    - Control vs. computation
    - Video
    - Graphics
    - Cryptography
    - Digital signal processing
    - Vector processing



Homogeneous design



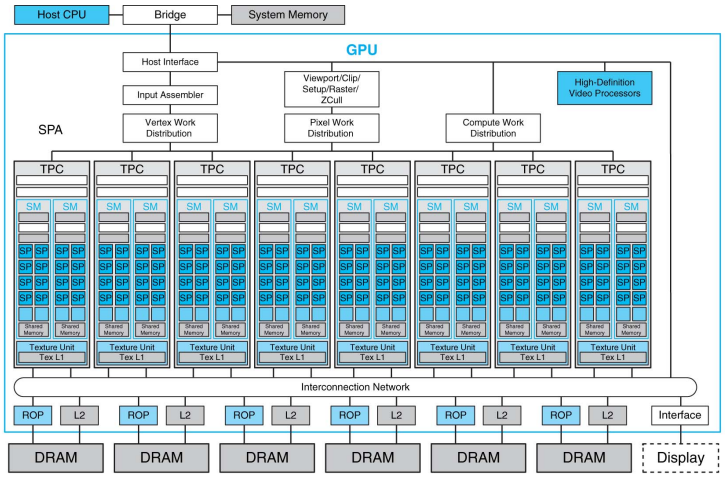
Heterogeneous design

Graphics processing units (GPUs)

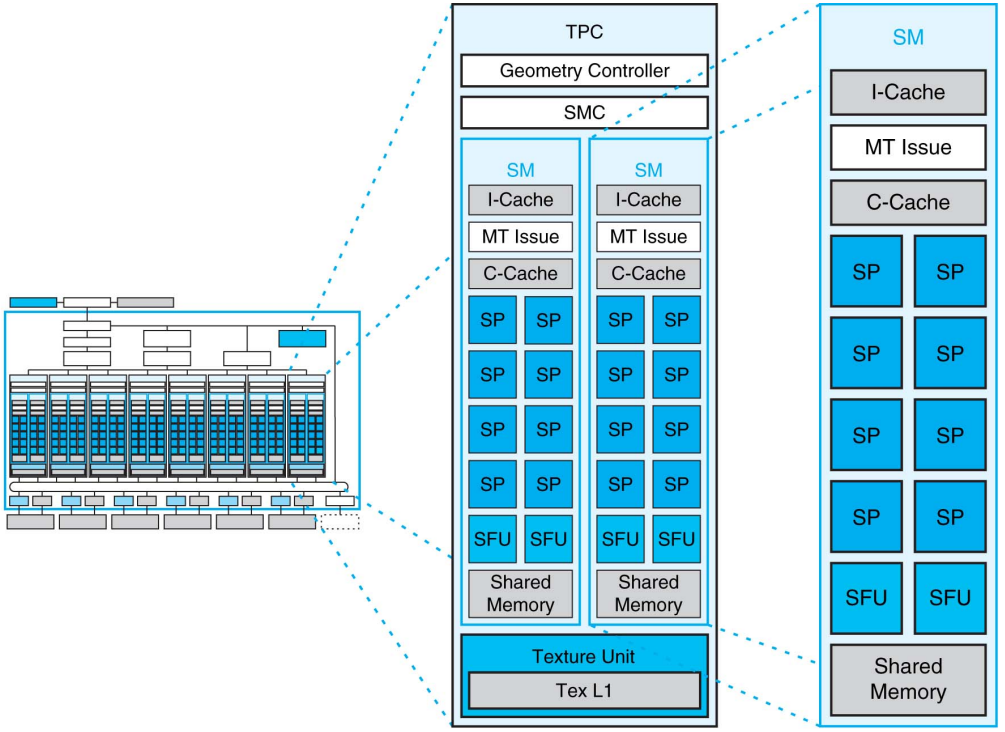
TECHNISCHE UNIVERSITÄT DARMSTADT

- Processors optimized for 2D and 3D graphics and video
- Became more programmable over time
  - Dedicated logic replaced by programmable processors
- New paradigm at the intersection of graphics processing and parallel computing: visual computing
  - Enables new graphics algorithms
- GPU computing
  - Using a GPU for computing via a parallel programming language and API (e.g., CUDA, OpenCL)

Example: NVIDIA G80

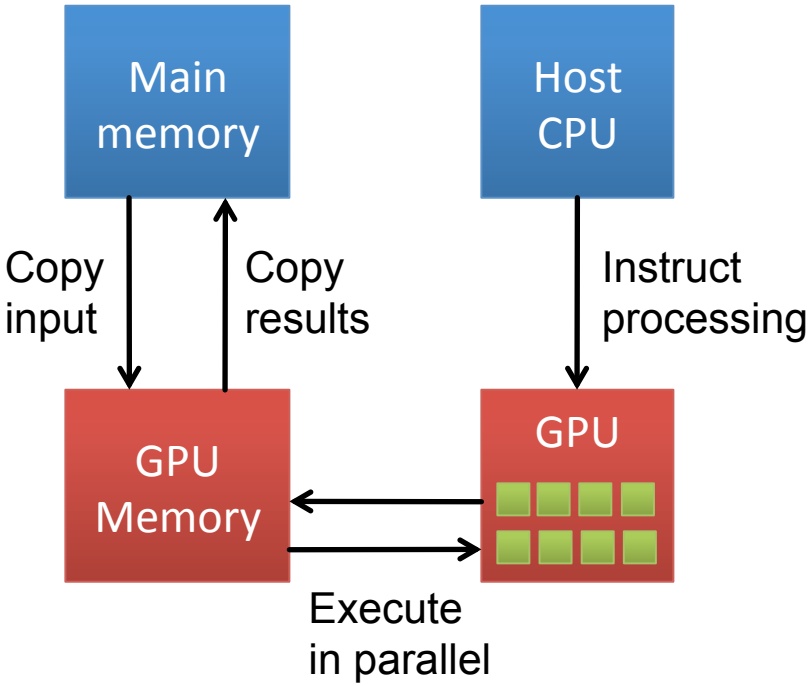


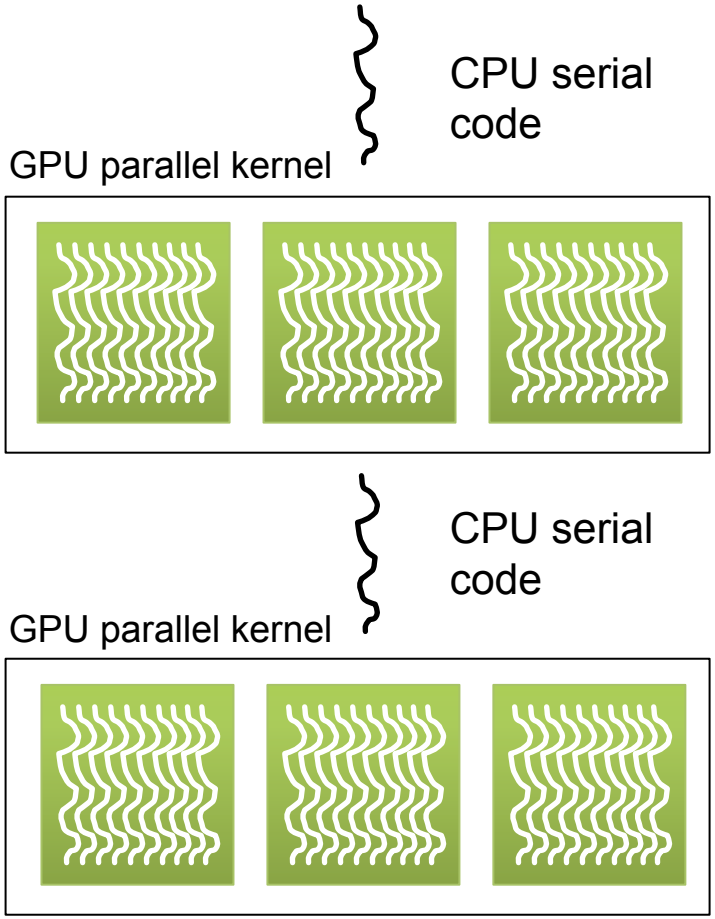
Source: Patterson, Hennessy: Computer Organization & Design, 4th edition, Morgan Kaufmann



GPU forms heterogeneous system with general-purpose CPU

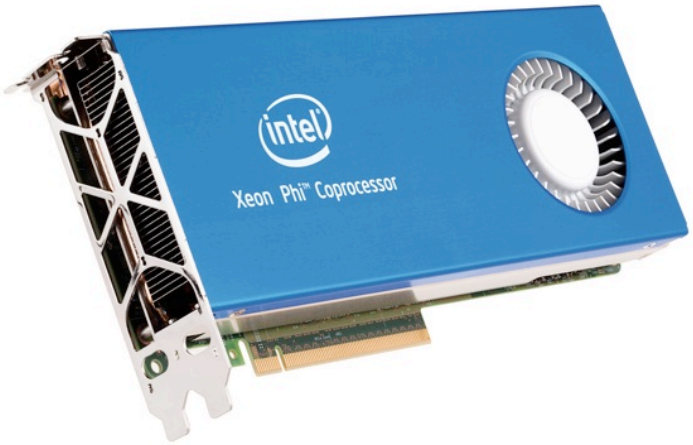
GPU computing

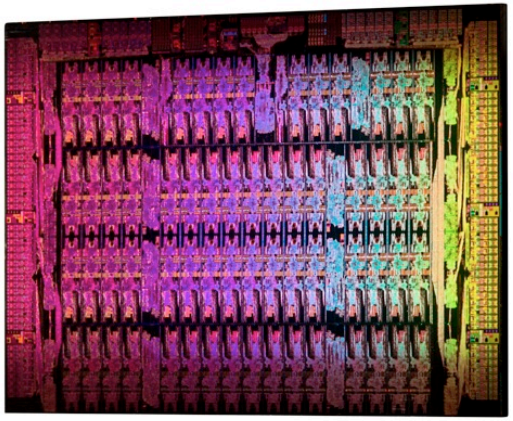




Intel Xeon Phi Coprocessor

TECHNISCHE UNIVERSITÄT DARMSTADT





Model 7120

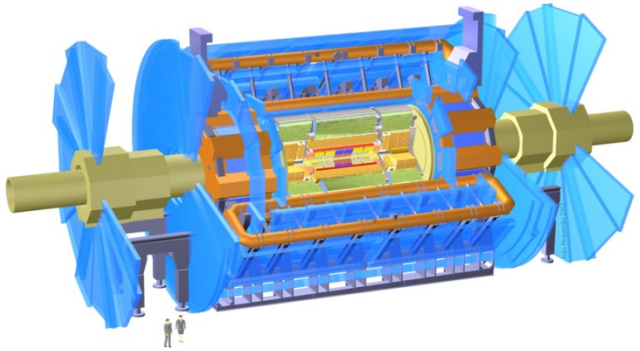
- 61 x86-based cores
- Core frequency 1.238 GHz
- 4 hardware threads per core
- 32 KB L1 cache per core
- 512 KB L2 cache per core
- Cache coherence across entire coprocessor
- 244 hardware threads in total
- 512-bit wide SIMD instructions
- 16 GB GDDR5 memory

Distributed computing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aggregation of dispersed computing resources
  - Idle workstations or dedicated workstation farms
  - Example: SETI@home
- Usually more loosely coupled than parallel computing
  - Most suitable for embarrassingly-parallel problems
- Emphasis on high throughput instead of high performance
  - Larger number of smaller jobs

PC farms at CERN

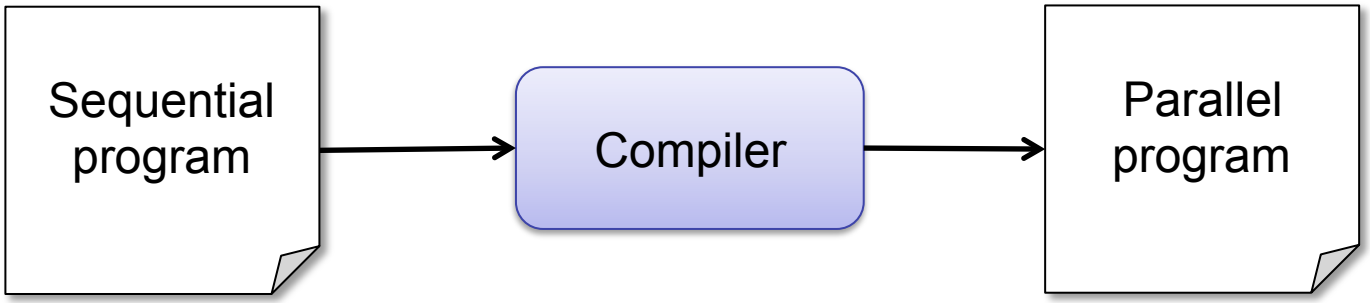


Detector ATLAS

Why parallel programming?

TECHNISCHE UNIVERSITÄT DARMSTADT

Why don't we have auto-parallelizing compilers?



- Not all parallelization opportunities statically visible
- Would result in very conservative parallelization
- Practical only for certain types of loops (e.g., Intel compiler)
- Also auto-vectorization of suitable code possible

Types of parallelism

TECHNISCHE UNIVERSITÄT DARMSTADT

Functional parallelism

- Views problem as a stream of instructions that can be broken down into functions to be executed simultaneously
- Each processing element performs a different function
- Sometimes also called task parallelism

Data parallelism

- Views problem as an amount of data that can be broken down into chunks to be independently operated upon (e.g., array)
- Each processing element performs the same function but on different pieces of data

Example

TECHNISCHE UNIVERSITÄT DARMSTADT

Several tutors grade a test

- The task sheet contains several tasks

Functional parallelism	Data parallelism
Each tutor grades a different subset of the tasks	Each tutor grades a subset of the students

Another example

TECHNISCHE UNIVERSITÄT DARMSTADT

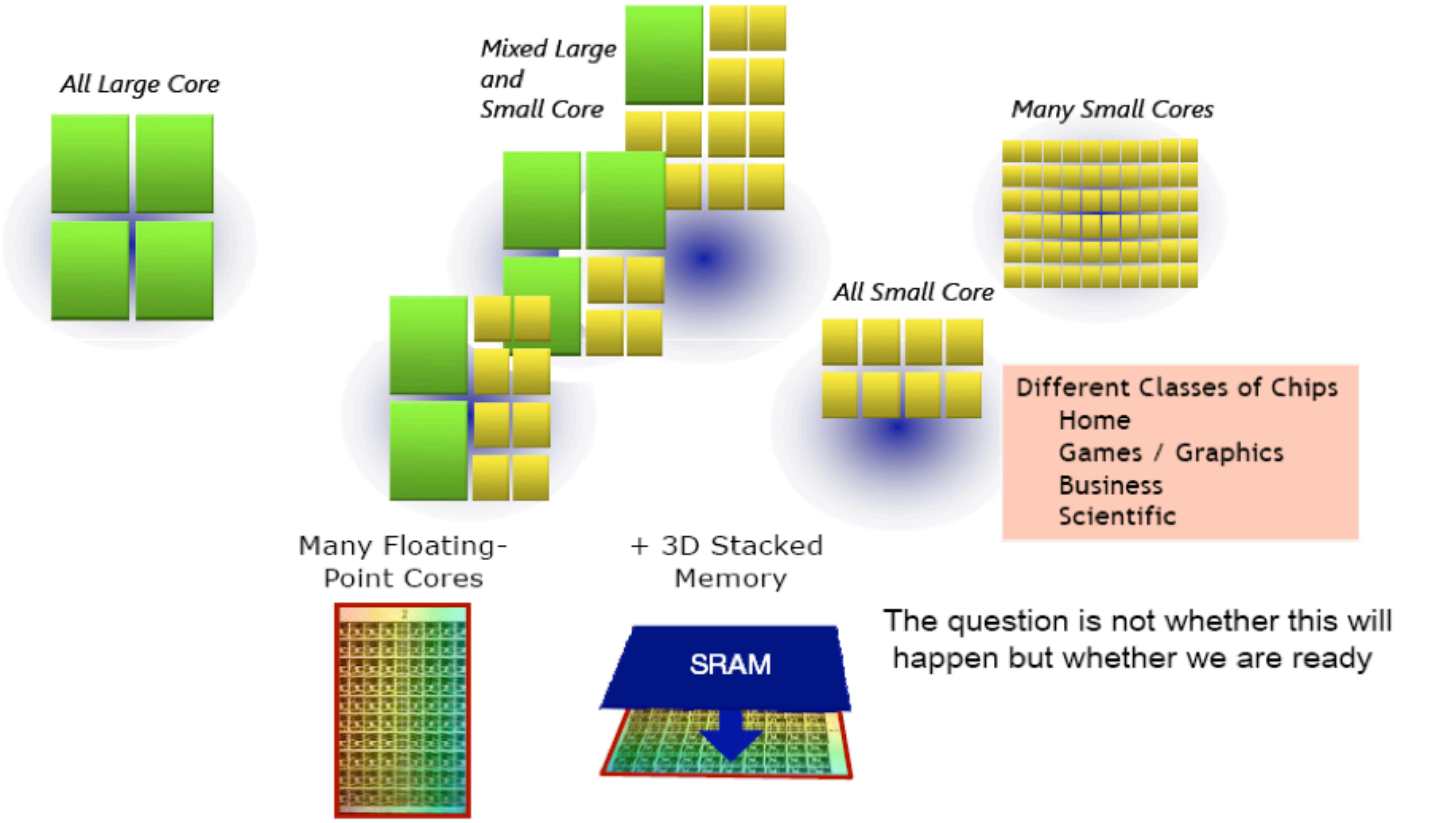
Functional parallelism

Construction workers

Data parallelism

Hollow square formation

Heterogeneity generalized



Developing parallel software

TECHNISCHE UNIVERSITÄT DARMSTADT

Three scenarios

- Writing parallel code from scratch
- Parallelizing a sequential program
- Modifying a parallel program

} Modifying existing software

Redesign is normal, and software design “de novo” is the exception

Ralph E. Johnson

Cost and benefits of parallelization

TECHNISCHE UNIVERSITÄT DARMSTADT

Benefit

- Speedup
- Sometimes improved interactivity and cleaner design (separation of concerns)
- More aggregate memory (distributed memory parallelization)

Cost

- Programming effort
- Program complexity
- Overhead (communication & synchronization)
- Bugs
- Potentially non-determinism
- Extra dependencies (library, compiler)

Parallelization strategy

Sequential program with 150 loops

- Where to look for potential parallelism?
- Which loops should be parallelized?
- Which loops cannot be parallelized?

Dependences

TECHNISCHE UNIVERSITÄT DARMSTADT

Two types

- Control dependences

```
if (condition) then
   do_work();
```

- Data dependences

```
for ( i = 1; i <= 2; i++ )
   a[i] = a[i] + a[i-1];
```

Dependences may prevent parallelization

Summary

- No more improvements of scalar performance
  - Frequency wall
  - Memory wall
  - Power wall
- Data parallelism usually more scalable than functional parallelism
- Development of parallel software occurs rarely from scratch
- When parallelizing a program, pay attention to
  - Correctness
  - Performance
  - Cost

Preview

TECHNISCHE UNIVERSITÄT DARMSTADT

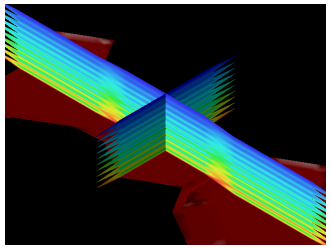
- Large-scale parallel computing
- Distributed-memory architectures
- Foundations of message passing
- Collective operations
- Data types
- Remote memory access
- Hybrid programming
- Parallel I/O
- Partitioned global address space

Coverage in course as far as we get…

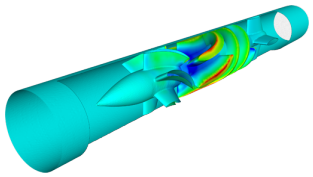
Large-scale parallel computing

TECHNISCHE UNIVERSITÄT DARMSTADT

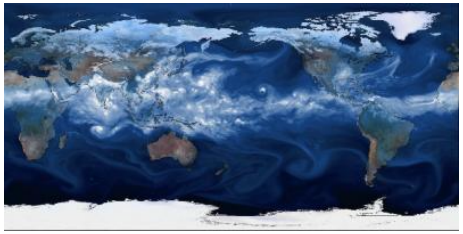
- Solution of very big problems / processing of very big data sets using very big machines



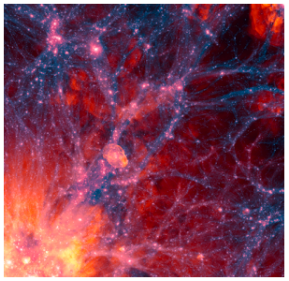
Life sciences



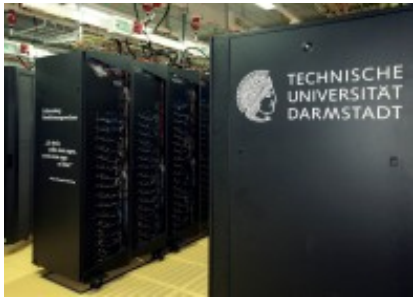
Engineering



Climate prediction



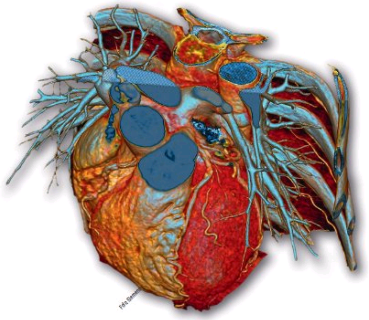
Astrophysics

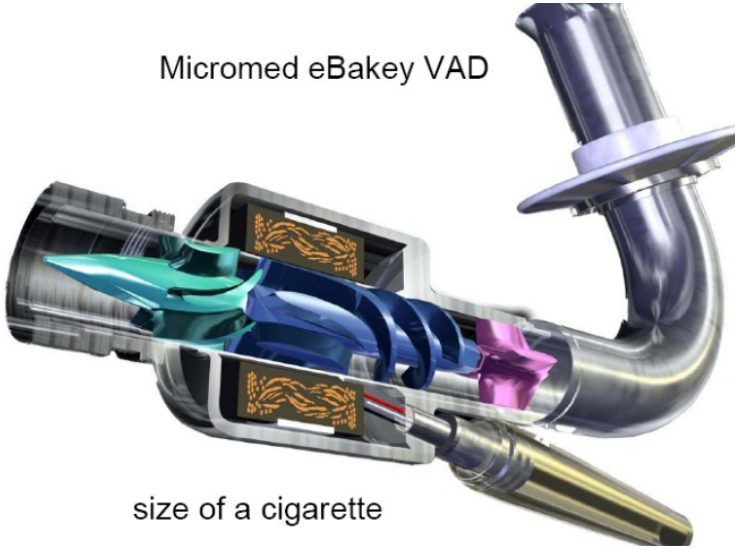


Example: blood pump

TECHNISCHE UNIVERSITÄT DARMSTADT

- Heart disease is a major cause of death in industrialized nations
- Alternative: ventricular assist device
- Impeller drives blood circulation and disburdens the heart
- Complications
  - Damage of blood cells (hemolysis)
  - Thrombosis



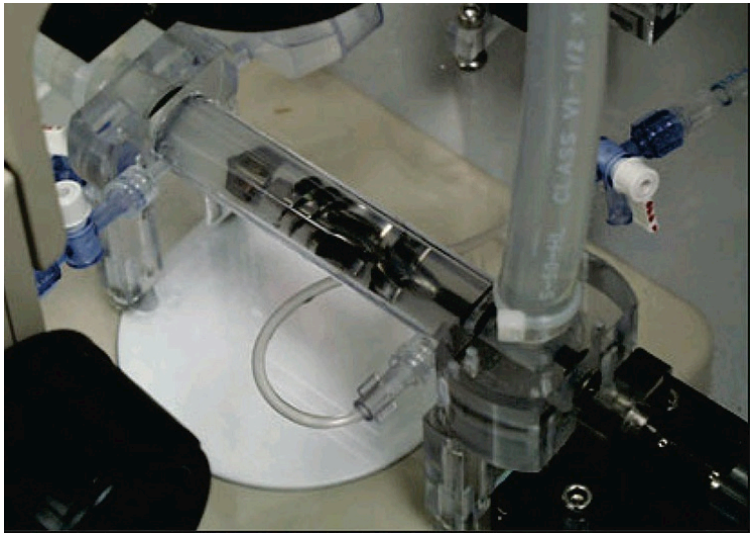


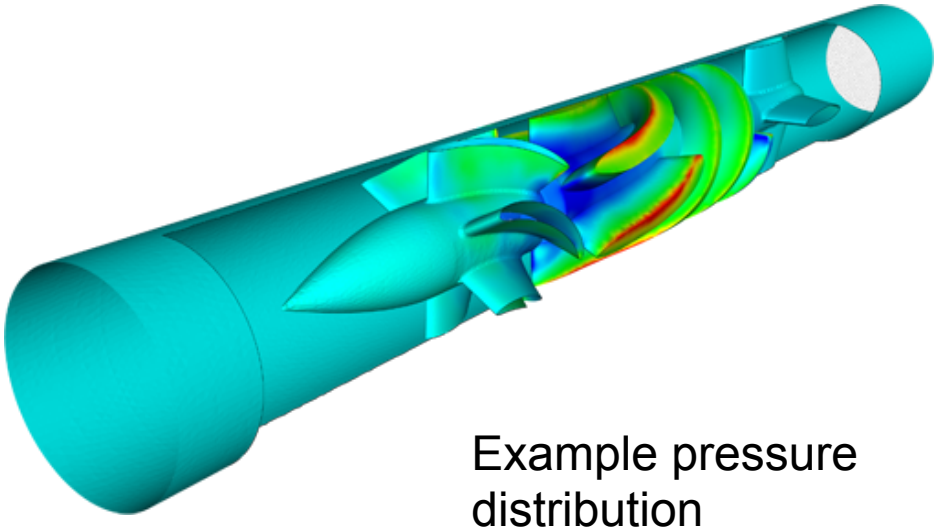
Source: Computer Assisted Analysis of Technical Systems, RWTH Aachen, Prof. Marek Behr

Optimization

TECHNISCHE UNIVERSITÄT DARMSTADT

- Trial and error on the basis of experiments
  - Expensive prototypes
  - Little insight

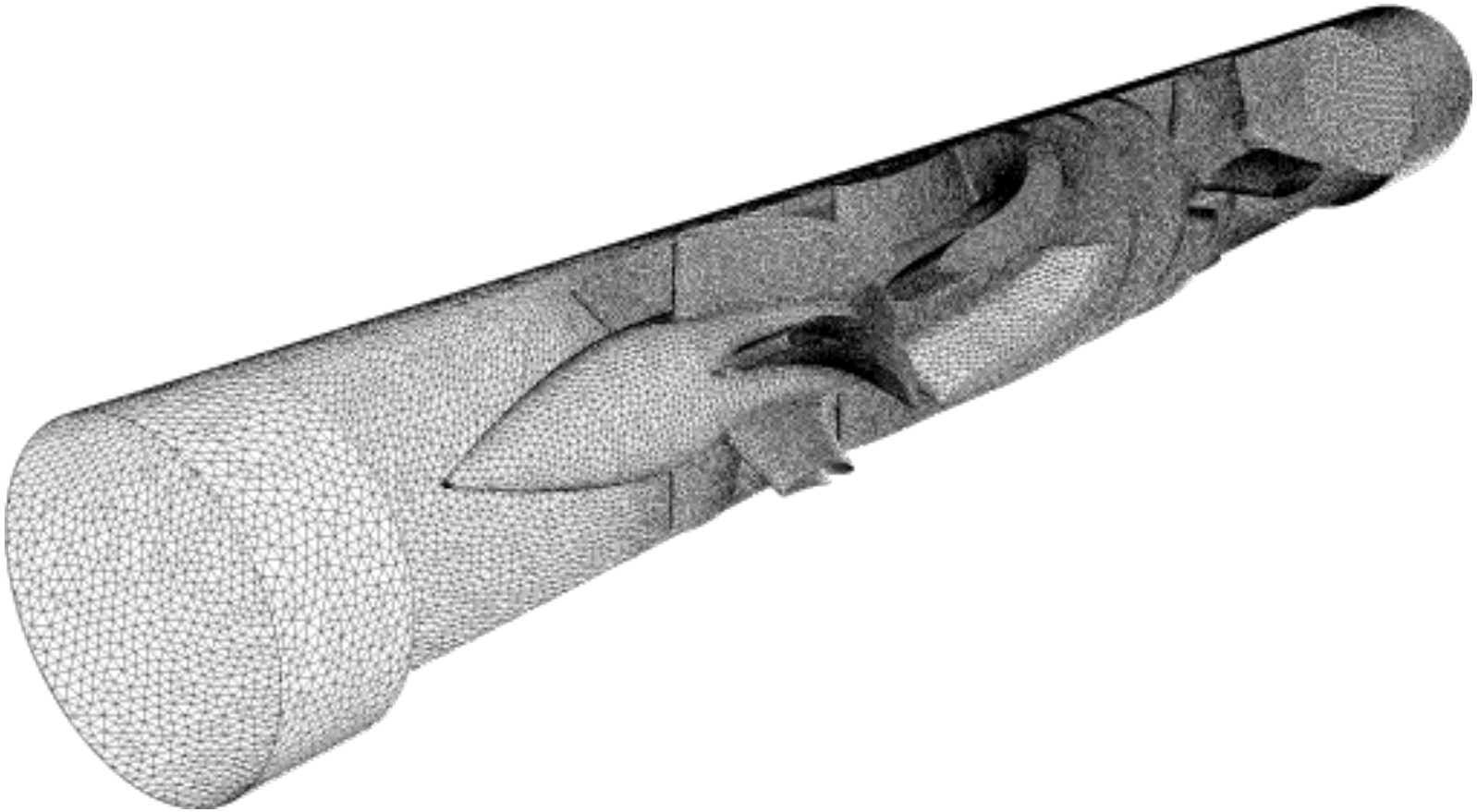




Example pressure distribution

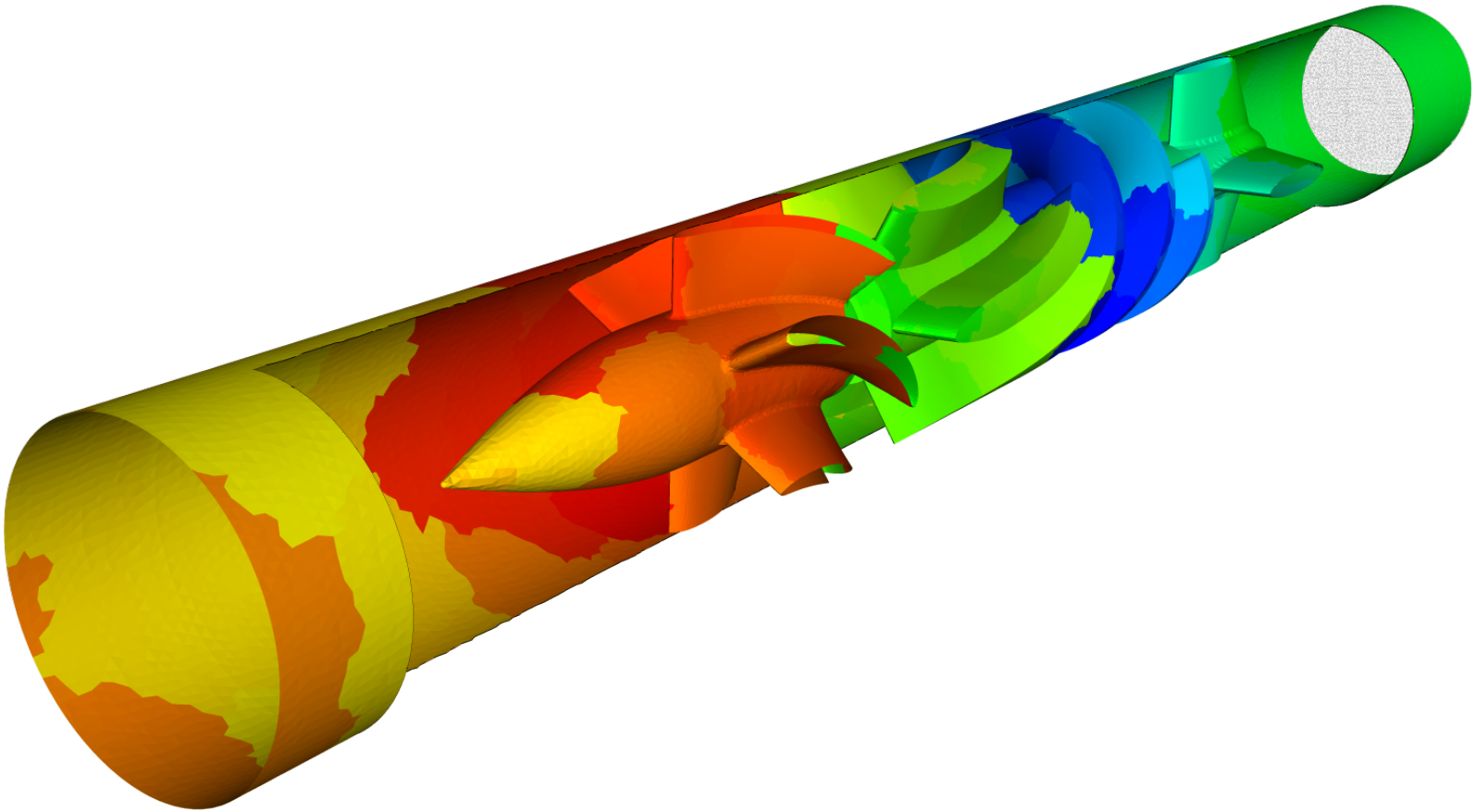
Discretization

TECHNISCHE UNIVERSITÄT DARMSTADT

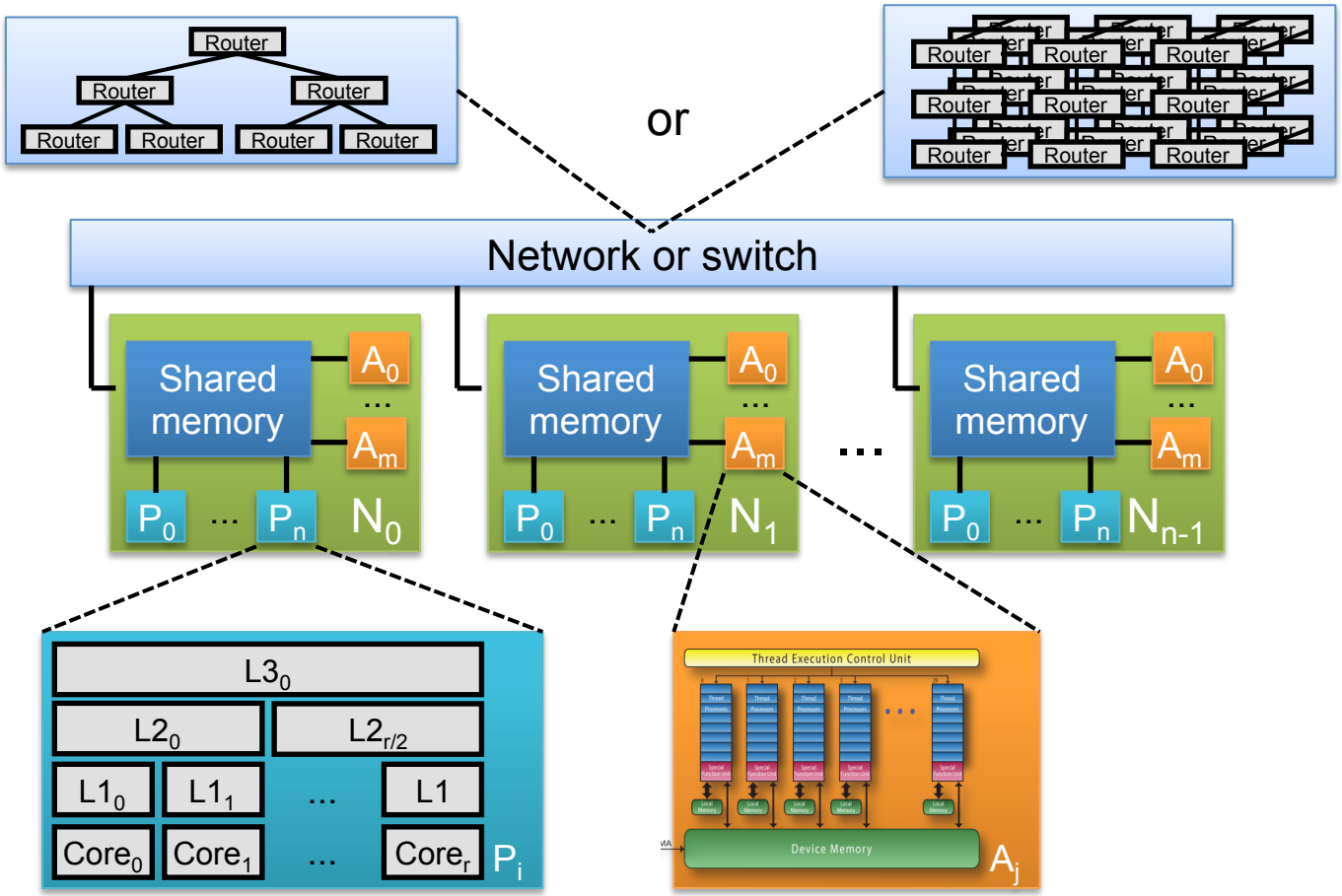


Partitioning

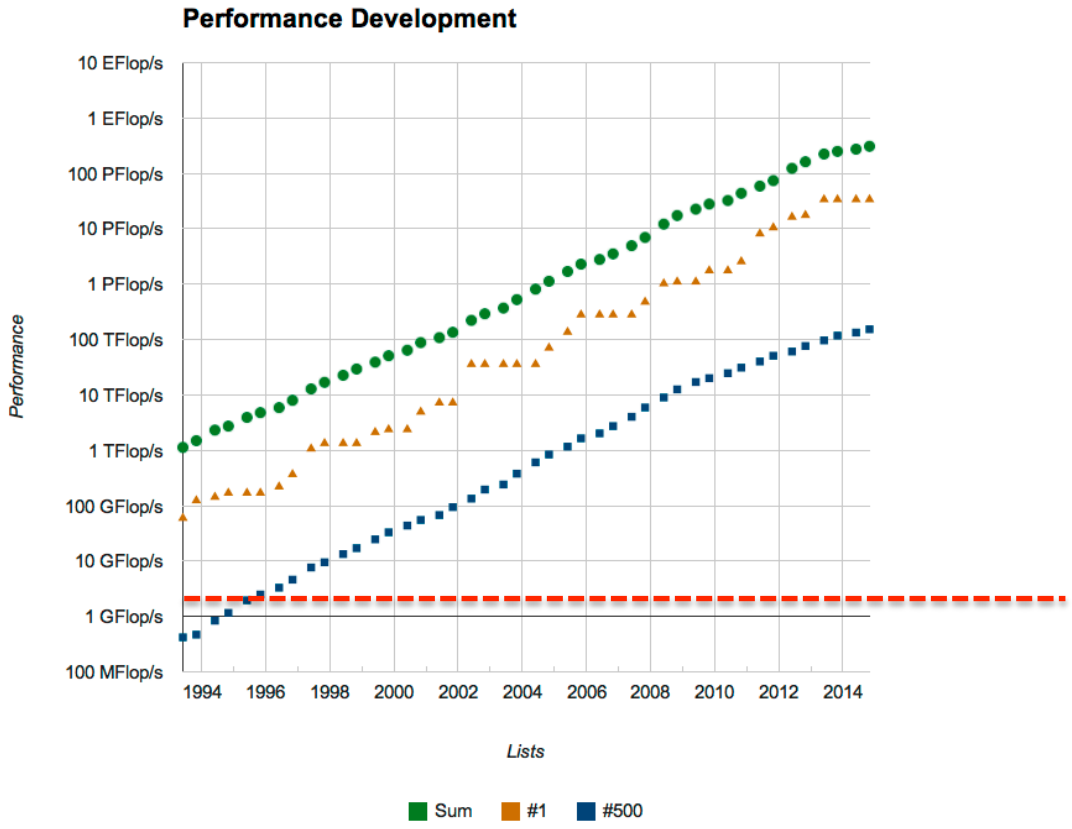
TECHNISCHE UNIVERSITÄT DARMSTADT



Typical supercomputer architecture



Top 500 supercomputers



Top 500 supercomputers - projection

TECHNISCHE UNIVERSITÄT DARMSTADT

Exascale: 10^{18} FLOP/s

TECHNISCHE UNIVERSITÄT DARMSTADT

Systems	2015 Tianhe-2	2020-2023	Difference Today & Exa
System peak	55 Pflop/s	1 Eflop/s	~20x
Power	18 MW (3 Gflops/W)	~20 MW (50 Gflops/W)	O(1) ~15x
System memory	1.4 PB (1.024 PB CPU + .384 PB CoP)	32 - 64 PB	~50x
Node performance	3.43 TF/s (.4 CPU +3 CoP)	1.2 or 15TF/s	O(1)
Node concurrency	24 cores CPU + 171 cores CoProc	O(1k) or 10k	~5x - ~50x
Node Interconnect BW	6.36 GB/s	200-400 GB/s	~40x
System size (nodes)	16,000	O(100,000) or O(1M)	~6x - ~60x
Total concurrency	3.12 M 12.48M threads (4/core)	O(billion)	~100x
MTTF	Few / day	Many / day	O(?)

#1

TECHNISCHE UNIVERSITÄT DARMSTADT

Tianhe-2 (MilkyWay-2)

National University of Defense Technology

Processor

- Intel Xeon
- Intel Xeon Phi

Cores: 3,120,000

Linpack performance: 33.9 PF

Theoretical peak: 54.9 PF

Power: 17.8 MW

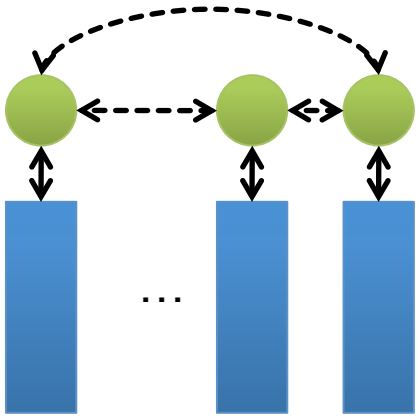
Memory: 1,024 TB

Tianhe-2

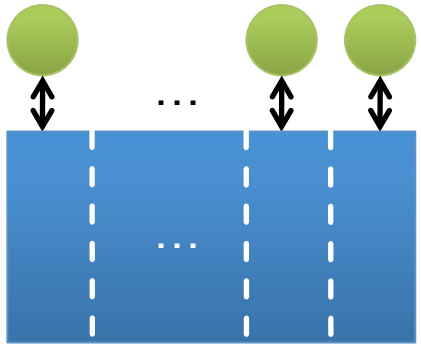
Focus: inter-node parallelism

TECHNISCHE UNIVERSITÄT DARMSTADT

- Distributed (private) memory & message passing
- Alternative: partitioned global address space (PGAS)
- Scalability (1000s of nodes)



Message passing



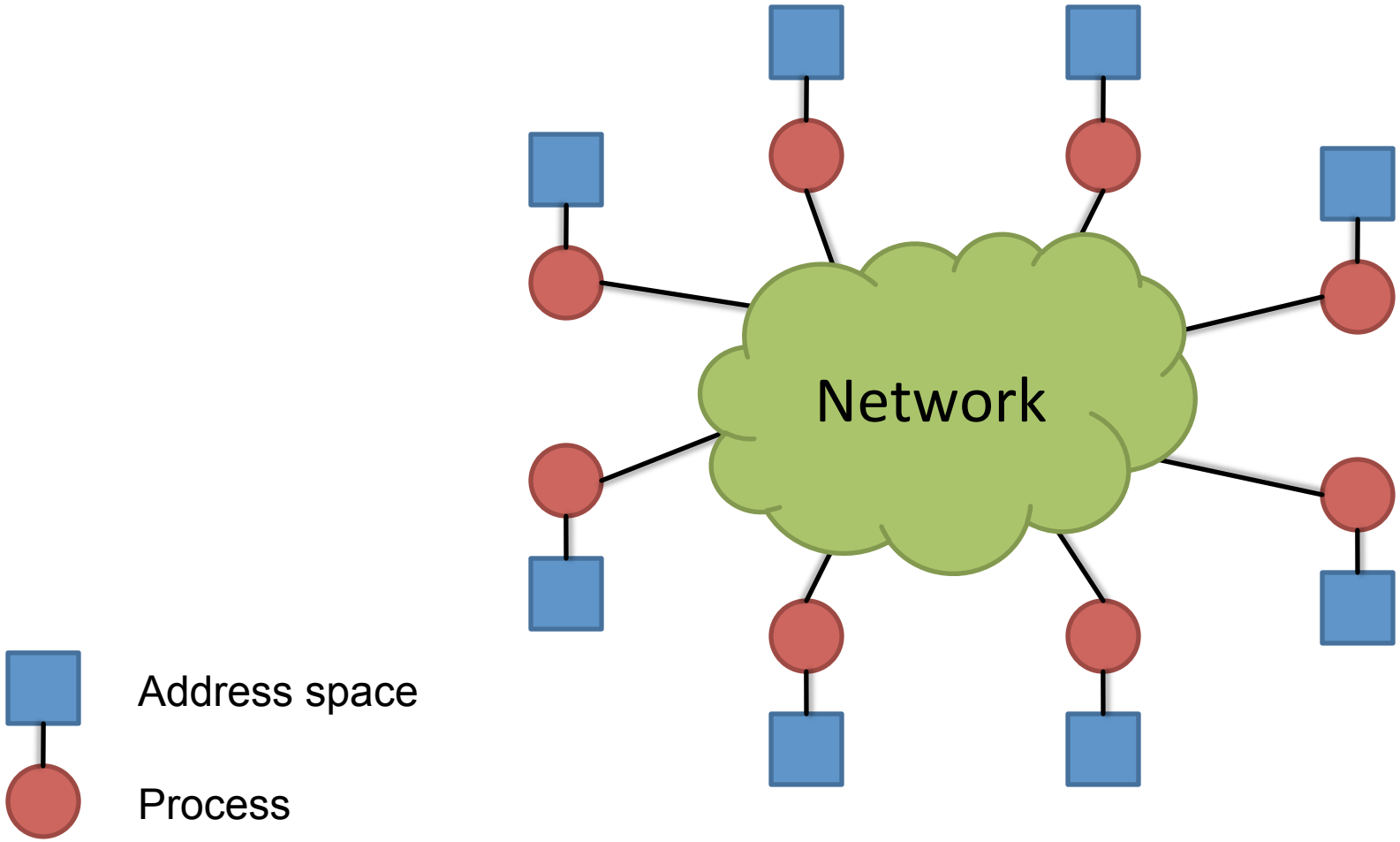
Partitioned global address space

Single Program Multiple Data (SPMD)

- Execution model underlying most parallel programming models
- The same program is executed on multiple processors
- Processes or threads are enumerated; each process or thread "knows" its number (ID)
- Case distinctions based on the process or thread number lead to different control flows (i.e., multiple instruction streams)

```
if (process_id == 42) then
  call do_something()
else
  call do_something_else()
endif
```

Message passing



Message passing (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Suitable for distributed memory
- Multiple processes each having their own private address space
- Access to (remote) data of other processes via sending and receiving messages (explicit communication)
  - Sender invokes send routine
  - Receiver invokes receive routine

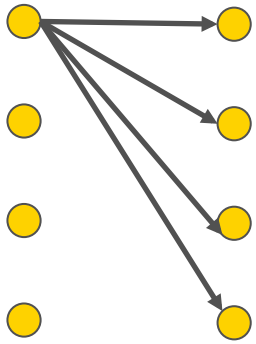
```
if (my_id == SENDER)
   send_to(RECEIVER, data);

if (my_id == RECEIVER)
   recv_from(SENDER, data)
```

- De-facto standard MPI: www.mpi-forum.org

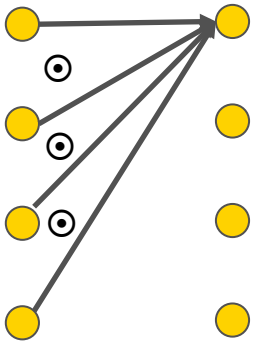
Group communication & computation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Example: broadcast, reduction
- Different flavors: 1 -> n, n -> 1, n->n



Broadcast



Reduction

- Manual implementation via point-to-point messages cumbersome and often suboptimal in terms of performance
- MPI offers a range of predefined collective operations
  - Embody recurring parallel communication / design patterns
  - Will study how to use them and their performance characteristics

Non-contiguous data

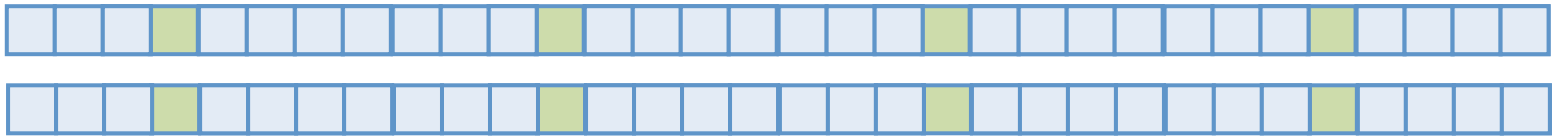
TECHNISCHE UNIVERSITÄT DARMSTADT

- Certain data types may lack contiguous memory representation
- Sending them across the network requires tedious packing and unpacking
- MPI data type concept avoids to do this manually

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Matrix

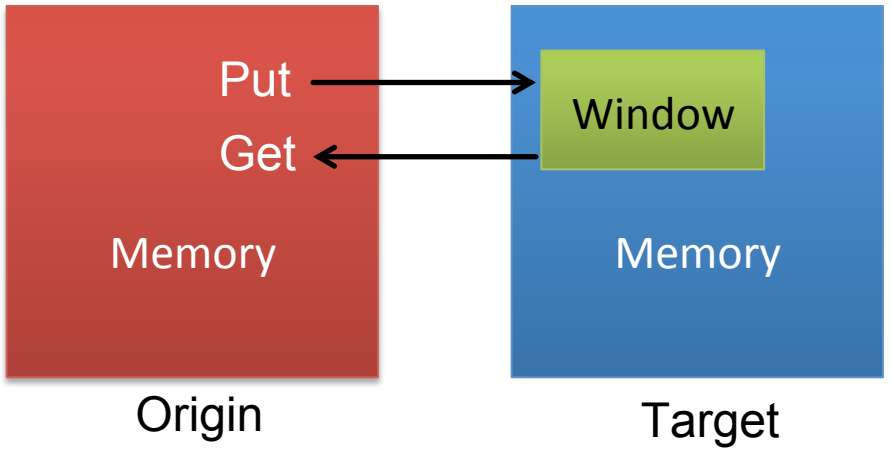
Memory layout in C (row major)



Remote memory access

TECHNISCHE UNIVERSITÄT DARMSTADT

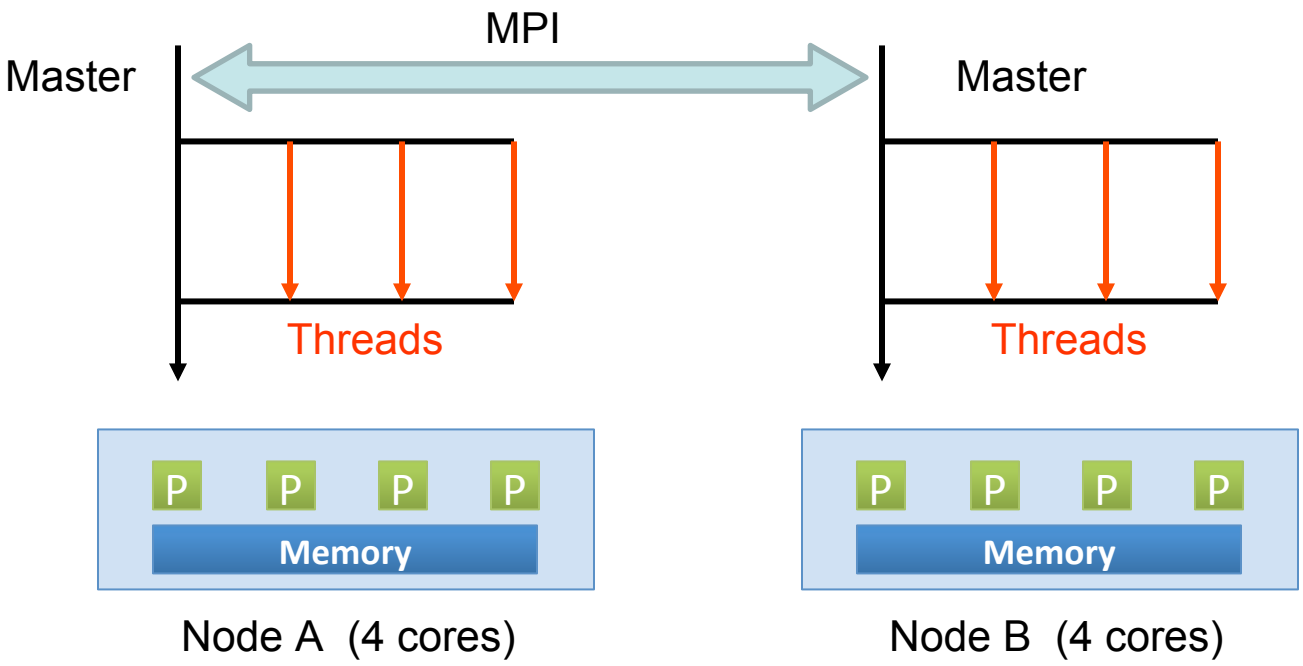
- Conventional message passing is two-sided
  - Send to destination process / receive from source process
  - Both processes specify message parameters
- Remote memory access
  - Also called one-sided communication
  - Parameters of data transfer are determined by one process only
  - Typically expressed through get and put operations
- Can be used to build powerful distributed data structures
- Underlying communication substrate for PGAS languages



Hybrid programming

TECHNISCHE UNIVERSITÄT DARMSTADT

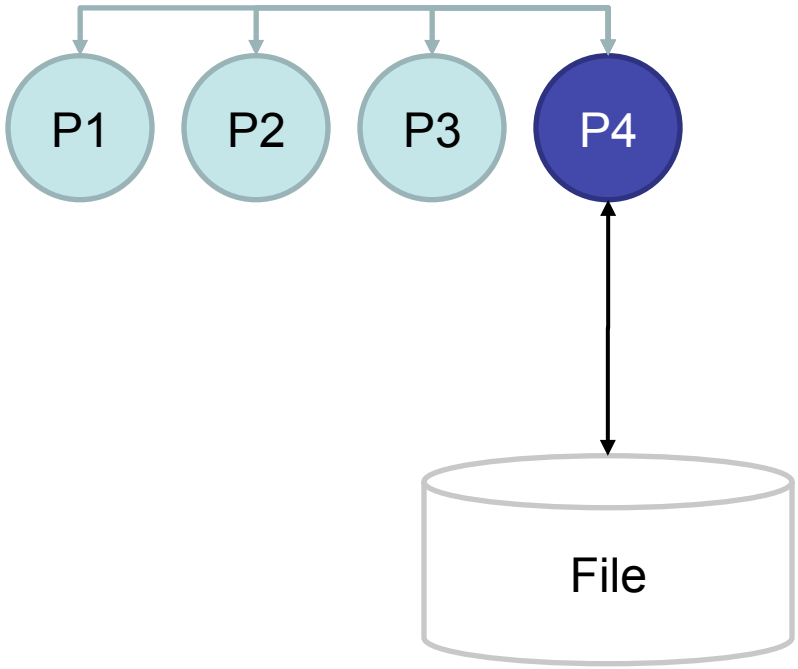
- Between processes parallelism via MPI
- Thread parallelism inside a process (e.g., via OpenMP)



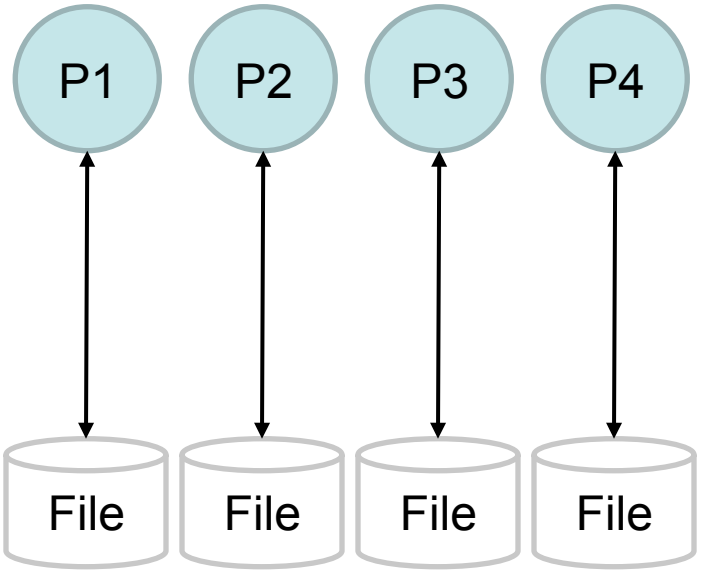
File access in parallel applications

TECHNISCHE UNIVERSITÄT DARMSTADT

Two traditional models



Sequential file access – all file accesses through a single process.



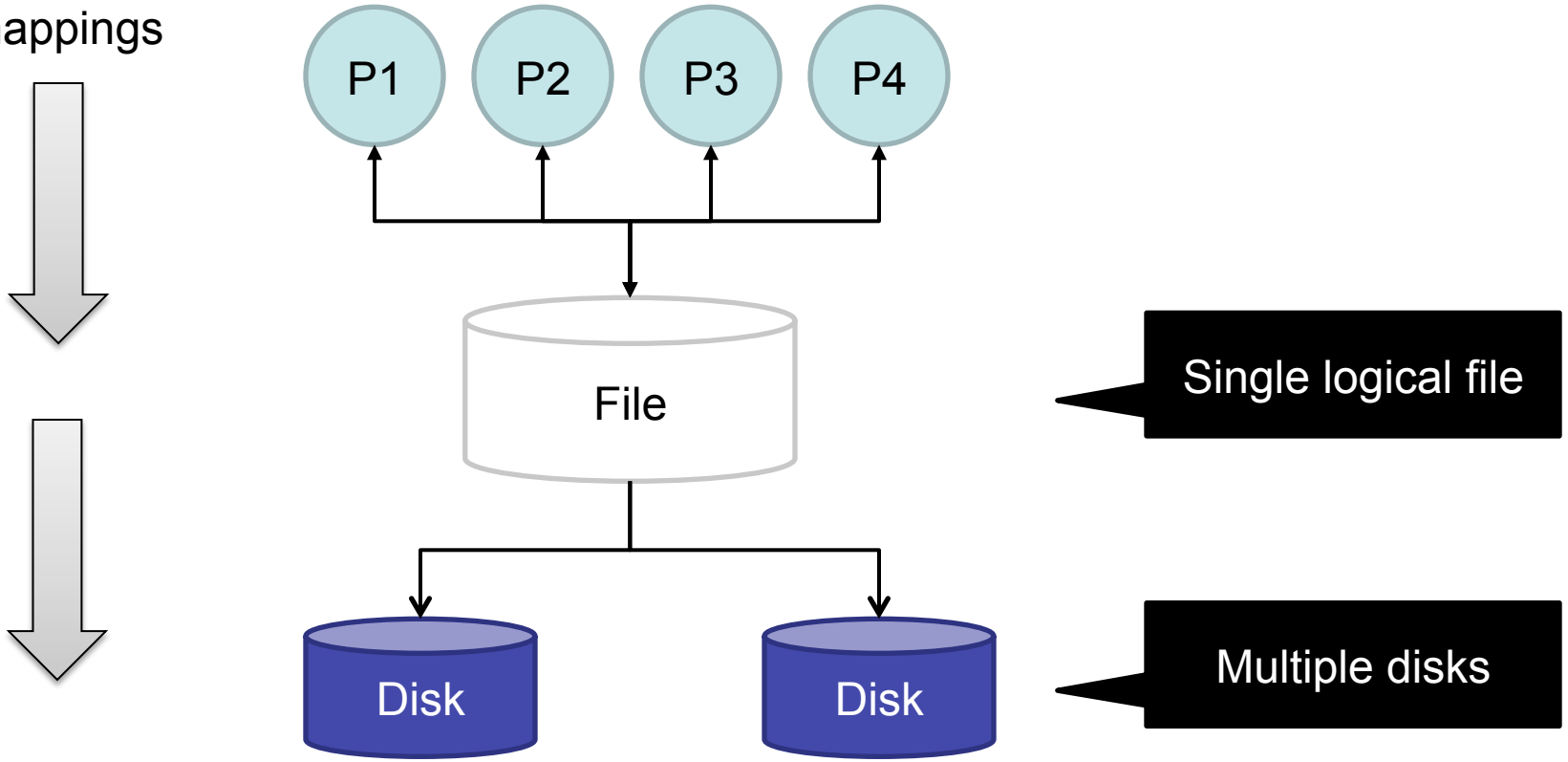
Multiple file access – each process writes its own file.

Parallel file I/O

TECHNISCHE UNIVERSITÄT DARMSTADT

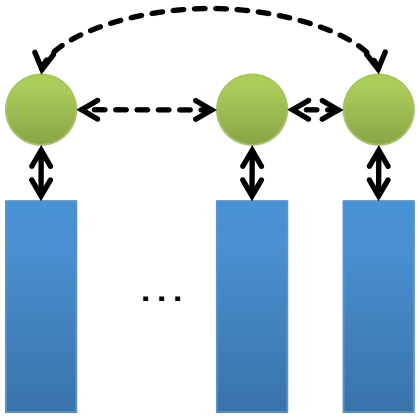
Multiple processes access the same file

2 mappings

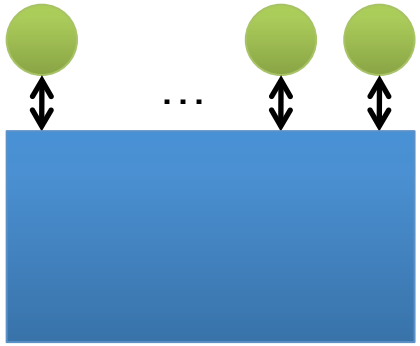


Parallel programming models

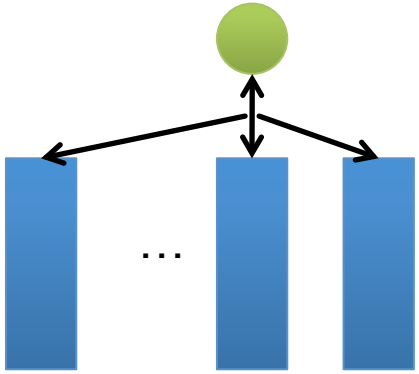
TECHNISCHE UNIVERSITÄT DARMSTADT



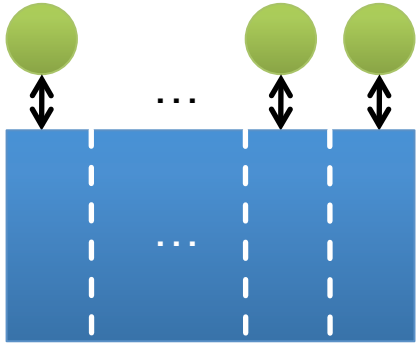
Message passing



Shared memory



Data parallel

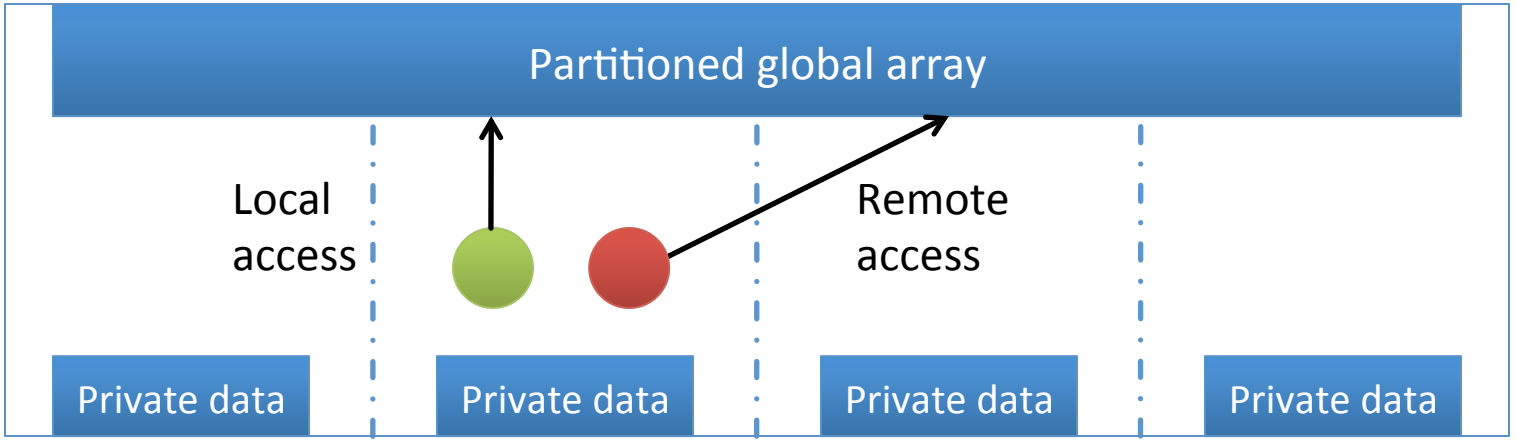


Partitioned global address space (PGAS)

Partitioned global address space (PGAS)

TECHNISCHE UNIVERSITÄT DARMSTADT

- The PGAS memory model allows any thread to read or write memory anywhere in the system
- It is partitioned to indicate that some data is local, whereas other date is further away (slower to access)



Unified Parallel C (UPC)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Extension of the C programming language
- Designed for high performance computing on large-scale parallel machines
- Provides uniform programming model for both shared and distributed memory hardware
- The programmer is presented with a single shared, partitioned address space
  - Variables may be directly read and written by any processor
  - But each variable is physically associated with a single processor
- SPMD model of computation in which the amount of parallelism is fixed at program startup time, typically with a single thread of execution per processor
- Implementation Berkeley UPC http://upc.lbl.gov/