Parallel and Distributed Systems / 1

Pavel Krömer,
Dept. of Computer Science,
VSB – Technical University of Ostrava

Agenda

- Course introduction and organization
- Motivation
- Basic concepts of parallelism

Introduction

Contact

- Location
  - office: FEI EA444
  - office hours: per e-mail/phone
  - office phone: +420597325898
- E-communication
  - pavel.kromer@vsb.cz

Course objectives

- Introduction to the basic concepts of parallel and distributed platforms and algorithms
  - full range of parallel architectures (edge, multicore, distributed (HPC), cloud, edge and fog)
- Working knowledge of basic parallel algorithms and parallelization strategies
  - methods to solve computationally extensive problems from different application areas
- Practical introduction of selected parallel languages / language extensions
  - practical development of parallel codes

Credit

- Active participation on lectures and seminars / tutorials
- Submission of a project on an assigned topic
  - list of indicative themes will be available
  - each topic individually approved with instructor
- Deadline
  - end of semester for final years

Motivation

Model	Release date	Price (USD)	Fab	Chiplets	Cores (Threads)	Core config[i]	Clock rate (GHz)		Cache in MB		
							Base	Boost	L1	L2	L3
Mainstream Enterprise											
9124	November 10, 2022	$1,083	TSMC N5	4 × CCD 1 × I/OD	16 (32)	4 × 4	3.0	3.7	1	16	64
9224		$1,825			24 (48)	4 × 6	2.5	3.7	1.5	24	
9254		$2,299				4 × 6	2.9	4.15			128
9334		$2,990			32 (64)	4 × 8	2.7	3.9	2	32	
9354		$3,420		8 × CCD 1 × I/OD		8 × 4	3.25	3.75			256
9354P		$2,730									
Performance Enterprise											
9174F	November 10, 2022	$3,850	TSMC N5	8 × CCD 1 × I/OD	16 (32)	8 × 2	4.1	4.4	1	16	256
9184X	June 13, 2023	$4,928					3.55	4.2			768
9274F	November 10, 2022	$3,060			24 (48)	8 × 3	4.05	4.3	1.5	24	256
9374F		$4,860			32 (64)	8 × 4	3.85	4.3	2	32	
9384X	June 13, 2023	$5,529					3.1	3.9			768
9474F	November 10, 2022	$6,780			48 (96)	8 × 6	3.6	4.1	3	48	256
Cloud & HPC											
9454	November 10, 2022	$5,225	TSMC N5	8 × CCD 1 × I/OD	48 (96)	8 × 6	2.75	3.8	3	48	256
9454P		$4,598									
9534		$8,803			64 (128)	8 × 8	2.45	3.7	4	64	
9554		$9,087					3.1	3.75			
9554P		$7,104									
9634		$10,304		12 × CCD 1 × I/OD	84 (168)	12 × 7	2.25	3.7	5.25	84	384
9654		$11,805			96 (192)	12 × 8	2.4	3.7	6	96	
9654P		$10,625									
9684X	June 13, 2023	$14,756					2.55	3.7			1152
9734		$9,600		8 x CCD 1 x I/OD	112 (224)	16 x 7	2.2	3.0	7	112	256
9754S		$10,200			128 (128)	16 x 8	2.25	3.1	8	128	
9754		$11,900			128 (256)						

AMD 5nm EPYC Genoa

- EPYC 9374F, announced 10. 11. 2022
- 5nm Zen 4 core architecture
- up to 96 cores/192 threads
- 3.85 GHz / 4.3 GHz, 256 MB L3 cache
- designed for liquid-cooled spaces

Model	Release date	Price (USD)	Fab	Chiplets	Cores (Threads)	Core config[i]	Clock rate (GHz)		Cache in MB		
							Base	Boost	L1	L2	L3
Mainstream Enterprise											
9124	November 10, 2022	$1,083	TSMC N5	4 × CCD 1 × I/OD	16 (32)	4 × 4	3.0	3.7	1	16	64
9224		$1,825			24 (48)	4 × 6	2.5	3.7	1.5	24	
9254		$2,299				4 × 6	2.9	4.15			128
9334		$2,990			32 (64)	4 × 8	2.7	3.9	2	32	
9354		$3,420		8 × CCD 1 × I/OD		8 × 4	3.25	3.75			256
9354P		$2,730									
Performance Enterprise											
9174F	November 10, 2022	$3,850	TSMC N5	8 × CCD 1 × I/OD	16 (32)	8 × 2	4.1	4.4	1	16	256
9184X	June 13, 2023	$4,928					3.55	4.2			768
9274F	November 10, 2022	$3,060			24 (48)	8 × 3	4.05	4.3	1.5	24	256
9374F		$4,860			32 (64)	8 × 4	3.85	4.3	2	32	
9384X	June 13, 2023	$5,529					3.1	3.9			768
9474F	November 10, 2022	$6,780			48 (96)	8 × 6	3.6	4.1	3	48	256
Cloud & HPC											
9454	November 10, 2022	$5,225		8 × CCD 1 × I/OD	48 (96)	8 × 6	2.75	3.8	3	48	256
9454P		$4,598									
9534		$8,803			64 (128)	8 × 8	2.45	3.7	4	64	
9554		$9,087					3.1	3.75			
9554P		$7,104									
9634		$10,304							5.25	84	384
9654		$11,8									
9654P		$10,0							6	96	
9684X	June 13, 2023	$14,7						3.7			1152
9734		$9		8 × CCD 1 x I/OD		x 7	2.2	3.0	7	112	256
9754S		$10,200			128 (128)	16 x 8	2.25	3.1	8	128	
9754		$11,900			128 (256)						

~ 10.000 EUR on a CZ e-shop

AMD 5nm EPYC Bergamo

- announced June 2023
- 5nm Zen 4 core architecture
- 128 cores and 256 threads
- 2.2(5) GHz / 3.(1) GHz
- giant TDP (up to 320-360W)
- clock rate vs core count opt. versions

Zen 5 microarchitecture in 2024

NVIDIA TeslaV100 Tensor Core GPU (Volta)

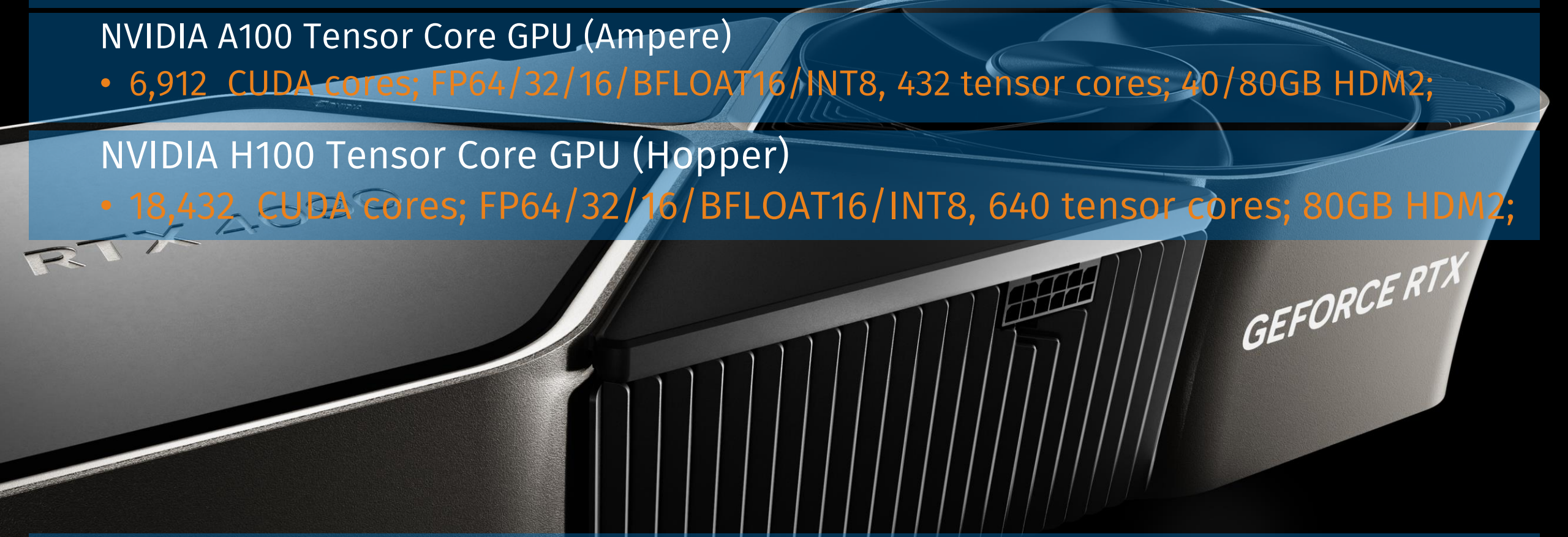
- 5,120 CUDA cores; 640 Tensor cores; 32GB HDM2;
- Deep learning: TensorFlow, PyTorch, theano, Caffe2; Applications: OpenFoam etc.

NVIDIA A100 Tensor Core GPU (Ampere)

- 6,912 CUDA cores; FP64/32/16/BFLOAT16/INT8, 432 tensor cores; 40/80GB HDM2;

NVIDIA H100 Tensor Core GPU (Hopper)

- 18,432 CUDA cores; FP64/32/16/BFLOAT16/INT8, 640 tensor cores; 80GB HDM2;

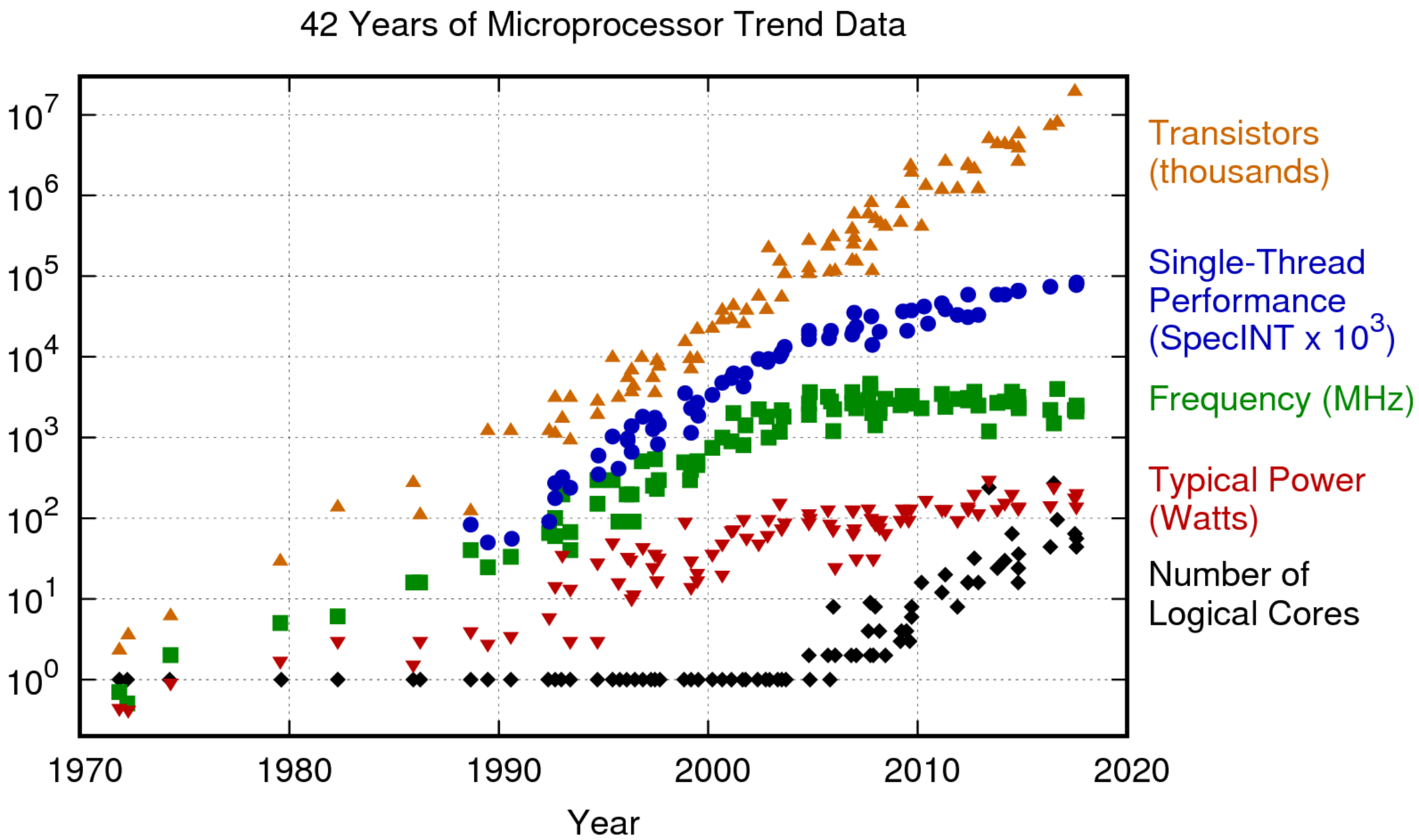


TITAN RTX (2019)

- 4,608 CUDA cores; 576 Tensor cores; 24 GB GDDR6; NVIDIA’s CUDA-X AI SDK; 70,000 CZK

RTX 4090 (2023)

- 16,384 CUDA cores; 512 Tensor cores (4th gen); 24 GB GDDR6X; cca 54,000 CZK

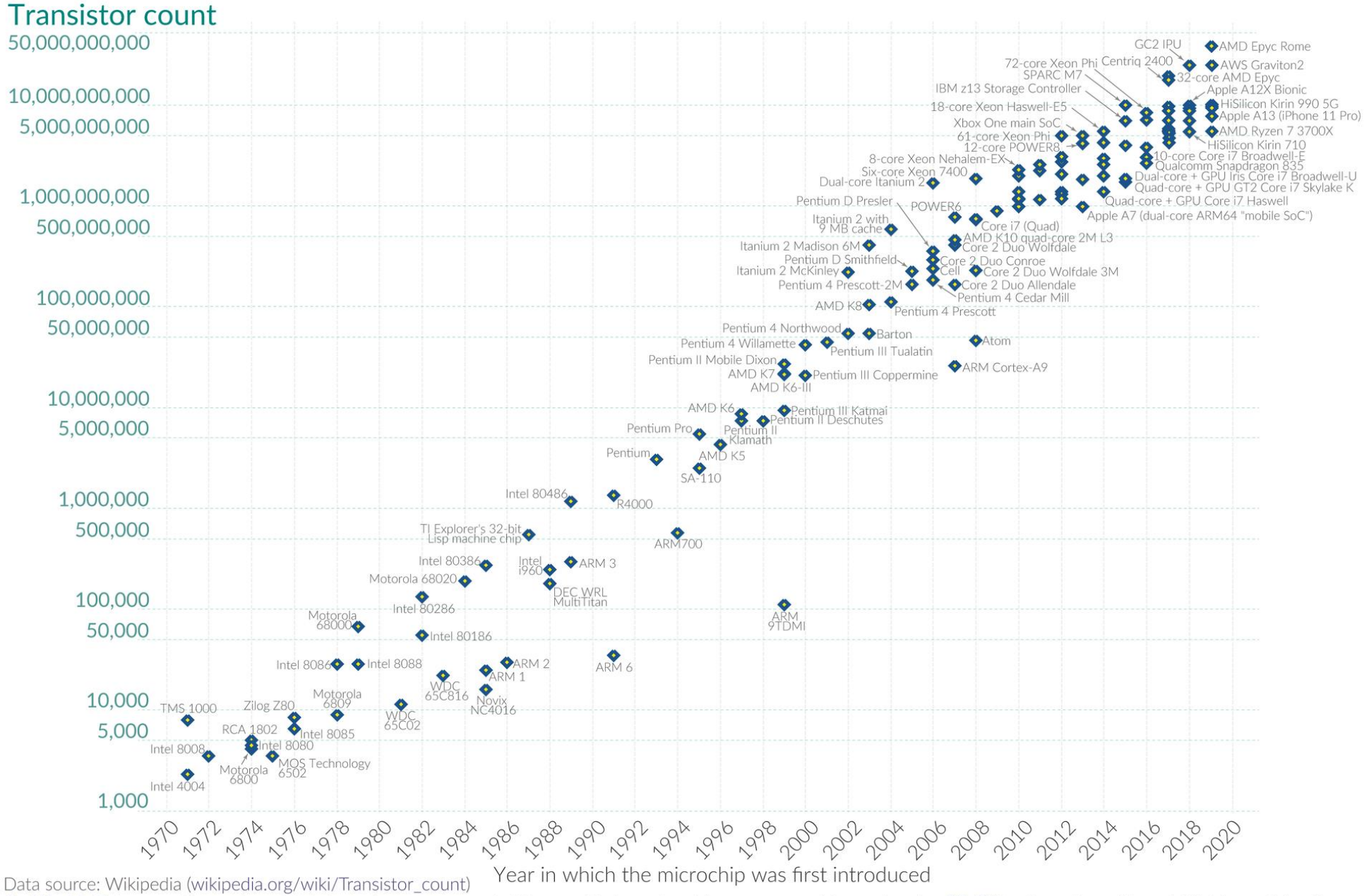


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

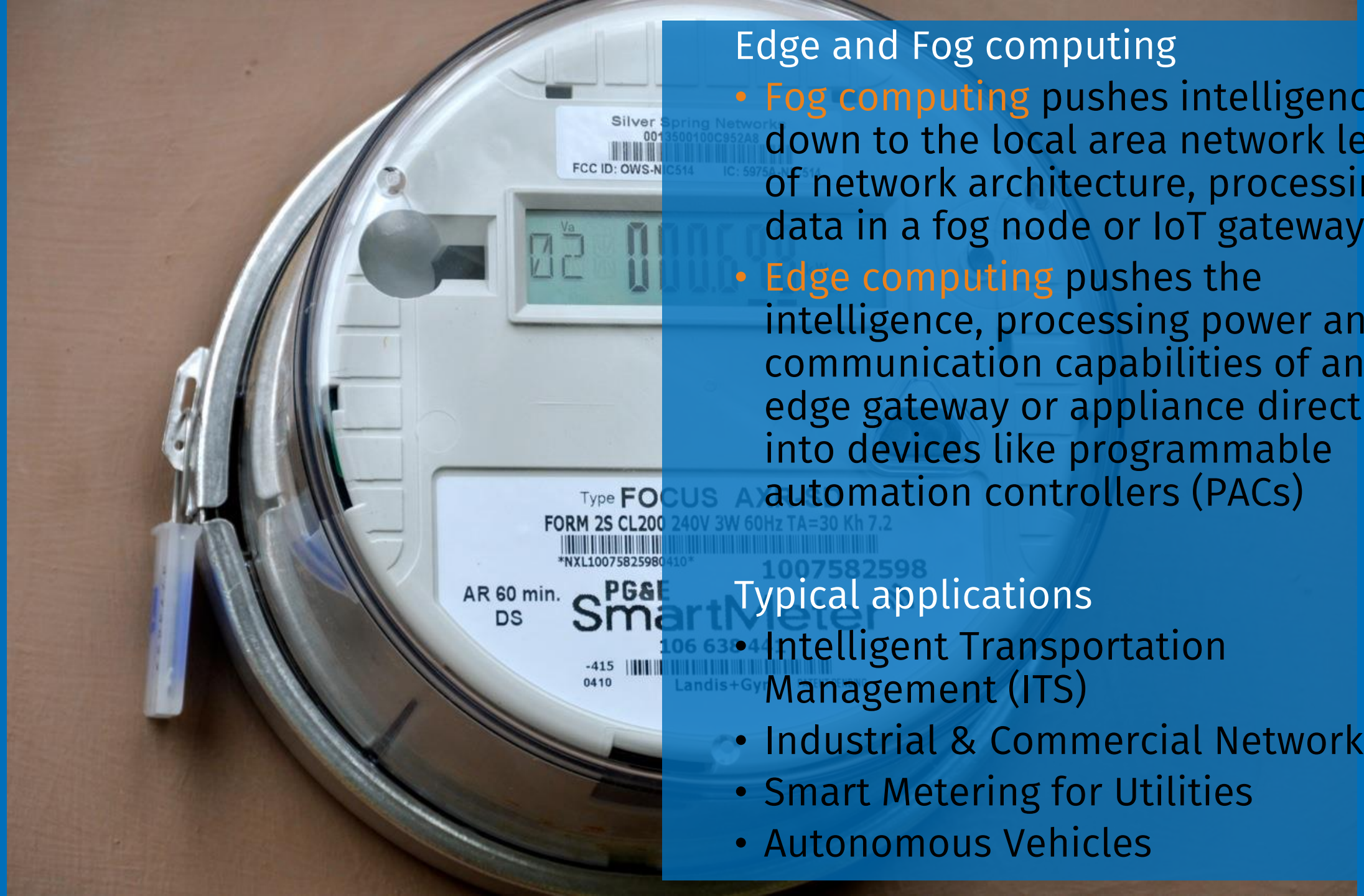
Moore's Law: The number of transistors on microchips doubles every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)



Edge and Fog computing

- Fog computing pushes intelligence down to the local area network level of network architecture, processing data in a fog node or IoT gateway.
- Edge computing pushes the intelligence, processing power and communication capabilities of an edge gateway or appliance directly into devices like programmable automation controllers (PACs)

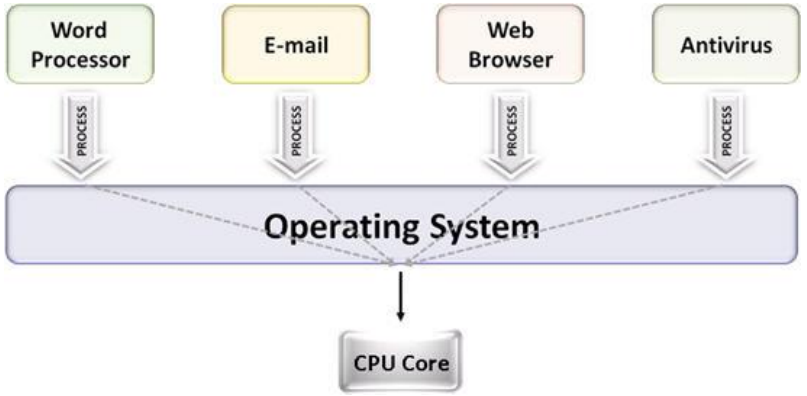
Typical applications

- Intelligent Transportation Management (ITS)
- Industrial & Commercial Networking
- Smart Metering for Utilities
- Autonomous Vehicles

Parallelism and Concurrency

Multiprocessing and multithreading

- Multitasking OS
  - Allows the execution of more than one process (computer programme + resources in memory) at a time



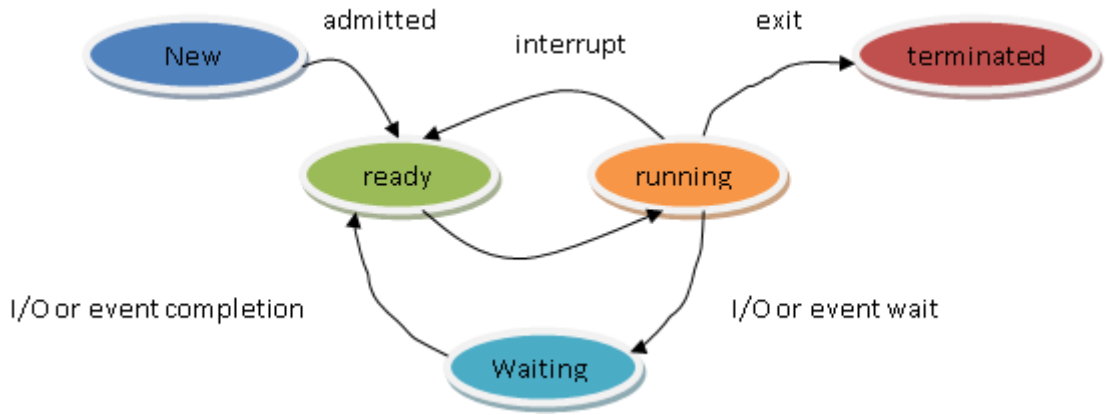
  - The number of tasks can exceed the number of physical cores / threads
  - Multitasking OS has all the components necessary to handle this situation by task switching

Multiprocessing and multithreading

- Process
  - runnable software is in the process model organized into sequential processes
  - a running program, including the current values of the program counter, registers, and variables. It has a virtual CPU and shares physical CPU(s) via task-switching
  - processes are independent (isolated by the OS) and communicate via inter-process communication (IPC)

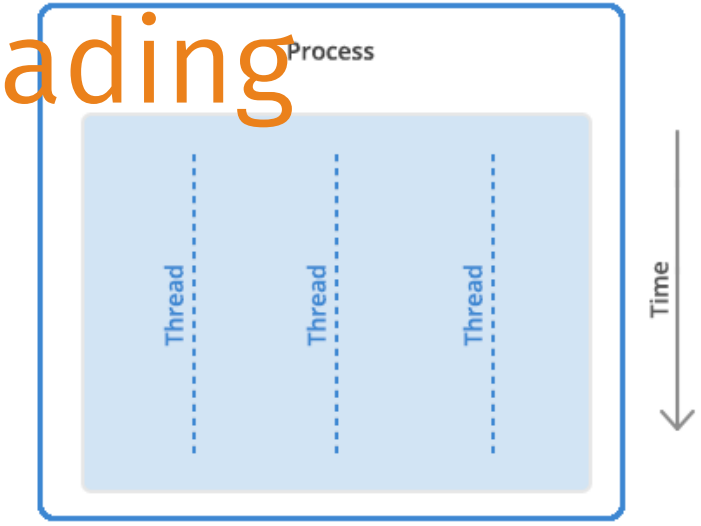
- Task-switching
  - when more processes require a shared resource (CPU), fast task switching can give an illusion of parallelism (aka. pseudo-parallelism, concurrency)

Multiprocessing and multithreading



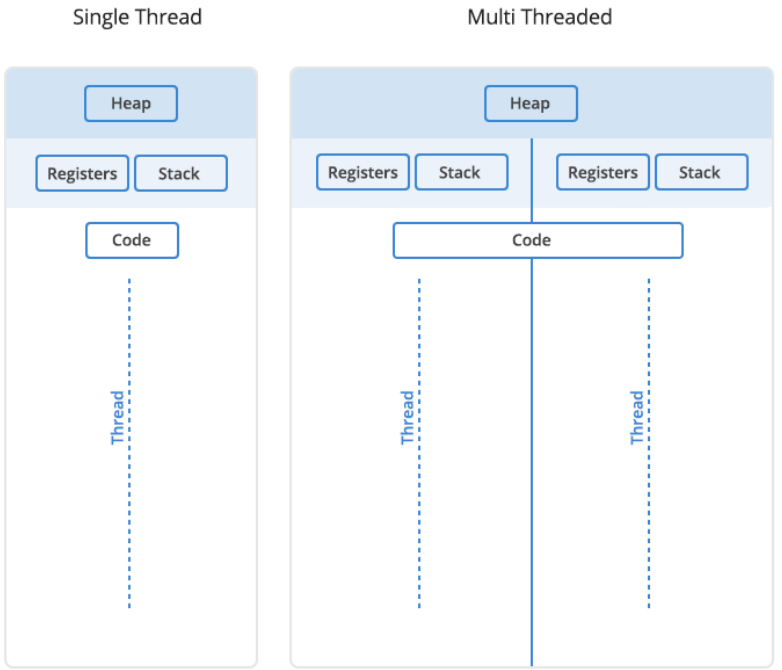
- Process states
  - a process can be in one of several internal states:
    - Running -> Waiting: process blocks for input
    - Waiting -> Ready: input becomes available
    - Ready -> Running: scheduler picks the process and allocates CPU
    - Running -> Ready: scheduler removes process from CPU
  - all is done according to a specific scheduling algorithm

Multiprocessing and multithreading



- Thread
  - a unit of (parallel) execution within a process (1-N)
  - each thread in a process shares its resources (that makes them more lightweight)
  - communication between threads is cheaper than IPC
  - problem of one thread can easily affect other threads, too (and the whole process might have to be killed)

- Properties
  - each thread has its own program counter, registers, stack
  - and shares data space, address space, resources (and limits)



Threads vs. processes

Process

- Processes are heavyweight operations
- Each process has its own memory space
- Inter-process communication is slow as processes have different memory addresses
- Context switching between processes is more expensive
- Processes don’t share memory with other processes

Thread

- Threads are lighter weight operations
- Threads use the memory of the process they belong to
- Inter-thread communication can be faster than inter-process communication because threads of the same process share memory with the process they belong to
- Context switching between threads of the same process is less expensive
- Threads share memory with other threads of the same process

Inter-process communication (IPC)

- Communication between cooperating processes, preferably in a well-structured way and not using interrupts. It must deal with:
  - information passing,
  - clash of activities
  - dependencies

- Usually implemented by
  - shared memory
  - message passing
  - -> pipes and named pipes; message queueing; semaphores; shared memory; and sockets

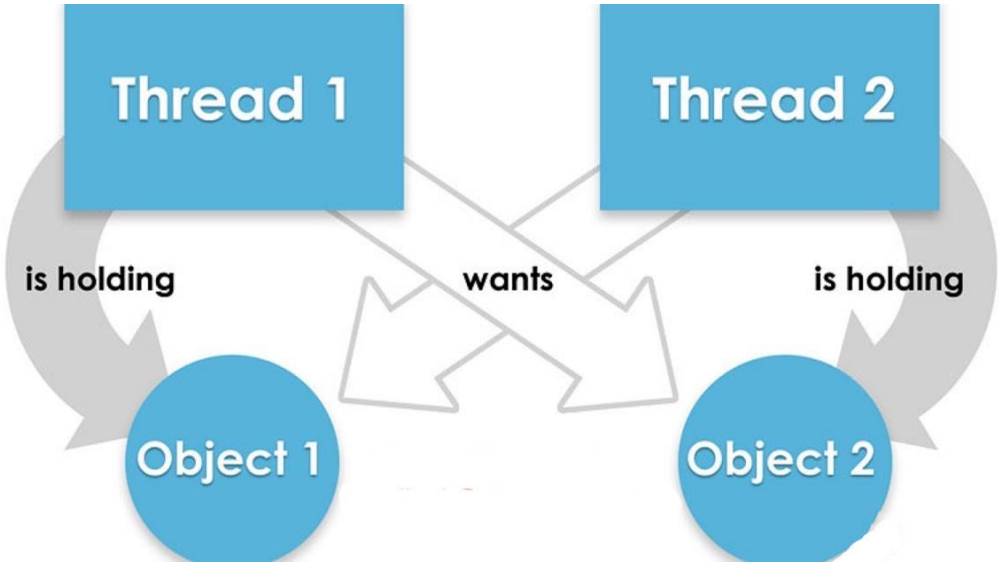
- IPC concepts and issues
  - synchronization, critical regions, mutual exclusion, semaphores, mutexes, monitors, barriers
  - race conditions, deadlocks, busy waiting

Inter-process communication (IPC)

- Deadlocks
  - a set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.
  - a situation in which one process, *A*, waits for a resource exclusively used by another process, *B*, which, in turn, waits for another resource owned by *A*. Both processes wait forever.

  - Conditions for a deadlock are:
    - mutual exclusion,
    - hold and wait,
    - no preemption, and
    - circular wait

  - Dealing with deadlocks:
    - do nothing
    - detection and recovery
    - avoidance by careful resource allocation
    - preventing (one) of the conditions for deadlock



Parallelism vs. concurrency

- Parallelism vs. concurrency
  - Parallelism: using multiple processors/cores running at the same time. Property of the machine (parallel machine).
  - Concurrency: non-determinacy due to interleaving threads. Property of the application (concurrent tasks).

		Concurrency	
		sequential	concurrent
Parallelism	serial	Traditional programming	Traditional OS
	parallel	Deterministic parallelism	General parallelism

Parallel vs. sequential programme

Sequential vector add

```
void vector_add(double* a, double *b, const unsigned int len)
{
  for (unsigned int i = 0; i < len; i++)
    a[i] += b[i];
}

int main (void)
{
  const unsigned int len = 200;

  double a[len];
  double b[len];

  rand_vec(a, len);
  rand_vec(b, len);

  vector_add(a, b, len);

  return 0;
}
```

Parallel vector add (pseudocode)

```
void vector_add_p(double* a, double *b, const unsigned int len)
{
  const unsigned int from = THREAD_ID == 1 ? 0: len / 2;
  const unsigned int to = THREAD_ID == 1 ? len/2 : len;
  for (unsigned int i = 0; i < len; i++)
    a[i] += b[i];
}

int main (void)
{
  const unsigned int len = 200;
   double a[len];
  double b[len];

  rand_vec(a, len);
  rand_vec(b, len);

  vector_add_p(a, b, len);    // Parallel section
  return 0;
}
```

Summary

- parallel code is intuitive, but requires additional attention and brings more challenges, including IPC, scalability, portability

Parallel vs. sequential programming

- Summary
  - parallel code is intuitive*, but requires additional attention
  - brings more challenges, including IPC, scalability, portability
- Parallel programming is a two-step process
  - design a work-efficient, low-span parallel algorithm
  - implement it on the target hardware
- In reality: many systems require different code to implement the algorithm efficiently
  - huge effort to generate efficient portable parallel code.

* a manual implementation of Quicksort in MPI can be 1700 lines of code, and about the same in CUDA

Parallel vs sequential programming

- Problem solving based on parallel thinking
  - Recognizing true dependences (cw. sequential programming)
  - Parallel algorithm design
    - operations on aggregates: map/reduce/scan
    - divide & conquer, contraction
    - viewing computation as DAG (based on dependences)