INF236 – Parallel Programming

Lecturer:	Fredrik Manne	Fredrik.Manne@uib.no	Room 3145
TA:	FM & Kristian Sørdal	krisor99@gmail.com	

Classes:	Tuesday	08:15-10:00	Active room 3
	Thursday	10:15-12:00	

Exercises: Monday 08:15-10:00 Room 30202 (Rips)

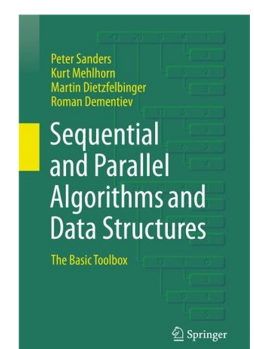
Grading:	3 compulsory programming projects	50% of total grade
	Final exam June 3rd, 09:00	50% of total grade

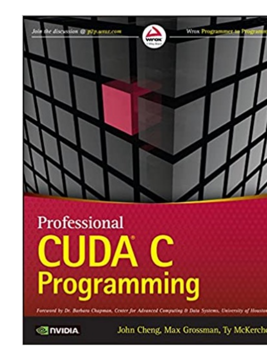
Textbooks:

- *Sequential and Parallel Data Structures and Algorithms, The Basic Toolbox* by Sanders, Mehlhorn, Dietzfelbinger, and Dementiev

- CUDA Programming : *Professional CUDA C Programming* by Cheng, John, et al.

- Handouts + online material





Information: http://mitt.uib.no

More information

Compulsory projects are individual work, unless otherwise specified.

This is strictly enforced!!!

Marcos Alonso Pardo
Simon Andersen
Thorgal Blanco
Lars Bysheim
Laxmi Dhital
Østen Vestby Edvardsen
Vegard Haugland
Jonas Haukenes
Thomas Svendsen Holden
Oskar Aaltvedt Høgseth
Magnus Haaland
Gard Heine Kalland
Sophie Koenig
Felix Mariendal Kaasa
Tarjei Landøy
Fredrik Lauritzen
Mariann Teigland Lepsøy

Moritz Nicolas Julian Makowsk
Olav Høysæther Opheim
Jorge Pilan Saez Benito
Sebastian Lillevik Refsnes
Berenika Richterova
Maya Alexandra Robbestad
Jørgen Mykleby Storum
Steffen Storøy
Josef Sykora
Markus Sandvær Sylta
Kristian Sandvik Sørdal
Lars Martin Taraldset
Stian Aas Trohaug
Carlos Vega de Alba
Patrik Welters
Yuhe Zhang

Introduction

- Why we need powerful computers
- Trends in CPU development
- Why parallel computing is the only way to achieve performance
- Challenges when writing parallel algorithms

But first some terminology..

Big numbers: 10^3 kilo, 10^6 mega, 10^9 giga, 10^{12} tera, 10^{15} peta, 10^{18} exa

Small numbers: 10^{-3} milli, 10^{-6} micro, 10^{-9} nano, 10^{-12} pico, 10^{-15} femto, 10^{-18} at

A computer running at 1 GHz:
- Performs 1 giga operations per second.
- Each operation is taking 1 nanosecond.
- Light travels 30cm in 1 nanosecond.

Flops: Number of 64-bit floating-point operations per second

One operation consists of one multiply and one add: $a = b + c*d$

So why do we need powerful computers?

Scientific Applications

Simulation: The third pillar of science

• Traditional scientific and engineering paradigm:

1) Do theory or paper design.

2) Perform experiments or build system.

• Limitations:

-Too difficult -- build large wind tunnels.

-Too expensive -- build a throw-away oil platform.

-Too slow -- wait for climate or galactic evolution.

-Too dangerous -- weapons, drug design, climate experimentation.

• Computational science paradigm:

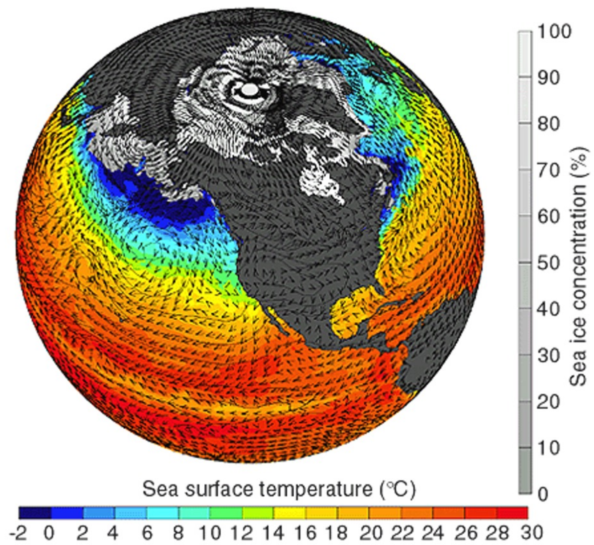
Use high performance computer systems to simulate the phenomenon

Based on known physical laws and efficient numerical methods.

Weather and Climate Simulations

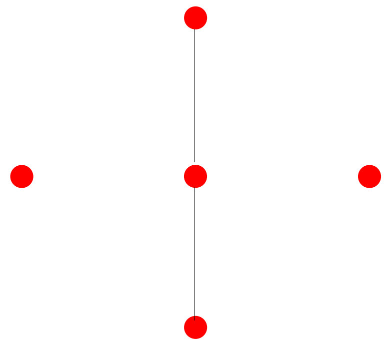
Need model of interacting complex systems:

- Air: Wind, temperature, pressure, moisture, clouds
- Sea: Temperature, currents, salinity
- Land: Temperature, terrain
- Other: Sun, season, earth rotation, etc.



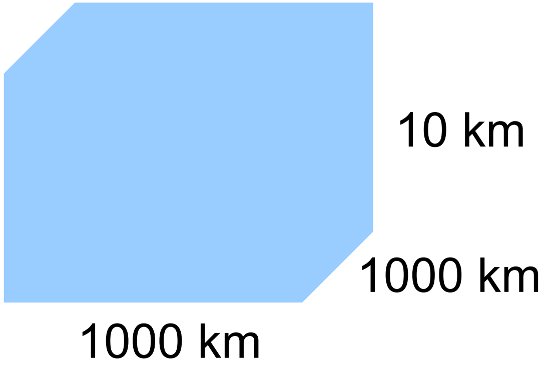
Simulate continuous system:

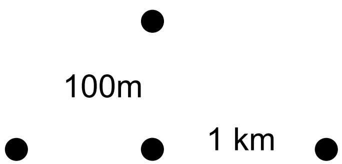
- Discretize system using a grid model
- What happens in one point is determined by what just happened in the surrounding points
- Apply initial conditions and run



A Weather Model

- Simulate weather for 24 hours for a volume of 1000km x 1000km and of height 10km.
- Evenly spaced calculation points 1km apart and 100m above each other.
- Must make 100 calculations per point.
- Simulate for one day with 1.5 minute intervals ≈ 1000 time steps.





Total number of calculations = 1000 * 1000 * 100 * 100 * 1000 = 10^{13} operations.

Computer running at 2GHz:
- One operation per clock cycle,
- $2 * 10^9$ operations per second
- Requires $10^{13} / (2 * 10^9)$ = 5000 seconds ≈ 1.5 hours

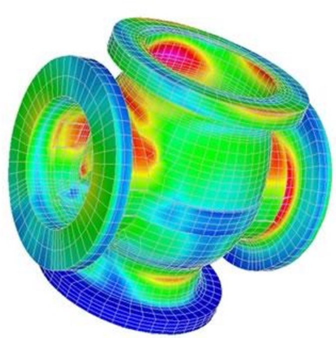
Refined mode: Put calculation points 100m apart in horizontal direction.
- Increases number of points to 10000 * 10000 * 100 = 10^{10}
- Requires $100*1000*10^{10} / (2 * 10^9) = 0.5 * 10^6$ seconds ≈ 139 hours > 5 days

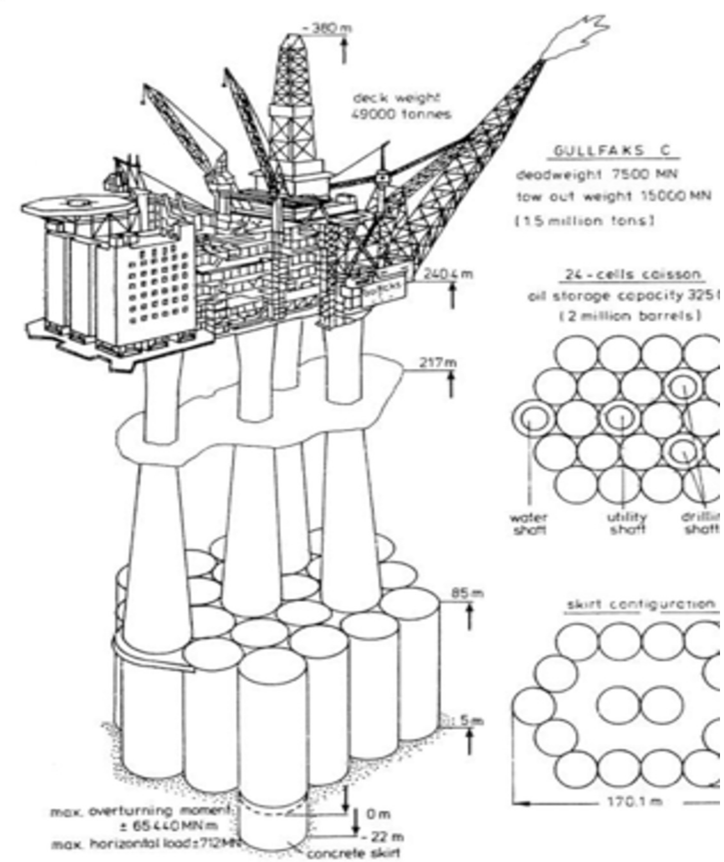
Engineering and Construction

Discretize object depending on geometry.

Apply various external forces and simulate.

Each time step typically requires solution of system of partial differential equations (PDE).



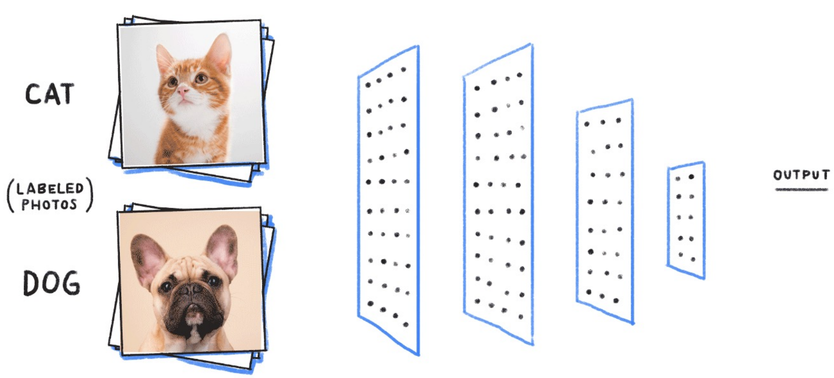


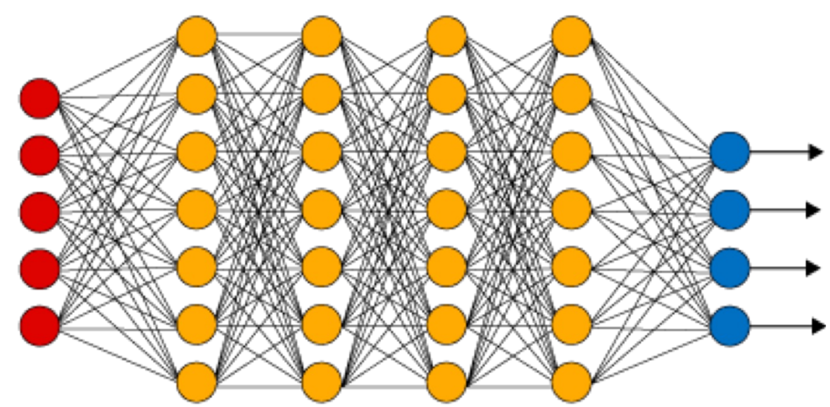
Deep Learning

Machine learning method for classifying input

System is trained on large data sets, computationally demanding

Proposed around 1980, only became viable with the use of parallel processing





3D Computer Games

Maintains a 3D model and renders it on a 2D screen

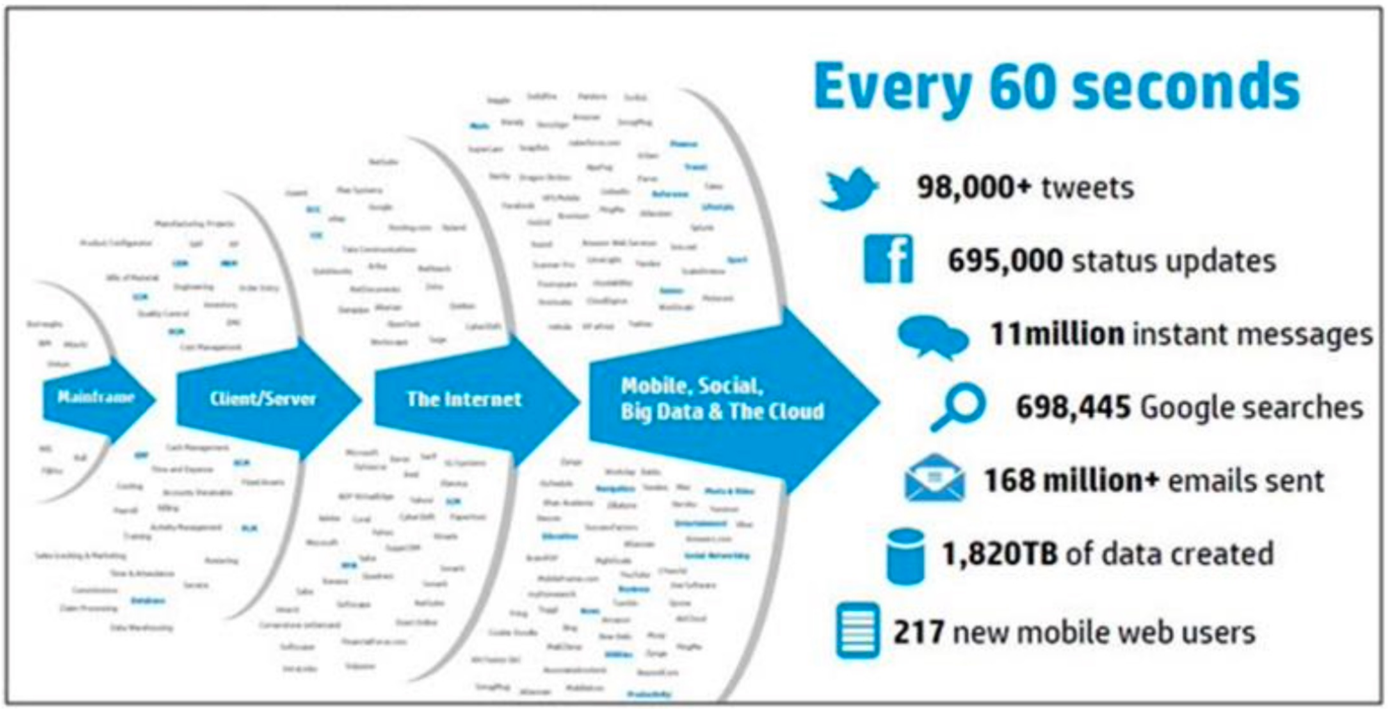
Each pixel can be computed simultaneously

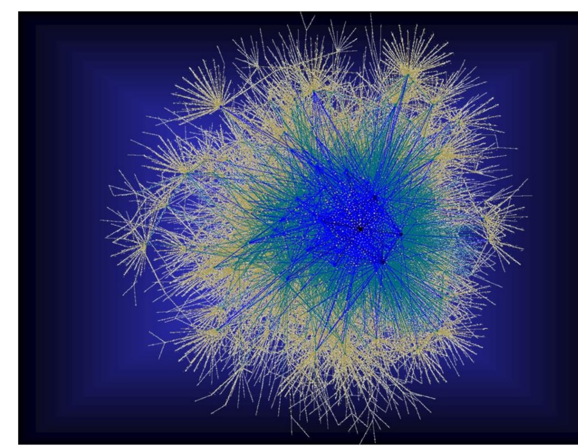
Driving force behind development of Graphical Processing Units (GPUs)

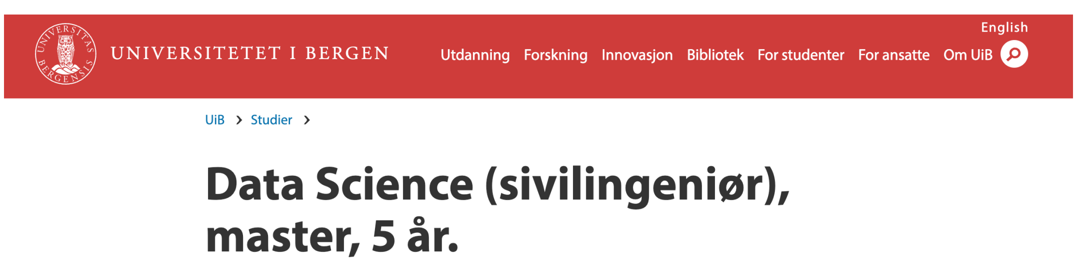
1992

2021

Data Explosion



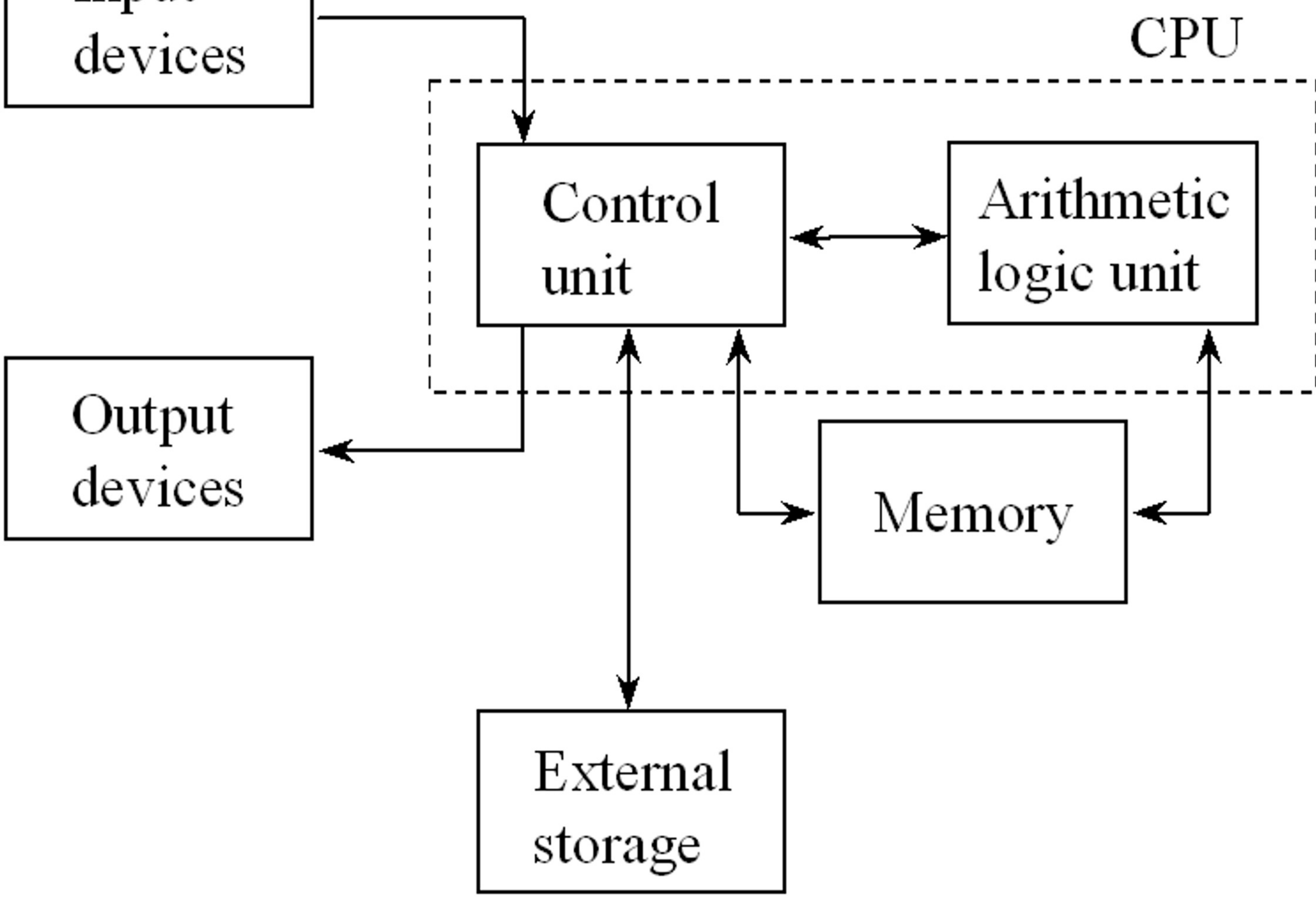




Trends in CPU development

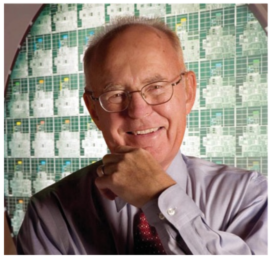
The Von Neumann Architecture

Clock speed is measured in Hertz (oscillations per second) ≈ Operations per second



Exponential Growth

Moore's law [1965]: The number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years.



$$T = c * 2^{0.5k}$$

Applies to other areas as well:

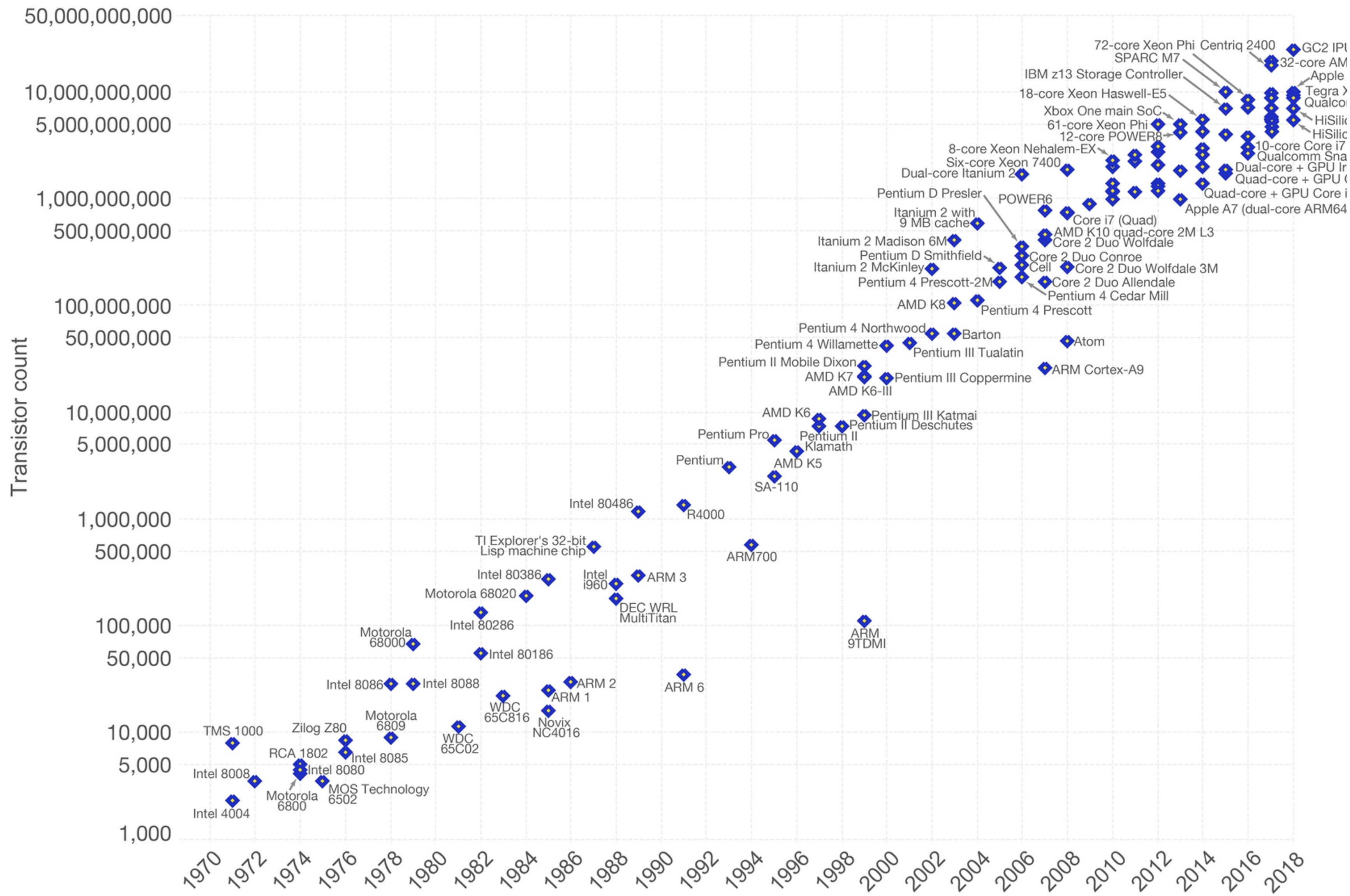
- The clock rate of computers doubles every two years (used to be true).

- The amount of memory and the compute power of a computer, as well as the hard disk storage per dollar, grows exponentially.

- The capacity of optical fiber cables is doubled every nine months

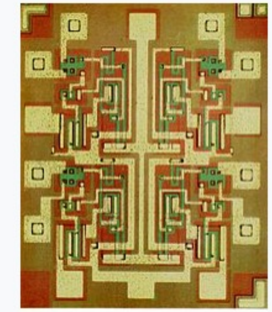
Moore’s Law – The number of transistors on integrated circuit chips (1971-2018)

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



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Semiconductor device fabrication

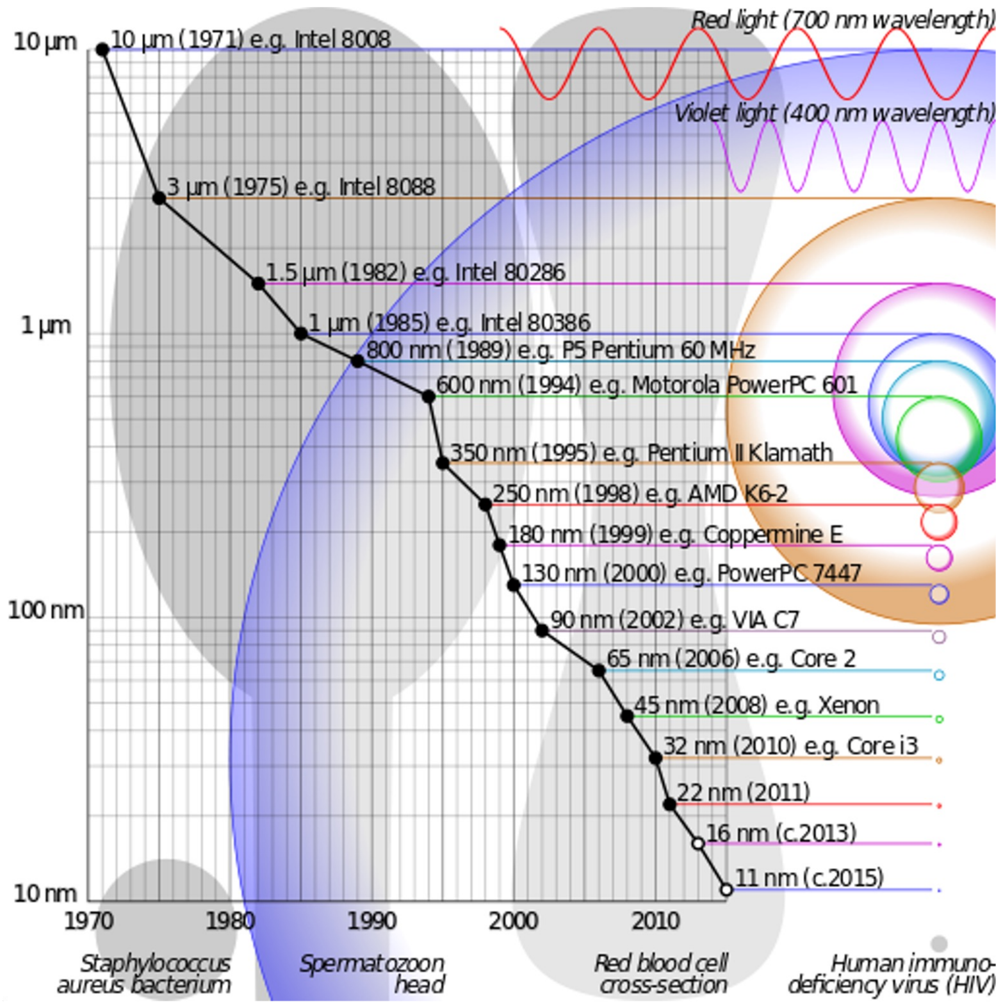


MOSFET scaling (process nodes)

10 μm – 1971
6 μm – 1974
3 μm – 1977
1.5 μm – 1981
1 μm – 1984
800 nm – 1987
600 nm – 1990
350 nm – 1993
250 nm – 1996
180 nm – 1999
130 nm – 2001
90 nm – 2003
65 nm – 2005
45 nm – 2007
32 nm – 2009
22 nm – 2012
14 nm – 2014
10 nm – 2016
7 nm – 2018
5 nm – 2020
3 nm – 2022

Future
2 nm ~ 2024

Transistor size



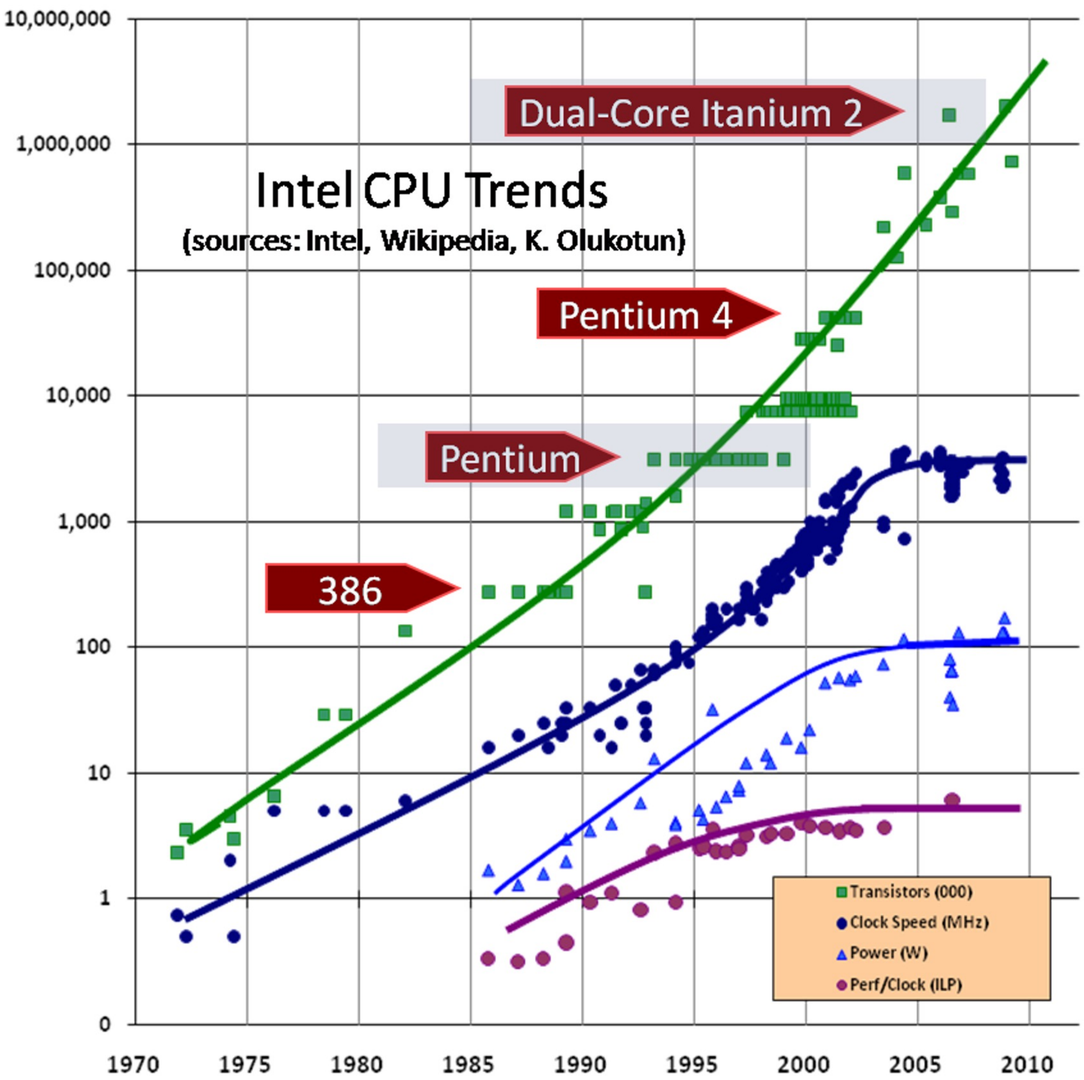
Nano meter (nm) = 1 x 10^{-9} meter

Helium atom = 0.1 nm
Hair width = 60 000 nm
A fingernail grows by 1 nm per second

Machines used in this class:

brake.ii.uib.no: 32 nm technology.
lyng.ii.uib.no: 22 nm
birget.uib.no: 7nm

Clock Frequency and Power



$P = CU^2f$

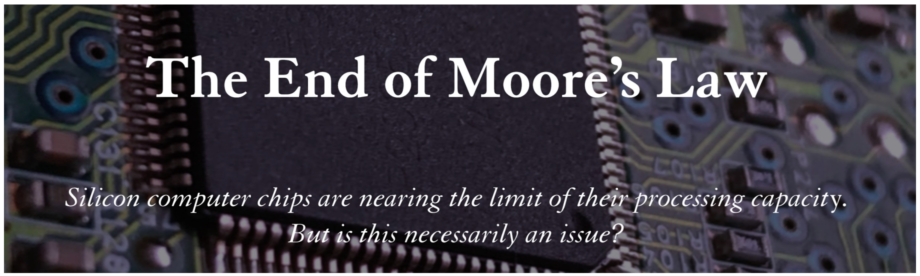
P:	Power
U:	Voltage
f:	frequency

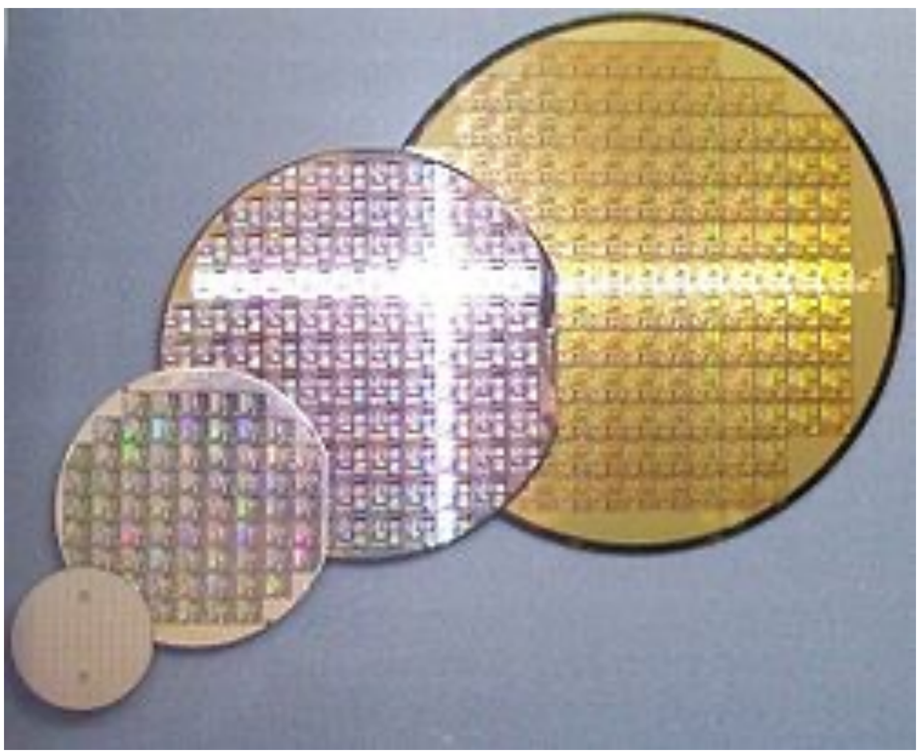
Frequency grows with the voltage. Thus $P \approx \Theta(f^3)$

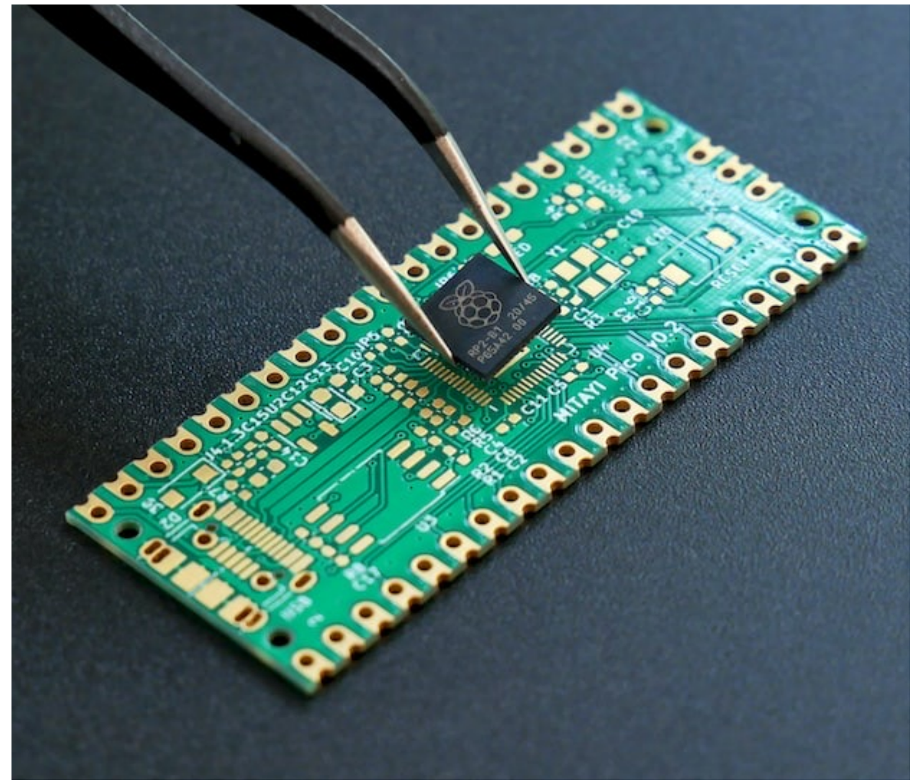
Clock frequency has not increased (much) since 2000!

CPUs were getting too hot and using too much power.

The Future







ANNALS OF TECHNOLOGY

THE WORLD'S LARGEST COMPUTER CHIP

In the race to accelerate A.I., the Silicon Valley company Cerebras has landed on an unusual strategy: go big.

By Matthew Hutson

August 20, 2021

Parallel Computing

What to do?

What we have seen so far:

- Processor speed is not increasing (much)
- Transistor size is going down (at least for now)
- More space to fill with transistors: On the same area, count is going up as n^2

What to do with the extra transistors?

- More memory
- More caches

But we still need more processing power!

Obvious solution:

Put multiple processors in the same computer!

Parallel Computing is already here!

Samsung Galaxy S21:

2.2 GHz quad core processor (5nm) + Mali- G78 GPU

iMac:

4.5GHz 4-18-cores + GPU

Nvidia GeForce GTX 1660

1536 cores (12 nm)

Playstation 5:

8 cores CPU (7 nm) + GPU

<u>Today</u>

- PC's and phones: Mainly used for running multiple programs
- Graphic cards (GPU): Used to accelerate video and games (and ML)

<u>Future</u>

- Clock rates are not increasing => Larger and more complex programs *must* use multiple
- processors
- Power goes down rapidly with decreasing frequency, remember $P = \Theta(f^3)$

  → Mobile devices can save battery by reducing clock rate while using more processors.

Parallel Computing

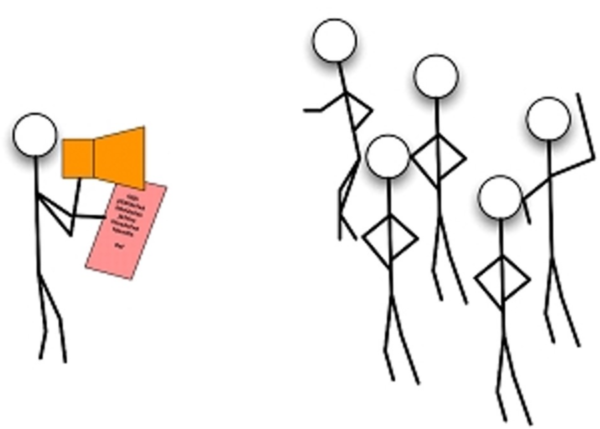
Solve traditional computing problems using multiple interconnected processors

<u>Why we do it:</u>

- Get solution faster
- Gives access to more memory, can solve larger problems

<u>Issues:</u>

- Harder to develop algorithms → "must find meaningful work for everyone"
- Harder to program → "do more than one thing at a time"
- Harder to debug → "execution is not deterministic"



Other ways to exploit parallelism

- Bit level parallelism
- within floating point operations, etc.

- Instruction level parallelism (ILP)
- multiple instructions execute per clock cycle

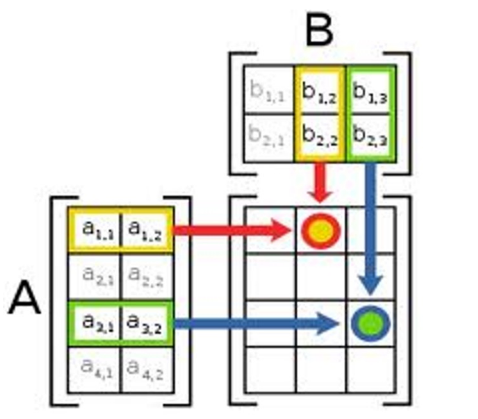
- Memory system parallelism
- overlap of memory operations with computation

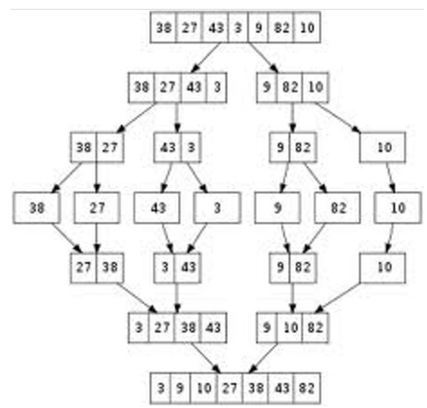
- OS parallelism
- multiple jobs run in parallel

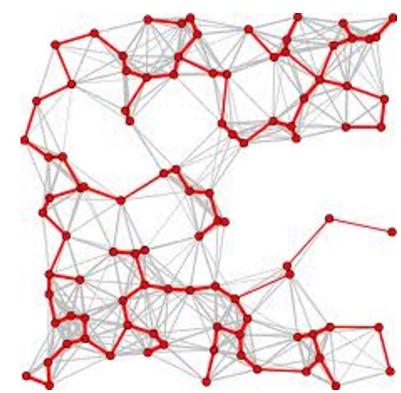
Parallel Algorithms

We will develop and analyze parallel algorithms,

i.e., solve traditional algorithmic problems using parallel computers.
Main motivation comes from compute intensive applications:

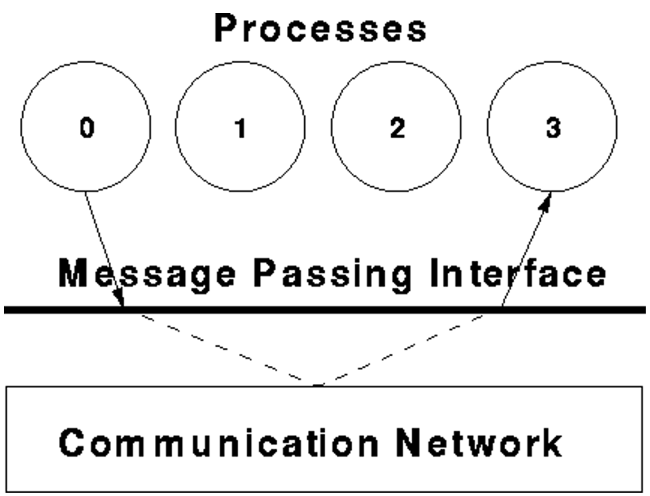


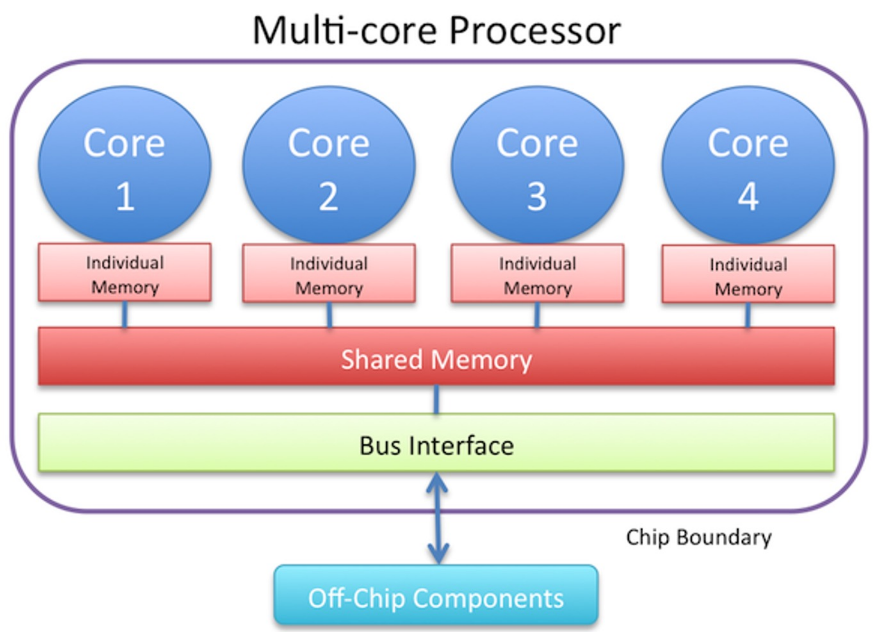




We will not consider
- Distributed computing
- Executing separate jobs on multiple processors (OS parallelism)

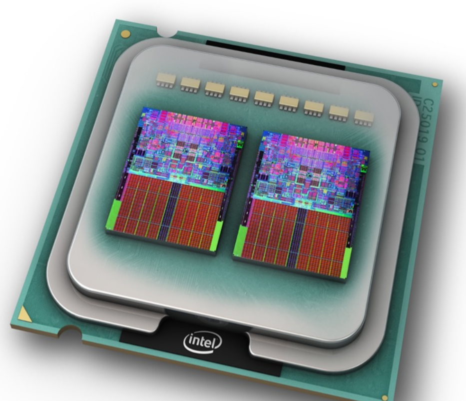
Types of Parallel Computers

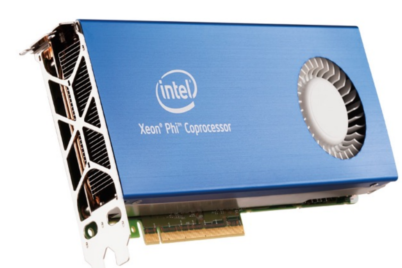




Distributed memory computers

Shared memory computers





Top 500

Ranking of the 500 fastest computers in the world

Published twice a year

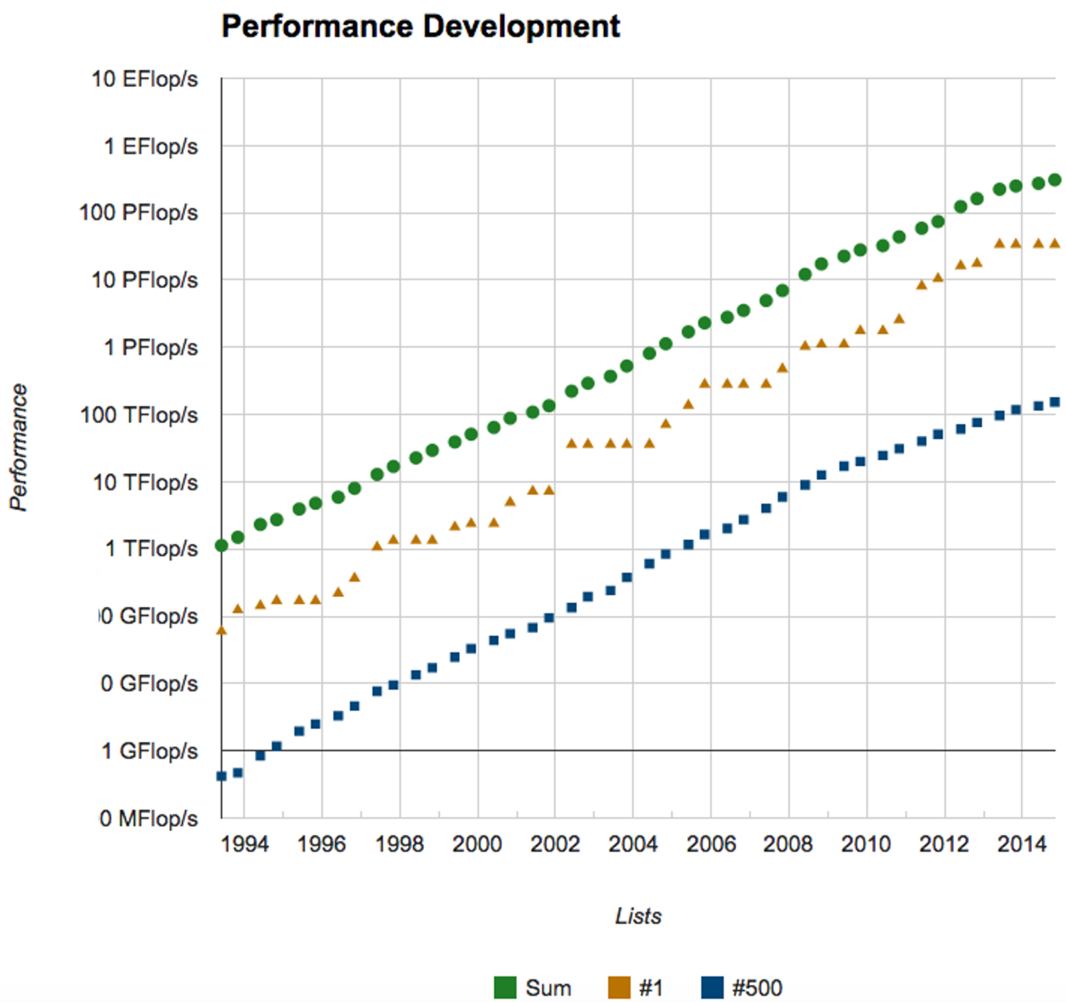
Based on solving Ax = b using Gaussian elimination with partial pivoting

Speed is measured in flops: Floating point operations per second

Current leader (2023):

Frontier at Oak Ridge National Laboratories, USA
8.7M Cores, 1102 petaflops (> 1 exaflop)

2022:
Fugaku at RIKEN Center for Computational Science, Japan
7.6M Cores, 442 petaflops (peta = 10^{15} 0.44 exaflops)

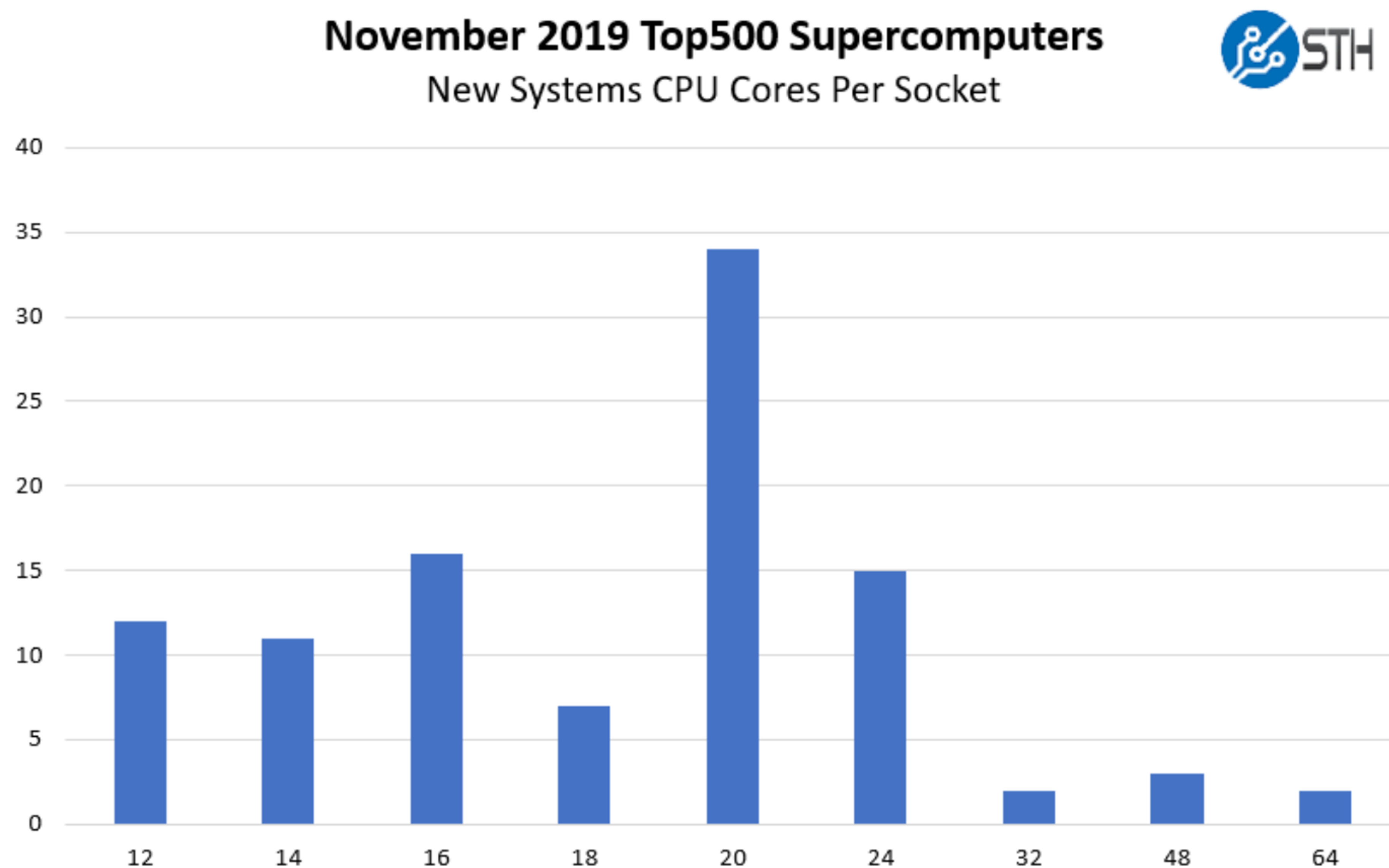


Performance Development
10 EFlop/s
1 EFlop/s
100 PFlop/s
10 PFlop/s
1 PFlop/s
100 TFlop/s
10 TFlop/s
1 TFlop/s
0 GFlop/s
0 GFlop/s
1 GFlop/s
0 MFlop/s
Performance
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
Lists
Sum
#1
#500

Iphone 6

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
6	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox	1,572,480	94.64	125.71	7,438

November 2019 Top500 New Systems By CPU Cores Per Socket



November 2019 Top500 New Systems By CPU Cores Per Socket

Hexagon

- Most powerful computer in Norway 2008 - 2012
- Highest ranked Norwegian computer ever (48, June 2008)

	Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
Nov 08	66	University of Bergen Norway	Cray XT4 QuadCore 2.3 GHz Cray Inc.	5,550	40.6	51.1	274
	82	University of Tromso Norway	**Stallo** - Cluster Platform 3000 BL460c, Xeon 53xx 2.66GHz, GigEthernet Hewlett-Packard	5,632	31.9	59.9	
Nov 09	121	University of Bergen Norway	Cray XT4 QuadCore 2.3 GHz Cray Inc.	5,550	40.6	51.1	274
	193	University of Tromso Norway	**Stallo** - Cluster Platform 3000 BL460c, Xeon 53xx 2.66GHz, GigEthernet Hewlett-Packard	5,632	31.9	59.9	
Nov 10	255	University of Bergen Norway	Cray XT4 QuadCore 2.3 GHz Cray Inc.	5,550	40.6	51.1	274
	363	Statoil Norway	Cluster Platform 3000 BL460c G6, Xeon X5570 2.93 GHz, GigE Hewlett-Packard	5,664	35.3	66.4	
	458	University of Tromso Norway	**Stallo** - Cluster Platform 3000 BL460c, Xeon 53xx 2.66GHz, GigEthernet Hewlett-Packard	5,632	31.9	59.9	
Nov 12	56	Norwegian University of Science and Technology Norway	SGI ICE X, Xeon E5-2670 8C 2.600GHz, Infiniband FDR SGI	22,048	396.7	458.6	537
	134	University of Oslo Norway	**Abel** - MEGWARE MiriQuid, Xeon E5-2670 8C 2.600GHz, Infiniband FDR Megware	10,080	178.6	209.7	227
	157	University of Bergen Norway	**Hexagon** - Cray XE6m-200, Opteron 6276 16C 2.300GHz, Cray Gemini interconnect Cray Inc.	22,272	160.1	204.9	342.3

Nov 13

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
82	Norwegian University of Science and Technology Norway	SGI ICE X, Xeon E5-2670 8C 2.600GHz, Infiniband FDR SGI	22,048	396.7	458.6	537
240	University of Oslo Norway	**Abel** - MEGWARE MiriQuid, Xeon E5-2670 8C 2.600GHz, Infiniband FDR Megware	10,080	178.6	209.7	227
281	University of Bergen Norway	**Hexagon** - Cray XE6m-200, Opteron 6276 16C 2.300GHz, Cray Gemini interconnect Cray Inc.	22,272	160.1	204.9	342.3

Jun 14

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
98	Norwegian University of Science and Technology Norway	SGI ICE X, Xeon E5-2670 8C 2.600GHz, Infiniband FDR SGI	22,048	396.7	458.6	537
300	University of Oslo Norway	**Abel** - MEGWARE MiriQuid, Xeon E5-2670 8C 2.600GHz, Infiniband FDR MEGWARE	10,080	178.6	209.7	227
382	University of Bergen Norway	**Hexagon** - Cray XE6m-200, Opteron 6276 16C 2.300GHz, Cray Gemini interconnect Cray Inc.	22,272	160.1	204.9	342.3

Nov 14

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
127	Norwegian University of Science and Technology Norway	SGI ICE X, Xeon E5-2670 8C 2.600GHz, Infiniband FDR SGI	22,048	396.7	458.6	537
368	University of Oslo Norway	**Abel** - MEGWARE MiriQuid, Xeon E5-2670 8C 2.600GHz, Infiniband FDR MEGWARE	10,080	178.6	209.7	227
462	University of Bergen Norway	**Hexagon** - Cray XE6m-200, Opteron 6276 16C 2.300GHz, Cray Gemini interconnect Cray Inc.	22,272	160.1	204.9	342.3

Developing Parallel Algorithms

Challenges when writing parallel programs

- Finding enough parallelism (Amdahl's Law)

- Granularity

- Locality

- Load balance

- Coordination and synchronization

⇨ All these challenges make parallel programming even harder than sequential programming.

Until next time...

- Check out www.top500.org
- Check out www.green500.org
- Check out www.graph500.org
- Intel’s tick-tock cycle of development
- Read about Gordon Moore and John Von Neumann
- Read about Cerebras