University of Zurich[UZH]

High Performance Computing
Lecture 5

Douglas Potter
Jozef Bucko
Noah Kubli

Parallel Programming: an introduction

History

- Moore's law

Parallel processing

- Amdahl's and Gustafson's laws

Hardware evolution

- Vector, parallel and accelerated processors

Programming languages

- MPI, OpenMP, CUDA, OpenACC

Moore's law and hardware constraints

Gordon Moore: founder of Intel.

Statement of the law:

The number of transistors that can be placed on an integrated circuit at a reasonable cost doubles every two years.

Energy consumption and cost:

- Dissipated electric power scales at clock frequency to the cube.
- Dissipated power by square cm is limited by cooling

Constraints:

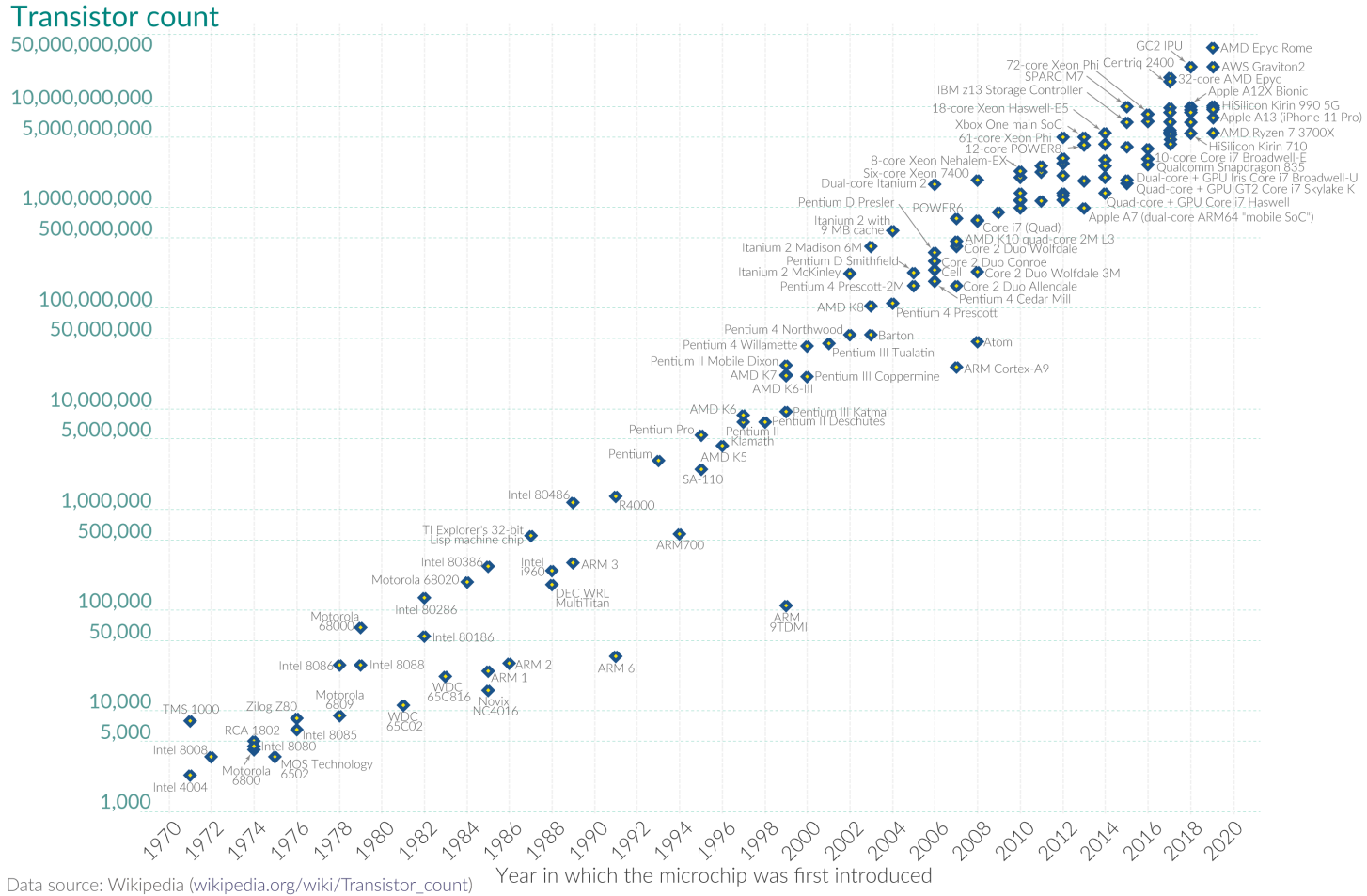
- Processor frequency reached a plateau around 3GHz since 2002-2004 but Moore's law still true
- More cores per chip (many cores architectures, GPU…)

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

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.

Our World in Data

University of Zurich[UZH]

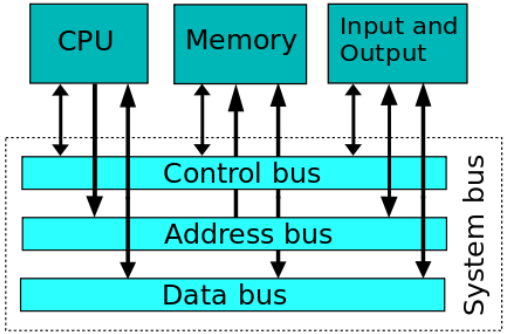


Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world's largest problems.

The Memory Wall

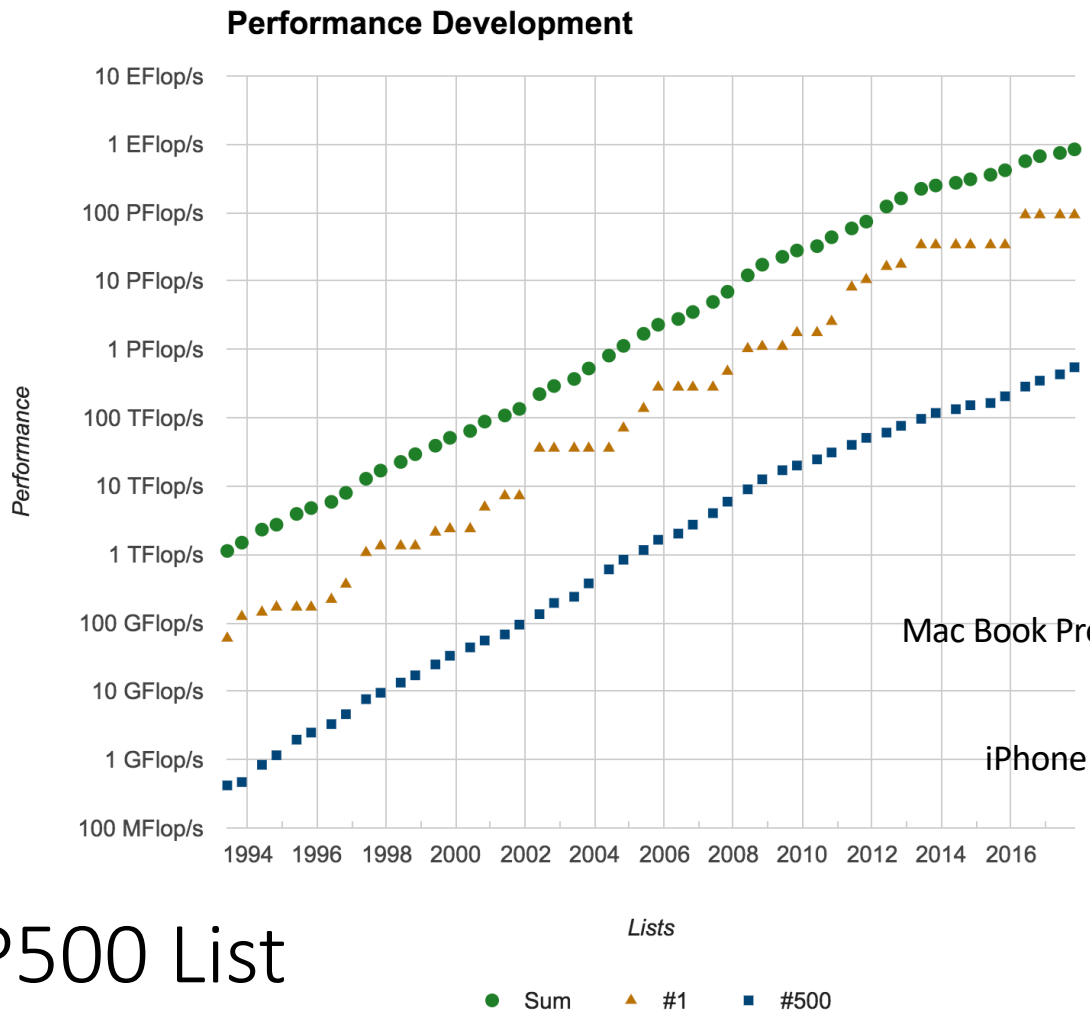
Von Neumann Architecture
Programming instructions (stored-program) and data share the memory unit. They are loaded to the Central Processing Unit (CPU) for execution. Results are loaded back to the memory unit.



The Von Neumann Bottleneck:
- Memory bandwidth is not increasing as quickly as processor computing power
- Memory latency is decreasing very slowly
- Number of computing cores per memory units is increasing

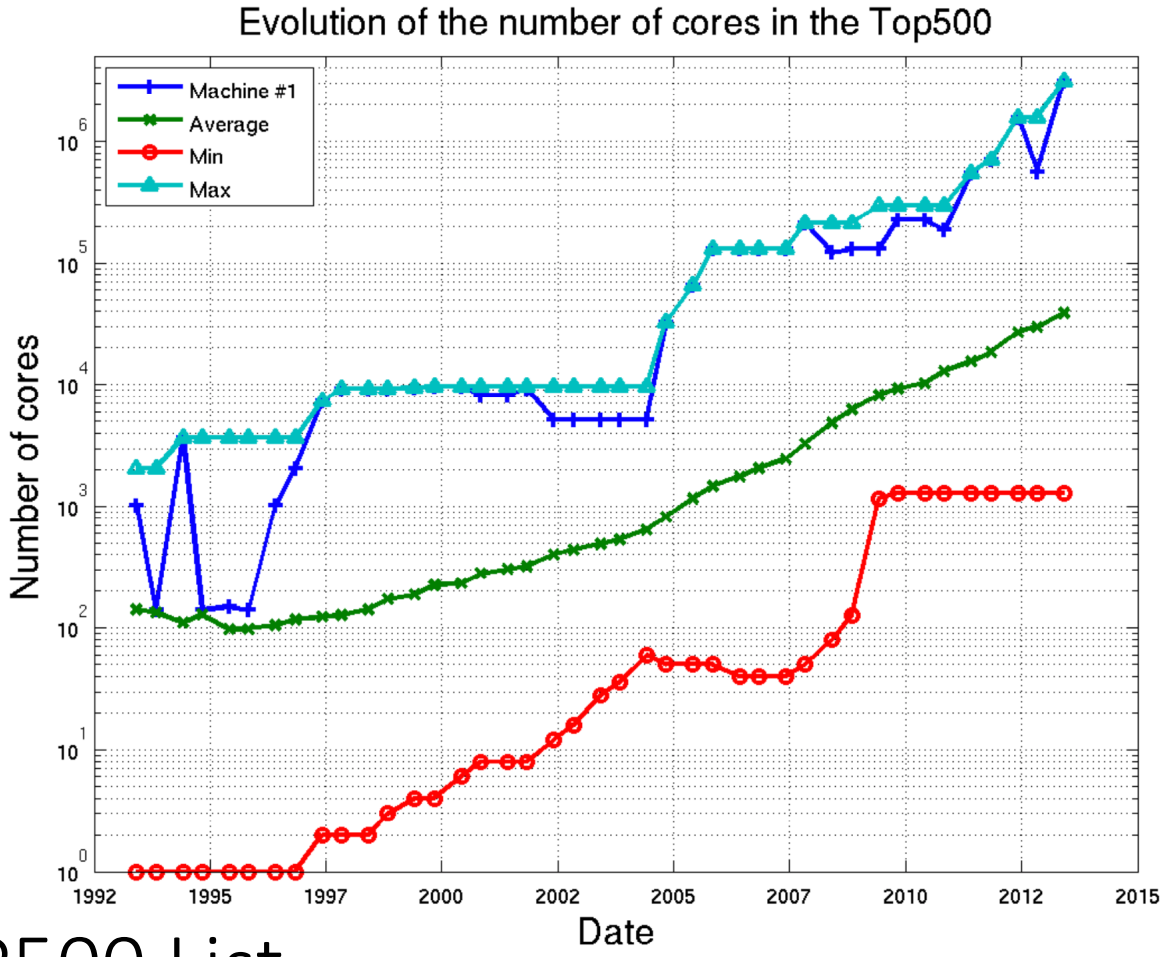
Consequences and solutions:
- The CPU wastes cycles while waiting for data.
- Introduction of cache memory (L1, L2, L3)
- Parallel access to memory (vector architectures, AVX)

TOP500 List

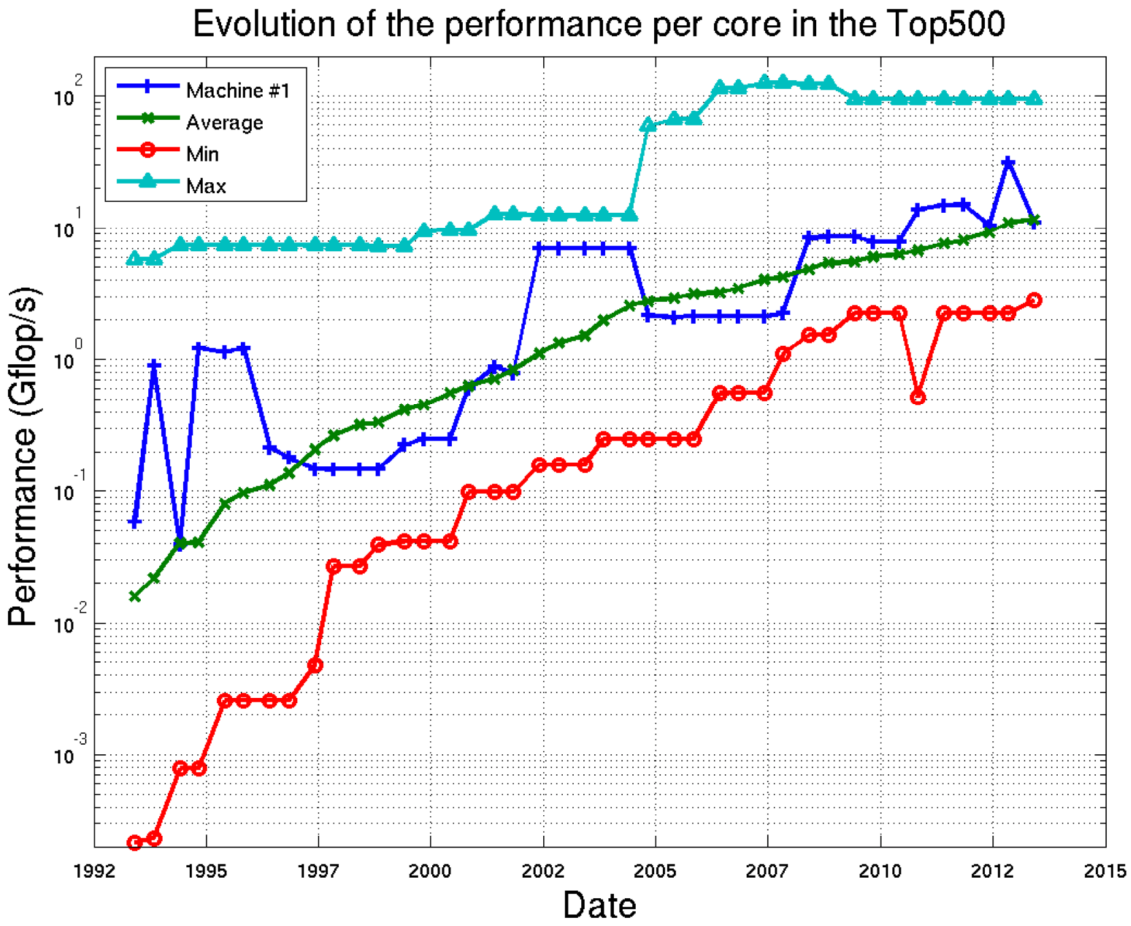


Mac Book Pro: 100 GFlops

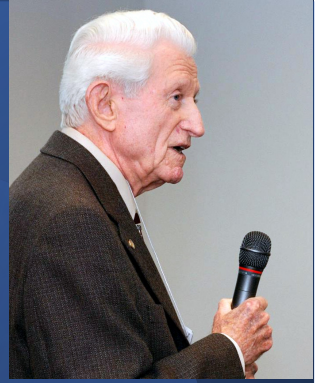
iPhone 6: 1 GFlops


Evolution of the number of cores in the Top500
Machine #1
Average
Min
Max
Number of cores
Date
1992
1995
1997
2000
2002
2005
2007
2010
2012
2015

TOP500 List


Evolution of the performance per core in the Top500
Machine #1
Average
Min
Max
Performance (Gflop/s)
10^2
10^1
10^0
10^-1
10^-2
10^-3
1992
1995
1997
2000
2002
2005
2007
2010
2012
2015
Date

TOP500 List



Parallel Processing

Gene Amdahl (1967): Amdahl's Law

Statement of the law:

The theoretical maximum speedup obtained by parallelizing a code ideally, *for a given problem with a fixed size*:

$$\text{Speedup}(N) = \frac{T_s}{T_p(N)} = \frac{T_s}{\alpha T_s + (1-\alpha)\frac{T_s}{N}} = \frac{1}{\alpha + \frac{1-\alpha}{N}}$$

T_s : execution time of the serial code

T_p : execution time of the parallel code

α : fraction of the code that is not parallel

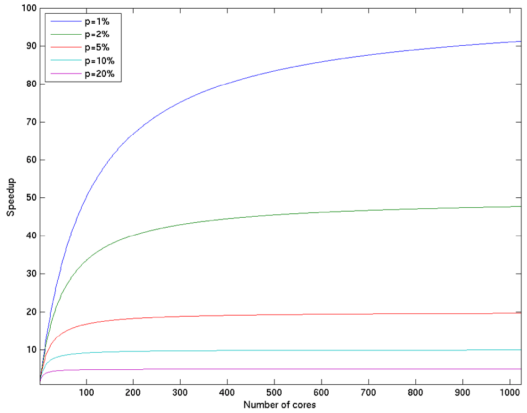
N : number of processors

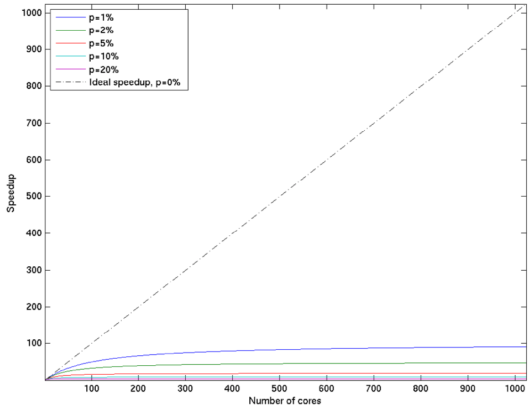
$$\text{Speedup}(N) \to \frac{1}{\alpha} \text{ as } N \to +\infty$$

Strong scaling

Theoretical Maximum Speedup

Cores	α (%)								
	0	0.01	0.1	1	2	5	10	25	50
10	10	9.99	9.91	9.17	8.47	6.90	5.26	3.08	1.82
100	100	99.0	91.0	50.2	33.6	16.8	9.17	3.88	1.98
1000	1000	909	500	91	47.7	19.6	9.91	3.99	1.998
10000	10000	5000	909	99.0	49.8	19.96	9.99	3.99	2
100000	100000	9091	990	99.9	49.9	19.99	10	4	2
∞	∞	10000	1000	100	50	20	10	4	2





Weak scaling

John Gustafson and Edwin Barsis (1988): Gustafson's Law

Statement of the law:

The theoretical maximum speedup obtained by parallelizing a code ideally *for a problem of constant size per core*:

$$\text{Speedup}(N) = \frac{T_S}{T_P(N)} = \alpha + (1-\alpha)N$$

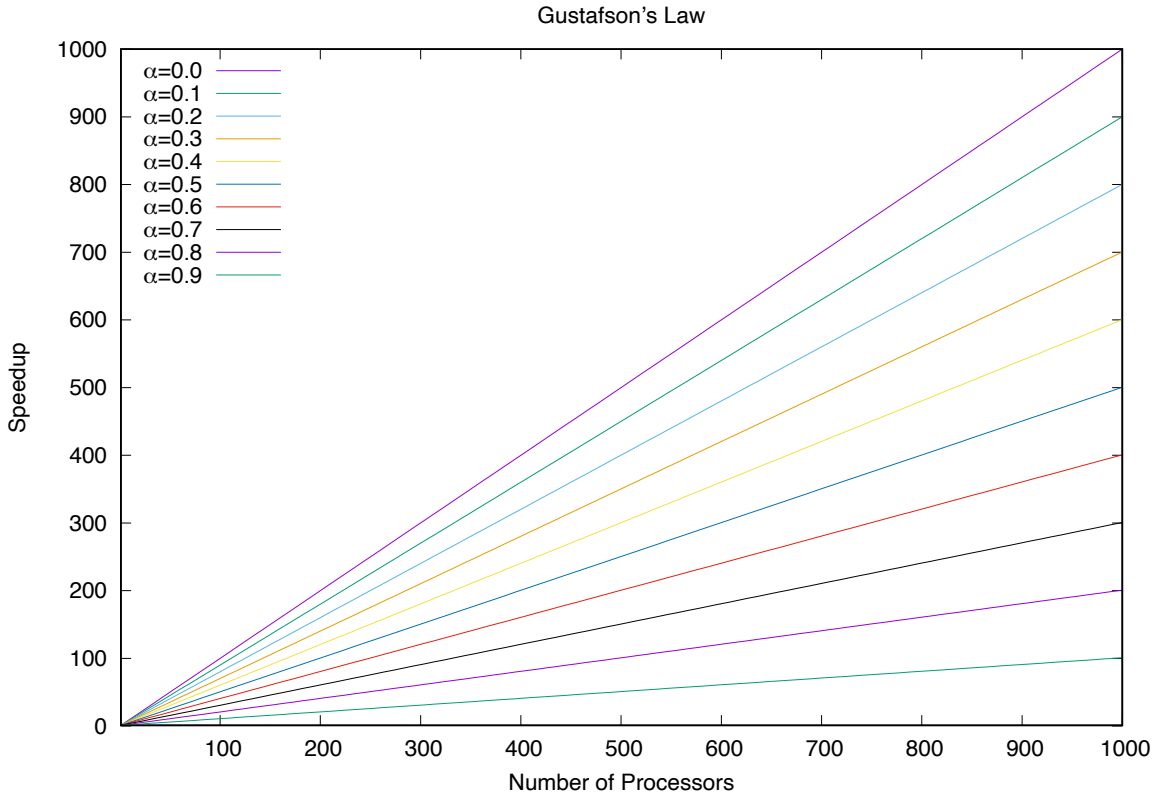
Basic premise: **with an already parallelized task, what is the theoretical "slowdown" of the task if run serially?**

This law is more optimistic than Amdahl's law:

$$\text{Speedup}(N) \to (1-\alpha)N \text{ as } N \to +\infty$$

In both laws, everything is determined by α, the non-parallel fraction of the code. Reducing α is called "parallel code optimization".

Weak scaling



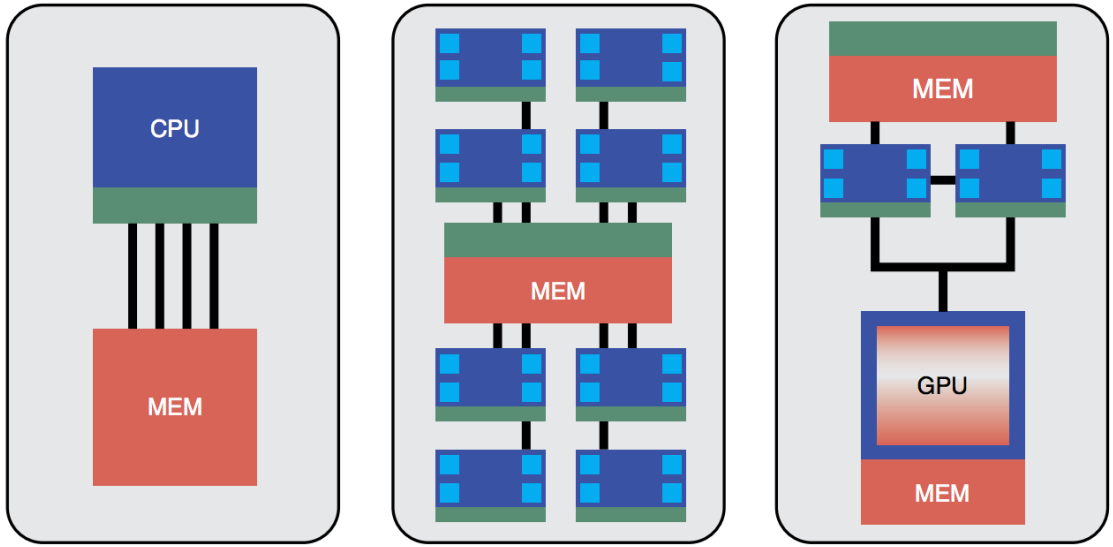
Hardware Evolution

Technical trends:

- Computing power is doubling every year
- Massively parallel and many-cores architectures are dominant
- Since 2010, Graphical Processing Units become a key player
- Increasing hardware complexity (hybrid system, multiple cache layers)
- Memory per core is stagnating or decreasing
- Performance per core is stagnating
- Disk Input/Output bandwidth is increasing very slowly

Cray XC50 with GPU

Evolution of supercomputers

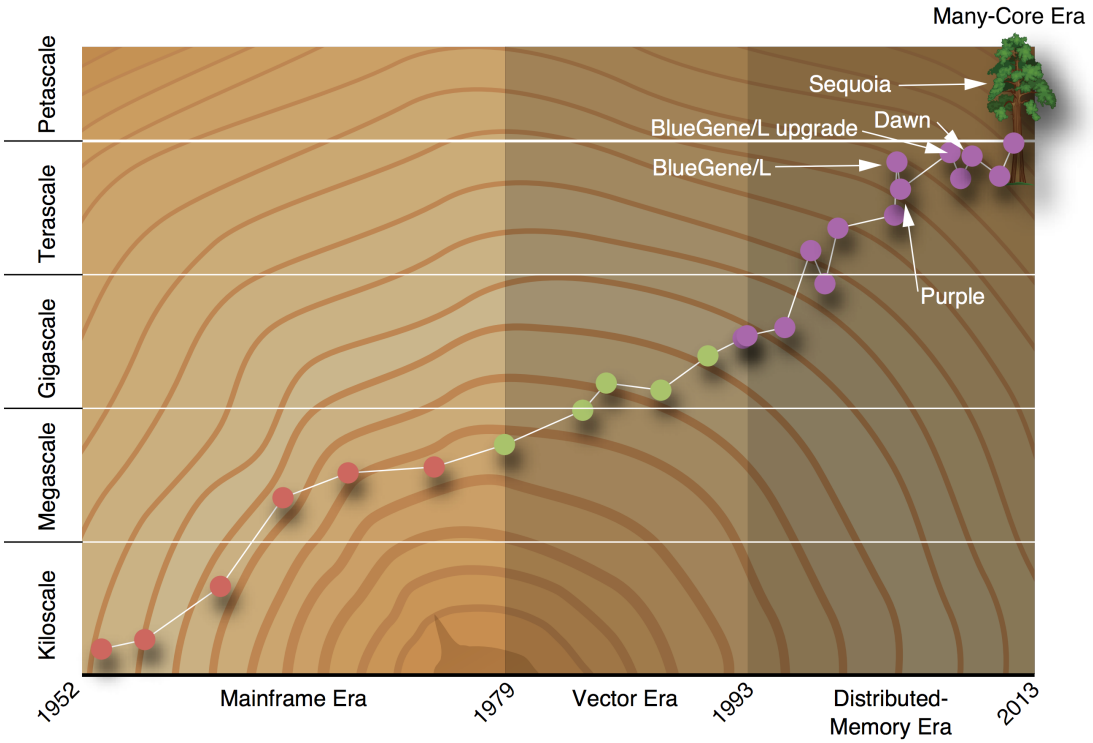


1995
Single CPU per node with main memory

2000–2010
Multiple CPUs per node sharing main memory

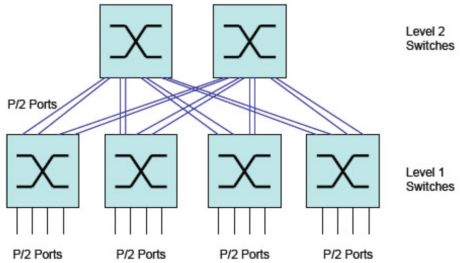
2000–2010
Accelerators usher in era of heterogeneity

Evolution of supercomputers



Communication Network

- Connecting millions of processors requires a high-performance network
- Most existing systems use “Infiniband” or variations.
- Need a network **switch** to interconnect all the nodes
- Need high quality fiber **cables** to connect each node to the switch



- Fat-tree network: non-blocking network topology invented by Charles Clos (1953)
- Depending on how many switches you can afford, you might choose blocking configurations.
- http://www.mellanox.com/clusterconfig/
- http://clusterdesign.org/fat-trees/
- Key network parameters (switches and cables) are the **latency** and the **bandwidth**
- **Infiniband network:** latency 0.5-1 microsec, bandwidth 5-40 GB/sec

Consequences

For Users

- It is necessary to exploit many (relatively slow) cores
- The memory per core is constant or even decreasing (memory limited)
- Higher level of parallelism
- I/O bottlenecks

For Developers

- Raw performance of individual core not increasing anymore
- More complex architecture (cache levels, accelerators, latency…)
- Multi-disciplinary approach is required, leading to the concept of “co-design”

Programming Languages

Evolution of programming methods:

- MPI is still the dominant programming technique
- Hybrid OpenMP/MPI approach most effective on supercomputers
- GPU programming develops quickly (CUDA and OpenACC)
- Message Passing directly within the GPU
- New specific parallel programming languages are developed: Co-array Fortran, PGAS, X10, Chapel…
- New runtime systems to handle task-based parallelism: Charm++, HPX

PizDaint Architecture

Piz Daint Cooling

Piz Daint Architecture

Model	Cray XC40/Cray XC50
Number of Hybrid Compute Nodes	5 320
Number of Multicore Compute Nodes	1 431
Peak Floataing-point Performance per Hybrid Node	4.761 Teraflops Intel Xeon E5-2690 v3/Nvidia Tesla P100
Peak Floating-point Performance per Multicore Node	1.210 Teraflops Intel Xeon E5-2695 v4
Hybrid Peak Performance	25.326 Petaflops
Muliticore Peak Performance	1.731 Petaflops
Hybrid Memory Capacity per Node	64 GB; 16 GB CoWoS HBM2
Multicore Memory Capacity per Node	64 GB, 128 GB
Total System Memory	437.9 TB; 83.1 TB
System Interconnect	Cray Aries routing and communications ASIC, and Dragonfly network topology
Sonexion 3000 Storage Capacity	6.2 PB
Sonexion 3000 Parallel File System Peak Performance	112 GB/s
Sonexion 1600 Storage Capacity	2.5 PB
Sonexion 1600 Parallel File System Peak Performance	138 GB/s