Parallel Algorithms and Distributed Systems

A.A. 2021/2022
6 ECTS credits, 4 theory(32 hours) + 2 lab(24 hours)

Lecturer: William Spataro
Department of Mathematics and Computer Science - UNICAL
Email: spataro@unical.it
Web: www.mat.unical.it/spataro
Phone: 0984.494875-3691-6464

Assistant: Andrea Giordano
ICAR-CNR
Email: giordano@icar.cnr.it

Course Program

- The course is divided into three main parts:
  - Basic Concepts of Parallel Computing
  - Languages / Programming Libraries:
    - MPI library (Message Passing Interface)
    - OpenMP API and Posix threads
    - CUDA ?
  - Parallel Algorithms (Simulation modeling, Numerical Integration, Matrix Calculation, Sorting, etc.)

- **Lessons on GPGPU (General Purpose Programming on Graphic Processor Units) programming using CUDA C are scheduled**

- The part devoted to the laboratory consist mainly in carrying out MPI in OpenMP and in the implementation of parallel algorithms

Course Program

- **Introduction to Parallel Computing**
  - Aims, Concepts and Terminology
  - Flynn's taxonomy

- **Parallel Architectures**
  - Overview of parallel machines
  - Shared Memory machines (Multiprocessors)
  - Distributed Memory machines (Multicomputers)
  - Multi-core architectures
  - Cache Coherence
  - Hybrid Architectures

Course Program

- **Parallel Programming Models**
    - Thread Model
    - Shared Memory Model
    - Distributed Memory Model
    - Parallel Data Models
    - Other Models

- **Design of Parallel Programs**
    - Automatic and manual parallelization
    - Partitioning
    - Communication
    - Synchronization Load Balancing
    - Granularity
    - Parallel I / O

Course Program

•**Overview of OpenMP - The reference language for programming environments in shared memory / data parallel models**

- General Concepts
- Parallel loops
- Private and shared type variables
- Critical Sections
- Functional parallelism

Course Program

MPI - The reference language for programming distributed memory environments

- General Concepts
- Environment Management Routines
- Point-to-point communications: MPI_Recv and MPI_Send
- Non-blocking communications
- Collective Communications
- Derived data-types
- Communicators and Groups
- Virtual topologies

Course Program

**Introduction to GPGPU programming –
The new frontier of Parallel Computing**

- CUDA C - The reference language for programming of graphics cards
- General concepts, limits
- CUDA, CUDA C
- Thread and Memory Hierarchy
- Concepts of threads, blocks and grid kernels
- Optimization strategy: use of shared memory, coalescing

Course Program

Performance Analysis

- Parallel Overhead
- Speedup and Efficiency
- Superlinear speedup
- Effect of Granularity on Performance
- Scalability
- Amdahl's Law
- Iso-efficiency

Parallel Algorithms

- Matrix calculation
- Numerical/Grid modelling
- Numerical integration, Linear Systems, Searching and sorting algorithms
- Performance analysis of parallel algorithms

Course Program

Laboratory

•Parallel software development in pThreads, OpenMP and MPI on parallel computers (?!)

•Performance measurements

Books/ Didactic sources

Introduction to Parallel Computing, 2/E, Ananth Grama et al. Addison-Wesley

Peter Pacheco, Parallel Programming with MPI. Morgan Kaufmann, 1997

Programming Massively Parallel Processors: A Hands-on approach, David B. Kirk, Web-mei W. Hwu, 2° Edition, Morgan Kaufmann

G. Spezzano, D. Talia "Calcolo parallelo, automi cellulari e modelli per sistemi complessi", Franco Angeli, Milano, 1999.

I. Foster. Designing and Building Parallel Programs. Addison-Wesley, 1995, Versione online disponibile presso http://www-unix.mcs.anl.gov/dbpp

Professor's website/TEAMS

Final exam, etc

- MPI (Message Passing Interface)

- Project discussion or Seminar
  - Computational model parallelization: Cellular Automata or similar
  - Sorting algorithms, matrix algorithms, etc
  - Anything you like that I approve!

- Written exam

- Office hours: by appointment

- Laboratory (personal notebooks!)

Prerequisites

- Basic GNU/Linux (preferable)
- Compilation in cc, gcc, c++, g++
- Numerical Analysis (Numerical Integration, matrix calculation, etc)
- Modelling and Simulation Course (Complex Systems, Cellular Automata, Genetic Algorithms, etc)
- English ☺

From sequential computing to parallel computing

Sequential computation

- Resolves a problem by means of an algorithm, whose statements are executed in sequence

- Computational model characterized by a single processor

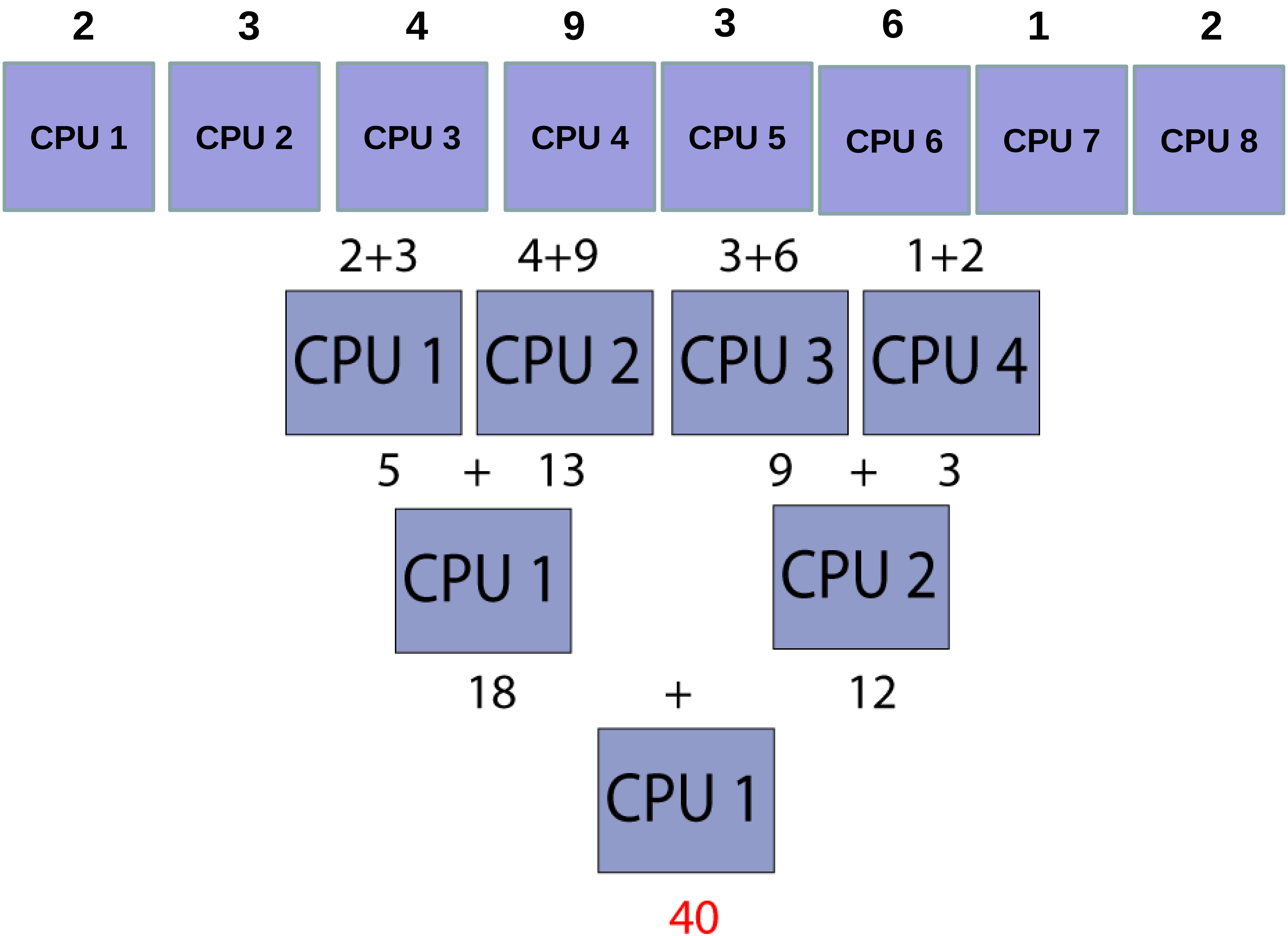
- **Example**: Suppose you want to make the sum of 8 numbers. To perform the sum we needs 7 sums, in addition to an initial allocation (`sum = i[0]`)

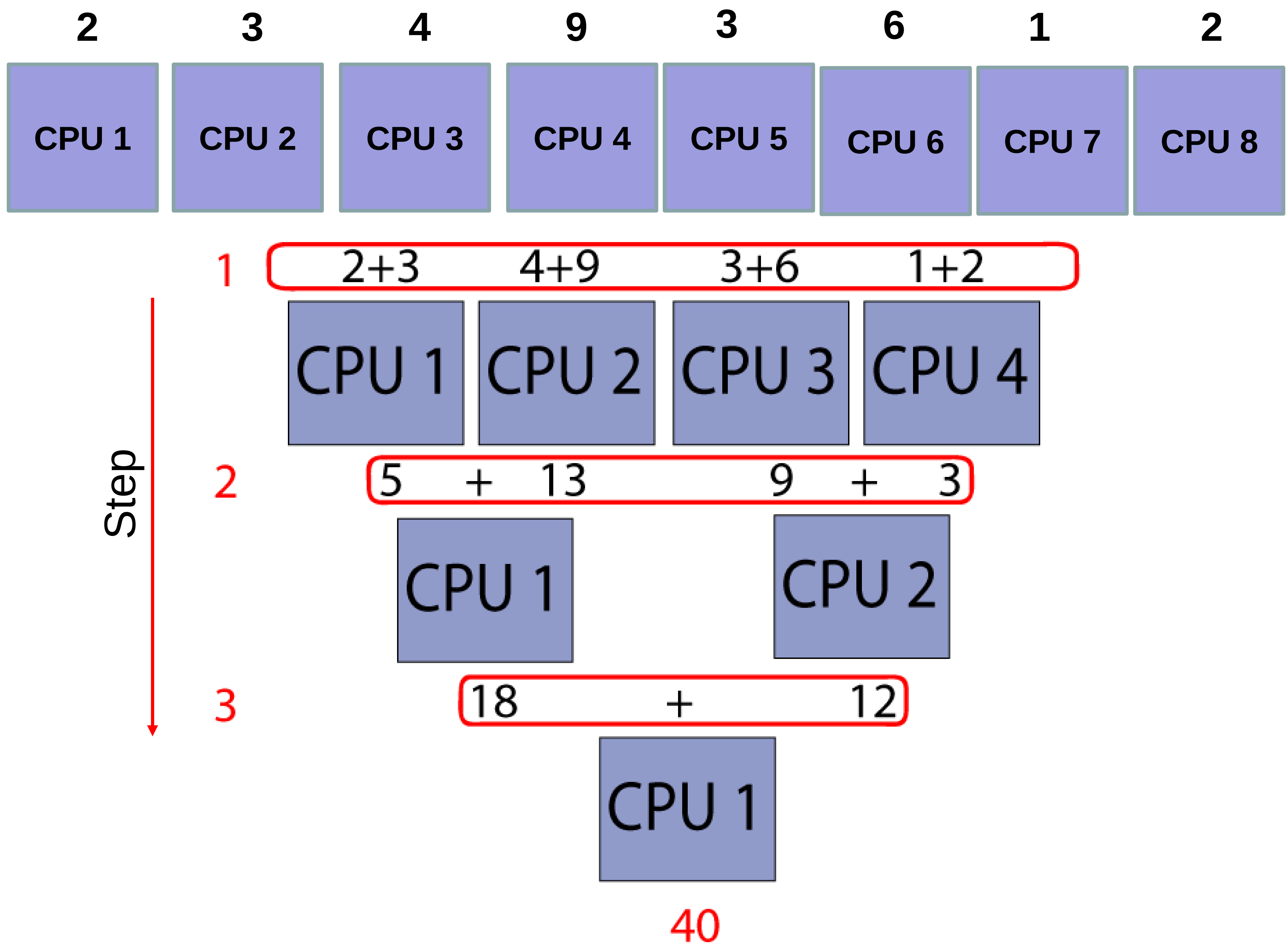
From sequential computing to parallel computing

Parallel Computing

- Solves the same problem by an algorithm whose instructions are executed in parallel mode
- Computational model that provides multiple processors and related cooperation mechanisms
- **Example**: Suppose we perform the sum of eight numbers with 4 CPUs, each of which performs the sum of two variables. After, 2 CPU perform the sum of the two previous results and then CPU 1 for the final sum of the two results obtained.

It takes 3 steps instead of the previous 7 ($\log_2$ n)


2
3
4
9
3
6
1
2
CPU 1
CPU 2
CPU 3
CPU 4
CPU 5
CPU 6
CPU 7
CPU 8
2+3
4+9
3+6
1+2
CPU 1
CPU 2
CPU 3
CPU 4
5
+
13
9
+
3
CPU 1
CPU 2
18
+
12
CPU 1
40



2
3
4
9
3
6
1
2
CPU 1
CPU 2
CPU 3
CPU 4
CPU 5
CPU 6
CPU 7
CPU 8
Step
1
2+3
4+9
3+6
1+2
CPU 1
CPU 2
CPU 3
CPU 4
2
5 + 13
9 + 3
CPU 1
CPU 2
3
18 + 12
CPU 1
40

Famous Last Words

- “I think there is a world market for maybe five computers.”
    - Thomas Watson, chairman of IBM, 1943.
- “There is no reason for any individual to have a computer in their home”
    - Ken Olson, president and founder of Digital Equipment Corporation, 1977.
- “640K [of memory] ought to be enough for anybody.”
    - Bill Gates, chairman of Microsoft,1981.
- “On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it.”
    - Ken Kennedy, CRPC Directory, 1994

Parallel Computing anyone?

Not at all trivial that more processors help to achieve these goals:

- “If a man can dig a hole of 1 m^3 in 1 hour, can 60 men dig the same hole in 1 minute (!) ? Can 3600 men do it in 1 second (!!) ?”

- “I know how to make 4 horses pull a cart, but I do not know how to make 1024 chickens do it” (*Enrico Clementi*)

Important Issues in Parallel Computing

- Task/Program Partitioning.
  - How to split a single task among the processors so that each processor performs the same amount of work, and all processors work collectively to complete the task.

- Data Partitioning.
  - How to split the data evenly among the processors in such a way that processor interaction is minimized.

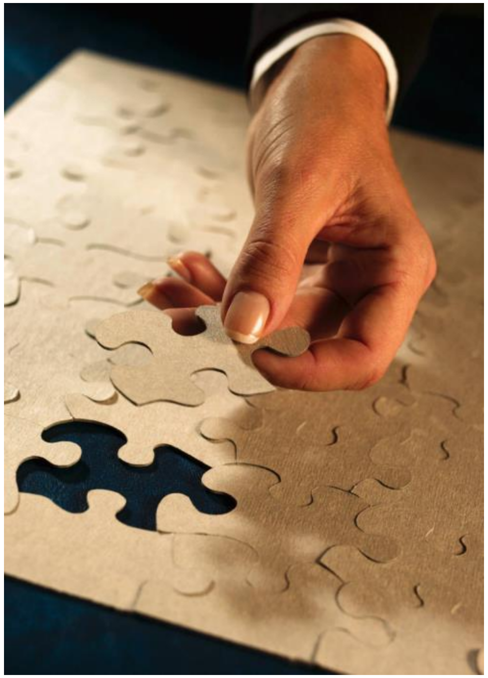
- Communication/Arbitration.
  - How we allow communication among different processors and how we arbitrate communication related conflicts.

Parallel computing concepts

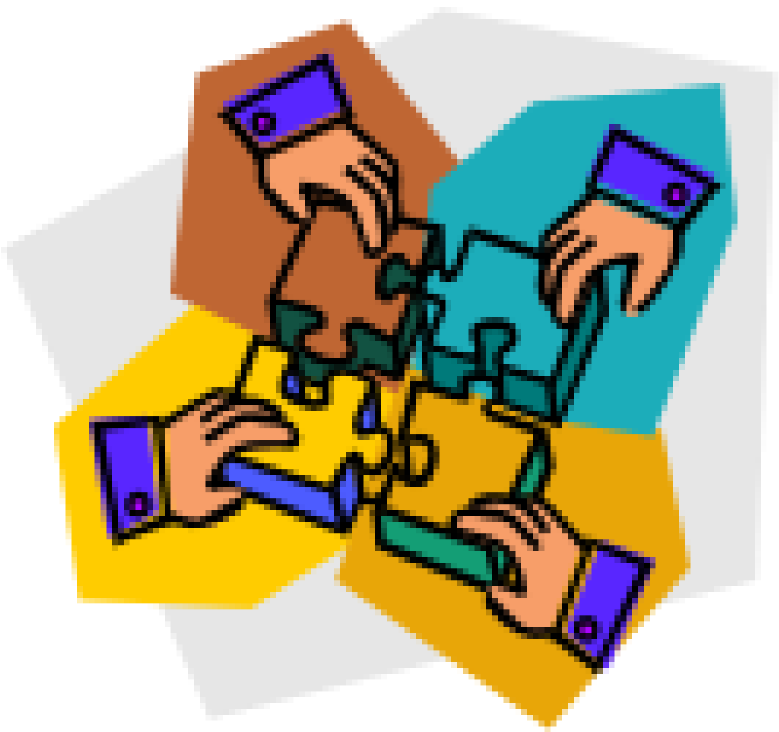
- When performing task, some subtasks depend on one another, while others do not
- Example: Preparing dinner
  - Salad prep independent of lasagna baking
  - Lasagna must be assembled before baking
- Likewise, in solving scientific problems, some tasks independent of one another

Parallel vs Serial : a simple example

- Suppose we want to do 5000 piece jigsaw puzzle
- Time for one person to complete puzzle: n hours
- How can we decrease walltime to completion?



Parallel vs Serial : a simple example



- Add another person at the table
  - Effect on wall time
  - Communication
  - Resource contention
- Add p people at the table
  - Effect on wall time
  - Communication
  - Resource contention

Parallel Computing in a nutshell

- **Compute resources**:
  - A single computer with multiple processors;
  - A number of computers connected by a network;
  - A combination of both.

- **Problems are solved in parallel only when** :
  - Broken apart into discrete pieces of work that can be solved simultaneously;
  - Execute multiple program instructions at any moment in time;
  - Solved in less time with multiple compute resources than with a single compute resource

Parallel Programming

A sequential program can be divided into two types of sections:
- **Inherently sequential sections** (that can not be executed in parallel)
- **Potentially parallelizable sections**

• Methods for the subdivision of parallelizable sections are **<u>critical</u>** for achieving high performance

• *It's fundamental that the sections which are <u>inherently sequential are small</u>*

• The purpose of a **good parallelization** is to:
- Keep all processors (equally) occupied (**load balancing**)
- Minimize the inter-processor communication (**coarse grain parallelism**)
- To limit the **replication** of the calculation to the minimum necessary

Low-cost parallel computers ?

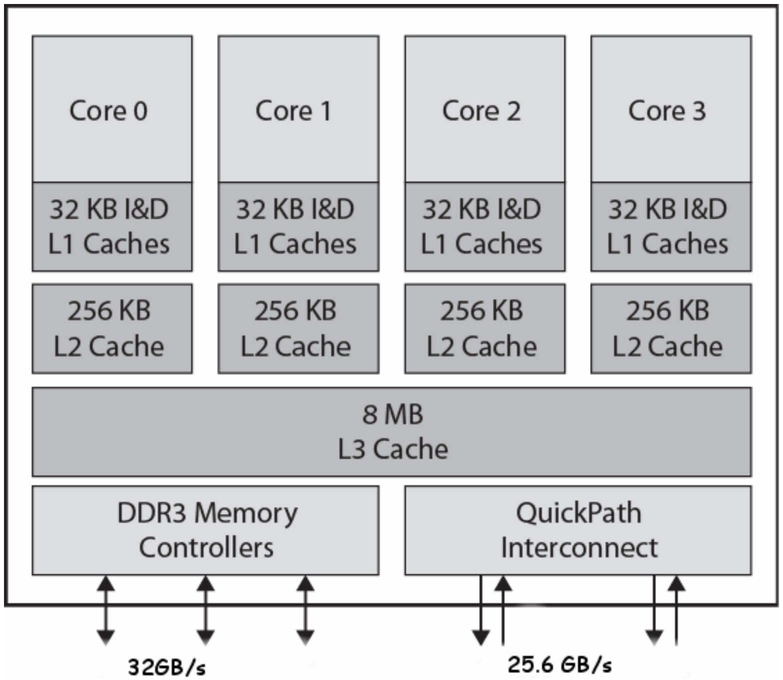
RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4

Flop/s: floating point operations per second

My notebook - 4 cores
250 GFlops

CPU – multi-core: why?

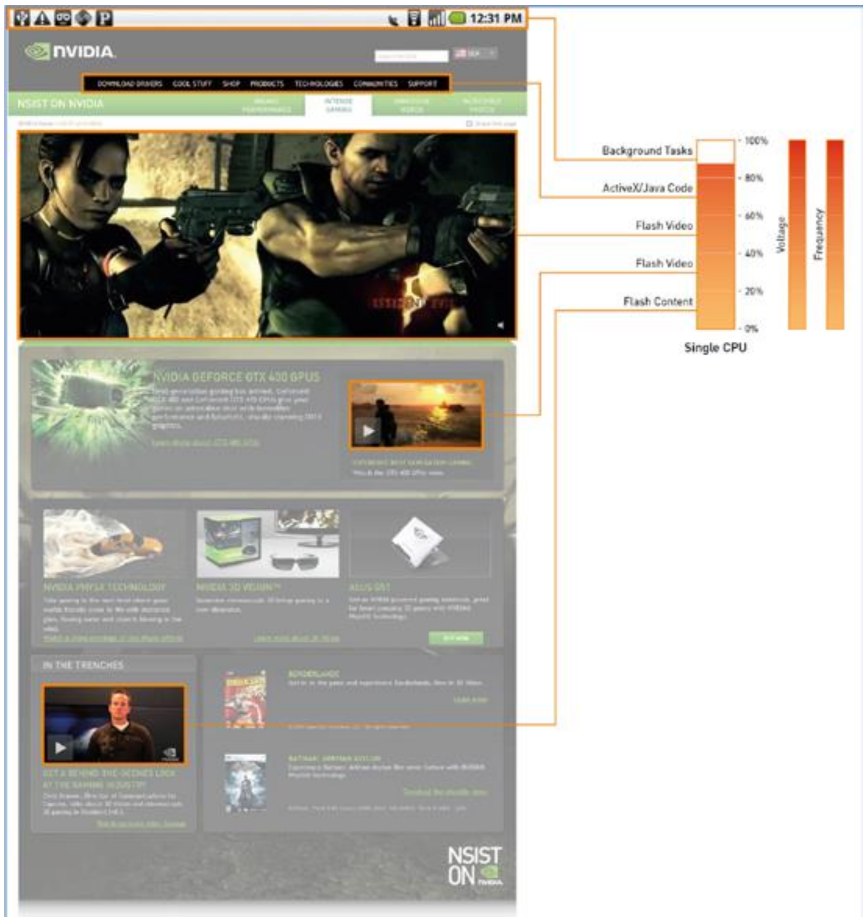
Intel Core i7 Block Diagram



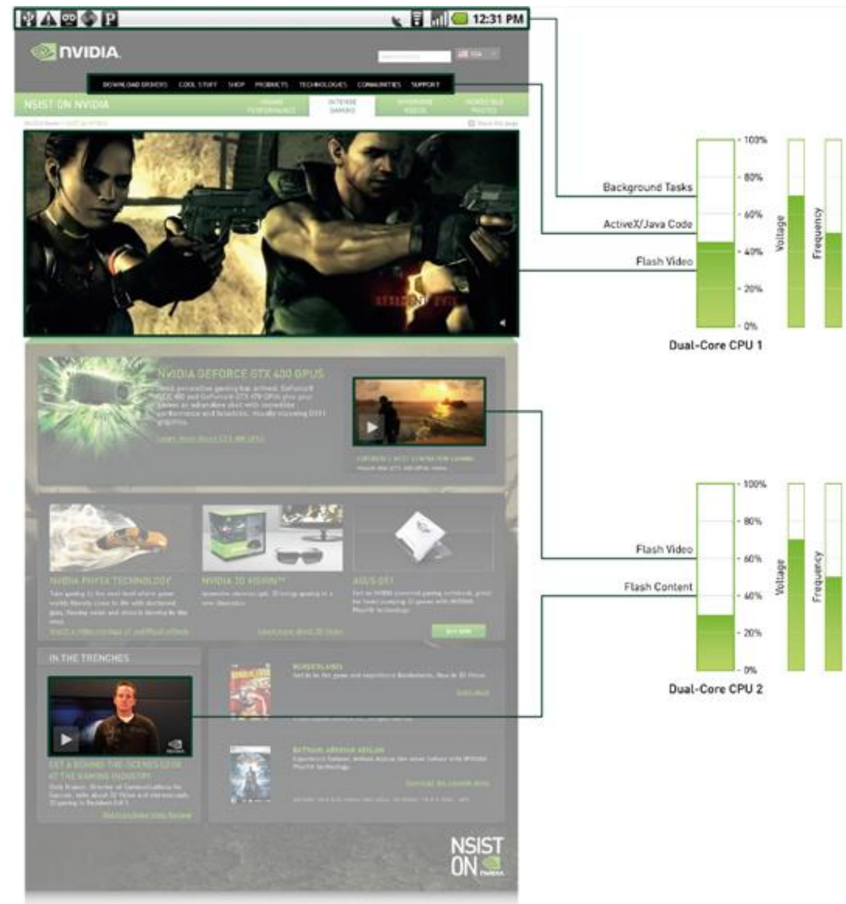
- Increasing performance only by increasing clock performance leads to bottlenecks
- Energy consumption also increased
- It becomes difficult to cool the machines especially for laptops, etc

CPU – multi-core: why?

1 Core



2 Core



Current commercial PCs have 24 cores

Why use parallel computing

- Use of non-local resources:
  - SETI@home : about 1,000,000 computers delivering 1 Exaflop!
  - folding@home : around 340,000 delivering 4.2PetaFlops

- Limits to serial computing:
  - Transmission speeds
  - Limits to miniaturization
  - Economic limitations
  - Current computer architectures are increasingly relying upon hardware level parallelism to improve performance:
    - Multiple execution units
    - Pipelined instructions
    - Multi-core

Units of Measure in HPC

- High Performance Computing (HPC) units are:
  - Flops: floating point operations
  - Flops/s: floating point operations per second
- Typical sizes are millions, billions, trillions…

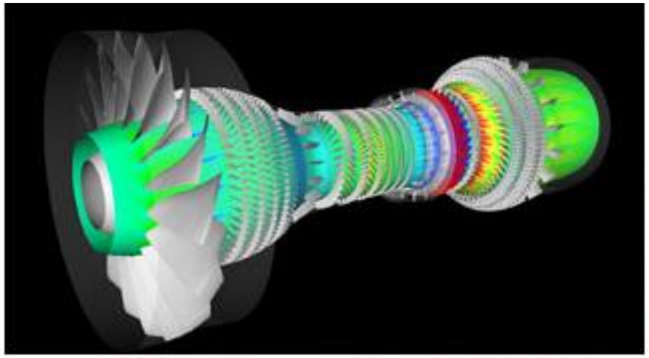
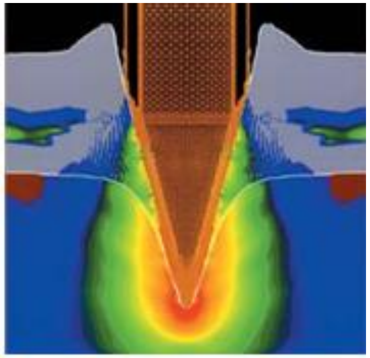
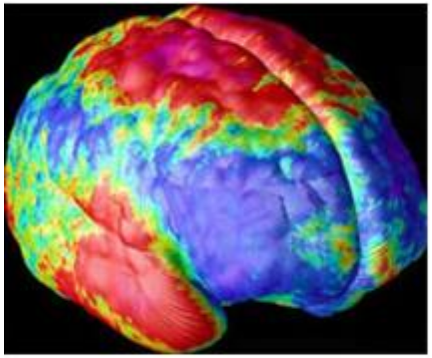
Mega	Mflop/s = 10^6 flop/s	Mbyte = 2^{20} = 1048576 ~ 10^6 bytes
Giga	Gflop/s = 10^9 flop/s	Gbyte = 2^{30} ~ 10^9 bytes
Tera	Tflop/s = 10^{12} flop/s	Tbyte = 2^{40} ~ 10^{12} bytes
Peta	Pflop/s = 10^{15} flop/s	Pbyte = 2^{50} ~ 10^{15} bytes
Exa	Eflop/s = 10^{18} flop/s	Ebyte = 2^{60} ~ 10^{18} bytes
Zetta	Zflop/s = 10^{21} flop/s	Zbyte = 2^{70} ~ 10^{21} bytes
Yotta	Yflop/s = 10^{24} flop/s	Ybyte = 2^{80} ~ 10^{24} bytes

Current fastest (public) machine ~ 500 Pflop/s peak (?!)
Up-to-date list at www.top500.org

Why we need **powerful** computers ?

Uses for Parallel computing

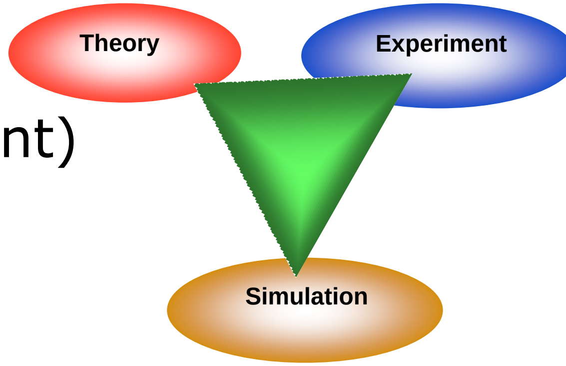
- Historically, parallel computing has been considered to be "the high end of computing", and has been used to model difficult scientific and engineering problems found in the real world.

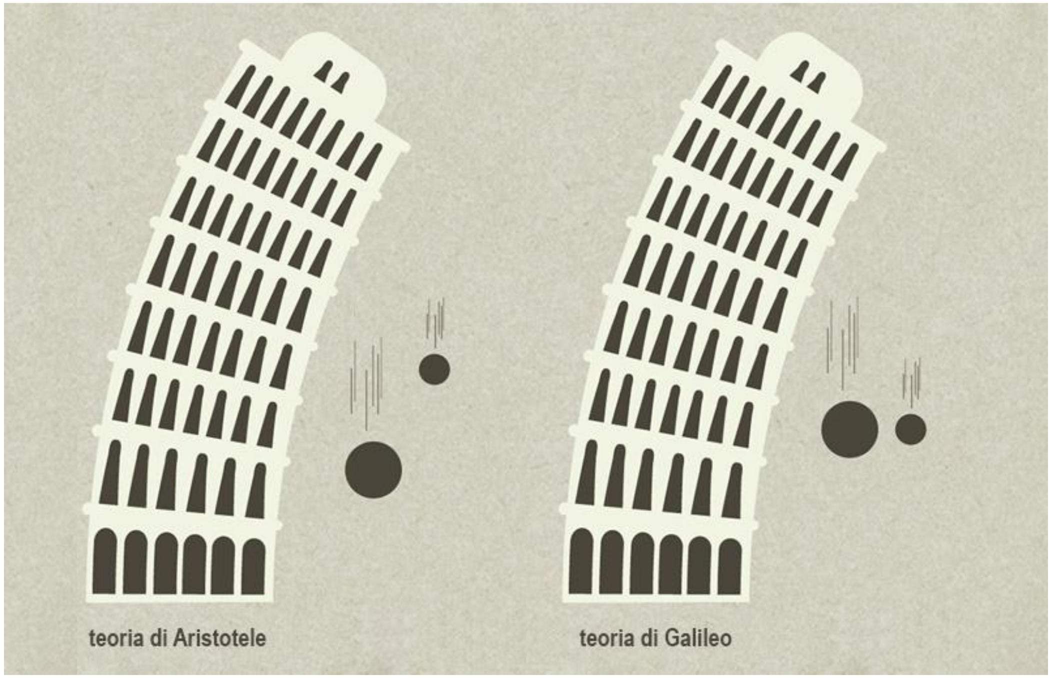




Simulation:
The third pillar of Science

The two traditional paradigms of Science and Engineering are:

1) Make Theory and Design "on paper" (theory)
2) Run or Build a System Experiments (experiment)



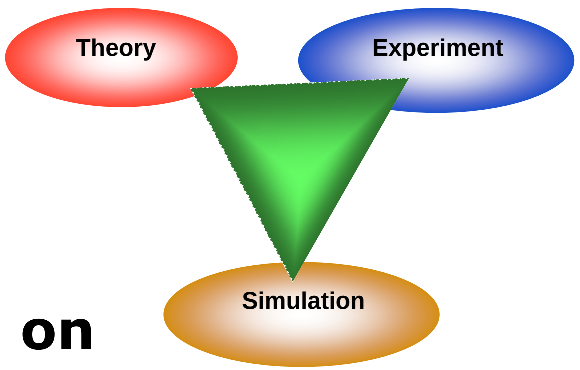


«le velocità de' mobili dell'istessa materia, disegualmente gravi, movendosi per un istesso mezzo, non conservano altrimenti la proporzione delle gravità loro, assegnatagli da Aristotele, anzi che si muovon tutti con pari velocità» - Cit. Galileo 1590 AD

Simulation:
The third pillar of Science

Computational Science Paradigm:

Limitations:
Too Hard (ex: Building of Wind Galleries)
Too Expensive (ex: Build a throw-away airplane)
Too slow (ex: wait for the climate evolution, etc.)
Too Dangerous (ex: nuclear weapons, design of new drugs, etc.)



Computational Science Paradigm:

3) **Use high-performance computers systems to simulate phenomena, based on physical laws and efficient numerical methods (simulation)**

Some Particularly Challenging Computations

- **Science**
  - Global climate modeling, weather forecasts
  - Astrophysical modeling
  - Biology: Genome analysis; protein folding (drug design)
  - Medicine: cardiac modeling, physiology, neurosciences
- **Engineering**
  - Airplane design
  - Crash simulation
  - Semiconductor design
  - Earthquake and structural modeling
- **Business**
  - Financial and economic modeling
  - Transaction processing, web services and search engines
- **Defense**
  - Nuclear weapons (ASCI), cryptography, …

Global Climate Modeling Problem

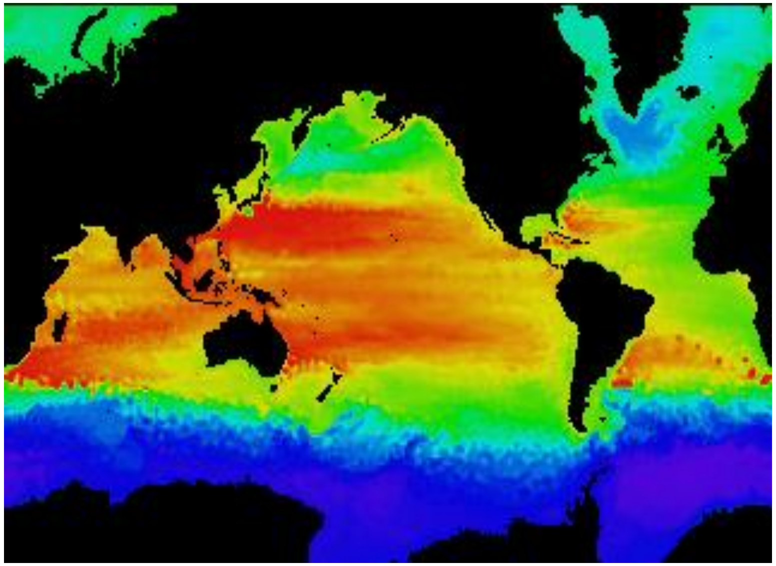
- Problem is to compute:

  f(latitude, longitude, elevation, time) →

  temperature, pressure, humidity, wind velocity
- Atmospheric model: equation of fluid dynamics →

  Navier-Stokes system of nonlinear partial differential equations
- Approach:
  - Discretize the domain, e.g., a measurement point every 1km
  - Devise an algorithm to predict weather at time t+1 given t

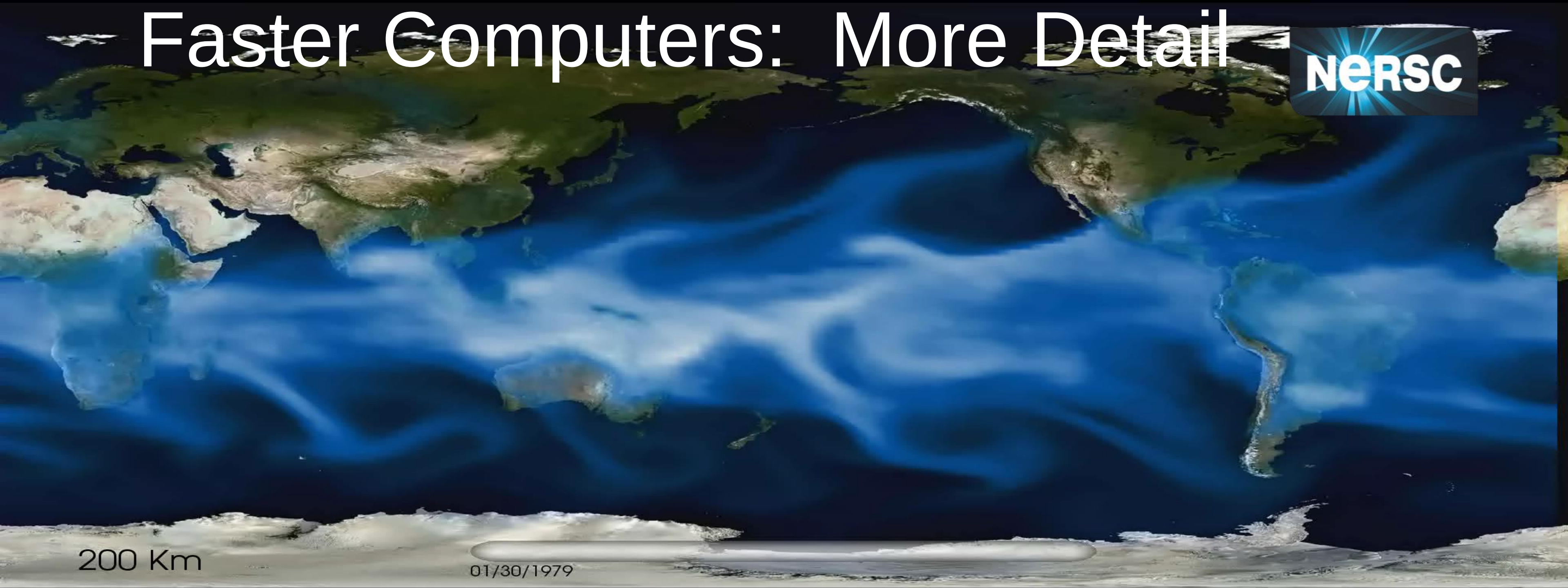
- Uses:
  - Predict major events, e.g., El Nino
  - Use in setting air emissions standards

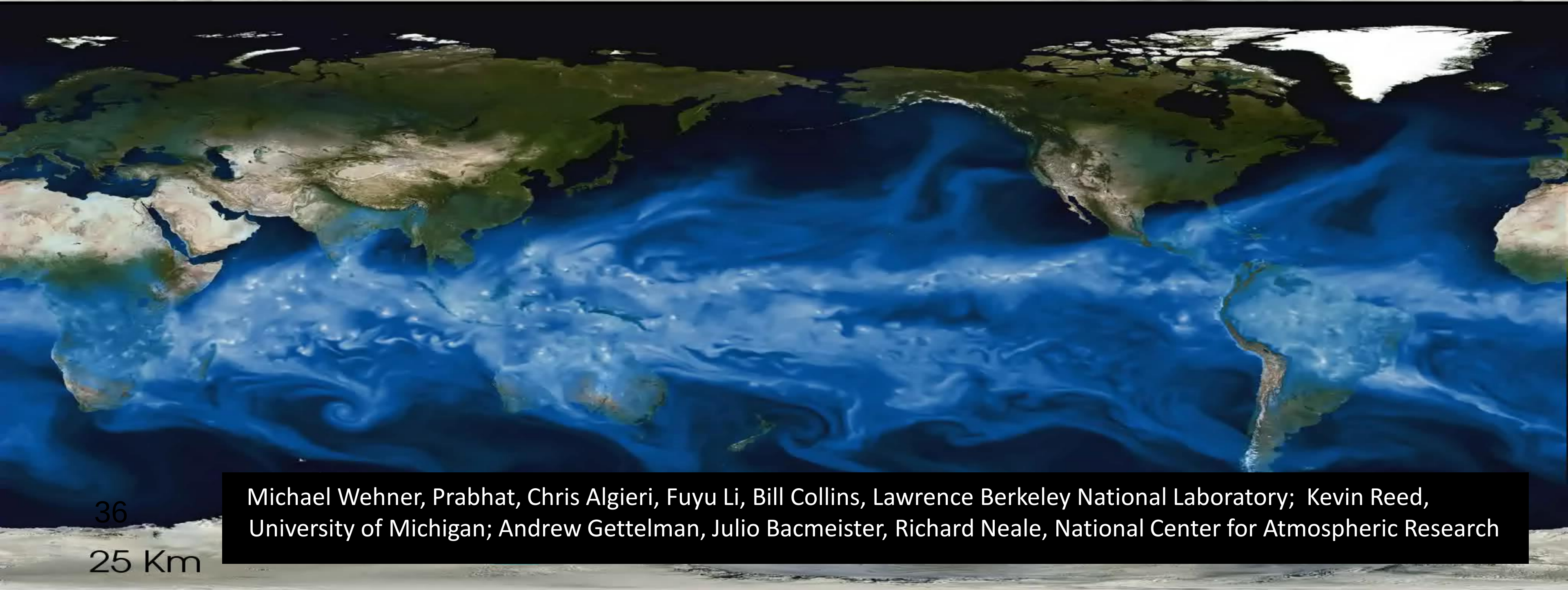


Source: http://www.epm.ornl.gov/chammp/chammp.html

Faster Computers: More Detail

NERSC

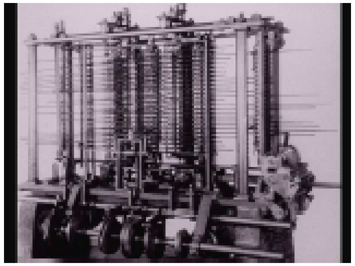




Michael Wehner, Prabhat, Chris Algieri, Fuyu Li, Bill Collins, Lawrence Berkeley National Laboratory; Kevin Reed, University of Michigan; Andrew Gettelman, Julio Bacmeister, Richard Neale, National Center for Atmospheric Research

Supercomputer

- Supercomputers are the most «powerful» computing devices that are available in a certain time period.
- Powerful in the sense of execution velocity, memory capacity and precision



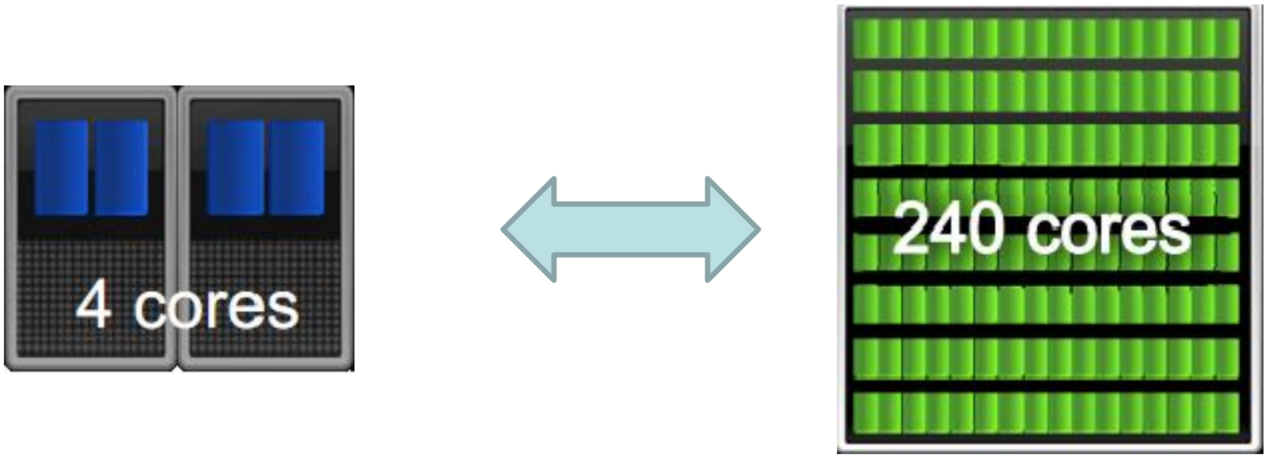
Supercomputer:"*new statistical machines with the mental power of 100 skilled mathematicians in solving even highly complex algebraic problems*"..

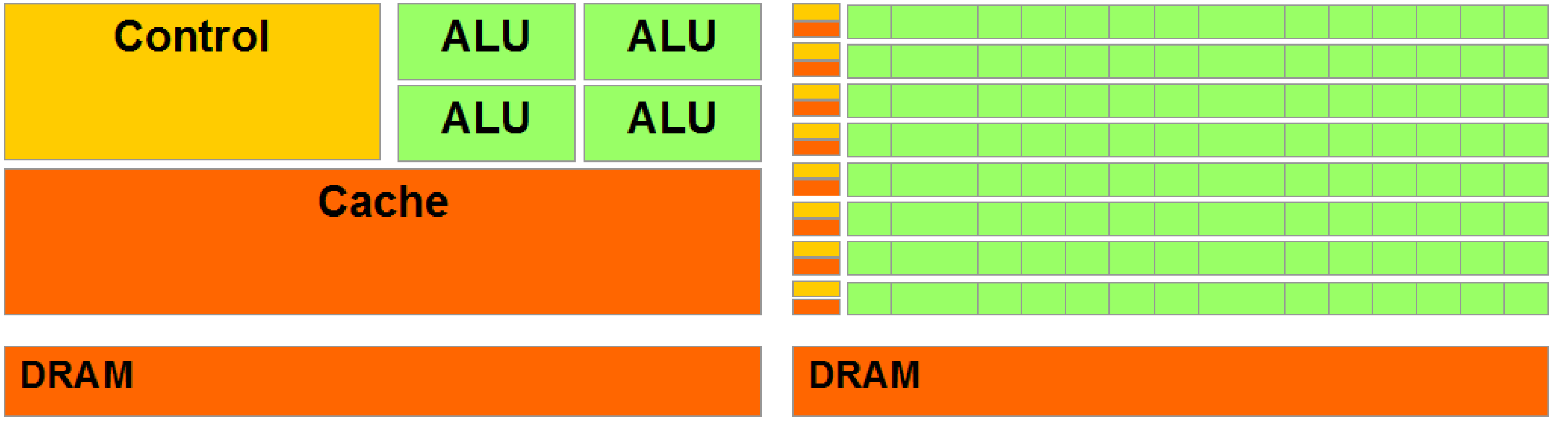
New York World, march 1920

GPGPU and CUDA

- **GPGPU** (**General Purpose programing on GPUs**): Use of graphics processor units (GPU) for purposes other than the traditional three-dimensional creating processes
- GPUs are **high performance multi-core** processors
- The first programmable solutions date back to 2006, were previously dedicated only to the development of graphics and video games.
- GPUs are now considered as parallel processors with general-purpose programming interfaces with support for programming languages such as **CUDA-C** (nVidia) or **Stream** (ATI).
- **OpenCL** standard, framework for writing programs that execute across **heterogeneous platforms** consisting of central processing units (CPUs) and graphics processing units (GPUs),

Macroscopic differences CPU / GPU





… aim of Parallel Computing …

Speedup factor

$$S(n) = \frac{\text{Execution time using one processor (single processor system)}}{\text{Execution time using a multiprocessor with n processors}} = \frac{t_s}{t_p}$$

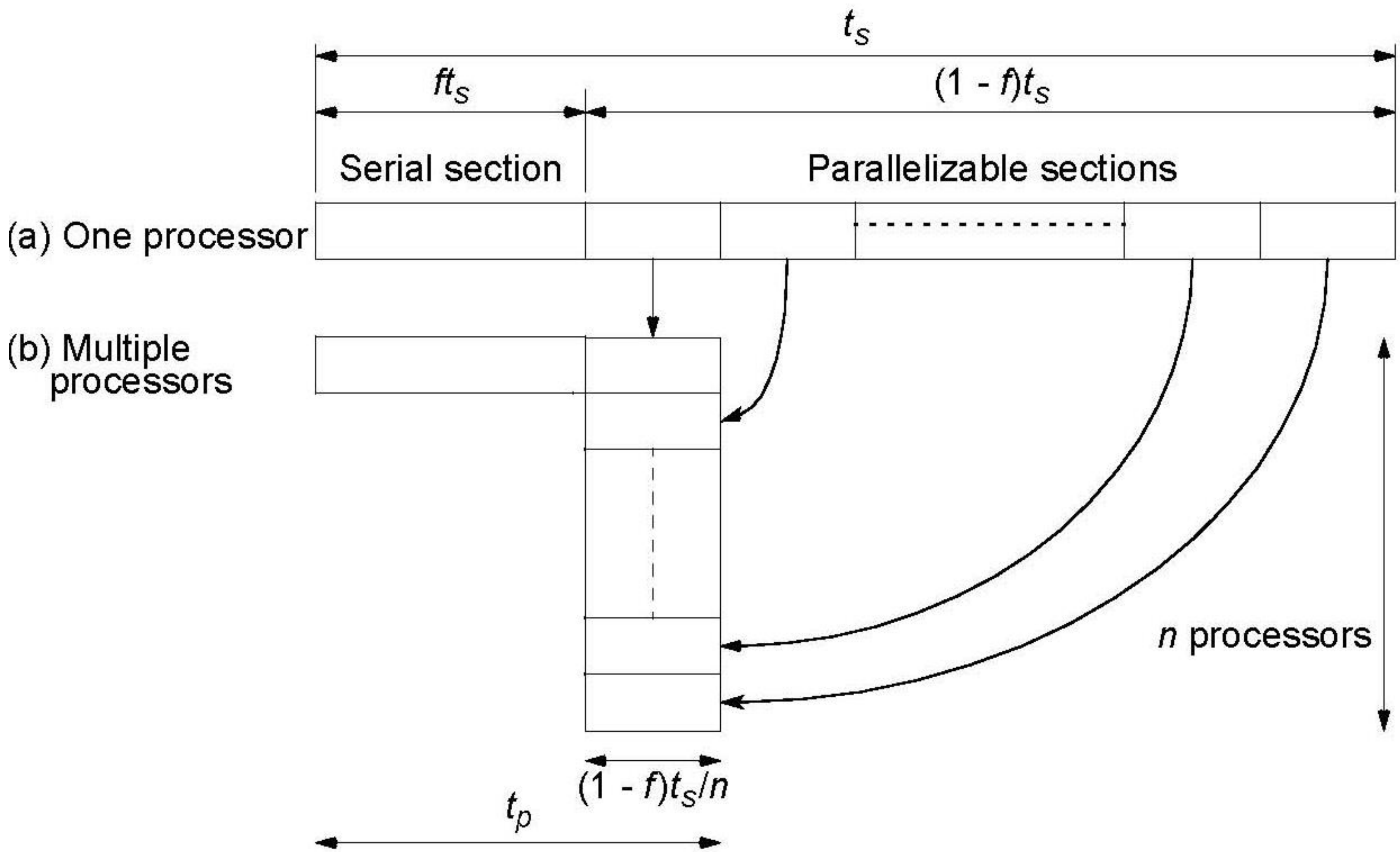
where t_s is execution time on a single processor and t_p is execution time on a multiprocessor. $S(n)$ gives increase in speed by using multiprocessor. Underlying algorithm for parallel implementation might be (and is usually) different.

Speedup factor can also be cast in terms of computational steps:

$$S(n) = \frac{\text{Number of computational steps using one processor}}{\text{Number of parallel computational steps with n processors}}$$

Maximum speedup is (usually) n with n processors (*linear speedup*).

Maximum Speedup – Amdahl's law



Maximum Speedup – Amdahl's law

Speedup factor is given by:

$$S(n) = \frac{t_s}{ft_s + (1 - f)t_s/n} = \frac{n}{1 + (n - 1)f}$$

This equation is known as *Amdahl's law*

Amdahl's Law

- Thus, for n $\to \infty$:

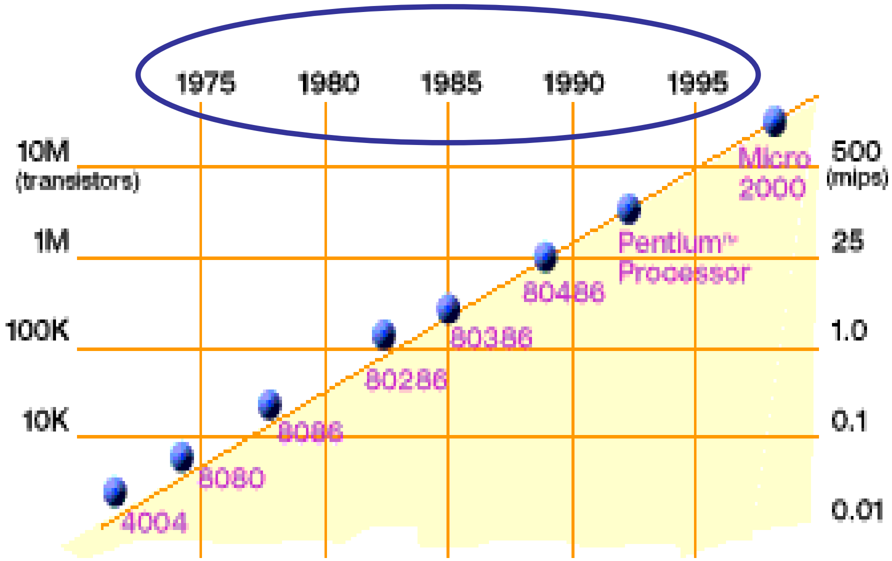
$$S(n) \to \frac{1}{f}$$

- For instance, if the serial fraction is f = 5% (very plausible!) the maximum speed-up is 20!

- OBS The **serial fraction** represents that "fraction" of code that can not be **parallelized** (eg I / O, critical sections, etc.)

… during the course, the solution to the problem!

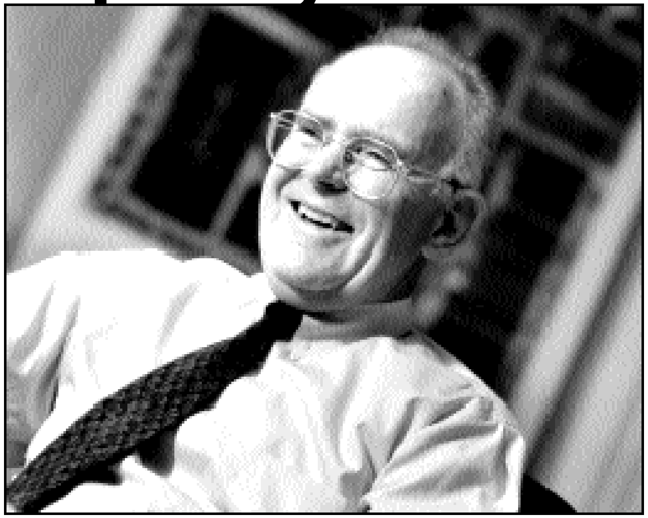
Technology Trends: Microprocessor Capacity



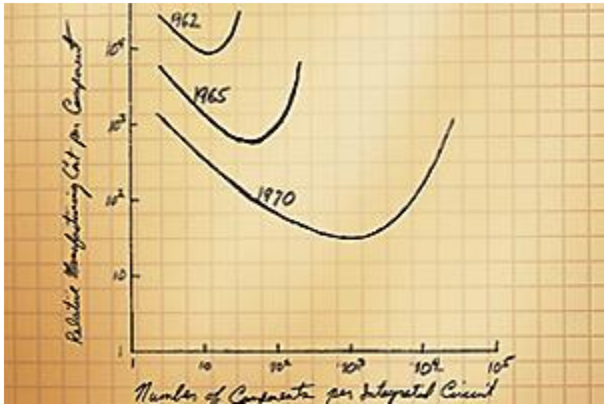
2X transistors/Chip Every 1.5 - 2 years

Called "[Moore's Law]"

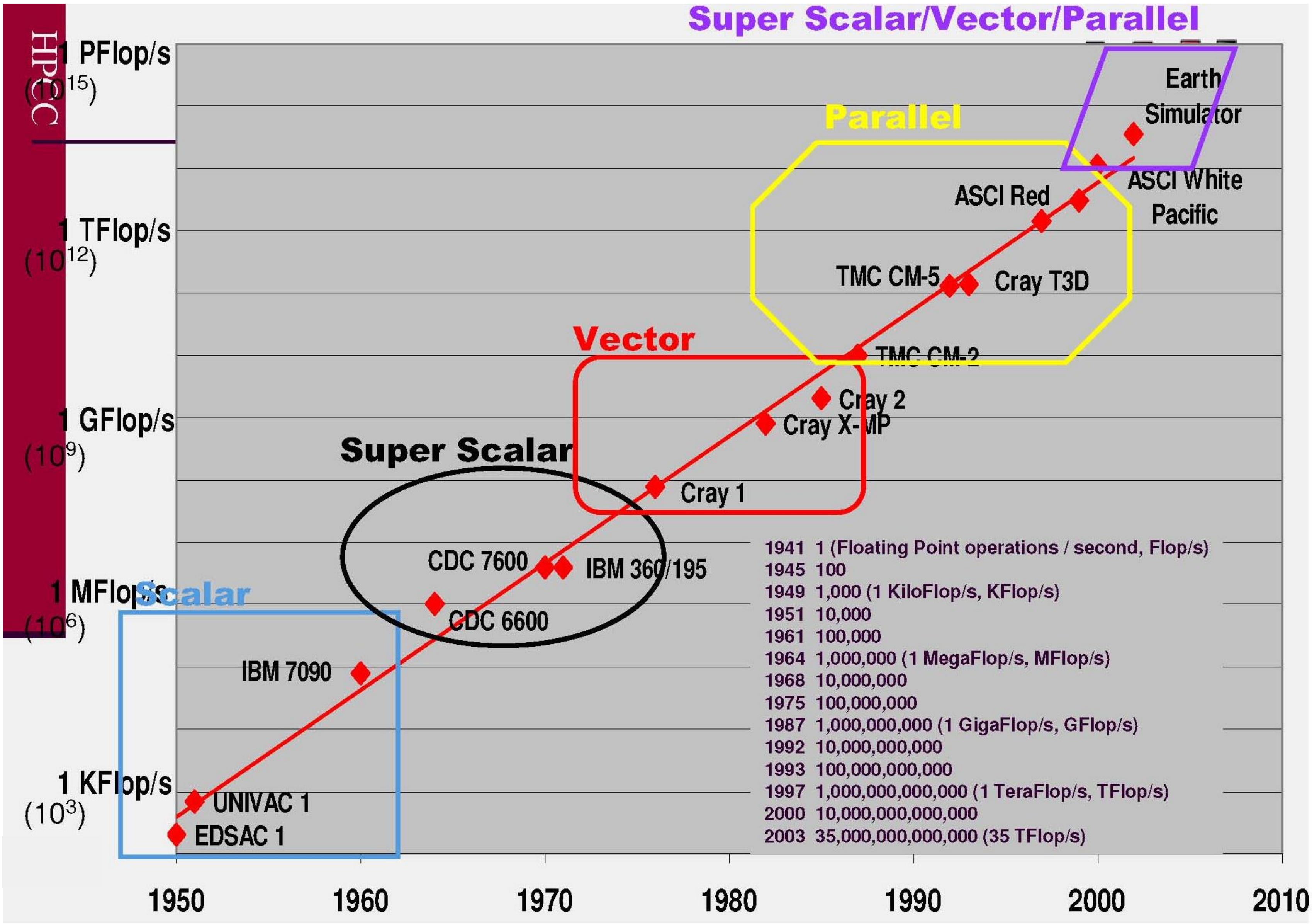
Microprocessors have become smaller, denser, and more powerful.



Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

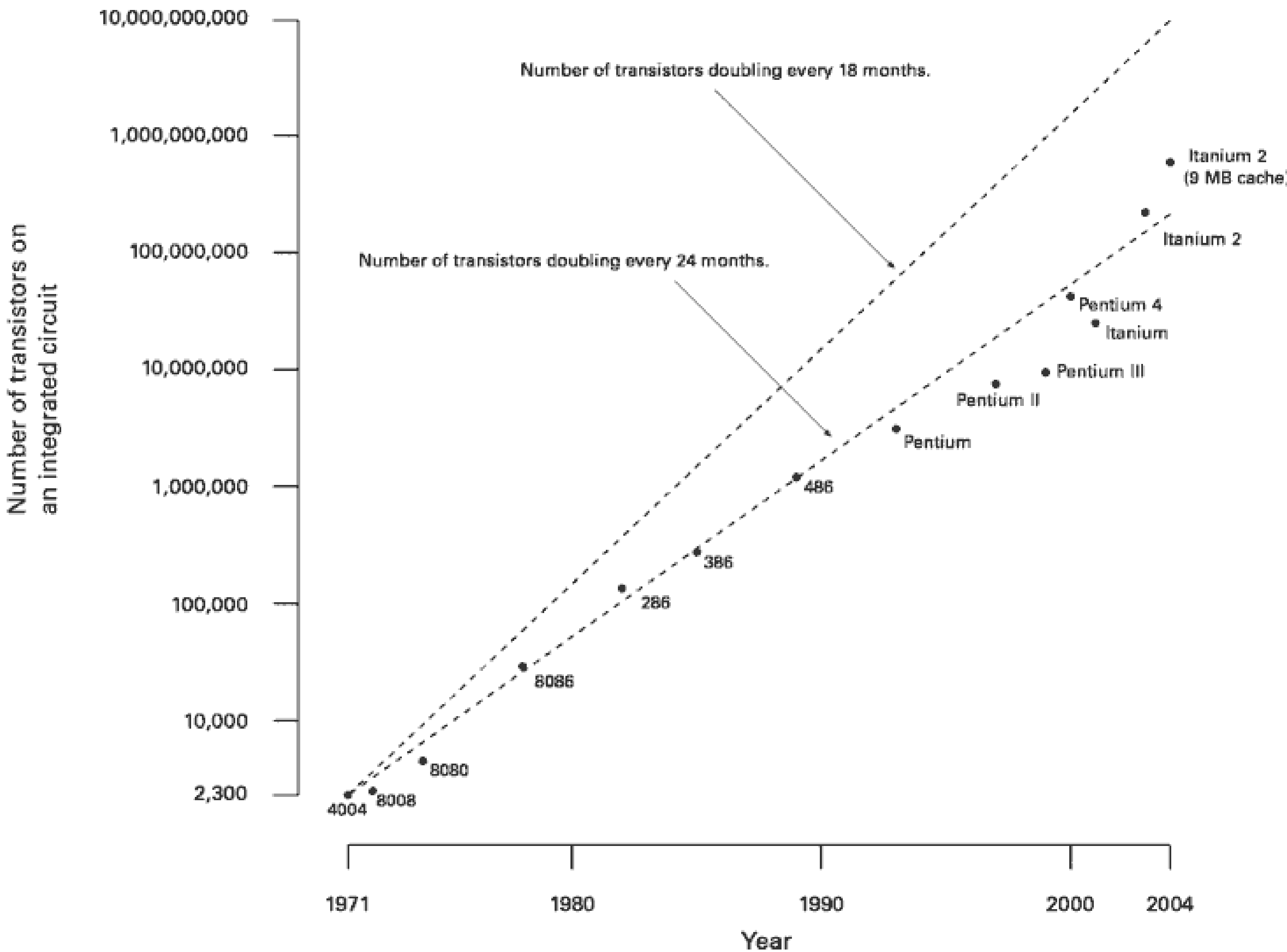


Moore's Law



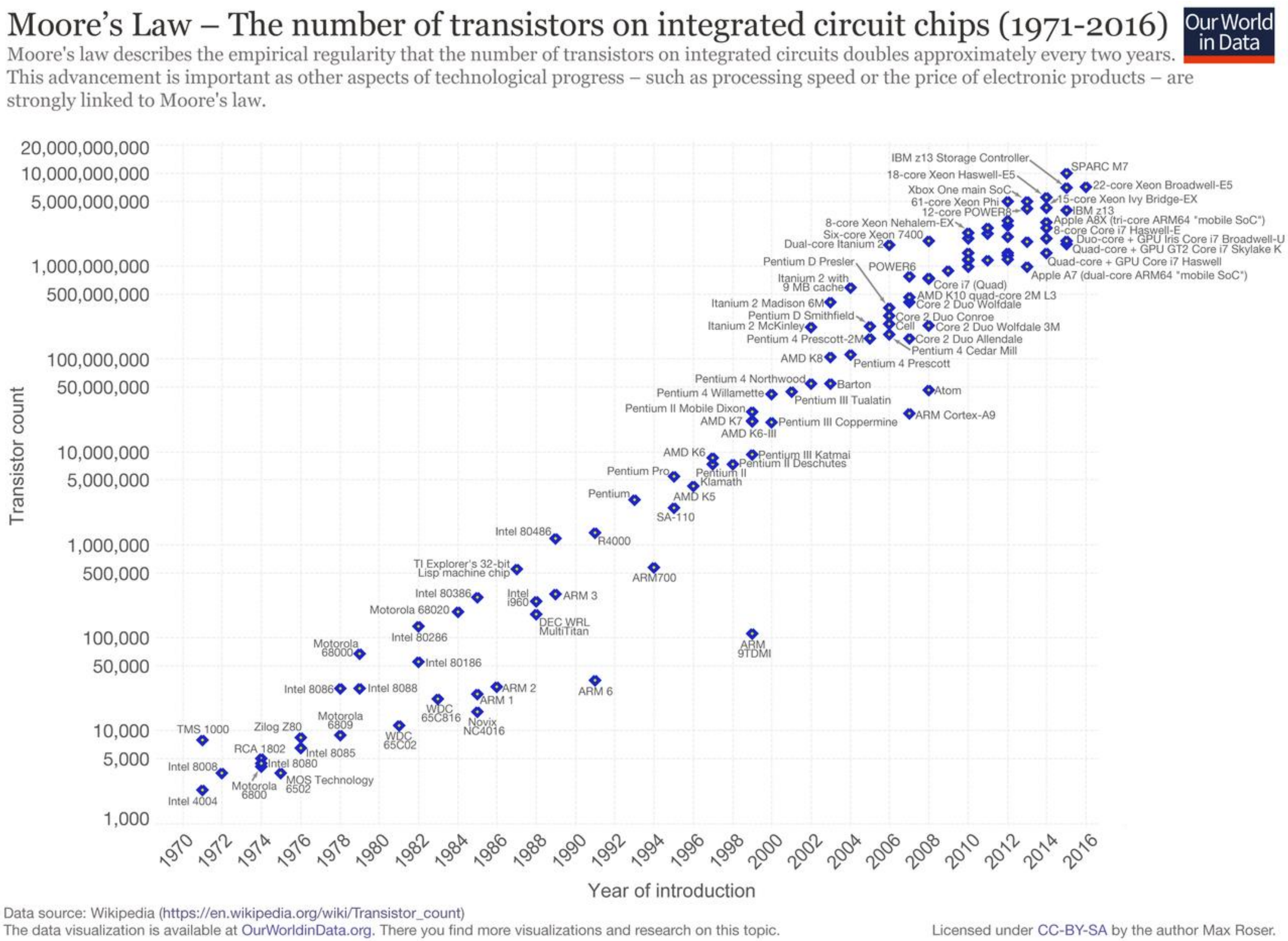
Super Scalar/Vector/Parallel
1 PFlop/s
(10^{15})
1 TFlop/s
(10^{12})
1 GFlop/s
(10^{9})
1 MFlop/s
(10^{6})
1 KFlop/s
(10^{3})
HPEC
Earth Simulator
Parallel
ASCI White Pacific
ASCI Red
TMC CM-5
Cray T3D
Vector
TMC CM-2
Cray 2
Cray X-MP
Super Scalar
Cray 1
CDC 7600
IBM 360/195
Scalar
CDC 6600
IBM 7090
UNIVAC 1
EDSAC 1
1941 1 (Floating Point operations / second, Flop/s)
1945 100
1949 1,000 (1 KiloFlop/s, KFlop/s)
1951 10,000
1961 100,000
1964 1,000,000 (1 MegaFlop/s, MFlop/s)
1968 10,000,000
1975 100,000,000
1987 1,000,000,000 (1 GigaFlop/s, GFlop/s)
1992 10,000,000,000
1993 100,000,000,000
1997 1,000,000,000,000 (1 TeraFlop/s, TFlop/s)
2000 10,000,000,000,000
2003 35,000,000,000,000 (35 TFlop/s)
1950
1960
1970
1980
1990
2000
2010

Moore’s Law



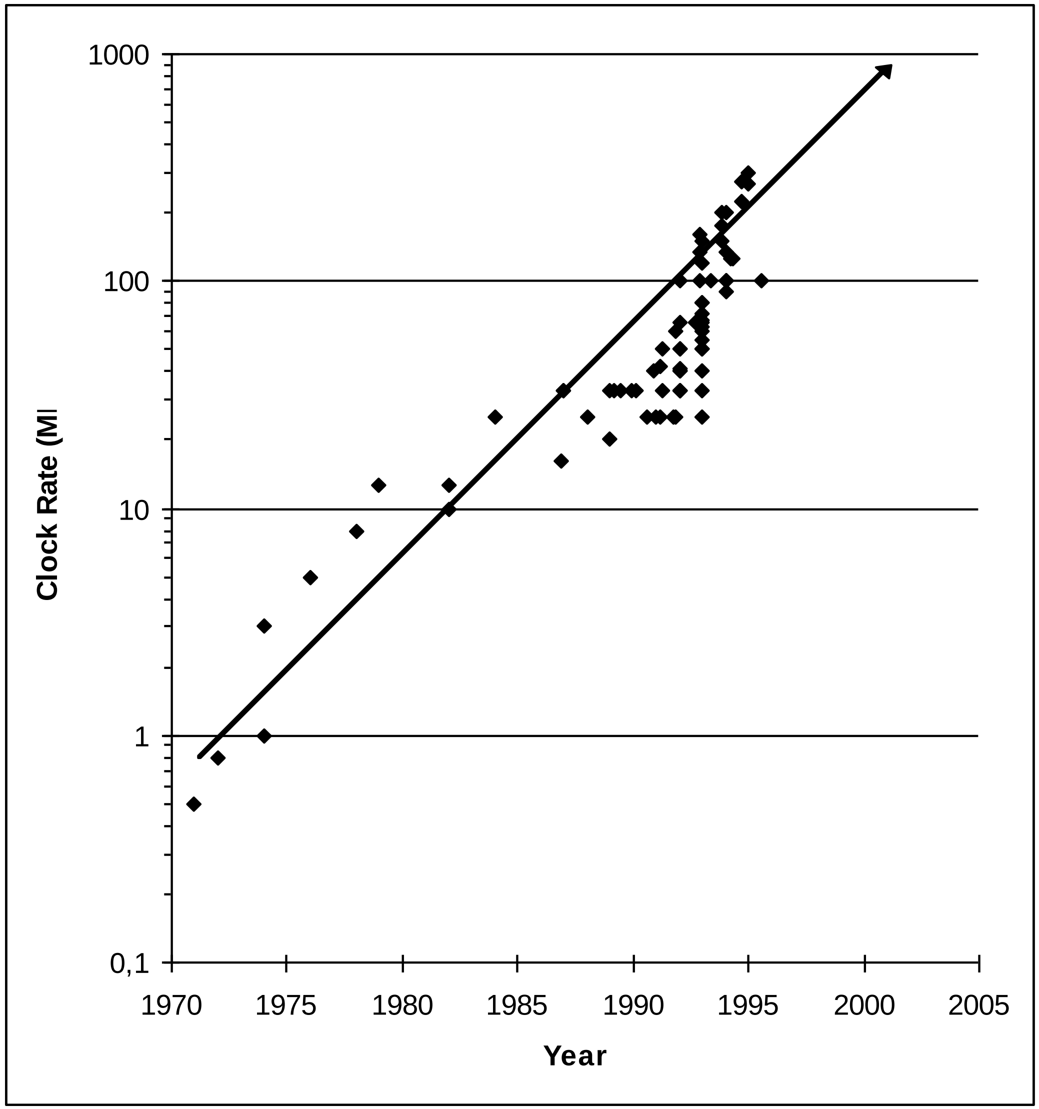
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
Number of transistors on an integrated circuit
Number of transistors doubling every 18 months.
Number of transistors doubling every 24 months.
Itanium 2 (9 MB cache)
Itanium 2
Pentium 4
Itanium
Pentium III
Pentium II
Pentium
486
386
286
8086
8080
8008
4004
1971
1980
1990
2000
2004
Year

Moore’s Law

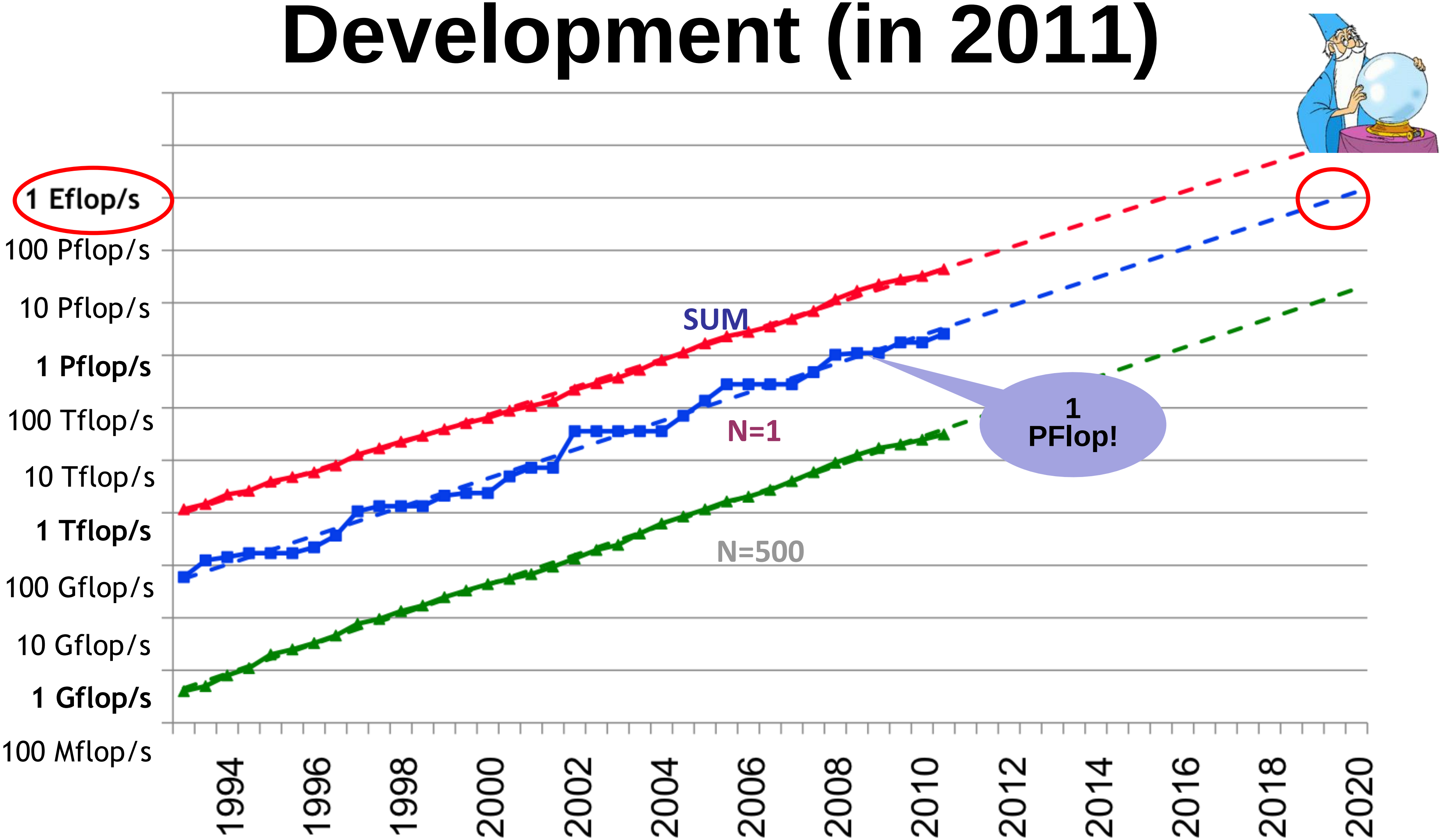


Moore’s Law – The number of transistors on integrated circuit chips (1971-2016)
Our World in Data
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.
Transistor count
20,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
Year of introduction
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
IBM z13 Storage Controller
SPARC M7
18-core Xeon Haswell-E5
22-core Xeon Broadwell-E5
Xbox One main SoC
15-core Xeon Ivy Bridge-EX
61-core Xeon Phi
12-core POWER8
IBM z13
Apple A8X (tri-core ARM64 "mobile SoC")
8-core Xeon Nehalem-EX
8-core Core i7 Haswell-E
Six-core Xeon 7400
Duo-core + GPU Iris Core i7 Broadwell-U
Dual-core Itanium 2
Quad-core + GPU GT2 Core i7 Skylake K
Quad-core + GPU Core i7 Haswell
Pentium D Presler
POWER6
Apple A7 (dual-core ARM64 "mobile SoC")
Itanium 2 with 9 MB cache
Core i7 (Quad)
AMD K10 quad-core 2M L3
Itanium 2 Madison 6M
Core 2 Duo Wolfdale
Pentium D Smithfield
Core 2 Duo Conroe
Itanium 2 McKinley
Cell
Core 2 Duo Wolfdale 3M
Pentium 4 Prescott-2M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
AMD K8
Pentium 4 Prescott
Pentium 4 Northwood
Barton
Pentium 4 Willamette
Pentium III Tualatin
Atom
Pentium II Mobile Dixon
AMD K7
Pentium III Coppermine
ARM Cortex-A9
AMD K6-III
AMD K6
Pentium III Katmai
Pentium II Deschutes
Pentium Pro
Pentium II Klamath
Pentium
AMD K5
SA-110
Intel 80486
R4000
TI Explorer's 32-bit Lisp machine chip
ARM700
Intel 80386
Intel i960
ARM 3
Motorola 68020
DEC WRL MultiTitan
Intel 80286
ARM 9TDMI
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
ARM 1
ARM 6
WDC 65C816
Novix NC4016
Motorola 6809
TMS 1000
Zilog Z80
WDC 65C02
RCA 1802
Intel 8085
Intel 8080
Intel 8008
MOS Technology 6502
Motorola 6800
Intel 4004
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.
Licensed under CC-BY-SA by the author Max Roser.

Microprocessor Clock Rate



Projected Performance Development (in 2011)



… infact

DATA CENTER EXPLORER

By Andy Patrizio, Network World | APR 1, 2020 9:00 PM PDT

About

Andy Patrizio is a freelance technology writer based in Orange County, California. He's written for a variety of publications, ranging from Tom's Guide to Wired to Dr. Dobbs Journal.

Thousands of PCs break exaFLOP barrier

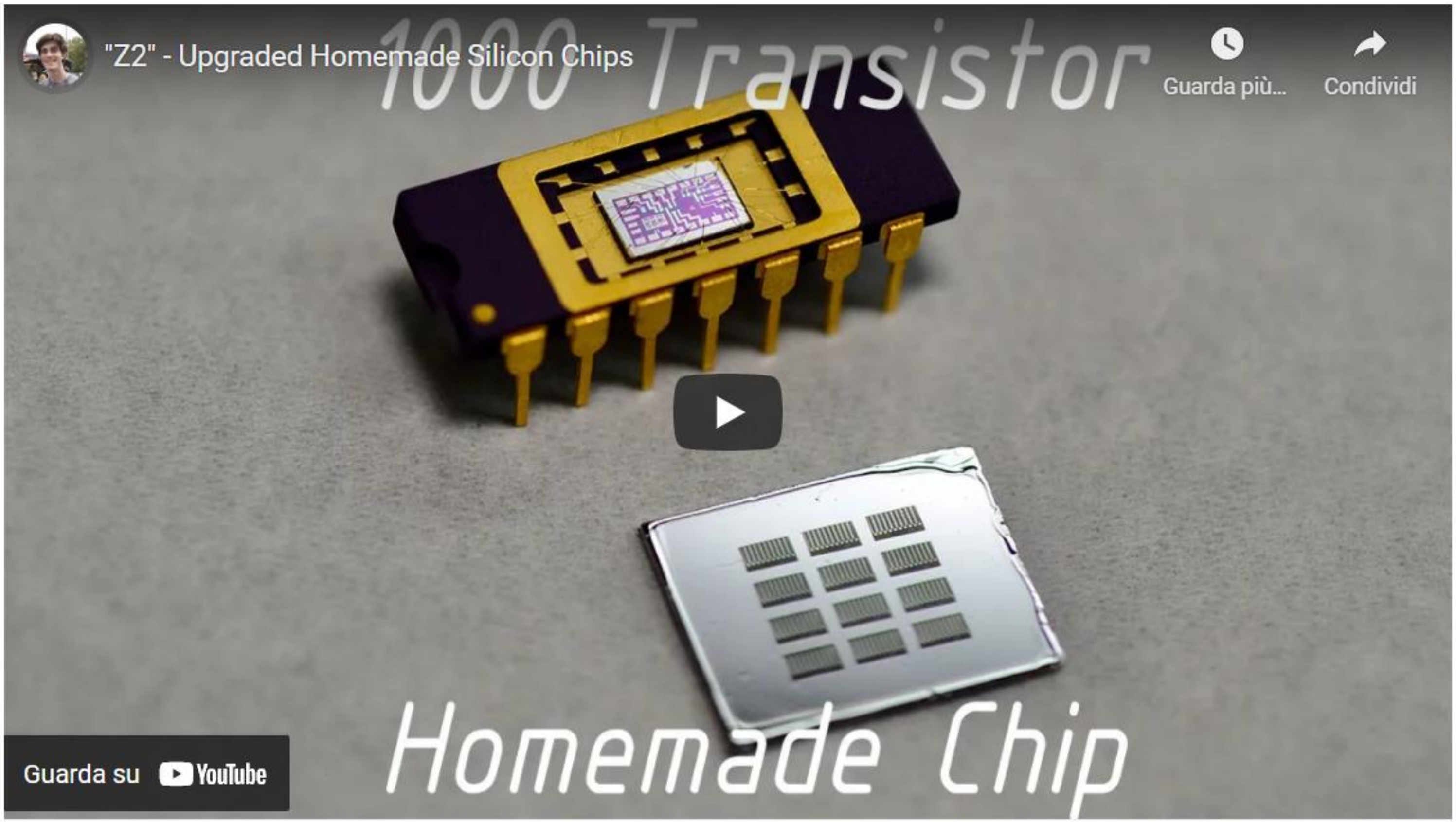
Folding@home has done what IT vendors and the federal government have been racing to do – break the exaFLOP barrier – and the crowdsourced distributed-computing program did it while fighting coronavirus.

Moore busted?

NOTIZIE TECNOLOGIA

SAM ZELOOF, IL RAGAZZO CHE HA "ROTTO" LA LEGGE DI MOORE CON UN CHIP FAI-DA-TE

Di **Alberto Lala**, 23 Gennaio 2022, ore 11:03



https://youtu.be/IS5ycm7VfXg