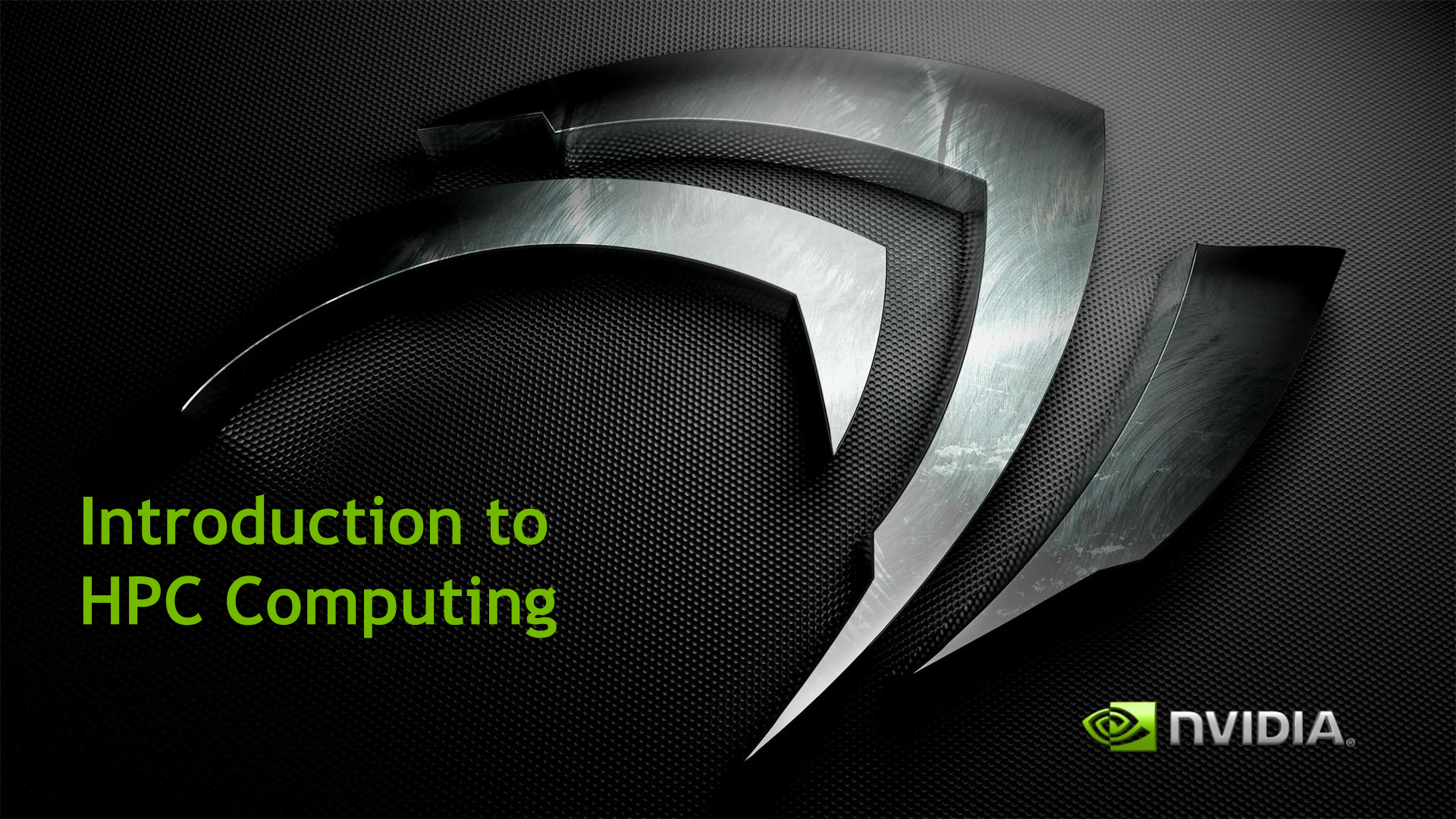
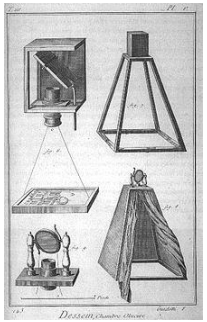

Introduction to
HPC Computing
NVIDIA

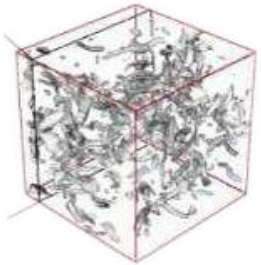
Agenda

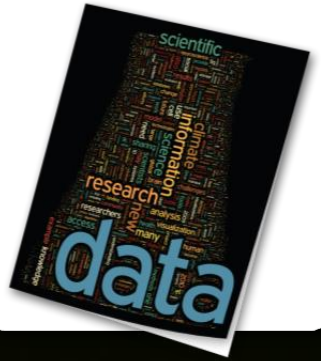
- HPC Definition, Customers and Motivations
- HPC Principles and Common Architectures
- HPC Challenges

Computation: 3rd Pillar of Scientific Research



$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi G\rho}{3} - \mathrm{K}\frac{c^2}{a^2}$$





1,000 years ago

Experimental

Description of natural phenomena

Experimental methods and quantification

Last 500 years

Theoretical

Formulation of Newton's laws, Maxwell's equations …

Last 50 years

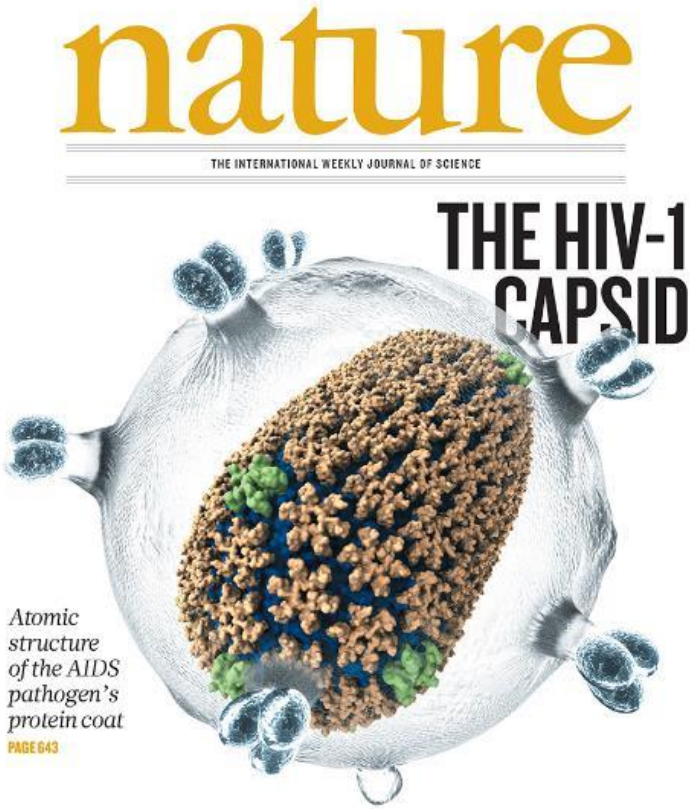
Computational

Simulation of complex phenomena

Today

Data

Distributed communities unifying theory, experiment and simulation with massive data sets from multiple sources and disciplines



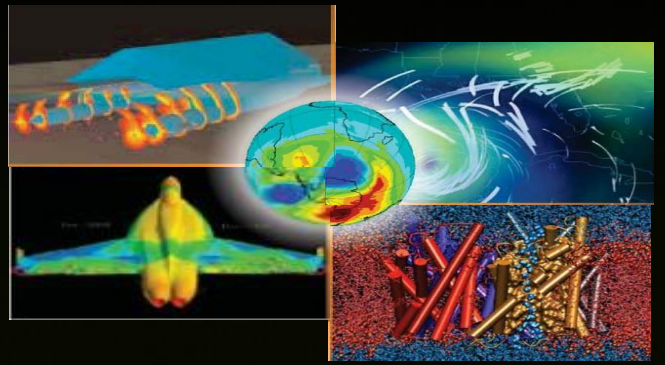
ACCELERATING discoveries

USING A SUPERCOMPUTER POWERED BY 3,000 TESLA PROCESSORS, UNIVERSITY OF ILLINOIS SCIENTISTS PERFORMED THE FIRST ALL-ATOM SIMULATION OF THE HIV VIRUS AND DISCOVERED THE CHEMICAL STRUCTURE OF ITS CAPSID — "THE PERFECT TARGET FOR FIGHTING THE INFECTION."

WITHOUT GPU, THE SUPERCOMPUTER WOULD NEED TO BE 5X LARGER FOR SIMILAR PERFORMANCE.

High Performance Computing

- High Performance Computing refers to the use of high advanced computational capabilities to solve cutting edge problems in science, engineering and business.
- The complex problems HPC has to deal with require capabilities beyond those of standalone computers
  - Aircraft design utilizing composite materials
  - Vehicle fuel efficiency and safety improvements
  - Simulations of enzyme catalysis, protein folding
  - Reservoir Simulation & Seismic Processing
  - Targeted material and drug design
  - Financial portfolio risk modeling
  - …
- High Performance Computing enables scientists and engineers to perform simulations to:
  - Investigate phenomena where economics and constraints preclude experimentation
  - Evaluate complex models and manage massive data volumes
  - Transform business and engineering practices



Simulation is key for Scientists and Engineers

- *Previously*: Theory – Experiments
- *Now*: Theory – Simulations – Experiments

Who is using High Performance Computing?

Digital Media

Digital content creation, management and distribution

Petroleum

Oil and gas exploration and production

Industrial/Product Lifecycle Management

CAE, EDA, CAD/PDM for electronics, automotive, and aerospace

Life Sciences

Research, drug discovery, diagnostics, information-based medicine

Business Intelligence

Data warehousing and data mining

Financial Services

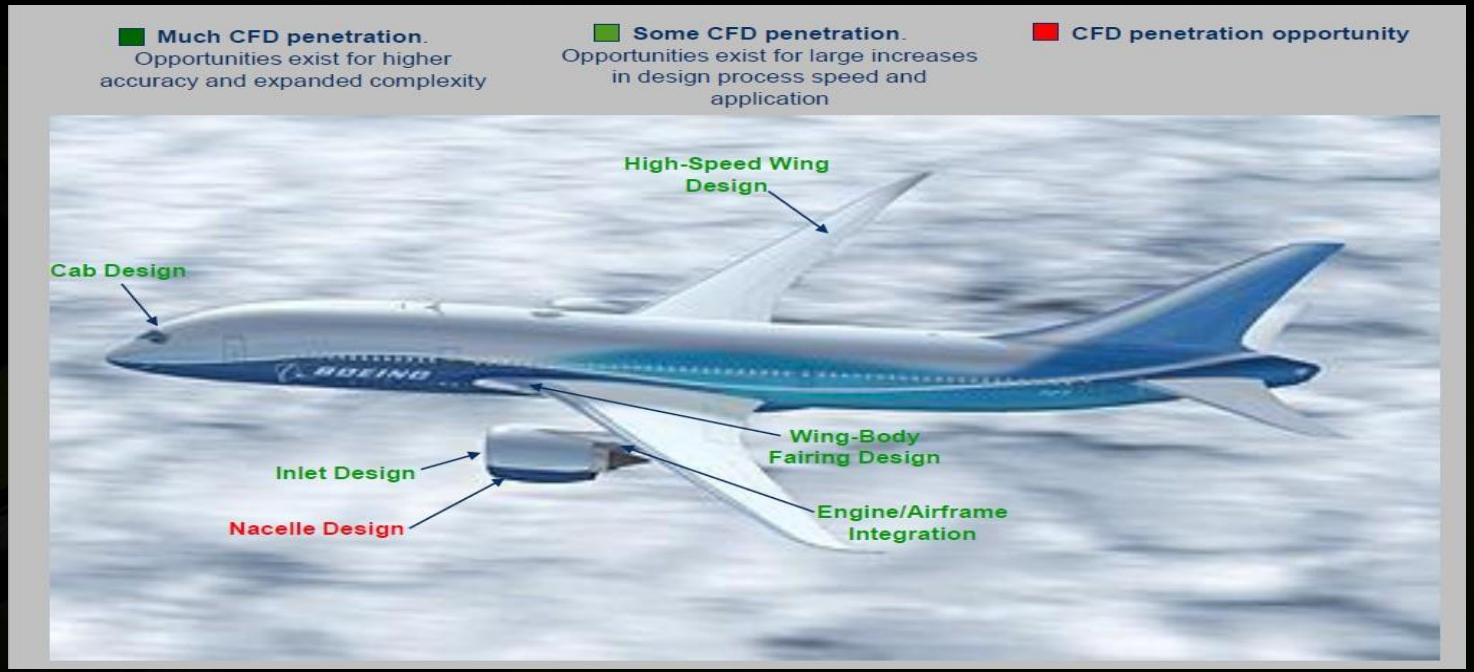
Optimizing IT infrastructure, risk management and compliance, analytics

Government & Higher Education

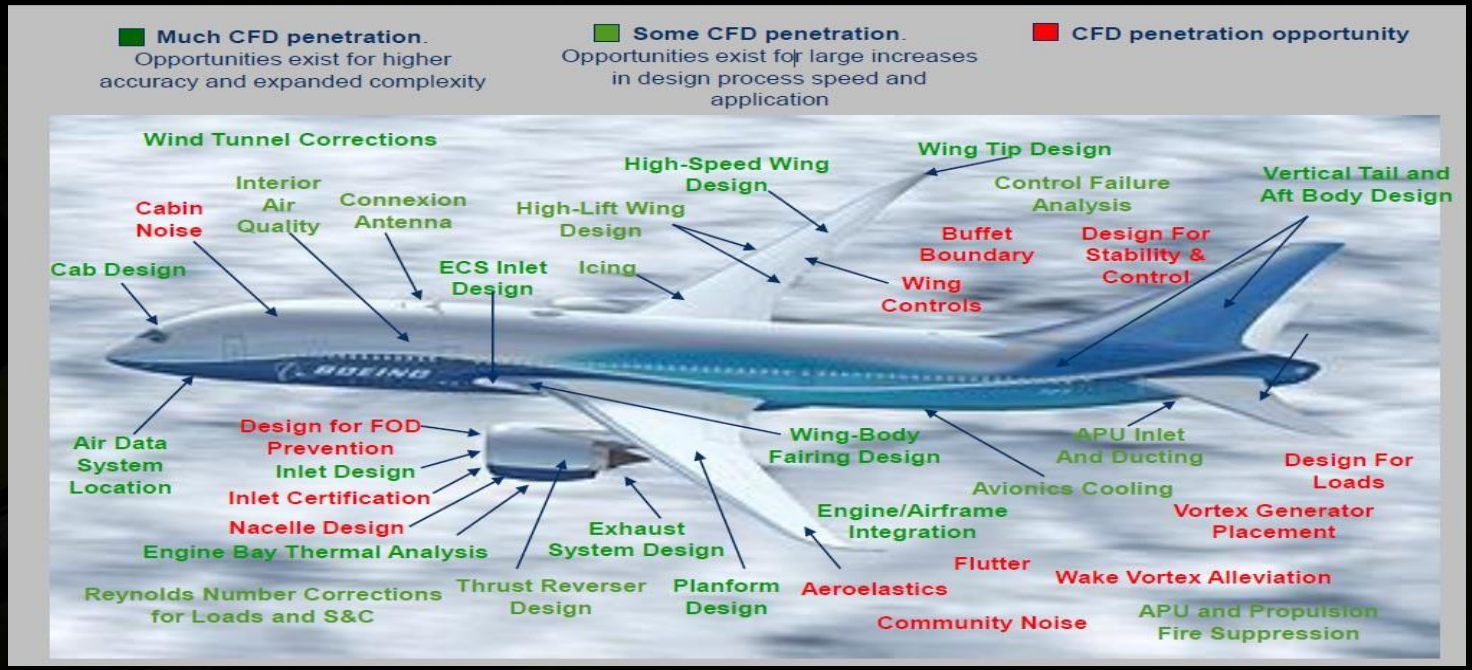
Scientific research, classified/defense, weather/environmental sciences

Weather Forecast

CFD Contribution to the 767 in 1975



CFD Contribution to the 767 in 2005



HPC Units

- High Performance Computing usually try to solve data- and/or compute-intensive problems
- The number of floating point operations per second – called Flops – is the unit used to evaluate compute intensive workloads and the computational performance of HPC computers.
  - Latest processors are able to issue 4 Floating Point Operations per clock cycle per core.
  - Ex: Dual Core processor @ 2.8GHz – 4 x 2 x 2.8 = 22.4 GFlops = 22.6 10^9 Flops
  - Ex: RoadRunner ~ 1 PFlops = 10^{15} Flops
- The amount of data being used during HPC workloads is measure is Bytes.

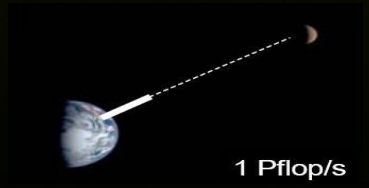
  We also measure the amount of time needed for the data to be transferred within the different components of a HPC system in seconds.

- Mega = $2^{20} \approx 10^6$
- Giga = $2^{30} \approx 10^9$
- Tera = $2^{40} \approx 10^{12}$
- Peta = $2^{50} \approx 10^{15}$

Let's say you can write 1000 numbers on 1 sheet of paper 0.1mm thick,

10^{15} numbers represents 100.000km of stacked sheets = ¼ distance to the moon

(Source: Jack Dongarra)



HPC Units(Continued)

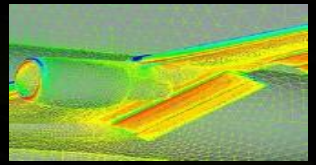
- High Performance Computing (HPC) units are:
  - Flop: floating point operation, usually double precision unless noted
  - Flop/s: floating point operations per second
  - Bytes: size of data (a double precision floating point number is 8 bytes)
- Typical sizes are millions, billions, trillions…

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

- Current fastest (public) machine ~ 55 Pflop/s, 3.1M cores
  - Up-to-date list at www.top500.org

What Drives HPC? – "The Need for Speed…"

Computational Needs exceed the Petaflops Range



- CFD Wing Simulation

512x64x256 Grid
(8.3 x 10^6 mesh points)
5000 Flops per mesh points
5000 time steps / cycle
2.15 x 10^{14} Flop

Source: A Jameson, et al

- CFD Full Plane Simulation

512x64x256 Grid
(3.5 x 10^{17} mesh points)
5000 Flops per mesh points
5000 time steps / cycle
8.17 x 10^{24} Flop

- Digital Movies and Special Effects

~1E14 Flops per frame
50 frames / sec
90 mins movie
2.7 x 10^{19} Flop
~150 days on 2000 1 GFlops CPU

- Spare Parts Inventory Planning

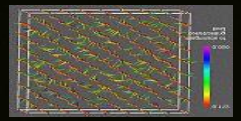
Modeling the optimized deployment of 10,000 part numbers across 100 parts depots and requires:**2.14 x 10^{15} Flops**

(1 hour turnaround time)

Source: D Bailey, NERSC

- Industry trend for rapid, frequent modeling for timely business decision support drives higher sustained performance

- Materials Science



Source: B Dietrich, IBM

Magnetic Materials:
Future: HDD Simulation – **30TF, 2TB**
Current: 2000 atoms – **2.64TF, 512GB**

Electronic Structures:
Current: 300 atoms – **0.5TF, 100GB**
Future: 3000 atoms – **50TF, 2TB**

ACCELERATING INSIGHTS

"Now You Can Build Google's $1M Artificial Brain on the Cheap"

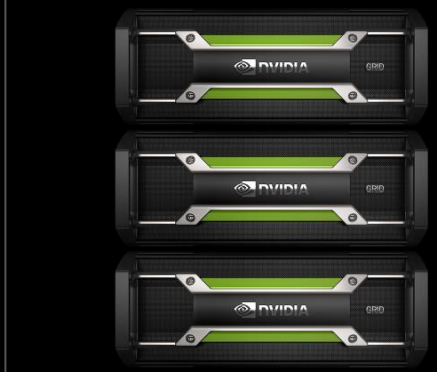
WIRED

GOOGLE DATACENTER

1,000 CPU Servers
2,000 CPUs • 16,000 cores

600 kWatts
$5,000,000

STANFORD AI LAB

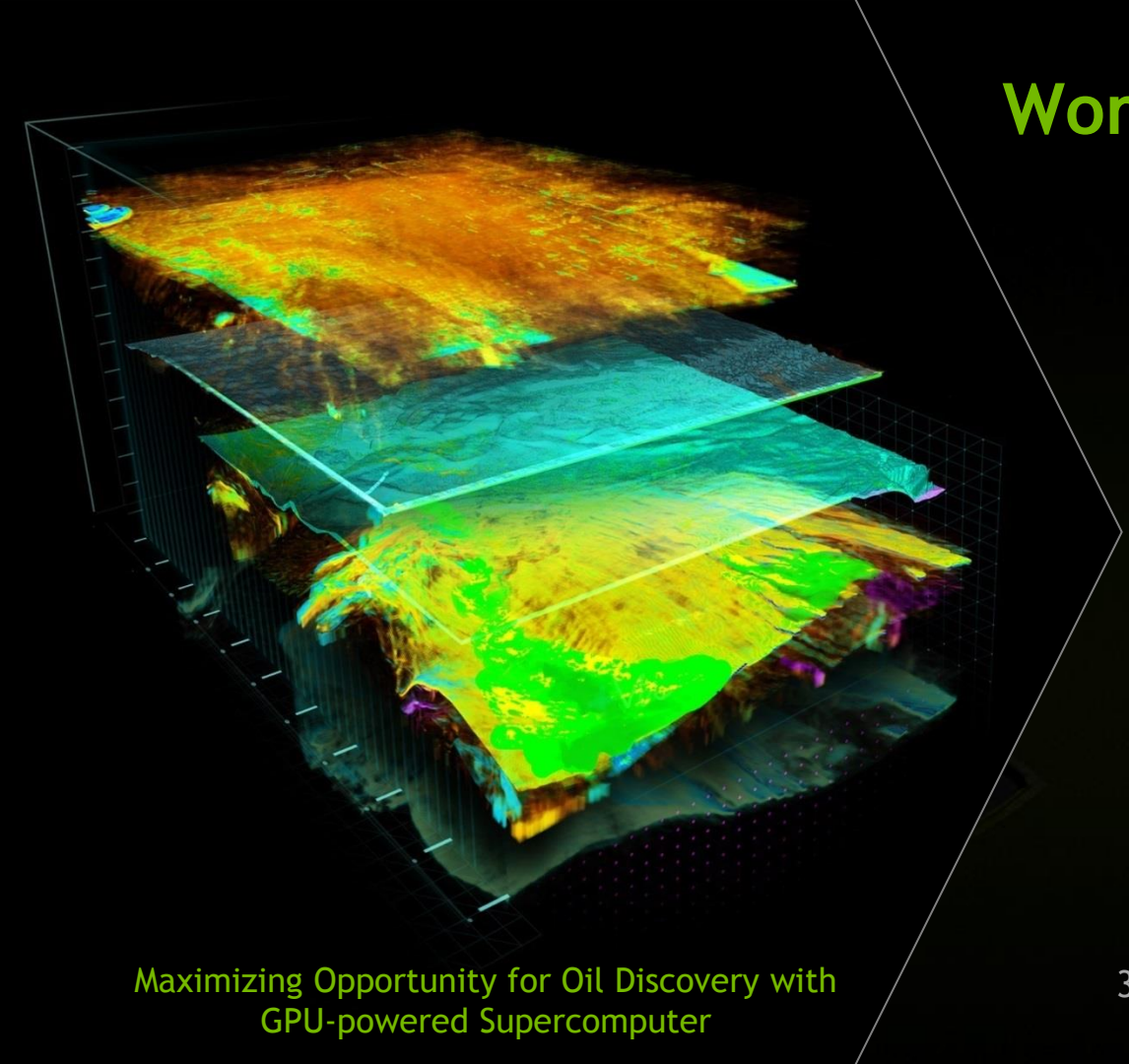


3 GPU-Accelerated Servers
12 GPUs • 18,432 cores

4 kWatts
$33,000

Deep learning with COTS HPC systems, A. Coates, B. Huval, T. Wang, D. Wu, A. Ng, B. Catanzaro ICML 2013

World's Fastest Enterprise Supercomputer



Maximizing Opportunity for Oil Discovery with GPU-powered Supercomputer

eni

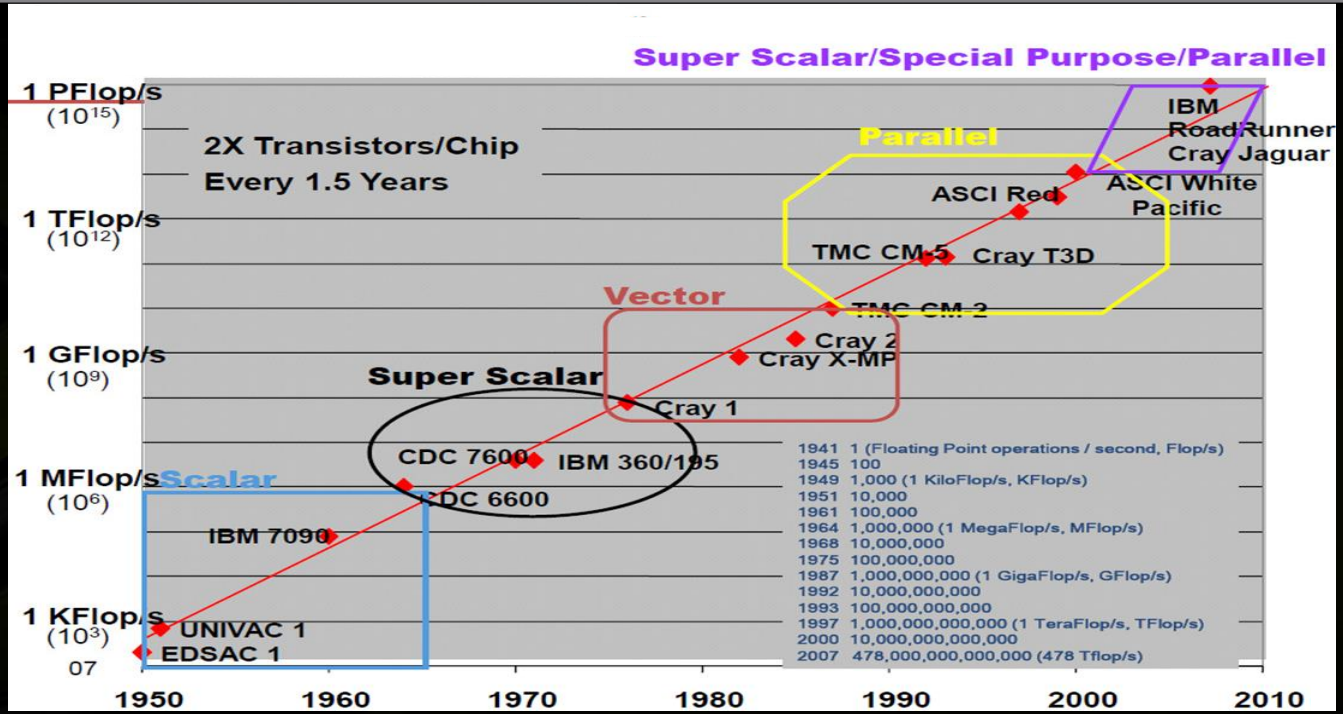
IBM

3 Petaflops Linpack Performance

Most Energy-Efficient Petascale System in the World

3,000 NVIDIA Tesla K20X GPU Accelerators

History of fastest Architectures

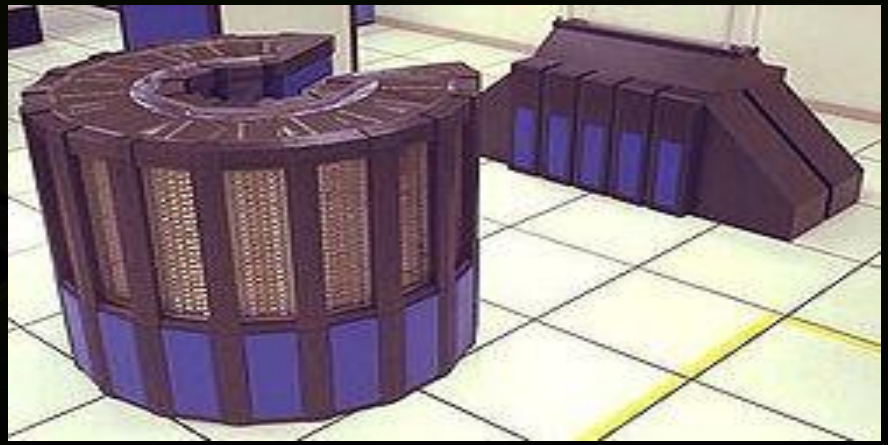


History of fastest Architectures Old times

- Basic ideas
  - A super computer is like a Ferrari → use specific components
  - Super computing is like F1 or WRG → adapt to the application scenario
- Vector Computer (Cray/ETA/Convex)
- Database Computer

- An alternative idea: Super SIMD (Connection Machines)

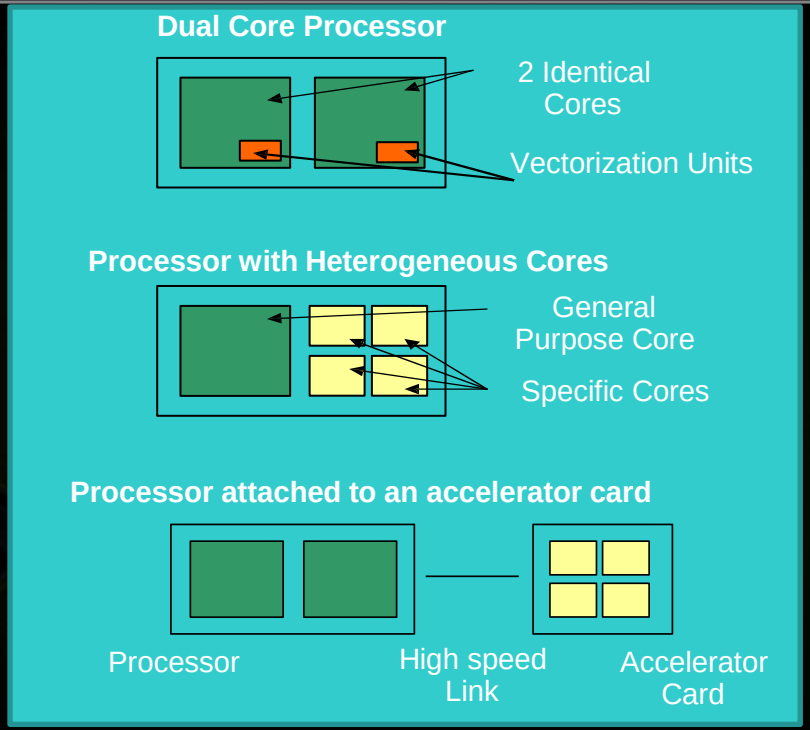
- Incredible creativity in architecture and network design



A Cray-2

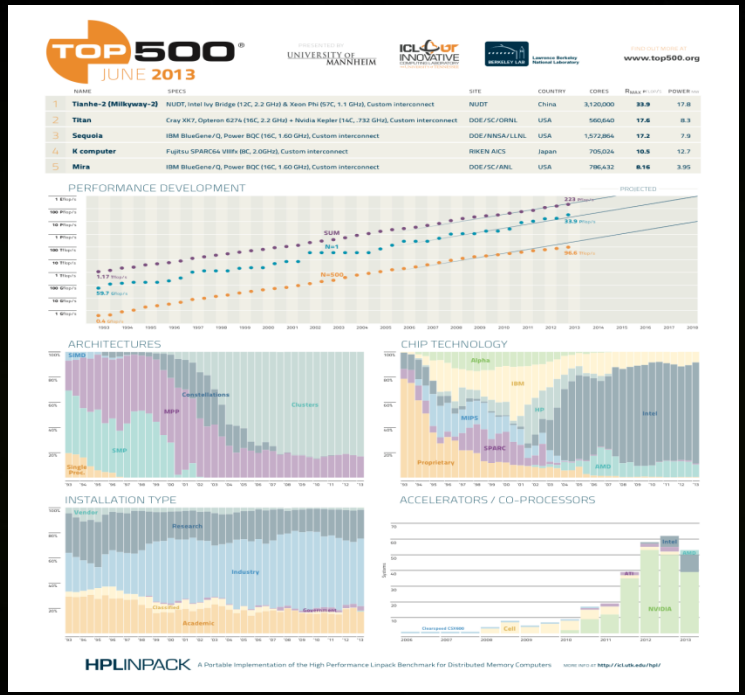
What Accelerators

- There exists different type of processor accelerators…*
  - An additional computation unit within a core – ex: vectorization unit
  - SOC “System On a Chip” with lightweight generalist cores and integrated GPU – ex: NVIDIA Tegra K1
  - Accelerator cards which can be plugged within the same server – ex: GPGPU (General Purpose Graphic Processing Unit)



Supercomputing… A Quick Look at the Web

- Top500.org
  - performance development
  - logarithmic progression! (x10 in 3years)
  - clusters, clusters (84%)!
  - 54% in industry
  - max power efficiency: 2.9 Gflops/W
  - #500: 96 TFlops! - Total : 223 Pflops
  - poster Top500
- Graph500.org
  - BlueGene++
- Green500.org
  - GPU++
  - max: 4+ Gflops/W (Immersive cooling)
  - #1 green500 = #467 top500 (1 T00flops)
  - #1 top500 = #32 green500



History of fastest Architectures 2013… The Tianhe-2 (Milky Way-2)

- Ranked 1st in the top500 list of the most “powerful” (computing intensive) computers (June 2013)
- Ranked 6th in the graph500 list of the most “powerful” (data intensive processing) computers (June 2013)
- Ranked 32nd in the green500 list of the most energy efficient computer (June 2013)
- China (National University of Defense Technology)

2013 #2… The Titan (Cray XK7)

- Ranked 2nd in the top500 list (1st in Nov. 2012)
- 299008 cores - Memory: 710 TB - Cray Gemini Interconnect
- 18688 Opteron 6274 16 cores 2.200GHz + 18,688 Nvidia Tesla K20X GPUs
- Rmax = 17590 - Rpeak = 27112 (computing efficiency : 65 %)
- Power: 8.2 MW… only!

CSCS Piz Daint among Top 10 in Top 500 and Green 500

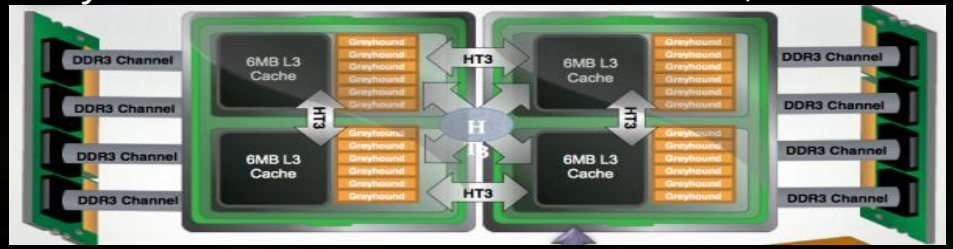
Top500 Rank	TFLOPS/s	Site
1	33,862.7	National Super Computer Centre Guangzhou
2	17,590.0	Oak Ridge National Lab #1 USA
3	17,173.2	DOE, United States
4	10,510.0	RIKEN Advanced Institute for Computational Science
5	8,586.6	Argonne National Lab
6	6,271.0	Swiss National Supercomputing Centre (CSCS) #1 Europe
7	5,168.1	University of Texas
8	5,008.9	Forschungszentrum Juelich
9	4,293.3	DOE, United States
10	3,143.5	Government

Piz Daint: 5,272 nodes with 1 x Xeon E5-2670 (SB) CPU, 1 x NVIDIA K20X GPU, Cray XC30 at 2.0 MW

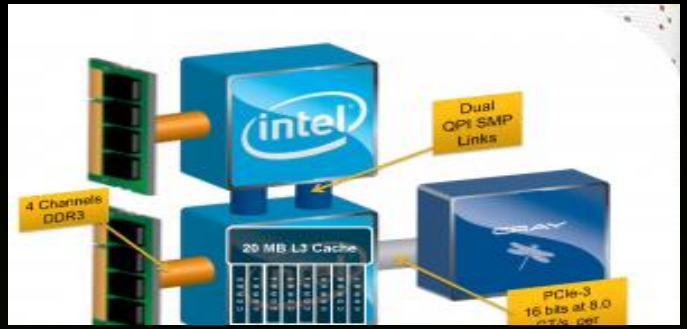
Green500 Rank	MFLOPS/W	Site
1	4,389.82	GSIC Center, Tokyo Tech
2	3,631.70	Cambridge University
3	3,517.84	University of Tsukuba
4	3,459.46	SURFsara
5	3,185.91	Swiss National Supercomputing (CSCS)
6	3,131.06	ROMEO HPC Center
7	3,019.72	CSIRO
8	2,951.95	GSIC Center, Tokyo Tech
9	2,813.14	Eni
10	2,629.10	(Financial Institution)
16	2,495.12	Mississippi State (top non-NVIDIA)
59	1,226.60	ICHEC (top X86 cluster)

Variety of node architectures

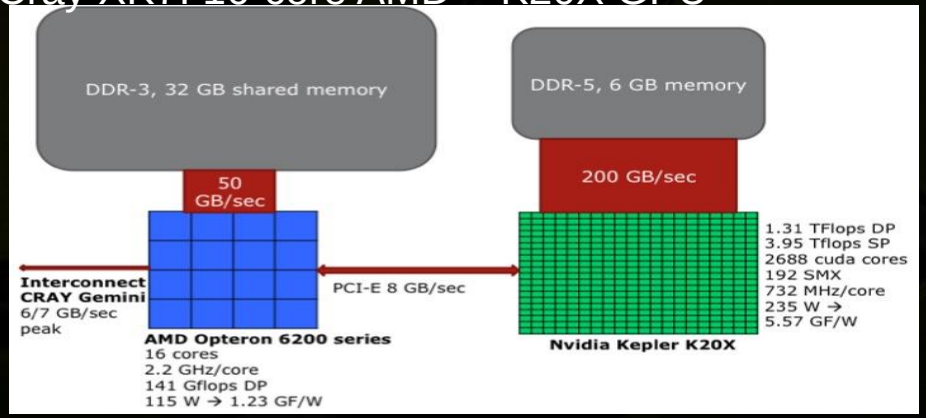
Cray XE6: dual-socket x 2-die x 6-core, 24 cores



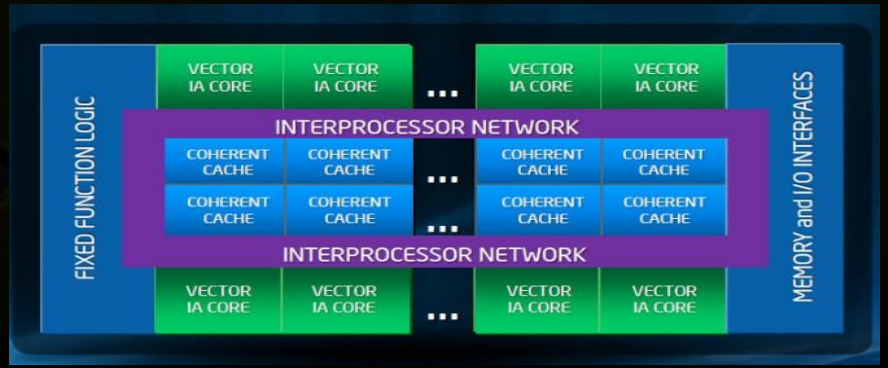
Cray XC30: dual-socket x 8-core, 16 cores



Cray XK7: 16-core AMD + K20X GPU

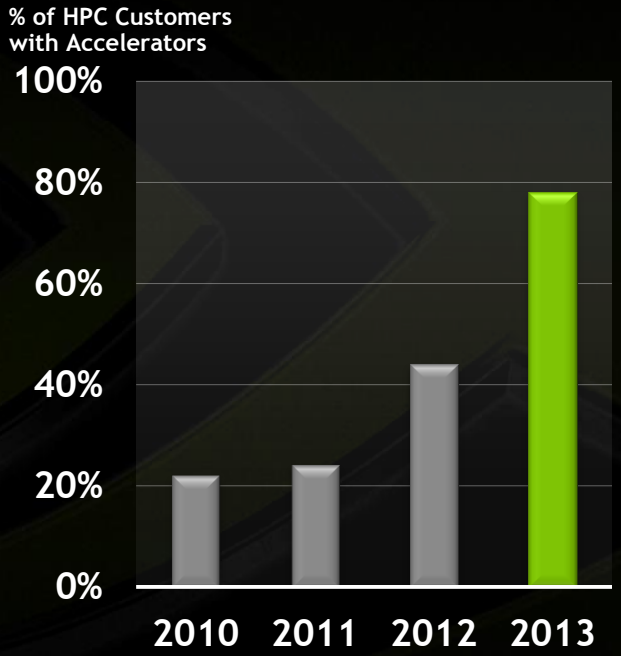


Intel MIC: 16-core host + 60+ cores co-processor



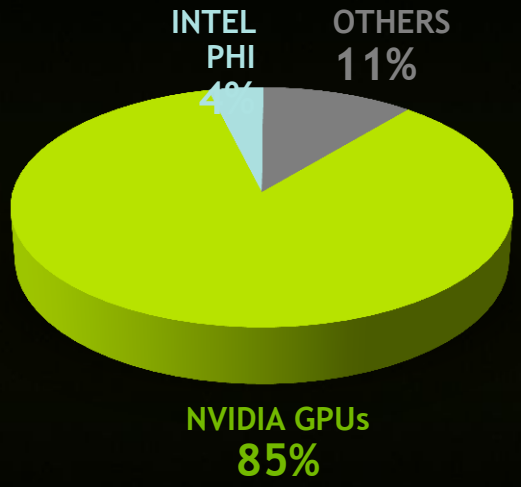
Performance History of HPC Systems

Rapid Adoption of Accelerators



Intersect360 HPC User Site Census: Systems, July 2013
IDC HPC End-User MSC Study, 2013

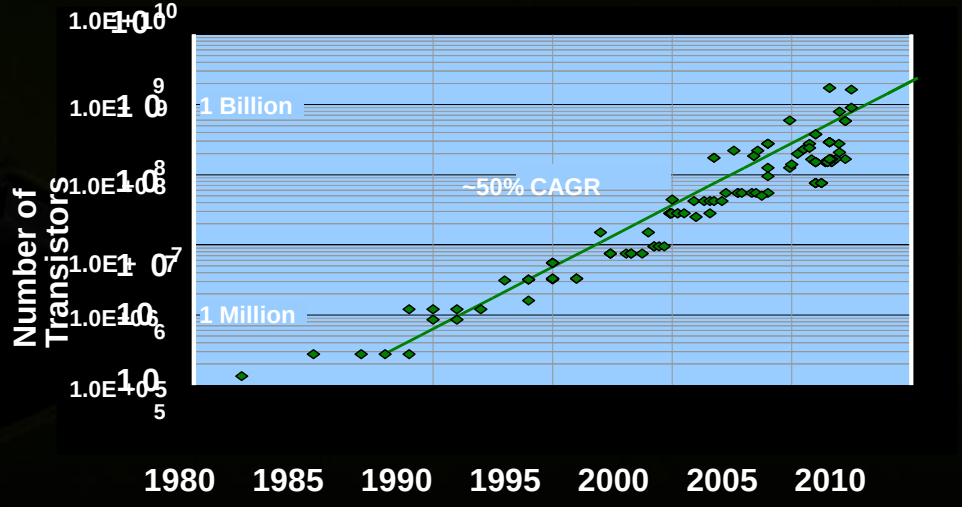
NVIDIA GPU is Accelerator of Choice



Intersect360 Research
HPC User Site Census: Systems, July 2013

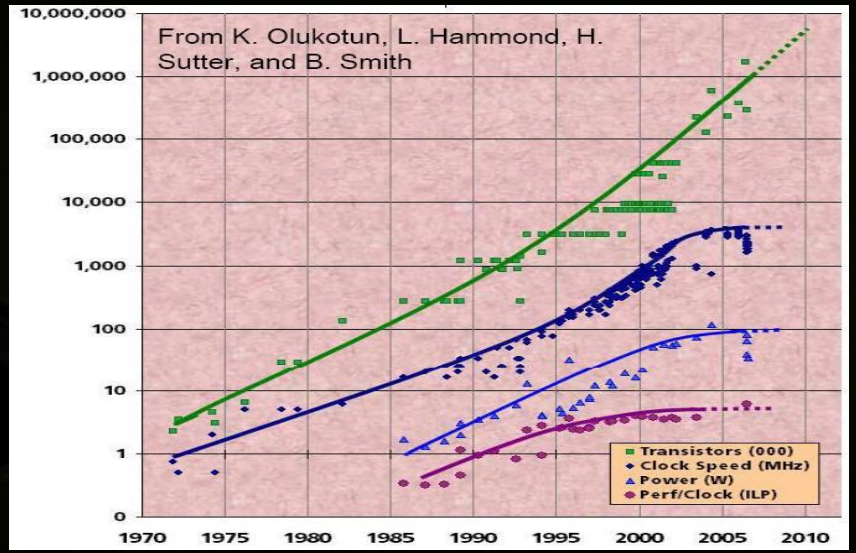
Microprocessor Transistors Trend

- Moore's Law
  - Gordon E Moore, Co-Founder of Intel
  - 1965 paper called "Cramming more components onto integrated circuits"
  - The number of transistors per chip doubles every 18 months.
- The law has been true for ~45 years!
- Thanks to manufacturing progress
  - 45 nm, 32 nm, 22 nm



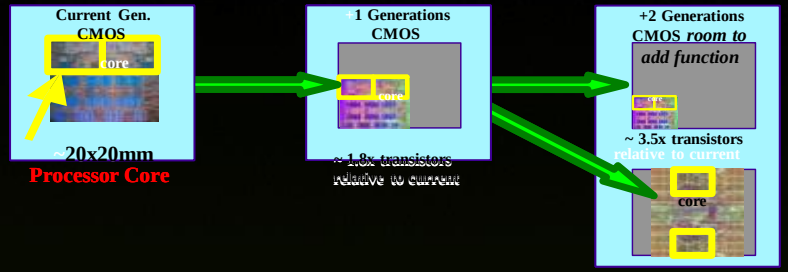
Microprocessor Clock Speed Trend

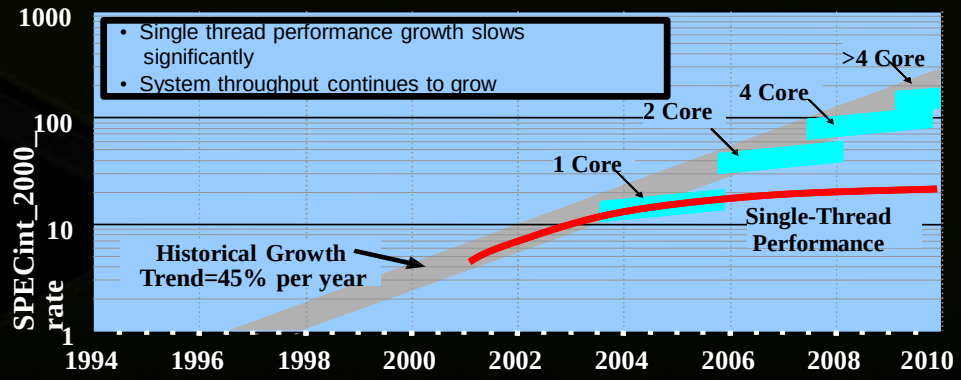
- Clock Speed and Power Consumption used to increase with the numbers of transistors and was thus associated with Moore's Law
- Clock Speed is not increasing any more because it involves a power consumption and dissipation increase that manufacturers are not able to follow.
- Power evolves like frequency^3



Microprocessor Computational Power Trend

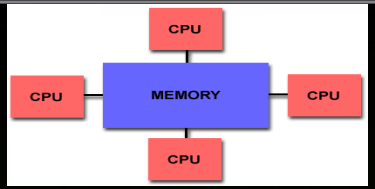
- The divergence between transistor density and frequency scaling drives toward multicore design
- Multicores trend makes the computational power of processors and systems to continue to grow.
- SpecInt rate is a standard benchmark measuring the performance of a full system – www.spec.org



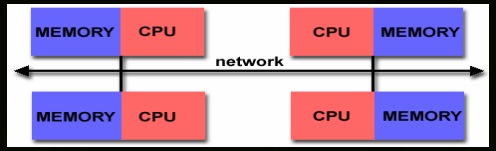


Parallel Computing

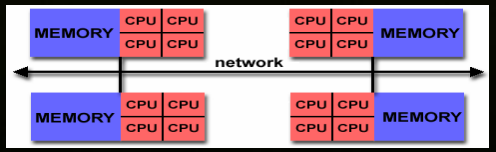
- The complex problems HPC has to deal with require capabilities beyond those of standalone computers
- Parallel Computing enables scientists and engineers to compute faster and/or to solve bigger, more realistic problems
- The parallelism can be:
  - Internal inside the processor architecture – ex: Vectorization
  - By adding more processors
    - Shared Memory: SMP
    - Distributed Memory: MPP
    - Mixed Mode: both SMP and MPP



- SMP
  - Multiple processors can operate independently but share the same memory resources



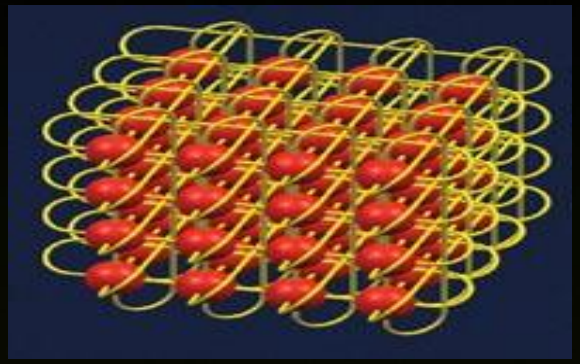
- MPP
  - Processors have their own local memory
  - Nodes are interconnected using a network, used by the processes to communicate



- Mixed
  - MPP system with SMP Nodes
  - Currently 90% of HPC Systems

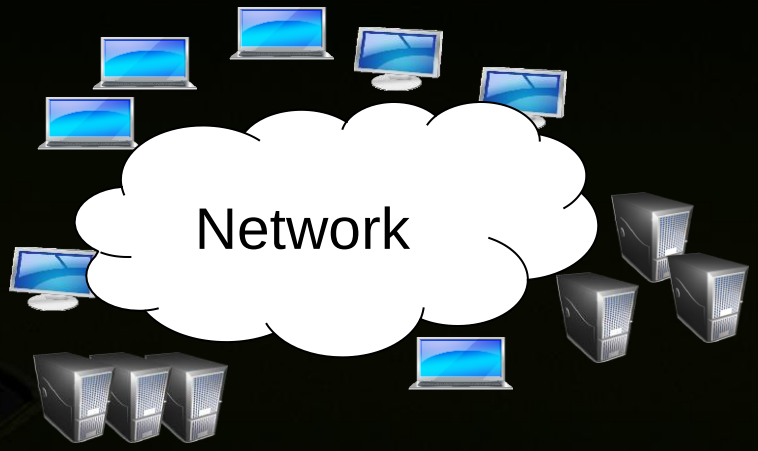
Massively parallel

- The massively parallel system is a system with many interconnected cores
- The network is designed to be very efficient for communication between process
- The idea is that each core taken individually is not very good but taken globally it provides very good performances
- University of Juelich's supercomputer is running around 300 000 cores running at 0.85 GHz
- The interconnection was a 3D Torus based on standard 10 Gigabit Ethernet
- The performances is around 800 TFlops



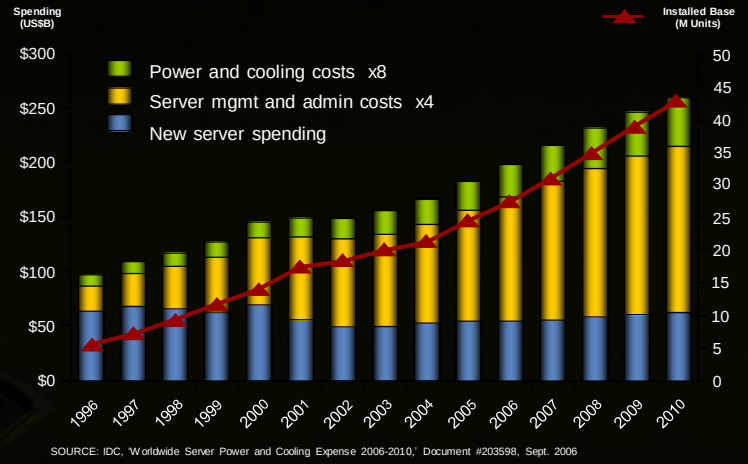
Grids

- The grids consist of the interconnection of many computers
- This is an heterogeneous cluster: the computers and networks configurations differ from one to an other
- To be able to use the grid as a mean for computation, the problem must be split in many small problem that can be solved independently
- The servers then collect the results to build the global solution
- Example of project using grids are seti@home and folding@home
- Seti@home annouce a performance of 364.8 TFlops. The average is 40 TFlops



Power Consumption in the Data Center

- Power and Cooling costs are increasing faster than acquisition costs
  - Today, each dollar of new servers cost more than $0.50 to power and cool
- The current Data Centers are not able to sustain the electrical power needs of big HPC systems
- Data Center Power Consumption limits for the size of the HPC systems
- If we scale the current technology, each next generation exascale computer will require ~125MW of Total Power
  - For information, this is roughly equivalent to 120 Wind Turbines or 10% of a nuclear reactor



SOURCE: IDC, 'Worldwide Server Power and Cooling Expense 2006-2010,' Document #203598, Sept. 2006

Performance vs Power Consumption

- The ratio to optimize for HPC systems is the *Flops/Watt ratio*
  - Power Consumption issues are currently addressed by the fastest HPC systems either by:
    - Using a large number of low power processors with limited single thread performance - Massively Parallel Computers
    - Using high single thread performance processors with heterogeneous cores or accelerators
  - With both solutions, *the Flops/Watt ratio is improved*

Power is *the* Problem

=

Jaguar (Nov. ‘11)
2.3 petaflops @ 7 megawatts

7,000 Homes = 7 megawatts
Small city

Power is *the* Problem

120 petaflops | 376 megawatts

Enough to power all of San Francisco

Concepts and Terminology

Flynn's Taxonomy of Parallel Arch.

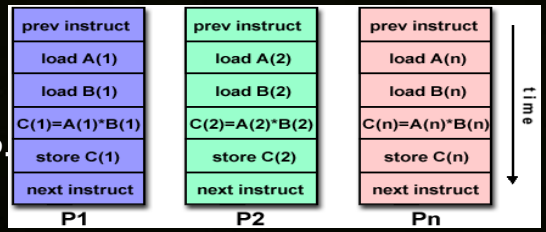
- Distinguishes parallel architecture by instruction and data streams
  - SISD: classical uniprocessor architecture

S I S D Single Instruction, Single Data	**S I M D** Single Instruction, Multiple Data
M I S D Multiple Instruction, Single Data	**M I M D** Multiple Instruction, Multiple Data

(Introduction to Parallel Computing, Blaise Barney)

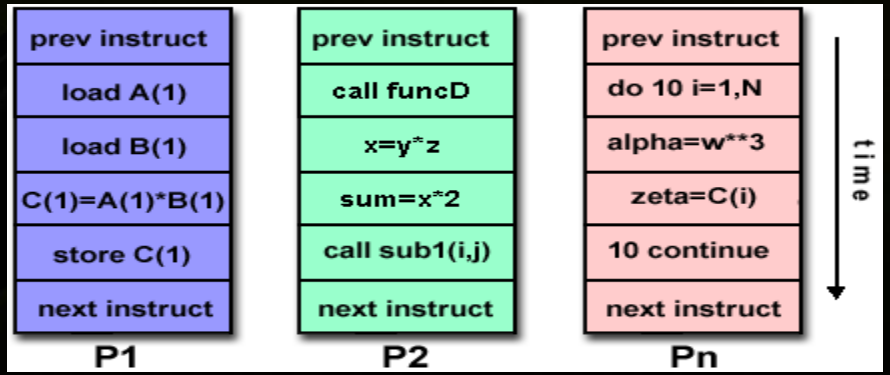
Single Instruction, Multiple Data (SIMD)

- A type of parallel computer
- Single instruction: All processing units execute the same instruction at any given clock cycle
- Multiple data: Each processing unit can operate on a different data element
- This type of machine typically has an instruction dispatcher, a very high-bandwidth internal network, and a very large array of very small-capacity instruction units.
- Best suited for specialized problems characterized by a high degree of regularity,such as image processing.
- Synchronous (lockstep) and deterministic execution
- Two varieties: Processor Arrays and Vector Pipelines
- Examples:
  - Processor Arrays: Connection Machine CM-2, Maspar MP-
  - Vector Pipelines: IBM 9000, Cray C90, Fujitsu VP, NEC SX-2, Hitachi S820



Multiple Instruction, Multiple Data (MIMD)

- Currently, the most common type of parallel computer. Most modern computers fall into this category.
- Multiple Instruction: every processor may be executing a different instruction stream
- Multiple Data: every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Examples: most current supercomputers, networked parallel computer "grids" and multi-processor SMP computers - including some types of PCs.



Parallel Control Mechanisms

Name	Meaning	Examples
Single Instruction, Multiple Data (SIMD)	A single thread of control, same computation applied across "vector" elts	Array notation as in Fortran 90: `A[1:n] = A[1:n] + B[1:n]`
Multiple Instruction, Multiple Data (MIMD)	Multiple threads of control, processors periodically synch	Parallel loop: `forall (i=0; i<n; i++)`

Single Program, Multiple Data (SPMD)	Multiple threads of control, but each processor executes same code	Processor-specific code:
		`if ($myid == 0) {`
		`}`

(Prof. Mary Hall, Univ. of Utah)

Some General Parallel Terminology

Like everything else, parallel computing has its own "jargon". Some of the more commonly used terms associated with parallel computing are listed below. Most of these will be discussed in more detail later.

- Task
  - A logically discrete section of computational work. A task is typically a program or program-like set of instructions that is executed by a processor.
- Parallel Task
  - A task that can be executed by multiple processors safely (yields correct results)
- Serial Execution
  - Execution of a program sequentially, one statement at a time. In the simplest sense, this is what happens on a one processor machine. However, virtually all parallel tasks will have sections of a parallel program that must be executed serially.

- Parallel Execution
  - Execution of a program by more than one task, with each task being able to execute the same or different statement at the same moment in time.
- Shared Memory
  - From a strictly hardware point of view, describes a computer architecture where all processors have direct (usually bus based) access to common physical memory. In a programming sense, it describes a model where parallel tasks all have the same "picture" of memory and can directly address and access the same logical memory locations regardless of where the physical memory actually exists.
- Distributed Memory
  - In hardware, refers to network based memory access for physical memory that is not common. As a programming model, tasks can only logically "see" local machine memory and must use communications to access memory on other machines where other tasks are executing.

- Communications
  - Parallel tasks typically need to exchange data. There are several ways this can be accomplished, such as through a shared memory bus or over a network, however the actual event of data exchange is commonly referred to as communications regardless of the method employed.
- Synchronization
  - The coordination of parallel tasks in real time, very often associated with communications. Often implemented by establishing a synchronization point within an application where a task may not proceed further until another task(s) reaches the same or logically equivalent point.
  - Synchronization usually involves waiting by at least one task, and can therefore cause a parallel application's wall clock execution time to increase.

- Granularity
  - In parallel computing, granularity is a qualitative measure of the ratio of computation to communication.
  - *Coarse:* relatively large amounts of computational work are done between communication events
  - *Fine:* relatively small amounts of computational work are done between communication events
- Observed Speedup
  - Observed speedup of a code which has been parallelized, defined as:

$$\frac{\text{wall-clock time of serial execution}}{\text{wall-clock time of parallel execution}}$$

  - One of the simplest and most widely used indicators for a parallel program's performance.

- Parallel Overhead
  - The amount of time required to coordinate parallel tasks, as opposed to doing useful work. Parallel overhead can include factors such as:
    - Task start-up time
    - Synchronizations
    - Data communications
    - Software overhead imposed by parallel compilers, libraries, tools, operating system, etc.
    - Task termination time
- Massively Parallel
  - Refers to the hardware that comprises a given parallel system - having many processors. The meaning of many keeps increasing, but currently BG/L pushes this number to 6 digits.

- Scalability
  - Refers to a parallel system's (hardware and/or software) ability to demonstrate a proportionate increase in parallel speedup with the addition of more processors. Factors that contribute to scalability include:
    - Hardware - particularly memory-cpu bandwidths and network communications
    - Application algorithm
    - Parallel overhead related
    - Characteristics of your specific application and coding

Parallel machines & programming model
(hardware & software)

Parallelism within single processor – pipelining

- Like assembly line in manufacturing
- Instruction pipeline allows overlapping execution of multiple instructions with the same circuitry

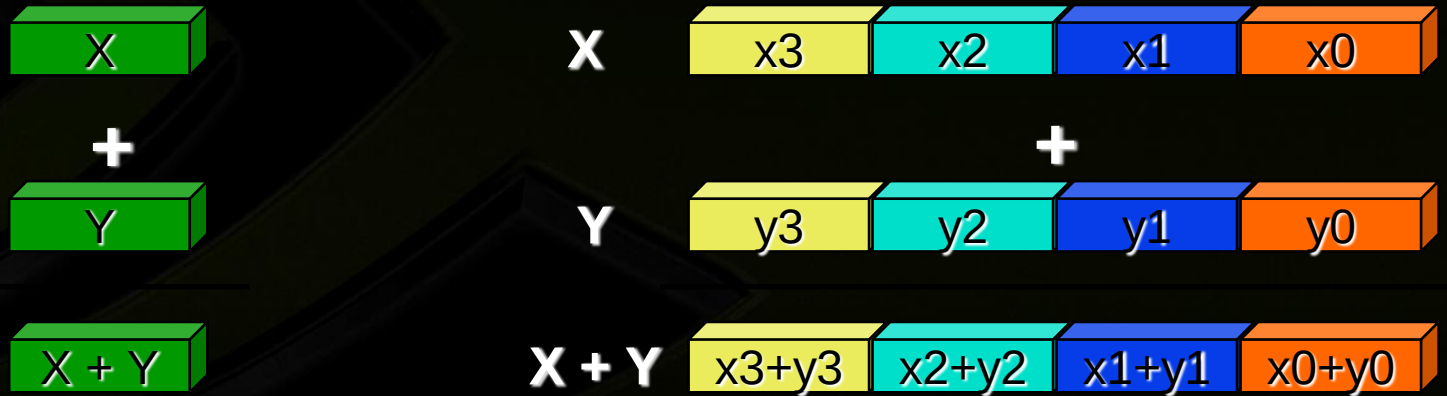
Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

IF = Instruction Fetch
ID = Instruction Decode
EX = Execute
MEM = Memory access
WB = Register write back

  - Sequential execution: 5 (cycles) * 5 (inst.) = 25 cycles
  - Pipelined execution: 5 (cycles to fill the pipe, latency) + 5 (cycles, 1 cycle/inst. throughput) = 10 cycles
- Arithmetic unit pipeline: A FP multiply may have latency 10 cycles, but throughput of 1/cycle
- Pipeline helps throughput/bandwidth, but not latency

Parallelism within single processor – SIMD

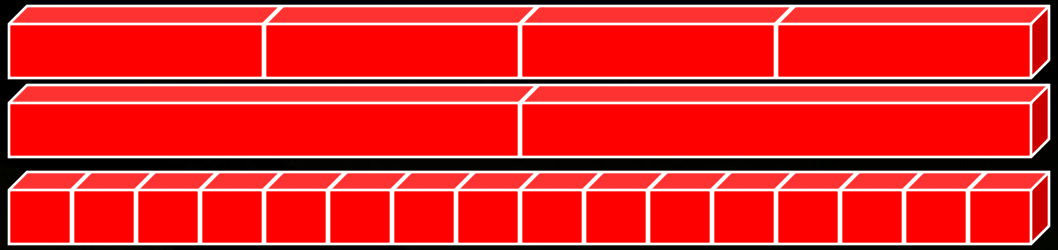
- **SIMD: Single Instruction, Multiple Data**
  - Scalar processing
    - traditional mode
    - one operation produces one result
  - SIMD processing
    - with SSE / SSE2
    - SSE = streaming SIMD extensions
    - one operation produces multiple results



Slide Source: Alex Klimovitski & Dean Macri, Intel Corporation

SSE / SSE2 SIMD on Intel

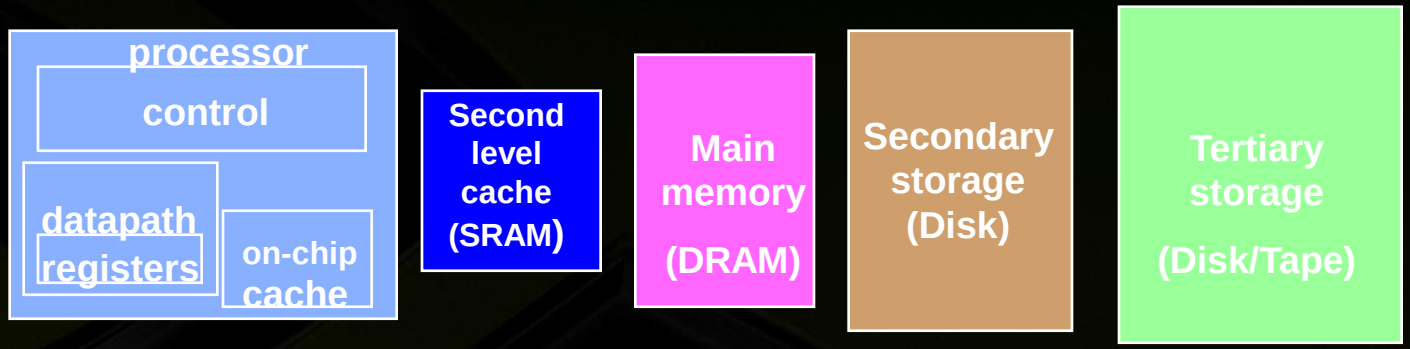
- SSE2 data types: anything that fits into 16 bytes, e.g.,



- Instructions perform add, multiply etc. on all the data in this 16-byte register in parallel
- Challenges:
  - Need to be contiguous in memory and aligned
  - Some instructions to move data around from one part of register to another
- Similar on GPUs, vector processors (but many more simultaneous operations)

Memory Hierarchy … Flops is not everything

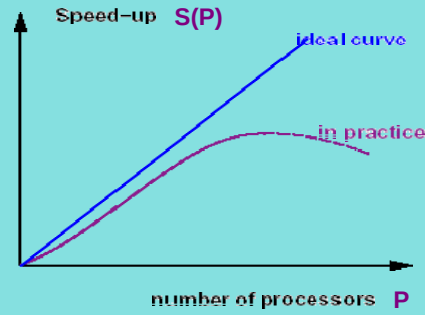
- Most programs have a high degree of locality in their accesses
  - **spatial locality: accessing things nearby previous accesses**
  - **temporal locality: reusing an item that was previously accessed**
- Memory hierarchy tries to exploit locality to improve average



Speed	1ns	10ns	100ns	10ms	10sec
Size	KB	MB	GB	TB	PB

Amdahl's Law – Limitations to Parallelism

Are the HPC applications able to use a high number of processors?



Speedup

Theoritical Maximum
S(P)=P

We can decompose the execution of a program on a parallel computer into:

$T(1) = T_p + T_s$ *and* $T(P) = T_p/P + T_s$

Where
P is the number of processors

T_p is the part of the program which can be made parallel

T_s is the part of the program which cannot be made parallel

We measure scalability of an application with the speedup:

$Speed\ Up : S(P) = T(1) / T(P) = P / (f_p + f_s.P)$

For example: f_p=0.8 and f_s=0.2 leads to $S(P) < 1/T_s < 5$

Amdahl's Law defines that the non-parallelizable fraction of a program limits performance and poses an upper limit on the scalability

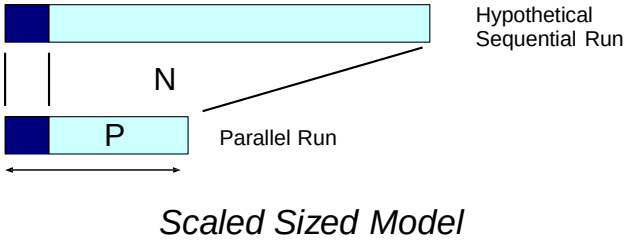
Gustafson's Law – Re-evaluating Amdahl's Law

- One of Amdahl's law hypothesis is that the problem size remains constant with the number of processors.
- Gustafson refuted this assumption and reevaluated Amdahl's law stating that:
  - when given a more powerful system, the problem generally expands to make use of the increased facilities.
  - the fraction of the program which can me made parallel increases with the problem size.

Speed Up : $S(P) = T(1) / T(P) = P / (f_p(N) + f_s(N).P)$

Where
P is the number of processors
N is the problem size
fp is the fraction of the program which can be made parallel
fs is the fraction of the program which cannot be made parallel

- $f_s(N)$ diminishes with the problem size then speedup approaches P as N approaches infinity



Scaled Sized Model

- Gustafson's Law states that any sufficiently large problem can be efficiently parallelized.

Summary

- Latest innovations have enable us to build reliable and programmable Petascale systems
- But new architectures will be needed to reach the ExaFlops barrier
  - The main challenges next HPC architectures will have to deal with are:
    - Power
    - Resiliency
    - Memory Capacity
    - Efficiency
    - Programmability