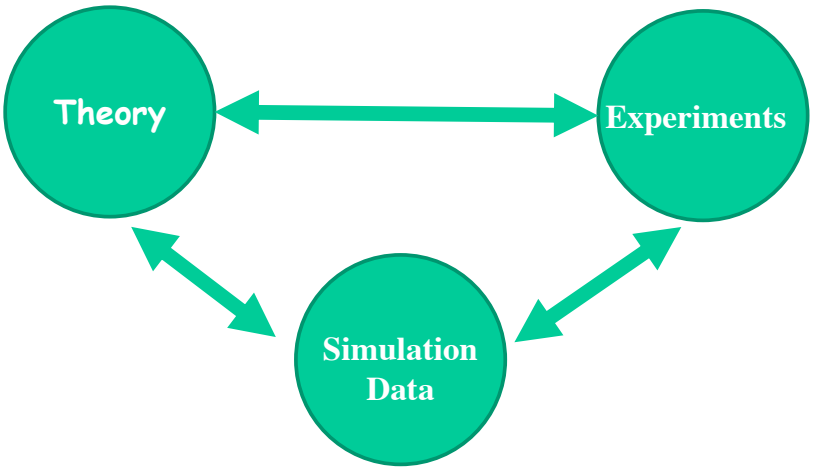
High Performance Computing in Numerical Simulations

L. Giraud - Inria

Computational sciences in science discovery

- *Numerical simulation (and data): third pillar for the development of scientific discovery at the same level as theory and experimentation.*



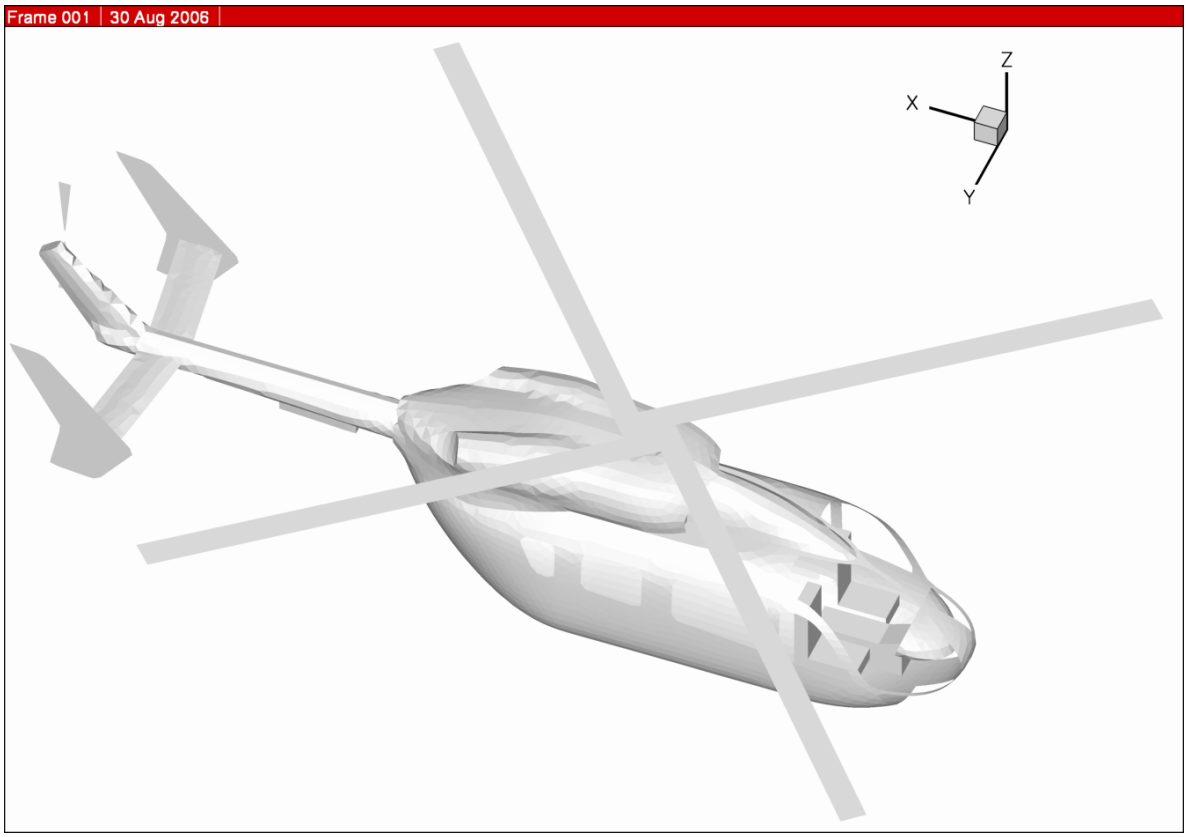
Outline intro HPC

- Motivations
- Basic concepts
- Memory limitations
- Architecture overview
  - taxonomy
  - memory point of view
- Prospective and trends

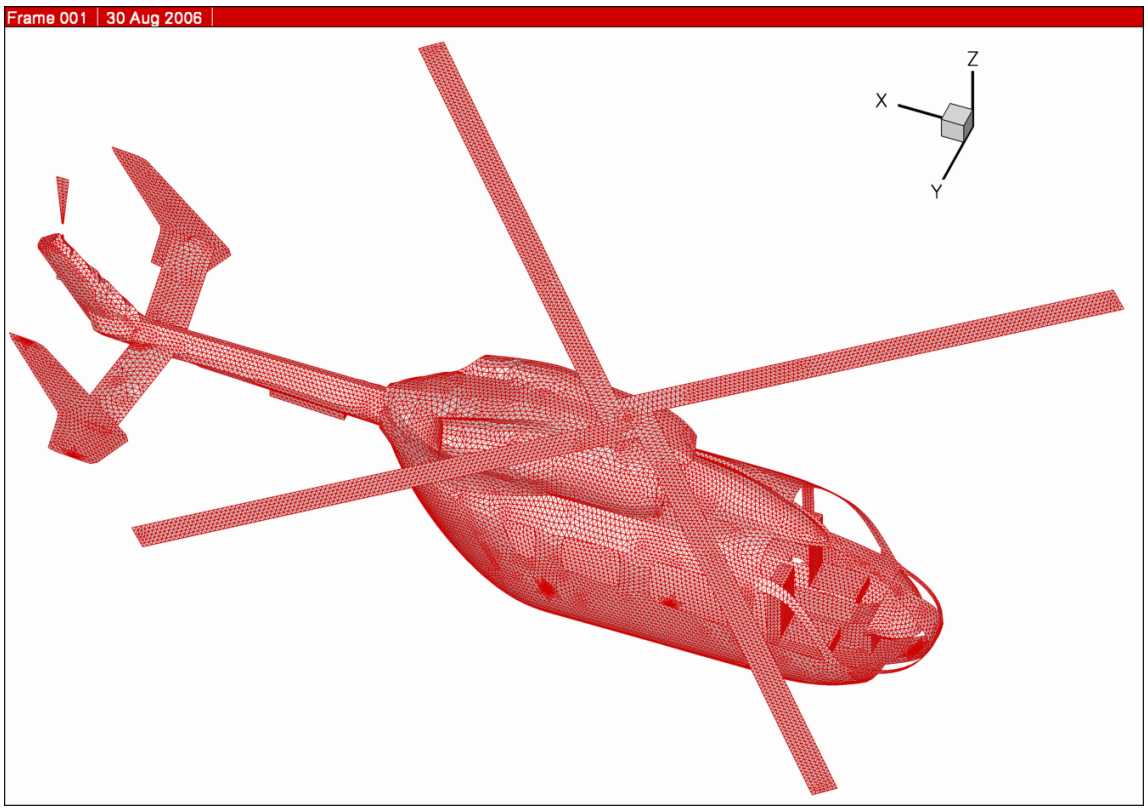
Main steps in simulation design

- Modeling: physics/biology/..-applied math.
  - Systems of equations (ODE, PDE, …)
- Analysis: applied math
  - Existence/uniqueness in ad-hoc spaces
  - Approximation/discretization
  - Solution techniques and their properties (stability, convergence speed, …)
- Software : computational science
  - Computer selection
  - Algorithm selection
  - Library selection
  - Code tuning

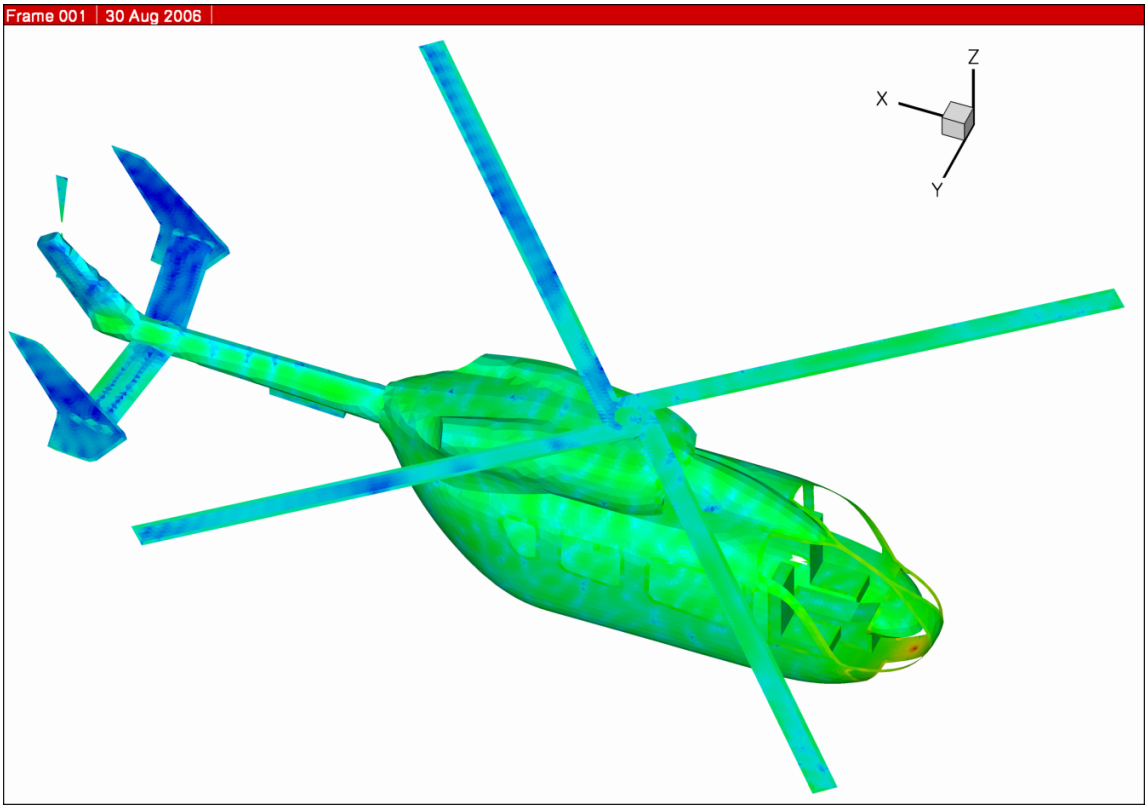
Courtesy Airbus Group Innovation



Courtesy Airbus Group Innovation

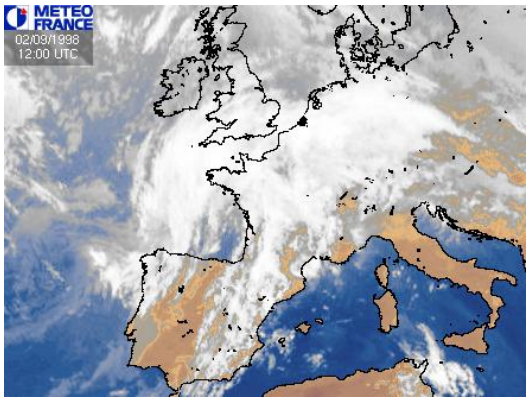


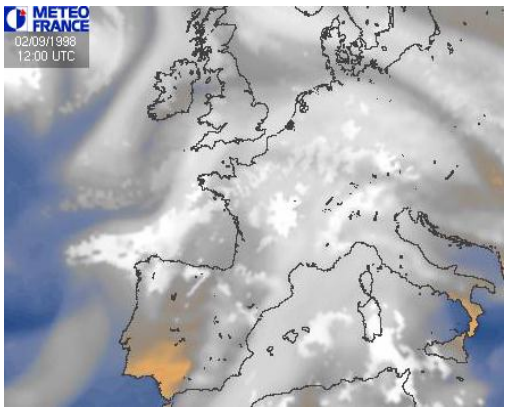
Courtesy Airbus Group Innovation



Motivations some examples (I)

- Time constraints : weather forecasting

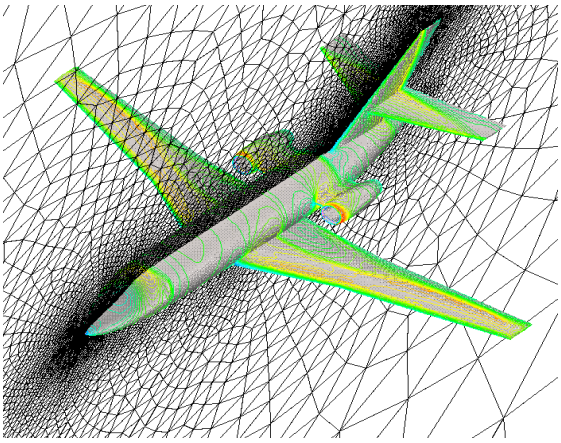




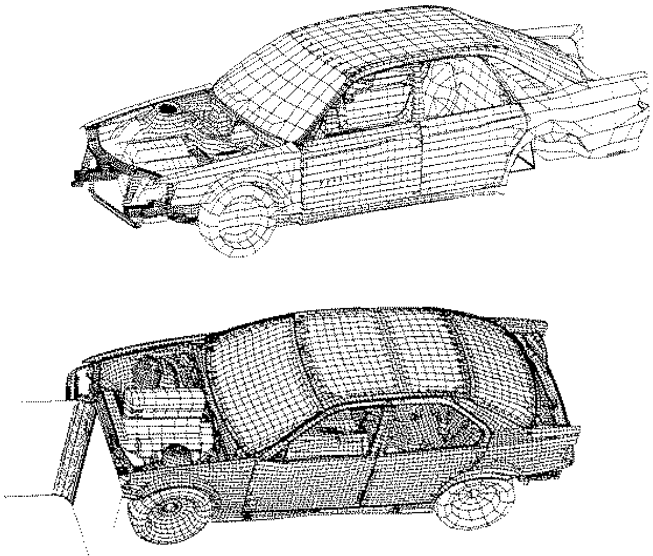
Meteo France pictures

Motivations some examples (II)

- Cost constraints : wind tunnel, crash simulation, ...



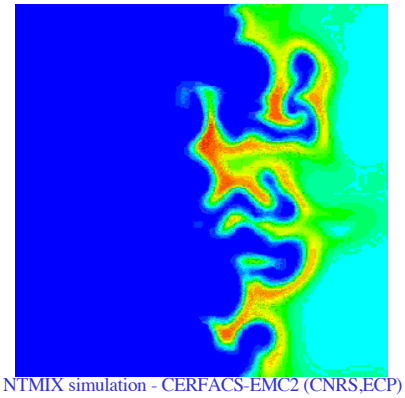
AVBP mesh - CERFACS



PAM-CRASH meshes - ESI

Motivations some examples (III)

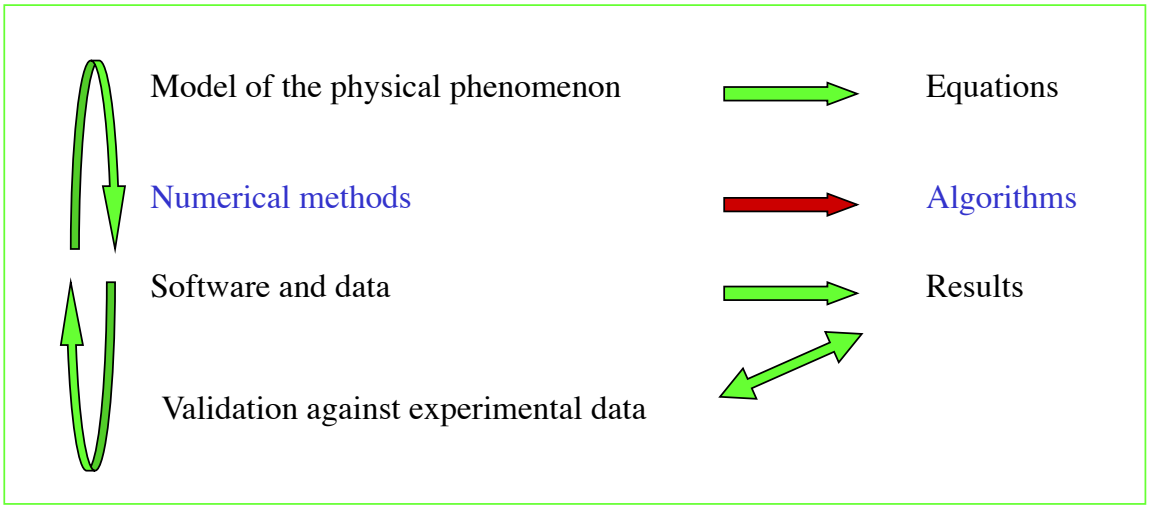
- Scale constraints
  - large scale : climate modelling, pollution, astrophysics
  - tiny scale : combustion, quantum chemistry



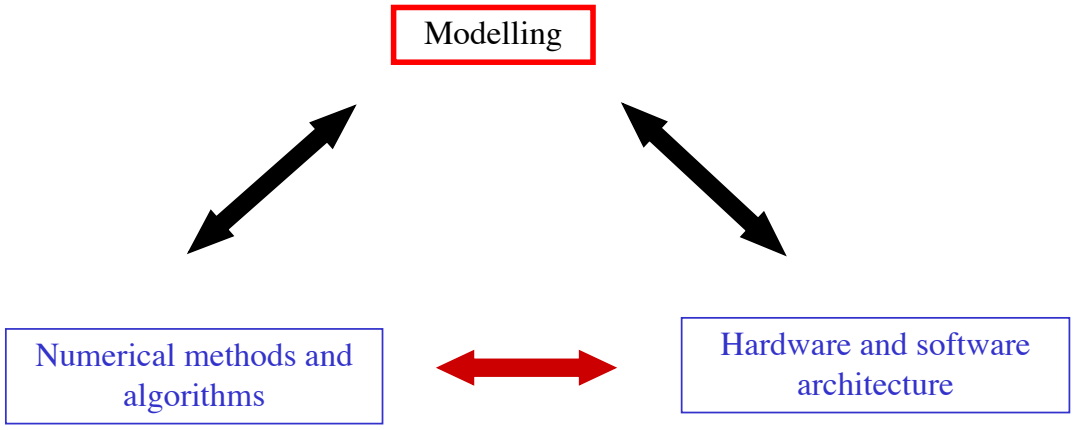
NTMIX simulation - CERFACS-EMC2 (CNRS,ECP) -IFP

Motivations

Predict the behaviour of complex systems through numerical simulations

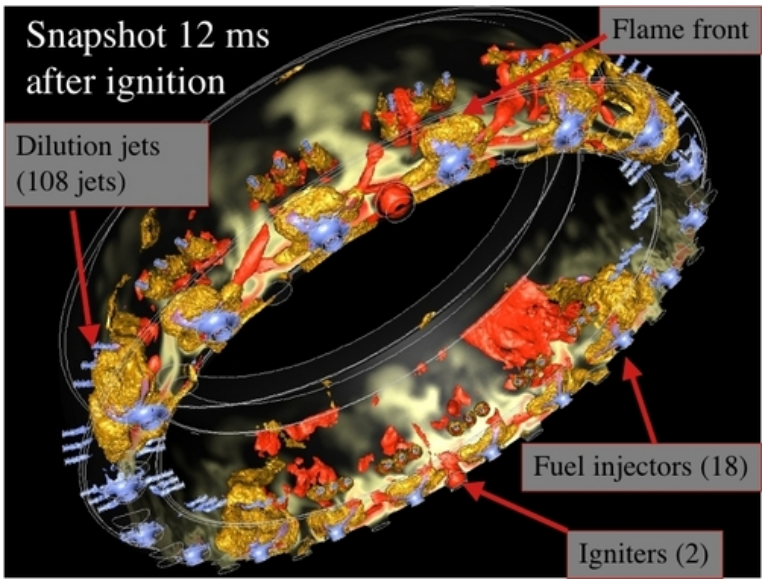


Main components involved in the design of simulation software



Motivations: interdisciplinary component

- Flame ignition



Courtesy of CERFACS - IBM - TURBOMECA (Safran group) and CINES

First Historical Classification

- Flynn's taxonomy based on two features
  - Flow of instructions : 1 or multiple
  - Flow of data operated upon : 1 or multiple

SISD : Single Instruction Single Data

SIMD : Single Instruction Multiple Data

MIMD : Multiple Instruction Multiple Data

Basic Architectural components

- CPU
  symbol : 

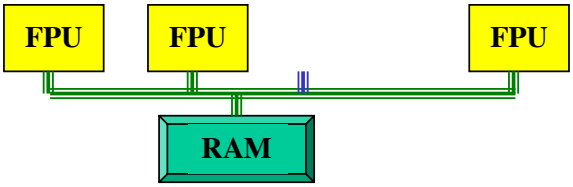
  Flops : floating point operations per second
  (Mflops = 10^6, Gflops = 10^9, Tflops = 10^{12},
  Pflops= 10^{15} , Exaflops = 10^{18})
- Memory
  symbol : 

  Mbytes, Gbytes, Tbytes
- Interconnection path
  symbol :
  Mbits :

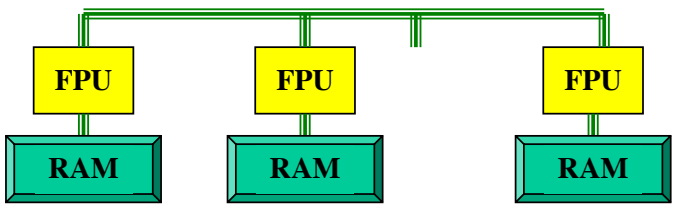
Architecture overview (Hardware)

- Workstation - PC

- Symmetric multiprocessor (SMP)

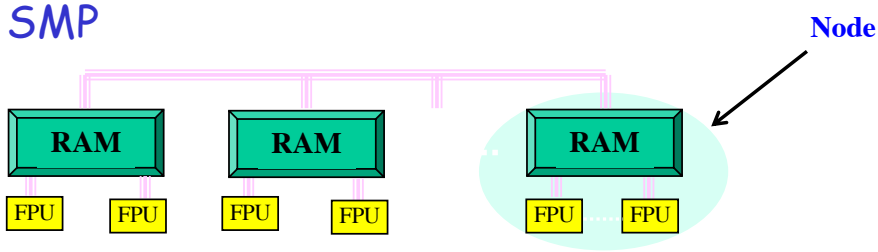


- NUMA, cluster



Architecture overview (Hardware II)

- Cluster of SMP

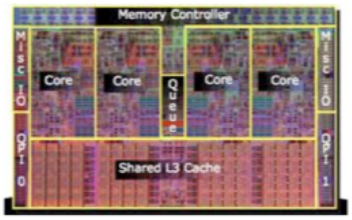


- Heterogeneous many-core nodes

  The computing units are no longer just FPU but a socket itself compose of core and accelerators (GPGPU, FPGA, …).

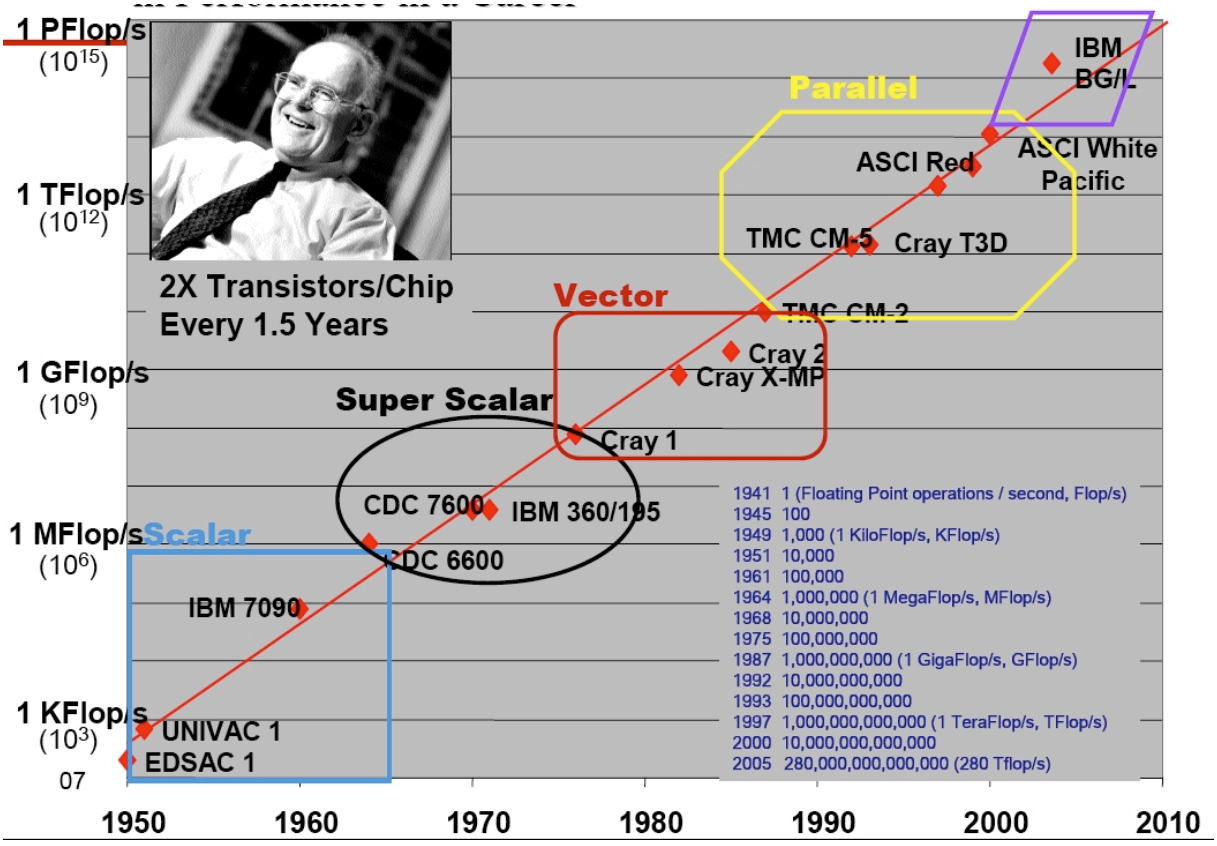
Architecture overview (Hardware)

- Workstation - PC



- Symmetric multiprocessor (SMP)

- NUMA, cluster



Fugaku – Riken - Japan

Top 6 - November 2020 HPL (High Performance Linpack)

4 TGV at full power

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**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	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	79,215.0	2,646
6	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482

France:
- **Total : 18**
- **CEA : 24**
- **Météo-F : 30-34**

Europe:
- **Germany : 7**
- **Italy: 8-11**
- **Switzerland: 12**

Industry
- **Eni S.p.A.: 8**
- **Saudi Aramco: 10**
- **Total: 18**

Top 6 - November 2020 HPCG

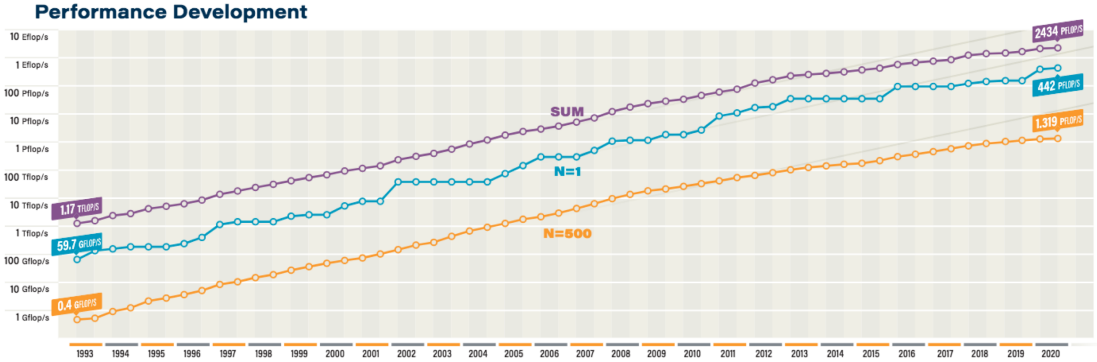
Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	HPCG (TFlop/s)	
1	1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	6004.50	**27 x decrease (3.6% peak)**
2	2	**Summit** - IBM Power System AC922, 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	148,600.0	2925.75	**50 x decrease (2% peak)**
3	3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	1795.67	**52 x decrease**
4	5	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	1622.51	
5	7	**JUWELS Booster Module** - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR InfiniBand/ParTec ParaStation ClusterSuite, Atos Forschungszentrum Juelich (FZJ) Germany	449,280	44,120.0	1275.36	
6	10	**Dammam-7** - Cray CS-Storm, Xeon Gold 6248 20C 2.5GHz, NVIDIA Tesla V100 SXM2, InfiniBand HDR 100, HPE Saudi Aramco Saudi Arabia	672,520	22,400.0	881.40	

Green 6 - November 2020

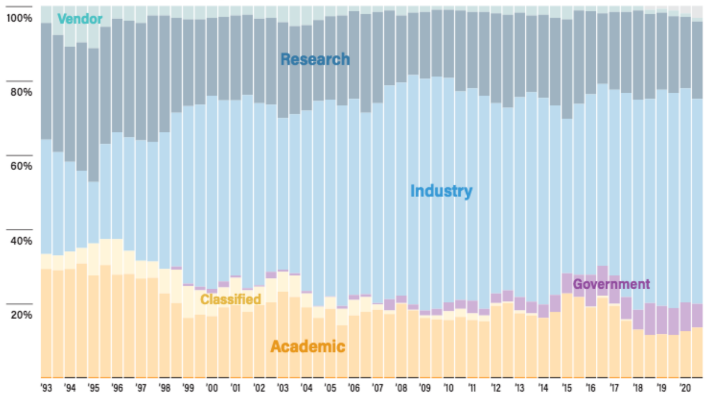
Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	Power (kW)	Power Efficiency (GFlops/watts)
1	170	**NVIDIA DGX SuperPOD** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	19,840	2,356.0	90	26.195
2	330	**MN-3** - MN-Core Server, Xeon Platinum 8260M 24C 2.4GHz, Preferred Networks MN-Core, MN-Core DirectConnect, Preferred Networks Preferred Networks Japan	1,664	1,652.9	65	26.039
3	7	**JUWELS Booster Module** - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR InfiniBand/ParTec ParaStation ClusterSuite, Atos Forschungszentrum Juelich (FZJ) Germany	449,280	44,120.0	1,764	25.008
4	146	**Spartan2** - Bull Sequana XH2000 , AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR Infiniband, Atos Atos France	23,040	2,566.0	106	24.262
5	5	**Selene** - NVIDIA DGX A100, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	555,520	63,460.0	2,646	23.983
6	239	**A64FX prototype** - Fujitsu A64FX, Fujitsu A64FX 48C 2GHz, Tofu interconnect D, Fujitsu Fujitsu Numazu Plant Japan	36,864	1,999.5	118	16.876

NOVEMBER 2020	SYSTEM	SPECS	SITE	COUNTRY	CORES	RMAX PFLOP/S	POWER MW
1	Fugaku	Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D	RIKEN R-CCS	Japan	7,630,848	442.0	29.9
2	Summit	IBM POWER9 (22C, 3.07GHz), NVIDIA Volta GV100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/SC/ORNL	USA	2,414,592	148.6	10.1
3	Sierra	IBM POWER9 (22C, 3.1GHz), NVIDIA Tesla V100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/NNSA/LLNL	USA	1,572,480	94.6	7.44
4	Sunway TaihuLight	Shenwei SW26010 (260C, 1.45 GHz) Custom Interconnect	NSCC in Wuxi	China	10,649,600	93.0	15.4
5	Selene	NVIDIA DGX A100, AMD EPYC 7742 (64C, 2.25GHz), NVIDIA A100, Mellanox HDR Infiniband	NVIDIA Corporation	USA	555,520	63.4	2.65

Performance Development

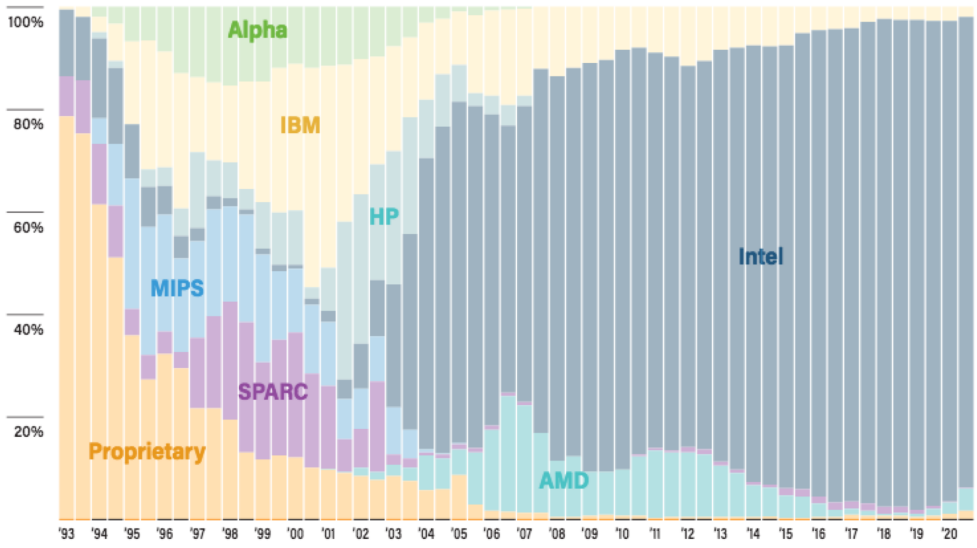


Installation Type

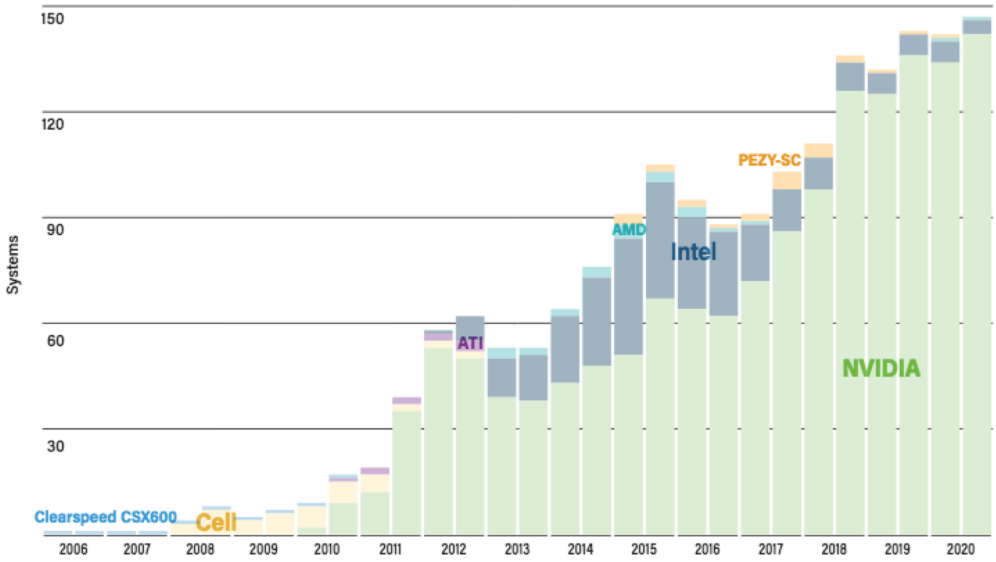


	Segments	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
1	Industry	274	54.8	616,818,610	1,238,357,136	21,167,452
2	Research	103	20.6	1,332,775,286	1,827,127,915	40,973,694
3	Academic	66	13.2	278,709,108	431,108,787	6,453,148
4	Government	34	6.8	81,008,598	133,850,181	2,100,636
5	Others	14	2.8	29,042,270	101,509,615	671,252
6	Vendor	9	1.8	90,407,980	114,082,243	1,099,944

Chip Technology



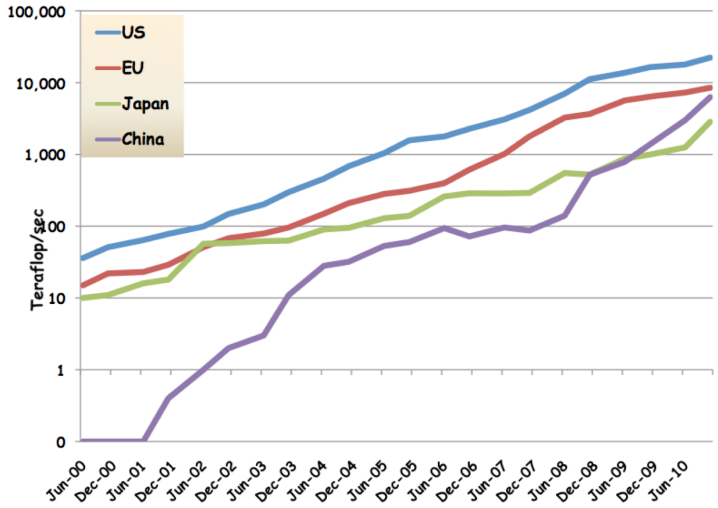
Accelerators/Co-processors



Installations by countries

	Countries	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
1	China	214	42.8	566,635,422	1,231,757,033	30,704,256
2	United States	113	22.6	668,704,300	942,339,598	15,373,432
3	Japan	34	6.8	593,700,080	766,195,745	10,947,524
4	France	18	3.6	89,828,330	135,469,318	2,669,472
5	Germany	17	3.4	131,048,770	197,689,472	2,664,446
6	Netherlands	15	3	24,736,650	31,795,200	864,000
7	Ireland	14	2.8	23,087,540	29,675,520	806,400
8	United Kingdom	12	2.4	34,067,502	44,283,532	1,248,840
9	Canada	12	2.4	26,698,060	47,707,321	716,096
10	Italy	6	1.2	78,529,000	114,511,528	1,447,536
11	Saudi Arabia	5	1	35,997,040	76,126,574	1,084,020
12	Brazil	4	0.8	10,991,000	19,270,566	214,040
13	Singapore	4	0.8	6,596,440	8,478,720	230,400
14	South Korea	3	0.6	18,720,660	31,496,620	709,220
15	Taiwan	3	0.6	12,622,710	21,651,750	247,952

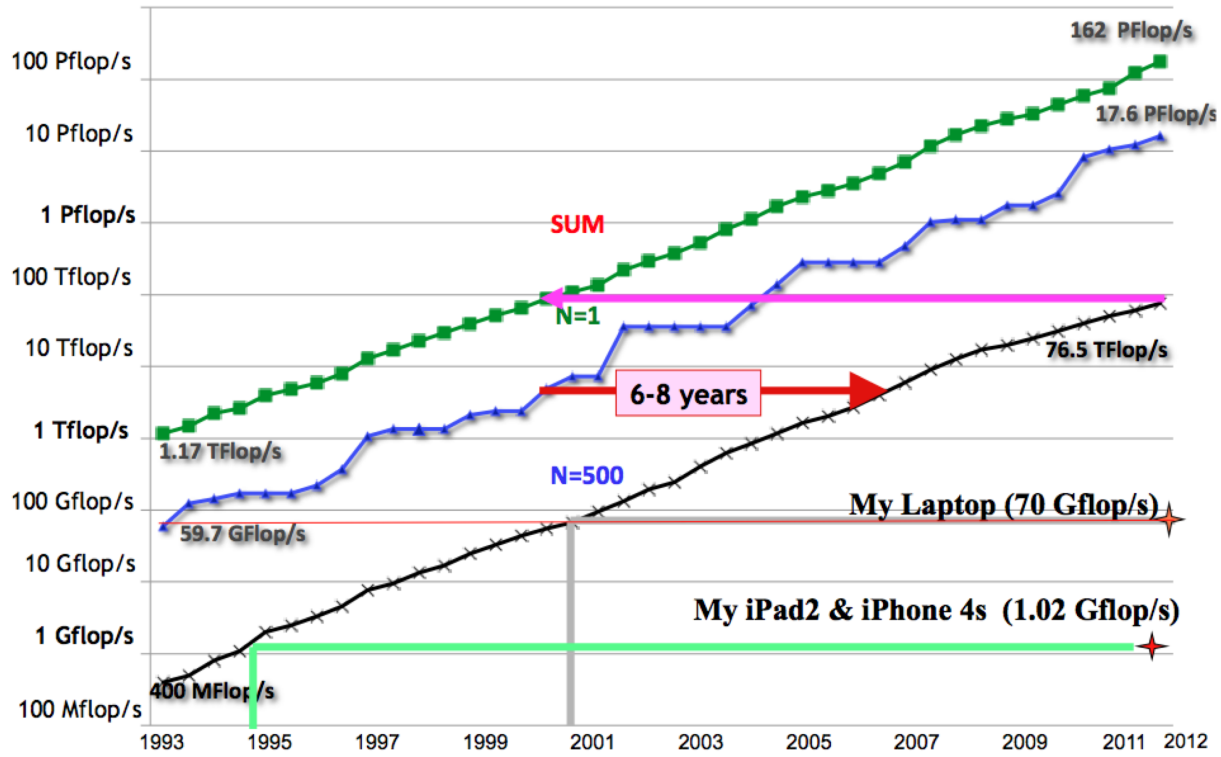
Geographical distribution



Potential System Architecture with a cap of $200M and 20MW

Systems	2012 Titan Computer	2022	Difference Today & 2022
System peak	27 Pflop/s	1 Eflop/s	O(100)
Power	8.3 MW (2 Gflops/W)	~20 MW (50 Gflops/W)	O(10)
System memory	710 TB (38*18688)	32 - 64 PB	O(100)
Node performance	1,452 GF/s (1311+141)	1.2 or 15TF/s	O(10)
Node memory BW	232 GB/s (52+180)	2 - 4TB/s	O(10)
Node concurrency	16 cores CPU 2688 CUDA cores	O(1k) or 10k	O(100) - O(10)
Total Node Interconnect BW	8 GB/s	200-400GB/s	O(100)
System size (nodes)	18,688	O(100,000) or O(1M)	O(10) - O(100)
Total concurrency	50 M	O(billion)	O(100)
MTTF	?? unknown	O(<1 day)	O(?)

Performance Development of HPC Over the Last 20 Years



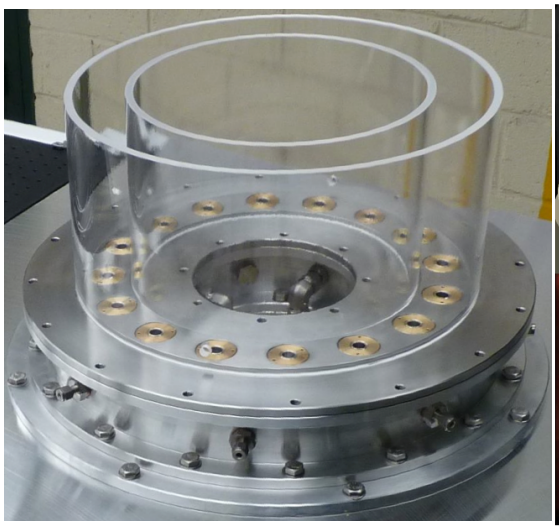
Emerging Computer Architectures

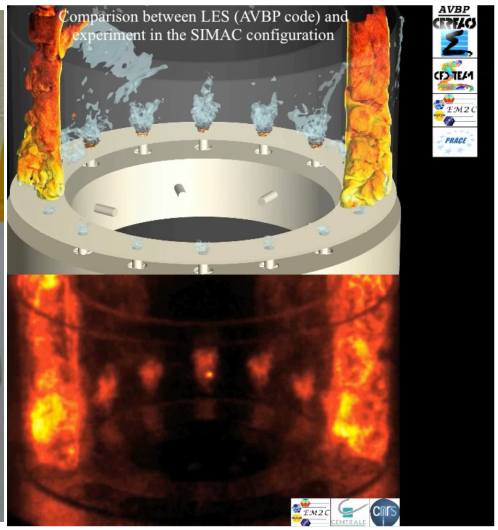
- Are needed by applications
- Applications are given (as function of time)
- Architectures are given (as function of time)
- Algorithms and software must be adapted or created to bridge to computer architectures for the sake of the complex applications

CURIE
6144 cores / run
15M CPU.h

E. Riber, B. Cuenot, F. Duchaine (CERFACS),
R. Vicquelin, M. Boileau, M. Philip , T. Schmitt, S. Candel (EM2C)

Simulation of Ignition in a Multiple Annular Combustor injector and comparison with experiments





CURIE
6144 cores / run
15M CPU.h

E. Riber, B. Cuenot, F. Duchaine (CERFACS),
R. Vicquelin, M. Boileau, M. Philip , T. Schmitt, S. Candel (EM2C)

Simulation of Ignition in a Multiple Annular Combustor injector and comparison with experiments

Floating point calculation

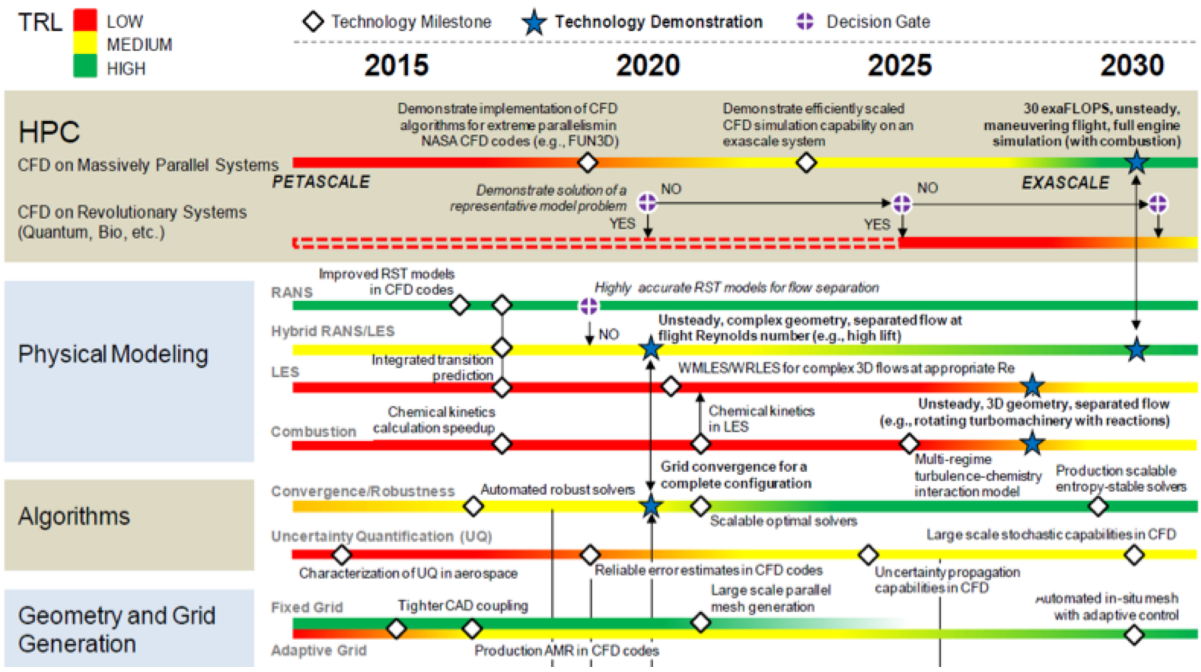
2,236273287 x 7,98729837 = 17,86178198 : 1 flop ~ 10 s

15M CPU.h ~ 900 10^{18} flop = 900 Exaflop

- **Amount of work:**
  world population ~ 10 milliards (10^{10})
  => 90 milliard flop / person
- **Time to solution :**
  world population busy for 685 000 years
- **On an Exacsale computer: 15 mn**

(100 000 000 flop/person/sec)

Enabling industrial progresses



Boeing, Pratt & Whitney, NASA, Stanford, MIT: J.Slotnik et al., CFD Vision 2030 Study. Tech. rep. NASA, 2014

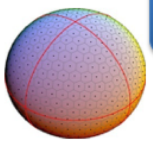
Enabling scientific … and societal progresses

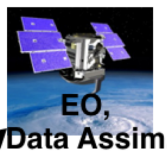
enes
EUROPEAN NETWORK FOR EARTH SYSTEM MODELLING

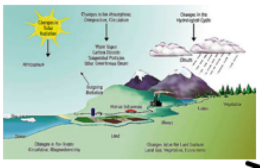
Infrastructure Strategy Roadmap

Needs for HPC

x 3 ≈ x 27
x 100 ≈ x 10^6



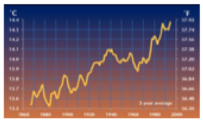


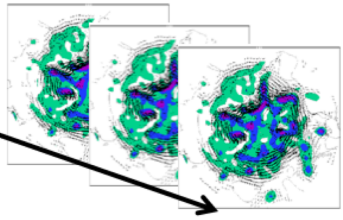


x 5-10

Resolution
Computing Resources
Complexity
Duration and/or Ensemble size

From Jim Kinter, the World Modelling Summit, 2008



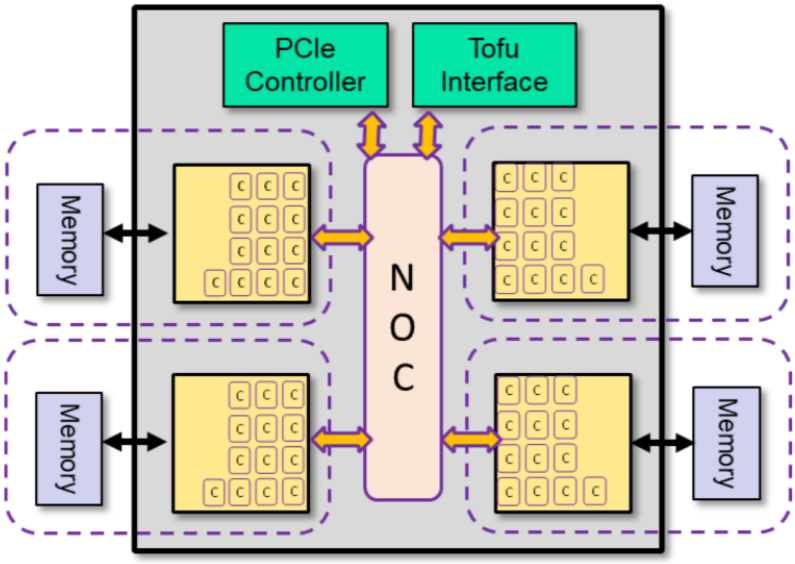


ensemble: x 10

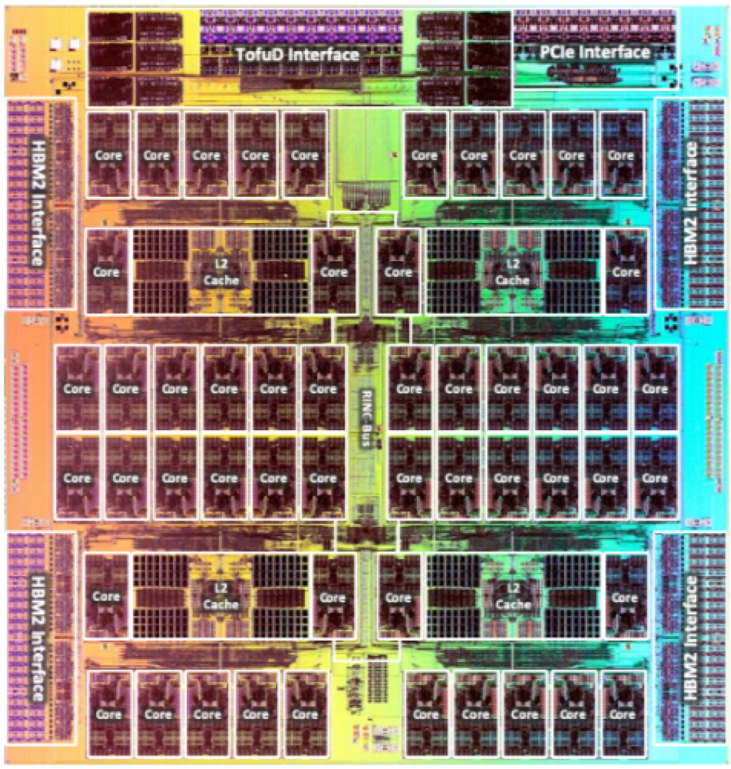
duration: x 10-100

Fugaku – Riken - Japan

A64FX Processor



A64FX Processor

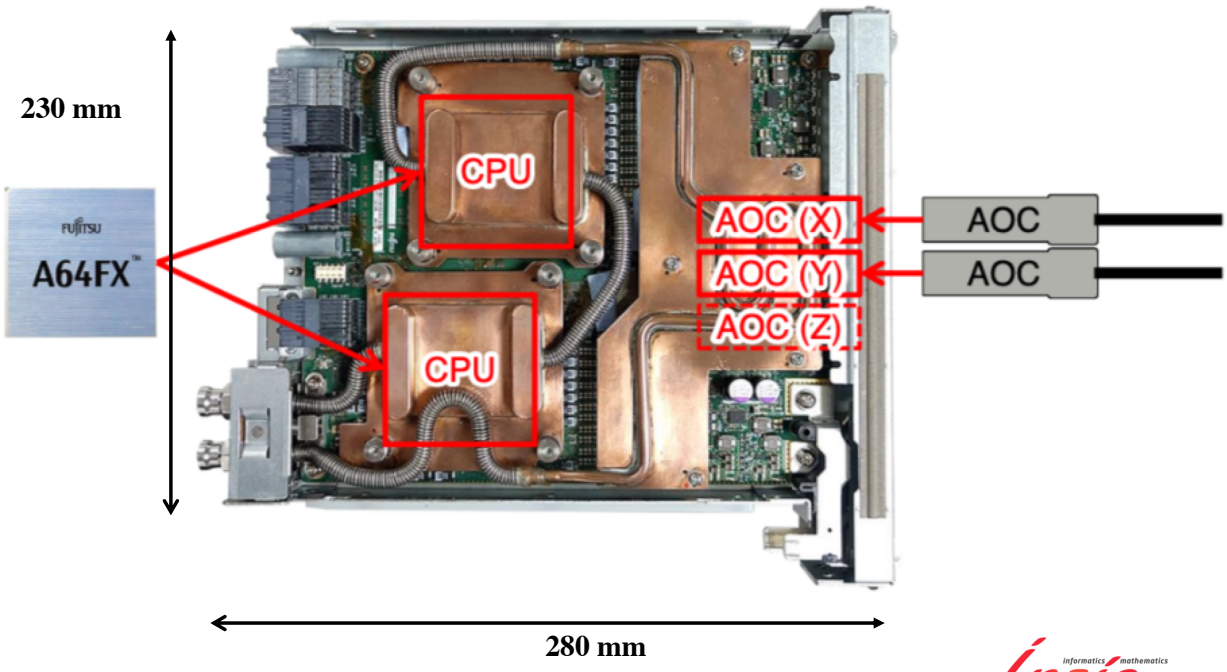


A64FX Processor

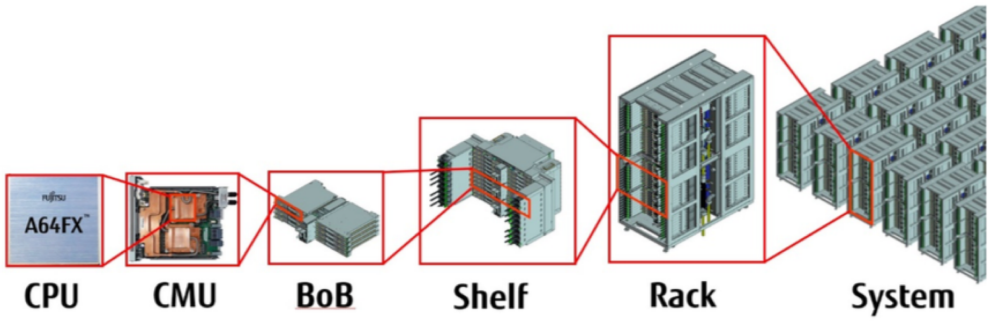
60 mm

60 mm

A64FX CMU
CPU Memory Unit

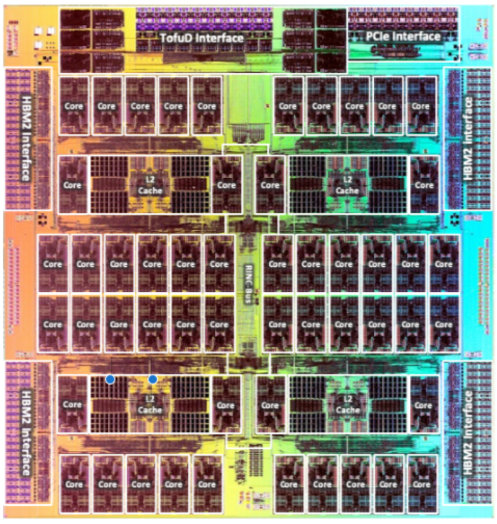


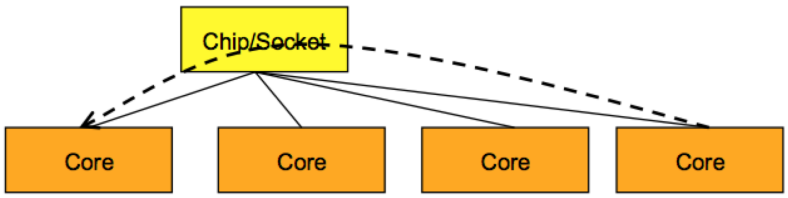
Fugaku System



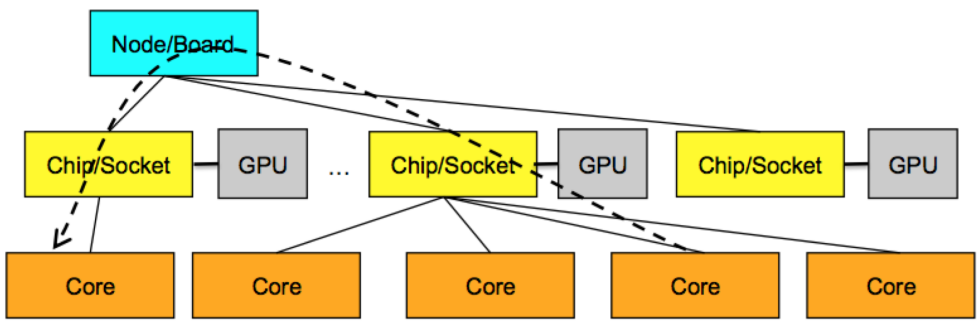
Unit	# of nodes	Description
CPU	1	Single-socket node with HBM2 and Tofu interconnect D
CMU	2	CPU Memory Unit: 2× CPU
BoB	16	Bunch of Blades: 8× CMU
Shelf	48	3× BoB
Rack	384	8× Shelf
System	152,064	As a Fugaku system 396 x Rack

Example of typical parallel machine



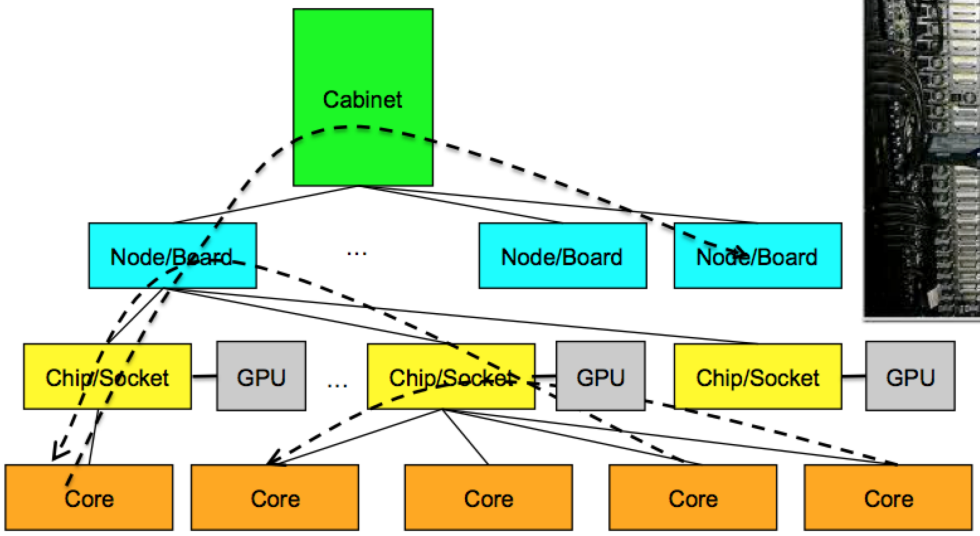


Example of typical parallel machine



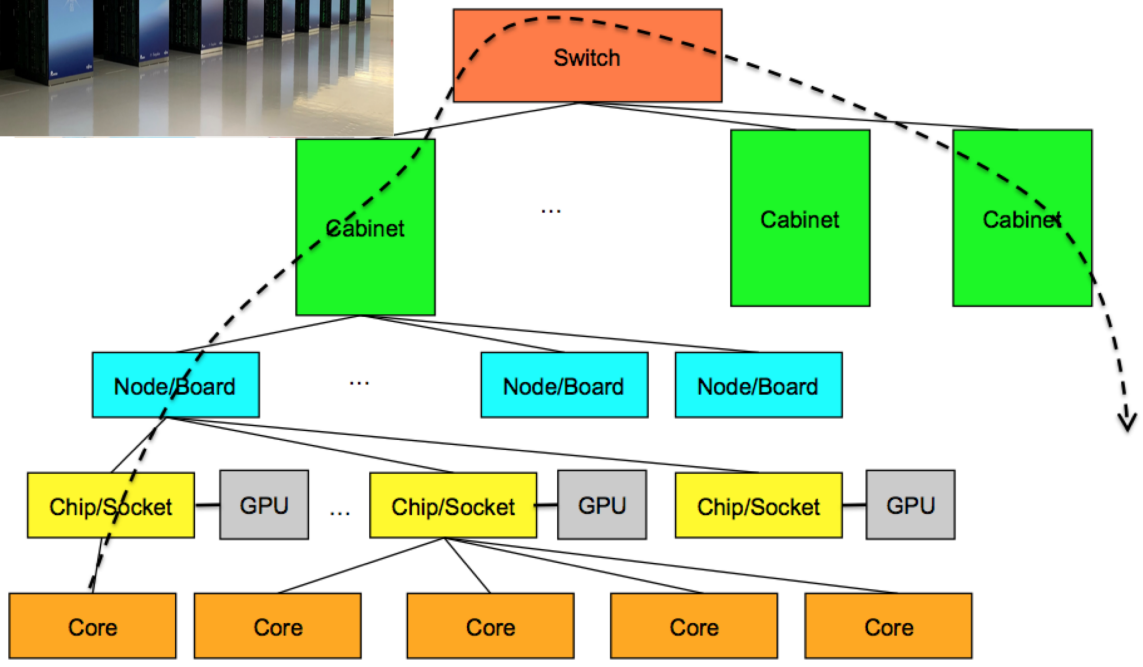
Example of typical parallel machine

Shared memory programming between processes on a board and
a combination of shared memory and distributed memory programming
between nodes and cabinets



Example of typical parallel machine

Combination of shared memory and distributed memory programming



The High Cost of Data Movement

- Flop/s or percentage of peak flop/s become much less relevant

Approximate power costs (in picoJoules)

	2011
DP FMADD flop	100 pJ
DP DRAM read	4800 pJ
Local Interconnect	7500 pJ
Cross System	9000 pJ

Source: John Shalf, LBNL

- Algorithms & Software: minimize data movement; perform more work per unit data movement.

Power Cost of Frequency

- Power ∝ Voltage^2 x Frequency (V^{2}F)
- Frequency ∝ Voltage
- Power ∝ Frequency^3

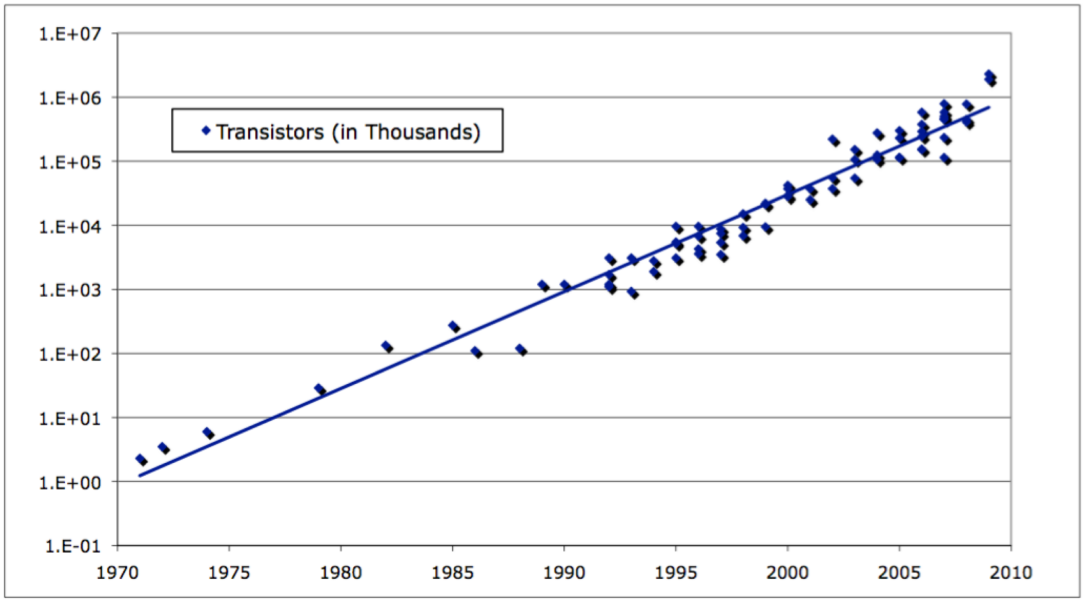
	Cores	V	Freq	Perf	Power	PE (Bops/watt)
Superscalar	1	1	1	1	1	1
"New" Superscalar	1X	1.5X	1.5X	1.5X	3.3X	0.45X
Multicore	2X	0.75X	0.75X	1.5X	0.8X	1.88X

(Bigger # is better)

50% more performance with 20% less power

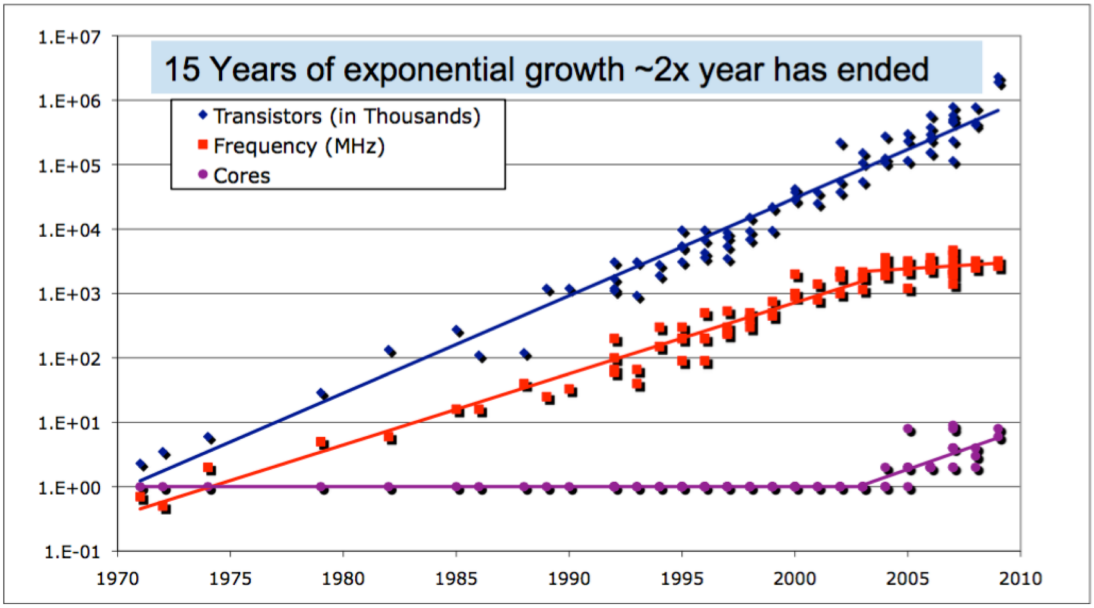
Preferable to use multiple slower devices, than one superfast device

Moore's Law is Alive and Well



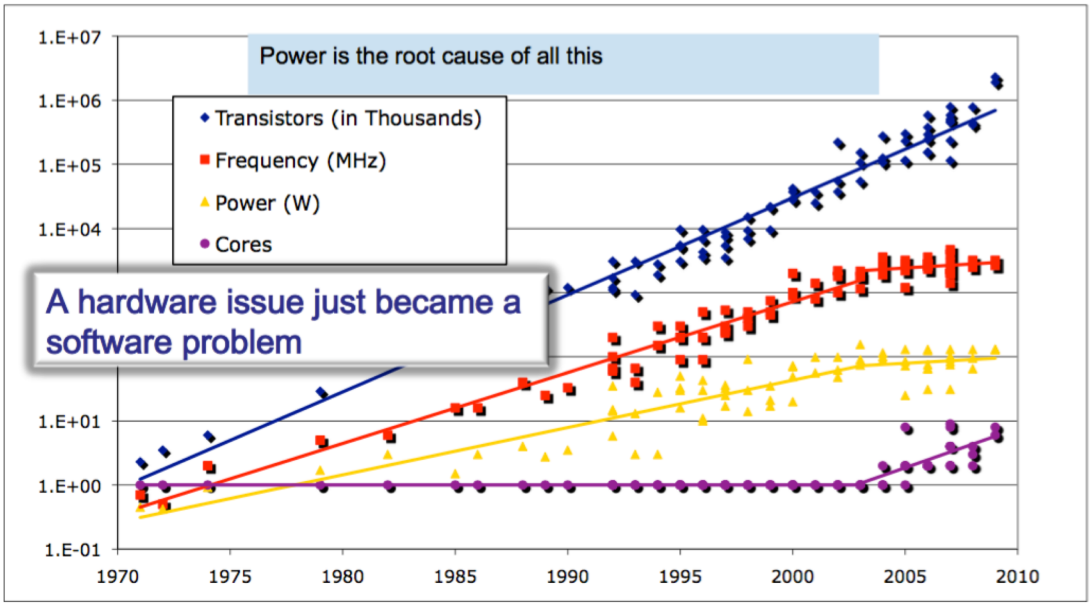
Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanoviç
Slide from Kathy Yelick

But Clock Frequency Scaling Replaced by Scaling Cores / Chip



Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanoviç
Slide from Kathy Yelick

Performance Has Also Slowed, Along with Power



Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanoviç
Slide from Kathy Yelick

Major Changes to Software & Algorithms

- **Must rethink the design of our algorithms and software**
  - **Another disruptive technology**
    - **Similar to what happened with cluster computing and message passing**
  - **Rethink and rewrite the applications, algorithms, and software**
  - **Data movement is expense**
  - **Flop/s are cheap, so are provisioned in excess**

Critical Issues at Peta & Exascale for Algorithm and Software Design

- Synchronization-reducing algorithms
  - Break Fork-Join model
- Communication-reducing algorithms
  - Use methods which have lower bound on communication
- Mixed precision methods
  - 2x speed of ops and 2x speed for data movement
- Autotuning
  - Today's machines are too complicated, build "smarts" into software have experiment to optimize.
- Fault resilient algorithms
  - Implement algorithms that can recover from failures/bit flips
- Reproducibility of results
  - Today we can't guarantee this. We understand the issues, but some of our "colleagues" have a hard time with this.

GPU vs CPU

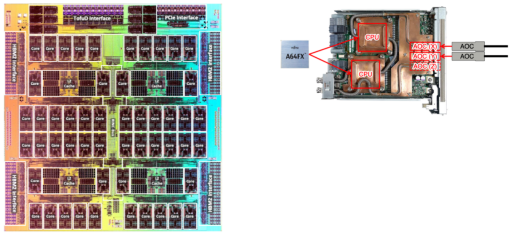
GPU

- hundred of simpler cores
- thousand of concurrent hardware threads
- maximize floating-point (fp) throughput
- most die surface for integer and fp units

Nvidia A 100: 8 192 cores ; 19,5 Tflops (fp32) / 9,7 Tflops (fp64)

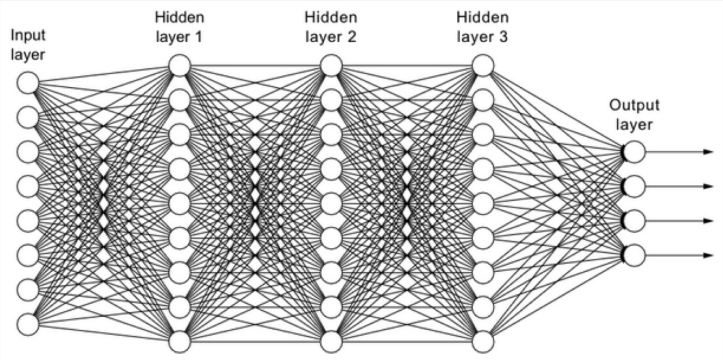
CPU

- few very complex cores
- single-thread performance optimization
- transistor space dedicated to complex instruction-level parallelism (ILP)
- few die surface for integer and fp units



A64X ARM, SVE: 48 cores; 3.4 Tflops (fp64)

Platforms for data-driven simulation: DNN (training+inference)



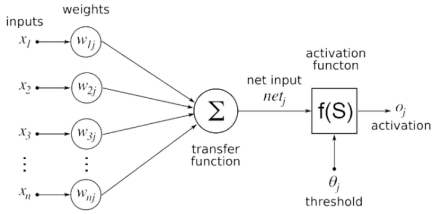


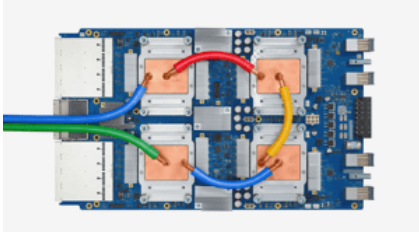
Figure 1 An example of a fully connected neural network (fcNN) model with three hidden layers

Calculation between the layers essentially reduces to matrix-matrix multiplications (mixed arithmetic)

TensorFlow PyTorch

Google platform: Tensor Processing Unit

(application-specific integrated circuit)

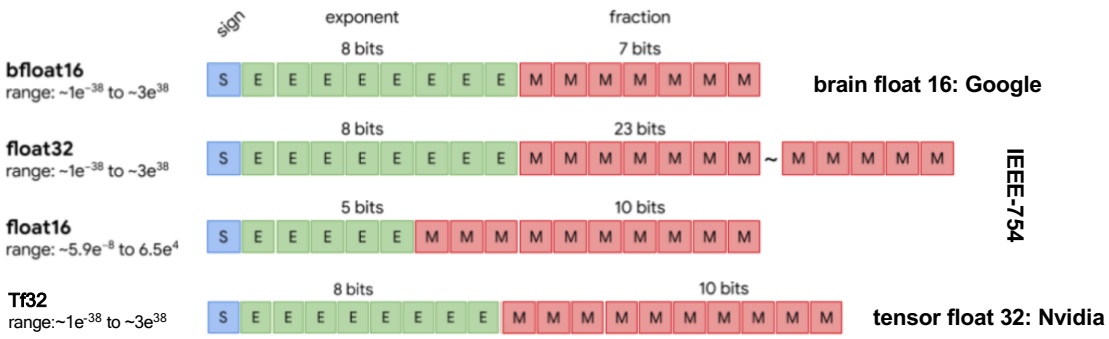


Cloud TPU v3
420 TFlops, 128 GB HBM
(High Bandwidth Memory)

Pod Cloud TPU v3
> 100 PFlops, 32 TBHBM
2D torus network

AI calculations do not require « high » accuracy

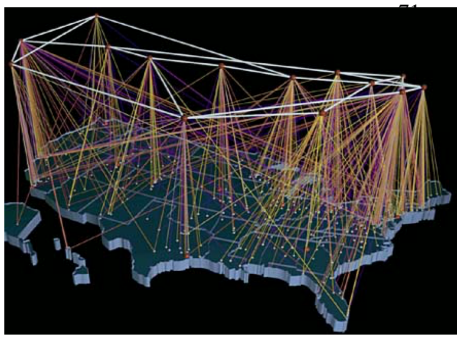
new floating point formats and hardware



Enhance performance, thanks to vendor hardware design (systolic mat-mat) and reduce memory traffic

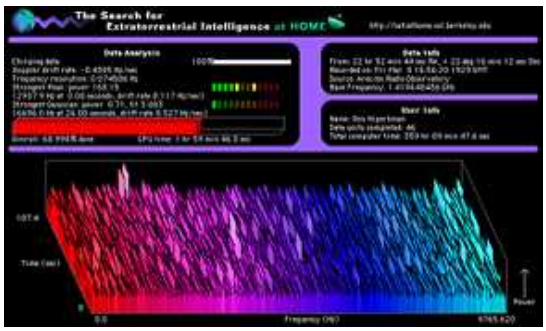
Nvidia A 100: FP64=9.7 Tflops, FP32=19.5 Tflops, b16=312 Tflops – 400 W
A64FX : FP64 = 3.4 Tflops – 190 W

Prospectives: the GRID …



- Users will have access to "power" on demand.
- Computing power is produced much like utilities such as power and water are produced for consumers.
- To treat CPU cycles and software like commodities.
- Enable the coordinated use of geographically distributed resources – in the absence of central control and existing trust relationships.

SETI@home



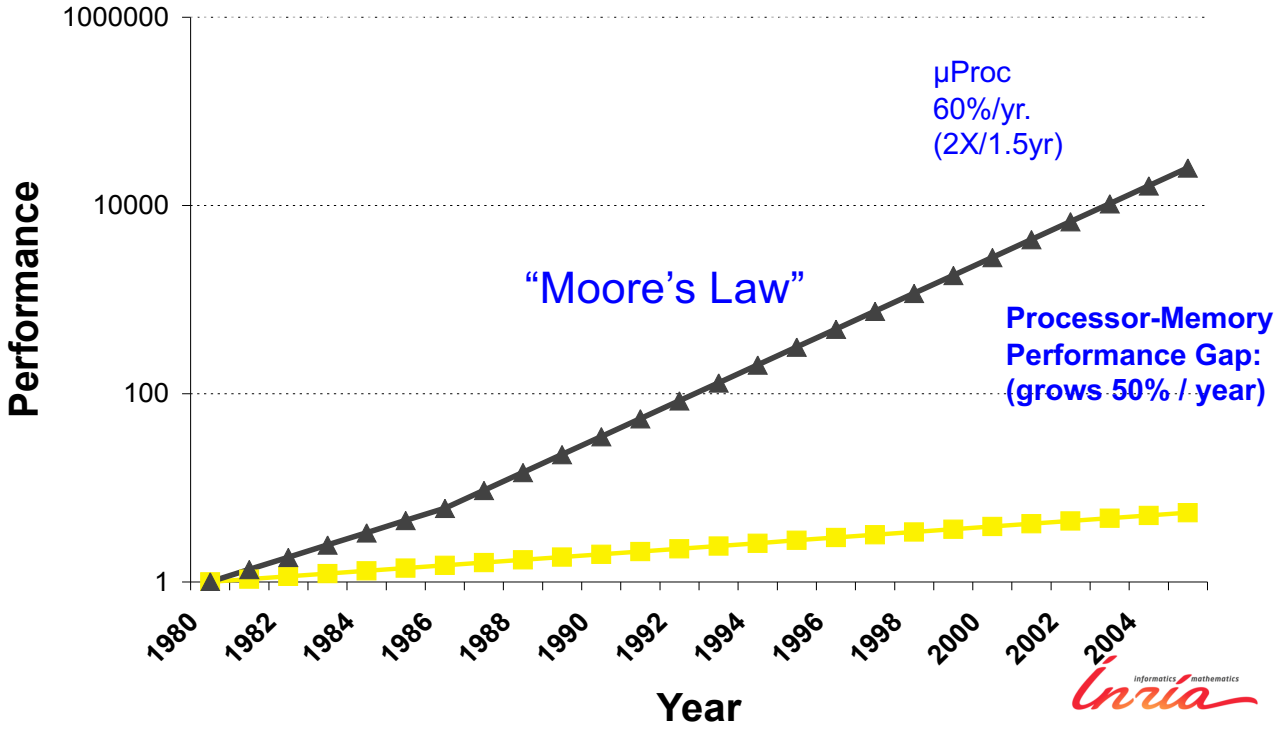
- Use thousands of Internet-connected PCs to help in the search for extraterrestrial intelligence.
- Uses data collected with the Arecibo Radio Telescope, in Puerto Rico
- When their computer is idle or being wasted this software will download a 300 kilobyte chunk of data for analysis.
- The results of this analysis are sent back to the SETI team, combined with thousands of other participants.

- Largest distributed computation project in existence
  - ~ 400,000 machines
  - Averaging 26 Tflop/s
- Today many companies trying this for profit …

Memory v.s. CPU peak performance

- Computational optimizations
  - **Theoretical peak:(# fpus)*(flops/cycle) * Mhz**
    - PIII: (1 fpu)*(1 flop/cycle)*(850 Mhz) = 850 MFLOP/s
    - Athlon: (2 fpu)*(1flop/cycle)*(600 Mhz) = 1200 MFLOP/s
    - Power3: (2 fpu)*(2 flops/cycle)*(375 Mhz) = 1500 MFLOP/s
    - SX-8R: (4fpu)*(4flops/cycle)*(2.2 Ghz) = 35.2 GFLOP/s
- Memory optimization
  - **Theoretical peak: (bus width) * (bus speed)**
    - PIII : (32 bits)*(133 Mhz) = 532 MB/s = 66.5 MW/s
    - Athlon: (64 bits)*(133 Mhz) = 1064 MB/s = 133 MW/s
    - Power3: (128 bits)*(100 Mhz) = 1600 MB/s = 200 MW/s
- Operations like:
  - $\alpha = x^T y$: 2 operands (16 Bytes) needed for 2 flops; at 850 Mflop/s will require **850** MW/s bandwidth
  - $y = \alpha x + y$: 3 operands (24 Bytes) needed for 2 flops; at 850 Mflop/s will require **1275** MW/s bandwidth

Memory v.s. Performance
The technological constraints



Processor performance v.s. memory

Increase the processor performance implies to sophisticate the data access management

Two constraints on memory systems

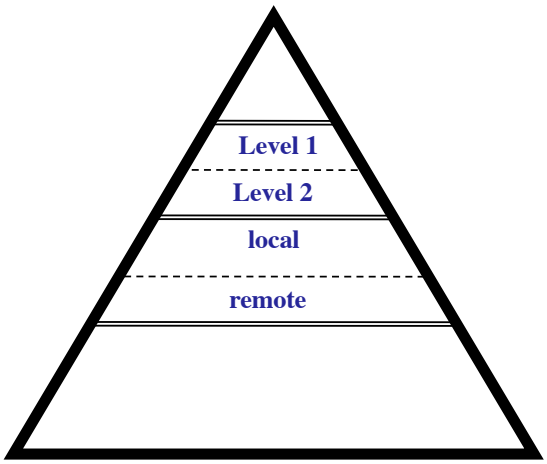
- Infinite size (or at least as big as your application)
- Access down to one clock cycle

⇒ **contradictory at the hardware technology level !!**

The real performance of a computer depends entirely upon the performance of its memory system.

Memory system

- **Memory hierarchy**
  - technology constraints => small but fast memory close to the CPU.



Memory reference behaviour

- **Locality properties**
  - temporal locality
  - spatial locality

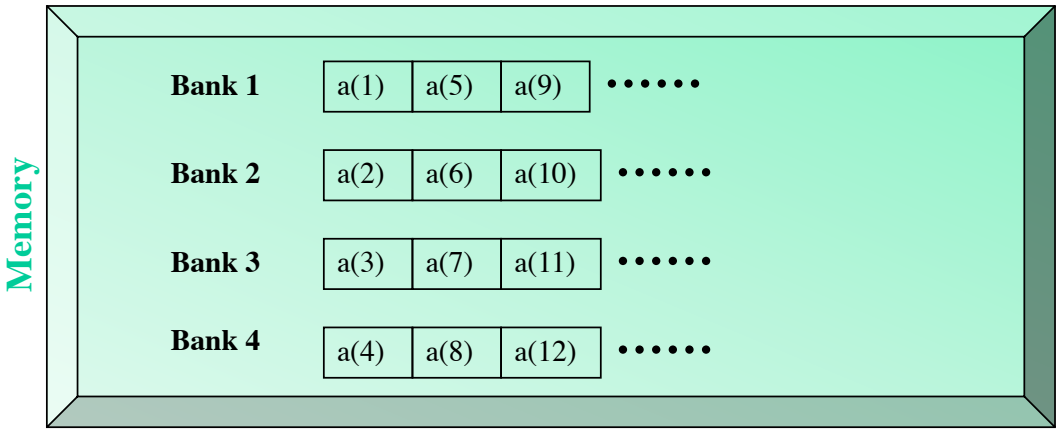
Exploit these properties to manage the memory

- memory interleaving (vector machine mainly)
- cache management policy

Memory interleaving

The memory is subdivided into several independent memory banks/modules.

Low order interleaving : well suited for pipelining memory access



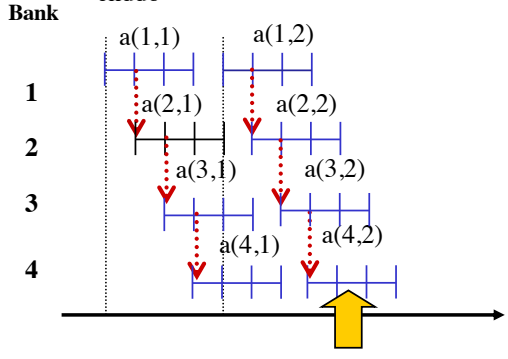
Memory interleaving : main features

Bank cycle time : time interval during which the bank cannot be referenced again
Bank conflict : consecutive accesses to the same bank in less than bank cycle time
Stride : memory address interval between successive elements

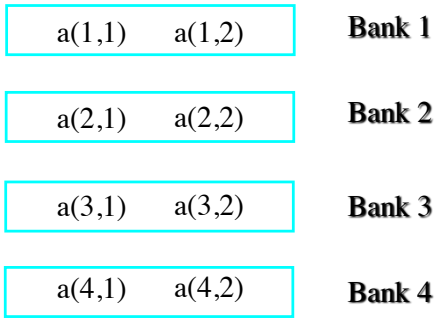
Low interleaved memory, 4 banks, bank cycle time = 3 clock period

```
% column access
real a(4,2)
do j=1,2
  do i=1,4
    … = a(i,j)
  enddo
enddo
```

Stride = 1 => 8 CP



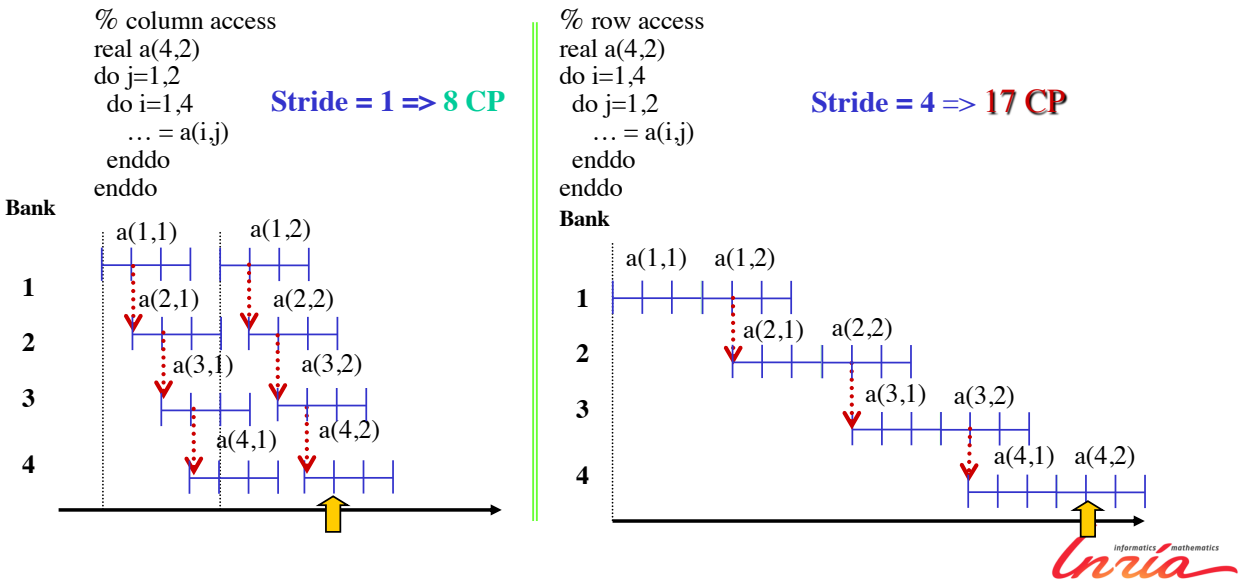
real a(4,2)



Memory interleaving : main features

Bank cycle time : time interval during which the bank cannot be referenced again
Bank conflict : consecutive accesses to the same bank in less than bank cycle time
Stride : memory address interval between successive elements

Low interleaved memory, 4 bank, bank cycle time = 3 clock period



Possible remedy

real a(4,2)
real b(5,2)

a(1,1)	a(1,2)	b(1,1)	b(5,1)	b(4,2)	Bank 1
a(2,1)	a(2,2)	b(2,1)	b(1,2)	b(5,2)	Bank 2
a(3,1)	a(3,2)	b(3,1)	b(2,2)		Bank 3
a(4,1)	a(4,2)	b(4,1)	b(3,2)		Bank 4

On the SX-8R: row access is 100 times slower than column acces for lda = 4096

Cache management policy

Cache line : unit of transfer between cache and memory (contiguous addresses)

Cache fault : reference to a data not loaded in the cache

Placement policy : correspondence between main memory and cache memory address.

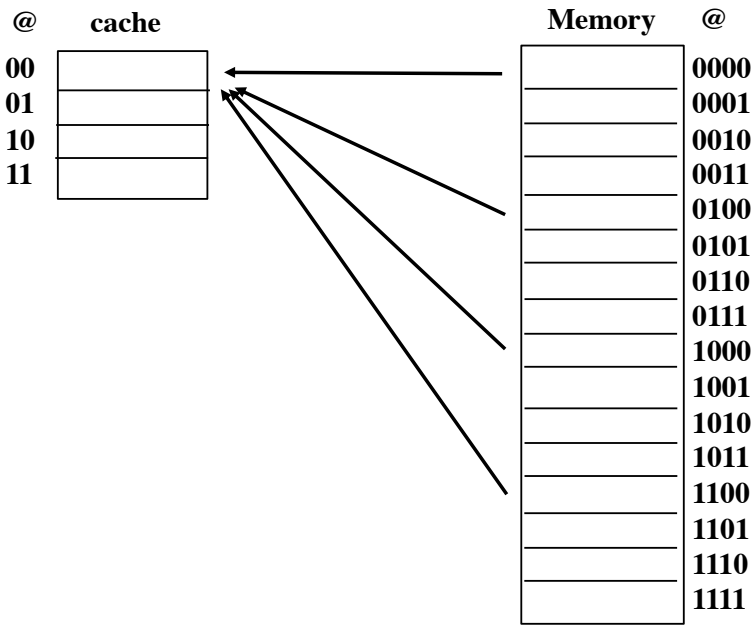
- Direct mapping :
  - last digit of memory address define cache address
  - a data can only be stored in a given location
- Fully associative :
  - No predefined correspondence => replacement policy (LRU)
- Set associative (trade off) :
  - a data can be stored only in a given set but any where within that set

  ex : 4 way associative => each set contains 4 data

Direct mapping

Memory has 16 entries => @ mem = 4 bits

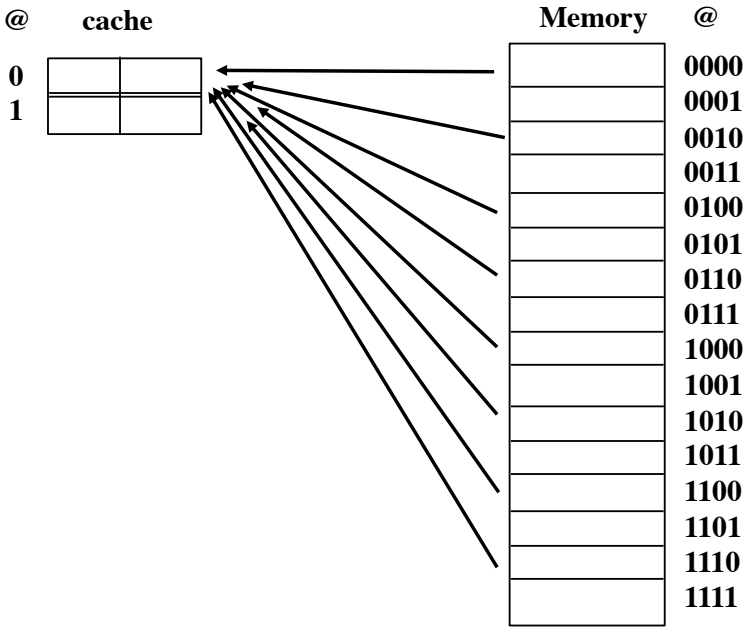
Cache has 4 entries => @ cache = 2 bits



Set associative

Memory has 16 entries => @ mem = 4 bits

Cache has 2 sets => @ cache = 1 bit



Cache effect

Cache hit : data in cache (load data from cache to processor)

Cache miss : data not in cache (load cache line containing data from memory to cache, then from cache to processor)

Typically cache miss 10 times slower than cache hit

When cache is full : Least recently used cache line is flushed to memory (LRU)

Cache size = 2 words, cache line = 2 words, cache hit = 1 clock period

```
% column access – column major
real  a(4,2)
do j=1,2
  do i=1,4
    … = a(i,j)
  enddo
enddo
```

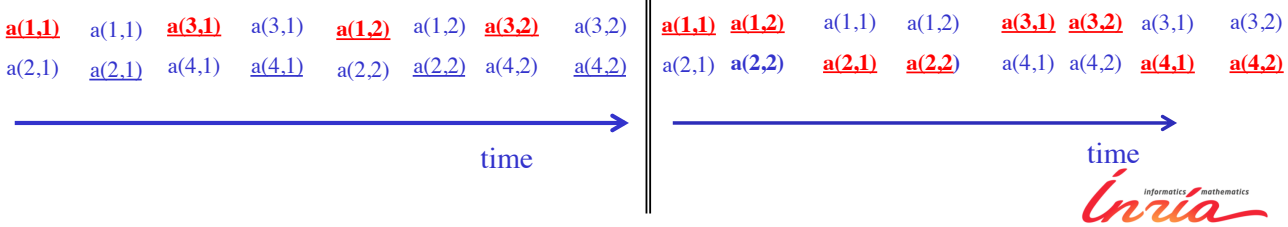
Stride = 1 => 44 CP

4 hits , 4 misses

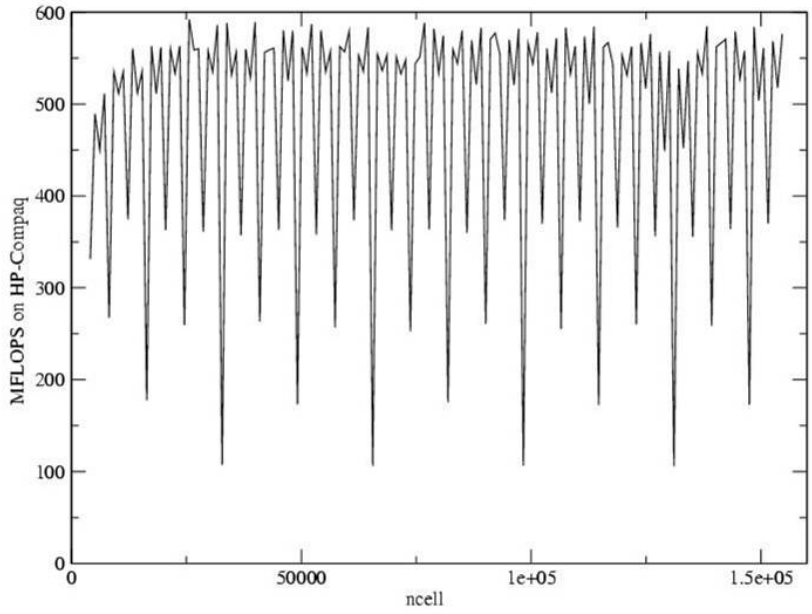
```
% row access – column major
real a(4,2)
do i=1,4
  do j=1,2
    … = a(i,j)
  enddo
enddo
```

Stride = 4 => 80 CP

8 misses



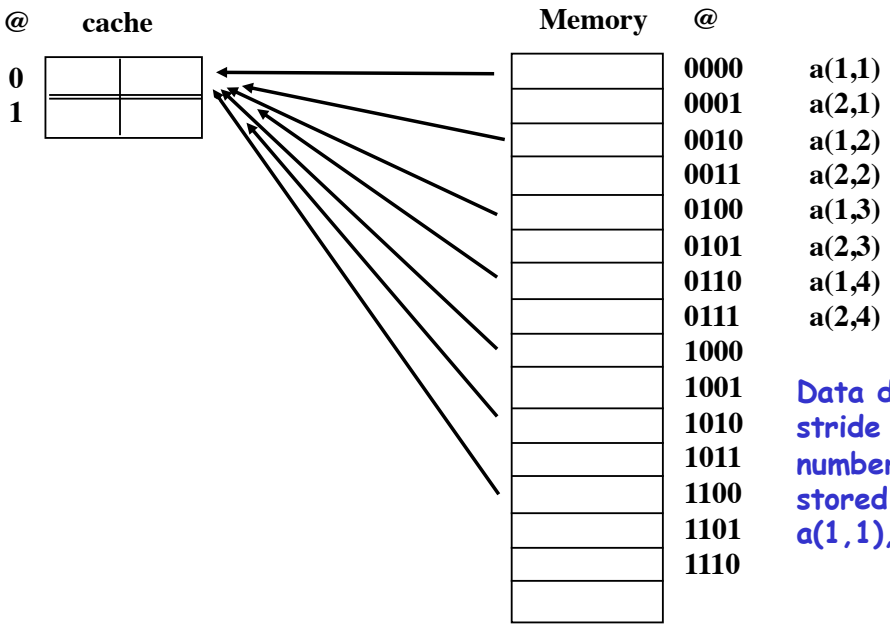
Impact of the leading dimension



Set associative

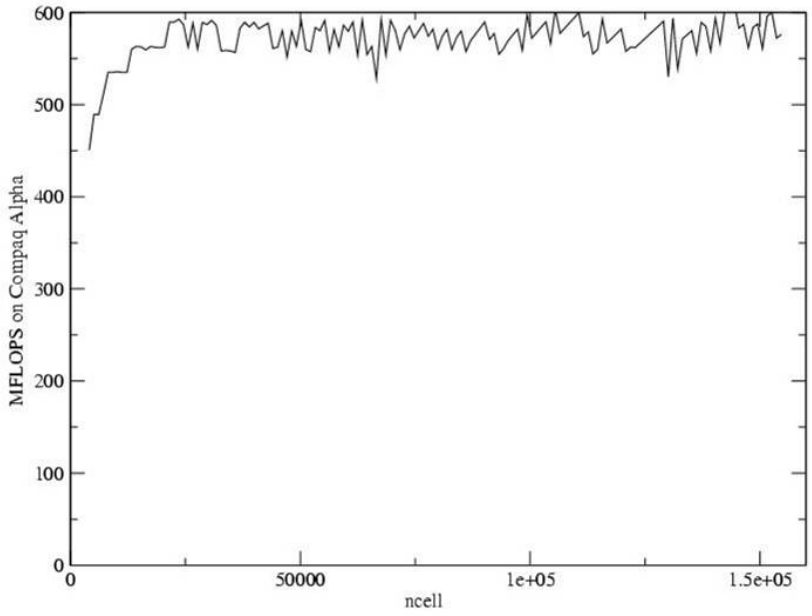
Memory has 16 entries => @ mem = 4 bits

Cache has 2 sets => @ cache = 1 bit



Data distant in memory from a stride that is a multiple of the number of cache sets, will be stored in the same set (e.g., a(1,1), a(1,2), …

Impact of the leading dimension

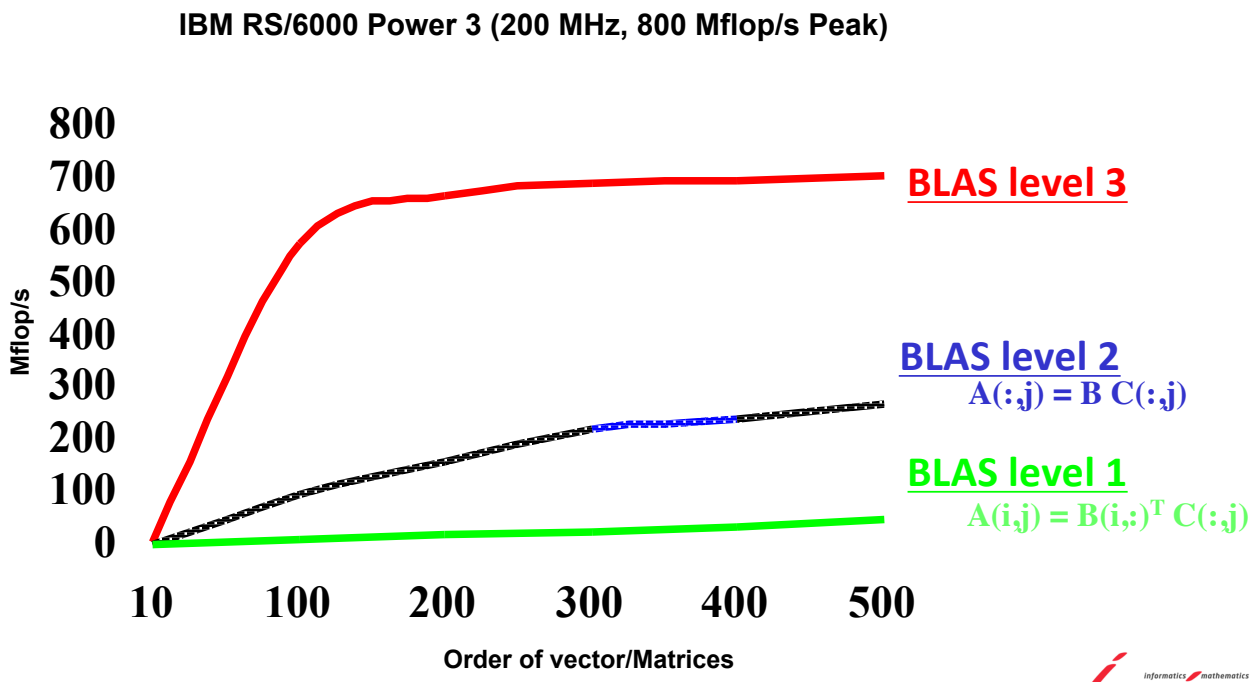


BLAS efficiency

BLAS	Nb flop	Nb mem acces	Ratio
niveau 1 $y=\alpha x+y$	$2n$	$3n$	2/3
niveau 2 $y= \alpha Ax + \beta y$	$2n^2$	$n^2 +3n$	2
niveau 3 $C = \alpha AB + \beta C$	$2n^3$	$4n^2$	n/2

BLAS Performance

$$A = B \times C$$



Processor/core features: How to compute faster ?

- **Pipelining/Vectorization** : parallelism within elementary instruction

  idea : decompose an instruction into sub-instructions that may be applied simultaneously to different data.

- **Overlapping** : parallelism between elementary instructions

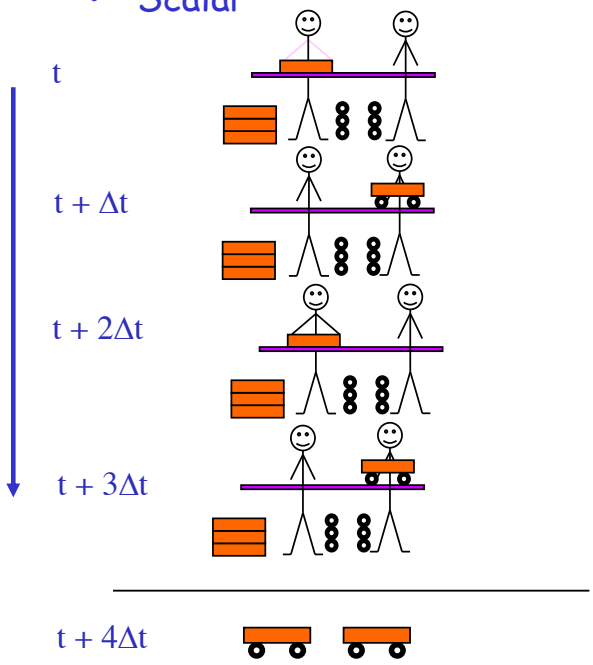
  idea : start next independent operation on a separate functional unit at the next cycle

- **Chaining** : parallelism between elementary instructions

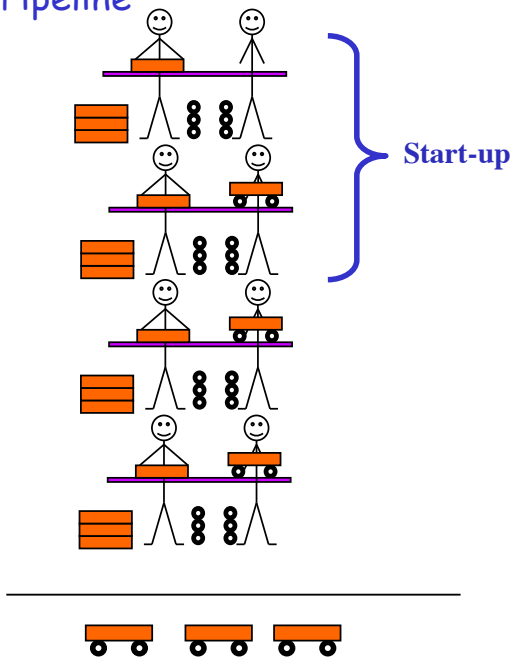
  idea : output from one functional unit directly fed into next functional unit.

Pipeline/vectorization v.s. Scalar

- Scalar

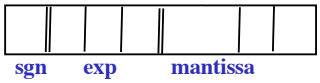


- Pipeline



Multiplier floating point unit

Floating point arithmetic



Compute: -12 *48 = -576
-12 = 0 | 001|0110 = 6*2
48 = 1 | 010|1100 = 12*2²

1-Product of the mantissa
0110
*** 1100**
11000
110000
=1001000 = 1001*2³

2-Add the exponent
001
+ 010
= 011

3-Normalize (update the exponent)
011
+ 011
= 110

4- result
0 | 110 |1001 = -9 * 2⁶ = -9*64 = -576

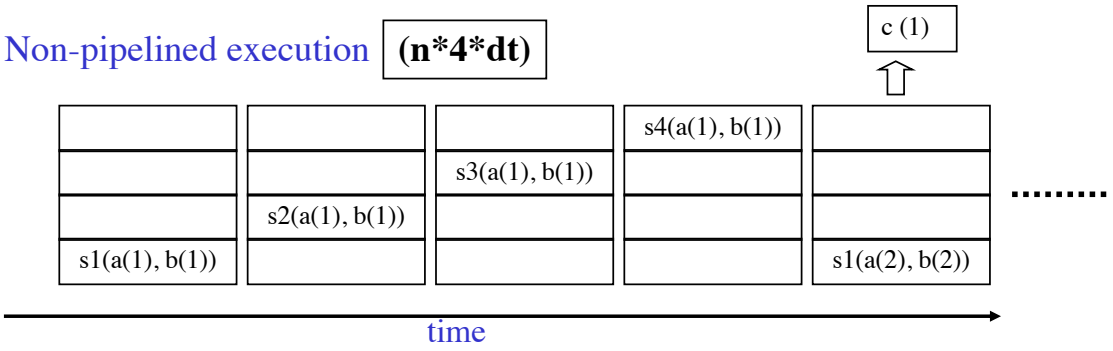
Pipelined floating point functional unit

- Multiplier example
  - s1 : separate mantissa from exponent
  - s2 : multiply mantissa
  - s3 : add exponent
  - s4 : normalise the result

- Loop to be performed

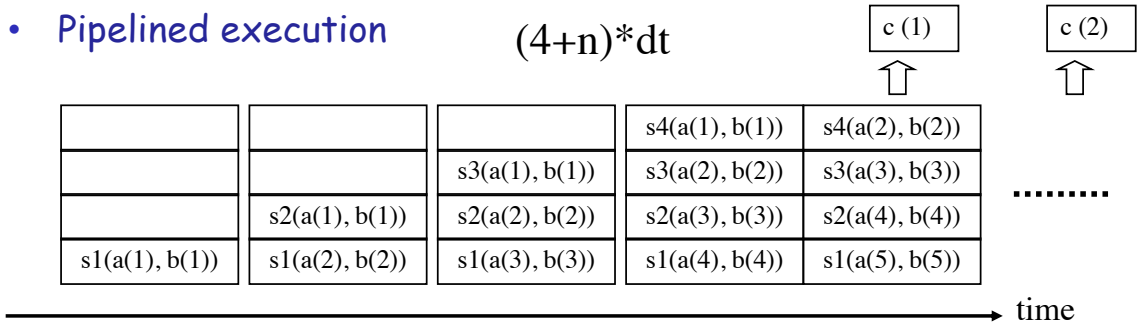
```
do  i = 1,n
      c(i) = a(i)*b(i)
enddo
```

- Non-pipelined execution **(n*4*dt)**



Pipelined floating point functional unit

- Pipelined execution (4+n)*dt



Characteristics

number of stages, clock cycle => start-up time

Constraints

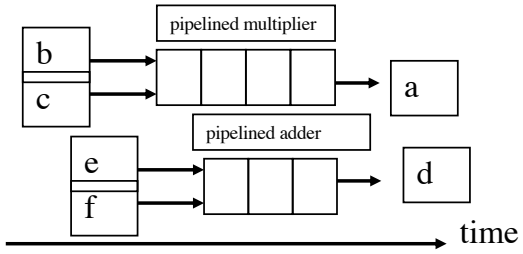
Vector load/store must be done at the same rate to avoid interruptions during pipelined execution.

Overlapping

idea :

start next independent operation on a separate functional unit at the next cycle

```
do i=1,n
   a(i) = b(i)*c(i)
   d(i) = e(i)+f(i)
enddo
```



Execution time

- with overlap : Max[(start-up Mul),(start-up Add)+1] + n*dt
- without overlap :[(start-up Mul) + n*dt] + [(start-up Add) + n*dt]

Advantages

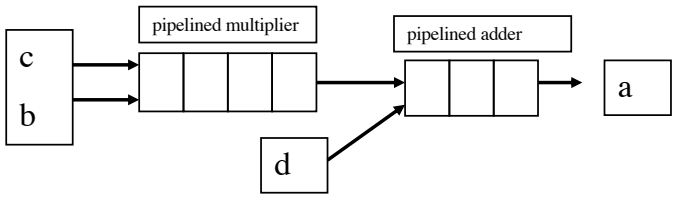
- more flops per clock cycle

Chaining

idea :

output from one functional unit directly fed into next functional unit

```
do i=1,n
   a(i) = (b(i)*c(i)) + d(i)
enddo
```



Execution time

- **with chaining : (start-up Mul) + (start-up Add) + n*dt**
- **without chaining :[(start-up Mul) + n*dt] + [(start-up Add) + n*dt]**

Advantages

- more flops per clock cycle
- save intermediate storage
- good use of data locality

Memory path v.s. performance Constraints on the architecture design

(J. Dongarra, Univ. Tenessee)

Triadic vector operation

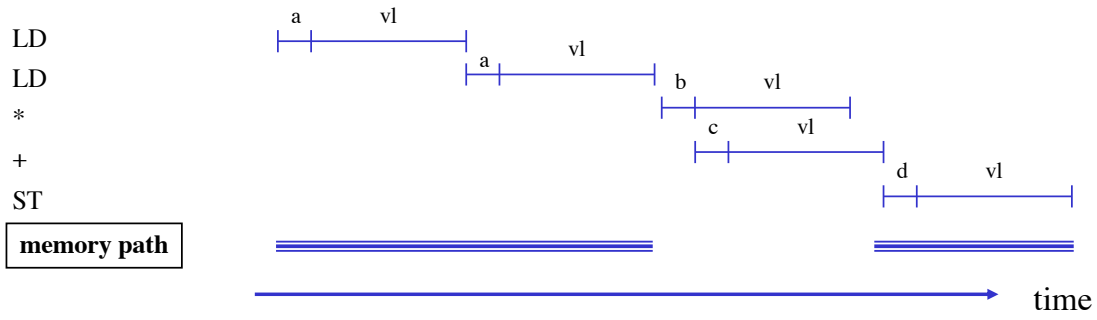
$$Y(1:n) = Y(1:n) * X(1:n) + Y(1:n)$$

This requires :

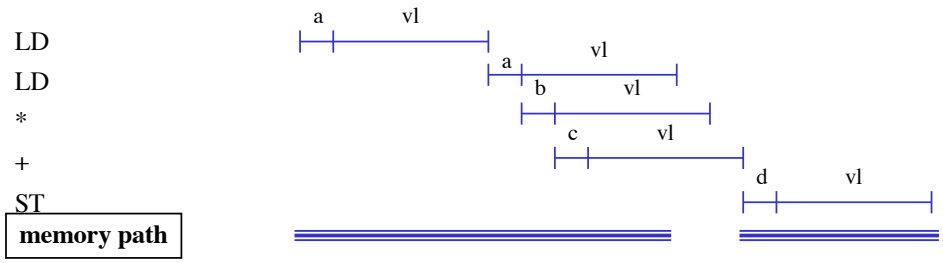
- Load vector Y (a= start-up time for load)
- Load vector X
- Vector Multiplication (b= start-up time for mult.)
- Vector Addition (c= start-up time for addition)
- Store vector Y (d= start-up time for store)
- vl = n*dt

Memory path v.s. performance (II)

Chained arithmetic

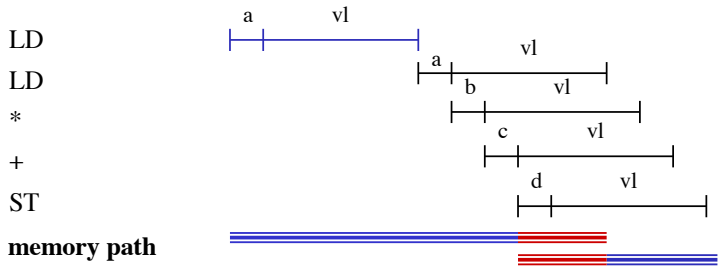


Chained load and arithmetic



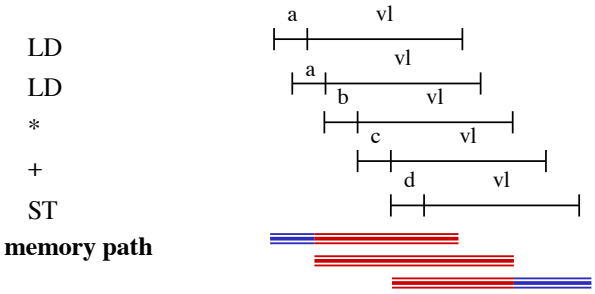
Memory path v.s. performance (III)

Chained load, arithmetic and store



2 memory paths

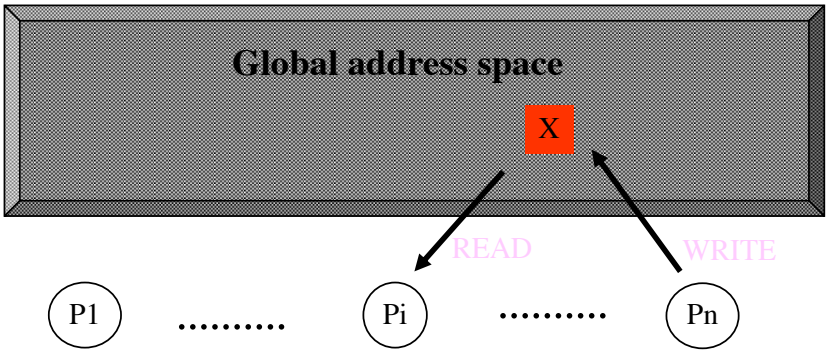
Overlapped load with chained operations



3 memory paths

Memory point of view
Global address space

Existence of an address space accessible directly through read/write to all the processors (enabling communication and synchronisation).

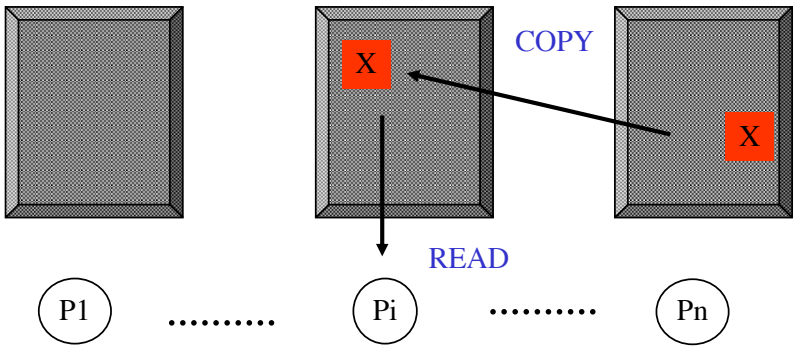


Memory point of view Global address space

- **Advantages**
  - simplicity for programming, managing resources and compiling.
- Drawbacks
  - Performance (memory contention).
  - interconnect complexity.
  - Hardware/software implementation (DVSM).

Memory point of view disjoint address space

Each processor has its own address space
Communication is done through explicit copy (message passing) between address space.



Memory point of view Disjoint address space

- Advantages
  - simplicity to develop
  - performance - local access
- Drawbacks
  - difficulty of programming
  - performance - remote access.

Combination

- subsets of processors having a global address space limited to that subset
- cluster of SMP

Main programming tools

- **Global address space:**

  Open-MP directives

- **Disjoint address space:**

  Message passing library (PVM, MPI).

Modelling the communication cost

- Cost of the message exchange
  - $T_{com} = \alpha + \beta n$

    α = latency

    β = inverse of bandwidth

  => Gather as much as possible in one message all the information to be exchanged between two processes

Measuring parallel performance

- Strong speedup

  $S_p = T_1/T_p$

  T_i : elapsed time to solve the problem on i processor(s)

  => ideal speedup = p

  => super-linear speedup > p (due to cache effect)

- Weak speedup
  - $\underline{S}_p = p\ T_1/T_p$

T_i : elapsed time to solve the scaled problem on i processor(s)

=> do not hide communication, no sensitive to cache effect.

The Amdhal's law !!

f: fraction of the code that is parallel

The execution time of N procs

$$\text{serial time} \times (f/N + (1-f))$$

The speed-up is on N procs

$$(f/N + (1-f))^{-1}$$

The assymptotic speed-up

$$(1-f)^{-1}$$

Ex: code 99% parallel => asymptotic speed-up: 100 !!

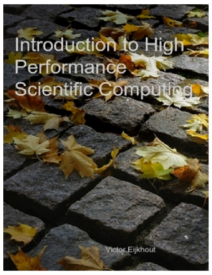
Useful source of information

- **www.cs.utk.edu/~dongarra/**
- **www.phys.uu.nl/~steen**
- **www.netlib.org/top500**

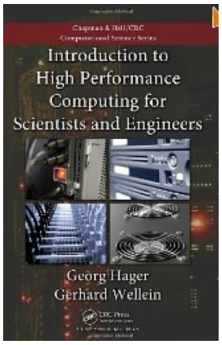
Many slides were borrowed from J. Dongarra's talks.

Some books

- **Introduction to High Performance Scientific Computing - Victor Eijkout**
  http://www.lulu.com/product/file-download/introduction-to-high-performance-scientific-computing/14605650



- **Introduction to High Performance Computing for Scientists and Engineers (Chapman & Hall/CRC Computational Science) - George Hager, Gerhard Wellein**



Inria informatics mathematics