Plan du cours

- Introduction (1 cours)
- Approche mémoire partagée (5 cours)
  - programmation (pthreads, OpenMP)
  - architecture (processeur, cache, SMP, NUMA)
- Calcul sur carte graphique (3 cours)
  - calcul vectoriel
  - architecture des GPGPU
  - Programmation (OpenCL, CUDA)
- Approche mémoire distribuée (2 cours)
  - programmation (MPI)
  - architectures (cluster, MPP, réseaux rapides)
  - approche mixte (MPI + thread)
- Mémoire virtuellement partagée (1 cours)

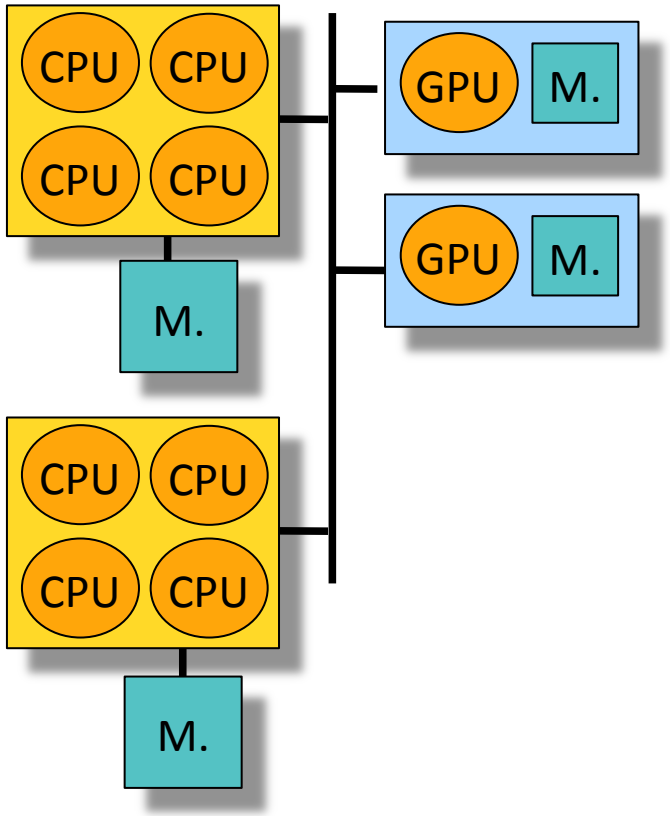
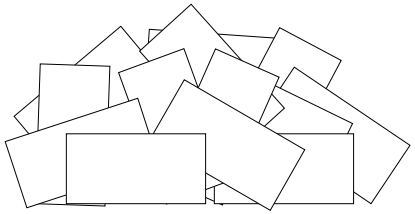


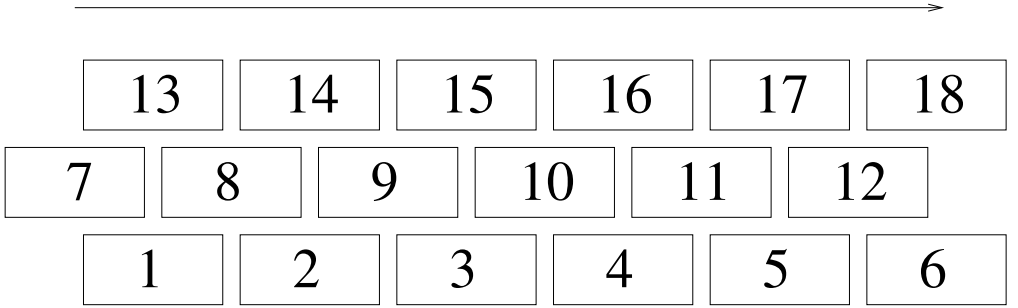
Illustration du parallélisme

[F. Pellegrini]

- Un maçon construit un mur



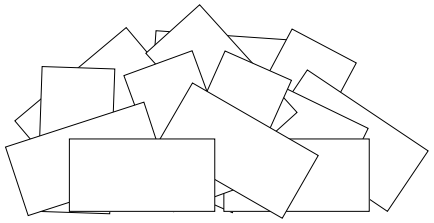
Tas de briques



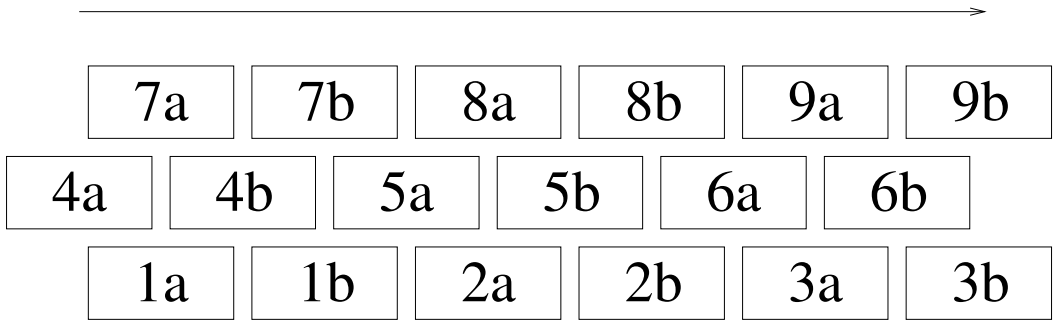
Mur

Illustration du parallélisme

- Deux maçons a et b posent les briques de façon consécutives



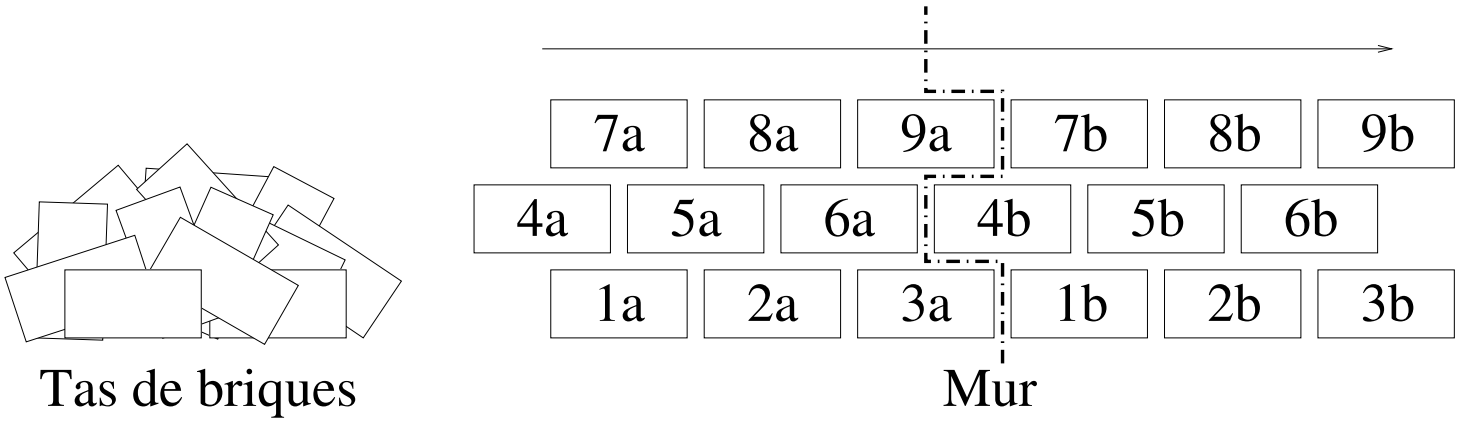
Tas de briques



Mur

Illustration du parallélisme

- Deux maçons a et b posent les briques sur deux portions différentes



- Soit le maçon b fait plus de chemin
- Soit le a maçon « envoie » les briques au maçon b

Illustration du parallélisme

- Deux maçons a et b posent les briques sur deux portions différentes

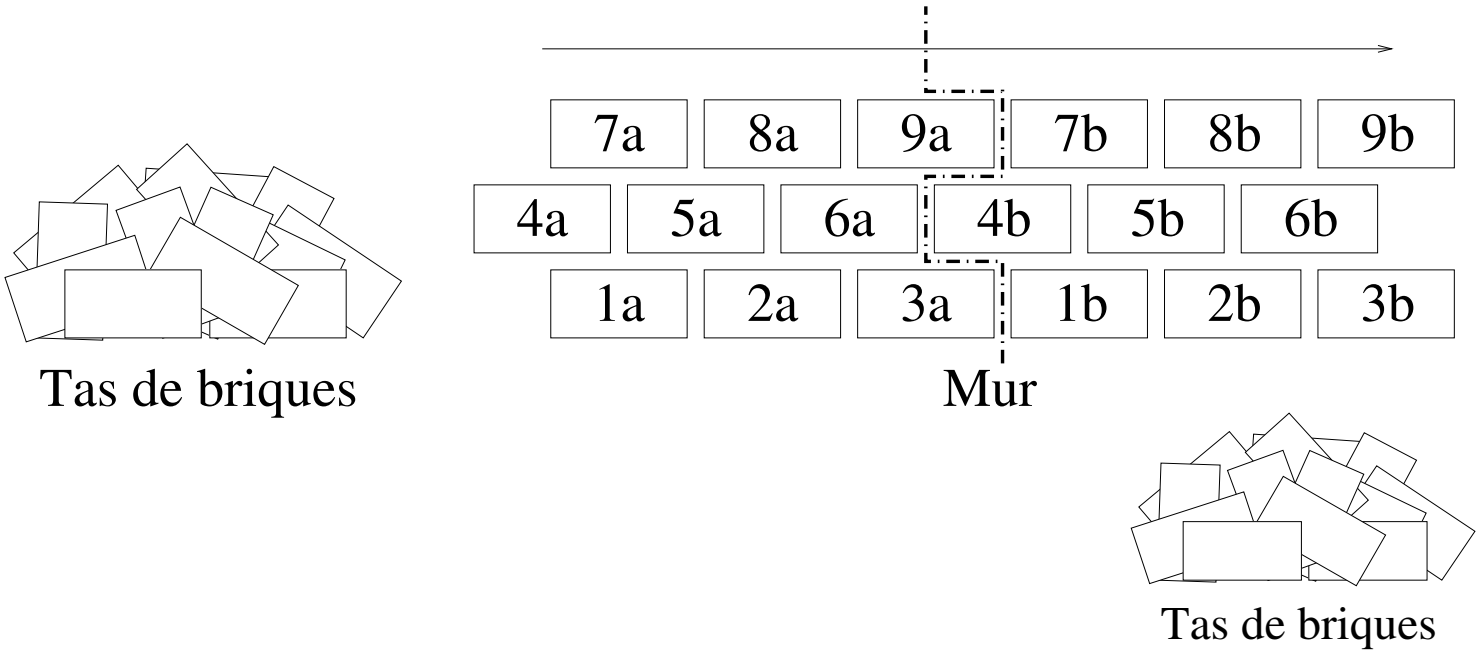


Illustration du parallélisme

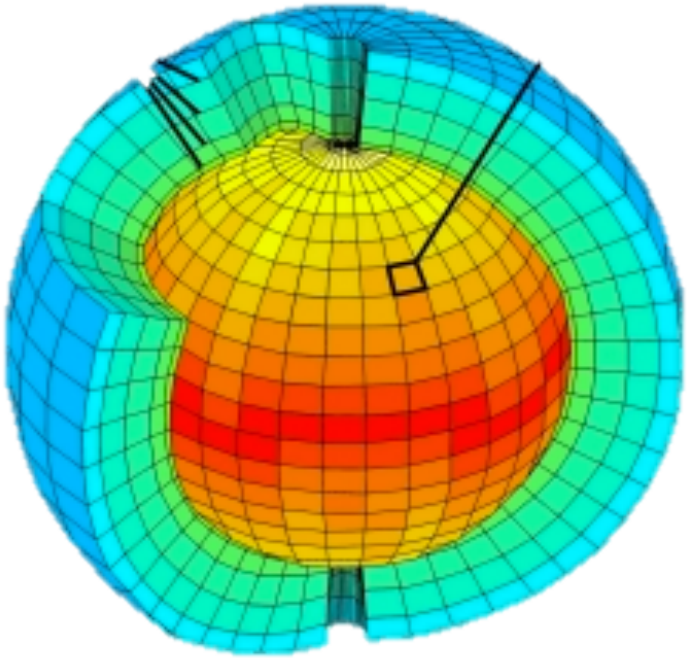
- Paralléliser un traitement nécessite :
  - La décomposition du traitement en sous traitements suffisamment indépendants.
    - Les processeurs doivent peu se gêner et peu s'attendre
  - L'organisation efficace du travail
    - Le travail doit être partagé équitablement
- Importance cruciale de la granularité
  - Trop petite => synchronisation très fréquente
  - Trop grande => iniquité de la répartition du travail
- Et donc :
  - concevoir des algorithmes et des structures de données pour adapter le problème à la machine.

Intérêts du calcul intensif

- Gagner du temps
  - Gagner en précision (fusion nucléaire)
  - Faire plus d'expériences (météo)
  - Aller plus vite que les autres (finance)
- Traiter des problèmes importants
  - dépasser la limitation mémoire d'une machine (cosmologie)
- Programmer plus simplement une simulation
  - Objets parallèles (Simuler un avion)
  - Analyser des phénomènes multi-physiques (Météo : interaction air / terre – air / Océan)

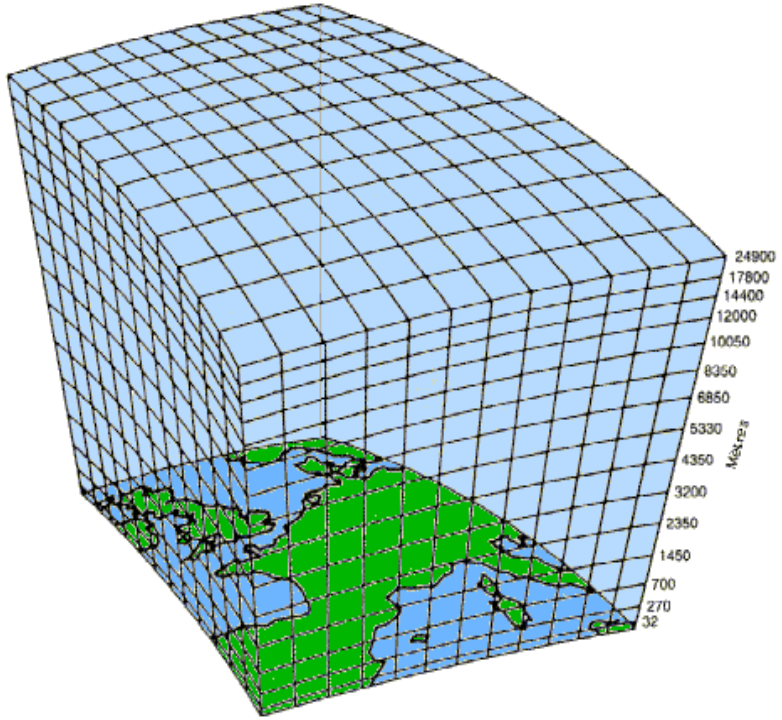
Modélisation météorologique

Calculer humidité, pression, température et vitesse du vent en fonction de x,y,z et t



- Discrétiser l'espace
  - Ex.: 1 point pour 2km^3 sur 20 km d'atmosphère => 5.10^9 points
- Initialiser le modèle
- Faire évoluer le modèle
  - Discrétiser le temps
    - Ex.: Calculer par pas de 60s
  - Résoudre pour chaque maille un système d'équations
    - calculer l'état suivant d'une maille en fonction de son voisinage
    - Coût : 100 FLOP / maille

Modélisation météorologique



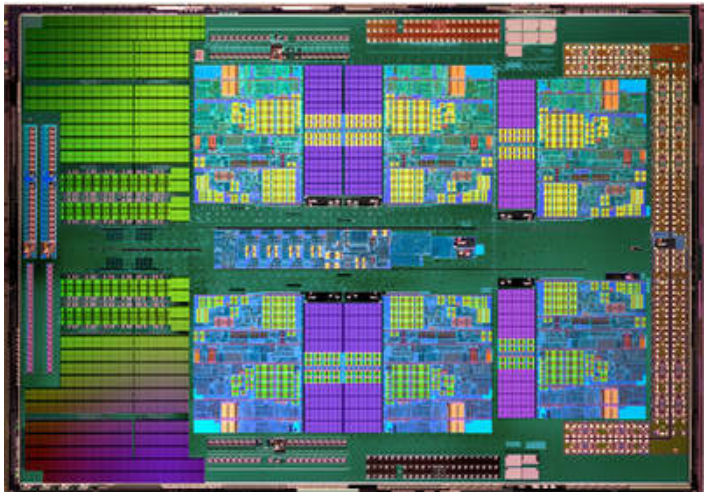
1 pas de calcul coûte *5.10^11 flop*

Besoin d'une simulation:

- Temps réel : 1 pas en 60s
  - *5.10^11 / 60 = 8 Gflop/s*
- Prévision : 7 jours en 24h
  - 7 * 8 = 56 Gflop/s
- climatologie : 50 ans en 30 jours
  - 4,8 Tflop/s

Modélisation météorologique

- Puissance *théorique* des processeurs actuels :
  - phenom II X6 : 79 GFlop/s (6 x 4 FP x 3.3 GHz)
  - core i7 980 : 79 GFlop/s (6 x 4 FP x 3.3 GHz)



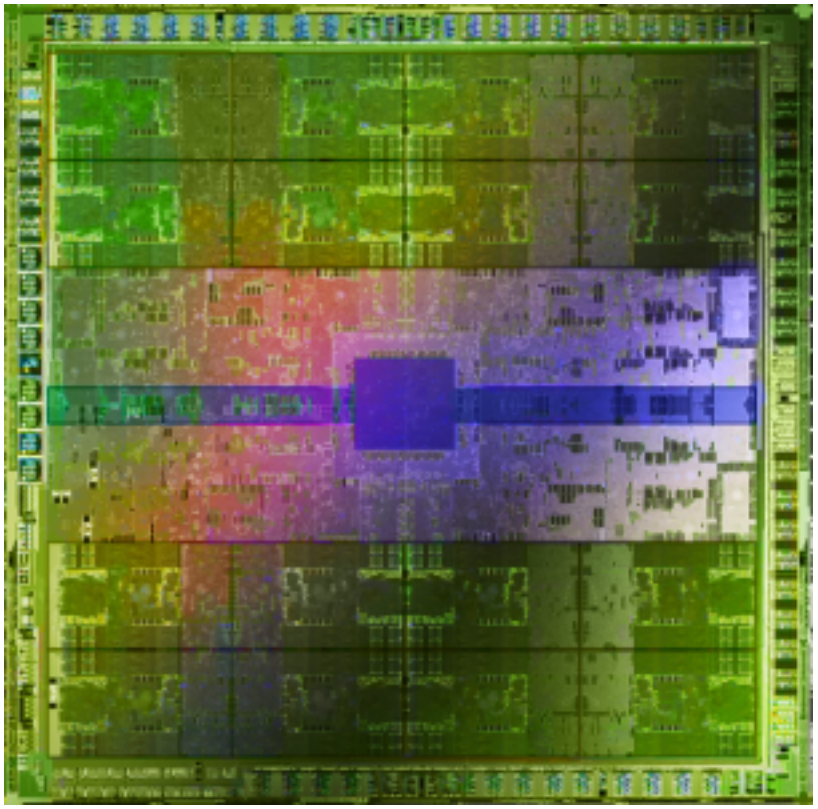
1 pas de calcul coûte *5.10^11 flop*

Besoin d'une simulation:

- Temps réel : 1 pas en 60s
  - *5.10^11 / 60 = 8 Gflop/s = 1 coeur*
- Prévision : 7 jours en 24h
  - 7 * 8 = 56 Gflop/s = 1 processeur
- climatologie : 50 ans en 30 jours
  - 4,8 Tflop/s = 60 processeurs

Modélisation météorologique

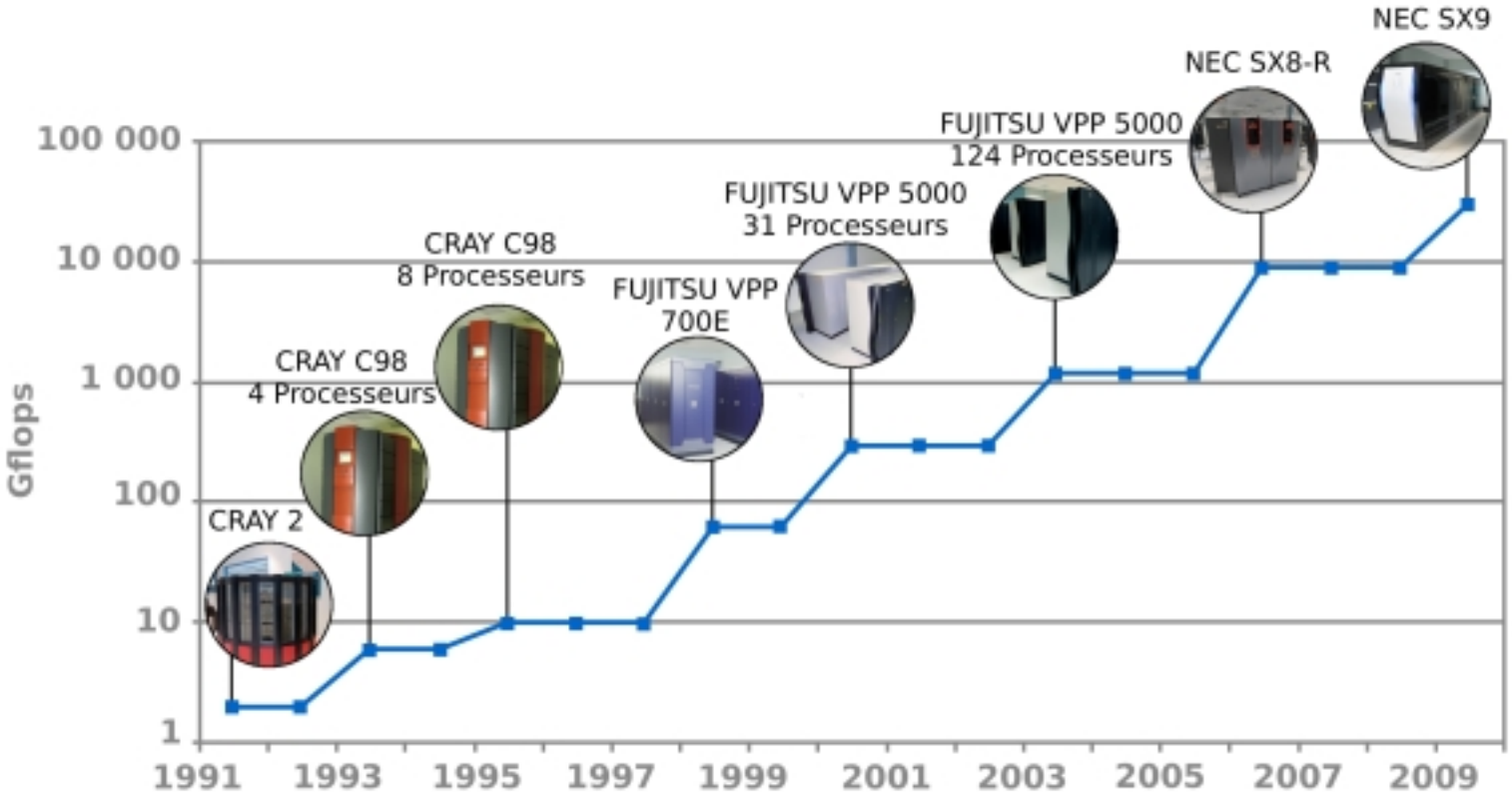
- Puissance théorique des processeurs actuel 80 Gflop/s



- Processeur Cell : 205Gflop/s
- Puissance théorique des cartes graphiques :
  - Nvidia Fermi : 512 Gflop/s (fused mult. & add)
  - ATI RD 6970 : 683 Gflop/s

- Climatologie : 4,8 Tflop/s = 8 GPU pendant 30 jours
- 1 PC sur vitaminés
  = 1 GPU + 2 processeurs
  = 670 Gflop/s
  = 12 prévisions météo / 24h

Plateforme Meteo France



« L'établissement dispose d'un centre de calcul à Toulouse 40 000 fois plus puissant qu'il y a 20 ans et qui traite 20 milliards d'opérations par seconde. Mais pour réduire la maille d'un modèle d'un facteur 2 il faut multiplier par 16 la puissance de calcul. Il faudra donc franchir très bientôt une nouvelle étape pour pouvoir bénéficier au plan opérationnel des avancées de la recherche en prévision numérique. »

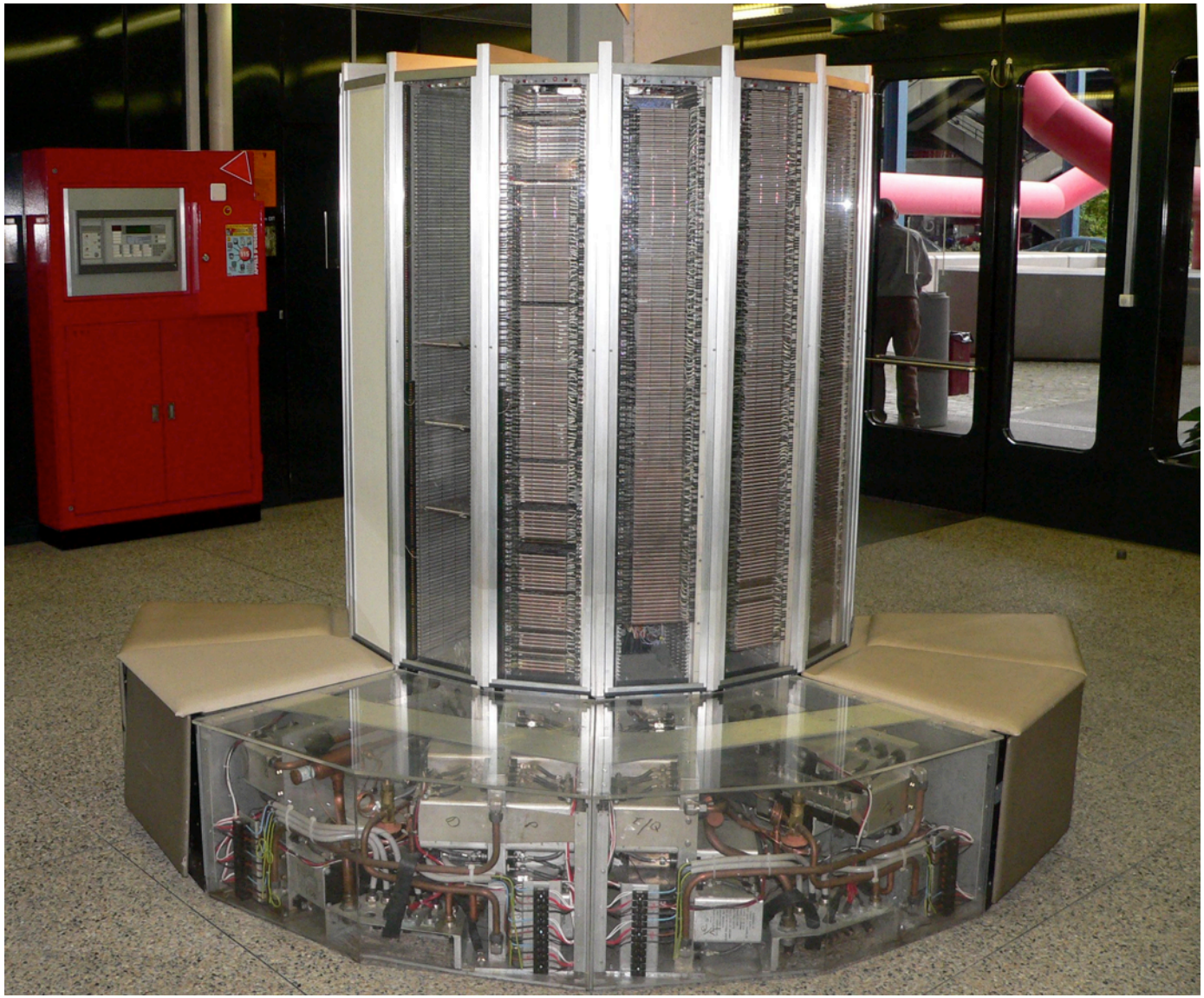
Le TOP 500


TOP500
SUPERCOMPUTER SITES
Performance Development
100 PFlops
10 PFlops
1 PFlops
100 TFlops
10 TFlops
1 TFlops
100 GFlops
10 GFlops
1 GFlops
100 MFlops
Performance
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
43673.10 TF
2566.00 TF
31112 GF
1.12 TF
59.7 GF
0.42 GF
#1
#500
Sum
19/11/2010
http://www.top500.org/

TOP500

Rank	Site	Manufacturer	Computer	Country	Year	Cores	RMax	RPeak
1	National Supercomputing Center in Tianjin	NUDT	**NUDT TH MPP, X5670 2.93Ghz 6C, NVIDIA GPU, FT-1000 8C**	China	2010	**186 368**	**2 566 000**	**4 701 000**
2	DOE/SC/Oak Ridge National Laboratory	Cray Inc.	**Cray XT5-HE Opteron 6-core 2.6 GHz**	United States	2009	**224 162**	**1 759 000**	**2 331 000**
3	National Supercomputing Centre in Shenzhen (NSCS)	Dawning	**Dawning TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU**	China	2010	**120 640**	**1 271 000**	**2 984 300**
4	GSIC Center, Tokyo Institute of Technology	NEC/HP	**HP ProLiant G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows**	Japan	2010	**73 278**	**1 192 000**	**2 287 630**
5	DOE/SC/LBNL/NERSC	Cray Inc.	**Cray XE6 12-core 2.1 GHz**	United States	2010	**153 408**	**1 054 000**	**1 288 630**
6	Commissariat a l'Energie Atomique (CEA)	Bull SA	**Bull bullx super-node S6010/ S6030**	France	2010	**138 368**	**1 050 000**	**1 254 550**
7	DOE/NNSA/LANL	IBM	**PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband**	United States	2009	**122 400**	**1 042 000**	**1 375 780**
8	National Institute for Computational Sciences/ University of Tennessee	Cray Inc.	**Cray XT5-HE Opteron 6-core 2.6 GHz**	United States	2009	**98 928**	**831 700**	**1 028 850**
9	Forschungszentrum Juelich (FZJ)	IBM	**Blue Gene/P Solution**	Germany	2009	**294 912**	**825 500**	**1 002 700**
10	DOE/NNSA/LANL/SNL	Cray Inc.	**Cray XE6 8-core 2.4 GHz**	United States	2010	**107 152**	**816 600**	**1 028 660**

Cray one 136 MFLOPS en 1976



TH 2 566 000 MFLOPS en 2010

Jaguar

Roadrunner (Cell + opteron)

Jügene (BlueGene)

Tera 100

Plateforme de calcul Peer to Peer

OS Type	Native TFLOPS*	x86 TFLOPS*	Active CPUs	Total CPUs
Windows	304,00	304,00	292 089,00	3 579 988,00
Mac OS X/PowerPC	4,00	4,00	4 607,00	142 661,00
Mac OS X/Intel	108,00	108,00	26 400,00	138 520,00
Linux	400,00	400,00	148 225,00	578 898,00
ATI GPU	752,00	793,00	5 298,00	142 556,00
NVIDIA GPU	3 041,00	6 417,00	19 123,00	229 696,00
PLAYSTATION®3	895,00	1 888,00	31 735,00	1 058 835,00
Total	5 504,00	9 914,00	509 077,00	5 871 154,00

Est-il facile d'obtenir des performances ?

- Il faut exploiter un processeur au mieux alors que :
  - Une application moyenne utilise un cœur à 20% de sa capacité
  - Une application idéale tel le produit de matrice utilise à :
    - 93% un cœur
    - 90% 4 cœurs
    - 25% un GPU
- En plus il faut extraire suffisamment de parallélisme.

Loi d'Amdahl

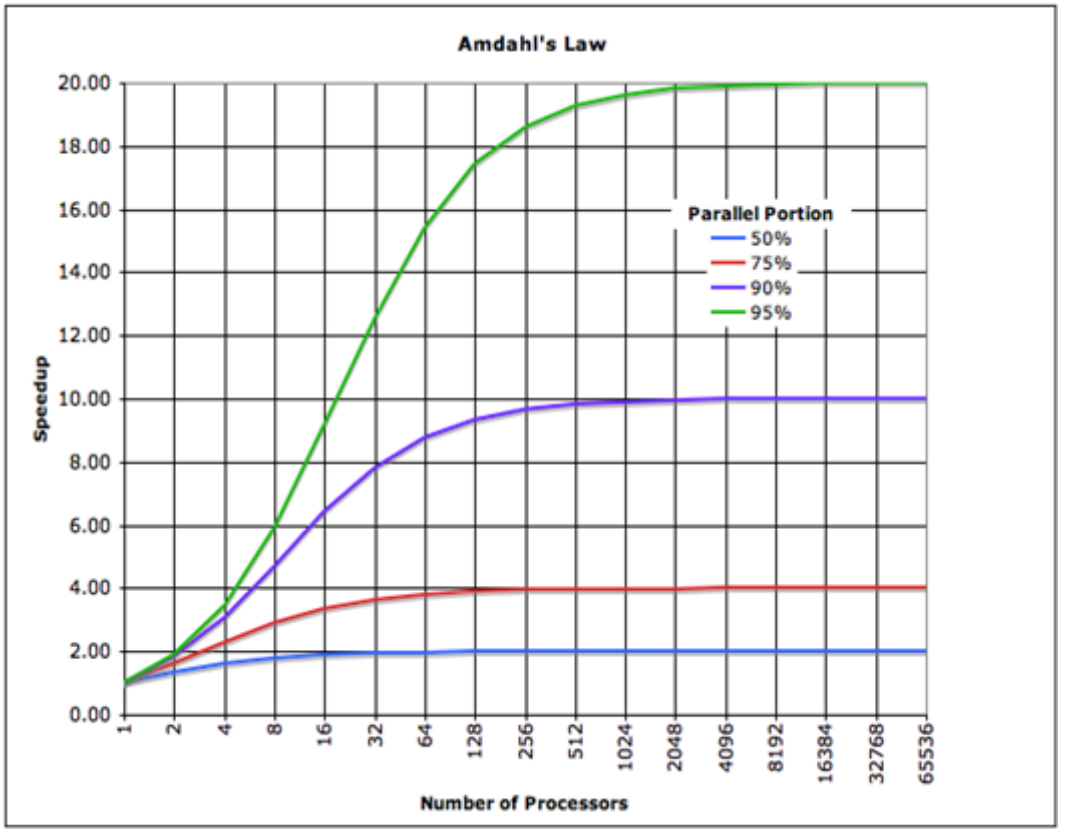
$$\text{accélération} = \text{speedup} = \frac{\text{temps du meilleur prog. séquentiel}}{\text{temps du prog. parallèle}}$$

Soit *S* la part séquentielle de l'application A pour une donnée d. L'accélération de l'exécution de A(d) sur une machine à p processeur est bornée par

$$speedup(p) \leq \frac{1}{S + \frac{1-S}{p}}$$

« si 1% de l'application est séquentielle on n'arrivera pas à aller 100 fois plus vite »

Loi d'Amdahl

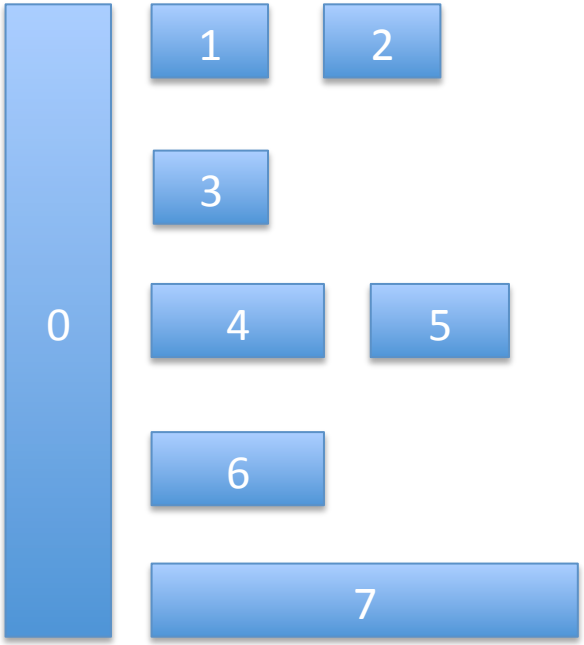

Amdahl's Law
Speedup
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
Parallel Portion
50%
75%
90%
95%
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
Number of Processors

Loi d'Amdhal

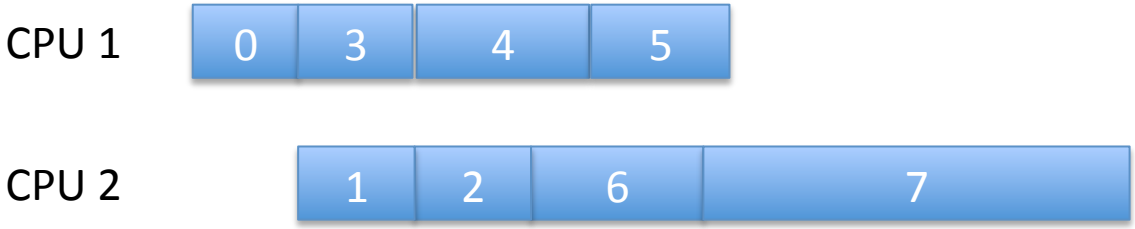
- La loi d'Amdahl n'est pas optimiste
  - elle raisonne en taille constante,
  - or augmenter le nombre de processeurs permet d'augmenter la taille du problème,
  - et la proportion de parallélisme peut augmenter comparativement à la partie séquentielle
- Cependant :
  - Elle oblige à faire la chasse aux partie séquentielles :
    - Synchronisation, exclusion mutuelle
    - Redondance de calcul
    - Initialisation et terminaison des flots de calcul
    - Déséquilibre de charge entre processeurs

Équilibrage de charge

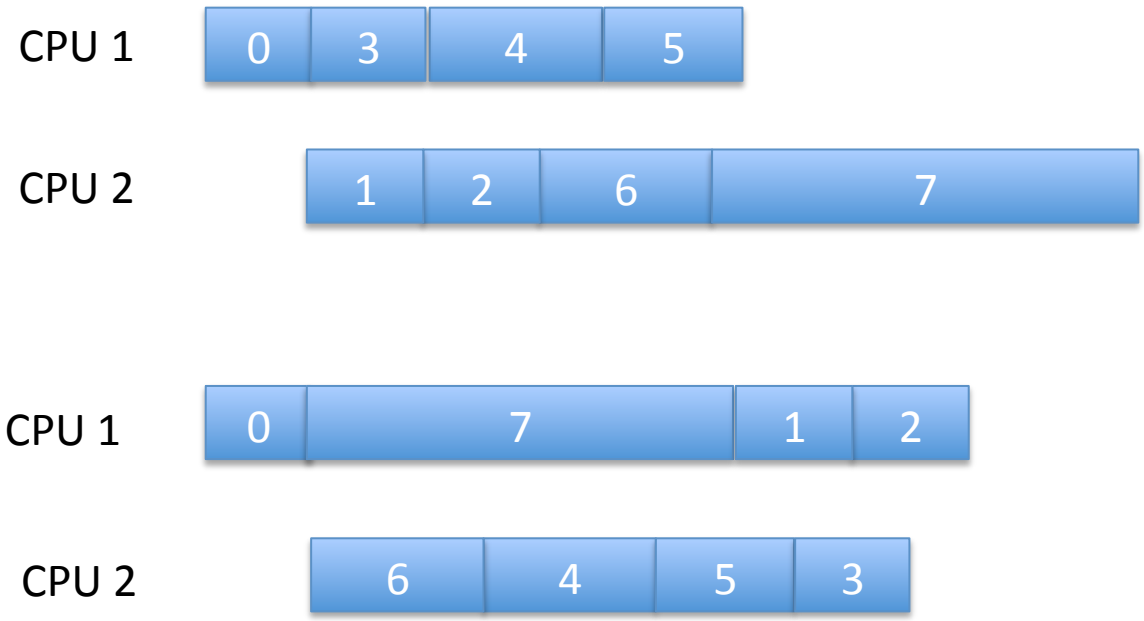
- Comment ordonnancer efficacement un ensemble de tâche sur un ensemble de processeur ?



Équilibrage de charge



Équilibrage de charge



Équilibrage de charge

- Problèmes réguliers
  - Calcul prévisible ne dépendant pas des résultats intermédiaires
    - Ex. Algèbre linéaire classique
    - Une distribution équitable *a priori* est possible
      - De façon statique, à la compilation
      - Juste avant l'exécution
    - Problème NP-Complet (analogue au *bin packing*)
- Problèmes irréguliers
    - Ex. calcul fractal, décomposition en facteur premier
    - Il faut rééquilibrer dynamiquement la charge au fur et à mesure