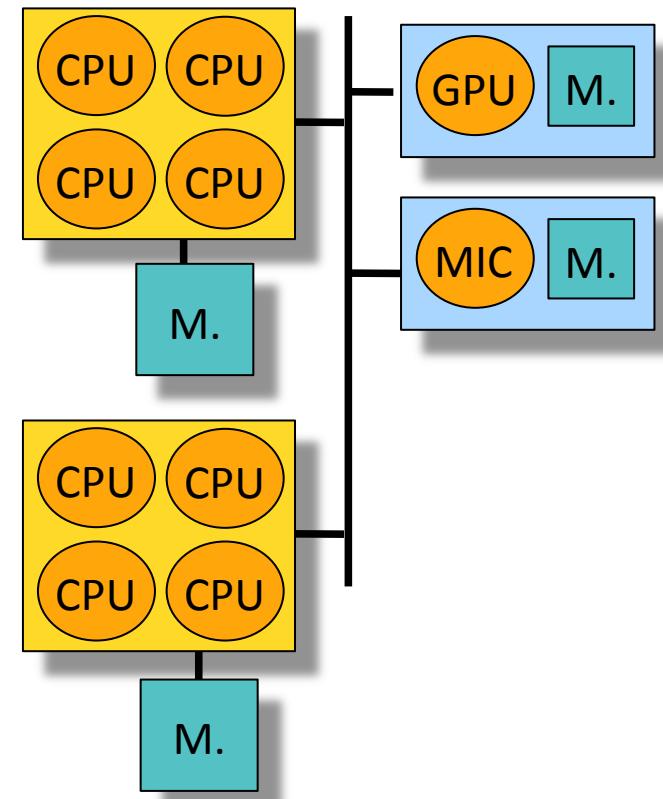


Plan du cours

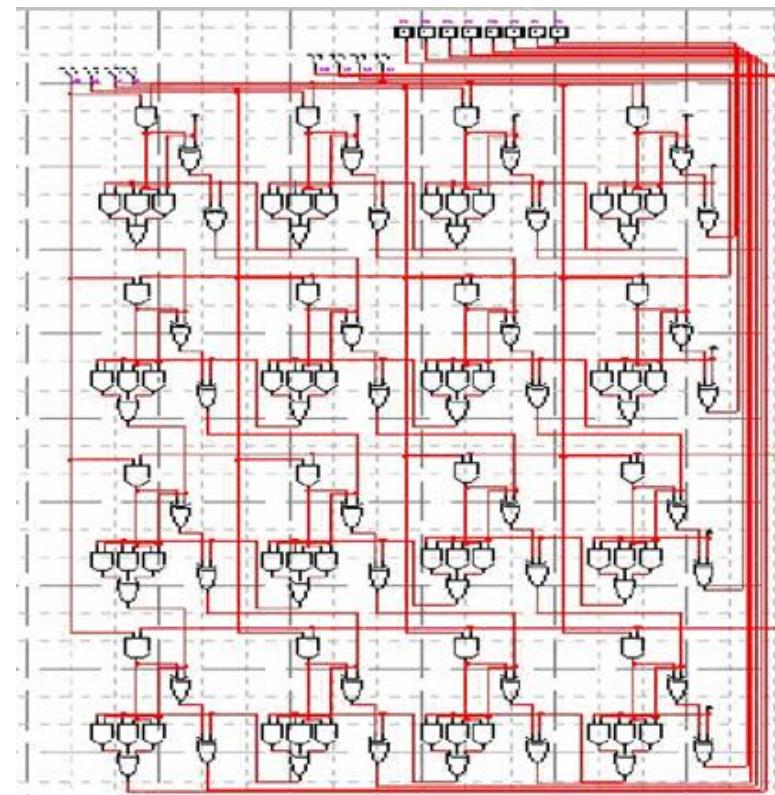
- Introduction (1 cours)
- Approche mémoire partagée (5 cours)
 - programmation (pthreads, OpenMP)
 - architecture (pipeline, cache, SMP, NUMA)
- Approche mémoire distribuée (2 cours)
 - programmation (MPI)
 - architectures (topologie, réseaux rapides)
- Calcul sur accélérateur (4 cours)
 - architecture des GPU / MIC
 - Programmation (OpenCL)



Où trouve-t-on du parallélisme dans nos ordinateurs ?

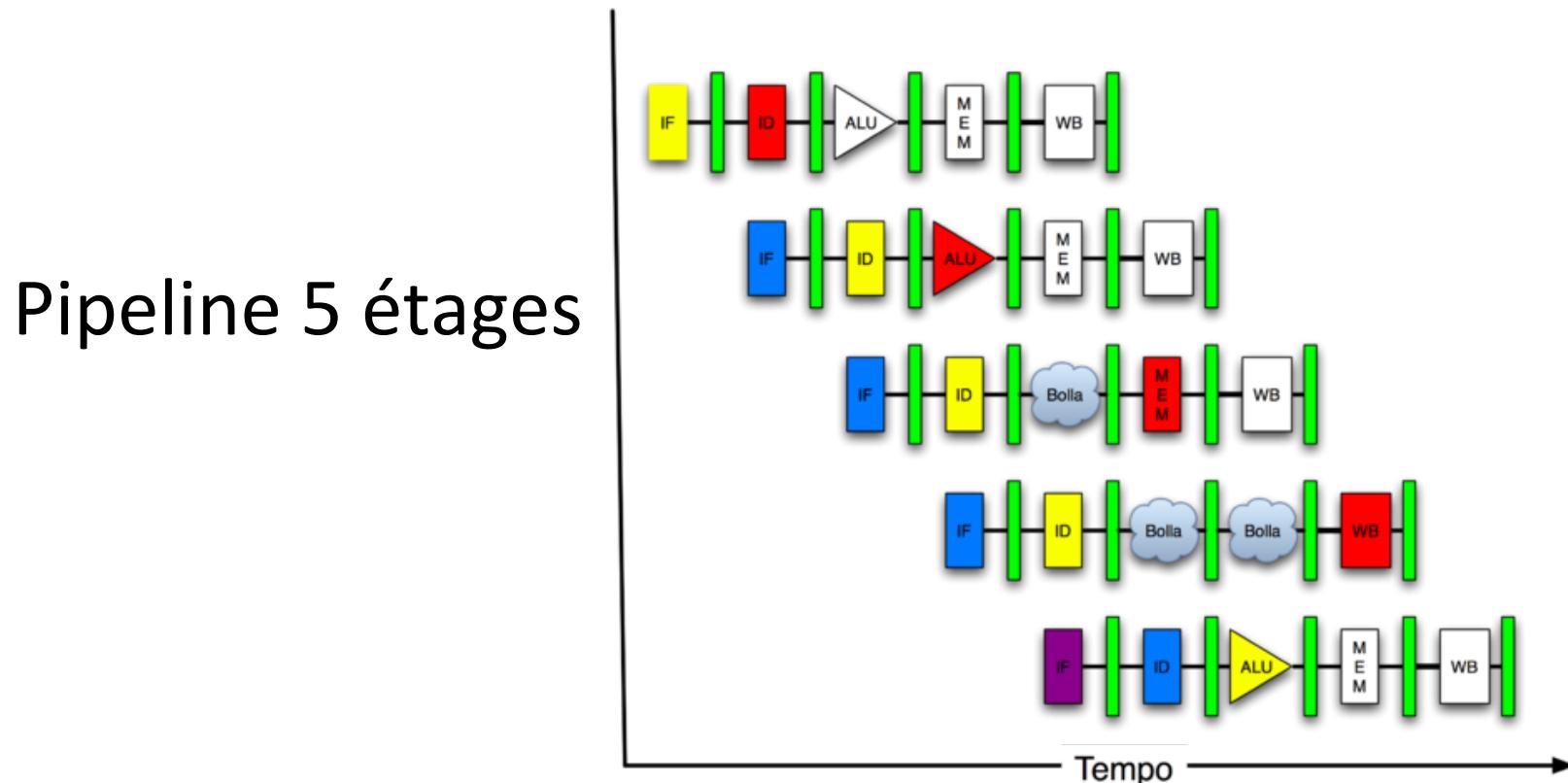
- Au niveau des circuits

Multiplicateur 4x4



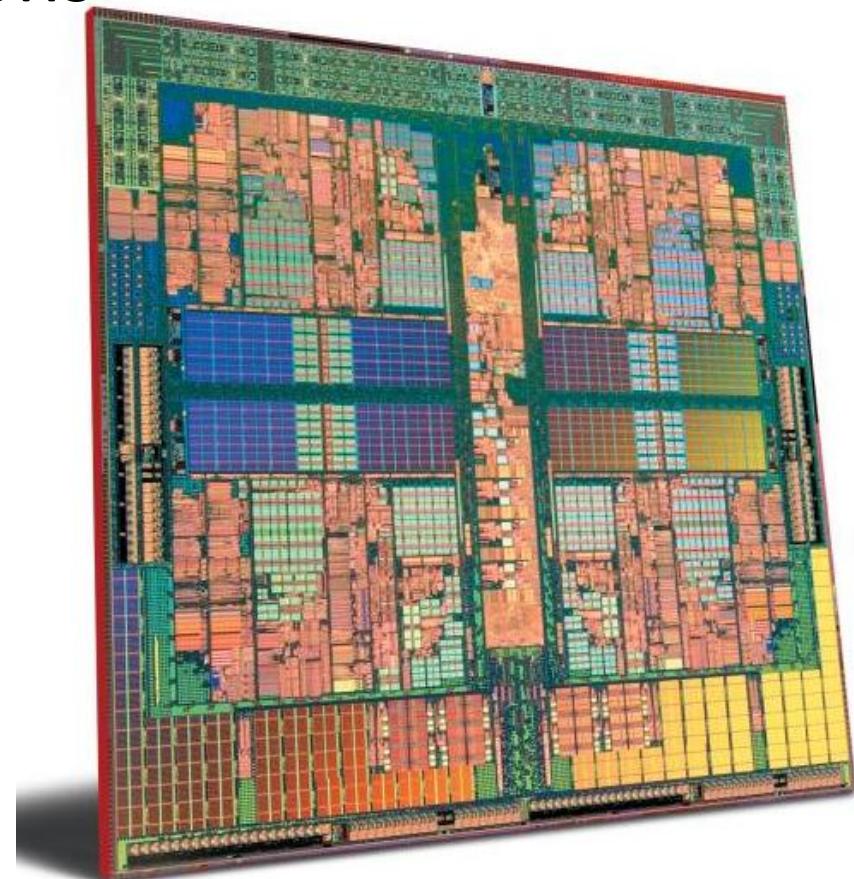
Où trouve-t-on du parallélisme

- Au niveau des circuits
- Au niveau des instructions



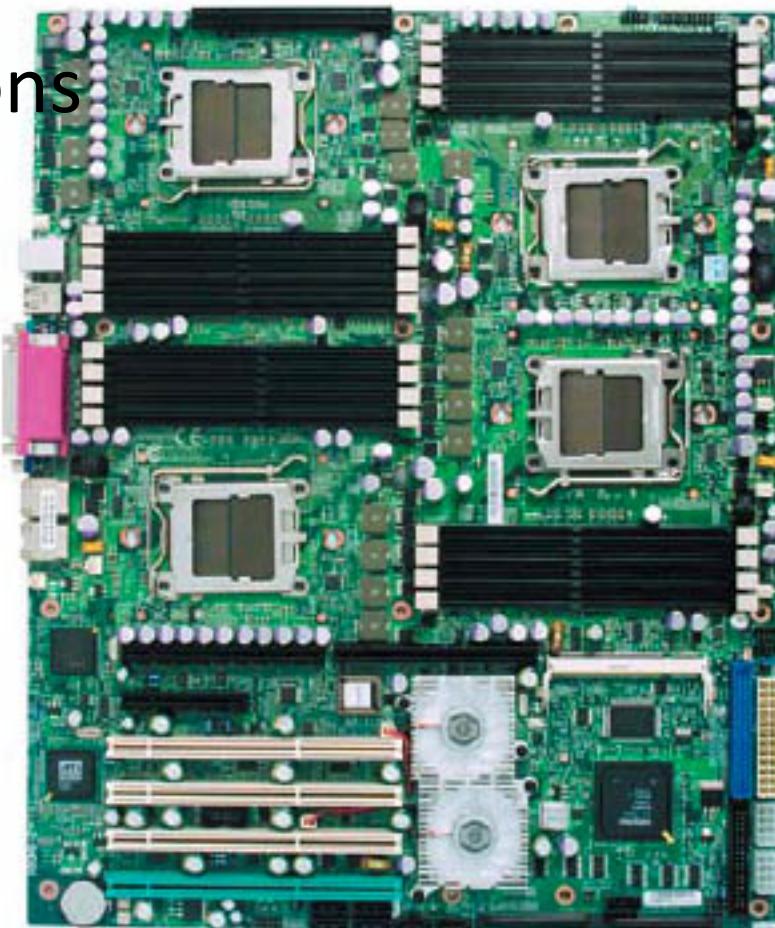
Où trouve-t-on du parallélisme

- Au niveau des circuits
- Au niveau des instructions
- Au niveau des threads
 - Multicœur



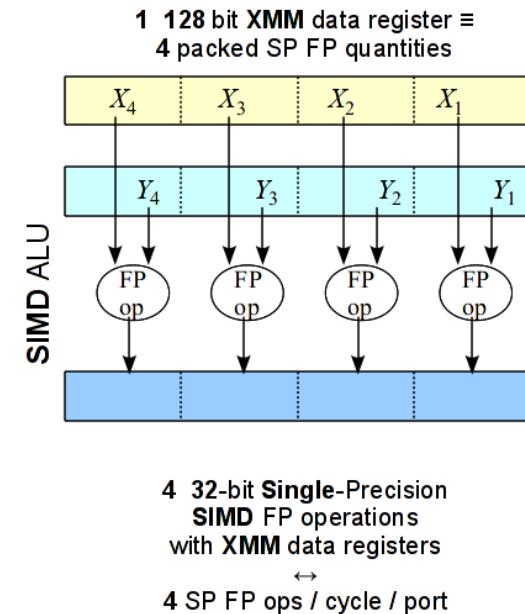
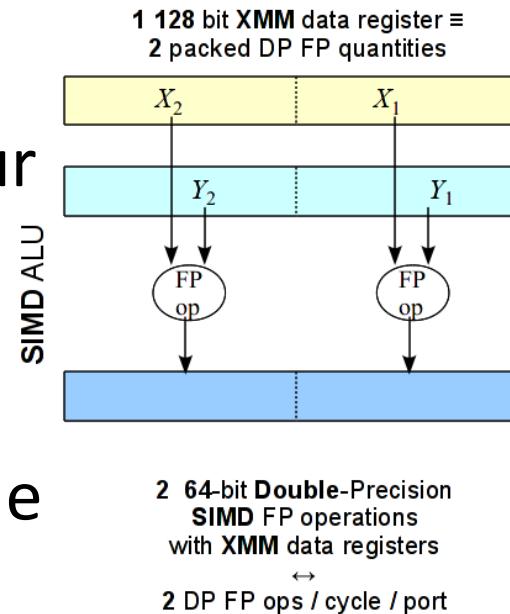
Où trouve-t-on du parallélisme

- Au niveau des circuits
- Au niveau des instructions
- Au niveau des threads
 - Multicœur
 - Multiprocesseur



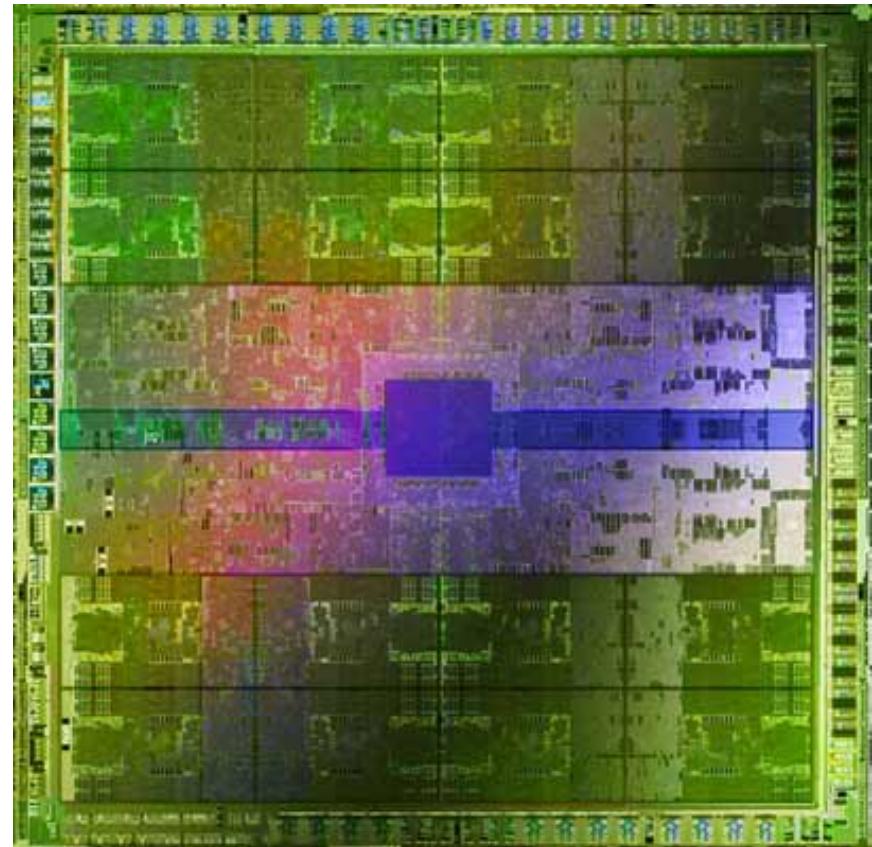
Où trouve-t-on du parallélisme

- Au niveau des circuits
- Au niveau des instructions
- Au niveau des threads
 - Multicœur
 - Multiprocesseur
- Au niveau des données
 - Unité vectorielle



Où trouve-t-on du parallélisme

- Au niveau des circuits
- Au niveau des instructions
- Au niveau des threads
 - Multicœur
 - Multiprocesseur
- Au niveau des données
 - Unité vectorielle
 - Accélérateur



Pourquoi concevoir des programmes parallèles ?

- Bénéficier des progrès technologiques
 - La fréquence de cadencement des processeurs n'augmente plus depuis 10 ans
 - Les techniques de parallélisation automatique ont une portée limitée
 - Le problème est avant tout algorithmique
- Gagner du temps
 - Gagner en précision (jeux, 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 (simulation)
- Programmer plus simplement :
 - Interface graphique
 - Objets parallèles
 - Analyser des phénomènes multi-physiques (ex. climatologie : interaction air / terre – air / Océan)

Modélisation météorologique

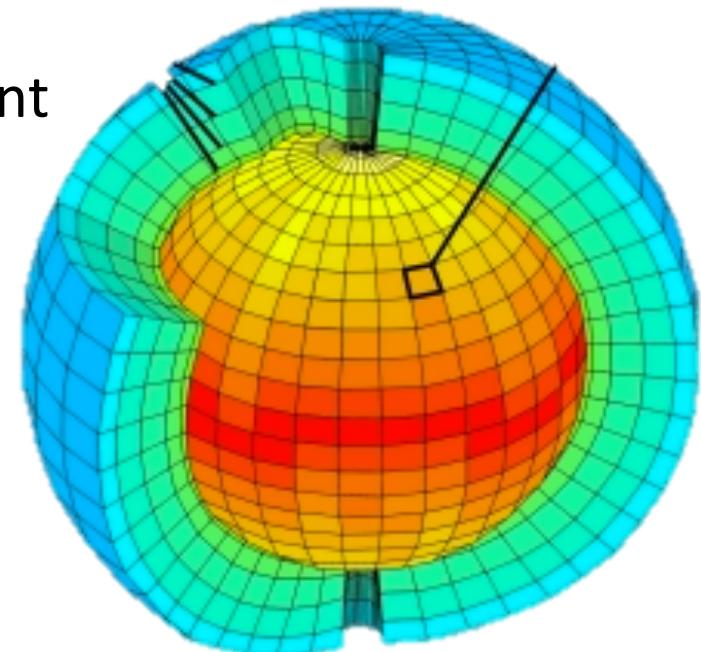
Objectif : calculer humidité, pression, température et vitesse du vent en fonction de x,y,z et t.

Résolution d'un système dynamique non linéaire (équation de Navier-Stokes)

=> impossible à résoudre formellement

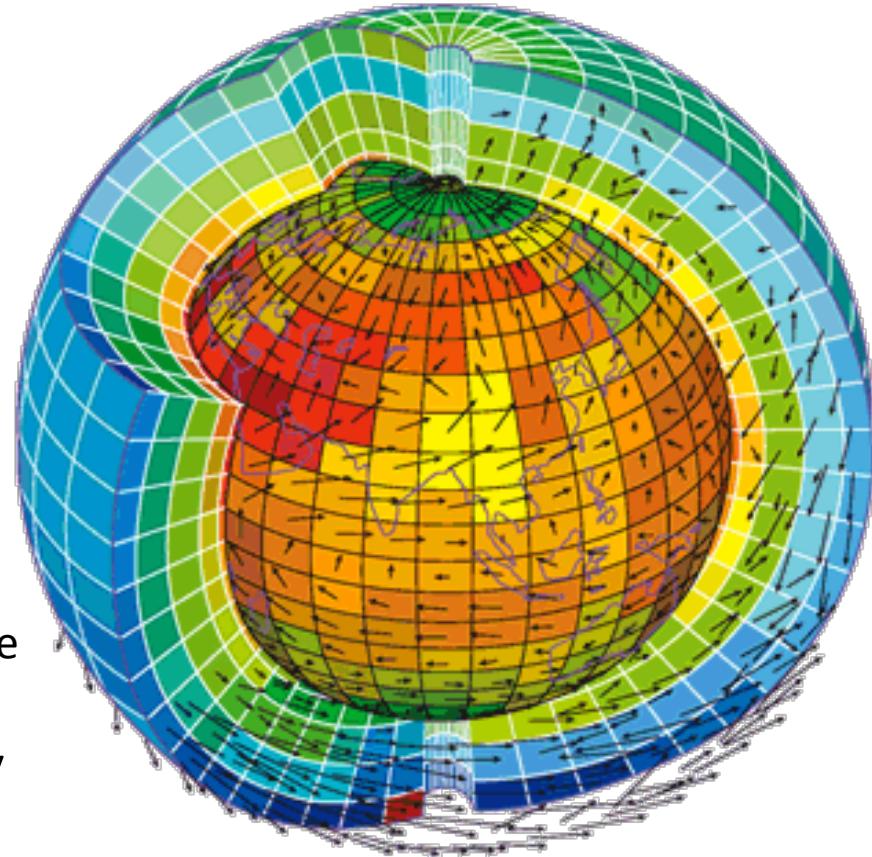
Simulation numérique déterministe :

- Discréteriser l'espace
 - Exemple imaginaire : 1 point pour 2km^3 sur 20 km d'atmosphère => 5.10^9 points
- Initialiser le modèle
 - Interpoler les valeurs manquantes



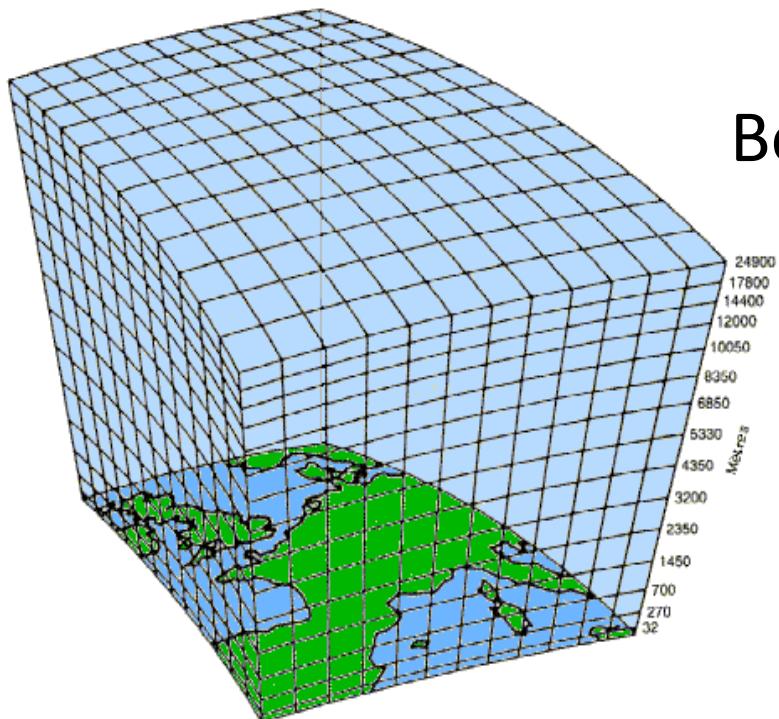
Modélisation météorologique

- Faire évoluer le modèle
 - Discréteriser le temps
 - Ex: calculer par pas de 60 secondes
 - Résoudre pour chaque maille le système d'équations
 - calculer l'état suivant d'une maille en fonction de son voisinage, de l'apport solaire, de la rotation de la terre
 - Coût imaginaire : 100 FLOP / maille



Modélisation météorologique

1 pas de calcul coûte $5.10^{11} flop$

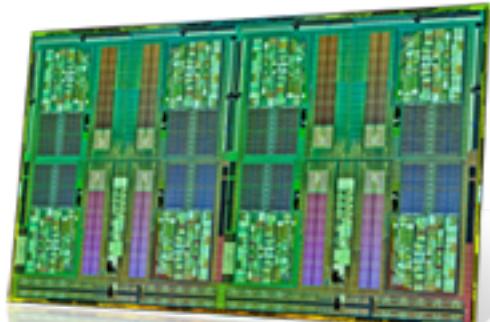


Besoin en calcul d'une simulation:

- Temps réel : 1 pas en 60s
 - $5.10^{11} / 60 = 8 \text{ Gflop/s}$
- Prévision : 7 jours en 1h
 - $7 * 24 * 8 = 1344 \text{ Gflop/s}$
- Climatologie (farfelue) : 50 ans en 30 jours
 - 4.8 Tflop/s

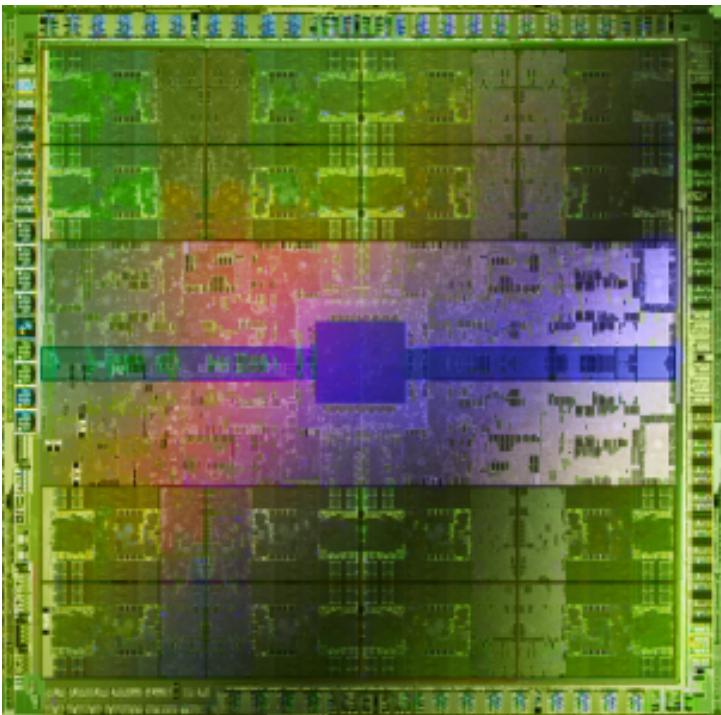
Modélisation météorologique

- Puissance *théorique* des processeurs actuels :
 - Opteron 6300 : 179 GFlop/s (16 c x 4 FP x 2.8 GHz) (boost 3.4 GHz)
 - Intel E7-8870 : 96 GFlop/s (10 c x 4 FP x 2.4 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 cœur
 - Prévision : 7 jours en 1h
 - $7 * 24 * 8 = 1344$ Gflop/s < 8 processeurs opteron (2 serveurs)
 - climatologie : 50 ans en 30 jours
 - 4,8 Tflop/s = 27 processeurs opteron (= 8 serveurs)



Modélisation météorologique

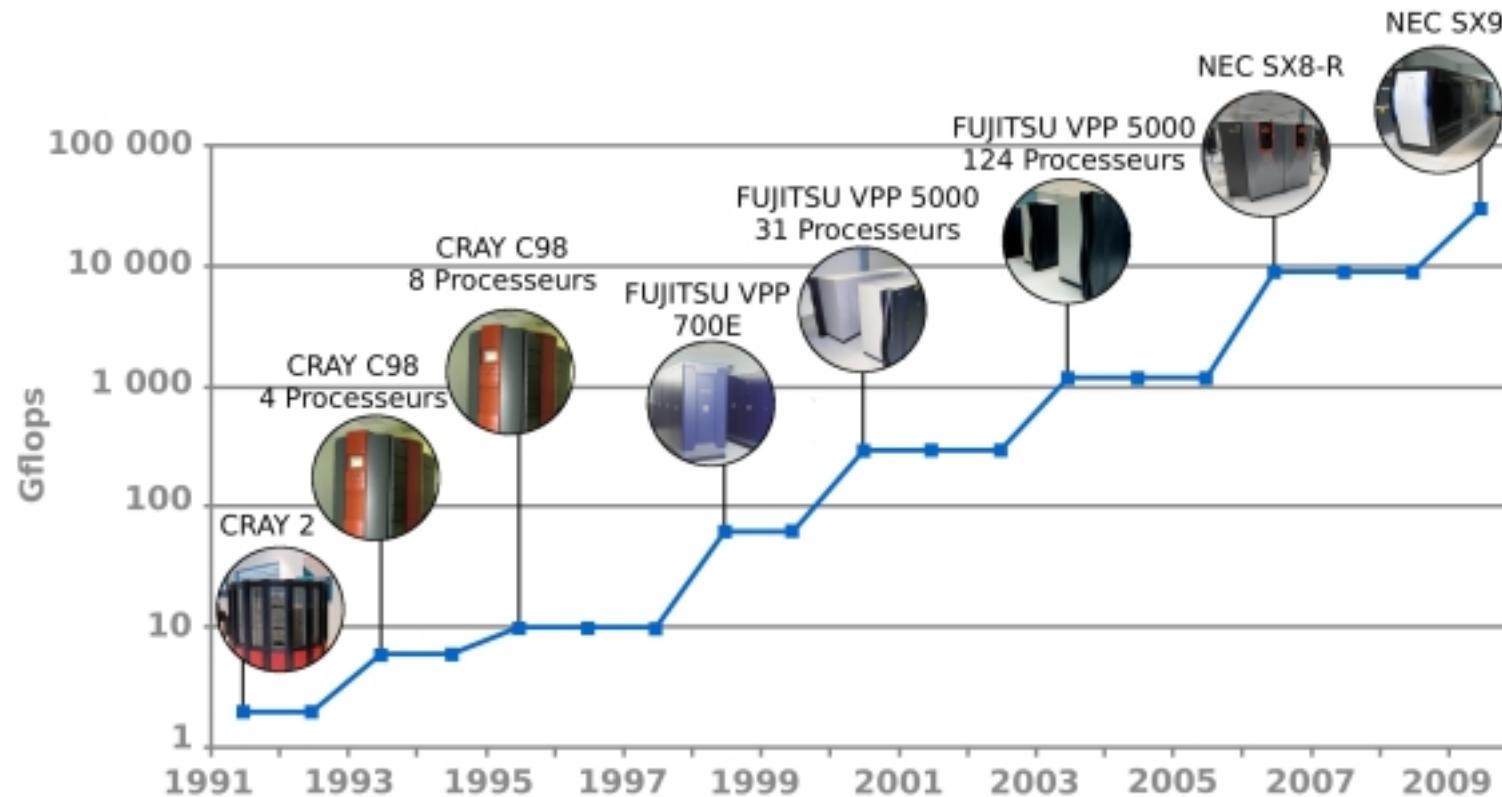
- Puissance théorique des processeurs actuels = 179 Gflop/s
- Processeur Cell : 205Gflop/s
- Puissance théorique des accélérateurs (64 bits) :
 - Nvidia kepler: 1,43 Tflop/s (fused mult. & add)
 - Intel MIC : 1 Tflop/s
- Prévision : 1,3Tflop/s = 1 GPU
- Climatologie : 4,8Tflop/s < 4 GPU
- Puissance d'un PC sur-vitaminé
4 GPU + 4 processeurs
= 6316 Gflop/s
= 4,6 prévisions météo / h



La simulation colle-t-elle bien à la réalité ?

- Difficultés liées au manque d'information :
 - Problème central : les erreurs sont elles dues au modèle physique (moteur du simulateur) ou aux approximations des conditions initiales ?
 - Métier du physicien / de l'ingénieur
 - Effet papillon fixe des limites
 - Dans un système *chaotique* un changement minime de configuration initiale peut entraîner un bouleversement à long terme.
 - Voir <http://www.chaos-math.org/fr/le-film>
- Météo :
 - Prévision déterministe = simulation sur 1 jeu de données
 - Tester une multitude de situations initiales et de paramètres de simulation (« prévision d'ensemble »)
 - Indice de confiance
 - Nécessite de gros moyens de calcul

Plateforme Meteo France



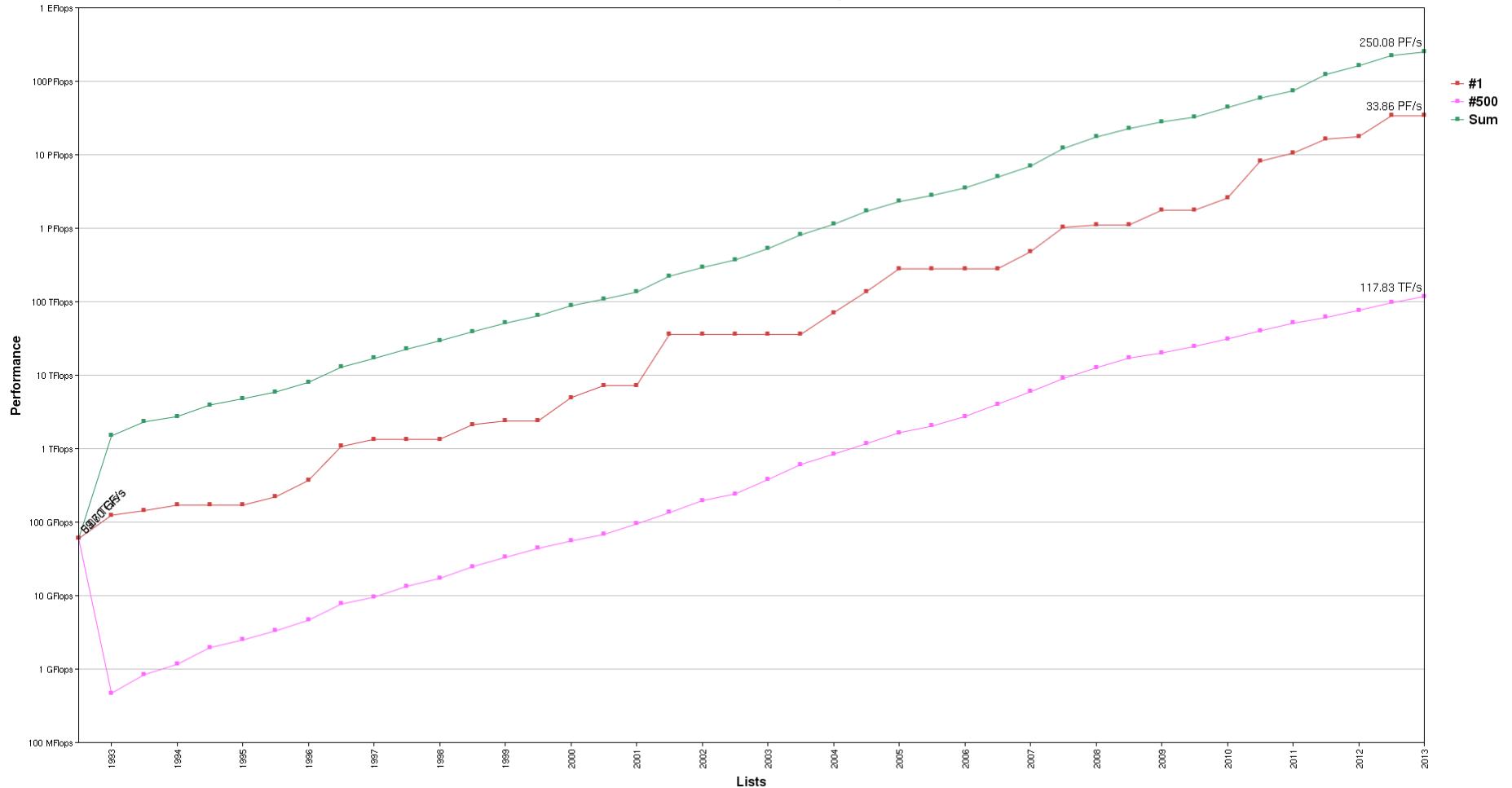
« ... pour réduire la maille d'un modèle d'un facteur 2 il faut multiplier par 16 la puissance de calcul. Il faudra donc franchir très bientôt une nouvelle étape pour pouvoir bénéficier au plan opérationnel des avancées de la recherche en prévision numérique. »
Janvier 2014 : 500 Tera FLOP/s – prévision 2016 : 5 Peta FLOP/s

TOP 500

Rank	Site	System		Rmax Cores	Rpeak (TFlop/s)	Power (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT		3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.		560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM		1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu		705,024	10,510.0	11,280.4	12,659.9
5	DOE/SC/Argonne National Laboratory United States	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM		786,432	8,586.6	10,066.3	3,945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.		115,984	6,271.0	7,788.9	2,325
7	Texas Advanced Computing Center/Univ. of Texas United States	Stampede - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell		462,462	5,168.1	8,520.1	4,510
8	Forschungszentrum Juelich (FZJ) Germany	JUQUEEN - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM		458,752	5,008.9	5,872.0	2,301
9	DOE/NNSA/LLNL United States	Vulcan - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM		393,216	4,293.3	5,033.2	1,972
10	Leibniz Rechenzentrum Germany	SuperMUC - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM		147,456	2,897.0	3,185.1	3,422.7

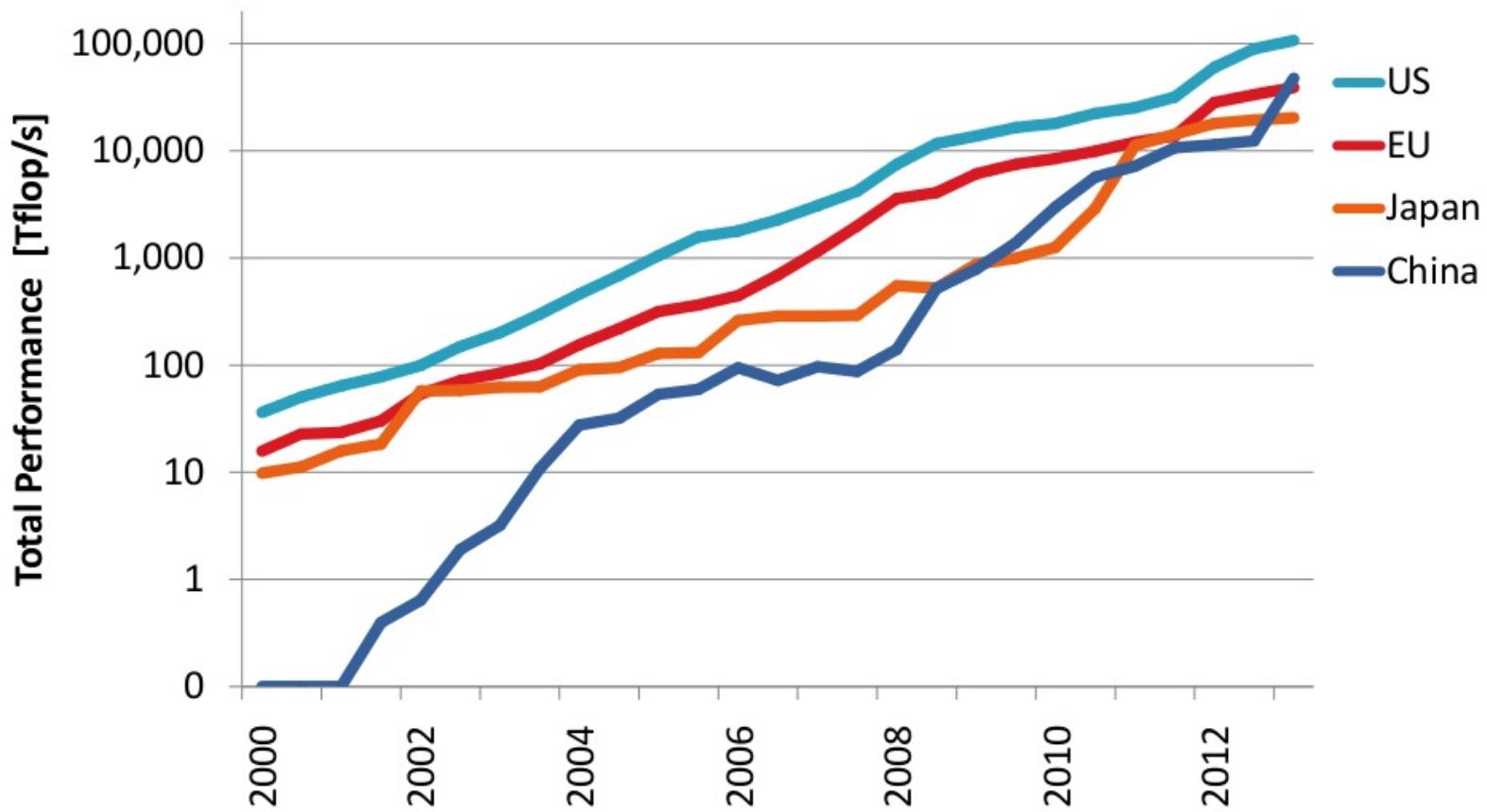
TOP 500

Performance Development



De 100 Gflop/s à 34 Pflop/s en 20 ans pour le #1
De 1 Gflop/s à 120 Tflop/s pour le #500

Compétition internationale

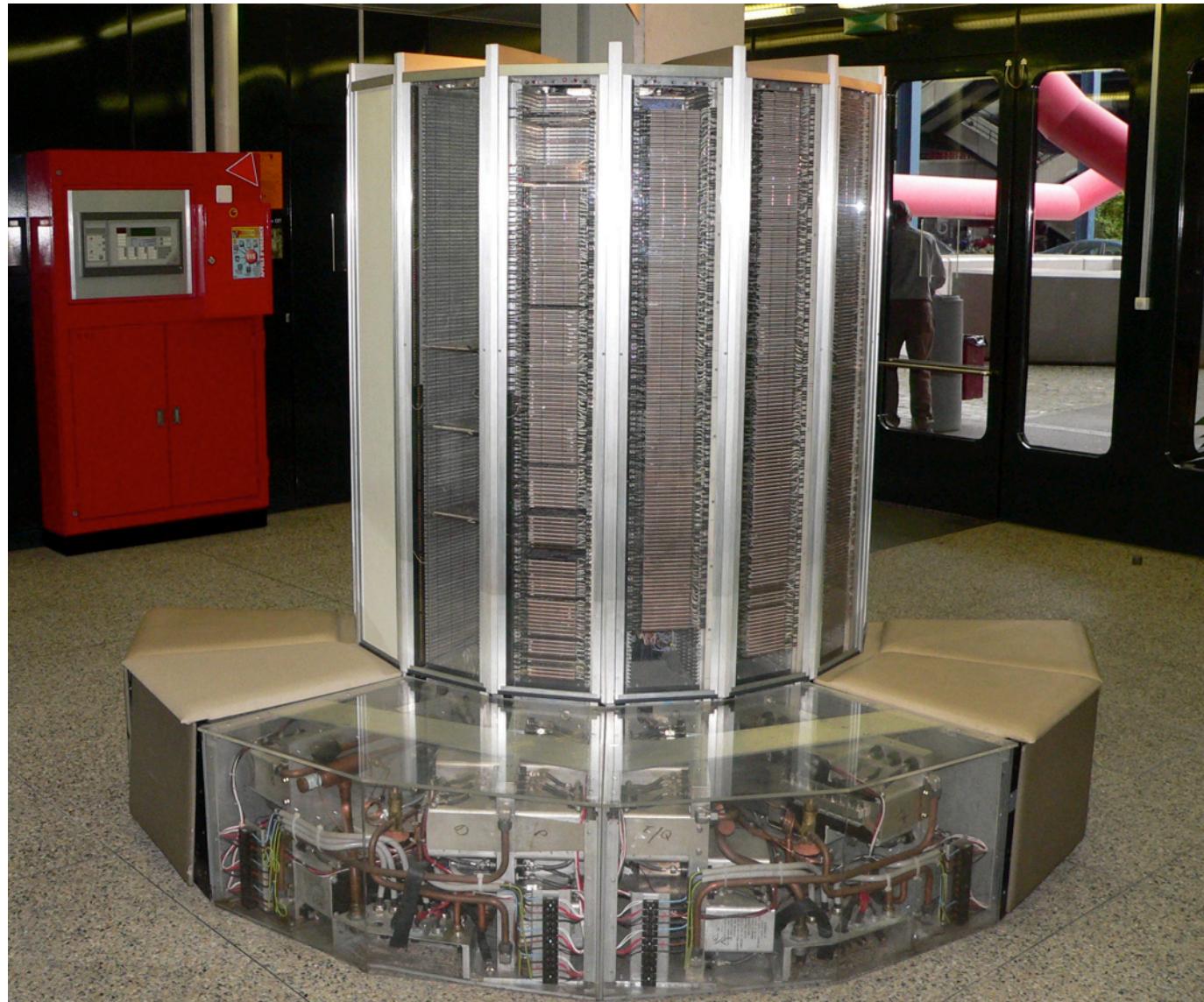


TOP 500 2013

Countries	Count
United States	264
China	63
Japan	28
United Kingdom	23
France	22
Germany	20
India	12
Canada	10

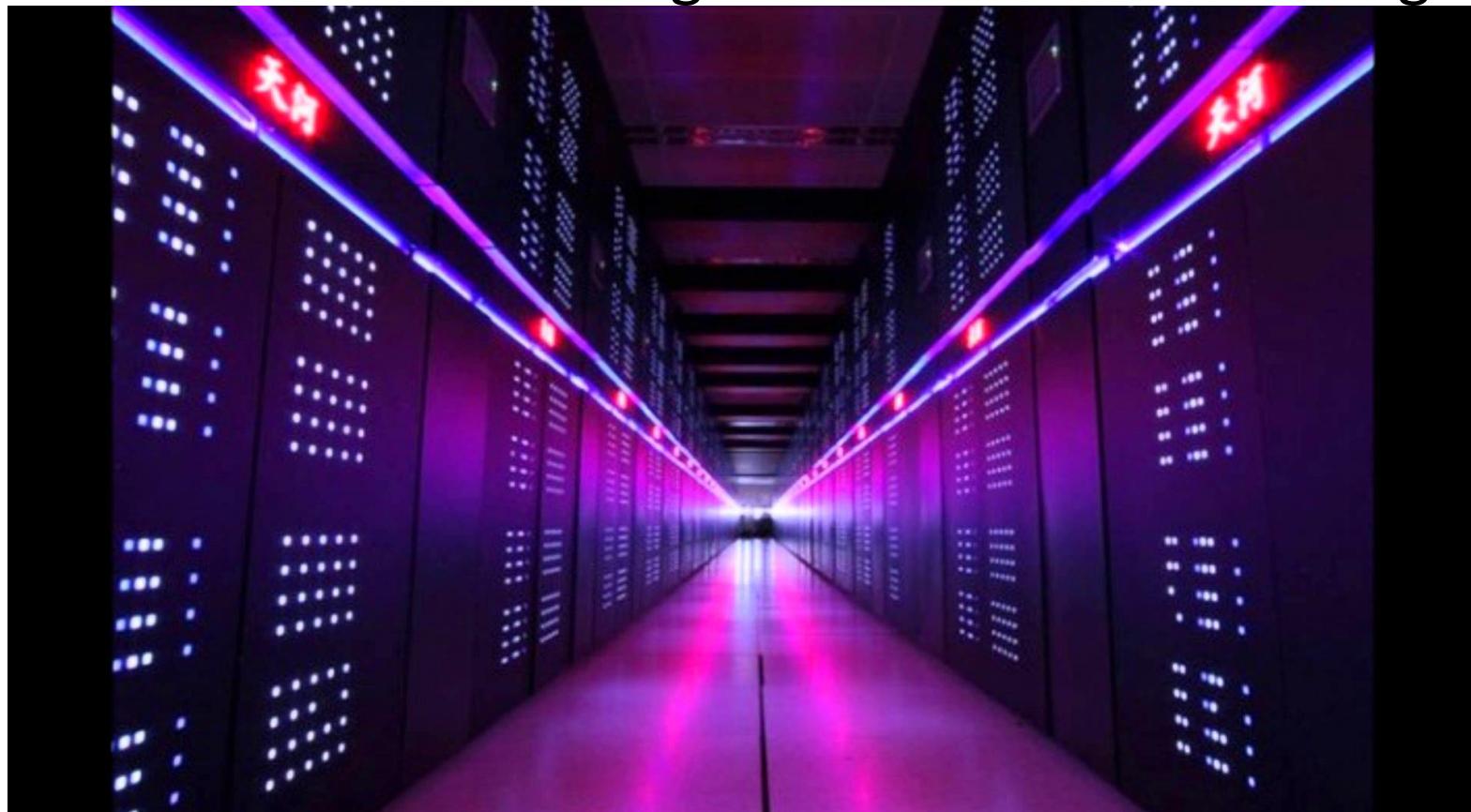
Russia	5
Australia	5
Italy	5
Switzerland	5
Korea, South	5
Sweden	5
Netherlands	3
Brazil	3
Norway	3
Saudi Arabia	3
Spain	2
Ireland	2
Israel	2
Finland	2
Hong Kong	2
Poland	2
Belgium	1
Austria	1
Denmark	1
Taiwan	1

Cray one 136 MFLOPS en 1976



天河二號 (rivière céleste)

Université de la technologie de la défense de Changha



33,86 10^{15} instructions par seconde

32 000 processeurs Ivy Bridge + 48 000 accélérateurs xeon phi

3 132 000 cœurs -

TITAN

17,6 Peta flops



Cray XK7 , Opteron 6274 16C NVIDIA K20x

Sequoia

16,3 Pflops



BlueGene/Q, Power BQC 16C 1.60 GHz, Custom

K computer

10 Pflops



World's No.1 Again on TOP500 List

Performance of over 10 Peta*flops



(*)10 Peta=10,000,000,000,000,000

K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect

Curie

1,3 Pflops

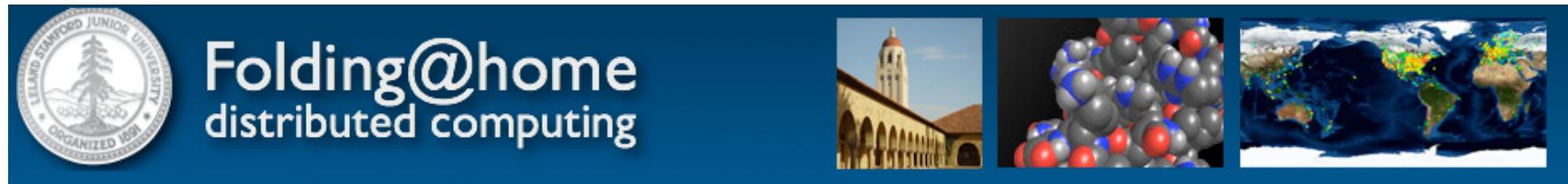


Bullx B510, Xeon E5-2680 8C 2.700GHz, Infiniband QDR

TOP français

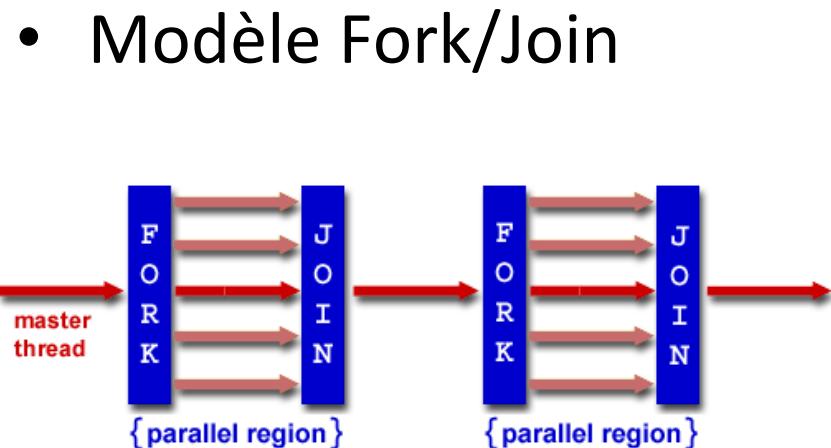
Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
14	Total Exploration Production France	Pangea - SGI ICE X, Xeon E5-2670 8C 2.600GHz, Infiniband FDR SGI	110,400	2,098.1	2,296.3	2,118
20	CEA/TGCC-GENCI France	Curie thin nodes - Bullx B510, Xeon E5-2680 8C 2.700GHz, Infiniband QDR Bull SA	77,184	1,359.0	1,667.2	2,251
29	Commissariat a l'Energie Atomique (CEA) France	Tera-100 - Bull bullx super-node S6010/S6030 Bull SA	138,368	1,050.0	1,254.5	4,590
45	CNRS/IDRIS-GENCI France	BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	65,536	715.6	838.9	328.8
46	EDF R&D France	Zumbrota - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	65,536	715.6	838.9	328.8
61	Météo France France	Bullx DLC B710 Blades, Intel Xeon E5 v2 12C 2.700GHz, Infiniband FDR Bull SA	25,800	500.3	557.3	401
91	EDF R&D France	Athos - iDataPlex DX360M4, Intel Xeon E5-2697v2 12C 2.700GHz, Infiniband FDR14 IBM	18,144	352.7	391.9	347.3
96	Commissariat a l'Energie Atomique (CEA)/CCRT France	airain - Bullx B510, Xeon E5-2680 8C 2.700GHz, Infiniband QDR Bull SA	18,144	346.1	391.9	500
151	ROMEO HPC Center - Champagne-Ardenne France	romeo - Bull R421-E3 Cluster, Intel Xeon E5-2650v2 8C 2.600GHz, Infiniband FDR, NVIDIA K20x Bull SA	5,720	254.9	384.1	81.4
161	Airbus France	HP POD - Cluster Platform 3000 BL260c G6, X5675 3.06 GHz, Infiniband Hewlett-Packard	24,192	243.9	296.1	

Plateforme de calcul Peer to Peer

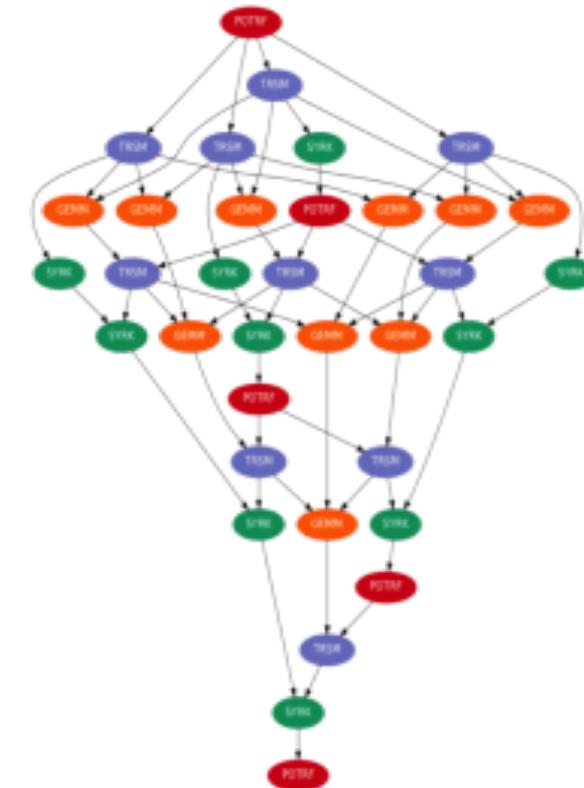


OS Type	Native TFLOPS*	x86 TFLOPS*	Active CPUs	Total CPUs
Windows	304,00	304,00	292 089,00	3 579 988,00
Mac OS X/PowerPC	4,00	4,00	4 607,00	142 661,00
Mac OS X/Intel	108,00	108,00	26 400,00	138 520,00
Linux	400,00	400,00	148 225,00	578 898,00
ATI GPU	752,00	793,00	5 298,00	142 556,00
NVIDIA GPU	3 041,00	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

Techniques de programmation parallèle



- Modèle task based



- Modèle Master / Slave
- Modèle SPMD (Single Program Multiple Data)

Est-il facile d'obtenir des performances ?

- Il faut exploiter un processeur au mieux, or :
 - Une application idéale (DGEMM : produit de matrices) exploite à :
 - 93% un cœur
 - 90% 4 cœurs
 - 80% un GPU
 - Mal programmé ce même produit de matrice exploitera 20% de la performance crête du processeur.
 - Savoir faire peu partagé :
 - Qui utilise gcc -O3 ?
 - Qui optimise les accès à la mémoire ?
- Il faut extraire suffisamment de parallélisme
 - Objectif : occuper toutes les unités de calcul

Loi d'Amdahl

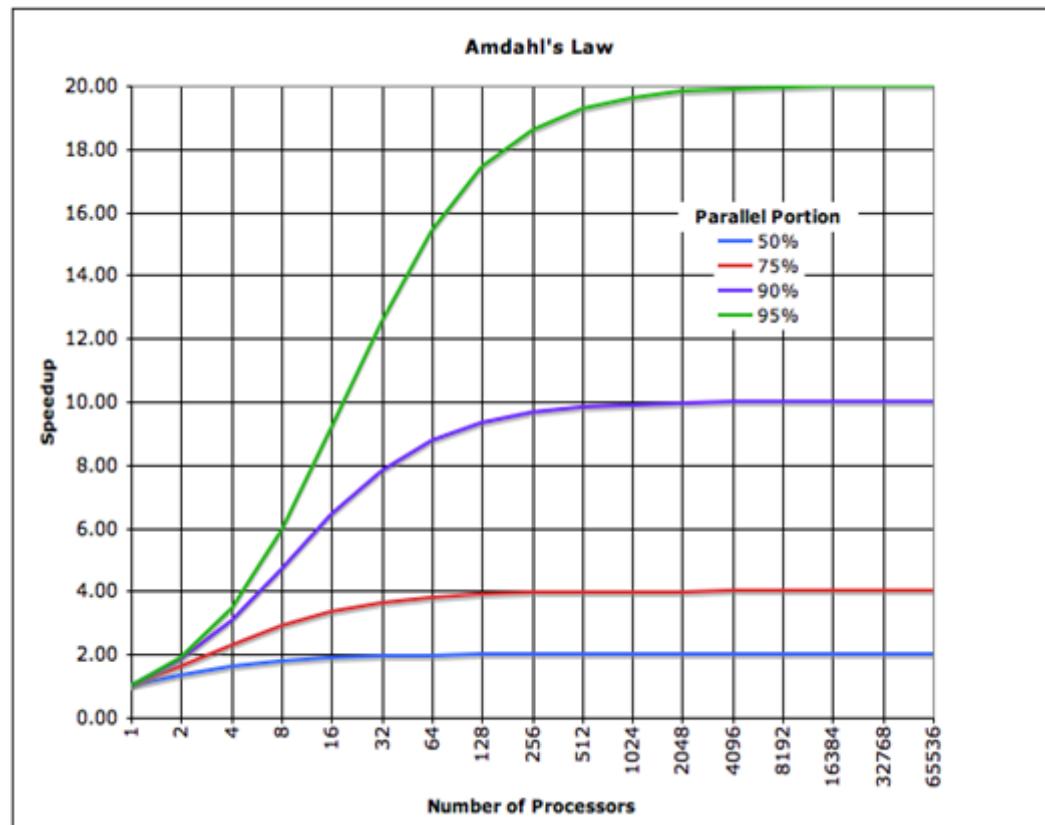
$$\text{accélération} = \text{speedup} = \frac{\text{temps du meilleur prog. séquentiel}}{\text{temps du prog. parallèle}}$$

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

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

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

Loi d'Amdahl



Loi de Gustafson

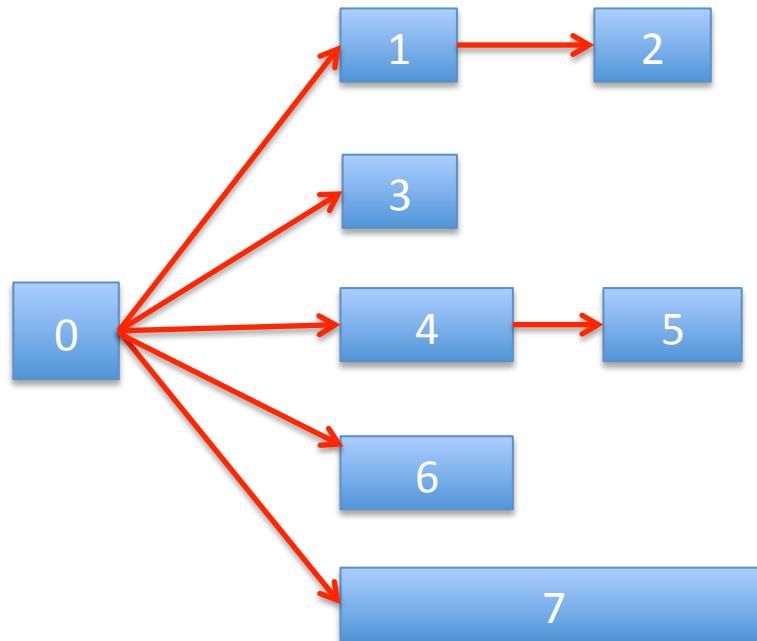
- On suppose ici que la taille du problème traité croît avec le nombre de processeurs et que le temps de traitement est, lui, constant :
 - Temps de calcul = $T_{seq} + T_{par}$
 - Temps du calcul en séquentiel $T_{seq} + p \cdot T_{par}$
 - Accélération en fonction de s (part séquentielle):
 - $s + (1-s) * p$
 - Pour $s = 0,1$ et $p = 1000$ on a 900,1 (contre >10 pour Amdahl)

Lois d'Amdhal & de Gustafson

- Taille du problème et accélération parallèle
 - La proportion de parallélisme peut augmenter avec la taille du problème.
 - Augmenter le nombre de processeurs permet d'augmenter la taille du problème (plus de mémoire).
- La loi de Gustafson est trop optimiste
 - Elle néglige les goulets d'étranglement (mémoire, réseau) et les synchronisations dues à l'augmentation du nombre de processeurs (*overhead*)
 - Mais elle motive l'utilisateur
- La loi d'Amdahl est pessimiste
 - Elle raisonne en taille constante
 - Mais elle motive le programmeur pour à faire la chasse aux partie séquentielles :
 - Synchronisation, exclusion mutuelle
 - Redondance de calcul
 - Déséquilibre de charge entre processeurs

Équilibrage de charge

- Comment ordonner efficacement un ensemble de tâches sur un ensemble de processeurs ?



Problème général NP-Complet

Équilibrage de charge



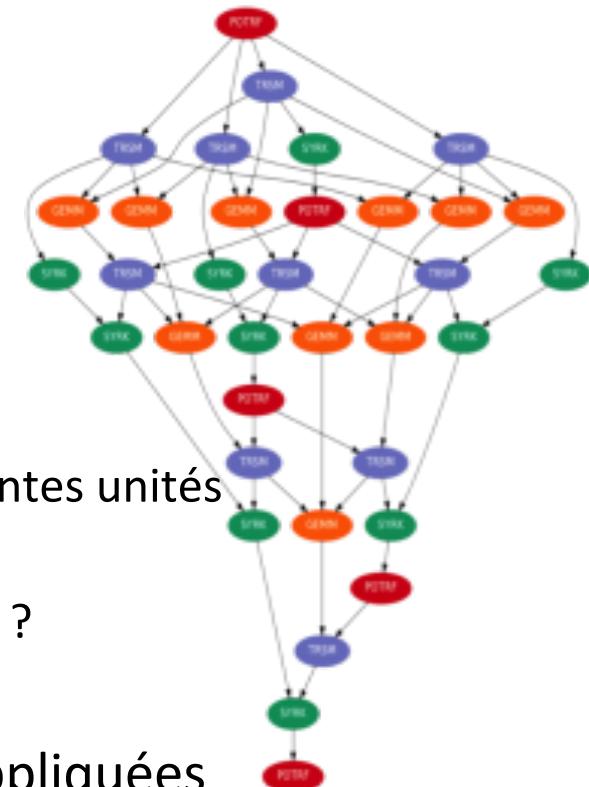
Équilibrage de charge



Problème général NP-Complet mais approximable

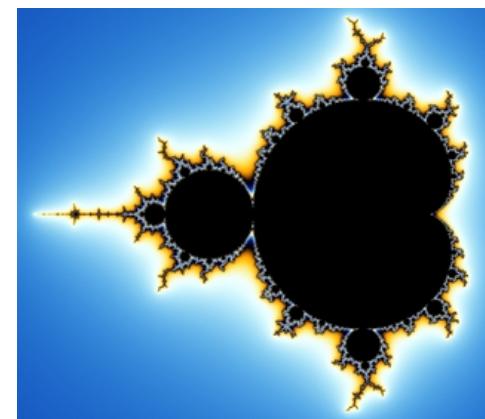
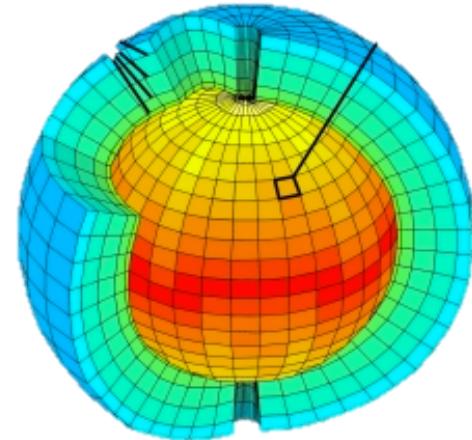
Équilibrage de charge

- Comment ordonner efficacement un ensemble de tâches sur un ensemble de processeurs ?
- Connait on à l'avance :
 - Le graphe ?
 - Le nombre de FLOP de chaque tâche ?
 - Les données utilisées par chaque tâche ?
 - La vitesse d'exécution des unités d'exécutions ?
 - Le coût de transfert d'un buffer entre les différentes unités de traitement ?
 - Le débit maximal des canaux de communication ?
 - ...
- Domaine actif de recherches théoriques & appliquées



Équilibrage de charge

- Problèmes réguliers
 - Charge de calcul prévisible ne dépendant pas des résultats intermédiaires
 - Ex. Algèbre linéaire classique, prévision météo,...
 - Une distribution équitable *a priori* est possible
 - De façon statique, à la compilation
 - Juste avant l'exécution
- Problèmes irréguliers
 - Ex. calcul fractal, décomposition en facteur premier
 - Il faut rééquilibrer dynamiquement la charge au fur et à mesure



En résumé

- 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
 - Limiter le travail redondant
- Importance cruciale de la taille des tâches
 - Trop petite => synchronisation très fréquente
 - Trop grande => iniquité de la répartition du travail

Il s'agit de concevoir des algorithmes et des structures de données pour adapter le problème à la machine.