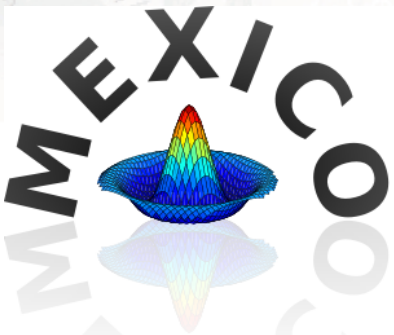
ÉCOLE DE PHYSIQUE
des HOUCHES

Groupement de Recherche
MASCOT-NUM
Méthodes d'Analyse Stochastique pour les COdes et Traitements NUMériques



Quantification d'incertitude et Tendances en HPC

Laurence Viry

École de Physique des Houches

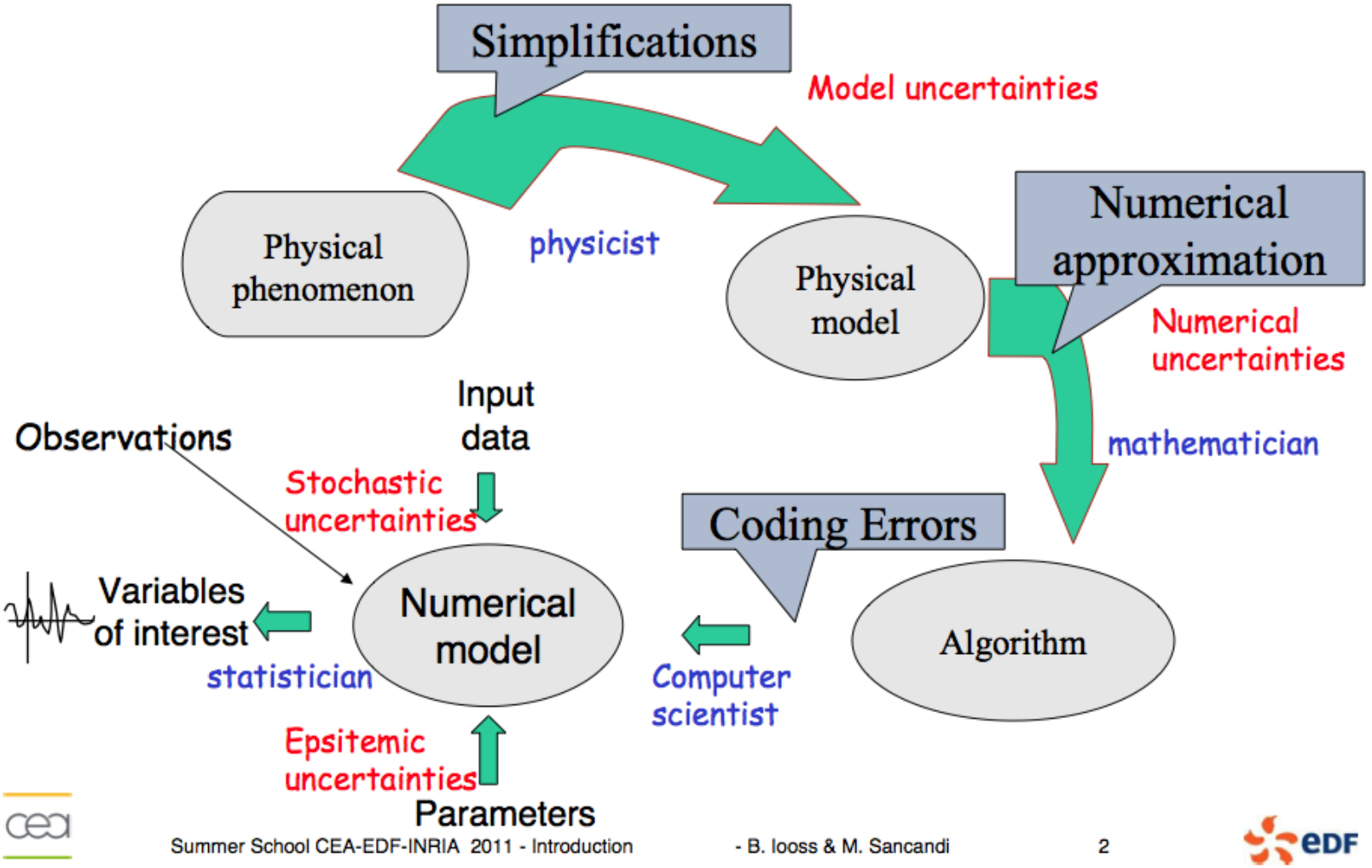
10 Avril 2013

Applications/Modélisation/Simulation Numérique

1. Une application est fréquemment décrite par un modèle mathématique, souvent un système d'équations aux dérivées partielles non-linéaires couplées.
2. Rôle prédictif des applications de simulations numériques.
3. Écart entre l'application (observations) et la simulation numérique issue de la modélisation.
4. La modélisation et la simulation numérique qui en découle introduisent des sources d'erreurs:
   - Erreur sur le modèle,
   - erreur sur les entrées/sorties et/ou variabilité naturelle de ces entrées,
   - erreurs sur les conditions initiales,
   - erreur d'approximation numérique du modèle,
   - ...
5. Chaque source d'erreur devra être intégrée dans le processus de prédiction.

Starting point: uncertainties everywhere in the modeling chain !

Main problem: credibility of predictions



Principaux enjeux dans la gestion des incertitudes

- Phase de modélisation
  - Améliorer le modèle
  - Explorer les meilleures combinaisons dans l'espace des entrées
  - Identifier les entrées les plus influentes
- Phase de validation
  - Réduire les incertitudes des sorties du modèle pour améliorer les prédictions
  - Calibration des paramètres d'entrées
- Utilisation du modèle
  - Optimiser les performances et la robustesse du système.
  - Estimation des événements rares

Sources d'incertitude en calcul scientifique

Toutes les sources d'incertitudes doivent être identifiées et caractérisées. Elles peuvent être réparties dans ces trois catégories:

- Les entrées du modèle peuvent être aléatoires ou épistémiques.
  1. Géométrie, paramètres du modèle, conditions initiales.
  2. Données externes au système incluant conditions frontières, excitation du système.
- Approximation numérique
  1. Discretisation,
  2. erreur de convergence des méthodes itératives,
  3. erreur d'arrondi,
  4. erreurs dans le code (les effets sur la solution numérique sont difficiles à estimer).
- Incertitude sur le modèle (hypothèses, abstractions, conceptions, formulation mathématique,...).

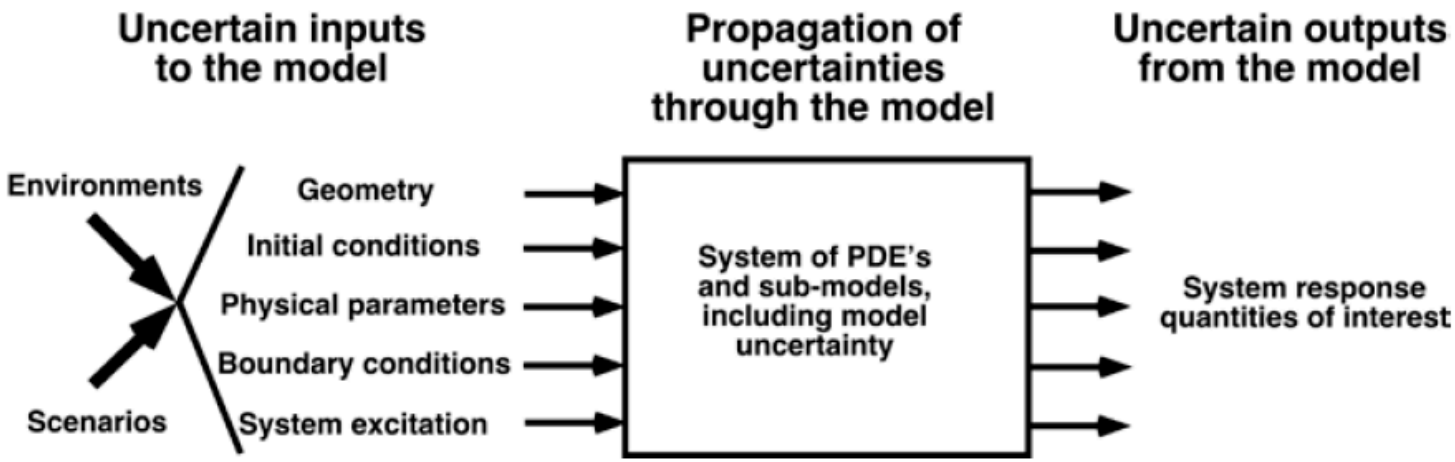
Types d'incertitudes

Toutes ces sources d'incertitudes peuvent être classifiées en:

- Incertitude aléatoire: les variations peuvent être caractérisées par une distribution de probabilité.
- Incertitude épistémique: il n'y a pas suffisamment d'information pour spécifier une seule valeur ou une distribution de probabilité.
- Incertitude mixte aléatoire et épistémique

Propagation d'incertitude à travers du modèle

Dans les méthodes non-déterministes, un nombre de calculs du modéle doit être effectué pour recouvrir l'espace des entrées vers l'espace des sorties du modèle.

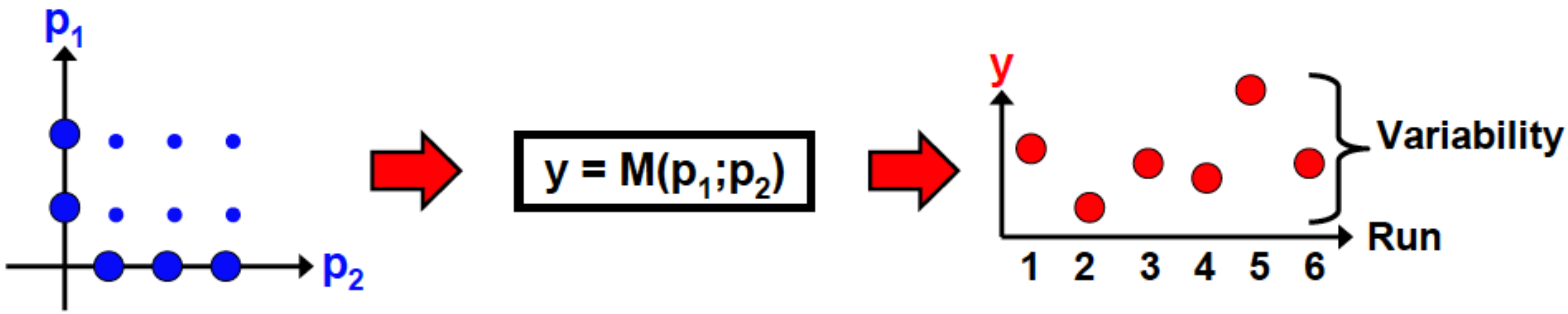


Le nombre de calculs nécessaire dépend:

- de la non-linéarité des équations aux dérivées partielles,
- de la structure de dépendance entre les quantités incertaines,
- de la nature des incertitudes (aléatoire, épistémique)
- des méthodes numériques utilisées.
- ...

Designs of Computer Experiments (DOE)

- Un plan d'expérience (DOE) fournit une méthode pour propager les incertitudes sur les entrées $p_1, p_2, \ldots, p_N$ d'une simulation numérique ou d'un problème physique vers les sorties Y.



- Un plan d'expérience sera utile pour explorer la variabilité totales des réponses d'un système lorsque les entrées varient simultanément sur leur interval $min(p_k) \leq p_k \leq max(p_k) \quad k = 1 \ldots N$

- La variabilité totale peut être explorée différemment par échantillonnage de la loi de probabilité conjointe des N-uplets $(p_1, \ldots, p_N)$ des entrées.

Propagation d'incertitude à travers dle modèle

- ☞ Arbitrer entre coût de calcul et précision des estimateurs
  - ➢ Monte Carlo: indépendant de la dimension des entrées, estimation non biaisée, fournit un intervalle de confiance sur l'estimation mais converge lentement.
    - ☞ coût important en nombre d'évaluations du modèle.
  - ➢ Le code de calcul est trop coûteux en CPU, il existe des méthodes alternatives :
    - Méthodes quasi-Monte Carlo
    - Hypercube Latin
    - Plan factoriel complet ou fractionnaire
    - ...
    - ☞ Utilisation d'un métamodèle.
      - un nouveau terme d'erreur
      - Le calage du métamodèle demande aussi un certain nombre d'appels au vrai modèle

Analyse de sensibilité: objectifs

- Réduction de l'incertitude de la sortie par hiérarchisation des sources
  - Variables à fixer pour obtenir la plus forte réduction (ou une réduction donnée) de l'incertitude de la sortie
  - Variables les plus influentes dans un domaine de valeurs des entrées
- Simplifier un modèle
  - détermination des variables non influentes, que l'on pourra fixer.
  - construire un modèle simplifié, un métamodèle

Analyse de sensibilité globale versus locale

- Approche déterministe: à partir de l'évaluation numérique des dérivées partielles du modèle.
  - **Inconvénients**: coût de calcul potentiellement important.
  - **Meilleure solution**: calcul par la méthode adjointe, adaptée à p élevé.

  ☞ Information purement locale

- Approche stochastique:
  - Le criblage (screening) : plans d'expériences classiques, numériques, outils graphiques.
  - Les mesures d'influence globale:
    - corrélation/régression sur les valeurs/rangs,
    - décomposition de la variance fonctionnelle (Sobol).
  - Exploration fine des sensibilités :
    - méthodes de lissage (param./non param.),
    - métamodèles (surface de réponse, modèle mathématique simplifié, . . .).

  ☞ Méthodes basées sur un échantillonnage du domaine des entrées.

  ☞ Information globale

Vérification et Validation

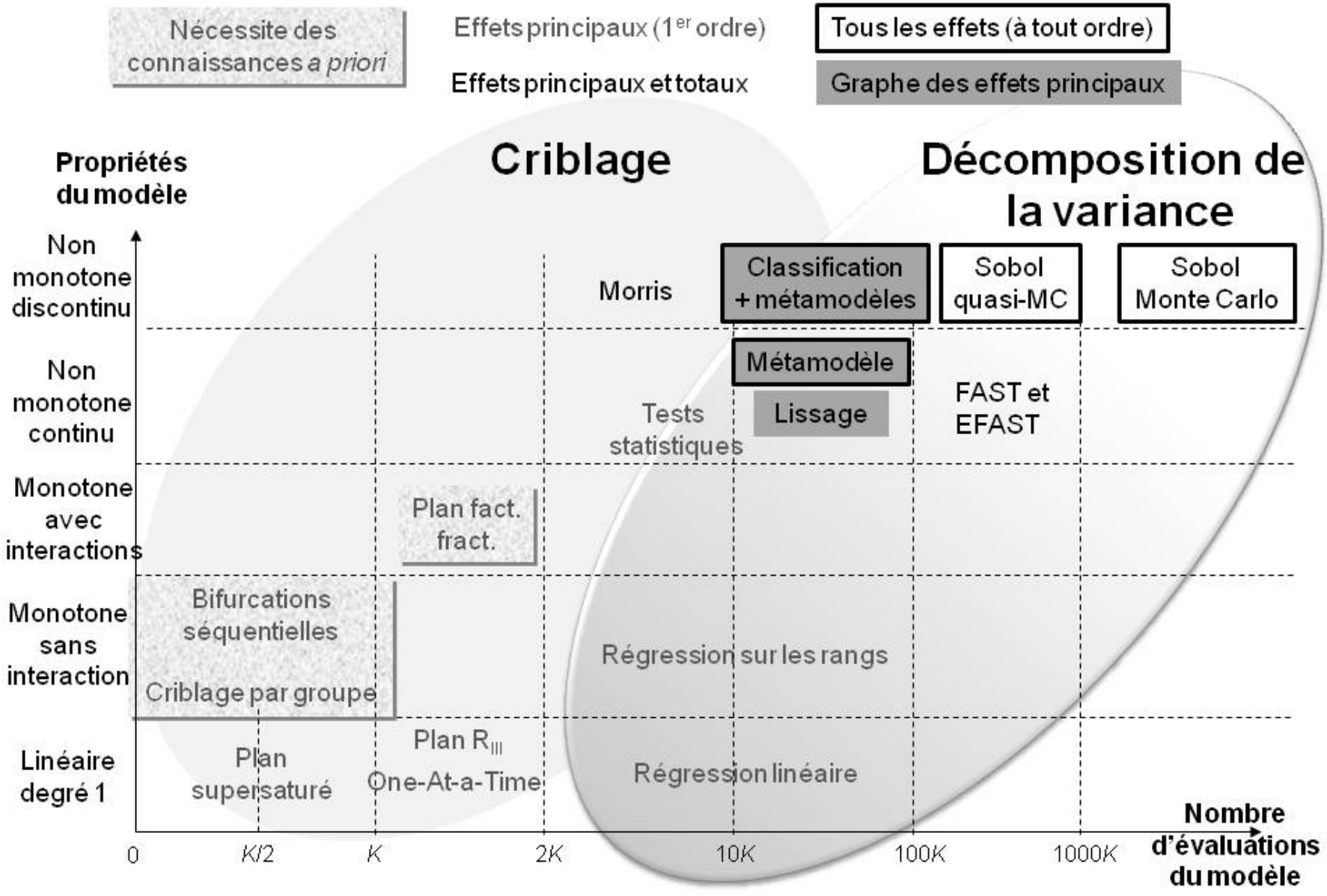
- Vérification: l'implémentation fournit **les solutions des équations du modèle**.

  Deux types de vérification:

  - Vérification de code
    - Trouver et corriger les erreurs dans les sources du code.
    - Améliorer la robustesse des algorithmes numériques.
    - Améliorer la fiabilité des logiciels en utilisant les techniques éprouvées de qualité logicielle.
  - Vérification de la solution
    - Garantir la pertinence des données d'entrée et de sortie pour le problème d'intérêt.
    - Estimation de l'erreur numérique de la solution, erreur due à la méthode de résolution (éléments finis, différences finies,...), la taille du maillage, la discrétisation en temps, les erreurs d'arrondis. . .

  La vérification concerne la qualité des solutions numériques mais pas l'adéquation des équations au problème posé (validation).

- Validation: l'implémentation fournit les solutions du problème physique posé.


Nécessite des connaissances *a priori*
Effets principaux (1er ordre)
Effets principaux et totaux
Tous les effets (à tout ordre)
Graphe des effets principaux
Criblage
Décomposition de la variance
Propriétés du modèle
Non monotone discontinu
Non monotone continu
Monotone avec interactions
Monotone sans interaction
Linéaire degré 1
Morris
Classification + métamodèles
Sobol quasi-MC
Sobol Monte Carlo
Métamodèle
Lissage
Tests statistiques
FAST et EFAST
Plan fact. fract.
Bifurcations séquentielles
Criblage par groupe
Régression sur les rangs
Plan supersaturé
Plan R_{III}
One-At-a-Time
Régression linéaire
0
K/2
K
2K
10K
100K
1000K
Nombre d'évaluations du modèle

Quantification d'incertitudes - Approche stochastique

- Caractéristiques de calcul: Temps de calcul du modèle, stockage, communication, distribution des données + outils pour gérer la complexité des processus.
- Les processus
  1. Paramétrisation
  2. Planification
  3. Répartition des données en entrée du modèle
  4. Calcul séquentiel/ parallèle
  5. Récupération des résultats
  6. Estimation des quantité d'intérêt

Lorsqu'on utilise par un méta-modèle on y rajoute:

1. Récupération des résultats du modèle
2. Construction du méta-modèle
3. Planification
4. Répartition des données en entrée du méta-modèle
5. Calcul de méta-modèle
6. Récupération des résultats
7. Estimation des quantité d'intérêt

Le calcul parallèle ?

Qu'est ce que le calcul parallèle?

Plusieurs calculs exécutés simultanémant sur une ou plusieurs unités de calcul.

Quand?

Quand les limites des ressources nécessaires, des performances et/ou de la taille des problèmes traités sont atteintes sur un seul processeur

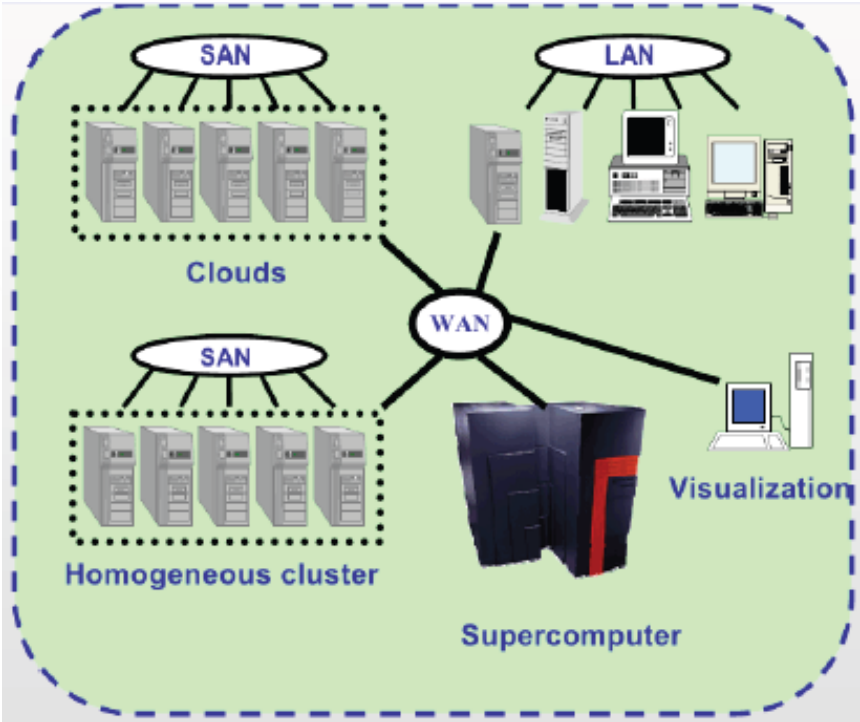
Comment?

Par décomposition:

- des données pour traiter un plus gros volume de données ou traiter par une même tâche des données différentes: "data parallelism"
- de tâches indépendantes: "task parallelism "

Contexte HPC: Super calculateurs, Clusters, Grille, Cloud

- Ressources de calcul
  - Clusters Homogène: noeuds de plusieurs CPU multi-coeurs avec ou sans GPU: SGI ALTIX ICE, . . .
  - Super calculateur: IBM Blue Gene (4 096 nœuds de 16 cœurs à 1,6 GHz) - 836 TFlop/s de puissance crêteà l'IDRIS), . . .
  - Grille de calcul: GRID500, . . .
  - Cloud
- Hiérarchie de réseaux
  - WAN: Internet, Private WAN, . . .
  - LAN: Ethernet
  - SAN: InfiniBand, . . .



Motivations en simulation numérique

- La vitesse des processeurs a cessé d'augmenter (x2 tous les 18mois).
- Parallélisme naturel dans la plupart des applications.
- Possibilité de traiter des gros volumes de données.
- Nécessité d'exécuter un grand nombre de runs.
- Meilleure gestion des accès mémoire.
- La vitesse des réseaux augmentant, c'est la possibilité d'utiliser des ressources réparties sur un site ou sur plusieurs sites
- ...

En majorité: Codes parallèles

- Plusieurs codes indépendants
- Un seul code parallèle intensif

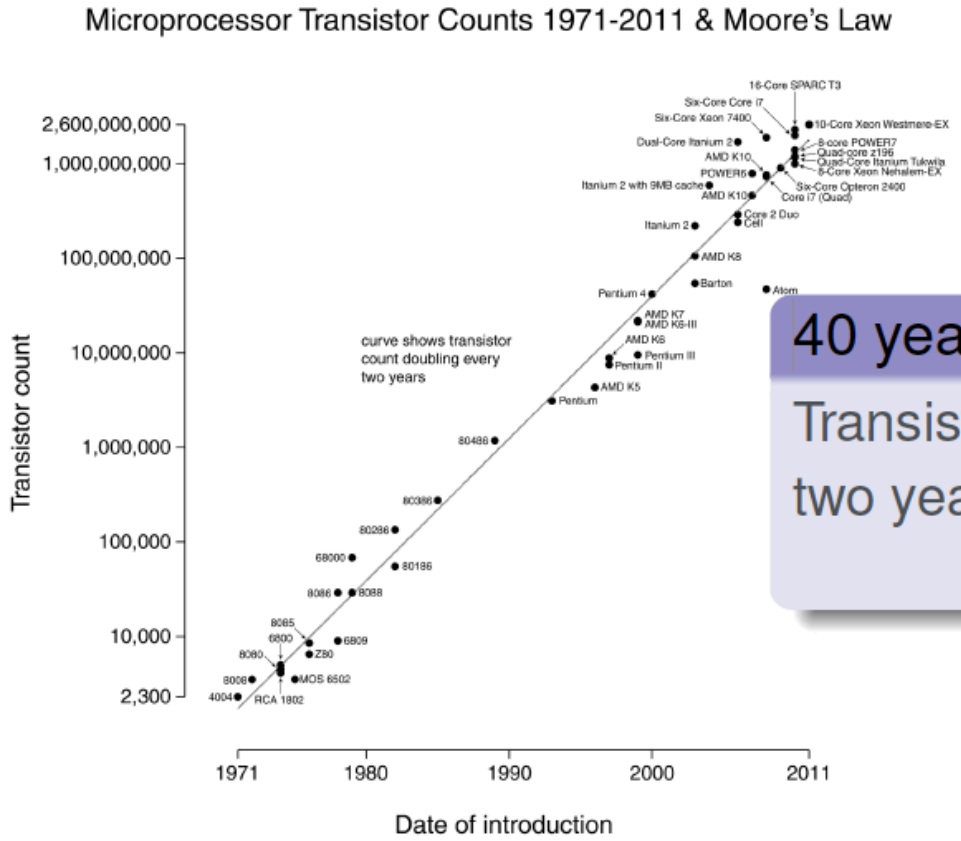
Unité de calcul: CPU versus GPU

Composition d'un noeud de calcul (Avec des proportions différentes)

- Une unité de contrôle
- Une unité d'arithmétique et logique (ALU)
- Mémoire(DRAM, Caches,. . .)

- Un grand nombre de coeurs capables de traiter rapidement des calculs simples sur des grandes quantités de données (GPU)
- Les processeurs classiques (CPU) sont moins rapides mais savent résoudre des problèmes plus compliqués
- Une combinaison de ces deux types de processeurs (accélérateurs)

Loi de Moore - Top500



40 years of improvements

Transistor counts double every two years...

...but how?

Le nombre de transistors continue à augmenter par unité de surface.

La puissance consommée et dissipée est le facteur limitant

- $Puissance \propto frequence^3$ ☛ fréquence limitée
- Plus de performance avec moins de puissance ☛ Adaptation software?

Implication software

Le coût en énergie de la fréquence

	Nb Cores	Fréquence(Hz)	Flops/s	W	Flops/s/W
Superscalar	1	1.5X	1.5X	3.3X	0.45X
Multicore	2	0.75X	1.5X	0.8X	1.88X

Conséquences sur les applications

- Exécuter une application à un temps t_0 sur un calculateur
- Pas de modifications fondamentales du code entre t_1 et t_0
- Exécuter l'application au temps t_1 sur un nouveau calculateur

☛ quelles sont les performances?

Évolution des performances en HPC

Période de fréquence croissante ($<$ 2003)

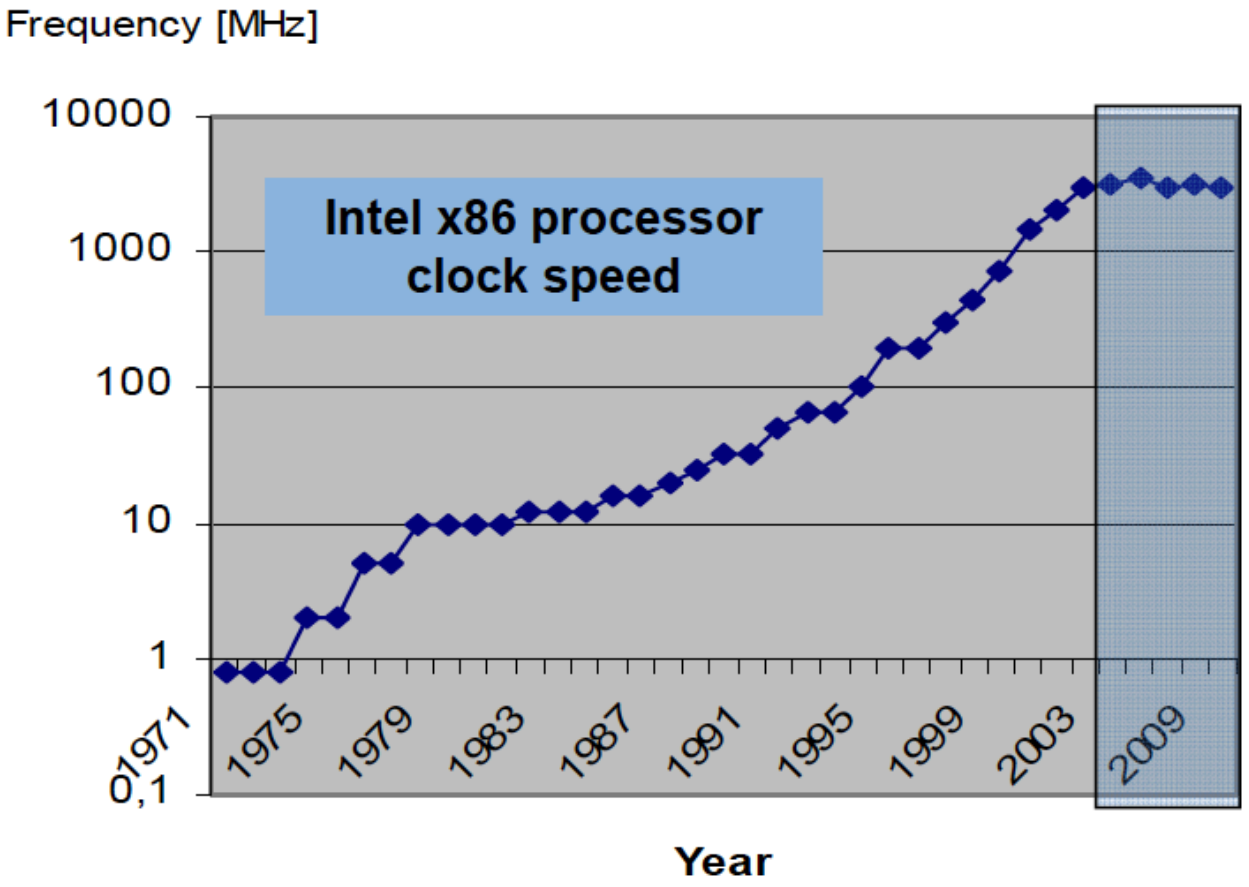
- Peu de parallélisme dans les architectures
- La fréquence augmente $\Rightarrow$ Nb Flops/s augmente
- Les codes sont plus rapides

☛ L'implémentation des codes n'a pas besoin d'être modifiée pour exploiter les nouvelles architectures

Période Multi-core ($>$ 2003)

- Plus de parallélisme dans l'architecture (core,mémoire, stride des disque,. . .)
- La fréquence décroit
- Les codes sont plus lents

☛ Les codes doivent être modifiées pour exploiter le parallélisme de ces nouvelles architectures



Single core: Instruction level parallelism:

- Superscalarity
- **Single Instruction Multiple Data (SIMD) → SSE / AVX**

Investing the transistor budget:

- **Multi-Core/Threading**
- **Complex on chip caches**
- New on-chip functionalities (GPU, PCIe,...)

Implications Multi-core

Quelques définitions

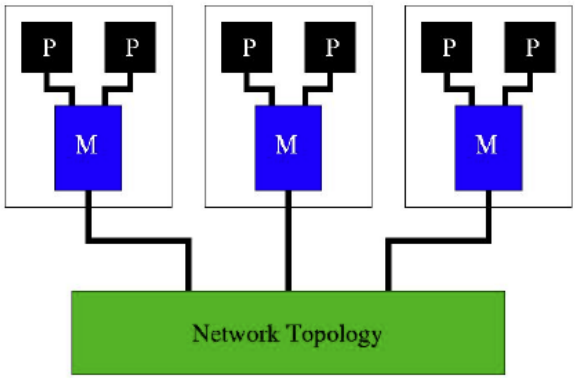
- CPU (Central Processing Unit): puce ou processeur qui effectue les opérations de base du système.
- socket: le socket fournit au CPU les connections au bus système et tous les devices attachés à ce bus (mémoire, adaptateur réseau, I/O,...).
- Core: unité de calcul contenu dans un CPU, ses propres registres et cache L1.

Ressources partagées

- caches: les cores d'un CPU peuvent se partager les caches L2 ou L3
- socket: Les cores d'un même CPU se partagent le même socket du CPU

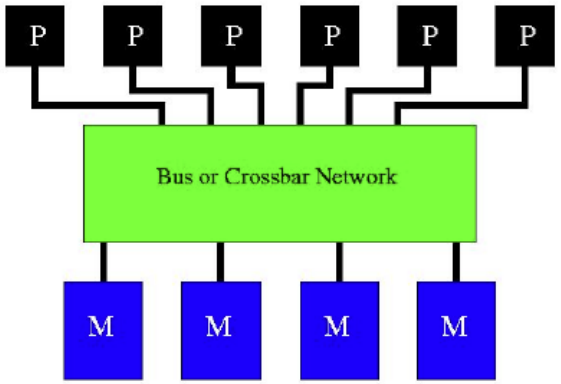
Ressources partagées ☛ Potentiel de conflits d'accès aux ressources plus important

Mémoire partagée distribuée



- Chaque processeur a sa propre mémoire.
- Les processus s'exécutent indépendamment
- Ils communiquent entre eux par échange de message.
- Overhead dû aux communications
  - ☞ Réseau performant

Mémoire partagée

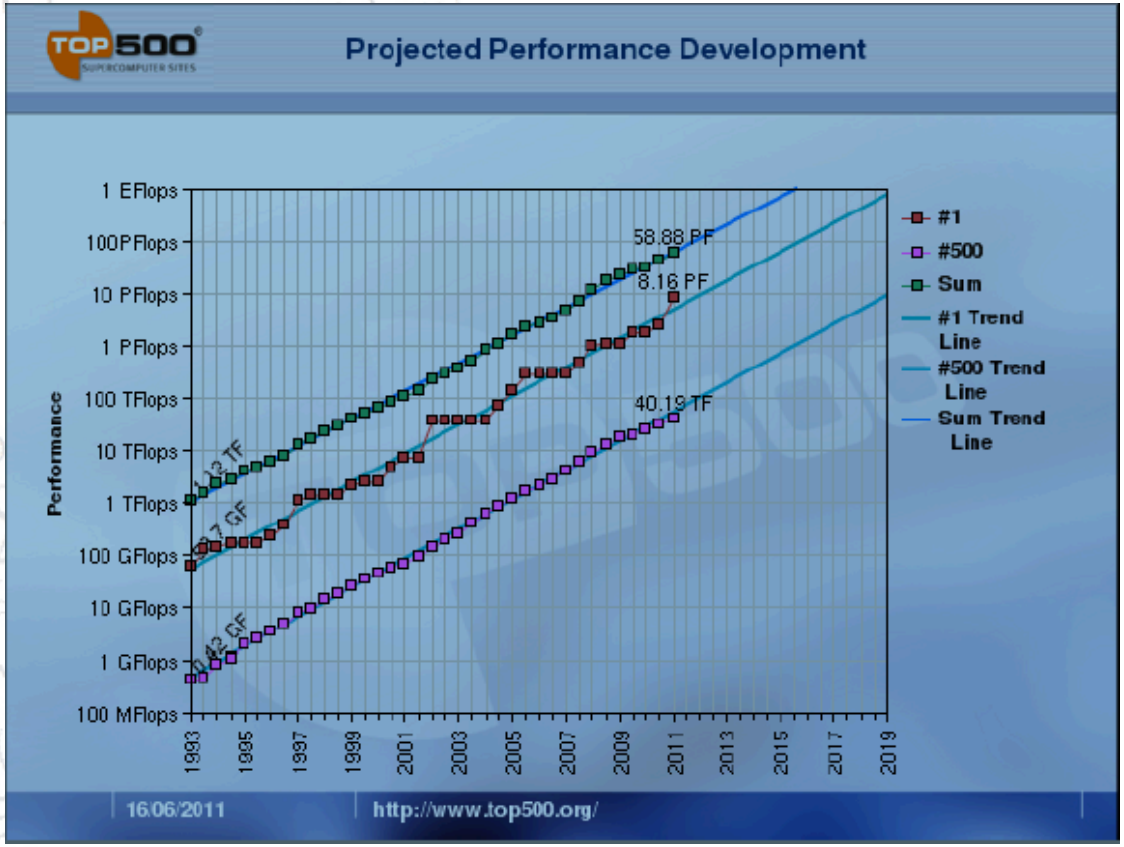


- La mémoire globale est commune à tous les processeurs.
  - ☞ Plus facile à utiliser
- ☞ cohérence de cache.
- Performance de la bande passante mémoire.
- Défaut de scalabilité des accès mémoire.
- Problème de " False Sharing" ou de " Race Dectection"
  - ☞ Difficile à déboguer

Deux classes d'architectures à mémoire partagée

- UMA (Uniform Memory Access)
  - Machine SMP (Symmetric Multi Processor)
  - Temps d'accès à la mémoire est identique depuis chaque processeur.
  - CC-UMA: Système de cohérence des caches
- NUMA (Non-Uniform Memory Access)
  - Liens physiques entre plusieurs machine SMP
  - Temps d'accès à la mémoire n'est pas uniforme depuis chaque processeur.
  - CC-NUMA: Système de cohérence des caches
  - Physiquement distribué, virtuellement partagée (soft)

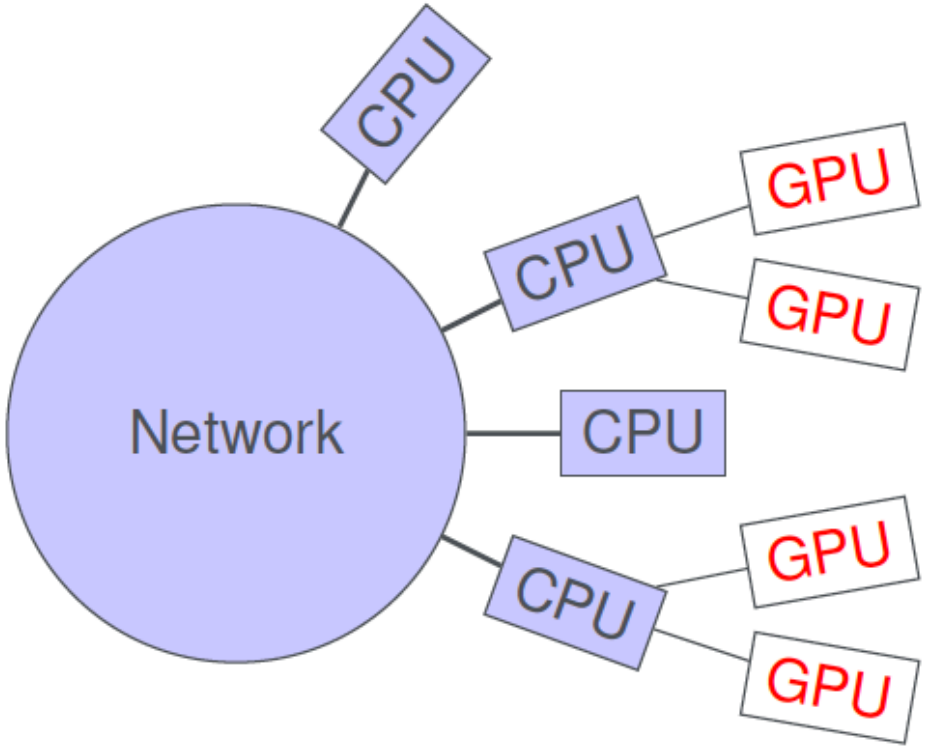
☞ Pour les performances sur les machines NUMA, on devra tenir compte du caractère physiquement distribué de la mémoire.



- Plus rapide que la loi de Moore (double tous les 14 mois)
  - ☞ Le Top1 sort du Top500 en 8 ans
- La consommation électrique, l'encombrement et le refroidissement sont devenu des freins à l'extension des systèmes.
  - ☞ Les architectures Hybrides (CPU/GPU) apparaissent.

Distribution des données sur un calculateur Hybride

How a code can be executed on hybrid CPU-GPU architectures?



Data transfer is still MPI-based

Only on-board communication between GPU and CPU

Data distribution should depend on the presence of GPUs on the nodes $\rightarrow$ Multilevel parallelization required

Architectures Hybrides

GPGPU sur les calculateurs

- Les architectures traditionnelles ont atteint leurs limites
  Plus de cores/noeud, plus de mémoire mais accès pas plus rapide.
- Actuellement, 3 sur 5 gros calculateurs sont des machines hybrides

Plus de parallélisme dans le Hardware ☛ Adaptation des codes pour obtenir de la performance

Les codes doivent être conçus différemment

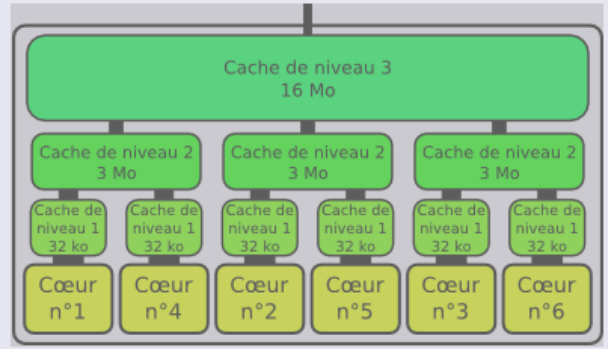
- Les codes MPI ne passent pas au massivement parallèle.
- les performances augmentent en améliorant le parallélisme dans le code.
- Les performances augmentent en ajoutant des niveaux de parallélisme (OpenMP, GPU,. . .)

Et plus de performances si affinités

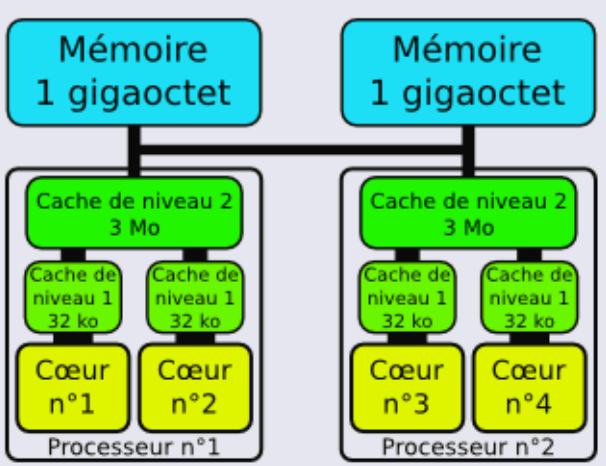
L'apport de toujours plus de parallélisme dans les calculateurs pose le problème de la gestion des affinités.

☞ Les performances en dépendent

- Principe de localité et mémoire cache
  - ☞ optimisation de l'accès aux données.
- Caches et affinité entre tâches et données: le déplacement d'une tâche sur un autre coeur.
- Partage de données entre tâches: accès à la mémoire par différent coeurs.
- Machine NUMA et affinités pour la mémoire: accès plus rapide aux données proche du processeur qui effectue l'accès.



Un processeur contenant six cœurs et 3 niveaux de cache.

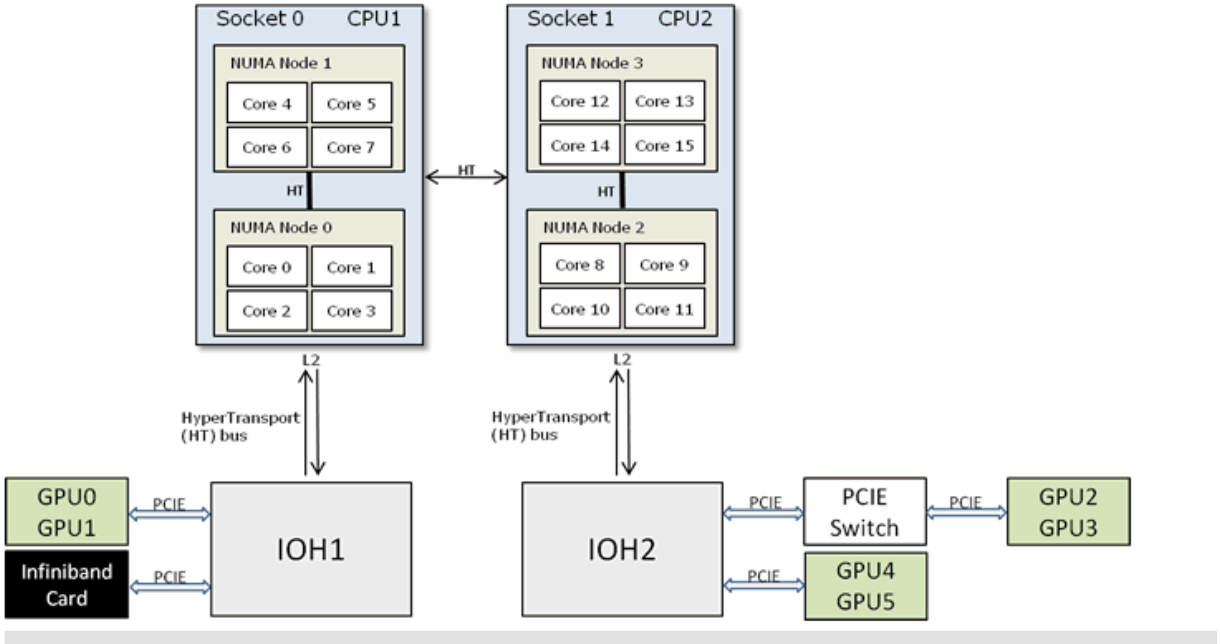


Une machine NUMA (*Non-Uniform Memory Access*) constituée de deux processeurs bi-cœurs.

☞ Nos programmes peuvent s'exécuter sans tenir compte de toutes ces contraintes, mais ils iront moins vite...

Et bien d'autres affinités

- Périphériques qui peuvent être plus proches de certains cœurs que d'autres (GPU,carte réseau).



- Calculateurs hétérogènes ou hybrides contenant différents types de cœurs de calcul:

  ☞ Combiner ces différents types de processeurs permet d'atteindre des puissances de calcul gigantesques si on arrive à déterminer les affinités entre les # coeurs et les # tâches.

Architecture hautement parallèle: Accélérateurs

Intel Xeon/Phi

- 60+ cores 512 Bit SIMD
- Clock Speed: 1000MHz
- Nombre de transistors: 3 B (22nm)
- Consommation énergétique: 250W
- Performance Max (DP): 1TF/s
- Bande passante Mémoire: 250GB/s (GDDR5)
- NB Threads: 100-200
- Language: Fortran, C/C++ + OpenMP+vectorisation

NVIDIA Kepler (GK110)

- 15 unités SMX avec chacune 192 SP cores
  - ☞ 2880 SP cores au total
- Vitesse de l'horloge: 700MHz
- Nombre de transistors: 7.1 B (28nm)
- Consommation énergétique: 250W
- Performance Max (DP): 1TF/s
- Bande passante Mémoire: 250GB/s (GDDR5)
- NB Threads: 10 000+
- Language: CUDA, OpenCL, OpenACC

Évolution rapide du HPC

- Hardware
  - Toujours plus de parallélisme (multi-core, mémoire, stockage,. . .).
  - Architectures hétérogènes.
  - Les accès mémoire sont le principal frein aux performances.
- Software
  - Parallélisme multi niveaux.
  - Optimisation: gestion des affinités,. . .
  - Gestion de la précision, de la stabilité.

☞ Maintenance et robustesse?

- Point de vue utilisateur
  - Nouveaux types d'architectures (GPU, MIC, BG.Q,. . .)
  - Nouveaux paradigmes de programmation (MPI, OPENMP, Hybrid, OpenCL,PGAS,. . .).
  - Un code unique pour CPU et Accélérateurs?
  - Les gros codes de calcul doivent suivre.

☞ Quelle pérennité?

Efforts d'optimisation

Compromis entre efforts d'optimisation et gain de performance

- Optimisation rapide: performances modérées. . .
  Le Hardware, le compilateur ne réussit pas à exploiter le parallélisme de l'architecture.

  Une meilleure connaissance de l'application est nécessaire.

- Optimisation approfondie: algorithme, implémentation, déploiement, gestion des affinités.
  La maintenance des efforts n'est pa assuré.

  ☞ Génération automatique ou semi-automatique?

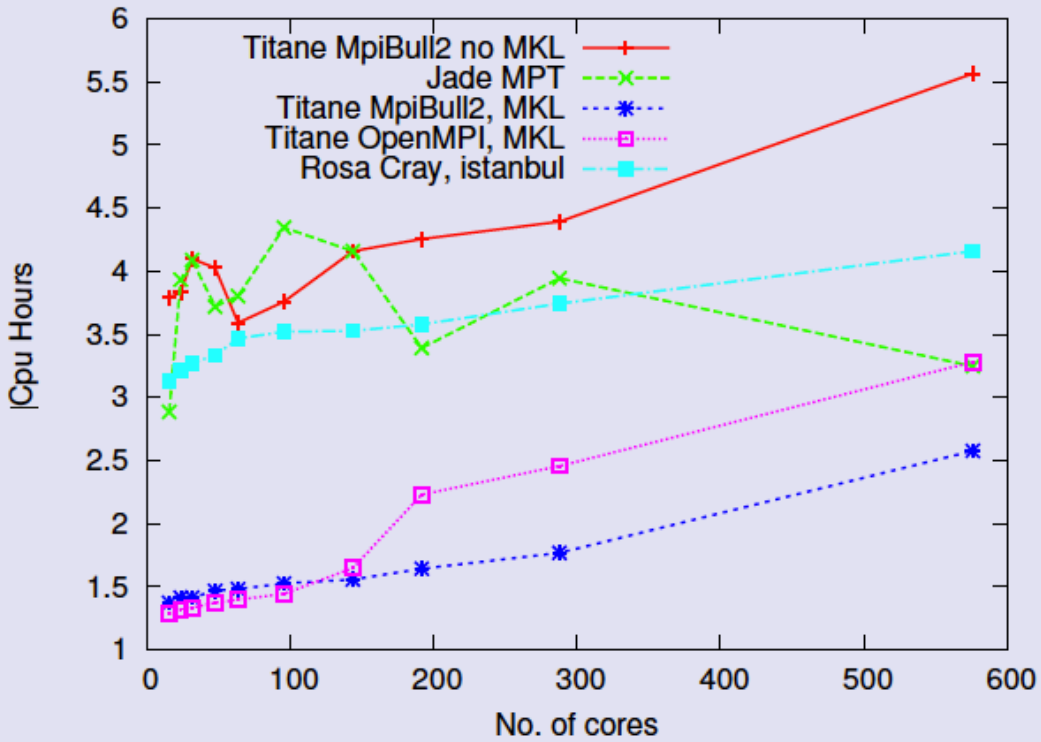
  (quelques propositions: HMPP, OpenACC,. . .)

- De nouveaux paradigmes de programmation exploitant la parallélisme du Hardware sont attendus.

  ☞ Maintenance, portabilité?

Architecture, bibliothèques et réseau

Même run, mêmes sources: différents environements utilisateurs



Differences up to a factor of 3!

Étude au cas par cas

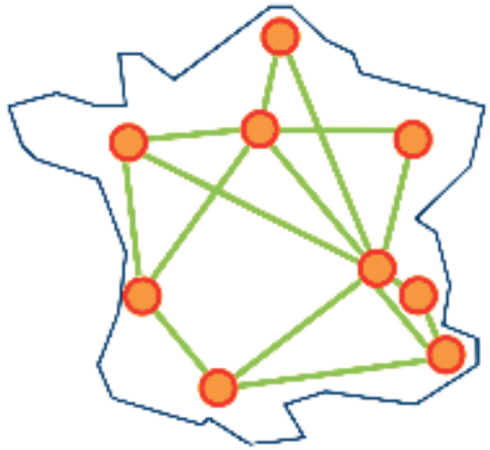
Les analyses sont système-dépendantes, pas de règles systématiques.

☞ Bien connaître son application

Grille de calcul

Grille de calcul

Projet Grid5000 (www.grid5000.fr)



- Ressources informatiques hétérogènes (clusters, supercalculateur, stockage,. . .)
- Mise en réseau de ces ressources
- Stockage.
- . . .

Middleware (NetSolve, DIET, . . .): couche réseau + services logiciels

- Gestion de l' hétérogénéité des ressources,
- allocation des données et des tâches sur les ressources,
- Fault detection, mécanisme de checkpoint/restart,. . .
- récupération et réorganisation des résultats
- éviter le transfert multiple de données
- . . .

Grâce aux grilles

- Les calculs au temps d'exécution très longs deviennent possibles en temps raisonnable.
- Possibilité de lancer un nombre très grand de runs (plusieurs millions)

Cloud

Cloud

- Basé sur la virtualisation
- Habituellement basé sur des processeurs de faible performance: presque sans contrainte sur le réseau
  - ☞ Fault-tolerance?

HPC Cloud

- Noeuds de calcul puissants avec beaucoup de mémoire,
- Performances du réseau garanties (F E.g., 10 Gbps, low latency)
- Opérationnel mais pas simple. . . (panne des infrastructures, noeuds non accessibles,. . .
- Coût
- Quelques tentatives pour mettre un calculateur (Blue Gene) dans le Cloud.

Scénario futur en HPC

Exascale en 2018 ???

Blue Gene Scénario ???

Accélérateurs Scenario ???

UQ et environnement logiciel

- Cougar (IFPEN & partners) - reservoir simulation
  - Commercial
  - Fonctionnalités
    - Analyse de sensibilité (basé sur la variance)
    - Surface de réponse paramétrique (polynomiale)
    - Surface de réponse non-paramétrique (Krigeage)
    - Planification
    - ...
  - Calcul sur grille
- OpenTURNS: exposé R Barate

  Calcul Haute Performance avec OpenTURNS
- Uranie: exposé F. Gaudier

  Plate-forme Incertitude Uranie: Fonctionnalités et Applications