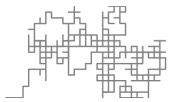
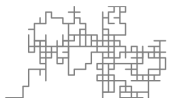
GPU Computing : début d'une ère ou fin d'une époque ?

eric.mahe@massiverand.com

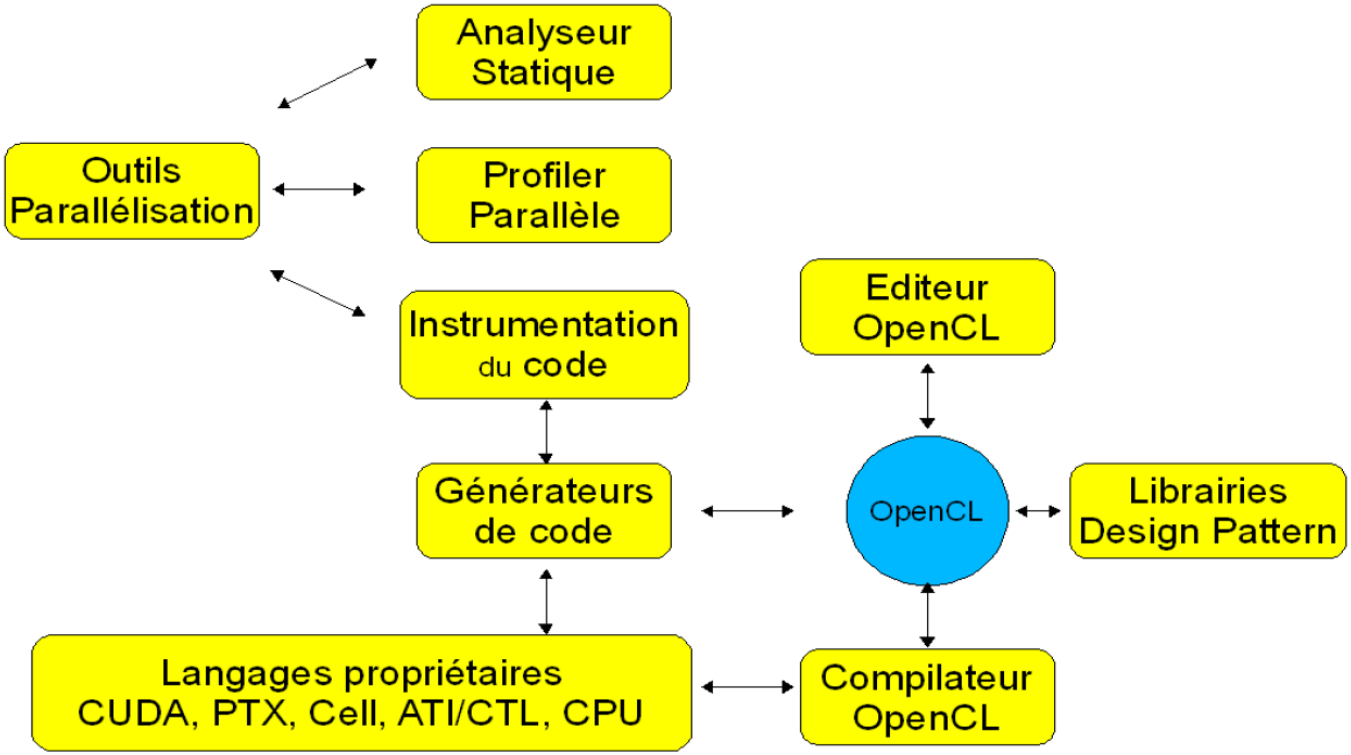




Plan

- Génèse du projet OpenGPU
- Misères et grandeurs des GPUs
- Quelle place pour OpenCL ?
- Les avancées de l'architecture Kepler
- Quelques remarques sur la consommation

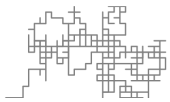

Génèse du projet OpenGPU (1)



Génèse du projet OpenGPU (2)

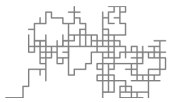
Génèse du projet OpenGPU (3)

- Budget : 6M€
- Durée : 2 ans
- 4 sous-projets :
  - SP1 : organisation
  - SP2 : outils
  - SP3 : matériel
  - SP4 : démonstrateurs

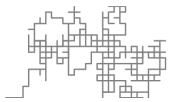

Misères et grandeurs des GPUs (1)

- Le GPU Computing : c'est (très) compliqué
  - Compliqué à apprendre
  - Compliqué à enseigner
  - Compliqué à comprendre
  - Compliqué à démarrer
  - Compliqué à debugger
  - Compliqué à optimiser

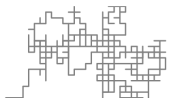

Misères et grandeurs des GPUs (2)

- Compliqué à apprendre :
  - Grande diversité des développeurs :
    - C, Fortran, C++, Java, ...
  - Penser parallèle n'est pas inné :
    - for(int i = 0, …) C[i] = A[i] + B[i]
    - C[threadIdx.x] = A[threadIdx.x] + B[threadIdx.x] ;
    - C[blockIdx.x] = …
    - atomicAdd(&C[0],val) ;


Misères et grandeurs des GPUs (4)

- Compliqué à comprendre :
  - Des notions très proches du hardware :
    - Warp, hiérarchie mémoire, PCI-E, cache L2
  - De nouvelles (ou très anciennes) obligations :
    - Frontières de 128 bits, coalesence, padding, malloc/free
  - De nouvelles notions :
    - SIMD, grille de threads, barrière de synchronisation, ...


Misères et grandeurs des GPUs (4)

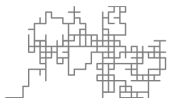
- Compliqué à démarrer :
  - Qui a essayé d'installer CUDA sur Linux ?
    - Blacklist du driver nouveau
    - Recompilation du kernel
    - Modification des paramètres Xconf
    - 32 bits vs 64 bits
    - Driver propriétaire obligatoire
  - Sous Windows ?
    - Compilateur Microsoft obligatoire (pas de gcc)
    - #include <cutil_inline.h> ??????
    - Règles CUDA de compilation ????????

Misères et grandeurs des GPUs (5)

- Compliqué à debugger :
  - GPU + Driver propriétaire = boîte(s) noire(s)
    - Pas de spécifications publiques
    - Une abstraction fournie par CUDA
    - Que fait vraiment le driver ??
  - Outils de debug :
    - Linux : gdb-cuda + DDD
    - Windows ? Parallele Nsight : patche CUDA
  - 10 M de threads … où est l'erreur ? printf dans le kernel !!
  - “Unknown error at line 457” ?????

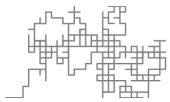
Misères et grandeurs des GPUs (6)

- Compliqué à optimiser :
  - Accès mémoire :
    - Coalesced read/write
  - Loi de Brent :
    - Minimum de calcul par thread
  - Gestion de la mémoire partagée
    - Un vrai casse-tête : conflit d'accès, 16 banques, pénalités
    - Utilisation du cache L2 très efficace
  - Vectorisation, unroll des boucles, ...
  - Exemple : réduction !!

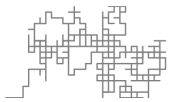

Misères et grandeurs des GPUs (7)

- Mais, ça marche (très) bien:
  - INRA (3 M de gènes):
    - 16 processus / Sun = 3 ans
    - 6 processus OpenMP / 12 Xeons = 18 jours
    - 160 cœurs + 40 Tesla 2070 = 1 heure
  - Numtech (solveur fluide compressible, 2048 x 1024) :
    - CPU : 44 heures
    - GPU : 1 heure


Misères et grandeurs des GPUs (8)

- Mais, ça marche (très) bien:
  - IBISC (recherche de palindromes, 120 M de bases):
    - CPU : 20 minutes
    - GPU : 1 minute
  - ESI (systèmes linéaires creux) :
    - CPU : 700 minutes
    - GPU : 350 minutes

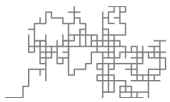




Misères et grandeurs des GPUs (9)

- Mais, ça marche (très) bien:
  - Apprentissage de la parallélisation
    - Intégration de ce nouveau paradigme
  - “Repenser le code”
    - Modernisation et optimisation de la partie séquentielle
  - Arrivée de nouvelles architectures
    - GPU, IGP, openMP, MPI, OpenCL, CUDA, …
  - Nouvelles publications pour les laboratoires

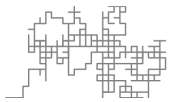
Quelle place pour OpenCL ? (1)

- OpenGPU = OpenCL
- Mais :
  - CUDA plus mature
  - OpenCL moins performant
  - Très très verbeux
  - Gestion des priorités
  - Démonstration ...


Quelle place pour OpenCL ? (2)

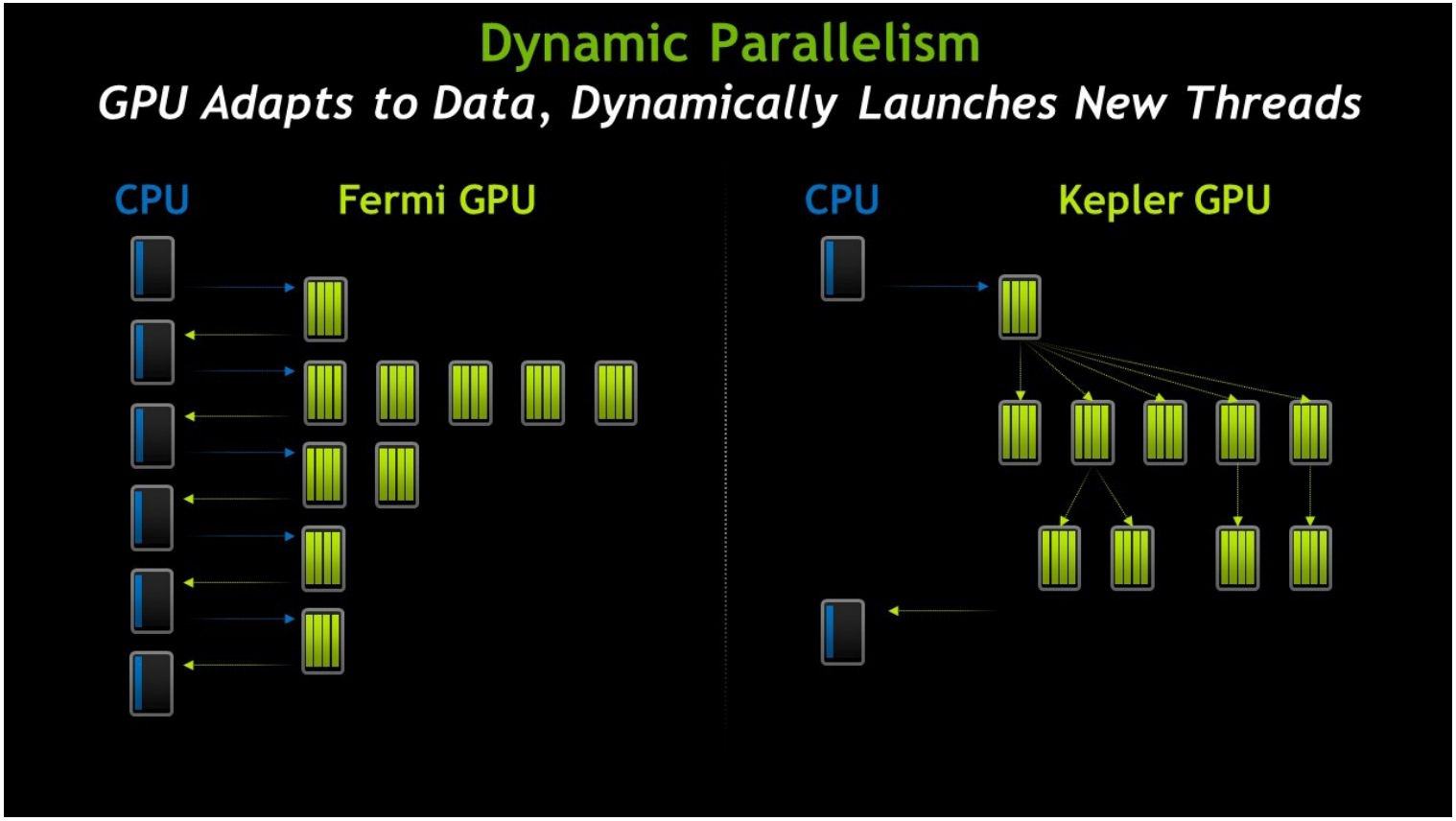
- Intel SDK 1.5 pour Ivy Bridge :
  - Prise en compte de l'IGP
  - Très bonne vectorisation du code
- CUDA vs OpenCL :
  - ~ 1,5 fois plus rapide
- Portabilité sans optimisation envisageable


Les avancées de l'architecture Kepler (1)

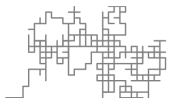
- GK104 = GTX 680
  - Pour les joueurs uniquement
- GK110 = GPU Computing
  - Disponible pour la fin de l'année
  - 4 fois plus rapide que Fermi !!
  - Plus de registres, grille plus grande, 4 scheduleurs, …
  - Doublement du cache L2 !!

Les avancées de l'architecture Kepler (2)

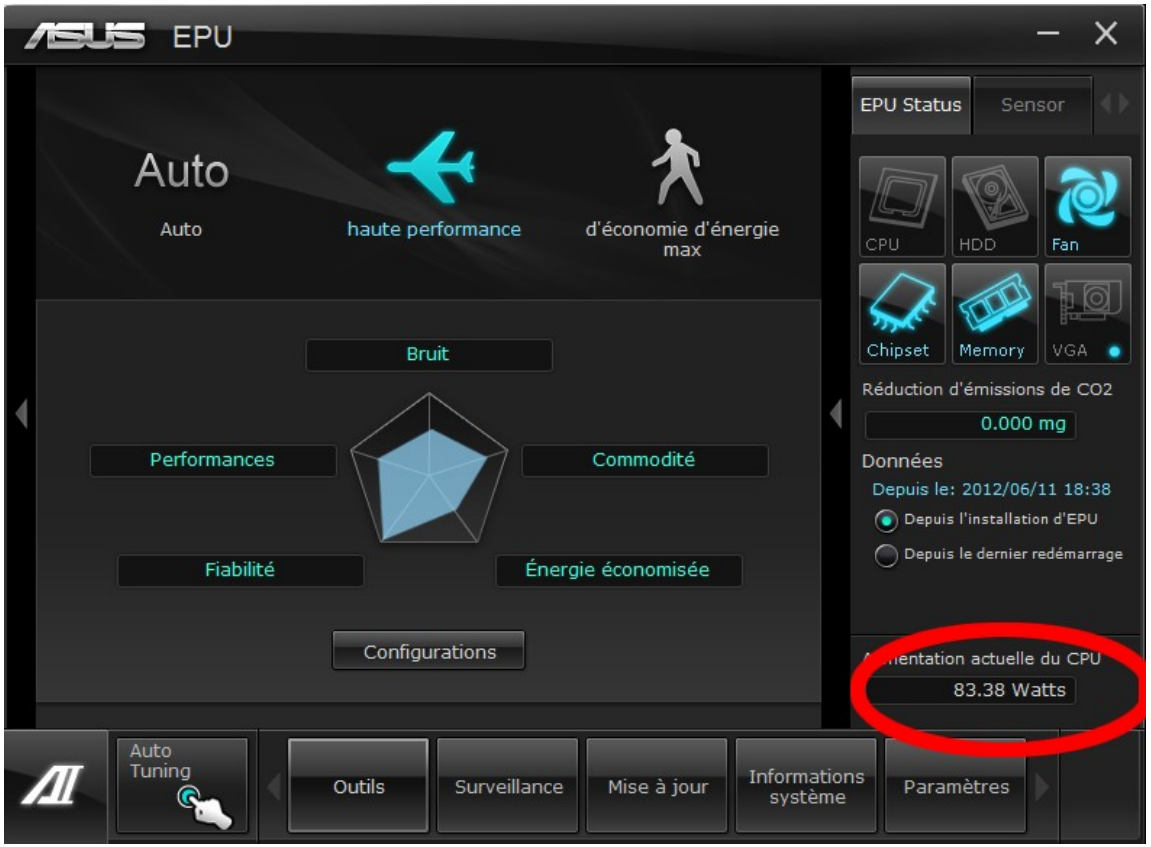


Quelques remarques sur la consommation

- Exécution d'un kernel OpenCL sur un 3770 K
  - Première exécution sur les 4 cœurs
    - 22 secondes
  - Deuxième exécution sur l'IGP (HD 4000)
    - 20 secondes
  - Et la consommation ?

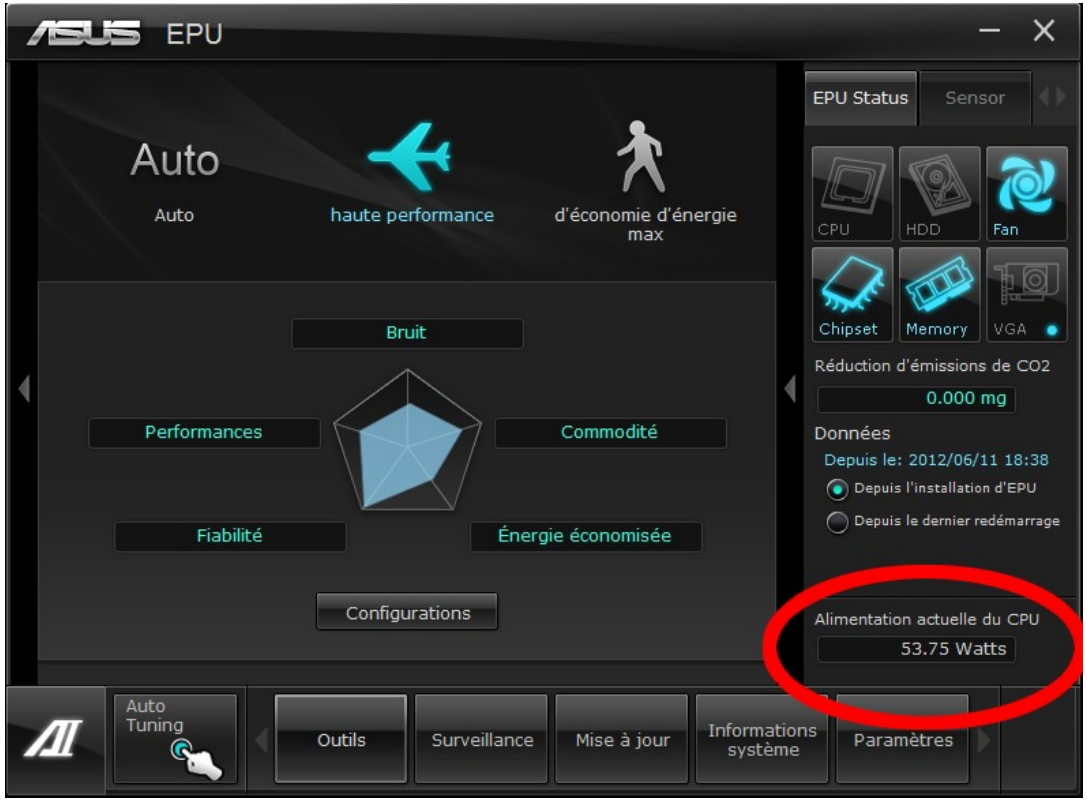




Quelques remarques sur la consommation



CPU = 83.38 Watts

Quelques remarques sur la consommation



IGP = 53.75 Watts

Conclusion

- Fin d'une ère ?
  - Certainement !
  - Penser uniquement séquentiel est “has been”
- Début d'une époque ?
  - Cartainement !
  - La parallélisation est rentrée dans les crânes !
- De nouvelles directions :
  - Map and Reduce, IGP, ARM, Kepler, OpenCL 1.2, ...