

Utilisation du GPU pour la radio logicielle

Pierre-Henri Horrein

Journées scientifiques SEmba 2011

20/20/2011

Sommaire

- 1 Contexte et objectif
- 2 Approches pour la conception
- 3 Approches pour l'intégration
- 4 Expérimentations
- 5 Conclusion

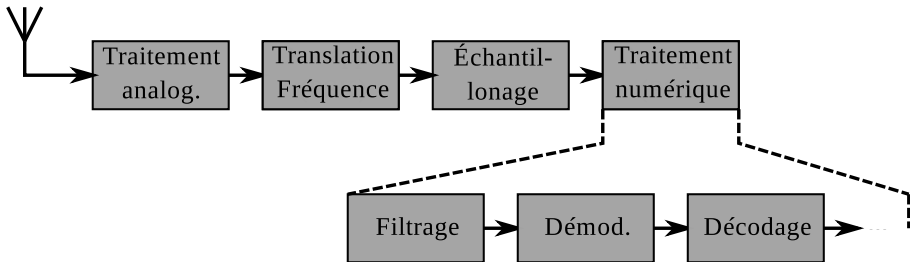
Radio flexible/reconfigurable/logicielle

- Évolution des terminaux :
 - de plus en plus de modes de connexion
 - évolution rapide des normes réseaux
 - normes de plus en plus gourmandes en puissance de calcul
 - contraintes de place et de consommation

- Évolution des unités de calcul :
 - augmentation de la puissance de calcul
 - intégration de plus en plus d'éléments sur puces

- Nouveaux objectifs :
 - la puissance de calcul n'est plus la seule contrainte
 - la flexibilité de l'implantation est à prendre en compte

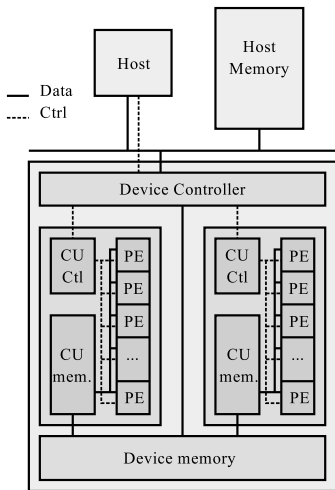
Radio logicielle : représentation et exécution



- Graphe de tâches, modèle de type SDF ou KPN
- Opérations :
 - connues (transformées, filtres, codage, ...)
 - intensives (haut débit, beaucoup de calculs flottants)
- Utilisation dans le cadre d'environnement (GNURadio, SCA)

- General Purpose computation on Graphic Processing Units
- Exécution d'opérations génériques sur un processeur graphique
- Différents environnements selon les constructeurs: CUDA (NVidia), Stream (ATI), ...
- Open Computing Language (OpenCL): unification des interfaces, en surcouche des environnements existants

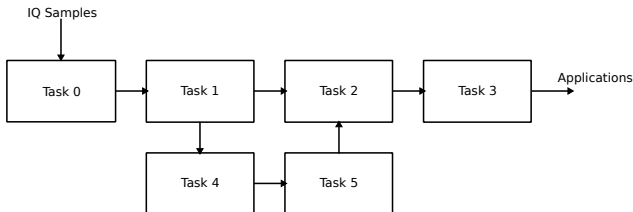
GPU: architecture OpenCL



- Application séparée entre l'hôte (host) et le périphérique (device)
- Opération unique (kernel), appliquée sur tous les éléments d'un vecteur (SIMD)
- L'hôte gère :
 - le périphérique (mémoire, exécution)
 - la mémoire hôte
 - la communication entre l'hôte et le périphérique
- Le périphérique est responsable de:
 - ses unités d'exécution
 - l'ordonnancement d'un kernel sur ses unités
 - de sa mémoire interne et de son cache
- Bande passante mémoire (ordres de grandeur)
 - host/device: 8 GB/s (PCIe)
 - interne: ~60 GB/s

SDR: GNURadio

- Framework pour la radio logicielle
- GNURadio définit :
 - un ensemble d'opérations de bases pour la radio
 - une gestion de l'exécution de ses opérations
 - une interface pour définir les applications
 - l'intégration des E/S (Ettus, audio)
- Applications GNURadio :
 - Synchronous Data Flow
 - FIFO logicielles entre les tâches
 - Ordonnancement entre les tâches basées sur l'état des buffers



Objectif

Radio logicielle

SDF, graphe de tâches
distribué
basé sur des FIFO

GPU

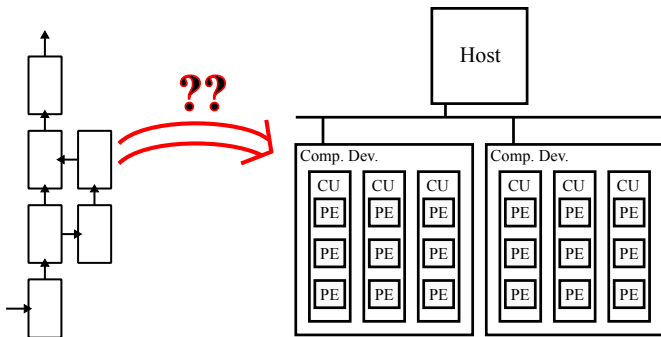
SIMD

centralisé

complexité de la gestion de la mémoire

Deux axes :

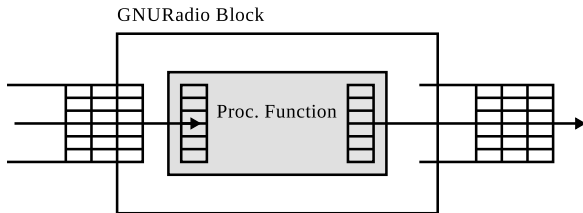
- conception d'une opération unique
- intégration dans un environnement



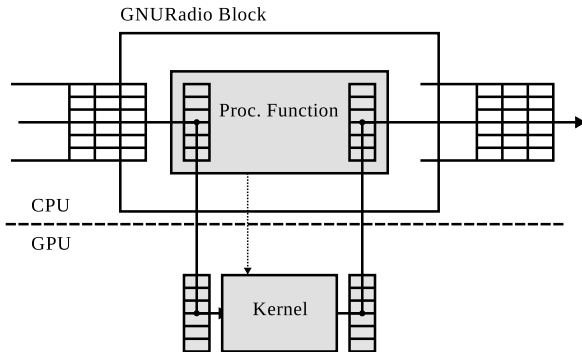
Sommaire

- 1 Contexte et objectif
- 2 Approches pour la conception
- 3 Approches pour l'intégration
- 4 Expérimentations
- 5 Conclusion

Approche classique



Approche classique



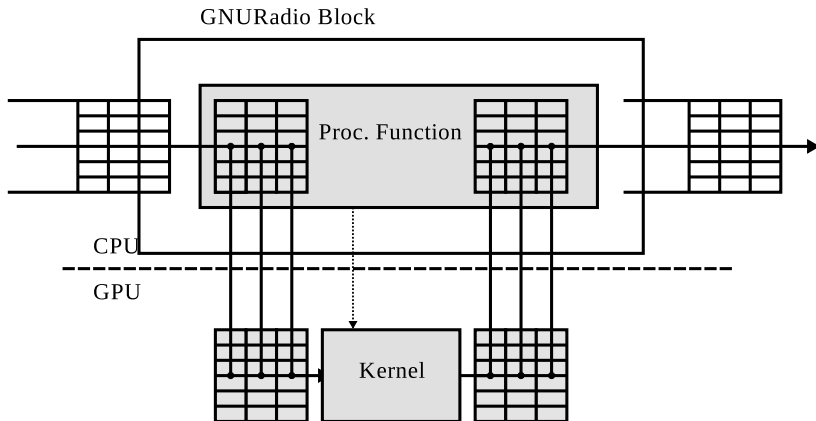
Avantages :

- Très efficace pour les vecteurs larges
- Utilisation de bibliothèques GPGPU

Inconvénients :

- inefficace pour les petits vecteurs
- impossible pour certaines opérations (IIR)

Extension de l'approche classique



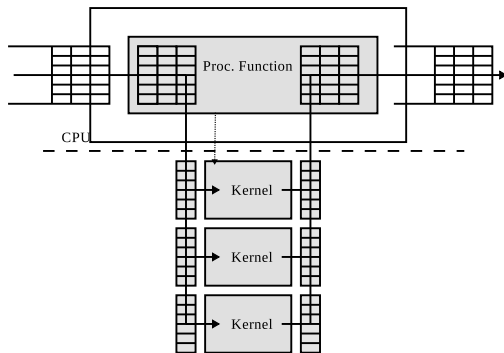
Avantages :

- Permet d'optimiser l'utilisation du GPU pour les petits vecteurs
- Utilisation de bibliothèques toujours possible

Inconvénients :

- Toujours impossible pour certaines opérations (IIR)
- Augmente la latence

Approche à gros grain



Avantages :

- meilleure utilisation du GPU
- efficace quelque soit la taille du vecteur

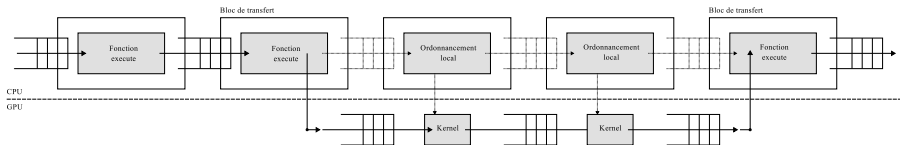
Inconvénients :

- Latence plus importante
- Plus d'occupation mémoire

Sommaire

- 1 Contexte et objectif
- 2 Approches pour la conception
- 3 Approches pour l'intégration
- 4 Expérimentations
- 5 Conclusion

Approche distribuée



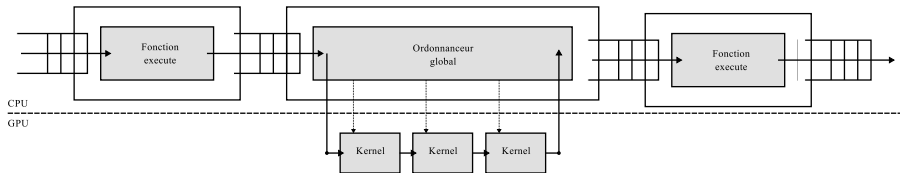
Avantages :

- Réutilisation de l'environnement (intégration transparente)
- Flexible

Inconvénients :

- Buffer OpenCL lourd à gérer
- Peu efficace (surcout important)

Approche centralisée



Avantages :

- Peu de surcout lié à la gestion de l'exécution
- Utilisation mémoire optimisée

Inconvénients :

- Moins flexible (ordonnancement figé)

Sommaire

- 1 Contexte et objectif
- 2 Approches pour la conception
- 3 Approches pour l'intégration
- 4 Expérimentations
- 5 Conclusion

Plateforme de test

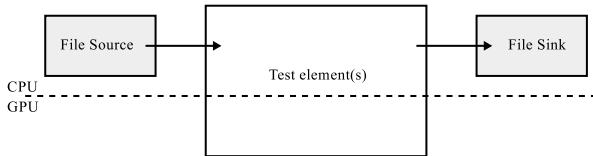
Plateforme de test

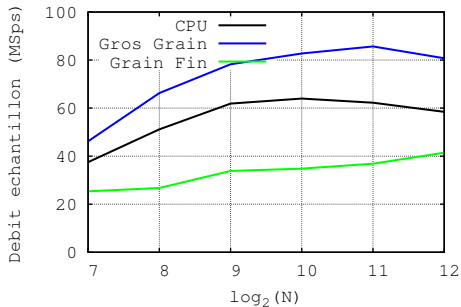
- Intel Core i5 760 CPU (4 cores, 2.8GHz, 8MB cache)
- 4GB de mémoire DDR3
- Linux 2.6.36 kernel
- NVidia GTS 450 GPU, Asus DirectCU Card, 1GB de mémoire DDR5

Applications

Applications

- 3 opérations individuelles :
 - FFT (algorithme adapté existant)
 - IIR (récursivité : pas d'algorithme)
 - Mapping (faible complexité)
- Une séquence des trois opérations
- Indicateur utilisé : débit échantillons en Méch/s

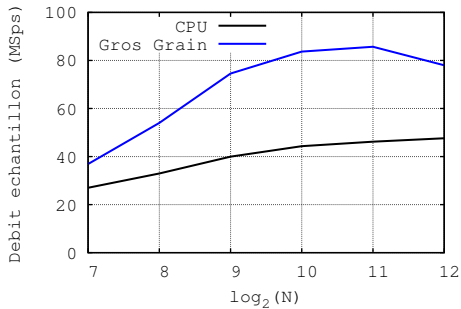




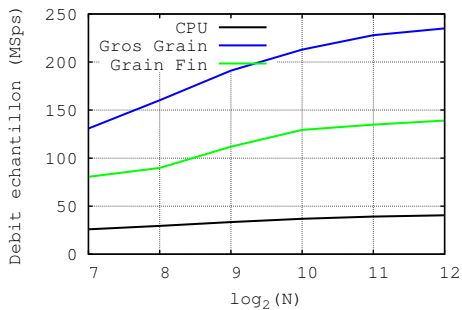
- La solution classique est inefficace
- La solution classique adaptée reste moins efficace que le CPU
- La méthode à gros grain permet une amélioration des performances
- Les performances sont faussées par la prédominance du transfert mémoire
- Monitoring GPU :
 - 10% pour la solution classique
 - 98% pour les solutions adaptées

Opérations individuelles

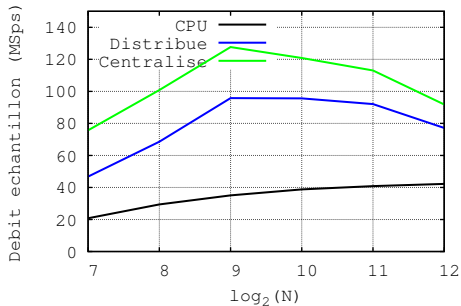
IIR



Demapping



Multitasking



- Pas d'utilisation du GPU multitâche (exécution séquentielle)
- 20% gain for 3 tasks for size 1024
- Gain d'un facteur 3-4
- La diminution des performances est liée à une taille de FIFO inadaptée
- L'intégration centralisée est visiblement plus efficace

Sommaire

- 1 Contexte et objectif
- 2 Approches pour la conception
- 3 Approches pour l'intégration
- 4 Expérimentations
- 5 Conclusion

Conclusion et perspectives

Contributions

- Étude de plusieurs solutions pour l'intégration du GPU dans la radio logicielle
 - Définition d'une méthode à gros grains pour la conception des opérations
 - Présentation d'une méthode d'intégration centralisée

Perspectives

- Utiliser les capacités multitâches pour réduire l'occupation mémoire
- Étude dans le cadre de systèmes embarqués (ARM Mali T604)