Technologies des réseaux et NTICs

2e Séminaire calcul hybride Aristote

&

Kick off meeting du projet OpenGPU

Jeudi 25 mars 2010

Organisation : Aristote

Coordination scientifique : ***Stéphane Requena (GENCI)***, ***Jean-Noël de Galzain (IF Research-WALLIX)*** , ***Eric Mahé (Minds Planet)***

Amphithéâtre Gay-Lussac, École Polytechnique, Palaiseau

http://www.aristote.asso.fr Contact : **info@aristote.asso.fr**

Edition du 7 germinal an CCXVIII (*vulg.* 27 mars 2010) ©2010 Aristote

ARISTOTE Association Loi de 1901. Siège social : CEA-DSI CEN Saclay Bât. 474, 91191 Gif-sur-Yvette Cedex.
Secrétariat : Aristote, École Polytechnique, 91128 Palaiseau Cedex.
Tél. : +33(0)1 69 33 99 66 Fax : +33(0)1 69 33 99 67 Courriel : Marie.Tetard@polytechnique.edu
Site internet http://www.aristote.asso.fr

Table des matières

Chapitre 1

Programme de la journée

1.1 2e Séminaire Calcul Hybride Aristote

Ce séminaire « Calcul Hybride » est organisé par le groupe de travail Gus'G/Calcul Hybride de l'association Aristote en collaboration avec le projet OpenGPU, il fait suite à un premier séminaire organisé fin 2008 qui avait rassemblé plus d'une centaine de participants.

Depuis cette date, l'utilisation d'accélérateurs de calcul basés sur des processeurs graphiques (dits GPU ou Graphics Processing Unit) en complément des processeurs traditionnels n'a cessé de progresser, poussée notamment par des évolutions matérielles (meilleure performance en double précision, support de la norme IEEE, mémoire ECC, ...) mais aussi logicielles constantes.(évolutions de CUDA, développement de l'approche HMPP et apparition d'un standard de programmation des architectures hybrides : OpenCL). Cet engouement est particulièrement visible en France où s'est développé un écosystème hybride riche permettant de rassembler à la fois des ressources hybrides de calcul importantes (de la station de travail jusqu'aux supercalculateurs hybrides de GENCI et de TOTAL), des éditeurs de logiciels et fournisseurs de services innovants, des projets fédérateurs comme OpenGPU et un nombre croissant d'utilisateurs académiques et industriels issus de domaines scientifiques variés.

Ce séminaire donnera l'occasion de faire un point sur les offres matérielles et logicielles mais aussi d'écouter des témoignages d'utilisateurs qui présenteront leur retour d'expériences quant à l'utilisation des accélérateurs graphiques, il sera de plus complété dans l'après midi par le kick off meeting du projet OpenGPU.

1.2 Kick off du projet OPEN GPU

Pour répondre aux besoins croissants de puissance de calcul des applications informatiques émergentes, les GPUs (Graphics Processing Unit) constituent une alternative très prometteuse à l'utilisation des CPUs (Central Processing Unit) classiques. Actuellement, leur utilisation reste confidentielle car freinée par la difficulté à exploiter ces architectures complexes sans standard réel.

Le projet OpenGPU vise à construire le 1er pôle européen de recherche et de développement dans le domaine des architectures hybrides (CPU + GPU). Basé en Ile de France, il est destiné à attirer des acteurs étrangers autour des 19 partenaires français contributeurs de ce projet.

Labellisé par le pôle de compétitivité SYSTEM@TIC PARIS-REGION et le Pôle Cap Digital, et piloté par WALLIX, leader français des solutions de traçabilité informatique et de gestion des accès

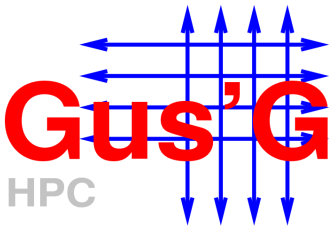
sécurisés, en lien avec Ter@tec, les sociétés Bull et AS+, ce projet veut capitaliser sur la puissance et le ratio puissance / consommation des GPUs pour atteindre un triple objectif :

Construire une plateforme intégrée et ouverte d'outils Open Source d'aide à la parallélisation du code existant, basée sur le standard Open CL, Expérimenter les gains de cette parallélisation au travers de démonstrateurs industriels et académiques, Construire des architectures matérielles et logicielles adéquates pour l'exploitation de ces nouvelles puissances de calcul et l'amélioration de la consommation électrique.

Photo ©Philippe Lavialle 2010

1.3 Programme du 25 mars 2010 (matin)

	2e séminaire Aristote sur le Calcul Hybride	
8h30-9h10	*Accueil-café*	
9h10-9h20	Présentation de la journée	Stéphane Requena (GENCI)
9h20-10h15	**Roadmap matérielle & logicielle**	
	- GPU Computing, NVIDIA Stratégie & Ecosystème	Jean-Christophe Baratault (NVIDIA)
	- Fusion et OpenCL	Bruno Stefanizzi (AMD)
	- Présentation OpenCL et HMPP	Romain Dolbeau (CAPS Entreprise)
10h15-11h15	Retours d'expériences de l'appel à projets CAPS/GENCI 2009	Laurent Bertaux (CAPS Entreprise)
11h15-11h40	*Pause café*	
	Portage d'une application de propagation d'ondes sismisques en multi-GPU	Dimitri Komatitsch (Université de Pau/INRIA)
	Retour d'expériences du calcul hybride au CEA DAM	Guillaume Colin de Verdière (CEA DAM/DIF)
12h30-14h00	*Repas* (salle «aquarium»)	



1.4 Programme du 25 mars 2010 (après-midi)

	Kick off meeting du projet OpenGPU	
14h00-14h30	**Présentation du projet OpenGPU**	
	- Historique	Jean François Lavignon (BULL)
	- Verrous technologiques et Objectifs	Eric Mahé (Minds Planet)
	- Organisation et communication	Jean-Noël de Galzain (Wallix)
14h30-16h00	**Outils et méthodes du projet OpenGPU**	
	- Présentation générale du SP2	Ronan Keryell (HPC Project)
	- GCC, POCC, MELT GCC et la génération de code pour GPU *via* des greffons	Basile Starynkevitch (CEA LIST), Cedric Bastoul (INRIA Saclay/Univ. Paris Sud XI)
	- PIPS, PA4RALL, Spear DE Les outils source à source et la génération de code CUDA et Open CL	Ronan Keryell (HPC Project)
	- MDM Nouveaux algorithmes pour architectures hybrides	Frédéric Magoulès, (ECP/CRSA)
	- Scilab Exploitation des GPU à partir de Scilab	Sylvestre Ledru (Consortium Scilab/ Digiteo)
	- HMPP Programmation parallèle multi-core hétérogène : HMPP Workbench	Laurent Morin (CAPS Entreprise)
	-OCAML Extension de OCaml pour le calcul scientifique sur GPU	E. Chailloux (LIP6)
16h00-16h15	*Pause*	

16h15-16h45	**Architecture matérielle et logicielle du projet OpenGCU**	
	-BULL, CEA, GENCI Systèmes hybrides : une architecture pour le Green IT	Jean François Lemerre (BULL)
16h45-17h45	**Démonstrateurs industriels et académiques**	
	- NUMTECH L'avenir du risque environnemental : le GP-GPU ?	J. Bost (NUMTECH)
	- CEA DAM Retour d'expérience du calcul hybride	Guillaume Colin de Verdière (CEA DAM)
	- INRA Génomique et GPU	Jean-Michel Batto (INRA)
	- IBISC Recherche à grande échelle d'ARNs non-codants	Fariza Tahi (Université d'Evry)
	- ESI Group Utilisation de Co-processeurs Graphiques dans des Codes Industriels de Simulation Numérique	Antoine Petitet (ESI Group)
	- Thales Calcul Temps-Réel embarqué sur GPU	Michel Barreteau (Thales)
	- Wallix GPU et sécurité	Jean-Noël de Galzain (Wallix)
17h45-18h00	**Conclusions de la journée**	
18h00	**Cocktail (salon d'honneur de l'École Polytechnique)**	

Chapitre 2

Présentations (matin)

2.1 Stéphane Requena (GENCI)

Ouverture du séminaire

Photos ©Aristote 2010

GENCI

aristote OpenGPU

2ème séminaire « Calcul Hybride »

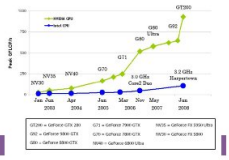
Introduction

GENCI

Evolutions matérielles depuis fin 2008

- Performances en hausse
  - 512 à 1600 coeurs
  - Plusieurs niveaux cache
  - Perf en SP
  - Meilleure perf en DP
  - Capacité et débit memoire

- Schedulers threads + performants
- Support mémoire ECC
- Support norme IEEE 754, …
- Solutions intégrées HPC



GENCI

Evolutions logicielles depuis fin 2008

- Evolution continue des langages de programmation
  - Support transferts asynchrones, kernels concurrents, Fortran, C ++, …
  - CUDA coté nVIDIA (cuda 3.0 annoncé lundi)
  - ATI Stream SDK coté AMD
- Apparition d'un standard : OpenCL
  Khronos group avec entre autre AMD, Apple, Intel, IBM, nVIDIA, Nokia, ST Micro, Samsung, …

- Fort développement de la suite HMPP de CAPS Entreprise

GENCI

Le GPU Computing est disponible partout !

- Depuis le poste de travail

- Jusqu'aux supercalculateurs hybrides

cea GENCI

TOTAL

Titane : Cluster BULL de 2136 processeurs Intel NH et 48 serveurs nvIDIA Tesla (192 GPUs)

Rostang : Cluster SGI de 4096 processeurs Intel HT/NH et 64 serveurs nvIDIA Tesla (256 GPUs)

GENCI

Ecosystème hybride en France

- La France dispose ainsi d'un écosystème hybride TRES riche
  - Utilisateurs académiques et industriels
  - Editeurs de logiciels et prestataires de services
  - Labos et centres de calcul équipés de solutions hybrides
- Et TRES actif
  - Projets de R&D hybride : ANR, Europe, … avec expertise reconnue
  - Projets fédérateurs : OpenGPU
  - Grands challenges scientifiques
  - Appels à projets
  - Prix scientifiques tel que le lauréat du prix BULL Joseph Fourier 2009

GENCI

Agenda de la matinée

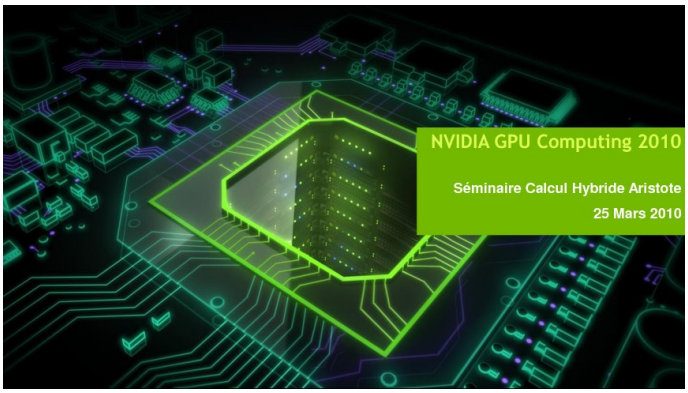
- 8h30-9h15 Accueil café
- 9h15-10h15 Roadmap matérielle & logicielle
  - GPU Computing, NVIDIA Stratégie & Ecosystème par JC. Baratault (nVIDIA)
  - Fusion et OpenCL par B. Stefanizzi (AMD)
  - Présentation OpenCL et HMPP par R.Dolbeau (CAPS Entreprise)
- 10h15-11h15 Retours d'expériences de l'appel à projets CAPS/GENCI 2009 par L. Bertaux (CAPS Entreprise)
- 11h15-11h40 Pause café
- 11h40-12h05 Portage d'une application de propagation d'ondes sismiques en multi-GPU par D. Komatitsch (Université de Pau/INRIA)
- 12h05-12h30 Retour d'expériences du calcul hybride au CEA DAM par G. Colin de Verdière (CEA DAM/DIF)
- 12h30-14h Repas

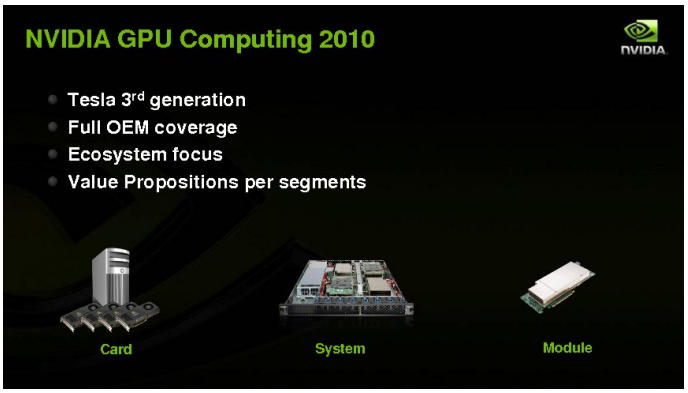
2.2 Roadmap matérielle & logicielle

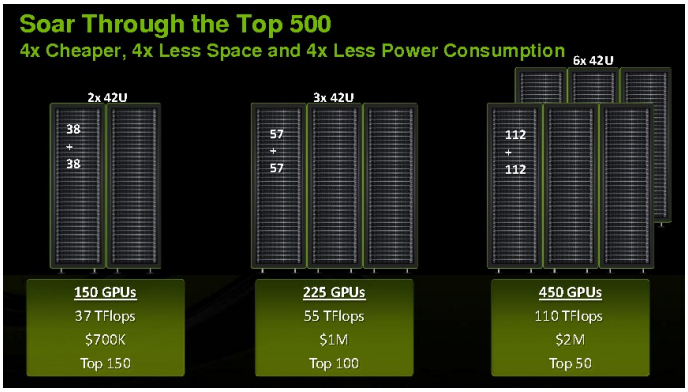
2.2.1 Jean-Christophe Baratault (NVIDIA)

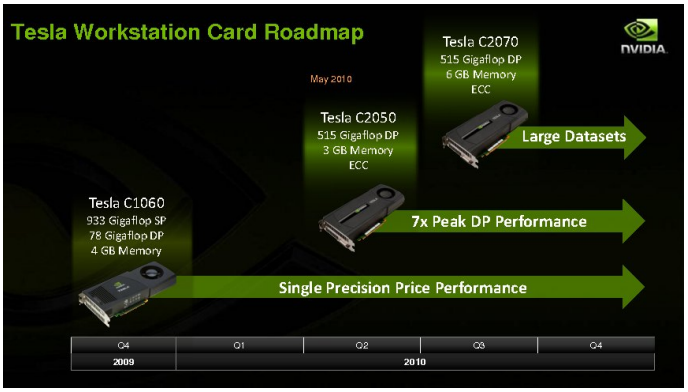
GPU Computing, NVIDIA Stratégie & Ecosystème

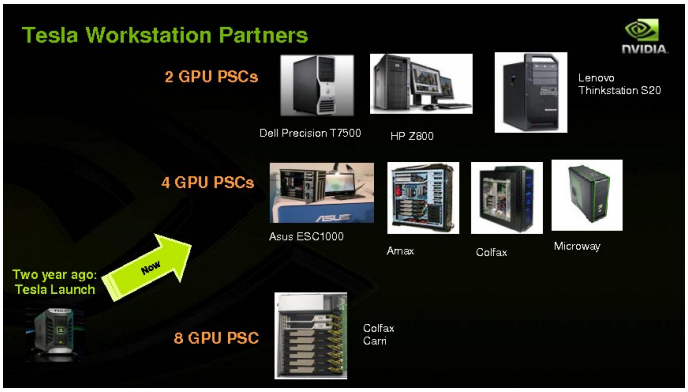
En 2010 NVIDIA introduit sa 3e génération de solutions HPC sur architecture GPU et dispose d'un écosystème complet de services associés. La présentation détaille les segments de marché adressés par NVIDIA, les outils de développement ainsi que les partenaires industriels.

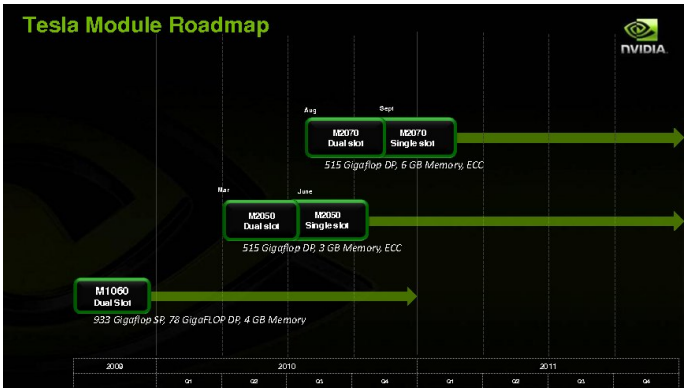

NVIDIA GPU Computing 2010
Séminaire Calcul Hybride Aristote
25 Mars 2010

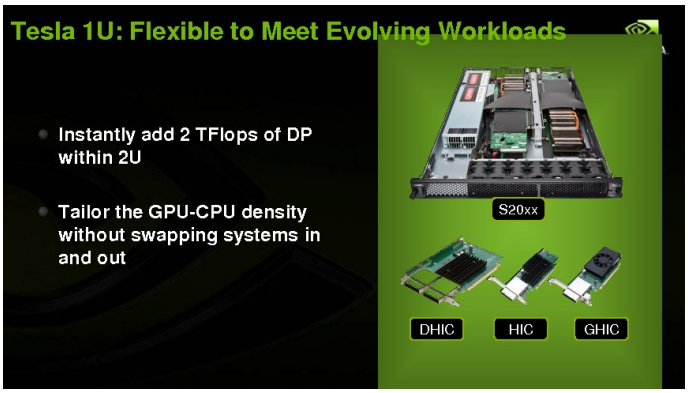

NVIDIA GPU Computing 2010
Tesla 3rd generation
Full OEM coverage
Ecosystem focus
Value Propositions per segments
Card
System
Module

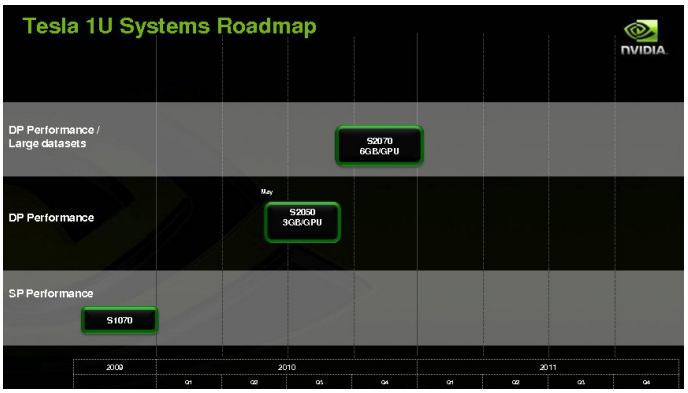

Soar Through the Top 500
4x Cheaper, 4x Less Space and 4x Less Power Consumption
2x 42U
38
+
38
3x 42U
57
+
57
6x 42U
112
+
112
150 GPUs
37 TFlops
$700K
Top 150
225 GPUs
55 TFlops
$1M
Top 100
450 GPUs
110 TFlops
$2M
Top 50

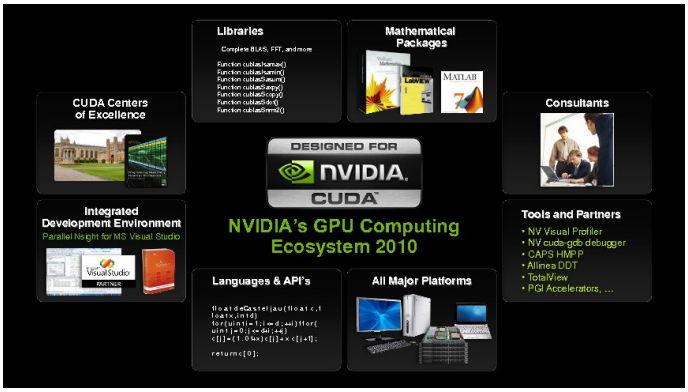

Tesla Workstation Card Roadmap
Tesla C2070
515 Gigaflop DP
6 GB Memory
ECC
May 2010
Tesla C2050
515 Gigaflop DP
3 GB Memory
ECC
Large Datasets
Tesla C1060
933 Gigaflop SP
78 Gigaflop DP
4 GB Memory
7x Peak DP Performance
Single Precision Price Performance
Q4
2009
Q1
Q2
Q3
Q4
2010

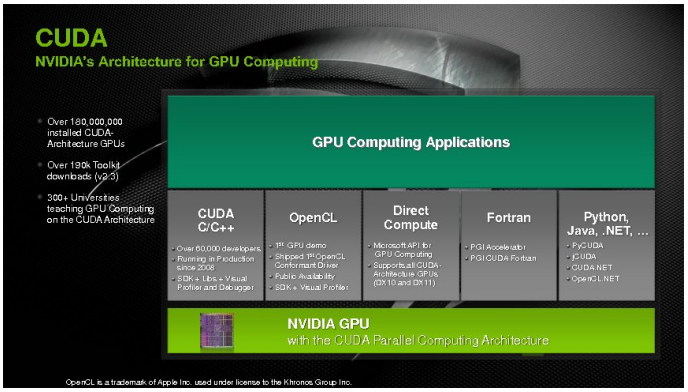

Tesla Workstation Partners
2 GPU PSCs
Dell Precision T7500
HP Z800
Lenovo
Thinkstation S20
4 GPU PSCs
Asus ESC1000
Amax
Colfax
Microway
Two year ago:
Tesla Launch
Now
8 GPU PSC
Colfax
Carri

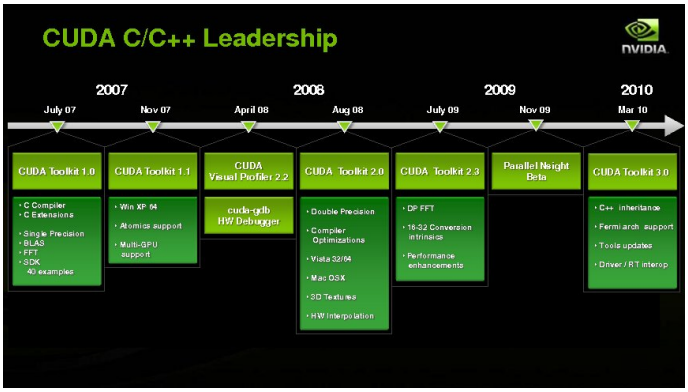

Tesla Module Roadmap
Aug
Sept
M2070
Dual slot
M2070
Single slot
515 Gigaflop DP, 6 GB Memory, ECC
Mar
June
M2050
Dual slot
M2050
Single slot
515 Gigaflop DP, 3 GB Memory, ECC
M1060
Dual Slot
933 Gigaflop SP, 78 GigaFLOP DP, 4 GB Memory
2009
2010
2011

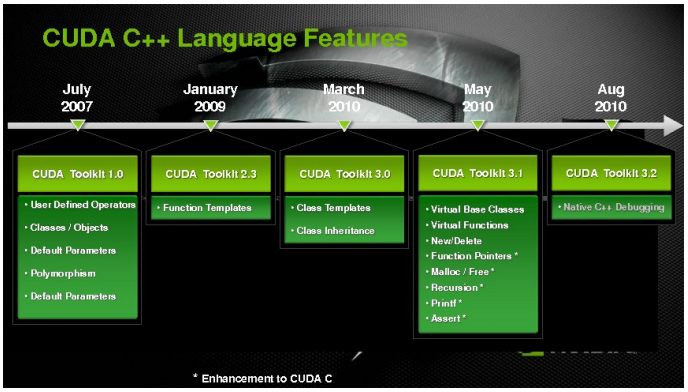

Tesla 1U: Flexible to Meet Evolving Workloads
Instantly add 2 TFlops of DP within 2U
Tailor the GPU-CPU density without swapping systems in and out
S20xx
DHIC
HIC
GHIC

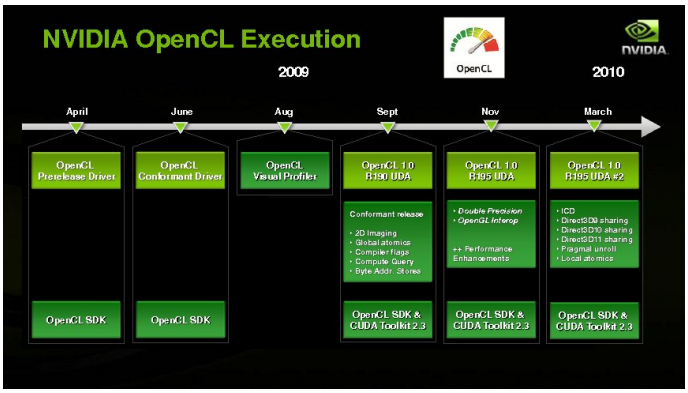

Tesla 1U Systems Roadmap
NVIDIA
DP Performance / Large datasets
S2070 6GB/GPU
DP Performance
May
S2050 3GB/GPU
SP Performance
S1070
2009
2010
2011

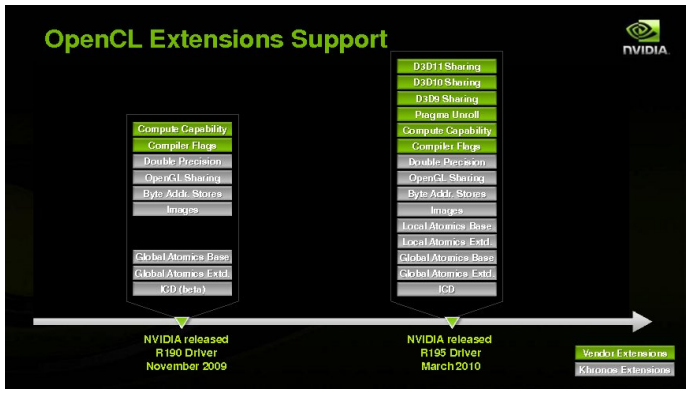

Libraries
Mathematical Packages
CUDA Centers of Excellence
Consultants
DESIGNED FOR NVIDIA CUDA
Integrated Development Environment
Parallel Nsight for MS Visual Studio
NVIDIA's GPU Computing Ecosystem 2010
Tools and Partners
• NV Visual Profiler
• NV cuda-gdb debugger
• CAPS HMPP
• Allinea DDT
• TotalView
• PGI Accelerators, ...
Languages & API's
All Major Platforms

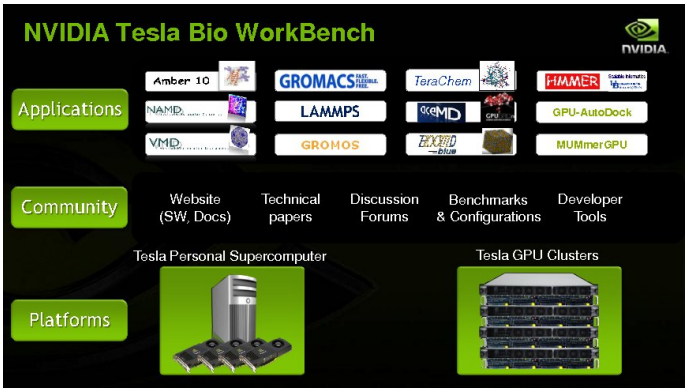

CUDA
NVIDIA's Architecture for GPU Computing
Over 180,000,000 installed CUDA-Architecture GPUs
Over 190k Toolkit downloads (v2.3)
300+ Universities teaching GPU Computing on the CUDA Architecture
GPU Computing Applications
CUDA C/C++
OpenCL
Direct Compute
Fortran
Python, Java, .NET, ...
NVIDIA GPU with the CUDA Parallel Computing Architecture
OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

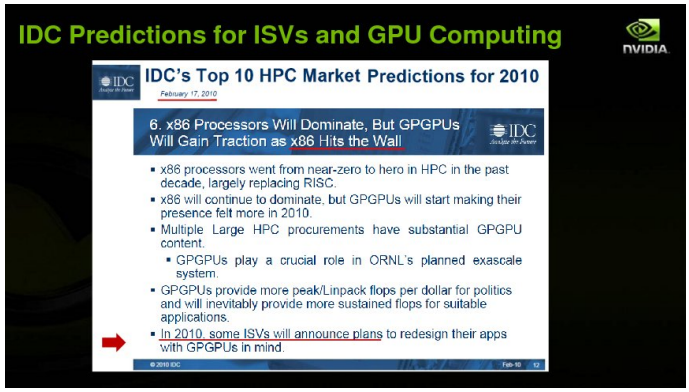

CUDA C/C++ Leadership
NVIDIA
2007
2008
2009
2010
July 07
Nov 07
April 08
Aug 08
July 09
Nov 09
Mar 10
CUDA Toolkit 1.0
CUDA Toolkit 1.1
CUDA Visual Profiler 2.2
CUDA Toolkit 2.0
CUDA Toolkit 2.3
Parallel Nsight Beta
CUDA Toolkit 3.0
cuda-gdb HW Debugger

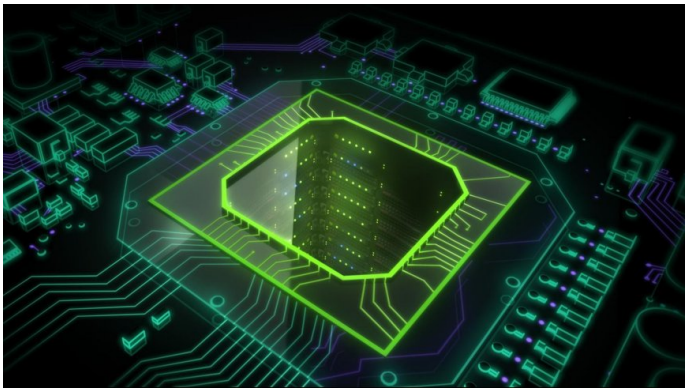

CUDA C++ Language Features
July 2007
January 2009
March 2010
May 2010
Aug 2010
CUDA Toolkit 1.0
CUDA Toolkit 2.3
CUDA Toolkit 3.0
CUDA Toolkit 3.1
CUDA Toolkit 3.2
• User Defined Operators
• Classes / Objects
• Default Parameters
• Polymorphism
• Default Parameters
• Function Templates
• Class Templates
• Class Inheritance
• Virtual Base Classes
• Virtual Functions
• New/Delete
• Function Pointers *
• Malloc / Free *
• Recursion *
• Printf *
• Assert *
• Native C++ Debugging
* Enhancement to CUDA C


NVIDIA OpenCL Execution
OpenCL
NVIDIA
2009
2010
April
June
Aug
Sept
Nov
March
OpenCL Prerelease Driver
OpenCL Conformant Driver
OpenCL Visual Profiler
OpenCL 1.0 R180 UDA
OpenCL 1.0 R185 UDA
OpenCL 1.0 R195 UDA #2
Conformant release
• 2D Imaging
• Global atomics
• Compiler flags
• Compute Query
• Byte Addr. Stores
• Double Precision
• OpenGL Interop
++ Performance Enhancements
• ICD
• Direct3D9 sharing
• Direct3D10 sharing
• Direct3D11 sharing
• Pragma unroll
• Local atomics
OpenCL SDK
OpenCL SDK
OpenCL SDK & CUDA Toolkit 2.3
OpenCL SDK & CUDA Toolkit 2.3
OpenCL SDK & CUDA Toolkit 2.3


OpenCL Extensions Support
NVIDIA
Compute Capability
Compiler Flags
Double Precision
OpenGL Sharing
Byte Addr. Stores
Images
Global Atomics Base
Global Atomics Extd.
ICD (beta)
D3D11 Sharing
D3D10 Sharing
D3D9 Sharing
Pragma Unroll
Compute Capability
Compiler Flags
Double Precision
OpenGL Sharing
Byte Addr. Stores
Images
Local Atomics Base
Local Atomics Extd.
Global Atomics Base
Global Atomics Extd.
ICD
NVIDIA released R190 Driver November 2009
NVIDIA released R195 Driver March 2010
Vendor Extensions
Khronos Extensions


NVIDIA Tesla Bio WorkBench
NVIDIA
Applications
Amber 10
NAMD
VMD
GROMACS
LAMMPS
GROMOS
TeraChem
GPU-HMMER
GPU-AutoDock
MUMmerGPU
Community
Website (SW, Docs)
Technical papers
Discussion Forums
Benchmarks & Configurations
Developer Tools
Platforms
Tesla Personal Supercomputer
Tesla GPU Clusters


IDC Predictions for ISVs and GPU Computing
NVIDIA
IDC's Top 10 HPC Market Predictions for 2010
February 17, 2010
6. x86 Processors Will Dominate, But GPGPUs Will Gain Traction as x86 Hits the Wall
x86 processors went from near-zero to hero in HPC in the past decade, largely replacing RISC.
x86 will continue to dominate, but GPGPUs will start making their presence felt more in 2010.
Multiple Large HPC procurements have substantial GPGPU content.
GPGPUs play a crucial role in ORNL's planned exascale system.
GPGPUs provide more peak/Linpack flops per dollar for politics and will inevitably provide more sustained flops for suitable applications.
In 2010, some ISVs will announce plans to redesign their apps with GPGPUs in mind.




Extra Slides
NVIDIA

Tesla C-series Display Features

Why does Tesla C2050 have DVI out?
Even researchers and engineers need to see their work!

	Feature	Tesla C2050
Performance	Double Precision FP Perf	Same as Quadro
	Geometry Perf (Tris/Sec)	Much lower than Quadro
	Viewperf Perf (Geomean)	Lower than Quadro
Graphics	Level	Baseline Level
	12-bit/30-bit	No
	Quad-Buffered Stereo	No
Features	Display	Single DVI-I Dual Link
	SLI MultiOS	No
ISV Certs / SW	Workstation ISV Certs	No
	Tesla ISV Certs	Yes

Tesla: High Performance Computing

Quadro: Visual Computing

Tesla C-series Support Matrix

	C-Series: Active Fan Sink
Windows Operating Systems	Windows 7 Windows Vista Windows XP
Linux Operating Systems	RHEL Enterprise Desktop SUSE Enterprise Desktop openSUSE Ubuntu Desktop Edition
Display Output (T20 Series only)	Single DVI-I Connector
OpenGL Perf (T20 Series Only)	GeForce Performance
TCC Driver for Windows	Yes, on Desktop OS
Qualified Chassis	Standard PC and Workstation Chassis

Datacenter Features only on M and S products:
- InfiniBand acceleration
- Thermal monitoring
- Server OSes

CPUs and M1060s in servers have passive sinks for airflow

CPUs and C1060s in workstation have active fans for airflow

Tesla M-Series is Engineered for Servers

	M-Series: Passive Heatsink	C-Series: Active Fansink
Windows Operating Systems	Win Server 2008 R2 Win Server 2008	Windows 7 Windows Vista Windows XP
Linux Operating Systems	RHEL Enterprise Server SUSE Enterprise Server openSUSE Ubuntu Server Edition	RHEL Enterprise Desktop SUSE Enterprise Desktop openSUSE Ubuntu Desktop Edition
Display Output (T20 Series only)	Some SKUs have DVI-I Connector	Single DVI-I Connector
OpenGL Perf (T20 Series Only)	Professional OpenGL Features and Performance	GeForce Performance
TCC Driver for Windows	Yes, on Server OS	Yes, on Desktop OS
Faster Transfers between Infiniband and GPU	Yes	No
Thermal Monitoring by Host System	Yes	No
System Information (NV-SMI)	Yes	No
GPU Monitoring with Ganglia	Yes	No
GPU Scheduling	Yes	No
Cluster SW Support (ROCKS, Platform, ClusterOS, Scyld)	Yes	No

Tesla S-Series 1U GPU Systems

	S2050	S2070
Processors	4 Tesla 20-series GPUs	
Number of Cores	1792 per 1U (448 / GPU)	
Single precision performance	4120 GFlops per 1U *1030 GFlops / GPU*	
Double precision performance	2060 GFlops per 1U *515 GFlops / GPU*	
GPU Memory	12 GB per 1U *3 GB / GPU* *2.625 GB with ECC on*	24 GB per 1U 6 GB / GPU 5.25 GB with ECC on
Memory Interface	GDDR5	
System I/O	2x PCIe x16 Gen2 HICs OR 2x PCIe x8 Gen2 HICs OR 1x PCIe x16 Gen2 DHIC	
Power	1200 W (max)	
Available	May 2010	Q3 2010

2.2.2 Bruno Stefanizzi (AMD)

Fusion et OpenCL

OpenCL, le nouveau standard de programmation des architectures hybrides, permet une flexibilité et une abstraction inédite dans ce domaine, qui facilitera les changements futurs de celles-ci. Fusion représentant le changement majeur pour ces architectures par une intégration et interconnexion novatrice entre CPU et GPU, nous verrons comment OpenCL prépare les logiciels a cette nouvelle plateforme.

AMD/ATI Hybrid Computing

Bruno Stefanizzi
Senior MTS Engineer
March 2010

ATI STREAM TECHNOLOGY

1

September 2009
AMD is Changing the Game

*World's **FIRST** And **ONLY** DirectX11 OpenCL GPU*

2

ATI EverGreen GPU

The World's Most Powerful and Advanced GPU

- Accelerating PCs with nearly 3 teraFLOPS of compute power
- Ultimate immersion with DirectX® 11 and ATI Eyefinity Technology
- Extreme game play at high resolutions and maximum settings*

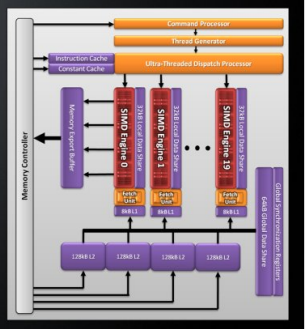
Compute Power SP	2.72 TFLOPS
Core Clock Speed	850 MHz
Stream Processors	1600
Memory Bandwidth	160Gb/s
Max/Idle Board Power	150W/27W
Compute Power DP	544 GFLOPS

3

Stream Computing View

2.72 Teraflops Single Precision,
544 Gigaflops Double Precision

- Compute Features:
  - Full Hardware Implementation of DirectCompute 11 and OpenCL™ 1.0
  - IEEE754 Compliance Enhancements
  - 32-bit Atomic Operations
  - Flexible 32kB Local Data Shares
  - 64kB Global Data Share
  - Global synchronization
  - Append/consume buffers



4

ATI Stream Development and Deployment

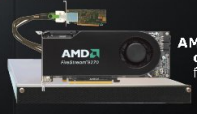
Development

Industry Standards: OpenCL and DirectCompute

Deployment

ATI Radeon™ graphics for consumer applications

ATI FirePro™ graphics for professional graphics



AMD FireStream™ compute accelerator for computation

5

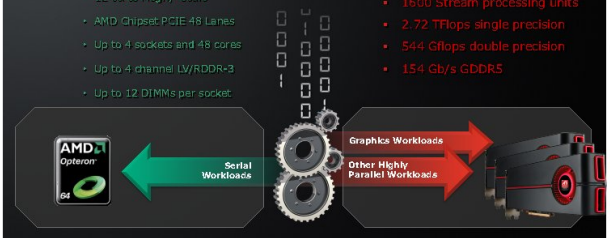
AMD Balanced Platform Advantage

AMD Opteron™ 6000

- 12 cores Magny-Cours
- AMD Chipset PCIE 48 Lanes
- Up to 4 sockets and 48 cores
- Up to 4 channel LV/RDDR-3
- Up to 12 DIMMs per socket

EverGreen GPU

- 1600 Stream processing units
- 2.72 TFlops single precision
- 544 Gflops double precision
- 154 Gb/s GDDR5



Delivers **optimal performance** for a wide range of platform configurations

6

Programming the CPU and GPU: ATI Stream SDK

7

Open Standards:

Accelerate

Maximize Developer Freedom and Addressable Market

Vendor specific Cross-platform limiters

- ~~Apple Display Connector~~
- ~~3dfx Glide~~
- ~~Nvidia CUDA~~
- ~~Nvidia Cg~~
- ~~Rambus~~
- ~~Unified Display Interface~~

Vendor neutral Cross-platform enablers

Digital Visual Interface | OpenCL™ | DirectX® | Certified DP | JEDEC | OpenGL®

OpenCL™ and DirectX® are emerging as the two most important standards for heterogeneous (CPU+GPU) compute

8

OpenCL: Game-Changing Development

Enabling Broad Adoption of GP-GPU Capabilities

Industry standard API
- Open, multiplatform development platform for heterogeneous architectures

The power of Fusion
- Leverages CPUs and GPUs for balanced system approach
- Not just another GPU or CPU exclusive proprietary

Broad industry support
- Created by architects from AMD, Apple, IBM, Intel, Nvidia, Sony, etc.
- Apple have released OpenCL support as part of Snow Leopard

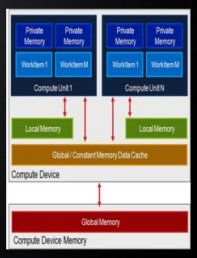
Momentum
- Enormous interest from mainstream developers
- Available free of charge

Fast track deployment
- Specification was ratified in 6 months and several implementations are now appearing
- OpenCL1.1 will be ratified in 2010

OpenCL Programming Model

Execution Model
- *Compute kernel* is basic unit of execution
- Execution can occur in-order or out-of-order
- Kernel can be *data-parallel* (GPU) or *task-parallel* (CPU)
- N-dimensional *execution domain* for kernels
- Ability to group *work-items* into *work-groups* for sync/comm

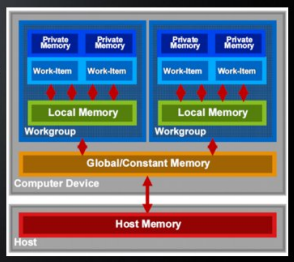
Memory Model
- Multi-level memory model: *private*, *local*, *constant* and *global*



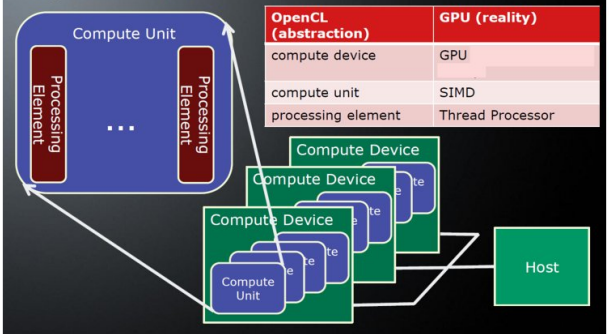
10

OpenCL Memory Model

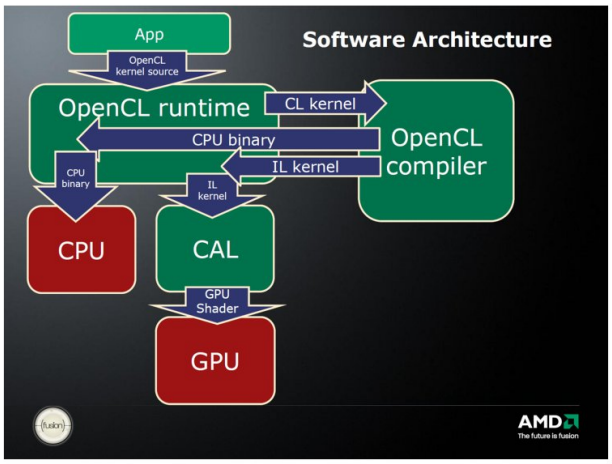
- Shared Memory Model
  - Relaxed consistency
- Multiple distinct address spaces
  - Private
    - Per work-item
  - Local
    - Shared within workgroup
  - Local Global/Constant
    - Not synchronized
  - Host Memory
    - On the CPU



OpenCL™ view



OpenCL (abstraction)	GPU (reality)
compute device	GPU
compute unit	SIMD
processing element	Thread Processor



How OpenCL™ maps on CPU

OpenCL (abstract)	CPU (reality)
compute device	PC system
compute unit	processor core
processing element	processor core
work-item	software thread
work-group	hardware thread
local memory	CPU data L1 (per work-group allocated RAM)
global memory	system RAM
private memory	per work-item allocated system RAM
scalar variable	CPU register or stack space
vector variable	SSE register or stack space
image	(not yet) array in system RAM

AMD The future is fusion

How OpenCL™ maps onto GPU

OpenCL (abstract)	ATI
work-item	CS instance, pixel, vertex
work-group	group of wavefronts (waves)
local memory	Local Data Share *
private memory	scratch memory (x[] in CAL/IL)
global memory	global buffer (g[] in CAL/IL)
scalar variable	single or double component of a 128-bit register
vector variable	one or more 128-bit registers
image	texture ** (not yet)

AMD The future is fusion

Extensions

- OpenCL supports an extension mechanism
- 1.0 extensions include
  - Atomics functions to global and local memory
  - Select rounding mode for a group of instructions at compile time
  - Double precision
  - Interop. - OpenGL and DX
  - Images

AMD The future is fusion

OpenCL benefits to you and your customer

- Reduces cost of development
  - Avoid the need to have two codebases (CPU and GPU)
  - Avoid the need to have to always write specialized SSE2 code
  - Avoid the need to have to write specialized GPU shader code
- Efficiencies for the customer
  - Utilize all the compute using GPU and CPU
  - More flexibility
    - eg CPU render farm you can easily switch to "CPU" rendering
    - Ideal for building Tools

AMD The future is fusion

OpenCL1.1

Expected Timeframe for ratification: May 2010

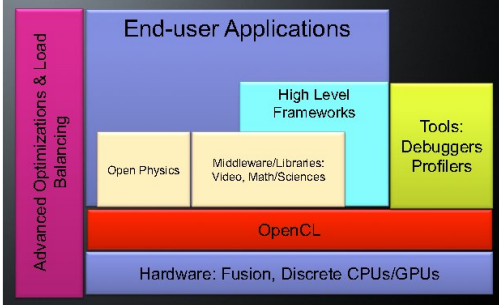
- Summary of what is expected in the specification:
- Atomics, byte address both become core
- User events and event callbacks
- Sub buffers and regions
- Global offset for launch
- Thread safety for API calls (except setKernelArg)

Summary of optional extensions:

- GL sharing will include cubemaps and mipmapped textures
- Floating point rounding modes (requiring FMA)
- Mapping a GL sync object to OpenCL event

AMD The future is fusion

OpenCL and the Ecosystem



SDK High-level Development Tools

GPU/CPU - accelerated Libraries
- ACML-GPU: AMD Core Math Library for the GPU/CPU

Performance Analysis Tools
- Standard CPU profiling tools
- GDB and/or printf()s
- Stream KernelAnalyzer: Static analysis tool for OpenCL kernels
- MSVS 2008-integrated profiler

3rd Party Tools / Libraries
- OpenCL
- CAPS

ATI Stream SDK v2.0 (Production): OpenCL™ For Multicore x86 CPUs and GPUs

The Power of Fusion: Developers leverage heterogeneous architecture to enable superior user experience
- **First complete OpenCL™ development platform**
- **Certified OpenCL™ 1.0 compliant by the Khronos Group**[1]
- Write code that can scale well on multi-core CPUs and GPUs
- AMD delivers on the promise of support for OpenCL™, with both high-performance CPU and GPU technologies
- Available for download now – includes documentation, samples, and developer support

Product Page: http://developer.amd.com/stream

[1] Conformance logs submitted for the ATI Radeon™ HD 5800 series GPUs, ATI Radeon™ HD 5700 series GPUs, ATI Radeon™ HD 4800 series GPUs, ATI FirePro™ V8700 series GPUs, AMD FireStream™ 9200 series GPUs, ATI Mobility Radeon™ HD 4800 series GPUs and x86 CPUs with SSE3.

ATI Stream SDK v2.0 (Production): Key Features

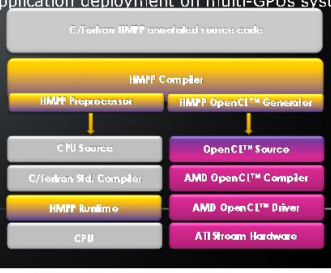
- **New:** Support for OpenCL™ ICD (Installable Client Driver)
- **New:** Support for atomic functions for 32-bit integers
- **New:** Microsoft® Visual Studio® 2008-integrated ATI Stream Profiler performance analysis tool
- **Preview:** Support for OpenCL™ / OpenGL® interoperability
- **Preview:** Support for OpenCL™ / Microsoft® DirectX® 10 interoperability
- **Preview:** Support for double-precision floating point basic arithmetic in OpenCL™ C kernels
- Added support for ATI Radeon™ HD 5970 GPU[1]
- Supported OSes:
  Microsoft® XP / Vista® / 7 and Linux® openSUSE™ 11.0 / Ubuntu® 9.04
- Supported Compilers:
  Microsoft® Visual Studio® 2008 Pro / GCC 4.3 or later / ICC 11.x

Product Page: http://developer.amd.com/stream

[1] Conformance logs submitted for the ATI Radeon™ HD 5800 series GPUs, ATI Radeon™ HD 5700 series GPUs, ATI Radeon™ HD 4800 series GPUs, ATI FirePro™ V8700 series GPUs, AMD FireStream™ 9200 series GPUs, ATI Mobility Radeon™ HD 4800 series GPUs and x86 CPUs with SSE3.

OpenCL™ Backend for HMPP (Scheduled to be released)

- A compiler integrating OpenCL™ stream generator
  - Build portable CPU and GPU hardware specific computations
- C & Fortran programming directives
  - High level programming interface for scientific applications
- Runtime library
  - Ease application deployment on multi-GPUs systems



OpenCL GPGPU Benchmarks

http://www.sisoftware.net/

Device Name	Cores / Speed / Memory	Native Float / Double Performance	OpenCL Float / Double Performance	Comments
ATI Radeon HD 4850 New!	800 / 625MHz / 512MB	359.7 / 177.7 MPixel/s (CAL)	508.7 / 25.5[2] MPixel/s	OpenCL allows us to achieve even faster performance than CAL, 50% better which is just incredible!
ATI Radeon HD 5870 New!	1600 / GHz / 1GB	912 / 459.478 MPixel/s (CAL)	1588 / 69.509[2] MPixel/s	We see again ~50% gains in OpenCL versus CAL, the compiler doing better than us in optimising the code. Fantastic result!
nVidia GeForce 9600 GT	64 / 1.6GHz / 512MB	164.381 / 12.471[2] MPixel/s (CUDA)	153.104 / 11.783[2] MPixel/s	Tie: CUDA 7% faster on float test, 6% slower on double-emulation, great result for OpenCL.
nVidia GeForce 9500 GT	32 / 1.4GHz / 512MB	74.319 / 5.807[2] MPixel/s (CUDA)	70.754 / 5.488[2] MPixel/s	Tie: CUDA 5% faster on float test, 5% slower on double-emulation, great result for OpenCL.
nVidia ION	16 / 1.1GHz / 128MB	31.897 / 2.332[2] MPixel/s (CUDA)	30.250 / 2.208[2] MPixel/s	Tie: CUDA 5% faster on float test, 5% slower on double-emulation, great result for OpenCL.
nVidia GeForce 9400M G5	16 / 800MHz / 128MB	22.607 / 1.720[2] MPixel/s (CUDA)	18.9 / 1.589[2] MPixel/s	CUDA slightly faster on both, though by a minor amount ~15%, should improve with newer drivers.

Training and Related Resources

- Training Resources
  - Introductory Tutorial to OpenCL™
  - AMD Developer Inside Track: Introduction to OpenCL™
  - ATI Stream OpenCL™ Technical Overview Video Series
  - Porting CUDA to OpenCL™
  - Image Convolution Using OpenCL™ - A Step-by-Step Tutorial
  - OpenCL™ Tutorial: N-Body Simulation
- Related Resources
  - OpenCL™: The Open Standard for Parallel Programming of GPUs and Multi-core CPUs
  - The Khronos™ Group – OpenCL™ Overview Page
  - ATI Stream Profiler Product Page
  - ACML-GPU Product Page
  - ATI Stream Power Toys Product Page
  - ATI Stream Developer Articles & Publications
  - ATI Stream Developer Showcase
  - ATI Stream Developer Training Resources
  - KB75 - Tips and suggestions for running SiSoftware Sandra 2010 OpenCL™ GPGPU benchmarks
- ATI Stream SDK v2.0 Documentation

25 AMD The future is fusion

Win with AMD Fusion™

fusion

26 AMD The future is fusion

Today

Multi-core CPU

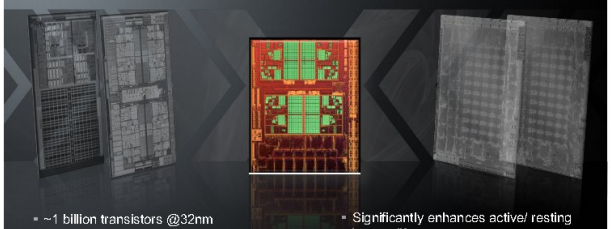
758 million transistors @45nm
Multi-tasking
Most compute tasks

TeraFLOPS-class GPU

2.15 billion transistors @40nm
3D OS
Multi-panel HD gaming
Full HD video and audio

27 AMD The future is fusion

Now the AMD Fusion™ Era of Computing Begins



- ~1 billion transistors @32nm in one design
- APU: Fusion of CPU & GPU compute power within one processor
- Significantly enhances active/ resting battery life
- High-bandwidth I/O

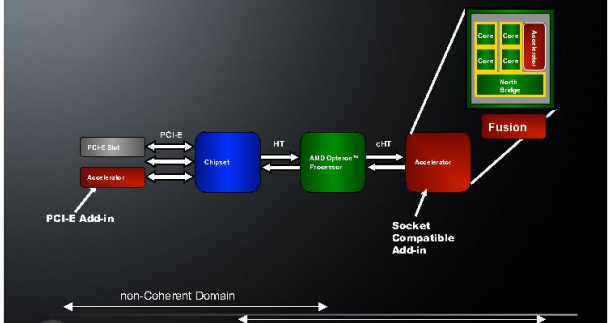
28 AMD The future is fusion

AMD Fusion™ Delivers Breakthroughs in 3 Dimensions

29 AMD The future is fusion

AMD's Accelerated Computing

Accelerator Integration Options



30 AMD The future is fusion

Fusion changes

- New Architecture
- New data exchange bandwidth (CPU<-> GPU)
- OpenCL keeps the abstraction

How OpenCL™ maps on CPU

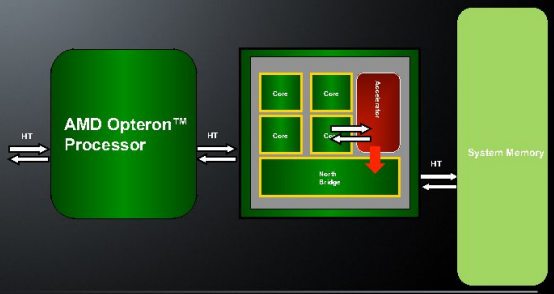
OpenCL (abstract)	CPU (reality)
compute device	PC system
compute unit	processor core
processing element	processor core
work-item	software thread
work-group	hardware thread
local memory	CPU data L1 (per work-group allocated RAM)
global memory	system RAM
private memory	per work-item allocated system RAM
scalar variable	CPU register or stack space
vector variable	SSE register or stack space
image	(not yet) array in system RAM

How OpenCL™ maps onto GPU

OpenCL (abstract)	ATI
work-item	CS instance, pixel, vertex
work-group	group of wavefronts (waves)
local memory	Local Data Share *
private memory	scratch memory (x[] in CAL/IL)
global memory	global buffer (g[] in CAL/IL)
scalar variable	single or double component of a 128-bit register
vector variable	one or more 128-bit registers
image	texture ** (not yet)

31 AMD The future is fusion

Fusion – Q1 2011 - llano



32 AMD The future is fusion

ONLY AMD!

33 AMD The future is fusion

Disclaimer and Attribution

DISCLAIMER
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.
The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.
AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.
AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
ATTRIBUTION
© 2008 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ATI, the ATI logo, Catalyst, CrossFireX, and Radeon and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other names are for informational purposes only and may be trademarks of their respective owners.
CAUTIONARY STATEMENT
This release contains forward-looking statements concerning revenue and other expectations which are made pursuant to the safe harbor provisions of the Private Securities Litigation Reform Act of 1995. Forward-looking statements are commonly identified by words such as "would," "may," "expects," "believes," "plans," "intends," "projects," and other terms with similar meaning. Investors are cautioned that the forward-looking statements in this release are based on current beliefs, assumptions and expectations, speak only as of the date of this release and involve risks and uncertainties that could cause actual results to differ materially from current expectations.

34 AMD The future is fusion

2.2.3 Romain Dolbeau (CAPS Entreprise)

Présentation OpenCL et HMPP

Cette présentation aborde tout d'abord le nouveau standard de programmation OpenCL proposé par le consortium Kronos. OpenCL vise la programmation des processeurs graphiques AMD et NVIDIA mais aussi celle des CPU multicoeur généralistes. Nous décrivons ensuite l'approche HMPP de programmation de haut niveau qui, à base de directives, permet la génération de code OpenCL.

CAPS

Heterogeneous Programming
OpenCL and HMPP
Aristote Presentation – 03/25/2010

Romain Dolbeau, CAPS entreprise

Outline

- Introduction to accelerator technologies
- OpenCL: overview, portability and performance
- OpenCL with HMPP

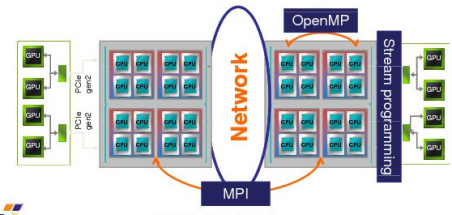
Accelerator Technologies
Introduction

Multicore/Manycore Architectures

- Multicore/manycore is mainstream
  - Intel Nehalem, Westmere
  - AMD Shangai and Magny-Cours
  - About 100GFLOPS for 100 W/core
- About performance not parallelism
- Manycore is now
  - Performance/Watt is the new efficiency scale
  - NVIDIA Tesla as coprocessor (4TFLOPS SP S1070), Fermi
  - AMD FireStream and Fusion project
  - Intel Larrabee: CPU-GPU convergence

Multiple Levels of Parallelism

- Amdahl's law is forever, all levels of parallelism need to be exploited
- Programming various hardware components of a node cannot be done separately



Stream Programming

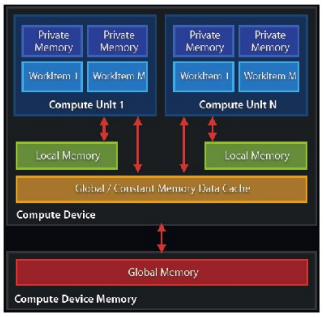
- Hardware languages and API
  - Brook, the pioneer (Stanford university)
  - AMD Brook+ with CAL/IL
  - NVIDIA CUDA compiler and libraries
  - Intel Ct library with compiler runtime
  - OpenCL standardization (Apple initiative, Khronos group)
- Directive-based compiler technologies
  - CAPS HMPP Workbench
  - PGI compiler

OpenCL
Overview

OpenCL Overview

- Open Computing Language
  - Royaltee-free, cross platform
  - C-based cross-platform programming interface
  - Subset of ISO C99 with language extensions
  - Data-, vector-, and task- parallel compute model
- Host-to-Compute Devices (GPUs) model
- Platform layer API and runtime API
  - Hardware abstraction layer, …
  - Manage resources
- Just-in-time compilation
  - Kernel source is sent to the driver
- Supported on most Oses

OpenCL Memory Hierarchy



Platform Layer API and Runtime

- Context
  - Collection of devices
- Workgroups and work-items
- Data buffer objects
- Command queues
  - Kernel execution commands
  - Memory commands (transfer or mapping)
  - Synchronization
- Platform Layer
  - Querying devices
  - Creating contexts

OpenCL Data Parallelism

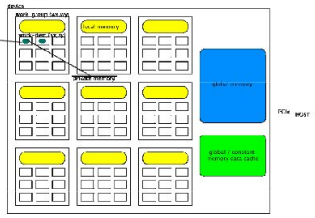
- A kernel is executed by the work-items

```
// OpenCL Kernel Function for element by element vector addition
  kernel void VectorAdd(  global const float8* a,   global const float8* b,   global float8* c)
{
    // get oct-float index into global data array
    int iGID = get_global_id(0);

    // read inputs into registers
    float8 f8InA = a[iGID];
    float8 f8InB = b[iGID];
    float8 f8Out = (float8)0.0f;

    // add the vector elements
    f8Out.s0 = f8InA.s0 + f8InB.s0;
    f8Out.s1 = f8InA.s1 + f8InB.s1;
    f8Out.s2 = f8InA.s2 + f8InB.s2;
    f8Out.s3 = f8InA.s3 + f8InB.s3;
    f8Out.s4 = f8InA.s4 + f8InB.s4;
    f8Out.s5 = f8InA.s5 + f8InB.s5;
    f8Out.s6 = f8InA.s6 + f8InB.s6;
    f8Out.s7 = f8InA.s7 + f8InB.s7;

    // write back out to GMEM
    c[get_global_id(0)] = f8Out;
}
```



How Portable is an OpenCL application?

- OpenCL provides a standard syntax to program GPUs (and CPUs)
- Stream programming is a mix of parallel programming and hardware resources management
  - GPUs are not time-shared devices (contrarily to CPUs)
  - Memory hierarchy is exposed
- When moving to a different architecture, hardware resource constraints may make OpenCL programs: non efficient in some cases, incorrect some other times
- Optimized OpenCL codes tend to map very accurately to a given set of resources

Can the OpenCL Compiler Help?

- Very unlikely
- Work-groups are declared in data structures

```
// Total # of work items
size_t szGlobalWorkSize[1];
// # of work items in the work group
size_t szLocalWorkSize[1];
```

- The kernel code itself is text and thread id dependent, the JIT cannot change the work-group configuration

```
char* cDotProduct =
ocutLoadProgramSource
    (cPathAndName, "// My comment\n",
     &szKernelLength);
```


OpenCL Portability and Performance

- OCL kernels to be tuned for each device
  - Express data parallelism
  - Consider device-specific information
    - Number of work-items in a work-group
    - Local memory
  - Dramatic impact on performance
    - Might even be under performing
- OpenCL to support a large variety of devices
  - Abstract the specifics of hardware
  - Is OpenCL a good candidate for high level compilers?

HMPP
A directive-based compiler

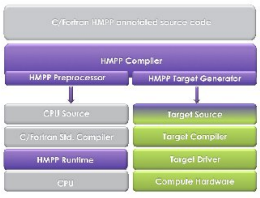
HMPP Objectives

To give developers a high level abstraction for manycore programming

- Incrementally program GPU-accelerated applications
- Rapidly build hybrid applications
- Keep accelerated kernels hardware independent
- Ensure application portability and interoperability

Overview

- C and Fortran GPU programming directives
  - Define and execute GPU-accelerated versions of code
  - Optimize CPU-GPU data movements
  - Complementary to OpenMP and MPI
- A source-to-source hybrid compiler
  - Generate powerful accelerated kernels
  - Works with standard compilers and target tools
  - Tuning directives to optimize accelerated kernels
- A runtime library
  - Dispatch computations on available GPUs
  - Scale to multi-GPUs systems

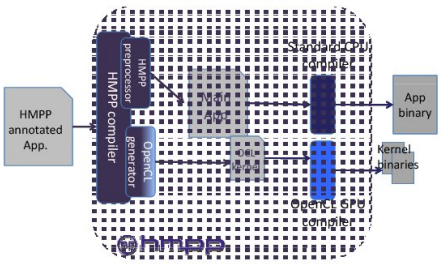


History and Status

- HMPP 1.x: High level abstraction of GPU programming
  - Focuses on efficiently offloading computations in remote accelerators
  - Introduction of CUDA back-end generator
- HMPP 2.x: Achieving a set of directives that fully exploit GPU capabilities
  - Programming directives
    - Group of codelets and data mapping
    - Partial transfers, resident data
    - Regions of code
  - Tuning directives
    - Exploit device memories
    - Complex, reduction
    - Parallelizing directives
  - New CAL/IL and OpenCL back-ends to be released
  - Windows version

HMPP Targeting OpenCL

- Naturally integrate in HMPP workflow



HMPP & OpenCL

- Higher level interface
- Native Fortran support
- Retain full compatibility with all OpenCL-supporting devices
- Hardware-specific optimizations in directives rather than source code and data structures
- Compatible with more specific back-ends (CUDA, CAL/IL)

HMPP: Tuning Directives

- To add code properties
  - Force loop parallelization
  - Indicate parameter aliasing
- To apply code transformations
  - Loop unrolling and jam, blocking, tiling, permute, …
- To control mapping of computations
  - Gridification
  - Place variables in shared or constant memory
  - Threads synchronization barriers

Conclusion

Conclusion

- OpenCL
  - Low level programming interface
  - Portable across various devices
  - Not simple but much simpler than OpenGL graphics programming
  - Can be compared as x86 assembly for manycore programming
  - Target expert developers
  - Suitable for higher level programming languages and tools
- HMPP
  - OpenMP-like directives for programming and tuning GPU-accelerated applications
  - Offer incremental levels of programming from minimal to advanced and expert
  - A source-to-source C and Fortran compiler targeting OpenCL

CAPS

Innovative Software for Manycore Paradigms

2.3 Laurent Bertaux (CAPS Entreprise)

Retours d'expériences de l'appel à projets CAPS/GENCI 2009

Afin de pouvoir développer cet écosystème au niveau national et accompagner l'installation de moyens hybrides graphiques installés récemment au CCRT et au CINES, GENCI et la société CAPS Entreprise lancent un second appel à projet autour du portage avec des outils de programmation de haut niveau d'applications scientifiques sur ces architectures. Les résultats du premier appel à projet, qui a permis le portage de trois applications dans des domaines aussi divers que la physique atomique, la mécanique des fluides et l'Imagerie 3D, vous seront présentés.

GENCI / CAPS 2009 Call for Project

Aristote Presentation – 03/25/2010

Laurent Bertaux, CAPS entreprise

GENCI Call for Project Presentation

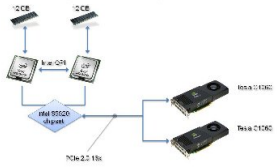
- Objective
  - To promote hybrid programming in France
  - To create a user community for the GENCI hybrids computers
    - CINES, CCRT
- Board
  - Stéphane Requena (GENCI)
  - Christophe Calvin (CEA)
  - Jean-Christophe Penalva (CINES)
  - Guillaume Colin-de-Verdiere (CEA)
  - Pierre-Francois Lavallee (IDRIS)
  - Henri Calandra (TOTAL)

GENCI Call for Project Presentation

- Prop
  - Atomic physic from 2DRMP
  - LIP6 : JEZEQUEL Fabienne
- Gravi3D
  - Gravimetric
  - CNRS MIGP / Université de Pau : MARTIN Roland
- SPH-Flow
  - Smoothed Particles Hydrodynamics
  - Hydrocéan : JACQUIN Erwan
- 2010 : second call for project
  - More projects should be selected
  - Submission deadline : 31 mars 2010

CAPS Compute Lab

- CAPS Compute Lab is composed of:
  - 1 login node
  - 3 storage nodes over Lustre (4 To)
  - 20 compute nodes (42 Teraflops)
- Each compute node is made up of:
  - A dual-socket Intel Nehalem (bi-processor) machine
  - Each Nehalem processor is quad-core (4 CPU cores)
  - 2 Nvidia Tesla C1060 GPUs
  - 24 GB of memory



Feedbacks / REX

- Use highly optimized libraries (MKL, …)
- Pointer-based code needed a lot of work : GPUs only work well on arrays
- Kernels can be optimized with HMPP using only directives but very long and complicated kernels are sub-optimal on GPUs
- Double-precision performance is not good enough on current Nvidia GPUs to show significant improvements but single-precision performance is !
- Hybrid version (using both CPU and GPU) is the best way but needs more efforts
- Code-heavy real-world application can be accelerated

Physique Atomique : PROP

Jean-Charles Vasnier, CAPS entreprise

Credits

- CAPS Team
- Fabienne Jézéquel, LIP6

Agenda

- Introduction
- Implementation focus
  - Implementation changes
- Porting with HMPP
  - Matrix multiplication
- Conclusion & Future work

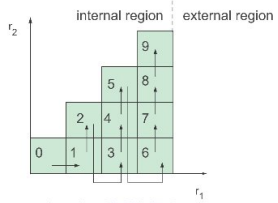
Introduction

The 2DRMP Suite
2-Dimensional R-Matrix Propagator

- Models electron collisions with H-like atoms and ions
- Is included in the CPC (Computer Physics Communications) International Program Library which contains over 2300 programs in computational physics and physical chemistry
  - http://www.cpc.cs.qub.ac.uk
- Was awarded the HPC Prize for Machine Utilization by the UK Research Councils' High End Computing Strategy Committee in 2006
- Consists of seven programs (including PROP) and approximately 20,000 lines of code

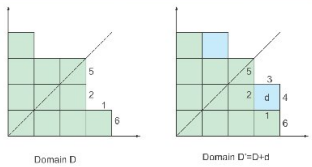
The PROP Program
R-matrix propagation

- The two-electron configuration space (r1 ,r2) is divided into sectors.



- The key computation in 2DRMP is the propagation of the global R-matrix, R, from sector to sector.

Global R-matrix From D to D'



- I are input edges : 1, 2
- O are output edges : 3, 4
- X are common edges : 5, 6

$$\Re^I = \begin{pmatrix} \Re^I_{II} & \Re^I_{IX} \\ \Re^I_{XI} & \Re^I_{XX} \end{pmatrix} \Rightarrow \Re^O = \begin{pmatrix} \Re^O_{OO} & \Re^O_{OX} \\ \Re^O_{XO} & \Re^O_{XX} \end{pmatrix}$$

Propagation Equations

- The propagation across each sector is characterized by:

$$\mathcal{R}^O_{OO} = r_{OO} - r_{OI}(r_{II} + \mathcal{R}^I_{II})^{-1} r_{IO}$$
$$\mathcal{R}^O_{OX} = r_{OI}(r_{II} + \mathcal{R}^I_{II})^{-1} \mathcal{R}^I_{IX}$$
$$\mathcal{R}^O_{XO} = \mathcal{R}^I_{XI}(r_{II} + \mathcal{R}^I_{II})^{-1} r_{IO}$$
$$\mathcal{R}^O_{XX} = \mathcal{R}^I_{XX} - \mathcal{R}^I_{XI}(r_{II} + \mathcal{R}^I_{II})^{-1} \mathcal{R}^I_{IX}$$

- r_{II} , r_{IO} , r_{OI} , r_{OO} : local R-matrices associated to subregion d
- The matrix inversions are not explicitly performed. For instance, to compute the inversion, the linear system (r_{II} + R_I)X = r_{IO} is solved.
- The most relevant matrix operations are performed in
  - the dgetrf and dgetrs LAPACK routines
  - the dgemm BLAS routine.

Dynamically Changing Matrices

- Propagation equations are dominated by matrix multiplications ...

… which are not straightforward!

- As the propagation proceeds, R dynamically increases in size and changes the designation of its rows and columns.
- For a configuration involving 210 sectors,
- A typical size for the local R-matrices r is 200x200
- The global R-matrix will start off at 200x200 and will eventually reach 5,000x5,000.
- The propagation will need to be performed for hundreds of collision energies which are independent.
- The focus of the GPU work will be to parallelize the propagation associated to one energy.

Implementation Focus

Application Description

- Code source of PROP :
  - 10 000 lines in Fortran 95
  - MPI version already exists (no transfers)
  - Only double precision version, LIP6 still validating a simple precision version
- Datasets :

Dataset	Size of sectors	Number of sectors	Number of energy
DATA_SMALL	90*90	10	6
DATA_MEDIUM	90*90	10	64
DATA_LARGE	383*383	210	6
DATA_HUGE	383*383	210	64

Application Overview

- Application delivered with a sequential version of the lapack library (self-content)
- MPI version enables multiple energy computations in parallel (1 energy per process)
- First task, linked with MKL to take advantage of multicore processors

Profiling Results

- With DATA_LARGE dataset
- 3 nodes with 2 processes per node
- 1 process → 4 MKL threads on 1 socket
- Principal functions are :
  - Matrix copy function : matrix_copy (49%)
  - Matrix multiplication : dgemm (34 %)
  - IO function : create_amplitude_arrays (11%)

Application Algorithm

- For each sector, computation of the 4 equations

$$\mathcal{R}^o_{oo} = r_{oo} - r_{oI}(r_{II} + \mathcal{R}^I_{II})^{-1} r_{Io}$$
$$\mathcal{R}^o_{ox} - r_{oI}(r_{II} + \mathcal{R}^I_{II})^{-1}\mathcal{R}^I_{Ix}$$
$$\mathcal{R}^o_{xo} - \mathcal{R}^I_{xI}(r_{II} + \mathcal{R}^I_{II})^{-1} r_{Io}$$
$$\mathcal{R}^o_{xx} = \mathcal{R}^I_{xx} - \mathcal{R}^I_{xI}(r_{II} + \mathcal{R}^I_{II})^{-1}\mathcal{R}^I_{Ix}$$

- Using GPU becomes interesting with large matrices
  - Large computation throughput
  - But transfers are the bottleneck
- With help from Jocelyne Erhel (INRIA), change to a matrix formulation

$$\mathcal{R}^G - \begin{pmatrix} \mathcal{R}^o_{oo} & \mathcal{R}^o_{ox} \\ \mathcal{R}^o_{xo} & \mathcal{R}^o_{xx} \end{pmatrix} - \begin{pmatrix} r_{oo} & 0 \\ 0 & \mathcal{R}^I_{xx} \end{pmatrix} + \begin{pmatrix} -r_{oI} \\ \mathcal{R}^I_{xI} \end{pmatrix} (r_{II} + \mathcal{R}^I_{II})^{-1} (r_{Io} \quad -\mathcal{R}^I_{Ix})$$

- Allow to increase matrix sizes

Result of the Reimplementation

- Execution profile has changed :
  - Calls to matrix copy : 22 → 5
  - Calls to solver : 2 → 1
  - Calls to dgemm : 4 → 1
- Larger sizes for the dgemm
- Performance consequences
  - With DATA_LARGE on 3 nodes and 2 processes per node

Configuration	Time	Speedup (/Original + MKL)
Original + MKL (4 cores)	10m53s	1,00
Reimplemented + MKL (4 cores)	8m17s	1,31

Porting with HMPP

Porting DGEMM

- Introduction of Fortran module that contains all the HMPP programming directives
- As the size of DGEMM dynamically changes, allocate matrices with size larger than required
  - Make use of HMPP partial transfer
- To have the best performance
  - Call to CUBLAS DGEMM for the GPU version
  - Call to MKL DGEMM for the native version

DGEMM Sizes

- DGEMM sizes with execution time

M	N	K	% dgemm time
383	383	5997	34
7660	7660	766	14
7277	7277	766	11,5
6894	6894	766	9,2
6511	6511	766	7,1
6128	6128	766	6,2
5745	5745	766	4,9
5362	5362	766	3,5
...			

- Most calls with size
  - 383 x 383 x 5997
  - 34% of the overall DGEMM time
- Second more frequent size
  - 7660 x 7660 x 766
  - 14% of the DGEMM time

...

CPU/GPU Execution Decision

M	N	K	% dgemm time	4-core MKL CPU Time (ms)	GPU time* (ms)
383	383	5997	34	**44**	51
7660	7660	766	14	2150	**2030**
7277	7277	766	11,5	1940	**1830**
6894	6894	766	9,2	1730	**1630**
6511	6511	766	7,1	1550	**1390**
6128	6128	766	6,2	1360	**1230**
5745	5745	766	4,9	1200	**1130**
5362	5362	766	3,5	1050	**990**

*GPU time includes data transfers

CPU/GPU Execution Decision (2)

- DGEMM with large m and n matrices executed on GPU
- Others executed on the CPU
  - Limit data transfer overhead
- Experiments show size limit of DGEMM offloaded onto GPU
  - M > 1300
  - N > 1300
  - K > 700

Final Performance

- Performance of the modified algorithm

Configuration	Time	Speedup (/Original + MKL)	Speedup (/Original + MKL)
Original + MKL (4 cores per process)	10m53s	1,00	-
Reorganized + MKL (4 cores per process)	8m17s	1,31	1,00
Reorganized + Cublas/MKL (4 cores per process)	7m15s	1,50	1,15

- With DATA_LARGE on 3 nodes and 2 processes per node (4 CPU cores and 1 GPU per process)

Conclusion & Future Work

Conclusion & Future Work

- Hybrid application (CPU + GPU)
- Limited speed-up
  - Modification of the algorithm and integration of the MKL → bigger DGEMM but fewer
  - Use of DGEMM with different data sizes not adapted to GPU
- Possible GPU performance improvement
  - Review the algorithm
  - Make resident the whole R-matrix on GPU

CAPS

3D Finite-Volume Modeling for Gravity Anomaly Calculation

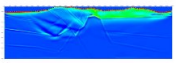
Georges-Emmanuel Moulard, CAPS entreprise
Roland Martin, Laboratoire de Modélisation et Imagerie en Géosciences de Pau/CNRS UMR 5212/Magique3D INRIA

Introduction

Magique 3D (3D Advanced Numerical Modelling in geophysics)
Overview of our Research Activities (Project Leader H. Barucq)

- 1. **Geophysical seismology: to draw hasard maps quickly** by computing possible aftershocks of an earthquake (**Univ. Princeton/USA, Univ. Toulouse, Barcelona Supercomputing Center**), global seismology (whole Earth). D. Komatitsch et al. **SPECFEM3D** software package, freely available from www.geodynamics.org
- 2. **Inverse problems : to improve the knowledge of the subsurface**

  Seismic imaging: to produce images of the subsurface by applying the **Reverse Time Migration** (imaging the interfaces), adjoint inversion for full-wave form inversion using **SPECFEM or** discontinuous Galerkin method **(IPDGFem) . Collaborations: TOTAL**, ANR project **NUMASIS**




- **Boundary conditions** (reduced size of the computational box) : ABCs and PMLs. **SEISMIC-CPML** (finite difference method): optimized CPMLs; open source software (MPI+OpenMP +CUDA). Used by **TOTAL** and **BRGM.** Martin, Komatitsch, Ezziani, Gedney, Michea
- **Discontinuous finite elements: IPDG method** (tetrahedra, Hp adaptativity in space, high-order approximation). Diaz, Agut, Barucq
- **Local time stepping** (Control of dispersion effects, Adaptativity in time)
- **HPC:** a carefully optimized HPC code greatly facilitates the resolution of inverse problems, which requires solving thousands of direct problems. Hundreds of GPUs (Komatitsch, Michea, Erlebacher, Tromp, Martin). Univ. Princeton, Univ. Talahasee/USA.

CAPS 03/25/2010 – Aristote Presentation 31

Inversion Using Potential Methods

- Joint Inversion problems : seismic/gravimetry (magnetometry) imaging, acoustic/electrical imaging. ANR AHPI.
- Capacitance tomography using Finite Volumes and MPI (Ortiz-Aleman, Martin, Roffé)

  Laplace equation : $\nabla.(M\nabla\phi) = S$

  (isotropic/anisotropic) M tensor : electrical capacitance/impedance tomography multiphase flow pressure-variable viscous stress tensor in CFD. Discontinuities of the M tensor.
- Gravimetry or magnetometry for regional imaging : salt dome regions and Yucatan/Mexico region (Chicxulub crater). Martin, Ortiz-Aleman, Fukugauchi

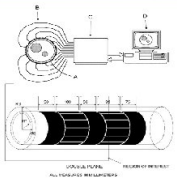
  Poisson equation : $\nabla^2\phi = S$

  Geophysical imaging using Gravimetry, (aero)magnetism, or electrical resistivity coupled to seismic imaging, pressure-constant viscosity (Stokes, incompressible flow) equations in CFD

Ortiz-Aleman, Martin and Gamio-Roffé (2003, 04, 05, 09)

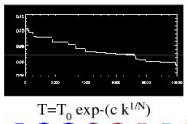
CAPS 03/25/2010 – Aristote Presentation 32

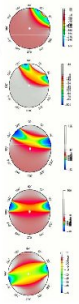
Electrical Capacitance Tomography



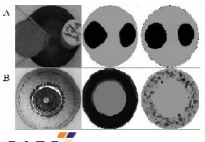
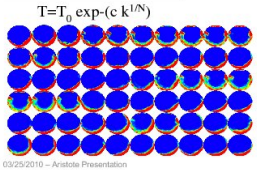
Simulated annealing/Gradient technique:

- Sensitivity Matrix Sijk computed by cross correlation of several solutions ϕi and ϕj for a given set of parameters εk





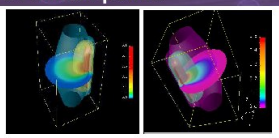
$T=T_0 \exp(-c\, k^{1/N})$

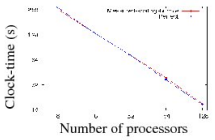
CAPS 03/25/2010 – Aristote Presentation

Scaling of 3D ECT Modeling on 128 processors of Jade Xeon cluster/CINES, 64 million points

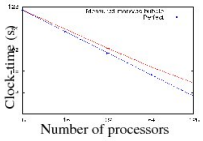
Error of 2-3% in 3D, 7 to 10% in 2D (20% adjacent electrodes)



Strong scaling with MPI



Matrix vector scaling



- High number of cache misses (low Number of processors) and too low Instructions per cycle (high number of processors). Efficient parallelism. Needs more calculations in each subdomain.

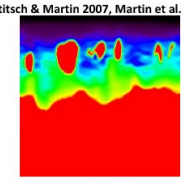
GPU implementation is under its way in CUDA and HMPP. (Martin, Komatitsch, Erlebacher,CAPS)

CAPS 03/25/2010 – Aristote Presentation 34

Application to Gravimetry and Magnetism using GPUs.Yucatan plate/Mexico and salty domes regional modelling. Institute of Geophysics, UNAM, Mexico City

Seismic modeling using CPML boundary conditions

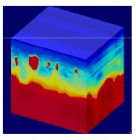
(Komatitsch & Martin 2007, Martin et al. 2008a,b, 2009,2010)



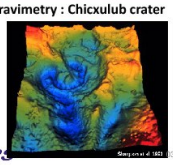
Loi de Gardner : correlation of density and velocities from seismic data

$\rho = aV^b$

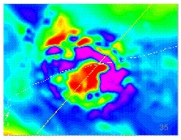
Salt domes



Gravimetry : Chicxulub crater

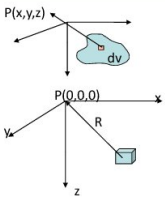


New aeromagnetic survey (2004) Ortiz-Aleman & Fukugauchi (2009)



CAPS 03/25/2010 – Aristote Presentation 35

Gravimetry Solvers: GBOX（R.J.Blakely,1995)/ FVM



$$g(x,y,z)=G\iiint_V \rho(\xi,\eta,\zeta)\frac{z-\zeta}{\left[(x-\xi)^2+(y-\eta)^2+(z-\zeta)^2\right]^{3/2}}d\xi\,d\eta\,d\zeta$$

$$g = \gamma\rho\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2}\mu_{ijk}\left[z_k \operatorname{tg}^{-1}\frac{x_i y_j}{z_k R_{ijk}} - x_i \log(R_{ijk}+y_j) - y_j \log(R_{ijk}+x_i)\right]$$

$$R_{ijk}=\sqrt{x_i^2+y_j^2+z_k^2}, \qquad \mu_{ijk}=(-1)^i(-1)^j(-1)^k$$

GBOX:Algorithm is Θ(N⁵)

$$\sum_{e=1}^{6}\iint_{S_e} Grad(\phi).ndS = -4\pi G\rho Vol_e + \iint_{S_e\in S} gdS$$

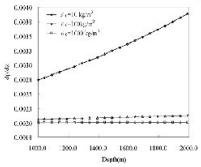
K Φ = F **FVM: Algorithm is ϴ(N⁴)**

Matrix **K** is hepta-diagonal in 3D at second order (13 diagonals at 4th order)

CAPS 03/25/2010 – Aristote Presentation 36

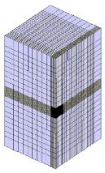
Accuracy Verification

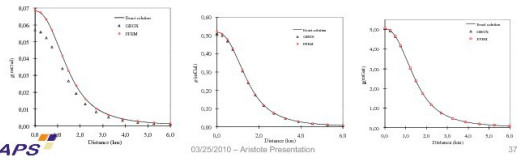
Density model for verifying



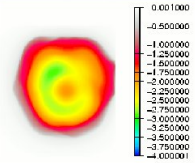
$$\rho(z) = \frac{\rho_0^3}{(\rho_0 - c\ z)^2}$$

(c= 0.001 kg/m4)

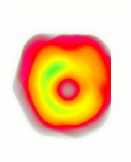




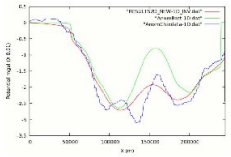
Comparison between FVM and GBOX



FVM: used cpu time : 592 s 290*290*750 points : 126 Million points. Model same as in GBOX in subdomain (250*250*250)



GBOX: used cpu time : 1057 s on 250 Xeon Jade cores. 250x250x250 points model (1point/km)



Over estimation of GBOX potential solution

Conclusion

- Application of 3D Parallel /GPU Finite Volumes (previously used in capacitance tomography, OAleman and Martin 2004, 2005) to the gravity and magnetics forward modeling
- The FVM software is more accurate and much faster than the classical integration method GBOX, particularly if density in the material body is highly heterogeneous.
- The computational efficiency for the FVM method is more pronounced in regions with greater heterogeneities. 1GPU for FVM and 250 processes for GBOX.

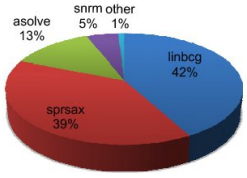
Near future :

- Needs cache misses reduction.
- Needs better matrix preconditionning to reduce number of iterations.
- Multi-GPU: optimization of data transfers between CPUs and between Host and GPU.
- More access memories per cycle to be performed.
- High performance inversion. Joint modeling of 3D gravity, magnetics and seismic data using HPC multi-GPU. Blocking, non-blocking strategies (Martin et al. , Lecture Notes Comp. Science 2008)
- Hybrid programming on GPU under its way.

GRAVI3D Acceleration with HMPP

Application Analysis

- Code overview
  - Fortran language
  - About 1800 lines
  - Double precision
  - No MPI or OpenMP parallelization
- Profile
  - Compiled with ifort 11.1 (flag –O3)
  - Profiled with gprof



Performance Results

- Initial CPU optimization
  - Suppress redundant data
- GPU-only version using HMPP
  - linbcg, sprsax, asolve and snrm have been ported

	Reference CPU version (1 core)	Optimized CPU version (1 core)	GPU version
Execution time (sec)	2370	1371	266

Domain size: 400*400*300

Speed-up

	Reference CPU version (1 core)	Optimized CPU version (1 core)	GPU version
Reference CPU version (1 core)	1	1,73	8,91
Optimized CPU version (1 core)	-	1	5,15
GPU version	-	-	1

CPU-GPU Hybrid Version

- Hybrid version concurrently using 2 CPU cores and 1 GPU
  - 3 OpenMP threads:
    - 2 threads to compute on CPU the top and the bottom of the domain
    - 1 thread to compute on GPU the rest of the domain
- No speed-up compared to the GPU-only version
  - Add extra data transfers
  - CPU memory bandwidth saturation?

Summary

- x5.15 speed-up between the optimized CPU version and the GPU-only version
- x6 speed-up for the accelerated parts
- Hybrid version is not convincing

Implementation Details

Code Porting and Data Transfers Optimizations

- Use of HMPP to port linbcg, sprsax, snrm and asolve on GPU
  - Modification of 5% of the source code
- Data movement optimizations
  - Avoid useless data transfers
    - Use HMPP directives to load only once constant data
    - All the arrays stay resident on the GPU during the computation
- Only 6 scalars are transferred between the host and the GPU per iteration

Sprsax Codelet Optimizations

- Sprsax is bandwidth bound
  - 1 point calculation requires 14 GPU memory accesses and 13 floating operations
  - Some read accesses could be shared among the GPU threads
- Reduce memory accesses from 14 to 6
  - Apply software pipelining
  - Exploit CUDA shared memory with HMPP tuning directives
  - Suppress redundant data accesses
- 1.8 speed-up

	Before optimizations	After optimizations
1 kernel call (ms)	71000	39000

CAPS

SPH-flow

Lionel Guivarch, Guillaume Oger, HydrOcean
Pierre-Louis Cabelguen, CAPS entreprise

HydrOcean, SPH-flow Application

HydrOcean: Presentation

- Spin-off from École Centrale Nantes/CNRS (Fluid Mechanics Laboratory)
- Specialized in hydrodynamic simulation (study/software)
- Co-owner of innovative numerical simulation software (based on 9 years of research, 7 thesis and several industrial applications)

HydrOcean

SPH-flow: Example of Needs in Maritime Industry

- Industry needs
  - Anticipating and avoiding damages to structures (ship, platform, airplane...)


- Benefits
  - Human and environmental safety (capsizing), operating losses in M€/day (damage) ...
- Tools
  - The innovative software SPH-flow can model these complex phenomena by replacing the tests (expensive and difficult to scale)

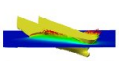
SPH-flow: innovations

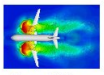
- SPH-flow is a meshless code with short set-up time
- SPH-flow code can model flows which are difficult to simulate with conventional methods:
  - Impact, high dynamic, non diffusive interfaces, complex geometries, fluid/structure interaction...
- SPH-flow is used in different sectors:
  - Marine (DCNS), Oil&gaz (Saipem, GTT), Automotive (Michelin)...

SPH-flow: Applications

- http://www.sph-flow.com/application.html


Hydroplaning


Impact of a ship bulb


Plane ditching

SPH-flow: GPU Benefits

- SPH-flow seems particularly well-fitted to GPU acceleration
  - Particle method: interpolations achieved through convolutions on local particle neighborhoods (loops with disordered sums)

Typical industrial 3D applications	
Particle numbers	1 to 10 millions (with 80 neighbor particles/particle)
Time steps number	~ 50,000 (for simulations of phenomena with high dynamics and small physical durations ~ 0.5 second)
CPU times	6 days on 32 cores (CPU cost : 2,5.10-4 seconds/particle/time step/core)

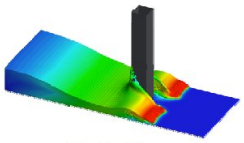
- New massive 3D industrial applications can be considered

SPH-flow: source code

- Features
  - Free-form Fortran 90
  - 22,000 lines
  - Existing MPI-based parallelization (domain decomposition)
  - Initially implemented in double precision format and recently translated into single precision by HydrOcean
- Current constraints
  - SPH-flow does not allow single-process runs
  - Each MPI process should own at least 3,000 particles
  - Presence of load-balancing procedures between processes

Test Case

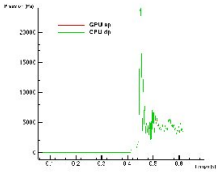
- Dam-break:
  - Reference test case
  - Impact of a wave on a vertical beam
  - Representative case of SPH-dedicated applications



- 3 configurations proposed by HydrOcean
  - Coarse resolution: 47,150 particles
  - Medium resolution: 155,550 particles
  - Fine resolution: 2,474,010 particles

GPU Porting Validation

- Comparison of CPU/GPU outputs
  - Pressure sensors (8 sensors placed on the vertical beam)



- Validated when CPU and GPU curves match

Code Profiling

Code Profiling

- Dam-break, single precision,155,550 particules, 8 monothread MPI processes
  - Exactriemannsolver_tait: 25.13%
  - Calculinteractionfehlberg: 23.97%
  - Calcmatl_gradmuscl: 16.53%
  - Linearmuscl: 9.23%
- Similar profiles for other test configurations
  - Wedge Single Precision and Double Precision
  - Dam-break Double Precision
- HMPP port focuses on the 2 main subroutines
  - Represent 49.1% of total execution time

HMPP Port

Source Code: General Structure

- Temporal loop

```
do while ( t < tmax )
  ! init
  ...
  if ( neighbour_search )
    call RungeKutta
  else
    call RungeKutta_fast
  end if
end do ! while ( t < tmax )
```

- Both Runge-Kutta subroutines have the same structure
  - 4 calls to the interaction computation subroutine (*calculinteractionfehlberg*)
- Interactions are computed for 3 kinds of particles

Basic Kernel Identification

```
! Local-Local interactions
  do i=1,InIn_Particles_Count
    do j=1,Int_Particles(i)
      call calculinteractionfehlberg(i,j)
    end do
  end do

! Local-Tank interactions
  do i=InIn_Particles_Count+1,InTankGhost_Particles_Count
    do j=1,Int_Particles(i)
      call calculinteractionfehlberg(i,j)
    end do
  end do

! Local-Body interactions
  do i=InTankGhost_Particles_Count+1,InBodyGhost_Particles_Count
    do j=1,Int_Particles(i)
      call calculinteractionfehlberg(i,j)
    end do
  end do
```

- Outer loops over ***i*** are parallel
- Inner loops over ***j*** are sequential and perform result accumulation

Approach

- Objective: to compute the 3 loops on GPU
  - Some data are shared between the loops
  - No intermediate data transfers required
- How?
  1. HMPP « Codeletization » of one loop to confirm the approach
  2. Exploit HMPP groups of codelets to offload the 3 loops on GPU
  3. Optimize GPU allocation and data transfers

HMPP Codeletization

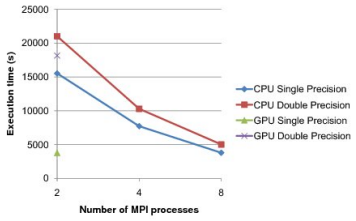
- « Codeletization » allows to reduce the global execution time on CPU
- Dam-break test case
  - 7,128 particles
  - 2 MPI processes
  - 1 thread per process
  - The 3 loops are executed on CPU

Source code version	Application execution time
Original, Single Precision	10.15 minutes
Modified, Single Precision	9.16 minutes
Original, Double Precision	12.28 minutes
Modified, Double Precision	11.29 minutes

Performance Figures

Overview

- Dam-break test case, medium resolution (155,550 particles)



- 1 thread per MPI process

Performance Details: Single Precision

- Dam-break, medium resolution (155,550 particles)
  - 2 MPI processes
  - 1 thread per process
  - 1 process = 1 CPU core **or** 1 CPU core + 1 GPU

	Allocation release GPU	Transfers	Callsite	Index	Total	Kernel Speed-up	Speed-up with I/O
CPU	0	0	15,480.99	57.43	15,538.43	-	-
GPU	329.46	488.25	2,895.98	59.63	3,773.32	×5.35	×4.12

- Application speed-up: x1.86
  - CPU version: 422.30 minutes
  - GPU version: 227.15 minutes

Performance Details: Double Precision

- Dam-break, medium resolution (155,550 particles)
  - 2 MPI processes
  - 1 thread per process
  - 1 process = 1 CPU core **or** 1 CPU core + 1 GPU

	Allocation release GPU	Transfers	Callsite	Index	Total	Kernel Speed-up	Speed-up with I/O
CPU	0	0	20,965.56	56.97	21,022.53	-	-
GPU	377.25	867.26	16,849.35	59.67	18,153.54	×1.24	×1.16

- Application speed-up: ×1.10
  - CPU version: 548.00 minutes
  - GPU version: 499.37 minutes

Conclusion

Conclusion

- The primary work on the code shows a good potential of acceleration in a hybrid context
- GPU performance figures are promising
  - Single precision, GPU vs. CPU = x1.86
  - Double precision, limited improvements, GPU vs. CPU = x1.10
- Thanks to the source code refactoring, the application is ready for manycore architectures

Future Work

- Extend the set of tests
  - Increase the number of particles
  - Change the MPI configuration
    - Number of processes
    - Process repartition
    - …
  - Study scaling and use other simulation configurations
- Optimizations
  - Reduce the number of GPU allocation/release sequences
  - Implement an optimized version of the kernel (using CUDA shared memory)
  - Implement an improved MPI load-balancing version
  - Compute all interactions on GPU (provided that MPI communications can be modified)

Q & A

Thank you for your attention

CAPS

Innovative Software for Manycore Paradigms

2.4 Dimitri Komatitsch (Université de Pau/INRIA)

Portage d'une application de propagation d'ondes sismisques en multi-GPU

Nous illustrerons le fait que l'augmentation spectaculaire des performances des ordinateurs au cours des dernières années et en particulier l'apparition du calcul GPU permet l'étude de la propagation des ondes sismiques résultant de tremblements de terre à haute résolution ainsi que la modélisation de l'aléa sismique.

Portage d'une application de propagation d'ondes sismisques en multi-GPUs

Dimitri Komatitsch and David Michéa
(Univ Pau, CNRS and INRIA Sud-Ouest Magique3D)

Gordon Erlebacher (Department of Scientific Computing, Florida State University, USA)

Dominik Göddeke (TU Dortmund, Germany)

OpenGPU, École Polytechnique
March 25, 2010

Brief history of numerical methods

Seismic wave equation : tremendous increase of computational power
⇒ development of numerical methods for accurate calculation of synthetic seismograms in complex 3D geological models has been a continuous effort in last 30 years.

Finite-difference methods : Yee 1966, Chorin 1968, Alterman and Karal 1968, Madariaga 1976, Virieux 1986, Moczo et al, Olsen et al..., difficult for boundary conditions, surface waves, topography, full Earth

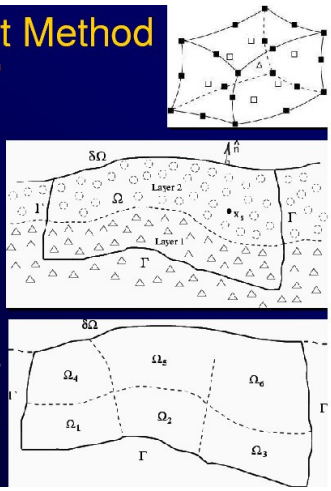
Boundary-element or boundary-integral methods (Kawase 1988, Sanchez-Sesma et al. 1991) : homogeneous layers, expensive in 3D

Spectral and pseudo-spectral methods (Carcione 1990) : smooth media, difficult for boundary conditions, difficult on parallel computers

Classical finite-element methods (Lysmer and Drake 1972, Marfurt 1984, Bielak et al 1998) : linear systems, large amount of numerical dispersion

Spectral-Element Method

- Developed in Computational Fluid Dynamics (Patera 1984)
- Accuracy of a pseudospectral method, flexibility of a finite-element method
- Extended by Komatitsch and Tromp, Capdeville et al.
- Large curved "spectral" finite-elements with high-degree polynomial interpolation
- Mesh honors the main discontinuities (velocity, density) and topography
- Very efficient on parallel computers, no linear system to invert (diagonal mass matrix)



Equations of motion (solid)

Differential or *strong* form (e.g., finite differences):

$$\rho \partial_t^2 \mathbf{s} = \nabla \cdot \mathbf{T} + \mathbf{f}$$

We solve the integral or *weak* form:

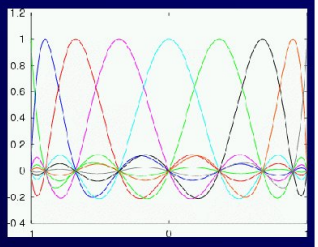
$$\int \rho \mathbf{w} \cdot \partial_t^2 \mathbf{s} \, d^3\mathbf{r} = -\int \nabla \mathbf{w} : \mathbf{T} \, d^3\mathbf{r}$$

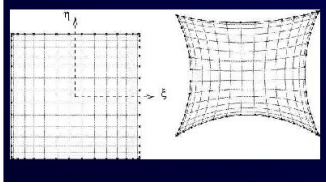
$$+ \mathbf{M} : \nabla \mathbf{w}(\mathbf{r}_s) S(t) - \int_{F-S} \mathbf{w} \cdot \mathbf{T} \cdot \hat{\mathbf{n}} \, d^2\mathbf{r}$$

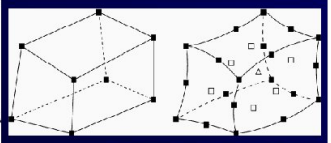
+ attenuation (memory variables) and ocean load

Finite elements

- High-degree pseudospectral finite elements
- N = 5 to 8 usually
- *Exactly* Diagonal mass matrix
- No linear system to invert





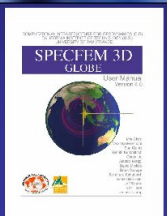


Global Simulations: SPECFEM3D_GLOBE

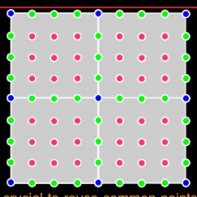
Open source: geodynamics.org
On-demand TeraGrid applications:
- Automated, near real-time simulations of all M>6 earthquakes
- Analysis of past events (more than 20,000 events)
- Seismology Web Portal (geodynamics.org)

Petascale simulations:
- Global simulations at 1-2 Hz
- Reached 1.15 s period on 149,784 cores at ORNL
- Moving towards global 'adjoint tomography'

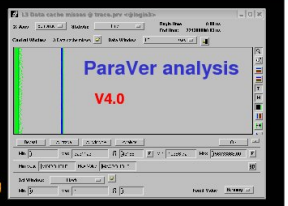
SPECFEM3D_GLOBE Users Map

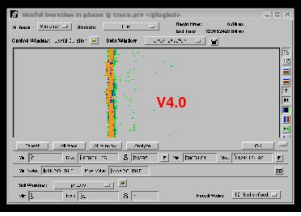


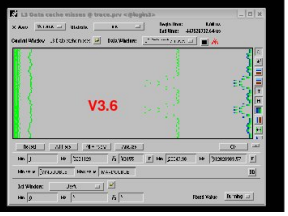
Results for load balancing: cache misses (J. Labarta, BSC)



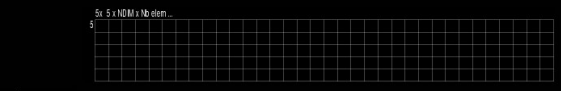
=> it is crucial to reuse common points by keeping them in the cache







BLAS 3 (Basic Linear Algebra Subroutines)



⇒ No need to, and cannot easily use BLAS

⇒ Compiler already does an excellent job for small static loops

⇒ Cannot use cuBLAS on GPUs

Can we use highly optimized BLAS matrix/matrix products (90% of computations)?

- For one element: matrices (5x25, 25x5, 5 x matrices of (5x5)), BLAS is not efficient: overhead is too expensive for matrices smaller than 20 to 30 square.
- If we build big matrices by appending several elements, we have to build 3 matrices, each having a main direction (x,y,z), which causes a lot of cache misses due to the global access because the elements are taken in different orders, thus destroying spatial locality.
- Since all arrays are static, the compiler already produces a very well optimized code.

Minimize CPU ↔ GPU data transfers

- CPU ↔ GPU memory bandwidth much lower than GPU memory bandwidth
  - Use page-locked host memory (`cudaMallocHost()`) for maximum CPU ↔ GPU bandwidth
- Minimize CPU ↔ GPU data transfers by moving more code from CPU to GPU
  - Even if that means running kernels with low parallelism computations
  - Intermediate data structures can be allocated, operated on, and deallocated without ever copying them to CPU memory
- Group data transfers
  - One large transfer much better than many small ones
- Fit all the arrays on the GPU card to avoid costly CPU ↔ GPU data transfers
- But of course the MPI buffers must remain on the CPU, therefore we can not avoid a small number of transfers (of 2D cut planes)

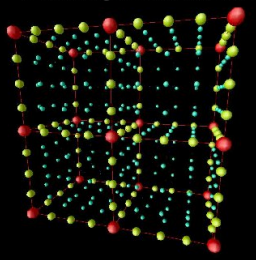
Porting SPECFEM3D on CUDA

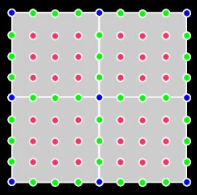
- At each iteration of the serial time loop, three main types of operations are performed:
  - update (with no dependency) of some global arrays composed of the unique points of the mesh
  - purely local calculations of the product of predefined derivative matrices with a local copy of the displacement vector along cut planes in the three directions (i, j and k) of a 3D spectral element
  - atomic operation to sum (assemble) the contributions

Graphics cards

Porting SPECFEM3D on CUDA: global numbering versus local numbering

- In 3D and for NGLL = 5, for a regular hexahedral mesh there are:
  - 125 GLL integration points in each element
  - 27 belong only to this element
  - 98 belong to several elements

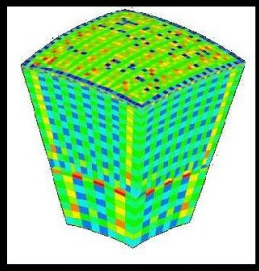


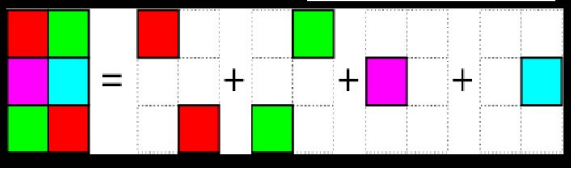


=> one thread per grid point (i.e., 125 threads per finite element)

Porting SPECFEM3D on CUDA: mesh coloring

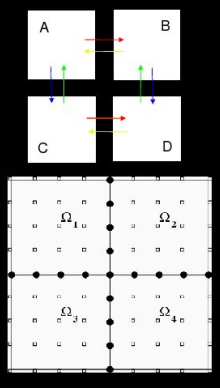
- Key challenge: ensure that contributions from two local nodes never update the same global value from different warps
- Use of mesh coloring: suppress dependencies between mesh points inside a given kernel

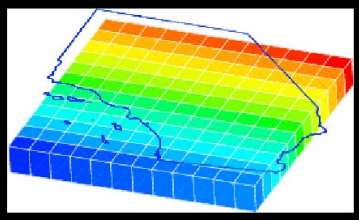




CUDA + MPI

- Previous (classical) communication scheme (blocking MPI)





Cost of exchanging and processing all the MPI buffers ~ 3%

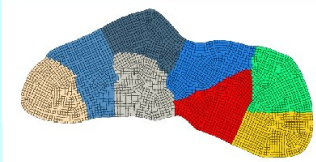
Use non-blocking MPI

80 domaines : nombre équivalent d'éléments internes et aux interfaces

Danielson and Namburu (1998)

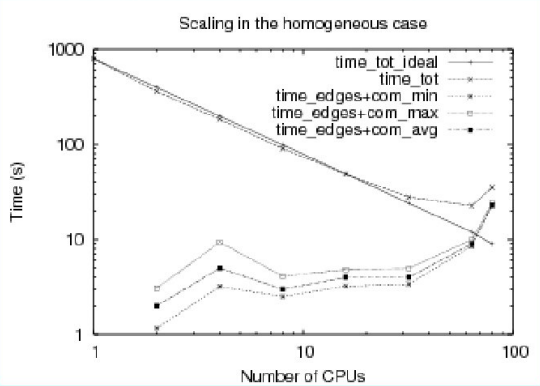
8 domaines : nombre d'éléments aux interfaces

< nombre d'éléments internes



Collaboration with Roland Martin and Nicolas Le Goff (Univ of Pau, France)

Scaling and communications

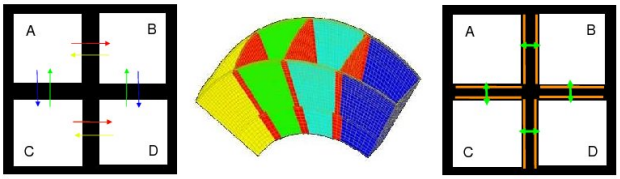


Problem: not true any more if calculations are too fast!! (e.g. if speedup = 25)
Collaboration with Roland Martin and Nicolas Le Goff (Univ of Pau, France)

Porting SPECFEM3D on CUDA: adding MPI

Old communication scheme (blocking MPI)
Update done in the whole arrays
(all elements computed before starting MPI calls)

New communication scheme
(non blocking MPI)
Update done in buffers (for outer mesh elements first)

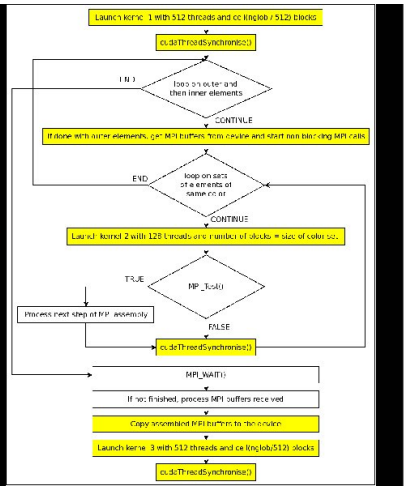


MPI communications cost on GPU version ~ 5%,

> We need to use non-blocking MPI communications.

> MPI communications are very well overlapped by computations on the GPU.

Structure of one time step in the CUDA code

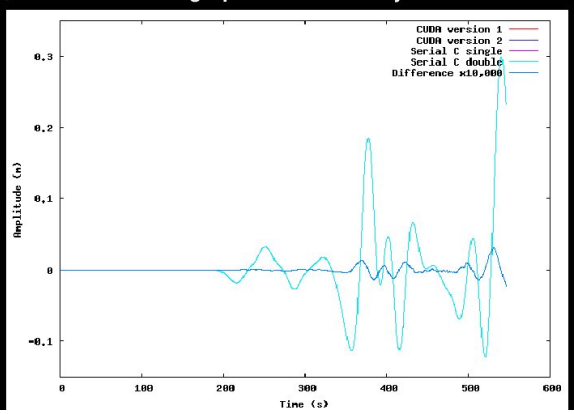


Porting SPECFEM3D on CUDA: Coalesced Global Memory Accesses

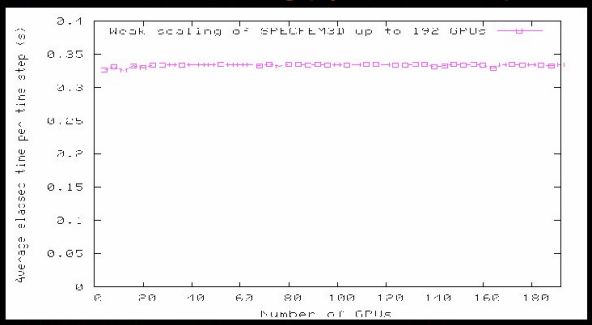
- **To ensure coalesced reads from global memory, the local array sizes are a multiple of 128 floats (which is itself a multiple of the half-warp size of 16) instead of 5^3 = 125 (thus purposely wasting 128/125 = 1.023 = 2.3% of memory)**
- **Each thread is responsible for a different point in the element. Consequently, the threads of a half-warp load adjacent elements of a (float) array. Access to global memory is thus perfectly coalesced in kernels 1 and 3, as well as in the parts of kernel 2 that access local arrays**
- **When accessing global arrays in kernel 2, the indirect addressing necessary to handle the unstructured mesh topology results in non-coalescent accesses and 5-way bank conflicts**

Porting SPECFEM3D on CUDA: validation

- **Validation and single precision efficiency:**



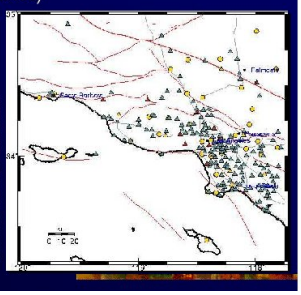
Multi-GPU weak scaling (up to 192 GPUs)



- **It is difficult to define speedup: versus what?**
- **For us, on the CEA/CCRT/GENCI GPU/Nehalem cluster, about 10x versus all the CPU cores, 20x for one GPU versus one CPU core.**

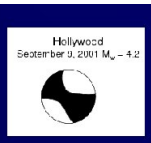
Wave propagation in basins

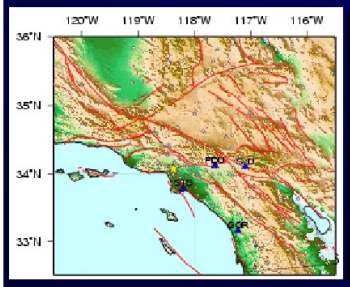
- Need accurate numerical methods to model seismic hazard – very densely populated areas
- Large and complex 3D models (e.g., L.A., Tokyo, Mexico)
- Wealth of high-quality data (TriNet)



Hollywood earthquake

Small M 4.2 earthquake on Sept 9, 2001



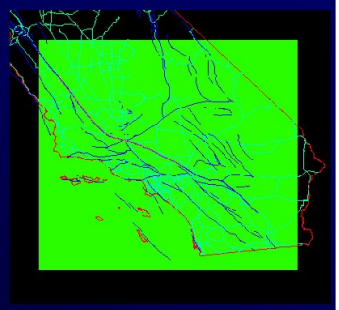


Amplification in basin

San Andreas fault - Carrizo Plain

Horizontal scale approximately 200 m

San Andreas – January 9, 1857



Vertical scale approximately 1 km

Scale approximately 500 km

Carrizo Plain, San Andreas Fault, California, USA

Conclusions and future work

Dimitri Komatitsch, David Michéa and Gordon Erlebacher, Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA, Journal of Parallel and Distributed Computing, vol. 69(5), p. 451-460, doi: 10.1016/j.jpdc.2009.01.006 (2009).

Dimitri Komatitsch, Dominik Göddeke, Gordon Erlebacher and David Michéa, Modeling the propagation of elastic waves using spectral elements on a cluster of 192 GPUs, Computer Science – Research and Development, in press (2010).

- CUDA on a single GPU leads to a speedup of 25x for our application versus a single CPU core
- We keep a speedup of 20x when we use a cluster of GPUs with non-blocking MPI (10x if we compare to all the CPU cores)
- But it is crucial to use larger domains to compensate for the very high speedup
- In future work, we could use OpenCL
- Need some kind of OpenMP for GPUs: CAPS HMPP, StarSs, StarPU...

2.5 Guillaume Colin de Verdière (CEA DAM/DIF)

Retour d'expériences du calcul hybride au CEA DAM

Sur la base d'un travail d'adaptation de code pour GPU, cette présentation dégage quelques recommandations pour le portage d'applications. Nous aborderons des aspects pratiques de programmation et montrerons comment se préparer dès maintenant à l'arrivée du calcul hybride.

Retour d'expérience de portages de codes sur GPU

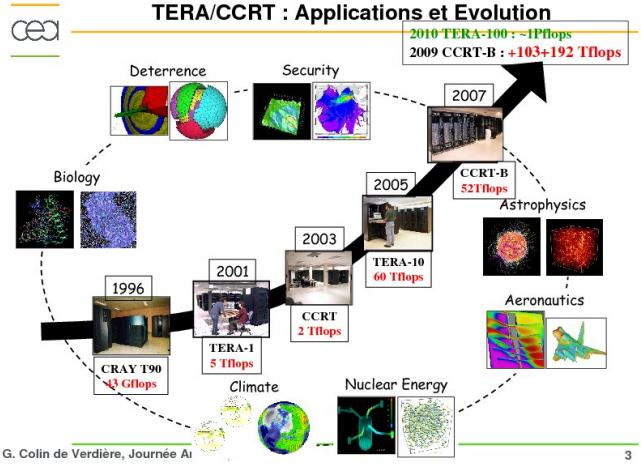
G. Colin de Verdière

Journée Aristote 25 mars 2010

Plan

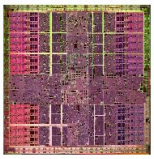
- Configurations matérielles au CEA
- Les langages de programmation étudiés
- Les tests PRACE/WP8
- Les portages de code
- Quelques remarques
- Conclusions

TERA/CCRT : Applications et Evolution



Les diverses configurations au CEA

Tesla S1070 credit NVIDIA



T10 credit NVIDIA

- **Machines de test**
  - Bull servers – 2 Haperton, 8GB
    - 2 **NVIDIA Tesla S1070**
    - IB DDR
  - AMD/ATI
    - Turion II dual core
    - **ATI 4850**
- **CCRT Cluster Graphique**
  - 40 Quadro FX 5800 (8 cores Haperton, 64GB/128GB)
    - T10 based
- **CCRT Titane**
  - Deux partitions dans 1068 nœuds
    - Partition standard : 972 nœuds 103TFlops
    - Partition hybride : 96 serveurs + **48 NVIDIA Tesla** S1070 **192 TFlops SP**
  - Serveurs = BULL Novascale R42x
    - Intel Nehalem-EP, 8GB, IB DDR

Langages de programmation pour GPU étudiés

- **CUDA**
  - Natif sur Tesla, apprentissage aisé
  - Les performances les plus élevées
  - Des debuggers et de profilers existent
    - Action CEA / Allinea sur portage DDT pour CUDA
- **CAPS/HMPP**
  - Apprentissage aisé
  - C et FORTRAN
  - Multi cible (CUDA, CAL/IL, [OpenCL])
- **OpenCL**
  - Prometteur mais axé portabilité aujourd'hui (performances ?)
  - Verbeux, peu outillé, …
  - Usage compatible avec des gros codes (FORTRAN) ?

Tests WP8 axés Kernels

- **Dans le cadre de PrACE WP8**
  - Objectif = tester les environnements de programmation et surtout HMPP sur machine hybride.
- **Mod2am : multiplication de matrices**
  - Portage HMPP aisé, performances décentes comparées à la DGEMM de cublas (~40GFlops sur 78Gflops peak DP).
- **Mod2as : sparse matrix vector multiplication**
  - Le benchmark le plus difficile
    - Indirection + réduction
  - Performances faibles de ce noyau
- **Leçons:**
  - Avoir une vision globale du portage, de simples noyaux ne suffisent pas
  - HMPP permet d'atteindre des bonnes performances

Portage de codes au CEA

- **Première expérience : EOS en SP**
  - problème "0D" simple à isoler
  - EOS de type tabulées
  - Gain significatif sur le temps calcul total
  - Permet d'envisager des formulations complexes
  - Motivation forte pour investiguer les GPU en profondeur
- **Code de propagation de polluants**
- **Code "équation de la chaleur"**
- **Code Hydro**
- **Voir aussi les grands challenges GPU au CCRT**
  - www-ccrt.cea.fr

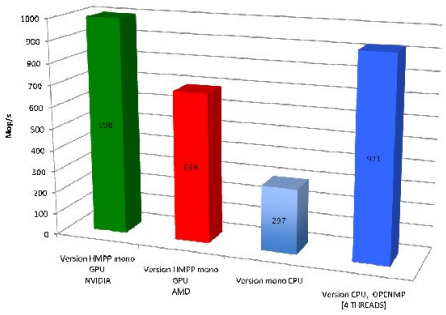
Migration de polluants

- **Code d'advection de particules**
  - Grille définie par des conditions météorologiques
  - Les particules sont avancées par le vent avec une dispersion simulée par une perturbation aléatoire des vecteurs vitesse
- **Objectif initial : portage CUDA (puis HMPP)**
- **Des difficultés importantes de portage**
  - Restructuration majeure imposée : lecture des données très enfouie dans le code, boucles de calcul mal découplées
  - Adaptation de la grille pour accroitre la localité des données
    - Perturbation significative des résultats rendant difficile la non régression
  - Générateur aléatoire utilisé pour chacune de particules
    - Difficulté d'assurer le comportement gaussien en parallèle
- **Au final, il faudrait ré-écrire le code …**

Equation de la chaleur

- **© Olivier Cessenat (CEA/CESTA) + Romain Dolbeau (CAPS)**
  - Grille régulière
- **Test des gains apportés par HMPP**
  - Performances significatives atteintes rapidement
- **Tests des nouveaux drivers de HMPP**
  - Backend CAL/IL (utilisation des cartes ATI)

Application HEAT - Résultats



Résultats hors transfert des données

www.caps-entreprise.com

Hydro

- **© Romain Teyssier (CEA/IRFU), PF Lavallée (IDRIS)**
- **Code d'hydrodynamique**
  - Solveur de Godunov et résolution du problème de Riemann aux interfaces
  - Grille fixe, directions alternées
- **Au départ = un code F90**
  - Programmation claire, sert de support de cours à la parallélisation OpenMP et MPI.
- **But : avoir un benchmark représentatif de la migration de code à l'usage de GPU.**
- **Démarche**
  - Portage en C (pour préparer la suite)
  - Portage HMPP
  - Portage CUDA (performances maximales)
  - A venir : portage OpenCL natif (profiter de la base CUDA)
    - OpenMP et version optimisée aussi
    - Version Hybride MPI + GPU
- **On cherche des MODIFICATIONS FAIBLES de l'esprit initial du code**
  - Mêmes routines, variables, structures de contrôle

Hydro : portage C

- **Remplacement des modules par des structures (et .h)**
- **Gestion des tableaux 2D/3D par des macros**
  - C++ aurait pu être plus confortable ici.
    - Pas supporté par HMPP ni CUDA (attendre 3.0)
- **Transformation des subroutines fortran en fonctions pures**
  - Le plus contraignant (passage par arguments)
- **Suppression des mélanges calcul / « ossature »**
- **Résultat : code complètement fonctionnel**
  - Code C aussi rapide qu'en FORTRAN

Module

```
module hydro_commons
  use hydro_precision
  integer(kind=prec_int) ::
  imin,imax,jmin,jmax

  real(kind=prec_real),allocatable,
  dimension(:,:,:) :: uold
  real(kind=prec_real)    :: t=0.
  integer(kind=prec_int) :: nstep=0
end module hydro_commons
```

- Masque l'aspect « global » de la variable
- Peut être « allocatable »
  - Ne définit pas sa durée de vie
  - Centraliser les allocations
- (mêmes remarques pour le C)

Fonctions pures

```
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <stdio.h>

#include "parametres.h"
#include "utils.h"
#include "slope.h"

#define DABS(x) (double) fabs((x))

void
slope(double *RESTRICT q,
      double *RESTRICT dq,
      const long narray,
      const long Hnvar,
      const long Hnxyt,
      const double slope_type)
{
    long n, i, ijmin, ijmax;
    double dlft, drgt, dcen, dsgn, slop, dlim;
    long ihvwin, ihvwimn, ihvwipn;

#define IHVW(i, v) ((i) + (v) * Hnxyt)

    ijmin = 0;
    ijmax = narray;
    for (n = 0; n < Hnvar; n++) {
        for (i = ijmin + 1; i < ijmax - 1; i++) {
            ihvwin = IHVW(i, n);
            ihvwimn = IHVW(i - 1, n);
            ihvwipn = IHVW(i + 1, n);
            dlft = slope_type * (q[ihvwin] - q[ihvwimn]);
            drgt = slope_type * (q[ihvwipn] - q[ihvwin]);
            dcen = half * (dlft + drgt) / slope_type;
            dsgn = (dcen > 0) ? (double) 1.0 : (double) -
1.0;    // sign(one, dcen);
            slop = (double) MIN(DABS(dlft), DABS(drgt));
            dlim = slop;
            if ((dlft * drgt) <= zero) {
                dlim = zero;
            }
            dq[ihvwin] = dsgn * (double) MIN(dlim,
DABS(dcen));
        }
    }
}                               // slope

#undef IHVW
//EOF
```


Mélange calcul ossature : avant

```
subroutine godunov(idim,dt)
! ...

 ! Update boundary conditions
 call make_boundary(idim)

 if (idim==1)then

   ! Allocate work space for 1D sweeps
   call allocate_work_space(imin,imax,nx+1)

   do j=jmin+2,jmax-2

     ! Gather conservative variables
     do i=imin,imax
       u(i,ID)=uold(i,j,ID)
       u(i,IU)=uold(i,j,IU)
       u(i,IV)=uold(i,j,IV)
       u(i,IP)=uold(i,j,IP)
     end do
     if(nvar>4)then
       do in = 5,nvar
         do i=imin,imax
           u(i,in)=uold(i,j,in)
         end do
       end do
     end if
```

```
! Convert to primitive variables
   call constoprim(u,q,c)

   ! Characteristic tracing
   call trace(q,dq,c,qxm,qxp,dtdx)

   do in = 1,nvar
     do i=1,nx+1
       qleft (i,in)=qxm(i+1,in)
       qright(i,in)=qxp(i+2,in)
     end do
   end do

   ! Solve Riemann problem at interfaces
   call riemann(qleft,qright,qgdnv,
                rl,ul,pl,cl,wl,rr,ur,pr,cr,wr,ro,uo,po,co,wo, &
                rstar,ustar,pstar,cstar,sgnm,spin,spout, &
                ushock,frac,scr,delp,pold,ind,ind2)

   ! Compute fluxes
   call cmpflx(qgdnv,flux)

   ! Update conservative variables
   do i=imin+2,imax-2
     uold(i,j,ID)=u(i,ID)+(flux(i-2,ID)-flux(i-1,ID))*dtdx
     uold(i,j,IU)=u(i,IU)+(flux(i-2,IU)-flux(i-1,IU))*dtdx
     uold(i,j,IV)=u(i,IV)+(flux(i-2,IV)-flux(i-1,IV))*dtdx
     uold(i,j,IP)=u(i,IP)+(flux(i-2,IP)-flux(i-1,IP))*dtdx
   end do
```


Mélange calcul ossature : après

```
make_boundary(idim, H, Hv);

// Allocate work space for 1D sweeps
allocate_work_space(H, Hw, Hvw);

if (idim == 1) {
  for (j = H.jmin + ExtraLayer; j < H.jmax - ExtraLayer; j++) {
    double *qID = &q[IHvw(0, ID)];
    double *qIP = &q[IHvw(0, IP)];
    gatherConservativeVars(idim, j, uold, u, H.imin, H.imax, H.jmin,
                           H.jmax, H.nvar, H.nxt, H.nyt, H.nxyt);

    // Convert to primitive variables
    constoprim(u, q, e, H.nxt, H.nxyt, H.nvar, H.smallr);
    equation_of_state(qID, e, qIP, c, 0, H.nxt, H.smallc, H.gamma);
    Dmemset(dq, 0, (H.nxyt + 2) * H.nvar);

    // Characteristic tracing
    if (H.iorder != 1) {
      slope(q, dq, H.nxt, H.nvar, H.nxyt, H.slope_type);
    }
```

```
trace(q, dq, c, qxm, qxp, dtdx, H.nxt, H.scheme, H.nvar,
      H.nxyt);
      qleftright(idim, H.nx, H.ny, H.nxyt, H.nvar, qxm, qxp,
qleft, qright);

      // Solve Riemann problem at interfaces
      riemann(qleft, qright, qgdnv,
              rl, ul, pl, cl, wl, rr, ur, pr, cr, wr, ro, uo, po, co, wo,
              rstar, ustar, pstar, cstar,
              sgnm, spin, spout, ushock, frac,
              scr, delp, pold, ind, ind2,
              H.nx + 1, H.smallr, H.smallc, H.gamma,
H.niter_riemann,
              H.nvar, H.nxyt);

      // Compute fluxes
      cmpflx(qgdnv, flux, H.nxt, H.nxyt, H.nvar, H.gamma);
      updateConservativeVars(idim, j, dtdx, uold, u, flux,
H.imin,
                             H.imax, H.jmin, H.jmax, H.nvar, H.nxt,
H.nyt, H.nxyt);
    }                           // for j
```


Exemples tirés du code : MAX / réduction / USE

```
subroutine cmpdt(dt)
 ! // déclarations idoines ...
 ! compute time step on grid interior
 cournox = zero
 cournoy = zero
 allocate(q(1:nx,1:lP))
 allocate(e(1:nx),c(1:nx))
 do j=jmin+2,jmax-2
   do i=1,nx
     q(i,ID) = max(uold(i+2,j,ID),smallr)
     q(i,IU) = uold(i+2,j,IU)/q(i,ID)
     q(i,IV) = uold(i+2,j,IV)/q(i,ID)
     eken = half*(q(i,IU)**2+q(i,IV)**2)
     q(i,IP) = uold(i+2,j,IP)/q(i,ID) - eken
     e(i)=q(i,IP)
   end do
   call eos(q(1:nx,ID),e,q(1:nx,IP),c)
   cournox=max(cournox,maxval(c(1:nx)+abs(q(1:nx,IU))))
   cournoy=max(cournoy,maxval(c(1:nx)+abs(q(1:nx,IV))))
 end do
 deallocate(q,e,c)
 dt = courant_factor*dx/max(cournox,cournoy,smallc)
end subroutine cmpdt
```

Tableaux temporaires

Variable globale

Branchements cachés

Réductions

« Array syntax » masquant la boucle

Algorithme « spécial CPU »

```
subroutine riemann(qleft,qright,qgdnv, &
! Pressure, density and velocity
 do i=1,nface
   rl(i)=MAX(qleft (i,ID),smallr)
   ul(i)=   qleft (i,IU)
   pl(i)=MAX(qleft (i,IP),rl(i)*smallp)
   rr(i)=MAX(qright(i,ID),smallr)
   ur(i)=   qright(i,IU)
   pr(i)=MAX(qright(i,IP),rr(i)*smallp)
 end do

 ! Lagrangian sound speed
 do i=1,nface
   cl(i) = gamma*pl(i)*rl(i)
   cr(i) = gamma*pr(i)*rr(i)
 end do

 ! First guess
 wl = sqrt(cl); wr = sqrt(cr)
 pstar = ((wr*pl+wl*pr)+wl*wr*(ul-ur))/(wl+wr)
 pstar = MAX(pstar,0.d0)
 pold = pstar

 n = nface
 do i = 1,n
   ind(i)=i
 end do
```

```
! Newton-Raphson iterations to find pstar at the required accuracy
do iter = 1,niter_riemann
 do i=1,n
   wwl=sqrt(cl(ind(i))*(one+gamma6*(pold(i)-pl(ind(i)))/pl(ind(i))))
   wwr=sqrt(cr(ind(i))*(one+gamma6*(pold(i)-pr(ind(i)))/pr(ind(i))))
   ql=two*wwl**3/(wwl**2+cl(ind(i)))
   qr=two*wwr**3/(wwr**2+cr(ind(i)))
   usl=ul(ind(i))-(pold(i)-pl(ind(i)))/wwl
   usr=ur(ind(i))+(pold(i)-pr(ind(i)))/wwr
   delp(i)=MAX(qr*ql*(qr+ql)*(usl-usr),-pold(i))
 end do
 do i=1,n
   pold(i)=pold(i)+delp(i)
 end do
 ! Convergence indicator
 do i=1,n
   uo(i)=ABS(delp(i)/(pold(i)+smallpp))
 end do
 n_new=0
 do i=1,n
   if(uo(i)>1.d-06)then
     n_new=n_new+1
     ind2(n_new)=ind (i)
     po (n_new)=pold(i)
   end if
 end do
 j=n_new
 do i=1,n
   if(uo(i)<=1.d-06)then
     n_new=n_new+1
     ind2(n_new)=ind (i)
     po (n_new)=pold(i)
   end if
 end do
 ind (1:n)=ind2(1:n)
 pold(1:n)=po (1:n)
 n=j
end do ! Niter_riemann
```

La boucle sur **n** sert à réduire le travail à chaque itération sur les faces non convergées

Version compatible GPU – version 1

```
for (iter = 0; iter < Hniter_riemann; iter++) {
    double precision = 1.e-6;
    for (i = 0; i < nface; i++) {
      if (ind[i] == 1) {
        wwl = sqrt(cl[i] * (one + gamma6 * (pold[i] - pl[i]) / pl[i]));
        wwr = sqrt(cr[i] * (one + gamma6 * (pold[i] - pr[i]) / pr[i]));
        ql = two * wwl * Square(wwl) / (Square(wwl) + cl[i]);
        qr = two * wwr * Square(wwr) / (Square(wwr) + cr[i]);
        usl = ul[i] - (pold[i] - pl[i]) / wwl;
        usr = ur[i] + (pold[i] - pr[i]) / wwr;
        delp[i] =
          MAX((double) (qr * ql / (qr + ql) * (usl - usr)),
              (double) (-pold[i]));
        pold[i] = pold[i] + delp[i];
        uo[i] = DABS(delp[i] / (pold[i] + smallpp));
        if (uo[i] <= precision) {
          ind[i] = 0; // cellule qui n'est plus a considerer
        }
      }
    }                  // i
}                      // iter_riemann
```


Version compatible GPU – version 2

```
for (i = 0; i < nface; i++) {
    for (iter = 0; iter < Hniter_riemann; iter++) {
    double precision = 1.e-6;
    if (ind[i] == 1) {
        wwl = sqrt(cl[i] * (one + gamma6 * (pold[i] - pl[i]) / pl[i]));
        wwr = sqrt(cr[i] * (one + gamma6 * (pold[i] - pr[i]) / pr[i]));
        ql = two * wwl * Square(wwl) / (Square(wwl) + cl[i]);
        qr = two * wwr * Square(ww
        usl = ul[i] - (pold[i] - pl[i]) / w
        usr = ur[i] + (pold[i] - pr[i]) /
        delp[i] =
          MAX((double) (qr * ql / (q
              (double) (-pold[i]));
        pold[i] = pold[i] + delp[i];
        uo[i] = DABS(delp[i] / (pold[i] + smallpp));
        if (uo[i] <= precision) {
          ind[i] = 0; // cellule qui n'est plus a considerer
        }
      }
    }                  // iter_riemann
}                      // i
```

Inversion des deux boucles pour itérer sur les faces qui seront traitées par des threads CUDA

Fusion des boucles sur i au sein d'un unique kernel

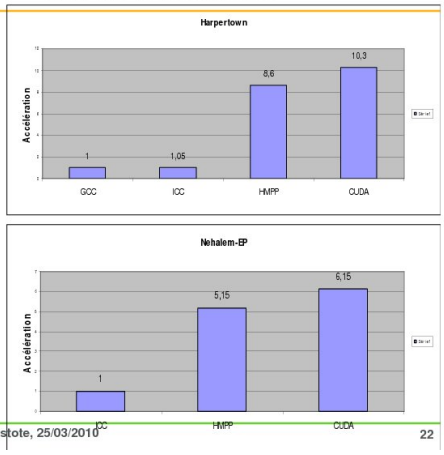
Hydro : portages HMPP & CUDA

- **HMPP**
  - Mise en place des directives incrémentalement
    - Mauvaises performances intermédiaires normales
      - Déplacements de données inutiles mais inévitables
  - La grosse difficulté : la maîtrise des mouvements de mémoire
    - Pas de grosses difficultés pour les noyaux
      - Aider le compilateur avec « hmppcg Parallel »
  - Au final des performances satisfaisantes
- **CUDA**
  - Portage simple grâce au travail préparatoire pour HMPP
  - Contrôle total des mouvements mémoire
  - Performances intéressantes atteintes rapidement

Bilan des performances

- **Cas test = 10000x10000 sur 10 itérations**
  - DP

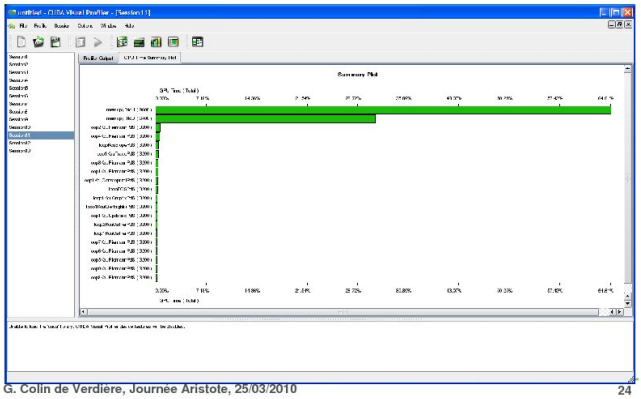
Harpertown 2.800Ghz 1626s gcc
Harpertown 2.800Ghz 1540.583s icc
Nehalem EP 2.933GHz 975.590s icc
Tesla T10 1.3GHz 189.087s HMPP
Tesla T10 1.3GHz 158.621s CUDA



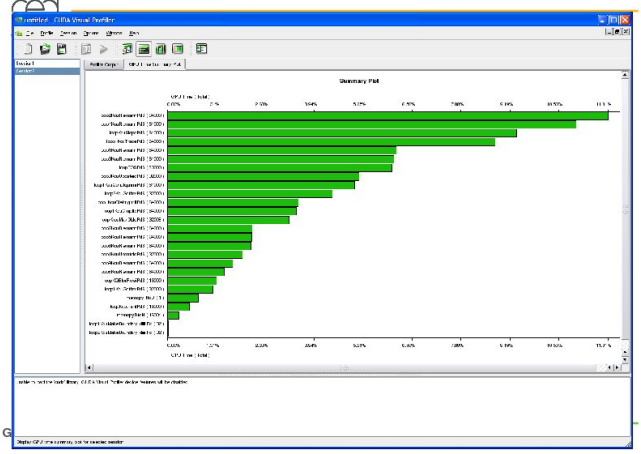
Leçons apprises avec Hydro

- **Accès mémoire**
  - Localité = éviter les indirections (quid du *non structuré*)
  - Distance = éviter l'usage du PCI-Express
- **Module/use = variables globales à isoler**
  - Privilégier les arguments
- **MAX/MIN = branchements néfastes cachés**
  - Sérialisation des threads CUDA SIMT
- **Réductions**
  - Attention à la parallélisation de cette construction (cf mod2as)
- **Attention aux algorithmes optimisés CPU (Ex: Riemann)**

Performances avant optimisation des mouvements mémoire



Performances après optimisation



Hydro : Conclusion

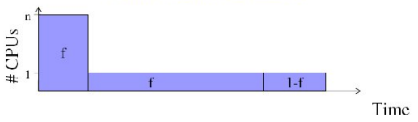
- **Avec la structure actuelle : gain de performances possible**
  - Il faut néanmoins beaucoup de mailles par direction
- **La méthode des directions alternées devrait se faire sur le maillage entier et non plus ligne à ligne**
  - Utilisation plus importante de la mémoire
    - (plans temporaires au lieu de lignes [en 2D])
  - Avantage = beaucoup plus de threads possibles
    - Réduction de la fraction séquentielle du code significative
  - Impact plus important sur le code à prévoir
    - Déplacement des boucles dans les sous-programmes
    - Tableaux de travail avec une dimension de plus
    - Probablement éclater les tableaux compactés

Numérique : pistes à explorer

- **Montée en ordre des schémas**
  - Inverser le ratio accès mémoire / nombre d'opérations de calcul
  - Nodal Discontinuous Galerkin Methods on Graphics Processors, A. Klockner, T. Warburton, J. Bridge, J. S. Hesthaven

Ne pas oublier la loi d'Amdahl

n=∞
f=0.00 s=1
f=0.10 s=1.1
f=0.20 s=1.25
f=0.30 s=1.42
f=0.40 s=1.66
f=0.50 s=2
f=0.60 s=2.5
f=0.70 s=3.33
f=0.80 s=5
f=0.90 s=10
f=0.95 s=20
f=0.99 s=100
f=0.999 s=1000



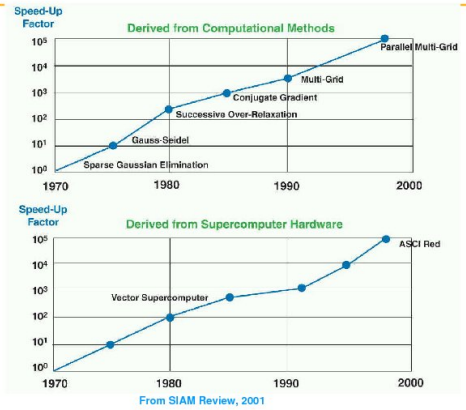
f – fraction that can run in parallel
1-f – fraction that must run serially

$$Speedup = \frac{1}{(1-f) + \frac{f}{n}} \qquad \lim_{n\to\infty} \frac{1}{1-f+\frac{f}{n}} = \frac{1}{1-f}$$

Penser massivement parallèle!

Ne pas oublier les algorithmes ;-)

"Is There A Moore's Law For Algorithms?" David E. Womble Sandia National Laboratories Presented at Salishan April 19, 2004



Conclusion

- ***Si possible*** **: réécrire complètement le code**
  - Quid des codes de productions « précieux » ?
- **Repenser les algorithmes**
  - Monter en ordre
  - Privilégier les méthodes simples à paralléliser
- **VECTORISER**
  - Pour passer facilement au massivement parallèle
- **Faire très attention aux mouvements de données**
- **Fonctions pures le plus souvent possible**

Chapitre 3

Présentations (après-midi)

3.1 Présentation du projet OpenGPU

3.1.1 Jean François Lavignon (BULL)

Historique

3.1.2 Jean-Noël de Galzain (Wallix)

Organisation et communication

Photos ©Aristote 2010

ADMINBASTION

KICK-OFF OPENGPU

www.opengpu.net

JNdeGALZAIN – jndegalzain@wallix.com

25/03/2010

WALLIX Infrastructure and Security Solutions

ADMINBASTION

GENESE

- CONSTRUCTION DU PROJET
  - Rapprochement des projets GTE et AGPU
  - Constitution d'un consortium académique et industriel
- PROCESSUS DE LABELLISATION
  - Présentation au GT OCDS de System@tic Paris Region
  - Présentation à Cap Digital
  - Présentation à Ter@tec
- LABELLISATION
  - Double labellisation et partenariat Ter@tec le 30 avril 2009
  - Projet retenu en Juillet 2009 par la DGCIS

Contenu confidentiel et propriété de Wallix

ADMINBASTION

CONSTAT

- UNE TECHNOLOGIE ORIENTEE HW MASS MARKET
  - Leaders : NVIDIA, INTEL, IBM – ATI
  - Marché : 10 Milliards €
  - Positionnement : le jeu vidéo, le graphisme
- DES PERFORMANCES EXPLOSIVES - 2015
  - 20 x la performance d'aujourd'hui
  - 5000 cœurs à 3Ghz (50mW each)
  - 20 Tflops
  - 1,2TB/s de mémoire vive
- DES NOUVEAUX MARCHES
  - Le calcul, la simulation
  - L'industrie
  - La génomique
  - La finance
  - Etc.

=> Les GPU POUR L'INDUSTRIE

Contenu confidentiel et propriété de Wallix

ADMINBASTION

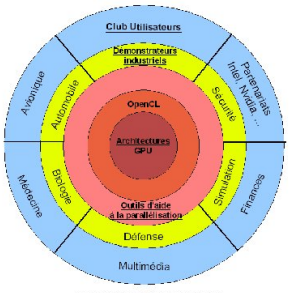
AMBITION & DEFIS

- UN PROJET AMBITIEUX ORIENTE INDUSTRIE
  - PARTENAIRES (nombre et qualité) : GE – PME – LABs et ORGANISMES R&D
  - DUREE (24 mois) et BUDGET (13,3 M€)
  - PARTICIPATION DE 2 GRANDS PÔLES : CAP Digital et System@tic
  - APPUI DE TERATEC
- DES VERROUS TECHNOLOGIQUES
  - Accès parallélisation : complexité et hétérogénéité ➔ outils, standard OpenCL
  - Modernisation des logiciels : quel code optimiser, gains potentiels, outils
  - Propriétaire ➔ standards (OpenCL + Eclipse + OpenSource)
  - Optimisation des architectures (cluster hybrides) : performances et consommation
  - Librairies industrielles pour faciliter déploiement

Contenu confidentiel et propriété de Wallix

ADMINBASTION

ORGANISATION & COMMUNICATION



Eco-système du projet OpenGPU

MARCHE du HPC :

- Matériel HPC
- Outils logiciels pour HPC
- Services liés au HPC
- Logiciels applicatifs HPC

PLANNING :

- Début des travaux Mars 2010
- Kick-Off 25/03/2010
- Court terme Lct Groupes W
- **Séminaire européen février 2011**
- Clôture projet février 2012

Contenu confidentiel et propriété de Wallix

ADMINBASTION

ORGANISATION & COMMUNICATION

- DECOUPAGE EN SOUS-PROJETS :

Package	Description	Partenaires	Durée	Livrables
WP1	Pilotage, organisation, dissémination	Resp: IF Research Wallix AS+ (contrôle qualité) Ter@tec	24 mois	
WP2	Outils d'aide à la parallélisation	Resp: HPC Project ARMINES, Atej ,CAPS Entreprises, CEA LIST, DIGITEO/Scilab,ECP/CRSA, INRIA/ALCHEMY, LIP6, Thalès	24 mois	Spécifications et programmation des outils d'aide à la parallélisation, intégration Eclipse
WP3	Architecture matérielle et logicielle	Resp: BULL CEA DAM GENCI	24 mois	Evaluation d'architectures hybrides, développement de clusters HPC à base de lames hybrides, chaîne de déploiement, benchmarks
WP4	Démonstrateurs industriels	Resp: NumTech/AS+ DIGITEO/Scilab, ECP/CRSA, ESI Group, INRA, IBISC/Université-Evry, Thales, Wallix	24 mois	Mise à disposition des librairies, évaluation des benchmarks, « Grand Challenge »

Contenu confidentiel et propriété de Wallix

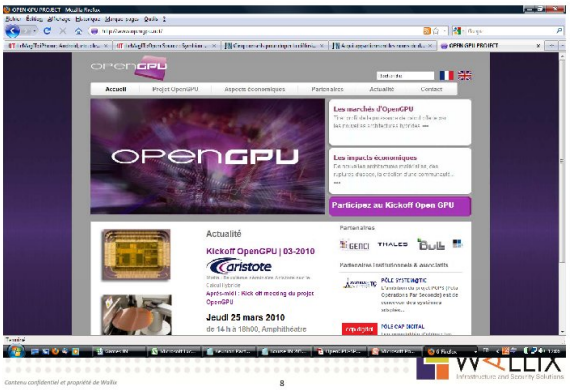
ADMINBASTION

OBJECTIFS

- Proposer une plate-forme logicielle et des outils Open Source autour du Standard OPEN CL
- **Industrialiser** des architectures **matérielles et logicielles** adéquates pour exploiter ces nouvelles puissances de calcul avec une composante **Green IT**
- **Construire en IDF un pôle d'excellence technologique et économique autour des architectures hybrides**

Contenu confidentiel et propriété de Wallix

ADMINBASTION



Contenu confidentiel et propriété de Wallix

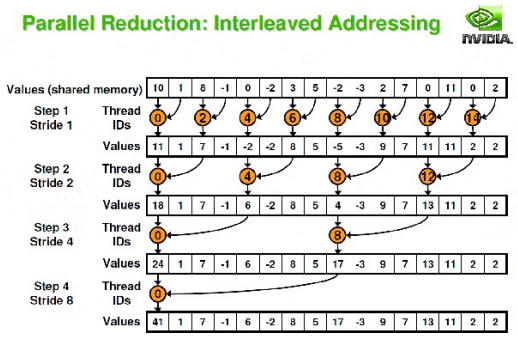
3.1.3 Eric Mahé (Minds Planet)

Verrous technologiques et Objectifs

Le Projet OpenGPU

Verrous Technologiques et Objectifs

Verrou #1: la complexité (1)



Verrou #1: la complexité (2)

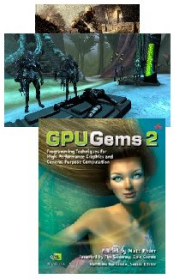
Performance for 4M element reduction NVIDIA

	Time (2^{22} ints)	Bandwidth	Step Speedup	Cumulative Speedup
Kernel 1: interleaved addressing with divergent branching	8.054 ms	2.083 GB/s		
Kernel 2: interleaved addressing with bank conflicts	3.456 ms	4.854 GB/s	2.33x	2.33x
Kernel 3: sequential addressing	1.722 ms	9.741 GB/s	2.01x	4.68x
Kernel 4: first add during global load	0.965 ms	17.377 GB/s	1.78x	8.34x
Kernel 5: unroll last warp	0.536 ms	31.289 GB/s	1.8x	15.01x
Kernel 6: completely unrolled	0.381 ms	43.996 GB/s	1.41x	21.16x
Kernel 7: multiple elements per thread	0.268 ms	62.671 GB/s	1.42x	30.04x

Kernel 7 on 32M elements: 73 GB/s!

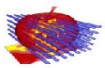
34

Verrou #2: l'opacité (1)



- La boîte noire des jeux vidéos:
  - Textures, occlusions, PBO, FBO, …
  - OpenGL, Direct X, PhysX, …
  - Cg, GLSL, HLSL, ...
  - Console, API, DRM, …
  - Cell, Nvidia, ATI, ...
- Les premiers pas du GPGPU:
  - Textures = données
  - Shaders = kernels

Verrou #2: l'opacité (2)

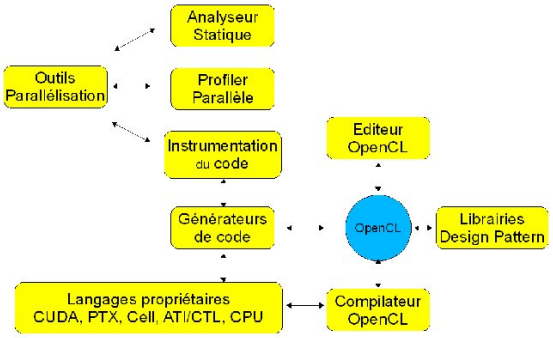


- Les exigences du HPC:
  - Maîtrise des compilateurs
    - Gestion des arrondis, Fortran, ...
  - Gestion des architectures
    - Modèles: OpenMP, MPI, …
    - Matériels: grid, clusters, …
    - Données: sauvegarde, duplication, …
  - Open Source everywhere:
    - Linux, Lustre, GCC, ZFS, …
- GPU = « boîte noire » ???

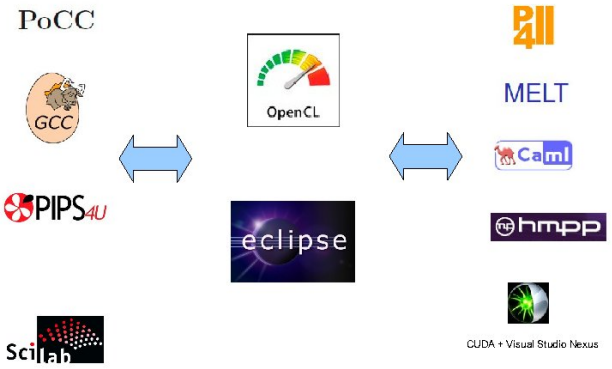
Verrou #3: l'estimation de la rentabilité (1)

- Quelles parties du code optimiser ?
- Quelle architecture choisir ?
- Comment mesurer les performances ?
- Comment estimer le ROI ?
- Quels impacts sur les développements ?
- Quels impacts sur la maintenance ?
- Quels impacts sur l'environnement ?

Objectif #1: un ensemble d'outils GPU



Objectif #2: Standards et Open Source

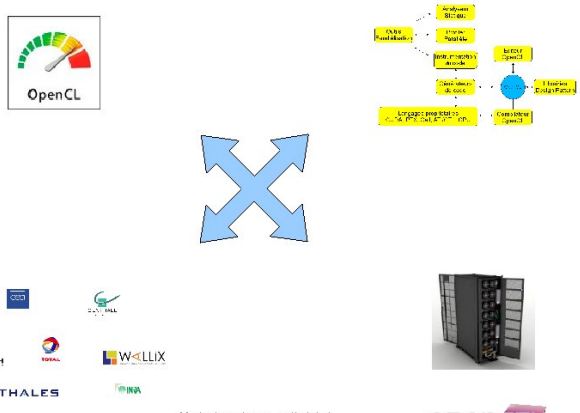


Objectif #3: Une architecture de référence

- Benchmarks
- Test, montée en charge, …
- Provisionning
- Green IT
  - Mesure de consommation
  - Mesures comparatives

Objectif #4: des codes optimisés

OpenGPU: un projet global



3.2 Outils et méthodes du projet OpenGPU

3.2.1 Ronan Keryell (HPC Project)

Présentation générale du SP2

OpenGPU
SP2

Outils d'aide à la parallélisation pour machines moulticœurs

Ronan Keryell

HPC Project

25/3/2010

http://hpc-project.com

Programmation parallèle

- Généralisation des machines moulticoeurs & hétérogènes
  - Ordinateurs de bureaux ou portables
  - Supercalculateur (domaine d'origine)
  - Systèmes embarqués (téléphones, voitures, radars...)
- 1 carte graphique (GPU nVidia Fermi ou AMD/ATI HD 5870) ≡ 2+ TFLOPS
- Futur : encore + hétérogène
- 3 standards de programmation parallèle
  - OpenMP : multithread pour les nuls (?), mémoire partagée
  - MPI : passage de message pour les nuls (?), mémoire distribuée, tâches
  - OpenCL : architectures hétérogènes (CPU, GPU, accélérateurs) pour les nuls (?)

Dure réalité du parallélisme (I)

http://www.phdcomics.com/comics/archive.php?comicid=1292

- Rajoute du sucre syntaxique
- Problème de DRH
- Importance croissante des pannes...

PhDcomics du 3/15/2010, visite de Jorge CHAM à l'X

Dure réalité du parallélisme (II)

We *really* need an elevator...

OpenCL

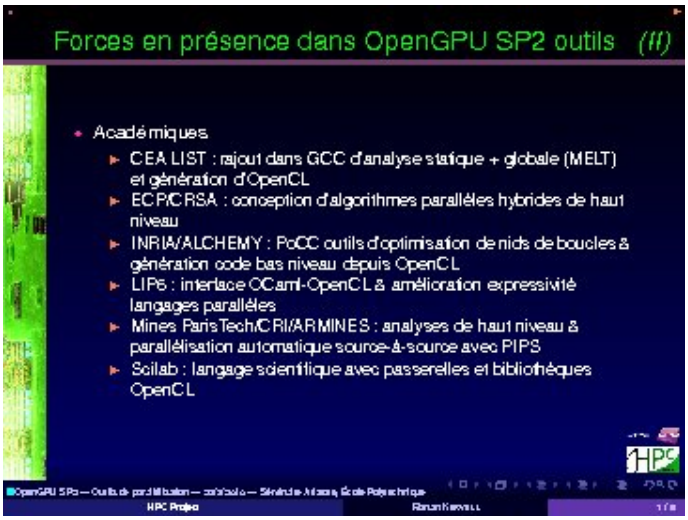
- Standard ouvert gratuit pour plateformes hétérogènes (CPU, GPU, accélérateurs)
- Khronos Group (OpenGL, OpenAL...) avec AMD, Apple, ARM, Broadcom, Freescale, IBM, Intel, nVidia, Sony...
- « OpenCL is a trademark of Apple Inc. used under license by Khronos » (≈ Java ∈ Sun/ORACLE, PDF ∈ Adobe...)
- Modèle de programmation basé sur
  - Parallélisme de tâches
  - Parallélisme de données
- Sous-ensemble de C99 avec... extensions pour parallélisme
- Calculs flottants standard IEEE-754
- Interaction avec graphisme (OpenGL...)

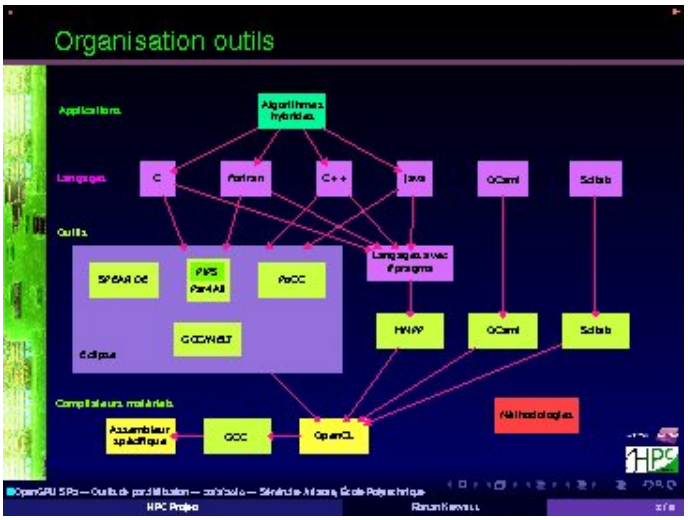
Bon début concret mais...

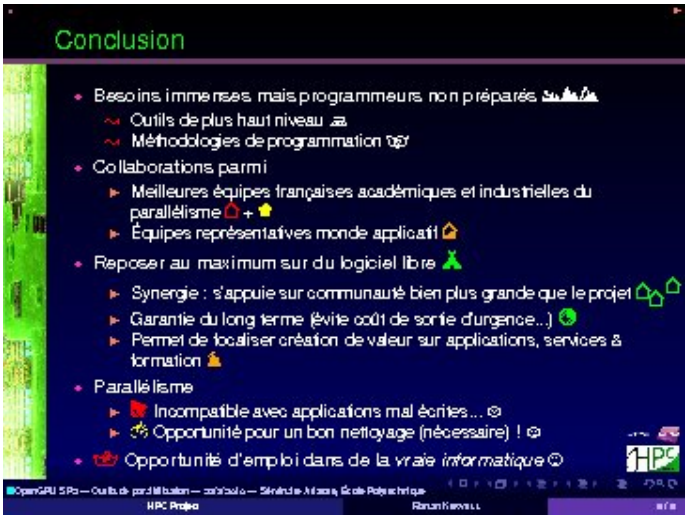
- Compliqué de réécrire ses applications, même 1 fois...
  - Applications en C, C++, Java, Fortran, Scilab, Matlab, OCaml...
- Ne résout pas difficultés intrinsèques de programmation parallèle
- Besoin de méthodologies & outils ⇝ OpenGPU SP2

Forces en présence dans OpenGPU SP2 outils (I)

- Industriels
  - CAPS Entreprise : HMPP Workbench pour programmation parallèle multicible avec #pragma communs, sortie OpenCL, intégration Eclipse
  - HPC Project : Par4All pour parallélisation multicible, instrumentation, sortie OpenCL, intégration Eclipse
  - Thales Research & Technology : environnement d'implémentation d'applications flots de données sur cibles parallèles embarquées, sortie OpenCL


Forces en présence dans OpenGPU SP2 outils (II)
Académiques
CEA LIST : rajout dans GCC d'analyse statique + globale (MELT) et génération d'OpenCL
ECP/CRSA : conception d'algorithmes parallèles hybrides de haut niveau
INRIA/ALCHEMY : PoCC outils d'optimisation de nids de boucles & génération code bas niveau depuis OpenCL
LIP6 : interface OCaml-OpenCL & amélioration expressivité langages parallèles
Mines ParisTech/CRI/ARMINES : analyses de haut niveau & parallélisation automatique source-à-source avec PIPS
Scilab : langage scientifique avec passerelles et bibliothèques OpenCL
HPC Project
Ronan Keryell


Organisation outils
Applications
Algorithmes hybrides
Langages
C
Fortran
C++
Java
OCaml
Scilab
Outils
SPEAR-DE
PIPS Par4All
PoCC
Langages avec #pragma
Eclipse
GCC/MELT
HMPP
OCaml
Scilab
Compilateurs matériels
Méthodologies
Assembleur spécifique
GCC
OpenCL
HPC Project
Ronan Keryell


Conclusion
Besoins immenses mais programmeurs non préparés
Outils de plus haut niveau
Méthodologies de programmation
Collaborations parmi
Meilleures équipes françaises académiques et industrielles du parallélisme
Équipes représentatives monde applicatif
Reposer au maximum sur du logiciel libre
Synergie : s'appuie sur communauté bien plus grande que le projet
Garantie du long terme (évite coût de sortie d'urgence...)
Permet de localiser création de valeur sur applications, services & formation
Parallélisme
Incompatible avec applications mal écrites...
Opportunité pour un bon nettoyage (nécessaire) !
Opportunité d'emploi dans de la vraie informatique
HPC Project
Ronan Keryell

3.2.2 Basile Starynkevitch (CEA LIST), Cedric Bastoul (INRIA Saclay/U. Paris Sud XI)

GCC, POCC, MELT

GCC et la génération de code pour GPU *via* des greffons.

OpenGPU GCC plugins work

CEA LIST [LSL]

MELT: Middle End Lisp Translator

INRIA/Saclay [Alchemy]

PoCC: Polyhedral Compilation Collection

Graphite: Loop Transformations and Automatic Vectorization in GCC

GCC assets & OpenGPU goals

- GCC: **huge** (4MLOC) growing (+35%/2y) legacy (1985!) & (≈ 400) developers' community
- New in 4.5: plugins + link-time optimizations
  - Many source languages & target systems
  - Internal representations (Generic Tree, Gimple, SSA)
  - Many (≈ 250) passes
- Contributed technology: Graphite, plugins, MELT
- OpenGPU goals on GCC:
  - Generate OpenCL (from Gimple)
  - Explore GPGPU issues in GCC

Extending & customizing GCC

- Extend GCC to suit your own needs
  - Specific optimizations (for GPU, for a software, for a system; machine learning techniques = CTuning/Milepost)
  - Custom diagnostics (for a software, a company policy)
  - Any source code processing (navigation, retro-factoring, aspect oriented programming)
- Develop painfully your complex plugins in C ☠
- Better use MELT ♥ to code your extensions

OpenGPU will provide GCC extensions/plugins

MELT

- MELT "Middle End Lisp Translator" = a GCC plugin providing a high-level *domain specific language* to extend GCC
- Lispy dialect (with garbage collection, functional/applicative/object programming, pattern-matching, C code chunks, ...)
- Dynamically translated into C code
  - MELT modules dlopen-ed by melt.so
- Fits tightly into GCC
  - Thru many linguistic devices

http://gcc.gnu.org/wiki/MiddleEndLispTranslator

Motivation

Portable performance

Polyhedral Compiler Technology

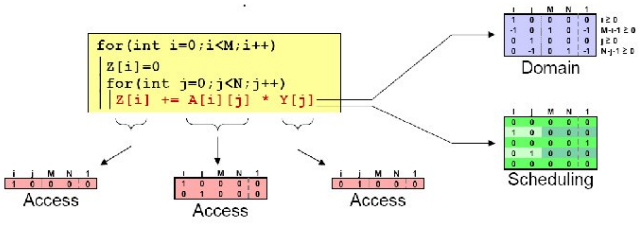
Balance multiple levels of parallelism and locality

Integrated in standard tool chains (GCC)

Variety of manycore processors and accelerators (GPU)

Polyhedral Representation

Algebraic program representation to compose complex loop transformations, for parallelism, locality and data layout of arrays



PoCC: Polyhedral Compilation Collection

- Integrated suite of research prototypes
  - Stand-alone code analysis and transformation tools (free software, LGPL and BSD-like)
  - Common text and library interface (`scoplib`)
    - Coupling with GCC/Graphite
    - Coupling with PIPS (École des Mines, HPC Project)
    - Worldwide contributors (Ohio State U., IBM Research Watson, K. U. Leuven)
  - Code generation for GPUs
    - CUDA and OpenCL backends almost ready
    - Automatic kernel offloading and array localization

`http://pocc.sourceforge.net`

7

GCC/Graphite Project

- Major ongoing R&D project in GCC
  - Part of GCC 4.4 (free software, GPL)
  - Started at INRIA, large contribution from AMD
  - Status: stable framework, optimization and parallelization heuristics being implemented
  - Ambitions
    - Most advanced loop transformation and automatic parallelization infrastructure in a production compiler
    - Modular design: usable outside GCC
    - In OpenGPU: integrate with PoCC and source-to-source compilation with GCC/MELT

`http://gcc.gnu.org/wiki/Graphite`

8

Contacts

- INRIA/Alchemy (PoCC, Graphite)
  - albert.cohen@inria.fr cedric.bastoul@inria.fr
- CEA LIST/LSL (MELT)
  - basile.starynkevitch@cea.fr

9

3.2.3 Ronan Keryell (HPC Project)

PIPS, PA4RALL, Spear DE

Les outils source à source et la génération de code CUDA et Open CL.

THALES

Thales Research & Technology

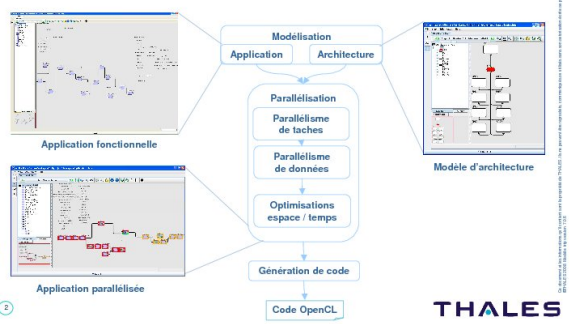
Kick-off OpenGPU – SP2

SPEAR Design Environment

Research & Technology

SPEAR Design Environment

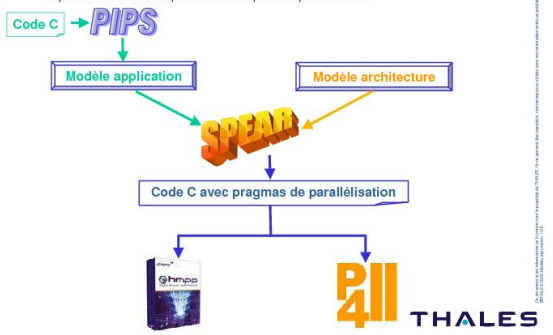
- Approche semi-automatique complémentaire des compilateurs parallèliseurs
- Flot de conception de haut niveau (vision applicative élargie)
- Génération de code OpenCL



THALES

Couplages avec SPEAR

- Import dans SPEAR d'un graphe applicatif depuis PIPS (parsing et analyse d'un code legacy)
- Génération par SPEAR d'un fichier C annoté (ex: transferts mémoire, scope des noyaux à placer sur GPU) conforme aux règles de codage de PAR4ALL et HMPP pour faciliter l'analyse et exploiter au mieux les capacités des compilateurs parallèliseurs



THALES

PIPS

François IRIGOIN

Mines ParisTech/CRI/ARMINES

25/3/2010

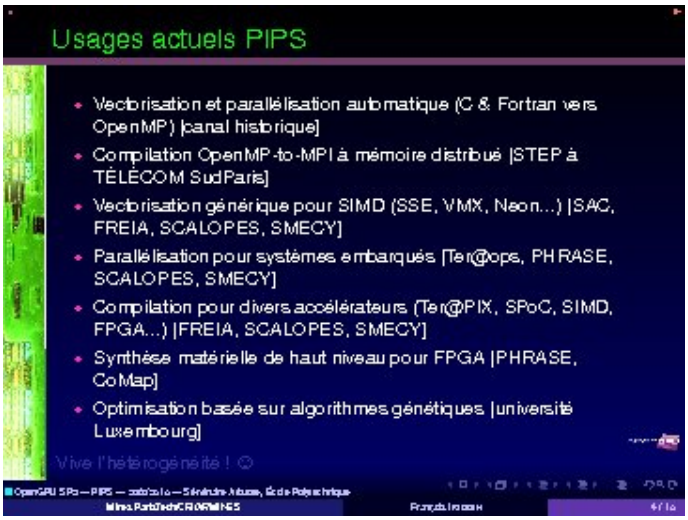
http://pips4u.org

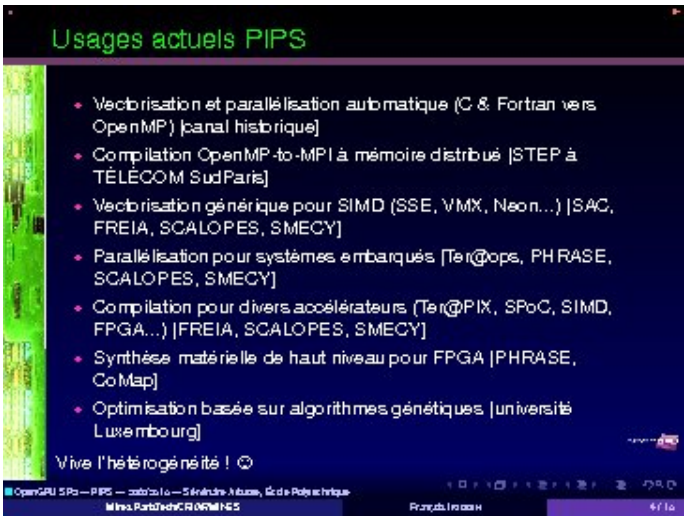
PIPS (I)

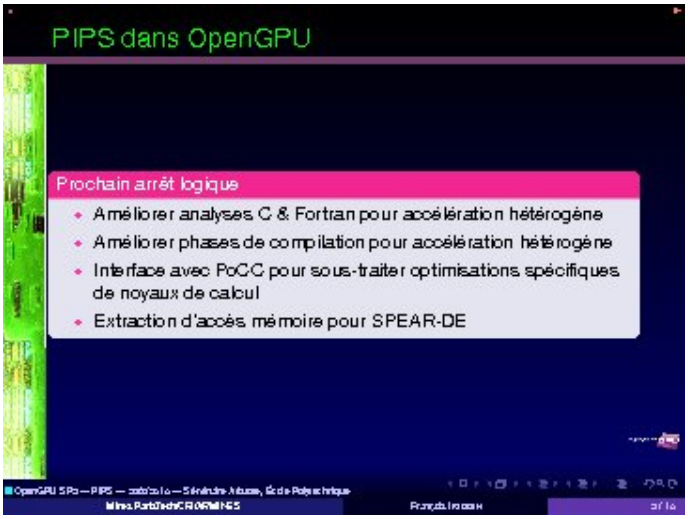
- PIPS (Parallélisation Interprocédurale de Programmes Scientifiques) : projet libre Mines ParisTech depuis 1988
- Source-à-source : portabilité + bénéficie des meilleurs infrastructures cibles
- Financé par nombreux organismes (DGA, ministère industrie & recherche, universités, CEA, IFP, Onera, ANR, projets européen, pôles de compétitivité...)
- Projet qui a déployé compilation polyédrique sur programmes complets
- ≈ 456 KLOC (9/2009) selon SLOCCount de David A. WHEELER
- ... mais approche modulaire et avant-gardiste qui a traversé les âges
  - ≈300 phases (parsers, analyses, transformations, optimiseurs, parallèliseurs, code générateurs, pretty-printers...) qui peuvent être combinées selon ses besoins

PIPS (II)

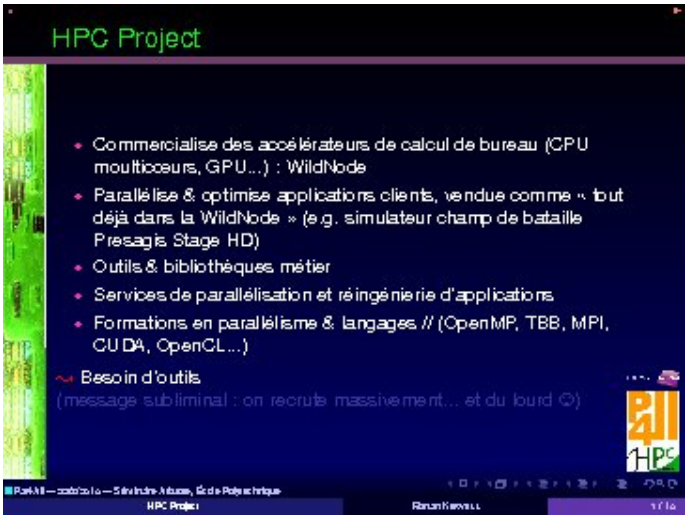
- Langage de description d'objets NewGen indépendante du langage de programmation : génération automatique de méthodes, persistance, introspection, visiteurs, accesseurs, constructeurs, sérialisation XML pour interfaçage avec autres outils...
- Moteur interprocédural à la make paresseux pour chaîner phases à la demande
- Programmation littérale et automatisation infrastructure avec générateurs de code
- Polyèdres (algèbre linéaire) pour analyse sémantique, transformations, génération de code... pour gérer avec gros programmes, pas que nids de boucle
- Extension infrastructure pour aller de Fortran77 vers du C99
- ≈ 20 programmeurs (Mines ParisTech, HPC Project, TÉLÉCOM SudParis, TÉLÉCOM Bretagne, RPI, université Luxembourg) via svn, Trac, listes de diffusion, IRC, Plone, Skype... qui l'utilisent dans leurs projets

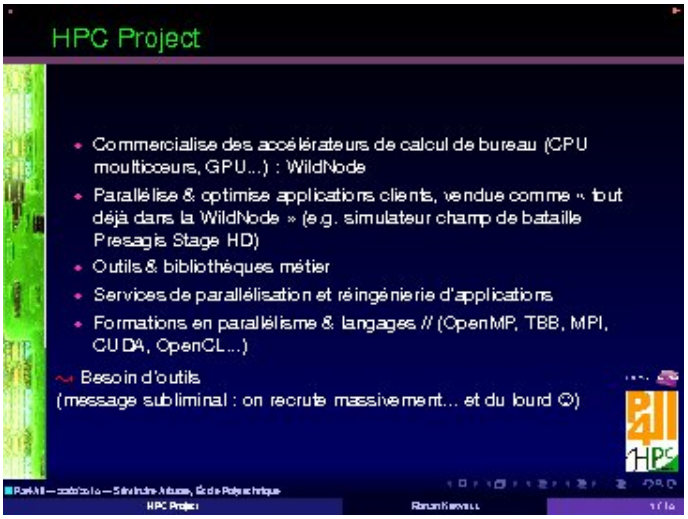

Usages actuels PIPS
Vectorisation et parallélisation automatique (C & Fortran vers OpenMP) [canal historique]
Compilation OpenMP-to-MPI à mémoire distribué [STEP à TÉLÉCOM SudParis]
Vectorisation générique pour SIMD (SSE, VMX, Neon...) [SAC, FREIA, SCALOPES, SMECY]
Parallélisation pour systèmes embarqués [Ter@ops, PHRASE, SCALOPES, SMECY]
Compilation pour divers accélérateurs [Ter@PIX, SPoC, SIMD, FPGA...] [FREIA, SCALOPES, SMECY]
Synthèse matérielle de haut niveau pour FPGA [PHRASE, CoMap]
Optimisation basée sur algorithmes génétiques [université Luxembourg]
Vive l'hétérogénéité ! ☺


Usages actuels PIPS
Vectorisation et parallélisation automatique (C & Fortran vers OpenMP) [canal historique]
Compilation OpenMP-to-MPI à mémoire distribué [STEP à TÉLÉCOM SudParis]
Vectorisation générique pour SIMD (SSE, VMX, Neon...) [SAC, FREIA, SCALOPES, SMECY]
Parallélisation pour systèmes embarqués [Ter@ops, PHRASE, SCALOPES, SMECY]
Compilation pour divers accélérateurs [Ter@PIX, SPoC, SIMD, FPGA...] [FREIA, SCALOPES, SMECY]
Synthèse matérielle de haut niveau pour FPGA [PHRASE, CoMap]
Optimisation basée sur algorithmes génétiques [université Luxembourg]
Vive l'hétérogénéité ! ☺

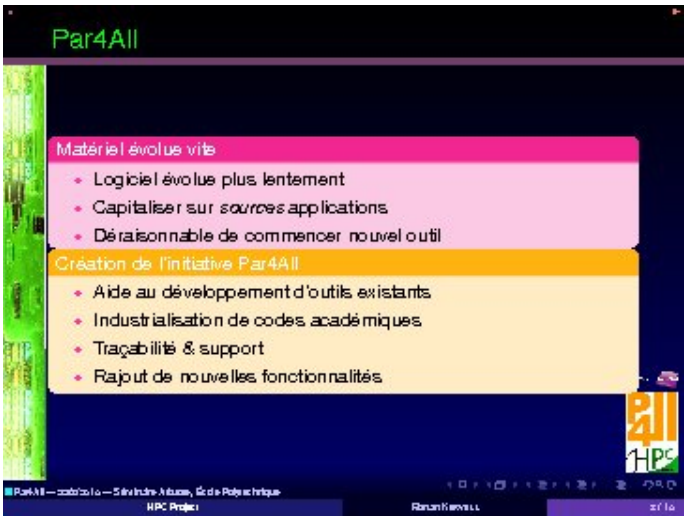

PIPS dans OpenGPU
Prochain arrêt logique
Améliorer analyses C & Fortran pour accélération hétérogène
Améliorer phases de compilation pour accélération hétérogène
Interface avec PoCC pour sous-traiter optimisations spécifiques de noyaux de calcul
Extraction d'accès mémoire pour SPEAR-DE

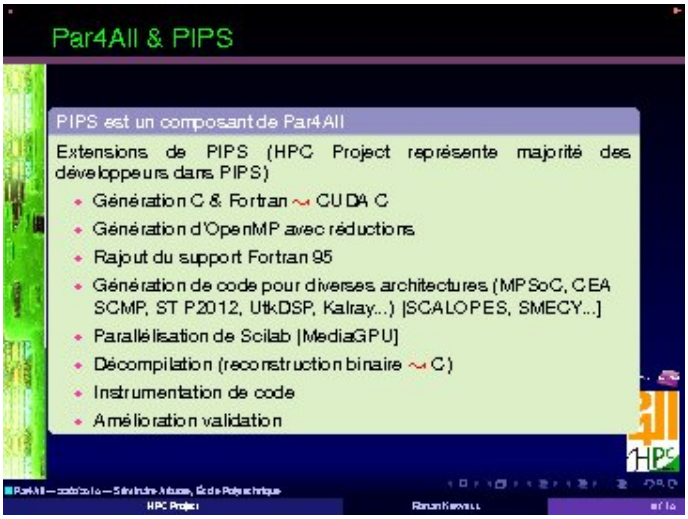

Par4All
Ronan Keryell
HPC Project
25/3/2010
http://par4all.org

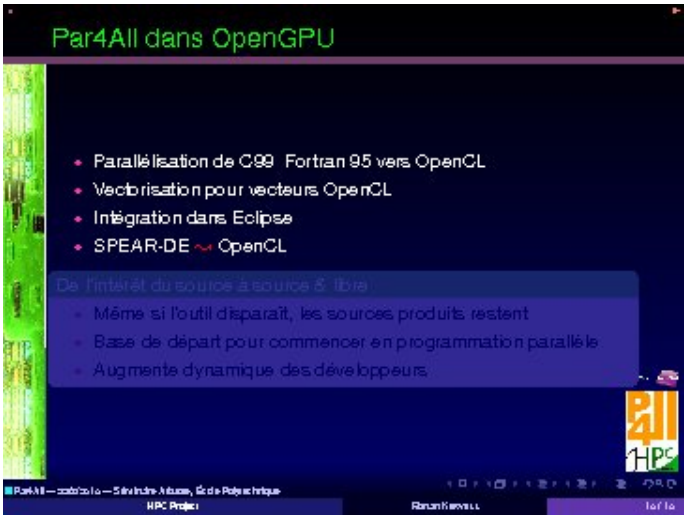

HPC Project
Commercialise des accélérateurs de calcul de bureau (CPU multicoeurs, GPU...) : WildNode
Parallélise & optimise applications clients, vendue comme « tout déjà dans la WildNode » (e.g. simulateur champ de bataille Presagis Stage HD)
Outils & bibliothèques métier
Services de parallélisation et réingénierie d'applications
Formations en parallélisme & langages // (OpenMP, TBB, MPI, CUDA, OpenCL...)
Besoin d'outils
(message subliminal : on recrute massivement... et du lourd ☺)

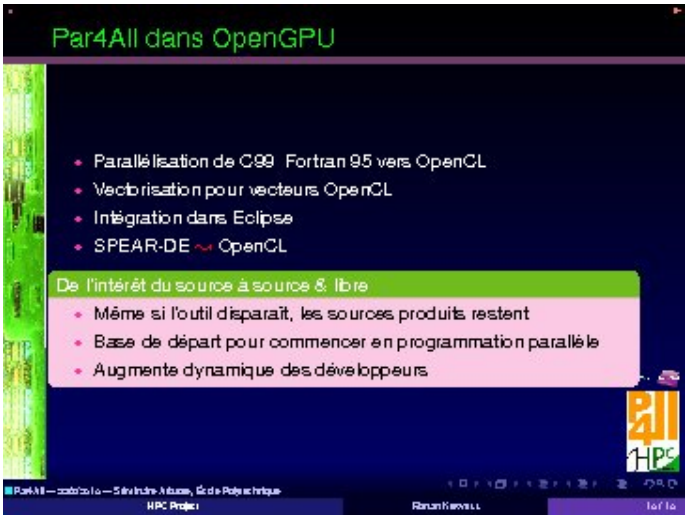

HPC Project
Commercialise des accélérateurs de calcul de bureau (CPU multicoeurs, GPU...) : WildNode
Parallélise & optimise applications clients, vendue comme « tout déjà dans la WildNode » (e.g. simulateur champ de bataille Presagis Stage HD)
Outils & bibliothèques métier
Services de parallélisation et réingénierie d'applications
Formations en parallélisme & langages // (OpenMP, TBB, MPI, CUDA, OpenCL...)
Besoin d'outils
(message subliminal : on recrute massivement... et du lourd ☺)


Par4All
Matériel évolue vite
Logiciel évolue plus lentement
Capitaliser sur *sources* applications
Déraisonnable de commencer nouvel outil
Création de l'initiative Par4All
Aide au développement d'outils existants
Industrialisation de codes académiques
Traçabilité & support
Rajout de nouvelles fonctionnalités
HPC Project


Par4All
Matériel évolue vite
Logiciel évolue plus lentement
Capitaliser sur *sources* applications
Déraisonnable de commencer nouvel outil
Création de l'initiative Par4All
Aide au développement d'outils existants
Industrialisation de codes académiques
Traçabilité & support
Rajout de nouvelles fonctionnalités
HPC Project


Par4All & PIPS
PIPS est un composant de Par4All
Extensions de PIPS (HPC Project représente majorité des développeurs dans PIPS)
Génération C & Fortran ~ CUDA C
Génération d'OpenMP avec réductions
Rajout du support Fortran 95
Génération de code pour diverses architectures (MPSoC, CEA SCMP, ST P2012, UtkDSP, Kalray...) [SCALOPES, SMECY...]
Parallélisation de Scilab [MediaGPU]
Décompilation (reconstruction binaire ~ C)
Instrumentation de code
Amélioration validation
HPC Project


Par4All dans OpenGPU
Parallélisation de C99 Fortran 95 vers OpenCL
Vectorisation pour vecteurs OpenCL
Intégration dans Eclipse
SPEAR-DE ~ OpenCL
De l'intérêt du source à source & libre
Même si l'outil disparaît, les sources produits restent
Base de départ pour commencer en programmation parallèle
Augmente dynamique des développeurs
HPC Project


Par4All dans OpenGPU
Parallélisation de C99 Fortran 95 vers OpenCL
Vectorisation pour vecteurs OpenCL
Intégration dans Eclipse
SPEAR-DE ~ OpenCL
De l'intérêt du source à source & libre
Même si l'outil disparaît, les sources produits restent
Base de départ pour commencer en programmation parallèle
Augmente dynamique des développeurs
HPC Project

3.2.4 Frédéric Magoulès, (ECP/CRSA)

MDM

Nouveaux algorithmes pour architectures hybrides.

Ecole Centrale Paris

Centrale Recherche S.A.

Frédéric Magoulès

URL: http://www.ecp.fr

Qui sommes nous…

Equipe de Recherche Calcul à Haute Performance

- *Recherche et Développement de nouvelles méthodes numériques*
- *Implémentation de ces algorithmes sur architectures parallèles (supercalculateur, réseaux de PC's, machines hybride)*
- *Diffusion auprès de l'industrie (PME, Grand Groupes, …)*

Un peu d'histoire…

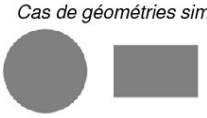
- *Une personne :*
  Karl Hermann Amandus Schwarz (1843 –1921)

- *Une méthode :*
  La méthode de de Schwarz additive
  $f_{xx}(x,y) + f_{yy}(x,y) = 0$
  $f(0,y) = 1$; $f(x,0) = f(x,1) = f(1,y) = 0$

- *Une publication:*
  H. A. Schwarz, *Über einen Grenzübergang durch alternierendes Verfahren*, Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15 (1870), pp. 272--286.

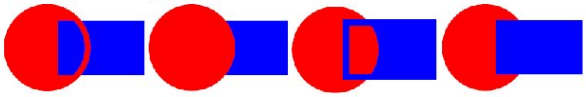
Comment ça fonctionne…

- *Solution analytique :*
  Cas de géométries simples

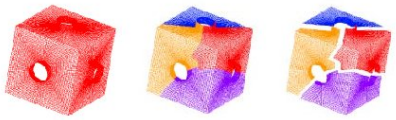


- *Solution analytique :*
  Cas de géométries complexes

- *Principe de l'algorithme de Schwarz*



Algorithme et parallélisme…



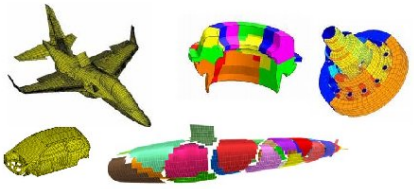
Méthodes de décomposition de domaines

- ❒ *Reformulation en 1989 par P.L. Lions, puis extension au cas sans recouvrement des sous-domaines*
- ❒ *Développement de nombreuses méthodes et techniques de préconditionnement (scalability, robustesse, …)*
- ❒ *Parallélisme intrinsèque: granularité, efficacité, …*

Applications industrielles…

Méthodes de décomposition de domaines

- ❒ *Implémentation sur supercalculateur, réseaux de PCs*
- ❒ *Applications à des problèmes des Sciences de l'Ingénieur*



Objectifs du projet…

Calcul Scientifique et Analyse Numérique

« Etudier et développer de nouveaux algorithmes parallèles pour machines hybrides, basés sur des approches multi-niveaux (méthodes de décomposition de domaines, …)»

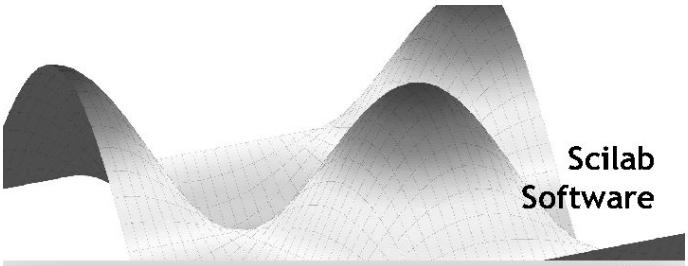
« Application et validation de ces nouvelles méthodes pour des problèmes de propagation d'ondes (ondes sismiques, tsunamis …) »

Merci pour votre attention

3.2.5 Sylvestre Ledru (Consortium Scilab/ Digiteo)

Scilab

Exploitation des GPU à partir de Scilab.



March 2010

Sylvestre Ledru
Scilab Consortium

The Scilab Consortium

History

- **Supported by INRIA**

The French National Institute for Research in Computer Science and Control

- **Hosted by the DIGITEO Foundation**

First network of excellence in information science and technology in France

The Consortium

Higher Education Schools

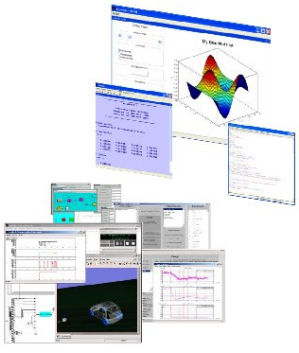
Public organizations

Companies

The Scilab Software

The Free solution

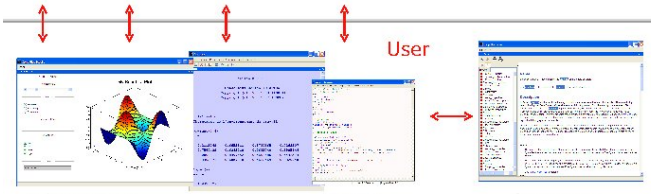
- **Powerful Computation Platform**
- **High-level, numerically** oriented programming language.
- Strong basis for **dedicated modules**
- **Hide the complexity** of native or/and complex libraries
- Works under Windows 2000/ XP/Vista/7, GNU/Linux, Unix and Mac OS X



What is Scilab?

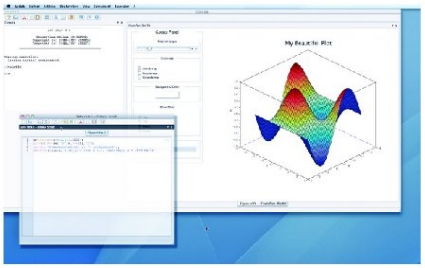
Computation library: more than 1700 functions

- Mathematical functions
- Matrix computation, sparse matrices
- Polynomials and rational functions
- Simulation: ODE and DAE
- Classic and robust control, LMI optimization
- Differentiable and non differentiable optimization
- Interpolation, approximation
- Signal processing
- Statistics
- Graphs and networks
- Xcos: hybrid dynamical systems modeler and simulator



Latest release

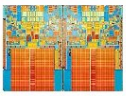
Scilab 5.2.1 *(February 2009)*



- Xcos 1.0
- Integrated editor
- LaTeX rendering
- Modules manager: ATOMS
- ...

Future

An Ambitious Roadmap



Covering Strategic Fields

→ From HPC to multicore

→ Code generation for Embedded Systems

- Mathematics and linear algebra
- Simulation
- Visualization
- Control
- Optimization
- Probabilities, statistics
- Signal processing

A new HPC-ready kernel

- Rewrite of the Scilab Kernel
- Developed with HPC features in mind (multithread, threadsafe...)
- Will allow simple and quick evolutions

Scilab in OpenGPU

Scilab in OpenGPU

- Two main goals:
  - Improves performances
  - Facilitate the usage of GPU features

Scilab in OpenGPU

- **First axis:**

  Enable access to CUDA / OpenCL functions from the Scilab language / interpreter.

  Allow the management of Scilab datatypes and simplifies access to the underlying technologies

Scilab in OpenGPU

- **Example:**

```
a = gpuarray.to_gpu(rand(100,100));

mod = SourceModule(' ..
_global__ void doublify(float *a) { ..
    int idx = threadIdx.x + threadIdx.y*4; a[idx] *= 2;  ..
}');

my_function = mod.get_function('doublify');

func(a, block=(4,4,1))
```

Scilab in OpenGPU

- **Second axis:**

  Update computation functions on the matrices to use the GPU

Example:

```
a = rand(1000, 1000); b = rand(1000, 1000);
enable_gpu();
c = a*b;
```

Scilab in OpenGPU

- **Third axis:**

Collaboration with CEA or INRIA to test the new GPU-oriented features in gcc.

3.2.6 Laurent Morin (CAPS Entreprise)

HMPP

Programmation parallèle multicore hétérogène : HMPP Workbench.

CAPS

hmpp openGPU

Heterogeneous Multicore Parallel Programming

Kick-off OpenGPU - S. Bihan & F. Bodin & L. Morin

HMPP & OpenGPU

- Main stream applications will rely on new multicore / manycore architectures
- Numerous legacy applications can benefit from GPU computing
- The OpenGPU project tends to build a European leadership on these new technologies
- Heterogeneous Multicore Parallel Programming (HMPP™) has been designed to exploit new architectures in legacy codes

OpenGPU Challenges for CAPS

Standard Platform
- Integration in Legacy Codes
- Programming Tools

High Performance
- Parallelization Tools
- Code Generation: OpenCL

Green Computing
- Measure Benefits on real applications

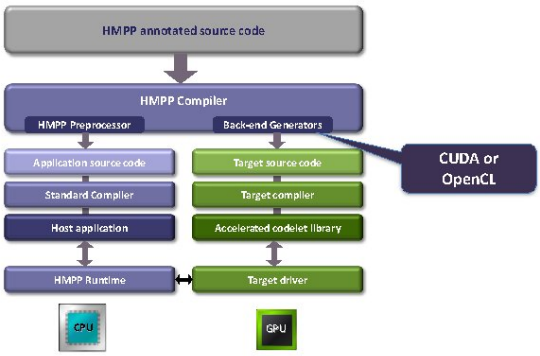
HMPP Overview

from the Application to the Device

What is HMPP?

- A directive based multi-language programming environment
  - Help keeping software independent from hardware targets
  - Provide an incremental tool to exploit GPU in legacy applications
  - Avoid exit cost
- HMPP provides
  - Code generators from C and Fortran to GPU (CUDA or OpenCL)
  - A compiler driver that handles all low level details of GPU compilers
  - A runtime to allocate and manage GPU resources
- Source to source compiler
  - CPU code does not require compiler change
  - Complement existing parallel APIs (OpenMP or MPI)

HMPP Compilation Flow



HMPP Main Design Considerations

- Focus on the main bottleneck
  - Communication between GPUs and CPUs
- **Allow incremental development**
  - Up to full access to the hardware features
- Work with other parallel APIs (e.g. OpenMP, MPI)
  - Orchestrate CPU and GPU computations
- Consider multiple languages
  - Avoid asking users to learn a new language
- Consider resource management
  - Generate robust software
- Exploit vendor tools/compilers
  - Do not replace, complement

Accelerate Codelet Function

- Declare GPU-accelerated versions of a function

```
#pragma hmpp sgemm codelet, target=OCL, args[vout].io=inout
extern void sgemm( int m, int n, int k, float alpha,
                   const float vin1[n][n], const float vin2[n][n],
                   float beta, float vout[n][n] )
{
  /* . . . */
}

int main(int argc, char **argv) {
    /* . . . */
 for( j = 0 ; j < 2 ; j++ ) {
#pragma hmpp sgemm callsite
    sgemm( size, size, size, alpha, vin1, vin2, beta, vout );
  }
    /* . . . */
 }
```

Declare OCL codelets

Synchronous codelet call

HMPP Tuning Directives

- To add code properties
  - Force loop parallelization
  - Indicate parameter aliasing
- To apply code transformations
  - Loop unrolling and jam, blocking, tiling, permute, …
- To control mapping of computations
  - Gridification
  - Place variables in shared or constant memory
  - Threads synchronization barriers

How Does HMPP Differ from CUDA or OpenCL?

- HMPP parallel programming model is **parallel loop centric**
- CUDA and OpenCL parallel programming models are **thread centric**

```
void saxpy(int n, float alpha,
           float *x, float *y){
#pragma hmppcg parallel
for(int i = 0; i<n; ++i)
    y[i] = alpha*x[i] + y[i];
}
```

```
__global__
void saxpy_cuda(int n, float alpha,
                float *x, float *y) {
int i =
blockIdx.x*blockDim.x + threadIdx.x;
 if(i<n) y[i] = alpha*x[i]+y[i];
}
/* . . . */
{
 int nblocks = (n + 255) / 256;
 saxpy_cuda<<<nblocks,256>>>
                (n, 2.0, x, y);
}
```


Tuning Directive Example

```
#pragma hmpp dgemm codelet, target=CUDA:CAL, args[C].io=inout
void dgemm( int n, double alpha, const double *A, const double *B,
            double beta, double *C ) {
int i;

#pragma hmppcg(CUDA) grid blocksize "64x1"
#pragma hmppcg(CUDA) permute j,i
#pragma hmppcg(CUDA) unroll(8), jam, split, noremainder
#pragma hmppcg parallel
 for( i = 0 ; i < n; i++ ) {
   int j;
#pragma hmppcg(CUDA) unroll(4), jam(i), noremainder
#pragma hmppcg parallel
   for( j = 0 ; j < n; j++ ) {
     int k; double prod = 0.0f;
     for( k = 0 ; k < n; k++ ) {
       prod += VA(k,i) * VB(j,k);
     }
     VC(j,i) = alpha * prod + beta * VC(j,i);
   }
 }
}
```

1D gridification Using 64 threads

Loop transformations

CAPS entreprise in OpenGPU

Contribution in OpenGPU Project

Main Contributions in Work Package 2: Parallelization Tools

- OpenCL Code Generation
  - Optimizations driven by High level Tuning Directives
- Integration in Eclipse

Collaboration with Work Package 4: Demonstrators

- Formations to GPU Computing
  - HMPP
  - CUDA, OpenCL
- Help Porting Applications
- HMPP used as a Gateway for High Level Parallelization tools

Conclusion

Conclusion

- Standardization effort is going on
  - OpenCL from Kronos
  - HMPP as an Open Standard
- High level GPU code generation allows many GPU code optimization opportunities
  - Easier to tune applications at high level
- HMPP helps in providing a standard GPU programming infrastructure
  - Seamless Integration in Applications
  - Gateway for OpenCL Code Generation

CAPS

Innovative Software for Manycore Paradigms

3.2.7 E. Chailloux (LIP6)

OCAML

Extension de OCaml pour le calcul scientifique sur GPU.

Extension d'Objective Caml pour le calcul scientifique sur GPU

Plan

- Présentation UPMC / LIP6 / PEQUAN - APR
- Objectif et choix
- Existant et expériences
- Passage sur GPU

Université Pierre et Marie Curie (Paris 6) en quelques chiffres

personnes

- président : Jean-Charles Pomerol
- 30 000 étudiants (10 000 en médecine, 20 000 en sciences)
- 4000 enseignants-chercheurs
- 3000 ingénieurs, techniciens et administratifs

diplômes par an

- 3000 diplômes de Master
- 200 diplômes d'ingénieurs
- 300 diplômes de doctorat de médecine
- 700 thèses scientifiques

Université Pierre et Marie Curie (Paris 6) en informatique

études

- passage au LMD en 2004/2005
- 900 étudiants en L1 : Mathémathiques-Informatique-Mécanique-Electronique
- environ 300-350 étudiants par année (L2, L3, M1, M2)
- 7 spécialités de master

recherche

- LIP6 : Laboratoire d'Informatique de Paris 6
- ISIR : Institut des Systèmes intelligents et de Robotique

Laboratoire d'Informatique de Paris 6 en quelques chiffres

personnes

- directeur : Patrick Gallinari
- 183 chercheurs permanents
- 263 doctorants

organisation et thématiques

- unité mixte CNRS : UMR 7606
- 5 départements, 18 équipes
  - Systèmes embarqués sur puce
  - Réseaux et Systèmes Répartis
  - Calcul Scientifique
  - Décision, Systèmes intelligents, Recherche opérationnelle
  - Données et Apprentissage Artificiel

LIP6 : Equipes participantes au projet

PEQUAN

- PErformances et QUalité des Algorithmes Numériques
- thématiques liées au projet
  - programmation parallèle
  - calcul scientifique

APR

- Algorithmes, Programmation et Résolution
- thématiques liées au projet
  - conception et implantation de langages typés statiquement
  - outils et environnements de programmation

Objectif et choix — calculs et compositions de calculs pour GPU

Objectif

- utilisation de langages de haut niveau
- pour le calcul scientifique
- dans le cadre des GPU

Choix

- langage Objective Caml pour la compositon de calcul
  - langage statiquement typé
  - multi-paradigmes : fonctionnel, impératif, modulaire, objet, générique
  - nombreuses bibliothèques de calcul et extensions parallèles
  - nombreuses applications

un mot sur Objective Caml — distribué par l'Inria

version 3.11 : caml.inria.fr

- langage fonctionnel,
- typé statiquement,
- polymorphe paramétrique,
- avec inférence de types,
- muni d'un mécanisme d'exceptions,
- et de traits impératifs
- possédant un système de modules paramétrés
- et un modèle objet
- exécutant des processus légers
- et communiquant sur le réseau Internet,
- indépendant de l'architecture machine.

Bibliothèques et Extensions — calcul et parallélisme

calcul

- grands tableaux
- interfaces pour gmp, fftw
- lacaml : interface BLAS, LAPACK (`http://hg.ocaml.info/release/lacaml`)

extensions parallèles

- jocaml (calculs concurrents et répartis)
- ocamlp3l (squelettes de parallélisme)
- caml-flight (data-parallélisme)
- bsml (Bulk synchronous ML).

Expérimentations — sur le processeur Cell

Oskell

- communications entre le monde Caml sur PPE et le code de calcul sur SPE
- implantation de squelettes de parallélisme
  - Map : PPE vers SPE
  - Pipe : SPE vers SPE et PPE
- performances dépendant du rapport mémoire transférée et calculs effectués

Projet (1)

passage sur les GPU

- 1ère étape : appel de fonction (BLAS - LAPACK) sur GPU à partir de Caml
- 2ème étape : composition de calculs, de fonctions de calcul
  - minimisation des transferts : composition sur le GPU mais de fonctions prédéfinies
  - idéalement génération de code openCL

Projet (2)

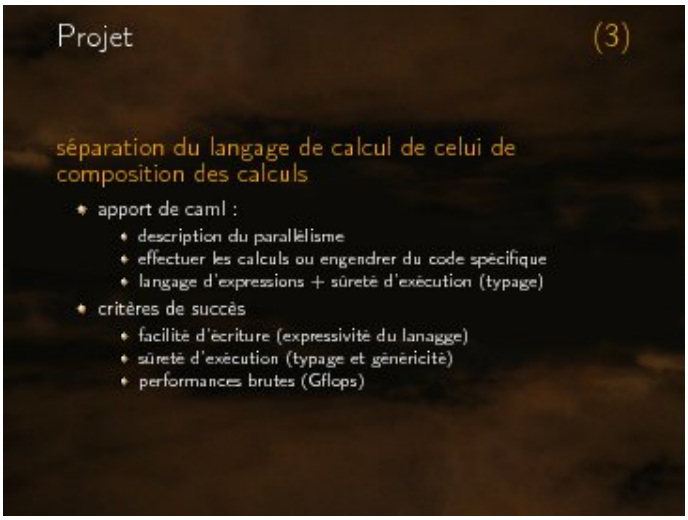
programme support : code 2DRMP

- bibliothèque CPC (Computer ...)
- calcul d'interactions en physique atomique
- programme PROP
- pour la mesure de performances

portage du code 2DRMP

- en FORTRAN/C
- en langage de haut niveau pour utiliser de manière transparente les accélateurs

application à d'autres applications


Projet (3)
séparation du langage de calcul de celui de composition des calculs
apport de caml :
description du parallélisme
effectuer les calculs ou engendrer du code spécifique
langage d'expressions + sûreté d'exécution (typage)
critères de succès
facilité d'écriture (expressivité du lanagge)
sûreté d'exécution (typage et généricité)
performances brutes (Gflops)


Des sites
UPMC : www.upmc.fr
Formation :
Licence http://www.licence.info.upmc.fr
Master http://www.master.info.upmc.fr
Doctorat http://www.edite-de-paris.com.fr
Recherche :
LIP6 www.lip6.fr
PEQUAN www-pequan.lip6.fr
APR www-apr.lip6.fr

3.3 Architecture matérielle et logicielle du projet OpenGCU

3.3.1 Jean François Lemerre (BULL)

BULL, CEA, GENCI Systèmes hybrides : une architecture pour le Green IT

SP3: Une architecture hybride pour le Green IT

bullx
instruments for innovation

Table des matières

- Introduction: Une démarche hybride entamée depuis longtemps
- Travaux prévus SP3: architecture matérielle
- Travaux prévus SP3: architecture logicielle et efficacité énergétique

Introduction: Une démarche GPU existant depuis longtemps

La famille bullx

Dédiée à l'Extreme Computing

bullx
instruments for innovation

Une offre Extreme Computing toujours innovante

Clusters hybrides (Xeon + GPUs)

Clusters intégrés

Systèmes haut de gamme pour clusters pétaflopiques

2008 2009 2010

Systèmes rack bullx

	R422 E2	R423 E2	R425 E2
			
NŒUD DE CALUCL	**2 nœuds dans 1U pour une densité inégalée** **NOUVEAU : plus de mémoire** ✓ Xeon 5500-5600 ✓ 2x 2 socket ✓ 2x 12 DIMMs ✓ QPI 6,4 GT/s ✓ 2x 1 PCI-Express x16 Gen2 ✓ InfiniBand DDR/QDR embarqué (option) ✓ 2x 2 disques SATA2 échangeables à chaud ✓ Efficacité énergétique 92%		
NŒUD DE SERVICE		**Connectivité et capacité de stockage renforcées** ✓ 2U ✓ Xeon 5500-5600 ✓ 2 socket ✓ 18 DIMMs ✓ 2 PCI-Express x16 Gen2 ✓ 6x disques SATA2 ou 8x SAS ✓ Alimentation redondante échangeable à chaud ✓ Ventilateurs échangeables à chaud	
NŒUD DE VISUALISATION			**Supporte les cartes graphiques et accélérateur les plus récentes** ✓ 4U ou tour ✓ 2 socket ✓ Xeon 55005600 ✓ 18 DIMMs ✓ 2 PCI-Express x16 Gen2 ✓ 6x disques SATA2 ou 8x SAS ✓ Alimentation puissante ✓ Ventilateurs échangeables à chaud

Accélérateurs GPU pour systèmes rack bullx

Systèmes NVIDIA® Tesla™ : des processeurs multi cœur téraflopiques qui fournissent une puissance de calcul parallèle exceptionnelle, avec un rendement énergétique inégalé

NVIDIA Tesla C1060

Transforme un serveur R425 E2 en superordinateur
- ✓ Carte double largeur
- ✓ Processeur Tesla T10P
- ✓ 240 cœurs
- ✓ Performance: près de 1 Tflops (32 bit FP)
- ✓ Connexion sur PCIe x16 Gen2

NVIDIA Tesla S1070

Un accélérateur idéalement adapté aux clusters à base de serveurs R422 E2
- ✓ Tiroir 1U
- ✓ 4 processeurs Tesla T10P
- ✓ 960 cœurs
- ✓ Performance: 4 Tflops (32 bits FP)
- ✓ Connexion sur 2 PCIe x16 Gen2

GENCI - CEA

Une architecture hybride répondant à des besoins de production et de recherche, avec un grand cluster associant serveurs génériques & serveurs spécialisés
- 1068 nœuds Bull, soit 8544 cœurs Intel® Xeon® 5500 de nouvelle génération (puissance crête 103 Tflops)
- 48 nœuds GPU NVIDIA®, soit 46000 cœurs, (puissance théorique supplémentaire 192 Tflops)
- 25 To de mémoire
- Réseau d'interconnexion InfiniBand
- Environnement logiciel Bull intégré à base de composants Open Source
- Système de fichier partagé Lustre®
- Densité optimale grâce au refroidissement par eau

295 Tflops crête : 1er grand système hybride européen

Mise à disposition d'une partie du cluster hybride GENCI pour les partenaires

Le système lame bullx

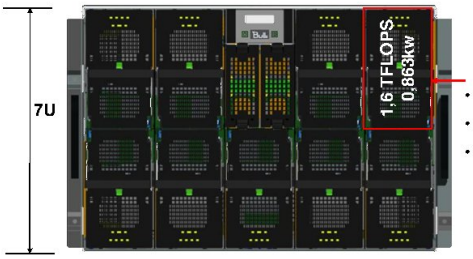
Conçu pour l'Extreme Computing

HPC Editors' Choice Awards 2009 — Meilleur produit ou technologie HPC

HPC Readers' Choice Awards 2009 — Top 5 produits ou technologies à suivre

Lame accélérateur bullx B505

Accélérateur intégré pour des performances élevées avec une faible consommation

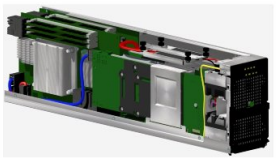


- 2 x Intel Xeon 5500
- 2 x NVIDIA T10
- 2 x IB QDR

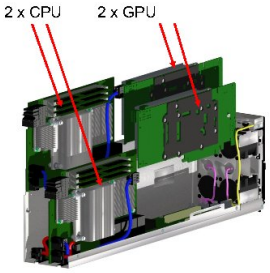
18,9 TFLOPS dans 7 U

Lame accélérateur bullx B505

- Lame double largeur
- 2 GPU NVIDIA Tesla M1060
- 2 CPU Intel® Xeon® 5500 quadri cœur
- 1 connexion PCI-e 16x dédiée pour chaque GPU
- Double connexion InfiniBand QDR entre lames

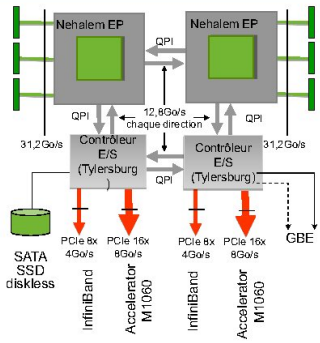


Vue de face



Vue arrière

Lame accélérateur B505



Travaux prévus dans le SP3: Matériel

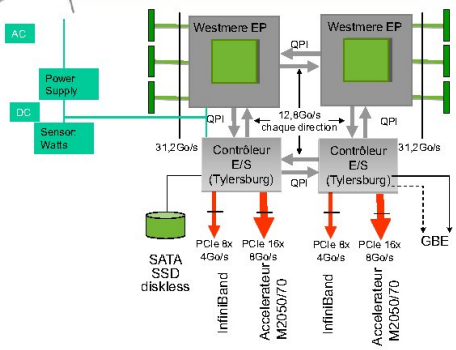
Travaux d'évaluation

- Comparaison de différentes alternatives:
  - Etude ratio puissance processeur /puissance GPU
  - Débit CPU-GPU
  - Debit IB versus débit CPU/GPU
  - Consommation électrique
- Evaluation solutions alternatives
  - Fermi
  - ATI
  - Autres solutions accélératrices?

Travaux de définition et de développement

- Définition fonctionnelle d'un cluster hybride avec CEA
  - Châssis hybride?
  - Gflops/watt
  - Gflops/m²
  - Gflops/€
  - Comportemant grosses applications hybrides
- Prototype Blade Fermi + Westmere
  - Développement
  - Mise au point
  - Expérimentations

Nouvelle lame accélérateur Westmere +Fermi



Mise en œuvre prototype

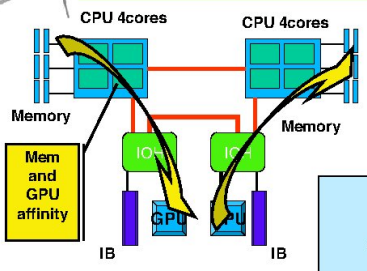
- Fourniture d'un cluster hybride au CEA
  - A priori Westmere + Fermi
- Mise en œuvre de ce cluster dans un environnement similaire à de la production:
  - Etude stabilité système
  - Consommation sur des codes représentatifs
  - Comparaison avec cluster existant (R422 +Tesla)
  - Mise à disposition des partenaires

Travaux prévus dans le SP3: Architecture logicielle et efficacité energétique

Architecture logicielle

- Distributions Linux adaptées au cluster hybride
  - Drivers adaptés
  - SDK
  - Incluant partie des travaux de SP2
- Expérimentation d'améliorations fonctionnelles

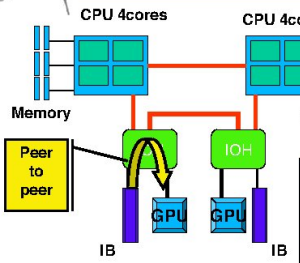
Améliorations logicielles: GPUset



GPUset

Assurer une optimisation permettant d'associer CPU, mémoire et GPU de même localité géographique

Améliorations logicielles: Peer to peer



Peer to peer:

Permet un DMA direct
Entre IB et GPU
L'espace d'adresse GPU est visible de l'IB et IB peut effectuer des transferts sans passer par la mémoire

Améliorations logicielles: partage des GPUs

Partage des GPUS par les coeurs

Besoin:

Quand le système a moins de GPUs que de process en ayant besoin

A traiter:

Applications ne sont pas liées à un numéro de GPU (notion de GPU virtuel)

Eviter la contention PCI (ne pas dupliquer les accès GPU)

Améliorations des informations sur les GPU utilisés (mémoire disponible, coeurs GPU libres,...)

Bibliothèque d'allocation GPU? (get device)

Améliorations logicielles: programmation hybride MPI GPU

Programmation hybride MPI GPU

Description du Problème:

MPI utilise des mécanismes de locks en mémoire avant d'envoyer ou recevoir des buffers

Le Soft GPU utilise d'autres locks pour envoyer les buffers entre CPU et GPU

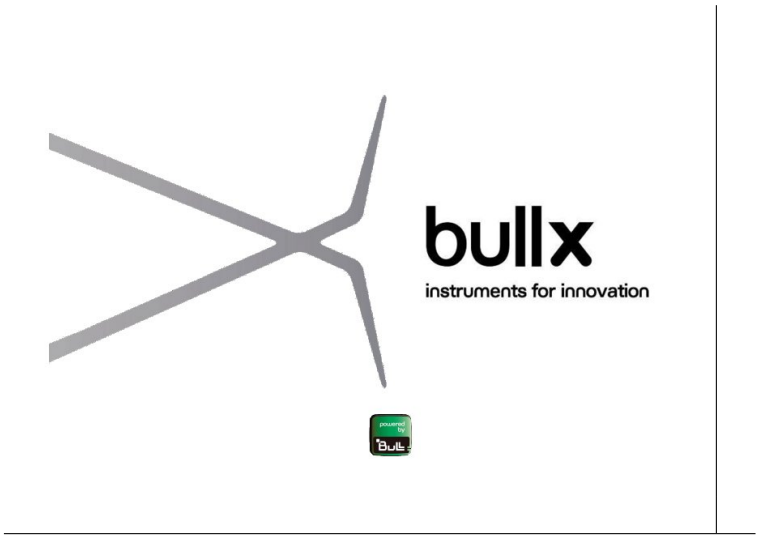
=>Très gros overhead de lock/unlock

A traiter:

Un système de locks coherents entre MPI et GPU

Efficacité énergétique

- But: réduire la consommation énergétique globale
  - Remplacer des processeurs généralistes par des GPUs: plus puissants en flops
  - Répartition des consommations différentes
  - GigaFlops /Watt
- Sur applications partenaires:
  - Mesure de la consommation énergétique sur des processeurs généralistes → Jobs/Watt
  - Mesure consommation sur premier portage GPU
  - Mesure consommation après optimisation


bullx
instruments for innovation

3.4 Démonstrateurs industriels et académiques

3.4.1 J. Bost (NUMTECH)

L'avenir du risque environnemental : le GP-GPU ?

NUMTECH
Les modèles
GPGPU
OpenGPU

L'avenir du risque environnemental : le GPGPU ?

Kick off OpenGPU
25 Mars 2010

Bost Julien
NUMTECH, Ingénieur informatique.

NUMTECH OpenGPU aristote

© NUMTECH – Octobre 2009

NUMTECH
Les modèles
GPGPU
OpenGPU

Création en avril **2000** par deux scientifiques,
P. BEAL et E. BUISSON

Siège social à **Clermont-Ferrand**
Établissement secondaire à **Bruyères-Le-Châtel** (91)

Filiales :
- CEIES (santé publique, 63/38)
- AIRbe (GES – ACV, 92),
- SILLAGES Environnement (mécanique des fluides, 69),
- Weather Measures (Mesures météo, dynamique atmosphérique, 63).

Équipe de 26 personnes :
- 12 Docteurs, 3 DEA, 4 ingénieurs.

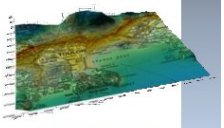
© NUMTECH – Octobre 2009

NUMTECH
Les modèles
GPGPU
OpenGPU

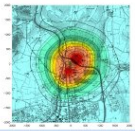
Domaine d'activités :
Le risque environnemental.

Expertise :
Les sciences de l'atmosphère.

Objectif :
L'aide à la décision.



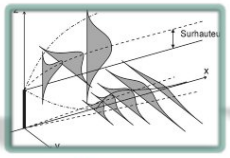
- Modélisation de la dispersion atmosphérique
- Études de risque : industrie / transports / énergie
- Prévisions & fournitures données météorologiques
- Expertise de la qualité de l'air (urbain, régional)
- Évaluation des risques sanitaires



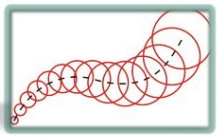
© NUMTECH – Octobre 2009

NUMTECH
Les modèles
GPGPU
OpenGPU

Les principaux types de modèles

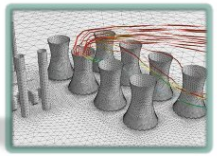


Gaussien



Bouffées gaussiennes

Lagrangien

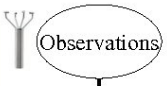


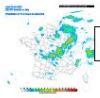
Eulérien

© NUMTECH – Octobre 2009

NUMTECH
Les modèles
GPGPU
OpenGPU

Les principaux types de modèles





Modèles météos

Réduction d'échelle,
Reconstruction de champs de vent
Turbulence.

Modèles de
dispersion atmosphérique

© NUMTECH – Octobre 2009

NUMTECH
Les modèles
GPGPU
OpenGPU

Le GPGPU : Les performances

Les objectifs

- Réduction des coûts
  - Green computing
  - Planning réduit pour les projets
- Gain de précision
  - Résolution plus fine
  - Modèle plus complexe
- Application temps réel (gestion de crise)

© NUMTECH – Octobre 2009

NUMTECH
Les modèles
GPGPU
OpenGPU

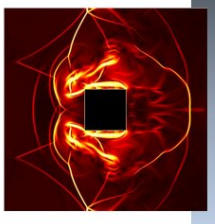
Nos Premiers tests

Objectifs

Valider l'utilisation du GPU

Tester les performances

Acquérir les compétences

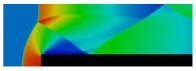


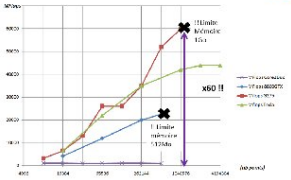
- *Application from scratch*
- *Cuda*
- *Résolution des équations d'Euler pour les fluides compressibles.*

NUMTECH
© NUMTECH – Octobre 2008

NUMTECH
Les modèles
GPGPU
OpenGPU

Les résultats





Il faut donner à manger à la carte graphique !

Les concepts de programmation sont différents.

Il faut parfois changer les algorithmes / les méthodes de calcul.

Tous les outils classiques n'existent pas encore ou ne sont pas encore au point.

La taille de la communauté : manque d'information / méthodologie.

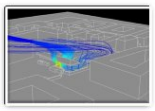
NUMTECH
© NUMTECH – Octobre 2008

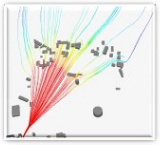
NUMTECH
Les modèles
GPGPU
OpenGPU

Aller plus loin avec le projet OpenGPU

La performance

- Accélération de l'ensemble de nos modèles
- Généraliser l'utilisation des modèles lagrangiens
- Échelle plus fine
- Application temps réel pour la gestion de crise sur des architectures portables





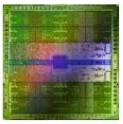
NUMTECH
© NUMTECH – Octobre 2008

NUMTECH
Les modèles
GPGPU
OpenGPU

Aller plus loin avec le projet OpenGPU

La pérennité des développements

- Cuda / OpenCL
- L'avenir du GPGPU
- Minimiser l'impact sur les codes existants



NUMTECH
© NUMTECH – Octobre 2008

NUMTECH
Les modèles
GPGPU
OpenGPU

Aller plus loin avec le projet OpenGPU

Développement, mise au point

- Analyse de code, aide à la parallélisation
- Portage automatique / semi-automatique
- Débuggeur, profileur, IDE
- Expérience

```
#pragma omp parallel shared(x, y, z) num_threads(2)
{
  int tid = omp_get_thread_num();
  if (tid == 0) {
    y = fn70(tid);
  }
  else {
    z = fn80(tid);
  }
#pragma omp barrier
#pragma omp for
  for (k = 0; k < 100; k++ ) {
    x[k] = y + z + fn10(k) + fn20(k);
  }
}
```

NUMTECH
© NUMTECH – Octobre 2008

NUMTECH
Les modèles
GPGPU
OpenGPU

Merci de votre attention.

NUMTECH
OpenGPU
aristote

NUMTECH
© NUMTECH – Octobre 2008

3.4.2 Guillaume Colin de Verdière (CEA DAM)

Retour d'expérience du calcul hybride

(cf. les transparents de la matinée)

3.4.3 Jean-Michel Batto (INRA)

Génomique et GPU

Génomique et GPU

Jean-Michel Batto
jean-michel.batto@jouy.inra.fr

INRA, Institut MICALIS
département **MICA** (**MI**crobiologie de la **C**haîne **A**limentaire)
Centre de Recherche de Jouy-en-Josas (78)

Kickoff OpenGPU / 25 mars 2010 / Aristote-Ecole Polytechnique (91)

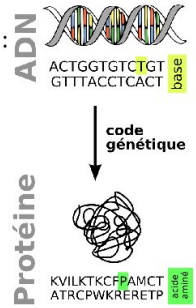
INRA

Génomique : une donnée simple

- L'ADN : une chaîne de texte dans un alphabet de 4 lettres

Génomique : une donnée encore simple

- Les objets d'intérêt :
  - ADN
  - Traduction
  - ARN
  - Structure



La diversité des structures de données

- Les ordres de grandeur
  - Gène->1..n protéines : 100 - 10000 lettres
  - ARN messager : ~100 - 10000 lettres
  - ARN antisens : ~100 lettres
  - Opéron : ~ 3 gènes
  - Plasmide/Virus/Phage : ~100 gènes
  - Un génome bactérien : ~2000 gènes
  - Un génome humain : ~30000 gènes

La génomique 'haut-débit'

- Des nouveaux séquenceurs haut-débits (2008)
  - 454 - Roche

  - Solexa - Illumina
  - Solid - Applied Biosystems

600 Mb / jour
50 X ce que produit un séquenceur de 2002 (3730XL AB)

Une application : Le métagénome humain

- Un nouveau champ d'investigation:
  - Dans le tractus digestif, il y a 2 à 4 kg de bactéries
  - Il y a plus de 1000 espèces présentes
  - L'intestin est un organe clé dans la réponse immunitaire

Projet MetaHIT : www.metahit.eu
Coordinateur : S. D. Ehrlich

MetaHIT

ARTICLES

A human gut microbial gene catalogue established by metagenomic sequencing

Junjie Qin[1*], Ruiqiang Li[1*], Jeroen Raes[2,3], Manimozhiyan Arumugam[2], Kristoffer Solvsten Burgdorf[4], Chaysavanh Manichanh[5], Trine Nielsen[6], Nicolas Pons[7], Florence Levenez[7], Takuji Yamada[2], Daniel R. Mende[2], Junhua Li[1,8], Junming Xu[1], Shaochuan Li[1], Dongfang Li[1,9], Jianjun Cao[1], Bo Wang[1], Huiqing Liang[1], Huisong Zheng[1], Yinlong Xie[1,9], Julien Tap[5], Patricia Lepage[5], Marcelo Bertalan[2], Jean-Michel Batto[5], Torben Hansen[6], Denis Le Paslier[10], Allan Linneberg[11], H. Bjørn Nielsen[12], Eric Pelletier[10], Pierre Renault[5], Thomas Sicheritz-Ponten[13], Keith Turner[14], Hongmei Zhu[1], Chang Yu[1], Shengting Li[1], Min Jian[1], Yan Zhou[1], Yingrui Li[1], Xiuqing Zhang[1], Songgang Li[1], Nan Qin[1], Huanming Yang[1], Jian Wang[1], Søren Brunak[4], Joel Doré[5], Francisco Guarner[15], Karsten Kristiansen[16], Oluf Pedersen[16,17], Julian Parkhill[18], Jean Weissenbach[10], MetaHIT Consortium†, Peer Bork[2], S. Dusko Ehrlich[5] & Jun Wang[1,9]

- Consortium **MetaHIT** (**Meta**genomics of the **H**uman **I**ntestinal **T**ract) : UE-FP6 (Espagne, Pays-Bas, Belgique, Danemark, France, Italie)
- Autres contributeurs : BGI-Shenzhen (Chine), EMBL (Allemagne), Wellcome Trust Sanger Institute (GB), ...

Est-ce prévisible?

- Octobre 2008 : Un déluge d'informations

EDITOF

nature biotechnology

iotechnology

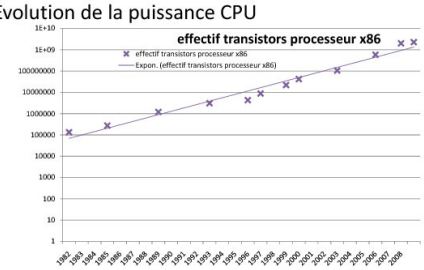
Prepare for the deluge

The gobs of data produced by next-generation sequencing are a key problem limiting wider adoption.

8/16

La loi de Moore et la génomique

- Evolution de la puissance CPU

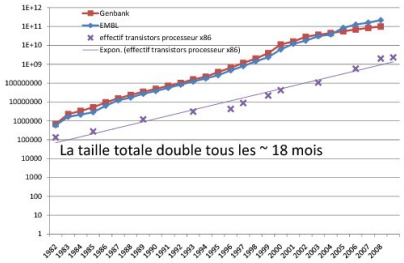


  - L'effectif du nombre de transistors du x86 le plus puissant, double tous les 18 mois
    - http://en.wikipedia.org/wiki/Transistor_count

9/16

La loi de Moore et la génomique

- Evolution de la taille des banques de séquence

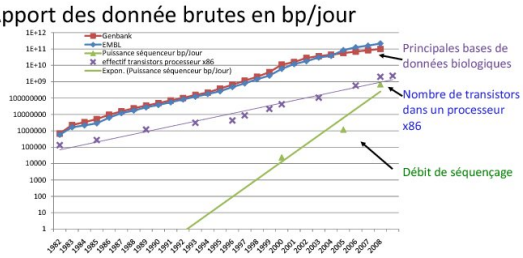


Sources : EMBL et NCBI

10/16

La loi de Moore et la génomique

- Apport des donnée brutes en bp/jour



Les séquenceurs haut-débits-> technologie nouvelle
De l'ordre de la loi de Moore.

11/16

Survol de l'utilisation du GPU en bioinformatique

- 2005 : RAxML / Phylogeny / BrookGPU
  gain gpu/cpu : 2,3x
- 2006 : GPU-ClustalW / mult. alignement / OpenGL
  gain gpu/cpu : 11,2x
- 2007 : MUMmerGPU / mult. alignement / CUDA
  gain gpu/cpu : 3,8x
- 2008 : Smith-Waterman / algo alignement / CUDA
  gain gpu/cpu : 2,4x
- 2009 : GPU-HMMER / HMM / CUDA
  gain gpu/cpu : 30x

12/16

Dans notre labo : GPUArrayScan-2009

[Fouad Boumezbeur, Nicolas Pons, Pierre Renault, S. Dusko Ehrlich and Jean-Michel Batto
MonBUG Symposium 2009, Montreal (Canada)]

Comptage d'occurences de gènes de référence dans un jeu de données de séquences courtes sur carte Tesla C870 – GPU non optimisé.

Machine de test x86-Tesla:
Xeon E5550 @ 2.84 GHz 8 Go RAM 12Mo cache
+ *nVidia* CUDA Tesla™ C870 1.5Go RAM --> ~3k$
Machine de référence (SOLID workstation) :
2x Xeon L5410@ 2.3 GHz 32 Go RAM ECC 12Mo cache --> ~9k$

CPU/GPU gain de la cible du traitement	GPUarrayScan/ CORONA Lite v.4 r2 gain	Bénéfice en coût
20x Non significatif	Pas de gain en perf (GPU non optimisé)	3x

→ Au-delà du gain en vitesse, il y a le bénéfice du coût!

13/16

14/16

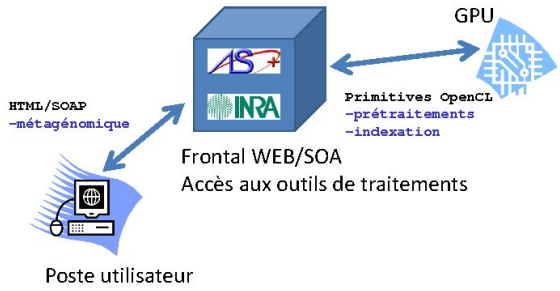
Du point de vue de l'utilisateur

- Accès en ligne de commande
  Après compilation, sur une machine linux
- Accès via une API Python
  http://www.biomanycores.org/

Comment concilier la haute spécificité du GPU et l'utilisateur?
->architecture 3-tiers

15/16

Le projet OpenGPU (FUI) 2010-2012
SP4 / démonstrateurs

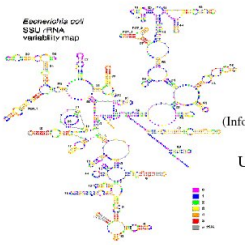


16/16

3.4.4 Fariza Tahi (Université d'Evry)

Recherche à grande échelle d'ARNs non-codants

La recherche à grande échelle d'ARNs non-codants



Fariza Tahi

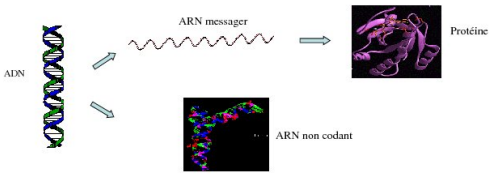
Laboratoire IBISC
(Informatique, Biologie Intégrative et Systèmes Complexes)

Université d'Evry-Val d'Essonne/Genopole

La fonction biologique

La fonction biologique s'exprime par deux voies : protéines et ARN

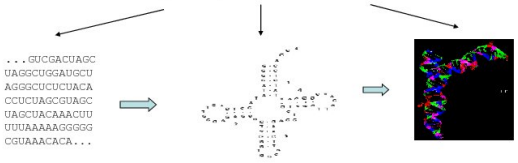
Les ARN et les protéines possèdent la variabilité structurale nécessaire à l'apparition d'une fonction



Structure ➡ Fonction

L'ARN

Trois structures : primaire, secondaire et tertiaire



Prédire la structure d'un ARN

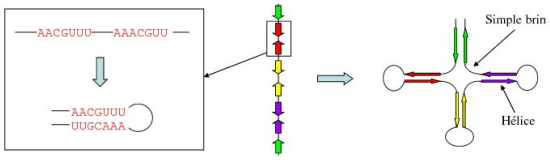
➡ Trouver les repliements que subit la structure primaire pour former la structure secondaire puis tertiaire.

Méthodes expérimentales (RMN et cristallographie) coûteuses en temps et argent

➡ Importance de prédiction ou modélisation « *in silico* »

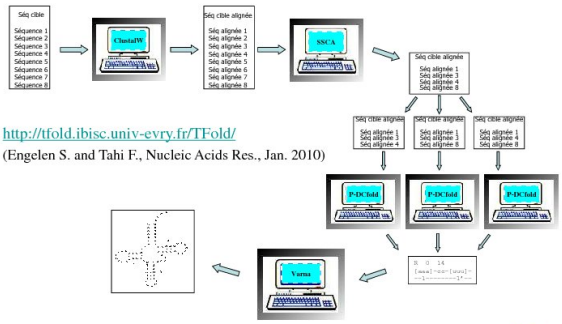
Structure secondaire d'ARN

- Structure secondaire : ensemble d'hélices résultant de l'appariement d'une succession de paires de bases complémentaires (A – U, G – C et G – U)



Le but de la prédiction de structure secondaire est de trouver ces appariements (hélices)

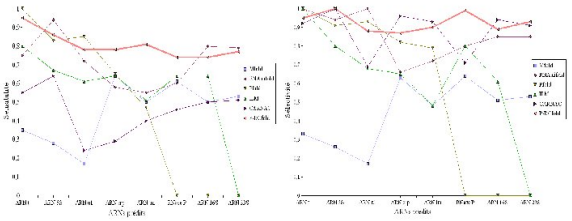
Logiciel Tfold



http://tfold.ibisc.univ-evry.fr/TFold/

(Engelen S. and Tahi F., Nucleic Acids Res., Jan. 2010)

Logiciel Tfold



- Séquences d'ARN de tailles variables (76 à 2904 pb)
- Sensibilité moyenne de 0,85 et sélectivité moyenne de 0,95
- Homogénéité de la qualité des résultats
- Complexité en temps : $O(n^2)$
- Temps de calcul : quelques secondes pour une prédiction (P-DCFold)

Recherche d'ARN dans les séquences génomiques

- L'un des challenges actuels en bioinformatique est l'analyse à grande échelle de séquences génomiques.
- Annoter ou analyser une séquence génomique : rechercher des gènes, des motifs particuliers, des répétitions, … et des ARN.
- Les ARN peuvent être trouvés grâce à leur structure secondaire.
- Les nouvelles technologies de séquençage permettent d'obtenir en des temps records des séquences génomiques de tailles importantes (tailles pouvant atteindre plusieurs millions de pb.).
- Nécessité de disposer de méthodes et d'outils efficaces et rapides pour l'annotation des séquences génomiques, y compris la recherche d'ARN.
- Nécessité de paralléliser et d'utiliser des HPC.

Kick-off OpenGPU 25 mars 2010 Fariza Tahi ibisc

OpenGPU et la recherche d'ARN

- Tfold : logiciel efficace et rapide pour prédire la structure secondaire d'ARN
- Tfold peut être adapté pour la recherche d'ARN dans les séquences génomiques
- Mais les temps de calcul sont très élevés sur des séquences de taille importante (complexité de $O(n^2)$)
- Pour cela :
  - Nécessité de paralléliser efficacement l'algorithme et d'utiliser des HPC

⇩

Projet OpenGPU : opportunité de développer et de proposer à la communauté scientifique (académique, biotechnologies, compagnies pharmaceutiques, ..) un logiciel efficace et rapide de recherche d'ARN dans les génomes

Kick-off OpenGPU 25 mars 2010 Fariza Tahi ibisc

3.4.5 Antoine Petitet (ESI Group)

Utilisation de Co-processeurs Graphiques dans des Codes Industriels de Simulation Numérique

get it right™

OpenGPU

Utilisation de Co-processeurs Graphiques dans les Codes Industriels de Simulation Numérique

Mars 2010

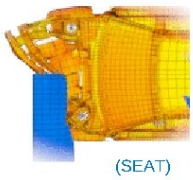
www.esi-group.com

Copyright © ESI Group, 2009. All rights reserved. 1

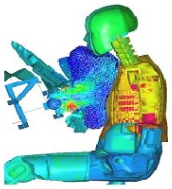
get it right™

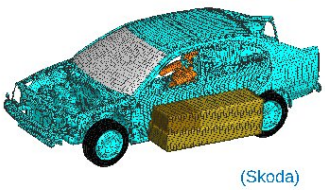
PAM-CRASH : Collision de Véhicules

- Impact frontal
- Impact latéral
- Déploiement d'airbag



(SEAT)





(Skoda)

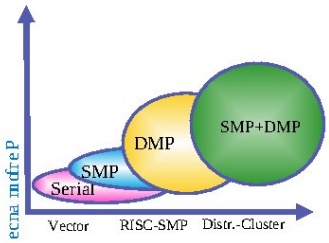
www.esi-group.com

Copyright © ESI Group, 2009. All rights reserved. 2

get it right™

Evolution du Logiciel avec l'Architecture des Calculateurs

- Vectoriel (Cray X-MP)
- SMP (SGI Origin)
- DMP (IBM SP)
- SMP + DMP (Multi-cœurs, POPS)



- CPU(s) + GPU(s) ?

www.esi-group.com

Copyright © ESI Group, 2009. All rights reserved.

get it right™

Objectifs d'ESI dans le cadre d'OpenGPU

- Evaluation du gain de performance résultant de l'utilisation de librairies spécialisées e.g. CUBLAS.
- A l'aide des outils développés dans le cadre du projet, étude et optimisation de portions de code appropriées.
- A terme, fournir à nos clients une version de nos produits efficace sur calculateurs hybrides CPU + GPU, et donc conserver une place de leader technologique dans le domaine de la simulation numérique en exploitant les architectures matérielles les plus modernes.

www.esi-group.com

Copyright © ESI Group, 2009. All rights reserved.

get it right™

www.esi-group.com

3.4.6 Michel Barreteau (Thales)

Calcul Temps-Réel embarqué sur GPU

THALES

Thales Research & Technology

Kick-off OpenGPU – SP4 (Michel BARRETEAU)

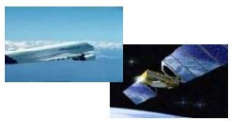
Démonstrateur : secteur « Embarqué »

Research & Technology

Activités Thales

Trois marchés principaux

- Aéronautique et Espace
- Défense
- Sécurité



2

THALES

Calcul embarqué haute performance

Radars

Sonars

Télécommunications

Traitement d'image

Analyse de l'information

Aide à la décision

3

THALES

Expertise

Domaines

- Traitement du signal
- Traitement d'image

Architectures de processeurs

- Ter@ops
- Cell
- Ambric
- Tilera

Méthodes et intégration d'outils liés à la parallélisation

- SPEAR Design Environment

4

THALES

Attentes

Contexte général

- Exploitation de solutions standard du commerce (architectures et/ou programmation) pour les besoins spécifiques de l'embarqué (pérennité, efficacité Flops/Watt)

Objectifs principaux

- Evaluation d'OpenCL (compilation, couplage à des outils de parallélisation de plus haut niveau) à travers un flot de conception facilitant la programmabilité
- Évaluation de performances sur GPU en vue de solutions plus « ouvertes »

Radars

Sonars

Télécommunications

Traitement d'image

Analyse de l'information

Aide à la décision

Formation de faisceaux adaptative

ou

Flot optique (analyse de scènes depuis un véhicule en mouvement)

5

THALES

3.4.7 Jean-Noël de Galzain (Wallix)

GPU et sécurité

ADMINBASTION

Un démonstrateur Sécurité
KICK-OFF OPENGPU

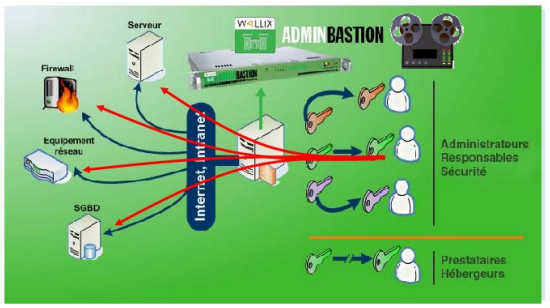
JNdeGALZAIN – jndegalzain@wallix.com

25/03/2010

WALLIX Infrastructure and Security Solutions

WALLIX Infrastructure and Security Solutions

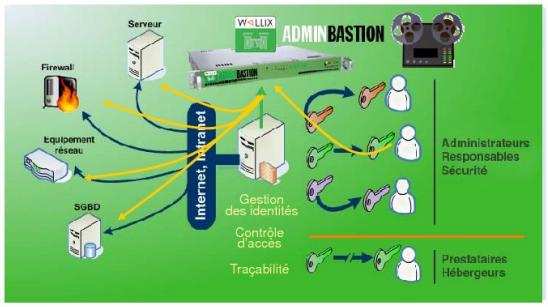
WAB
WALLIX ADMIN BASTION



WALLIX Infrastructure and Security Solutions

WALLIX Infrastructure and Security Solutions

WAB
WALLIX ADMIN BASTION



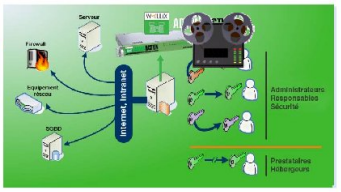
WALLIX Infrastructure and Security Solutions

WALLIX Infrastructure and Security Solutions

WAB
WALLIX ADMIN BASTION

FONCTIONNALITES

- Authentification forte des intervenants sur le réseau (Single Sign-On)
- Gestion centralisée des mots de passe et des clés serveurs
- Traçabilité des accès et des opérations menées sur les serveurs et le réseau



BENEFICES CLIENTS

- Sécurité renforcée : définition de la politique de sécurité
- Mise en conformité: lois / règlementations, normes / standards, référentiels d'entreprise
- Gestion des accès des prestataires externes au SI
- Identification & recherche de responsabilité en cas de panne
- Durcissement et simplification de la gestion des mots de passe

WALLIX Infrastructure and Security Solutions

ADMINBASTION

CONSTAT

- WAB = serveur mandataire protocolaire sur les protocoles chiffrés RDP et SSH
  - Gourmands en crypto asymétrique à clé publique (RSA)
  - Signature numérique (DSA)
  - Echange de clé (D-H)
  - Chiffrement (Cipher CBC)

=> Goulet d'étranglement lié au calcul cryptographique : T = V x BP

Contenu confidentiel et propriété de Wallix

WALLIX Infrastructure and Security Solutions

ADMINBASTION

DEMONSTRATEUR SECURITE

- CRYPTOPROCESSEURS
  - Architecture à processeur spécialisé : résolution/calcul des algorithmes de chiffrement/déchiffrement
  - Optimiser la consommation des protocoles rdp ou ssh pour
    - Le chiffrement / déchiffrement
    - La Signature numérique (DSA)
    - L'échange de clef
  - Optimiser l'équation actuelle T = V * BP sur une architecture hybride

- METHODOLOGIE
  - Cryptanalyse / Tests de performance et de comportement
  - Optimisation sur architecture hybride
  - Implémentation de type ASIC

Contenu confidentiel et propriété de Wallix

WALLIX Infrastructure and Security Solutions


ADMINBASTION
ROADMAP
ENREGISTREMENT VIDEO
Implémentation d'algorithmes de détection de motifs
Module d'encodage / décodage d'enregistrements vidéo
+ TARD
Crypto : génération de Rainbow table (AES 256)
Vidéo : vidéo surveillance du SI
Passage au Temps réel
Contenu confidentiel et propriété de Wallix
7
WALLIX

ARISTOTE Association Loi de 1901. Siège social : CEA-DSI CEN Saclay Bât. 474, 91191 Gif-sur-Yvette Cedex.
Secrétariat : Aristote, École Polytechnique, 91128 Palaiseau Cedex.
Tél. : +33(0)1 69 33 99 66 Fax : +33(0)1 69 33 99 67 Courriel : Marie.Tetard@polytechnique.edu
Site internet http://www.aristote.asso.fr