GPU & Open Source

Ronan KERYELL

rk@hpc-project.com

HPC Project
http://www.hpc-project.com

—

9 Route du Colonel Marcel Moraine
92360 Meudon La Forêt

—

Rond Point Benjamin Franklin
34000 Montpellier

Associated researcher at Institut TÉLÉCOM, TÉLÉCOM Bretagne/HPCAS, Plouzané
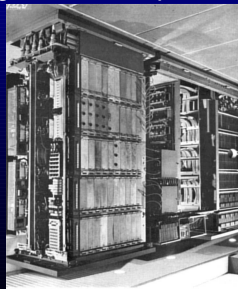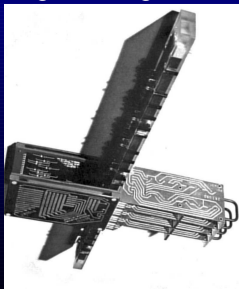
1/7/2009
Forum Ter@tec 2009

Multicores strike back... *(I)*

Gamma 60 from Compagnie des Machines Bull

- Increase the performance
- 100 kHz memory clock
- *Heterogeneous* multicore because the memory is too... fast! ☺
- 24-bit words, 96 KiB of core memory
- Punch-cards with ECC, magnetic tapes, magnetic drums
- Highly integrated logic in 1-mm germanium bipolar lithography ☺

Multicores strike back... *(II)*

- Gamma 60 multithread programming with `SIMU` ($\approx$ fork) & `CUT` (launch a program on a functional unit) instructions
- Synchronization barrier by concurrent branching on a same target
- Scheduling of threads based on a queue per functional unit stored just... inside the code after each `CUT` instruction!
- Optional hardware critical section on subprograms (cf. `synchronized` of Java)

Multicores strike back... *(III)*

- Installation around 1959
- Already hard to program since the concepts was not here, at most the (grand-)parents of anyone who were to know them...
- ⤳ For celebrating the 50th birthday of the Gamma 60: conference at RenPar/SympA/CFSE 9-11/9/2009! ☺

`http://www.irit.fr/Toulouse2009/`

`http://www.feb-patrimoine.com/projet/gamma60/gamma_60.htm`

GPGPUs: just more integrated... *(I)*

- The "Distributed Computer"
  - ▶ Toward computing on spatial data : pattern recognition, mathematical morphology...
 - ▶ Massive parallelism to reduce the cost
  - ▶ SIMD

  S. H. Unger. « A Computer Oriented Toward Spatial Problems. » *Proceedings of the IRE*. p. 1744–1750. oct.

1958

GPGPUs: just more integrated... *(II)*
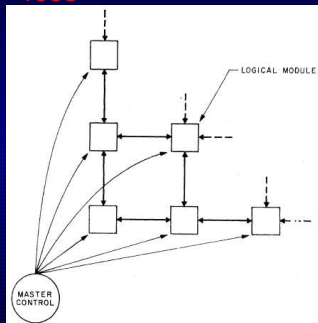
- SOLOMON

  ▶ Target application: "data reduction, communication, character recognition, optimization, guidance and control, orbit calculations, hydrodynamics, heat flow, diffusion, radar data processing, and numerical weather forecasting"

  ▶ Diode + transistor logic in 10-pin TO5 package
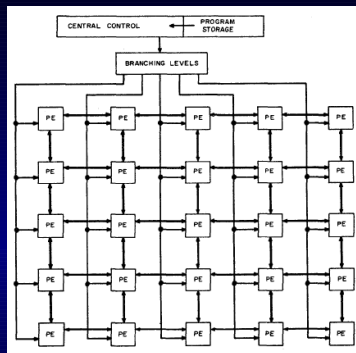
Daniel L. Slotnick. « The SOLOMON computer. » *Proceedings of the December 4-6, 1962, fall joint computer conference.* p. 97–107. 1962

Outline

Software?

- At this time software was free for a costly machine
- Software was implicitly Open Source
- Since all this GPU-like stuff is old too, why not Open Source software for GPU nowadays too?
- Software is evolving slower than hardware architecture ☹
- Lot of legacy software need to survive to manycore revolution
- Need software tools and environments that will last through business plans or companies (HyperC... ☹)
- We need open standards to avoid sticking to some architectures
- Open implementations are a warranty for long time support for a technology (cf. current tendency in military and national security projects)

Extracting parallelism in applications...

- The implicit attitude
  - ▶ Hardware: massively superscalars processors
  - ▶ Software: auto-parallelizing compilers
- The ($\pm$) explicit attitude
  - ▶ Languages ($\pm$ extensions): OpenMP, UPC, HPF, Co-array Fortran (F- -), Fortran 2008, X10, Chapel, Fortress, Matlab, SciLab, Octave, Mapple, LabView, nVidia CUDA, AMD/ATI Stream (Brook+, Cal), OpenCL, HMPP, *insert your language here* ...
  - ▶ Libraries: application-oriented (mathematics, coupling...), parallelism (MPI, concurrency *pthreads*, SPE/MFC on Cell...), Multicore Association MCAPI, objects (parallel STL, TBB, Ct...)

... but multidimensional heterogeneity!

Welcome into Parallel Hard-Core Real Life 2.0!

- Heterogeneous execution models
  - ► Multicore SMP $\pm$ coupled by caches
  - ► SIMD instructions in processors (VMX, SSE 4.2, 3DNow!, LRBni...)
  - ► Hardware accelerators (MIMD, MISD, SIMD, SIMT, FPGA...)
- New heterogeneous memory hierarchies
  - ► Classic caches/physical memory/disks
  - ► Flash SSD is a new-comer to play with
  - ► NUMA (*Non Uniform Memory Access*) : sockets-attached memory banks, remote nodes...
  - ► Peripherals attached to sockets : NUPA (*Non Uniform Peripheral Access*). GPU on PCIe $\times 16$ in this case...
  - ► If non-shared memory: remote memory, remote disks...
  - ► Inside GPU : registers, local memory, shared memory, constant memory, texture cache, processor grouping, locked physical pages, host memory access...
- Heterogeneous communications
  - ► Anisotropic networks
  - ► Various protocols

⚠ Several dimensions to cope with at the same time☺ ◄ ≣ ► ◄ ≣ ►

Use already optimized Open Source code!

- Quite complex... ☹
- If some parts of codes are available for free : use them ☺
- http://gpgpu.org
-
  http://developer.amd.com/samples/streamshowcase/Pages/default.aspx
- http://www.nvidia.com/object/cuda_home.html

Outline

BigDFT

- Won the Joseph Fourier Prize 2009! ☺
- Compute electron density of multi-atom systems with Density Functional Theory
- Use a wavelet basis set
- Fortran + MPI + S_GPU
- S_GPU (LIG): C++ library to shared multiple GPU on a SMP node
- Added a Fortran binding to S_GPU

`http://inac.cea.fr/L_Sim/BigDFT`

Outline

OpenMM

- Library which provides tools for modern molecular modeling simulation
  - ▶ Use of modern force fields (CHARMM, AMBER, OPLS, GROMOS, GROMACS, AMOEBA)
  - ▶ Polarizable force fields (CHARMM and AMOEBA)
  - ▶ Explicit and implicit solvation, including different Generalized Born (GB) variants
  - ▶ Support for a variety of integrators, thermostats, barostats, and dynamics methods
- Support for various accelerators (multicores, SSE, GPU)
- Part of the Simulation Toolkit in the Simbios project funded by NIH

```
https://simtk.org/home/openmm
```

GpuCV

- Image processing
- Intel's OpenCV-like fully compatible programming interface
- GpuCV operators definitions are based on OpenCV definitions
- Support of native OpenCV structures like `CvArr`, `CvMat`, `IplImage`...
- Automatic data transfer management between central memory and graphics memory
- Dynamic switching mechanism between GpuCV and OpenCV operators are available to fit application needs
- Developed at Institut TÉLÉCOM SudParis!

`https://picoforge.int-evry.fr/cgi-bin/twiki/view/Gpucv/Web`

GPU VSIPL

- GPU implementation of the Vector Signal Image Processing Library (`http://www.vsipl.org`)
  - ▶ Vector and matrix types: real, complex, integer, boolean
  - ▶ Element-wise arithmetic, logical, and comparison operators
  - ▶ Linear algebra procedures
  - ▶ Generalized matrix product
  - ▶ Fast FIR filtering
  - ▶ Correlation
  - ▶ Fast Fourier Transform
  - ▶ QR decomposition
  - ▶ Random number generation
- Freeware from Georgia-Tech

`http://gpu-vsipl.gtri.gatech.edu`

Mars

- *Lisp is back ☺
- MapReduce framework on GPU
- MapReduce made fashionable again by Google and other cloud hype
- Lock-free implementation (...on G80)
- (Not the same as *Machine À Réduction Symbolique* from Toulouse in the 80's)

`http://www.cse.ust.hk/gpuqp/Mars.html`

Outline

Thrust: a Template Library for CUDA Applications

- Object programming using parallelized classes ⤳ parallel programming!
- Provide C++ STL-like interface for CUDA
- Containers for host and GPU
- Sorts
- Parallel prefix and reduction operations

`http://thrust.googlecode.com/`
Other approach: CuPP

-
    `http://www.plm.eecs.uni-kassel.de/plm/fileadmin/pm/publications/bre`

GPUmat

- Execute GPU computations from Matlab
- Use new Matlab objects such as `GPUsingle`
- Computation on these objects are executed on GPU
- Need to change the code to use these new objects
- Freeware developed by GP-you group

`http://gp-you.org`

Outline

Barra: A Modular Functional GPU Simulator

- Simulate CUDA programs at PTX assembly level
- Bit-accurate simulation of a G80
- Used to debug, profile and optimize CUDA applications
- Cycle-accurate on-going...

http://gpgpu.univ-perp.fr/index.php/Barra

- Also use http://www.cs.rug.nl/~wladimir/decuda to disassemble CUDA binaries
- Another similar project used to do architectural research : GPGPU-Sim http://www.ece.ubc.ca/~aamodt/gpgpu-sim

Outline

F2C-ACC

- Fortran 95 to C for CUDA translator
- Parse all the language but do not produce complete code
- `m4`-based! Good challenge! ☺
- But may be useful as a preprocessor before hand-made translation
- All the CUDA iteration space is to be added by hand...

`http://www-ad.fsl.noaa.gov/ac/Accelerators.html`

Jacuzzi

- Java binding for Cuda
- Only CUDA driver layer right now
- CUDA `__global__` functions are not in Java
- Work in progress...

`http://sourceforge.net/apps/wordpress/jacuzzi`

Outline

OpenGPU

- Cluster of academic and industrial partners to provide an open software platform around GPU for research and industry
- Submitted project to the *Pôle de Compétitivité System@TIC FUI8*
- Benchmarking platform: experimentation and measure on a powerful heterogeneous parallel computer with real applications
- Ease parallelization and modernization of applications in C, C++, Java & Fortran
- Extraction of design patterns and BCP (Best Current Practices) for HPC on GPU
- Green computing & green IT

OpenGPU core technologies

We will not spoil your taxes!

¡Open Standards & Open Source! ☺

- OpenCL
  - ▶ New standard for GPU and manycore programming
  - ▶ Supported by all major industries in the business
  - ▶ Expected to be as widely adopted as OpenGL for graphics
- Eclipse integrated development environment
  - ▶ Widely adopted Open Source environment
  - ▶ Run on all operating systems
  - ▶ Extensive plug-in architecture for various extensions
  - ▶ Plug-in for parallel programming already available
    `http://www.eclipse.org/ptp`

OpenGPU partners

Rethink classical industry/academic split world

- Users
  - ▶ Total (seismic), Thales (embedded systems), Numtech (decision), IBISC/Evry University & INRA (bioinformatics), ESI Group (engineering simulation), Digiteo/Scilab (numerical computing), Wallix (security), GENCI (dissemination), ECP/CRSA (hybrid computing), Alliance Service+ (heterogeneous computing SaaS), CEA DAM (secret stuff ☺), Ter@tec (relation with external partners)
- Programming
  - ▶ CAPS Entreprise (parallelization HMPP $\leadsto$ OpenCL), ATEJI (Java $\leadsto$ OpenCL), INRIA/Alchemy & CEA LIST (OpenCL & GCC), LIP6 (parallel prototypes), Thales TRT (SpearDE $\leadsto$ OpenCL), Mines ParisTech ARMINES/CRI & HPC Project (parallelization C & Fortran $\leadsto$ OpenCL & Eclipse)
- Hardware architecture
  - ▶ Bull
- Validation

Par4All & HPC Project

- HPC Project needs tools for its hardware accelerators (*Wild Nodes* from *Wild Systems*) and to parallelize, port & optimize customers applications
  `http://www.wild-systems.com`
- Unreasonable to begin yet another new compiler project...
- Many academic Open Source projects are available...
- ...But customers need products ☺
- ~ Funding an initiative to industrialize Open Source tools
- PIPS is the first project to enter the Par4All initiative
- OpenGPU project aims at increasing Par4All with HPC Project participation

`http://www.par4all.org`

PIPS *(I)*

- Since all the domain is quite new, we've started a quite new project ☺
- PIPS (Interprocedural Parallelizer of Scientific Programs) Open Source project from Mines ParisTech... 21-year old!
- Funded by many people in the audience (Industry & Research Ministry, University, CEA, DGA, IFP, Onera, ANR, European projects, pôles de compétitivité...)
- 550 KLOC...
- ... but modular and sensible approach to pass through the years
  - ▶ 242 phases (parsers, analyzers, transformations, optimizers, parallelizers, code generators, pretty-printers...)
  - ▶ Object description language for automatic generation of methods, persistence, XML marshaling for interfacing with external tools...
  - ▶ Interprocedural *à la* `make` engine to chain the phases as needed
  - ▶ Polytope lattice (linear algebra) used for semantics analysis

PIPS *(II)*

- ▶ Huge on-going efforts to industrialize the project, extension of the semantics analysis for C
- Many programmers (14) develop in PIPS (Mines ParisTech, IT SudParis, TÉLÉCOM Bretagne, RPI, HPC Project) around a public `svn`, Trac, mailing lists, IRC, Plone, Skype... and use it for many projects
  - ▶ Automatic parallelization (C & Fortran to OpenMP)
  - ▶ Distributed memory computing (OpenMP to MPI)
  - ▶ Automatic vectorizer (SSE*x*, VMX...)
  - ▶ Compilation for hardware accelerators (SIMD, FPGA, GPU...)
  - ▶ High-level hardware synthesis (FPGA)
  - ▶ Reverse engineering & decompiler
- But...
  - ▶ Huge need of documentation (even if PIPS uses literate programming...)
  - ▶ Need of industrialization

PIPS *(III)*

▶ Need further communication (even academic publications...) to increase community size

↝ Par4All

C to OpenCL in one slide (+ few PhD reports...)

- Use various interprocedural analysis from PIPS
- Use various parallelization phases of PIPS or OpenMP `#pragma`
- Outline parallel code with array region analysis (a former Supelec student with Mines ParisTech PhD! ☺)

```
1  //  <h_A[PHI1][PHI2]-W-EXACT-{0<=PHI1, PHI2+1<=n, PHI1<=PHI2}>
     for(i = 0; i <= n−1; i += 1)
3  //    <h_A[PHI1][PHI2]-W-EXACT-{PHI1==i, i<=PHI2, PHI2+1<=n, 0<=i}>
       for(j = i; j <= n−1; j += 1)
5  //      <h_A[PHI1][PHI2]-W-EXACT-{PHI1==i, PHI2==j, 0<=i, i<=j, 1+j<=n}>
         h_A[i][j] = 1;
```

- Allocate variables on the accelerator using array region analysis
- Generate host-GPU transfers using IN/OUT... array region analysis!
- Add an OpenCL code generator
- Pretty-print target codes

Conclusion *(I)*

- GPU (and other heterogeneous accelerators) : impressive peak performances and memory bandwidth, power efficient
- Looks like to the 80's : many accelerators, vectors, parallel computers
- Need to port the programs to new architectures and many... companies! ☺
- ...but some hypothesis changed, hardware constraints ⤳ we can no longer escape parallelism!
- And programs are quite bigger now! ☺
- Non-functional specifications (speed, time, power, parallelism...) not taken into account by main-stream programming environment and teaching (high-level object programming...)
- Programming is *quite more* complex... ☺

Conclusion *(II)*

- ...and most programmers in the wild did not even study « normal » computer science...
- Development environments & methods are young compared to « classical » (sequential) programming
- Standards? Often *de facto* from vendors
- ⚠ Entry cost
- ⚠ ⚠ ⚠ Exit cost! ☹
- Before addressing parallelism, may be interesting to optimize the sequential program (algorithmic, coding, right tools...)
- Take a positive attitude...Parallelization is a good opportunity for deep cleaning (refactoring, modernization...) ⤳ improve also the original code
- ...but need good competences in application domain *and* computer science. Sunny days for people with these multi competences ☺

Conclusion *(III)*

- Hard to address multidimensional heterogeneity... ↝ *real politik* ? Restriction to stream programming...?
- `#pragma`tic attitude at all levels to deal with legacy applications
  - ▶ Use source-to-source compilers to surf on the best back-ends
  - ▶ Use `#pragma` when necessary to help the tools (OpenMP, HMPP, various extensions...)
  - ▶ Back-ends use more an more research results (gcc, LLVM, Brook+, nvcc, icc, PGI...)
  - ▶ Hide manual work in preprocessing, tools, code generators...
  - ▶ Use standards and Open Source for long-term investment

Conclusion *(IV)*

- Many companies and academic teams committed to GPU in France: critical mass to influence GPU history
- Rejoignez le groupe de travail « Architectures émergentes pour le calcul hautes performances » du GdR CNRS ASR & INRIA ! `http://gpgpu.univ-perp.fr`
- École thématique 7-11 décembre 2009, Banyuls : « Calcul hautes performances sur accélérateurs matériels » (GPU, FPGA, Cell) `http://surena.univ-perp.fr/GPGPU/school.php`