

Some personal views about compilation for embedded systems

Ronan KERYELL
rk@hpc-project.com

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

9 Route du Colonel Marcel Moraine
92360 Meudon La Forêt, FRANCE

Rond Point Benjamin Franklin
34000 Montpellier, FRANCE

Associate researcher at Institut TÉLÉCOM/TÉLÉCOM Bretagne/HPCAS, Plouzané

10/14/2009

<http://www.par4all.org>



WildNode from Wild Systems subsidiary company

- Low noise hardware desktop accelerator for in-office operation
- x86 manycore, GPU, FPGA, Linux & Windows
- Parallelize and optimize customer applications, co-branded as a bundle product in a WildNode (e.g. Presagis Stage battle-field simulator)



<http://www.wild-systems.com>



HPC Project software and services

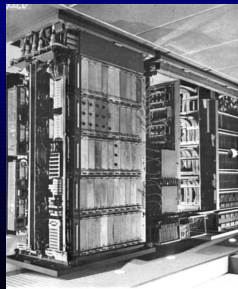
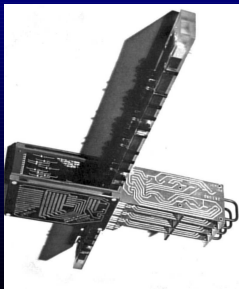
- Optimization and parallelization of applications
- GPU-accelerated libraries for Matlab/Octave/R
- Embedded systems and application design
- Parallelizing and optimizing compiler
- Training in parallel programming (OpenMP, MPI, TBB, CUDA, OpenCL...)



Multicores strike back...

Gamma 60 from Compagnie des Machines Bull ≈ 1959

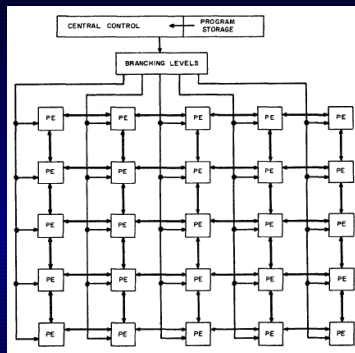
- Tremendous ultrasonic 100 kHz memory clock
- *Heterogeneous* multicore because the memory is too... fast! ☺
- SIMU (\approx fork) & CUT (launch a program on a functional unit) instructions
- Huge 24-bit words, 96 KiB of core memory, punch-cards with ECC, magnetic tapes, magnetic drums
- Blazing integrated logic in 1-mm germanium bipolar lithography ☺



GPGPUs: just more integrated...

- SOLOMON, 1962

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



That was the oldies... Now:

- System... ~ on Chip
- Multi-Processor System... ~ on Chip
- Network... ~ on Chip
- Heat... ~ on Chip ☹

Off-the-shelf AMD/ATI Radeon HD 5870 GPU at \$379

- 2.15 billion 40nm transistors
- 1600 stream processors @ 850 MHz, 2.7 TFLOPS SP, 0.5 TFLOPS DP
- + External 1 GB GDDR5 memory 4.8 Gt/s, 153.6 GB/s
- 188 W on board (27 idle), PCI Express 2.1 x16 bus interface
- OpenGL, OpenCL



×1000 high-end real-man DSP peak performance... Motivate some time at considering this ridiculous kiddie game non-serious technology... ☺

SoCftware...

... on Chip?

↪ ⚠ The ultimate impedance (mis)match! ☹

CoCmpiler...

... on Chip?

↪ The ultimate Swiss toCoCI?



Outline

1 SoCftware

2 Par4All



¿Did you say software?

Software tools for parallel computers from the oldies

- The Holly Hope in Automatic Parallelization of sequential languages
- Many parallel languages created
- No real winning solution ☹
- Ferrari, Porsche, rocket science,... are very attractive but also niche market ☹
- Tough work ☹: exhausted research pioneers migrated to ever-green hype paradises (Web 3.0, Grid/Cloud/Green computing, UML/MDA/(meta-) $n \gg 1$ stuff... ☹)

But few ones *did* survive and are in this panel or hide them-self in the wild ☹

Parallelism is no longer a niche market!

Parallelism is everywhere (choose your religion here: ☺ or ☹?)



Real software

- Often sequential
- Real codes are often not well written to be parallelized... even by human being ☹
- At least writing clean C99 code should be a prerequisite
 - ▶ Language more subtle than often thought ⚠
 - ▶ It really does need some learning, imitation is not enough to avoid `malloc()`, `free()`, pointer... spamming ☹
- Object oriented software
 - ▶ Quite useful at high level...
 - ▶ Often not written for low-end optimized execution ⚠
 - ▶ Optimization at low level may need to change all the model ⚠ ⚠



Extracting parallelism in applications...

- The implicit attitude
 - ▶ Hardware: massively superscalars processors:
Only works for low level of parallelism
 - ▶ Software: auto-parallelizing compilers:
May not extract all parallelism either
- 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, PGI Accel, *insert your language here*...
 - ▶ Libraries: application-oriented (mathematics, coupling...), parallelism (MPI, concurrency *pthreads*, SPE/MFC on Cell...), Multicore Association MCAPI, objects (parallel STLs, TBB, Ct, Thrust, CuPP...)
Avoid introducing new languages



... but multidimensional heterogeneity!

Welcome into Parallel Hard-Core Real Life on Chip 2.0!

- Heterogeneous execution models
 - ▶ Multicore (\pm S)MP \pm coupled by caches
 - ▶ SIMD instructions in processors (Neon, VMX, SSE 4.2, LRBni...)
 - ▶ Hardware accelerators (MIMD, MISD, SIMD, SIMT, FPGA, ASIC...)
- 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 ☹️

Not reinventing the wheel

Want to create your own tool?

- House-keeping and infrastructure in a compiler is a **huge** task
- ~→ Integrate your ideas and developments in existing project
- ...or buy one if you can afford (ST with PGI...) ☺
- Some projects to consider
 - ▶ Old projects: gcc, PIPS...
 - ▶ But new ones appear too: LLVM, RoseCompiler, Cetus...



Associate different tools!

- Real life \equiv whole project, not individual issues
- Whole issue is too complex, no tool or programming model can fit all problems ☹
- Many compilers & tools available for different parts
- Each one often good for its own job ☺
- \rightsquigarrow Need global integration of many different tools
- Need a global vision of the whole process (hardware, software, application...) ⚠

Small detail...

It has not happened yet (at least for the last 20 years among the French projects...) ☹

(\rightsquigarrow Pile up a subliminal question to my French panelist colleagues: Why?)



Bridging tools from different domains...

- Academic: glueing is « trivial » task ☺, not research (except if your research domain is physical chemistry about glues or quantum physics on gluons ☺)
- Industry: we use products, not research toys!

Come on... stop kidding! The world is waiting for salvation from all of us... ☺

Few glue brands

- XML... Useful but just syntax. The *real* issue is semantics ☺
- Exchange Plain-ole C99 programs (the poor man executable semantics ☺) with some directives/#pragma/... if not enough
 ~ Source-to-source up to the back-end tools
- Scripting is back (SWIG...): most promising technology to manage all the interactions



Bridging tools from different domains...

- Academic: glueing is « trivial » task ☺, not research (except if your research domain is physical chemistry about glues or quantum physics on gluons ☺)
- Industry: we use products, not research toys!

Come on... stop kidding! The world is waiting for salvation from all of us... ☺

Few glue brands

- XML... Useful but just syntax. The *real* issue is semantics ☺
- Exchange ~~Plain old~~ C99 programs (the poor man executable semantics ☺) with some directives/#pragma/... if not enough
 ↪ Source-to-source up to the back-end tools
- Scripting is back (SWIG...): most promising technology to manage all the interactions



Outline

1 SoCftware

2 Par4All



We need software tools

Issue to solve

Save the world: the world needs tools

Mmmm... Huge work... Too difficult... ☺ Try to simplify and restrict the problem... ☺

Issue to solve

HPC Project (and its customers) need tools

- Parallelize, port & optimize customer applications
- Application development: long-term business ~ long-term commitment in a tool that needs to survive to (too fast) technology change
- Unreasonable to begin yet another new compiler project...
- Many academic Open Source projects are available...
- ...But customers need products ☺



We need software tools

Issue to solve

Save the world: the world needs tools

Mmmm... Huge work... Too difficult... ☹ Try to simplify and restrict the problem... ☺

Issue to solve

HPC Project (and its customers) need tools

- Parallelize, port & optimize customer applications
- Application development: long-term business ~ long-term commitment in a tool that needs to survive to (too fast) technology change
- Unreasonable to begin yet another new compiler project...
- Many academic Open Source projects are available...
- ...But customers need products ☺



Use the Source, Luke...

Hardware is moving quite (too) fast but...

What has survived for 50+ years?

Fortran programs...

What has survived for 30+ years?

C programs, Unix...

- A lot of legacy code could be pushed onto parallel hardware (accelerators) with automatic tools...
- Need automatic tools for source-to-source transformation to leverage existing software tools for a given hardware
- Not as efficient as hand-tuned programs, but quick production phase



- PIPS (Interprocedural Parallelizer of Scientific Programs): Open Source project from Mines ParisTech... \approx 150 h·y R&D, 21-year old!
- Funded by many people (French DoD, Industry & Research Departments, University, CEA, IFP, Onera, ANR (French NSF), European projects, regional research clusters...)
- One of the project that coined polytope model-based compilation
- \approx 456 KLOC according to David A. Wheeler's SLOCcount
- ... but modular and sensible approach to pass through the years
 - ▶ \approx 300 phases (parsers, analyzers, transformations, optimizers, parallelizers, code generators, pretty-printers...) that can be combined for the right purpose
 - ▶ NewGen object description language for language-agnostic automatic generation of methods, persistence, object introspection, visitors, accessors, constructors, XML marshaling for interface with external tools...



- ▶ Interprocedural *à la make* engine to chain the phases as needed.
Lazy construction of resources
- ▶ Polytope lattice (linear algebra) used for semantics analysis, transformations, code generation... to deal with big programs, not only loop-nests
- ▶ Huge on-going efforts to industrialize the project, extension of the semantics analysis for C
- Around 15 programmers currently developing in PIPS (Mines ParisTech, HPC Project, IT SudParis, TÉLÉCOM Bretagne, RPI) with public `svn`, `Trac`, `git`, mailing lists, IRC, Plone, Skype... and use it for many projects
- But still...
 - ▶ Huge need of documentation (even if PIPS uses literate programming...)
 - ▶ Need of industrialization
 - ▶ Need further communication to increase community size



Par4All

- ~> Funding an initiative to industrialize Open Source tools
- PIPS is the first project to enter the Par4All initiative

<http://www.par4all.org>



Current PIPS usage

- Automatic parallelization (C & Fortran to OpenMP)
- Distributed memory computing with OpenMP-to-MPI translation [STEP project]
- Generic vectorization for SIMD instructions (SSE, VMX, Neon...) (SAC project) [SCALOPES]
- Parallelization for embedded systems [SCALOPES]
- Compilation for hardware accelerators (Ter@PIX, SPoC, SIMD, FPGA...) [FREIA, SCALOPES]
- High-level hardware accelerators synthesis generation for FPGA
- Reverse engineering & decompiler (reconstruction from binary to C)
- GPU CUDA code generation [TransMedi@, FREIA, OpenGPU]
- Genetic algorithm-based optimization [Luxembourg university]



- PYPS: polishing Python scripting with SWIG (~ easy extension to other religions ☺)
- Mix heterogenous //: GPU and CPU execution, with SSEx, OpenMP and/or MPI...
- OpenCL output [OpenGPU project]
- If someone spent some time to write `#pragma` information... Use them! ☺
- Eclipse (EcliPIPS) integration to apply transformations and get easy feedback [OpenGPU project]
- Use advanced tiling to take advantage of various weird memory hierarchy: cache/shared/non-coherent/shared memory/texture cache/...
- Select optimal transformations and code generation parameter by using genetic algorithm system



- Using complexity information to select the best execution mode (GPU or CPU)
- Matlab to CUDA compiler
- Generate run-time specialization and speculation when some static information is lacking at compilation time



Conclusion



- We need open standards to avoid sticking to some architectures
- Need software tools and environments that will last through business plans or companies
- Open implementations are a warranty for long time support for a technology (cf. current tendency in military and national security projects)
- End of clear separation of embedded/MID/PC/HPC systems
- Tools too complex to afford 1 different tool per application/architecture domain
- Scripting is an efficient way to team different tools
- Lot of legacy software need to survive to manycore revolution
- Source-to-source compiler a good place to begin with: the source is the value of the application *right now*
- Use `#pragma` to exchange meta-informations



- Let the Source be with You!

¡We need you!

We have already PhD students, engineers and researchers around Par4II, but many hard issues remains...



Par4All is currently supported by...

- HPC Project (France)
- Mines ParisTech (France)
- Institut TÉLÉCOM/TÉLÉCOM Bretagne (France)
- Rensselaer Polytechnic Institute (USA)
- European ARTEMIS SCALOPES project
- French NSF (ANR) FREIA project
- French Images and Networks research cluster TransMedi@ project
- French System@TIC research cluster OpenGPU project



