Performance Analysis of HPC Applications on Low-Power Embedded Platforms

Luka Stanisic, Brice Videau, Johan Cronsoie, Augustin Degomme, Vania Marangozova, Arnaud Legrand, Jean-François Méhaut (CEA, CNRS, INRIA, UJF, Grenoble, France)
Edson Padoin, Laercio Pilla, Pedro Velho, Marcio Castro, Alexandre Carissimi, Philippe Navaux (UFRGS, Porto Alegre, Brazil)
HPC: “We have a dream…”
“Reaching the Exaflop before 2020”

- Current efficiency: ~2 GFLOPS/W
- 20 MW Budget
- Target efficiency: 50 GFLOPS/W
- 25x better efficiency needed
Potential Solution: the Embedded Way

- Use components designed and produced by the embedded (European) industry (ARM, STMicroelectronics, Kalray,...)
- Power efficiency is excellent
- Use off-the-shelf components to keep costs down
- The Mont-Blanc European Project is working on the use of these components for HPC
Mont-Blanc 1 - Objectives

- Develop prototypes of HPC clusters using low power (ARM) commercially available embedded technology.
- Design the next generation in HPC systems based on embedded technologies and experiments on the prototypes.
- Develop a portfolio of existing applications to test these systems and optimize their efficiency, using BSC's OmpSs programming model (11 existing applications were selected for this portfolio).
Scalability of the ARM Architecture

- Low power foundation enables ARM processor to scale from sensors to servers
  - Software compatibility makes it easier to scale up
  - Lower power processors enables a more integrated system-on-chip
- Power efficiency is important for all applications
  - More attractive system design – thinner and lighter
  - Lower cost – plastic chip packaging, no heat sink or fan, cheaper components
  - Lower energy use – reduced running costs and carbon footprint

*Infrastructure, Consumer, Mobile, Microcontrollers, Sensors*
ARM Applications Processors

Cortex-A Series
“Low-Power Leadership”

**Cortex-A15**
- >2GHz in 28HPM
- Virtualization
- 1TB physical addressing
- big.LITTLE with Cortex-A7

**Cortex-A15**
- ARMv8 64-bit
- Same power, 3x the performance of today's superphones

**Cortex-A9**
- Shipping in mobile since 2009
- 2nd generation 1-4X SMP
- 4x1750DMIPS@700MHz+ in 40LP

**Cortex-A7**
- 1/5 the power of Cortex-A15
- Architectural alignment with Cortex-A15

**Cortex-A53**
- ARMv8 64-bit
- Same performance, ¼ the power of today's superphones

**Cortex-A57**
- ARMv8 64-bit
- Same power, 3x the performance of today's superphones

Performance
Mainstream
High Efficiency
Future
Cortex-A57: Implementation for High Performance

- Maximum performance in smartphone power budget
  - 3x performance of 2012 super phones, in 32-bit mode
  - Performance per clock comparable to today’s PC’s

- Driving advanced computing
  - 5x power-efficiency for tomorrow’s tablets and notebooks
  - Mobile-level power consumption for enterprise and cloud

- Enhanced capabilities for enterprise and servers
  - 64-bit support for full range of applications
  - High performance IEEE compliant DP floating point / SIMD
  - Solutions already scaling up to a 16x SMP compute node
  - SoC flexibility to scale efficiently to 100K’s or more nodes
## Mont-Blanc 1 – Applications Beyond Linpack

<table>
<thead>
<tr>
<th>Code</th>
<th>Scientific Domain</th>
<th>Contact</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>YALES2</td>
<td>Combustion</td>
<td>V. Mouveau</td>
<td>CNRS/CORIA</td>
</tr>
<tr>
<td>EUTERPE</td>
<td>Fusion</td>
<td>T. Akgun</td>
<td>BSC</td>
</tr>
<tr>
<td>SPECFEM3D</td>
<td>Wave propagation</td>
<td>D. Komatitsch</td>
<td>CNRS</td>
</tr>
<tr>
<td>MP2C</td>
<td>Multi-particle collision</td>
<td>G. Sutmann, A. Schiller</td>
<td>JSC</td>
</tr>
<tr>
<td>BigDFT</td>
<td>Elect. Structure</td>
<td>T. Deutsch</td>
<td>CEA</td>
</tr>
<tr>
<td>Quantum Expresso</td>
<td>Elect. Structure</td>
<td>C. Cavazzoni</td>
<td>CINECA</td>
</tr>
<tr>
<td>PEPC</td>
<td>Coulomb + gravitational forces</td>
<td>P. Gibbon, L. Arnold</td>
<td>JSC</td>
</tr>
<tr>
<td>SMMP</td>
<td>Protein folding</td>
<td>J. Meinke</td>
<td>JSC</td>
</tr>
<tr>
<td>ProFASI</td>
<td>Protein folding</td>
<td>S. Mohanty</td>
<td>JSC</td>
</tr>
<tr>
<td>COSMO</td>
<td>Weather forecast</td>
<td>M. Culpo</td>
<td>CINECA</td>
</tr>
<tr>
<td>BQCD</td>
<td>Particle physics</td>
<td>M. Allalen</td>
<td>LRZ</td>
</tr>
</tbody>
</table>
SPECFEM3D (INRIA, CNRS,...)

- acoustic wave propagation for geophysics (earthquakes) or industrial problems (non destructive testing, ocean acoustics...)

- Implementation details
  - 80,000 lines of Fortran 90 – MPI, CUDA; a mini-app port in OMPSs + MPI exists; External libs : none
  - Portability : runs on almost all types of existing systems
  - Scalability : excellent, 0.7 PF on 149 784 cores of Jaguar and up to 1152 GPUs (TSUBAME 2.0)
BigDFT (CEA INAC, UJF,...)

- DFT code using Daubechies wavelets for electronic structure calculation in nano-sciences and biology

- Implementation details
  - 200,000 lines of Fortran 90, C
  - Supports MPI, OpenMP, CUDA and OpenCL
  - External libs: BLAS
  - Portability: wide
  - Scalability: up to 4000 cores of BG/P and 288 GPUs
Snowball Board

- Full embedded system by Calao Systems
  - Dual Core **ST-Ericsson** ARM A9 1GHz SoC
  - Neon Floating point Unit
  - Integrated GPU
  - 1 Gbyte RAM
  - HDMI output
  - Ethernet
  - Runs Linux (Linaro Ubuntu) or Android
  - Igloo community for development and support
  - 2.5W maximum energy consumption
Experiments on Snowball

- Benchmarks
  - Linpack
    - Standard benchmark for HPC
  - CoreMark
    - Aims at becoming and industry standard for embedded platforms
    - Lots of results available on the website, allows to compare quickly CPU efficiency
  - StockFish
    - Open Source chess engine with benchmarking capabilities
## Snowball results

- Compared to an Intel Xeon® x5550 (4 Nehalem cores, TDP:95W)
  - Ratio : Xeon perf / Snowball perf
  - Perf/Watt ratio : (Xeon perf/95)/(Snowball perf/2.5)

<table>
<thead>
<tr>
<th>Bench</th>
<th>Snowball</th>
<th>Xeon 4 cores</th>
<th>Ratio</th>
<th>Perf / Watt Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linpack (MFlops)</td>
<td>620</td>
<td>24000</td>
<td>38.7</td>
<td>1.0</td>
</tr>
<tr>
<td>CoreMark (ops/s)</td>
<td>5877</td>
<td>41950</td>
<td>7.1</td>
<td>0.2</td>
</tr>
<tr>
<td>StockFish (ops/s)</td>
<td>224,113</td>
<td>4,521,733</td>
<td>20.2</td>
<td>0.5</td>
</tr>
<tr>
<td>Specfem3D (s)</td>
<td>186.8</td>
<td>23.5</td>
<td>7.9</td>
<td>0.2</td>
</tr>
<tr>
<td>BigDFT (s)</td>
<td>420.4</td>
<td>18.1</td>
<td>23.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Tibidabo cluster

32 blades of 8 harmony Tegra 2 boards already deployed

- 35W/blade
- 16 ARM Cortex A9 1GHz cores/blade

Tegra2 SoC:
2x ARM Coretx-A9 Cores
2 GFLOPS
0.5 Watt

Tegra2 Q7 module:
1x Tegra2 SoC
2x ARM Coretx-A9 Cores
1 GB DDR2 DRAM
2 GFLOPS
~4 Watt
1 GbE interconnect

1U Multi-board container:
1x Board container
8x Q7 carrier boards
8x Tegra2 SoC
16x ARM Coretx-A9 Cores
8 GB DDR2 DRAM
16 GFLOPS
~35 Watt

Rack:
32x Board container
10x 48-port 1GbE switches
256x Q7 carrier boards
256x Tegra2 SoC
512x ARM Coretx-A9 Cores
256 GB DDR2 DRAM
512 GFLOPS
~1.7 Kwatt
300 MFLOPS / W
Slightly quicker than snowball board, but given for 4 Watts

<table>
<thead>
<tr>
<th></th>
<th>Tegra 2</th>
<th>Snowball</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coremark</td>
<td>5952</td>
<td>5877</td>
</tr>
<tr>
<td>Specfem3D 2 cores</td>
<td>178.9s</td>
<td>186.8s</td>
</tr>
<tr>
<td>Linpack 2 cores peak</td>
<td>698 MFlops</td>
<td>620 MFlops</td>
</tr>
</tbody>
</table>
Impossible d'afficher l'image. Votre navigateur manque peut-être de mémoire pour ouvrir l'image ou l'image est endommagée. Redémarrez l'ordinateur, puis ouvrez le fichier. Si le x rouge est toujours affiché, vous devrez peut-être supprimer l'image avant de la réinsérer.
Tibidabo - scaling

Most of BIGDFT communications are delayed.

Poor scaling due to collective communications problems: speedup 15 for 36 cores

Normal communication (small communications)

Delayed communication

Communications

Computations
Optimization is Never-ending

- HPC environment is ever-changing, Multiple paradigms coexist
- Architectures have to be efficiently used

<table>
<thead>
<tr>
<th>Architecture</th>
<th>AMD/Intel CPU</th>
<th>ARM</th>
<th>GPU</th>
<th>MIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores</td>
<td>4-12</td>
<td>2-4</td>
<td>512-1536</td>
<td>64</td>
</tr>
<tr>
<td>Cache</td>
<td>3l coherent</td>
<td>2l coherent</td>
<td>1l incoherent</td>
<td>2l coherent</td>
</tr>
<tr>
<td>Memory (GiB)</td>
<td>2-4 (per core)</td>
<td>1 (per core)</td>
<td>2-6</td>
<td>2-?</td>
</tr>
<tr>
<td>Vector Size</td>
<td>2-4</td>
<td>1-2</td>
<td>1-2</td>
<td>8</td>
</tr>
<tr>
<td>Peak GFLOPS</td>
<td>10-20(per core)</td>
<td>2-4 (per core)</td>
<td>500-1000</td>
<td>1000</td>
</tr>
</tbody>
</table>
Counter-Intuitive Results

Seemingly harmless optimization can prove harmful on different architectures

Intel Xeon 5500

ARM A9500
Memory alignment and scheduling policies are important:

- Real-time is counter productive
- Variations due to memory continuity
MB2: Use Auto-Tuning in Applications

- Test case: BigDFT
- Build a parametrized generator that can optimize BigDFT building blocks: convolutions
- Those operations are badly optimized and vectorized by compilers (Intel Nehalem, also ARM):
  - Simple: 0.3 GFLOPS
  - Hand unrolled: 3 GFLOPS
  - Hand vectorized: 6.5 GFLOPS
MB2: Harness the Developer Knowledge

- The developer often knows the kind of optimization that would be beneficial
- Even when asked nicely compilers don't always comply
- Give developers tool to meta-program optimizations in their code
MB2: Source-to-Source Transformation

- Bringing Optimization through Automatic Source-to-Source Transformations
- Used to generate multiple versions of the reference convolution
- Versions are automatically benchmarked using performance counters.
- Selection of best versions
Different optimal unroll degree and resource usage pattern, depending on the architecture.
Conclusions – Mont-Blanc 1

- Porting to Tibidabo is a success
- Most limitation come from the prototype (network) and should be lifted in the next version
- Second round of porting is to begin
- SPECFEM3D and BigDFT kernels have been selected specifically for optimization
Perspectives – Mont-Blanc 2

- Generator is a success
- Porting to the second Mont-Blanc prototype: an opportunity to generalize our generator?
- Apply generator to OpenCL kernels instead of using C++
- Can we generate vectorized convolutions?