Efficient numerical simulation on multicore processors (MuCoSim)

07.10.2014

Prof. Gerhard Wellein, Dr. G. Hager
HPC Services
Regionales Rechenzentrum Erlangen (RRZE)
Department für Informatik

http://moodle.rrze.uni-erlangen.de/moodle/course/view.php?id=307
http://goo.gl/43NbHH
Efficient numerical simulation on multi-core processors

- We do performance optimization, performance modeling, parallelization for
  - Multi-core CPUs: core, socket, node and large scale 10,000+ cores
  - GPGPUs: single devices and cluster
  - Many-core CPUs: Intel Xeon Phi

- We collaborate with many users doing numerical simulation, e.g.:
  - Prof. Rüde: waLBerla / efficient C++; Prof. Fey; Prof. Teich
  - Chemistry, Physics
  - Engineering: Prof. Schwieger, PD Dr. S. Becker
  - Medical Image Reconstruction: Prof. Hornegger

- We operate the compute resources at FAU

**Our group:**
- 5 senior scientists (incl. RRZE) (GW/GH/TZ/MM/JT)
- 5 PhD students (MW/FS/MK/HS/TR)
- 2 Master students

07.10.2014 hpc@rrze.uni-erlangen.de
Effiziente numerische Simulation auf multicore-Proz.

Hintergrund

- **RRZE Testcluster („Playground“)**
  - Octo-Core (Intel Sandy Bridge) – *10-core Ivy Bridge*
  - nVIDIA & AMD GPUs & Intel Xeon Phi

- **RRZE production machines (Infiniband Interconnect):**
  - 84 nodes Nehalem-Cluster → 672 cores
  - 500 nodes Intel Westmere → 6.000 cores
  - **544 nodes Intel Ivy Bridge** → 10.880 cores
  - + 8 x (2 x Intel Ivy Bridge 10 core + 2 x NVIDIA K20 GPGPU)
  - + 8 x (2 x Intel Ivy Bridge 10 core + 2 x Intel Xeon Phi)

- **Access to external machines**
  - IBM BlueGene/Q (Jülich): 450.000+ cores (6 PFLOP/s)
  - SuperMUC (LRZ Garching): 150.000+ cores (3 PFLOP/s) → see next slide
  - Hermit (HLR Stuttgart) → CRAY XE 6 (1 PFLOP/s)
SuperMUC – LRZ Garching: TOP 4 (June 2012)

- **Thin nodes:**
  - 18 Islands with 512 nodes each
  - 2 Intel Xeon E5-2680 8C processors (2.7 GHz baseline clock speed) per node
  - 147,456 cores
  - 3.2 Pflop/s Peak
  - 2.9 Pflop/s LINPACK

- **Fat nodes:**
  - 1 Island: 205 nodes
  - 4 Intel Xeon E7-4870 10 C per node
  - 256 GB/node

- **Total power consumption:** 2.5 MW – 3 MW
SuperMUC – far more than some islands.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (TFlop/s)</th>
<th>Rpeak (TFlop/s)</th>
<th>Power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DOE/SC/Oak Ridge National Laboratory</td>
<td><strong>Titan</strong> - Cray XK7, Opteron 6274 16C, 2.2GHz, Cray Gemini interconnect, NVIDIA K20x</td>
<td>560640</td>
<td>17590.0</td>
<td>27112.5</td>
<td>8209</td>
</tr>
<tr>
<td>2</td>
<td>DOE/NNSA/LLNL</td>
<td><strong>Sequoia</strong> - BlueGene/Q, Power BOC 16C, 1.6GHz, Custom</td>
<td>1572864</td>
<td>16324.8</td>
<td>20132.7</td>
<td>7890</td>
</tr>
<tr>
<td>3</td>
<td>RIKEN Advanced Institute for Computational Science (AICS)</td>
<td><strong>K computer</strong>, SPARC64 VIIIfx, 2.0GHz, Tofu interconnect</td>
<td>705024</td>
<td>10510.0</td>
<td>11280.4</td>
<td>12660</td>
</tr>
<tr>
<td>4</td>
<td>DOE/SC/Argonne National Laboratory</td>
<td><strong>Mira</strong> - BlueGene/Q, Power BOC 16C, 1.6GHz, Custom</td>
<td>786432</td>
<td>8162.4</td>
<td>10056.3</td>
<td>3945</td>
</tr>
<tr>
<td>5</td>
<td>Forschungszentrum Juelich (FZJ)</td>
<td><strong>JUQUEEN</strong> - BlueGene/Q, Power BOC 16C, 1.6GHz, Custom Interconnect</td>
<td>393216</td>
<td>4141.2</td>
<td>5033.2</td>
<td>1970</td>
</tr>
<tr>
<td>6</td>
<td>Leibniz Rechenzentrum</td>
<td><strong>SuperMUC</strong> - iDataPlex DX360M4, Xeon E5-2660 8C 2.7GHz, Infiniband FDR</td>
<td>147456</td>
<td>2597.0</td>
<td>3185.1</td>
<td>3423</td>
</tr>
<tr>
<td>7</td>
<td>Texas Advanced Computing Center/Univ. of Texas</td>
<td><strong>Stampede</strong> - PowerEdge C8220, Xeon E5-2660 8C 2.7GHz, Infiniband FDR, Intel Xeon Phi</td>
<td>204900</td>
<td>2660.3</td>
<td>3959.0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>National Supercomputing Center in Tianjin</td>
<td><strong>Tianhe-1A</strong> - NUDT YH MPP, Xeon X5670 6C 2.93GHz, NVIDIA 2050</td>
<td>186368</td>
<td>2566.0</td>
<td>4701.0</td>
<td>4040</td>
</tr>
<tr>
<td>9</td>
<td>CINECA</td>
<td><strong>Fermi</strong> - BlueGene/Q, Power BOC 16C, 1.6GHz, Custom</td>
<td>163840</td>
<td>1725.5</td>
<td>2097.2</td>
<td>822</td>
</tr>
<tr>
<td>10</td>
<td>IBM Development Engineering</td>
<td><strong>DARPA Trial Subset</strong> - Power 775, POWER7 8C 3.836GHz, Custom Interconnect</td>
<td>63360</td>
<td>1515.0</td>
<td>1944.4</td>
<td>3576</td>
</tr>
</tbody>
</table>
Motivation for this seminar

- Understand serial and parallel performance
  - on a wide range of multi-/manycore architectures:
    - Intel & AMD based x86 processors
    - GPGPUs
    - Intel Xeon Phi
    - Others?!
  - for various
    - low level benchmarks (STREAM, Stephanov,…)
    - benchmark kernels (LBM, spMVM,…)
    - programming models & languages
    - application scenarios (Fault Tolerance)
  - Using a model based approach

- Requirements:
  - C/C++ or FORTAN & basic knowledge of computer architecture (PTfS,…)
  - Interest to look under the hood
About this seminar

- Central component of most projects: Establish performance models to understand serial/parallel performance
- Access to latest hardware
- Interact with your tutor!
- Tell us if you have an interesting problem!
- Format:
  - 2 talks each 30 minutes
  - Written summary of the project (about 3-5 pages)
Performance Engineering – Our approach

1. Carefully analyze the minimum computational requirements (data volume, FLOP-ops) of the algorithm

2. Carefully analyze the computational requirements (data access in cache/main memory, FLOPS, instruction mix,..) of the implementation. Optimize if they do not fit to data from 1.

3. Analyze the available computational resources of the target hardware: Cache/Memory bandwidth, SIMD capabilities,..

4. Determine runtime / performance number based on 2 and 3.

5. Measure runtime / performance and compare with 4. Go back to 2. / 3. if numbers differ substantially
Potential topics

- **Relaxed thread synchronisation for multi-core architectures:**
  - Effect of replacing OpenMP barriers by relaxed synchronization constructs
  - Replace global synchronization by point to point synch.
  - Target architecture: 8-core / 10-core Intel CPUs; Intel Xeon Phi

- Potential kernels / applications:
  - 7pt-Gauß-Seidel
  - SIP Solver (Strongly Implicit Procedure after Stone)
  - Temporal blocking of 2D (and 3D) Jacobi smoother
Potential topics

- ILBDC Lattice Boltzmann solver (RRZE, T. Zeiser, M. Wittmann)

- Project 1:
  - GPGPU implementation of ILBDC kernel in OpenCL/CUDA
  - Evaluate impact of list ordering

- Project 2:
  - SIMD Vectorized TRT/MRT kernel
  - Code generator for automatic SIMDfication
Evaluation of the OpenACC directives on CRAY XE6

- OpenACC tries to standardize the way compiler directives are used to program accelerator devices like GPGPUs. It is available, e.g., on recent CRAY supercomputers like the HERMIT system at HLRS Stuttgart.

- STREAM benchmarks
- Jacobi solver
- spMVM
Potential topics (ESSEX project /DFG)

- **Iterative methods for sparse matrix problems**, e.g.:
  - Conjugate gradient → Solve \( A x = b \) (LSE)
  - Implement full kernel incl. spMVM
  - Performance analysis (and modeling) of the full kernel
  - Test block-spMVM version (multiple RHS vectors \( b \))

- **Parallel programming:**
  - ghost library developed by Moritz Kreutzer (within DFG Exascale project) –

  or

  - OpenCL / CUDA (**Phillip Killermann**) or CoArray Fortran -

  or

  - GPI (Fraunhofer)

- **Target machines**: Nodes with CPU / GPGPU / Intel XeonPhi
Potential topics


- Implement simple parallel kernels using GA:
  - 3D Jacobi and / or Gauß-Seidel (ppp)
  - Simple 3D Lattice Boltzmann flow solver

- Performance analysis and evaluation – establish low level benchmarks to determine qualitative difference with pure MPI

- Target machine: (Large)= compute cluster: lima / emmy

07.10.2014  hpc@rrze.uni-erlangen.de
Potential topics

- **Stone’s Strongly Implicit Procedure (SIP):**
  - Old but still frequently solver in finite volume codes
  - performs incomplete LU factorization
  - solves through iterative LU steps → carries data dependency

- **Establish benchmark framework using OpenMP from scratch**

- **CoArray Fortran implementation** → lima-cluster / CRAY XE6
Potential topics

- Asynchronous MPI communication: Using explicit threading ("task mode") to implement explicit overlap between communication and computation in different solvers.
- Non-blocking communication calls basically allow asynchronous communication – but no MPI library fully supports that.
- Test cases:
  - MPI parallel Lattice Boltzmann 3D solver on CPUs
  - MPI parallel Jacobi 3D solver on GPUs
Potential topics

- Stepanov test for modern Fortran

- Development of a modern test for the optimization capabilities of compilers, including SIMD vectorization, auto-parallelization etc. in the Fortran2008 language.
Potential topics

- Evaluation of optimization strategies for matrix-matrix multiply on modern processors.
  
  ```
  do i = 1 , N 
    do j = 1 , N 
      do k = 1 , N 
        a(i,j) = a(i,j) + b(i,k)*c(k,j) 
      enddo 
    enddo  
  enddo 
  ```

- Set up an automatic framework which generates unrolling and blocking strategies for matrix-matrix multiplication.

- Evaluate the efficiency of those strategies and the impact of/interaction with compiler optimizations.
Potential Topics (Benchmark Evaluation, Jan Treibig)

- Benchmark and analyse one Mini-App from the Mantevo Benchmark suite.

- Benchmark and analyse a test case from the Lonestar Benchmark suite.

Task: Classify the benchmark kernels with regard to performance patterns. Do this on multiple architectures. If applicable try to model the performance.
Potential topics

Can we do better than Intel mkl on sparse matrix vector multiply?

- Reproduce Intel’s numbers for a standard set of matrices
- Determine impact of pinning, NUMA placement, …
- Evaluate potential data transfer bottlenecks
- Compare with GHOST (our solution)
Fault tolerance algorithm (LBM or spMVM algorithms) using ULFM library

- User Level Failure Migration (ULMF)* provides extensions to MPI library to restore the communication infrastructure after process failure.

- Restructuring the algorithm to effectively utilize ULFM features
  - idle processes, checkpoint/restart

- Analysis of overheads in case of failure-free execution and process recovery cost.

- Optimization to reduce failure-free and recovery overheads.

- Scaling analysis of the fault tolerance algorithm.

Head start:

- An MPI implementation of LBM or spMVM.

http://fault-tolerance.org/ulfm/
07.10.: Introduction (GW)
14.10.: Pattern-driven performance engineering process (Jan Treibig)
21.10.: Student from last semester (*CG solver with OpenMP*)
28.10.: B. Keck (*High Performance Iterative X-Ray CT with Application in 3-D Mammography and Interventional C-arm Imaging Systems*)

- Introduction to likwid (JT/TR)
- Fooling the masses with performance results (GW/GH)

http://moodle.rrze.uni-erlangen.de/moodle/course/view.php?id=307