Energy-Efficiency Programming
Introduction

Dr. Manuel Dolz, Michael Kuhn, Dr. Julian Kunkel, Konstantinos Chasapis
Prof. Dr. Thomas Ludwig
Wissenschaftliches Rechnen
Wintersemester 2014/2015
Motivation

• High Performance Computing
  • Optimization of algorithms applied to solve scientific complex problems

• Technological advance ⇒ Performance improvement
  • More computing power and storage space
  • Multicore processors, accelerators and coprocessors

• HPC data centers ⇒ High energy consumption!
  • Growth of the Total Cost of Ownership (TCO)
  • Power wall towards exascale computing
Concurrency and efficiency

- Green500 vs Top500 (June 2014)

<table>
<thead>
<tr>
<th>Rank Top/Green</th>
<th>Site</th>
<th>Technology</th>
<th>Performance (TFLOPS)</th>
<th>Power (MW)</th>
<th>Efficiency (MFLOPS/W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/49</td>
<td>Tianhe-2</td>
<td>Intel Xeon E5 + Intel Xeon Phi</td>
<td>33,862.7</td>
<td>17.8</td>
<td>1,901.82</td>
</tr>
<tr>
<td>435/1</td>
<td>TSUBAME-KFC</td>
<td>Intel Xeon E5 + NVIDIA K20x</td>
<td>151.8</td>
<td>0.035</td>
<td>4,389.82</td>
</tr>
</tbody>
</table>
Concurrencity and efficiency

- Green500 vs Top500 (June 2014)

<table>
<thead>
<tr>
<th>Rank Top/Green</th>
<th>Site</th>
<th>Technology</th>
<th>Performance (TFLOPS)</th>
<th>Power (MW)</th>
<th>Efficiency (MFLOPS/W)</th>
<th>MW to EXAFLOPS?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/49</td>
<td><strong>Tianhe-2</strong> National University of Defense Technology</td>
<td>Intel Xeon E5 + Intel Xeon Phi</td>
<td>33,862.7</td>
<td>17.8</td>
<td>1,901.82</td>
<td>525.65</td>
</tr>
<tr>
<td>435/1</td>
<td><strong>TSUBAME-KFC</strong> GSIC Center Tokyo Institute of Technology</td>
<td>Intel Xeon E5 + NVIDIA K20x</td>
<td>151.8</td>
<td>0.035</td>
<td>4,389.82</td>
<td>230.56</td>
</tr>
</tbody>
</table>

Most powerful nuclear reactor under construction in France: Flamanville (EDF, 2017 for 10 billion €) 1650 MW
Concurency and efficiency

- Green500 vs Top500 (June 2014)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Top/Green</th>
<th>Site</th>
<th>Technology</th>
<th>Performance (TFLOPS)</th>
<th>Power (MW)</th>
<th>Efficiency (MFLOPS/W)</th>
<th>MW to EXAFLOPS?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/49</td>
<td>1/49</td>
<td>Tianhe 2</td>
<td>National University of Defense Technology</td>
<td>Intel Xeon</td>
<td>33,862.7</td>
<td>17.8</td>
<td>1,901.82</td>
</tr>
<tr>
<td>435/1</td>
<td>435/1</td>
<td>TSUBAME</td>
<td>GSIC Center Tokyo Institute of Technology</td>
<td>Intel Xeon + NVIDIA K20x</td>
<td>151.8</td>
<td>0.035</td>
<td>4,389.82</td>
</tr>
</tbody>
</table>

1MW ≈ 1M€!!

Most powerful nuclear reactor under construction in France: Flamanville (EDF, 2017 for 10 billion €) 1650 MW
Performance and efficiency trends

- Goal ⇒ Build and Exascale system ($10^{18}$ FLOPS) without exceeding 20 MW

Goal: 20MW for 1 EXAFLOP by 2020
Maintaining the improvement rate of last five years (x5) → 40 MW by 2020!!!
Performance and efficiency trends

- Goal ⇒ Build and Exascale system \((10^{18} \text{ FLOPS})\) without exceeding 20 MW

Performance-Efficiency scalar graph for the Top500 supercomputers from 2011 to 2013

Power trends of some supercomputers have almost reached the power wall being 100 away of the Exascale goal!
What we can do?

• Reduce energy consumption!
  • Costs over lifetime of an HPC facility often exceed acquisition costs
  • Carbon dioxide (CO$_2$) is a hazard for health and environment
  • Heat reduces hardware reliability
  • Scientific apps. are in general energy-oblivious

• Solutions?
  • Optimize applications from performance and energy!
    • Use hardware features for power-saving mechanisms

Energy-Efficient Programming
Energy Efficient Programming

• Methods for energy efficient programming at different levels:

  • Approaches and concepts for Energy Efficient Programming:
    • Application software efficiency
    • Operating system optimizations
    • Common problems and solutions
Energy Efficient Programming

- Computational Efficiency (Performance) ➔ Get the work done as quickly as possible!

- Energy efficiency ➔ Minimizing energy used to complete a task!

- A task does not necessarily have to be completed in a shorter time
  - however, the computer can return sooner to a low power-state!

- Approaches to increase Energy Efficiency:
  - Use of efficient algorithms and data structures
  - Multi-threading
  - Efficient use of loops
  - Vectorizing and instructions sets
  - Efficient use of programming language
  - Energy efficient libraries and drivers
Use of efficient algorithms and data structures

- Problem: Insensitive choice of algorithms and data structures may lead to significant energy wasting!

- A complete area of research in Computer Science!

- The right choice of algorithms and data structures can make a massive difference in software performance!

- Therefore, energy efficient programming requires high performance algorithms!
  - Please, complete the work faster and go to sleep! 😊

- Example 1: Sorting 200,000 double values
  - Bubble sort: $O(n^2)$ - 10,800 Joules
  - Heap sort: $O(n \log(n))$ - 7325 Joules!

- Example 2: Solving the Towers of Hanoi Puzzle (in C++)
  - Iterative version: 1656.26 Joules
  - Recursive version: 322.22 Joules

But take care! Each single algorithm should be measured to find the greenest configuration!
Efficient use of loops

- Problem: Careless programming of loops and overuse of spinning polling loops may lead important energy wastings!

- Efficient design of loops:
  - Loop unrolling: combine instructions called in multiple iter. Into a single one
  - Avoid polling loops: avoid to repeatedly check to see if a condition is true!
    - Use asynchronous/blocking methods
    - MPI/OpenMP: Use blocking waiting modes
    - Runtimes: Replace busy-waitings and use asynchronous waiting methods

  Do nothing efficiently!

The hardware does not know about the program that is being executed!

Only if processes/threads are efficiently (idle) waiting, hardware can promote processors (cores) to energy-saving states!

But take care: going to sleep for a very short period may not be efficient!
Multithreading

• **Problem:** Single threaded applications are inefficient and waste energy

• **Thread:** Smallest sequence of programmed instructions, inside a process, that can be managed by an operating system scheduler
  • *Implementations:* P-Threads, OpenMP, TBB, etc.
  • *Multithreaded programming:*
    • Parallel programming inside a process

• **Technology advance:** Multi-core processors, Coprocessors, GPU, etc.
  • Doing an efficient use of the core’s architecture we can go faster and save energy
Vectorization and Instruction Sets

- **Problem:** Use of scalar code rather than vectorizing may lead to inefficient software and waste of energy.

- **Code vectorization:** Single Instruction Multiple Data (SIMD)
  - **SIMD:** Perform different operations with different data into the same instruction!
    - Increase performance and reduce energy consumption!

- **Example for Intel:**
  - SSE (Streaming SIMD Extensions)
  - AVX (Advanced Vector Extensions) 128/256 bit instruction types
Programming language

• **Problem:** Choosing an efficient programming language may lead to significant energy waste

• **Programming languages** have a great impact on the performance and energy consumption!
  - Different causes: compiled or interpreted language, memory management, etc.
  - Depending on the level of abstraction of the language the user has more or less opportunities to introduce optimizations!

• Example with the Tower of Hanoi in different languages (Energy To Solution - ETS)

Choosing the most efficient programming language is crucial energy efficient software!
Energy Efficient Libraries and Drivers

- **Problem:** Not exploiting well-proven energy efficient solutions can lead to inefficient software!

- Select the most efficient library routines to increase performance and reduce energy consumption!
  - Look at the use of multithreading inside the routines
  - If necessary, trace them from the performance and energy perspectives!

- Example with linear algebra routines to perform matrix decompositions:

<table>
<thead>
<tr>
<th></th>
<th>LU factorization</th>
<th>Cholesky factorization</th>
<th>Reduction to tridiagonal form</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LAPACK</td>
<td>MKL</td>
<td>SMPSs</td>
</tr>
<tr>
<td>$T$ (s)</td>
<td>18.37</td>
<td>10.99</td>
<td>13.25</td>
</tr>
<tr>
<td>GFLOPS</td>
<td>38.96</td>
<td>65.13</td>
<td>54.02</td>
</tr>
<tr>
<td>$P_{\text{max}}$ (W)</td>
<td>390.70</td>
<td>385.78</td>
<td>392.81</td>
</tr>
<tr>
<td>$P_{\text{min}}$ (W)</td>
<td>301.64</td>
<td>294.37</td>
<td>328.12</td>
</tr>
<tr>
<td>$P_{\text{avg}}$ (W)</td>
<td>359.72</td>
<td>377.94</td>
<td>385.56</td>
</tr>
<tr>
<td>$P_{\text{wrk}}$ (W)</td>
<td>112.22</td>
<td>130.44</td>
<td>138.06</td>
</tr>
<tr>
<td>$E_{\text{tot}}$ (J)</td>
<td>6,608.60</td>
<td>4,155.61</td>
<td>5,109.44</td>
</tr>
<tr>
<td>$E_{\text{wrk}}$ (J)</td>
<td>2,061.48</td>
<td>1,433.54</td>
<td>1,829.30</td>
</tr>
</tbody>
</table>

**TABLE I**

*Performance, power and energy of the different implementations for the three dense linear algebra operations.*

In the example it is always more efficient to use MKL Intel library!
Minimizing data movement

- **Problem:** Unnecessary data movement may lead to energy wasting

- **Energy efficient software should minimize data movement!**
  - Move data over short distances
  - Execute tasks with fewest memory accesses

- The less energy is consumed for a memory access, the closer data is stored to CPU

**Same energy:**
1 access to RAM = 7 instructions executed in CPU = 40 cache accesses

- A solution: buffer and batch data requests in one operation/instruction
Design Energy Efficient software!

• **Engineering practices...**
  
  • **Problem:** Traditional software engineering models do not support energy efficiency as a relevant concern
  
  • **Solution:** The software life cycle should be optimized and energy efficiency be integrated as a non-functional requirement into the software engineering process model

• **Energy Efficient software is still not well perceived!**
  
  • **Problem:** Despite the fact that software can influence the energy consumption of HPC systems dramatically, the importance of software aspects of energy efficiency is still not perceived
  
  • **Solution:** Encouraging further research and better education of all stakeholders of HPC systems
Evaluation of the seminar

- A topic for each student will be assigned
  - Individual presentations:
    - 30 slides (approx.)
    - 60 minutes + discussion
    - The slides should contain notes that clarify their content

- More information at:
  - http://wr.informatik.uni-hamburg.de/teaching/wintersemester_2014_2015/energy-efficient_programming
  - Please register the mailing list: http://wr.informatik.uni-hamburg.de/listinfo/eep-1415

- Contact and supervision:
  - Dr. Manuel Dolz (manuel.dolz@informatik.uni-hamburg.de) - Coordinator
  - Michael Kuhn (michael.kuhn@informatik.uni-hamburg.de)
  - Dr. Julian Kunkel (julian.kunkel@informatik.uni-hamburg.de)
  - Konstantinos Chasapis (konstantinos.chasapis@informatik.uni-hamburg.de)
  - Prof. Dr. Thomas Ludwig (ludwig@dkrz.de)
Thanks for you attention!

Questions?