Connecting the Dots between Parallel Programming and Energy


Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Publisher rights
© 2013 The Authors

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen’s institutional repository that provides access to Queen’s research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person’s rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.
Connecting the dots between parallel programming and energy

Professor Dimitrios S. Nikolopoulos
d.nikolopoulos@qub.ac.uk
Points to get across

• PDP and HPC should be leading (not lagging) research on energy-efficient computing
• Parallel programming, runtimes, systems and energy-efficiency are connected
• What lies ahead for the parallel programmer
Acknowledgments
Our resources
Energy in HPC

Papers tagged "Power/Energy/HPC" ACM Digital Library

- 2012: 289 papers
- 2011: 257 papers
- 2010: 158 papers
- 2009: 93 papers
- 2008: 81 papers
- 2007: 59 papers
- 2006: 57 papers
- 2005: 52 papers
- 2004: 31 papers
- 2003: 25 papers
- 2002: 15 papers
- 2001: 4 papers

Friday, March 1, 13
EuroMicro PDP'13 Keynote
Leader or laggard?

- The history of BlueGene
  - Based on processors for the embedded systems market (PowerPC)
  - Pioneered “scale-out” idea, now common in datacentres
    - Many nodes with simple cores, custom interconnect
  - Dominated Top-500 list for performance
BlueGene on the Green500

BlueGene and the Green-500 List

MFLOPS per Watt

- BlueGene/L-P-Q
- Peak Green-500

Date:
- Nov-07
- Jan-08
- Mar-08
- Jun-08
- Nov-08
- Jan-09
- Mar-09
- Jun-09
- Nov-09
- Jan-10
- Mar-10
- Jun-10
- Nov-10
- Jan-11
- Mar-11
- Jun-11
- Nov-11
- Jan-12
- Mar-12
- Jun-12
- Nov-12
## Leader or laggard?

<table>
<thead>
<tr>
<th>Processor</th>
<th>Type</th>
<th>GFLOPS (32bit)</th>
<th>GFLOPS (64bit)</th>
<th>Watt (TDP)</th>
<th>GFLOPS/Watt (32bit)</th>
<th>FLOPS/Watt (64bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapteva Epiphany-IV</td>
<td>Epiphany</td>
<td>100</td>
<td>N/A</td>
<td>2</td>
<td>50</td>
<td>N/A</td>
</tr>
<tr>
<td>Movidius Myriad</td>
<td>ARM SoC: LEON3+SHAVE</td>
<td>15.28</td>
<td>N/A</td>
<td>0.32</td>
<td>48</td>
<td>N/A</td>
</tr>
<tr>
<td>ZiiLabs</td>
<td>ARM SoC</td>
<td>58</td>
<td>N/A</td>
<td>?</td>
<td>20?</td>
<td>N/A</td>
</tr>
<tr>
<td>Nvidia Tesla K10</td>
<td>X86 GPU</td>
<td>4577</td>
<td>190</td>
<td>225</td>
<td>20.34</td>
<td>?</td>
</tr>
<tr>
<td>ARM + MALI T604</td>
<td>ARM SoC</td>
<td>8 + 68</td>
<td>N/A</td>
<td>4?</td>
<td>19?</td>
<td>N/A</td>
</tr>
<tr>
<td>Nvidia GTX 690</td>
<td>X86 GPU x 2</td>
<td>5621</td>
<td>234</td>
<td>300</td>
<td>18.74</td>
<td>0.78</td>
</tr>
<tr>
<td>GeForce GTX 680</td>
<td>X86 GPU</td>
<td>3090</td>
<td>128</td>
<td>195</td>
<td>15.85</td>
<td>0.65</td>
</tr>
<tr>
<td>AMD Radeon HD 7970 GHz</td>
<td>X86 GPU</td>
<td>4300</td>
<td>1075</td>
<td>300+</td>
<td>14.3</td>
<td>3.58</td>
</tr>
<tr>
<td>AMD A10-5800K + HD 7660D</td>
<td>X86 SoC</td>
<td>121 + 614</td>
<td>?</td>
<td>100</td>
<td>7.35</td>
<td>?</td>
</tr>
<tr>
<td>Intel Core i7-3770 + HD4000</td>
<td>X86 SoC</td>
<td>225 + 294.4</td>
<td>112 + 73.6</td>
<td>77</td>
<td>6.74</td>
<td>2.41</td>
</tr>
<tr>
<td>NVIDIA CARMA (complete board)</td>
<td>ARM + GPU</td>
<td>? + 200</td>
<td>?</td>
<td>40</td>
<td>5.00</td>
<td>?</td>
</tr>
<tr>
<td>IBM Power A2</td>
<td>Power CPU</td>
<td>204?</td>
<td>204</td>
<td>55</td>
<td>3.72?</td>
<td>3.72</td>
</tr>
<tr>
<td>Intel Core i7-3770</td>
<td>X86 CPU</td>
<td>225</td>
<td>112</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>AMD A10-5800K</td>
<td>X86 CPU</td>
<td>121</td>
<td>60?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Leader or laggard?

- Is HPC just reusing solutions?
  - Processors originally designed for the mobile phones market
  - Clock gating, DVFS, device sleep states well known for 20 years
Still looking for a proper metric...

- FLOPS/Watt
- FLOPS/Watt/square inch
- Energy × Delay
- Energy × Delay × Delay
- Energy × Time to solution
- £, $, €...

- Or should we just measure execution time and forget about the rest?
Is there a role for HPC & PDP in green computing innovation?
Exascale projections

• Assume that currently most energy-efficient supercomputer sustains improvement towards an Exaflop
  – Factor of 2384× in performance
  – 202.7 MegaWatt of power

• Assume target power cap of 20 MegaWatt
  – Need two orders of magnitude improvement in FLOPS/Watt
  – What hardware devices can achieve this improvement without compromising performance?
Parallel software is (or should be) energy-efficient

• Parallel software can scale up, down and out, if written with appropriate programming models

• Scale-free software can adapt to power constraints and control energy budgets
Dynamic concurrency throttling
Concurrency throttling

- Dynamic power increases linearly with number of active cores
- Scalability curves have knees
  - Locate the knee
  - Energy-efficient parallel programs execute at the starting point of the knee

CT opportunity

Speedup

Cores
Exploiting the concurrency trade-off

- Programs execute distinct phases
  - Compute-, memory- or communication-bound
- Phase characterization indicates concurrency sweet spot
  - DCT can both improve performance and save power if the sweet spot is found
- DCT is one of several optimizations
  - Can easily complement DVFS to maximize energy-efficiency
Unified program adaptation framework

- Multi-dimensional online performance predictor [Curtis-Maury et al., TPDS08, PACT08, ICS06]
  - Statistically analyzes samples of hardware event rates collected at runtime
  - Derives predictions based on data from short execution samples
- Runtime adaptation per program phase
  - Thread-based programming model (OpenMP, Cilk)
  - DCT and DVFS based on phase scaling predictions

![Diagram showing sample and targets with green active cores at full frequency and red idle cores]
Training an adaptation model

- Map event rates collected during sample configurations to predict performance on alternate, untested configurations

\[
\text{uIPC}(t) = \text{uIPC}(s) \cdot a(e_1(s),...,e_n(s)) + \beta
\]

\[
a = \sum_{i=1}^{n} (e_i(s) \cdot x_i + y_i) + z
\]

\[
\text{uIPC}(t) = \text{uIPC}(s) \cdot \sum_{i=1}^{n} (e_i(s) \cdot x_i) + \text{uIPC}(s) \cdot \lambda + \beta
\]

- Regression, ANN, classifiers (e.g. SVM),...
- Simple models capture scalability curves well, albeit with lower absolute accuracy
Taming locality issues

• Original DCT failed to consider implications of cache refills, thread and data mapping

Example: 45% execution time variation across 85 mappings

4, 4-core nodes: 43,680 mappings.
16, 4-core nodes: 63 million mappings.
1000, 4-core nodes: $10^{43}$ mappings.
DyNUMA training using ANN

Optimize for concurrency, vertical and horizontal locality

[Su et al., IISWC12, SIGMETRICS PER12]
Modeling accuracy

![Graph showing predicted and measured wall-clock time for different categories of NPB benchmarks. The graph compares normalized prediction against measured wall-clock time, with X-axis representing normalized value (%) and Y-axis representing time (seconds).]
DyNUMA performance

- NPB.FT
- NPB.CG
- NPB.SP
- NPB.BT
- NPB.MG
- NPB.UA
- AMG.Relax
- AMG.Matvec

Wall-clock time and EDP normalized time (%) for Barcelona, Magny-Cours, and Tilera64.
Performance on TilePro64

Tile64Pro OS default Linux mapping is inefficient
More concurrency does not necessarily improve performance
Power-aware hybrid parallel programming
Energy-Aware Hybrid Programming

DVFS can save power and energy by adjusting the clock frequency according to task load imbalance.
Critical path based modeling

• Predicting time vs. predicting scaling function

\[ t_i = \sum_{j=1}^{M} \min_{1 \leq |thr| \leq X \cdot Y} t_{i,j,thr} \]

\[ t_c = \max_{1 \leq i \leq N} \sum_{j=1}^{M} \min_{1 \leq |thr| \leq X \cdot Y} t_{i,j,thr} \]
Time modeling enables slack dispersion

- Slack dispersing DCT&DVFS (Li et. al, TPDS13)

Use critical path time to determine slack (essentially imbalance)

$$\Delta t_i^{slack} = t_c - t_i - t_i^{comm} - t_{dvfs}$$

Time constraint:

$$\sum_{1 \leq j \leq M} \Delta t_{ijk} \leq \Delta t_i^{slack}$$

Energy constraint:

$$\sum_{1 \leq j \leq M} t_{ijk} f_k \leq t_i f_0$$
Consistent (or improving) energy savings with weak scaling
Where is the connection?
Energy and the programmer

• Understand performance and power implications:
  – Thread mapping
  – Scheduling
  – Data placement

• Use adaptable, scale-free programming models
  – Phase-aware optimization infeasible otherwise

• Domain-specific knowledge can save energy
  – DAG, dataflow representations have enough information for a first-order approximation of the optimization problem
  – Proactive application-specific program control likely to outperform any system-level approach
Connecting the dots

• Parallel programs are unfortunately not prepared for a power crisis
• Programming models and runtime systems still are not (yet) up for the task
  – Adapt concurrency at runtime on non-coherent many-core architectures?
  – Control power states within the runtime with input from the application?
  – Scale-free, scale-aware?
  – How much waste is there in the runtime and the OS?
We educate two kinds of (parallel) programmers

Conclusive proof it pays more to do nothing than it does to get a Ph.D.:

Average Maximum Annual Unemployment Benefit

$21,060

Average Graduate Student Stipend

$18,779

Sources: U.S. Department of Labor (via SF Chronicle), The Chronicle of Higher Education 2008-2009 survey of pay and benefits for teaching and research assistants. Unemployment benefits computed from average maximum state weekly benefits (typically 50% of base wages, capped by state) multiplied by 52 (in some cases, benefits can be extended up to 79 weeks). Academic year stipends extrapolated to 12 months.

WWW.PHDCOMICS.COM
We educate two kinds of (parallel) programmers

Joe the Plumber
Programmer
Revisiting parallel programming

- Environmentally conscious programmers know the hard limits of their algorithms given architectural or power constraints (e.g. DAG, roofline models)
- Environmentally conscious runtime systems eliminate waste
  - No more runtimes antagonizing operating systems
  - No more cross-runtime system noise
Revisiting parallel programming

- Your power budget may well be your most critical resource
  - Awareness of this issue must somehow become part of programmer education
  - Neither Joe nor graduate students are ready for this
  - Hardware and system software will not solve the problem by themselves, without a steep performance penalty

- Defensive parallel programming
  - Scale-free, react to power caps
  - Explicit resource proportioning
More information

http://www.qub.ac.uk/research-centres/HPDC/
Connecting the dots

- A power-aware parallel program of the future:
  - adapts concurrency
  - eliminates waste (idle time, redundant communication)
  - controls component power at a fine granularity (e.g. per task)

Manousakis et al, [SBAC-PAD12]