Iso-energy-efficiency: an approach to power constrained parallel computation

Shuaiwen Song, Chun-Yi Su, and Kirk W. Cameron, SCAPE lab, Virginia Tech

Rong Ge, Marquette University

Abhinav Vishnu, Pacific Northwest National Laboratory
Why is power/energy efficiency important for HPC community?

- Increasing system size and number of cores on single chip increases demand for power/energy.

- High operating power reduces system reliability and increases the total operating cost.

- Future exascale machines require high system energy efficiency in order to achieve greater performance.
Problem statement

- Energy efficiency is now a key constraint in HPC system design.
- Past system level approaches focus on power mode **predictor and controller** design:

  - **Key limitation:**
    We lack a fundamental understanding of the impact of system-level power management on application performance.
    (i.e. we need to predict energy use)
Approach

- Build analytical models of both performance and power to gain insight of power-performance tradeoffs.

Challenges:

- Resulting model must be Practical.
- Resulting model must be Accurate.
- Resulting model must be Useful.
Related work

- Direct measurement
  e.g. PowerPack (Ge, et al)

- Simulations
  e.g. Wattch (Princeton) and SimplePower (PSU)

- Analytical modeling
  e.g. Power-aware speedup (Ge et al) and statistical power modeling using regression (Bellosa et al).
Iso-energy-efficiency basics

- Iso-efficiency function (Grama et al, 1993)
  Maintain performance efficiency when system (e.g. nprocs) and applications change (e.g. problem size)

- Power-aware applications/systems

  Similarities:
  - Processor (cores) number affect performance
  - Problem sizes affect performance

  Differences:
  - Energy not considered
  - Power modes also affect performance and energy

- Iso-energy-efficiency (predict energy use)

\[
EE = \frac{E_1}{E_p} = \frac{E_1}{E_1 + E_o} = \frac{1}{1 + \frac{E_o}{E_1}}
\]
Iso-energy-efficiency Model

- General form of our Iso-energy-efficiency model:

\[
EE = \frac{E_1}{E_p} = \frac{E_1}{E_1 + E_o} = \frac{1}{1 + \frac{E_o}{E_1}}
\]

**EE**: system-wide energy efficiency

**\(E_1\)** (baseline): total energy consumption of sequential execution on one processor

**\(E_p\)**: the total energy consumption of parallel execution for a given application on \(p\) parallel processors

**\(E_o\)**: the additional energy overhead required for parallel execution and running extra system components
Model Methodology

- **Performance model:** Build accurate, practical performance model capturing key application/system parameters.

- **Power model:** Build accurate, practical power model capturing key application/system parameters.

- **Energy model:** Combine models to explain causal relationships between application and system parameters with regard to energy efficiency.

- **Key parameters:** Need to explain effects of system scale, power mode, and workload characteristics.
Performance, Power and Energy Models

- **Performance Modeling** using on-off chip workload characteristics:

\[
T = T_w + T_0 = T_w(f_{on}) + T_w(f_{off}) + T_0 \\
T = \alpha(W_c t_c + W_m t_m + T_{10} + T_o)
\]

- System component level **Power Modeling**:

\[
P = mf^\gamma + n \quad (\gamma \geq 1, m \text{ and } n \text{ are coefficients})
\]

We use PowerPack data to tune power consumption parameters

- Component-wise **system energy modeling**:

\[
E = E_{cpu} + E_{mem} + E_{IO} + E_{other} \\
= \Delta E_{cpu} + \Delta E_{c-idle} + \Delta E_{mem} + \Delta E_{mem-idle} + \Delta E_{IO} + E_{IO-idle} + E_{other} \\
= TP_{total-idle} + \Delta E_{cpu} + \Delta E_{mem} + \Delta E_{IO}
\]
Too many details for this talk…

- Lots of parameters & formulae…

- see paper for details…

Stuffed !
Iso-energy-efficiency Model

- We use energy models for Eo and E1 to solve for iso-energy-efficiency:

\[ EE = \frac{1}{1 + \frac{E_o}{E_1}} = \frac{1}{(1 + \frac{\alpha T_o P_{total-idle} + W_{co} t_c \Delta P_c + W_{mo} t_m \Delta P_m}{\alpha T_1 P_{total-idle} + W_c t_c \Delta P_c + W_m t_m \Delta P_m})} \]

\[ T_1 = (W_c t_c + W_m t_m) \quad T_o = (W_{co} t_c + W_{mo} t_m + \sum_{i=1}^{p} T_{net_i}) \quad (1 \leq i \leq p) \]

- Network model can be replaced by more sophisticated model according to specific application.

- I/O component can also be addressed in the model if the application is I/O intensive.
Example of Input Variable Impacts on Model

Power Profiling of MPI_FFT program in HPCC Benchmark
Iso-energy-efficiency is Practical.

- All model parameters are measurable.
Iso-energy-efficiency is Accurate.

- Validation on *Dori* cluster (shown) and SystemG supercomputer (not shown)

- All the applications run on 8 nodes under same CPU clock frequency (1.8 GHz).
- Model accuracy for all the benchmarks (and all frequencies) are over 95%.
Iso-energy-efficiency is Accurate.

- System scale accuracy on SystemG cluster (InfiniBand)

Absolute Average Error Rate on SystemG with node number scales from 1 to 128

- The average error rate of three applications under workload Class=B and two frequency levels (2.4 and 2.8 GHz) with node number scaling from 1 to 128.

A little higher error for CG caused by modeling memory behavior of CG. This will be improved in future work.
Iso-energy-efficiency is useful.
NAS PB Application Energy Scalability Analyses

- We chose a spectrum of applications to evaluate the energy scalability of different execution patterns
  - **FT**: communication intensive with dominating communication overhead for the all-to-all phases.
  - **CG**: higher computation to communication ratio than FT, but less than EP.
  - We’ve also analyzed EP and HPL (see papers)
Platform

- Systems used in our experiments.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>System size</th>
<th>Processor</th>
<th>Memory</th>
<th>L1 cache</th>
<th>L2 cache</th>
<th>Interconnection</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemG</td>
<td>325 Mac Pro nodes</td>
<td>two quad-core 2.8 GHz Intel Xeon processor</td>
<td>8GB RAM</td>
<td>32KB</td>
<td>Shared, 6MB</td>
<td>Mellanox 40Gbytes/sec InfiniBand</td>
<td>2.8 and 2.4 GHz</td>
</tr>
<tr>
<td>Dori</td>
<td>8 blades</td>
<td>AMD Opteron dual core dual processor</td>
<td>6GB RAM</td>
<td>64KB</td>
<td>Shared, 1MB</td>
<td>1Gbytes/sec Ethernet</td>
<td>1.8, 1.6, 1.4, 1.2, 1.0 GHz</td>
</tr>
</tbody>
</table>

- We focus on SystemG to verify Energy Scalability Analysis.
Predicting Energy Scalability for 3-D FFT

FT’s system-wide energy efficiency with p and n as variables

FT’s system-wide energy efficiency with p and f as variables

Note: accuracy of the predicted EE has been validated to be within 5% error rate.
Key results for FT

- **Problem size scaling under fixed frequency is effective in maintaining overall system energy efficiency for FT.**
  - Model shows that increasing problem size can stress all variables in the model which are functions of problem size, so that both E1 and Eo increase.

- **CPU frequency scaling: lowering CPU frequency under fixed problem size can only slightly improve the energy efficiency for FT.**
  - The effects of CPU clock frequency on on-chip workload diminish while scaling up system size.

- **Conclusion: Energy efficiency can be maintained by scaling number of processors and problem size simultaneously.**
Predicting Energy Scalability for CG

CG’s system-wide energy efficiency with p and n as variables

Note: accuracy of the predicted EE has been validated to be within 5% error rate.
Key results for CG

- **Energy efficiency decreases with system scale.**
  - Model shows parallel energy overhead $E_o$ increases due to parallel performance overhead and supporting power for additional components (e.g. mem, network).

- **Energy efficiency can be maintained/improved by scaling up problem size.**
  - Model shows that increasing problem size can stress all variables in the model which are functions of problem size, so that both $E_1$ and $E_o$ increase.

- **Applying higher frequency will improve system-wide energy efficiency while system size scales up.**
  - Effects of frequency on on-chip workload doesn’t diminish as fast as $FT$. 

Virginia Tech
Invent the Future
Conclusions I

- Iso-energy-efficiency is practical
  - All parameters measurable directly

- Iso-energy-efficiency is accurate
  - Accurate within 5%

- Iso-energy-efficiency is useful.
  - Predicts total system energy consumption
  - Identifies causal energy efficiency relationships, what-if analyses
Conclusions II

- EE decreases with system scale for FFT and CG (but not EP)
  - Model shows that parallel energy overhead $E_o$ increases due to parallel performance overhead and supporting extra parallel system components.
  - It is likely that apps with similar comp/comm ratio present the similar energy efficiency scalability while scaling up system size.

- Problem size and frequency can be adjusted to maintain EE at scale
  - Problem size scaling can stress all variables in the model which are functions of problem size so that both $E_1$ and $E_o$ increase.
  - We can use the model to identify pairs of problem and system size that exhibit the similar energy efficiencies.
  - Model shows frequency scaling can help maintain energy efficiency at times, but depends on the effects of frequency on on-chip workload while system scales.
Current Limitations

- Model limitations
  - not fully automatic for model building.
  - initial focus on homogeneous systems.
  - improved accuracy of CG memory model.
- Experimental limitations
  - Number discrete power modes could improve EE (e.g. Problem size is basically continuous)
Future work

- Automate application model identification
  - First step is to simplify the model with less parameters.

- Extend to heterogeneous systems such as GPGPU
  - First step is to accurately measure the power consumption of GPU’s internal major components.

- Revisit memory models

- Experiment with other power modes and systems
Thank you for listening!

Contact information:

Shuaiwen Song

Email: s562673@cs.vt.edu

SCAPE lab Link: http://scape.cs.vt.edu/
### Table 1 Machine dependent parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_c$</td>
<td>Total on-chip computation workload</td>
</tr>
<tr>
<td>$W_m$</td>
<td>Total off-chip memory access workload.</td>
</tr>
<tr>
<td>$W_{so}$</td>
<td>Total parallel computation overhead</td>
</tr>
<tr>
<td>$W_{mo}$</td>
<td>Total number of memory access overhead in parallelization</td>
</tr>
<tr>
<td>$M$</td>
<td>Total number of messages packaged in parallelization</td>
</tr>
<tr>
<td>$B$</td>
<td>Total number of bytes transmitted</td>
</tr>
<tr>
<td>$P$</td>
<td>Number of homogeneous processors available for computing the workloads</td>
</tr>
<tr>
<td>$N$</td>
<td>Workload or total amount of work (in instructions or computations)</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Corrector factor including components such as overlap among computation, memory access and network transmission, additional cost by code scaling, etc.</td>
</tr>
<tr>
<td>$\tau_o$</td>
<td>Total overhead time due to parallelism</td>
</tr>
<tr>
<td>$T_1$</td>
<td>Total execution time of an application running on a single processor</td>
</tr>
</tbody>
</table>

### Table 2 Application dependent parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_c$</td>
<td>$\frac{E_{Plcm}}{f}$ [28], Average time per on-chip computation instruction (including on-chip caches and registers)</td>
</tr>
<tr>
<td>$t_m$</td>
<td>Average memory access latency</td>
</tr>
<tr>
<td>$t_{msg}$</td>
<td>Average start up time to send a message</td>
</tr>
<tr>
<td>$t_{byte}$</td>
<td>Average time of transmitting a 8-bits word</td>
</tr>
<tr>
<td>$T_{IO}$</td>
<td>Total I/O access time</td>
</tr>
</tbody>
</table>

#### Power-related

- $P_{c-on}$: Average CPU power in running state
- $P_{c-idle}$: Average CPU power in idle state
- $\Delta P_c$: $P_{c-on} - P_{c-idle}$
- $P_{m-on}$: Average memory power in running state
- $P_{m-idle}$: Average memory power in idle state
- $\Delta P_m$: $P_{m-on} - P_{m-idle}$
- $P_{IO-on}$: Average IO device power in running state
- $P_{IO-idle}$: Average IO device power in idle state
- $\Delta P_{IO}$: $P_{IO-on} - P_{IO-idle}$
- $P_{other}$: Average sum of other devices' power such as motherboard, System/CPU fans, NIC, etc.
- $P_{total-idle}$: Average system power on idle state
- $f$: The clock frequency in clock cycles per second
Machine dependent vector

- Variables in machine dependent vector can be modeled as functions of machine related parameters such as frequency $f$ and interconnect $I$:

$$V_{\text{machine}}(f,I) = (t_c, t_m, t_{msg}, t_{Byte}, P_{\text{total-idle}}, \Delta P_c, \Delta P_m)$$

- For example:
  
  $t_c$ can be described as $\frac{CPI_{on}}{f}$
Application dependent vector

- Variables in application dependent vector can be modeled as functions of application related parameters such as problem size $N$ and system size $p$:

$$V_{app}(p, N) = (W_c, W_m, W_{co}, W_{mo}, M, B)$$

- These variables are more affected by application related parameters.
- For example,

$$W_c, W_m$$
$$M, B$$

- can be modeled as a function of $N$

- can be modeled as a function of $N$ and $P$

- For the high level meaning of the application vector:

  We can try to **optimize** the application (such as parallel algorithm) in order to improve energy efficiency.
Another example, corrector factor $\alpha$ in application vector addresses:
   a) computational overlap
   b) parallel and synchronization overhead
   c) extra computation overhead

$\alpha$ value varies not only by application and compiler setting, but also by underlying architecture.