HPC systems energy efficiency optimization thru hardware-software co-design on Intel technologies

Andrey Semin
HPC Technology Manager
Intel Corporation, EMEA
Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.

Intel may make changes to specifications and product descriptions at any time, without notice.

This document may contain information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Wireless connectivity and some features may require you to purchase additional software, services or external hardware.

Nehalem, Penryn, Westmere, Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.


*Other names and brands may be claimed as the property of others.
Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101
Agenda

• Exascale challenges
• Hardware/software co-design
• Total vs. Local Energy Optimization
• Putting it all together
Challenges Need to Be Addressed to Reach Exascale

Energy Per Operation
Associated with Computation, Data Transport, Memory, and other overheads

Extreme Concurrency and Locality
Associated with programming billions of threads

Resiliency
Associated with growth in component count, lower voltages, security, etc

Memory/Storage Capacity, Bandwidth and Power
Associated with inability of current technology trend to meet requirements

The answer is: Hardware/Software Co-Design

- **Programming Language** should provide locality Information
- **Compiler** should provide Data Access Patterns, Memory Footprints, and Checkpointing Regions
- **Algorithms** Reducing Data Movement
- **Perf. tools** use the information from HW to guide future actions
- **Runtime** should guide Hardware on Caching, Prefetching, and Power Control
- **Hardware** should provide performance counter information such as memory access patterns without high overhead

**Energy Efficiency is best achieved with a HW-SW Co-Design**
Hardware/software co-design means meeting system-level objectives by exploiting the synergism of hardware and software through their concurrent design.

Co-design problems have different flavors according to the application domain, implementation technology and design methodology.

Optimizing Energy Consumption and Efficiency

Total vs. Local Energy Optimization

Power Per Unit

- < 50 Watts
- ~50-200 Watts
- Kilowatts
- Megawatts

App Workload

1-2% Data Center Power

Processes, Memory
8-10% Data Center Power

IT Equipment, Systems
30-40% Data Center Power

Changes Here Affect Data Center Efficiency

Changes Here Affect Data Center Power Consumption
Terascale Power Use Today  (Not to Scale)

- **Compute**: 200 W
- **Memory**: 0.1 byte/flop @ 1.5 nJ per byte
- **Comm.**: 100 pJ comm. per flop
- **Disk**: 100 W
- **Control**: 7.5 nJ/instruction @ 0.2 instruction/flop

- **Total Power**: ~5 kW
- **Heat removal**: (all levels, chip to facility) 40% of total power consumed
- **Power supply loss**: 19% (81% efficient power supplies)

Derived from data from S. Borkar and J. Gustafson
Let’s See that Drawn to Scale…

A SIMD accelerator approach gives up Control to reduce wattage per TFLOPS. Which can work, for some applications that are very regular and SIMD-like: vectorizable with long vectors.
**Cooling**

**PUE optimization thru thermal density and inlet coolant temperature increases**

CRAC, 8 & 16°C inlet water in CRAC, 22°C inlet air, overflow ca. 1m/s

- Inlet water is still too cold for all-year-round free cooling
- PUE improvements flattens with density increases over 20kW/rack
Cooling

Pushing the limits further...

• Increase density and inlet temperature:
  – Can reach over 60KW/rack with air cooling
  – Can increase datacenter temperature to high 20°C
  – Increased thermal density helps increase power delivery efficiency

• But...
  – PUE drops at very high thermal densities with air-cooling
    – Need high Δ between hot and cold isles
    – Need increase fan speeds to maintain high air flow
  – PUE drops as Si temperatures increase
    – Higher leakage
    – Throttling
Why Liquid Cooling?

Heat Capacity of this much air

Heat Capacity of this much water

Fans move energy less efficiently

Source: Overview of Liquid Cooling Systems, Peter Rumsey, Rumsey Engineers
Available at: hightech.lbl.gov/presentations/Dominguez/5_LiquidCooling_101807.ppt
HPC Solutions with Direct Liquid Cooling

- IBM HS22 Blade (liquid)
- QPACE
- IBM BlueWaters
- RSC
- EUROTCH
- IBM Power 575
- ICEOTOPE
- FUJITSU

Not complete list

*Other names and brands may be claimed as the property of others.*

Energy: Efficiency, Environment & Performance
Measured PUE

Cluster with 96 nodes: 2x Xeon X5680 (liquid) and X5670 (air), 24GB RAM, QDR IB fat tree.
Workload: HPL

- 30TF/rack peak
- 100KW/rack
- Rack-level PSU: 94% efficiency

- "hot"-liquid
- Free cooling

Liquid cooling delivers great PUE

Source: Intel internal (air) and RSC (liquid) measurements. Evaluated systems are comparable in performance.
Energy-aware Programming

Measuring Efficiency

• Efficiency = Useful Work / Resource Consumed
  – Increasingly applied to Energy Efficiency:  
    \textbf{Energy Efficiency} = \text{Useful Work} / \text{Energy Consumed}

• Seems simple, except:

  \textbf{There is no universal quanta of work}

• Also need:
  – Easy way to measure energy consumption and correlate it with system and application activity
  – Vendor-agnostic tools to ensure openness
  – Low-cost toolkit to encourage adoption

\textbf{You Can’t Manage What You Can’t Measure}
Energy-aware Programming

Intel® Energy Checker (EC) SDK

Measure energy consumed for a workload
- Define “Work” to be measured
- Instrument code
- Collect data and compute Energy Efficiency
- Analyze
  - System productivity
  - Application’s energy profile

Write energy-aware SW with minimal effort, focusing on relevant energy heuristics

Not included in EC SDK:
- Use Recommended models or follow User Guide to enable another model
Energy-aware Programming
Example of Energy Checker Data Collected

SW phase helps to segment SW activity

Energy per SW phase as computed and exposed by SW

ESRV Energy data

---

1 Data was collected using pl_csv_logger (a tool shipped with the Intel® EC SDK). Instrumented from outside (scripting tools) and VBA [no source code access].
### Energy Cost by Operation Type

<table>
<thead>
<tr>
<th>Operation</th>
<th>Approximate energy consumed today</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit multiply-add</td>
<td>200 pJ</td>
</tr>
<tr>
<td>Read 64 bits from cache</td>
<td>800 pJ</td>
</tr>
<tr>
<td>Move 64 bits across chip</td>
<td>2000 pJ</td>
</tr>
<tr>
<td>Execute an instruction</td>
<td>7500 pJ</td>
</tr>
<tr>
<td>Read 64 bits from DRAM</td>
<td>12000 pJ</td>
</tr>
</tbody>
</table>

- Notice that 12000 pJ @ 3 GHz = 36 watts!
- A solution: drop the memory speed, but the performance of HPC applications will be dropped proportionately!
- Larger caches actually reduce power consumption.

**Application software should also be energy aware**
System Control

Intel Intelligent Power Node Manager

- Dynamic power limits
- System, rack, datacenter monitoring and management
Optimizing Energy Consumption and Efficiency

Putting it all together

Power Per Unit

- < 50 Watts
- ~50-200 Watts
- Kilowatts
- Megawatts

Changes Here Affect Data Center Efficiency

Changes Here Affect Data Center Power Consumption

Optimizing Legacy Datacenter Infrastructure

Platform Configuration & Operation

Component Choice & Local Automatisms

Energy-aware programming

- Energy:
- Efficiency:
- Environment:
- Performance:
Summary

• Build your energy savings pyramid from bottom to top:
  – Each homework done in the lower layer maximizes leverage in the upper layer.

• Typical data center infrastructure offers many improvement opportunities for rather simple best practices or advanced techniques

• Application software should also be energy aware
  – Energy-aware programming needs to be based on consistent measurement of executed work and Watt-hours consumed

Energy Efficiency is best achieved with a HW/SW Co-Design
Energy-aware Programming

The Intel® Energy Checker SDK

• Intel® Energy Checker SDK is a series of routines that can be integrated into any application to write out counters in a standard manner
  – No external libraries or run-time software must be installed with the application; this is standalone
  – Becomes part of the application, not the system

• These counters can be easily read and aggregated to report the productivity of a system
• These counters can be used during benchmarking
• They are also lightweight enough to be used in production systems with negligible impact on performance
Energy-aware Programming

Major Pieces of the Intel® EC SDK

- Windows®, Linux®, Solaris® 10, and MacOS® X support
- Core API (C/C++, C#, and Java® interfaces)
- PL Scripting Tools
- ESRV/TSRV energy/temp monitoring tools
- Windows interoperability tools
- GUI Monitor and CSV Logger
- Additional sample code
- Installation code
- Documentation (3 manuals)

- See http://whatif.intel.com for details.