Node Modeling and Simulation at Cray

Jim Kohn
Senior Principal Engineer
Application Modeling Group

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

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

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

Cray hardware and software products 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.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

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

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Other names and brands may be claimed as the property of others. Other product and service names mentioned herein are the trademarks of their respective owners.

Status of Node Modeling at Cray

- **Focus is on modeling future "what if" architectural designs**
  - ISA extensions, memory systems, NICs, expanded vector technology, thread synchronization, etc.
- **Model performance of hybrid parallel apps**
  - MPI + OpenMP + Vector
  - Up to 256 threads on a node (SoC)
  - Simulate and model real apps at a net rate of 10-20 MIPS
  - Parameterized to model sensitivity to varying architectural features
  - Characterize how real at-scale apps perform on a given target architecture
- **Adding power modeling and management (DVFS, etc.)**
- **Modeling alternatives to improve fine-grain threading**
- **Validation kernels with known performance attributes are used to verify accuracy against hardware or calculated performance**

What's Lacking?

- **Node-side interactions of network components difficult to model**
  - We could model on-node NICs, injection bandwidth, performance of various message sizes, message matching cost
  - Could maybe extrapolate on-node NIC reference patterns to a larger multi-node network
  - Model how a larger network might affect node performance
- **Need representative Mini-Apps**
  - Huge representative memory footprints to study tiering options for HBM, NVM
  - Wider vector loop trip counts to study power options, cache performance, fine-grain threading, etc.
  - Hybrid MPI + OpenMP parallel implementations
  - Wider and deeper threading or tasking implementations
  - Reflective of future application development
- **Need parameterized power models to study *future* architecture designs rather than measuring only current implementations.**

Modeling the TaihuLight Node

- **Node = 4 clusters – each with**
  - 1 master (supervisory) core with L1 and L2 cache, 16 DP flops/cycle
  - 64 slave (processing) cores
    - 64K scratchpads (instead of cache hierarchy)
    - 8 DP flops/cycle
  - 1 MC, 128-bit DDR3 at 2133 MHz => 34 GB/s
  - 8GB memory bank (memory partition)
  - Uses NoC (network-on-chip) vs. cache hierarchy
- **At 1.45 GHz => 3.06 peak TFLOPS (includes master cores) (40,960 nodes => 125.4 peak PFLOPS)**
- **Parallelism based on MPI and OpenAcc 2.0**
- **With a few modifications we could model TaihuLight with its known features (assuming we have apps built for it)**

TaihuLight Implications

- **Path to the first exascale system in 2020?!**
  - Potentially, but only for HPL-like apps (93 sustained PFLOPS = 74% of peak)
  - BUT, for HPCG achieves only 0.3% of peak
  - Which of these do DOE applications look more like?
    - Our experience is some of each, but more HPCG
- **Software implications:**
  - No hardware cache hierarchy/coherence means software must:
    - Very carefully align, block, and migrate data to/from core scratchpads
    - Generate explicit memory barriers
    - Be aware/manage data cached in the master core
  - Responsibility is mostly in the applications - compiler has insufficient info to manage fine-grain coherence and coarse-grain might kill performance
  - OpenAcc has the directives to manage memory
- **How many apps are designed to run well without cache or cache coherence?**
- **What effort is required to restructure them if they aren't?**

Open Questions

- **Rather than improving modeling accuracy or only measuring power and performance, could we generate metrics to measure an app's "fit" on any proposed architectural design?**
  - For example, HPL is a fit on TaihuLight, but HPCG is not. How would we characterize ***why*** HPL is a fit and HPCG is not besides performance?
- **How much effort and lead time is necessary to move DoE apps to more radical architectures (e.g., memory tiers, scratchpads instead of caches, processing-in-memory, very wide vector units, support for wider/deeper levels of threading, etc.)?**
  - Is it even practical to consider such architectures in the exascale timeframe?
- **What should a system vendor like Cray be providing to scientists to support the *development* of exascale applications?**