Quantifying the Dominance of Leakage Energy in Large-Scale System Caches

Aditya Deshpande, Jeffrey Draper {adeshpan, draper} @ ISI.EDU

Marina Group :: Information Sciences Institute :: Ming Hsieh Department of Electrical Engineering :: University of Southern California, Los Angeles, CA, USA

Motivation

- Energy consumption is a big concern in HPC systems.
- Energy usage has been identified as a key barrier to exa-scale system deployments.
- HPC system energy includes four major components whose interactions affect total system energy —
  - Core energy
  - Memory energy
  - Network energy
  - Energy of I/O peripherals
- The first step in reducing total system energy is to identify how and where energy is consumed in HPC application execution.
  - No such study found in prior research
- Our focus is on the energy consumption in the memory subsystem during an application's execution (in particular for the on-chip caches present in the processors).

On-chip caches make an ideal candidate for energy studies from **S**ystem, **A**rchitecture and **T**echnology perspectives —

S
- Large number of processing cores
- Increased parallelization in application code (Synchronization leads to increased aggregated runtime across all the cores)

A
- Trend of having large on-chip caches in a deep hierarchy to increase performance of multi-core systems

T
- Leakage in transistors increasing with every generation of transistor scaling
- Caches occupy up to 75% of chip area & up to 50% total chip power

We compute dynamic and static (leakage) energy dissipated in the various levels of on-chip caches for various applications when they are executing either in a single-thread environment or distributed system environment.

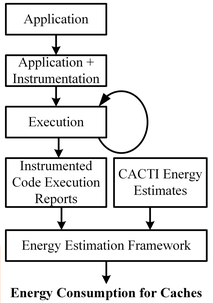
In this work we show results for the L1 data and instruction cache energy consumption across a wide range of HPC applications.

Our Modeling Scheme

Steps in energy modeling in caches

- Application is compiled.
- Application executable is instrumented using performance tuning tools for measuring various Hardware Performance Counters.
- The instrumented code is executed multiple times to record various hardware performance counter information.
- The recorded counter values along with cache access energy info from CACTI are plugged into our energy estimation framework.
- The energy estimation framework essentially multiplies cache energy values from CACTI with the number of accesses to a particular level of cache as recorded from instrumented code execution.

This scheme combines cache activity information obtained from performance tuning tools with an accurate cache-energy modeling tool to generate fast and accurate energy consumption information for various levels of caches with minimal intervention and substantially lower overheads.



System Information

Benchmarks considered in this work

SCALAPACK – PDGEMM (a simple compute kernel)
- A parallel implementation of Double Precision General Matrix Multiply operation routine from the Scalable Linear Algebra Package

NAS Parallel Benchmarks (NPB)
- Benchmarks derived from computational fluid dynamics (CFD) applications & mimic the computation and data-movement in CFD applications
- Benchmarks considered: CG, EP, FT, IS, LU, and MG

Mantevo Miniapps/Real-world Applications
- **HPCCG** — a simple proxy for unstructured implicit finite element applications (a linear system solver using conjugate gradient method)
- **MiniGhost** — a proxy for finite difference application, implements a difference stencil across a homogenous 3-D domain
- **MiniFE** — a proxy for implicit unstructured finite element codes
- **GTC** — used for Gyro-kinetic Particle Simulation of Turbulent Transport in Burning Plasmas

Machine Info
- Experiments were conducted on Hopper, a Cray XE6 machine located at National Energy Research Scientific Computing Center (NERSC).
- A compute node on Hopper uses AMD 'Magny-Cours' processors built using 45nm process technology.
- Each core has its own L1 (64KB-D, 64KB-I), 512KB unified L2 cache, and six - cores share a 6MB L3 cache.

CACTI Configuration

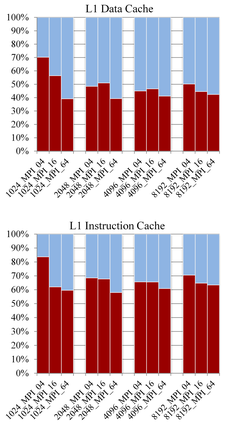
L1 Data/Instruction cache modeled for estimating dynamic-energy/access and leakage energy with the following parameters –
- Associativity = 2, Block Size = 64B, Capacity = 64KB
- Feature size = 45nm and Operating temperature = 350K
- Cache is modeled as a fast cache with design weights set accordingly.

We evaluate these applications to measure dynamic and leakage energy in L1 data/instruction caches

Results

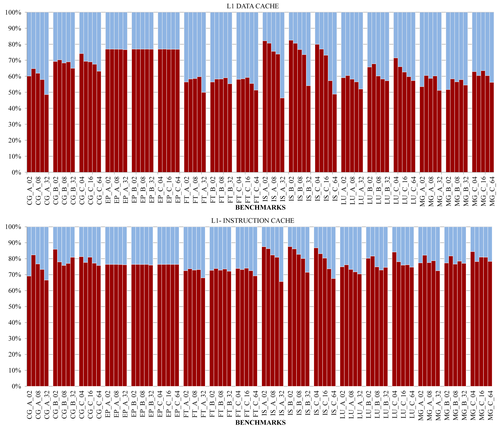
Dynamic Energy / Leakage Energy

SCALAPACK– PDGEMM



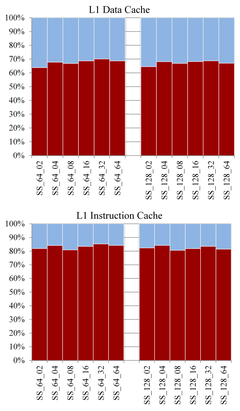
- Leakage energy varies between 40-70% for D-cache and 60-80% for I-cache
- For matrix size 1024 running 4-MPI nodes, leakage is maximum as data communication overhead outweighs benefits of parallelism, execution time ↑, leading to leakage energy ↑
- As # MPI nodes↑, cache accesses ↑, causing an increase in total energy

NAS Parallel Benchmarks (NPB)



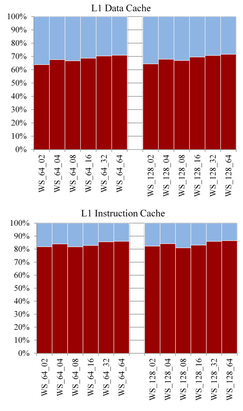
- # MPI nodes used for Class A/B and Class C problem sizes varies between 2-32, and 4-64 respectively
- As # MPI nodes ↑:
  - Performance ↑ for apps sensitive to computation speed and memory size (More cache → better performance → ↓ leakage energy as execution time ↓) Ex. CG, IS
  - % leakage component of total cache energy ↓, total energy required for same computation ↑↑ significantly
  - For applications (Ex. EP) which require very little inter-process communications (overhead of MPI communication and synchronization is avoided), leakage energy remains constant for total cache energy for both data/instruction caches.

Strong Scaling Experiment (HPCCG)



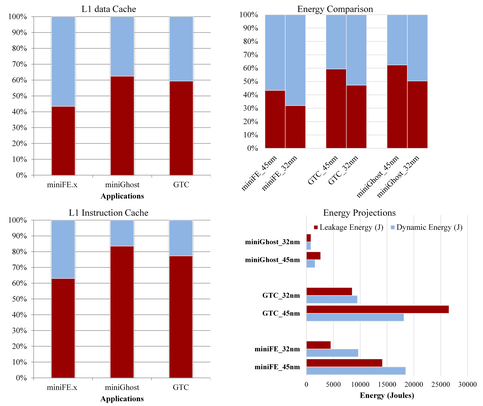
- Global problem size kept constant at 64x64x1024 and 128x128x128 with processors varied between 2-64 at powers of 2
- As # MPI nodes ↑:
  - % leakage energy of total cache energy ↑ — as both problem size fits into cache, no major reduction in leakage energy was observed.
  - Total energy ↑ with increase in leakage component more pronounced than dynamic energy increase (due to MPI communication overheads)

Weak Scaling Experiment (HPCCG)



- The local problem kept constant at 64x64x64 and 128x128x128 with the number of processors used for scaling varying between 2-64 at powers of 2
- As HPCCG code has irregular memory access patterns, leakage energy ↑ as global problem size ↑ with nodes ↑
- With more nodes, MPI synchronization overhead for data-communication affects total execution time which in-turn affects (increases) leakage energy component.

Miniapps / Real-world Applications



- All three apps have very high D-cache hit rates: MiniFE (99.83%), MiniGhost (99.4%), GTC (97.8%). Still leakage energy accounts for 40-60% of total cache energy.
- To study the effect of transistor manufacturing and technology changes we estimate energies for caches at 32nm technology —
  - The leakage energy reduces as a percentage of total energy in 32nm technology.
  - This reduction in leakage energy occurred due to the use of high-k metal gate devices in semiconductor manufacturing processes technology. This use of high-k metal gate devices provided a one-time improvement; the leakage component is expected to increase with even smaller geometries.
  - Even with use of high-k metal gate devices, leakage still ranges between 30-50%.

Conclusions

- Cache utilization has a strong impact on the leakage energy component of total cache energy
- For applications with irregular memory access patterns, by increasing the number of nodes, cache utilization improves, thus improving performance, but the aggregated total cache energy increases enormously
- Trade-off that application developers need to consider — performance improvement either by increasing number of nodes for computation (requires more energy!) or by optimal use of resources and using least possible number of nodes (requires less energy, more effort!)
- The trade-off between performance improvement & reduction in execution time is critical for the cache leakage energy component
- Our results show that leakage energy is the dominant form of energy dissipation in smaller L1 data/instruction caches
  - L2 and L3 caches are order of magnitude larger than L1 caches; thus, the leakage energy is expected to be an even more dominant part of their energy consumption
- Leakage energy optimization should be targeted at design time for efficient energy management in memories
  - Architectural innovation is needed to address this challenge!

References:

1. Peter Kogge et al. Exascale computing study: Technology challenges in achieving exascale systems.
2. ITRS Roadmap. http://www.itrs.net/
3. Naveen Muralimanohar et al. CACTI 6.0: A Tool to Model Large Caches, Technical Report, HP Laboratories, April 2009.
4. S. Browne, J Dongarra, et al. A portable programming interface for performance evaluation on modern processors. The Int. Journal of High Performance Computing Applications, 14:189–204, 2000.
5. Cray Performance Measurement and Analysis Tools. http://docs.cray.com/
6. L. S. Blackford et al. ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers - Design Issues and Performance. In Supercomputing, 1996. Proceedings of the 1996 ACM/IEEE Conference on, pages 5–5, 1996.
7. D. H. Bailey et al. The NAS Parallel Benchmarks. Technical report, The International Journal of Supercomputer Applications, 1991.
8. Mantevo Project. http://www.mantevo.org/
9. Michael A. Heroux et al. Improving performance via mini-applications.

Acknowledgements:

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

USC Viterbi School of Engineering — Ming Hsieh Department of Electrical Engineering

USC Viterbi School of Engineering — Information Sciences Institute