Thrifty: An Exascale Architecture for Energy Proportional Computing
Final Progress Report

PI: Josep Torrellas (torrellas@cs.uiuc.edu)

DOE Award #: DE-FC02-10ER25992/DE-SC0005512
Recipient Institution: University of Illinois at Urbana-Champaign
Termination Date: August 31, 2014

1. Introduction

The objective of this project is to design different aspects of a novel exascale architecture called *Thrifty*. Our goal is to focus on the challenges of power/energy efficiency, performance, and resiliency in exascale systems. The project includes work on computer architecture (Josep Torrellas from University of Illinois), compilation (Daniel Quinlan from Lawrence Livermore National Laboratory), runtime and applications (Laura Carrington from University of California San Diego), and circuits (Wilfred Pinfold from Intel Corporation). More specifically, the aims of this project are:

Power/Energy Efficiency: Attain efficiency gains over current computer systems by developing:
• Novel circuits and architecture technologies for Near-Threshold Voltage Computing (NTC) and fine-grain management of static and dynamic power.
• A power/energy-aware compiler for Fortran and C based on ROSE that uses static and dynamic code analysis to drive the power-management hardware for power/energy efficiency.
•Application models for efficient energy-proportional computing and a means to insert power-management pragmas in the application.

Performance: Attain performance increases over current designs by:
•Proposing performance-enhancing architectural features, including primitives for fine-grain synchronization and communication.
• Developing a compiler that efficiently drives these performance features.
• Identifying application Idioms and mapping them efficiently on Thrifty, using them to insert tuned libraries, and performing novel thread scheduling.

Resiliency: Reduce the waste in computing due to faults and recovery in high-end systems by developing:
• Novel circuit technologies for high resiliency at low supply voltage.
• New circuits and architecture technologies for efficient error detection and tolerance.
• A novel architectural scheme for energy-efficient, diskless, scalable check-pointing.
• Applications and compiler support to drive the novel check-pointing scheme.

In this report, we focus on the progress at the University of Illinois during the last year of the grant (September 1, 2013 to August 31, 2014). We also point to the progress in the other collaborating institutions when needed.

2. Progress During the Period

2.1. Overall View of the Effort

This year we have written a paper that gives an overall view of the project. It is called "*Extreme-Scale Computer Architecture: Energy Efficiency from the Ground Up*", and it appeared in the International Conference on Design, Automation and Test in Europe (DATE), March 2014 [4]. It describes the challenges faced by exascale architectures and how we are addressing them.

The paper describes that, as we move to integration levels of 1,000-core processor chips, it is clear that energy and power consumption are the most formidable obstacles. To construct such a chip, we need to rethink the whole compute stack from the ground up for energy efficiency --- and attain Extreme-Scale Computing. First of all, we want to operate at low voltage, since this is the point of maximum energy efficiency. Unfortunately, in such an environment, we have to tackle substantial process variation. Hence, it is important to design efficient voltage domains on the chip that can operate at the most efficient voltage and frequency point. At the architecture level, we require simple cores organized in a hierarchy of clusters. Moreover, we also need techniques to reduce the leakage of on-chip memories and to lower the voltage guard-bands of logic. Finally, data movement should be minimized, through both hardware and software techniques. With a systematic approach that cuts across multiple layers of the computing stack, we can deliver the required energy efficiencies.

2.2. Power/Energy Efficiency

2.2.1. Energy Management API

A significant contribution of the Thrifty project over these years is that we have developed an Energy Management API[1] that enables the compiler or programmer to change chip configuration parameters. The goal is to execute in the most energy-efficient manner. The API consists of library calls to:

1. Select the Voltage and Frequency Bin for each of the processors and for each of the L2 caches in the chip.
2. Turn-off (power-gate) and turn-on various resources in the chip. They include individual processors, caches, ways of associative caches, and functional units.

The Thrifty hardware architecture is designed with the ability to change the voltage and frequency bins of individual components, as well as to power-gate individual components of the hardware. The SESC architecture simulator that we developed uses this API to reconfigure the hardware architecture.

This API has been transferred to the Intel-lead Traleika Glacier X-Stack project.

The ROSE compiler has been extended to be able to analyze the code and then call the API to configure the architecture in the most energy-efficient mode. In particular, the ROSE compiler:

1. Detects serial and parallel sections in OpenMP programs. Before the serial section starts, it calls the API to power-off all the processors that will remain idle (and their caches). Before the parallel section starts, it calls the API to power-on all the processors and caches.

2. Detects at the source code sections of the code that use different functional unit types. For example, it detects the use of floating point and integer functional units. When such units are not used, it power-gates them.

The programmer is also allowed to call the API based on his/her knowledge of the program.

2.2.2. Cache Hierarchy Reconfiguration for Energy Efficiency

As an example of using this API in the SESC architecture simulator, we have performed a study reconfiguring the cache hierarchy of a large scale multiprocessor. The cache hierarchy often consumes a large portion of a system's energy. To save energy in high-performance environments, we propose to reconfigure the cache hierarchy by reducing the number of ways of associative caches and the sizes of the caches, with a software-controlled adaptive runtime system. Our approach also lets the user specify the best cache hierarchy configuration for a given application.

Our approach uses formal language theory to express the application's pattern and help predict its future. Furthermore, it uses the prevalent Single Program Multiple Data (SPMD) model of HPC codes to find the best configuration in parallel quickly. Our experiments using cycle-level simulations indicate that 67% of the cache energy can be saved with only a 2.4% performance penalty on average. Moreover, we demonstrate that, for some applications, switching to a software-controlled reconfigurable streaming buffer configuration can improve performance by up to 30% and save 75% of the cache energy.

This work appeared as "*Using an Adaptive HPC Runtime System to Reconfigure the Cache Hierarchy*" in the International Conference for High Performance Computing, Networking, Storage and Analysis (SC) in November 2014 [8].

2.2.3. Reducing the Refresh Energy in On-Chip eDRAM Modules

As the Thifty chip uses dynamic energy ever more efficiently, static power consumption becomes a major concern. In particular, leakage in on-chip memory modules contributes substantially to the chip's power draw. This is unfortunate given that, intuitively, the large multi-level cache hierarchy of a many-core is likely to contain a lot of useless data.

An effective way to reduce this problem is to use a low leakage technology such as embedded DRAM (eDRAM). However, such systems require refresh. In this work, we examined how to significantly reduce the refresh energy for large last-level caches. In practice, it is well known that different eDRAM cells can exhibit very different charge-retention properties. Unfortunately, current systems pessimistically assume worst-case retention times, and end up refreshing all the cells at a conservatively-high rate.

In this work, we propose an alternative approach. We use known facts about the factors that determine the retention properties of cells to build a new model of eDRAM retention times. The model is called *Mosaic*. The model shows that the retention times of cells in large eDRAM modules exhibit *spatial correlation*. Therefore, we logically divide the eDRAM module into regions or *Tiles*, profile the retention properties of each tile, and program their refresh requirements in small counters in the cache controller. With this architecture, also called Mosaic, we refresh each tile at a different rate. The result is a 20x

reduction in the number of refreshes in large eDRAM modules --- practically eliminating refresh as a source of energy consumption.

This work appeared as "*Mosaic: Exploiting the Spatial Locality of Process Variation to Reduce Refresh Energy in On-Chip eDRAM Modules*" in the International Symposium on High Performance Computer Architecture (HPCA), February 2014.

2.3. Performance

2.3.1. API for Software-Managed Caches

Another significant contribution of the Thrifty project over these years is that we have developed an API for software-managed caches (SMC). It enables the compiler or the programmer to manage the coherence of a multiprocessor cache hierarchy in software [2]. The goal is to execute in a more energy-efficient manner and still retain high performance.

With SMCs, as a processor references a variable, the memory line containing the variable is automatically brought from memory or from the lower layers of the cache hierarchy (e.g., L3) and copied into the processor's local cache. However, caches are not coherent, which means that the processor simply gets the value that is currently in memory (or in the first level of the cache hierarchy that it finds it in). If a processor P2 wants a value that another processor P1 wrote, then P1 has to first explicitly *write back* the variable to memory, and then P2 has to *invalidate* its own cached copy before reading the variable.

The ISA can be used by the compiler or the programmer to make sure the correct data is accessed. The ISA includes: 1) *write-back* of a word (or a range of addresses) from the cache to memory (or shared caches), 2) *invalidation* of a cached word (or range of addresses), 3) *write-back and invalidation* of a word (or range of addresses), and 4) *cache-bypassing* loads and stores. When a processor reads a variable and the data is brought from memory because the word was invalid in the cache, the whole cache line is brought into the cache. When it writes back a line, only dirty words of the line are written back to memory in a writeback, reducing the traffic significantly.

This API has been transferred to the Intel-lead Traleika Glacier X-Stack project.

2.3.2. Compiler-Driven Software Managed Caches

In collaboration with Professor Sadayappan from Ohio State University, we have developed a compiler algorithm using exact polyhedral dependence analysis to optimize coherence instructions for affine computations [9]. The developed approach inserts coherence primitives – invalidate and write-back instructions – at a coarse granularity and combines invalidations and write-backs of a number of words together, to reduce both the frequency of coherence operations and the volume of words moved between private caches and the shared level cache or main memory. The experimental evaluations over a number of benchmarks show that the developed system is effective both for energy and performance metrics.

We develop compiler algorithms for two types of applications. For regular applications, we precisely mark variables for invalidation in a processor's private cache because they have become stale; and accurately determine data that are to be written from the private cache of a processor to shared cache because other processors will access those data values.

For iterative irregular applications, we present *inspector*-based schemes to exactly demarcate data for coherence. Other irregular parallel applications are handled via conservative methods that do not preserve cache coherence at all times, but still enable coherent data access in cache between after synchronization points.

Compared to prior works on compiler-directed cache coherence, the compiler support developed in this paper is more general as it is applicable to a larger class of programs and it is more precise as the compiler analysis takes task-to-processor mapping into account.

We also find that it is best to write applications for SMC from scratch, rather than simply taking existing codes and translating them. By writing from scratch, the programmer can take advantage of the data that can remain in the caches across barriers (or synchronizations) because the same processor will use it next. There is no need to move such data. In addition, data that will not be used by anyone any more can be discarded silently. This work is submitted for publication [9].

2.3.3. User-Driven Software-Managed Caches

We have also attempted to rely on the programmer's knowledge of a program to use this API for software-managed caches (SMC). We insert *write-back* and *self-invalidation* directives in the source code of non-trivial parallel code. We show the subtle issues that these instructions need to face, including reordering in the pipeline, and effective use of caches organized in clusters for energy efficiency.

We show a simple approach that the programmer can use to orchestrate the movement of data. It is based on exploiting the synchronization points in the program and the type of synchronization operations. Moreover, if the programmer provides additional communication pattern information, we can improve the performance in a cluster-based hierarchy.

Our simulation results show that the execution of applications on incoherent cache hierarchies can deliver reasonable performance. For execution within a single cluster, the performance is comparable to

simple support for hardware coherence. For execution across multiple clusters, the performance is lower, but it scales with the processor count. In the general case, programming for performance is a challenge. This work is submitted for publication [10].

2.4. Resiliency

2.4.1. Coping with Parameter Variations

By lowering the supply voltage to be slightly above the threshold voltage (Vth), we reduce the energy per operation substantially. This regime of operation, called Near-Threshold Voltage Computing (NTC), allows many more cores to operate under a given manycore power envelope.

Unfortunately, with voltage in close proximity to Vth, devices have a higher susceptibility to parametric variation -- i.e., the deviation of device parameters from their nominal specifications. Variation creates substantial differences in speed and power across the cores in the chip. Relying on the worst-case margins does not represent a practical design option, as the nominal frequency of operation is already low. As a result, we may suffer timing errors, and we need to design techniques that guarantee the resilience of processors and memories in the presence of these errors --- for example, by designing stronger ECC schemes for memories. A further difficulty stems from the diminishing efficacy and increasing cost of state-of-the-art variation mitigation techniques for conventional operation when applied to NTC. These techniques rely heavily on voltage tuning in independent voltage domains on chip.

As part of this grant, we have addressed variation at the architecture level. First, we introduced an architectural model of parameter variation at NTC. We used the model to show the shortcomings of adapting state-of-the-art techniques for variation mitigation to NTC. Finally, we show how we can tailor variation mitigation to NTC, with the use of a many-core organization called EnergySmart.

While we have published several papers on this topic over the last three years, we summarized the relevant contributions in a paper that we recently published in the IEEE Micro Magazine, Special Issue on Reliability-Aware Microarchitecture Design, in July-Aug. 2013. The paper title is "*Coping with Parametric Variation at Near-Threshold Voltages*" [3].

2.4.2. Reliable Energy-Efficient On-Chip Networks

To build a power-efficient Thrifty chip, we reduce the supply voltage and operate the chip at near-threshold voltage. However, in an NTC environment, there are significant variations in speed and power consumption across the chip. In particular, the on-chip interconnection network is especially vulnerable to variations. This is because the network connects distant parts of the chip which, due to variations, exhibit different speed and power characteristics. The network has to be designed conservatively, to work under the most unfavorable parameter values in the chip. This results in energy-inefficient designs.

On-chip networks can already consume a substantial fraction of the on-chip power --- potentially up to 30--40%, according to the literature. Conservative future network designs, needed to tolerate parameter variations, may be unable to reduce the value of this fraction much.

Fortunately, the voltage guard-bands present in the network to tolerate parameter variations offer an opportunity for energy savings: due to variations, the guard-bands in some areas of the chip are likely to be significantly over-provisioned. The insight in our work is to reduce these guard-bands to save energy, while being mindful not to reduce voltage so much as to cause timing errors in the network. Interestingly, well-known error detection and tolerance mechanisms in the network can be used to find out when voltage has reached a tolerable lower bound.

Finding which groups of routers should lower their voltage and by how much in a distributed environment is not trivial. Thus, in this work we have proposed a mechanism, called *Tangle*, that dynamically measures the errors of messages and scales the voltage of groups of routers to an error-free minimum. The frequency of the network remains unchanged.

Tangle augments a multi- or many-core chip that has multiple voltage domains and the capability to perform dynamic voltage scaling. Tangle monitors the errors of messages as they traverse the network. If errors are observed, Tangle dynamically increases the voltage of the router groups used by the erroneous messages. Tangle also periodically decreases the voltage of all the routers. With this approach, the voltage values applied to different groups of routers progressively converge to their lowest, variation-aware, error-free values. This saves substantial network energy. Tangle has no noticeable performance overhead because it does not reduce frequencies and keeps the error rate to a bare minimum.

We evaluate Tangle with simulations of a variation-afflicted 64-router network. With 4 voltage domains in the network, Tangle reduces the network energy consumption by an average of 22% with negligible performance impact. In a future network design with one voltage domain per router, Tangle lowers the network voltage by an average of 21%, reducing the network energy consumption by an average of 28% with negligible performance impact.

We published this work as "*Tangle: Route-Oriented Dynamic Voltage Minimization for Variation-Afflicted, Energy-Efficient On-Chip Networks*" in the International Symposium on High Performance Computer Architecture (HPCA), February 2014[6]. It was one of the Best Paper Nominees.

2.5. Outreach

As part of this grant, we have made substantial efforts in outreach, including giving talks, organizing panels and workshops, and teaching courses. In this last year of the grant, we note three important outreach activities.

We organized the "*2013 Illinois Symposium on Parallelism: Current State of the Field and the Future*" on September 2013, at the Siebel Center for Computer Science of the University of Illinois at Urbana-Champaign [7]. We had talks, discussions and panels related to exascale computing and energy-efficient computing. Program director Dr. Sonia A. Sachs attended the symposium.

We organized and participated in the Second Workshop on Near-threshold Computing (WNTC), which was held in Minneapolis, MN, in June 2014.

As part of The 23rd International Conference on Parallel Architectures and Compilation Techniques (PACT), we invited Dr. Sacks to give a talk on DOE's Exascale effort. The talk was on tuesday, August 26 and was titled "*Exascale Software Stack: Present, Future*".

2.6. Contributions of the Collaborating Institutions

Our partner institutions are Lawrence Livermore National Laboratory (LLNL) and The University of California San Diego (UCSD). While they have submitted their final reports separately, their work is fully integrated with the work presented here. In particular, PI Quinlan from LLNL has been in charge of the compilation aspects, while PI Carrington from UCSD has been in charge of the applications and scheduling work. At the same time collaborator Pinfold from Intel has been advising on circuits work.

This last year of the project, PI Quinlan from LLNL has extended ROSE's OpenMP implementation to work with our architecture simulator, has developed a new Thrifty specific source-to-source translator that can translate directives to call Power API functions provided by the simulator, and has created a NUMA-aware runtime library. This last year, PI Carrington from UCSD has evaluated and compared scratchpads and caches, analyzed co-scheduling of applications, and proposed application-aware reconfiguration of the cache hierarchy.

One of the contributions of the Thirfty project has been the construction of an extensive tool chain that includes application instrumentation, compilation analysis, and architecture simulation, all integrated into a large software system. This software is currently being used in the Intel-lead Traleika Glacier X-Stack project, and is available for use in other DOE projects.

References:

[1] Aditya Agrawal and Josep Torrellas , "*Power API for Thrifty*", Technical Report, University of Illinois at Urbana-Champaign, December 2013.

[2] Wooil Kim and Josep Torrellas, "*Programmer Managed Caches: Concepts, APIs and Example Codes*", Technical Report, University of Illinois at Urbana-Champaign, February 2014.

[3] Ulya Karpuzcu, Nam Sung Kim, and Josep Torrellas, "*Coping with Parametric Variation at Near-Threshold Voltages*", IEEE Micro Magazine, Special Issue on Reliability-Aware Microarchitecture Design, Volume:33 Issue:4, July-Aug. 2013.

[4] Josep Torrellas , "*Extreme-Scale Computer Architecture: Energy Efficiency from the Ground Up*", International Conference on Design, Automation and Test in Europe (DATE), March 2014.

[5] Aditya Agrawal, Amin Ansari, and Josep Torrellas, "*Mosaic: Exploiting the Spatial Locality of Process Variation to Reduce Refresh Energy in On-Chip eDRAM Modules*", International Symposium on High Performance Computer Architecture (HPCA), February 2014.

[6] Amin Ansari, Asit Mishra, Jianping Xu, and Josep Torrellas, "*Tangle: Route-Oriented Dynamic Voltage Minimization for Variation-Afflicted, Energy-Efficient On-Chip Networks*", International Symposium on

High Performance Computer Architecture (HPCA), February 2014. Best Paper finalist.

[7] Josep Torrellas, Sarita V. Adve, Vikram S. Adve, Danny Dig, Minh N. Do, Maria Jesus Garzaran, John C. Hart, Thomas S. Huang, Wen-mei W. Hwu, Samuel T. King, Darko Marinov, Klara Nahrstedt, David A. Padua, Madhusudan Parthasarathy, Sanjay J. Patel, and Marc Snir, "*Making Parallel Programming Easy: Research Contributions from Illinois*", September 2013.

[8] Ehsan Totoni, Josep Torrellas, and Laxmikant V. Kale, "*Using an Adaptive HPC Runtime System to Reconfigure the Cache Hierarchy*", International Conference for High Performance Computing, Networking, Storage and Analysis (SC), November 2014.

[9] Sanket Tavarageri, Wooil Kim, Josep Torrellas, and P. Sadayappan, "*Automatic Generation of Coherence Instructions for Software-Managed Multiprocessor Caches*", in submission.

[10] Wooil Kim and Josep Torrellas, "*Architecting and Programming an Incoherent Multiprocessor Cache Hierarchy*", in submission.