Toward Application-Specific Memory Reconfiguration for Energy Efficiency

Pietro Cicotti
San Diego Supercomputer Center
University of California
San Diego

Laura Carrington
San Diego Supercomputer Center
University of California
San Diego

Andrew Chien
Department of Computer Science
University of Chicago
Argonne National Laboratory

ABSTRACT

The end of Dennard scaling has made energy-efficiency a critical challenge in the continued increase of computing performance. An important approach to increasing energy-efficiency is hardware customization. In this study we explore the opportunity for energy-efficiency via memory hierarchy customization and present a methodology to identify application-specific energy efficient configurations. Using a workload of 37 diverse benchmarks, we address three key questions: 1) How much energy saving is possible?, 2) How much reconfiguration is required?, and 3) Can we use application characterization to automatically select an energy-optimal memory hierarchy configuration? Our results show that the potential benefit is large – average reductions close to 70% in memory hierarchy energy with no performance loss. Further, our results show that the number of configurations need not be large; 13 carefully chosen configurations can deliver 93% of this benefit (64% energy reduction), and even coarse-grain reconfigurations of an existing hierarchy can deliver 81% of this benefit (56% energy reduction), suggesting that reconfigurable hierarchies may be practically realizable. Finally, as a first step towards automatic reconfiguration, we explore application characterization via reuse distance as a guide to select the best memory hierarchy configuration; we show that reuse distance can effectively predict the application-specific configuration which will both maintain performance and deliver energy efficiency.

Categories and Subject Descriptors

D.3.3 [**Memory Structures**]: Semiconductor Memories–*Static memory*.

General Terms

Performance, Design.

1. INTRODUCTION

A major factor driving research in computer architecture is the increasing pressure for energy efficiency due to the end of Dennard scaling [1] combined with practical limits to the power a processor can consume [2, 3] in smartphone, tablet, laptop, and server environments. As Moore's law continues to deliver increases in transistors density [4], there are growing opportunities for CPU customization ranging from core extensions [5-7] to accelerators on systems-on-chip [8, 9]. These systems all exploit customization in the datapath – matching the transistor structure of the processor to the needs of the application – to improve both performance and energy efficiency.

 E2SC '13, November 17 - 21 2013, Denver, CO, USA, Copyright 2013 ACM 978-1-4503-2504-2/13/11…$15.00.
http://dx.doi.org/10.1145/2536430.2536435

While the aforementioned efforts focus on the micro-architecture, our focus in this work is complementary. We explore memory hierarchy customization for energy efficiency. Memory hierarchies have long been critical elements of system performance. As processors have continued to improve at a faster rate than memory, commonly known as the memory wall [10], multi-level caches have been a key element of computer architecture for decades with research studies spanning organization [11], write and prefetch techniques [12-14], and many other aspects. In short, all modern microprocessors depend on highly optimized memory hierarchies to achieve good performance.

High performance often comes at a high power cost. The memory sub-system of modern processors is a significant fraction of the systems power budget [15]. In this work, we explore the potential of a configurable memory hierarchy to reduce this power cost by matching its structure to an application's needs. Our methodology characterizes an application's data locality and uses that characterization to determine its energy optimal configuration. This approach posits CPUs with reconfigurable cache sizes at multiple levels, and with the growing presumption of dark silicon [2, 3], we discard normalized comparison based on equal area or capacity. In many cases, using less (e.g. turning off parts or levels of the hierarchy) produces higher energy efficiency. This holistic view builds on a wide variety of work to optimize memory hierarchy energy-efficiency, including reconfiguration for dynamic energy reduction (e.g. cache and TLB access mechanism and organization [16, 17]), circuit level efforts to reduce leakage (e.g. drowsy caches [18]), even architecture level techniques to reduce leakage (e.g. cache decay [19]). However, by opening the design space by presuming the availability of dark silicon and doing macroscopic cache array configuration (in/out), we can address all aspects of dynamic and static energy, including even the perhaps 20% remaining leakage, which at deep submicron technologies and 16MB+ cache sizes is quite significant. While we do not study a detailed hardware design, we believe such macroscopic reconfiguration can be achieved with high efficiency, and is much less aggressive than recent proposals for reconfiguration deep into the core [20]. Finally, we reconsider our results, in a context with limited reconfigurability, assuming a reconfiguration mechanism based on today's processors cache design [21, 22].

In summary, we focus on energy savings via reconfiguration of the bulk of the on-chip memory hierarchy, setting cache sizes and number of levels in a modern technology context. In this paper, we explore three critical questions for such reconfigurable memory hierarchies:

- What is the potential energy efficiency benefit to application programs of a reconfigurable memory hierarchy?
- How much of that benefit can be achieved by coarse-grain reconfiguration?

- Can we exploit that benefit automatically, reconfiguring the system to best match the application's needs?

To explore these questions, we use a diverse workload of 37 benchmarks to explore a wide range of memory hierarchy configurations of varying depth and size. Using CACTI [23] and DRAM energy modeling, we explore the overall potential for energy savings by first identifying the best memory hierarchy configuration for each application, optimizing for performance and energy. With this foundation, we explore the question of how much configurability is needed to capture the majority of the energy efficiency benefit – whilst preserving performance, and further explore how to choose those configurations based on application properties. The major contributions of this paper are the results of this inquiry and include:

- The opportunity for energy savings via memory hierarchy reconfiguration is large – over 70% with no performance loss. If modest performance degradation is acceptable, as much as 95% of memory hierarchy energy can be saved.
- Across a wide range of leakage models, this picture changes little, with reconfiguration enabling 69% potential energy savings with no loss of performance assuming 50% leakage reduction by techniques such as drowsy caches.
- Reducing the space to coarse-grain reconfigurations (e.g. enabled by power gating techniques) can still deliver a large fraction of the benefit. Our analysis finds that the ability to power off half or the entire L2 and L3 caches is sufficient to deliver an average 56% energy reduction.
- Finally, we show that automatic reconfiguration may be feasible based on reuse distance, demonstrating a 45% energy saving in a small reconfiguration space.

The rest of the paper is organized as follows. Section 2 outlines the methodology of our study, the memory hierarchy space, and energy models used in the study. In Section 3, we explore the opportunity for energy reduction by customizing the memory hierarchy, also exploring the potential performance impact. Section 4 addresses the question of the energy reduction achievable with little reconfigurability, and Section 5 the challenge of how to automatically select memory hierarchy reconfigurations. Section 6 discusses our results and the most relevant related work, and then we summarize and point out some directions for future work in Section 7.

2. METHODOLOGY

This study explores energy-optimal memory configurations for a workload of 37 benchmarks. For a given application and memory configuration pair, energy and performance are estimated using details about the memory configuration and the application behavior (i.e. a characterization of the application).

For each cache configuration, we obtained latency, access energy, and leakage parameters using CACTI [23, 24] for the 32nm technology node. For DRAM we assumed 400 cycle access latency and 40pJ/bit access energy [24]. For the application characterization we used PEBIL [25], a binary instrumentation tool. The remainder of this Section describes the characterization, the energy model used to estimate energy, and the process for selecting the optimal configuration.

2.1 Workload Characterization

In order to estimate energy consumed in the memory hierarchy we characterize each application by hits and misses, at each level, and for each hierarchy configuration. The applications are instrumented to dynamically run the address stream against a cache simulator and collect hit/miss counts. Hits and miss counts are then used to estimate energy consumption and performance.

Performance is estimated using the Average Memory Access Time (AMAT) and by comparison to a reference system. First, each application is run native (without instrumentation) on a reference system to measure its runtime, and then instrumented to simulate the configuration of any target system and the reference system. From the simulation results, AMAT is computed for any target system and the reference system. Finally, the runtime for a target system is estimated scaling the measured runtime by the ratio of AMAT of the target system to the reference system.

2.2 Energy Model

Energy consumption is modeled as the sum of two terms: dynamic energy, consumed by reading and writing data, and static leakage.

The characterization of an application provides the energy model with a count of per-level cache hits. Hits are used together with access energy parameters to compute dynamic energy for the caches (misses at the LLC are accounted for as DRAM accesses).

The characterization also provides a runtime estimate which is used to estimate energy leaked.

3. ENERGY SAVING OPPORTUNITY

In this Section we present results for the overall energy saving potential for customized memory. The energy savings are estimated for the 37 benchmark applications, which are described in Section 3.1.

To explore the potential saving with maximal configurability, we started from a very broad range of configurations. The search space is bounded at 8KB (minimum L1 cache size) and 64MB (maximum cache size). Parameters such as associativity, banking, and access latency, are tuned to approximate existing architectures and kept constant for any given size. The resulting search space includes one-level, two-level, and three-level configurations, with the constraint that any given level has to be at least twice as big as the previous level. In total, the search space includes 2,652 cache configurations.

After characterizing an application for each single point in the search space, we evaluated energy consumption and performance at each point and selected the energy optimal configuration given a set of restrictions.

In Section 3.2 we select the optimal configuration solely with respect to energy consumption; in Section 3.3 we select the optimal configuration that does not degrade performance, to avoid energy savings at the expense of performance; and in Section 3.4 we study how the selection changes when considering leakage-reducing techniques.

For the all experiments, we used a reference system with an Intel Sandy Bridge CPU with a 32KB L1, a 256KB L2, and a 16MB L3 [26] (one of the Sandy Bridge configurations).

3.1 Workload

The workload is selected to cover a broad variety of computational domains, including traditional HPC applications as well as emerging data intensive applications. The workload consists of 37 different applications from 6 different benchmark suites: NPB, PARSEC, Mantevo, UHPC, Minebench, and Biobench, and include a variety of compute and memory-bound applications as well as a range of server and desktop applications.

The NAS Parallel Benchmarks (NPB) is a collection of kernels and pseudo-applications that represent computation and data

movement in computational fluid dynamics workloads [27]. The Princeton Application Repository (Parsec) [28] is a general workload suite designed to represent emerging applications, including vision, analytics, and image processing. The Mantevo suite is a collection of proxy applications intended to mimic large scale Finite Elements and Molecular Dynamics applications [29]. The Ubiquitous High Performance Computing (UHPC) application suite is a set of five applications representing DoD workloads and were used as reference in DARPA's UHPC program [30]. Minebench is a data mining benchmark suite with applications in different algorithmic categories such as clustering, classification, and optimization [31]. Biobench is a benchmark suite of bioinformatics applications including sequence alignment and assembly code [32]. The benchmark suites and the applications used in our study are listed in Table 1.

Table 1: Benchmark Suite

Package	Application
NPB	BT, CG, DC, EP, FT, IS, LU, MG, SP, UA
PARSEC	blackscholes, bodytrack, facesim, ferret, freqmine, swaptions, fluidanimate, vips, canneal, dedup, streamcluster
Mantevo	miniMD, miniFE, HPCCG
UHPC	chess, graph, lulesh, md, sensor
Minebench	ECLAT, Baeysian, semphy
Biobench	mummer, clustalw, hmmer, phyilip, fasta

3.2 Performance Oblivious Selection

We first look at the energy optimal configuration for each application without restrictions. This results in dramatic benefits, but at a performance price. Remarkably, the energy saving for each (and every) application is greater than 80%, with an average at 95% saving, as shown in Figure 1. However, the savings come at a stiff performance price.

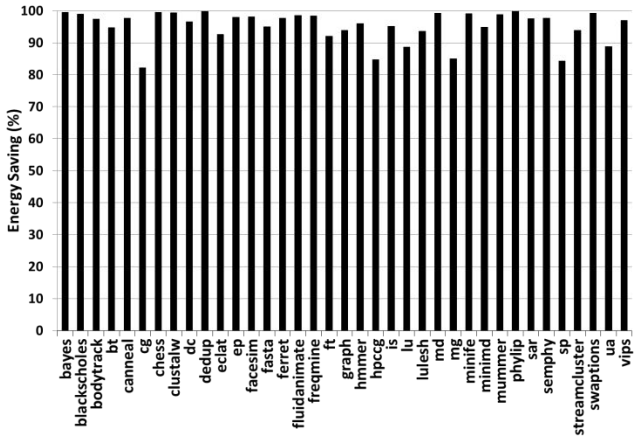

Figure 1. Energy savings with performance oblivious selection.

The performance price is a consequence of not having an L3 cache in the configurations selected. All the energy-optimal configurations selected have no L3 cache, and the majority do not have an L2 cache either. The result is that many of the applications run significantly slower than on the reference configuration, which results in an average 1.4 slowdown over the entire workload. Essentially, all of these configurations trade performance for energy leaked by the L3 cache: as soon as the energy leaked at a certain level outweighs the energy consumed to access the next level or DRAM, then the optimal configuration does not have such a level. For a large L3, this is often the case since often L1 and L2 caches capture most of the useful locality. Another aspect of this selection is that applications with high locality can enjoy high energy saving, because the overall dynamic energy cost in accessing DRAM is low and easily outweighed by leakage.

If we target the selection to reduce power draw we see a large reduction but an even worse performance penalty. In the model assumed, power is averaged over all the execution, and both lower energy and longer runtime contribute to reducing power draw.

Figure 2 highlights the difference in performance when optimizing for energy and power. While in some cases the optimization target aligns with performance, in most cases the goals are in conflict. As a result, optimizing for energy yields an average 0.7x speedup (1.4x slowdown), and optimizing for power yields an average 0.5x speedup (2x slowdown).

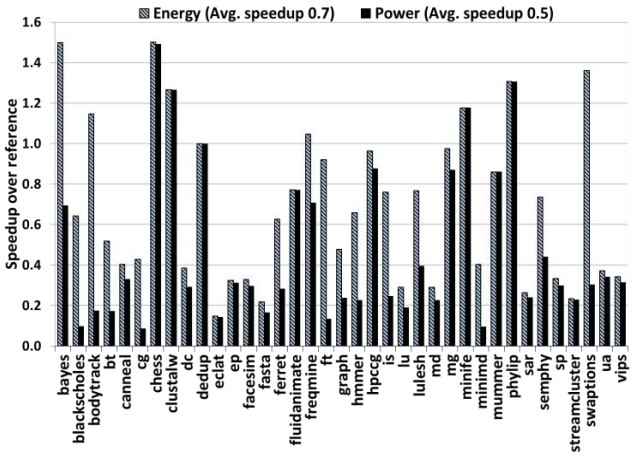

Figure 2. Relative speedup for each application when optimizing for energy and power.

3.3 Performance Preserving Selection

To refocus the analysis onto more practical results we restrict the selection of the optimal configuration to those that do not degrade performance. In this way, we eliminate some of the low-energy configurations and the result is that the average energy saving drops to 70%. However, despite the restriction, the energy saving is still significant and comes with no slowdown in any of the 37 benchmarks. In fact, overall there is an average 1.2x speedup. Energy savings for each application are shown in Figure 3.

While the average saving does not drop significantly, some of the applications enjoy a much lower saving as they are well matched to the reference memory hierarchy configuration. Nevertheless, in many cases we still observe high savings and in the average as well. In the remainder of the paper we restrict the selection to performance preserving configurations.

Five applications had best energy efficiency with a single-level cache of modest size, reflecting the fact that the remainder of their data was streamed or had no locality or reuse, and no size of L2 or L3 would yield a net energy (and performance) benefit. Half of

the applications did not benefit from an L3 cache for similar reasons. These results show the major disincentive for large on-chip caches and associated leakage costs.

From the large memory hierarchy configuration space, nearly every application is matched with a unique configuration and 32 different configurations are selected for the 37 applications. This substantiates the use of a large configuration space to best match the unique needs of each application's locality structure. However, in Section 4 we investigate the energy saving when further restricting the search space to a set of few carefully selected configurations.

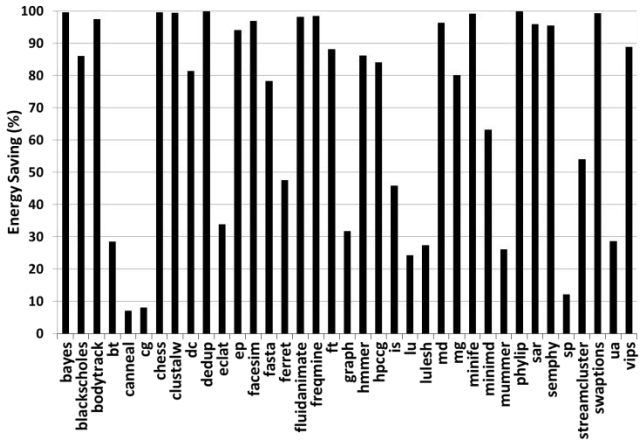

Figure 3. Energy savings while preserving performance.

3.4 Reduced Leakage Analysis

In CPUs with large on-chip memories, energy is largely dissipated due to leakage. To reduce leakage, several leakage-reducing techologies have been proposed. To validate our assessment and verify the sensitivity leakage energy, we reassess the potential benefit under four different leakage models, which are described in Table 2 (expanding the search space to nearly 400,000 data points).

Table 2: Leakage models.

DynOnly	**Dynamic Only** Includes dynamic power only (no leakage).
AggLkR	**Aggressive Leakage Reduction** Includes 10% of the leakage. Models very aggressive leakage reduction, including transistor optimization, drowsy caches, and other techniques with optimistic assumptions.
LkR	**Leakage Reduction:** Includes 50% of the leakage. Consistent with the published benefits for drowsy caches and transistor threshold optimization leakage reduction.
Full	**Full Leakage**: No leakage-reduction is assumed.

The results are summarized in Table 3 and show that configurability can save energy even with reduced leakage. That is, even with reduced leakage there is a substantial opportunity to reduce energy consumption, with a lower bound of an average 44% saving for the dynamic-energy-only model (DynOnly). In addition, only under very aggressive low-leakage models a large difference with respect to the Full models is seen, and even with the LkR model (50% leakage) there is almost no difference in the energy saving achieved. In the following Sections we assume the LkR model.

Table 3: Average energy saving for different leakage models.

Model	Average Energy Saving (%)
DynOnly	44
AggLkR	65
LkR	69
Full	70

4. RESTRICTED SPACE SELECTION

Restricting the search space has clear benefits in terms of complexity of the implementation of the hierarchy and of the selection functionality. In addition, we consider reducing the search space to a handful of configurations that are subsets of the reference hierarchy, and as such can result from coarse-grain reconfiguration mechanisms (e.g. power gating).

We gradually reduce the configuration space and observe how much energy saving is retained when reducing the search space. First, we coarsen the granularity of the space by removing all cache sizes that are not a power of 2. As a further refinement step we restrict the space around the reference cache by allowing only an 8x variation. In this way, for each cache level a configuration cannot be more than 8x smaller or larger than the corresponding level in the reference although we keep the option for not having L2 or L3 caches. We further reduce the space by using hierarchical clustering to group similar configurations and obtain 13 and 6 *centroid* configurations. Finally, we consider a search space obtained by reconfigurations of the reference system, assuming that L2 and L3 can be reduced to half size, or turned off entirely. The configuration spaces corresponding to the complete space and the refinement steps described are called *Full*, 2^n, *Restricted, Clustering, Clustering6,* and *Reconfigurable*, respectively.

The slow-progressing decrease in energy saving resulting from restricting the configuration space is shown in Table 4. The average energy saving slowly decreases from 69% to 62% with 6 configurations, and to 56% for the *Reconfigurable* space. The benefit loss is relatively small, even when we restrict our attention to *Restricted* and smaller spaces (regardless of how arbitrarily large is the *Full* space) and between a carefully selected space and one derived by the reference (e.g. comparing *Clustering* and *Configurable* which have about the same number of configurations). Figure 4 shows the per-application energy saving across the different configuration spaces.

Table 4: Cache configuration spaces and energy savings.

Space Name	# of Conf. in search space	# of Unique Conf. selected	Avg. Energy Savings (%)
Full	2652	33	69
2^N	469	26	66
Restricted	224	23	66
Clustering	13	13	64
Clustering6	6	6	62
Reconfigurable	14	7	56

Further reducing the space reveals how small the space can be to still save energy. In the process we find that reducing to fewer than 6 configurations requires including the reference configuration in the search space; this is necessary to preserve performance and it means that for certain applications, energy savings can be realized only with a certain degree of reconfigurability. Finally, reaching a one-configuration space, the process converges to the reference configuration, which is the configuration that guarantees no performance loss, indicating that the reference itself strikes a good balance between performance and energy consumption.

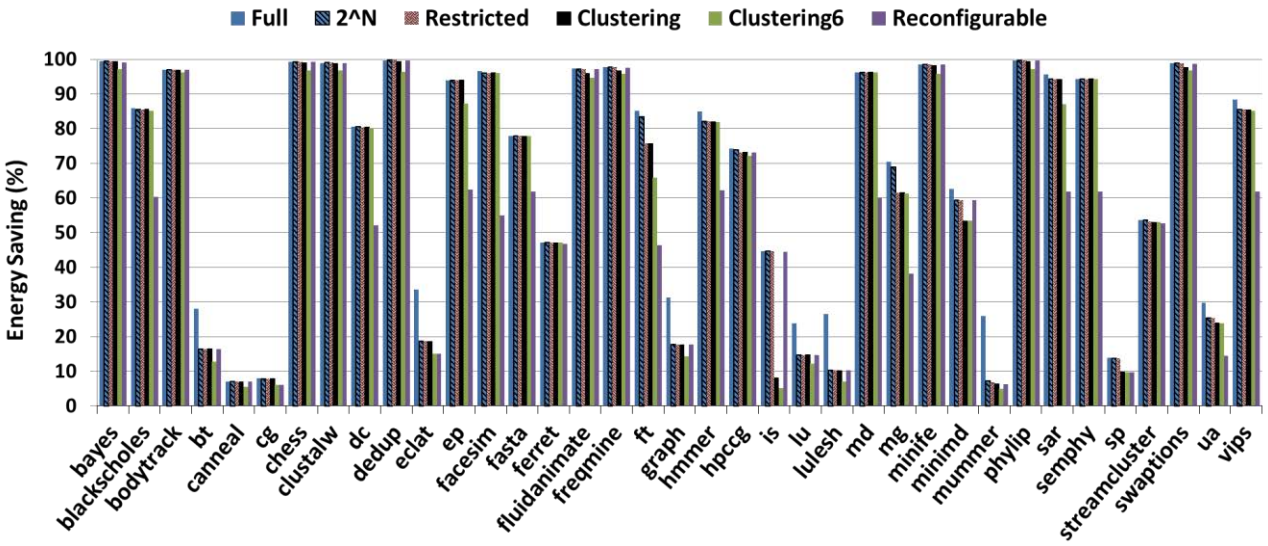

Figure 4. Energy saving across configuration spaces.

Table 4 also shows that even within the Restricted space, applications continue to select unique configurations, for the most part, and span the whole 13 configurations in the *Clustering* space. This indicates that the workload selected represents different locality patterns, and that it is important to maintain a diverse set of configuration to satisfy the diversity in locality patterns. It also appears that workloads have a clustered distribution therefore, having regularly distributed configuration spaces, such as the *Reconfigurable* space, may not guarantee an optimal coverage of the workloads. Figure 5 shows the distribution of the workloads on *Clustering* and *Reconfigurale.*

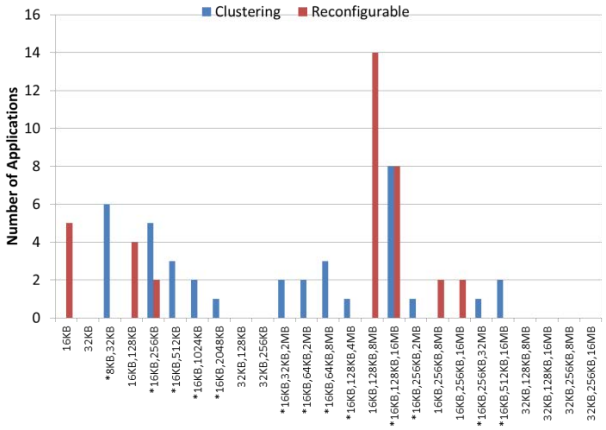

Figure 5. Distribution of selected cache configurations in Clustering and Reconfigurable search spaces.

In the *Clustering* space, all configurations are selected despite some imbalance toward the two extremes (small caches and no L3, and 16MB L3). This distribution addresses recurring patterns with very high locality in one case, and relatively low locality in the other, and in general, half of the applications have enough locality and gravitate towards *L3-less* configurations. But, since *Reconfigurable* does not result from any of those considerations, some of its configurations simply do not match any of the benchmarks, which are forced to select less suited configurations; only half of the configurations in *Reconfigurable* are selected by any benchmark.

5. TOWARD AUTOMATIC SELECTION

The selection of the optimal configuration should leverage a characterization of the application to select an optimal reconfiguration. Toward this goal, we consider reuse distance as a tool to characterize an application and that can be used to model energy consumption. Reuse distance has been studied and utilized as a platform independent metric representative of cache behavior. Assuming 64B cache lines, we use the following definition of reuse distance: *A 64B-aligned memory line is referenced with reuse distance d, where d is the number of unique memory lines referenced since the last reference to that memory line [33]*. From this definition it follows that a reference must be satisfied by a fully-associative LRU cache with capacity greater than or equal to the reuse distance (a compulsory miss has infinite reuse distance).

We use a Pin tool [34] to collect reuse distance data for each benchmark of the workload. During execution Pin supplies the address stream of an application to a library that collects a histogram of reuse distances. Each element of the histogram represents the number of references within a reuse distance range that corresponds to a cache size in our design space. For convenience, we used bins with ranges matching the points of our cache configuration space with an additional "*anything greater than"* bin to capture distances outside the design space (beyond 64MB and to infinity for compulsory misses).

The reuse distance distribution provides an estimate for performance and energy usage for a given cache configuration based on the hits and misses at each level of cache and memory as represented by the histogram. From the histogram we infer hit counts assuming that references in a bin for distances up to *d* will hit the smallest cache with size greater than or equal to *d*; if no such cache level exists then those are counted as references to DRAM. Equation (1) defines hits by level for caches and DRAM, with the number of bins, bin upper bound, and bin reference count represented by B, U(j), and R(j) respectively [33]. For simplicity, in Equation (1) it is implied that h(0)=0 and that S(L+1)=∞

$$(1)\quad h(i) = \sum_{\{j \mid U(j) \le S(i) \wedge U(j) > S(i-1)\}} R(j)$$

Finally, the same models as in the simulation-based selection are applied to identify an optimal configuration.

We evaluate the selection by comparing the energy savings to the savings previously obtained while preserving performance, by assuming leakage reduction, and within the *Reconfigurable* search space. The reuse distance-based selection achieves comparable savings in most applications, and sometime even higher savings, as shown in Figure 6. While this is a successful result, it also shows a side effect of the approximation error in using reuse distance to predict cache behavior. The reuse distance-based selection makes a choice based on a different performance estimate than the simulation-based selection; as a result, reuse distance may select configurations that were otherwise discarded by the simulation-based selection due to performance degradation. In fact, the reuse distance based selection results in slightly lower average speedup (1.19 vs. 1.21) and a significant slowdown in some of the applications.

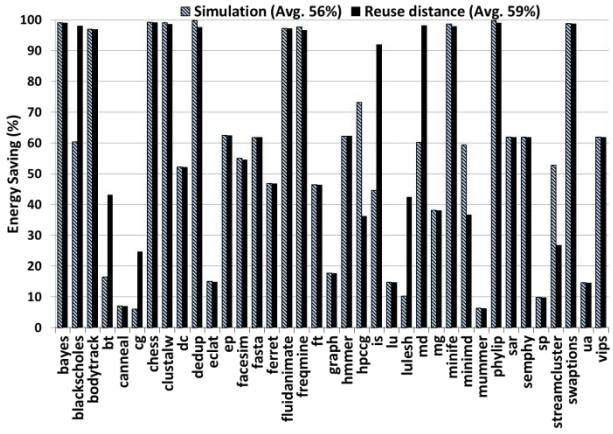

Figure 6. Comparing the selection based on exhaustive search vs. Reuse distance-based selection.

Figure 7 shows the difference in energy savings and the relative speedup of the selection based on reuse distance. The selection is different in 16 out of the 37 benchmarks, and while the average savings and performance are similar, there are significant differences in some of the 16 benchmarks, namely *blackscholes, bt, cg, is, lulesh,* and *md.* In these, the simulation correctly selects a configuration with the L3 cache half or fully powered, when the reuse distance selects a configuration with no L3 cache or a half powered L3 cache, respectively. As a consequence, reuse distance achieves a higher energy saving, but a lower performance.

To prevent mispredictions and preserve performance, a reduced configuration space could be used, for example a space in which caches can only be reduced to half size. In such 8-configuration space, the selection would be identical for both methods in 30 of the 37 benchmarks, with little difference in the other 7; both selections would achieve an average energy saving of 45%, and an average 1.25x speedup. This result indicates that, despite the inaccuracy due to the inherent approximation of the model, when the search space is sufficiently small reuse distance can effectively selects energy optimal configurations while preserving performance.

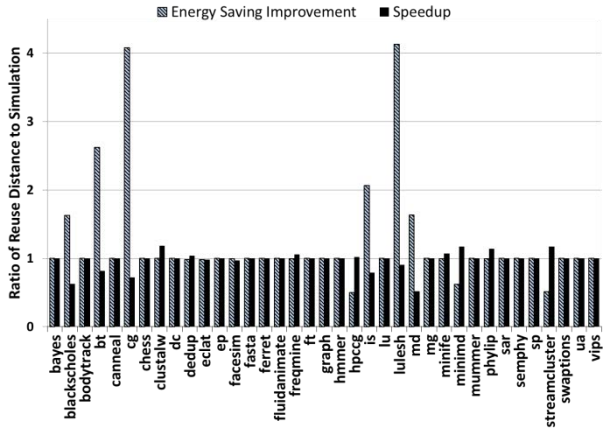

Figure 7. Comparing the selection based on exhaustive search vs. Reuse distance-based selection.

6. DISCUSSION AND RELATED WORK

Our results show that significant energy savings can be achieved if the memory hierarchy can be customized to the application's needs. The promise of reconfigurable memory hierarchies aim to deliver these potential benefits as the energy consumed by memory hierarchies is an increasingly important element of overall system energy. As such, our results cannot be fairly compared to the wealth of research on memory hierarchy optimization that targets a single fixed design and aspects of organization and cache management policies to increase performance and other attributes. This rich vein of research underlies the design of nearly all microprocessor designs today.

Numerous elements of memory hierarchy research employ elements of adaptation, a close relative to reconfiguration, to customize their behavior on a per-cache line basis or over short periods of time to better match application needs. Examples include adaptive caches [18, 19], adaptive protocols [35], and prefetching engines [12-14]. These systems typically focus on improving execution performance and perhaps secondarily reducing miss traffic. These contrast to our goal of exploiting reconfiguration in a memory hierarchy – per application -- to improve energy efficiency.

Work in the early 2000's [16, 17] explored processor, TLB, and cache structures to optimize energy efficiency, focusing on dynamic energy, in an era of higher clock rates, much smaller caches, and less leakage. As a result, reconfiguration focused on access method (sequential way access), tags then array, and even TLB energy optimization. We focus on cache block/array level reconfiguration, enabling us to optimize dynamic energy, leakage, and even the remaining leakage after aggressive leakage reduction has been pursued. In short, for current device technology (billions of transistors operating at a few GHz) and

chip-scale (16MB caches), access latencies are less critical and not using large sections of a chip ("dark silicon") is conceivable. Remarkably, turning entire sections of caches off in many cases reduces overall energy consumption.

The feasibility of cache reconfiguration is well-established with studies going back over a dozen years [16-18] and the well-known work on smart memories as a landmark effort [36]. In fact, recent proposals [20] go even further, unifying register file, scratch, and cache memories that require configuration deep into the core, substantiating the viability of reconfigurable structures at the highest levels of the memory hierarchy.

Reuse distance is a well-established and widely-used platform-independent approach to characterizing application data locality and reference structure [33, 35, 37-46]. It has been used to model program locality [37, 39], to select ISA-specific hints for locality [40], to predict performance [41], and to improve locality by code reordering [38, 42, 44]. More recently, reuse distance has been extended to model cache sharing and interference in multi-core processors [43, 45]. Finally, as a way to drive the design of performance optimal caches, reuse distance has been used to select sharing and size of LLC [46]. Ours is yet another use of reuse distance – to select the configuration of a reconfigurable memory hierarchy.

Modern memory hierarchies have grown to megabytes [26] and combined with deep submicron CMOS technologies, leakage (or static) power has become an important concern. As we discussed in Section 3.4, a wealth of circuit techniques have been developed to reduce the leakage energy penalty in large on-chip caches, including drowsy caches and transistor engineering [18], cache decay [19], etc. These techniques have been shown to reduce leakage energy by 50-70%, so we have explored a set of possible design points exploiting these techniques. These studies show that such reductions in leakage power can affect the energy savings modestly, but they do not strongly affect the selection of optimal hierarchy, nor the relative benefit. Given the poorer voltage scaling of SRAM cells due to instability, we expect leakage power is likely to remain an important factor in memory hierarchy energy.

7. SUMMARY AND FUTURE WORK

Our results show that the energy benefits of reconfigurable memory hierarchies can be significant, ranging from 70% for extreme reconfigurability to 56% for simple reconfigurability; all while maintaining performance. Further, simulations show that these benefits are robust across a range of cache energy models, and that amongst the constrained memory hierarchy spaces, characteristics such as reuse distance can reliably select a close-to-optimal memory hierarchy configuration.

While we have studied the potential energy benefits of a reconfigurable memory hierarchy, a per-application phase approach may take advantage of different locality structures within an application and reveal even greater potential energy benefits. In addition, much work remains to explore the detailed design and implementation of such memory hierarchies. For example, studies of the interplay between a wide range of cache features (sophisticated replacement, associativity, write policy, etc.) and the reconfigurable structure should be studied. Further, a study of the impact of parallel (multi-core) access is clearly important. Finally, while our results show the promise of reuse distance to automatically select cache configuration, a wealth of static and dynamic techniques bear investigation to reduce the cost and increase the accuracy of automatic reconfiguration.

8. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under award NSF OCI-10-1057921 and the Defense Advanced Research Projects Agency under award HR0011-13-2-0014. This work was supported in part by the DOE Office of Science through the Advanced Scientific Computing Research (ASCR) award titled "Thrifty: An Exascale Architecture for Energy-Proportional Computing". The contents do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the national Science Foundation grant number OCI-1053575.

9. REFERENCES

[1] R. H. Dennard, F. H. Gaensslen, H.-N. Yu, V. L. Rideout, E. Bassous, and A. R. Leblanc, "Design Of Ion-implanted MOSFET's with Very Small Physical Dimensions," *Proceedings of the IEEE,* vol. 87, pp. 668-678, 1999.

[2] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger, "Power Limitations and Dark Silicon Challenge the Future of Multicore," *ACM Transactions on Computer Systems (TOCS),* vol. 30, p. 11, 2012.

[3] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM,* vol. 54, pp. 67-77, 2011.

[4] ITRS. (2012). *2012 International Technology Roadmap for Semiconductors*. Available: http://www.itrs.net/Links/2012ITRS/Home2012.htm

[5] S. Gupta, S. Feng, A. Ansari, S. Mahlke, and D. August, "Bundled execution of recurring traces for energy-efficient general purpose processing," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, 2011, pp. 12-23.

[6] V. Govindaraju, C. Ho, T. Nowatzki, J. Chhugani, N. Satish, K. Sankaralingam*, et al.*, "Dyser: Unifying functionality and parallelism specialization for energy efficient computing," 2012.

[7] G. Venkatesh, J. Sampson, N. Goulding-Hotta, S. K. Venkata, M. B. Taylor, and S. Swanson, "QsCores: trading dark silicon for scalable energy efficiency with quasi-specific cores," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, 2011, pp. 163-174.

[8] "OMAP 5 Mobile Application Platform," *Texas Instruments*.

[9] "Introducing Tegra 4, Worlds Fastest Mobile Processor," *NVIDIA*.

[10] W. A. Wulf and S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," *Computer Architecture News,* vol. 23, pp. 20-24, 1995.

[11] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*: Morgan Kaufmann, 2011.

[12] I. Chung, C. Kim, H.-F. Wen, and G. Cong, "Application data prefetching on the IBM blue gene/Q supercomputer," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, 2012, pp. 1-8.

[13] S. P. Vanderwiel and D. J. Lilja, "Data prefetch mechanisms," *ACM Computing Surveys (CSUR),* vol. 32, pp. 174-199, 2000.

[14] T.-F. Chen, "An effective programmable prefetch engine for on-chip caches," in *Proceedings of the 28th annual international symposium on Microarchitecture*, 1995, pp. 237-242.

[15] P. Kogge, "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems," *CSE Dept. Tech. Report TR-2008-13,* 2008.
[16] D. H. Albonesi, "Selective cache ways: On-demand cache resource allocation," in *Microarchitecture, 1999. MICRO-32. Proceedings. 32nd Annual International Symposium on*, 1999, pp. 248-259.
[17] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas, "Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures," in *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, 2000, pp. 245-257.
[18] N. S. Kim, K. Flautner, D. Blaauw, and T. Mudge, "Single-v DD and single-v T super-drowsy techniques for low-leakage high-performance instruction caches," in *Proceedings of the 2004 international symposium on Low power electronics and design*, 2004, pp. 54-57.
[19] Z. Hu, S. Kaxiras, and M. Martonosi, "Let caches decay: reducing leakage energy via exploitation of cache generational behavior," *ACM Transactions on Computer Systems (TOCS),* vol. 20, pp. 161-190, 2002.
[20] M. Gebhart, S. W. Keckler, B. Khailany, R. Krashinsky, and W. J. Dally, "Unifying Primary Cache, Scratch, and Register File Memories in a Throughput Processor."
[21] S.-H. Yang, B. Falsafi, M. D. Powell, K. Roy, and T. N. Vijaykumar, "An Integrated Circuit/Architecture Approach to Reducing Leakage in Deep-Submicron High-Performance I-Caches," presented at the Proceedings of the 7th International Symposium on High-Performance Computer Architecture, 2001.
[22] P. Ranganathan, S. Adve, and N. P. Jouppi, "Reconfigurable caches and their application to media processing," *SIGARCH Comput. Archit. News,* vol. 28, pp. 214-224, 2000.
[23] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, and N. P. Jouppi, "A Comprehensive Memory Modeling Tool and Its Application to the Design and Analysis of Future Memory Hierarchies," presented at the Proceedings of the 35th Annual International Symposium on Computer Architecture, 2008.
[24] T. Vogelsang, "Understanding the Energy Consumption of Dynamic Random Access Memories," in *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM International Symposium on*, 2010, pp. 363-374.
[25] M. Laurenzano, M. Tikir, L. Carrington, and A. Snavely, "PEBIL: Efficient Static Binary Instrumentation for Linux.," presented at the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), White Plains, NY, 2010.
[26] "The Sandy Bridge E Processor," *Intel*.
[27] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, L. R. Carter, L. Dagum*, et al.*, "NAS Parallel Benchmark Results," *International Journal of Supercomputing Applications,* vol. 5, 1991.
[28] C. Bienia, "Benchmarking Modern Multiprocessors," Princeton University, 2011.
[29] M. A. Heroux, D. W. Doefler, P. S. Crozier, J. Willenbring, M., C. H. Edwards, A. Williams*, et al.*, "Improving Performance via Mini-applications," Sandia National Laboratory2009.
[30] DARPA, "Ubiquitous High Performance Computing (UHPC)."
[31] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, and A. Choudhary, "MineBench: A Benchmark Suite for Data Mining Workloads," in *Workload Characterization, 2006 IEEE International Symposium on*, 2006, pp. 182-188.
[32] K. Albayraktaroglu, A. Jaleel, X. Wu, M. Franklin, B. Jacob, C. W. Tseng*, et al.*, "BioBench: A Benchmark Suite of Bioinformatics Applications," in *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on*, 2005, pp. 2-9.
[33] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation techniques for storage hierarchies," *IBM Syst. J.,* vol. 9, pp. 78-117, 1970.
[34] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney*, et al.*, "Pin: building customized program analysis tools with dynamic instrumentation," *SIGPLAN Not.,* vol. 40, pp. 190-200, 2005.
[35] G. Kurian, O. Khan, and S. Devadas, "The Locality-Aware Adaptive Cache Coherence Protocol," in *International Symposium on Computer Architecture (ISCA) (to appear)*, 2013.
[36] M. Ken, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz, "Smart Memories: a modular reconfigurable architecture," in *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, 2000, pp. 161-171.
[37] C. Ding and Y. Zhong, "Predicting whole-program locality through reuse distance analysis," presented at the Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation, San Diego, California, USA, 2003.
[38] K. Beyls and E. H. D'Hollander, "Platform-independent cache optimization by pinpointing low-locality reuse," in *International Conference On Computational Science*, 2004, pp. 463-470.
[39] K. Beyls and E. H. D'Hollander, "Reuse Distance as a Metric for Cache Behavior," in *In Proceedings of the IASTED Conference on Parallel and Distributed Computing and Systems*, ed, 2001, pp. 617-662.
[40] K. Beyls and E. H. D'Hollander, "Reuse Distance-Based Cache Hint Selection," presented at the Proceedings of the 8th International Euro-Par Conference on Parallel Processing, 2002.
[41] G. Marin and J. Mellor-Crummey, "Cross-architecture performance predictions for scientific applications using parameterized models," *SIGMETRICS Perform. Eval. Rev.,* vol. 32, pp. 2-13, 2004.
[42] G. Marin and J. Mellor-Crummey, "Pinpointing and Exploiting Opportunities for Enhancing Data Reuse," in *Performance Analysis of Systems and software, 2008. ISPASS 2008. IEEE International Symposium on*, 2008, pp. 115-126.
[43] X. Xiang, B. Bao, T. Bai, C. Ding, and T. Chilimbi, "All-window profiling and composable models of cache sharing," presented at the Proceedings of the 16th ACM symposium on Principles and practice of parallel programming, San Antonio, TX, USA, 2011.
[44] C. Ding and K. Kennedy, "Improving cache performance in dynamic applications through data and computation reorganization at run time," *SIGPLAN Not.,* vol. 34, pp. 229-241, 1999.
[45] D. L. Schuff, B. S. Parsons, and V. S. Pai, "Multicore-aware reuse distance analysis," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, 2010, pp. 1-8.
[46] M.-J. Wu and D. Yeung, "Identifying optimal multicore cache hierarchies for loop-based parallel programs via reuse distance analysis," presented at the Proceedings of the 2012 ACM SIGPLAN Workshop on Memory Systems Performance and Correctness, Beijing, China, 2012.