
THE CACHE AND CODEC MODEL FOR STORING AND MANIPULATING DATA

THIS ARTICLE DESCRIBES AN ANALYTICAL MODEL OF SYSTEMS THAT STORE AND MANIPULATE DATA. THE CACHE AND CODEC MODEL CAPTURES THE INTERPLAY BETWEEN TWO CENTRAL ASPECTS OF SUCH SYSTEMS: INFORMATION REPRESENTATION AND DATA-PROCESSING EFFICIENCY. THIS ARTICLE DESCRIBES THE MODEL AND EXPLORES DESIGN OPTIONS FOR SYSTEMS THAT INCORPORATE DATA ENCRYPTION AND COMPRESSION. IN EACH CASE STUDY, THE AUTHORS RAISE SEVERAL RESEARCH IMPLICATIONS AND DIRECTIONS FOR FUTURE RESEARCH.

Van Bui
Martha A. Kim
Columbia University

..... The recent explosion of data has thrown traditional data management questions into stark relief. As the data growth rate outpaces technology scaling, computer systems designers will need to become increasingly strategic about what data they choose to store, where they store it, and in what representation (for example, compressed, encrypted, processed, or raw). Although modern computer systems span a range of scales and applications, at their core they all store and transform information. The *cache and codec model* that we developed captures, at a high abstraction level, the ways a computer system stores and transforms information. This simple model of how data flows through a system, potentially manipulated or transformed along the way, can help researchers and designers confront and explore a range of design options.

The cache and codec model represents the system as a set of interacting components, each of which has two properties. The first is the cost per bit of data processed. For a storage component, this might be the time or energy per bit read or written. For encryption,

this might be the time or energy per bit encrypted or decrypted. This term is directly linked to how a particular function is implemented. For example, DRAM requires more time and energy per bit read than static RAM (SRAM). Similarly, software encryption is slower and more power hungry than a hardware implementation.

The second property is bits of information per bit of data. Data's value is in the information it contains. Depending on how the information is encoded, the data might use more or fewer bits to represent it. A storage component that does not manipulate the data or change the representation would not change this term. In contrast, a lossless compressor would maintain the information while shrinking the data, thereby increasing this term.

Ultimately, what matters is the cost per bit of information produced—that is, $\frac{\text{cost}}{\text{bit}_{\text{info}}} = \frac{\text{cost}}{\text{bit}_{\text{data}}} \times \frac{\text{bit}_{\text{data}}}{\text{bit}_{\text{info}}}$. The cache and codec model captures the interplay between the data representation and the characteristics of

the components that manipulate it, thereby enabling high-level design-space explorations for systems that must manipulate large volumes of data efficiently.

In this article, we describe the cache and codec model, along with two case studies that use the model to explore the properties of encrypted and compressed storage systems.

Cache and codec model

We model a simple yet general class of computer systems consisting of a processor that draws data from a storage hierarchy. This storage hierarchy consists of caches that store information, and, optionally, codecs that re-encode data along the way.

Because the storage system is often a hierarchy, the model is an extension of the classic expression for the expected access time to the i th level of a cache hierarchy:

$$E_i = k \times C_i + R_i \times E_{i+1} \quad (1)$$

In this expression, k is the total number of bits accessed, and C_i is the access cost per bit at level i . There are two key conceptual differences with respect to the classic expression. First, each level of the hierarchy consists of either a cache (or other storage module) or a codec. In either case, the first term in Equation 1 captures the cost of accessing that module, whether it is storing or re-encoding data. Second, we have substituted R_i for the traditional miss rate in Equation 1. R_i is the “ratio” of a module, which for both caches and codecs, captures the module’s influence on the volume of data. In the case of a cache module, R_i is simply the miss rate, indicating that for every n bits requested from a cache, it is expected to request $R_i \times n$ bits from the lower levels of the hierarchy. For a $4\times$ compression module, the R value would be either 4 or 0.25, depending on the direction of the compression.

As a concrete example, consider a simple two-level hierarchy. In this scenario, Equation 1 expands to

$$E_1 = k \times C_1 + k \times R_1 \times C_2$$

The first term, $k \cdot C_1$, is the access cost to the first level; the second term, $k \cdot R_1 \cdot C_2$, is the access cost at level 2. The access cost at a

particular level, which we will call A_i , depends on the volume of data drawn from the module at the i^{th} level and the cost of doing so:

$$A_i = V_i \times C_i \quad (2)$$

where V_i is the expected volume of data drawn from a module in bits. The volume of data at each level is a function of k and the R values of the upper-level module(s) (for example, the number of accesses to DRAM depends on what is happening in the last-level cache [LLC] and above):

$$V_i = V_{i-1} \times R_{i-1} \quad (3)$$

The volume of data being accessed at the first level, V_0 , is simply k , which is an input to the model. In addition to k , the other inputs to the model are the order of the modules and an (R_i, C_i) pair for each module.

Figure 1 illustrates a slightly more complex example modeling memory compression. This hierarchy contains three levels of cache, backed by compressed memory. Thus, between the LLC and the memory sits a compressor/decompressor that compresses incoming data on writes and decompresses outgoing data on reads. The key to using the model is reasoning about appropriate R and C values for each module. The compressor modeled in this example compresses data by a factor of 4, so $R = 0.25$. Similarly, in this example we model an L2 with a 20-percent miss rate, so for that module $R = 0.20$.

Using Equation 3, we can calculate the expected volume of data drawn from each module, as illustrated by the dotted red lines in Figure 1. Then, using those volume estimates and the efficiency parameter for each module, C , we can calculate the cost contribution for each module according to Equation 2 and shown with the dashed lines in Figure 1.

Although the case studies that follow center on big data, the model is applicable to various other systems where data gets manipulated and stored in some form or fashion.

Case studies

We present two case studies of the cache and codec model for encrypted and compressed storage systems, demonstrating the sort of design space explorations that it supports.

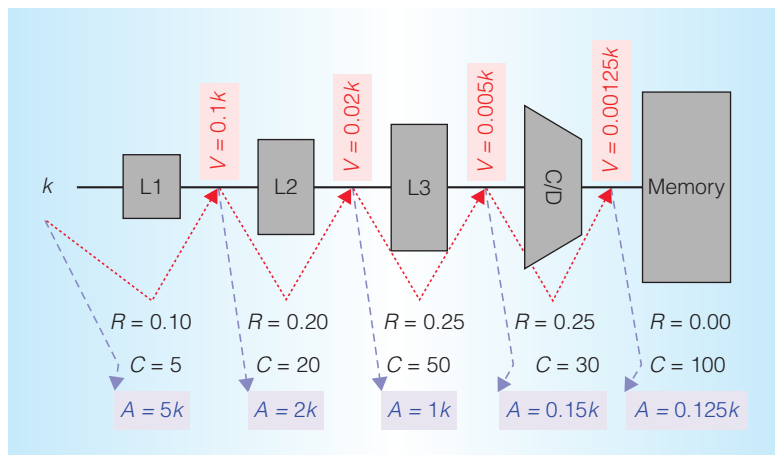


Figure 1. Illustration of how the cache and codec model captures the primary features of a memory compression system, namely the compression's impact on the volume of data stored in memory and its cost relative to the cost of the system's other components.

Data encryption

Big data is a rich target for hackers because of the sheer amount of information stored on those servers, with high-profile data breaches or unauthorized accesses regularly reported. Even RSA, a company with expertise in data security technology, is not immune to these breaches.¹ Applying data encryption to big-data storage systems could be a second line of attack for security. This not only is a deterrent to hackers, but also can minimize the damage should a breach occur.

The sooner or higher in the memory hierarchy that data is encrypted, the more security this technique offers. However, early encryption can also be more costly, because data in the upper levels of a hierarchy is accessed, by design, more often, and thus will be encrypted and decrypted more often. There is thus a tradeoff between complexity, cost, and security.

In this case study, we use the cache and codec model to analyze the energy cost associated with encrypting data in the LLC, memory, and nonvolatile storage. For each of the three levels, we examine several technologies, as summarized in Table 1. The table also lists the R and C parameters for each cache and codec used in this analysis.

We use the SPEC OMP benchmarks, which have an average dataset size of 31.09 Mbytes, and an average LLC miss rate of 36

percent on Intel Dunnington.² The page fault rates were estimated to be an average of typical virtual-memory miss rates on systems with page sizes ranging from 4 to 64 Kbytes.³ The read and write energy efficiency values (C parameters) for LLC are from Chang et al.,⁴ and the values for memory and disk are from Ramos and Bianchini.⁵

LLC encryption. For each LLC encryption scenario, we consider three cache technologies. In the ideal case, the LLC will have both high density and low power. SRAM is the more traditional cache implementation with fast access times, but low density relative to other technology and high leakage current. Spin-transfer torque magnetic RAM (STT-RAM) is an alternative technology with higher density, but it has high write latency and write energy consumption. A third option is embedded DRAM (eDRAM), which also features high density and low leakage. A drawback of eDRAM is that it requires refresh operations, and this is a major source of power dissipation. Each of these LLC technologies has tradeoffs with respect to density and power.

Because the LLC is already power constrained, adding a data encryption module to the system could further increase total system energy costs. We use our cache and codec model to explore a system with an encryption unit followed by the LLC (SRAM, STT-RAM, or eDRAM), DRAM, and finally a hard disk. We are interested in understanding the energy efficiency that data encryption technologies should achieve in order to prevent significant increases in total energy cost in these systems.

Figure 2a shows the total access energy as a function of data encryption and decryption energy, ranging from an idealized point where encryption is free to it costing 10 nJ/bit. We observe that regardless of the LLC technology, energy overheads can be made quite low provided the encryption is implemented efficiently (at 0.2 nJ/bit or lower), but that beyond that point, the overhead will rise very rapidly to as much as 251 percent when encryption/decryption runs at 10 nJ/bit. Of the three locations to place an encryption module, before the LLC is, not surprisingly, the most sensitive to encryption efficiency.

Table 1. Technology parameters for the encryption case study.

Level	Technology	R	C_{read}	C_{write}	Source
Last-level cache (LLC)	SRAM	0.36	2.10 nJ/block	2.21 nJ/block	Zhang et al. ² and Chang et al. ⁴
	STT-RAM	0.36	0.94 nJ/block	20.25 nJ/block	Zhang et al. ² and Chang et al. ⁴
	eDRAM	0.36	1.74 nJ/block	1.79 nJ/block	Zhang et al. ² and Chang et al. ⁴
Memory	PCM	5.05e-6	4.94 pJ/bit	33.64 pJ/bit	Hennessey et al. ³ and Ramos et al. ⁵
	DRAM	5.05e-6	1.17 pJ/bit	0.39 pJ/bit	Hennessey et al. ³ and Ramos et al. ⁵
Disk	SSD	0	150 μ J/4KB	\sim 150 μ J/4KB	Ramos et al. ⁵
	HDD	0	16.8 mJ/4KB	\sim 16.8 mJ/4KB	Ramos et al. ⁵

.....
 *eDRAM: embedded DRAM; HDD: hard disk drive; PCM: phase-change memory; SRAM: static RAM; SSD: solid-state drive; STT-RAM: spin-transfer torque magnetic RAM.

Memory encryption. For memory encryption, we evaluate DRAM and phase-change memory (PCM) memory using the cache and codec model. PCM is being explored as an alternative to DRAM because it is denser and has a lower idle power. Although DRAM has lower access times, PCM does not require refreshing. Given these tradeoffs, researchers have proposed hybrid PCM-DRAM systems.⁸

We do not expect the energy overhead of memory encryption to be as high as at the LLC, because of the less-frequent memory reads and writes that require encryption and decryption. Nonetheless, it is important to test this hypothesis. Figure 2b shows the data access energy for encrypted memory as a function of the encryption efficiency. As with the LLC, the total energy cost differs little for PCM and DRAM. To keep energy costs from increasing above 5 percent, encryption efficiency should be less than 0.6 nJ/bit, 3 times less efficient than the 0.2 nJ/bit required for 5 percent overhead with LLC encryption. On the other hand, the overhead does not rise as steeply as at the LLC, with an 88 percent overhead when encryption requires 10 nJ/bit.

Nonvolatile storage encryption. We evaluate two encrypted nonvolatile storage media, solid state drives (SSDs) and hard disk drives (HDDs). SSDs offer roughly three orders of magnitude lower access latency than HDDs,

and are also more energy efficient. However, they have lower capacity per dollar,⁹ creating a tradeoff between performance, energy, and cost. We focus on evaluating the energy costs with the cache and codec model, but we could adjust C model parameters to analyze performance and dollar costs as well. Examining Figure 2c, we note that encryption at this level is far less sensitive to encryption efficiency than the other two, suggesting that lower cost, possibly even software encrypters, would be sufficient. Although the energy overhead for both SSDs and HDDs is just a fraction of a percent, the total SSD energy costs are 92 percent lower than for HDDs, which is what we would expect to find when analyzing the energy dimension of the tradeoff.

The energy efficiency for various software AES implementations can vary between 1.55 to 28 nJ/bit depending on how optimized the design is.¹⁰ For ASIC AES-128 implementations, energy efficiency can range from 0.42 pJ/bit to 0.18 nJ/bit.¹¹ Our models suggest that hardware-based AES encryption is more suitable for LLC and memory encryption in order to minimize energy overheads. At 1.55 nJ/bit energy efficiency, software AES would have to improve energy efficiency by at least 87 percent for LLC and 61 percent in the case of memory encryption to keep energy overheads low.

Combining data encryption with compression can provide strong protection for data,

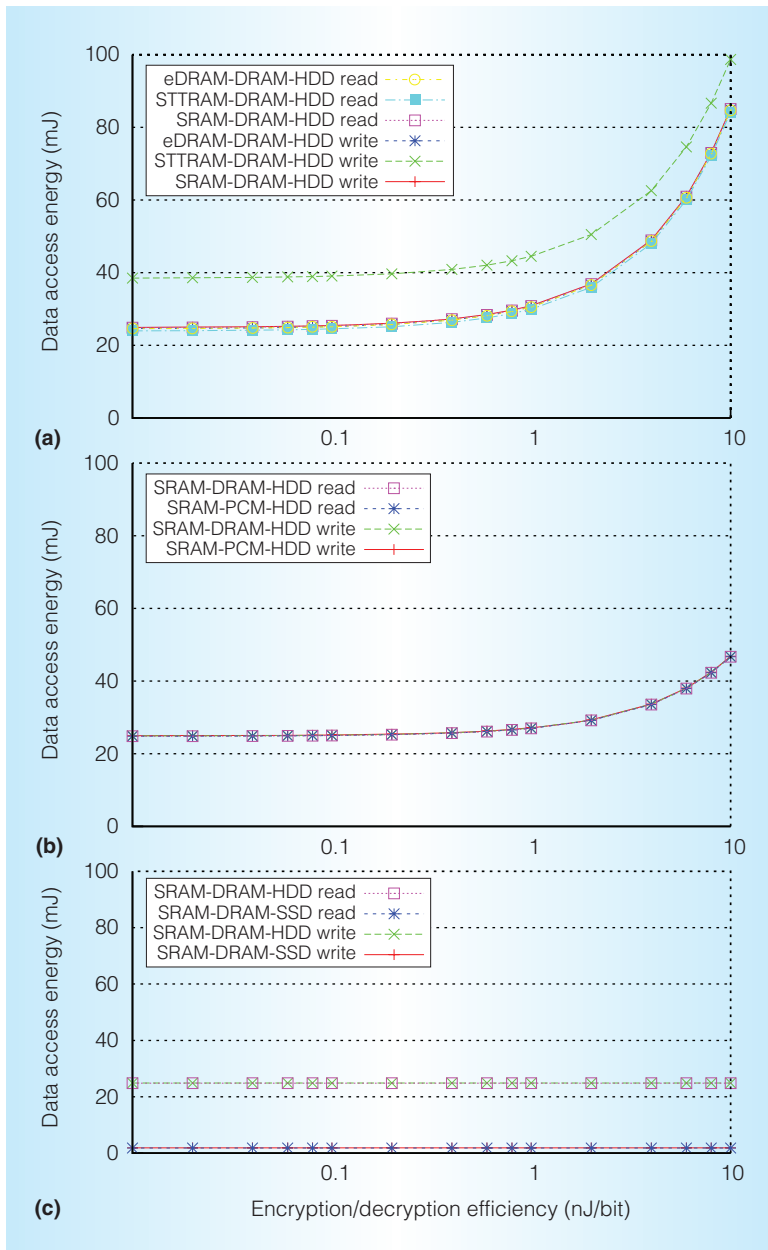


Figure 2. Encrypted data access costs as a function of encryption/decryption efficiency. Although efficient encryption and decryption can keep the overheads low regardless of placement, the encrypted LLC (a) is the most sensitive to efficiency compared to the encrypted memory (b) and encrypted disk (c).

within the performance constraints required to support big data analytics. We consider data compression in our second case study.

Data compression

Memory compression is not a new technology; it dates back to the HP Omnibook

300 introduced in 1993. Thanks to ever-growing data sizes and concerns about energy consumption, cache and memory compression are once again a topic of great interest, with significant academic scrutiny.^{6,12}

The space of potential compressed storage designs is vast. Which modules in the memory hierarchy should store compressed or uncompressed data? How will each choice impact performance and energy? Does the answer differ according to the anticipated workload? Should compression be done in hardware or software? Will new technologies such as 3D stacking and embedded DRAM make a difference? These questions, and others, should be evaluated during the design phase. The cache and codec model can capture the first-order effects of such systems and help designers perform the necessary high-level, path-finding design-space explorations. In this case study, we demonstrate how the cache and codec model can be used to explore extensions of Co-DCC,⁶ a recently proposed compressed LLC.

Decoupled Compressed Cache (DCC) and an optimized design called Co-Compacted DCC (Co-DCC) seek to increase effective cache capacity at area overheads comparable to previous compressed caches.⁶ DCC reduces area overheads using superblocks, which contain contiguous cache blocks that share a single address tag. Fragmentation is reduced in a superblock by decoupling the address tags, allowing subblocks in a set to map to any tag in that set. Co-DCC further optimizes DCC to reduce fragmentation by compacting the blocks in a superblock into the same set of sub-blocks. DCC increases normalized effective capacity by 2.2 times, whereas Co-DCC increases this average by 2.6 times for several workloads.⁶ Co-DCC addresses the common issues in cache compression design, including internal fragmentation and high area overheads, and can improve both performance and energy efficiency across a range of workloads.

Using the cache and codec model, we explore what would happen if the Co-DCC techniques were applied to other caches in addition to the LLC. As in Figure 1, each component in the system, including the compressor/decompressor (C/D), has an associated (R, C) pair.

Table 2. Experimental R and C values for the compression case study.

Level	R	C_{latency}	C_{energy}	Source
Compressor/ decompressor (C/D)	0.26	9 cycles decompression, 16 cycles compression	0.05 nJ/block decompression, 0.13 nJ/block compression	Sardashti et al. ⁶ and Muralimanohar et al. ⁷
Level 1 (L1)	0.06	2 cycles	0.17 nJ/block	Sardashti et al. ⁶ and Muralimanohar et al. ⁷
Level 2 (L2)	0.43	10 cycles	0.32 nJ/block	Sardashti et al. ⁶ and Muralimanohar et al. ⁷
Level 3 (L3)	0.36	30 cycles	1.47 nJ/block	Sardashti et al. ⁶ and Muralimanohar et al. ⁷
Memory	0.00	160 cycles	60.35 nJ/block	Sardashti et al. ⁶ and Muralimanohar et al. ⁷

The R values for the caches and memory are the miss rates, whereas for the C/D module the R value is the compression factor (that is, the compressed size over the original size). Co-DCC uses the CPACK+Z dictionary-based compression algorithm, which combines CPACK and the zero-block detection algorithm. The compression factor for CPACK+Z is on average 0.26 with a decompression latency of nine cycles.⁶ Cache compression lowers the cache miss rate, which is captured in the R values for the storage modules containing compressed data. In particular, Co-DCC achieves on average 24 percent lower LLC miss rate across a range of workloads,⁶ so the R values of any compressed caches are tuned accordingly:

$$R_i = (1 - P_c) \times M,$$

where P_c is the percentage decrease in miss rate (that is, 24 percent), and M is the original miss rate.

In this case study, we set k to be the average data size for the SPEC OMP benchmark suite, which is 31.09 Mbytes.² To explore the system's dynamics, we model the average miss rates ranging from 0 to 100 percent. Additionally, we model the average miss rates (R parameter) for the data storage modules using SPEC OMP running on Intel's Dunnington.² The R parameter for the C/D module is based on the compression factor of

CPACK+Z used in the implementation of Co-DCC.⁶

The C values are either the per-bit time or energy cost for each module in the system. We model the same system as in the original Co-DCC experiments, a 3.2-GHz processor with a 32-Kbyte private L1 cache, 256-Kbyte private L2 cache, 8-Mbyte shared L3 cache, and 4 Gbytes of main memory.⁶ We use Cacti to derive both the energy and latency cost of each data storage module.⁷ The energy and latency cost for the C/D module are nine cycles and 0.05 nJ per bit for CPACK+Z.⁶ Table 2 summarizes the R and C parameters for this experiment.

We use the cache and codec model to analyze four systems that differ in terms of where the C/D is placed in the hierarchy, thereby varying how much of the hierarchy caches compressed versus uncompressed data, and how much data is expected to flow through the compressor/decompressor.

Figure 3 shows the time and energy breakdown for each configuration, normalized to the access time and energy of a hierarchy with no compression.

Examining both access time and energy, we observe that the higher in the storage hierarchy the compressor is placed, the more time (or energy) is spent on compression. This is because the higher in the hierarchy the compressor sits, the more data it is called on to compress or decompress, thus

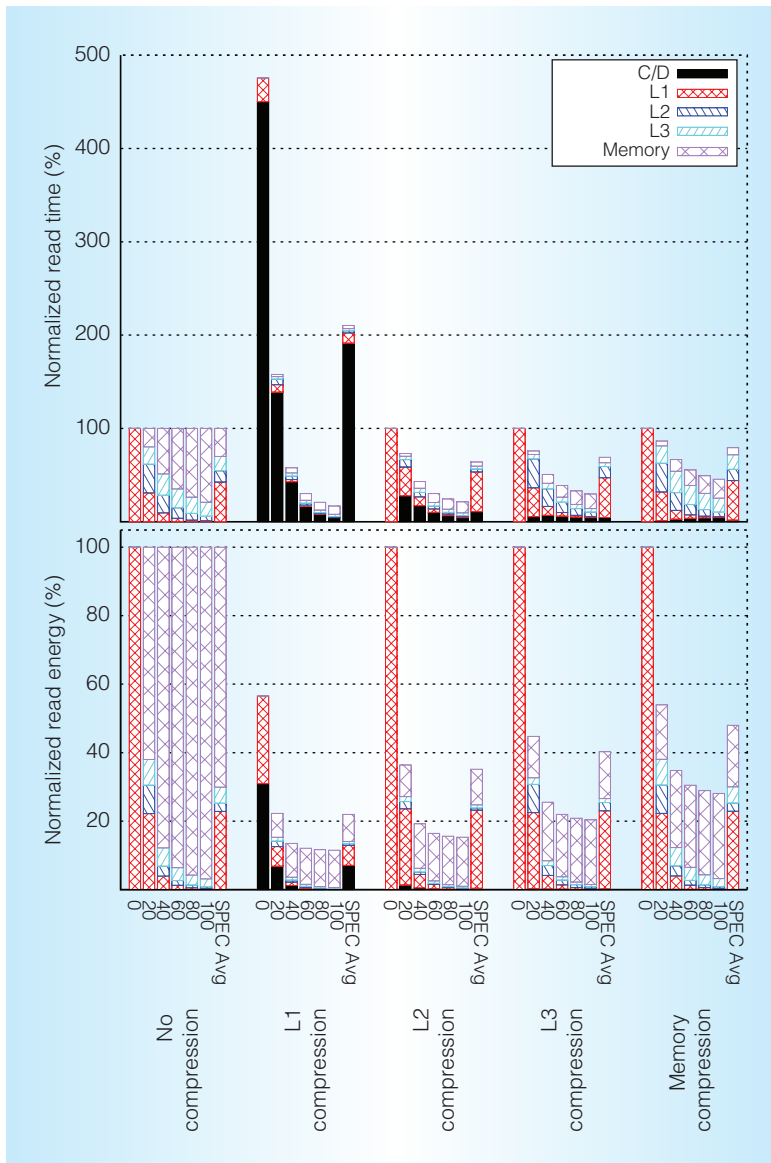


Figure 3. Time and energy breakdown for different cache miss rates and data compressor locations in the memory hierarchy for Co-Compacted Decoupled Compressed Cache (Co-DCC) compression. Data compression at L1 has excessively high performance costs for memory-efficient workloads, whereas energy costs are more manageable.

increasing its cost. When miss rates are low, this is not offset by savings from the lower levels of the hierarchy caching compressed data. However, when miss rates are high and more data is flowing through the lower levels, the savings due to compression offset its cost. We observe that the higher the miss rates, the higher in the hierarchy the compressor

should be placed. Although the focus to date has been on memory and LLC compression, this exploration suggests that there are many circumstances in which pulling the compressor higher into the storage hierarchy could be beneficial. Compression might even be thought of as a way to mitigate high cache-miss rates, and, using compression, a system could perhaps substitute smaller or lower-associativity (and lower-power) caches without impacting performance.

It is also instructive to compare and contrast the performance and energy trends. For example, although Figure 3 suggests that L1 compression on workloads with very low miss rates is detrimental to performance, it is a clear win from an energy perspective, reducing it by almost half.

This article has presented the cache and codec model for the storage and manipulation of data. As data volumes and system design spaces grow, thanks in part to new algorithms and technologies, high-level path finding will increase in importance. This model provides one tool for this purpose. Thanks to its generality, it can be adapted and applied to a broad range of systems and applications beyond the two described here.

MICRO

References

1. J. Markoff, "SecurID Company Suffers a Breach of Data Security," *New York Times*, 17 Mar. 2011; www.nytimes.com/2011/03/18/technology/18secure.html.
2. Y. Zhang, M. Kandemir, and T. Yemliha, "Studying Inter-core Data Reuse in Multicores," *Proc. ACM SIGMETRICS Joint Int'l Conf. Measurement and Modeling of Computer Systems*, 2011, pp. 25–36.
3. J.L. Hennessy and D.A. Patterson, *Computer Architecture, Fifth Edition: A Quantitative Approach*, 5th ed., Morgan Kaufmann, 2011.
4. M.-T. Chang et al., "Technology Comparison for Large Last-Level Caches (L3Cs): Low-Leakage SRAM, Low Write-Energy STT-RAM, and Refresh-Optimized


eDRAM," *Proc. IEEE 19th Int'l Symp. High Performance Computer Architecture*, 2013, pp. 143–154.

5. L. Ramos and R. Bianchini, "Exploiting Phase-Change Memory in Cooperative Caches," *Proc. IEEE 24th Int'l Symp. Computer Architecture and High Performance Computing*, 2012, pp. 227–234.
6. S. Sardashti and D.A. Wood, "Decoupled Compressed Cache: Exploiting Spatial Locality for Energy-Optimized Compressed Caching," *Proc. 46th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2013, pp. 62–73.
7. N. Muralimanohar, R. Balasubramonian, and N. Jouppi, *CACTI 6.0: A Tool to Model Large Caches*, tech. report HPL-2009-85, Hewlett-Packard, 2009.
8. M.K. Qureshi, V. Srinivasan, and J.A. Rivers, "Scalable High Performance Main Memory System using Phase-Change Memory Technology," *Proc. 36th Ann. Int'l Symp. Computer Architecture*, 2009, pp. 24–33.
9. D. Narayanan et al., "Migrating Server Storage to SSDs: Analysis of Tradeoffs," *Proc. 4th ACM European Conf. Computer Systems*, 2009, pp. 145–158.
10. B. Liu and B.M. Baas, "Parallel AES Encryption Engines for Many-Core Processor Arrays," *IEEE Trans. Computing*, vol. 62, no. 3, 2013, pp. 536–547.
11. H.W. Chung, "An Energy Efficient AES Engine with DPA-Resistance," PhD dissertation, Dept. Electrical Eng. and Computer Science, Massachusetts Institute of Technology, 2009.
12. A. Shafiee et al., "Memzip: Exploiting Unconventional Benefits from Memory Compression," to be published in *Proc. 20th Int'l Symp. High-Performance Computer Architecture*, 2014.

Van Bui is a PhD candidate in the Computer Science Department at Columbia University. Her research interests include computer architecture, parallel computing, and high-performance and energy-efficient computing. Bui has an MS in computer science from the University of Houston. She is a member of IEEE.

Martha A. Kim is an assistant professor in the Computer Science Department at Columbia University. Her research interests include computer architecture, parallel hardware and software systems, and energy-efficient computation on big data. Kim has a PhD in computer science from the University of Washington. She is a member of IEEE and the ACM.

Direct questions and comments about this article to Martha Kim, 450 Computer Science Building, 1214 Amsterdam Ave., Mailcode: 0401, New York, NY 10027-7003; martha@cs.columbia.edu.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.



NEW
IEEE  **computer society**
STORE

Find the latest trends and insights for your

- presentations
- research
- events

webstore.computer.org