Fat Caches for Scale-Out Servers

The authors propose a high-capacity cache architecture that leverages emerging high-bandwidth memory modules. High-capacity caches capture the secondary data working sets of scale-out workloads while uncovering significant spatiotemporal locality across data objects. Unlike state-of-the-art DRAM caches employing in-memory block-level metadata, the proposed cache is organized in pages, enabling a practical tag array, which can be implemented in the logic die of the high-bandwidth memory modules.

Stavros Volos
Microsoft Research

Djordje Jevdjic
University of Washington

Babak Falsafi
EPFL

Boris Grot
University of Edinburgh

Scale-out datacenters host a variety of data-intensive services, such as search and social connectivity. To concurrently support billions of users, latency-sensitive online services and analytic engines that create user-specific content (such as advertisements and recommendations) rely on large amounts of memory to minimize dataset access latency. The ever-growing popularity of the in-memory computing paradigm—which will be further broadened by the emergence of storage-class memory—leads to datacenter deployments in which memory accounts for a big share of the datacenter's total cost of ownership (TCO).[1]

Optimizing for a datacenter's TCO calls for customized architectures that maximize computational density. Following a considerable amount of research identifying the requirements of scale-out workloads, and indicating that these workloads benefit from thread-level parallelism and fast access to multimegabyte instruction footprints,[2,3] the industry has started employing specialized many-core processors with modestly sized last-level caches (such as Cavium ThunderX and EZchip Tile-MX) due to the substantial performance and TCO advantages offered by specialization.

Memory systems in scale-out servers are of paramount importance because they need to sustain the vast bandwidth demands of many-core chip multiprocessors (CMPs).[3,4] Recent advances in on-chip stacked DRAM technology eliminate the bandwidth bottleneck that plagues conventional DRAM.[5] This technology is capacity-limited owing to thermal constraints, so prior research advocates for using it as a cache to provide access to secondary data working sets.[4,6–8]

Our analysis shows that on-chip stacked DRAM caches are unattractive for scale-out servers. We find that memory accesses follow power-law distributions, so that a modest portion of memory (about 10 percent) accounts for the majority of accesses (65 to 95 percent). Thus, although the vast working sets of scale-out workloads are amenable to caching, high-capacity caches (tens of Gbytes) are required, given main memory sizes trending toward hundreds of Gbytes. The required cache capacities greatly exceed those of low-capacity caches, including on-chip stacked DRAM caches.

This article seeks to develop a scalable, high-capacity, and high-bandwidth memory system for scale-out servers by leveraging

Table 1. Requirements of one scale-out server.

	Processor		Memory system	
Year	**Cores**	**Bandwidth**	**Bandwidth**	**Capacity**
2015	96	115 Gbytes/s	288 Gbytes/s	384 Gbytes
2018	180	216 Gbytes/s	540 Gbytes/s	720 Gbytes
2021	320	384 Gbytes/s	960 Gbytes/s	1,280 Gbytes

emerging high-bandwidth memory modules as a high-capacity cache. High-bandwidth interconnect technologies allow for connecting the processor to multiple high-bandwidth memory modules via a silicon interposer (for example, Hynix High Bandwidth Memory [HBM]), thus forming an on-package cache, or via high-speed serial links (for example, Micron Hybrid Memory Cube [HMC]), thus forming an off-package cache.

In contrast to prior stacked DRAM cache proposals, which advocate for block-based[7,8] and sector-based organizations,[4,6] we find that page-based organizations are favored in scale-out servers. High-capacity caches—effective in capturing the secondary data working sets of scale-out workloads—uncover significant spatiotemporal locality across dataset objects due to long cache residency periods. The improved spatiotemporal locality allows for employing a page-based cache organization, thereby minimizing tag storage requirements and enabling a practical in-SRAM tag array architecture, which can be implemented in the logic die of the high-bandwidth memory modules. This design offers fundamental complexity advantages over state-of-the-art DRAM caches, which suffer from high tag and/or metadata overheads that mandate in-DRAM storage.

Emerging Scale-Out Servers and DRAM Technologies

In this section, we examine the memory requirements of emerging scale-out servers and review the features of emerging DRAM technologies.

Scale-Out Server Requirements

Processor and system vendors resort to many-core processors (such as Cavium ThunderX) to boost server throughput and rely on buffer-on-board chips (such as Cisco's extended memory technology[9]) to increase memory capacity. In doing so, datacenter operators can deploy fewer servers for the same throughput requirements and dataset size, thus lowering TCO significantly.[9,10]

We quantify the memory bandwidth and capacity requirements of emerging scale-out servers for various manufacturing technologies in Table 1. Our configuration maximizes throughput by integrating the maximum number of cores for a given die area and power budget of 250 to 280 mm^2 and 95 to 115 W. The modeled organization resembles that of many-core servers, such as Cavium ThunderX.

Bandwidth. We measure the processor's off-chip bandwidth demands by scaling per-core bandwidth consumption with the total number of cores. We measure per-core bandwidth by simulating a 16-core server and find that per-core bandwidth ranges from 0.4 to 1.2 Gbytes/second (GBps). Peak bandwidth demands are 115 GBps (2015), 216 GBps (2018), and 384 GBps (2021).

High bandwidth utilization levels can adversely impact end-to-end memory latency because of heavy contention on memory resources. Because performance of scale-out services is characterized by tail latencies, memory latency and queuing delays must be minimized. Thus, system designers overprovision memory bandwidth to ensure low utilization (less than 40 percent) and avoid queuing.[2] As such, memory systems need to supply 288 GBps (2015), 540 GBps (2018), and 960 GBps (2021). Such requirements exceed the capabilities of conventional DRAM systems by 5.5 to 7.5 times.

Capacity. We estimate required memory capacity by examining various system deployments. Today, data analytic engines are provisioned with 2 to 8 Gbytes per core (Cloudera), web search engines deploy 64 Gbytes for 16 cores (Microsoft Bing), and web and streaming servers require 1 to 2 Gbytes per core.[2] With the emergence of extended memory technology and storage-class memory, we anticipate that datacenter operators will continue deploying 4 Gbytes of per-core memory cost effectively, resulting in the deployment of several hundreds of Gbytes of memory per server.

Emerging DRAM Technologies

Stacked DRAM can provide an order of magnitude higher bandwidth than conventional DRAM due to dense through-silicon vias. It also offers low latency and low DRAM energy due to reduced wire spans and smaller page sizes. However, existing deployment options for stacked DRAM fail to satisfy the joint capacity, bandwidth, and power requirements mandated by scale-out servers. Next, we review the deployment options for stacked DRAM and their respective limitations.

On-chip and on-package stacked DRAM. Through-silicon vias provide high-bandwidth connectivity between the processor and on-chip stacked DRAM. Thermal constraints, however, limit the number of DRAM stacks that can be integrated on top of the processor, confining on-chip stacked DRAM to sizes that are two to three orders of magnitude smaller than the servers' memory capacity demands. Similarly, the high cost of big packages and area-intensive silicon interposers limit the number of stacked DRAM modules in on-package stacked DRAM systems. When combined with the thermally constrained capacity of a few Gbytes per module, an on-package DRAM solution fails to provide the requisite memory capacity for servers.

Off-package stacked DRAM. High-speed serial interfaces can break the bandwidth wall by connecting the processor to multiple off-package stacked DRAM modules. The high signal integrity of serial interfaces allows for achieving an order of magnitude higher data rates than double data rate (DDR) with the same number of pins.

Although off-package stacked DRAM systems deliver much greater memory capacity than on-chip and on-package stacked DRAM systems, two main factors prevent such systems from replacing conventional DRAM. First, serial channels impose high idle power because keep-alive packets must be sent at frequent intervals to maintain lane alignment across the channel's lanes. Second, thermal constraints limit the number of stacked layers per module and necessitate a blade-level network of these modules for a big-memory server. Such a network comes at the cost of high idle power consumption due to the use of many serial links resulting from a multihop chip-to-chip network.

State-of-the-Art DRAM Caches

Given the disparity between memory capacity requirements and the capacity provided by emerging DRAM technologies, most proposals advocate employing stacked DRAM as a cache to filter accesses to main memory. State-of-the-art cache proposals—leveraging mainly on-chip stacked DRAM—have to contend with relatively high miss rates owing to limited capacity. As a result, they are primarily optimized for low cache-memory bandwidth utilization through block-based organizations,[7,8] sector-based footprint-predicting organizations,[4,6] and address-correlated filter-based caching mechanisms.[11]

Unfortunately, such organizations come with high tag and/or metadata overhead and high design complexity, making such cache designs impractical. For instance, state-of-the-art block-based and footprint-predicting caches require 4 Gbytes and 200 Mbytes of tags, respectively, for a capacity of 32 Gbytes. Due to the prohibitive tag array overhead, recent proposals implement the tag array in DRAM.[6–8] In-DRAM tag arrays, however, require substantial engineering effort, making state-of-the-art caches less attractive. In addition, footprint-predicting caches rely on instruction-based prediction.[4,6] However, the program counter of an instruction is not available in the memory hierarchy, thus requiring the core-to-cache transfer of the program counter for all memory references, further increasing design complexity.

Memory Access Characterization of Scale-Out Servers

High-bandwidth memory modules are an ideal building block for a high-capacity, high-bandwidth cache. However, state-of-the-art DRAM caches are hindered by the need to keep metadata in DRAM. In this section, we study the application characteristics that enable architecting an effective, practical, and scalable cache.

Temporal Characterization

We examine the memory access distribution of scale-out applications by looking at the characteristics of the dominant types of memory accesses.

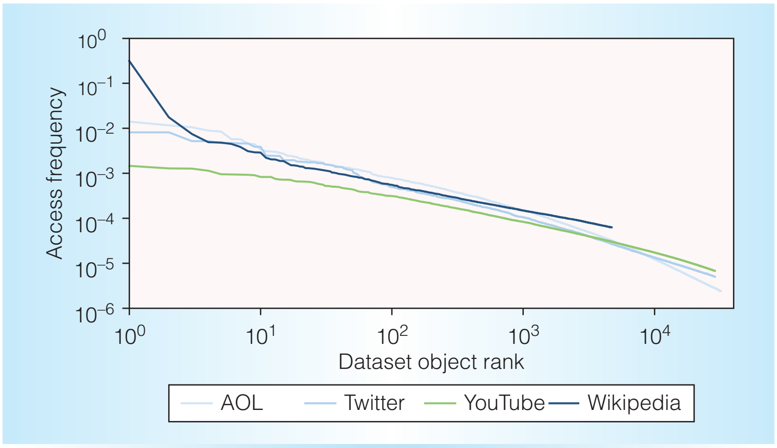

Figure 1. Dataset object popularity exhibits power-law distribution. Power-law relationships show linear trends in log-log scale.

Dataset accesses. We examine the dataset object popularity (that is, how frequently a dataset object is accessed) of search query terms (AOL), tweets (Twitter), videos (YouTube), and webpages (Wikipedia) based on publicly available data. Figure 1 plots the probability for a dataset object to be referenced as a function of popularity, showing that the dataset object popularity is highly skewed with a small set of dataset objects (10 to 20 percent) contributing to most of the dataset object accesses (65 to 80 percent). For instance, a small fraction of users and their pictures account for most of the user traffic in picture-sharing services, such as Flickr. Due to power-law popularity distributions, dataset accesses in data stores, object caching systems, streaming servers, web search engines, and web servers exhibit power-law distributions.

Accesses to dynamically allocated memory. Server applications frequently access dynamically allocated memory with high temporal reuse.

For instance, server applications use software caches to keep a set of hot objects, such as rows in data stores and compiled script code in web servers. As they host dataset-relevant data and metadata, the distributions of their accesses will follow those of the datasets. Another example is data structures employed by server applications or operating systems (OSs) per client or network connection, such as buffers for media packets in streaming servers and OS data structures storing TCP/IP state. The large number of concurrent connections in many-core CMPs results in a footprint that dwarfs on-chip cache capacity. The reuse of these structures is high because they are accessed multiple times during a connection.

We expect the skew in object popularity and temporal reuse of dynamically allocated memory to be mirrored in the memory access distribution. To confirm this, we examine the memory access distribution of a simulated 16-core scale-out server. To estimate the hot memory footprint of scale-out applications, we employ a state-of-the-art DRAM cache and measure its miss ratio for various capacities.[8]

Figure 2 plots the cache miss ratio for various cache-to-memory capacity ratios. The markers denote measurements while contiguous lines show *x*-shifted power-law fitted curves. The figure shows that memory accesses are skewed so that 6.25 to 12.5 percent of the memory footprint accounts for 65 to 95 percent of total accesses. The figure confirms that existing low-capacity caches (left points)—such as on-board SRAM caches (IBM Centaur), on-package embedded DRAM (eDRAM) caches, and on-chip stacked DRAM caches—cannot exploit temporal locality in scale-out servers. In extreme cases, such as Data Serving and Online Analytics, on-chip stacked DRAM caches are bandwidth-constrained with less than 40 percent of memory

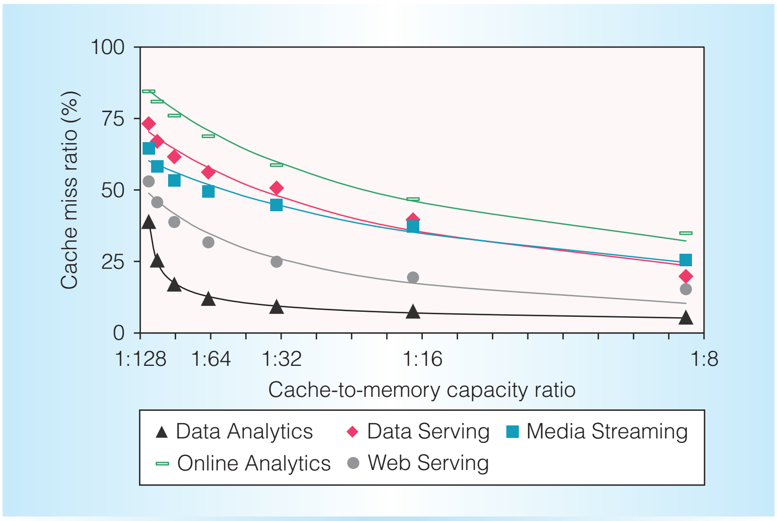

Figure 2. Miss ratio for various cache-to-memory capacity ratios. Lines denote *x*-shifted power-law fitting curves.

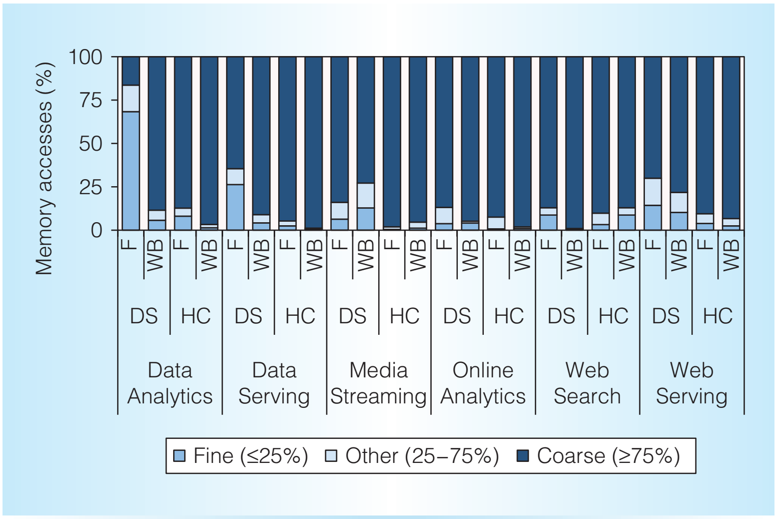

Figure 3. Granularity at which page-sized lines are fetched (F) from and written back (WB) to DRAM for the Die-Stacked (DS) cache and a high-capacity cache (HC) of 1:128 and 1:8 cache-to-memory capacity ratio, respectively.

accesses filtered. We thus conclude that the combination of poor cache performance and technological complexity of die stacking limits the usefulness of on-chip stacked DRAM caches in servers.

Spatial Characterization

Scale-out applications often operate on bulk objects (such as database rows), thus exhibiting a high incidence of coarse-grained accesses.[12] To allow for the retrieval of an object in sublinear time, objects are pinpointed through pointer-intensive indexing structures, such as hash tables and trees. For instance, data stores and object caching systems use a hash table to retrieve data objects. Although objects are accessed at coarse granularity, finding them requires performing a sequence of pointer dereferences. Thus, a non-negligible fraction of accesses is fine-grained.[12]

We examine the granularity at which high-capacity caches access memory by measuring the access density at which page-sized lines are fetched from and written back to memory in Figure 3. We define page access density as the fraction of 64-byte blocks within a page accessed between the page's first access and the page's eviction from the cache. We use a page size of 2 Kbytes because it reduces the tag array size significantly with limited tolerance for overfetch. Thus, fine-grained pages have low access density (up to 8 unique cache blocks accessed), whereas coarse-grained pages have high access density (at least 24 unique cache blocks accessed). For comparison, we include a low-capacity cache, labeled "Die-Stacked."

We find that Die-Stacked exhibits bimodal memory access behavior—that is, fine-grained and coarse-grained accesses account for 21 and 68 percent of accesses, respectively. Although coarse-grained accesses are prevalent, the frequent incidence of fine-grained accesses must also be accommodated effectively. Due to the limited capacity of on-chip stacked DRAM caches, pointer-containing pages show low temporal reuse and are frequently evicted. To avoid massive bandwidth waste in accesses to such pages, state-of-the-art stacked DRAM caches rely on block-based or footprint-predicting organizations that are bandwidth-frugal but carry a high metadata storage cost.

In contrast, high-capacity caches exhibit coarse-grained memory access behavior—that is, 93 percent of all accesses. This behavior is attributed to two phenomena. First, the lifetime of pages in the cache is on the order of tens to hundreds of milliseconds. Thus, pages containing a collection of fine-grained objects (such as hash bucket headers) can enjoy spatial locality uncovered through long cache residency times, stemming from skewed access distributions. Second, low-access-density pages containing pointer-intensive indexing structures with good temporal reuse (for example, intermediate tree nodes) are preserved across accesses.

Summary

Our study demonstrates that high-capacity caches are needed to capture the skewed memory access distributions of servers. We also find that the improved spatiotemporal behavior of high-capacity caches offers an opportunity to use a simple page-based organization, thus avoiding the storage and complexity overheads associated with state-of-the-art stacked DRAM caches.

Memory System Architecture for Scale-Out Servers

We present MeSSOS, a memory system architecture for scale-out servers that provides the required bandwidth and capacity for a scale-out server. High bandwidth is delivered through caching of data working sets in a high-capacity scale-out cache (soCache), which comprises multiple off-package stacked DRAM modules. High memory capacity is achieved through the deployment of multiple conventional (DDR-based) memory modules (DIMMs).

Figure 4 shows the design overview. In MeSSOS, an on-board building block comprises an soCache slice fronting a set of DIMMs. The design of each building block (for example, serial link and DDR bandwidth, cache-to-memory capacity ratio) is guided by our memory access characterization. Capacity and bandwidth can be seamlessly scaled by adjusting the number of building blocks. Next, we examine the soCache architecture and its integration with the rest of the system.

soCache Architecture

MeSSOS uses multiple off-package stacked DRAM modules as a high-capacity cache. To avoid communication between soCache slices, memory addresses are statically interleaved across the slices. Figure 5a shows the organization of an soCache slice. Stacked DRAM is internally organized as a set of vaults, which are connected to the serial link via an interconnect.

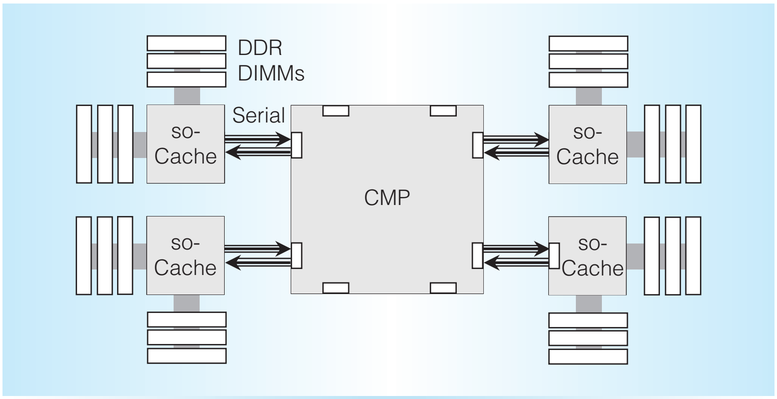

Figure 4. Overview of MeSSOS, a memory system architecture for scale-out servers. MeSSOS employs multiple off-package stacked DRAM modules as a scale-out cache (soCache) in front of DDR-based memory.

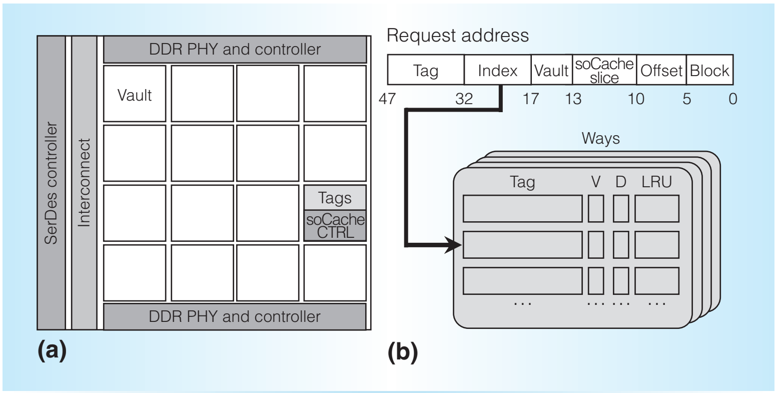

Figure 5. The organization of an soCache slice. (a) Logic die organization and (b) tag array architecture.

Cache organization. soCache uses a page-based organization, which leverages the observation that high-capacity caches uncover spatial locality that is beyond the temporal reach of lower-capacity caches. The page-based design not only naturally captures spatial locality, but also minimizes metadata storage requirements over block-based and footprint-predicting designs thanks to fewer cache sets and/or smaller tag entries. The page-based design also reduces dynamic DRAM energy by exploiting DRAM row buffer locality and fetching the entire page with one DRAM row activate, thus minimizing the number of DRAM row activates that dominate energy consumption in conventional DRAM.[12]

On the basis of page-size sensitivity analysis, we find that a page size of 2 Kbytes offers a good tradeoff between tag array size and bandwidth overhead stemming from over-fetch. We also observe that low associativity (four-way in the preferred design) is sufficient for minimizing the incidence of conflicts while also reducing tag and least recently used (LRU) metadata costs.

Tag array. The page-level organization reduces the tag array overhead significantly. For instance, an soCache of 32 Gbytes, consisting of eight 4-Gbyte slices, requires 5 Mbytes of tags per slice, or 20 mm^2 in 40-nm technology (obtained using CACTI). The small tag array size lets us embed it in the logic die of the modules comprising soCache. These logic dies are underutilized, typically housing per-vault memory controllers, an on-chip interconnect, and off-chip I/O interfaces and controllers. In our specialized HMC, these components occupy approximately 70 mm^2 in 40-nm technology (estimated by scaling die micrographs), leaving sufficient room for the tags on a typical HMC logic die (approximately 100 mm^2).

To enable low tag lookup latency, we distribute the tag array across the high-bandwidth memory module, placing each tag array slice beneath a vault. For a 4-Gbyte four-way associative soCache slice, each slice of the in-SRAM tag array requires only 320 Kbytes. A tag array slice corresponds to 32,768 sets, and each tag entry is 20 bits—that is, 15 bits for the tag, 2 page-level valid and dirty bits, and 3 bits for maintaining the pseudo-LRU tree. Low associativity and small in-SRAM tags allow for searching the ways in parallel at small latency (three to four processor cycles) and energy overheads, allowing for a feasible and practical set-associative cache organization.

Processor-soCache Interface

The processor is connected to the soCache via high-bandwidth serial links. Both the processor and soCache slices implement simple controllers to orchestrate communication (see Figure 4). The controllers comprise a pair of queues to buffer incoming and outgoing packets, and a SerDes interface. Processor-side controllers serialize outgoing requests into packets before routing them to the soCache slice based on corresponding address bits (Figure 5b), and they deserialize incoming data and forward them to the last-level cache. An soCache-side controller deserializes incoming memory requests and forwards them to the vault's soCache controller based on corresponding address bits (Figure 5b), and it serializes outgoing data into packets and forwards them to the processor.

Because scale-out workloads exhibit variable read-write ratios,[12] each serial link comprises 16 request lanes and 16 response lanes. Thus, a serial link requires approximately 70 pins (control and double-ended signaling for data lanes) as opposed to a DDR channel, which requires approximately 150 pins. The lower number of per-serial-link pins allows for integrating a high number of processor-side SerDes channels without increasing the number of the processor's pins compared to a processor with DDR channels, thereby keeping the cost associated with the processor's packaging constant.

soCache—Main Memory Interface

The off-package high-bandwidth memory modules provide the communication bridge between processor and main memory. Memory requests that miss in soCache are forwarded directly to local memory modules. To do so, the soCache slice integrates DDR controllers to control the local DDR channels, requiring the implementation of the DDR PHY and protocol in the logic die of the soCache modules.

DDR channels. Thanks to the high degree of bandwidth screening provided by soCache, the DDR channels operate at low frequency to reduce idle power. Compared to conventional HMCs hosting four SerDes interfaces, each of our specialized HMCs hosts only one SerDes interface (of approximately 9 mm^2 of area and 1.5 W of power), freeing up area and power resources for the required low-frequency DDR interfaces (approximately 10 mm^2 each). Our estimates show that the power consumption of an soCache slice lies within the power budget of conventional HMCs. The total number of pins required by each soCache slice matches that of on-board chips in buffer-on-board systems.

DDR controllers. These employ first-ready, first-come first-serve (FR-FCFS) open-row policy with page-level address interleaving. We map an entire soCache's page-sized cache line to one DRAM row by using the addressing scheme `Row:ColumnHigh:Rank:Bank:LocalChannel:soCacheSlice:ColumnLow:WordOffset`, in which `ColumnHigh` is 2 bits and `ColumnLow` is 8 bits. To guarantee that requests missing in an soCache slice are served by local DRAM, the mapping scheme interleaves addresses across local channels using the least-significant vault bit.

System-Level Considerations

We qualitatively examine the system-level implications of the cache substrate and discuss the scalability and cost of the proposed memory system.

Cache substrate. Although we choose off-package stacked DRAM as our cache substrate, our insights on high-capacity cache design also apply to on-package stacked DRAM. Such a design can lower cache access latency by avoiding chip-to-chip links, but at the cost of lower cache hit rates in big-memory systems and additional buffer-on-board chips, which would be required to afford high memory capacity with DIMMs given the pin-count limitations of a single package.

Scalability. MeSSOS delivers high memory capacity in a scalable manner while relying on cost-effective DIMMs. MeSSOS distributes the required number of DDR channels and their pins across multiple soCache modules as opposed to a single processor chip. This approach resembles that of buffer-on-board systems, which employ on-board chips to cost effectively boost memory capacity. In contrast to these systems, MeSSOS does not require additional on-board buffer chips, because the functionality of those chips is implemented in the logic die of the soCache modules.

TCO. MeSSOS achieves substantial system cost savings due to lower acquisition and operating costs. By providing the required bandwidth and capacity for a server, MeSSOS maximizes server throughput, thus reducing the number of servers required for the same throughput. MeSSOS also lowers memory energy by minimizing the static power footprint of its underlying memory interfaces. As MeSSOS employs off-package stacked DRAM as a cache, it bridges the processor-bandwidth gap with the bare minimum number of power-hungry serial links, efficiently uses serial link bandwidth and amortizes their high idle power consumption, and filters a high degree of memory accesses, and thus infrequent main memory accesses can be served by underclocked DIMMs.

Experimental Methodology

We evaluate MeSSOS performance and energy efficiency using a combination of cycle-accurate full-system simulations, analytic models, and technology studies.

Scale-Out Server Organization

We model chips with an area of 250 to 280 mm^2 and a power budget of 95 to 115 W. We use the scale-out processor methodology to derive the optimal ratio between core count and cache size in each technology.[3] The configuration resembles that of specialized many-core CMPs, such as Cavium ThunderX.

Table 2 summarizes the details of the evaluated designs across technology nodes. For a given technology node, the processor configuration and memory capacity are fixed. We evaluate the following memory systems:

- DDR-only memory.
- Buffer-on-Board (BOB),[9] which relies on on-board chips to boost bandwidth and capacity through additional DDR channels, but at the cost of higher end-to-end memory latency and energy consumption due to (processor-BOB) serial links and intermediate buffers.
- High-Bandwidth Memory Modules (HBMM), which replaces DDR-based memory with off-package stacked DRAM—that is, stacked DRAM is deployed as main memory. HBMM employs a tree network topology to reduce the number of network hops; the average and maximum number of

Table 2. System configuration.

System	2015 (22 nm)	2018 (14 nm)	2021 (10 nm)
Chip multiprocessor (CMP)	96 cores, 3-way out-of-order (OoO), 2.5 GHz	180 cores, 3-way OoO, 2.5 GHz	320 cores, 3-way OoO, 2.5 GHz
Last-level cache	24 Mbytes	45 Mbytes	80 Mbytes
Memory	384 Gbytes	720 Gbytes	1,280 Gbytes
Double data rate (DDR)*	4 DDR-1600	5 DDR-2133	6 DDR-2667
High-Bandwidth Memory Modules (HBMM)†	8 32-lane at 10 Gbps	10 32-lane at 15 Gbps	12 32-lane at 20 Gbps
Buffer-on-Board (BOB)‡	8 32-lane at 10 Gbps 16 DDR-1600	10 32-lane at 15 Gbps 20 DDR-2133	12 32-lane at 20 Gbps 24 DDR-2667
Die-Stacked§	Cache: 1 Gbyte Off-chip: 4 DDR-1600	Cache: 2 Gbytes Off-chip: 5 DDR-2133	Cache: 4 Gbytes Off-chip: 6 DDR-2667
MeSSOS‖	CMP-Cache: 8 32-lane at 10 Gbps Cache: 8 × 4 Gbytes Cache-Memory: 16 DDR-1066	CMP-Cache: 10 32-lane at 15 Gbps Cache: 10 × 8 Gbytes Cache-Memory: 20 DDR-1066	CMP-Cache: 12 32-lane at 20 Gbps Cache: 12 × 16 Gbytes Cache-Memory: 24 DDR-1066

*DDR memory latency: 55 ns including off-chip link (15 ns) and DRAM core (40 ns).
†HBMM memory latency: hop-count × 35 ns (SerDes and pass-through logic) + 20 ns (stacked DRAM access).
‡BOB memory latency: 95 ns including SerDes and buffer (40 ns), buffer-DDR link (15 ns), and DRAM core (40 ns).
§Die-Stacked hit latency: approximately 20 ns, including predictor lookup and stacked DRAM access (20 ns). Die-Stacked miss latency: approximately 55 ns, including predictor lookup and off-chip DRAM access (55 ns).
‖MeSSOS tag lookup latency: 35 ns, including SerDes (30 ns) and distributed tag array lookup (5 ns). MeSSOS hit latency: 55 ns, including tag lookup (35 ns) and stacked DRAM access (20 ns). MeSSOS miss latency: 95 ns, including tag lookup (35 ns) and off-chip DRAM access (60 ns).

network hops are three and four, respectively.

- A Die-Stacked cache with a block-based organization[8] that maximizes effective capacity and minimizes off-chip bandwidth waste. The cache is backed by DDR-based memory.
- MeSSOS, which deploys off-package stacked DRAM modules as a cache in front of DDR-based memory.

HBMM, BOB, and MeSSOS provide the required memory bandwidth for a scale-out server, summarized in Table 1. DDR is bandwidth-constrained for all workloads, and Die-Stacked's provided effective bandwidth is limited due to poor cache hit ratios. Following the memory capacity requirements discussed in Table 1, all systems provide 4 Gbytes per core. For the DDR baseline and Die-Stacked, we consider DIMMs with a higher number of memory ranks than those deployed in the rest of the systems.

Performance and Energy Models

Due to space constraints, we present only a summary of our framework. The details of the framework, including system performance, energy modeling, and projection to future technologies can be found elsewhere.[13,14]

Performance. We measure performance using analytic models, which are validated against cycle-accurate full-system simulations of a 16-core CMP with high accuracy (5 percent average error). Our model extends the classical average memory access time analysis to predict per-core performance for a given memory system. The model is parameterized

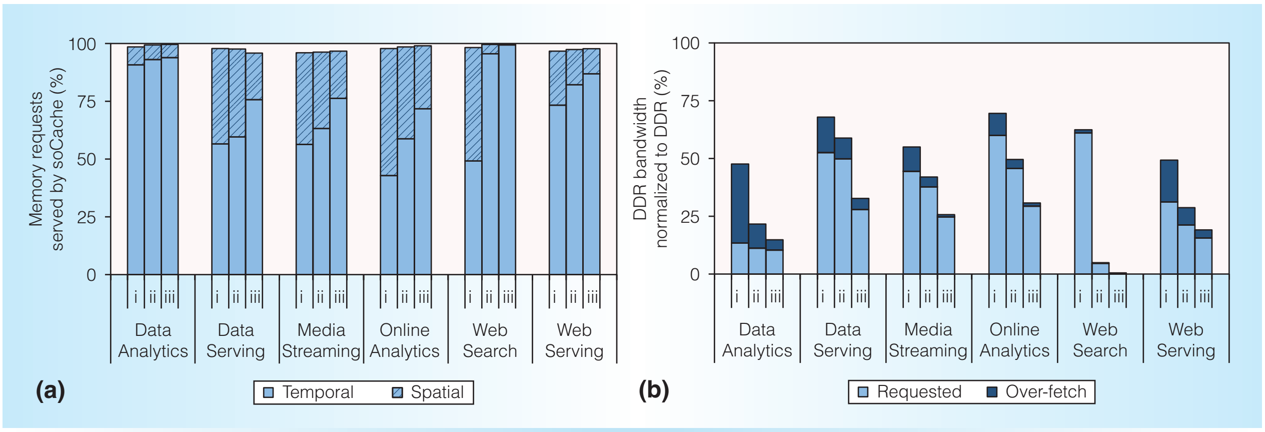

Figure 6. MeSSOS effectiveness for cache-to-memory capacity ratios of (i) 1:32, (ii) 1:16, and (iii) 1:8. (a) Memory requests served by soCache and (b) DDR bandwidth consumption compared to the DDR baseline.

by 16-core full-system simulations results (using Flexus[15]), including core performance, miss rates of on-chip and stacked DRAM caches, and interconnect delay. For off-chip memory access latency, we include link latency, memory core latency, and queuing delays. We model queuing delays by running cycle-accurate simulations to measure memory latency for various bandwidth utilization levels for each workload separately.

Energy. We develop a custom energy modeling framework to include various system components, such as cores, on-chip interconnects, caches, memory controllers, and memory. Our framework draws on several specialized tools (such as CACTI and McPAT) to maximize fidelity through detailed parameter control.

Future technologies. To understand the effect of technology scaling on the examined memory systems, we model our systems in 2018 and 2021. Per *International Technology Roadmap for Semiconductors* (ITRS) estimates, processor supply voltages will scale from 0.85 V (2015) to 0.8 V (2018) and 0.75 V (2021). We use Micron's datasheets to examine the impact of data rate and memory density on DDR energy. We also study the impact of manufacturing technology on power consumption and data rate of SerDes interfaces based on numerous published measurements.

Workloads. Our analysis is based on a wide range of scale-out workloads taken from CloudSuite 2.0.[2] We also evaluate Online Analytics running a mix of TPC-H queries on a modern column-store database engine called MonetDB.

Evaluation

We compare MeSSOS to various memory systems in terms of system performance and energy efficiency across technology generations.

Performance and Energy-Efficiency Implications

We begin our study with a 96-core CMP in the 22-nm technology. Figure 6a plots the fraction of memory requests that are served by soCache for various cache-to-memory capacity ratios. The figure demonstrates the ability of MeSSOS to serve the bulk (more than 95 percent) of those using its soCache thanks to temporal locality arising from skewed access distributions (solid bar) and spatial locality arising from page-based organizations and high cache residency times stemming from high cache capacity (striped bar).

Figure 6b illustrates the DDR bandwidth consumption compared to the DDR baseline. As expected, DDR bandwidth savings increase with bigger caches. For a 1:8 cache-to-memory capacity ratio, soCache captures the hot data working sets, and can therefore absorb 65 to 95 percent of memory traffic. The dark bars illustrate the extra traffic generated due to coarse-grained transfers between soCache and the DIMMs. The absolute increase in traffic is small (3 percent on

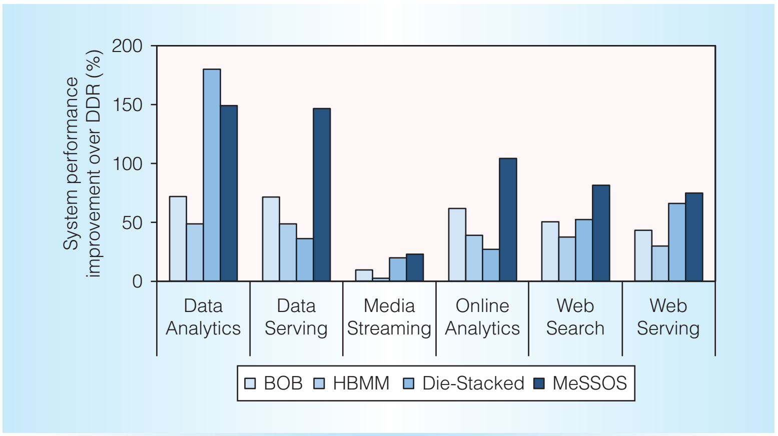

Figure 7. System performance improvement of various memory systems over DDR. System performance is measured by User IPC, defined as the ratio of application instructions committed to the total number of cycles, including cycles spent executing operating system code.

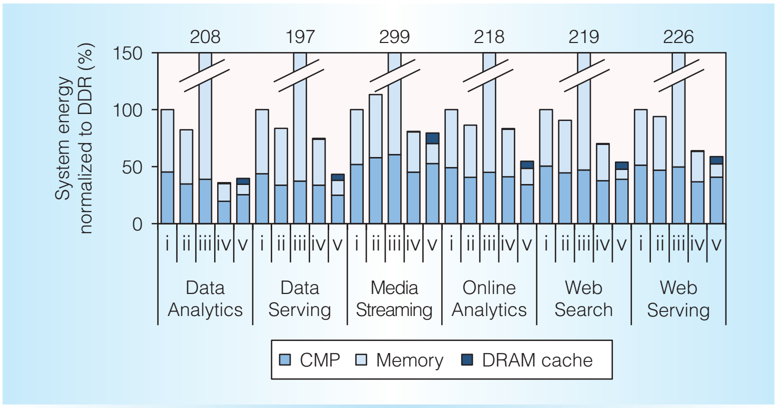

Figure 8. System energy breakdown for (i) DDR, (ii) BOB, (iii) HBMM, (iv) Die-Stacked, and (v) MeSSOS. Energy is broken down into processor (CMP), memory, and DRAM cache components.

average). For the rest of the evaluation, we use the 1:8 cache-to-memory capacity ratio, unless stated otherwise.

Performance. Figure 7 compares MeSSOS to the DDR baseline as well as HBMM (which employs high-bandwidth memory modules as main memory), BOB, and Die-Stacked. BOB and HBMM improve performance over DDR by 49 percent and 33 percent, respectively, as they provide sufficient bandwidth to the processor. However, the bandwidth increase comes at the cost of higher memory latency. BOB adds an extra 40 ns whereas HBMM requires a point-to-point network, which adds a latency of 35 ns per network hop. Because HBMM accesses are frequently multihop, BOB outperforms HBMM by 12 percent. Our analysis (not shown) also finds that on-board SRAM caches found in some BOB chips exhibit low temporal locality (average hit ratio of 25 percent), and thus provide negligible performance gains.

MeSSOS outperforms all systems due to its ability to provide high bandwidth at low latency. Compared to the DDR baseline, MeSSOS improves system performance by approximately 2 times. MeSSOS outperforms BOB and HBMM by 28 and 43 percent, respectively, due to lower memory latency.

MeSSOS outperforms Die-Stacked by 23 percent due to lower off-chip bandwidth pressure, resulting from its greater cache capacity. On average, MeSSOS filters 84 percent of DDR accesses as compared to 45 percent in Die-Stacked. For Data Serving and Online Analytics, MeSSOS outperforms Die-Stacked by 81 and 61 percent, as Die-Stacked is bandwidth-constrained due to its inability to filter off-chip bandwidth (only 38 and 13 percent of accesses). One exception is Data Analytics, where memory accesses are extremely skewed; hence, Die-Stacked achieves a high hit ratio, outperforming MeSSOS due to lower cache access latency.

Energy. Figure 8 plots system energy for the examined designs normalized to the DDR baseline. BOB reduces energy by 12 percent compared to DDR mainly due to performance gains. HBMM increases energy by 2.3 times compared to DDR due to its power-hungry memory network.

MeSSOS reduces system energy by 1.9, 1.7, and 4.3 times compared to DDR, BOB, and HBMM, respectively. Because soCache serves the bulk of the accesses, MeSSOS exploits the low-energy access of stacked DRAM modules, thus reducing memory energy consumption significantly. Furthermore, MeSSOS enforces coarse-grained data movement between soCache and DRAM, thus amortizing energy-intensive DRAM row activates.[12] Compared to Die-Stacked, MeSSOS reduces energy by 23 percent due to lower DDR energy resulting from lower off-chip bandwidth consumption.

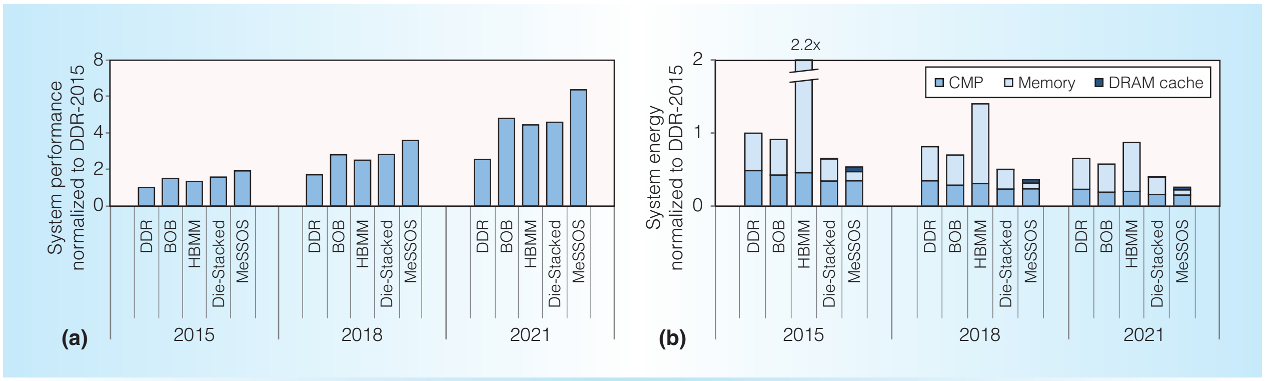

Figure 9. Technology's effect on MeSSOS. (a) System performance and (b) system energy consumption for various technology generations normalized to DDR-2015.

Projection to Future Technologies

We conclude our evaluation by studying the effect of technology scaling on MeSSOS's performance and energy efficiency in 14 nm (2018) and 10 nm (2021) technologies (see Figure 9).

MeSSOS leverages the abundant bandwidth provided by SerDes, increasing performance almost linearly with the number of cores and by 3.7 times (2018) and 6.6 times (2021) compared to DDR-2015. Due to poor scalability of DDR interfaces, the bandwidth gap between DDR-based systems and the processor is increasing rapidly. Thus, MeSSOS's performance improvement over DDR and Die-Stacked increases across technologies. MeSSOS improves performance by 2.3 times (2018) and 2.7 times (2021) over DDR, and by 30 percent (2018) and 43 percent (2021) over Die-Stacked.

Regarding energy efficiency, the DDR energy footprint increases across technologies. Because MeSSOS employs underclocked DIMMs, its energy footprint increases by only a small factor. Thus, MeSSOS reduces energy by 1.7 times (2015), 2 times (2018), and 2.6 times (2021) as compared to DDR and BOB, and by 23 percent (2015), 40 percent (2018), and 60 percent (2021) as compared to Die-Stacked. Compared to HBMM, MeSSOS reduces energy by 4 to 4.4 times.

There is a great need for a high-capacity high-bandwidth memory system for scale-out servers. Memory system architects are increasingly relying on heterogeneity to provide the required bandwidth and capacity for a server. In heterogeneous memory systems, high-bandwidth memory is deployed as a cache to lower-bandwidth, yet higher-density, memory.

In this article, we focus on near-term future memory systems (till 2021) and seek to identify the right granularity at which high-capacity caches should be organized. Our analysis shows that high-capacity caches for scale-out servers exhibit high spatiotemporal locality across data objects owing to long cache residency stemming from skewed dataset access distributions. We exploit this phenomenon to reduce the design complexity of state-of-the-art hardware-managed high-capacity caches and propose a practical memory system architecture that uses emerging high-bandwidth memory as a high-capacity cache in front of conventional memory.

We leave the architecture of longer-term future memory systems (beyond 2021) for future work. Nevertheless, we summarize important system-level design considerations, including cache management and the choice of substrate for system memory.

First, in terms of cache management, advancements in cooling of high-bandwidth memory modules might increase stacked memory capacities and consequently tag storage requirements of hardware-managed caches. Although the logic die in today's high-bandwidth memory modules is underutilized, the emergence of near-memory accelerators might reduce available silicon for a fixed memory package size. The design decision of organizing the cache at page granularity raises the question of whether the operating system should be managing the cache so as to free logic die resources for near-memory accelerators.

Finally, with regards to the choice of the system memory substrate, storage-class memory emerges as a cost-effective solution to the capacity scalability limitation that plagues conventional memory. The high cache hit ratio—achieved by page-based caches—can mask the high latency of emerging storage-class memory, raising the question of whether the latter could replace conventional memory in the memory hierarchy. MICRO

Acknowledgments

We thank Cansu Kaynak, Onur Kocberber, and the anonymous reviewers for their insightful feedback and comments on earlier versions of this article. This work was partially supported by the NanoTera project "YINS."

References

1. L.A. Barroso and U. Holzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machine*, Morgan & Claypool, 2009.
2. M. Ferdman et al., "Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware," *Proc. 17th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 12), 2012, pp. 37–48.
3. P. Lotfi-Kamran et al., "Scale-Out Processors," *Proc. 39th Ann. Int'l Symp. Computer Architecture* (ISCA 12), 2012, pp. 500–511.
4. D. Jevdjic et al., "Die-Stacked DRAM Caches for Servers: Hit Ratio, Latency, or Bandwidth? Have It All with Footprint Cache," *Proc. 40th Ann. Int'l Symp. Computer Architecture* (ISCA 13), 2013, pp. 404–415.
5. B. Black et al., "Die Stacking (3D) Microarchitecture," *Proc. 39th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO 39), 2006, pp. 469–479.
6. D. Jevdjic et al., "Unison Cache: A Scalable and Effective Die-Stacked DRAM Cache," *Proc. 47th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO 47), 2014, pp. 25–37.
7. G.H. Loh and M.D. Hill, "Efficiently Enabling Conventional Block Sizes for Large Die-Stacked DRAM Caches," *Proc. 44th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO 44), 2011, pp. 454–464.
8. M. Qureshi and L.H. Gabriel, "Fundamental Latency Trade-off in Architecting DRAM Caches: Outperforming Impractical SRAM Tags with Simple and Practical Design," *Proc. 45th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO 45), 2012, pp. 235–246.
9. *Cisco Unified Computing System Extended Memory Technology*, Cisco, report C45-555038-03, Feb. 2010.
10. B. Grot et al., "Optimizing Datacenter (TCO) with Scale-Out Processors," *IEEE Micro*, vol. 32, no. 5, 2012, pp. 52–63.
11. X. Jiang et al., "CHOP: Adaptive Filter-based DRAM Caching for CMP Server Platforms," *Proc. IEEE 16th Int'l Symp. High Performance Computer Architecture*, 2010; doi:10.1109/HPCA.2010.5416642.
12. S. Volos et al., "BuMP: Bulk Memory Access Prediction and Streaming," *Proc. 47th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO 47), 2014; doi:10.1109/MICRO.2014.44.
13. S. Volos et al., *An Effective DRAM Cache Architecture for Scale-Out Servers*, tech. report MSR-TR-2016-20, Microsoft Research, 2016.
14. S. Volos, "Memory Systems and Interconnects for Scale-Out Servers," PhD dissertation, EPFL-THESIS-6682, Dept. Computer & Communication Sciences, EPFL, 2015.
15. T.F. Wenisch et al., "SimFlex: Statistical Sampling of Computer System Simulation," *IEEE Micro*, vol. 26, no. 4, 2006, pp. 18–31.

Stavros Volos is a researcher at Microsoft Research. His research interests include processors, memory systems, and system architectures for efficient and secure cloud computing. Volos has a PhD in computer and communication sciences from EPFL. Contact him at svolos@microsoft.com.

Djordje Jevdjic is a postdoctoral research associate at the University of Washington. His research interests include DRAM caches,

near-data processing, and approximate computing. Jevdjic has a PhD in computer and communication sciences from EPFL. Contact him at jevdjic@cs.washington.edu.

Babak Falsafi is a professor in the School of Computer and Communication Sciences at EPFL and the founding director of EcoCloud, an interdisciplinary research center targeting robust, economic, and environmentally friendly cloud technologies. Falsafi has a PhD in computer science from the University of Wisconsin–Madison. Contact him at babak.falsafi@epfl.ch.

Boris Grot is an assistant professor in the School of Informatics at the University of Edinburgh. His research interests include processors, memory systems, and software stacks for data-intensive computing. Grot has a PhD in computer science from the University of Texas at Austin. Contact him at boris.grot@ed.ac.uk.

myCS Read your subscriptions through the myCS publications portal at http://mycs.computer.org.

Are Enemy Hackers Slipping through Your Team's Defenses?

Protect Your Organization from Hackers by Thinking Like Them

Take Our E-Learning Courses in the Art of Hacking

You and your staff can take these courses where you are and at your own pace, getting hands-on, real-world training that you can put to work immediately.

www.computer.org/artofhacking

IEEE NOT SO SECURE IEEE computer society