Memory Systems and Interconnects for Scale-Out Servers

THÈSE N^{O} 6682 (2015)

PRÉSENTÉE LE 18 SEPTEMBRE 2015

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE D'ARCHITECTURE DE SYSTÈMES PARALLÈLES
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Stavros VOLOS

acceptée sur proposition du jury:

Prof. W. Zwaenepoel, président du jury
Prof. B. Falsafi, directeur de thèse
Prof. R. Balasubramonian, rapporteur
Prof. Y. Xie, rapporteur
Prof. P. Ienne, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

The path of virtue goes through toils

— Euripides

To my parents for their endless love and support

Acknowledgments

My PhD journey would have not been memorable, joyful, and successful if not for the support and friendship of many people. The credit which needs to be given to these people is massive; so, this section will be a bit long.

First and foremost, I would like to thank my academic advisor, Babak Falsafi, for providing me with the opportunity for pursuing a PhD under his supervision. His financial support throughout my studies allowed me to focus on developing my teaching, research, presentation, and writing skills. Pursuing a PhD under Babak's supervision goes beyond academic horizons and feels more like a one-time life experience. Babak's persistence and will to aim always for perfection were two of his characteristics that brought the best out in me. His quotes *"Just make it perfect. How hard is it?"* and *"It will make you a better person"* had constantly been inspiring me and guiding me through the long PhD journey. Babak has been a great teacher in and out of the office. His teaching methods might be a bit harsh, but they have been really successful. Babak taught me how to see the big picture, to methodically identify and scientifically solve a research problem, and finally how to talk and write about it. Babak's input was instrumental in understanding how humanity really works, and developing a mature way of thinking. Special thanks for the off-slope skiing sessions during our group's ski retreats. Thanks for helping me to become a researcher and a better person.

I would like to thank Rajeev Balasubramonian, Paolo Ienne, and Yuan Xie for serving as my thesis committee, and for their time and insightful comments and feedback. I wish to thank Willy Zwaenepoel for serving as the jury president of my thesis exam.

I had the fortune to work in a great and fun research group, Parallel Systems Architecture Lab (PARSA), surrounded by amazing people whose assistance, feedback, and support helped me become a researcher while their friendship improved me as a person and guaranteed a memorable PhD journey:

- Cansu Kaynak, Djordje Jevdjic, and Onur Kocberber have been great academic siblings and friends. We enrolled to the PhD program in the same year, and as a result our PhD journeys were fully overlapped. Our relationship can be characterized in one sentence *"All for one and one for all"*. We shared a common office for almost four years, spent unlimited amounts of time at the lab working on team projects, brainstorming, and giving each other insightful and valuable feedback. Besides being great group-mates, collaborators, and teachers, they have been great friends whose friendship kept my sanity intact, moral support helped me overcoming tough moments, and fruitful discussions enriched my knowledge. We spent a lot of time procrastinating, having fun at restaurants and bars, and traveling for conferences and ski retreats. Special tribute to Onur who had to tolerate me as an office-mate for an additional two years, as a beer-mate, and as a ski-mate in the beautiful Alps. Additional thanks to Djordje for boosting my consciousness about green computing. Dudes (and dudette), big thanks for everything. Your input played a pivotal role in ensuring a memorable and joyful PhD journey. This journey would have not been successful if not for you.

- Javier Picorel and Sotiria Fytraki have been great friends and group-mates for over five years. Together, we endured lots of tough moments and had many fun experiences. I will never forget the long psychotherapy sessions with Sotiria in the aftermath of paper rejections. Besides being a great friend, Javier also served as a trustworthy collaborator in various research projects. Javier's life- and academic-wise input was of great importance to me. We had lots of valuable discussions about problems regarding our life and research during dinners, drinking nights, and long days in the lab. I am pretty sure we helped and improved each other considerably in so many ways.

- Special thanks to Boris Grot and Mike Ferdman for everything I learned from them. Boris has been a great mentor. Through our collaboration he taught me how to think, how to write, and many other skills that were instrumental in accomplishing my academic goals. It was always a pleasure talking with him as his positive attitude always inspired me to keep trying. Mike has been a great role model and has served as a great senior student in the first three years of my PhD studies. Mike had the responsibility of enriching our technical and presentation/writing skills. Additional thanks to them for all the nice moments we spent in Lausanne, conferences, and ski retreats.

- Alexandros Daglis has been a great group-mate and friend for over three years. His feedback during paper deadlines and practice talks was valuable and insightful. He has also been a great team-mate during our football games against the Lebanese opponents. Alex has been the only person in Switzerland that understood my passion for trance music, and one of the few who shared my preference for Indian Pale Ales in our drinking and burger nights. Thanks for all the fun moments we shared in Lausanne, conferences, ski retreats, and other places in the world.

- Thanks to Alisa Yurovsky and Effi Georgala for being good friends. Alisa has made many weekend days of the PARSA members more colorful due to her amazing organization skills, including barbecues by the lake and Halloween parties. Thanks to Effi for the fun breaks in the lab, her advice about job interviews and thesis exams, and for having the patience to help me dropping a part of my Greek accent in English.

- Thanks to Adileh Almutaz and Pejman Lotfi Kamran for the fruitful collaborations in the early stages of my PhD studies. Thanks to other members of the PARSA group — Stanko Novakovic, Georgios Psaropoulos, Nooshin Mirzadeh, Mario Drumond, and Dmitrii Ustiugov — for contributing in various ways to the completion of this thesis.

- I would like to thank Rodolphe Buret for providing technical support, Stéphanie Baillargues and Valérie Locca for providing administrative advice, and Ousmane Diallo for his advice on job finding.

- Kudos to our ISOMAC coffee machine which served our lab with the finest espresso in Switzerland for over four years.

I would like to thank the Greek community for making the life in Lausanne more fun and for reminding me my Greek origins:

- Manos Athanassoulis and Yannis Alagiannis have been great friends since my first day in the PhD program. Unfortunately, the work load of the PhD program was prohibitive in some cases, causing my temporal disappearance. Among other things, Manos had been a great host for Euroleague nights; Yannis had been a great team-mate in various courses, including Advanced Algorithms, the core exam required for becoming a PhD candidate. Lots of pain and uncertainty there. Our trip to Barcelona was a memorable one after which they started calling me *Storus the Panoramicus*.
- Thanks to Evangelos Vlachos, Aris Ioannou, and Manos Karpathiotakis for being good friends even if our overlap at EPFL was a short one. Lots of fun and delightful discussions during lunch, dinners, drinking nights, sports, and many other events.
- The list of Greek friends is so long that I would have certainly forgotten someone if I had tried to name individuals. Thank you all for making my life in Lausanne more colorful. Thanks to AEGEL for organizing Christmas and carnival parties as well as unforgettable barbecue events by the lake.

Special thanks to my lifelong friends Tryphon Athanassiades, Christos Dionysiou (and Maria Faladou), Giorgos Vassiliou, and Pambos Yioufkas from Cyprus for making my life joyful and colorful, and for having amazing experiences and fun with them during summer and winter holidays in various places across the world, bars, rocky and sandy beaches, and their apartments in my hometown. Through their friendship, I was able to spend wonderful time during precious time-off necessary for "rebooting" while long-distance interactions with them helped me maintaining my sanity.

Many thanks to my friends, Dimitris Dimitriou and Yiannis Tofis, for long and entertaining chats over Skype and for all the cool drinking nights in bars and other fun moments in sandy beaches.

Kudos to Armin Van Buuren's and Prisma Storm's weekly shows for delivering euphoric sessions of trance music which has been inspiring me for almost a decade now.

I would like to express my appreciation to Georgios Psaropoulos for preparing the German version of my thesis abstract. The thesis was supported by the Swiss National Science Foundation project 200021_127021 and the NanoTera project "YINS".

Last but not least, I would like to express my deep indebtedness to my family, including my parents, Michael and Maro, and my two elder brothers, Haris and Yiannis. Words cannnot describe my gratitude to these people. Throughout my life, they have been great supporters and teachers in so many ways. Skype calls with them have been reminding me who I am and what my origins are. The constant support and the endless love of my parents have been guiding me throughout my life and helped me overcoming the dark stages of my PhD studies. Haris has been a great role model during my graduate studies. His guidance and support helped me during tough moments of my PhD. Due to mutual research interests, I was lucky to spend lots of time with him during various conferences and in various places across the world. Yiannis' desire for high-risk projects has motivated me to be less conservative in my research and to keep fighting after paper rejections. Papa, Mama, Hari, Yianni thanks for everything!!!

Lausanne, 25 Juillet 2015 S. V.

Abstract

The information revolution of the last decade has been fueled by the digitization of almost all human activities through a wide range of Internet services. The backbone of this information age are scale-out datacenters that need to collect, store, and process massive amounts of data. These datacenters distribute vast datasets across a large number of servers, typically into memory-resident shards so as to maintain strict quality-of-service guarantees.

While data is driving the skyrocketing demands for scale-out servers, processor and memory manufacturers have reached fundamental efficiency limits, no longer able to increase server energy efficiency at a sufficient pace. As a result, energy has emerged as the main obstacle to the scalability of information technology (IT) with huge economic implications.

Delivering sustainable IT calls for a paradigm shift in computer system design. As memory has taken a central role in IT infrastructure, memory-centric architectures are required to fully utilize the IT's costly memory investment. In response, processor architects are resorting to manycore architectures to leverage the abundant request-level parallelism found in data-centric applications. Manycore processors fully utilize available memory resources, thereby increasing IT efficiency by almost an order of magnitude.

Because manycore server chips execute a large number of concurrent requests, they exhibit high incidence of accesses to the last-level-cache for fetching instructions (due to large instruction footprints), and off-chip memory (due to lack of temporal reuse in on-chip caches) for accessing dataset objects. As a result, on-chip interconnects and the memory system are emerging as major performance and energy-efficiency bottlenecks in servers.

Abstract

This thesis seeks to architect on-chip interconnects and memory systems that are tuned for the requirements of memory-centric scale-out servers. By studying a wide range of data-centric applications, we uncover application phenomena common in data-centric applications, and examine their implications on on-chip network and off-chip memory traffic. Finally, we propose specialized on-chip interconnects and memory systems that leverage common traffic characteristics, thereby improving server throughput and energy efficiency.

Key words: cloud, scale-out, datacenters, interconnects, memory systems, DRAM

Zusammenfassung

Die Informationsrevolution des letzten Jahrzehnts wurde durch die Digitalisierung aller menschlichen Aktivitäten mittels einer breiten Palette von Internetdienstleistungen befördert. Die Basis dieser Informationszeitalter feststellen die Scale-Out-Rechnenzentren, die große Datenmengen sammeln, speichern und verarbeiten. Diese Rechenzentren verteilen große Datasets über eine große Anzahl von Servern, in speicherresidenter Shards normalerweise, um bestimmte Quality-of-Service-Garantien zu erhalten.

Während Daten die explodierenden Nachfrage für Scale-Out-Server fährt, haben Prozessor- und Speicher-Hersteller grundlegende Effizienzgrenzen, denn die Serverenergieeffizienz kann nicht mehr ausreichend steigern. Energie entsteht als das größte Hindernis für die Skalierbarkeit der Informationstechnologie (IT) mit starken wirtschaftlichen und ökologischen Auswirkungen.

Die Lieferung nachhaltiger IT fordert einen Paradigmenwechsel im Computersystementwurf. Wegen der zentralen Rolle des Speichers in IT-Infrastruktur, Speicherorientierte Architekturen sind erforderlich, um die kostspielig Speicher Investition in vollem auszunutzen. Im Gegenzug sind Prozessorarchitekten auf Manycore-Architekturen zurückgegriffen, um die reichliche Anforderungsebenenparallelität datenorientierter Applikationen zu verwenden. Manycore-Prozessoren voll ausnutzen verfügbaren Speicherressourcen, wodurch IT-Effizienz um fast eine Größenordnung verbessert ist.

Denn Manycore-Server-Chips führen eine große Anzahl von gleichzeitigen Anforderungen aus, zeigen sie viele Zugriffe auf die Last-Level-Cache zum Instruktionsabrufen (aufgrund der großen Anweisungsbedarf), und Off-Chip-Speicher (aufgrund fehlenden zeitlichen Wiederverwendung in On-Chip-Caches) für den Zugriff auf Dataset-Objekte. Demzufolge entstehen

die Chip-Interconnects und das Speichersystem als große Leistung- und Energieeffizienz-Engpässe der Server.
Diese These zielt darauf ab, On-Chip-Netzwerke und Speichersystemen zu entwickeln, die für die Anforderungen der Speicherzentrierten Scale-Out-Servern optimiert werden. Durch das Studium einer Vielzahl von datenzentrischen Applikationen, wir entdecken gewöhnliche Phänomene dieser Applikationen und untersuchen ihre Auswirkungen auf das Netzwerkverkehr und das Speicherverkehr. Schließlich schlagen wir spezialisierten On-Chip-Netzwerk und Speichersysteme, die das gemeinsame Traffic Charakteristiken verwenden, um das Throughput und die Energieeffizienz zu verbessern.

Stichwörter: Cloud, Scale-Out, Rechnenzentren, Netzwerke, Speichersystemen, DRAM

Contents

List of Figures

List of Tables

1 Introduction

Over the past five decades, information technology (IT) has gone through multiple phases. Driven by the continuous digitization of human activities, IT has transitioned from being compute-centric, to being network-centric, to being data-centric. IT was born in the form of *mainframes* to fulfill the need for number manipulation. The daily interaction between individuals and computers gave rise to *personal computers*, which were instantly made available in enterprises and homes, and digitized new flavors of activities, such as text writing. The need for fast interaction and communication among individuals in enterprises caused a shift in IT toward *networked computers*. A massive paradigm shift in IT followed when computer networks led to the invention of *Internet* due to the desire of individuals to connect and interact with each other across the world. Today, almost all our daily activities have been digitized through a wide range of Internet services, such as online banking, social networking, and video streaming. The information revolution of the last decade has been fueled by the digitization of all kinds of data, granting to individuals ubiquitous access to data and capturing information of value to business and societies.

Technology innovations and advancements in semiconductor fabrication industry have been powering IT with scalable computing platforms for decades. Two powerful paradigms have enabled computing scalability: *Moore's Law* and *Dennard Scaling*. Moore's Law postulates that fabrication advancements enable reduction in transistor size, doubling transistor density

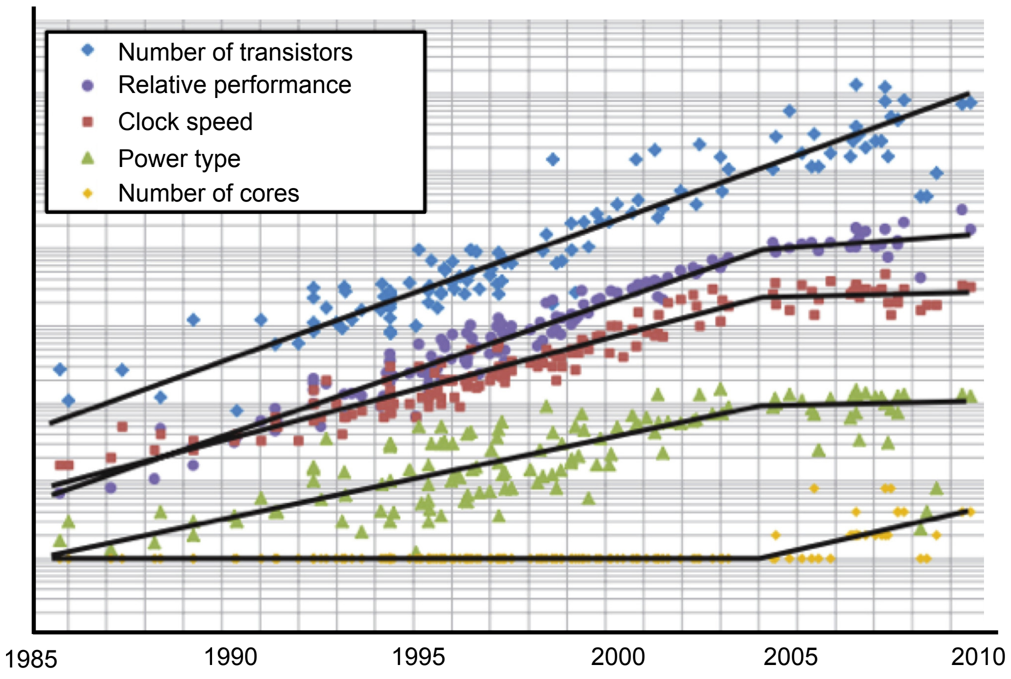


Figure 1.1 – Scaling trends for the transistor count, clock frequency, number of cores, and single-thread performance of processor chips. Source: Graph created by C. Batten based on data from M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten.

for processor and memory chips approximately every two years, while Dennard Scaling states that as transistors get smaller, their power density stays constant due to reduction of chip voltages. As shown in Figure 1.1, smaller and faster transistors have allowed processor designers to scale performance exponentially through higher core clock frequencies and micro-architectural advancements until 2004, and higher core counts for the last decade (2004-today) once frequency scaling became prohibitive due to chip-level cooling, thermal, and power constraints. Moore's Law coupled with Dennard Scaling have enabled an exponential increase in computing energy efficiency, and have been fueling IT with scalable computing platforms for over four decades.

1.1 Data-Centric Information Technology Meets Energy Wall

Data has taken a central stage in our world, driving the skyrocketing demands of the IT sector for computational resources. The backbone of today's IT is large-scale datacenters, which host a myriad of IT services and consequently collect, store, and process massive amounts of data. State-of-the-art datacenters deployed by technology giants, such as Google and Microsoft, host tens of thousands of servers, and have huge acquisition costs ($100+ M), occupy enormous space (same as a football-sized pitch), and have vast power footprints (5-20 MW). Datacenter energy footprint has been estimated to be at 1.3% of global energy usage [66], and to grow at 20% per year due to a rapid pace of deployment of new datacenters.

The information revolution of the last decade has been accompanied by three IT colliding trends. First, the central role of data in our world has resulted in a rapid increase in data that needs to be collected, stored, and processed. IDC estimates a 300-fold-increase in the size of the digital universe over the span of 15 years, totaling over 40 zetabytes (i.e., over 40 billion terabytes) by 2020 [47]. Second, memory density and bandwidth cannot scale up at a sufficient pace, no longer satisfying the massive memory requirements of data-centric IT in an energy-efficient way [129]. Third, the semiconductor manufacturing industry has reached its fundamental efficiency limits, entering a post-Dennard Scaling Era, where on-chip voltages cannot be scaled down at a sufficient pace [27, 41, 45], no longer being able to increase energy efficiency of computing platforms exponentially.

With the growth of the digital universe outpacing technology scaling, energy is becoming the main scalability bottleneck to IT with huge economic and ecological implications. Based on projections, a ten-fold-increase in datacenter energy efficiency is required in the next decade to make IT sustainable. Achieving this goal, however, will require rethinking datacenter design, calling for innovation across all layers of the data-centric computing stack.

1.2 Toward Specialized Memory-Centric Servers

Datacenter operators rely on *scale-out* architectures to deliver a scalable data-centric computing platform. In essence, scale-out datacenters distribute the vast datasets of IT services across a large number of servers, and typically exhibit a low degree of inter-server communication as servers mostly handle independent requests that do not share any state. IT services rely on in-memory processing to boost throughput and lower response latency [12, 19, 94]. As a result, DRAM accounts for a significant share of both acquisition and operating costs of datacenters [9, 66, 83]. Maximizing datacenter efficiency calls for architectures that exhibit high memory resource utilization and minimize the overhead to access memory.

Although there has been a shift toward data-centric IT, servers – the heart of datacenters – still employ processor-centric architectures that were proposed in the early stages of compute-centric IT. In these systems, memory along with storage and networking are built around the processor. The mismatch between the requirements of data-centric IT and traditional computer system architectures [29] leads to severe under-utilization of datacenters' memory resources [28, 67, 80], and consequently poor datacenter efficiency [33].

Delivering sustainable IT infrastructure calls for a paradigm shift in computer system design toward *specialized memory-centric* architectures. Specialized memory-centric architectures seek to ensure efficient usage of memory resources by designing the entire computing stack – including processors [37, 78], networking [93], and software – around memory, and to fit the unique characteristics of data-centric applications [28, 29].

1.2.1 What do data-centric workloads need?

Data-centric workloads, or scale-out workloads, exhibit abundant request-level (e.g., online services) and/or data-level parallelism (e.g., analytics) [21, 28]. The existing parallelism is leveraged through multi-threaded software stacks, where incoming requests in online services are assigned to individual worker threads, and datasets in analytics processing are partitioned and processed by multiple processes or threads [28, 65]. While threads are running on individual cores, they need to access instructions and data.

Instructions. These workloads deploy complex and deep software stacks and heavily use third-party libraries, resulting in multi-MB-sized application instruction working sets [2, 28, 29, 30]. Furthermore, the workloads spent significant fraction of their execution time in the operating system, mainly for network activity [28, 29, 73], resulting in even larger instruction

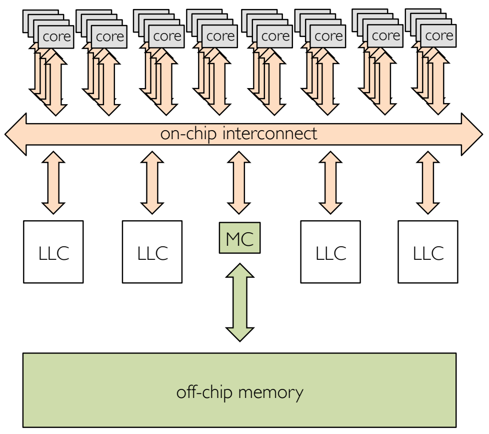


Figure 1.2 – Architecture of a specialized memory-centric server.

working sets. Because of large instruction footprints and their read-only nature, the shared instruction working set is accommodated in the last-level cache (i.e., the last level of the on-chip cache hierarchy). As a result, all cores frequently access the last-level cache (LLC) to fetch instructions.

Data. These workloads need to access vast memory-resident datasets for retrieving request- or task-dependent objects. Due to the disparity between dataset sizes (several tens of GBs) and on-chip cache capacity (few tens of MBs), there is negligible temporal dataset reuse in on-chip caches, resulting in a high incidence of off-chip memory accesses to fetch dataset objects. To allow for constant-time (or sub-linear-time) dataset object retrievals, the datasets are typically organized as pointer-intensive indexing data structures (e.g., a hash table or a tree). Accesses to pointer-intensive data structures, however, result in a limited degree of ILP and MLP within each thread due to high data dependency.

1.2.2 Processor architecture

Specialized processors [37, 78] employ a large number of cores with customized complexity to strike for a balance between available instruction-, memory-, and thread-level parallelism (Figure 1.2). Based on the observation that last-level caches in data-centric workloads exploit mostly instruction-level temporal reuse, specialized processors reduce last-level cache capacity (a few MBs) to free area and power resources in favor of more cores, and to provide fast access to LLC-resident instructions. This specialized processor architecture delivers an order of magnitude higher server throughput over conventional processors while minimizing processor energy consumption in the face of long-latency memory stalls, thereby fully utilizing available memory resources and reducing the energy overhead to access memory.

1.2.3 Efficiency bottlenecks

With specialized processors improving server and datacenter efficiency by almost an order of magnitude [33], the server efficiency bottlenecks are shifting to the memory system, including on-chip interconnects and the on-chip and off-chip memory subsystems. Manycore processors exhibit high incidence of accesses to the last-level-cache (LLC) for fetching instructions, and off-chip memory for fetching dataset objects. Maximizing efficiency calls for on-chip interconnects and memory systems that provide efficient access to LLC-resident instruction footprints and memory-resident datasets.

On-chip interconnects. They serve as the means of communication between cores and the last-level cache. They play a pivotal role in ensuring the performance and power scalability of server manycore chips as they provide the path to performance-critical LLC-resident instructions [28, 29], and communication power is emerging as a significant fraction of the total chip power [35, 118]. Designing an efficient on-chip interconnect is challenging as achieving both low latency and high bandwidth objectives comes at the cost of area/power overheads, prohibitive under fixed area/power budgets.

Multicore processors, featuring 2-16 cores, have relied on conventional crossbar interconnects [112] to achieve uniform and low network latency as well as high bandwidth by connecting each core to all last-level-cache banks. However, as the number of cores (and consequently number of ports) grow, crossbar interconnects face scalability limitations as their area and power footprints scale quadratically with the crossbar radix (i.e., number of crossbar ports) [110], and hence require prohibitive amount of on-chip resources.

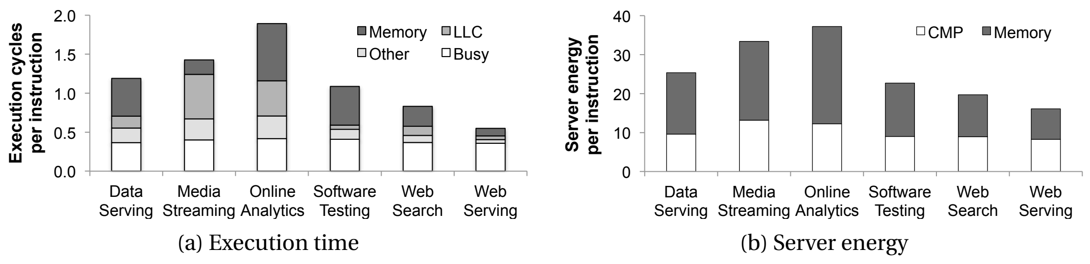


Figure 1.3 – Efficiency bottlenecks in memory-centric servers.

Tiled interconnects, referred to as Networks-on-Chip (NoCs), are emerging as the architecture of choice for providing a scalable interconnect for manycore processors [126] by employing packet-switched architectures and regular topologies, and decoupling the number of cores from the router radix. Their power footprint, however, is emerging as a significant obstacle in the quest for efficient manycore chips [35], accounting for as high as 40% of chip power [39, 118], calling for architectures that maximize NoC performance for a given power budget.

Memory System. The memory system plays a key role in IT efficiency as it hosts and provides access to the vast datasets of data-centric applications. With memory capacity driving memory system design, DRAM manufacturing industry has focused on improving DRAM density rather than DRAM efficiency. Over the past decade, DRAM density improved by 16x (256 Mb in 2004 to 4 Gb in 2014) as opposed to other DRAM parameters, such as latency and frequency. For instance, DRAM latency improved only by 50% (60 ns in 2004 to 40 ns in 2014). Due to the ever-increasing reliance of servers on DRAM, memory system is emerging as the major efficiency bottleneck as it has to serve frequent accesses from many cores to DRAM:

- **Latency.** Over the past decades, memory access latency has steadily increased relatively to the computation time, and hence a significant fraction of the server execution time is spent on waiting for memory accesses to be served by main memory (Figure 1.3a).
- **Bandwidth.** The growth rate in core count outpaces bandwidth scaling of conventional memory interfaces, driving designs into a memory bandwidth wall [49, 105, 129]. Conventional interfaces employ parallel buses to connect the processor to a set of dual-inline memory modules (DIMMs). Unfortunately, parallel interfaces exhibit poor signal

integrity which limits bus frequency [129]. Furthermore, the low pin-count scalability limits the number of memory channels integrated on a commodity processor [49]. Thus, high memory capacity requires that multiple DIMMs are deployed per memory channel, degrading signal integrity, and consequently lowering the bus frequency further.

- **Energy.** Memory energy is emerging as a major energy-efficiency bottleneck in servers, accounting for 48-62% of total server energy (Figure 1.3b) primarily due to architectural choices in memory interface design and DRAM organization.
  - High-speed memory interfaces require energy-intensive DIMM-side clock recovery circuits which are kept active [81] regardless of the bus utilization, resulting in high static power consumption.
  - DRAM memory uses a page-based organization, whereby the first access to a page must activate (or open) the page, requiring significant energy. Once a page is open, subsequent accesses to that page are served from the row buffer, avoiding the high energy and latency cost of a page activation. However, inter- and intra-thread contention on row buffer resources in manycore server processors prevent memory systems from fully exploiting row buffer locality. As a result, page activations are a major contributor to memory energy.

1.3 Thesis Goals

This thesis proposes novel on-chip interconnects and memory systems tuned for the requirements of memory-centric scale-out servers.

Thesis Statement

Architecting high-throughput and energy-efficient memory-centric scale-out servers requires tuning their on-chip interconnect and memory system to fit the common traffic access characteristics of data-centric applications.

1.4 Efficient On-Chip Communication

While today's on-chip interconnects are designed to provide low-latency and high-bandwidth core-to-core and core-to-LLC communication, our study of server workloads demonstrates that their on-chip network activity is dominated by core-to-LLC communication consisting of short requests (for instructions and clean data) and associated long responses.

We propose Cache-Coherence Network-on-Chip (CCNoC), a specialized on-chip interconnect to fit the bimodal network traffic characteristics of servers via a pair of asymmetric request and response networks. The networks are tuned for the type of traffic traversing them and differ in their datapath width and router micro-architecture. CCNoC improves on-chip interconnect area/power efficiency and boosts server throughput under fixed area/power budgets.

1.5 Efficient LLC-Memory Communication

Improving memory system efficiency requires amortizing the costly DRAM page activations over multiple row buffer accesses. Although temporal locality at the LLC in scale-out workloads (due to vast datasets and large reuse distances) is scarce, spatial locality is abundant. Our study of scale-out applications shows that these applications commonly operate on coarse-grained objects (e.g., database rows, memory-mapped files) that are accessed through a pointer-intensive indexing data structure (e.g., a hash table, a tree). However, due to absence of information within the memory hierarchy about memory access patterns, last-level caches and memory controllers fail to exploit the coarse-grained memory accesses of scale-out applications.

We propose Bulk Memory Prediction and Streaming (BuMP) to identify accesses to coarse-grained objects, and trigger bulk transfers of coarse-grained objects between processor and off-chip memory. In doing so, BuMP exploits the spatial locality of scale-out applications and leverages the coarse granularity at which off-chip memory is organized, thereby improving server throughput and memory energy efficiency.

1.6 Scalable Off-Chip Memory Systems

Emerging SerDes-connected memory (SCM) can break the pin bandwidth constraints of modern DDR interfaces and provide the required bandwidth, but at a significant power cost and high latency overhead due to large point-to-point memory networks required to host the vast datasets of scale-out workloads.

Our study of scale-out workloads shows that their memory access distributions are skewed, and hence a small portion of memory accounts for bulk of memory activity. This phenomenon primarily originates from the skewed dataset access distribution found in scale-out applications. For instance, a small fraction of popular users and their pictures in image sharing services account for the majority of user activity. However, in real-world setups with memory sizes of 100s of GBs, the hot dataset exceeds the capacity of on-chip and die-stacked caches.

We introduce MeSSOS, a Memory System for Scale-Out Servers. MeSSOS employs SCM modules as a high-bandwidth cache (HBC) in front of conventional DRAM. As the HBC is effective in filtering most of memory accesses, DCM modules can be clocked at low frequency, thus enabling high memory capacity at relatively low static power overhead. Overall, MeSSOS satisfies the required memory bandwidth and capacity requirements of a scale-out server while minimizing the power consumption of underlying memory technologies and interfaces.

1.7 Thesis Contributions

Through a combination of analytic modeling models, trace-driven analysis, and cycle-accurate full-system simulation of manycore servers, we demonstrate:

- **Bimodal on-chip network traffic.** On-chip network traffic in servers consists of short requests for instructions and clean data, and their associated long responses. In particular, 95% of all network messages fall into one of the two categories.

- **Inefficiency of existing on-chip interconnects.** Existing single- and multi-network on-chip interconnects are sub-optimal as they do not exploit the bimodal network traffic characteristics of servers, leading to incorrect use of available network resources.

- **Efficient on-chip communication.** On-chip interconnect efficiency can be maximized through a pair of asymmetric request and response networks. Each network is optimized for the dominant traffic type, and hence the networks have different datapath width, different buffer architecture, and different pipeline length. Specialization allows for reducing communication delay and area/energy footprints of crossbars and buffers.

- **Bimodal off-chip memory traffic.** Memory accesses in servers occur at fine and coarse granularities. In particular, 59-79% of all memory accesses fall into memory pages with high access density while the majority of remaining access go to low-density pages.

- **Inefficiency of existing memory systems.** Existing memory systems do not exploit the coarse granularity at which memory is accessed and organized due to inter- and intra-core contention on memory resources. State-of-the-art prefetching [113] and scheduled writeback [116] mechanisms can exploit limited row buffer locality as they target only a subset of types of memory accesses. Because row buffer locality is poorly exploited, memory page activations are a major contributor to memory energy.

- **Granularity prediction.** The first access within a page is highly accurate in predicting the granularity at which the page will be accessed for both memory reads (corroborating prior work on spatial footprint prediction [68, 113]) and memory writes.

- **Efficient off-chip communication.** Memory system performance and energy efficiency can be improved by exploiting the coarse-grained memory access patterns. A simple predictor can identify high-density pages and enforce bulk transfers between processor and off-chip memory upon the first access to a page, with minimal on-chip power (50mW) and low storage (14KB) overhead.

- **Dataset popularity.** Dataset popularity in servers is highly skewed, so that a hot portion of memory (5-10%) serves the bulk of memory accesses (65-95%). In real-world setups with memory sizes of hundreds of GBs, the hot portion of memory considerably exceeds the capacity of on-chip and die-stacked caches.

- **Scalable off-chip memory systems.** Conventional DRAM and emerging SerDes-connected memory show complementary characteristics with regards to capacity, bandwidth, and power consumption. Skewed dataset access distributions can be leveraged for architecting a heterogeneous memory system by employing SerDes-connected memory as a cache in front of conventional DRAM.

- **Die-stacked DRAM caches are obsolete.** There is a disparity between die-stacked cache capacity and the hot dataset size. Due to their inability in exploiting temporal reuse, die-stacked caches provide marginal system efficiency gains when integrated to a system that utilizes emerging SerDes-connected memory modules.

The rest of this thesis is organized as follows. In Chapter 2, we introduce CCNoC, a specialized on-chip interconnect for efficient core-LLC communication. In Chapter 3, we present BuMP, a low-cost enhancement to the on-chip memory system for efficient LLC-memory communication. In Chapter 4, we present MeSSOS, a scalable off-chip memory system organization for satisfying the required memory bandwidth and capacity of a scale-out server. Finally, we discuss related work in Chapter 5, and conclude in Chapter 6.

2 Specialized On-Chip Interconnects for Efficient Core-LLC Communication

Processor manufacturers are resorting to chip multiprocessors (CMPs) with a large number of cores [37, 126, 127] to leverage the abundant request-level parallelism found in server applications [29, 78]. In light of scalability limitations of crossbar-based CMPs, emerging manycore chips rely on a network-on-chip (NoC) and a tiled organization to lower design complexity and improve scalability. Recent research has identified high NoC power consumption as a significant obstacle in a quest for efficient manycore chips [35], calling for energy-efficient network architectures. Multiple networks have been proposed as a practical way to improve NoC efficiency in tiled CMPs, and have been examined extensively in the context of general-purpose chips [7]. In this chapter, we seek to architect multi-network NoCs for server CMPs, and to optimize their energy efficiency by designing each network to fit the traffic characteristics of the dominant message types of server workloads.

2.1 Multi-Network NoCs: The Way to Specialization and Efficiency

Building an efficient NoC can be challenging as achieving both low latency and high bandwidth objectives comes at the cost of area and power overheads, often prohibitive under fixed area and power budgets. Balfour and Dally [7] advocate using multiple networks as a practical way to improve performance, power, and area in tiled chip multiprocessors.

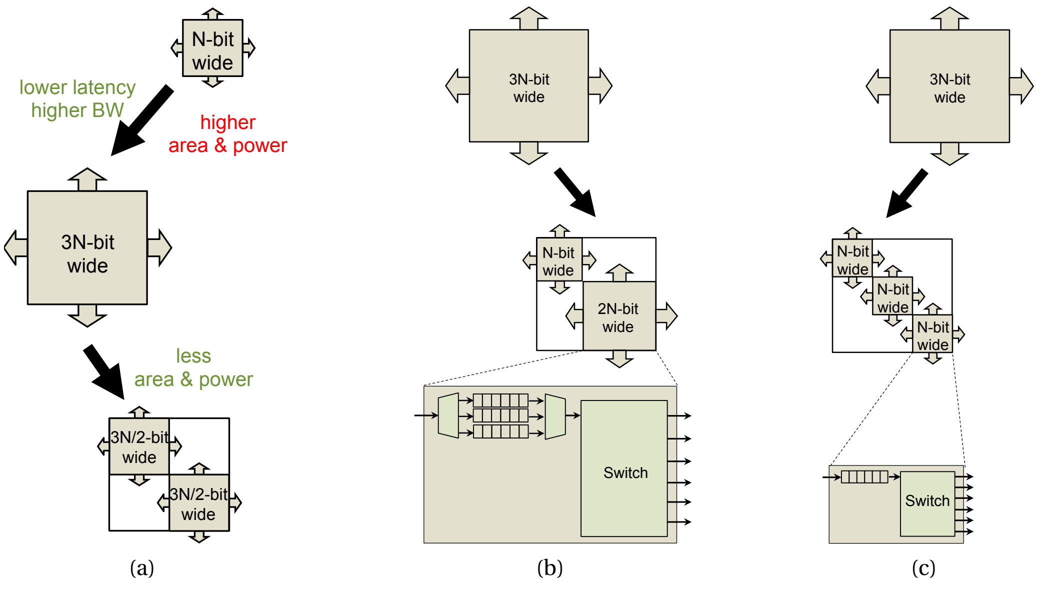


Figure 2.1 – Multi-network NoCs can improve network efficiency (a). Prior work has examined splitting networks by either message size (b) or message class (c).

2.1.1 Why multi-network NoCs?

As Figure 2.1a shows, network performance can be improved by increasing the network width, thus providing high bandwidth and lowering network latency for cache-block-sized network messages, but at the cost of (a) poor wire utilization when transferring short messages, and (b) higher area and power overhead; to allow a regular and compact layout, crossbars are built as canonical matrices in which one edge consists of input ports and the other edge consists of output ports. Each edge of the crossbar must be wide enough to accommodate $Ni*w$ input signals and $No*w$ output signals, where Ni and No denote the number of input and output ports, and w denotes the network width. As a result, crossbar area and power footprints grow quadratically and linearly with the network width [7].

Prior work has advocated using multiple narrow networks to improve NoC efficiency [7]. Narrower networks allow for reducing area and power for fixed NoC bandwidth due to smaller area and power crossbar footprints. Increase in network latency (transferring a cache-block-sized message through a narrow network requires additional flits/cycles as compared to a wider network) is mitigated by the lower load of each individual network. Finally, narrower

Table 2.1 – Message classes and types for the MESI coherence protocol.

Message Class	Message Type	Message Size
Request	Instruction Fetch	Short
	Read/Write	Short
	Upgrade	Short
	Evict Clean	Short
	Evict Dirty	Long
Coherence Request	Downgrade	Short
	Invalidate	Short
Response	Instruction Reply	Long
	Read/Write Reply	Long
	Upgrade Reply	Short
	Invalidate/Downgrade ACK	Short
	Invalidate/Downgrade Update ACK	Long

networks improve wire utilization when transferring short network messages, thus allowing for better resource use under fixed wire budgets.

2.1.2 Design considerations in multi-network NoCs

A key question in efficient design of multi-network NoCs is *"How to split traffic across multiple networks, while maximizing NoC energy efficiency, and minimizing resource overhead and network complexity"*.

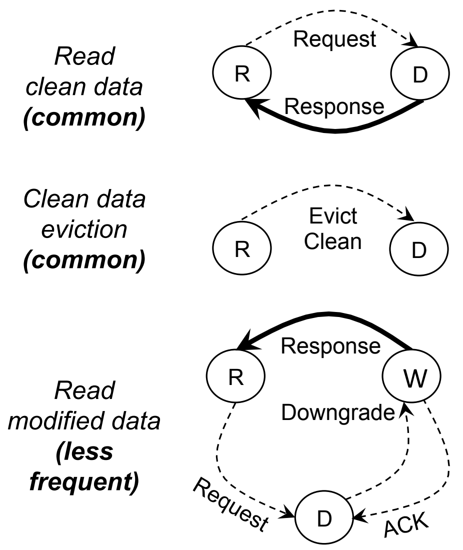
Whereas multi-network NoC design can be simple for simple messaging protocols, design of multi-network NoCs for server chips can be complicated. Server chips rely on cache-coherent architectures for software transparency, and for facilitating software development and porting. Because coherence protocols are critical to performance, they rely on multiple message type and classes to enhance performance (Table 2.1), thus complicating the NoC design:

- **Protocol-level deadlock avoidance.** Due to multiple message classes co-existing in cache-coherence protocols (i.e., requests, coherence requests, and responses), protocol-level cyclic dependencies and deadlocks may occur due to messages sharing network resources. To avoid protocol-level deadlocks, conventional NoCs partition the physical resources at each router's input port among multiple virtual channels to allow independent routing of different message types. An alternative organization consists of multiple physical networks, with a dedicated network for each message class. In both cases, a protocol-level deadlock is avoided by routing messages of each class on a dedicated network (virtual or physical), thereby preventing the formation of cyclic dependencies across message classes.

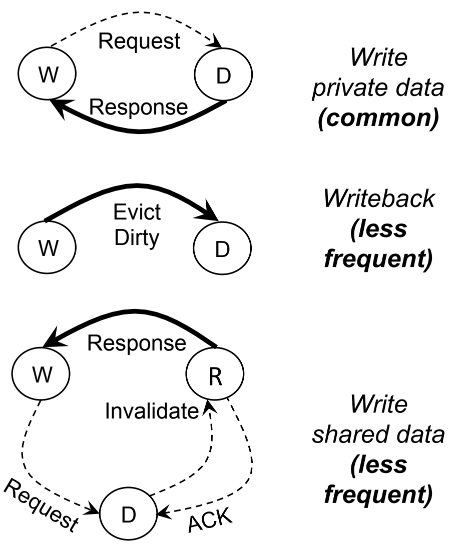
- **Network partitioning.** Network traffic has to be split across multiple networks, requiring non-straightforward design decisions with regards to how to split network traffic and network resources across multiple networks. Existing multi-network NoCs split traffic by either message size or message class:

  - *Message size.* Splitting networks by message size [7] (Figure 2.1b) allows for specializing network width, thus reducing crossbar area and power by a square and a linear factor, respectively. However, such a design introduces high buffer requirements, and router complexity and delay, as each network requires multiple virtual channels to guarantee protocol-deadlock freedom.
  - *Message class.* Splitting networks by message class [126, 135] (Figure 2.1c) allows for eliminating virtual channels, thus lowering buffer requirements and reducing router complexity and delay. However, such designs lead to resource over-partitioning, causing network under-utilization due to traffic variation across classes. As a result, designs that divide network traffic by message class are sub-optimal in terms of both cost and performance.

Multi-network NoCs can greatly improve network efficiency. Unfortunately, existing multi-network NoCs are sub-optimal as they either introduce high resource overhead or lead to sub-optimal performance due to resource over-partitioning, calling for a hybrid organization that takes the advantages of both approaches. To uncover opportunities in hybrid multi-network organizations, we examine the on-chip communication activity of server CMPs.



(a) Protocol transitions initiated by readers.


Write private data (common)
Request
W
D
Response
Writeback (less frequent)
Evict Dirty
W
D
Write shared data (less frequent)
Response
W
R
Invalidate
Request
D
ACK

(b) Protocol transitions initiated by writers.

Figure 2.2 – Cache-coherence protocol transitions initiated by readers (a) and writers (b). Narrow dashed lines denote control messages. Thick solid lines denote messages that carry a cache block. R: Reader, W: Writer, D: Directory.

2.2 On-Chip Communication Activity

As on-chip interconnects provide connectivity among cores, the last-level cache, and the directory, the traffic traversing them is tightly correlated with the cache-coherence protocol activity. Thus, on-chip communication activity will resemble the activity of the most frequent protocol transitions that occur in server applications. In this section, we first review coherence protocol activity in server workloads, and then we examine the implications of the frequent coherence protocol transitions on on-chip network traffic.

2.2.1 Coherence protocol activity

Server applications execute a large number of independent requests, whereby each request accesses a different fraction of the dataset —– e.g., transactions issued by different users have no commonality as each transaction updates a different bank account. As a request is assigned entirely to one core, fine-grained synchronization and data migration between cores is negligible and data reuse happens beyond the residency of cache blocks in L1 caches [28, 40].

To shed light on the coherence protocol activity, Figure 2.2 depicts the protocol transitions for reading (data and instruction) and writing into cache blocks.

Reads. Figure 2.2a shows the communication between a reader core, a directory slice, and a potential writer. In the common case, a reader sends a request for the read-only copy of a cache block, followed by a response from the L2 cache with the data. Similarly, to keep the directory up-to-date with sharer information, clean cache block replacements are also notified with small request messages. In the less frequent case of a read from an active writer of a block, the protocol implements a 3-hop transition. The read request is forwarded to the writer, which then responds directly to the reader and the directory with a data message and a notification response, respectively. Thus, most requests (reads or eviction notifications) for clean blocks are short messages and most responses carry a cache block.

Writes. Figure 2.2b depicts the protocol transitions for writing into cache blocks. In general, write requests (i.e., fetch-block or upgrade requests) are less frequent. Moreover, in server workloads, writebacks account for a negligible fraction of the overall traffic because data are rarely updated and instructions are virtually never modified at runtime [40].

Even among the writes, there are common transitions that fit the duality in traffic. For example, write misses to blocks that are not actively shared have the same request/response behavior as reads of clean blocks. Other transitions include upgrade requests to a non-shared block, requiring a short request message and a short response message as well. Among write requests, those involving other readers and consequently a large number of control messages for both requests and responses are quite rare. In server workloads, data sharing and migration across threads happen over large windows of time – well beyond a typical L1 residency period [40]. As a result, writers rarely modify blocks shared by other cores [77].

2.2.2 On-chip network traffic characterization

Figure 2.3 illustrates the distribution of on-chip traffic across three message classes —– i.e., requests, coherence requests, and responses. We break down request and response message

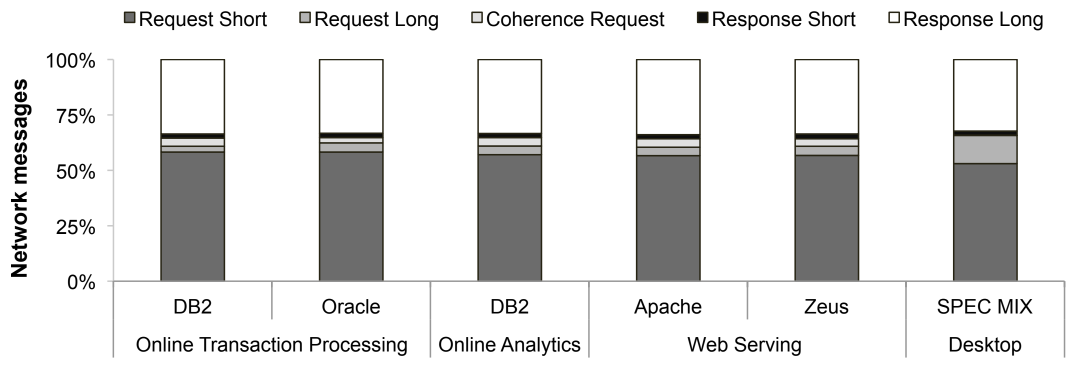


Figure 2.3 – On-chip network message class and size distribution.

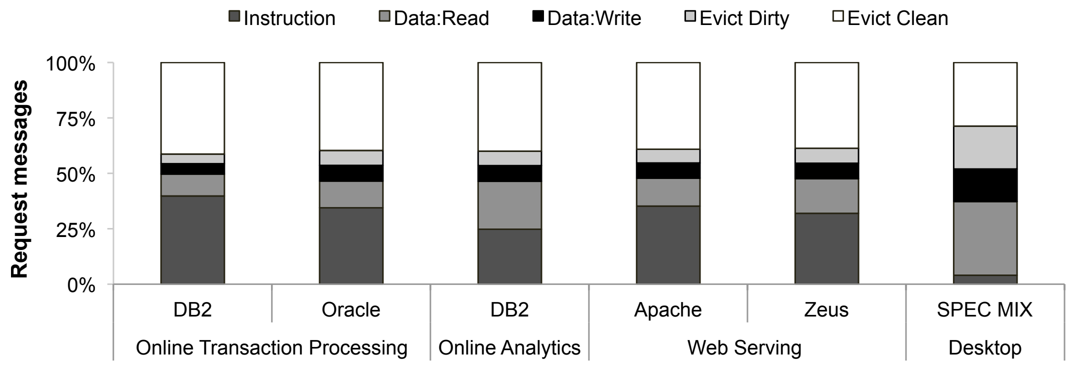


Figure 2.4 – Request message class distribution.

classes to short and long messages. All messages falling into the coherence request message class are short. Figure 2.4 breaks downs the request message class into message types. We examine a wide range of server applications, including online transaction processing, online analytics and web serving running on a 16-core CMP.

The request traffic primarily consists of instruction fetches, data reads, and clean evictions as shown in Figure 2.4. Across server workloads, short messages account for 94% of the request traffic. In Online Transaction Processing and Web Serving, which have big instruction footprints [30], instruction block requests dominate the request traffic, excluding evict traffic. In Online Analytics and Desktop, all with small instruction footprints [30], data requests account for the majority of the request traffic. In servers, the writeback traffic accounts only for 6% of the request traffic, illustrating that clean evictions dominate eviction traffic. In contrast, in Desktop, the writeback traffic accounts for 19% of the request traffic due to a large degree of data write traffic.

Coherence requests, consisting of directory-generated invalidate and downgrade requests, are short and account for a small fraction of the request traffic. For the server workloads, coherence requests account for only 5% of the request traffic. The desktop workload does not exhibit such traffic because each core runs a different application and hence there is not any sharing among the cores.

Figure 2.3 indicates that the response traffic is also highly skewed. On average, long response messages account for 95% of the response traffic. The majority of long responses are messages carrying either instruction or read data cache blocks.

2.2.3 Summary

Server applications are optimized for high reuse in the L1 data cache, whereas their instruction working sets exceed the L1 cache capacity. In the presence of request-level parallelism abundant in server environments, fine-grained sharing between cores is negligible. As a result, coherence activity, and consequently on-chip network activity, is dominated by core-to-LLC communication consisting of short requests and associated long responses.

2.3 Dual-Network NoC Organization

We propose Cache-Coherence Network-on-Chip, or CCNoC, a design that capitalizes on the bimodal network traffic characteristics of cache-coherent servers. CCNoC uses two asymmetric networks to achieve higher efficiency as compared to existing designs; one request network and one response network.

The two networks featured by CCNoC are optimized for the dominant traffic type traversing each of them, and hence have have different datapath widths, different buffer architectures, and different pipeline lengths. Based on the observation that coherence traffic is negligible, CCNoC makes optimal use of wire resources by requiring a minimal number of physical networks, improves network load balance, and boosts performance under a fixed wire budget by supporting a wider response network as compared to a design with multiple dedicated NoCs

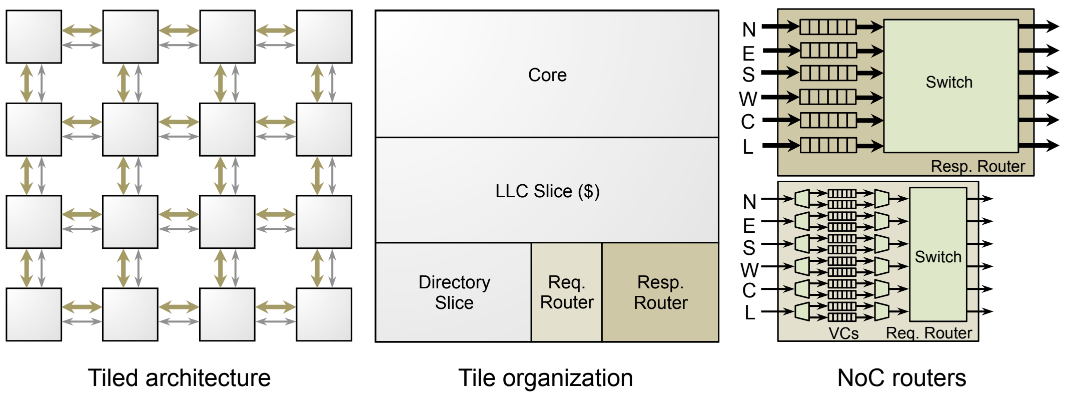


Figure 2.5 – CCNoC-enabled tiled organization. Each tile features a core, LLC and directory slices, and two network routers; one for the narrow request network and the other for the wide response network.

for each message class. Unlike NoCs for CMPs with simple messaging protocols that favor homogeneous networks, NoCs for cache-coherent CMPs require specialization and heterogeneity to best take advantage of the network traffic characteristics presented in Section 2.2.2.

Figure 2.5 depicts the CCNoC architecture with a mesh topology. Each tile consists of two routers, one of them specialized for the request and the other specialized for the response network. Because requests primarily consist of short messages, the request network can be built with a narrow datapath to reduce switch (crossbar) area and power with minimal impact on the system performance. The response messages usually carry a cache block, and hence benefit from wider channels. The NI connecting the core to the network has physical interfaces to both request and response routers. The NI routes requests into the narrow request network and responses into the wide response network. To ensure that the cache coherence protocol is not affected, the receiver NI maintains the order between the request and response messages coming from the same source.

2.3.1 Protocol-level deadlock avoidance

Our system features three message classes: (a) normal requests, (b) coherence requests, and (c) responses. Avoiding protocol-level deadlock among different messages classes requires either dedicated virtual channels (VCs) or different physical networks, as discussed in Section 2.1.2.

Thus, single-network NoCs require three VCs per input port to guarantee deadlock freedom. The same is true for homogeneous dual-network schemes and heterogeneous organizations that split the traffic based on message size.

In contrast, the proposed CCNoC organization segregates requests and responses via dedicated networks, and as a result, requires virtual channels only on the narrow request network to avoid any cyclic dependencies between normal and coherence requests. The wide response network is deadlock-free by default, since all response messages are guaranteed to be consumed at the destination. As such, it does not require virtual channels for deadlock freedom, thereby improving area and energy efficiency through reduced buffer requirements.

2.3.2 Router microarchitecture

Specializing the CCNoC networks to traffic type enables further optimizations at the router level. Conventional single- and dual-mesh topologies use a three-stage router pipeline that consists of virtual channel allocation (VA), switch allocation (SA), and switch traversal (ST) stages. In CCNoC, the VC-enabled request network features the same router pipeline; however, wormhole routers in the response network can be simplified by eliminating the VC allocation stage. This optimization reduces communication delay and diminishes router complexity in the CCNoC response network.

While speculation may also be used to reduce router delay [96], existing schemes tend to increase router complexity and adversely impact cycle time. More generally, speculation and other potential micro-architectural mechanisms do not change the benefits that CCNoC provides as CCNoC builds on the server traffic characteristics in cache-coherent CMPs.

2.4 Evaluation

We compare CCNoC to conventional single-NoC and state-of-the-art dual-NoC networks, and show that CCNoC is superior in terms of area, power, and performance.

Table 2.2 – CMP configuration for cycle-accurate simulations.

Parameter	Value
CMP	16 cores, 2-way OoO, 32-entry ROB and LSQ, 2 GHz
L1-I/D	32KB, 2-way, 64B blocks, 10 MSHRs, 2-cycle load-to-use
LLC NUCA	Unified, 512 KB per tile, 8-way, 64B blocks, 32 MSHRs, 10-cycle hit latency
Memory	3 GB, 45 nsec access latency

2.4.1 Methodology

CMP configuration. We evaluate CCNoC in the context of a manycore CMP with 16 cores and a modestly-sized last-level cache (8 MB). Prior research has shown that large LLC capacities are counter-productive for server workloads [42]. We model a distributed (NUCA) LLC with cache coherence based on the MESI protocol. Cores are modeled after a mobile-class two-way out-of-order core. Table 2.2 summarizes the CMP configuration parameters.

We compare CCNoC to state-of-the-art single- and dual-network topologies. Our reference single-network organization is a mesh-based interconnect with a 176-bit datapath (Mesh-176). We also evaluate a single-network 128-bit mesh (Mesh-128) with the understanding that it offers inferior performance due to lower bisection bandwidth, but higher energy efficiency than the wide mesh.

We consider two dual-network schemes. The first is a Homogeneous organization that features two identical 88-bit networks. In this design, traffic is evenly distributed among the two networks to maximize load balance. The other organization is Heterogeneous, featuring a wide network for long messages and a narrow network for short messages. The networks are 112 and 64 bits wide, respectively. Both single- and dual-network NoCs feature a three-stage router pipeline and three VCs per router input port to avoid protocol-level deadlocks.

The proposed CCNoC architecture features a narrow (64-bit) request and a wide (112-bit) response networks. The request network carries data request messages and coherence protocol traffic. As such, it requires two VCs per router input port for deadlock avoidance and a three-stage router pipeline similar to reference NoC organizations. The wide response network, on the other hand, is dedicated to only one message class. This feature enables a simpler router design based on wormhole flow control with no VCs and a two-stage pipeline.

Technology parameters. We use a custom methodology to assess the energy efficiency of the examined NoC organizations. We target 32 nm technology with an on-chip voltage of 0.9 V and a frequency of 2 GHz. We use detailed wire parameters derived from published sources [7, 49] to model the energy expanded in links and router switch fabrics. To reduce link power, we employ differential signaling with 125 mV swing voltage in network channels routed on an intermediate metal layer [109]. Our crossbars are segmented [122] and use full-swing signaling on local wiring with 2x spacing. Each VC uses six flit buffers. We estimate the energy expanded in flit buffers by modifying CACTI 6 [90] to model shallow FIFO configurations representative of typical NoC routers. We also measure leakage power in router buffers and switch fabrics using models derived from CACTI. In all network organizations, we assume that power gating techniques are applied to eliminate spurious toggling of inactive portions of the datapath. This feature saves power in the cases that the width of the transmitted flit is smaller than the width of the network's datapath [7].

We use Orion 2.0 [57] to estimate the area of router buffers and use our custom methodology to estimate the area of links and crossbar.

Simulation infrastructure. We evaluate CCNoC using full-system simulation using Flexus [125]. Flexus models the SPARC v9 ISA and runs unmodified operating systems and applications. Flexus extends the Simics functional simulator with timing models of out-of-order cores, caches, on-chip protocol controllers and interconnects.

We run a wide range of commercial server workloads, including online transaction processing, online analytics, and web servers. We use the TPC-C v3.0 OLTP benchmark [119] on IBM DB2 v8 ESE and Oracle 10g Enterprise Database Server. We run a mix of queries 1, 6, 13, and 16 from the TPC-H benchmark [119] on DB2. Queries 1 and 6 are scan-bound, Query 16 is join-bound, and Query 13 exhibits a hybrid behavior. To evaluate web server performance, we use the SPECweb99 benchmark on Apache HTTP Server v2.0 and Zeus Web Server v4.3. For comparison, we also simulate a desktop workload that consists of SPEC CPU2K applications running the reference input set. We run this workload on an 8-core CMP as desktop applications lack concurrency and do not benefit from manycore execution substrates [11].

For energy and performance evaluation of CCNoC, we run cycle-accurate simulations using the SMARTS sampling methodology [131]. Our samples are drawn over the entire query execution for Online Analytics, and over an interval of 10 seconds of simulated time for the rest of the workloads. For each measurement, we launch simulations from checkpoints with warmed caches and branch predictors, and run 100K cycles to achieve a steady state of detailed cycle-accurate simulation prior to collecting measurements for the subsequent 50K cycles. To measure performance, we use the ratio of the aggregate number of application instructions committed to the total number of cycles (including cycles spent executing operating system code); this metric has been shown to reflect system throughput [125]. Performance is measured at a 95% confidence level and an average error below 2%.

2.4.2 Comparison to single-network NoCs

We compare CCNoC to single-network NoCs to show the energy and performance advantages of using multi-network organizations.

NoC power. We measure the network power consumption and break it down into major components: buffers, crossbars, and links. For buffers and crossbars, we track both dynamic and leakage power consumption. Figure 2.6 illustrates the network power breakdown for all networks across our server workloads normalized to Mesh-176. The figure shows that CCNoC reduces network power by 28% and 18% when compared to Mesh-176 and Mesh-128, respectively. The power savings of CCNoC compared to these networks are two-fold.

First, CCNoC requires less total flit storage by virtue of requiring fewer VCs than both Mesh-128 and Mesh-176. This feature reduces combined dynamic and leakage buffer power compared to the reference designs by 42% on average. As buffer power accounts for up to 35% of the

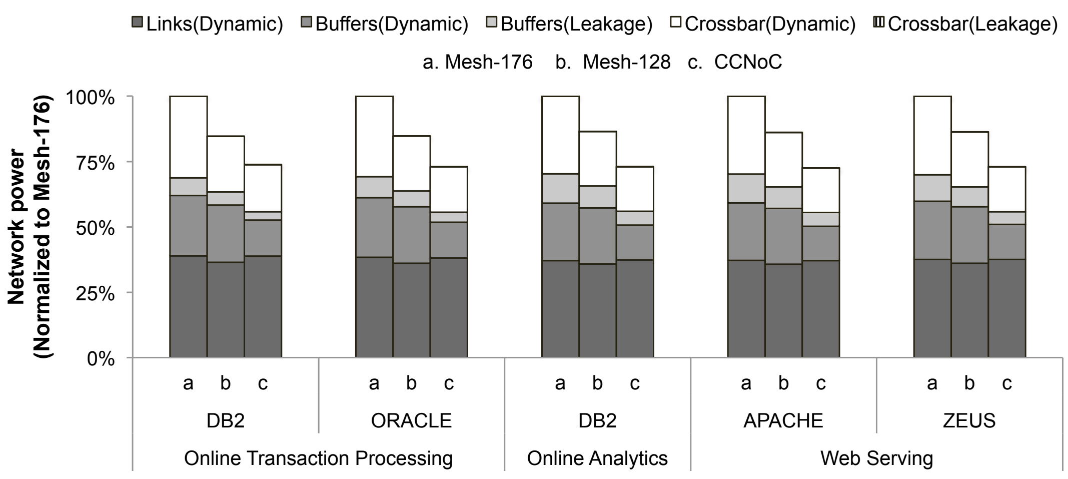


Figure 2.6 – CCNoC power consumption compared to single-network NoCs.

network power, the savings are significant. Second, as noted in Section 2.2.2, a significant fraction of the traffic are short request messages that travel through the narrow *request* network. The compact crossbar in the associated routers diminishes CCNoC's switch power by 40% and 13% when compared to Mesh-176 and Mesh-128, respectively. As crossbars account for 24-30% of the network power in single-mesh topologies, CCNoC considerably reduces their effect on the NoC power consumption.

Performance implications. We evaluate CCNoC's impact on performance by showing both network and system performance across our benchmark suite in Figures 2.7 and 2.8, respectively. Figure 2.7 (above) shows the injection rate (flits/node/cycles) and Figure 2.7 (below) shows the network latency (i.e., number of cycles to transfer a message from source to destination). Compared to Mesh-176, Mesh-128 has a narrower channel width resulting in a higher effective load and, consequently, higher network latency, resulting in a minor loss of performance of 5%.

Long responses in a CCNoC system require one (two) additional flits when compared to Mesh-128 (Mesh-176). However, the extra serialization delay is offset by the reduced load on the *response* network thanks to the separate *request* NoC, as well as the shallower pipeline of wormhole routers in the response network. Figure 2.7 shows that Mesh-128 and Mesh-176 exhibit, respectively, worse and similar injection rate (and network latency), compared

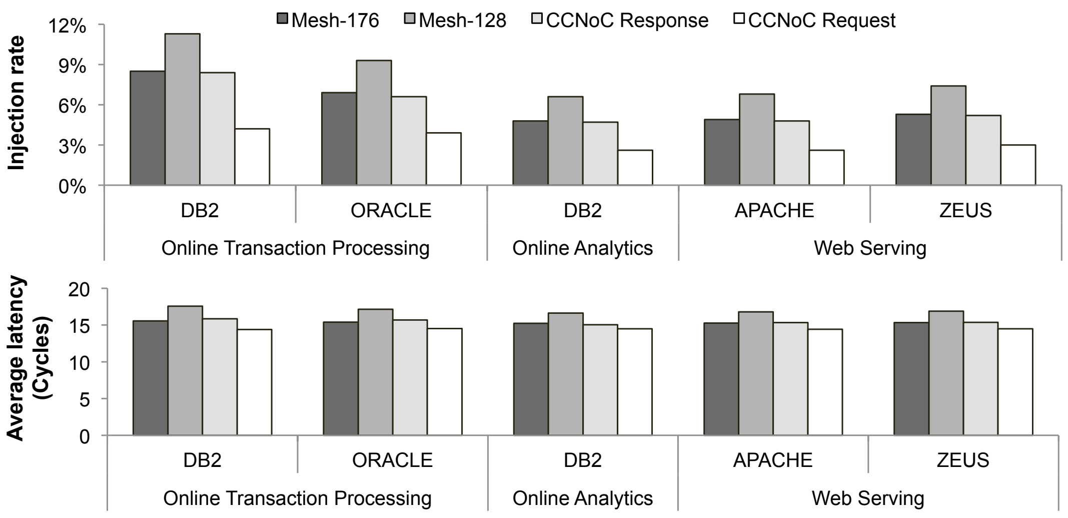


Figure 2.7 – CCNoC network performance compared to single-network NoCs.

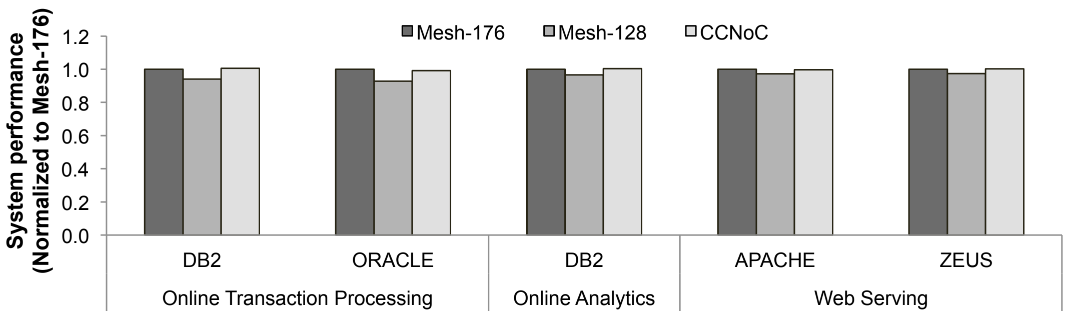


Figure 2.8 – CCNoC system performance compared to single-network NoCs.

to CCNoC's response network. The request network exhibits lower injection rate, because the majority of injected messages are single-flit. Consequently, the request network latency is slightly lower. Overall, as Figure 2.8 shows, a CCNoC-enabled CMP matches the system performance of a Mesh-176 organization and outperforms a design based on Mesh-128 by 5%, and up to 8% for workloads with higher network utilization.

2.4.3 Comparison to dual-network NoCs

We compare CCNoC to dual-network NoCs proposed by Balfour and Dally. The Homogeneous organization splits the traffic across two identical networks to maximize the load balance and thus reduce network congestion. The Heterogeneous design uses a narrow network to transport short messages and a wide network to transport long messages.

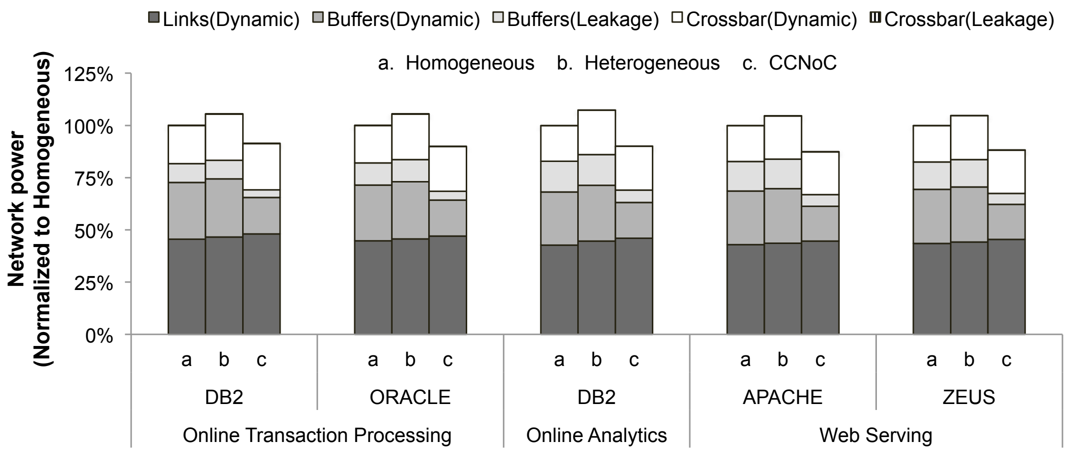


Figure 2.9 – CCNoC power consumption compared to dual-network NoCs.

NoC power. Figure 2.9 shows that CCNoC consumes 11% and 18% less power than Homogeneous and Heterogeneous, respectively. The gains are largely due to the efficient buffer architecture of CCNoC. Compared to other dual-network designs, CCNoC features lower VC and buffer requirements across the two networks reducing dynamic and leakage power draw by 44%, on average. Compared to the single-network Mesh-176 design, all three dual-network organizations are effective in reducing switch power by 40-47% (not shown in the figure).

Performance implications. Figure 2.10 shows the impact of CCNoC on network performance. The figure (above) illustrates the load balance between the two networks. We define load balance as the ratio of the injection rate, in flits, in the narrow network over the injection rate in the wide network. The closer to one this ratio is, the better is the achieved load balance. The figure (below) illustrates the average latency across both networks.

Homogeneous evenly splits the traffic across the two networks and achieves a perfect load balance. In comparison, CCNoC and Heterogeneous achieve a load balance ratio of 0.55 and 0.33, respectively. CCNoC significantly improves the load balance over Heterogeneous due to the presence of long request messages (dirty evictions), which comprise 6% of the request traffic. In CCNoC, these multi-flit messages travel on the narrow *request* network, which improves its utilization. As dirty evictions are not on the critical path, the impact on the system performance is negligible.

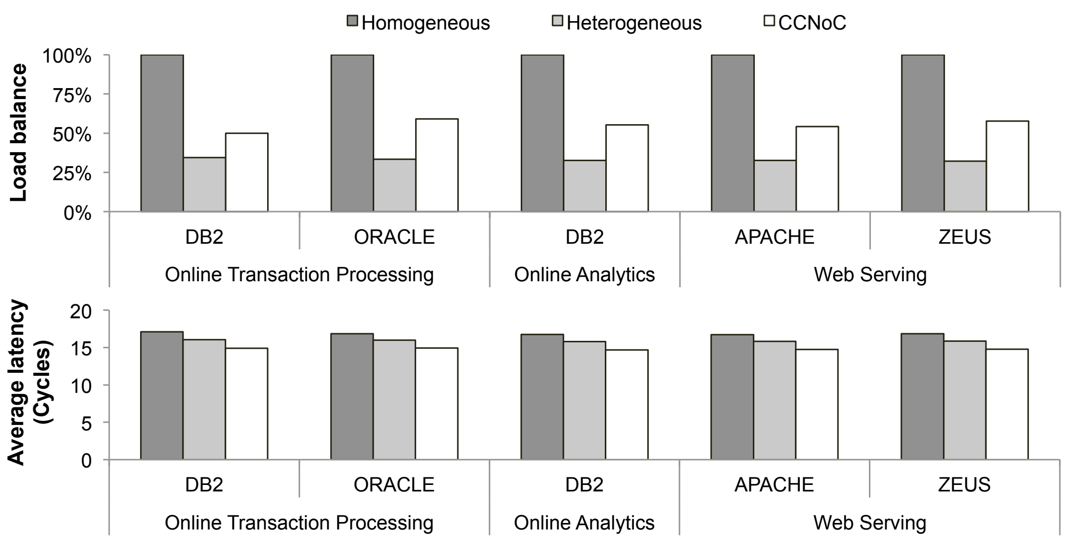


Figure 2.10 – CCNoC network performance compared to dual-network NoCs.

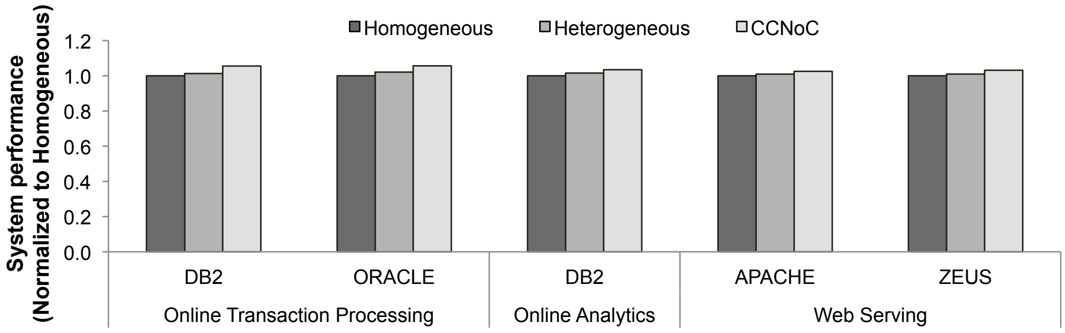


Figure 2.11 – CCNoC system performance compared to dual-network NoCs.

In terms of latency, CCNoC bests other dual-network designs due to its efficient architecture. The majority of long messages traverse the wide *response* CCNoC, which improves performance compared to the narrower networks in the Homogeneous design. The performance is also improved against other dual-network organizations thanks to the shallower 2-cycle router pipeline in the CCNoC *response* network.

Figure 2.11 plots the system performance of various dual-NoC designs. CCNoC slightly outperforms Homogeneous and Heterogeneous by 4% and 3%, respectively, thanks to its architecture that specializes each network to the dominant message class.

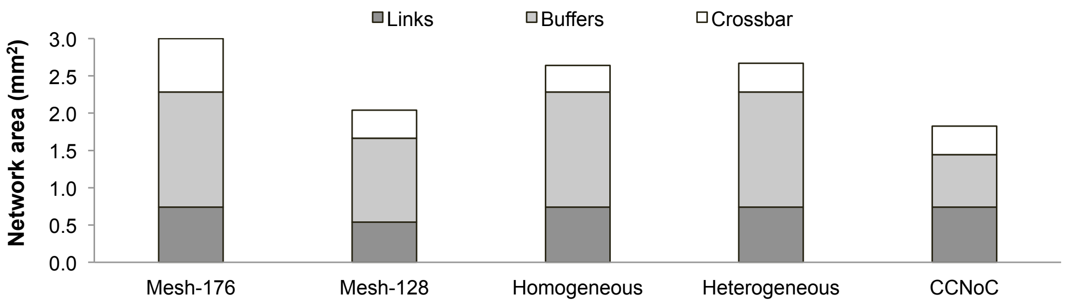


Figure 2.12 – Network area for single-NoC and dual-NoC networks.

2.4.4 NoC area analysis

Figure 2.12 plots the NoC area for various single- and dual-network designs. Compared to designs with same bisection bandwidth, CCNoC reduces network area by 1.4-1.6x. As buffers occupy 52-58% of the area in single- and dual-network designs, the gains are largely due to the efficient buffer architecture of CCNoC. In particular, CCNoC reduces the buffer area by 2.2x compared to Mesh-176, Homogeneous, and Heterogeneous designs. Compared to the single-network Mesh-176 design, all three dual-network organizations are effective in reducing crossbar area by 1.8-2x.

Compared to a single-network NoC design with smaller bisection bandwidth (i.e., Mesh-128), CCNoC reduces the NoC area by 12%. The gains are largely due to the efficient buffer architecture of CCNoC which offsets the increase in the link area. CCNoC reduces the buffer area by 1.6x. As crossbar area is proportional to the square of the datapath width, the cost of adding a narrow network is relatively low; as such, CCNoC and Mesh-128 feature similar crossbar footprints.

2.4.5 Sensitivity analysis

To understand the sensitivity of CCNoC networks to channel bandwidth, we vary the width of the request and response networks while keeping the bisection bandwidth across the system constant. As the size of a request message is eight (8) bytes, an increase of the request network's width above sixty-four (64) bits will not provide any benefit for the dominant type of request

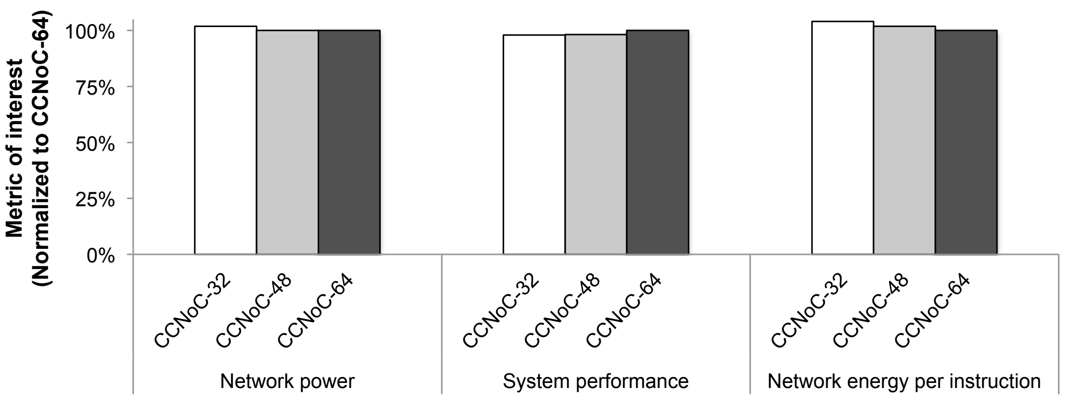


Figure 2.13 – NoC power, system performance, and NoC energy per instruction for various CCNoC design points.

messages (i.e., short). In fact, such a design will degrade performance as the dominant type of response messages (i.e., long) will be penalized traveling through a narrower response network. We also do not consider a request network whose width is smaller than thirty-two (32) bits, because such a design splits short request message into more than two flits, which has an adverse effect of performance.

Figure 2.13 illustrates the power consumption and performance for three CCNoC design points where the request network's width is equal to: a) 32 bits ($CCNoC_{32}$), b) 48 bits ($CCNoC_{48}$), and c) 64 bits ($CCNoC_{64}$). $CCNoC_{48}$ and $CCNoC_{64}$ consume 2% less power than $CCNoC_{32}$. The small difference in power comes from the fact that while $CCNoC_{32}$ enjoys the lowest power in its narrow *response* network, it consumes higher power in its wide *request* NoC.

$CCNoC_{64}$ improves server throughput by 2% over the rest of CCNoC design points due to its wider response network. We estimate the energy consumption per instruction for gauging the power-performance efficiency of different designs. Our results show that $CCNoC_{64}$ is more energy efficient than $CCNoC_{32}$ and $CCNoC_{48}$ by 4% and 2%, respectively.

2.4.6 Summary

We summarize our results by calculating the energy efficiency and energy-delay product of various NoCs with same bisection bandwidth. We use the energy consumption per instruction

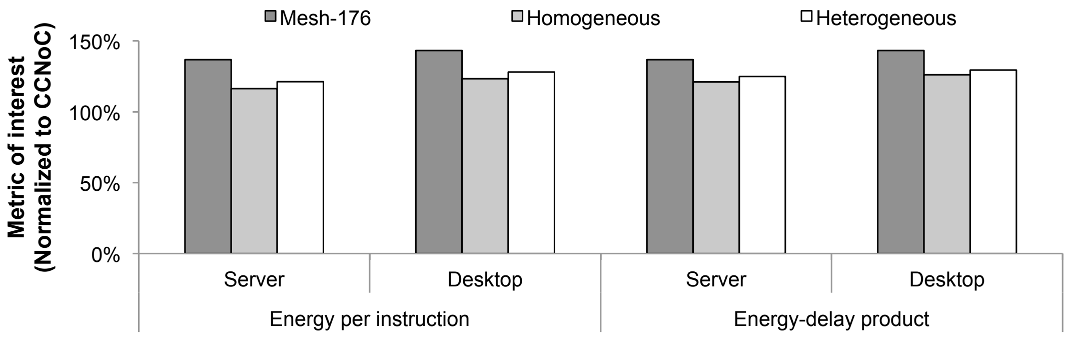


Figure 2.14 – Network energy per instruction and energy-delay product.

as our energy-efficiency metric. We calculate energy-delay product by multiplying energy per instruction by the cycles per instruction (CPI). Figure 2.14 illustrates the efficiency metrics of all network organizations normalized to CCNoC. Across the server workloads, Mesh-176, Homogeneous, and Heterogeneous achieve 37% (37%), 16% (21%), and 21% (25%) higher energy per instruction (energy-delay product) compared to CCNoC.

In workloads with lower network utilization, such as the desktop workload, leakage power constitutes a much higher fraction of the overall buffer power. Because the combined storage footprint of CCNoC networks is lower than that of conventional organizations, CCNoC offers a significant reduction in buffer power. Thus, CCNoC achieves higher network power savings when network utilization is lower. The figure shows that CCNoC improves energy efficiency (energy-delay product) by 23-43% (26-43%) compared to Mesh-176, Homogeneous, and Heterogeneous.

Our findings show that hybrid dual-network NoCs, which split network traffic based on coherence protocol patterns and specialize networks based on the traffic traversing them, are superior to protocol-agnostic single- and dual-network NoCs. Across a wide range of server workloads, CCNoC is more energy (area) efficient compared to single- and dual-network NoCs by 16-28% (31-39%) while improving system performance by up to 8%.

3 On-Chip Support for Efficient LLC-Memory Communication

Emerging manycore designs [37, 112, 115, 126] are well-suited for exploiting the rich request-level parallelism of server applications, while providing greater energy efficiency in the face of frequent long-latency memory stalls [41, 78]. As a result, the memory system is emerging as the efficiency bottleneck as it must serve frequent accesses from many cores to memory-resident datasets. In this chapter, we seek to optimize memory system efficiency by enhancing the last-level cache and memory controllers with information about the memory access patterns of server workloads, so as to leverage the coarse granularity at which DRAM is organized.

3.1 Background and Limitations

We briefly review the basics of today's main memory architecture, and examine limitations in today's server memory systems to exploit the coarse granularity at which DRAM operates.

A DRAM chip houses multiple memory arrays organized in a set of banks. Each bank operates at the granularity of a row, also referred to as a DRAM page. Today's DDR3 memory chips employ a row size of 1024 bytes. A set of DRAM chips, called a rank, is activated together upon a memory access to achieve high bandwidth through parallelism given pin-limited DRAM chips. Multiple DRAM chips are packaged on a DIMM, and multiple DIMMs comprise a memory channel managed by a processor-side memory controller.

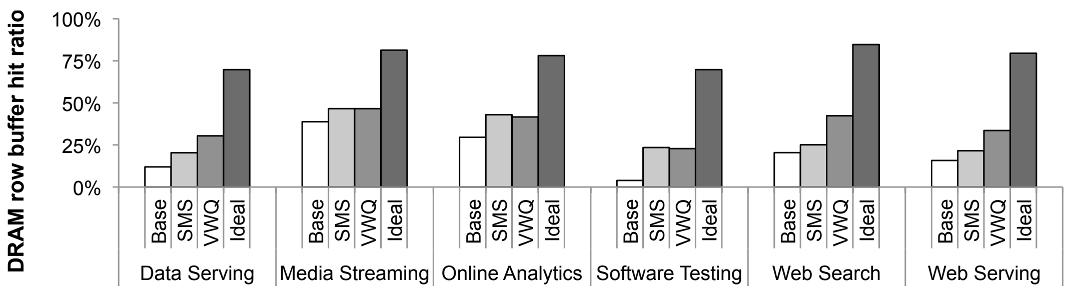


Figure 3.1 – DRAM row buffer hit ratio in servers.

Memory operations comprise DRAM reads, triggered by processor load or store requests, and DRAM writes, which occur upon a dirty eviction from the last-level cache (LLC). A typical memory access consists of two operations. The first is a DRAM page activation, which involves copying an entire DRAM page into the so-called row buffer. The second is the transfer of data to/from the row buffer, consisting of two sub-operations, a burst and the actual I/O activity. Because a DRAM page activation operates at a granularity of a row while a transfer is per cache block, a page activation consumes 3x more energy than a transfer [88].

One way to lower the relative energy expense of activating a page is by amortizing the activation energy over multiple row buffer accesses. For instance, assuming that a processor will access 16 cache blocks within a DRAM row, up to 65% of the memory energy can be saved by fetching all cache blocks at once with a single row activation.

The challenge in maximizing row buffer hits is in discovering accesses to the open DRAM page before that page is closed and a new one is opened. Part of the challenge stems from the over-subscribed memory systems in manycore server processors, whereby memory requests from different cores contend for resources and destroy whatever row buffer locality may be present in another thread's request stream [117]. Another challenge is the frequent occurrence of data-dependent accesses in server workloads [28], which delays requests to a given DRAM page. Finally, massive data working sets of server workloads further compromise row buffer locality by minimizing the likelihood of temporal row buffer reuse across threads.

We quantify the ability of today's servers to exploit DRAM row buffer locality in Figure 3.1. The baseline system integrates a conventional stride prefetcher and employs FR-FCFS open-row policy to exploit row buffer locality by keeping pages open and prioritizing requests to already open pages [104]. Moreover, we include a system that integrates Spatial Memory Streaming [113], a state-of-the-art spatial prefetcher, referred to as SMS. We also include a system that augments the baseline system with Virtual Write Queue [116], a state-of-the art eager writeback mechanism [70], referred to as VWQ. Finally, we include an ideal system that exhibits maximal row buffer locality — i.e., it exploits all available row buffer locality during the residency of a row in the LLC.

Across all workloads, the baseline system achieves a row buffer hit ratio of 21% as compared to 77% of the ideal system. As a result, DRAM page activation energy accounts for a significant fraction of dynamic memory energy (Base-open in Figure 3.9), corroborating prior work [4, 117, 133] and demonstrating that row buffer locality is not fully exploited in server CMPs and calling for techniques to maximize row buffer hits.

SMS utilizes spatial footprint prediction [68, 113] to identify repetitive access patterns and to fetch only the predicted useful cache blocks within a page. As SMS is effective in capturing regular and irregular access patterns, row buffer hit ratio increases to 30%. However, its row buffer hit ratio is limited as it targets only memory accesses that are critical to performance (i.e., load-related traffic), ignoring store-triggered memory reads and memory writes (store misses are hidden through store buffers whereas LLC writebacks are not on the critical path of the processor). As store-triggered reads and memory writes compromise a significant share of memory activity, SMS's energy-efficiency gains are small.

VWQ generates writebacks of adjacent cache blocks upon dirty LLC evictions, thereby exploiting writeback locality and increasing row buffer hit ratio to 36%. However, the row buffer hit ratio is still low as VWQ (a) exploits low degree of read row buffer locality (similar to the baseline system), and (b) performs lookups for a small number of cache blocks within an open row so as to minimize increase in LLC traffic.

Summary. Server applications require large memory capacities, resulting in high memory energy consumption. As dynamic memory energy is dominated by row activations, it is

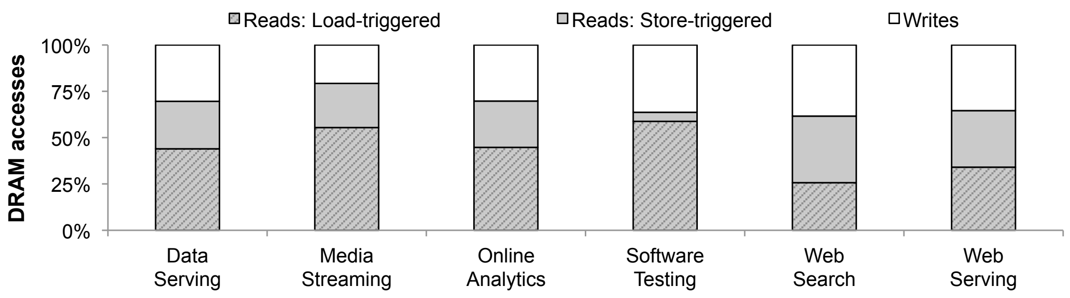


Figure 3.2 – DRAM accesses broken down into reads (load- and store-triggered) and writes (LLC writebacks).

essential to increase the fraction of accesses served by the row buffer to improve average memory energy per access.

3.2 Memory Traffic Characterization

Memory energy efficiency can be improved by exploiting row buffer locality. To uncover opportunities for increasing the row buffer hit ratio, we define region access density as the fraction of cache blocks in a memory region accessed between the first access to the region and the first LLC eviction of a block belonging to the region. The first eviction is a good indicator that the coarse-grained software object is dead with regard to its on-chip usefulness. High-density regions see most of their cache blocks accessed within a modest time window of the first access. Conversely, low-density regions have just one or a few blocks accessed.

We study both DRAM reads (triggered by load and store instructions) and DRAM writes (triggered by LLC writebacks) as both components are important in servers. Figure 3.2 breaks down memory traffic into reads and writes, illustrating that writes account for 21-38% of memory accesses.

3.2.1 Memory reads

Server applications organize and access data at either fine (several bytes) or coarse (few kilobytes) granularities. Examples of fine-grained operations are key lookups in key-value data stores and file systems, hash table walks in data stores [65], and pointer-chasing across

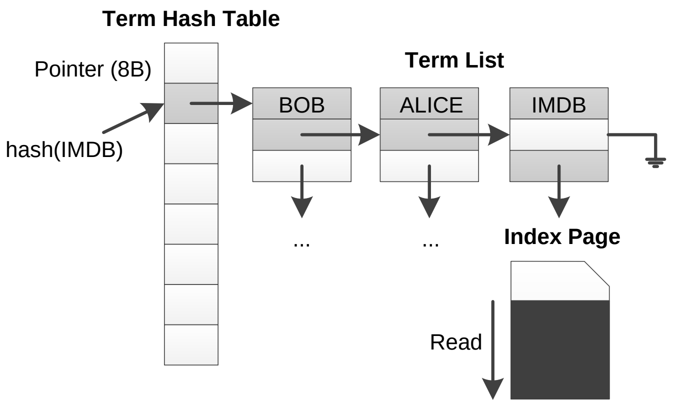


Figure 3.3 – The inverted index in Web Search. Terms are associated with rank metadata.

software objects and operating system structures. Examples of coarse-grained operations include index page traversals in web search, data copying from media files into network packets in streaming servers, row (column) accesses in NoSQL (column-oriented) data stores, and object accesses in object caching systems.

An example that illustrates the phenomenon of operations on fine- and coarse-grained objects is demonstrated in Figure 3.3, which presents the organization of the inverted index in Web Search. The inverted index keeps query terms in a hash table and associates each term with a set of index pages. Index pages store all web pages that contain the term along with rank metadata (e.g., term frequency in the web page). Upon a search query, web pages that contain the term are ranked based on their relevance to the term. First, the term of interest (e.g., IMDB) is looked up in the hash table to find the index pages that contain the term. The hash table walk requires traversing a pointer chain over a large memory space, resulting in low DRAM page access density. Once the matching term is found, the rank metadata for all web pages containing the term is extracted from the index page and used for computing the relevance of each web page. The read of the index page leads to high DRAM page access density due to the contiguous layout of the rank metadata in application memory.

Intuitively, operations on coarse-grained software objects have high DRAM page access density. However, accesses to fine-grained software objects with good spatial locality also result in high DRAM page access density, offering opportunity for exploiting row buffer locality.

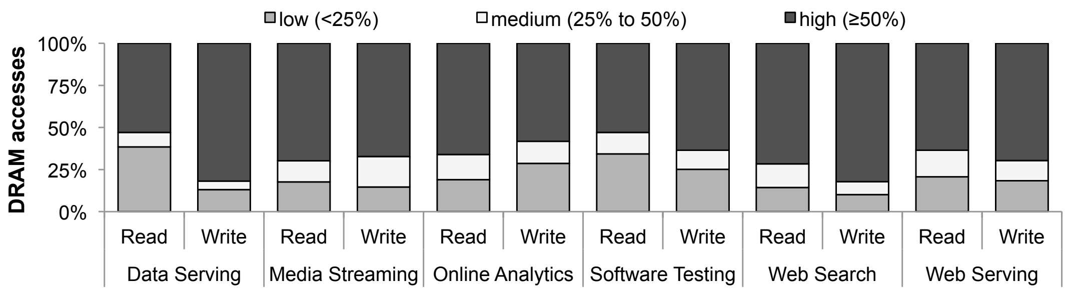


Figure 3.4 – Region access density for a region size of 1KB.

Figure 3.4 shows the region access density of the examined applications. We use a region of 1KB as larger regions do not provide much opportunity in amortizing the energy cost of DRAM page activations. Each segment represents the percentage of cache blocks touched within the region prior to the first LLC eviction of one of its cache blocks. The three segments correspond to high ($\geq$ 50%), medium (25-50%), and low ($<$ 25%) access density.

We observe that memory reads to high-density regions account for a significant fraction of memory accesses, ranging from 57% to 75% (66% on average). Due to their high spatial locality, these regions can be fetched in bulk to maximize row buffer locality. Low-density regions account for most of the remaining accesses, ranging from 17% to 36% (25% on average). Accesses to these regions, such as hashed key look-ups, are difficult to predict and offer little opportunity for exploiting row buffer locality. Finally, a small fraction of accesses go to medium-density regions. Of these, a considerable fraction is attributed to coarse-grained software objects unaligned to region boundaries.

Exploiting row buffer locality on DRAM reads requires identifying accesses to high-density regions upon the first read to the region. Due to high correlation between code and data, the instruction triggering the first access to a region provides information whether the region belongs to a coarse-grained software object. For instance, software developers use a set of functions to access software objects. As such, the first call of those functions can be used to identify accesses to software objects of the same type.

Implications. The majority of reads fall into high-density regions. By identifying such regions using code correlation and by fetching them in bulk, row buffer locality can be improved.

Table 3.1 – Fraction of LLC blocks of a high-density region that are modified once an LLC block within the region is evicted.

Workload	Value	Workload	Value	Workload	Value
Data Serving	8%	Media Streaming	11%	Online Analytics	6%
Software Testing	3%	Web Search	6%	Web Serving	9%

3.2.2 Memory writes

Server applications have a high incidence of store-related DRAM traffic from high-density regions. Often, spatially clustered stores arise when coarse-grained software objects are manipulated. For instance, web servers and NoSQL data stores allocate web pages and frequently-used rows in software caches. Other examples include per-client buffers in a media streaming server to store media packets, and sockets for inter-process communication. While the store instructions themselves result in DRAM reads that allocate blocks in the cache, the subsequent writeback of modified blocks evicted from the LLC trigger DRAM writes.

As noted earlier, DRAM writes account for 21-38% of memory traffic. As Figure 3.4 shows, these writes share similar region density characteristics with DRAM reads. We quantify the fraction of DRAM writes going to high-density regions by measuring the number of modified blocks in a kilobyte-sized region. Across our applications, 62-86% (73% on average) of writes go to high-density regions.

Exploiting row buffer locality on DRAM writes requires eagerly writing back to DRAM all modified cache blocks of a memory region, upon the region's first dirty LLC eviction. Such optimization is effective only if the set of stores defining a memory region as high density have actually completed, meaning that their respective cache blocks have been modified by the time the first block is evicted. We quantify this phenomenon and summarize our results in Table 3.1. On average, only 8% of the blocks of a high-density region are modified after the first dirty LLC eviction within that region. As such, the first dirty LLC eviction is a good indicator that the coarse-grained software object will not be modified in the future.

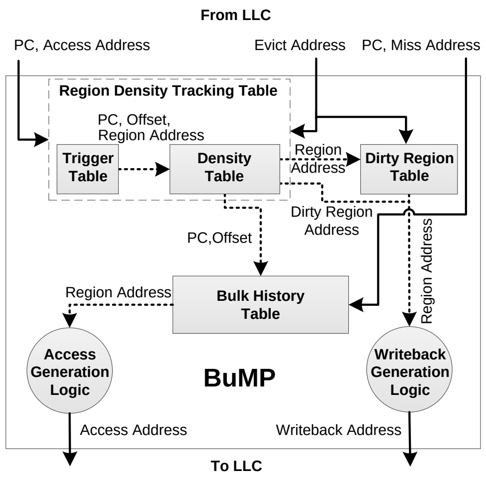


Figure 3.5 – BuMP design overview.

Implications. DRAM writes offer significant opportunity for exploiting row buffer locality in the context of high-density modified regions. The first dirty eviction is a good indicator that the entire region can be written back.

3.3 Bulk Memory Access Prediction and Streaming

As shown in Section 3.2, 59-79% (68% on average) of all memory accesses go to high-density regions. As memory energy per access can be improved by exploiting row buffer locality upon accesses to high-density regions, we propose Bulk Memory Access Prediction and Streaming, or BuMP. We identify DRAM accesses (both reads and writes) to high-density regions and trigger bulk transfers upon a first access to the region, thus eliminating redundant row activations.

3.3.1 Design overview

Figure 3.5 illustrates an overview of the BuMP design. BuMP employs three microarchitectural components shared by all the cores: (a) the region density tracking table to identify high-density regions, (b) the bulk history table to keep prediction metadata for high-density regions, and (c) the dirty region table to keep track of cache-resident high-density modified regions. As

BuMP monitors LLC activity, it benefits from being physically near the LLC but is a standalone component. Therefore, BuMP is not on the critical path of the processor.

BuMP uses the region density tracking table (RDTT) to identify high-density regions. The RDTT monitors the LLC access and eviction streams to track accessed and modified cache blocks within active memory regions. A memory region is considered active in the time interval between its first access (triggering) and its first LLC eviction. Upon an eviction in an active region, the region is terminated.

For terminated regions identified as high-density, the RDTT informs the bulk history table (BHT) with prediction metadata associated with high-density regions. For high-density modified regions terminated due to a clean LLC eviction, the RDTT also informs the dirty region table (DRT).

The BHT monitors LLC misses and uses its prediction metadata to predict if an LLC miss falls into a high-density region. For an access predicted as going to a high-density region, the BHT informs the access generation logic to launch access requests for the region's blocks.

The DRT monitors LLC evictions. Upon identifying a dirty LLC eviction falling into a high-density modified region, the DRT utilizes the writeback generation logic to trigger eager writeback requests for each of the cache blocks in the region.

3.3.2 Bulk memory read prediction and streaming

BuMP employs the RDTT to track the access density of cache-resident regions. Internally, the RDTT is comprised of two tables: the *trigger* table and the *density* table. The trigger table tracks regions with a single accessed block whereas the density table tracks the density of regions with more than one accessed block. Splitting the RDTT into two tables allows for (a) reducing interference between regions with a single access from high-density regions, and (b) energy per access over a unified and larger table as most of the RDTT accesses hit in the density table —– most of accesses go to high-density regions.

The trigger and density tables are built as set-associative structures to minimize conflicts and are accessed using the region address. The region address is calculated by shifting the physical address by the number of region offset bits to the right.

The benefit of BuMP hinges on its ability to accurately identify DRAM accesses to high-density regions. To do so, BuMP's prediction mechanism relies on the observation that there is a high correlation between code and software objects that lead to high-density regions. To account for misalignment of a software object with the beginning of a region, the BuMP predictor augments the program counter (PC) of the instruction triggering the memory access with the *offset*, defined as the distance between the triggering cache block and the beginning of the region. For a kilobyte-sized region, the offset is 4 bits. The PC and offset combination is similar to that used in spatial footprint prediction [17, 113].

The RDTT associates each entry with a *PC,offset* pair. The PC is carried with LLC requests. When an RDTT region identified as having high access density is terminated due to an eviction, an entry is allocated in the bulk history table (BHT). Doing so simply requires indexing the BHT with the PC,offset pair and setting a valid bit.

On an LLC miss, the BHT is probed using *PC,Offset*. In case of a BHT hit (identified through a tag match), BuMP generates cache block requests for the entire region (except for the triggering block) and forwards them to the LLC.

Bulk memory read tracking example. Figure 3.6 illustrates the operation of the RDTT in detail. In event 1, the triggering instruction requests the block *A+2*, missing in both the density and the trigger tables. This results in an allocation in the trigger table. The allocated entry consists of the region address *A*, and the *PC,Offset*.

In event 2, the second access to region *A*, hits in the trigger table, requiring the transfer of the entry to the density table (DT). The DT entry is augmented with a bit vector (noted as pattern) that summarizes the cache blocks that have been accessed within a region. In our example, the third and fourth bits are set to one accordingly. A subsequent access to region *A* in event 3 hits in the DT and updates the pattern accordingly.

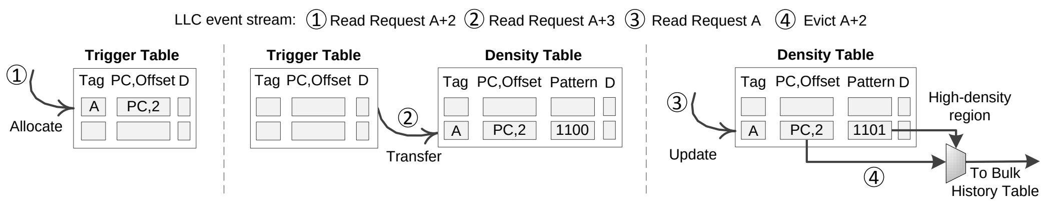


Figure 3.6 – Example for region density tracking and identifying instructions that access high-density regions.

Upon an LLC eviction within an active region or upon a table conflict, the corresponding entry in the density table is invalidated and a simple logic checks if the corresponding region has a high access density, by (a) counting the number of bits set in the pattern, and (b) comparing the resulting density to a pre-defined threshold (event 4 in Figure 3.6). If the resulting density is high, the *PC,Offset* is inserted into the bulk history table; otherwise, it is only invalidated.

3.3.3 Bulk memory write prediction and streaming

BuMP relies on LLC dirty evictions to enforce bulk DRAM writes. In particular, BuMP leverages the observation that the first LLC dirty eviction is a good indicator that a high-density modified region will not be modified prior to its eviction from the last-level cache.

BuMP uses the region density tracking mechanism to identify modified high-density regions by extending the density and trigger tables with a dirty bit which is set upon receiving a write/writeback request from the L1 data cache. Upon an LLC dirty eviction within an RDTT region identified as modified high-density region, BuMP triggers bulk writeback requests to the LLC for the entire region. This operation is denoted by the dashed line labeled as *Dirty Region Address* in Figure 3.5.

In practice, we find that most region terminations in the density table occur due to a table conflict —– before the first dirty LLC eviction within the region. Therefore, to minimize premature writebacks, BuMP employs the dirty region table (DRT) to keep track of cache-resident high-density modified regions that are evicted from the density table. The DRT is set associative and is accessed using the region address.

Upon an LLC dirty eviction, BuMP probes the DRT to check whether the cache block belongs to a high-density region. In case of a hit, BuMP generates bulk writebacks (except for the triggering block), forwards them to the LLC, and invalidates the corresponding DRT entry.

3.3.4 Transfer of program counter

BuMP's read prediction mechanism requires knowledge of instructions that miss in the last-level cache. As this information is not available in the last-level cache, our design needs to extract the program counter (PC) from the pipeline at a memory instruction basis, and then transfer the extracted PC information along with read/write requests to the memory hierarchy.

- **PC extraction.** Instruction-based prefetching mechanisms employed by conventional processors (e.g., stride prefetcher) provide means of extracting the PC from the pipeline as they correlate load instructions with stride access patterns. As such, the PC information is available in the L1 cache.
- **PC transfer.** After extracting the PC information from the pipeline, we need a way to transfer the PC to the last-level cache and/or the memory controller. We leverage the observation that the most significant bits of a memory address are not used in today's processors. Although memory addresses are 64-bit, the memory address space is smaller and only 48 bits are required. As such, we hash the PC obtained by the L1 cache to the 16 highest bits of the memory address, and rely on the on-chip interconnect to move the information along with the memory request.

3.3.5 Configuration and hardware cost

BuMP employs a region size of 1KB, which corresponds to 16 cache blocks and allows for amortization of the DRAM page activation energy. The threshold for labeling regions as having high access density is eight cache blocks. This configuration balances the opportunity for energy savings by targeting a large fraction of DRAM accesses with limited tolerance for overfetch (Section 3.2).

Memory controller. BuMP employs FR-FCFS open-row policy [41] with region-level address interleaving. BuMP maps an entire region to one DRAM row by using the addressing scheme *Row:ColumnHigh:Rank:Bank:Channel:ColumnLow:WordOffset*, where ColumnHigh is 3 bits and ColumnLow is 7 bits (complements the region offset). Although BuMP's addressing scheme diminishes parallelism (accesses to consecutive cache blocks within a kilobyte-sized region are serialized) accesses to a high-density region serve as prefetches that counter-act the serialization delay.

Hardware cost. For our server workloads, an RDTT with 256-entry trigger table (2.5KB) and 256-entry (3KB) density table provides almost the same accuracy as a predictor with unlimited storage (results not shown) except for Software Testing. Both DRT and BHT employ 1024 entries each to minimize premature writebacks and to maximize prediction accuracy, for 4.25KB and 4.5KB of storage, respectively. All structures are 16-way set-associative. In total, BuMP requires 14KB.

3.4 Evaluation

We first evaluate BuMP's accuracy in identifying high-density regions. Then, we examine the energy and performance implications of bulk memory streaming including BuMP's on-chip energy and bandwidth overheads. Finally, we include a comparison between BuMP and prior proposals on prefetching and eager writeback mechanisms.

3.4.1 Methodology

Baseline systems. We evaluate BuMP in the context of a manycore CMP with 16 cores, a modestly sized last-level cache, and a crossbar-based NOC that minimizes the delay to the LLC. Prior research has shown that large LLC capacities are counter-productive for server workloads and that a fast path to the LLC is critical to server processor performance [29, 28, 78, 112]. The chip is modeled in 22nm technology with high-performance process for all the components except the LLC which uses low-leakage process. The chip features two DDR3-1600 channels.

Table 3.2 – System configuration for cycle-accurate simulations.

Parameter	Value
CMP	16 cores, 3-way OoO, 48-entry ROB and LSQ, 2.5 GHz
L1-I/D	32KB, 2-way, 64B blocks, 10 MSHRs, 2-cycle load-to-use
LLC	Unified, 4 MB, 16-way, 64B blocks, 8 banks, 8-cycle hit latency
NoC	16x8 crossbar, 5 cycles
Memory	16 GB, 4 ranks per memory channel 2Gbit, x8, 8 banks per rank, 8KB row buffer 2 DDR3-1600 channels (Max. BW: 25.6GB/s) close- and open-row FR-FCFS policy [104] 64-entry transaction and command queues
tCAS-tRCD-tRP-tRAS tRC-tWR-tWTR-tRTP-tRRD-tFAW	11-11-11-28 39-12-6-6-5-24

Cores are modeled after a high-end mobile-class three-way out-of-order core [36]. Table 3.2 details the parameters for the processor.

Our baseline systems employ a stride prefetcher that predicts stride accesses if two consecutive addresses accessed are separated by the same stride, and prefetches the subsequent four cache blocks into the last-level cache.

We consider two memory controller configurations for our baseline systems: (a) FR-FCFS close-row policy with block-level address interleaving, referred to as Base-close, and (b) FR-FCFS open-row policy with region-level address interleaving, referred to as Base-open, same as BuMP's. Base-close maximizes channel-/rank-/bank-level parallelism by distributing consecutive cache blocks across channels/ranks/banks, thereby reducing serialization delays upon sequential accesses. This is accomplished by using the addressing scheme *Row:ColumnHigh:Rank:Bank:Channel:ColumnLow:WordOffset*, where ColumnHigh is 7 bits and ColumnLow is 3 bits (complements the cache block offset).

We also evaluate SMS, a state-of-the-art spatial prefetcher [113]. SMS was originally designed for performance and was integrated next to the core at a storage cost of 60KB per core. In this work, we incorporate SMS next to the last-level cache. This optimization allows for higher energy-efficiency gains due to higher prediction accuracy while reducing the storage costs

Table 3.3 – Power and energy for system components.

Parameter	Value
Core	Peak Dynamic Power / Leakage Power: 700mW / 70mW
LLC	Read/Write Energy: 0.63nJ/0.70nJ, Leakage Power: 750mW
NoC	Peak Dynamic Power / Leakage Power: 55mW / 30mW
Memory Controller	Dynamic Power @ 12.8GB/s: 250mW
DRAM (per 2GB rank and 64-byte transfer)	Background Power: 540-770mW Activation Energy: 29.7nJ Read/Write Energy: 8.1nJ/8.4nJ I/O Termination (Read/RRead): 1.5nJ/3.8nJ I/O Termination (Write/RWrite): 4.6nJ/4.6nJ

as prediction metadata can be shared across cores. The memory controller is configured the same way as BuMP's.

Finally, we consider a state-of-the-art eager writeback mechanism [116], referred to as VWQ. The writeback mechanism generates eager writeback requests for three adjacent cache blocks upon a dirty LLC eviction. The memory controller is configured the same way as BuMP's.

Energy modeling framework. We use energy consumed per instruction as our energy-efficiency metric. To measure energy per instruction, we develop a custom energy modeling framework to include various system components, such as cores, network-on-chip (NoC), caches, memory controllers, and main memory. Our framework, summarized in Table 3.3, draws on several specialized tools to maximize fidelity through detailed parameter control.

We estimate dynamic core power by scaling published measurements of dynamic power on a high-IPC workload by the ratio of the actual workload IPC and the reference IPC [4, 52, 71]. We measure core leakage power using McPAT [72]. We use CACTI's models to obtain LLC read and write energy estimates and we account for advanced LLC leakage reduction techniques [71, 90]. We obtain NoC power from McPAT, finding it to be negligible compared to other system components; hence, we assume constant NoC power. We use McPAT to measure memory controller's dynamic power at peak memory bandwidth. We estimate DRAM background power and energy based on Micron models and data sheets [86, 88].

Simulation infrastructure. We evaluate BuMP using full-system simulation using Flexus [125]. Flexus models the SPARC v9 ISA and runs unmodified operating systems and applications. Flexus extends the Simics functional simulator with timing models of out-of-order cores, caches, on-chip protocol controllers and interconnect, and DRAM. DRAM is modeled by integrating DRAMSim2 [106] directly into Flexus. DRAMSim2 is instantiated based on DDR3 device specifications [86]. Table 3.2 details the simulated architecture.

We evaluate BuMP using contemporary server workloads. The workloads, taken from CloudSuite 2.0 [28], include Data Serving, Media Streaming, Web Search, and Web Serving (frontend). We evaluate online analytics on a commercial database. In particular, we run a mix of queries 1, 6, 13, and 16 from the TPC-H benchmark. Queries 1 and 6 are scan-bound, Query 16 is join-bound, and Query 13 exhibits a mixed behavior [42]. We consider a large-scale scientific task that might run in the cloud. In particular, we benchmark one instance per core of the Klee SAT Solver, an important component of CloudSuite 2.0's Software Testing workload.

For energy and performance evaluation of BuMP, we run cycle-accurate simulations using the SMARTS sampling methodology [131]. Our samples are drawn over the entire query execution for Online Analytics, and over an interval of 10-30 seconds of simulated time for the rest of the workloads. For each measurement, we launch simulations from checkpoints with warmed caches and branch predictors, and run 800K cycles (2M cycles for Data Serving) to achieve a steady state of detailed cycle-accurate simulation prior to collecting measurements for the subsequent 400K cycles. To measure performance, we use the ratio of the aggregate number of application instructions committed to the total number of cycles (including cycles spent executing operating system code); this metric has been shown to reflect system throughput [125]. Performance is measured at a 95% confidence level and an average error below 2%.

3.4.2 Region density prediction accuracy

We evaluate BuMP's ability to exploit reads to high-density regions by measuring the number of cache blocks that are correctly fetched prior the processor's request, referred to as predicted

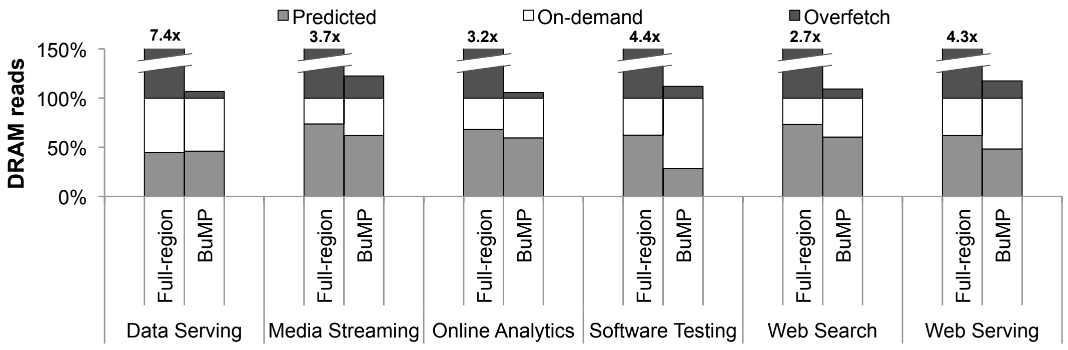


Figure 3.7 – BuMP memory read prediction accuracy.

DRAM reads. Similarly, for writebacks we measure the fraction of blocks that are timely written back to memory, referred to as predicted DRAM writes. To show the importance of triggering bulk transfers only upon accesses to high-density regions, we include a system that always triggers bulk transfers [74, 130], referred to as Full-region.

DRAM reads. Figure 3.7 plots the fraction of predicted DRAM reads as well as the overfetch rate. Across all applications except for Software Testing, BuMP predicts 45-55% of DRAM reads at the cost of a small overfetch rate (5-22%) due to its ability to identify accesses to high-density regions and to enforce bulk transfers upon an LLC miss.

For Software Testing, BuMP predicts only 28% of DRAM reads, despite the high fraction of memory accesses to high-density regions (57%). This disparity is attributed to the large number of active regions that compete for space in the region density tracking table (RDTT). Our analysis of an RDTT with 256-entry and 2048-entry trigger and density tables showed that BuMP can predict up to 44% of DRAM reads.

The figure also plots the coverage and overfetch rate of Full-region, a design that fetches the entire region upon the first access to the region. The coverage increases from 50% (BuMP) to 63% (Full-region), on average. However, the small increase in coverage comes at the cost of prohibitive overfetch (4.3x, on average) and cache thrashing. In case of workloads with a high number of accesses to low-density regions (Data Serving), Full-region loses coverage compared to BuMP due to excessive cache thrashing.

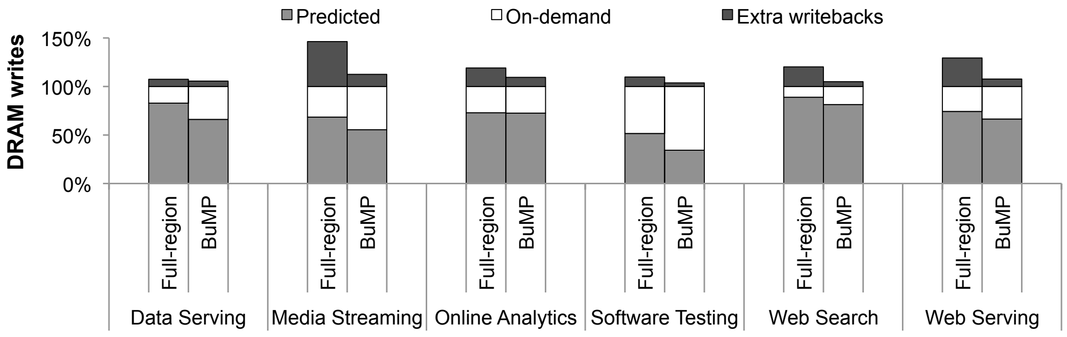


Figure 3.8 – BuMP memory write prediction accuracy.

Table 3.4 – BuMP's DRAM row buffer hit ratio.

Workload	Value	Workload	Value	Workload	Value
Data Serving	54%	Media Streaming	64%	Online Analytics	57%
Software Testing	34%	Web Search	62%	Web Serving	56%

DRAM writes. Figure 3.8 plots the BuMP coverage (i.e., fraction of predicted DRAM writes) as well as the extra writebacks that result from premature writebacks and pressure to the last-level cache due to overfetched cache blocks.

On average, BuMP predicts 63% of DRAM writes, matching our observation that most of the writeback traffic comes from high-density regions. Similar to DRAM reads, BuMP achieves the lowest coverage for Software Testing.

The increase in writeback traffic due to BuMP is low, less than 10% across all applications, as the first LLC dirty eviction within a high-density region is a good indicator that the region will not be modified. The figure also plots the coverage of DRAM writes for Full-region. Full-region triggers bulk writeback requests to the last-level cache upon every dirty LLC eviction. As a result, the DRAM write coverage increases from 63% (BuMP) to 73% (Full-region). However, this increase comes at the cost of an increase in writeback traffic — from 7% to 22%, on average, due to premature writebacks. This is because the first eviction within a low-density region is not a good indicator that the region will be written back as cache blocks within low-density regions belong to different software objects.

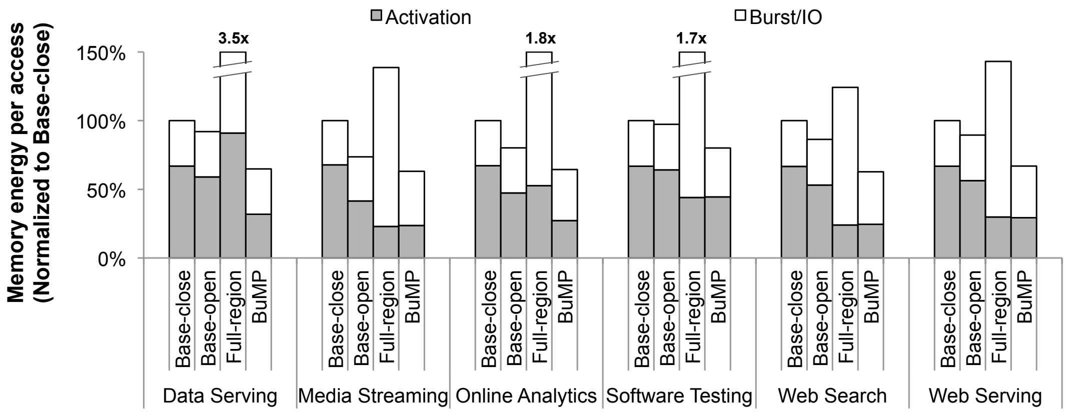


Figure 3.9 – Memory energy per access for various systems normalized to Base-close.

Overall. BuMP predicts 55% of all memory accesses at a small increase in memory traffic (11%). By doing so, BuMP recovers 87% of the locality that exists in high-density regions, thus achieving high row buffer hit ratio (Table 3.4).

3.4.3 Energy efficiency implications

We examine the implications of bulk memory streaming on memory system energy efficiency. Figure 3.9 illustrates the memory energy consumed per access for BuMP normalized to our baseline systems: Base-close and Base-open.

Compared to Base-close, Base-open increases row buffer hit ratio to 21%, as it keeps pages open and prioritizes requests to already open pages. The increase in row buffer hit ratio allows for reducing memory energy by 14%. However, the row buffer hit ratio is low due to interleaving of accesses among cores, preventing the memory controller from exploiting high row buffer locality.

BuMP achieves high gains in energy efficiency compared to the baseline systems. On average, BuMP reduces energy per access by 34% and 23% compared to the close- and open-mode baseline systems, respectively, thanks to its ability to exploit high row buffer locality upon an access to a high-density region.

The figure also plots the memory energy consumption of Full-region, showing that Full-region achieves the worst energy efficiency due to excessive overfetch. Furthermore, we find that for bandwidth-intensive workloads (Data Serving, Online Analytics, and Software Testing), contention in queuing structures in the LLC and the memory controller prevents requests within the same region to arrive within the memory controller's scheduling window. Thus, the memory controller cannot fully exploit the open-mode benefits of DRAM. Furthermore, this phenomenon results in a high number of additional row activations even for blocks that will not be accessed by the processor. In the extreme case (Data Serving), the combination of additional row activations and extreme LLC thrashing results in higher activation energy for Full-region than for the baseline systems.

3.4.4 Performance implications

While our primary motivation is to improve energy efficiency, we find that BuMP also improves system performance. Figure 3.10 plots the performance improvement of various systems over Base-close, a system that employs a stride prefetching scheme and maximizes DRAM parallelism. Base-open delivers 1-2% lower performance than Base-close as it delays precharging open pages, penalizing future accesses going to a different page in the bank. BuMP outperforms Base-close by 9% and Base-open by 11%, as bulk transfers from memory allow for fetching cache blocks prior to the processor's demand, thus hiding a fraction of off-chip memory stalls. Media Streaming exhibits the smallest performance improvement as most of the predicted memory accesses exhibit high memory-level parallelism, and hence the out-of-order processor is able to overlap a significant fraction of off-chip memory stalls. Compared to SMS and VWQ, BuMP improves performance by 3% and 9%, on average, due to higher read coverage.

On the other hand, Full-region hurts performance by 67% (up to 4x for Data Serving), due to oversaturation of memory bandwidth, highlighting the importance of avoiding bulk transfer operations upon accesses to low-density regions.

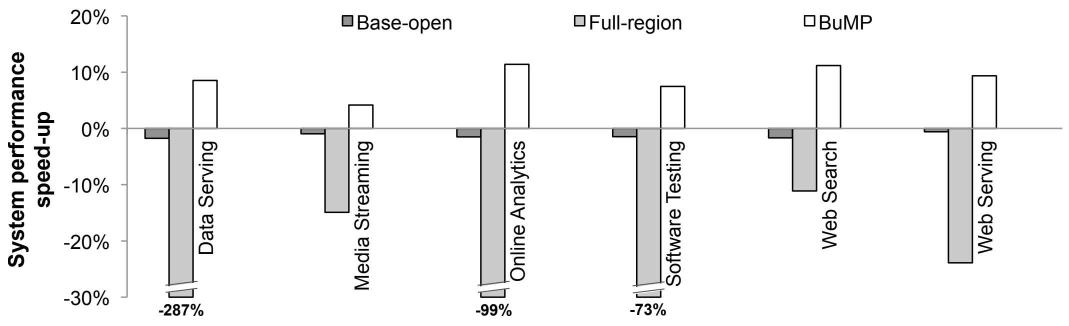


Figure 3.10 – System performance comparison to Base-close.

3.4.5 On-chip bandwidth and energy overheads

We examine BuMP's on-chip bandwidth and energy overheads focusing on the LLC, NOC, and BuMP's structures.

Last-level cache. The LLC traffic increases due to (a) data overfetch (b) bulk read and writeback requests, and (c) extra writeback traffic. Figure 3.11 plots the LLC traffic of BuMP normalized to the baseline system. BuMP increases LLC traffic by 10%, on average. The increase in LLC bandwidth utilization has small impact on system performance as such a small traffic overhead is easily absorbed by the LLC.

The increase in LLC traffic comes at the cost of modest LLC energy overheads (7% on average). As the LLC energy accounts for only 3% of server energy, on-chip LLC energy overheads contribute to less than 0.3% of server energy while LLC power overheads are small (on average, 32mW).

Network-on-chip. The NoC traffic increases due to (a) L1-D requests to the LLC are augmented with the program counter of the associated instruction (PC), (b) LLC data accesses and evictions have to be forwarded to BuMP, (c) BuMP-generated access and writeback requests have to be forwarded to LLC, and (d) data overfetch and extra writeback traffic. As shown in Figure 3.11, network traffic increases by 11%, on average. Similar to traditional server workloads [121], the NoC bandwidth utilization is low. Thus, the extra traffic is absorbed by the NoC, and consequently does not affect the system performance.

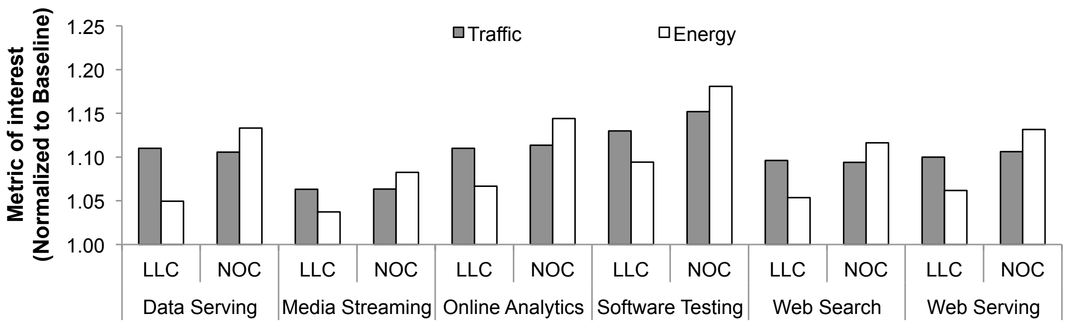


Figure 3.11 – BuMP's on-chip energy overheads.

The increase in NoC traffic comes at the cost of 13% higher NoC energy — half of which is due to the transfer of the PC. As NoC accounts for a small fraction of server energy and power, BuMP's NoC energy and power overheads (8mW) are negligible.

BuMP structures. BuMP introduces multiple structures for region density tracking, summarizing high-density modified regions, and bookkeeping the virtual address of instructions that trigger access to high-density regions. We model these structures using CACTI, and find energy per access to be 2pJ for the region-density tracking tables and 4pJ for the bulk history and dirty region tables. In total, energy consumed to access the structures accounts for 0.1% of server energy. On-chip power consumption is 10mW.

Summary. BuMP on-chip energy overheads account for less than 0.4% of server energy, negligible compared to the registered memory energy gains. On-chip power overheads are smaller than 50mW. Finally, on-chip bandwidth overheads are small, and do not impact performance.

3.4.6 Sensitivity analysis

Sensitivity to threshold and region size. Figure 3.12 illustrates the improvement in memory energy per access for various BuMP configurations. We vary both the region size and threshold for labeling regions as high-density. BuMP with region size of 1KB and threshold of eight cache blocks (noted as 50%) maximizes energy improvements as it targets a higher fraction of DRAM accesses, thus amortizing a higher degree of row activations compared to configurations with

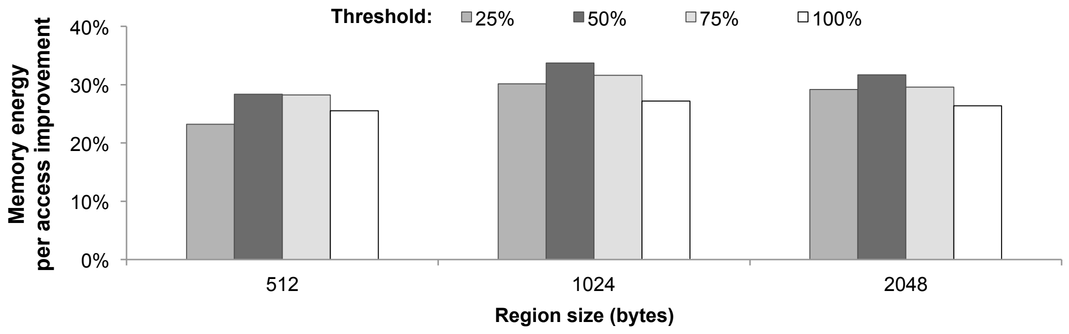


Figure 3.12 – Memory energy efficiency sensitivity to BuMP's threshold and region size.

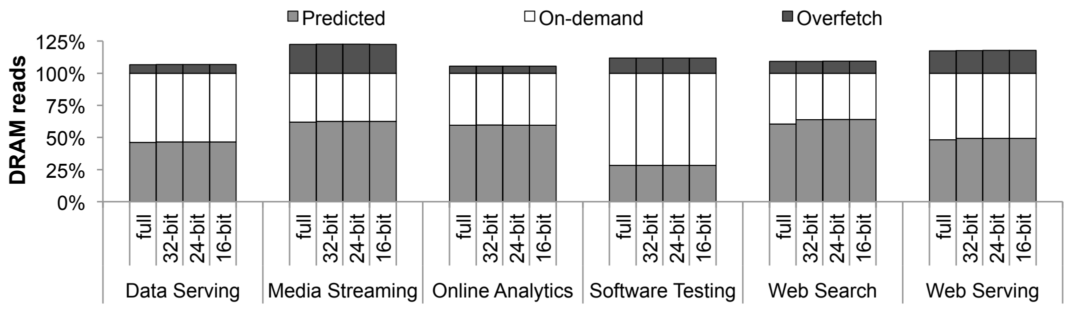


Figure 3.13 – Read prediction accuracy sensitivity to the number of program counter bits used for indexing the bulk history table.

smaller regions while exhibiting lower overfetch rate than configurations with larger regions.

Sensitivity to program counter bits. Figure 3.13 illustrates the read prediction accuracy when varying the number of program counter bits used for indexing the bulk history table. For designs that are noted as 32-bit, 24-bit, and 16-bit, we hash the pair *PC, offset* using the FNV hashing function, and re-size the hash size accordingly with xor-folding. We compare these designs to a design that uses the entire *PC, offset.* Our results demonstrate that there is minimal or no sensitivity to the number of bits used for indexing the bulk history table. This is primarily due to the fact that the number of triggering instructions is much smaller than the number of instructions that can be indexed even with 16 bits. As such, we conclude that an 16-bit FNV hash of the *PC, offset* pair can be used for indexing the bulk history table.

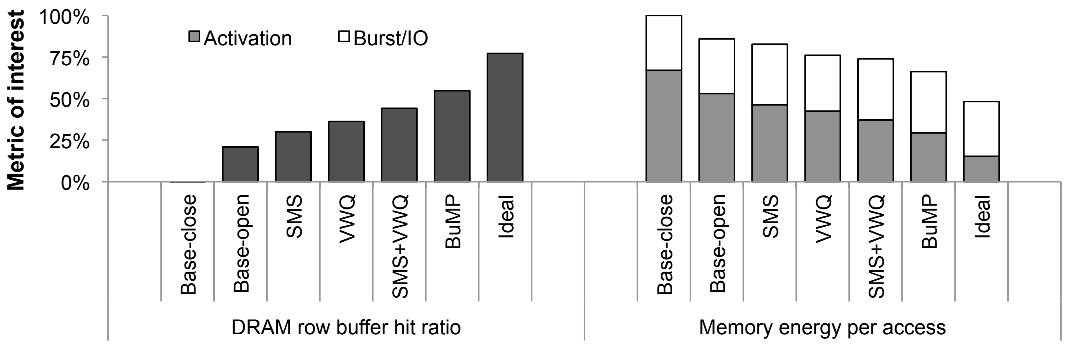


Figure 3.14 – Comparison between BuMP and other systems.

3.4.7 Summary

In Figure 3.14, we plot the DRAM row buffer hit ratio and energy per access of BuMP, averaged across all workloads, compared to the close- and open-mode baselines (both with a stride prefetcher), a system with SMS, a system with VWQ, a system with SMS and VWQ, and an ideal system that exhibits maximal row buffer locality — i.e., it exploits all row buffer locality that exists in an access stream of a thread.

The open-mode baseline system exploits a low degree of DRAM row buffer locality, thus achieving a row buffer hit ratio of 21%. The SMS-enabled system provides a row buffer hit ratio of 30% as it is able to predict irregular access patterns. However, the row buffer locality improvement is small as it targets only load-triggered reads. The VWQ-enabled system improves row buffer hit ratio to 36%, as it exploits a high degree of writeback locality. When SMS and VWQ are combined, row buffer hit ratio increases to 44%.

BuMP improves row buffer hit ratio to 55%, outperforming systems with SMS and/or VWQ. Compared to SMS, BuMP increases row buffer hit ratio by 2x at 3x lower storage requirements. BuMP reduces storage cost as (a) it correlates triggering instructions with coarse-grained regions instead of individual cache blocks, and (b) few instructions trigger accesses to high-density regions. Compared to VWQ, BuMP increases row buffer hit ratio by an additional 19% as VWQ (a) exhibits poor read row buffer locality, and (b) performs lookups for a small number of cache blocks within an open row, thus failing to maximize open-mode gains. Finally, BuMP improves row buffer hit ratio by an additional 11%, compared to the SMS+VWQ system, as it maximizes open-mode gains upon accessing a high-density region.[1]

BuMP achieves high row buffer locality, improving memory energy efficiency by 23-34%. BuMP improves memory energy efficiency by 20% and 13% over SMS- and VWQ-enabled systems. Compared to SMS+VWQ, BuMP reduces memory energy per access by 10%. Finally, BuMP's memory energy efficiency is within 73% of the ideal system.

3.5 Discussion

Memory access scheduling policy. FR-FCFS open-row policy [104] employed by BuMP can hurt system fairness in systems running multi-programmed and parallel applications due to memory behavior variation [63, 92]. Server applications execute almost identical instruction sequences across requests and cores [60], and hence cores exhibit almost same memory behavior. Thus, row buffer locality can be maximized with small impact on system fairness. Nevertheless, scheduling policies that trade off row buffer hit ratio for system fairness [58] can be used complementary with BuMP.

Design scalability. BuMP can scale well with the CMP size, requiring the following modifications for larger configurations: (a) the region density tracking table needs to increase linearly with the number of cores to support a higher degree of interleaving among cores, and (b) the dirty region table needs to increase linearly with the LLC size to support a higher number of active modified regions.

Server virtualization. BuMP is applicable to server applications running in a virtualized environment. In doing so, the bulk history table needs to increase to accommodate the triggering instructions of all active workloads. Even with extreme heterogeneity (one workload per core), the storage requirement of the bulk history table is 72KB in a 16-core system, for total storage requirement of 5KB per core.

[1] BuMP can be combined with VWQ to exploit higher writeback locality by targeting memory writes to non-high-density memory regions.

4 Scalable Off-Chip Memory Systems

With manycore chip multiprocessors improving datacenter efficiency and total cost of ownership [33], the performance [54, 78] and energy-efficiency [120] bottlenecks are shifting to the memory system, which has to sustain the massive bandwidth requirements of manycore processors, and to provide energy-efficient access to vast amounts of memory. Conventional and emerging memory interfaces and technologies, however, fail to match the requirements of scale-out servers as they force a compromise between the conflicting goals of memory capacity, memory bandwidth, and power consumption. In this chapter, we seek to architect a scalable off-chip memory system that provides the required memory bandwidth and capacity for scale-out servers while minimizing the power consumption of the memory system.

4.1 Motivation

In this section, we examine the demands of a scale-out server with regards to the amount of memory bandwidth and memory capacity it requires for efficient execution of scale-out applications. We also study the limitations of conventional and emerging memory interfaces and technologies in the quest for a high-capacity and high-bandwidth memory system.

Table 4.1 – Requirements of one scale-out server.

Year	Processor		Memory System	
	Cores	Bandwidth	Bandwidth	Capacity
2015	96	115 GB/s	288 GB/s	384 GB
2018	180	216 GB/s	540 GB/s	720 GB
2021	320	384 GB/s	960 GB/s	1280 GB

4.1.1 Scale-out server requirements

We quantify the memory bandwidth and capacity requirements of a scale-out server for various manufacturing technologies (denoted by their year) in Table 4.1. Specifically, the table captures the off-chip bandwidth and memory capacity requirements of a server CMP running scale-out applications [78]. The modeled organization is similar to emerging server CMPs, such as Cavium's recently-announced 48-core CMP in the 28nm technology [37]. Our CMP maximizes chip throughput by integrating the maximum number of cores for a given die area and power budget of 250-280 mm^2 and 95-115 Watt, respectively. Our models for future technologies are derived from ITRS projections [49].

We estimate processor's off-chip bandwidth requirements by scaling the per-core bandwidth requirements of scale-out applications with the number of cores available on the chip [78]. We measure per-core bandwidth requirements by simulating a 16-core CMP server (Section 4.4.2 details the configuration). Our analysis illustrates that their per-core bandwidth consumption ranges from 0.4 GB/s to 1.2 GB/s corroborating prior work [54]. Processor's peak bandwidth demands are 115 GB/s (2015), 216 GB/s (2018), and 384 GB/s (2021).

High bandwidth utilization levels can adversely impact the end-to-end memory latency due to high contention on memory resources. Because scale-out applications are sensitive to memory latency (due to their strict response-latency requirements), queuing delays need to be minimized. Thus, system designers over-provision the available memory bandwidth so that utilization levels are kept low (< 40%) [28]. As such, memory systems need to provide available memory bandwidth totaling 288 GB/s (2015), 540 GB/s (2018), and 960 GB/s (2021).

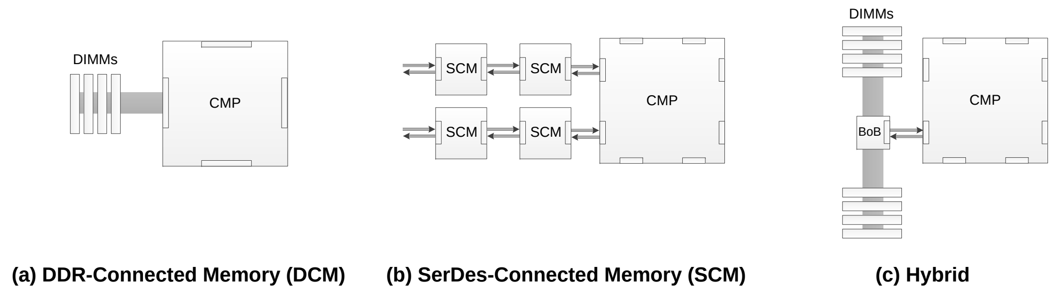

DIMMs
CMP
SCM
SCM
SCM
SCM
CMP
DIMMs
BoB
CMP
(a) DDR-Connected Memory (DCM)
(b) SerDes-Connected Memory (SCM)
(c) Hybrid

Figure 4.1 – Conventional and emerging memory systems.

Optimal amount of memory per server is a function of the application's requirements. For instance, data analytic engines are provisioned with 2-8 GBs per core [19]. Web search engines deploy 64 GBs for 16 cores [100] while our experience with web and media streaming servers shows that cores require 1-2 GBs per core. Hence, we assume 4 GB of memory per core.

Our analysis of scale-out applications illustrates that by 2021 scale-out servers will need to deploy a memory system that provides 1 TB/s of memory bandwidth and over 1 TB of capacity at low power due to strict blade-level power budgets in datacenters.

4.1.2 Conventional and emerging memory systems

In the light of the massive memory requirements of scale-out applications, we examine the ability of conventional and emerging technologies to achieve the conflicting goals of high bandwidth, high capacity, and low power consumption.

DDR-Connected Memory (DCM). In conventional memory systems (Figure 4.1a), DDR (e.g., DDR4) interfaces connect the processor directly to a set of dual-inline memory modules (DIMMs) via a set of parallel buses. However, these interfaces face two main limitations:

- Signal integrity of parallel buses is poor, limiting the I/O data rate at which the data bus can operate, and consequently the bandwidth that one DDR channel can provide (only 25.6 GB/s per channel by 2021 [129]).

- The large number of pins required by one DDR channel (e.g., DDR3's 147 pins) and the low pin-count scalability (only 7% per year [49]) limit the number of memory interfaces and channels that can be integrated on a commodity processor (only six channels by 2021).

The combination of the two limitations above places a trade-off between bandwidth and capacity. To enable high memory capacity in the face of a small number of channels, multiple DIMMs have to be deployed per memory channel. The increased channel load causes degradation in signal integrity, which requires lowering the DRAM bus frequency, thus reducing the available memory bandwidth even further.

SerDes-Connected Memory (SCM). High-speed serial interfaces have the potential to break the bandwidth wall by providing connectivity to high-bandwidth and energy-efficient stacked memory modules via narrow and high-speed channels (Figure 4.1b). SerDes employ point-to-point differential links with excellent signal integrity in contrast to DDR interfaces which employ parallel buses and single-ended signaling. The high signal integrity combined with clock recovery of embedded clocking allows for achieving an order of magnitude higher data rates than DDR. For instance, the recently announced hybrid memory cube (HMC) utilizes two SerDes links clocked at 10 Gbps to provide up to 80 GB/s of bandwidth [95] with the same number of pins as a DDR channel. Furthermore, stacked memory modules offer low latency and exhibit low DRAM energy per access due to reduced wires and smaller row buffers. However, SerDes systems face two main limitations:

- SerDes channels impose high idle power as they are always on; they constantly send idle packets to maintain lane alignment across the channel's lanes [1]. Furthermore, high sleep and wake-up times prevent these channels from going to power-efficient sleep states [1, 3, 46, 99].
- Due to thermal constraints that limit the number of stacked layers, which can be integrated in one module, the amount of memory capacity connected to one SerDes channel is limited to only 4 GB in 2015, 8 GB in 2018, and 16 GB in 2021 [95].

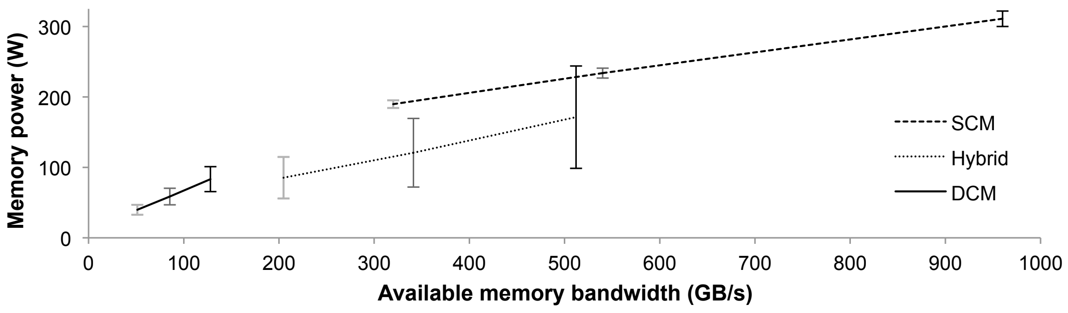


Figure 4.2 – Memory power–bandwidth trade-off. Range bars are used to indicate ranges from idle to peak bandwidth for 2015 (light gray), 2018 (dark gray), and 2021 (black).

As a result, a point-to-point network of these SCM modules is necessary for a high-capacity system, which comes at the cost of high static power consumption due to the use of many SerDes interfaces, and high end-to-end memory latency resulting from the high interconnect latency (Figure 4.1b).

Hybrid. Compared to DCM systems, buffer-on-board (BoB) systems increase available memory bandwidth by using both SerDes and DDR channels. These systems connect the processor to the memory indirectly using a set of SerDes interfaces and intermediate buffers [22, 48, 115] (Figure 4.1c). The buffer provides better signal integrity, and controls multiple DDR channels by forwarding incoming requests and outgoing responses between the DDR channels and processor. As a result, this organization increases memory bandwidth by up to 4x compared to DDR systems.[1] The increase in bandwidth, however, comes at the cost of higher end-to-end latency (2x higher than DCM) and higher power due to serial links and intermediate buffers that increase latency and energy per access.

We quantify the power-bandwidth trade-off for the memory systems above in Figure 4.2. All memory systems are provisioned to maximize available memory capacity so as to match the capacity requirements of scale-out applications in each technology (i.e., 384 GB in 2015, 720 GB in 2018, and 1280 GB in 2021). For each memory system, the x-axis illustrates available memory bandwidth, while y-axis shows the power consumption. Range bars are used to indicate power ranges from idle to peak bandwidth. Range bars with different color represent

[1]Hybrid systems double the number of memory channels for a given pin budget as each SerDes requires half the pins of a DDR channel. Buffers can integrate two DDR channels for a combined improvement of 4x.

different years. Details on the memory system configuration and power modeling can be found in Sections 4.4.1 and 4.4.2.

As illustrated in the figure, the examined memory systems provide various bandwidth-power trade-offs, which follow the linear relationship between bandwidth and power. The DCM system is bandwidth-limited; in particular, it provides up to 50 GB/s (2015), 85 GB/s (2018), and 128 GB/s (2021) of memory bandwidth which accounts only for 15-25% of the maximum requirements of a scale-out server. The SCM system matches the bandwidth requirements of scale-out applications, but it requires 195-320 Watt of power, most of which is static due to the use of many SCM modules. The Hybrid system increases bandwidth as compared to the DCM system by 4x. However, as hybrid memory systems still rely on conventional DDR interfaces, they exhibit high dynamic power consumption. Compared to SCM systems, the Hybrid system exhibits up to 4x lower static power consumption due to DDR's higher energy proportionality. Furthermore, as shown in the figure, Hybrid becomes bandwidth-constrained in late technologies (2018 and 2021) due to poor bus frequency scalability.

4.1.3 Summary

As memory systems are key to efficiency of scale-out applications, they need to (a) provide high memory capacity so as to guarantee fast access to the vast datasets of scale-out applications, (b) sustain high memory bandwidth consumption so as to satisfy the excessive requirements of emerging chip multiprocessors, and (c) minimize their power footprint. However, conventional and emerging memory system designs fail to satisfy these conflicting goals.

4.2 Utilizing SerDes-Connected Memory Modules as a New Level in the Memory Hierarchy

Emerging SCM modules can complement conventional DCM modules in the quest for high-bandwidth and high-capacity memory systems. On the one hand, SCM modules provide high bandwidth, but exhibit high static power per GB. On the other hand, DCM modules face

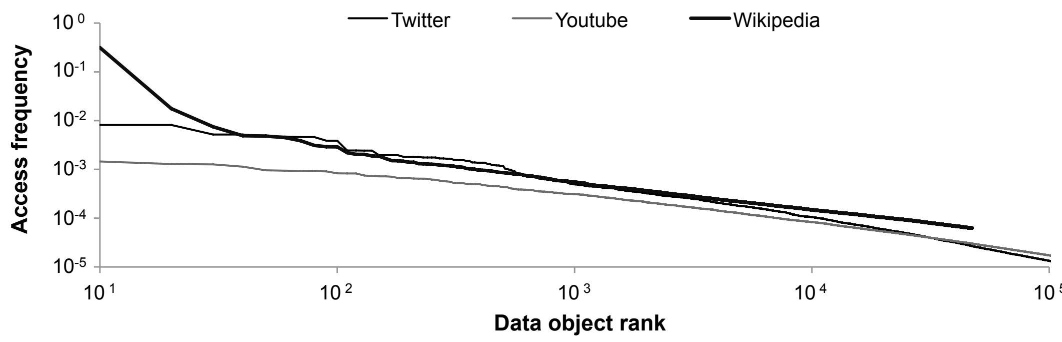


Figure 4.3 – Dataset accesses in web services exhibit power-law distribution. Please note that power-law relationships appear as linear relationships when plotted in log-log scale.

bandwidth scalability limitations and suffer from high energy per access, but exhibit lower static power per GB than SCM modules. Based on the observation that the distribution in memory accesses is skewed, a small number of SCM modules can be employed as a high-bandwidth cache in front of conventional DRAM for serving the frequently-accessed portion of memory. As accesses to the rest of the memory are infrequent, the rest of the memory can be served by under-clocked conventional DCM modules.

Employing SCM as a new level in the hierarchy enables combining high bandwidth and low energy per access of emerging technologies with high capacity of conventional memory technologies at low static power cost. In this section, we examine the application characteristics that enable this new memory system organization. In particular, we shed light on the memory access distributions of scale-out applications (i.e., fraction of memory accesses that go to a memory object) by looking at the characteristics of the dominant types of memory accesses; memory-resident dataset accesses and accesses to dynamically-allocated chunks of memory.

Dataset accesses. A high fraction of memory accesses in scale-out applications are accesses to memory-resident datasets. We use publicly available data to examine the popularity and dataset access distribution of tweets (Twitter), videos (Youtube), and HTML pages (Wikipedia) in Figure 4.3. The figure plots the probability for a dataset item (data object) to be referenced as a function of popularity. As shown in the figure, the dataset access distribution of various web services is highly skewed with a small set of dataset objects (10-20%) contributing to most of the dataset accesses (65-75%). Other examples include:

- **Analytics.** Dataset accesses in analytics are highly skewed as more recent data are more frequently accessed than archived data and analysts often restrict their analysis to a small dataset, which can result in a refined dataset being frequently accessed [20].
- **File servers.** Services hosting a large pool of static pages, such as Wikipedia, follow Zipfian distribution as a set of articles are more popular than the rest [128]. Similar distributions are found in streaming services, such as Youtube [18].
- **Search engines.** Popularity of search query terms follows Zipfian-like distribution in web search engines and private search engines, where a set of events (e.g., celebrity scandals) or items are frequently searched by online users [132].
- **Social networks.** Popular users along with their activity absorb the bulk of user requests. For instance, a small fraction of users and their pictures accounts for most of the viewing, liking, and commenting traffic in picture sharing services, such as Flickr [16]. This Zipfian distribution is found in the broader space of social networks, including Facebook and Twitter [5, 73].

Due to the power-law popularity distribution, dataset accesses in analytic engines [20], data stores [24], object caching systems [73], streaming servers [18], and web search engines [132] exhibit power-law distributions (e.g., Zipfian).

Accesses to dynamically-allocated memory. Scale-out applications exhibit a considerable fraction of accesses to dynamically-allocated chunks of memory with high temporal reuse. For instance:

- **Software caches.** To improve server throughput, server applications utilize software structures to cache a set of hot objects (e.g., rows and pages in data stores and compiled script code in web servers) or to speed up object lookups by caching their exact position in the memory-resident dataset (e.g., key caches in data stores). As these data structures host data/metadata relevant to the dataset, the distribution of memory accesses going to them will follow the power-law distribution of dataset accesses.

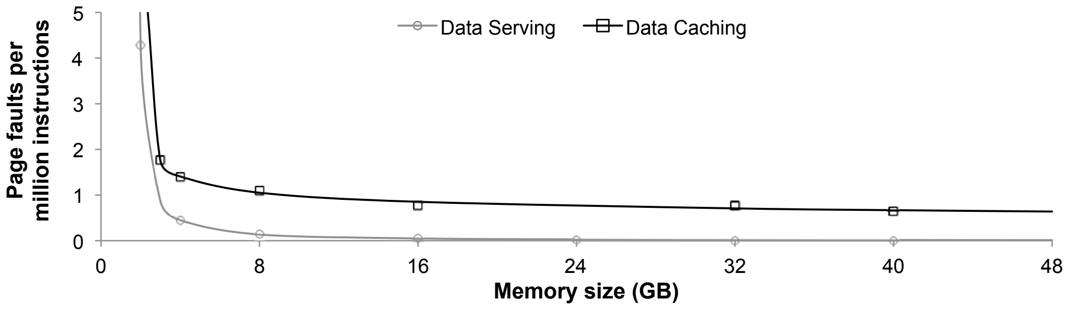


Figure 4.4 – Page fault rate for various memory sizes. Contiguous lines denotes the x-shifted power-law fitting curves.

- **Active connections.** Server applications employ various data structures per connection. While each connection allocates only a few hundreds of KBs, the large number of concurrent requests results in a footprint of a few GBs that dwarfs the capacity of on-chip caches. Examples include: (a) buffers in streaming servers keeping media packets, and (b) statistics for tracking quality of service of streaming connections. The reuse of these data structures at the memory level is high as they are accessed multiple times over the duration of an active connection.

- **Partitioning and hash tables.** Analytic engines rely on partitioning or dynamically built hash tables to improve spatio-temporal locality of dataset accesses upon iterative computations in graph analysis [107] and join operations in relational database systems.

The skew in object popularity and temporal reuse of dynamically allocated memory is expected to be mirrored in the memory access distribution. To confirm this, we examine the memory access distribution of scale-out applications running on a 12-core Xeon server. We use performance counters to measure the application's page fault rate while varying the available memory size from 2 GBs to 48 GBs. The page fault rate is indicative of the memory footprint of the application. We use the following scale-out applications:

Data Caching. We run Memcached 1.4.21 with a 60 GB Twitter dataset and object caching pool of 40 GB. Server load is generated following a real-world distribution (Zipfian-like) with 80:20 get to set request ratio.

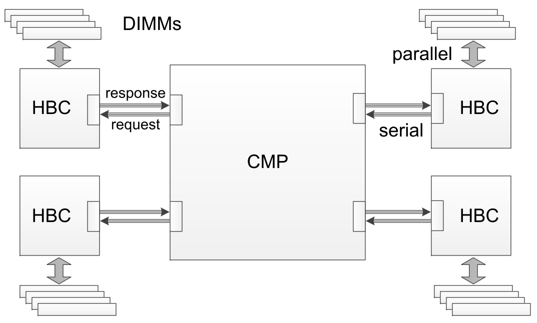
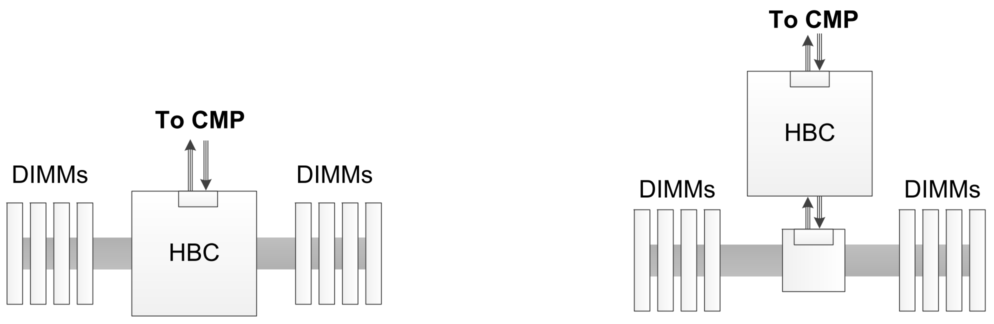


Figure 4.5 – MeSSOS overview.

Data Serving. We run Cassandra 1.2.6 with a 32 GB Yahoo! Cloud Serving Benchmark dataset. Server load is generated following a Zipfian distribution with 95:5 read to write request ratio [24]. Cassandra is configured with 8 GB of heap and 1.2 GB for garbage collection.

We plot the page fault rate of the examined workloads in Figure 4.4. The markers denote actual measurements while contiguous lines illustrate the fitting x-shifted power-law curve. The figure shows that memory accesses are skewed, such that the majority of them go to a small portion of memory, and hence are amenable to caching. Page faults are rare and stable once the memory captures 10% of the dataset (4 GB for Data Serving with a 40 GB dataset, 8 GB for Data Caching with a 60 GB dataset). Our results corroborate earlier work [41, 54, 79] showing the power-law relationship between cache capacity and miss ratio in multi-MB caches.

Die-stacked DRAM caches [53, 54, 55, 75, 76, 101, 130] can provide vast amounts of memory bandwidth at low power. However, thermal constraints in limit the number of DRAM stacks that can be integrated on top of the processor [10, 23], limiting die-stacked cache capacity to several hundreds of MBs. On a similar note, high-bandwidth caches implemented in on-board buffers [115] have been proposed to bridge the ever-increasing gap between processor requirements and DDR interfaces. Our findings show that the hot portion of memory of scale-out applications, exceeds the capacity of die-stacked and off-chip caches by an order of magnitude.



(a) MeSSOS-Direct

(b) MeSSOS-Buffer

Figure 4.6 – HBC-Main Memory interface.

4.3 MeSSOS: A Memory System Organization for Scale-Out Servers

Scale-out servers impose high bandwidth and capacity demands on the memory system. We present MeSSOS, our technology-scalable memory system organization for scale-out servers, which builds on the insight that SerDes-connected memory modules can be employed as a high-bandwidth cache (HBC) in front of conventional DRAM. In doing so, MeSSOS minimizes the number of SerDes-connected memory modules, and hence enables high bandwidth at low static power cost. As most of memory accesses are served by the HBC, accesses to the DIMMs are rare, and hence high capacity is achieved at low power cost with under-clocked DIMMs.

Figure 4.5 shows the overview of MeSSOS, illustrating its integration with the processor. In the rest of the section, we examine the interface between HBC and main memory, the HBC organization, and the processor-HBC connectivity.

4.3.1 HBC-main memory interface

The SerDes-connected memory modules not only provide the functionality of a cache, but also act as the bridge between processor and main memory. Memory requests that miss in the high-bandwidth cache have to be forwarded to main memory. We examine two possible options (shown in Figure 4.6) for providing connectivity between an HBC module and multiple main memory modules:

- **MeSSOS-Direct.** Opting for integration and higher efficiency, the HBC's logic layer integrates a DDR controller to directly control the DDR channels (Figure 4.6a). This approach requires implementing both DDR protocol and PHYs on the logic die, providing less flexibility and requiring a large number of pins (i.e., 294 for two DDR3 channels).

- **MeSSOS-Buffer.** Opting for flexibility at the cost of lower efficiency, an HBC module connects to the DIMMs via an intermediate buffer (Figure 4.6b). This approach resembles a Hybrid system (introduced in Section 4.1.2). The buffer integrates a simple controller to communicate with the HBC via a SerDes interface, and a DDR controller to control the DDR channels. This approach, however, imposes significant energy and latency overheads compared to MeSSOS-Direct. In particular, it requires an extra pair of SerDes interfaces, which comes at the cost of static power, and increases energy/latency per access due to the extra SerDes link and intermediate buffer.

4.3.2 High-bandwidth cache organization

An important component of MeSSOS is the high-bandwidth cache (HBC), which utilizes multiple memory modules to serve the bulk of memory accesses. To avoid communication between HBC modules and to minimize the number of SerDes links, memory addresses are statically interleaved among the HBC modules. This distributed organization enables scaling capacity and bandwidth linearly with the number of HBC modules, thereby allowing for a technology-scalable architecture.

Cache architecture. We organize the HBC as a page-based cache to exploit coarse-grained memory accesses found in scale-out applications [120]. This organization improves system performance by boosting the cache hit ratio, and reduces dynamic DRAM energy by minimizing the number of DRAM row activates, the most energy-consuming operation in conventional DRAM architecture [54, 120]. Based on page-size sensitivity analysis, we find that a page size of around 1 KB balances the opportunity for targeting a large fraction of memory accesses with limited tolerance for overfetch. The placement of huge tag arrays and tag-lookup latency represent the biggest challenges in large-capacity cache designs. Recent die-stacked DRAM cache proposals [53, 101] provide scalable solutions to these problems by integrating the tags into DRAM while hiding the tag-lookup latency. We leverage these ideas [53, 101] to embed the tags in DRAM next to the data and access both in parallel. The cache is 4-way set-associative and features a way predictor, located in the logic die of the HBC module.

4.3.3 Processor-HBC interface

To enable connectivity between the processor and HBC, MeSSOS employs point-to-point serial links with multiple request and response lanes. Both processor and HBC modules implement simple controllers to orchestrate communication between the processor and the memory system (Figure 4.5). On the processor side, the controllers serialize outgoing requests into packets, before routing them to the HBC module, and deserialize incoming data and forwards them to the last-level cache. On the HBC side, the controller deserializes incoming memory requests and forwards them to the local cache controller, and serializes outgoing data into packets and forwards them to the processor. Given their simple functionality, these controllers consist of only a pair of queues to buffer incoming and outgoing packets, and a SerDes interface.

4.4 Experimental Methodology

We compare the performance and energy efficiency of MeSSOS to various systems using a combination of cycle-accurate simulations, analytic models, and technology studies.

4.4.1 Scale-out server organization

We model chips with an area of 250-280 mm^2, and a power budget of 95-115 Watt. We use the scale-out processor methodology to derive the optimal ratio between core count and cache size in each technology [78]. Core micro-architecture is modeled after Cortex-A15, a three-way out-of-order core.

Table 4.2 – Systems configuration parameters.

System	2015	2018	2021
CMP	96 cores	180 cores	320 cores
LLC	24 MB	45 MB	80 MB
Memory	384 GB	720 GB	1280 GB
DCM	4 DDR-1600	5 DDR-2133	6 DDR-2667
SCM	8x10 Gbps	10x15 Gbps	12x20 Gbps
Hybrid	8x10 Gbps, 16 DDR-1600	10x15 Gbps, 20 DDR-2133	12x20 Gbps, 24 DDR-2667
Die-stacked	Cache: 1 GB, 4 DDR-1600	Cache: 2 GB, 5 DDR-2133	Cache: 4 GB, 6 DDR-2667
MeSSOS	8x10 Gbps, 16 DDR-1067 HBC: 8x4GB	10x15 Gbps, 20 DDR-1067 HBC: 10x8GB	12x20 Gbps, 24 DDR-1067 HBC: 12x16GB

Table 4.2 summarizes the core count, the LLC size, and the memory interfaces for the baseline memory systems, and MeSSOS for three different technology generations. For the SCM system, we assume a tree network topology to reduce the number of hops in the point-to-point memory network (average and maximum number of network hops is 3 and 4). We also evaluate the effectiveness of state-of-the-art die-stacked caches [101] in bridging the bandwidth gap between processors and memory. Due to thermal constraints, the amount of stacked DRAM that can be integrated is limited to several hundreds of MBs (2015), and to a few GBs (2018, 2021). Due to the limited capacity, we organize the die-stacked cache as a block-based [101], thus using its capacity more effectively and minimizing off-chip BW consumption.

4.4.2 Performance and energy evaluation

System performance. We evaluate system performance through full-system simulation using Flexus [125]. Flexus models the SPARC v9 ISA and runs unmodified operating systems and applications. Flexus extends the Simics functional simulator with timing models of out-of-order cores, caches, on-chip protocol controllers and interconnect, and DRAM. DRAM is modeled by integrating DRAMSim2 [106] directly into Flexus. The parameters for DRAMSim2 are based on commercial DDR3 device specifications [87].

Table 4.3 – System configuration for cycle-accurate simulations.

Parameter	Value
CMP	16 cores, 2.5GHz, 3-way OoO, 60-entry ROB and LSQ
L1-I/D	64KB, 2-way, 64B blocks, 2-cycle load-to-use, 10 MSHRs
LLC	Unified, 4MB, 16-way, 64B blocks, 8 banks, 8-cycle hit latency
NOC	16x8 crossbar, 5 cycles
Memory	HBC: 512 MB-16 GB; DRAM: 16-128 GB, Off-chip links: 15ns/30ns (parallel/serial)

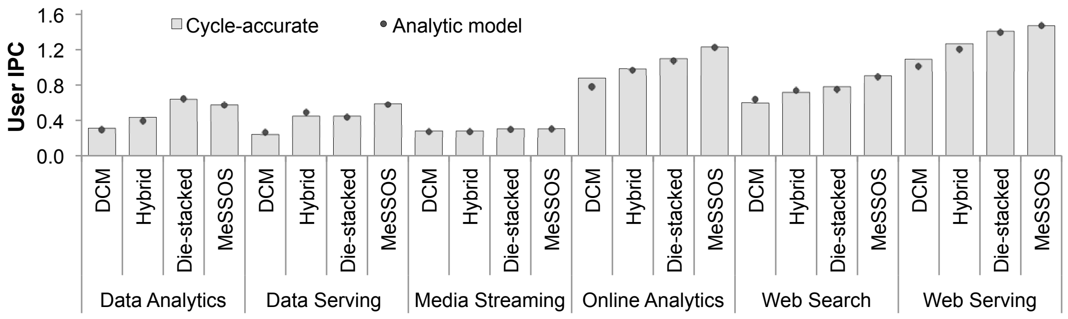


Figure 4.7 – Validation of analytic models against cycle-accurate simulations.

Our analysis is based on a wide range of scale-out workloads. The examined workloads, taken from CloudSuite 2.0 [21, 28], include Data Analytics, Data Serving, Media Streaming, Web Search, and Web Serving. We evaluate online analytics running a mix of TPC-H queries on a modern column-store database engine, MonetDB [61].

We run cycle-accurate simulations using the SMARTS sampling methodology [131]. Our samples are drawn over an interval of 10-30 seconds of simulated time. For each measurement, we launch simulations from checkpoints with warmed caches and branch predictors, and run 800K cycles (2M cycles for Data Serving) to achieve a steady state of detailed cycle-accurate simulation prior to collecting measurements for the subsequent 400K cycles. For performance, we measure User IPC, defined as the ratio of the number of application (user) instructions committed to the total number of cycles (including cycles spent executing operating system code); this metric has been shown to reflect system throughput [125].

As simulating processors with hundreds of cores and memory capacity of hundreds of GBs would be prohibitively slow, we augment our simulation-based studies with an analytic model.

Table 4.4 – System power model in 2015.

Parameter	Value
Core	Peak power: 770mW
LLC	Read/Write energy: .63nJ/.70nJ Leakage power: 750mW per 4MB
Memory controller	Front-end engine: 0.24mW/Gbps Transaction engine: 1.37 mW/Gbps Physical interface: 1.95 mW/Gbps
DDR-1600 (per 4GB rank and 64-byte transfer)	Idle power: 377mW Activation energy: 19nJ, Read/Write energy: 5.2nJ/5.4nJ I/O energy (Read/RRead): 0.6nJ/1.7nJ, I/O energy (Write/RWrite): 2.1nJ
SCM (4GB)	DRAM array per 64-byte access: 1.9nJ, Logic + SerDes: 2.34 + 4.5mW/Gbps
Buffers	SerDes: 4.5mW/Gbps, DDR PHY: 1.95mW/Gbps Hybrid Logic: 0.24mW/Gbps, MeSSOS Logic: 1.61 mW/Gbps

Our model extends the classical average memory access time analysis [39] to predict per-core performance for a given off-chip memory system; the model is parameterized by simulations results, including core performance, on-chip cache miss rates, and interconnect delay. For off-chip access latency, we include link latency, memory core latency, and queuing delays. To model queuing delays in the examined workloads, we run a set of cycle-accurate simulations to measure the memory access latency for various bandwidth utilization levels for each workload.

We validate our analytic models against cycle-accurate full-system simulations of a scaled-down CMP. Table 4.3 illustrates the configuration of our 16-core CMP simulated system. We scale down the LLC size, memory capacity, and off-chip bandwidth resources (from Table 4.2) accordingly to account for smaller CMP (e.g., 16 cores instead of 96 cores). Figure 4.7 shows that our analytic models (denoted as circle markers) achieve high performance prediction accuracy (average error of 5%).

Power and energy modeling framework. We use energy consumed per instruction as our energy-efficiency metric. We develop a custom energy modeling framework to include various system components, such as cores, network-on-chip (NOC), caches, memory controllers, and memory. Our framework draws on several specialized tools to maximize fidelity through detailed parameter control. Table 4.4 summarizes our system power and energy models.

We estimate core power by scaling published measurements of power on a high-IPC workload by the ratio of the actual workload IPC and the reference IPC, assuming that half of the power scales with the workload IPC and the rest is static including leakage [4, 71]. We use CACTI's models to obtain LLC energy estimates, and we account for advanced LLC leakage reduction techniques [71, 90]. We use custom models to estimate energy consumed by on-chip network links and switch fabrics derived by public sources [7, 49]. We measure memory controller's power using McPAT [72].

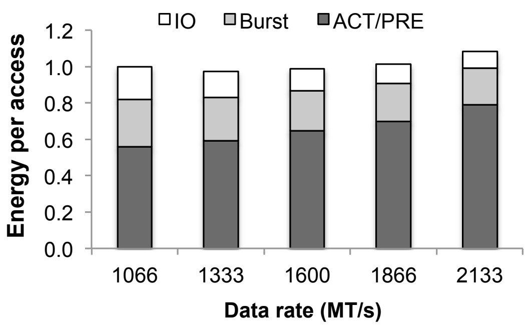
We estimate DDR background power and energy per operation based on Micron models and data sheets [86, 88]. Per JEDEC's specifications and Intel's estimations [6], we anticipate a voltage reduction from 1.5 (DDR3) to 1.2 (DDR4), and various DDR4 optimizations that halve termination energy and reduce refresh power by 20%.

We quantify the energy consumption of a SerDes-connected memory module using energy measurements reported for the recently announced Hybrid Memory Cube (HMC) [51]. Compared to conventional DRAM, HMC reduces energy per access by leveraging through-silicon-via technology [51].

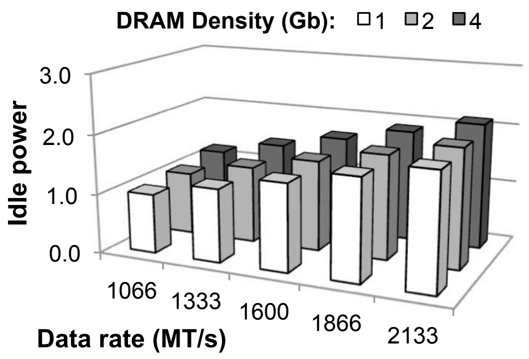
We model buffers for the Hybrid and MeSSOS-Buffer systems to include the power consumption of the SerDes interface (mostly static) and the DDR physical interfaces. For Hybrid, we account for the buffering of incoming requests and outgoing data similar to an integrated memory controller's front-end engine. For MeSSOS-Buffer, we include the memory controller's logic, which buffers incoming requests and outgoing data, and reschedules and translates requests to DDR commands.

4.4.3 Projection to future technologies

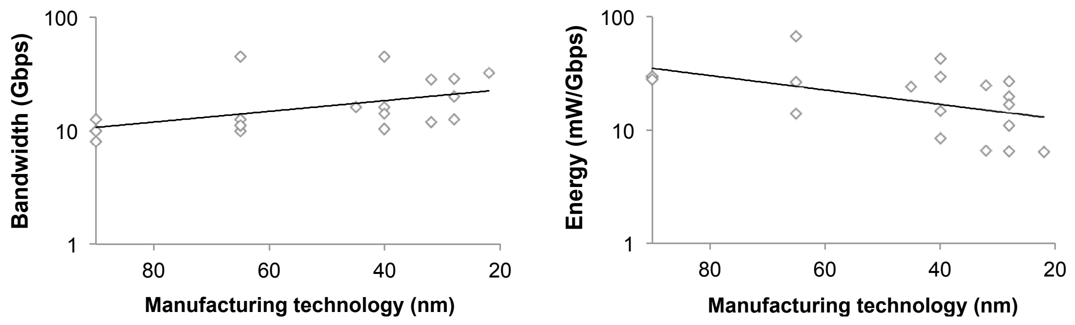
To understand the effect of technology scaling on the different memory system designs, we also model our systems in 2018 and in 2021. Per ITRS estimates, processor supply voltages will scale from 0.85V (2015) to 0.8V (2018) and 0.75V (2021).



(a) DDR energy per access as a function of the data rate



(b) DDR idle power consumption as a function of data rate and DRAM density

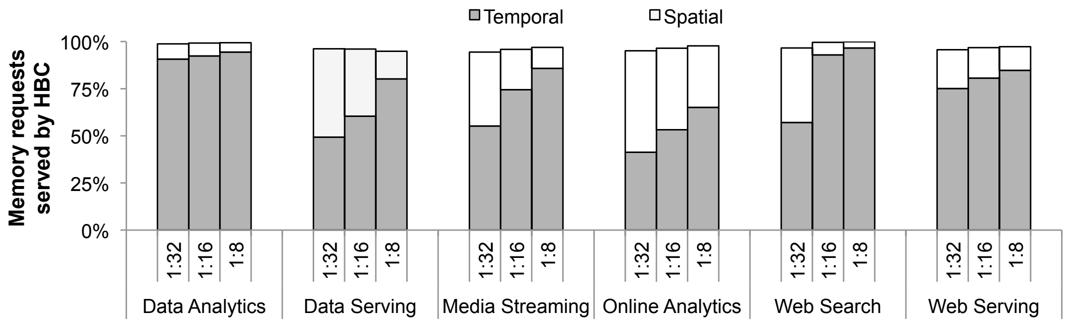


(c) SerDes bandwidth and energy scalability across manufacturing technologies

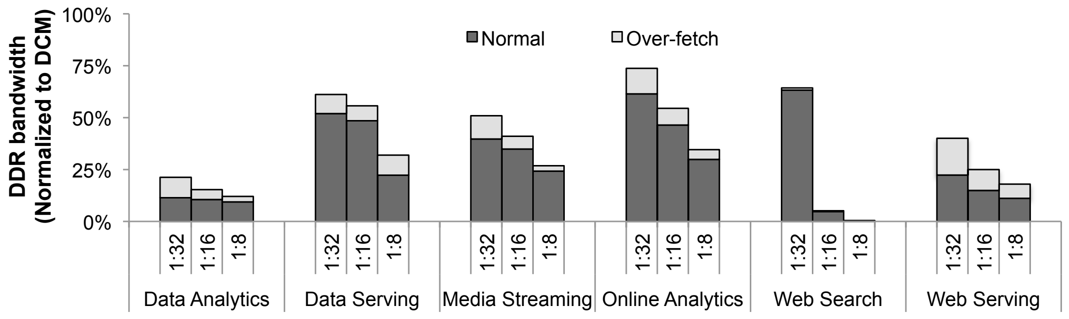
Figure 4.8 – Impact of technology scaling on memory technology and memory interfaces.

We examine the impact of data rate and memory density on DDR energy consumption. We compute the per-access energy consumption of DDR3 devices for different data rates based on Micron's data sheets [86]. Figure 4.8a illustrates that DDR energy remains almost constant, but the breakdown differs. We examine the background power of DDR devices for various data rates and memory densities. Static power consumed by DRAM core (leakage) and DLL (active mode) scales linearly with data rate while refresh power scales linearly with memory density. Figure 4.8b plots this relationship, which draws on Micron's data sheets [85, 86, 87].

Finally, we study the impact of manufacturing technology on power consumption of SerDes interfaces. Figure 4.8c plots our bandwidth and energy analysis based on published measurements of various SerDes interfaces in different technologies [8, 13, 14, 31, 32, 43, 44, 50, 56, 64, 91, 98, 103, 108, 111, 114]. Our analysis demonstrates that bandwidth and energy scale by 20% and 27% per technology node, respectively.



(a)



(b)

Figure 4.9 – MeSSOS effectiveness for various HBC:Memory capacity ratios.

4.5 Evaluation

We compare MeSSOS to conventional and emerging memory systems. First, we examine the performance of MeSSOS and its implications on system performance and system energy efficiency in 2015. Finally, we examine the effect of technology scaling on MeSSOS.

4.5.1 Baseline study

We begin our study with a 96-core CMP in the 22 nm technology. Figure 4.9a plots the fraction of memory requests that are served by the high-bandwidth cache (HBC) for various cache-to-memory capacity ratios. The figure demonstrates the ability of MeSSOS to serve the majority of those using its HBC (>95%) as it exploits temporal locality arising from the skewed memory access distribution and temporal reuse (gray bar) as well as spatial locality arising from operations on coarse-grained data and software objects (white bar).

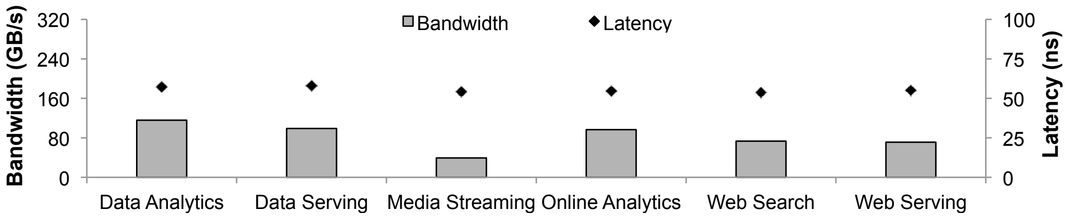


Figure 4.10 – Bandwidth and memory latency in MeSSOS.

Figure 4.9b illustrates the DDR bandwidth consumption compared to the baseline systems without a cache, resulting from unfiltered memory accesses. As expected, DDR bandwidth savings increase with bigger caches. The HBC is able to absorb 65-95% of memory traffic for a cache size of 12.5% of the memory size (1:8), thereby reducing DDR bandwidth consumption by 3-20x. This result corroborates our observation that memory access distribution in scale-out applications is skewed. The light gray bars illustrate the extra traffic generated due to coarse-grained transfers between the HBC and the DIMMs. The extra traffic is small (7% on average) as most of the accesses in scale-out applications are coarse-grained (Section 3.2).

Memory performance. Figure 4.10 plots the end-to-end memory latency and the off-chip bandwidth between the processor and the HBC. MeSSOS's off-chip bandwidth consumption ranges from 42 GB/s (Media Streaming) to 114 GB/s (Data Analytics) for a 96-core CMP, which corresponds to modest off-chip bandwidth utilization levels (14-38%). Due to low link and memory bandwidth utilization levels and the high hit ratio in the HBC, MeSSOS achieves low queuing times and fully leverages the low core latency of emerging SerDes-connected memory modules. In doing so, MeSSOS provides both high bandwidth and low latency.

System performance. Figure 4.11 compares the system performance of MeSSOS against the DDR-Connected Memory (DCM), the SerDes-Connected Memory (SCM), the Hybrid, and the Die-stacked systems, configured as in Table 4.2. For MeSSOS, we plot only MeSSOS-Direct as MeSSOS-Buffer performs similarly, due to high hit ratio in the HBC, which hides the extra latency of the serial link and intermediate buffer found in MeSSOS-Buffer.

Because DCM provides insufficient memory bandwidth, its system performance is significantly hurt. The Hybrid and SCM systems improve system performance over DCM by 49%

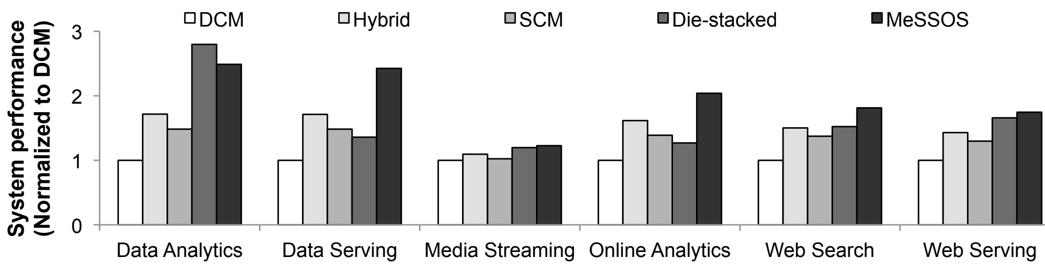


Figure 4.11 – System performance of various memory systems.

Table 4.5 – Memory requests served by the die-stacked cache in the Die-Stacked system.

Workload	Value	Workload	Value	Workload	Value
Data Analytics	83%	Data Serving	39%	Media Streaming	50%
Online Analytics	13%	Web Search	37%	Web Serving	55%

and 33%, respectively, as the they provide sufficient bandwidth to the processor. However, the increase in bandwidth comes at the cost of higher end-to-end memory latency. The Hybrid system adds an extra 40 ns due to the serial link and the intermediate buffer while the SCM system requires a point-to-point memory network, which adds a latency of 35 ns per network hop (serial link and pass-through logic). As a result, Hybrid outperforms SCM by 12%. MeSSOS outperforms all these systems due to its ability to provide high bandwidth at low latency (Figure 4.10). Compared to the DCM system, MeSSOS improves system performance by 2x on average. MeSSOS outperforms Hybrid and SCM by 28% and 43%, respectively.

MeSSOS outperforms Die-stacked by 23% due to lower off-chip bandwidth pressure, resulting from its greater cache capacity which filters a higher fraction of DDR accesses. On average, MeSSOS filters 84% of DDR accesses as compared to 45% of the Die-stacked system (Table 4.5). For Data Serving and Online Analytics, MeSSOS outperforms Die-stacked by 81% and 61%, as Die-stacked is bandwidth-constrained due to is inability to achieve significant off-chip bandwidth reduction (filters only 38% and 13% of DDR accesses, respectively). One exception is Data Analytics where dataset accesses are highly skewed (Figure 4.9a), and hence Die-stacked achieves high hit ratio (83%) and outperforms MeSSOS by 12% due to lower cache access latency.

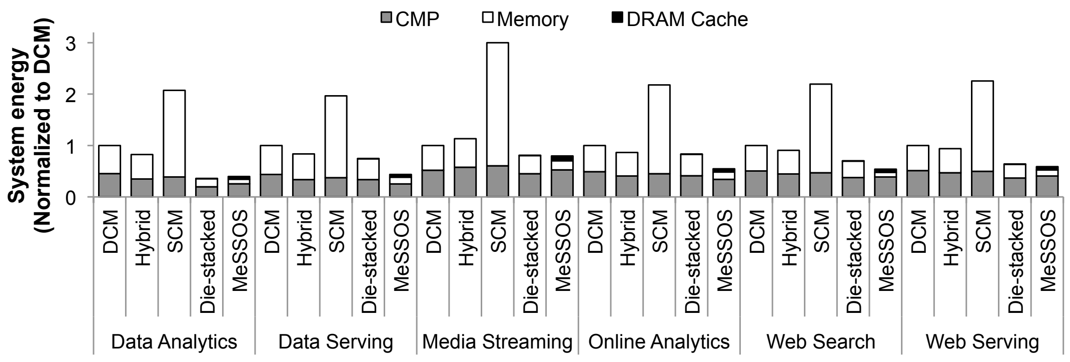


Figure 4.12 – System energy broken into processor, memory, and cache components.

System energy. We examine the energy-efficiency of MeSSOS compared to the baselines systems in Figure 4.12. As the figure shows, Hybrid reduces system energy consumption per operation by 12% compared to DCM, primarily due to system performance gains. SCM increases system energy per operation by 2.3x compared to DCM. Although SCM provides sufficient bandwidth to the processor, leading to higher system throughput, it requires a large network of SCM modules, which comes at the cost of high power.

MeSSOS improves system energy per operation by 1.9x, 1.7x, and 4.3x compared to the DCM, Hybrid, and SCM systems, respectively. MeSSOS utilizes just a few Serdes-connected memory modules to break the bandwidth wall at relatively low power cost. As MeSSOS serves the bulk of memory accesses from its high-bandwidth cache, it exploits the low-energy access of stacked memory modules, thereby reducing memory energy consumption significantly. Furthermore, MeSSOS enforces coarse-grained data movement between DRAM and the HBC, thus amortizing the energy-intensive row activates of conventional DRAM. Compared to Die-stacked, MeSSOS improves system energy per operation by 1.23x due to a combination of higher cache hit ratio and lower off-chip bandwidth consumption, which lead to smaller DDR energy footprint.

MeSSOS-Buffer (not plotted) achieves lower energy efficiency compared to MeSSOS-Direct as it requires an extra pair of SerDes interfaces to connect the HBC module to the intermediate buffer. On average across our workloads, MeSSOS-Buffer increases energy per operation by 19% over MeSSOS-Direct.

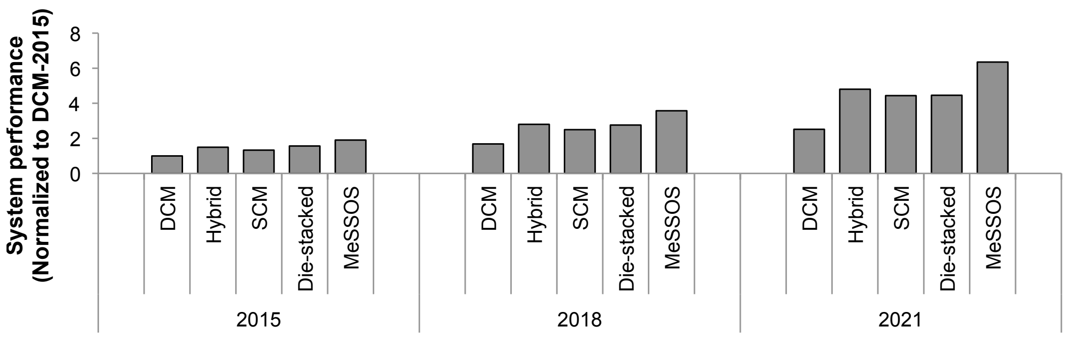


Figure 4.13 – System performance for various technologies normalized to DCM (2015).

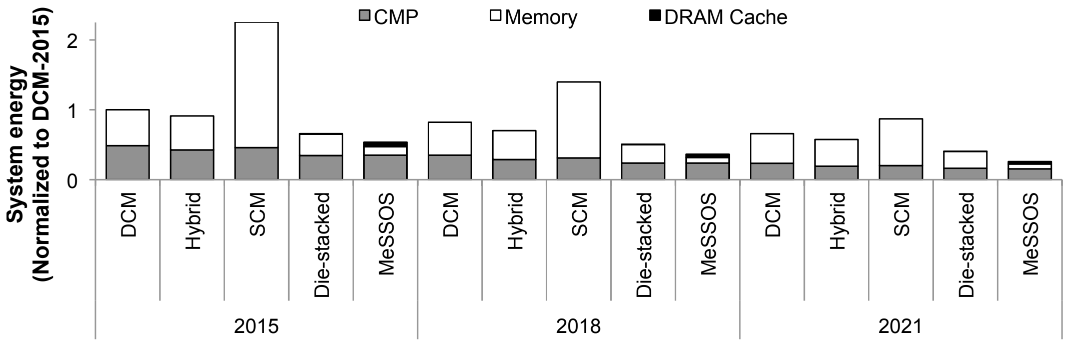


Figure 4.14 – System energy consumption for various technologies normalized to DCM (2015).

4.5.2 Projection to future technologies

We study the effect of technology scaling on MeSSOS in 2018 and 2021. Figures 4.13 and 4.14 plot the system performance and system energy consumption per operation averaged across the applications and normalized to DCM in 2015. MeSSOS leverages the abundant bandwidth provided by the SerDes technology, increasing system throughput almost linearly with the number of cores. This near-perfect scalability results in an increase in system throughput of 3.7x and 6.6x in 2018 and 2021 technologies, respectively, compared to DCM (2015).

Due to poor scalability of DDR interfaces and increase in processor throughput, the bandwidth gap between DDR-based systems and the processor is increasing rapidly. As a result, MeSSOS's performance improvement over DCM and Die-stacked increases across technologies. In particular, MeSSOS improves system performance by 2.3x (2018) and 2.7x (2021) over DCM, and by 30% and 43% over Die-stacked.

Table 4.6 – Server throughput when adding a die-stacked DRAM cache to a MeSSOS-enabled system (2015).

Workload	Value	Workload	Value	Workload	Value
Data Analytics	19%	Data Serving	8%	Media Streaming	4%
Online Analytics	2%	Web Search	4%	Web Serving	6%

Regarding system energy consumption, the energy footprint of a DDR module increases over technologies, primarily due to an increase in static power of their active interfaces. Because MeSSOS employs under-clocked DDR modules, its total energy footprint increases only by a small amount over technologies. As a result, MeSSOS improves system energy per operation by 1.7x in 2015, 2x in 2018, and 2.6x in 2021 compared to DCM and Hybrid, and by 23% in 2015, 40% in 2018, and 60% in 2021 compared to Die-stacked. Compared to SCM, MeSSOS improves system energy per operation by 4-4.4x by using fewer SerDes interfaces.

4.5.3 MeSSOS with a die-stacked DRAM cache

Die-stacked DRAM caches have been studied to mitigate the gap between available DDR bandwidth and bandwidth requirements of manycore processors. In Section 4.5.1, we demonstrated that a heterogeneous off-chip memory system (MeSSOS), which employs emerging SerDes-connected memory modules as a cache in front of conventional DRAM modules, can satisfy the required off-chip bandwidth of scale-out servers. In this subsection, we quantify the performance benefits of adding a die-stacked DRAM cache to a MeSSOS-enabled system.

Table 4.6 illustrates the improvement in server throughput for a 96-core MeSSOS-enabled system with a die-stacked DRAM cache. Across our scale-out workloads, the server throughput improves only by 7% on average, and up to 19% for Data Analytics. The impact on system performance is small as (a) off-chip bandwidth in a MeSSOS-enabled system is abundant, and hence adding a die-stacked cache provides negligible bandwidth advantages, and (b) temporal reuse in the die-stacked cache is low (Table 4.5) due to disparity between die-stacked cache capacity and hot dataset size, and hence improvements in memory access latency are small.

4.6 Discussion

Impact of a new level in the memory hierarchy on QoS. MeSSOS is effective in capturing the hot portion of dataset in its high-bandwidth and low-latency cache. However, this extra level in the memory hierarchy can potentially increase the response latency of infrequent queries that access cold portions of the dataset, and hence miss in the cache.

Scale-out applications access pointer-intensive indexing structures to retrieve coarse-grained data objects. Because these objects span several kilobytes, coarse-grained access patterns dominate memory traffic in scale-out applications [120]. As shown in Figure 4.9a, MeSSOS achieves high hit ratios due to its page-based organization which allows for exploiting the coarse granularity at which scale-out applications access their datasets. Thus, MeSSOS amortizes the extra latency of infrequent misses (e.g., pointer chasing to retrieve a cold object) over multiple hits in the cache (e.g., coarse-grained operation on the retrieved object).

Impact on datacenter's total cost of ownership (TCO). Large-scale services distribute and replicate their datasets across servers to ensure in-memory processing and to meet their throughput requirements. Increasing computing and memory density of one server leads to TCO savings, as datacenters require deploying fewer servers for the same throughput and dataset size. Compared to a conventional scale-out server, MeSSOS improves throughput by 2.5x, resulting in 2.5x fewer servers.

Similarly, Cisco, IBM, and Intel sell extended memory technology that relies on multiple on-board chips specifically with the aim of reducing overall system cost (e.g., Cisco's UCS-5100 series). In contrast to buffer-on-board systems, MeSSOS does not require additional on-board chips as the functionality of buffer chips is implemented on the logic layer of the HBC modules.

Sensitivity to LLC size. We employed an LLC of 4 MB per 16 cores as this core-to-cache ratio maximizes throughput for a given die size [78]. Our analysis with twice the LLC, shows that bandwidth requirements reduce by 1.17x. The reduction in bandwidth requirements, however, comes at the cost of silicon area (equal to 4 cores), and consequently lower throughput (25%).

5 Related Work

5.1 On-Chip Interconnects

The MIT RAW architecture [118] uses four symmetric NoCs (two static networks and two dynamic) which use packet-switched flow control. The Tilera chip [126] extends the MIT RAW architecture and uses five identical wormhole-routed networks to isolate: (a) communication to different sub-systems, (b) memory traffic, and (c) user-specified traffic. In contrast, CCNoC, introduced in this thesis, requires only a pair of networks to divide packets at the protocol level. In addition, the networks in CCNoC are asymmetric, yielding greater efficiency and performance through specialization.

Prior work split network traffic into two heterogeneous or homogeneous networks to improve performance and power efficiency [7]. The former splits traffic based on message size and the latter strives for load balance across the two networks. Using simple messaging protocols, the authors conclude that homogeneous designs are preferred to heterogeneous. In this thesis, we show that, for cache-coherent CMPs, a heterogeneous design which splits network traffic based on message class leads to better performance and power efficiency than a homogeneous design, as it requires fewer VCs for deadlock avoidance. In addition, we show that a design which splits network traffic based on message class rather than message size leads to better performance and power efficiency, as it achieves better load balance and requires fewer VCs.

Yoon et al. propose using four networks (one for each message class of the MOESI protocol) as an alternative to using virtual channels [135]. In contrast, the dual-network hybrid organization, proposed in this thesis, makes better use of wire resources by requiring fewer networks, improves network load balance, and boosts performance under a fixed wire budget by supporting a wider response network as compared to a design with multiple dedicated NoCs.

Manevich et al. propose using a hybrid NoC architecture with a bus to broadcast the short commands of the coherence protocol [82]. This work can be considered similar to the heterogeneous approach of Balfour and Dally [7], except that it relies on a bus for the short messages instead of a second network. While appealing for CMPs integrating a small number of cores, a bus-based architecture is hard to scale to manycore configurations that are likely in future server chips.

Higher radix topologies [34] are considered a viable approach for reducing NoC power consumption. These topologies rely on rich physical connectivity to eliminate a fraction of router traversals. However, the duality-based concept is orthogonal to the network topology and hence the CCNoC concept can be extended to higher radix topologies.

Much research has focused on reducing router buffer power which accounts for a substantial fraction of overall NoC power. The techniques range from those that target a more efficient implementation of buffers [84], bypass buffers [122], or eliminate buffers all together [89]. While many of these techniques increase complexity, they are equally applicable and complementary to CCNoC, as CCNoC targets both buffer and crossbar power consumption.

5.2 Memory Systems

A substantial body of work has identified DRAM to be a significant contributor to system power. Leveraging the observation that off-chip memory bandwidth is not utilized by today's processors, prior work has either applied voltage frequency scaling to the memory controller and frequency scaling to the memory interface and devices [25, 26], or proposed using low-power low-speed memory interfaces [80, 133]. However, emerging manycore processors require significant amounts of memory bandwidth, thereby mandating high-speed memory interfaces to maximize per-pin available bandwidth. In this thesis, we use a high-bandwidth cache to filter most of DDR accesses, and utilize under-clocked DIMMs to reduce idle power. Nevertheless, frequency scaling could be leveraged to allow for adjusting available DDR bandwidth dynamically.

5.2.1 Mitigating static power

Prior work has sought to reduce static power of high-speed interfaces by employing a dynamic data rate range or exploiting various power-down states [3]. However there are two main issues that prevent such mechanisms from gaining traction. First, interfaces that support a range of data rates increase dynamic power consumption as compared to fixed-rate interfaces [97], and data rate adjustment requires a time-consuming (e.g., 100 nanoseconds) process of re-locking the receiving clock data recovery bandwidth [1]. Second, exploiting power-down states can hurt performance due to high sleep and wake-up times (a few microseconds) [1, 3].

In contrast to high-speed interfaces, which have high wake-up times, wake-up times of conventional DDR interfaces can be reduced to a few tens of nanoseconds by re-engineering the delay-locked loop mechanisms [81]. Such techniques can leverage power-down states of conventional DIMMs at the cost of small performance degradation. In MeSSOS, most of accesses are served by its high-bandwidth cache, and hence DDR access latency is not critical to system performance. Thus, MeSSOS could leverage power-down states of DIMMs with negligible impact on system performance.

Previous heterogeneous systems focused on utilizing memory emerging technologies, such as PCM, to enable high memory capacity at low static power. To achieve high performance, these systems utilize conventional DRAM as a cache for a larger PCM-based memory system [38, 102]. However, as they are using conventional DDR/LPDDR interfaces, the memory bandwidth they provide is not sufficient to satisfy the demands of emerging scale-out servers.

5.2.2 Mitigating activation power

One way to minimize activation energy is through modifications to DRAM chips or interfaces. Leveraging the observation that applications access only a few words within a cache block, researchers have proposed re-engineering the processor/memory interface to allow for activating only the DRAM chips at which the requested words are stored [4, 52, 134, 136]. While potentially effective in the server context as well, these proposals require disruptive changes to commodity memory technology. Historically, such disruptive proposals have failed to gain traction in the DRAM industry, which is focused on commoditization and adheres to rigid standards to ensure broad compatibility.

A non-disruptive approach to improving DRAM energy efficiency is to increase the fraction of DRAM accesses that hit in the row buffer, thus eliminating redundant page activations.

Sudan et al. [117] have observed that desktop and parallel applications consist of a small number of OS pages (100) that account for a high fraction of memory accesses (25%) and that these accesses are centered around only a few cache blocks. To improve row buffer locality, the authors have proposed mechanisms that identify and merge such pages. However, server applications operate on vast datasets and a large number of accesses go to pages that were not previously visited. Any energy gains of this technique would come at the cost of high storage requirements as it has to track an enormous number of pages.

Advanced prefetchers can enforce row buffer hits by predicting spatial footprints of load-triggered memory reads within a page [113]; however, these proposals carry a high storage overhead (60KB per core) and ignore store-triggered memory reads and LLC writebacks. Eager writeback mechanisms [70, 116] can improve row buffer locality for writebacks, but only to a limited degree as they schedule writebacks of only a few adjacent cache blocks at a time. In contrast to these works, BuMP maximizing row buffer locality by employing a comprehensive mechanism that targets all types of memory accesses while incurring a nominal area and energy cost.

Prior work has proposed hardware and software mechanisms to guide page-based prefetching [15, 123]. Stealth prefetching requires keeping address-related metadata (hundreds of KBs per core) to decide the subset of a page which should be fetched after a certain number of cache blocks are accessed within the page [15]. This thesis makes the observation that page density is code-correlated, thus allowing for fetching high-density pages with the first read to the page as opposed to multiple reads (Stealth prefetching) while introducing much lower storage overhead.

Guided region prefetching [123] uses compiler hints to direct both spatial and non-spatial (e.g., pointer-chasing) prefetches. However, rapid dataset changes in server applications present a challenge for static and profile-based approaches. Moreover, these approaches require application changes or recompilation, while our work is software transparent and adapts to changing application behavior.

5.3 Instruction-Based Prediction

Instruction-based predictors have been extensively used in the context of data prefetching [68, 113], cache static power management [17], cache-coherence action prediction [59], NOC power reduction [62], last-write prediction [124], on-chip granularity prediction [69], and off-chip bandwidth reduction [54, 134]. However, none of these works has targeted improving row buffer locality by predicting the memory page access density.

6 Concluding Remarks

The information revolution of the last decade has been fueled by the digitization of almost all human activities. Scale-out datacenters are the workhorses of this information age as they collect, store, and process all the data. These datacenters distribute vast datasets across a large number of servers, typically into memory-resident shards so as to maximize throughput and maintain tight tail latencies.

While data is driving the skyrocketing demands for computational resources, processor and memory manufacturing industries have reached fundamental efficiency limits, no longer able to increase server energy efficiency at a sufficient pace. Energy is becoming as the main impediment to the scalability of information technology (IT) with huge economic and ecological implications. Delivering scalable and sustainable IT calls for innovation in IT infrastructure design.

As memory has taken a central role in IT infrastructure, specialized memory-centric architectures are required to fully utilize the IT's costly memory investment. In response, processor architects are resorting to manycore architectures to leverage the abundant request-level parallelism found in data-centric applications. Specialized processors fully utilize available memory resources, thereby increasing IT efficiency by almost an order of magnitude.

On-chip interconnects and the memory system are emerging as major performance and energy-efficiency bottlenecks in specialized memory-centric servers. Because manycore server chips execute a large number of concurrent requests, they frequently access last-level cache for fetching instructions, and off-chip memory for accessing dataset objects.

In Chapter 2, we demonstrated that existing on-chip interconnects are area- and power-inefficient as they are designed to support both core-to-core and core-to-LLC communication although on-chip network activity in servers is dominated by core-to-LLC communication. We proposed Cache-Coherence Network-on-Chip (CCNoC), a specialized on-chip interconnect to fit the traffic characteristics of core-to-LLC communication via a pair of asymmetric request and response networks with different datapath width, router microarchitecture, and flow control strategy. CCNoC improves on-chip interconnect area and power efficiency and boosts server throughput for given area and power budgets.

In Chapter 3, we demonstrated that last-level caches and memory controllers of scale-out servers fail to exploit the coarse granularity at which memory is accessed and organized due to absence of information within the memory hierarchy about memory access patterns, resulting in poor memory system performance and energy efficiency. We proposed Bulk Memory Prediction and Streaming (BuMP) to identify accesses to coarse-grained objects, and to trigger bulk transfers of coarse-grained objects between processor and off-chip memory. In doing so, BuMP exploits the abundant spatial locality found in scale-out servers, and exploits the coarse granularity at which off-chip memory is organized, thus improving server throughput and memory energy efficiency.

In Chapter 4, we demonstrated that manycore servers impose high memory capacity and bandwidth requirements. Emerging SerDes-connected memory (SCM) can break the pin bandwidth constraints of modern DDR interfaces and provide the required bandwidth, but at a significant power cost and high latency overhead due to large point-to-point memory networks required to host vast datasets. Leveraging the observation that dataset popularity in servers is highly skewed, we proposed a heterogeneous Memory System for Scale-out Servers (MeSSOS) that employs SCM modules as an off-chip high-bandwidth cache in front of conventional DRAM modules. As the HBC is effective in filtering most of memory accesses, conventional DRAM modules can be clocked at low frequency, thus enabling high memory capacity at relatively low static power overhead.

6.1 Future Directions

While the memory-centric server architecture evaluated and presented in this thesis delivers almost a ten-fold-increase in datacenter efficiency, it still views the processor as the central system component, requiring data to be moved constantly from memory and storage to the processor. In this thesis, we architected a novel memory system for memory-centric servers, providing efficient access to the memory-resident datasets of data-centric applications. However, this thesis does not directly tackle the fundamental efficiency challenges that computing will be facing in the next decades.

Moving forward, alternative architectures are required so as to provide a scalable and sustainable infrastructure for data-centric IT. Near-memory computing sounds as an attractive architecture to mitigate the ever-increasing energy requirements of data-centric IT. These architectures are becoming feasible as die-stacking technology allows for fusing high-performance logic and memory devices together by stacking multiple memory layers on top of a logic layer. The emergence of these hybrid modules presents an opportunity of executing critical tasks near memory by off-loading memory-intensive and/or compute-intensive operations to specialized units that reside on the logic layer. Near-memory computing can provide a 100-fold-improvement in datacenter efficiency by minimizing the energy consumption of computation (specialized units consume less energy than general-purpose cores), and reducing energy-intensive off-chip data movement.

Bibliography

[1] D. Abts, M. R. Marty, P. M. Wells, P. Klausier, and H. Liu. Energy proportional datacenter networks. In *International Symposium on Computer Architecture*, June 2010.

[2] V. Agrawal, A. Dabral, T. Palit, Y. Shen, and M. Ferdman. Architectural support for dynamic linking. In *International Symposium on Architectural Support for Programming Languages and Operating Systems*, Mar. 2015.

[3] J. Ahn, S. Yoox, and K. C. Choi. Dynamic power management of off-chip links for hybrid memory cubes. In *Design Automation Conference*, June 2014.

[4] J. H. Ahn, J. Leverich, R. Schreiber, and N. P. Jouppi. Multicore DIMM: An energy efficient memory module with independently controlled DRAMs. *Computer Architecture Letters*, 8(1):5–8, Jan.-June 2009.

[5] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny. Workload analysis of a large-scale key-value store. In *ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, June 2012.

[6] K. Bains. DDR4 power features for servers.

[7] J. Balfour and W. J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *International Conference on Supercomputing*, June 2006.

[8] V. Ballan, O. Oluwole, G. Kodani, C. Zhong, S. Maheswari, R. Dadi, A. Amin, G. Bhatia, P. Mills, A. Ragab, and E. Lee. A 130mW 20Gb/s half-duplex serial link in 28nm CMOS. In *International Solid-State Circuits Conference*, Feb. 2014.

[9] L. A. Barroso and U. Holzle. 2009. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machine. 1st Edition. Madison: Morgan & Claypool.

[10] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCauley, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die stacking (3D) microarchitecture. In *International Symposium on Microarchitecture*, Dec. 2006.

[11] G. Blake, R. G. Dreslinski, T. Mudge, and K. Flautner. Evolution of thread-level parallelism in desktop applications. In *International Symposium on Computer Architecture*, June 2010.

[12] N. Bronson, Z. Amsden, G. Cabrera, P. Chakka, P. Dimov, H. Ding, J. Ferris, A. Giardullo, S. Kulkarni, H. Li, M. Marchukov, D. Petrov, L. Puzar, Y. J. Song, and V. Venkataramani. TAO: Facebook's distributed data store for the social graph. In *USENIX Conference on Annual Technical Conference*, June 2013.

[13] J. Bulzacchelli, T. Beukema, D. Storaska, P. Hsieh, S. Rylov, D. Furrer, D. Gardellini, A. Prati, C. Menolfi, D. Hanson, J. Hertle, T. Morf, V. Sharma, R. Kelkar, H. Ainspan, W. Kelly, G. Ritter, J. Garlett, R. Callan, T. Toifl, and D. Friedman. 28Gb/s 4-tap FFE/15-tap DFE serial link transceiver in 32nm SOI CMOS technology. In *International Solid-State Circuits Conference*, Feb. 2012.

[14] J. Bulzacchelli, T. Dickson, Z. T. Deniz, H. Ainspan, B. Parker, M. Beakes, S. Rylov, and D. Friedman. 78mW 11.1Gb/s 5-tap DFE receiver with digitally calibrated current-integrating summers in 65nm CMOS. In *International Solid-State Circuits Conference*, Feb. 2009.

[15] J. F. Cantin, M. H. Lipasti, and J. E. Smith. Stealth prefetching. In *International Conference on Architectural Support for Operating Systems and Programming Languages*, Oct. 2006.

[16] M. Cha, A. Mislove, and K. P. Gummadi. A measurement-driven analysis of information propagation in the Flickr social network. In *International World Wide Web Conference*, Apr. 2009.

[17] C. F. Chen, S.-H. Yang, B. Falsafi, and A. Moshovos. Accurate and complexity-effective spatial pattern prediction. In *International Symposium on High Performance Computer Architecture*, Feb. 2004.

[18] X. Cheng, C. Dale, and J. Liu. Statistics and social network of YouTube videos. In *International Workshop on Quality of Service*, June 2008.

[19] Cloudera. How-to: Select the Right Hardware for Your New Hadoop Cluster.

[20] Cloudera. What Do Real-Life Apache Hadoop Workloads Look Like?

[21] CloudSuite 2.0. http://parsa.epfl.ch/cloudsuite.

[22] E. Cooper-Balis, P. Rosenfold, and B. Jacob. Buffer-on-board memory system. In *International Symposium on Computer Architecture*, June 2012.

[23] A. K. Coskun. 3D stacking as an enabler for low-power high-performance computing. In *Workshop on Design for 3D Silicon Integration*, June 2013.

[24] Datastax. Benchmarking top NoSQL databases.

[25] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu. Memory power management via dynamic voltage/frequency scaling. In *International Conference on Autonomic Computing*, June 2011.

[26] Q. Deng, D. Meisner, L. E. Ramos, T. F. Wenisch, and R. Bianchini. MemScale: Active low-power modes for main memory. In *International Conference on Architectural Support for Programming Languages and Operating Systems)*, Mar. 2011.

[27] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *International Symposium on Computer Architecture*, June 2011.

[28] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2012.

[29] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. A case for specialized processors for scale-out workloads. *IEEE Micro*, 34(3):31–42, May-June 2014.

[30] M. Ferdman, T. F. Wenisch, A. Ailamaki, B. Falsafi, and A. Moshovos. Temporal instruction fetch streaming. In *International Symposium on Microarchitecture*, Nov. 2008.

[31] K. Fukuda, H. Yamashita, F. Yuki, M. Yagyu, R. Nemoto, T. Takemoto, T. Saito, N. Chujo, K. Yamamoto, H. Kanai, and A. Hayashi. An 8Gb/s transceiver with 3x-oversampling 2-threshold eye-tracking CDR circuit for -36.8dB-loss backplane. In *International Solid-State Circuits Conference*, Feb. 2008.

[32] G. Gangasani, C.-M. Hsu, J. Bulzacchelli, S. Rylov, T. Beukema, D. Freitas, W. Kelly, M. Shannon, J. Qi, H. Xu, J. Natonio, T. Rasmus, J.-R. Guo, M. Wielgos, J. Garlett, M. Sorna, and M. Meghelli. A 16-Gb/s backplane transceiver with 12-tap current integrating DFE and dynamic adaptation of voltage offset and timing drifts in 45-nm SOI CMOS technology. In *Custom Integrated Circuits Conference*, Sept. 2011.

[33] B. Grot, D. Hardy, P. Lotfi-Lamran, C. Nicopoulos, Y. Sazeides, and B. Falsafi. Optimizing datacenter TCO with scale-out processors. *IEEE Micro*, 32(5):52–63, Sep.-Oct. 2012.

[34] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. Express cube topologies for on-chip interconnects. In *International Symposium on High-Performance Computer Architecture*, Feb. 2009.

[35] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. Kilo-noc: A heterogeneous network-on-chip architecture for scalability and service guarantees. In *International Symposium on Computer Architecture*, June 2011.

[36] L. Gwennap. Qualcomm Krait 400 hits 2.3GHz. *Microprocessor Report*, 27(1):1, 6–8, Jan. 2013.

[37] L. Gwennap. Thunderx rattles server market. *Microprocessor Report*, 28(6):1–4, June 2014.

[38] T. J. Ham, B. K. Chelepalli, N. Xue, and B. Lee. Disintegrated control for energy-efficient and heterogeneous memory systems. In *International Symposium on High-Performance Computer Architecture*, Feb. 2013.

[39] N. Hardavellas, M. Ferdman, A. Ailamaki, and B. Falsafi. Power scaling: the ultimate obstacle to 1K-core chips. In *Technical Report NWU-EECS-10-05, Northwestern University, Evanston, IL*, Mar. 2010.

[40] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Reactive NUCA: near-optimal block placement and replication in distributed caches. In *International Symposium on Computer Architecture*, June 2009.

[41] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, Jul.-Aug. 2011.

[42] N. Hardavellas, I. Pandis, R. Johnson, N. Mancheril, A. Ailamaki, and B. Falsafi. Database servers on chip multiprocessors: Limitations and opportunities. In *Biennial Conference on Innovative Data Systems Research*, Jan. 2007.

[43] M. Harwood, N. Warke, R. Simpson, T. Leslie, A. Amerasekera, S. Batty, D. Colman, E. Carr, V. Gopinathan, S. Hubbins, P. Hunt, A. Joy, P. Khandelwal, B. Killips, T. Krause, S. Lytollis, A. Pickering, M. Saxton, D. Sebastio, G. Swanson, A. Szczepanek, T. Ward, J. Williams, R. Williams, and T. Willwerth. A 12.5Gb/s SerDes in 65nm CMOS using a baud-rate ADC with digital receiver equalization and clock recovery. In *International Solid-State Circuits Conference*, Feb. 2007.

[44] Y. Hidaka, T. Horie, Y. Koyanagi, T. Miyoshi, H. Osone, S. Parikh, S. Reddy, T. Shibuya, Y. Umezawa, and W. W. Walker. A 4-channel 10.3Gb/s transceiver with adaptive phase

equalizer for 4-to-41dB loss PCB channel. In *International Solid-State Circuits Conference*, Feb. 2011.

[45] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein. Scaling, power, and the future of CMOS. In *Electron Devices Meeting, 2005. IEDM Technical Digest. IEEE International*, Dec. 2005.

[46] Hybrid memory cube consortium. Hybrid memory cube specification 1.0.

[47] IDC. The digital universe in 2020: Big Data, bigger digital shadows, and biggest grow in the far east.

[48] Intel scalable memory buffer. http://www.intel.com/content/dam/doc/datasheet/7500-7510-7512-scalable-memory-buffer-datasheet.pdf.

[49] International technology roadmap for semiconductors.

[50] J. Jaussi, G. Balamurugan, S. Hyvonen, T.-C. Hsueh, T. Musah, G. Keskin, S. Shekhar, J. Kennedy, S. Sen, R. Inti, M. Mansuri, M. Leddige, B. Horine, C. Roberts, R. Mooney, and B. Casper. 205mW 32Gb/s 3-Tap FFE/6-tap DFE bidirectional serial link in 22nm CMOS. In *International Solid-State Circuits Conference*, Feb. 2014.

[51] J. Jeddeloh and B. Keeth. Hybrid memory cube new DRAM architecture increases density and performance. In *2012 International Symposium on VLSI Technology - Digest of Technical Papers*, June 2012.

[52] M. K. Jeong, D. H. Yoon, D. Sunwoo, M. Sullivan, I. Lee, and M. Erez. Balancing DRAM locality and parallelism in shared memory CMP systems. In *International Symposium on High-Performance Computer Architecture*, Feb. 2012.

[53] D. Jevdjic, G. H. Loh, C. Kaynak, and B. Falsafi. Unison Cache: A scalable and effective die-stacked DRAM Cache. In *International Symposium on Microarchitecture*, Dec. 2014.

[54] D. Jevdjic, S. Volos, and B. Falsafi. Die-Stacked DRAM caches for servers: Hit-Ratio, latency, or bandwidth? Have it all with Footprint Cache. In *International Symposium on Computer Architecture*, June 2013.

[55] X. Jiang, N. Madan, L. Zhao, M. Upton, R. Iyer, S. Makineni, D. Newell, Y. Solihin, and R. Balasubramonian. Chop: Adaptive filter-based dram caching for cmp server platforms. In *International Symposium on High Performance Computer Architecture*, Jan. 2010.

[56] A. Joy, H. Mair, H.-C. Lee, A. Feldman, C. Portmann, N. Bulman, E. Crespo, P. Hearne, P. Huang, B. Kerr, P. Khandelwal, F. Kuhlmann, S. Lytollis, J. Machado, C. Morrison, S. Morrison, S. Rabii, D. Rajapaksha, V. Ravinuthula, and G. Surace. Analog-DFE-based 16Gb/s SerDes in 40nm CMOS that operates across 34dB loss channels at Nyquist with a baud rate CDR and 1.2Vpp voltage-mode driver. In *International Solid-State Circuits Conference*, Feb. 2011.

[57] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. ORION 2.0: A fast and accurate noc power and area model for early- stage design space exploration. In *Proceedings of Design, Automation, and Test in Europe*, Apr. 2009.

[58] D. Kaseridis, J. Stuecheli, and L. K. John. Minimalist Open-page: A DRAM page-mode scheduling policy for the many-core era. In *International Symposium on Microarchitecture*, Dec. 2011.

[59] S. Kaxiras and J. R. Goodman. Improving CC-NUMA performance using instruction-based prediction. In *International Symposium on High-Performance Computer Architecture*, Jan. 1999.

[60] C. Kaynak, B. Grot, and B. Falsafi. SHIFT: Shared history instruction fetch for lean-core server processors. In *International Symposium on Microarchitecture*, Dec. 2013.

[61] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou. The researcher's guide to the data deluge: Querying a scientific database in just a few seconds. In *International Conference on Very Large Data Bases (VLDB)*, Aug. 2011.

[62] H. Kim, P. Ghoshal, B. Grot, P. V. Gratz, and D. A. Jimenez. Reducing network-on-chip energy consumption through spatial locality speculation. In *International Symposium on Networks-on-Chip*, May 2011.

[63] Y. Kim, M. Papamichael, O. Mutlu, and M. Harchol-Blter. Thread Cluster Memory Scheduling: exploiting differences in memory access behavior. In *International Symposium on Microarchitecture*, Dec. 2010.

[64] H. Kimura, P. Aziz, T. Jing, A. Sinha, R. Narayan, H. Gao, P. Jing, G. Hom, A. Liang, E. Zhang, A. Kadkol, R. Kothari, G. Chan, Y. Sun, B. Ge, J. Zeng, K. Ling, M. Wang, A. Malipatil, S. Kotagiri, L. Li, C. Abel, and F. Zhong. 28Gb/s 560mW multi-standard SerDes with single-stage analog front-end and 14-tap decision-feedback equalizer in 28nm CMOS. In *International Solid-State Circuits Conference*, Feb. 2014.

[65] O. Kocberber, B. Grot, J. Picorel, B. Falsafi, K. Lim, and P. Ranganathan. Meet the walkers: Accelerating index traversals for in-memory databases. In *International Symposium on Microarchitecture*, Dec. 2013.

[66] J. G. Koomey. Growth in data center electricity use 2005 to 2010.

[67] C. Kozyrakis, A. Kansal, S. Sankar, and K. Vaid. Server engineering insights for large-scale online services. *IEEE Micro*, 30(4):8 –19, Jul.-Aug. 2010.

[68] S. Kumar and C. Wilkerson. Exploiting spatial locality in data caches using spatial footprints. In *International Symposium on Computer Architecture*, June 1998.

[69] S. Kumar, H. Zhao, A. Shriraman, E. Matthews, S. Dwarkadas, and L. Shannon. Amoeba-cache: Adaptive blocks for eliminating waste in the memory hierarchy. In *International Symposium on Microarchitecture*, Dec. 2012.

[70] H.-H. S. Lee, G. S. Tyson, and M. K. Farrens. Eager Writeback - a technique for improving bandwidth utilization. In *International Symposium on Microarchitecture*, Dec. 2000.

[71] S. Li, K. Chen, J. Ho Ahn, J. B. Brockman, and N. P. Jouppi. CACTI-P: Architecture-Level modeling for SRAM-based structures with advanced leakage reduction techniques. In *International Conference on Computer-Aided Design*, Nov. 2011.

[72] S. Li, J. Ho Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *International Symposium on Microarchitecture*, Dec. 2009.

[73] K. Lim, D. Meisner, A. G. Saidi, P. Ranganathan, and T. F. Wenish. Thin servers with smart pipes: Designing SoC accelerators for memcached. In *International Symposium on Computer Architecture*, June 2013.

[74] W.-F. Lin, S. K. Reinhardt, and D. Burger. Reducing DRAM latencies with an integrated memory hierarchy design. In *International Symposium on High-Performance Computer Architecture*, Jan. 2001.

[75] G. H. Loh. 3D-stacked memory architectures for multi-core processors. In *International Symposium on Computer Architecture*, June 2008.

[76] G. H. Loh and M. D. Hill. Efficiently enabling conventional block sizes for very large die-stacked DRAM caches. In *International Symposium on Microarchitecture*, Dec. 2011.

[77] P. Lotfi-Kamran, M. Ferdman, D. Crisan, and B. Falsafi. Turbotag: Lookup filtering to reduce coherence directory power. In *International Symposium on Low Power Electronics and Design*, Aug. 2010.

[78] P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi. Scale-Out processors. In *International Symposium on Computer Architecture*, June 2012.

[79] P. Lu and K. Shen. Virtual machine memory access tracing with hypervisor exclusive cache. In *Usenix Annual Technical Conference*, June 2007.

[80] K. T. Malladi, F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis, and M. Horowitz. Towards energy-proportional datacenter memory with mobile DRAM. In *International Symposium on Computer Architecture*, June 2012.

[81] K. T. Malladi, I. Shaeffer, L. Gopalakrishnan, D. Lo, B. C. Lee, and M. Horowitz. Rethinking DRAM power modes for energy proportionality. In *International Symposium on Microarchitecture*, Dec. 2012.

[82] R. Mavenich, I. Walter, I. Cidon, and A. Koldny. Best of Both Worlds: A bus-enhanced NoO (BENoC). In *International Symposium on Networks-on-Chip*, May 2009.

[83] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating server idle power. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2009.

[84] G. Michelogiannakis, J. Balfour, and W. Dally. Elastic-buffer flow control for on-chip networks. In *International Symposium on High-Performance Computer Architecture*, Feb. 2009.

[85] Micron 1Gb: x4, x8, x16 DDR3 SDRAM.

[86] Micron 2Gb: x4, x8, x16 DDR3 SDRAM.

[87] Micron 4Gb: x4, x8, x16 DDR3 SDRAM.

[88] Micron DDR3 SDRAM Power Calculation. http://www.micron.com/products/support/power-calc.

[89] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. In *International Symposium on Computer Architecture*, June 2009.

[90] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *International Symposium on Microarchitecture*, Dec. 2007.

[91] N. Nedovic, A. Kristensson, S. Parikh, S. Reddy, S. McLeod, N. Tzartzanis, K. Kanda, T. Yamamoto, S. Matsubara, M. Kibune, Y. Doi, S. Ide, Y. Tsunoda, T. Yamabana, T. Shibasaki, Y. Tomita, T. Hamada, M. Sugawara, T. Ikeuchi, N. Kuwata, H. Tamura, J. Ogawa, and W. Walker. 3 Watt 39.8–44.6 Gb/s dual-mode SFI5.2 SerDes chip set in 65 nm CMOS. *IEEE Journal of Solid-State Circuits*, 45(10), Oct. 2010.

[92] K. J. Nesbit, N. Aggarwal, J. Laudon, and J. E. Smith. Fair queuing memory systems. In *International Symposium on Microarchitecture*, Dec. 2006.

[93] S. Novakovic, A. Daglis, E. Bugnion, B. Falsafi, and B. Grot. Scale-ouut NUMA. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2014.

[94] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazieres, S. Mitra, A. Narayanan, G. Parulkar, M. Rosenblum, S. M. Rumble, E. Stratmann, and R. Stutsman. The Case for RAMClouds: Scalable high-performance storage entirely in DRAM. *SIGOPS Operating Systems Review*, 43(4):92–105, Dec. 2009.

[95] J. T. Pawlowksi. Hybrid memory cube.

[96] L.-S. Peh. *Flow Control and Micro-Architectural Mechanisms for Extending the Performance of Interconnection Networks.* PhD thesis, Stanford University, 2001.

[97] J. Poulton, R. Palmer, A. M. Fuller, T. Greer, J. Eyles, W. J. Dally, and M. Horowitz. A 14-mW] 6.25-Gb/s Transceiver in 90-nm CMOS. *IEEE Journal of Solid-State Circuits*, 42(12), Dec. 2007.

[98] M. Pozzoni, S. Erba, P. Viola, M. Pisati, E. Depaoli, D. Sanzogni, R. Brama, D. Baldi, M. Repossi, and F. Svelto. A multi standard 1.5 to 10Gb/s latch-based 3-tap DFE receiver with a SSC tolerant CDR for serial backplane communication. In *International Symposium on VLSI circuits*, June 2008.

[99] S. H. Pugsley, J. Jestes, H. Zhang, R. Balasubramonian, V. Srinivasan, A. Buyuktosunoglu, A. Davis, and F. Li. NDC: Analyzing the impact of 3D-stacked memory+logic devices on

MapReduce workloads. In *International Symposium on Performance Analysis of Systems and Software*, Mar. 2014.

[100] A. Putman, A. Caulfield, E. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Xiao, and D. Burger. A reconfigurable fabric for accelerating large-scale datacenter services. In *International Symposium on Computer Architecture*, June 2014.

[101] M. K. Qureshi and G. Loh. Fundamental latency trade-off in architecting DRAM caches: Outperforming impractical SRAM-tags with a simple and practical design. In *International Symposium on Microarchitecture*, Dec. 2012.

[102] M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable high-performance main memory system using phase-change memory technology. In *International Symposium on Computer Architecture*, June 2009.

[103] M. Ramezani, M. Abdalla, A. Shoval, M. van Ierssel, A. Rezayee, A. McLaren, C. Holdenried, J. Pham, E. So, D. Cassan, and S. Sadr. An 8.4mW/Gb/s 4-lane 48Gb/s multi-standard-compliant transceiver in 40nm digital CMOS technology. In *International Solid-State Circuits Conference*, Feb. 2011.

[104] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens. Memory access scheduling. In *International Symposium on Computer Architecture*, June 2000.

[105] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin. Scaling the bandwidth wall: challenges in and avenues for cmp scaling. In *International Symposium on Computer Architecture*, June 2009.

[106] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. DRAMSim2: A cycle accurate memory system simulator. *Computer Architecture Letters*, 10(1):16 –19, Jan.-June 2011.

[107] A. Roy, I. Mihailovic, and W. Zwaenepoel. X-Stream: Edge-centric graph processing using streaming partitions. In *International Symposium on Operating Systems Principles*, Nov. 2013.

[108] J. Savoj, K. Hsieh, P. Upadhyaya, F.-T. An, A. Bekele, S. Chen, X. Jiang, K. W. Lai, C. F. Poon, A. Sewani, D. Turker, K. Venna, D. Wu, B. Xu, E. Alon, and K. Chang. A wide common-mode fully-adaptive multi-standard 12.5Gb/s backplane transceiver in 28nm CMOS. In *International Symposium on VLSI circuits*, June 2012.

[109] D. Schinkel, E. Mensink, E. Klumperink, van Tuijl, and B. Nauta. Low-power, high-speed transceivers for network-on-chip communication. *IEEE Transactions on VLSI Systems*, 17(1):12–21, Jan. 2009.

[110] K. L. Sewell. *Scaling high-performance interconnect architectures to many-core systems*. PhD thesis, 2012.

[111] Q. Shaolei, F. Zhong, W. Liu, P. Aziz, T. Jing, J. Dong, C. Desai, H. Gao, M. Garcia, G. Hom, T. Huynh, H. Kimura, R. Kothari, L. Li, C. Liu, S. Lowrie, K. Ling, A. Malipatil, R. Narayan, T. Prokop, C. Palusa, A. Rajashekara, A. Sinha, C. Zhong, and E. Zhang. 1.0625-to-14.025Gb/s multimedia transceiver with full-rate source-series-terminated transmit driver and floating-tap decision-feedback equalizer in 40nm CMOS. In *International Solid-State Circuits Conference*, Feb. 2011.

[112] J. L. Shin, H. Park, H. Li, A. Smith, Y. Choi, H. Sathianathan, S. Dash, S. Turullols, S. Kim, R. Masleid, G. Konstadinidis, R. Golla, M. J. Doherty, G. Grohoski, and C. McAllister. The next-generation 64b SPARC core in a T4 SoC processor. In *International Conference on Solid-State Circuits*, Feb. 2012.

[113] S. Somogyi, T. F. Wenisch, A. Ailamaki, B. Falsafi, and A. Moshovos. Spatial memory streaming. In *International Symposium on Computer Architecture*, June 2006.

[114] F. Spanga, L. Chen, M. Deshpande, Y. Fan, D. Gambetta, S. Gowder, S. Iyer, R. Kumar, P. Kwok, R. Krishnamurthy, C.-C. Lin, R. Mohanavelu, R. Nicholson, J. Ou, M. Pasquarella,

K. Prasad, H. Rustam, L. Tong, A. Tran, J. Wu, and X. Zhang. A 78mW 11.8Gb/s serial link transceiver with adaptive RX equalization and baud-rate CDR in 32nm CMOS. In *International Solid-State Circuits Conference*, Feb. 2010.

[115] J. Stuecheli. IBM Power8.

[116] J. Stuecheli, D. Kaseridis, D. Daly, H. C. Hunter, and L. K. John. The Virtual Write Queue: Coordinating DRAM and last-level cache policies. In *International Symposium on Computer Architecture*, June 2010.

[117] K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramonian, and A. Davis. Micro-Pages: Increasing DRAM efficiency with locality-aware data placement. In *International Conference on Architectural Support for Programming Languages and Operating Systems)*, Mar. 2010.

[118] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The RAW Microprocessor: A computational fabric for software circuits and general purpose programs. *IEEE Micro*, 22(2):25–35, Mar.-Apr. 2002.

[119] T. Transaction Processing Performance Council. http://www.tpc.org/default.asp.

[120] S. Volos, J. Picorel, B. Falsafi, and B. Grot. BuMP: Bulk memory access prediction and streaming. In *International Symposium on Microarchitecture*, Dec 2014.

[121] S. Volos, C. Seiculescu, B. Grot, N. Khosro Pour, B. Falsafi, and G. De Micheli. CCNoC: Specializing on-chip interconnects for energy efficiency in cache-coherent servers. In *International Symposium on Networks-on-Chip*, May 2012.

[122] H. Wang, L.-S. Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *International Symposium on Microarchitecture*, Dec. 2003.

[123] Z. Wang, D. Burger, K. S. McKinley, S. K. Reinhardt, and C. C. Weems. Guided region prefetching: A cooperative hardware/software approach. In *International Symposium on Computer Architecture*, June 2003.

[124] Z. Wang, S. M. Khan, and D. A. Jimenez. Improving writeback efficiency with decoupled last write prediction. In *International Symposium on Computer Architecture*, June 2012.

[125] T. F. Wenisch, R. E. Wunderlich, M. Ferdman, A. Ailamaki, B. Falsafi, and J. C. Hoe. SimFlex: Statistical sampling of computer system simulation. *IEEE Micro*, 26(4):18–31, Jul.-Aug. 2006.

[126] B. Wheeler. Tilera sees opening in clouds. *Microprocessor Report*, 25(7):13–16, July 2011.

[127] B. Wheeler. Cavium joins ARM-server fray. *Microprocessor Report*, 26(8):1,5–6,16, Aug. 2012.

[128] Wikipedia. Does Wikipedia traffic obey Zipf's law?

[129] R. D. Williams, T. Sze, D. Huang, S. Pannala, and C. Fang. Server memory road map.

[130] D. H. Woo, N. H. Seong, D. L. Lewis, and H.-H. S. Lee. An optimized 3D-stacked memory architecture by exploiting excessive, high-density TSV bandwidth. In *International Symposium on High Performance Computer Architecture*, Jan. 2010.

[131] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe. SMARTS: Accelerating microarchitecture simulation via rigorous statistical sampling. In *International Symposium on Computer Architecture*, June 2003.

[132] H. Xi, J. Zhan, Z. Jia, X. Hong, L. Wang, L. Zhang, N. S. Sun, and G. Lu. Characterization of real workloads of web search engines. In *International Symposium on Workload Characterization*, Nov. 2011.

[133] D. H. Yoon, J. Chang, N. Muralimanohar, and P. Ranganathan. BOOM: Enabling mobile memory based low-power server DIMMs. In *International Symposium on Computer Architecture*, June 2012.

[134] D. H. Yoon, M. K. Jeong, M. Sullivan, and M. Erez. The dynamic granularity memory system. In *International Symposium on Computer Architecture*, June 2012.

[135] Y. J. Yoon, N. Concer, M. Petracca, and L. Carloni. Virtual channels vs. multiple physical networks: A comparative analysis. In *Design Automation Conference*, June 2010.

[136] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu. Mini-Rank: Adaptive DRAM architecture for improving memory power efficiency. In *International Symposium on Microarchitecture*, Dec. 2008.

STAVROS VOLOS

Contact email: svolos@gmail.com

RESEARCH INTERESTS

Computer architecture, server design for data-centric computing with emphasis on processor architectures, uncore technologies, and memory system organizations.

EDUCATION

Ecole Polytechnique Federale de Lausanne (EPFL) *September 2015*
Ph.D. in Computer & Communication Sciences *Lausanne, Switzerland*
Thesis: Memory Systems and Interconnects for Scale-Out Servers
Advisor: Prof. Babak Falsafi

National Technical University of Athens (NTUA) *July 2009*
Dipl.-Ing. in Electrical & Computer Engineering *Athens, Greece*
Highest Honors – top 1%

HONORS & AWARDS

IEEE Micro Top Picks from Computer Architecture Conferences of 2013, "A Case for Specialized Processors for Scale-Out Workloads", 2014.

Best Paper Award at ASLPOS-XXVII for Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware, 2012.

EPFL Computer Science Department Fellowship, 2009 – 2010.

Cyprus State Scholarship Foundation Fellowship, 2004 – 2012.

Greek State Scholarship Foundation Fellowship, 2004 – 2009.

Annual Greek State Scholarship Foundation Award, awarded for the highest GPA, 2004 – 2007.

PUBLICATIONS

PEER-REVIEWED CONFERENCE PAPERS

1. **BuMP: Bulk Memory Access Prediction and Streaming.**
Stavros Volos, Javier Picorel, Babak Falsafi, and Boris Grot. In *47th International Symposium on Microarchitecture,* December 2014.

2. **Die-Stacked DRAM Caches for Servers: Hit Ratio, Latency, or Bandwidth? Have It All with Footprint Cache.**
Djordje Jevdjic, Stavros Volos, and Babak Falsafi. In *40th International Symposium on Computer Architecture,* June 2013.

3. **Scale-Out Processors.**
Pejman Lotfi-Kamran, Boris Grot, Michael Ferdman, Stavros Volos, Onur Kocberber, Javier Picorel, Almutaz Adileh, Djordje Jevdjic, Sachin Idgunji, Emre Ozer, and Babak Falsafi. In *39th International Symposium on Computer Architecture,* June 2012.

4. **CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache-Coherent Servers.**
Stavros Volos, Ciprian Seiculescu, Boris Grot, Naser Khosro Pour, Babak Falsafi, and Giovanni De Micheli. In *6th International Symposium on Networks-on-Chip,* May 2012.

5. **Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware.**
Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. In *17th International Conference on Architectural Support for Programming Languages and Operating Systems,* March 2012.

JOURNALS

6. **A Case for Specialized Processors for Scale-Out Workloads.**
Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. In *IEEE Micro Top Picks,* May/June 2014.

7. **Quantifying the Mismatch Between Emerging Scale-Out Applications and Modern Processors.**
Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. In *ACM Transactions on Computer Systems (TOCS),* November 2012.

WORK EXPERIENCE

Ecole Polytechnique Federale de Lausanne September 2010 – September 2015
Research assistant at Parallel Systems Architecture Lab *Lausanne, Switzerland*

Microsoft Research June 2010 – September 2010
Intern at Computer Architecture Group *Redmond, WA, USA*

TEACHING ASSISTANTSHIPS

Information, Computation, Communication (undergraduate), Fall 2014.
Introduction to Database Systems (undergraduate), Spring 2014.
Advanced Multiprocessor Architecture (graduate), Fall 2013, Fall 2010.
Information Technology Project (undergraduate), Fall 2011.
Introduction to Multiprocessor Architecture (undergraduate), Spring 2011.

PROFESSIONAL SERVICE

External reviewer: MICRO'15, IEEE Micro'15, ISCA'15, MICRO'14, MemForum'14, HiPEAC'13, IISWC'12, HPCA'12, IISWC'11, HOTOS'11, DATE'11.

Primary architect of CloudSuite, a benchmark suite for scale-out applications.

Co-developer of Flexus, a full-system cycle-accurate multi-processor simulation framework, 2010 – 2015.

CloudSuite on Flexus Tutorial at EPFL (2015), ISCA (2012).

Organizer of the Fall 2012 weekly seminar of the Parallel Systems Architecture Lab at EPFL.

ACM SIGARCH and IEEE student member.