Optimising DRAM Caches for Latency in Datacenter Servers

Amna Shahab

Doctor of Philosophy

Institute of Computing Systems Architecture
School of Informatics
University of Edinburgh

2024

Abstract

The unyielding growth in the amount of data processed by datacenter servers warrants an unabated increase in cache capacities. Modern servers already dedicate a big proportion of their die real-estate to on-chip caches, with the largest last-level cache (LLC) accounting for up to a third of the die. Problematically, the slowdown in technology scaling constrains the expansion of LLC capacity as demanded by growing datasets. In the face of on-chip cache capacity constraints, systems today are increasingly being equipped with multi-gigabyte die-stacked DRAM caches. These DRAM stacks provide high access bandwidth through a combination of many DRAM banks and a wide interface, but fall short in improving access latency compared to main memory. Datacenter applications are performance sensitive to access latency as they exhibit low memory-level parallelism (MLP). High-bandwidth, high-latency stacked DRAM caches fail to benefit these datacenter applications. This thesis aims to optimise stacked DRAM caches for low latency by first identifying the sources of latency and then devising cache organisations that address them.

This thesis observes that the factors contributing to the high access latency of DRAM caches are: (i) on-chip interconnect (NOC) routing delay to reach the DRAM cache controller, (ii) queuing delay in the DRAM cache controller, (iii) horizontal traversal between the processor die and the DRAM stack, (iv) addressing and access latency in the DRAM core. We find that the aforementioned factors may be addressed at the architecture level to latency-optimise the DRAM cache, and propose two cache organisations.

We propose *On-PaCkage Partitioned DRAM Victim Cache* (CARVE), which minimises the various interconnect latencies by partitioning the DRAM stack into logically independent units, vaults, and utilising each vault as a victim cache for a shared on-chip LLC slice. This design retains the conventional on-chip cache hierarchy, and augments it with a new level of victim cache in DRAM vaults on package. We demonstrate that through a combination of fine-grained connections between the processor die and the DRAM stack, DRAM stack partitioning, and DRAM technology latency optimisation it is possible to architect a low-latency on-package DRAM cache.

We propose a novel *Die-Stacked Private LLC Organisation* (SILO) – which combines on-chip private caches with per-core LLC slices in die-stacked DRAM,

which enables further reduction of interconnect latency. Per-core private caches overcome the latency bottleneck of shared caches by limiting the length of interconnect that needs to be traversed on an access. To avoid long interconnect delays and maintain the latency benefits of a private cache, SILO organises the DRAM into vaults, each of which sits above a processor core.

To summarise, this thesis addresses the factors that lead to the long access latency of DRAM caches through two DRAM cache designs which minimise interconnect delays. We show, through simulation, that datacenter applications running on processors equipped with latency-optimised DRAM caches observe significant performance improvement compared to when running on conventional server processors.

Lay Summary

Datacenters running user-facing online applications such as web search employ hundreds of thousands of high-performance server processors. The size of the datasets that datacenter applications process is growing at a rapid rate. To sustain real-time responses to user requests, the servers require fast service of data from the memory subsystem. Unfortunately, the main memory latency has experienced only modest improvements over the past two decades, bottlenecking system performance.

In order to filter frequent long-latency main memory accesses, processors employ relatively small and fast storage close to the processing cores, in the form of a multi-level cache hierarchy. The last level of cache is the largest and consumes up to a third of the processor die area in recent server products. The cache storage capacity is required to grow in line with the size of the datasets in order to maintain real-time responses to users. Unfortunately, the processor die sizes are limited due to manufacturing challenges, restricting the scaling of cache capacities. Simultaneously, the larger the caches grow, the slower they respond due to the longer connecting wires, adding further difficulty to maintain real-time responses to users.

Meanwhile, die stacking technology has emerged as a means to overcome the area limitations of planar silicon, providing a manifold increase in storage capacity within a given area footprint. DRAM stacks have been deployed in recent products, placed next to the processor die, providing up to tens of gigabytes of capacity but at a high access latency. While DRAM stacks possess the ability to scale in capacity, their present arrangement and architecture hinders fast accesses, rendering them unsuitable. Accesses to DRAM stacks are slow for two reasons: (1) significant interconnect delays; (2) use of cost-optimised DRAM architectures, which favor area efficiency over access latency.

This thesis aims to address the interconnect and DRAM architecture latency sources in DRAM stacks. To that end, we first analyse the the sources of DRAM stack latency and address them through two new DRAM cache organisations.

To my son, Yahya. The sparkle in your eyes keeps me going.

Acknowledgements

My long and enriching PhD journey, filled with numerous twists and turns, was helped along by many. I express my deepest gratitude to everyone who has contributed to this thesis, both directly and indirectly. I will now attempt to specifically acknowledge a subset of these people whose contributions were crucial for the completion of my thesis.

I sincerely thank Prof. Boris Grot, my thesis advisor and my mentor. Your understanding, support, and patience in tough times made this thesis possible. You taught me what research is and how to ask critical questions when assessing data. Perhaps the most important skill you taught me is clear communication – crucial to research and teaching. I greatly appreciate you acknowledging my passion for teaching and offering me more teaching support duties.

I express my gratitude to my colleagues from EASE Lab - Antonios, Artemiy, David, Dmitrii, Mingcan, Priyank, Siavash, and Vasilis. I thoroughly enjoyed our group discussions, meetings, and outings. I owe special thanks to Artemiy and Priyank who not only helped me in the initial stages of my PhD, but also went through various intense submissions with me. We surely have many amusing submission stories to tell. Thank you to Artemiy and Dmitrii for our walks around the Meadows to help restore sanity during the pandemic. My deepest thanks to Artemiy, who was a lifesaver countless times when I encountered hardware failures. Your expertise and calm during stressful times was unparalleled.

I would like to thank my CDT cohort for all the good times when we all embarked upon our research journeys. Thank you Daniel H. for your crazy ideas for computer architecture research, and for being a source of constant positive energy. Thank you Daniel M. and Paul for all the tea breaks which helped clear my mind and revitalise me for the rest of the day.

I am deeply indebted to my family for their constant love and support. My mother and father, who instilled confidence and drive in me. My sister who always treated me with the most delicious food. My parents-in-law who essentially doubled the parental love and support I received, and for lending an ear when times were tough. Thank you to my husband, Junaid, who has seen me at my lowest times, reassured me, and urged me to believe in my work. Thank you for your love and understanding. Together we have received the most precious gift we could ever hope for: Yahya. Thank you Yahya for lighting up my days, for

motivating me to keep on going, and for even helping me write up this thesis with your tiny little fingers.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. Some of the material used in this thesis has been published in the following paper:

A. Shahab, M. Zhu, A. Margaritov, B. Grot. "Farewell My Shared LLC! A Case for Private Die-Stacked DRAM Caches for Servers". In Proceedings of the 51st International Symposium on Microarchitecture (MICRO). 2018 [1]

(*Amna Shahab*)

Table of Contents

List of Figures

List of Tables

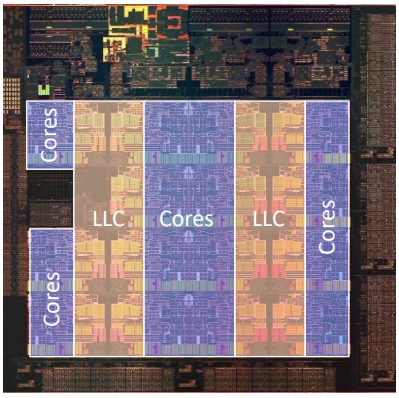
Chapter 1

Introduction

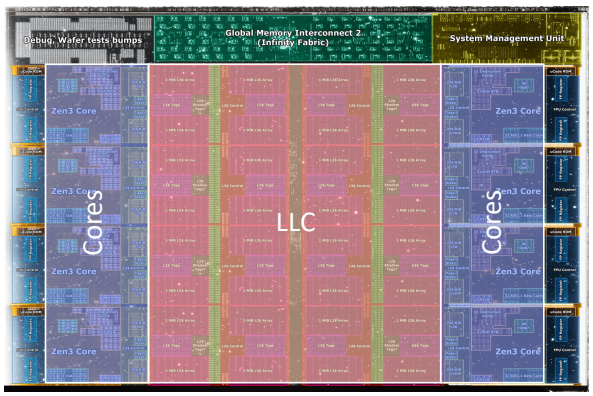
Datacenters underpin today's digital society by providing real-time storage, retrieval and processing capabilities for increasingly complex information-centric tasks. Datacenters already account for 3% of the global energy consumption [6] and datacenter demand is projected to grow by around 10% every year until 2030 [7]. The servers inside today's datacenters use many-core high-performance server processors to maximize overall throughput and control tail latency in latency-critical services [8]. As the volume of data consumed and created by both human and machine actors continues to grow, it is essential to increase per-server performance to keep up with increasing demand.

Problematically, the looming end of traditional technology scaling presents a challenge for extracting further processor performance. Power constraints have largely flattened the improvement in single-thread performance over the past decade. Meanwhile, the slowdown in Moore's Law combined with skyrocketing manufacturing costs for leading-edge technology nodes [9] spell an approaching end to growth in core count. These trends motivate the need to look beyond traditional chip-multiprocessor (CMP) architectures to mine further performance and efficiency.

The data-intensive services running on the servers put an enormous pressure on the server memory subsystem. The unrelenting growth in data set sizes demands a commensurate expansion of memory and cache capacity. Large on-chip cache capacities attempt to capture the application working sets in order to prevent frequent long latency main memory accesses. At the same time, datacenter servers run user-facing latency critical services where high server performance is necessary for a good user experience. High server performance relies on fast on-chip cache accesses.



(a) Intel Sapphire Rapids die (single tile) with 15 cores and 1.875MB LLC per core, 28.125 MB aggregate shared capacity [11]



(b) AMD Zen3 die (core complex/CCX) with 8 cores and 4MB LLC per core, 32MB aggregate shared capacity [12]

Figure 1.1: Die area consumed by LLC in modern servers.

1.1 Scaling Challenges for On-Chip Server Caches

Servers are equipped with a multi-level on-chip cache hierarchy to keep frequently used data close to the operating cores and filter frequent long latency accesses to main memory. On-chip caches, typically made up of SRAM cells, are hierarchically arranged, where the first few cache levels are private to the cores, backed by a large last-level cache (LLC) shared by the cores. The LLC occupies a big proportion of the die area, up to 30-40% in modern processors, similar to the die area occupied by cores. Figure 1.1 shows die photos of two recent server processors with the large die area consumed by the LLC labelled.

SRAM scaling is becoming increasingly challenging, especially below the 7nm technology node where SRAM is scaling considerably differently than logic [10]. In fact, SRAM cell size in TSMC's 3nm node is nearly the same as in the 5nm node, as shown in figure 1.2. Even a specialised 3nm SRAM variant is only 5% smaller than the 5nm SRAM cell [2]. To make matters worse, poor yields in the leading technology nodes limit processor die sizes, making die real-estate scarce, and limiting LLC capacity. Therefore, the on-chip LLC capacity in modern processors cannot grow in line with the growing dataset sizes of data-intensive services. In the face of on-chip cache capacity constraints, systems today are starting to be equipped with DRAM caches, backing the on-chip cache hierarchy.

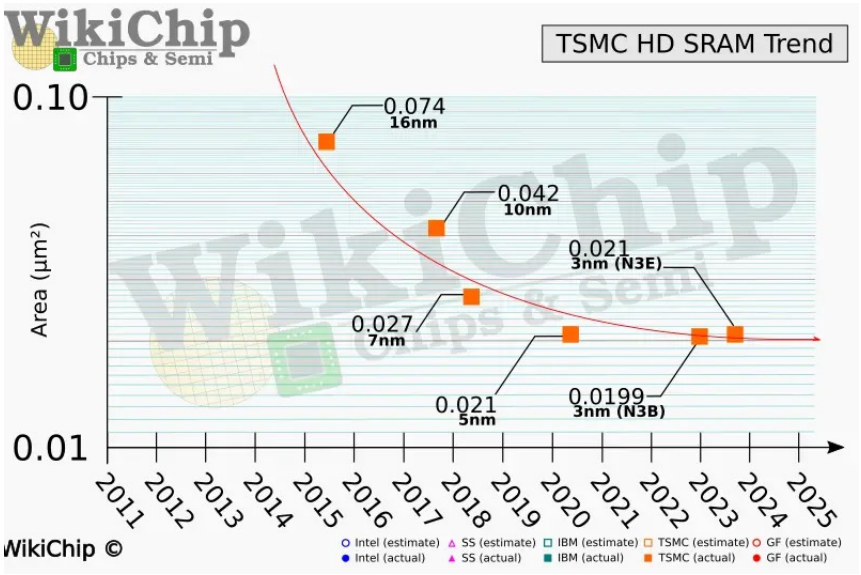


Figure 1.2: SRAM scaling trend for TSMC from [2].

1.2 Problem

Commercial multi-gigabyte DRAM caches leverage 3D DRAM stacks provisioned on package [13, 14]. DRAM stacks comprise of multiple through-silicon-via (TSV)-connected layers of densely-packed DRAM cells, divided into a large number of banks, and a wide output interface. These features specifically target bandwidth optimisation in order to alleviate the memory bandwidth bottleneck for data-intensive operations common in scientific and high performance computing (HPC) applications [15]. Training of deep learning models is also an extremely bandwidth hungry application which typically runs on GPUs and relies on special high-bandwidth memory on-board the GPU to keep the processing elements fed with input training data. However, the bulk of datacenter applications do not saturate the main memory bandwidth as shown by prior work [16, 17], and thus do not benefit from the additional bandwidth.

Datacenter applications are performance sensitive to cache access latencies due to their low memory-level parallelism (MLP) [16, 18]. Existing DRAM cache architectures have high access latencies that are on par with main memory [13, 19, 20]. Therefore, datacenter applications fail to extract performance benefits from high-capacity DRAM caches.

With the scaling of on-chip caches breaking down, die-stacked DRAM provides a capacity-viable solution for scaling caches to meet the demand of growing datasets. However, without considerable latency improvements, DRAM stacks

are rendered unsuitable for use as caches in datacenter servers. This calls for recognising the long-latency contributors in accessing DRAM stacks and addressing them. With stacking technology maturing and die-stacked components becoming commonplace in modern systems, it is imperative to optimise DRAM cache latency in order to build upcoming systems.

1.3 Contributions

The aim of this thesis is:

"To identify the latency sources in conventional DRAM caches and devise lower latency DRAM cache organisations addressing them".

1.3.1 Sources of Latency in DRAM Caches

Current die-stacked DRAM caches have access times similar to that of main memory DRAM because they are designed around capacity-optimised commodity DRAM technology and are accessed through a controller interface shared by multiple cores. We observe that the factors contributing to the high access latency of stacked DRAM caches are:

- on-chip interconnect routing delay to reach the DRAM cache controller,
- queuing delay in the DRAM cache controller,
- horizontal traversal between the processor die and the memory stack,
- addressing and decoding delay on the stacked DRAM core, followed by DRAM bank access latency.

Our insight is that none of these problems are fundamental and can be readily addressed at the architecture level.

1.3.2 On-Package Partitioned DRAM Victim Cache

We propose an On-package Partitioned DRAM Victim Cache - CARVE - in which the on-chip LLC is supplemented by a victim cache in DRAM. To minimise the long interconnect delays the stacked DRAM is partitioned into functionally-independent DRAM vaults each of which is deployed as a victim cache for a distributed on-chip LLC slice. In this manner, the DRAM vaults follow the address interleaving scheme of the shared LLC, and the aggregate vaults capacity

is logically shared. Each LLC slice has a dedicated DRAM cache controller on chip for its corresponding DRAM vault, thus eliminating any considerable controller queuing delay. Additionally, a custom DRAM core, engineered for low latency is leveraged for the DRAM vaults.

This design retains the conventional on-chip cache hierarchy, and augments it with a new level of victim cache in logical DRAM vaults on package. As the DRAM vaults sit logically below the shared on-chip LLC, they do not introduce any additional coherence implications.

1.3.3 Die-Stacked Private DRAM LLC Organisation

We propose a novel Die-Stacked Private LLC Organisation – SILO – which dispenses with a shared on-chip LLC, and combines on-chip private caches with per-core LLC slices in die-stacked DRAM. To avoid long interconnect delays and maintain the latency benefits of a private cache, SILO organises the DRAM into vaults, each of which sits above a processor core. Each vault has its own memory controller and is completely independent of other vaults in data storage and access. The DRAM core is optimised for latency, at the expense of capacity, to further reduce the access time to the LLC vaults.

The private LLC vaults are kept coherent through a conventional directory-based protocol with in-DRAM metadata. The high hit-rate of large, private DRAM caches and the use of low latency DRAM for storing metadata makes directory accesses not detrimental to performance.

1.4 Published Work

During my PhD programme, I contributed to the publications listed below.

- **A. Shahab**, M. Zhu, A. Margaritov, B. Grot. "Farewell My Shared LLC! A Case for Private Die-Stacked DRAM Caches for Servers". In Proceedings of the 51st International Symposium on Microarchitecture (MICRO). 2018 [1]

- M. Zhu, **A. Shahab**, A. Katsarakis, B. Grot. "Invalidate or Update? Revisiting Coherence for Tomorrow's Cache Hierarchies". 30th International Conference on Parallel Architectures and Compilation Techniques (PACT). 2021 [21]

- A. Margaritov, D. Ustiugov, **A. Shahab**, and B. Grot. "Ptemagnet: Fine-grained physical memory reservation for faster page walks in public clouds." In Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2021 [22]

- **A. Shahab**, B. Grot. "Population-based evolutionary distributed SGD". In Proceedings of the 2020 Genetic and Evolutionary Computation Conference (GECCO) Companion. [23]

1.5 Thesis Organisation

The remainder of this thesis is organized as follows. Chapter 2 provides the background and discussion about the cache hierarchy in server processors, with a focus on high-capacity DRAM caches. Chapter 3 analyzes the sources of latency in conventional DRAM caches. Chapter 4 presents an on-package DRAM cache design that cuts down interconnect traversals to access the DRAM cache, thereby reducing the overall access latency. Chapter 5 presents an LLC organisation using stacked DRAM utilised as a per-core private LLC, dispensing with the traditional on-chip shared LLC. Finally, chapter 6 concludes the work and talks about future directions.

Chapter 2

Background

This chapter covers the relevant background material essential to understand the motivation and contributions of this thesis.

2.1 Cache Requirements of Server Applications

Datacenter servers run latency-critical, user-facing services that operate on large datasets to respond to user requests. Expectedly, the server memory subsystem takes centre stage in delivering a good user experience. With the main memory observing nominal access latency improvements for more than a decade, caches play a vital role in ensuring that required data is served in a timely manner. We now discuss the characteristics and trends of datacenter workloads and the resulting cache requirements they impose.

2.1.1 Large Cache Capacity for Growing Datasets

The amount of data being produced and consumed has been rising exponentially. Data volume grew 5x from 2018 to 2022 [24]. IDC predicts an average acceleration of 26% per year in data volume growth, growing up to 175 zettabytes in 2025 [25]. Even more importantly, the percentage of real time data that requires residence in memory and caches is growing rapidly, reaching up to 24% in 2024 [26]. The continual data growth demands an in line scaling of cache capacities to avoid frequent long-latency main memory accesses.

2.1.2 Fast Cache Access for High Application Performance

Delays in online service responses e.g., search results, leads to a bad user experience, which negatively affects both short-term and long-term usage by users as observed by Google and Bing [27]. Google's experiment observed a 0.2-0.6% drop in searches conducted per user when the search results were displayed in 400ms as opposed to 100ms. Additionally, prolonged exposure to long delays deters users increasingly. In the Google experiment, users observing a consistent 400ms delay did 0.44% fewer searches during the first three-week period and 0.76% fewer searches in the following three-week period. Worryingly, even after the elimination of the additional delay, these users only went up to 0.21% fewer searches than the pre-delay levels [28, 29]. These figures carry serious negative impacts for large-scale online services like web search in the form of significant revenue loss. Therefore, online services demand fast responses to data requests, predominantly from processor caches, in order to achieve high performance and user satisfaction.

2.2 Cache Hierarchy in Servers

2.2.1 On-Chip Caches: Limited Capacity and Increasingly Slow Accesses

Modern server processors are equipped with a hierarchical on-chip cache organisation, typically spanning three levels. In this hierarchy, the first two cache levels are private to the cores and are positioned close to the cores. The last level cache (LLC), is shared by the cores in order to maximize effective capacity. Owing to the size of the LLC, it is split into slices and distributed across the processor die. An on-chip network (NOC) such as a mesh or ring connects the LLC slices, which employ a non-uniform cache access (NUCA) organisation.

SRAM cells, which are compatible with CMOS logic process, make up on-chip caches. SRAM arrays provide modest capacities with fast access times close to the processor cores. SRAM technology has low density: 6 logic transistors make up a single SRAM cell, and therefore the high area cost limits the cache capacity that can be afforded on chip. CMOS-logic-compatible embedded-DRAM (eDRAM) has also been used to provide on-chip LLCs [30], which is denser than SRAM and consequently, allows higher LLC capacity on chip. However, available die area

fundamentally constrains cache capacity. On-chip LLC capacities have observed only modest improvements across processor generations in the last decade, as shown in Figure 2.1.

AMD has pioneered stacking SRAM directly on top of the compute die to boost LLC capacity in the Milan-X servers [31]. These server offerings, targeted at HPC and AI workloads, extend the on-chip LLC capacity with an additional layer of SRAM (3D V-Cache), effectively tripling the shared LLC size [31]. Reportedly, servers equipped with 3D V-Cache significantly outperform their non-V-Cache counterparts. However, SRAM density and manufacturing cost limits the scaling of LLC capacity even in the 3D V-Cache approach. Figure 2.1 shows the two generations of AMD servers that employ 3D V-Cache. Due to area footprint constraints, both server generations have the same 3D V-Cache capacity.

At the same time, the distribution of the LLC slices across the processor incurs long on-chip wires. Routing an LLC request over the long wires adds delays to the access latency, proportional to the length of the wires. Large processor die have long interconnect spans which add significant delays, making overall LLC accesses slow. While large server processor die are starting to be split into small chiplets (or tiles) to improve manufacturing yields (and in turn costs), inter-chiplet communication, when incurred, still experiences traversal over long interconnect wires, adding significant delays on the cross-chiplet requests path. In fact, inter-chiplet communication complicates NOC design, further exacerbating the latency problem.

2.2.2 On-Package DRAM Caches: High Capacity and Very Slow Accesses

In the face of tight die area constraints, there has been a shift towards leveraging die-stacking technology for caches. Die-stacking technology allows dense packing of DRAM cells in multiple layers, providing multi-gigabyte capacity within a small area footprint. The resulting DRAM stack may be integrated on the package and connected to the processor die through a dedicated memory controller. The DRAM stack may serve as an extension to the main memory or as an intermediary cache between the main memory and the on-chip cache hierarchy.

The DRAM stack consists of multiple DRAM die stacked on top of each other, tightly packed to form a stack height of less than a millimeter. The total

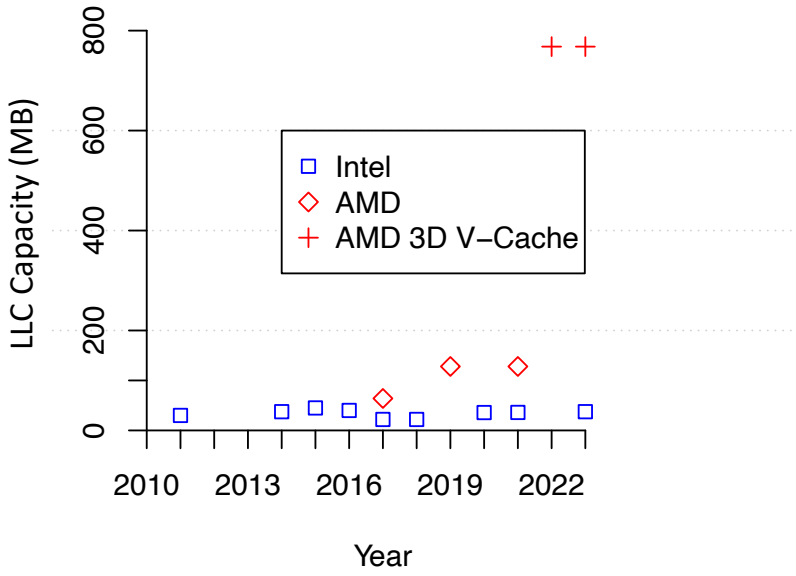


Figure 2.1: LLC capacity over processor generations for 16-core servers with TDP under 200W. TDP for AMD 3D V-Cache equipped servers is 320W [3].

stack capacity is the aggregate of all the individual DRAM die. The DRAM stack capacity depends on the density of the DRAM die and the number of DRAM layers. With die-stacking technology maturing, stack capacities have been improving consistently. The JEDEC Solid State Technology Association supports one such DRAM stack standard, HBM, (details in the following section). Figure 2.2 shows the trend of massive improvements in stack capacity supported by the different generations of the JEDEC standard [1].

Unfortunately, caches based on DRAM stacks have high access latencies that are on par with main memory [13, 32]. The access latencies to existing DRAM caches are high for two reasons: (1) significant interconnect delays incurred both on the CPU die (when routing to and from the DRAM cache interface) and on the DRAM die when accessing the target bank; (2) use of capacity-oriented DRAM architectures, which favor area efficiency over access latency.

[1]The stack capacities of commercial products based on the HBM generations are lower than the maximum supported, based on customer demand and use cases. However, the trend of stack capacity in HBM products across HBM generations follows an upward trend, similar to the maximum stack capacity supported by each HBM generation.

2.3 Understanding Stacked DRAM Caches

2.3.1 Stacked DRAM

Technological constraints limit the capacity of DRAM stacks to tens of GBs, far below the main memory capacity level – up to TBs, for most use cases. Stacked DRAM is thus suitable as a memory-side cache (or as on-package memory for systems running bandwidth-hungry applications such as [33]). There are two major stacked DRAM solutions: High bandwidth memory (HBM), and Hybrid memory cube (HMC), which have been deployed in commercial products. Their market demand is expected to be worth roughly $5 billion by 2026, with an annual growth rate of 30% [34].

HBM: The HBM standard, provided by JEDEC, comprises multiple layers of highly-banked DRAM chips in a tightly-packed vertical stack. HBM is not a stand-alone solution but is intended to be used in conjunction with the processor die. The DRAM stack may be connected with the processor die via a silicon

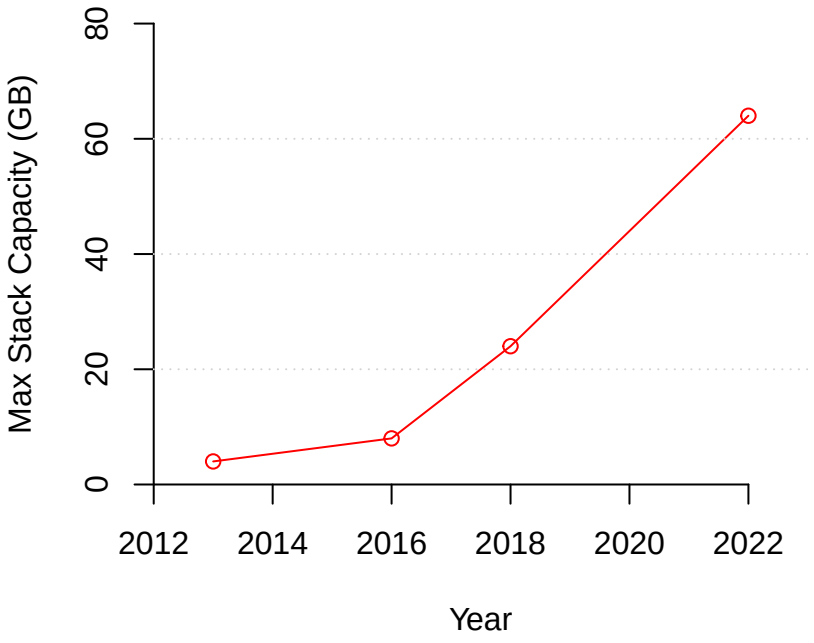


Figure 2.2: DRAM stack capacity supported over JEDEC specification generations.

interposer, an embedded multi-die interconnect bridge, or an elevated fanout bridge. HBM features in Intel's Xeon CPU Max line of servers [14].

HMC: HMC is envisioned as a discrete chip to which the processor offloads operations over a packetized interface. HMC consists of multiple highly-banked DRAM layers stacked above a logic die, vertically partitioned into independently operating units called "vaults". HMC-based DRAM caches feature in Intel's Xeon Phi Knight's Landing (KNL) as multi-channel DRAM (MCDRAM) [13]. As of 2018, Micron, the leading HMC-maker, shifted away from the HMC and refocused efforts on other high performance memory technologies [35]. Lessons learned from the HMC efforts, reportedly, continue to inspire path-finding projects [36].

Others: Tezzaron's Disintegrated RAM (DiRAM) employs transistor-level 3D stacking to create a specialized, high-performance memory stack which can be incorporated with a processor [37]. It connects multiple layers of storage cells with an underlying layer containing all the control peripheral circuitry. Through aggressive wafer thinning techniques, DiRAM achieves considerably lower stack heights compared to HBM or HMC. MonolithIC 3D DRAM stack dispenses with the capacitor-based DRAM in favour of a capacitor-less DRAM cell, achieving 3.3x the density of capacitor-based DRAM, while using roughly the same number of lithography steps [38].

We will focus our discussion on HBM due to its wide commercial adoption.

2.3.2 Physical Layout

Figure 2.3 is a basic depiction of a system equipped with HBM. The main components of the HBM are as follows:

DRAM die

The DRAM layers use conventional DRAM architecture, with both storage cells and peripheral logic (sense amplifiers and decoders) in each layer. The die are stacked vertically and connected by TSVs. Each DRAM die has a central I/O lane for all the TSV connections.

Logic die

A logic layer lies underneath all the DRAM die which hosts the physical interface between the stack and the processor-side cache/memory controller, I/O buffers, and test logic. The logic layer also has a central I/O lane.

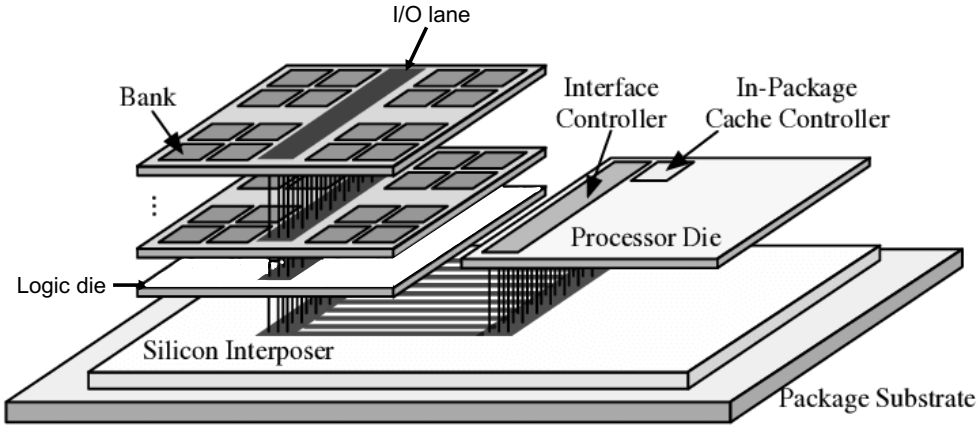


Figure 2.3: Detailed view of HBM from [4] with slight modifications.

I/O lane

The central I/O lane houses three kinds of TSVs: for DRAM column and row commands, for data transport, and power.

2.3.3 Connection to the Processor Die

Figure 2.3 illustrates the processor die and DRAM stack connections. The main components are as follows:

Interposer/Bridge

A silicon interposer may be used to package and connect the processor die and the DRAM stack(s) through fine-pitch metal wires. A silicon interposer may be active (connections and logic) or passive (connections only). A passive interposer may be used for this purpose. An interposer is a large silicon die, typically manufactured in an older and cheaper technology node, with wires running through it. The interposer should be strictly larger than the aggregate area footprint of the processor die, the DRAM stack(s), and the connection overheads. To cut down on silicon cost, an embedded silicon bridge may be used. A silicon bridge is a small segment of silicon embedded in the package substrate. Unlike an interposer, an embedded silicon bridge is required only in the area where the die-to-stack connections are placed, cutting down massively on the silicon cost. Additionally, embedded silicon bridges do not have size constraints as the interposer. However, embedding a silicon bridge into the substrate may add steps to the manufacturing process. Alternatively, an elevated fanout bridge solution, such as AMD EFB [39], may be employed which builds above the substrate instead, cutting

down on complexity and cost. Connections through an organic substrate, similar to what AMD uses to connect chiplets [40, 41], provide low connection density and bandwidth, rendering them unsuitable. Integrated fanout technology on substrate is also being considered to provide fine-pitch connections between adjacent chips through horizontal redistribution layers [42].

We will focus our discussion on interposer-based integration.

Microbumps and wires

A die connecting to an interposer (or bridge) requires a set of microbumps. The microbumps serve as the endpoints for the fine-pitch connecting wires running through the interposer. The number of microbumps determine the width of the interface connection. The microbumps are separated by a minimum distance – pitch, a manufacturing technology constraint.

DRAM Cache Controller

The DRAM cache controller(s) is integrated on the processor die. In case of multiple DRAM stacks, (at least) a cache controller per stack is required. An on-chip LLC miss is forwarded to the DRAM cache controller which communicates with the DRAM stack.

2.3.4 Cache Characteristics

Conventional DRAM caches leverage stacked DRAM to provide multi-gigabyte cache capacities. The DRAM stack capacity is proportional to the number of DRAM layers and the density of the DRAM die. The DRAM die are packed tightly together vertically and connected to the processor die through a wide input/output interface. The combination of highly banked DRAM die and a wide interface provides high bandwidth (hence the name HBM). The DRAM die used in the stack use commodity DRAM technology – optimised for cost-per-bit – and experience access latencies similar to that for main memory.

2.3.4.1 Organisation

The DRAM cache is shared by the cores that enjoy the entire cache capacity without any data replication. DRAM caches may be block-based: fine-grain cache block size (64B) [43, 44], or page-based: coarse-grain block size (1KB to 8KB) [15, 45]. Block-based caches leverage temporal locality over spatial locality, maximising available cache capacity and reducing off-chip bandwidth due to its

fine granularity. However, the fine-grain block size results in a sizeable tag overhead and missed opportunity for exploiting spatial locality. Page-based caches favour spatial locality over temporal locality, and have a relatively modest tag overhead. However, page-based caches are more prone to cache thrashing and consume much larger off-chip bandwidth. DRAM caches may be direct-mapped or set-associative. Direct-mapped caches result in low lookup latency as only a single cache entry is read to service a cache request. Cache conflicts are more probable in direct-mapped caches. Set-associativity helps circumvent a vast majority of cache conflict but result in a higher lookup latency due to multiple cache entries being read per request.

2.3.4.2 Management

To keep the DRAM cache software-transparent, i.e., no changes required in existing software, the DRAM can be hardware-managed just like on-chip caches. Hardware cache management may be more complex and may require significant overhead. Software cache management may be simpler but necessitates modifying already-existing deep software stacks.

2.4 Summary

In this chapter we discussed the limitations of on-chip caches in meeting the requirements of datacenter applications. We then identified the ability of on-package stacked DRAM caches to scale to tens of gigabytes of capacity and studied their internal structure and organisation in detail. On-package stacked DRAM caches provide high capacity through densely-packed DRAM die, high bandwidth through a large number of banks and a wide interface, but experience access latencies similar to that of main memory. It is important to understand the factors that lead to long access latencies in on-package stacked DRAM caches and if the sources of latency are fundamental.

Chapter 3

Optimising Stacked DRAM for Low Latency

In this chapter we show that today's high latency DRAM caches yield negligible system performance improvements, motivating the need to investigate the factors contributing to their slow access. We then break down the sources of latency in today's stacked DRAM caches. Following that, we present optimisation techniques targeting the latency-contributing factors, targeting both connections to the DRAM stack and the DRAM core within the stack.

3.1 Performance Impact of DRAM Cache Access Latency

Conventional stacked DRAM caches provide multi-gigabyte capacities at a high access latency. We attempt to understand the performance gains offered by the high capacity and high access latency combination of conventional DRAM caches for cache latency-sensitive datacenter applications. We run our study on a 16-core setup, with 3-way OoO cores running at 2.0GHz. Table 3.1 details the system parameters. The baseline system uses a three-level on-chip cache hierarchy. We augment the baseline with a conventional HBM-based DRAM cache – baseline+DRAM$. The DRAM cache is hardware managed and uses a page-based arrangement considered state-of-the-art for servers [15, 45]. A page-based arrangement aids in exploiting spatial locality as discussed in section 2.3.4.1. The DRAM cache is direct-mapped to limit the number of cache lines read per

Processor	16-core, 2GHz, 3-way OoO, 128 ROB, ISA: UltraSPARC v9
L1-I/D	64KB, 8-way, 64B line, 3 cycles, private, stride data prefetcher
L2	512KB, 8-way, 64B line, 5-cycle, private, stride prefetcher
Interconnect	4x4 2D mesh, 3 cycles/hop
Baseline on-chip LLC	32MB shared NUCA, 7 cycles, 16-way, 64B line, non-inclusive MESI, LRU
Conv. DRAM cache	8GB, page-based, 2KB pages, direct-mapped, access latency 40ns, closed page policy
Main memory	Access latency 50ns, closed page policy

Table 3.1: Microarchitectural parameters of the simulated systems. Extended parameters table in appendix A.

lookup to one. While conventional DRAM caches have latencies similar to that of main memory as we mentioned earlier [13, 32], we use an optimistic representation of a conventional DRAM cache which is 20% faster than main memory, similar to a prior work [20]. We run our evaluation on scale-out workloads from Cloudsuite [46] and parallel applications from Parsec-3.0 [47].

Figure 3.1 presents the results of this study, with results normalized to the baseline. We observe that the addition of a conventional DRAM cache provides almost no system performance gains, with `Web Frontend` and `canneal` achieving the maximum gain of 2%. The large capacity of the conventional DRAM cache yields no performance benefits due to the high access latency. These observations necessitate investigating the sources of latency in conventional DRAM caches and address them to unlock higher system performance with DRAM caches.

In the following section, we divide the end-to-end latency into its various components and discuss each one separately.

3.2 Access Latency Breakdown

In this section, we investigate the factors contributing to the long delays in accessing current die-stacked DRAM caches. Figure 3.2 shows the request access path from the LLC slice to the DRAM bank holding the required data and ta-

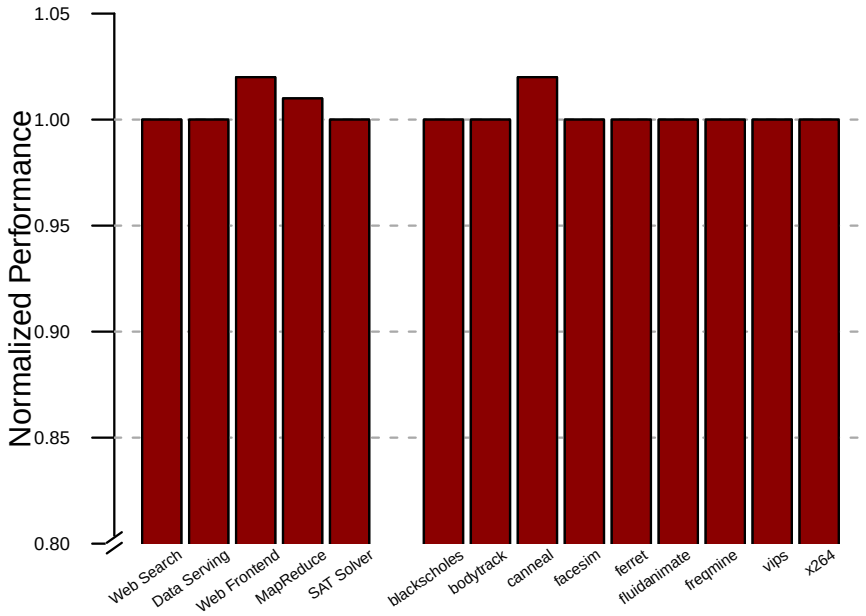


Figure 3.1: System performance of baseline+DRAM$. Performance is normalized to a system without a conventional DRAM cache.

ble 3.2 presents the typical range of values for each factor. We will now look at each factor in detail.

3.2.1 NOC Traversal on the Processor Die

Upon a request miss in an LLC slice, the request is routed to the DRAM cache controller interface over the NOC. In a mesh-based NOC, the number of hops is determined by the LLC slice and the DRAM cache controller placement. Subsequently, once the DRAM cache returns the requested block, the block needs to

NOC traversal	9-18ns
Controller	20-90ns
Die to stack interconnect	2-10ns
DRAM addressing and access	10-30ns

Table 3.2: Typical latency value ranges.

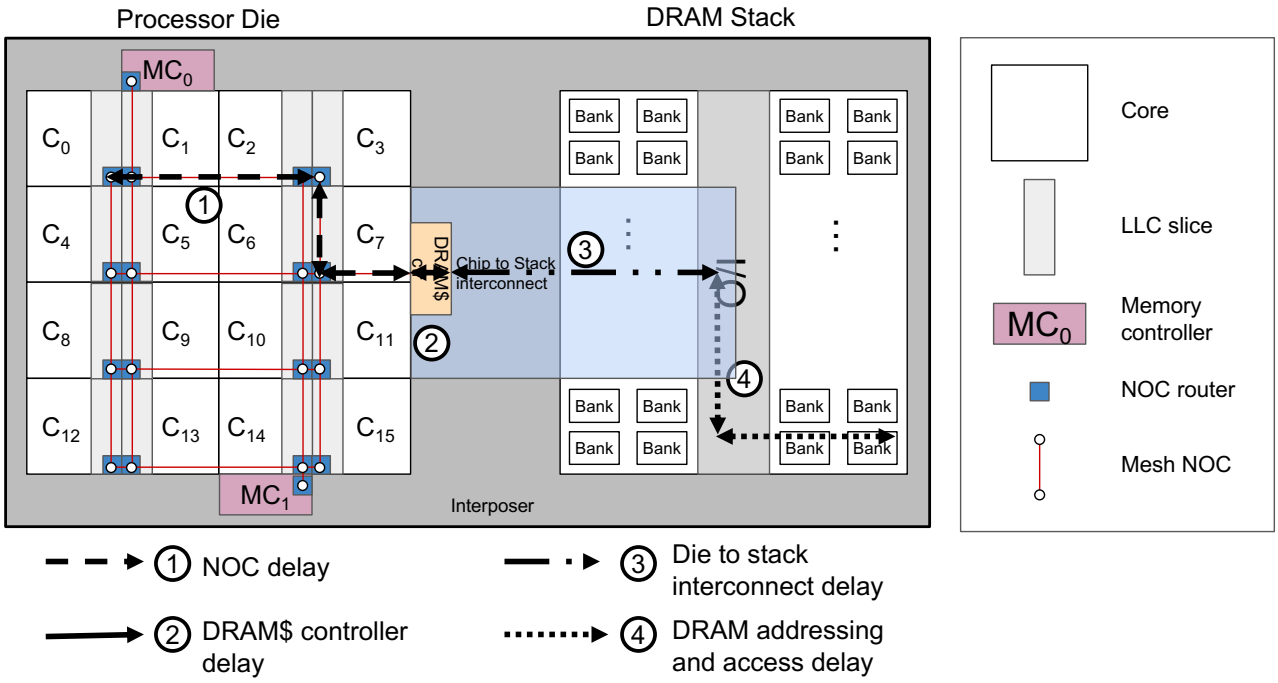


Figure 3.2: Access path from processor die to DRAM stack.

be communicated to the requesting core and the responsible LLC slice. In a 4x4 mesh NOC, a request incurs 3 hops on average. An access request missing in an LLC slice incurs 3 hops to reach the DRAM cache controller and then 3 more hops on the return trip, adding up to 6 hops on average (9ns at 2GHz). In the worst case, the round trip incurs a total of 12 hops (18ns at 2GHz)[1].

3.2.2 Cache Controller Delay

The DRAM cache controller is a shared interface to which all LLC miss requests are sent. With a high LLC miss rate the controller receives frequent requests and the queues build up. The queuing delay adds to the unloaded controller delay which already consumes numerous cycles. In fact, recent work shows that queuing at the controller constitutes most of the HBM latency, accounting up to 90ns in the worst case with random memory traffic [48]. Figure 3.2 presents the simple case with a single DRAM cache controller. Having multiple DRAM cache controllers will reduce the queuing per cache controller, but as long as a controller

[1]Note that this is a simplistic arrangement with a single DRAM cache controller for ease of understanding. In the case of a distributed and address aligned DRAM cache interface, *e.g.* , two channels per row, two East-West links per row (to match the North-South ones), and same address interleaving (*e.g.* , LLC slice misses in the first row are address mapped to only the two DRAM cache channel in that same row), the round trip would consume 3 hops on average.

is shared by a number of cores/LLC slices, it will experience significant queuing delay.

3.2.3 Die-to-Stack Interconnect Delay

Requests between the processor die and DRAM stack cross the boundary twice on the round trip. In the case where the processor die and the DRAM stack are connected through a slow organic substrate (similar to what AMD uses to connect chiplets [40, 41]), the latency across the substrate is estimated to be around 10ns [49]. Our discussion focuses on the processor die and DRAM stack connected through a silicon interposer.

We estimate the die-to-stack latency through the available technology parameters. The boundary crossing constitutes passing the microbumps and the interposer. Microbump capacitance imposes a signal traversal delay on each end, estimated to be within a fraction of a nanosecond by prior works [50, 51]. Signal transmission delay through interposer wires is similar to on-chip wires, with prior literature estimating it in the range 125-250ps per mm length [51, 52]. DRAM stacks are generally placed close to the processor die on the interposer, with a 100 μm minimum gap constraint imposed by technology. The distance over which the request travels the interposer wires is the distance between the corresponding microbump array pair. This length is estimated to be around 7mm for HBM [53]. The round trip incurs about 2ns. In total, signal transmission through the microbumps and the interposer incurs 3-4ns on the round trip. A non-simplistic communication protocol between the chip and the stack may further add to the transmission delay.

3.2.4 DRAM Addressing and Access Delay

HBM features a central I/O lane through which all requests are received and sent. Once a request arrives at the I/O lane, the address decoding logic determines which DRAM bank is responsible for serving the request. The address decoding and the subsequent forwarding of the request to the target bank incurs a significant delay. The exact delay depends on the dimensions of the DRAM stack and the distance of the target bank from the I/O lane. Reading a block from a DRAM bank requires activating a DRAM row and reading the row contents into a row buffer, incurring latency.

3.3 Exploring Latency Reduction Methods

In the previous section, we identified the components of DRAM cache access latency. These fall broadly into two categories: (1) the interconnect: the connections from the processor die to the DRAM stack; and (2) the DRAM core: addressing and bank access within the DRAM. We will now investigate their respective latency-reduction techniques.

3.3.1 Interconnect

Long wire traversals and shared interfaces when accessing a conventional DRAM cache elevate the access latency. We will explore how the traversal paths may be shortened and the shared inteface may be avoided.

3.3.1.1 Limiting NOC traversals on the processor die

Requests destined for the DRAM cache need to be routed to the DRAM cache controller. Given that there is a single or a few DRAM cache controllers, this incurs a multi-hop path over the mesh NOC. If the responsible DRAM cache controller is placed right next to the on-chip cache where the request misses, no NOC traversals would be required. In order to achieve that, we require a DRAM cache controller placed in each core tile.

While a per-core tile DRAM cache controller is necessary to cut down on NOC traversals, each DRAM cache controller also requires a direct interface to the DRAM stack. This may be achieved through a combination of TSVs and microbump array for each controller, placed next to it.

3.3.1.2 Minimizing cache controller delay

A request queue in a single DRAM cache controller which serves the entire address space of the DRAM stack is bound to grow significantly when the on-chip cache miss rate is high. Also, any request reordering logic employed to optimise the processing of the queue will add complexity and hence will add processing cycles. Instead, the DRAM stack may be logically partitioned into independent units. Each unit may then be accessed by its own cache controller, with each controller receiving a fraction of the access requests compared to a single controller. This helps prevent the request queue size growth at any controller and limits

queuing delays. While partitioning the DRAM stack into any number of units may be beneficial, matching the number of units to core tiles simplifies controller placement.

3.3.1.3 Die-to-Stack Interconnect

Using a simplistic interconnect over a silicon interposer provides the lowest latency. Additionally, a set of interposer wires for each core tile helps avoid NOC traversals and congestion.

3.3.1.4 Summary

We find that fine-grain connections between the core tiles and independent units within the DRAM stack shorten the traversal path and sidestep shared interfaces, thus yielding the lowest latency. This scheme requires a cache controller, microbump+interposer connections, and an independent DRAM unit within the DRAM stack.

3.3.2 DRAM Core

While DRAM capacity and bandwidth have improved greatly over the past two decades, DRAM latency improvements have been modestly low. To find the root cause of long latency in DRAM, we look into the internal design and organisation within DRAM chips. We then analyze the effect of these design parameters on DRAM latency.

3.3.2.1 DRAM Technology Basics

Today's DRAM chips are comprised of DRAM cells and peripheral circuitry organized in a hierarchical structure as shown in Fig. 3.3. At the top-most level, a DRAM chip is divided into *banks* where cells share peripherals, including row and column decoders. The column width of a bank is referred to as a *page*. A bank is further divided into *subarrays* in which cells are connected through a common bitline and share sense amplifiers. In turn, a subarray consists of a number of *tiles* which have common global wordlines. Each tile has local wordlines and drivers. The thick grey segments in Fig. 3.3 represent the peripherals; vertical segments include wordline drivers and horizontal segments are sense amplifiers.

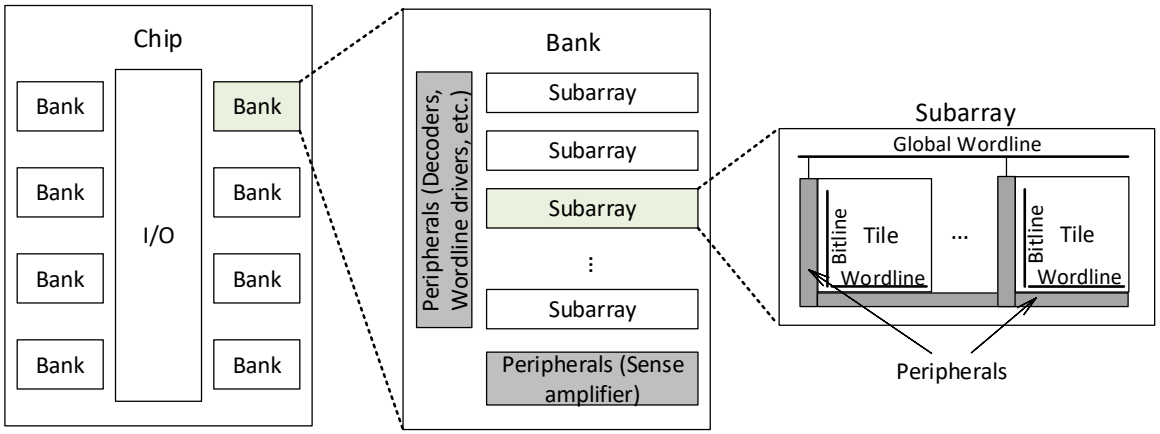


Figure 3.3: DRAM internal design.

Tile dimensions determine the lengths of bitlines and local wordlines (subarrays and tiles have the same bitline length but different local wordline length as shown in Fig. 3.3). The electrical load observed on the lines is proportional to their lengths, resulting in higher transmission delays for longer lines. At the same time, longer bitlines and wordlines require less peripheral circuitry such as sense amplifiers and wordline drivers. Thus, while longer lines are good for area efficiency (defined as DRAM cell area divided by total chip area), they are bad for latency. Conversely, shorter lines reduce the electrical load but require more peripheral circuitry. The choice of line lengths is governed by design optimisation targets.

3.3.2.2 Commodity DRAM Designs

Commodity DRAM products are designed to minimize cost-per-bit. This design target affects each level of the DRAM hierarchy. Firstly, DRAM manufacturers choose to limit I/O and peripheral circuitry by sharing I/O between banks, thus allowing more chip area for DRAM cells. Secondly, in order to minimize the area of row and column decoders, a small number of banks is employed (e.g., 8 banks per chip in DDR3). Next, a bank is divided into only a small number of subarrays to minimize the area occupied by subarray-level sense amplifiers. Reducing the footprint of sense amplifiers is important for a density-optimised design because a sense amplifier can be 100 times larger than a DRAM cell [54]. Lastly, a subarray comprises of only a few tiles to reduce the number, and hence the area footprint, of the local wordline drivers. Fewer subarrays and tiles improve DRAM

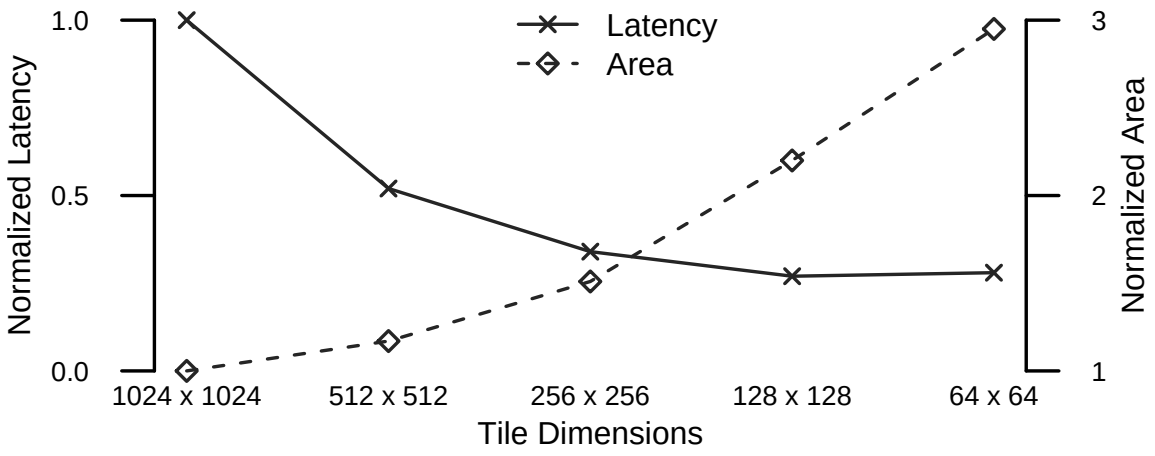


Figure 3.4: Effect of DRAM tile dimensions on access latency and area.

area efficiency but lead to longer bitlines and wordlines, which naturally increases latency as discussed above. In effect, tile dimensions determine the access latency of a DRAM core.

3.3.2.3 Effect of Tile Dimensions on DRAM Latency and Area

To quantify the effect of tile dimensions on DRAM area and latency we model a 1Gb DRAM die (details can be found in Sec. 5.4.3). In order to reduce the line lengths, we change tile dimensions by using a combination of DRAM parameters, namely: number of banks, page size, number of divisions per bitline (Ndbl) and number of divisions per wordline (Ndwl). Smaller tile dimensions correspond to shorter bitlines/wordlines.

Fig. 3.4 plots area and access latency as a function of tile dimensions. The values are normalized to a baseline design based on Micron DDR3 [55] having tile dimensions 1024x1024 DRAM cells. We observe that reducing the tile dimensions from the baseline 1024x1024 down to 256x256 decreases the access latency by 64% at a cost of a 49% increase in die area. Beyond that point, a mere 6% drop in access latency achieved with a tile size of 128x128 comes at a hefty 150% increase in area. Thus, we conclude that up to a certain point, reducing tile dimensions (and thus, increasing the number of tiles/subarrays for a fixed area) is effective in trading off capacity for latency, which makes for a valuable optimisation space. However, beyond that, small latency gains come at exorbitant area cost and are not justified.

3.3.2.4 Summary

We find that the latency to access the DRAM core may be considerably reduced by adding peripheral circuitry to effectively reduce line lengths, at the expense of area efficiency (and in turn, capacity).

3.4 Latency-Optimised DRAM Vaults

Based on our discussion from section 3.3.1, we propose organizing the DRAM stack into vaults inspired by the HMC [56] to reduce addressing delay. Each vault is a multi-die stack of DRAM banks with a dedicated vault controller in the logic layer at the base of the stack. Each vault has its own cache controller on the processor die, and is completely independent of other vaults in data storage and access. To reduce latency within a vault, we sacrifice area efficiency by using a latency-optimised custom DRAM core. Following the discussion from section 3.3.2.3, we introduce additional peripheral circuitry to effectively reduce line lengths in order to lower delays. This is achieved through the following optimisations:

- Large number of banks per vault to increase parallelism and minimize queuing at the memory interface.
- Shorter pages to reduce DRAM row size and hence global wordline length.
- Large number of subarrays per bank to reduce bitline length within a subarray.
- Many tiles per subarray to reduce wordline length.

Die Stacking and Thermal Feasibility. In general, the height of a DRAM stack is limited by thermal constraints. Up to 8 additional layers of DRAM have been shown to increase the temperature of the chip by only 6.5 degrees Celsius [57] and have a negligible effect on thermal distribution of the die [58]. Industrial specifications indicate feasibility of DRAM stacks with 12 layers and a logic die underneath [59], while 4- and 8-layered stacks are widely available in products [13, 59]. We will conservatively model a 4-layer DRAM stack in our study, but the trends observed will carry over to higher stacks.

We will now model the capacity and latency of a DRAM stack using the vault partitioning and DRAM latency reduction techniques.

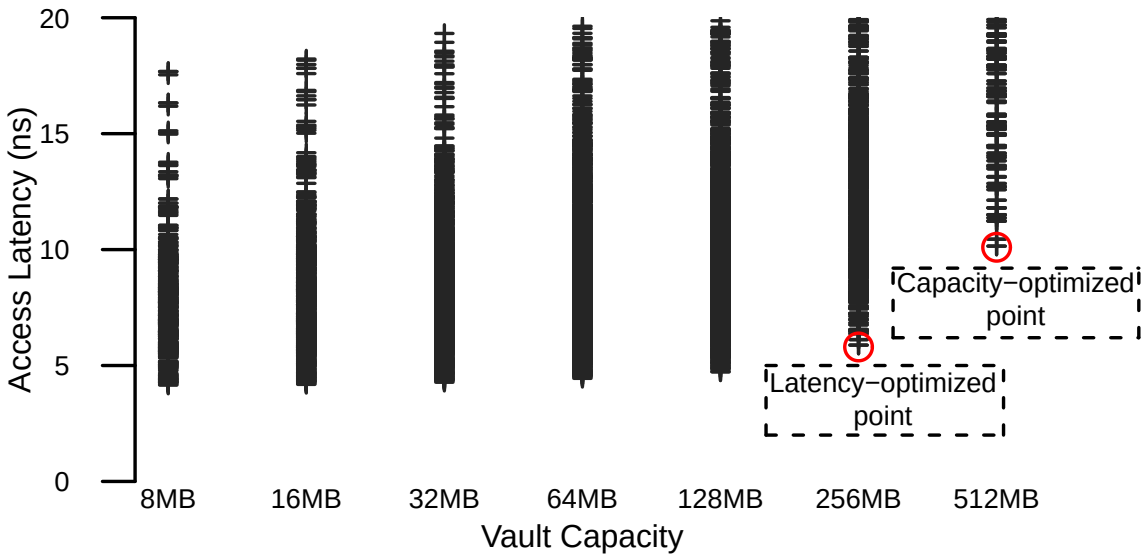


Figure 3.5: A scatter plot of vault capacity vs access latency as a function of DRAM parameters.

Mapping the Design Space. To model the area and timing for a stacked DRAM vault, we perform technology analysis in CACTI. We conservatively model a 4-die DRAM stack and assume a 5mm^2 area per vault, totalling up to a DRAM stack area of 80mm^2 (HBM2 stacks are reported to be around 90mm^2 [60]). Using these constraints, we perform a DRAM parameter sweep to find all possible vault designs that fit in the area budget. For each vault capacity we vary the following parameters: number of banks, page size, number of divisions per bitline, and number of divisions per wordline. For each combination of capacity and parameters, Cacti provides the vault access latency (time to read the first bit from a DRAM row assuming the wrong row was open). The resulting designs are plotted as capacity-latency pairs in Fig. 3.5[2].

From the figure, we observe that lower capacity designs fit easily in the area budget while maintaining low access latency. Moving from 8MB to 128MB, the capacity increases by 16x while the latency increases by less than 10%. Going from 128MB to 256MB, the capacity doubles at the cost of a 15% latency increase. From there, another doubling in capacity to 512MB results in an 80% increase in access latency. Thus, for the set of parameters considered in this study, we find the per-vault capacity of 256MB at a 5.5ns access latency to be the sweet spot for a latency-optimised design.

[2]This study is designed to obtain latency,capacity pairs but power consumption of the chosen design is discussed in chapter 5.

	Latency-optimised	Capacity-optimised
Area efficiency	1x	1.74x
Number of tiles	1x	0.25x
Vault capacity	256MB (1x)	512MB (2x)
Aggregate capacity	4GB (1x)	8GB (2x)
Vault access latency	5.5ns (1x)	10ns (1.8x)

Table 3.3: Comparison of latency- vs capacity-optimised vault design points. Values normalized to the latency-optimised point.

We further note that for a traditional DRAM cache, the higher-capacity (and higher latency) 512MB per-vault design point would be well justified, since the interconnect delays on the processor side and in the chip-to-chip interface would add tens of nanoseconds to the DRAM access latency. Given that, an additional 4.5ns of latency, which is the difference between the lowest-latency 256MB and 512MB design points, would amount to a modest fraction of the overall delay, pointing to 512MB as a sweet spot for off-chip DRAM (note that the access latency of the chosen 512MB capacity-optimised vault design point falls at the lower end of the DRAM addressing and access latency range presented in table 3.2). Table 3.3 highlights the key differences in the two designs.

Technology Scaling. As both SRAM and DRAM memory technology are experiencing a gradual slowdown in their feature scalability, exploiting vertical stacking to overcome constraints of traditional cache hierarchies is an attractive option explored in this work. But how well is DRAM stacking projected to scale in terms of capacity?

The number of layers, which ultimately determines the capacity of a die-stacked cache, is limited by two primary factors: thermals, which dictate the maximum height of the stack, and manufacturing technology (including testing and integration). Future improvements in wafer-thinning and integration technology will allow more dies to be integrated in a fixed-height stack, thus providing a viable path to higher capacities. Indeed, ITRS 2.0 roadmap [61] projects die thickness to shrink to 5-15µm in the next 10 years, from the current 50-100µm, allowing tens of stacked layers.

3.5 Summary

Based on our design and technology analysis, we note that there is potential for DRAM cache access latency reduction through careful interconnect layout and DRAM core latency optimisation. We arrive at latency-optimised DRAM vaults, logically partitioned independent units in a DRAM stack, comprising of latency-optimised DRAM technology. In the following chapters, we present two DRAM cache organisations which leverage the low latency DRAM vaults presented in the current chapter.

Chapter 4

On-Package Partitioned DRAM Victim Cache

In the previous chapter, we identified the sources of latency in conventional DRAM caches and showed that latency to access the on-package DRAM stack may be drastically reduced through partitioning the stack into independent vaults, each of which comprises of latency-oriented DRAM technology. In this chapter, we present a cache organisation which integrates latency-optimised DRAM vaults as a victim cache for each on-chip LLC slice.

4.1 Introduction

Modern datacenter applications operate on large datasets and have proportionally large application working sets. While the primary application working sets may be accommodated in the on-chip cache hierarchy, the secondary working sets tend to spill from the on-chip caches. This leads to frequent accesses to the on-package, memory-side DRAM cache if present, or to main memory itself, leading to data stalls. Prior work shows that datacenter applications exhibit low MLP which exposes the latency of misses in the higher level caches, making them performance sensitive to cache and memory access latency [16, 18]. As noted in the previous chapters, conventional DRAM caches have access latencies in the same range as main memory. A conventional DRAM cache, even with its high storage capacity, fails to alleviate the memory latency bottleneck and improve application performance.

In order for datacenter applications to benefit from the high capacity DRAM

caches they need to be organised in a manner that significantly reduces their access latency. In chapter 3, we devised latency-optimised DRAM vaults, accesses to which are considerably faster than conventional DRAM caches. However, unlike a conventional DRAM cache stack which is fully shared by all processor cores and accessed through a shared interface, the DRAM vaults are independent units, each of which need to connect to the processor die. The logical and physical arrangement of connections from the processor die to the DRAM vaults is not straightforward. To minimize the latency of serving a request from the DRAM vaults, only a single DRAM lookup must be incurred, and the request must follow the shortest path to the DRAM bank holding the data.

Figure 4.1 traces out the optimised access path between the processor die and the DRAM stack which offers the lowest access latency. We observe that in order to exploit the optimised access path, an on-chip miss [1] should only access the DRAM vault where the data (potentially) resides and needs to be sent to the DRAM vault without any NOC traversals. Therefore, the miss request needs to be forwarded to the target DRAM vault from the location where the final miss is discovered – the NUCA LLC slice. Our insight is that if the requested data may *only* reside in the DRAM vault connected to that LLC slice tile, the optimised access path may be realized.

In this chapter, we introduce *On-PaCkage Partitioned DRAM Victim Cache* (CARVE) – a DRAM cache design utilising latency-optimised independent DRAM vaults, each of which is connected to a core+LLC tile on the processor die. Based on our insight, CARVE uses the DRAM vaults as logically shared (from the cores' perspective), and each DRAM vault serves as a victim cache to an LLC slice; capturing the evictions from its corresponding LLC slice and moving cache blocks to the LLC upon a hit, essentially being exclusive of the LLC. The LLC slices and the DRAM vaults follow the same address interleaving scheme as the on-chip LLC slices [2] . This arrangement ensures that an access request that maps to an LLC slice also maps *only* to the DRAM vault which is connected to the LLC slice

[1] miss in all levels of on-chip caches.

[2] We discuss the case where the number of DRAM vaults = the number of cores/LLC slices. In the case where the number of vaults < number of cores/LLC slices, a DRAM vault will serve as a victim cache for multiple LLC slices, together with being directly connected to them. In the case where the number of vaults > number of cores/LLC slices, multiple DRAM vaults will serve as victim caches for an LLC slice, with direct connections to it. The optimal number of DRAM vaults could be decided upon by factoring in the core count and the layout of the processor die.

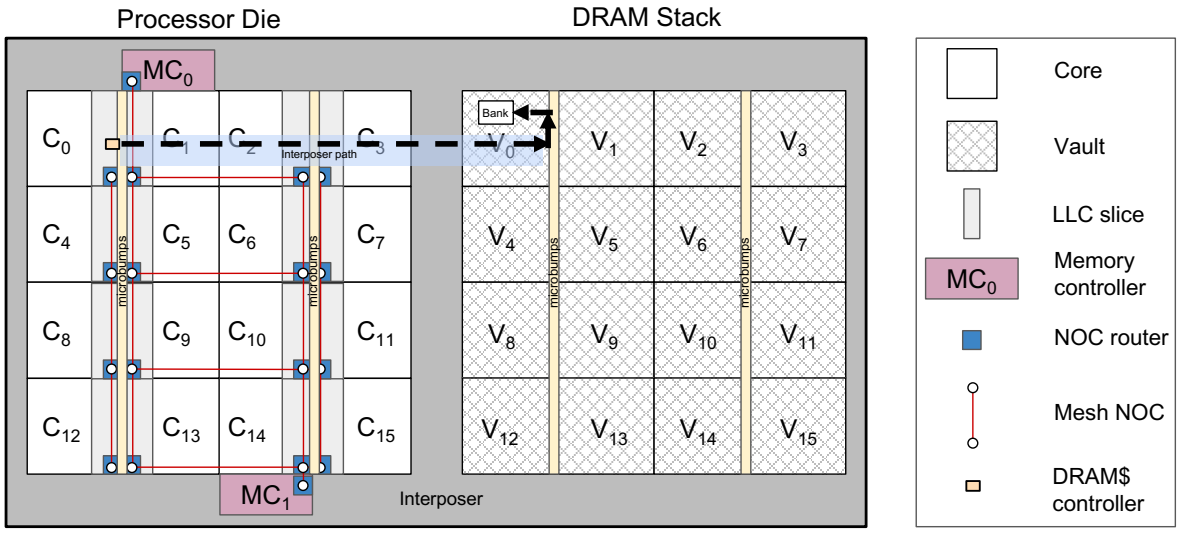


Figure 4.1: Optimised access path from the processor die to DRAM vault.

tile. An LLC miss, therefore, incurs a single DRAM lookup in the DRAM vault and follows the shortest path as highlighted in figure 4.1.

Through judicious logical organisation of the DRAM vaults, CARVE ensures that an on-chip miss destined for the DRAM cache incurs no NOC traversals, is processed by a per-core DRAM cache controller, and sent over a per-core interposer path. In doing so, CARVE achieves a significantly lower latency to access the DRAM cache compared to conventional DRAM caches, up to 61% lower in our analysis. CARVE is non-disruptive of the directory-based coherence at the LLC as it introduces no additional coherence overheads.

4.2 CARVE Design

CARVE retains the traditional on-chip cache hierarchy and augments each on-chip LLC slice with a functionally independent vault in die-stacked DRAM, which serves its respective LLC slice as a victim cache. We refer to the DRAM vaults in CARVE as victim vaults (VV) . The DRAM vaults, together, constitute a DRAM cache which is shared by all cores, with high effective capacity. A vault is directly connected to its corresponding LLC slice through microbumps and wires running through the silicon interposer, and accessed through a dedicated DRAM cache controller. This way, no NOC traversal is required to access the VV. Additionally, instead of in a shared I/O lane, the request lands in the exact DRAM vault where the requested block resides, eliminating request diversion on

the DRAM stack side. Figure 4.2 illustrates CARVE in comparison with the baseline.

In the following sections, we will discuss the design details of CARVE.

4.2.1 CARVE Layout

In this section, we discuss the physical placement of VV in CARVE on the package in relation to the processor die. A silicon interposer enables connections between the processor die and the DRAM stack in CARVE.

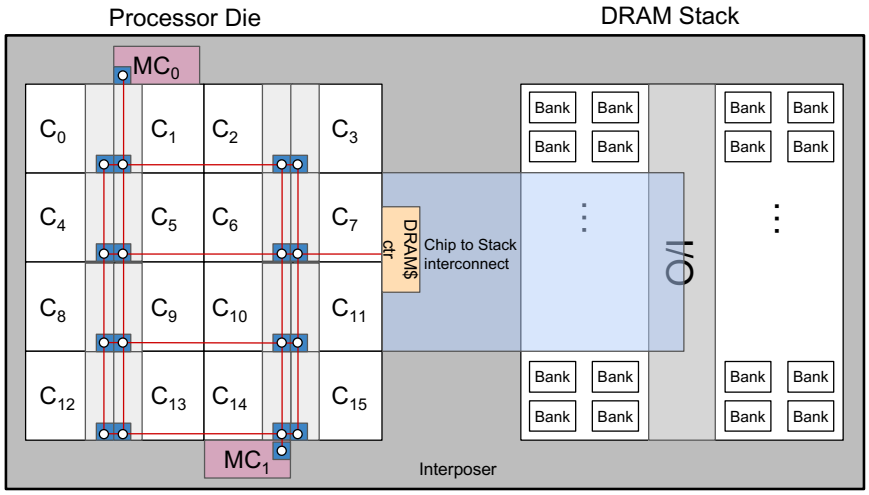
DRAM stack. In CARVE, the VV DRAM stack[3] sits next to one edge of the processor die on the interposer. This is non-disruptive to the main memory controller(s) placement which are positioned on one edge (or two edges) of the processor die [11, 13]. Technological constraints impose a minimum gap of 100μm between the processor die and the DRAM stack.

Microbump arrays. CARVE requires microbumps and TSVs on the processor die and the DRAM stack in order to connect to the interposer. Microbumps are placed at the extremities of the interposer connections, *i.e.*, the core+LLC tile and the DRAM vault. TSVs are required to establish connections between the die and the interposer. CARVE uses microbump arrays on the processor die as shown in figure 4.3. Our discussion and main evaluation will focus on the two microbump array layout of Figure 4.3(a). As discussed earlier, a tile's microbumps sit right next to the LLC slice. Upon a miss in the LLC slice, the request may be forwarded to the relevant vault using these microbumps and the connecting wires on the interposer. Similarly, the microbumps for each vault sit next to it on the base die of the DRAM stack.

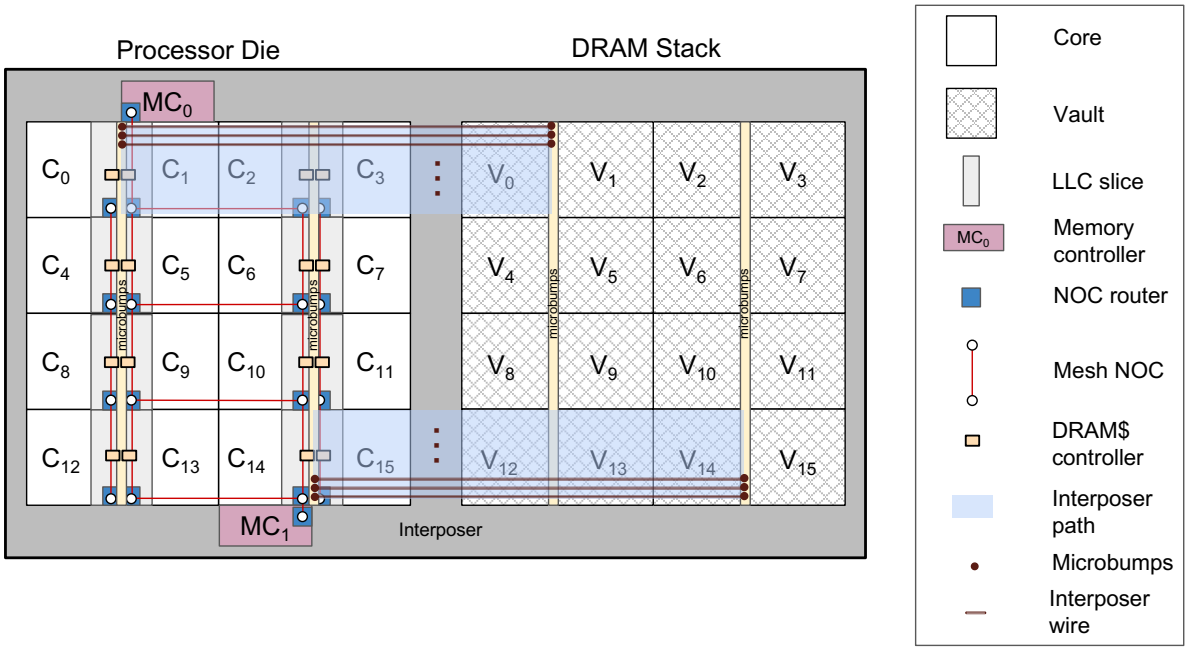
The microbump array affordable is narrower than the cache block width (64-bit vs 64-byte) which adds serialisation cycles to the request transmission. We estimate the affordable number of microbumps using a pitch of 50μm (within the range reported by literature [62, 63]) and the core+LLC slice edge length of about 4mm. We investigate the area overheads of CARVE in section 4.4.3.

Interposer connections. The number of interposer wires correspond to the number of microbumps that can be afforded. Each core+LLC tile has 64 microbumps and interposer wire connections. In chapter 3 we discussed how the transmission delay through interposer wires is similar to on-chip wires. We esti-

[3]For simplicity, the design discussion is in the context of a single DRAM stack, but is also applicable to multiple stacks.

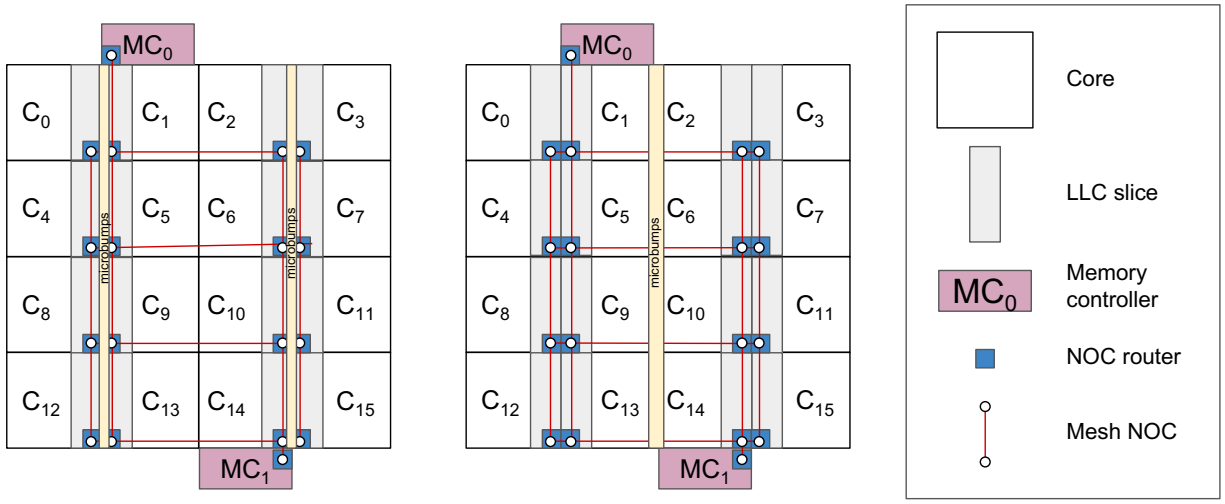


(a) Baseline with conventional DRAM cache.



(b) CARVE with latency-optimised DRAM vaults. Only two LLC-to-vault connections shown for simplicity.

Figure 4.2: Logical representation of (a) baseline and (b) CARVE.



(a) Two microbump arrays.

(b) One microbump array.

Figure 4.3: Processor die layout with microbump arrays connecting to CARVE. The DRAM stacks, which follow the same microbump array layout respectively, have been omitted for simplicity.

mated that in total, signal transmission through the microbumps and the interposer incurs 3-4ns on the round trip.

4.2.2 Cache Organisation and Operations

The die-stacked DRAM vaults in CARVE store data and the associated tags. Figure 4.4 shows the cache block tag and layout in a DRAM row. CARVE uses a block-based cache organisation to maximise effective capacity. Each block unifies tag and data into a single unit called TAD. Prior work has identified this organisation to be beneficial for latency as tag and data fetch may be performed at the same time [43], instead of serialised accesses to both in a non-unified arrangement. Additionally, the DRAM vaults employ a direct-mapped arrangement to further reduce cache lookup time by limiting the number of TADs read for a single request to one, given that CARVE is a latency-optimised DRAM cache organisation. The DRAM vaults in CARVE are unlikely to benefit from set-associativity given the low probability of conflict in a large capacity cache.

Each DRAM vault is employed as a victim cache each for the LLC slice that it is directly connected to. The victim policy triggers a cache block fill in the vault only upon an eviction from the LLC slice and reduces thrashing in the directly-mapped DRAM vaults. The DRAM vaults are exclusive of the on-chip LLC, *i.e.*,

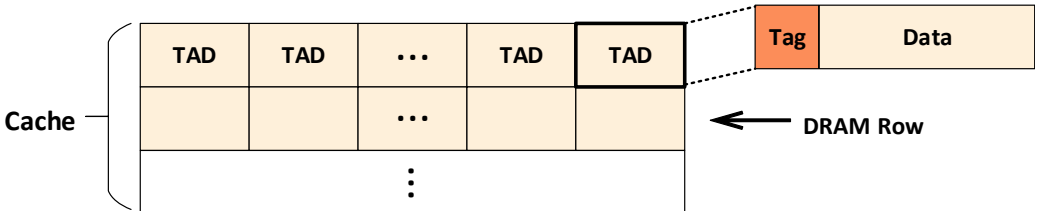


Figure 4.4: DRAM vault organisation.

a cache block may reside *either* in an LLC slice *or* in a DRAM vault. An exclusive cache policy maximises effective capacity and, more importantly, ensures that an eviction from a DRAM vault does not trigger an invalidation in the LLC (called inclusion victim). Additionally, the DRAM victim vaults sit logically below the on-chip LLC which does not introduce any coherence implications. Therefore, cache coherence remains unchanged at the on-chip LLC, and is enforced through a directory-based MESI protocol.

A cache block evicted from the on-chip LLC slice is inserted into the corresponding victim DRAM vault. Upon a request hit to the victim vault, the cache block is invalidated in the DRAM vault and installed into the on-chip LLC.

4.2.3 Address Interleaving in CARVE

Address interleaving is a method through which consecutive memory addresses are scattered across different units. In the context of static NUCA LLCs, address interleaving scatters memory address across the LLC slices, through a statically determined scheme. An address maps to only a single LLC slice based on its address, *i.e.*, the cache block containing the memory address is allocated in a given LLC slice. The address space is, thus, evenly spread across the LLC slices, with the LLC capacity logically shared by all cores.

In CARVE, the independent VV are logically shared by the cores and employ address interleaving, similar to the LLC. In fact, the address interleaving scheme of the VV matches that of the LLC slices. In effect, the LLC address interleaving scheme is projected on to the VV, as shown in figure 4.5. Upon an access request for address `A+n`, LLC_n is looked up. If the LLC lookup results in a miss, only VV_n may be caching the block corresponding to `A+n`. LLC_n directly connects to VV_n (as shown in figure 4.2), and the request may be easily forwarded. Similarly, when the cache block containing address `A+n` is evicted from LLC_n, VV_n serving as the victim cache for LLC_n captures and allocates the cache block.

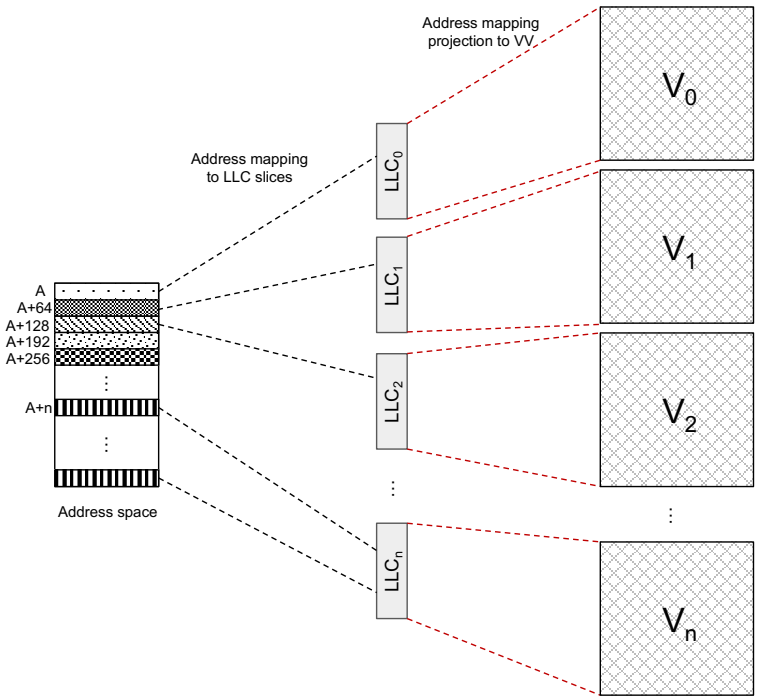


Figure 4.5: Address interleaving in the NUCA LLC slices projected to the VV. The integers represent 64B cache block offsets.

4.2.4 Discussion

Latency-Optimised DRAM. CARVE uses latency-optimised DRAM vaults based on non-commodity DRAM technology. While DRAM technology has long been aimed at optimising for cost-per-bit, adoption of DRAM stacks as caches motivates the need to reassess this optimisation target, and prioritize low latency. Latency-optimised DRAM stacks are well-suited for providing large, scalable cache capacities which is vital for datacenter servers running data-intensive applications. With on-chip caches failing to keep up with the capacity scalability demands, latency-optimised stacked DRAM caches offer great promise, and latency-optimised DRAM technology may start experiencing greater adoption.

Vault Miss Prediction. The TAD organisation in CARVE means that a miss is not discovered until the DRAM access completes. A miss predictor, such as a MissMap [64], can avoid DRAM accesses if they are known to be misses, thus avoiding the associated latency cost. We consider miss prediction for the vaults in CARVE and evaluate system performance in section 4.4.2.

Defect Tolerance. The DRAM victim vault organisation in CARVE has two advantages with respect to defect-tolerance. First, due to logical partitioning of

the DRAM stack into vaults, a critical defect is likely to be limited to a single DRAM vault, with the other vaults completely unaffected. Therefore, a critical fault does not render the entire DRAM stack unusable. Second, even with a faulty DRAM victim vault, the corresponding core+LLC slice may continue to function as normal, albeit without a level of victim cache. Both these factors contribute towards a higher yield, which directly improves manufacturing cost.

CARVE Design Dependence on Stacked DRAM. CARVE retains the on-chip cache hierarchy and augments it with VV in stacked DRAM. As a result, the CARVE design scheme works even in the absence of a DRAM stack, albeit with lower overall system performance due to a missing level of cache. From a manufacturing perspective, this allows processor line variants both with and without stacked DRAM based on the same overall design.

Logically Private Vaults Undesirable. A simplistic approach to accommodate the latency-optimised DRAM vaults in the cache hierarchy would be to deploy the DRAM vaults as per-core private caches, logically behind the on-chip shared LLC. In this arrangement, on-chip miss of an access request issued by a core could be directed through the DRAM cache controller and the interposer connections residing on the core tile. A directory would keep track of all the blocks cached in the DRAM vaults, which, due to its size, would also be distributed and physically stored in the DRAM vaults.

Upon an access request issued by a core which misses in the core's private cache levels, the request needs to be forwarded, over the NOC, to the LLC slice where the address of the requested block maps. This step incurs significant latency; 3 mesh NOC hops on average in a 4x4 mesh. Now if the the request also misses in the LLC slice, it needs to be sent to the DRAM cache. In the case of core-private DRAM vaults, the request will be sent back to the requesting core's tile, incurring a further 3 NOC hops on average. Once on the requesting core's tile, the request is forwarded to the core's private DRAM vault through the dedicated DRAM cache controller and interposer wires. If the requested block is found in the core's private DRAM vault, the block is returned to the requested core over the same path. In the case of a private DRAM vault miss however, first the directory needs to be looked up to check if the block resides in any of the remote DRAM vaults. As the directory itself is located in the DRAM vaults, first the directory slice storing information about the requested block needs to be looked up. Through the NOC, the request needs to be routed to the core tile corresponding to the

directory slice's DRAM vault. The directory slice responds with information about which core's private DRAM vault is caching the requested block over the same path. In order to access the target DRAM vault, the request is sent to the core tile corresponding to the DRAM vault over the NOC. This request trajectory visibly incurs significant NOC traversals at various levels and multiple DRAM vault lookups, massively inflating the access latency. Additionally, the effective capacity of the DRAM vaults is lowered as private caches allow data replication.

Power Delivery and Wiring Challenges on the Interposer.

The power delivery network on the silicon interposer is laid out as a grid due to the fabrication metal density rule. Because the interposer serves as the power supplier for die/chips placed on it, their power delivery network follows a similar grid patterning, as opposed to solid planes in a conventional package substrate. Additionally, both the processor die and the VV stack use the same power domain, resulting a high number of power and ground TSVs connecting to the top of the package substrate. Existing literature studies the implications of power delivery in the presence of a silicon interposer in depth through detailed modelling [65, 66, 67, 68, 69].

CARVE requires fine-grain connections between the processor die and the VV in the DRAM stack. This results in high-density interposer wires, posing a significant challenge for providing short wirelengths with limited routing resources available on the interposer, together with the power delivery network grid. Selection of the minimal number of interposer metal layers and a routable wiring placement becomes challenging, and existing literature investigates this problem [70, 71, 72, 73].

4.3 Methodology

4.3.1 Evaluated Systems

We model a 16-core CMP 3-way OoO cores running at 2.0 GHz. Table 4.2 details the system parameters. We extract DRAM cache latency from CACTI and model a constant access latency for DRAM in our simulations. We assume a closed-page policy for DRAM, both in cache and main memory which is beneficial for server workloads [74]. We use a fairly aggressive memory access latency of 50ns. Our

	DRAM access	64-bit serial	Microbumps	Interposer	Controller	Total
Latency	11cyc	8cyc	3cyc	5cyc	4cyc	31cyc

Table 4.1: Access latency breakdown of a DRAM victim vault in CARVE.

evaluation focuses on 3-level on-chip cache hierarchies, representative of modern processors.

Baseline: The baseline uses a shared 32MB LLC split into 16 banks, with a 7-cycle bank access latency. The average round trip time for an LLC hit, including the NOC, is 25 cycles.

Baseline+DRAM$: Baseline augmented with an 8GB conventional DRAM cache. The DRAM cache is hardware managed and uses a page-based arrangement considered state-of-the-art for servers [15, 45]. Conventional DRAM caches use the same DRAM technology as main memory and as such have similar access latency as we discussed earlier. However, we assume an optimistic conventional DRAM cache access latency 20% faster than main memory. We also assume perfect miss prediction and infinite bandwidth.

CARVE: Baseline augmented with a tightly coupled victim vault per LLC slice in stacked DRAM. Each latency-optimised vault has a capacity of 256MB, adding up to a total of 4GB capacity for the DRAM stack, and an access latency of 11 cycles. The vaults utilise latency-optimised DRAM technology. This system configuration uses two microbump arrays on the processor die. We use a 64-bit wide interface adding 8 serialisation cycles and 4 cycles of vault controller delay. The latency-optimised path from an LLC slice to the victim vault is 31 cycles. The latency breakdown is shown in table 4.1.

CARVE-CO: CARVE with capacity-optimised vaults of 512MB each, adding up to a total capacity of 8GB, at an access latency of 20 cycles. The total access latency from an LLC slice miss to the corresponding victim vault is 40 cycles.

ideal4GBLLC: This ideal configuration uses a shared 4GB LLC split into 16 vaults, with a 7-cycle bank access latency. The average round trip time for an LLC hit, similar to the baseline system, is 25 cycles. This design point represents an idealised scenario in which the 4GB aggregate vault capacity serves as the on-chip LLC at the same latency as the typical 32MB LLC.

Processor	16-core, 2GHz, 3-way OoO, 128 ROB, ISA: UltraSPARC v9
L1-I/D	64KB, 8-way, 64B line, 3 cycles, private, stride data prefetcher
L2	512KB, 8-way, 64B line, 5-cycle, private, stride prefetcher
Interconnect	4x4 2D mesh, 3 cycles/hop
Baseline on-chip LLC	32MB shared NUCA, 7 cycles, 16-way, 64B line, non-inclusive MESI, LRU
Stacked DRAM Victim Vault	Per-LLC slice, direct-mapped, 64B line, 512B page CARVE: 256MB vault, 31 cycles CARVE: 512MB vault, 40 cycles
Conv. DRAM cache	8GB, page-based, direct-mapped, 40ns
Main memory	Access latency 50ns

Table 4.2: Microarchitectural parameters of the simulated systems. Extended parameters table in appendix A.

4.3.2 Simulation Infrastructure

We use Flexus [75], a full system multiprocessor simulator, based on Simics. Flexus models the SPARC v9 ISA and extends Simics with out-of-order (OoO) cores, memory hierarchy, and NOC. To reduce simulation time, Flexus integrates the SMARTS [76] methodology for sampled execution. For each sample, we first warm-up architectural and microarchitectural state, then run cycle-accurate simulation and measure performance. In order to evaluate performance, we measure the number of application instructions executed per cycle (including time spent executing operating system code); this metric has been shown to reflect system throughput [75].

To ensure high confidence in our developed simulation model, we adopt a stepwise development scheme and conduct validation at each step. We validate the models using some sanity checks – checking if the performance and hit/miss data correlates well, checking if each relevant simulation statistic is meaningful, and using a number of simple test cases with deterministic output behaviour which we track during the simulations. We conduct a final validation step of the completed

model by comparing its performance in an idealised case against an opportunity study which establish the performance limits of the proposed model.

4.3.3 Workloads

We evaluate the systems on scale-out server workloads including Web Search, Data Serving, MapReduce and SAT Solver, which are taken from CloudSuite [46], and a Web Frontend workload from SPECweb2009. We do not use the Media Streaming workload from CloudSuite, as it does not scale beyond 2-4 threads [77]. Details of these workloads are listed in Table 4.3. We also evaluate the systems on contemporary parallel applications from the PARSEC-3.0 benchmark suite [47], listed in Table 4.4. We simulate a 16-core setup with the number of threads equals to the number of cores. We use the native input sets and only simulate the Region of Interest (ROI). Compilation and runtime issues prevented us from being able to run three workloads: `dedup`, `streamcluster`, and `swaptions`.

For simulation, samples are drawn over 80 billion instructions (5 billion per core) for each workload. For each sample, we run cycle-accurate simulations from checkpoints that include full architectural and partial microarchitectural state, which includes caches and branch prediction structures. We run for 100K cycles to achieve steady state and measure over the following 200K cycles per sample.

4.4 Evaluation

We first evaluate CARVE against the baselines on the scale-out server workloads. Next, we extend our evaluation to Parsec workloads.

4.4.1 Evaluation on Scale-Out Workloads

Figure 4.6 plots the system performance on scale-out workloads, with the results normalised to baseline (see Section 4.3.1 for a description of evaluated systems). We observe that CARVE provides higher performance than the baselines across all the workloads. CARVE improves performance by 2-23% (geomean 12%) over the baseline, with Web Search observing the highest performance improvement of 23%. Figure 4.7 presents the misses per kilo instructions (MPKI) to main

Scale-Out	
Web Search	*Nutch 1.2 / Lucene 3.0.1,* 230 clients, 2 GB index, 23 GB data segment
Data Serving	*Apache Cassandra 0.7.3,* 150 clients, 8000 operations per second, 15GB YCSB dataset, 7GB Java heap, 400MB garbage collector space
Web Frontend	*Apache HTTP Server v2.0,* e-banking, 16K connections, fastCGI, worker threading model, 6GB dataset
MapReduce	*Hadoop MapReduce, Apache Mahout 0.6,* Bayesian classification algorithm, 4.5GB set of pages, 2GB Java heap
SAT Solver	*Cloud9 parallel symbolic execution engine* Klee SAT Solver, Symbolic execution of `printf` with four 5-byte and one 10-byte symbolic commandline arguments, one instance per core, about 700MB working set per instance

Table 4.3: Server workloads used for evaluation.

Parsec			
Name	Domain	Input set	Est virt mem use
blackscholes	Financial analysis	10,000,000 options	>652MB
bodytrack	Computer vision	4 cameras, 261 frames, 4,000 particles, 5 annealing layers	380MB
canneal	Engineering	15,000 swaps/temperature step, 2,000deg start temp, 2,500,000 netlist elements	1239MB
facesim	Animation	80,598 particles, 372,126 tetrahedra, 100 frames	552MB
ferret	Similarity search	3,500 image queries, database with 59,695 images, find top 50 images	1330MB
fluidanimate	Animation	500,000 particles, 500 frames	642MB
freqmine	Data mining	Database of 250,000 web html documents, minimum support 11,000	941MB
vips	Media processing	18,000 × 18,000 pixels	>361MB
x264	Media processing	1,920 × 1,080 pixels (HDTV resolution), 512 frames	>310MB

Table 4.4: List of PARSEC 3.0 workloads used for evaluation. Memory footprints from [5].

memory for the evaluated systems. Unsurprisingly, Web Search also experiences the biggest main memory MPKI reduction (93%).

On Web Frontend, CARVE achieves a performance improvement of 4% over the baseline while baseline+DRAM$ delivers a performance improvement of 2%. However, the MPKI to main memory for baseline+DRAM$ is lower than CARVE. This reduction in MPKI does not result in higher performance because the DRAM cache hit latency in baseline+DRAM$ is 80 cycles, only 20% higher than main memory, and significantly higher than the 31 cycles in CARVE. Therefore, CARVE outperforms baseline+DRAM$ on Web Frontend even with a higher MPKI to main memory.

The capacity-optimised variant of CARVE observes a geomean speedup of 11%, slightly lower than the 12% achieved by the latency-optimised variant. The main memory MPKI for CARVE-CO is expectedly lower than CARVE across all workloads as shown in figure 4.7. Although CARVE-CO has a higher aggregate DRAM cache capacity of 8GB, the higher vault access latency limits the

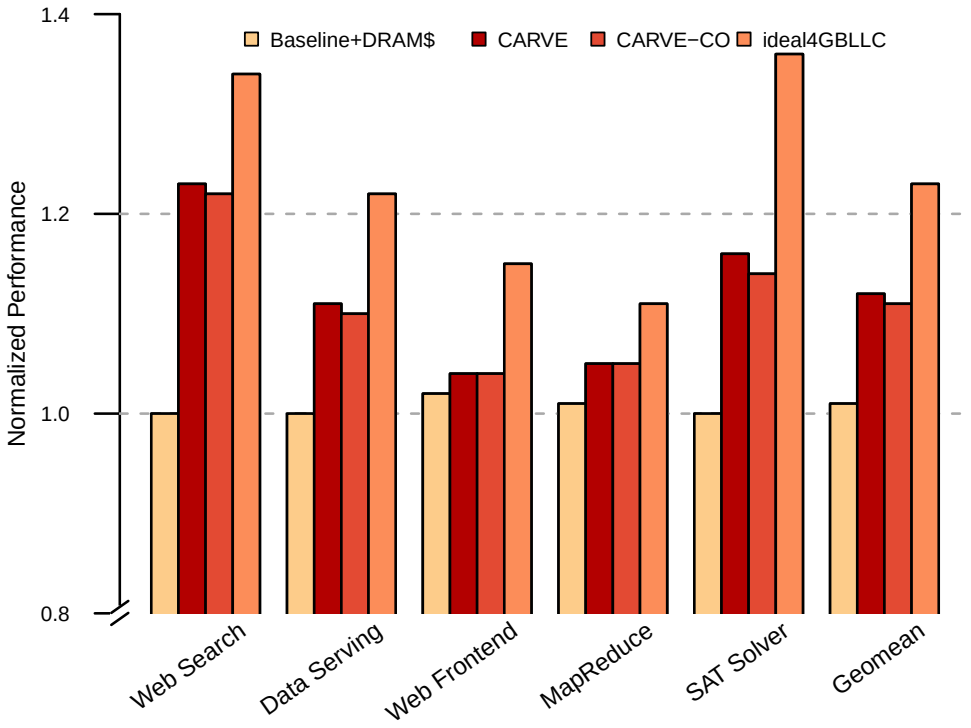


Figure 4.6: Performance results on scale-out workloads.

performance gains from the added capacity.

Figure 4.7 shows that CARVE and ideal4GBLLC have very similar MPKI's, and lower than baselines. This is due to CARVE and ideal4GBLLC having the same aggregate DRAM cache capacity. ideal4GBLLC yields higher performance, however, due to a significantly lower access latency.

4.4.2 CARVE Miss Prediction

We identified the opportunity for DRAM miss prediction in CARVE. We now evaluate the usefulness of this optimisation in the limit. We equip CARVE with a Miss Predictor for vault accesses. The predictor is perfect, requiring 0 time and having 100% accuracy. Figure 4.8 plots the performance of CARVE with

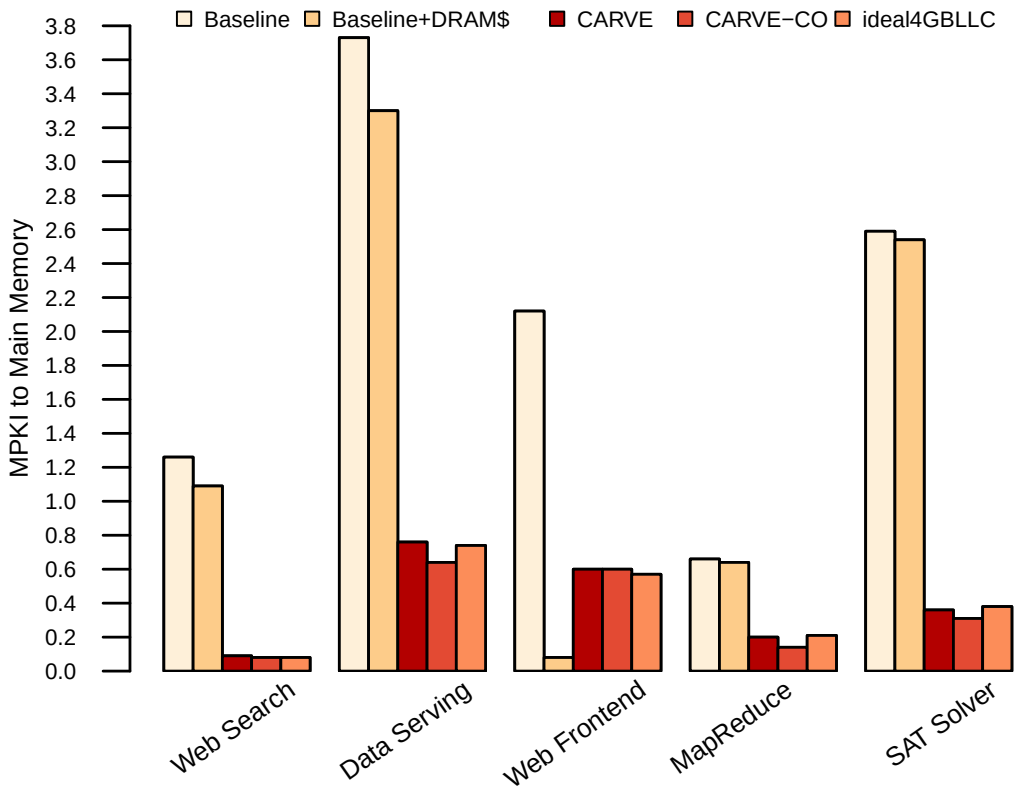


Figure 4.7: MPKI to main memory for the evaluated systems.

and without the Miss Predictor. Data Serving and Web Frontend observe the highest performance improvement of 2%. Overall, the benefits of miss prediction are small and we conclude that the benefits do not outweigh their cost and extra design complexity.

4.4.3 CARVE Overheads

In CARVE, each of the VV is connected to its corresponding on-chip LLC slice via a 64-bit interface. This interface constitutes microbumps on both ends connected by wires running through the interposer. We study the area overhead and power dissipation of this entire interface using the parameters in Table 4.5.

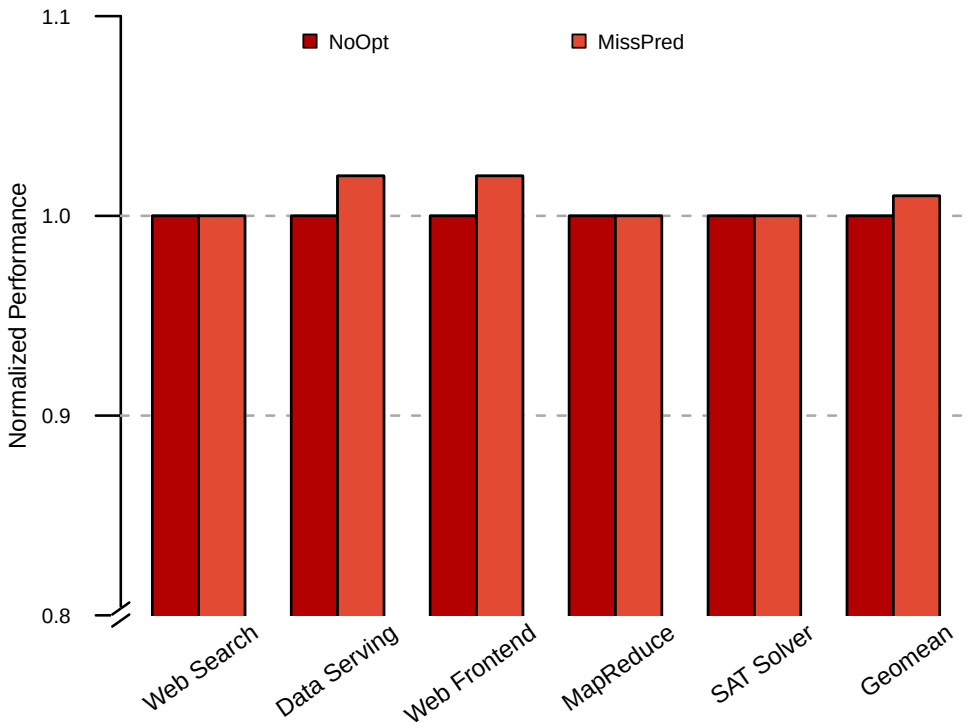


Figure 4.8: Effect of miss prediction optimisation on CARVE. The study assumes perfect miss prediction.

Area: We use a conservative microbump pitch estimate of 50μm [62, 63]. For the microbump array layout, the area overhead of the microbumps per tile is 0.16mm^2. Additionally, prior literature reports the die-to-die interconnect physical layer (PHY) circuits account for around 0.105mm^2 die area by conservative estimates, and 686μm^2 by more aggressive design techniques [78]. We estimate a wire length of 8mm between a processor tile and its corresponding vault based on estimates of the edge length of the processor die and the DRAM stack as mentioned in chapter 3.

Power: CARVE uses microbump arrays and interposer wires. The microbumps, together for all the processor tile to vault connections, consume a total of 2.15mW assuming a 1Gbps rate. Even at a more aggressive 2Gbps rate,

	Length/Area	Power
Microbumps	50um pitch	1.075uW/Gbps
Interposer wire	8mm maximum	0.1075uW/1um/Gbps

Table 4.5: Microbump and interposer parameters in CARVE.

the power dissipation is 4.3mW. Power dissipation in the interposer wires is proportional to the wire length. At a wire length of 8mm, the total power dissipation in the interposer wires is 880mW at a 1Gbps rate, or 1.761W at 2Gbps. The total overhead is under 2W which is a small fraction of the total processor budget.

4.4.4 Evaluation on Parsec Workloads

We extend our evaluation of CARVE on Parsec workloads. Figure 4.9 plots system performance normalised to the baseline system. CARVE outperforms the baseline system with the biggest performance improvements on `canneal` and `vips`. Figure 4.10 plots the MPKI to the main memory for the systems. We observe that CARVE and ideal4GBLLC have very similar MPKI's, and lower than baseline+DRAM$. This is expected as CARVE and ideal4GBLLC have the same aggregate DRAM cache capacity. However, ideal4GBLLC yields higher performance due to a significantly lower access latency.

4.4.5 Evaluation of CARVE with a single microbump array

We also evaluate CARVE with a more conservative single, central on-chip microbump array (similar to AMD's Zen chiplets [40]). In this design, CARVE-1u, The microbump array lies one NOC hop away for half of the core+LLC tiles, incurring 39 cycles for the VV access. Figure 4.11 plots the system performance in comparison with CARVE. As expected, system performance with CARVE-1u is lower than CARVE due to its higher vault access latency.

4.4.6 CARVE with an Organic Substrate

We argue for interposer-based CARVE in our proposal. The recent AMD chiplet-based designs connect the chiplets through wires on an organic substrate which incur high latency as discussed in chapter 2. We also consider CARVE using an

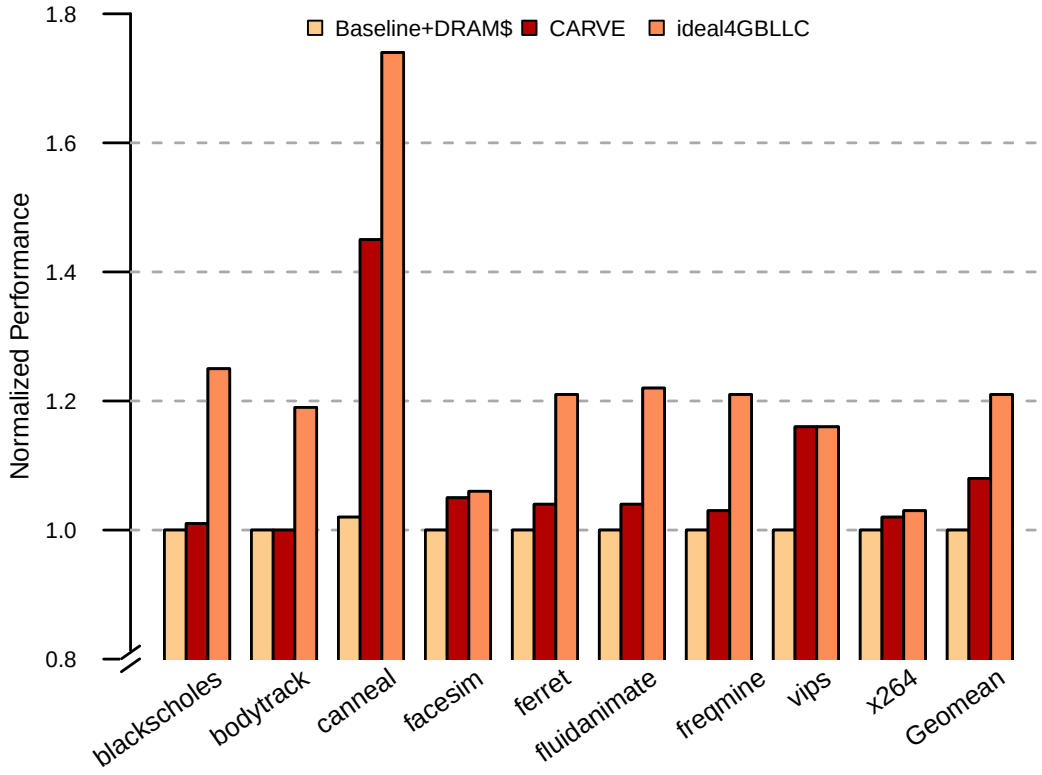


Figure 4.9: Performance results on Parsec workloads.

organic substrate, CARVE-organicsubstrate, with a much higher latency. Recent research reports the latency across the substrate to be around 10ns[49]. Figure 4.12 plots the system performance. As expected, CARVE-organicsubstrate performs significantly lower than CARVE, providing a performance gain of 6% over the baseline vs 12% achieved by CARVE.

4.4.7 CARVE as a Replacement for On-Chip LLC

CARVE deploys DRAM vaults as per-LLC-slice victim caches, supplementing the traditional on-chip cache hierarchy. We evaluate an additional design point in which the on-package DRAM vaults act as a direct replacement for the traditional

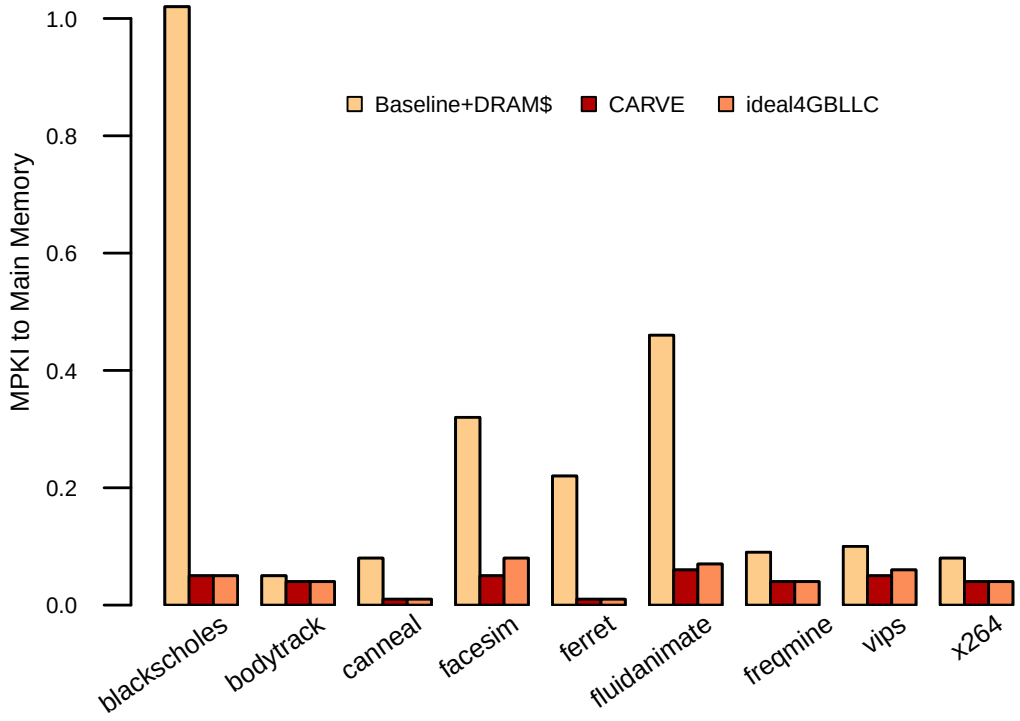


Figure 4.10: MPKI to main memory for CARVE for the evaluated systems.

on-chip LLC. The design, *CARVEasLLC*, dispenses with the on-chip LLC slices but uses the per-core+LLC tile connections to the DRAM vaults. In CARVE, every miss from an LLC slice triggers a lookup in the corresponding victim vault. In contrast, in CARVEasLLC, a miss in core's private L2 triggers a lookup in a DRAM vault, as per the address interleaving scheme. Figure 4.13 shows a representation of this system. In this design, *CARVEasLLC*, each vault has a capacity of 256MB, adding up to a total of 4GB capacity for the DRAM stack. The average round trip time for a hit, including a vault access and NOC traversal, is 49 cycles.

Figure 4.14 plots the system performance of CARVEasLLC in comparison

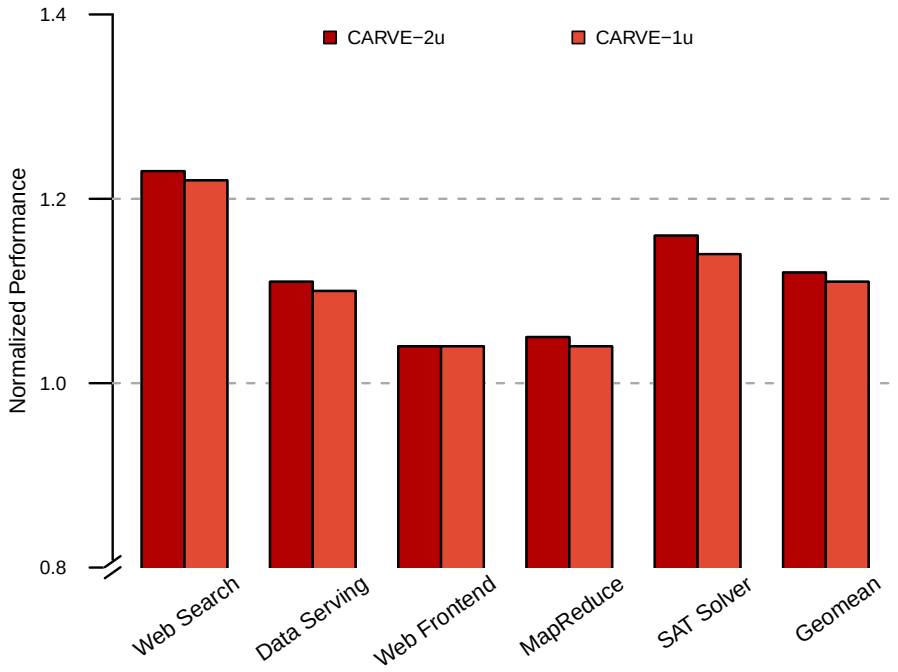


Figure 4.11: Performance results for different CARVE layouts. CARVE labelled as CARVE-2u.

with CARVE. We observe that CARVEasLLC performs consistently worse than CARVE, with a geomean performance improvement of 1% vs 12% for CARVE. In fact, CARVEasLLC performs even lower than the baseline on Web Frontend and MapReduce. This is expected as the extremely high LLC latency of CARVEasLLC outweighs the gains of the higher LLC capacity. Therefore, we conclude that CARVE is not a suitable replacement for the traditional on-chip LLC.

4.5 Related Work

Stacked DRAM Caches: Die-stacking enables packing of many gigabytes of DRAM cells together in a single stack, divided into many banks and accessed

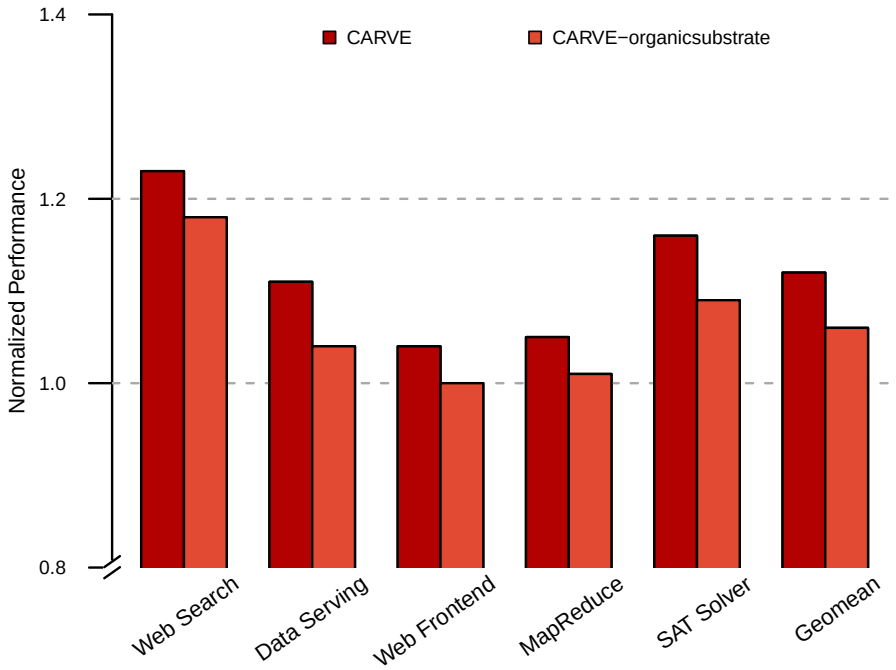


Figure 4.12: Evaluation of CARVE with an organic substrate instead of a silicon interposer.

through a wide interface which together provide high bandwidth. Scientific and HPC applications are bandwidth-hungry and benefit greatly from high-bandwidth DRAM stacks. Recent GPUs and accelerators are thus equipped with HBM stacks [79, 80, 81, 13, 82]. Unfortunately, HBM stacks have high access latencies, comparable to that of main memory as discussed in chapter 2.

Prior works have proposed latency reduction of DRAM caches through placing tags in SRAM [15, 44, 83, 84], using a direct-mapped cache arrangement instead of employing set-associativity to reduce the number of lookups required for caches with tags stored in DRAM [43, 13], and by storing tag and data in one unified cache block to reduce the number of DRAM lookups required [43]. These works do not target the interconnect or DRAM core latency.

Victim Caches: Victim caching was proposed to help capture recent evictions

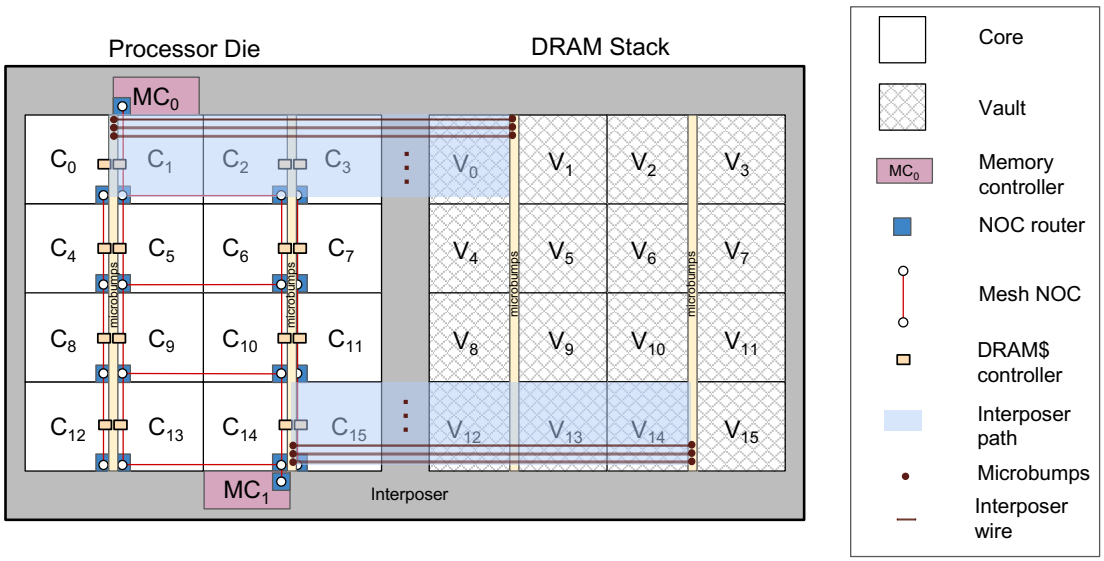


Figure 4.13: CARVE as a replacement for the on-chip LLC.

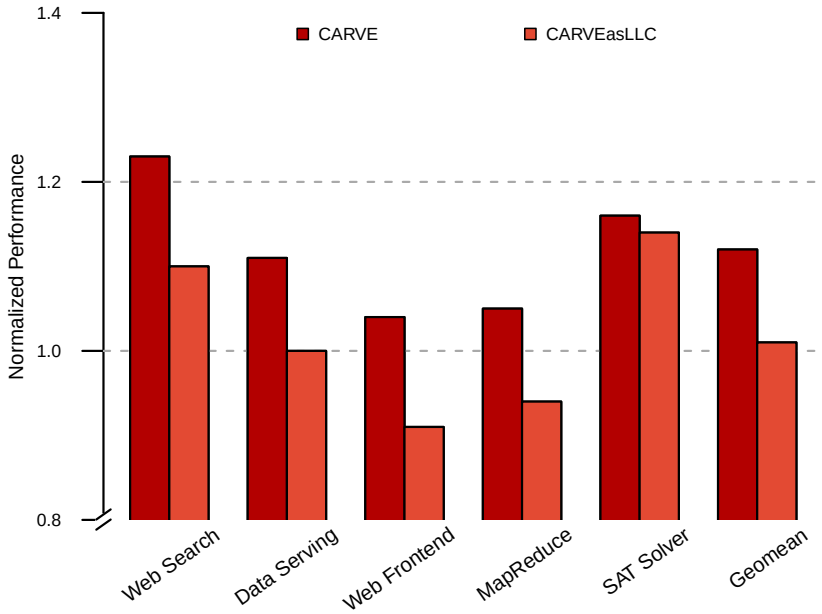


Figure 4.14: Performance results for CARVE as a replacement for the on-chip LLC.

from caches resulting from cache conflicts [85]. Victim caches are, typically, fully-associative structures with a few entries. Lira *et al.* propose deploying a central SRAM cache bank as a shared victim cache for all the NUCA LLC slices [86]. CARVE differs from these approaches in that it deploys large DRAM vaults as per-LLC-slice victim caches.

Co-design of DRAM Core and Interconnect: Recent work identifies and characterises the high access latency of HBM [48]. Similar to the observations in this thesis, Fariborz *et al.* show that interconnect and controller queuing latency dominate the overall access latency, with shared resources being the root cause. The authors argue for a co-design of the DRAM core, the memory controller, and the interconnect to reduce the access latency of a DRAM stack, and leverage silicon photonics in providing the processor to memory network. CARVE differs from this proposal in its use of the DRAM stack as a victim cache and leveraging interposer-based electrical connectivity.

4.6 Summary

Server applications exhibit limited MLP and have modest bandwidth requirements and thus do not benefit significantly from the traditional high-latency DRAM caches. In this chapter we presented CARVE which yields significantly lower latency than traditional DRAM caches by deeply integrating the stacked DRAM caches with the processor die. CARVE deploys latency-optimised DRAM vaults as victim caches for each LLC slice. Our evaluation shows that CARVE provides a 12% performance speedup over the baseline on a set of scale-out server workloads. Our evaluation further shows that the on-package DRAM vaults are ill-suited to directly replace the shared on-chip LLC.

Chapter 5

Die-Stacked Private DRAM LLC Organisation

In the previous chapter, latency-optimised DRAM vaults were used as a victim cache for the traditional on-chip LLC. We observed that on-package shared DRAM vaults were unsuitable as a direct replacement for the on-chip LLC due to their high access latency. This chapter presents an LLC organisation incorporating latency-optimised DRAM vaults, where the DRAM vaults are stacked on top of the processor die and deployed as per-core private LLCs.

5.1 Introduction

Today's server processors tend to employ large on-die shared LLC capacities that attempt to capture the massive data and instruction working sets of server workloads. We observe that while large cache capacities are, indeed, useful for servers, existing configurations are not ideal. First, area and power constraints limit the LLC capacities that can be afforded within a yield-effective die size. Secondly, the larger the capacity (and hence, the larger the die size), the more time it takes to access an LLC slice due to slow wires and multi-hop on-chip network topologies. Last but not least, the shared LLC designs in use today create a significant challenge in isolating the co-running workloads, as evidenced by a large body of recent work exploring the issues around LLC contention in multi-core chips [87, 88, 89]. And while a shared LLC is effective in facilitating low-latency inter-thread data sharing, this capability is not useful for server workloads that are engineered for high scalability and thus have minimal inter-thread data

sharing [16].

We argue for *private* LLCs in die-stacked DRAM as a preferred alternative to traditional on-chip shared LLC architectures. Die stacking naturally overcomes the area limitations of planar silicon by offering multiple layers of densely-integrated memory cells [90, 91]. We propose using the latency-optimised DRAM vaults discussed in chapter 3, stacked directly over the processor die such that each vault sits directly above a core. This organisation naturally avoids long on-chip wire delays inherent in shared LLC architectures and provides a low-latency path from the core to its private DRAM cache slice. The DRAM core in the vaults is engineered for low-latency access, at the expense of capacity, by provisioning a large number of banks, divided into many subarrays and tiles (refer to chapter 3 for details).

The resulting *Die-Stacked Private LLC Organization (SILO)* combines conventional on-chip per-core private L1's and L2's, with private LLC slices in die-stacked DRAM. The DRAM is optimised for latency, at the expense of capacity, to further reduce the access time to the LLC vaults. The caches are kept coherent through a conventional directory-based protocol with in-DRAM metadata. The high hit-rate of large, private DRAM caches and the use of low-latency DRAM for storing metadata makes directory accesses not detrimental to performance. Meanwhile, usage of private caches naturally eliminates inter-core LLC contention, facilitating workload isolation in a many-core setting.

5.2 Motivation

We examine representative scale-out server workloads (details in section 5.4.4) to investigate their requirements from an LLC perspective. The goal is to characterise the performance sensitivity of these workloads to LLC capacity, access latency and inter-thread data sharing.

5.2.1 Performance Sensitivity to Capacity

To understand the sensitivity of scale-out workloads to higher LLC capacities, we sweep the capacity range at a fixed access latency. We present the results for a 16-core setup, the details of which are available in Sec. 5.4. The baseline, with 8MB of LLC, is configured per Scale-out Processors [18], a state-of-the-art

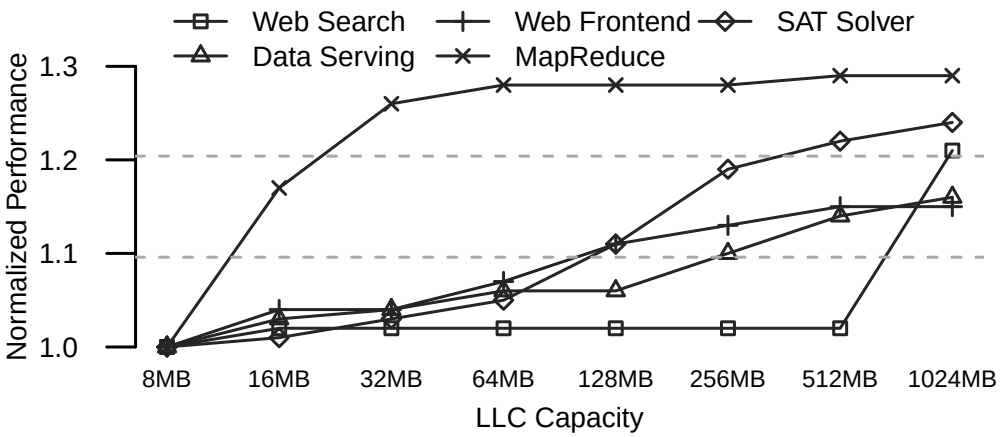


Figure 5.1: Sensitivity to LLC capacity at fixed latency.

specialised server processor architecture targeting scale-out workloads. For larger LLC capacities, the access latency is unchanged from the baseline design.

Figure 5.1 plots workload performance with increasing LLC capacity. All data points are normalised to 8MB. We observe that for most workloads, there is a marginal performance gain from 8MB to 64MB. This can be attributed to the fact that although the increased capacity can hold some part of the secondary working set, it is not large enough to capture it fully, thus limiting the performance benefit. This finding supports the Scale-out Processors design [18], which advocates a small LLC to minimize access latency and area footprint. Beyond 64MB, however, we observe greater performance benefits as the secondary working set starts fitting into the LLC. For Data Serving, Web Frontend and SAT Solver, the performance gain over the 8MB baseline is 10-20% at 256MB but only 2-6% at 64MB. Web Search differs somewhat, showing little benefit from increased capacity up to 512MB, but then gaining 20% in performance at 1024MB (1GB) as the secondary working set starts to fit.

5.2.2 Performance Sensitivity to Latency

We analyse the performance sensitivity of scale-out workloads to the LLC access latency for a range of LLC sizes. Fig. 5.2 plots the results. To minimise clutter, we show only LLC capacities in the range of 64MB and beyond, as that was the region identified in the previous section as delivering the greatest benefit. For each capacity, we sweep the access latency from the baseline (an 8MB LLC) to twice the baseline latency. Each line in the figure represents a geomean performance of

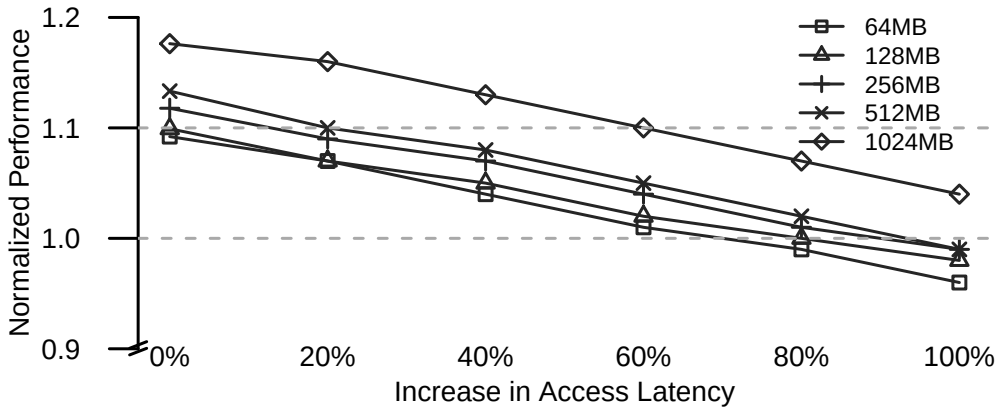


Figure 5.2: Performance sensitivity to LLC latency at different capacities normalized to an 8MB baseline. The isocurves show geomean of scale-out workloads.

the scale-out workloads normalised to the baseline for a given LLC capacity.

We observe that larger LLCs translate to higher performance only at lower latencies. The gains from higher capacity rapidly diminish with increased latency. For example, a 1024MB (1GB) LLC with latency 40% higher than the baseline performs only as well as a 64MB LLC at baseline latency and only 10% better than the 8MB LLC. In fact, as the latency approaches twice the baseline (i.e., the 100% point in the figure), most configurations approach or fall below the performance of the 8MB LLC. This result further corroborates Scale-out Processors [18], which showed larger and slower LLCs to be sub-optimal. Server workloads are highly sensitive to LLC access latency because of their low memory-level-parallelism [18, 16], which exposes the latency of an L1 (and L2) miss to the issuing core. These results show that higher LLC access latencies are detrimental to scale-out workloads even if they are accompanied by a larger LLC capacity.

5.2.3 Performance Sensitivity to Read/Write Sharing

Existing server processors deploy shared LLCs that naturally accommodate inter-thread read-write data sharing arising from producer-consumer data exchange or synchronization. Shared LLCs facilitate such sharing patterns by capturing dirty evictions from a writer's private cache and serving subsequent read requests from other cores without any indirection.

We study the access patterns of scale-out server workloads to characterise the extent of read-write sharing (RW-sharing) and the benefit delivered by a shared

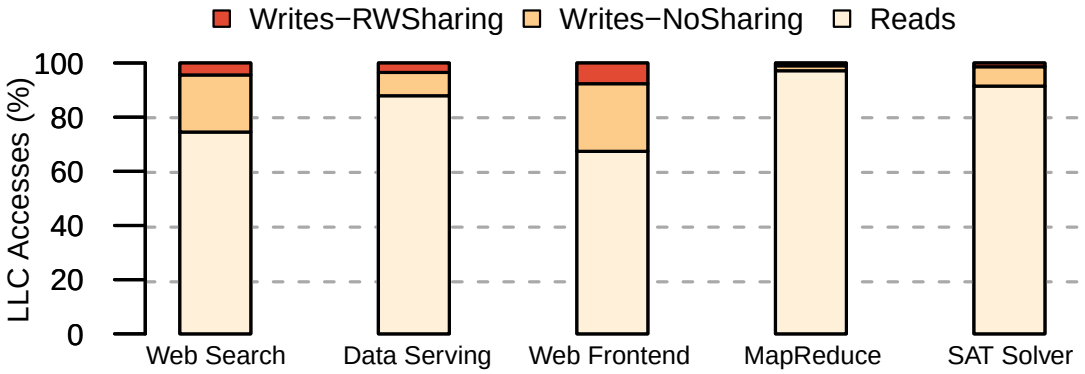


Figure 5.3: Breakdown of accessed LLC blocks.

LLC in accommodating it. For this study, we use an 8MB shared LLC; full system parameters are described in Sec. 5.4. Results are shown in Fig. 5.3, which breaks down LLC accesses into three categories: (1) Reads; (2) Writes that see no reads by non-writing cores (Writes-NoSharing), and (3) Writes that see read(s) by at least one core that is not the writer (Writes-RWSharing).

Generally, we observe limited RW-sharing across the scale-out workloads, which matches the findings of Ferdman *et al.* [16]. MapReduce and SAT Solver have negligible RW-sharing. Web Search and Data Serving exhibit little RW-sharing (4% and 3%, respectively) due to the use of a parallel garbage collector that potentially runs a collector thread on a remote core [16]. This can appear as inter-thread communication between application threads. Web Frontend shows the biggest proportion of RW-sharing.

We further evaluate the impact of RW-shared data on system performance. In order to quantify the performance impact, we artificially increase the access latency of RW-shared blocks compared to other blocks by up to a factor of 4x. Fig. 5.4 presents the results of this experiment. We observe that increasing the access latency of the RW-shared blocks carries a small performance degradation. Doubling the latency of RW-shared blocks results in a performance drop of 0-8%. Even at 4x, the biggest drop observed is for Data Serving and Web Frontend, which lose 10% of performance, consistent with the biggest RW-sharing percentage in figure 5.3.

Our conclusion is that the low incidence of true RW-sharing in scale-out workloads makes them largely insensitive to the LLC access latency for RW-shared data. This implies that the value delivered by a shared LLC in accommodating accesses to such data is low for the scale-out workload domain.

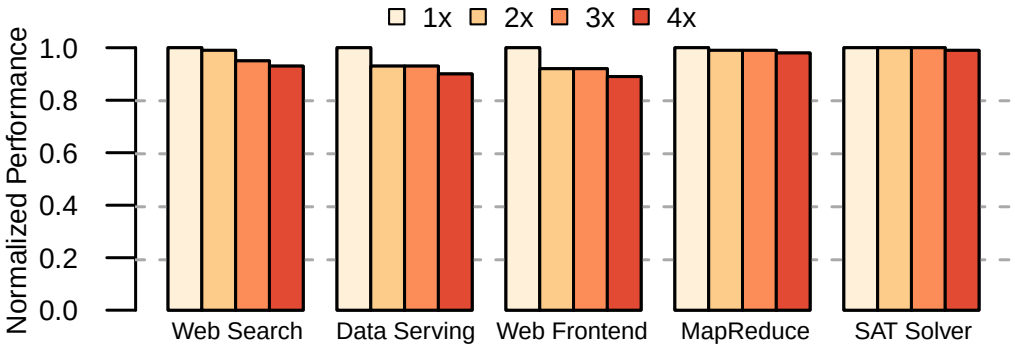


Figure 5.4: Performance impact of increased latency to RW-shared blocks. The latency is increased as a multiple of the baseline.

5.2.4 Summary

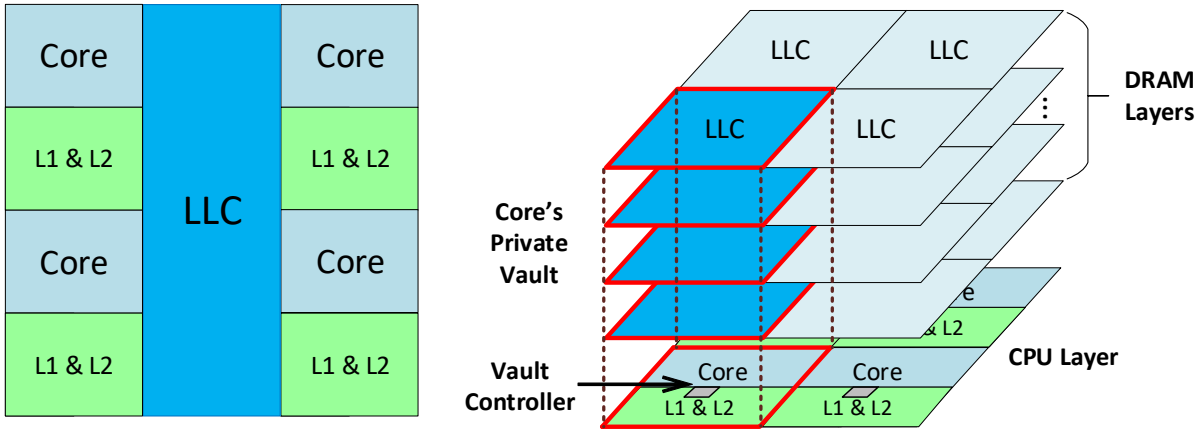
Overall, scale-out workloads benefit from large LLC capacities that help capture their vast working sets. However, the performance benefits diminish if larger capacities are accompanied by an increase in access latency. By design, scale-out workloads also have low degrees of inter-core data sharing making them insensitive to the latency of shared data.

Shared LLCs found in today's server processors fail to accommodate these workload characteristics. On-chip area constraints limit the LLC capacity that can be afforded on a die, while planar interconnect delays incur high access latencies to remote cache banks. The shared LLC organisation does facilitate low-latency inter-thread data exchange; however, the impact of such sharing is low in scale-out workloads.

5.3 SILO Design

SILO directly accommodates the needs and characteristics of server workloads through three optimisations that directly address the deficiencies of today's shared LLC designs.

First, SILO completely dispenses with a shared LLC in favor of an all-private cache hierarchy. Per-core private caches overcome the latency bottleneck of shared caches by limiting the length of interconnect that needs to be traversed on an access. Whereas a conventional shared LLC employing a non-uniform cache access organisation may require a request to be routed over a large silicon plane to a remote cache bank, a private cache located near a core naturally minimises the



(a) Conventional LLC (b) Die-Stacked Private LLC

Figure 5.5: Conventional (a) and proposed (b) cache architectures. A 4-core CPU is shown for simplicity.

interconnect delay. Another advantage of private caches is that they are naturally immune to cache contention, which is a significant concern for shared LLCs in deployment scenarios involving multiple workloads.

Unfortunately, simply converting a conventional shared LLC into a private design would effectively constrain each core to a capacity of just a few MBs. For instance, the current generation of Intel's mainstream server processors (14nm Broadwell family) features a shared LLC with 2.5MB of capacity per core (in fact, the newer server processors have even lower LLCs per core such as [92]). In a shared configuration, the aggregate LLC capacity provided by a multi-core Broadwell CPU is sufficient to capture a meaningful portion of a server workload's instruction and data working set; however, in a private configuration, this capacity would be woefully inadequate.

To overcome the area limitations of planar silicon, SILO deploys the second optimisation: die-stacked DRAM. To avoid long interconnect delays and maintain the latency benefits of a private cache, SILO organizes the DRAM into *vaults*, each of which sits above a processor core. A vault, as discussed in chapter 3, is a multi-die stack of DRAM banks with a dedicated vault controller in the logic layer at the base of the stack. As shown in Fig. 5.5, in the SILO design, the DRAM controller for a vault is located on the CPU die, next to the core directly beneath the vault.

The third optimisation is aimed at reducing the access latency to a vault by

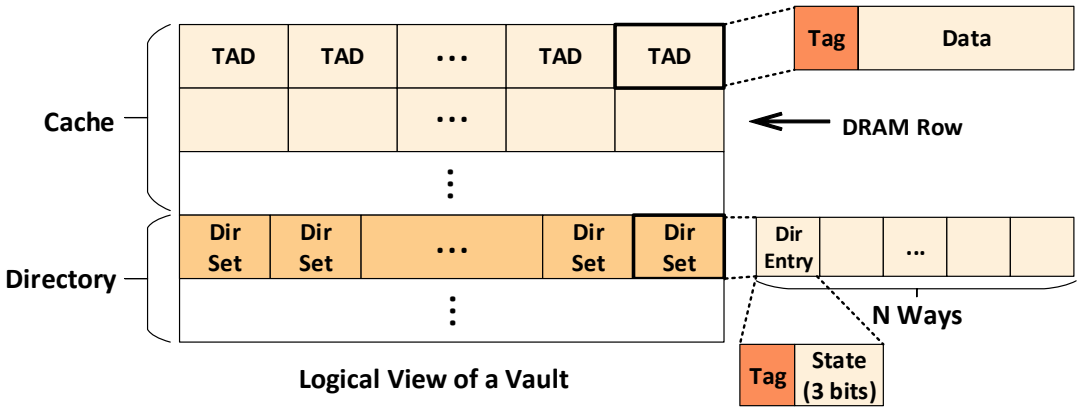


Figure 5.6: DRAM organization.

engineering the DRAM stack for low access latency. As explained in chapter 3, this involves using many banks, shorter pages and shorter bitlines/wordlines. While these optimisations reduce the storage capacity per vault compared to traditional capacity-optimised DRAM, they afford ultra-low access latency while still providing over a hundred MBs of capacity per core in today's process technology.

To summarise, SILO overcomes the area and delay constraints of shared LLCs through private die-stacked DRAM caches. An additional benefit of private caches is their immunity to cache contention, which plagues shared LLC designs. Finally, SILO reduces the access latency to the DRAM cache by engineering the DRAM for low latency at the expense of capacity. In the following two sections, we first describe our latency-optimised DRAM cache, followed by a detailed discussion of other aspects of the SILO architecture, including cache coherence.

The following sections detail aspects of SILO, including the organisation of the DRAM cache (i.e., tag and data placement), cache coherence support and performance optimisations.

5.3.1 DRAM Cache Organisation

In SILO, the die-stacked DRAM cache stores data, associated tags, and the directory metadata. Fig. 5.6 shows the layout of data and metadata in the cache. In this section, we focus on data and tag placement, while the next section discusses cache coherence and directory organisation.

To maximize available capacity, SILO uses a block-based cache organisation. Based on the observations that separating the tag store and data store will lead to a significant latency increase due to the serialised accesses to both on a hit [93],

SILO leverages a previously-proposed technique to integrate each data block with the corresponding tag into a single unified fetch unit called (TAD) [43]. Each access to the DRAM cache provides a single TAD, thus avoiding the delay of tag serialisation.

SILO is inclusive of the on-chip private caches and is organised as a direct-mapped structure. The direct-mapped organisation avoids the latency and energy overheads of a set-associative design, and is compensated for by the high capacity of the DRAM cache. Meanwhile, inclusion simplifies coherence and is easily afforded given the high capacity of the DRAM cache.

5.3.2 Directory-Based Cache Coherence

SILO uses a conventional directory-based MOESI protocol to maintain full coherence among its private caches. Misses in a core's private cache hierarchy (which consists of one or two levels of on-chip cache backed by a die-stacked DRAM vault) trigger a directory access. Logically, the directory sits below the DRAM LLC (i.e., logically closer to main memory). Physically, the directory is distributed in an address-interleaved fashion, with directory metadata stored in the DRAM caches as explained below.

Directory Organisation: The fact that the LLC is private, inclusive and direct-mapped has two implications. First, because the LLC is private and inclusive, the size of the tag store is directly proportional to the total LLC capacity. Assuming every vault stores a unique set of blocks with respect to every other vault, the tag store must be able to accommodate the full set of tags across all vaults. Secondly, associativity at the directory is dictated by the core count, not by the higher-level caches. This is because the LLC is direct-mapped and is inclusive of higher-level caches.

Based on these observations, we use a duplicate-tag directory organisation *without* a sharing vector. Fig. 5.6 shows the design. Logically, a directory is organised as an N-way associative tag store, where N is equal to the core count. Each directory entry stores a tag and the coherence state of the block. The way position of a given directory entry indicates the core caching the associated block. For instance, a tag in the directory way position 1 indicates that Core 1 is caching the block. Finding sharers requires reading the tags for all N logical ways in the directory. Most coherence state updates require modifying only one directory

entry (tag and/or state bits); however, in the worst case, all N directory entries in a given set may need to be updated (e.g., when a block shared by all cores transitions to an exclusive state).

Coherence Protocol: In a processor with a conventional on-chip shared LLC, the LLC serves as the point of coherence. In such a system, a writeback from a core's L2 involves simply updating the LLC. However, in a system with only private caches, the point of coherence is main memory, so a writeback incurs the high latency, energy and bandwidth overhead of a main memory access. To avoid the need for such an expensive writeback on a dirty eviction from a core's L2, SILO uses the MOESI protocol for maintaining coherence. The O state indicates that the block is valid, dirty and Owned; that is, the cache that has the block in this state must respond to coherence requests for the block. Compared to the MESI protocol, the primary advantage of MOESI is that a modified block can be directly supplied to other cores that want to read it without first writing it back to memory.

5.3.3 Performance Optimisations

In SILO, a miss in the local DRAM cache vault requires an access to the directory node for the block. Because the directory metadata resides in the DRAM cache, fetching it requires a DRAM cache access. And if the requested block is found at another node, yet another DRAM cache access is incurred to get the block. In total, up to three DRAM cache lookups may be needed to access a block on chip.

In light of the high miss penalties on misses in the private DRAM cache, we consider two performance optimisations in the SILO architecture:

Local Vault Miss Predictor: The TAD organisation in the DRAM cache means that a miss is not discovered until the DRAM access completes. A miss predictor, such as a MissMap [44], can avoid DRAM accesses if they are known to be misses, thus avoiding the associated latency cost.

Directory Cache: A miss in a core's private cache hierarchy triggers a DRAM access at the requested block's directory node to fetch the directory metadata. A directory cache [94] can eliminate the DRAM access for directory metadata by serving it from a fast on-chip SRAM.

These two optimisations can be applied separately or in concert. We consider all three options and show results in Sec. 5.5.2.

5.3.4 Discussion

Latency-Optimised DRAM. SILO requires non-commodity DRAM to achieve low latency and maximise performance gains. Traditionally, the DRAM industry has resisted such designs; however, the booming datacenter market and the presence of a few hyper-scale players (e.g., Google, Amazon, Facebook) may tilt the dynamic toward DRAM customization to accommodate specific needs of datacenter customers. The trend of customising processors [95] and deploying custom accelerators [96] for datacenters is already underway. We show the large gains that can be reaped by specialising the DRAM.

Implication of SILO for Die Real-estate. To help offset the cost of non-commodity DRAM, we note that on-chip shared LLCs occupy around a third of chip area in today's server processors [97, 98]. The associated die real-estate is expensive because server CPUs are generally built with leading-edge process technology. Because SILO completely avoids the need for an on-chip LLC, it vacates the associated die real-estate. In turn, this can afford either a reduction in die size, thus improving cost and yield, or addition of more cores within the same die area as a baseline processor with a shared LLC.

Another benefit of eliminating the on-chip LLC is that it reduces demands on the on-chip interconnect. High hit rates in the private die-stacked DRAM help reduce on-chip instruction and data traffic, while a smaller die (afforded by eliminating the on-chip LLC) helps reduce wire delays. Together, these features may lead to a less costly (area-wise) and/or faster NOCs. Our evaluation is conservative and does not take advantage of any such NOC optimisations afforded by SILO.

Defect Tolerance. The DRAM vault organisation in SILO has an advantage with respect to defect-tolerance. Due to logical partitioning of the DRAM stack into vaults, a critical defect is likely to be limited to a single DRAM vault, with the other vaults completely unaffected. Therefore, a fault does not render the entire DRAM stack unusable. However, with a faulty DRAM vault, the corresponding core is rendered non-functional.

SILO Design Dependence on Stacked DRAM. A critical implication of replacing the on-chip LLC with stacked DRAM vaults in SILO is that the stacked DRAM becomes an integral part of the system design. The absence of stacked DRAM would eliminate the crucial last-level cache from the design, which not

	DRAM access	64-bit serial	Controller	Total
Latency	11cyc	8cyc	4cyc	23cyc

Table 5.1: Access latency breakdown of a private DRAM vault.

only provides high storage capacity, but also maintains coherence. All SILO-based systems would require stacked DRAM to function. From a manufacturing perspective, no stacked DRAM-less variants may be supported in the SILO-based processor line.

Power Delivery and Wiring Challenges. In a vertical stack of multiple die, the power is delivered through the C4 bumps on the bottom die and then through the power TSVs to each individual die. The power delivery network needs to be able to mitigate the voltage variation between the die resulting from the power supply propagating through TSVs. Furthermore, the floorplan needs to avoid placement of high performance/power blocks on the same position in the different die to add stress to the power delivery TSVs. Existing literature presents various detailed 3D power delivery network models for this purpose [99, 100, 101, 102].

In a densely-connected architecture like SILO, the power TSVs need to be co-designed with the signal TSVs because of a trade-off between a robust power delivery network and high bandwidth signaling. The placement of blocks in the 3D stack and their respective signal TSVs greatly impacts the system performance, with inefficient placement leading to longer wirelengths and routing congestion. These factors amplify the routability problem in a 3D stack, which already is a challenging task. A large body of work attempts to devise accurate models for TSV placement capturing all the power, noise, and stress effects such as [103, 104, 105], and some treat it as a multi-objective optimisation problem [106].

5.4 Methodology

5.4.1 Evaluated Systems

We model a 16-core CMP with 3-way OoO cores running at 2.0 GHz. Table 5.2 details the system parameters. We extract LLC and DRAM cache latencies from CACTI (details in Sec. 5.4.3) and model a constant access latency for DRAM in our simulations . For DRAM (both cache and main memory), we assume

a closed page policy, which has been shown to outperform open-page on server workloads [74]. We assume a fairly aggressive memory access latency of 50ns; the combination of modest core frequency and low memory access latency is disadvantageous for SILO since LLC misses are relatively "cheap". A faster core and/or slower memory would amplify the penalty of a miss in the LLC, providing a larger benefit to SILO, which has a lower LLC miss rate than conventional LLC organisations.

Our main evaluation focuses on three level cache hierarchies, which are typical for server processors. Section 5.5.7 evaluates cache hierarchies with two levels, which have been shown to be superior to three level designs for scale-out workloads [18].

Baseline: The baseline processor uses a 32MB LLC split into 16 banks, with a 7-cycle bank access latency. The average round trip time for an LLC hit, including the NOC, is 25 cycles.

Baseline+DRAM$: Baseline augmented with an 8GB conventional DRAM cache. The DRAM cache is hardware managed and uses a page-based arrangement considered state-of-the-art for servers [15, 45]. Conventional DRAM caches use the same DRAM technology as main memory and as such have similar access latency. Indeed, the on-package DRAM cache in Intel's Xeon Phi Knight's Landing is slightly slower than main memory [13]. We optimistically assume that the access latency of the conventional DRAM cache is 20% faster than that of main memory. We further assume perfect miss prediction and infinite bandwidth.

SILO: A fully private three-level cache hierarchy with die-stacked DRAM vaults as the LLC. We use a custom, latency-optimised vault design with 256MB of capacity per vault as discussed in Sec. 3.4. The vault access latency is 11 cycles. We use a 64-bit wide interface adding 8 serialisation cycles for a TAD block. We add 4 cycles of vault controller delay bringing the total cache access latency to 23 cycles. The latency breakdown is shown in table 5.1.

SILO-CO: SILO with capacity-optimised vaults of 512MB at an access latency of 20 cycles. The total cache access latency, including vault controller and serialisation delay, is 32 cycles.

Vaults-Sh: Die-stacked *shared* LLC organisation with latency-optimised vaults. This design point evaluates the effect of DRAM latency optimisation without the private organisation. The latency-optimised vaults are stacked directly on top of

Processor	16-core, 2GHz, 3-way OoO, 128 ROB, ISA: UltraSPARC v9
L1-I/D	64KB, 8-way, 64B line, 3 cycles, private, stride data prefetcher
L2	512KB, 8-way, 64B line, 5-cycle, private, stride prefetcher
Interconnect	4x4 2D mesh, 3 cycles/hop
Baseline on-chip LLC	32MB shared NUCA, 7 cycles, 16-way, 64B line, non-inclusive MESI, LRU
SILO die-stacked DRAM LLC	Private, direct-mapped, 64B line, 512B page, inclusive MOESI SILO: 256MB vault/core, 23 cycles SILO-CO: 512MB vault/core, 32 cycles
Conv. DRAM cache	8GB, page-based, 2KB pages, direct-mapped, 40ns
Main memory	Access latency 50ns

Table 5.2: Microarchitectural parameters of the simulated systems. Extended parameters table in appendix A.

cores (just like in SILO) but the aggregate vault capacity of 4GB is shared by all cores in a NUCA address-interleaved manner. The average round trip time for a hit, including a vault access and NOC traversal, is 41 cycles.

5.4.2 Simulation Infrastructure

We use Flexus [75], a full system multiprocessor simulator, based on Simics. Flexus models the SPARC v9 ISA and extends Simics with out-of-order (OoO) cores, memory hierarchy, and on-chip interconnect (NOC). To reduce simulation time, Flexus integrates the SMARTS [76] methodology for sampled execution. For each sample, we first warm-up architectural and microarchitectural state, then run cycle-accurate simulation and measure performance. In order to evaluate performance, we measure the number of application instructions executed per cycle (including time spent executing operating system code); this metric has been shown to reflect system throughput [75].

To ensure high confidence in our developed simulation model, we adopt a step-wise development scheme and conduct validation at each step. We validate the

Baseline on-chip SRAM LLC	49mW per bank static power, 0.32nJ/access dynamic energy
SILO die-stacked DRAM LLC	120mW per vault static power, 0.4nJ/access dynamic energy
Main memory	4W static power, 20nJ/access dynamic energy

Table 5.3: Memory subsystem energy/power parameters.

models using some sanity checks – checking if the performance and hit/miss data correlates well, checking if each relevant simulation statistic is meaningful, and using a number of simple test cases with deterministic output behaviour which we track during the simulations. We conduct a final validation step of the completed model by comparing its performance in an idealized case against an opportunity study which establish the performance limits of the proposed model.

5.4.3 DRAM and SRAM Technology Modeling

We use CACTI-3DD to model DRAM and SRAM access latencies. We model DRAM and SRAM technologies at 22nm. For the SRAM LLC, we account for advanced latency reduction techniques [107] and use the low-standby-power cell type. Area and/or capacity constraints imposed by individual studies are highlighted in the text where appropriate.

To measure the energy and power consumed in the memory subsystem, including (as appropriate) the SRAM LLC, DRAM cache and main memory, we use a hybrid energy modelling framework that makes use of technology-specific parameters and cycle-accurate simulation statistics. We use CACTI-3DD to extract energy and power parameters for SRAM and stacked DRAM technology [107, 108]. We estimate main memory DRAM parameters using commercial DDR3 device specifications [109]. Table 5.3 summarizes the energy and power values obtained from these tools and used in the evaluation.

5.4.4 Workloads

We evaluate the various cache architectures using a range of workloads. Our scale-out workloads include Web Search, Data Serving, MapReduce and SAT Solver workloads, which are taken from CloudSuite [46], and a Web Frontend workload

from SPECweb2009. The latter replaces the Cloudstone Web Frontend workload from CloudSuite, which exhibits poor scalability at high core counts [18]. For the same reason, we do not use the Media Streaming workload from CloudSuite, as it does not scale beyond 2-4 threads [77]. We further investigate the utility of SILO for traditional enterprise applications. Details of these workloads are listed in Table 5.4. We also evaluate the systems on contemporary parallel applications from the PARSEC-3.0 benchmark suite [47], listed in Table 5.5. We simulate a 16-core setup with the number of threads equals to the number of cores. We use the native input sets and only simulate the Region of Interest (ROI). Compilation and runtime issues prevented us from being able to run three workloads: `dedup`, `streamcluster`, and `swaptions`.

For simulation, samples are drawn over 80 billion instructions (5 billion per core) for each workload. For each sample, we run cycle-accurate simulations from checkpoints that include full architectural and partial microarchitectural state, which includes caches and branch prediction structures. We run for 100K cycles to achieve steady state and measure over the following 200K cycles per sample.

We also consider multi-programmed batch workload deployments, representative of public cloud use cases. We generate 10 randomly-drawn mixes, each consisting of four workloads from SPEC'06 [110]. For each mix, workloads are drawn without replacement. The mixes are listed in Table 5.6. We draw samples over 20 billion instructions (5 billion per core) for the 4-core setup. Cycle-accurate simulation is run for 300K cycles with measurement over the last 200K cycles.

5.5 Evaluation

5.5.1 Performance on Scale-Out Workloads

Fig. 5.7 plots the performance of the evaluated systems on scale-out workloads, with results normalised to the baseline system (see Sec. 5.4 for a description of evaluated systems). We observe that both SILO designs consistently provide better performance than the baseline designs for all workloads except Web Frontend. This is an expected result as SILO provides a higher LLC capacity with the same hit latency as the baseline. SILO improves performance by 13-34%, with a geomean performance improvement of 16%, across the scale-out workloads. The highest performance gain is observed for SAT Solver at 34%. On Web Search,

Scale-out	
Web Search	*Nutch 1.2 / Lucene 3.0.1,* 230 clients, 2 GB index, 23 GB data segment
Data Serving	*Apache Cassandra 0.7.3,* 150 clients, 8000 operations per second, 15GB YCSB dataset, 7GB Java heap, 400MB garbage collector space
Web Frontend	*Apache HTTP Server v2.0,* e-banking, 16K connections, fastCGI, worker threading model, 6GB dataset
MapReduce	*Hadoop MapReduce, Apache Mahout 0.6,* Bayesian classification algorithm, 4.5GB set of pages, 2GB Java heap
SAT Solver	*Cloud9 parallel symbolic execution engine* Klee SAT Solver, Symbolic execution of `printf` with four 5-byte and one 10-byte symbolic commandline arguments, one instance per core, about 700MB working set per instance

Table 5.4: Server workloads used for evaluation.

Parsec			
Name	Domain	Input set	Est virt mem use
blackscholes	Financial analysis	10,000,000 options	>652MB
bodytrack	Computer vision	4 cameras, 261 frames, 4,000 particles, 5 annealing layers	380MB
canneal	Engineering	15,000 swaps/temperature step, 2,000deg start temp, 2,500,000 netlist elements	1239MB
facesim	Animation	80,598 particles, 372,126 tetrahedra, 100 frames	552MB
ferret	Similarity search	3,500 image queries, database with 59,695 images, find top 50 images	1330MB
fluidanimate	Animation	500,000 particles, 500 frames	642MB
freqmine	Data mining	Database of 250,000 web html documents, minimum support 11,000	941MB
vips	Media processing	18,000 × 18,000 pixels	>361MB
x264	Media processing	1,920 × 1,080 pixels (HDTV resolution), 512 frames	>310MB

Table 5.5: List of PARSEC 3.0 workloads used for evaluation. Memory footprints from [5].

SILO achieves a speedup of 22%. Sec. 5.2.1 identified that aggregate LLC capacities greater than 512MB are beneficial for the performance of Web Search. Thus SILO, which has an aggregate LLC capacity of 4GB (256MB per vault), delivers higher performance on this workload than the baseline.

Web Frontend observes a 2% slowdown in system performance with SILO. Web Frontend observes negligible reduction in off-chip misses and a 20% of the hits are from a remote vault as shown in figure 5.8. This observation is consistent with the trends observed in section 5.2, where Web Frontend exhibited higher data sharing behaviour. The minimal reduction in off-chip misses and long latency accesses to remote vaults limit SILO's potential for Web Frontend, resulting in a minimal slowdown. At the same time, baseline+DRAM$ observes a 2% speedup. The Web Frontend workload is set up with a modestly sized dataset. Once the large conventional DRAM cache has warmed up, the DRAM cache miss rate is very low (less than 4%), because of fairly small dataset and a large cache page size. A larger dataset (refer to chapter 2 for discussion about growing dataset

Name	Description
mix1	sjeng-calculix-mcf-omnetpp
mix2	lbm-gamess-namd-gromacs
mix3	mcf-zeusmp-calculix-lbm
mix4	tonto-gamess-bzip2-namd
mix5	mcf-povray-gcc-cactusADM
mix6	gobmk-perlbench-milc-astar
mix7	xalancbmk-sjeng-cactusADM-bwaves
mix8	calculix-leslie3d-astar-gcc
mix9	gromacs-gobmk-gamess-astar
mix10	omnetpp-zeusmp-soplex-povray

Table 5.6: SPEC'06 mixes used for evaluation.

sizes) is expected to thrash the conventional DRAM cache, leading to higher miss rates. In that case, SILO will be more beneficial for system performance due to the fast private DRAM vaults with a fine-grain block size of 64B.

The capacity-optimised SILO design (SILO-CO) delivers a geomean performance improvement of 12%, slightly below the 16% speedup provided by the latency-optimised SILO. Despite twice the per-vault capacity, the SILO-CO design has higher vault access latency, as shown in Sec. 3.4. Consistent with our sensitivity studies in Sec. 5.2, higher capacity is only beneficial if not accompanied by a higher access latency. Similarly, the shared vaults design (Vaults-Sh), delivers a geomean performance improvement of only 6%, despite employing latency-optimised vaults. NOC traversal adds to the overall access latency of Vaults-Sh, diminishing the performance benefits of the high capacity vaults.

Finally, we observe that baseline and baseline+DRAM$ designs provide similar performance. We identify the high access latency to the conventional DRAM cache as the main reason for the limited benefit offered by the baseline+DRAM$ design. As noted in section 5.2.2, benefits of large cache capacities disappear at high access latencies.

5.5.2 Analysis

In this section, we characterise LLC effectiveness in SILO as compared to the baseline, and explore the usefulness of SILO design optimisations.

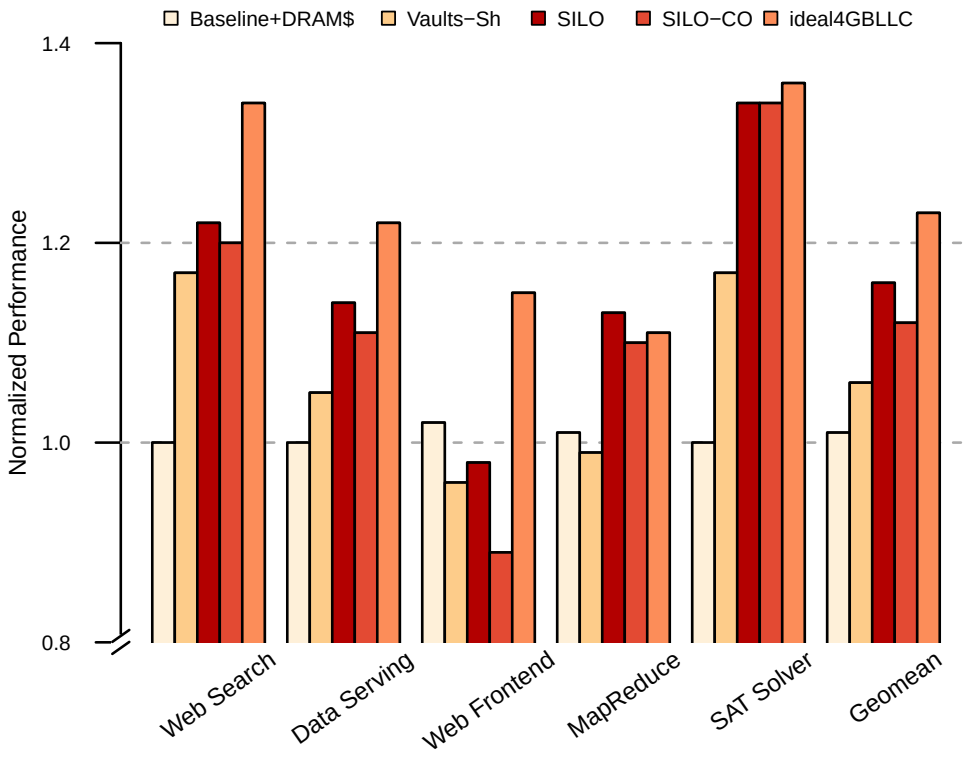


Figure 5.7: Performance on scale-out workloads.

5.5.2.1 LLC Hit Rate

Figure. 5.8 plots normalised LLC hits and misses for the baseline and SILO designs. In general, SILO consistently reduces off-chip misses compared to the baseline across all workloads. Miss rate reductions range from 1% to 69%, with the largest reduction on SAT Solver (69%). Not surprisingly, SAT Solver observed the greatest performance improvement as noted in Sec. 5.5.1.

As the figure shows, the majority of hits in SILO come from the local vault (57-88% of all hits). This is important for performance, because local hits are faster than remote hits, which incur a directory lookup and a multi-hop NOC traversal. Nonetheless, remote hits in SILO are also beneficial as they are faster than main memory accesses.

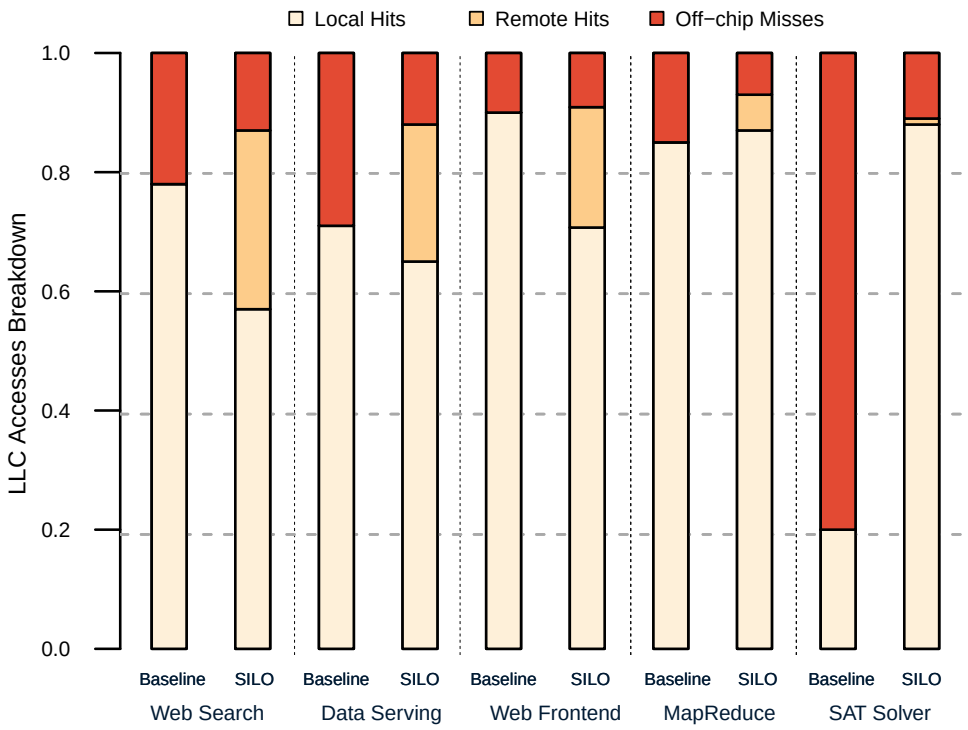


Figure 5.8: Normalized LLC hits and misses for SILO vs baseline. Note that all hits in the baseline shared NUCA LLC are shown as 'local'.

5.5.2.2 SILO Performance Optimisations

Sec. 5.3.3 identified two possible optimisations to reduce the latency incurred by DRAM accesses to (i) the local vault, and (ii) the in-DRAM directory in the case of a local vault miss. We now evaluate the usefulness of these optimisations *in the limit*. We consider the following configurations:

- *NoOpt*: SILO with no optimisations.
- *LocalMP*: SILO with a Miss Predictor for local vault accesses. The predictor is perfect, requiring 0 time and having 100% accuracy.
- *DirCache*: SILO with a directory cache. The directory cache is perfect, requiring 0 time and having 100% accuracy.
- *LocalMP+DirCache*: SILO with both local vault Miss Predictor and an ideal directory cache.

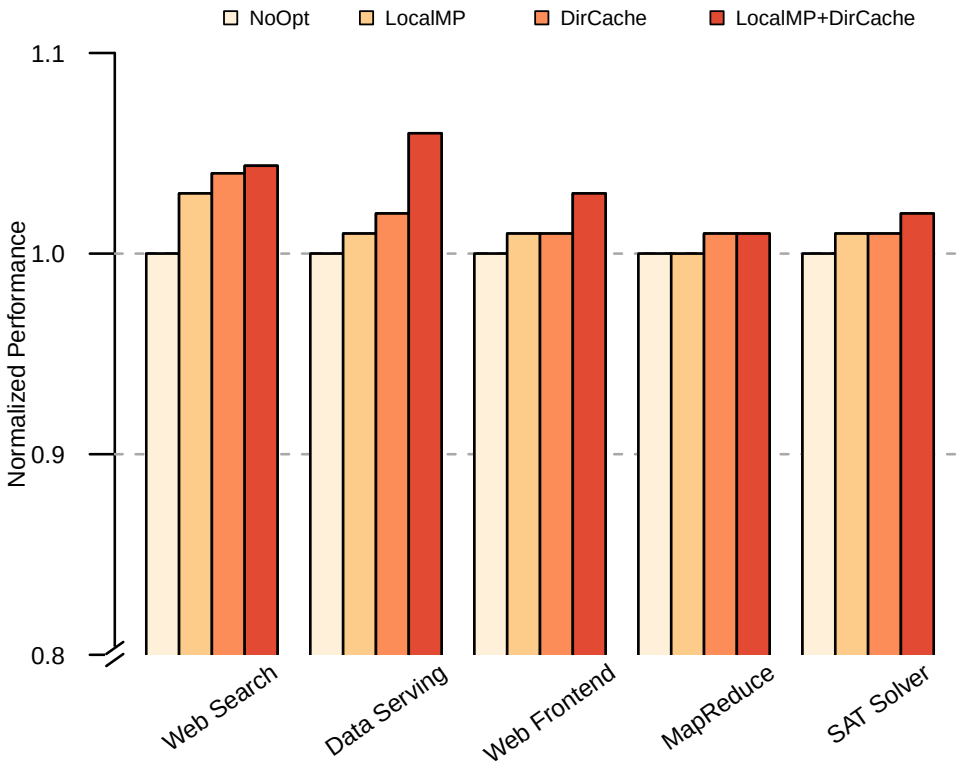


Figure 5.9: Effect of SILO design optimisations on scale-out workloads. Study assumes ideal vault miss predictor and ideal directory cache.

Fig. 5.9 plots the performance of the four designs. We observe marginal performance improvements in the optimised designs. Data Serving observes the largest benefit at 6% speedup with both local vault miss predictor and a directory cache. This result is consistent with the RW-sharing characterization study of Sec. 5.2.3, which shows Data Serving to have a high sensitivity to the LLC access latency for RW-shared data. Additionally, Data Serving shows a significant amount of remote vault hits (as shown in Fig. 5.8). LocalMP and DirCache optimisations reduce the latency of remote vault hits, thus improving performance. We conclude that the benefits of the considered design optimisations do not outweigh their cost and extra design complexity.

5.5.3 Energy Efficiency

We examine the effects of SILO architecture on the memory subsystem energy dissipation using the parameters in Sec. 5.4.3. Fig. 5.10 illustrates memory subsystem dynamic energy in SILO normalised to that in the baseline system. Compared to the baseline, SILO reduces dynamic energy by 30-83% across the evaluated workloads except Web Frontend. The high hit rate in SILO significantly reduces off-chip traffic, thereby reducing dynamic energy in main memory and I/O, which explains SILO's energy-efficiency advantage.

On Web Frontend, the difference in main memory energy dissipation between the baseline and SILO is minimal. Together with the higher LLC energy dissipation in SILO, the normalised dynamic energy per access is about 10% higher in SILO compared to the baseline. As discussed in the previous section, a bigger dataset will result in a greater reduction in main memory accesses in SILO and consequently, the energy dissipated. The normalised dynamic energy per access will reduce due to the lower amount of energy dissipated in the main memory.

While not shown in the figure, we note that SILO expends more power in the LLC than the baseline due to a combination of (i) higher static power of the large number of DRAM banks, (ii) higher dynamic energy per LLC access, and (iii) more accesses per unit time due to higher IPC. The total LLC power consumption in SILO does not exceed 2.7W across all evaluated workloads, which is a small fraction of the total power budget of a 16-core server processor.

5.5.4 Comparison with eDRAM-based LLC

We compare SILO against a 128MB eDRAM-based on-chip NUCA LLC similar to the POWER 9 [30] or certain Intel Haswell processors [111], referred to as eDRAMLLC. We optimistically assume the access latency for the eDRAM banks to be 7 cycles, similar to the smaller 32MB SRAM LLC in the baseline system. Figure 5.11 presents the result of this study. eDRAMLLC achieves a geomean speedup of 4% compared to 16% achieved by SILO. SILO outperforms eDRAM-LLC across all workloads except Web Frontend. eDRAM-based on-chip LLCs experience the same scalability issues as traditional SRAM LLCs with growing datasets, as discussed in chapter 2.

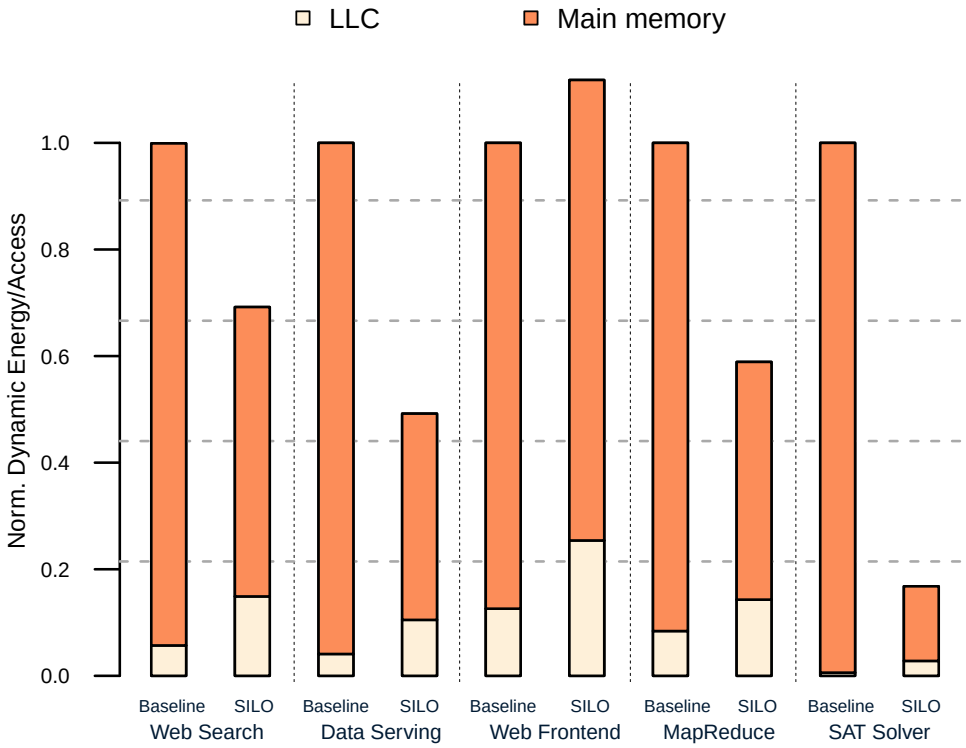


Figure 5.10: Dynamic energy of the memory subsystem.

5.5.5 Performance on Other Workloads

We extend our evaluation of SILO to Parsec and multi-programmed SPEC mixes.

5.5.5.1 Parsec

Figure 5.12 presents the performance of the Parsec workloads with the system performance normalised to the baseline. SILO achieves a geomean performance improvement of 14%, outperforming the baseline on all the workloads except `x264`. On `x264`, SILO observes a 1% slowdown which is within statistical measurement error.

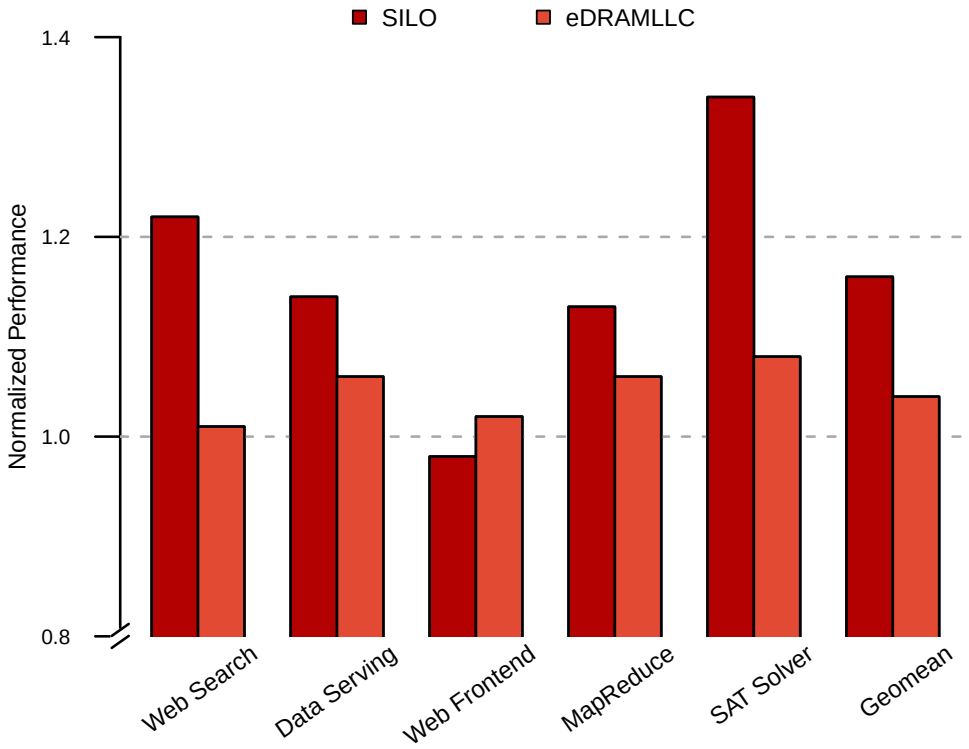


Figure 5.11: Performance results for a baseline system with 128MB eDRAM LLC vs SILO.

5.5.5.2 Multi-programmed SPEC

Figure 5.13 plots the performance of 4-core SPEC'06 mixes for baseline and SILO designs. Overall, SILO delivers a significant performance gain of up to 40% (25% on average) due to its massive core-private cache capacity. The capacity advantage comes at a similar latency as the shared LLC and in a contention-free manner – an issue further explored in the next section. While SILO performs better on all mixes, we observe higher performance gains on certain mixes, e.g. mix3, mix6, mix8, and mix9. These mixes include memory-intensive applications such as `mcf`, `lbm`, `milc` and `astar`, which benefit the most from the larger capacity, and therefore, exhibit higher performance improvement.

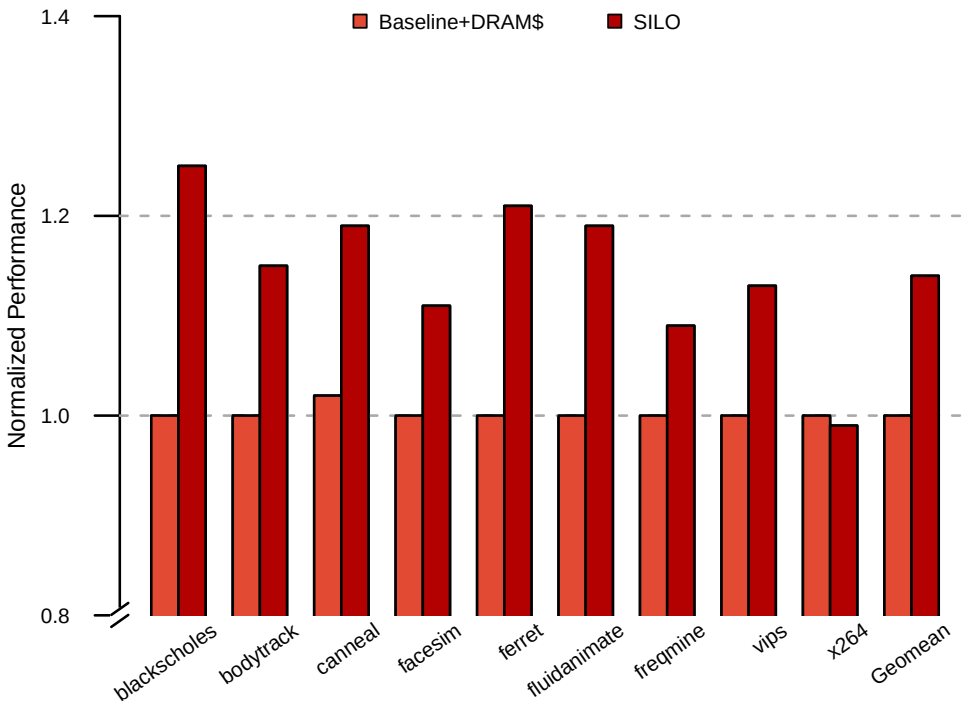


Figure 5.12: Performance results for Parsec.

5.5.6 Performance Isolation

Heterogeneous applications colocated on the same physical server contend for the available shared LLC. This inter-core contention can compromise performance, which is a particular concern for applications with strict latency targets. An all-private cache hierarchy, provided by SILO, offers the premise of removing LLC contention and guaranteeing strong performance isolation.

In order to evaluate the degree of performance isolation SILO provides, we measure the performance of Web Search running on a processor (i) alone and (ii) together with `mcf`, a memory-intensive SPEC'06 benchmark. Web Search runs on 8 cores while `mcf`, when present, runs on the other 8 cores of the 16-core setup. We use two LLC configurations: a traditional shared LLC and SILO.

Table 5.7 shows the results of the experiment where the performance of Web

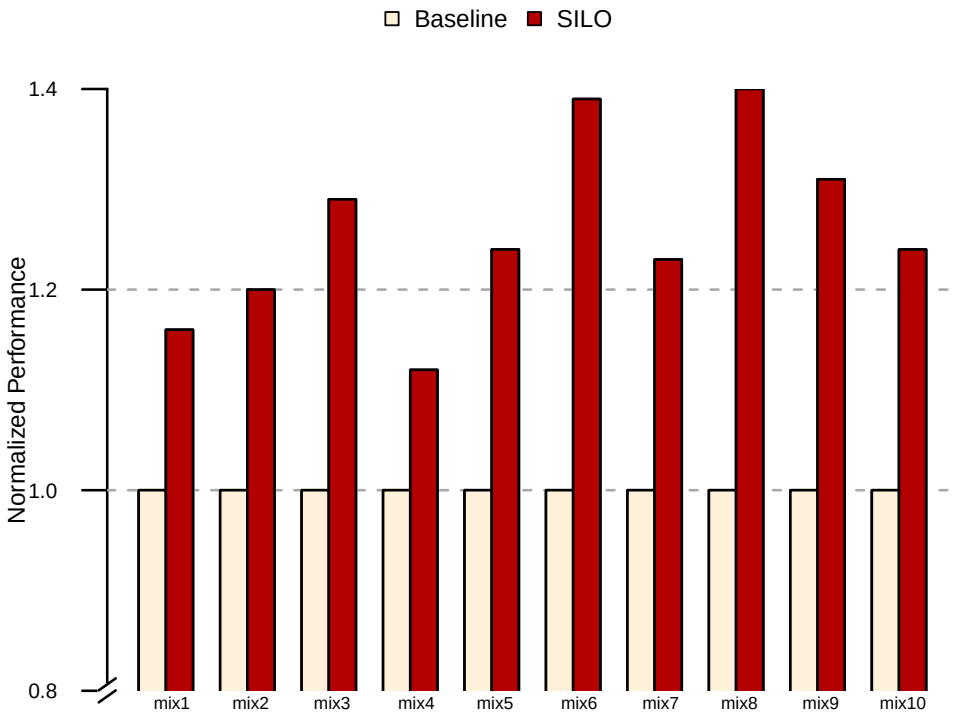


Figure 5.13: Results for 4-core SPEC2006 mixes.

Search is normalised to stand-alone Web Search setup with a shared LLC. We observe two trends. First, SILO improves performance of Web Search by 20% when running alone. Secondly, the performance in a system with SILO is unaffected by colocation with `mcf`. In contrast, Web Search suffers a 10% performance degradation when running on a shared LLC system under colocation. We conclude that SILO not only delivers a significant performance improvement compared to a shared LLC baseline, but also provides performance isolation under colocation.

5.5.7 Two-Level Cache Hierarchy

So far, our evaluation has focused on a three-level cache hierarchy, typical in server processors. In this section, we evaluate a two-level cache hierarchy based on Scale-Out Processors [18]. The systems have a private 64KB private L1,

	Shared LLC	SILO
Web Search alone	-	+20%
Web Search + `mcf`	-10%	+20%

Table 5.7: Performance of Web Search under different setups.

backed by an LLC. The baseline systems are equipped with an 8MB LLC with a bank access latency of 5 cycles (according to Cacti [107]). In SILO, the 64KB private L1's are backed by 256MB private LLC vaults in DRAM, with a vault access latency of 23 cycles.

Figure 5.14 plots the performance of the two-level systems. SILO outperforms the baselines across all the scale-out workloads, with the largest performance improvements on MapReduce and SAT Solver of 54% and 37%. The general trend is in line with our three-level evaluation.

5.5.8 Interposer-based SILO and CARVE

SILO stacks private DRAM LLC vaults directly on the processor die to avoid long planar interconnect spans. We also consider a more conservative, interposer-stacked design in this section. Commonly referred to as "2.5D stacking", DRAM stacks may be integrated on a silicon interposer next to the processor die. This arrangement, SILO2.5D, incurs additional latency due to the horizontal wires, adding up to a total of 31 cycles for a vault access. Additionally, we compare the performance of SILO2.5D with CARVE from chapter 4.

Figure 5.15 plots the performance of the evaluated systems. As expected, SILO2.5D yields lower performance improvement compared to SILO, across all the workloads. SILO2.5D achieves a geomean speedup of 11% compared to the 16%. Figure 5.16 presents the MPKI to main memory for the evaluated systems. SILO and SILO2.5D experience similar MPKIs. SILO2.5D's lower performance can be attributed to the longer vault access latency compared to SILO.

CARVE observes the lowest MPKI to main memory as shown in Figure 5.16. In CARVE, the 32MB on-chip LLC is backed by a 4GB shared DRAM vaults capacity. In SILO2.5D, the DRAM vaults serve as per-core private LLCs, leading to a lower effective capacity than CARVE. On the performance side, CARVE achieves a geomean speedup of 12% compared to 11% for SILO2.5D. CARVE outperforms SILO2.5D on Web Search, Data Serving, and Web Frontend. On these three work-

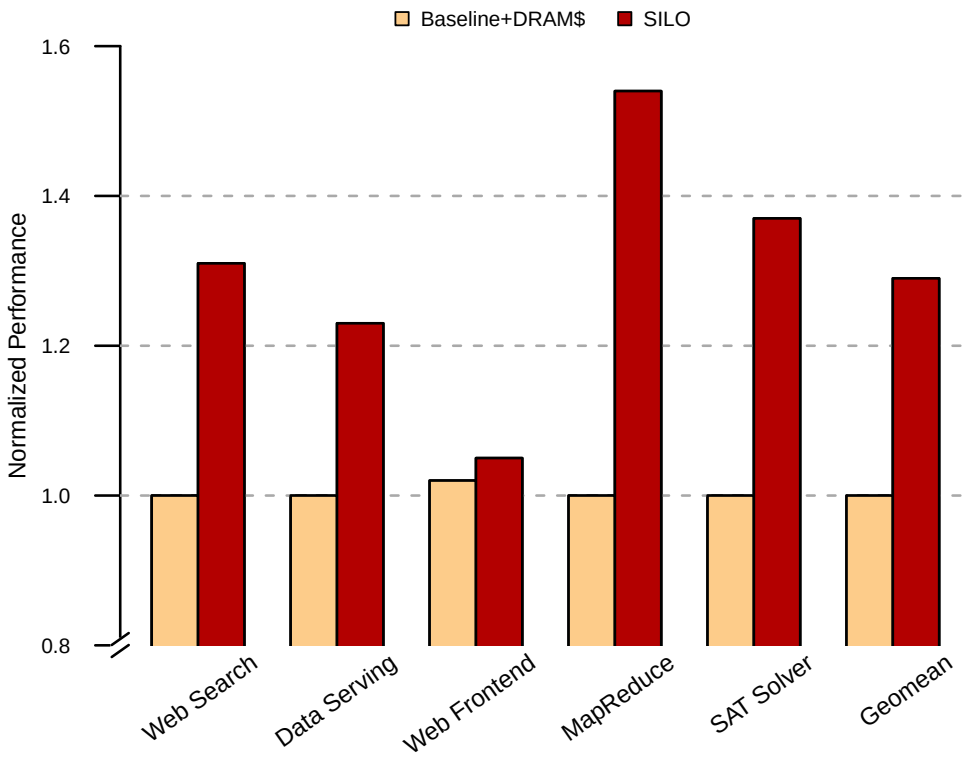


Figure 5.14: Performance on scale-out workloads with a 2-level hierarchy

loads, CARVE observes significantly lower MPKIs than SILO and SILO2.5D. However, on MapReduce and SAT Solver, SILO2.5D significantly outperforms CARVE, where the MPKI reductions in CARVE are modest.

Overall, SILO yields the highest geomean speedup of 16% and outperforms the other systems across all workloads expect Web Frontend (we discuss Web Frontend in detail in section 5.5.1).

5.6 Related Work

Maximizing performance for planar LLCs: To reduce the average access time for large, distributed LLCs, prior work has proposed Non-Uniform Cache Architecture (NUCA) [93]. Static NUCA (S-NUCA) designs use address interleaving to spread data across cache banks distributed on chip. While simple to

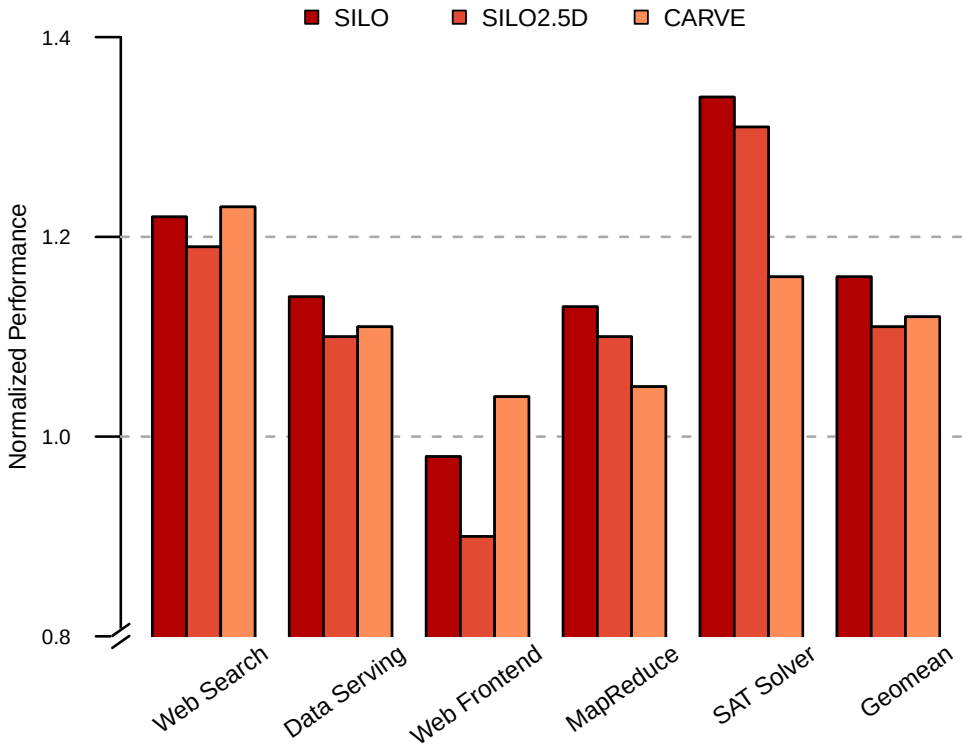


Figure 5.15: Performance results for interposer-based SILO - SILO2.5D.

implement, such designs require multi-hop NOC traversals in the common case, which result in high access latencies to remote cache banks. Dynamic NUCA (D-NUCA) designs use adaptive data placement to reduce the average access latency through a combination of data placement, replication and migration to make data available in the cache banks nearest to the requesting core [112, 113, 114, 115, 116]. Fundamentally, such schemes are limited by the small capacity of nearby banks on a planar die. SILO circumvents capacity-latency trade-off of planar caches by providing core-private die-stacked DRAM vaults with hundreds of MBs of capacity.

Stacked DRAM Caches: Die-stacked DRAM technology has been identified as a suitable means to provide gigascale caches [43, 15, 45]. The technology extends high density commodity DRAM with higher bandwidth and better power efficiency. The target applications are bandwidth-intensive, such as those running

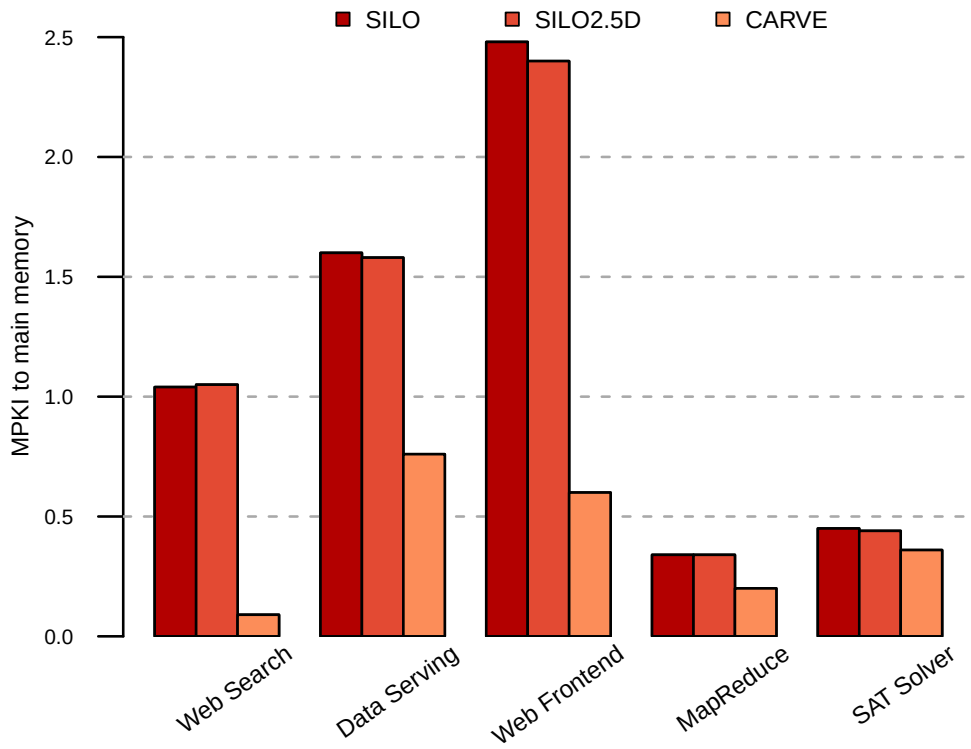


Figure 5.16: Miss rates for interposer-based SILO.

on GPUs and many-core HPC processors. Indeed, the latest Nvidia and AMD GPUs and Intel's Knight's Landing feature die-stacked DRAM [79, 80, 81, 13, 82]. Due to the significant delays involved in routing the request from the requesting core to the desired DRAM bank, access latencies are comparable to main memory [117]. In fact, Intel's Knight's Landing has a higher access latency to the DRAM cache than to main memory [13].

To improve the high access latency of a serialised tag and data lookup, research proposals have argued for tag placement in SRAM [15, 44, 83, 84] and for using direct-mapped in-DRAM tag designs [43, 13]. These policies reduce the tag lookup cost, but leave the underlying DRAM technology and processor organisation unchanged. Jenga [118] introduced a reconfigurable cache hierarchy composed of SRAM and die-stacked DRAM tiles. While improving access locality is part of Jenga, cores of a given application share the full set of cache

banks allocated to that application; as such, Jenga's caches are fundamentally shared across cores. SILO differs from these works in its use of an all-private cache hierarchy and custom DRAM technology.

DRAM Latency Optimisation: Various custom DRAM technologies have been introduced in commercial products to provide lower latency but at higher cost-per-bit than commodity DRAM [119, 120, 121, 122]. Technical details are generally scarce for these products, but they tend to advertise more banks and subarrays than commodity DRAM, making them similar to the vaults in SILO. However, due to the fact that the latency-optimised dies are packaged into discrete DRAM chips, the actual end-to-end latency savings are small due to the CPU-side and chip-to-chip interconnect delays. In contrast, SILO minimizes interconnect delays by using DRAM stacked directly on top of the processor die, and treating each DRAM vault as a core-private cache.

On the research side, prior works have looked at mitigating the overhead of additional peripheral circuitry for latency reduction by using segmented bitlines in DRAM [123], providing additional circuitry for selective banks [124] and partitioning the DRAM die into independent units [125, 126]. These techniques can be applied to the custom DRAM technology in SILO to increase vault capacities without compromising the access latency. Other techniques target reducing DRAM latency by overlapping accesses to different subarrays [127] and improving row-buffer locality by exploiting access patterns [128, 129, 130, 131]. While these techniques allow overlapping access latencies of different requests, they do not reduce the actual access latency.

5.7 Summary

As traditional technology scaling nears its end, processor architectures must embrace emerging technologies and new architectural paradigms in the quest for higher performance. We take a step in this direction by showing that traditional shared LLCs offer limited room for improving performance in future server processors as they are unable to satisfy the requirement of large cache capacity and low access latency demanded by scale-out workloads. In response, we introduce SILO – a Die-Stacked Private LLC Organisation which combines on-chip private caches with per-core LLC slices in die-stacked DRAM. SILO resolves the latency/capacity conundrum through the use of a private LLC organisation and latency-optimised die-stacked DRAM.

Chapter 6

Conclusion

The end of traditional technology scaling is approaching and power constraints have permitted only minimal improvement in single-thread performance in the recent decades. Adding to this challenge, the relentless growth of dataset sizes imposes ever increasing data storage and processing demands. Processor architectures must integrate emerging technologies and new architectural paradigms in pursuit of higher server performance and efficiency.

On-chip caches constitute a hefty proportion of the die transistor budget, with LLC accounting for up to a third of the die area. Limits on yield-effective die sizes and SRAM cell scaling challenges fundamentally limit the ability of on-chip caches to scale to larger capacities, in line with the growth of datasets. User-facing online applications running in datacenters bank on fast data accesses in order to provide real-time user response, and rely on a high performing cache hierarchy as frequent long-latency main memory accesses bottleneck application performance.

Servers are increasingly being equipped with gigascale DRAM stacks which are utilised as memory-side caches. While the DRAM stacks provide tremendous cache capacity, they experience high access latencies, in the same range as main memory accesses. Therefore, conventional stacked DRAM caches fail to benefit datacenter applications which are performance sensitive to cache access latency.

6.1 Contributions

In this thesis we argued that the sources of high latency in DRAM caches are not fundamental and can be mitigated at the architectural level. Using full-system

cycle-accurate simulation, this thesis demonstrates the following:

- Server applications operating on large datasets can benefit from the large capacities provided by DRAM caches but are sensitive to DRAM cache latencies due to limited MLP. However, current interposer-based DRAM caches provide access latencies similar to that of main memory, and therefore do not provide performance benefits for cache-latency-sensitive server applications.
- We observe that the factors contributing to the high access latency of stacked DRAM caches are: (i) on-chip interconnect (NOC) routing delay to reach the DRAM cache controller, (ii) queuing delay in the DRAM cache controller, (iii) horizontal traversal between the processor die and the DRAM stack, (iv) addressing and access latency in the DRAM core.
- We present optimisation techniques targeting the latency-contributing factors in DRAM caches, targeting both connections to the DRAM stack (interconnect) and the DRAM technology within the stack (DRAM core).
- We propose organising the DRAM stack into functionally-independent vertical slices called vaults which reduce the interconnect latency compared to conventional DRAM caches.
- We demonstrate that DRAM vaults leveraging latency-optimised DRAM technology have a 45% lower access latency compared to a capacity-optimised design in the 22nm technology node.
- We present *On-PaCkage Partitioned DRAM Victim Cache* (CARVE), which supplements the conventional on-chip cache hierarchy with latency-optimised victim vaults in DRAM – VV. The VV are logically shared, serve as the victim cache for each on-chip LLC slice, and introduce no coherence overheads.
- We evaluate the effectiveness of CARVE as a direct replacement for traditional on-chip LLCs and find the design to be unsuitable due to the high access latency in comparison with on-chip LLCs.
- We identify the LLC requirements of server workloads, corroborating prior work showing that while scale-out server workloads benefit from large LLC capacities, they are highly sensitive to LLC access latency.
- We introduce *Die-Stacked Private LLC Organization* (SILO), a cache architecture with all-private caches. SILO avoids high interconnect latencies and overcomes on-chip area constraints by using a die-stacked DRAM LLC,

with a private slice directly above each core. The private caches are kept coherent through a conventional coherence protocol with directory metadata embedded in the die-stacked LLC.

6.2 Future Work

6.2.1 Facilitating Data Sharing in Private LLCs

While scale-out server workloads are largely insensitive to long access latency for RW-shared data due to limited RW-shared data [16], private LLCs impede the performance of parallel workloads with significant data sharing. Shared on-chip LLCs facilitate low-latency data sharing; servicing the request if the requested data block resides in the LLC, or quickly redirecting access to the core where the data block is resident upon directory response. The coherence directory is often co-located with the LLC allowing the directory lookup and LLC access to occur concurrently. In the case of a private LLC, a requested cache block is first looked up in the private LLC, then the distributed coherence directory must be accessed, and finally the multiple levels of private caches of the core where the cache block is resident must be looked up. Accesses to RW-shared data experience this long latency pathology, adding up to a very significant latency overhead if the data is regularly RW-shared by multiple cores.

In order to cut down the access latency of RW-shared data, writer cores may eagerly push the updated data to the potential readers, as employed in updating cache coherence protocols. This approach, however, inflates NOC traffic due to unnecessary updates. Recent work attempts to filter updates that are unlikely to be consumed and reduce NOC traffic by only pushing updates after a certain number of writes to a cache block through a simplistic prediction mechanism, 1-Update [21]. This approach relies on RW-shared blocks to observe a steady number of writes before they are read by other cores (write stability). For applications where the number of writes to a RW-shared block before being read varies significantly, 1-Update's predictor is unable to issue correct predictions. As a result, 1-Update either boils down to a standard invalidating protocol or issues unnecessary updates. A potential future work could characterise the read/write access patterns in applications from various domains, and based on the findings devise a predictor suited to a broad range of applications, or even domain-specialized predictors.

6.2.2 Defect-tolerance in DRAM Vaults

CARVE and SILO utilise DRAM vaults in their cache hierarchy. Tolerance to manufacturing defects is an important aspect to consider in DRAM stacks. Defects in the DRAM stack fall in two main categories: (i) the DRAM cells, and (ii) addressing logic and bus. Defects confined to a few DRAM cells(s) or row(s) are tolerated through error correction code (ECC) support by adding extra DRAM chips in commodity DRAM technology. Prior work has identified providing ECC support in DRAM stacks as complex with various practical difficulties [132, 133], but ECC support was added in the HBM3 standard [59, 134]. Critical defects in resources that affect every cache access (such as a defect in the address or data bus) may render the entire vault completely unusable. Some research works explore tolerance of large granularity errors in die-stacked DRAM by employing spare resources [135, 136]. The body of work studying defect tolerance in stacked DRAM caches with low access latency being the optimisation target is lacking. A possible future work could study ECC- and non-ECC-based error tolerance in DRAM vaults while optimising for low access latency. Another possible future work could study how to circumvent critical, irrecoverable defects in DRAM vaults, while keeping access latency low, and minimising the amount of system resources that are rendered non-functional.

6.2.3 CARVE vs SILO in Chiplet-Based Servers

Low manufacturing yields for the bleeding edge technology nodes are pushing the industry towards chiplet disaggregation of processor die [40, 11]. Chiplet-based processors not only face inter-chiplet communication challenges, but also open doors to rethink and specialize the architecture of different components of the processor. Chiplet architectures have been identified as a path to optimising yield and power consumption while improving performance [137]. As a result, chiplets have been deployed in recent server products [40, 11], and are projected to rapidly observe a widespread adoption [138]. Chiplet disaggregation lends itself to incorporating die-stacked DRAM into the processor cache hierarchy in close proximity, allowing deep integration with the processor chiplets.

This thesis evaluated CARVE vs SILO2.5D in the context of a monolithic processor die and found that neither of these designs is the outright winner across the evaluated workloads. A possible future work could revisit this comparison

for chiplet-based processors, where the shared LLC is distributed across all the chiplets, and the on-chip(let) LLC access latencies are potentially higher due to inter-chiplet communication, impacting CARVE.

Appendix A

Experimental Details

This section presents the extended details of the simulated systems to aid reproducibility.

Table A.1 presents the core parameters. The simulated cores are based on the SPARC v9 ISA, and run at 2GHz. As the thesis mainly targets scale-out workloads from Cloudsuite, modest cores are used as opposed to fat cores, as these workloads enjoy very little benefit from the large ROBs of fat cores [139].

Dispatch/Retire width	3
Reorder buffer	128
Load-Store queue	32
Store buffer	16
Fetch instruction queue	8
Store prefetches	16 simultaneously
Branch predictor unit	Hybrid (16K gShare and 4K bimodal), 2K entry BTB
Functional units	Int ALUs: 2 Add + 1 Mult, 2 FPU
Registers	8 reg window sets, 16 regs/window, 3 special regs (Y, ASI, GSR)

Table A.1: Core.

Table A.2 details the cache and memory parameters. Our baseline uses a shared NUCA LLC with 2MB per core which is more than the newest Intel Xeon server range [92], and same as AMD's [140]. We assume a fairly aggressive memory access latency of 50ns; the combination of modest core frequency and low memory access latency is disadvantageous for our proposed designs since LLC misses are relatively "cheap". A faster core and/or slower memory would amplify the penalty of a miss in the LLC, providing a larger benefit to CARVE and SILO, which have a lower miss rate to main memory than the baseline.

L1I	64KB, 8-way, 64B line, 3 cycles, private, 1 bank
L1D	64KB, 8-way, 64B line, 3 cycles, private, LRU replacement, stride data prefetcher tracking up to 32 load/store PCs, 16 MSHRs, 1 bank, 8-entry evict buffer
L2	512KB, 8-way, 64B line, 5-cycle, private, LRU replacement, stride data prefetcher tracking up to 32 load/store PCs, 32 MSHRs, 8-entry evict buffer
Baseline on-chip LLC	32MB shared NUCA, 7 cycles, 16-way, 64B line, non-inclusive MESI, LRU, 64 MSHRs, 16-entry evict buffer
Conv. DRAM cache	8GB, page-based, 2KB page, direct-mapped, fixed access latency of 40ns, closed-page policy
Interconnect	4x4 2D mesh, 3 cycles/hop
Main memory	Fixed access latency of 50ns, closed page policy, 4 memory controllers

Table A.2: Caches, memory, and NOC.

Bibliography

[1] A. Shahab, M. Zhu, A. Margaritov, and B. Grot, "Farewell my shared llc! a case for private die-stacked dram caches for servers," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 559–572, 2018.

[2] D. Schor, *TSMC N3, And Challenges Ahead*. `https://fuse.wikichip.org/news/7375/tsmc-n3-and-challenges-ahead/`.

[3] *CPU Specs Database*. `https://www.techpowerup.com/cpu-specs/`.

[4] M. N. Bojnordi and F. Nasrullah, "Retagger: An efficient controller for dram cache architectures," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

[5] M. Giardino, K. Doshi, and B. Ferri, "Soft2lm: Application guided heterogeneous memory management," in *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pp. 1–10, IEEE, 2016.

[6] M. Law, *Energy efficiency predictions for data centres in 2023*. `https://datacentremagazine.com/articles/efficiency-to-loom-large-for-data-centre-industry-in-2023`.

[7] M. Technology, *Investing in the rising data center economy*. `https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/investing-in-the-rising-data-center-economy`.

[8] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[9] N. R. Council *et al.*, *Productivity and Cyclicality in Semiconductors: Trends, Implications, and Questions: Report of a Symposium*. National Academies Press, 2004.

[10] T. P. Morgan, *The Steady Hand Guiding AMD's "Prudently Expanding" Datacenter Business*. https://www.nextplatform.com/2022/10/03/the-steady-hand-guiding-amds-prudently-expanding-datacenter-business/.

[11] *Intel "Sapphire Rapids" Xeon 4-tile MCM Annotated*. https://www.techpowerup.com/292204/intel-sapphire-rapids-xeon-4-tile-mcm-annotated.

[12] *Quick Zen3 die shot annotations*. https://www.reddit.com/r/Amd/comments/jqjg8e-/quick_zen3_die_shot_annotations_die_shot_from/.

[13] A. Sodani, "Knights landing (KNL): 2nd generation Intel® Xeon Phi processor," in *Hot Chips 27 Symposium*, pp. 1–24, 2015.

[14] *HBM Gives Xeon SPs s Big Boost on Bandwidth Bound Work*. https://www.nextplatform.com/2022/11/15/sapphire-rapids-xeon-sps-plus-hbm-offer-big-performance-boost/.

[15] 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 *40th International Symposium on Computer Architecture*, pp. 404–415, 2013.

[16] 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 *17th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 37–48, 2012.

[17] S. Kanev, J. P. Darago, K. Hazelwood, P. Ranganathan, T. Moseley, G.-Y. Wei, and D. Brooks, "Profiling a warehouse-scale computer," in *42nd International Symposium on Computer Architecture*, pp. 158–169, 2015.

[18] P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, *et al.*, "Scale-out processors," in

Computer Architecture (ISCA), 2012 39th Annual International Symposium on, pp. 500–511, Ieee, 2012.

[19] *High Bandwidth Memory (HBM) DRAM JESD235A.* `https://www.jedec.org/`.

[20] C.-C. Huang, R. Kumar, M. Elver, B. Grot, and V. Nagarajan, "C^3d: Mitigating the NUMA bottleneck via coherent DRAM caches," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, pp. 1–12, IEEE, 2016.

[21] M. Zhu, A. Shahab, A. Katsarakis, and B. Grot, "Invalidate or update? revisiting coherence for tomorrow's cache hierarchies," in *2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 226–241, IEEE, 2021.

[22] A. Margaritov, D. Ustiugov, A. Shahab, and B. Grot, "Ptemagnet: Fine-grained physical memory reservation for faster page walks in public clouds," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 211–223, 2021.

[23] A. Shahab and B. Grot, "Population-based evolutionary distributed sgd," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pp. 153–154, 2020.

[24] *New Memory Technologies Poised for High Volume Production.* `https://www.nextplatform.com/2019/08/01/new-memory-technologies-poised-for-high-volume-production/`.

[25] *Being Persistent with Persistent Memory.* `https://www.nextplatform.com/2020/12/18/being-persistent-with-persistent-memory/`.

[26] *The Era of Big Memory is Upon Us.* `https://www.nextplatform.com/2020/09/23/the-era-of-big-memory-is-upon-us/`.

[27] *Bing and Google Agree: Slow Pages Lose Users.* `http://radar.oreilly.com/2009/06/bing-and-google-agree-slow-pag.html`.

[28] J. Brutlag, "Speed matters for google web search," 2009.

[29] *Speed Matters.* https://ai.googleblog.com/2009/06/speed-matters.html.

[30] S. K. Sadasivam, B. W. Thompto, R. Kalla, and W. J. Starke, "Ibm power9 processor architecture," *IEEE Micro*, vol. 37, pp. 40–51, Mar 2017.

[31] *Vertical L3 Cache Raises the AMD Server Performance Bar.* https://www.nextplatform.com/2021/11/08/vertical-l3-cache-raises-the-amd-server-performance-bar/.

[32] J. Kim and Y. Kim, "HBM: Memory solution for bandwidth-hungry processors," in *Hot Chips 26 Symposium*, pp. 1–24, 2014.

[33] R. Okazaki, T. Tabata, S. Sakashita, K. Kitamura, N. Takagi, H. Sakata, T. Ishibashi, T. Nakamura, and Y. Ajima, "Supercomputer fugaku cpu a64fx realizing high performance, high-density packaging, and low power consumption," *Fujitsu Technical Review*, pp. 2020–03, 2020.

[34] ReportLinker, *The global HMC and HBM market is expected to reach an estimated $5.1 billion by 2026 and with a CAGR of 30% from 2020 to 2026.* https://www.globenewswire.com/news-release/2021/03/04/2187132/0/en/The-global-HMC-and-HBM-market-is-expected-to-reach-an-estimated\-5-1-billion-by-2026-and-with-a-CAGR-of-30-from-2020-to-2026.html.

[35] *Micron Announces Shift in High-Performance Memory Roadmap Strategy.* https://www.micron.com/about/blog/2018/august/micron-announces-shift-in-high-performance-memory-roadmap-strategy.

[36] *HBM Flourishes, But HMC Lives.* https://www.eetimes.com/hbm-flourishes-but-hmc-lives/.

[37] *DiRAM4™ 3D Memory.* https://tezzaron.com/applications/diram4-3d-memory/.

[38] *Introducing our Monolithic 3D DRAM technology.* http://www.monolithic3d.com/blog/introducing-our-monolithic-3d-dram-technology.

[39] R. Swaminathan, M. J. Schulte, B. Wilkerson, G. H. Loh, A. Smith, and N. James, "Amd instincttm mi250x accelerator enabled by elevated fanout bridge advanced packaging architecture," in *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, pp. 1–2, 2023.

[40] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families: Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 57–70, IEEE, 2021.

[41] N. Beck, S. White, M. Paraschou, and S. Naffziger, "'zeppelin': An soc for multichip architectures," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 40–42, IEEE, 2018.

[42] Y. P. Chiang, S. P. Tai, W. Wu, J. Yeh, C. T. Wang, and D. C. H. Yu, "Info_os (integrated fan-out on substrate) technology for advanced chiplet integration," in *2021 IEEE 71st Electronic Components and Technology Conference (ECTC)*, pp. 130–135, 2021.

[43] M. K. Qureshi and G. H. Loh, "Fundamental latency trade-off in architecting dram caches: Outperforming impractical sram-tags with a simple and practical design," in *45th International Symposium on Microarchitecture*, pp. 235–246, 2012.

[44] G. H. Loh and M. D. Hill, "Efficiently enabling conventional block sizes for very large die-stacked DRAM caches," in *44th International Symposium on Microarchitecture*, pp. 454–464, 2011.

[45] D. Jevdjic, G. H. Loh, C. Kaynak, and B. Falsafi, "Unison cache: A scalable and effective die-stacked dram cache," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 25–37, IEEE Computer Society, 2014.

[46] *CloudSuite: The Benchmark Suite of Cloud Services*. `http://cloudsuite.ch/`.

[47] X. Zhan, Y. Bao, C. Bienia, and K. Li, "Parsec3. 0: A multicore benchmark suite with network stacks and splash-2x," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 5, pp. 1–16, 2017.

[48] M. Fariborz, M. Samani, P. Fotouhi, R. Proietti, I.-M. Yi, V. Akella, J. Lowe-Power, S. Palermo, and S. B. Yoo, "Llm: Realizing low-latency memory by exploiting embedded silicon photonics for irregular workloads," in *International Conference on High Performance Computing*, pp. 44–64, Springer, 2022.

[49] P. Fotouhi, S. Werner, J. Lowe-Power, and S. B. Yoo, "Enabling scalable chiplet-based uniform memory architectures with silicon photonics," in *Proceedings of the International Symposium on Memory Systems*, pp. 222–334, 2019.

[50] X. Wu, W. Zhao, M. Nakamoto, C. Nimmagadda, D. Lisk, S. Gu, R. Radojcic, M. Nowak, and Y. Xie, "Electrical characterization for intertier connections and timing analysis for 3-d ics," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 20, no. 1, pp. 186–191, 2010.

[51] A. Hahn Pereira and V. Betz, "Cad and routing architecture for interposer-based multi-fpga systems," in *Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays*, pp. 75–84, 2014.

[52] S. Werner, J. Navaridas, and M. Luján, "Designing low-power, low-latency networks-on-chip by optimally combining electrical and optical links," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 265–276, IEEE, 2017.

[53] N. Pantano, C. R. Neve, G. Van der Plas, M. Detalle, M. Verhelst, M. Heyns, and E. Beyne, "Technology optimization for high bandwidth density applications on 3d interposer," in *2016 6th Electronic System-Integration Technology Conference (ESTC)*, pp. 1–6, IEEE, 2016.

[54] D. Lee, S. Ghose, G. Pekhimenko, S. Khan, and O. Mutlu, "Simultaneous multi-layer access: Improving 3D-stacked memory bandwidth at low cost," *ACM Transactions on Architecture and Code Optimization*, vol. 12, no. 4, pp. 63:1–63:29, 2016.

[55] K. H. Kyung, C. W. Kim, J. Y. Lee, J. H. Kook, S. M. Seo, J. H. Kim, J. Sunwoo, H. C. Lee, C. S. Kim, B. H. Jeong, *et al.*, "A 800mb/s/pin 2Gb DDR2 SDRAM using an 80nm triple metal technology," in *International Digest of Technical Papers. Solid-State Circuits Conference*, pp. 468–610, 2005.

[56] *Hybrid Memory Cube Specification 2.1.* `http://hybridmemorycube.org/`.

[57] G. H. Loh, "3D-stacked memory architectures for multi-core processors," in *35th International Symposium on Computer Architecture*, pp. 453–464, 2008.

[58] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. P. Shen, and C. Webb, "Die stacking (3D) microarchitecture," in *39th International Symposium on Microarchitecture*, pp. 469–479, 2006.

[59] *JEDEC Publishes HBM3 Update to High Bandwidth Memory (HBM) Standard.* `https://www.jedec.org/news/pressreleases/jedec-publishes-hbm3-update-high-bandwidth-memory-hbm-standard`.

[60] A. Shilov, "Jedec publishes hbm2 specification as samsung begins mass production of chips," 2016.

[61] *International Technology Roadmap for Semiconductors 2.0.* `http://www.itrs2.net/`.

[62] P. Ehrett, V. Goyal, O. Matthews, R. Das, T. Austin, and V. Bertacco, "Analysis of microbump overheads for 2.5 d disintegrated design," *UMich. Ann Arbor Tech. Rep. CSE-TR-002-17*, 2017.

[63] *The Race To Next-Gen 2.5D/3D Packages.* `https://semiengineering.com/the-race-to-next-gen-2-5d-3d-packages/`.

[64] G. H. Loh and M. D. Hill, "Efficiently enabling conventional block sizes for very large die-stacked dram caches," in *44th International Symposium on Microarchitecture (MICRO)*, pp. 454–464, Dec 2011.

[65] K. Kim, J. M. Yook, J. Kim, H. Kim, J. Lee, K. Park, and J. Kim, "Interposer power distribution network (pdn) modeling using a segmentation method for 3-d ics with tsvs," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 3, no. 11, pp. 1891–1906, 2013.

[66] K. Cho, Y. Kim, S. Kim, H. Park, J. Park, S. Lee, D. Shim, K. Lee, S. Oh, and J. Kim, "Fast and accurate power distribution network modeling of a silicon interposer for 2.5-d/3-d ics with multiarray tsvs," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 9, no. 9, pp. 1835–1846, 2019.

[67] K. Cho, Y. Kim, H. Lee, H. Kim, S. Choi, S. Kim, and J. Kim, "Design and analysis of power distribution network (pdn) for high bandwidth memory (hbm) interposer in 2.5 d terabyte/s bandwidth graphics module," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pp. 407–412, IEEE, 2016.

[68] J. Kim, W. Lee, Y. Shim, J. Shim, K. Kim, J. S. Pak, and J. Kim, "Chip-package hierarchical power distribution network modeling and analysis based on a segmentation method," *IEEE Transactions on advanced packaging*, vol. 33, no. 3, pp. 647–659, 2010.

[69] Z. Xu and J. J.-Q. Lu, "Hybrid modeling and analysis of different through-silicon-via (tsv)-based 3d power distribution networks," in *2012 International Electron Devices Meeting*, pp. 30–6, IEEE, 2012.

[70] J. Minz and S. K. Lim, "Block-level 3-d global routing with an application to 3-d packaging," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2248–2257, 2006.

[71] E. J.-W. Fang, T. C.-J. Shih, and D. S.-Y. Huang, "Ir to routing challenge and solution for interposer-based design," in *The 20th Asia and South Pacific Design Automation Conference*, pp. 226–230, IEEE, 2015.

[72] W.-H. Liu, T.-K. Chien, and T.-C. Wang, "Metal layer planning for silicon interposers with consideration of routability and manufacturing cost," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2014.

[73] S. Osmolovskyi and J. Lienig, "Physical design challenges and solutions for interposer-based 3d systems," in *Reliability by Design; 9. ITG/GMM/GI-Symposium*, pp. 1–8, VDE, 2017.

[74] S. Volos, J. Picorel, B. Falsafi, and B. Grot, "Bump: Bulk memory access prediction and streaming," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pp. 545–557, IEEE, 2014.

[75] 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*, vol. 26, no. 4, pp. 18–31, 2006.

[76] R. E. Wunderlich, T. F. Wenisch, B. Falsafi, and J. C. Hoe, "Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling," in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pp. 84–95, IEEE, 2003.

[77] K. Biswas, *Darwin Streaming Server 6.0.3 - Performance and Load tests*. `https://www.codeproject.com/Articles/41874/Darwin-Streaming-Server-setup-customization`.

[78] Y. Yang, Y. Wang, T. Yi, C. Chen, and Q. Liu, "A 6.4-gbps 0.41-pj/b fully-digital die-to-die interconnect phy for silicon interposer based 2.5 d integration," *Integration*, p. 102170, 2024.

[79] *NVIDIA H100 Tensor Core GPU*. `https://www.nvidia.com/en-us/data-center/h100/`.

[80] *NVIDIA TESLA V100*. `https://www.nvidia.com/en-us/data-center/tesla-v100/`.

[81] *High Bandwidth Memory*. `https://www.amd.com/en/technologies/hbm`.

[82] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, "A domain-specific supercomputer for training deep neural networks," *Communications of the ACM*, vol. 63, no. 7, pp. 67–78, 2020.

[83] N. Madan, L. Zhao, N. Muralimanohar, A. Udipi, R. Balasubramonian, R. Iyer, S. Makineni, and D. Newell, "Optimizing communication and capacity in a 3D stacked reconfigurable cache hierarchy," in *15th Interna-*

tional Conference on High-Performance Computer Architecture, pp. 262–274, 2009.

[84] S. Franey and M. Lipasti, "Tag tables," in *21st International Symposium on High Performance Computer Architecture*, pp. 514–525, 2015.

[85] N. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," in *[1990] Proceedings. The 17th Annual International Symposium on Computer Architecture*, pp. 364–373, 1990.

[86] J. Lira, C. Molina, and A. González, "Last bank: dealing with address reuse in non-uniform cache architecture for cmps," in *Euro-Par 2009 Parallel Processing: 15th International Euro-Par Conference, Delft, The Netherlands, August 25-28, 2009. Proceedings 15*, pp. 297–308, Springer, 2009.

[87] H. Yang, A. Breslow, J. Mars, and L. Tang, "Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers," in *40th International Symposium on Computer Architecture*, pp. 607–618, 2013.

[88] H. Kasture and D. Sanchez, "Ubik: efficient cache sharing with strict qos for latency-critical workloads," in *19th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 729–742, 2014.

[89] D. Sanchez and C. Kozyrakis, "Vantage: scalable and efficient fine-grain cache partitioning," in *38th International Symposium on Computer Architecture*, pp. 57–68, 2011.

[90] 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 *16th International Conference on High-Performance Computer Architecture*, pp. 1–12, 2010.

[91] T. Kgil, S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Reinhardt, and K. Flautner, "Picoserver: using 3D stacking technology to enable a compact energy efficient chip multiprocessor," in *12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 117–128, 2006.

[92] *Intel Xeon 6780E.* `https://www.techpowerup.com/cpu-specs/xeon-6780e.c3664`.

[93] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 211–222, 2002.

[94] A. Gupta, W.-D. Weber, and T. Mowry, "Reducing memory and traffic requirements for scalable directory-based cache coherence schemes," in *Scalable shared memory multiprocessors*, pp. 167–192, Springer, 1992.

[95] L. Gwennap, "ThunderX rattles server market," *Microprocessor Report*, vol. 29, no. 6, pp. 1–4, 2014.

[96] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *44th International Symposium on Computer Architecture*, pp. 1–12, 2017.

[97] *Intel Xeon Processor E7-8890 v3.* https://www.intel.co.uk/content/www/uk/en/-products/processors/xeon/e7-processors/e7-8890-v3.html.

[98] *IBM Power8.* https://www.ibm.com/power/hardware.

[99] P. Singh, R. Sankar, X. Hu, W. Xie, A. Sarkar, and T. Thomas, "Power delivery network design and optimization for 3d stacked die designs," in *2010 IEEE International 3D Systems Integration Conference (3DIC)*, pp. 1–6, IEEE, 2010.

[100] S. Wang, F. Firouzi, F. Oboril, and M. B. Tahoori, "P/g tsv planning for ir-drop reduction in 3d-ics," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2014.

[101] B. Bose and I. Thakkar, "Characterization and mitigation of electromigration effects in tsv-based power delivery network enabled 3d-stacked drams," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pp. 101–107, 2021.

[102] Y. Wang, G. Dong, W. Xiong, D. Song, Z. Zhu, and Y. Yang, "Impedance modeling and analysis of multi-stacked on-chip power distribution network in 3d ics," *Journal of Computational Electronics*, vol. 21, no. 6, pp. 1282–1292, 2022.

[103] J. Cong and G. Luo, "A multilevel analytical placement for 3d ics," in *2009 Asia and South Pacific Design Automation Conference*, pp. 361–366, IEEE, 2009.

[104] M.-K. Hsu, V. Balabanov, and Y.-W. Chang, "Tsv-aware analytical placement for 3-d ic designs based on a novel weighted-average wirelength model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 4, pp. 497–509, 2013.

[105] J.-M. Lin, H.-Y. Hsieh, H. Kung, and H.-J. Lin, "Routability-driven analytical placement with precise penalty models for large-scale 3d ics," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, 2022.

[106] D. Saha and S. Sur-Kolay, "Multi-objective optimization of placement and assignment of tsvs in 3d ics," in *2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID)*, pp. 372–377, IEEE, 2017.

[107] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0," in *40th International Symposium on Microarchitecture*, pp. 3–14, 2007.

[108] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "CACTI-3DD: Architecture-level modeling for 3D die-stacked dram main memory," in *Design, Automation & Test in Europe Conference & Exhibition*, pp. 33–38, 2012.

[109] *Micron. "1.35V DDR3L power calculator (4Gb x16 chips)*, 2013.

[110] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[111] P. Hammarlund, A. J. Martinez, A. A. Bajwa, D. L. Hill, E. Hallnor, H. Jiang, M. Dixon, M. Derr, M. Hunsaker, R. Kumar, R. B. Osborne,

R. Rajwar, R. Singhal, R. D'Sa, R. Chappell, S. Kaushik, S. Chennupaty, S. Jourdan, S. Gunther, T. Piazza, and T. Burton, "Haswell: The fourth-generation intel core processor," *IEEE Micro*, vol. 34, pp. 6–20, Mar 2014.

[112] M. Awasthi, K. Sudan, R. Balasubramonian, and J. Carter, "Dynamic hardware-assisted software-controlled page placement to manage capacity allocation and sharing within large caches," in *15th International Conference on High-Performance Computer Architecture*, pp. 250–261, 2009.

[113] B. M. Beckmann, M. R. Marty, and D. A. Wood, "ASR: adaptive selective replication for CMP caches," in *39th International Symposium on Microarchitecture*, pp. 443–454, 2006.

[114] M. Zhang and K. Asanovic, "Victim replication: maximizing capacity while hiding wire delay in tiled chip multiprocessors," in *32nd International Symposium on Computer Architecture*, pp. 336–345, 2005.

[115] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, "A NUCA substrate for flexible CMP cache sharing," *IEEE Transactions on Parallel Distributed Systems*, vol. 18, no. 8, pp. 1028–1040, 2007.

[116] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Reactive NUCA: near-optimal block placement and replication in distributed caches," in *36th International Symposium on Computer Architecture*, pp. 184–195, 2009.

[117] D. W. Chang, G. Byun, H. Kim, M. Ahn, S. Ryu, N. S. Kim, and M. Schulte, "Reevaluating the latency claims of 3D stacked memories," in *18th Asia and South Pacific Design Automation Conference*, pp. 657–662, 2013.

[118] P.-A. Tsai, N. Beckmann, and D. Sanchez, "Jenga: Software-defined cache hierarchies," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 652–665, ACM, 2017.

[119] *Micron. RLDRAM 2 and 3 Specifications.* https://www.micron.com/products/dram/rldram-memory.

[120] Y. Sato, T. Suzuki, T. Aikawa, S. Fujioka, W. Fujieda, H. Kobayashi, H. Ikeda, T. Nagasawa, A. Funyu, Y. Fuji, K. Kawasaki, M. Yamazaki,

and M. Taguchi, "Fast cycle RAM (FCRAM); a 20-ns random row access, pipe-lined operating DRAM," in *Symposium on VLSI Circuits. Digest of Technical Papers*, pp. 22–25, 1998.

[121] W. Leung, F.-C. Hsu, and M. E. Jones, "The ideal soc memory: 1T-SRAM™," in *13th International ASIC/SOC Conference*, pp. 32–36, 2000.

[122] *Tezzaron DiRAM4™ 3D Memory*. `https://tezzaron.com/applications/diram4-3d-memory/`.

[123] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu, "Tiered-latency DRAM: A low latency and low cost DRAM architecture," in *19th International Symposium on High Performance Computer Architecture*, pp. 615–626, 2013.

[124] Y. H. Son, O. Seongil, Y. Ro, J. W. Lee, and J. H. Ahn, "Reducing memory access latency with asymmetric DRAM bank organizations," in *40th International Symposium on Computer Architecture*, pp. 380–391, 2013.

[125] M. O'Connor, N. Chatterjee, D. Lee, J. Wilson, A. Agrawal, S. W. Keckler, and W. J. Dally, "Fine-grained DRAM: energy-efficient DRAM for extreme bandwidth systems," in *50th International Symposium on Microarchitecture*, pp. 41–54, 2017.

[126] T. Zhang, K. Chen, C. Xu, G. Sun, T. Wang, and Y. Xie, "Half-DRAM: A high-bandwidth and low-power DRAM architecture from the rethinking of fine-grained activation," in *41st International Symposium on Computer Architecture*, pp. 349–360, 2014.

[127] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A case for exploiting subarray-level parallelism (SALP) in DRAM," in *39th International Symposium on Computer Architecture*, pp. 368–379, 2012.

[128] K. Sudan, N. Chatterjee, D. W. Nellans, M. Awasthi, R. Balasubramonian, and A. Davis, "Micro-pages: increasing DRAM efficiency with locality-aware data placement," in *15th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 219–230, 2010.

[129] R. Ausavarungnirun, K. K.-W. Chang, L. Subramanian, G. H. Loh, and O. Mutlu, "Staged memory scheduling: Achieving high performance and scalability in heterogeneous systems," in *39th International Symposium on Computer Architecture*, pp. 416–427, 2012.

[130] O. Mutlu and T. Moscibroda, "Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared DRAM systems," in *35th International Symposium on Computer Architecture*, pp. 63–74, 2008.

[131] H. Hassan, G. Pekhimenko, N. Vijaykumar, V. Seshadri, D. Lee, O. Ergin, and O. Mutlu, "Chargecache: Reducing DRAM latency by exploiting row access locality," in *22nd International Symposium on High Performance Computer Architecture*, pp. 581–593, 2016.

[132] J. Sim, G. H. Loh, V. Sridharan, and M. O'Connor, "Resilient die-stacked dram caches," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 416–427, 2013.

[133] G. Mappouras, A. Vahid, R. Calderbank, D. R. Hower, and D. J. Sorin, "Jenga: Efficient fault tolerance for stacked dram," in *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 361–368, IEEE, 2017.

[134] P. J. Nair, V. Sridharan, and M. K. Qureshi, "Xed: Exposing on-die error detection information for strong memory reliability," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 341–353, 2016.

[135] P. J. Nair, D. A. Roberts, and M. K. Qureshi, "Citadel: Efficiently protecting stacked memory from tsv and large granularity failures," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 12, no. 4, pp. 1–24, 2016.

[136] F. Garcia-Herrero, A. SÁnchez-MaciÁn, and J. A. Maestro, "Combined symbol error correction and spare through-silicon vias for 3d memories," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 2139–2145, 2020.

[137] J. Goldstein, "Chiplets and sustainability," 2022.

[138] F. von Trapp, "Why chiplets aren't your grandparents mcms and other takeaways from imaps advanced sip 2022," 2022.

[139] A. Margaritov, S. Gupta, R. Gonzalez-Alberquilla, and B. Grot, "Stretch: Balancing qos and throughput for colocated server workloads on smt cores," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 15–27, IEEE, 2019.

[140] *AMD EPYC 9754*. `https://www.techpowerup.com/cpu-specs/epyc-9754.c3257`.