At the Locus of Performance: Quantifying the Effects of Copious 3D-Stacked Cache on HPC Workloads

At the Locus of Performance: Quantifying the Effects of Copious 3D-Stacked Cache on HPC Workloads

JENS DOMKE and EMIL VATAI, RIKEN Center for Computational Science, Japan
BALAZS GEROFI, Intel Corporation, USA
YUETSU KODAMA and MOHAMED WAHIB, RIKEN Center for Computational Science, Japan
ARTUR PODOBAS, KTH Royal Institute of Technology, Sweden
SPARSH MITTAL, Indian Institute of Technology, Roorkee, India
MIQUEL PERICÀS, Chalmers University of Technology, Sweden
LINGQI ZHANG, Tokyo Institute of Technology, Japan
PENG CHEN, National Institute of Advanced Industrial Science and Technology, Japan
ALEKSANDR DROZD and SATOSHI MATSUOKA, RIKEN Center for Computational Science, Japan

Over the last three decades, innovations in the memory subsystem were primarily targeted at overcoming the data movement bottleneck. In this paper, we focus on a specific market trend in memory technology: 3D-stacked memory and caches. We investigate the impact of extending the on-chip memory capabilities in future HPC-focused processors, particularly by 3D-stacked SRAM. First, we propose a method oblivious to the memory subsystem to gauge the upper-bound in performance improvements when data movement costs are eliminated. Then, using the gem5 simulator, we model two variants of a hypothetical LARge Cache processor (LARC), fabricated in 1.5 nm and enriched with high-capacity 3D-stacked cache. With a volume of experiments involving a broad set of proxy-applications and benchmarks, we aim to reveal how HPC CPU performance will evolve, and conclude an average boost of 9.56× for cache-sensitive HPC applications, on a per-chip basis. Additionally, we exhaustively document our methodological exploration to motivate HPC centers to drive their own technological agenda through enhanced co-design.

J. Domke and E. Vatai are co-first authors.
"New Paper, Not an Extension of a Conference Paper." as requested by the TACO journal.
This work was supported by PRESTO Grant Number JPMJPR20MA, Japan; the New Energy and Industrial Technology Development Organization (NEDO); and the AIST/TokyoTech Real-world Big-Data Computation Open Innovation Laboratory (RWBC-OIL).
Authors' addresses: J. Domke, E. Vatai, Y. Kodama, M. Wahib, A. Drozd, and S. Matsuoka, RIKEN Center for Computational Science, 7-1-26 Minatojima-minamimachi, Chuo-ku, Kobe, Hyogo, Japan, 650-0047; e-mails: jens.domke@riken.jp, emil.vatai@riken.jp, yuetsu.kodama@riken.jp, mohamed.attia@riken.jp, aleksandr.drozd@riken.jp, matsu@acm.org; B. Gerofi, Intel Corporation, 2111 NE 25th Ave, Hillsboro, Oregon, United States, 97124; e-mail: balazs.gerofi@intel.com; A. Podobas, KTH Royal Institute of Technology, Brinellvägen 8, Stockholm, Stockholm, Sweden, 114 28; e-mail: podobas@kth.se; S. Mittal, Indian Institute of Technology, Roorkee - Haridwar Highway, Roorkee, Uttarakhand, India, 247667; e-mail: sparsh.mittal@ece.iitr.ac.in; M. Pericàs, Chalmers University of Technology, Chalmersplatsen 4, Göteborg, Västra Götaland, Sweden, 412 96; e-mail: miquelp@chalmers.se; L. Zhang, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo, Tokyo, Japan, 152-8550; e-mail: zhang.l.ai@m.titech.ac.jp; P. Chen, National Institute of Advanced Industrial Science and Technology, 1-8-31 Midorigaoka, Ikeda-ku, Osaka, Osaka, Japan, 563-0026; e-mail: chin.hou@aist.go.jp.

1544-3566/2023/12-ART57
https://doi.org/10.1145/3629520

CCS Concepts: • **Computing methodologies** → **Discrete-event simulation**; • **Hardware** → *Emerging architectures*; *Memory and dense storage;*

Additional Key Words and Phrases: Emerging architecture study, 3D-stacked memory, gem5 simulation, proxy-applications

ACM Reference format:
Jens Domke, Emil Vatai, Balazs Gerofi, Yuetsu Kodama, Mohamed Wahib, Artur Podobas, Sparsh Mittal, Miquel Pericàs, Lingqi Zhang, Peng Chen, Aleksandr Drozd, and Satoshi Matsuoka. 2023. At the Locus of Performance: Quantifying the Effects of Copious 3D-Stacked Cache on HPC Workloads. *ACM Trans. Arch. Code Optim.* 20, 4, Article 57 (December 2023), 26 pages.
https://doi.org/10.1145/3629520

1 INTRODUCTION

Historically, the reliable performance increase of von Neumann-based general-purpose processors (CPUs) was driven by two technological trends. The first, observed by Gordon E. Moore [76], is that the number of transistors in an integrated circuit doubles roughly every two years. The second, called Dennard's scaling [30], postulates that as transistors get smaller their power density stays constant. These trends synergized well, allowing computer architectures to continuously improve performance through, for example, aggressive pipelining and superscalar techniques without running into thermal limitations by, e.g., reducing the operating voltage. In the early 2000s, Dennard's scaling ended [51] and forced architects to shift their attention from improving instruction-level parallelism to exploiting on-chip multiple-instruction multiple-data parallelism [43]. This immediate remedy to the end of Dennard's scaling applies to this day in the form of processors such as Fujitsu A64FX [96], AMD Ryzen [105], or NVIDIA GPUs [79, 86].

Unfortunately, Moore's law is impending termination [107], and we are entering a post-Moore era [112], home to a diversity of architectures, such as quantum-, neuromorphic-, or reconfigurable computing [49]. Many of these prototypes hold promise but are still immature, focus on a niche use case, or incur long development cycles. However, there is one salient solution that is growing in maturity and which can facilitate performance improvements in the decades to come even for the classic von Neumann CPUs we have come to rely upon—3D **integrated circuit (IC)** stacking [14]. 3D ICs refer to the general technologies of vertically building integrated circuits and can be done in multiple ways, such as by stacking multiple discrete dies and connecting them using coarse **through-silicon vias (TSVs)** or growing the 3D integrated circuit monolithically on the wafer [100].

Recent advances in 3D integrated circuits have enabled many times higher capacity for on-chip memory (caches) than traditional systems (e.g., AMD V-Cache [40]). Intuition tells us that an increased cache size, resulting from 3D-stacking, will help alleviate the performance bottlenecks of key scientific applications. To demonstrate this, we conduct a pilot study where we execute one of the important proxy-apps from the DoE **ExaScale Computing Project (ECP)** suite, MiniFE [50] (cf. Section 3.3), on AMD EPYC Milan and Milan-X CPUs—two architecturally similar processors with vastly different L3 cache sizes [17]. Figure 1 overviews our result of the pilot study, and we see that for a subset of problem sizes, in particular the $160 \times 160 \times 160$ input, the 3-times larger L3 capacity of Milan-X yields up-to 3.4× improvements over baseline Milan for this memory-bound application, which motivates us to further research 3D-stacked caches.

3D integrated circuits have various benefits [52], including (i) shorter wire lengths in the interconnect leading to reduced power consumption, (ii) improved memory bandwidth through on-chip integration that can alleviate performance bottlenecks in memory-bound applications, (iii) higher package density yielding more compute and smaller system footprint, and (iv) possibly lower

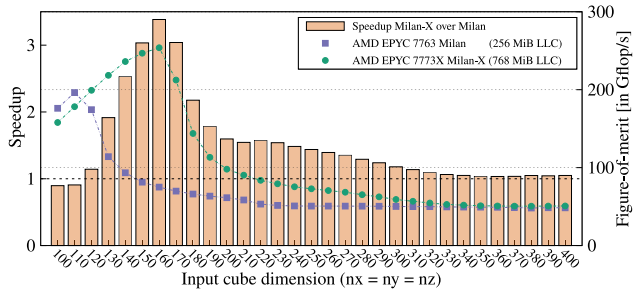


Fig. 1. MiniFE: relative performance improvement of AMD EPYC 7773X Milan-X over 7763 Milan (for details cf. Table 1), and Figure of Merit; Input problem scaled from 100 × 100 × 100 to 400 × 400 × 400; Benchmarks executed with 16 MPI ranks and 8 OpenMP threads.

Table 1. Systems Configuration for the Benchmarked AMD EPYC 7763 Milan and 7773X Milan-X (for More Details: See Zen 3 Microarch)

	7763 Milan	7773X Milan-X
Sockets	2	2
	CPU config. per Socket:	
Cores	64	64
CCDs	8	8
Freq.	2.45 GHz	2.20 GHz
TDP	280 W	280 W
L3	256 MiB	768 MiB
	Cache per core:	
L2	512 KiB	512 KiB
L1 I+D	32+32 KiB	32+32 KiB
Memory	1 TiB DDR4, 16 cha., 409.6 GB/s	

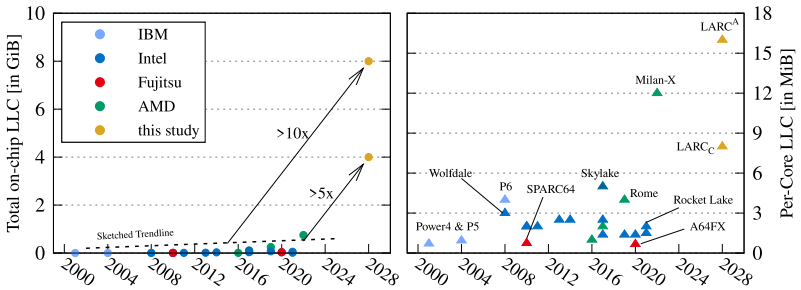


Fig. 2. A sample of representative server-grade CPUs of each generational micro-architecture in comparison to our study of LARC; Left: total on-chip last-level cache (in GiB); Right: per-core last-level cache (in MiB) for the same CPUs; The two LARC variants will be discussed in detail in Section 5.1.

fabrication cost due to smaller die size (thus improved yield). All these are very desirable benefits in today's exascale (and future) **High-Performance Computing (HPC)** systems. But how far can 3D ICs (with a focus on increased on-chip cache) take us in HPC?

Contributions: We study our research questions from three different levels of abstraction: (i) we design a **novel exploration framework** that allows us to simulate HPC applications running on a hypothetical processor having infinitely large L1D cache. We use this framework, that is **orders of magnitude faster** than cycle-accurate simulators, to estimate an upper-bound for cache-based improvements; (ii) we **model a hypothetical LARge Cache processor (*LARC*)**, that builds on the design of A64FX, with an **LLC (Last Level Caches)** designed with eight stacked SRAM dies under 1.5 nm manufacturing assumption; (iii) we complement our study with a plethora of **simulations of HPC proxy-applications** and CPU micro-benchmarks; and lastly (iv) we find that over half (31 out of 52) of the simulated applications experience a ≥ 2× speedup on LARC's **Core Memory Group (CMG)** that occupies only one fourth the area of the baseline A64FX CMG. For applications that are responsive to larger cache capacity, this would translate to an average improvement of 9.56× (geometric mean) when we assume ideal scaling and compare at the full chip level.

The novelty in this paper lies in the purpose which LARC serves, and not the design of LARC itself. As Figure 2 shows, the capacity (and bandwidth; not shown) of the LLC have increased at a moderately gradual slope over the last two decades—with Milan-X being a noticeable outlier

in per-core LLC. However, we are querying the effect of an LLC, that is an order of magnitude above the trend line as depicted in Figure 2, on HPC applications. On top of our provided baseline, further application-specific restructuring to utilize large caches [69] will result in even greater benefit.

2 CPUS EMPOWERED WITH HIGH-CAPACITY CACHE: THE FUTURE OF HPC?

The memory bandwidth of modern systems has been the bottleneck (the "memory wall" [71]) ever since CPU performance started to outgrow the bandwidth of memory subsystems in the early 1990s [70]. Today, this trend continues to shape the performance optimization landscape in high-performance computing [83, 85]. Diverse memory technologies are emerging to overcome said data movement bottleneck, such as **Processing-in-Memory (PIM)** [12], 3D-stackable **High-Bandwidth Memory (HBM)** [74], deeper (and more complex) memory hierarchies [115], and—the topic of the present paper—novel 3D-stacked caches [14, 68, 98].

In this study, our aspiration is to gauge the far end of processor technology and how it may evolve in six to eight years from now, circa 2028, when processors using 1.5 nm technology are expected to be available according to the IEEE IRDS Roadmap [53, Figure ES9]. More specifically, as 3D-stacked SRAM memory [120] becomes more common, what are the performance implications for common HPC workloads, and what new challenges lie ahead for the community? However, before attempting to understand what performance may look like six years from now, we must describe how the processor itself might change. In this section, we introduce, motivate, and reason about our design choices of what we envision as a hypothetical CPU that capitalizes on large capacity 3D-stacked cache, briefly called **LARC (LARge Cache processor)**. Before looking at LARC, we must first set and analyze a baseline processor.

2.1 LARC's Baseline: The A64FX Processor

We choose to base our future CPU design on the A64FX [118]. Fujitsu's Arm-based A64FX is powering Supercomputer Fugaku [96], leader of the HPCG (TOP500 [104]; cf. Section 3.3) and Graph500 performance charts. A64FX is manufactured in 7 nm technology and has a total of 52 Arm cores (with Scalable Vector Extensions [103]) distributed across four compute clusters, called **Core Memory Groups (CMGs)**. Twelve cores are available to the user, and one core is exclusively used for management. Each core has a local 64 KiB instruction and data-cache, and is capable of delivering 70.4 Gflop/s (IEEE-754 double-precision) performance—accumulated: 845 Gflop/s per CMG (user cores) or 3.4 Tflop/s for the entire chip. Each CMG contains a 8 MiB L2 cache slice, delivering over 900 GB/s bandwidth to the CMG [118]. The combined L2 cache, which is the CPU's 32 MiB **last level cache (LLC)**, is kept coherent through a ring interconnect that connects the four CMGs. Inside the CMG, a crossbar switch is used to connect the cores and the L2 slice. The L2 cache has 16-way set associativity, a line-size of 256 bytes, and the bus-width between the L1 and L2 cache is set to be 128 bytes (read) and 64 bytes (write).

We emphasize that our aim is not to propose a successor of A64FX, nor are we particularly restricting our vision by the design constrains of A64FX (e.g., power budget). However, we build our design on A64FX because: (i) as mentioned above, A64FX represents the high-end in performance for commercially available CPUs, so it is a logical starting point. (ii) A64FX is the only commercially-available CPU, currently in continued production, with HBM. The expected bandwidth ratio between future HBM and future 3D-stacked caches is similar to the ratio between traditional DRAM and LLC bandwidths [80], which is what applications and performance models are accustomed to. (iii) The A64FX LLC cache design (particularly the L2 slices connected by a crossbar switch) happens to be convenient and thus, requires a minimal effort to extend the design in a simulated environment.

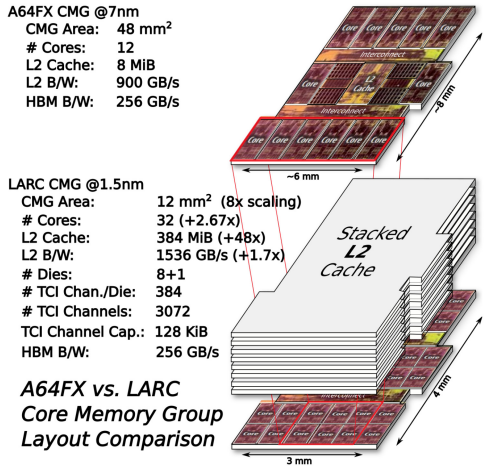


Fig. 3. Difference between A64FX's Core Memory Group (CMG) and a LARC CMG in various performance-governing parameters; Most notable (for our study) is the 48× increase in per-CMG L2 cache capacity; **Note:** despite appearing similar in the figure, the LARC CMG is, in fact, four times smaller.

In conclusion, while we extend the A64FX architecture, our workflow itself can be generalized to cover any of the processors supported by CPU simulators (e.g., variants of gem5 [13] can simulate other architectures, including x86).

2.2 Floorplan Analysis for Fujitsu A64FX

In order to estimate the floorplan of the future LARC processor built on 1.5 nm technology, we first need the floorplan of the current A64FX processor built at 7 nm. We do know that the die size of A64FX is $\approx 400\,mm^2$ [96]. With the openly-available die shots including processor core segments highlighted [82], we can estimate most of the A64FX floorplan, including the size of CMGs and processor cores, as shown in Figure 3. Overall, each CMG is $\approx 48\,mm^2$ in area, where an A64FX core occupies $\approx 2.25\,mm^2$ area. The remaining parts of the CMG consist of the L2 cache slice and controller as well as the interconnect for intra-CMG communication.

2.3 From A64FX's to LARC's CMG Layout

Knowing the floorplan, we proceed to describe how we envision the CMG design with 1.5 nm technology. We scale the CMG by moving four generations, from 7 nm to 1.5 nm, and reduce the silicon footprint by around 8× (≈ 1.7× per generation) for the entire CMG [39]. The new CMG consumes as little as $6\,mm^2$ of silicon area. Next, we reclaim the area currently occupied by the L2 cache and controller and replace it with three additional CPU cores, yielding a total of 16. Further, inline with the projected year 2019→2028 growth in the number of cores [54, Table SA-1], we double the core count of the CMG to 32, which leads to it occupying $\approx 12\,mm^2$ of silicon area. We pessimistically leave the interconnect area unchanged and continue to use it as the primary means for communication. We call this new variant as LARC's CMG. Finally, we assume the same die size, and hence, LARC would have 16 CMGs, each with 32 cores, in comparison to A64FX's 4 CMG with 12+1 cores each. For LARC, we ignore the management core. However, our performance analysis will remain on the CMG level, instead of full chip, due to limitations we detail in Section 3.2.

2.4 LARC's Vertically Stacked Cache

In the above design, we removed the L2 cache and controller from the CMG of LARC. We now assume that the L2 cache can be directly placed vertically on the CMG through 3D stacking [68]. We build our estimations based on experiments from Shiba et al. [98], who demonstrated the feasibility of stacking up-to-eight SRAM dies on top of a processor using a **ThruChip Interface (TCI)**. The capacity and bandwidth of stacked memory is a function of several parameters: the number of channels available (N_{ch}), the per-channel capacity (N_{cap} in KiB), their width (W in bytes), the number of stacked dies (N_{dies}), and the operating frequency (f_{clk} in GHz). Shiba et al. [98] estimated that at a 10 nm process technology, eight stacks would provide ≈ 512 MiB of aggregated SRAM capacity for a footprint of ≈ 121 mm^2. In their design, each stack has 128 channels of 512 KiB capacity. In our work, we conservatively assume an 8× scaling from 10 nm to 1.5 nm, and thus, at 12 mm^2 area (the size of one LARC CMG), N_{ch} on each die would be ≈ 102 (=128*8/10).

We approximate N_{ch} to a nearby sum of power-of-two number, viz., $N_{ch} = 96$. Thus, with eight stacked dies ($N_{dies} = 8$), our 3D SRAM cache has a total storage capacity of $N_{dies} \cdot N_{ch} \cdot N_{cap} =$ 384 MiB per CMG. We estimate the bandwidth in a similar way. We know from previous studies [98] that 3D-stacked SRAM, built on 40 nm technology, can operate at 300 MHz. We conservatively expect the same SRAM to operate at (f_{clk}=)1 GHz when moving from 40 nm→1.5 nm. To account for the increased working set size of future applications, we assume a channel width (W) of 16 byte, compared to the 4 byte width assumed in [98]. With this, the CMG bandwidth becomes: $N_{ch} \cdot f_{clk} \cdot W = 1536$ GB/s. The read- and write-latency of their SRAM cache is 3 cycles, including the vertical data movement overhead [98].

While stacked DRAM caches theoretically provide higher capacity than stacked SRAM caches, they have limitations. For example, the latency of stacked DRAM is only 50% lower compared to DDR3 DRAM, and hence, they exacerbate miss latency; they requires refresh operations which consumes energy and reduces availability; and due to their large size, the stacked DRAM caches require special techniques for managing metadata and avoiding bandwidth bloat [23, 74]. The tag size of a stacked DRAM may exceed the LLC capacity, and hence, the tags may need to be stored in the DRAM itself which worsens hit latency. Set-associative designs and serial tag-data accesses further increase hit latency. Proposed architectural techniques and mitigation strategies, such as Loh-Hill cache [67], have yet to solve these problems. By contrast, 3D SRAM caches do not suffer from any of these issues. In fact, at iso-capacity, a 3D SRAM cache has even lower access latency than a 2D SRAM cache. Since stacked 3D SRAM caches have lower capacity than stacked DRAM, its metadata (e.g., tag) can be easily stored in SRAM itself, further reducing the access latency.

For our cache design, we assume a 256 B cache block design, which avoids bandwidth bloat. Each tag takes 6 B and as such, the total tag array size for each CMG becomes 9 MiB. This tag array can be easily placed in the cache itself. We assume that tag and data accesses happen sequentially. The tags and data of a cache set are stored on a single die. Hence, on every access, only one die needs to be activated. Since this takes only few cycles, the overall miss penalty remains small and comparable to that of A64FX' LLC.

To show that our cache projections are realistic, we compare it with AMD's 3D V-cache design. It uses a single stacked die for the L3 cache, providing 64 MiB capacity (in addition to the 32 MiB cache in the base die) at 7 nm [26, 40] and only 3 to 4 cycles of extra latency compared to the non-stacked version [21]. It has 36 mm^2 area and has a bandwidth of 2 TB/s. When stacking additional dies on top, and assuming an 8× scaling of the area by going from 7 nm to 1.5 nm, we speculate that the LLC capacity of this commercial processor could easily exceed that of our proposed LARC.

2.5 LARC's Core Memory Group (CMG)

At last, we detail our experimental CMG built on a hypothetical 1.5 nm technology: the **LARC CMG**. An illustration of this system is shown in Figure 3. Each CMG consists of 32 A64FX-like cores, which keeps the L1 instruction- and data-cache to 64 KiB each, yielding a per CMG performance of ≈ 2.3 Tflop/s (IEEE-754 double-precision). A 384 MiB L2 cache is stacked vertically on the top of the CMG through eight SRAM layers.

We keep the HBM memory bandwidth per CMG to its current A64FX value of 256 GB/s to be able to quantify performance improvements from the proposed large capacity 3D cache in isolation from any improvements that would come from increased HBM bandwidth. Furthermore, we make no assumption on the technology scaling of blocks that contain hard-to-scale-down analog components (e.g., TofuD or PCIe IP blocks) and instead focus exclusively on scaling the CMG-part of the System-on-Chip (i.e., processing cores, L1/L2 caches, and intra-chip interconnects).

While our study focuses on evaluating a single CMG, we conclude that a complete, hypothetical LARC CPU, with a die size similar to the current A64FX, would contain 512 processing cores, 6 GiB of stacked L2 cache, a peak L2 bandwidth of 24.6 TB/s, a peak HBM bandwidth of 4.1 TB/s, and a total of 36 Tflop/s of raw, double-precision, compute. The A64FX processor has a peak HBM bandwidth of 1 TB/s, whereas our envisioned LARC CPU has 4× more CMGs and hence, a peak HBM bandwidth of 4.1 TB/s. Thus, compared to A64FX, LARC has higher *effective bandwidth* of external memory. Further changes to the HBM generation are beyond the scope of this study.

2.6 LARC's Power and Thermal Considerations

To estimate the power consumption of LARC, we analyze A64FX's current consumption and extrapolate to 1.5 nm by leveraging public technology roadmaps. A64FX's peak power, achieved while running DGEMM, is 122 W [117]; where 95 W correspond to core power and 15 W correspond to the **memory interface (MIF)**, and hence, we conclude 1.98 W/core and 3.75 W/MIF. Therefore, a LARC CMG with 32 cores in 7 nm would consume 67.1 W. TSMC projects that shrinking from 7 nm to 5 nm yields a power reduction of about 30% [99], i.e., 46.98 W for LARC's CMG in 5 nm. IRDS's roadmap [53, Figure ES9] indicates a further compounded power reduction (at iso frequency) of 42% when moving from 5 nm to 1.5 nm, i.e., 27.37 W for LARC's CMG in 1.5 nm. As the full LARC chip is estimated to include 16 CMGs, we project a total power of 438 Watt (not including the L2 cache).

Next, we estimate the power consumed by the principal part of this study—the 384 MiB L2 cache. A 4 MiB SRAM L2 cache in 7 nm consumes 64 mW of static power [44]. Assuming a similar (pessimistic) static power consumption at 1.5 nm and extrapolated to 384 MiB, we find that our cache would have a static power consumption of 6.14 W. Scaled to the full 16 CMGs of our hypothetical LARC, we arrive at a static power consumption of 98.3 W. This static power consumption of caches represents between 90% and 98% of the entire power consumption (at 350 K temperature, see, e.g., [5, 20]), where the remainder is the dynamic power consumption. If we assume a pessimistic 9:1 ratio between static and dynamic power, then this yields a total power consumption of 109.23 W for 6 GiB of chip-wide stacked L2 cache.

To conclude, a LARC processor (16 CMG) would have to be designed for a **thermal design power (TDP)** of **547 W**. While this expected TDP is more than the current A64FX, it is not entirely unlike emerging architectures, such as NVIDIA's H100 [81] that consumes up to 700 W or the AMD Instinct MI250X GPU [3] at 560 W. We stress that our estimate of 547 W is peak power draw achieved only during parallel DGEMM execution. Adjusting for Stream Triad, based on the breakdown in [117], we conclude a realistic, and considerably lower, power consumption of 420 W for bandwidth-bound applications running on the whole LARC chip.

Finally, while this L2 cache power estimation might appear pessimistic, there are ample opportunities to further reduce power consumption. To save static energy, all the un-accessed dies can be changed to data-retentive, low-power (sleep) state. To deal with remaining thermal issues after stacking the cache layers underneath the cores instead of on top, one can additionally adapt simple direct-die cooling or advanced techniques [18, 106], such as high-κ thermal compound [42], microfluid cooling [114], or thermal-aware floorplanning, task-scheduling and data-placement optimizations. Specifically, microfluid cooling can handle power densities of 3.5 W/mm^2 and hot-spot power levels of over 20 W/mm^2 for 3D-stacked chips [1]. By contrast, our LARC CPU has a power density of 2.85 W/mm^2 at 192 mm^2 if we ignore adjunct components such as I/O die, PCIe, TofuD interface, and the like, and around half the power density at 400 mm^2 if these components are included.

3 PROJECTING PERFORMANCE IMPROVEMENT IN SIMULATED ENVIRONMENTS

Analyzing LARC's feasibility is only the first step, and hence we have to demonstrate the effects of the proposed changes on real workloads to allow a meaningful cost-benefit analysis by CPU vendors. This section details two simulation approaches (one novel; one established) and discusses the HPC applications, which we evaluate extensively in Sections 4 and 5.

3.1 Simulating Unrestricted Locality with MCA

Designing and executing even initial studies (i.e., no complex memory models, etc.) with cycle-level gem5 simulations for realistic workloads takes substantial time with unknown outcome. Therefore, one would want to have a first-order approximation of a very large and fast cache. Regrettably, and to the best of our knowledge, existing approaches for fast first-order approximations do generally not support complex HPC applications, i.e., the existing tools neither handle multi-threading correctly nor do they have support for MPI applications [6]. Hence, we design a simulation approach, using **Machine Code Analyzers (MCA)**, which can estimate the speedup for a given application orders-of-magnitude faster than gem5 (typically hours instead of months; cf. next section). This upper bound in expected performance improvement allows us to: (i) get a perspective on the best possible performance improvement if all read/writes can be satisfied from the cache; and (ii) justify more accurate simulations and classify their results with respect to the baseline and the upper bound.

Machine Code Analyzers, such as llvm-mca [66], have been designed to study microarchitectures, improve compilers, and investigate resource pressure for application kernels. Usually, the input for these tools is a short Assembly sequence and they output, among other things, an expected throughput for a given CPU when the sequence is executed many times and all data is available in L1 data cache. For most real applications, the latter assumption is obviously incorrect, however, it is ideal to gauge an upper bound on performance when all the memory-bottlenecks disappear.

Unfortunately, it is neither feasible to record all executed instructions in one long sequence, nor to analyze a full program sequence with llvm-mca. Hence, we break the program execution into basic blocks (at most tens or hundreds of instructions) and evaluate their throughput individually. For a given combination of a program and input (called *workload* hereafter), the basic blocks and their dependencies create a directed **Control Flow Graph (CFG)** [56] with one source (program start) and one sink (program termination). All intermediate nodes (representing basic blocks) of the graph can have multiple parent- and dependent-nodes, as well as self-references (e.g., basic blocks of `for`-loops). Knowing the "runtime" of each basic block and the number of invocations per basic block, we can estimate the runtime of the entire workload by summation of the parts.

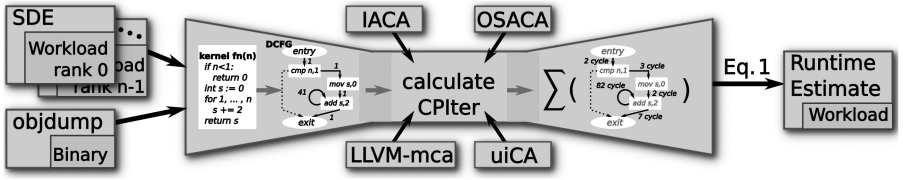


Fig. 4. Illustration of our runtime estimation pipeline with the MCA-based tool for an accumulative kernel executed with $n = 42$; Dotted line: branch not taken; Solid line: kernel execution as recorded by SDE; Edges in directed CFG annotated by number of jumps between basic blocks; Details in Section 3.1.

We utilize the **Software Development Emulator (SDE)** [57] from Intel to record the basic blocks and their caller/callee dependencies for a workload with modest runtime overhead (typically in order of 1000× slowdown). SDE also notes down the number of invocations per CFG edge for a workload, i.e., how often the **program counter (PC)** jumped from one specific basic block to another specific block. We developed a program which parses the output of Intel SDE and establishes an internal representation of the Control Flow Graph. The internal CFG nodes are then amended with Assembly extracted from the program's binary, since SDE's Assembly output is not compatible with Machine Code Analyzers. Our program subsequently executes a Machine Code Analyzer for each basic block, getting in return an estimated **cycles-per-iteration metric (CPIter)**. We record the per-block CPIter at the directed CFG edge from caller to callee, which already holds the number of invocations of this edge, effectively creating a "weighted" graph. Figure 4 showcases the result and it is easy to see that the summation of all edges in the CFG is equivalent to the estimated runtime of the entire workload (assuming all data is inside the L1 data cache).

The above outlined approach works for both sequential and parallel programs. Intel SDE can record the instruction execution and caller/callee dependencies for thread-parallel programs, e.g., pthreads, OpenMP, or TBB. Furthermore, we can attach SDE to individual MPI ranks to get the data for it. Therefore, we are able to estimate the runtime for MPI+X parallelized HPC applications by the following equation:

$$t_{\text{app}} := \frac{\max\limits_{r \in \text{ranks}} \left(\max\limits_{t \in \text{threads}_r} \left(\sum\limits_{\text{edges } e \in \text{CFG}_{t,r}} \text{CPIter}_e \cdot \#\text{calls}_e \right) \right)}{\text{processor frequency in Hz}} \tag{1}$$

under the assumption that MPI ranks and threads do not share computational resources,[1] where we sum up the number of cycles required for each block (i.e., CFG edges) considering only the "slowest" thread and rank, and divide by the CPU frequency to convert the total cycles into runtime.

The self-imposed restriction of Machine Code Analyzers is the limited accuracy compared to cycle-accurate simulators, due to their distinct design goal. To improve our CPIter estimate, we rely on four different MCAs, namely llvm-mca [66], **Intel ACA (IACA)** [55], uiCA [2], and OSACA [65], and take the median of the results. Another shortcoming of MCA tools is that most of them estimate the throughput of basic blocks in isolation while assuming looping behavior of the assembly block (PC jumps from last back to first instruction). Neither "block looping" nor an empty instruction pipeline (single iteration of the block) are realistic for some blocks. Hence, for non-looping basic blocks, we estimate the CPIter by feeding the MCA tool with the blocks of caller and callee, and the callee's CPIter is calculated by subtracting the cycle of retirement of its last instruction from the caller's last instruction retirement (instead of when the callee's first

[1]Resource over-subscription is outside the scope of this study and our tool.

instructions are decoded, which can overlap with execution of caller instructions). Further, we correct some cycle estimates for specific instructions within our tool in post-processing, since we encountered a few unsupported or grossly mis-estimated instructions while validating our tool against benchmarks. We refer the reader to Section 4.1 for more details.

3.2 Cycle-level Accuracy: CPUs Simulated in gem5

While the MCAs can give a first-order approximation, we still require highly accurate predictions for our 3D-stacked, cache-rich CPU. Hence, we employ an open-source system architecture simulator, called gem5 [13]. It supports Arm, x86, and RISC-V CPUs to varying degrees of accuracy, and can be extended with memory models for higher simulation fidelity of the memory subsystem. We use gem5's "syscall emulation" mode to execute applications directly without booting a Linux kernel.

Fortunately, RIKEN released their gem5 version which was specially tailored for A64FX's co-design to support SVE, HBM2, and other advanced features [94]. Hence, it is well suited to simulate our LARC proposal in Section 2.4. This version of gem5 has been validated for A64FX [62], and can be used with production compilers from Fujitsu. Albeit, while evaluating RIKEN's gem5, we noticed a few drawbacks, such as the lack of support for: (i) dynamically linked binaries; (ii) adequate memory management (freeing memory after application's `free()` calls); (iii) simulating more than 16 CPU cores due to limits in the cache coherence protocol; (iv) multi-rank MPI-based programs; and (v) simulating more than one A64FX CMG.

We modify gem5 to remedy the first three problems. However, the last two problems remain intractable without major changes to the simulator's codebase, and hence we limit ourselves to single-CMG simulations (with one MPI rank). Relying on the assumption that most HPC codes are weak scaled across multiple NUMA domains and compute nodes, we believe the single-rank approach still serves as a solid foundation for future performance projection. However, even single-rank MPI binaries require numerous unsupported system calls. To circumvent this problem, we extend and deploy an MPI stub library [101].

3.3 Relevant HPC (Proxy-)Apps and Benchmarks

Instead of relying on a narrow set of cherry-picked applications, we attempt to cover a broad spectrum of typical scientific/HPC workloads. We customize and extend a publicly available benchmarking framework[2] [34, 35] with a few additional benchmarks and necessary features to perform the MCA- and gem5-based simulations. The benchmark complexity ranges from simple kernels to large code bases (O(100,000s) lines-of-code) which are used by vendors for architecture comparisons and used by HPC centers for hardware procurements [41]. Hereafter, we detail the list of 127 included workloads, summed up across all benchmark suites, which are sized to fit within a single node and which could be simulated with gem5 in a reasonable time ($\leq$ six months).

Polyhedral Benchmark Suite. The PolyBench/C suite contains 30 single-threaded, scientific kernels which can be parameterized in memory occupancy ($\in$ [16 KiB, 120 MiB]) [90]. Unless stated otherwise, we use the largest configuration.

TOP500, STREAM, and Deep Learning Benchmarks. **High Performance Linpack (HPL)** [36] solves a dense system of linear equations $Ax = b$ of size 36,864 in our case. **High Performance Conjugate Gradients (HPCG)** [37] applies a conjugate gradient solver to a system of linear equation (with sparse matrix A). We choose 120^3 for HPCG's global problem size. BabelStream [29] evaluates the memory subsystem of CPUs and accelerators, and we configure 2 GiB input vectors.

[2]Exact benchmark versions, git commits, inputs, and the like, are provided in our artifacts which are referenced in Section 9.

Moreover, we implement a micro-benchmark, DLproxy, to isolate the single-precision GEMM operation ($m = 1577088; n = 27; k = 32$) which is commonly found in 2D deep convolutional neural networks, such as 224×224 ImageNet classification workloads [111].

NASA Advanced Supercomputing Parallel Benchmarks. The **NAS Parallel Benchmarks (NPB)** [11, 110] consists of nine kernels and proxy-apps which are common in **computational fluid dynamics (CFD)**. The original MPI-only set has been expanded with ten OpenMP-only benchmarks [60] and we select the `class B` input size for all of them.

RIKEN's Fiber Mini-Apps and TAPP Kernels. To aid the co-design of Supercomputer Fugaku, RIKEN developed the Fiber proxy-application set [92], a benchmark suite representing the scientific priority areas of Japan. Additionally, RIKEN released scaled-down TAPP kernels [93] of their priority applications which are tailored for fast simulations with gem5 [62]. Our workloads are as follows: *FFB* [46] with the 3D-flow problem discretized into 50×50×50 sub-regions; *FFVC* [84] using 144×144×144 cuboids; *MODYLAS* [9] with the `wat222` workload; *mVMC* [73] with the strong-scaling test reduced to 1/8th of the samples and 1/3rd of the lattice size; *NICAM* [108] with a single (not 11) simulated day; *NTChem* [78] with the H_2O workload; *QCD* [16] with the `class 2` input.

Exascale Computing Project Proxy-Applications. The US-based supercomputing centers curated a co-design benchmarking suite for their recent exascale efforts [41]. We select eleven applications of the aforementioned benchmarking framework with the following workloads. *AMG* [87] with the `problem 1` workload; *CoMD* [75] with the 256,000-atom strong-scaling test; *Laghos* [32] modelling a 3D Sedov blast but with 1/6th of the timesteps; *MACSio* [31] with an ≈ 1.14 GiB data dump distributed across many `JSON` files; *MiniAMR* [50] simulating a sphere moving diagonally through 3D space; *MiniFE* [50] with 128×128×128 grid size; *MiniTri* [116] testing triangle- and largest clique-detection on `BCSSTK30` (MatrixMarket [15]); *Nekbone* [10] with 8,640 elements and polynomial order of 8; *SW4lite* [88] simulating a `pointsource`; *SWFFT* [47] with 32 forward and backward tests for a 128×128×128 grid; *XSBench* [109] with the `small` problem and 15 million particle lookups.

3.3.1 SPEC CPU & SPEC OMP Benchmarks. The Standard Performance Evaluation Corporation [102] offers, among others, two HPC-focused benchmark suits: SPEC CPU® 2017[speed] (ten integer-heavy, single-threaded; ten OpenMP-parallelized, floating-point benchmarks) and SPEC OMP® 2012 (14 OpenMP-parallelized benchmarks). All SPEC tests hereafter are based on non-compliant runs with the `train` input configuration.

4 MCA-BASED SIMULATION RESULTS

Sections 4.1 and 4.2 are dedicated to our MCA-based estimation of the upper bound on performance improvement with abundant L1 cache. First, we evaluate the accuracy of this approach, and then apply the novel methodology to our benchmarking sets.

4.1 MCA-based Simulator Validation

During the development of our MCA-based simulator, we implemented numerous micro-benchmarks to fine-tune the CPI estimation capabilities while comparing the results to an Intel® Xeon® processor E5-2650v4 (formerly code named Broadwell). Our micro-benchmarks comprise MPI-/OpenMP-only, MPI+OpenMP, and single-threaded tests (exercising recursive functions, floating-point- or integer-intensive operations, L1-localised, or stream-like operation).

Needless to say, applying MCA-based simulations to full workloads or complex application kernels is still error-prone, since these tools are designed to analyze small Assembly sequences without guarantee for accurate absolute performance numbers. Regardless, we validate the current

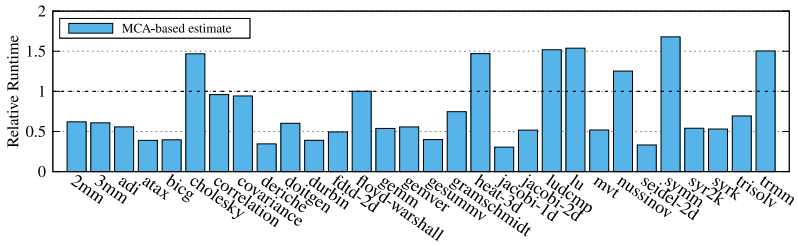


Fig. 5. Validation of MCA-based runtime predictions against PolyBench/C `MINI` with inputs fitting into L1D; Relative runtime shown (vs. Intel E5-2650v4 measurements); Values ≤ 1 show prediction of faster execution.

status of our tool using PolyBench/C with `MINI` inputs. In theory, these input sizes (≈ 16 KiB) should all fit into the 32 KiB L1D cache of the Broadwell. Hence, measuring the kernel execution time for these PolyBench tests should yield numbers close to MCA-based runtime estimates. For the baseline measurements, we set all cores of the Broadwell to 2.2 GHz, set the uncore to 2.7 GHz, and disable turbo boost; compile each workload with Intel's Parallel Studio XE,[3] and execute every test for 100 times (since many only run for a few ms) to determine the fastest possible execution time. The difference between the real baseline results and our MCA-based estimates is visualized in Figure 5 as projected relative runtime difference.

The data shows that on average our MCA-based method slightly overestimates: MCA approach predicts faster execution times than it should. Only seven out of 30 workloads are expected to run slower than what we observe on the real Broadwell (i.e., y-value ≤1). For eight of the PolyBench tests, our tool estimates the runtime to be over 2× faster than our measurements. Hence, we can conclude that for 73% of the micro-benchmarks, the MCA-based method is reasonably accurate: within 2× slower-to-2× faster. While a 2× discrepancy might appear high, we have to point out that our cross-validations using SST [95, 113] and third-party gem5 models [7] for Intel CPUs yield similar inaccuracies,[4] but our MCA-based method is substantially faster.

Another indicator for the accuracy of our MCA-approach can be drawn from **DGEMM (double precision `gemm`** benchmark in Figure 5). Theoretically, DGEMM performs close to peak and is not memory-bound for large matrices, and hence the measured runtime and MCA-based estimates are expected to match. Unfortunately, PolyBench's Gflop/s rate for `gemm` is far from peak (due to its hand-coded loop-nest), and therefore we replace it with an Intel MKL-based implementation of equal matrix dimensions. For the PolyBench input sizes `MINI`, . . . , `EXTRALARGE` in our MKL-based implementation, our MCA tool estimates a faster runtime by 6.4×, 75%, 11%, 1.9%, and 1.5%, respectively. This closely matches the achievable single-core Gflop/s of the E5-2650v4: for `MINI` and the MKL-based runs, we measure only 2 Gflop/s, while for `EXTRALARGE` we peak out at the expected 32 Gflop/s. The low Gflop/s measurements for `MINI` (and `SMALL`) demonstrate that MKL is not yet compute-bound, and hence causes the 6.4× (and 75%) misprediction.

4.2 Speedup-potential with Unrestricted Locality

In this section, we take on the entire benchmark suite from Section 3.3 with the MCA-based approach and evaluate their speedup potential when all data fits into L1.

[3]For details of flags, tools, versions, and executions environments, please refer to Section 9.

[4]A large-scale survey of academic simulators in realistic scenarios, beyond carefully selected and tuned micro-kernels, is—in our humble opinion—consequential, and yet outside the scope of this paper. Although, reference [6] provides a data point.

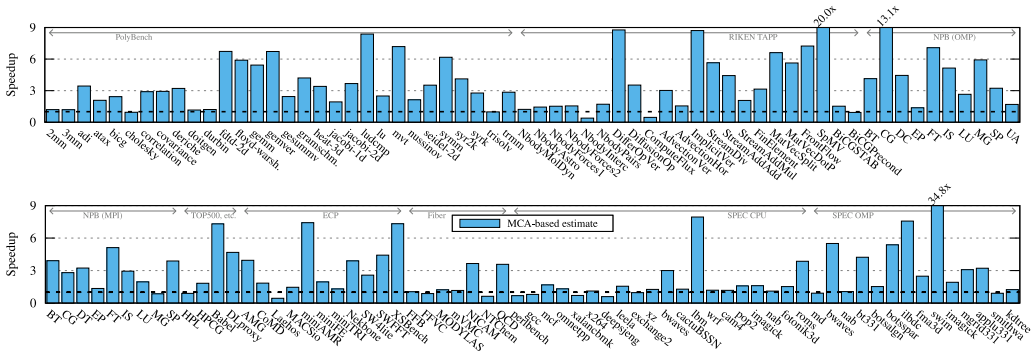


Fig. 6. Projected speedup against a baseline dual-socket Intel Broadwell E5-2650v4 system while assuming all data fits into L1D with "optimistic" load-to-use latency; Top row, left to right: PolyBench, RIKEN TAPP kernels, NPB (OMP); Bottom row, left to right: NPB (MPI), TOP500, etc., ECP proxies, RIKEN Fiber apps, SPEC CPU[int/single] and CPU[float/OMP], SPEC OMP.

The baseline measurements for the speedup estimates are conducted on a dual-socket Intel Broadwell E5-2650v4 system with 48 cores (2-way hyper-threading enabled, cores are set to 2.2 GHz, turbo boost disabled). For all listed benchmarks, excluding SPEC CPU and OMP, we focus on the solver times only, i.e., we ignore data initialization and post-processing phases. Since most proxy-apps are parallelized with MPI and/or OpenMP, we perform an initial sweep of possible configurations of ranks and threads to determine the fastest **time-to-solution (TTS)** for our strong-scaling benchmarks, and the highest figure-of-merit (as reported by the benchmarks) for weak-scaling workloads. The highest performing configurations is executed ten times to determine the TTS of the kernel as our reference point in Figure 6.

The same MPI/OMP configurations are then used for our MCA-based estimate. Under the assumption that some MPI-parallized benchmarks experience imbalances, we randomly sample up to nine ranks (in addition to rank 0),[5] execute the selected rank with Intel SDE (and the remaining ranks normally), and calculate the estimated runtime using Equation (1) and the 2.2 GHz processor frequency. The resulting runtime estimate is divided by the measured runtime to determine the upper-bound speedup potential per application when all its data would fit into L1D, see Figure 6.

For PolyBench/C workloads, we see similar speedup trends as for its smallest inputs which we used in Figure 5, although the expected speedup for EXTRALARGE increases to a peak of 8.4× for the ludcomp kernel. Only four kernels show no performance increase, presumably by being compute-bound and not bandwidth-bound: 2mm, 3mm, doitgen, and trisolv. Overall, the MCA-based approach estimates a **geometric mean (***GM***)** speedup of 2.9× from fitting all data into L1D. RIKEN's TAPP kernels benefit the most from unrestricted locality. Especially kernel 20 (SpMV), which represents one core function of the FFB application, shows a speedup of 20×. Altogether, we see a projection of (*GM*=)2.6× increased performance, but also two cases (kernels 5 and 9) where the MCA tool estimates a ≈ 50% slowdown. These two are from GENESIS [61] and NICAM, respectively, but as detailed in Section 4.1, some inaccuracy is expected as the trade-off for the faster simulation time.

NPB's OpenMP version of a **conjugate gradient (CG)** solver is another workload with a large theoretical performance gain of 13.1×. In total, we expect a (*GM*=)3× gain for all NAS Parallel Benchmarks; specifically, (*GM*=)4× for the OpenMP versions and (*GM*=)2.3× for the MPI

[5]Sampling at most ten out of all MPI ranks should not substantially alter the result but saves resources, since we have to execute SDE once per rank.

versions. The potential gain for CG is not surprising, since these solvers are predominantly bound by memory bandwidth and are sensitive to memory latency [38]. High Performance Linpack is unsurprisingly not expected to gain any performance by placing all its data into L1 cache, as this benchmark is compute-bound. In fact, our MCA tool expected a small runtime decrease of 11%. By contrast, DLproxy, which uses MKL's SGEMM, would benefit from a large L1, since MKL cannot achieve peak Gflop/s for the tall/skinny matrix in this workload (cf. Section 3.3). XSBench and miniAMR show the highest gains for ECP's and RIKEN's proxy-apps, with a value of 7.3× and 7.4×, respectively. This appears to be in line with the expectation from the roofline characteristics of the benchmarks when measured on a similar compute node [33].

A deeper look at roofline analysis in [33] reveals that there is no strong correlation between the position of an application on the roofline model and the expected performance gain from solely running out of L1D cache. We speculate that other, hidden bottlenecks are exposed by our MCA approach, such as data dependencies and lack of concurrency in the applications, which limit the expected speedup. Apart from noticeable outliers in the expected speedup, such as lbm, ilbdc, and especially swim, the potential from enlarged L1D is rather slim for SPEC, and only (*GM*=)1.9× runtime reduction can be expected across all 34 workloads.

5 GEM5-BASED SIMULATION RESULTS

In Section 5.1, we detail our choice for the simulated architectures in gem5. Similarly structured to the MCA-based simulations, Sections 5.2 and 5.3 highlight our validation of gem5 for our proposed CPU architectures and evaluate numerous benchmarks and proxy applications on said architecture, and we summarize the results in Section 5.4.

5.1 LARC CMG Models in gem5 and A64FX$_S$ Baseline

As we discussed in Section 2.4, we envision one LARC CMG to have 32 cores, 384 MiB L2 cache, and 1.6 TB/s L2 bandwidth. Regretfully, gem5 (at least RIKEN's version) can only be configured with L2 cache sizes that are 2^X, and therefore we either have to scale up or down LARC's L2 cache size. Hence, we explore both as distinct options, one conservative and one technologically aggressive configuration. The conservative option, called LARC$_C$, is limited to 256 MiB L2 cache at ~ 800 GB/s, while the aggressive version, LARC^A, doubles both values, to 512 MiB and ~ 1.6 TB/s, respectively.

Starting at a baseline, i.e., a simulated version of A64FX which we label as A64FX$_S$, and in order to materialize the properties of the LARC CMG (cf. Section 2.4), we modify three parameters in our gem5 model. We modify: (i) the number of cores in the system to match 32 (up from A64FX$_S$' baseline of 12); (ii) the size of the total L2 cache to match the capacity of the eight stacked layers (256/512 MiB, up from A64FX$_S$' L2 size of 8 MiB per CMG); and (iii) we adjust the number of L2 banks in LARC^A to control the bandwidth.

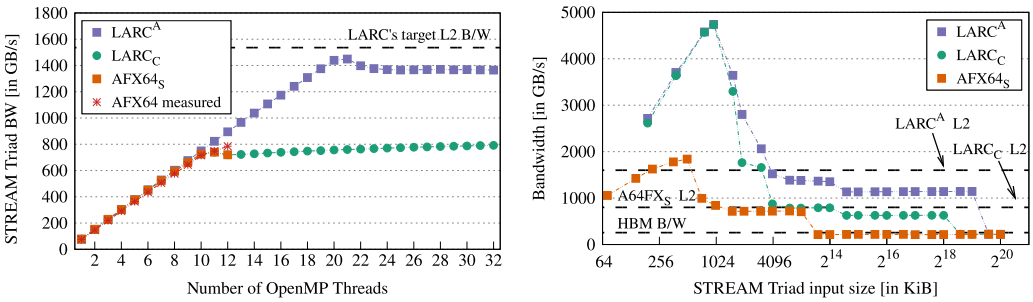
We introduce a fourth gem5 configuration, called A64FX^{32}, which simulates one baseline A64FX$_S$ CMG but with 32 cores. These four configurations A64FX$_S$ →A64FX^{32} →LARC$_C$ →LARC^A should allow us to determine the speedup gains from the larger core count and larger L2 cache, individually. The core frequency is universally set to 2.2 GHz. Table 2 summarizes the four gem5 configurations.

5.2 gem5-based Simulation and Configuration Validation

We perform OpenMP tests to verify our gem5 simulator for up to 32 cores. For the L2 cache size and bandwidth changes, we employ a STREAM Triad benchmark, parameterized to avoid cache line conflicts among participating threads. Splitting the A64FX$_S$ CMG L2 cache into 12 chunks (one per thread) yields a working size of 683 KiB. Hence, the three 128 KiB vectors of the Triad operation

Table 2. Chip Area and Simulator Configurations for gem5

	A64FX$_S$	**A64FX^{32}**	**LARC$_C$**	**LARC^A**
Cores	12	32	32	32
CMGs	4	4	16	16
Core config.	Arm v8.2 + SVE, 512 bit SIMD, 2.2 GHz, OoO 128 ROB entries, dispatch width 4			
Branch pred.	Bi-mode: 16 K global predictor, 16 K choice predictor			
Per-core L1D	64 KiB 4-way set-assoc, 3 cycles, adjacent line prefetcher			
	L2 cache per CMG:			
L2 size	8 MiB	256 MiB		512 MiB
BW	~ 800 GB/s	~ 800 GB/s		~ 1600 GB/s
	L2 Cache Aggregated:			
L2 size	32 MiB	4096 MiB		8192 MiB
BW	~ 3.2 TB/s	~ 12.8 TB/s		~ 25.6 TB/s
L2 config.	16-way set-associative, 37 cycles, inclusive, 256 B block			
Main Memory	32 GiB HBM2, 4 channels, 256 GB/s			



(a) Validation using fixed 128 KiB vectors per core (b) Validation using input range from few KiB to 1 GiB

Fig. 7. Validation with simulated STREAM Triad; Both LARC configurations with 32 cores; A64FX$_S$ scaled to 12 cores; Real A64FX measurements on 1 CMG for reference; Dashed lines highlight trend (not measured).

will fit into the L2 cache. We increase the total vector size in proportion to the number of threads and test the achievable L2 bandwidth for LARC$_C$ and LARC^A. Additionally, Figure 7(a) includes the baseline A64FX$_S$ CMG scaled to 12 cores. The simulation shows that LARC$_C$'s L2 bandwidth peaks out at 792 GB/s and LARC^A's bandwidth goes up to 1450 GB/s for this particular test case, which is, respectively, 1% and 9% lower than our estimates shown above. The baseline A64FX$_S$ closely matches the bandwidth of the real A64FX CPU executing this test.

Another validation test we perform is setting the number of cores to the maximum (12 and 32, respectively) and scale the vector size from 2 KiB per core to a total of 1 GiB for the three vectors. Figure 7(b) shows the results for this simulation. In the memory range of tens to hundreds of KiB, the Triad operation can be done from L1 cache, for which LARC$_C$ and LARC^A show higher bandwidth. Their 2.7× higher core count results in 2.6× higher aggregated L1 bandwidth. For the Triad, for the memory sizes that fit into L2 cache, we see a behavior similar to Figure 7(a). Past 8 MiB, the A64FX$_S$ configuration shows the expected bandwidth drop to HBM2 level, while for LARC$_C$ and LARC^A, the expected L2 cache bandwidth is maintained until 256 MiB and 512 MiB, respectively. This validates that our gem5 settings yield the expected LLC characteristics.

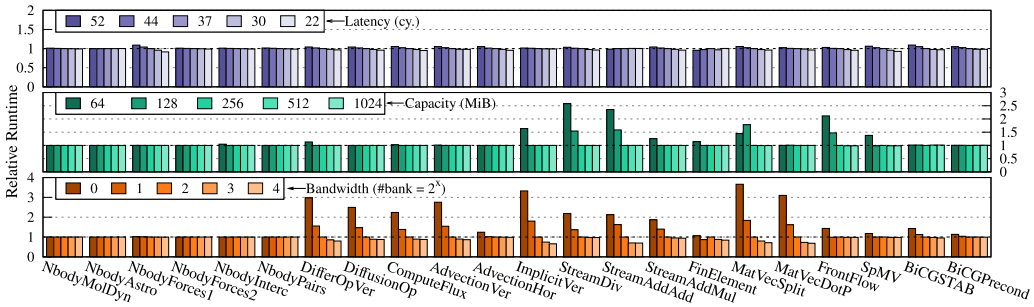


Fig. 8. Sensitivity study of cache parameters using RIKEN's TAPP [93] kernels; Relative runtime compared to LARC_C baseline (37 cycle latency, 256 MiB, 2 bankbits; middle bar among the five) is shown; Top row: L2 latency modified; Middle row: L2 capacity; Bottom row: adjusting L2 bandwidth via bankbits (#banks = 2^x).

Lastly, to validate the LARC configuration and to see the changes applied to more complex science kernels, we perform a sensitivity analysis of cache parameters with the RIKEN TAPP kernels. In Figure 8, we vary L2 cache access latency, size, and bandwidth in ranges beyond our LARC_C and LARC^A target architectures. This analysis will help us in adjusting our expectations when future LARC-like architectures deviate from our design parameters, e.g., by stacking less SRAM layers or having higher L2 access latency. In this parameter sweep, LARC_C will be the baseline and we vary one parameter while keeping the others fixed. The top row of Figure 8 shows the latency sweep, where we choose 22 cycles as best latency (which is 2× the data load latency from L1 for SVE instructions in A64FX). The worst case of 52 cycles is equidistant to our baseline in the opposite direction, and two additional latencies are selected in between. Similarly, we adjust the L2 size (middle row; simulating more or less SRAM stacks or a larger or smaller semiconductor process nodes) and L2 bank bits in gem5, see bottom row of Figure 8. The latter indirectly controls the L2 bandwidth of the simulated architectures. The latency change has minimal impact, since HPC applications are typically not latency bound. However, the L2 cache capacity and bandwidth can have a significant impact on performance, as expected, since they determine the amount of data that can be stored and accessed quickly. For some of the TAPP kernels, though, the performance is unaffected by these parameters,[6] since these kernels are actually shrunk-down versions specifically designed for cycle-level architecture simulations, and therefore have low memory footprint.

5.3 Speedup-potential with Restricted Locality

To further refine our projections gained by abundant cache, we proceed with the cycle-level simulations of the proxy-applications and benchmarks listed in Section 3.3.

We compile all benchmarks with Fujitsu's Software Technical Computing Suite (v4.6.1) targeting the real A64FX, and simulate the single-rank workloads in gem5 for our four configurations. Unfortunately, three of our MPI-based benchmarks require multi-rank MPI: MODYLAS, NICAM, and NTChem, and hence we omit them. Furthermore, we skip the MPI-only versions of NPB. Hereafter, we only report proxy applications and benchmarks which ran to completion within gem5 (i.e., gem5-crashes or simulated application-crashes are excluded when infeasible to patch, and simulations exceeding the 6-months time limit are ignored).

[6]The MatVecSplit oddity (runtime increases for 128 MiB) needs further investigation. It shows an enlarged counter of `LoadLockedRequests`—this artifact could be attributed to software (such as barrier implementation in the OpenMP runtime).

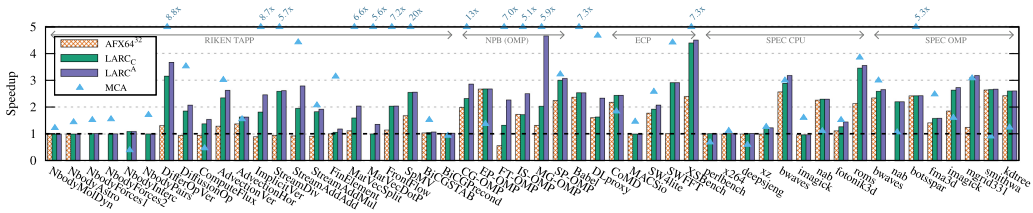


Fig. 9. gem5-based, simulated speedups of A64FX^{32}, LARC$_C$ and LARC^A in comparison to baseline A64FX$_S$; Left to right: RIKEN TAPP kernels, NPB (OMP), TOP500 etc., ECP proxies, SPEC CPU[int/single] and CPU[float/OMP], SPEC OMP; Added MCA-based estimations from Figure 6 for reference; TAPP kernels 3–6 (multiple Nbody kernels) and 18 (MatVecDotP) are limited to 12 threads, hence we omit A64FX^{32}; Missing benchmarks (cf. Figure 6) primarily due to gem5 issues or exceeding simulation time limit. PolyBench results (single core) are also omitted due to limited speedup across all of them and no noteworthy outliers.

The per-configuration speedup is given relative to the baseline A64FX$_S$ configuration. We exclude initialization and post-processing times, and measure only the main kernel runtime, except for the SPEC benchmarks as described in Section 4.2. These results are presented in Figure 9 and show the effects of the gradual expansion of simulated resources. The average (single CMG) speedups from LARC$_C$ and LARC^A are $\approx 1.9\times$ and $\approx 2.1\times$, respectively, with some applications reaching $\approx 4.4\times$ for LARC$_C$ and $\approx 4.6\times$ for LARC^A.

As expected, most benchmarks benefit from the additional cores and cache capacity, most prominently MG-OMP which gains a small speedup of $\approx 1.3\times$ from the extra cores, $\approx 2\times$ speedup from the extra cache, and with 512 MiB cache and higher bandwidth reaches $\approx 4.6\times$ speedup. Comparable incremental improvements with all three architecture steps are observable in other workloads, such as TAPP kernel 7 (DifferOpVer) and 17 (MatVecSplit), showing good scaling on multiple cores and being memory-bound since they benefit from the additional cores and cache capacity. TAPP kernels 19 and 20, XSBench, roms, and imagick (SPEC OMP) show similar gain in runtime, but the difference between LARC$_C$ and LARC^A is smaller, implying that the problem size either fits into the 256 MiB L2 (e.g., XSBench) or the workload arrives at a point of diminishing returns from the 2× larger cache. TAPP kernels 8, 9, 12–15, and FT-OMP suffer a slowdown from cache contention in A64FX^{32}. LARC$_C$ and LARC^A avoid the cache contention, resulting in speedups similar to the benchmarks discussed earlier. EP-OMP, CoMD, and other compute-bound benchmarks benefit only from the higher core count, with both LARCs providing similar speedup as A64FX^{32}.

Expectedly, single-threaded workloads (all of PolyBench's benchmarks) show little to no improvements over A64FX$_S$, i.e., they do not benefit from more cores. However, these benchmarks also do not show a performance gain from a larger 3D-stacked L2 cache, albeit their working set size exceeding A64FX$_S$' 8 MiB L2 yet fitting into LARC' larger cache. We only see a limited speedup of (*GM*=)4.3% across all of them and no noteworthy outliers, and hence omit them in Figure 9. We attribute other outliers, such as the slowdown of imagick (SPEC-CPU), to similar intrinsic property of the benchmark: our testing on a real A64FX reveals that imagick has a sweetspot at 8 OpenMP threads, and scales negatively thereafter; and the TAPP kernels 3–6 and 18 were customized for the 12-core A64FX CMG and cannot run effectively on 32 threads without a rewrite. Hence, we limit gem5 to 12 cores for these TAPP kernels, and we see that only the MatVecDotP kernel of the ADVENTURE application [4] benefits from a larger L2. Further proxy-applications and benchmarks missing from Figure 9, yet appearing in Figure 6, are the unfavorable result of persistent, repeatable simulator errors—sometimes occurring after months of simulation.

We should note that in some cases the benchmarks' implementation and the quality of the compiler may skew the results, for instance, BabelStream measuring memory bandwidth on a 2 GiB

Table 3. L2 Cache-miss Rate [in %] of Representative Proxies

Proxy-App	A64FX$_S$	A64FX^{32}	LARC$_C$	LARC^A
NICAM's ImplicitVer (kernel 12)	36.6	47.6	10.5	9.1
ADVENTURE's MatVecSplit (kernel 17)	46.7	49.5	48.7	34.8
FFB's FrontFlow (kernel 19)	73.8	69.6	49.1	48.9
FT-OMP	11.6	48.2	6.4	3.8
MG-OMP	59.8	70.9	29.4	0.4
XSBench	32.1	36.4	0.1	0.1

buffer. Being unoptimized for A64FX, BabelStream's baseline underperforms in terms of per-core bandwidth (compared to STREAM Triad tests in Figures 7(a) and 7(b)) which in turn results in performance gain when the number of cores increases to 32.

Overall, the speedup on A64FX^{32} can originate from the following reasons: (i) the program is compute-bound (a valid result); (ii) the workload exhibits both compute-bound and memory-bound tendencies in different components of a proxy-application (a valid result); (iii) the program is highly latency-bound, and hence the speedup can be the result of the larger aggregate L1 cache (a valid result); or (iv) a poor baseline resulting in a slightly misleading result.

We confirm the validity of attributing improvement to the high capacity L2 by inspecting the L2 cache-miss rates of our gem5 simulations (with the miss rate of some selected examples listed in Table 3). The reduction in cache-miss rates reported in the table is consistent with the performance improvements we observe in Figure 9.

5.4 Summary of the Results

Our gem5 simulations indicate that more than half (31 out of 52) of the applications experienced a larger than two times speedup on LARC^A compared to the baseline A64FX$_S$ CMG. For over two-thirds (24 out of 31) of these applications, the performance gains are directly attributed to the larger (3D-stacked) cache, i.e., with at least 10% gain by either of the two LARC configurations over the A64FX^{32} variant. Most notably, out of all the RIKEN TAPP kernels that experienced meaningful speedup on LARC, a majority benefited from the expanded cache, rather than the increase in number of cores. This carries particular importance as these kernels are highly tuned for A64FX.

6 DISCUSSION AND LIMITATIONS

In this study, we simulated a single LARC CMG in gem5, and its potential future effect on common HPC workloads.

6.1 The Prospect of LARC

In reality, if a LARC processor were incepted in 2028, it would contain 16 LARC CMGs, which correspond to the same silicon area as the current A64FX CPU, and it is important to understand what impact such a processor would have on the HPC community and its applications. Unfortunately, it is hard to give a conclusive answer to such a forward looking question today. However, if we do ideal scaling of both A64FX and LARC CMGs and compare at the full chip level, then a LARC system in 2028 could give between 4.91× (`xz`; SPEC CPU) and 18.57× (`MG-OMP`; NPB) performance improvements over the current A64FX processor with an average improvement of (GM=)9.56× for applications that are responsive to larger cache capacity. For applications that do not yet benefit from a larger cache, future studies should (continue to) consider algorithmic improvements [69],

as well as investigate the potential of allocating parts of the cache to vary compute capabilities, for example, processing-in-memory [12] or alternative compute modules, e.g., CGRAs [89].

6.2 Considerations and Limitations

Our MCA-based estimation framework only gives a first-order approximation for a hypothetical CPU with sufficiently large L1 cache to host the entire data structures of a specific workload. This approach has some advantages and disadvantages and should be used with caution, but it also has capabilities which we have not yet detailed, such as estimating the runtime of the same binary/workload for different (ISA-compatible) x86 systems by simply replacing the MCA target architecture and altering the CPU clock frequency.

We emphasize that we run applications as they are, i.e., without any algorithmic optimizations to the larger last level cache, in our MCA- and gem5-based simulators. This is also true to our motivating experiment shown in Figure 1. While the cache capacity of AMD's Milan-X CPU is about three times that of Milan, it is far from what we envision in 2028. Hence, our Milan-X results serve as a first-order indication of what SRAM—in its current available SoTA—can offer.

Another notable aspect, which is outside the main scope of this extrapolation study, is the heat dissipation of CPU cores in the face of the 3D-stacked cache. It has been reported that AMD's Milan-X carefully stacks caches above areas of the chip that are not used for compute, i.e., mostly above caches [77]. Our assumption is that, by 2028, manufacturing technologies will have advanced enough to overcome this limitation. Yet, for interested readers we provide further details on thermal and power estimates for our hypothetical LARC CPU in Section 2.6.

7 RELATED WORK

Stacked Memory and Caches: The size of LLC has increased for the last 25 years [58], a trend anticipated to continue into the future. Yet, 2D IC becomes hard to exploit for additional performance, despite recent attempts by IBM [27, 59]. However, 3D-stacking is becoming a promising alternative [52], as demonstrated by AMD's 3D V-Cache [40], Samsung's proposed 3D SRAM stacking solution [64] based on 7 nm TSVs, or the most recent study of 7 nm TCI-based 2- and 4-layer SRAM stacks by Shiba et al. [97]. Moreover, academics explored 3D-stacked DRAM cache [48, 119], but these incur much higher latency and power consumption [74, 98]. Non-Volatile Memory is considered as LLC alternative, yet it suffers similar latency issues [63]. Lastly, NVIDIA applied for a patent of an 8-layer memory stack fused with a processor die [28], theorizing a 50× improvement in bytes-to-flop ratio. However, what differs our work from the work of our peers is: (i) we focus on the real-world impact of future caches, several magnitudes larger than those found today.

Performance modeling tools and methodologies: Computer architecture research is often based on simulators, such as the **Structural Simulation Toolkit (SST)** [95] or CODES [24], for efficiently evaluating and optimizing HPC architectures and applications. The gem5 simulator, by Binkert et al. [13], is widely used by academia and vendors for micro-and full-system architecture emulation and simulation. It supports validated models for x86 [7] and Arm [62]. We refer the interested reader to www.gem5.org/publications/ for an comprehensive library of gem5-based research and derivative works. However, what differs our work from the work of our peers is: (ii): unlike prior work that utilizes (relatively) small kernels, our work operates on large-scale MPI/OpenMP-parallelized proxy-applications in order to quantify the impact of caches on realistic workloads. To our knowledge of reported research-driven gem5 simulations, this is the largest scale of cycle-accurate simulations conducted in terms of the aggregate number of instructions simulated (6.08×10^{13}).

Other methods such as MUSA by Grass et al. [45] are closer to our MCA-based approach, since MUSA uses PIN which is the basis for Intel SDE (used in this study), but focus on MPI analysis

and multi-node workloads. We are not the first to utilize Machine Code Analyzers, see [2, 65] and derivative works such as [8, 19, 22, 25, 72, 91]. However, what differs our work from the work of our peers is: (iii): instead of estimating accurate performance of existing system architectures, our MCA-based approach tries to gauge the upper-bound in obtainable performance, and exposes bottlenecks better than the roofline approach, for common HPC applications.

8 CONCLUSION

We aspire to understand the performance implications of emerging SRAM-based die-stacking on future HPC processors. We first designed a methodology to project the upper bound that an infinitely large cache would have on relevant HPC applications. We find that several well-known HPC applications and benchmarks have ample opportunities to exploit an increased cache capacity.

We further expand our study by proposing a hypothetical processor (called LARC) in 1.5 nm technology. This processor would have nearly 6 GiB L2 cache memory; compared to our baseline A64FX$_S$ CPU architecture with 32 MiB L2 cache. Next, we exercise a single LARC CMG using a plethora of HPC applications and benchmarks using the gem5 simulator and contrast the observed performance against the existing A64FX$_S$ CMG. We find that the LARC CMG would (on average) be 1.9× faster than the corresponding A64FX$_S$ CMG, albeit consuming $\frac{1}{4}$th of the area. When area-normalized to the real A64FX CMG (by assuming optimistic ideal scaling), we can expect to see an average boost of 9.56× for cache-sensitive HPC applications by the end of this decade.

Finally, we expect that the larger caches will motivate and facilitate algorithmic advances that in combination with the abundant cache can potentially yield an order of magnitude gain in performance, as demonstrated by the **tile low-rank (TLR)** approximations [69]. These approaches however require a minimum size of the cache to reach their fullest potential. We firmly believe that the combination of high-bandwidth, large, 3D-stacked caches, and algorithmic advances, is the path forward for the next generation of HPC processors when attempting to break the "memory wall".

9 FAIR COMMITMENT BY THE AUTHORS

We developed a framework of scripts and git submodules to manage the R&D of LARC, to set up the benchmarking infrastructure, and to perform the simulations. After cloning our repository https://gitlab.com/domke/LARC (or downloading the artifacts from https://doi.org/10.5281/zenodo.6420658), one has access to all benchmarks (see Section 3.3), patches, scripts, and our collected data. Only minor modifications to the configuration files should be necessary, such as changing host names, paths to compilers, or downloading licensed third-party software, before testing on another system. If users deviate from our OS version (CentOS Linux release 7.9.2009, and `intel_pstate=disable` kernel parameter) then some additional changes might be required.

ACKNOWLEDGMENT

Furthermore, we thank Masazumi Nakamura from AMD for providing us early access to the Milan-X platform.

JD, EV, BG, and MW designed the study; JD and EV conducted the experiments; BG fixed gem5 issues; JD, EV, MW, AP, MP, LZ and PC analyzed the results; SM and AP developed the stacked cache model, and all authors participated in brainstorming and writing the manuscript.

REFERENCES

[1] 2023. *Heterogeneous Integration Roadmap 2023 Edition - Chapter 20: Thermal.* Technical Report. IEEE Electronics Packaging Society. 1–39. https://eps.ieee.org/images/files/HIR_2023/ch20_thermalfinal.pdf

[2] Andreas Abel and Jan Reineke. 2021. A Parametric Microarchitecture Model for Accurate Basic Block Throughput Prediction on Recent Intel CPUs. https://arxiv.org/pdf/2107.14210.pdf
[3] Advanced Micro Devices, Inc. 2021. AMD Instinct™ MI250X Accelerator. https://www.amd.com/en/products/server-accelerators/instinct-mi250x
[4] ADVENTURE Project. 2019. Development of Computational Mechanics System for Large Scale Analysis and Design — ADVENTURE Project. https://adventure.sys.t.u-tokyo.ac.jp/
[5] A. Agarwal, Hai Li, and K. Roy. 2003. A single-V/Sub t/ low-leakage gated-ground cache for deep submicron. *IEEE Journal of Solid-State Circuits* 38, 2 (2003), 319–328. https://doi.org/10.1109/JSSC.2002.807414
[6] Ayaz Akram and Lina Sawalha. 2019. A survey of computer architecture simulation techniques and tools. *IEEE Access* 7 (2019), 78120–78145. https://doi.org/10.1109/ACCESS.2019.2917698
[7] Ayaz Akram and Lina Sawalha. 2019. Validation of the Gem5 simulator for X86 architectures. In *2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, Denver, CO, USA, 53–58. https://doi.org/10.1109/PMBS49563.2019.00012
[8] Christie Alappat, Nils Meyer, Jan Laukemann, Thomas Gruber, Georg Hager, Gerhard Wellein, and Tilo Wettig. 2021. Execution-cache-memory modeling and performance tuning of sparse matrix-vector multiplication and lattice quantum chromodynamics on A64FX. *Concurrency and Computation: Practice and Experience* (Aug. 2021), 30. https://doi.org/10.1002/cpe.6512
[9] Yoshimichi Andoh, Noriyuki Yoshii, Kazushi Fujimoto, Keisuke Mizutani, Hidekazu Kojima, Atsushi Yamada, Susumu Okazaki, Kazutomo Kawaguchi, Hidemi Nagao, Kensuke Iwahashi, Fumiyasu Mizutani, Kazuo Minami, Shin-ichi Ichikawa, Hidemi Komatsu, Shigeru Ishizuki, Yasuhiro Takeda, and Masao Fukushima. 2013. MODYLAS: A highly parallelized general-purpose molecular dynamics simulation program for large-scale systems with long-range forces calculated by fast multipole method (FMM) and highly scalable fine-grained new parallel processing algorithms. *Journal of Chemical Theory and Computation* 9, 7 (2013), 3201–3209. https://doi.org/10.1021/ct400203a
[10] Argonne National Laboratory. 2022. NEK5000. http://nek5000.mcs.anl.gov
[11] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. 1991. The NAS parallel benchmarks – summary and preliminary results. In *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing (SC'91)*. ACM, New York, NY, USA, 158–165. https://doi.org/10.1145/125826.125925
[12] Rajeev Balasubramonian, Jichuan Chang, Troy Manning, Jaime H. Moreno, Richard Murphy, Ravi Nair, and Steven Swanson. 2014. Near-data processing: Insights from a MICRO-46 workshop. *IEEE Micro* 34, 4 (2014), 36–42. https://doi.org/10.1109/MM.2014.55
[13] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The Gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (Aug. 2011), 1–7. https://doi.org/10.1145/2024716.2024718
[14] Bryan Black, Murali Annavaram, Ned Brekelbaum, John DeVale, Lei Jiang, Gabriel H. Loh, Don McCaule, Pat Morrow, Donald W. Nelson, Daniel Pantuso, Paul Reed, Jeff Rupley, Sadasivan Shankar, John Shen, and Clair Webb. 2006. Die stacking (3D) microarchitecture. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 39)*. IEEE Computer Society, USA, 469–479. https://doi.org/10.1109/MICRO.2006.18
[15] Ronald F. Boisvert, Roldan Pozo, Karin Remington, Richard F. Barrett, and Jack J. Dongarra. 1997. Matrix market: A web resource for test matrix collections. In *Proceedings of the IFIP TC2/WG2.5 Working Conference on Quality of Numerical Software: Assessment and Enhancement*. Chapman & Hall, Ltd., London, UK, 125–137. http://dl.acm.org/citation.cfm?id=265834.265854
[16] T. Boku, K. I. Ishikawa, Y. Kuramashi, K. Minami, Y. Nakamura, F. Shoji, D. Takahashi, M. Terai, A. Ukawa, and T. Yoshie. 2012. Multi-block/Multi-core SSOR preconditioner for the QCD quark solver for K computer. *Proceedings, 30th International Symposium on Lattice Field Theory (Lattice 2012): Cairns, Australia, June 24–29, 2012* LATTICE2012 (2012), 188. https://doi.org/10.22323/1.164.0188
[17] Gavin Bonshor. 2022. AMD Releases Milan-X CPUs With 3D V-Cache: EPYC 7003 Up to 64 Cores and 768 MB L3 Cache. https://www.anandtech.com/show/17323/amd-releases-milan-x-cpus-with-3d-vcache-epyc-7003
[18] Kun Cao, Junlong Zhou, Tongquan Wei, Mingsong Chen, Shiyan Hu, and Keqin Li. 2019. A survey of optimization techniques for thermal-aware 3D processors. *Journal of Systems Architecture* 97, C (Aug. 2019), 397–415. https://doi.org/10.1016/j.sysarc.2019.01.003
[19] Victoria Caparrós Cabezas and Phillip Stanley-Marbell. 2011. Parallelism and data movement characterization of contemporary application classes. In *Proceedings of the Twenty-Third Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'11)*. Association for Computing Machinery, New York, NY, USA, 95–104. https://doi.org/10.1145/1989493.1989506

[20] Shounak Chakraborty and Hemangee K. Kapoor. 2018. Analysing the role of last level caches in controlling chip temperature. *IEEE Transactions on Sustainable Computing* 3, 4 (2018), 289–305.
[21] Cheese. 2022. AMD's V-Cache Tested: The Latency Teaser. https://chipsandcheese.com/2022/01/14/amds-v-cache-tested-the-latency-teaser/
[22] Yishen Chen, Ajay Brahmakshatriya, Charith Mendis, Alex Renda, Eric Atkinson, Ondřej Sýkora, Saman Amarasinghe, and Michael Carbin. 2019. BHive: A benchmark suite and measurement framework for validating X86-64 basic block performance models. In *2019 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE Press, Orlando, FL, USA, 167–177. https://doi.org/10.1109/IISWC47752.2019.9042166
[23] Chiachen Chou, Aamer Jaleel, and Moinuddin K. Qureshi. 2015. BEAR: Techniques for mitigating bandwidth bloat in gigascale DRAM caches. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA'15)*. Association for Computing Machinery, New York, NY, USA, 198–210. https://doi.org/10.1145/2749469.2750387
[24] J. Cope, N. Liu, Samuel Lang, C. D. Carothers, and Robert B. Ross. 2011. CODES: Enabling co-design of multi-layer exascale storage architectures. In *Workshop on Emerging Supercomputing Technologies 2011 (WEST 2011)*. OSTI.GOV, Tuscon, Arizona, USA, 1–6. https://doi.org/10.2172/1311761
[25] Stefano Corda, Gagandeep Singh, Ahsan Javed Awan, Roel Jordans, and Henk Corporaal. 2019. Memory and parallelism analysis using a platform-independent approach. In *Proceedings of the 22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES'19)*. Association for Computing Machinery, New York, NY, USA, 23–26. https://doi.org/10.1145/3323439.3323988
[26] Ian Cutress. 2021. AMD Demonstrates Stacked 3D V-Cache Technology: 192 MB at 2 TB/Sec. https://www.anandtech.com/show/16725/amd-demonstrates-stacked-vcache-technology-2-tbsec-for-15-gaming
[27] Ian Cutress. 2021. Did IBM Just Preview the Future of Caches? https://www.anandtech.com/show/16924/did-ibm-just-preview-the-future-of-caches
[28] William James Dally, Carl Thomas Gray, Stephen W. Keckler, and James Michael O'Connor. [n. d.]. Memory Stacked on Processor for High Bandwidth. https://patents.justia.com/patent/20230275068
[29] Tom Deakin, James Price, Matt Martineau, and Simon McIntosh-Smith. 2016. GPU-STREAM v2.0: Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models. In *High Performance Computing*, Michela Taufer, Bernd Mohr, and Julian M. Kunkel (Eds.). Springer, Cham, 489–507.
[30] Robert H. Dennard, Fritz H. Gaensslen, Hwa-Nien Yu, V. Leo Rideout, Ernest Bassous, and Andre R. LeBlanc. 1974. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9, 5 (1974), 256–268. https://doi.org/10.1109/JSSC.1974.1050511
[31] James Dickson, Steven Wright, Satheesh Maheswaran, Andy Herdmant, Mark C. Miller, and Stephen Jarvis. 2016. Replicating HPC I/O Workloads with proxy applications. In *Proceedings of the 1st Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS'16)*. IEEE Press, Piscataway, NJ, USA, 13–18. https://doi.org/10.1109/PDSW-DISCS.2016.6
[32] V. Dobrev, T. Kolev, and R. Rieben. 2012. High-order curvilinear finite element methods for lagrangian hydrodynamics. *SIAM Journal on Scientific Computing* 34, 5 (2012), B606–B641. https://doi.org/10.1137/120864672
[33] Jens Domke, Kazuaki Matsumura, Mohamed Wahib, Haoyu Zhang, Keita Yashima, Toshiki Tsuchikawa, Yohei Tsuji, Artur Podobas, and Satoshi Matsuoka. 2019. Double-precision FPUs in high-performance computing: An embarrassment of riches?. In *2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019, Rio de Janeiro, Brazil, May 20–24, 2019*. IEEE Press, Rio de Janeiro, Brazil, 78–88. https://doi.org/10.1109/IPDPS.2019.00019
[34] Jens Domke and Emil Vatai. 2021. Matrix Engine Study. https://gitlab.com/domke/MEstudy
[35] Jens Domke, Emil Vatai, Aleksandr Drozd, Chen Peng, Yosuke Oyama, Lingqi Zhang, Shweta Salaria, Daichi Mukunoki, Artur Podobas, Mohamed Wahib, and Satoshi Matsuoka. 2021. Matrix engines for high performance computing: A paragon of performance or grasping at straws?. In *2021 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2021, Portland, Oregon, USA, May 17–21, 2021*. IEEE Press, Portland, Oregon, USA, 1056–1065.
[36] Jack Dongarra. 1988. The LINPACK benchmark: An explanation. In *Proceedings of the 1st International Conference on Supercomputing*. Springer-Verlag, London, UK, UK, 456–474. http://dl.acm.org/citation.cfm?id=647970.742568
[37] Jack Dongarra, Michael Heroux, and Piotr Luszczek. 2015. *HPCG Benchmark: A New Metric for Ranking High Performance Computing Systems*. Technical Report ut-eecs-15-736. University of Tennessee. https://library.eecs.utk.edu/pub/594
[38] Jack Dongarra, Michael A. Heroux, and Piotr Luszczek. 2016. A new metric for ranking high-performance computing systems. *National Science Review* 3, 1 (2016), 30–35. https://doi.org/10.1093/nsr/nwv084
[39] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA'11)*. Association for Computing Machinery, New York, NY, USA, 365–376. https://doi.org/10.1145/2000064.2000108

[40] Mark Evers, Leslie Barnes, and Mike Clark. 2022. The AMD next generation Zen 3 core. *IEEE Micro* 42, 3 (2022), 7–12. https://doi.org/10.1109/MM.2022.3152788

[41] Exascale Computing Project. 2018. ECP Proxy Apps Suite. https://proxyapps.exascaleproject.org/ecp-proxy-apps-suite/

[42] Wilfred Gomes, Sanjeev Khushu, Doug B. Ingerly, Patrick N. Stover, Nasirul I. Chowdhury, Frank O'Mahony, Ajay Balankutty, Noam Dolev, Martin G. Dixon, Lei Jiang, Surya Prekke, Biswajit Patra, Pavel V. Rott, and Rajesh Kumar. 2020. 8.1 Lakefield and mobility compute: A 3D stacked 10nm and 22FFL hybrid processor system in 12×12mm^2, 1mm package-on-package. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. IEEE Press, San Francisco, CA, USA, 144–146. https://doi.org/10.1109/ISSCC19947.2020.9062957

[43] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir. 1983. The NYU ultracomputer—designing an MIMD shared memory parallel computer. *IEEE Trans. Comput.* C-32, 2 (1983), 175–189. https://doi.org/10.1109/TC.1983.1676201

[44] A. Arun Goud, Rangharajan Venkatesan, Anand Raghunathan, and Kaushik Roy. 2015. Asymmetric underlapped FinFET based Robust SRAM design at 7nm node. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE'15)*. EDA Consortium, San Jose, CA, USA, 659–664.

[45] Thomas Grass, César Allande, Adrià Armejach, Alejandro Rico, Eduard Ayguadé, Jesus Labarta, Mateo Valero, Marc Casas, and Miquel Moreto. 2016. MUSA: A multi-level simulation approach for next-generation HPC machines. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'16)*. IEEE Press, Salt Lake City, UT, USA, 526–537. https://doi.org/10.1109/SC.2016.44

[46] Yang Guo, Chisachi Kato, and Yoshinobu Yamade. 2006. Basic features of the fluid dynamics simulation software "FrontFlow/Blue". *Seisan Kenkyu* 58, 1 (2006), 11–15. https://doi.org/10.11188/seisankenkyu.58.11

[47] Salman Habib, Vitali Morozov, Nicholas Frontiere, Hal Finkel, Adrian Pope, Katrin Heitmann, Kalyan Kumaran, Venkatram Vishwanath, Tom Peterka, Joe Insley, David Daniel, Patricia Fasel, and Zarija Lukić. 2016. HACC: Extreme scaling and performance across diverse architectures. *Commun. ACM* 60, 1 (Dec. 2016), 97–104. https://doi.org/10.1145/3015569

[48] Fazal Hameed, Asif Ali Khan, and Jeronimo Castrillon. 2021. Improving the performance of block-based DRAM caches via tag-data decoupling. *IEEE Trans. Comput.* 70, 11 (2021), 1914–1927. https://doi.org/10.1109/TC.2020.3029615

[49] Nicole Hemsoth. 2018. A Rogues Gallery of Post-Moore's Law Options. https://www.nextplatform.com/2018/08/27/a-rogues-gallery-of-post-moores-law-options/

[50] Michael A. Heroux, Douglas W. Doerfler, Paul S. Crozier, James M. Willenbring, H. Carter Edwards, Alan Williams, Mahesh Rajan, Eric R. Keiter, Heidi K. Thornquist, and Robert W. Numrich. 2009. *Improving Performance via Mini-applications*. Technical Report SAND2009-5574. Sandia National Laboratories.

[51] Joel Hruska. 2012. The Death of CPU Scaling: From One Core to Many – and Why We're Still Stuck. https://www.extremetech.com/computing/116561-the-death-of-cpu-scaling-from-one-core-to-many-and-why-were-still-stuck

[52] Xing Hu, Dylan C. Stow, and Yuan Xie. 2018. Die stacking is happening. *IEEE Micro* 38, 1 (2018), 22–28. https://doi.org/10.1109/MM.2018.011441561

[53] IEEE IRDS™. 2021. *International Roadmap for Devices and Systems (IRDS™) 2021 Edition – Executive Summary*. IEEE IRDS™ Roadmap. IEEE. 64 pages. https://irds.ieee.org/images/files/pdf/2021/2021IRDS_ES.pdf

[54] IEEE IRDS™. 2021. *International Roadmap for Devices and Systems (IRDS™) 2021 Edition – Systems and Architectures*. IEEE IRDS™ Roadmap. IEEE. 23 pages. https://irds.ieee.org/images/files/pdf/2021/2021IRDS_SA.pdf

[55] Intel Corporation. 2012. Intel® Architecture Code Analyzer – User's Guide. https://www.intel.com/content/dam/develop/external/us/en/documents/intel-architecture-code-analyzer-2-0-users-guide-157548.pdf

[56] Intel Corporation. 2020. Dynamic Control- Flow Graph (DCFG) and DCFG-Trace Format Specifications – For Format Version 1.00. https://www.intel.com/content/dam/develop/external/us/en/documents/dcfg-format-548994.pdf

[57] Intel Corporation. 2021. Intel® Software Development Emulator. https://www.intel.com/content/www/us/en/developer/articles/tool/software-development-emulator.html

[58] Ravi R. Iyer, Vivek De, Ramesh Illikkal, David A. Koufaty, Bhushan Chitlur, Andrew Herdrich, Muhammad M. Khellah, Fatih Hamzaoglu, and Eric Karl. 2021. Advances in microprocessor cache architectures over the last 25 years. *IEEE Micro* 41, 6 (2021), 78–88. https://doi.org/10.1109/MM.2021.3114903

[59] Christian Jacobi. 2021. Real-time AI for enterprise workloads: The IBM Telum processor. In *2021 IEEE Hot Chips 33 Symposium (HCS)*. IEEE Computer Society, Palo Alto, CA, USA, 22. https://doi.org/10.1109/HCS52781.2021.9567422 https://hc33.hotchips.org/assets/program/conference/day1/HC2021.C1.3IBMCristianJacobiFinal.pdf

[60] H. Jin, M. Frumkin, and J. Yan. 1999. *The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance*. Technical Report NAS-99-011. NASA Ames Research Center. 26 pages. https://www.nas.nasa.gov/assets/pdf/techreports/1999/nas-99-011.pdf

[61] Jaewoon Jung, Takaharu Mori, Chigusa Kobayashi, Yasuhiro Matsunaga, Takao Yoda, Michael Feig, and Yuji Sugita. 2015. GENESIS: A hybrid-parallel and multi-scale molecular dynamics simulator with enhanced sampling algorithms for biomolecular and cellular simulations. *WIREs Computational Molecular Science* 5, 4 (2015), 310–323. https://doi.org/10.1002/wcms.1220

[62] Yuetsu Kodama, Tetsuya Odajima, Akira Asato, and Mitsuhisa Sato. 2020. Accuracy improvement of memory system simulation for modern shared memory processor. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region (HPCAsia2020)*. Association for Computing Machinery, New York, NY, USA, 142–149. https://doi.org/10.1145/3368474.3368483

[63] Kunal Korgaonkar, Ishwar Bhati, Huichu Liu, Jayesh Gaur, Sasikanth Manipatruni, Sreenivas Subramoney, Tanay Karnik, Steven Swanson, Ian Young, and Hong Wang. 2018. Density tradeoffs of non-volatile memory as a replacement for SRAM based last level cache. In *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA'18)*. IEEE Press, Los Angeles, CA, USA, 315–327. https://doi.org/10.1109/ISCA.2018.00035

[64] John H. Lau. 2021. 3D IC integration and 3D IC packaging. In *Semiconductor Advanced Packaging*. Springer, Singapore, 343–378. https://doi.org/10.1007/978-981-16-1376-0_7

[65] Jan Laukemann, Julian Hammer, Johannes Hofmann, Georg Hager, and Gerhard Wellein. 2018. Automated instruction stream throughput prediction for Intel and AMD microarchitectures. In *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE Press, Dallas, TX, USA, 121–131. https://doi.org/10.1109/PMBS.2018.8641578

[66] LLVM Project. 2022. Llvm-Mca - LLVM Machine Code Analyzer. https://llvm.org/docs/CommandGuide/llvm-mca.html

[67] Gabriel H. Loh and Mark D. Hill. 2011. Efficiently enabling conventional block sizes for very large die-stacked DRAM caches. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44)*. Association for Computing Machinery, New York, NY, USA, 454–464. https://doi.org/10.1145/2155620.2155673

[68] Gabriel H. Loh, Yuan Xie, and Bryan Black. 2007. Processor design in 3D die-stacking technologies. *IEEE Micro* 27, 3 (May 2007), 31–48. https://doi.org/10.1109/MM.2007.59

[69] Hatem Ltaief, Jesse Cranney, Damien Gratadour, Yuxi Hong, Laurent Gatineau, and David E. Keyes. 2021. Meeting the real-time challenges of ground-based telescopes using low-rank matrix computations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'21)*. ACM, New York, NY, USA, 29:1–29:16. https://doi.org/10.1145/3458817.3476225

[70] J. D. McCalpin. 1995. Memory bandwidth and machine balance in current high performance computers. *IEEE Technical Committee on Computer Architecture (TCCA) Newsletter* 2, 19–25 (Dec. 1995), 1–7.

[71] Sally A. McKee. 2004. Reflections on the memory wall. In *Proceedings of the 1st Conference on Computing Frontiers (CF'04)*. Association for Computing Machinery, New York, NY, USA, 162. https://doi.org/10.1145/977091.977115

[72] Charith Mendis, Alex Renda, Dr. Saman Amarasinghe, and Michael Carbin. 2019. Ithemal: Accurate, portable and fast basic block throughput estimation using deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, Long Beach, California, USA, 4505–4515. https://proceedings.mlr.press/v97/mendis19a.html

[73] Takahiro Misawa, Satoshi Morita, Kazuyoshi Yoshimi, Mitsuaki Kawamura, Yuichi Motoyama, Kota Ido, Takahiro Ohgoe, Masatoshi Imada, and Takeo Kato. 2018. mVMC–open-source software for many-variable variational Monte Carlo method. *Computer Physics Communications* 235, Feb. 2019 (2018), 447–462. https://doi.org/10.1016/j.cpc.2018.08.014

[74] Sparsh Mittal and Jeffrey S. Vetter. 2016. A survey of techniques for architecting DRAM caches. *IEEE Transactions on Parallel and Distributed Systems* 27, 6 (June 2016), 1852–1863. https://doi.org/10.1109/TPDS.2015.2461155

[75] Jamaludin Mohd-Yusof, Sriram Swaminarayan, and Timothy C. Germann. 2013. *Co-Design for Molecular Dynamics: An Exascale Proxy Application.* Technical Report LA-UR 13-20839. Los Alamos National Laboratory. http://www.lanl.gov/orgs/adtsc/publications/science_highlights_2013/docs/Pg88_89.pdf

[76] Gordon E. Moore. 1975. Progress in digital integrated electronics. *International Electron Devices Meeting, IEEE* 21 (1975), 11–13.

[77] Timothy P. Morgan. 2022. "Milan-X" 3D Vertical Cache Yields Epyc HPC Bang for the Buck Boost. https://www.nextplatform.com/2022/03/21/milan-x-3d-vertical-cache-yields-epyc-hpc-bang-for-the-buck-boost/

[78] Takahito Nakajima, Michio Katouda, Muneaki Kamiya, and Yutaka Nakatsuka. 2014. NTChem: A high-performance software package for quantum molecular simulation. *International Journal of Quantum Chemistry* 115, 5 (Dec. 2014), 349–359. https://doi.org/10.1002/qua.24860

[79] John Nickolls and William J. Dally. 2010. The GPU computing era. *IEEE Micro* 30, 2 (2010), 56–69. https://doi.org/10.1109/MM.2010.41

[80] Anant Vithal Nori, Jayesh Gaur, Siddharth Rai, Sreenivas Subramoney, and Hong Wang. 2018. Criticality aware tiered cache hierarchy: A fundamental relook at multi-level cache hierarchies. In *Proceedings of the 45th Annual*

International Symposium on Computer Architecture (ISCA'18). IEEE Press, Los Angeles, CA, USA, 96–109. https://doi.org/10.1109/ISCA.2018.00019

[81] NVIDIA Corporation. 2022. NVIDIA H100 Tensor Core GPU. https://www.nvidia.com/en-us/data-center/h100/

[82] Ryohei Okazaki, Takekazu Tabata, Sota Sakashita, Kenichi Kitamura, Noriko Takagi, Hideki Sakata, Takeshi Ishibashi, Takeo Nakamura, and Yuichiro Ajima. 2020. *Supercomputer Fugaku CPU A64FX Realizing High Performance, High-Density Packaging, and Low Power Consumption*. Fujitsu Technical Review. Fujitsu Limited. 9 pages. https://www.fujitsu.com/global/documents/about/resources/publications/technicalreview/2020-03/article03.pdf

[83] Geraldo F. Oliveira, Juan Gómez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan Fernandez, Mohammad Sadrosadati, and Onur Mutlu. 2021. DAMOV: A new methodology and benchmark suite for evaluating data movement bottlenecks. *IEEE Access* 9 (2021), 134457–134502. https://doi.org/10.1109/ACCESS.2021.3110993

[84] Kenji Ono, Masako Iwata, Tsuyoshi Tamaki, Yasuhiro Kawashima, Kei Akasaka, Soichiro Suzuki, Junya Onishi, Ken Uzawa, Kazuhiro Hamaguchi, Yohei Miyazaki, and Masashi Imano. 2016. FFV-C Package. http://avr-aics-riken.github.io/ffvc_package/

[85] Zvi Or-Bach. 2017. A 1,000x improvement in computer systems by bridging the processor-memory gap. In *2017 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*. IEEE Press, Burlingame, CA, USA, 1–4. https://doi.org/10.1109/S3S.2017.8309202

[86] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. 2008. GPU computing. *Proc. IEEE* 96, 5 (2008), 879–899. https://doi.org/10.1109/JPROC.2008.917757

[87] Jongsoo Park, Mikhail Smelyanskiy, Ulrike Meier Yang, Dheevatsa Mudigere, and Pradeep Dubey. 2015. High-performance algebraic multigrid solver optimized for multi-core based distributed parallel systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15)*. ACM, Austin, TX, USA, 54:1–54:12. https://doi.org/10.1145/2807591.2807603

[88] N. A. Petersson and B. Sjögreen. 2017. *User's Guide to SW4, Version 2.0*. Technical Report LLNL-SM-741439. Lawrence Livermore National Laboratory.

[89] Artur Podobas, Kentaro Sano, and Satoshi Matsuoka. 2020. A survey on coarse-grained reconfigurable architectures from a performance perspective. *IEEE Access* 8 (July 2020). https://doi.org/10.1109/ACCESS.2020.3012084

[90] Louis-Noel Pouchet and Mark Taylor. 2016. PolyBench/C 4.2.1 (Beta). https://sourceforge.net/projects/polybench/

[91] Alex Renda, Yishen Chen, Charith Mendis, and Michael Carbin. 2020. DiffTune: Optimizing CPU simulator parameters with learned differentiable surrogates. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE Press, Athens, Greece, 442–455. https://doi.org/10.1109/MICRO50266.2020.00045

[92] RIKEN AICS. 2015. Fiber Miniapp Suite. https://fiber-miniapp.github.io/

[93] RIKEN Center for Computational Science. 2021. The Kernel Codes from Priority Issue Target Applications. https://github.com/RIKEN-RCCS/fs2020-tapp-kernels

[94] RIKEN-RCCS. 2020. Riken_simulator. https://github.com/RIKEN-RCCS/riken_simulator

[95] Arun Rodrigues, Elliot Cooper-Balis, Keren Bergman, Kurt Ferreira, David Bunde, and K. Scott Hemmert. 2012. Improvements to the structural simulation toolkit. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS'12)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, Belgium, 190–195.

[96] Mitsuhisa Sato, Yutaka Ishikawa, Hirofumi Tomita, Yuetsu Kodama, Tetsuya Odajima, Miwako Tsuji, Hisashi Yashiro, Masaki Aoki, Naoyuki Shida, Ikuo Miyoshi, Kouichi Hirai, Atsushi Furuya, Akira Asato, Kuniki Morita, and Toshiyuki Shimizu. 2020. Co-design for A64FX manycore processor and "Fugaku". In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'20)*. IEEE Press, Atlanta, GA, USA, 1–15.

[97] Kota Shiba, Mitsuji Okada, Atsutake Kosuge, Mototsugu Hamada, and Tadahiro Kuroda. 2022. A 7-Nm FinFET 1.2-TB/s/Mm^2 3D-Stacked SRAM module with 0.7-pJ/b inductive coupling interface using over-SRAM coil and Manchester-encoded synchronous transceiver. *IEEE Journal of Solid-State Circuits* (2022), 1–12. https://doi.org/10.1109/JSSC.2022.3224421

[98] Kota Shiba, Tatsuo Omori, Kodai Ueyoshi, Shinya Takamaeda-Yamazaki, Masato Motomura, Mototsugu Hamada, and Tadahiro Kuroda. 2021. A 96-MB 3D-Stacked SRAM using inductive coupling with 0.4-V transmitter, termination scheme and 12:1 SerDes in 40-Nm CMOS. *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)* 68, 2 (Feb. 2021), 692–703. https://doi.org/10.1109/TCSI.2020.3037892

[99] Anton Shilov. 2022. TSMC Roadmap Update: N3E in 2024, N2 in 2026, Major Changes Incoming. https://www.anandtech.com/show/17356/tsmc-roadmap-update-n3e-in-2024-n2-in-2026-major-changes-incoming

[100] Max M. Shulaker, Tony F. Wu, Mohamed M. Sabry, Hai Wei, H.-S. Philip Wong, and Subhasish Mitra. 2015. Monolithic 3D integration: A path from concept to reality. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE'15)*. EDA Consortium, San Jose, CA, USA, 1197–1202.

[101] Hugh Sorby. 2017. MPI Stub. https://github.com/hsorby/mpistub

[102] Standard Performance Evaluation Corporation. 2020. SPEC's Benchmarks. https://www.spec.org/benchmarks.html

[103] Nigel Stephens, Stuart Biles, Matthias Boettcher, Jacob Eapen, Mbou Eyole, Giacomo Gabrielli, Matt Horsnell, Grigorios Magklis, Alejandro Martinez, Nathanael Premillieu, Alastair Reid, Alejandro Rico, and Paul Walker. 2017. The ARM scalable vector extension. *IEEE Micro* 37, 02 (March 2017), 26–39. https://doi.org/10.1109/MM.2017.35

[104] Erich Strohmaier, Jack Dongarra, Horst Simon, and Martin Meuer. 2021. TOP500. http://www.top500.org/

[105] David Suggs, Mahesh Subramony, and Dan Bouvier. 2020. The AMD "Zen 2" processor. *IEEE Micro* 40, 2 (2020), 45–52. https://doi.org/10.1109/MM.2020.2974217

[106] Fatemeh Tavakkoli, Siavash Ebrahimi, Shujuan Wang, and Kambiz Vafai. 2016. Analysis of critical thermal issues in 3D integrated circuits. *International Journal of Heat and Mass Transfer* 97 (2016), 337–352. https://doi.org/10.1016/j.ijheatmasstransfer.2016.02.010

[107] Thomas N. Theis and H.-S. Philip Wong. 2017. The end of Moore's law: A new beginning for information technology. *Computing in Science Engineering* 19, 2 (2017), 41–50. https://doi.org/10.1109/MCSE.2017.29

[108] Hirofumi Tomita and Masaki Satoh. 2004. A new dynamical framework of nonhydrostatic global model using the icosahedral grid. *Fluid Dynamics Research* 34, 6 (2004), 357–400. http://stacks.iop.org/1873-7005/34/i=6/a=A03

[109] John R. Tramm, Andrew R. Siegel, Tanzima Islam, and Martin Schulz. 2014. XSBench - The development and verification of a performance abstraction for Monte Carlo reactor analysis. In *PHYSOR 2014 - The Role of Reactor Physics toward a Sustainable Future*. JAEA, Kyoto, 1–13. https://doi.org/10.11484/jaea-conf-2014-003

[110] Rob F. Van der Wijngaart. 2002. *The NAS Parallel Benchmarks 2.4*. Technical Report NAS-02-007. NASA Ames Research Center. 8 pages. https://www.nas.nasa.gov/assets/pdf/techreports/2002/nas-02-007.pdf

[111] Aravind Vasudevan, Andrew Anderson, and David Gregg. 2017. Parallel multi channel convolution using general matrix multiplication. In *2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE Press, Seattle, WA, USA, 19–24. https://doi.org/10.1109/ASAP.2017.7995254

[112] Jeffery S. Vetter, Erik P. DeBenedictis, and Thomas M. Conte. 2017. Architectures for the Post-Moore era. *IEEE Micro* 37, 04 (July 2017), 6–8. https://doi.org/10.1109/MM.2017.3211127

[113] Gwendolyn Voskuilen, Arun F. Rodrigues, and Simon D. Hammond. 2016. Analyzing allocation behavior for multi-level memory. In *Proceedings of the Second International Symposium on Memory Systems (MEMSYS'16)*. Association for Computing Machinery, New York, NY, USA, 204–207. https://doi.org/10.1145/2989081.2989116

[114] Shaoxi Wang, Yue Yin, Chenxia Hu, and Pouya Rezai. 2018. 3D integrated circuit cooling with microfluidics. *Micromachines* 9, 6 (2018), 1–14. https://doi.org/10.3390/mi9060287

[115] James Warnock, Brian Curran, John Badar, Gregory Fredeman, Donald Plass, Yuen Chan, Sean Carey, Gerard Salem, Friedrich Schroeder, Frank Malgioglio, Guenter Mayer, Christopher Berry, Michael Wood, Yiu-Hing Chan, Mark Mayo, John Isakson, Charudhattan Nagarajan, Tobias Werner, Leon Sigal, Ricardo Nigaglioni, Mark Cichanowski, Jeffrey Zitz, Matthew Ziegler, Tim Bronson, Gerald Strevig, Daniel Dreps, Ruchir Puri, Douglas Malone, Dieter Wendel, Pak-Kin Mak, and Michael Blake. 2015. 4.1 22nm Next-generation IBM system z microprocessor. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*. IEEE Press, San Francisco, CA, USA, 1–3. https://doi.org/10.1109/ISSCC.2015.7062930

[116] M. M. Wolf, J. W. Berry, and D. T. Stark. 2015. A task-based linear algebra building blocks approach for scalable graph analytics. In *2015 IEEE High Performance Extreme Computing Conference (HPEC'15)*. IEEE Press, Waltham, MA, USA, 1–6. https://doi.org/10.1109/HPEC.2015.7322450

[117] Shuji Yamamura, Yasunobu Akizuki, Hideyuki Sekiguchi, Takumi Maruyama, Tsutomu Sano, Hiroyuki Miyazaki, and Toshio Yoshida. 2022. A64FX: 52-core processor designed for the 442PetaFLOPS Supercomputer Fugaku. In *IEEE International Solid-State Circuits Conference, ISSCC 2022, San Francisco, CA, USA, February 20–26, 2022*. IEEE, San Francisco, CA, USA, 352–354. https://doi.org/10.1109/ISSCC42614.2022.9731627

[118] Toshio Yoshida. 2018. Fujitsu high performance CPU for the Post-K computer. In *2018 IEEE Hot Chips 30 Symposium (HCS)*. IEEE Computer Society, California, USA, 22. http://www.fujitsu.com/jp/Images/20180821hotchips30.pdf

[119] Vinson Young, Chiachen Chou, Aamer Jaleel, and Moinuddin Qureshi. 2018. ACCORD: Enabling associativity for gigascale DRAM caches by coordinating way-install and way-prediction. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA'18)*. IEEE, Los Angeles, CA, USA, 328–339. https://doi.org/10.1109/ISCA.2018.00036

[120] Yuang Zhang, Li Li, Zhonghai Lu, Axel Jantsch, Minglun Gao, Hongbing Pan, and Feng Han. 2014. A survey of memory architecture for 3D chip multi-processors. *Microprocessors and Microsystems* 38, 5 (2014), 415–430. https://doi.org/10.1016/j.micpro.2014.03.007

Received 21 December 2022; revised 5 October 2023; accepted 13 October 2023