Serverless? RISC more!

Roberto Starc
Systems Group, D-INFK, ETH Zurich
Zürich, Switzerland

Tom Kuchler
Systems Group, D-INFK, ETH Zurich
Zürich, Switzerland

Michael Giardino[*]
Huawei Technologies, Computing Systems Group
Zürich, Switzerland

Ana Klimovic
Systems Group, D-INFK, ETH Zurich
Zürich, Switzerland

ABSTRACT

The growth of serverless computing has led to a widespread reexamination of the cloud software upon which it is based. In parallel, the flattening of single core performance has led to a resurgence of interest in manycore systems, trading absolute performance for system throughput, an appropriate match for the serverless paradigm. However, the combination of deep cloud system software stacks and slow hardware simulation techniques has limited the exploration of serverless-native CPUs. We argue that the RISC-V ecosystem offers an opportunity to tackle the intersection of these topics. We present an exploratory comparison of several RISC-V SoC configurations and commercial products running serverless workloads. We find that the RISC-V cores offer reasonable performance, but more importantly provide researchers the ability to run more realistic software workloads. This allows for meaningful exploration of the interactions between system software, serverless workloads, and specialized hardware.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; *Architectures*; **Reduced instruction set computing**; • **Hardware** → *Best practices for EDA.*

KEYWORDS

serverless,risc-v,system software,fpga,synthesis,function as a service,faas,cloud software,simulation,microarchitecture,co-design

ACM Reference Format:
Roberto Starc, Tom Kuchler, Michael Giardino, and Ana Klimovic. 2024. Serverless? RISC more!. In *The 2nd Workshop on SErverless Systems, Applications and MEthodologies (SESAME '24), April 22, 2024, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3642977.3652095

1 INTRODUCTION

Serverless computing, also known as Function-as-a-Service (FaaS), is increasingly being adopted in many application domains due to its ease-of-use, high elasticity, and fine-grained billing benefits [66, 25, 26, 39, 19, 20, 59]. The rise of FaaS has prompted researchers and developers to rethink nearly every aspect of the cloud computing software stack, from the high-level programming model to the low-level system execution implementations, including the design of primitives for secure task isolation, networking, and storage. Serverless functions' unique characteristics (i.e. stateless, short-lived, and sporadically invoked functions with small resource footprints) have motivated new systems software to quickly boot and execute functions with secure isolation [2, 30], densely pack functions per machine [49, 64], and manage resources efficiently at cluster scale [84, 50, 42, 62].

As serverless computing's share of the cloud is growing [66, 20], it is important to not only consider the implications for cloud software stacks, but also for cloud hardware. While FaaS was originally imagined as a technique for using spare capacity in otherwise used machines, the reality is that cloud providers have dedicated clusters of standard servers to serve function requests. These FaaS clusters must aggressively multiplex hundreds or thousands of incoming and running functions in order to obtain acceptable throughput.

However, initial studies have found that modern CPU features often provide limited benefits for short-lived functions [68, 65]. For example, the effective (but complex) branch predictors found in server-class CPUs take time to warm up for maximum performance, making them less effective for short-running tasks [68]. Thus there is renewed interest [70, 72, 54] in exploring manycore CPU architectures of the past [35, 57, 52, 67]. These systems have CPUs with simpler microarchitectures, trading single-threaded performance for greater computing density and overall throughput.

Unfortunately, the performance implications of new CPU architecture features for serverless computing have been explored predominantly in slow and/or simplified simulators. This approach, discussed further in Section 2.3, limits researchers' ability to run the deep software stacks that are used in modern serverless systems. High-level programming language runtimes with memory, file, and network accesses are significantly more difficult to explore in simulators. This difficulty is exacerbated when one includes virtualization, containerization, orchestration, and scheduling present in serverless systems. Put another way, unlike hardware design space exploration for compute intensive workloads that results in application specific accelerators (e.g. a TPU), FaaS hardware exploration must include the system software (hypervisor, kernel, containers), disk I/O, and networking (TCP/IP, RPC) in addition to a broad swath of diverse workloads. To fully explore the promise of co-designed, application-specific hardware [34], we need tools that work at the system level.

[*]work partially done while at ETH Zurich

SESAME '24, April 22, 2024, Athens, Greece
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0545-8/24/04
https://doi.org/10.1145/3642977.3652095

In this paper, we argue that systems researchers should look to the RISC-V ecosystem to explore the design of future, cloud- and serverless-native CPU architectures. Modern out-of-order RISC-V cores [13, 86, 87, 83] are quite capable and this work demonstrates that they can offer reasonable performance for common serverless workloads. These cores are developed in frameworks which allow for agile, parameterized development of pipelines, caches, branch predictors, prefetchers, and even coherence protocols [6, 83, 81]. The large community (both academic and commercial) built around RISC-V has lead to a broad improvement of performance and features including hardware threads, vectorization instructions, virtualization support, and more. Moreover, these designs can be evaluated in a cycle-accurate way while running full-stack workloads at interactive speeds (25–100 MHz).

Linux RISC-V supports a full range of runtimes and applications, providing accurate experiments using full systems including memory hierarchies and networking [43]. The synthesized designs can be used to accurately estimate power consumption and silicon area [51]. These same designs can even be taped-out SoCs for real-world prototyping [83, 86, 10]. Independently of the role commercial RISC-V silicon ends up playing in datacenters—if any at all—we argue that using more accurate CPU models and SoC infrastructure will lead to more meaningful and impactful architecture and systems research.

Section 2 discusses the characteristics of serverless workloads, previous cloud-native systems, and current techniques for hardware simulation. We present the opportunities available for hardware/software co-design in Section 3. In Section 4 we demonstrate the feasibility of this ecosystem for research by evaluating serverless workflows on parameterized RISC-V SoCs. We conclude in Section 5 with a discussion of the missing pieces for a broader adoption of RISC-V as a research platform, and perhaps as a serverless-native CPU.

2 BACKGROUND

Much like HPC or ML workloads, serverless computing possesses several unique characteristics (§ 2.1), suggesting that systems researchers should look beyond traditional server processors and explore a broader hardware space. We give a brief overview of other attempts at cloud-native CPUs (§ 2.2) and discuss limitations of current hardware simulation techniques (§ 2.3).

2.1 FaaS Characteristics

Data about *what* functions users are running on FaaS offerings is limited, however we do have some data on *how* functions are running. The clearest sources are industry serverless function traces [69, 75, 40]. These reveal that functions running on commercial FaaS offerings show markedly different behavior than traditional datacenter applications. These functions are extremely *short-lived* with median execution times reported anywhere from seconds [75], less than a second [69], to as low as 60 ms [20]. Functions also demonstrate *highly variable invocation patterns* [21, 40], both in periods of large peak demand followed a low trough (up to 500× [75] difference between peak and trough). The time between invocations varies as well, with medians measured from seconds [75] to hours [69]. As Wang *et al.* point out [75], almost half the functions request a new instance to be cold started every second or less. Shahrad *et al.* [69] are not as explicit about the frequency of cold starts but they highlight that 45% of all applications are invoked less than once per hour, which strongly indicates that these functions are highly likely to experience a cold start. FaaS applications have *small resource footprints*. 90% of functions use less than 400 MB and the median application uses only 170 MB of memory [69]. These properties stand in contrast to more traditional cloud applications, which usually run for a long time on a fixed amount of resources.

Serverless CPU Design Exploration. The aforementioned FaaS workload properties suggest that serverless functions do not benefit as much from many performance optimizations built into modern CPUs as long-running applications do [68, 74, 65]. Microarchitectural state that needs to be warmed up, e.g., branch predictors and caches, is not as effective as it is for traditional applications, motivating exploration into faster training predictors and smaller LLC-to-core ratios [68]. The significant data movement required to pull function snapshots has lead to software mechanisms for prefetching data [74], which could be augmented by hardware extensions. To handle the challenge of ephemeral data in serverless scenarios Wang *et al.* propose Memento [77], a set of architectural mechanisms to allocate and free directly in cache, and effectively manage a memory pool. Due to the high degree of function interleaving on a system, instruction cache misses are a major source of slowdown, prompting proposals for hardware mechanisms to save instruction state [65].

The large caches, heavy-duty predictors, aggressive reordering, and specialized instruction set extensions all take up a large amount of silicon space and energy budget, especially for short-lived functions that often spend a significant amount of time blocking for data. This growing body of work indicates that existing server-class processors are not necessarily well-matched to short-running, independent, bursty functions.

2.2 Cloud Hardware Architectures

The rise of new hardware architectures. As the single-core CPU performance gains decreased over the past decade, other considerations besides absolute per-core performance have come into play [73]. Factors such as maximum power consumption, energy efficiency, and core density are playing an increasing role. This has lead to the prevalence of specialized hardware such as GPUs, TPUs [41], VCUs [58] and FPGAs, as well as core-dense, energy-efficient SoCs. Several recent and upcoming systems favor simple cores to achieve high density and compute throughput: ARM-based cloud SoCs [4, 5, 7], manycore supercomputing platforms [27], and recent announcements of AMD Zen4 [1] and Intel Sierra Forest [36]. There is also ongoing work in commercializing RISC-V-based cloud chips such as the Xuantie 910 [14] and Ventana Veyron [12]. This trend suggests that cloud providers justify significant hardware engineering costs to improve performance-per-watt and compute density.

Compute density optimized CPUs. The desire to trade-off single-threaded performance for compute density in the cloud has been explored by both academic and industrial researchers. "Scale-out" processors [52] for the cloud attempt to maximize overall throughput of a given size die, with the goal of greater *performance density*. Early commercial attempts to increase compute density by using simpler cores include Intel's Single-chip Cloud Computer [35], Tilera [57], and Cavium ThunderX-1 [15]. The SPARC M7 [45] attempted to increase density via symmetric multithreading (SMT), offering up to 256 threads per socket. These products were primarily aimed at monolithic multi-threaded applications, which, while offering significant parallelism, are not as short-running and independent as serverless functions. Building upon this work into early so-called "cloud-native" CPUs [52, 35, 57], a new generation of manycore systems has been developed [54, 10]. Other work aims to maximize compute density while adding cloud- and serverless-specific features such as RPC acceleration and hardware support for context switching [72].

The evaluation of these scale-out systems has taken many forms over the years, spanning a large number of cloud benchmarks [24, 56, 29]. These workloads are complex and have varying execution characteristics not only between applications but in different phases within the same application. While this newest generation of cloud-native CPUs has not yet been thoroughly evaluated, there are indications that less-powerful, simpler CPUs can offer an attractive price-to-performance ratio [16]. The complexity of communication, scheduling, and orchestration is an integral part of cloud workloads, and work has shown that the performance of interdependent services have far-reaching effects on the system as a whole [15, 29]. Even if the functions themselves are fairly short-lived, the complex interactions and depth of the software stack make simulation and modeling challenging.

2.3 Hardware Simulation

Given this rapidly changing landscape, systems programmers need to explore hardware architectures for specific workloads. However, simulating even a simple CPU is a complex process, and there are dozens of simulators for all aspects of computer systems. There exist several commonly used simulators for CPU architectures (gem5 [53], ZSim [63], Sniper [11]), memory simulation (e.g. DRAMSim [76, 61, 48]) and cache simulation (e.g. CMP$im [38]). While detailing the broad array of methods for computer architecture simulation is beyond the scope of this paper (or indeed a book!), we will briefly discuss its current state and its challenges. For more details, Akram *et al.* [3] offer an overview of several common simulators, assessing their accuracy and performance.

As with most techniques of modeling, there is a trade-off between *model accuracy*, *simulation speed*, and *cost*. A functional simulation is much faster to run and update than a cycle-accurate (CA) simulator, while a CA model can give much more accurate results. For most simulators, speed is often between one thousand and one million instructions per second (1 KIPS–1 MIPS). For comparison, our simple image processing workflow executes about three billion instructions in $\approx$0.5 s. For a simulation running at the high end (1 MIPS), this is nearly an hour of simulation time. At the low end (1 KIPS), as is often the case in complex cycle-accurate models, simulating this function would require over a month. While simulation techniques are continuously improving [32, 31], we argue that for system-level analysis, these models are still too slow, low-fidelity, and don't map directly into hardware.

On the other end of the spectrum are register-transfer level (RTL) models of CPUs which, historically, have been chip designers' most closely guarded secrets. With the advent of the RISC-V ISA, and open, agile toolchains for generating RTL from high-level descriptions, researchers can design, simulate, and synthesize for FPGAs, and even tape out SoCs. With FPGAs, RTL models can be synthesized and simulated on dedicated simulation engines [22], or run in a cluster using FireSim [43]. FPGA-accelerated simulation offers interactive speeds (up to 100 MHz), allowing for full-system simulations which map directly to real systems. Until now, this was not possible for almost anyone, requiring resources only found inside major chip designers.

3 CO-DESIGN OPPORTUNITIES

The ability to run full system stacks on synthesizable hardware has several implications for systems researchers. To evaluate the performance effects of hardware features (e.g. vector instructions), researchers use a set of representative benchmarks (e.g. SPEC or PARSEC). Because these traditional workloads are highly optimized and computationally intense, monolithic applications, one can extract meaningful traces to feed into architectural simulators. However, the cloud is built upon deep software stacks consisting of some combination of user code, runtimes (including JIT), containers, virtual machines, operating systems, and hypervisors. Therefore, gaining meaningful insight into the impact of microarchitectural features on cloud workloads requires a much more complex simulation. The interplay between application code, guest kernel, hypervisor, and the hardware eliminates the possibility of obtaining realistic results from simplified simulations (e.g. *gem5* syscall emulation mode).

In fact, much of the fundamental "cloud-ness" of a workload stems from the fact that it is virtualized (or containerized) than from the actual computation it does. For example, video transcoding and web page templating are both considered realistic "serverless workloads" [18, 44], even though transcoding builds on decades of computationally complex, highly optimized, hardware accelerated code and web page templating is implemented using simple Python scripts. Moreover, when examining microservices and serverless workflows, faithful modeling must include the communication between services because of their complex interplay and cascading effects [29, 15]. Therefore, if we want to examine microarchitectural features that accelerate the cloud (e.g. RPC acceleration, virtualization extensions), we need to use the entire stack. Modern techniques for isolation such as CHERI hardware capabilities [79] or enclaves in the context of serverless [46, 88] can be evaluated and expanded upon in RISC-V [80, 89, 47, 23] We argue that one of the best tools available for this type of work, the RISC-V hardware/software ecosystem, is too often ignored by both systems and computer architecture researchers in favor of off-the-shelf hardware or low-fidelity simulations.

Table 1: Per-Core Configurations

Name	Core	ISA	OoO	Issue	L1 Size (I/D)	L2 Size	CoreMark/Mhz
Rocket	Rocket	riscv64	✗	1	16/16 KiB	512 KiB	2.14
SmallBoom	BOOM	riscv64	✓	3	16/16 KiB	512 KiB	2.27
MediumBoom	BOOM	riscv64	✓	4	16/16 KiB	512 KiB	3.76
LargeBoom	BOOM	riscv64	✓	5	32/32 KiB	512 KiB	4.88
MegaBoom	BOOM	riscv64	✓	8	32/32 KiB	512 KiB	5.31
StarFive VisionFive2 [71]	JH7110	riscv64	✗	2	32/32 KiB	2 MiB	3.30
Huawei Kunpeng 920 [82]	ARMv8.2	aarch64	✓	4	64/64 KiB	512 KiB	7.20
Intel Xeon Gold 6238T [37]	Cascade Lake	x86-64	✓	8	32/32 KiB	1 MiB	7.54

4 EVALUATION

To demonstrate the feasibility of our integrated systems approach to FaaS architecture exploration, we evaluate the performance of several RISC-V configurations and three different commercial processors (Table 1) microbenchmarks and a set of serverless workflows written in Python (Table 2). Python is often cited as the most commonly used FaaS runtime [20], and has very good support from cloud providers. However, as shown in Table 2, these Python scripts often call specialized libraries (e.g. OpenBLAS, OpenCV) or compiled programs, demonstrating performance beyond the Python runtime. Section 4.1 introduces the benchmarks, Section 4.2 presents the test platforms, and Section 4.3 discusses the results obtained.

Table 2: Benchmarks Evaluated

Benchmark	Type	Language
matmul	Micro	Python (*numpy*)
floater	Micro	Python
linpack	Micro	Python (*numpy*)
image processing	Workflow	Python (*OpenCV*)
text processing	Workflow	Python
compilation	Workflow	Python, GCC, Make

4.1 Benchmarks

There are several serverless benchmarking suites available [44, 85, 18, 74]. However, these often model entire FaaS systems across multiple nodes using containerization/virtualization, and orchestration frameworks. Our initial goal is to determine the feasibility of running serverless workloads on open RISC-V cores, *ergo* we opt to write stand-alone benchmarks inspired by the aforementioned workloads. Additionally, because we want to evaluate the development ecosystem for testing microarchitectural features in relation to FaaS functions, we chose to collect data on the workloads themselves without containerization. Nevertheless, we do have Docker running in our testbed and a deeper exploration of virtualization overheads is ongoing.

Microbenchmarks. The three microbenchmarks are written in Python and are similar to those found in other suites [44]. Matrix multiplication (`matmul`) and `linpack` consist of floating point manipulation of $n \times n$ matrices. `linpack` is a traditional linear algebra benchmark consisting of three matrix manipulations: solving $Ax = b$ for x, inverting matrix A, and computing $A' \times x = b$. Both use *numpy* which in turn calls an optimized C linear algebra library (OpenBLAS) to do the actual computation. The floating point microbenchmark calculates a series of floating point operations [1]. These microbenchmarks are a stand-in for compute-intensive workloads, approaching a lower performance bound for the less complex CPUs. They also demonstrate the value of vectorized and other specialized instructions.

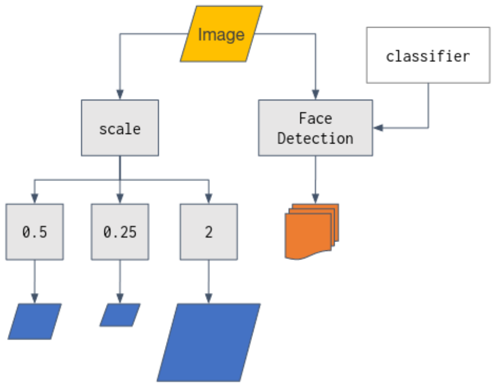

Figure 1: The image processing workflow consists of a simplified dataflow graph for processing photos including, thumbnailing, and face detection. While the graph does suggest a degree of parallelism, we note that in our evaluation, the pipeline is executed as a single thread.

Serverless Workflows. To evaluate workloads that are more representative of those found in serverless suites, we developed three workflows, each consisting of several chained functions. These functions pass data between them, each one taking a single action, creating a data flow graph.

We developed workflows for image processing, text processing, and compilation, similar to those found in many serverless benchmarks [18, 85, 74, 44]. The image processing workflow is a simplified version of cloud processing an uploaded image using OpenCV [55, 21]. Figure 1 shows the workflow as tested. An image is passed

[1] $$\begin{cases} a = \sin x \\ b = \sqrt{a} \\ c = \cos b \\ d = \sqrt{c} \end{cases}$$

as input to the function and various scaling functions are called, making thumbnails of different sizes. The image is also sent to a face detection algorithm which pulls a classifier, and identifies the number of faces found in the image. This workflow can be expanded to include additional processing steps, such as metadata processing, more complex inference algorithms or filtering. The text processing workflow exercises several commonly used cloud functions (MD5 hashing, BZ2 compression, and AES encryption). The compilation workflow aims to replicate the functionality of a Gitlab runner CI pipeline. The hash of a compressed source tarball is checked and then the source is decompressed. The resulting code is configured, compiled, then cleaned up. For this example, we compile Apache v2.4.41. Note that while the benchmarks themselves consist of a single thread, they are running on top of the Python runtime, which in turn is running in a full version Linux which is regularly context switching to handle standard operating system daemons and handles events including long-latency operations such as filesystem I/O.

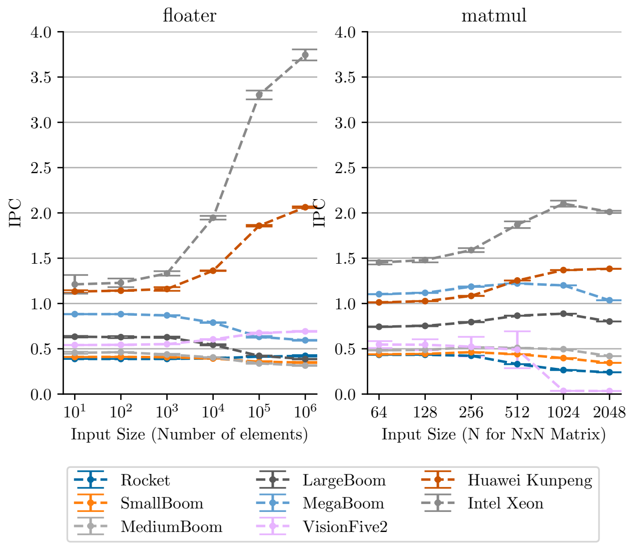

Figure 2: We compare the instructions-per-cycle (IPC) of the Python `matmul` and `floater` microbenchmarks. `linpack` (not shown) shows very similar behavior to the `matmul` benchmark. The input size is on the x-axis and the instructions retired per cycle is found on the y-axis. In all of these benchmarks, the Xeon processor outperforms the ARM and RISC-V cores. For small input sizes, the majority of the program is spent on initializing the Python interpreter and importing modules, which explains why there is little difference for small input sizes.

4.2 Experimental Platform

We evaluated the in-order Rocket core [9] and four configurations of the out-of-order (OoO) BOOM core [87] (Table 1). The BOOM design exposes many parameters, which are adjusted heuristically for each configuration according to the decode width. To evaluate the fidelity of FPGA-based experiments, we also ran the serverless workflows on a StarFive VisionFive2 RISC-V SBC with the JH7110 core [71]. Additionally, to compare the performance of simple RISC-V cores with server-class CPUs, we ran all the benchmarks on a Huawei Kunpeng 920 [82] and an Intel Xeon Gold 6238T [37]. Per-core configurations can be found in Table 1. The BOOM and Rocket cores implement the RV64GC ISA [78], while the JH7110 additionally supports the *B* extension for bit manipulation [60].

The RISC-V platforms run Debian GNU/Linux with the soft cores using kernel 6.2.5 and the StarFive uses 5.15.0. The Kunpeng and Xeon systems run Ubuntu 20.04 LTS. All platforms run Python 3.11. All FPGA-based experiments are conducted on Enzian, a CPU/FPGA research platform [17]. We obtain RTL for configuration using Chipyard [6] and synthesize FPGA bitstreams. We boot a full Debian Linux image using the FPGA DRAM as both a `tmpfs` filesystem and main memory. To conduct our experiments we interact with the system over UART, which we access over `ssh` through the CPU. An advantage of this experimental method is that the CPU architecture can be saved as a bitstream, while the filesystem can be easily changed to expand the evaluation without time-consuming resynthesis of the system.

4.3 Comparative Performance

For these preliminary results we measure the instruction and cycle counts using `perf stat`, using *taskset* to pin our workloads to a core. The measured CoreMark/Mhz [28] scores can be found in Table 1. The instructions-per-cycle (IPC) of microbenchmarks, run on all machines, are shown in Figure 2. We note that the IPC is bounded by decode width, though this is only the upper-most bound.

Instruction Count. We first compare the number of instructions required to run each workflow (Figure 3). The difference in instruction counts indicates the relative state of the RISC-V ISA and compiler.RISC-V is still undergoing changes to its ISA specification, and is expected to improve in this regard over time. As we can see from ARMv8's example, a RISC ISA can achieve significantly higher code density than the evaluated RISC-V variants. Both the Rocket and BOOM cores only support the `RV64GC` variant. Since their release, many extensions to the RISC-V ISA have been ratified [60], (e.g. bit manipulation extension) as well as support for vector instructions. The VisionFive2 results show that support for these more specialized instructions can dramatically increase code density, reducing the total instruction count required for a given workload. While further exploration of these results is necessary to draw definitive conclusions, the large differential between the number of instructions necessary to execute the same workload suggests further exploration of specialized instructions or accelerators for RISC-V and greater optimizations of compilers and runtimes.

IPC. We also compare the IPC across platforms and workloads, shown in Figure 4. In general, the Xeon and Kunpeng significantly outperform the RISC-V cores. Although the Kunpeng has comparable IPC to the MegaBoom, both the ARMv8 and x86 cores require significantly fewer instructions to execute the workload, resulting in fewer cycles. Relative to MegaBoom, Kunpeng and Xeon require

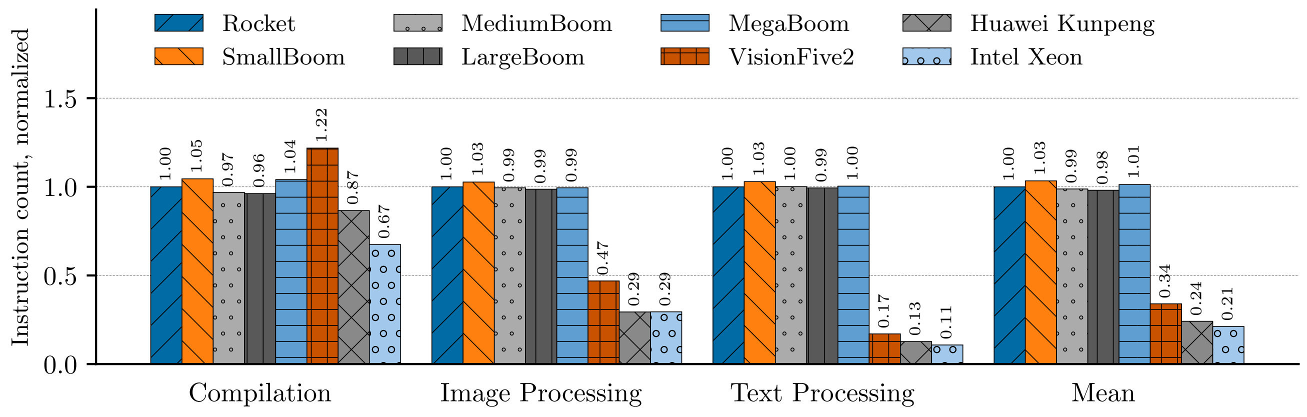

Figure 3: FaaS pipeline relative (to Rocket) instructions. The instruction count difference is most pronounced for the text processing pipeline. This workload consists mostly of compression and encryption operations, which benefit from highly specialized instructions present in the ARM and x86 ISAs. Note the mean presented is the harmonic mean.

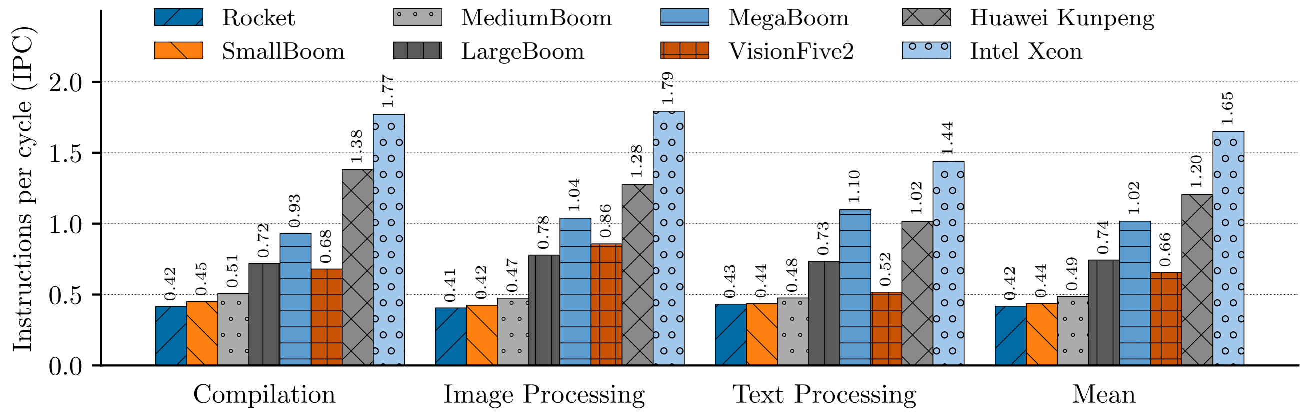

Figure 4: We compare IPC across all of the experimental platforms for each of the FaaS pipelines described in Section 4.1. The bar height indicates the IPC for each evaluated platform. As BOOM core issue width grows, the IPC grows as well, approaching in some cases the commercial processors. Note the presented mean is the harmonic mean.

4.55x and 7.19x fewer cycles on average for the workflows. While wider BOOM cores show a significant improvement over their narrower counterparts, we also note that the BOOM cores achieve much less of their maximum IPC (3 and 4 for the Large- and MegaBoom) compared to Kunpeng and Xeon processors, suggesting that its design is bottlenecked. This difference is exacerbated for the Python microbenchmarks, which represent workloads closer to traditional CPU application domains. Furthermore, the commercial JH7110 core, an in-order dual-issue design, has comparable IPC to the significantly more complex LargeBoom, further indicating that there is much room for improvement in optimizing existing designs.

Conclusion. While the RISC-V cores are overall less performant than their counterparts, the MegaBoom offers an IPC comparable to a modern ARMv8 core for our representative FaaS workloads. While the `RV64GC` variant of RISC-V does require many more instructions for the same computation, our evaluation of the `RV64GCB` variant shows that adding support for more recent RISC-V extensions can significantly improve code density.

5 CONCLUSIONS AND FUTURE WORK

Our preliminary work demonstrates the value of having an open, fast, and accurate technique for analyzing the impact of microarchitectural features on native-scale workloads. These applications make use of a complete system software stack, including system calls and I/O. The cores are parametrically synthesized, generating a varied set of modern SoCs, able to be tested interactively. Putting this together, we believe that the RISC-V ecosystem shows significant maturity and therefore, computer architects and systems researchers should use it as a tool to better understand the microarchitectural implications of cloud workloads.

Our work proceeds in three primary directions. First, we're continuing our parametric exploration of microarchitectural features (e.g. cache size, cache layout, ROB sizes, etc.) to make stronger hypotheses about workload sensitivity to SoC characteristics. Second, we want to examine virtualization and containerization overheads by containerizing our benchmarking suite, and incorporating more complex orchestration using vHive [74] and SeBS [18]. Finally, we aim to use our insight to better evaluate the feasibility of a serverless-native CPU, which trades off single-threaded performance for significant improvements in throughput (via density) and power-efficiency (in instructions per Joule).

While we believe that RISC-V is a well-suited platform for both research and ultimately production systems, there are still several areas in which it can improve. There remains a gap between the single-thread performance of open source RISC-V cores and modern, equivalent ARM or x86 cores. Similarly, toolchain improvements (e.g. optimizing compiler support) could further improve performance. Fortunately, there are several academic and industrial groups steadily improving the performance of RISC-V cores. Hardware threads (*harts*) are part of the RISC-V ISA specification, allowing researchers to reexplore "the valley" between manycores and many threads [33] in the context of modern architectures and workloads. Furthermore, there are several projects to bring state-of-the-art features such as CHERI hardware capabilities [79], persistent memory support [8], and confidential computing/trusted execution extensions [47, 23] to RISC-V.

Given the heterogeneity of both modern hardware and software, we believe that in the era of co-design, cloud researchers cannot afford to ignore hardware design, and hardware designers must use more realistic cloud workloads. While developing in RISC-V has its challenges, the combination of accuracy, performance, and speed of simulation for real workloads makes a strong argument for a prominent space in the systems researcher's toolbox.

REFERENCES

[1] Advanced Micro Devices. 2023. 4th gen AMD EPYC proccessor architecture whitepaper. Advanced Micro Devices. (June 2023). https://www.amd.com/en/campaigns/epyc-9004-architecture.

[2] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. 2020. Firecracker: lightweight virtualization for serverless applications. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, (Feb. 2020), 419–434. ISBN: 978-1-939133-13-7. https://www.usenix.org/conference/nsdi20/presentation/agache.

[3] Ayaz Akram and Lina Sawalha. 2019. A survey of computer architecture simulation techniques and tools. *IEEE Access*, 7, 78120–78145. DOI: 10.1109/ACCESS.2019.2917698.

[4] Alibaba Cloud. 2021. Alibaba cloud unveils new server chips to optimize cloud computing services. Alibaba Cloud. (Oct. 19, 2021). https://www.alibabacloud.com/blog/598159.

[5] Amazon Web Services. 2023. AWS graviton processor. Amazon Web Services. Retrieved July 5, 2023 from https://aws.amazon.com/ec2/graviton/.

[6] Alon Amid, David Biancolin, Abraham Gonzalez, Daniel Grubb, Sagar Karandikar, Harrison Liew, Albert Magyar, Howard Mao, Albert Ou, Nathan Pemberton, Paul Rigge, Colin Schmidt, John Wright, Jerry Zhao, Yakun Sophia Shao, Krste Asanović, and Borivoje Nikolić. 2020. Chipyard: integrated design, simulation, and implementation framework for custom SoCs. *IEEE Micro*, 40, 4, 10–21. DOI: 10.1109/MM.2020.2996616.

[7] Ampere Computing. 2023. Ampereone family product brief. Ampere Computing. (May 2023). https://amperecomputing.com/briefs/ampereone-family-product-brief.

[8] Shashank Anand, Michal Friedman, Michael Giardino, and Gustavo Alonso. 2024. Skip it: take control of your cache! In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS 2024). Association for Computing Machinery, La Jolla, United States, (Apr. 2024), 434–451. ISBN: 9798400703850. DOI: 10.1145/3620665.3640407.

[9] Krste Asanović, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Ben Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David A. Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. 2016. The Rocket Chip Generator. Tech. rep. UCB/EECS-2016-17. EECS Department, University of California, Berkeley, (Apr. 2016). http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html.

[10] Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen, Yanqi Zhou, Alexey Lavrov, Mohammad Shahrad, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff. 2019. Openpiton: an open source hardware platform for your research. *Commun. ACM*, 62, 12, (Nov. 2019), 79–87. DOI: 10.1145/3366343.

[11] Trevor E. Carlson, Wim Heirman, Stijn Eyerman, Ibrahim Hur, and Lieven Eeckhout. 2014. An evaluation of high-level mechanistic core models. *ACM Transactions on Architecture and Code Optimization (TACO)*, Article 5, 23 pages. DOI: 10.1145/2629677.

[12] Aaron Carman. 2022. Server-class RISC-V core unveiled by Ventana at RISC-V summit. All About Circuits. (Dec. 19, 2022). https://www.allaboutcircuits.com/news/server-class-risc-v-core-unveiled-by-ventana-at-risc-v-summit/.

[13] Christopher Celio, David A. Patterson, and Krste Asanović. 2015. The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor. Tech. rep. UCB/EECS-2015-167. EECS Department, University of California, Berkeley, (June 2015). http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-167.html.

[14] Chen Chen, Xiaoyan Xiang, Chang Liu, Yunhai Shang, Ren Guo, Dongqi Liu, Yimin Lu, Ziyi Hao, Jiahui Luo, Zhijian Chen, Chunqiang Li, Yu Pu, Jianyi Meng, Xiaolang Yan, Yuan Xie, and Xiaoning Qi. 2020. Xuantie-910: a commercial multi-core 12-stage pipeline out-of-order 64-bit high performance risc-v processor with vector extension : industrial product. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 52–64. DOI: 10.1109/ISCA45697.2020.00016.

[15] Shuang Chen, Shay GalOn, Christina Delimitrou, Srilatha Manne, and José F. Martínez. 2017. Workload characterization of interactive cloud services on big and small server platforms. In *2017 IEEE International Symposium on Workload Characterization (IISWC)*, 125–134. DOI: 10.1109/IISWC.2017.8167770.

[16] Xinghan Chen, Ling-Hong Hung, Robert Cordingly, and Wes Lloyd. 2023. X86 vs. arm64: an investigation of factors influencing serverless performance. In *Proceedings of the 9th International Workshop on Serverless Computing* (WoSC '23). Association for Computing Machinery, Bologna, Italy, 7–12. ISBN: 9798400704550. DOI: 10.1145/3631295.3631394.

[17] David Cock, Abishek Ramdas, Daniel Schwyn, Michael Giardino, Adam Turowski, Zhenhao He, Nora Hossle, Dario Korolija, Melissa Licciardello, Kristina Martsenko, Reto Achermann, Gustavo Alonso, and Timothy Roscoe. 2022. Enzian: an open, general, cpu/fpga platform for systems software research. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS '22). Association for Computing Machinery, Lausanne, Switzerland, 434–451. ISBN: 9781450392051. DOI: 10.1145/3503222.3507742.

[18] Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, and Torsten Hoefler. 2021. Sebs: a serverless benchmark suite for function-as-a-service computing. In *Proceedings of the 22nd International Middleware Conference* (Middleware '21). Association for Computing Machinery, Québec city, Canada, 64–78. DOI: 10.1145/3464298.3476133.

[19] Datadog. 2020. The state of serverless 2020. Datadog. https://www.datadoghq.com/state-of-serverless-2020.

[20] Datadog. 2022. The state of serverless 2021. Datadog. (June 2022). https://www.datadoghq.com/state-of-serverless.

[21] Simon Eismann, Joel Scheuner, Erwin van Eyk, Maximilian Schwinger, Johannes Grohmann, Nikolas Herbst, Cristina L. Abad, and Alexandru Iosup. 2022. The state of serverless applications: collection, characterization, and community consensus. *IEEE Transactions on Software Engineering*, 48, 10, 4152–4166. DOI: 10.1109/TSE.2021.3113940.

[22] Mahyar Emami, Sahand Kashani, Keisuke Kamahori, Mohammad Sepehr Pourghannad, Ritik Raj, and James R. Larus. 2023. Manticore: hardware-accelerated rtl simulation with static bulk-synchronous parallelism. (2023). arXiv: 2301.09413 `[cs.AR]`.

[23] Erhu Feng, Xu Lu, Dong Du, Bicheng Yang, Xueqiang Jiang, Yubin Xia, Binyu Zang, and Haibo Chen. 2021. Scalable memory protection in the PENGLAI enclave. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. USENIX Association, (July 2021), 275–294. ISBN: 978-1-939133-22-9. https://www.usenix.org/conference/osdi21/presentation/feng.

[24] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In http://infoscience.epfl.ch/record/173764.

[25] Sadjad Fouladi, Francisco Romero, Dan Iter, Qian Li, Shuvo Chatterjee, Christos Kozyrakis, Matei Zaharia, and Keith Winstein. 2019. From laptop to lambda: outsourcing everyday jobs to thousands of transient functional containers. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. USENIX Association, Renton, WA, (July 2019), 475–488. ISBN: 978-1-939133-03-8. http://www.usenix.org/conference/atc19/presentation/fouladi.

[26] Sadjad Fouladi, Riad S. Wahby, Brennan Shacklett, Karthikeyan Vasuki Balasubramaniam, William Zeng, Rahul Bhalerao, Anirudh Sivaraman, George Porter, and Keith Winstein. 2017. Encoding, fast and slow: Low-Latency video processing using thousands of tiny threads. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, (Mar. 2017), 363–376. ISBN: 978-1-931971-37-9. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/fouladi.

[27] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, et al. 2016. The sunway taihulight supercomputer: system and applications. *Science China Information Sciences*, 59, 1–16. DOI: 10.1007/s11432-016-5588-7.

[28] Shay Gal-On and Markus Levy. 2012. Exploring coremark a benchmark maximizing simplicity and efficacy. *The Embedded Microprocessor Benchmark Consortium*.

[29] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, Kelvin Hu, Meghna Pancholi, Yuan He, Brett Clancy, Chris Colen, Fukang Wen, Catherine Leung, Siyuan Wang, Leon Zaruvinsky, Mateo Espinosa, Rick Lin, Zhongling Liu, Jake Padilla, and Christina Delimitrou. 2019. An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS '19). Association for Computing Machinery, Providence, RI, USA, 3–18. ISBN: 9781450362405. DOI: 10.1145/3297858.3304013.

[30] Google. 2023. gVisor. Google. https://gvisor.dev.

[31] Qi Guo, Tianshi Chen, Yunji Chen, and Franz Franchetti. 2016. Accelerating architectural simulation via statistical techniques: a survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35, 3, 433–446. DOI: 10.1109/TCAD.2015.2481796.

[32] Xuan Guo and Robert Mullins. 2020. Accelerate cycle-level full-system simulation of multi-core RISC-V systems with binary translation. *arXiv preprint arXiv:2005.11357*.

[33] Zvika Guz, Evgeny Bolotin, Idit Keidar, Avinoam Kolodny, Avi Mendelson, and Uri C. Weiser. 2009. Many-core vs. many-thread machines: stay away from the valley. *IEEE Computer Architecture Letters*, 8, 1, 25–28. DOI: 10.1109/L-CA.2009.4.

[34] John L. Hennessy and David A. Patterson. 2019. A new golden age for computer architecture. *Commun. ACM*, 62, 2, (Jan. 2019), 48–60. DOI: 10.1145/3282307.

[35] Jason Howard, Saurabh Dighe, Yatin Hoskote, Sriram Vangal, David Finan, Gregory Ruhl, David Jenkins, Howard Wilson, Nitin Borkar, Gerhard Schrom, Fabrice Pailet, Shailendra Jain, Tiju Jacob, Satish Yada, Sraven Marella, Praveen Salihundam, Vasantha Erraguntla, Michael Konow, Michael Riepen, Guido Droege, Joerg Lindemann, Matthias Gries, Thomas Apel, Kersten Henriss, Tor Lund-Larsen, Sebastian Steibl, Shekhar Borkar, Vivek De, Rob Van Der Wijngaart, and Timothy Mattson. 2010. A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, 108–109. DOI: 10.1109/ISSCC.2010.5434077.

[36] Intel Corporation. 2023. Four takeaways from Intel's investor webinar. Intel Corporation. (Mar. 29, 2023). https://www.intel.com/content/www/us/en/newsroom/news/four-takeaways-from-intel-investor-webinar.html.

[37] Intel Corporation. 2023. Intel Xeon Gold 6238T processor ARK. https://ark.intel.com/content/www/us/en/ark/products/192439/intel-xeon-gold-6238t-processor-30-25m-cache-1-90-ghz.html.

[38] Aamer Jaleel, Robert S Cohn, Chi-Keung Luk, and Bruce Jacob. 2008. Cmp$im: a pin-based on-the-fly multi-core cache simulator. In *Proceedings of the Fourth Annual Workshop on Modeling, Benchmarking and Simulation (MoBS), co-located with ISCA*, 28–36.

[39] Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. 2017. Occupy the cloud: distributed computing for the 99%. In *Proceedings of the 2017 Symposium on Cloud Computing* (SoCC '17). Association for Computing Machinery, Santa Clara, California, 445–451. ISBN: 9781450350280. DOI: 10.1145/3127479.3128601.

[40] Artjom Joosen, Ahmed Hassan, Martin Asenov, Rajkarn Singh, Luke Darlow, Jianfeng Wang, and Adam Barker. 2023. How does it function? characterizing long-term trends in production serverless workloads. In *Proceedings of the 2023 ACM Symposium on Cloud Computing* (SoCC '23). Association for Computing Machinery, , Santa Cruz, CA, USA, 443–458. ISBN: 9798400703874. DOI: 10.1145/3620678.3624783.

[41] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45, 2, (June 2017), 1–12. DOI: 10.1145/3140659.3080246.

[42] Kostis Kaffes, Neeraja J. Yadwadkar, and Christos Kozyrakis. 2022. Hermod: principled and practical scheduling for serverless functions. In *Proceedings of the 13th Symposium on Cloud Computing* (SoCC '22). Association for Computing Machinery, San Francisco, California, 289–305. ISBN: 9781450394147. DOI: 10.1145/3542929.3563468.

[43] Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, Dayeol Lee, Nathan Pemberton, Emmanuel Amaro, Colin Schmidt, Aditya Chopra, Qijing Huang, Kyle Kovacs, Borivoje Nikolic, Randy Katz, Jonathan Bachrach, and Krste Asanovic. 2018. Firesim: fpga-accelerated cycle-exact scale-out system simulation in the public cloud. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 29–42. DOI: 10.1109/ISCA.2018.00014.

[44] Jeongchul Kim and Kyungyong Lee. 2019. Functionbench: a suite of workloads for serverless cloud function service. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, 502–504. DOI: 10.1109/CLOUD.2019.00091.

[45] Georgios K. Konstadinidis, Hongping Penny Li, Francis Schumacher, Venkat Krishnaswamy, Hoyeol Cho, Sudesna Dash, Robert P. Masleid, Chaoyang Zheng, Yuanjung David Lin, Paul Loewenstein, Heechoul Park, Vijay Srinivasan, Dawei Huang, Changku Hwang, Wenjay Hsu, Curtis McAllister, Jeff Brooks, Ha Pham, Sebastian Turullols, Yifan Yanggong, Robert Golla, Alan P. Smith, and Ali Vahidsafa. 2016. SPARC M7: a 20 nm 32-core 64 MB L3 cache processor. *IEEE Journal of Solid-State Circuits*, 51, 1, 79–91. DOI: 10.1109/JSSC.2015.2456902.

[46] Tom Kuchler, Michael Giardino, Timothy Roscoe, and Ana Klimovic. 2023. Function as a function. In *Proceedings of the 2023 ACM Symposium on Cloud Computing* (SoCC '23). Association for Computing Machinery, Santa Cruz, USA, 81–92. ISBN: 9798400703874. DOI: 10.1145/3620678.3624648.

[47] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: an open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems* (EuroSys '20) Article 38. Association for Computing Machinery, Heraklion, Greece, 16 pages. ISBN: 9781450368827. DOI: 10.1145/3342195.3387532.

[48] Shang Li, Zhiyuan Yang, Dhiraj Reddy, Ankur Srivastava, and Bruce Jacob. 2020. Dramsim3: a cycle-accurate, thermal-capable dram simulator. *IEEE Computer Architecture Letters*, 19, 2, 106–109. DOI: 10.1109/LCA.2020.2973991.

[49] Zijun Li, Jiagan Cheng, Quan Chen, Eryu Guan, Zizheng Bian, Yi Tao, Bin Zha, Qiang Wang, Weidong Han, and Minyi Guo. 2022. RunD: a lightweight secure container runtime for high-density deployment and high-concurrency startup in serverless computing. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. USENIX Association, Carlsbad, CA, (July 2022), 53–68. ISBN: 978-1-939133-29-27. https://www.usenix.org/conference/atc22/presentation/li-zijun-rund.

[50] Zijun Li, Yushi Liu, Linsong Guo, Quan Chen, Jiagan Cheng, Wenli Zheng, and Minyi Guo. 2022. Faasflow: enable efficient workflow execution for function-as-a-service. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS '22). Association for Computing Machinery, Lausanne, Switzerland, 782–796. ISBN: 9781450392051. DOI: 10.1145/3503222.3507717.

[51] Harrison Liew, Daniel Grubb, John Wright, Colin Schmidt, Nayiri Krzysztofowicz, Adam Izraelevitz, Edward Wang, Krste Asanović, Jonathan Bachrach, and Borivoje Nikolić. 2022. Hammer: a modular and reusable physical design

flow tool: invited. In *Proceedings of the 59th ACM/IEEE Design Automation Conference* (DAC '22). Association for Computing Machinery, San Francisco, California, 1335–1338. ISBN: 9781450391429. DOI: 10.1145/3489517.3530672.

[52] Pejman Lotfi-Kamran, Boris Grot, Michael Ferdman, Stavros Volos, Onur Kocberber, Javier Picorel, Almutaz Adileh, Djordje Jevdjic, Sachin Idgunji, Emre Ozer, and Babak Falsafi. 2012. Scale-out processors. *SIGARCH Comput. Archit. News*, 40, 3, (June 2012), 500–511. DOI: 10.1145/2366231.2337217.

[53] Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amslinger, Matteo Andreozzi, Adrià Armejach, Nils Asmussen, Srikant Bharadwaj, Gabe Black, Gedare Bloom, Bobby R. Bruce, Daniel Rodrigues Carvalho, Jerónimo Castrillón, Lizhong Chen, Nicolas Derumigny, Stephan Diestelhorst, Wendy Elsasser, Marjan Fariborz, Amin Farmahini Farahani, Pouya Fotouhi, Ryan Gambord, Jayneel Gandhi, Dibakar Gope, Thomas Grass, Bagus Hanindhito, Andreas Hansson, Swapnil Haria, Austin Harris, Timothy Hayes, Adrian Herrera, Matthew Horsnell, Syed Ali Raza Jafri, Radhika Jagtap, Hanhwi Jang, Reiley Jeyapaul, Timothy M. Jones, Matthias Jung, Subash Kannoth, Hamidreza Khaleghzadeh, Yuetsu Kodama, Tushar Krishna, Tommaso Marinelli, Christian Menard, Andrea Mondelli, Tiago Mück, Omar Naji, Krishnendra Nathella, Hoa Nguyen, Nikos Nikoleris, Lena E. Olson, Marc S. Orr, Binh Pham, Pablo Prieto, Trivikram Reddy, Alec Roelke, Mahyar Samani, Andreas Sandberg, Javier Setoain, Boris Shingarov, Matthew D. Sinclair, Tuan Ta, Rahul Thakur, Giacomo Travaglini, Michael Upton, Nilay Vaish, Ilias Vougioukas, Zhengrong Wang, Norbert Wehn, Christian Weis, David A. Wood, Hongil Yoon, and Éder F. Zulian. 2020. The gem5 simulator: version 20.0+. *CoRR*, abs/2007.03152. https://arxiv.org/abs/2007.03152 arXiv: 2007.03152.

[54] Michael McKeown, Alexey Lavrov, Mohammad Shahrad, Paul J. Jackson, Yaosheng Fu, Jonathan Balkind, Tri M. Nguyen, Katie Lim, Yanqi Zhou, and David Wentzlaff. 2018. Power and energy characterization of an open source 25-core manycore processor. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 762–775. DOI: 10.1109/HPCA.2018.00070.

[55] OpenCV team. 2023. OpenCV documentation. OpenCV team. Retrieved July 5, 2023 from https://docs.opencv.org/index.html.

[56] Tapti Palit, Yongming Shen, and Michael Ferdman. 2016. Demystifying cloud benchmarking. In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. (Apr. 2016), 122–132. DOI: 10.1109/ISPASS.2016.7482080.

[57] Carl Ramey. 2011. TILE-gx100 manycore processor: acceleration interfaces and architecture. In *2011 IEEE Hot Chips 23 Symposium (HCS)*, 1–21. DOI: 10.1109/HOTCHIPS.2011.7477491.

[58] Parthasarathy Ranganathan, Daniel Stodolsky, Jeff Calow, Jeremy Dorfman, Marisabel Guevara, Clinton Wills Smullen IV, Aki Kuusela, Raghu Balasubramanian, Sandeep Bhatia, Prakash Chauhan, Anna Cheung, In Suk Chong, Niranjani Dasharathi, Jia Feng, Brian Fosco, Samuel Foss, Ben Gelb, Sara J. Gwin, Yoshiaki Hase, Da-ke He, C. Richard Ho, Roy W. Huffman Jr., Elisha Indupalli, Indira Jayaram, Poonacha Kongetira, Cho Mon Kyaw, Aaron Laursen, Yuan Li, Fong Lou, Kyle A. Lucke, JP Maaninen, Ramon Macias, Maire Mahony, David Alexander Munday, Srikanth Muroor, Narayana Penukonda, Eric Perkins-Argueta, Devin Persaud, Alex Ramirez, Ville-Mikko Rautio, Yolanda Ripley, Amir Salek, Sathish Sekar, Sergey N. Sokolov, Rob Springer, Don Stark, Mercedes Tan, Mark S. Wachsler, Andrew C. Walton, David A. Wickeraad, Alvin Wijaya, and Hon Kwan Wu. 2021. Warehouse-scale video acceleration: co-design and deployment in the wild. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS 2021). Association for Computing Machinery, Virtual, USA, 600–615. ISBN: 9781450383172. DOI: 10.1145/3445814.3446723.

[59] Ali Raza, Ibrahim Matta, Nabeel Akhtar, Vasiliki Kalavri, and Vatche Isahagian. 2021. SoK: function-as-a-service: from an application developer's perspective. *Journal of Systems Research*, 1, 1. DOI: 10.5070/SR31154815.

[60] RISC-V International. 2023. RISC-V recently ratified extensions. RISC-V International. Retrieved July 16, 2023 from https://wiki.riscv.org/display/HOME/Recently+Ratified+Extensions.

[61] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. 2011. DRAMSim2: a cycle accurate memory system simulator. *IEEE Computer Architecture Letters*, 10, 1, 16–19. DOI: 10.1109/L-CA.2011.4.

[62] Rohan Basu Roy, Tirthak Patel, and Devesh Tiwari. 2022. Icebreaker: warming serverless functions better with heterogeneity. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS '22). Association for Computing Machinery, Lausanne, Switzerland, 753–767. ISBN: 9781450392051. DOI: 10.1145/3503222.3507750.

[63] Daniel Sanchez and Christos Kozyrakis. 2013. ZSim: fast and accurate microarchitectural simulation of thousand-core systems. In *Proceedings of the 40th Annual International Symposium on Computer Architecture* (ISCA '13). Association for Computing Machinery, Tel-Aviv, Israel, 475–486. ISBN: 9781450320795. DOI: 10.1145/2485922.2485963.

[64] Divyanshu Saxena, Tao Ji, Arjun Singhvi, Junaid Khalid, and Aditya Akella. 2022. Memory deduplication for serverless computing with medes. In *Proceedings of the Seventeenth European Conference on Computer Systems* (EuroSys '22). Association for Computing Machinery, Rennes, France, 714–729. ISBN: 9781450391627. DOI: 10.1145/3492321.3524272.

[65] David Schall, Artemiy Margaritov, Dmitrii Ustiugov, Andreas Sandberg, and Boris Grot. 2022. Lukewarm serverless functions: characterization and optimization. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (ISCA '22). Association for Computing Machinery, New York, New York, 757–770. ISBN: 9781450386104. DOI: 10.1145/3470496.3527390.

[66] Johann Schleier-Smith, Vikram Sreekanti, Anurag Khandelwal, Joao Carreira, Neeraja J. Yadwadkar, Raluca Ada Popa, Joseph E. Gonzalez, Ion Stoica, and David A. Patterson. 2021. What serverless computing is and should become: the next phase of cloud computing. *Commun. ACM*, 64, 5, (Apr. 2021), 76–84. DOI: 10.1145/3406011.

[67] Larry Seiler, Doug Carmean, Eric Sprangle, Tom Forsyth, Michael Abrash, Pradeep Dubey, Stephen Junkins, Adam Lake, Jeremy Sugerman, Robert Cavin, Roger Espasa, Ed Grochowski, Toni Juan, and Pat Hanrahan. 2008. Larrabee: a many-core x86 architecture for visual computing. *ACM Trans. Graph.*, 27, 3, (Aug. 2008), 1–15. DOI: 10.1145/1360612.1360617.

[68] Mohammad Shahrad, Jonathan Balkind, and David Wentzlaff. 2019. Architectural implications of function-as-a-service computing. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (MICRO '52). Association for Computing Machinery, Columbus, OH, USA, 1063–1075. ISBN: 9781450369381. DOI: 10.1145/3352460.3358296.

[69] Mohammad Shahrad, Rodrigo Fonseca, Inigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the wild: characterizing and optimizing the serverless workload at a large cloud provider. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, (July 2020), 205–218. ISBN: 978-1-939133-14-4. https://www.usenix.org/conference/atc20/presentation/shahrad.

[70] Harsh Sharma, Sumit K. Mandal, Janardhan Rao Doppa, Umit Ogras, and Partha Pratim Pande. 2023. Achieving datacenter-scale performance through chiplet-based manycore architectures. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1–6. DOI: 10.23919/DATE56975.2023.10137125.

[71] StarFive. 2023. VisionFive2 Technical Documentation. https://doc-en.rvspace.org/Doc_Center/visionfive_2.html.

[72] Jovan Stojkovic, Chunao Liu, Muhammad Shahbaz, and Josep Torrellas. 2023. μManycore: a cloud-native CPU for tail at scale. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (ISCA '23) Article 33. Association for Computing Machinery, Orlando, FL, USA, 15 pages. ISBN: 9798400700958. DOI: 10.1145/3579371.3589068.

[73] Neil C. Thompson and Svenja Spanuth. 2021. The decline of computers as a general purpose technology. *Commun. ACM*, 64, 3, (Feb. 2021), 64–72. DOI: 10.1145/3430936.

[74] Dmitrii Ustiugov, Plamen Petrov, Marios Kogias, Edouard Bugnion, and Boris Grot. 2021. Benchmarking, analysis, and optimization of serverless function snapshots. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS '21). Association for Computing Machinery, Virtual, USA, 559–572. ISBN: 9781450383172. DOI: 10.1145/3445814.3446714.

[75] Ao Wang, Shuai Chang, Huangshi Tian, Hongqi Wang, Haoran Yang, Huiba Li, Rui Du, and Yue Cheng. 2021. FaaSNet: scalable and fast provisioning of custom serverless container runtimes at alibaba cloud function compute. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, (July 2021), 443–457. ISBN: 978-1-939133-23-6. https://www.usenix.org/conference/atc21/presentation/wang-ao.

[76] David Wang, Brinda Ganesh, Nuengwong Tuaycharoen, Kathleen Baynes, Aamer Jaleel, and Bruce Jacob. 2005. Dramsim: a memory system simulator. *SIGARCH Comput. Archit. News*, 33, 4, (Nov. 2005), 100–107. DOI: 10.1145/1105734.1105748.

[77] Ziqi Wang, Kaiyang Zhao, Pei Li, Andrew Jacob, Michael Kozuch, Todd Mowry, and Dimitrios Skarlatos. 2023. Memento: architectural support for ephemeral memory management in serverless environments. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture* (MICRO '23). Association for Computing Machinery, , Toronto, ON, Canada, 122–136. ISBN: 9798400703294. DOI: 10.1145/3613424.3623795.

[78] Andrew Waterman, Yunsup Lee, David A. Patterson, and Krste Asanović. 2019. The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 20191214. Tech. rep. RISC-V Foundation, (Dec. 2019). https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf.

[79] Robert N.M. Watson, Jonathan Woodruff, Peter G. Neumann, Simon W. Moore, Jonathan Anderson, David Chisnall, Nirav Dave, Brooks Davis, Khilan Gudka, Ben Laurie, Steven J. Murdoch, Robert Norton, Michael Roe, Stacey Son, and

Munraj Vadera. 2015. Cheri: a hybrid capability-system architecture for scalable software compartmentalization. In *2015 IEEE Symposium on Security and Privacy*, 20–37. DOI: 10.1109/SP.2015.9.

[80] Robert NM Watson, Peter G Neumann, Jonathan Woodruff, Michael Roe, Hesham Almatary, Jonathan Anderson, John Baldwin, David Chisnall, Brooks Davis, Nathaniel Wesley Filardo, et al. 2019. Capability hardware enhanced RISC instructions: CHERI instruction-set architecture (version 9). Tech. rep. University of Cambridge, Computer Laboratory, (Sept. 2019). https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-987.pdf.

[81] Mark Wyse, Daniel Petrisko, Farzam Gilani, Yuan-Mao Chueh, Paul Gao, Dai Cheol Jung, Sripathi Muralitharan, Shashank Vijaya Ranga, Mark Oskin, and Michael Taylor. 2022. The blackparrot bedrock cache coherence system. (2022). arXiv: 2211.06390 [cs.AR].

[82] Jing Xia, Chuanning Cheng, Xiping Zhou, Yuxing Hu, and Peter Chun. 2021. Kunpeng 920: the first 7-nm chiplet-based 64-core ARM SoC for cloud services. *IEEE Micro*, 41, 5, 67–75. DOI: 10.1109/MM.2021.3085578.

[83] Yinan Xu, Zihao Yu, Dan Tang, Guokai Chen, Lu Chen, Lingrui Gou, Yue Jin, Qianruo Li, Xin Li, Zuojun Li, Jiawei Lin, Tong Liu, Zhigang Liu, Jiazhan Tan, Huaqiang Wang, Huizhe Wang, Kaifan Wang, Chuanqi Zhang, Fawang Zhang, Linjuan Zhang, Zifei Zhang, Yangyang Zhao, Yaoyang Zhou, Yike Zhou, Jiangrui Zou, Ye Cai, Dandan Huan, Zusong Li, Jiye Zhao, Zihao Chen, Wei He, Qiyuan Quan, Xingwu Liu, Sa Wang, Kan Shi, Ninghui Sun, and Yungang Bao. 2022. Towards Developing High Performance RISC-V Processors Using Agile Methodology. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 1178–1199. DOI: 10.1109/MICRO56248.2022.00080.

[84] Minchen Yu, Tingjia Cao, Wei Wang, and Ruichuan Chen. 2023. Following the data, not the function: rethinking function orchestration in serverless computing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, (Apr. 2023), 1489–1504. ISBN: 978-1-939133-33-5. https://www.usenix.org/conference/nsdi23/presentation/yu.

[85] Tianyi Yu, Qingyuan Liu, Dong Du, Yubin Xia, Binyu Zang, Ziqian Lu, Pingchao Yang, Chenggang Qin, and Haibo Chen. 2020. Characterizing serverless platforms with serverlessbench. In *Proceedings of the 11th ACM Symposium on Cloud Computing* (SoCC '20). Association for Computing Machinery, Virtual Event, USA, 30–44. ISBN: 9781450381376. DOI: 10.1145/3419111.3421280.

[86] Florian Zaruba and Luca Benini. 2019. The cost of application-class processing: energy and performance analysis of a Linux-ready 1.7-GHz 64-bit RISC-V core in 22-nm FDSOI technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27, 11, (Nov. 2019), 2629–2640. DOI: 10.1109/TVLSI.2019.2926114.

[87] Jerry Zhao, Ben Korpan, Abraham Gonzalez, and Krste Asanovic. 2020. SonicBOOM: the 3rd generation Berkeley out-of-order machine. In *Fourth Workshop on Computer Architecture Research with RISC-V*. (May 2020).

[88] Shixuan Zhao, Pinshen Xu, Guoxing Chen, Mengya Zhang, Yinqian Zhang, and Zhiqiang Lin. 2023. Reusable enclaves for confidential serverless computing. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, (Aug. 2023), 4015–4032. ISBN: 978-1-939133-37-3. https://www.usenix.org/conference/usenixsecurity23/presentation/zhao-shixuan.

[89] Ziqiao Zhou, Yizhou Shan, Weidong Cui, Xinyang Ge, Marcus Peinado, and Andrew Baumann. 2023. Core slicing: closing the gap between leaky confidential VMs and bare-metal cloud. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. USENIX Association, Boston, MA, (July 2023), 247–267. ISBN: 978-1-939133-34-2. https://www.usenix.org/conference/osdi23/presentation/zhou-ziqiao.