MicroFaaS: Energy-efficient Serverless on Bare-metal Single-board Computers

Anthony Byrne[†], Yanni Pang[†*], Allen Zou[†*], Shripad Nadgowda[§] and Ayse K. Coskun[†]
[†]Boston University, Boston, MA 02215; Emails: {abyrne19, yanni, allenzou, acoskun}@bu.edu
[§]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598; Email: nadgowda@us.ibm.com

***Abstract*—Serverless function-as-a-service (FaaS) platforms offer a radically-new paradigm for cloud software development, yet the hardware infrastructure underlying these platforms is based on a decades-old design pattern. The rise of FaaS presents an opportunity to reimagine cloud infrastructure to be more energy-efficient, cost-effective, reliable, and secure. In this paper, we show how replacing handfuls of x86-based rack servers with hundreds of ARM-based single-board computers could lead to a virtualization-free, energy-proportional cloud that achieves this vision. We call our systematically-designed implementation *MicroFaaS*, and we conduct a thorough evaluation and cost analysis comparing *MicroFaaS* to a throughput-matched FaaS platform implemented in the style of conventional virtualization-based cloud systems. Our results show a 5.6x increase in energy efficiency and 34.2% decrease in total cost of ownership compared to our baseline.**

***Index Terms*—serverless infrastructure, bare metal, function-as-a-service, single-board computers, energy-proportional computing**

I. Introduction

The story of cloud computing has long been driven by a desire to improve efficiency. Office space efficiency drove servers out of the basements of corporate headquarters and into colocation facilities. Cost efficiency then motivated the multiplexing of those physical servers with virtual machines (VMs). Finally, what some CEOs might call "personnel efficiency" catalyzed the shift of those VMs out of the care of expensive in-house IT staff and into the vast datacenters and billed-by-the-hour auspices of Amazon, Google, Microsoft, and other commercial cloud providers. Thus, the story continues with the rise of "microVMs," containers, and most recently, function-as-a-service (FaaS) or "serverless."[1]

Despite this focus on efficiency and the fact that cloud datacenters consumed over 1% of all electricity used globally in 2020, the hardware architectures underlying the cloud have largely remained centered around the venerable x86-based rack server—and its lackluster energy efficiency—for decades [1], [2]. Meanwhile, the x86 CPUs powering most of these servers have stagnated, struggling to eke out >5% estimated year-over-year improvements in single-program speed since 2017 [3]. In this paper, we imagine a cloud architecture that breaks free of the conventional model by exploiting the stateless and granular nature of serverless functions, which can be securely run-to-completion on energy-efficient ARM-based single-board computers (SBCs) without the need for expensive virtualization.[2] We call our prototype implementation of this vision *MicroFaaS*, and we hope that by replacing rack servers with hundreds of hardware-isolated worker nodes, *MicroFaaS* will significantly increase cloud energy efficiency and security while reducing costs for both cloud operators and users.

It is worth noting that we are neither arguing for a wholesale rejection of the conventional cloud datacenter model nor proposing a complete replacement. Instead, we are suggesting there exists a cheaper, more energy-efficient, and more secure infrastructural model *specifically for serverless FaaS platforms* compared to the conventional cloud infrastructures that most cloud operators have built their recent serverless platforms atop. In addition to proposing and implementing such a novel model, we argue that the rise of serverless computing presents a unique opportunity for radical infrastructural change due to FaaS platforms' inherently high-level interface and recent work showing that conventional cloud architectures handle serverless workloads rather inefficiently [4].

MicroFaaS differentiates itself from previous energy-efficient cloud proposals by focusing entirely on FaaS workloads and altogether eschewing virtualization in favor of hardware isolation, single-tenancy, and a run-to-completion scheduling model. More specifically, our contributions include:

- a detailed proposal for *MicroFaaS*, an energy-efficient FaaS platform architecture based on a cluster of SBCs acting as single-tenant worker nodes with highly-reproducible and virtualization-free execution environments.
- a systematically-designed prototype implementation of *MicroFaaS*, including open-source cluster orchestration software, a Linux distribution highly optimized for use as a worker's execution environment, and a workload suite composed of both original and previously-published serverless functions.[3]
- a thorough evaluation comparing our implementation of *MicroFaaS* to a traditional virtualization-based serverless platform along the axes of throughput, energy consumption, and economic cost efficiency, in which we demonstrate a 5.6x increase in energy efficiency and a 34.2% decrease in total cost of ownership (TCO).

We begin by offering background information, including related works, in Section II. Then, after explaining our proposed

This work is partially supported by the IBM T.J. Watson Research Center and the Boston University Undergraduate Research Opportunities Program.

*Yanni Pang and Allen Zou contributed equally to this work.

[1]We use the terms "serverless" and "FaaS" interchangeably to refer to an event-driven cloud computing paradigm in which developers write stateless functions to be hosted on and automatically managed by a cloud platform.

[2]We use a broad definition of virtualization in this paper that includes both conventional VMs and containerization methods like LXC.

[3]See github.com/peaclab for all open-source artifacts of this work.

architectural approach to serverless in Section III, we design and evaluate a prototype implementation of *MicroFaaS* in Section IV. Next, we discuss the results of our evaluation in Section V. Finally, we conclude and discuss future work in Section VI.

II. Background & Related Work

Serverless represents the next generation of cloud computing, offering cloud platform users fine-grained resource metering and near-instantaneous autoscaling with little-to-no server administration knowledge necessary. By design, these benefits give cloud providers unprecedented levels of freedom in the design and management of the hardware and software stacks underlying their customers' applications, as most serverless users, unlike users of VMs or containers, do not care to directly manipulate infrastructure-level details (e.g., virtual hard disk size). In other words, serverless allows cloud providers to radically reconfigure their infrastructure to be optimized for any metric(s) (e.g., energy efficiency, reliability, or both) however they see fit, as long as they fulfill customer expectations.

Of course, many cloud computing platforms have existed for decades longer than modern serverless offerings. We refer to the infrastructural model these older platforms typically follow as "conventional cloud architecture," which we define as a hardware/software design pattern centered around warehouse-like datacenters filled with large numbers of racks, each holding dozens of x86-based rack servers typically acting as hypervisors for the "instances" (i.e., VMs or containers) they host. Customers can then upload software to these instances and pay-as-they-go for the hardware resources they consume.

Conventional cloud architecture has been and will likely continue to be a performant and profitable model for most general-purpose cloud computing. Therefore, it is understandable that most cloud providers have opted to build their serverless platforms atop their existing conventional cloud infrastructures, despite the aforementioned design freedoms and research showing that conventional cloud architectures handle serverless workloads somewhat inefficiently [4], [5]. As a result, however, today's serverless platforms inherit many of conventional cloud architecture's problems, including virtualization-related security vulnerabilities and high performance variability due to practices like multi-tenancy [6], [7]. Furthermore, serverless functions' short-lived, bursty nature can intensify these issues to the point of essentially negating the core benefits of serverless computing: e.g., high performance/latency variability makes it challenging to predict runtime costs or satisfy real-time guarantees.

An emerging body of work seeks to measure the potential benefits of diverging or breaking *entirely* away from conventional cloud architecture. This body includes *λ-NIC* [8], which finds that running serverless functions on ASIC-based network interface cards (a.k.a. SmartNICs) can result in "two orders of magnitude improvement in latency and throughput compared to existing serverless compute frameworks." Said functions, however, must be written in a restrictive C-like language (Micro-C) that lacks many of the conveniences serverless developers have come to expect from the more-robust languages supported by conventional platforms (e.g., Python, Golang, Javascript). *dReDBox* [9] finds that decoupling resources like RAM and hardware accelerators from conventional server mainboards leads to more-elastic scaling and lower datacenter TCO, but its approach still requires a conventional VM-based abstraction layer. Other works in this area explore serverless computing on edge devices; for example, Gand *et al.* [10] experiment with a cluster of 8 Raspberry Pi SBCs running Docker Swarm, openFaaS, and Prometheus and find that throughput was bottlenecked by the SBCs' limited hardware resources and networking capacity.

It is worth noting that, except for *dReDBox*, none of these works address the energy- or cost-efficiency of their proposed optimizations. Previous proposals for more energy-efficient cloud platforms have been mainly restricted to incremental optimizations of the conventional hardware model, e.g., consolidation of VMs onto fewer physical rack servers such that excess servers can be powered-down in an effort to approximate energy-proportional computing [2], [11]. More radical proposals, such as those that explore non-x86 cloud datacenters (e.g., ARM-based proposals like *Cloud/IX* [12]) and clouds composed of clusters of SBCs (e.g., *Kittyhawk* [13]), have demonstrated measurable energy- or cost-efficiency benefits, but neither commercial nor research cloud operators have widely adopted them. We largely attribute this to a mismatch between the often monolithic and long-running workloads that defined the first two decades of cloud computing (e.g., web servers and databases) and the highly parallelized nature and low single-thread execution speed of early ARM CPUs [14]. We contrast this with today's serverless functions, which are short-running, stateless, and therefore much better suited to an ARM-based cloud platform.

III. MicroFaaS Approach

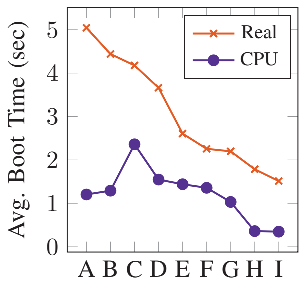
In this paper, we propose an architecture for serverless hardware infrastructure based on the hypothesis that serverless functions are better suited to smaller, low-overhead execution environments than the highly multiplexed and virtualized environments provided by conventional infrastructure. Therefore, we propose replacing most large multi-core rack servers in a conventional serverless cluster with **many single-core hardware-isolated compute nodes**, i.e., SBCs (such as the BeagleBone Black or Raspberry Pi Compute Module). In light of such devices' resource constraints, each node runs only the **bare minimum amount of software** required to provide an execution environment—e.g., a "just-enough operating system," similar to *Container Linux* or *Alpine Linux*, running just a lightweight Python interpreter such as *MicroPython* [15]. Nodes operate under a **single-tenant, run-to-completion model**—i.e., no other function will run on a node once the assigned function has started executing. Between function executions, the node is reinitialized to a known state (e.g., by rebooting), guaranteeing each function a **fully reproducible environment** untainted by potentially-malicious code leftover from previous tenants. Furthermore, if the computational capacity offered by a node is not needed at any given time, the node is put into a low-energy sleep state or powered-off entirely, allowing for **nearly-linear energy-proportional computing** [16].

We believe our approach offers several benefits, which we discuss in detail below.

a) Hardware-based Isolation: Just like "air-gapping" a computer is the most effective way to ensure it is safe from network-based attacks, physical host separation is the most effective way to protect a serverless application from its would-be cotenants. Even applications running on fully virtualized hardware (e.g., VMs or Kata Containers) have been shown to be vulnerable to attacks such as Spectre and Meltdown [17], [18]. Allocating a single serverless function to a conventional rack server would be far too costly and inefficient, but allocation to a small, low-cost SBC mitigates these inefficiencies. Similarly, the best way to ensure the currently-scheduled FaaS function remains unaffected by previous tenants of its worker node (e.g., from rootkits or leftover cache entries) is to electronically reset its hardware to a known state—i.e., to reboot. Rebooting a conventional rack server after each function execution would be impractical, often taking 55 seconds or more to do so [19]. SBCs, on the other hand, can be rebooted in less than 2 seconds (see Sec. IV-A).

b) Reduced Energy Use: Researchers have long agreed that high-performance computing clusters built with "wimpy cores" like ARM-based CPUs can achieve significantly more computation per unit power (i.e., FLOPS per watt) than those built on "brawny cores" like x86-based CPUs [20], [21]. However, this insight has not led to widespread adoption of ARM-based CPUs in cloud datacenters, as the limited parallelizability and low average utilization patterns of classical cloud applications reduce the marginal benefits of a wimpy-core-powered cloud [14]. While this may have been true for previous cloud software architectures, we argue that serverless applications are well-positioned to reap the benefits of wimpy cores due to their discrete, stateless, and largely IO-bound functions written in platform-independent high-level languages like Python. Furthermore, *MicroFaaS*'s policy of completely powering down unutilized nodes effectively implements *energy-proportional computing* to a degree that conventional clouds have yet to achieve [2], [16]. For example, imagine a rack server consumes 100 watts while hosting ten active VMs; it is highly unlikely this server will consume 50 fewer watts once five of its VMs are suspended. In contrast, a 10-node *MicroFaaS* cluster indeed uses roughly 50% less power when five of its nodes are powered-off, and this linear relationship holds regardless of scale (see Fig. 5).

c) Cost Transparency & Reduction: Similar to the energy-proportionality benefits discussed above, a *MicroFaaS* cluster would have the added benefit of being *transparently* cost-proportional. Consider, for example, how a cloud provider would estimate the marginal and operational costs of increasing a datacenter's computational capacity such that 100,000 additional serverless functions could be in-flight (i.e., concurrently executing) at any given time (assuming the datacenter's networking, cooling, and power infrastructure is already equipped for such an upgrade). Providers operating conventional architectures would need to estimate the range of functions that a single rack server could be running at one time, the resulting range of per-function energy consumption rates, the overhead of virtualization or other isolation methods, etc., before being able to produce a loosely-bounded cost range. A provider operating a *MicroFaaS* architecture, on the other hand, can produce a tightly-bounded



	Optimization
A	Linux 5.11 default config
B	Kernel config reduction
C	Python → MicroPython
D	rootfs → initramfs
E	Enab. UBoot falcon mode
F	Disab. Ethernet auto-neg
G	Disab. PHY soft reset
H	DHCP → static IP
I	Net-config in kernelspace

Figure 1: Illustrating changes in the boot time of our worker OS throughout its development process (see Sec. IV-A). *Real* refers to the "wall clock" time that passes between power-on and the device's first network connection. *CPU* refers to the amount of time the CPU was not idle during boot (as recorded by the Linux kernel). We use an arrow (→) to mean "replaced by."

cost estimate simply by multiplying the hardware and average energy cost of a single node by 100,000. Furthermore, due to reduced cooling needs, a lack of moving parts, and lower average time-powered-on (due to energy-proportionality), we expect the SBCs that act as *MicroFaaS* nodes to fail less frequently than traditional rack servers[4], leading to an even tighter bound on operating costs. This increased cost certainty would allow serverless cloud providers to more precisely build their computational supply to match their customers' demands, leading to lower costs for both providers (due to less overprovisioning and underutilization) and customers.

IV. MicroFaaS Prototype

Our aim in designing and implementing this prototype is to assess the feasibility, performance, and energy efficiency of *MicroFaaS* as an alternative to conventional serverless architectures. To this end, we build and experimentally compare two FaaS clusters: one of low-cost SBCs (modeling *MicroFaaS*) and another of x86-based VMs (modeling a conventional serverless platform). Furthermore, we design both clusters to be *throughput-equivalent*, i.e., roughly equal in terms of functions executed per unit time, allowing us to directly compare both clusters on the bases of energy consumption, end-to-end latency, and economic cost.

A. Worker OS

Each worker machine (physical or virtual) in our test clusters runs a "barebones" Linux distribution systematically designed and built in a manner similar to *Linux From Scratch* [24]. As depicted in Fig. 1, we rigorously evaluate each change to our worker OS with the aim of reducing boot time as much as possible, beginning with our choice of Linux kernel version (**A**). We configure our kernel such that only the bare minimum features and drivers required to run on our two target platforms are compiled into the final binary (**B**). We also patch the source code of

[4]The manufacturers of the hardware we use in our evaluation, like most computer manufacturers, do not publish statistics on mean time between failures (MTBF), and calculation of such metrics is outside of the scope of this paper. Instead, we invite the reader to make comparisons between SBCs such as the Technologic Systems TS-7800-V2 (MTBF = 2,320,456 hours [22]) and server components such as the Intel Server Board S2600CW (MTBF = 234,708 hours [23]).

Table I: Workload Functions

CPU- or RAM-bound		*Network-bound*	
Name	**Description**	**Name**	**Description**
FloatOps*	floating-point trigonometric operations	RedisInsert	insert Redis key-value record
CascSHA	cascading SHA256 hash calculations	RedisUpdate	update Redis key-value record
CascMD5	cascading MD5 hash calculations	SQLSelect	query our PostgreSQL server using SELECT
MatMul*	large random matrix multiplication	SQLUpdate	query our PostgreSQL server using UPDATE
HTMLGen	dynamically generate and serve HTML	COSGet*	download from MinIO cloud object store
AES128*	cascading AES128 encryption/decryption	COSPut*	upload to MinIO cloud object store
Decompress*	extract a DEFLATE-compressed string	MQProduce	send message to Kafka topic
RegExSearch	find all regular expr. matches in input	MQConsume	receive message from Kafka topic
RegExMatch	determine if input matches regular expr.		

*Adapted from or inspired by *FunctionBench* [25].

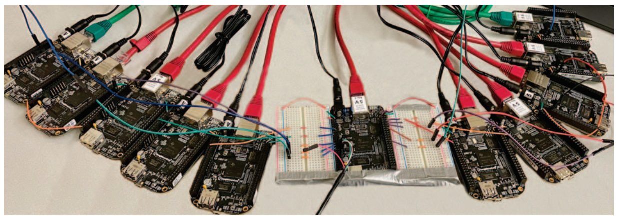


Figure 2: Our *MicroFaaS* prototype test cluster.

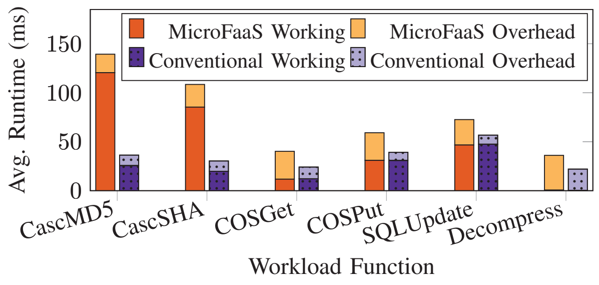


Figure 3: Runtime of select workload functions, broken into time each worker spends executing the function (*Working*) and time each worker spends receiving the function input and returning the function result over the network (*Overhead*).

certain networking drivers such that they (1) are initialized earlier in the boot process, (2) skip the auto-negotiation phase of the Ethernet connection handshake (F), and (3) avoid unnecessarily resetting PHY hardware (G)—altogether significantly minimizing NIC-related boot delays. We then create an initial ramdisk (a.k.a. initramfs) containing only the MicroPython interpreter (C) and a stripped-down version of the BusyBox software suite. Finally, we set the kernel command line arguments such that the kernel configures networking on-boot with a static IPv4 address (I, H) and uses the initramfs as the sole root filesystem (D). This results in an OS that boots quickly (1.51 seconds on ARM; 0.96 seconds on x86) and reproducibly (as the bootloader loads a clean copy of the initramfs into RAM on each boot) and can be easily ported to other SBC platforms (with the exception of one vendor-specific patch to our SBC's PHY driver, G).

B. *MicroFaaS-based Cluster*

Our *MicroFaaS*-based test cluster (shown in Fig. 2) acts as a small-scale proof-of-concept for a future datacenter-scale serverless platform. This small cluster comprises ten BeagleBone Black SBCs, each containing a single-core 1 GHz ARM Cortex-A8-based microprocessor (the TI Sitara AM3358), 512 MB of off-chip DDR3 RAM, 4 GB of eMMC flash storage, and a 10/100 Ethernet network interface. We flash each SBC's eMMC storage with our worker OS and the U-Boot bootloader compiled in "falcon mode" to further minimize boot time (E). We assign each SBC a static IPv4 address and connect it to a 24-port managed Gigabit Ethernet switch, to which we also connect our orchestration server (see Sec. IV-D).

C. *Test Workloads*

We perform all experiments in this paper using MicroPython adaptions of six Python functions from the *FunctionBench* FaaS benchmark suite[5] [25] and eleven benchmark functions of our own creation. We select or design each of the resulting 17 functions, shown in Table I, to stress the worker's CPU and memory while simulating many types of applications currently found on FaaS platforms. For functions that involve accessing a resource over the network (e.g., SQLSelect), we connect an additional SBC dedicated to hosting that resource (e.g., PostgreSQL server) to our Ethernet switch.

[5]We unfortunately cannot use all of the functions provided by *FunctionBench* (as well as most functions found in other benchmark suites we considered, e.g., *ServerlessBench* [26] and *SeBS* [27]) because they contain cloud-platform-specific code or are otherwise incompatible with MicroPython or our control plane.

D. *Cluster Orchestration*

Because most widely available open-source orchestration platforms (OPs) are not designed with bare-metal workers in mind, we build a proof-of-concept OP in Python to handle function allocation, invocation, and data collection for all experiments in this paper. We run our OP on a dedicated SBC connected to the same Ethernet network as our test clusters. In addition, we establish GPIO connections between the OP SBC and each worker SBC's `PWR_BUT` pin, allowing the OP to power on/off each worker as needed.

The OP maintains a job queue for each worker, and a job is added to a random sampling of those queues every second, simulating the arrival of function invocations. Upon assignment of a job to its queue, a worker node powers on and begins executing the job. Upon completion, the worker either reboots and executes its next job or powers down until the OP assigns it another job.

V. Evaluation Results

In order to demonstrate the feasibility of *MicroFaaS* as an energy-efficient serverless architecture, we evaluate our *MicroFaaS* prototype against a cluster of VMs acting as a small-scale model of existing serverless platforms built atop conventional cloud architecture. This cluster is composed of several QEMU "microVM" virtual machines (similar to Amazon's Firecracker

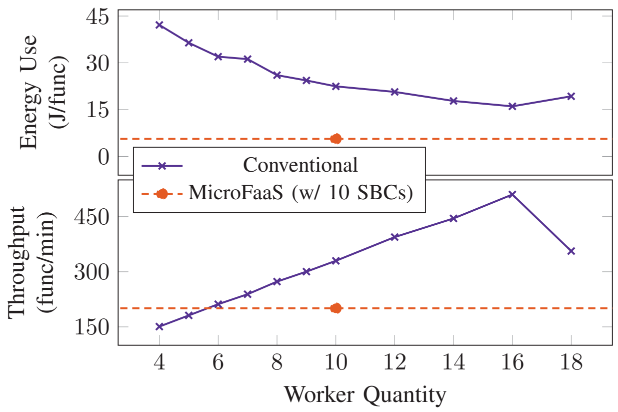


Figure 4: Analyzing the energy efficiency and throughput of a conventional virtualization-based FaaS cluster when varying the number of VMs running on a single rack server. Notice how our *MicroFaaS* cluster's energy use is consistently lower than that of the conventional cluster.

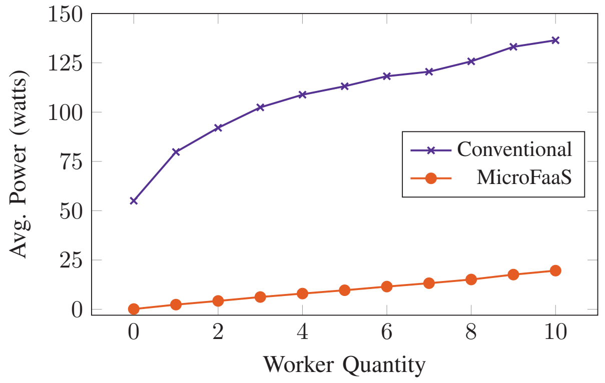


Figure 5: Demonstrating energy-proportional computing by comparing the average power consumption of multiple SBCs and that of multiple VMs on a single rack server. Notice the difference in idle (i.e., worker qty. = 0) power consumption.

concept), each allocated one vCPU and 512 MB of RAM, running on a Thinkmate RAX rack server equipped with 16 GB of RAM and a 12-core AMD Opteron 6172 processor with a maximum clock frequency of 2.1 GHz. Because we wish to ensure our conventional cluster can execute roughly the same number of functions per minute as our ten-SBC *MicroFaaS* cluster (which is capable of 200.6 func./min. on average), we choose to use six VMs (altogether capable of 211.7 func./min. on average) for most experiments. We assign each VM a static IPv4 address and a virtual NIC bridged with our host machine's physical Gigabit Ethernet NIC. We then connect the host machine to our orchestration server through a 24-port managed Gigabit Ethernet switch.

We begin evaluating both our *MicroFaaS* and conventional clusters (separately) by instructing our OP to issue 1,000 invocation requests for each of our 17 workload functions (shown in Table I), distributing each request to a random set of workers (see Sec. IV-D). Then, using the timestamps recorded by our OP and each worker, we calculate the average execution time and network-related overhead for each workload function. We show a representative set of these runtimes in Fig. 3, and we find that out of 17 functions, the *MicroFaaS* cluster executes four faster than the conventional cluster and nine at more than half the speed of the conventional cluster.

These performance results are consistent with previous observations that the ARM CPUs of most SBCs underperform the "brawnier" x86 CPUs in most servers on certain workload classes (see Sec. III-b). However, enhanced hardware or application-specific accelerators could mitigate such performance differences, albeit at the price of increased component costs or energy use. For example, upgrading our evaluation SBC's NIC from Fast Ethernet to Gigabit Ethernet would likely reduce the overhead of functions like COSGet, and adding a cryptographic accelerator might significantly reduce the runtime of CascSHA.

In exchange for the above-mentioned reduced execution speed, ARM-based SBCs are remarkably energy efficient. To demonstrate this, we measure the total energy use of each cluster using a *WattsUp Pro* power meter and find that our *MicroFaaS* cluster uses 5.7 Joules to execute a single function on average, while our conventional cluster consumes 32.0 Joules per function—**a 5.6x increase in energy efficiency**. Moreover, our *MicroFaaS* cluster's advantage holds even when we enlarge our conventional cluster to its point of peak efficiency (16.1 J/func.) by increasing the number of VMs until the resources of its underlying rack server are saturated, as shown in Fig. 4.

Additional advantages emerge upon analysis of the lifetime costs associated with *MicroFaaS* and conventional serverless clusters. We apply a simplified version[6] of a recent datacenter technology TCO model by Cui *et al.* [28] to a hypothetical 42U server rack containing 41 mid-range rack servers and a refurbished Gigabit Ethernet top-of-rack (ToR) switch. Next, we apply the same model to a hypothetical *MicroFaaS* cluster with equivalent throughput, which we estimate would be composed of 989 SBCs and 21 of the same ToR switches. We perform this analysis using the assumptions described in the Appendix over a 5-year lifespan under two sets of conditions: "ideal" conditions where every node is constantly fully-utilized and never needs replacement (i.e., 100% online rate or OR), and "realistic"

[6]We choose not to consider the infrastructural cost component of Cui *et al.*'s model due to how wildly land, building construction, and cooling costs can vary across time, geography, and specific technologies in use. We also do not consider the maintenance component of the original model due to similar variations in technician labor costs and hardware warranty terms.

Table II: 5-Year Single Rack Lifetime Cost Comparison*

	Ideal (100% Util., 100% OR)		***Realistic (50% Util., 95% OR)***	
Expense	**Conventional**	**MicroFaaS**	**Conventional**	**MicroFaaS**
Compute	$82,451	$51,923	$86,791	$54,655
Network	$574	$12,280	$574	$12,280
Energy	$41,676	$17,884	$29,242	$11,778
Total	**$124,701**	**$82,087**	**$116,607**	**$78,713**

*Based on TCO model by Cui *et al.* [28]. See the Appendix for assumptions. Costs shown in U.S. dollars.

conditions where nodes are only 50%-utilized on average and 5% of nodes require replacement at some point (i.e., 95% OR).

As demonstrated by the results shown in Table II, **the *MicroFaaS* cluster is 32.5-34.2% less expensive than a conventional cluster** with equivalent throughput, largely thanks to the low cost of SBCs and the energy-proportionality inherent in *MicroFaaS*'s design allowing unutilized nodes to consume almost no power (see Fig. 5). We highlight that this result is in spite of the high acquisition and energy costs associated with (somewhat inefficiently) networking 989 SBCs using 21 ToR switches and 1.8 kilometers (1.1 miles) of Cat6 cabling.

VI. Conclusion

In this paper, we presented *MicroFaaS*, an energy-efficient alternative to the conventional cloud architectures underlying most modern serverless platforms. We evaluated our systematically-designed *MicroFaaS* prototype against a conventional serverless platform with equivalent throughput and found that *MicroFaaS*'s use of SBCs as virtualization-free worker nodes offers measurable energy efficiency and cost benefits at the expense of performance. For many users, these significant benefits will outweigh the current performance penalty, and we hope to explore optimizations (e.g., application-specific hardware accelerators and integrations for widely-used FaaS orchestration software) in future work that will only further mitigate these penalties and demonstrate additional advantages of the *MicroFaaS* model.

Appendix: Cost Model Assumptions

When calculating "compute" costs (known as "server acquisition costs" or C_s by Cui *et al.* [28]), we assume a modern mid-range rack server (C_{server}) costs \$2,011—the retail price listed on the Dell website for a PowerEdge R6515 with 16 GB of RAM and 8-core AMD EPYC 7232P CPU, which we assume would perform similarly to the rack server we use in our evaluation. We also assume an SBC costs \$52.50—the retail price of the BeagleBone Black SBC we use in our evaluation. Throughout our analysis, we assume a hardware depreciation period of 5 years (i.e., $T_{server} = T_{core} = T_{net-rack} = 5$ years).

When calculating network costs (a.k.a. "network acquisition costs" or C_n), we assume a refurbished 48-port ToR switch ($C_{netperrack}$) costs \$500—the average price of a used Cisco Catalyst 2960S-48LPS managed switch based on several networking equipment suppliers' catalogs. We also assume that each node will connect to its ToR switch using 6 feet (1.8 meters) of Cat6 cable priced at \$0.30/foot (i.e., $C_{core-node} = \$1.80$).

When calculating energy costs (a.k.a. "power costs" or C_p), we assume the characteristics of the "benchmark datacenter" described by Cui *et al.*; i.e., our hypothetical datacenter's power usage effectiveness (PUE), server power usage effectiveness (SPUE), and price of electricity (C_{e-kwh}) are 1.3, 1.2, and \$0.10/kilowatt-hour, respectively. We assume each rack server consumes an average of 150 watts under load (P_{ss}) and 60 watts when idle ($P_{ss-idle}$), and we assume $P_{ss} = 1.96$ watts and $P_{ss-idle} = 0.128$ watts for each SBC (which fully powers down when "idle"). Finally, we assume each ToR switch consumes an average of 40.87 watts as listed on Cisco's website (i.e., $P_{net} = 40.87 * N_{rack}$ watts, where $N_{rack} = \lceil N_{server-IT} \div 48 \rceil$).

Acknowledgment

The authors thank Prof. J. Appavoo for his invaluable early feedback and the anonymous reviewers for their detailed comments.

References

[1] E. Masanet *et al.*, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, Feb. 2020. https://doi.org/ggm3sk

[2] C. Jiang *et al.*, "Energy proportional servers: Where are we in 2016?" in *37th Int. Conf. Distrib. Comput. Sys. (ICDCS)*. IEEE, 2017, pp. 1649–1660. https://doi.org/gmsk69

[3] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed. Elsevier, 2017.

[4] M. Shahrad, J. Balkind, and D. Wentzlaff, "Architectural implications of function-as-a-service computing," in *Proc. 52nd Annu. Int. Symp. Microarchitecture (MICRO)*. ACM, Oct. 2019, pp. 1063–1075.

[5] V. Yussupov *et al.*, "A systematic mapping study on engineering function-as-a-service platforms and tools," *Proc. 12th Int. Conf. Utility and Cloud Comput. (UCC)*, pp. 229–240, 2019. https://doi.org/gmsk63

[6] O. AbdElRahem, A. M. Bahaa-Eldin, and A. Taha, "Virtualization security: A survey," in *11th Int. Conf. Comput. Eng. Sys. (ICCES)*. IEEE, 2016, pp. 32–40. https://doi.org/gf8xtq

[7] J. Ericson, M. Mohammadian, and F. Santana, "Analysis of performance variability in public cloud computing," in *Int. Conf. Inform. Reuse and Integr. (IRI)*. IEEE, 2017, pp. 308–314. https://doi.org/gmsk65

[8] S. Choi *et al.*, "λ-NIC: Interactive serverless compute on programmable SmartNICs," in *40th Int. Conf. Distrib. Comput. Sys. (ICDCS)*. IEEE, 2020, pp. 67–77. https://doi.org/gv66

[9] M. Bielski *et al.*, "dReDBox: Materializing a full-stack rack-scale system prototype of a next-generation disaggregated datacenter," in *Des., Automat. & Test in Europe Conf. & Exhib. (DATE)*, Mar. 2018, pp. 1093–1098. https://doi.org/gmsk7f

[10] F. Gand *et al.*, "Serverless container cluster management for lightweight edge clouds," in *Proc. 10th Int. Conf. Cloud Comput. and Services Sci. (CLOSER)*. SCITEPRESS, Feb. 2020, pp. 302–311.

[11] A. Pahlevan *et al.*, "Energy proportionality in near-threshold computing servers and cloud data centers: Consolidating or not?" in *Des., Automat. & Test in Europe Conf. & Exhib. (DATE)*. IEEE, Mar. 2018, pp. 147–152.

[12] Y. Leokhin and P. Panfilov, "A study of Cloud/IX operating system for the ARM-based data center server platform," *Procedia Eng.*, vol. 100, pp. 1696–1705, 2015. https://doi.org/gmsk7b

[13] J. Appavoo *et al.*, "Kittyhawk: Enabling cooperation and competition in a global, shared computational system," *IBM J. Res. and Develop.*, vol. 53, no. 4, pp. 9:1–9:15, Jul. 2009. https://doi.org/dpg6v2

[14] T. Mudge and U. Hölzle, "Challenges and opportunities for extremely energy-efficient processors," *IEEE Micro*, vol. 30, no. 4, pp. 20–24, 2010.

[15] D. George *et al.*, "MicroPython," 2021. https://micropython.org

[16] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[17] P. Kocher *et al.*, "Spectre attacks: Exploiting speculative execution," in *Symp. Secur. and Privacy (SP)*. IEEE, 2019, pp. 1–19.

[18] M. Lipp *et al.*, "Meltdown: Reading kernel memory from user space," in *27th USENIX Secur. Symp.*, 2018, pp. 973–990.

[19] A. Kinzhalin *et al.*, "Enabling dynamic data centers with a smart bare-metal server platform," *Cluster Comput.*, vol. 14, no. 3, pp. 245–258, Sep. 2011. https://doi.org/fp5sx3

[20] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, p. 13–23, Jun. 2007. https://doi.org/bsdxcd

[21] D. Yokoyama *et al.*, "The survey on ARM processors for HPC," *J. Supercomput.*, vol. 75, no. 10, pp. 7003–7036, Oct. 2019. https://doi.org/gmsk68

[22] Technologic, "TS-7800 MTBF Data," 2019. https://perma.cc/QD95-7HP4

[23] Intel, "Server Board S2600CW TPS," 2020. https://perma.cc/CAB8-LMH8

[24] G. Beekmans *et al.*, *Linux From Scratch*, 10th ed., B. Dubbs, Ed., 2021.

[25] J. Kim and K. Lee, "FunctionBench: A suite of workloads for serverless cloud function service," in *12th Int. Conf. Cloud Comput. (CLOUD)*. IEEE, Jul. 2019, pp. 502–504. https://doi.org/gmsk64

[26] T. Yu *et al.*, "Characterizing serverless platforms with ServerlessBench," in *Proc. 11th Symp. Cloud Comput. (SoCC)*. ACM, Oct. 2020, pp. 30–44.

[27] M. Copik *et al.*, "SeBS: A serverless benchmark suite for function-as-a-service computing," in *22nd Int. Middleware Conf.*, Dec. 2021, pp. 64–78.

[28] Y. Cui *et al.*, "Total cost of ownership model for data center technology evaluation," in *16th Intersoc. Conf. Thermal and Thermomech. Phenomena in Electron. Sys. (ITherm)*, 2017, pp. 936–942. https://doi.org/gmtnbh