SoC-Cluster as an Edge Server: an Application-driven Measurement Study

LI ZHANG, Beijing University of Posts and Telecommunications, China
ZHE FU, Tsinghua University, China
BOQING SHI, Beijing University of Posts and Telecommunications, China
XIANG LI, Beijing University of Posts and Telecommunications, China
RUJIN LAI, Vclusters, China
CHENYANG CHEN, Vclusters, China
AO ZHOU, Beijing University of Posts and Telecommunications, China
XIAO MA, Beijing University of Posts and Telecommunications, China
SHANGGUANG WANG, Beijing University of Posts and Telecommunications, China
MENGWEI XU, Beijing University of Posts and Telecommunications, China

Huge electricity consumption is a severe issue for edge data centers. To this end, we propose a new form of edge server, namely SoC-Cluster, that orchestrates many low-power mobile system-on-chips (SoCs) through an on-chip network. For the first time, we have developed a concrete SoC-Cluster server that consists of 60 Qualcomm Snapdragon 865 SoCs in a 2U rack. Such a server has been commercialized successfully and deployed in large scale on edge clouds. The current dominant workload on those deployed SoC-Clusters is cloud gaming, as mobile SoCs can seamlessly run native mobile games.

The primary goal of this work is to demystify whether SoC-Cluster can efficiently serve more general-purpose, edge-typical workloads. Therefore, we built a benchmark suite that leverages state-of-the-art libraries for two killer edge workloads, i.e., video transcoding and deep learning inference. The benchmark comprehensively reports the performance, power consumption, and other application-specific metrics. We then performed a thorough measurement study and directly compared SoC-Cluster with traditional edge servers (with Intel CPU and NVIDIA GPU) with respect to physical size, electricity, and billing. The results reveal the advantages of SoC-Cluster, especially its high energy efficiency and the ability to proportionally scale energy consumption with various incoming loads, as well as its limitations. The results also provide insightful implications and valuable guidance to further improve SoC-Cluster and land it in broader edge scenarios.

1 INTRODUCTION

Energy efficiency has been recognized as a crucial criterion for data centers [53]. It is reported that 76.8 TWh of electricity was consumed by EU data centers in 2018, which accounted for 2.7% of the total electricity usage within the EU. This power consumption is estimated to rise to 98.52 TWh by 2030 [2]. Such high energy usage also burdens the cooling infrastructure and causes high bills for those who maintain the data centers [85, 99].

Edge clouds, which provide in-proximity computing resources to users/devices, are emerging as the new critical infrastructure for people's daily life. According to Gartner, around 75% of enterprise-generated data will be processed at the edge by 2025 [10]. The energy issue will likely be aggravated on edges for a few reasons. First, the power supply to edges is more constrained and costly, as edge servers tend to be deployed closer to the population, while cloud data centers are often placed in cherry-picked locations such as those close to hydro power supplies [94]. Edge servers are also space-restricted and have one magnitude higher power density, putting pressure on their cooling mechanisms [99]. Furthermore, edge workloads vary more than clouds, owing to the unique edge application characteristics [122]. Conventional servers can hardly scale their energy consumption with dynamic workloads in a proportional manner, as demonstrated both by prior work [53, 74] and our experiments (explained below).

A fundamental path towards green data centers is to improve the energy efficiency of every single cloud/edge server. Following this principle, most prior efforts reside at the software stack [83, 86, 89, 92, 103, 128, 133]. At the very least, hardware re-design sees more profits [104] but is hard to deploy as most cloud infrastructure is already well established. However, we sense the opportunity with edge: its infrastructure is still being finalized and we are now on the eve of its inauguration [122].

In this work, we advocate for a new form of edge server, namely SoC-Cluster, which consists of tens or hundreds of mobile SoCs like Qualcomm Snapdragon series, as a vital complement to the existing edge infrastructure. The underlying rationale is that mobile SoCs are designed to be energy efficient by using a simpler instruction set (mostly ARM) and smaller transistors than traditional high-end processors. For instance, the Qualcomm Snapdragon 888 SoC released in 2021 is based on 5nm technology, while Xeon is still using ≥10nm. Moreover, each SoC can be independently managed (e.g., turned on/off and dynamic voltage/frequency scaling) to adapt to variational workloads, which is much more flexible than a monolithic powerful processor like NVIDIA GPU.

Apart from the low-power vantage, a few more incentives lead us to explore SoC-Cluster on edges. First, mobile SoCs can run mobile OSs/apps seamlessly, therefore supporting computation/code offloading from mobile devices. This characteristic makes them an ideal platform for *cloud gaming* [12, 19, 24, 26, 48], which enables wimpy or low-battery smartphones to run resource-consuming mobile games anytime and anywhere. Furthermore, mobile SoCs are highly scalable: each SoC has a standard CPU that can serve general workloads but also heterogeneous co-processors like GPU, DSP, and NPU that accelerate domain-specific workloads. A monolithic server sees diminished performance worsen with more processors as the memory/disk I/O could bottleneck; an SoC-Cluster can linearly scale up its compute capacity with the number of SoCs integrated. Last but not least, mobile SoC is still fast evolving [40], and brings together the wisdom from many leading chip companies. Building a server on those SoCs takes such free lunch. Furthermore, the software stacks on mobile SoCs and OSs are mature enough and highly optimized, e.g., deep learning inference/training [44, 75], multimedia data processing [32], or even containers [42, 51].

There have been very few, scattered thoughts on organizing mobile SoCs as servers. [101] conducts an early analysis of using mobile SoCs for HPC. Some work focuses on reusing decommissioned mobile devices to reduce e-waste [107, 114]. Other studies focus on using SoCs for very specific applications like parallel computing [55], key-value storage [50], web search [73], and video transcoding [87]. That work either contributes only theoretical analysis, carries out experiments with small-scale toy implementation consisting of 2–8 smartphones/RPIs, or evaluates "wimpy" workloads [81]. Instead, we seek to materialize the "SoC-as-server" concept in a more realistic and industry-level context.

Hardware prototyping and commercialization. We have developed a concrete SoC-Cluster server, which integrates 60 Qualcomm Snapdragon 865 SoCs into a 2U rack. The SoCs are attached to 12 independent PCBs with dedicated power and network supplies. The detailed specifications of our SoC-Cluster are discussed in §2.2. Within two years, we have manufactured more than 10,000 such SoC-Clusters, most of which go to edge service providers. Those deployed SoC-Clusters are mainly used to serve cloud gaming workloads. However, according to monitored traces (§2.3), utilization of those deployed SoC-Clusters varies widely and is low on average. Apparently the potential of those SoC-Clusters is far from being fully realized. To fill the gap, the very first but critical step is to understand whether SoC-Clusters can efficiently serve other applications other than mobile cloud gaming.

Measurement methodology. Therefore, we present a first-of-its-kind measurement study to quantitatively disclose *how efficiently SoC-Cluster can support typical edge workloads* on our COTS SoC-Cluster machine. The study is application-driven, and we focus on two types of modern and computation-intensive workloads: video transcoding and deep learning (DL) serving. The former is the de-facto dominant workload on edges [122], supporting applications like live streaming, online conferences, and content archives. The latter is the key building block of many intelligent applications like AR/VR and autonomous driving. It is also one of the hottest research topics in the edge computing field [108, 119, 131]. In the foreseeable future, those two applications will be the major hardware resource consumers of edge servers, making it critical to demystify their performance on SoC-Cluster. For comparison, we used a typical edge server comprised of dual Intel Xeon Gold 5128R CPUs and 8 NVIDIA A40 GPUs. To expand the results to more SoC models, we also tested on smartphones with five different high-end Qualcomm SoCs.

Benchmark suite. We built an automatic benchmark suite to test application performance on both SoC-Cluster and the conventional edge server. The benchmark suite leverages state-of-the-art libraries for each application. For video transcoding, we used FFmpeg [22] to process six videos judiciously selected from vbench [90] with disparate characteristics. For DL serving, it employed TFLite [44] (SoC-Cluster), TVM [56] (Intel CPU), and TensorRT [35] (NVIDIA GPU). The NN models used were ResNet-50, ResNet-152 [69], YOLOv5x [77], and BERT [18]. The reason we sometimes used different software stacks on different hardware platforms is that there is no single software that can perform well on heterogeneous processors. The benchmark reports comprehensive metrics (throughput, latency, energy consumption) under various constraints (physical size, electricity, cost). It also reports application-specific metrics such as video quality and bitrate.

Key findings are summarized below.

(1) SoC-Cluster, and especially its co-processors (GPU, DSP, and hardware codec), shows superior energy efficiency than conventional servers. In DL serving, the per-Joule throughput on SoC-Cluster is up to 42× and 2.3× higher than Intel CPU and NVIDIA GPU, respectively. This means that SoC-Cluster consumes less energy to process each sample. In live streaming transcoding, the per-stream energy saving is even up to 17.1×/13× respectively. SoC-Cluster also proportionally scales its energy consumption with dynamic loads. For instance, when the number of live video streams decreases from 20 to 5, energy efficiency incurs negligible degradation on SoC-Cluster but decreases by an average of 3.3× on the NVIDIA GPU.

(2) The throughput of SoC-Cluster per rack unit is noticeably higher than conventional servers in video transcoding services but lower in DL serving. The maximal number of live video streams supported by SoC-Cluster is 60–900 depending on video characteristics, which equals 102 Intel CPU cores or 14 NVIDIA A40 GPUs on average. Note that our SoC-Cluster fits into a 2U-rack case, which can typically only hold 4–8 GPUs. In DL serving, SoC-Cluster delivers a throughput of 321 FPS on the ResNet-152 FP32 model and 5,870 FPS on the INT8 model, which equals 84/191 Intel CPU cores, 0.75/0.79 NVIDIA A40 GPU, or 0.38/0.48 NVIDIA A100 GPU.

(3) Lacking software support for cross-SoC collaboration, SoC-Cluster struggles to handle delay-critical workloads. For a medium-sized DNN model (ResNet-50), the inference latency of SoC-Cluster is low enough to meet the requirements of most edge applications (8.8ms after quantization). On large models like YOLOv5x, however, the latency can be up to hundreds of milliseconds. It urgently demands a DL library to enable collaborative inference on multiple SoCs.

(4) Based on the total cost of ownership analysis, SoC-Cluster delivers higher throughput per cost than the traditional edge server for live streaming transcoding. But for DL workloads, NVIDIA GPU shows significantly higher throughput per cost than SoC-Cluster. Such a result could negatively impact the intention of purchasing new SoC-Clusters to serve DL workloads. However, we advocate that it's still beneficial to migrate parts of DL workloads to the deployed, under-utilized SoC-Clusters for higher energy efficiency on edges.

(5) Through a longitudinal study, we find mobile SoC has shown a tremendous performance increase in the past five years. Specifically, the mobile CPU performance evolution is slowing, but

the capacity of mobile co-processors (Adreno GPU and hardware codec) is still fast growing and their benefits over CPU are increasing. The improvement of embedded GPU over time is even more profound than high-end NVIDIA GPU in DL serving. To achieve sustained performance evolution on SoC-Cluster, it is critical to improve the existing software stack to fully harness mobile co-processors.

In a nutshell, SoC-Cluster has pros and cons, which also vary across different workloads. The most promising workload is live streaming transcoding, for which SoC-Cluster delivers not only higher energy efficiency but also higher throughput and cost reduction. For archive video processing and DL serving, SoC-Cluster sometimes shows attractive improvements in energy efficiency but has lower throughput and cost efficiency than data-center-level GPUs. Nevertheless, we believe our results point to a promising future for SoC-Cluster for: (i) video transcoding services such as online meeting (Skype and Zoom) and live streaming (Twitch and TikTok) applications; and (2) general-purpose edge platforms whose workloads cannot always be accelerated by GPUs. In such scenarios, the huge number of CPU cores in SoC-Cluster (~500 in 2U) is much more powerful than conventional CPU servers. This is also demonstrated through our micro-benchmarks in §2.2.

Contributions. In this work, we discuss the rationales and possibilities of organizing mobile SoCs as edge servers. We present our first-of-its-kind hardware prototyping of this concept and how it has been commercially deployed on a large scale by edge service providers. To disclose the performance of this SoC-Cluster, we built a benchmark suite that leverages state-of-the-art libraries for two killer edge workloads, i.e., video transcoding and DL serving. We then performed a comprehensive measurement study and compared SoC-Cluster with a traditional edge server (Intel CPU and NVIDIA GPU) in terms of physical server size, energy efficiency, and monetary cost. The results reveal both advantages and disadvantages of SoC-Cluster, especially its high energy efficiency, and provide valuable guidance to fully unleash its power in edge scenarios.

The remainder of the paper is organized as follows. We provide the motivation for this paper and the very first revelation of SoC-Cluster in §2. The measurement methodology and metrics are illustrated in §3. The video transcoding and DL serving performance are shown in §4 and §5, respectively. We give the total cost of ownership analysis in §6. §7 further presents the SoC performance evolution over the years. We then show the implications and discussion in §8, related work in §9, and our conclusions in §10.

2 BACKGROUND AND MOTIVATION

2.1 The Status Quo of Edge Servers

Edge servers provide computing and storage resources in proximity to end-users and devices. They can be deployed as a micro data center in each city, or sunk into cellular stations and buildings/houses [15]. With the increasing demands from ultra-low-delay applications like AR/VR and autonomous driving, edge computing is expected to play a critical role in future IT infrastructures.

The current edge server architecture still inherits the legacy of cloud computing that has been established for tens of years. Generally speaking, a server is organized as a many-core CPU plus a series of domain-specific accelerators (GPUs, TPUs, FPGAs, etc.). The largest edge resource providers worldwide, including Azure, AWS, and Alibaba, all follow this tradition [11, 28, 47].

However, there are two notable differences between the edge and the cloud. (1) **Electricity supply.** Data centers, either centralized or distributed, are extremely power-hungry [62]. Therefore, data centers are typically built close to a green electricity supply such as hydro power. Instead, edge server deployment is limited to certain spots where the power supply is much more constrained and costly [99]. (2) **Workload dynamics.** A recent empirical study shows that edge servers undergo much more dynamic workloads [122]. This is mainly attributed to the unique applications served

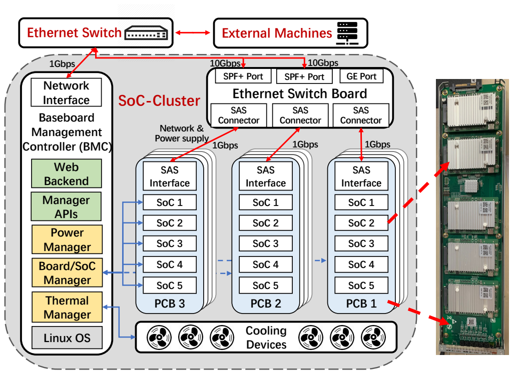


Fig. 1. Architecture of SoC-Cluster and network topology setup of this work.

Hardware	SoC-Cluster (per-SoC)	Traditional Edge Server
CPU	Octa-core (1 x 2.84 GHz Kryo 585 + 3 x 2.42 GHz Kryo 585 + 4 x 1.8 GHz Kryo 585)	Intel Xeon Gold 5218R Processor @ 4.0GHz (40 cores/80 threads)
GPU	Qualcomm Adreno 650	8 x NVIDIA A40 PCIe 48GB
Memory	12GB LPDDR5	4 x Samsung M393A4K40CB2-CVF 32GB DDR4 + 10 x SK Hynix HMAA8GR7AJR4N-XN 64GB DDR4
Disk / Flash	SK Hynix HN8T15BZGKX016 UFS 3.1 256GB	2 x Samsung MZ7LH960 SSD 960GB + 5 x Seagate ST6000 HDD 6TB
OS	Android 10 (Kernel 4.19.81)	Ubuntu 18.04 LTS (Kernel 5.4.0)
Network Interface	1 x 1GE RJ45 Port (1Gbps) + 2 x 10GE SPF+ Port (2 x 10Gbps)	2 x 1GE RJ45 Port (2 x 1Gbps) + 2 x 10GE RJ45 Port (2 x 10Gbps)
Physical size	2 Rack Unit	4 Rack Unit

Table 1. The two major hardware platforms used in our measurements. Additionally, we also used an NVIDIA A100 GPU from a GCP server in our DL serving experiments (§5) and four more SoC models in our longitudinal study (§7).

by edges that are mostly user-oriented. Such high dynamics complicate hardware and software design, e.g., how to scale energy consumption with workloads.

The traditional cloud server architecture is not designed for the above characteristics. While our research community has realized this issue and proposed various software-level optimizations [83, 92, 106], we advocate it's time to reconsider the architecture of edge servers. More specifically, in this work, we explore if it's feasible and beneficial to organize many low-end SoCs as an edge server.

2.2 A glance at our SoC-Cluster

The design space is huge in materializing the concept of an SoC-Cluster into a server machine. In this work, we use specific server architecture that we have built and commercialized with success. This type of SoC-Cluster has been densely deployed in the wild, as will be discussed in §2.3, and so is representative of the SoC-Cluster status quo. For simplicity, in the rest of the paper, we still refer to this particular server as SoC-Cluster. In this subsection, we look into its architecture, internal network/disk performance, power consumption, and running results with a list of micro-benchmarks.

Server architecture. Figure 1 shows the architecture of the SoC-Cluster used in our measurements. The major component of the server is a pool of 60 Qualcomm Snapdragon 865 SoCs [3]. Each SoC contains an Octa-core CPU, an Adreno 650 GPU, a Hexagon 695 digital signal processor (DSP), and a 12 GB LPDDR5 DRAM. The detailed hardware/OS specifications per SoC are summarized in Table 1. The server is organized as 12 PCBs where each PCB integrates 5 SoCs. Each PCB has its dedicated power supply and network capabilities (served as a switch) for SoCs attached to it. Network interface bandwidth between SoC and PCB is constrained to 1 Gbps. Plugging PCB into SoC-Cluster will establish a physical connection with the Ethernet Switch Board (ESB) in SoC-Cluster. The network capacity between each PCB and ESB is also 1 Gbps. The ESB is responsible for exposing all SoCs for external access through its single RJ45 interface (1 Gbps) or dual SPF+ interfaces (2×10 Gbps). In our case, we use an external network switch (10 Gbps) to connect the external machine and SoC-Cluster through their SPF+ interfaces to ensure our benchmarks will not be bounded by network throughput. SoC-Cluster also uses a Baseboard Management Controller (BMC) to manage and monitor the server status (e.g., power, temperature, and fans).

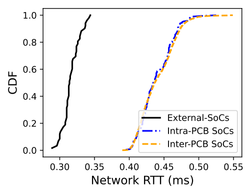


Fig. 2. RTT.

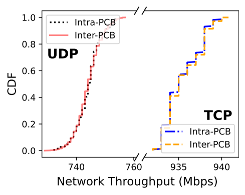


Fig. 3. Throughput.

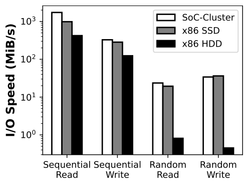


Fig. 4. Disk I/O.

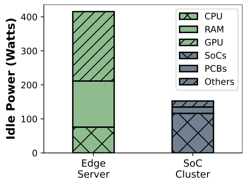


Fig. 5. Idle power.

Network capacity. We use `ping` to measure the network latency and `iPerf3` to measure the network throughput between SoCs as well as SoCs in SoC-Cluster and the external machine. As shown in Figure 2, the RTT between SoCs, either in the same PCB or across PCBs, is around 0.44ms, which is a little bit higher than the RTT between an SoC and the external machine (0.32ms). It is likely attributed to the defective implementation of network interfaces on PCBs. Figure 3 shows the TCP and UDP throughput. We observe that the bandwidth between intra-PCB and inter-PCB SoCs is almost identical, which is bounded by the underlying network interfaces of SoCs. The average bandwidth is quite stable around 935 Mbps and 744 Mbps with TCP and UDP, respectively. In summary, the network throughput of each SoC is lower than the 25/40/100 GbE networks in data centers today. To fully release its power for applications that need cross-SoC orchestration, we anticipate an enhanced network design at both the hardware and software levels.

Disk I/O. Like mobile devices, each SoC in SoC-Cluster is attached to a 256 GB Sk-Hynix flash storage (UFS 3.1), so the total storage capacity of SoC-Cluster is 15.36 TB. The disk I/O speed could be crucial for data-intensive applications [127, 132], so we compare that of a single SoC with HDD (Seagate ST6000 [20]) and SSD (Samsung MZ7LH960 [5]), commonly used in traditional edge servers. All tests were performed using `fio` [16] in synchronous mode. The block is 1 MB for sequential read/write and 4 KB for random read/write. As shown in Figure 4, the sequential read speed of SoC-Cluster is 1,733 MiB/s, which is 1.76× and 4.1× faster than SSD and HDD, respectively. For random read/write, SoC-Cluster shows similar performance levels to the current enterprise SSD but significantly outperforms HDD in edge servers. In summary, SoC-Cluster inherits the high-performance disk from mobile devices because mobile apps' launching performance heavily relies on the disk I/O speed [66, 96, 97, 126]. However, unlike traditional servers where the disks can be plugged in/out flexibly, the flash storage is integrated into each SoC, making expansion or repair difficult.

Idle power consumption. While we will quantify the detailed energy efficiency of SoC-Cluster for specific applications in later sections, here we report the idle power consumption as well as its breakdown for SoC-Cluster and the traditional edge server. By manually turning on and off a single SoC and PCB in SoC-Cluster, we can get the power breakdown result. The overall power consumption of SoC-Cluster is read from its BMC's API (through PMBus protocol). We read CPU and RAM power on our edge server using `turbostat`, which leverages the Intel RAPL system interfaces [4]. The GPU power is read through the `nvidia-smi` command. We only consider the power contributed by these three types of hardware, which is sufficient to draw a conclusion. The results are in Figure 5. The average power consumption of SoC-Cluster in idle mode is 153 Watts, which is 2.7× smaller than the 8×GPU server. Even without the GPUs, the traditional server consumes more idle power than SoC-Cluster. This is mainly because mobile SoCs are designed with more advanced transistor technology for low-power scenarios. For SoC-Cluster, the SoCs contribute most of the idle-time power consumption (76.2%). It should be noted that the power consumption of all PCBs excludes that of the SoCs.

Micro-benchmark metrics	Per-core performance		Whole server performance	
	SoC-Cluster	Traditional edge server	SoC-Cluster	Traditional edge server
CPU score	911	840	194,100	15,450
Integer score	842	800	184,500	16,224
Floating score	948	886	191,820	15,793
Text compression	4.42 MB/s	4.08 MB/s	810 MB/s	135.5 MB/s
SQLite	257 Krows/s	249.3 Krows/s	60.6 Mrows/s	9.24 Mrows/s
PDF rendering	38.1 Mpixels/s	41.1 Mpixels/s	11.406 Gpixels/s	709.9 Mpixels/s
Speech recognition	25.2 Words/s	26.2 Words/s	3,234 Words/s	259.2 Words/s

Table 2. A list of micro-benchmark (from Geekbench 5 [23]) results on our SoC-Cluster and the traditional edge server shown in Table 1.

Micro-benchmarks. We use a cross-platform benchmark, Geekbench 5 [23], to perform a list of micro-benchmarks on both SoC-Cluster and the traditional edge server. The results in Table 2 provide the following major observations. First, the per-core performance of SoC-Cluster is close to the traditional Intel Xeon CPU on both computation-intensive and I/O-intensive workloads. Second, from the whole server's perspective, the large number of SoCs makes SoC-Cluster much more powerful than traditional CPU servers, e.g., 12.6× CPU cores, 16.1× PDF rendering speed, and 12.5× speech recognition throughput. SoC-Cluster's workload throughput can linearly scale with the number of integrated SoCs, as each SoC has its own memory and storage. For a monolithic multi-core server, however, the performance gain from increasing the core number could be diminished by the memory/disk I/O speed.

These benchmarks are CPU-only and do not reveal the end-to-end performance of typical edge workloads. The major contribution of this work, as will be presented below, is a comprehensive measurement study on two typical edge applications.

2.3 SoC-Clusters in the wild

The SoC-Cluster used in this work has been on sale for more than a year. Up to September 2022, more than 10,000 had been shipped and deployed in the wild. The majority of customers are edge service providers. From collaborating with a leading worldwide edge service provider, we know that *deployed SoC-Clusters are mostly used to serve one particular application: cloud gaming*, as Android-native games like Genshin Impact [25] can be seamlessly supported by the mobile SoC without any revision. The cloud gaming deployed on edges enables wimpy or low-power devices to run resource-hungry games with a good user experience. Developers are now able to access the resources of those SoC-Clusters through either a virtualized or native context, just like traditional cloud services. In total, our shipped SoC-Clusters are estimated to serve millions of game sessions per day.

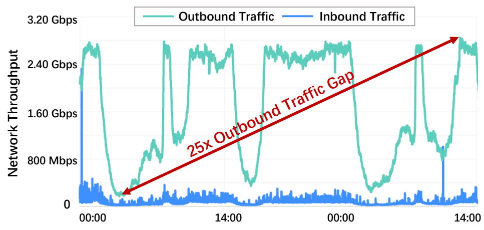


Fig. 6. The network throughput of an in-the-wild SoC-Cluster that serves cloud gaming workloads over 38 hours. The server is randomly picked from an edge site of our collaborator. Full network capacity of SoC-Cluster: 20 Gbps.

However, according to the runtime traces collected from our customers' deployed SoC-Clusters, we conclude that *those servers experience workloads with relatively low utilization and high dynamics.* In general, the resource usage of all deployed SoC-Clusters is below 20%. Figure 6 shows the temporal network traffic of one specific SoC-Cluster randomly sampled from real-world edge sites. As observed, the gap between its highest and lowest outbound traffic is up to 25×. Such an observation is consistent with a recent empirical study on large-scale edge clouds [122]. The reason is that edge applications are mostly user-centric and thus highly relate to user activities. Motivated by this observation, this work explores how efficiently SoC-Cluster can serve more general workloads beyond cloud gaming.

App performance	Condition constraint	Comparison metric	Application/Section
Max. throughput	Energy (Joule)	Throughput per energy (TpE)	Video (§4.2), DL (§5.3)
	Server rack size (U)	Throughput per size (TpS)	Video (§4.1), DL (§5.1)
	Monthly cost ($)	Throughput per cost (TpC)	Video, DL (§6)
Latency	Batch size	Model inference latency	DL (§5.2)
Total power consumption	Load level	Power consumed per second	Video (Live) (§4.2), DL (§5.3)
Energy proportionality	Load level	Throughput per energy (TpE)	Video (Live) (§4.2), DL (§5.3)
PSNR	-	PSNR	Video (Archive) (§4.4)
Bitrate	-	Bitrate	Video (Archive) (§4.3)

Table 3. Application performance and comparison metrics for the edge scenarios used in this work.

3 METHODOLOGY AND BENCHMARK

In this work, we perform an application-driven measurement study to demystify the performance of SoC-Cluster and traditional edge servers. This section elaborates on how we set up the experiments for a fair comparison with best efforts.

Applications. We select two modern and computation-intensive applications. (1) *Video transcoding* [120] converts the format (FPS, resolution, etc.) of a given video stream, widely adopted in applications such as online conferences and live streaming. It is reported to be a dominant workload at the edge [122]. We identify two scenarios for this workload (live streaming transcoding and archive transcoding) and illustrate their characteristics in §4. (2) *DL serving* is the vital building block of intelligent applications like AR/VR and autonomous driving [36, 43, 45]. A DL serving system takes a stream of input data sent from end devices and executes it with a specified DL model. Academia has invested tremendous effort into optimizing DL serving performance [59, 64, 98, 129].

In summary, the above applications cover the interests of both industry and academia, representing the major workloads on edges now and in the near future. They are both resource-hungry, indicating that, if SoC-Cluster can serve those applications well, it's likely that it can serve others well too.

Metrics. Table 3 summarizes the metrics used in this paper, which are divided into two categories: application performance and comparison metrics. The application performance metrics are mainly about throughput (processed samples/videos per second), latency (model inference time), and power consumption while serving the workloads. To make a relatively fair comparison between SoC-Cluster and the traditional edge server, we further judiciously select three condition constraints by considering the following aspects:

- *Energy.* As mentioned previously, energy has become a killing constraint for edge sites. Thus, we use throughput per Energy (TpE) to compare application energy efficiency. Besides the energy efficiency under a given (often full) load, we also care about the energy proportionality under various load levels since edge servers experience high load variations. An ideal edge server should be able to proportionally scale its power consumption with the load to minimize wasted energy.
- *Rack size.* Edge sites often have space constraints (e.g., deployed in cell sites). Therefore, delivering higher throughput within a given rack size is important. We normalize the application throughput to the server rack size (TpS) to reveal the per-rack unit performance.
- *Cost.* We perform a total cost of ownership (TCO) analysis to show the relationship between application performance and monthly TCO, which might help edge operators to make purchase/scheduling choices clearly. Section 6 presents our TCO analysis results.

Hardware. The SoC-Cluster server used in measurements was introduced in §2.2. For comparison, we used a traditional server with an Intel Xeon Gold CPU (4.0 GHz and 40 physical cores), 8× NVIDIA A40 GPUs, and 768 GB DRAM. The OS was Ubuntu 18.04 LTS. Server hardware specifications are summarized in Table 1. We confirmed that this type of server is widely used in the edge sites deployed by our collaborator. Additionally, we use a more high-end NVIDIA A100 GPU from the Google Cloud Platform in our DL serving experiments for a more comprehensive comparison. Both the NVIDIA A40/A100 and the Intel CPU were released in the same year (2020) as the Qualcomm Snapdragon 865 SoC that makes up our SoC-Cluster. We did not use an NVIDIA A100 GPU for video transcoding experiments due to the lack of support for NVENC on NVIDIA A100 as of September 2022 [46]. In §7, we also experiment with four more SoC models to understand the evolution of SoC performance over time.

Benchmark suite. One challenge of our measurement study is the software stack selection. Given the inherently different hardware architectures and their heterogeneity, we found there is rarely software that is compatible with each processor (i.e., SoC CPU/GPU/DSP, Intel CPU, and NVIDIA GPU). If such software exists, its performance could be far from the state of the art. To obtain meaningful results, we consulted our industry partners and also tested with common software. We then selected the best-performing option for each workload and hardware type.

For video transcoding, we used FFmpeg (v4.4) [22] with libx264 [49] and NVDEC/ENC [37] support. We cross-compiled FFmpeg to SoC-Cluster with ARMv8 NEON acceleration. One exception is that FFmpeg has poor support for the hardware codec of Qualcomm SoCs. Thus, we used a popular open-source Android library, LiTr [30]. We built our benchmark on vbench [90], a widely used benchmark tool for cloud video transcoding.

For DL serving, we used TFLite [44] on SoC-Cluster, TVM [56] for Intel CPU, and TensorRT [35] for NVIDIA GPU. We selected one medium-sized DNN (ResNet-50 [69]) and three large DNNs (ResNet-152 [69], YOLOv5x [77], and BERT [18]), which are representative for DL serving workloads. All software was carefully tuned and orchestrated into the benchmark suite for automatic, flexible testing on both SoC-Cluster and the traditional edge server.

Measuring power consumption. Power consumption is measured through the software-level APIs, as described in §2.2. Note that the reported power consumption of SoC-Cluster obtained through its BMC's API includes not only SoCs but also the PCBs and cooling devices (i.e., fans). The workload power consumption report excludes idle power consumption, as measured in §2.2. By default, we always carry out our experiments with hardware fully loaded, e.g., using batched DL serving or many transcoding processes of live video streams. Only when testing the energy proportionality of different hardware do we vary the load levels.

Setups. For Intel CPU experiments, we split the 80 cores (80 hardware threads [80]) into 10 separate 8-core docker containers. We selected 8 cores for two reasons. (1) Based on previous edge workloads [122], 8 is the median number of vCPU cores for edge IaaS VMs and is enough for serving most edge services. (2) The SoC's embedded CPU also contains 8 cores, making the comparison more straightforward. Additionally, to avoid the docker start-up overhead, we kept the container process in daemon and invoked test commands inside the container. Furthermore, to mitigate power fluctuations during experiments: (1) for live streaming transcoding, we simultaneously started the maximum number of streams supported by each hardware, ensuring that no stream degraded its transcoding FPS below the origin stream's; (2) for archive transcoding, we repeatedly transcoded the same video ten times; (3) for DL serving, we set the DL inference iteration to 1,000 for each test and then got the average power consumption for each frame.

4 VIDEO TRANSCODING

In this section, we compare the video transcoding performance in two scenarios: live streaming transcoding and archive transcoding. The former takes video streams with a constant frame rate, widely used in live streaming and online conferences. The latter processes a video clip from file storage, often used as an uploading stage before delivering it to the video content provider [41, 90]. The computing complexity of video transcoding leaves a lot of differences in transcoding parameters between live streaming and archive transcoding. Following the prior video benchmark [90], we

Video	Video Metadata					Network Bound Analysis		
	Resolution	FPS	Source Entropy	Source Bitrate	Target Bitrate	Max. Stream Num (per SoC)	Max. Network BW/usage (per PCB, 1 Gbps)	Max. Network usage (whole server, 20 Gbps)
V1 - holi	854x480	30	7.0	2.8 Mbps	819.8 Kbps	13 (CPU) / 16 (HW)	534 Mbps (53.4%)	6,407 Mbps (32.0%)
V2 - desktop	1280x720	30	0.2	181 Kbps	90.5 Kbps	15 (CPU) / 16 (HW)	43 Mbps (4.3%)	505 Mbps (2.5%)
V3 - game3	1280x720	59	6.1	5.6 Mbps	2.7 Mbps	4 (CPU) / 12 (HW)	673 Mbps (67.3%)	8,072 Mbps (40.3%)
V4 - presentation	1920x1080	25	0.2	430 Kbps	215 Kbps	9 (CPU) / 16 (HW)	81 Mbps (8.1%)	968 Mbps (4.8%)
V5 - hall	1920x1080	29	7.7	16 Mbps	4.1 Mbps	3 (CPU) / 7 (HW)	1,008 Mbps (100.8%)	12,010 Mbps (60.5%)
V6 - chicken	3840x2160	30	5.9	49 Mbps	16.6 Mbps	1 (CPU) / 2 (HW)	985 Mbps (98.5%)	11,821 Mbps (59.1%)

Table 4. The metadata of the tested video in our transcoding experiments and the network bound analysis. The videos were carefully picked from vbench [90] to ensure diverse coverage of resolution, FPS, and entropy. Entropy is calculated by bits per pixel per second and thus relates to the scene complexity. The network analysis included both inbound and outbound traffic. CPU and HW represent live streaming transcoding using embedded CPU and hardware codec of SoC-Cluster.

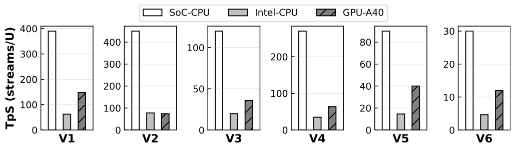


Fig. 7. Live streaming transcoding throughput per rack unit.

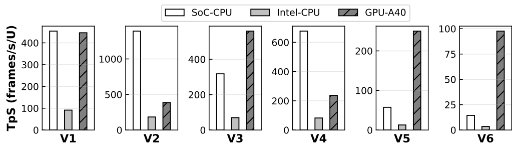


Fig. 8. Archive transcoding throughput per rack unit.

kept the same target bitrate for live streaming transcoding and the same output video quality for archive transcoding. The metadata of videos used in this section are shown in Table 4.

4.1 Transcoding throughput

For live streaming transcoding, since the video streams are ingested at a constant frame rate, we use the maximum video stream number that can be supported by each hardware to indicate the transcoding throughput. For archive transcoding, a video clip with a fixed duration needs to be processed as fast as possible, therefore the throughput is calculated by how many frames can be processed per second. The results of live streaming and archive transcoding are illustrated in Figures 7 and 8, respectively. We make the following key observations.

(1) For live streaming transcoding, embedded CPUs always provide higher transcoding throughput than both the Intel CPU and the NVIDIA GPU. For example, SoC-Cluster can transcode 390 V1 streams simultaneously, which is 6.29× more than the Intel CPU and 2.64× more than the NVIDIA GPU per rack unit on average, respectively. The advantage of SoC-Cluster over the Intel CPU comes from its denser CPU cores. More specifically, the Octa-core CPU (four big cores and four LITTLE cores) of a single SoC can support timely transcoding of nine low-complexity 1080p video (V4) streams and three high-complexity 1080p video (V5) streams, which is slightly fewer than eight Intel CPU cores (14 and 5). However, the total CPU cores of an SoC-Cluster is 6× more than our Intel CPU server. The throughput of the NVIDIA GPU is always bounded by its hardware encoder, as observed, leaving most of its general-purpose processor under-utilized. Indeed, improving video transcoding performance is not a primary design goal of mainstream GPUs [34].

(2) For archive transcoding, embedded CPUs also outperform the Intel CPU but underperform against the NVIDIA GPU for high-complexity videos. Among all videos, embedded CPUs provide 4.14×–8.22× higher throughput compared with the Intel CPU. For low-complexity videos (low resolution or low entropy) that may not exhaust hardware resources, i.e., V1, V2, and V4, embedded CPUs can also perform better than the NVIDIA GPU by 1.02×, 3.61×, and 2.85×, respectively. However, on more complex videos, the NVIDIA GPU shows a much higher transcoding throughput. The reasons are twofold. First, the increasing video complexity demands more computations from the encoder paid to the hard searching problems [113], which is well optimized on the NVIDIA GPU with its high clock frequency. Second, due to the different processing capacities of the embedded CPU's big.LITTLE cores, simply increasing the encoding threads to leverage all cores

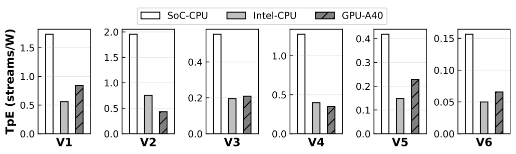


Fig. 9. Live streaming transcoding energy efficiency.

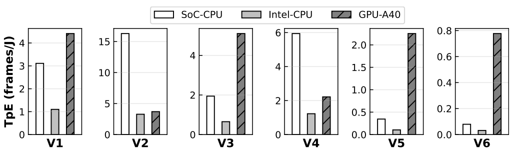


Fig. 10. Archive transcoding energy efficiency.

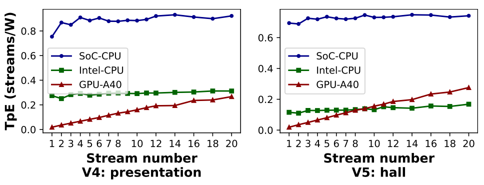


Fig. 11. Energy efficiency of live streaming transcoding with different numbers of live video streams being processed simultaneously.

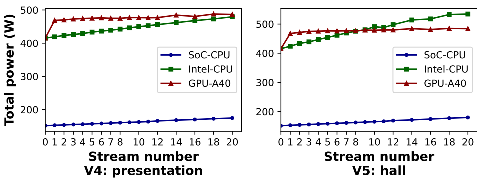


Fig. 12. Total energy consumption of the whole server with different numbers of live video streams. The y-axis stream number "0" represents the server's idle power.

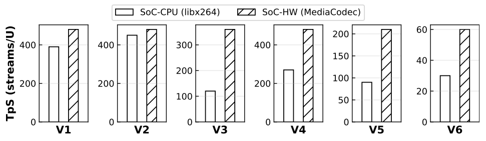
causes a synchronization overhead and cannot accelerate the transcoding speed. This calls for a heterogeneity-aware video transcoder that can fully utilize the asymmetric embedded CPU cores to accelerate archive video transcoding.

4.2 Energy efficiency

For live streaming transcoding, energy efficiency is measured as how many live video streams can be supported per Watt (streams/W). For archive video transcoding, energy efficiency is measured as how many frames can be processed per Joule (frames/J). The following results are performed within the hardware processing capabilities (e.g., without 8-core constraint on the Intel CPU). Results are shown in Figures 9 (live streaming transcoding) and 10 (archive transcoding). Our key observation is that *SoC-Cluster shows significant energy savings in all live streaming and some archive video transcoding tasks as compared to the Intel CPU and the NVIDIA GPU.*

For live streaming transcoding, embedded CPUs in SoC-Cluster are 2.58×–3.21× more energy efficient than the Intel CPU, and 1.83×–4.53× more energy efficient than the NVIDIA A40 across different videos. In archive transcoding, embedded CPUs are always more energy efficient than the Intel CPU, but the comparison with the NVIDIA GPU varies across videos. More specifically, the NVIDIA GPU is less energy efficient on V2 and V4. We find the common feature of V2 and V4 is that they have low entropy (e.g., low motion or infrequent scene transitions) and therefore require fewer encoding computations. For those videos, we observe that the NVIDIA GPU stays in a high-power mode (high clock frequency) while embedded CPUs can complete such tasks with only a little CPU usage. The same conclusion – that SoC-Cluster shows higher energy efficiency on low-complexity videos – can also be proven in the live streaming scenario: the relaxed video quality constraint makes it consume less power than the Intel CPU and NVIDIA GPU.

We also show how energy efficiency scales with dynamic workloads at the edge, which is represented by various live video stream numbers in Figure 11. We used two 1080p videos with diverse scene complexity and present the relationship between per-Watt throughput (TpE) and processed stream numbers simultaneously. Results of the two videos show the same trend: embedded CPUs and the Intel CPU show nearly constant energy efficiency for launching each transcoding stream, indicating a linear power increase when the workload increases. However, the NVIDIA GPU can only process 0.018 live video streams per Watt for transcoding only one video (V4), which is 14.9× less than the Intel CPU and 40.8× less than embedded CPUs. With the increasing stream number, the GPU's energy efficiency gradually increases but is still lower than that of


(a) Transcoding throughput per rack unit.

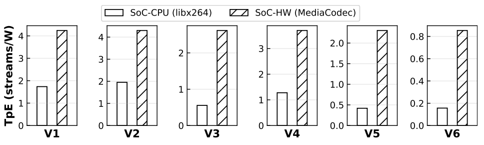

(b) Energy efficiency.

Fig. 13. Live streaming transcoding performance of embedded CPU and hardware codec on SoCs.

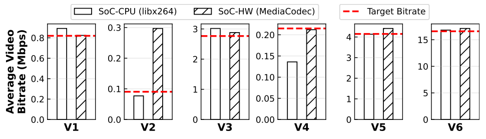


Fig. 14. Target bitrate and output bitrate of live streaming transcoding.

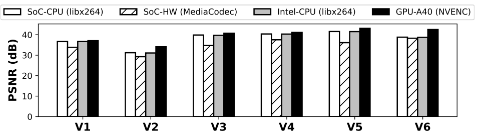


Fig. 15. Live streaming transcoding quality of different encoders with the same bitrate constraint.

embedded CPUs. The fine-grained usage control on embedded CPU cores or SoCs shows better energy scalability when serving dynamic video transcoding workloads on SoC-Cluster.

The traditional edge server always consumed more energy than SoC-Cluster, as shown in Figure 12, mainly due to its high static power draw. The NVIDIA GPU also showed a power consumption spike with light incoming workloads, but its power consumption increased slowly and might be less than performing the same workloads on the Intel CPU. Taking V5 as an example, the total power consumption of the traditional edge server reached 467 Watts when there was one transcoding stream using the NVIDIA GPU, while only increasing to 424 Watts using the Intel CPU. The gradually increasing processed stream number makes the server power consumption using the Intel CPU more than that of using the NVIDIA GPU (e.g., more than 9 V5 streams). The above results also provide insightful scheduling trade-offs when there are only traditional edge servers to serve video transcoding workloads.

4.3 Hardware-accelerated video transcoding on mobile SoCs

Mobile SoCs are equipped with hardware codecs that handle most of the video encoding/decoding on smartphones, e.g., the videos captured from cameras. We only tested their performance for live streaming transcoding, because the APIs exposed by Android MediaCodec [32] cannot control the video quality for a fair comparison in archive transcoding. We used LiTr [30] to perform hardware-accelerated video transcoding instead, as FFmpeg only supports hardware decoding but not encoding for Android [27]. We compare the performance and transcoding behavior of the hardware codec with embedded CPU in terms of the throughput per rack unit (TpS), energy efficiency (TpE), output bitrate, and Peak Signal-to-Noise Ratio (PSNR) [71] below.

Our key observation is that *the hardware codec of the mobile SoC provides a further significant improvement over its CPU.* As Figure 13a shows, using the hardware codec increases the maximal number of supported live video streams per rack unit by 1.07×–3×. The increase in energy efficiency is even more impressive, as shown in Figure 13b. For videos with low-complexity scenes (V1, V2, and V4), hardware transcoders in SoC-Cluster could support on average 2.5× more streams per Watt than embedded CPUs. When transcoding high-entropy and high-resolution videos (i.e., V3, V5, and V6), embedded CPUs will consume more power for encoding, while delegating transcoding workloads to the hardware codec can significantly improve energy efficiency by 4.7×–5.5×.

To demystify whether the hardware codec in SoC-Cluster could handle live streaming transcoding workloads or not, we also present the output video bitrate, which is one of the most critical metrics affecting user experience. As shown in Figure 14, we use red dash lines to represent the target bitrate for each transcoding task. The key observation is that, *in most cases, the hardware codec* *can meet the bitrate constraint, but it's hard for it to meet a relatively low bitrate cap.* For example, setting the target bitrate to 90.5 Kbps for V2 will make the encoder create a higher bitrate output (even higher than the origin video stream). Such unexpected behavior goes against the intention of most transcoding services to compress a video stream. The same behaviors were confirmed by our additional experiments on other videos by setting ultra-low bitrates. This is possibly a design trade-off decided by the mobile SoC vendor for energy efficiency and chip size.

4.4 Transcoding quality

Transcoding quality is how the output video is perceived by consumers. The live streaming transcoding experiments were performed with a fixed bitrate target. However, due to the nuances at both the software (libx264 vs. MediaCodec vs. NVENC) and hardware (CPU vs. GPU vs. ASIC) levels, their video quality could differ observably even under the same bitrate constraint. We saved the live streaming transcoding output to files during previous experiments and used PSNR to represent the video quality (higher is better).

As shown in Figure 15, *the software encoder using embedded CPUs can preserve almost the same video quality as the Intel CPU and NVIDIA GPU, while videos generated by SoC-Cluster's hardware codec have sightly poorer quality than others.* The general-purpose computing unit (i.e., the embedded CPU and Intel CPU) and software encoders with the same transcoding configurations always result in the same video quality. However, videos generated by MediaCodec have about 1.35%–14.77% lower PSNR values compared with those generated by libx264 using embedded CPUs. This is caused by the loose quality and bitrate requirements of mobile encoders [104]. To further investigate what bitrate should be set to achieve the same video quality using MediaCodec, we manually tuned the target bitrate and then compared it with the origin video to get the PSNR value. From our additional experiment, we find that simply loosening the target bitrate constraint using MediaCodec still couldn't meet the same video quality as libx264. As such, if the quality loss incurred by MediaCodec is not bearable, it's better to choose embedded CPUs for video transcoding.

4.5 Network bound analysis

If the SoC-Cluster is fully occupied with video transcoding workloads, will the network capacity become the bottleneck? Our analysis in Table 4 says not. Among the six videos tested, network usage will slightly exceed the network capacity of PCB (1 Gbps) only when fully leveraging the embedded CPUs and hardware codecs together to transcode V5 video. In practice, it is hard to fully load the embedded CPU and hardware codec simultaneously due to the software delegation daemon process or thermal control. Considering the whole SoC-Cluster, under no circumstances will the ESB (20 Gbps) bottleneck. However, with a denser SoC integration in a future-generation SoC-Cluster, the network could be a limiting factor and needs to be enhanced.

4.6 Summary

Embedded CPUs in SoC-Cluster provide both higher throughput per rack unit and greater energy efficiency than the traditional Intel CPU (up to 7.7× and 3.2× respectively) and NVIDIA GPU (up to 6.08× and 4.53× respectively) we used in experiments for live streaming transcoding. The fully fledged software stack makes it possible to seamlessly migrate current transcoding services to SoC-Cluster. Although the hardware codecs of SoC-Cluster deliver even higher throughput and energy efficiency than embedded CPUs, they sacrifice a portion of video quality (1.35%–14.77%) and cannot guarantee a relatively low bitrate. The software stack of the mobile SoC's hardware codec is also not readily integrated into current transcoding infrastructure due to a lack of flexibility. Nonetheless, the embedded CPUs show high and solid potential for general video transcoding

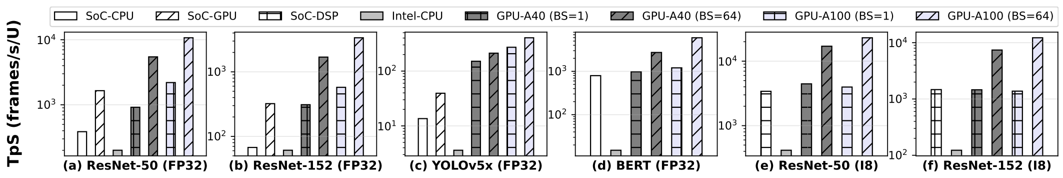


Fig. 16. Deep learning serving throughput per rack unit on SoC-Cluster and the traditional edge server.

services, and the hardware codecs provide even higher profits for a relatively narrower scope of video transcoding scenarios.

5 DEEP LEARNING SERVING

In this section, we report the DL serving performance with various hardware types and models. Besides models in floating-point format (FP32), we also tested models in INT8, which is widely used in DL serving workloads to speed up inference.

5.1 Inference throughput

Inference throughput indicates the maximal workload (and thus users/applications) that can be served by a server. It is measured as the number of samples processed by the whole server (60 SoCs, 8 A40/A100 GPUs, etc.). We used two different batch sizes (BS = 1 or 64) for the NVIDIA GPU, which affects hardware utilization. For other hardware, we only set the batch size to 1 as that is enough to fully utilize the hardware capacity. Here, we used per-rack unit throughput (TpS) as the space-constrained comparison metric between diverse hardware types. The results are shown in Figure 16.

Our key observation is that *SoC-Cluster, and especially its embedded GPUs, provides much higher throughput than CPU-only servers but less throughput than multi-GPU servers per rack unit.* SoC-Cluster's embedded CPUs/GPUs can process 386/1,648 samples with ResNet-50 and 67/321 samples with ResNet-152 (both FP32 format) per second and rack unit. This throughput is 1.11×–8.22× more than that on the Intel CPU. However, compared to NVIDIA A40 and A100 GPUs, the throughput of SoC GPUs is 3.31×–5.41× and 6.51×–10.51× smaller on ResNet-50 and ResNet-152, respectively. In other words, 60 embedded GPUs provide a processing capacity of as much as 0.9× the NVIDIA A40, 0.46× the NVIDIA A100, or 325 Intel CPU cores on average. When a smaller batch size is allowed (e.g., to guarantee low latency), the throughput of the NVIDIA A100 GPU is reduced by 432% on average because the hardware resources cannot be fully utilized. The gap between SoC-Cluster and NVIDIA GPUs is much smaller on the BERT model, mainly because TensorRT is not yet as well optimized for NLP models as for widely used CNNs.

The inference throughput of SoC-Cluster on INT8-quantized models (Fig. 16e and Fig. 16f) is even more impressive. For instance, embedded DSPs can collectively process 1,467 samples per second and rack unit with ResNet-152, which is 12.02× more than the Intel CPU. Its performance is even close to the NVIDIA A40 GPU and NVIDIA A100 GPU (BS=1). Credit for this lies with the powerful SIMD co-processor in the Qualcomm Hexagon DSP, which is designed for efficient vector operations in a fixed-point format [6]. It is claimed that the Hexagon 698 (Snapdragon 865) can provide 15 trillion TOPS [3] and the number goes up to 26 trillion on the Hexagon 780 (Snapdragon 888) [7], which was released just one year later.

5.2 Inference latency

Inference latency is also critical to delay-sensitive workloads and thus the corresponding user experience. As above, we tested this on NVIDIA GPUs with different batch sizes to trade off latency and energy efficiency but only used batch size 1 on other hardware as further increasing the batch

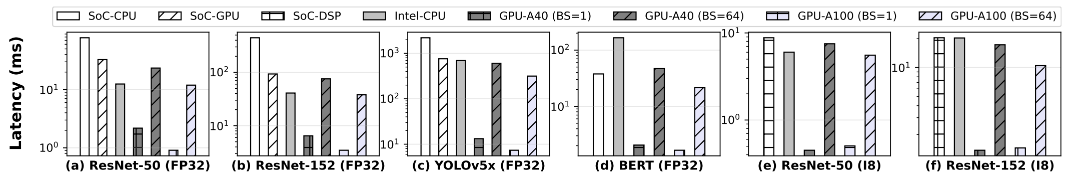


Fig. 17. Deep learning serving latency on SoC-Cluster and the traditional edge server.

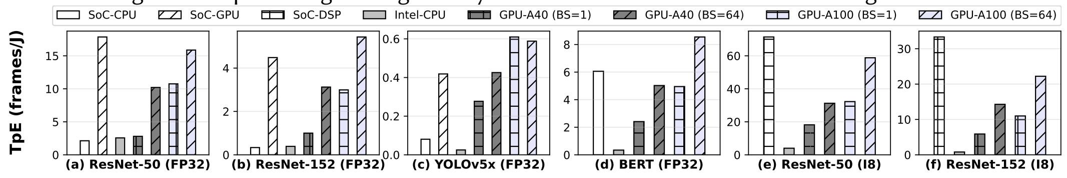


Fig. 18. Deep learning serving throughput per Joule on SoC-Cluster and the traditional edge server.

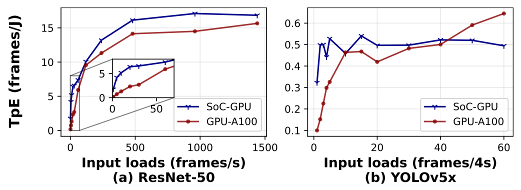


Fig. 19. Energy efficiency of SoC-Cluster and the traditional edge server under various DL input loads. SoC-Cluster delivers higher throughput per energy when workloads are light.

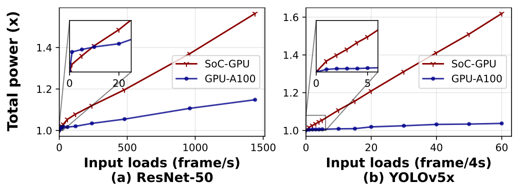


Fig. 20. Total energy consumption times to server idle power with various workload levels. SoC-Cluster can scale its power consumption well with increasing workload levels considering the low idle power draw.

size only incurred higher latency without benefiting energy efficiency. The results are shown in Figure 17.

We make the following major observations. (1) SoC-Cluster's embedded GPUs deliver 2.37×–4.77× lower latency than its embedded CPUs and are comparable to the Intel CPU with eight cores used. (2) The lowest latency provided by the NVIDIA GPU with batch size 1 is much lower than other processors due to its high hardware-level parallelism and well-optimized software (i.e., TensorRT). However, if larger batch sizes are used, the latency on the NVIDIA GPU increases significantly and could be close to or even higher than SoC-Cluster, e.g., inference using the NVIDIA A40 GPU (BS=64) on YOLOv5x (FP32) and BERT (FP32). (3) On a medium-sized DNN – ResNet-50, embedded GPUs/DSPs are often good enough to deliver satisfactory inference latency, i.e., 32.7 ms in FP32 format and 8.8 ms in INT8 format. On those models, using the NVIDIA GPU only provides a trivial increase in speed (e.g., about 8 ms on a quantized ResNet-50) but incurs much higher energy consumption, as we will discuss below. (4) On large DNNs such as ResNet-152, the inference latency of SoC-Cluster is 20.4 ms–444.8 ms, which is unbearable for real-time applications. This can potentially be addressed by a cooperative inference framework among multiple SoCs.

5.3 Energy efficiency

We used throughput per energy (TpE) (e.g., the number of samples processed per Joule) as the metric to test the energy efficiency for each hardware type. A server with higher energy efficiency can process more samples with the same electricity unit. The results are illustrated in Figure 18. Our key observation is that *SoC-Cluster is much more energy efficient than the Intel CPU and mid-end NVIDIA GPU, and is comparable to the high-end NVIDIA GPU.* Especially on ResNet-50, embedded GPUs can process about 19 frames per second and Joule, which is 6.96× more than the Intel CPU, 1.75× more than the NVIDIA A40 (BS=64), and 1.13× more than the NVIDIA A100 (BS=64). The benefit of SoC-Cluster is even more significant in quantized models. Taking ResNet-152 (INT8) as an example, the energy efficiency of embedded DSPs is 42× higher than the Intel CPU and 1.5× higher than the NVIDIA A100 (BS=64). This is because mobile DSPs are designed for low-power data processing running at ≤500MHz.

The above energy efficiency is mainly reported when the servers are fully loaded. We also measured energy efficiency when workloads varied, as in §4.2. Figure 19 illustrates this scalability of energy efficiency on different hardware types. We only used the embedded GPU and NVIDIA A100 for comparison based on their high energy efficiency on SoC-Cluster and the traditional edge server, respectively. For ResNet-50, SoC-Cluster shows a significant energy benefit when the workload is lightweight, e.g., 5.71× more energy efficient than the NVIDIA A100 GPU on average with only five samples coming per second. This is mainly because SoC-Cluster provides a much more fine-grained scheduling unit (i.e., per SoC) to process incoming requests. When incoming data can be sufficiently processed by only ten SoCs, we can keep the other 50 SoCs in a low-power state – a free lunch originally provided for mobile devices [38] – or even turned off. Instead, datacenter-level GPUs have much coarser granularity to scale its energy consumption with dynamic workloads. Figure 20 further presents the total power consumption times to the server idle power under various load levels. When workloads are light (e.g., only a few inputs per second on ResNet-50), the NVIDIA A100 GPU shows a power spike compared with its idle power. Instead, SoC-Cluster always linearly increases its total power consumption. Considering the huge gap in the static power draws of the two servers when idle, SoC-Cluster might be more suitable for serving and scaling its energy consumption with various load levels.

5.4 Summary

The embedded GPUs and DSPs on SoC-Cluster deliver superior energy efficiency than traditional server-level CPUs (up to 17.86×) or GPUs (up to 6.49×) in FP32 and INT8 formats, respectively. The inference latency of SoC-Cluster on medium-sized DNNs like ResNet-50 is satisfactory to meet the requirements of typical edge applications, but more advanced software that can orchestrate many SoCs is urgently demanded to serve large DNNs like YOLOv5x efficiently.

6 TOTAL COST OF OWNERSHIP ANALYSIS

Cost is another critical dimension to evaluate the validity of new hardware. In this section, we conduct a TCO analysis on SoC-Cluster and traditional edge servers. The TCO consists of two parts: the capital expenditure to purchase the servers (CapEx, with a breakdown of each hardware component) and the operational expenditure (OpEx). We use the retailing purchase cost as the total capital expenditure. For the OpEx, we only consider the electricity cost, as in prior work [81]. Besides the traditional edge server that mainly consists of an Intel CPU and 8 NVIDIA GPUs, we additionally add a "virtual server" by removing all 8 NVIDIA GPUs. We use throughput per cost (TpC) as the normalized performance.

Capital expenditure. As shown in Table 5, the GPUs almost dominate the total CapEx in a traditional edge server. For a CPU-only server, the CapEx is amortized into different hardware components. For SoC-Cluster, 60 SoCs and 12 PCBs constitute nearly 87% of the CapEx. In summary, SoC-Cluster has a lower CapEx than the traditional edge server with 8 NVIDIA GPUs but costs about 2.8× more than a non-GPU edge server.

Operational expenditure. We did not use complex OpEx models [107] as we observed that the amortized CapEx always dominates the TCO, as we describe later. This was also observed in a previous study from Google [52]. As a result, we only report the electricity cost here. The monthly electricity cost is calculated by the monthly power consumption (kWh) multiplied by the electricity unit cost ($/kWh). The monthly power consumption of all workloads relates to their average power usage. For example, performing live streaming transcoding when fully loading all 8

TCO Component	Parameter	Edge Server Cost	Edge Server (W/O GPU) Cost	Parameter	SoC-Cluster Cost
Capital Expenditure (CapEx)	Intel CPU	$2,740 (5.7%)	$2,740 (21.0%)	60× SoC	$24,489 (67.5%)
	DRAM	$3,540 (7.3%)	$3,540 (27.1%)	12× PCB	$7,075 (19.5%)
	Disk	$1,220 (2.5%)	$1,220 (9.4%)	Ethernet Switch Board	$689 (1.9%)
	8× NVIDIA A40 GPU	$35,192 (73.0%)	$0 (0%)	BMC	$1,923 (5.3%)
	Others	$5,544 (11.5%)	$5,544 (42.5%)	Others	$2,104 (5.8%)
	Total CapEx	$48,236	$13,044	Total CapEx	$36,280
	Total CapEx/36 months	$1,340	$363	Total CapEx/36 months	$1,008
Operational Expenditure (OpEx)	Avg. peak power consumption	1,231 Watts	633 Watts	Avg. peak power consumption	589 Watts
	Monthly kWh (50% Util.)	443 kWh	228 kWh	Monthly kWh (50% Util.)	212 kWh
	Electricity unit cost [21]	$0.0786/kWh	$0.0786/kWh	Electricity unit cost [21]	$0.0786/kWh
	Server electricity cost	$35	$18	Server electricity cost	$17
	PUE Overhead (PUE=2.0 [52])	$35	$18	PUE Overhead (PUE=2.0 [52])	$17
	Monthly electricity cost	$70	$36	Monthly electricity cost	$34
Total	Monthly TCO	$1,410	$399	Monthly TCO	$1,042

Table 5. Capital expenditure (CapEx), monthly operational expenditure (OpEx), and calculated monthly TCO of each edge server. We additionally estimated the TCO of traditional edge servers without NVIDIA GPUs. As an example, we use the average peak power consumption when performing live streaming transcoding on V5 to get the monthly electricity cost.

NVIDIA A40 GPUs incurs 1,251 Watts of power consumption on average. By assuming servers' average power consumption is 50% of their peak power, monthly power consumption is calculated as $1231W * 50\% * 24h * 30/1000 = 443kWh$. For the electricity unit cost, we referred to the U.S. industrial average electricity price over one year (from August 2021 to July 2022) [21]. Thus, the monthly electricity cost directly related to computation is $\$0.0786/kWh * 443kWh \approx \$35$. The PUE (Power Usage Effectiveness) overhead, which is determined by the PUE value, also contributes to the monthly electricity cost [52]. The PUE value reflects the ratio of total building power consumption to IT infrastructure power consumption. We used a slightly higher PUE value (2.0) at the edge, compared to 1.5 at cloud data centers [52]. The overall monthly electricity cost is $\$35 + \$35 * (2.0 - 1) = \$70$.

In line with prior work [81], we categorized the monthly TCO into the following:

- **Total CapEx amortized to 36 months.** By assuming a 3-year lifetime of the whole server, as in previous work [52, 73, 81], we amortized the CapEx of each server to 36 months, as shown in Table 5.
- **Monthly operational expenditure** mainly refers to the electricity cost. Notably, the monthly OpEx is much less than the amortized CapEx (e.g., $70 vs. $1,340 for the traditional edge server).

We sum the above two expenditure figures to get the monthly TCO. We then normalize the application throughput (measured in previous sections) to the monthly TCO to present the TpC metric in Table 6.

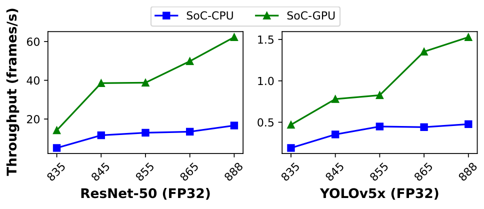
Server	Hardware	Live Streaming Transcoding TpC (streams/$)					
		V1	V2	V3	V4	V5	V6
Traditional Edge Server	Intel CPU	0.180	0.223	0.057	0.101	0.042	0.013
	GPU A40	0.420	0.210	0.102	0.181	0.114	0.034
Traditional Edge Server (W/O GPU)	Intel CPU	0.627	0.777	0.200	0.351	0.146	0.047
SoC-Cluster	SoC-CPU	0.748	0.863	0.230	0.519	0.173	0.058

Server	Hardware	Archive Transcoding TpC (frames/s/$)					
		V1	V2	V3	V4	V5	V6
Traditional Edge Server	Intel CPU	0.027	0.053	0.020	0.024	0.004	0.001
	GPU A40	0.162	0.140	0.203	0.086	0.091	0.035
Traditional Edge Server (W/O GPU)	Intel CPU	0.094	0.189	0.072	0.085	0.013	0.004
SoC-Cluster	SoC-CPU	0.015	0.046	0.010	0.022	0.002	<0.001

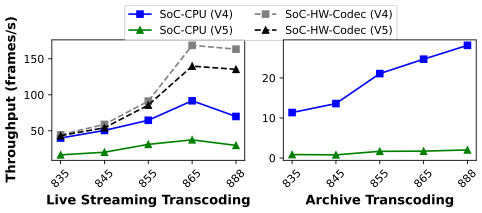
Server	Hardware	DL Serving TpC (frames/s/$)					
		R50 (FP32)	R152 (FP32)	YOLO (FP32)	BERT (FP32)	R50 (INT8)	R152 (INT8)
Traditional Edge Server	Intel CPU	0.579	0.176	0.010	0.044	1.201	0.355
	GPU A40	14.631	4.535	0.571	7.311	45.684	19.840
Traditional Edge Server (W/O GPU)	Intel CPU	2.026	0.617	0.036	0.152	4.199	1.242
SoC-Cluster	SoC-CPU	0.750	0.131	0.026	1.840	-	
	SoC-GPU	3.210	0.628	0.077	-		
	SoC-DSP	-				6.673	2.871

Table 6. Normalized application throughput to monthly TCO. We highlight the highest TpC among used hardware for each video/model.

For live streaming transcoding, SoC-Cluster is much more cost-efficient than the traditional server with GPUs, e.g., 4.19× higher than the Intel CPU and 2.33× higher than the NVIDIA GPU. Moreover, the Intel CPU in the non-GPU server



(a) DL Serving



(b) Video Transcoding

Fig. 21. Performance evolution of five high-end Qualcomm Snapdragon SoCs released between 2017–2021.

shows a higher TpC than NVIDIA GPUs. For archive transcoding, SoC-Cluster is less cost-efficient than the traditional edge server due to its low throughput on a single SoC and relatively high CapEx. In most cases, processing archive transcoding using the NVIDIA GPU provides higher TpC than using other hardware. Finally, DL serving favors data-center GPUs, whose cost efficiency is much higher than SoC-Clusters. This is mainly attributed to the fact that NVIDIA GPUs can be fully loaded with batched DL requests and therefore deliver very high throughput.

Summary. *The capital expenditure of SoC-Cluster is close to an 8-GPU server and much higher than a CPU server. Under the same utilization, SoC-Cluster reduces the operational expenditure by saving electricity, but the total cost is still dominated by the capital expenditure. Regarding concrete workloads, SoC-Cluster delivers much higher cost efficiency than both traditional CPU and GPU servers on live streaming transcoding, but it underperforms on archive video transcoding (huge computing capacity required) and DL serving (highly optimized workloads) compared to the NVIDIA GPU.*

7 SOC LONGITUDINAL STUDY

Previous experiments are performed on the SoC-Cluster consisting of one specific SoC model, i.e., the Qualcomm Snapdragon 865 released in 2020. To expand the observations to more hardware types, and more importantly to understand the SoC performance evolution over time, we performed a longitudinal study on five SoCs released in different years. We selected five smartphones equipped with different high-end SoCs (Qualcomm Snapdragon 8xx series) released from 2016 to 2021 in Table 7. We repeated our throughput experiments for two targeted workloads on those SoCs.

Devices	SoC	CPU	GPU	DSP	Release Yr.
Xiaomi 11 Pro	SD 888	Kryo 680	Adreno 660	Hexagon 780	Jun. 2021
Meizu 17	SD 865	Kryo 585	Adreno 650	Hexagon 698	Mar. 2020
Meizu 16T	SD 855	Kryo 485	Adreno 640	Hexagon 690	Mar. 2019
Xiaomi 8	SD 845	Kryo 385	Adreno 630	Hexagon 685	Feb. 2018
Xiaomi 6	SD 835	Kryo 280	Adreno 540	Hexagon 682	Mar. 2017

Table 7. Device specifications used in this section. “SD” represents “Snapdragon”.

First, we measured the DL serving throughput using ResNet-50 and YOLOv5x. The experimental settings were the same as §5. Based on the results in Figure 21a, embedded GPUs see a significant performance boost from the Snapdragon 835 (2017) to the Snapdragon 888 (2021), e.g., 3.25× higher throughput on YOLOv5x. Embedded CPUs are also gradually improving but not as significantly as GPUs, e.g., 2.52× higher throughput on YOLOv5x. As a consequence, the performance gap between embedded CPUs and GPUs is growing quickly. In the same period (2017–2020), the throughput of the Qualcomm Snapdragon (835 vs. 865) on YOLOv5x improved by 2.88×, while server-end NVIDIA GPUs (V100 vs. A100) only increased by 2.56× according to our extra experiments.

For video transcoding workloads, we represent throughput using the average FPS while transcoding a fixed-duration video for simplicity. Our key observation is that video transcoding tasks also reflect the gradual improvement in DL serving SoC performance, especially for the hardware codec. For live streaming transcoding using embedded CPUs, the throughput for V4 on the Snapdragon 865 was 1.42×, 1.82×, and even 2.3× higher than the 855, 845, and 835, respectively. Such an improvement is much more impressive on SoCs’ hardware codec. The live streaming transcoding throughput of the Snapdragon 865 was 3.8× and 3.24× higher than the 835 for V4 and V5, respectively. The performance improvements of the hardware codec not only come from the evolution of hardware but also the enhancement of software [9, 31].

Summary. *Mobile SoCs have shown tremendous performance improvements in the past five years. With Moore's Law coming to an end, CPU performance evolution is slowing down, but the capacity of co-processors on mobile SoCs (GPU and hardware codec) is still fast growing and their benefits over CPU are increasing. To achieve a sustained performance evolution for SoC-Cluster, it is pivotal to leverage these co-processors.*

8 IMPLICATIONS AND DISCUSSION

Potential killer apps on SoC-Cluster. This work shows the superior performance of SoC-Cluster for two critical workloads: video transcoding and DL serving. Beyond that, we envision that SoC-Cluster can efficiently serve much richer applications than we have explored. For instance, SoC-Cluster is endowed with large storage space and a high I/O speed, as presented in §2.2, making it a suitable platform for database systems with compatible design patterns [130]. In addition, the SoC-level workload scheduling granularity makes a good case for serverless workloads [78], which often execute very briefly during a session [106]. Besides that, DL training [54], which is notoriously power-hungry and can benefit from high hardware-level parallelism, is also suitable for SoC-Cluster with inter-SoC cooperation.

Hardware design. The SoC-Cluster we experimented with is specifically designed for cloud gaming. According to our observations, its hardware must be improved or even re-designed to embrace more general workloads.

• *Network infrastructure and topology.* We show in §4 that an overall 20 Gbps network capacity is mostly enough to support video transcoding on all SoCs. But with the SoC integrated more densely or its computing continuously improving, the network will likely bottleneck and leave the SoC hardware under-utilized. More importantly, workloads such as DL inference/training or HPC demand frequent, high-volume data exchanges across SoCs within a server. The current network infrastructure is not equipped for those workloads due to the high RTT and low bandwidth, especially the per-PCB network capacity (1 Gbps). SoC-Cluster should incorporate recent progress from network research/industry [1, 29, 110].

• *Hardware reliability.* Unlike data-center hardware, mobile SoCs are not designed to operate at full speed and 24/7, but SoC-Cluster requires them to do so. Moreover, the corruption of one SoC module (e.g., flash) makes the whole SoC unusable. As such, fault tolerance is critical. On the one hand, it is good to have a better PCB design to support pluggable SoCs. On the other, a software-level failure handler is required to quickly detect and respond to SoC failure.

Software stacks. While the software is quite ample to support different kinds of workloads on mobile SoCs, we lack middleware to bridge the gap when SoCs are organized as a cluster instead of as a single smartphone chip.

• *Operating system.* Mobile OSs are designed and optimized for interactive scenarios rather than server workloads. For instance, most Android framework components are not needed, such as the telephony manager. Although it is viable to run Linux or even Windows [39] on ARM SoCs, simply replacing Android with other OSs like Ubuntu makes SoC-Cluster no longer compatible with native apps (like cloud gaming) or unable to leverage certain hardware accelerators such as GPU and hardware codec because their drivers are vendor-specific and proprietary [116]. To get the most from both, the right way seems to be to revise the original Android OS to fit edge workloads.

• *Intelligent scheduling.* Workload management and scheduling is a hot topic in data centers to improve resource utilization and load balance [91, 95, 100, 117]. Workload scheduling on SoC-Cluster is also urgently required but could give rise to unique challenges. For example, each

SoC can serve multiple workloads simultaneously with its heterogeneous processors. This adds another dimension of flexibility (and thus complexity) in designing a workload hardware placement algorithm. Furthermore, the asymmetric SoC processor (e.g., ARM big.LITTLE CPU) needs to be considered to achieve load balance.

• *SoC virtualization.* Virtualization is the key to the success of cloud computing by offering a more flexible business model to service providers. In SoC-Cluster, each SoC consists of eight CPU cores and many co-processors and therefore can be virtualized into smaller units. We are aware of a few engineering efforts [14, 57, 111, 121] to build containers on SoC, but their performance and capability are limited. Two unique challenges need to be addressed to virtualize SoCs: first, the virtualization runtime must be lightweight to minimize hardware resource waste; second, since each SoC has very few resources, virtualization might lead to high resource fragmentation with an inefficient bin-packing strategy.

9 RELATED WORK

Grouping mobile SoCs as servers. We are not the first to try to conceptualize a server consisting of tiny SoCs. There have been attempts [101, 102] to investigate whether mobile SoCs can provide sufficient performance and reduce costs for HPC. To reduce e-waste, Shahrad et al. [107] built computation nodes with used smartphones and analyzed server design, but they didn't evaluate real workloads. Switzer et al. used only five smartphones to build a junkyard data center [114] with carbon concerns. Some work uses IoT/mobile SoCs to support specific applications like video transcoding [87], key-value storage [50], web search [73], and parallel computing [55]. However, they mostly focus on specific app types and lack a performance comparison with traditional servers. In contrast to the above work, we used a COTS SoC-Cluster, performed an application-driven measurement on it, and presented detailed performance metrics to show its capabilities for modern, computation-intensive edge workloads.

DL model serving has become a pivotal module in modern intelligent applications and has received extensive attention from both academia and industry [13, 17, 56, 58, 64, 105]. Existing work has focused on optimizing DL performance on traditional data-center-level CPUs and GPUs. Meanwhile, on-device DL is rising towards low inference delay and user privacy preservation [79, 82, 88, 115, 123, 124] powered by the continuously improving SoC hardware. This trend has inspired us to treat DL serving as a killer workload to be supported by SoC-Cluster.

Video transcoding is the fundamental cloud service in our digital world to power cloud video infrastructures [8, 33, 72] or applications like AR/VR [108, 109, 119], campus TV [87], or video analytics [68, 118, 125]. The previous measurement also shows that video-related applications contribute to most of the workloads at the edge [122]. For low-cost video transcoding, Video-CoreCluster [87] used Raspberry Pi to build a video transcoder cluster. Google presented its new hardware-accelerated video transcoding block in data centers [104]. In this work, we show that SoC-Cluster, and especially its hardware codec, can handle video transcoding with high throughput and low energy consumption.

Energy-efficient cloud/edge. Energy efficiency has been recognized as a crucial factor in data centers [62]. Various techniques have been explored in a move towards green data centers, including workload scheduling and management [83, 86, 89, 103, 133], resource under-provisioning [92, 128], greening the data-center network [63, 65, 70], etc. In contrast to these software-level approaches, we propose a re-design of servers to fundamentally boost energy efficiency. As the edge infrastructure is still at a preliminary stage, we deem such a radical measure possible.

Measurements at the edge. Measurement efforts have been made from centralized clouds [61, 67, 84, 112] and geo-distributed clouds [60, 76, 93], and towards decentralized edge clouds [122]. They mostly focus on workload scheduling, network routing/topology, or billing models. Some prior work built benchmarks for specific applications such as video transcoding [90]. We inherited some of the spirit of previous work (e.g., the metrics and settings) to build our first-of-its-kind benchmark suite for SoC-Cluster.

10 CONCLUSIONS

In this work, we investigated the possibility and implications of a new form of edge server, i.e., an SoC-Cluster consisting of many mobile SoCs. Through extensive benchmarks on two killer edge workloads (i.e., video transcoding and DL serving), we show that SoC-Cluster achieves significant energy savings as compared to conventional edge servers, as well as demonstrating its limitations. The observations paint a promising future for SoC-Cluster in edge scenarios and also guide us to further improve it, in terms of both hardware and software.

REFERENCES

[1] Infiniband - Wikipedia. https://en.wikipedia.org/wiki/InfiniBand.
[2] Energy-efficient cloud computing technologies and policies for an eco-friendly cloud market. https://digital-strategy.ec.europa.eu/en/library/energy-efficient-cloud-computing-technologies-and-policies-eco-friendly-cloud-market, 2020.
[3] Hardware | Qualcomm Snapdragon 865 5G Mobile Platform | 5G Mobile Processor | Qualcomm. https://www.qualcomm.com/products/application/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-865-5g-mobile-platform, 2020.
[4] Running average power limit energy reporting / cve-2020-8694 , cve-2020-8695 / intel-sa-00389. https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html, 2020.
[5] Mz7lh960hajr(960gb) | SSD | Samsung Semiconductor Global. https://semiconductor.samsung.com/ssd/datacenter-ssd/pm883/mz7lh960hajr/, 2021.
[6] Qualcomm Hexagon DSP. https://developer.qualcomm.com/sites/default/files/docs/adreno-gpu/developer-guide/dsp/dsp.html, 2021.
[7] Qualcomm Snapdragon 888 5G Mobile Platform | Qualcomm. https://www.qualcomm.com/products/application/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-888-5g-mobile-platform, 2021.
[8] Reimagining video infrastructure to empower youtube. https://blog.youtube/inside-youtube/new-era-video-infrastructure/, 2021.
[9] Video encoding improvemens - Android 12. https://developer.android.com/about/versions/12/features#video-encoding, 2021.
[10] What edge computing means for infrastructure and operations leaders. https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders, 2021.
[11] Amazon EC2 Instance Types. https://aws.amazon.com/ec2/instance-types/, 2022.
[12] Amazon Luna. https://www.amazon.com/luna/landing-page, 2022.
[13] Amazon SageMaker. https://aws.amazon.com/sagemaker/, 2022.
[14] anbox - android in a box. https://anbox.io/, 2022.
[15] Aws Wavelength. https://aws.amazon.com/wavelength/, 2022.
[16] axboe/fio: Flexible i/o tester. https://github.com/axboe/fio, 2022.
[17] Azure Machine Learning. https://docs.microsoft.com/en-us/azure/machine-learning/, 2022.
[18] BERT en uncased. https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1, 2022.
[19] Cloud Gaming, Meet Facebook Gaming. https://www.facebook.com/fbgaminghome/blog/cloud-gaming-meetfacebook-gaming, 2022.
[20] Data Sheet - Seagate. https://www.seagate.com/www-content/datasheets/pdfs/ent-cap-3-5-hdd-data-sheetDS1882-3-1610US-en_US.pdf, 2022.
[21] Electric power monthly - u.s. energy information administration (eia). https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=table_5_03, 2022.
[22] Ffmpeg. https://ffmpeg.org/, 2022.
[23] Geekbench 5 - cross-platform benchmark. https://www.geekbench.com/, 2022.
[24] Geforce Now. https://www.nvidia.com/en-us/geforce-now/, 2022.
[25] Genshin Impact. https://genshin.hoyoverse.com/en/, 2022.
[26] Google Stadia. https://stadia.google.com/, 2022.
[27] Hwaccelintro - FFmpeg. https://trac.ffmpeg.org/wiki/HWAccelIntro, 2022.
[28] Instance family. https://www.alibabacloud.com/help/en/elastic-compute-service/latest/instance-family, 2022.

[29] Intel Tofino. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html, 2022.

[30] linkedin/litr: Lightweight hardware accelerated video/audio transcoder for Android. https://github.com/linkedin/LiTr, 2022.

[31] Low Latency Decoding in MediaCodec - Android Open Source Project. https://source.android.com/devices/media/low-latency-media, 2022.

[32] Mediacodec. https://developer.android.com/reference/android/media/MediaCodec, 2022.

[33] Netflix. https://www.netflix.com/, 2022.

[34] Nvenc Application Note. https://docs.nvidia.com/video-technologies/video-codec-sdk/nvenc-application-note/#nvenc-performance, 2022.

[35] Nvidia TensorRT. https://developer.nvidia.com/tensorrt, 2022.

[36] Nvidia triton inference server. https://developer.nvidia.com/nvidia-triton-inference-server, 2022.

[37] Nvidia Video Codec SDK. https://developer.nvidia.com/nvidia-video-codec-sdk, 2022.

[38] Optimize for Doze and App Standby | Android Developers. https://developer.android.com/training/monitoring-device-state/doze-standby, 2022.

[39] Project volterra: Everything you need to know about microsoft's arm developer kit. https://www.windowscentral.com/hardware/laptops/surface/project-volterra-everything-you-need-to-know, 2022.

[40] Qualcomm showcases future technology roadmap to drive the connected intelligent edge and lead the world to 5g advanced and beyond. https://www.qualcomm.com/news/releases/2022/02/28/qualcomm-showcases-future-technology-roadmap-drive-connected-intelligent, 2022.

[41] Recommended upload encoding settings - YouTube Help. https://support.google.com/youtube/answer/1722171?hl=en, 2022.

[42] remote-android/redroid-doc. https://github.com/remote-android/redroid-doc, 2022.

[43] Serving Models | TFX | TensorFlow. https://www.tensorflow.org/tfx/guide/serving, 2022.

[44] Tensorflow Lite. https://www.tensorflow.org/lite, 2022.

[45] Torchserve. https://pytorch.org/serve/, 2022.

[46] Video Encode and Decode GPU Support Matrix | NVIDIA Developer. https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new, 2022.

[47] Virtual machine series | Microsoft Azure. https://azure.microsoft.com/en-us/pricing/details/virtual-machines/series/, 2022.

[48] X-cloud Game Pass. https://www.xbox.com/en-US/xbox-game-pass/cloud-gaming?xr=shellnav, 2022.

[49] x264, the best H.264/AVC encoder - VideoLAN. https://www.videolan.org/developers/x264.html, 2022.

[50] David G Andersen, Jason Franklin, Michael Kaminsky, Amar Phanishayee, Lawrence Tan, and Vijay Vasudevan. Fawn: A fast array of wimpy nodes. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 1–14, 2009.

[51] Jeremy Andrus, Christoffer Dall, Alexander Van't Hof, Oren Laadan, and Jason Nieh. Cells: a virtual mobile smartphone architecture. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 173–187, 2011.

[52] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. The datacenter as a computer: Designing warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 13(3):i–189, 2018.

[53] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *IEEE Computer*, 40, 2007.

[54] K. A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé M Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019. To appear.

[55] Felix Büsching, Sebastian Schildt, and Lars Wolf. Droidcluster: Towards smartphone cluster computing–the streets are paved with potential computer clusters. In *2012 32nd International Conference on Distributed Computing Systems Workshops*, pages 114–117. IEEE, 2012.

[56] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. TVM: An automated End-to-End optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594, Carlsbad, CA, October 2018. USENIX Association.

[57] Wenzhi Chen, Lei Xu, Guoxi Li, and Yang Xiang. A lightweight virtualization solution for android devices. *IEEE Transactions on Computers*, 64(10):2741–2751, 2015.

[58] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In Aditya Akella and Jon Howell, editors, *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, pages 613–627. USENIX Association, 2017.

[59] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A Low-Latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 613–627, Boston, MA, March 2017. USENIX Association.

[60] The Khang Dang, Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Jörg Ott, and Jussi Kangasharju. Cloudy with a chance of short rtts: Analyzing cloud connectivity in the internet. In *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, page 62–79, New York, NY, USA, 2021. Association for Computing Machinery.

[61] Haotian Deng, Chunyi Peng, Ans Fida, Jiayi Meng, and Y. Charlie Hu. Mobility support in cellular networks: A measurement study on its configurations and implications. In *Proceedings of the Internet Measurement Conference 2018, IMC 2018, Boston, MA, USA, October 31 - November 02, 2018*, pages 147–160. ACM, 2018.

[62] Wei Deng, Fangming Liu, Hai Jin, Bo Li, and Dan Li. Harnessing renewable energy in cloud datacenters: opportunities and challenges. *iEEE Network*, 28(1):48–55, 2014.

[63] Peter Xiang Gao, Andrew R. Curtis, Bernard Wong, and Srinivasan Keshav. It's not easy being green. *SIGCOMM Comput. Commun. Rev.*, 42(4):211–222, aug 2012.

[64] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving DNNs like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 443–462. USENIX Association, November 2020.

[65] Zehua Guo, Yang Xu, Ya-Feng Liu, Sen Liu, H Jonathan Chao, Zhi-Li Zhang, and Yuanqing Xia. Aggreflow: Achieving power efficiency, load balancing, and quality of service in data center networks. *IEEE/ACM Transactions on Networking*, 29(1):17–33, 2020.

[66] Sangwook Shane Hahn, Sungjin Lee, Inhyuk Yee, Donguk Ryu, and Jihong Kim. FastTrack: Foreground App-Aware I/O management for improving user experience of android smartphones. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 15–28, Boston, MA, July 2018. USENIX Association.

[67] Osama Haq, Mamoon Raja, and Fahad R. Dogar. Measuring and improving the reliability of wide-area cloud paths. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 253–262. ACM, 2017.

[68] Brandon Haynes, Maureen Daum, Dong He, Amrita Mazumdar, Magdalena Balazinska, Alvin Cheung, and Luis Ceze. Vss: A storage system for video analytics. In *Proceedings of the 2021 International Conference on Management of Data*, pages 685–696, 2021.

[69] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[70] Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, page 17, USA, 2010. USENIX Association.

[71] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[72] Qi Huang, Petchean Ang, Peter Knowles, Tomasz Nykiel, Iaroslav Tverdokhlib, Amit Yajurvedi, Paul Dapolito IV, Xifan Yan, Maxim Bykov, Chuen Liang, Mohit Talwar, Abhishek Mathur, Sachin Kulkarni, Matthew Burke, and Wyatt Lloyd. SVE: distributed video processing at facebook scale. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 87–103. ACM, 2017.

[73] Vijay Janapa Reddi, Benjamin C Lee, Trishul Chilimbi, and Kushagra Vaid. Web search using mobile cores: quantifying and mitigating the price of efficiency. *ACM SIGARCH Computer Architecture News*, 38(3):314–325, 2010.

[74] Congfeng Jiang, Yumei Wang, Dongyang Ou, Bing Luo, and Weisong Shi. Energy proportional servers: Where are we in 2016? In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1649–1660. IEEE, 2017.

[75] Xiaotang Jiang, Huan Wang, Yiliu Chen, Ziqi Wu, Lichuan Wang, Bin Zou, Yafeng Yang, Zongyang Cui, Yu Cai, Tianhang Yu, et al. Mnn: A universal and efficient inference engine. *arXiv preprint arXiv:2002.12418*, 2020.

[76] Yuchen Jin, Sundararajan Renganathan, Ganesh Ananthanarayanan, Junchen Jiang, Venkata N. Padmanabhan, Manuel Schröder, Matt Calder, and Arvind Krishnamurthy. Zooming in on wide-area latencies to a global cloud provider. In Jianping Wu and Wendy Hall, editors, *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, pages 104–116. ACM, 2019.

[77] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, February 2022.

[78] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, João Carreira, Karl Krauth, Neeraja Jayant Yadwadkar, Joseph E. Gonzalez, Raluca Ada Popa, Ion Stoica, and David A.

Patterson. Cloud programming simplified: A berkeley view on serverless computing. *CoRR*, abs/1902.03383, 2019.

[79] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor N. Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In Yunji Chen, Olivier Temam, and John Carter, editors, *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2017, Xi'an, China, April 8-12, 2017*, pages 615–629. ACM, 2017.

[80] David Koufaty and Deborah T. Marr. Hyperthreading technology in the netburst microarchitecture. *IEEE Micro*, 23(2):56–65, mar 2003.

[81] Willis Lang, Jignesh M Patel, and Srinath Shankar. Wimpy node clusters: What about non-wimpy workloads? In *Proceedings of the Sixth International Workshop on Data Management on New Hardware*, pages 47–55, 2010.

[82] Stefanos Laskaridis, Stylianos I. Venieris, Mário Almeida, Ilias Leontiadis, and Nicholas D. Lane. SPINN: synergistic progressive inference of neural networks over device and cloud. In *MobiCom '20: The 26th Annual International Conference on Mobile Computing and Networking, London, United Kingdom, September 21-25, 2020*, pages 37:1–37:15. ACM, 2020.

[83] Seunghak Lee, Ki-Dong Kang, Hwanjun Lee, Hyungwon Park, Younghoon Son, Nam Sung Kim, and Daehoon Kim. Greendimm: Os-assisted dram power management for dram with a sub-array granularity power-down state. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 131–142, 2021.

[84] Fangfan Li, Arian Akhavan Niaki, David R. Choffnes, Phillipa Gill, and Alan Mislove. A large-scale analysis of deployed traffic differentiation practices. In Jianping Wu and Wendy Hall, editors, *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, pages 130–144. ACM, 2019.

[85] Shaohong Li, Xi Wang, Xiao Zhang, Vasileios Kontorinis, Sreekumar Kodakara, David Lo, and Parthasarathy Ranganathan. Thunderbolt: Throughput-Optimized, Quality-of-Service-Aware power capping at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 1241–1255. USENIX Association, November 2020.

[86] Fangming Liu, Zhi Zhou, Hai Jin, Bo Li, Baochun Li, and Hongbo Jiang. On arbitrating the power-performance tradeoff in saas clouds. *IEEE Transactions on Parallel and Distributed Systems*, 25(10):2648–2658, 2013.

[87] Peng Liu, Jongwon Yoon, Lance Johnson, and Suman Banerjee. Greening the video transcoding service with {Low-Cost} hardware transcoders. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 407–419, 2016.

[88] Sicong Liu, Yingyan Lin, Zimu Zhou, Kaiming Nan, Hui Liu, and Junzhao Du. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In Jörg Ott, Falko Dressler, Stefan Saroiu, and Prabal Dutta, editors, *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2018, Munich, Germany, June 10-15, 2018*, pages 389–400. ACM, 2018.

[89] Zhenhua Liu, Yuan Chen, Cullen Bash, Adam Wierman, Daniel Gmach, Zhikui Wang, Manish Marwah, and Chris Hyser. Renewable and cooling aware workload management for sustainable data centers. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, pages 175–186, 2012.

[90] Andrea Lottarini, Alex Ramírez, Joel Coburn, Martha A. Kim, Parthasarathy Ranganathan, Daniel Stodolsky, and Mark Wachsler. vbench: Benchmarking video transcoding in the cloud. In Xipeng Shen, James Tuck, Ricardo Bianchini, and Vivek Sarkar, editors, *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2018, Williamsburg, VA, USA, March 24-28, 2018*, pages 797–809. ACM, 2018.

[91] Kshiteej Mahajan, Arjun Balasubramanian, Arjun Singhvi, Shivaram Venkataraman, Aditya Akella, Amar Phanishayee, and Shuchi Chawla. Themis: Fair and efficient GPU cluster scheduling. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 289–304, Santa Clara, CA, February 2020. USENIX Association.

[92] Ioannis Manousakis, Íñigo Goiri, Sriram Sankar, Thu D Nguyen, and Ricardo Bianchini. Coolprovision: Underprovisioning datacenter cooling. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 356–367, 2015.

[93] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. Pruning edge research with latency shears. In Ben Zhao, Heather Zheng, Harsha V. Madhyastha, and Venkat N. Padmanabhan, editors, *HotNets '20: The 19th ACM Workshop on Hot Topics in Networks, Virtual Event, USA, November 4-6, 2020*, pages 182–189. ACM, 2020.

[94] David Mytton. Data centre water consumption. *npj Clean Water*, 4, 12 2021.

[95] Deepak Narayanan, Keshav Santhanam, Fiodar Kazhamiaka, Amar Phanishayee, and Matei Zaharia. Heterogeneity-Aware cluster scheduling policies for deep learning workloads. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 481–498. USENIX Association, November 2020.

[96] David T. Nguyen, Gang Zhou, Guoliang Xing, Xin Qi, Zijiang Hao, Ge Peng, and Qing Yang. Reducing smartphone application delay through read/write isolation. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, page 287–300, New York, NY, USA, 2015. Association for Computing

Machinery.

[97] Abhinav Parate, Matthias Böhmer, David Chu, Deepak Ganesan, and Benjamin M. Marlin. Practical prediction and prefetch for faster access to applications on mobile phones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, page 275–284, New York, NY, USA, 2013. Association for Computing Machinery.

[98] Jongsoo Park, Maxim Naumov, Protonu Basu, Summer Deng, Aravind Kalaiah, Daya Shanker Khudia, James Law, Parth Malani, Andrey Malevich, Nadathur Satish, Juan Miguel Pino, Martin Schatz, Alexander Sidorov, Viswanath Sivakumar, Andrew Tulloch, Xiaodong Wang, Yiming Wu, Hector Yuen, Utku Diril, Dmytro Dzhulgakov, Kim M. Hazelwood, Bill Jia, Yangqing Jia, Lin Qiao, Vijay Rao, Nadav Rotem, Sungjoo Yoo, and Mikhail Smelyanskiy. Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications. *CoRR*, abs/1811.09886, 2018.

[99] Qiangyu Pei, Shutong Chen, Qixia Zhang, Xinhui Zhu, Fangming Liu, Ziyang Jia, Yishuo Wang, and Yongjie Yuan. Cooledge: Hotspot-relievable warm water cooling for energy-efficient edge datacenters. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, page 814–829, New York, NY, USA, 2022. Association for Computing Machinery.

[100] Aurick Qiao, Sang Keun Choe, Suhas Jayaram Subramanya, Willie Neiswanger, Qirong Ho, Hao Zhang, Gregory R. Ganger, and Eric P. Xing. Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 1–18. USENIX Association, July 2021.

[101] Nikola Rajovic, Paul M Carpenter, Isaac Gelado, Nikola Puzovic, Alex Ramirez, and Mateo Valero. Supercomputing with commodity cpus: Are mobile socs ready for hpc? In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2013.

[102] Nikola Rajovic, Alejandro Rico, Nikola Puzovic, Chris Adeniyi-Jones, and Alex Ramirez. Tibidabo: Making the case for an arm-based hpc system. *Future Generation Computer Systems*, 36:322–334, 2014.

[103] Yongyi Ran, Han Hu, Xin Zhou, and Yonggang Wen. Deepee: Joint optimization of job scheduling and cooling control for data center energy efficiency using deep reinforcement learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 645–655. IEEE, 2019.

[104] 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, J. P. Maaninen, Ramon Macias, Maire Mahony, David Alexander Munday, Srikanth Muroor, Narayana Penukonda, Eric Perkins-Argueta, Devin Persaud, Alex Ramírez, 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. Warehouse-scale video acceleration: co-design and deployment in the wild. In Tim Sherwood, Emery D. Berger, and Christos Kozyrakis, editors, *ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021*, pages 600–615. ACM, 2021.

[105] Francisco Romero, Qian Li, Neeraja J. Yadwadkar, and Christos Kozyrakis. Infaas: Automated model-less inference serving. In Irina Calciu and Geoff Kuenning, editors, *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*, pages 397–411. USENIX Association, 2021.

[106] Mohammad Shahrad, Rodrigo Fonseca, Iñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In Ada Gavrilovska and Erez Zadok, editors, *2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020*, pages 205–218. USENIX Association, 2020.

[107] Mohammad Shahrad and David Wentzlaff. Towards deploying decommissioned mobile devices as cheap energy-efficient compute nodes. In *9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17)*, 2017.

[108] Shu Shi, Varun Gupta, Michael Hwang, and Rittwik Jana. Mobile VR on edge cloud: a latency-driven design. In Michael Zink, Laura Toni, and Ali C. Begen, editors, *Proceedings of the 10th ACM Multimedia Systems Conference, MMSys 2019, Amherst, MA, USA, June 18-21, 2019*, pages 222–231. ACM, 2019.

[109] Shu Shi, Varun Gupta, and Rittwik Jana. Freedom: Fast recovery enhanced VR delivery over mobile networks. In Junehwa Song, Minkyong Kim, Nicholas D. Lane, Rajesh Krishna Balan, Fahim Kawsar, and Yunxin Liu, editors, *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2019, Seoul, Republic of Korea, June 17-21, 2019*, pages 130–141. ACM, 2019.

[110] Anirudh Sivaraman, Thomas Mason, Aurojit Panda, Ravi Netravali, and Sai Anirudh Kondaveeti. Network architecture in the age of programmability. *ACM SIGCOMM Computer Communication Review*, 50(1):38–44, 2020.

[111] Wenna Song, Jiang Ming, Lin Jiang, Yi Xiang, Xuanchen Pan, Jianming Fu, and Guojun Peng. Towards transparent and stealthy android os sandboxing via customizable container-based virtualization. In *Proceedings of the 2021 ACM*

SIGSAC Conference on Computer and Communications Security, CCS '21, page 2858–2874, New York, NY, USA, 2021. Association for Computing Machinery.

[112] Neil T. Spring, Ratul Mahajan, and Thomas E. Anderson. The causes of path inflation. In Anja Feldmann, Martina Zitterbart, Jon Crowcroft, and David Wetherall, editors, *Proceedings of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 25-29, 2003, Karlsruhe, Germany*, pages 113–124. ACM, 2003.

[113] Gary J. Sullivan and Thomas Wiegand. Video compression - from concepts to the H.264/AVC standard. *Proc. IEEE*, 93(1):18–31, 2005.

[114] Jennifer Switzer, Ryan Kastner, and Pat Pannuto. Architecture of a junkyard datacenter. *arXiv preprint arXiv:2110.06870*, 2021.

[115] Hao Wu, Jinghao Feng, Xuejin Tian, Edward Sun, Yunxin Liu, Bo Dong, Fengyuan Xu, and Sheng Zhong. EMO: real-time emotion recognition from single-eye images for resource-constrained eyewear devices. In Eyal de Lara, Iqbal Mohomed, Jason Nieh, and Elizabeth M. Belding, editors, *MobiSys '20: The 18th Annual International Conference on Mobile Systems, Applications, and Services, Toronto, Ontario, Canada, June 15-19, 2020*, pages 448–461. ACM, 2020.

[116] Lei Wu, Michael Grace, Yajin Zhou, Chiachih Wu, and Xuxian Jiang. The impact of vendor customizations on android security. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 623–634, 2013.

[117] Wencong Xiao, Shiru Ren, Yong Li, Yang Zhang, Pengyang Hou, Zhi Li, Yihui Feng, Wei Lin, and Yangqing Jia. AntMan: Dynamic scaling on GPU clusters for deep learning. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 533–548. USENIX Association, November 2020.

[118] Zhujun Xiao, Zhengxu Xia, Haitao Zheng, Ben Y. Zhao, and Junchen Jiang. Towards performance clarity of edge video analytics. In *6th IEEE/ACM Symposium on Edge Computing, SEC 2021, San Jose, CA, USA, December 14-17, 2021*, pages 148–164. IEEE, 2021.

[119] Chenhao Xie, Xie Li, Yang Hu, Huwan Peng, Michael B. Taylor, and Shuaiwen Leon Song. Q-VR: system-level design for future mobile collaborative virtual reality. In Tim Sherwood, Emery D. Berger, and Christos Kozyrakis, editors, *ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021*, pages 587–599. ACM, 2021.

[120] Jun Xin, Chia-Wen Lin, and Ming-Ting Sun. Digital video transcoding. *Proceedings of the IEEE*, 93(1):84–97, 2005.

[121] Lei Xu, Guoxi Li, Chuan Li, Weijie Sun, Wenzhi Chen, and Zonghui Wang. Condroid: a container-based virtualization solution adapted for android devices. In *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pages 81–88. IEEE, 2015.

[122] Mengwei Xu, Zhe Fu, Xiao Ma, Li Zhang, Yanan Li, Feng Qian, Shangguang Wang, Ke Li, Jingyu Yang, and Xuanzhe Liu. From cloud to edge: a first look at public edge platforms. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 37–53, 2021.

[123] Mengwei Xu, Feng Qian, Qiaozhu Mei, Kang Huang, and Xuanzhe Liu. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(4):197:1–197:26, 2018.

[124] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. Deepcache: Principled cache for mobile deep vision. In Rajeev Shorey, Rohan Murty, Yingying (Jennifer) Chen, and Kyle Jamieson, editors, *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, MobiCom 2018, New Delhi, India, October 29 - November 02, 2018*, pages 129–144. ACM, 2018.

[125] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. Vstore: A data store for analytics on large videos. In George Candea, Robbert van Renesse, and Christof Fetzer, editors, *Proceedings of the Fourteenth EuroSys Conference 2019, Dresden, Germany, March 25-28, 2019*, pages 16:1–16:17. ACM, 2019.

[126] Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal, and Jie Liu. Fast app launching for mobile devices using predictive user context. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, page 113–126, New York, NY, USA, 2012. Association for Computing Machinery.

[127] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Steven D. Gribble and Dina Katabi, editors, *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012*, pages 15–28. USENIX Association, 2012.

[128] Chaojie Zhang, Alok Gautam Kumbhare, Ioannis Manousakis, Deli Zhang, Pulkit A Misra, Rod Assis, Kyle Woolcock, Nithish Mahalingam, Brijesh Warrier, David Gauthier, et al. Flex: High-availability datacenters with zero reserved power. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 319–332. IEEE, 2021.

[129] Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. MArk: Exploiting cloud services for Cost-Effective, SLO-Aware machine learning inference serving. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 1049–1062, Renton, WA, July 2019. USENIX Association.

[130] Tao Zhang, Aviad Zuck, Donald E. Porter, and Dan Tsafrir. Apps can quickly destroy your mobile's flash: Why they don't, and how to keep it that way. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '19, page 207–221, New York, NY, USA, 2019. Association for Computing Machinery.

[131] Xumiao Zhang, Anlan Zhang, Jiachen Sun, Xiao Zhu, Y Ethan Guo, Feng Qian, and Z Morley Mao. Emp: edge-assisted multi-vehicle perception. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 545–558, 2021.

[132] Jingyu Zhou, Meng Xu, Alexander Shraer, Bala Namasivayam, Alex Miller, Evan Tschannen, Steve Atherton, Andrew J. Beamon, Rusty Sears, John Leach, Dave Rosenthal, Xin Dong, Will Wilson, Ben Collins, David Scherer, Alec Grieser, Young Liu, Alvin Moore, Bhaskar Muppana, Xiaoge Su, and Vishesh Yadav. Foundationdb: A distributed unbundled transactional key value store. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 2653–2666. ACM, 2021.

[133] Zhi Zhou, Fangming Liu, Ruolan Zou, Jiangchuan Liu, Hong Xu, and Hai Jin. Carbon-aware online control of geo-distributed cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 27(9):2506–2519, 2015.

A ETHICAL CONSIDERATIONS

This work does not raise any ethical issues.