Portable Energy-Aware Cluster-Based Edge Computers

Thomas Rausch, Cosmin Avasalcai, Schahram Dustdar
Distributed Systems Group
TU Wien, Vienna, Austria
{t.rausch, c.avasalcai, dustdar}@dsg.tuwien.ac.at

***Abstract*—Computational resources distributed at the edge of the network are the fundamental infrastructural component of edge computing. The operational scale of edge computing introduces new challenges for building and operating suitable computation platforms. Many application scenarios require edge computing resources to provide reliable response times while operating in dynamic and resource-constrained environments. In this paper, we present a novel architecture for energy-aware, cluster-based edge computers that are designed to be portable and usable in fieldwork scenarios. We use compact general-purpose commodity hardware to build a high-density cluster prototype, and implement a power-management runtime to enable real-time energy-awareness. Furthermore, we present an experimental analysis of the energy and resource-consumption characteristics of our prototype in the context of a data analytics application. The results show the feasibility of our prototype for the presented scenarios, but also reveal the intricacies of power-management approaches already built into modern CPUs. We show that different load balancing policies and cluster configurations have a significant impact on energy consumption and system responsiveness. Our insights lay the groundwork for future research on energy-consumption optimization approaches for cluster-based edge computers.**

I. Introduction

The underlying premise of the edge computing vision is a distribution of heterogeneous computational resources deployed at the edge of the network [1]. We consider edge computers as a specialized type of server computer, expressly designed for the use in edge environments. These edge computers, supported by novel middleware and deployment platforms, promise to enable highly-responsive software services that tackle the challenges of today's mobile application scenarios, ranging from data analytics in Internet of Things (IoT) [2] to the seamless augmentation of human cognition [3].

A significant number of these application scenarios can be characterized by the dynamic and resource-constrained environment in which supporting edge computers have to operate. For example, mobile applications are used by emergency response teams for on-premises decision making [4], or by military field personnel for image or speech recognition [5]. Handheld devices stream data and offload compute-intensive tasks to nearby edge computers. Designing and operating edge computing infrastructure for and in these environments is challenging, as edge computers have to be portable to fit on, e.g., an emergency vehicle or a drone; deal with unpredictable client load; provide sufficient performance to host, e.g., data analytics or machine learning applications; and, at the same time, be energy-efficient to be powered by secondary power supplies such as batteries.

In this paper, we present a design, prototype, and evaluation of a portable energy-aware, cluster-based edge computer that aims to address these challenges. It is portable because it is compact in size, and consumes energy at a scale that could be served by medium sized batteries. It is energy-aware because it provides a power-management runtime to access energy consumption data in real-time, and control the power state of its nodes. Finally, it is cluster-based because it comprises multiple physical nodes to provide reliable and scalable computing.

We recognize advanced power-management techniques, such as energy-aware load balancing [6] and dynamic cluster adjustment [7] to be of primary concern when serving applications on portable cluster-based edge computers. Most related work on energy-aware approaches for compute clusters focus either on reducing the operational costs or carbon footprint in Cloud data centers [8], [6], [9], conserving energy of mobile devices by offloading tasks to nearby mobile edge computing (MEC) resources [10], or consider a power supply of infinite capacity [7], [11]. The difference in scale between the Cloud and portable edge computers require new considerations when doing optimizations and load balancing partly because powerful dedicated hardware with unlimited energy supplies may not be available for such computations. Furthermore, most existing research on edge computers, e.g., cloudlets, evaluate only a single-node system, typically a commodity desktop PC [3], [5]. However, understanding the energy consumption and performance capabilities of portable cluster-based edge computers is important for determining the efficacy of such edge computers for real-world use cases, as well as providing groundwork for future research on energy-aware load balancing, scheduling, and provisioning techniques. A major goal of this paper is therefore to provide empirical evidence on the energy consumption characteristics of cluster-based edge computers with modern hardware. To that end, we present the results of several experiments running on top of our energy-aware, cluster-based edge computer, and report on the following research questions: *RQ1.* What are the energy consumption characteristics of cluster-based edge computers? *RQ2.* How do different cluster configurations and load balancing policies affect the responsiveness and energy consumption of cluster-based edge computers?

978-1-5386-9445-9/18/$31.00 ©2018 IEEE
DOI 10.1109/SEC.2018.00026

In summary, this paper presents a prototype for a portable cluster-based edge computer, and lays the groundwork for developing energy-efficiency control mechanisms for operating applications on such infrastructure. The contributions can be summarized as follows:

1) We propose a novel infrastructure architecture for portable energy-aware, cluster-based edge computers. The core of the architecture are multiple physical nodes and a power-management framework that monitors energy and resource consumption of the nodes, controls the power state of nodes, and does energy-aware load balancing of application requests.
2) We implement a prototype of an energy-aware, cluster-based edge computer based on our architecture. We use compact, general-purpose commodity hardware to build a high-density compute cluster, and deploy a power-management framework with power senors and micro-controllers.
3) We present an experimental analysis of the resource and energy consumption characteristics of our prototype in the context of a modern data analytics application, namely image recognition using deep learning. The experimental results shed light on how different cluster configurations affect application responsiveness and overall energy consumption.

The remainder of the paper is structured as follows: Section II motivates our work by discussing the current landscape of cloudlet infrastructure. In Section III we describe our reference architecture for energy-aware, cluster-based edge computers. In Section IV we present the developed prototype. Section V and Section VI present the methodology and results of our in-depth analysis of energy and resource-consumption characteristics of our prototype. In Section VII we discuss the implications of the results on future power-management mechanisms. Section VIII summarizes related work on energy-awareness for edge computing. Finally, Section IX concludes the paper and provides an outlook on future work.

II. Motivation: A Perspective on Edge Computing Resources

Computational resources placed in close proximity to data producers and consumers at the edge of the network are the fundamental infrastructural component to enable edge computing [1]. Many different types of computational infrastructures, such as cloudlets [3], micro-datacenters [12], or even clusters of mobile devices [13] have been proposed. We motivate our work by discussing the characteristics of these infrastructures in more detail.

A. Cloudlets: Providing Cloud Services at the Edge

The idea of cloudlets has been discussed for several years. In an early definition by Satyanarayanan et al. [3], a cloudlet is "a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and is available for use by nearby mobile devices." Since then, many other definitions have been put forth [5], [14], [15], but the general consensus is that cloudlets aim to bring services otherwise hosted in the Cloud closer to the edge of the network to improve Quality of Service for end users. Cloudlets have had a large impact on the idea of mobile edge computing (MEC), where mobile operators place servers at base stations of cellular networks [16]. This allows operators to provide highly-responsive Cloud services for mobile users based on their proximity to base stations, or optimize the battery lifetime of mobile devices by offloading compute tasks to these cloudlets.

B. Cluster-Based Edge Resources

Clustered edge resources of all shapes and sizes have been discussed and evaluated. These proposals range from Sun's Modular Datacenters that come in shipping containers[1]; over Canonical's (discontinued) Orange Box[2], a cluster of Intel NUCs; to a cluster of single-board computers (SBC), such as Raspberry Pis, federated to form a micro datacenter [12]. However, none of these come with an integrated power-management runtime, or other mechanisms for native energy-awareness. Although cloudlets are seen in general as either single-node or cluster-based systems, most existing evaluations of cloudlets in research consider only a single node, typically a commodity desktop PC [3], [5]. We argue that this size of hardware (i.e., commodity computers) is useful for many use cases, as it is a compromise between the two previously presented extremes. However, we also argue that single-node systems are challenged to provide reliable services in the face of varying and unpredictable request arrival rates. A cluster of high-density compute nodes, for example Mini-ITX motherboards with server CPUs, can provide a great deal of computational power, while still being compact and portable.

C. General-Purpose Edge Computers

Most definitions consider a cloudlet to be static infrastructure, well-connected to the Internet, deployed statically at a specific location. Recently, researchers have shown that such cloudlets may be impractical for use cases that require on-premises decision making for military field personnel in resource-constrained tactical environments [5]. To bridge this gap, they propose *tactical cloudlets*, i.e., portable cloudlets deployed on, e.g., vehicles that support computation offload in such resource-constrained environments. We argue that there are many more use cases to be considered that require portable cloudlets, or, more generally, general-purpose edge computers. We consider edge computers a specialized type of server computer, that is designed for the use in edge environments. For example, portable compute clusters could be deployed on emergency vehicles to allow complex decision-making and task-planning analytics applications in emergency response scenarios [4]. Other field-based work, such as archaeological dig sites, field experiments in geology, oceanography, etc., could all benefit from edge computing, but require portable

[1]https://docs.oracle.com/cd/E19115-01/mod.dc.s20/

[2]http://blog.dustinkirkland.com/2014/05/the-orange-box-cloud-for-free-man.html (Accessed 2018-09-03)

and reliable edge computers. Companies are even exploring space-to-cloud analytics for remote regions using satellites[3].

D. Energy-Aware Portable Cluster-Based Edge Computers

Infrastructure required for such scenarios faces numerous challenges. In particular, these edge computers work in energy-constrained environments, and may therefore have to be partially powered by secondary energy sources such as batteries. As we have discussed, single-node systems are challenged to enable scalability and reliability required for dealing with unpredictable workloads at the Edge. While statically scaling out the infrastructure would provide better service reliability, it also significantly increases energy consumption, thereby diminishing portability. However, by integrating power-management approaches for server clusters, such as vary-on/vary-off (VOVO) algorithms [7], cluster-based edge computers could become the ideal architecture to address these challenges. However, these and other energy-efficiency mechanisms have been developed largely in the context of large-scale data-center infrastructure [6], [17], [8], and typically depend on a model of energy consumption based on proxy metrics such as CPU utilization [18]. As we discuss in Section V, our data indicates that these models are inaccurate for edge computers with modern, high-density, general-purpose hardware. With future developments and the integration of specialized hardware into these edge computers, such as GPUs or single board AI modules [19], simple models will become infeasible for novel energy-efficiency mechanisms that control these heterogeneous infrastructures. Suppose, for example, a mechanism that optimizes energy consumption based on machine learning techniques such as reinforcement learning (which is also being explored in related cloud-operations research [18], [20]). A learning algorithm will continuously attempt different load balancing configurations given different workload types to minimize energy consumption. To facilitate such mechanisms in a real-world deployment, given the complexity of energy consumption of novel architectures, edge computers have to be natively *energy-aware*, i.e., they need access to energy consumption data in real-time to accurately determine the impact of different workloads on the energy consumption characteristics, learn from past observations, and continuously adapt to changes in the environment that may have an impact on energy consumption. Furthermore, control mechanisms need to be able to control the power state of nodes, i.e, turn them on and off.

In summary, we identify the following key requirements for edge computers for supporting the discussed scenarios:

- **Performance**: an edge computer needs to be able to host a variety of different services that are otherwise Cloud based, for example data analytics platforms. It therefore has to be powerful enough to host such services.
- **Reliability**: an edge computer needs to maintain low response times even in the face of unpredictable client loads and request arrival patterns inherent to IoT scenarios and Edge environments.
- **Portability**: an edge computer has to be portable, i.e., compact enough to be mounted on, e.g., a vehicle, or be carried around easily by a person, and not bound by static power or network infrastructure.
- **Energy efficiency**: because the edge computer is portable and needs to operate in resource-constrained environments, it has to be energy efficient. This is especially critical when the edge computer is battery powered.

[3]https://spire.com/

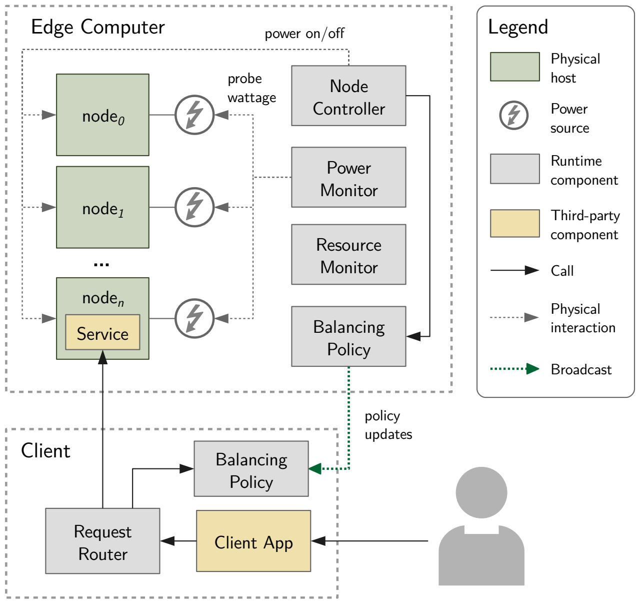


Fig. 1. Architecture for an energy-aware, cluster-based edge computer

III. Portable Energy-Aware Cluster-Based Edge Computers

We propose an architecture for a general-purpose energy-aware, cluster-based edge computer. In our reference architecture, a cluster-based edge computer is a closed system of n physical general-purpose compute nodes, and a power-management runtime that enables energy-awareness. We focus explicitly on enabling energy-awareness as opposed to general deployment and management of applications, as solutions already exist for these aspects [5], [3]. Figure 1 shows a simplified view of our proposed architecture.

Nodes host third-party services that are deployed via, e.g., container-based virtualization. The power-management runtime comprises components to monitor and control the nodes, and to provide energy-aware optimization mechanisms for task scheduling or load balancing. These runtime components are hosted on a separate auxiliary device that is part of the edge computer, but independent of power-management mechanisms and much smaller in scale than the compute nodes to minimize the impact on the overall power consumption (in our prototype we use two SBC, see Section IV). Between each node's power source, a power sensor is placed that allows the power monitor component to probe the current wattage of each node. A client device hosts the client side of the runtime: a request router that forwards requests to an actual physical node hosting the service, as well as third-party client application.

A. *Power-Management Runtime*

The power-management runtime comprises the following core components:

a) Node controller: the node controller decides at run-time, based on values received from the power monitor and resource monitor, to modify the power state of nodes, i.e., powering them on or off. The node controller relies on an optimization strategy implementation that aims to minimize energy consumption while maintaining responsiveness of the system. When the controller changes the power state of a node, it informs other runtime components about these changes.

b) Power monitor: the power monitor probes the nodes' power sensors, and provides an API for other runtime components to access node energy consumption data in real-time.

c) Resource monitor: the resource monitor tracks the resource utilization of nodes, and also provides an API to access the data in real-time. Specifically, it provides data on the CPU frequency (which may be dynamically adjusted by the underlying hardware), CPU utilization, memory usage, or I/O bandwidth of nodes.

d) Balancing policy: the balancing policy aims to optimize energy consumption and resource utilization by distributing request load across physical hosts. It uses the power and resource monitoring data to intelligently balance service requests among physical hosts. Because request dispatching actually happens on the client, the runtime broadcasts changes in the balancing policy to the client-side balancer. For example, when the node controller changes the power state of a node, the balancing strategy is informed and adapts accordingly (e.g., by removing a turned off node from the pool), and then forwards the information to the client.

B. *Client*

The client hosts the third-party client application and the client-side components of our power-management runtime. A client device could be an edge device such as a smartphone or an SBC.

a) Request router: In traditional cloud-based clusters, load balancing is typically done in the Cloud via, e.g., L7 switches or reverse proxies hosted on powerful machines that forward requests to the actual services depending on some strategy [21]. As discussed in Section II, we cannot rely on data-center scale routing hardware in a small portable edge cluster, and dedicating a node of the cluster to serve as reverse proxy would require this node to be online continuously, and therefore significantly increase the overall energy consumption. Furthermore, for high-performance applications, a load balancer that has the same computational resources as the nodes it manages would quickly create a bottleneck. Instead, in our architecture, we offload request dispatching to the client. The request router on the client device is a proxy for an

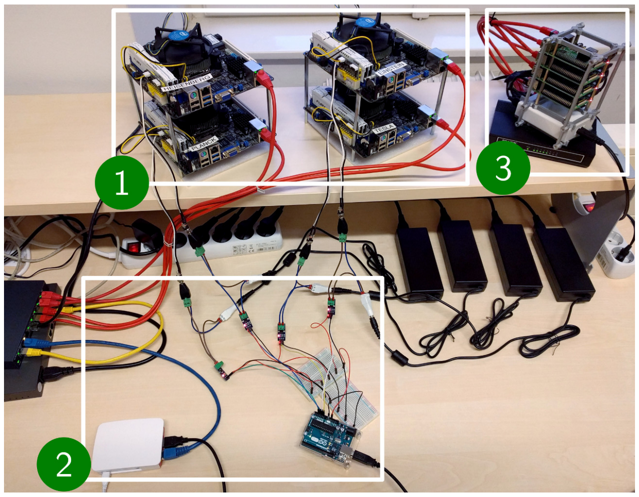


Fig. 2. Energy-aware cluster-based edge computer prototype

actual service request and is responsible for dispatching a request from the client app to a node that hosts the service. The runtime broadcasts changes in the balancing policy to the client-side request router. This includes the network addresses or specific service endpoints of currently active nodes. The requests are routed based on the current active balancing policy dictated by the edge computer, e.g., the percentage of requests that should be routed to a specific node or service endpoint. This way, the energy cost of load balancing is decentralized among clients. A drawback of this approach is the increased management complexity of balancing policies.

b) Balancing policy: The request router relies on a balancing policy defined by the power-management runtime. For each request, the request router queries the balancing policy for a node to send the service call to. A balancing strategy could simply be a round robin or weighted distribution among all currently powered nodes. Implementing and evaluating more complex strategies is out of scope of this paper, but part of our future work.

IV. Prototype

Based on our architecture, we have developed a prototype for an energy-aware portable cluster-based edge computer. The prototype comprises four compute nodes, networking and power infrastructure, a power and resource monitoring system, and a small cluster of client devices. Figure 2 shows our prototype infrastructure. It shows (1) the cluster compute nodes, (2) the monitoring infrastructure (with the Raspberry in the bottom left corner in a white case, and the Arduino in the bottom right), and (3) Raspberry Pis that serve as clients. Next, we describe each component in more detail.

A. *Compute Nodes*

The overall design goal of our prototype is to enable high-density computing, i.e., to have good compromise between compactness that enables portability, and performance to host typical Cloud-based services. We use Mini-ITX form-factor hardware, with server capabilities in mind, to build a high-density cluster. Each node is a mid-priced but powerful server computer made up of the following components:

- Motherboard: ASUS P10S-I Mini-ITX[4]
- CPU: Intel Xeon E3-1230 (4 cores, 8 threads)[5]
- RAM: 2x16GB Kingston HyperX Fury DDR4
- SSD: Intel SSD 600p 128GB M.2.[6]
- Power supplies: picoPSU-90 12V

Overall, each node takes up roughly 17x17x10cm of space.

To host applications, the compute nodes run a common server operating system, namely CentOS 7[7], with a standard configuration. Furthermore, the nodes run Docker CE[8] as application deployment platform. We later present energy consumption data to show that this hardware configuration could be powered by commodity battery packs.

B. Power-Management Runtime

There are many ways to implement the power-management runtime described in Section III. Specifically, there are a variety of ways to monitor energy consumption. Although we do not require real-time access to the energy consumption data for the evaluation of this prototype, we wanted a portable and cost-efficient solution to show the feasibility of our design, in particular because the power-management runtime is an integral part of the system and should therefore be portable and small scale. We deploy the power-management runtime on a Raspberry Pi Model 3 B running Raspbian 9. To monitor energy consumption, we developed a compact and cost-efficient monitoring infrastructure using an Arduino Uno, and four ACS712[9] Hall-effect-based linear current senors. It is important to note that we measure between the picoPSU power supply and the AC adapter because we intentionally do not want to include the power dissipation of the adapter (which we found in experiments to vary greatly between 10-25%). This is also the reason why we do not use consumer-grade power meters that typically sit between the power socket and the AC adapter. Reading the sensors is done via the analog inputs and a simple Arduino program. The Raspberry provides the Power Monitor, Resource Monitor, and Node Controller component. The Power Monitor accesses the power readings via the Arduino at a specified sample interval, and calculates from the raw readings the effective power in Watts (the data necessary for this calculation can be found on the ACS712 data sheet). The Resource Monitor uses standard Linux facilities to read resource utilization data from the nodes such as `/proc/stat` or `/proc/cpuinfo` which gives details about the current frequency and idle state of CPU cores. The nodes are only required to provide SSH access to the monitor; no other software needs to be installed.

C. Clients

For the evaluation of our system we need to generate client load. We cannot run clients on the compute nodes in parallel, because this would skew our energy and resource consumption results. Instead, we set up a small Raspberry Pi cluster where each node hosts a client application and a tool for generating load. We describe this in more detail in Section V. The client cluster is connected via a separate switch to the LAN of the cluster, and is powered by a separate power source. The Rapsberry Pis are Model 3 B Rev 1.2 and run Raspbain 9.

V. Experimental Evaluation

As stated in Section I, a major goal of this work is to gain insights on the performance and energy consumption characteristics of cluster-based edge computers, that can drive the design decisions for building energy-aware control mechanisms. To that end, we run several experiments to gather data on the general energy consumption characteristics of the cluster, as well as concrete performance data when running a data analytics application in different cluster configurations. We use the power-management runtime prototype to collect data on the current energy consumption and resource utilization of each node, and to test different balancing policies during real-world application workloads.

A. Application Scenario

Data analytics applications are considered a prime example that can benefit from edge computing [22], [23]. For our application we use the Apache MXNet deep learning library [24] with pre-trained models to perform image recognition tasks. Although specialized hardware exists for such applications, for example the Nvidia Jetson [19] platform optimized for deep learning, a general-purpose edge computer should be flexible enough to host a large variety of applications, and we therefore consider applications that make use of machine learning models a useful benchmark. Furthermore, we recognize that container-based deployments are common in modern application scenarios and DevOps workflows. We therefore use Docker to deploy instances of an MXnet Model Server[10], which exposes an MXnet classifier as a web service via an NGINX[11] web server. We then use a simple Python HTTP client application that we developed to send images to the exposed endpoint and await classification results. The clients receive from the power-management runtime the network addresses of endpoints and a weighted load balancing policy as indicated in Figure 3. We use the pre-trained SqueezeNet [25] model, a small-footprint model that has been used in other evaluations of image recognition applications [26].

[4]https://www.asus.com/Commercial-Servers-Workstations/P10S-I/

[5]https://ark.intel.com/products/52271/Intel-Xeon-Processor-E3-1230-8M-Cache-3_20-GHz

[6]https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/consumer-ssds/600p-series/600p-128gb-m-2-80mm-3d1.html

[7]https://www.centos.org/

[8]https://www.docker.com/

[9]https://www.allegromicro.com/en/Products/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS712.aspxx

[10]https://github.com/awslabs/mxnet-model-server

[11]https://www.nginx.com/

B. Idle and Offline Energy Consumption

The power characteristics of a system include offline and idle energy consumption that can be considered waste or overhead. An assumption sometimes made in simulations for energy-aware approaches is that powered-down nodes require no energy [27], [8]. Although this may be a reasonable assumption for the initial development of algorithms, as we will show, this is however not the case for real-world edge computers. Furthermore, dynamic frequency scaling techniques of modern CPUs have a significant impact on energy consumption. Especially during periods of low CPU usage, such as in idle phases, CPU throttling can significantly reduce energy consumption and heat dissipation. We investigate these aspects by measuring the energy consumption of nodes in both powered-down and idle states and report the aggregated measurement values.

C. Boot-Cycle

Modifying the power state of nodes at runtime comes at the cost of additional management complexity and overhead. While powering off a node may subsequently save energy necessary to otherwise maintain the node's idle state, the additional costs of the boot-cycle (power-off/power-on) have to be considered. Not only does booting or shutting down a node take time, which may have a significant impact on the system's responsiveness, it also consumes additional energy. There is an inherent tradeoff between keeping a node in idle mode and shutting it off. Quantifying this tradeoff is important for, e.g., VOVO [7] techniques, where the system determines at runtime to power nodes on and off. We run several experiments to examine this behavior in our prototype using our power-management runtime to boot and shutdown nodes. Specifically, we run experiments to a) quantify the duration of a boot-cycle, and b) quantify the energy consumption of a boot-cycle. A boot-cycle experiment is structured as follows: we start measuring the energy consumption, send a wake-on-lan packet to boot the node, and record the timestamp when the boot was initiated. We then wait for the node to appear by waiting until the TCP port of the Docker host becomes available and record the timestamp (which we consider the time at which workload requests can be sent to the node). The node then runs for 60 seconds in idle mode, after which we initiate a shutdown, and record for another 30 seconds. Because we cannot determine the concrete timestamp when the node is truly shut down, we estimate the timestamp from the offline energy consumption data. We perform each boot-cycle experiment ten times for each node and report aggregated results to include the measurement variations between nodes. We control for heat dissipation by waiting several minutes between experiments.

D. Energy Consumption during Workloads

We examine the system's energy consumption and responsiveness under varying client load and different load balancing policies. The main goal is to examine whether the overall system's energy consumption can be improved with specific load balancing policies given otherwise identical experiment parameters. Furthermore, we are interested in the trade-off between energy consumption and system responsiveness. To that end, we stress nodes by sending image classification requests from our Raspberry Pi clients. A client is a simple Python application that spawns several worker processes that send a randomly selected image (pre-loaded into memory) to a node and wait for a response. To make sure the work required for each request is the same, we pre-load five similar images, each with dimensions around 300x220- pixels, and a file size of around 150kB. In exploratory experiments we found that with these parameters, four Raspberry Pis each running eight worker processes to send requests can easily saturate a node's request capacity without slowing down the clients. For each request, we measure the round-trip time (RTT) in milliseconds with which we quantify the responsiveness of the system. To control the request rate we use a load generator that generates a pyramid arrival pattern, starting from 1 request per second (r/sec), to a peak of 300 r/sec, and down to 1 again. The load is adjusted every 20 seconds, and the experiment runs for 20 minutes. Figure 3 illustrates the process.

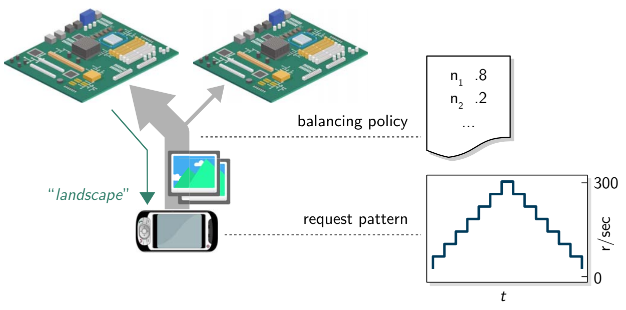


Fig. 3. Overview of load balancing experiment and parameters

In total, we run eight experiments with which we examine the system's performance and energy consumption under different weighted random load balancing policies. The first experiment sends all requests to n_1. The second experiment balances load between two nodes n_1, n_2 in a ratio of $\frac{n_1}{n_2} = \frac{.9}{.1}$, the third $\frac{n_1}{n_2} = \frac{.8}{.2}$, and so forth, until $\frac{.5}{.5}$ (round robin scheduling). In the last two experiments we perform round robin scheduling with three and four nodes respectively. Table I summarizes this. The columns indicate the percentage of requests that are routed to the respective node, or whether the node is kept offline.

We calculate for each experiment the total energy required for each node, and report the CPU utilization, CPU frequency (dynamically adjusted by the CPU), and statistics on the RTT.

VI. Results

We now present the results of the described experiments.

A. Idle and Offline Energy Consumption

We measure the energy consumption of each node during idle and offline state for several hours and report on the

TABLE I
LOAD BALANCING POLICIES FOR EXPERIMENTS

Experiment	n_1	n_2	n_3	n_4
1	100%	-	-	-
2	90%	10%	-	-
3	80%	20%	-	-
4	70%	30%	-	-
5	60%	40%	-	-
6	50%	50%	-	-
7	33%	33%	33%	-
8	25%	25%	25%	25%

aggregated results. Values for these states, and in particular their margin to energy consumption during peak loads, have a large impact on established energy consumption models [17], [28]. Figure 4 shows a summary of the power readings (read at 2 Hz from the sensors) in Watts over all nodes across our measurement periods.

The figures show the distribution of readings over the measurement duration. A node requires on average 2.3 W even when it is offline. The power supply draws a small amount of power when attached to the AC adapter. Furthermore, parts of the hardware are kept in a low-power sleep mode, to enable, e.g., wake-on-lan functionality. In idle mode, i.e., when a node was booted into the operating system and has a running Docker container with MXnet deployed but serves no requests, a node draws on average 9.7 W, with outliers up to 24 W. These high power readings can most likely be attributed to regular tasks performed by the operating system (such as flushing IO buffers for log files to disk).

B. Boot-Cycle Operations

Two metrics that will impact dynamic energy-aware control mechanisms are the duration and energy consumption of a node's boot-cycle, i.e., the time and energy it takes to power on and shut down a node. The two phases can be observed in Figure 5, which is a sample reading from one of our boot-cycle experiments. A detailed understanding of the boot-cycle is particularly important for dynamic cluster adjustment and VOVO techniques.

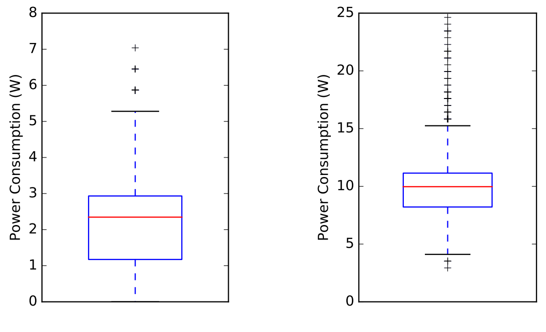


Fig. 4. Power consumption readings of nodes in offline and idle states

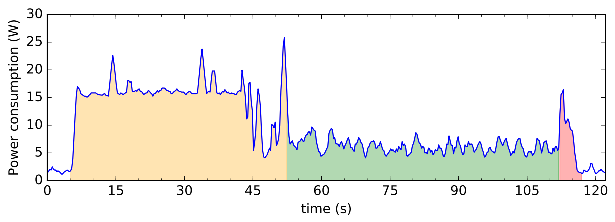


Fig. 5. Energy usage of a node during boot-cycle and idle period

The boot phase of the boot-cycle is marked yellow, and covers the period of time in which a node is powering up and the operating system is booted. The green area marks an idle period during which a node is ready to accept workload requests. Lastly, the red area marks the shutdown phase, where the operating system and device is shut down.

We estimate the overall energy consumption of the boot operation by solving Equation (1):

$$E_{boot} = T_{sample} \times \sum_{n=b}^{B-1} P(n) \tag{1}$$

where T_{sample} represents the interval between measurements in seconds (e.g., 0.25 for 4Hz), B represents the total number of measurements in the boot period (calculated from the time between the boot initiation and the time the Docker host became available), b is the measurement at which the boot process was triggered, and the function $P(n)$ returns the energy consumption of the node in Watt at a given measurement n.

Similarly, we calculate the energy consumption of the shutdown operation by Equation (2), where S and s are analogous to B and b for the shutdown period

$$E_{shutdown} = T_{sample} \times \sum_{n=s}^{S-1} P(n) \tag{2}$$

We find the end time of the shutdown period by looking for the first measurement value that is similar to the average offline energy consumption.

With these computations we can now find the overall duration and energy consumption of a node's boot-cycle. The results are presented in Figure 6 and Figure 7.

From our experiments we found that booting a node of our prototype takes on average 46.5 seconds and requires 688 Ws of energy. A node shutdown takes on average 5.3 seconds and 38 Ws of energy, making a complete boot-cycle require on average 726 Ws.

a) Boot-Cycle Energy Consumption Equality: With the energy consumption data gathered on node idle states, and the power and duration data for a boot-cycle known, we can now calculate the break-even point at which the total energy consumption for a boot-cycle is equal to that of a node in idle for a time t. Specifically, we are interested in finding how much time a node can remain in an idle state before it

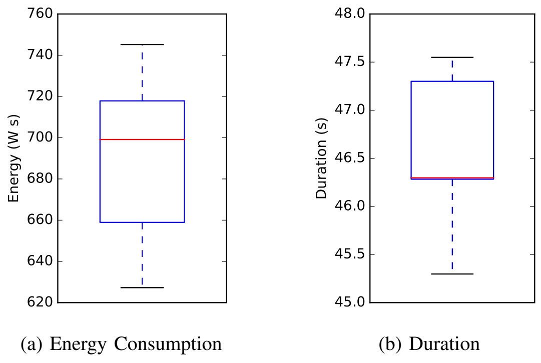


Fig. 6. Characteristics of boot operation

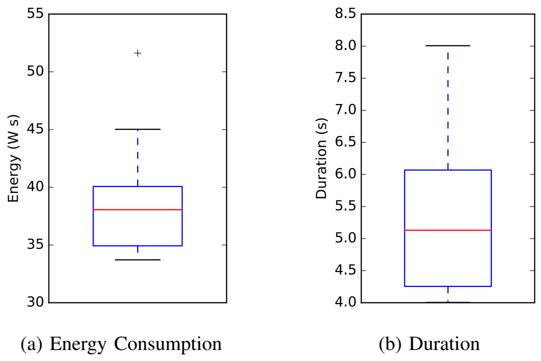


Fig. 7. Characteristics of shutdown operation

becomes more energy-efficient to perform a boot-cycle of that node. These values are necessary for a VOVO controller to make accurate decisions at runtime about when to turn a node on or off. We use Equation 3 for the calculation:

$$E_{boot} + E_{shutdown} = E_{idle}(t) \tag{3}$$

Where $E_{idle}(t)$ is the energy consumption of a node when it remains in an idle state for t seconds, which we compute using Equation 4, and where $E_{boot} + E_{shutdown}$ represents the total energy consumption of a boot-cycle.

$$E_{idle}(t) = \bar{P}_{idle} \times t \tag{4}$$

where $\bar{P}_{idle}$ represents the average energy consumption of a node in idle state (determined from our data), and t is the unknown duration in seconds that we want to calculate. To find t we divide the boot-cycle energy consumption (i.e., the left term of Equation 3) by $\bar{P}_{idle}$.

We compute for each boot-cycle measurement the break-even point (i.e., idle time t), and the results are presented in Figure 8. The break-even point averages around 109 seconds of idle time, without considering penalties on the responsiveness on the system.

C. Energy Consumption during Workloads

Table II shows the results of the load balancing experiments. We performed the workload experiments six times with similar results. Instead of aggregating the results, which would hide nuances of the experiment runs, we report a representative sample. Column 1 shows the index of the experiment. Column 2 shows the energy consumed in Wh over the entire course of the experiment for each node and in total. To get a more representative result, we list for offline nodes a value that corresponds to the average offline energy consumption over that period of time across all nodes. Column 3 shows the CPU utilization of each online node over time during the experiment. Column 4 shows the corresponding CPU frequency, where the value is the sum over all cores in MHz (8 cores with hyper-threading, where each core has a maximum frequency of 3700 MHz). Column 5 shows the request RTT in milliseconds (mean, 90th percentile, and 99th percentile) over time during the experiment. Note the y-axis scale changes from experiment 5. Column 6 shows details about the total requests processed during the experiment, and RTT statistics.

a) Energy consumption: Although experiment 1 shows the lowest energy consumption, it should be noted that a single node was not able to handle the peak load of 300 r/sec, but instead was capped at 250 r/sec, which should be taken into consideration when interpreting the results. Looking at the the sustained peak of CPU utilization and RTT after minute 8 shows that the node was busy working a congested request queue, and was able to pick up with the reduced request rate at minute 13. Experiment 2 exhibits similar but less extreme characteristics. In terms of energy consumption, overall we can see that there are only slight variations between the different load balancing policies in experiments 2 through 6. We also observe that adding a new node in experiment 7 does not increase the energy consumption significantly. Looking at the data, we can see that the CPU load is balanced across nodes, which has a cumulative effect on the energy consumption reduction of individual nodes, and this reduction is higher than the cost of adding an additional node. However, there are diminishing returns: at some point the energy cost of powering on additional nodes will be larger than the benefits of balancing load further, as made evident by experiment 8 (4 nodes round-robin scheduling), where the energy consumption is significantly increased compared to the other experiments.

b) Responsiveness: Overall we can see that, unsurprisingly, the RTT and system responsiveness is generally improved when balancing load between nodes. However, our results reveal interesting insights on the effect of dynamic frequency adjustment on the overall system responsiveness. For example, the first minutes of experiment 6-8 show that, when

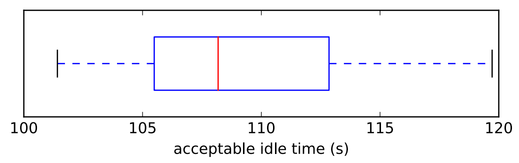


Fig. 8. Different break-even points of idle and boot-cycle energy consumption

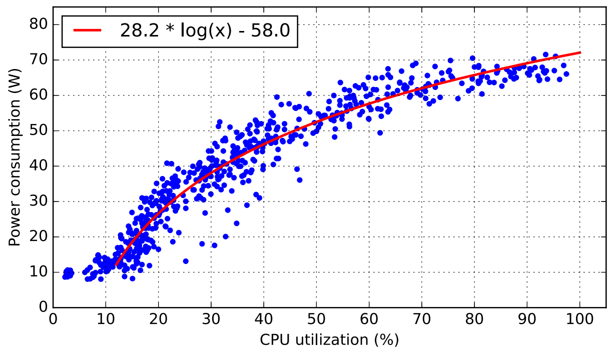


Fig. 9. Relation between CPU utilization and energy consumption

load is balanced in a way that nodes become underutilized, the reduced CPU frequency will lead to a worse responsiveness, despite more nodes being used than in experiment 6.

c) Power Consumption vs CPU Utilization: Assumptions often made when building simulation models for energy-efficiency algorithms in cloud computing are: a) that CPU utilization maps directly to energy consumption [18], [29], b) that there is a linear relationship between CPU utilization and energy consumption [17], [6], [28], and c) a low peak-to-idle energy consumption ratio [7], [18], [29]. These assumptions may be valid for data-center scale servers [17], but our results indicate that these assumptions will lead to inaccurate results when applied to edge computers. Figure 9 shows a scatter plot of the CPU utilization and energy consumption measurements of experiment 3 and node 1.

First, we observe that the relationship is more complicated. Until a CPU utilization of 12%, the energy consumption only rises very slightly. The relation then exhibits logarithmic growth. Second, we observe that for some regions of utilization, the energy consumption variance is very high. For example, around a utilization of 16%, the energy consumption varies between 9W and 30W, and at 90% it only varies between 63W and 70W. Third, we observe that the peak-to-idle energy consumption ratio (at around $\frac{70W}{10W} = 7$) is much higher in our prototype compared to what is assumed for data-center scale hardware (e.g., 1.36 or 1.45 [29]).

VII. Discussion

A. Research Questions

In Section I we introduced two research questions to investigate our cluster-based edge computer prototype in terms of energy consumption and responsiveness when employed for a data analytics application. We now discuss each question w.r.t. our results:

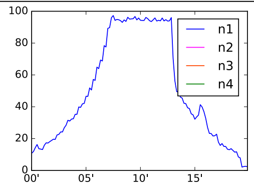
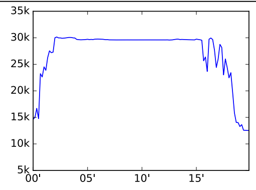
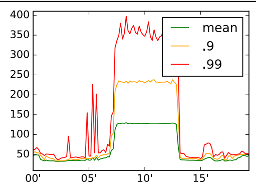
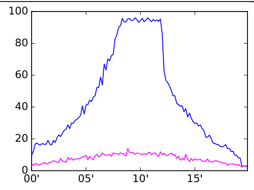
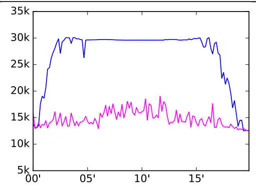
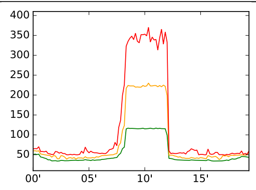
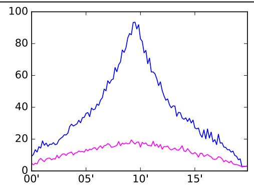
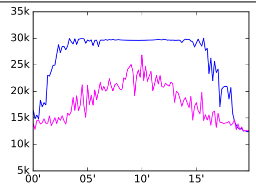
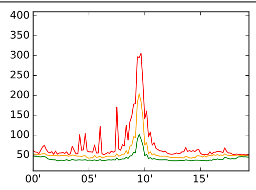
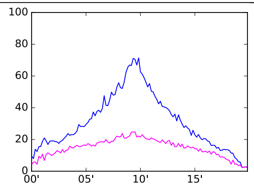
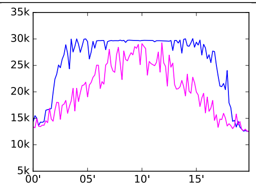
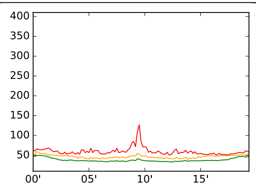
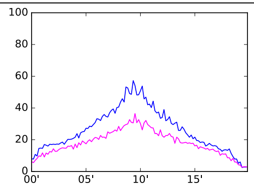
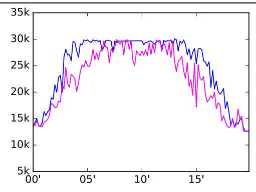
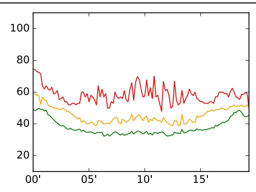
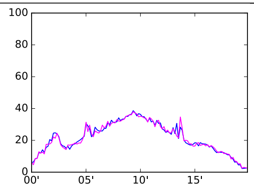
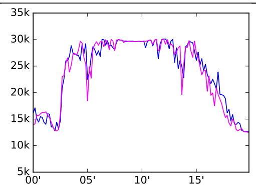
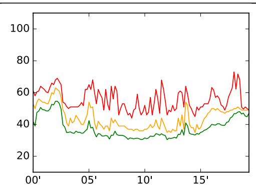
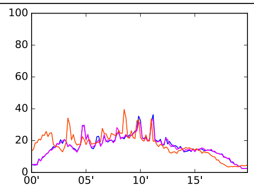
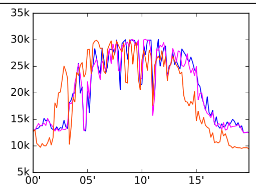
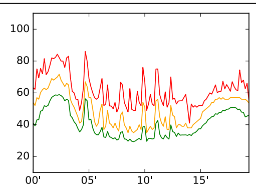
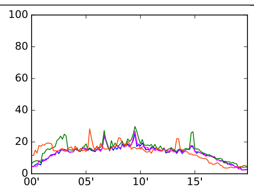
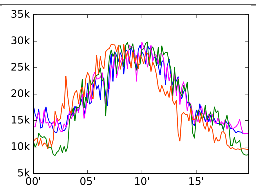
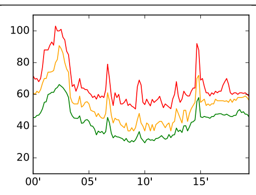
a) RQ1. What are the energy consumption characteristics of cluster-based edge computers?: An active node in our cluster draws power at a rate between 10-80W, depending on the resource utilization. Even when shut down, a node draws power at a rate of 0.5-2W. This large margin in energy consumption between idle and fully utilized nodes is due to the power management mechanisms of modern CPUs, in particular voltage and dynamic frequency adjustment. What this means for, e.g., simulation environments, is that using CPU utilization as a proxy for energy consumption will lead to inaccurate results. It is crucial to account for adjusted CPU frequencies based on the current utilization. Even then, estimates of energy consumption based on performance indicators alone may not be accurate. Because energy consumption is a complex process dependent on many different factors that are difficult to accurately measure in their entirety, different type of workloads may yield very different energy consumption patterns [30], which makes it necessary for power-management mechanisms to have real-time access to energy consumption data. Further investigation is needed to understand these factors, and how for example, integrating specialized hardware affects the overall energy consumption behavior.

Furthermore, we have seen that the boot-cycle of a node consumes on average as much energy as leaving it in idle for roughly 109 seconds. Although these results are highly dependent on the specific hardware and operating systems used, the provided insights in to the scale of the system, and can also be used as real-world parameters for simulation environments.

Overall, our results show that our prototype requires energy at a scale that can be managed with commodity portable energy supplies. For example, to get a rough idea, a typical 20 Ah lead-acid battery at 12V corresponds to 240 Wh. In our 20 minute experiment the cluster processed around 190 000 requests, and consumed roughly 20 Wh, with a power draw of about 80W if we consider the additional infrastructure energy consumption (switch and router, which averages at 4.1W). Even if we account for Peukert's law and the resulting diminished effective capacity, a single battery of this type could power the system for over an hour in the experiment conditions. This indicates that the general scale of the cluster is feasible for the application scenarios we presented.

b) RQ2. How do different cluster configurations and load balancing policies affect the responsiveness and energy consumption of cluster-based edge computers?: Cloud operations research has traditionally assumed high peak-to-idle energy consumption ratios of compute nodes [7], [18], [29] Our initial intuition was therefore that the idle energy consumption of nodes would be so high, that the optimal configuration would be to fully utilize each node until the application responsiveness drops below a specific threshold (e.g., a maximum RTT value). Our data show the opposite, which we largely attribute to the power-management mechanisms of high-density compute hardware, i.e., dynamic frequency adjustment, and the relationship between CPU frequency and energy consumption. When load is balanced among cores, the CPU utilization is low, and therefore the frequency is scaled down. There is a delicate trade-off between CPU frequency, energy consumption, and responsiveness, which greatly increases complexity for energy-aware load balancing or scheduling techniques, and is likely to have different characteristics for different types of applications and workloads. An approach to handle

TABLE II
RESULTS OVERVIEW OF LOAD EXPERIMENTS IN DIFFERENT CLUSTER CONFIGURATIONS

#	Energy (Wh)	CPU Utilization (%)	CPU Frequency $\sum MHz$	RTT (ms)	RTT (ms) Stat.
1	n_1 15.063 n_2 0.644 n_3 0.644 n_4 0.644 $\sum$ **16.995**				requests 184k $\bar{x}$ 79.5 $Q_{.9}$ 183 $Q_{.99}$ 313 max 677
2	n_1 14.071 n_2 4.039 n_3 0.644 n_4 0.644 $\sum$ **19.398**				requests 190k $\bar{x}$ 64.5 $Q_{.9}$ 143 $Q_{.99}$ 287 max 654
3	n_1 12.775 n_2 5.084 n_3 0.644 n_4 0.644 $\sum$ **19.147**				requests 191k $\bar{x}$ 42.7 $Q_{.9}$ 52 $Q_{.99}$ 161 max 415
4	n_1 11.491 n_2 6.534 n_3 0.644 n_4 0.644 $\sum$ **19.313**				requests 191k $\bar{x}$ 36.6 $Q_{.9}$ 48 $Q_{.99}$ 62 max 237
5	n_1 10.165 n_2 7.922 n_3 0.644 n_4 0.644 $\sum$ **19.375**				requests 191k $\bar{x}$ 35.4 $Q_{.9}$ 46 $Q_{.99}$ 59 max 161
6	n_1 8.797 n_2 9.326 n_3 0.644 n_4 0.644 $\sum$ **19.411**				requests 191k $\bar{x}$ 34.9 $Q_{.9}$ 46 $Q_{.99}$ 60 max 115
7	n_1 6.477 n_2 6.924 n_3 5.345 n_4 0.644 $\sum$ **19.391**				requests 191k $\bar{x}$ 37.1 $Q_{.9}$ 52 $Q_{.99}$ 68 max 115
8	n_1 5.423 n_2 5.856 n_3 4.454 n_4 5.794 $\sum$ **21.527**				requests 191k $\bar{x}$ 38.8 $Q_{.9}$ 54 $Q_{.99}$ 76 max 134

this complexity would be to turn off any internal power-management mechanisms, but we argue that, instead, a multi-layer view on power-management is needed to fully address the challenges. Energy-efficiency systems for clusters need to understand the behavior of the underlying hardware's power-management algorithms, and cooperate with these lower-level mechanisms to provide optimal balancing of workloads. These are particularly important observations for future work on building clusters of hardware that already makes use of energy efficiency techniques (such as dynamic frequency scaling), but further investigation is needed to fully understand this relationship with respect to different workload types and node configurations.

B. Limitations

There are some limitations in our experiment setup that should be considered when interpreting the data. First, the ACS712 sensors have a total output error of 1.5%, and readings are subject to inherent noise. For this first evaluation, we have only taken simple measures to control for the sensor inherent noise. We sample at a rate of 2Hz-4Hz, which is a relatively high rate compared to the length of each experiment. For future work, we plan to experiment with other measurement devices and signal processing techniques to better control sensor errors and noise. Second, although all nodes have the exact same hardware specifications, BIOS and operating system setup, we observed some variations in the dynamic frequency adjustment between nodes. Specifically, nodes showed significantly different CPU frequencies when in idle mode (in a range of 1000-1500 MHz), which may skew results that include idle energy consumption. We were unable to determine the definitive reason, but we suspect it has to do with slight variations in 12V system voltage attributed to the power supplies, as well as manufacturing differences in CPU and circuit boards.

VIII. Related Work

As mentioned in the introduction of this paper, power management techniques are critical for portable cluster-based edge computers. Existing literature has focused in the last years on reducing the operational costs of data centers. Beloglazov et al. [8] propose an energy-efficient load balancing algorithm to dynamically migrate virtual machines (VMs) between hosts in a Cloud data center. The authors evaluate a novel heuristic algorithm that adapts its behavior based on an analysis of historical data from VMs resource management. A similar approach is suggested in [31] where a dynamic VM selection algorithm to migrate VMs from overloaded or underloaded hosts to minimize the energy consumption and maximize the Quality of Service (QoS). Multiple other solutions have been proposed based on machine learning [32], [27] and metaheuristic algorithms [33]. Despite proposing different variation of energy-efficient optimization algorithms, in all scenarios the devices are connected to an infinite energy source, and the power models are built on assumptions that are suboptimal when dealing with hardware presented in this paper.

One of the first energy-efficiency techniques for cluster-based systems is presented in [34]. The authors propose a load balancing and unbalancing decision making algorithm taking into consideration the workload on the cluster and the energy consumption costs of turning on and off nodes. Only the energy consumption in idle and maximum utilization of a node are taken into consideration when taking decisions regarding the state of a node (i.e., turn the node on or off). However, as demonstrated in this paper, in order to save more energy consumption and take better decision the boot-cycle of a node should be considered.

Researchers have proposed multiple surveys on energy-awareness focusing on understanding how to minimize the energy consumption of each individual component in a Cloud data center. The most relevant survey is presented in [35] and identifies causes and problems of high energy consumption in Cloud data centers. Based on this, they propose a taxonomy of energy-efficient design of computing systems that will help developing energy-aware systems.

Compared to related work, our work provides a deeper understanding of how to minimize energy consumption in an portable cluster-based edge computer, and lays the foundation for developing energy-efficiency mechanisms which builds upon existing algorithms, but consider limited energy supply such as batteries. Energy-awareness and load-balancing techniques [6] from cloud computing may have limited applicability in this context, as the operational scale is vastly different. Portable edge computers do not have available the same dedicated hardware to operate optimizations and request routing as cloud data centers.

IX. Conclusion and Future Work

By bringing Cloud services and data analytics closer to the edge of the network, edge computers can greatly improve many application areas. In this paper, we made the case for portable energy-aware, cluster-based edge computers usable for data analytics in forward-deployed scenarios. These types of edge computers must be portable, i.e., compact enough to be carried by a person or mounted on vehicle, energy-aware because the computers may be battery powered, and cluster-based to provide the necessary reliability to deal with varying workloads at the Edge. Efficiently operating such edge computers requires advanced power-management strategies, such as energy-aware load balancing or task scheduling. However, developing such strategies is challenging because we currently lack a deep enough understanding of how different cluster configurations, balancing policies, workload types and other factors affect the overall energy consumption of real-world systems at this scale. To address this, we presented a design, prototype and first experimental evaluation of a portable energy-aware, cluster-based edge computer. We showed the general feasibility of our prototype, and reported the results of multiple experiments on the energy and resource consumption characteristics of the cluster in the context of an image recognition application. The results show that energy-efficiency in these small clusters involves a delicate trade-off between system responsiveness and energy consumption, and reveal some of the intricacies of modern, high-density compute hardware. Also, our results indicate that previous models used for simulating energy consumption of data-center scale servers may not be applicable to edge computers. Further research is therefore needed to assert whether energy-efficiency techniques developed for cloud computing are applicable to edge computing. Overall, the data we gathered are valuable for researchers working on the design and simulation of algorithms for energy-aware edge computers. As a general conclusion, according to the results of our study we have observed that by knowing the boot-cycle and shutdown costs (i.e., the energy consumption needed to perform this actions and the duration), the idle energy consumption and the responsiveness of the system a energy-aware scheduling of tasks on the edge computer will extend the running time of the device.

For future work, we plan to use the results we have obtained to develop an energy-aware task scheduler and cluster controller to run on the proposed infrastructure. The goal is to optimize the overall energy efficiency of the cluster by dynamically scaling nodes at runtime (e.g., using VOVO[7] techniques) and balancing load between nodes in a way that minimizes energy consumption. We also plan to extend the cluster with specialized hardware such as GPUs and evaluate more complex application scenarios with different and heterogeneous types of workloads.

ACKNOWLEDGMENT

This work is partially supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłukodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and Industrial Automation; the Austrian Federal Ministry of Science within the CPS/IoT Ecosystem project; and TU Wien research funds. We also thank Alexander Knoll and Jacob Palecek for supporting us in setting up the hardware and the monitoring infrastructure.

REFERENCES

[1] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, oct 2009.

[4] K. Lorincz, D. J. Malan, T. R. F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton, "Sensor networks for emergency response: challenges and opportunities," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 16–23, Oct 2004.

[5] G. Lewis, S. Echeverria, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *2014 IEEE Military Communications Conference*. IEEE, oct 2014, pp. 1440–1446.

[6] V. Mathew, R. K. Sitaraman, and P. Shenoy, "Energy-aware load balancing in content delivery networks," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 954–962.

[7] E. N. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-Efficient Server Clusters," in *Power-Aware Computer Systems*, B. Falsafi and T. N. Vijaykumar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 179–197.

[8] A. Beloglazov and R. Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 2010, pp. 826–831.

[9] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos, "Cloud computing: survey on energy efficiency," *ACM Computing Surveys*, vol. 47, no. 2, pp. 1–36, dec 2014.

[10] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, jan 2016.

[11] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2018.

[12] Y. Elkhatib, B. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, and E. Riviere, "On Using Micro-Clouds to Deliver the Fog," *IEEE Internet Computing*, vol. 21, no. 2, pp. 8–15, mar 2017.

[13] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*. IEEE, 2015, pp. 9–16.

[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[15] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*. ACM, 2012, pp. 29–36.

[16] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *AFIN 2014 : The Sixth International Conference on Advances in Future Internet*. Citeseer, 2014.

[17] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster Computing*, vol. 12, no. 1, pp. 1–15, mar 2009.

[18] J. Wu, X. Xu, P. Zhang, and C. Liu, "A novel multi-agent reinforcement learning approach for job scheduling in grid computing," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 430 – 439, 2011.

[19] D. Franklin, "Nvidia® jetson^{TM} tx1 supercomputer-on-module drives next wave of autonomous machines," *Parallel Forall. NVIDIA Corporation*, vol. 12, 2016.

[20] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-Efficient Virtual Machines Consolidation in Cloud Data Centers Using Reinforcement Learning," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, feb 2014, pp. 500–507.

[21] N. Grozev and R. Buyya, "Multi-cloud provisioning and load distribution for three-tier applications," *TAAS*, vol. 9, pp. 13:1–13:21, 2014.

[22] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the internet of things," *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015.

[23] S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, and R. Prodan, "A serverless real-time data analytics platform for edge computing," *IEEE Internet Computing*, vol. 21, no. 4, pp. 64–71, 2017.

[24] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[25] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[26] V. Ishakian, V. Muthusamy, and A. Slominski, "Serving deep learning models in a serverless platform," in *IC2E 2018-IEEE International Conference on Cloud Engineering*, 2018.

[27] J. L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres, "Towards Energy-aware Scheduling in Data Centers Using Machine Learning," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking - e-Energy '10*, ser. e-Energy '10. New York, New York, USA: ACM Press, 2010, p. 215.

[28] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "A Framework and Algorithm for Energy Efficient Container Consolidation in

Cloud Data Centers," *Proceedings - 2015 IEEE International Conference on Data Science and Data Intensive Systems; 8th IEEE International Conference Cyber, Physical and Social Computing; 11th IEEE International Conference on Green Computing and Communications and 8th IEEE Inte*, pp. 368–375, 2015.

[29] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, sep 2012.

[30] D. Guyon, A.-C. Orgerie, and C. Morin, "An experimental analysis of paas users parameters on applications energy consumption," in *IC2E 2018-IEEE International Conference on Cloud Engineering*, 2018, pp. 1–7.

[31] R. Yadav, W. Zhang, H. Chen, and T. Guo, "Mums: Energy-aware vm selection scheme for cloud data center," in *2017 28th International Workshop on Database and Expert Systems Applications (DEXA)*, Aug 2017, pp. 132–136.

[32] R. Shaw, E. Howley, and E. Barrett, "An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec 2017, pp. 61–66.

[33] M. Bakalla, H. Al-Jami, H. Kurdi, and S. Alsalamah, "A qos-aware and energy-efficient genetic resource allocation algorithm for cloud data centers," in *2017 9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Nov 2017, pp. 244–249.

[34] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," in *Workshop on compilers and operating systems for low power*, vol. 180. Barcelona, Spain, 2001, pp. 182–195.

[35] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," ser. Advances in Computers, M. V. Zelkowitz, Ed. Elsevier, 2011, vol. 82, pp. 47–111.