Intelligent Sampling: A Novel Approach to Optimize Workload Scheduling in Large-Scale Heterogeneous Computing Continuum

Victor Casamayor Pujol
Distributed Systems Group
TU Wien, Austria
v.casamayor@dsg.tuwien.ac.at
0000-0003-2830-8368

Andrea Morichetta
Distributed Systems Group
TU Wien, Austria
a.morichetta@dsg.tuwien.ac.at
0000-0003-3765-3067

Stefan Nastic
Distributed Systems Group
TU Wien, Austria
s.nastic@dsg.tuwien.ac.at
0000-0003-0410-6315

***Abstract*—Scheduling workloads on large-scale infrastructures, such as in the Edge-Cloud continuum is a challenging task. Usually, the scheduling algorithm considers only a limited sample of the infrastructure nodes, typically obtained through random sampling. The sampling reduces the number of nodes, which need to be evaluated in the scheduling pipeline, making the scheduling process more saleable. Unfortunately, current sampling approaches become largely inefficient when the infrastructure is heterogeneous and specific, scarce node characteristics are required to successfully execute a workload. Computing continuum infrastructures are heterogeneous, hence, we need to re-think the sampling process to keep it viable at scale while also being able to identify and leverage the heterogeneity of the Edge-Cloud continuum resources. In this article, we present Intelligent Sampling – a novel technique for improving sampling in large-scale and heterogeneous infrastructures. We develop a model for any heterogeneous infrastructure. Based on this model, we provide a method to sample the infrastructure nodes more accurately, considering the specific task at hand. Finally, we leverage the Alibaba PAI dataset to show that our approach is 2.5x times more accurate compared with other state-of-the-art sampling mechanisms while retaining comparable performance and scalability.**

***Index Terms*—Computing continuum, Intelligent sampling, Workloads scheduling, Heterogeneous infrastructure model**

I. Introduction

The emerging computing paradigm, the computing continuum [1], [2], merges all computing tiers, from the IoT or Edge computing to Cloud computing or High-Performance Computing (HPC). This will allow future applications to selectively configure their underlying infrastructure to provide the best services to their users. Nevertheless, reaching this novel paradigm is full of challenges [3], which are accentuated if the computing continuum paradigm also embraces the Serverless capabilities [4], [5]. It requires managing applications' needs autonomously, calling for a set of methods and techniques that reduce the management burden while being fast and effective.

Scheduling has been under research for many years now. It is an NP-hard problem [6], hence optimal solutions can be unpractical in large-scale scenarios, and developing heuristics [7] or meta-heuristics [8] to solve the problem is a common approach.

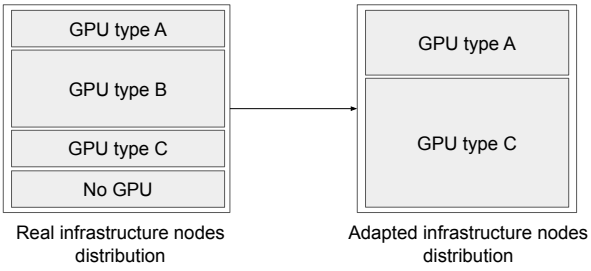


Fig. 1: Infrastructure virtual adaptation. The left schema shows the infrastructure nodes distribution with respect to the types of GPUs. The right represents the same infrastructure virtually adapted to workload requirements.

Computing Continuum systems, such as Edge-Cloud continuum, are distributed and large-scale. These characteristics preclude fully centralized scheduling architectures [9]. For example, the Edge's capacity to lower the latency of applications, given its proximity to the data generation is undermined if the provision of the infrastructure resources is centralized in the Cloud and requires long-range connections.

Scheduling on large-scale and distributed computing infrastructures is typically a 2-step process that deals with the infrastructure scale and relieves the scheduler from having to score all the nodes. The first step can be performed through sampling [10] or auctioning [11]. The first method returns a sampled set of nodes from the infrastructure to the scheduler. The second changes the angle by transforming the nodes into agents able to bid/steal workloads, making them responsible for the scheduling process. Random sampling produces excellent results when the infrastructure is assumed to be homogeneous [10], given that the probability of obtaining a node with the expected characteristics is 1. However, when the infrastructure is heterogeneous, its efficiency deteriorates proportionally to the heterogeneity of the cluster. Observing Figure 1, if the workload to schedule can only support having GPUs of type A and C, then the probability of finding any

of these nodes from a random sampler is below 0.5. Bidding on workloads have also shown promising results but it cannot guarantee a specific placement for workloads [12].

The main contribution of this article is two-fold. On the one side, we propose an Intelligent Sampling mechanism to improve the node sampling process for large-scale and heterogeneous computing infrastructures. Intelligent Sampling is able to provide a sample of the most convenient nodes for the incoming workload based on the historical usage of the infrastructure. On the other side, we develop a model for heterogeneous infrastructures, which is leveraged by the Intelligent Sampling mechanism. The model can abstract infrastructure complexities to the scheduler while providing infrastructure-wise enhancement capabilities due to its awareness of the heterogeneity characteristics. The developed model is applied to the Alibaba PAI dataset [13], and the Intelligent Sampling mechanism is evaluated and compared with other state-of-the-art sampling approaches obtaining almost 2.5X more accurate samples while retaining a comparable performance and scalability. The code and results can be found in our GitHub repository[1].

This work is conducted within the Polaris project [14]–[16]. Polaris and all its frameworks and tools a fully open-sourced and publicly available in GitHub[2]. The Polaris project is part of the Centaurus Infrastructure initiative[3][4], which is hosted by the Linux Foundation[5]

This article is organized as follows. Section II provides an overview of the Intelligent Sampling mechanism and the required background for our work. Then, Section III presents the developed infrastructure model, and Section IV formally describes the Intelligent Sampling mechanism. Section V presents a model use case by implementing it in a real dataset and evaluates the performance of Intelligent Sampling. Next, the related work is presented in Section VI, and finally, Section VII presents the future work and the conclusions for this article.

II. Approach overview & Background

A. Overview of Intelligent Sampling

Intelligent Sampling aims to enable workloads' scheduling process on large-scale, distributed, and heterogeneous computing infrastructures by providing the scheduler with a set of conveniently selected nodes. Further, Intelligent Sampling brings the possibility of leveraging the proposed infrastructure model, which exposes to the sampling mechanism the characteristics of the heterogeneity types.

Following Figure 2, we see that matching the infrastructure's characteristics with the workload's requirements is an NP-Hard problem. Our approach abstracts both the infrastructure characteristics and the workload requirement to find confluent factors. We obtain the abstraction of the workloads by leveraging an apriori profiling methodology for workloads, see next subsection II-B, which drastically reduces their differences by grouping them into profiles that exhibit similar behaviors. Similarly, we present in this article a model for the infrastructure, see subsection III-B, that characterizes and classifies heterogeneous computing resources. Only through modeling the infrastructure and profiling the workloads, we can tackle large-scale, heterogeneous, and distributed computing infrastructures.

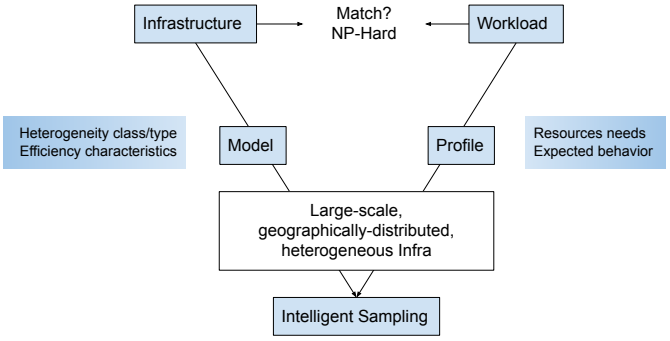


Fig. 2: Overview of the process required to develop Intelligent Sampling.

As shown in Figure 2, by leveraging the infrastructure model and the workload profile, we can develop the Intelligent Sampling mechanism for large-scale, geographically distributed, and heterogeneous computing infrastructure. Intelligent Sampling consists of providing a sample of adequate infrastructure nodes for the incoming workload. This is done by virtually reconfiguring the infrastructure nodes distribution. Hence for the sampler, the available nodes to select fulfill the requirements of the incoming workload. Figure 1 shows the actual heterogeneity distribution of the computing infrastructure, and how the sampler sees it, Figure 1 right side.

As an example, imagine a computing infrastructure with computing nodes containing GPUs. Each node has a type of GPU and there are 3 types of GPUs. Then, from the history of executions, we know that the workload's profile works well with only 2 out of the three available GPUs. Hence, the Intelligent Sampling method will only sample nodes from those containing adequate GPUs.

B. Background information on profiling

Intelligent Sampling leverages the *PolarisProfiler* to abstract within profiles the incoming workloads to schedule, the work is published in this same proceedings [17]. This is a profiling methodology for workloads based on static, apriori metadata. The aim is to generate profiles that include homogeneous workload traces from the perspective of infrastructure usage and seamlessly map new workloads to them. Figure 3 provides a high-level representation of the *PolarisProfiler* concept. It shows the mapping of apriori metadata features from new workloads to existing workload profiles. The Profile classifier module performs the mapping, while the Profile generator

[1]https://github.com/polaris-slo-cloud/intelligent-sampling
[2]https://github.com/polaris-slo-cloud
[3]https://www.centaurusinfra.io/
[4]https://github.com/CentaurusInfra/polaris
[5]https://www.linuxfoundation.org/

derives workload patterns from infrastructure usage. It also shows how to trace metadata back to workload pattern types. At the profile creation time, the method exploits the similarity in the workloads' execution patterns using unsupervised learning techniques. This task is performed by the *profile generator*. The profiles generated expose apriori, static metadata for matching new workloads. The *profile classifier* assigns the profile that best fits the profile metadata characteristics. The profiles contain rich and specific insights into workload usage patterns that can be used for improving diverse resource provisioning strategies, from scheduling to bootstrapping to proactive monitoring.

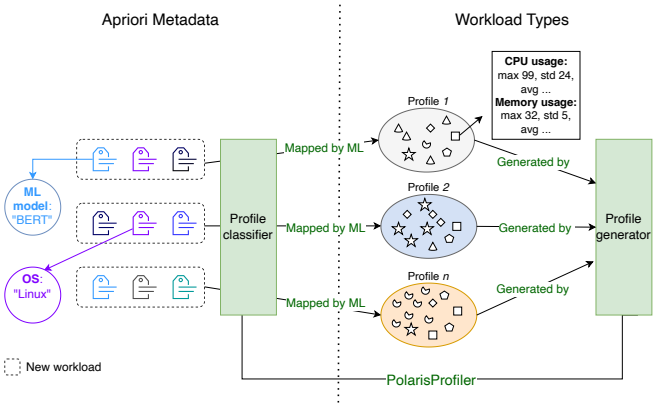


Fig. 3: Concept of the PolarisProfiler model.

Profiling is part of the Polaris project, available at the project GitHub's page[6]. The profiling methodology has been adapted and applied to the Alibaba PAI data set for this article. It is essential to consider that profile similarity is based on unsupervised methods leading to sporadic miss-classifications and that some workloads are considered outliers, i.e., they do not belong to a profile. For the performance evaluation (subsection V-B) we use the Alibaba PAI dataset as the baseline. This means that if the profiling approach produces miss-classifications this also negatively affects the sampling results.

III. Modelling Heterogeneous Infrastructure

A. Infrastructure heterogeneity definition

The Cloud-Edge continuum has many sources of heterogeneity that affect infrastructure performance. However, in this work we focus mainly on hardware, proposing a model to harness its heterogeneity. We consider heterogeneity that accounts for static differences in the hardware's infrastructure, which is not frequently seen in the literature. This refers to differences in the behavior of similar (interchangeable) hardware components, for instance, the type of CPU cores, the GPU types, etc. It is important to remark that this heterogeneity might not be designed nor desired, but the progress in hardware development brings it to any computing infrastructure. As can be seen from Table I, the different GPU types have a variety of release dates, which influence the final cluster configuration. Interestingly, one can find many references to heterogeneous infrastructure in the literature, such as in [18] or [19]. However, in most cases, heterogeneity comes from the state of the infrastructure's consumable resources, so its heterogeneous configuration of nodes changes with execution time.

[6]https://github.com/polaris-slo-cloud/Profiling

B. Heterogeneity model

Several infrastructure characteristics can be defined as heterogeneity classes, such as CPU type, GPU type, TPU, FPGAs, etc. We provide a simplified relative model of the infrastructure's heterogeneity. First, we model within the scope of Intelligent Sampling, and we frame this within a scheduling pipeline. Hence, we value a fast and differentiated model classification. And second, the differences between components of a heterogeneous trait can be very subtle, and managing this detail is costly. Hence, the model aims to enable Intelligent Sampling and simplify and enhance the usage of the computing infrastructure.

As shown in Figure 4, the heterogeneity model has two layers. The first layer considers whether the infrastructure node has a specific heterogeneity class. This is a simple distinction, but it allows a fast classification of resources. Simply put, an ML training workload should only consider GPU nodes, and this first layer will enable us to capture similar requirements. Further, this classification is cost-efficient for the IoT or Edge tiers where the number of devices is large, and they can have very specialized hardware. This first layer brings a dichotomy to the model's structure, so we differentiate between *core heterogeneity classes* and *additional heterogeneity classes*. Indeed, all nodes from a computing infrastructure have a CPU core or RAM, hence these are *core heterogeneity classes*. However, some classes might or might not be present in a node, such as a GPU, storage, or sensors. These are *additional heterogeneous classes*. Hence, the first layer of the model only evaluates if an additional heterogeneity class is included in the node.

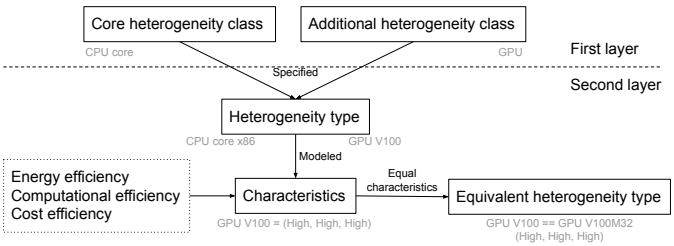


Fig. 4: View of the different elements considered in the proposed heterogeneity model, a simplified example in grey.

The second layer (Figure 4) provides a more detailed description of the heterogeneity class being considered, which we call heterogeneity type. Further, this second layer is able to model arbitrary heterogeneity classes. We follow a similar approach done in [20]. However, we go one step beyond, considering computation, energy, and cost efficiency as the

three characteristics of any heterogeneity type. We classify heterogeneity types for each characteristic as high or low compared to the other types in the infrastructure. Simply put, we can model a type of GPU as highly computationally efficient while being low on energy and cost efficiency. It is important to remark that this has to be done relative to the nodes present in the infrastructure. Otherwise, the optimization capabilities of the scheduling pipeline are not correctly tailored to the infrastructure. To do so, we leverage the mid-range value (M), see equation 1, to split each characteristic of any heterogeneity type as "High" (if above or equal to M) or "Low" (if below M). We use M conversely of a quantile-based separation because M is agnostic to the distribution of data points, i.e., the classification as "High" or "Low" only depends on the range of values obtained for the heterogeneity type.

$$M = \frac{\max(x) - \min(x)}{2} \tag{1}$$

This abstraction reduces the heterogeneity of the infrastructure and allows us to define equivalent heterogeneity types. Simply put, two different GPUs classified as high on computational efficiency and low on energy and cost efficiency can be considered the same type. This eases the management of the infrastructure by reducing its heterogeneity while accounting for it. Indeed, the granularity chosen is coarse, and it is possible to have a more fine-grained division, i.e., "High", "Mid-high", "Mid-low", or "Low". However, for the scope of this article, the selected coarse granularity is enough; in future work, when refining the approach and its implementation, other divisions can be considered and evaluated.

IV. Intelligent Sampling

A. Definitions

- **Heterogeneous infrastructure**: We define a computing infrastructure as a set of computing nodes: $Infra = \{n_1, \dots n_k\}$. Infrastructure nodes can have one or more heterogeneity classes (Ht), which by definition are constant in time, i.e., at time t_0, n_i has Ht_a and Ht_e, and this holds true for any other time t_i considered. Further, we assume that the number of nodes k is much larger than the number of different heterogeneity traits l in the infrastructure ($k >> l$). Hence, we can group the computing nodes by their heterogeneity class: $\forall\, Ht \in Infra : \mathbf{H}t_i = \{n_\alpha, \dots n_\gamma\}$. Notice that bold characters identify sets while the plain character refers to the shared property of the set. This grouping enables the querying of nodes with respect to their heterogeneity class. Notice that we are not differentiating at this stage if the heterogeneity class is *core* or *additional*. Further, consider that given the definition provided, a node can be found in more than one group.
- **Heterogeneous type**: $\vec{ht}_i$ is defined as a three dimensional vector, where i identifies the heterogeneity type. The class is determined unequivocally by the type. Each vector dimension specifies the type's characteristics in terms of energy-efficient, computation-efficient, and cost-efficient. Hence, following the defined model, 1 stands for high efficiency, while 0 represents low efficiency. We leverage the vector representation to organize the infrastructure nodes, i.e., all nodes belonging to a heterogeneity class ($\mathbf{H}t_i = \{n_\alpha, \dots n_\gamma\}$) are now classified accordingly to their characteristics in a three-dimensional array. Hence, each possible combination of the characteristics vector specifies nodes with equivalent heterogeneity types. We can express it as follows: $\vec{\mathbf{h}t} = (x, y, z) = \{n_\alpha, \dots n_\delta\}$. Notice that the (i) from $\vec{ht}$ has disappeared as we are now considering equivalent heterogeneity types.
- **Execution**: $R_n(W)$ is defined as the execution of the workload W in a specific node n of the infrastructure.
- **History of execution**: is defined as the value of a counter C that is increased if the execution of the pair n, W is terminated within the expected requirements for W.
- **Workload profiling system**: we define a profile as a group of workloads $\mathbf{P} = \{W_0 \dots W_m\}$) where $\forall\ W \in \mathbf{P}$ workload characteristics at run-time are differentiable from workloads (W') of other profiles $\mathbf{P}$'. This implies that the requirements for workloads within the same profile are similar. Hence, we can use its profile to refer to the incoming workload that requires to be scheduled without losing knowledge of its requirements and characteristics.
- **Random sampling**: consists of sampling nodes over an infrastructure. In other words, randomly selecting a set of nodes $\mathbf{N}$ from $Infra = \{n_1, \dots n_k\}$. The sample size is a parameter that can be configured and in general: $|\mathbf{N}| << |Infra|$
- **Intelligent Sampling**: we define it as the ability of randomly sampling a set of nodes from $\mathbf{H}t_j = \{n_\alpha \dots n_\gamma\}$, such that the heterogeneity trait (Ht_j) is adequate for the incoming workload belonging to profile $\mathbf{P}$. Similarly, we consider Intelligent Sampling randomly selecting a set of nodes from $\vec{\mathbf{h}t}$, when the characteristics from $\vec{ht}$ are required given the profile $\mathbf{P}$.

B. Intelligent Sampling procedure

The Intelligent Sampling procedure can be applied in our proposed infrastructure model at both the heterogeneity class level and heterogeneity type level. At the class level, the procedure identifies the adequacy of using additional heterogeneity classes for the incoming workload. At the type level, the procedure identifies which specific class instances are better suited to the workload. Simply put, the first would identify that the incoming workload should be in a node with a GPU, while the second would recommend the usage of a GPU T4 instead of V100. From a formal angle, the procedure is the same for both cases (class or type). The only difference is that in the first case a variable (Ht) identifies the heterogeneity class, while in the second case, a three-dimensional vector ($\vec{Ht}$) identifies the heterogeneity type. Next, we show the required steps for the heterogeneity class (Ht).

In the following, we assume that we do not know the specific requirements or optimization policies in place, e.g.,

whether the infrastructure provider aims at minimizing energy consumption. We show how to leverage the history of executions to unveil the expected heterogeneity classes and characteristics required for the incoming workload. Hence, we assume that the history of executions has not been obtained randomly but through a set of "optimal" decisions.

The Bayes theorem is a powerful tool for describing and evaluating hypotheses. It allows us to describe different scenarios (i.e. the performance of a workload given a specific heterogeneity characteristic) and to evaluate them given historical data. Hence, we can formulate the hypothesis of adequacy in terms of posterior probability: $P(Ht|\mathbf{P})$ which is the probability that a heterogeneity class (Ht) is beneficial for a workload with profile $\mathbf{P}$. Or in other words which is the convenient node for a given workload that requires to be scheduled, however, we are abstracting the node by means of its heterogeneity trait and the workload as a profile. The required steps are summarized in Algorithm 1.

Algorithm 1 Intelligent sampling procedure

1: Compute $P(\mathbf{P}|Ht)$ ▷ Likelihood of the profile
2: Compute $P(Ht)$ ▷ The prior of the infrastructure
3: Compute $P(\mathbf{P})$ ▷ The marginal probability of the profile
4: Compute $P(Ht|\mathbf{P})$ ▷ The posterior probability
5: Sample nodes according to $P(Ht|\mathbf{P})$

Hence we compute $P(Ht|\mathbf{P})$ by leveraging the Bayes theorem:

$$P(Ht|\mathbf{P}) = \frac{P(\mathbf{P}|Ht)P(Ht)}{P(\mathbf{P})} \tag{2}$$

When a new workload arrives with its profile specified the following procedure for each heterogeneity class in the infrastructure needs to be executed, i.e., $\forall\, H_t \in Infra$ **compute** :

1) $P(\mathbf{P}|Ht)$: the likelihood, expresses how frequently a profile $\mathbf{P}$ has been executed in a node with an heterogeneity class Ht. We leverage the history of executions and compute it as the ratio between the number of times that $\mathbf{P}$ has been executed on $\mathbf{H}t$ divided by the total times that $\mathbf{P}$ has been executed. Notice that we use $\mathbf{H}t$ to consider all nodes that have this specific heterogeneity class:
$$P(\mathbf{P}|Ht) = \frac{C_{P,Ht}}{C_{P,-}} \tag{3}$$
   a) $C_{P,Ht}$ corresponds to the counter C of execution of any workload of the profile $\mathbf{P}$ in any node containing the heterogeneity trait Ht.
   b) $C_{P,-}$ corresponds to the counter C of execution of any workload of the profile $\mathbf{P}$ in any node of the infrastructure.
2) $P(Ht)$: the prior, describes how common is the heterogeneity trait Ht among all infrastructure nodes:
$$P(Ht) = \frac{|\mathbf{H}t|}{|Infra|} \tag{4}$$
   a) $|\mathbf{H}t|$ is the cardinality of the set of nodes with the heterogeneity trait.
   b) $|Infra|$ is the number of nodes in the infrastructure.

   Interestingly, in the case of having a scheduler with specific preferences over a type of node, that could be expressed in terms of the prior, i.e., the prior probability could be tweaked to incorporate infrastructure usage policies.
3) $P(\mathbf{P})$: is the marginal probability of the profile $P(\mathbf{P})$, and expresses in which conditions the profile has been executed in the infrastructure regardless of the heterogeneity class, hence to obtain this "regardless" we marginalize over Ht as seen in the formula:
$$P(\mathbf{P}) = \sum_{\forall\, Ht \in Infra} P(\mathbf{P}|Ht)P(Ht) \tag{5}$$
   To compute this last term it is just required to compute steps (1) and (2) for each heterogeneity class in the infrastructure and sum them.

Assuming that the number of heterogeneity classes is much lower than the number of nodes ($k >> l$) in a large-scale distributed and heterogeneous infrastructure, the required terms to compute $P(Ht|\mathbf{P})$ can be easily pre-computed, updating their values after each successful execution.

4) $P(Ht|\mathbf{P})$ is computed following Equation 2 providing a posterior probability of the usage of a specific heterogeneity class for the incoming profile.
5) Finally, we could sample over each of the $\mathbf{H}t_i$ a number of times proportional to the result of $P(Ht|\mathbf{P})$. However, there are several specific policies that can be applied at this stage depending on the overall infrastructure management strategy.

V. Case study & Performance evaluation

A. Case study

To clarify the model, we will apply it to the Alibaba Platform of Artificial Intelligent (PAI) [13], we can find 2 types of nodes: those containing a GPU and those without a GPU (named CPU). Then, among the nodes with a GPU, there is the following classification, according to the GPU type: P100, T4, V100, V100M32, and MISC. The MISC category groups old GPU versions such as the K80. Hence, there is a first layer of the model that specifies if the node has GPU or not. Then, if the node has a GPU we can go to the second layer of the model. GPUs are classified according to their power consumption, computational capacity, and cost[7]. Figure 5 shows their values for each aspect, obtained from [21]–[24]. Notice that for visualization purposes V100* corresponds to V100M32, also consider that the specific cost for V100M32 has not been found and it has been assigned the same as the V100. Considering these values, we can use the

[7]The cost is obtained from "GPU pricing — Compute Engine: Virtual Machines (VMs)," Google Cloud. https://cloud.google.com/compute/gpus-pricing (accessed Feb. 14, 2023). The price for the V100M32 is assumed to be the same as for the V100

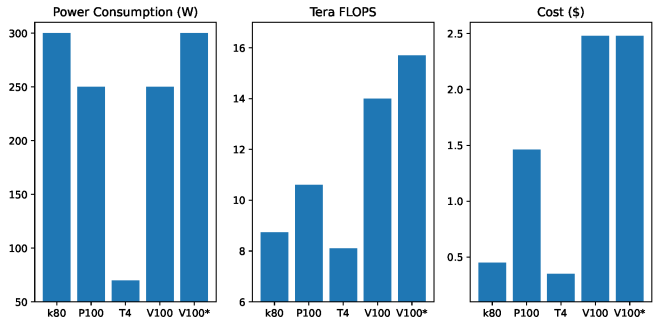


Fig. 5: Characteristics of each heterogeneity trait (GPU type) with respect to each heterogeneity aspect (Energy efficiency, Computation efficiency, and Cost efficiency).

mid-range value (M), equation 1, to split each characteristic as "High" or "Low". Table I follows the example of the GPU type, but we can foresee these tables per infrastructure class.

TABLE I: GPU types modeled with respect to each heterogeneity characteristic.

	k80	**P100**	**T4**	**V100**	**V100M32**
Energy-efficiency	Low	Low	High	Low	Low
Computation-efficiency	Low	Low	Low	High	High
Cost-efficiency	High	Low	High	Low	Low
Date	Nov 2014	Jun 2016	Sep 2018	Jun 2017	May 2017

At this point, given a request for a GPU with a set of specific characteristics Table I can identify the type of GPUs required and then expose them to the sampler. Further, Table I shows that GPU models V100 and V100M32 are equivalent heterogeneity types. Hence they are always exposed to the sampler under the same conditions. It can be anticipated that all combinations might not be present in the infrastructure. Similarly, a greedy policy looking only for a "High" value in all fields is not realistic, given that these characteristics generally require a certain level of compromise among them. We envision the possibility of requesting a specific characteristic (or two of them), leaving the others unspecified. In such a case, it could be specified a GPU with "High" Cost-efficiency, while ignoring the other two characteristics. Then, following the previous example, we would have GPU types *K80* and *T4* behaving like equivalent heterogeneity types. This type of aggregation can ease the resource management modules of the infrastructure. Finally, while the information on the convenient heterogeneity class for the workload comes from the profile description, the suggested characteristics defining the heterogeneity type are envisioned to be set by the infrastructure provider to optimize its usage. Nevertheless, with the improvement and fine-grained tuning of the profiles, heterogeneity types options could also be defined at the profile level. In such a situation, the logic on the infrastructure provider side is expected to make the decisions, e.g., a sample only from the types requested through the profile, a sample from all aggregated options, etc.

From the model use case, we make several observations. First, we can see that heterogeneity is not only at design (i.e., variety on Edge computing devices), but it is part of the computing infrastructure evolution, we see in Table I 5 types of GPU in only 4 years of difference. Second, we realize that the differences in the 3 modeled characteristics are significant. In addition, the 3 characteristics used are not correlated, meaning that the usage of equivalent classes not using the 3 characteristics can provide different and adjustable results. Similarly, we see that there is no time-wise correlation, i.e., the newest GPU is not the best in all characteristics. Last, we see that using the mid-range value performs well to separate GPU types, however, the *P100* is close to the mid-range value in 2 out of 3 characteristics. Hence, with the increase of heterogeneity types and the will to use the model to perform infrastructure usage optimizations a more fine-grained metric can be used to classify the characteristics.

B. Performance evaluation

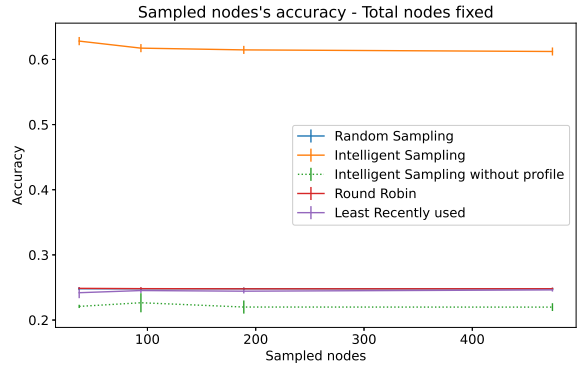
The evaluation of Intelligent Sampling is in comparison to other state-of-the-art sampling approaches. We aim to show that Intelligent Sampling is a better option in terms of sampling accuracy and execution time. We choose 3 of the most well-known sampling approaches: Random sampling (RS), Round-robin sampling (RR), and Least recently used (LR). In brief, RS will randomly sample infrastructure nodes, RR will shuffle the infrastructure nodes and then sample them in order, and finally, LR will favor sampling the less used nodes in the infrastructure. We define the sampling accuracy (Acc_s) in a multi-class classification problem as follows:

$$Acc_s = \frac{|\text{Correctly selected nodes}|}{|\text{All selected nodes}|} \tag{6}$$

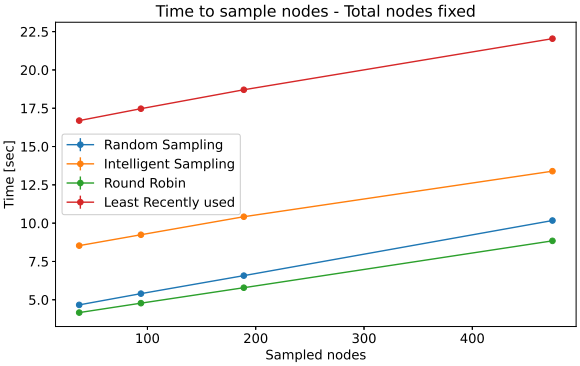
Consider that the absence of true negatives and using an averaged value (micro) for the F_{score} makes it equivalent to accuracy, the metric computation has leveraged Scikit-learn Python package [25].

The analysis is based on the Alibaba PAI dataset [13]. Hence, a node is correctly selected if it matches the one selected from the Alibaba PAI dataset. We use the description of their infrastructure and the data of the executed workloads. Hence, we will sample nodes for 5 000 randomized incoming workloads. For each incoming workload, all infrastructure nodes will be available. We are evaluating the sampling capacity, allowing us to marginalize the available resources of the infrastructure. The total number of nodes sampled will depend on the experiment. Also, the accuracy provided for each experiment is averaged over all incoming workloads and the 10 repetitions performed. Time is averaged over the 10 repetitions but aggregated for the incoming workloads. Hence, if we want an idea of the sampling time required for 1 workload, we should divide the obtained values by 5 000.

The following assumptions are taken into account for the evaluation. Only successfully completed workloads are considered; hence, the scheduling decision is considered the correct one. The nodes without GPU are regarded as if they had a different heterogeneity type of GPU. This is done because the number of nodes without GPU is minimal, and for research

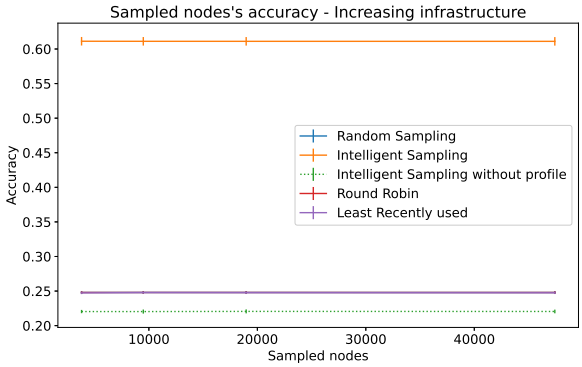


(a) Sampling accuracy with respect to the number of nodes sampled, averaged over 5 000 incoming workloads.

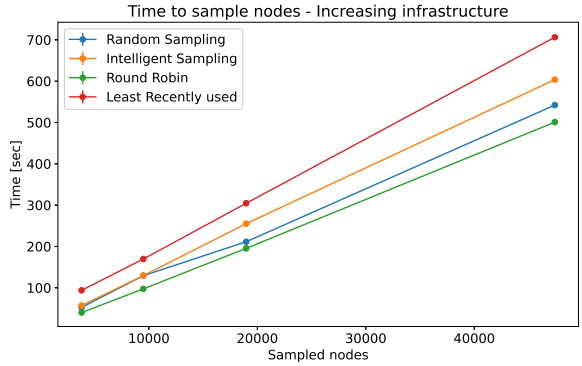


(b) Aggregated time required to sample nodes for 5 000 incoming workloads.

Fig. 6: Evaluation of the Intelligent Sampling against RS, RR, and LR algorithms, when the total number of nodes for sampling is circa 2 000. The experiment is averaged over ten repetitions.



(a) Sampling accuracy with respect to the number of nodes sampled, averaged over 5 000 incoming workloads.



(b) Aggregated time required to sample nodes for 5 000 incoming workloads.

Fig. 7: Evaluation of the Intelligent Sampling against RS, RR, and LR algorithms, when the total number of nodes for sampling is circa 200 000. The experiment is averaged over 10 repetitions.

purposes looked more interesting having a wider variety of heterogeneity types rather than discarding 5% of the nodes at an early stage. The workload profiling results are inherited from another work (under acceptance of publication), and we use the results obtained there. It is important to remark that the profiling mechanism leaves a quarter of the workloads without an associated profile as outliers. Hence, when a workload comes with an outlier label, Intelligent Sampling applies random sampling to get the sampled nodes.

The first experiment, see Figure 6, evaluates the capacity of the 4 strategies to get increasingly larger samples of nodes (precisely 2%, 5%, 10%, and 25%) over a fixed infrastructure setting (i.e., the one defined in [13]). In Figure 6a, we see the accuracy of the methods. Notice that Intelligent Sampling without profile is applying random sampling. Interestingly its behavior is below the random sampling approach identifying that such workloads are not similar to the others. In any case, we can see an outperform of Intelligent Sampling, achieving more than 60% accuracy on the sample, while the other methods are around 25%. Notice that in Figure 6a the vertical markers are showing the standard deviation of the accuracy. Figure 6b shows the time required to sample nodes for all incoming workloads averaged over the ten repetitions, we see in all cases a linear increase in time for the number of nodes sampling, and we see Intelligent Sampling being slower than RS or RR, but much faster than LR. In this case, the standard deviation is not visible and the dots are just marking the evaluated points.

The second experiment, see Figure 7, is similar to the previous one. The aim is to sample an increasing number of nodes, as previously 2%, 5%, 10%, and 25%. But the infrastructure is expanded in a simulated manner to circa 200 000 nodes, keeping the nodes' proportion from the original infrastructure. From Figure 7a, we can see that the accuracy obtained is the same as in the previous experiment, again in this Figure the vertical markers indicate the standard deviation. Also, the measured time is linear and keeps the same order in the long run; see Figure 7b.

Finally, in the third experiment, we keep constant the number of nodes sampled (100) but increase the infrastruc-

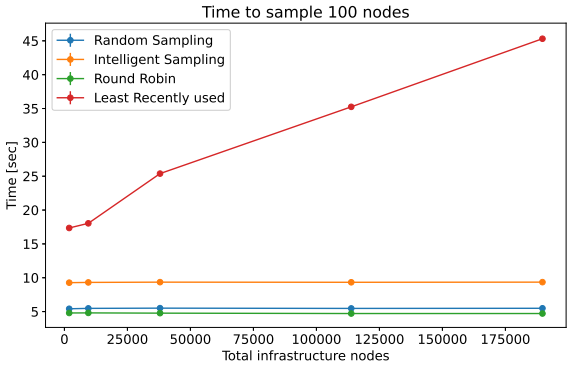


Fig. 8: Evaluation of the scalability when the number of nodes increases, but the sampling size is fixed (100 nodes).

ture's size. The infrastructure expansion is simulated, keeping constant the ratio of node types. This experiment aims at discovering if the scale of the infrastructure plays a role in the performance of the sampling method when the number of nodes to sample is fixed. We find that in terms of accuracy, it has the same behavior as in the two previous experiments. However, we can see in Figure 8 that the sampling time is constant for Intelligent Sampling as well as RS and RR, while linear for LR. This behavior was expected given that Intelligent Sampling, RS, and RR only operation on the data is retrieving a fixed number of data points, while LR has to read and write on the data, which penalizes the total time.

VI. Related work

A. Sampling methods for scheduling in large-scale, heterogeneous, and distributed computing infrastructures

There is a plethora of articles and work in scheduling. If interested, the reader can refer to [26], [27] for Edge-specific scenarios. It is important to remark that both surveys detect scalability and infrastructure heterogeneity as key research challenges for the community, which are the topics we aim to address in this work. An interesting review focusing on scheduling for IoT and micro-services applications is done by Pallewatta et al. [28]. There, we can also find research gaps in terms of infrastructure heterogeneity and system scalability. Finally, for a Cloud K8s-specific survey on scheduling, one can refer to the work of Carrion et al. [29], where scalability is also seen as a required future work. However, due to the homogeneity assumption in Cloud infrastructure, device heterogeneity is not deeply addressed.

As briefly mentioned in the introduction, sampling, and auctioning are the two main approaches to scheduling in large-scale, heterogeneous, and distributed infrastructures. Sampling consists of randomly selecting a set of candidate nodes (resources) for the workload and assessing their suitability instead of doing it with the entire infrastructure [10]. Auctioning moves the responsibility to the nodes (resources), who are in charge of bidding for the workload [12], and the highest bid takes it. Inconveniently, this methodology falls short when dealing with common scheduling requirements such as affinity/anti-affinity placement of workloads due to outsourcing the decision to individual cluster nodes.

In general, random sampling has been very successful in large-scale and homogeneous infrastructures, as shown in [10]. More recent works include other sampling solutions for heterogeneous infrastructures [30]. However, their statistical guarantees on sample quality make their samples large, hampering system scalability.

As a side note for sampling methods, the advent of federated learning in the Edge provides new sampling strategies to select nodes [31]. However, these methods are specifically tailored to the FL algorithm's needs, not suitable for a more generic problem. Nevertheless, synergies between the two research fields can eventually arise.

B. Heterogeneous infrastructure

Most of the literature on heterogeneous infrastructure focuses on dynamic characteristics, e.g., CPU, memory, network [18], [32]. Further, to the best of our knowledge, there is limited work proposing models, classifications, or taxonomies for heterogeneous computing infrastructure. However, working toward computing continuum systems forces to take into account heterogeneity and models or taxonomies will be required to include and eventually manage the whole spectrum of computing devices, from IoT hardware [33] to FPGAs [34], GPUs [35], or TPUs [36].

Many survey papers in the literature provide taxonomies for resource management at the Edge [27], [37]–[39], but, as explained in [38], the works tackling heterogeneous Edge computing resources are at its infancy.

In general, the modeling of computing resources aims at understanding their behavior towards predictability, as done in [40], but this fine-grained modeling is not well-suited when managing large-scale systems. Interestingly, the work of Shukla et al. [20] deals with heterogeneous CPU core types and provides a model based on the energy efficiency and service capacity used as a basis to build our generic approach to model heterogeneous infrastructure.

VII. Conclusion

Intelligent Sampling enables precise and scalable sampling over large-scale and heterogeneous computing infrastructures, a cornerstone for ML clusters, Edge Computing, and the computing continuum. The model and the method presented in this article can leverage computing infrastructure heterogeneity while keeping the complexity hidden from the scheduler.

The experiments show that the overall results are satisfactory. We have demonstrated that Intelligent Sampling can achieve a sampling accuracy of more than 60% regardless of the sample size, while its competitors barely reach 25%. This means that in a sample from Intelligent Sampling we would have 3 out of 5 nodes containing the specific heterogeneity class, while when using other state-of-the-art methods we would have only 1 out of 4. Time-wise, Intelligent Sampling is slightly slower than RS and RR; however, it behaves similarly,

i.e., the sampling time is constant when the sample size is maintained but the infrastructure increases (see Figure 8).

In future work, we will define the precise implementation of Intelligent Sampling on a real computing infrastructure. We also aim to research different sampling policies. Simply put, we are sampling proportional to the posterior probability, but, as an example, we could take the type of node with the maximum posterior probability. Further, we can study the performance of these policies considering the quality and quantity of data gathered from the cluster usage and the incoming workloads, framing the problem in the classical exploitation/exploration trade-off. We will develop system-wide tests to evaluate the benefits of the infrastructure model.

In a broader scope, we will keep looking at the sources of heterogeneity in Cloud-Edge computing systems to identify their characteristics and provide techniques and models capable to leverage them.

ACKNOWLEDGMENTS

This work is supported by Futurewei's Cloud Lab. as part of the overall open source initiative.

REFERENCES

[1] P. Beckman, J. Dongarra, N. Ferrier, G. Fox, T. Moore, D. Reed, and M. Beck, "Harnessing the computing continuum for programming our world," in *Fog Computing* (A. Zomaya, A. Abbas, and S. Khan, eds.), pp. 215–230, John Wiley & Sons, Ltd, Apr. 2020.

[2] S. Dustdar, V. C. Pujol, and P. K. Donta, "On Distributed Computing Continuum Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, pp. 4092–4105, Apr. 2023.

[3] V. Casamayor Pujol, A. Morichetta, I. Murturi, P. Kumar Donta, and S. Dustdar, "Fundamental Research Challenges for Distributed Computing Continuum Systems," *Information*, vol. 14, Mar. 2023.

[4] J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson, "What serverless computing is and should become," *Communications of the ACM*, vol. 64, pp. 76–84, May 2021.

[5] S. Nastic, P. Raith, A. Furutanpey, T. Pusztai, and S. Dustdar, "A Serverless Computing Fabric for Edge & Cloud," in *2022 IEEE 4th International Conference on Cognitive Machine Intelligence (CogMI)*, pp. 1–12, Dec. 2022.

[6] Z. Zhao, G. Verma, C. Rao, A. Swami, and S. Segarra, "Distributed scheduling using graph neural networks," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2021-June, pp. 4720–4724, 2021.

[7] Q. Wang, X. Mei, H. Liu, Y.-W. Leung, Z. Li, and X. Chu, "Energy-Aware Non-Preemptive Task Scheduling With Deadline Constraint in DVFS-Enabled Heterogeneous Clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, pp. 4083–4099, Dec. 2022.

[8] S. K. Saurav and S. Benedict, "A Taxonomy and Survey on Energy-Aware Scientific Workflows Scheduling in Large-Scale Heterogeneous Architecture," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, pp. 820–826, Jan. 2021.

[9] S. Nastic, T. Pusztai, A. Morichetta, V. C. Pujol, S. Dustdar, D. Vij, and Y. Xiong, "Polaris scheduler: Edge sensitive and slo aware workload scheduling in cloud-edge-iot clusters," in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pp. 206–216, IEEE, 2021.

[10] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: distributed, low latency scheduling," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, (New York, NY, USA), pp. 69–84, Association for Computing Machinery, Nov. 2013.

[11] A. Attanasio, G. Ghiani, L. Grandinetti, and F. Guerriero, "Auction algorithms for decentralized parallel machine scheduling," *Parallel Computing*, vol. 32, pp. 701–709, Oct. 2006.

[12] D. Bermbach, J. Bader, J. Hasenburg, T. Pfandzelter, and L. Thamsen, "AuctionWhisk: Using an auction-inspired approach for function placement in serverless fog platforms," *Software: Practice and Experience*, vol. 52, no. 5, pp. 1143–1169, 2022.

[13] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, (Renton, WA), pp. 945–960, USENIX Association, Apr. 2022.

[14] S. Nastic, A. Morichetta, T. Pusztai, S. Dustdar, X. Ding, D. Vij, and Y. Xiong, "Sloc: Service level objectives for next generation cloud computing," *IEEE Internet Computing*, vol. 24, no. 3, pp. 39–50, 2020.

[15] T. Pusztai, S. Nastic, A. Morichetta, V. Casamayor Pujol, S. Dustdar, X. Ding, D. Vij, and Y. Xiong, "A novel middleware for efficiently implementing complex cloud-native slos," in *IEEE 14th International Conference on Cloud Computing (CLOUD)*, 2021.

[16] T. Pusztai, S. Nastic, A. Morichetta, V. Casamayor Pujol, S. Dustdar, X. Ding, D. Vij, and Y. Xiong, "Slo script: A novel language for implementing complex cloud-native elasticity-driven slos," in *IEEE International Conference on Web Services (ICWS)*, 2021.

[17] A. Morichetta, V. Casamayor Pujol, S. Nastic, S. Dustdar, D. Vij, Y. Xiong, and Z. Zhang, "PolarisProfiler: A novel metadata-based profiling approach for optimizing resource management in the edge-cloud continnum," in *2023 18th Annual System of Systems Engineering Conference (SOSE)*, 2023. Accepted - To be published.

[18] A. K. Kulkarni and B. Annappa, "Context Aware VM Placement Optimization Technique for Heterogeneous IaaS Cloud," *IEEE Access*, vol. 7, pp. 89702–89713, 2019.

[19] Z. Zhong and R. Buyya, "A Cost-Efficient Container Orchestration Strategy in Kubernetes-Based Cloud Computing Infrastructures with Heterogeneous Resources," *ACM Trans. Internet Technol.*, vol. 20, pp. 15:1–15:24, Apr. 2020.

[20] S. K. Shukla, D. Ghosal, and M. K. Farrens, "Understanding and Leveraging Cluster Heterogeneity for Efficient Execution of Cloud Services," in *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*, pp. 56–64, Nov. 2021.

[21] "NVIDIA Tesla P100: der fortschrittlichste Grafikprozessor für Rechenzentren." https://www.nvidia.com/de-de/data-center/tesla-p100/ (accessed 2023-02-15).

[22] "NVIDIA Tesla V100." https://www.nvidia.com/en-gb/data-center/tesla-v100/ (accessed 2023-02-15).

[23] "NVIDIA T4 Tensor Core GPUs for Accelerating Inference." https://www.nvidia.com/en-us/data-center/tesla-t4/ (accessed 2023-02-15).

[24] "Nvidia tesla K80." https://www.nvidia.com/en-gb/data-center/tesla-k80/ (accessed 2023-02-14).

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource Scheduling in Edge Computing: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.

[27] M. Raeisi-Varzaneh, O. Dakkak, A. Habbal, and B.-S. Kim, "Resource Scheduling in Edge Computing: Architecture, Taxonomy, Open Issues and Future Research Directions," *IEEE Access*, vol. 11, pp. 25329–25350, 2023.

[28] S. Pallewatta, V. Kostakos, and R. Buyya, "Microservices-based IoT Applications Scheduling in Edge and Fog Computing: A Taxonomy and Future Directions," July 2022. arXiv:2207.05399 [cs].

[29] C. Carrión, "Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges," *ACM Comput. Surv.*, vol. 55, pp. 138:1–138:37, Dec. 2022.

[30] C. Delimitrou, D. Sanchez, and C. Kozyrakis, "Tarcil: Reconciling scheduling speed and quality in large shared clusters," *ACM SoCC 2015 - Proceedings of the 6th ACM Symposium on Cloud Computing*, pp. 97–110, Aug. 2015.

[31] H. Wu and P. Wang, "Probabilistic Node Selection for Federated Learning with Heterogeneous Data in Mobile Edge," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 2453–2458, Apr. 2022. ISSN: 1558-2612.

[32] Y. Hu, H. Zhou, C. de Laat, and Z. Zhao, "ECSched: Efficient Container Scheduling on Heterogeneous Clusters," in *Euro-Par 2018: Parallel Processing* (M. Aldinucci, L. Padovani, and M. Torquati, eds.), Lecture Notes in Computer Science, (Cham), pp. 365–377, Springer International Publishing, 2018.

[33] R. Morabito, I. Farris, A. Iera, and T. Taleb, "Evaluating Performance of Containerized IoT Services for Clustered Devices at the Network Edge," *IEEE Internet of Things Journal*, vol. 4, pp. 1019–1030, Aug. 2017.

[34] A. Putnam *et al.*, "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," *IEEE Micro*, vol. 35, pp. 10–22, May 2015.

[35] M. Guzek, D. Kliazovich, and P. Bouvry, "HEROS: Energy-Efficient Load Balancing for Heterogeneous Data Centers," in *2015 IEEE 8th International Conference on Cloud Computing*, pp. 742–749, June 2015. ISSN: 2159-6190.

[36] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, June 2017.

[37] K. Toczé and S. Nadjm-Tehrani, "A Taxonomy for Management and Optimization of Multiple Resources in Edge Computing," *Wireless Communications and Mobile Computing*, vol. 2018, p. e7476201, June 2018. Publisher: Hindawi.

[38] P. Kansal, M. Kumar, and O. P. Verma, "Classification of resource management approaches in fog/edge paradigm and future research prospects: a systematic review," *J Supercomput*, vol. 78, pp. 13145–13204, July 2022.

[39] Z. Sharif, L. T. Jung, M. Ayaz, M. Yahya, and S. Pitafi, "A Taxonomy for Resource Management in Edge Computing, Applications and Future Realms," in *2022 International Conference on Digital Transformation and Intelligence (ICDI)*, pp. 46–52, Dec. 2022.

[40] S. Lahmer, A. Khoshsirat, M. Rossi, and A. Zanella, "Energy Consumption of Neural Networks on NVIDIA Edge Boards: an Empirical Model," in *2022 20th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pp. 365–371, Sept. 2022.