2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing (UCC) | 979-8-3503-6720-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/UCC63386.2024.00013

HEET: A Performance Measure to Quantify Heterogeneity in Distributed Computing Systems

Ali Mokhtari
University of Louisiana at Lafayette
LA, USA
ali.mokhtari@louisiana.edu

Saeid Ghafouri
Queen Mary University
London, UK
s.ghafouri@qmul.ac.uk

Pooyan Jamshidi
University of South Carolina
SC, USA
pjamshid@cse.sc.edu

Mohsen Amini Salehi
University of North Texas
TX, USA
mohsen.aminisalehi@unt.edu

***Abstract*—Despite extensive research on system heterogeneity, there remains a gap in measuring its impact on system performance. Previous studies have primarily focused on the binary definition of heterogeneity based on architectural diversity, without exploring dimensions of system heterogeneity and their impacts on the system performance. To bridge this gap, in this study, we propose a heterogeneity measure that, for a given heterogeneous system, offers a representation of its performance. This heterogeneity measure is instrumental for solution architects in proactively defining their systems to be sufficiently heterogeneous to meet their desired performance objectives. Accordingly, we develop a mathematical model to characterize a heterogeneous system in terms of its task and machine heterogeneity dimensions and then condense it to a single value, called Homogeneous Equivalent Execution Time (HEET), which represents the execution time behavior of the entire system. We used AWS EC2 instances to implement a machine learning inference system. Using HEET scores in various heterogeneous system configurations demonstrated that HEET can accurately characterize the performance behavior of these systems. In particular, the results show that HEET can help predict the true makespan of heterogeneous systems with an average precision of 84%.**

***Index Terms*—Heterogeneous Computing, Performance Analysis, Distributed Computing Systems**

I. Introduction

A. Research Motivations and Goals

Heterogeneity has been an indispensable aspect of distributed computing throughout the history of these systems. In the modern era, as Moore's law is losing momentum due to power density and heat dissipation limitations [1], heterogeneous computing systems have attracted even more attention to overcome the slowdown in Moore's law and fulfill the desire for higher performance in various types of distributed systems. In particular, with the ubiquity of accelerators (e.g., GPUs and TPUs) and domain-specific computing (through ASICs and FPGAs), the matter of heterogeneity and harnessing it has become a more critical challenge than ever before to deal with.

In the literature, heterogeneity in computing systems is mainly identified based on the architectural diversity of their machines. If all machines share the same architecture, the computing system is homogeneous; if there is architectural diversity, the system is considered heterogeneous. However, the 0/1 heterogeneity metric may not accurately capture the intricacies of diverse and multifaceted systems. For example, cloud providers like AWS offer varying instance types, including GPU-based and CPU-based EC2 instances, each with different associated costs. These systems can be utilized to serve machine learning inference requests. Although multiple viable configurations meet the desired throughput, there is an optimal configuration that minimizes the incurred cost. To minimize the cost incurred by different heterogeneous computing systems, we need to be able to compare heterogeneous systems by quantifying the impact of heterogeneity on performance metric (e.g., throughput). This emphasizes the need for a more nuanced metric to inform decision-making and navigate the vast decision space.

As such, *our goal in this study is to propose a "performance-driven heterogeneity score" that can characterize the impact of the heterogeneity level of a system on its performance objective (i.e., throughput) and make the system comparable with its counterparts*. In essence, our goal is to go beyond a binary heterogeneity metric and embrace a more nuanced approach, *the heterogeneity spectrum*. This spectrum attributes a heterogeneity score to each heterogeneous system, offering a representation of its performance relative to others. In this way, the position of computing systems in the heterogeneity spectrum serves as a guiding metric and navigates the expansive decision space, ultimately aiding in making well-informed decisions.

B. Problem Statement and Summary of Contributions

We define a heterogeneous computing system as a set of architecturally diverse machines that work together to complete a set of requests (a.k.a. *tasks*) with different computational requirements. We categorize the tasks that arrive at a system based on the type of operation they perform and call them *task types*. For example, in a system that helps visually impaired people [2], [3], task types can be obstacle detection, face recognition, and speech recognition. Moreover, we classify machines of a computing system based on their architectural and performance characteristics and call each one a *machine type*. In this work, we consider the heterogeneity of the system resulting from the diversity in the types of machines and the computational requirements of the types of tasks. That is, system heterogeneity has two dimensions: (i) *machine heterogeneity* and (ii) *task heterogeneity*.

979-8-3503-6720-1/24/$31.00 ©2024 IEEE
DOI 10.1109/UCC63386.2024.00013

Profiling various task types on heterogeneous machine types can describe their execution time behavior. Variations in the average execution time of a given task type across all machine types demonstrate a dimension of the system heterogeneity and are defined as machine heterogeneity, whereas variations in the execution time of different task types on a given machine type highlight the other dimension of system heterogeneity and are defined as task heterogeneity. The *system heterogeneity* is defined as the compound heterogeneity of these dimensions.

To construct the heterogeneity spectrum across heterogeneous computing systems in a manner that can describe the performance behavior of systems in terms of makespan or throughput, we need to address the following research questions: (i) *How to find a heterogeneity score to make heterogeneous systems comparable?* (ii) *How to exploit the heterogeneity score to predict the performance of a computing system?*

We define *Expected Execution Time (EET)* as a matrix to store the expected execution time of each type of task on each machine. In this way, the entry $EET[i, j]$ (denoted e_{ij}) represents the expected execution time of the task type i in machine type j. For systems with multiple instances of the same machine type, the corresponding columns of those machines in the EET matrix are identical. The EET matrix, as a whole, represents the expected performance of the entire system in terms of the execution times of the tasks. As such, the matrix can be used as a guide to understand the throughput that the system can achieve.

In pursuit of developing a heterogeneity score, we present a method for transforming a heterogeneous computing system into a hypothetical equivalent homogeneous system. This homogenization ensures that the equivalent homogeneous system exhibits performance behavior similar to the original heterogeneous system for evaluation and comparison. We first establish a baseline homogeneous system, which is generally slower than the heterogeneous system. Then, we calculate the speedup factor, which quantifies how much faster the heterogeneous system is compared to the baseline. Finally, utilizing the speedup factor relative to the baseline, we determine the equivalent homogeneous system. This allows us to capture the essence of heterogeneity while simplifying the system for analysis and comparison. For that purpose, we perform homogenization in two dimensions of heterogeneity: (i) machine homogenization, where all diverse machines are transformed into hypothetical homogeneous machines, and (ii) task homogenization, where task types are transformed into a hypothetical equivalent task. We examine various measures of central tendency, namely arithmetic, geometric and harmonic means, and propose a single measure, called Homogeneous Equivalent Execution Time (*HEET*), which, for a given EET matrix, describes the expected execution time of a hypothetical homogeneous system whose performance is similar to the heterogeneous system represented by the EET matrix. Subsequently, we employ the HEET measure to determine the throughput of the heterogeneous computing system. We evaluated HEET score for various heterogeneous systems and showed that it can accurately describe the impact of heterogeneity on the desired throughput. In summary, the specific contributions of this paper are as follows:

- Providing a measure to quantify the system heterogeneity such that it can be used to determine the throughput of a system represented by the EET matrix.
- Proposing a systematic approach to analyzing heterogeneity of a computing system via decoupling the heterogeneity into machine and task heterogeneity dimensions and homogenizing each dimension separately.
- Proving the appropriateness of the arithmetic mean and harmonic mean to measure the central tendency of speedup vectors due to machine heterogeneity and task heterogeneity, respectively.
- Validating how system performance (makespan or throughput) can be derived as a function of the proposed heterogeneity score.

II. Heterogeneity to Homogeneity Transformation

A. Overview

System heterogeneity is the result of the synergy between machine heterogeneity and task heterogeneity dimensions. Accordingly, our approach is to individually homogenize each dimension of heterogeneity and then combine them to transform the heterogeneous system into an equivalent homogeneous one. To this end, we base our analysis on the notion of the EET matrix that is representative of the system performance, where the row-wise variations illustrate the machine heterogeneity, and the column-wise variations express the task heterogeneity. Note that if some machines in the system are the same, their corresponding columns in the EET matrix are repeated.

In the homogenization process, we establish a baseline homogeneous system, which is slower than the heterogeneous system. Then, we exploit the notion of speedup to quantify how much faster the heterogeneous system is compared to the baseline homogeneous one. In machine homogenization (MH), given a task type, the baseline is the slowest machine for that task type. Thus, the speedup vector for the task type i, denoted by $\overrightarrow{\alpha}^{(i)}$, is determined based on Equation 1.

$$\alpha_j^{(i)} = \frac{\max_{j=1}^{n} e_{ij}}{e_{ij}} \tag{1}$$

In the above equation, $\overrightarrow{\alpha}^{(i)}$ has the same dimension as the i$^{\text{th}}$ row of the EET matrix. Each entry $\alpha_j^{(i)}$ denotes the speedup that the system can achieve when the machine type j executes the task type i instead of running it on the slowest machine type.

Similarly, in task homogenization (TH) for a certain machine, we base our analysis on the slowest task type for that machine. Then, the speed-up vector for machine j, denoted $\overrightarrow{\beta}_{(j)}$, is calculated based on Equation 2.

$$\beta_{(j)}^{i} = \frac{\max_{i=1}^{m} e_{ij}}{e_{ij}} \tag{2}$$

The entry $\beta^{i}_{(j)}$ indicates the speedup achieved by executing a task of type i on the machine type j, as opposed to executing the slowest task type on that machine.

Given the speed-up vectors due to machine (task) heterogeneity, our objective is to determine the overall speed-up factor, a scalar value that quantifies the collective speed-up gain due to the heterogeneity of machines (tasks). This overall speed-up factor consolidates the effects of heterogeneity across all machines (tasks), so that if all machines (tasks) were replaced by a hypothetical machine (task) with an execution time represented by this factor relative to the slowest machine (task), the overall performance of the system would remain unchanged.

Based on the mean field method [4], the interaction of variables in a complex stochastic system can be replaced by the average interactions between these variables. As such, we can use the mean to represent the speedup behavior of all $(task, machine)$ pairs in both $\overrightarrow{\alpha}^{(i)}$ and $\overrightarrow{\beta}_{(j)}$. For this purpose, we employ statistical measures of central tendency (arithmetic, geometric, and harmonic mean) to accurately represent $\overrightarrow{\alpha}^{(i)}$ and $\overrightarrow{\beta}_{(j)}$. However, the challenge is that there is no consensus on the appropriateness of these measures to capture the central tendency of a specific use case [5], [6] and it must be investigated case by case. An appropriate mean speedup value derived from the EET matrix must precisely depict the "real speedup", a.k.a. *true speedup* (denoted by Γ), that the heterogeneous system can achieve for a given workload. We exploit the notion of makespan (i.e., the total time to execute the workload) to calculate the true speedup. According to Equation 3, the true speedup is determined based on the time to execute the workload in the heterogeneous system with respect to its homogeneous "counterpart homogeneous system", as the base system (a.k.a. baseline).

$$\Gamma = \frac{\text{homogeneous system makespan}}{\text{heterogeneous system makespan}} \tag{3}$$

Note that the homogeneous counterpart system is represented by an EET matrix whose entries are all equal to the maximum value of the EET matrix of the heterogeneous system. We use Γ_M and Γ_T to represent the true speed-up with respect to machine heterogeneity and task heterogeneity, respectively.

Given the true speedup value, we can compare it with the calculated overall speedup to determine its accuracy. We provide lemmas to introduce appropriate central tendency measures for speed-up due to machine and task heterogeneity. Taking into account that we consider the execution time of the slowest machine (task type) as the baseline, in the machine (task) homogenization process, given a task type (machine), we utilize the overall speedup factor and the baseline execution time to determine the expected execution time of the hypothetical equivalent machine (task type).

In the rest of this section, we elaborate on characterizing machine heterogeneity and task heterogeneity dimensions and explain how to homogenize each dimension. Then, we discuss how to fuse these dimensions to perform homogenization.

B. Machine Homogenization

In machine homogenization, given a task type, we use the expected execution times to aggregate the performance behavior of all machines. This allows us to introduce a hypothetical equivalent machine that can replace the heterogeneous machines without affecting the overall performance of the system. Specifically, for the task type i, we process the row i^{th} of the EET matrix with respect to the slowest machine for that task type to form a *row speedup vector*, denoted $\overrightarrow{\alpha}^{(i)}$. Then we use the central tendency measure (mean) of the components of the row speedup vector, denoted $\bar{\alpha}^{(i)}$, to aggregate the speedup behavior of all machines.

According to [7], in the circumstances where performance is expressed as a rate (e.g., flops), generally the harmonic mean can accurately express the central tendency. In addition, the central tendency can usually be represented by the arithmetic mean when performance is of a time nature (e.g., makespan or the total execution time of a benchmark). Lastly, they suggest avoiding the use of geometric mean when the performance is of the time or rate nature. In another study [6], the speedup is considered as the performance metric to compare an improved system with a baseline one. To do this, they used a set of benchmarks to evaluate the performance of each system. Based on the makespan of a benchmark, they calculated the speedup for the enhanced system. Next, the authors discussed different measures of central tendency (i.e., arithmetic, harmonic, and geometric mean) to summarize the speedup results of multiple benchmarks into a single number such that it appropriately describes the overall speedup for the entire benchmark suite. To validate the suitability of the mean speedup measure, they compared it with the speedup achieved by considering the makespan of the benchmark suite on both systems. In this research, we also follow the same approach as [6] to validate the accuracy of the central tendency measure of the speedup vector. In particular, we use the notion of true speed-up (Γ) to verify that the central tendency measure accurately represents the speed-up vector of the row. In addition, the intensity of task arrival to the system impacts machines' utilization, which, in turn, affects the true speedup. However, we are typically interested in studying system efficiency under high arrival rates.

In Lemma II-B, we study an extreme case where the task arrival rate is large enough so that all machines in the system have a task to perform at all times. We show that for such a system, the arithmetic mean can appropriately summarize $\overrightarrow{\alpha}^{(i)}$ and represent the mean speedup due to machine heterogeneity. For the other side of the spectrum, where the task arrival rate is low such that only one machine in the system is executing a task at a time, it can be shown that the harmonic mean should be used to represent the mean speedup due to machine heterogeneity. In these lemmas, we assume that there is a single unbounded FCFS queue of tasks that are all available for execution (like the bag-of-tasks [8]). Whenever a machine becomes free, it takes the next task from the queue to execute it.

Let $EET = [e_{ij}]$ $(1 \leqslant j \leqslant n)$ denote the EET vector of a heterogeneous computing system consisting of a set of machine types, $M = \{M_1, M_2, ..., M_n\}$, and a workload with $c > n$ tasks of type T_i that are all available for execution (such as the task bag). Tasks are queued upon arrival in a single unbounded FCFS queue. Whenever a machine becomes free, it takes a task from the queue and executes it. Then, the true speedup is calculated as follows:

$$\Gamma_M = \bar{\alpha}^{(i)(A)} = \frac{1}{n} \cdot \sum_{j=1}^{n} \alpha_j^{(i)} \tag{4}$$

We assume that machine type k is the slowest for task type T_i, that is, we have $\max_{j=1}^{n} e_{ij} = e_{ik}$. Then, the baseline system consists of n machines with an expected execution time of e_{ik}. For a single FCFS queue, c tasks are equally distributed between n homogeneous machines. Hence, each machine has to handle $\frac{c}{n}$ tasks, where the expected execution time of each task is e_{ik}. This means that the total time to complete those tasks c in the homogeneous system is $\lceil \frac{c}{n} \rceil \times e_{ik}$. We know $0 \leqslant \lceil \frac{c}{n} \rceil - \frac{c}{n} < 1$. If we replace $\lceil \frac{c}{n} \rceil$ with $\frac{c}{n}$, the error in calculating the total time is at most $(\lceil \frac{c}{n} \rceil - \frac{c}{n})/\lceil \frac{c}{n} \rceil$, which is negligible for a large number of tasks ($c \gg n$). Thus, for simplicity, we assume that the makepan to complete the c tasks in the homogeneous counterpart system is $\frac{c}{n} \times e_{ik}$. However, in a heterogeneous system, the proportions of the tasks handled by each type of machine are not equal because faster machines can execute more tasks.

Therefore, the speedup of the heterogeneous system is calculated as follows:

$$\Gamma_M = \frac{1}{n} \cdot \frac{c}{\left\lceil \frac{c}{\sum_{j=1}^{n} \alpha_j^{(i)}} \right\rceil} = \frac{1}{n} \cdot \frac{c}{\frac{c}{\sum_{j=1}^{n} \alpha_j^{(i)}}} = \frac{1}{n} \cdot \sum_{j=1}^{n} \alpha_j^{(i)} = \bar{\alpha}^{(i)(A)} \tag{5}$$

Where we assumed that $\left(c / \sum_{j=1}^{n} \alpha_j^{(i)}\right) \in \mathbb{Z}$, which may result in negligible error for a large number of tasks.

In Lemma II-B, we assumed that tasks are queued into a single unbounded FCFS queue. Whenever a machine becomes available, it selects a task from the queue and processes it. In support of this scheduling approach, we introduce the lemma II-B, demonstrating that it yields the minimum makespan for bag-of-tasks in heterogeneous computing systems.

Let $EET = [e_{ij}]$ $(1 \leqslant j \leqslant n)$ denote the EET vector of a heterogeneous computing system consisting of a set of machine types, $M = \{M_1, M_2, ..., M_n\}$, and a workload with c tasks of type T_i that are all available for execution (i.e., bag of tasks). Then, the minimum makespan (total time to complete the workload) is obtained using a Round-Robin load balancer across available machines.

Based on the Round-Robin load balancer for available machines, whenever a machine becomes free, it takes a task from the FCFS unbounded queue and executes it. Let $e_k = \max_{j=1}^{n} e_{ij}$, representing the slowest machine type for the task T_i. Based on the proof in Lemma II-B, the number of tasks completed on each machine is $n_j = c/\left(e_{ij} \sum_{j=1}^{n} \frac{1}{e_{ij}}\right)$ such that $\sum_{j=1}^{n} n_j = c$. Also, based on Lemma II-B, the makespan, denoted by τ^*, is determined as $\tau^* = n_j \cdot e_{ij} = \left(c / \sum_{j=1}^{n} \frac{e_k}{e_{ij}}\right) \cdot e_k$. If there are other fractions of tasks completed on machines, denoted by n'_j $(1 \leqslant j \leqslant n)$, such that the resultant makespan, denoted τ', is less than τ^*; thus, for each machine, n'_j must be less than n_j. If there exists a machine with $n'_j > n_j$, then the corresponding makespan of the tasks completed on that machine will be $\tau' = n'_j \cdot e_{ij} > n_j \cdot e_{ij} = \tau^*$, which is in contradiction with the primary assumption $\tau' < \tau^*$. Thus, we have $n'_j < n_j$, and for the total number of tasks completed on the machines, we have $\sum_{j=1}^{n} n'_j < \sum_{j=1}^{n} n_j = c$. Thus, we cannot obtain a makespan less than τ^* with the same number of tasks. This proves that assigning tasks that are all available for execution on available machines in a Round-Robin manner results in a minimum makespan.

C. Task Homogenization

In Section II-B, we consider the row speed-up vector to characterize machine heterogeneity for a given task type. Likewise, in task homogenization, we define *column speedup vector*, denoted by $\overrightarrow{\beta}_{(j)}$, to characterize task heterogeneity for machine type j. Then, we summarize the column speedup vector into a representative mean value, denoted $\bar{\beta}^H_{(j)}$. In Lemma II-C, we prove that the harmonic mean is an accurate measure of the central tendency of $\overrightarrow{\beta}_{(j)}$.

Let $EET = [e_{ij}]$ $(1 \leqslant i \leqslant m)$ denote the expected execution time (EET) vector of a set of task types, $T = \{T_1, T_2, ..., T_m\}$ in the machine type M_j. A workload trace of c tasks $(c > m)$ of type $T_i \in T$ arrive at the system. Tasks are queued on arrival in a single unbounded FCFS queue and executed by the machine M_j. Then, the true speedup is determined as follows:

$$\Gamma_T = \frac{1}{\sum_{i=1}^{m} \frac{\omega_i}{\beta^i_{(j)}}} = \bar{\beta}^H_{(j)} \tag{6}$$

Assume that the k^{th} task type is the largest task type. That is, e_{kj} is the maximum value in the j^{th} column of EET. Then, for a homogeneous workload that contains only the type of task T_k, the total time consumed by the machine M_j to perform those tasks is $c \times e_{kj}$. However, for a heterogeneous workload with ω_i as the proportion of each task type to the total number of tasks, the total time required to execute each task type by machine M_j is $c \times \omega_i \times e_{ij}$. Thus, the total time consumed to process all tasks is $\sum_{i=1}^{m} c \cdot \omega_i \cdot e_{ij}$. Then, the speed-up to execute the heterogeneous workload, as opposed to the homogeneous workload of type T_k, is determined based on Equation 7.

$$\Gamma_T = \frac{c \cdot e_{kj}}{\sum_{i=1}^{m} c \cdot \omega_i \cdot e_{ij}} = \frac{1}{\sum_{i=1}^{m} \omega_i \cdot \frac{e_{ij}}{e_{kj}}} \tag{7}$$

Based on Equation 2, we know that $\beta^i_{(j)} = \frac{e_{kj}}{e_{ij}}$. Furthermore, the weighted harmonic mean of the column speedup vector due to the heterogeneity of the task, denoted by $\bar{\beta}^H_{(j)}$, is calculated based on Equation 8.

$$\bar{\beta}^H_{(j)} = \frac{1}{\sum_{i=1}^{m} w_i \frac{1}{\beta^i_{(j)}}} \tag{8}$$

Based on Equations 7 and 8, we prove that the true speedup due to task heterogeneity is the weighted harmonic mean of $\overrightarrow{\beta_j}$.

D. Homogeneous Equivalent Execution Time (HEET)

From lemmas II-B, we learn that the arithmetic mean represents the overall speed-up factor due to machine heterogeneity. Moreover, Lemma II-C, proves that the weighted harmonic mean represents the mean speedup due to task heterogeneity. Taking these into account, we can perform task homogenization on each column of the EET matrix and reduce each column vector $\overrightarrow{\beta}_{(j)}$ to its mean value $\bar{\beta}^H_{(j)}$. In this manner, we construct a row vector, denoted $\bar{\beta}^H = [\bar{\beta}^H_{(1)}, \bar{\beta}^H_{(2)}, ..., \bar{\beta}^H_{(n)}]$, whose contents summarize the execution time behavior of all types of tasks into an equivalent hypothetical task type (denoted T^*). We use Equation 6 to determine $\bar{\beta}^H_{(j)}$ for machine type j. Note that the mean speed-up due to task heterogeneity for machine type j is calculated with respect to the slowest task type for machine type j (homogeneous counterpart). Therefore, to determine the expected execution time of T^* in the machine type j, we consider that T^* is $\bar{\beta}^H_{(j)}$ times faster than the slowest task type in the machine type j. The resultant row vector of the expected execution time of T^* on machines describes the machine heterogeneity. In a similar approach, we can aggregate the machine types into a single hypothetical machine type (denoted M^*) whose execution time behavior is representative of the entire set of machine types in the heterogeneous system. To this end, we construct the speedup vector due to machine heterogeneity for T^*, denoted by $\overrightarrow{\alpha}^{(*)}$, based on Equation 9.

$$\alpha^{(*)}_j = \frac{\max_{i=1}^{m} e_{ij}}{\bar{\beta}^H_{(j)}} \tag{9}$$

Then, we use the arithmetic mean of $\overrightarrow{\alpha}^{(*)}$ to determine the mean speed-up due to machine heterogeneity, denoted by $\bar{\alpha}^{(*)(A)}$ for T^*. In fact, $\bar{\alpha}^{(*)(A)}$ represents how much M^* is faster than the slowest machine type for the hypothetical equivalent task type (T^*). As a result, we can represent the execution time behavior of the heterogeneous computing system with Homogeneous Equivalent Execution Time, HEET, using the expected execution time of T^* on M^* as follows:

$$HEET = \frac{\max_{j=1}^{n} \alpha^{(*)}_j}{\bar{\alpha}^{(*)(A)}} \tag{10}$$

For clarity, we use Figure 1 to illustrate the derivation of HEET using an example. In Stage (a) of the figure, an EET matrix is considered. Then, in Stage (b), the EET matrix is used to derive the speedup matrix due to task heterogeneity based on Equation 2. The column j of this matrix demonstrates the speed-up due to the heterogeneity of the tasks for the type of machine j, denoted by $\overrightarrow{\beta}_{(j)}$. Next, in stage (c), we employ the harmonic mean (Equation 8) to represent each column vector $\overrightarrow{\beta}_{(j)}$ in the form of a scalar value. In this way, the set of speedup vectors due to task heterogeneity (which constructs a matrix) is reduced to a row speedup vector. In Stage (d), we calculate the expected execution time of T^* on machines by using the execution time of the slowest task type and $\bar{\beta}^H_{(j)}$. In Stage (e) we determine the speedup vector due to machine heterogeneity for T^*, based on Equations 1. In stage (f), we use the arithmetic mean to determine the mean speedup due to the heterogeneity of the machine for T^*. Lastly, in stage (g), we calculate the HEET measure by considering the speedup value obtained in stage (f) considering the execution time of the slowest machine type for T^* (M_1 in stage (d)) as the baseline.

HEET is able to accurately characterize system heterogeneity, and our hypothesis is that, for a given workload, systems with similar HEET scores exhibit similar makespan, too. For a given workload trace with T task types, the system A with the set of heterogeneous machine types M_A offers a larger makespan than the heterogeneous system B with machine types M_B ($|M_A| = |M_B|$), if and only if $HEET_A > HEET_B$.

E. Estimating Makespan and Throughput Using HEET

The mathematical approach used to obtain HEET is actually to transform the EET matrix that represents the heterogeneous system into a homogeneous EET matrix whose elements are values of HEET. In other words, we proposed a mathematical formulation for homogenization that transforms a heterogeneous computing system into a hypothetical homogeneous system such that both perform similarly in terms of performance metrics such as makespan and throughput.

Recall that the HEET metric is the expected execution time of the hypothetical equivalent task type (T^*) on the hypothetical equivalent machine type (M^*). Replacing the machine types with M^* and task types with T^*, results in a homogeneous EET matrix, denoted by EET^*, whose elements are HEET values. The lemmas II-B and II-C prove that the homogeneous system represented by the matrix EET^* exhibits a similar makespan as the heterogeneous system represented by the EET matrix. Given a workload of c of tasks, its makespan, denoted by τ, on a heterogeneous computing system can be estimated by the makespan of the same workload on the homogeneous equivalent system represented by EET^*. For a homogeneous system of n machines (M^*),

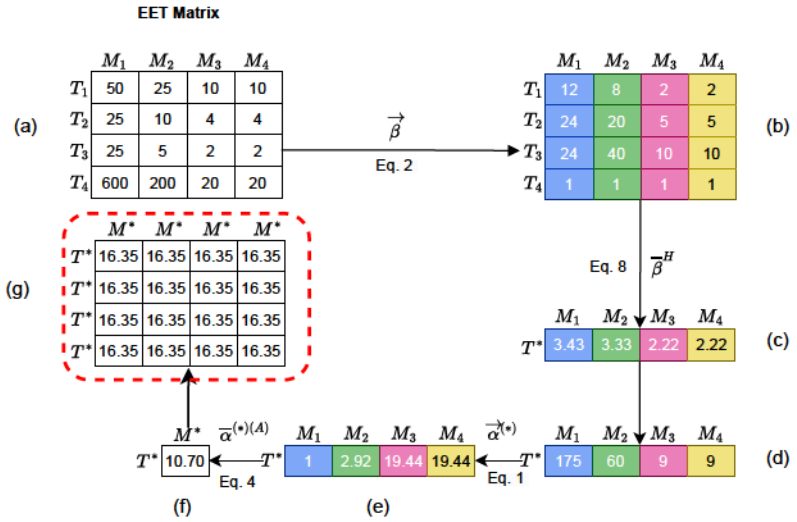


Fig. 1: An example illustrating the stages to calculate the heterogeneity measure (HEET). (a) The EET matrix representing a heterogeneous system. (b) The speedup matrix due to task heterogeneity is derived from EET matrix based on Equations 2. (c) Based on Equation 8, we consolidate each column into a mean value using the harmonic mean. (d) Calculate the expected execution time of T^* on machines using the execution time of the slowest task type and $\overline{\beta}_j^H$. (e) The speedup vector due to machine heterogeneity for T^*, based on Equations 1. (f) we employ Equation 4 on the resultant speedup vectors to determine the mean speedup value due to machine heterogeneity. (g) HEET score represents the execution behavior of the heterogeneous system.

the expected makespan of executing c tasks of the same type (T^*) is the number of tasks distributed on each machine ($\frac{c}{n}$) multiplied by the expected execution time of that task type on the machine type (i.e., HEET value). As a result, the makespan is determined as follows:

$$\tau = \frac{c}{n} \cdot HEET \tag{11}$$

Similarly, we can use the HEET score to estimate the system throughput, denoted by θ, as follows:

$$\theta = \frac{n}{HEET} \tag{12}$$

Furthermore, to facilitate comparison across heterogeneous computing systems with varying numbers of machines, we normalize the HEET score by the number of machines, resulting in $HEET/n$, denoted as S-HEET.

III. System Design

In this section, we present an overview of the components that we have designed to evaluate the HEET metric. Our implementation includes a real-world end-to-end "inference system", customized to match real-world production scenarios [9]. Throughout our experimentation, we used AWS EC2 instances as machines. Note that, with slight modifications, the system can also be readily deployed on alternative cloud platforms. The primary objective of our system is to validate the precision of the estimated makespan based on Equation 11 by comparing them against the actual makespan in various configurations of the system. Figure 2 illustrates the overall architecture of the system. The components of the system are explained in the following paragraphs.

Model Loader To encompass a diverse range of model types, we used the extensive model repository offered by HuggingFace [10]. We used four different models in our experiment: (1) Image classification: Resnet50 [11], (2) Object detection: Yolov5 [12], (3) Question answer: DistilBERT [13], and (4) Speech recognition: Wav2vec2 [14]. Given the variety of deep learning frameworks from which these models were sourced, we sought consistency in our experiments. To achieve this, we converted all models to the ONNX format (Open Neural Network Exchange) [15] using the PyTorch ONNX converter [16]. Using ONNX gives us the advantage of a unified model server setup, applicable across all model types.

Model Server Each of the models used during the experiments should be deployed as a machine learning service. We have used the multi-model serving capability of modern inference systems [17] to encapsulate multiple models into a single inference service. Each of the services is supported by an AWS EC2 instance. On each machine, we spin up a containerized version of the NVIDIA Triton Inference Server [18] and load all model variants onto it. Communications to model servers are implemented using gRPC [19] due to its superior performance compared to other transfer protocols [20].

Workload We synthesized the workload traces assuming that

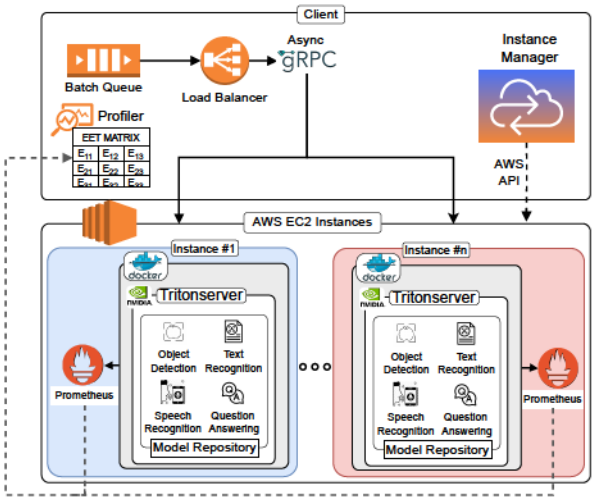


Fig. 2: Performance comparison of the same set of scheduling methods with the same workload across two computing systems, A and B, with different levels of heterogeneity (horizontal axis). The vertical axis shows the percentage of tasks completed on time.

the inter-arrival time between tasks in the workload traces follows an exponential distribution with the mean arrival rate as its parameter [21]. According to the bag-of-task assumption [2], we have also designed a workload of tasks that are all available for execution from the beginning (i.e., arrival time is zero). Tasks are sent asynchronously to the machines that host the ML services based on their arrival times.

Monitoring The monitoring component in each of the EC2 instances is equipped with Prometheus [22], a monitoring system, and a time series database. It records the inference time of all model servers during experiments and is stored in the database for later analysis. To construct the EET matrix, Profiler compares each task type with machine types and employs Prometheus to obtain the expected inference times.

Instance Manager During experiments, it is necessary to reconfigure heterogeneous computing systems with a different number of instances of each type. To support this, we have automated the process of reconfiguring the system in a central instance manager. The process of transitioning between two instance configurations includes (1) removing the current instances and cleaning the cluster, (2) setting up the new sets of machines with required dependencies, (3) bringing up the Triton container on top of the EC2 instances, and finally loading the models to the Triton inference server. We have automated all steps 1-4 using the AWS Python SDK [23]. Furthermore, the types of EC2 instance that we used during our experiments are (1) t2.large, (2) c5.2xlarge and (3) g4dn.xlarge. These instance types are similar to slow, medium, and fast machines in inferring the selected machine learning tasks.

Load Balancer Tasks are queued in a single unbounded FCFS queue upon arrival. Then, the load balancer assigns tasks to available machines in a round-robin manner. Mapping events are triggered by the completion or arrival of a task. In case a task arrives at the system while there is no available machine, the load balancer defers the mapping event until a machine becomes free.

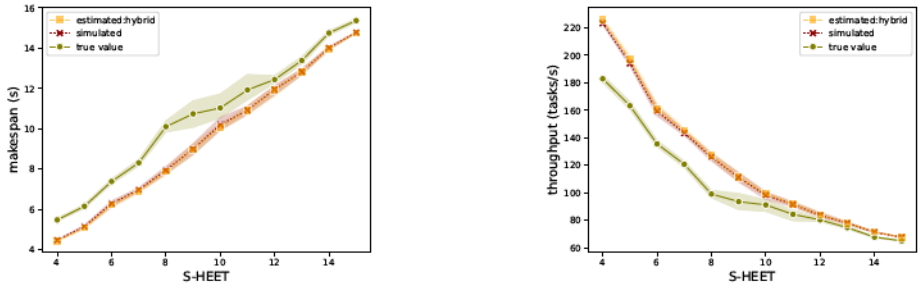
IV. Experimental Validation

A. Experimental Setup

To validate the developed heterogeneity measure, in this section, we execute a workload of four deep learning applications on various combinations of three types of Amazon EC2 instances (machine) to verify the HEET score in real-world settings. Specifically, we used four different applications/models in our experiment: (1) image classification implemented using the Resnet50 model [11], (2) object detection implemented according to the Yolov5 model [12], (3) question-answering based on the DistilBERT model [13], and (4) speech recognition using the Wav2vec2 model [14]. For machine types, we utilize GPU-based (`g4dn.xlarge`), compute-optimized (`c5.2xlarge`), and general-purpose CPU-based (`t2.large`) Virtual Machines offered by AWS EC2 services. To obtain the expected execution time of the image classification task on these machines, we processed 1000 sample images on each instance type. We repeat the experiment 10 times, and finally we use the expected value of these 10,000 inference operations in the EET matrix. Similarly, for the object recognition task, we ran the object recognition task for 1000 sample images 10 times to determine the expected execution time of the object recognition task. For the speech recognition task type, we execute a recorded audio of length 4 seconds on the machine types. Then, the average value of the inference times is used to fill the EET matrix. For question answering, we provide a sample context and question as input of the inference task and run the inference task 1000 times. We aggregated the inference times and determined the expected execution time for use in the EET matrix.

To synthesize the workload trace, we assume that the inter-arrival time between tasks in the workload traces follows an exponential distribution with the mean arrival rate as its parameter. For the bag-of-tasks, we assume that all tasks are available for execution from the beginning (arrival time is zero). Tasks are queued in a single unbounded FCFS arrival queue upon arrival and assigned to the available machines in the round-robin manner. Tasks are considered latency-sensitive and have hard deadlines. The performance metric (makespan) is defined as the time that the system requires to complete all tasks in the workload trace.

Recall that we employed hybrid central tendency measures (column-wise harmonic mean and row-wise arithmetic mean) to obtain a meaningful representative of the execution-time behavior of heterogeneous computing systems. To illustrate the effectiveness of our method, we compare the experimental results with the following baselines as representatives of the execution time behavior of heterogeneous systems: (1) Arithmetic mean, (2) Harmonic mean, and (3) Geometric mean of the elements of the EET matrix.



(a) (b)

Fig. 3: The system performance in terms of the makespan (left) and throughput (right) (vertical axis) upon varying S-HEET scores (horizontal axis) for workloads 1000 tasks. Each individual point represents the average result of multiple computing systems with the same S-HEET score. Furthermore, the colored area illustrates the 95% confidence interval of the results.

B. Estimating Throughput via HEET Score

In this experiment, our goal is to assess the ability of the HEET score to estimate the true throughput and the true makespan of the heterogeneous computing systems using Equations 12 and 11, respectively. To study the behavior of heterogeneous systems with varying HEET scores, we simulate a variety of heterogeneous systems with three types of machines and four types of tasks. To this end, we generate 228 different system configurations with a different number of instances of each type (i.e., `t2.large`, `c5.2xlarge`, and `g4dn.xlarge`). Then, for each system configuration, we feed it with a bag of 1000 tasks of four types (i.e., image classification, object detection, question answering, and speech recognition). Eventually, we averaged the makespan and throughput in different system configurations with the same HEET score.

Figure 3 shows the results of the true and estimated makespan and throughput with varying S-HEET score, which is the HEET score scaled with the number of machines (i.e., $\frac{HEET}{n}$). Each (makespan, S-HEET)/(throughput, S-HEET) point shows the average makespan and throughput of different system configurations with the same S-HEET value. The colored area illustrates the 95% confidence interval of the results. As shown in Figure 3, we can observe that systems with lower S-HEET scores generally perform better (i.e., lead to a smaller makespan or higher throughput) than those with higher HEET score values. This statement itself means that the S-HEET score can be effectively used as a measure to "*compare different heterogeneous computing systems*" in terms of makespan or throughput. Moreover, we can observe that the results exhibit a narrow confidence interval, that is, systems with the same S-HEET score will perform similarly in terms of makespan and throughput. The number of different configurations with S-HEET scores equal to 9, 10, or 11 is small (less than 9); therefore, we observe a wider confidence interval for these S-HEET scores in the results.

In addition, the results show that the estimated values (makespan and throughput) using the HEET score based on Equations 12 and 11, respectively, can predict the true values with an average accuracy of 84%. Note that we used the expected values of the execution times to determine the HEET score. As a result, the accuracy of the estimation depends on the degrees of uncertainty that exist in the execution times. Consequently, we ran a simulation with zero uncertainty in expected execution times to demonstrate the root cause of the error in estimating the makespan and throughput using the HEET score. The results show that the makespan calculated using the HEET score accurately estimates the makespan of the simulation. Thus, we can say that in heterogeneous systems with low levels of uncertainty in execution times, estimating makespan using HEET score is an accurate method.

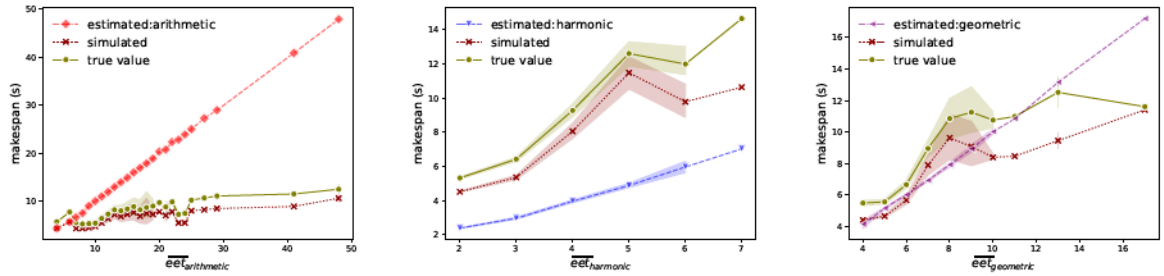
Given a user-defined throughput threshold, we can determine the corresponding HEET score and use it to configure a heterogeneous system with the desired throughput. For a desired throughput, the HEET score enables solution architects to proactively configure a heterogeneous system that can meet that objective (instead of try and error) with minimum cost.

In summary, the result of this experiment validates the applicability of the HEET score for real-world scenarios. In particular, when the HEET score is applied across systems, it can accurately compare different heterogeneous systems with respect to their performance metrics (makespan and throughput) without examining the workload in these systems.

C. Other heterogeneity measures

In this experiment, we investigate the effectiveness of the arithmetic, harmonic, and geometric mean of expected execution times of task types in machine types as heterogeneity measures representing the execution behavior of heterogeneous systems. To this end, we conducted a similar experiment as in Section IV-B to study the performance (in terms of makespan) of 228 heterogeneous computing systems. An appropriate heterogeneity measure should be able to identify similarity and superiority in terms of performance metrics (e.g., makespan or throughput) across heterogeneous systems.

Figure 4 shows the results of the makespan for different heterogeneous systems with varying mean expected execution



(a) (b) (c)

Fig. 4: The system performance in terms of the makespan (vertical axis) with respect to the varying mean of the EET matrix (horizontal axis) for a workload of 1000 tasks. In (a)–(c), $\overline{eet}_{\text{arithmetic}}$, $\overline{eet}_{\text{harmonic}}$, and $\overline{eet}_{\text{geometric}}$ are the arithmetic, harmonic, and geometric mean of the expected execution times of task types on machines types as represented in the EET matrix. Each individual point represents the average result of multiple computing systems with the same mean EET value.

times calculated based on arithmetic ($\overline{eet}_{arithmetic}$), harmonic ($\overline{eet}_{harmonic}$), and geometric ($\overline{eet}_{geometric}$) mean techniques. In Figure 4a, the results show that the estimated makespan using $\overline{eet}_{arithmetic}$ cannot follow the true makespan and is considerably inaccurate. Similarly, we observe that the harmonic mean and geometric mean heterogeneity measures are also inaccurate.

In summary, comparing the results for the baseline heterogeneity measures, as shown in Figure 4, with the results for the HEET score, as shown in Figure 3, verifies that heterogeneous systems are well characterized with the HEET measure and can be used to accurately estimate the performance of heterogeneous systems.

V. Related works

Heterogeneous computing systems utilize various computing machines to perform various tasks with different computational requirements. The idea of exploiting system heterogeneity to improve system performance considering different objectives such as energy [24], [25] and QoS [26], [27], [28] has been extensively explored in the literature. Based on these works, it has been proven that heterogeneity can play a crucial role in improving different system performance metrics; however, these works fall short of providing a concrete metric that can explain the impact of heterogeneity on system performance.

Expected Time to Compute Matrix (ETC) The idea of characterizing a heterogeneous computing system using the Expected Time-to-Compute (ETC) matrix was first explored by Ali et al. [29]. They used the coefficient-of-variation of expected execution times as a measure of heterogeneity. Then, they suggested an algorithm that takes the mean and standard deviation of execution times to generate the ETC matrix of the heterogeneous system. However, their method neither characterizes the performance of different heterogeneous systems nor makes them comparable. In contrast, we present a mathematical model to measure system heterogeneity such that it can characterize the overall system performance behavior and make different systems comparable.

Moreover, Mokhtari et al. [2], leveraged the ETC matrix to model the performance behavior of heterogeneous computing systems and introduced a fair and energy-aware scheduling algorithm. Narayanan et al. [30] proposed a throughput matrix to model the performance behavior of the system. Specifically, each entry (i, j) in the matrix represents the performance of the job i on the machine j. The matrix also implies system heterogeneity; hence, they leveraged it to devise a heterogeneity-aware scheduling method that can be optimized for different performance metrics.

Heterogeneity-aware Machine Learning Inference Serving Systems Several research works have been conducted to devise heterogeneity-aware machine learning inference services considering performance objectives such as cost, QoS, or throughput. Kairos [31] is a deep learning inference serving framework that maximizes throughput under the cost budget and QoS constraint. For that purpose, they proposed a heterogeneity-aware query distribution mechanism that maximizes throughput. In these works, it is assumed that the performance (i.e., throughput) of a heterogeneous system cannot be mathematically described; however, in our work, we propose an analytical approach to accurately estimate the throughput of a heterogeneous system. This would help to find the optimal configuration that minimizes cost while meeting the throughput target.

VI. Conclusion

In this research, we provided a measure to quantify the heterogeneity of the system with respect to a performance metric (make-span or throughput) and for a given set of task types. For this purpose, we proposed a homogenization process to determine a hypothetical homogeneous configuration such that the throughput remains unchanged. We characterize system heterogeneity by decoupling heterogeneity into machine and task dimensions and perform the homogenization process in each dimension. In machine homogenization, we proved that the expected execution time of the equivalent machine is determined using the arithmetic mean of the speed-up vector with the slowest machine as the baseline. We also proved that

the harmonic mean can measure the mean speedup due to task heterogeneity. Finally, we introduce the Homogeneous Equivalent Execution Time (HEET) score as a heterogeneity measure that represents the speed of a heterogeneous system for a given set of task types. In this way, we transform a heterogeneous system into a hypothetical equivalent homogeneous system with similar performance metrics (makespan and throughput). HEET can be used as a score in the heterogeneity spectrum to globally compare the performance of different heterogeneous systems. We observe that the HEET score can effectively estimate the makespan and throughput with an average and minimum accuracy of 84% and 80.0%, respectively. We also show that the main source of error is the uncertainty in execution times. We conclude that the HEET score enables solution architects to proactively configure a heterogeneous system, instead of the current approach which is primarily based on trial and error. In the future, we plan to incorporate probabilistic analysis into our analysis to capture uncertainties in execution times. Another avenue for future work is to extend the dimensions of heterogeneity by considering power heterogeneity.

Acknowledgment

We would like to thank anonymous reviewers for their constructive feedback. This project is supported by the National Science Foundation (NSF) through Award# 2107463, 2038080, 2419588, 2417064, 2007202, and 2233873.

References

[1] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th annual international symposium on Computer architecture*, 2011, pp. 365–376.

[2] A. Mokhtari, M. A. Hossen, P. Jamshidi, and M. A. Salehi, "Felare: Fair scheduling of machine learning tasks on heterogeneous edge systems," in *Proceedings of the 15th IEEE International Conference on Cloud Computing (IEEE Cloud)*, 2022, pp. 459–468.

[3] S. Zobaed, A. Mokhtari, J. P. Champati, M. Kourouma, and M. A. Salehi, "Edge-multiai: Multi-tenancy of latency-sensitive deep learning applications on edge," in *Proceedings of 15th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2022.

[4] S. Allmeier and N. Gast, "Mean field and refined mean field approximations for heterogeneous systems: It works!" in *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 6, no. 1, 2022, pp. 1–43.

[5] D. J. Lilja, *Measuring computer performance: a practitioner's guide*. Cambridge university press, 2005.

[6] L. K. John, "More on finding a single number to indicate overall performance of a benchmark suite," *ACM SIGARCH Computer Architecture News*, vol. 32, no. 1, pp. 3–8, 2004.

[7] J. E. Smith, "Characterizing computer performance with a single number," *Communications of the ACM*, vol. 31, no. 10, pp. 1202–1206, 1988.

[8] M. A. Oxley, S. Pasricha, A. A. Maciejewski, H. J. Siegel, J. Apodaca, D. Young, L. Briceno, J. Smith, S. Bahirat, B. Khemka *et al.*, "Makespan and energy robust stochastic static resource allocation of a bag-of-tasks to a heterogeneous computing system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2791–2805, 2014.

[9] Salesforce, "Real-world examples of machine learning." [Online]. Available: https://www.salesforce.com/eu/blog/2020/06/real-world-examples-of-machine-learning.html

[10] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] G. Jocher, "YOLOv5 by Ultralytics," https://github.com/ultralytics/yolov5, May 2020, (Accessed: Aug. 2, 2023).

[13] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[14] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[15] O. D. Team, "Onnx github repository," https://github.com/onnx/onnx, (Accessed: Aug. 2, 2023).

[16] Microsoft, "Convert a pytorch model to onnx for windows machine learning," 2023, (Accessed: August 2, 2023). [Online]. Available: https://learn.microsoft.com/en-us/windows/ai/windows-ml/tutorials/pytorch-convert-model

[17] Seldon, "What is multi-model serving and how does it transform your ml infrastructure?" https://www.seldon.io/what-is-multi-model-serving-and-how-does-it-transform-your-ml-/infrastructure, 2023, (Accessed: Aug. 2, 2023).

[18] N. Corporation, "Triton inference server github repository," https://github.com/triton-inference-server/server, (Accessed: Aug. 2, 2023).

[19] "GRPC." [Online]. Available: https://grpc.io/

[20] M. Techlabs, "Comparison between grpc vs. rest," https://marutitech.com/rest-vs-grpc/#Comparison_Between_gRPC_vs_REST, (Accessed: Aug. 2, 2023).

[21] M. Harchol-Balter, *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[22] "prometheus." [Online]. Available: https://prometheus.io/

[23] A. W. Services, "Aws sdk for python," https://aws.amazon.com/sdk-for-python/, (Accessed: Aug. 2, 2023).

[24] S. K. Panda and P. K. Jana, "An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems," *Journal of Cluster Computing*, vol. 22, no. 2, pp. 509–527, 2019.

[25] S. Ghafouri, A. A. Saleh-Bigdeli, and J. Doyle, "Consolidation of services in mobile edge clouds using a learning-based framework," in *Proceedings of IEEE World Congress on Services (SERVICES)*, 2020, pp. 116–121.

[26] S. M. Hussain and G. R. Begh, "Hybrid heuristic algorithm for cost-efficient qos aware task scheduling in fog–cloud environment," *Journal of Computational Science*, vol. 64, p. 101828, 2022.

[27] A. Mokhtari, C. Denninnart, and M. A. Salehi, "Autonomous task dropping mechanism to achieve robustness in heterogeneous computing systems," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020, pp. 17–26.

[28] C. Denninnart, J. Gentry, A. Mokhtari, and M. A. Salehi, "Efficient task pruning mechanism to improve robustness of heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, vol. 142, pp. 46–61, 2020.

[29] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, S. Ali *et al.*, "Representing task and machine heterogeneities for heterogeneous computing systems," *Journal of Applied Science and Engineering*, vol. 3, no. 3, pp. 195–207, 2000.

[30] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "Heterogeneity-aware cluster scheduling policies for deep learning workloads," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 481–498.

[31] B. Li, S. Samsi, V. Gadepally, and D. Tiwari, "Kairos: Building cost-efficient machine learning inference systems with heterogeneous cloud resources," in *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, 2023, pp. 3–16.