Fleet-Level Optimization for Heterogeneous Device Provisioning in Machine Learning Model Deployments

Harry H. Jiang
Carlee Joe-Wong
{hhj,cjoewong}@andrew.cmu.edu
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

1 INTRODUCTION

With the increasing popularity of machine learning (ML) applications, networked computing systems like cloud datacentres are facing rapidly growing demand for ML inference tasks. Expanding systems to meet these demands, however, can be both costly and environmentally unsustainable. While several computer architecture optimizations have been proposed to alleviate monetary and environmental costs, e.g., by reusing older devices or those less customized to ML workloads, the resulting cost and performance tradeoffs at a platform level have yet to be modelled.

A property of large language model (LLM) inference tasks which sets them apart from many other computing tasks is that the computation time of an inference task (dominated by output token inference) is correlated with the length of its input [6]. Especially as shorter-input requests reliably return fewer tokens, it is possible to isolate these tasks and allocate them on dedicated devices; indeed, a form of this segmenting currently occurs in deployment. As such, it is possible to use older or slower computation units to handle this segment of demand. We present a preliminary cost model for optimizing multiple device types within a fleet of servers, where for the case of workload deterministically correlated to input length, we devise an algorithm to fit linear bounds such that the devices may fulfill a set of service-level objectives. We show that in this scenario lends to optimization by linear programming, and we present future work to bring the model closer to realistic operational considerations.

This work aims to contribute to the existing area of research of ML deployment optimization, especially for LLMs. On the software and queuing level, there exist methods to partition a larger task into smaller jobs for efficient distribution within a network [2, 3]. Much existing work also occurs at the architectural support level in terms of dividing workloads across multiple processing units and optimizing throughput of ML inference requests [4, 6, 10]. Alongside throughput optimization, there are attempts to study and reduce environmental impact of ML datacenter operations, such as reducing carbon emissions, water use, or energy consumption [1, 5, 8]. We also draw upon research on optimization and provisioning in under device heterogeneity, especially for ML workloads [7, 9].

2 COST MODEL

We consider an ML service infrastructure composed of a variety of devices of different specifications and ages, and various cost modalities to minimize, e.g., money, carbon emissions, water use, etc. For simplicity, we define each type of device D as having the same set of costs and computing performance. In order to satisfy a request volume of R jobs (i.e., ML inference calls), jobs are allocated to the different devices under the constraint that, in aggregate, the device allocation must be able to serve the incoming job volume. In this initial model, we hold incoming volume constant.

Table 1: Key notation.

Symbol	Definition
v_D	Volume of device type D provisioned
u_D	Proportion of computing capacity from device D
$c_{D,i}$	Cost for a given cost type i, device D
R	Request volume per unit time
τ_D, $\bar{\tau}_D$	runtime and mean runtime of a job on device D

We propose a model for optimizing device procurement and deployment at the fleet level as follows:

$$\begin{aligned} \min_{u_D, v_D} \quad & Cost = \sum_D v_D \sum_i \gamma_i c_{D,i} \\ \text{s.t.} \quad & B\vec{u} \leq \vec{b}, \quad \sum_D u_D = 1 \\ \text{where} \quad & v_D = u_D R \bar{\tau}_D \end{aligned} \tag{1}$$

Table 1 summarizes the notation used in this problem formulation. The objective function in Equation (1) consists of a weighted sum of various cost modalities with weight parameters $\gamma_i \geq 0$; these are constant across all device types and represent the relative importance of the cost modality to the overall optimization objective. Different platforms may adjust the γ_i weights depending on their priorities. We choose the provisioned volume v_D of device type D and the proportion of computing capacity u_D used of device type D so as to minimize this objective. We ensure that these variables are compatible by specifying $v_D = u_D R \bar{\tau}_D$, where $\bar{\tau}_D$ denotes the average runtime of a job on device D. Thus, the provisioned volume is exactly determined by the proportion of computing capacity u_D and amount or workload $R\bar{\tau}_D$ on device type D. In addition, we constrain the sum of the proportions u_D to 1, so that they are well-defined.

The most important element of this model is the set of linear constraints $B\vec{u} \leq \vec{b}$, a set of bounds determined by the service-level objectives (SLO) of the inference task and the distribution of service demand. We next detail how these are specified.

2.1 Device allocation for *a priori* latency distribution

In this work, we present a simplified scenario where task workload is deterministic given some measurable property of the input (such as token length, for a large language model inference workload). As a result, for each device D, a given workload corresponds to a known runtime, and workloads are deterministically allocated to an appropriate device. We suppose that the input length is characterized by a cumulative probability distribution (CDF) $P_{in}(w)$, e.g., a distribution of token lengths. Since the runtime is deterministic given input size, this input distribution then induces a distribution on runtime for device D. The SLO can then be defined as bounds on the percentiles of the resulting runtime distribution, e.g., 95% of runtimes not exceeding a certain latency. Algorithm 1 computes the resulting linear bounds for our optimization formulation (1).

Algorithm 1: Construction of linear inequality $B\vec{u} \leq \vec{b}$.

Parameter: n_D device types, n_S SLOs,
x_S percentile requirement of SLO S,
$\vec{u}$ proportion of each device type,
$P_{in}^{-1}(\sigma)$ inverse cdf of input size,
$\sigma_{D,S}$ input size meeting SLO S on device D

1 sort all $\sigma_{D,S}$ into array $\vec{\sigma}$ from highest input size to lowest
2 create array $\vec{D}$ of corresponding index of $\vec{\sigma}$ such that for each index i in $\vec{\sigma}$, D_i is the resource type associated with σ_i
3 B := empty matrix with n_D columns and zero rows
4 $d := [0\ 0\ 0]$ //where each d_j is the highest index of SLO for each type of device j with an already allocated bound
5 **foreach** (i, σ_i) **do**
6 $\quad d_{D_i} := d_{D_i} + 1$ //increment SLO index for D_i
7 $\quad$ InsertRow(B, $[x_{d_j}$ for j in $(1 \ldots n_D)]$)
$\quad$ //all but one element of the row (at index D_i) should be same as the row above
8 $b := P_{in}^{-1}(\vec{\sigma})$ //map P_{in}^{-1} on every element of $\vec{\sigma}$

Since the cost function in (1) is a linear function of $\vec{v}$, the function is monotonic in all dimensions of the search space. Additionally, since the only constraints applied in the model are linear and convex (Figure 2), the optimization problem can be sufficiently solved using linear programming.

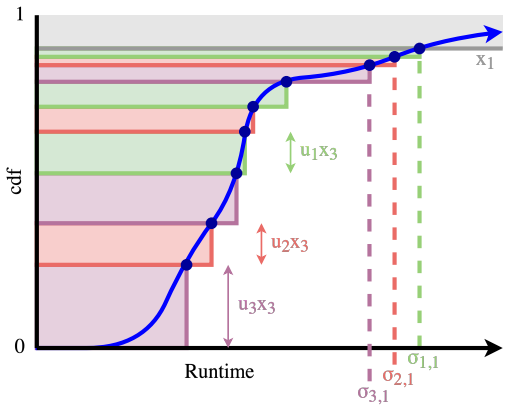

Figure 1: Visual interpretation of Algorithm 1. The rows of B correspond fo the rectangles from top to bottom, with the grey rectangle representing the first n_D rows as tasks longer than the highest SLO percentile do not have constraints. The remaining rectangles are "stacked" below the line at x_1 such that they can cover the entirety of the cdf function $P_{in}(w)$.

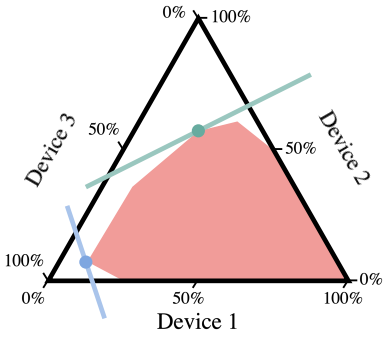

Figure 2: Visualization of bounds $B\vec{u} \leq \vec{b}$ (shaded red region) in (1) in a three-device scenario. The bounds form a convex polytope on the $n_D = 3$-simplex, for which the optimum can be solved using linear programming. The green and blue lines represent possible tangents formed by the cost function and the optimal device composition in each case.

3 FUTURE WORK

We plan to use this formulation to study realistic tradeoffs between different cost modalities, e.g., monetary, carbon emissions, and water usage cost in future work. As this model is significantly simplified from real-world operations, we further outline a set of extensions which will allow our model to better capture device procurement and deprecation practices and incentives for large-scale ML inference services. First, we aim to loosen the deterministic relationship between the input length of LLM inference tasks and runtime and develop an allocation method for this probabilistic relationship. Another element to study is the scenario of incomplete SLO compliance, and the tradeoffs between cost and various degrees of compliance. In addition, we intend to explore the properties of inference on other ML models and integrate a generalized representation of ML inference workloads into our model, as well as provisioning considerations in the case of dynamic demand volume and distribution.

On the cost function, we envision further work on expanding (1) to take into consideration the age of devices and capital costs (e.g., purchase cost of device, embodied carbon). Moreover, we intend to model the marginal cost to using and replacing different types and ages of devices within a system. These elements would contribute towards a holistic device procurement and replacement strategy for ML inference infrastructure.

ACKNOWLEDGMENTS

This research was supported by the US Department of Energy under grant DESC0025652.

REFERENCES

[1] Nidhal Jegham, Marwan Abdelatti, Chan Young Koh, Lassad Elmoubarki, and Abdeltawab Hendawi. 2025. How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference. arXiv:2505.09598 [cs.CY] https://arxiv.org/abs/2505.09598

[2] Thaha Mohammed, Carlee Joe-Wong, Rohit Babbar, and Mario Di Francesco. 2020. Distributed Inference Acceleration with Adaptive DNN Partitioning and Offloading. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 854–863. https://doi.org/10.1109/INFOCOM41043.2020.9155237

[3] Akrit Mudvari, Yuang Jiang, and Leandros Tassiulas. 2024. SplitLLM: Collaborative Inference of LLMs for Model Placement and Throughput Optimization. arXiv:2410.10759 [cs.DC] https://arxiv.org/abs/2410.10759

[4] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. 118–132. https://doi.org/10.1109/ISCA59077.2024.00019

[5] Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Inigo Goiri, and Josep Torrellas. 2024. Towards Greener LLMs: Bringing Energy-Efficiency to the Forefront of LLM Inference. arXiv:2403.20306 [cs.AI] https://arxiv.org/abs/2403.20306

[6] Jovan Stojkovic, Chaojie Zhang, Íñigo Goiri, Josep Torrellas, and Esha Choukse. 2025. DynamoLLM: Designing LLM Inference Clusters for Performance and Energy Efficiency. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1348–1362. https://doi.org/10.1109/HPCA61900.2025.00102

[7] H. Topcuoglu, S. Hariri, and Min-You Wu. 2002. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems* 13, 3 (2002), 260–274. https://doi.org/10.1109/71.993206

[8] Jaylen Wang, Daniel S. Berger, Fiodar Kazhamiaka, Celine Irvene, Chaojie Zhang, Esha Choukse, Kali Frost, Rodrigo Fonseca, Brijesh Warrier, Chetan Bansal, Jonathan Stern, Ricardo Bianchini, and Akshitha Sriraman. 2024. Designing Cloud Servers for Lower Carbon. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. 452–470. https://doi.org/10.1109/ISCA59077.2024.00041

[9] Qiannan Zhou, Fei Xu, Lingxuan Weng, Ruixing Li, Xudong Wu, Li Chen, Zhi Zhou, and Fangming Liu. 2025. Espresso: Cost-Efficient Large Model Training by Exploiting GPU Heterogeneity in the Cloud. In *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications*. 1–10. https://doi.org/10.1109/INFOCOM55648.2025.11044693

[10] Kan Zhu, Yufei Gao, Yilong Zhao, Liangyu Zhao, Gefei Zuo, Yile Gu, Dedong Xie, Tian Tang, Qinyu Xu, Zihao Ye, Keisuke Kamahori, Chien-Yu Lin, Ziren Wang, Stephanie Wang, Arvind Krishnamurthy, and Baris Kasikci. 2025. NanoFlow: towards optimal large language model serving throughput. In *Proceedings of the 19th USENIX Conference on Operating Systems Design and Implementation* (Boston, MA, USA) *(OSDI '25)*. USENIX Association, USA, Article 41, 17 pages.