

GOGH: Correlation-Guided Orchestration of GPUs in Heterogeneous Clusters

Ahmad Raeisi^{1,5}, Mahdi Dolati², Sina Darabi³, Sadegh Talebi^{*5}, Patrick Eugster³, and Ahmad Khonsari¹

¹University of Tehran

²Sharif University of Technology

³Università della Svizzera italiana (USI)

⁵University of Copenhagen

ahmad.raeisi@di.ku.dk, mdolati@sharif.edu, sina.darabi@usi.ch, sadegh.talebi@di.ku.dk, patrick.eugster@usi.ch, khonsari@ut.ac.ir

001 Abstract

002 In heterogeneous clusters with varying capabilities
 003 and energy efficiency, sustainable use of mixed-
 004 generation resources is essential. We propose a
 005 method for adaptive management of machine learn-
 006 ing jobs, aiming to minimize energy while meeting
 007 performance targets which uses two neural networks
 008 to cope with hardware utilization uncertainties. We
 009 demonstrate the efficacy of this adaptive process via
 010 the Gavel benchmark [1].

011 1 Introduction

Recent advances in machine learning have led to large models and datasets, creating high computational demands. While specialized accelerators address these needs, upgrading all infrastructure is impractical; thus, heterogeneous clusters combining legacy and modern GPUs remain common [2, 3]. Efficient scheduling is challenging due to diverse job structures, resource contention, and limited understanding of job–hardware interactions [4, 5]. Historical performance data, however, can reveal patterns useful for predicting throughput across jobs and devices [6]. We present *GOGH*, a framework for scheduling deep learning jobs on heterogeneous GPU clusters by leveraging correlations across jobs and accelerators to predict throughput and guide allocations. Its key components include: (i) a neural predictor for initial and refined throughput estimates (using RNN, Feedforward (FF), and Transformer models), and (ii) an ILP-based optimizer that assigns jobs to GPUs while meeting throughput guarantees and minimizing power use. Using the Gavel dataset [1], we demonstrate that *GOGH* improves both prediction accuracy and scheduling efficiency over baselines.

We make the following contributions: (1) A correlation-guided framework for heterogeneous GPU scheduling; (2) an ILP formulation balancing

throughput, efficiency, and guarantees; (3) evaluation of neural architectures for prediction accuracy; and (4) experimental validation showing significant scheduling and prediction gains. We refer to the full version of the article [7] for detailed discussions. 039
040
041
042
043

Related work. Heterogeneity-aware scheduling. Gavel [1] introduced effective throughput for accelerator-aware scheduling; Pollux [8] co-adapts resource allocation and training configurations. **Resource sharing.** TGS [9] enables transparent GPU sharing at the OS level. Hisi-D [10] provides integrated

sharing at the OS layer; HiveD [10] provides virtual clusters for multi-tenant fairness and affinity. **Elastic/serverless training.** ElasticFlow [11] shows benefits of elasticity for deadline-aware training. **LLM serving.** Parrot [12] exposes semantic signals to reduce redundancy during inference. Our work differs by learning correlations to predict cross-device throughput and refining estimates online to guide energy-aware ILP-based allocation.

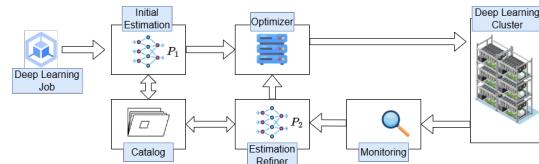


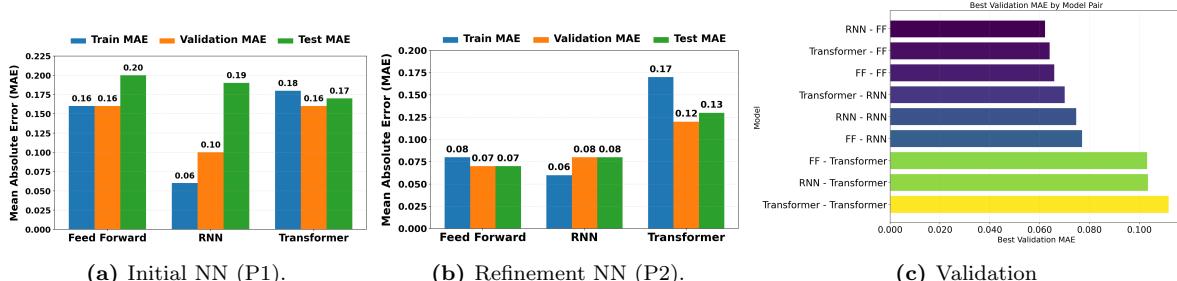
Figure 1. Architecture of GOGH.

2 GOGH Design

GOGH is a solution that dynamically decides GPU-job assignments in a deep learning cluster, as ML jobs arrive over time. Figure 1 illustrates the architecture of GOGH.

At its core, GOGH has an ILP-based *Optimizer* module that assigns jobs to GPUs subject to capacity and minimum-throughput constraints while minimizing power. However, job throughput on each GPU type is not known by the *Optimizer* apriori. Thus, GOGH uses four helper modules to improve its assignment in time. P_1 provides initial per-(job, accelerator, co-location) throughput estimates from a continuously updated *Catalog* of historical runs.

*Corresponding Author.



(a) Initial NN (P1).

(b) Refinement NN (P2).

(c) Validation

Figure 2. MAE of models; Combined validation of P1–P2 model pairs.

Monitor records realized throughput; P_2 refines the Catalog using these observations, improving future allocations. To cast this as an optimization problem, let \mathcal{S} be servers, \mathcal{A} GPU types, and \mathcal{J} ML jobs. GPU capacity of type a is K_a , which in practice for most GPUs is one or two. Define $\mathcal{C} \subseteq \{c \subseteq \mathcal{J} : |c| \leq 2\}$ as job combinations (job co-location) that can be assigned to GPUs. Let $T_{a,j}^c$ be throughput of job j on type a under co-location c . As is unknown, we define $\tilde{T}_{a,j}^{i,c}$ as the i -th estimate of $T_{a,j}^c$ computed by P_1 and P_2 . Specifically, P_1 writes the initial estimate $\tilde{T}_{a,j}^{0,c}$ into the Catalog and P_2 refines it after monitoring round $i > 0$.

Given estimated or measured throughput values, we allocate GPUs to jobs via an integer linear program (ILP) that minimizes total power consumption while meeting throughput and capacity constraints. We define binary variables $x_{a,s}^c$ indicating whether job combination c is assigned to accelerator a on server s . The objective (1a) minimizes power $\gamma_a(x)$, obtained from profiling. Constraints (1b)–(1e) ensure that each job is scheduled, accelerator capacities are respected, and minimum throughput T_j is met. We solve the formulation with a standard ILP solver; more efficient algorithms are left for future work.

$$\min_{x_{a,s}^c \in \{0,1\}} \sum_{a \in \mathcal{A}} \gamma_a \left(\sum_{c \in \mathcal{C}_j} T_{a,j}^c x_{a,s}^c \right) \quad (1a)$$

$$\text{s.t. } 1 \leq \sum_{s,a,c \in \mathcal{C}_j} x_{a,s}^c \leq D_j, \quad (1b)$$

$$\sum_{c \in \mathcal{C}} |c| x_{a,s}^c \leq K_a, \quad (1c)$$

$$\bar{T}_j \leq \sum_{a,c \in \mathcal{C}_j} T_{a,j}^c x_{a,s}^c, \quad (1d)$$

$$\sum_{c \in \mathcal{C}} x_{a,s}^c \leq 1. \quad (1e)$$

3 Evaluation

We use the Gavel benchmark [1], reporting throughput (iterations/sec) for diverse ML workloads (listed in Table 1) on six GPU types (k80, p100, v100 and unconsolidated variants). As for models, we implemented FF, RNN, and Transformer variants

Table 1. Workloads in simulation.

Application	Batch Sizes
ResNet-18	{16, 32, 64, 128, 256}
ResNet-50	{16, 32, 64, 128, 256}
Transformer	{16, 32, 128, 256}
LM	{5, 10, 20, 80}

for both P_1 and P_2 , with similar complexity. Models were evaluated using the mean absolute error (MAE) across the training, validation, and test sets.

Results. *Initial Estimation (P1).* On validation, RNN achieved the lowest MAE, while on test the Transformer generalized best. RNN fit training data strongly, but Transformer was more robust to distribution shift. *Refinement (P2).* FF provided the most consistent refinement with the lowest validation and test MAE, outperforming RNN and Transformer despite RNN’s lower training loss. Pairing P_1 and P_2 shows that $RNN \rightarrow FF$ yields the best overall validation MAE, outperforming $Transformer \rightarrow FF$ by $\approx 2.8\%$ in our setting. Capturing temporal patterns initially (RNN) and refining deterministically (FF) provides strong accuracy and generalization; see [7] for a detailed discussion.

Discussion. Estimates are stable under batch-size perturbations and reflect co-scheduling contention: similar jobs map to accelerators with comparable predicted throughput; conflicting co-locations show predicted degradation aligned with observations. These behaviors indicate that GOGH captures structural properties of workload–device interactions without hand-tuned rules.

4 Conclusion

We propose GOGH, a correlation-guided framework for managing GPUs in heterogeneous clusters. GOGH estimates job throughput across GPU types by leveraging inter-job and inter-GPU correlations, refining predictions with runtime observations. An ILP allocator minimizes power while meeting throughput goals. Among tested architectures, the RNN–FF pipeline achieved prediction errors as low as 5%. Future work includes ILP approximations and reinforcement learning for adaptive scheduling.

144

References

145 [1] D. Narayanan, K. Santhanam, F. Kazhami-
146 aka, A. Phanishayee, and M. Zaharia.
147 “Heterogeneity-Aware cluster scheduling poli-
148 cies for deep learning workloads”. In: *14th
149 USENIX Symposium on Operating Systems
150 Design and Implementation (OSDI 20)*. 2020,
151 pp. 481–498.

152 [2] T. Um, B. Oh, M. Kang, W.-Y. Lee, G. Kim,
153 D. Kim, Y. Kim, M. Muzzammil, and M. Jeon.
154 “Metis: Fast Automatic Distributed Training
155 on Heterogeneous GPUs”. In: *2024 USENIX
156 Annual Technical Conference (USENIX ATC
157 24)*. 2024, pp. 563–578.

158 [3] Y. Mei, Y. Zhuang, X. Miao, J. Yang, Z. Jia,
159 and R. Vinayak. “Helix: Serving Large Lan-
160 guage Models over Heterogeneous GPUs and
161 Network via Max-Flow”. In: *Proceedings of
162 the 30th ACM International Conference on
163 Architectural Support for Programming Lan-
164 guages and Operating Systems, Volume 1*. 2025,
165 pp. 586–602.

166 [4] G. Yeung, D. Borowiec, A. Friday, R. Harper,
167 and P. Garraghan. “Towards GPU utilization
168 prediction for cloud deep learning”. In: *12th
169 USENIX Workshop on Hot Topics in Cloud
170 Computing (HotCloud 20)*. 2020.

171 [5] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang,
172 J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding.
173 “MLaaS in the wild: Workload analysis and
174 scheduling in Large-Scale heterogeneous GPU
175 clusters”. In: *19th USENIX Symposium on
176 Networked Systems Design and Implemen-
177 tation (NSDI 22)*. 2022, pp. 945–960.

178 [6] W. Xiao, R. Bhardwaj, R. Ramjee, M. Si-
179 vathanu, N. Kwatra, Z. Han, P. Patel, X. Peng,
180 H. Zhao, Q. Zhang, et al. “Gandiva: Introspec-
181 tive cluster scheduling for deep learning”. In:
182 *13th USENIX Symposium on Operating Sys-
183 tems Design and Implementation (OSDI 18)*.
184 2018, pp. 595–610.

185 [7] A. Raeisi, M. Dolati, S. Darabi, S. Talebi,
186 P. Eugster, and A. Khonsari. “GOGH:
187 Correlation-Guided Orchestration of GPUs in
188 Heterogeneous Clusters”. In: *arXiv preprint*
189 (2025).

190 [8] A. Qiao, S. K. Choe, S. J. Subramanya, W.
191 Neiswanger, Q. Ho, H. Zhang, G. R. Ganger,
192 and E. P. Xing. “Pollux: Co-adaptive cluster
193 scheduling for goodput-optimized deep learn-
194 ing”. In: *15th USENIX Symposium on Oper-
195 ating Systems Design and Implementation
(OSDI 21)*. 2021.

196 [9] B. Wu, Z. Zhang, Z. Bai, X. Liu, and X. Jin.
197 “Transparent GPU sharing in container
198 clouds for deep learning workloads”. In: *20th
199 USENIX Symposium on Networked Systems
200 Design and Implementation (NSDI 23)*. 2023,
201 pp. 69–85.

202 [10] H. Zhao, Z. Han, Z. Yang, Q. Zhang, F. Yang,
203 L. Zhou, M. Yang, F. C. Lau, Y. Wang, Y. Xiong,
204 et al. “HiveD: Sharing a GPU cluster
205 for deep learning with guarantees”. In: *14th
206 USENIX symposium on operating sys-
207 tems design and implementation (OSDI 20)*.
208 2020, pp. 515–532.

209 [11] D. Gu, Y. Zhao, Y. Zhong, Y. Xiong, Z. Han,
210 P. Cheng, F. Yang, G. Huang, X. Jin, and X.
211 Liu. “Elasticflow: An elastic serverless train-
212 ing platform for distributed deep learning”.
213 In: *Proceedings of the 28th ACM Interna-
214 tional Conference on Architectural Support for
215 Programming Languages and Operating Systems,
216 Volume 2*. 2023, pp. 266–280.

217 [12] C. Lin, Z. Han, C. Zhang, Y. Yang, F. Yang,
218 C. Chen, and L. Qiu. “Parrot: Efficient Ser-
219 ving of LLM-based Applications with Semantic
220 Variable”. In: *18th USENIX Symposium on
221 Operating Systems Design and Implemen-
222 tation (OSDI 24)*. 2024, pp. 929–945.