香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
Alibaba Group 阿里巴巴集团

Beware of Fragmentation: Scheduling GPU-Sharing Workloads with Fragmentation Gradient Descent

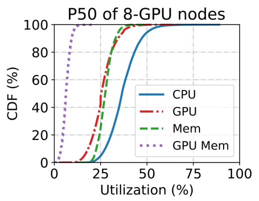
Qizhen Weng†* Lingyun Yang†* Yinghao Yu^† Wei Wang† Xiaochuan Tang^ Guodong Yang^ Liping Zhang^
†: Hong Kong University of Science and Technology ^: Alibaba Group (*: equal contribution)

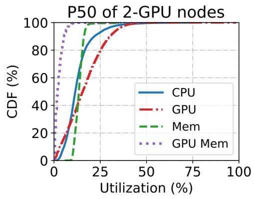
TL;DR: We propose a novel measure of fragmentation to statistically quantify the degree of GPU fragmentation caused by different sources. Based on this measure, we invent a scheduling policy **FGD** that packs tasks to minimize the growth of fragmentation and maximize GPU allocation.

*ML-as-a-Service clouds suffer **low GPU utilization***

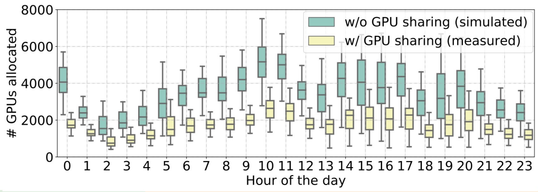
Avg. **25-50% GPU utilization** in production MLaaS clouds [1-3].

[1] Weng et al., "MLaaS in the Wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters," in NSDI 2022.
[2] Hu et al., "Characterization and prediction of deep learning workloads in large-scale GPU datacenters," in SC 2021.
[3] Narayanan et al., "Heterogeneity-aware cluster scheduling policies for deep learning workloads," in OSDI 2020





GPU sharing comes to rescue



GPU sharing lets multiple tasks run on a single GPU, via DL framework manipulation, or CUDA API interception, or hardware-assisted methods (e.g., MIG).

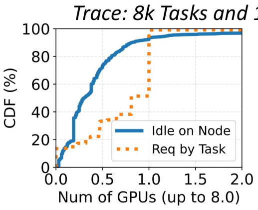
← ***Sharing*** saves **50%** GPUs in Alibaba [1].

*Yet, GPU sharing doesn't always improve allocation. Often, allocating partial GPUs results in **fragmentation***

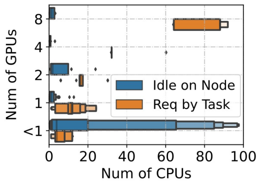
In many clusters, the GPU allocation rate can reach **85-90%** maximum, **leaving hundreds of GPUs unable to allocate!**

Many users experienced **scheduling failures even with sufficient GPU allocation quotas.**

Trace: 8k Tasks and 1.2k nodes with 6.2k GPUs



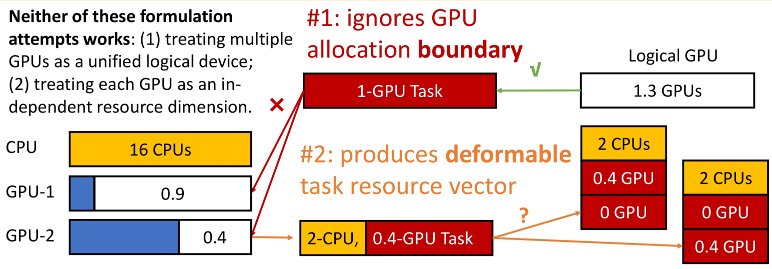
Insufficient GPUs on nodes given the requests of tasks



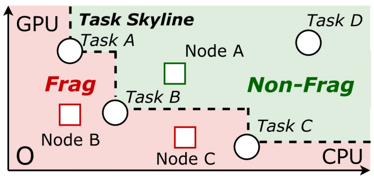
Insufficient CPUs (or stranded GPUs)

*Classical multi-resource bin-packing **cannot** work effectively on GPUs due to **formulation mismatch***

Neither of these formulation attempts works: (1) treating multiple GPUs as a unified logical device; (2) treating each GPU as an independent resource dimension.



*Definition of GPU Fragmentation: The absolute measure is defective. Be **statistical***

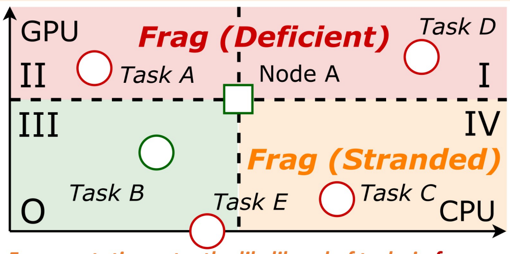


A **defective** definition of **fragmentation** in **absolute** terms — *"a node is fragmented **if and only if** it cannot run any task". **Task skyline*** determines the frag / non-frag boundary, yet, only **0.06%** task instances belong to the skyline ☹

Absolute fragmentation stays low (**<5%**) throughout scheduling simulation (8k tasks to 6.2k GPUs) — ☹ fail to provide useful feedback to the scheduling quality

$$F_n(M) = \sum_{m\in M} p_m F_n(m) \quad (p_m\text{: task popularity})$$

*For each task m in task set M, Sum the fragmentation **viewed by task m***



A **statistical** definition: summed by each task's own view of node fragmentation, weighted by their popularity.

Fragmentation region:
Q-I, Q-II: insufficient GPU
Q-IV: Stranded GPU
X-axis: Non-GPU tasks

Fragmentation rate: the likelihood of tasks in fragmentation regions:
☺ Aware of workload distribution while stable to small changes.
☺ Break down fragmentation into Deficient and Stranded.
☺ Independent of scheduling policy and node distribution.

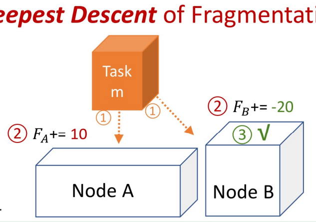
*Schedule Alg.: **Fragmentation Gradient Descent***

Algorithm 1: Node selecting process of FGD

Input : Node set N, incoming task m, workload set M
Output: Assigned node n^*

Initialize node score set $\mathcal{S} \leftarrow \varnothing$, and output $n^* \leftarrow \varnothing$.
① **parallel for** *node* $n \in N$ **do**
  if *Insufficient resources* ∥ *constraints not met* **then**
    Return ▷ Filter out unavailable nodes
  $n^- \leftarrow$ AssignTaskToNode(m, n) ▷ Hypothetically
② $\Delta \leftarrow F^{GPU}_{n^-}(M) - F^{GPU}_{n}(M)$ ▷ Fragmentation increment
  $\mathcal{S} \leftarrow \mathcal{S} \cup (n, \Delta)$
if $\mathcal{S} \neq \varnothing$ **then**
③ $n^* \leftarrow \arg\min_{n\in\mathcal{S}} \Delta$ ▷ pick the node with the least Δ.

Schedule Tasks towards the ***Steepest Descent*** of Fragmentation



Formal Description of Computation $F_n(m)$

- Case 1: All Residuals are Frag. (Q-I, Q-II, Q-IV, **x-axis**):

$$F_n(m) = \sum_{1\le g\le G_n} R^{GPU}_{n,g}$$

Residual resource on GPU g Node n
G_n: GPU set on node n

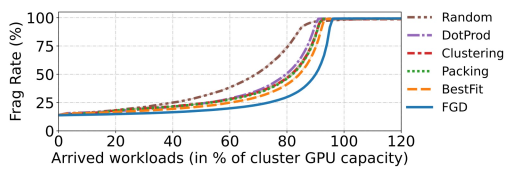
- Case 2: Partial or No Residuals are Frag. (Q-III):

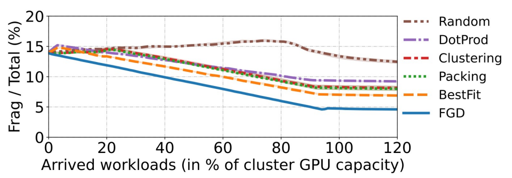
$$F_n(m) = \sum_{1\le g\le G_n} R^{GPU}_{n,g}\,\mathbb{1}\left(R^{GPU}_{n,g} < \min\{D^{GPU}_m, 1\}\right)$$

1, if remaining resource is smaller than the demand of task m, else 0.

Evaluation: Schedule 8k tasks to 6.2k GPUs (1.2k nodes)

FGD: Lowest Frag. Rate & Fewest GPUs Unallocated





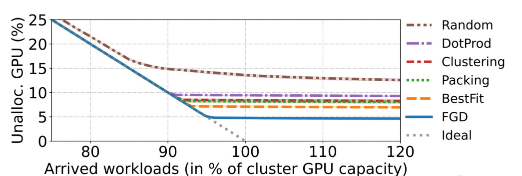
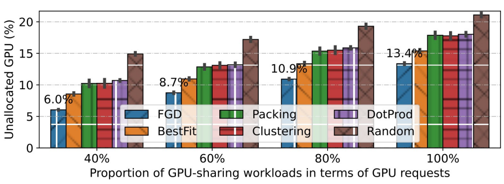


Fig 4a: FGD pursues the lowest fragmentation among various policies in scheduling production workloads, leading to fewest GPUs unallocated.

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED
ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL



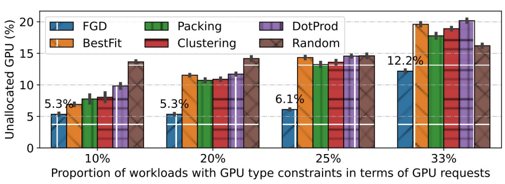


Fig 4b: FGD allocates more GPUs across a variety of settings. See more results and task distributions in paper and code.

Paper

Trace & Code