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].

![Graph showing GPU utilization](Here, you would have an image or graph showing the utilization of GPUs in MLaaS clouds.)

**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 experience scheduling failures even with sufficient GPU allocation quotas.

**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.

**Schedule Alg.: Fragmentation Gradient Descent**

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

  \[ F_n(m) = \frac{\sum_{s \in S} R_{GPU}^{s}}{n} \]

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

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

  \[ F_n(m) = \frac{\sum_{s \in S} R_{GPU}^{s}}{n} \] if remaining resource is smaller than the demand of task m, else 0.

**Formal Description of Computation** $F_n(m)$

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

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

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