Replication and Critical Evaluation of FGD

1 Reconsidering Use of Task Distribution of FGD

FGD explicitly assumes that the task popularity distribution of the entire workload is known in advance and directly leverages this information in scheduling decisions. In contrast, other baseline schedulers are evaluated without access to such prior knowledge. This assumption places FGD in an overwhelmingly favorable position: with prior distribution knowledge, FGD is effectively solving a *simpler problem*, while baselines are solving a *harder one*. As a result, comparing solutions to different problem formulations inherently biases the evaluation in favor of FGD.

2 Expected Limitations of the Assumed Task Distribution Knowledge

Limitation 1: Lack of consideration for distribution shift. In real-world cluster environments, accurately knowing the global task distribution beforehand is impractical. Even if an approximate distribution can be estimated, distribution shift in the workload is inevitable over time due to changes in user behavior, workload dynamics, and system-level factors. Consequently, the static distribution assumed by FGD may quickly become outdated, potentially degrading scheduling performance.

Limitation 2: Non-uniform task arrivals. Tasks are not sampled uniformly from the global distribution over time. In practice, tasks from specific regions of the distribution may arrive in bursts or dominate early phases of execution, which can significantly affect scheduling decisions despite an unchanged global distribution.

3 Experimental Design: Validating the Static-Distribution Assumption Hypothesis

Static-Distribution Assumption Hypothesis FGD's performance gains largely stem from its ability to exploit a *static, globally known task popularity distribution.* When this assumption is weakened due to distribution shift or temporally skewed arrivals, FGD may no longer effectively imitate the optimal packing implied by the true workload distribution, leading to degraded or suboptimal scheduling decisions.

3.1 Experiment 1: Impact of Distribution Shift

Objective Evaluate how sensitive FGD is to inaccuracies in the assumed task popularity distribution.

Setup

- Use the same Alibaba 2023 trace as in the original paper.
- Instead of computing the task popularity distribution from the entire workload:
  - Compute the distribution using only the **first subset of tasks** (e.g., first k% of arrivals).
  - Provide this partial distribution to FGD throughout the execution.

Evaluation

- Compare FGD's performance under this setting against baseline schedulers.
- Measure how performance degrades as the assumed distribution diverges from the true workload distribution.

Rationale This experiment simulates realistic distribution shift and tests whether FGD can still effectively imitate the optimal allocation implied by the true workload.

3.2 Experiment 2: Impact of Non-uniform Task Arrivals

Objective Examine the effect of temporally skewed and bursty task arrivals on FGD's optimization process.

Setup Reorder tasks in the workload to emulate bursty arrival patterns:

- Global ascending sort by task type or resource demand.
- Global descending sort.
- Interval-based ascending sort (within fixed-size segments).
- Interval-based descending sort.

Evaluation Compare fragmentation and scheduling performance across different arrival orderings.

Rationale Although the global distribution remains unchanged, early task arrivals may dominate specific regions of the distribution. This can bias gradient estimation and cause FGD to converge to a suboptimal local optimum.

Algorithm 1 FGD with Time-Window-Based Workload Estimation

```
Require: Cluster $N$, Task arrival stream $\mathcal{T}$, window size $W$
Ensure: Assigned node $n^*$
 1: Initialize task window $\mathcal{W} \leftarrow \emptyset$
 2: Initialize node score set $S \leftarrow \emptyset$, and output $n^* \leftarrow \emptyset$
 3: for all task $m \in \mathcal{T}$ do
 4:     Insert incoming task $m$ into $\mathcal{W}$
 5:     if $|\mathcal{W}| > W$ then
 6:         Remove the oldest task from $\mathcal{W}$ (FIFO)
 7:     end if
 8:     Compute task distribution of target workload $M_{\mathcal{W}}$ from tasks in $\mathcal{W}$
 9:     for all node $n \in N$ in parallel do
10:         if insufficient resources or constraints not met then
11:             return                                   ▷ filter out unavailable nodes
12:         end if
13:         $n^- \leftarrow$ AssignTaskToNode$(m, n)$            ▷ hypothetical assignment
14:         $\Delta \leftarrow F_n(M_{\mathcal{W}}) - F_n(M_{\mathcal{W}})$              ▷ fragmentation increment
15:         $S \leftarrow S \cup \{(n, \Delta)\}$
16:     end for
17:     if $S \neq \emptyset$ then
18:         $n^* \leftarrow \arg\min_{n \in S} \Delta$            ▷ pick node with least fragmentation
19:     end if
20: end for
```

4 Proposed Improvements: Time-Window-Based Online Distribution Estimation

Time-window-based online distribution estimation. We propose replacing the global, static task distribution used by FGD with a dynamic distribution estimated from a sliding window of the most recent N tasks. Each newly arriving task is inserted into the window, and once the window reaches its capacity, older tasks are evicted in a FIFO manner to keep the window size fixed.

This approach ensures that gradient updates are driven by recent workload behavior rather than the entire execution trace. Importantly, it preserves the core optimization structure of FGD while improving robustness to distribution shift, temporal skew, and practical online deployment constraints.