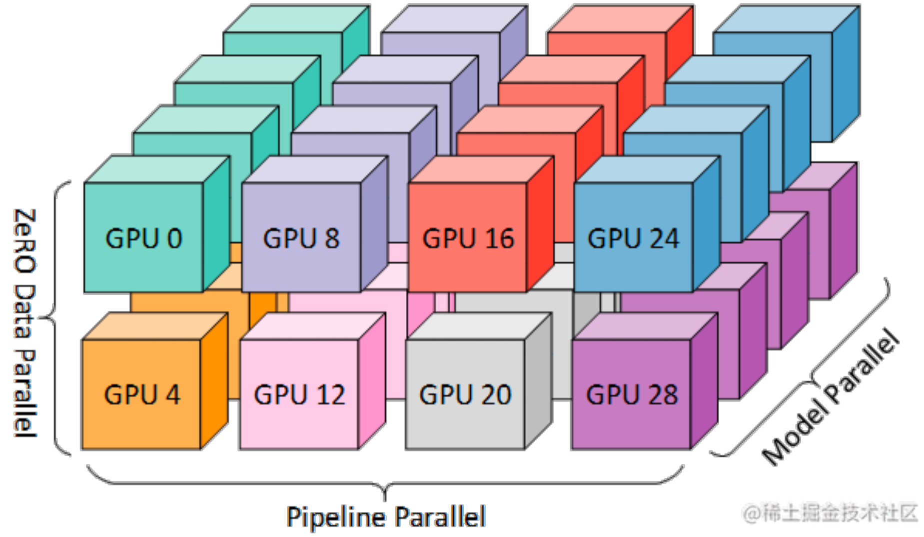
Non-Idle Machine-Aware Worker Placement for Efficient Distributed Training in GPU Clusters

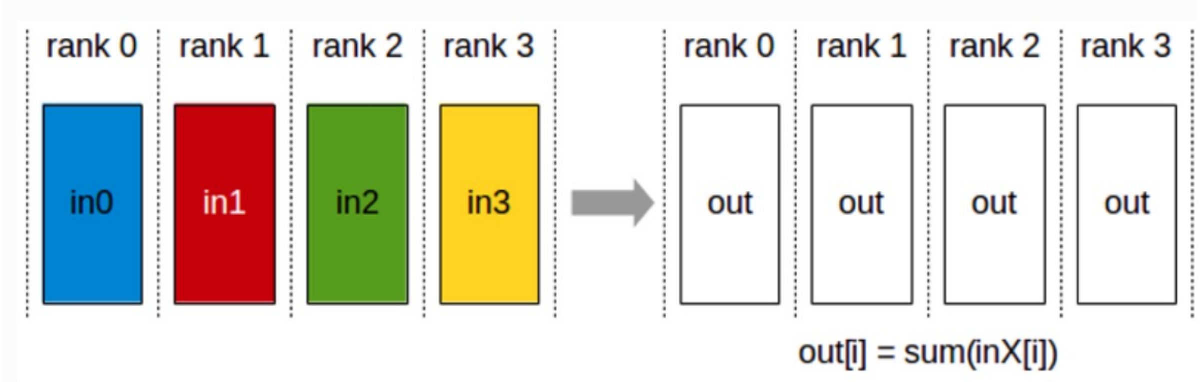
Jin Fang

Gongming Zhao, Hongli Xu, Luyao Luo, Zhen Yao, An Xie

Collective Communication in DT

- With the increasing complexity of machine learning (ML) applications, the scale of ML tasks grows explosively
- **Distributed training** is proposed to speed up the training of large-scale ML tasks
- Placing workers on GPUs of machines to perform one DT task, where workers communicates collectively to synchronize gradients/parameters

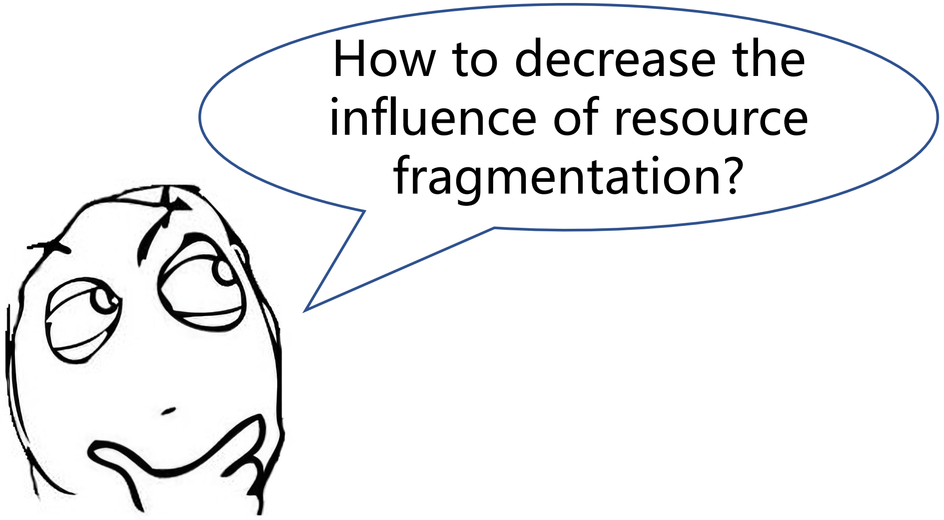




AllReduce

Problem: Resource Fragmentation

- One machine is equipped with multiple GPUs (e.g., 8 GPUs)
- The number of workers of a DT job varies
  - DP = 2 -> 2 workers
  - TP = 8 -> 8 workers
  - DP = 2 and TP =8 -> 2*8=16 workers
- The arrival time of DT jobs is unpredictable
- There exists a lot of fragmented idle GPUs, leading to low resource utilization



Researchers

Existing Solution

Consolidation-First Placement

- Elasticflow[1] (ASPLOS 23): allocate the resource at the scale of machine

Resource oversubscription

Fragmentation-First Placement

- HiveD[2] (OSDI 20): priority place placing workers in fragmented machines

Large comm. overhead

1. Gu D, Zhao Y, Zhong Y, et al. ElasticFlow: An elastic serverless training platform for distributed deep learning[C]//Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. 2023: 266-280.
2. Zhao H, Han Z, Yang Z, et al. {HiveD}: Sharing a {GPU} cluster for deep learning with guarantees[C]//14th USENIX symposium on operating systems design and implementation (OSDI 20). 2020: 515-532.

A Motivating Example

- Collective communication operations take multiple steps, where each step contains different communication pairs and amounts

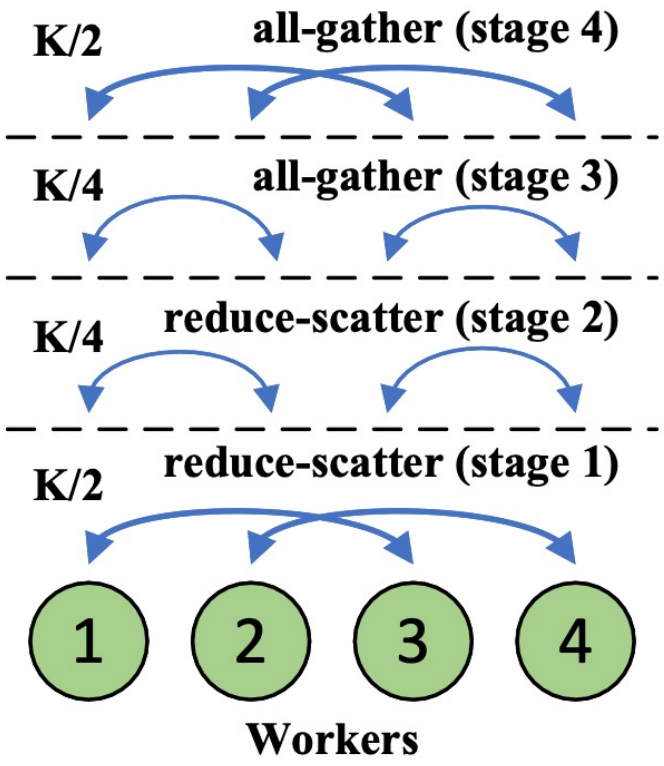
Workers in one DT job have different communication pattern.

Can we utilize this feature?

Researchers

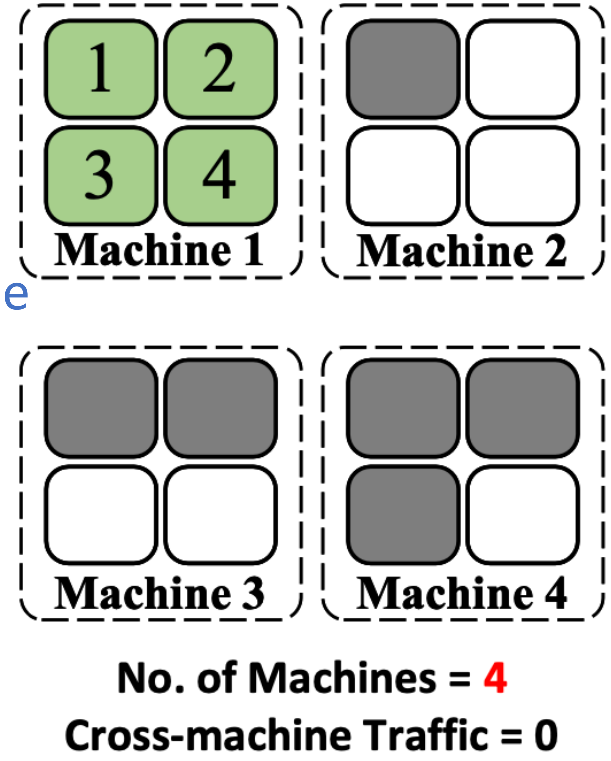
A Motivating Example-Elasticflow

- Consolidation placement to minimize the amount of cross-machine traffic



(a) Communication Pattern Example

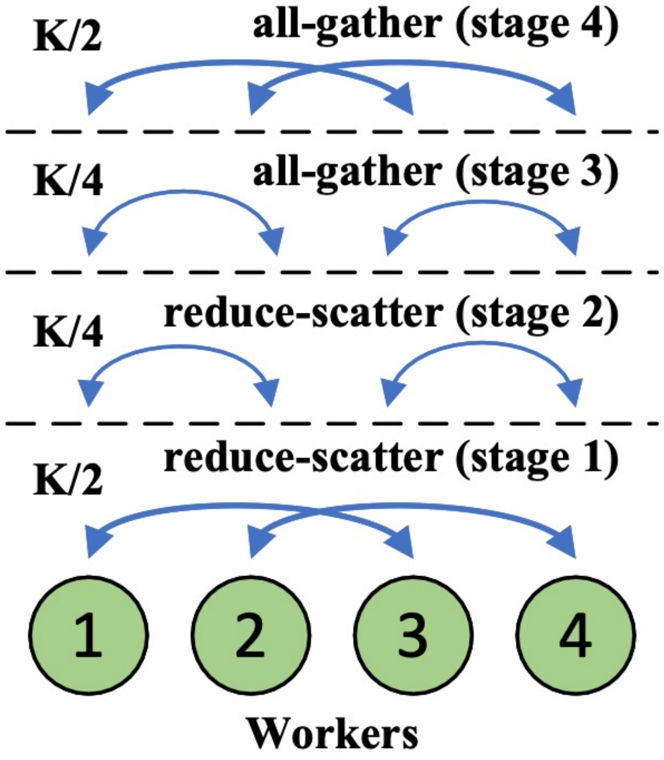
Launch an idle machine



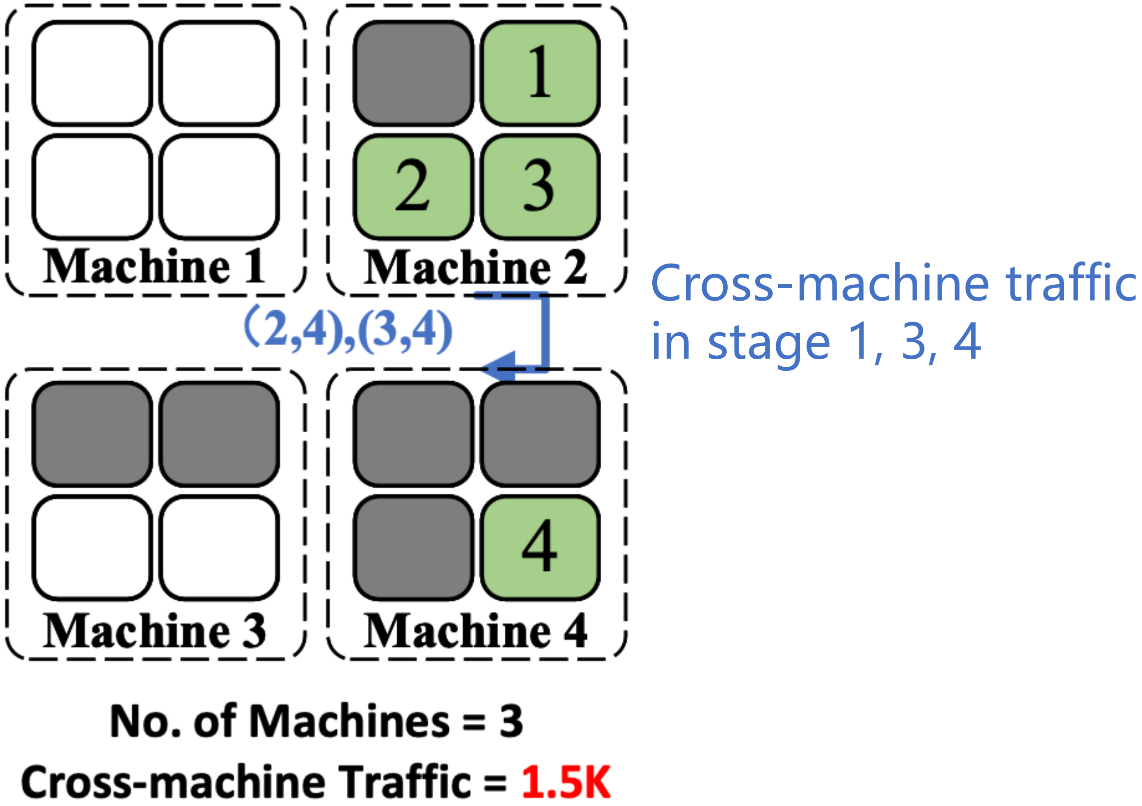
(b) Elasticflow

A Motivating Example-HiveD

- Using fragmented machines to minimize the number of used machines



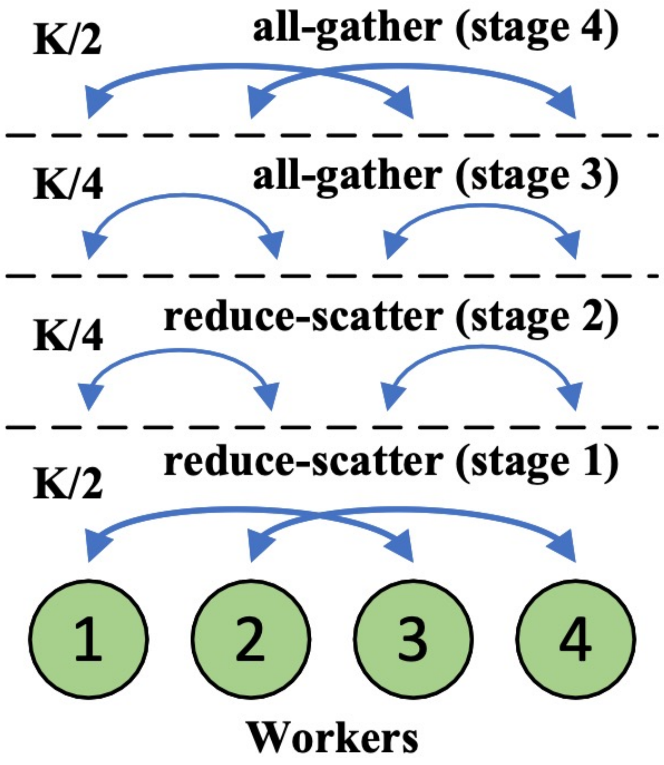
(a) Communication Pattern Example



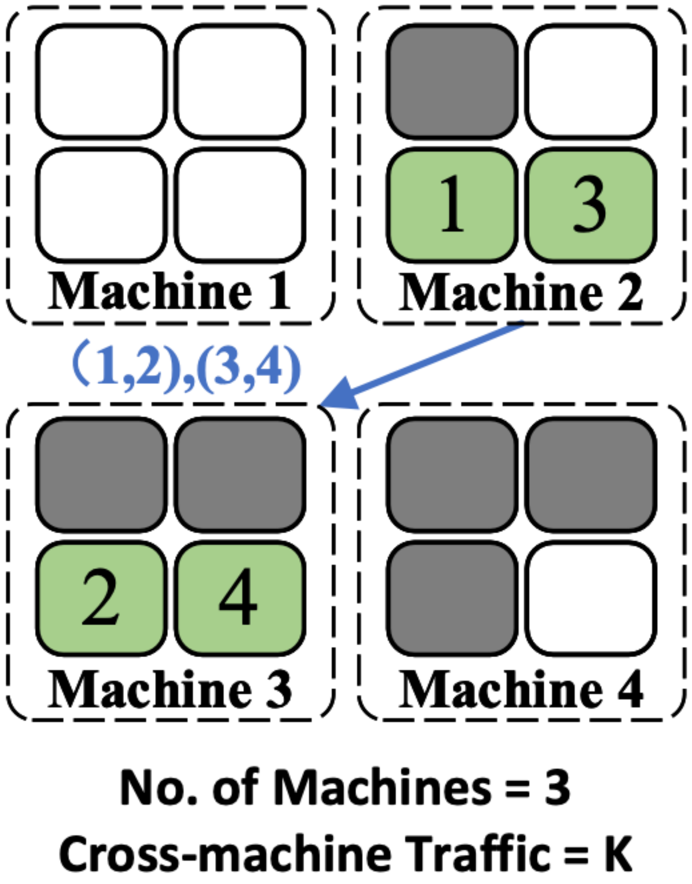
(c) HiveD

A Motivating Example-Titan (Ours)

- Consider the map of workers to GPUs according to collective communication algorithms
- Avoid the use of idle machines



(a) Communication Pattern Example



(d) Titan (Ours)

Titan: Problem Formulation

Network Model

- Machine set: $S = \{s_1, s_2, \dots, s_{|S|}\}$, each with K GPUs
- Non-Idle machine set: $S_f \subset S$
- Idle machine set: $S_n \subset S$
- Available bandwidth between machines s and s': $\mathrm{P}_{s,s'} \in \mathbb{Z}$

Communication Pattern

- Worker set: $N = \{n_1, n_2, \dots, n_{|N|}\}$
- The traffic amount between worker pair (n, n') in phase t : $C^t_{n,n'} \in \mathbb{Z}$

Titan: Problem Formulation

- Objective
  1. **Minimize the number of Idle-machines**: reduce resource fragmentation
  2. **Minimize the number of total machines**: reduce the amount of cross-machine traffic

$$\min\ O_1 = \sum_{s \in S_n} y_s$$

$$\min\ O_2 = \sum_{s \in S} y_s$$

- Placement constraint
- Resource constraint
- Bandwidth constraint

$$S.t. \begin{cases} \sum_{s \in S} x_n^s = 1, & \forall n \in N \\ \sum_{n \in N} x_n^s \le R_s, & \forall s \in S \\ \sum_{n \in N} \sum_{n' \in N} x_n^s \cdot x_{n'}^{s'} \cdot C_{n,n'}^t \le P_{s,s'}, & \forall s, s' \in S, t \in T \\ y_s \ge x_n^s, & \forall s \in S, n \in N \\ x_n^s \in \{0,1\}, & \forall s \in S, n \in N \\ y_s \in \{0,1\}, & \forall s \in S \end{cases} \quad (1)$$

Titan: Algorithm Design

- Convert the problem into an equivalent maximization problem
  - So we can construct the submodular function for the greedy algorithm

$$\min \ O_1 = \sum_{s \in S_n} y_s$$

$$\min \ O_2 = \sum_{s \in S} y_s$$

$$S.t. \begin{cases} \sum_{s \in S} x_n^s = 1, & \forall n \in N \\ \sum_{n \in N} x_n^s \le R_s, & \forall s \in S \\ \sum_{n \in N} \sum_{n' \in N} x_n^s \cdot x_{n'}^{s'} \cdot C_{n,n'}^t \le P_{s,s'}, & \forall s, s' \in S, t \in T \\ y_s \ge x_n^s, & \forall s \in S, n \in N \\ x_n^s \in \{0,1\}, & \forall s \in S, n \in N \\ y_s \in \{0,1\}, & \forall s \in S \end{cases} \tag{1}$$

$$\max \ O_1 = C_{\max} - \sum_{s \in S_n} C(\mathcal{N}_s)$$

$$\max \ O_2 = C_{\max} - \sum_{s \in S} C(\mathcal{N}_s)$$

$$S.t. \begin{cases} \sum_{s \in S} x_n^s = 1, & \forall n \in N \\ \sum_{n \in N} x_n^s \le R_s, & \forall s \in S \\ \sum_{n \in N} \sum_{n' \in N} x_n^s \cdot x_{n'}^{s'} \cdot C_{n,n'}^t \le P_{s,s'}, & \forall s, s' \in S, t \in T \\ x_n^s \in \{0,1\}, & \forall s \in S, n \in N \end{cases} \tag{3}$$

Titan: Algorithm Design

- Solve the converted problem with a **submodular-based greedy algorithm**
  - **Search the feasible worker set** for each machine to guarantee bandwidth constraint
  - **Merge the worker set with submodular function** to minimize the number of used idle machines
  - **Update and merge the worker set** to minimize the number of used machines

Algorithm 1 Search for Feasible Worker Set

Step 1: Initialization
Let feasible worker set $A(s) = \emptyset$ for machine s.
Let available worker set $N_c = N - \mathcal{N}$.
Step 2: Iterative update feasible worker sets according to collective communication
Use $\mathcal{C}_{\int,\int'}$ to denote the existing communication overhead between machines s and s'.
for $n \in N_c$ **do**
  for $n' \in \mathcal{N}$ **do**
    for $t \in T$ **do**
      if $C^t_{n,n'} + \mathcal{C}_{\int,\int'} \le P_{s,s'}$ **then**
        $A(s) \leftarrow A(s) + n$
      end if
    end for
  end for
end for
Output the feasible worker set $A(s)$ for machine s.

Algorithm 3 The Overall Algorithm

Step 1: Minimizing the Number of Deployed New Racks
Initiate $\mathcal{N}_s, \forall s \in S_n$ by randomly distributing workers.
Initiate $\Phi_n \leftarrow \emptyset$.
Calculate Φ_n on idle machine set S_n with Alg. 2.
Step 2: Minimizing the Total Number of Deployed Racks
Initiate $\mathcal{N}_s, \forall s \in S_f \cup \Phi_n$ by randomly distributing workers.
Initiate $\Phi \leftarrow \emptyset$.
Calculate Φ on machine set $S_f \cup \Phi_n$ with Alg. 2.
Step 3: Determining the Deployment of Workers
for $\mathcal{N}_s \in \Phi$ **do**
  Set $x^s_n = 1, \forall n \in \mathcal{N}_s, s \in S$.
end for

Titan: Algorithm Design

- ➢ Solve the converted problem with a **submodular-based greedy algorithm**

  - ➢ **Search the feasible worker set** for each machine to guarantee bandwidth constraint
  - ➢ **Merge the worker set with submodular function** to minimize the number of used idle machines
  - ➢ **Update and merge the worker set** to minimize the number of used machines
  - ➢ Tight approximate ratio (1-1/e)

Algorithm 2 Submodular-based Algorithm

Step 1: Initialization
Initiate $\mathcal{N}_s, \forall s \in S$ by randomly distributing workers.
Initiate $\Phi \leftarrow \emptyset$.
Step 2: Iterative Merging Worker Subsets
while $|\Phi| \leq K - 1$ **do**
  Set $tmp \leftarrow 0, opt \leftarrow 0$
  for $s \in S_n$ **do**
    for $n \in \mathcal{N}_s - \Phi$ **do**
      $tmp \leftarrow H(\Phi \cup \{n\})$
      if $tmp > opt$ **then**
        $opt \leftarrow tmp$, $\mathcal{N}^* \leftarrow \mathcal{N}^* + \{n\}$
      end if
    end for
  end for
  $\Phi \leftarrow \Phi + \mathcal{N}^*$
  Update the feasible worker sets based on the bandwidth constraint of Eq. (5) with Alg. 1.
end while
Deploy the remaining workers on one machine (*i.e.*, $\Phi \leftarrow \Phi + \{N - \bigcup_{\mathcal{N} \in \Phi} \mathcal{N}\}$).

$$\sum_{n \in \mathcal{N}_s} \sum_{n' \in \mathcal{N}_{s'}} C^t_{n,n'} \leq P_{s,s'}, \forall s' \in S, t \in T \quad (5)$$

Evaluation: Setup

Simulation Topology

- Fat-tree topology with 64 racks, each of which contains 8 machines
- Each machine is equipped with 8 GPUs
- All connected with 100Gbps links

Real-world Traces

- Microsoft cluster: 2-month trace with 69742 jobs
- Shanghai AI lab cluster: 6-month trace with 880740 jobs

Evaluation: Setup

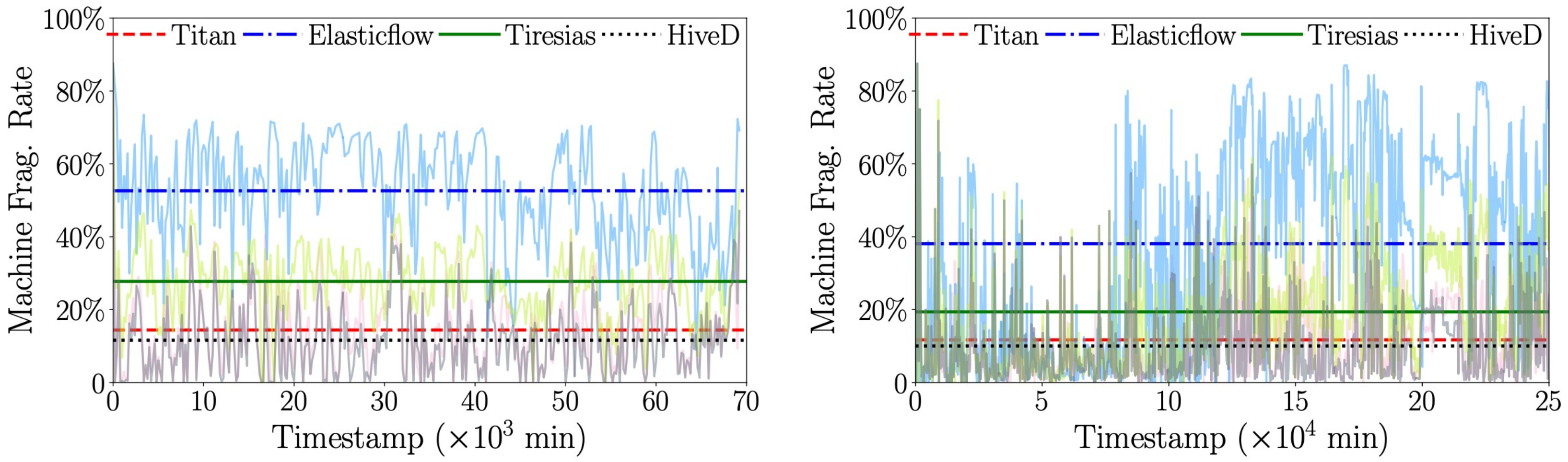
Benchmark

- **Elasticflow**[1] : consolidate the placed GPUs so that the job is allocated with the highest possible bandwidth between its workers
- **HiveD**[2] : prioritize placing workers in fragmented machines to reduce the machine fragmentation of the cluster
- **Tiresias**[3] : minimize the total network traffic and balance the network load across machines in the cluster, by profiling the characteristics of different models

3. Gu J, Chowdhury M, Shin K G, et al. Tiresias: A {GPU} cluster manager for distributed deep learning[C]//16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). 2019: 485-500.

Evaluation: Fragmentation Rate

➢ Record the ratio of non-idle machines in the cluster, after a job arrives



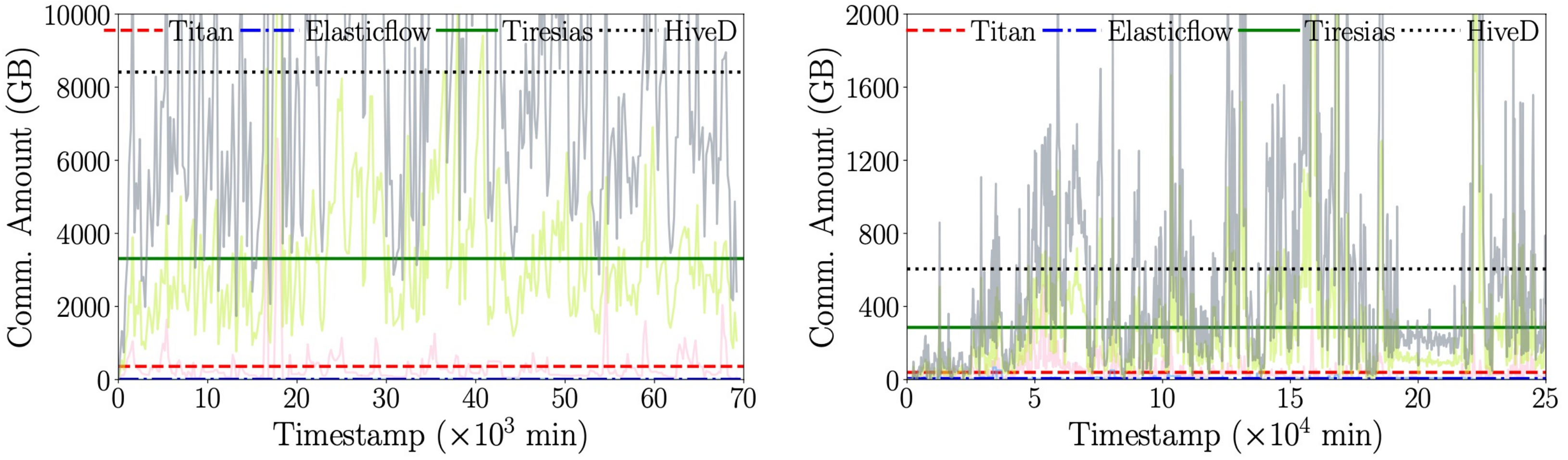
(a) Microsoft Trace (b) Shanghai AI Lab Trace

Fig. 4: Machine Fragmentation Rate vs. Timestamp

➢ Titan reduces machine fragmentation rate by **38.1%**

Evaluation: Comm. Overhead

- Use the HD algorithm and calculate the total communication overhead



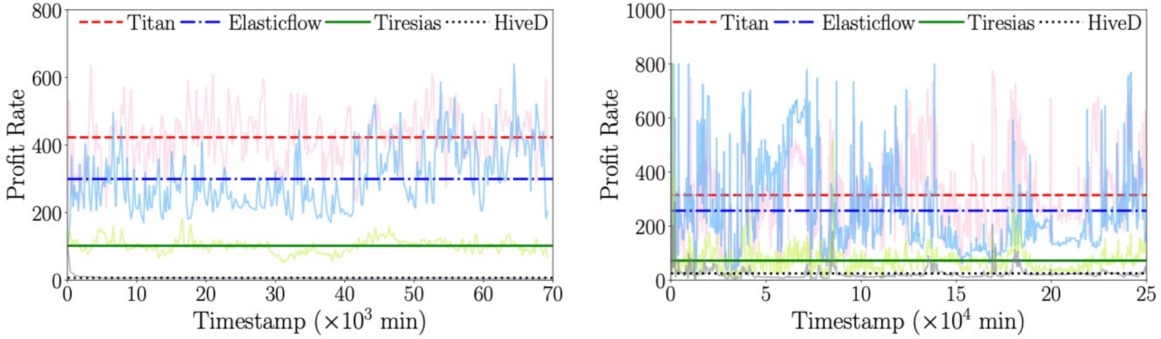
(a) Microsoft Trace (b) Shanghai AI Lab Trace

Fig. 5: Communication Overhead vs. Timestamp

- Titan reduces communication overhead of the cluster by **76.4%**

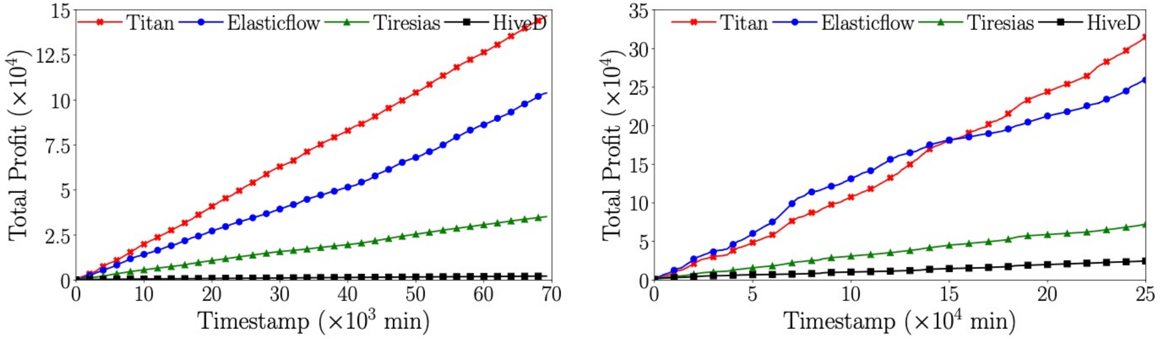
Evaluation: Profit

➢ Profit relates to No. of jobs, price of the job, No. of used machines and the duration of each job (See the manuscript for details)



(a) Microsoft Trace (b) Shanghai AI Lab Trace

Fig. 6: Profit Rate vs. Timestamp



(a) Microsoft Trace (b) Shanghai AI Lab Trace

Fig. 7: Total Profit vs. Timestamp

➢ Titan improves the total profit by **41.2%~65X**

Summary

Goal

- Minimize the resource fragmentation rate of GPU cluster with non-idle machine-aware worker placement

Challenges

- Collective Communication constraint
- Resource fragmentation limitation
- Model a multi-subjective non-linear problem

Solution

- Submodular-based greedy algorithm with a tight approximation ratio

Thank you!

IEEE International Conference on Network Protocols (ICNP 2024)

Committee, Reviewers, Volunteers

My Advisors and Collaborators!

Jin Fang

fangjin98@mail.ustc.edu.cn

www.fangjin.site