Heterogeneity-Aware Cluster Scheduling Policies for Deep Learning Workloads

Deepak Narayanan§, Keshav Santhanam§, Fiodar Kazhamiaka§, Amar Phanishayee★, Matei Zaharia§

★ Microsoft Research  § Stanford University
Hardware for ML training is becoming highly specialized and heterogeneous!

Nvidia GPUs: K80, P100, V100, A100
Google TPU
FPGAs in Azure

...and others
How should we allocate heterogeneous resources?

Objective (e.g., fairness)

Scheduler

PyTorch
TensorFlow

Training jobs written in existing frameworks

Heterogeneous cluster

V100 GPU
P100 GPU

How should one allocate **heterogeneous resources** to DL training jobs from multiple users while optimizing **different objectives**?
Challenge 1: Heterogeneous performance

- Models and operators (e.g., convolution, attention) perform differently across hardware architectures
- Disregarding heterogeneity can lead to unfair allocations

Magnitude of speedup across GPU generations varies significantly
Challenge 2: Diverse scheduling objectives

- Single-job objectives: “maximize throughput” or “minimize cost”
  - Minimizing cost subject to SLOs involves moving between fast but expensive, and slow but cheap instances

- Multi-job objectives: fairness or more complicated hierarchical policies

Hierarchical policy: Weighted fairness across sub-organizations, FIFO and fairness within
Related work

• Most existing cluster schedulers for deep learning (e.g., Gandiva [1], Themis [2], Tiresias [3]) disregard heterogeneity

• AlloX [4] and Gandiva_fair [5] do consider performance heterogeneity, but tightly couple their target objective to scheduling mechanism
  • Average JCT for AlloX, max-min fairness for Gandiva_fair

[1] Gandiva: Introspective Cluster Scheduling for Deep Learning, OSDI 2019, Xiao et al.
[2] Themis: Fair and Efficient GPU Cluster Scheduling, NSDI 2020, Mahajan et al.
[3] Tiresias: A GPU Cluster Manager for Distributed Deep Learning, NSDI 2019, Gu et al.
[4] AlloX: Compute Allocation in Hybrid Clusters, EuroSys 2020, Le et al.
[5] Balancing Efficiency and Fairness in Heterogeneous GPU Clusters for Deep Learning, EuroSys 2020, Chaudhary et al.
Gavel: A new heterogeneity-aware cluster scheduler

- Generalizes a wide range of existing scheduling policies by expressing policies as optimization problems over the allocation
- Provides abstraction to incorporate performance heterogeneity
- Round-based scheduling mechanism ensures jobs receive optimal allocation
- Improves objectives such as average job completion time by 3.5×

This talk!
Outline

• Background and Motivation

• Challenges with allocating resources over heterogeneous resources

• Heterogeneity-aware Policies

• Round-based Scheduling Mechanism

• Evaluation
Scheduling policies to be made heterogeneity-aware

- **FIFO**: First in, first out
- **Shortest Job First**: Minimize time taken by shortest job
- **Minimize Makespan**: Minimize time taken by batch of jobs
- **Minimize cost (w/ SLOs)**: Minimize total cost in public cloud (subject to SLOs)
- **LAS [1]**: Max-min fairness by total compute time
- **LAS w/ weights**: Max-min fairness by total compute time with weights
- **Finish Time Fairness [2]**: Maximize minimum job speedup
- **Hierarchical**: Multi-level policy with fairness as top-level policy, and FIFO or fairness as lower-level policies. Per-job weights can be specified

[1] Tiresias: A GPU Cluster Manager for Distributed Deep Learning, NSDI 2019, Gu et al.
[2] Themis: Fair and Efficient GPU Cluster Scheduling, NSDI 2020, Mahajan et al.
Policies as optimization problems

- In a homogeneous cluster, policy objectives are functions of throughput (e.g., duration = training steps / throughput) and allocation.
- On a homogeneous cluster, **Least Attained Service** policy is a max-min fairness policy that equalizes the total compute time each job receives.
- Jobs can see unequal throughput reductions on heterogeneous clusters.
Allocations ($X$) as time fractions

$X$ specifies the fraction of time a job spends on each accelerator between allocation recomputations.

$$X_{\text{example}} = \begin{pmatrix} V100 & P100 & K80 \\ 0.6 & 0.4 & 0.0 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.0 & 0.8 \end{pmatrix}$$

Allocations recomputed either at periodic intervals of time, or on a reset event (new job arrives, or old job completes).
Effective throughput

To make policies heterogeneity-aware, policy objectives can be expressed in terms of **effective throughput** (given allocation $X$ and throughputs $T$):

$$\text{throughput}(\text{job } m, X) = \sum_{\text{accelerator type } j} T_{mj} \cdot X_{mj}$$

$T$ is matrix of raw throughputs of each job on each accelerator type

$$T = \begin{pmatrix} V100 & K80 \\ 40.0 & 10.0 \\ 12.0 & 4.0 \\ 100.0 & 50.0 \end{pmatrix}$$

job 0  
job 1  
job 2
Policies as optimization problems

• In a homogeneous cluster, policy objectives are functions of throughput (e.g., duration = training steps / throughput)

• On a homogeneous cluster, **Least Attained Service** policy is a max-min fairness policy that equalizes the total compute time each job receives

\[
\text{Maximize}_x \min_m X_m
\]

• Jobs can see unequal throughput reductions on heterogeneous clusters

• Instead, compute max-min fairness over effective throughputs:

\[
\text{Maximize}_x \min_m \frac{\text{throughput}(m, X)}{\text{normalizing\_factor}_m}
\]
Scheduling policies to be made heterogeneity-aware

- **FIFO**: First in, first out
- **Shortest Job First**: Minimize time taken by shortest job
- **Minimize Makespan**: Minimize time taken by batch of jobs
- **Minimize cost (w/ SLOs)**: Minimize total cost in public cloud (subject to SLOs)
- **LAS**: Max-min fairness by total compute time
- **LAS w/ weights**: Max-min fairness by total compute time with weights
- **Finish Time Fairness**: Maximize minimum job speedup
- **Hierarchical**: Multi-level policy with fairness as top-level policy, and FIFO or fairness as lower-level policies. Per-job weights can be specified

See paper for details!
Performance optimizations: space sharing and placement

• Gavel can also deploy existing performance optimizations like space-sharing and placement awareness [1, 2] in a heterogeneity-aware way

• Objectives in terms of throughput($m, X$) unchanged

• $X$ needs to be modified to account for performance optimization (e.g., allocation for each job combination)

• Raw throughputs ($T$) for concurrently running applications might need to be measured / estimated on the fly (see paper for details)

[1] Gandiva: Introspective Cluster Scheduling for Deep Learning, OSDI 2018, Xiao et al.
[2] Themis: Fair and Efficient GPU Cluster Scheduling, NSDI 2020, Mahajan et al.
Outline

• Background and Motivation
• Challenges with allocating resources over heterogeneous resources
• Heterogeneity-aware Policies
• Round-based Scheduling Mechanism
• Evaluation
How do we realize an optimal allocation?

Given an optimal heterogeneity-aware allocation by a policy, how do we assign resources to jobs?

Assignments of jobs to heterogeneous cluster resources
Gavel’s round-based scheduling

- Round-based scheduler ensures jobs receive time on accelerator types according to the computed optimal allocation $X$

$$X_{\text{het.+SS}} = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.5 \\ 0.0 & 0.5 & 0.5 \end{pmatrix}$$

V100 | P100 | K80
---|---|---
1 | 1 | 1 | 1 | 1
3 | 2 | 3 | 2
2 | 3 | 2 | 3

Scheduling rounds
Gavel’s round-based scheduling

- Round-based scheduler ensures jobs receive time on accelerator types according to the computed optimal allocation $X$

- Priority score for every (job, accelerator) combination
  - $\text{priorities} = \frac{X^{\text{target}}}{\text{rounds\_received}}$ (element-wise division of matrices)

$$X_{\text{example}} = \begin{pmatrix}
0.6 & 0.4 & 0.0 \\
0.2 & 0.6 & 0.2 \\
0.2 & 0.0 & 0.8 \\
\end{pmatrix}$$

- Jobs placed on resources where they have high priority (marked in red)
Outline

• Background and Motivation
• Challenges with allocating resources over heterogeneous resources
• Heterogeneity-aware Policies
• Round-based Scheduling Mechanism
• Evaluation
Main questions

• Do Gavel’s policies improve objective metrics in a heterogeneous cluster?
• What is the impact of input load on objectives using Gavel’s policies?
• Can Gavel’s policy framework support hierarchical policies?
• How do Gavel’s policies scale with the number of active jobs?
Gavel improves objectives on a heterogeneous cluster

Physical cluster with 8 V100 GPUs, 16 P100 GPUs, 24 K80 GPUs

| System                                      | Policy                     | Physical   | Simulated |
|---------------------------------------------|----------------------------|------------|-----------|
| Heterogeneity-agnostic                      | Least Attained Service     | 5.1 hrs    | 5.4 hrs   |
| Heterogeneity-aware                         | (average JCT)              | 3.4 hrs    | 3.7 hrs   |
| Heterogeneity-agnostic (w/ ad hoc space sharing) | Makespan                  | 21.3 hrs   | 22.1 hrs  |
| Heterogeneity-aware                         |                            | 17.7 hrs   | 17.6 hrs  |

- Gavel reduces average JCT by 1.5x
- Gavel without space sharing reduces makespan by 1.2x compared to a baseline that uses ad-hoc space sharing
- Results in simulation reflect reality (< 8% difference)
Gavel can enable the same heterogeneous cluster to support higher input load

- **Simulated cluster** with 36 V100 GPUs, 36 P100 GPUs, 36 K80 GPUs
- Each policy evaluated on multiple traces (different Poisson arrival rates)
Gavel can support hierarchical policies

Weighted fairness at both levels

- Six jobs per entity
- \( w_{\text{entity } 0} < w_{\text{entity } 1} < w_{\text{entity } 2} \)
- \( w_{\text{entity } 1} = 2 \) implies that entity 1 should get \( 2 \times \) resources as entity 0
Gavel can support hierarchical policies

Widths of bars indicate that inter- and intra-entity weights are respected

Allocation in ratio of 3:2:1
Gavel scales to clusters with hundreds of active jobs

Gavel can compute heterogeneity-aware allocations over 2048 jobs in a minute
Main questions

• Do Gavel’s policies improve objective metrics in a heterogeneous cluster?
• What is the impact of input load on objectives using Gavel’s policies?
• Can Gavel’s policy framework support hierarchical policies?
• How do Gavel’s policies scale with the number of active jobs?
• How well does Gavel’s scheduling mechanism realize optimal allocations?
• What is the overhead of preemption in Gavel?

More results (including more objectives) in paper!
Conclusion

- Gavel is a heterogeneity-aware cluster scheduler able to optimize for many high-level objectives such as fairness, makespan, and cost.
- Gavel formulates existing policies as optimization problems, and extends these optimization problems to be heterogeneity-aware.
- Gavel can reduce average job completion time by $3.5 \times$

Code open sourced at https://github.com/stanford-futuredata/gavel

https://cs.stanford.edu/~deepakn/  deepakn@stanford.edu