Balancing Efficiency and Fairness in Heterogeneous GPU Clusters for Deep Learning

Shubham Chaudhary | Ramachandran Ramjee | Muthian Sivathanu

Nipun Kwatra | Srinidhi Viswanatha

Microsoft Research India

Scheduling of Deep Learning Workloads

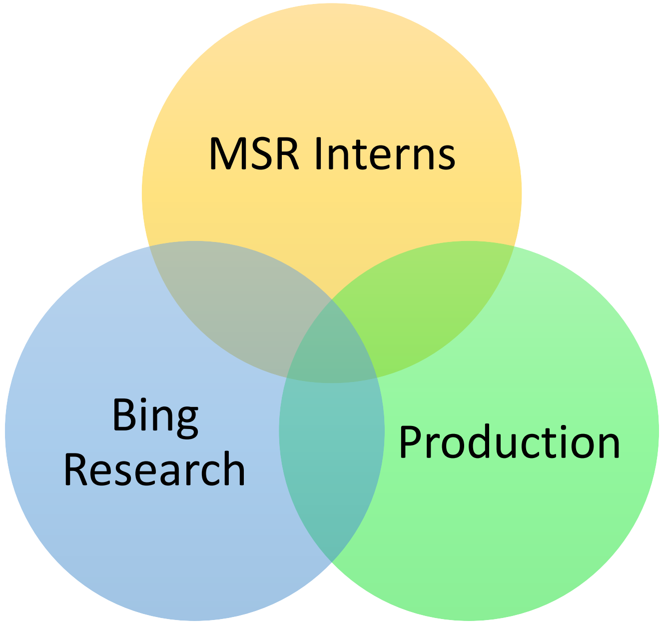
Scheduler	Exclusive GPU	Execution Model	Optimizes For	Fairness	Heterogeneity
FfDL[1]	✓	Generic	Scalability	✗	✗
Philly[2]	✓	Generic	Consolidation	**Static Partitioning + Preemption**	✗
Optimus[3]	✗	Parameter Server	Average JCT*	✗	✗
Tiresias[4]	✗	Parameter Server	Average JCT*	✗	✗
Gandiva[5]	✗	Generic	Utilization	✗	✗

[1] Boag, Scott, et al. "Scalable multi-framework multi-tenant lifecycle management of deep learning training jobs." Workshop on ML Systems, NIPS. 2017.
[2] Jeon, Myeongjae, et al. "Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads." 2019 (USENIX) Annual Technical Conference (USENIX ATC 19). 2019.
[3] Peng, Yanghua, et al. "Optimus: an efficient dynamic resource scheduler for deep learning clusters." Proceedings of the Thirteenth EuroSys Conference. 2018.
[4] Gu, Juncheng, et al. "Tiresias: A GPU cluster manager for distributed deep learning." 16th (USENIX) Symposium on Networked Systems Design and Implementation (NSDI 19). 2019.
[5] Xiao, Wencong, et al. "Gandiva: Introspective cluster scheduling for deep learning." 13th (USENIX) Symposium on Operating Systems Design and Implementation (OSDI 18). 2018.

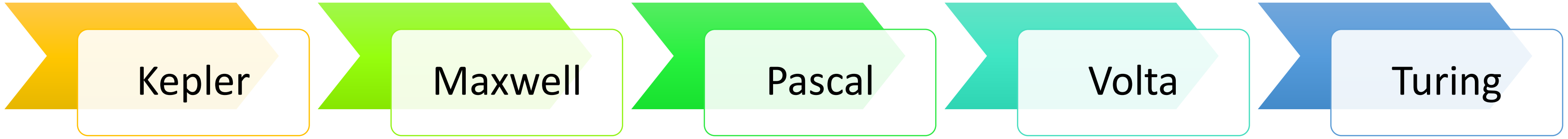
* Job Completion Time

Performance Isolation and Fair Share

- **How to share a large cluster among many different groups?**
- **Simple:** Perform static partitioning of a physical cluster into virtual clusters.
  - Makes sharing of underutilised resources hard.
- **Idea:** Provide performance isolation through proportional allocation of resources.



Heterogeneity



- New GPUs released each year.
- Separate physical clusters for each generation, users choose which cluster to submit to.
- Everyone wants newer GPUs, therefore older GPUs left underutilized.
- **How to choose the best GPU automatically?**

Contributions

Gandiva$_{\text{fair}}$ is the first Deep Learning Scheduler that does

- Efficient fair-sharing of cluster-wide GPU throughput.
- Transparent handling of resource heterogeneity.
- Migration to provide the above without preemption.

One cluster scheduler to rule them all.

System Model

- Users are assigned tickets and GPU throughput is allocated proportionally among all active users.
- Tickets are divided equally among all jobs of the same user.
- Jobs can be of varying sizes, GPUs should be gang-scheduled.
- We use the time-slicing and migration primitives implemented in Gandiva[5].

Split-Stride Scheduler

Stride Scheduling

Time	A’s pass	B’s pass	Schedule
0	0	0	B
1	0	1	A
2	0.25	1	A
3	0.5	1	A
4	0.75	1	A
5	1	1	B
6	1	2	A
7	1.25	2	A
8	1.5	2	A

Job	Tickets
A	4
B	1

```
/* called every time-quantum. */
def schedule:
    job = min(q, λj: j.pass)
    job.pass += 1 / job.tickets
    return {job}
```

Split-Stride Scheduler

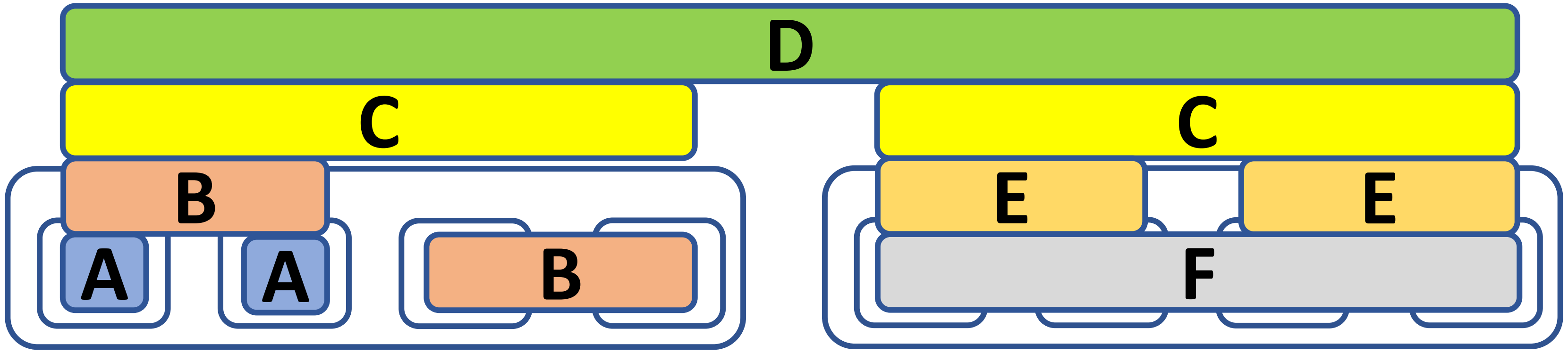
Gang-Aware Stride Scheduling

Job	Tickets	GPUs
A	1	1
B	1	1
C	1	2
D	1	2
E	1	4

Time	A	B	C	D	E	Schedule
0	0	0	0	0	0	E
1	0	0	0	0	4	A, B, C
2	1	1	2	0	4	A, B, D
3	2	2	2	2	4	A, B, C
4	3	3	4	2	4	A, B, D
5	4	4	4	4	4	E
6	4	4	4	4	8	A, B, C
7	5	5	6	4	8	A, B, D
8	6	6	6	6	8	A, B, C

```
/* called every time-quantum. */
def schedule:
    freeGPUs = numGPUs
    scheduled = {}
    jobs = sort(q, λj: j.pass)
    i = 0
    while freeGPUs > 0 and i < length(jobs):
        if jobs[i].size ≤ freeGPUs:
            scheduled ∪= {jobs[i]}
            freeGPUs -= jobs[i].size
            jobs[i].pass += jobs[i].size / jobs[i].tickets
    return scheduled
```

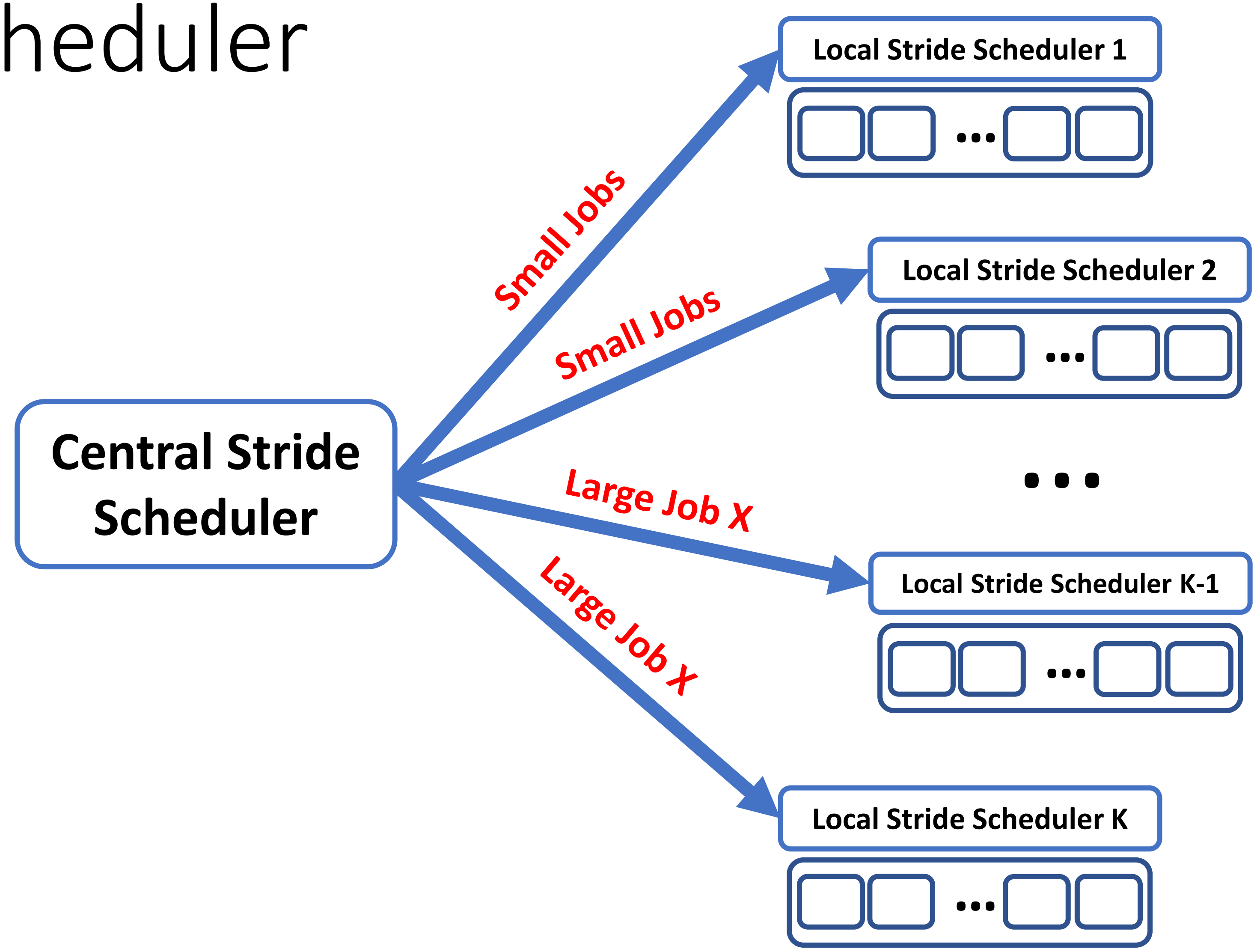
Split-Stride Scheduler



- **Simple:** Run Gang-Aware Stride across all GPUs on a cluster.
  - Not scalable and unbounded migrations.
- **Idea:** Run a Gang-Aware Stride locally on each server.
  - How to run multi-server jobs? Some central coordination is required.

Split-Stride Scheduler

Schedule is fair if the *load*[6] is balanced across all servers.



[6] Refer to the paper for details.

Handling GPU Heterogeneity

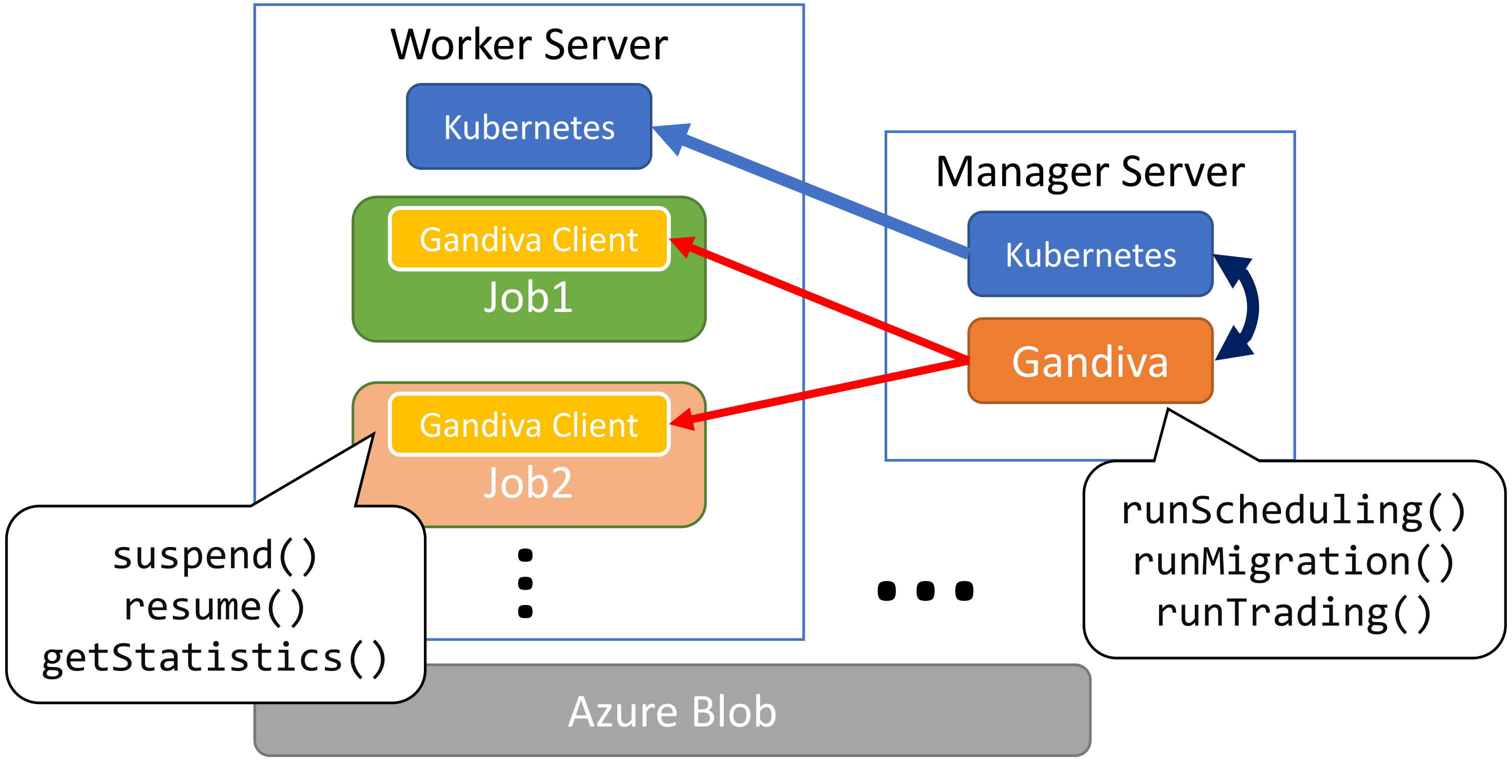
Job	K80 (ms)	K80 / P40	K80 / P100	K80 / V100
VAE	11.5	1.17	1.19	1.25
SuperResolution	207.5	1.43	1.73	1.87
DCGAN	183.4	4.34	4.31	6.42
GRU	48.4	3.00	2.58	4.81
LSTM	48.9	3.10	3.58	4.81
ResNet50	134	3.17	3.34	5.14
ResNeXt50	2005.7	3.70	4.12	6.33

- Transparently profile jobs to determine speedups on all GPU generations
- Assumption: each user submits the same *type* of job.
- For example, as a part of hyperparameter exploration.
- Place jobs on the fastest GPU subject to contention.

Automated Resource Trading

- **Idea:** If we exchange U1's **1** V100 for U2's **p** K80s, both users gain if **1.2 < p < 6.**
- For maximum efficiency gain, trade between highest and lowest speedup users.
- **Issue:** user gaming, for example, user artificially slows down their K80 jobs to win V100s.
- **Idea:** Use speedup as bids in a Vickrey auction, **p** as second-price is *incentive-compatible*. For example, if another user U3 exists with a 2X speedup, then **p** is 2.

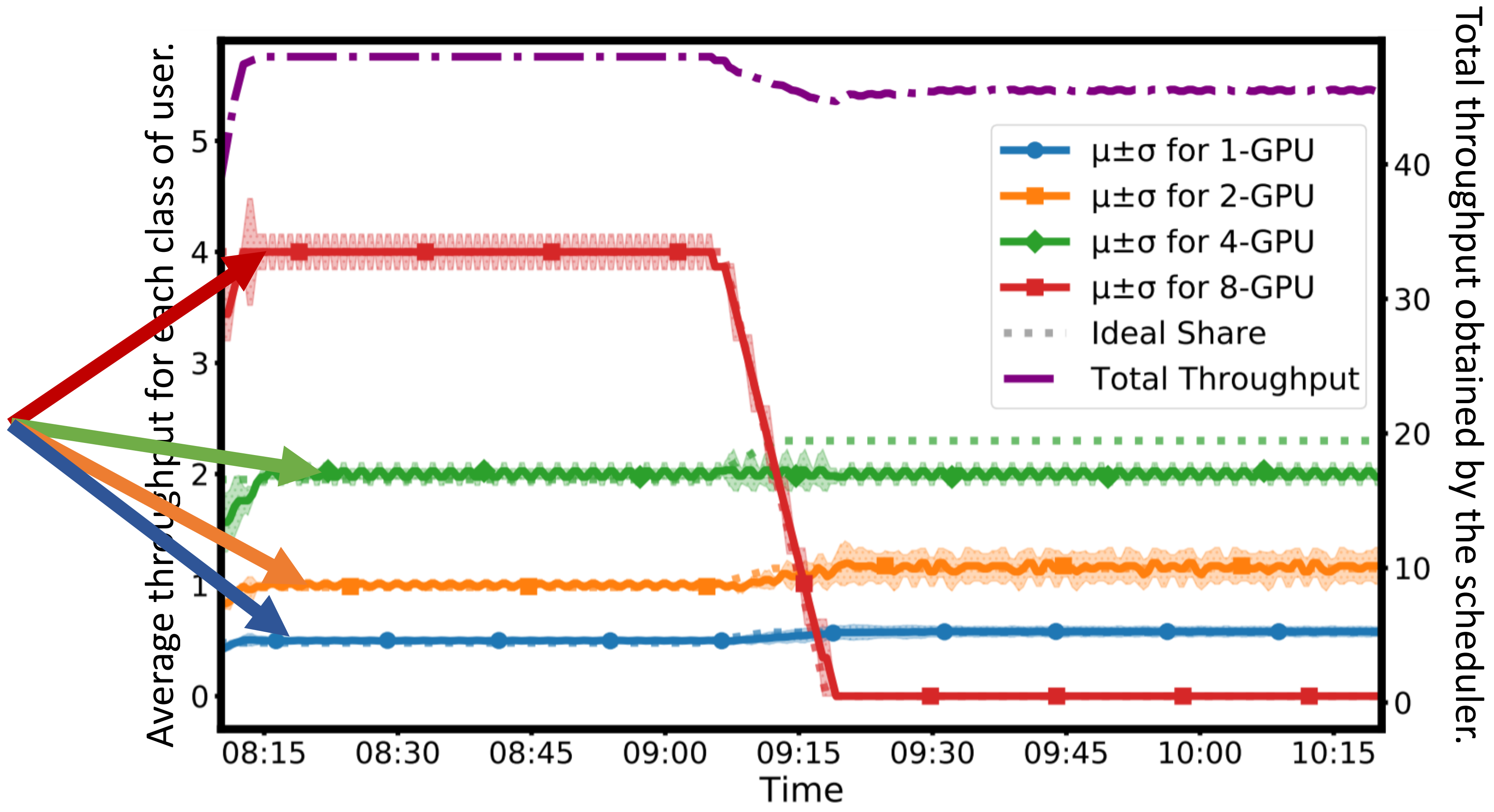
Implementation



- Implemented as a custom scheduler on Kubernetes.
- Manager contacts the Gandiva Client to perform operations like time-slicing.

Fair-Share on a Homogeneous Cluster

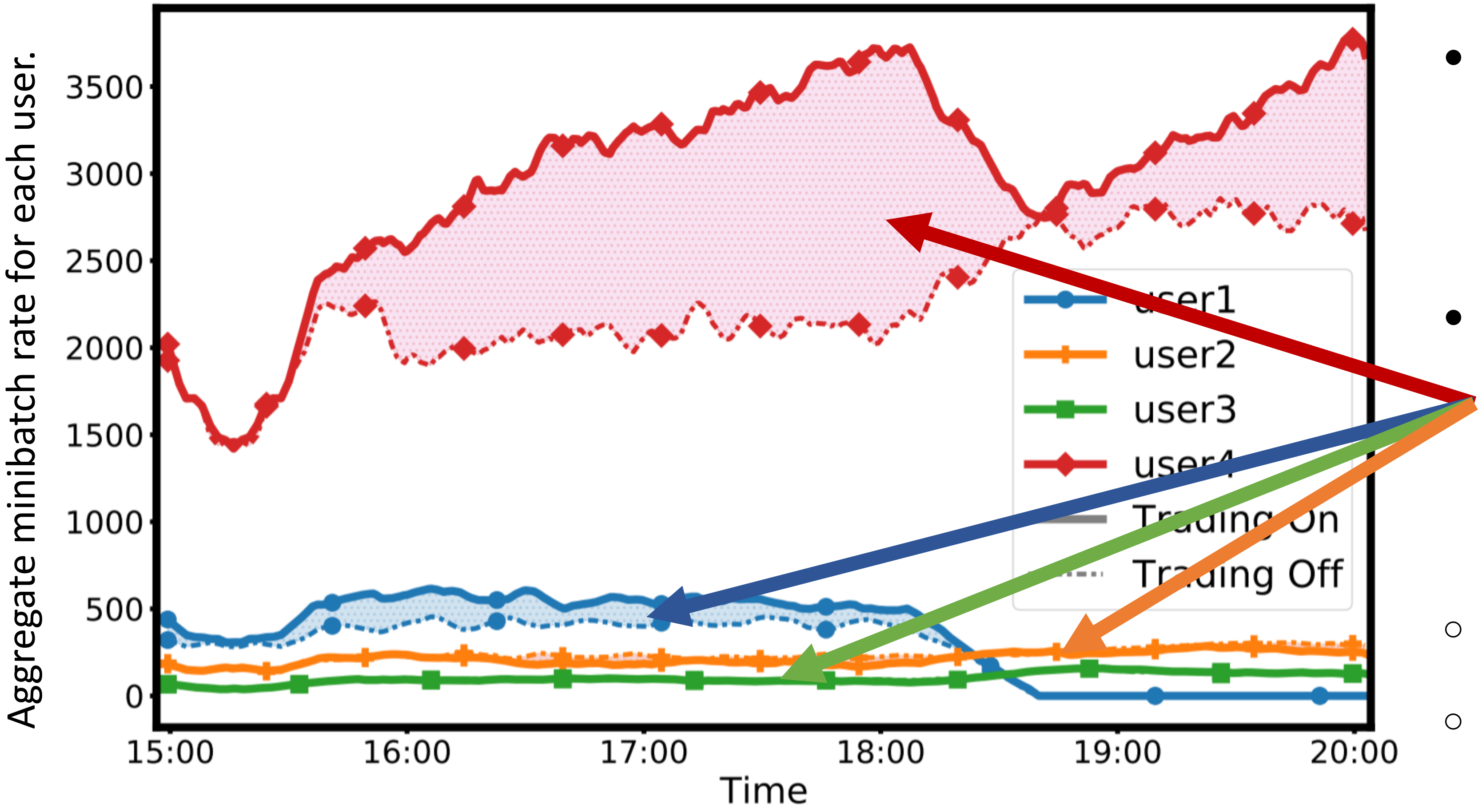
- Each user obtains close to their fair share.



- 48 P100 GPU cluster.
- 70 users with one 1, 2, 4 or 8 GPU jobs with job size distribution derived from Philly Trace[2,7].

[7] https://github.com/msr-fiddle/philly-traces

Benefit of Trading on Heterogeneous Cluster



- Users 1 and 4 exhibit about 30% increase in performance.
- Users 2 and 3 exhibit similar performance.

  - 100 GPU cluster with 12 V100s, 24 P100s, and 128 K80s.
  - 4 users with many 1, 2, or 4 GPU jobs with different speedups.

Summary

- Gandiva$_{fair}$ is a domain specific scheduler for Deep Learning workloads.
- Provides efficient fair-sharing of cluster-wide GPU throughput among users.
- Handles heterogeneous GPUs transparently using profiling and automated resource trading.