COMS E6998 012/15099

Practical Deep Learning Systems Performance

Lecture 10 11/07/19

Logistics

- Homework 3 is due by Nov 20 11:59 PM
- Homework 4 will be posted soon. Due by Nov. 21 11:59 PM.
- Start working on your 10% for technical community participation

Recall from last lecture

- Deep learning benchmarking
- Tensorflow benchmarks
- DAWNBench
- MLPerf
- Time to Accuracy (TTA) metric: variability, scaling, generalization
- Kaggle
- OpenML
- ONNX and ONNX runtime architecture

Optimus Scheduler

- Recall Optimus performance modeling from Lecture 8
- Dynamic scheduling
- Periodically allocates resources to the active jobs by adjusting the numbers and placement of parameter servers/workers in each job in the shared DL cluster.
- Two parts: resource allocation and task placement

Resource Allocation: Optimization Problem

- Optimization Problem: Minimize average completion time of active jobs

$$\text{minimize} \quad \sum_{j \in J} t_j$$

$$\text{subject to:} \quad t_j = \frac{Q_j}{f(p_j, w_j)} \qquad \forall j \in J$$

$$\sum_{j \in J} (w_j \cdot O_j^r + p_j \cdot N_j^r) \leq C_r \qquad \forall r \in R$$

$$p_j \in Z^+, w_j \in Z^+ \qquad \forall j \in J$$

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Resource Allocation: Marginal Gain

- Marginal gain in job completion time

$$\max\{\left(\frac{Q_j}{f(p_j, w_j)} - \frac{Q_j}{f(p_j + 1, w_j)}\right)/N_j^D,$$
$$\left(\frac{Q_j}{f(p_j, w_j)} - \frac{Q_j}{f(p_j, w_j + 1)}\right)/O_j^{D'}\}$$

$\left(\frac{Q_j}{f(p_j, w_j)} - \frac{Q_j}{f(p_j + 1, w_j)}\right)$ Reduction in job completion time when one PS is added to job i

$\left(\frac{Q_j}{f(p_j, w_j)} - \frac{Q_j}{f(p_j, w_j + 1)}\right)$ Reduction in job completion time when one worker is added to job i

- Marginal gain calculated using prediction model

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Marginal Gain based Heuristic Solution

- First allocate one worker and one parameter server to each active job to avoid starvation
- Sort all jobs in order of their marginal gains computed
- Iteratively select the job with the largest marginal gain and add one worker or parameter server to the job, depending on whether adding a worker or adding a parameter server brings larger marginal gain
- Marginal gains of the jobs are updated when their resource allocation changes. The procedure repeats until some resource in the cluster is used up, or marginal gains of all jobs become non-positive.

Task Placement

- Placement of workers and PSs on different servers in the cluster
- Reduce processing time at workers/PSs improve by reducing the time spent on parameters/gradients exchange among workers and PSs

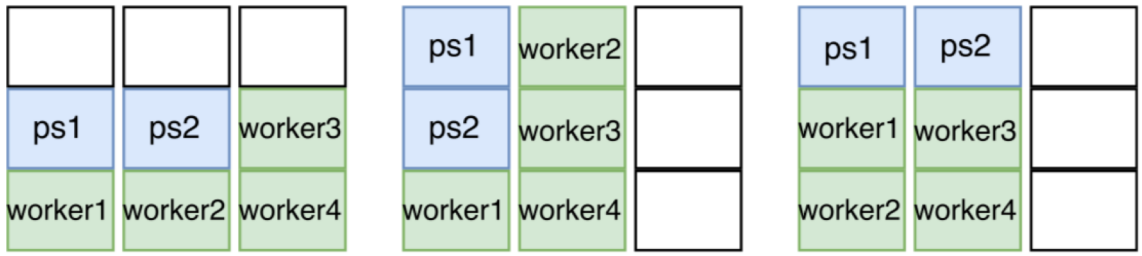


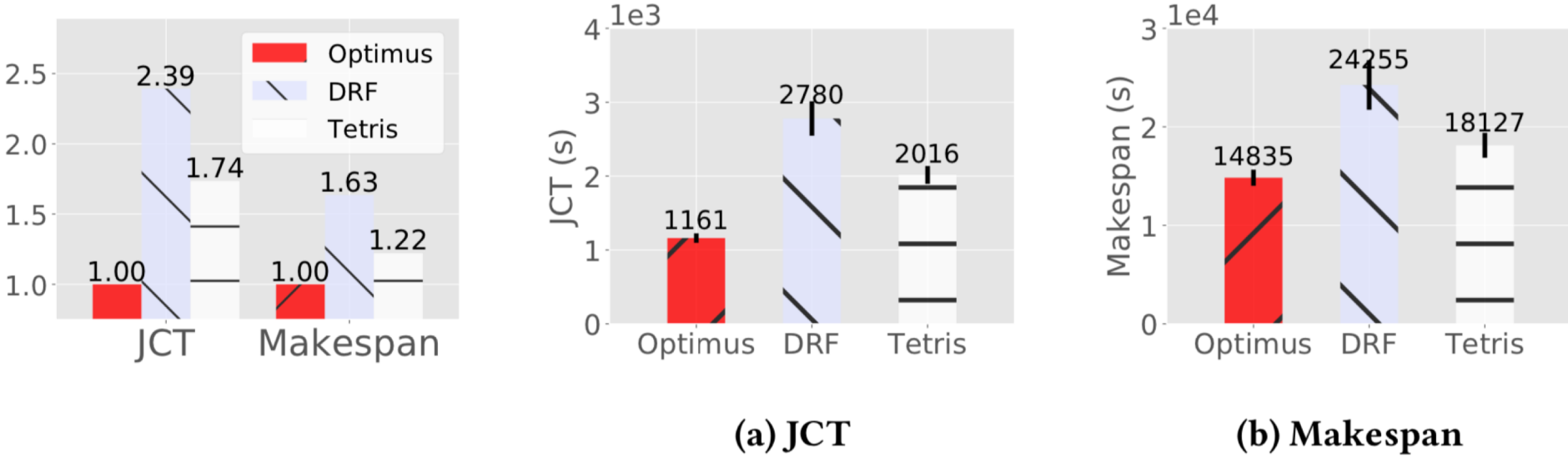
Figure 10: An example of worker/parameter server placement: (c) is the best

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Task Placement Algorithm

- *Principle*: Use the smallest number of servers to host the job, such that the same number of parameter servers and the same number of workers are deployed on each of these servers.

Performance: Job Completion Time and Makespan

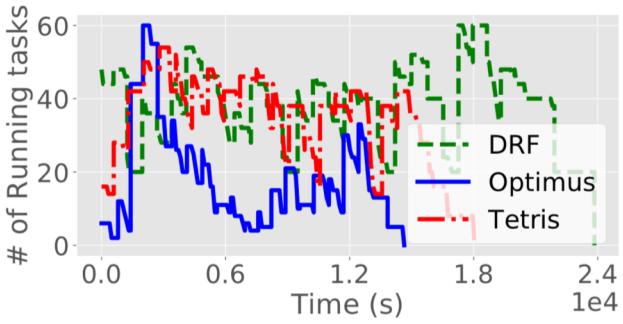


(a) JCT (b) Makespan

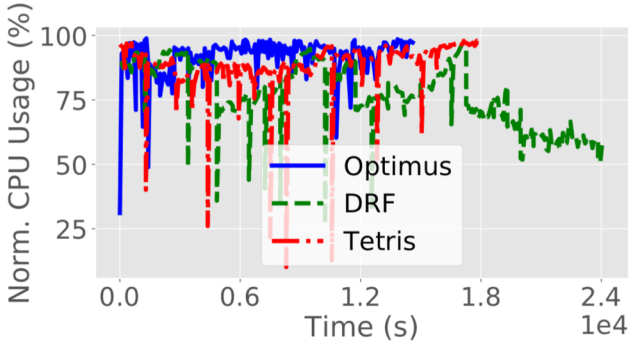
- Reduction in average completion time and makespan by 2.39x and 1.63x respectively
- Optimus, DRF and Tetris use 4.1, 6.7 and 5.0 hours to finish all jobs

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

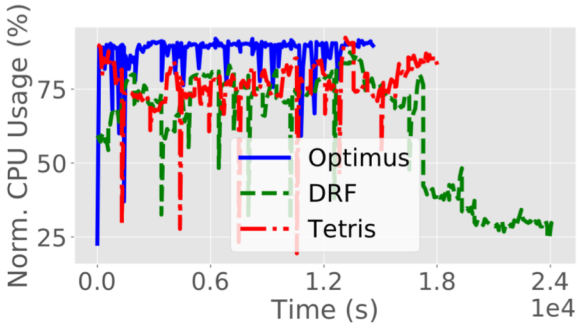
Resource utilization



(a) Number of running tasks



(b) CPU usage on parameter servers

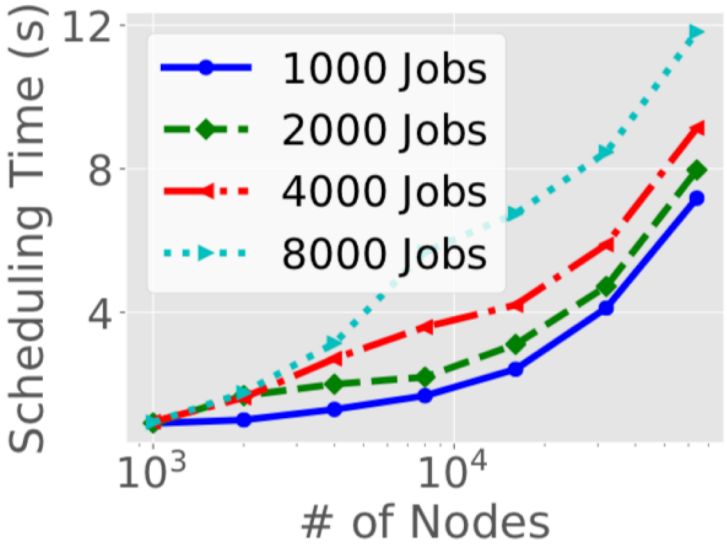


(c) CPU usage on workers

- DRF is work-conserving and allocates as many resources to a job as possible
- Optimus can utilize allocated resources more efficiently.

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Performance: Scalability

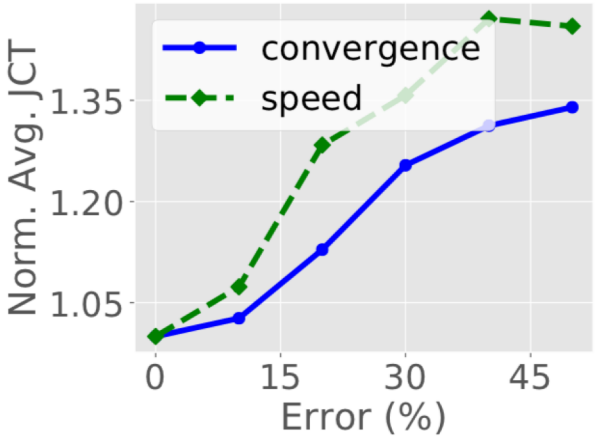


- Can schedule 4,000 jobs (about 100,000 tasks) within 5 seconds on a cluster of 16,000 nodes
- Comparable to the performance of Kubernetes' default scheduler, i.e., 150,000 tasks in 5,000 nodes within 5 seconds
- Overall resource adjustment overhead is 2.54% of the makespan

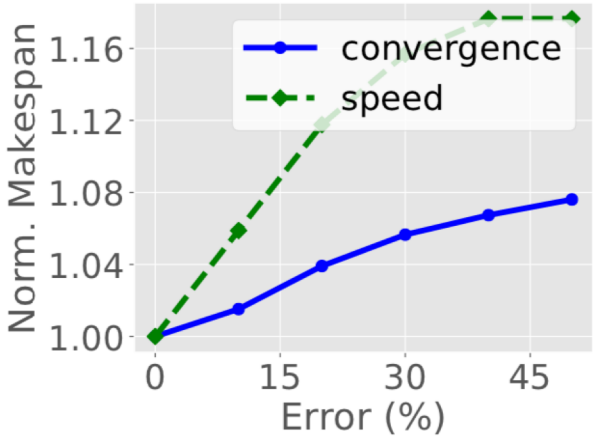
Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Optimus Scheduler: Sensitivity to Prediction Error

Sensitivity to prediction error



(a) Average completion time

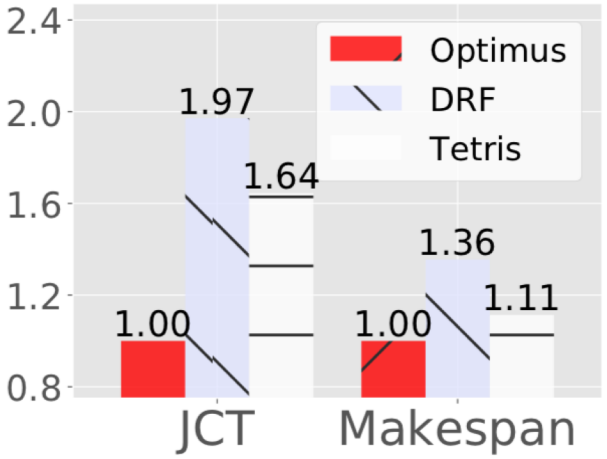


(b) Makespan

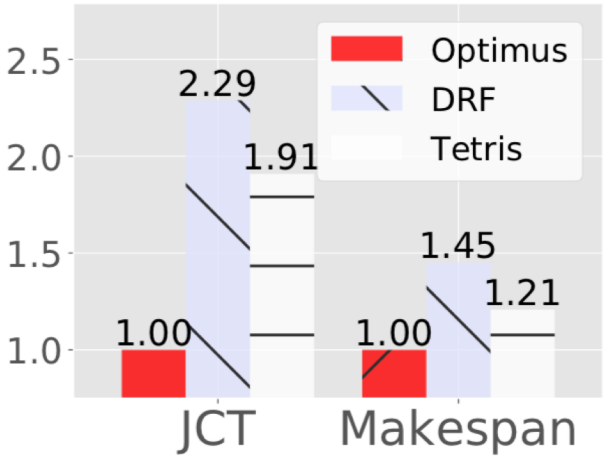
- Compare to the error of convergence estimation, the error of speed estimation affects the scheduler performance more
- Downgrade the priority of a job a bit when it is at the beginning state (i.e., larger prediction errors) by multiplying its marginal gain by a factor (e.g., 0.95). Smaller marginal gain of a job means less resources allocated to it, thus mitigating the influence of large prediction errors at the start of training.

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Optimus Scheduler: Sensitivity to Workload



(a) Async

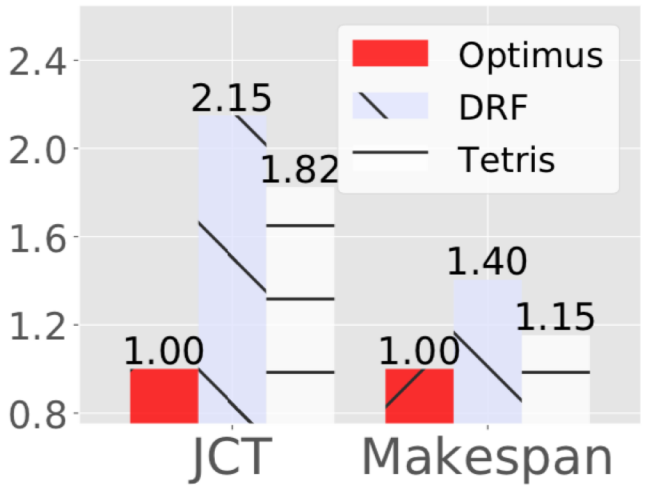


(b) Sync

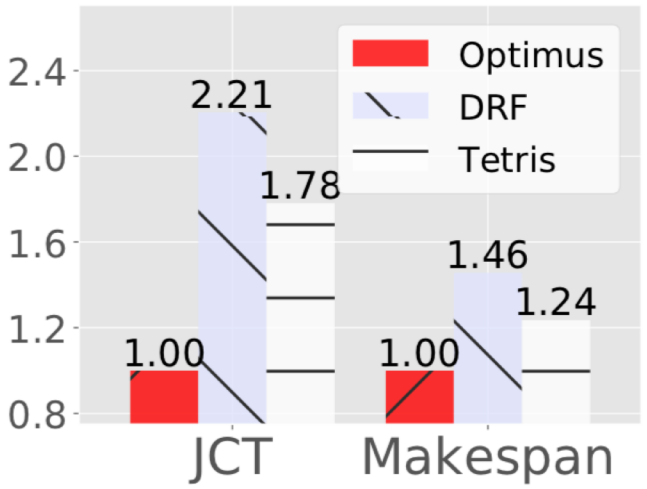
- Performance gains with synchronous are higher than with asynchronous
- Both convergence estimation error and speed estimation error are small with synchronous

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Optimus Scheduler: Sensitivity to Workload



(a) Poisson



(b) Google cluster trace

- Google cluster job arrival trace over a 7 hr long period

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Optimus: Where is the gain coming from ?

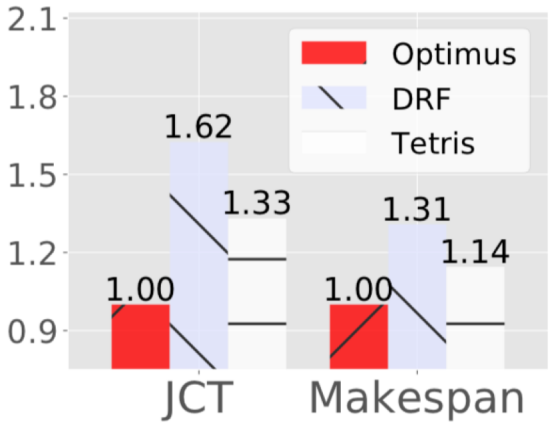


Figure 18: Effectiveness of resource allocation algorithm

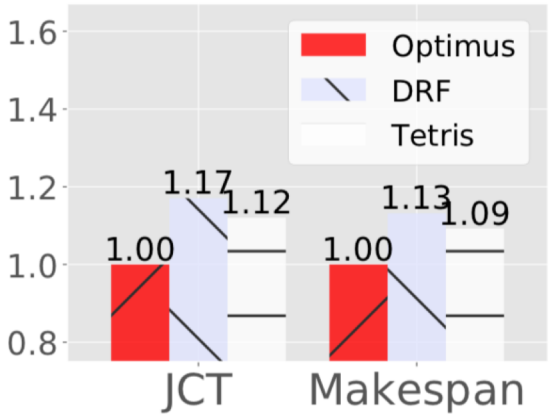


Figure 19: Effectiveness of task placement algorithm

Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018

Problem with current scheduling of Deep Learing Training (DLT) jobs on clusters

- Cloud operators and large companies that man- age clusters of tens of thousands of GPUs rely on cluster schedulers to ensure efficient utilization of the GPUs.
- Use a traditional cluster scheduler, such as Kubernetes or YARN treating a DL job simply as yet another big-data job
- DLT jobs are assigned a fixed set of GPUs at startup
  - Job holds exclusive access to its GPUs until completion
  - Head-of-line blocking, preventing early feedback and resulting in high queuing times for incoming jobs
  - Low GPU utilization
- Existing schedulers treat DLT job as a black box

DLT Job Features

- *Feedback-driven exploration*
  - Hyperparameter search
  - Do not read to run jobs to completion
  - Require early indication of performance
- *Head-of-line-blocking*, as long-running jobs hold exclusive access to the GPUs until completion, while multi- jobs depending on early feedback wait in queue
- DLT jobs are heterogeneous targeting diverse domains
  - Jobs widely differ in terms of memory usage, GPU core utilization, sensitivity to interconnect bandwidth, and/or interference from other jobs

DL Jobs: Sensitivity to Locality

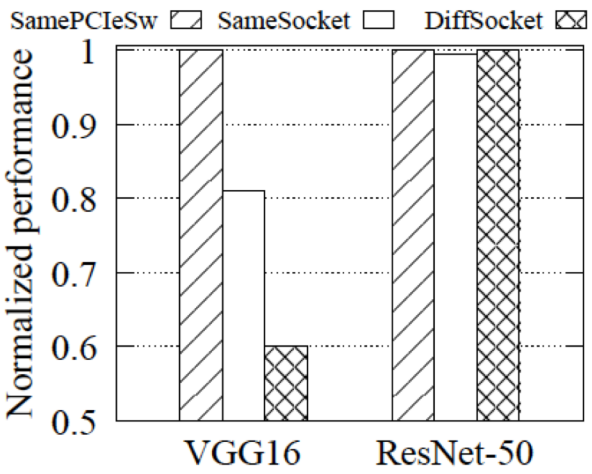


Figure 1: Intra-server locality.

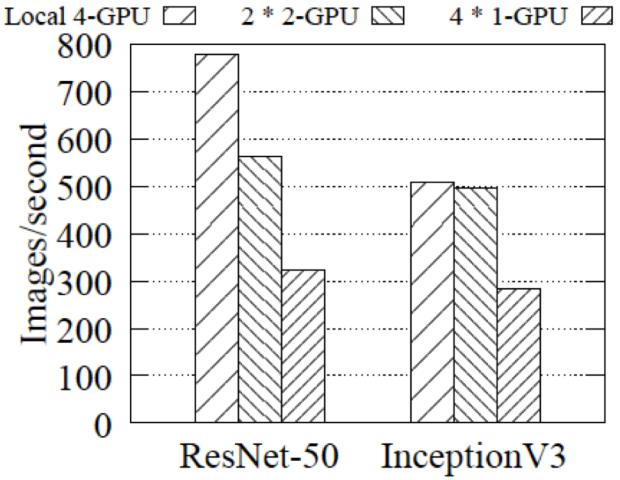


Figure 2: Inter-server locality.

- Different DLT jobs exhibit different levels of sensitivity to inter-GPU affin ity. Even for GPUs on the same machine, we observe dif- ferent levels of inter-GPU affinity due to asymmetric architecture
- DLT scheduler has to take into account a job's sensitivity to locality when allocating GPUs

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

DL Jobs: Sensitivity to Interference

Intra-server interference

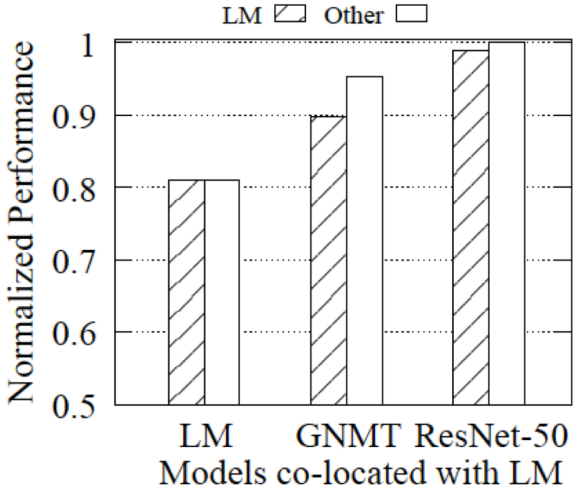


Figure 3: 1-GPU interference.

Inter-server interference

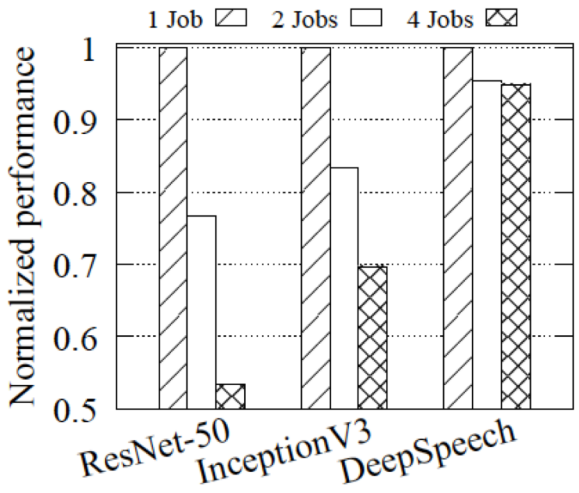
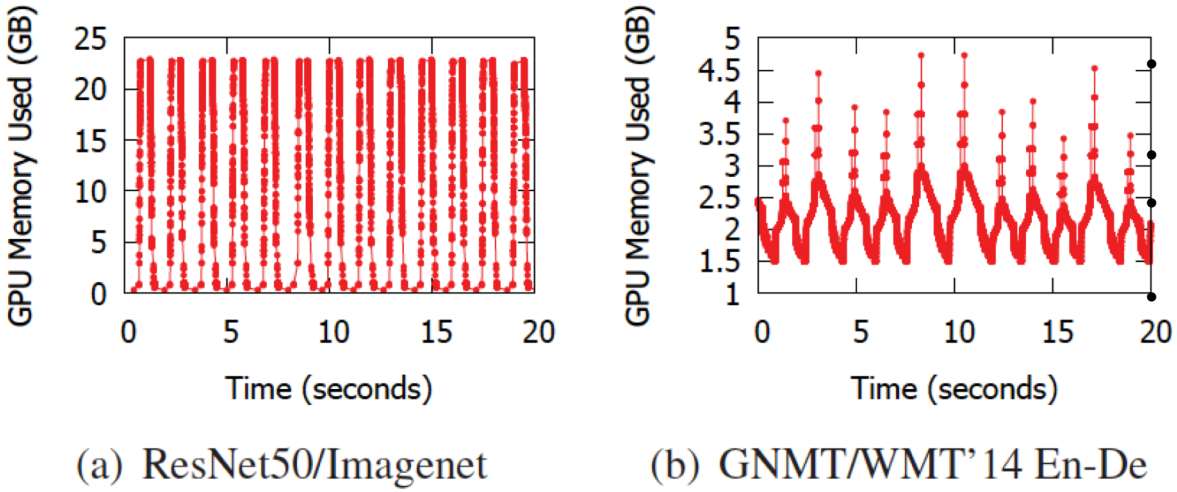


Figure 4: NIC interference.

- Different DLT jobs exhibit different degrees of interference
- Interference exists both for single-GPU and multi-GPU jobs
- When running multiple 2-GPU jobs, where each GPU is placed on different server, ResNet-50 shows up to 47% slowdown, InceptionV3 shows 30% slow- down

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

DL Jobs: Intra Job Predictability



(a) ResNet50/Imagenet (b) GNMT/WMT'14 En-De

Figure 5: GPU memory usage during training.

Gradient descent algorithm performing many mini-batch iterations

Cyclic pattern in GPU memory used

Each cycle corresponds to one mini-batch processing

Leveraging predictability

- Define micro-tasks (collection of cycles) as scheduling units
- Very efficient suspend/resume and Migration by performing them at end of cycle
- Proxy for applying techniques like packing and migration

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Scheduler

- **Goals**
  - Early feedback
    - Gandiva supports over-subscription by allocating GPUs to a new job immediately and using the suspend-resume mechanism to provide early results
  - Cluster efficiency
    - Through a continuous optimization process that uses profiling and a greedy heuristic that takes advantage of mechanisms such as packing, migration, and grow-shrink
- Modes of operations
  - Reactive
  - Introspective
- Scheduling framework to exploit the unique characteristics of the deep learning workload

Gandiva Mechanisms

- **Remove the exclusivity and fixed assignment of GPUs** to DLT jobs in three ways
- First, during overload, instead of waiting for current jobs to depart, *Gandiva* allows incoming jobs to **time-share GPUs** with existing jobs. This is enabled using a custom suspend-resume mechanism tailored for DLT jobs along with selective packing.
- Second, *Gandiva* supports **efficient migration of DLT jobs** from one set of GPUs to another. Migration allows time-sliced jobs to migrate to other (recently vacated) GPUs or for de-fragmentation of the cluster so that incoming jobs are assigned GPUs with good locality.
- Third, *Gandiva* supports a **GPU grow-shrink mechanism** so that idle GPUs can be used opportunistically. In order to support these mechanisms efficiently and enable ef- fective resource management, *Gandiva* introspects DLT jobs by continuously profiling their resource usage and estimating their performance. We now describe each of these mechanisms.

Gandiva Mechanisms

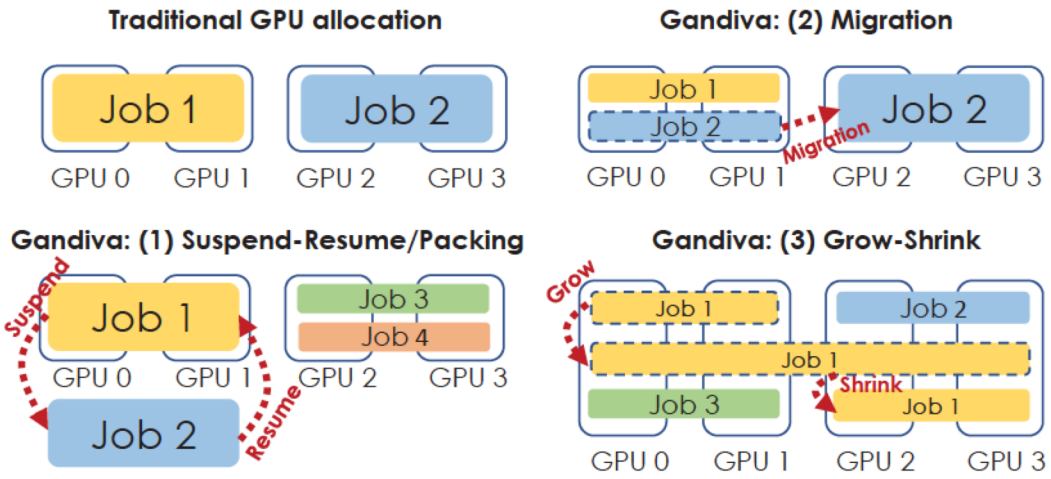


Figure 6: GPU usage options in Gandiva.

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Scheduler: Reactive Mode

- Designed to take care of events such as job arrivals, departures, and machine failures
- Conventional schedulers operate in this mode
- Gandiva allows over-subscription
  - When a server is oversubscribed, weighted round-robin scheduling to give each job its fair time-share

Node Allocation Policy in Gandiva

Algorithm 1 getNodes(in job, out nodes)

```
1:  nodes0 ← findNodes(job.gpu, affinity ← job.gpu)
2:  nodes1 ← minLoadNodes(node0)
3:  nodes2 ← findNodes(jog.gpu, affinity ← 0)
4:  nodes3 ← findNodes(job.gpu)
5:  if nodes1 and height(nodes1) < 1:
6:      return nodes1              // Same affinity with free GPUs
7:  if nodes2 and numGPUs(nodes2) ≥ job.gpu:
8:      return nodes2              // Unallocated GPU servers
9:  if nodes3:
10:     return nodes3              // Relax affinity constraint
11: elif nodes1:
12:     return nodes1              // Allow over-subscription
13: else:
14:     enqueue(job)               // Job queued
```

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Scheduling Example

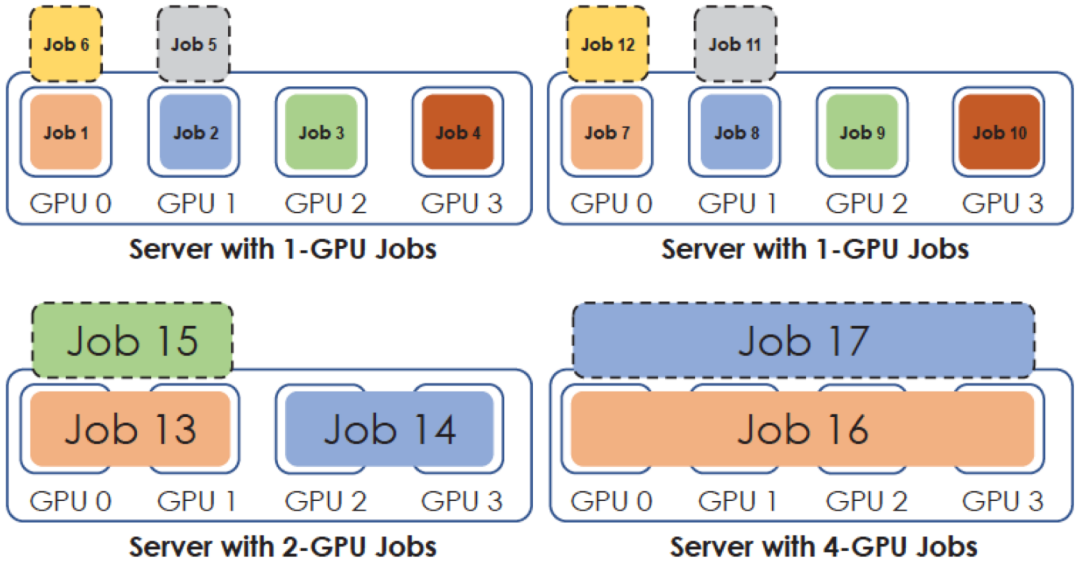


Figure 7: Scheduling example in a 16-GPU Cluster.

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Scheduler: Introspective Mode

- Gandiva continuously monitors and optimizes placement of jobs to GPUs in the cluster to improve the overall utilization and the completion time of DLT jobs
- Techniques in introspective mode:
  - Packing
  - Migration
  - Time Slicing
    - Considered only during overload.
    - Run two or more jobs simultaneously on a GPU to increase efficiency

Job Migration

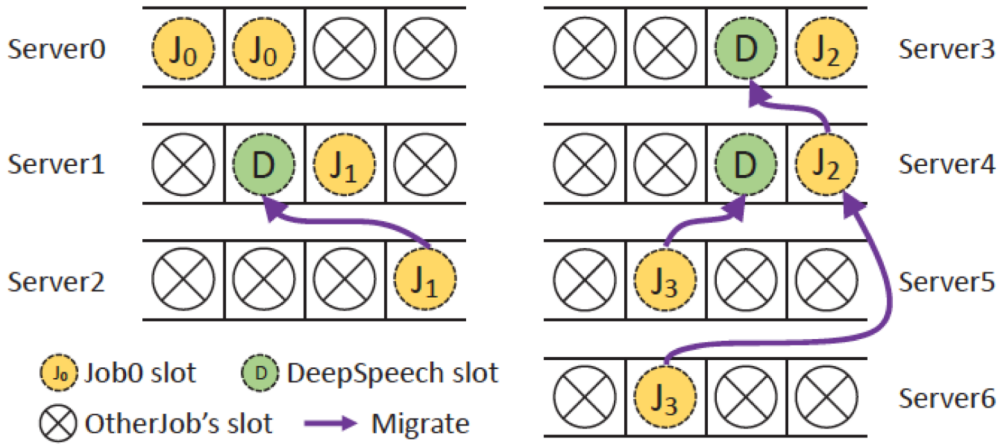


Figure 8: Job migration in a shared cluster.

Co-locate jobs after a departure

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Performance: Time Slicing

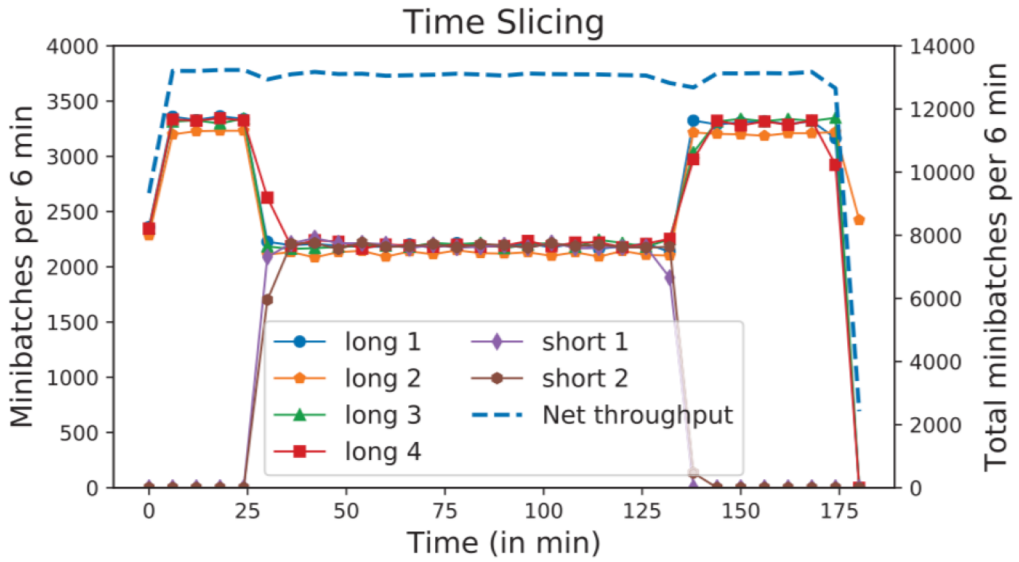


Figure 9: Time slicing six 1-GPU jobs on 4 GPUs.

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Performance: Packing

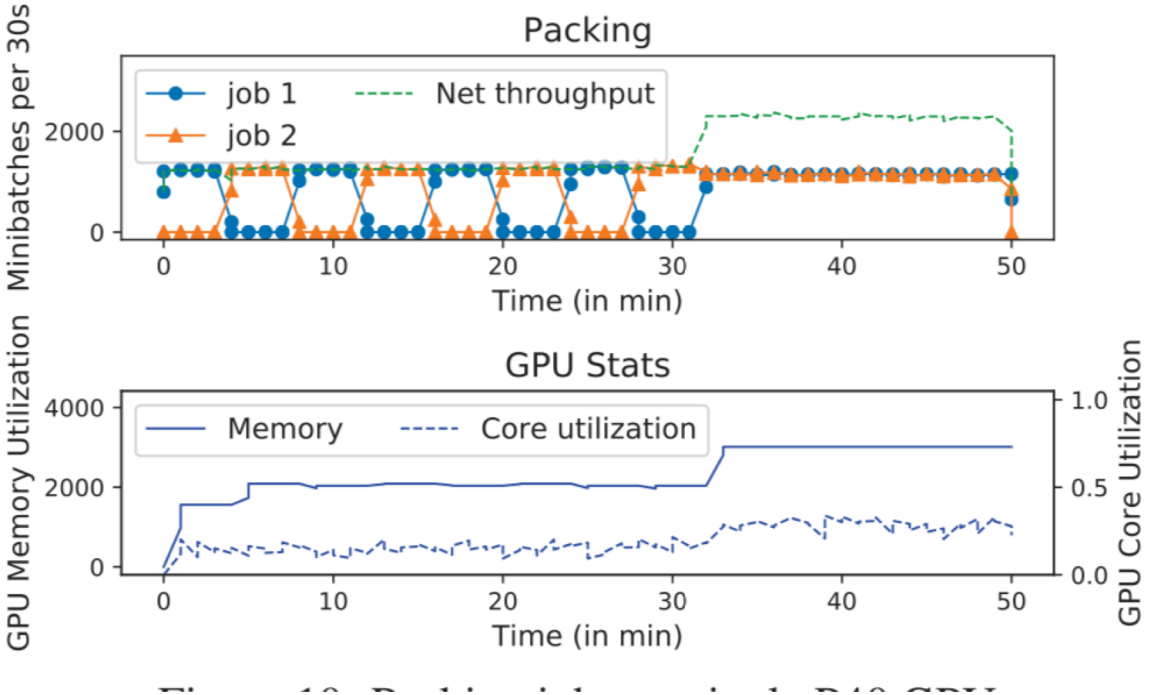
Job	GPU Util (%)	Time Slicing (mb/s)	Packing Max (mb/s)	Packing Gain (%)
VAE [29]	8.7	81.8	419.3	412
SuperResolution [43]	14.1	40.3	145.2	260
RHN [58]	61.6	10.1	14.8	46
SCRNN [37]	66.8	16.7	23.3	39
MI-LSTM [52]	76.2	22.2	25.9	17
LSTM [5]	87.2	63.8	53.0	-16
ResNet-50 [24]	94.0	10.3	9.0	-13
ResNext-50 [53]	98.9	83.6	74.4	-11

Table 1: Packing multiple jobs on P40 (mb/s = minibatches/s).

Gandiva adopts a profiling-based ap- proach to packing.

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Performance: Packing



Jobs are initially being time- sliced on the same P40 GPU. After some time, the scheduler concludes that their memory and GPU core utilization is small enough that packing them is feasible and schedules them together on the GPU. The scheduler continues to profile their performance. Because their aggregate performance improves, packing is retained; otherwise (not shown), packing is undone and the jobs continue to use time-slicing.

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Performance: Grow-Shrink

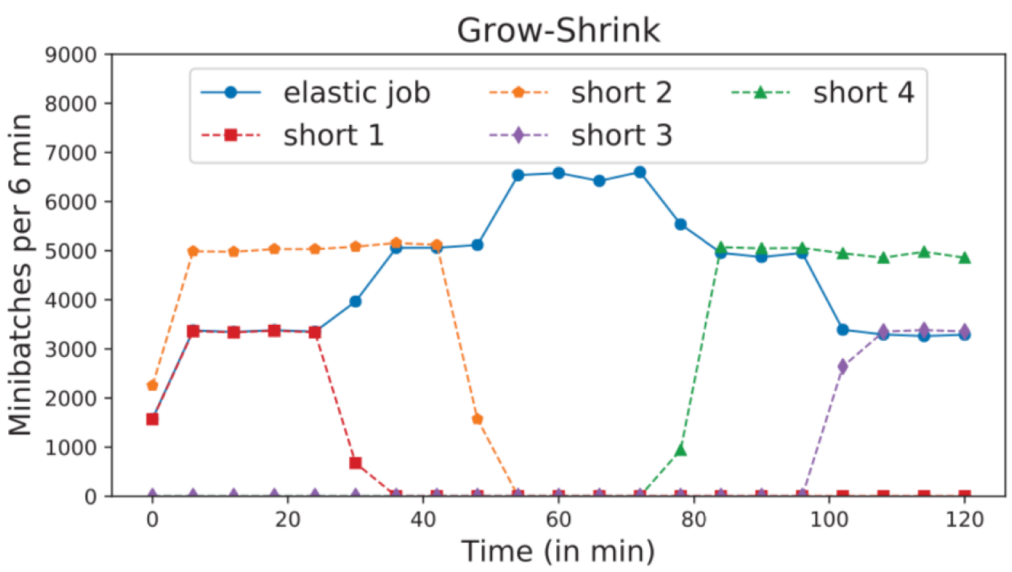


Figure 11: Grow from 1 to 4 GPUs, Shrink to 1-GPU.

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Performance: Migration

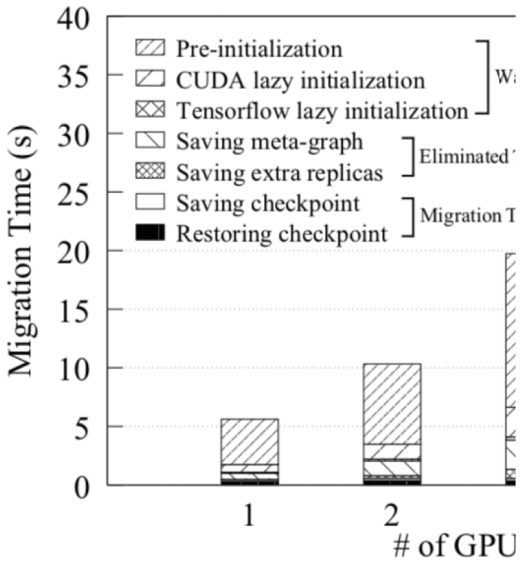


Figure 12: The breakdown of TF

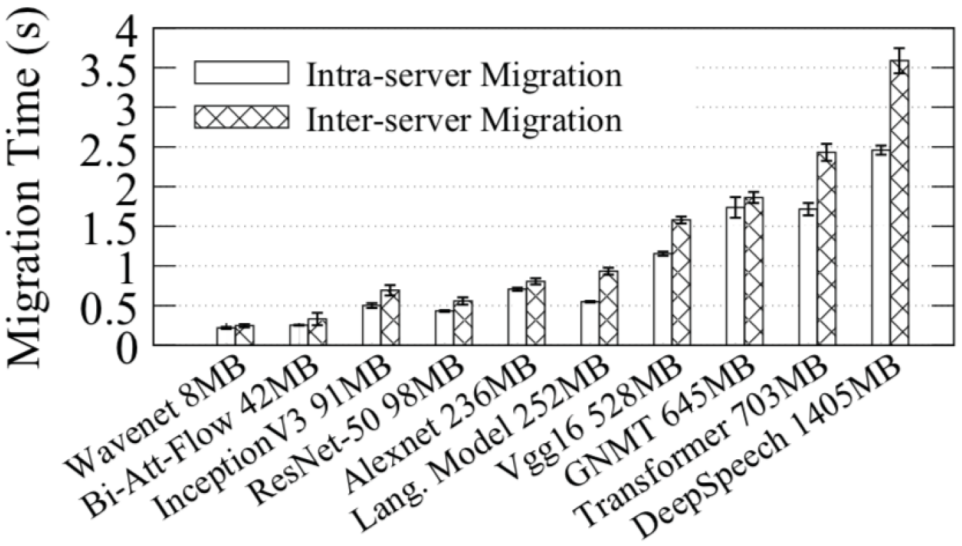


Figure 13: Migration time of real workloads.

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Gandiva Performance: Cluster GPU Utilization

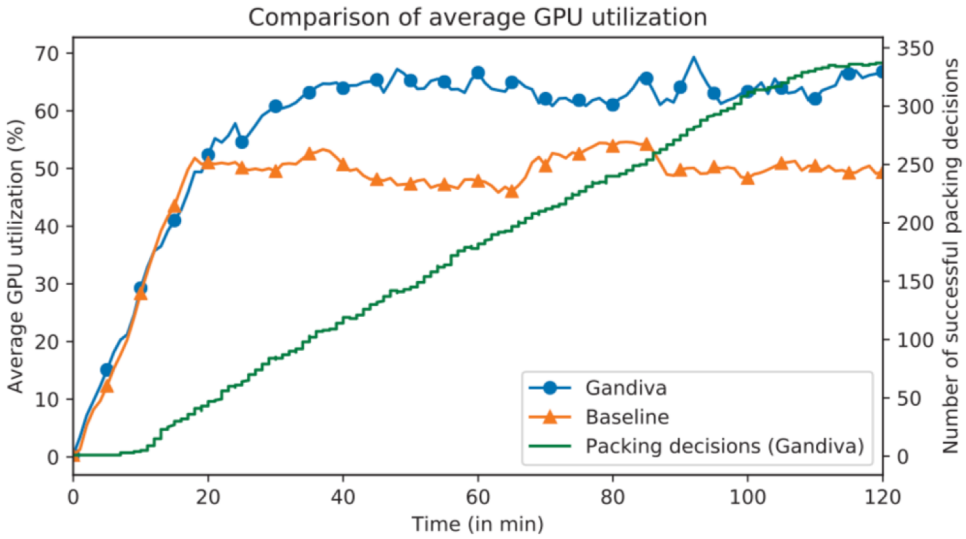


Figure 15: Cluster GPU utilization.

Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning. OSDI 2018

Seminar Reading List

- **DL Cluster Scheduling**
  - Xiao et al. Gandiva: Introspective Cluster Scheduling for Deep Learning**.** OSDI 2018
  - Peng et al. Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters. Eurosys 2018