Gandiva: Introspective Cluster Scheduling for Deep Learning

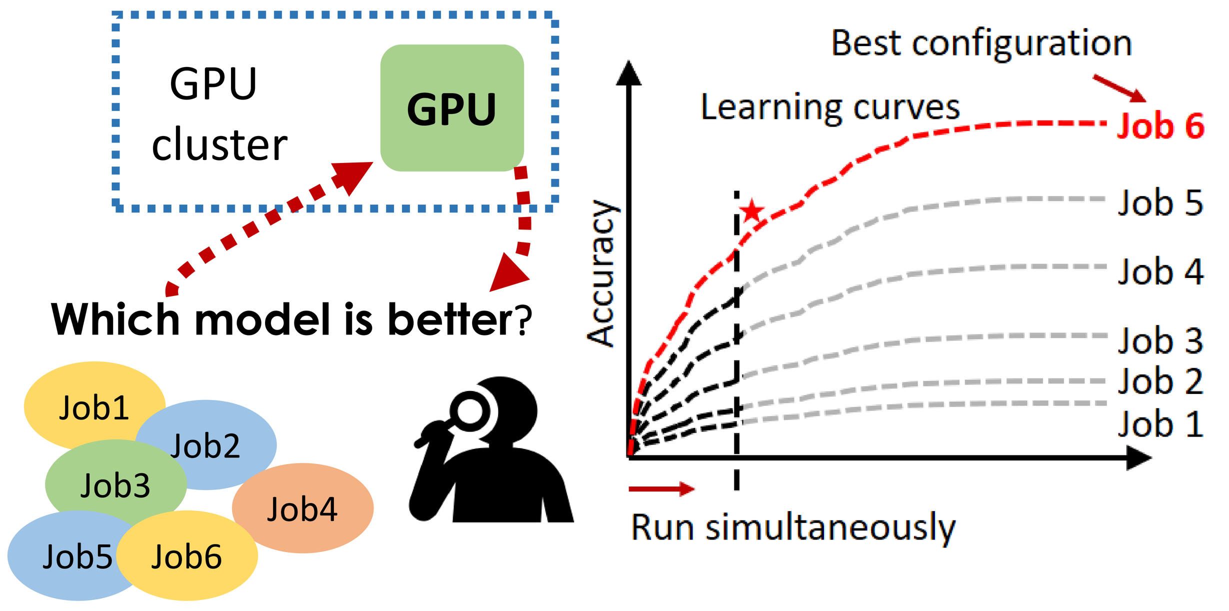
Wencong Xiao♠†*, Romil Bhardwaj†*, Ramachandran Ramjee†, Muthian Sivathanu†, Nipun Kwatra†, Zhenhua Han♣†, Pratyush Patel†, Xuan Peng♦†, Hanyu Zhao●†, Quanlu Zhang†, Fan Yang†, Lidong Zhou†

♠ Beihang University, † Microsoft Research, ♣ The University of Hong Kong,
♦ Huazhong University of Science and Technology, ● Peking University

Characteristics of Deep Learning Jobs

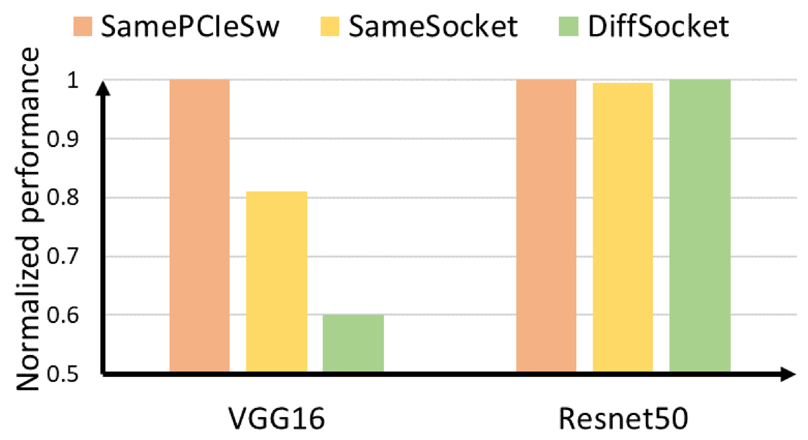
Feedback-driven exploration

- Deep learning experiments today use manual or automatic (AutoML) trial-and-error techniques to find the best model



Model Sensitivity

- DL jobs have different sensitivities to resource affinity, due to network architecture or hyper-parameters (e.g., batch-size)

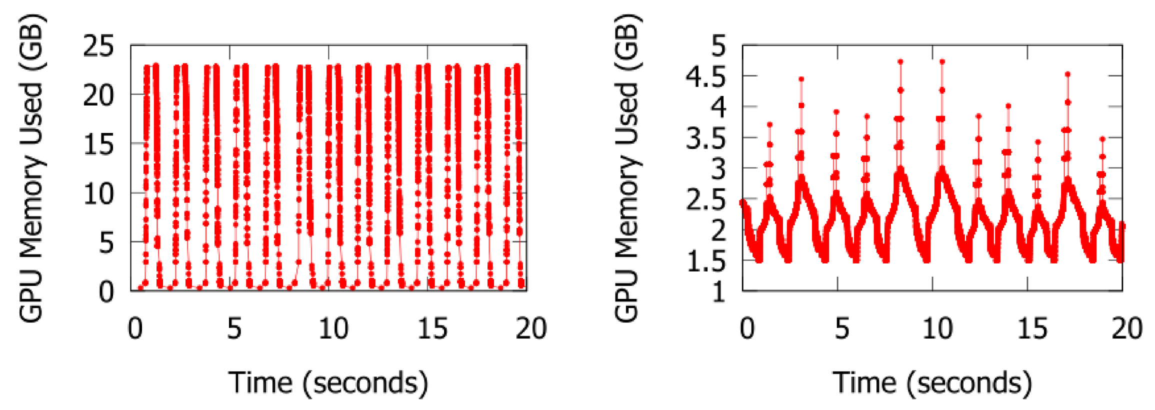


Intra-server locality

- Other cases: Inter-server locality, 1-GPU interference, NIC interference

Intra-job Predictability

- GPU Memory usage follows a cyclic pattern aligned with mini-batch boundaries, usually with more than 10x difference in utilization within a mini-batch

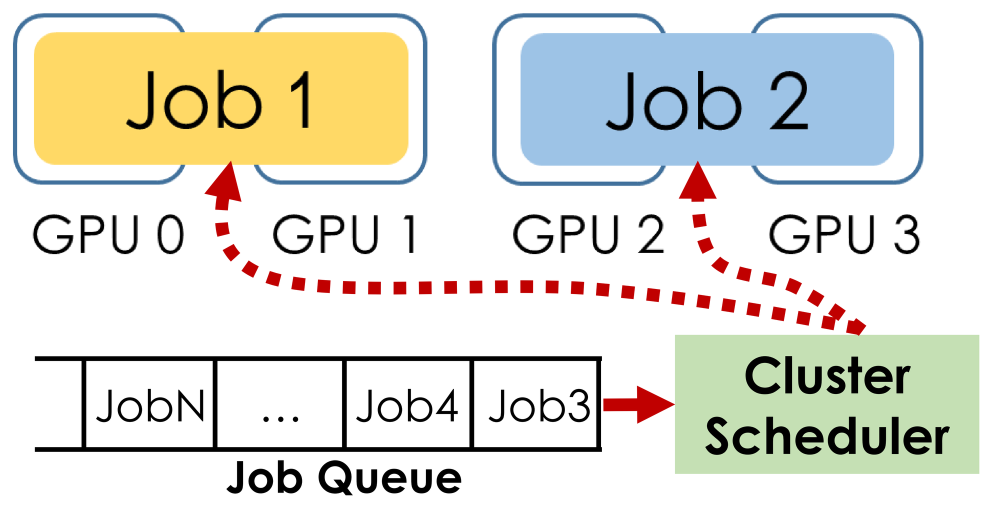


(a) ResNet50/Imagenet (b) GNMT/WMT'14 En-De

GPU Memory usage during training

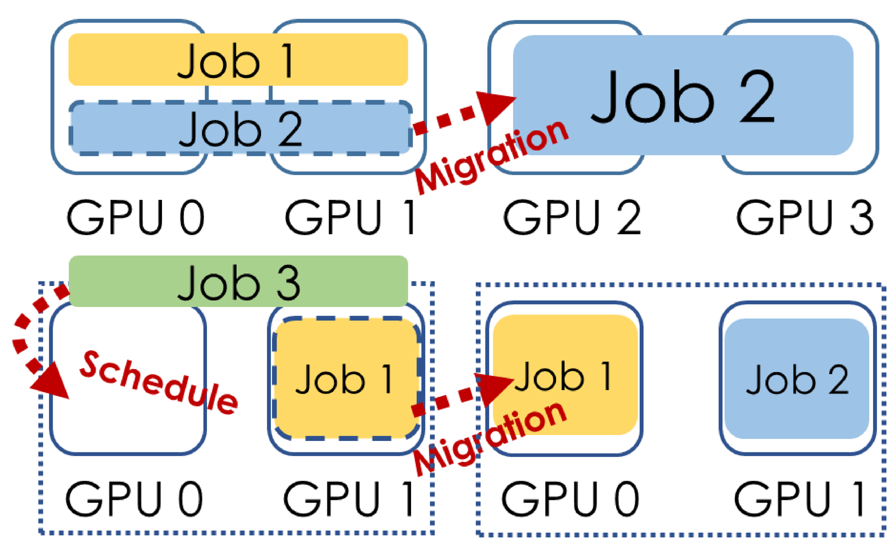
Scheduling Mechanisms for Deep Learning

Traditional GPU Allocation



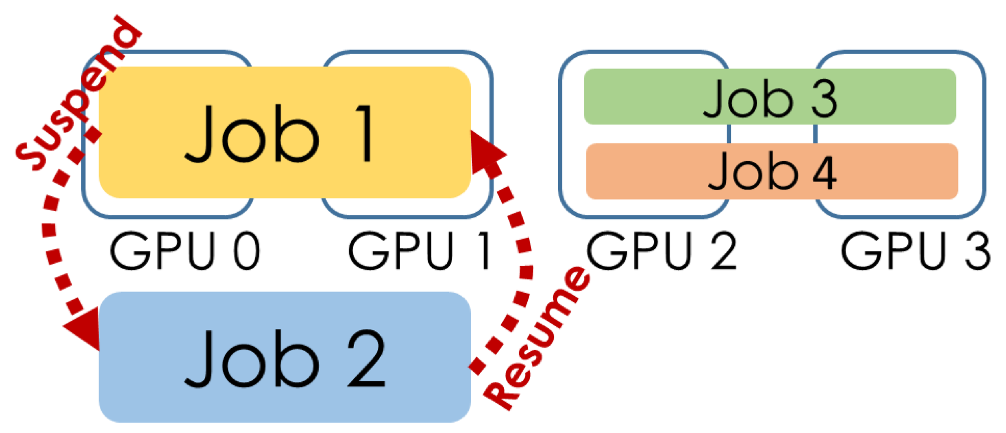
- Allocation reactively at job arrival and departure
- Dedicated GPUs for a job in its whole lifetime
- Jobs queued if no qualified resources

Migration



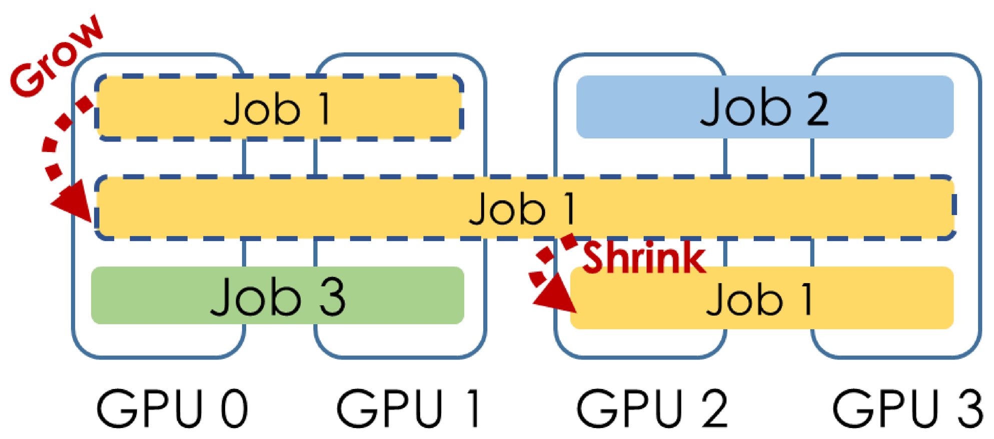
- **Generic GPU processes:** Use CRIU to dump and restore process state across machines
- **Checkpoint-aware processes:** repurpose TF checkpointing APIs to save and restore state. Pre-warm libraries for fast migration.

Suspend-Resume/Packing



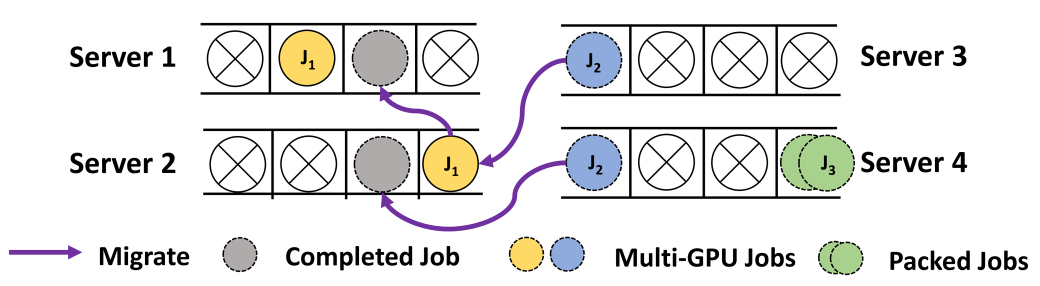
- Copy GPU memory to CPU at mini-batch boundaries
- Restore state from CPU memory on resume
- Or, run multiple processes simultaneously on a GPU.

Grow-Shrink



- Opportunistically scale jobs to idle GPUs
- Vacate GPUs on-demand
- Depends on job capabilities to utilize additional GPUs

Introspective Policies



Over-subscription

- **Time-slice** to allow multiple jobs to run simultaneously with a weighted time-share
- **Pack** multiple jobs in the same server if jobs have light-weight resource requirements

Runtime adjustment

- **Migrate** jobs at mini-batch boundary if better resources appear
- **Defrag** GPUs to better compact resources for multi-GPU jobs
- **Grow** to more resources when available and **shrink** when required

Profiling for introspection

- **Monitor** resource utilization (e.g., GPU utilization and memory)
- Non-invasive **progress rate estimation** for scheduling decisions

Experimental Results

Hyperparameter Search with Time-slicing

Search across 12 dimensions - LeNet on CIFAR-10

Up to 7x faster hyperparameter search

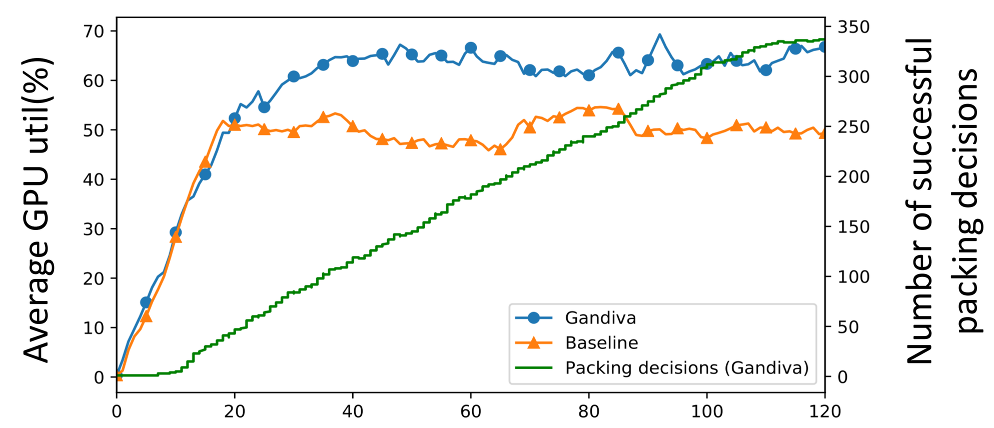
	Position	93th (25%)	187th (50%)	280th (75%)	365th (98%)
4 GPUs	Baseline	691.5	1373.0	2067.2	2726.4
	Gandiva	125.5	213.8	302.4	387.1
	Speedup	5.51x	6.42x	6.84x	7.04x
16 GPUs	Baseline	253.0	492.7	731.7	970.0
	Gandiva	74.4	103.7	135.4	162.6
	Speedup	3.40x	4.75x	5.40x	5.96x

Time to find a qualified model (minutes)

Cluster Experiment: Time-slicing + Packing

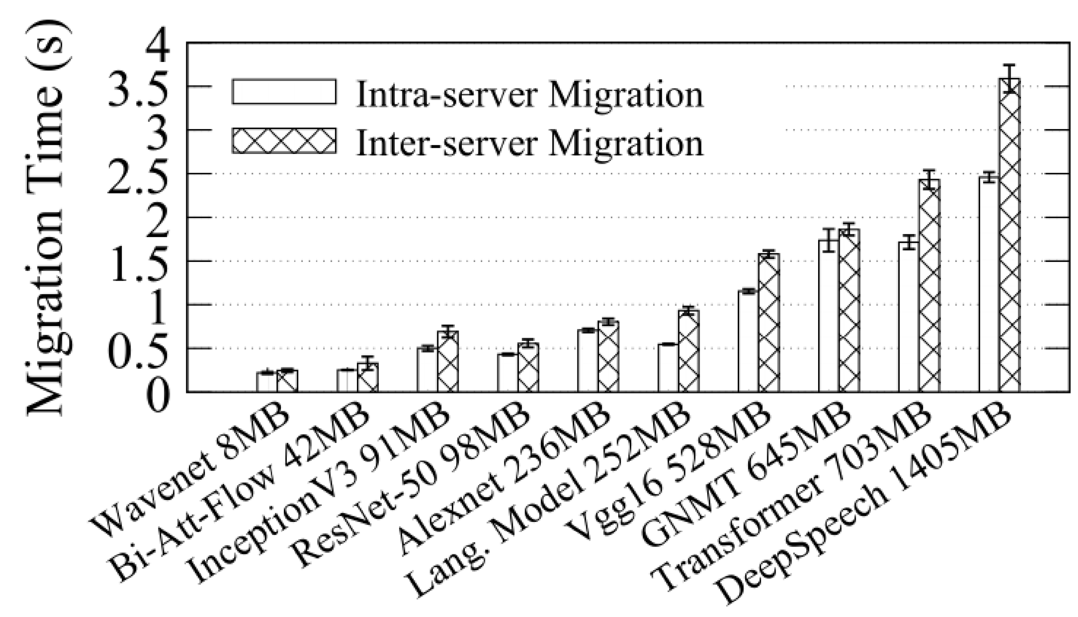
Mixed PyTorch jobs on 180 Tesla GPUs

26% increase in cluster GPU utilization
4.5x faster job feedback



Cluster GPU utilization

Low overhead Suspend/Resume & Migration

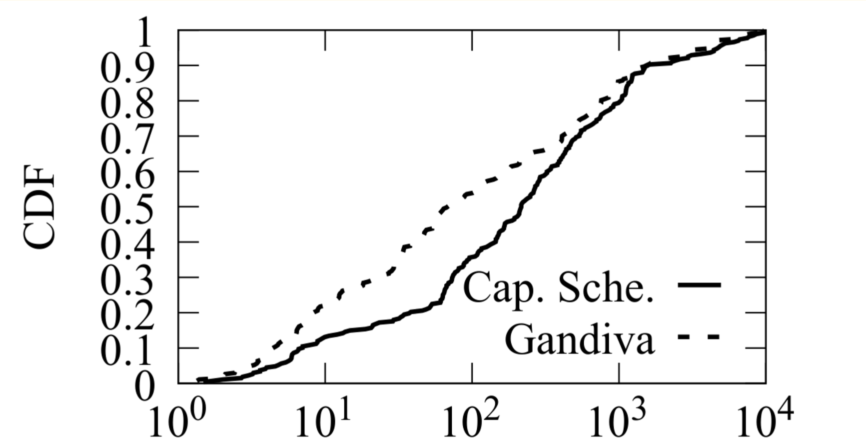


Migration time of real workloads

Cluster Experiment: Time-slicing + Migration

9-day trace from Microsoft servers on 100 GPUs

27% reduction in job completion time
13.6x faster AutoML in shared environments



CDF of job completion time