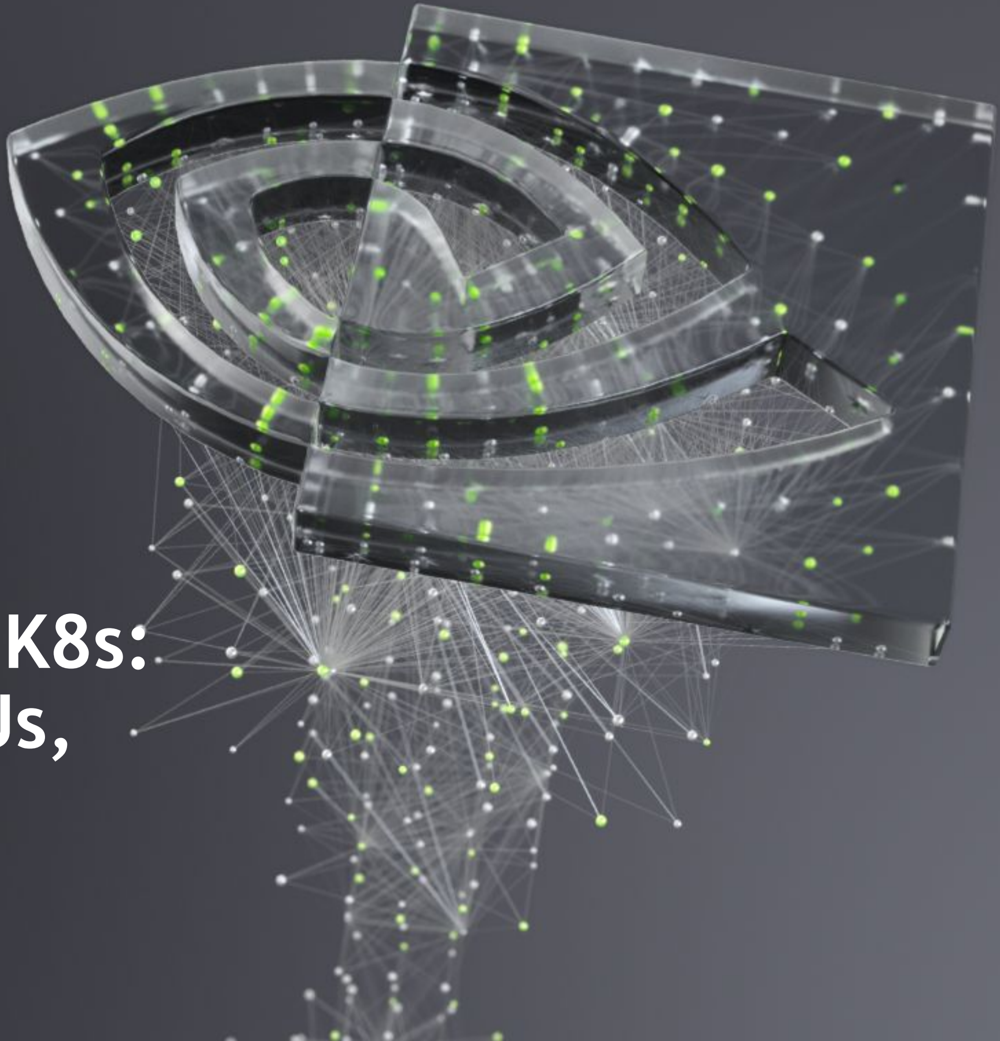
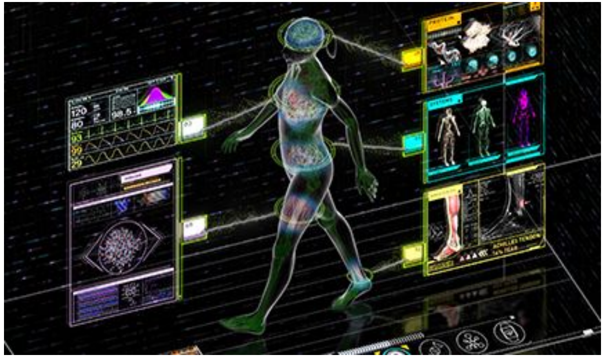


Multi-node Jobs with K8s: Gang Scheduling, GPUs, MPI and RDMA

Madhukar Korupolu, Sanjay Chatterjee

Deep Learning Applications



AI / DL: Models, Frameworks, Hardware

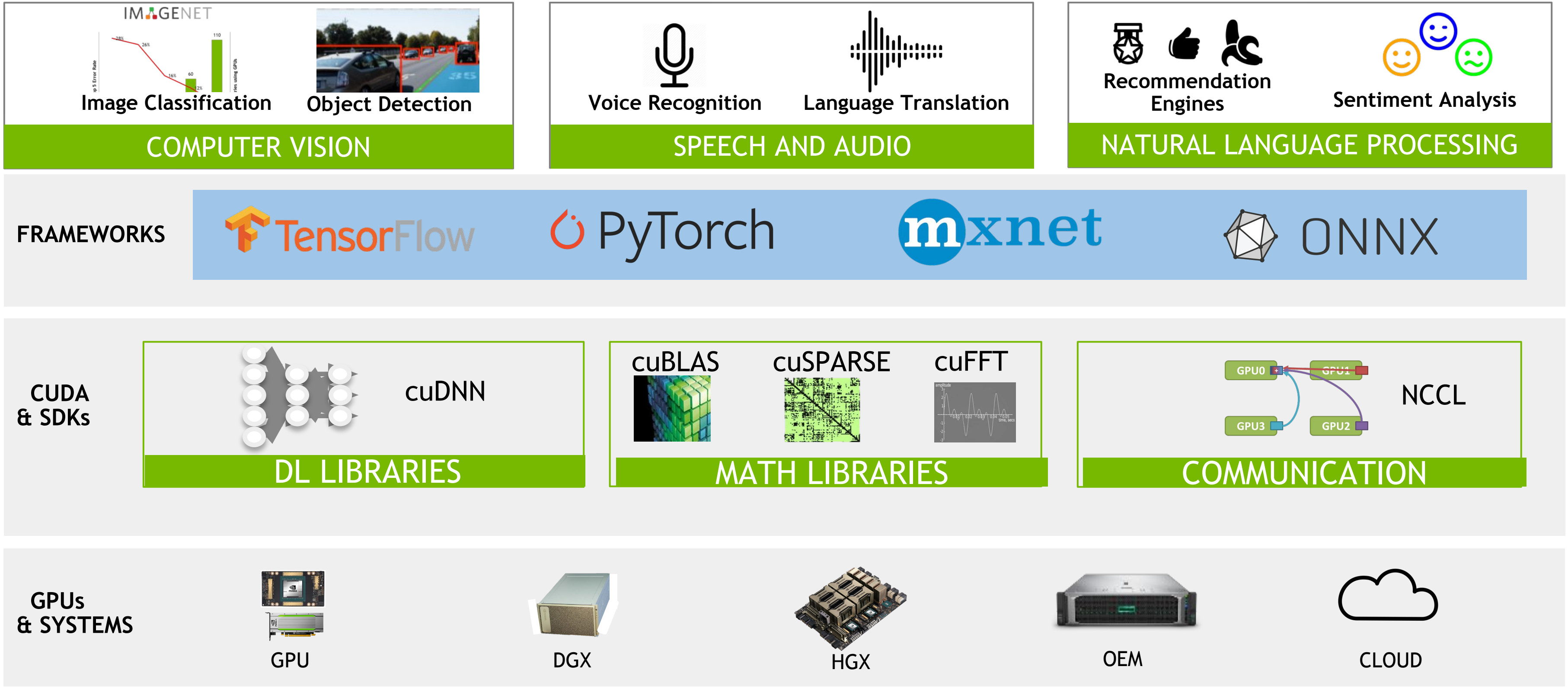


Image Classification

Object Detection

COMPUTER VISION

Voice Recognition

Language Translation

SPEECH AND AUDIO

Recommendation Engines

Sentiment Analysis

NATURAL LANGUAGE PROCESSING

FRAMEWORKS

TensorFlow

PyTorch

mxnet

ONNX

CUDA & SDKs

cuDNN

DL LIBRARIES

cuBLAS

cuSPARSE

cuFFT

MATH LIBRARIES

NCCL

COMMUNICATION

GPUs & SYSTEMS

GPU

DGX

HGX

OEM

CLOUD

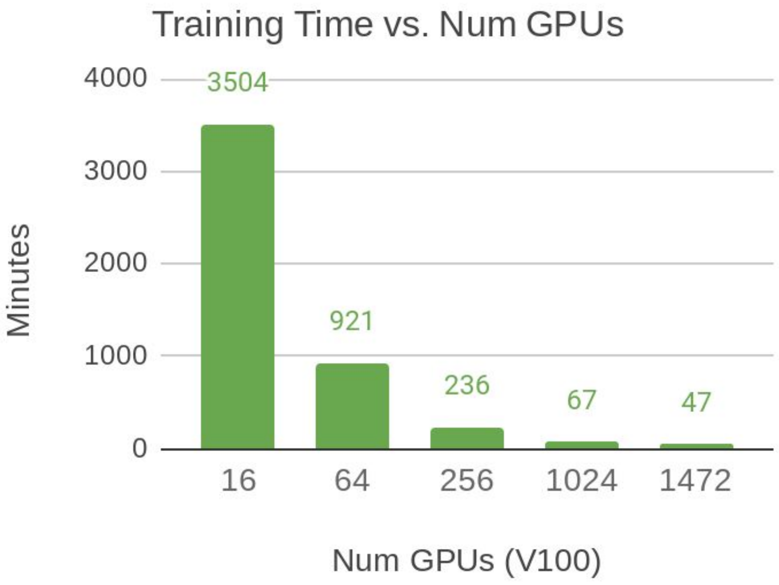
Trends: Big Data, Larger Models

- Data and model sizes increasing

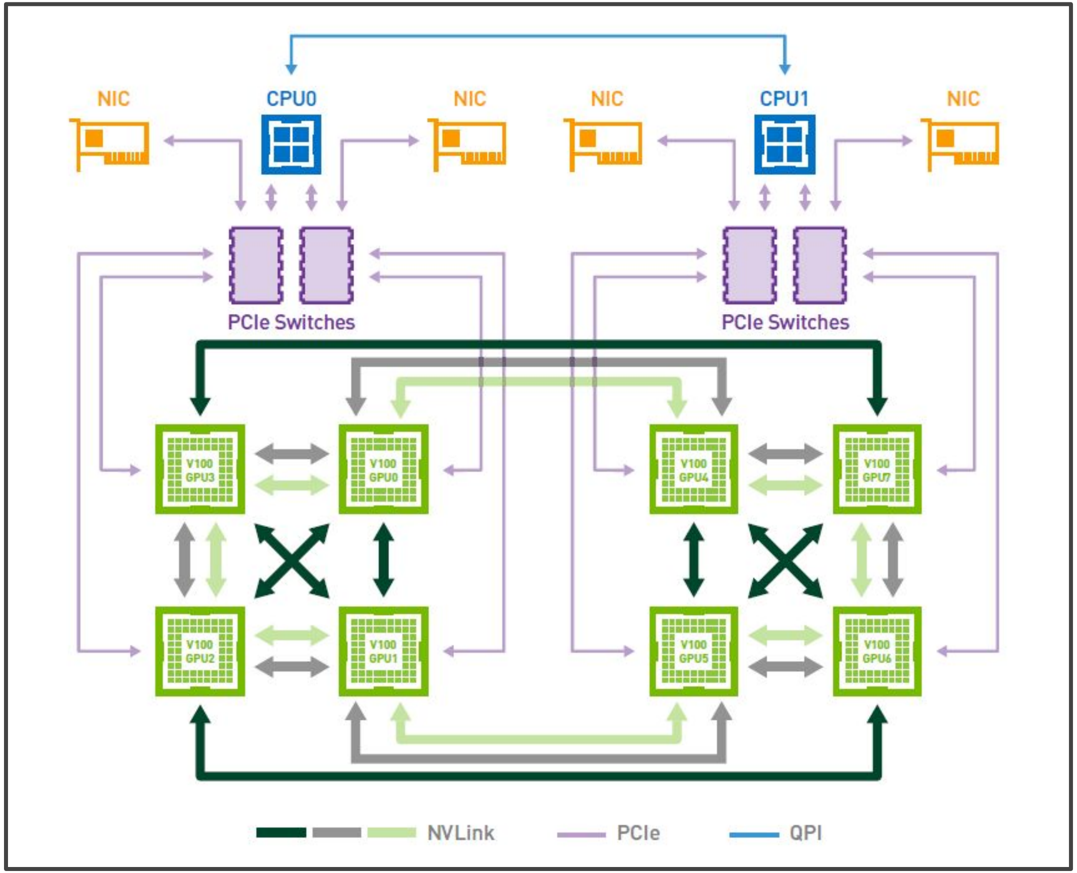
- BERT NLP: 110M, 330M params
  - Recent: 8B, 17B

- Strong demand for multi-gpu jobs
  - Larger problems
  - Faster turn-around
  - E.g., 128-GPUs per job

- ML Perf training results



Sample Multi-GPU Node: DGX-1



Single DGX-1 Node:

8 Nvidia V100 GPUs

Dual socket, NVLink in node

4 Mlnx EDR NICs (100 Gbps)

Dual 10GbE ports

Connecting Multiple nodes:

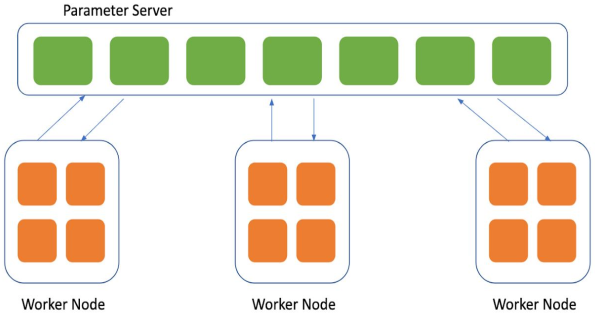
Infiniband or RoCE

4 x NICs to connected fabric

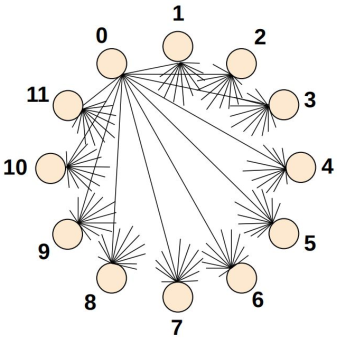
Distributed Training Applications

Multi-GPU, Multi-node

- Data / model parallelism
- Stochastic gradient descent (SGD)
- Async SGD: Parameter-server
- Sync SGD: All-reduce
  - NCCL / MPI
  - Utilize fast interconnects / RDMA
  - Horovod library
- Distributed TensorFlow / PyTorch



Parameter Server



All Reduce

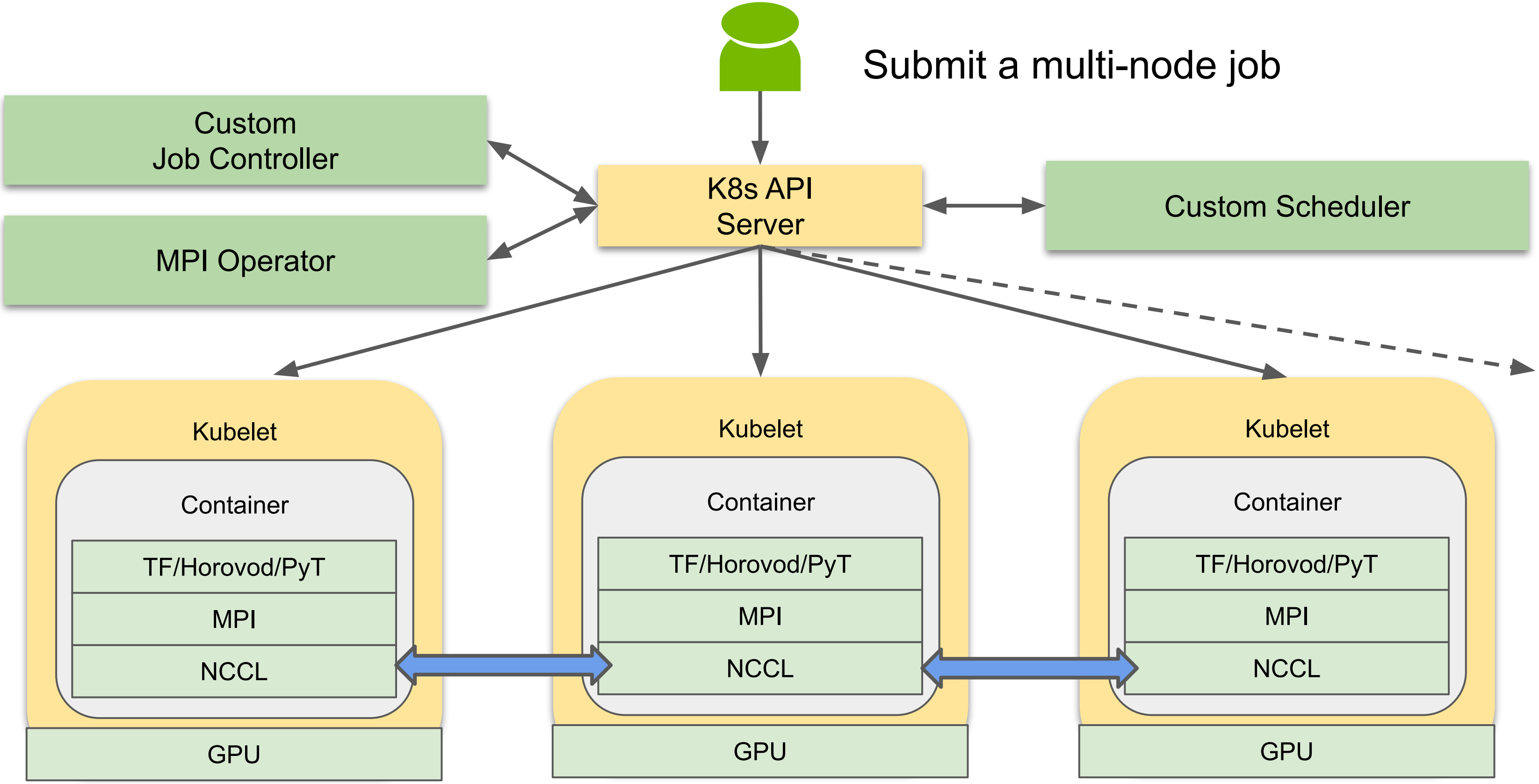
K8s Challenges & Outline

Multi-node and K8s Gaining Traction

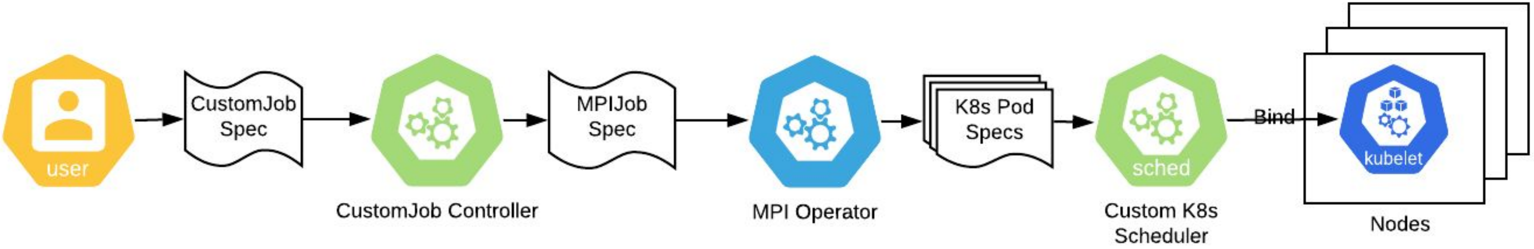
- Motivation & background
- End-to-end flow: Array jobs
    - MPI & job lifecycle
- Gang scheduling
- Multi-rail RDMA / CNI
- Application / BERT

- Production shared clusters
- Quotas, queues, time limits
- Backfilling, utilization
- Monitoring / Operations
    - Dashboards / CICD
- Conclusions / Future work

K8s Orchestration Flow



Sample PyTorch Job Launch



Sample Dist PyTorch job launch

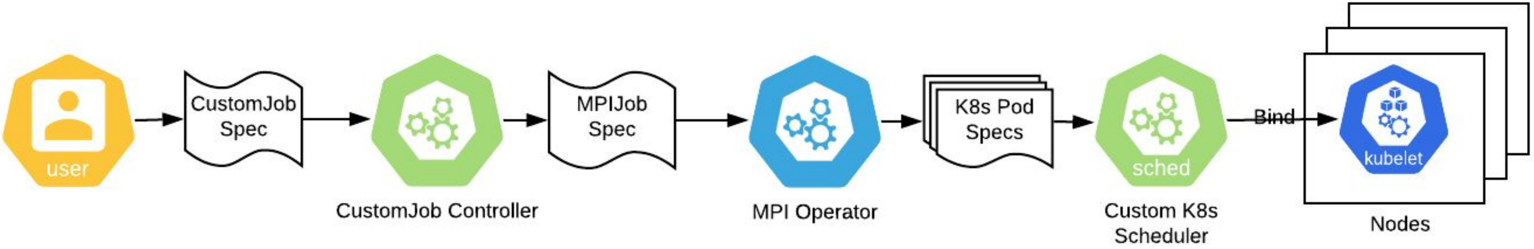
```
python -m torch.distributed.launch
--nproc_per_node=8 --nnodes=2 --node_rank=0
--master_addr=localhost bert_train.py <args>

python -m torch.distributed.launch
--nproc_per_node=8 --nnodes=2 --node_rank=1
--master_addr=<ip> bert_train.py <args>
```

K8s Multi-node PyTorch job launch

```
mpirun -np $ARRAY_SIZE -npernode 1
  python -m torch.distributed.launch
  --nproc_per_node=8 --nnodes=$ARRAY_SIZE
  --node_rank=$ARRAY_INDEX
--master_addr=$MASTER_IP bert_train.py <args>
nvrun ..
```

mpirun as launcher w/ NCCL backend

Array Jobs and MPI Operator



Array jobs

Abstraction for multi-node

Configurable type, size

Status msgs, Telemetry

```
...
Spec:
  Containers:
    Args:
      -c
      mpirun -np ${ARRAY_SIZE} -npernode 1
        python3 -m torch.distributed.launch
        --nproc_per_node=8
        --nnodes=${ARRAY_SIZE}
        --node_rank=${ARRAY_INDEX}
        --master_addr=${MASTER_IP}
        bert_train.py
    Command:
      /bin/sh
    ...
    Resources:
      Requests:
        Sriov Rdma:     4
        Nvidia.Com/Gpu: 8
    ...
```

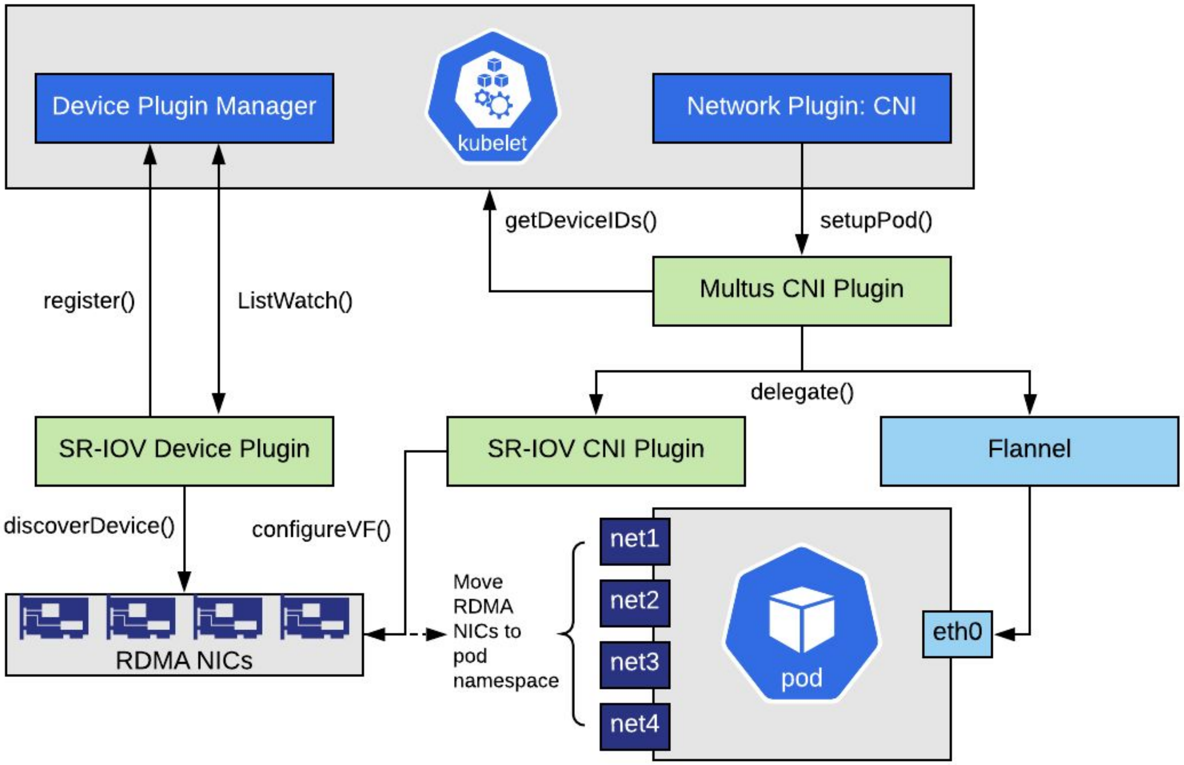
MPI Operator

Upstream in Kubeflow

Launch replicas on each node
- Kubectl exec, Lifecycle

Mods for gang scheduling

SRIOV CNI for K8s Multi-Rail



```
...
Resources:
  Requests:
    Sriov Rdma:      4
    Nvidia.Com/Gpu:  8
...
```

Exposing multiple NIC interfaces to K8s Pod

Multus delegates to SR-IOV CNI and Flannel

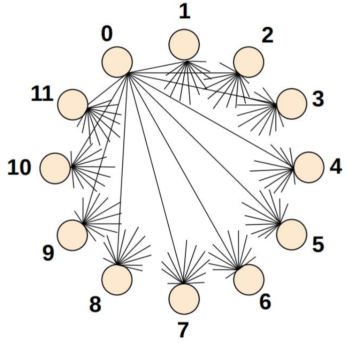
Base SR-IOV CNI from upstream w/ customizations

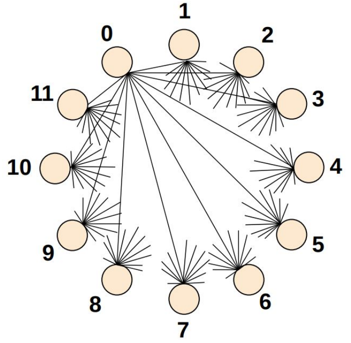
Gang Scheduling Multi-Node Pods

- Multi-node pods: All-or-none to make progress
  - Default K8s: pods one-by-one ⇒ deadlocks

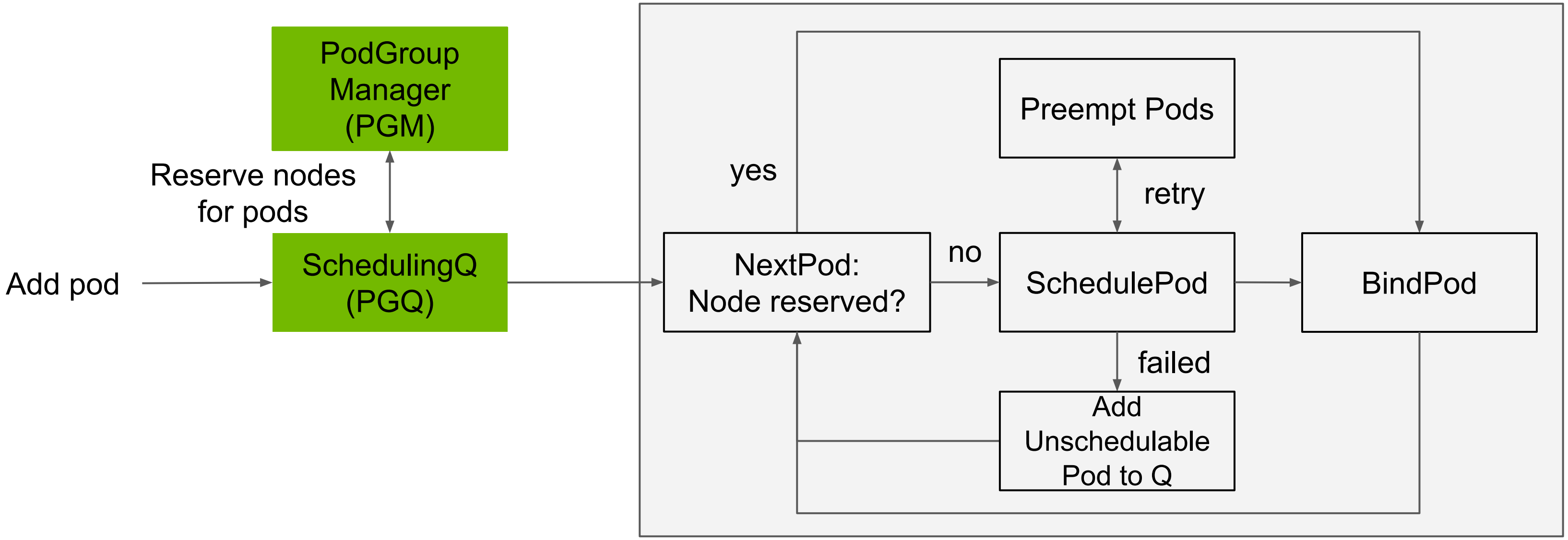
- Gang / co-scheduling in K8s:
  - Open item for default K8s scheduler (since 2015)
  - Basic: loop over pods → wait → timeout → release
  - Being considered via Volcano, Poseidon, etc.

- Approach: PodGroup structure
  - Full node pods only, reservation based





PodGroup Queue and Manager



PodGroup Manager (PGM)

Reserve nodes for pods

Add pod

SchedulingQ (PGQ)

yes

NextPod: Node reserved?

no

Preempt Pods

retry

SchedulePod

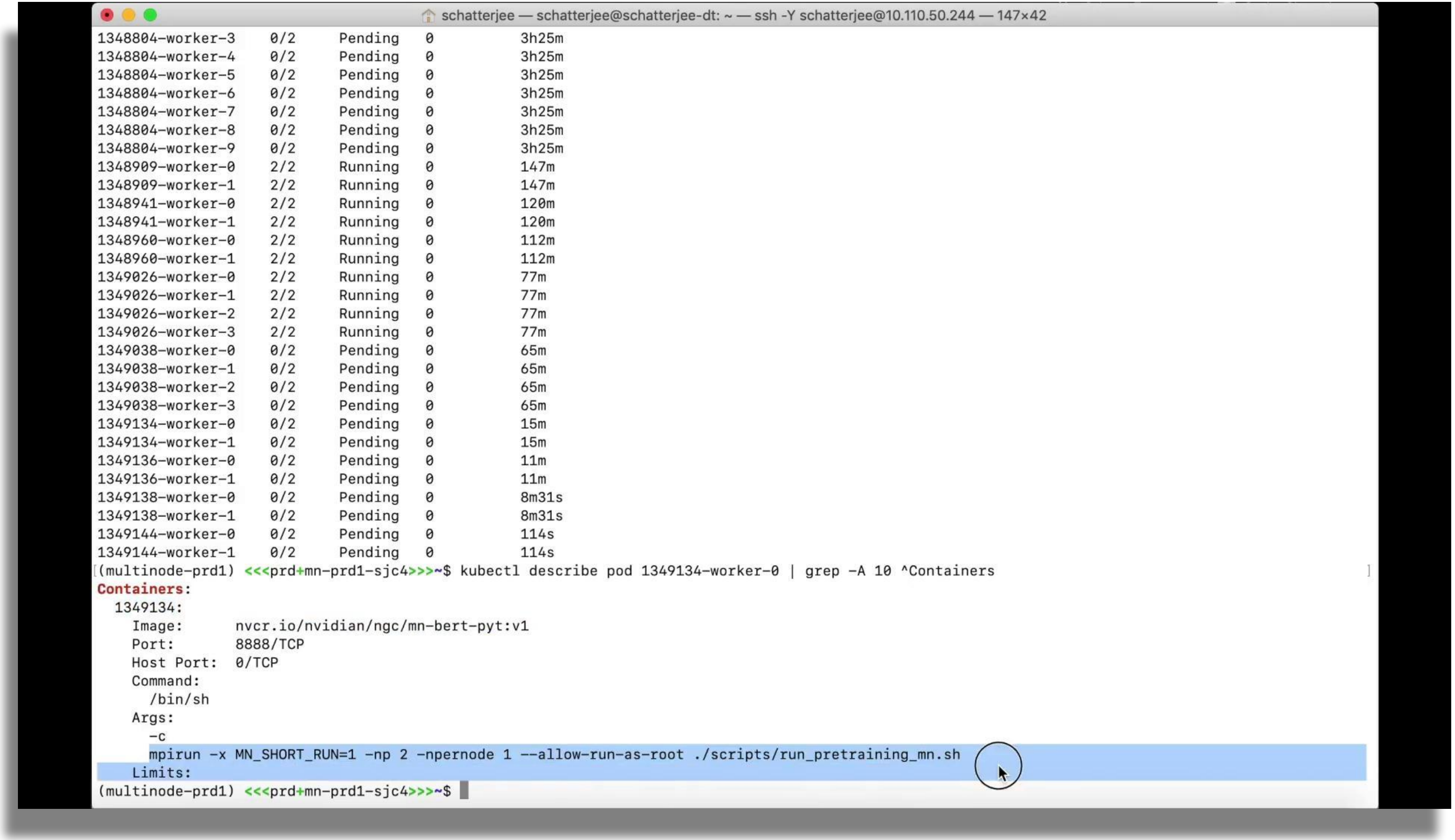
failed

Add Unschedulable Pod to Q

BindPod

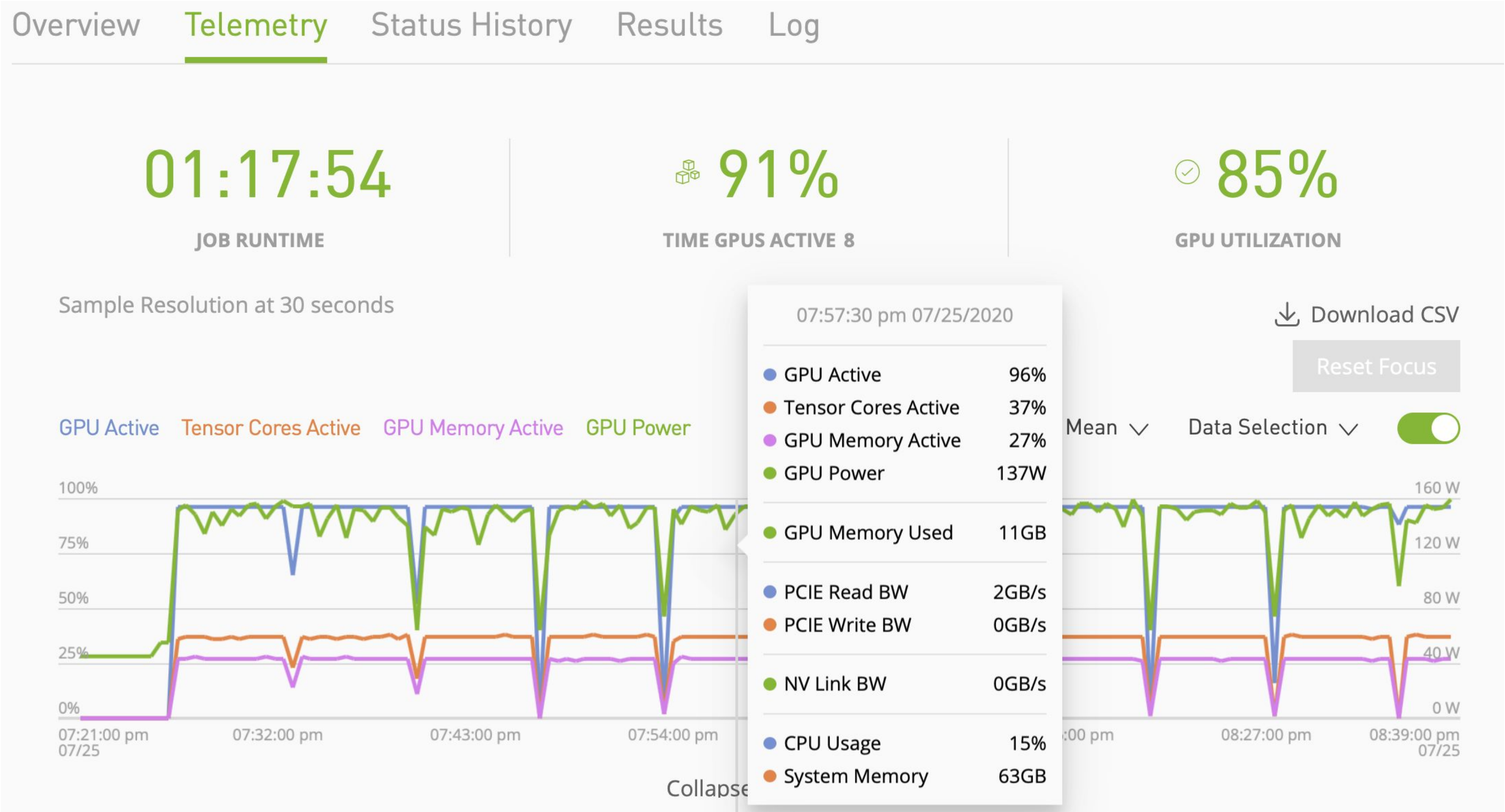
- Experimental extensions to K8s
- Reserve nodes for full-node pods

Demo


schatterjee — schatterjee@schatterjee-dt: ~ — ssh -Y schatterjee@10.110.50.244 — 147×42

```
1348804-worker-3    0/2     Pending   0          3h25m
1348804-worker-4    0/2     Pending   0          3h25m
1348804-worker-5    0/2     Pending   0          3h25m
1348804-worker-6    0/2     Pending   0          3h25m
1348804-worker-7    0/2     Pending   0          3h25m
1348804-worker-8    0/2     Pending   0          3h25m
1348804-worker-9    0/2     Pending   0          3h25m
1348909-worker-0    2/2     Running   0          147m
1348909-worker-1    2/2     Running   0          147m
1348941-worker-0    2/2     Running   0          120m
1348941-worker-1    2/2     Running   0          120m
1348960-worker-0    2/2     Running   0          112m
1348960-worker-1    2/2     Running   0          112m
1349026-worker-0    2/2     Running   0          77m
1349026-worker-1    2/2     Running   0          77m
1349026-worker-2    2/2     Running   0          77m
1349026-worker-3    2/2     Running   0          77m
1349038-worker-0    0/2     Pending   0          65m
1349038-worker-1    0/2     Pending   0          65m
1349038-worker-2    0/2     Pending   0          65m
1349038-worker-3    0/2     Pending   0          65m
1349134-worker-0    0/2     Pending   0          15m
1349134-worker-1    0/2     Pending   0          15m
1349136-worker-0    0/2     Pending   0          11m
1349136-worker-1    0/2     Pending   0          11m
1349138-worker-0    0/2     Pending   0          8m31s
1349138-worker-1    0/2     Pending   0          8m31s
1349144-worker-0    0/2     Pending   0          114s
1349144-worker-1    0/2     Pending   0          114s
(multinode-prd1) <<<prd+mn-prd1-sjc4>>>~$ kubectl describe pod 1349134-worker-0 | grep -A 10 ^Containers
Containers:
  1349134:
    Image:       nvcr.io/nvidian/ngc/mn-bert-pyt:v1
    Port:        8888/TCP
    Host Port:   0/TCP
    Command:
      /bin/sh
    Args:
      -c
      mpirun -x MN_SHORT_RUN=1 -np 2 -npernode 1 --allow-run-as-root ./scripts/run_pretraining_mn.sh
    Limits:
(multinode-prd1) <<<prd+mn-prd1-sjc4>>>~$
```

Sample Job Real-Time Telemetry



01:17:54

JOB RUNTIME

91%

TIME GPUS ACTIVE 8

85%

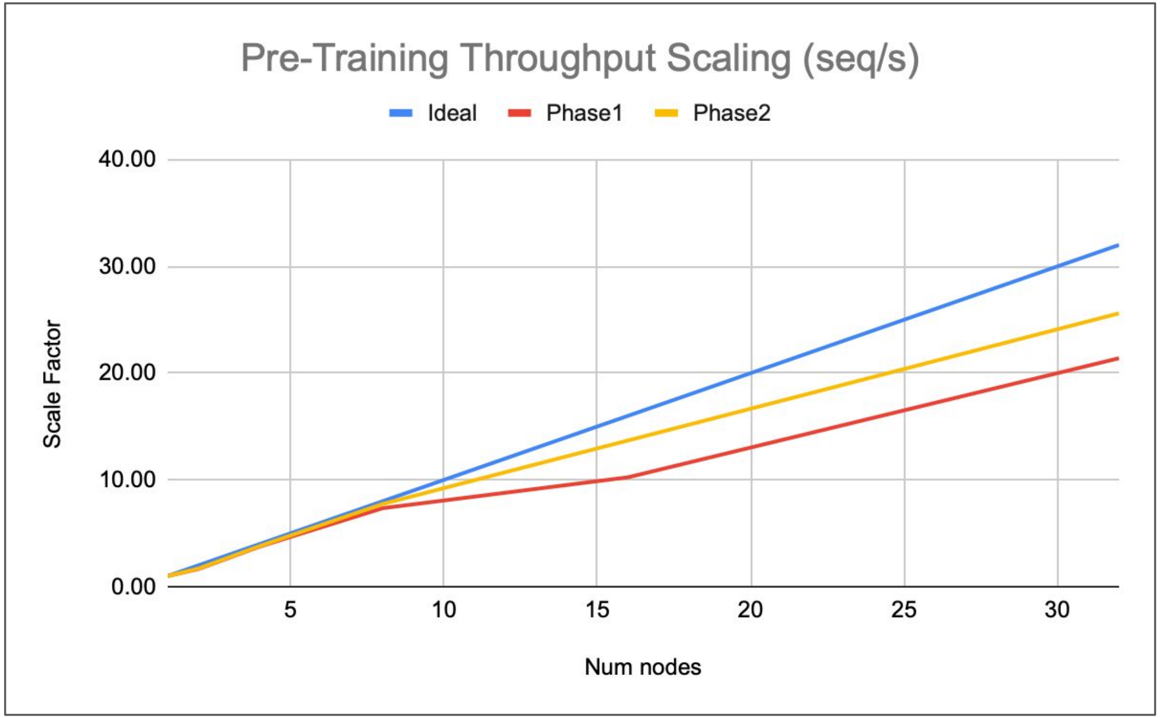
GPU UTILIZATION

Sample Resolution at 30 seconds

07:57:30 pm 07/25/2020

● GPU Active	96%
● Tensor Cores Active	37%
● GPU Memory Active	27%
● GPU Power	137W
● GPU Memory Used	11GB
● PCIE Read BW	2GB/s
● PCIE Write BW	0GB/s
● NV Link BW	0GB/s
● CPU Usage	15%
● System Memory	63GB

Sample BERT K8s Scaling



BERT Phase 1: batch_size_per_gpu = 64, seq length = 128

Phase 2: batch_size_per_gpu = 16, seq length = 512

K8s Challenges & Outline

- Motivation & background
- End-to-end flow: Array jobs
  - MPI & job lifecycle
- Gang scheduling
- Multi-rail RDMA / CNI
- Application / BERT

- Production shared clusters
- Quotas, queues, time limits
- Backfilling, utilization
- Monitoring / Operations
  - Dashboards / CICD
- Conclusions / Future work

Shared K8s Cluster for Multi-node

- Production on-prem cluster
  - Early internal users
  - 100 DGX nodes, single hop

- Quotas - Concurrent GPU usage
  - Configurable per user, default

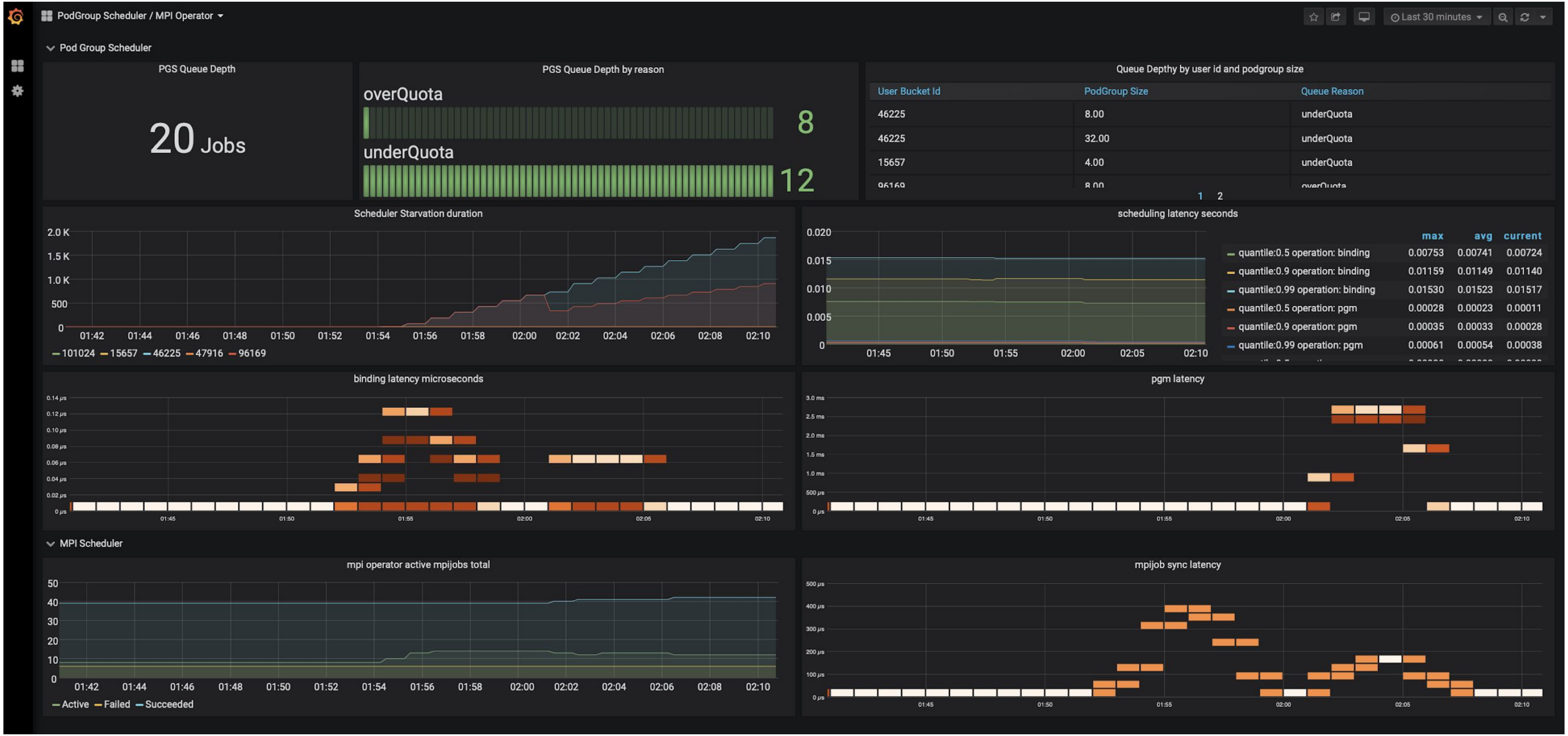
- Time limits
  - E.g. 128 node-hours
  - ⇒ 8 hours for 16-node job

- Starvation handling, backfilling
  - Blocking for large mn jobs
  - Backfilling for utilization

- Dynamic job priority
  - DRF fairness
  - Starvation, age etc
  - Weighted function

- Operations / dashboards

Scheduler Dashboard



Queue Depthy by user id and podgroup size

User Bucket Id	PodGroup Size	Queue Reason
46225	8.00	underQuota
46225	32.00	underQuota
15657	4.00	underQuota
96169	8.00	overQuota

1 2

scheduling latency seconds

	max	avg	current
— quantile:0.5 operation: binding	0.00753	0.00741	0.00724
— quantile:0.9 operation: binding	0.01159	0.01149	0.01140
— quantile:0.99 operation: binding	0.01530	0.01523	0.01517
— quantile:0.5 operation: pgm	0.00028	0.00023	0.00011
— quantile:0.9 operation: pgm	0.00035	0.00033	0.00028
— quantile:0.99 operation: pgm	0.00061	0.00054	0.00038

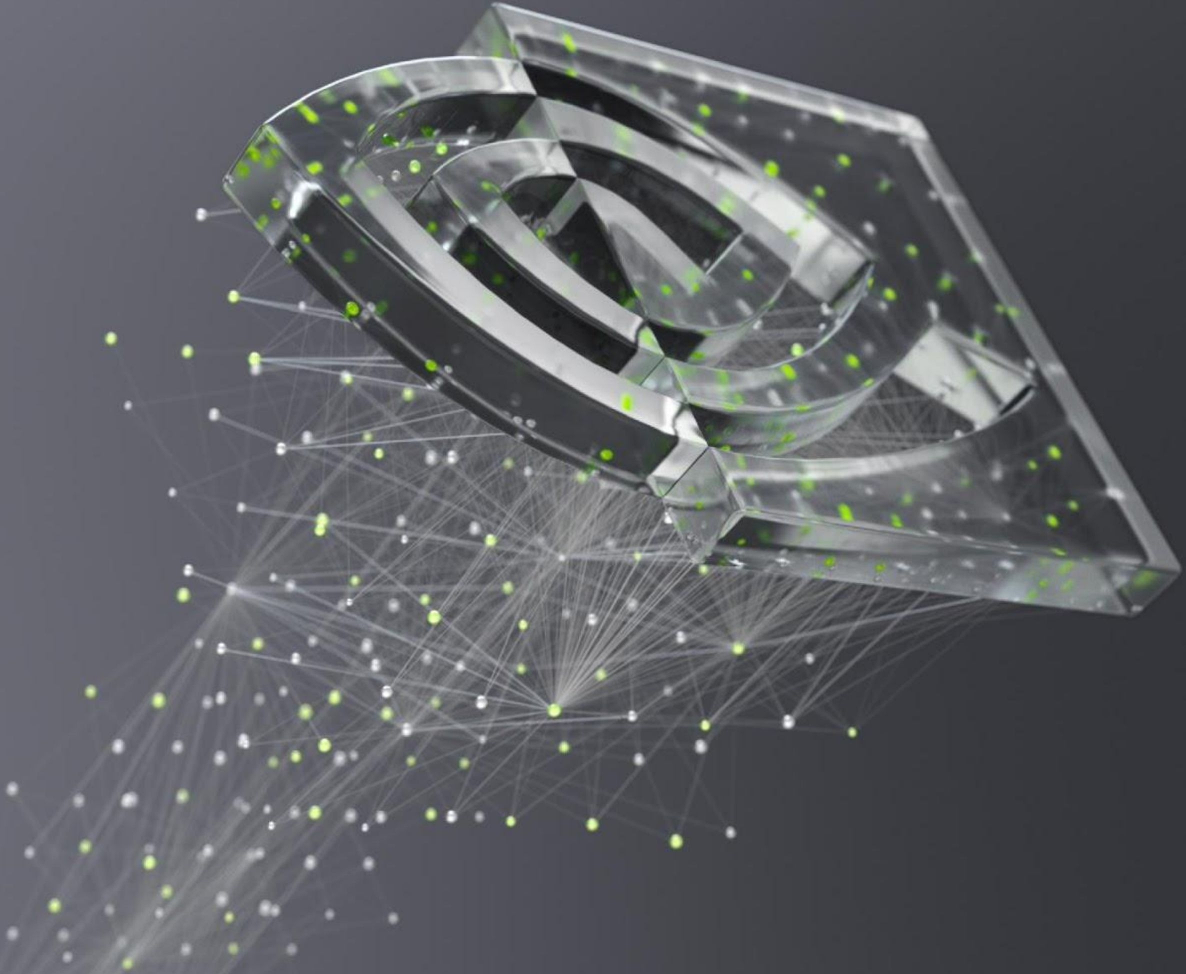
Summary and Future Work

- Multi-node clusters with K8s
  - Gang scheduling, GPUs, MPI, RDMA

- MN enabled containers / models
  - Available from ngc.nvidia.com

- Ongoing work
  - Production hardening
  - Performance, Storage caching
  - Other array types, K8s framework

- Acknowledgements



NVIDIA