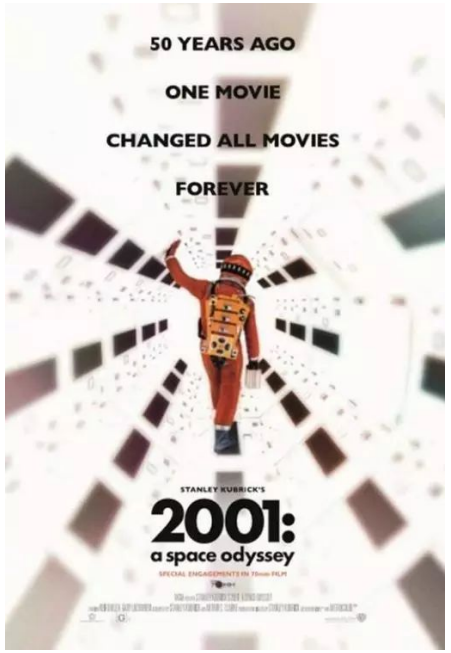
HAL: Computer System for Scalable Deep Learning

Volodymyr Kindratenko, Dawei Mu, Yan Zhan, John Maloney, Sayed Hadi Hashemi, Ben Rabe, Ke Xu, Roy Campbell, William Gropp, Jian Peng

ILLINOIS
NCSA | National Center for Supercomputing Applications

Introduction

- In 2017, NCSA was funded by the NSF's Major Research Instrumentation program.
- The goal is to develop and deploy a computational "instrument" for supporting deep learning applications at scale.
- The machine was named as Hardware Accelerated Learning (HAL) cluster.
- It became operational in March 2019.

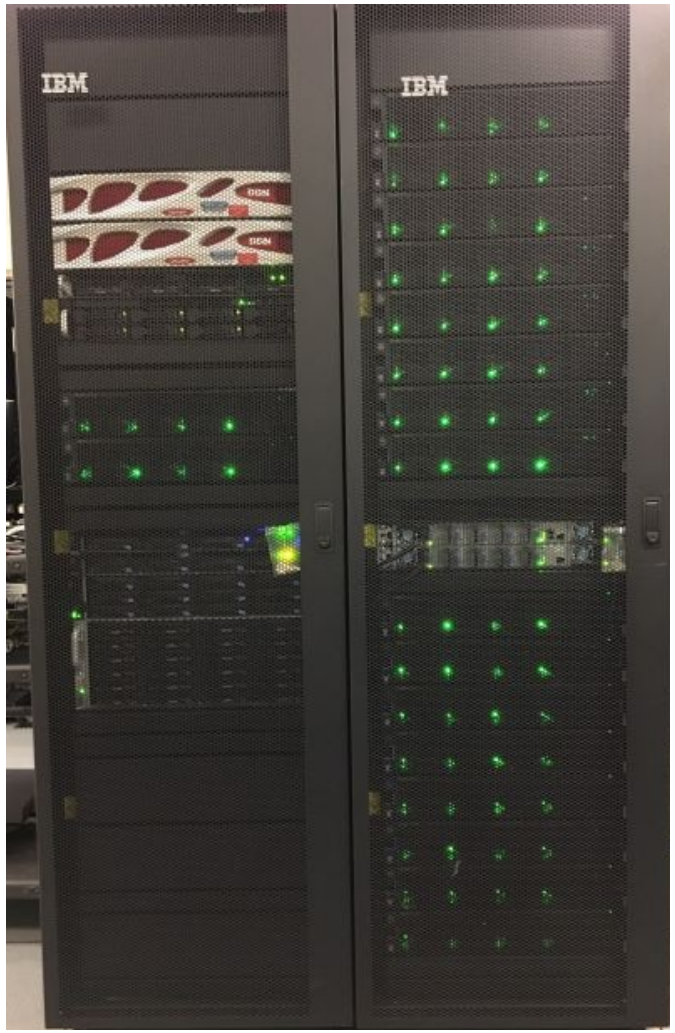


System Requirements

- Performance Requirements
  - Many modern DL frameworks have been optimized for NVIDIA GPUs
  - Fast cross-GPUs bandwidth delivered by NVLink 2.0 interconnection
  - Storage and node interconnects support simultaneous data processing
- Usability Requirements
  - Traditional HPC job submission and resource scheduling which access the system via ssh through the command line interface
  - Interactive apps like Jupyter Notebook through web-based interface.
  - Customizable environments via modules, containers, and isolation of Python environments within the user space.

HAL System Design

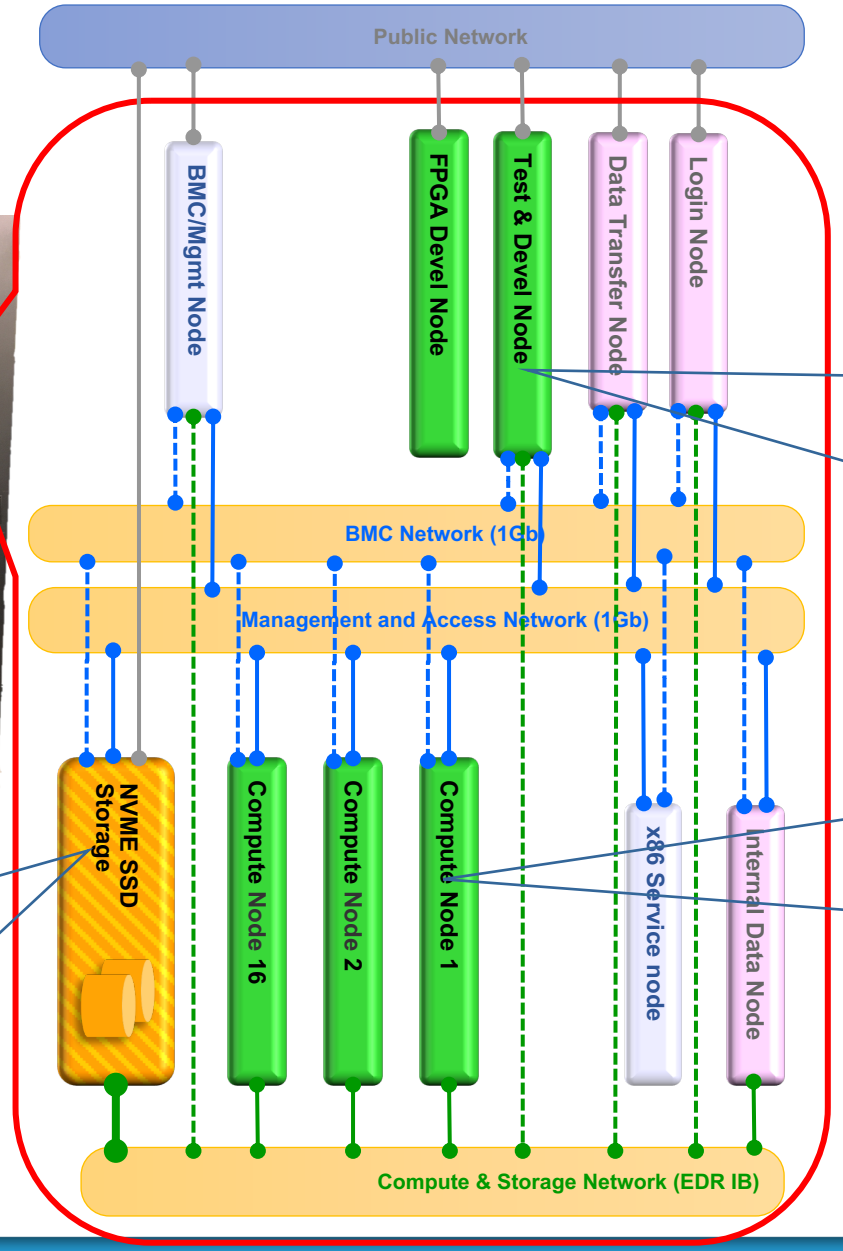
- Hardware Selection
  - Login Node
    - IBM LC921 server
  - Storage Node
    - IBM LC922 server (initial)
    - DDN GS400NVE flash arrays server (final)
  - Compute Node
    - IBM POWER9 AC922 server with NVLink 2.0 interconnected V100 GPUs (compute node)
  - Network
    - Mellanox EDR InfiniBand



HAL cluster

Hardware Accelerated Learning

- CentOS 7.7
- CUDA 10.2, cuDNN 7.6.5, NCCL 2.5.6
- IBM XLC 16.1.1, IBM XLFORTRAN 16.1.1
- Advance toolchain for Linux on Power 12.0
- IBM Watson Machine Learning Community Edition 1.7.0 (TensorFlow, PyTorch, RAPIDS cuML and cuDF)
- SLURM & Open OnDemand

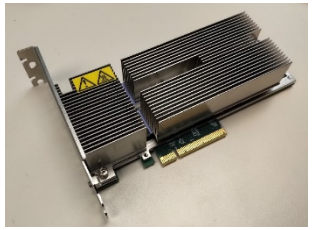


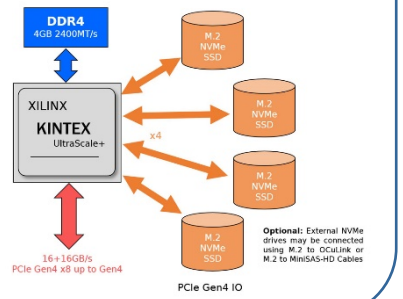
DDN GS400NVE Flash Array

- 244 TB usable
- NVME SSD-based storage
- Spectrum Scale File System

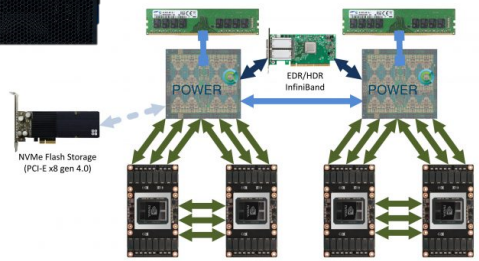
- 16 IBM AC922 nodes
  - IBM 8335-GTH AC922 server
    - 2x 20-core POWER9 CPU @ 2.4GHz
    - 256 GB DDR4
    - 4x NVIDIA V100 GPUs
      - 5120 cores
    - 16 GB HBM 2
  - 2-Port EDR 100 GB IB ConnectX-5 Adapter

Nallatech 250S+ CAPI FPGA board





IBM Power9 AC922

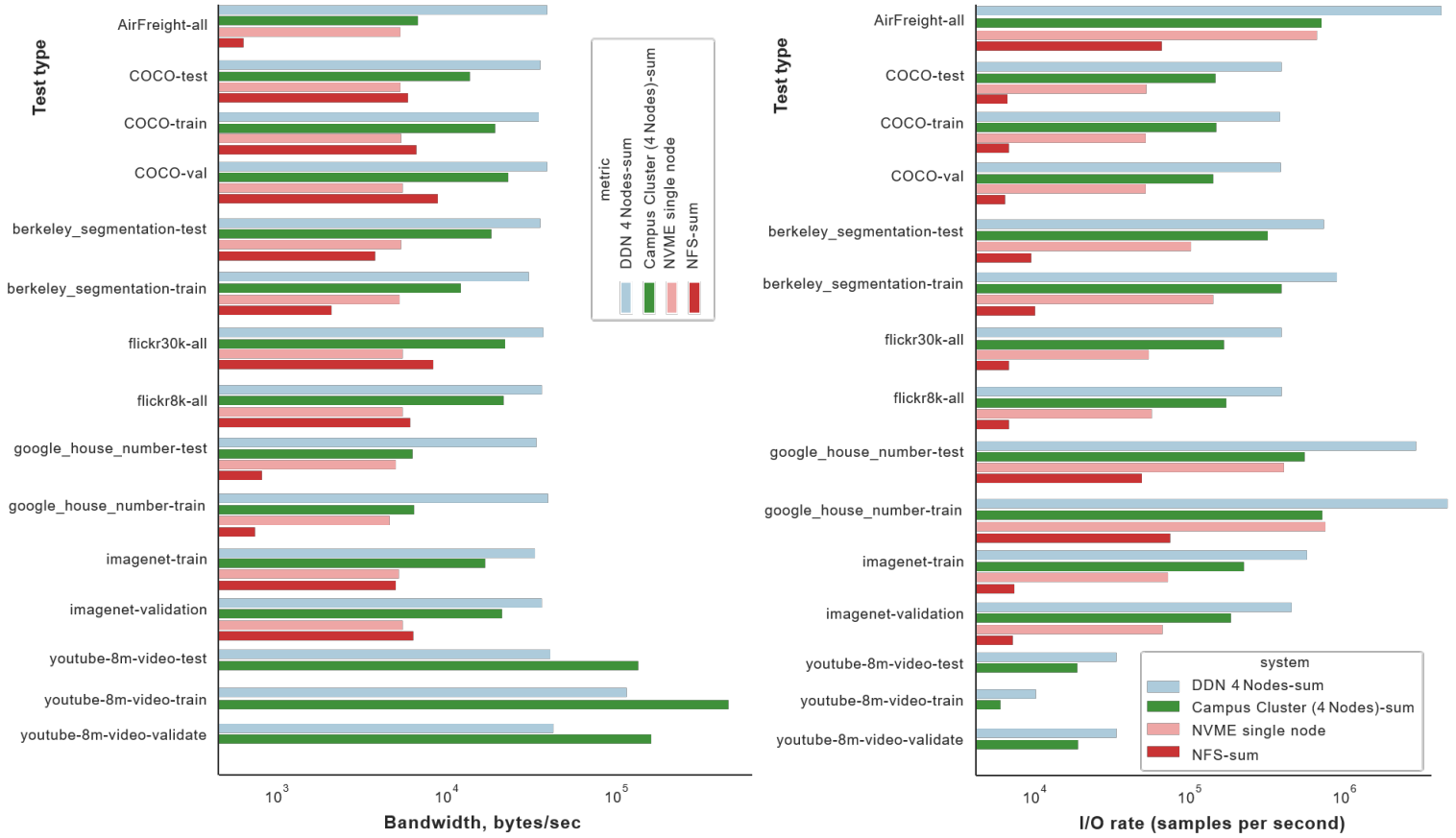


Award #1725729

ILLINOIS NCSA

HAL System Design

- Storage Selection
  - Multiple Vendors
  - Multiple Benchmarks
    - IOR
    - mdtest
    - DIOT
  - Cost per Usable TB
- Storage Tuning
  - balance between bandwidth and IOPs



HAL System Design

- Software Stack
  - OS
    - CentOS 7.7 LE ALT for POWER9
  - Environment Module
    - Lmod
  - ML/DL Tools
    - IBM Watson Machine Learning Community Edition, which includes Caffe, Tensorflow and Pytorch.
    - Both python 2.7 and 3.6/3.7 versions of WMLCE.
  - Other Software Components
    - NVIDIA CUDA tools
    - PGI compiler
    - IBM Advance toolchain
    - Jupyter Notebook and Jupyter Lab
    - Tensorboard
    - H2O AI

HAL System Design

- Software Stack
  - Slurm Wrapper Suite
    - Designed to simplify the use of the Slurm resource allocation and job submission utility.

```
srun –partition=gpu –time=24:00:00 –nodes=1 –ntasks-per-node=160 \
     –sockets-per-node=2 –cores-per-socket=20 –threads-per-core=4 \
     –mem-per-cpu=1200 –wait=0 –export=ALL –gres=gpu:v100:4 –pty /bin/bash
```

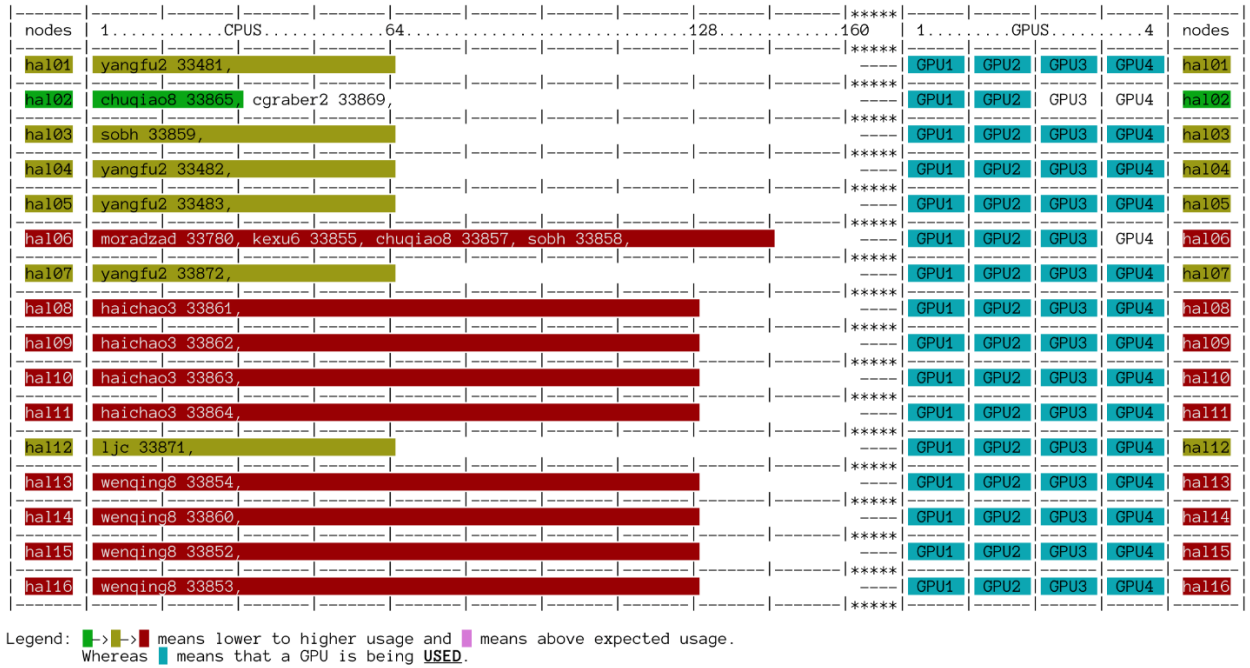
```
swrun -p gpux4 -c 40 -t 24
```

```
#!/bin/bash
#SBATCH --partition=gpu
#SBATCH --time=4:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --sockets-per-node=1
#SBATCH --cores-per-socket=4
#SBATCH --threads-per-core=4
#SBATCH --mem-per-cpu=1200
#SBATCH --export=ALL
#SBATCH --gres=gpu:v100:1

python3 mnist_train_pytorch.py
```

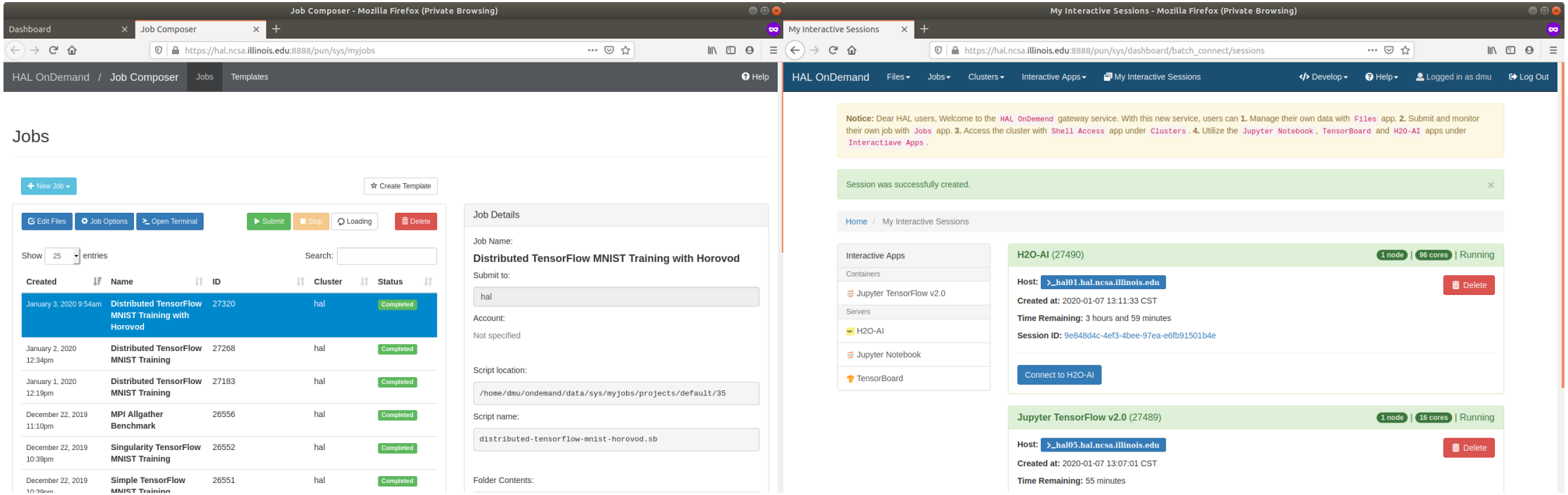
```
#!/bin/bash
#SBATCH --partition=gpux1

python3 mnist_train_pytorch.py
```



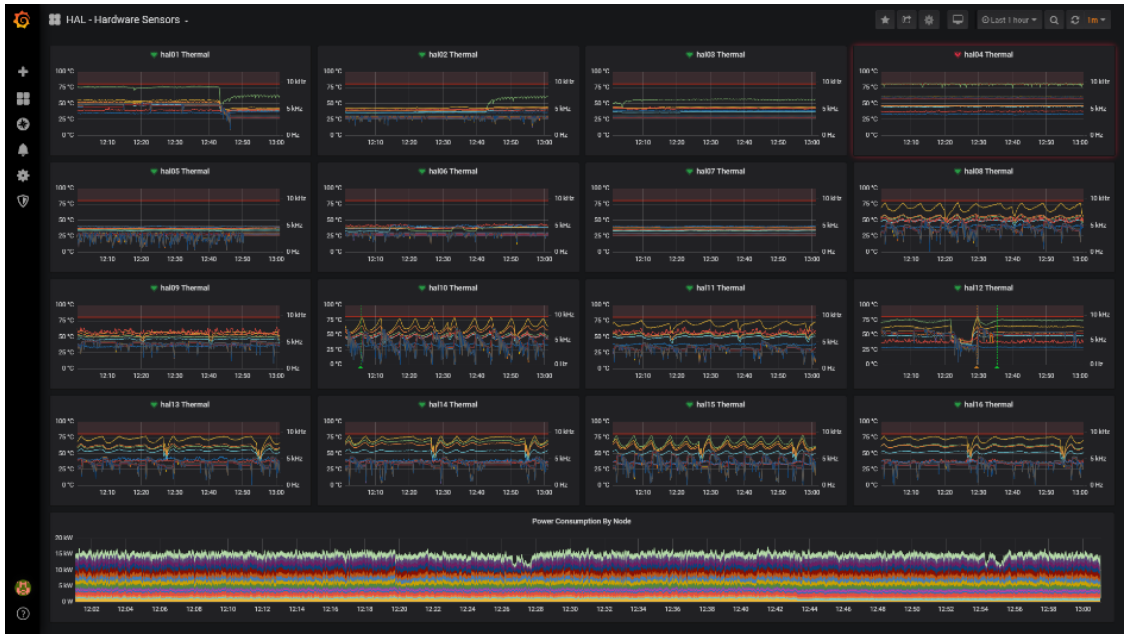
HAL System Design

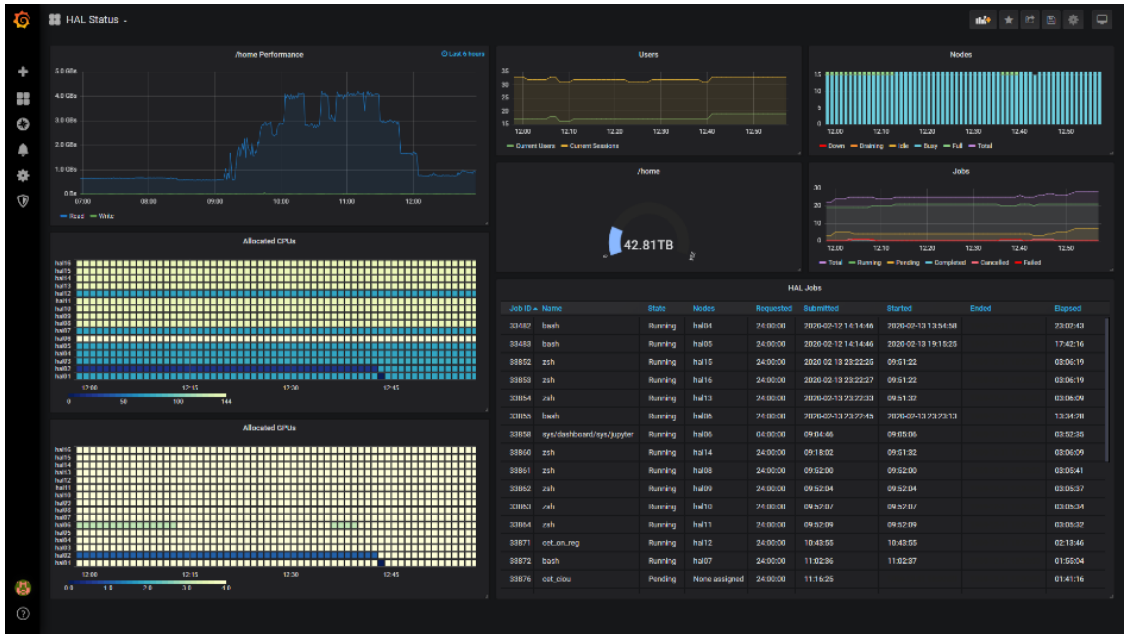
- Software Stack
  - HAL OnDemand is based on Open OnDemand Project, which provide web-based access to our HPC resources



HAL System Design

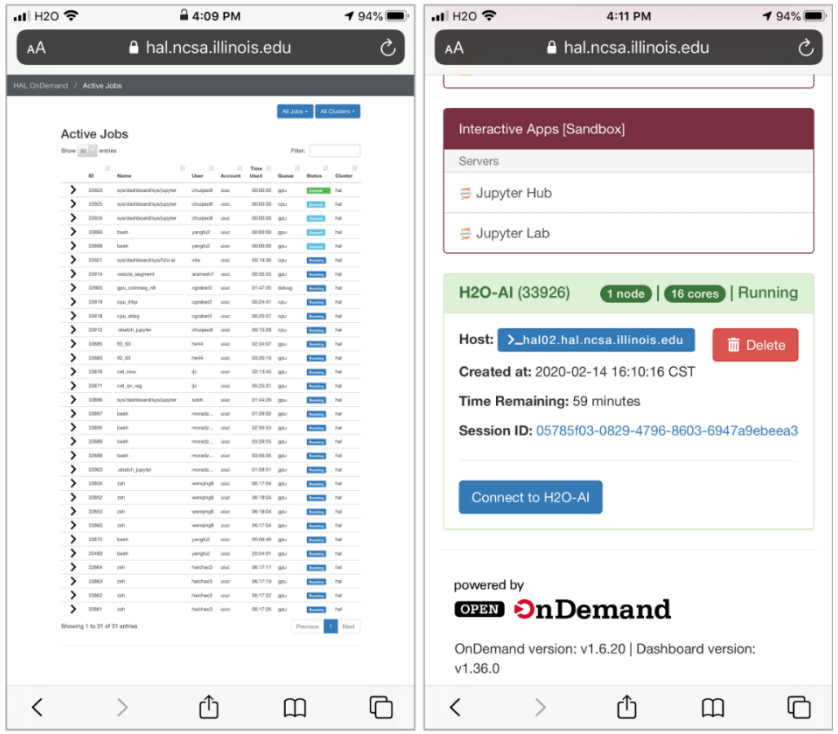
- Software Stack
  - Monitoring Stack
    - Telegraf collects metric
    - InfluxDB stores dataset
    - Grafana provides visualization





HAL System Design

- Software Stack
  - HAL System on Mobile Platforms



(a) (b)

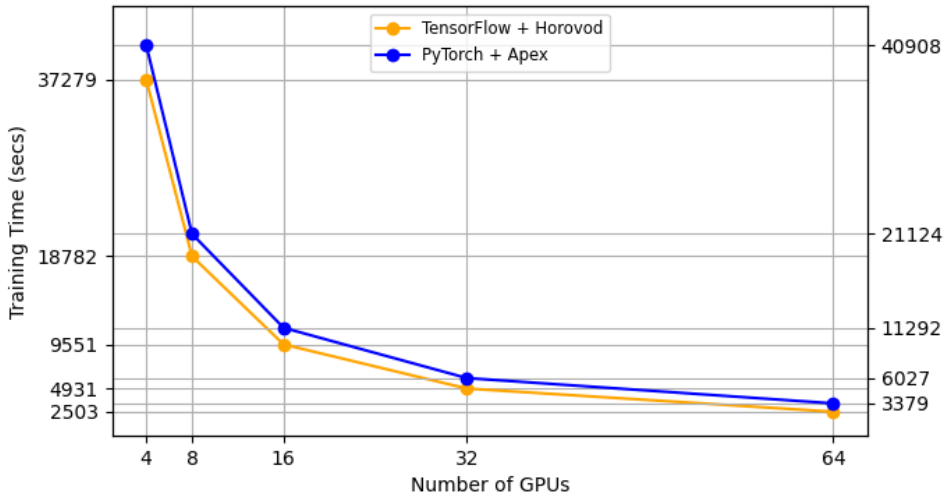
(a) (b)

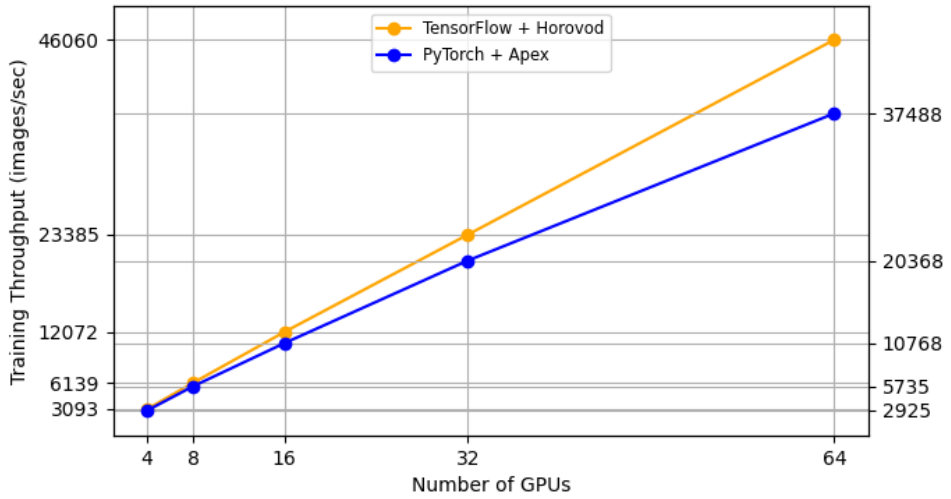
ImageNet Distributed Training Benchmark

- Implementation Details
  - Scaling ResNet-50 (TensorFlow and PyTorch) trained on ImageNet across multiple GPUs and multiple compute nodes
  - Official implementations of ResNet-50 for both frameworks
  - The optimizer utilized standard momentum with m of 0.9 and a weight decay λ of 0.0001
  - All models were trained for 90 epochs
  - Per-GPU batch size of 256 images
  - NVIDIA NCCL as communication backend
  - cnn_tf_v1.14_compatible branch and horovod for TensorFlow
  - Automatic Mixed Precision (Amp) and Distributed Data Parallel (DDP) from NVIDIA Apex for mixed-precision and distributed training for PyTorch

ImageNet Distributed Training Benchmark

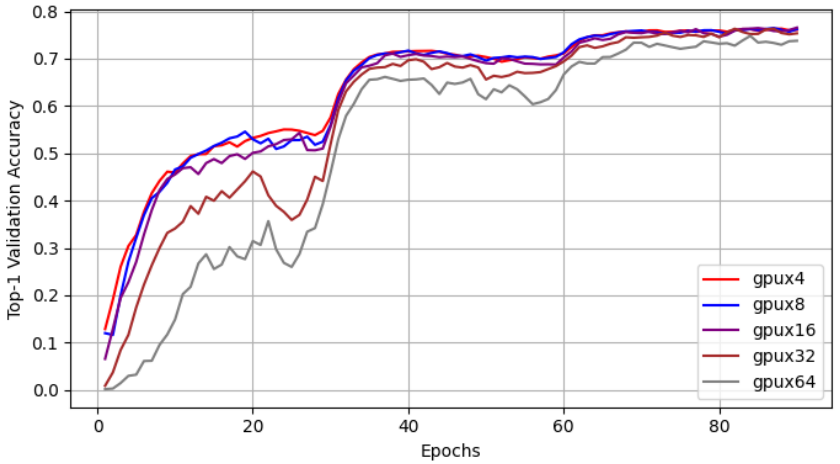
- Analysis of Results
  - With 64 GPUs across 16 compute nodes, we can train ResNet-50 in 41 mins, 43 secs with TensorFlow and 56 min, 18 sec with PyTorch, while maintaining comparable top-1 and top-5 accuracy
  - The global throughput shows that we were able to achieve near linear performance scaling in both frameworks

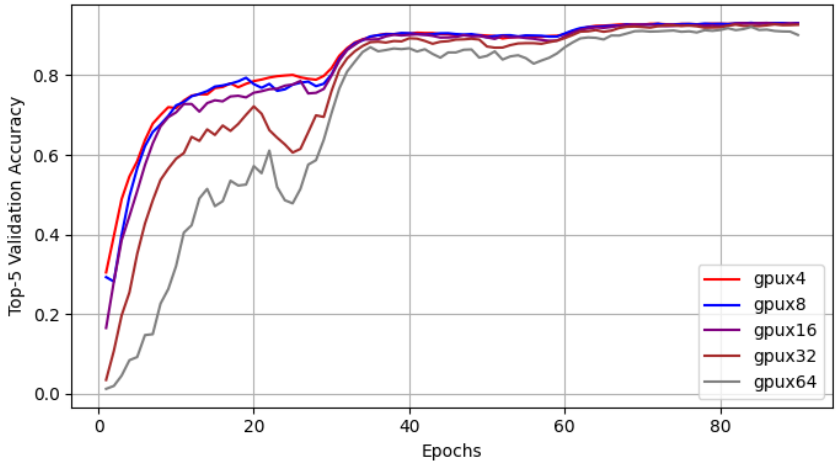


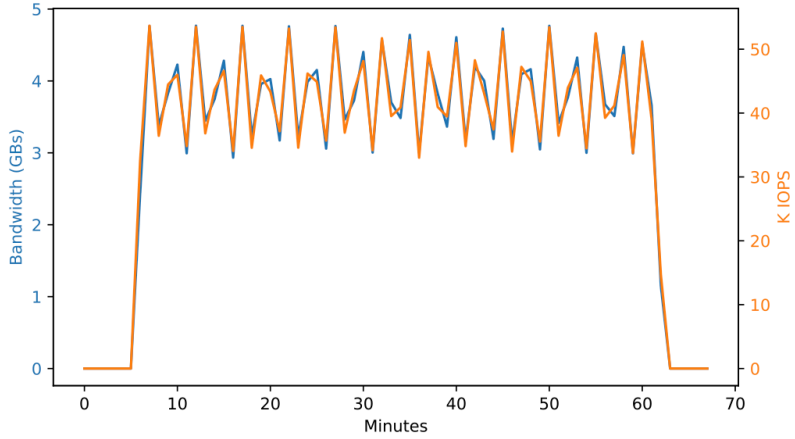


ImageNet Distributed Training Benchmark

- Analysis of Results
  - We achieve distributed training speed-up without the loss of accuracy.
  - Most experiments reached a top-1 validation accuracy of 76% in the PyTorch benchmark.
  - The Accuracy all peak at comparable accuracy near the end of training during the TensorFlow benchmark.
  - Average bandwidth is 3.84 GB/s and average IOPS is 42.95K.







Discussion and Lessons Learned

- Issued over 300 user accounts, supported over 70 faculty research groups from over 20 departments
- Over 30 publications along with 4+ Students Thesis
- Using AWS (p3.2xlarge instance) cost as a reference, HAL provides over $141,000.00 in value every month.
- Provide regular training sessions for new users
- Organize hackathons where students can work on various problems that require building and training DNN models.

Acknowledgments

- This work is supported by the National Science Foundation's Major Research Instrumentation program, grant No. 1725729, as well as the University of Illinois at Urbana-Champaign.
- We would like to thank Daniel Lapine from NCSA, Steve Zehner, Trish Froeschle, Thomas Prokop, Jamie Syptak, and Terry Leatherland from IBM for their expert advise and invaluable contributions to the development and operation of the system.
- We would also like to thank our student intern Nishant Dash for his contribution to the development of swsuite software.

ILLINOIS
NCSA | National Center for
Supercomputing Applications