Improving GPU Utilization for multi-tenant deep learning workloads on DGX and cloud platforms

Joongi Kim @ Lablup Inc.

2019. 7. 2
NVIDIA AI Conference

Make AI Accessible

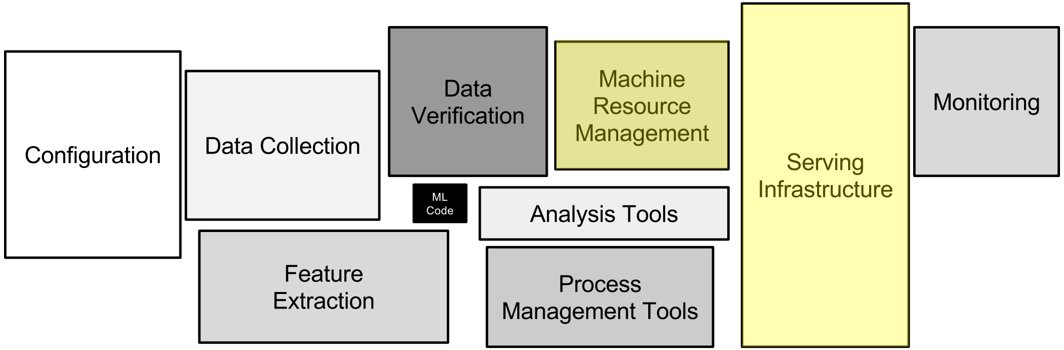
Why is it so difficult to build up a DL system?

- Open source + cloud computing = everything done?

Hidden Technical Debt in Machine Learning Systems

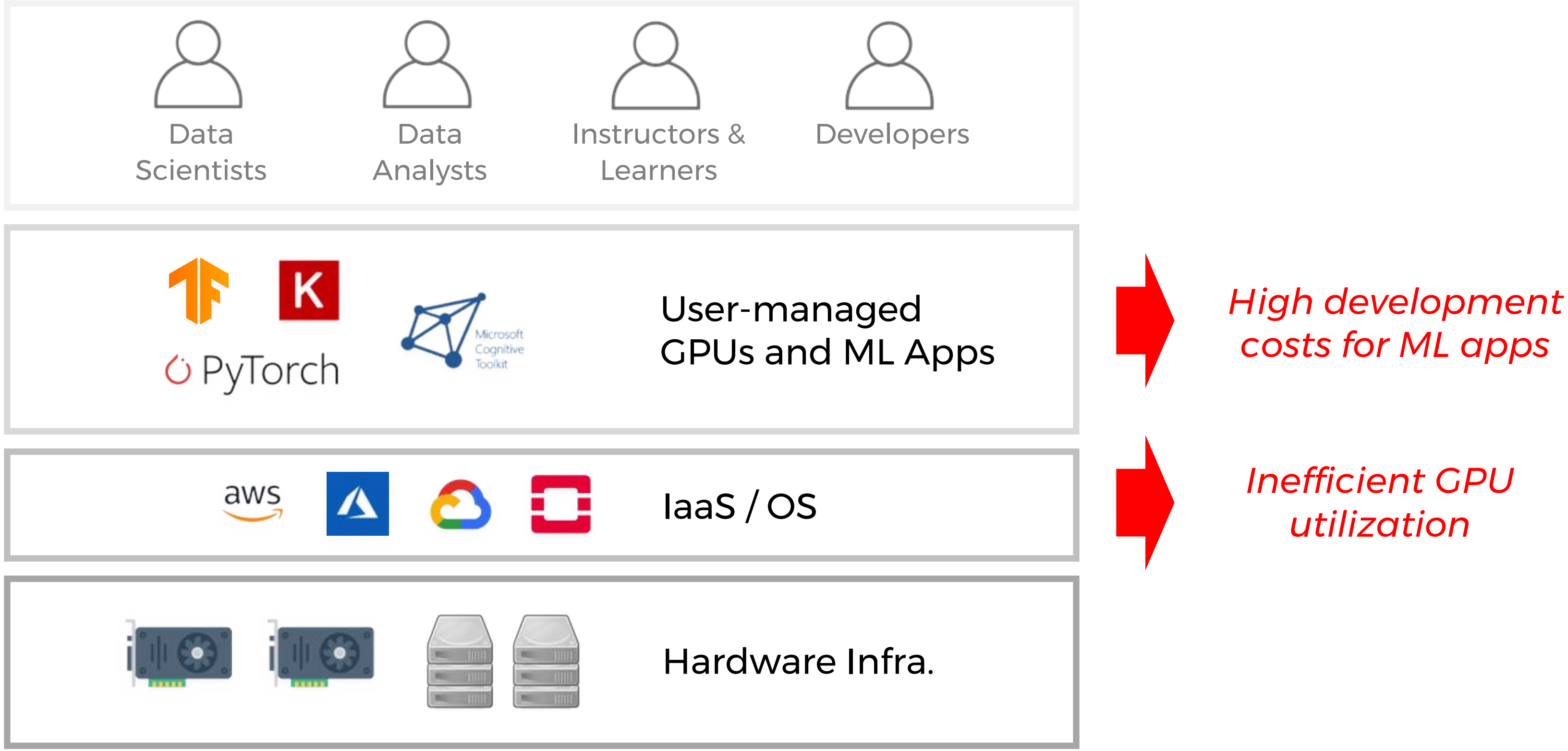
D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.

Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison
{ebner,vchaudhary,mwyoung,jfcrespo,dennison}@google.com
Google, Inc.

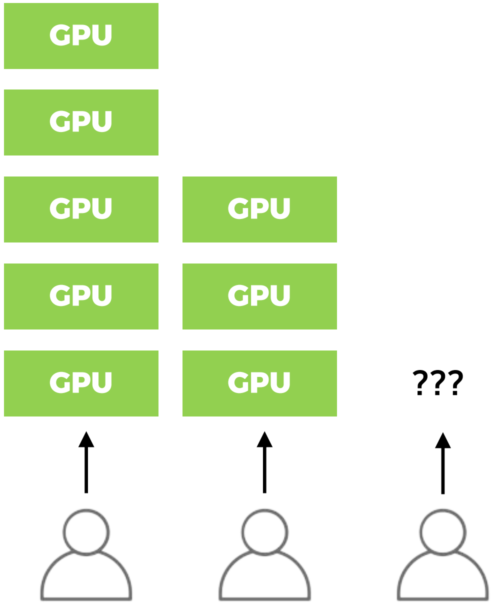


"Only a small fraction of real-world ML systems is composed of ML code"

Typical GPU Computing Stack

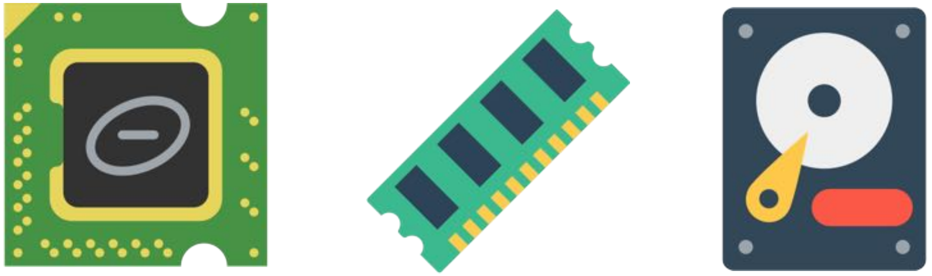


GPU management is difficult!

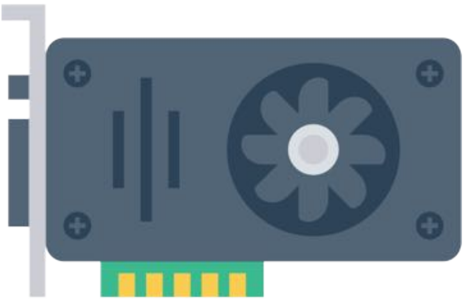


- Manual assignment of GPUs for researchers
- Both idle & insufficient at the same time
- Manual checks for SW compatibilities

Root cause of GPU mgmt. difficulties



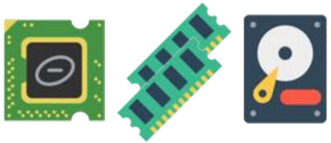
The OS knows
how to **partition**, **share**, and **schedule**
via standardized HW interfaces.



🤔

Lack of flexible GPU resource management

- Resource management / sharing technology is limited (as a peripheral device)



CPU / RAM / Disk

Process	Page
Thread	File

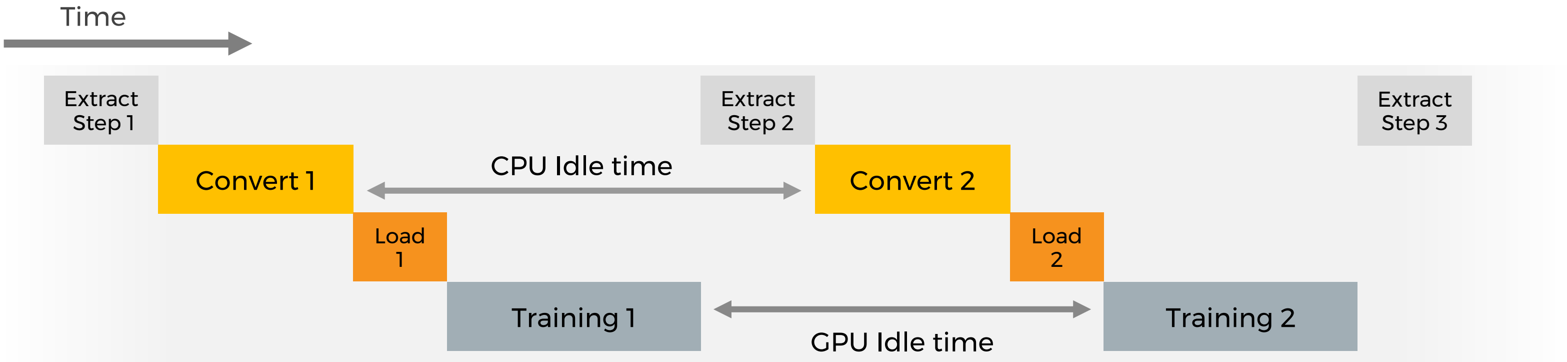
OS-level virtualization

GPU

?

blackbox `ioctl()`

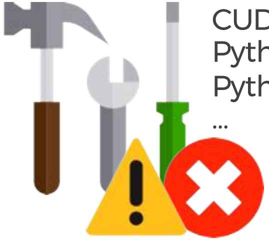
- Idle time from I/O latency



Complexity of SW management

Fast Software Release Cycles

Model/Framework Version Mgmt.



CUDA 9.x
CUDA 10.x
Python 2.x
Python 3.x
...

Compatibility Issues

Let

GPU computation

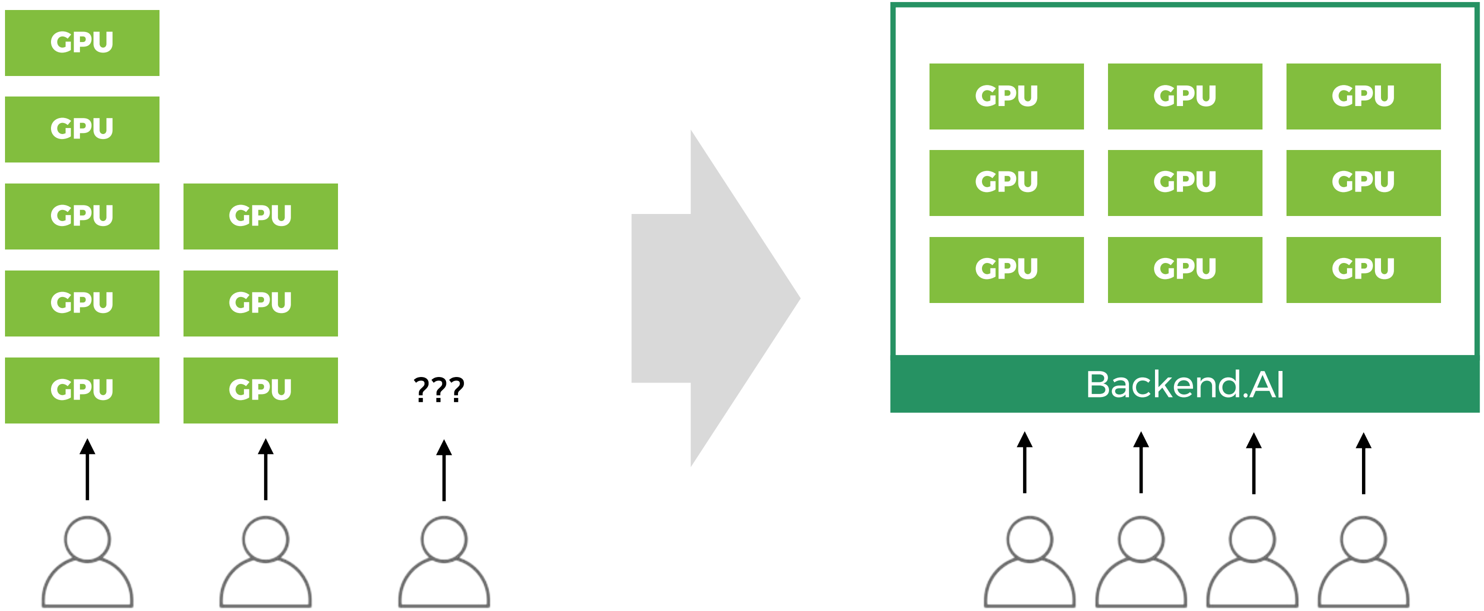
Be

Powerful and **Easy**

Streamlined platform
to **train** and **serve**
ML models
on **premises** and **clouds**
easy and **fast**

Streamlined platform
to **train** and **serve**
ML models
on **premises** and **clouds**
easy and **fast**

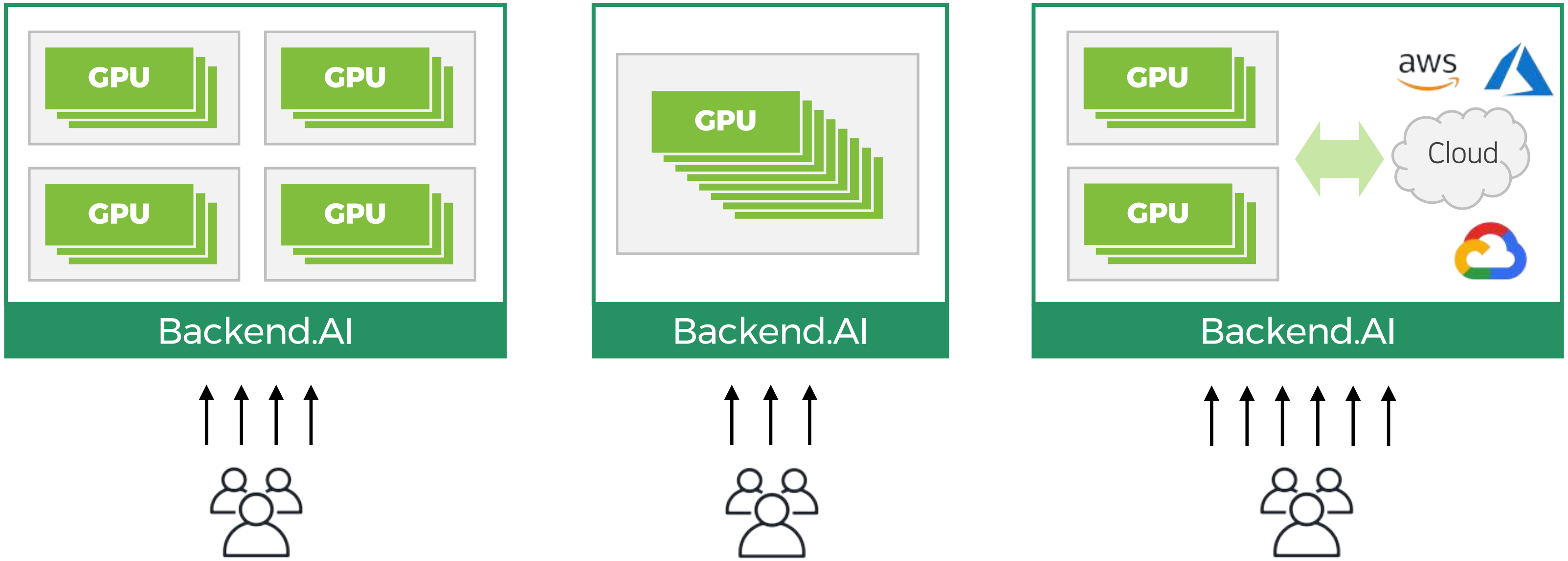
GPU management is difficult!



- Manual assignment of GPUs for researchers
- Both idle & insufficient at the same time
- Manual checks for SW compatibilities

- Sharing and consolidation of GPUs
- Use only what you need at that time
- Containerized runtime environments

Backend.AI Usage Scenario

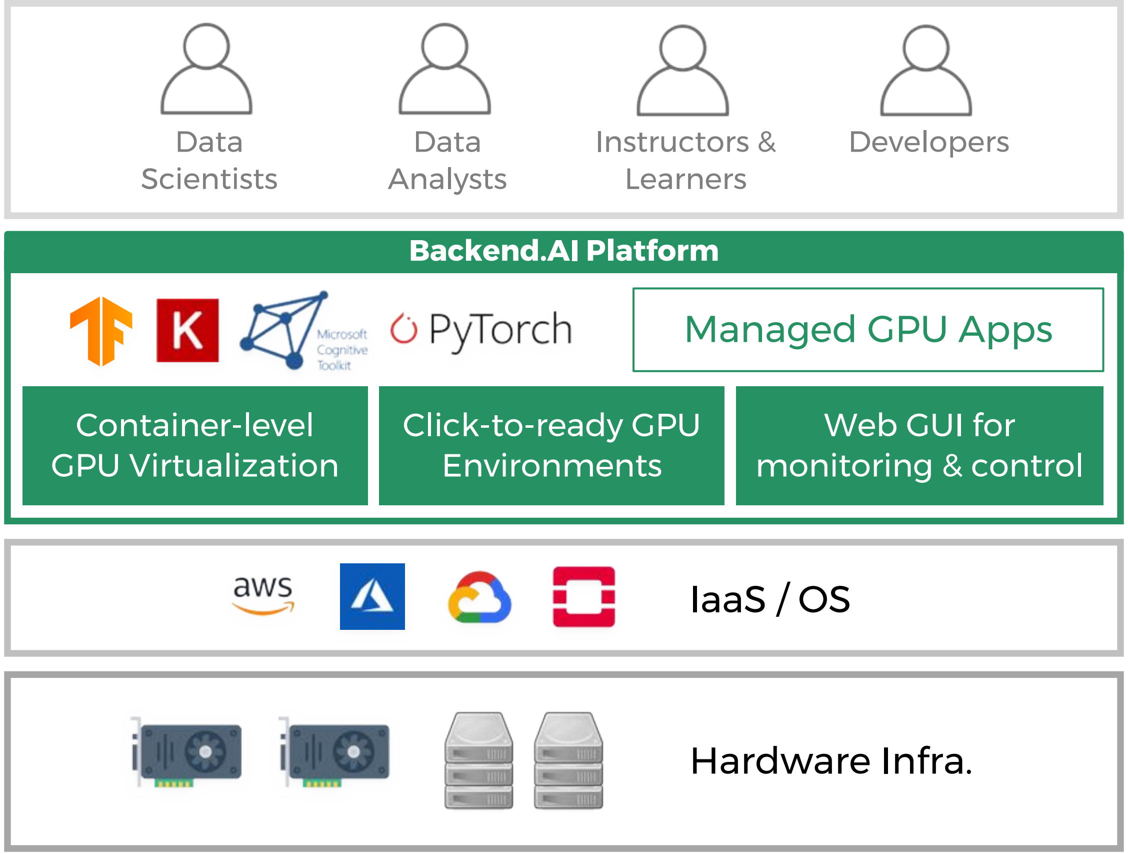


Building GPU clusters

Sharing high-end GPU nodes

Dynamic scaling out from on-prem to clouds

Backend.AI-powered GPU Computing Stack



Seamless migration of existing users

Reduced time to build ML apps

Flexible and efficient GPU utilization

Backend.AI Offerings

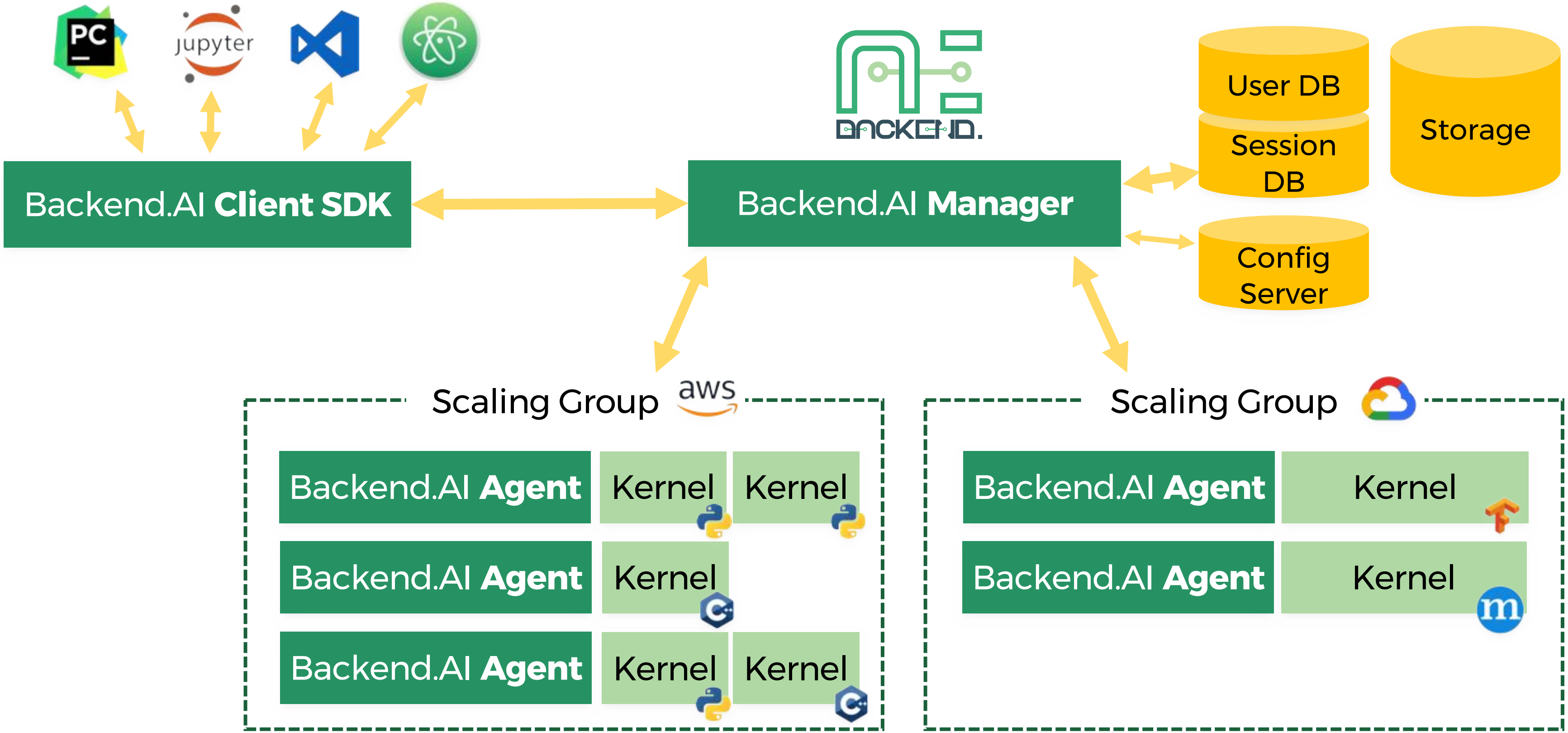
Cloud	Open Source	Enterprise
Fits with your needs instantly	Get the most out of your hardware	End-to-end ML Infra Manager
On-demand GPU envs for HPC and ML/DL with pay-as-you-go pricing	Hackable, customizable computing framework with cutting-edge technologies	Private GPU cloud & cluster managing solution for large-scale enterprises

Backend.AI

Backend.AI Advantages

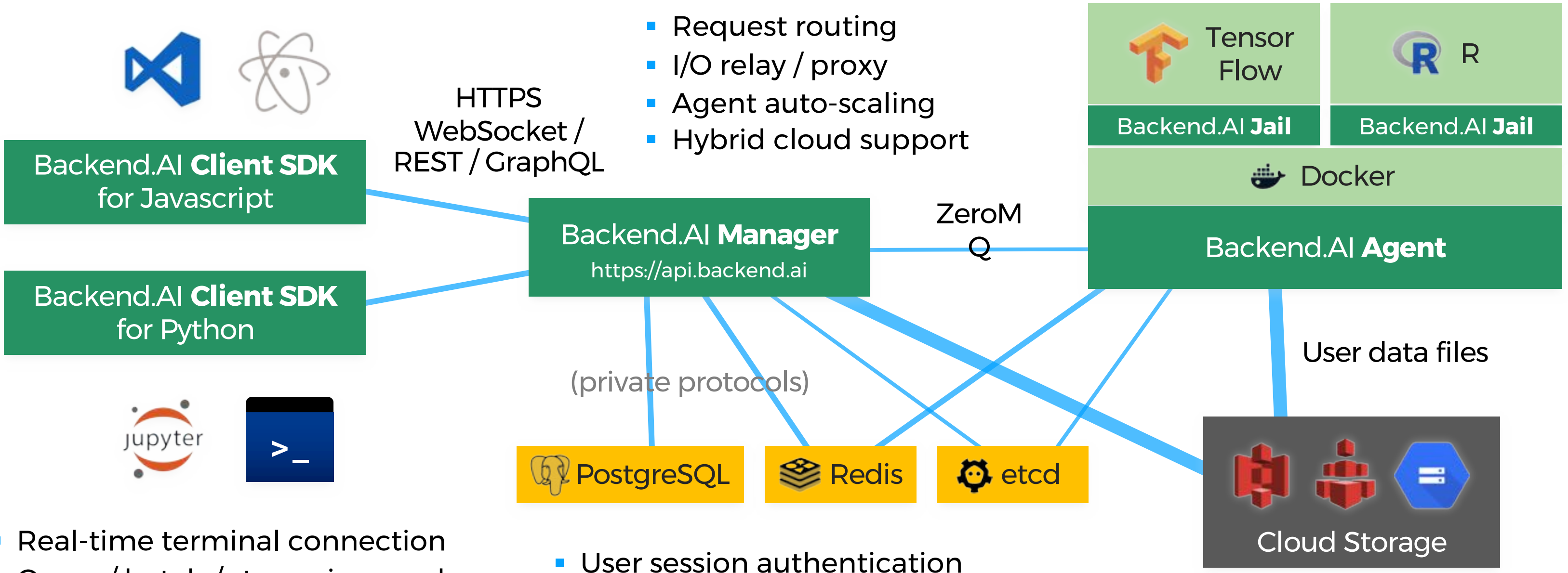
- Only & first solution
  - The market offers solutions specialized for specific functions such as batch scheduling and container hosting.
  - Backend.AI embraces headaches from both ML modelers and DevOps engineers.
- Backend.AI
  - GPU-first optimization
    - ✓ Extensible CUDA support via NVIDIA partnership
    - ✓ Fractional GPU scaling on device
  - Programmable sandboxing
    - ✓ syscall-level logging & customizable security policies
  - Legacy app support
    - ✓ Resource constraining without code changes
    - ✓ e.g., CPU core counting fix for old-school computation libraries

Backend.AI Components



Backend.AI: Detail

- Programmable SysCall Sandbox
- Container resource control including CPU/GPU Core, Memory, Storage

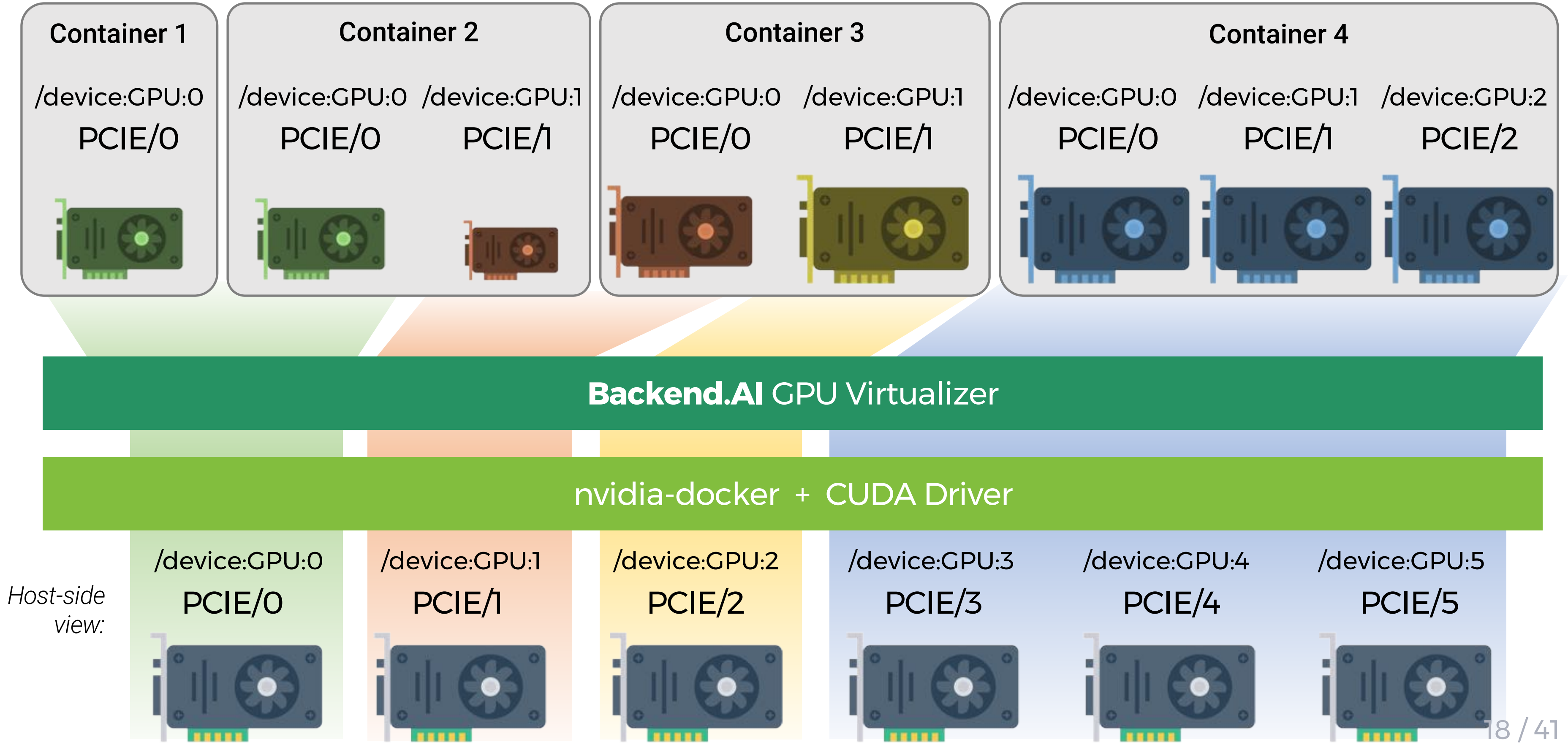


- Real-time terminal connection
- Query / batch / streaming mode
- Usage / session status monitor
- Multimedia I/O rendering

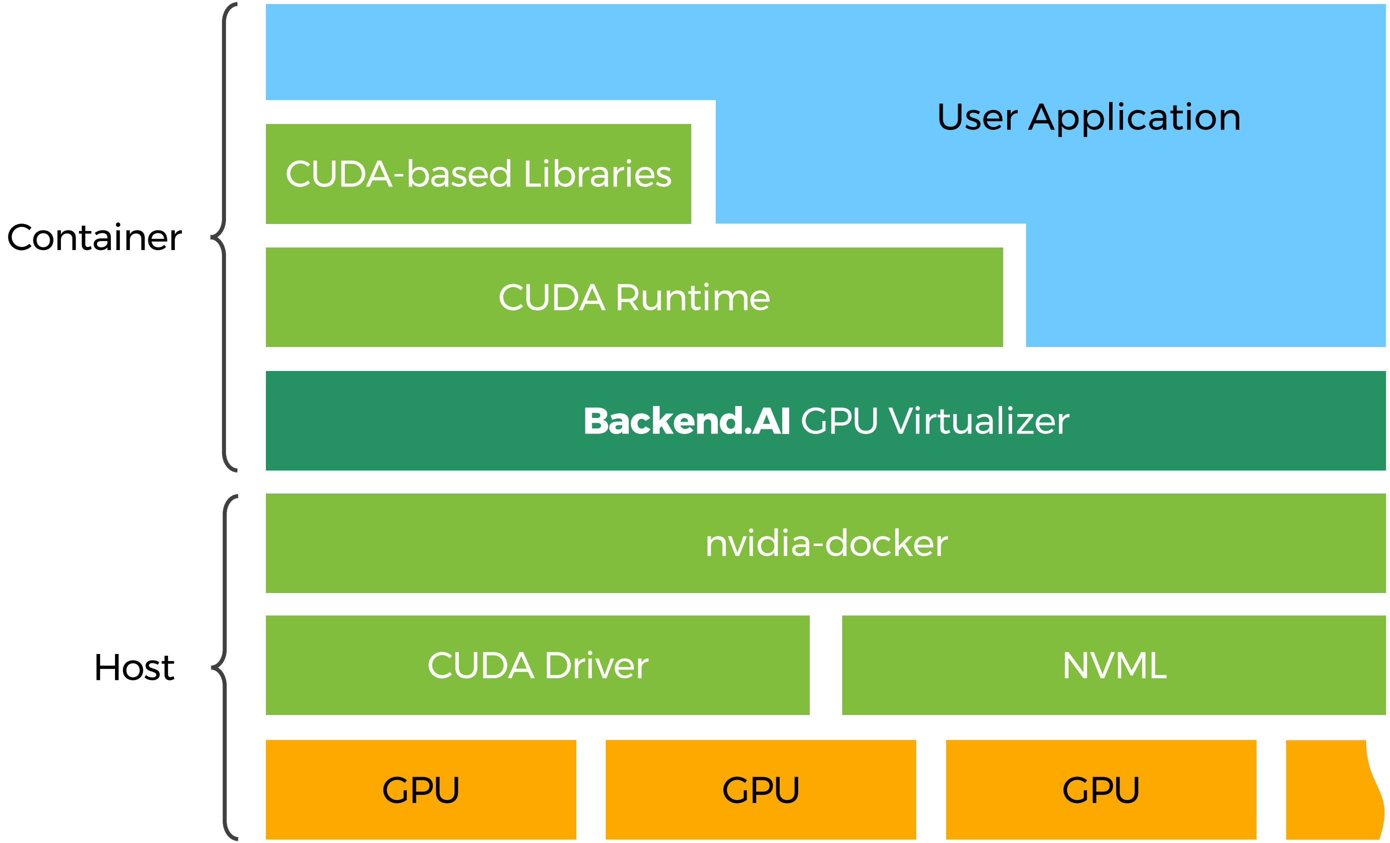
- User session authentication
- Real-time session usage statistics
- Automatic rolling upgrade

- Per-user virtual folder
- Sharing with permission control
- Example dataset

Fractional & Multi-GPU Scaling



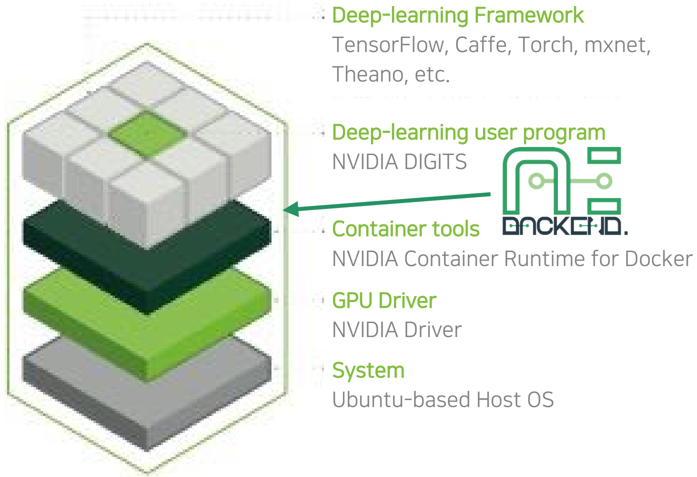
CUDA API Virtualization



- ✓ Takes all benefits of nvidia-docker
- ✓ Requires no user code changes
- ✓ Supports all NGC containers and user-written CUDA apps
- ✓ Enforces per-container GPU resource limits

NVIDIA Integration: DGX Family

- NVIDIA DGX-1/DGX-2
  - Most powerful GPU computing node
    - ✓ High-speed multi-GPU interconnects via NVLink & NVSwitch
    - ✓ Ability to run large-scale models



- Backend.AI Integration
  - Adds following features to NVIDIA container runtime
    - ✓ Fractional sharing of GPUs
    - ✓ ML pipeline components
    - ✓ Topology-aware CPU/GPU scheduling

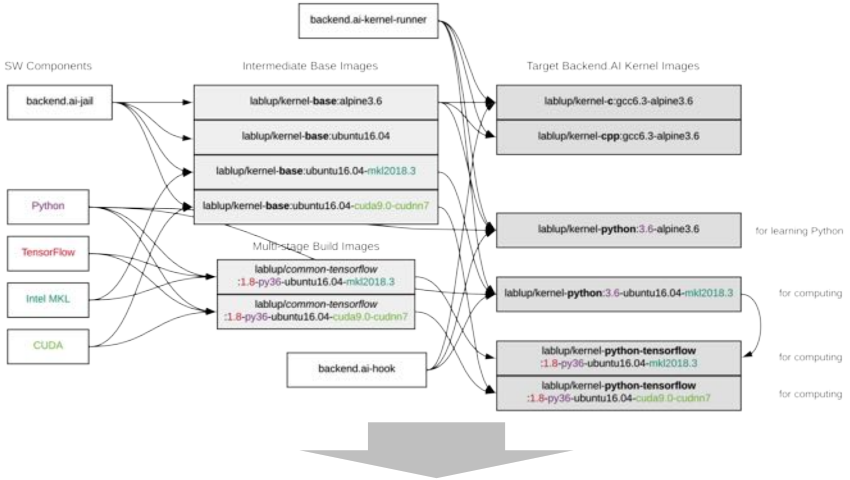
SYSTEM SPECIFICATIONS

GPUs	**16X NVIDIA® Tesla V100**
GPU Memory	**512GB total**
Performance	**2 petaFLOPS**
NVIDIA CUDA® Cores	**81920**
NVIDIA Tensor Cores	**10240**
NVSwitches	**12**
Maximum Power Usage	**10 kW**
CPU	**Dual Intel Xeon Platinum 8168, 2.7 GHz, 24-cores**
System Memory	**1.5TB**

NVIDIA Integration: NGC

- NVIDIA GPU Cloud
  - A curated set of Docker images optimized for NVIDIA GPUs
  - A hosted model zoo for easy start of ML-based apps and transfer learning (announced in GTC 2019)

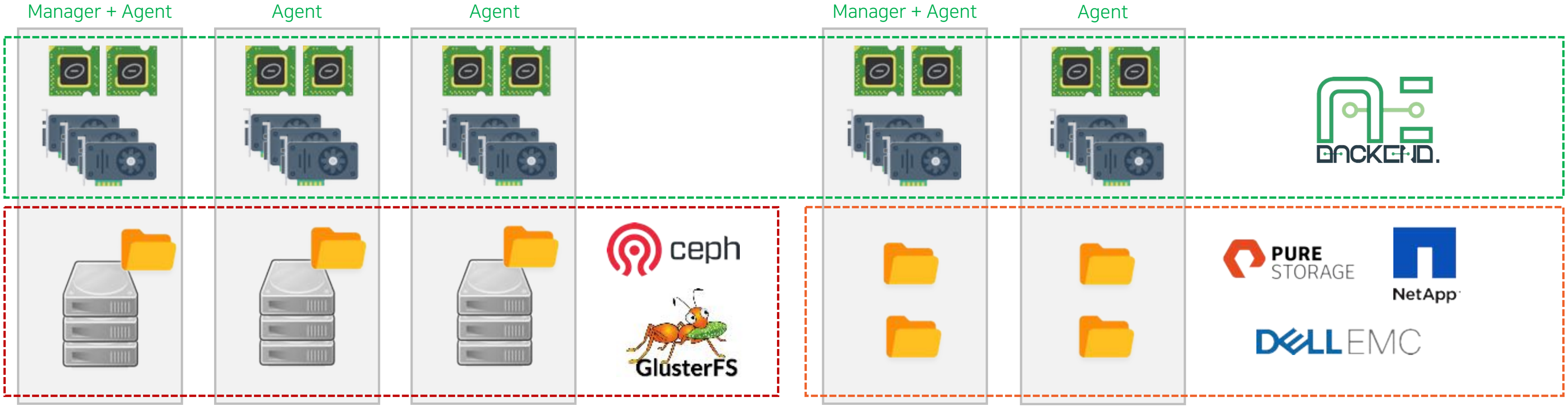
- Backend.AI Integration
  - Instantly pull and run any NGC images by adding some annotations
  - Model download / upload from NGC (coming soon!)




```
FROM nvcr.io/nvidia/digits:18.12-tensorflow
# ...
LABEL ai.backend.kernelspec="1" \
      ai.backend.envs.corecount="OPENBLAS_NUM_THREADS,OMP_NUM_THREADS,NPROC" \
      ai.backend.features="query batch uid-match" \
      ai.backend.accelerators="cuda" \
      ai.backend.resource.min.cpu="1" \
      ai.backend.resource.min.mem="1g" \
      ai.backend.resource.min.cuda.device=1 \
      ai.backend.resource.min.cuda.shares=0.1 \
      ai.backend.base-distro="ubuntu16.04" \
      ai.backend.runtime-type="python" \
      ai.backend.runtime-path="/usr/bin/python" \
      ai.backend.service-ports="digits:http:5000,tensorboard:http:6006,ipython
:pty:3000,jupyter:http:8080,jupyterlab:http:8090"
```

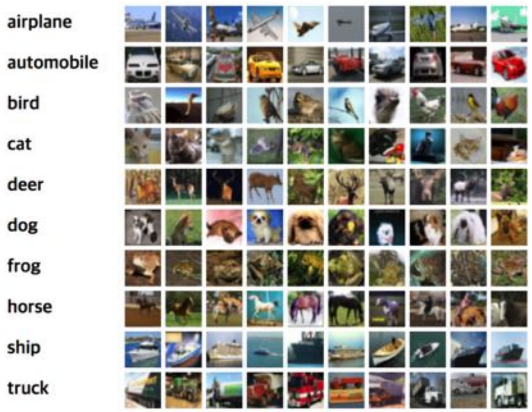
Storage Integration

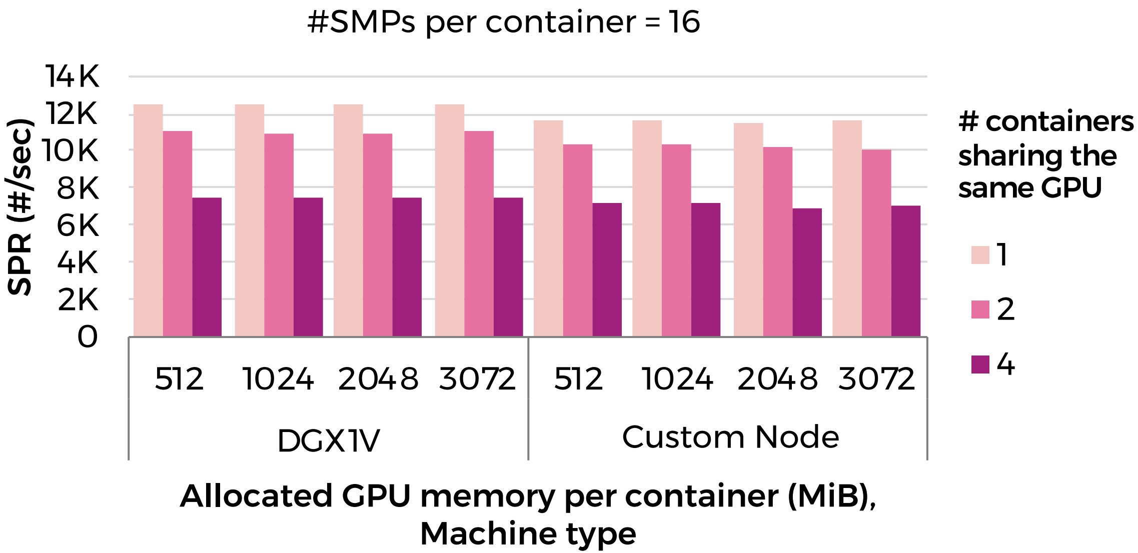
- Backend.AI's storage layer runs on top of any centralized/distributed storage.
- Personal & shared storage abstraction
  - Mount storages into containers like a local filesystem
  - Permission control for user-to-user & group sharing
  - API-level or filesystem-level integration depending on storage solutions



Performance: Single-GPU Fractional Sharing

- Benchmark: Sample processing rate of cifar-10 on a V100 GPU (16/32GB)

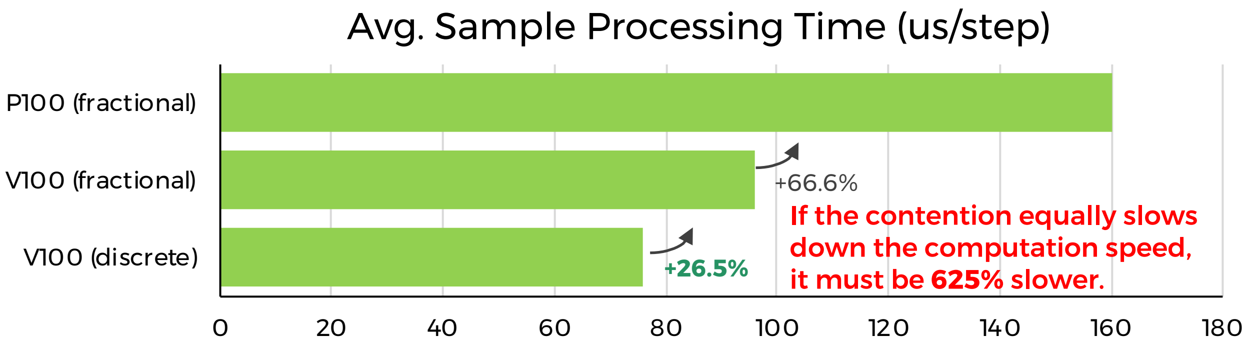




- Results
  - Sharing overhead : **-10% SPR** when a container is added to share the same GPU

Performance: Multi-GPU Fractional Sharing

- Setup : Customer's BMT environment (Intel-based custom GPU server)
- Workload : fashion-MNIST
- P100 GPU Cluster (2-node 16 GPUs)
  - Spec: GPU shared (SMP 4, GPU Memory 1 GiB)
  - Concurrency: 50 users
- V100 GPU Cluster (1-node 8 GPUs)
  - Spec: GPU shared (SMP 4, GPU Memory 1 GiB) / non-shared (whole device)
  - Concurrency: 50 users / 8 users



※ Lower is better

Just Model It (JMI) Contest

- “Standing on the Shoulders of Titans”
- Jan.~Mar. 2019
  - https://events.backend.ai/just-model-it/
  - Provides GPU resources to ML scientists / developers for free!
  - For us: system validation & tests
  - For participants: chances to creating machine learning models without huddle
- How
  - Setup an virtual Backend.AI GPU cluster with many remote GPU servers / Cloud instances
  - Provide resources via Backend.AI client CLI / GUI app

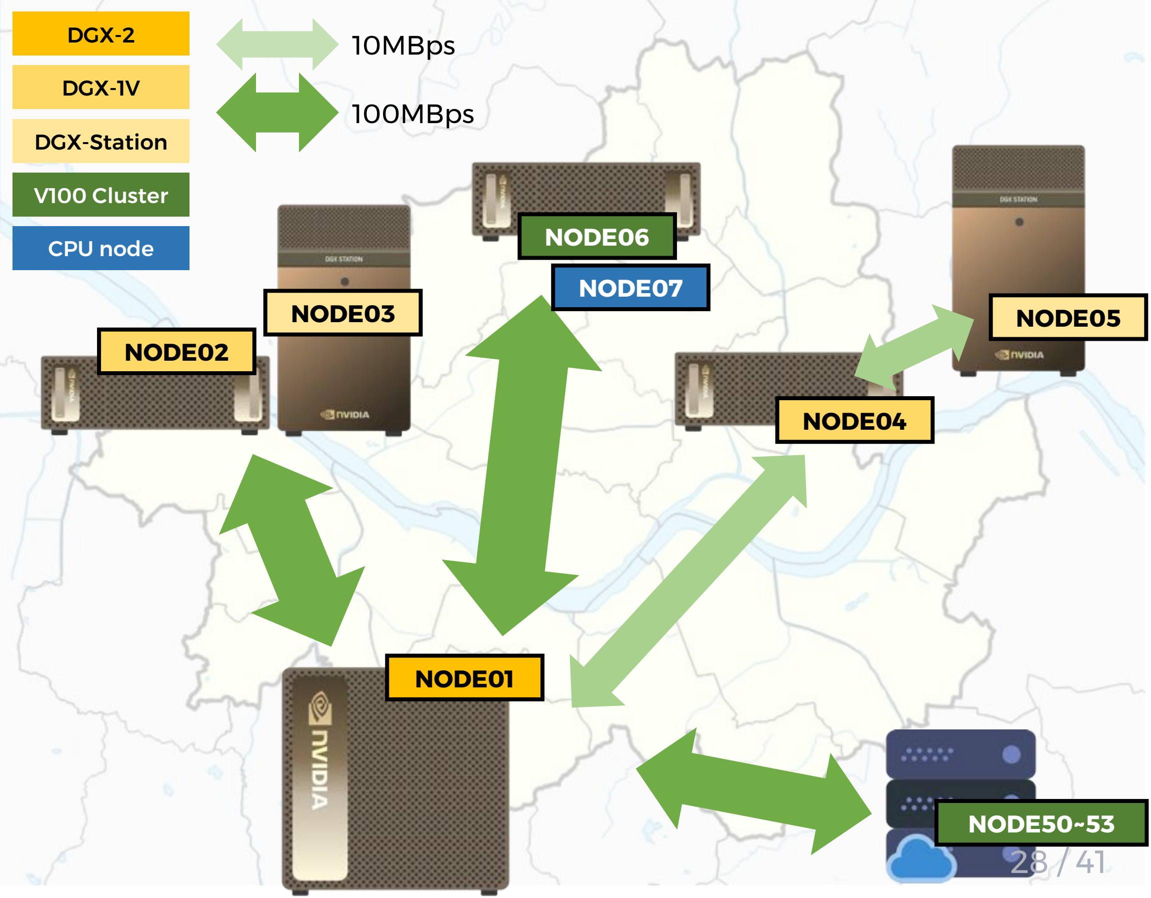
Creating virtual Backend.AI cloud with DGX series

- On-premise cluster for *Just model it* event
- **44** V100 on-premise GPUs + (**8~32**) V100 GPU instance on cloud
  - (16) 1 DGX-2 server NODE01
  - (4) 1 custom GPU server (with 4 V100 GPUs) NODE06
  - (16) 2 DGX-1V **(with support by Nvidia)** NODE02, NODE04
  - (8) 2 DGX Stations **(with support by Nvidia)** NODE03, NODE05
  - (8~32) Amazon EC2 instances (p3-8xlarge) as spot instances NODE50~NODE53
  - + CPU-only on-premise node (44-core Xeon) for compile / data preprocessing NODE07

- **4** geographically distant locations
  - DGX-2 + Custom GPU server (Lablup Inc.)
  - DGX-1V+DGX stations (Baynex , Local Nvidia Partner)
  - DGX-1V+DGX stations (Daebo, Local Nvidia Partner)
  - Amazon EC2 (ap-northeast-2)

Creating virtual Backend.AI cloud with DGX series

- Agent roles
  - **NODE01**: Backend.AI manager
  - **NODE01~05**: Active GPU Cluster
  - **NODE06**: Reserved / Staging area
  - **NODE07**: Image compilation / Julia
  - **NODE50~53**: Spot Instance on AWS
- Storage configuration
  - Scratch disk on each agent
  - Cachefilesd to each node
  - RedHat Ceph Storage as distributed storage backend
    - ✓ Disabled due to the limited network bandwidth



Configurations

- 12 independent teams
  - Research teams / Independent developer / Startups

- Resource allocation (for each team)
  - CPU: 22 Cores (various clock, followed by host CPU)
  - RAM: 512GB
  - Storage: 3TB scratch (8 NVMe RAID-0) + α
  - GPU:64GB (**32x2** or **16x4** V100s)
    - ✓ 32x2: Text workloads (RNN / BERT projects)
    - ✓ 16x4: Image / video workloads (CNN / GAN projects)
    - ✓ Multi-GPU scaling mode

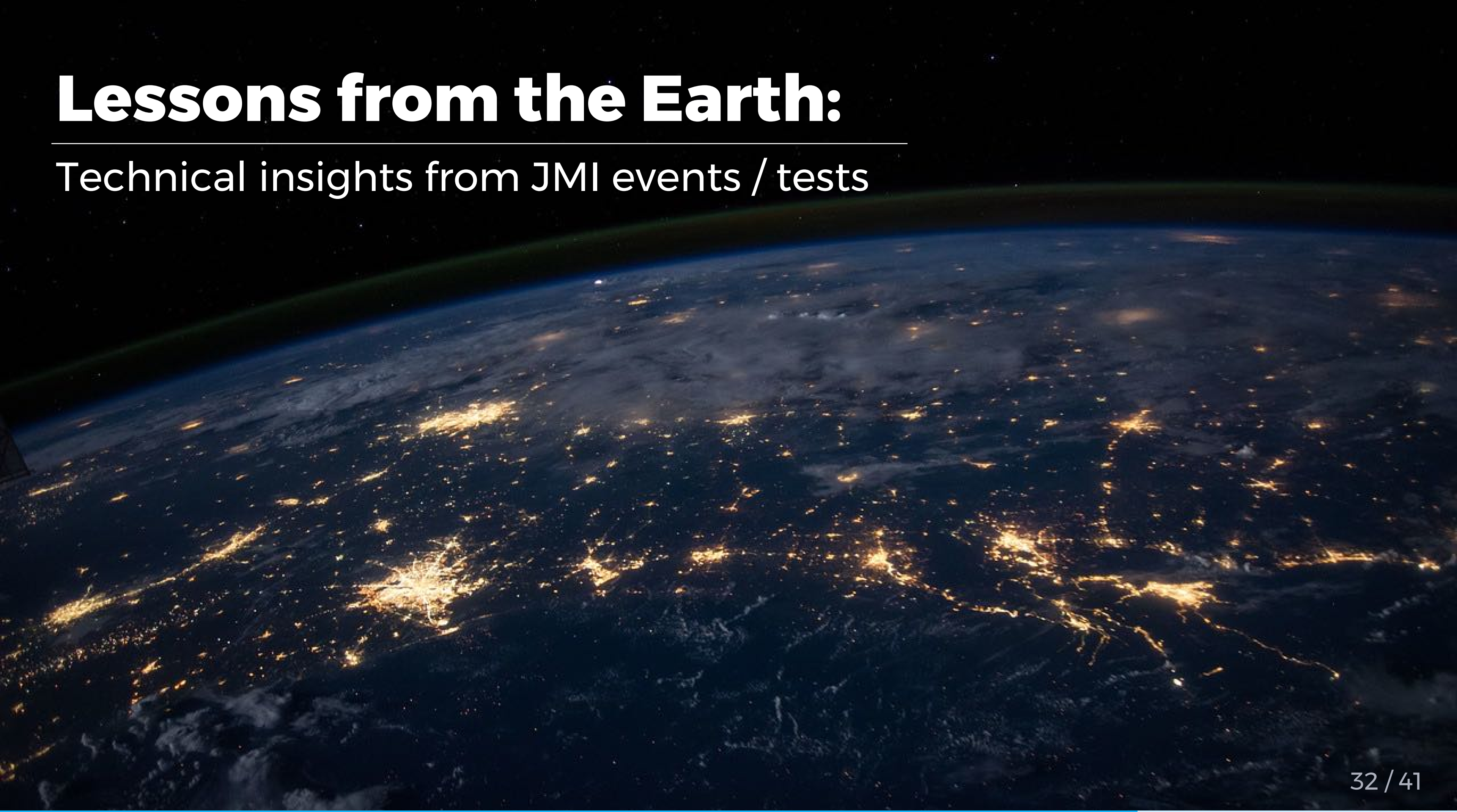
The Event Begins,

AI Tech Talk
21 Jan. 2019, Google Startup Campus

...and one month passed.

Lessons from the Earth:

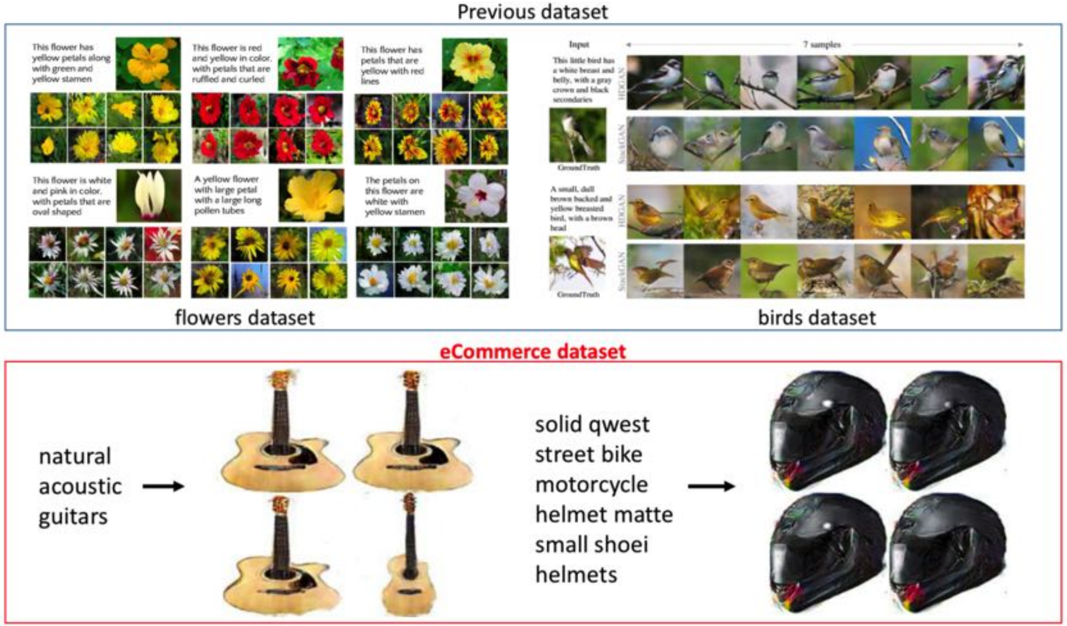
Technical insights from JMI events / tests



JMI Event Showcase: TAC-GAN-eCommerce

- Problem
  - 1. Missing image for product ad.
  - 2. Promotional text to product images → Generates unrelated meta data

- Solution: text to image synthesis
  - Meta data to product image
  - Prototyping TAC-GAN
  - 1. Creating production image using generator
  - 2. Judge abusing using discriminator

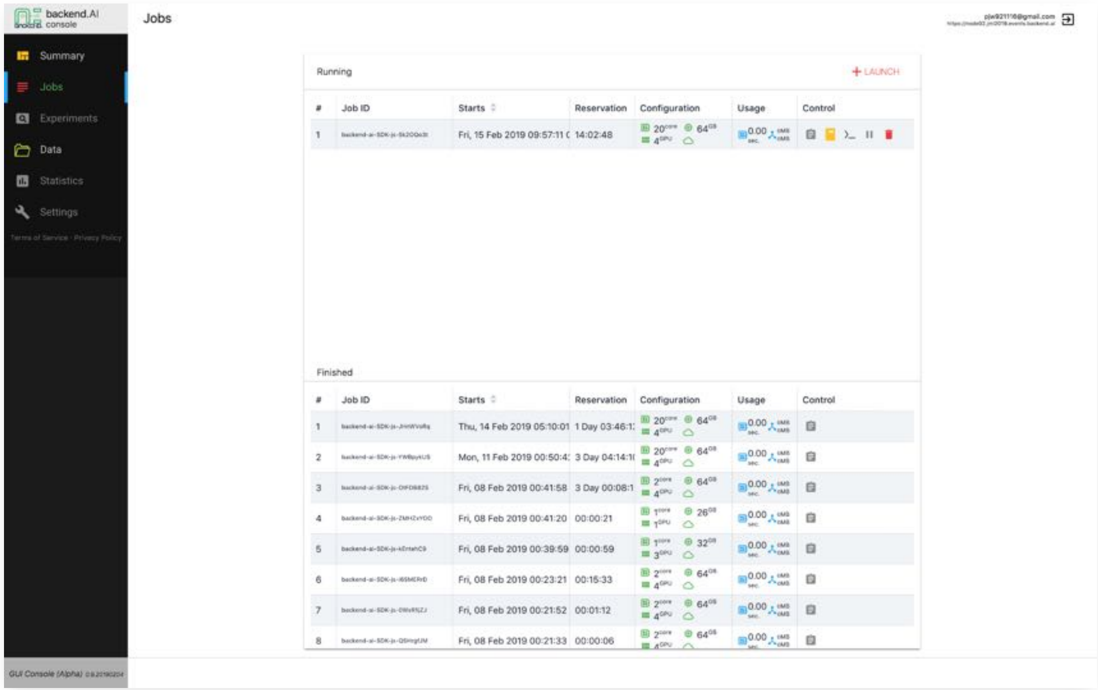


https://github.com/junwoopark92/TAC-GAN-eCommer

JMI Event Showcase: TAC-GAN-eCommerce

- Data specification
  - Amazon eCommerce Dataset
  - 9M products
  - 16,000 leaf categories
  - 260GB images

- Preprocessing Pipeline
  - Indexing using sntencepiece
  - Sentence embedding with doc2vec in genism
  - Data augmentation with label shuffling



https://github.com/junwoopark92/TAC-GAN-eCommer

JMI Event Showcase: TAC-GAN-eCommerce

- Generated image examples

https://github.com/junwoopark92/TAC-GAN-eCommer

JMI Event Showcase: TAC-GAN-eCommerce

- Generating product image from product metadata
- Example: Guitar + variations

+Accoustics

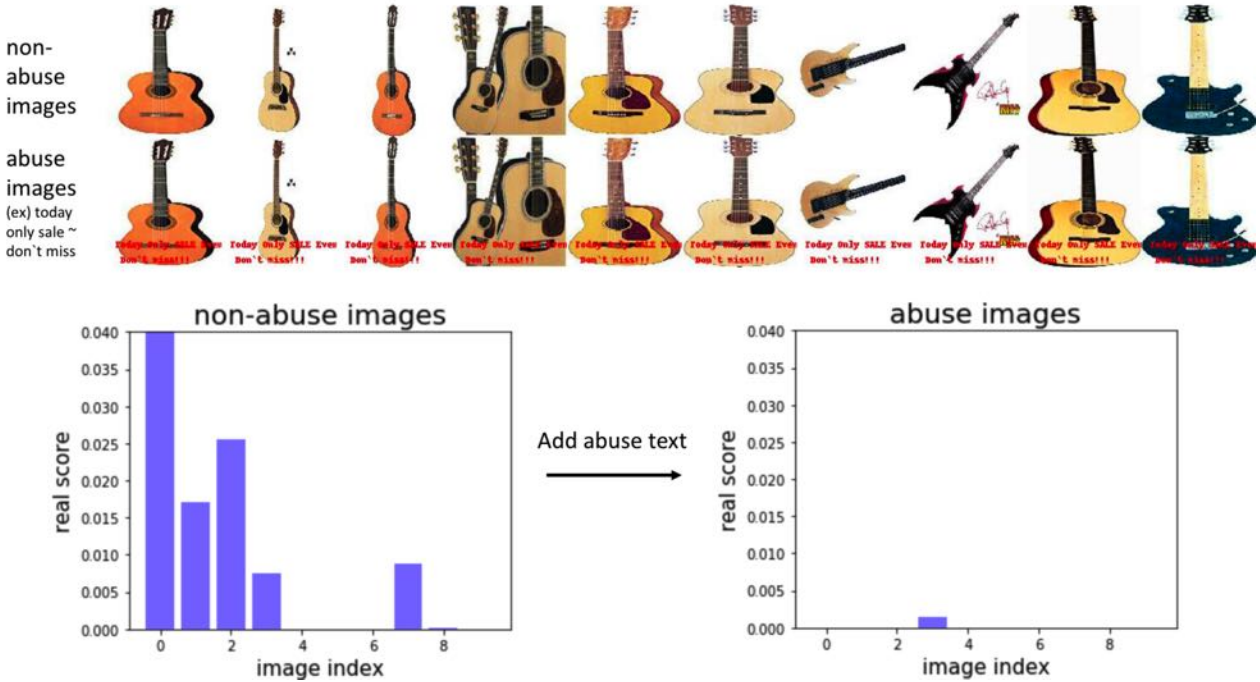
+Color

+Electric

https://github.com/junwoopark92/TAC-GAN-eCommer

JMI Event Showcase: TAC-GAN-eCommerce

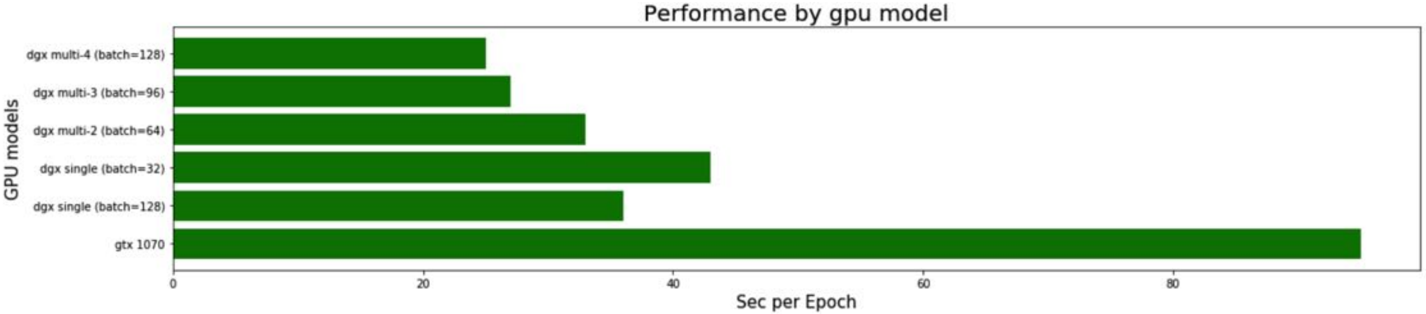
- Classify abused product image using discriminator



https://github.com/junwoopark92/TAC-GAN-eCommer

JMI Results: Benchmark (TAC-GAN-eCommerce)

- 1070 vs Tesla V100 16GB single (batch size = 128):
  - ~3X performance difference.
  - Adjusted the batch size until there was no performance degradation due to I/O.
  - Average load: 90~100 (1070), 80~90 (V100)
- Tesla V100 16GB (single ~ 4, batch size = 32 ~ 128)
  - Performance increases as the number of GPUs increases, but not linear.
  - TAC-GAN model size is small: Data feeding seems to be a bottleneck.
  - If the size of the batch is increased beyond a certain size, an error that exceeds the shared area of IO occurs.
  - Load average: Single GPU: 80~90, 4 GPUs: 40~50



JMI Event: Lessons

- Backend.AI offers what we intended to offer.
  - **nvidia-docker** → a *consistent* way of using GPUs inside containers.
  - **Backend.AI** → a *flexible* way of allocating GPUs to containers.

- Technical insights
  - **Unobtrusive upgrade** is essential to keep long-running computations successful.
    - ✓ The manager and agent may restart while keeping containers running.
    - ✓ Network tunneling for in-container services (e.g., Jupyter) enables seamless upgrades with brief reconnections.
- UX insights
  - **Non-developer users** often think containers same as persistent VMs.
    - ✓ Containers are *on-demand* and *volatile*.
    - ✓ The key advantage of containers (reproducibility) may be the key surprise ("my things are gone!") for some category of users.

Summary

- Goals towards real-world ML systems
  - Data collection & feature extraction
  - Hardware resource management
  - Model deployments & feedback monitoring

- **Backend.AI**: GPU-optimized middleware for ML model training & serving
  - Integration with NVIDIA platforms (DGX + NGC)
  - Integration with storage platforms (open-source, vendors)
  - GPU fractional scaling: reduce idle time of GPUs and TCO
  - Cloud and on-premise offerings
    (open-source / cloud subscription / enterprise support)

- Future roadmap: Evolution to an end-to-end ML pipeline system

Thank you

Inquiry : contact@lablup.com

Lablup Inc.	https://www.lablup.com
Backend.AI	https://www.backend.ai
Backend.AI GitHub	https://github.com/lablup/backend.ai
Backend.AI Cloud	https://cloud.backend.ai