Project MagLev: NVIDIA’s production-grade AI Platform
Divya Vavili, Yehia Khoja - Mar 21 2019
Agenda

- AI inside of NVIDIA
- Constraints and scale
- AI Platform needs
- Technical solutions
- Scenario walkthrough
- Maglev architecture evolution
AI inside of NVIDIA
Deep Learning is fueling all areas of business

Self-Driving Cars
Robotics
Healthcare

AI Cities
Retail
AI for Public Good
BUILDING AI FOR SDC IS HARD

Every neural net in our DRIVE Software stack needs to handle 1000s of conditions and geolocations.
# Constraints and scale

**SDC Scale Today at NVIDIA**

<table>
<thead>
<tr>
<th>12-camera+Radar+Lidar RIG mounted on 30 cars</th>
<th>1,500 labelers</th>
<th>4,000 GPUs in cluster = 500 PFLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 1PB collected/month</td>
<td>20M objects labeled/mo</td>
<td>100 DRIVE Pegasus in cluster (Constellations)</td>
</tr>
<tr>
<td>15PB active training+test dataset</td>
<td>20 unique models 50 labeling tasks</td>
<td>1PB of in-rack object cache per 72 GPUs, 30PB provisioned</td>
</tr>
</tbody>
</table>
Constraints and scale
What are our requirements?

Safety

Tons of data!

Inference on edge

Reproducibility
What testing scale are we talking about?

We’re on our way to 100s PB of real test data = millions of real miles + 1,000s DRIVE Constellation nodes for offline testing alone & billions of simulated miles

- NVIDIA’s data collection (miles)
- Active testing to date (miles)
- Target robustness (miles)

* DRIVE PEGASUS Nodes
Enable the development of AV Perception, fully tested across 1000s of conditions, and yielding failure rates $< 1$ in $N$ miles, $N$ large

<table>
<thead>
<tr>
<th>Scalable AI Training</th>
<th>PB-Scale AI Testing</th>
<th>AI-based Data Selection/Mining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traceability: model=&gt;code+data</td>
<td>Seamless PB-Scale Data Access</td>
<td>Workflow Automation</td>
</tr>
</tbody>
</table>
MagLev for SDC

“Collect → Select → Label → Train → Test” as programmatic, reproducible workflows.

Enables active learning for SDC, with labeling in the loop!

Data Lake → Selected Datasets → Labeled Datasets → Metrics & Logs → Trained Models

1PB per month

15PB Today

20M objects labeled per month

1,500 Labelers

ML/Metrics UI

Large AI Dev team

AWS

Data selection Job #1 → Testing Job #1 → Training Job #1

Data selection Job #2 → Testing Job #1 → Training Job #2

Data selection Job #N → Testing Job #N → Training Job #N

Run Multi-Step Workflow (workflow = sequence of map jobs)

Kubernetes over 4000 GPU Cluster (= 480 PFLOPs)
The need for an AI platform
Enabling automation of training, and testing workflows
So how did we solve for this?
Technical solution(s)

Safety

- Non-compromisable primary objective for the passengers

All other engineering requirements stem from this
- Models tested on huge datasets to be confident
- Faster iteration that aids in producing extremely good and well-tested models
- Reproducibility/Traceability
Technical solution(s)

Tons of data!

- Collecting enormous amounts of data under innumerable scenarios is key to building good AV models.

- Now that we have the data, what next?
  - How do engineers access this data?
  - How do you make sure that the data:
    - can be preprocessed for each team’s need?
    - is not corrupted by other members of the team or across teams?

- Lifecycle management of data
Technical solution(s)

Tons of data!

What is the solution?

**vdisk**

- Virtualized Immutable file-system
- Offers broad platform support
- Structured to support data deduplication
- Inherently supports caching
- Provides kubernetes integration making it cloud-native

---

Inference on edge

Reproducibility
Technical solution(s)

Inference on edge

- AV model inference is limited in terms of hardware capabilities

- So, finding a lighter model without losing performance is prudent and takes multiple and faster iterations
Technical solution(s)

Reproducibility

Why?
- Being able to run a 10 year old workflow and get the same results
- Faster iteration of model development
- Understand why a model behaved certain way

Requires:
- Proper version control of datasets, models and the experiments

Reproducibility
- ... and traceability go hand in hand
MagLev
Scenario walkthrough

- Predicting 12 month mortgage delinquency using Fannie Mae Single family home loan data

Key points:
  - Immutable dataset creation
  - Specifying workflows and launching them
  - End-to-traceability
Creating an immutable dataset

```bash
>>> maglev volumes create --name <my-volume> --path
</some/local/directory/path> [--resume-version <version>]

Creating volume: Volume(name = my-volume, version = 449c8efa-eaef-4d9b-81b9-3a59fe269e9b)
Uploading '<local-file>'...

Successfully created new volume.
Volume(name = my-volume, version = 449c8efa-eaef-4d9b-81b9-3a59fe269e9b)
```

Creates a ISO image

ISO image only contains the metadata for the dataset while the actual dataset resides in S3
MagLev
Scenario walkthrough

defaults:
  gpus: 1
  image: 'nvcr.io/nvidia_general/chrisg:d196974e8541'
  image_pull_secrets: ['nvcrio']
  tasks:
    - name: rapids-data-preprocessing
    - name: dnn-experiment
    - name: xgboost-experiment
    - name: dnn-train
    - name: dnn-evaluate
    - name: xgboost-train
    - name: xgboost-pick
    - name: xgboost-retrain
    - name: xgboost-test
MagLev
Scenario walkthrough

defaults:
  gpus: 1
  image: 'nvcr.io/nvidia_general/chrisg:d196974e8541'
image_pull_secrets: ['nvcrio']
tasks:
  - name: rapids-data-preprocessing
  - name: dnn-experiment
  - name: xgboost-experiment
  - name: dnn-train
  - name: dnn-evaluate
  - name: xgboost-train
  - name: xgboost-pick
  - name: xgboost-retrain
  - name: xgboost-test

- name: rapids-data-preprocessing
  args:
  - '0.05'
  inputs:
    volumes:
    - mount_path: /in/mortgage/
      name: artifact-rapids-mortgage-2000Q1-tiny
      version: 0df2b082-24fd-4934-9e18-2c3ed3d70cb2
  outputs:
    volumes:
    - mount_path: /out/dnn/
      name: dnn-data
    - mount_path: /out/xgboost/
      name: xgboost-data
MagLev
Scenario walkthrough

defaults:
  gpus: 1
  image: 'nvcr.io/nvidia_general/chrisg:d196974e8541'
image_pull_secrets: ['nvcrio']
tasks:
  - name: rapids-data-preprocessing
  - name: dnn-experiment
  - name: xgboost-experiment
  - name: dnn-train
  - name: dnn-evaluate
  - name: xgboost-train
  - name: xgboost-pick
  - name: xgboost-retrain
  - name: xgboost-test

  - name: dnn-experiment
    command: >-
      maglev experiment-sets create -f
      viaduct/rapids/mortgage/small/torch/experiments/adam_hyperopt.yaml
    outputs:
      volumes:
        - mount_path: /out/
          name: out
MagLev
Scenario walkthrough

defaults:
    gpus: 1
    image: 'nvcr.io/nvidia_general/chrisg:d19697e8541'
image_pull_secrets: ['nvcrio']
tasks:
    - name: rapids-data-preprocessing
    - name: dnn-experiment
    - name: xgboost-experiment
    - name: dnn-train
    - name: dnn-evaluate
    - name: xgboost-train
    - name: xgboost-pick
    - name: xgboost-retrain
    - name: xgboost-test

- name: dnn-train
  command: >-
    main --maglev --patience 4 --dataset_id 0df2b082-24fd-4934-9e18-2c3ed3d70cb2 --num_features 2048 --hidden_dims 512 512 512 512 --optimizer adam --data_dir /in/data/ --epochs 5 --ignore_bad_pr_auc
  completions: 10
  inputs:
    volumes:
      - from: rapids-data-preprocessing
        mount_path: /in/data
        name: dnn-data
        - from: dnn-experiment
          mount_path: experiment-set
          name: out
  outputs:
    volumes:
      - mount_path: out
        name: out
MagLev
Scenario walkthrough

defaults:
  gpus: 1
  image: 'nvcr.io/nvidia-general/chrisg:d196974e8541'
image_pull_secrets: ['nvcrio']
tasks:
- name: rapids-data-preprocessing
- name: dnn-experiment
- name: xgboost-experiment
- name: dnn-train
- name: dnn-evaluate
  command: |
    evaluate --dataset_id 0df2b082-24fd-4934-9e18-2c3ed3d70cb2 --num_features 2048 --hidden_dims 512 512 512 512 --test_dataset /in/data/encoded_test_discrete.csv.gz
inputs:
  volumes:
    - from: rapids-data-preprocessing
      mount_path: /in/data
      name: dnn-data
    - from: dnn-experiment
      mount_path: experiment-set
      name: out
- name: xgboost-train
- name: xgboost-pick
- name: xgboost-retrain
- name: xgboost-test
PyTorch DNN vs XGBoost with RAPIDS on MagLev

```python
# matplotlib inline

import maglev
import matplotlib.pyplot as plt

Workflow Metadata

# Update this with your workflow id
WORKFLOW_ID = "c4254540-76e1-5460-9e87-c4801c791a28"

DNN_MODEL_ID = "d1c8673b-a258-48e6-9360-fb758d8f134c"
XGBOOST_MODEL_ID = "c31d4b39-b86e-4c61-81db-dab979d96b8b"

# Used to track performance of models over time
DATASET_ID = "10ccd8e0-f172-443a-ad29-f52711ff9e3"

client = maglev.Client.default_service_client()

print(client.get_workflow(WORKFLOW_ID))
```

Workflow(workflow_id=u'c4254540-76e1-5460-9e87-c4801c791a28',
  namespace=u'maglev-849d16e8-3f2d-4615-b508-ba7f08925d0d',
  name=u'maglev-workflow-r6g5v',
  project_id=u'cebc2a8ba-5865-11e8-9c2d-fa7ae01bbebc'.
  creation_date=datetime.datetime(2019, 1, 26, 16, 29, 34, 470722))
Hyperparameter Experiments

Let's load and inspect the experiments created by the dnn-experiment and xgboost-experiment tasks.

```python
eperiment_sets = client.list_experiment_sets(workflow_id=WORkFLOW_ID)
```

XGBoost

```python
xgboost_es = list(filter(lambda x: 'xgboost' in x.get_description().lower(), experiment_sets))[0]
print("Description:
\n\t{}").format(xgboost_es.get_description())
```

Description:

```
FNMA Mortgage Dataset Small XGBoost Hyperopt
```

Experiment Parameters

```python
xgboost_es.get_parameters()[:"exp_id", "alpha", "eta", "gamma", "lambda", "max_depth"]).sort_values('exp_id')
```

<table>
<thead>
<tr>
<th>exp_id</th>
<th>alpha</th>
<th>eta</th>
<th>gamma</th>
<th>lambda</th>
<th>max_depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0.451904</td>
<td>0.007317</td>
<td>7.288823</td>
<td>3.532960</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>1.265562</td>
<td>0.107846</td>
<td>4.022320</td>
<td>3.569992</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>2.5863257</td>
<td>0.019554</td>
<td>7.396898</td>
<td>5.197621</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3.4923602</td>
<td>0.159066</td>
<td>4.052578</td>
<td>1.768556</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4.8972206</td>
<td>0.214875</td>
<td>9.208735</td>
<td>1.234819</td>
<td>7</td>
</tr>
</tbody>
</table>
Model Comparison on Test Set

Our workflow includes tasks to pick the best XGBoost and DNN models and then evaluate them on the holdout test set created in `rapids-data-preprocessing`. During this evaluation we create MagLev experiment metrics and mark them as optimum for easier fetching, which we do below.

```python
# Get the metrics created for this workflow
xgboost_metrics = client.list_metrics(workflow_id=WORKFLOW_ID, model_id=XBOOST_MODEL_ID)
dnn_metrics = client.list_metrics(workflow_id=WORKFLOW_ID, model_id=DNN_MODEL_ID)

# Find the optimum metrics that have names starting with 'test/

def best_metrics(metrics):
    return {m.name: m for m in metrics if m.optimum_is_maximum and 'test/' in m.name}

# Compare!
best_xgboost_metrics = best_metrics(xgboost_metrics)
best_dnn_metrics = best_metrics(dnn_metrics)
for m in ['test/pr_auc', 'test/roc_auc']:
    print("{}").format(m.upper(), best_xgboost_metrics[m].value, best_dnn_metrics[m].value)
print("Best XGBoost Model Version: \t{}").format(best_xgboost_metrics['test/pr_auc'].model_version_id))
print("Best PyTorch DNN Model Version: \t{}").format(best_dnn_metrics['test/pr_auc'].model_version_id))

TEST/PR_AUC:
    xgb:  0.6591
    dnn:  0.2556

TEST/ROC_AUC:
    xgb:  0.9509
    dnn:  0.8729

Best XGBoost Model Version: 3ec0be68-3cee-55d1-8650-c07f5e0fb65a
Best PyTorch DNN Model Version: ac07d7c8-c26d-5366-8d4b-cff48ae4f8b2
DNN Models Performance on Test Dataset Over Time

During creation of the evaluation metrics of the PyTorch model on the test set we’ve also assign datasets to these metrics. This allows us to track over time how models perform on a particular dataset:

```python
all_dnn_metrics = client.list_metrics(model_id=DNN_MODEL_ID)

def filter_dnn_metric(m):
    if m.name != 'test/pr_auc': return False
    if m.datasets is None: return False
    if not isinstance(m.datasets, dict): return False
    if 'dnn/test' not in m.datasets: return False
    if m.datasets['dnn/test'] != DATASET_ID: return False
    return True

test_pr_auc = list(filter(filter_dnn_metric, all_dnn_metrics))
test_pr_auc = sorted(test_pr_auc,
                     key=lambda m: m.creation_date)

x = [m.creation_date for m in test_pr_auc]
y = [m.value for m in test_pr_auc]

plt.figure(figsize=(11, 7))
plt.xlabel("DATE")
plt.ylabel("PR-AUC")
plt.title("PR-AUC on Dataset\n{};{}").format('dnn/test', DATASET_ID))
plt.plot_date(x, y);
```

Best run with this dataset
MagLev Architecture Evolution

Version 1 - Technical viability

Compute and data on public cloud
- Mostly for technical evaluation
- Costs skyrocketing
- Poor performance
  - clash between functionality and efficiency

Early decisions
- Cloud native platform
- General purpose services/ETL pipelines hosted on public cloud allows us to elastically scale based on requirements
MagLev Architecture Evolution

Version 2 - Minimize costs

Compute on internal data-center for GPU workloads
- Minimize costs
- Take advantage of innovation on GPUs before it hits the market
- Huge compute cluster that is always kept busy by the training/testing workflows

What needed to improve:
- Performance due to lack of data locality
MagLev Architecture Evolution

Version 3 - High performance

Internal data center specialized for both compute and data performance

- High performance due to data locality
- Better UX for data scientists
  - Programmatically create workflows
## MagLev Data Center Architecture

### MagLev Services

<table>
<thead>
<tr>
<th>Services on AWS</th>
<th>Kubernetes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2</td>
<td>35kW Rack</td>
</tr>
<tr>
<td></td>
<td>35kW Rack</td>
</tr>
<tr>
<td>...</td>
<td>35kW Rack</td>
</tr>
<tr>
<td>S3</td>
<td>35kW Rack</td>
</tr>
<tr>
<td></td>
<td>35kW Rack</td>
</tr>
<tr>
<td></td>
<td>35kW Rack</td>
</tr>
</tbody>
</table>

### Multi-PB Datasets stored on AWS S3

High-bandwidth interconnect to replicate them locally, on SwiftStack (S3-compat object API)

In-rack bandwidth between storage and DGX optimized for all our workloads (inference/mining and training)

Each rack:
- 9 DGX-1 = 72 TESLA V100 GPUs = 9 PFLOPs
- 1PB of object storage
MagLev Service Architecture

- General service cluster on public cloud
  - Authentication
  - Volume management
  - Workflow traceability
  - Experiment/Model management
- Compute cluster on internal NGC cloud
- Both clusters are cloud-native built on top of Kubernetes