Speeding up Deep Learning Services: When GPUs meet Container Clouds

Dr. Seetharami Seelam & Dr. Yubo Li
Research Staff Members
IBM Research
GPU Tech Conference: San Jose, CA – May 10, 2017
Outline

• Who are we
• Why should you listen to us
• What problems are we trying to solve
• Challenges with delivering DL on Cloud
• What have we done in Mesos and Kubernetes
• What is left to do
• How can you help
Who are we

Dr. Yubo Li (李玉博)

Dr. Yubo Li is a Research Staff Member at IBM Research, China. He is the architect of the GPU acceleration and deep learning service on SuperVessel, an open-access cloud running OpenStack on OpenPOWER machines. He is currently working on GPU support for several cloud container technologies, including Mesos, Kubernetes, Marathon and OpenStack.

Dr. Seetharami Seelam

Dr. Seelam is a Research Staff Member at the T. J. Watson Research Center. He is an expert at delivering hardware, middleware, applications as-a-service using containers. He delivered Autoscaling, Business Rules, Containers on Bluemix and multiple others internally.
Why should you listen to us

• We have multiple years of developing, optimizing, and operating container clouds
  • Heterogeneous HW (POWER and x86)
  • Long running and batch jobs
  • OpenStack, Docker, Mesos, Kubernetes
  • Container clouds with Accelerators (GPUs)
What problems are we trying to solve

• Enable Deep Learning in the Cloud
  • Need flexible access to hardware (GPUs)
  • Training times in hours, days, weeks, months
  • Long running inferencing services
  • Support old, new and emerging frameworks
  • Share hardware among multiple workloads and users
DL in the Cloud: State-of-the-art

- Historically DL is on-prem infrastructure and SW stack – high-performance environment
  - Baremetal GPU systems (x86 and POWER), Ethernet, IB network connectivity, GPFS
  - Spectrum LSF, MPI and RDMA support, single SW stack

- Cloud – Frees researchers & developers from infrastructure & SW Stack
  - All infrastructure from Cloud as services: GPUs, object store, NFS, SDN, etc,
  - Job submission with APIs: Torch, Caffe, Tensorflow, Theano
  - 24/7 service, elastic and resilient
  - Appropriate visibility and control
Challenges with DL on Cloud

- Data, data, data, data, ...
- Access to different hardware and accelerators (GPU, IB, ...)
- Support for different application models
- Visibility and control of infrastructure
- Dev and Ops challenges with 24/7 state full service
Journey started in 2016... promised to deliver DL on Cloud

- Excellent promise, go ahead and built a DL cloud service

  - Container support GPU: minimal or non-existent

  - The idea could have died on day 1 but failure is not an option ...

  - We chose containers with Mesos and Kubernetes to address some of these challenges

  - Developed and operated Mesos and Kubernetes based GPU Clouds for over a year

  - What follows are lessons learned from this experience
DL on Containers: DevOps challenges

• Multiple GPUs per node → multiple containers per node: need to maintain GPU <-> Container mapping (GPU Allocator)

• Images need NVIDIA Drivers: makes them non-portable (Volume Manager)

• Cluster quickly becomes heterogeneous (K80, M60, P100…): need to be able to pick GPU type (GPU Discovery)

• Fragmentation of GPUs is a real problem (Priority placement)

• Like everything else GPUs fail → must identify and remove unhealthy GPUs from scheduling (Liveness check)

• Visibility, control, and sharing (to be done)
High-level view of GPU support in containers clouds

Task Id: xxx
Resource Request: GPU: 1
...
...
...
Resource report: GPU: 2
...
...
Task Distribute GPU: 1
Allocate GPU#1
Discovery

Node
CPU
Mem
GPU1
GPU2

Job Executor
GPU isolator
Mem isolator
CPU isolator

GPU Allocator
GPU Volume Manager

Master Node
Scheduler

Insert Driver Volume
GPU Drive Volume
Health Check
GPU Allocator

- Allocator handles GPU number/device mapping
- Isolator uses cgroup(mesos)/docker(k8s) to control GPU access permission inside container

```
/dev
├── nvidia0 (data interface for GPU0)
├── nvidia1 (data interface for GPU1)
├── nvidia-ctl (control interface)
├── nvidia-uvm (unified virtual memory)
└── nvidia-uvm-tools (UVM tools, optional)
```

- Allocate/Release GPU

Request 1 GPU

NVIDIA GPU Allocator

GPU0: in use
GPU1: idle

Allocate

GPU1

NVIDIA GPU Allocator

GPU0: in use
GPU1: in use

Expose

GPU1

Container

/dev/nvidia1
/dev/nvidia-ctl
/dev/nvidia-uvm

Expose GPU 1
GPU Driver Challenges: Drivers in the container is an issue

Changes in the host driver require all containers to update their drivers!!!!
GPU Driver Challenges: NVIDIA-Docker solves it

App will not work if NVIDIA libraries and kernel module versions are not match
GPU Volume Manager

• Mimic functionality of nvidia-docker-plugin

• Finds all standard NVIDIA libraries / binaries on the host and consolidates them into a single place.

/var/lib/mesos/volumes
  └── nvidia_XXX.XX (version number)
      ├── bin
      │   └── ...
      └── lib
          └── ...
      └── lib64

• Inject volume with read-only (“ro”) to container if needed

Image label:
com.nvidia.volumes.needed = nvidia_driver

GPU container
/usr/local/nvidia
GPU Discovery

- Mesos-agent/kubelet auto detects GPU numbers
- Instead, we use `nvml` library to detect number of GPUs, model, etc

NVIDIA GDK (nvml library)

Compiled binary, dynamic link to nvml

Same binary works on both GPU node and non-GPU nodes

Run on GPU node

Discover GPU

Run on non-GPU node

Compile

nvml lib found

nvml lib not found
Fragmentation problem: GPU Priority Placement

- Three jobs are spread on 4 nodes
- New Job 4 needs 4 GPUs on a single node, can it run?
- Although there are 10 free GPUs

Three jobs are spread on 4 nodes

- New Job 4 needs 4 GPUs on a single node, can it run?
- Although there are 10 free GPUs
Fragmentation problem: GPU Priority Placement

- Solution: Bundle the jobs
- New Job 4 needs 4 GPUs

Bundle (Default)

- GPU node #1: J1, J2, J1, J2
- GPU node #2: J3, J4, J2, J4
- GPU node #3: J4
- GPU node #4: 4 GPUs

Used and Free status:
- Used: red
- Free: green

Jx → Job x
Fragmentation problem: GPU Priority Placement

• Solution: Bundle the jobs
• New Job 4 needs 4 GPUs on a single node

Bundle (Default)

- GPU node #1: J1, J2
- GPU node #2: J3, J2
- GPU node #3: J4, J4
- GPU node #4: Unused

Used
Free
Jx → Job x
Fragmentation problem: GPU Priority Placement

- GPU priority scheduler can bundle/spread GPU tasks across the cluster
  - **Bundle**: Reserve large idle GPU nodes for large tasks
  - **Spread**: Distribute GPU workload over cluster
GPU Liveness Check

- GPU errors due to:
  - Insufficient power supply
  - Hardware damage
  - Over heating
  - Software bugs
  - ...

- GPU liveness check
  - Agent will probe GPU through nvml periodically
  - If GPU probe fails, mark GPU as unavailable, no future applications are scheduled on that GPU

GPU failure sample
Implementation in Mesos and Kubernetes

- Open-source cluster manager
- Enables siloed applications to be consolidated on a shared pool of resources
- Rich framework ecosystem
- Emerging GPU support
GPU Support on Apache Mesos

**Resource Definition:**

cpus:8; mem:1024; disk:65536; gpus:4

**Scheduler Framework (Marathon)**
- Mesos Master

**Mesos Agent**

**Containerizer API**
- Composing Containerizer
- Docker Containerizer
- (Unified) Mesos Containerizer

**Isolator API**
- CPU
- Memory
- GPU

**NVIDIA GPU Allocator**

**NVIDIA Volume Manager**
Kubernetes

- Open source orchestration system for Docker containers
- Handles scheduling onto nodes in a compute cluster
- Actively manages workloads to ensure that their state matches the user’s declared intentions
- Emerging support for GPUs
GPU Support on Kubernetes -- Upstream

- Basic multi-GPU support in release 1.6 upstream
  - GPU discovery
  - GPU allocator
  - GPU isolator
GPU Support on Kubernetes – Internal DL cloud

- **GPU resource type extension**
  - kube-apiserver

- **GPU priority scheduler**
  - kubelet

- Docker API / CRI

- **GPU number display for PODs/Nodes**
  - kubectl

- **Community GPU Support**
  - NVIDIA GPU Allocator
  - NVIDIA Volume Manager
  - GPU Liveness Check

- **Our Extension**
  - libnvidia-ml library
    - Dynamic loading

- **Linux devices cgroup**

- **GPU number display for PODs/Nodes**
Demo
## Status of GPU Support in Mesos and Kubernetes

<table>
<thead>
<tr>
<th>Function/Feature</th>
<th>Nvidia-docker</th>
<th>Mesos</th>
<th>k8s upstream</th>
<th>k8s IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPUs exposed to Dockerized applications</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>GPU vendor</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
<td>NVIDIA</td>
</tr>
<tr>
<td>Support Multiple GPUs per node</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>No GPU driver in container</td>
<td>✔️</td>
<td>✔️</td>
<td>Future</td>
<td>✔️</td>
</tr>
<tr>
<td>Multi-node management</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>GPU Isolation</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>GPU Auto-discovery</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️ (no nvml)</td>
<td>✔️</td>
</tr>
<tr>
<td>GPU Usage metrics</td>
<td>✔️</td>
<td>On-going</td>
<td>Future</td>
<td>On-going</td>
</tr>
<tr>
<td>Heterogeneous GPUs in a cluster</td>
<td>✗</td>
<td>✔️</td>
<td>Partial</td>
<td>✔️</td>
</tr>
<tr>
<td>GPU sharing</td>
<td>✔️ (No control)</td>
<td>✔️ (No control)</td>
<td>Future</td>
<td>Future</td>
</tr>
<tr>
<td>GPU liveness check</td>
<td>✗</td>
<td>Future</td>
<td>Future</td>
<td>✔️</td>
</tr>
<tr>
<td>GPU advanced scheduling</td>
<td>✗</td>
<td>Future</td>
<td>Future</td>
<td>✔️</td>
</tr>
<tr>
<td>Compatible with NVIDIA official docker image</td>
<td>✔️</td>
<td>✔️</td>
<td>Future</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Our DL service

- **Mesos/Marathon GPU support**
  - Support NVIDIA GPU resource management
  - Developing and operating deep learning and AI Vision internal services
  - Code contributed back to community
  - Presentations at MesosCon EU 2016 and MesosCon Asia 2016

- **Kubernetes GPU support**
  - Support NVIDIA GPU resource management
  - Developing and operating deep learning and AI Vision internal services
  - GPU support in IBM Spectrum Conductor for Containers (CfC)
  - Engagement with community to bring several of these features
IBM Spectrum Conductor for Containers

- Community Edition available now! Free to download and use as you wish (optional paid support)
  - Customer-managed, on-premises Kubernetes offering from IBM on x86 or Power
  - Simple container based installation with integrated orchestration & resource management
  - Authorization and access control (built-in user registry or LDAP)
  - Private Docker registry
  - Dashboard UI
  - Metrics and log aggregation
  - Calico networking
  - Pre-populated app catalog
  - GPU support in 1.1; paid support in 1.2 (June)

- Learn more and register on our community page: [http://ibm.biz/ConductorForContainers](http://ibm.biz/ConductorForContainers)

Demo on YouTube
What is left to do

- GPU topology-aware scheduling (on-going)
  - Support GPU topology-aware scheduling to optimize performance

- GPU live metric collection (on-going)
  - Collect GPU live metrics (i.e., live core/mem usage)

- Support CRI interface (in plan)
  - kubernetes moves to CRI after release 1.6, we will not depend on docker API

- Support libnvidia-container (under discussion)
  - Use libnvidia-container instead of nvidia-docker logic to manage GPU container
New OpenPOWER Systems with NVLink

S822LC “Minsky”:
2 POWER8 CPUs with 4 NVIDIA® Tesla® P100 GPUs hooked to CPUs using NVIDIA’s NVLink high-speed interconnect

http://www-03.ibm.com/systems/power/hardware/s822lc-hpc/index.html
Enabling Accelerators/GPUs on OpenPOWER

Containers and images

NVIDIA DOCKER

IBM Spectrum Conductor

kubernetes

Apache MESOS

Caffe

NVIDIA Caffe

IBM Caffe

torch

TensorFlow

Theano

Chainer

OpenPOWER™

Accelerators

Spark

Lightning Fast Cluster Computing
- Requirements, requirements, requirements
- Comment on issues
- Hack it and PR
- Review PRs
Summary and Next Steps

• Cognitive, Machine and Deep Learning workloads are everywhere

• Containers can be leveraged with accelerators for agile deployment of these new workloads

• Docker, Mesos and Kubernetes are making rapid progress to support accelerators

• OpenPOWER and this emerging cloud stack makes it the preferred platform for Cognitive workloads

• Join the community, share your requirements, and contribute code
Community Activities: Mesos

• GPU features on Mesos/Marathon have been supported in upstream
  • GPU for Mesos Containerizer support added after Mesos 1.0
  • GPU support added after Marathon v1.3
  • GPU usage: http://mesos.apache.org/documentation/latest/gpu-support/

• Companies collaborating on GPU support on Mesos/Marathon

• Collaborators
  • Kevin Klues
  • Rajat Phull
  • Guangya Liu
  • Qian Zhang
  • Benjamin Mahler
  • Vikrama Ditya
  • Yong Feng
  • Yu Bo Li
  • Seetharami Seelam
Community Activities: Kubernetes

- Multi-GPU support starting in Kubernetes 1.6

- Collaborators
  - Vish Kannan
  - David Oppenheimer
  - Christopher M Luciano
  - Felix Abecassis
  - Derek Carr
  - Yu Bo Li
  - Seetharami Seelam
  - Hui Zhi
  - ...
Thank You

Contact:
sseelam@us.ibm.com
liyubobj@cn.ibm.com