IBM Research

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

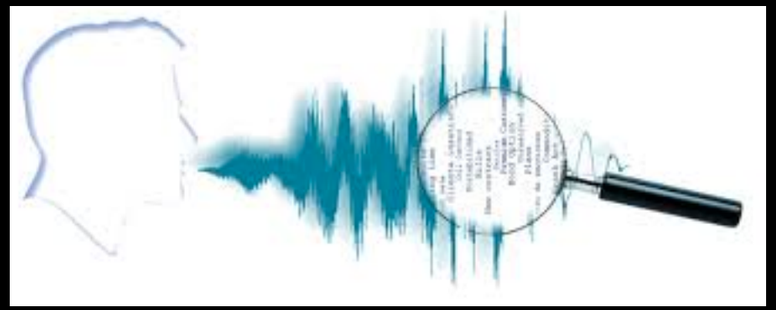
Dr. Seelam is a Research Staff Member at the T. J. Watson Research Center. He is an expert at delivering hardware, middleware, applications as-a-service using containers. He delivered Autoscaling, Business Rules, Containers on Bluemix and multiple others internally.

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



Speech

Vision

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

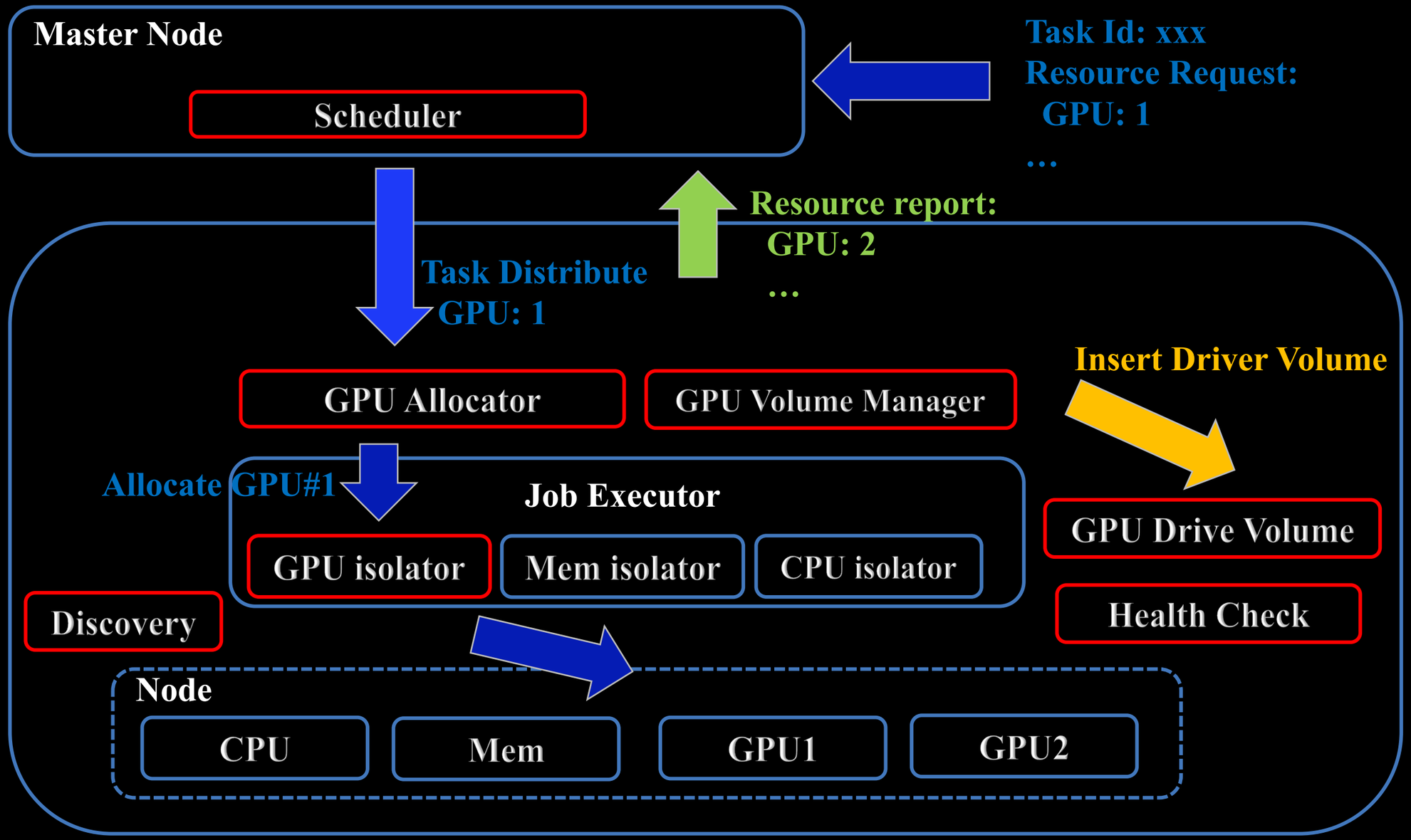
Journey started in 2016… promised to deliver DL on Cloud

- 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



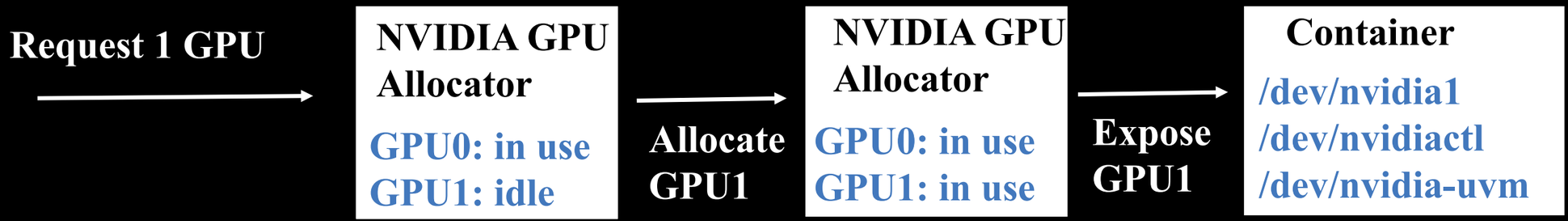
GPU Allocator

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

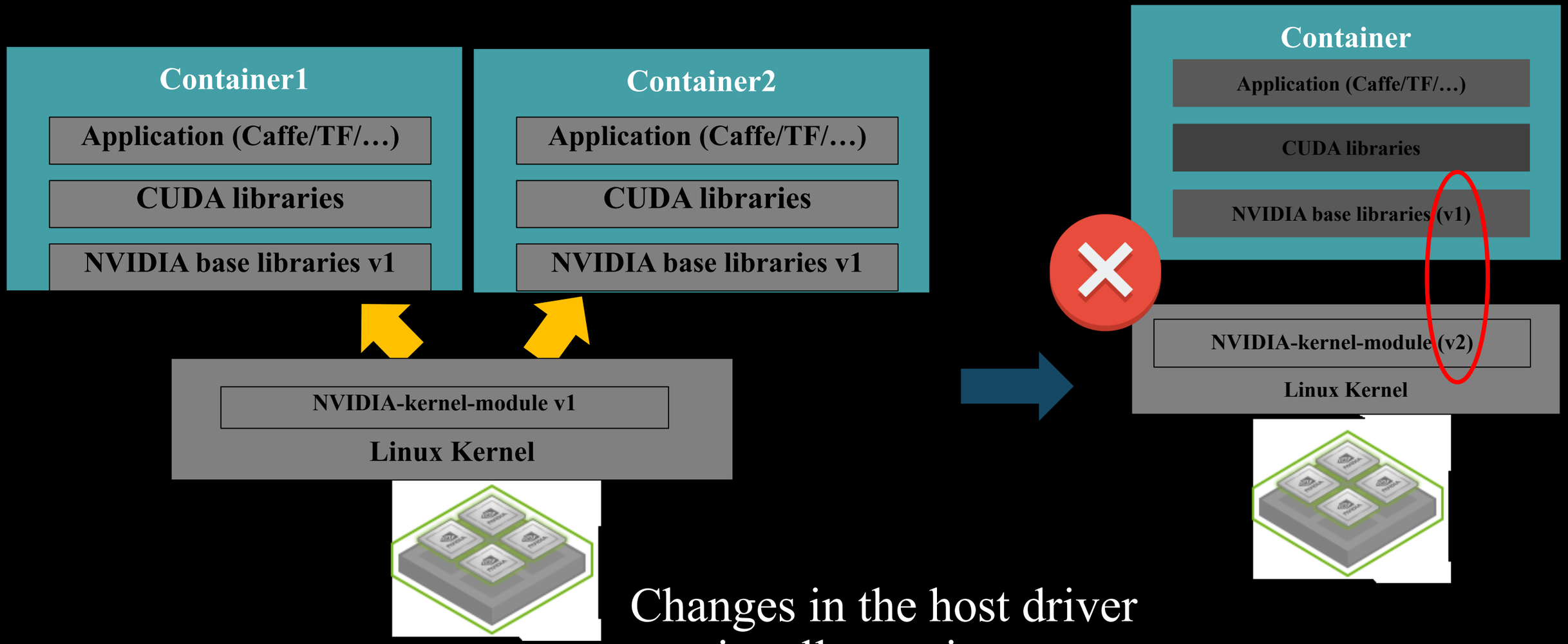
```
/dev
├── nvidia0          (data interface for GPU0)
├── nvidia1          (data interface for GPU1)
├── nvidiactl        (control interface)
├── nvidia-uvm       (unified virtual memory)
└── nvidia-uvm-tools (UVM tools, optional)
```

Expose GPU 1

- Allocate/Release GPU



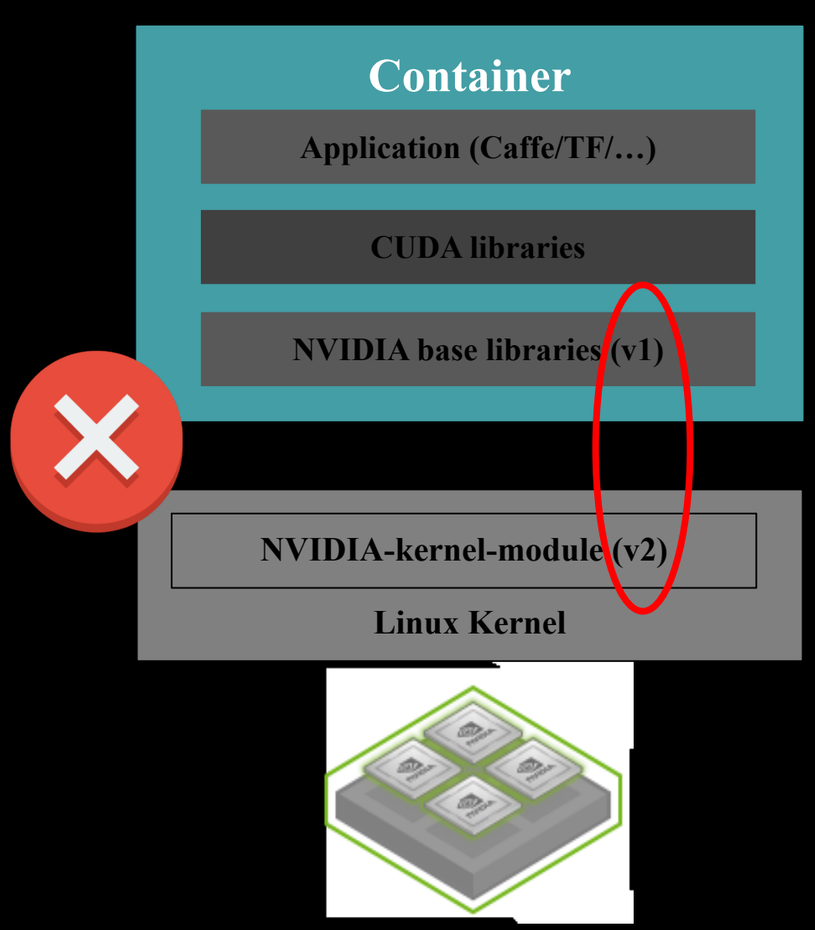
GPU Driver Challenges: Drivers in the container is an issue

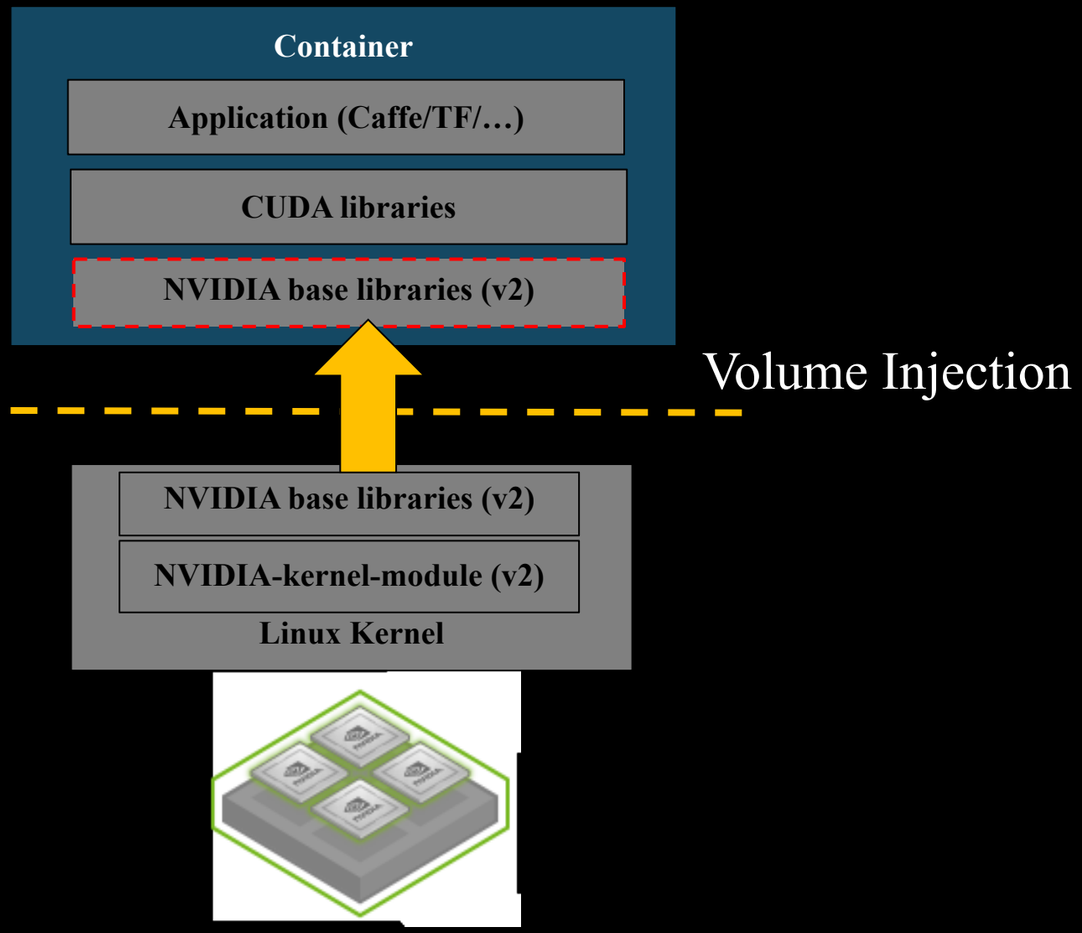


Changes in the host driver require all containers to update their drivers!!!!

Credit to Kevin Klues, Mesosphere

GPU Driver Challenges: NVIDIA-Docker solves it

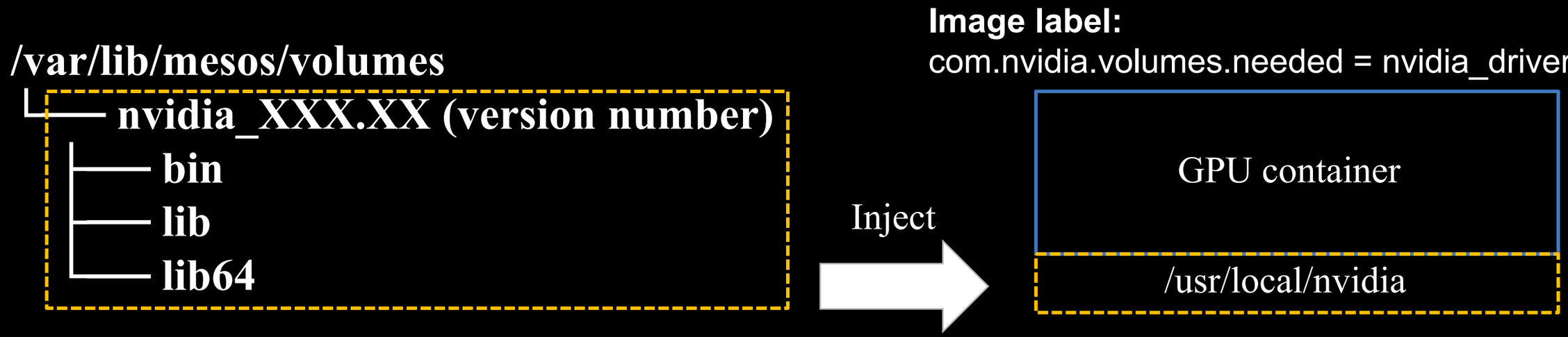




App will not work if NVIDIA libraries
and kernel module versions are not match

GPU Volume Manager

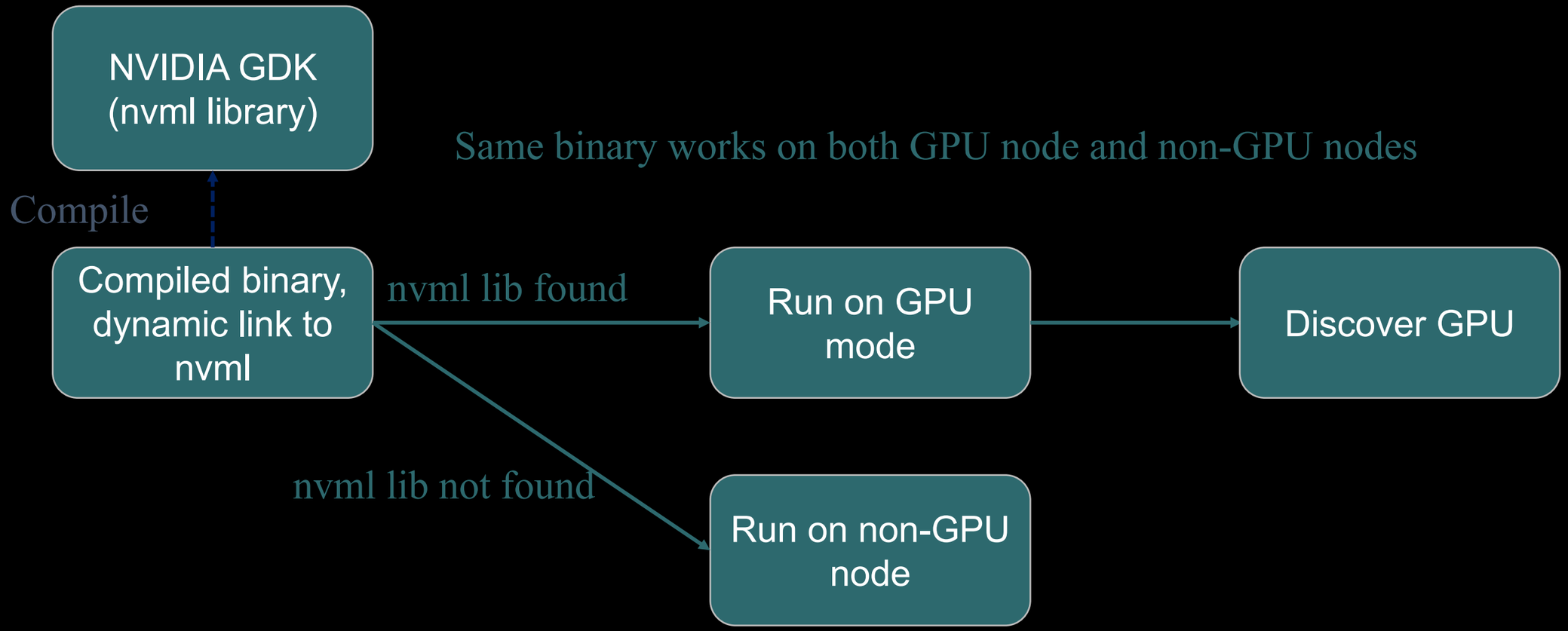
- **Mimic functionality of nvidia-docker-plugin**
  - Finds all standard NVIDIA libraries / binaries on the host and consolidates them into a single place.



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

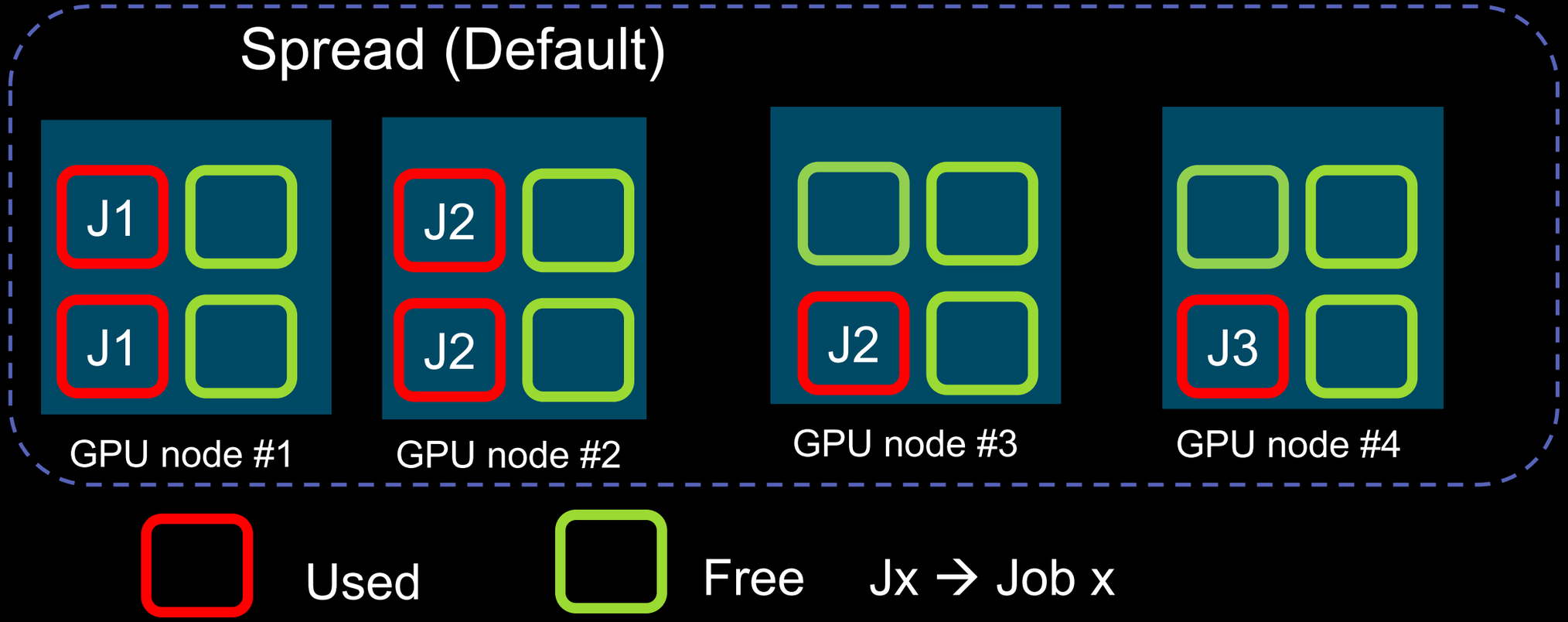
GPU Discovery

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



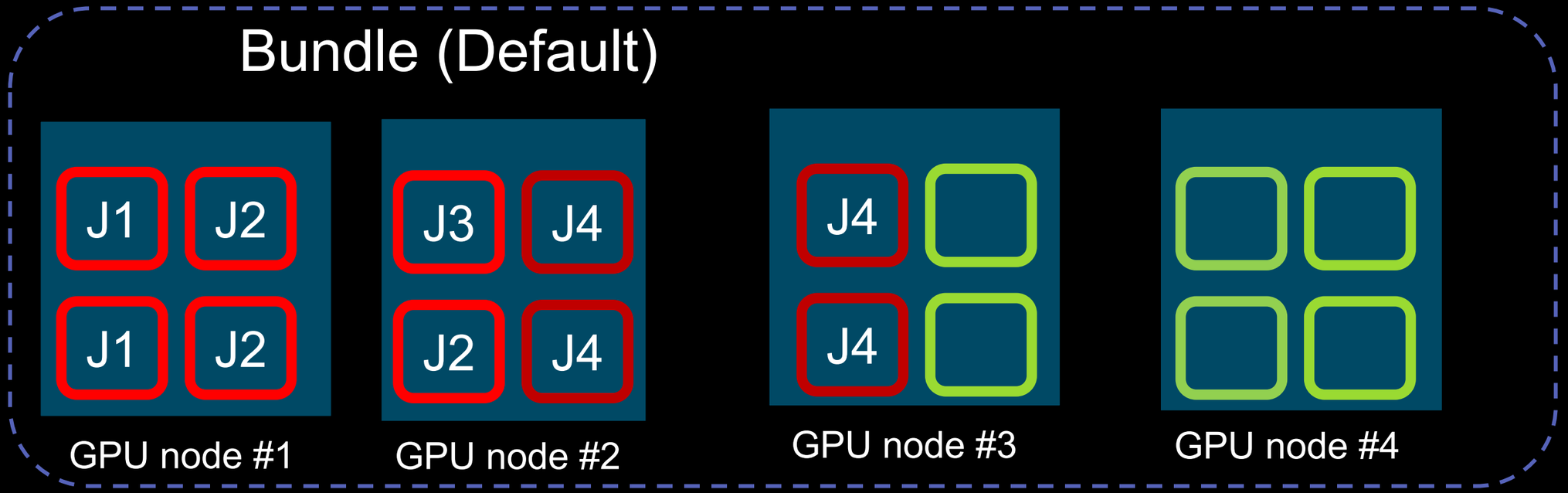
Fragmentation problem: GPU Priority Placement

- 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



Fragmentation problem: GPU Priority Placement

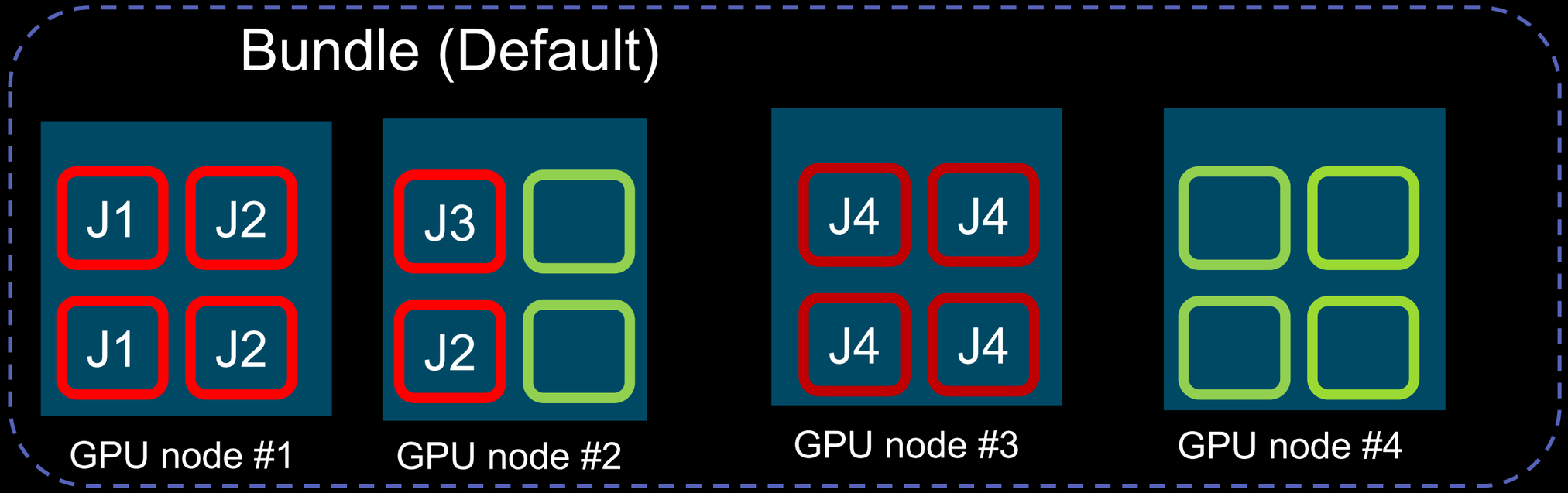
- Solution: Bundle the jobs
- New Job 4 needs 4 GPUs



Used Free Jx → Job x

Fragmentation problem: GPU Priority Placement

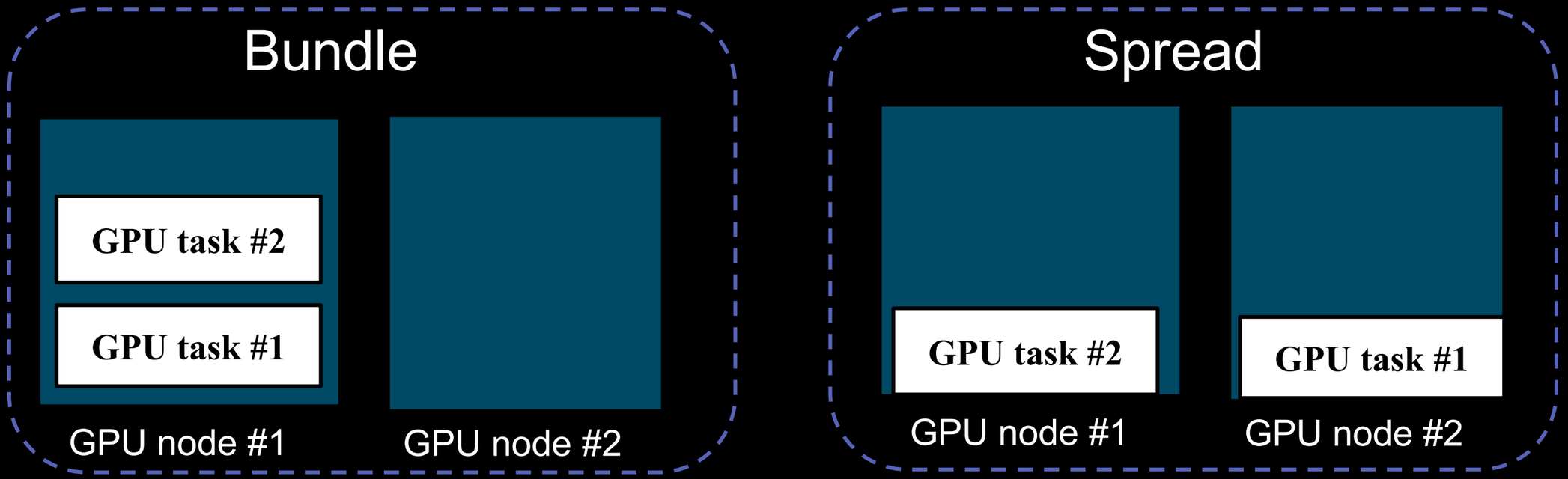
- Solution: Bundle the jobs
- New Job 4 needs 4 GPUs on a single node



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

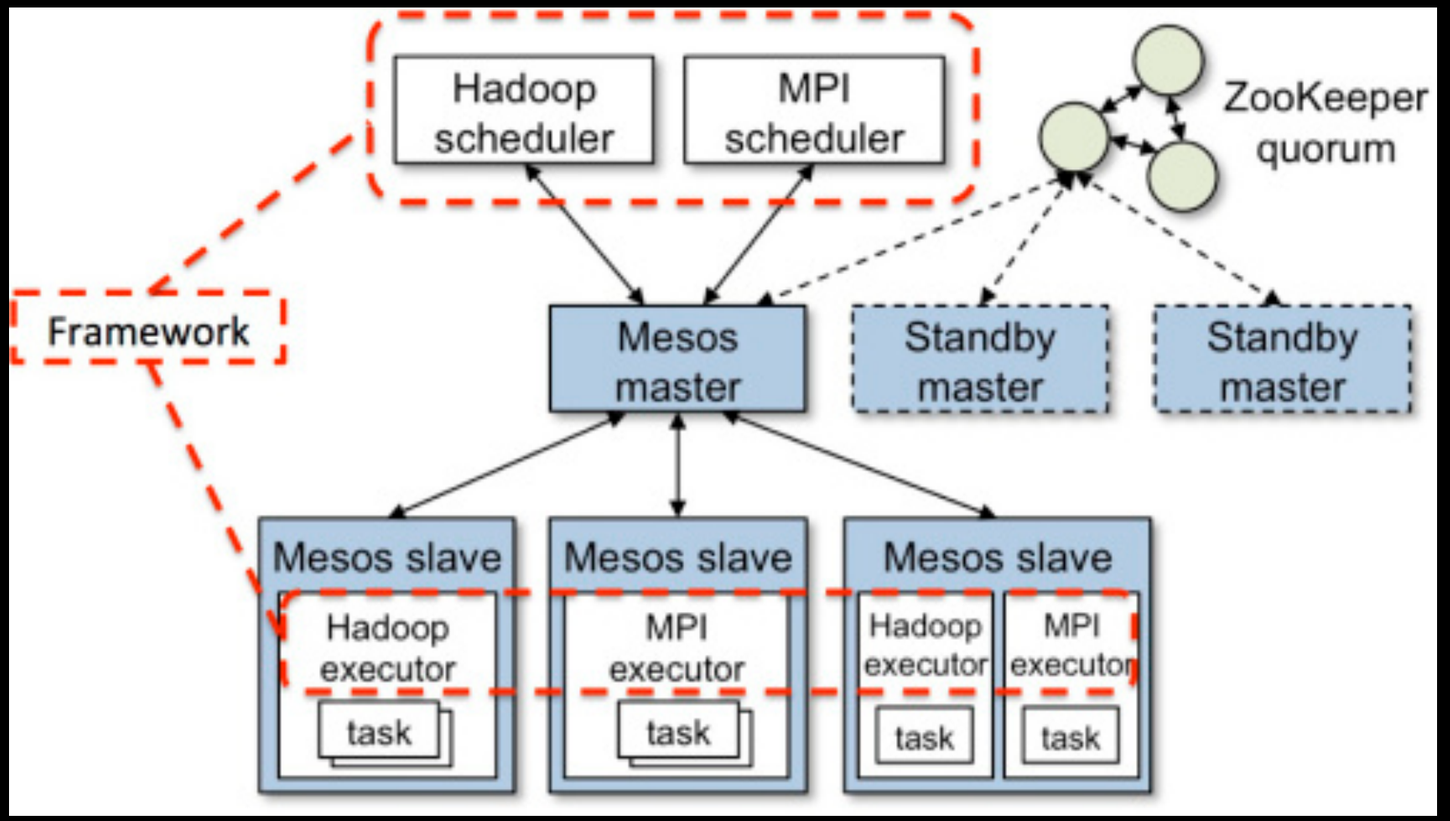
- GPU errors due to:
  - Insufficient power supply
  - Hardware damage
  - Over heating
  - Software bugs
  - ...

```
+-----------------------------------------------------+
| NVIDIA-SMI 352.39 Driver Version: 352.39 |
|-------------------------------+----------------------+----------------------+
| GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC |
| Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
|===============================+======================+======================|
| 0 ERR! ERR! | ERR! ERR! | ERR! |
|ERR! ERR! ERR! ERR! / ERR! | 22MiB / 11519MiB | ERR! ERR! |
+-------------------------------+----------------------+----------------------+
| 1 Tesla K80 On | 0000:06:00.0 Off | 0 |
| N/A 45C P8 69W / 150W | 22MiB / 11519MiB | 0% Default |
+-------------------------------+----------------------+----------------------+
```

GPU failure sample

- 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

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

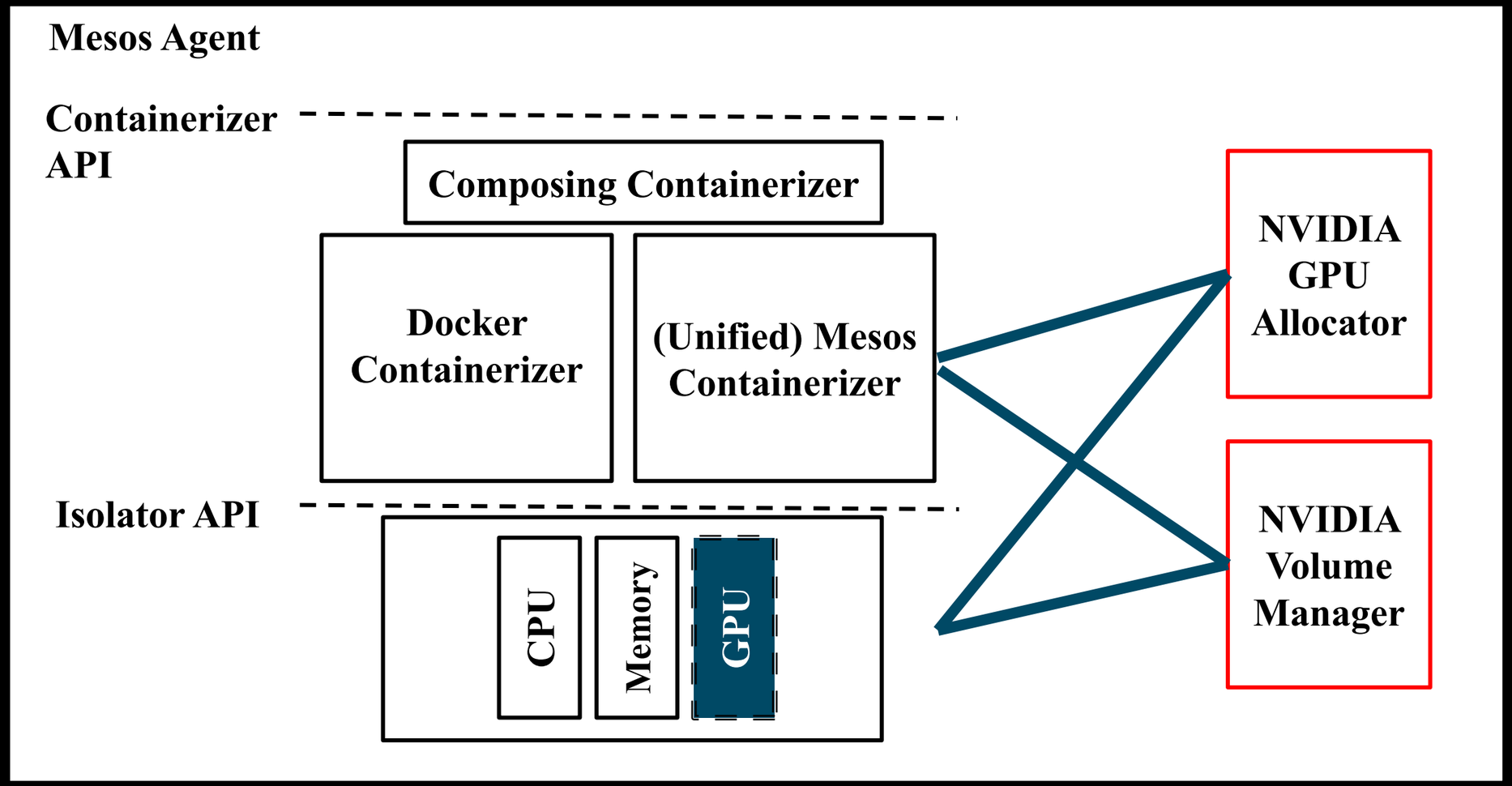
GPU Support on Apache Mesos

Scheduler Framework (Marathon)

Mesos Master

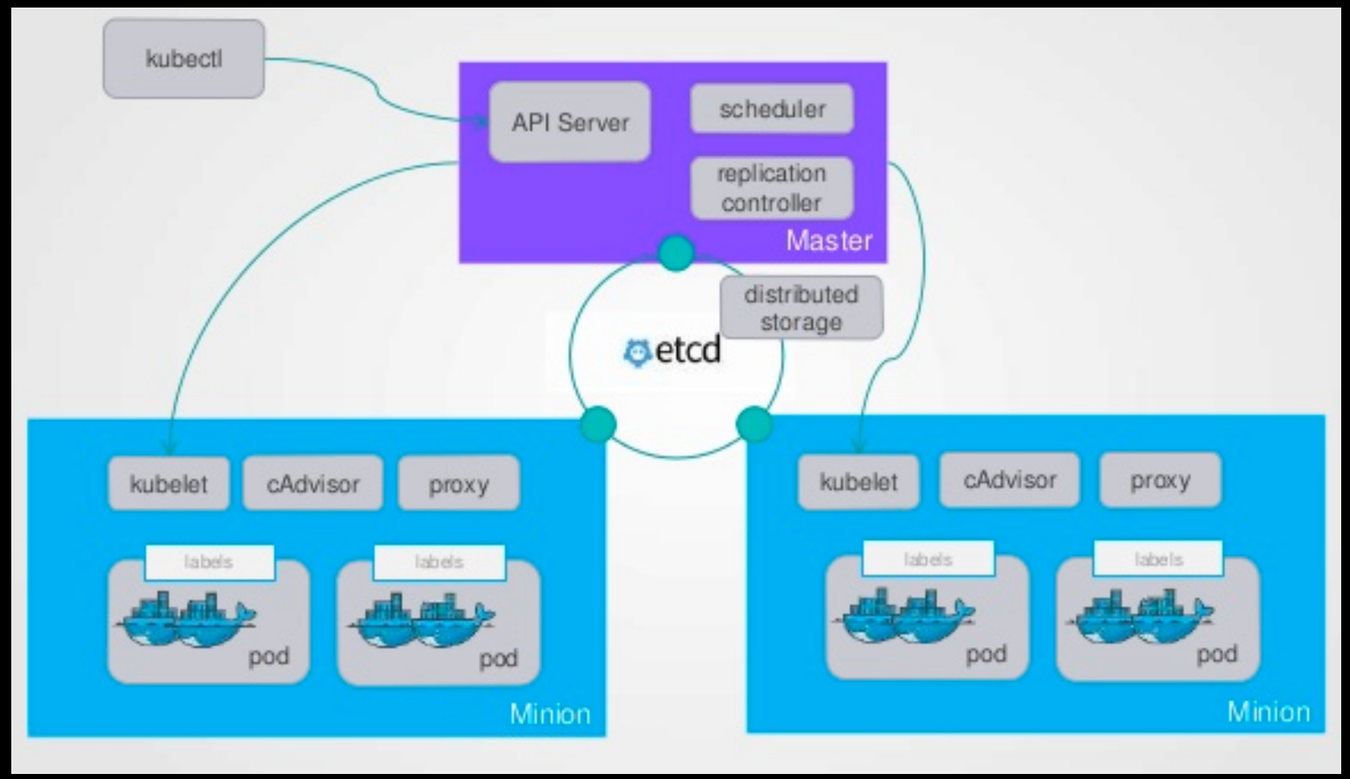
Resource Definition:

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



Kubernetes

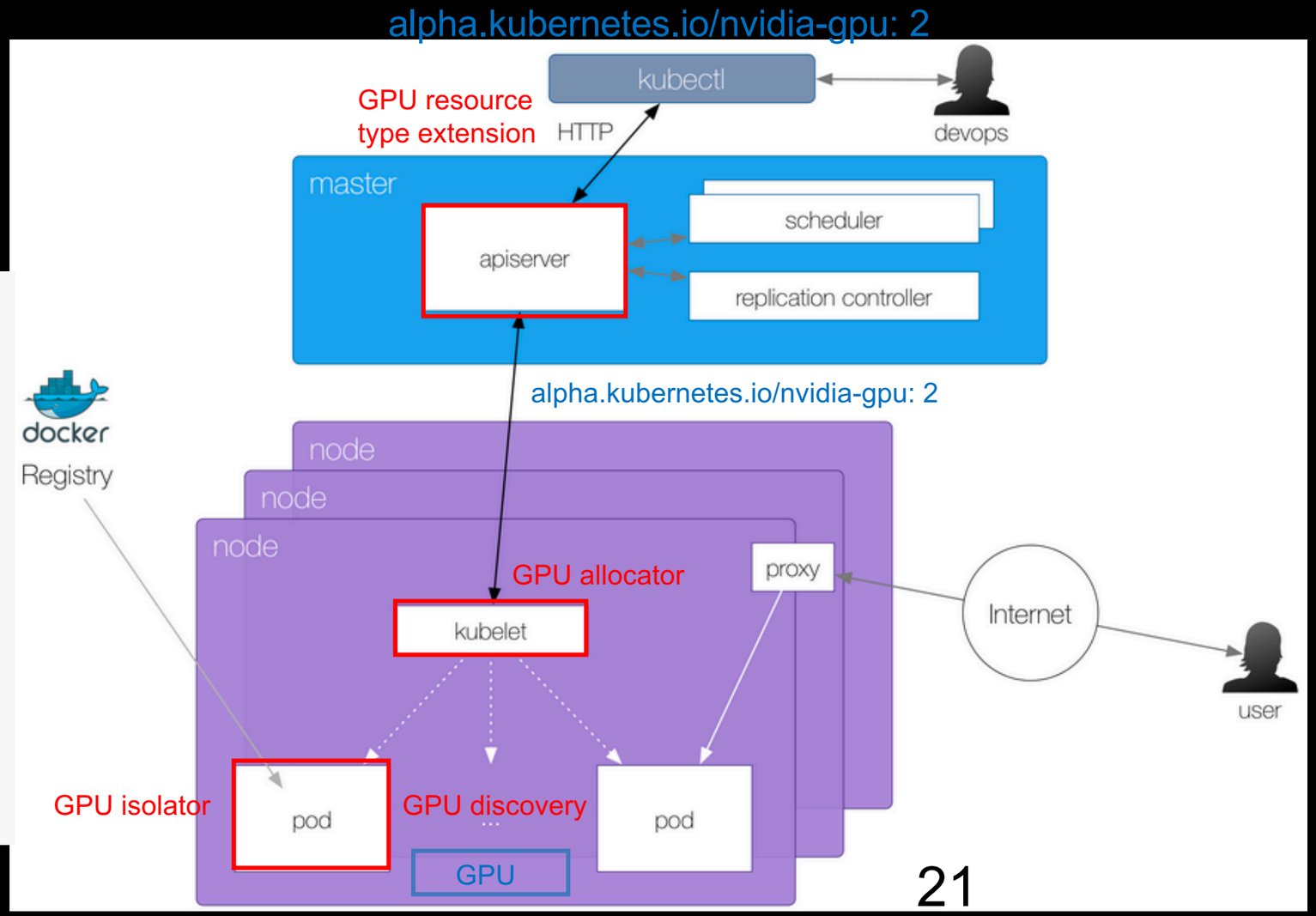
- Open source orchestration system for Docker containers
- Handles scheduling onto nodes in a compute cluster
- Actively manages workloads to ensure that their state matches the user's declared intentions
- Emerging support for GPUs



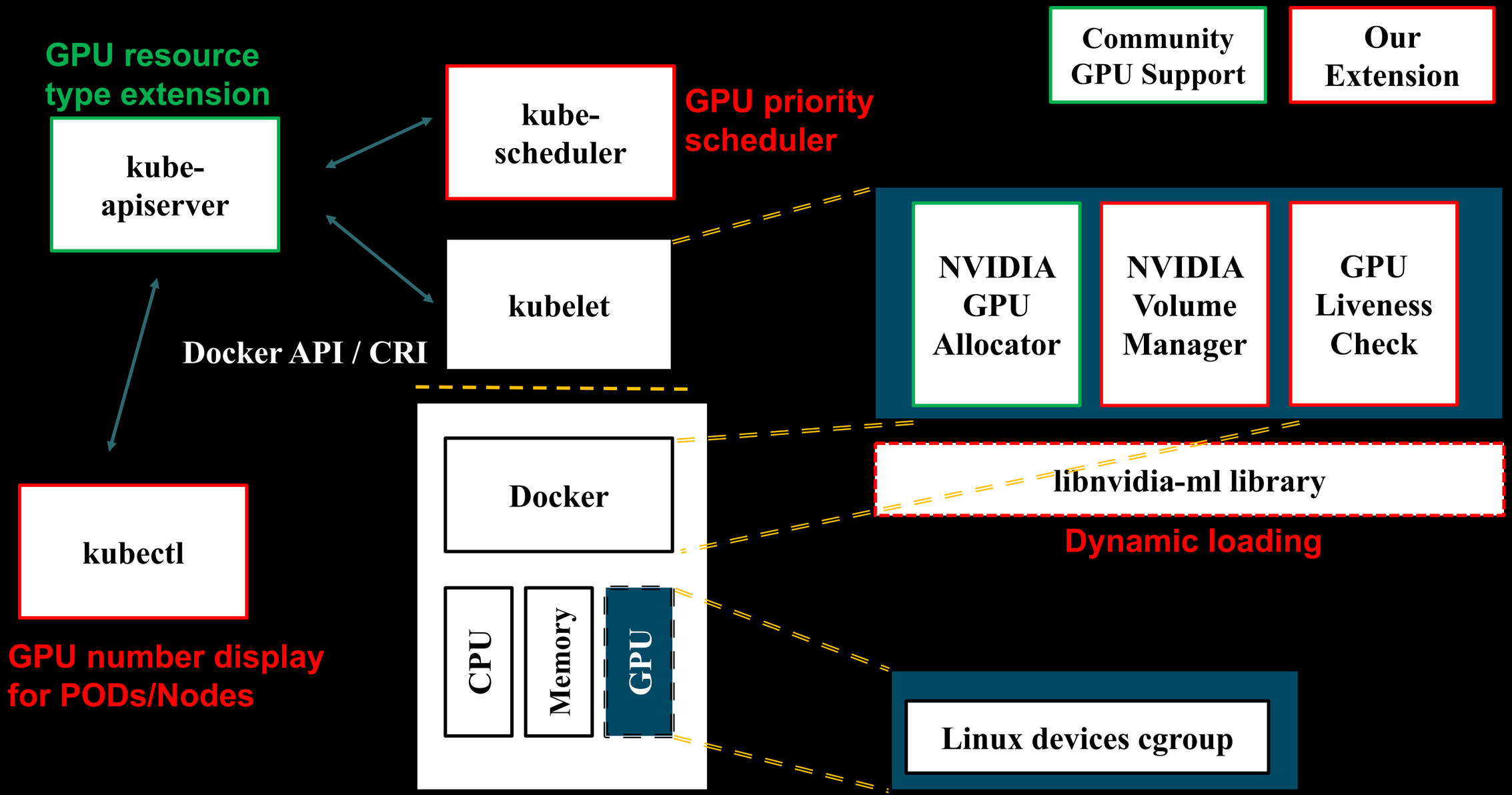
GPU Support on Kubernetes -- Upstream

- Basic multi-GPU support in release 1.6 upstream
  - GPU discovery
  - GPU allocator
  - GPU isolator

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: gpu-demo
spec:
  replicas: 1
  template:
    metadata:
      labels:
        run: gpu-demo
    spec:
      containers:
      - name: gpu-demo
        image: nvidia/cuda:7.5-runtime
        command: ["/bin/sh", "-c"]
        args: ["nvidia-smi && tail -f /dev/null"]
        resources:
          limits:
            alpha.kubernetes.io/nvidia-gpu: 1
```



GPU Support on Kubernetes – Internal DL cloud



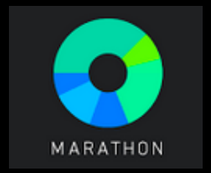
Demo

Status of GPU Support in Mesos and Kubernetes

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

Our DL service

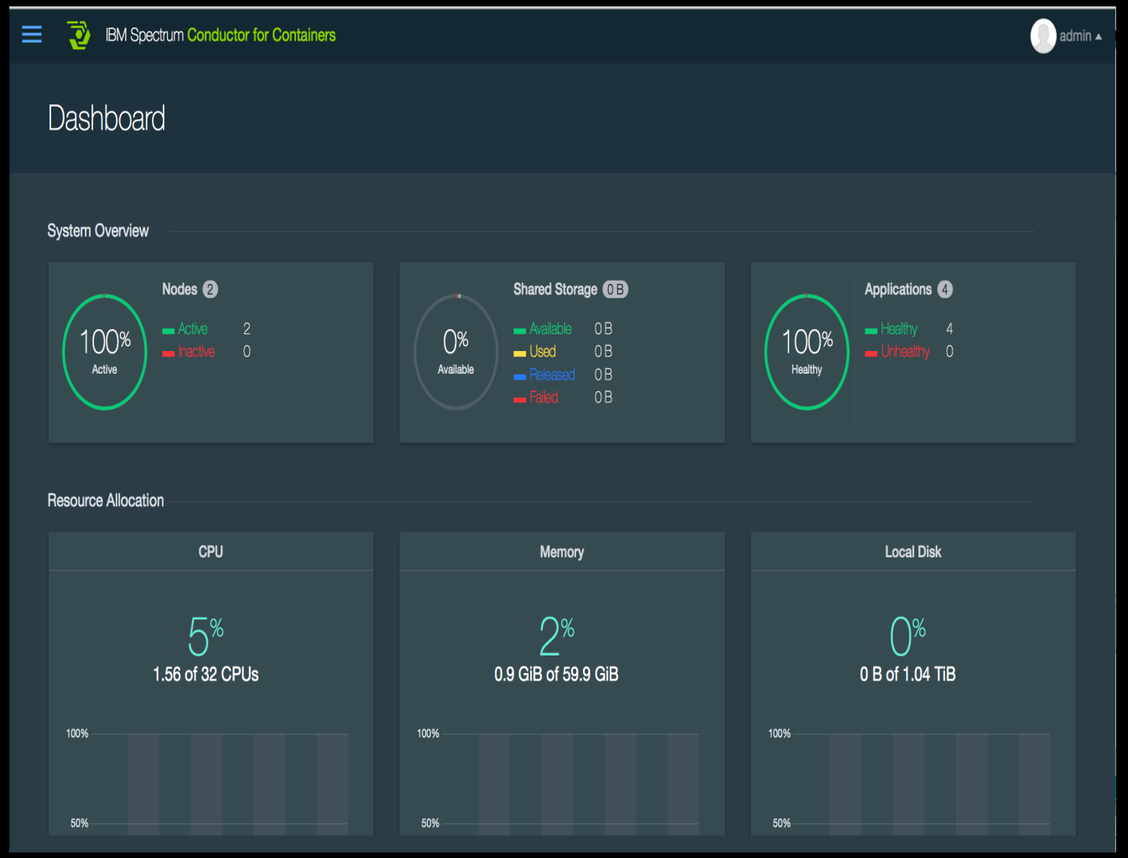
- Mesos/Marathon GPU support
  - Support NVIDIA GPU resource management
  - Developing and operating deep learning and AI Vision internal services
  - Code contributed back to community
  - Presentations at MesosCon EU 2016 and MesosCon Asia 2016



- 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**



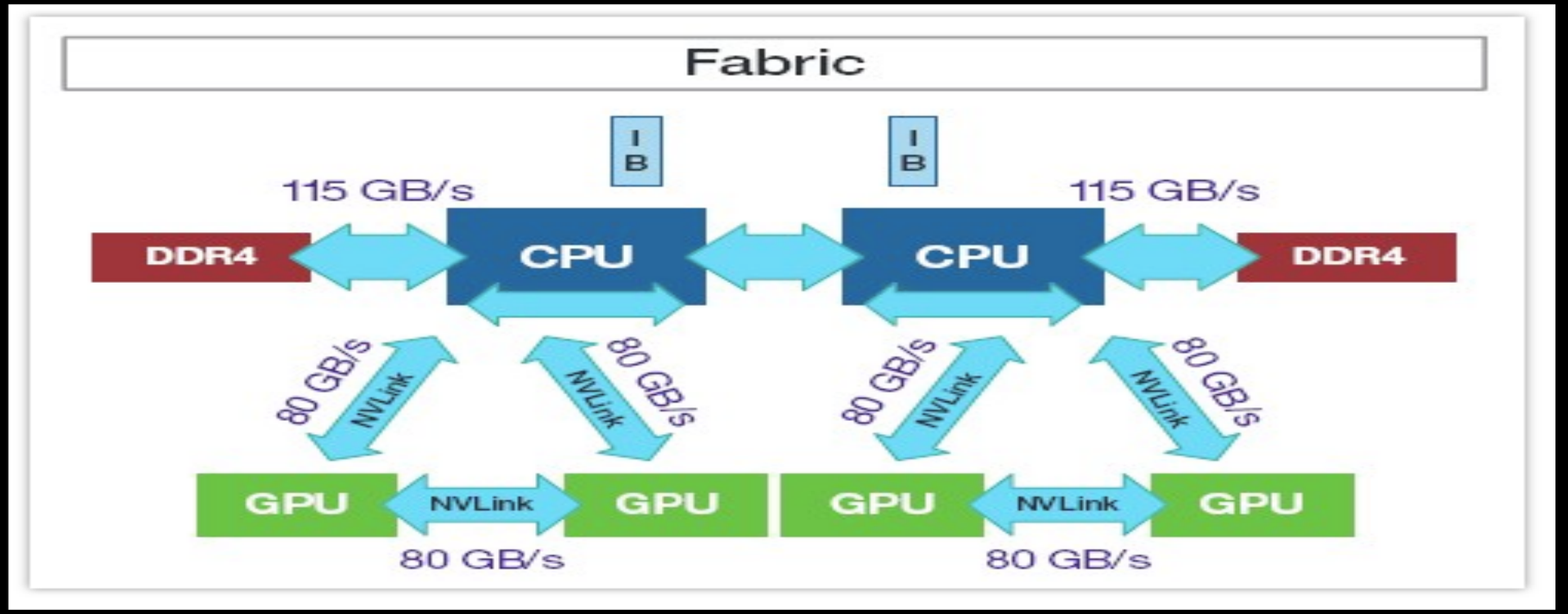
Demo on YouTube

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

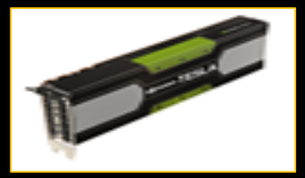
S822LC “Minsky”:
2 POWER8 CPUs with 4 NVIDIA® Tesla® P100 GPUs hooked to CPUs using NVIDIA’s NVLink high-speed interconnect
http://www-03.ibm.com/systems/power/hardware/s822lc-hpc/index.html



Enabling Accelerators/GPUs on OpenPOWER

Containers and images

NVIDIA DOCKER



Accelerators

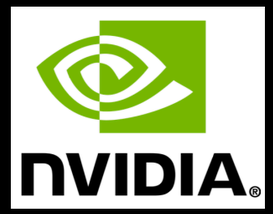
- Requirements, requirements, requirements
- Comment on issues
- Hack it and PR
- Review PRs

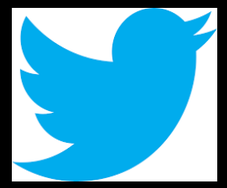
Summary and Next Steps

- 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