Nvidia GPU Support on Mesos: Bridging Mesos Containerizer and Docker Containerizer

MesosCon Asia - 2016

Yubo Li

Research Stuff Member, IBM Research - China

Email: liyubobj@cn.ibm.com

Yubo Li（李玉博）

Email: **liyubobj@cn.ibm.com**
Slack: **@liyubobj**
QQ: **395238640**

Dr. Yubo Li is a Researcher Stuff Member at IBM Research, China. He is the architect of the GPU acceleration and deep-learning as a service (DlaaS) components of 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.

Why GPUs?

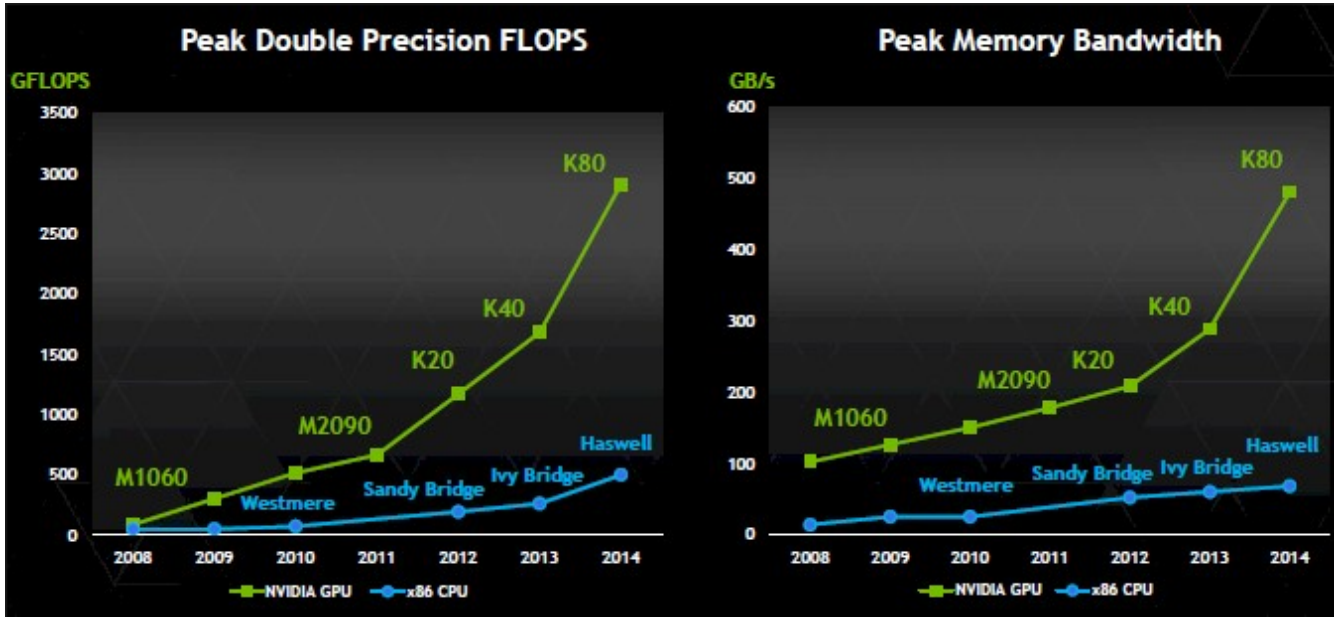
- GPUs are the tool of choice for many computation intensive applications

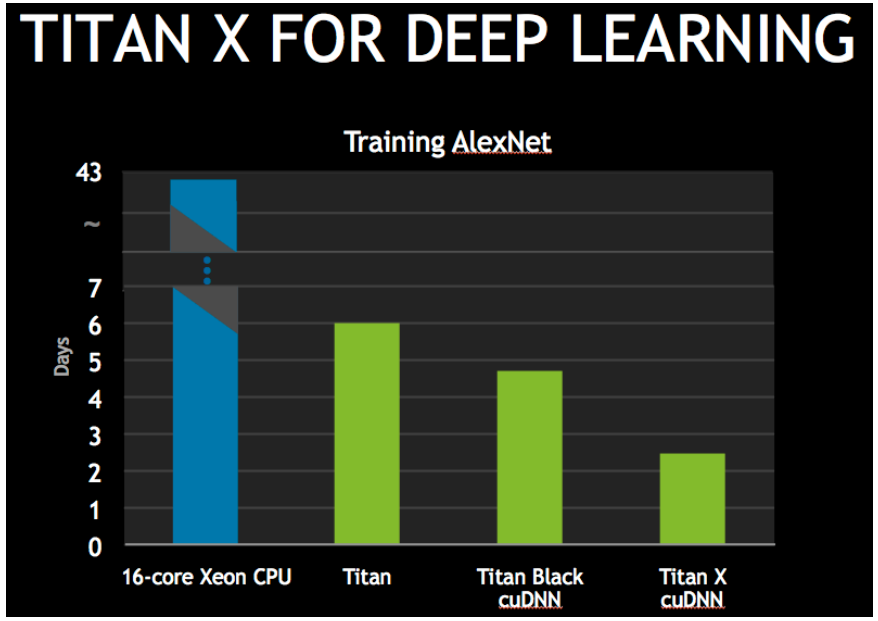
Deep Learning

Scientific Computing

Genetic Analysis

Why GPUs?





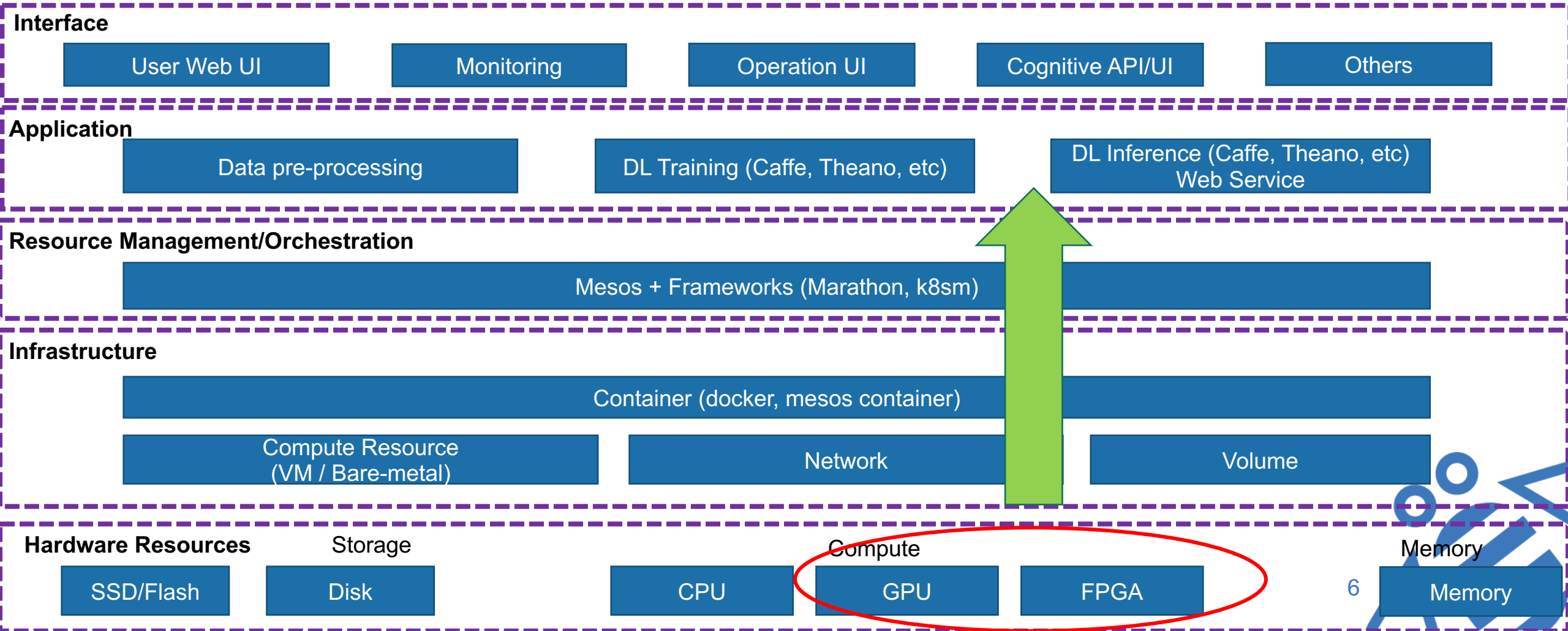
- GPU can shorten a deep learning training from tens of days to several days

Why GPUs?

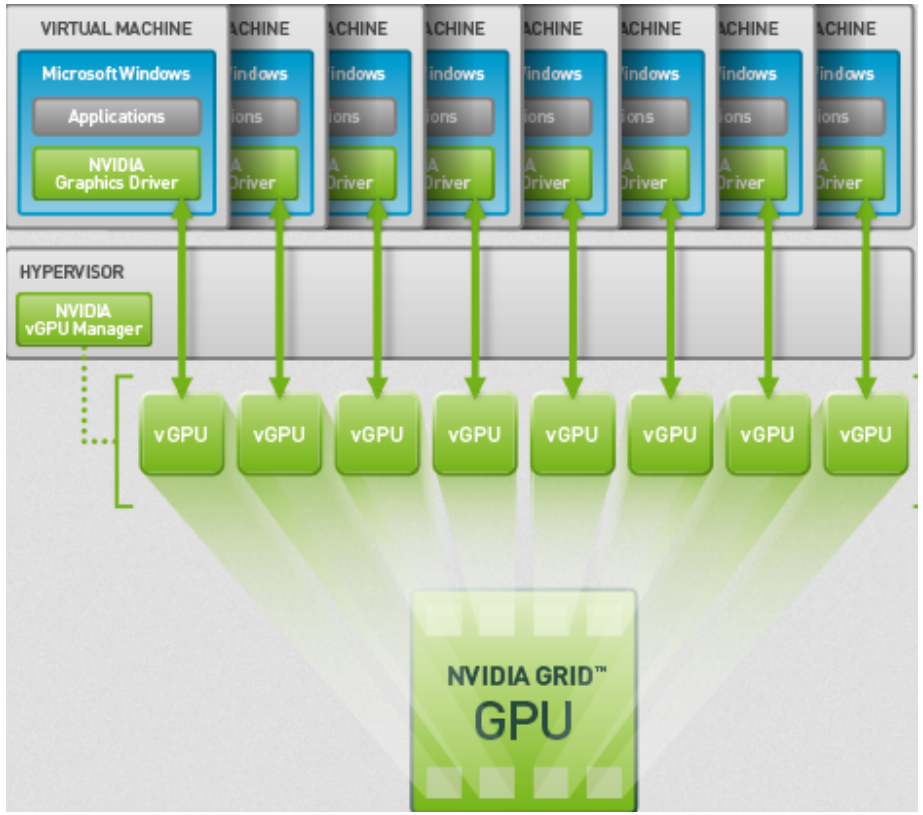
- Mesos users have been asking for GPU support for years
  - First email asking for it can be found in the dev-list archives from 2011
  - The request rate has increased dramatically in the last 9-12 months

Why GPUs?

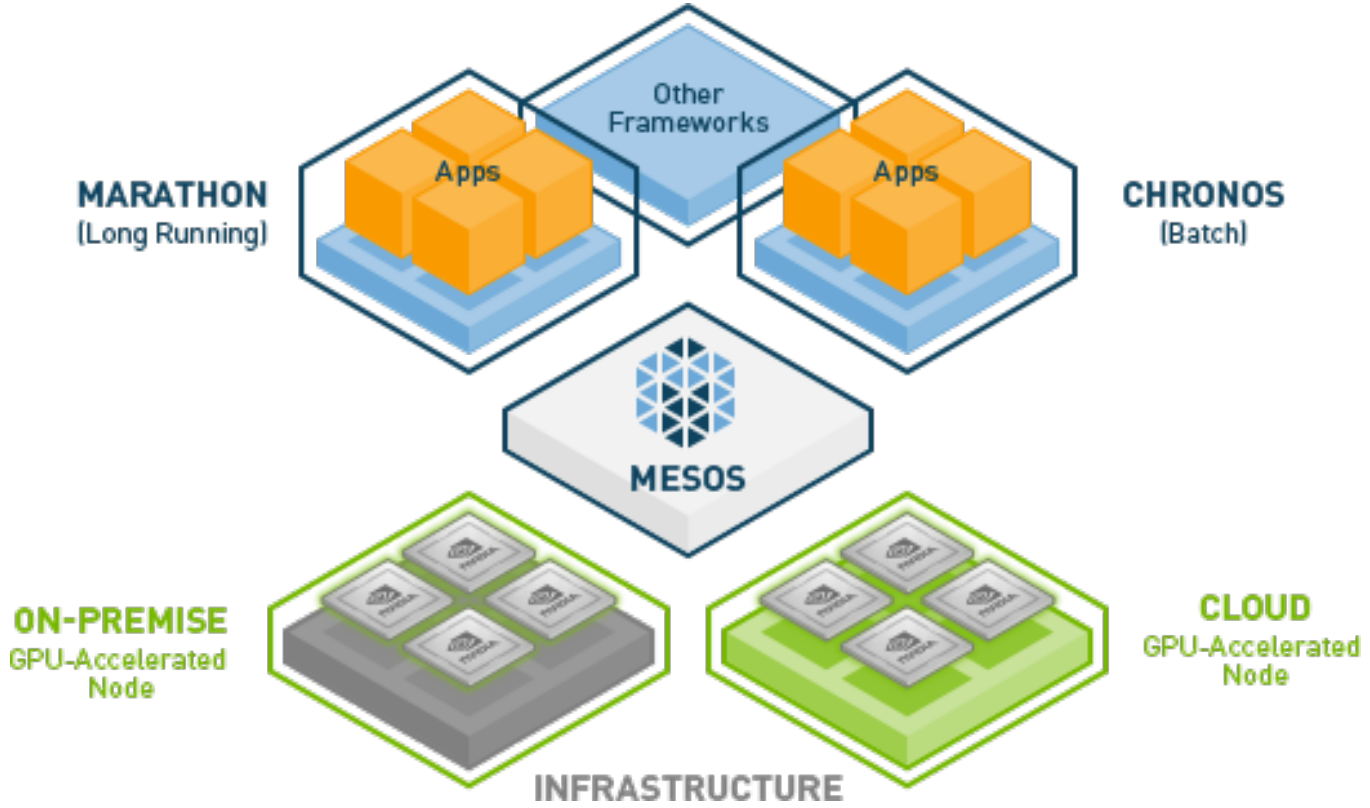
- We have internal need to support cognitive solutions on Mesos



Why GPUs?



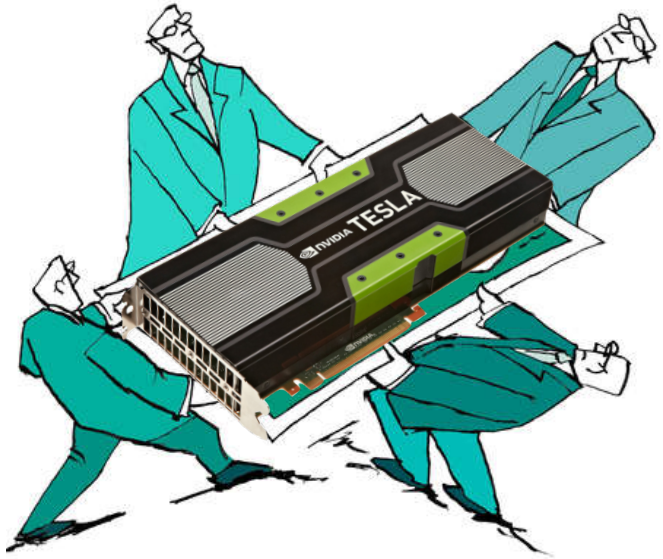
VM based GPU pass-through
Exclusively occupied GPUs



Container based GPU injection
Flexible apply and release

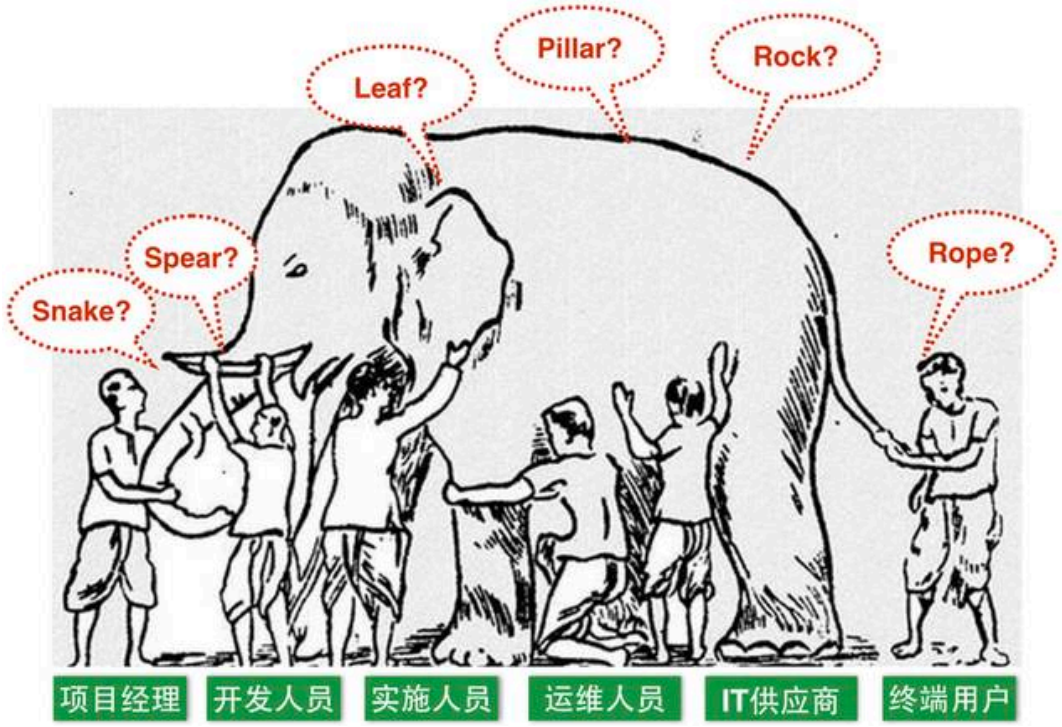
Why GPUs?

- Mesos has no isolation guarantee for GPUs without native GPU support
  - No built-in coordination to restrict access to GPUs
  - Possible for multiple frameworks / tasks to access GPUs at the same time



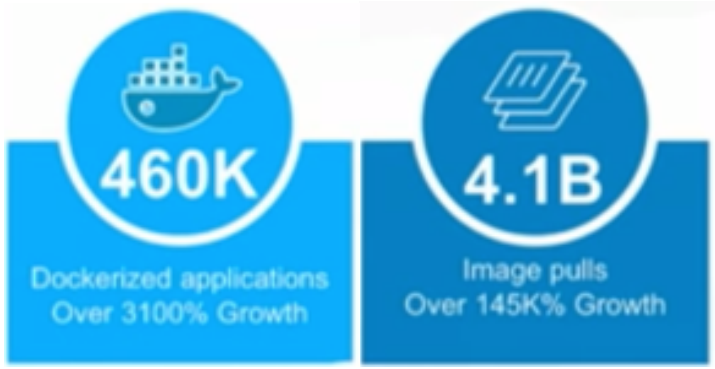
Why GPUs?

- Enterprise users want to see GPU support on container cloud
  - Deep learning / artificial intelligence need GPU as accelerator
  - Traditional HPC users turn to micro-service arch. and container cloud



Why Docker?

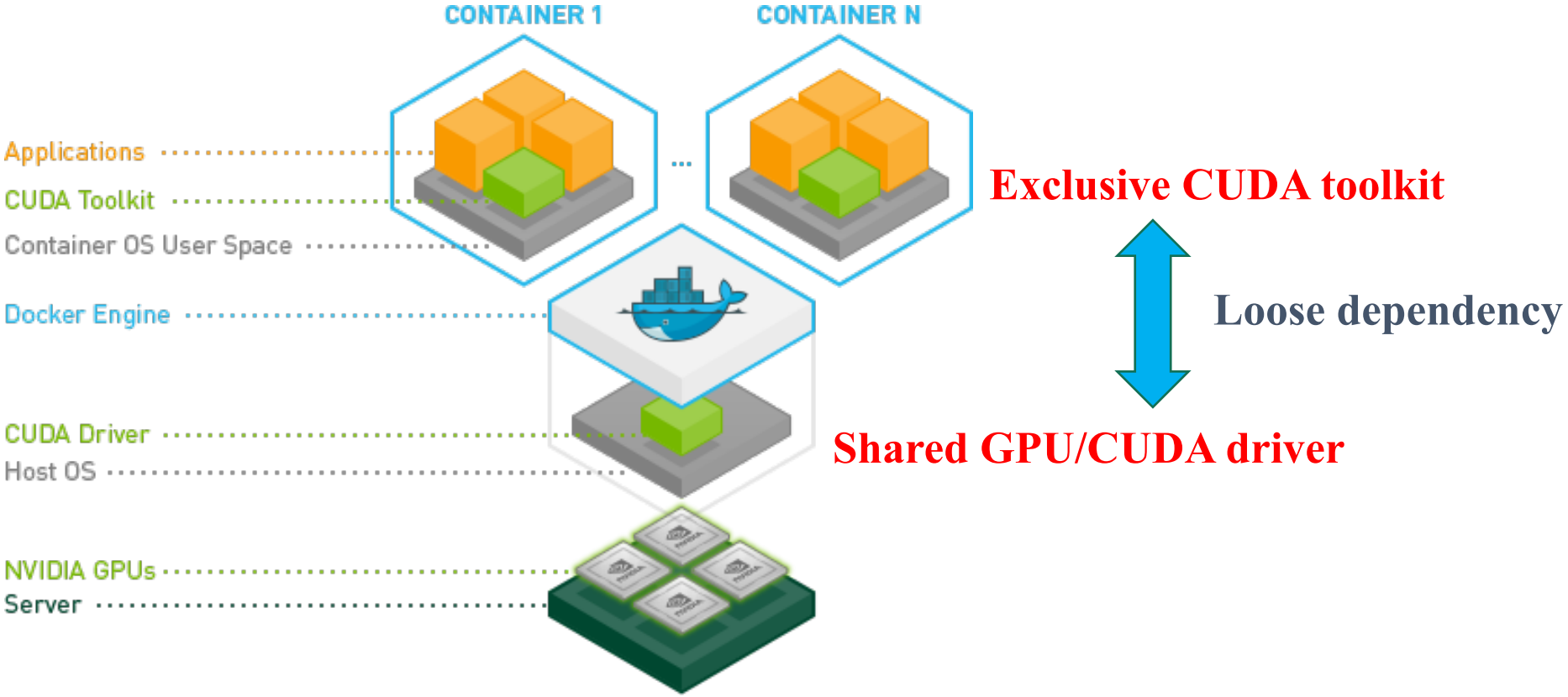
- Extremely popular image format for containers
  - Build once → run everywhere
  - Configure once → run anything



Source: DockerCon 2016 Keynote by Docker's CEO Ben Golub

Why Docker?

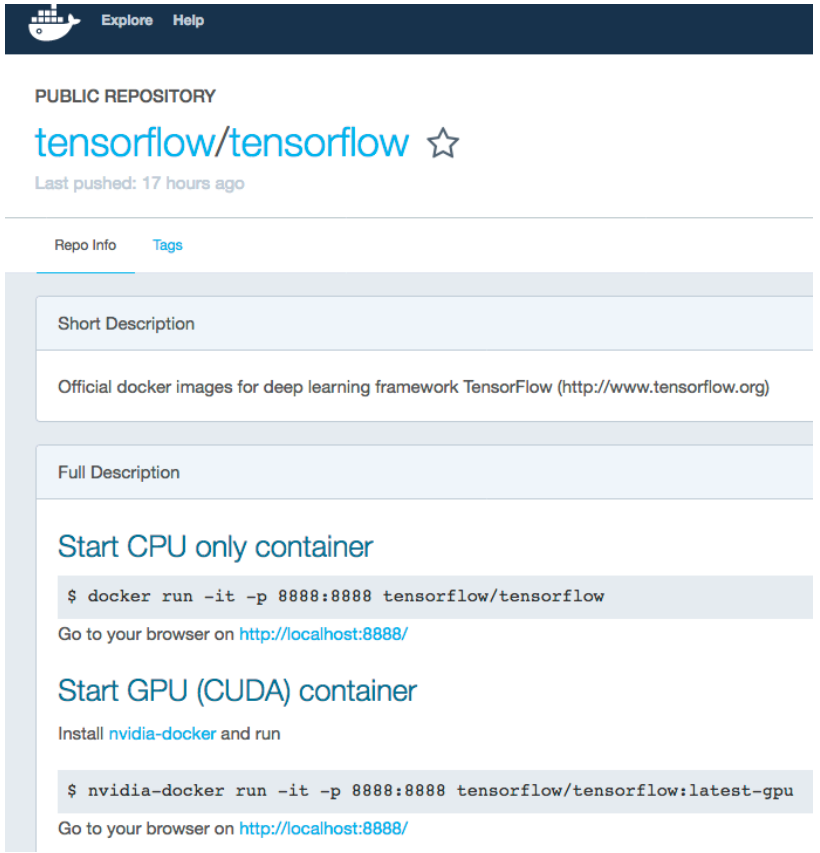
- Nvidia-docker
  - Wrap around docker to allow GPUs to be used/isolated inside docker containers
  - CUDA-ready docker images



https://github.com/NVIDIA/nvidia-docker

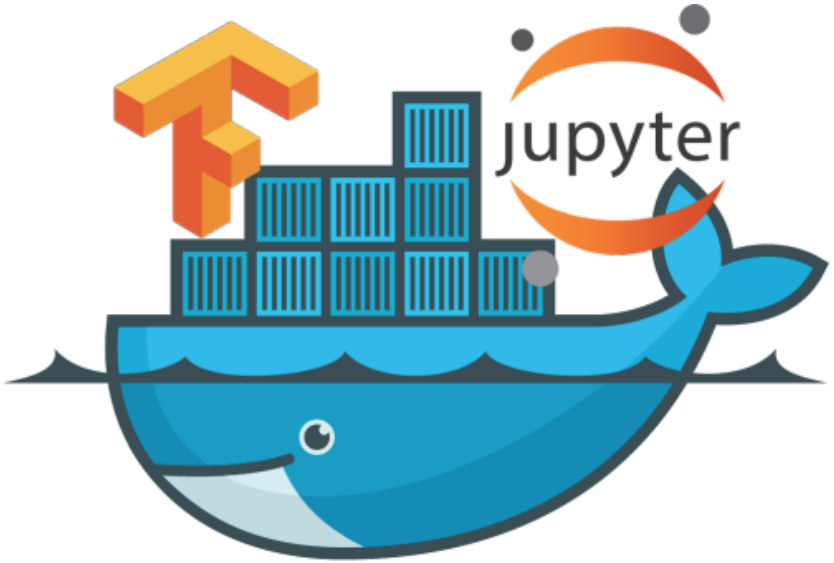
Why Docker?

- Ready-to-use ML/DL images
  - Get rid of tedious framework installation!



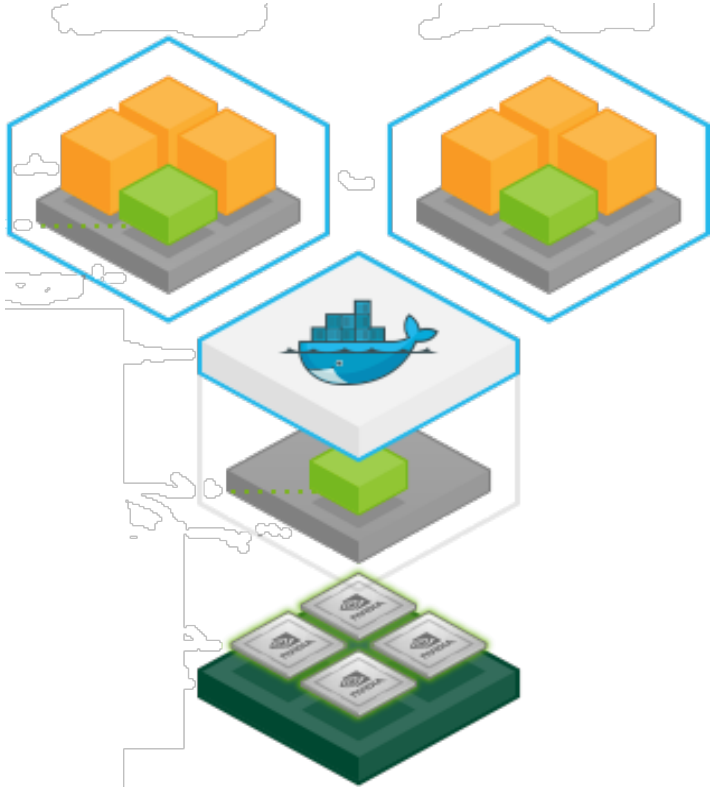
Why Docker?

- Our internal consideration
  - We want to re-use so many existing docker images/dockerfiles
  - Developers are familiar with docker

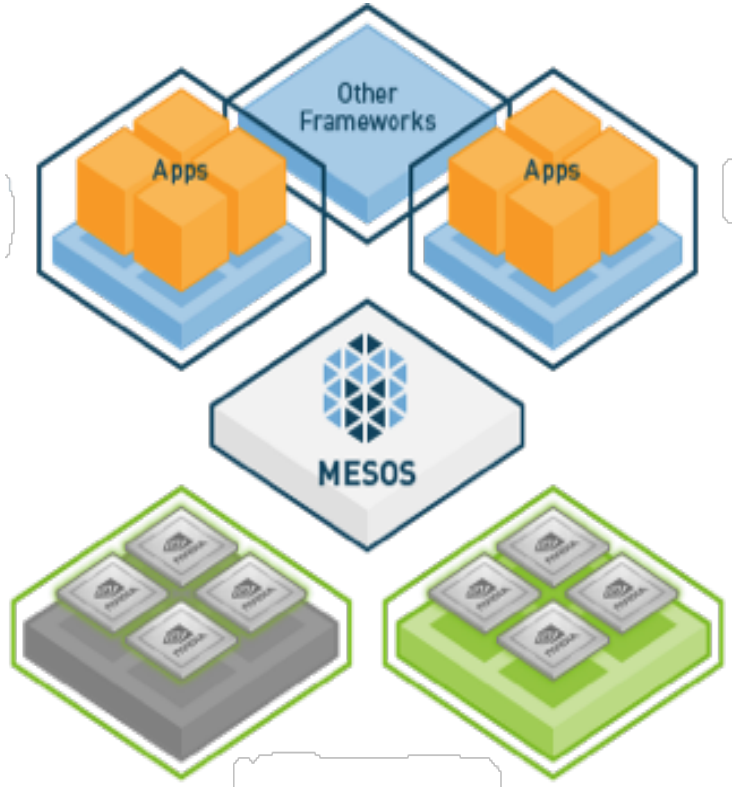


What We Want To Do?

Test locally with nvidia-docker



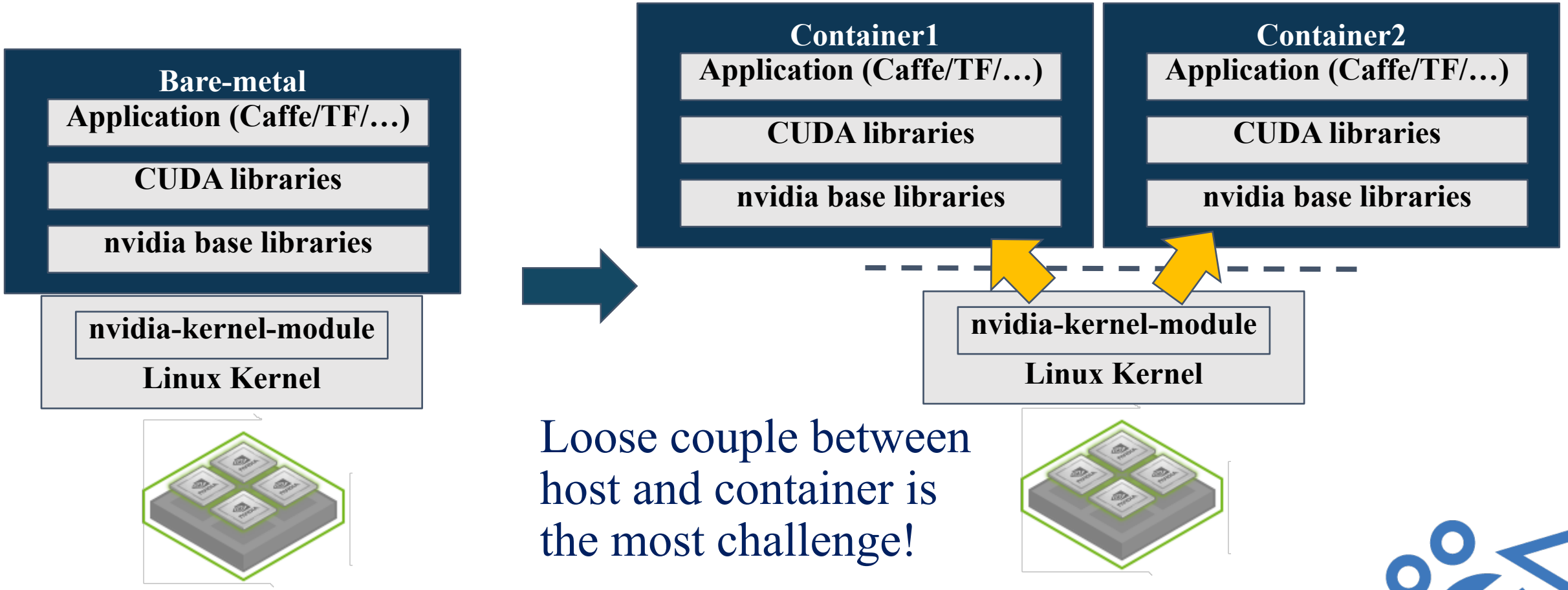
Deploy to production with Mesos



Talk Overview

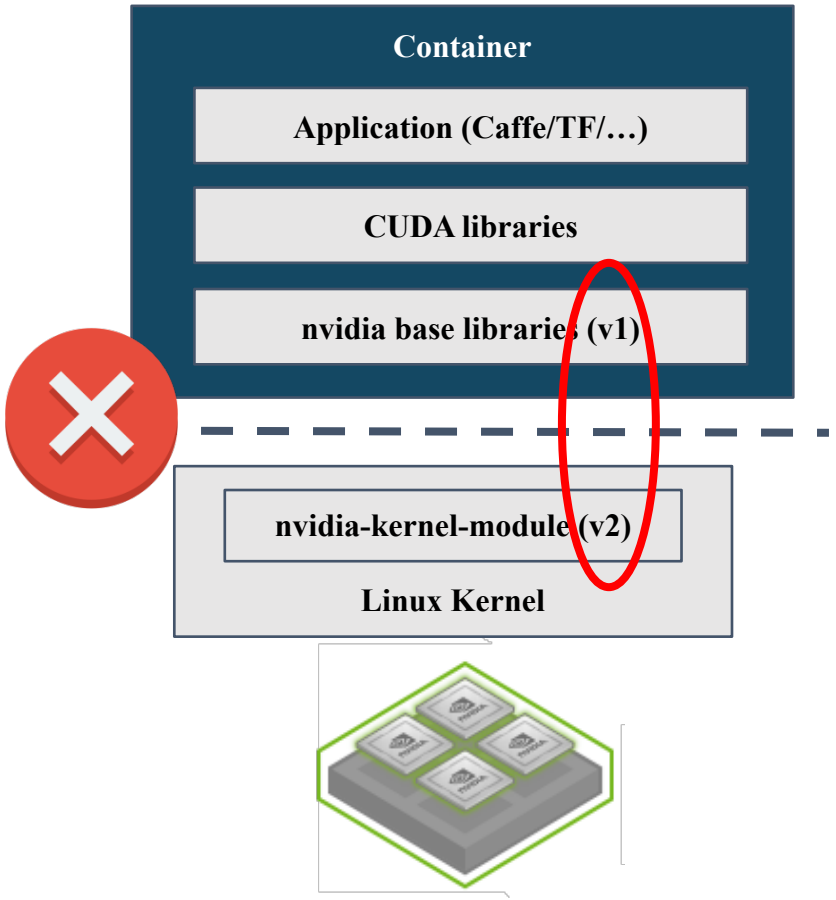
- Challenges and our basic ideas
- GPU unified scheduling design
- Future works
- Demo: running cognitive application with Mesos/Marathon + GPU

Bare-metal vs. Container for GPU

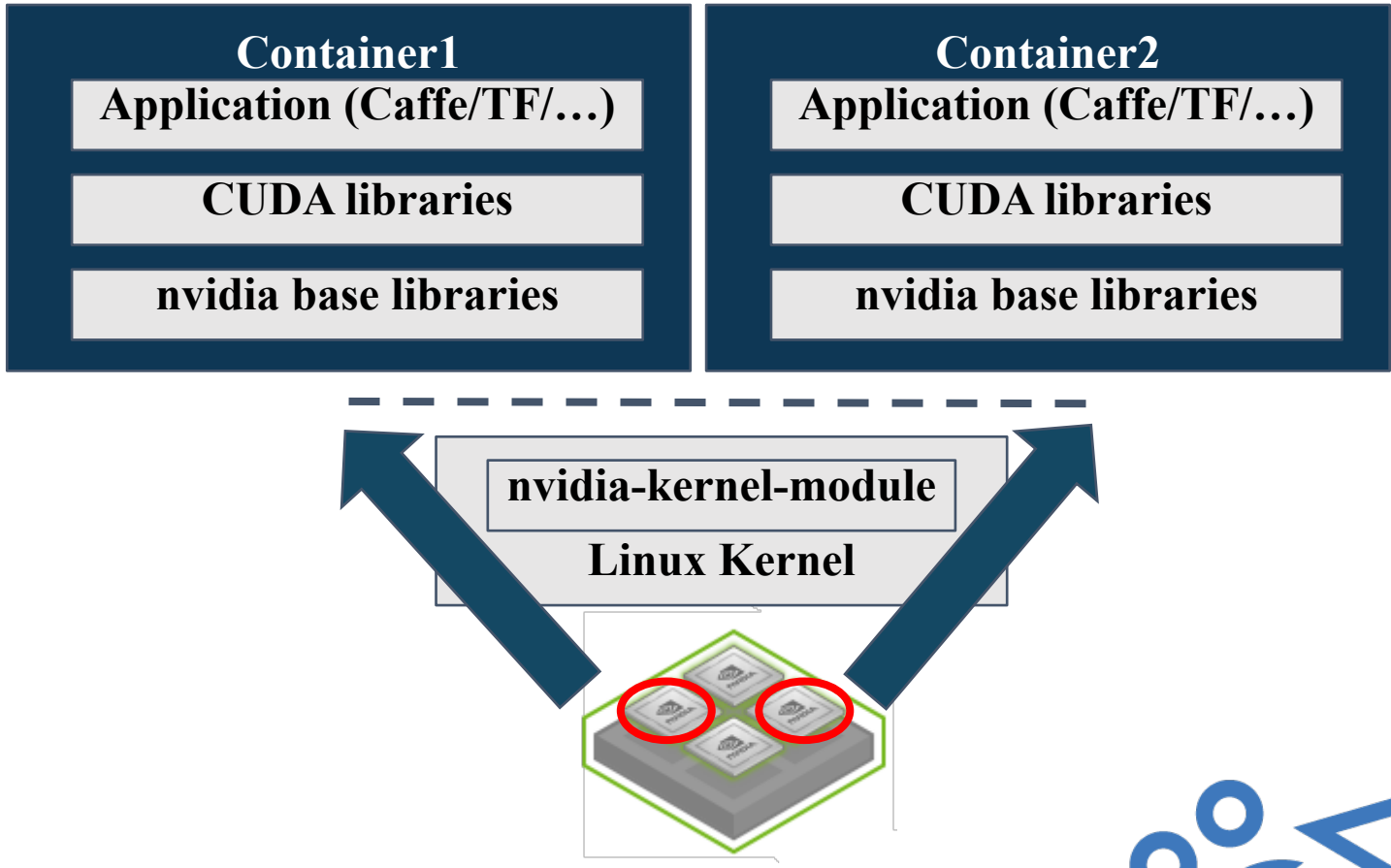


Loose couple between host and container is the most challenge!

Challenges

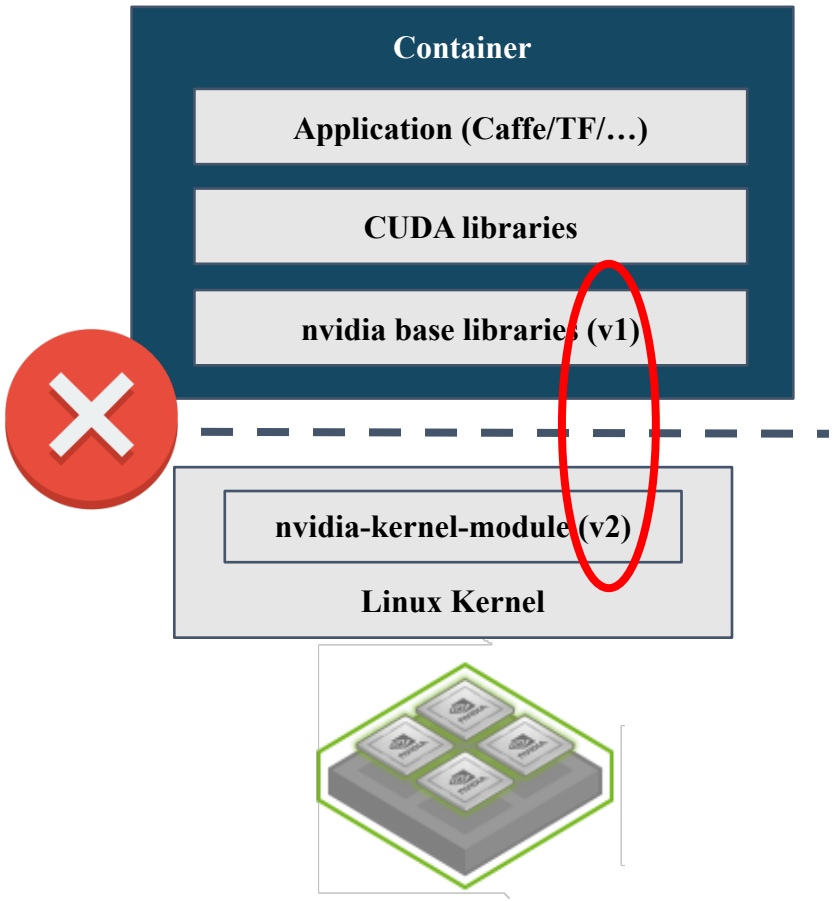


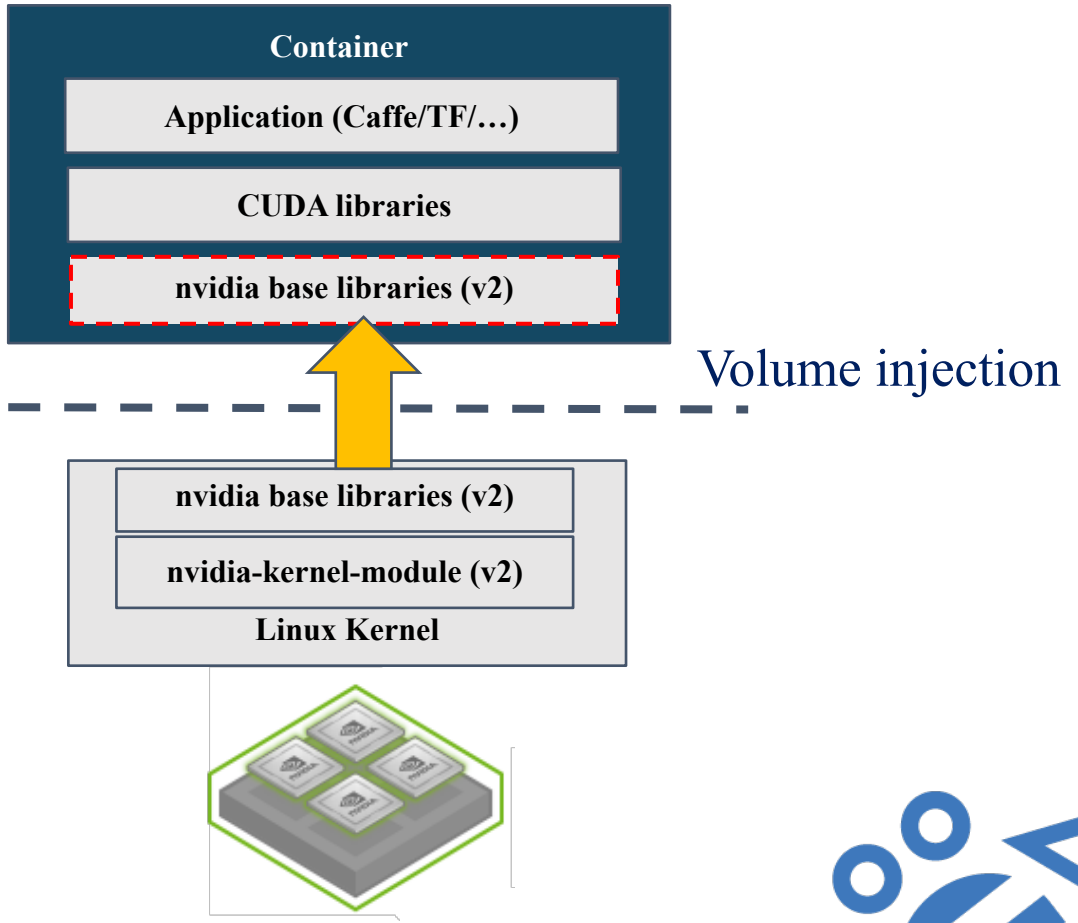
Not work if nvidia libraries and kenel module versions are not match



We also need GPU isolation control

How We Solve That?





Not work if nvidia libraries and kernel module versions are not match

How We Solve That?

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

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

  - Inject volume with "**ro**" to container if needed

How We Solve That?

- **Determine whether nvidia-volume is needed**
  - Check docker image label:
    com.nvidia.volumes.needed = nvidia_driver
  - Inject nvidia-volume to **/usr/local/nvidia** if the label found

This label certificates following things:

```
33  RUN echo "/usr/local/nvidia/lib" >> /etc/ld.so.conf.d/nvidia.conf && \
34      echo "/usr/local/nvidia/lib64" >> /etc/ld.so.conf.d/nvidia.conf
35
36  ENV PATH /usr/local/nvidia/bin:/usr/local/cuda/bin:${PATH}
37  ENV LD_LIBRARY_PATH /usr/local/nvidia/lib:/usr/local/nvidia/lib64:${LD_LIBRARY_PATH}
```

https://github.com/NVIDIA/nvidia-docker/blob/master/ubuntu-14.04/cuda/7.5/runtime/Dockerfile

How We Solve That?

GPU isolation

- Currently we support physical-core level isolation

Example	Isolation?
Per card	Yes
1 core of Tesla K80 (dual-core)	Yes
512 CUDA cores of Tesla K40	No

- GPU sharing is not supported
  - No process capping mechanism from nvidia GPU driver
  - GPU sharing is suggested for MPI/OpenMP case only

How We Solve That?

- GPU device control:

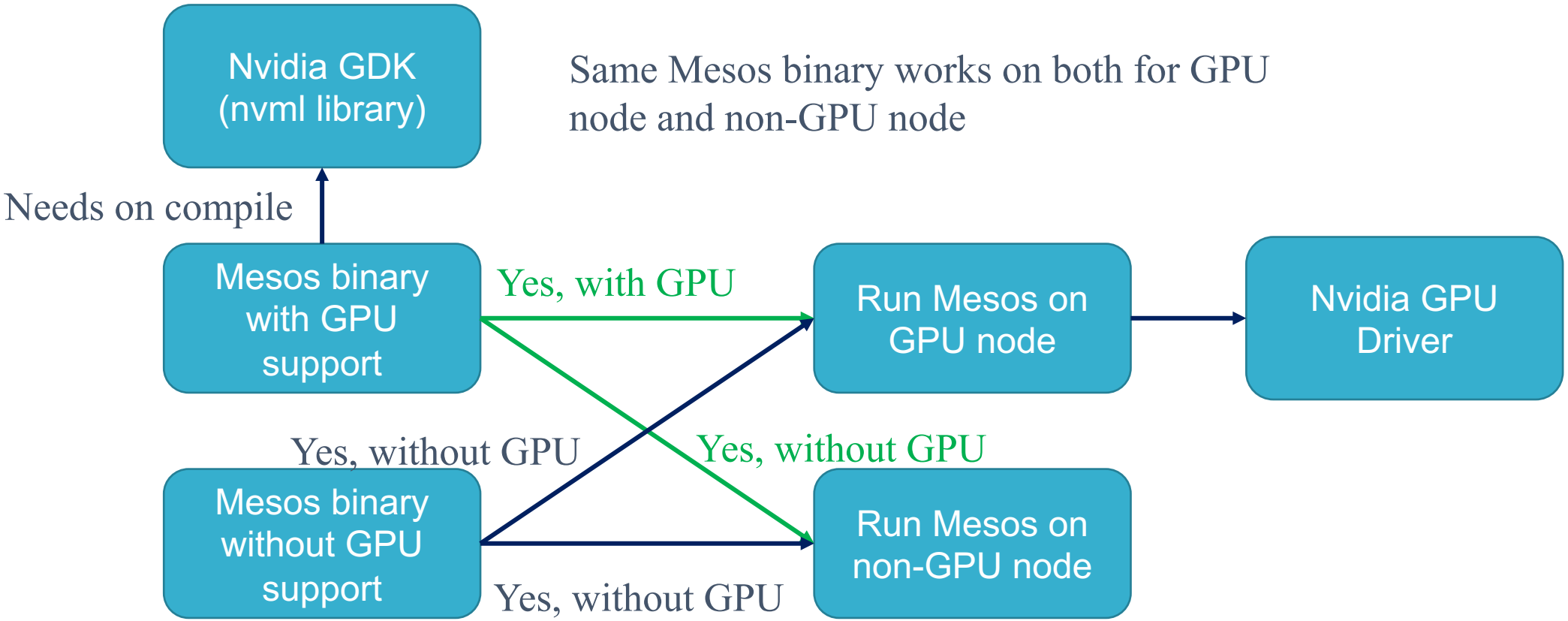
/dev

- **nvidia0** (data interface for GPU0)
- **nvidia1** (data interface for GPU1)
- **nvidiactl** (control interface)
- **nvidia-uvm** (unified virtual memory)
- **nvidia-uvm-tools** (UVM control)

- Isolation
  - Mesos containerizer: cgroups
  - Docker containerizer: “docker run –devices”

How We Solve That?

- Dynamic loading of nvml library



Apache Mesos and GPUs

- Multiple containerizer support
  - Mesos (aka unified) containerizer (fully supported)
  - Docker containerizer (code review, partially merged)

- Why support both?
  - Many people are asking for docker containerizer support to bridge the feature gap
  - People are already familiar with existing docker tools
  - Unified containerizer needs time to mature

Apache Mesos and GPUs

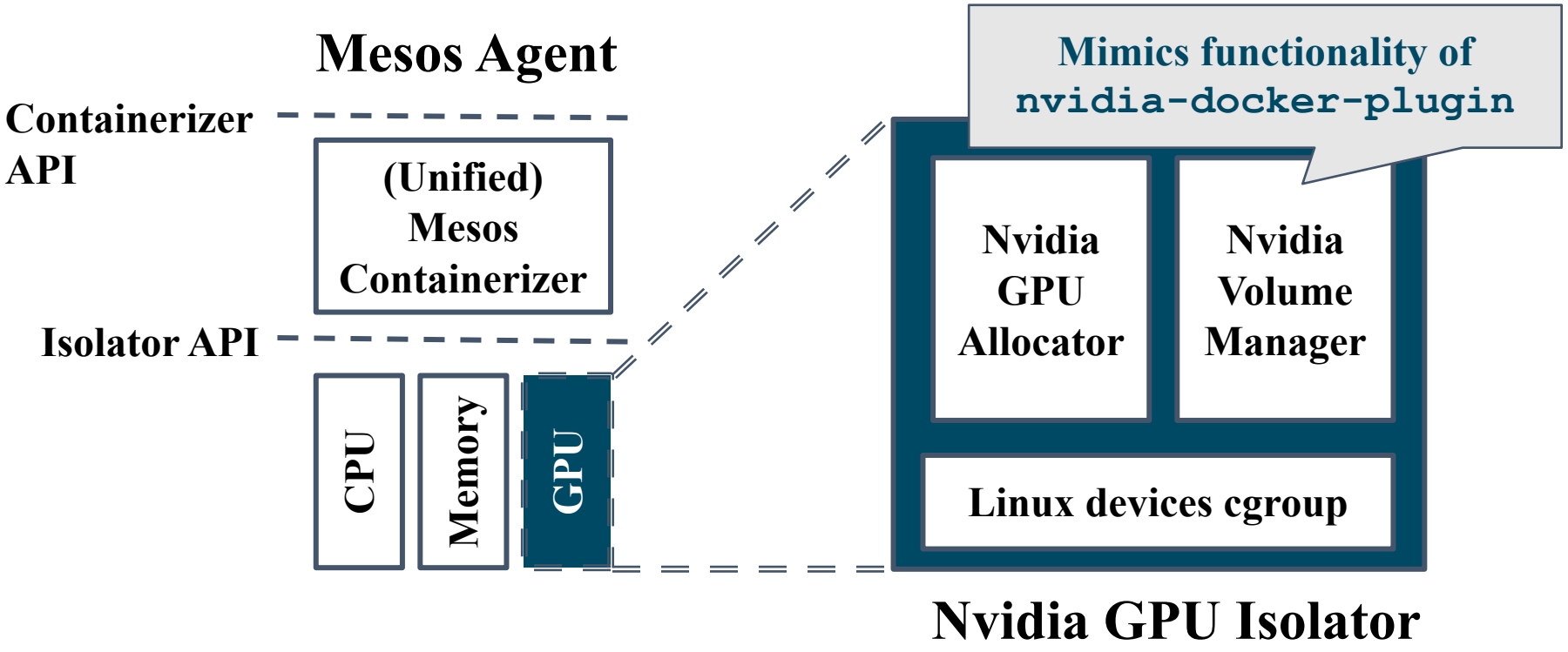
- **GPU_RESOURCES** framework capability
  - Frameworks must opt-in to receive offers with GPU resources
  - Prevents legacy frameworks from consuming non-GPU resources and starving out GPU jobs

- Use agent attributes to select specific type of GPU resources
  - Agents advertise the type of GPUs they have installed via attributes
  - Only accept an offer if the attributes match the GPU type you want

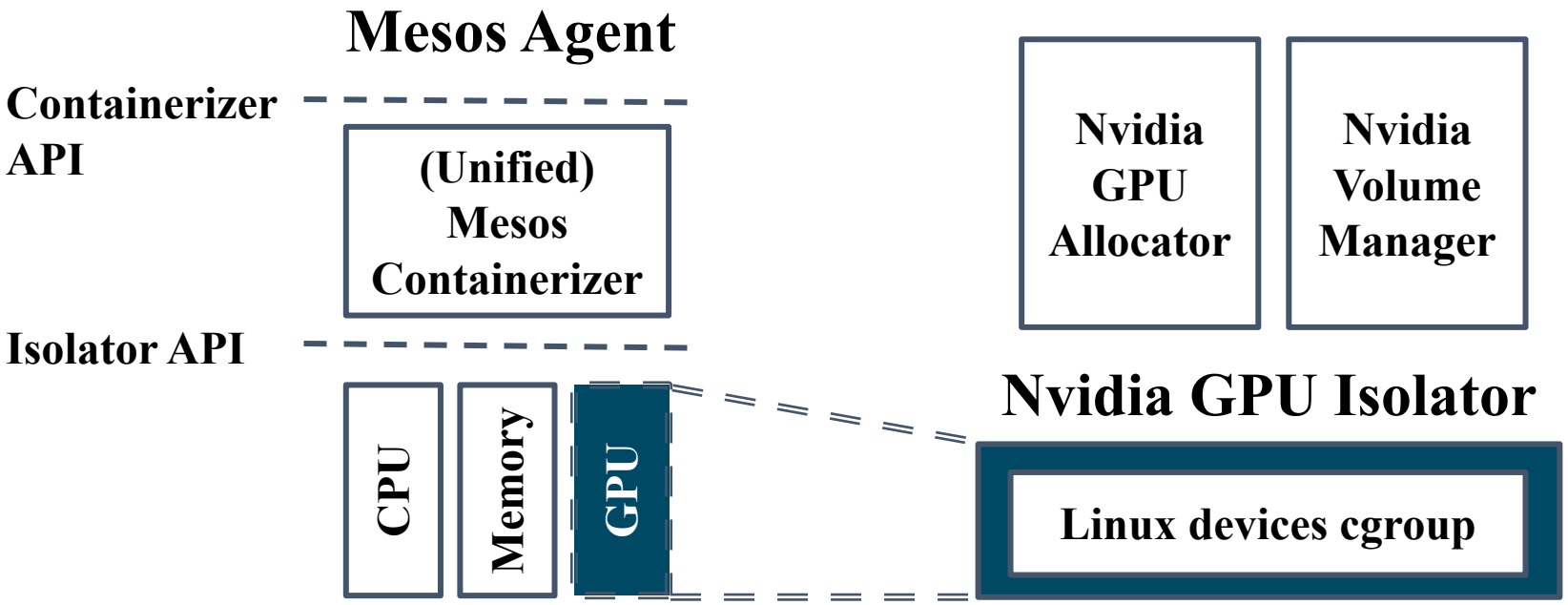
Usage

- Usage
  - Nvidia GPU and GPU driver needed
  - Install Nvidia GPU Deployment Toolkit (GDK)
  - Compile Mesos with flag: *../configure --with-nvml=/nvml-header-path && make –j install*
  - Build GPU images following nvidia-docker do: (https://github.com/NVIDIA/nvidia-docker)
  - Run a docker task with additional such resource "*gpus=1*"
  - Mesos Containerier: *--isolation="cgroups/devices,gpu/nvidia"*

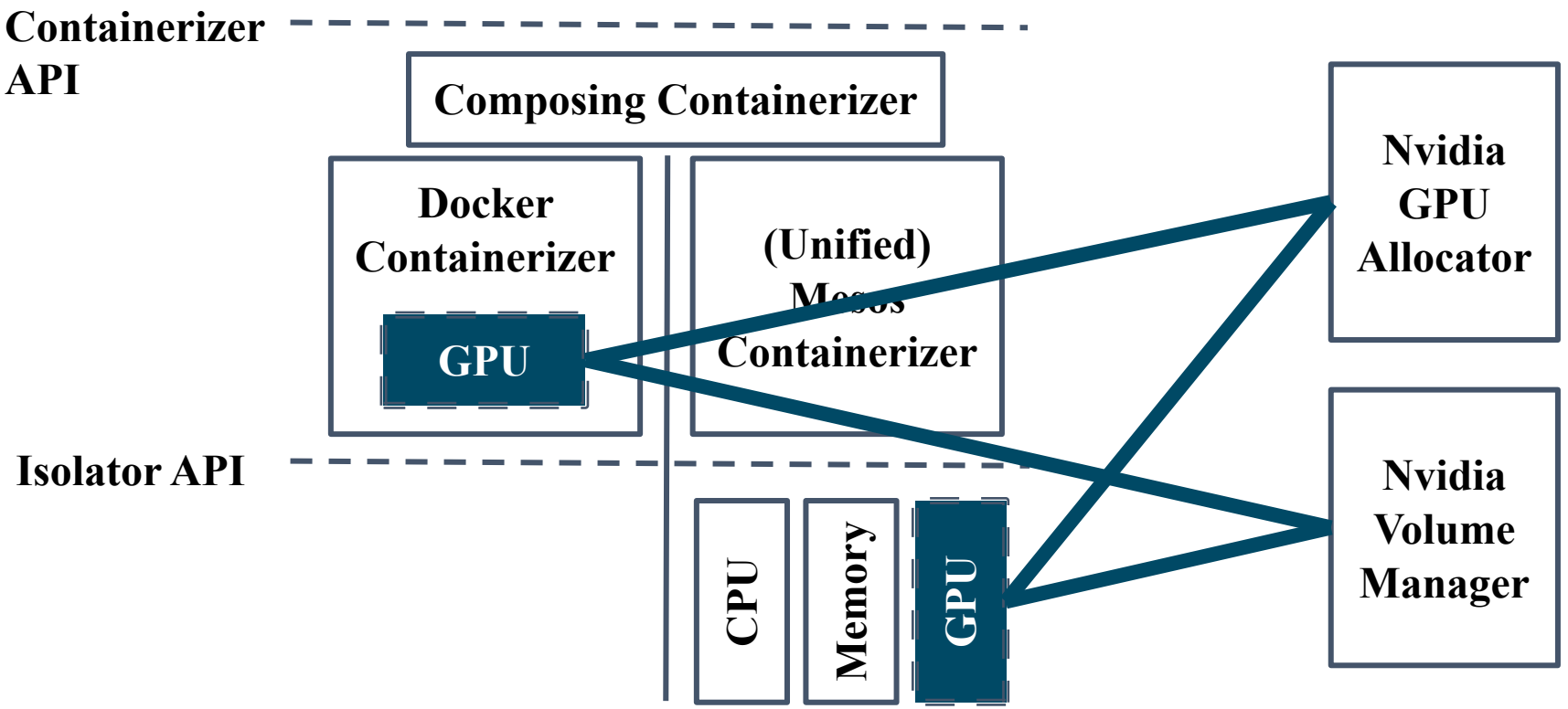
Apache Mesos and GPUs -- Evolution



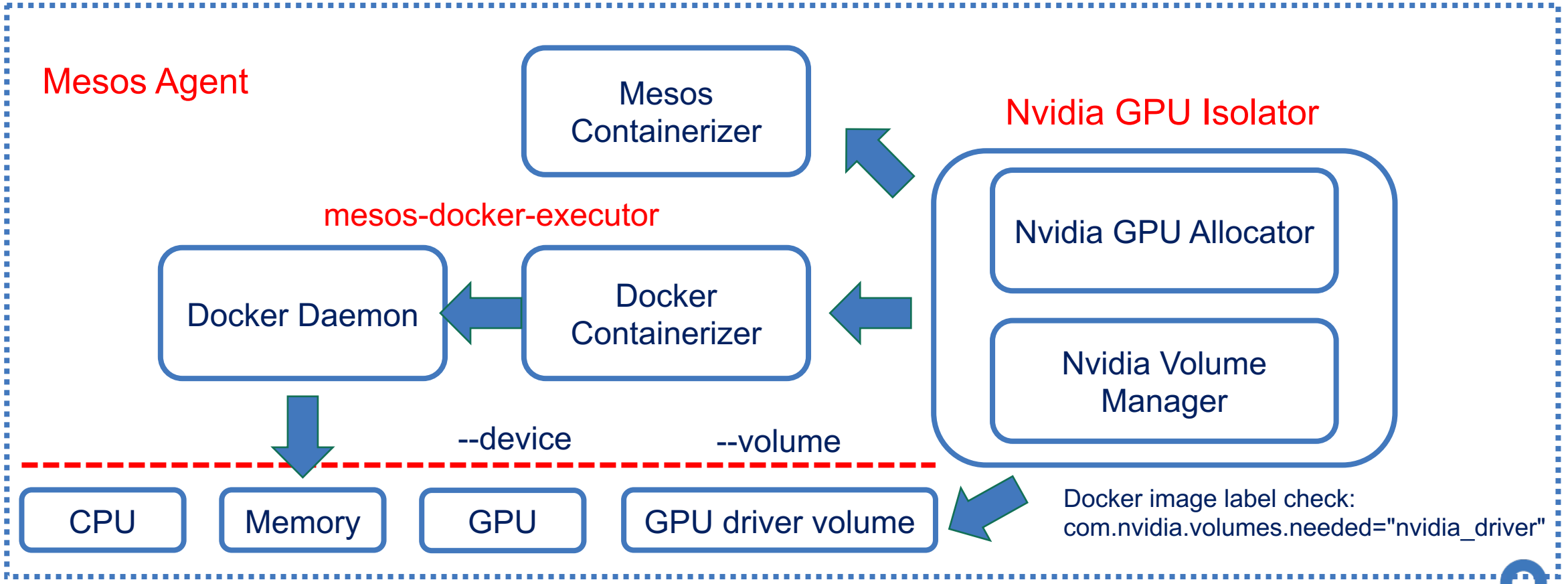
Apache Mesos and GPUs -- Evolution



Apache Mesos and GPUs -- Evolution



Apache Mesos and GPUs



Native docker arguments for GPU management

Release and Eco-syetems

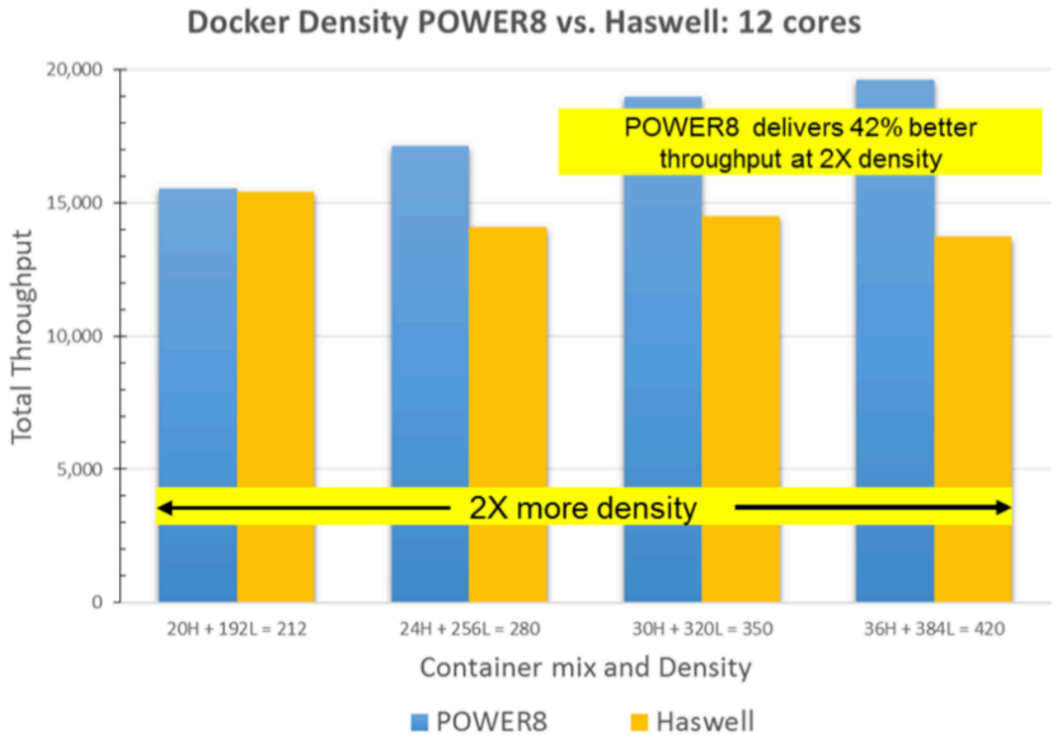
- Release
  - GPU for Mesos Containerizer: fully supported after Mesos 1.0 (supports both image-less and docker-image based containers)
  - GPU for Docker Containerizer: Expected to release on Mesos 1.1 or 1.2

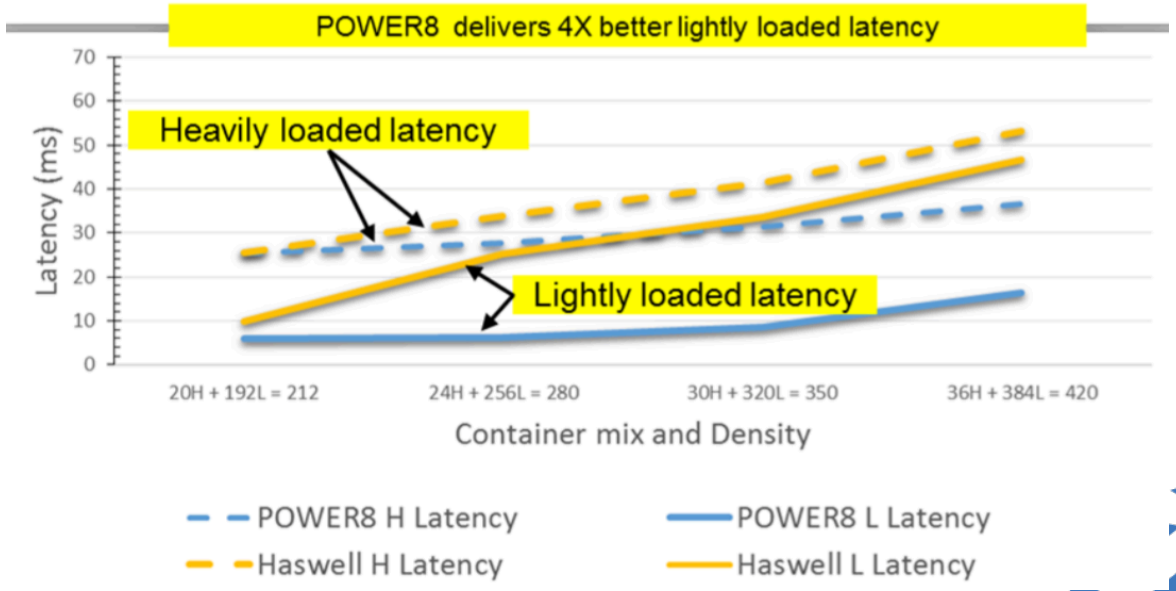
Eco-systems

- Marathon
  - GPU support for Mesos Containerizer after Marathon v1.3
  - GPU support for Docker Containerizer ready for release (wait for Mesos support)
- K8sm
  - On design

Mesos on IBM POWER8

- Apache Mesos 1.0 and GPU feature perfectly supports IBM POWER8
- IBM POWER8 Delivers Superior Cloud Performance with Docker





IBM LC产品为Power Systems增添新活力

High Performance Computing

Big Data

Compute Intensive

S812LC

- Storage rich single socket system for big data applications
- Memory Intensive workloads

S822LC

大器有为

要多大
才比的上它的认知高度

帮助企业加速获得洞察

- 以存储为中心、高数据吞吐量工作负载的理想之选
- 采用 2 个 POWER8 插槽，用于处理大数据工作负载
- 通过 CAPI 和 GPU 实现大数据加速

S822LC for HPC

大快人心

要多快
才比的上它的运算速度

开启新一波加速浪潮

- 引入了 CPU-GPU NVLink，可将进入 GPU 加速器的带宽提升 **2.5 倍**
- POWER8 与 NVIDIA NVLink 的完美结合

S821LC

大智若云

要多智
才比的上它的云端契合度

开启数据中心与云的无缝模式

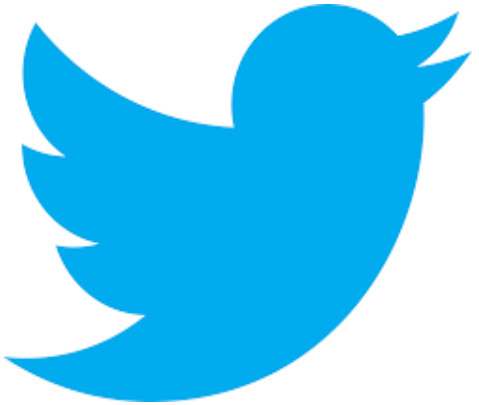
- Intel x86 系统内存带宽的 2 倍
- 内存密集型工作负载

S822LC for Commercial Computing

- 2X memory bandwidth of Intel x86 systems
- Memory Intensive workloads

Special Thanks to Collaborators

- Kevin Klues
- Rajat Phull
- Seetharami Seelam
- Guangya Liu
- Qian Zhang
- Benjamin Mahler
- Vikrama Ditya
- Yong Feng



MESOSPHERE

Demo

- Build a GPU-enabled cognitive web service in a minute!