KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

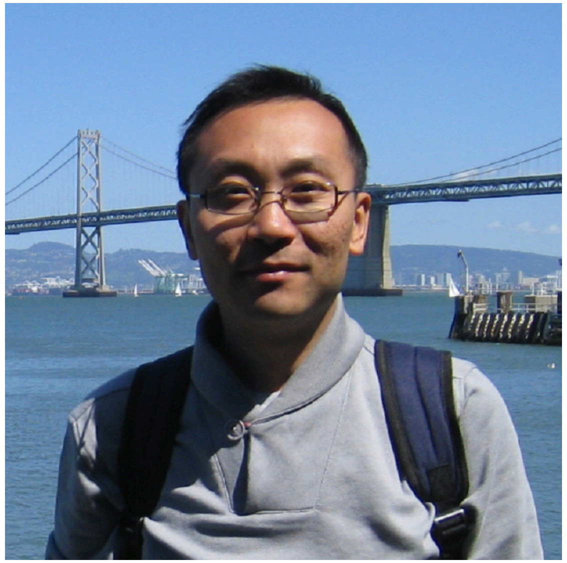
China 2019

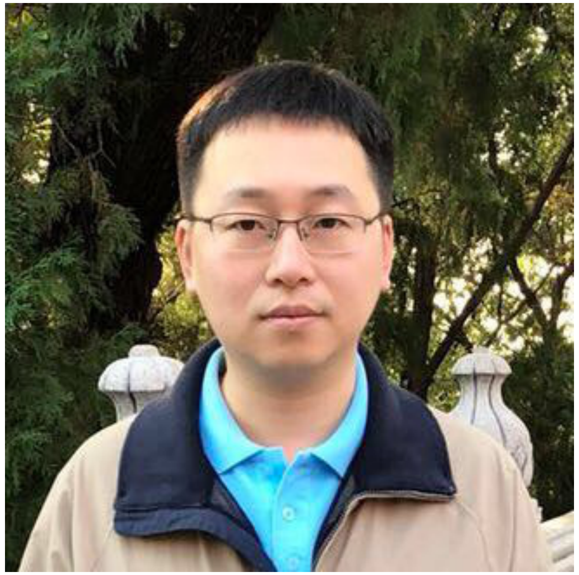
KubeCon CloudNativeCon
OPEN SOURCE SUMMIT
China 2019

Minimizing GPU Cost For Your Deep Learning Workload On Kubernetes

Yang Che, Alibaba Cloud
Kai Zhang, Alibaba Cloud

Who are we?



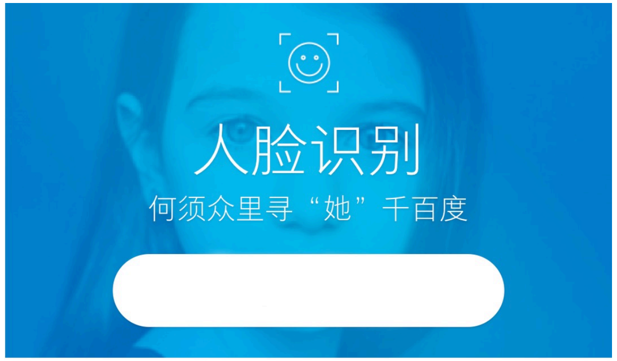


Kai Zhang
Staff engineer of Alibaba Cloud

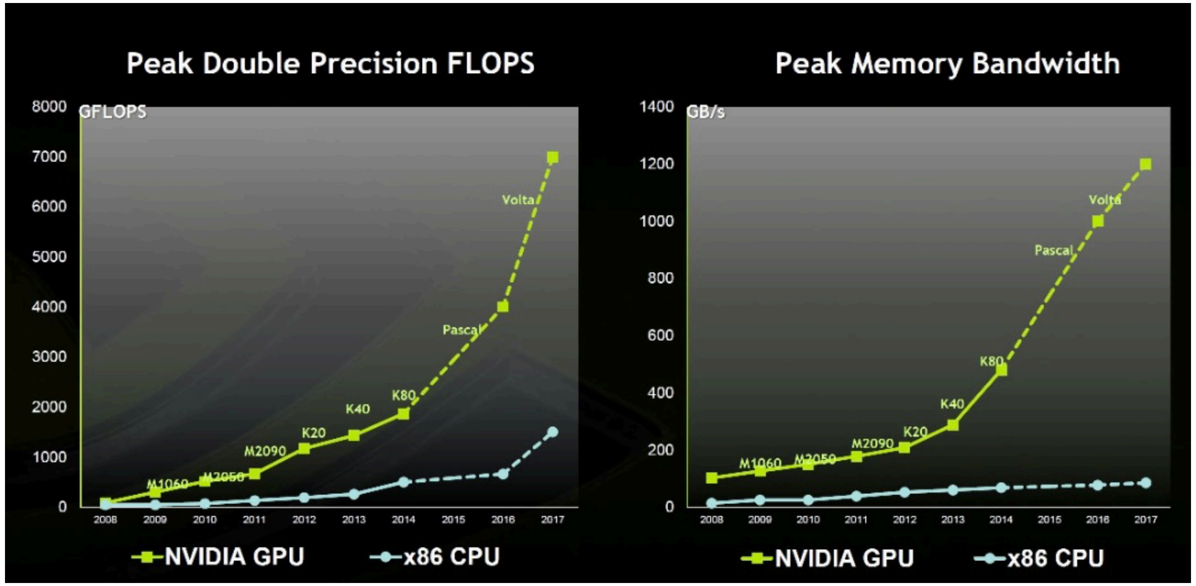
Yang Che
Senior engineer of Alibaba Cloud

Container service, Kubernetes, Deep learning platform

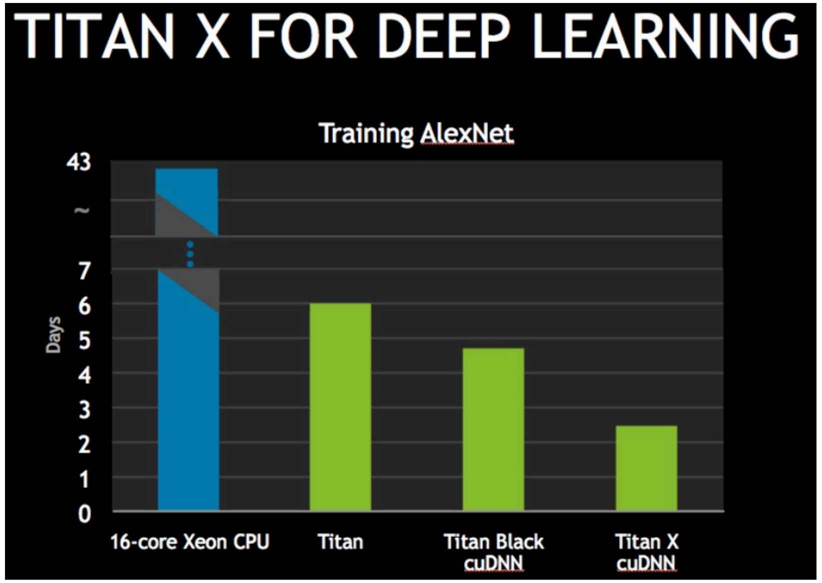
AI is everywhere



GPU speeds up AI



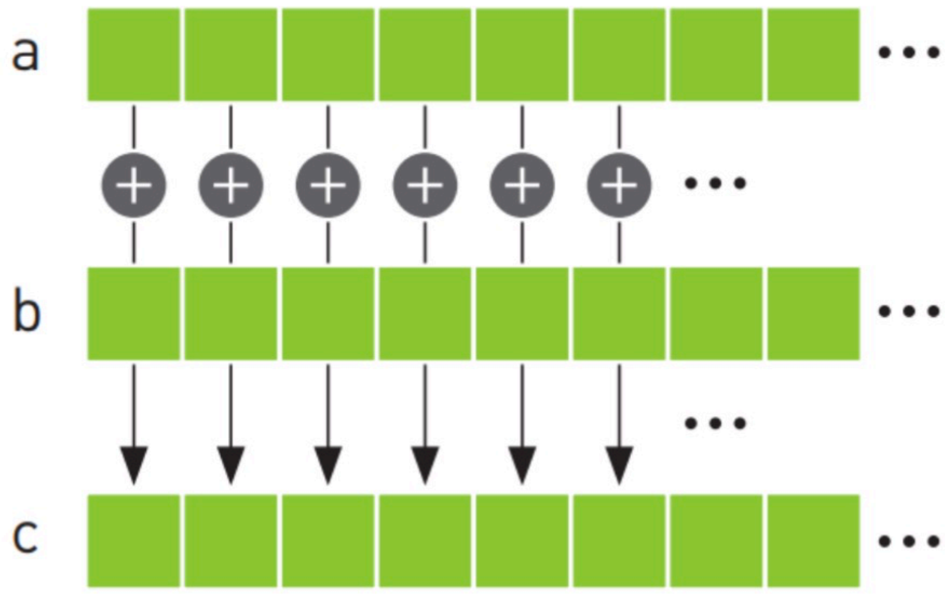
https://wccftech.com/nvidia-pascal-volta-gpus-sc15/



https://blogs.nvidia.com/blog/2015/03/17/digits-devbox/

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

Why GPU is so fast?



```
void vectorAddCPU(float *A, float *B, float *C) {
        for(int i=0;i < N; i++)
        {
        c[i] = A[i] + B[i];
        }
 }
```

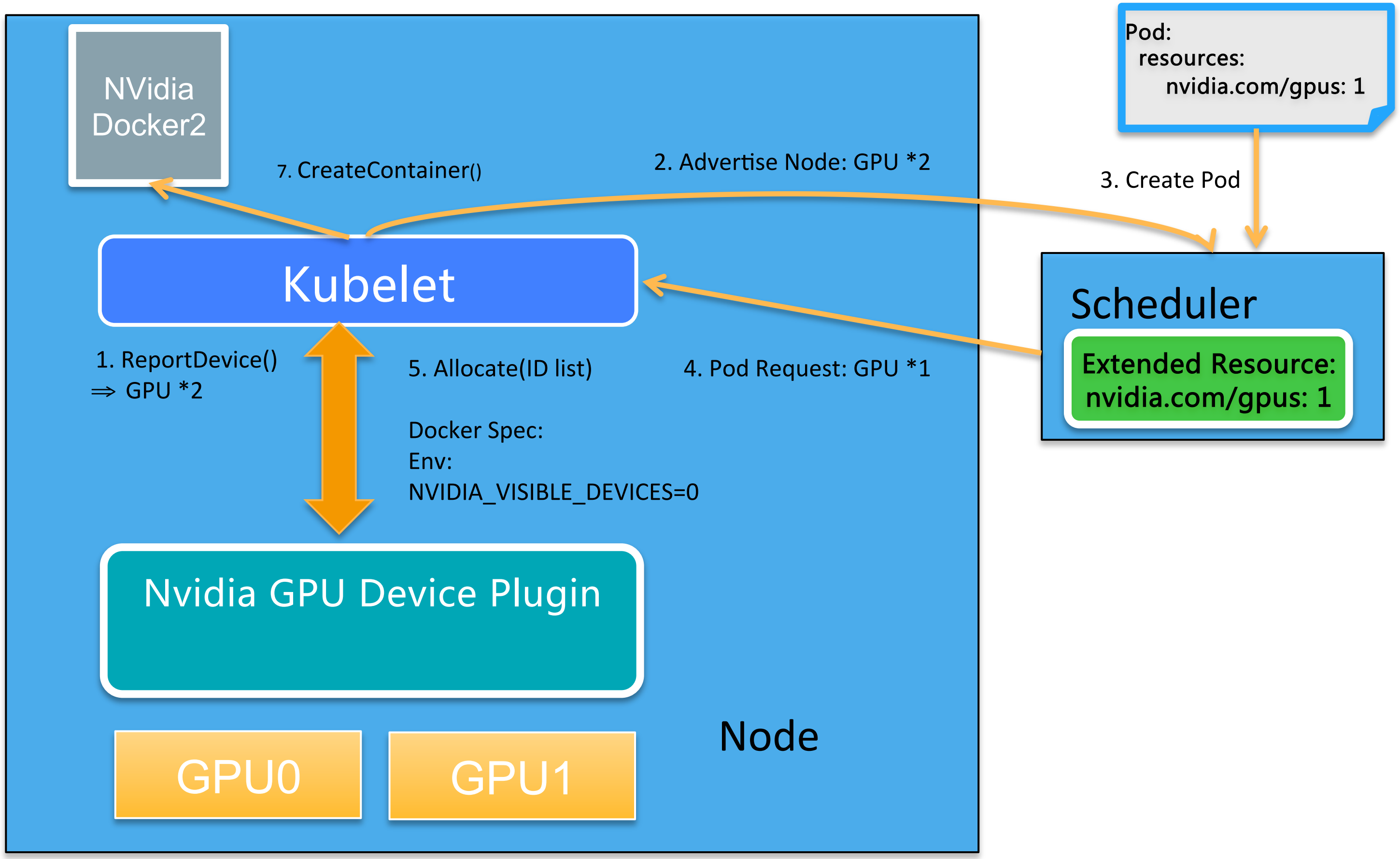
CPU Compute

```
void vectorAddGPU(float *A, float *B, float *C, int N) {
            if (tid < N)
              C[tid] = A[tid] + B[tid];
}
```

GPU Compute

Scheduling GPUs on Kubernetes

- Extended Resource
  - GPU, FPGA, RDMA
- Device Plugin framework
  - The vendor advertise their resources to the Kubernetes

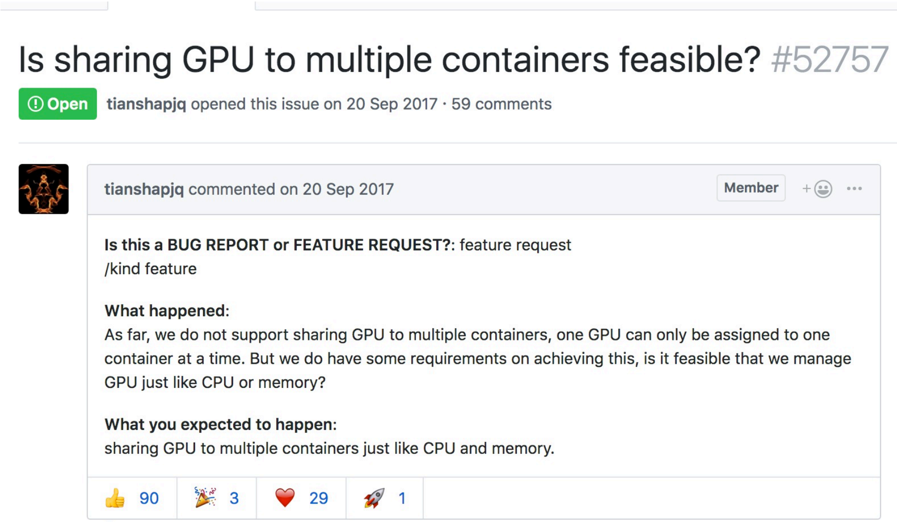


Why do we need to share GPU In Kubernetes?

- Increase GPU utilization in the cluster level
- Reuse existing resource to improve Business Efficiency
- Fine-grained GPU assignment to improve flexibility

The Challenges of Sharing GPU in Kubernetes

- Scheduling
  - Kubernetes only supports exclusive GPU assignment
- Isolation
  - NVIDIA GRID is for the Hypervisor, not for Kubernetes whose runc is default container runtime
  - MPS is only for Volta and is not ready for the production



Is sharing GPU to multiple containers feasible? #52757

Open tianshapjq opened this issue on 20 Sep 2017 · 59 comments

tianshapjq commented on 20 Sep 2017 Member

Is this a BUG REPORT or FEATURE REQUEST?: feature request
/kind feature

What happened:
As far, we do not support sharing GPU to multiple containers, one GPU can only be assigned to one container at a time. But we do have some requirements on achieving this, is it feasible that we manage GPU just like CPU or memory?

What you expected to happen:
sharing GPU to multiple containers just like CPU and memory.

👍 90 🎉 3 ❤️ 29 🚀 1

Design Thinking

Goal:

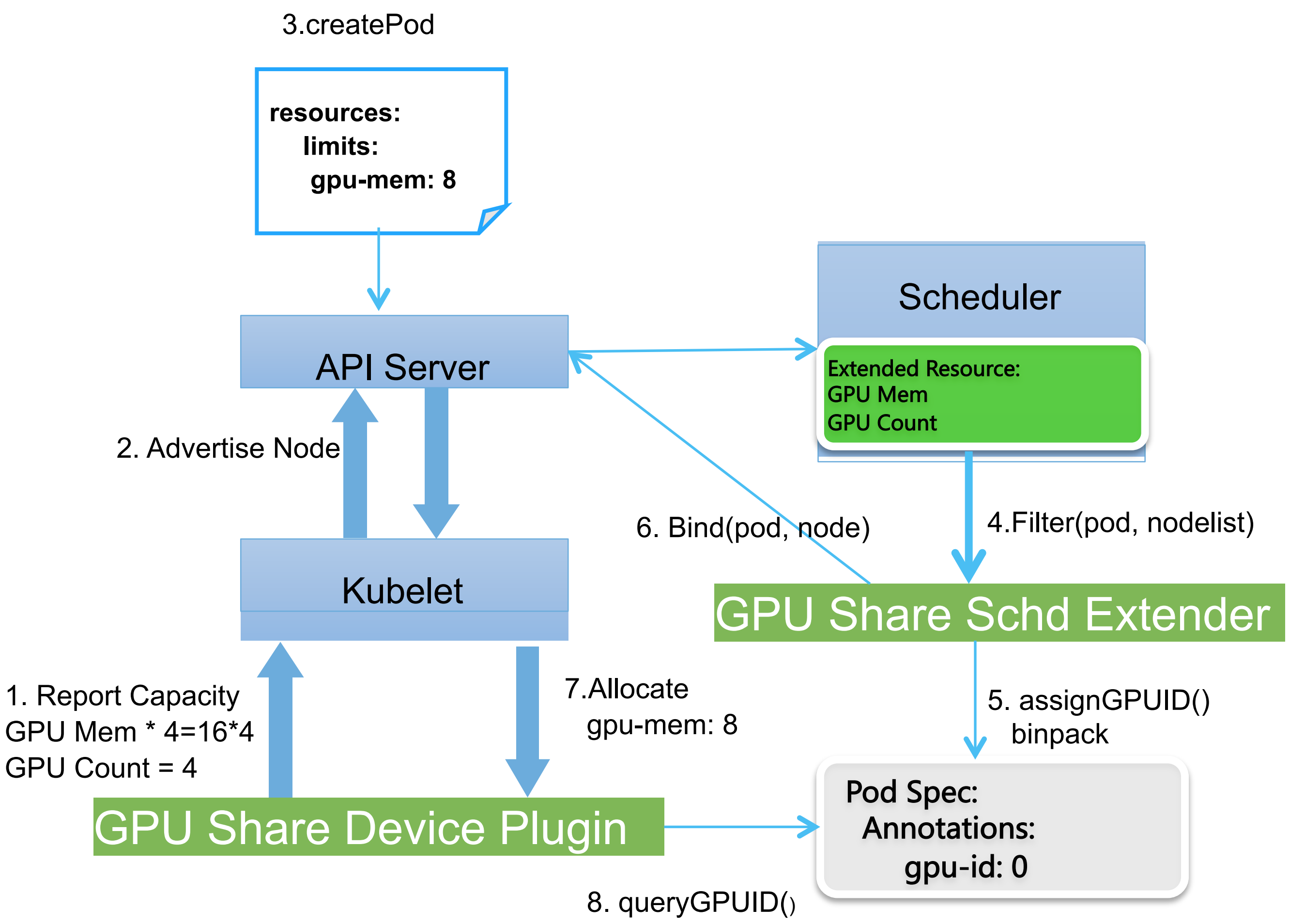
- Users can request for sharing GPU resource easiliy
- Only for scheduling
- Don’t change any Kubernetes core code

Non Goal:

- GPU resource Isolation

Architecture Overview

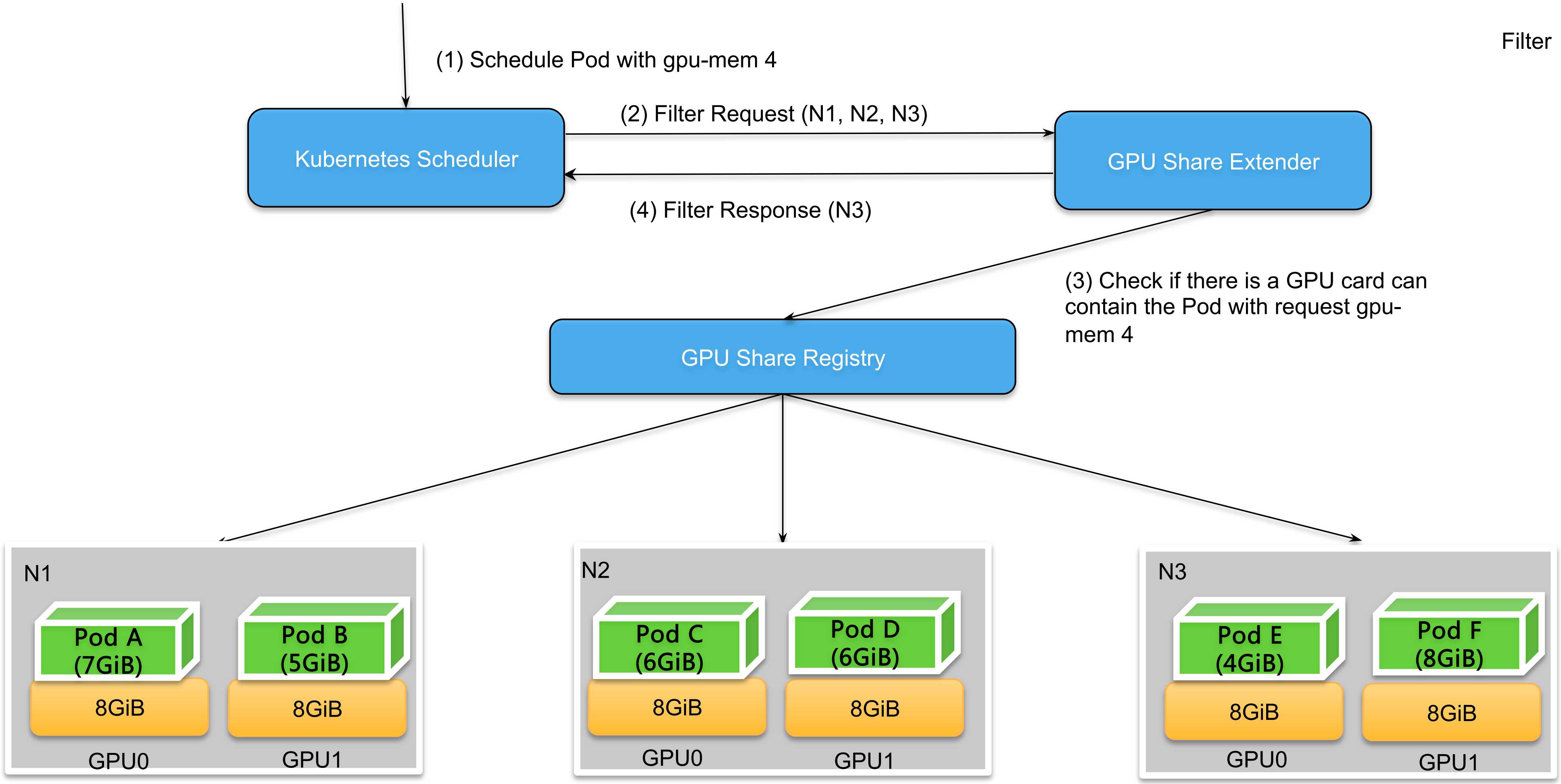
- Make the gpu-mem as extended resource
- The necessity of global scheduling
- Leverage scheduling extender mechanism



https://github.com/AliyunContainerService/gpushare-scheduler-extender

Architecture Overview(Cont.)

Filter



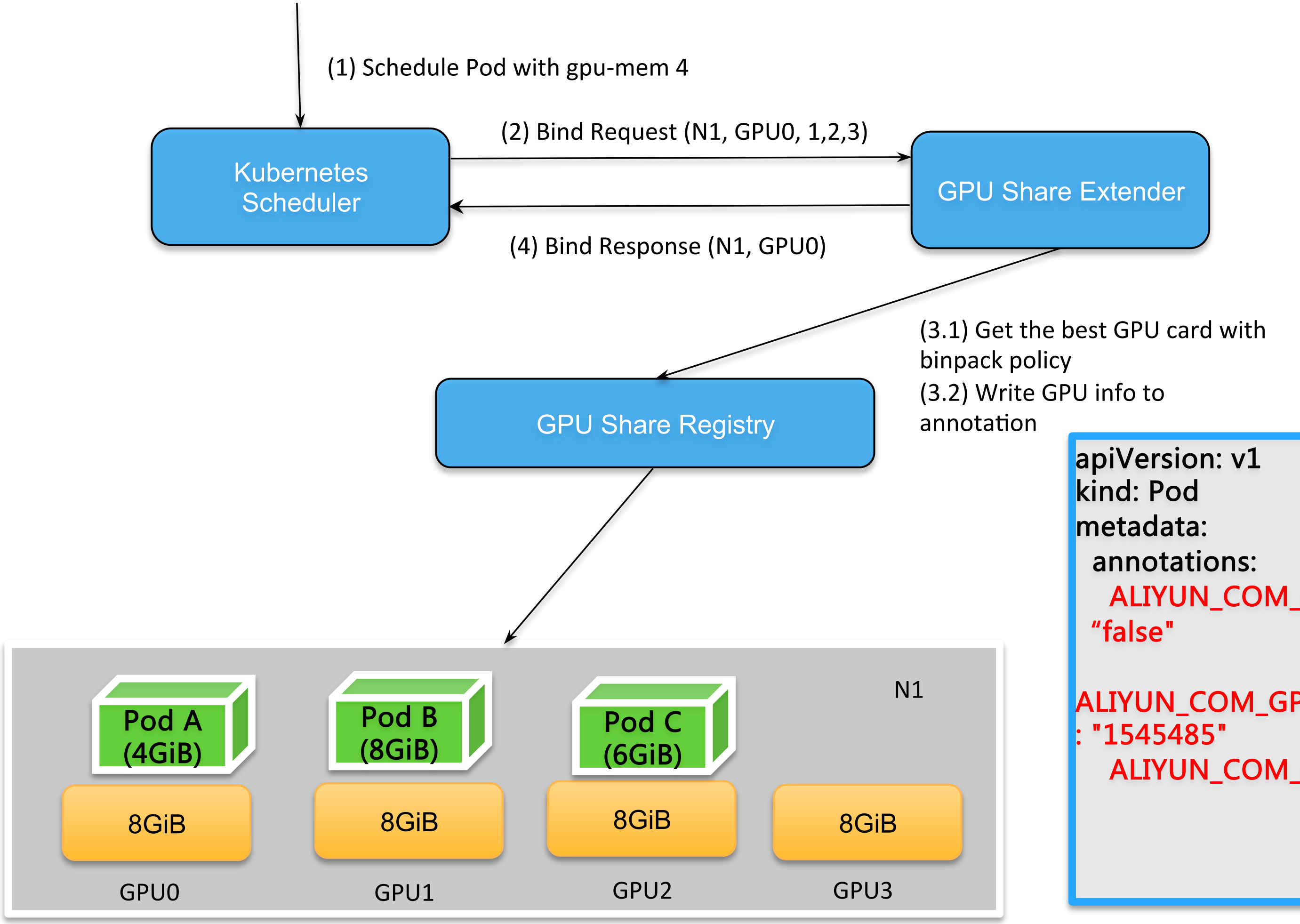
(1) Schedule Pod with gpu-mem 4

(2) Filter Request (N1, N2, N3)

(3) Check if there is a GPU card can contain the Pod with request gpu-mem 4

(4) Filter Response (N3)

Architecture Overview(Cont.)

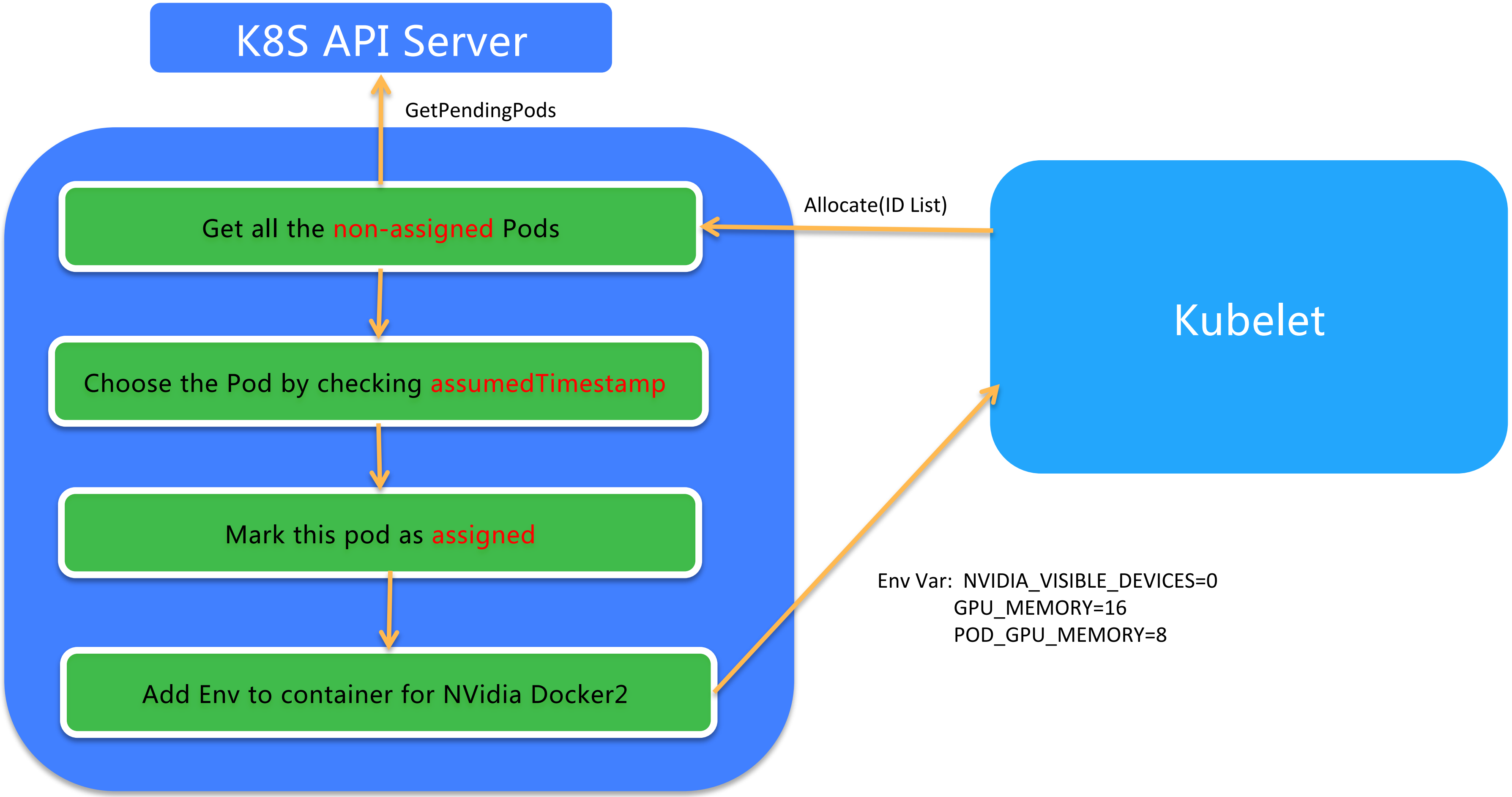

(1) Schedule Pod with gpu-mem 4
Kubernetes Scheduler
(2) Bind Request (N1, GPU0, 1,2,3)
GPU Share Extender
(4) Bind Response (N1, GPU0)
(3.1) Get the best GPU card with binpack policy
(3.2) Write GPU info to annotation
GPU Share Registry
N1
Pod A (4GiB)
Pod B (8GiB)
Pod C (6GiB)
8GiB
8GiB
8GiB
8GiB
GPU0
GPU1
GPU2
GPU3

Bind

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    ALIYUN_COM_GPU_MEM_ASSIGNED:
  “false"

ALIYUN_COM_GPU_MEM_ASSUME_TIME
: "1545485"
    ALIYUN_COM_GPU_MEM_IDX: "0"
```

Architecture Overview(Cont.)


K8S API Server
GetPendingPods
Get all the non-assigned Pods
Choose the Pod by checking assumedTimestamp
Mark this pod as assigned
Add Env to container for NVidia Docker2
Allocate(ID List)
Kubelet

Env Var: NVIDIA_VISIBLE_DEVICES=0
GPU_MEMORY=16
POD_GPU_MEMORY=8

Deploy GPU Sharing Capabilities in Kubernetes

1. Install with Helm

```
# git clone https://github.com/AliyunContainerService/gpushare-scheduler-extender.git
# cd gpushare-scheduler-extender/deployer/chart
# helm install --name gpushare --namespace kube-system --set kubeVersion=1.12.6 --set masterCount=3
gpushare-installer
```

2. Add node labels for GPU sharing

```
# kubectl label node <target_node> gpushare=true
```

3. Download and install the kubectl extension

```
# cd /usr/bin/
# wget https://github.com/AliyunContainerService/gpushare-device-plugin/releases/download/v0.3.0/
kubectl-inspect-gpushare
# chmod u+x /usr/bin/kubectl-inspect-gpushare
```

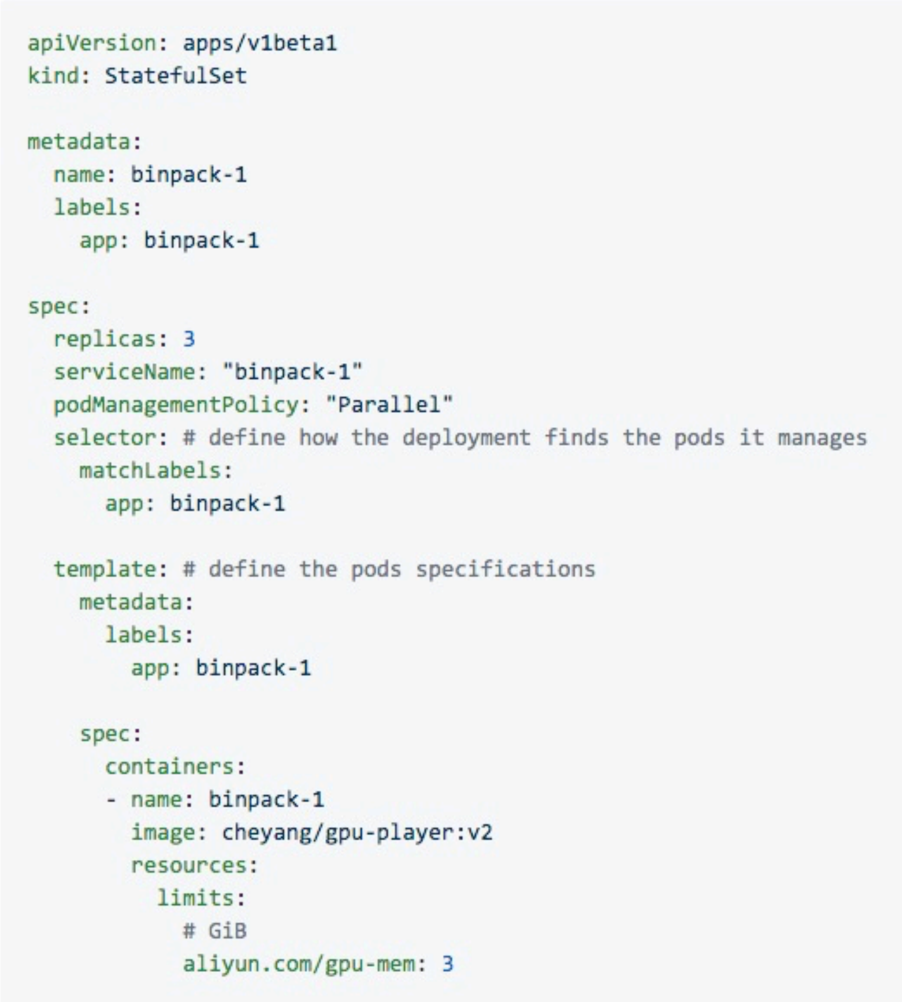
Use GPU Sharing in Kubernetes

1. Query the allocation status of the shared GPU

```
# kubectl inspect gpushare
NAME                               IPADDRESS     GPU0(Allocated/Total)  GPU Memory(GiB)
cn-shanghai.i-uf61h64dz1tmlob9hmtb  192.168.0.71  0/15                   0/15
cn-shanghai.i-uf61h64dz1tmlob9hmtc  192.168.0.70  0/15                   0/15
------------------------------------------------------------------------------
Allocated/Total GPU Memory In Cluster:
0/30 (0%)
```

2. Add node labels for GPU sharing

```
# kubectl apply -f binpack.yaml
```



```yaml
apiVersion: apps/v1beta1
kind: StatefulSet

metadata:
  name: binpack-1
  labels:
    app: binpack-1

spec:
  replicas: 3
  serviceName: "binpack-1"
  podManagementPolicy: "Parallel"
  selector: # define how the deployment finds the pods it manages
    matchLabels:
      app: binpack-1

  template: # define the pods specifications
    metadata:
      labels:
        app: binpack-1

    spec:
      containers:
      - name: binpack-1
        image: cheyang/gpu-player:v2
        resources:
          limits:
            # GiB
            aliyun.com/gpu-mem: 3
```

Use GPU Sharing in Kubernetes(Cont.)

3. Check the info from environment variables

```
# The total amount of GPU memory on the current device (GiB)
ALIYUN_COM_GPU_MEM_DEV=15

# The GPU Memory of the container (GiB)
ALIYUN_COM_GPU_MEM_CONTAINER=3
```

4. Limit GPU memory by setting fraction through TensorFlow API

```
fraction = round( 3 / 15 , 1 )
config = tf.ConfigProto()
config.gpu_options.per_process_gpu_memory_fract
ion = fraction
sess = tf.Session(config=config)
# Runs the op.
while True:
     sess.run(c)
```

Demo

Summary & Next Steps

- Some typical ML workloads requires GPU sharing to reduce cost
- Need a solution to support GPU sharing without changing Kubernetes core code
- Discuss the design and implementation of GPU sharing in Kubernetes
- Next Steps
  - Integrate Nvidia MPS as the option for isolation(Experiment)
  - Generic Solution for GPU, RDMA and other devices

免登陆听课　　动手实践课后自测

CKA课程内容同步　　阿里云原生最佳实践

立即听课

KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2019