KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon | CloudNativeCon

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

AI_dev Open Source GenAI & ML Summit

China 2024

Constructing 10^{x} Efficiency of Cloud-Native AI Infrastructure

Qiuping Dai

Peter Pan

KubeCon | CloudNativeCon

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

AI_dev Open Source GenAI & ML Summit

China 2024

Constructing 10^{x} Efficiency of Cloud-Native AI Infrastructure

Qiuping Dai

Peter Pan

About us

Qiuping Dai

Product Manager of Cloud Native

was major designer **of d.run**

skilled at network, storage, GPU area

Peter Pan

Cloud Native **Dev Lead**

employed by **DaoCloud**

open source advocate

Background

Opportunity

Government

MIIT + State Council 《 Action Plan for the High-quality Development of Computing Power Infrastructure 》

By 2025, up to **300EFLops**, AI computing reaches **35%** ,

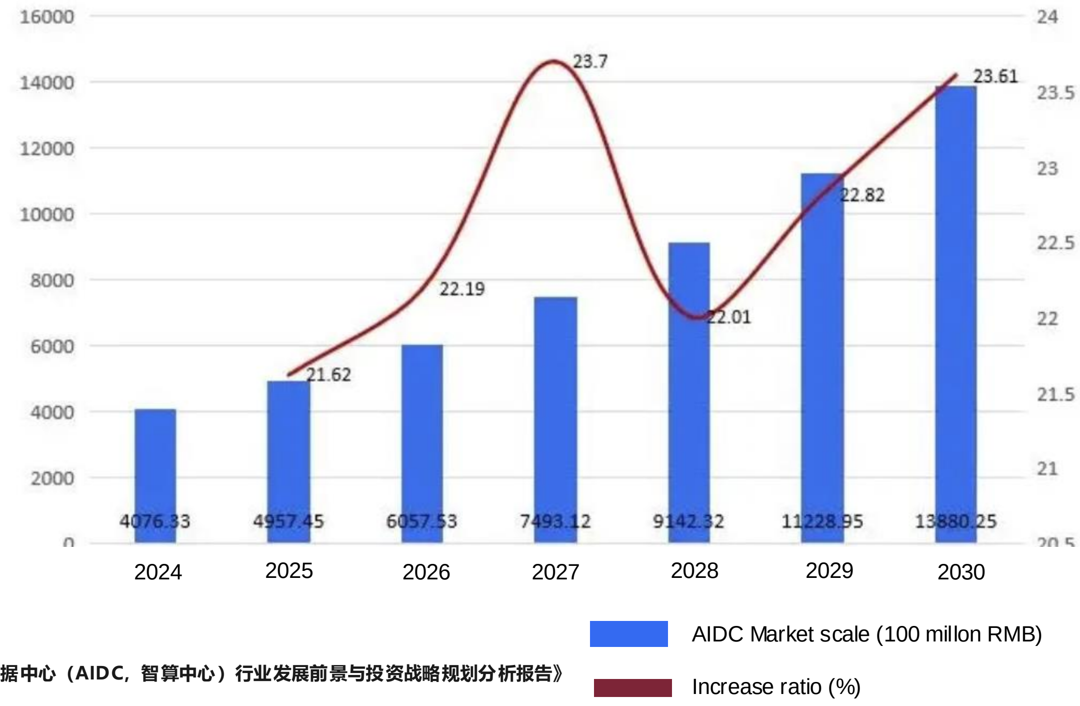
工业和信息化部、国务院国资委等6**部**门联合印发《**算力基**础设施高质量发展行动计划》，
提出到建设规划： 2025**年算力**规模超过300EFLOPS， **智能算力占比达到**35%，

Industry

Analytical report 《AIDC industrial trend in China》
AIDC > 50% growth rate

Challenge

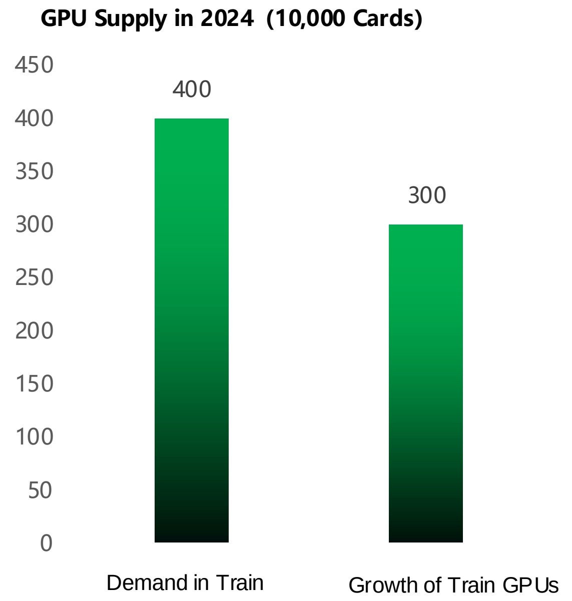
- Low Utilization and efficiency
- GPU Supply-Demand Imbalance
- Technical Obstacle
- Immature Operating Experience

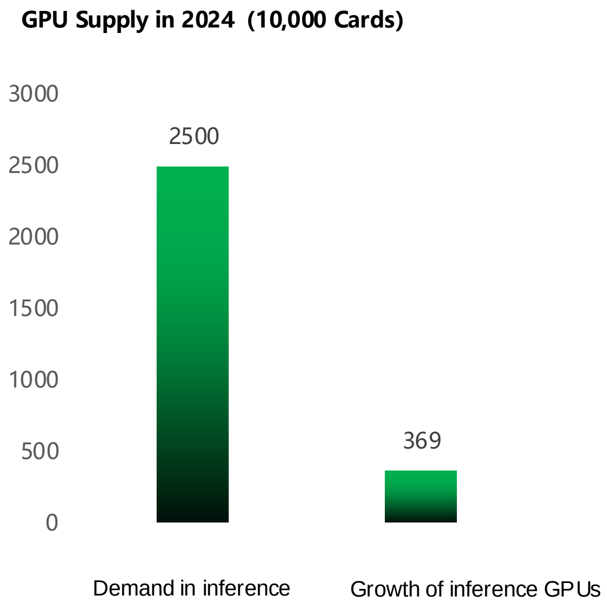


By 智研瞻：《中国人工智能数据中心（AIDC，智算中心）行业发展前景与投资战略规划分析报告》

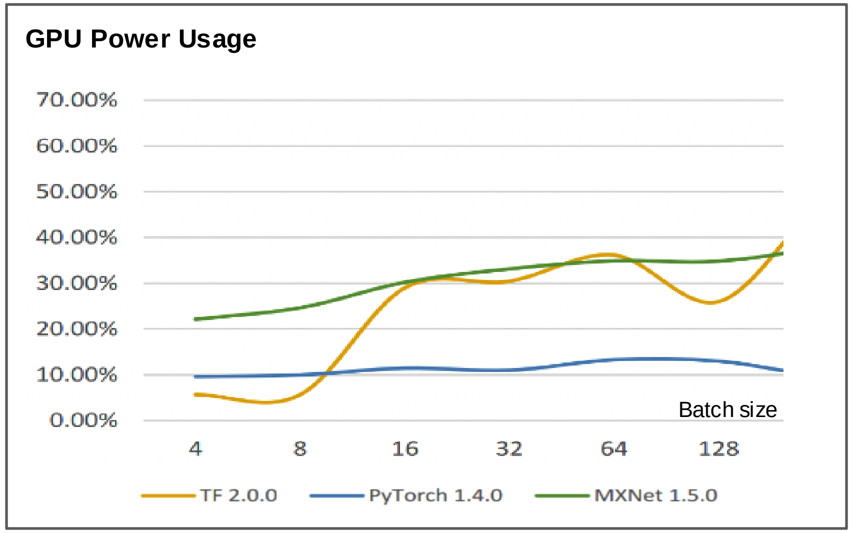
Challenges - GPU Supply & Utilization

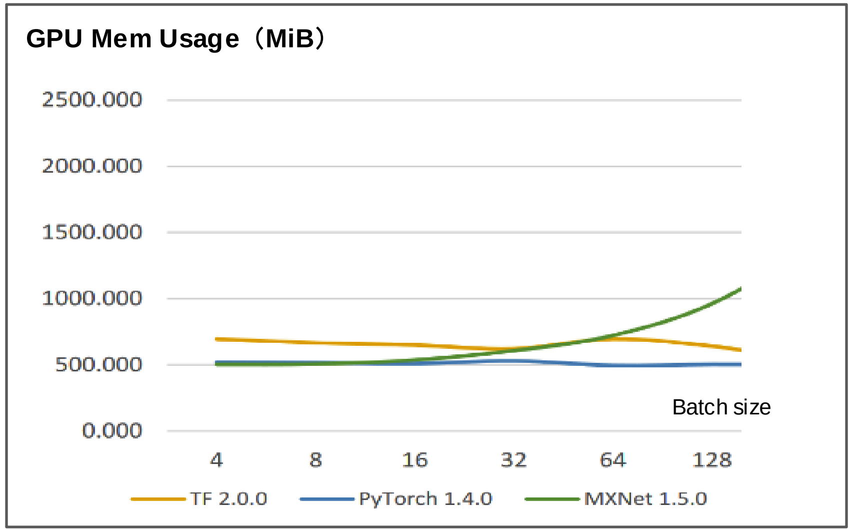
GPU Supply Shortage


GPU Supply in 2024 (10,000 Cards)


GPU Supply in 2024 (10,000 Cards)

Low Utilization


GPU Power Usage


GPU Mem Usage（MiB）

Low Efficiency - Reasons

1. Network Bottom Neck
   - Data communication in low efficiency

2. Storage Read & Write Efficiency
   - Dataset load is slow
   - checkpoint saving is slow

3. GPU Allocation is not Optimal
   - GPU Scheduling is not matching the AI scenario
   - Allocated GPU is more than required in Inference scenario

4. GPU Fault waste Training time

Success Story

d.run as a computing-hub for AIDC across China. empower

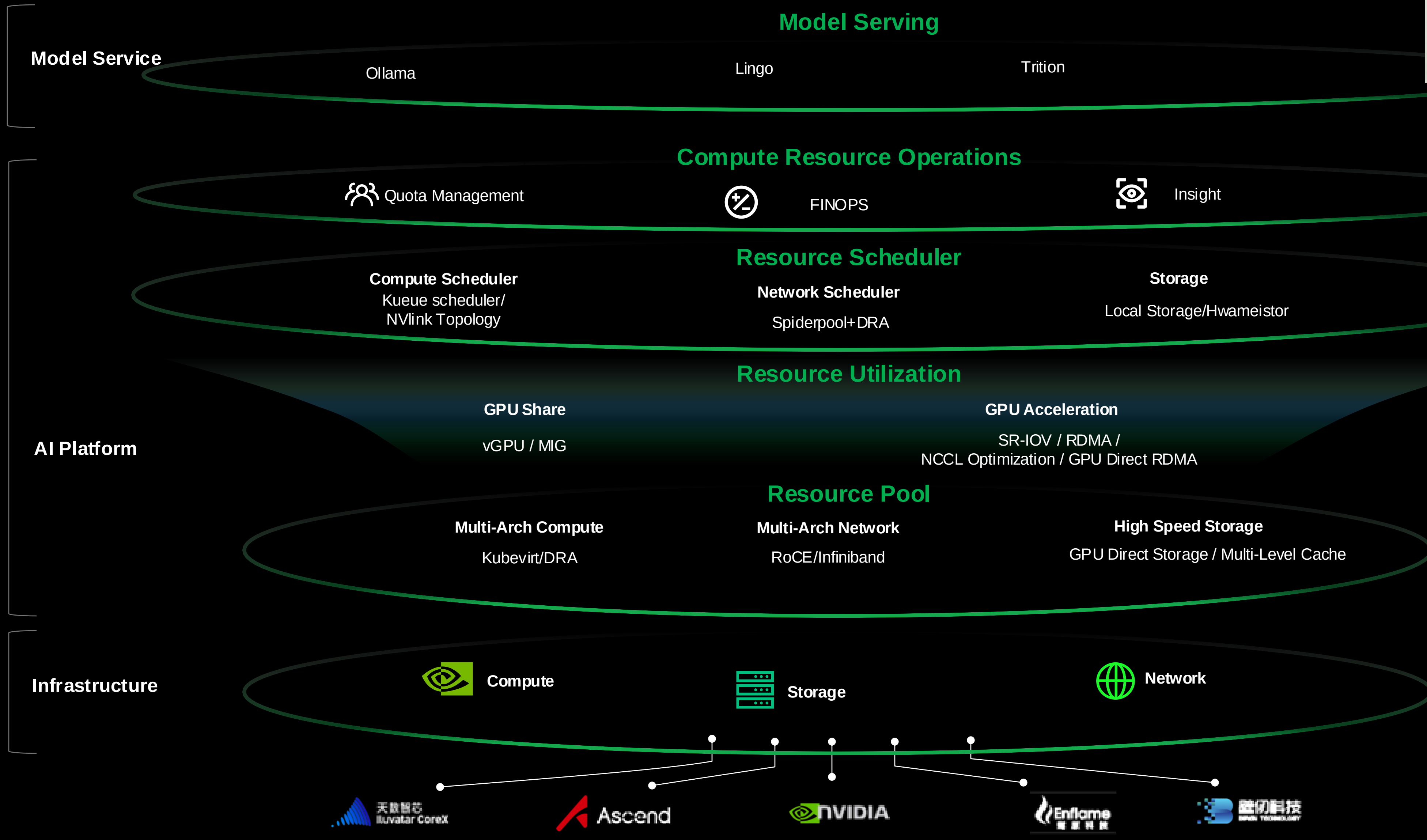
Both SaaS and On-Premises

GPU cost: 48% saving

GPU avg utilization : 25% --> 54% [1]

[1] based on statistic of two A100 clusters managed by d.run

Optimization in different Layers



Model Service

Model Serving

Ollama Lingo Trition

AI Platform

Compute Resource Operations

Quota Management FINOPS Insight

Resource Scheduler

Compute Scheduler
Kueue scheduler/
NVlink Topology

Network Scheduler
Spiderpool+DRA

Storage
Local Storage/Hwameistor

Resource Utilization

GPU Share
vGPU / MIG

GPU Acceleration
SR-IOV / RDMA /
NCCL Optimization / GPU Direct RDMA

Resource Pool

Multi-Arch Compute
Kubevirt/DRA

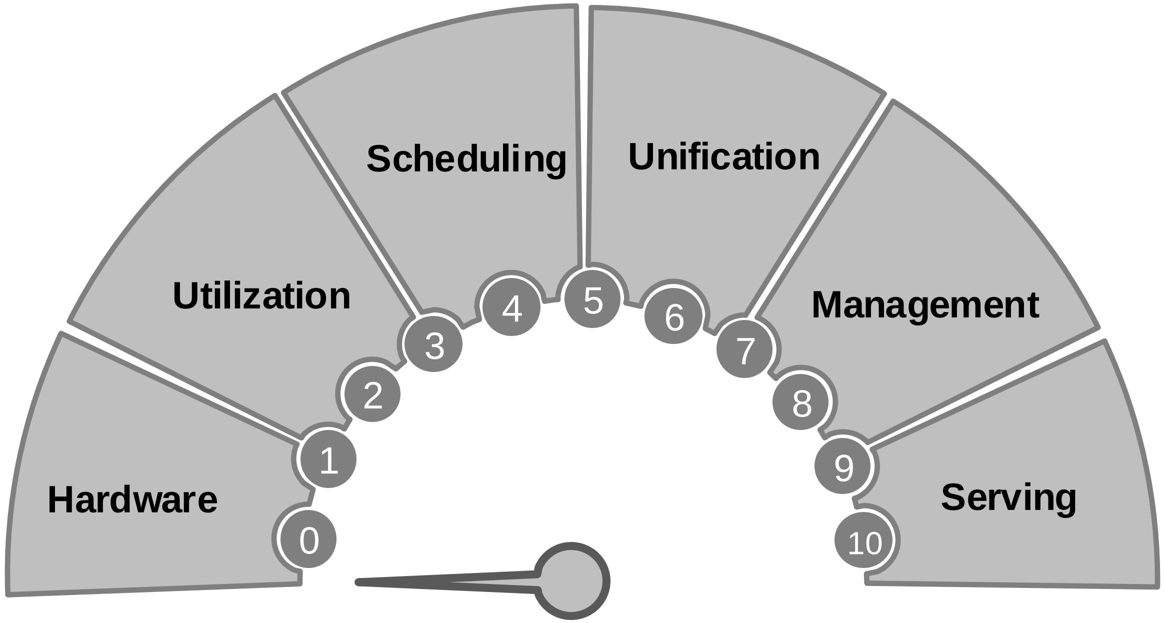
Multi-Arch Network
RoCE/Infiniband

High Speed Storage
GPU Direct Storage / Multi-Level Cache

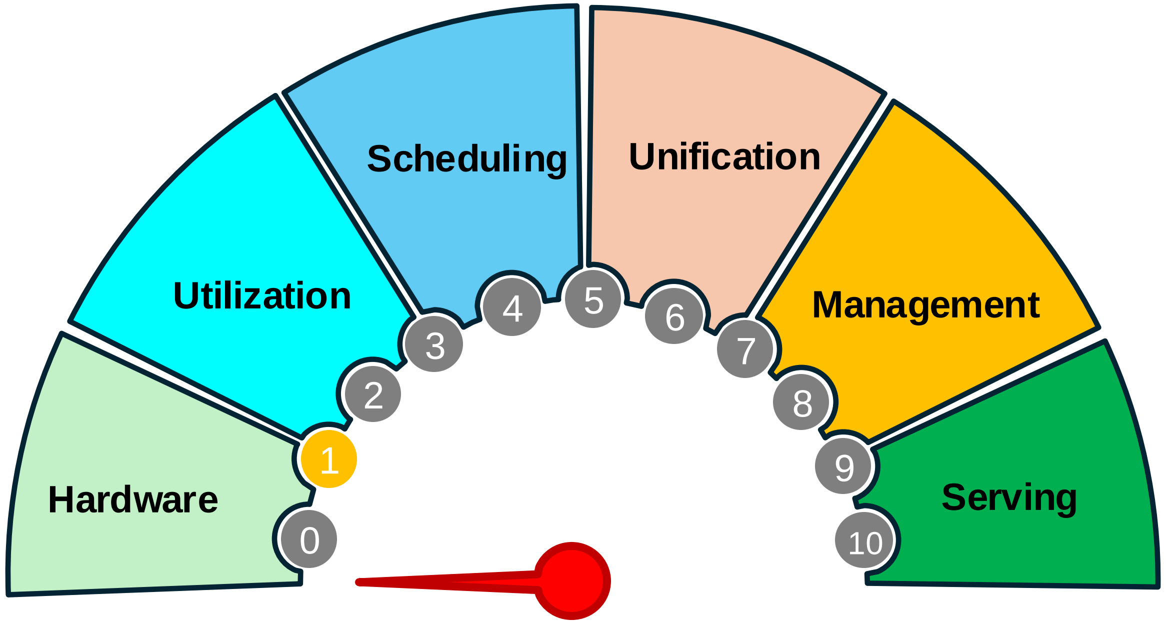
Infrastructure

Compute Storage Network

Journey starts !



1. Hardware Acceleration



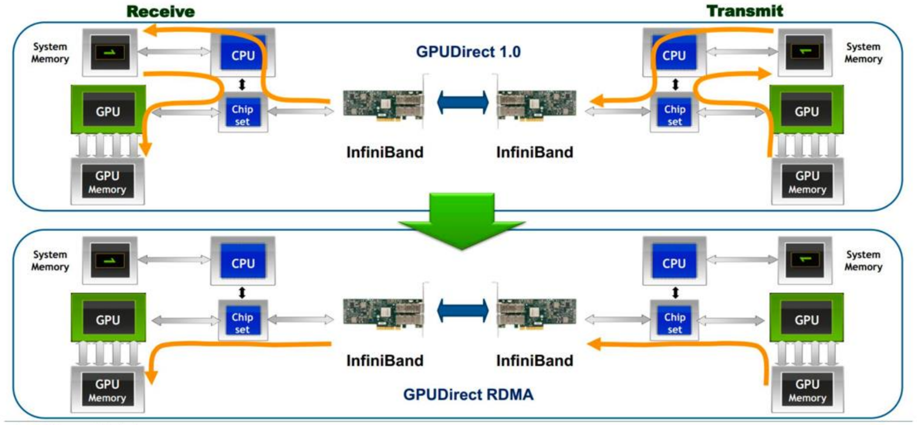
Network : GPU Acceleration

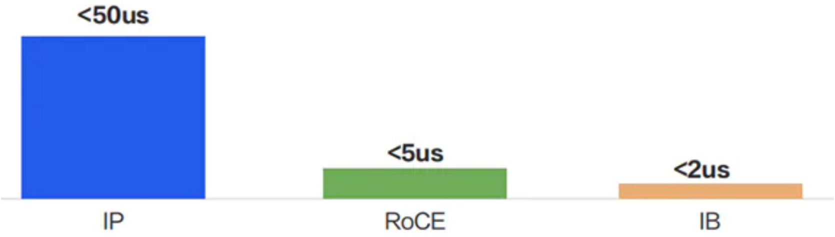
1. GPU Director RDMA to accelerate the GPU and RDMA Interface
2. Spider Pool Project Support different CNIs to work with RoCE/Infiniband

Spiderpool

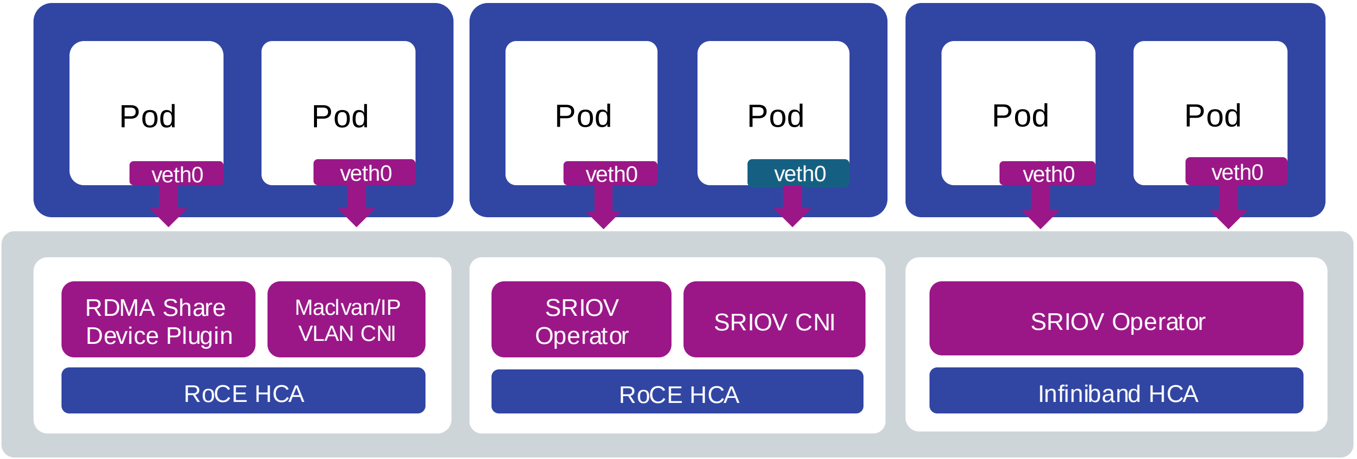
https://github.com/spidernet-io/spiderpool

- Based on Macvlan/IPVLAN CNI to Communicate by RoCE
- Based on SRIOV CNI to Communicate by RoCE
- Based on SRIOV CNI to Communicate by Infiniband





End-to-end communication latency of different technologies



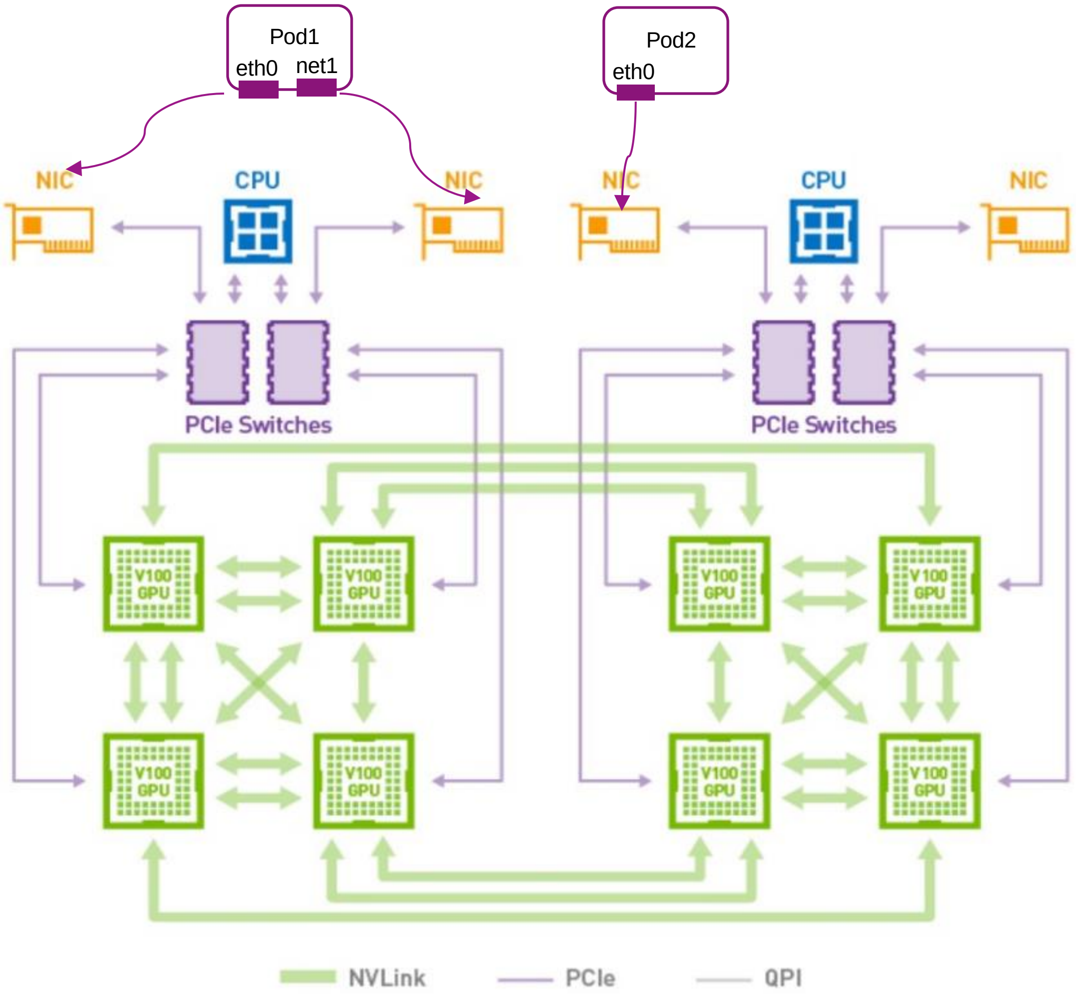
Macvlan/IPVLAN with RoCE Direct

SRIOV With RoCE Direct

SRIOV With Infiniband Direct

Network : GPU topology-aware scheduling

1. Pod Allocated with RDMA HCA affinity with same PCIe Switch
2. Based on DRA to dynamically select the RDMA device with best topology



Spiderpool

+ DRA

https://github.com/spidernet-io/spiderpool

Storage : fully leveraged

Use Legacy Inventory 利旧

- CSI enabling : External Storage, HPC Storage(Lustre, BeeGFS, HDFS)

Scenario Fit

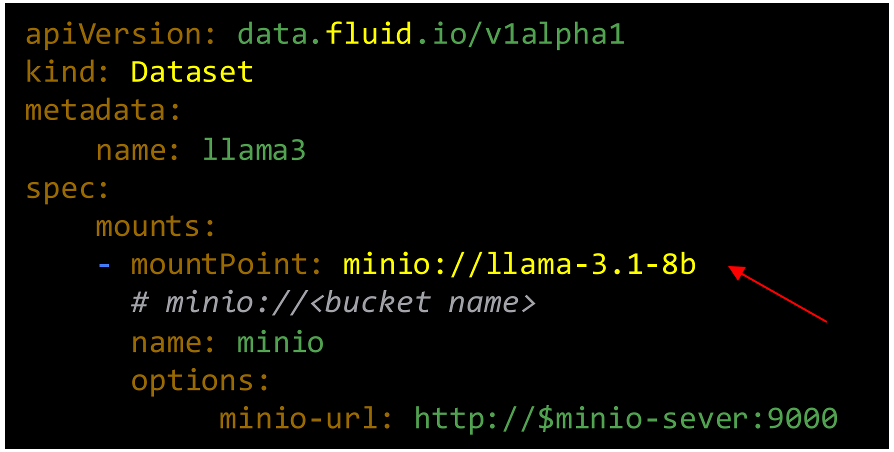
- Dataset: mostly R/O -> S3 + JuiceFS caching, Labeling -> write to S3
- Checkpoint Saving: IO Throughput(reduce GPU idle time), distributed storage(for resume training)
- Intermediate data: leverage local storage (HwameiStor , drbd HA is optional)
- Model: OCI image/raw file in S3
- Inference history: S3

Async checkpoint : dlrover
https://github.com/intelligent-machine-learning/dlrover

Storage : Accelerate

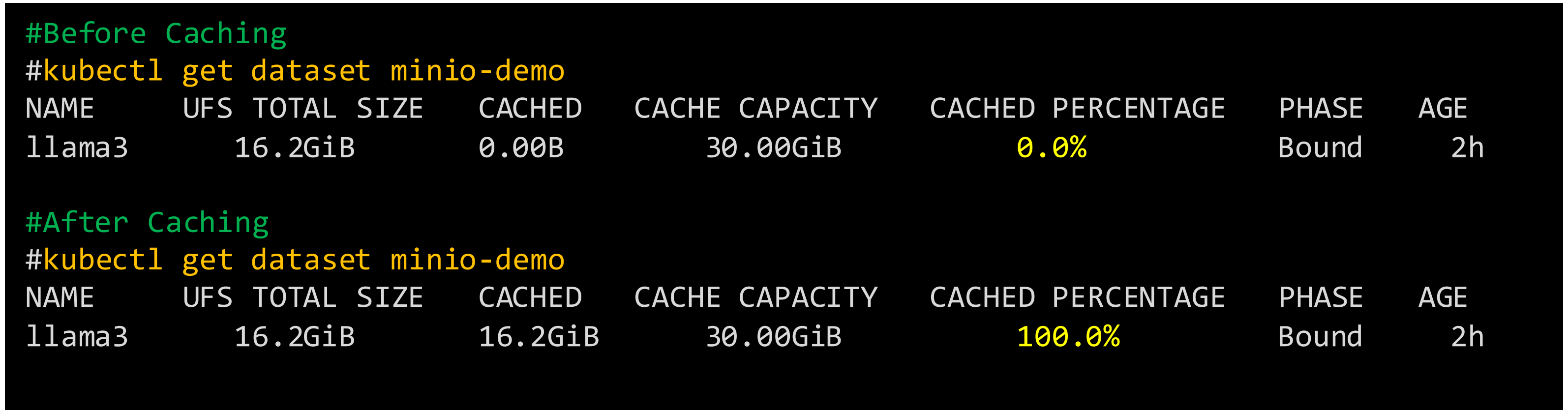
Fluid - K8s Data Acceleration (backed by Alluxio/JuiceFS..etc)

1. Define Dataset



```yaml
apiVersion: data.fluid.io/v1alpha1
kind: Dataset
metadata:
    name: llama3
spec:
    mounts:
    - mountPoint: minio://llama-3.1-8b
      # minio://<bucket name>
      name: minio
      options:
           minio-url: http://$minio-sever:9000
```

2. Mount PVC, trigger caching

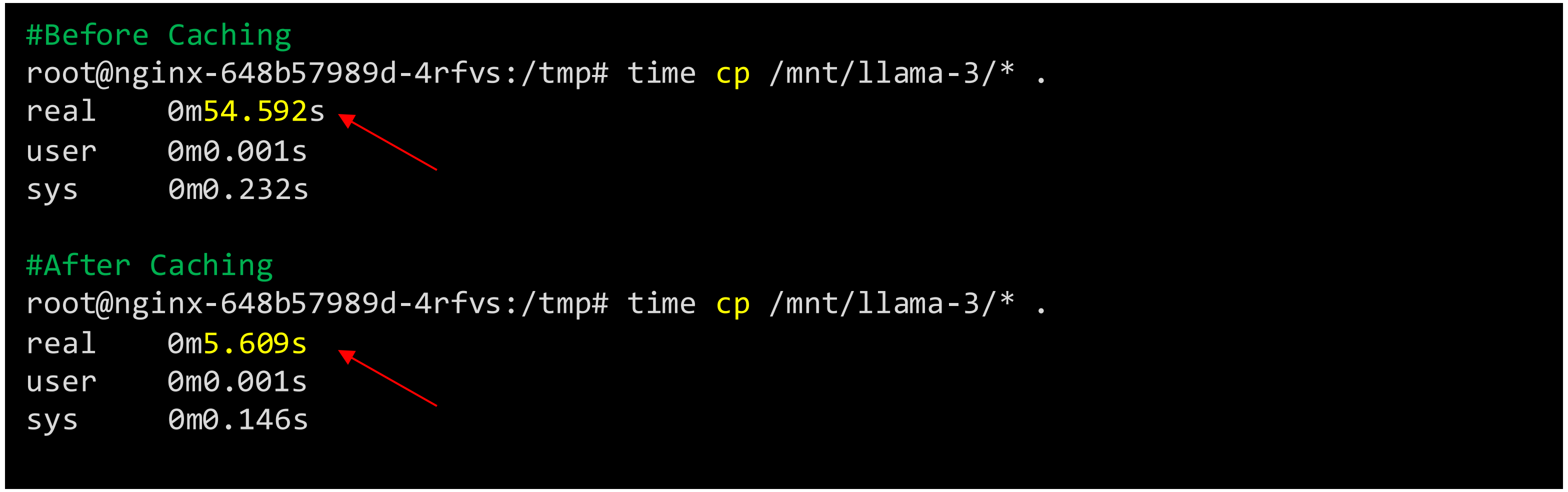


```
#Before Caching
#kubectl get dataset minio-demo
NAME      UFS TOTAL SIZE    CACHED    CACHE CAPACITY    CACHED PERCENTAGE    PHASE    AGE
llama3       16.2GiB        0.00B         30.00GiB            0.0%           Bound      2h

#After Caching
#kubectl get dataset minio-demo
NAME      UFS TOTAL SIZE    CACHED    CACHE CAPACITY    CACHED PERCENTAGE    PHASE    AGE
llama3       16.2GiB        16.2GiB       30.00GiB            100.0%         Bound      2h
```

3. x10 faster w/ cluster-local caching

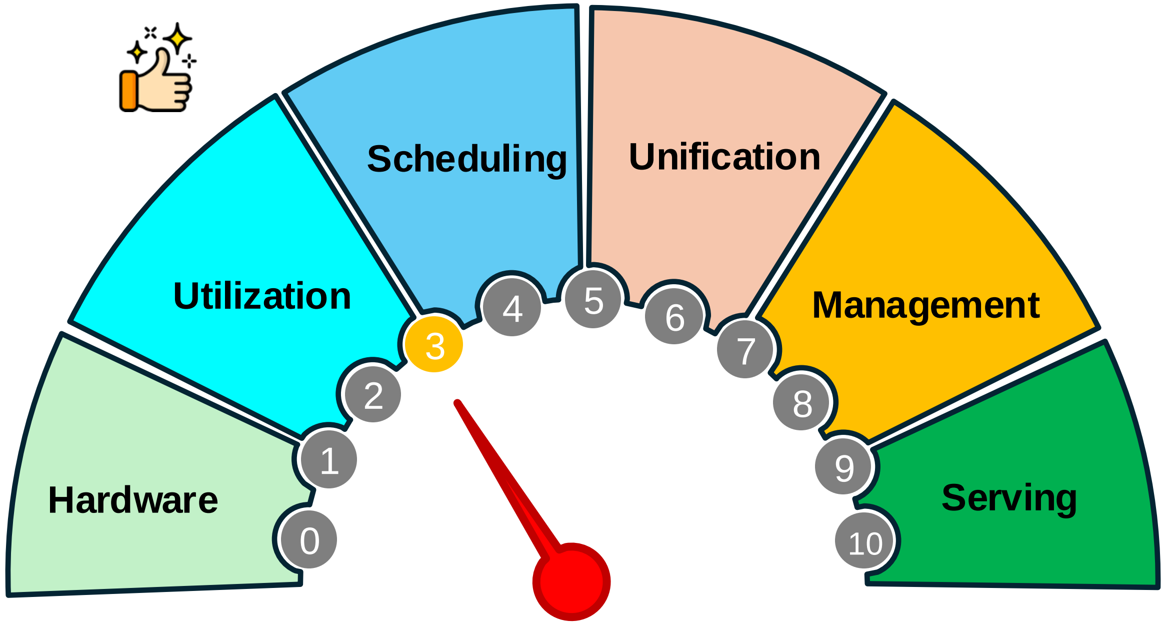
10 times faster !



```
#Before Caching
root@nginx-648b57989d-4rfvs:/tmp# time cp /mnt/llama-3/* .
real    0m54.592s
user    0m0.001s
sys     0m0.232s

#After Caching
root@nginx-648b57989d-4rfvs:/tmp# time cp /mnt/llama-3/* .
real    0m5.609s
user    0m0.001s
sys     0m0.146s
```

2. Resource Utilization



2. Maximize Utilization

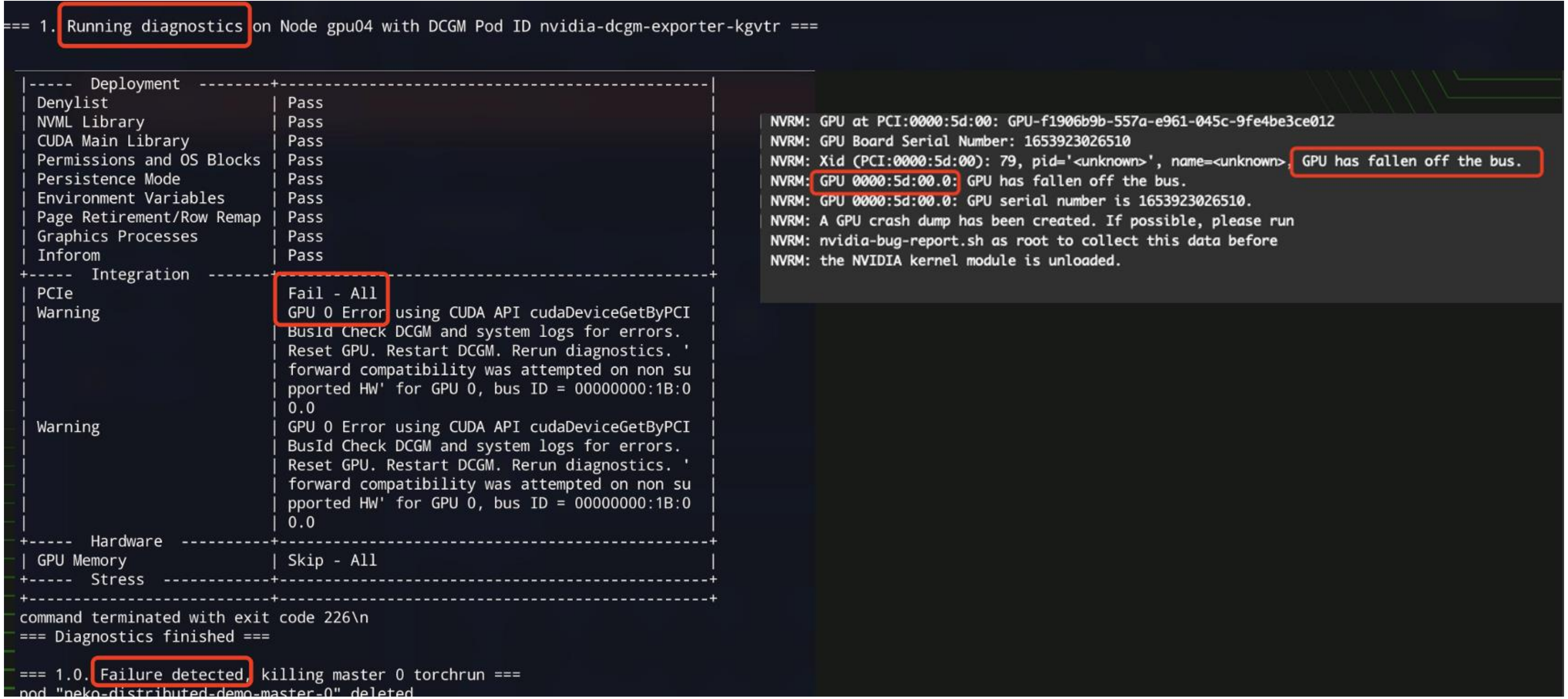
Challenges

- Workload mis-match (e.g. run a Phi-3-mini (3.8B) on a H100-80G)
- Occupied GPU is idle (e.g. occupied by notebook but just coding)
- GPU fragments (e.g. small task)
- Training recovery from failure (e.g.: waste in re-train from scratch)

2. Max Util - Training Recover

Recover from GPU failure

1. Detect : from log & nvdia diag
2. Recovery: re-schedule + re-run from checkpoint



```
=== 1. Running diagnostics on Node gpu04 with DCGM Pod ID nvidia-dcgm-exporter-kgvtr ===

|-----  Deployment  --------+------------------------------------------------|
| Denylist                  | Pass                                           |
| NVML Library              | Pass                                           |
| CUDA Main Library         | Pass                                           |
| Permissions and OS Blocks | Pass                                           |
| Persistence Mode          | Pass                                           |
| Environment Variables     | Pass                                           |
| Page Retirement/Row Remap | Pass                                           |
| Graphics Processes        | Pass                                           |
| Inforom                   | Pass                                           |
+-----  Integration  -------+------------------------------------------------+
| PCIe                      | Fail - All                                     |
| Warning                   | GPU 0 Error using CUDA API cudaDeviceGetByPCI  |
|                           | BusId Check DCGM and system logs for errors.   |
|                           | Reset GPU. Restart DCGM. Rerun diagnostics. '  |
|                           | forward compatibility was attempted on non su  |
|                           | pported HW' for GPU 0, bus ID = 00000000:1B:0  |
|                           | 0.0                                            |
| Warning                   | GPU 0 Error using CUDA API cudaDeviceGetByPCI  |
|                           | BusId Check DCGM and system logs for errors.   |
|                           | Reset GPU. Restart DCGM. Rerun diagnostics. '  |
|                           | forward compatibility was attempted on non su  |
|                           | pported HW' for GPU 0, bus ID = 00000000:1B:0  |
|                           | 0.0                                            |
+-----  Hardware  ----------+------------------------------------------------+
| GPU Memory                | Skip - All                                     |
+-----  Stress  ------------+------------------------------------------------+
+---------------------------+------------------------------------------------+
command terminated with exit code 226\n
=== Diagnostics finished ===

=== 1.0. Failure detected, killing master 0 torchrun ===
pod "neko-distributed-demo-master-0" deleted
```

```
NVRM: GPU at PCI:0000:5d:00: GPU-f1906b9b-557a-e961-045c-9fe4be3ce012
NVRM: GPU Board Serial Number: 1653923026510
NVRM: Xid (PCI:0000:5d:00): 79, pid='<unknown>', name=<unknown>, GPU has fallen off the bus.
NVRM: GPU 0000:5d:00.0: GPU has fallen off the bus.
NVRM: GPU 0000:5d:00.0: GPU serial number is 1653923026510.
NVRM: A GPU crash dump has been created. If possible, please run
NVRM: nvidia-bug-report.sh as root to collect this data before
NVRM: the NVIDIA kernel module is unloaded.
```

2. Max Util - Optimal GPU Fit

- Example: There’s a tiered GPU cluster: mixing H/A100, A6000, 4090
  - LLM **training**   : H/A100 with best NVLink connect
  - LLM **Inference**: A6000/4090 with better cost performance

“nvidia.com/gpu: 1” will not tell ->

DRA

```
apiVersion: apps/v1
kind: Deployment
…
    spec:
      resourceClaims:
      - name: a100
        source:
          resourceClaimTemplateName: a100
```

Workload specific A100

```
apiVersion: gpu.resource.nvidia.com/v1alpha1
kind: GpuClaimParameters
metadata:
  name: a100
spec:
    count: 1
     …
    - productName: "*a100*”
```

DRA defines A100 scale offering

```
apiVersion: gpu.resource.nvidia.com/v1al
kind: GpuClaimParameters
metadata:
  name: a6000
spec:
    count: 1
     …
    - productName: "*a6000*"
    - orExpression:
      - index: 0
      - index: 1
      - index: 3
      - index: 5
```

DRA defines A6000 offering

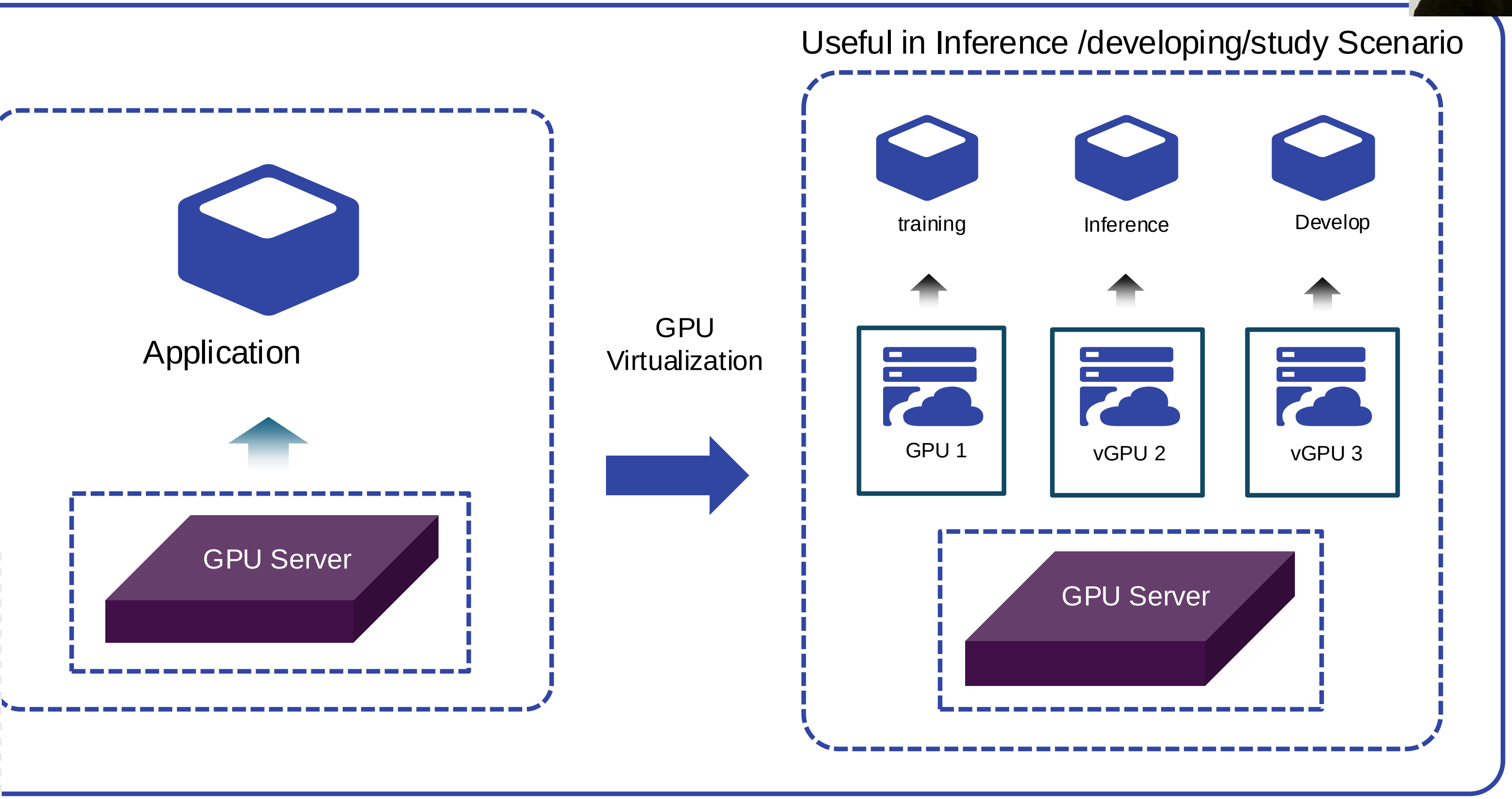
Moreover, the gpu0/1/3/5 on each nod

More example and detail in
https://github.com/NVIDIA/K8s-dra-driver

2. Max Utilization : vGPU

GPU Partition

Case1 :

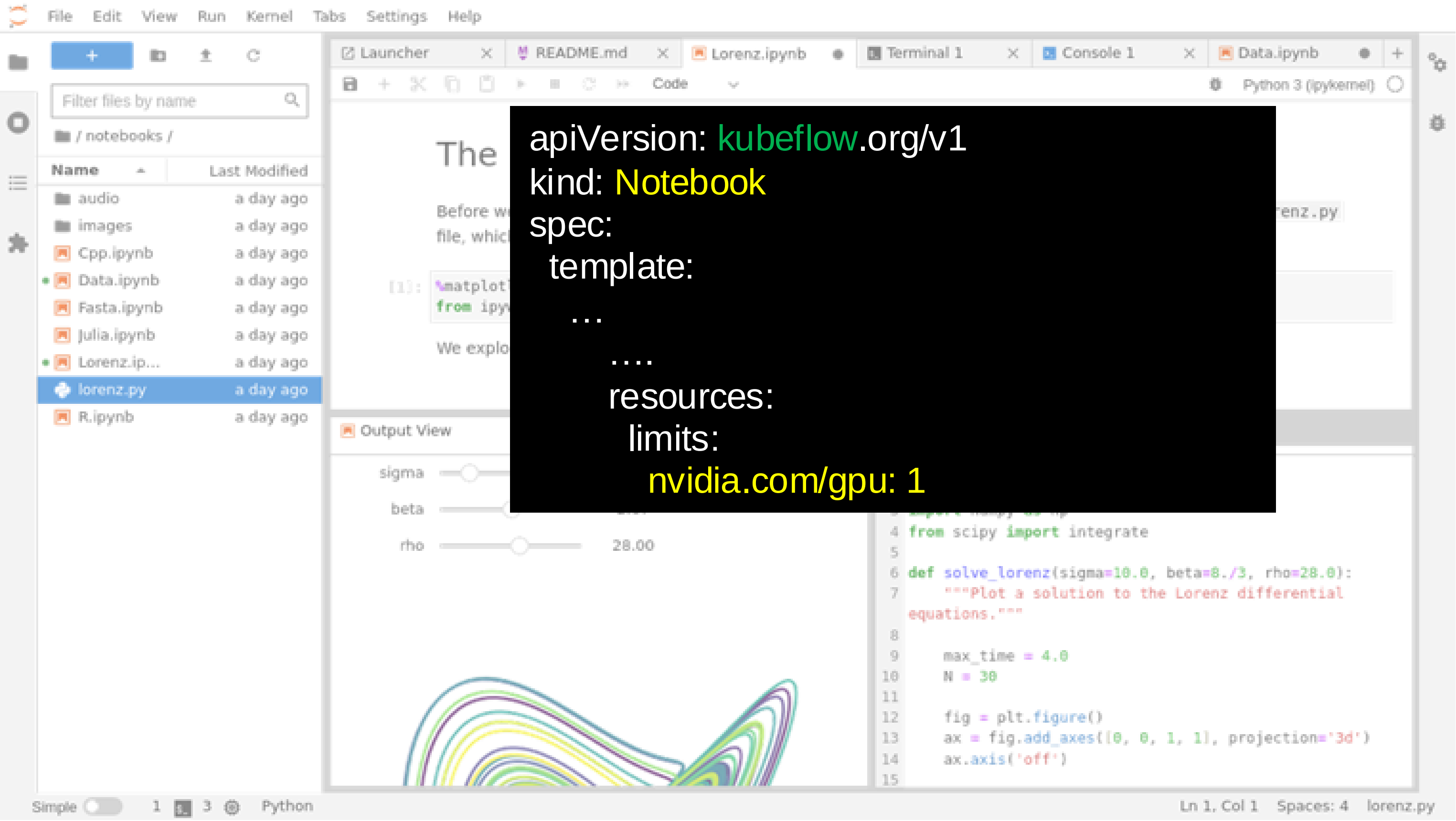


1 GPU Used by **1** App
Utilization is lower than 20%

1 GPU Used by **N** App
Utilization improved to 70%

2. Max Utilization : vGPU

Case 2: Jupyter notebook 80%~90% time for coding or non-gpu works



```
apiVersion: kubeflow.org/v1
kind: Notebook
spec:
  template:
    …
      ….
      resources:
        limits:
          nvidia.com/gpu: 1
```

```python
from scipy import integrate

def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential
equations."""

    max_time = 4.0
    N = 30

    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')
```

Some ways out below, but not user-friendly

- auto-shutdown of idle notebook
- using arena CLI to submit async job on demand

goal

- Split GPU at will
- Over-commit of GPU Memory

2. Max Utilization : vGPU

Solution 1

M.I.G - GPU partition (physical)

- Good performance/isolation, but limited on fixed scale/max 7 slices

- **Dynamic MIG** w/ **DRA** supported (create MIG “device” on demand)

https://github.com/NVIDIA/K8s-dra-driver/blob/main/demo/specs/quickstart/gpu-test4.yaml

Pod

```
resourceClaims:
   - name: mig-enabled-gpu
     source:
        resourceClaimTemplateName: mig-enabled-gpu
   - name: mig-2g
     source:
        resourceClaimTemplateName: mig-2g.10gb
```

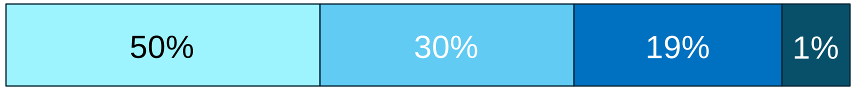
MIG = Multi-Instance GPU

2. Max Utilization : vGPU & Over-Commit

Solution 2

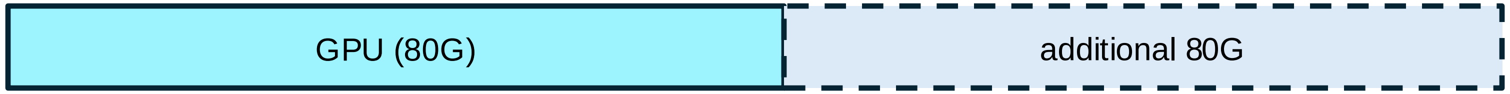
GPU Middleware – **HAMi** (logical partition)

1. GPU slicing size - "at will"



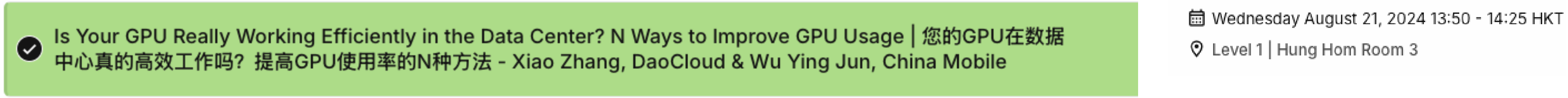
```
resources:
  limits:
    nvidia.com/gpucores :  "50"     # 50% GPU time
    nvidia.com/gpumem: 3000         # 3GB HBM
```

2. HBM sharing & over-commit



On single A100,
But 2 pods can request both 80G at the same time, and use/share HBM at diff time.
When two pods use both 80G, "swap" happens: system memory will help

more details in

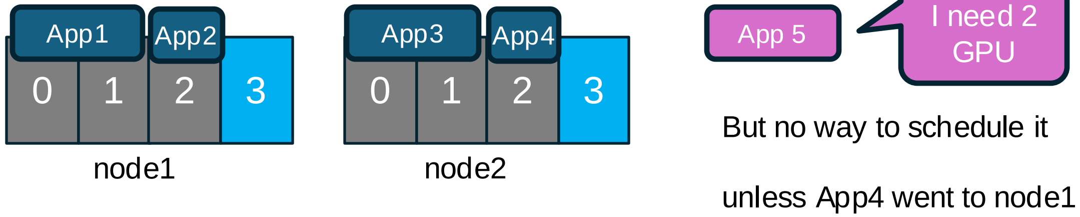


Is Your GPU Really Working Efficiently in the Data Center? N Ways to Improve GPU Usage | 您的GPU在数据中心真的高效工作吗？提高GPU使用率的N种方法 - Xiao Zhang, DaoCloud & Wu Ying Jun, China Mobile

Wednesday August 21, 2024 13:50 - 14:25 HKT
Level 1 | Hung Hom Room 3

Binpack

Resource fragments



App1 App2
0 1 2 3
node1

App3 App4
0 1 2 3
node2

App 5

I need 2 GPU

But no way to schedule it

unless App4 went to node1

Solution #1:

K8s scheduling-plugin “NodeResource”

1.MostAllocated policy : favoring nodes with higher allocation.

2. GPU with more weight than others

```
apiVersion: kubescheduler.config.K8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- pluginConfig:
  - name: NodeResourcesFit
    args:
      scoringStrategy:
       type: MostAllocated
        resources:
        - name: nvidia.com/gpu
          weight: 3           # more favor to GPU
        - name: cpu
          weight: 1
        - name: memory
          weight: 1
```

https://kubernetes.io/zh-cn/docs/concepts/scheduling-eviction/resource-bin-packing/
https://github.com/kubernetes-sigs/scheduler-plugins/tree/master/pkg/noderesources

Solution #2: VOLCANO

https://volcano.sh/en/docs/plugins/#binpack

Global config of volcano to enable binpack

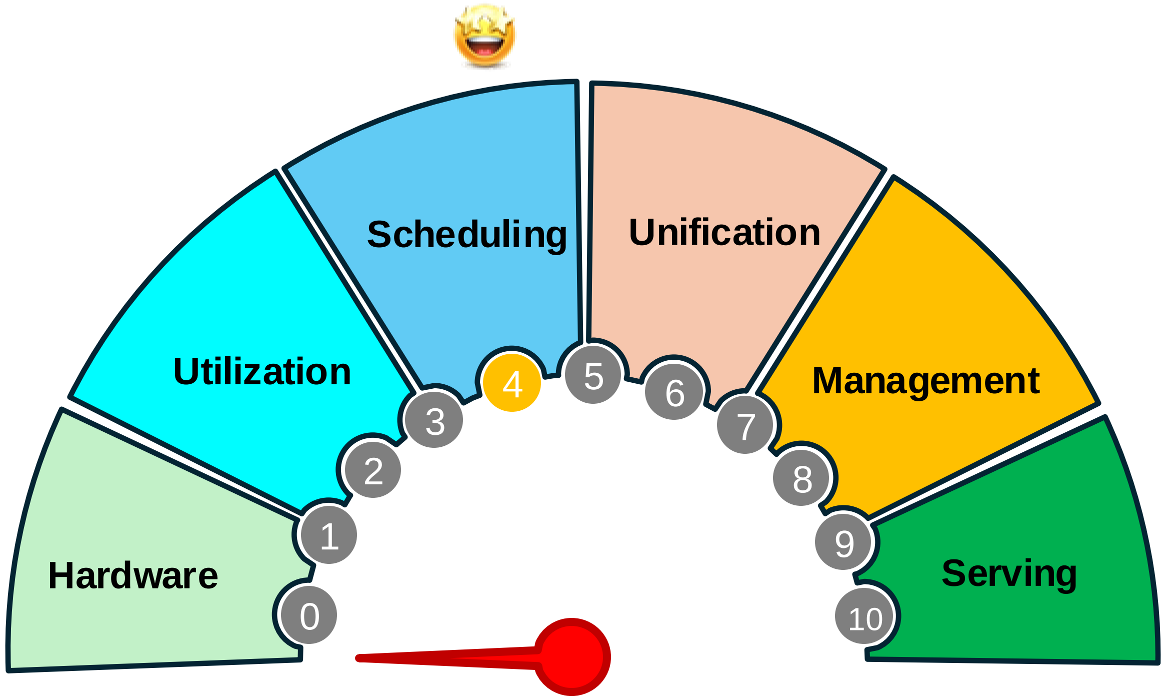
Solution #3 HAMi

https://github.com/Project-HAMi/HAMi

vGPU-level binpack

(put 0.5 vGPU + 0.3 vGPU pods on the same physical GPU)

3. Task Queue. & Scheduling



3. Task Mgmt : co-location

1. **Put co-locating jobs deployed** in one queue

2. **Setting Pod Priority Class,** inference Job seizes the CPU with high priority, and training task is evicted first in case of OOM

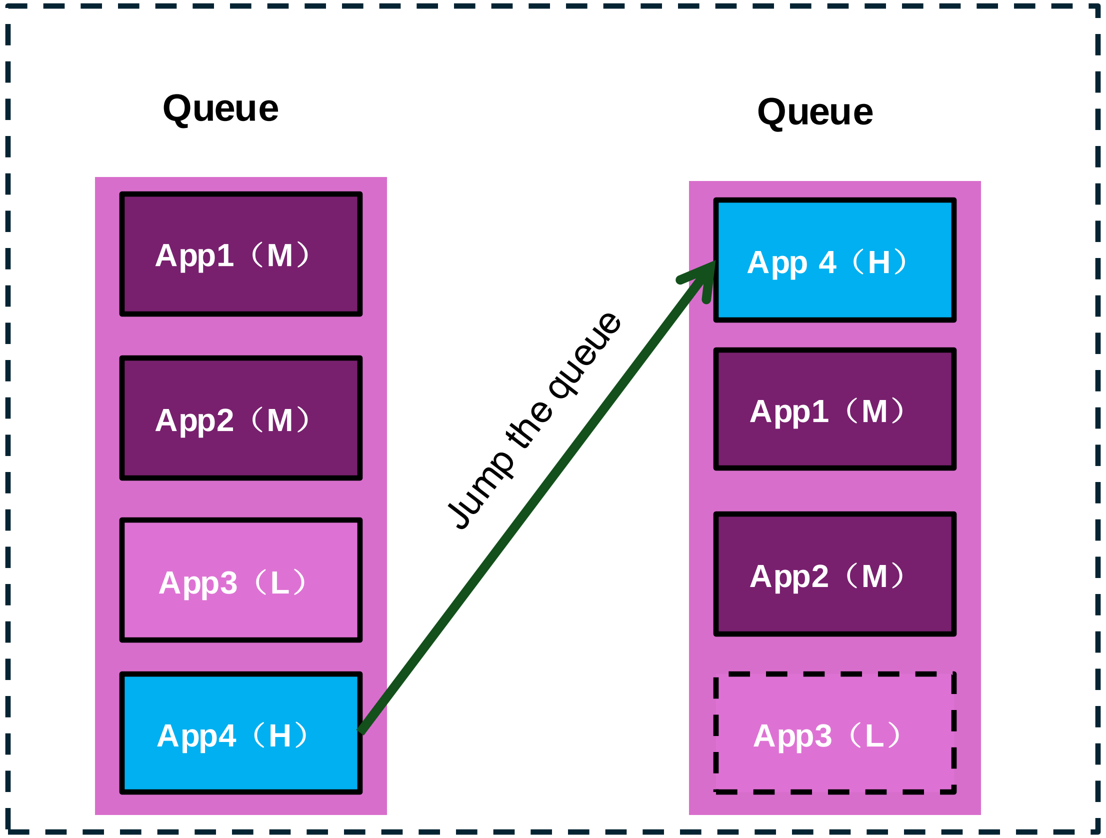
3. **Setting Job Priority in Queue**, inference Job use the GPU with high priority, Training Job will be suppressed

4. **Setting CPU Burst** (kernel 5.14+), It can help the latency sensitive workload to ”break” CPU limit sometimes.

5. **Namespace Elastic Quota,** different NS can share GPU quota, to improve the GPU usage

6. **Fair Schedule**, avoid some workload never get its requirement fulfilled , and “starve to death”

3. Task Mgmt : co-location



When App4(H) comes in, it “jumps the queue”

App3(L) be preempted or suspended

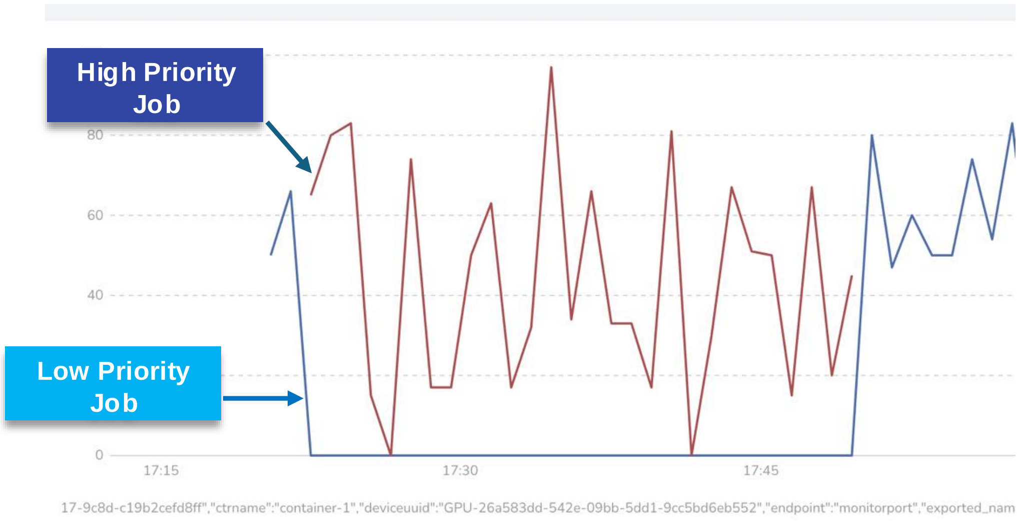
- Jobs/deployments mixed deployed in one queue
- Critical job sets high priority

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
name: my-high-priority
preemptionPolicy: PreemptLowerPriority
value: 100000
```

(1) Pod’s `PriorityClass`

```
apiVersion: kueue.x-k8s.io/v1beta1
kind: WorkloadPriorityClass
metadata:
  name: mediium-priority
value: 10000
```

(2) Kueue’s `WorkloadPriorityClass`



```
17-9c8d-c19b2cefd8ff","ctrname":"container-1","deviceuuid":"GPU-26a583dd-542e-09bb-5dd1-9cc5bd6eb552","endpoint":"monitorport","exported_nam
```

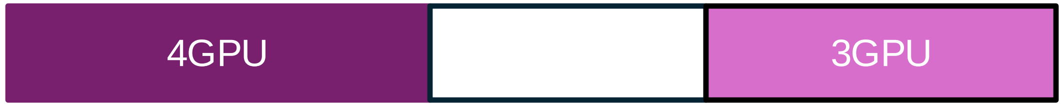
(3) ` CUDA_TASK_PRIORITY ` in HAMi

3. Task Mgmt : Elastic Quota & HPA

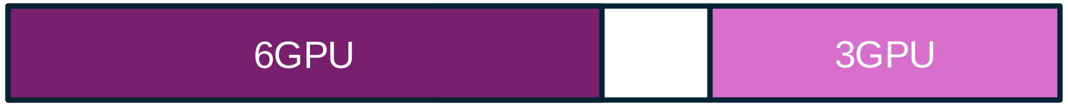
Training & Inference workload in different Namespace

Total: 10 GPU
NS Quota A (min:4, max:6) ➔ App-A
NS Quota B (min:6, max:8)) ➔ App-B

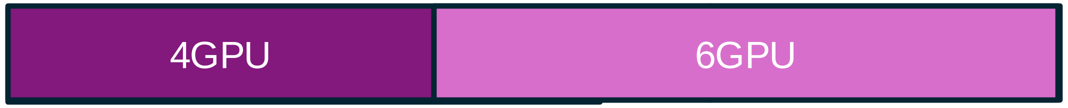
AppA use 4GPU (min), AppB use 3GPU, everything is OK



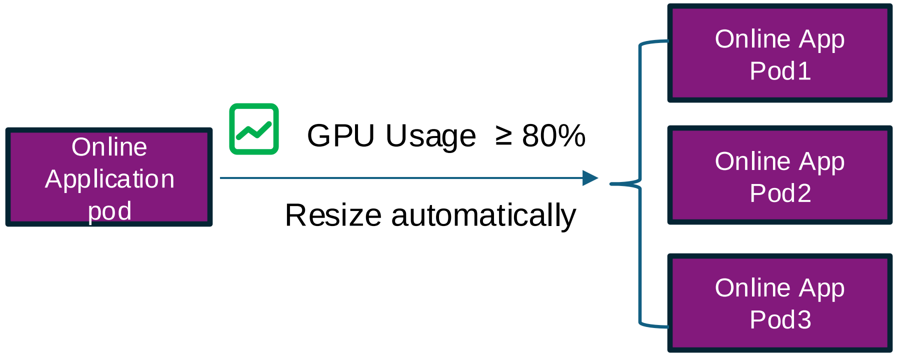
AppA use 6GPU (max), AppB use 3GPU, everything is OK



AppB need another 3 GPUs, resource eviction happened , AppA retain the GPU back , AppB get 6 GPUs (min)



HPA rule for Inference Application



Metrics:

GPU utilization may not be good practice

Try "Triton inference Queue-Time"
nv_inference_queue_duration_us

Elastic Quota by Capacity Scheduler:
https://github.com/kubernetes-sigs/scheduler-plugins/tree/master/pkg/capacityscheduling

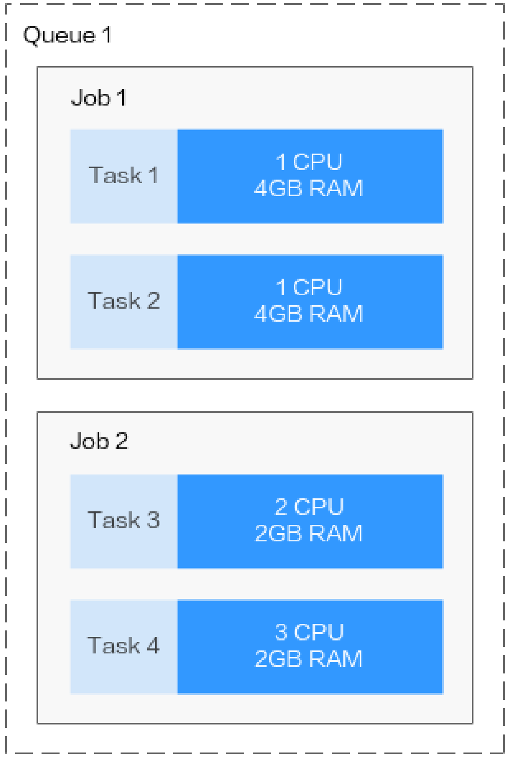
https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/user_guide/metrics.html

3. Task Mgmt: Fair Schedule

[Volcano DRF]

when in resource shortage, small tasks are fulfilled as possible, Then left part allocated fairly to other tasks

- Resource is allocated on an increasing demand basis
- Each user receives no more resources than needs
- Unsatisfied users share the remaining resources equally



Queue 1

Job 1

Task 1 — 1 CPU 4GB RAM

Task 2 — 1 CPU 4GB RAM

Job 2

Task 3 — 2 CPU 2GB RAM

Task 4 — 3 CPU 2GB RAM

Job 1:

Total CPU：2 CPU

CPU Share = 2/10 = 0.2

Total Memory：8GB

Memory Share = 8/20 = 0.4

Dominant resource is Memory

Share = 0.4

Job 2:

Total CPU：5 CPU

CPU Share = 5/10 = 0.5

Total Memory：4GB

Memory Share = 4/20 = 0.2

Dominant resource is CPU

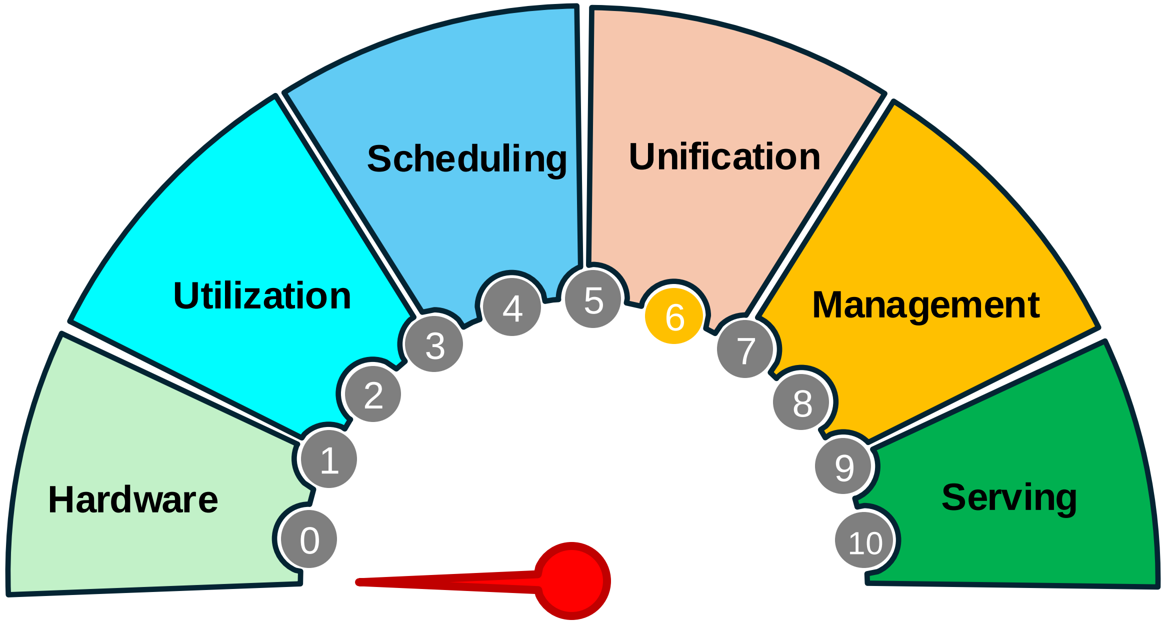
Share = 0.5

Cluster Resources：10 CPU, 20GB RAM, 0 GPU

Share = Total Request / Cluster Resources

Ways to calculate the jobs “share” value

4. Unified Architecture



4. Unified Architecture - VM

- Background : VM still lives, due to legacy apps / mgmt-policy
- Cure : **KubeVirt** can help out
- Benefits
  - Reuse the same underlaying hardware
  - Reduce the IaaS software cost

just 3 Steps:

1. **Nvidia gpu-operator: ** support container and VM

https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-operator-kubevirt.html#install-the-gpu-operator

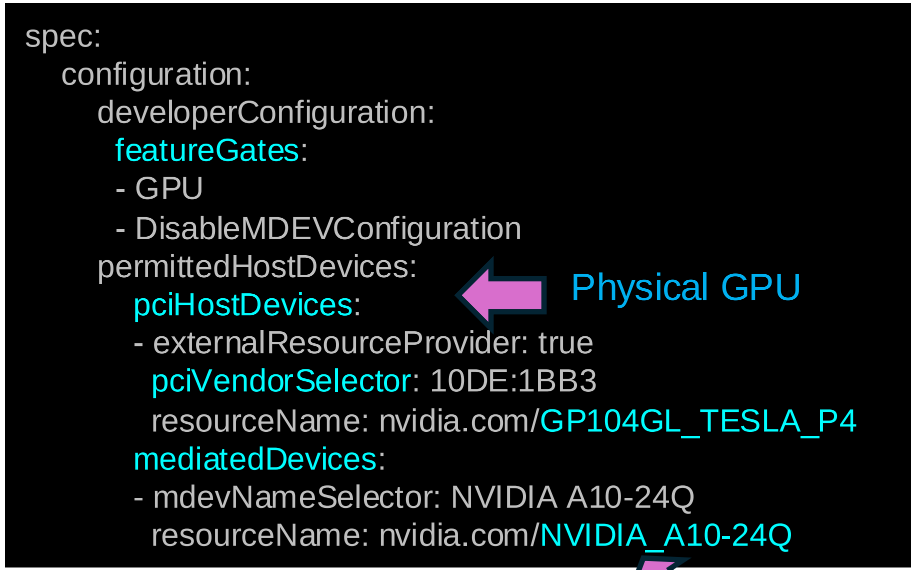
- Passthrough GPU
- Nvidia vGPU (License required)

2. Config KubeVirt CR

add specific gpu device

3. VM instance:

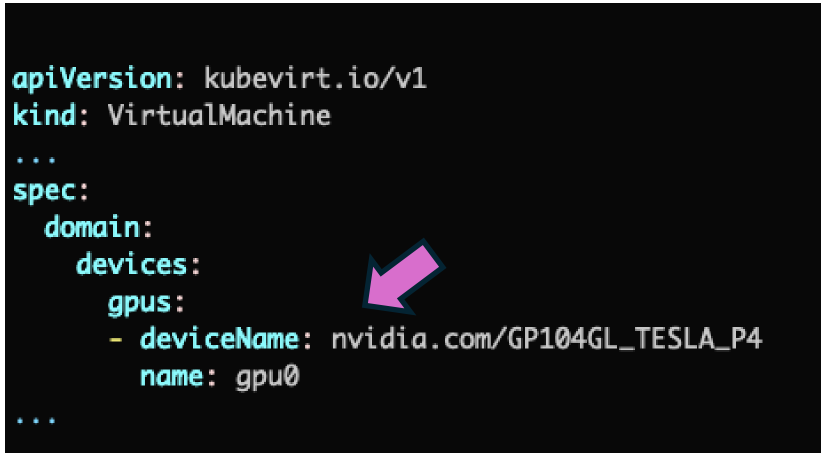
device yaml specific GPU



```
spec:
  configuration:
    developerConfiguration:
      featureGates:
      - GPU
      - DisableMDEVConfiguration
    permittedHostDevices:
      pciHostDevices:
      - externalResourceProvider: true
        pciVendorSelector: 10DE:1BB3
        resourceName: nvidia.com/GP104GL_TESLA_P4
      mediatedDevices:
      - mdevNameSelector: NVIDIA A10-24Q
        resourceName: nvidia.com/NVIDIA_A10-24Q
```

Physical GPU

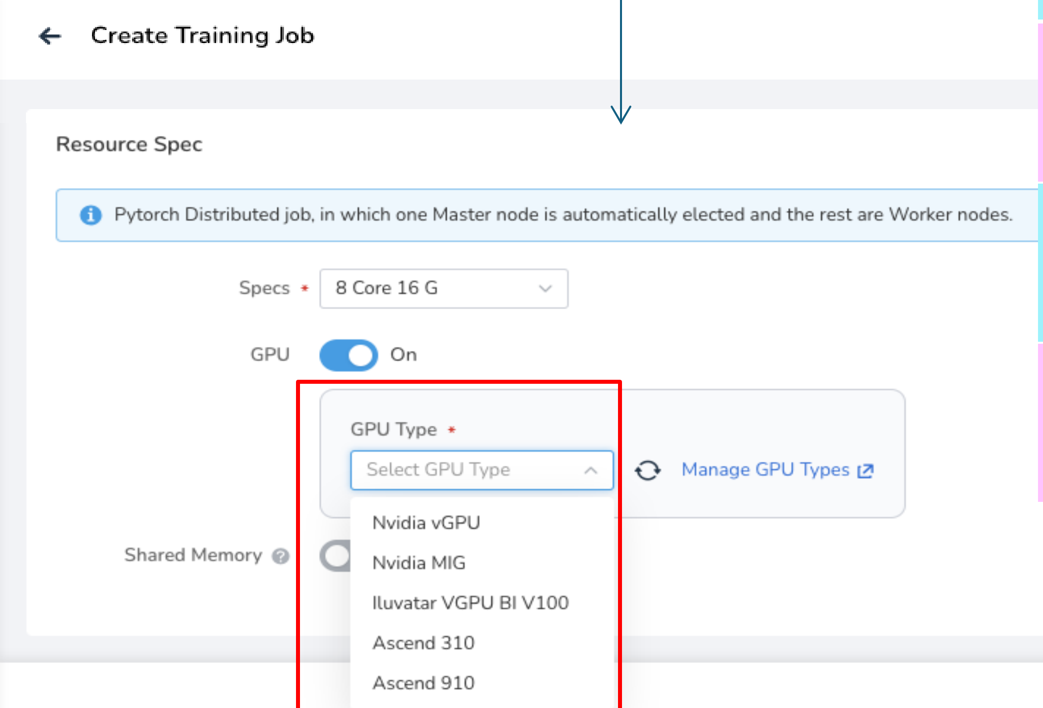
Nvidia vGPU



```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
...
spec:
  domain:
    devices:
      gpus:
      - deviceName: nvidia.com/GP104GL_TESLA_P4
        name: gpu0
...
```

4. Unified Architecture - Heterogeneous

- diff GPU model/brands together
- Training:
  - Mixed training is a great challenges - Buckets Effect
- Inference
  - Individual cluster for each xPU type - ✅
  - Mixed cluster, diff xGPU in diff node - ✅
  - Mixed node is challenging (DRA or change device-plugin)



← Create Training Job

Resource Spec

ⓘ Pytorch Distributed job, in which one Master node is automatically elected and the rest are Worker nodes.

Specs * 8 Core 16 G

GPU On

GPU Type *

Select GPU Type

Manage GPU Types

Nvidia vGPU

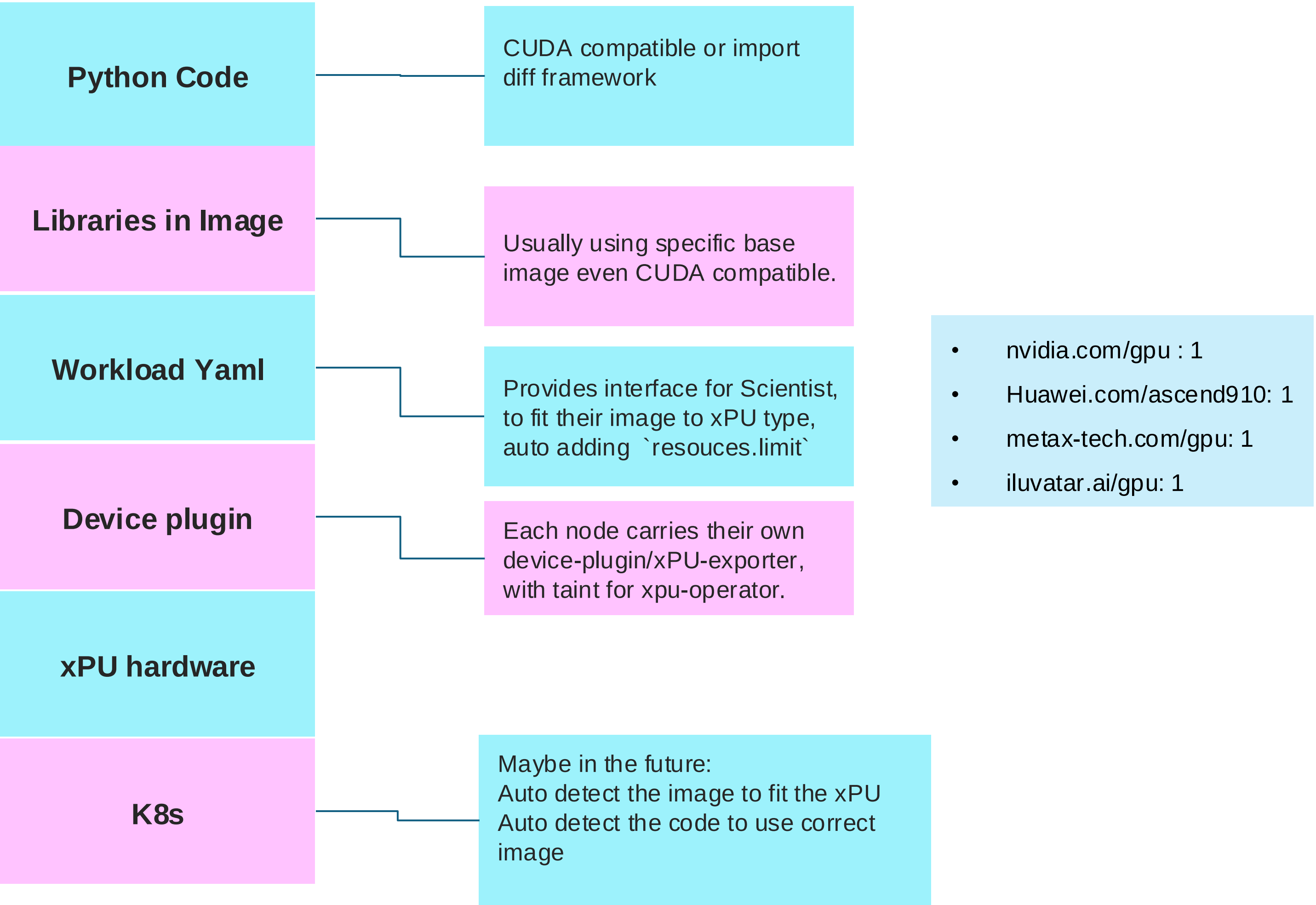
Nvidia MIG

Iluvatar VGPU BI V100

Ascend 310

Ascend 910

Shared Memory



Python Code — CUDA compatible or import diff framework

Libraries in Image — Usually using specific base image even CUDA compatible.

Workload Yaml — Provides interface for Scientist, to fit their image to xPU type, auto adding `resouces.limit`

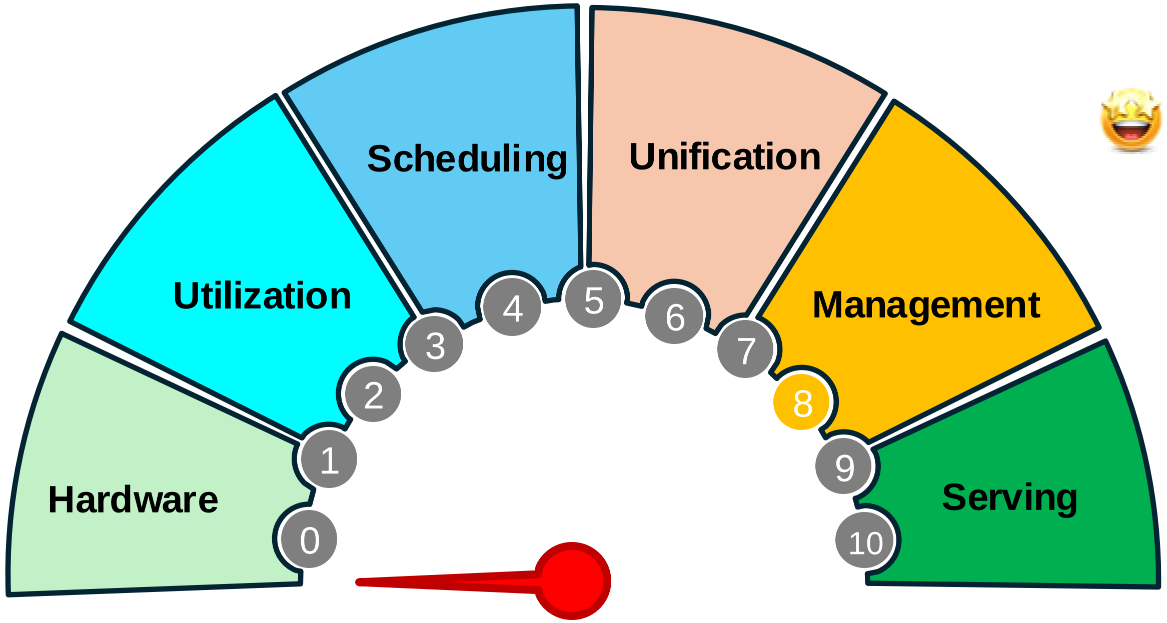
- nvidia.com/gpu : 1
- Huawei.com/ascend910: 1
- metax-tech.com/gpu: 1
- iluvatar.ai/gpu: 1

Device plugin — Each node carries their own device-plugin/xPU-exporter, with taint for xpu-operator.

xPU hardware

K8s — Maybe in the future:
Auto detect the image to fit the xPU
Auto detect the code to use correct image

5. Management Philosophy

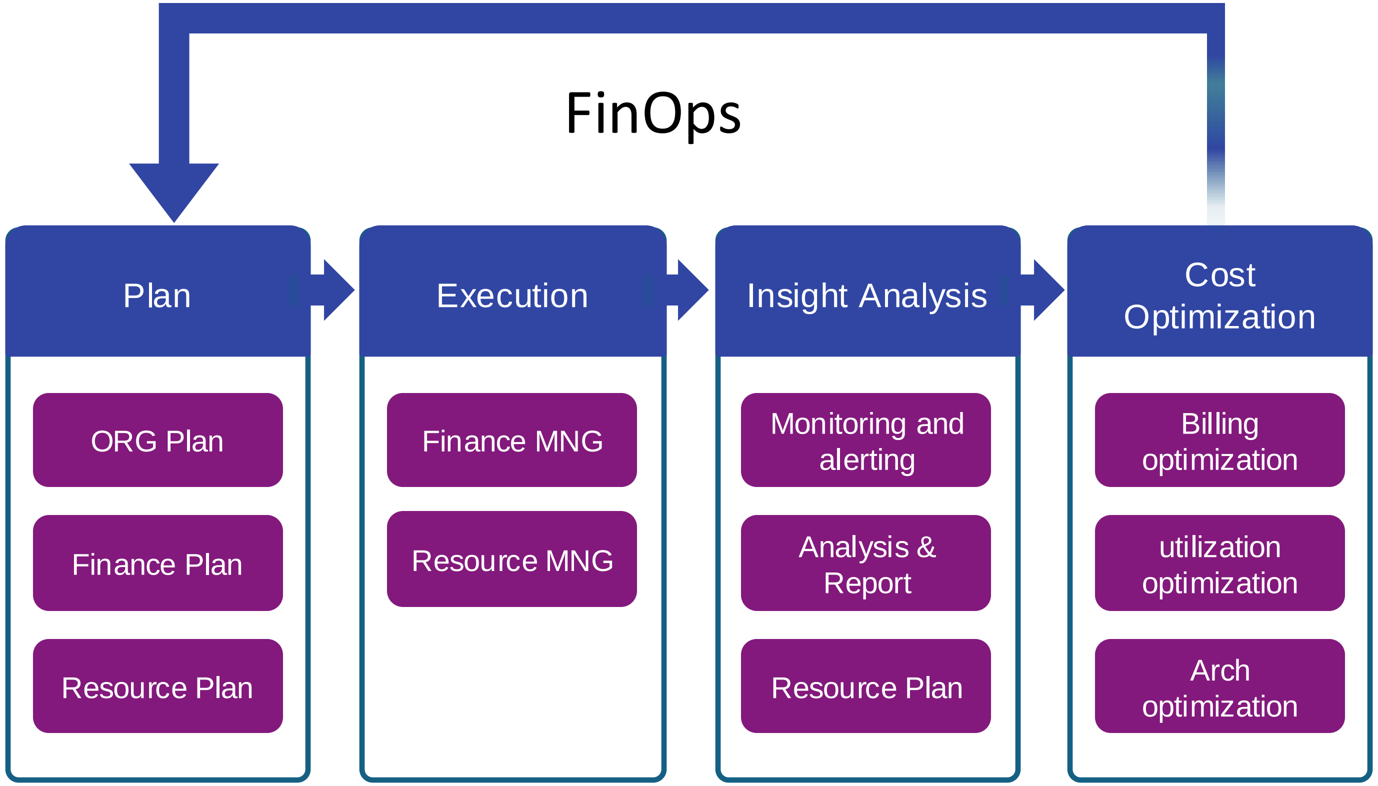


5. Management & Operation

- **Organization Management** 组织管理
  - Organization mapping with namespace and privilege
  - Tenant isolation & resource quota

- **Operation Management** 运营管理
  - Org Billing
  - Cost Management – FinOps

5. Mgmt - FinOps Philosophy

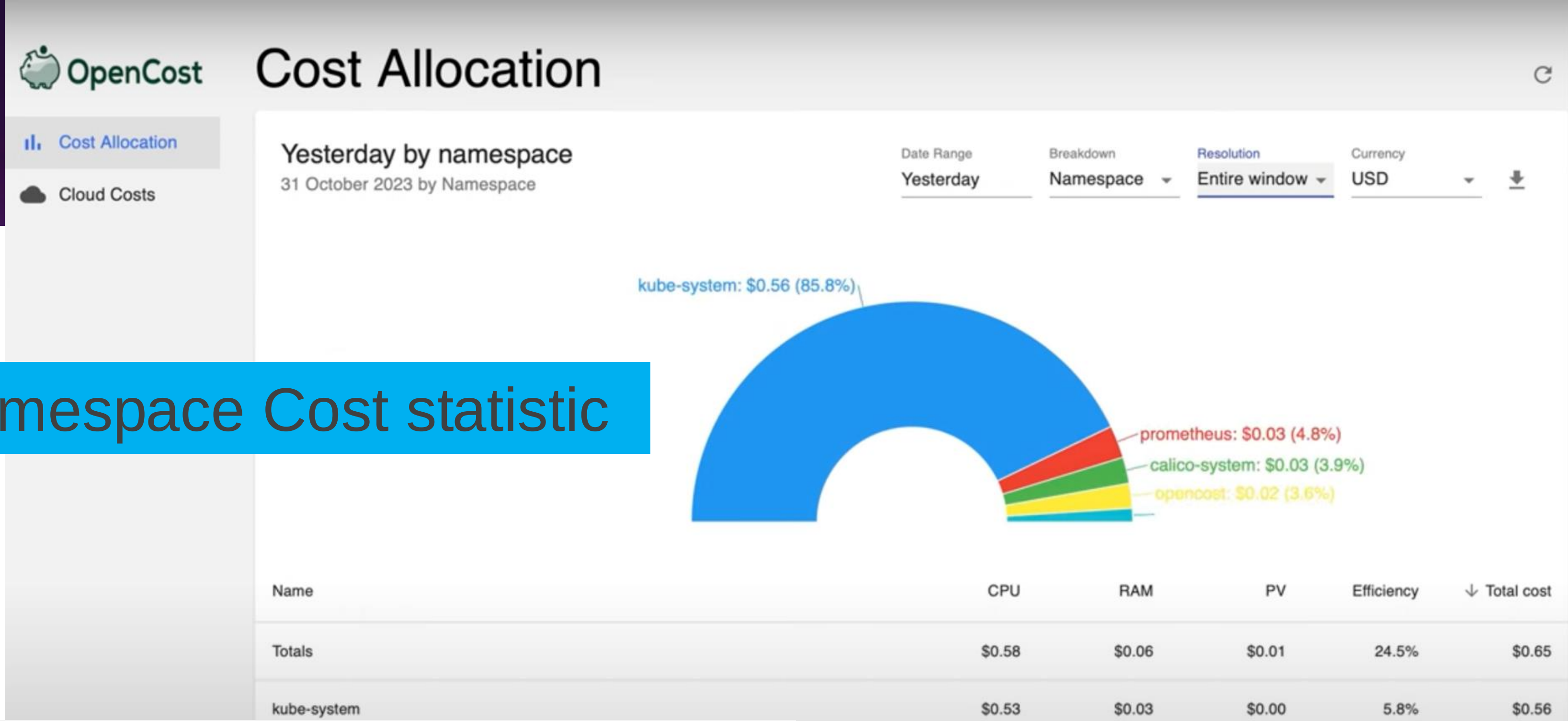


- Measure the return on investment (ROI)
- Choose appropriate resources and billing methods
- Introduce elasticity mechanisms for application workloads
- Continuously monitor and optimize

5. Mgmt - OpenCost

FinOps measurement

https://github.com/opencost



Cost Allocation

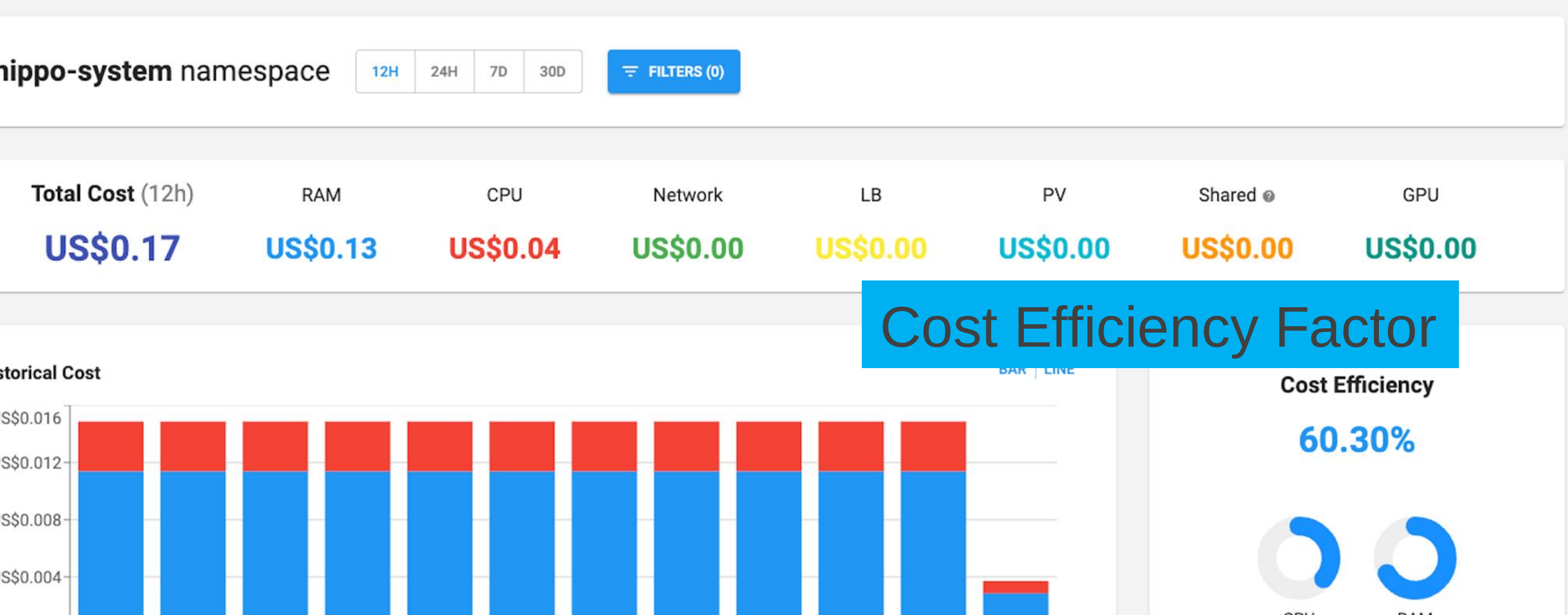
Yesterday by namespace

31 October 2023 by Namespace

Date Range: Yesterday
Breakdown: Namespace
Resolution: Entire window
Currency: USD

Name	CPU	RAM	PV	Efficiency	↓ Total cost
Totals	$0.58	$0.06	$0.01	24.5%	$0.65
kube-system	$0.53	$0.03	$0.00	5.8%	$0.56
	$0.00	$0.02	$0.01	Inf%	$0.03
	$0.02	$0.01	$0.00	Inf%	$0.03
	$0.02	$0.00	$0.00	12.3%	$0.02
	$0.01	$0.00	$0.00	Inf%	$0.01

Namespace Cost statistic



hippo-system namespace

Total Cost (12h)	RAM	CPU	Network	LB	PV	Shared	GPU
US$0.17	US$0.13	US$0.04	US$0.00	US$0.00	US$0.00	US$0.00	US$0.00

storical Cost

Cost Efficiency

60.30%

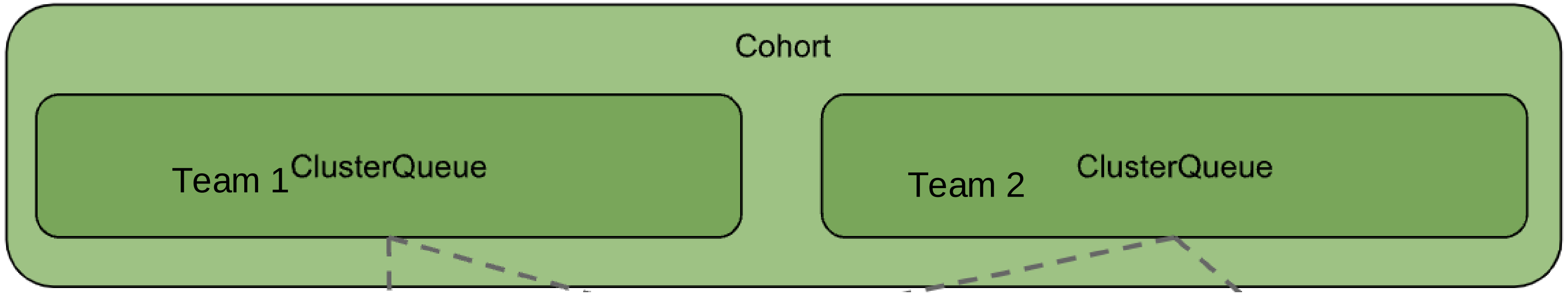
Cost Efficiency Factor

Bottomline rule for our team, e.g.:
Cost Efficiency >=40%
Mem Efficiency <=130%

5. Mgmt - Tenant

1. Namespace – `ResourceQuota`
2. Queue Quota – Kue’s `ClusterQueue` nominalQuota
3. “Cohort” – flexible quota between teams

- Borrow / Lend quota to others
- Reclaim/Preempt from others

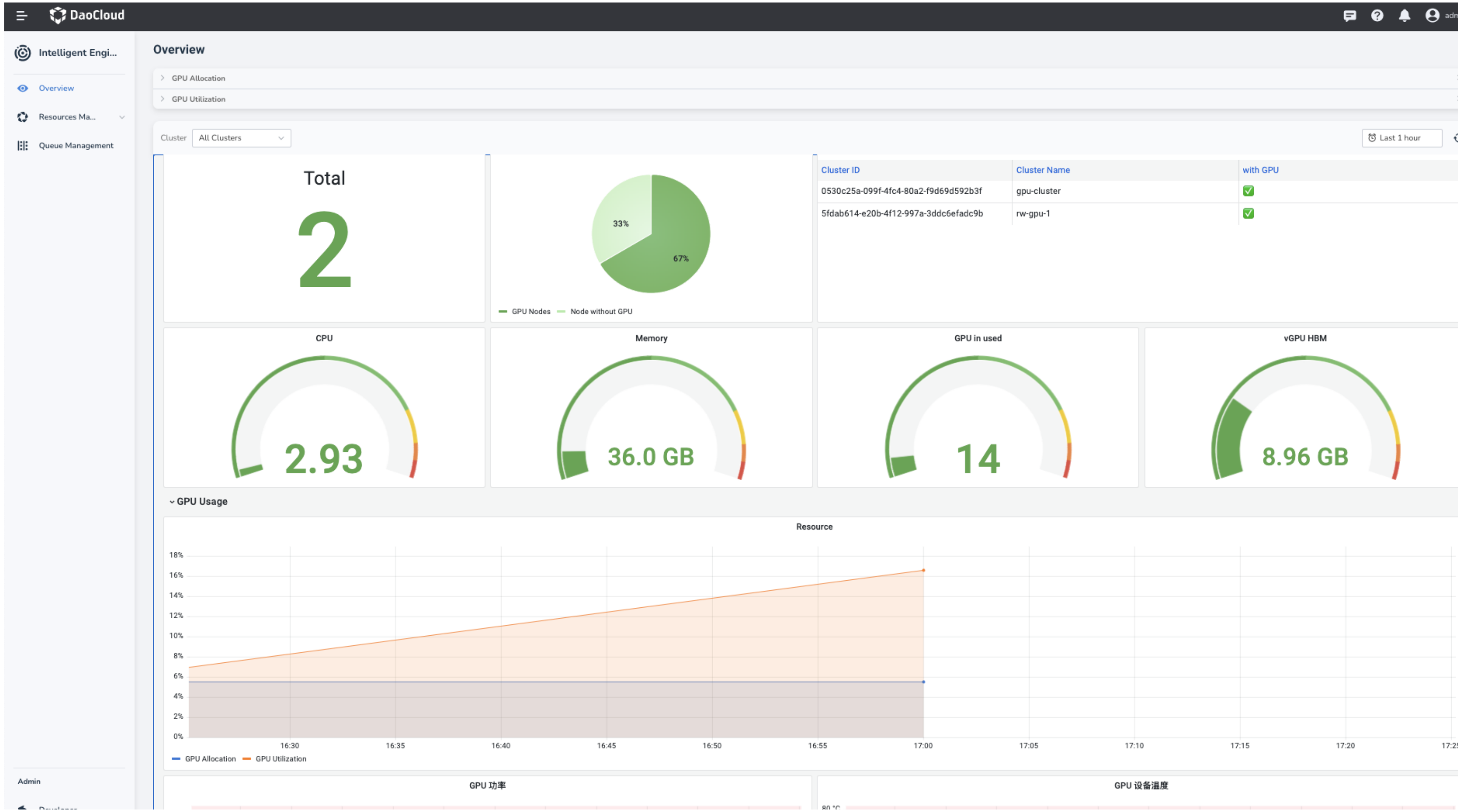


https://kueue.sigs.K8s.io/docs/concepts/cluster_queue/#cohort

5. Mgmt – Role base view

Separate **Operator** & Scientist

- **Operators** focus on :
  - Inventory
  - Resource Utilization
  - Tenant quota usage
  - Alerts / Monitor



Overview

GPU Allocation

GPU Utilization

Cluster: All Clusters

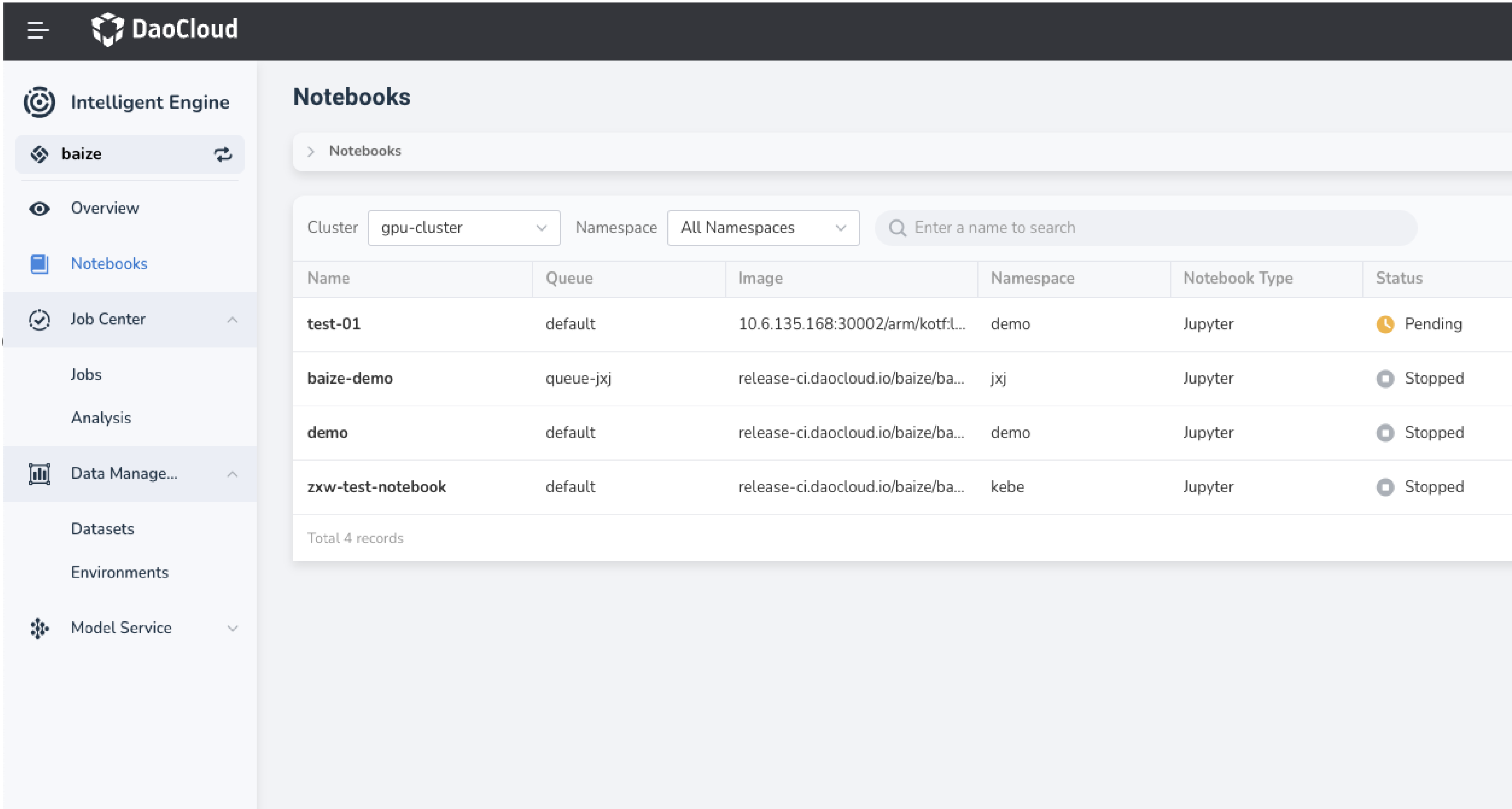
Cluster ID	Cluster Name	with GPU
0530c25a-099f-4fc4-80a2-f9d69d592b3f	gpu-cluster	✓
5fdab614-e20b-4f12-997a-3ddc6efadc9b	rw-gpu-1	✓

5. Mgmt – Role base view

Separate Operator & Scientist

Pros: Privilege & user-friendliness

- **Scientist** focus on :
  - Notebook
  - Training jobs
  - Dataset
  - Tenant quota total/usa



Notebooks

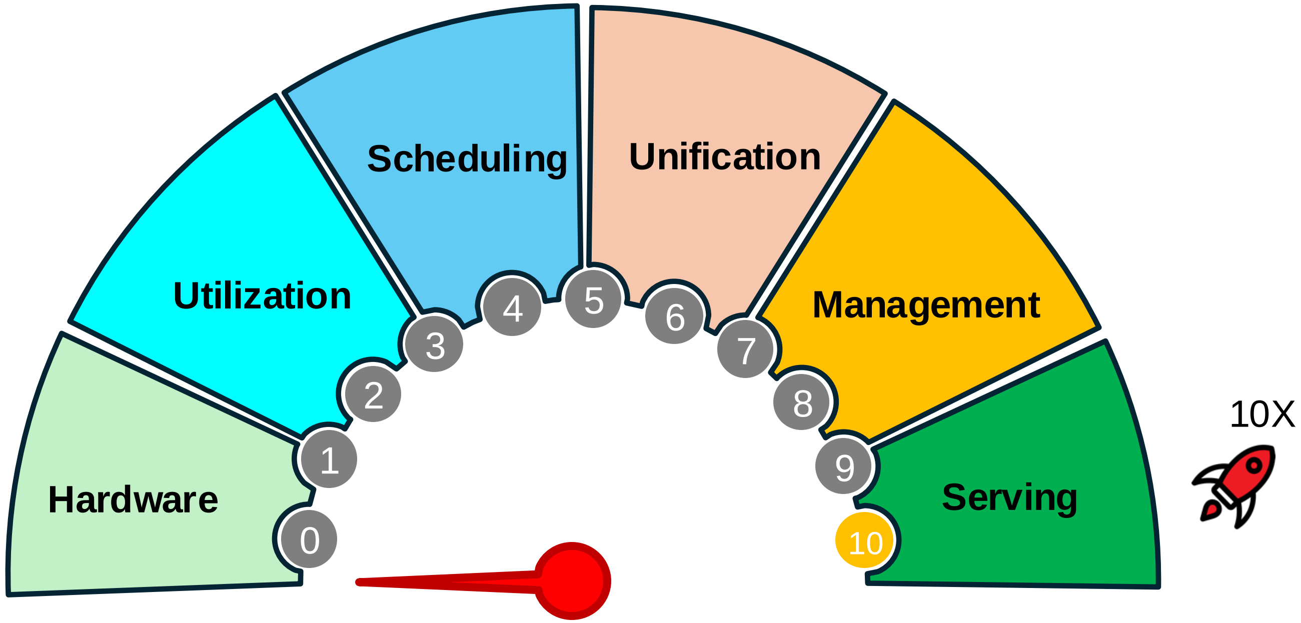
> Notebooks

Cluster: gpu-cluster
Namespace: All Namespaces
Enter a name to search

Name	Queue	Image	Namespace	Notebook Type	Status
test-01	default	10.6.135.168:30002/arm/kotf:l...	demo	Jupyter	Pending
baize-demo	queue-jxj	release-ci.daocloud.io/baize/ba...	jxj	Jupyter	Stopped
demo	default	release-ci.daocloud.io/baize/ba...	demo	Jupyter	Stopped
zxw-test-notebook	default	release-ci.daocloud.io/baize/ba...	kebe	Jupyter	Stopped

Total 4 records

6. Boost Model Serving !



6. Model Serving

Paints of Efficiency

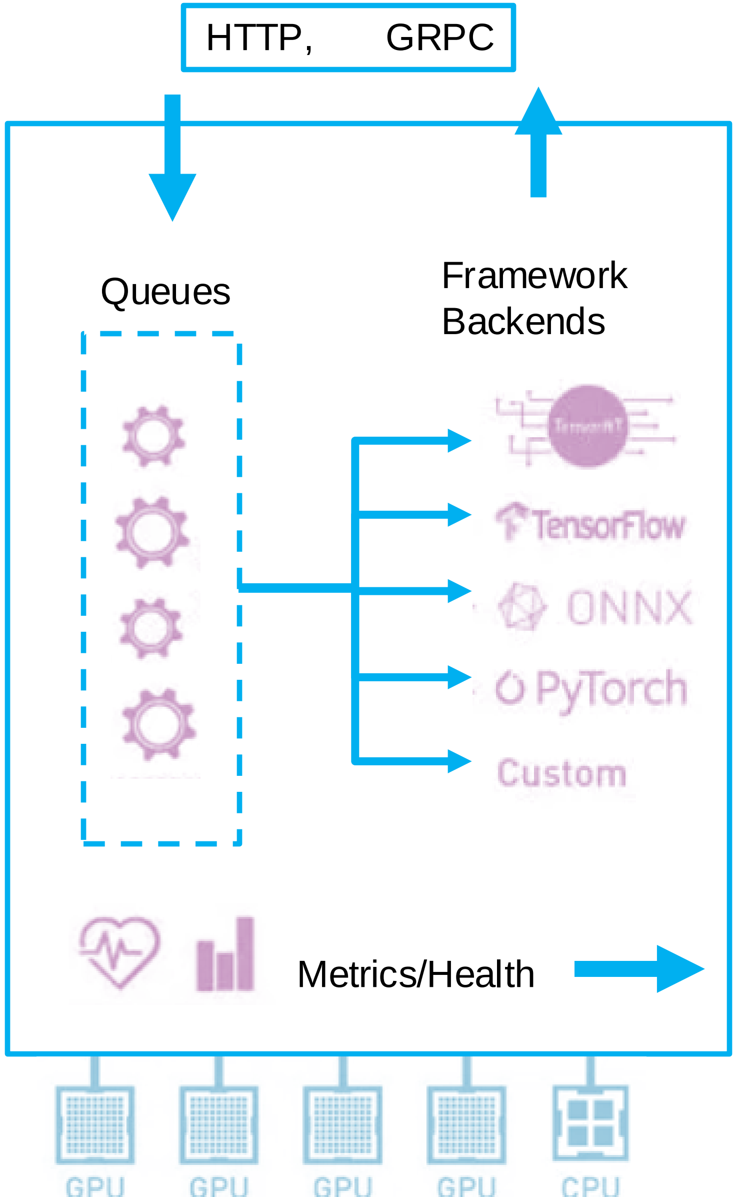
- **Complex** to deploy LLM model:
  - Need to Download model ->write transformer/llama code -> python env/run
- Slow model **download** from time to time
  - How to pack and reuse/cache the model in cluster
- Models **occupy** HBM but no request:
  - Knative can scale-to/from-zero , but somehow heavy (istio required)

6. Model Serving – Triton

- Triton
  - Handy: docker run -v ${model-path} nvcr.io/nvidia/tritonserver
  - All-in-one : serving / api-gateway / metrics / scheduling / batching
  - Wide range of backend supported :
    - vLLM、TensorFlow、PyTorch、
    - ONNX、OpenVINO、TensorRT..
  - Cons： lack of OpenAI format API

Our Practice in K8S : Triton-operator

```
apiVersion: serving.baize.io/v1alpha1
kind: Inference
spec:
  framework:
    triton:
     backend: vllm
  models:
  - modelPath: model/mnist_cnn/1/model.pt
     config: …
```



6. Model Serving – OCI Volume

```
apiVersion: serving.baize.io/v1alpha1
kind: Inference
spec:
  framework:
    triton:
     backend: vllm
  models:
  -  modelPath: model/mnist_cnn/1/model.pt
     config: …
```

OCI Artifacts Volume $^{\text{alpha}}$ in Kubernetes 1.31

```
spec:
  containers:
    - name: test
      image: nvcr.io/nvidia/tritonserver
      volumeMounts:
        - name: volume
          mountPath: /model
  volumes:
    - name: volume
      image:
        reference: quay.io/crio/artifact:v1
```

After 10 years long last, finally it comes

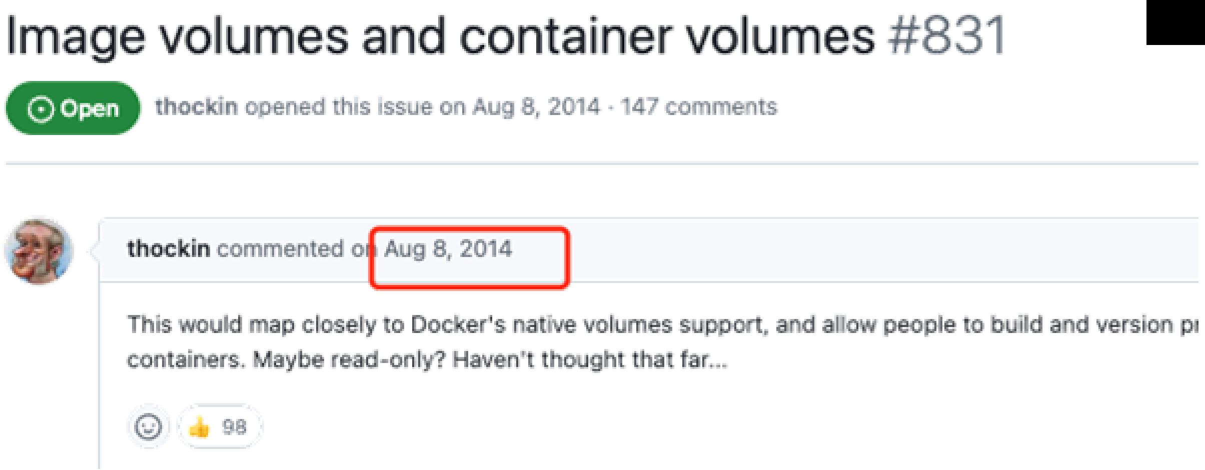


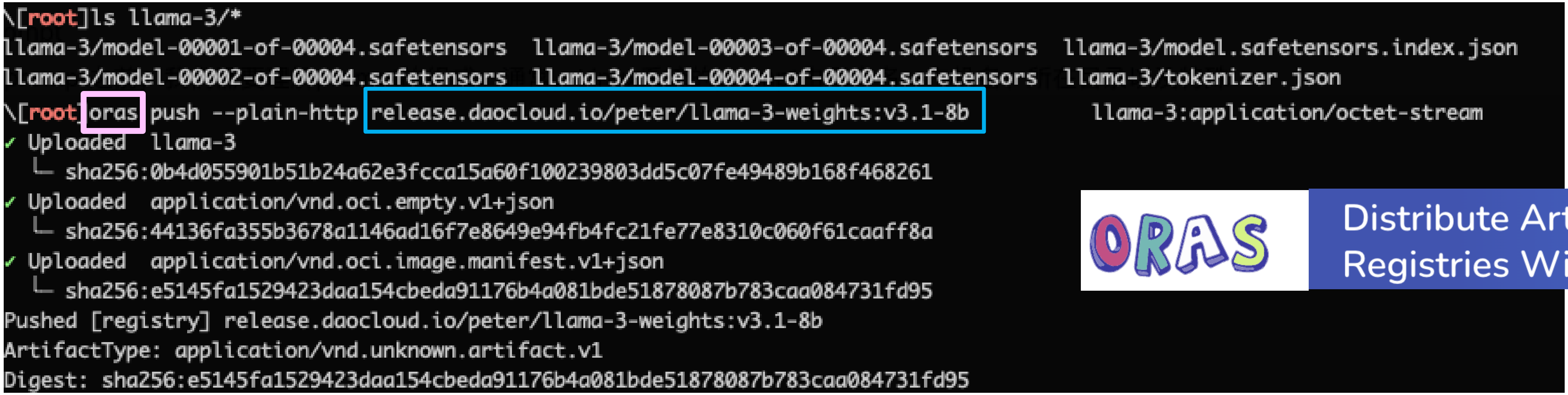
Image volumes and container volumes #831

Open thockin opened this issue on Aug 8, 2014 · 147 comments

thockin commented on Aug 8, 2014

This would map closely to Docker's native volumes support, and allow people to build and version pr
containers. Maybe read-only? Haven't thought that far...

6. Model Serving - ORAS

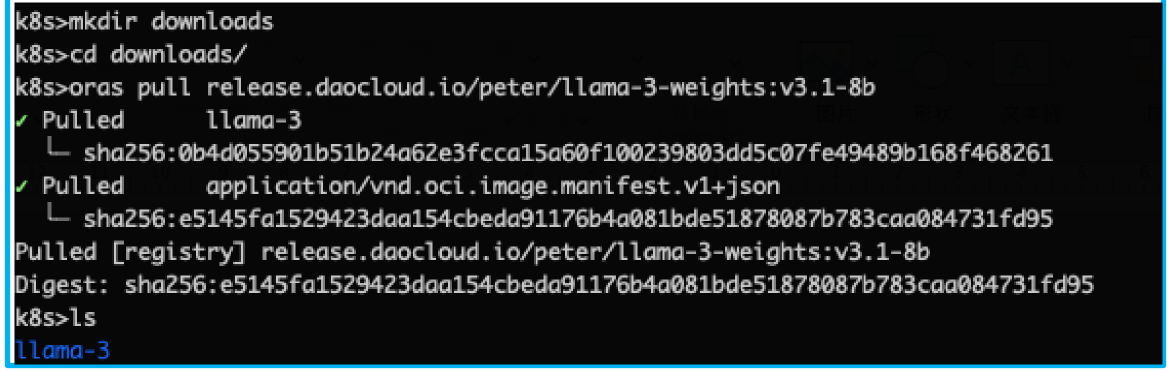


```
\[root]ls llama-3/*
llama-3/model-00001-of-00004.safetensors  llama-3/model-00003-of-00004.safetensors  llama-3/model.safetensors.index.json
llama-3/model-00002-of-00004.safetensors  llama-3/model-00004-of-00004.safetensors  llama-3/tokenizer.json
\[root] oras push --plain-http release.daocloud.io/peter/llama-3-weights:v3.1-8b        llama-3:application/octet-stream
✓ Uploaded  llama-3
  └ sha256:0b4d055901b51b24a62e3fcca15a60f100239803dd5c07fe49489b168f468261
✓ Uploaded  application/vnd.oci.empty.v1+json
  └ sha256:44136fa355b3678a1146ad16f7e8649e94fb4fc21fe77e8310c060f61caaff8a
✓ Uploaded  application/vnd.oci.image.manifest.v1+json
  └ sha256:e5145fa1529423daa154cbeda91176b4a081bde51878087b783caa084731fd95
Pushed [registry] release.daocloud.io/peter/llama-3-weights:v3.1-8b
ArtifactType: application/vnd.unknown.artifact.v1
Digest: sha256:e5145fa1529423daa154cbeda91176b4a081bde51878087b783caa084731fd95
```

ORAS Distribute Artifacts Across OCI Registries With Ease

Previous

1. `oras pull` to pull/extract artifacts into a local disk path
2. Mount the local path to the pod



```
k8s>mkdir downloads
k8s>cd downloads/
k8s>oras pull release.daocloud.io/peter/llama-3-weights:v3.1-8b
✓ Pulled      llama-3
  └ sha256:0b4d055901b51b24a62e3fcca15a60f100239803dd5c07fe49489b168f468261
✓ Pulled      application/vnd.oci.image.manifest.v1+json
  └ sha256:e5145fa1529423daa154cbeda91176b4a081bde51878087b783caa084731fd95
Pulled [registry] release.daocloud.io/peter/llama-3-weights:v3.1-8b
Digest: sha256:e5145fa1529423daa154cbeda91176b4a081bde51878087b783caa084731fd95
k8s>ls
llama-3
```

Now - in Pod’s OCI volume

```
spec:
  containers:
  ....
  volumes:
    - name: volume
      image:
        reference: release.daocloud.io/peter/llama-3-weight:v3.1-8b
```

https://github.com/kubernetes/kubernetes/issues/125463

6. Model Serving – Ollama

Ollama - most simple LLM tool (backed by llama.cpp)

1. Basic Usage

```
# ollama run llama3.1
pulling manifest
pulling 8934d96d3f08... 100%  ▇▇▇▇ 3.8 GB
pulling 170370233dd5... 100%  ▇▇▇▇  4.1 GB
….
>>> How is the weather today?
```

`#ollama serve` → serve as open-AI format API server

2. Modelfile ☺

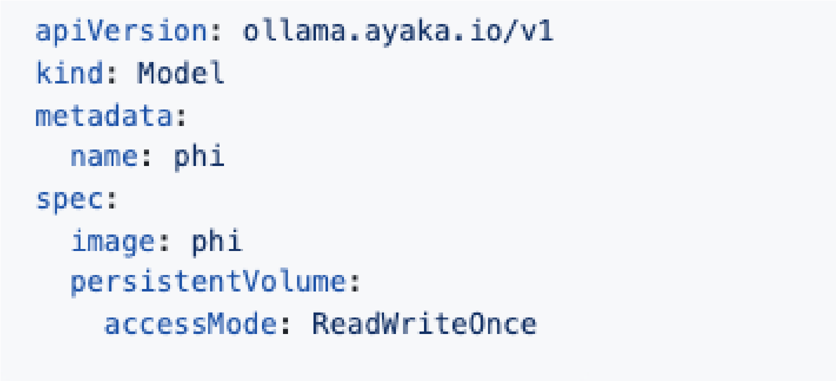
```
FROM llama3

PARAMETER temperature 1
PARAMETER num_ctx 4096
assistant SYSTEM $system-prompt
```

`#ollama create my-image  -f Modelfile`

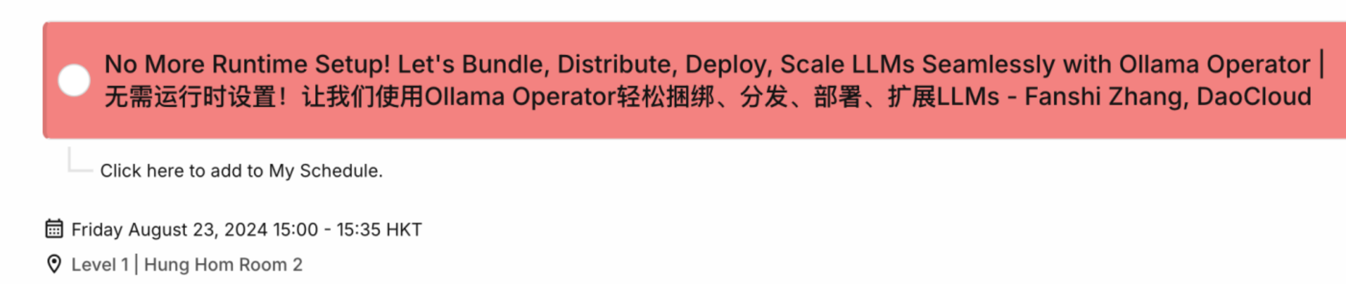
`#ollama run my-image`

3. Operator in K8s



```yaml
apiVersion: ollama.ayaka.io/v1
kind: Model
metadata:
  name: phi
spec:
  image: phi
  persistentVolume:
    accessMode: ReadWriteOnce
```

https://github.com/nekomeowww/ollama-operator



No More Runtime Setup! Let's Bundle, Distribute, Deploy, Scale LLMs Seamlessly with Ollama Operator | 无需运行时设置！让我们使用Ollama Operator轻松捆绑、分发、部署、扩展LLMs - Fanshi Zhang, DaoCloud

Click here to add to My Schedule.

Friday August 23, 2024 15:00 - 15:35 HKT

Level 1 | Hung Hom Room 2

6. Model Serving – Lingo

Lingo - LLM gateway

https://github.com/substratusai/lingo

Starred 103

- scale-to-zero & scale-from-zero
- Re-use same GPU for N models
- Lightweight , no dependency
- Queue, but no batch
- alternative to knative

6. Model Serving – Lingo

1.Serve
- MiniLM-L6
- Mistral-7b

```
# kubectl get deployment
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
mistral-7b-instruct-vllm    0/0    0            0           21d
minilm-l6-v2                0/0    0            0           85d
```

```
curl http://$IP:8080/v1/completions  -d '{"model": "text-embedding-ada-002", "prompt": ”…”..}'
```

2.miniLM scale from 0

```
# kubectl get deployment
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
mistral-7b-instruct-vllm    0/0    0            0           21d
minilm-l6-v2                1/1    1            1           85d
```

3.miniLM scale to 0

```
# kubectl get event

109s       …    Scaled up replica set minilm-l6-v2-5dd7bb4fc8 to 1 from 0
# a while later
15s        …  Scaled down replica set minilm-l6-v2-5dd7bb4fc8 to 0 from 1
```

```
curl http://$IP:8080/v1/completions  -d '{"model": "mistral-7b-instruct-v0.1", "prompt": ”…”..}'
```

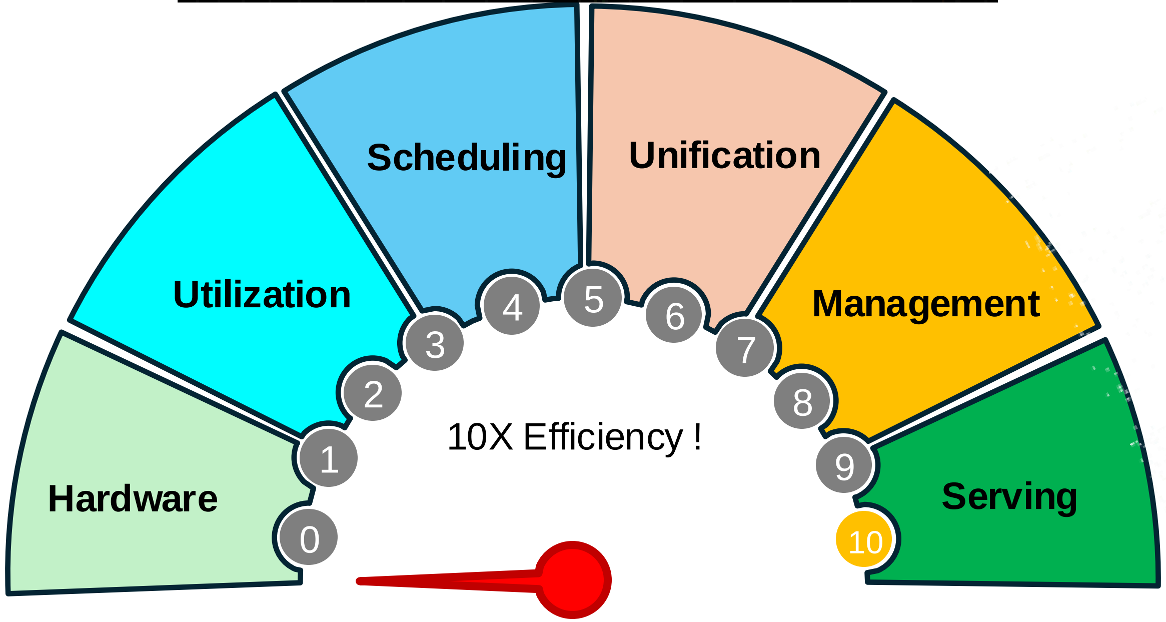
4.Mistral-7b wake up

```
# kubectl get deploy
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
mistral-7b-instruct-vllm    1/1    1            1           21d
minilm-l6-v2                0/0    0            0           85d
```

Kubectl annotate deploy minilm-l6-v2 lingo.substratus.ai/models: text-embedding-ada-002

Thanks

- d.run
- HAMi
- JuiceFS
- HwameiStor
- kubeFlow
- dcgmi-diag
- Kueue
- Volcano
- OpenCost
- ORAS
- Triton
- Ollama
- Lingo


Hardware
Utilization
Scheduling
Unification
Management
Serving
0
1
2
3
4
5
6
7
8
9
10
10X Efficiency !