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

Unlocking Heterogeneous AI Infrastructure K8s Cluster

Leveraging the Power of HAMi

About us

MengXuan Li

Github @archlitchi

4Paradigm

Xiao Zhang software engineer

Github @wawa0210

DaoCloud

Chanllange 1: Requirement for Computing Power is Growing

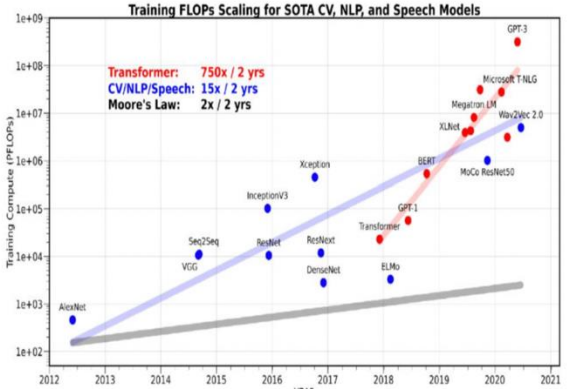


Figure 1: Training Flops trend

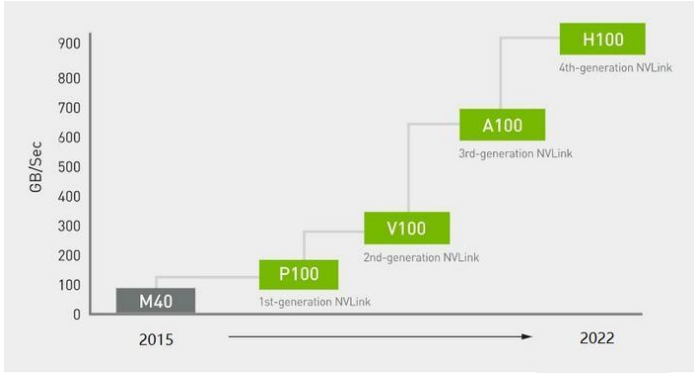
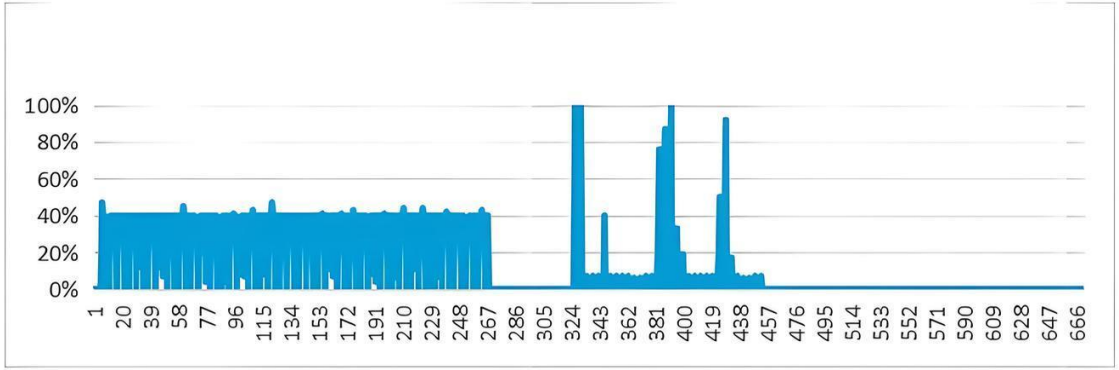


Figure 2: NVIDIA Flagship GPU for ML

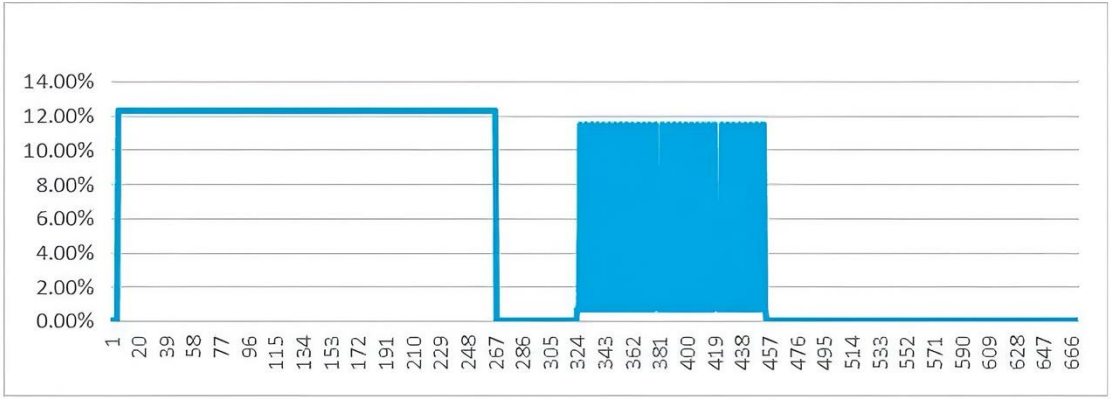
- AI technology has entered the stage of commercialization and requires more and more computing power.
- The demand for computing power for large langrage models can be quite exaggerated (375x/year)
- In order to match the trend of computing power growth, GPU manufacturers have released new GPUs rapidly, with more powerful computing power, and higher price.

Chanllange 2: Low Resource Utilization on GPU



A typical GPU utilization in GPU task in kubernetes:

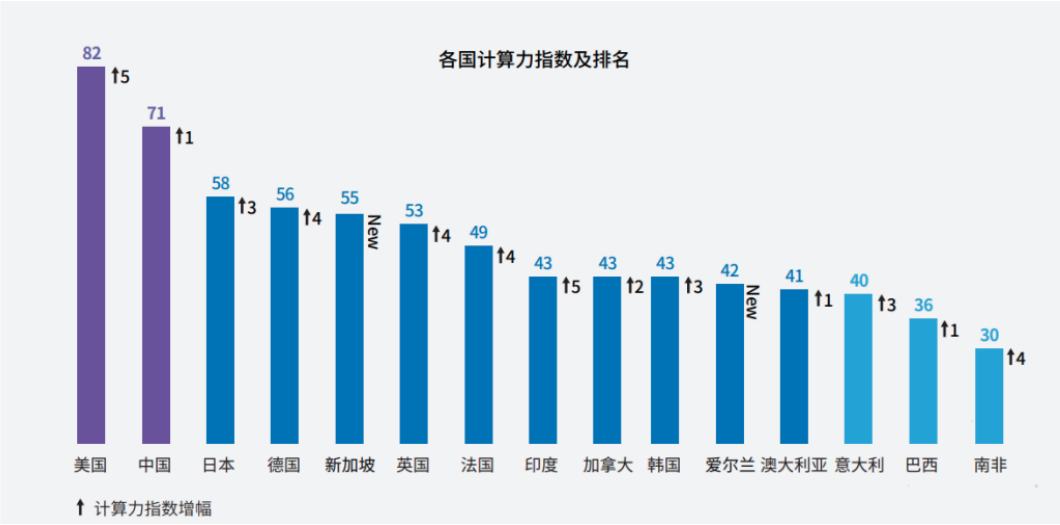
- Core utilization can be 0 for
- In order to match the trend of computing power growth, GPU manufacturers have released new GPUs rapidly, with more powerful computing power, and higher price.

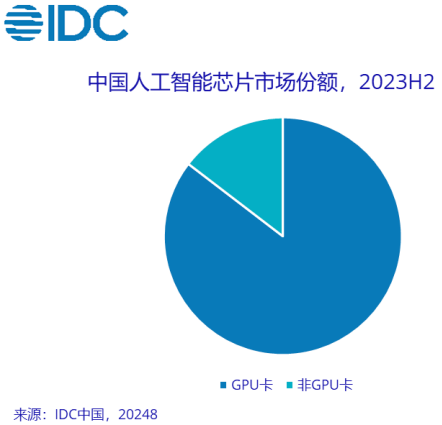


Two factors lead to low utilization of GPU devices in k8s clusters:

- GPU resources can only be applied by container in an exclusive manner
- In order to match the trend of computing power growth, GPU manufacturers have released new GPUs rapidly, with more powerful computing power, and higher price.

Chanllange 3: The demand for heterogeneous AI devices continues to grow





Shipments exceeded 1.4 million

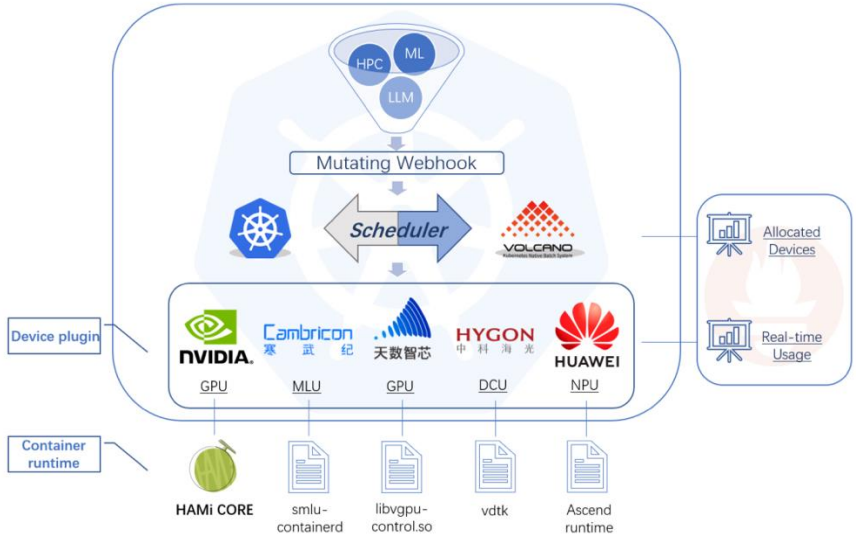
Nvidia for 85%, Huawei 10%, Baidu 2%, and others 2%

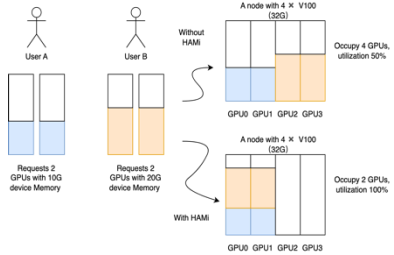
In addition to Nvidia GPUs, there are also Cambricon, Hygon, iluvatar, Huawei Ascend AI devices.

There are more and more AI smart devices. Unified orchestration scheduling and management will be very urgent.

What is HAMi

Heterogeneous AI Computing Virtualization Middleware (HAMi), is an "all-in-one" tool designed to manage Heterogeneous AI Computing Devices in Kubernetes cluster.



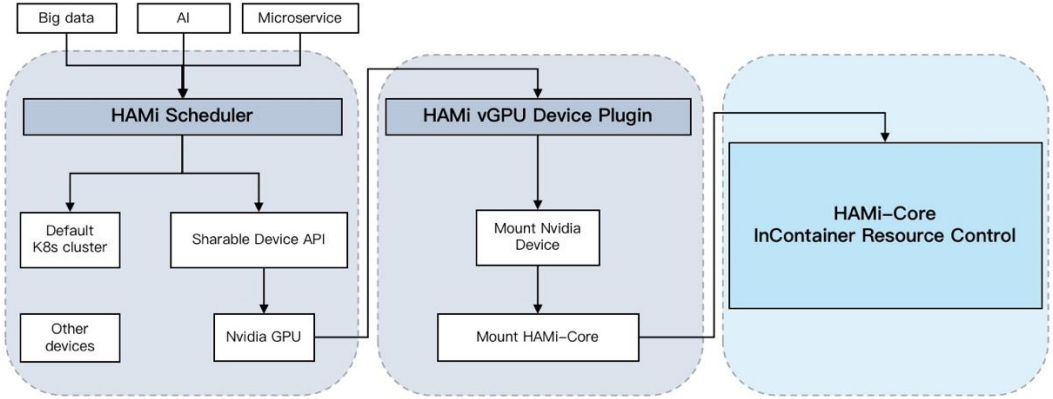


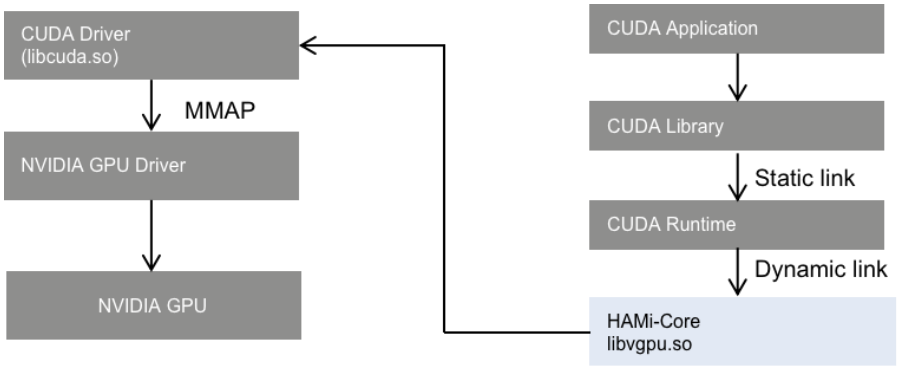
- A K8s cluster has consistent management of multiple heterogeneous AI device nodes(**NVIDIA, Cambricon, Hygon, iluvatar, Huawei Ascend**).
- **Device sharing** (or device virtualization) on Kubernetes.
- Scenarios where pods need to be allocated with specific device memory.
- Need to **balance GPU usage** in a cluster with multiple GPU nodes.
- Low utilization of device memory and computing units, such as running 10 TensorFlow servings on one GPU.
- Situations that require a large number of small GPUs.
- AI devices observability.

HAMi Key Features

- Support multiple AI devices, Provide unified scheduling capabilities(NVIDIA, Cambricon, Hygon, iluvatar, Huawei Ascend)
- Permits partial device allocation by specifying device core usage
- Device sharing
- Hard Resource Isolation inside container
- Device Type/UUID Specification
- Task priority
- Flexible Schecdule policy binpack & spread

How HAMi gpu share does it work?





HAMi-Core uses symbolic hijacking to operate inside containers

Prerequisites:

- Nvidia driver version >= 440
- CUDA version >= 10.2

Features:

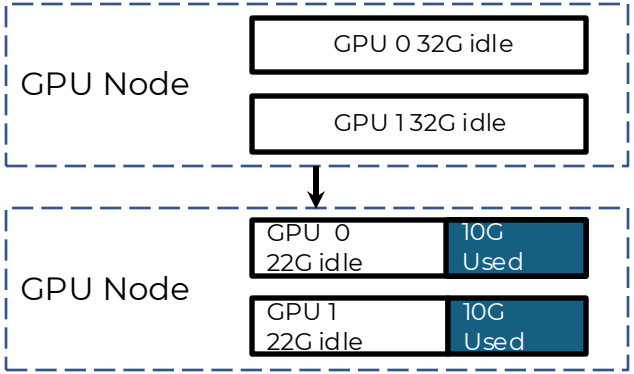
- Device Memory isolation
- Core utilization limitation
- Fault isolation
- Transparent to GPU tasks

Device sharing——NVIDIA

Parameter Description:

- nvidia.com/gpu: Specifies the number of visible GPUs in the container.
- nvidia.com/gpumem: Specifies the memory size to use for each GPU. If not set, the default is to use all available GPU memory.
- nvidia.com/gpucores: Specify the percentage used for each GPU.

```
$ cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod12
spec:
  containers:
    - name: ubuntu-container
      image: ubuntu:18.04
      command: ["bash", "-c", "sleep 86400"]
      resources:
        limits:
          nvidia.com/gpu: 2 # requesting 2 vGPUs
          nvidia.com/gpumem: 10240
          nvidia.com/gpucores: 30
```




```
root@gpu-demo-6b6b88b75b-x9tjb:~# nvidia-smi
[HAMI-core Msg(35:140111864956736:libvgpu.c:836)]: Initializing.....
Thu Apr 18 06:22:54 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.12             Driver Version: 535.104.12   CUDA Version: 12.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=================================+======================+======================|
|   0  NVIDIA A800 80GB PCIe          On  | 00000000:13:00.0 Off |                    0 |
| N/A   36C    P0              81W / 300W |      0MiB / 10240MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A800 80GB PCIe          On  | 00000000:1C:00.0 Off |                    0 |
| N/A   39C    P0              82W / 300W |      0MiB / 10240MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
[HAMI-core Msg(35:140111864956736:multiprocess_memory_limit.c:434)]: Calling exit handler 35
root@gpu-demo-6b6b88b75b-x9tjb:~#
```

Device sharing——Huawei Ascend 910 NPU

Parameter Description:

- huawei.com/ascend910: Specifies the number of visible Ascend 910s in the container.
- huawei.com/ascend910-memory: Specifies the memory size to use for each Ascend 910s. If not set, the default is to use all available device memory.

```
$ cat <<EOF | kubectl apply -f -

spec:
  containers:
    - ...
      resources:
        limits:
          huawei.com/Ascend910: 1
          huawei.com/Ascend910-memory: 16384
```

host

```
+-------------------------------------------------------------------------------------------+
| npu-smi 24.1.rc1                 Version: 24.1.rc1                                        |
+-------------------------+-----------------+-------------------------------------------------+
| NPU   Name              | Health          | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip                    | Bus-Id          | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+=========================+=================+=================================================+
| 0     910B3             | OK              | 95.9        39                0    / 0          |
| 0                       | 0000:C1:00.0    | 0           0    / 0          24082/ 65536         |
+-------------------------+-----------------+-------------------------------------------------+
```

container

```
+-------------------------------------------------------------------------------------------+
| npu-smi 24.1.rc1                 Version: 24.1.rc1                                        |
+-------------------------+-----------------+-------------------------------------------------+
| NPU   Name              | Health          | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip                    | Bus-Id          | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+=========================+=================+=================================================+
| 1     910B3vir05_1c_16g | OK              | 89.9        25                0    / 0          |
| 0                       | 0000:C2:00.4    | 0           0    / 0          1235 / 16384         |
+=========================+=================+=================================================+
+-------------------------+-----------------+-------------------------------------------------+
| NPU     Chip            | Process id      | Process name            | Process memory(MB)    |
+=========================+=================+=================================================+
| No running processes found in NPU 1                                                       |
+=========================+=================+=================================================+
```

Device sharing——Iluvatar GPU

Parameter Description:

- iluvatar.ai/gpu: Specifies the number of visible iluvatar GPUs in the container.
- iluvatar.ai/vcuda-memory: Specifies the memory size to use for each iluvatar GPU. If not set, the default is to use all available device memory.
- iluvatar.ai/vcuda-core: Specify the percentage used for each Iluvatar GPU.

```
$ cat <<EOF | kubectl apply -f -

spec:
  containers:
    - ...
      resources:
        limits:
          iluvatar.ai/gpu: 1
          iluvatar.ai/vcuda-core: 50
          iluvatar.ai/vcuda-memory: 64 #each unit
represents 256M device memory
```

host

```
+-----------------------------------------------------------------------------+
|  IX-ML: 4.0.0       Driver Version: 4.0.0       CUDA Version: 10.2          |
|-------------------------------+----------------------+----------------------|
| GPU  Name                     | Bus-Id               | Clock-SM  Clock-Mem  |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage    | GPU-Util  Compute M. |
|===============================+======================+======================|
| 0    Iluvatar MR-V100         | 00000000:40:00.0     | 1500MHz   1600MHz    |
| 0%   44C   P0    41W / 150W   | 114MiB / 32768MiB    | 0%        Default    |
+-------------------------------+----------------------+----------------------+
```

container

```
[root@poddemo corex-3.2.0]# ixsmi
Timestamp    Tue Feb 20 10:38:32 2024
+-----------------------------------------------------------------------------+
|  IX-ML: 3.2.0.2     Driver Version: 3.2.0       CUDA Version: 10.2          |
|-------------------------------+----------------------+----------------------|
| GPU  Name                     | Bus-Id               | Clock-SM  Clock-Mem  |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage    | GPU-Util  Compute M. |
|===============================+======================+======================|
| 0    Iluvatar MR-V100         | 00000000:40:00.0     | 1500MHz   1600MHz    |
| 0%   43C   P0    40W / 150W   | 0MiB / 16384MiB      | 0%        Default    |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU        PID      Process name                                Usage(MiB) |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Device sharing——Cambricon

Parameter Description:

- cambricon.com/vmlu: Specifies the number of visible MLUs in the container.
- cambricon.com/mlu.smlu.vmemory: Specifies the memory size to use for each MLU. If not set, the default is to use all available MLU memory.
- nvidia.com/gpucores: Specify the percentage used for each MLU.

```
$ cat <<EOF | kubectl apply -f -

spec:
  containers:
    - ...
      resources:
        limits:
          cambricon.com/vmlu: 1 # requesting 1 vGPUs
        cambricon.com/mlu.smlu.vmemory: 20 #
request 20% device memory
        cambricon.com/mlu.smlu.vcore: 10 # request
10% of compute cores
```

host

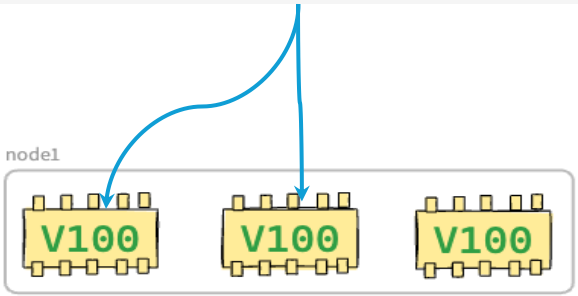
```
+-----------------------------------------------------------------------------+
| CNMON v5.10.29                                              Driver v5.10.29 |
+-----------------------------+----------------------+----------------------+
| Card  VF  Name       Firmware |              Bus-Id | Util        Ecc-Error |
| Fan   Temp      Pwr:Usage/Cap |       Memory-Usage | Mode     Compute-Mode |
|=============================+======================+======================|
| 0     /   MLU370-X4    v1.1.6 |       0000:B4:00.0 | 0%                 N/A |
|  0%   35C         28 W/ 150 W |    0 MiB/ 23308 MiB | FULL           Default |
+-----------------------------+----------------------+----------------------+
```

container

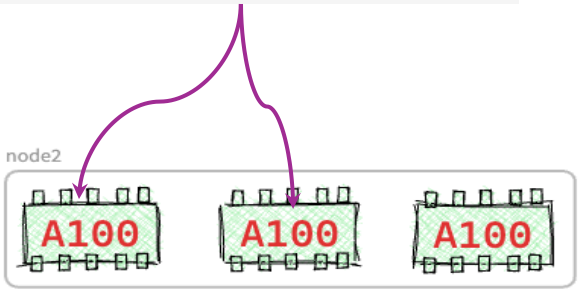
```
root@binpack-1-5bff578c96-bbnwk:/# cnmon
Fri Apr 19 10:20:44 2024
+-----------------------------------------------------------------------------+
| CNMON v5.10.29                                              Driver v5.10.29 |
+-----------------------------+----------------------+----------------------+
| Card  VF  Name       Firmware |              Bus-Id | Util        Ecc-Error |
| Fan   Temp      Pwr:Usage/Cap |       Memory-Usage | Mode     Compute-Mode |
|=============================+======================+======================|
| 0     /   MLU370-X4    v1.1.6 |       0000:B5:00.0 | N/A                N/A |
|  0%   34C         27 W/ 150 W |                N/A | sMLU           Default |
+-----------------------------+----------------------+----------------------+



+-----------------------------------------------------------------------------+
| sMLU Info:                                                                  |
+-----------------------------+----------------------+----------------------+
| Card    Profile  Instance   |                BDF | Usage                |
|         ID       ID         |       Memory-Usage |                  MLU |
+-----------------------------+----------------------+----------------------+
| 0       0        1          |       0000:B5:00.0 |                   0% |
|                             |    0 MiB/ 4660 MiB |                      |
+-----------------------------+----------------------+----------------------+



+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  Card  MI  PID     Command Line                         MLU Memory Usage    |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Device Specification Schedule


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod2
  annotations:
    nvidia.com/nouse-gputype: "A100"
spec:
  containers:
    - name: ubuntu-container
      image: ubuntu:18.04
      command: ["bash", "-c", "sleep 86400"]
      resources:
        limits:
          nvidia.com/gpu: 2
```




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
  annotations:
    nvidia.com/use-gputype: "A100"
spec:
  containers:
    - name: ubuntu-container
      image: ubuntu:18.04
      command: ["bash", "-c", "sleep 86400"]
      resources:
        limits:
          nvidia.com/gpu: 2
```



HAMi supports appoint tasks to certain types of GPU, or avoid certain type of GPU, as the figure shows

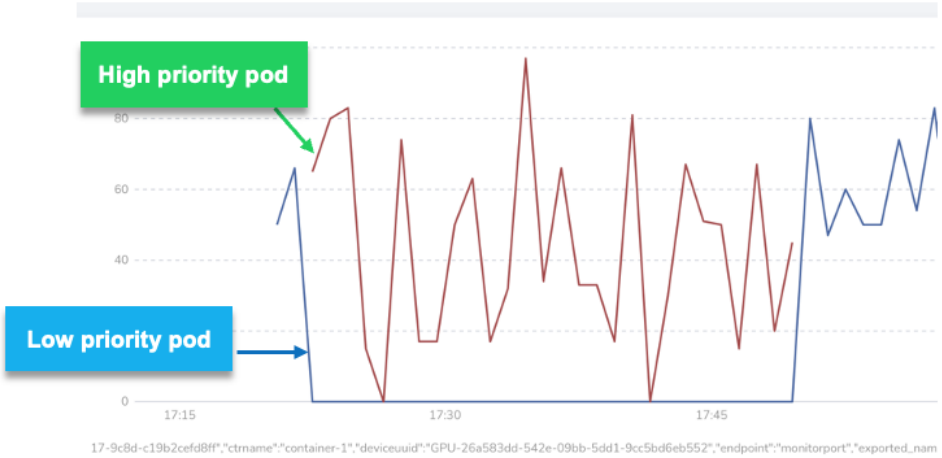
- Set whitelist by assigning `nvidia.com/use-gputype` in pod annotations.
- Set blacklist by assigning `nvidia.com/nouse-gputype` in pod annotations.

Task priority

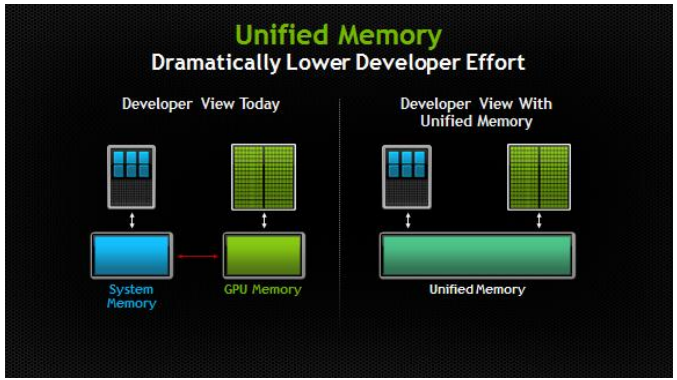

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
spec:
  containers:
    - name: ubuntu-container
      image: ubuntu:18.04
      command: ["bash", "-c", "sleep 86400"]
      env:
      # vgpu task priority 0 for high and 1 for low
        - name: CUDA_TASK_PRIORITY'
          value: '0'
      resources:
        limits:
          nvidia.com/gpu: 1
          nvidia.com/gpumem: 3000
          nvidia.com/gpucores: 30
```

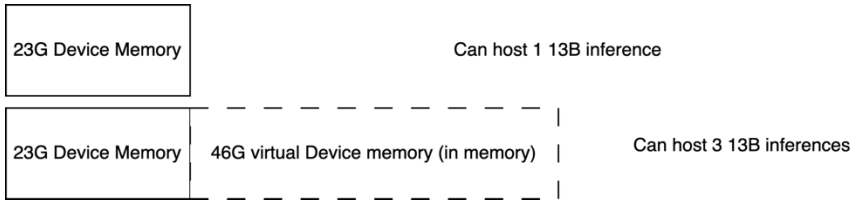
HAMi supports GPU utilization preemption：

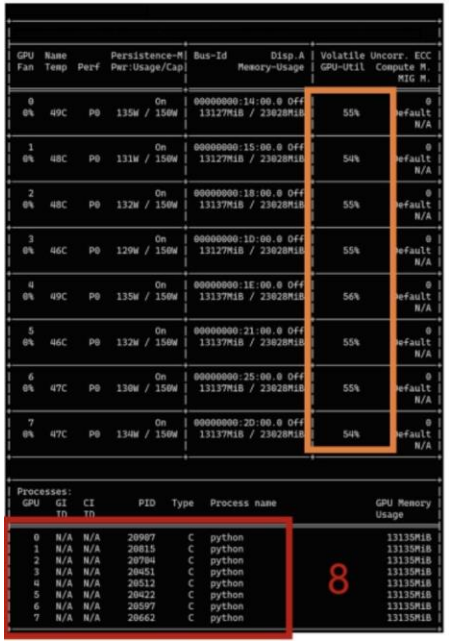
- Set the task priority in `container.env["CUDA_TASK_PRIORITY"]`, 0 for high, 1 for low
- low priority task will pause while high priority task submitting gpu kernel.
- Transparent to GPU task.



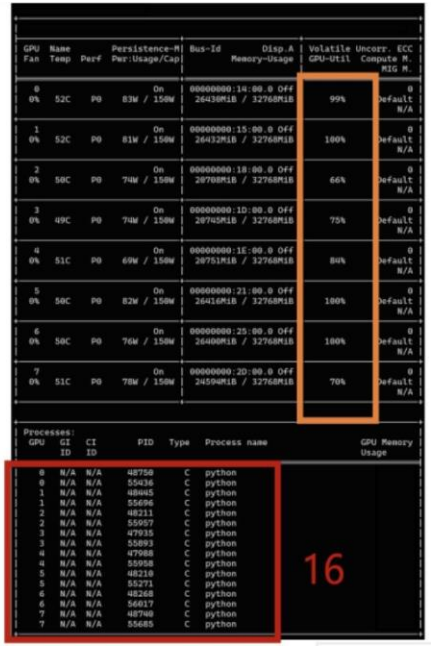
over-allocation mechanism







Before

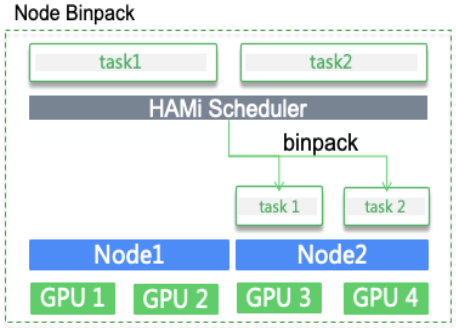


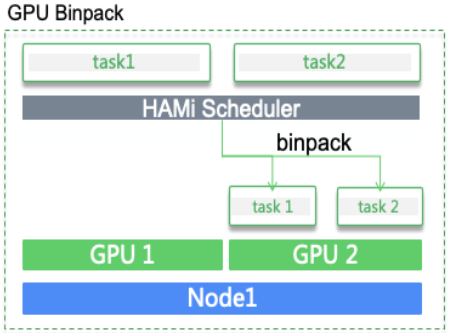
After

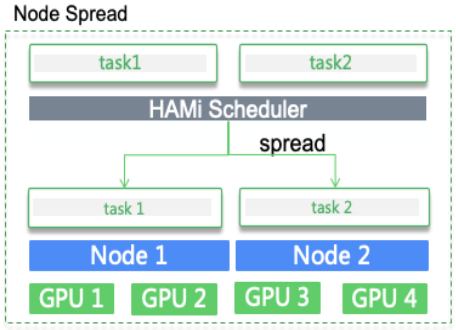
Through UMI, GPU and system memory are used together (CUDA allocated memory will be recalled to system memory if it is not used for a long time)

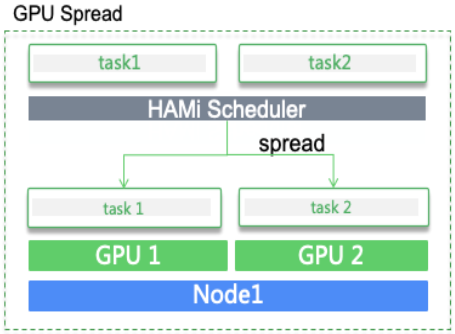
Typical use case: Hybrid Inference and time imbalance

binpack & spread scheduler polices







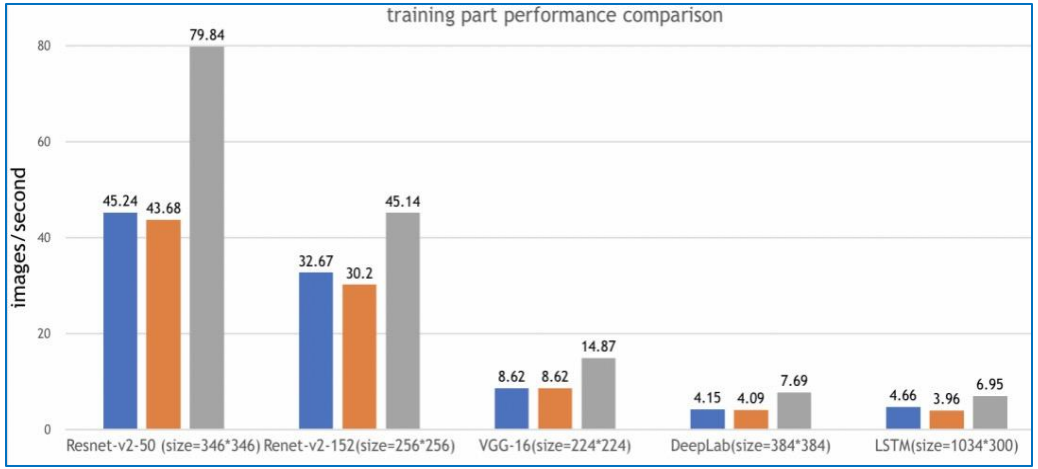


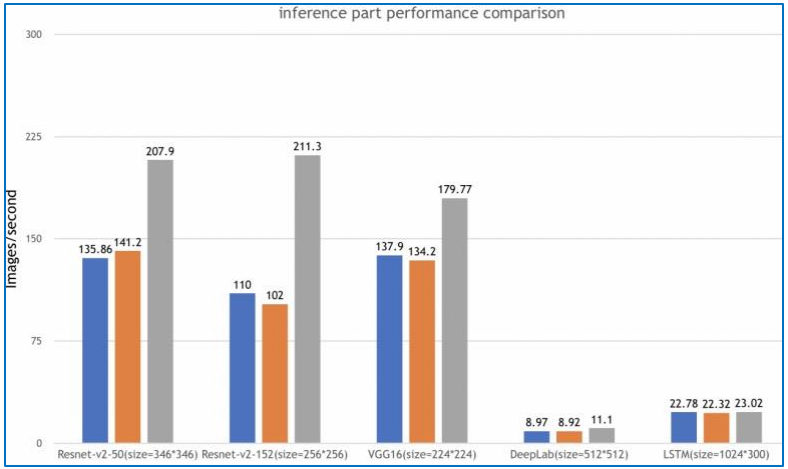
HAMi supports binpack and spread schedule policy on GPU and node level. As tge fugure indicates:

- The binpack policy uses fragment minimization to schedule and assign tasks to nodes and devices where existing tasks are running
- The spread policy assign tasks to the devices and nodes with the fewest number of sharing tasks to ensure the performance of tasks
- A pod can override default schedule policy by assigning schedule-policy in annotations, as the following figure shows:

```
metadata:
  name: gpu-pod
  annotations:
    hami.io/node-scheduler-policy: "spread"
    hami.io/gpu-scheduler-policy: "binpack"
```

HAMi Benchmark





Test Instance :

- nvidia-device-plugin: 1 instance/GPU
- volcano-vGPU : 1 instance/GPU
- volcano-vGPU: 2 instances/GPU

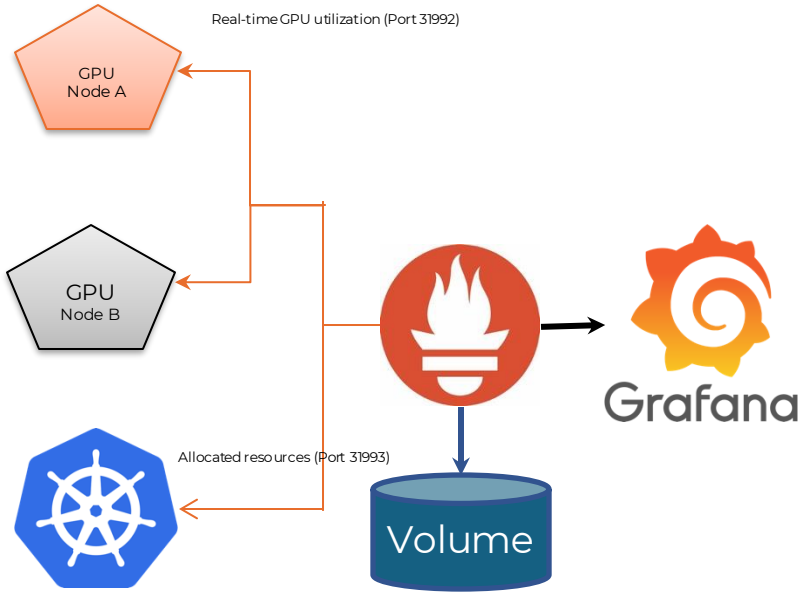
Test Environment :

- GPU Type : Tesla V100
- GPU Num : 1
- Kubernetes Version : v1.12.9
- Docker Version : v18.09.1

The benchmark is done by running multiple benchmarks on ai-benchmark. the following conclusions can be drawn：

- The overhead introduced by HAMi-vGPU is below 1%
- By using this component, more tasks shared on a GPU, increasing the overall throughput by 10%-90%
- Due to the limitation of computing resources, the use of VGPU has limited improvement for tasks have high GPU core utilization

HAMi Observability



Hami provides monitoring capability in two dimensions: Cluster level and Node Level, both in the form of metrics endpoint.

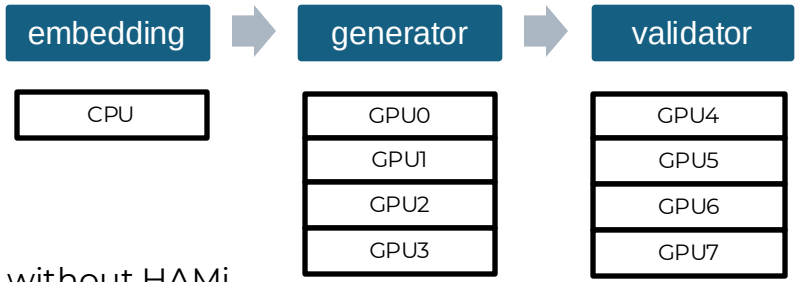
- {scheduler node ip}:31993/metrics records the snapshot of allocated devices, including, allocated device memory of each GPU, container name sharing each GPU, etc..

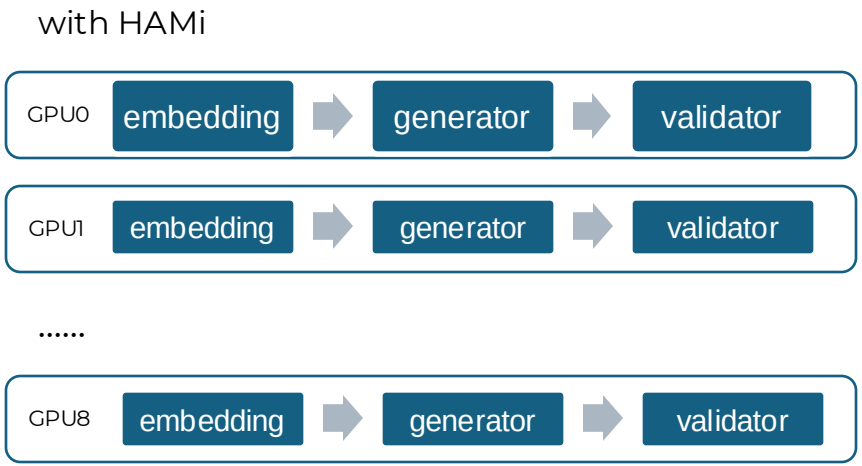
```
...
GPUDeviceMemoryAllocated{devicecores="0",deviceidx="0",deviceuuid="GPU-685ba63e-93b6-f43b-a55f-2c3377f45d70",nodeid="aio-a10",zone="vGPU"} 0
GPUDeviceMemoryAllocated{devicecores="0",deviceidx="0",deviceuuid="GPU-6d097a0e-596a-19b3-bffe-037f90c94e4a",nodeid="aio-a10",zone="vGPU"} 0
GPUDeviceMemoryAllocated{devicecores="0",deviceidx="0",deviceuuid="aio-node74-arm-Ascend310P-0",nodeid="aio-node74-arm",zone="vGPU"}3.221225472e+09
...
```

- {GPU node ip}:31992/metrics records real-tiume utilization of each container, including, real-time device memory usage, real-time device core utilization of certain container, etc..

```
...
Device_utilization_desc_of_container{ctrname="2-1-3-pod-1",deviceuuid="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec",podname="2-1-3-pod-1",podnamespace="default",vdeviceid="0",zone="vGPU"} 0
Device_utilization_desc_of_container{ctrname="2-1-3-pod-1",deviceuuid="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448",podname="2-1-3-pod-1",podnamespace="default",vdeviceid="1",zone="vGPU"} 0
...
```

Use Case：improve LLM inference





Generally speaking, a LLM product does not only contain a generator, but consists several ‘small’ models. Taking product [Shishuo] from 4th paradigm for example.

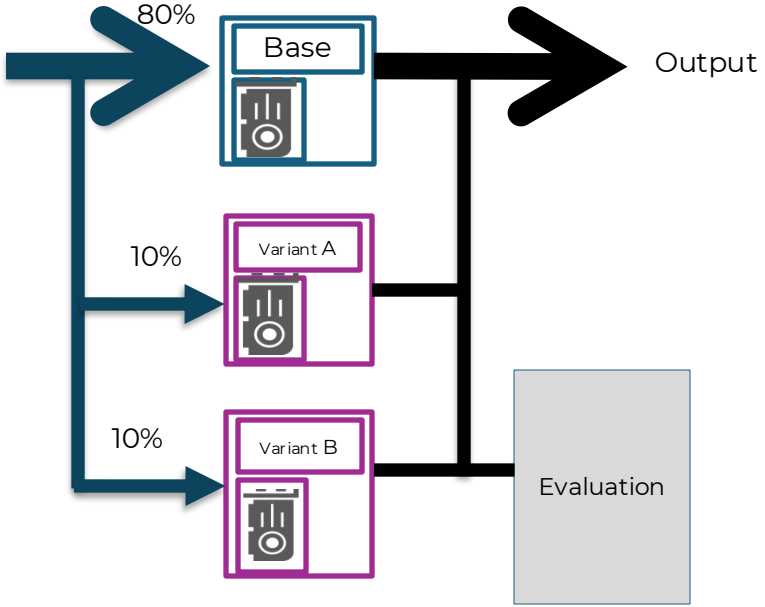
Without HAMi

- Embedding model in CPU
- Support 4 threads

With HAMi

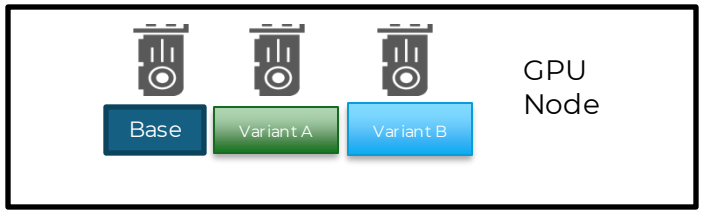
- All parts in GPU
- Support 8 threads
- Compatible with mainstream large model inference frameworks such as TGI, FastLLM, and vLLM
- The inference performance is improved by about 1-8 times.

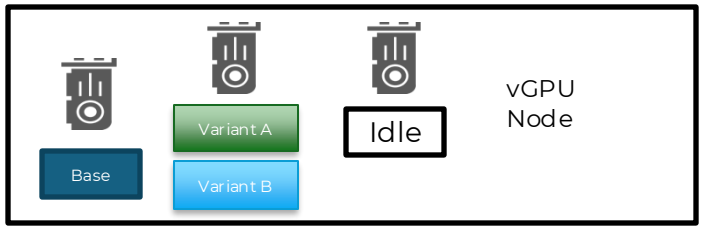
Use Case：Improve TCO on A/B test platform



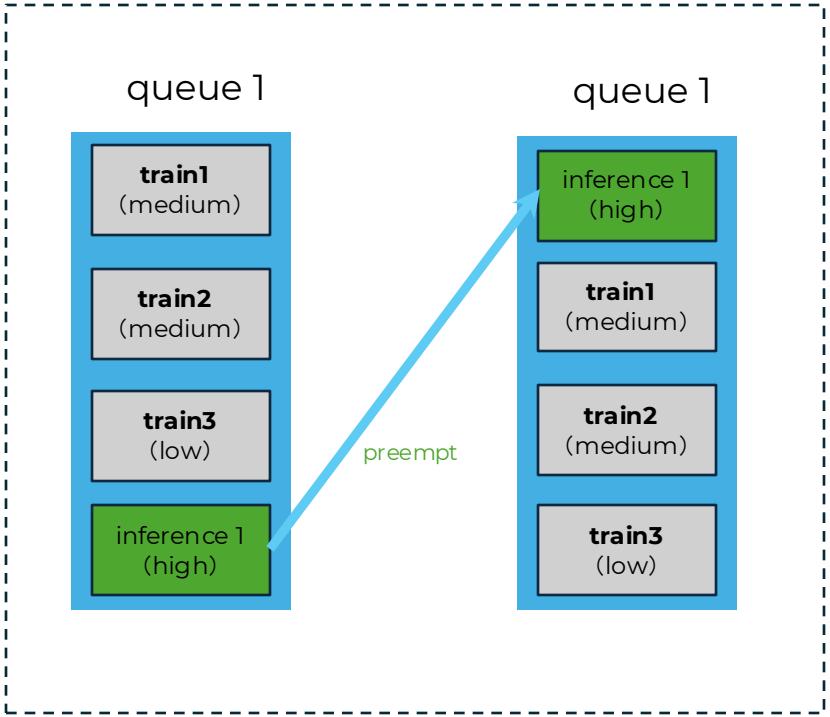
A typical A/B testing scenario is shown in the figure, it has the following features：

- The whole system consists of a base model and several experimental models, most of the input is processed by the production model, and a small part flows into the variant model
- Without the support of VGPU technology, each variant model requests an exclusive GPU, which is a serious waste of computing power





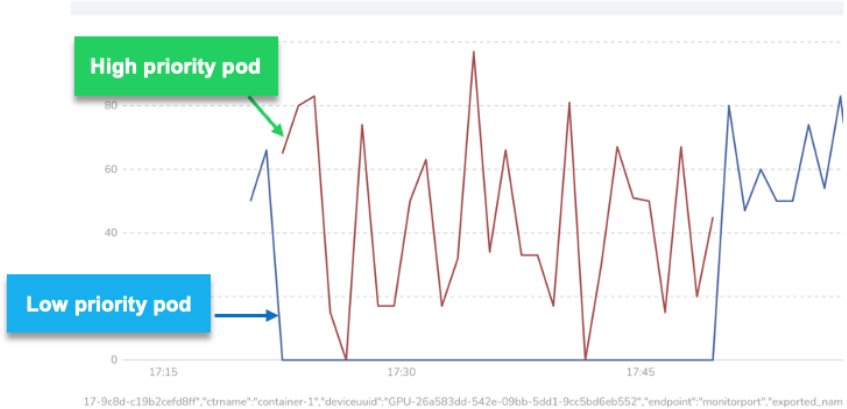
Use Case: co-located training&inference



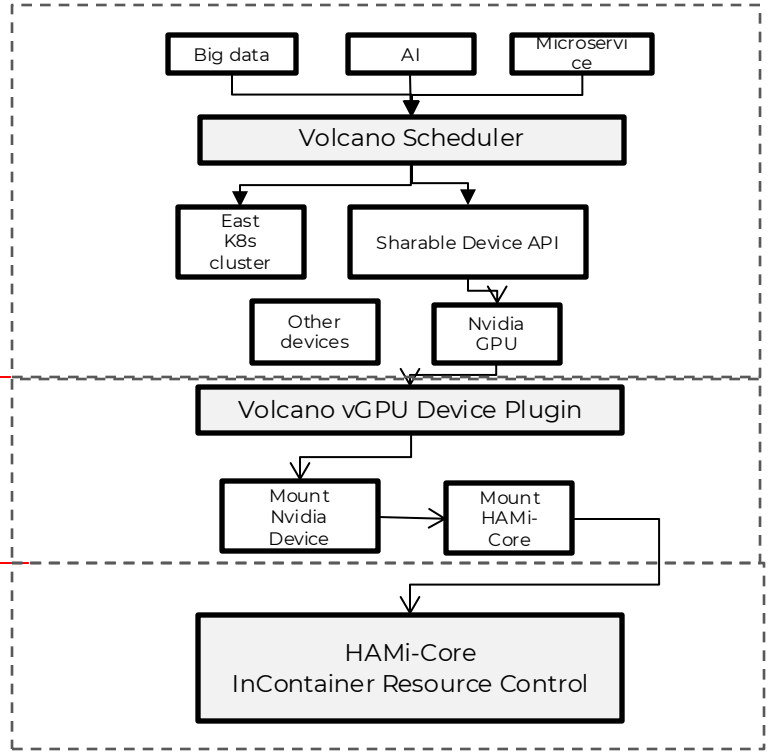
1. Enterprise GPU resources are limited, training and inference are co-located

2. Manage quotas through queue mechanisms (such as Kueue) to coordinate training and scheduling

3. If there is a high-priority task, it will be executed first, and other tasks will be in the pause state.



HAMi for volcano



Project-HAMi donates its GPU resource isolation components(HAMI-Core) to volcano, the whole architect is shown as the figure：

- At the scheduling layer, the Volcano scheduler is responsible for grasping the usage of devices in the cluster and assigning tasks to appropriate nodes
- At the device layer, the volcano device plugin is responsible for mounting the corresponding GPU device and HAMi-Core to the container at the same time, and configuring the HAMi-Core for it to take effect
- At the container layer, we use HAMi-core to limit device memory and device-cores

HAMi for volcano: User guide

HAMi works with volcano community to develop volcano-vgpu, follow the instructions below to try:

- Install volcano
- Configure volcano-scheduler-configmap, set `deviceshare.VGPUEnable` to true, as the figure below:
- Install volcano-vgpu-device-plugin from HAMi organization (https://github.com/Project-HAMi/volcano-vgpu-device-plugin)

```
$ kubectl edit cm -n volcano-system volcano-scheduler-configmap

kind: ConfigMap
...
    - plugins:
...
     - name: deviceshare
       arguments:
         deviceshare.VGPUEnable: true # enable vgpu
...
```

A typical vGPU-task in kubernetes is shown below:

- Specify number of GPU mounted in container using 'volcano.sh/vgpu-number' resource name
- Specify available device memory for each GPU mounted using 'volcano.sh/vgpu-memory' resource name
- Set '.spec.schedulerName' to volcano

```
$ cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod12
spec:
  schedulerName: volcano
  containers:
    - name: ubuntu-container
      image: ubuntu:18.04
      command: ["bash", "-c", "sleep 86400"]
      resources:
        limits:
          volcano.sh/vgpu-number: 2 # requesting 2 vGPUs
          volcano.sh/vgpu-memory: 10240
```

HAMi for volcano: Observability

volcano-scheduler-metrics records every GPU usage and limitation, visit the following address to get these metrics.

```
curl {vc-scheduler cluster ip}:8080/metrics
```

The snapshot of sharable GPUs in volcano cluster is shown as below:

```
# HELP volcano_vgpu_device_allocated_cores The percentage of gpu compute cores allocated in this card
# TYPE volcano_vgpu_device_allocated_cores gauge
volcano_vgpu_device_allocated_cores{NodeName="aio-node67",devID="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec"} 0
volcano_vgpu_device_allocated_cores{NodeName="aio-node67",devID="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448"} 0
volcano_vgpu_device_allocated_cores{NodeName="m5-cloudinfra-online01",devID="GPU-a88b5d0e-eb85-924b-b3cd-c6cad732f745"} 0
volcano_vgpu_device_allocated_cores{NodeName="m5-cloudinfra-online01",devID="GPU-d2407b50-70b1-f427-d712-801233c47b67"} 0
# HELP volcano_vgpu_device_allocated_memory The number of vgpu memory allocated in this card
# TYPE volcano_vgpu_device_allocated_memory gauge
volcano_vgpu_device_allocated_memory{NodeName="aio-node67",devID="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec"} 32768
volcano_vgpu_device_allocated_memory{NodeName="aio-node67",devID="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448"} 32768
volcano_vgpu_device_allocated_memory{NodeName="m5-cloudinfra-online01",devID="GPU-a88b5d0e-eb85-924b-b3cd-c6cad732f745"} 0
volcano_vgpu_device_allocated_memory{NodeName="m5-cloudinfra-online01",devID="GPU-d2407b50-70b1-f427-d712-801233c47b67"} 0
# HELP volcano_vgpu_device_core_allocation_for_a_vertain_pod The vgpu device core allocated for a certain pod
# TYPE volcano_vgpu_device_core_allocation_for_a_vertain_pod gauge
volcano_vgpu_device_core_allocation_for_a_vertain_pod{NodeName="aio-node67",devID="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec",podName="resnet101-deployment-7b487d974d-jjc8p"} 0
volcano_vgpu_device_core_allocation_for_a_vertain_pod{NodeName="aio-node67",devID="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec",podName="resnet101-deployment-7b487d974d-kpckz"} 0
volcano_vgpu_device_core_allocation_for_a_vertain_pod{NodeName="aio-node67",devID="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448",podName="resnet101-deployment-7b487d974d-8sjtk"} 0
volcano_vgpu_device_core_allocation_for_a_vertain_pod{NodeName="aio-node67",devID="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448",podName="resnet101-deployment-7b487d974d-xp4t4"} 0
# HELP volcano_vgpu_device_memory_allocation_for_a_certain_pod The vgpu device memory allocated for a certain pod
# TYPE volcano_vgpu_device_memory_allocation_for_a_certain_pod gauge
volcano_vgpu_device_memory_allocation_for_a_certain_pod{NodeName="aio-node67",devID="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec",podName="resnet101-deployment-7b487d974d-jjc8p"} 16384
volcano_vgpu_device_memory_allocation_for_a_certain_pod{NodeName="aio-node67",devID="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec",podName="resnet101-deployment-7b487d974d-kpckz"} 16384
volcano_vgpu_device_memory_allocation_for_a_certain_pod{NodeName="aio-node67",devID="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448",podName="resnet101-deployment-7b487d974d-8sjtk"} 16384
volcano_vgpu_device_memory_allocation_for_a_certain_pod{NodeName="aio-node67",devID="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448",podName="resnet101-deployment-7b487d974d-xp4t4"} 16384
# HELP volcano_vgpu_device_memory_limit The number of total device memory in this card
# TYPE volcano_vgpu_device_memory_limit gauge
volcano_vgpu_device_memory_limit{NodeName="aio-node67",devID="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec"} 32768
volcano_vgpu_device_memory_limit{NodeName="aio-node67",devID="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448"} 32768
volcano_vgpu_device_memory_limit{NodeName="m5-cloudinfra-online01",devID="GPU-a88b5d0e-eb85-924b-b3cd-c6cad732f745"} 32768
volcano_vgpu_device_memory_limit{NodeName="m5-cloudinfra-online01",devID="GPU-d2407b50-70b1-f427-d712-801233c47b67"} 32768
# HELP volcano_vgpu_device_shared_number The number of vgpu tasks sharing this card
# TYPE volcano_vgpu_device_shared_number gauge
volcano_vgpu_device_shared_number{NodeName="aio-node67",devID="GPU-00552014-5c87-89ac-b1a6-7b53aa24b0ec"} 2
volcano_vgpu_device_shared_number{NodeName="aio-node67",devID="GPU-0fc3eda5-e98b-a25b-5b0d-cf5c855d1448"} 2
volcano_vgpu_device_shared_number{NodeName="m5-cloudinfra-online01",devID="GPU-a88b5d0e-eb85-924b-b3cd-c6cad732f745"} 0
volcano_vgpu_device_shared_number{NodeName="m5-cloudinfra-online01",devID="GPU-d2407b50-70b1-f427-d712-801233c47b67"} 0
```

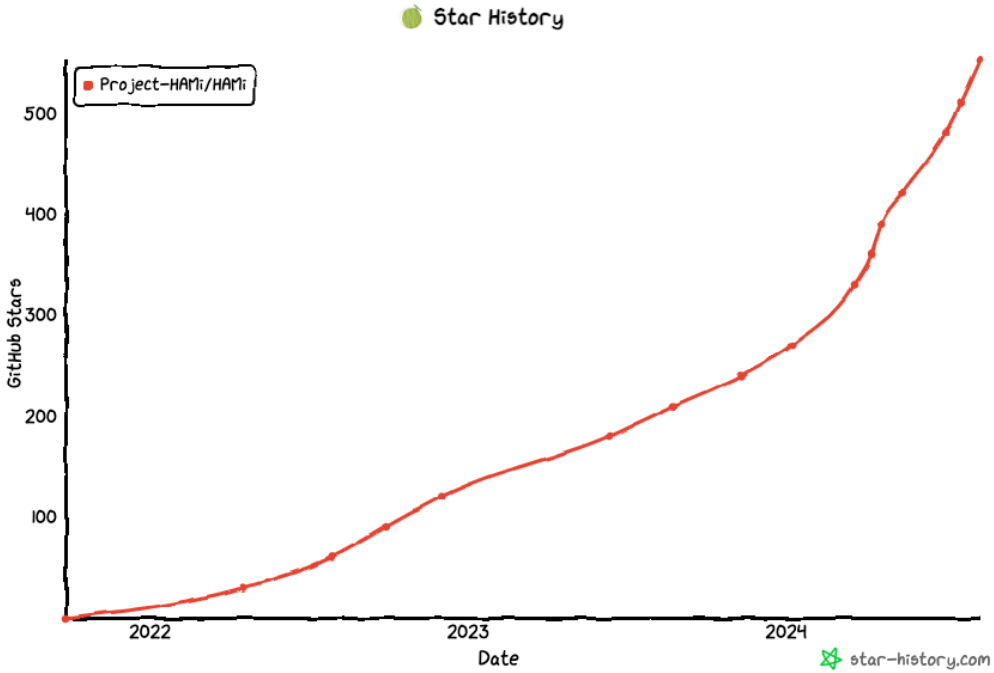
HAMi Compare with other projects

Key features	HAMi	NVIDIA/k8s-device-plugin	NVIDIA/k8s-dra-driver
Support Other AI devices	✅(NVIDIA, Cambricon, Hygon, iluvatar, Huawei Ascend)	❌ only nvidia gpu	❌ only nvidia gpu
gpu sharing	✅(software definition)	✅(timeslice + mps + mig)	✅(timeslice + mps + mig)
Observability	HAMi metrics + DCGM exporter	DCGM exporter	DCGM exporter
cuda support matrix	> 10.2	unknown	unknown
os support matrix	>= 3.10	unknown	unknown
Kubernetes support matrix	>= 1.16	>= 1.10	>= 1.26
ARCH	K8s scheduler-plugin device-plugin	device-plugin, no scheduler	DRA
Task priority	✅	❌	❌
GPU Core Over subscription	✅	❌	❌
GPU Memory Over subscription	✅(By CUDA Unified Memory)	❌	❌
Node level spread/binpack	✅	❌	❌
GPU level spread/binpack	✅	❌	❌
Deployment method	Helm	Helm	Helm

HAMi Compare with other GPU share proposal

	HAMi vgpu	CUDA Streams	MPS	Time-slicing	MIG	Nvidia vGPU
Target Use Cases	The same cluster contains multiple heterogeneous AI devices+ Gpu sharing + flexible scheduler policies	Optimized for concurrencywi thin a single application	When running multipleapplications in parallel butcan deal with limitedresiliency	When running multipleapplication s that are notlatency-sensitive or cantolerate jitter	When running multipleapplications in parallel butneed resiliencyand QoS	When needing to supportmulti-tenancy on the GPUthrough virtualization
Partition Type	Logical	Single Process	Logical	Temporal	Physical	Temporal & Physical (VM)
Max Partitions	Unlimited	Unlimited	48	Unlimited	7	Variable
SM Performance Isolation	Yes(by % not per client)	No	Yes(by % not per client)	Yes	Yes	Yes
Memory Protection	Yes	No	Yes	Yes	Yes	Yes
Memory Bandwidth QoS	No	No	No	No	Yes	Yes
Error Isolation	Yes	No	No	Yes	Yes	Yes
Cross-PartitionInteroperability		Always	IPC	Limited IPC	Limited IPC	No
Reconfiguration	At process Launch	Dynamic	At process Launch	Time-Slice Duration Only	When Idle	No
Telemetry	Yes	No	Limited	No	Yes(including in containers)	Yes(including live migration)
Other noteworthy	Supports all GPUs, open source		cudaCapability >= 3.5	cudaCapability >= 7.0	cuda capability >= 8.0 Hopper,Ampere	license required

HAMi Community



- 2022.04 Open Source
- 2024.04 CNCF Landscape project
- 2024.08 CNCF Sandbox project request
- Fast growinq community
  - 10K+ Downloads
  - 40+ Adopters
  - Already Support Nvidia,Cambricon,Hygon,Huawei ASCEND

HAMi 💘 Adopters

4Paradigm 第四范式

ICBC

DaoCloud

中国东信 China-ASEAN Information Harbor

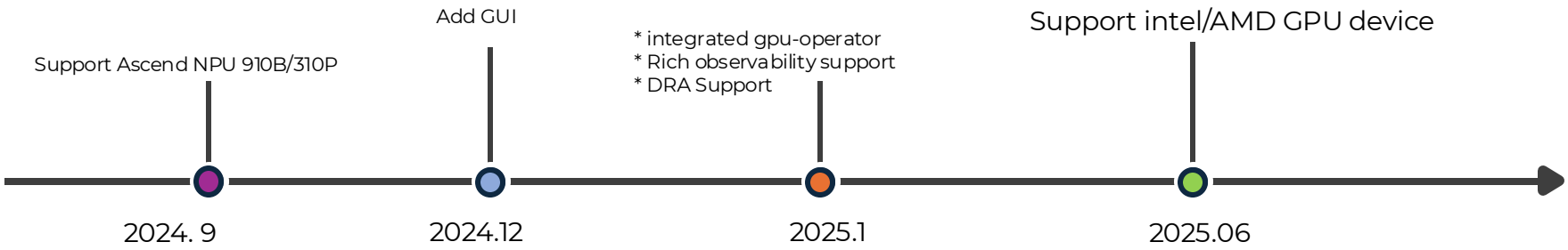
初灵股份 CNCR CHINESE CHURINGA

新网银行

科大讯飞 IFLYTEK

HUAWEI

Roadmap


Support Ascend NPU 910B/310P
Add GUI
* integrated gpu-operator
* Rich observability support
* DRA Support
Support intel/AMD GPU device
2024. 9
2024.12
2025.1
2025.06

Join Us

- **Github**

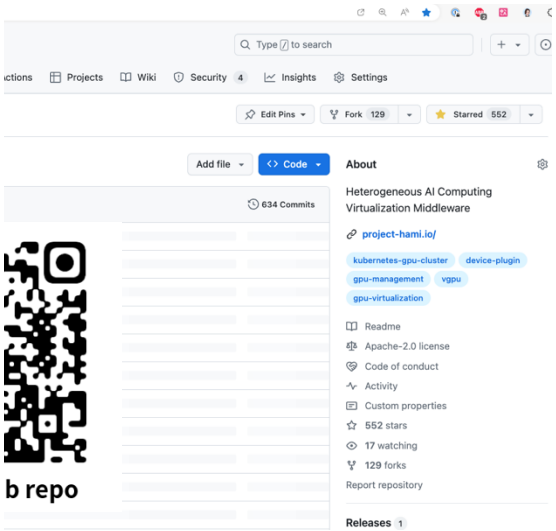
  https://githu

- **Website**

  http://projec

- **Slack**

HAMi 开发者交流群



Other Practices

THANKS