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

Is Your GPU Really Working Efficiently in the Data Center?

N Ways to Improve GPU Usage

Xiao Zhang, DaoCloud & Wu Ying Jun, China Mobile Cloud

About us

Xiao Zhang software engineer

Github @wawa0210

DaoCloud

Wu Ying Jun

Github @wuyingjun-lucky

China Mobile Cloud

Challenge: availability,cost,infrastructure,utilization

KEY FINDINGS

1 96% of companies plan to expand their AI compute capacity and investment with availability, cost, and infrastructure challenges weighing on their minds.

Nearly all respondents (96%) plan to expand their AI compute infrastructure, with 40% considering more on-premise and 60% considering more cloud, and they are looking for flexibility and speed. The top concern for cloud compute is wastage and idle costs.

When asked about challenges in scaling AI for 2024, compute limitations (availability and cost) topped the list, followed by infrastructure issues. Respondents felt they lacked automation or did not have the right systems in place.

The biggest concern for deploying generative AI was moving too fast and missing important considerations (e.g. prioritizing the wrong business use cases). The second-ranked concern was moving too slowly due to a lack of ability to execute.

2 A staggering 74% of companies are dissatisfied with their current job scheduling tools and face resource allocation constraints regularly, while limited on-demand and self-serve access to GPU compute inhibits productivity.

Job scheduling capabilities vary, and executives are generally not

3 The key buying factor for inference solutions is cost.

To address GPU scarcity, approximately 52% of respondents reported actively looking for cost-effective alternatives to GPUs for inference in 2024 as compared to 27% for training, signaling a shift in AI hardware usage. Yet, one-fifth of respondents (20%) reported that they were interested in cost-effective alternatives to GPU but were not aware of existing alternatives.

This indicates that cost is a key buying factor for inference solutions, and we expect that as most companies have not reached Gen AI production at scale, the demand for cost-efficient inference compute will grow.

4 The biggest challenges for compute were latency, followed by access to compute and power consumption.

Latency, access to compute, and power consumption were consistently ranked as the top compute concerns across all company sizes and regions. More than half of respondents plan to use LLMs (LLama and LLama-like models) in 2024, followed by embedding models (BERT and family) (26%) in their commercial deployments in 2024. Mitigating compute challenges will be essential in realizing their aspirations.

5 Optimizing GPU utilization is a major concern for 2024-2025, with the majority of GPUs underutilized during peak times.

40% of respondents, regardless of company size, are planning to use orchestration and scheduling technology to maximize their existing compute infrastructure.

When asked about peak periods for GPU usage, 15% of respondents report that less than 50% of their available and purchased GPUs are in use. 53% believe 51-70% of GPU resources are utilized, and just 25% believe their GPU utilization reaches 85%. Only 7% of companies believe their GPU infrastructure achieves more than 85% utilization during peak periods.

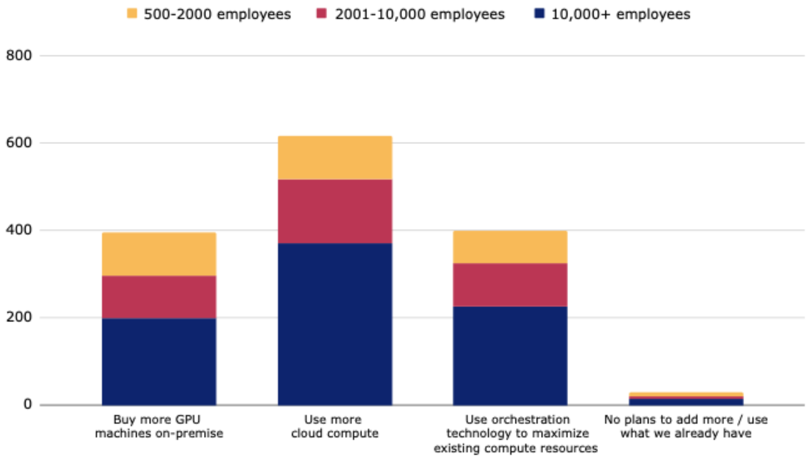
When asked about current methods employed for managing GPU usage, respondents are employing queue management and job scheduling (67%), multi-instance GPUs (39%), and quotas (34%). Methods of optimizing GPU allocation between users include Open Source solutions (24%), HPC solutions (27%), and vendor-specific solutions (34%). Another 11% use Excel and 5% have a home-grown solution. Only 1% of respondents do not maximize or optimize their GPU utilization.

6 Open Source AI solutions and model customization are top priorities, with 96% of companies focused on customizing primarily Open Source models.

Almost all executives (95%) reported that having and using external Open Source technology solutions is important for their organization.

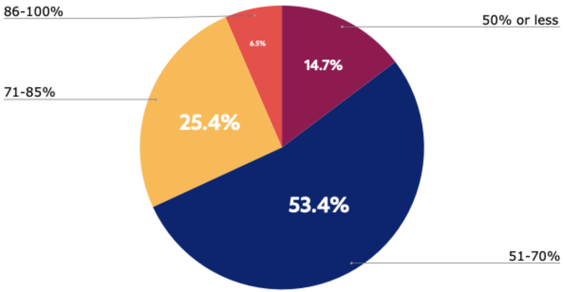
In addition, 96% of companies surveyed are currently or planning to customize Open Source models in 2024, with Open Source frameworks having the highest adoption globally. PyTorch was the leading framework for customizing Open Source models, with 61% of respondents using PyTorch, 43% using TensorFlow, and 16% using Jax. Approximately one-third of respondents currently use or plan to use CUDA for model customization.

Challenge: low utilization



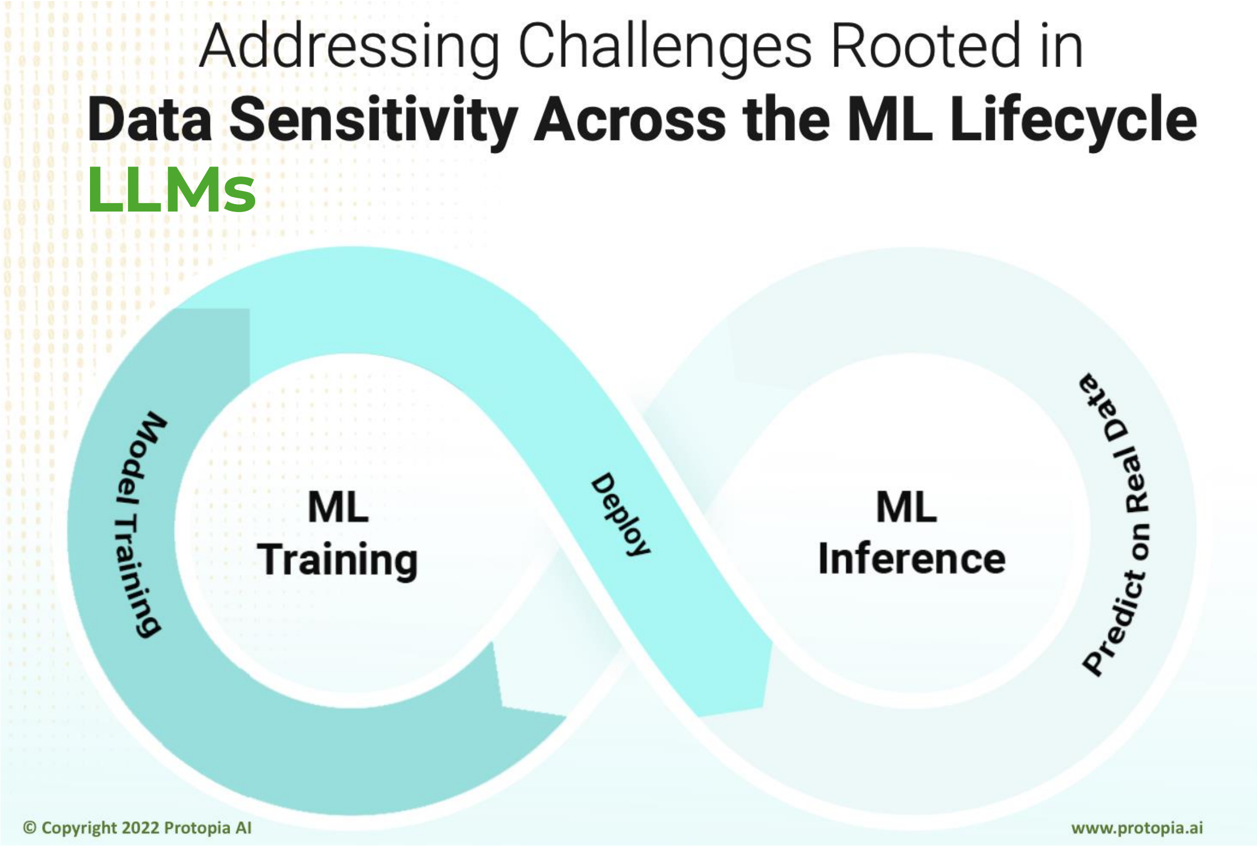
When asked about peak periods for GPU usage, 15% of respondents report that fewer than 50% of available GPUs are in use. 53% believe 51-70% of GPU resources are utilized, and 25% believe their GPU utilization reaches 85%. Only 7% of companies believe their GPU infrastructure achieves more than 85% utilization during peak periods.

Most respondents (78%) are using more than 50% of their total allocation of existing GPU resources during peak periods, indicating the need to better manage their existing compute and/or expand their compute with alternatives.



Nearly **75%** of users have a GPU utilization rate of no more than **70%**.
How to maximize resource utilization using orchestration or other tools becomes a consideration

train and inference



issues

Issues

Increasing parameter scale and sample data

Rapidly growing computing power demand

Insufficient training scale, inefficient, unstable

Demands

Improve LLMS training scale, efficiency

Increase stable training period

Cloud-Native becomes the solution for LLMs training

Challenge 1

How to use cloud-native technology to improve training scale and efficiency

Scale

Model Parallelism (Tensor + Pipeline)

Data Parallelism

Resolve the problem of excessive parameter scale

Address the issue of excessive sample data

Orchestration

KEYS

Communication overhead:

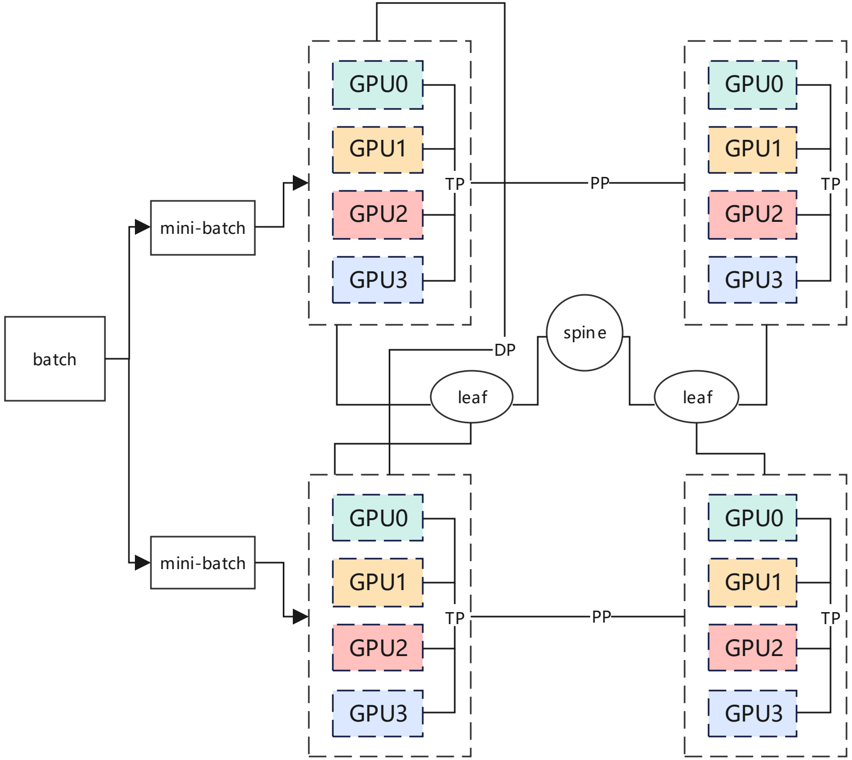
Tensor Parallelism > Data Parallelism > Pipeline Parallelism

Tensor Parallelism > Data Parallelism > Pipeline Parallelism

Network topology:

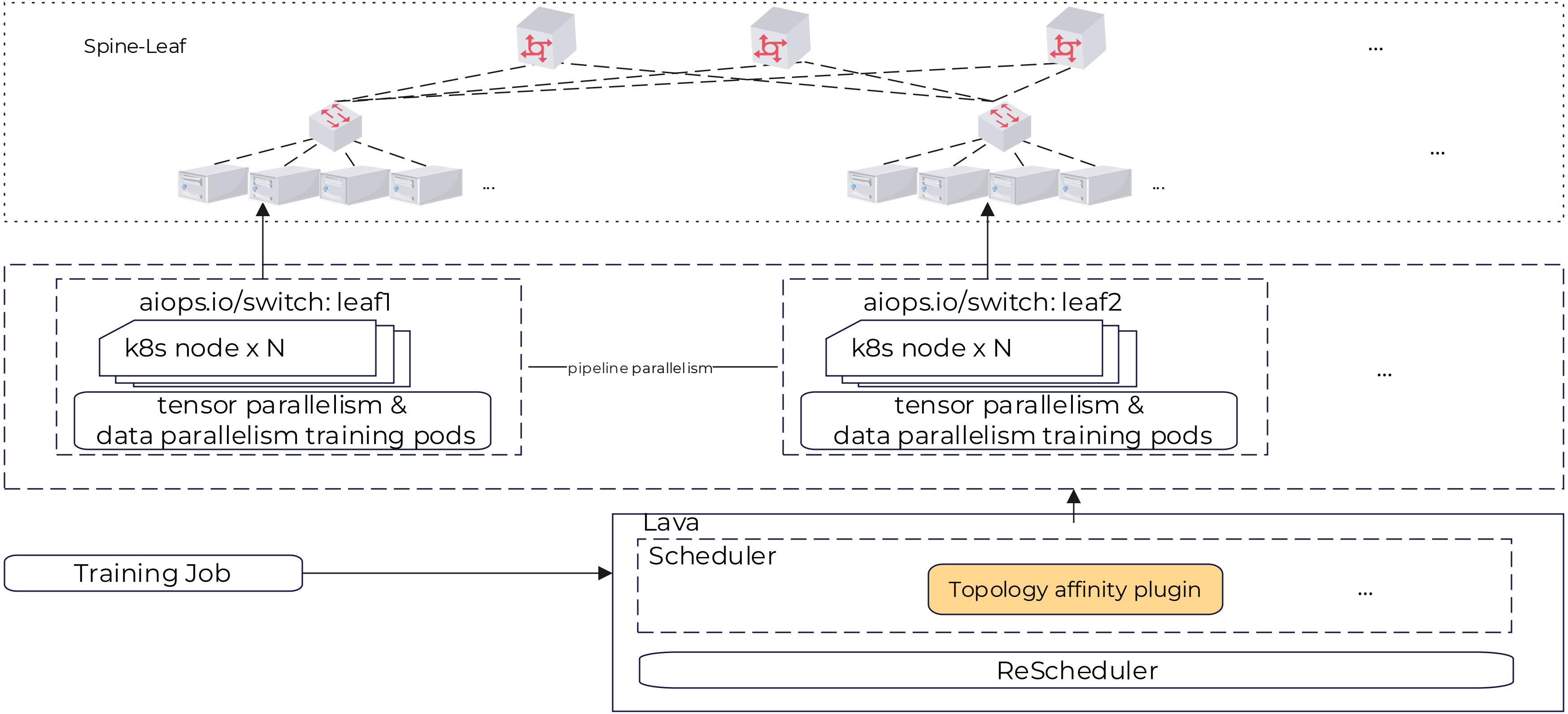
Dual (Triple) Layer Parameter TOR Switch

Orchestration



Orchestration In Kubernetes Cluster

Optimal Orchestration of LLMs training tasks based on the parameter network topology



Results

Scale: 8K NPUS (1000 nodes) parallel training
Efficiency: Linear acceleration ratio of 95%

```
modellink-test-gpt-worker-970   1/1     Running   0          30h
modellink-test-gpt-worker-971   1/1     Running   0          30h
modellink-test-gpt-worker-972   1/1     Running   0          30h
modellink-test-gpt-worker-973   1/1     Running   0          30h
modellink-test-gpt-worker-974   1/1     Running   0          30h
modellink-test-gpt-worker-975   1/1     Running   0          30h
modellink-test-gpt-worker-976   1/1     Running   0          30h
modellink-test-gpt-worker-977   1/1     Running   0          30h
modellink-test-gpt-worker-978   1/1     Running   0          30h
modellink-test-gpt-worker-979   1/1     Running   0          30h
modellink-test-gpt-worker-98    1/1     Running   0          30h
modellink-test-gpt-worker-980   1/1     Running   0          30h
modellink-test-gpt-worker-981   1/1     Running   0          30h
modellink-test-gpt-worker-982   1/1     Running   0          30h
modellink-test-gpt-worker-983   1/1     Running   0          30h
modellink-test-gpt-worker-984   1/1     Running   0          30h
modellink-test-gpt-worker-985   1/1     Running   0          30h
modellink-test-gpt-worker-986   1/1     Running   0          30h
modellink-test-gpt-worker-987   1/1     Running   0          30h
modellink-test-gpt-worker-988   1/1     Running   0          30h
modellink-test-gpt-worker-989   1/1     Running   0          30h
modellink-test-gpt-worker-99    1/1     Running   0          30h
modellink-test-gpt-worker-990   1/1     Running   0          30h
modellink-test-gpt-worker-991   1/1     Running   0          30h
modellink-test-gpt-worker-992   1/1     Running   0          30h
modellink-test-gpt-worker-993   1/1     Running   0          30h
modellink-test-gpt-worker-994   1/1     Running   0          30h
modellink-test-gpt-worker-995   1/1     Running   0          30h
modellink-test-gpt-worker-996   1/1     Running   0          30h
modellink-test-gpt-worker-997   1/1     Running   0          30h
modellink-test-gpt-worker-998   1/1     Running   0          30h
```

Challenge 2

How to use cloud-native technology to improve training stability

Stability

Checkpoint Optimization

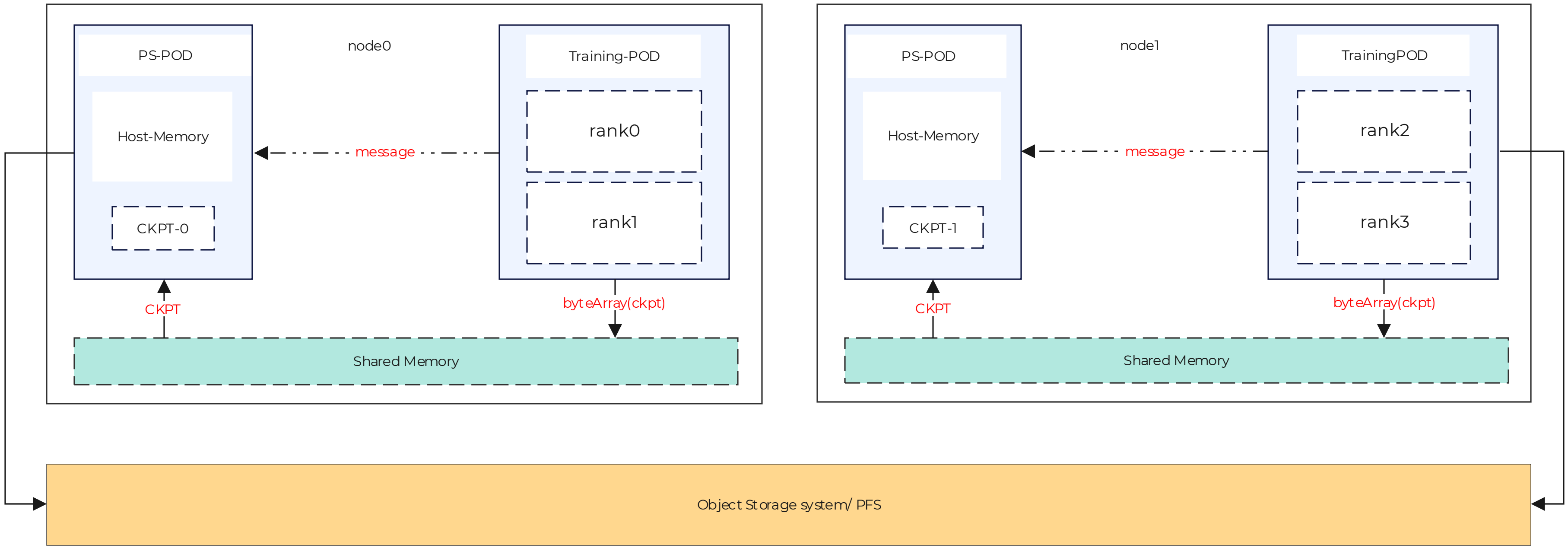
Checkpoint Recovery

Enhance the LLMs checkpoints persistence performance

Automatic recovery when LLMs training fail

Optimization： Soft Checkpoint

A training task is decomposed into N Training PODs and N ParameterServer PODs; Checkpoints are directly stored in memory through SharedMemory, accelerating the saving efficiency of Checkpoints. With pipeline parallelism and cutting Checkpoints, We can save CKPTs within 1 second.



Optimization Results - Soft Checkpoint


```
 iteration       59/    1000 | consumed samples:         3776 | elapsed time per iteration (ms): 16643.4 | learning rate: 1.243E-06 | glob
al batch size:    64 | lm loss: 6.073891E+00 | loss scale: 1.0 | grad norm: 32.791 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
 iteration       60/    1000 | consumed samples:         3840 | elapsed time per iteration (ms): 16665.3 | learning rate: 1.243E-06 | glob
al batch size:    64 | lm loss: 5.885174E+00 | loss scale: 1.0 | grad norm: 32.773 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
(min, max) time across ranks (ms):
    save-checkpoint .................................: (3371.84, 3372.05)
 iteration       61/    1000 | consumed samples:         3904 | elapsed time per iteration (ms): 16657.2 | learning rate: 1.243E-06 | glob
al batch size:    64 | lm loss: 6.043956E+00 | loss scale: 1.0 | grad norm: 34.839 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
 iteration       62/    1000 | consumed samples:         3968 | elapsed time per iteration (ms): 16637.7 | learning rate: 1.242E-06 | glob
al batch size:    64 | lm loss: 5.862806E+00 | loss scale: 1.0 | grad norm: 32.612 | number of skipped iterations:   0 | number of nan ite
```

use soft checkpoint

Soft Checkpoint (single node 112GB, without pipeline parallelism)

3.3S (28.7GB/s)

```
 iteration      299/    1000 | consumed samples:        19136 | elapsed time per iteration (ms): 16608.1 | learning rate: 1.030E-06 | glob
al batch size:    64 | lm loss: 7.321179E+00 | loss scale: 1.0 | grad norm: 23.954 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
 iteration      300/    1000 | consumed samples:        19200 | elapsed time per iteration (ms): 16632.0 | learning rate: 1.028E-06 | glob
al batch size:    64 | lm loss: 7.018424E+00 | loss scale: 1.0 | grad norm: 27.405 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
(min, max) time across ranks (ms):
    save-checkpoint .................................: (29978.50, 29978.72)
 iteration      301/    1000 | consumed samples:        19264 | elapsed time per iteration (ms): 16636.6 | learning rate: 1.027E-06 | glob
al batch size:    64 | lm loss: 7.166518E+00 | loss scale: 1.0 | grad norm: 26.494 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
 iteration      302/    1000 | consumed samples:        19328 | elapsed time per iteration (ms): 16646.2 | learning rate: 1.025E-06 | glob
al batch size:    64 | lm loss: 7.001492E+00 | loss scale: 1.0 | grad norm: 25.564 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
```

save checkpoint to ssd

SSD Checkpoint (single node 112GB)

29.9S (3.74GB/s)

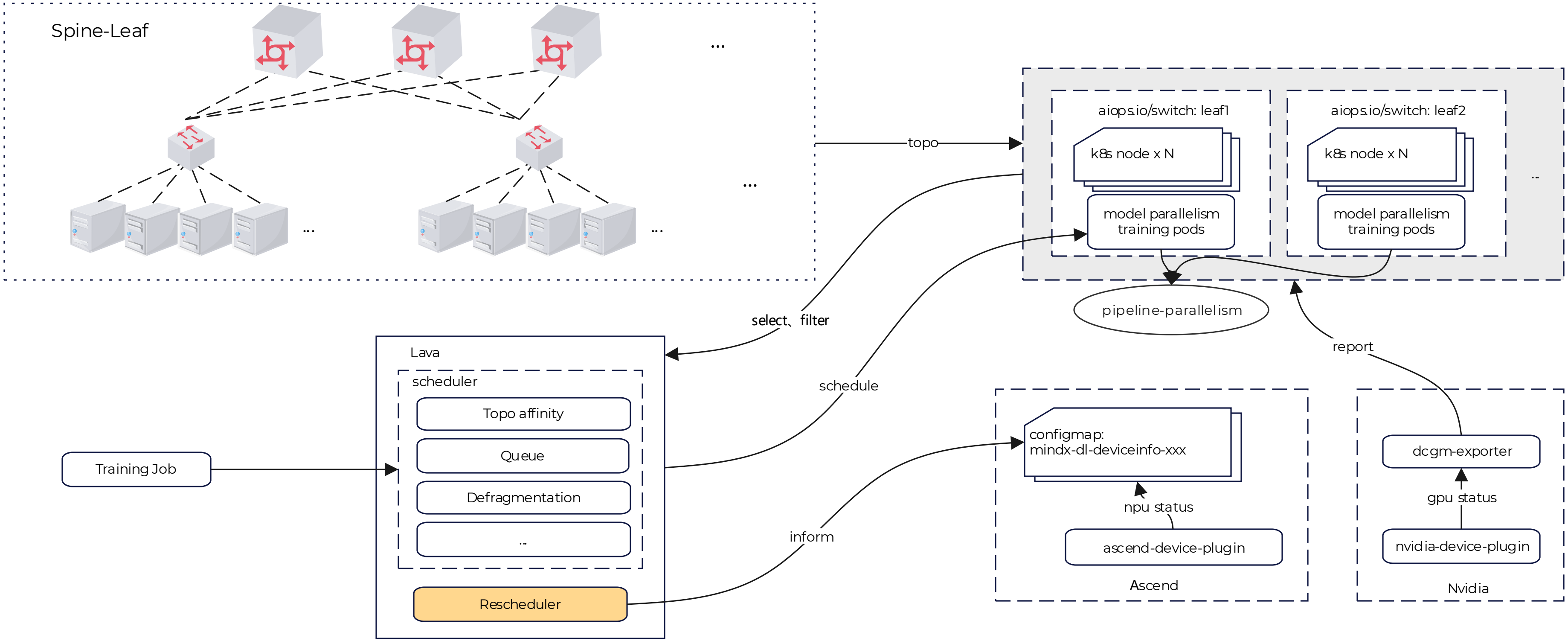
```
rations:   0 |
 iteration      449/    1000 | consumed samples:        28736 | elapsed time per iteration (ms): 16692.2 | learning rate: 7.869E-07 | glob
al batch size:    64 | lm loss: 6.601440E+00 | loss scale: 1.0 | grad norm: 27.309 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
 iteration      450/    1000 | consumed samples:        28800 | elapsed time per iteration (ms): 16660.8 | learning rate: 7.852E-07 | glob
al batch size:    64 | lm loss: 6.575200E+00 | loss scale: 1.0 | grad norm: 24.308 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
(min, max) time across ranks (ms):
    save-checkpoint .................................: (119853.55, 119853.78)
 iteration      451/    1000 | consumed samples:        28864 | elapsed time per iteration (ms): 16691.3 | learning rate: 7.834E-07 | glob
al batch size:    64 | lm loss: 6.572715E+00 | loss scale: 1.0 | grad norm: 26.287 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
 iteration      452/    1000 | consumed samples:        28928 | elapsed time per iteration (ms): 16649.2 | learning rate: 7.817E-07 | glob
al batch size:    64 | lm loss: 6.853421E+00 | loss scale: 1.0 | grad norm: 26.132 | number of skipped iterations:   0 | number of nan ite
rations:   0 |
```

save checkpoint to nfs

NFS Checkpoint (single node 112GB)

120S (1GB/s)

Checkpoint Recovery



Training task on thousands of NPUS running stably continuously for over 20 days.

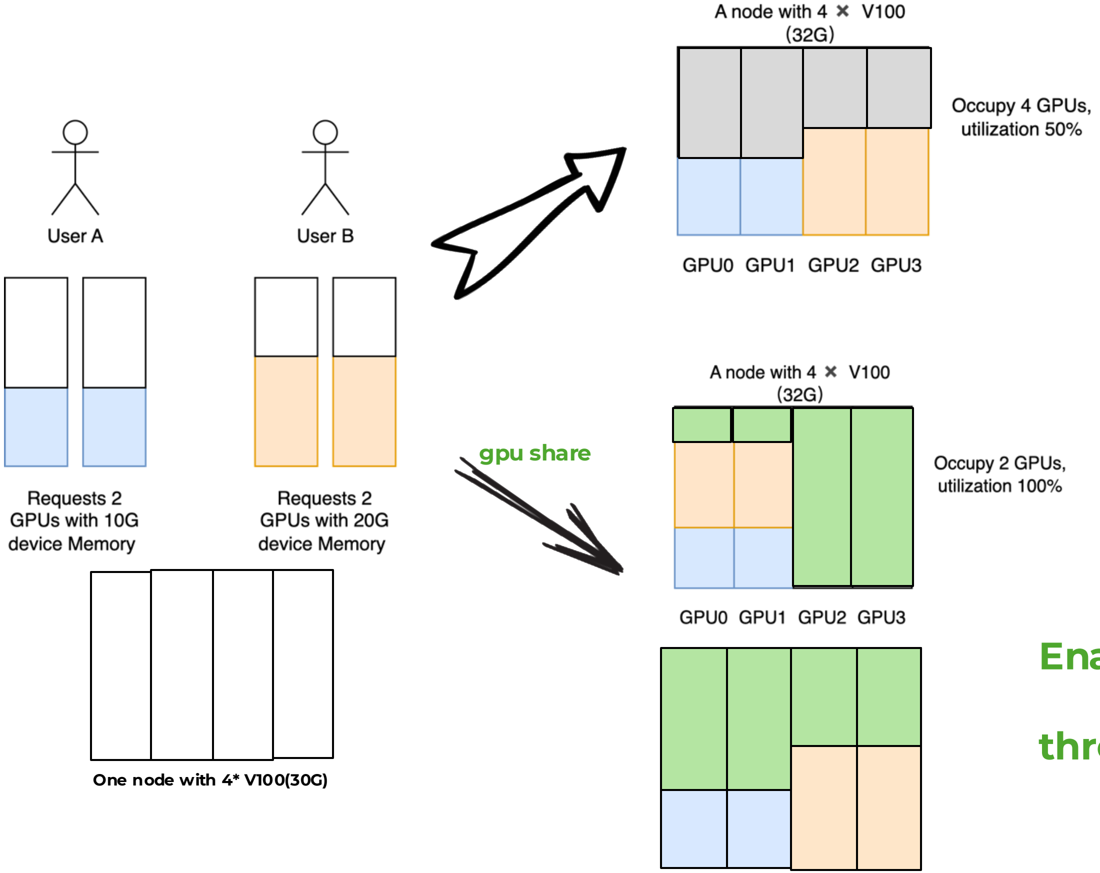
Minute-level fault detection, thousands of NPUS checkpoint recovery in less than 30 minutes.

Follow-up

Fused Kernel Acceleration

IO/Computation Overlap, etc

Optimize GPU utilization



Low gpu usage, but no new tasks can be scheduled

Enables more tasks to use GPU capabilities through GPU sharing

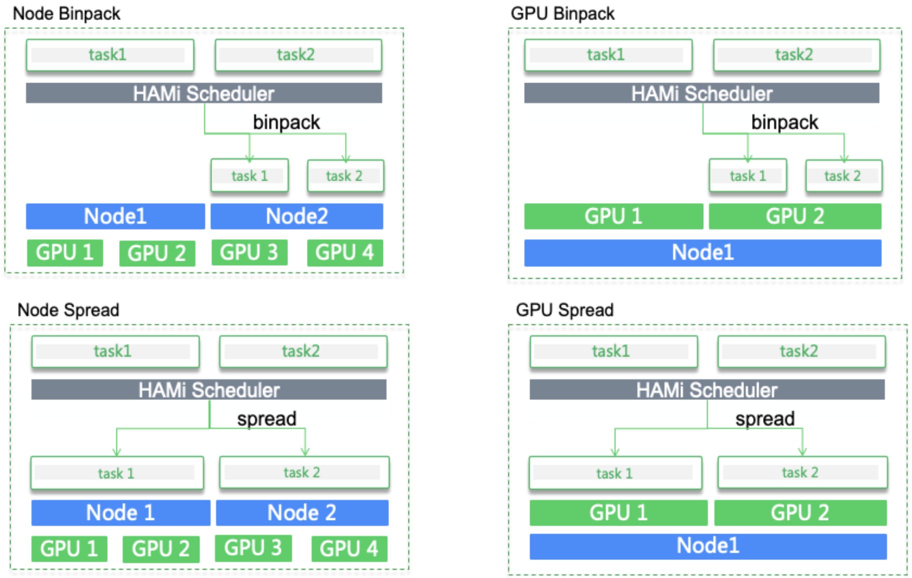
GPU share proposal

	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

Unbalanced Scheduling

More flexible scheduling strategies

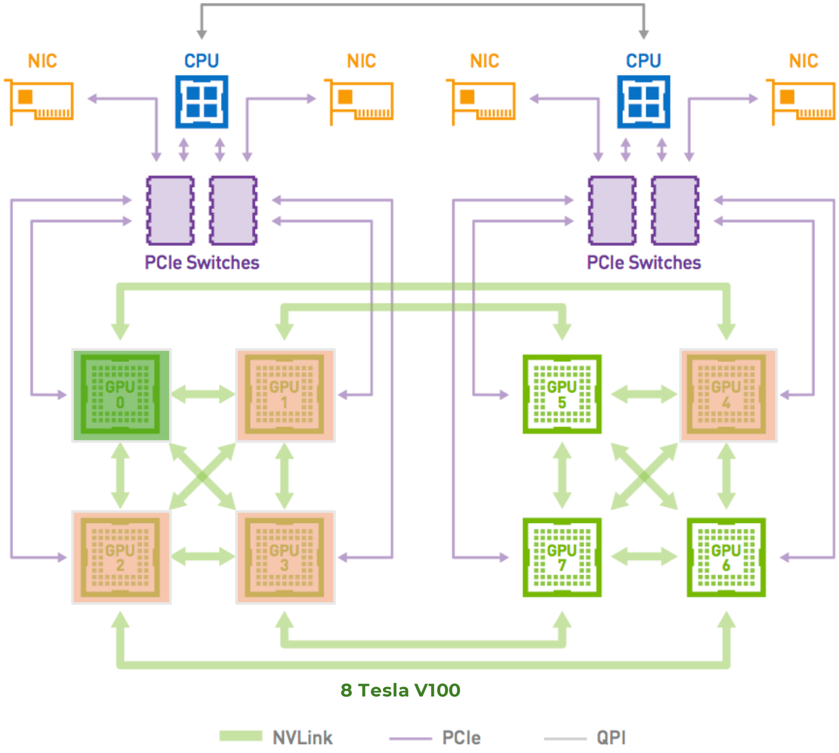
- ❖ Node Level binpack & spread
- ❖ GPU device Level binpack & spread



Best practice

- ❖ High QOS tasks prefer node-level spread.
- ❖ Low consume tasks prefer device-level binpack.
- ❖ For closely related tasks, node-level binpack and device-level spread are preferred.

GPU Topology-aware scheduling

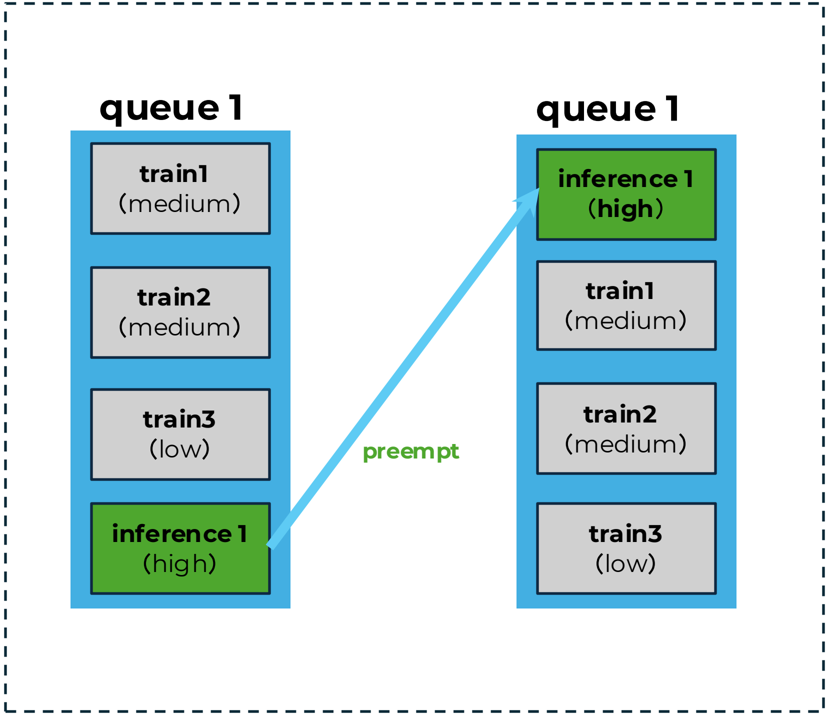


NVLink(25GB/s – 1800GB/s) > PCie(16GB/s)

Maximize task efficiency using **topology-aware scheduling**

If a task requires two GPUs, after selecting GPU0, which is the best choice for the remaining GPU?

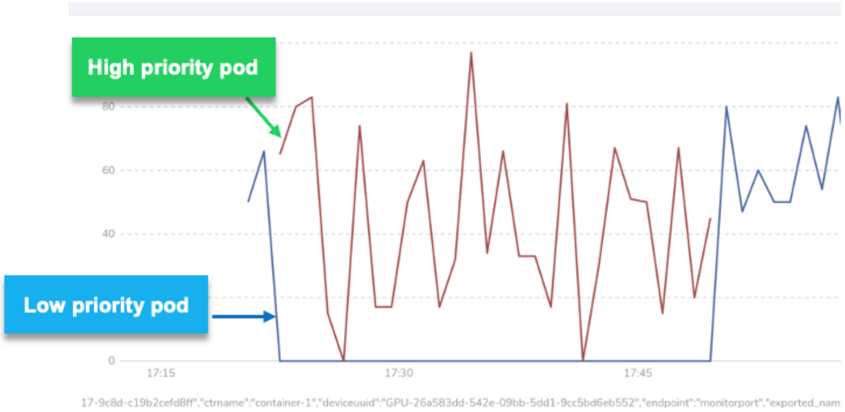
Priority scheduling & queue mechanism



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

2. Manage quotas through queue mechanisms (such as Kueue & Volcano) 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.



elastic quota

High and low priority tasks are deployed on different NS

NS QuotaA (min:4, max:6)
NS QuotaB (min:6, max:8)

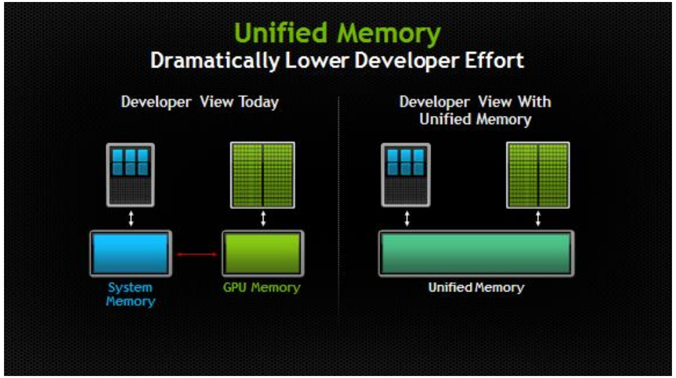
UserA use 4GPU (reach min),UserB use 3GPU,Sufficient resources, everything is ok

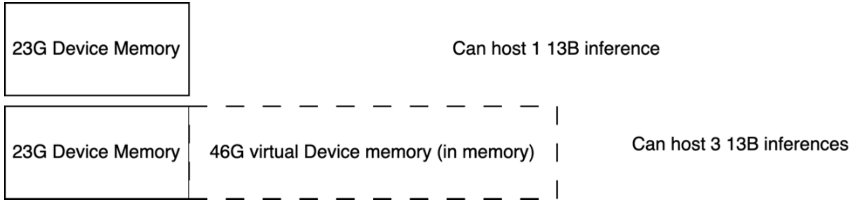
UserA max use 6GPU (reach max),UserB use 3GPU,Sufficient resources, everything is ok

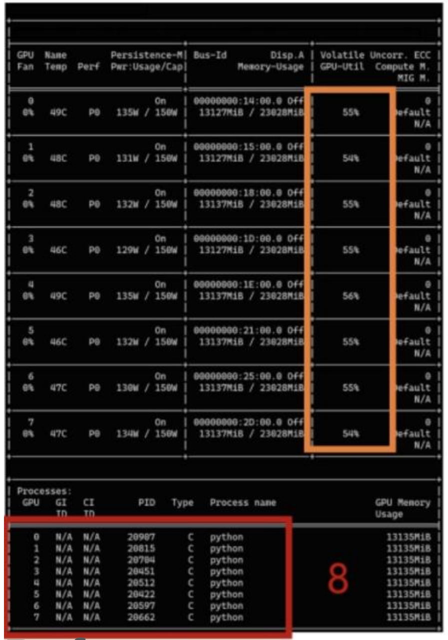
UserB continue use 3GPU, Eviction, preemption, UserA will return the resources to UserB

- https://github.com/kubernetes-sigs/scheduler-plugins/blob/master/pkg/capacityscheduling/README.md
- https://koordinator.sh/docs/user-manuals/capacity-scheduling/

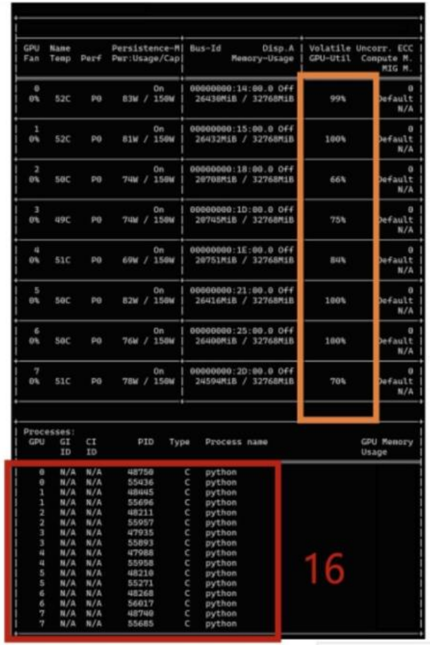
over-allocation mechanism







Before

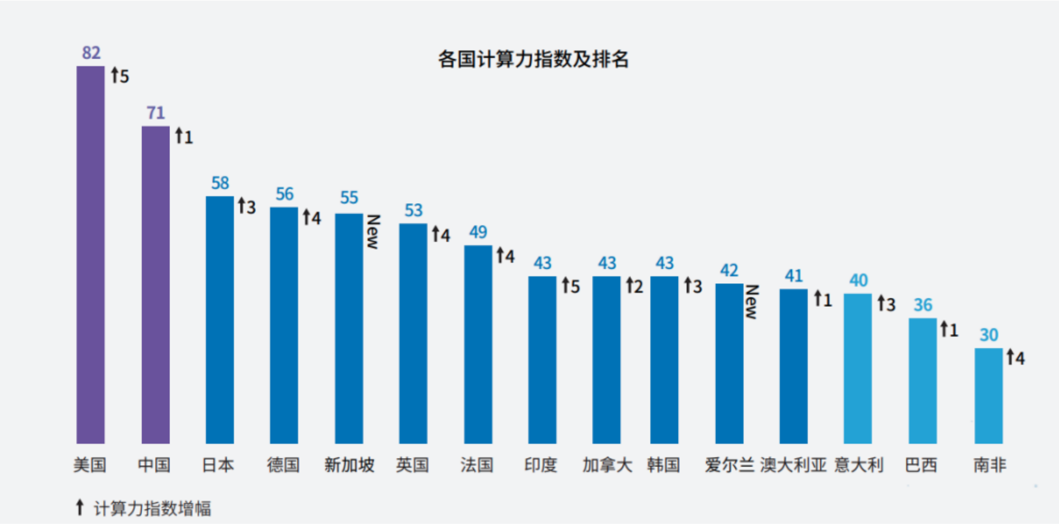


After

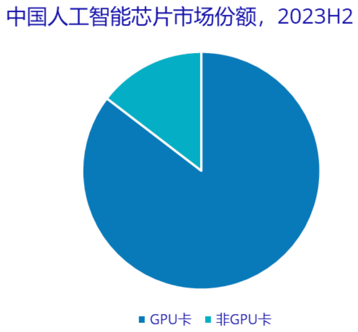
Through UMI, GPU and system memory are used together (CUDA allocated memory will be recalled to system memory if it is not used for a long time)

Typical use case: Hybrid Inference and time imbalance

Heterogeneous AI diversification



IDC



来源：IDC中国，20248

Shipments exceeded 1.4 million

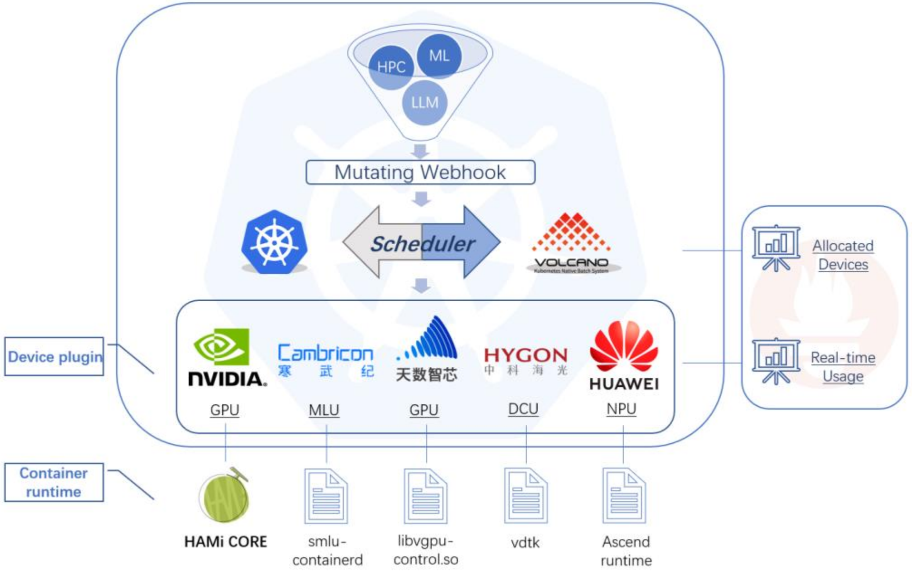
Nvidia for 85%, Huawei 10%, Baidu 2%, and others 2%

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.

Heterogeneous AI device management

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

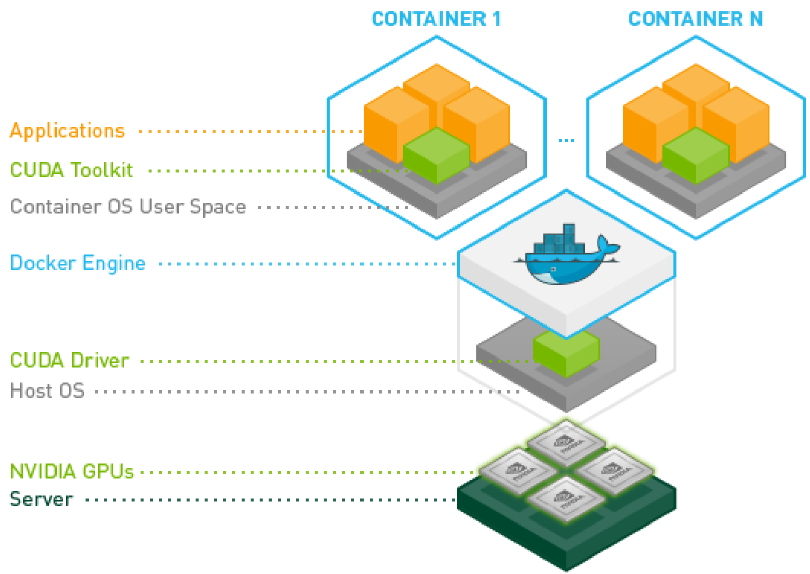


- Support multiple AI devices, Provide unified scheduling capabilities(NVIDIA, Cambricon, Hygon, iluvatar, Huawei Ascend)
- Device sharing
- Hard Resource Isolation inside container
- Device Type/UUID Specification
- Task priority
- CUDA Unified memory for NVIDIA
- Permits partial device allocation by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

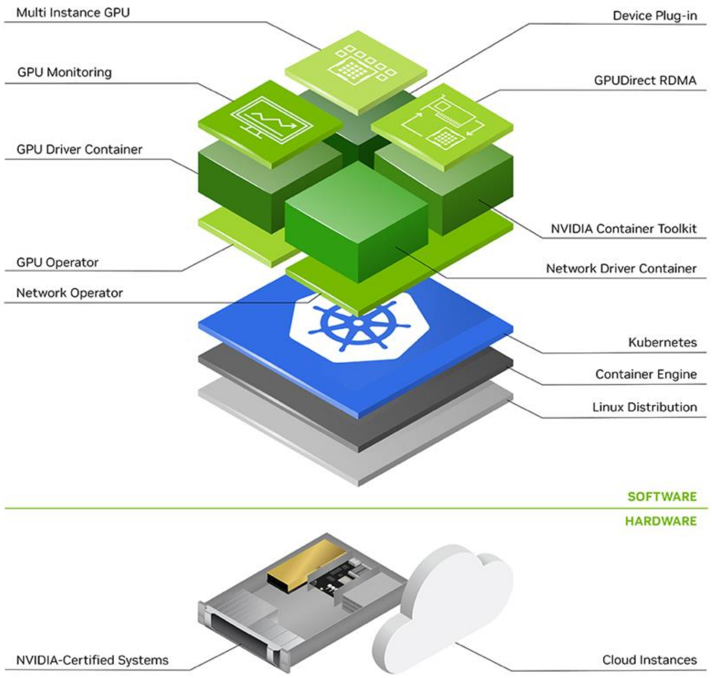
More about HAMi, https://sched.co/1eYYT(August 21(today), 2024 17:15 - 17:50 HKT, Hung Hom Room 3)

Other best practices

It is recommended to use gpu-operator to automatically manage the GPU software stack (driver management, CRI configuration, device-plugin, etc.)



→



Other Practices

THANKS