
KubeCon
CloudNativeCon
North America 2024

KubeCon | CloudNativeCon

North America 2024

Divide and Conquer:
Master GPU Partitioning and Visualize Savings with OpenCost

Kaysie Yu
Azure Kubernetes Service
Microsoft

Ally Ford
Azure Kubernetes Service
Microsoft

Agenda

- Why run AI workloads on AKS and GPUs?
- GPUs and cost…. what’s the issue?
- Monitor usage metrics and visualize costs
- Partitioning strategies
- Demo!

GPUs are scarce and expensive!

Utilization = amount of time during which one or more kernels was executing on the GPU

Exclusive GPU access is the default

You pay for the entire GPU despite the size of your workload

Is it feasible and economical to run AI workloads on GPUs at scale?

48% of orgs are using K8s for AI/ML workloads

State of Workloads Deployment on Containers and Kubernetes. Gartner peer Survey.

K8s is *the* ideal platform for GenAI, LLMs

- **Scalability**
  - Seamless and automatic for dynamic and resource intensive workloads like inferencing
- **Scheduling**
  - Reduce the operational overhead burden on MLOps teams
- **Dynamic**
  - Dynamically allocate resources
- **Flexibility**
  - Extensible and portable across clouds from public to onprem or hybrid

CPUs vs GPUs

CPUs

- **Ubiquitous** – not a single digital device that won't have one
- **Fast and versatile** – few operations executed quickly
- **Serialization** – can very efficiently switch between running different processes

GPUs

- **Specialized** – high performance, parallel processing, faster execution
- **Efficient resource utilization** – optimized for throughput, thousands of operations at once
- **Concurrency** – support large workloads, computational complex calculations without performance drops, concurrent tasks

GPU use cases


THIS IS NOT FINE!!
OH MY GOD EVERY THINGS ON FIRE
WHAT THE HELL IS MY PROBLEM


IT'S FINE
EVERYTHING WILL BE FINE

How can we control these costs?

Monitor GPU metrics

NVIDIA DCGM Exporter

Visualize GPU costs

OpenCost

Optimize costs

Rightsize and partition

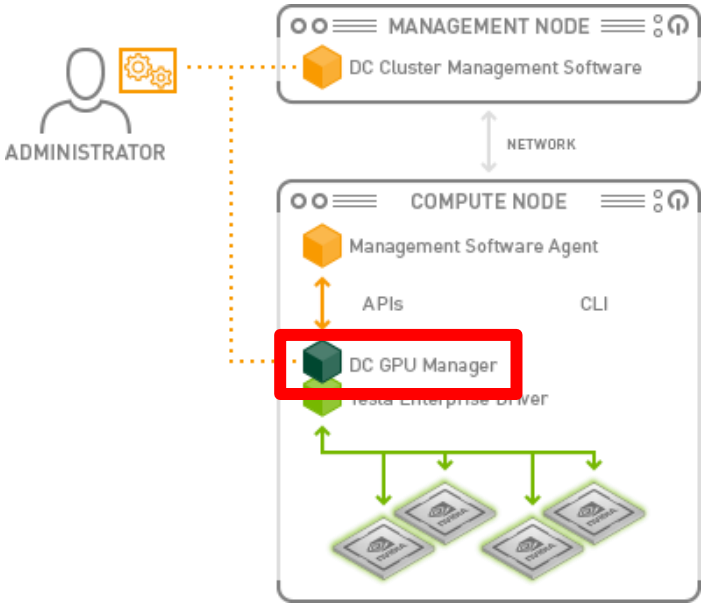
Monitor GPU metrics

NVIDIA DC GPU Manager (DCGM)

- Suite of tools from NVIDIA for managing and monitoring GPUs in cluster and datacenter environments

NVIDIA DCGM Exporter

- Metrics exporter for Prometheus
- Install methods
  - DCGM container
  - NVIDIA GPU Operator
- Integration with Grafana
- Extract valuable insights
  - Underutilization
  - GPU node health
  - Metrics customization



NVIDIA DCGM | NVIDIA Developer


General / Nvidia GPU Metrics
Last 1 minute
10s
GPU
1937b558-347d-0f30-105b-893b98985668
Name
NVIDIA GeForce RTX 2080 SUPER
P-State
P8
Driver Version
471.11
Vbios Version
90.04.7a.40.73
GPU Utilization %
0%
Power Draw %
11.5%
Fan Speed %
37%
Temperature
34 °C
Memory Utilization %
100%
75%
50%
25%
0%
21:43:00 21:43:10 21:43:20 21:43:30 21:43:40
2021-06-26 21:42:40
1937b558-347d-0f30-105b-893b98985668: 0%
Throttle Reasons
Idle Active
HW Thermal Slowdown Not Active
SW Power Cap Not Active
App Clocks Setting Not Active
HW Power Brake Not Active
SW Thermal Slowdown Not Active
Sync Boost Not Active
GPU Clock Speed %
0.395%
Memory Clock Speed %
0.310%
Memory Allocation %
11.2%
0%
GPU Utilization %
100%
75%
50%
25%
0%
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
Memory Allocation
1 GB
750 MB
500 MB
250 MB
0 B
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
Temperature
100 °C
75 °C
50 °C
25 °C
0 °C
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
Power Draw
40 W
30 W
20 W
10 W
0 W
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
Fan Speed %
100%
75%
50%
25%
0%
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
Graphics Clock Speed
15 MHz
10 MHz
5 MHz
0 Hz
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
Video Clock Speed
600 MHz
400 MHz
200 MHz
0 Hz
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
SM Clock Speed
15 MHz
10 MHz
5 MHz
0 Hz
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
Memory Clock Speed
30 MHz
20 MHz
10 MHz
0 Hz
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30


General / Nvidia GPU Metrics
Last 1 minute
10s
GPU
1937b558-347d-0f30-105b-893b98985668
Name
NVIDIA GeForce RTX 2080 SUPER
P-State
P8
Driver Version
471.11
Vbios Version
90.04.7a.40.73
GPU Utilization %
0%
Power Draw %
11.5%
Fan Speed %
37%
Temperature
34 °C
Memory Utilization %
100%
75%
50%
25%
0%
2021-06-26 21:42:40
1937b558-347d-0f30-105b-893b98985668: 0%
Throttle Reasons
Idle Active
HW Thermal Slowdown Not Active
SW Power Cap Not Active
App Clocks Setting Not Active
HW Power Brake Not Active
SW Thermal Slowdown Not Active
Sync Boost Not Active
GPU Clock Speed %
0.395%
Memory Clock Speed %
0.310%
Memory Allocation %
11.2%
0%
GPU Utilization %
21:42:40 21:42:50 21:43:00 21:43:10 21:43:20 21:43:30
Memory Allocation
1 GB
750 MB
500 MB
250 MB
0 B
Temperature
100 °C
75 °C
50 °C
25 °C
0 °C
Power Draw
40 W
30 W
20 W
10 W
0 W
Fan Speed %
Graphics Clock Speed
15 MHz
10 MHz
5 MHz
0 Hz
Video Clock Speed
600 MHz
400 MHz
200 MHz
0 Hz
SM Clock Speed
Memory Clock Speed
30 MHz
20 MHz
10 MHz

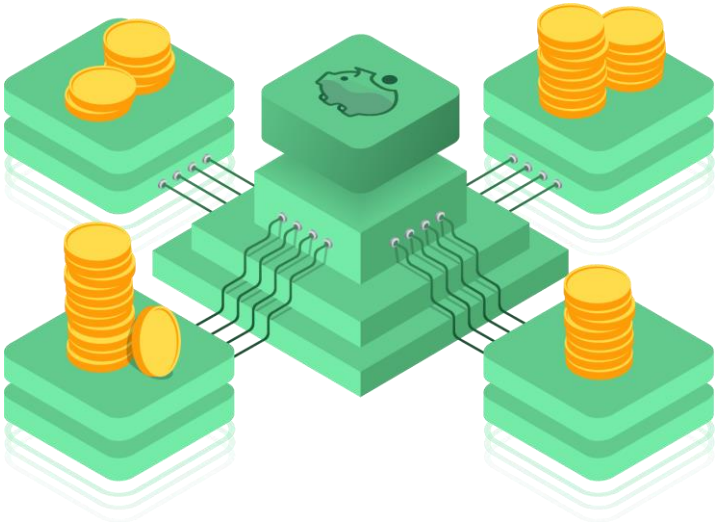

Home › Dashboards › Azure Managed Prometheus › NVIDIA DCGM Exporter Dashboard
Add
Last 15 minutes
Hostname
aks-nc24ads-12229106-vmss000000
GPU_I_ID
All
GPU Temperature
GPU Avg. Temp
35.6 °C
GPU Power Usage
GPU Power Total
16.4 kW
GPU Framebuffer Mem Used
Tensor Core Utilization
GPU SM Clocks
GPU Utilization

Visualize GPU costs

OpenCost

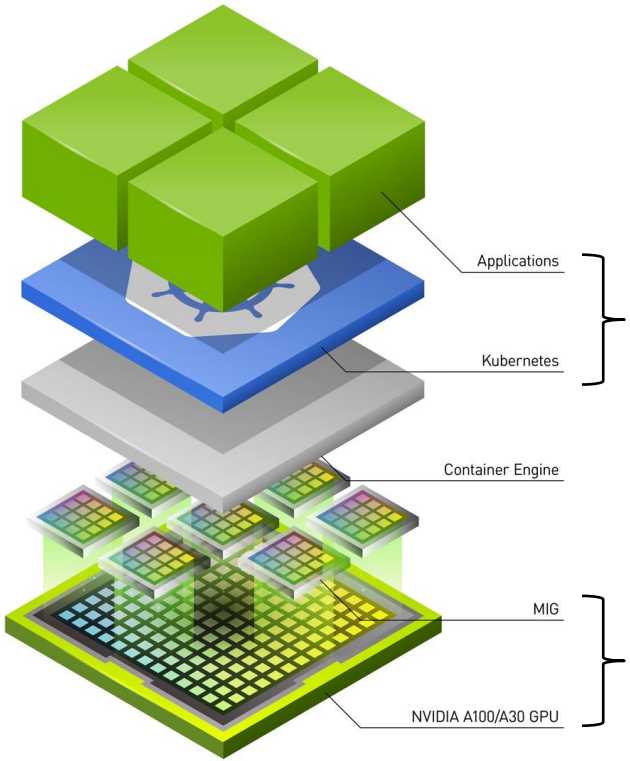
- OSS, vendor neutral project
- OpenCost Spec standard for measuring, reporting, and allocating costs across clouds
- Visualize in near real time

🎉 now a CNCF Incubating project 🎉
https://t.co/mywz7AKPTI



OpenCost

Optimize GPU costs

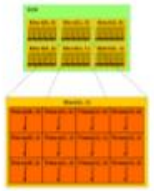


Rightsize your apps and infrastructure

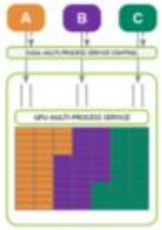
Divide and conquer via partitioning

NVIDIA

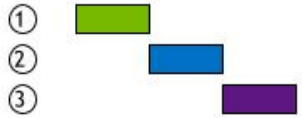
NVIDIA partitioning techniques



Single Process in CUDA

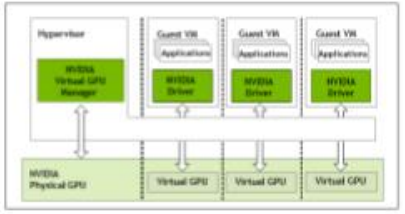


Multi-Process with CUDA MPS



Time-slicing

MIG



Virtualization with vGPU

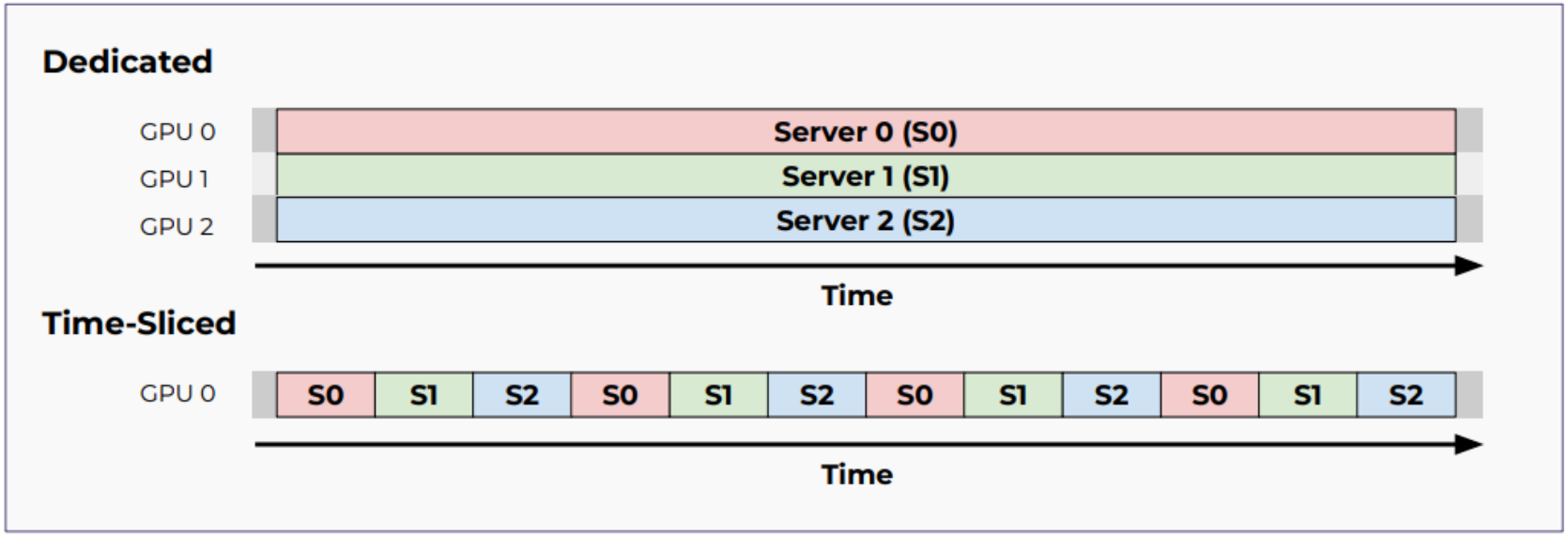
Application level

(using the CUDA programming model APIs - CUDA streams)

GPU System Software / Hardware
(Mostly transparent to CUDA applications)

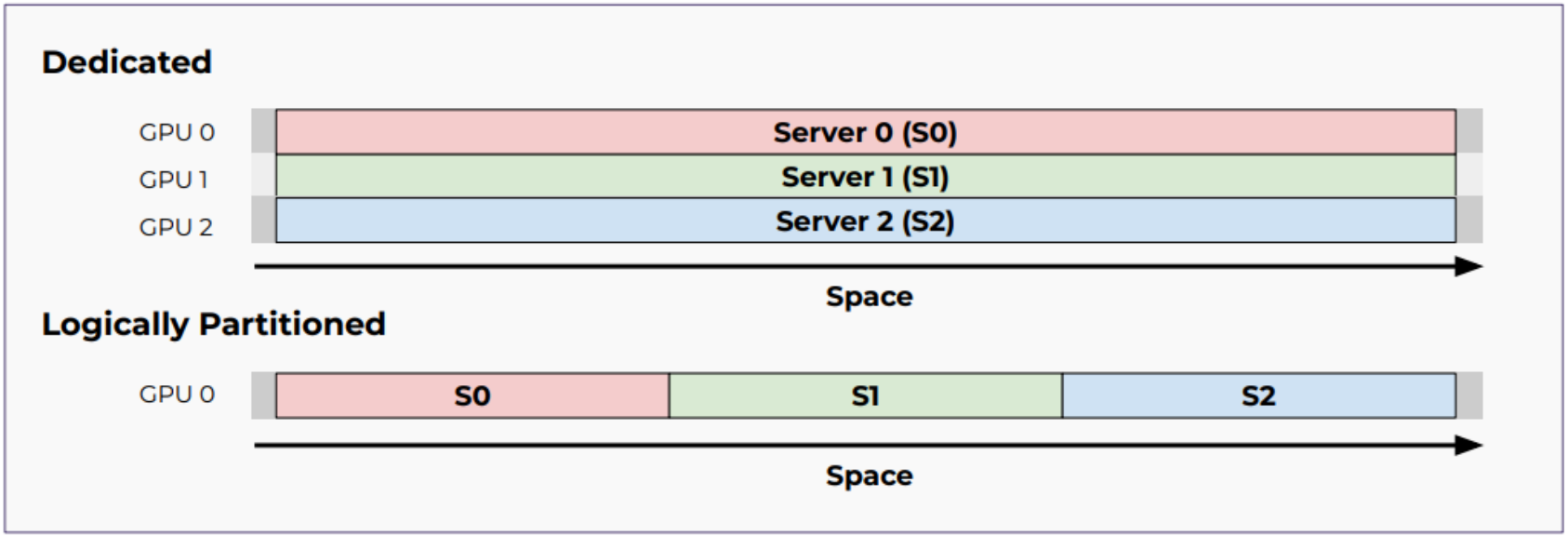
NVIDIA

Time-slicing



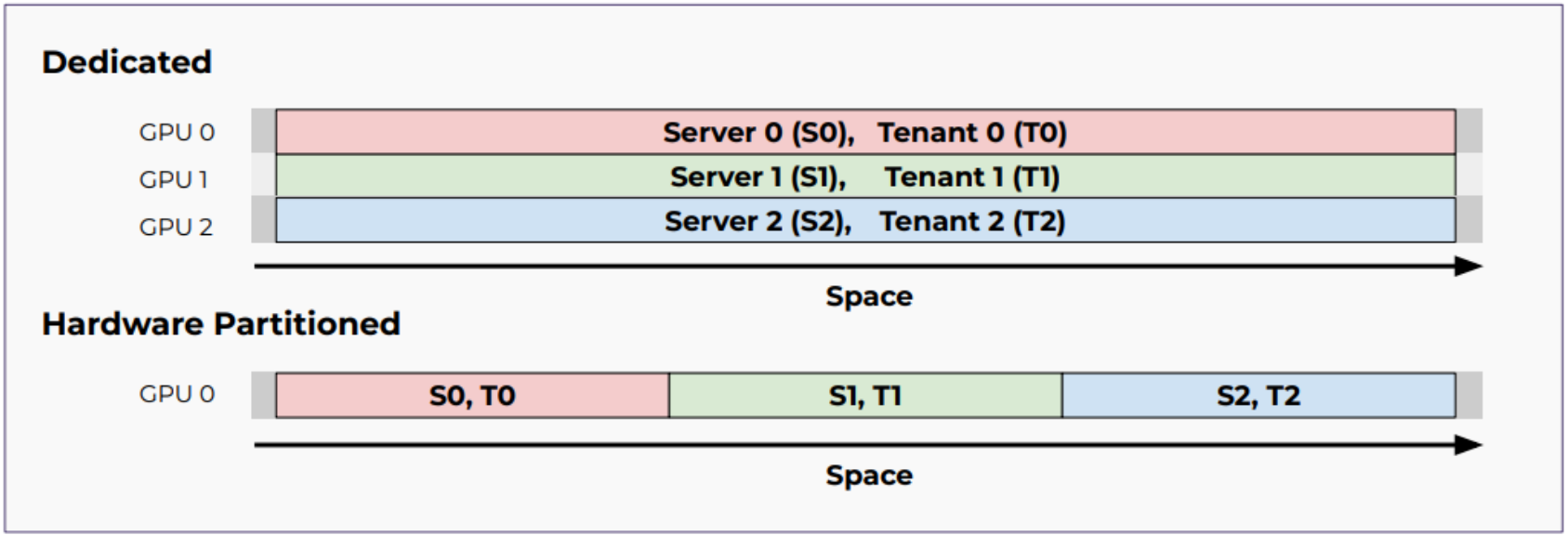
NVIDIA

Multi-Processing Service



NVIDIA

Multi-instance GPUs (MIGs)



NVIDIA

KubeCon | CloudNativeCon

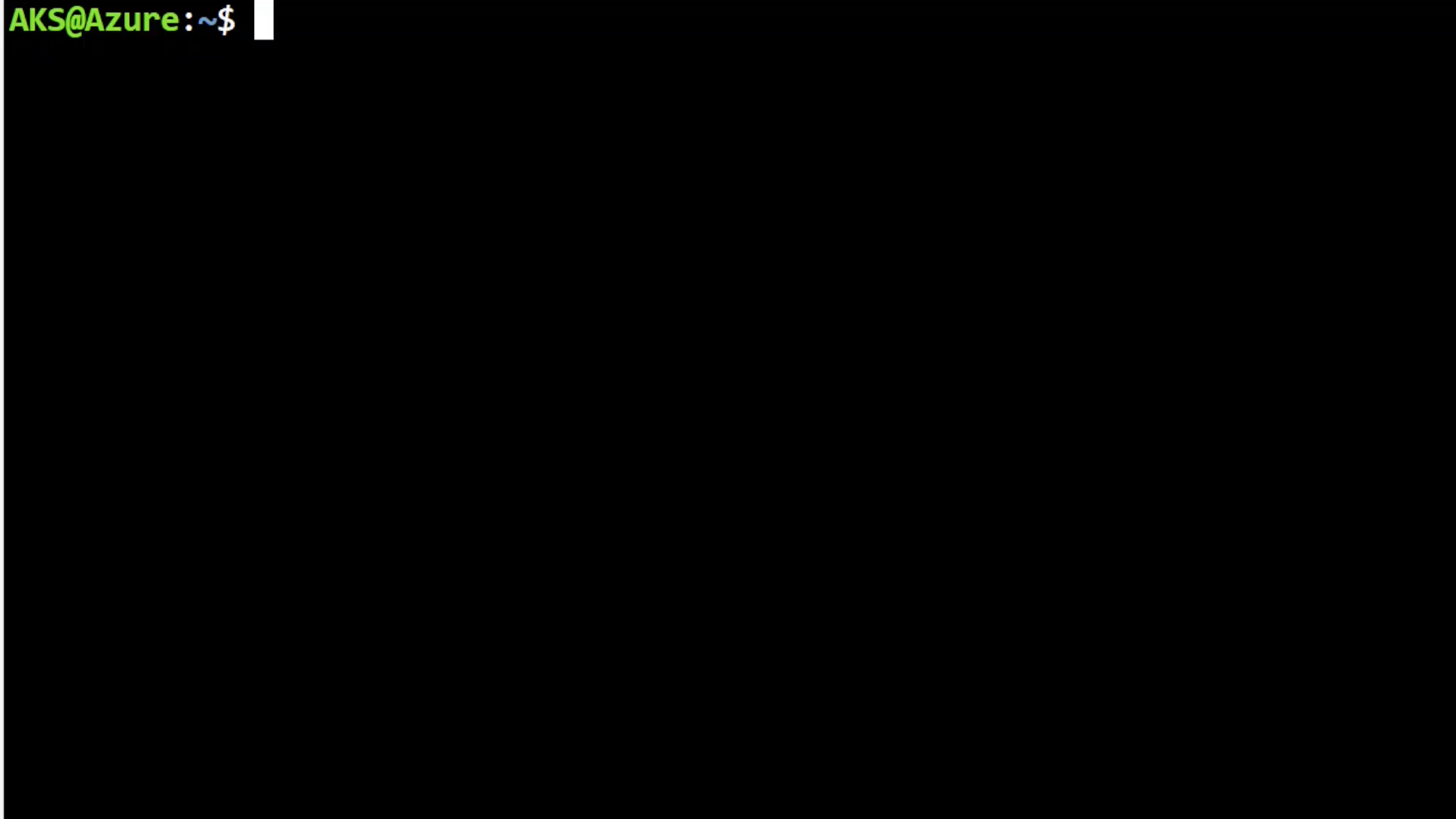
North America 2024

Demo

1. Provision GPU enabled cluster
2. Understand costs in OpenCost
3. Implement timeslicing

AKS@Azure:~$


AKS@Azure:~$


AKS@Azure:~$

Key takeaways

- GPUs are expensive but it can be feasible and economical to run AI workloads on GPUs at scale
- Monitor and visibility is key
  - Without insight into utilization and cost data, we don't know what to optimize and it's hard to keep teams accountable for their GPU spend
- Exclusive access is the default
- Partitioning is just one method of GPU optimization but it can really improve utilization and reduce costs
  - Timeslicing – best for apps that are not latency sensitive or can tolerate jitter
  - MIGs – best running multiple apps in parallel but need resiliency, QoS
  - MPS – best for running apps in parallel but can deal with limited resiliency


KubeCon
CloudNativeCon
North America 2024
Thank you

KubeCon | CloudNativeCon

North America 2024