KubeCon | CloudNativeCon

North America 2024

KubeCon CloudNativeCon
North America 2024

Better Together! GPU, TPU and NIC Topological Alignment with DRA

November, 2024

Patrick Ohly (Intel)
John Belamaric (Google)

Who are we?

Patrick Ohly
Principal Engineer
Intel

John Belamaric
Sr Staff Software Engineer
Google

Dynamic Resource Allocation (DRA) in Kubernetes

- New way of requesting resources
  - Alpha since Kubernetes 1.26
  - Major API changes in 1.31
  - Beta in 1.32
  - GA in ??
- An ***alternative*** to Device Plugin "count-based" interface
  - `nvidia.com/gpu: 2`
- Provides a much richer API for requesting / configuring resources
- Inspired by the persistent volume API

Dynamic Resource Allocation (DRA) in Four Parts

Part 1: New Kubernetes API to **describe** devices (`ResourceSlice`):

This device is an nvidia.com/gpu, its product ID is A100-SXM4-40GB, it has 40Gi of memory, and 3456 FP64 cores.

Part 2: New Kubernetes API to **request** devices (`ResourceClaim`):

I need an nvidia.com/gpu with at least 30Gi of memory and at least 3000 FP64 cores.

Part 3: Updated scheduler to **match** requests to devices.

Part 4: New Kubelet API to **actuate** the scheduler’s decisions.

DRA overcomes the limitations of device plugins

Can **subdivide** large devices

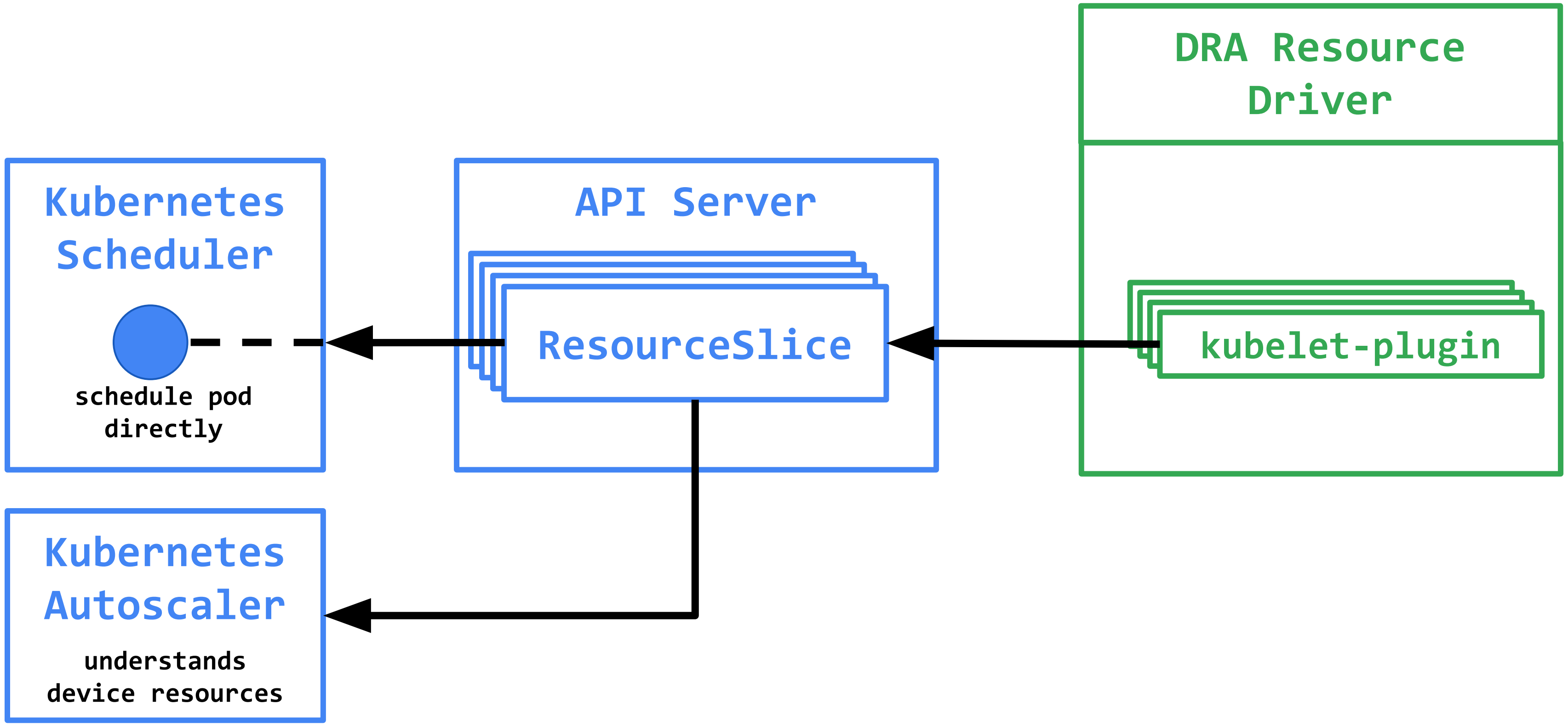
Can **configure** devices individually

Can **share** GPUs in the same node for diverse workloads

Foundational for new functionality:

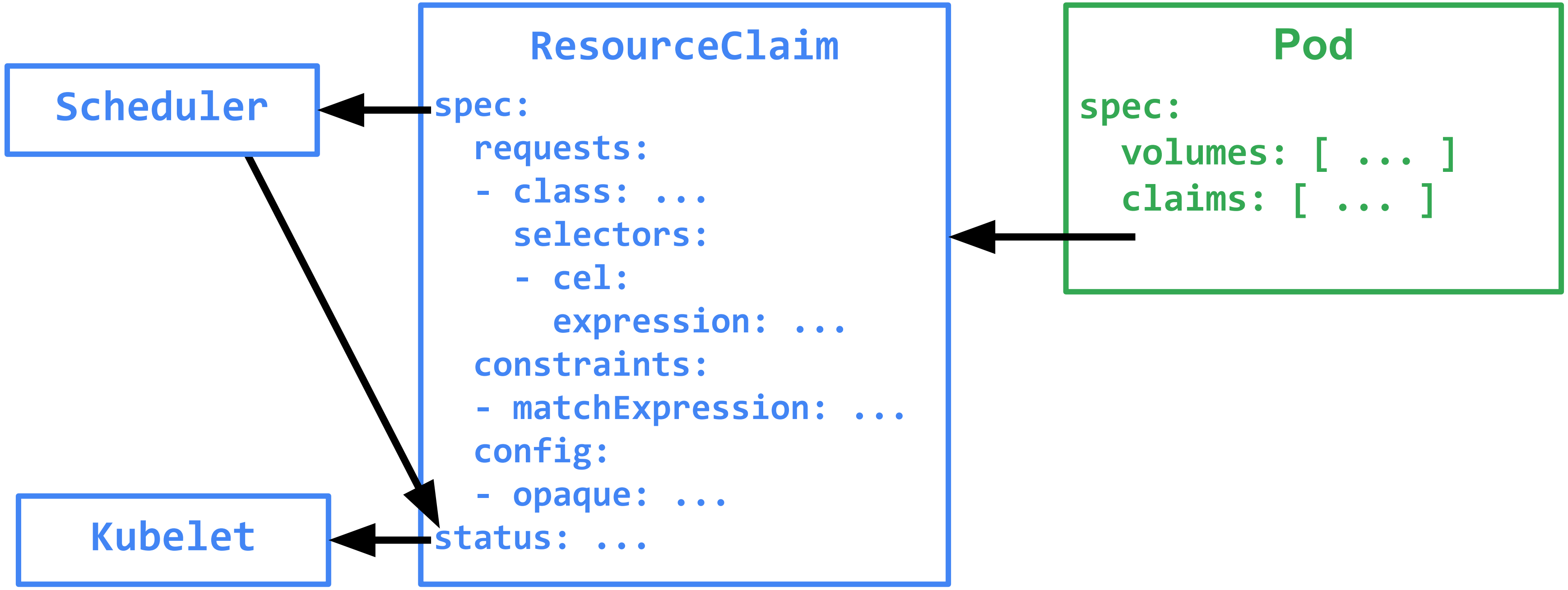
- ***Alignment of multiple, independent devices (GPU and NIC alignment on PCIe)***
- Workload-specific accelerator sharing configuration
- Dynamic MIG and TPU
- Consumption of multiple associated devices as a unit

DRA: Advertising resources



DRA: Requesting resources

KubeCon CloudNativeCon North America 2024

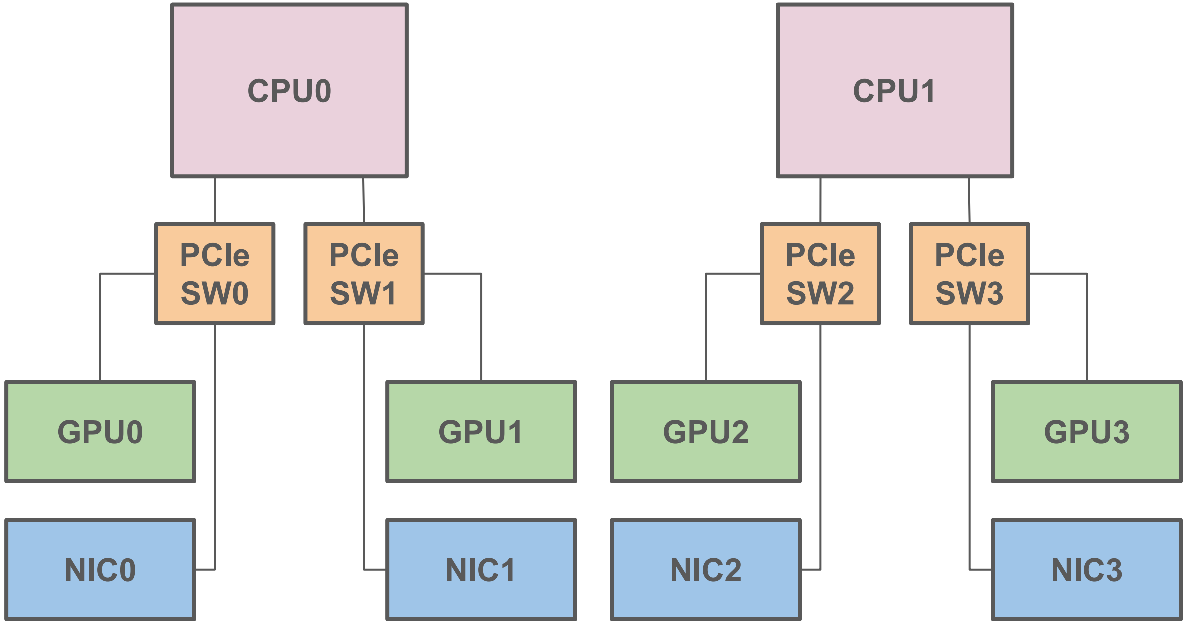


```
spec:
  requests:
  - class: ...
    selectors:
    - cel:
        expression: ...
  constraints:
  - matchExpression: ...
  config:
  - opaque: ...
status: ...
```

```
spec:
  volumes: [ ... ]
  claims: [ ... ]
```

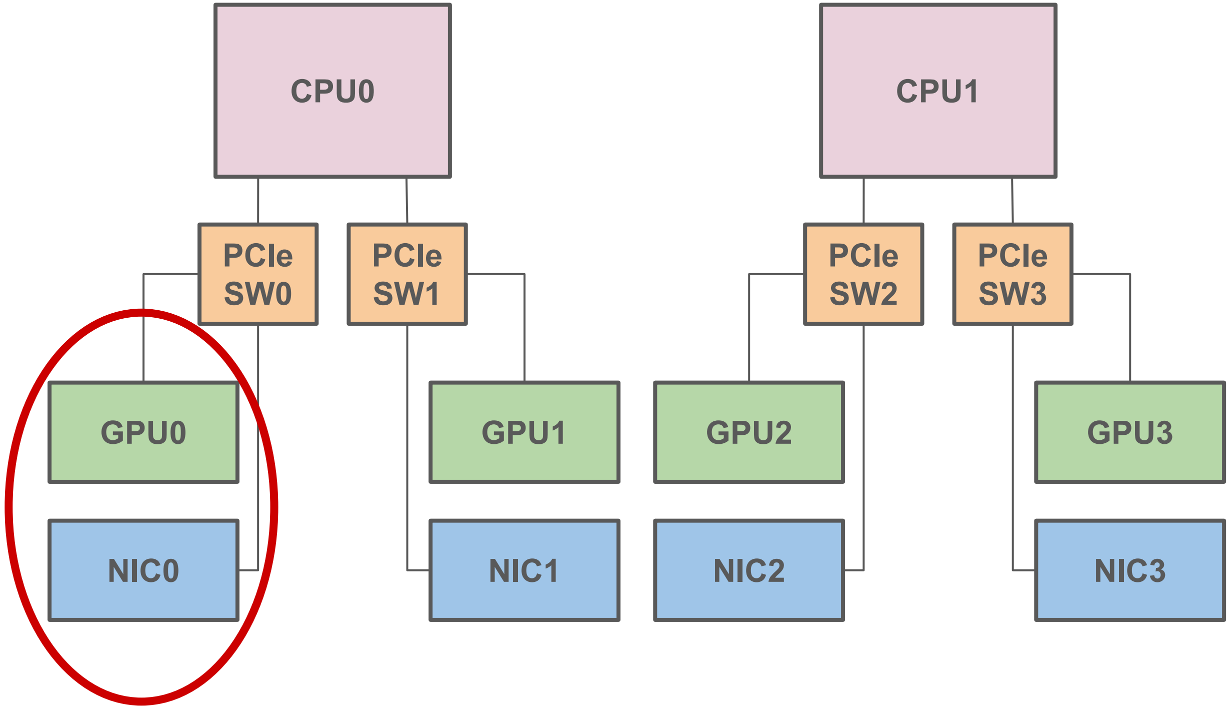
Device Plugin vs DRA for GPU / NIC Topology

Simplified Node Topology - GPU / NIC



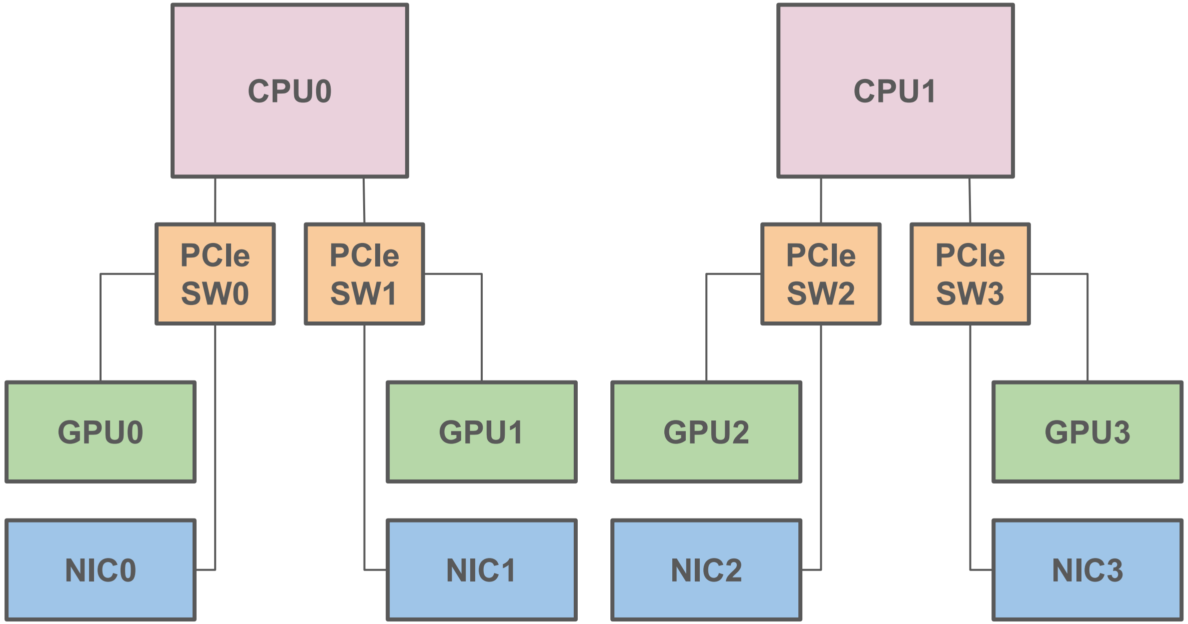
- 2 CPU
- 4 GPU
- 4 NICs
- Internal topology determines performance

Simplified Node Topology - GPU / NIC

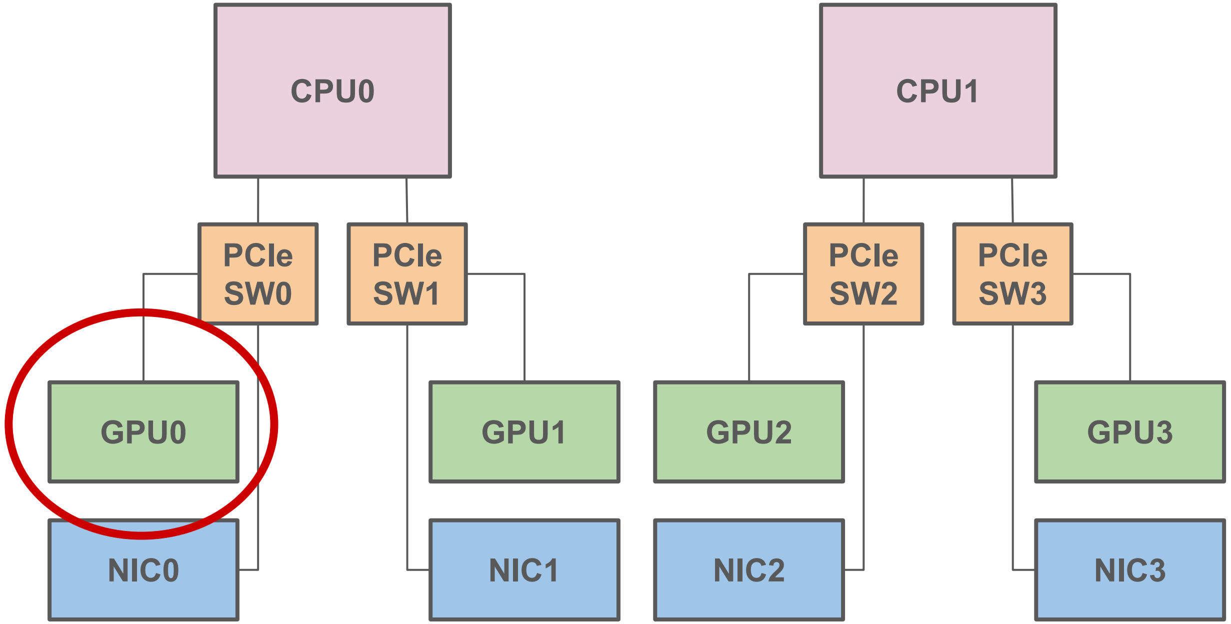


- 2 CPU
- 4 GPU
- 4 NICs
- Internal topology determines performance
- GPUDirect with GPU and NIC are on same root complex

Allocating Devices with Device Plugin

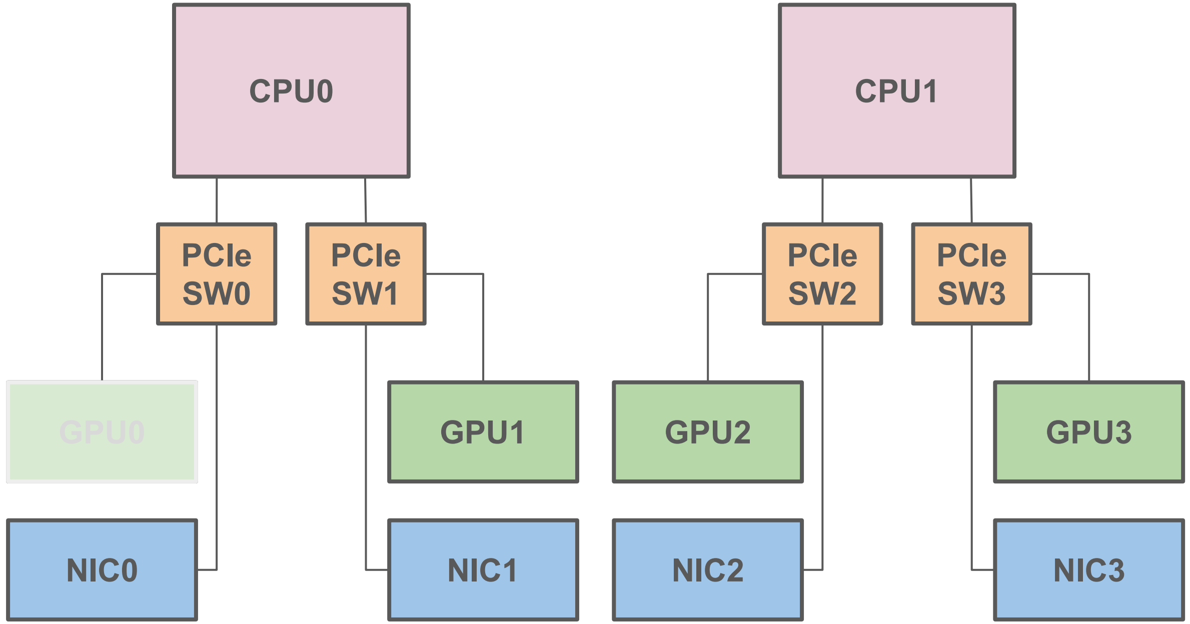


Allocating Devices with Device Plugin



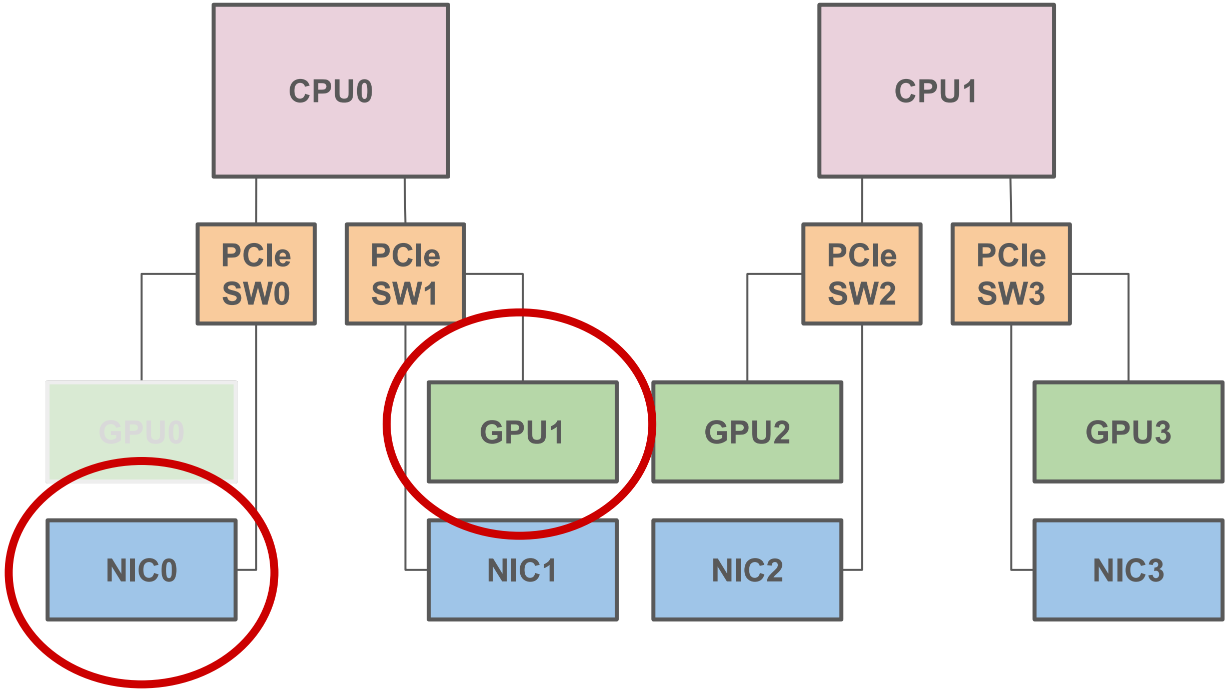
```
resources:
  limits:
    nvidia.com/gpu: 1
```

Allocating Devices with Device Plugin



```
resources:
  limits:
    nvidia.com/gpu: 1
```

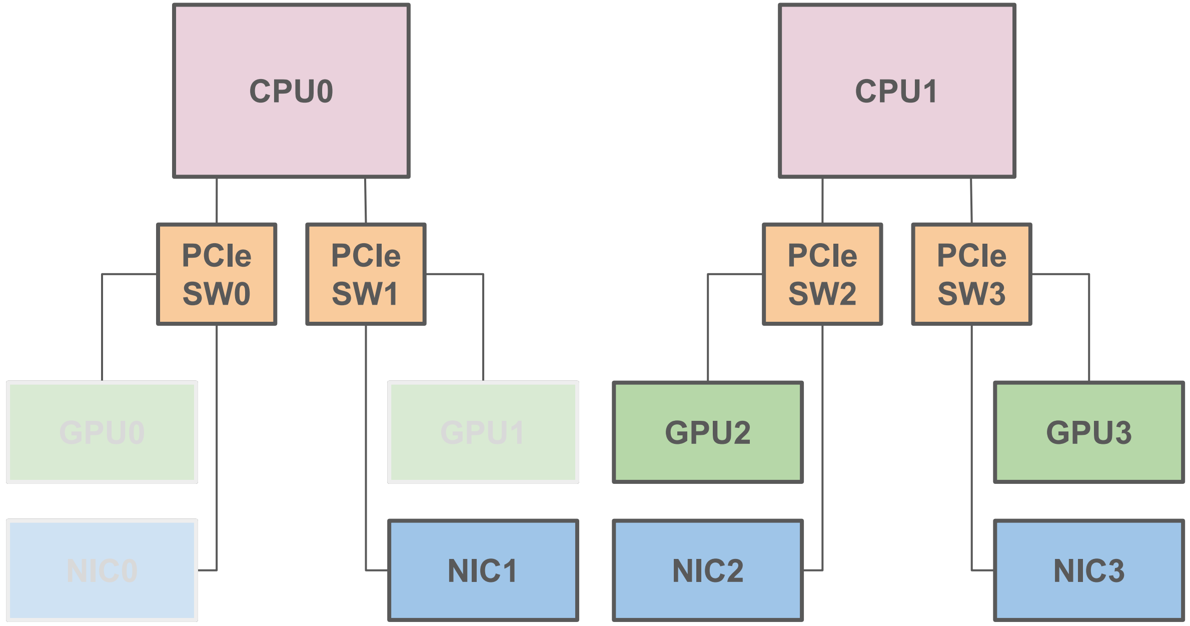
Allocating Devices with Device Plugin



```
resources:
  limits:
    nvidia.com/gpu: 1
```

```
resources:
  limits:
    nvidia.com/gpu: 1
    rdma/rdma_a: 1
```

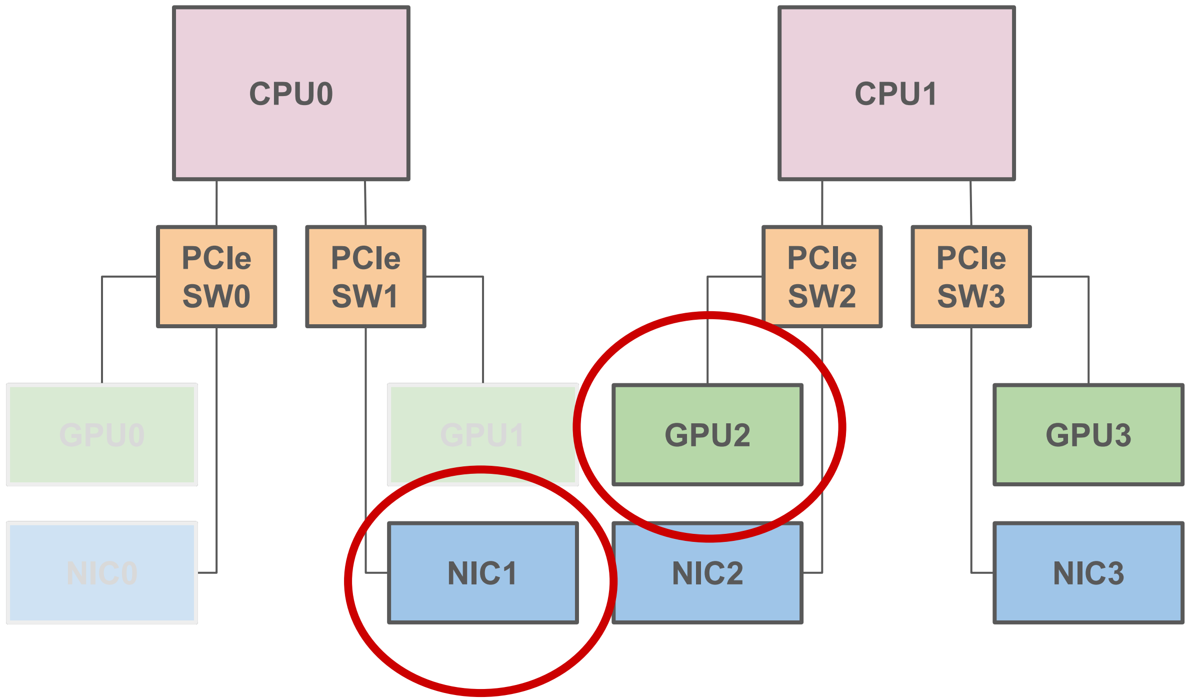
Allocating Devices with Device Plugin



```
resources:
  limits:
    nvidia.com/gpu: 1
```

```
resources:
  limits:
    nvidia.com/gpu: 1
    rdma/rdma_a: 1
```

Allocating Devices with Device Plugin



```
resources:
  limits:
    nvidia.com/gpu: 1
```

```
resources:
  limits:
    nvidia.com/gpu: 1
    rdma/rdma_a: 1
```

```
resources:
  limits:
    nvidia.com/gpu: 1
    rdma/rdma_a: 1
```

Dynamic Resource Allocation (DRA) in Kubernetes

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: big-gpu-with-aligned-nic
spec:
  devices:
    requests:
    - name: gpu
      deviceClassName: gpu.nvidia.com
      selectors:
      - cel:
          expression: "device.capacity['memory'].compareTo(quantity('80Gi')) >= 0"
```

Give me a GPU with at least 80GB of memory

Dynamic Resource Allocation (DRA) in Kubernetes

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: big-gpu-with-aligned-nic
spec:
  devices:
    requests:
    - name: gpu
      deviceClassName: gpu.nvidia.com
      selectors:
      - cel:
          expression: "device.capacity['memory'].compareTo(quantity('80Gi')) >= 0"

    - name: nic
      deviceClassName: rdma.nvidia.com
      selectors:
      - cel:
          expression: "device.attribute['sriovType'] == 'vf'"
```

Give me a GPU with at least 80GB of memory

Together with an RDMA virtual function

Dynamic Resource Allocation (DRA) in Kubernetes

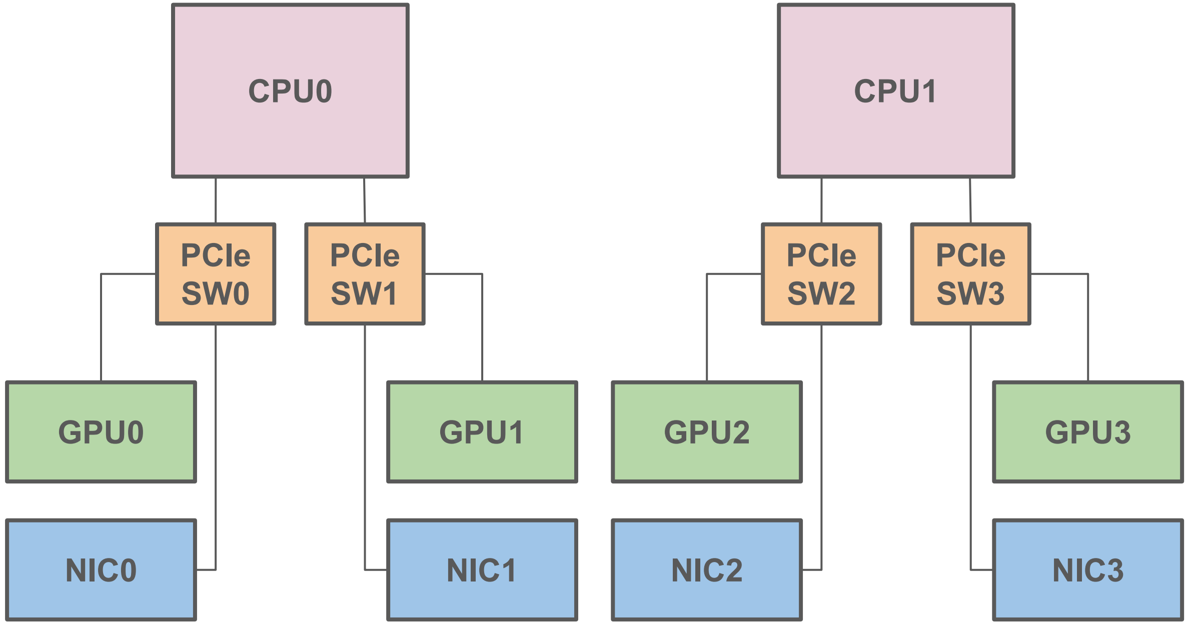
```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: big-gpu-with-aligned-nic
spec:
  devices:
    requests:
    - name: gpu
      deviceClassName: gpu.nvidia.com
      selectors:
      - cel:
          expression: "device.capacity['memory'].compareTo(quantity('80Gi')) >= 0"

    - name: nic
      deviceClassName: rdma.nvidia.com
      selectors:
      - cel:
          expression: "device.attribute['sriovType'] == 'vf'"

    constraints:
    - requestNames: ["gpu", "nic"]
      matchAttribute: k8s.io/pcieRoot
```

Give me a GPU with at least 80GB of memory

Together with an RDMA virtual function

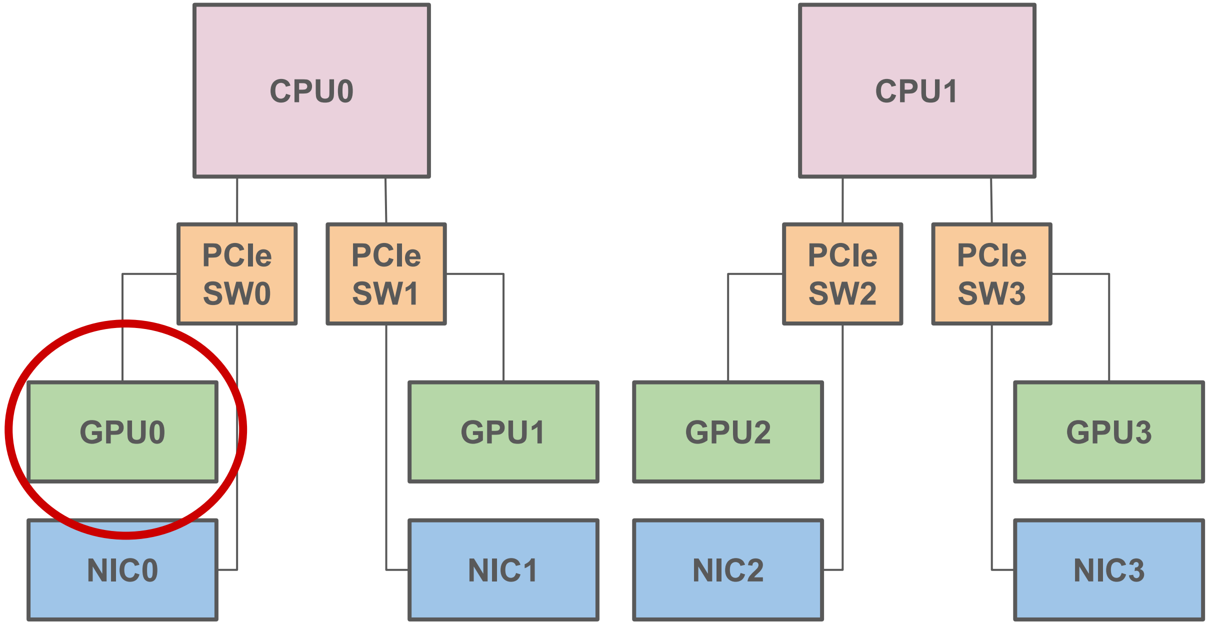
Make sure the GPU and NIC are aligned on the same PCIe root complex

Allocating Devices with DRA



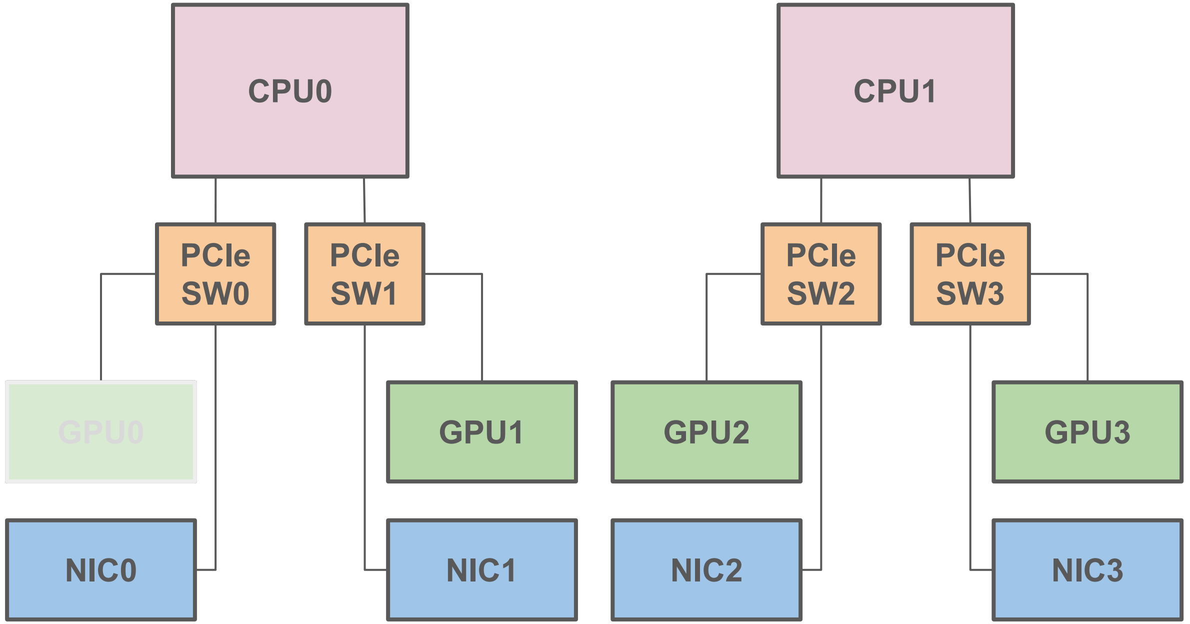
Allocating Devices with DRA

```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
```

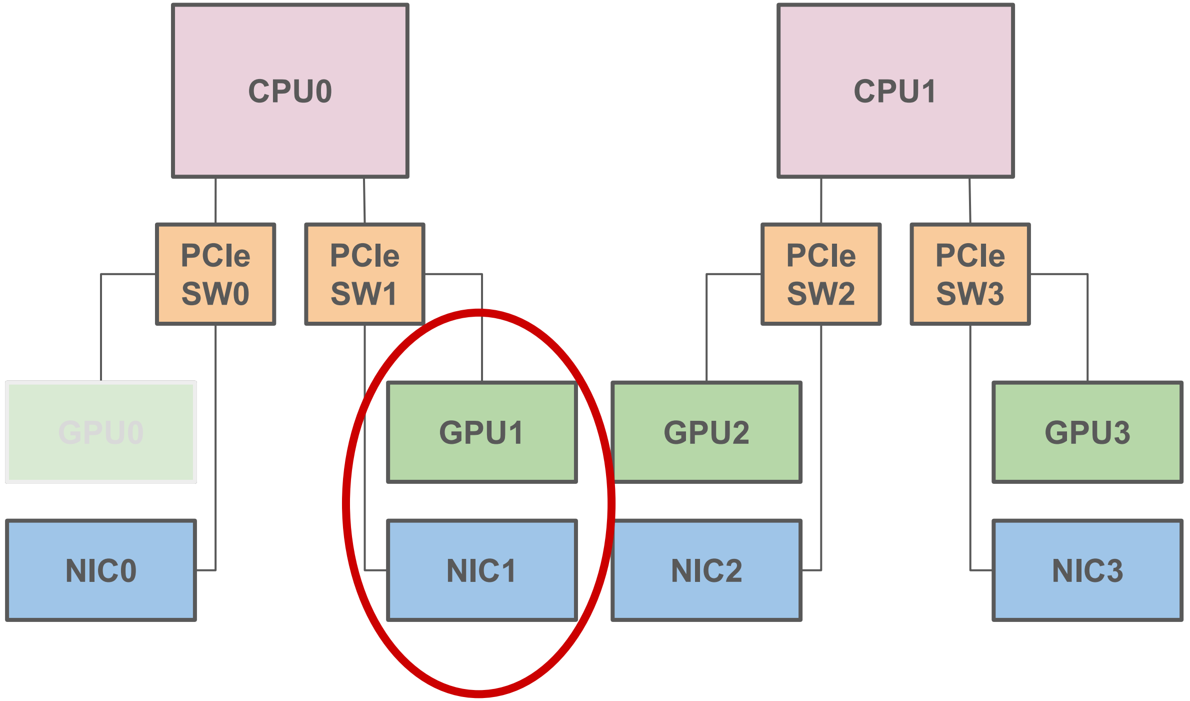


Allocating Devices with DRA

```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
```



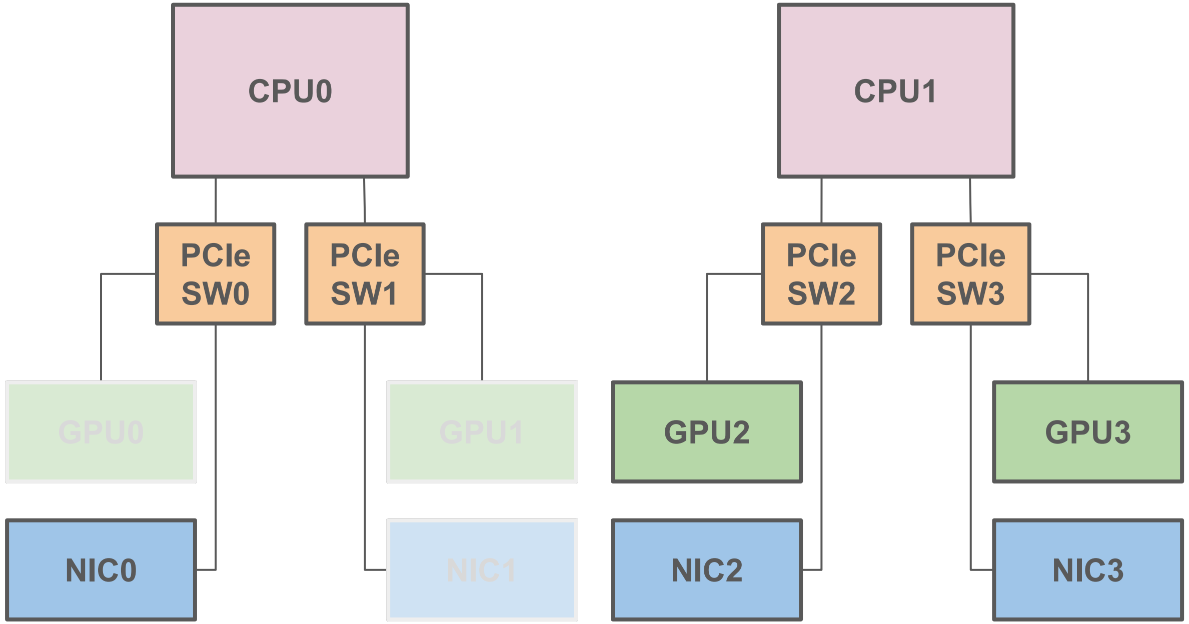
Allocating Devices with DRA



```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
```

```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
- name: nic
  deviceClassName: rdma.nvidia.com

constraints:
- requestNames: ["gpu", "nic"]
  matchAttribute: k8s.io/pcieRoot
```

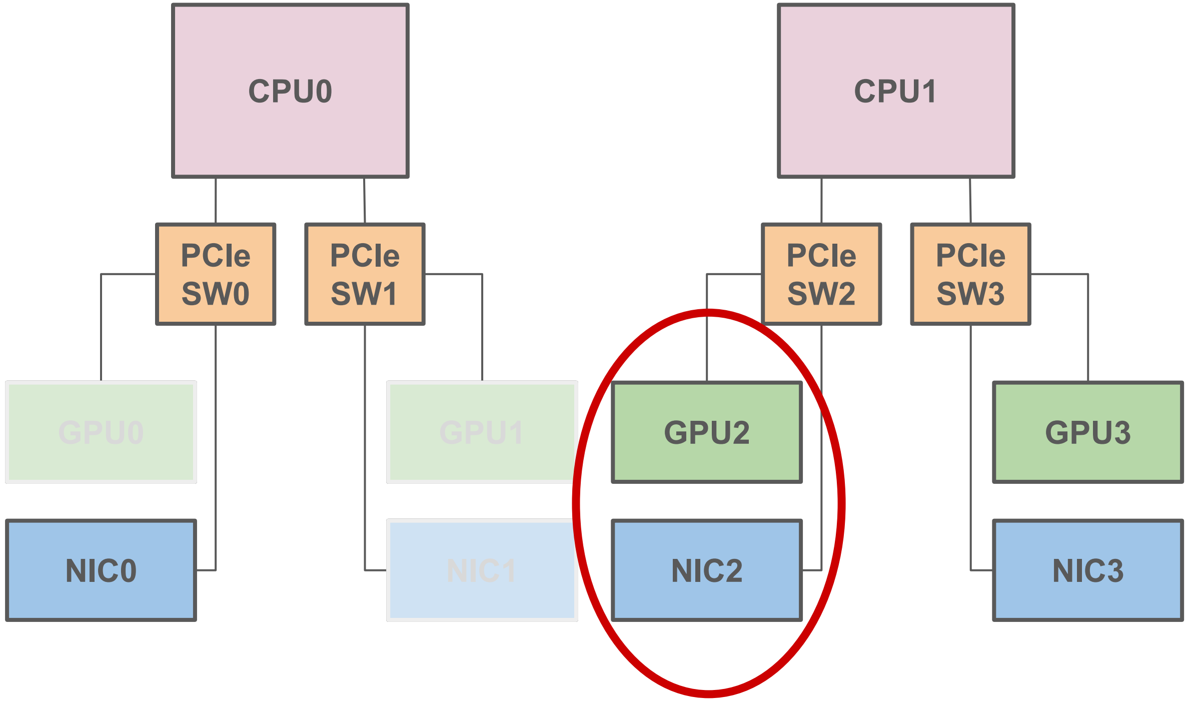
Allocating Devices with Device Plugin



```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
```

```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
- name: nic
  deviceClassName: rdma.nvidia.com

constraints:
- requestNames: ["gpu", "nic"]
  matchAttribute: k8s.io/pcieRoot
```

Allocating Devices with Device Plugin



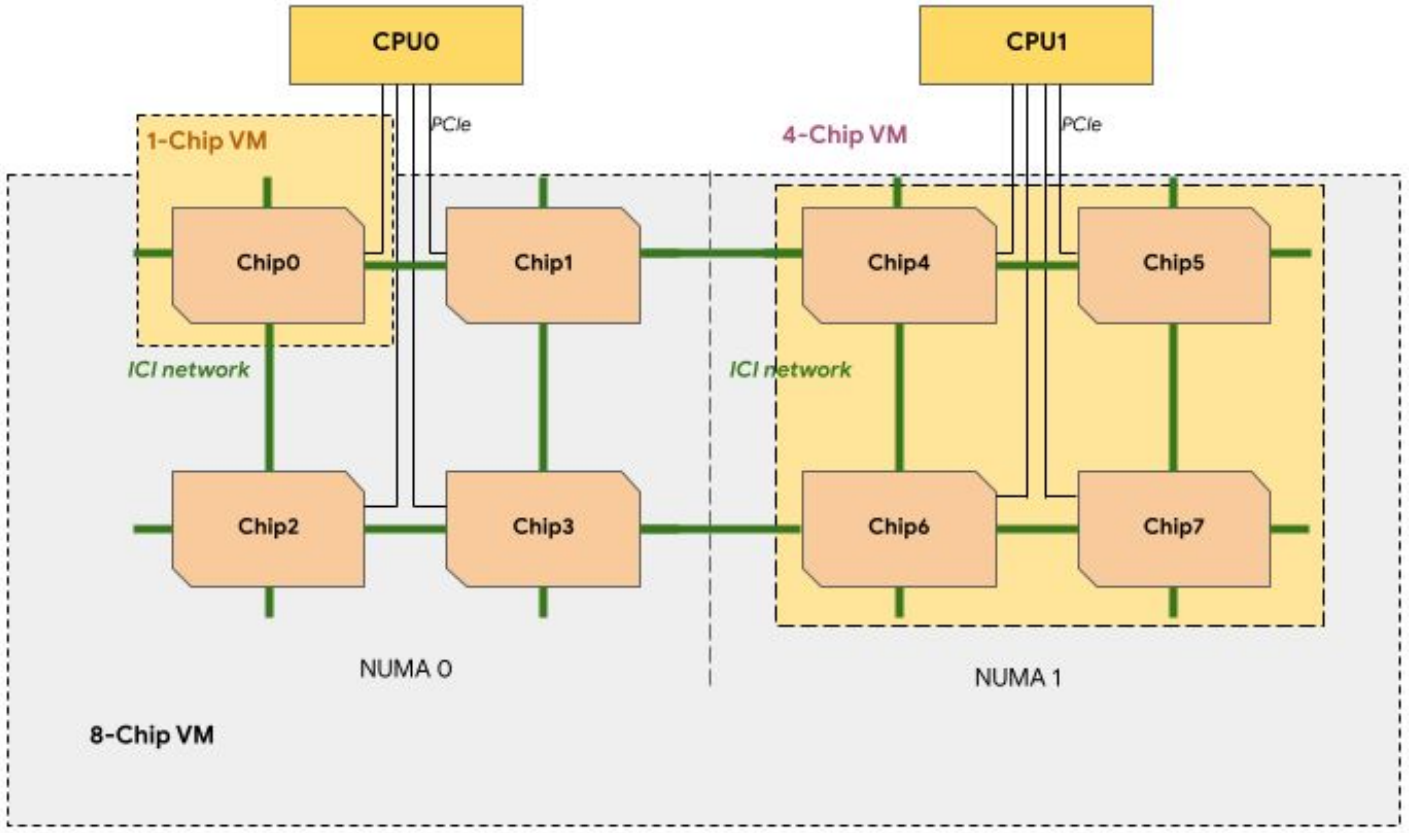
```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
```

```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
- name: nic
  deviceClassName: rdma.nvidia.com

constraints:
- requestNames: ["gpu", "nic"]
  matchAttribute: k8s.io/pcieRoot
```

```
requests:
- name: gpu
  deviceClassName: gpu.nvidia.com
- name: nic
  deviceClassName: rdma.nvidia.com

constraints:
- requestNames: ["gpu", "nic"]
  matchAttribute: k8s.io/pcieRoot
```

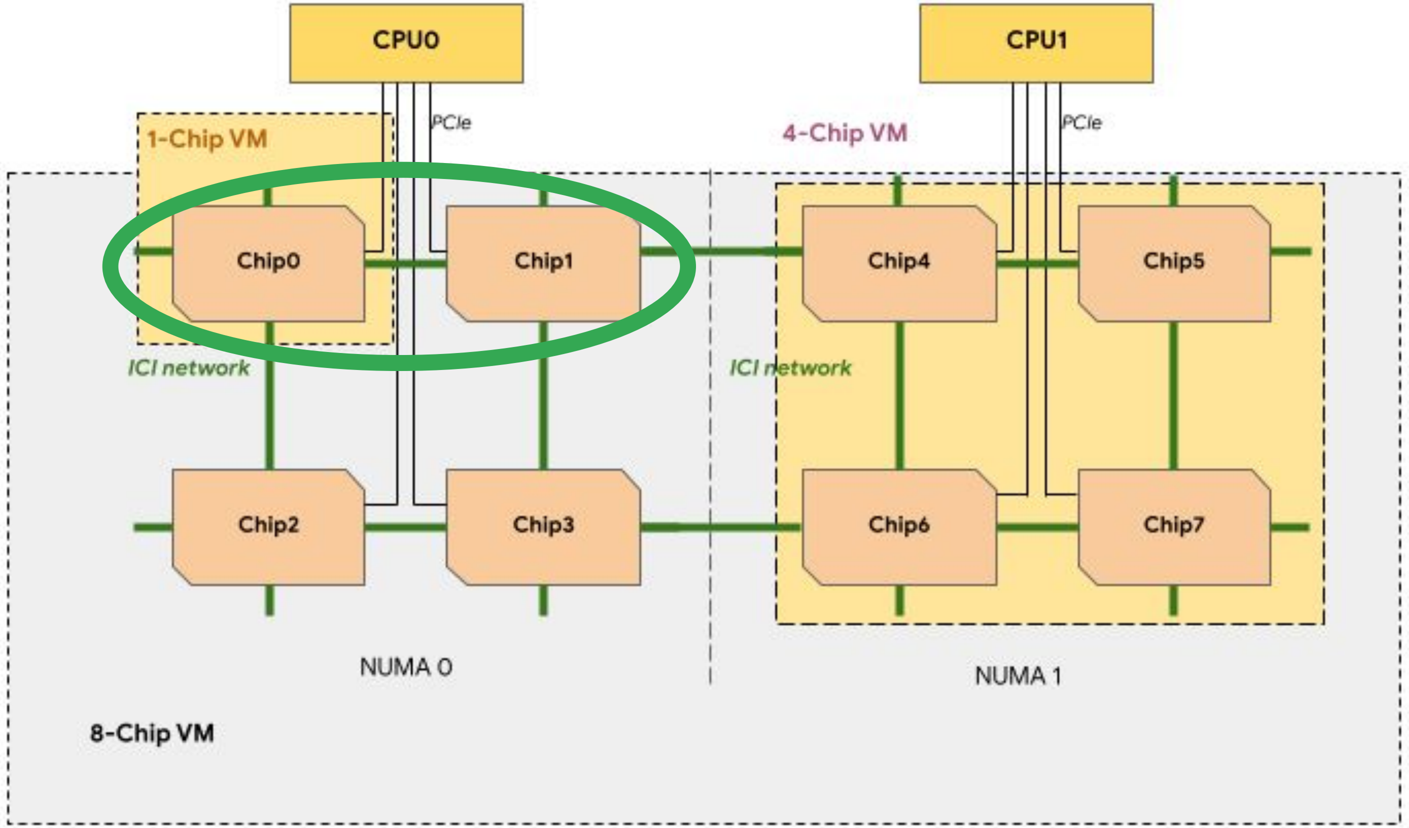
What about TPU?

TPU Host v5e Topology



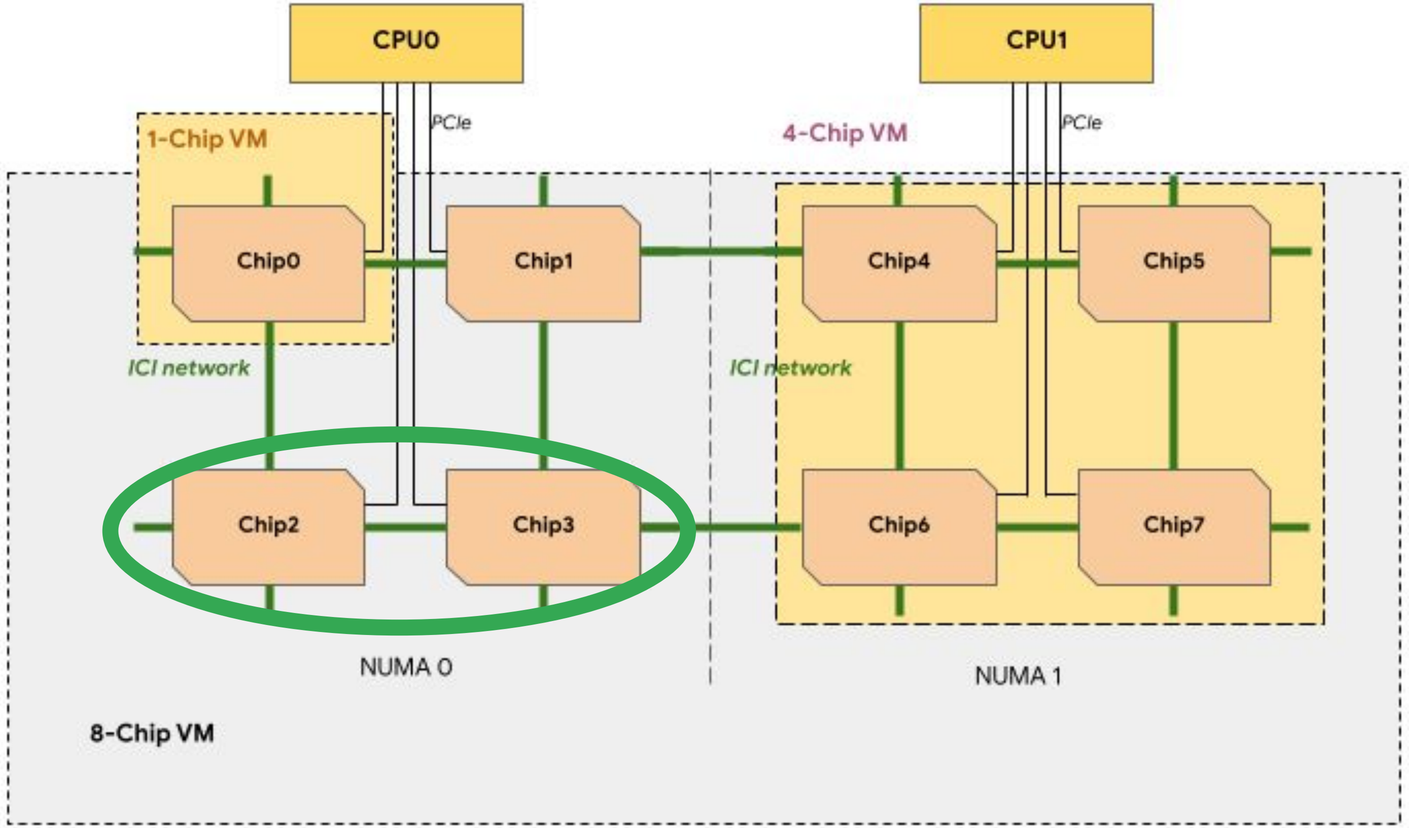
Source: https://cloud.google.com/tpu/docs/v5e

TPU Host v5e Topology



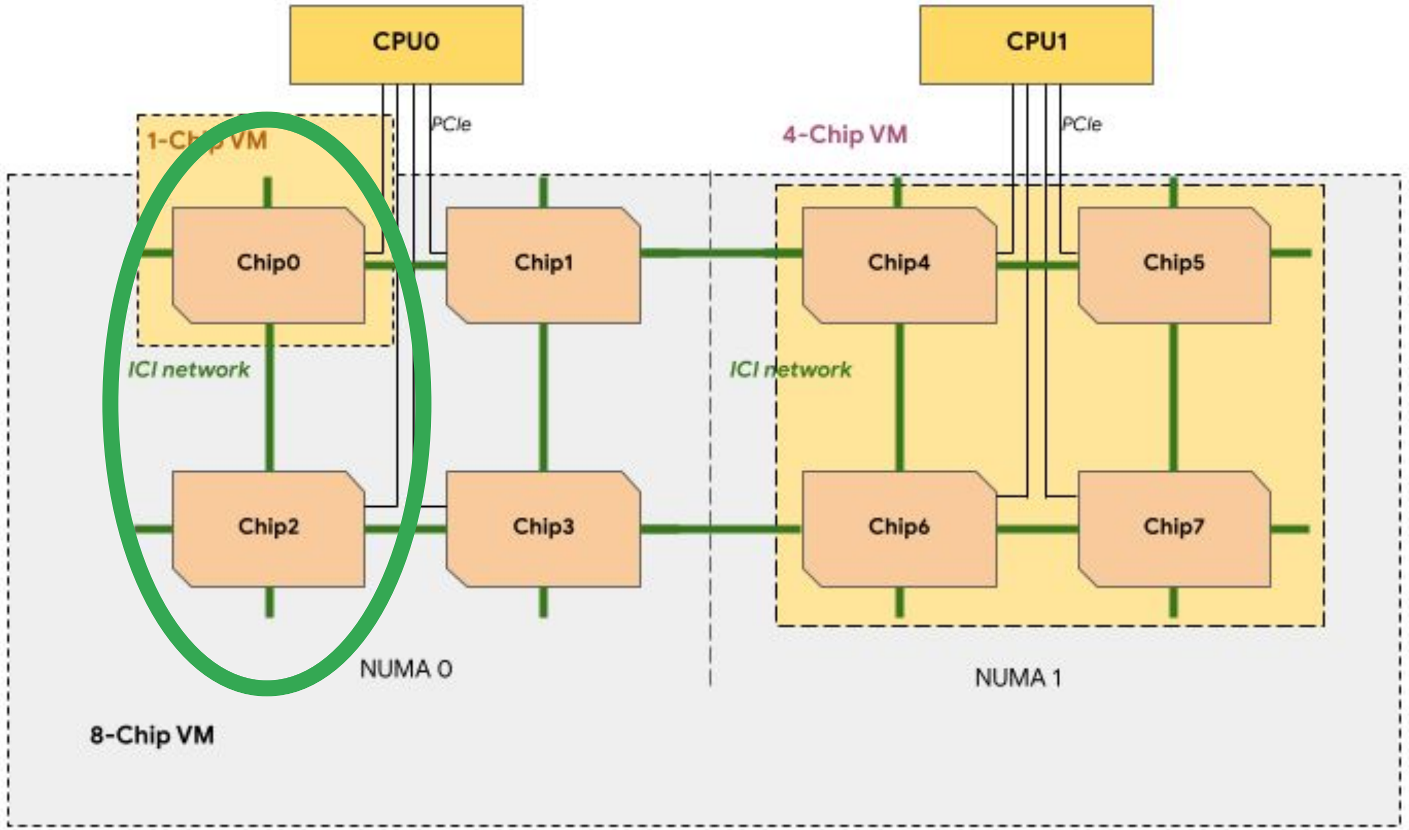
Source: https://cloud.google.com/tpu/docs/v5e

TPU Host v5e Topology



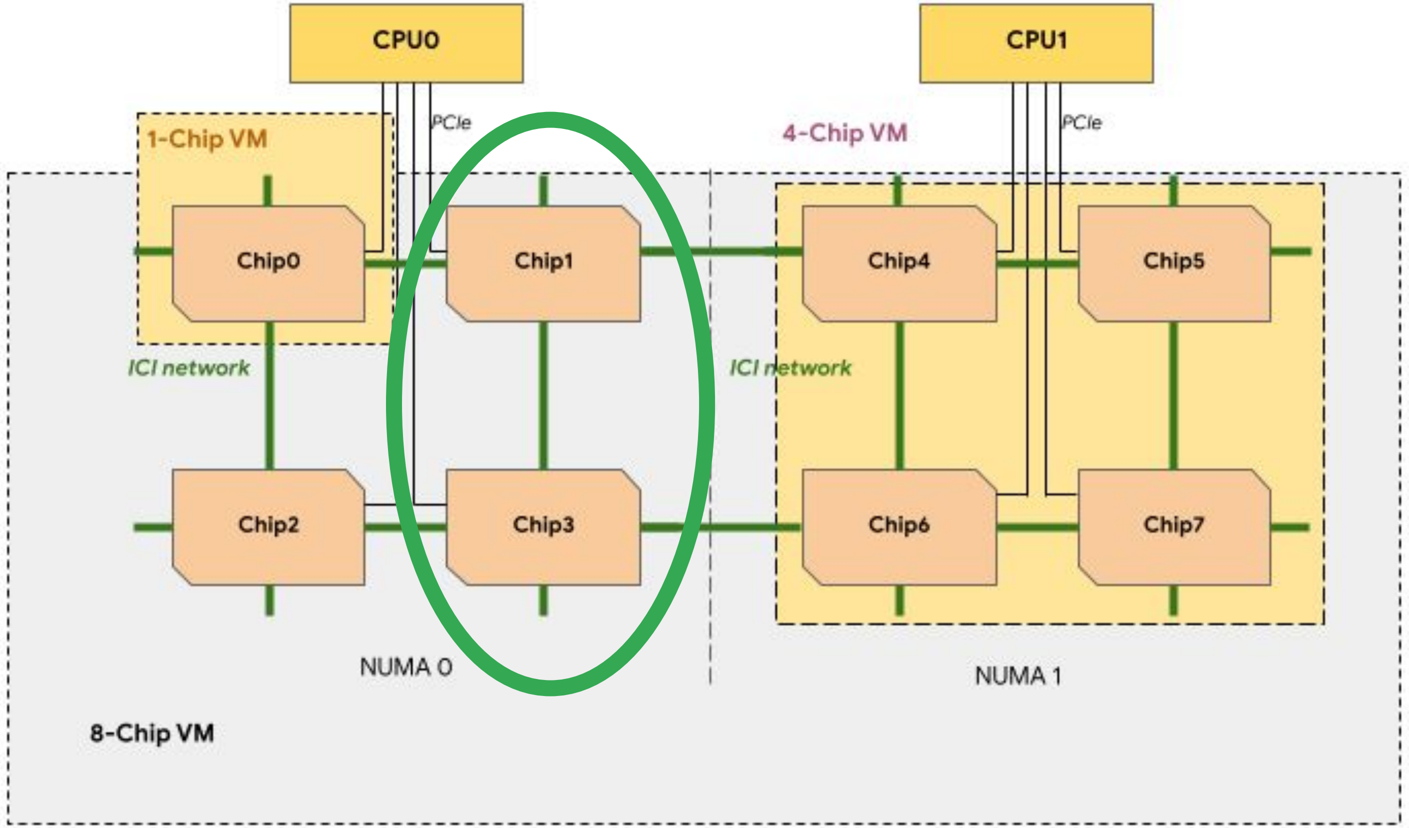
Source: https://cloud.google.com/tpu/docs/v5e

TPU Host v5e Topology



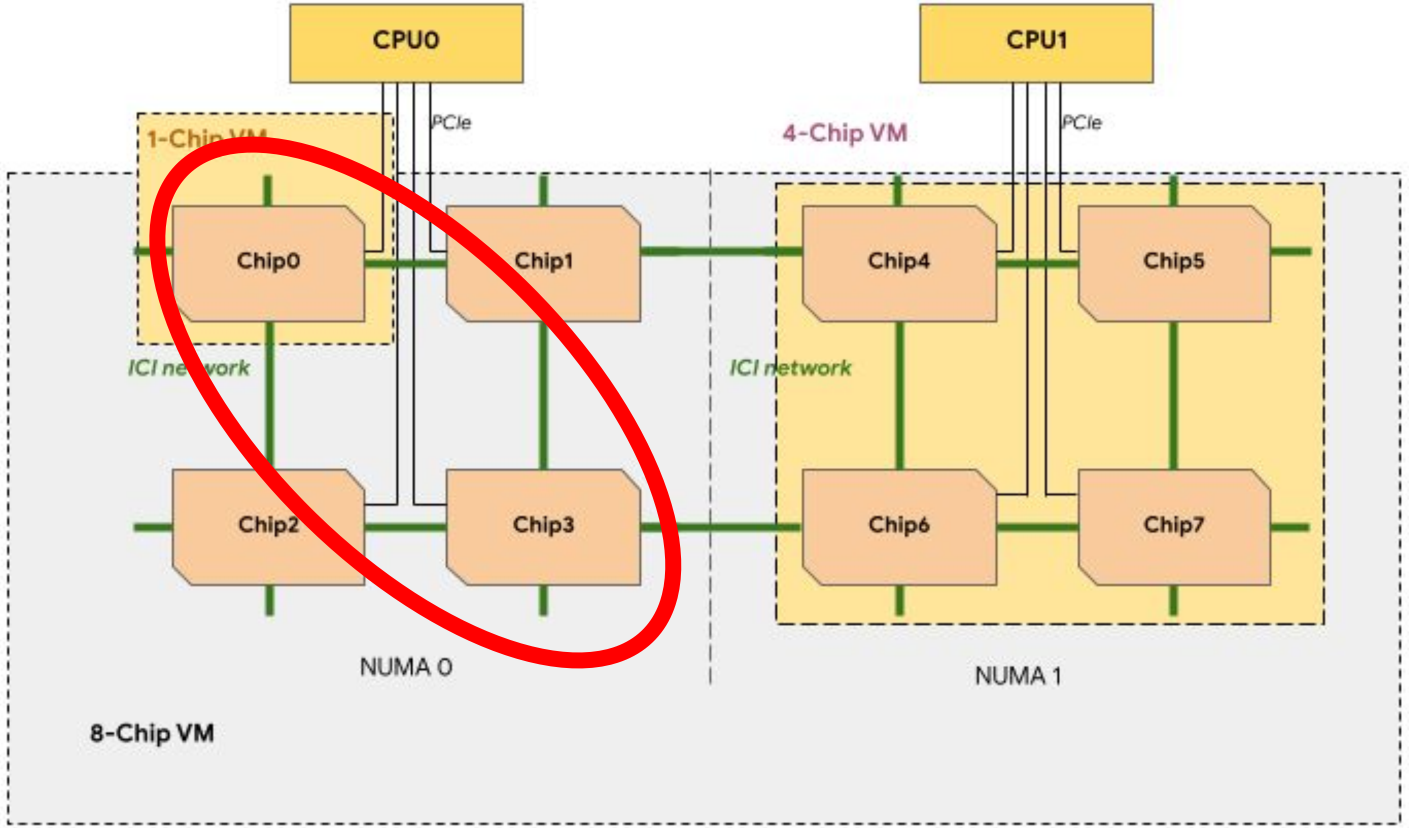
Source: https://cloud.google.com/tpu/docs/v5e

TPU Host v5e Topology



Source: https://cloud.google.com/tpu/docs/v5e

TPU Host v5e Topology



Source: https://cloud.google.com/tpu/docs/v5e

TPU and Match Attribute

- **`matchAttribute`** constraints provides exact match semantics
- Can achieve ***partial*** support for various TPU topologies with **`matchAttribute`**

```
name: tpu0
attributes:
  2x1:
    string: west
  1x2:
    string: north
```

```
name: tpu1
attributes:
  2x1:
    string: east
  1x2:
    string: north
```

```
name: tpu2
attributes:
  2x1:
    string: west
  1x2:
    string: south
```

```
name: tpu3
attributes:
  2x1:
    string: east
  1x2:
    string: south
```

- User needs to ask for either **2x1** or **1x2**, not just **2**
- Could lead to scheduling failures

Solving TPU Topology Allocations

`matchExpression`

- Proposed feature to use a CEL expression for a match constraint
- Also helps for AWS Neuron cores, which need ***sequential*** chips

All this is pretty hard on the user

- Is there something that can hide the complexity?

Solving TPU Topology Allocations

`matchExpression`

- Proposed feature to use a CEL expression for a match constraint
- Also helps for AWS Neuron cores, which need ***sequential*** chips

All this is pretty hard on the user

- Is there something that can hide the complexity?

YES!

Partitionable Devices

- DRA: Add support for partitionable devices · Issue #4815
- Alpha feature targeting 1.33
- Driver publishes valid topologies, consumed as a single device
- Much simpler for users - request single “device”
    - By size: tpu1x, tpu2x, tpu4x, tpu8x
    - Only valid topologies can be allocated
    - Underlying chips are counted under the hood
- Will support many vendor partitioning schemes:
    - AWS Neuron Cores
    - Google TPUs
    - Intel® Max Series
    - NVIDIA Multi Instance GPU (MIG)

Other Talks, Feedback, and Questions

Kubernetes WG Device Management - Advancing K8s Support for GPUs
John Belamaric, Google; Patrick Ohly, Intel; Kevin Klues, NVIDIA

A Tale of 2 Drivers: GPU Configuration on the Fly Using DRA
Alay Patel (NVIDIA), Varun Ramachandra Sekar US (NVIDIA)

Which GPU Sharing Strategy Is Right for You?
A Comprehensive Benchmark Study Using DRA
Kevin Klues (NVIDIA), Yuan Chen (NVIDIA)

Google Booth Lightning Talk
Deploying DRA for AI Infrastructure - Tech Talk & Ask the Experts Panel
Laura Lorenz (Google), Kevin Klues (NVIDIA), Tim Hockin (Google), John Belamaric (Google)

Friday, 12:50pm - 1:05pm MST | Salt Palace | Google Cloud Booth

Feedback