KubeCon | CloudNativeCon

North America 2024

WG Device Management

Playlist | Charter | Agenda | Mailing List | Zoom | Slack

November, 2024

Patrick Ohly (Intel)
Kevin Klues (NVIDIA)
John Belamaric (Google)

Working Group Co-Chairs

Patrick Ohly
Principal Engineer
Intel

Kevin Klues
Distinguished Engineer
NVIDIA

John Belamaric
Sr Staff Software Engineer
Google

What and Why?

Enable simple and efficient configuration, sharing, and allocation of accelerators and other specialized devices.

- Working Group Device Management
- New working group formed in April 2024, born out of KubeCon EU 2024
- SIGs: Architecture, Autoscaling, Network, Node and Scheduling
- Redefining Kubernetes relationship with hardware
- Accelerators but also other devices like NICs, FPGAs and even network-attached devices.

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 will be Beta in 1.32! 🎉

1.32 DRA Features

- Merged in 1.32
  - Removal of classic DRA
  - Structured parameters (DRA MVP) graduated to **Beta**
  - Faster scheduling (up to 16x)
  - Driver-owned resource claim status (for multi-networking use cases, primarily)
  - Significant progress on autoscaler integration
- Drivers for 1.32 (out-of-tree)
  - Example driver
  - Intel DRA Drivers for GPU, Gaudi and QAT
  - NVIDIA DRA Driver for GPUs and multi-node NVLink
  - CNI DRA Driver (in progress)
  - Google TPU Driver (in progress)

What's Next for DRA?

- **Claim API** - Flexibility in requests? New constraints? New config options? Quota? In place updates? Native resources?
- **Slice API** - New device models? Tracking allocation status for multi-scheduler support? Standardization of attributes? Native resources?
- **Drivers** - Downward API? New drivers? Driver framework features? (for example, annotating devices based on a file with node-level / cluster topology)?
- **Scheduling** - Preemption and priority? Scoring? User-provided scoring functions or hints?
- **Usability** - Automatic support for existing device plugin specs? CEL improvements?

Getting Involved

Join us to help shape how accelerators and other devices are used in Kubernetes.

- Bi-Weekly Meetings
  - Tuesdays 8:30am PST
  - Zoom
  - Agenda
  - Playlist
- Very active Slack channel
- Less active Mailing List
- Community Page

DRA Deep Dive

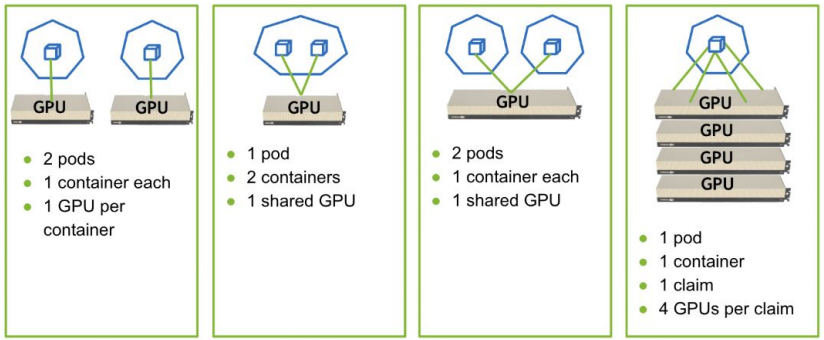
A Case Study with NVIDIA GPUs

Dynamic Resource Allocation (DRA) in Kubernetes

- New way of requesting resources available (as an alpha feature) in Kubernetes 1.26+
- Provides an ***alternative*** to the “count-based” interface of e.g. **nvidia.com/gpu:2**
- Provides a much richer API for requesting / configuring resources
- Inspired by the persistent volume API

Dynamic Resource Allocation (DRA) in Kubernetes

- New way of requesting resources available (as an alpha feature) in Kubernetes 1.26+
- Provides an ***alternative*** to the “count-based” interface of e.g. **nvidia.com/gpu:2**
- Provides a much richer API for requesting / configuring resources
- Inspired by the persistent volume API



DRA overcomes the limitations of device plugins

Can subdivide large devices

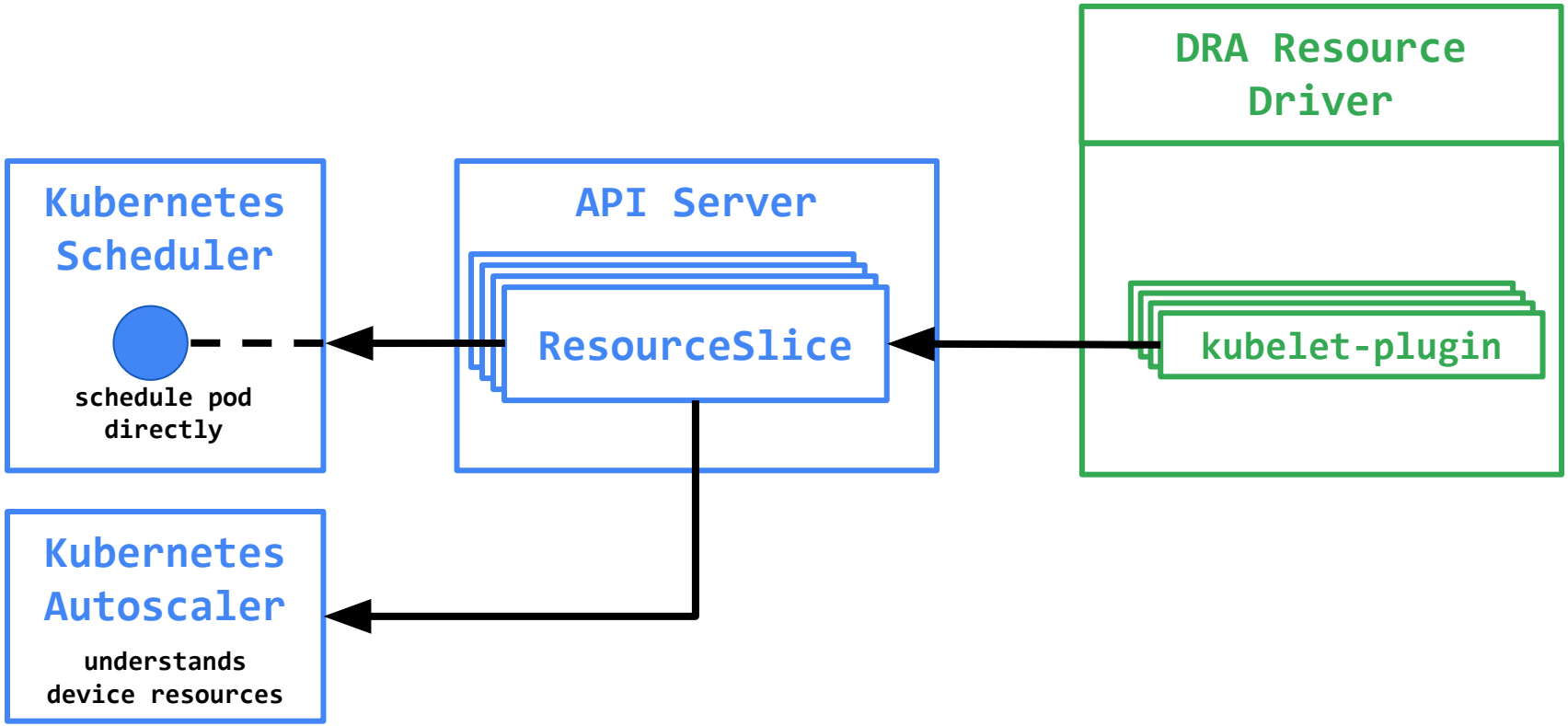
Can configure devices individually

Can share GPUs in the same node for diverse workloads

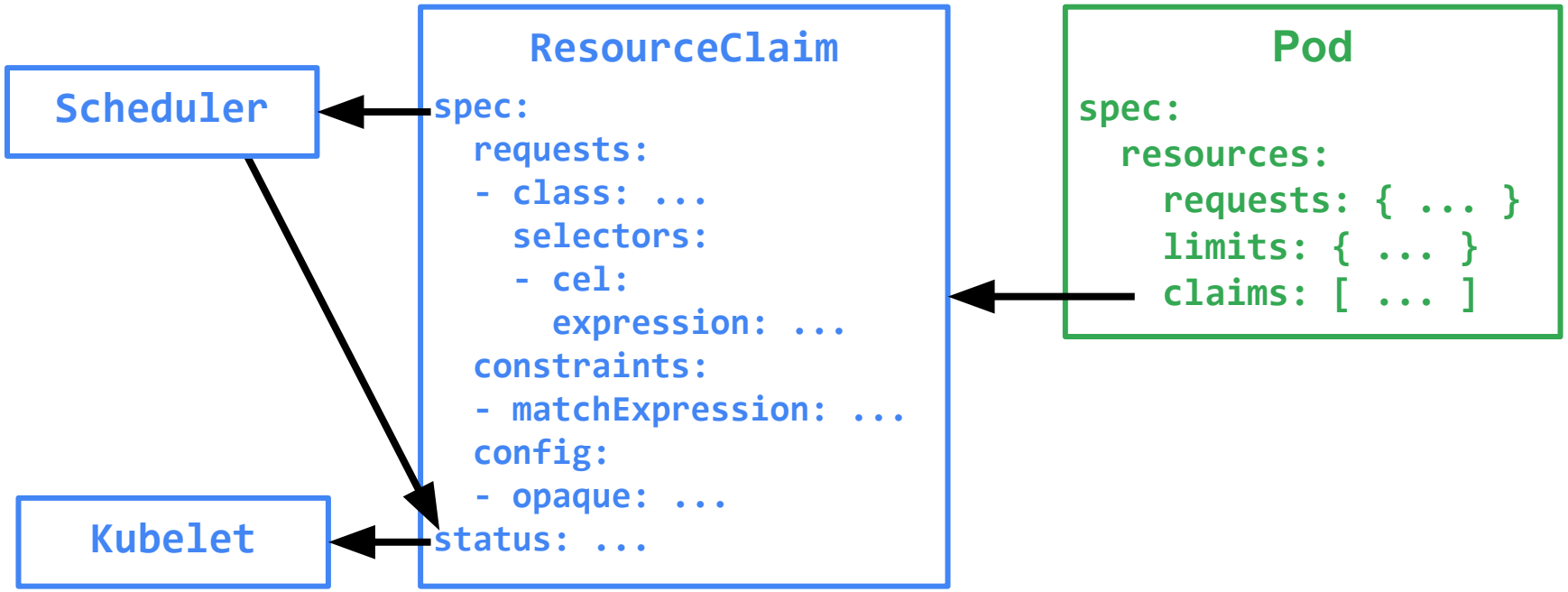
Foundational for new functionality:

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

DRA: Advertising resources



DRA: Requesting resources



Device Enumeration by a DRA Resource Driver

```yaml
apiVersion: resource.k8s.io/v1alpha3
  kind: ResourceSlice
  metadata:
    name: gpu-node-0
  spec:
    driver: gpu.nvidia.com
    nodeName: gpu-node-0
    pool: {...}
    devices:
    - basic:
        name: gpu-0
        attributes:
          uuid:
            string: GPU-18db0e85-99e9-c746-8531-ffeb86328b39
          index:
            int: 0
          model:
            string: "NVIDIA GH200 96GB HBM3"
          driverVersion:
            version: 550.94.0
          ...
        capacity:
          memory: 96Gi
          ...
  ...
```

A Device is a named list of attributes and capacities

Dynamic Resource Allocation (DRA) in Kubernetes

Device Enumeration by a DRA Resource Driver

```
apiVersion: resource.k8s.io/v1alpha3
  kind: ResourceSlice
  metadata:
    name: gpu-node-0
  spec:
    driver: gpu.nvidia.com
    nodeName: gpu-node-0
    pool: {...}
    devices:
    - basic:
        name: gpu-0
        attributes:
          uuid:
            string: GPU-18db0e85-99e9-c746-8531-ffeb86328b39
          index:
            int: 0
          model:
            string: "NVIDIA GH200 96GB HBM3"
          driverVersion:
            version: 550.94.0
          ...
        capacity:
          memory: 96Gi
          ...
  ...
```

A Device is a named list of attributes and capacities

Dynamic Resource Allocation (DRA) in Kubernetes

Device Enumeration by a DRA Resource Driver

```yaml
apiVersion: resource.k8s.io/v1alpha3
kind: ResourceSlice
metadata:
  name: gpu-node-0
spec:
  driver: gpu.nvidia.com
  nodeName: gpu-node-0
  pool: {...}
  devices:
  - basic:
      name: gpu-0
      attributes:
        uuid:
          string: GPU-18db0e85-99e9-c746-8531-ffeb86328b39
        index:
          int: 0
        model:
          string: "NVIDIA GH200 96GB HBM3"
        driverVersion:
          version: 550.94.0
        ...
      capacity:
        memory: 96Gi
        ...
  ...
```

A Device is a named list of attributes and capacities

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: DeviceClass
metadata:
  name: gpu.nvidia.com
spec:
  selectors:
  - cel:
      expression: "device.driverName == 'gpu.nvidia.com'"
```

Select any device governed by the NVIDIA DRA driver for GPUs

Dynamic Resource Allocation (DRA) in Kubernetes

Sharing across containers in a single Pod

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  devices:
    requests:
    - deviceClassName: gpu.nvidia.com
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example
spec:
  containers:
    - name: ctr0
      resources:
        claims:
        - name: gpu
    - name: ctr1
      resources:
        claims:
        - name: gpu
  resourceClaims:
  - name: gpu
    resourceClaimName: shared-gpu
```

Shared access to same underlying GPU

Dynamic Resource Allocation (DRA) in Kubernetes

Sharing across containers in a different Pods

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  devices:
    requests:
    - deviceClassName: gpu.nvidia.com
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example0
spec:
  containers:
    - name: ctr
      resources:
        claims:
        - name:gpu
resourceClaims:
  - name: gpu
    resourceClaimName: shared-gpu
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: gpu-example0
spec:
  containers:
    - name: ctr
      resources:
        claims:
        - name:gpu
resourceClaims:
  - name: gpu
    resourceClaimName: shared-gpu
```

Shared access to same underlying GPU

Dynamic Resource Allocation (DRA) in Kubernetes

Selection vs. Configuration of a Device

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: DeviceClass
metadata:
  name: gpu.nvidia.com
spec:
  selectors:
  - cel:
      expression: "device.driverName == 'gpu.nvidia.com'"
```

Select any device governed by the NVIDIA DRA driver for GPUs

Dynamic Resource Allocation (DRA) in Kubernetes

Selection vs. Configuration of a Device

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: DeviceClass
metadata:
  name: gpu.nvidia.com
spec:
  selectors:
  - cel:
      expression: "device.driverName == 'gpu.nvidia.com'"
```

Select any device governed by the NVIDIA DRA driver for GPUs

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: shared-gpu
spec:
  devices:
    requests:
    - name: gpu
      deviceClassName: gpu.nvidia.com
  config:
- requests: ["gpu"]
  opaque:
      driver: gpu.nvidia.com
      parameters:
        apiVersion: gpu.resource.nvidia.com/v1alpha1
        kind: GpuConfiguration
        sharing:
        - strategy: MPS
          mpsConfig:
            defaultActiveThreadPercentage: 20
            defaultPinnedDeviceMemoryLimit: 10Gi
```

Reference a DeviceClass to inherit its selectors

Configure it to give each client dedicated 10Gi of memory and 20% of active compute

Dynamic Resource Allocation (DRA) in Kubernetes

Complex Example

```yaml
apiVersion: resource.k8s.io/v1alpha2
kind: ResourceClaim
metadata:
  name: big-gpu-with-aligned-nic
spec:
  devices:
    requests:
      ...
```

Dynamic Resource Allocation (DRA) in Kubernetes

Complex Example

```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

Complex Example

```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

Complex Example

```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

DRA Use-Cases for NVIDIA GPUs

- With the model in Kubernetes 1.30, we cover **6 / 12** use-cases identified in NVIDIA GPU Use-Cases for Dynamic Resource Allocation (DRA)
- Supported:
  - Controlled GPU sharing
  - GPU selection via complex constraints
  - Multiple GPU types per node
  - User-driven time-slicing support across a subset of GPUs on a node
  - User-driven MPS support across a subset of GPUs on a node
  - Dynamic swapping of NVIDIA driver with vfio driver depending on intended use of GPU
- Unsupported:
  - “Management” pods with access to all GPUs without allocating them
  - Dynamic allocation of MIG devices
  - MIG device alignment
  - Subdivision of MIG devices with shared memory but dedicated compute resources
  - Custom policies to align multiple resource types (e.g. GPUs and NICs)
  - Application-specific policies for how GPUs are allocated across containers / pods

DRA Use-Cases for NVIDIA GPUs

- With the model in Kubernetes 1.30, we cover **6 / 12** use-cases identified in NVIDIA GPU Use-Cases for Dynamic Resource Allocation (DRA)
- Supported:
  - Controlled GPU sharing
  - GPU selection via complex constraints
  - Multiple GPU types per node
  - User-driven time-slicing support across a subset of GPUs on a node
  - User-driven MPS support across a subset of GPUs on a node
  - Dynamic swapping of NVIDIA driver with vfio driver depending on intended use of GPU
  - “Management” pods with access to all GPUs without allocating them
  - Custom policies to align multiple resource types (e.g. GPUs and NICs)
  - MIG device alignment
- Unsupported:
  - Dynamic allocation of MIG devices
  - Subdivision of MIG devices with shared memory but dedicated compute resources
  - Application-specific policies for how GPUs are allocated across containers / pods

In 1.31 we support 9/12

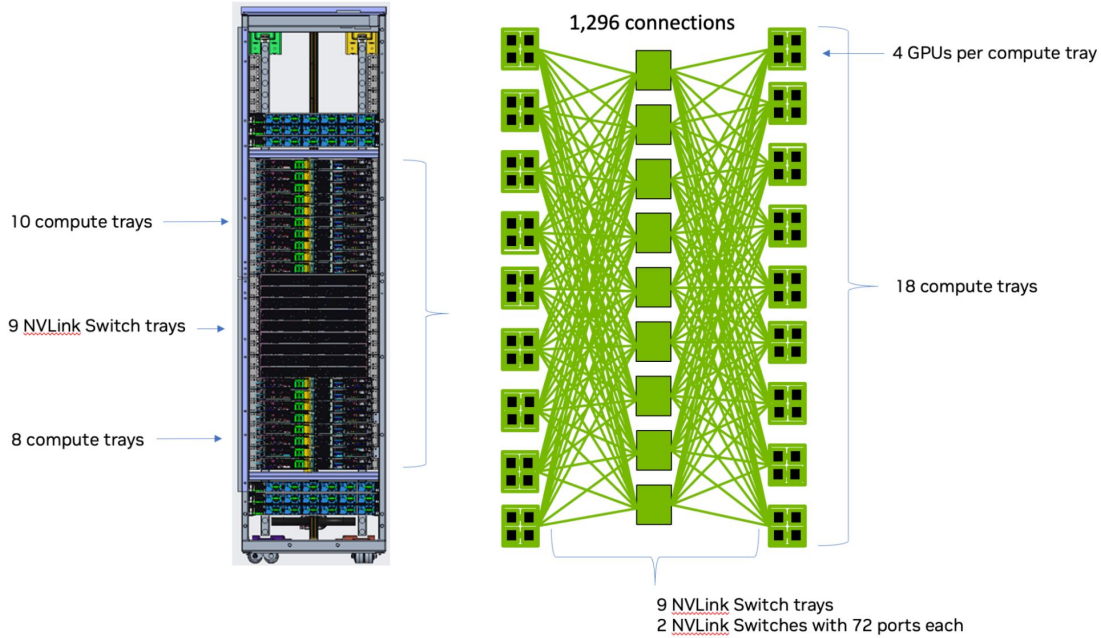
DRA Use-Cases for NVIDIA GPUs

- With the model in Kubernetes 1.30, we cover **6 / 12** use-cases identified in NVIDIA GPU Use-Cases for Dynamic Resource Allocation (DRA)
- Supported:
  - Controlled GPU sharing
  - GPU selection via complex constraints
  - Multiple GPU types per node
  - User-driven time-slicing support across a subset of GPUs on a node
  - User-driven MPS support across a subset of GPUs on a node
  - Dynamic swapping of NVIDIA driver with vfio driver depending on intended use of GPU
  - "Management" pods with access to all GPUs without allocating them
  - Custom policies to align multiple resource types (e.g. GPUs and NICs)
  - MIG device alignment
  - Dynamic allocation of MIG devices
  - Subdivision of MIG devices with shared memory but dedicated compute resources
- Unsupported:
  - Application-specific policies for how GPUs are allocated across containers / pods

By 1.33 we plan to support 11/12

DRA Use-Cases for NVIDIA GPUs

NVIDIA GB200 with Multi-Node NVLink



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

Upcoming Talks, Feedback, and Questions

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

Wednesday November 13, 2024 3:25pm - 4:00pm MST | Salt Palace | Level 2 | 255 B

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

Thursday November 14, 2024 4:30pm - 5:05pm MST | Salt Palace | Level 2 | 255 E

Better Together! GPU, TPU and NIC Topological Alignment with DRA
John Belamaric (Google), Patrick Ohly (Intel)

Friday November 15, 2024 11:00am - 11:35am MST | Salt Palace | Level 2 | 250 AD

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