Hewlett Packard Labs

SLA-Driven ML INFERENCE FRAMEWORK FOR CLOUDS WITH HETEROGENEOUS ACCELERATORS

Junguk Cho, email: junguk.cho@hpe.com

Diman Zad Tootaghaj, email: diman.zad-tootaghaj@hpe.com

Lianjie Cao, email: lianjie.cao@hpe.com

Puneet Sharma, email: puneet.Sharma@hpe.com

Hewlett Packard Enterprise

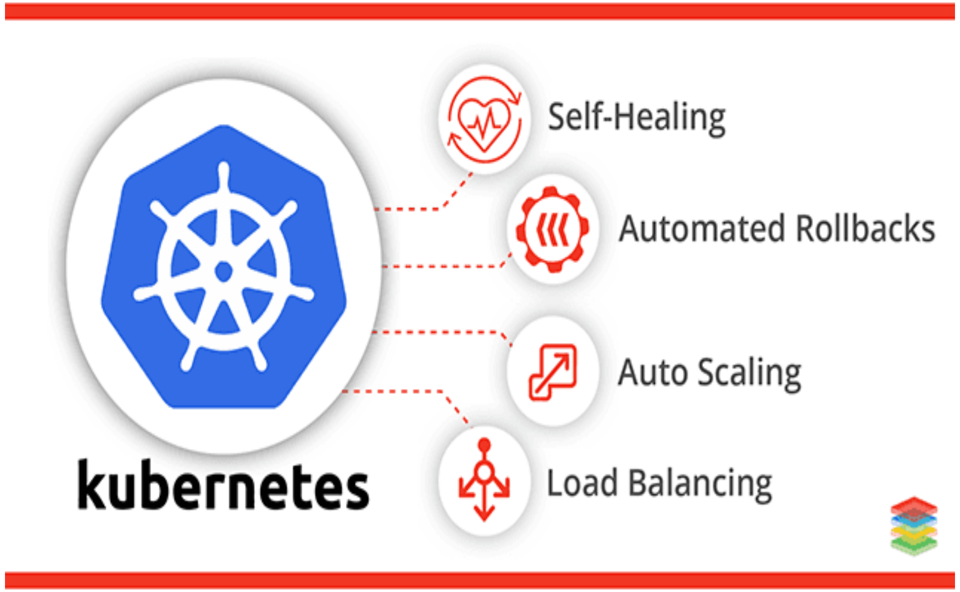
Deep Neural Network (DNN)-based Inference Applications

– High demand to serve inference requests per day
  – Facebook serves tens of trillions of time per day
  – Seagate performs inference on 3 million images every day

– DNN accelerators
  – Fundamental scale-up limitations of CPU
  – Accelerators specifically designed to optimize DL/ML compute operations like matrix computations

– **GPU is one of the popular DNN accelerators**

High Demand to Use GPUs on Kubernetes



https://www.nvidia.com/en-us/data-center/virtual-gpu-technology/
https://www.xenonstack.com/insights/kubernetes-deployment/

K8s is designed for running containers on homogeneous CPU resources

– No well-defined interface to manage heterogeneous GPUs on Kubernetes while each GPU has significantly different computation capability

– Support a model of exclusive GPU assignment to one container or a time multiplexing approach while GPU computation power significantly and sharing technology have been evolved

– The workload distribution (i.e., inference requests) is uniformly distributed regardless of the power of the underlying GPU

Cause resource inefficiency and performance degradation

Key Inventions

- Design and build a novel heterogeneous-aware GPU cluster management system for use on a container platform.
  - New GPU resource abstraction by expressing one physical GPU as multiple logical GPUs.
  - Efficient sharing of GPU resources among multiple applications by leveraging spatial GPU sharing support.
  - Leverages underlying GPU hardware heterogeneity and application characteristics to optimize workload distribution.

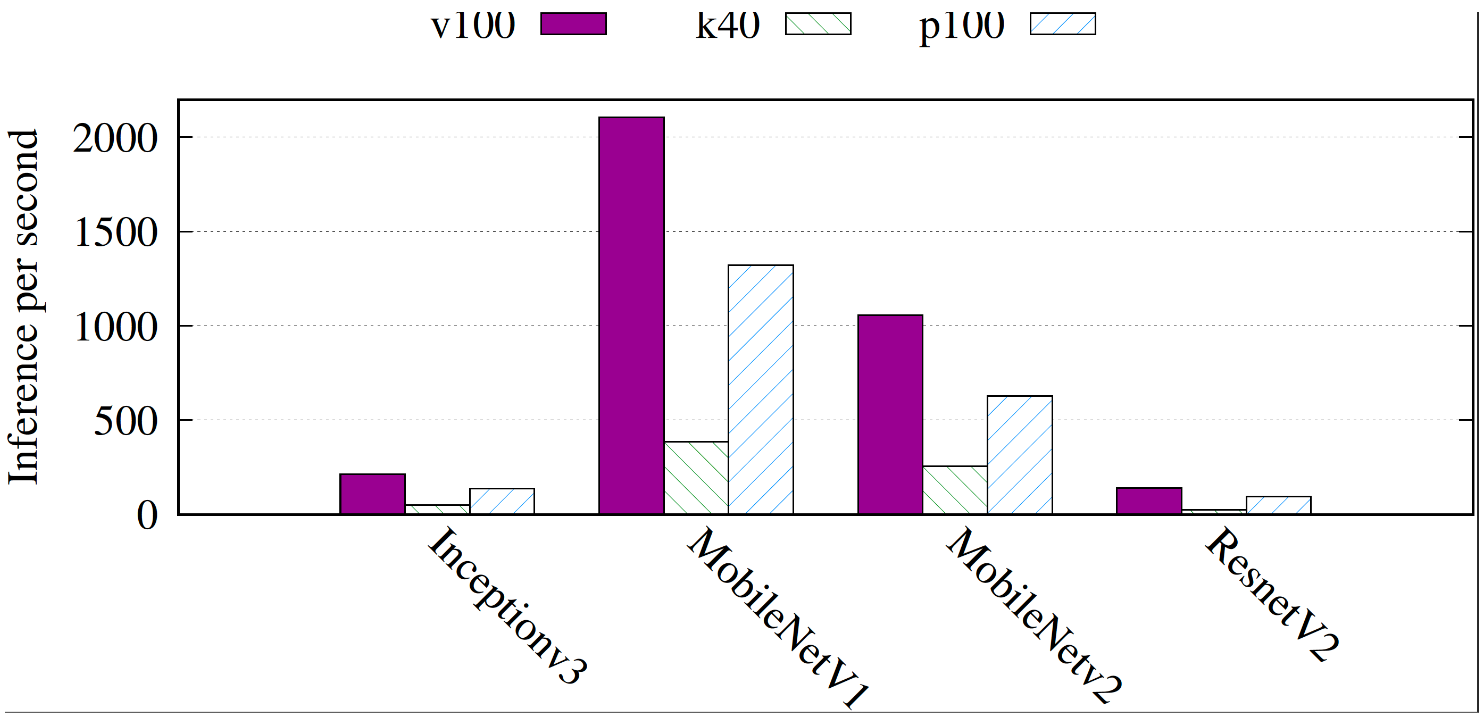
- **Key system components from key inventions**
  - Enabling heterogeneous GPU management: GPU operator, GPU scheduler & device plugin
  - Efficient GPU resource management : Bin-packing & hardware and application-aware workload management
  - Pluggable and evolvable solution based on Kubernetes well-defined interfaces (e.g., extended resource, custom resource and its controller, scheduler extender and device plugins) without modifying K8s by itself

Review of State-of-art GPUs & K8s supports for GPUs

- Heterogeneous GPUs inference performance & sharing impact
- Kubernetes extension for supporting GPUs

- Testbed setup for experiments and developments
  - Kubernetes v1.18.4
  - Docker v19.03.8 and use Nvidia docker to run inference containers
  - Cuda 10.2 version & Driver version 440.33.01
  - Workload
    - TensorRT Inference Server (TRTIS) v19.03
    - ImageNet DNN models (inceptionV3, mobilenet v1 and v2, resnet 50, 101, and 152)
  - V100, T4, P100 and K40 GPUs

Heterogeneous Inference Performance



Observation 1. Various performance according to GPU and DNN models (V100 >>> P100 > T4 > K40)

Insight 1. A container platform should treat different models of GPUs differently when performing application assignment

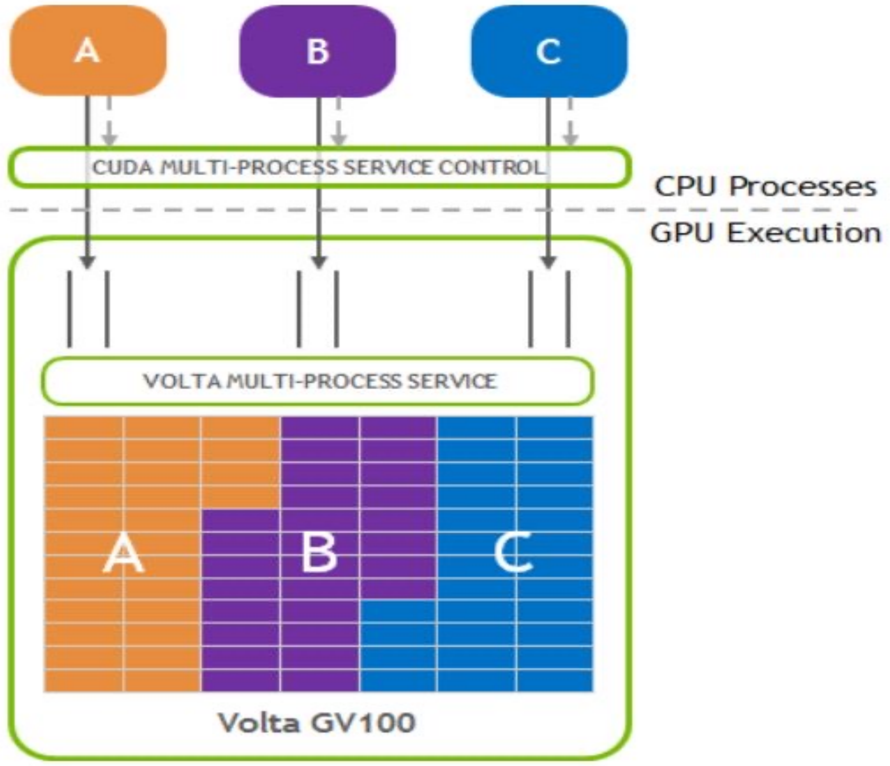
Multi-Process Service (MPS)

– Spatial sharing approach

– Supported in case NVIDIA compute power is higher than 7.0 (e.g., V100, T4)

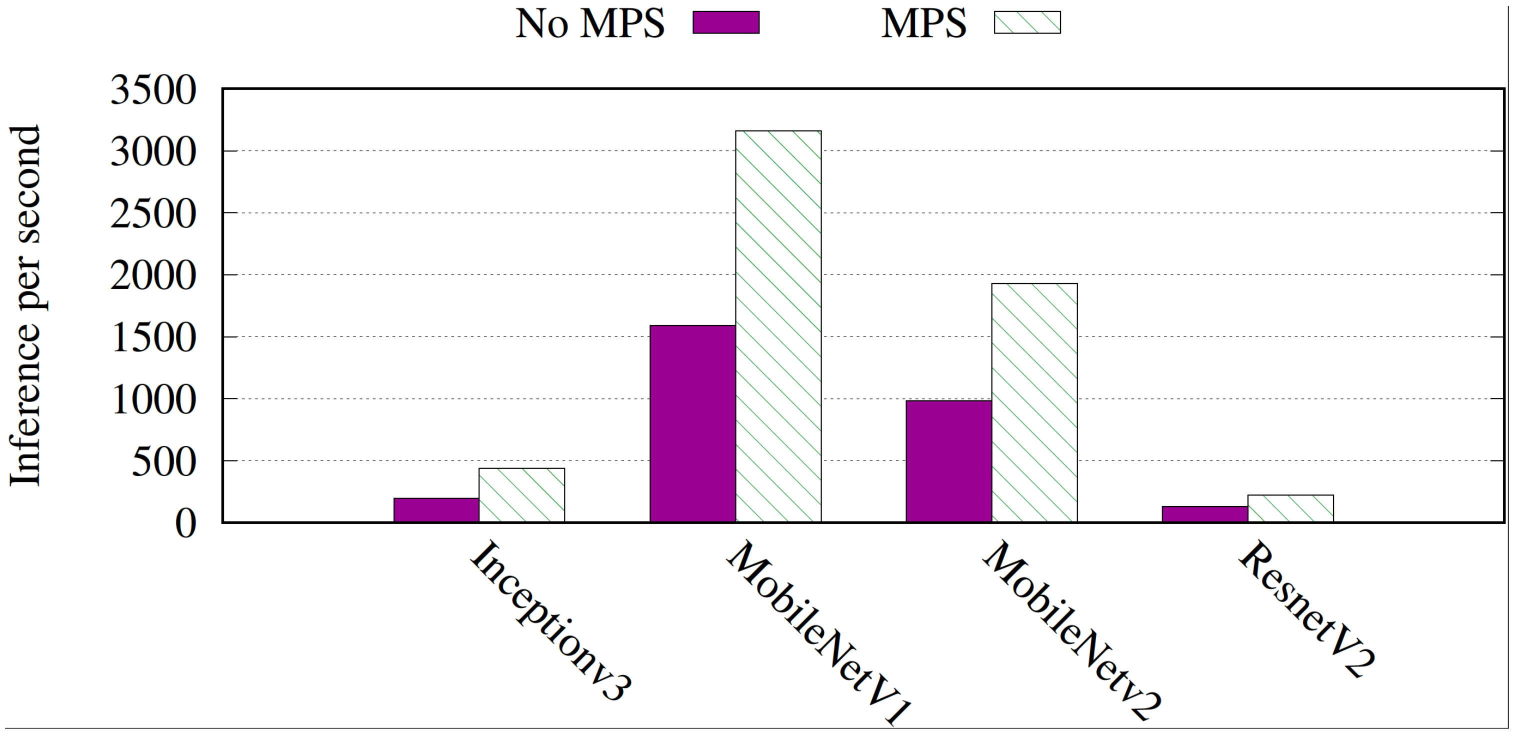
– Secure way of sharing GPU cores

– Assign GPU resource with CUDA_MPS_ACTIVE_THREAD_PERCENTAGE to application



https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

Sharing GPUs Impact



Observation 2. Spatial sharing >> Time-multiplexing

Insight 2. Leverage spatial sharing in container platform

Performance According to Various GPU Allocations with Spatial GPU sharing

- **Observation 3.** Performance according to various GPU allocations. It is not linear and more than 50% GPU resource allocation to an inference application, the performance is not significantly different.

- **Insight 3.**
- (i) DNN inference does not saturate GPU resources. Can share GPU for multiple inference containers
- (ii) The allocation of GPU resources should be aligned with application's requirements (e.g., low latency, high throughput).

- **Observation 4.** Spatial sharing guarantees stable performance isolation with minimal overhead

- **Insight 4.** Can assign multiple applications on the same GPU

Existing K8s Extension for Supporting GPUs

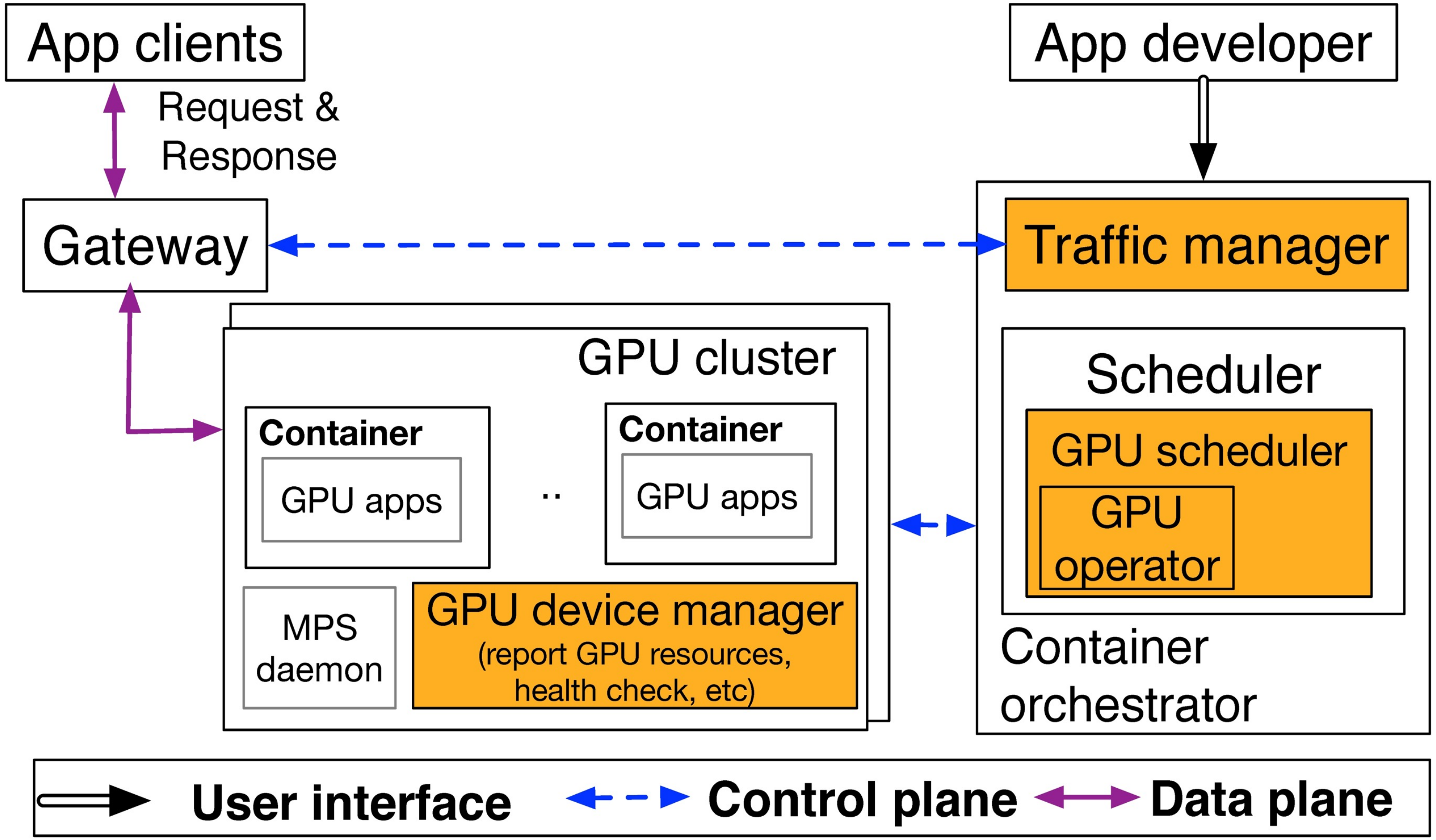
From	Project	GPU granularity	Features	Git repo
NVIDIA	NVIDIA device plugin for Kubernetes	The number of GPUs (e.g., GPU-count)	**Recently started adding Multi-Instance GPUs for A100**	https://github.com/NVIDIA/k8s-device-plugin
deepomatic	Fork of NVIDIA device plugin for Kubernetes with support for shared GPUs by declaring GPUs multiple times	Support for shared GPUs by declaring GPUs multiple times (e.g., GPU-count)	Time-multiplexing approach	https://github.com/Deepomatic/shared-gpu-nvidia-k8s-device-plugin
Alibaba	GPU Sharing Device Plugin for Kubernetes Cluster	Memory (e.g., Gpu-mem)	Time multiplexing approach, GPU scheduler extender (bin-packing based on homogeneity)	https://github.com/AliyunContainerService/gpushare-device-plugin
AMD	Kubernetes (k8s) device plugin to enable registration of AMD GPU to a container cluster	The number of GPUs (e.g., GPU-count)		https://github.com/RadeonOpenCompute/k8s-device-plugin
Run.ai	RUN:AI CREATES FIRST FRACTIONAL GPU SHARING FOR KUBERNETES DEEP LEARNING WORKLOADS	Allowing for fractions of GPUs to be assigned to containers		No information

Hewlett Packard Enterprise

Lack of efficient GPU sharing with spatial sharing (MPS) and managing heterogeneous GPUs

System Architecture Overview

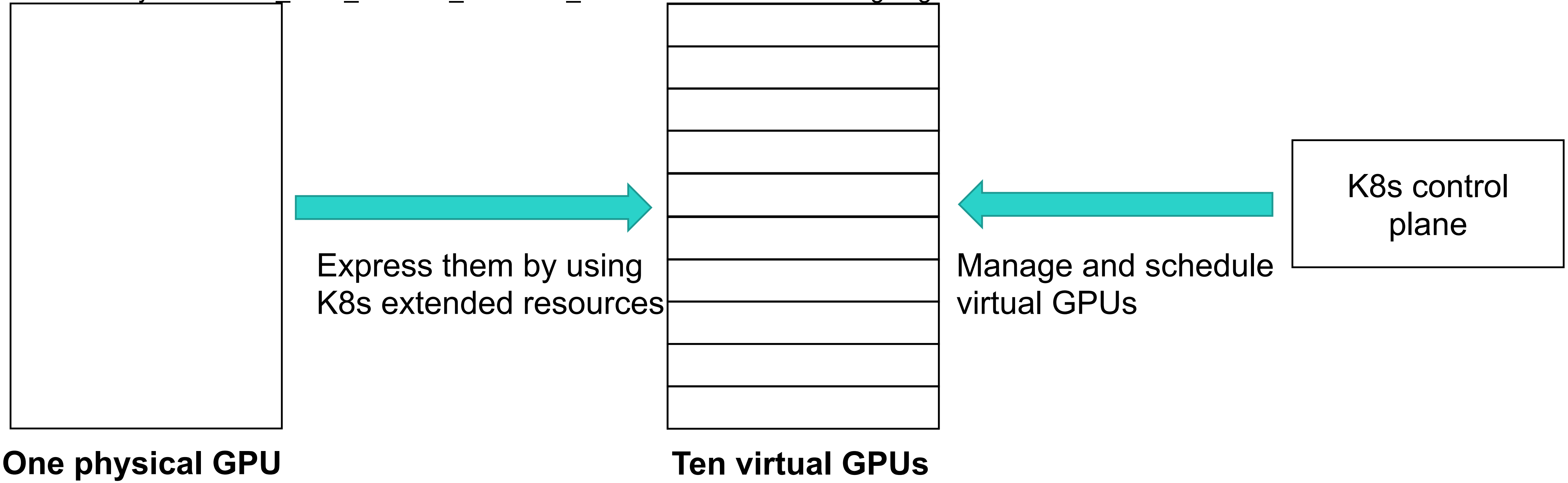
– A novel heterogeneous-aware GPU cluster management system to manage DNN inference applications.
  - **Enabling heterogeneous GPU management: GPU operator, GPU scheduler & device plugin**
  - **Efficient GPU resource management : Bin-packing & workload management**



Orion GPU Resource Abstraction

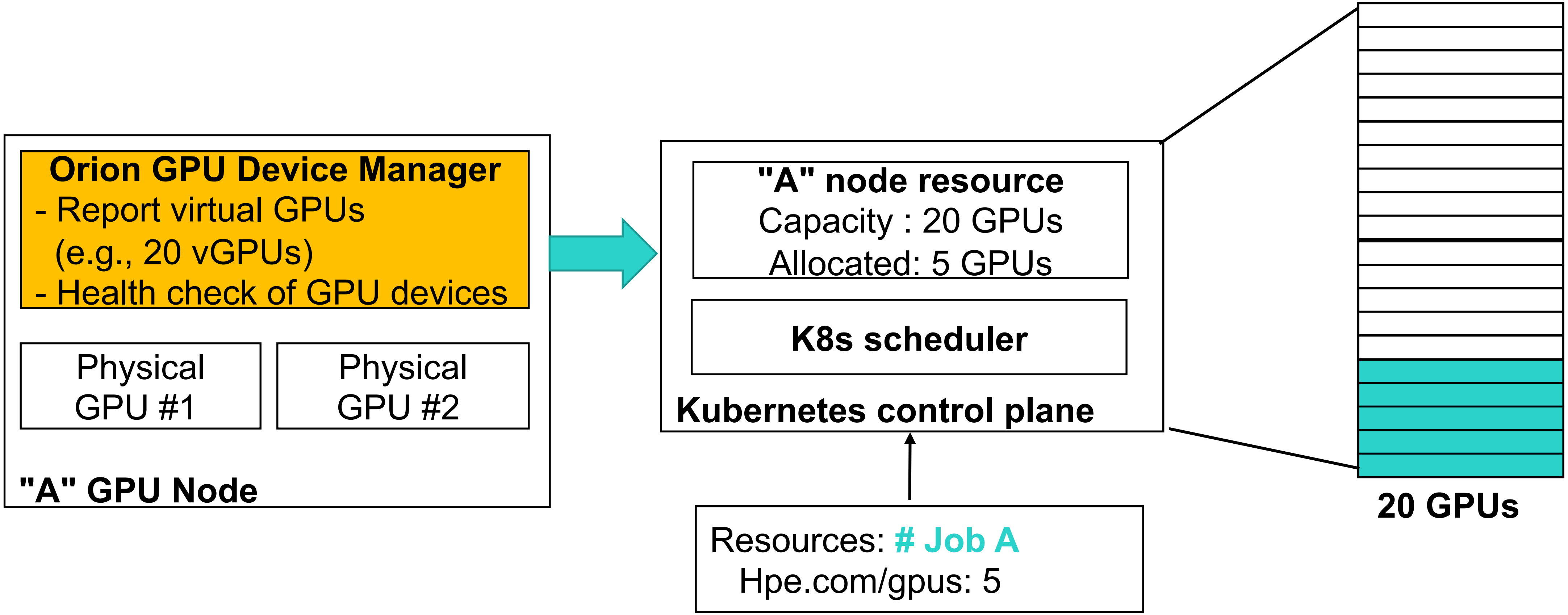
- Divide one physical GPU to multiple virtual GPUs
- Treats GPUs as first-class computing resources
  - Expressed multiple virtual GPUs as **K8s Extended Resources**
    - **Report resource name and quantity of the resource (e.g., hpe.com/gpus, 10)**
    - K8s can assign these virtual GPUs as assignable resources for pods
    - Internally use CUDA_MPS_ACTIVE_THREAD_PERCENTAGE when assigning virtual GPUs

Resources: **# Job A**
  Hpe.com/gpus: 5
Resources: **# Job B**
  Hpe.com/gpus: 6



View of GPU Resources with Extended Resource in Kubernetes

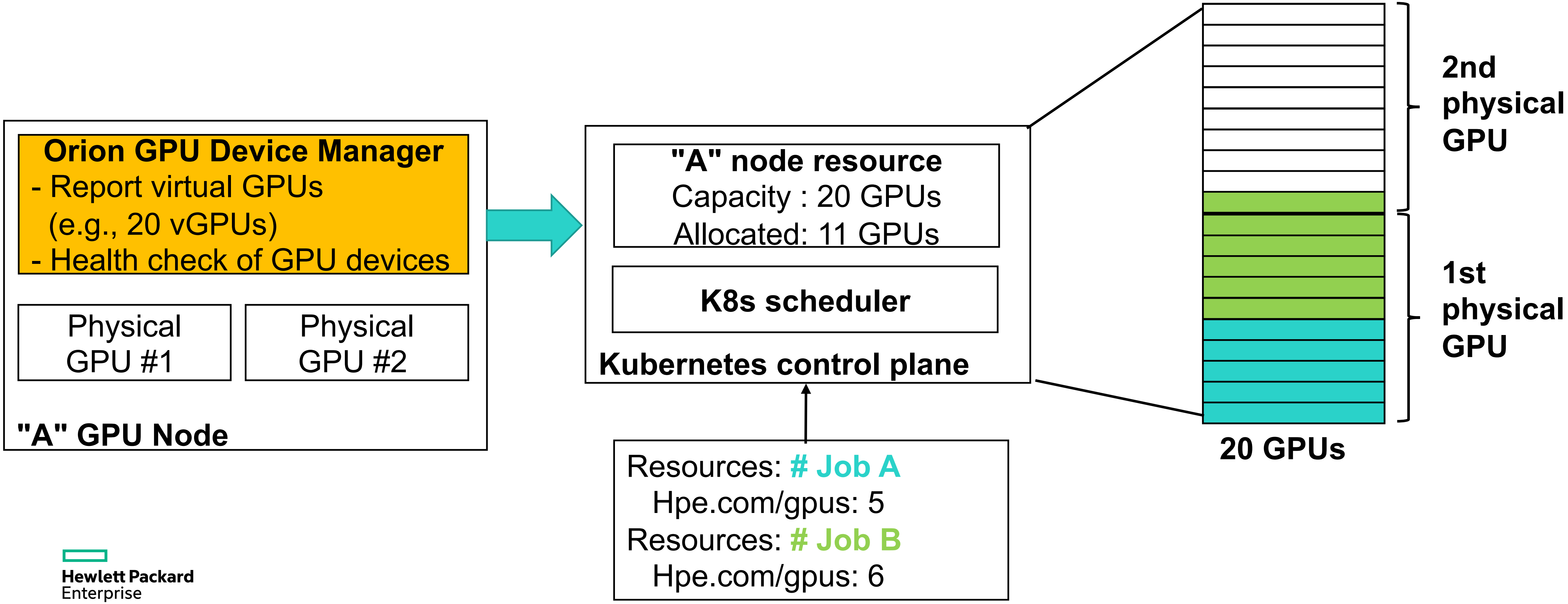
– K8s provides course-grained management for extended resource
  – Default scheduler calculates the extended resource and can only determine whether **the total amount of resources** has free resources to meet the demand



Limitation of Extended Resource and GPU Device Manager (1)

– **GPU resource oversubscription**

– Expected behavior : K8s scheduler assigns "Job B" into 2nd physical GPU



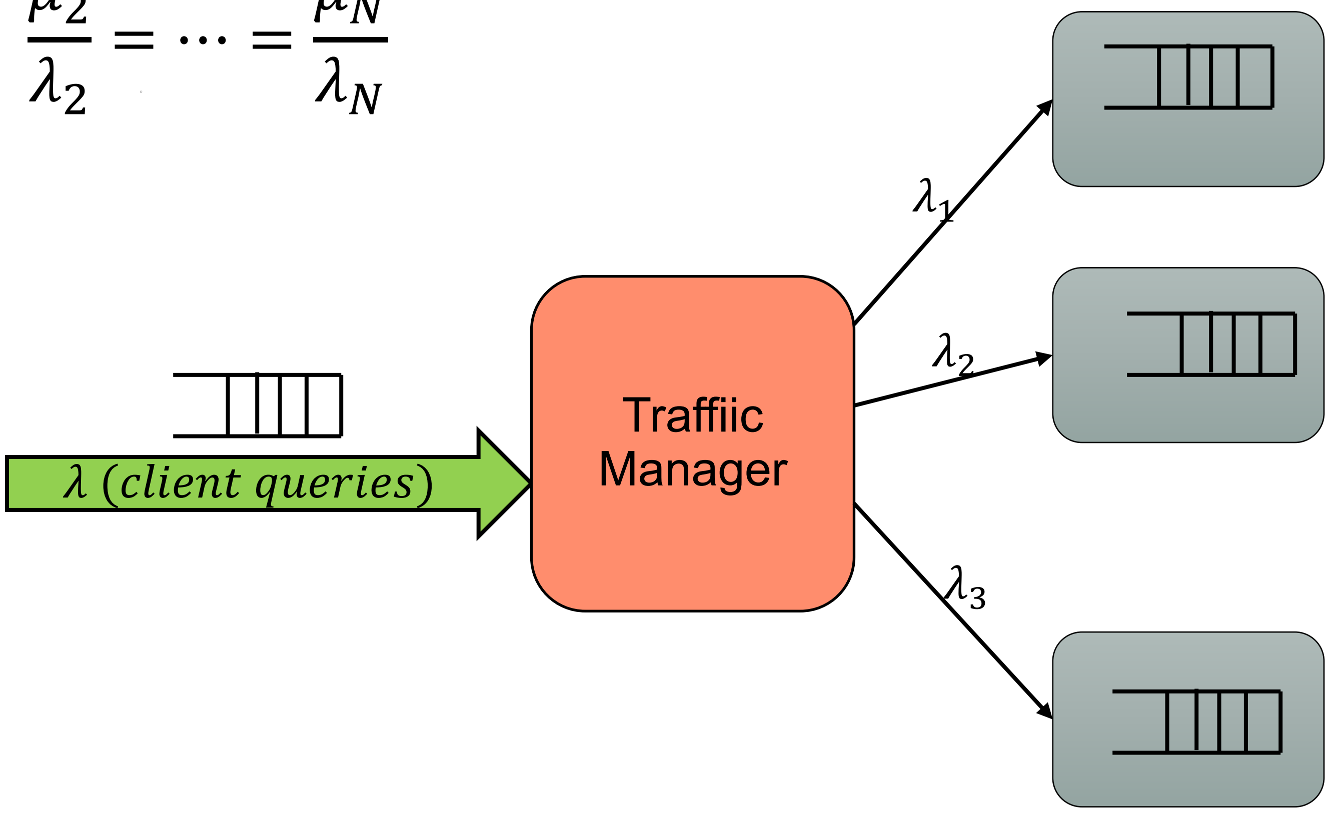
Hewlett Packard Enterprise

Traffic manager

How to distribute the traffic on a heterogeneous environment?

➢ **Our solution: A queueing-based load distributor module**

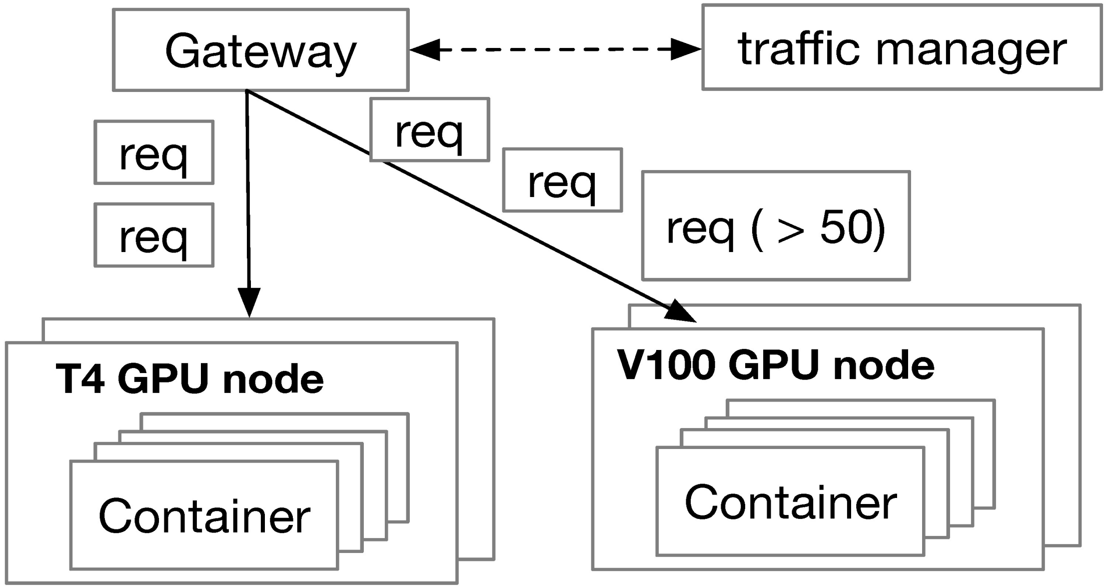
$$\frac{\mu_1}{\lambda_1} = \frac{\mu_2}{\lambda_2} = \cdots = \frac{\mu_N}{\lambda_N}$$



Traffic Manager

– Manage inference workload distribution by controlling the gateway

– Two inference workload routing policies
  – hardware-aware and inference request-aware routings
  – Leverage Service mesh capabilities(e.g, Istio)

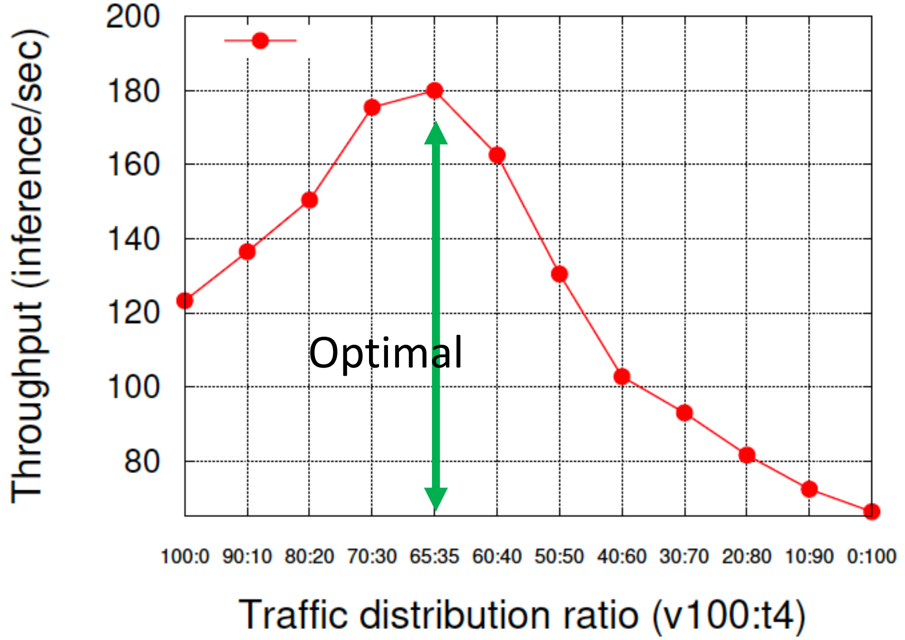


Routing policies

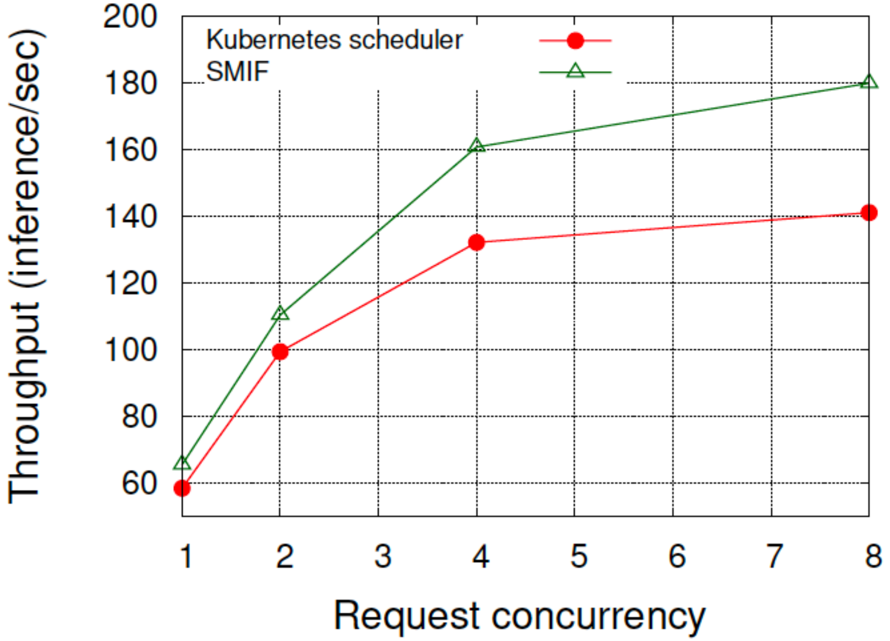
```
inferrequest:
batch_size: 100
input layer: \"input\" }
output layer: \"InceptionV3/Predictions/
Reshape_1\"
```

inference request header

Traffic Management results



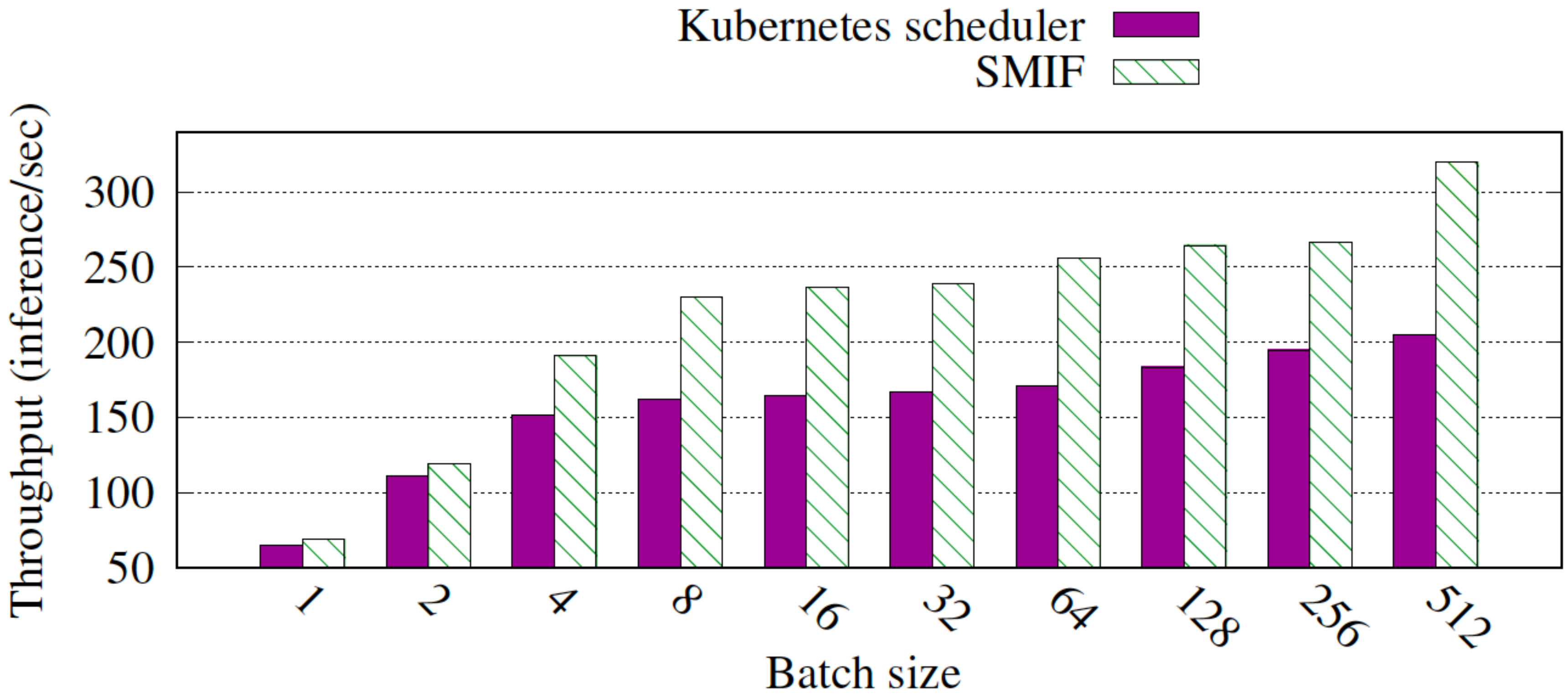
a. Traffic Distribution.



b. Comparison.

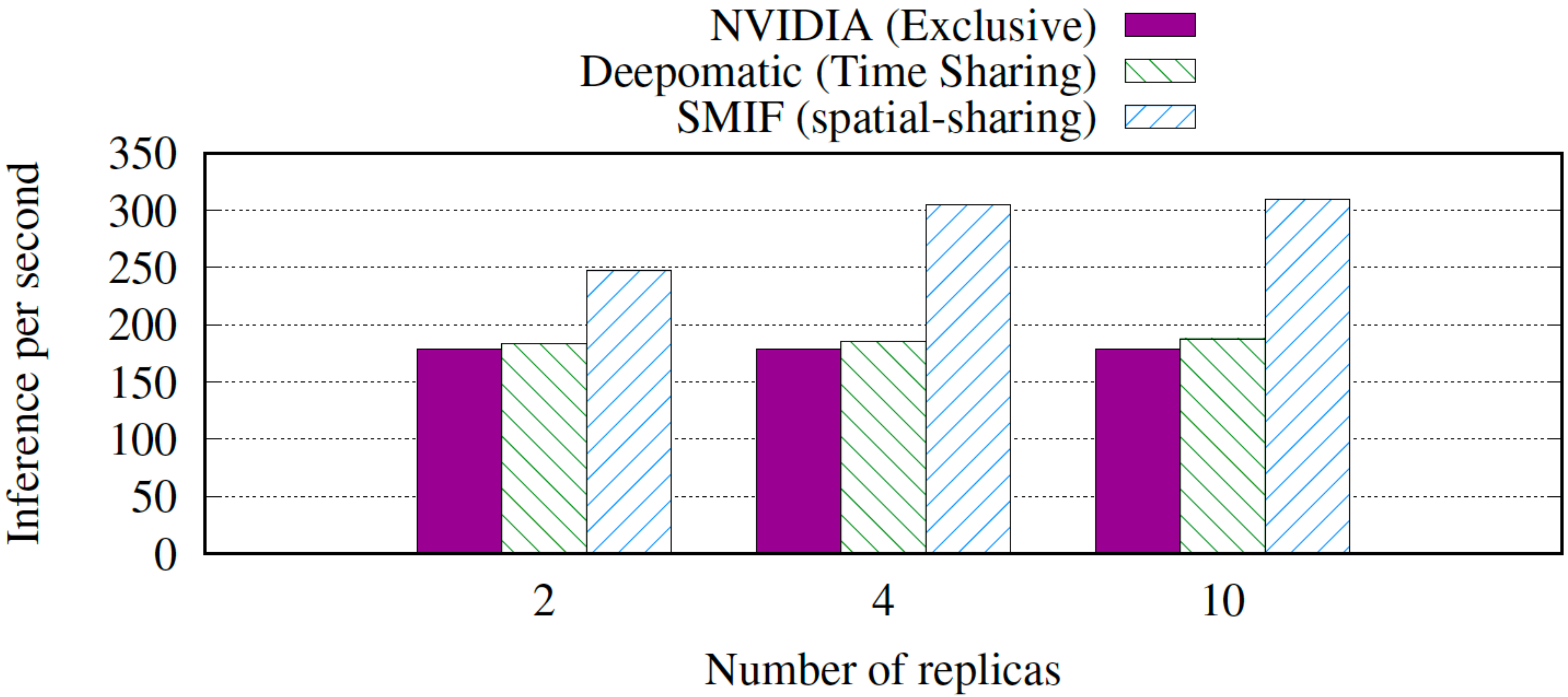
Throughput of SMIF for different traffic distribution ratios, and (b) Comparison of SMIF with respect to the default Kubernetes scheduler.

Throughput Result



Throughput as we increase the batch size of inference application on the Kubernetes scheduler and SMIF.

Throughput Result



Throughput comparison between SMIF, Nvidia exclusive and Deepomatic frameworks.

Conclusion

– Design and build automated and fine-grained GPU cluster management

– **Key contributions**

- Enabling heterogeneous GPU management: GPU operator, GPU scheduler & device plugin

- Efficient GPU resource management : Bin-packing & workload management

- Efficient traffic management : Hardware-aware and inference request-aware routings

- Pluggable and evolvable solution based on Kubernetes well-defined interfaces (e.g., extended resource, custom resource and its controller, scheduler extender and device plugins) without modifying K8s by itself

Thank you!

We are hiring at Hewlett Packard Labs

Talk to us:

Diman Zad Tootaghaj email: diman.zad-tootaghaj@hpe.com

Lianjie Cao, email: lianjie.cao@hpe.com