CLOUD NATIVE
COMPUTING FOUNDATION

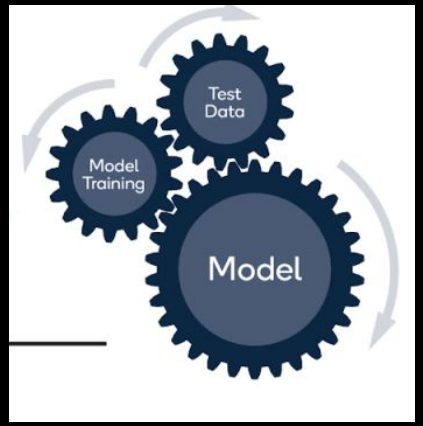
Unlocking the Future of GPU Scheduling in Kubernetes with Reinforcement Learning

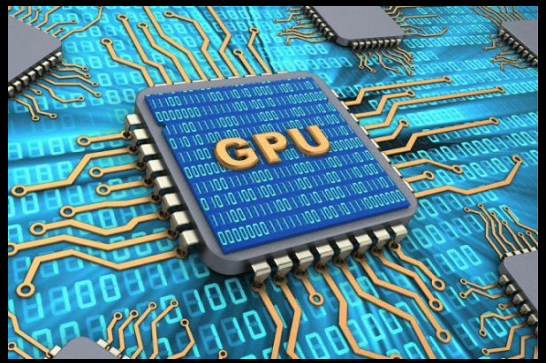
- Aditi Gupta, Nikunj Goyal

Presentation Outline

- **Background** - Need of GPUs for ML Workloads
- **Current Scenario** - GPU Scheduling in K8s
  - Kube Scheduler - Components , Architecture, Scheduling Workflow
  - Challenges and Issues
- **Proposed Solution** - Reinforcement Learning based Scheduler
  - Model Architecture
  - Training Algorithm
  - Results and Benefits
- **CNCF Landscape**
  - Benefits from RL Scheduler
  - **Kube Flow** - Integration and Improvements

Training Large scale ML models on cloud - GPU scheduling in Kubernetes





Compute Hungry Models

Manage and schedule the resources

Allows Efficient Parallel Computations

GPUs for Modern Workloads

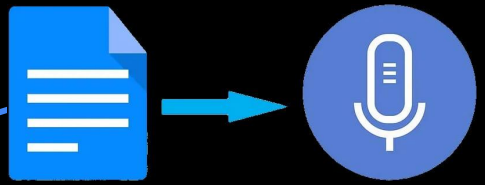
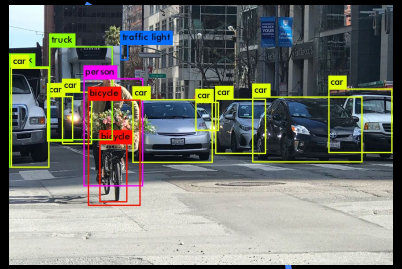


Image Classification for self driving cars

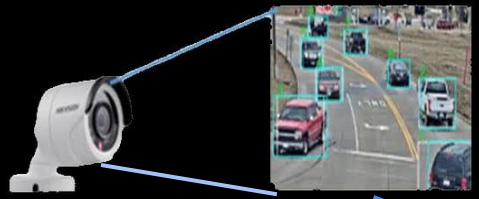


Recommendation Systems in E-commerce

Speech to text

Text to speech

Chat GPT



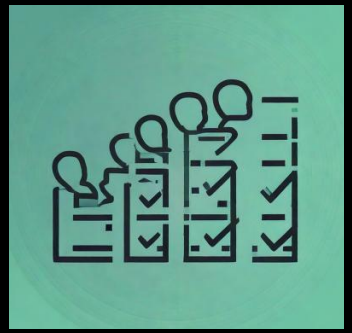
Video Analysis for Security Surveillance

GPUs & GPU Scheduling

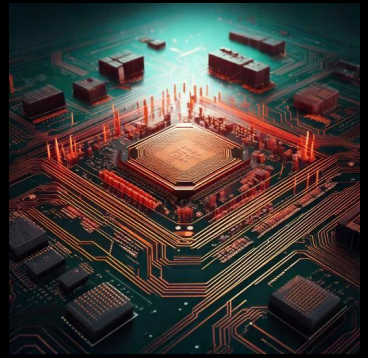
GPU - An electronic chip that can crunch numbers super fast thus very essential to all the things ML.

How Scheduling Works

Resource Request



Scheduling Decision



Job Placement

Monitoring & Scaling

GPUs in Kubernetes : Key Components

Resource Request

- Pods request GPUs in YAML manifest
- Scheduler places pods on nodes with GPUs

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

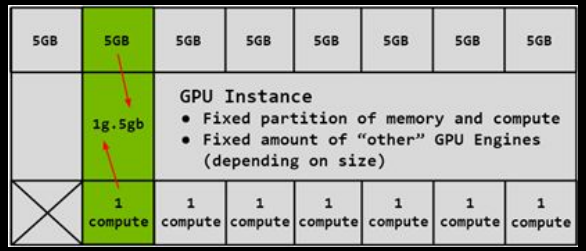
Device Plugins

- gRPC servers expose GPUs to kubelet
- NVIDIA's plugin manages allocation efficiently

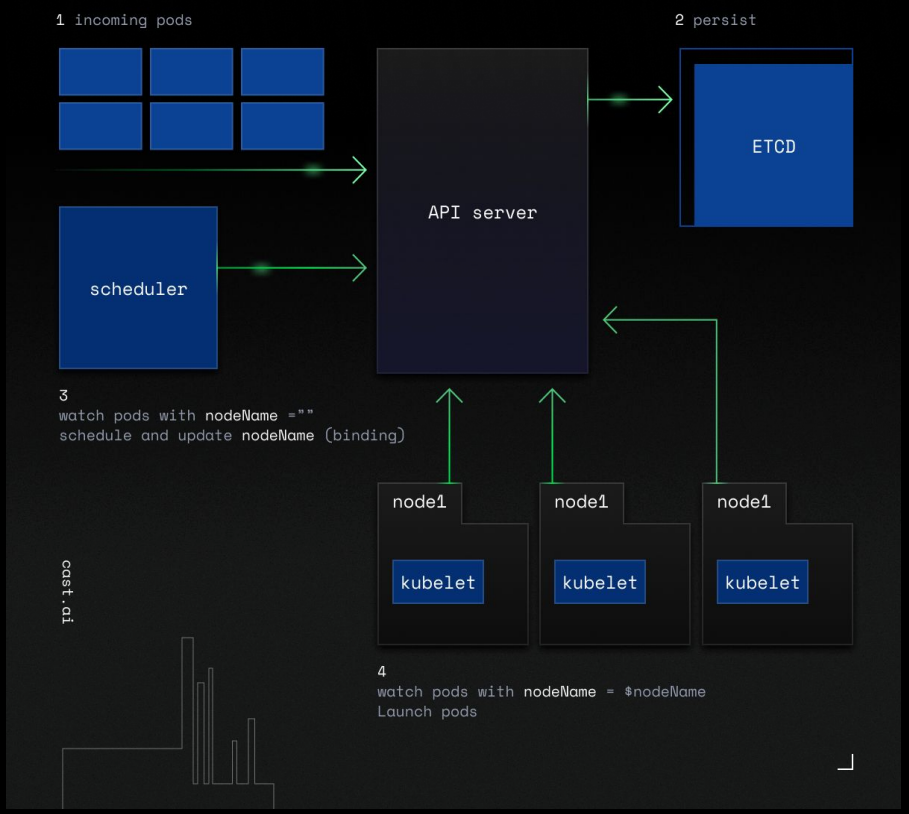
```yaml
spec:
    containers:
        name: device-plugin-ctr
        image: NVIDIA/device-plugin:1.0
        volumeMounts:
          - mountPath: /device-plugin
          - name: device-plugin
```

MIG and GPU Slicing

- Slices GPUs into instances
- Kubernetes allocates resources for optimal usage.



Default Scheduling in Kubernetes



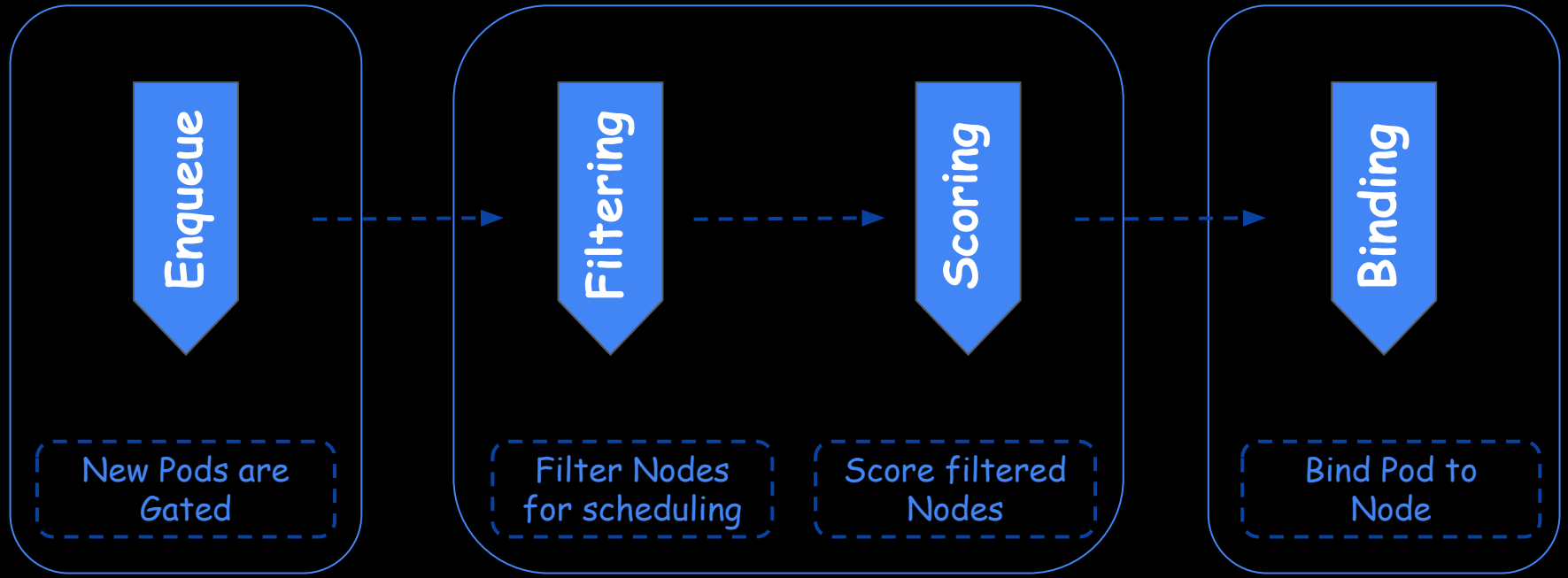
Kube Scheduler Components

Scheduler — Evaluates available nodes and makes scheduling decisions

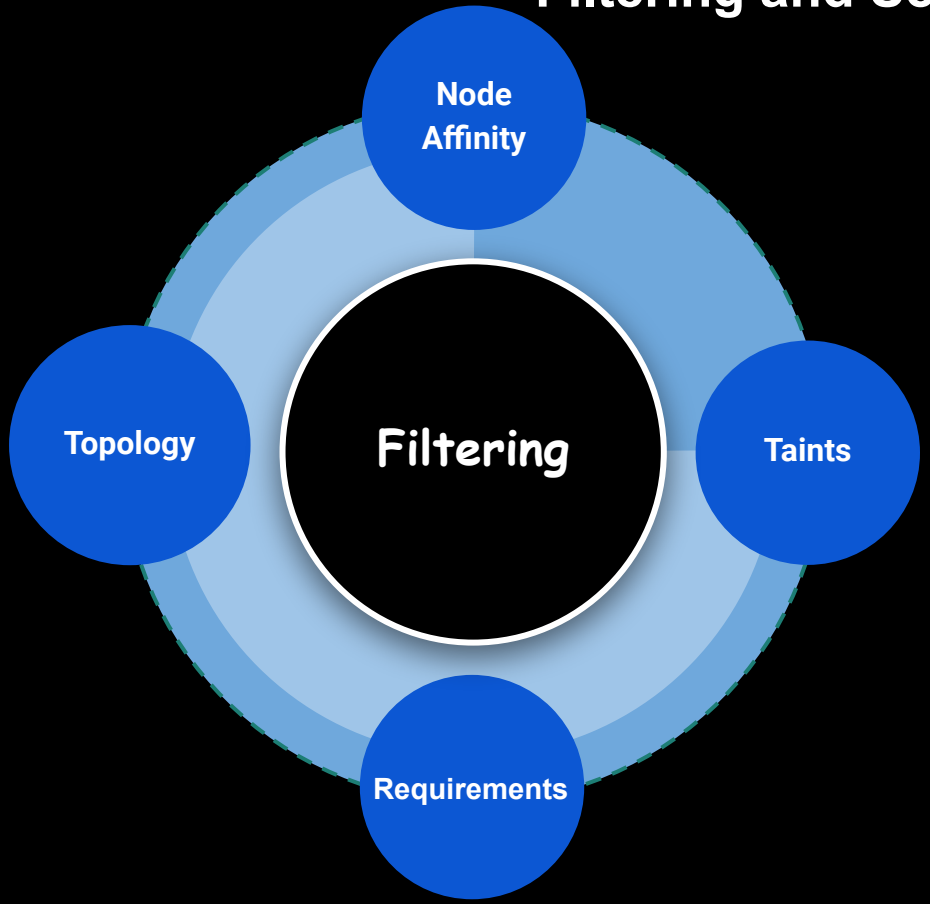
API Server — Entry point for interacting with the Kubernetes cluster

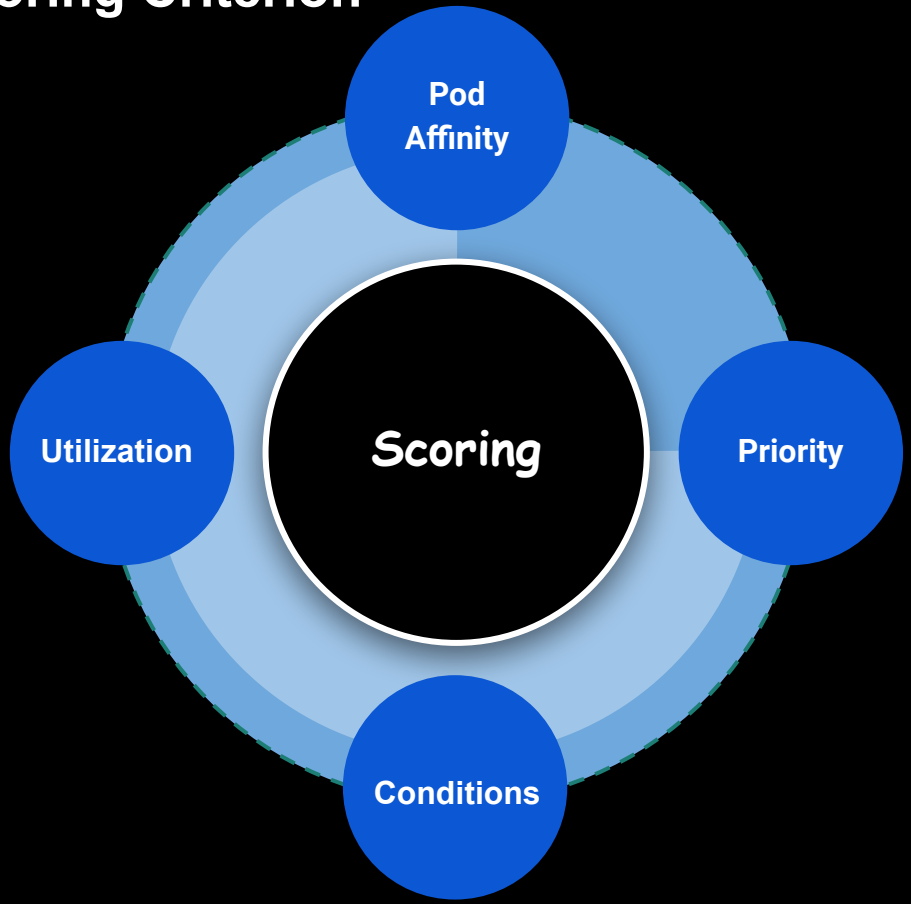
Extender — Allows addition of custom scheduling logic to enhance scheduler's capabilities.

Current Heuristic based algorithm used in Kube Scheduler



Filtering and Scoring Criterion





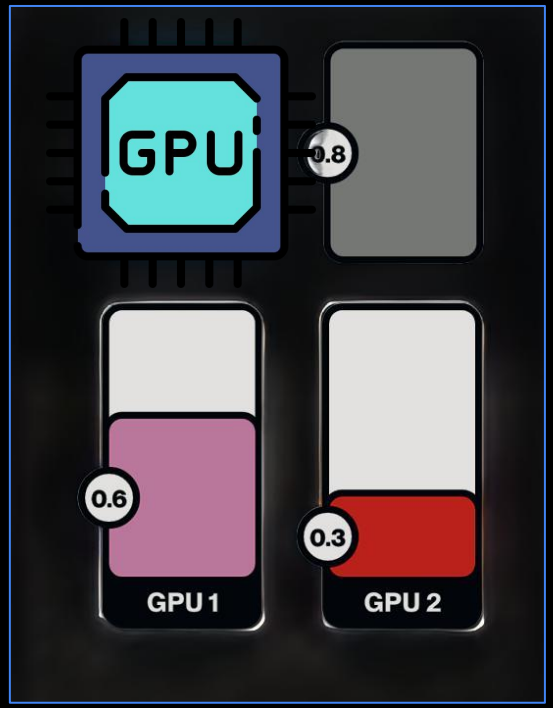
Kube
Scheduler
CHALLENGE

Resource Fragmentation

- Default scheduling leads to fragmentation.
- Small jobs consume entire GPUs, blocking larger jobs.

IMPACT

Inefficient GPU utilization with wasted capacity



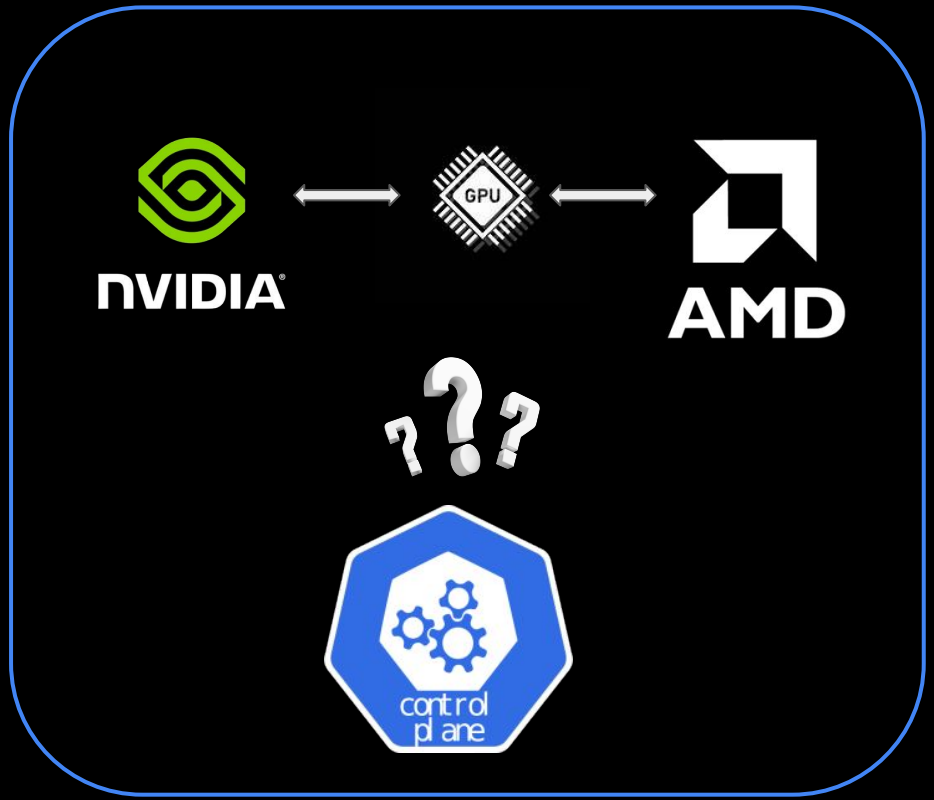
Limited Resource Sharing

- Kube-scheduler treats GPUs as integer units
- Inefficient utilization as workloads underuse GPU capacity.

IMPACT

Some workloads underutilize GPUs while others wait

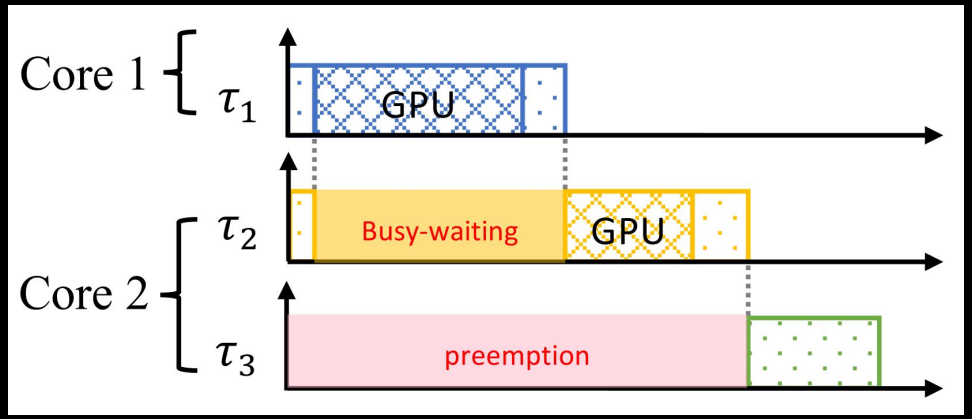
Heterogeneous Hardware Management



Kube-scheduler lacks built-in support for diverse GPU types.

Hence unable to manage mixed GPU environments

Lack of Preemptive Scheduling



High-priority jobs can not preempt lower-priority jobs.

Long-running jobs can monopolize GPU resources.

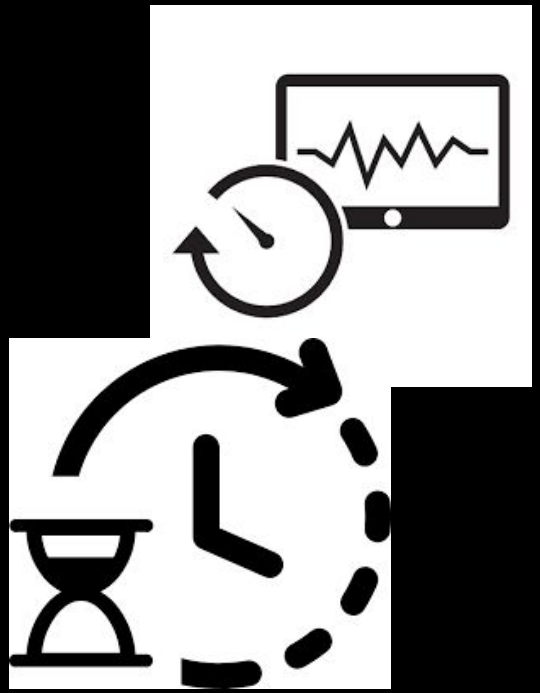
Impact - Increased latency for critical tasks

Limited Awareness of GPU Status

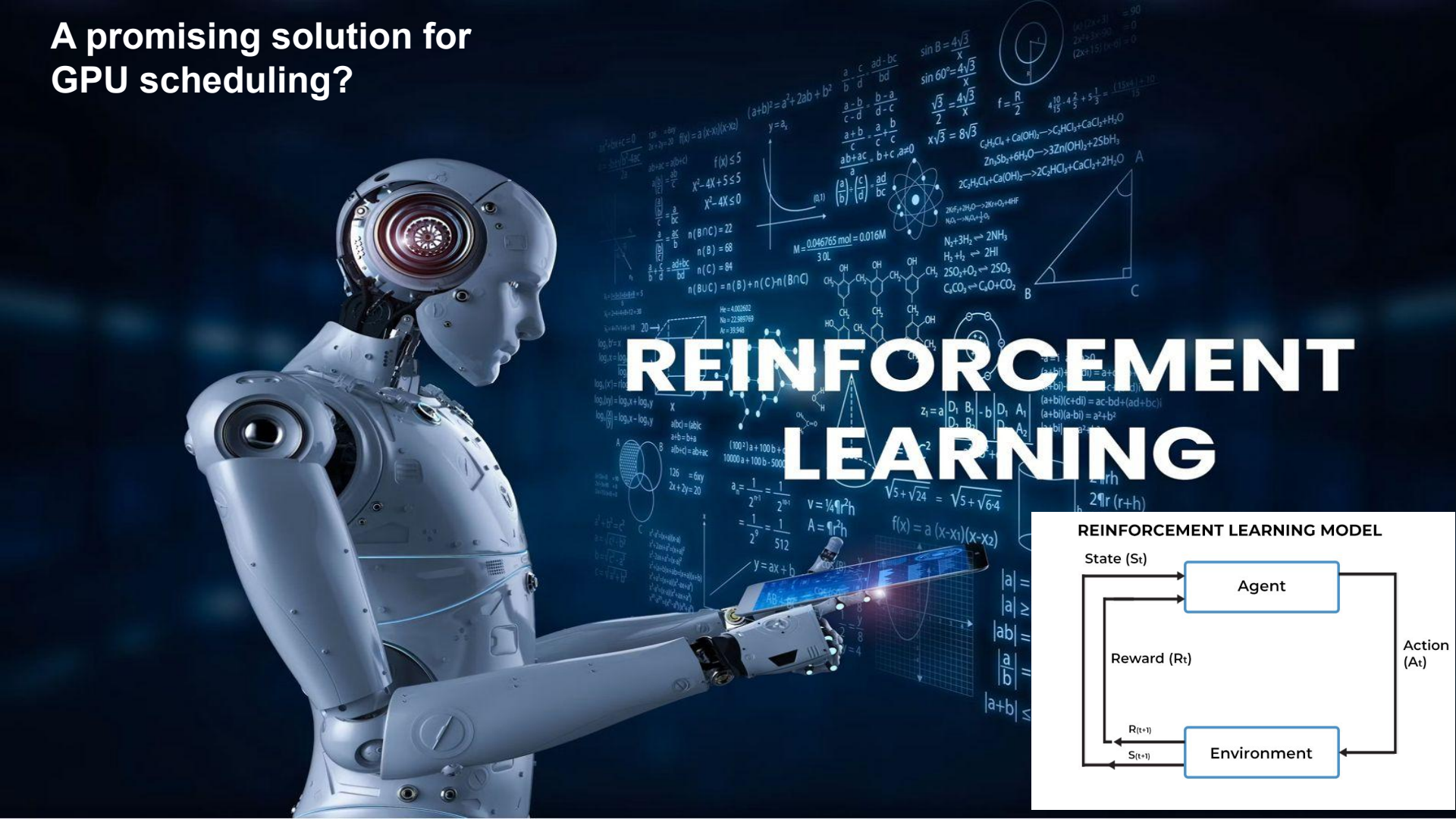
- Relies on static information for scheduling decisions.
- Poor awareness of real-time GPU health.

IMPACT

Scheduling delays and job failures due to dynamic changes in GPU availability



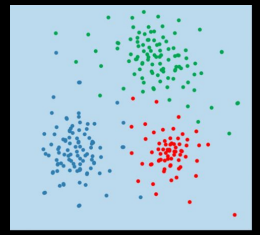
A promising solution for GPU scheduling?



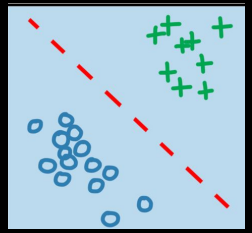
Let’s start with what is RL without going in Nitty-gritty

Machine Learning

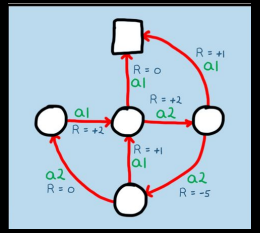
Unsupervised learning

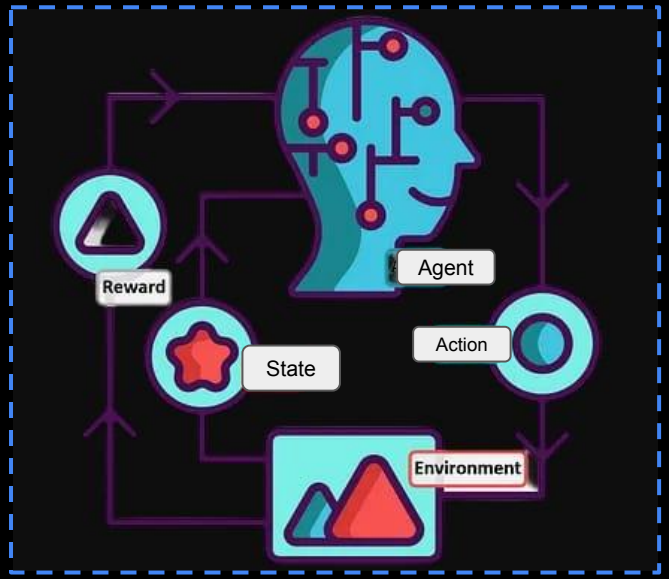


Supervised learning



Reinforcement learning





Why we think RL is a good option for scheduling GPUs??

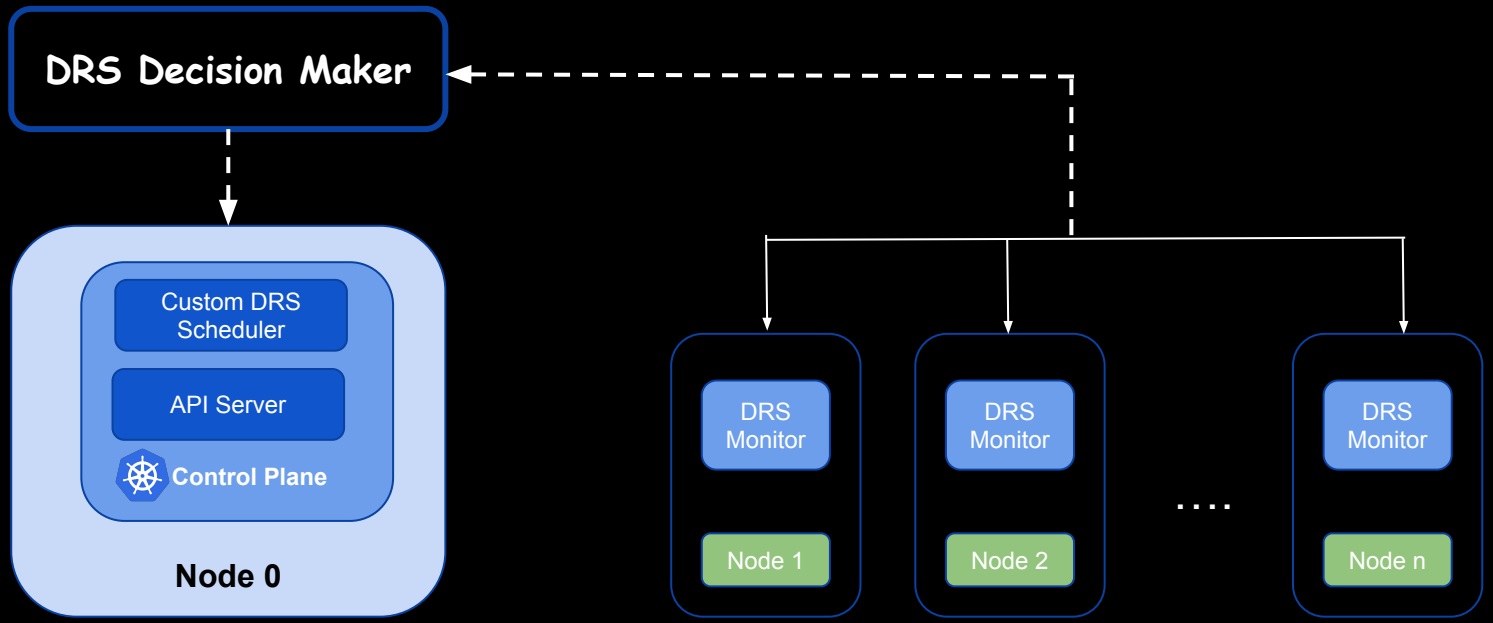
Optimization Across Multiple Objectives

Scalability with Complex State and Action Spaces

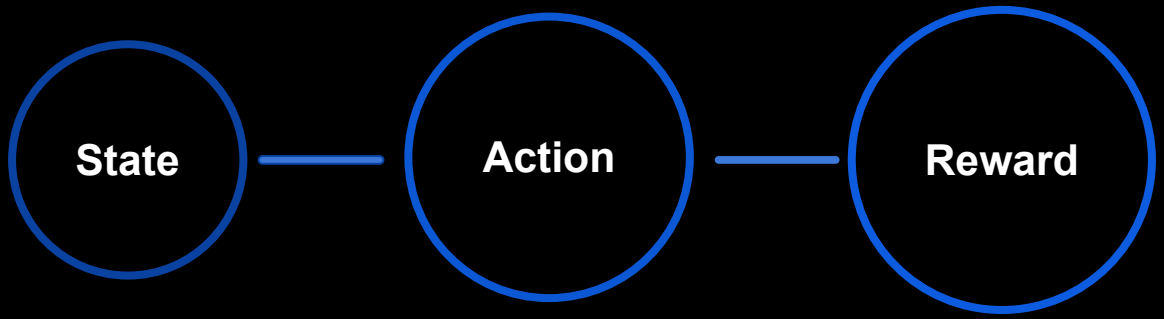
Feedback-Driven Improvement

Autonomous Decision-Making

Proposed DRS Model Architecture - Core Components



Modelling scheduling problem as a Markov decision process

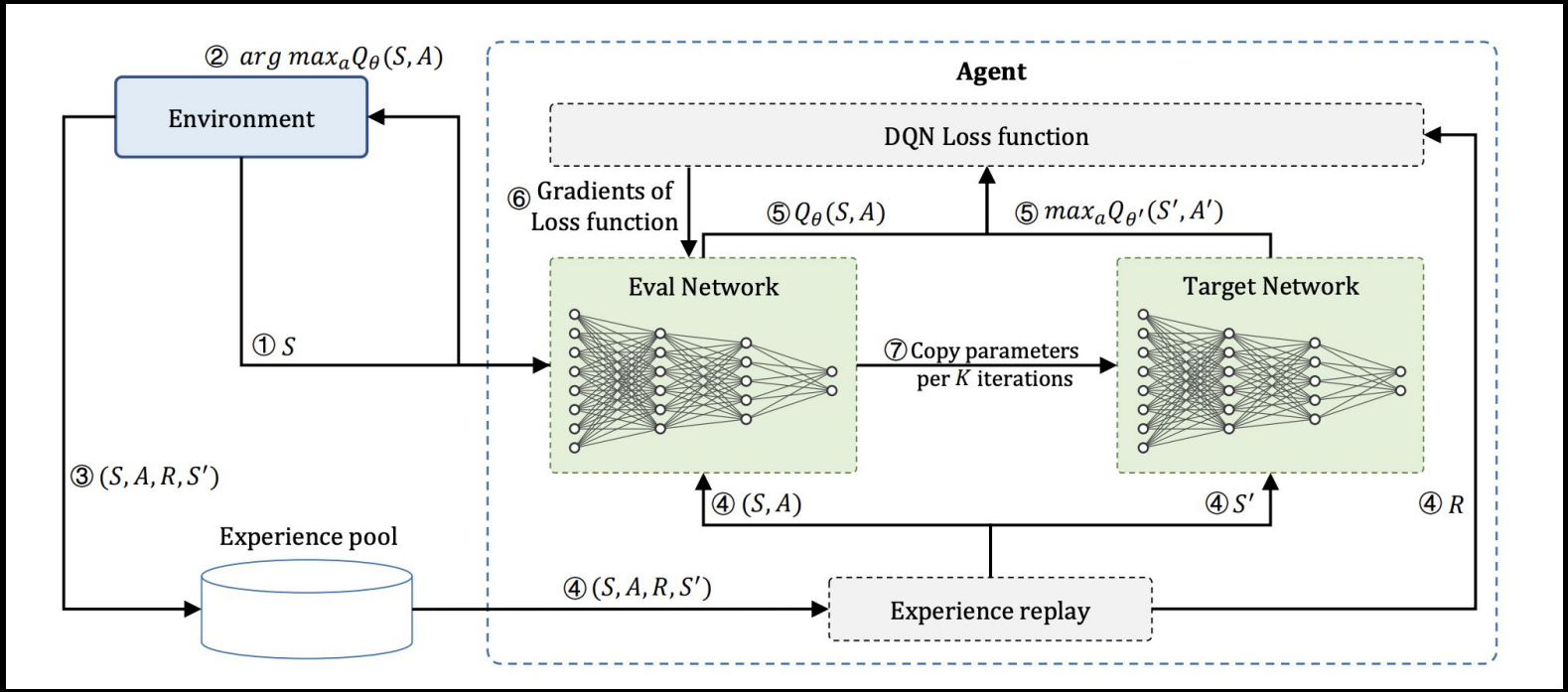


$$State_t = [Node1_t, Node2_t, ..., Noden_t, Pod_t]$$

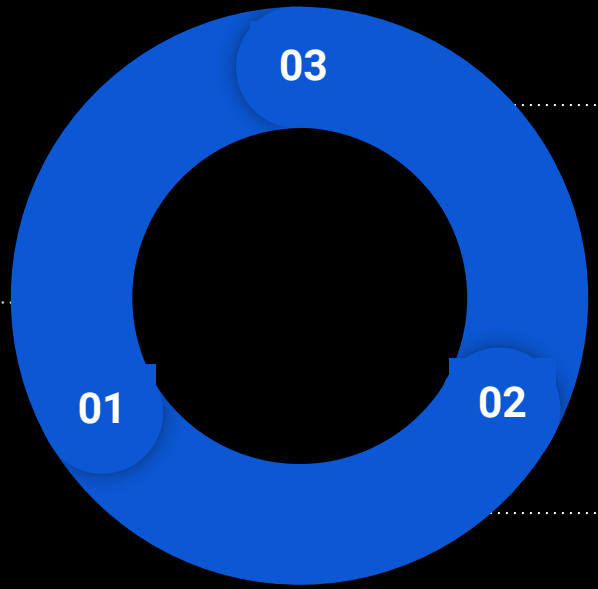
$$Action_t \in \{node_i \mid i = 1, 2, ..., n\}$$

$$Reward_t = \sum (Util_k), \; Util_k \in \{\text{GPU Metrics}\}$$

Discover Decision Maker



Training the decision maker - Details of Deep Q Learning Algorithm



Listen for scheduling requests from DRS scheduler

Request states and then calculate the "Q" Values

If experience pool D is full

Sample randomly and train the network with it

With probability ϵ, select the action with max Q-value

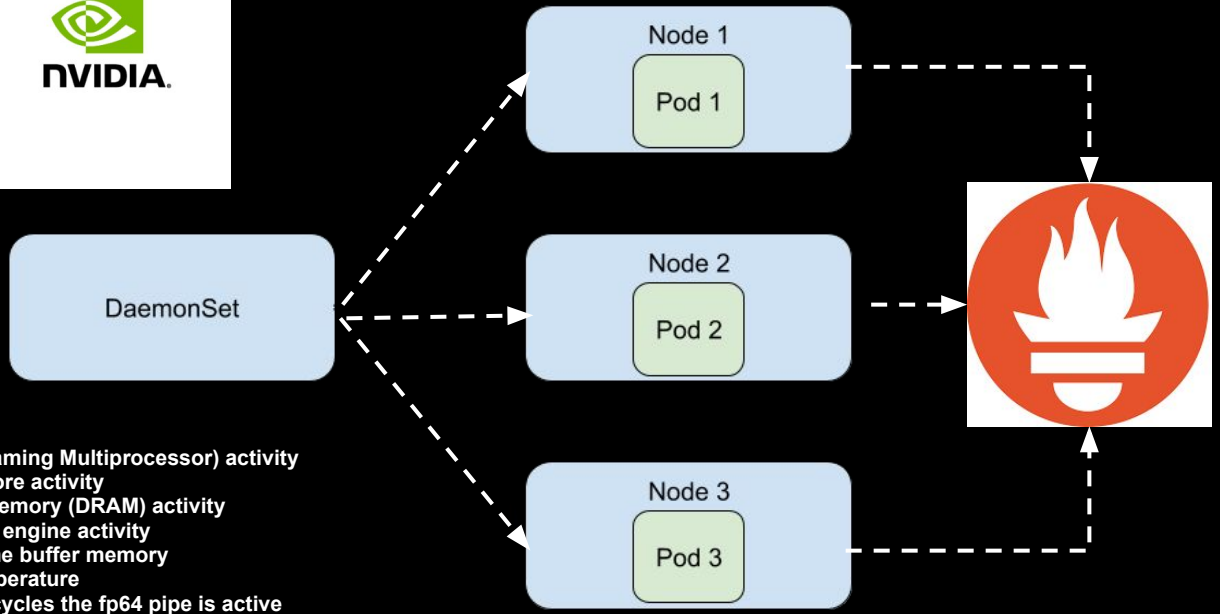
Send it to scheduler and obtain updated states and store in D

Initialize experience pool D with capacity N = 300
Initialize Q-function Q with random values
Establish socket connections with DRS scheduler and monitor

DRS Monitor

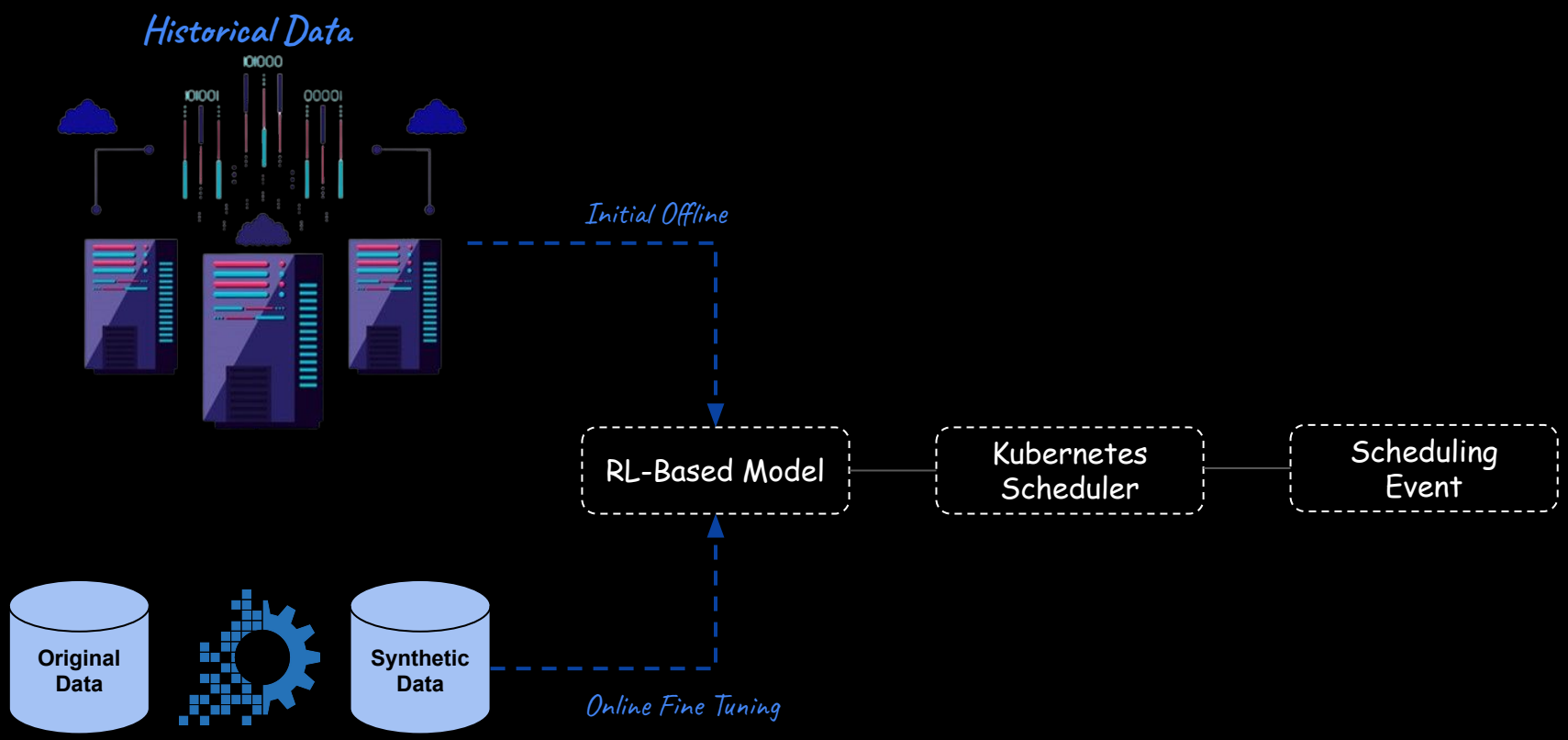
NVIDIA/**DCGM**

NVIDIA Data Center GPU Manager (DCGM) is a project for gathering telemetry and measuring the health of NVIDIA GPUs



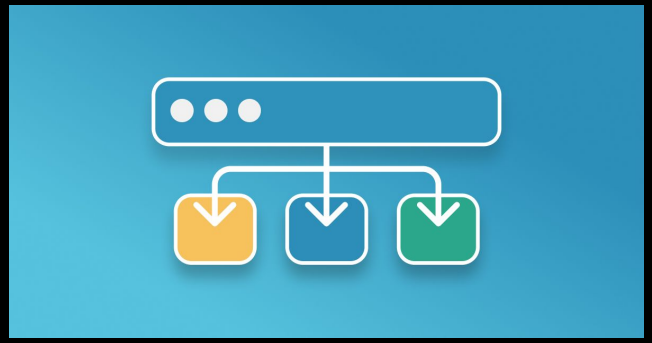
DCGM_FI_PROF_SM_ACTIVE	SM (Streaming Multiprocessor) activity
DCGM_FI_PROF_PIPE_TENSOR_ACTIVE	Tensor core activity
DCGM_FI_PROF_DRAM_ACTIVE	Device memory (DRAM) activity
DCGM_FI_PROF_GR_ENGINE_ACTIVE	Graphics engine activity
DCGM_FI_DEV_FB_FREE	Free frame buffer memory
DCGM_FI_DEV_GPU_TEMP	GPU temperature
DCGM_FI_PROF_PIPE_FP64_ACTIVE	Ratio of cycles the fp64 pipe is active

Training and Serving



But in what ways using RL-based GPU scheduler will help

Load Balancing and Overload Avoidance



Support for Mixed Workloads

Reduced Job Completion Time (Latency)

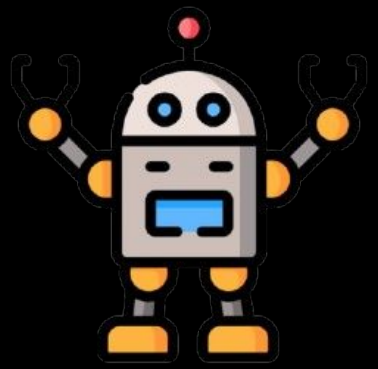
Energy Efficiency and Cost Savings

RL-Based Scheduling Benefits for CNCF Projects

CLOUD NATIVE COMPUTING FOUNDATION

	argo	Kubeflow	Knative
Current Approach	DAG-based workflow management	Kubernetes-native scheduling	Default demand based autoscaling
RL Benefits	Workflow optimization, dynamic resource allocation	Optimized job scheduling, improved pipeline efficiency	Predictive Autoscaling
Expected Change	Increased throughput, reduced latency	Reduced job completion time	Better performance , responsive autoscaling

Let's explore …



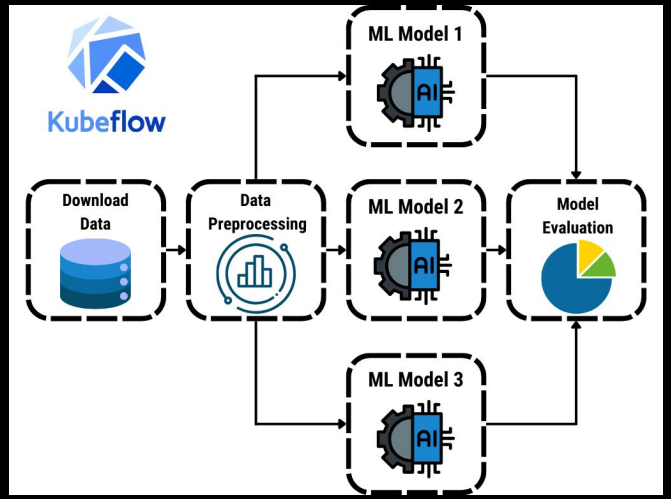
Reinforcement learning

+

Kubeflow

Kube flow Pipelines: Efficient GPU Allocation for Model Training

Concurrent
ML workflows

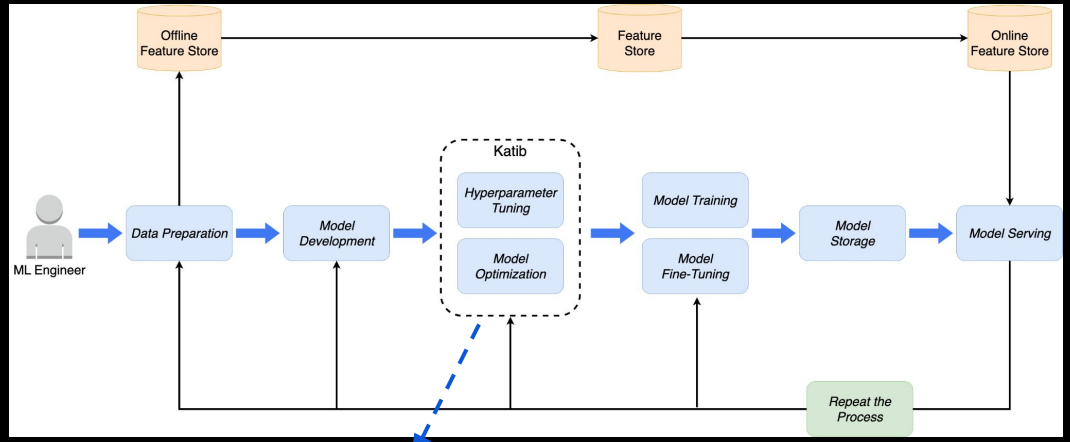


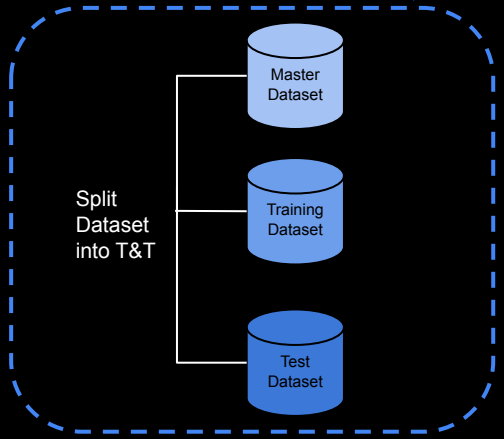
Traditional Scheduling
GPU bottlenecks when running concurrent ML workflows.

RL-Driven Improvement
Dynamic Resource Allocation adjusts GPU assignments based on real-time loads.

Outcome
Faster execution of GPU-intensive tasks, reducing overall workflow completion time.

Katib : Improved Hyperparameter Tuning Efficiency



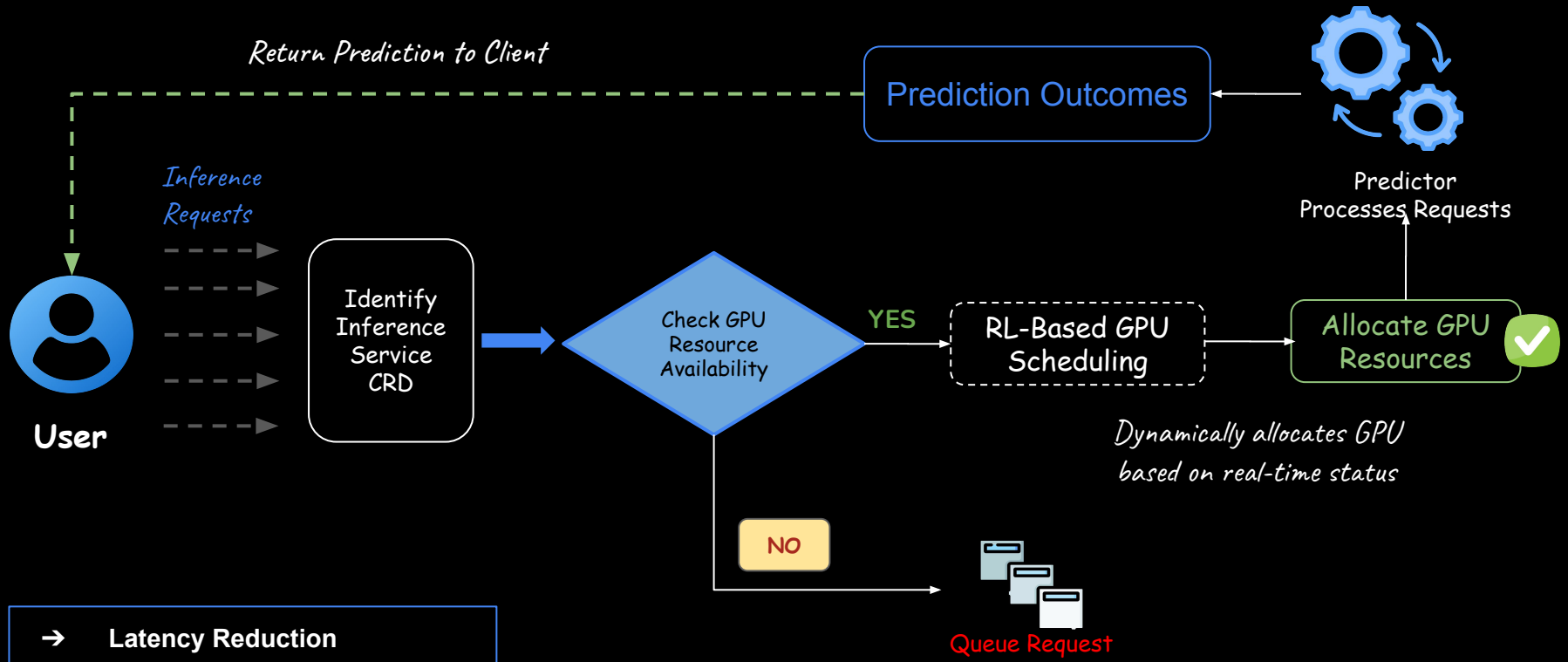


Traditional Scheduling
Overwhelming GPU resources from multiple experiments.

RL-Driven Improvement
Adaptive Scheduling allocates GPU resources based on experiment needs.

Outcome
Faster execution of GPU-intensive tasks, reducing overall workflow completion time.

KFServing: Reduced Inference Latency



- ➔ **Latency Reduction**
- ➔ **Dynamic Resource Allocation**
- ➔ **Cost Reduction**

CONCLUSION