KubeCon | CloudNativeCon

North America 2024

Gödel Scheduler

# Gödel: Unified Scheduler for Online and Offline Workloads

Lintong Jiang @ByteDance
Yue Yin @ByteDance

# Agenda



- Gödel Overview
  - Background & Architecture
- Key Features
  - Selected feature highlights and demos
- Achievements
  - Resource optimization at hyper-scale
  - Academic contributions
- Coming Soon
  - More optimizations & features

# Gödel Overview

# Hyper-Scale Infrastructure at ByteDance



- 500+ large-scale clusters worldwide
- ~20K heterogeneous servers in largest clusters
- ~1 millions of containerized tasks in largest clusters
- 200+ millions of containerized tasks running everyday

# Why Another Scheduler



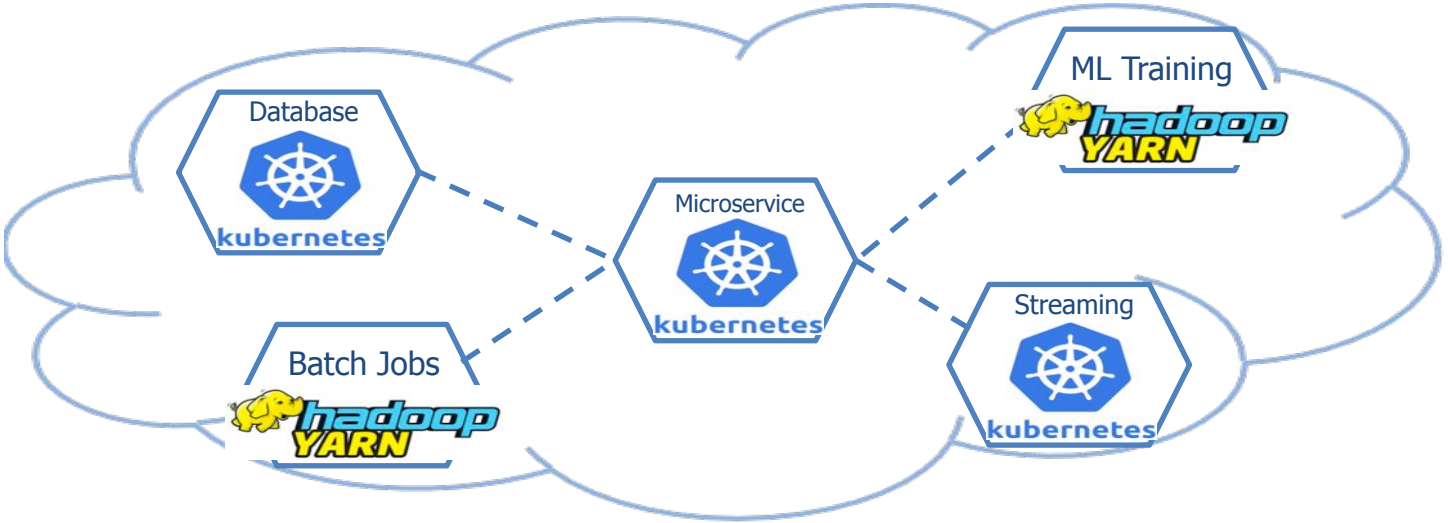
- Different business groups - heterogeneous workloads

| Applications | Critical | Performance Metrics | Topology-affinity |
|---|---|---|---|
| ML Training | No | **Completion time** | GPU |
| Batch Jobs | No | **Completion time** | N/A |
| Data Processing | No | **Throughput** | N/A |
| Video Coding | No | **Throughput** | N/A |
| FaaS | Yes | **Latency** | N/A |
| Inference | Yes | **Latency** | NUMA-node |
| Recommendation | Yes | **Latency** | NUMA-node |
| Micro-service | Yes | **Latency** | N/A |
| In-Memory DB | Yes | **Latency** | NUMA-node |
| Streaming | Yes | **Throughput** | Network-bandwidth |

# Why Another Scheduler



- Isolated compute infrastructures




Resource fragmentation

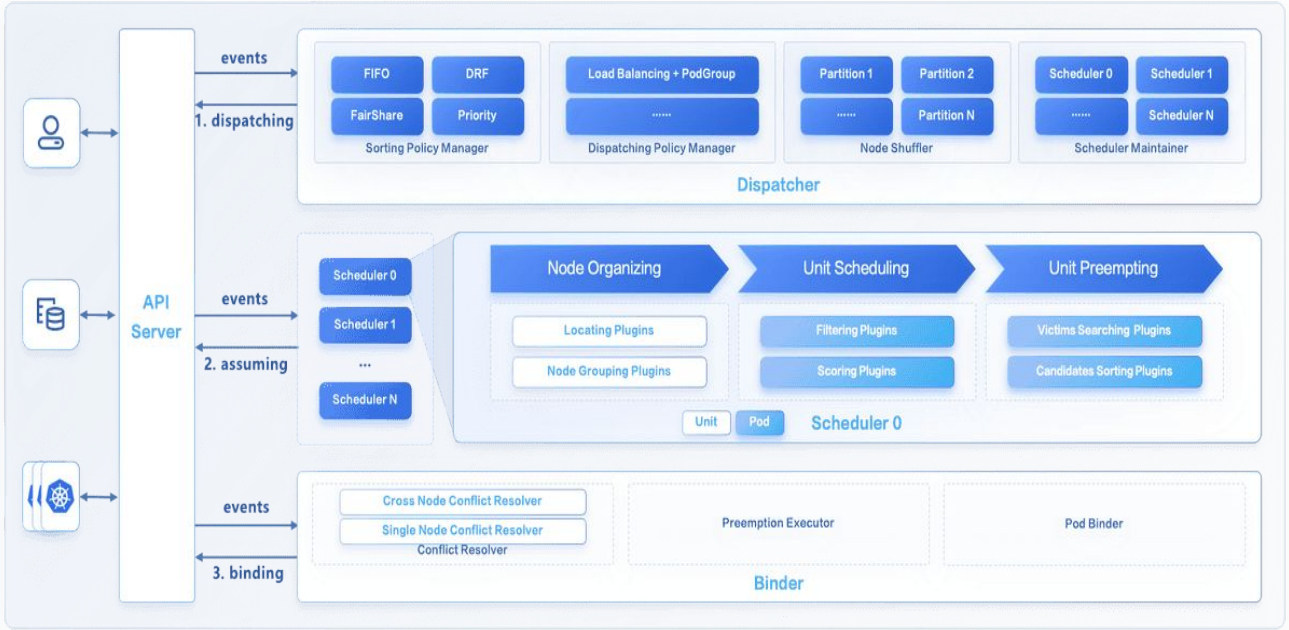
Low resource elasticity

Low resource utilization

# Godel Architecture

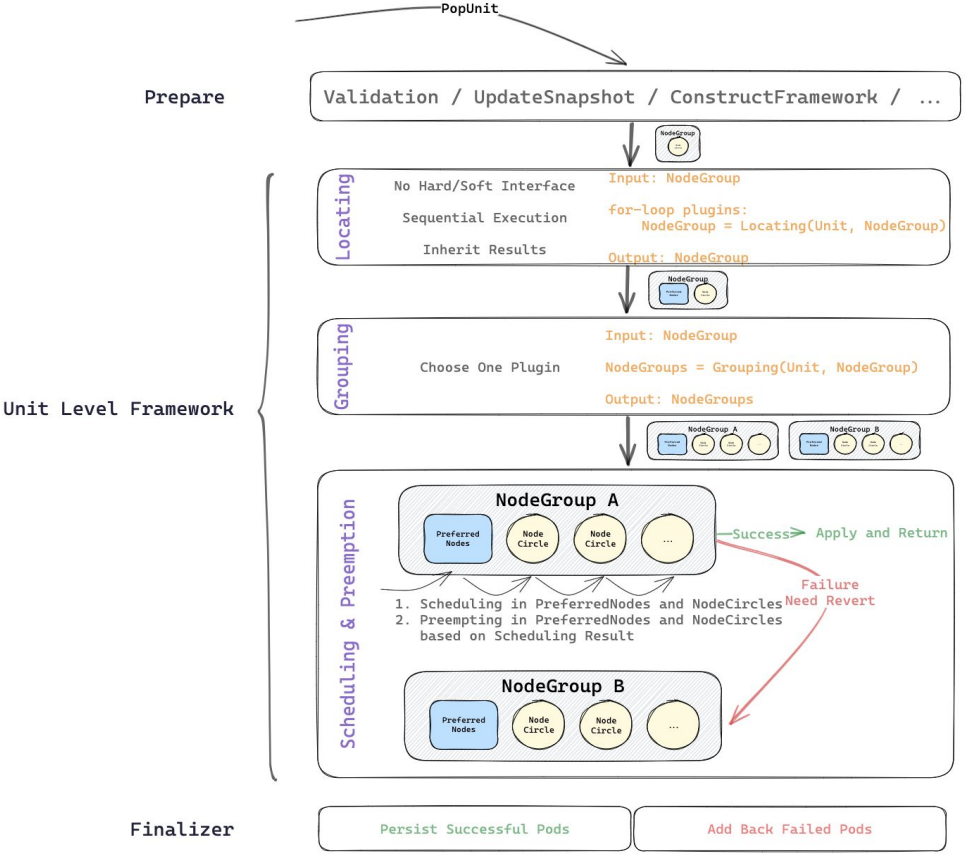


- Distributed Components with Shared States
- Optimistic Concurrency
- Two-layer Scheduling Abstraction



# Two-Layer Scheduling Abstraction

- Scheduling Unit Abstraction
  - e.g., PodGroup or Pod
- Two-Layer Scheduling Framework
  - Unit Level Framework
  - Pod Level Framework



# Scheduling at Hyper-Scale Infrastructure



- Service Level Objectives
  - Scheduling Latency P99 < 1 min
  - Scheduling Throughput <= 1000 pods/s
- Consideration for Large Clusters
  - No more than 20,000 nodes
  - No more than 1,000,000 total pods

# Key Features in Gödel

# Job-Level Affinity



- Gang Scheduling: All-or-Nothing
  - Pods can only be scheduled only if >= minMember can be scheduled at the same time
- Job-Level Affinity
  - *Preferred* and *Required* Affinity Terms
    - Soft & hard constraint for topology domain
  - Crucial for workloads such as machine learning jobs
    - Pods have to be scheduled within a specific network topology

```yaml
apiVersion: scheduling.godel.kubewharf.io/v1alpha1
kind: PodGroup
metadata:
  generation: 1
  name: demo-podgroup
spec:
  minMember: 10
  scheduleTimeoutSeconds: 3000
  affinity:
    podGroupAffinity:
      preferred:
        - topologyKey: micronet
      required:
        - topologyKey: mainnet
```

# Demo: Job-Level Affinity

```
➜  godel-scheduler git:(main)
➜  godel-scheduler git:(main)
➜  godel-scheduler git:(main)
```

# Job-Level Affinity - Continue



- Node Selector
  - Similar to the Pod Spec NodeSelector, allowing for scheduling based on specific node attributes

- Sort Rules
  - SortRules can be set to sort based on resource dimensions like GPU, CPU, MEM
  - Reduce resource fragmentation

```yaml
apiVersion: scheduling.godel.kubewharf.io/v1alpha1
kind: PodGroup
metadata:
  generation: 1
  name: demo-podgroup
spec:
  minMember: 10
  scheduleTimeoutSeconds: 3000
  affinity:
    podGroupAffinity:
      nodeSelector:
        nodeSelectorTerms:
          - matchExpressions:
              - key: machine
                operator: In
                values:
                  - machineA
      sortRules:
        - order: Ascending
          resource: GPU
        - order: Ascending
          resource: CPU
        - order: Ascending
          resource: Memory
```

# Resource Reservation



- During off-peak hours, to optimize resource utilization, we allow lower-priority tasks to **'borrow'** resources from online services
- However, we need these resources to be available for online services **as quickly as possible** whenever demand rises
- The high efficiency of **online services scaling-up** can be achieved through **Resource Reservation**

# Demo: Resource Reservation

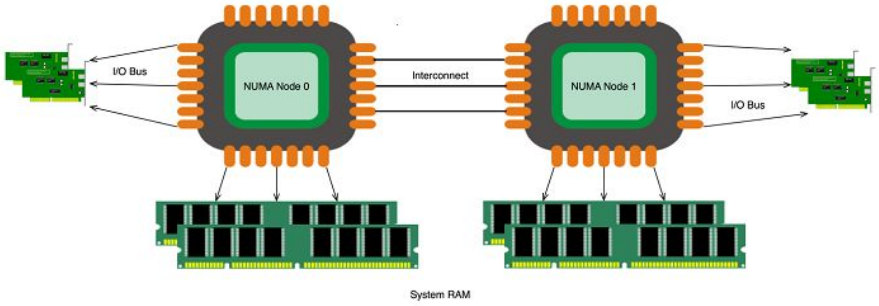
```
➜  godel-scheduler git:(main)
➜  godel-scheduler git:(main)
```

# Micro-Topology Scheduling



- Latency-sensitive workloads require that their pods be scheduled **within the same NUMA node** to minimize memory access latency that can arise from cross-NUMA access
  - Micro-Topology plugins of Godel

- Depends on **node agent** which can report numa-level resources
  - At ByteDance, we use [Katalyst] (also open-sourced) for its capability to report numa-level resource information



# RealTime Load-Aware Scheduling



- Traditionally, schedulers determine if a node has sufficient CPU by comparing the available CPU on a node with the pod's requested CPU, specified as a **static** configuration in the pod's spec

- However, the **requested** CPU sometimes differs significantly from the pod's **actual** CPU **usage** in real time

- Godel scheduler can filter out nodes whose loads exceed a specified threshold (e.g., 80%) and **prioritize** scheduling pods on nodes with **lower** CPU **usage**
  - Depends on node agent to report real-time load usage data. At ByteDance, we use Katalyst

# Katalyst-Related Session @KubeCon



- Highly Recommended!

Service Profiling Based Management and Scheduling in K8s - Jia Deng, Cong Xu & Mingmeng Luo, Bytedance

Friday November 15, 2024 4:55pm - 5:30pm MST

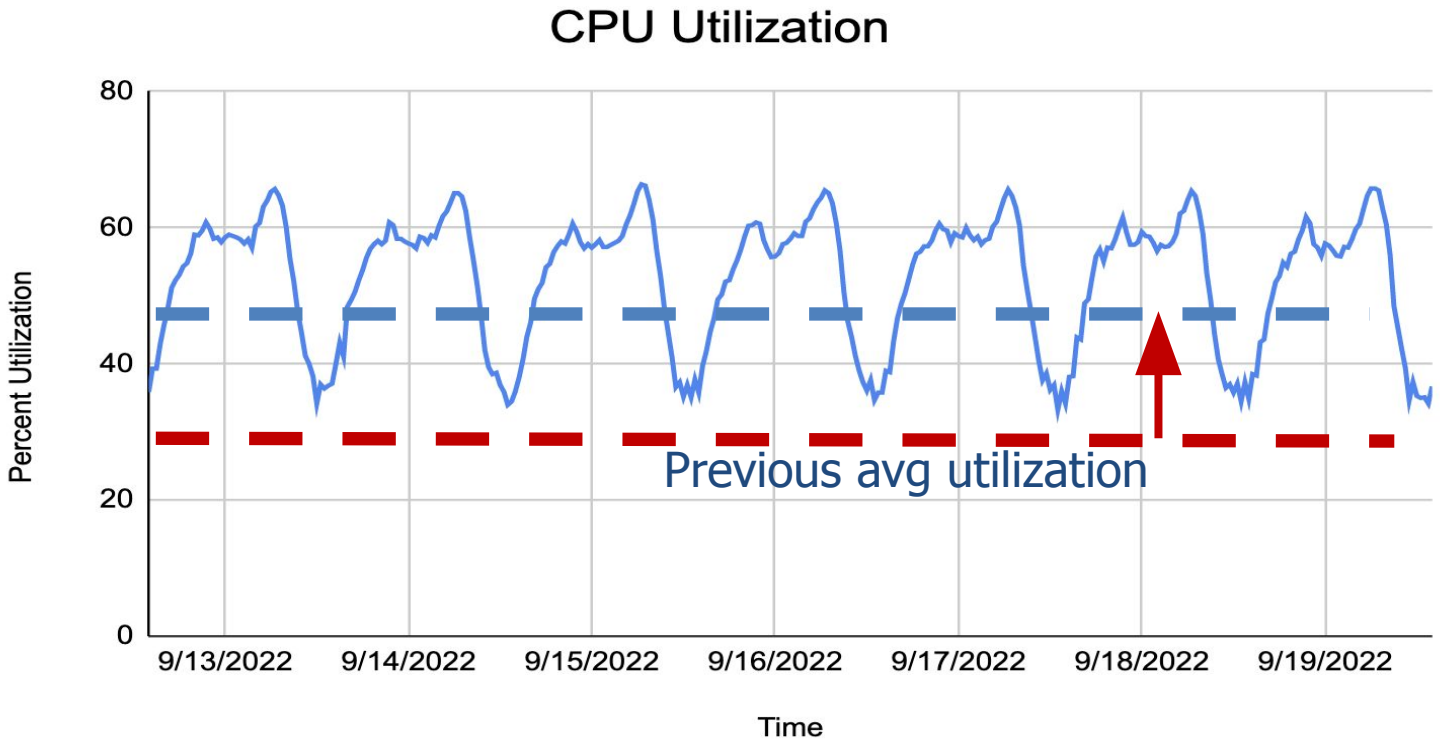
Salt Palace | Level 1 | 155 B

# Achievements

# Resource Utilization



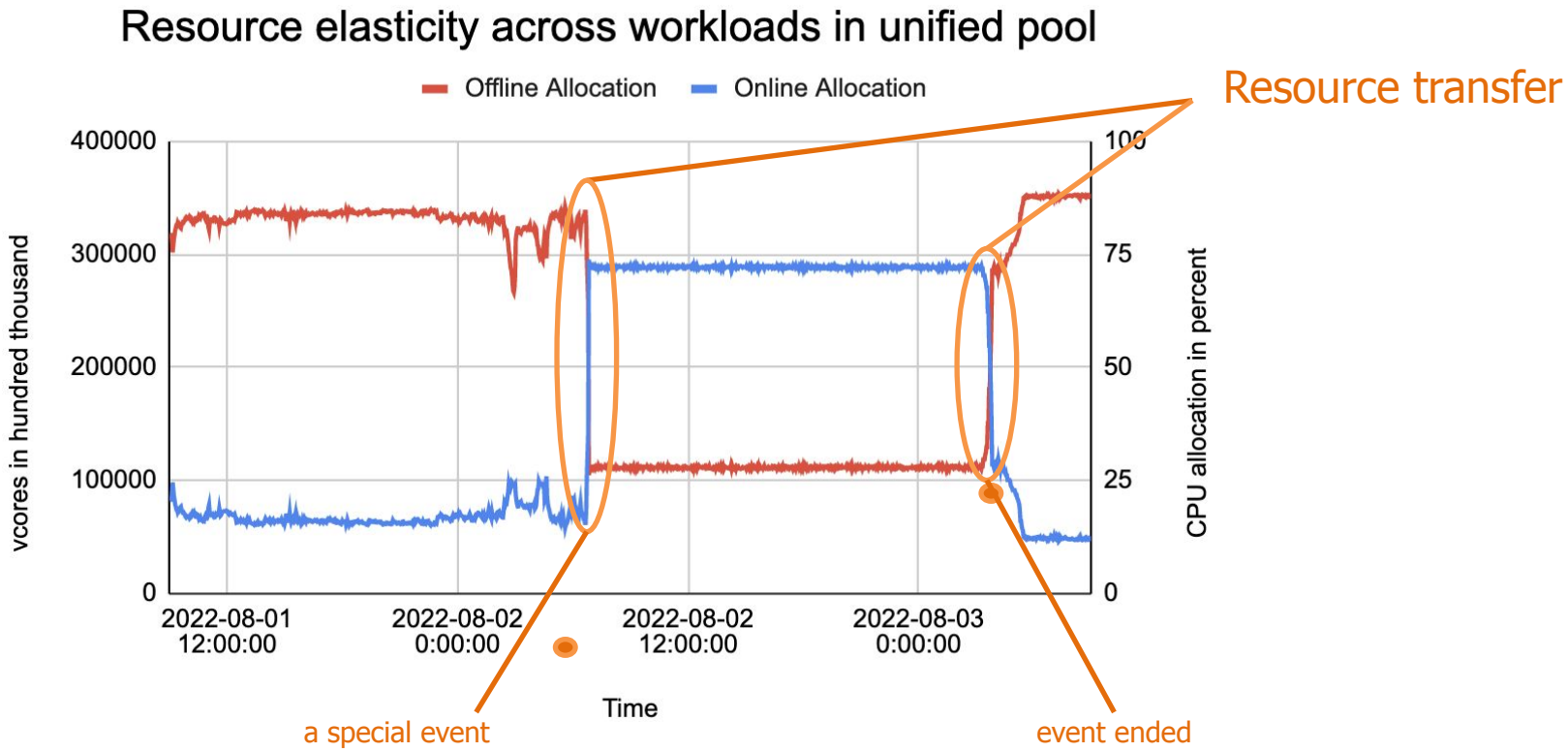
- Up to 60% CPU utilization in an ultra-large cluster
- Around 50% by average vs 30% by average before



# Resource Elasticity



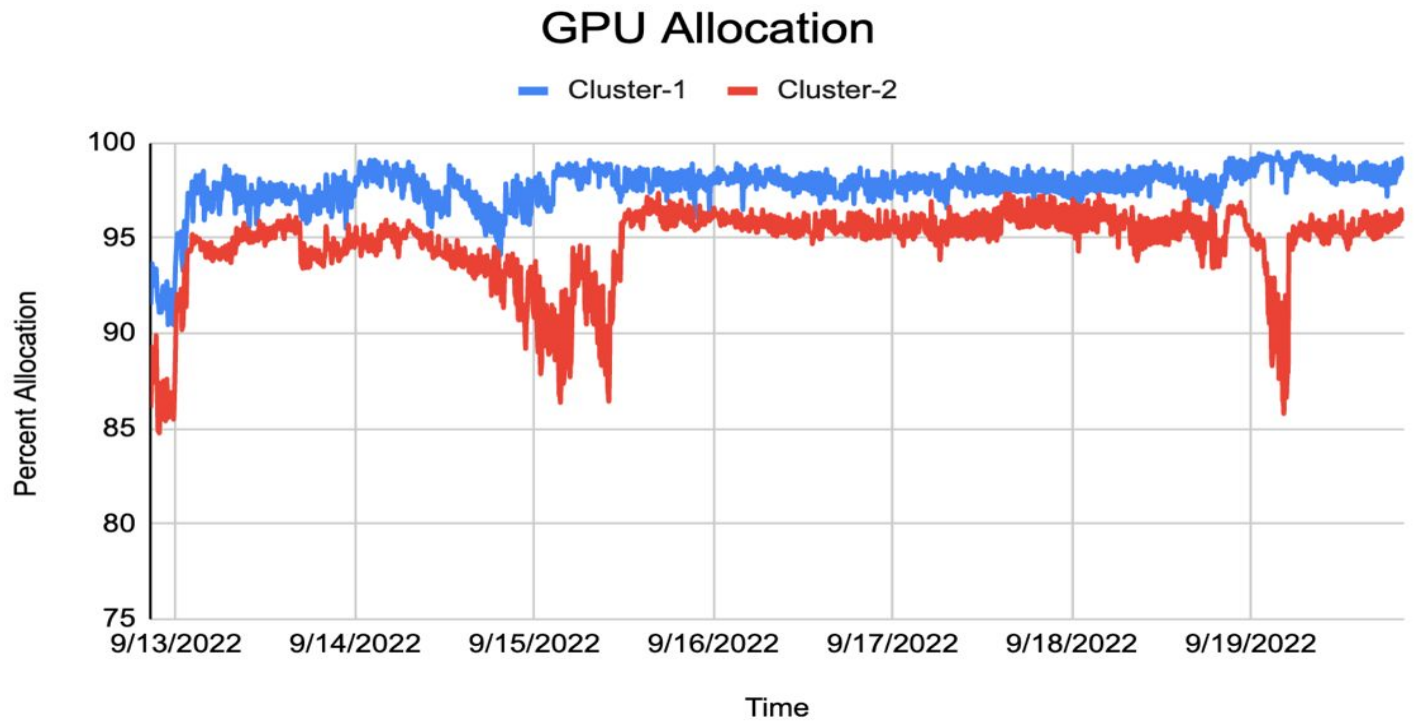
- Fast & automatic resource transfer between online/offline



# Resource Fragmentation



- 20% less fragmentation of GPU Pods
- Dropped from 30% to around 10%



# Academic Contributions



- SOCC'23: Gödel: Unified Large-Scale Resource Management and Scheduling at ByteDance

- Under Review: A paper about performance enhancement in Godel

# Coming Soon...

# Coming Soon...



- Gödel Rescheduler
  - Work with Godel scheduler to reschedule pods based on different strategies, such as GPU Bin-Packing
- All-in-One Mode
  - Designed for smaller scale cluster. Easier to deploy and maintain, while still benefiting from Godel (high throughput, rich functionalities, etc)
- More Optimizations
  - Further performance optimization;
  - More standardized and extensible framework, etc.
- Eco-System Construction
  - Support popular framework such as Pytorch and TensorFlow by providing compatible APIs

# Thank You!!

**godel-scheduler**

https://github.com/kubewharf/godel-scheduler