
YuniKorn 对 Flink on K8s 的调度优化
Optimize Running Flink on K8s with YuniKorn
Weiwei Yang CLOUDERA
Tao Yang Alibaba Cloud
FLINK FORWARD # ASIA
实时即未来 # Real-time Is The Future
FLINK FORWARD

FLINK FORWARD

Flink on K8s 现状

Flink on K8s Now

资源调度优化

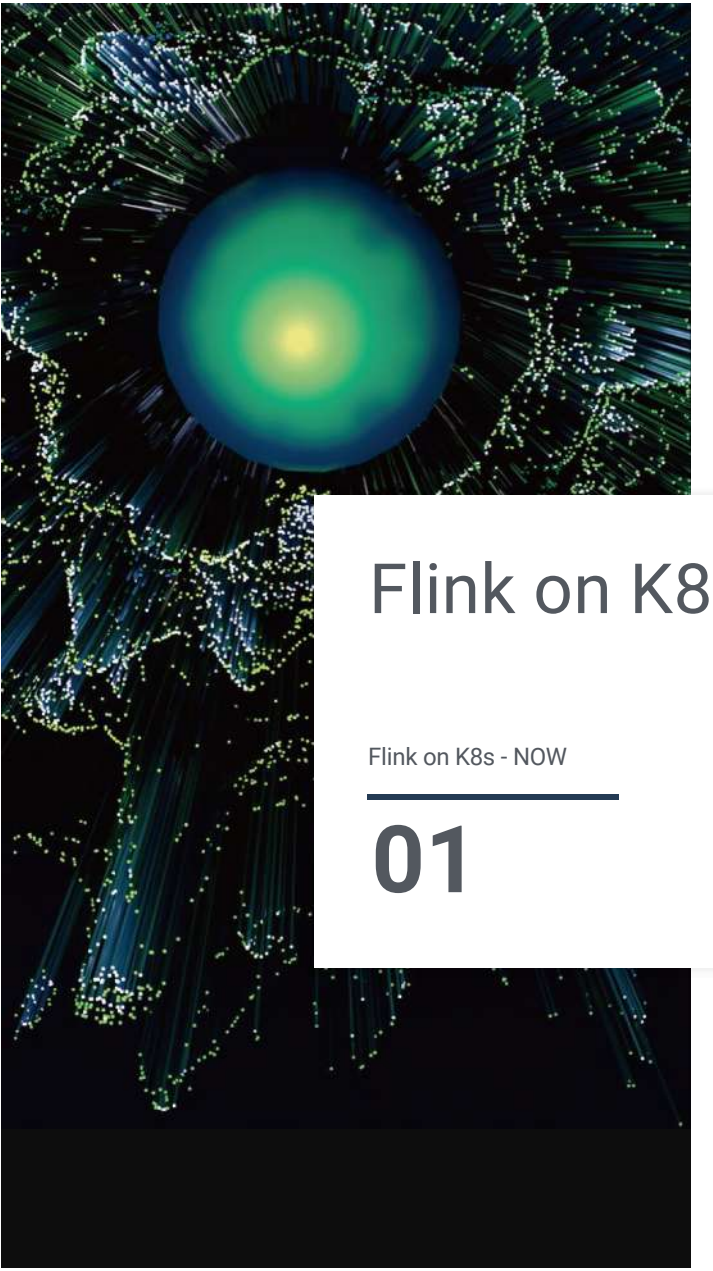
Optimization of Resource Scheduling

深入了解 YuniKorn

YuniKorn Deep Dive

未来发展与思考

Roadmap and Future



Flink on K8s 现状

Flink on K8s - NOW

01

Flink on K8s 部署

Flink Deployment on K8s

不支持弹性伸缩

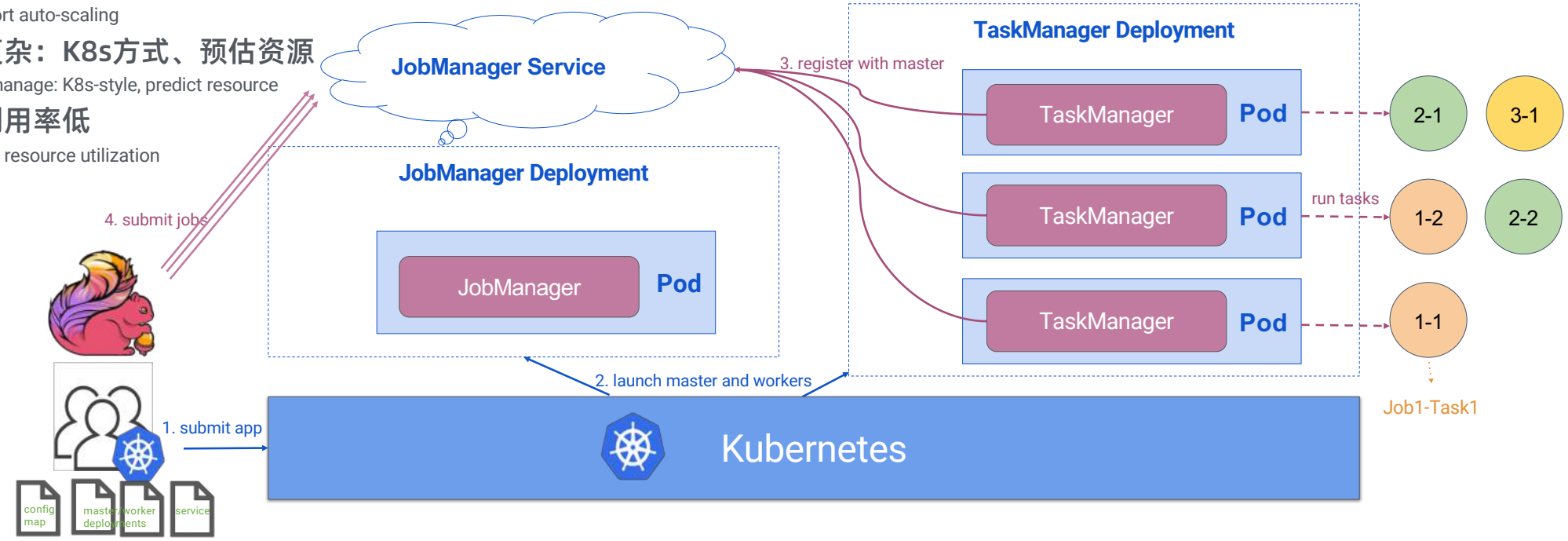
Not support auto-scaling

管理复杂：K8s方式、预估资源

Hard to manage: K8s-style, predict resource

资源利用率低

Inefficient resource utilization



简化作业管理：Helm Chart / K8s-flink-operator

Simplify management: Helm Chart / K8s-flink-operator

Flink 与 K8s 的原生集成

Flink natively integrate with K8s

FLIP-6 FLINK-9953

灵活易用

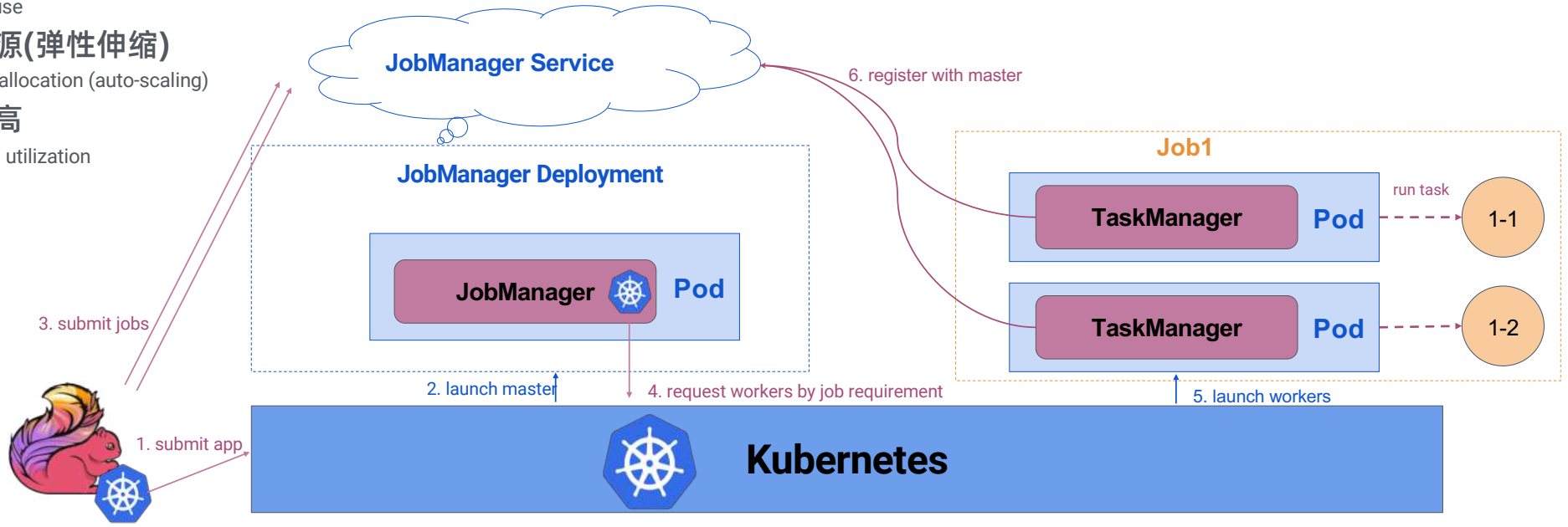
flexible & easy-to-use

动态申请资源(弹性伸缩)

dynamic resource allocation (auto-scaling)

资源利用率高

efficient resources utilization



面向短任务的改进

Improvement for short-time jobs

弹性伸缩

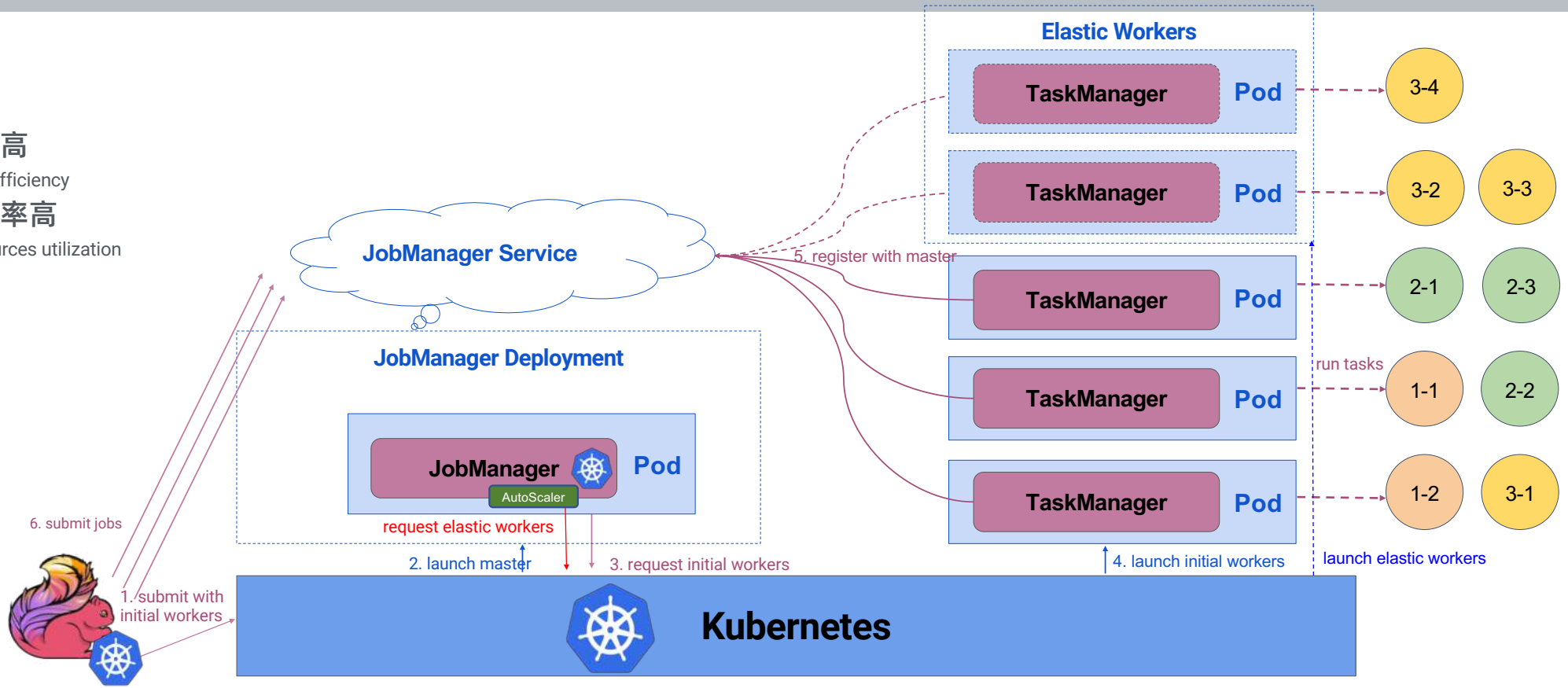
auto-scaling

运行效率高

high running efficiency

资源利用率高

efficient resources utilization



基于K8s的阿里云 Flink 计算平台

Computing Platform of Flink on K8s in Alibaba Cloud

Flink on Private Cloud

专用集群

dedicated cluster

持续使用

constantly in use

Flink Serverless

共享集群

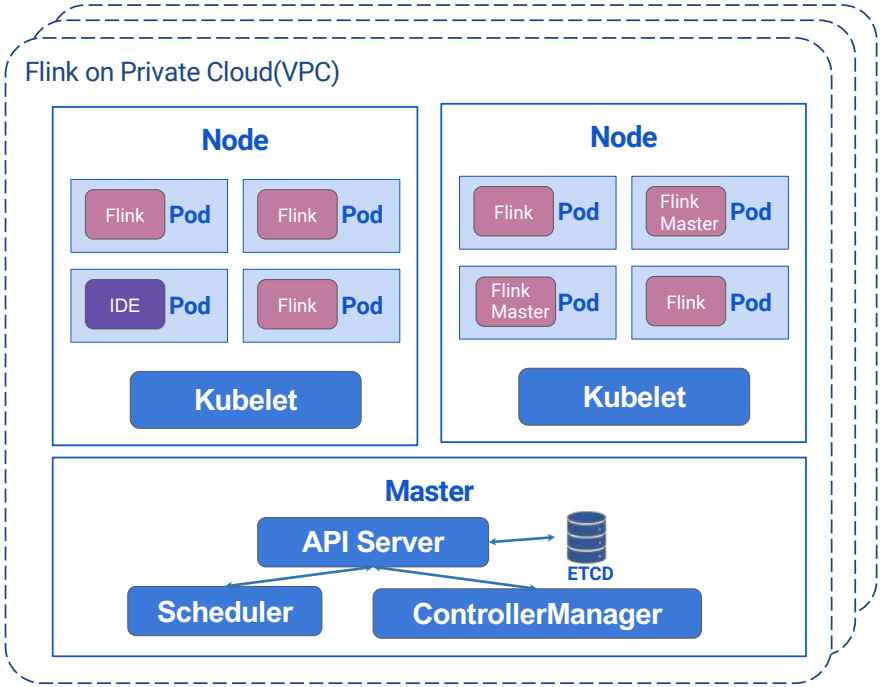
shared cluster

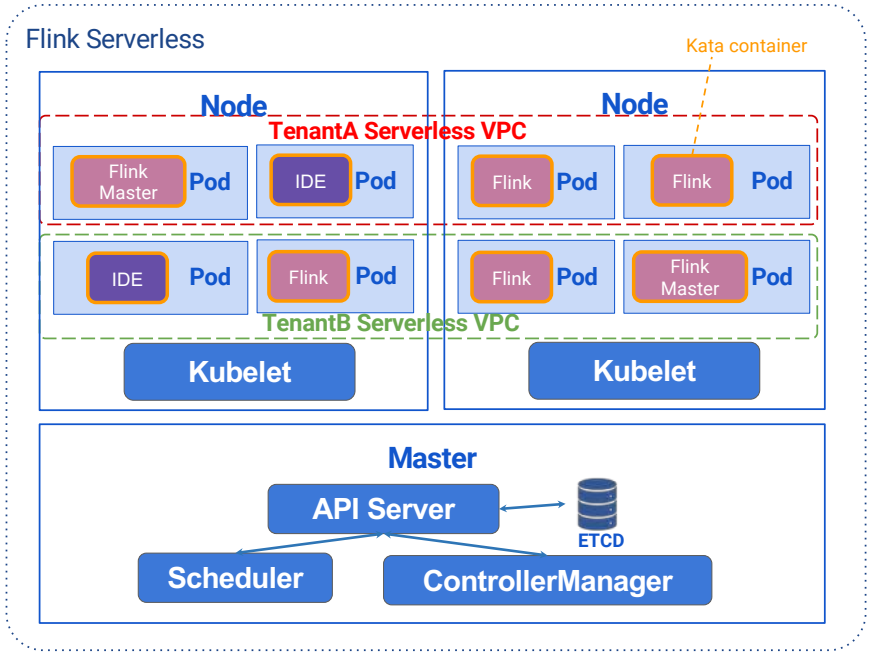
随用随取

use on demand

安全容器

kata container





K8s 资源调度问题

Resource scheduling problems on K8s

- 资源调度管理灵活性低，难以支持多租户场景对资源公平性、调度有序性和弹性资源管理的需求

  Low flexible for resource scheduling and management, hard to support resource fairness, scheduling order and elastic resource management for tenants
- 调度性能（吞吐）低

  Low scheduling throughput
- 作业信息分散，没有统一管理

  No unified management for workloads

资源调度优化

Optimization of Resource Scheduling and Management

02

基于 YUNIKORN 的资源调度优化

Optimize resource scheduling on K8s with YuniKorn

YuniKorn (/ˈjunɪˌkɔrn/, Y for YARN, K for K8s, uni- for Unified)

- **轻量级, 通用的资源调度器**

  A light-weighted, common resource scheduler

- **可适配各种资源管理系统** (YARN, K8s, etc)

  Platform independent

- **优化了对大数据作业的支持**

  Enhanced scheduling capabilities for Big Data workloads

如何解决 K8s 的资源调度问题

How to resolve resource scheduling problems on K8s

NOW

资源调度管理灵活性低

Low flexible for resource scheduling and management

调度性能（吞吐）低

Low scheduling throughput

作业信息分散，没有统一管理

No unified management for workloads

YUNIKORN

支持多种调度策略 (平铺/聚拢)
使用队列管理多租户资源，支持多级队列、资源公平性、调度有序性和弹性资源管理

Support multiple scheduling policies(Fair/Bin-packing), use queues to manage resources of tenants, support multi-level queues, resource fairness, scheduling order among workloads and elastic resource management.

多线程异步调度、批量分配
吞吐可达 1000 pods/s (<1k nodes)

Async-scheduling in multiple threads, batch allocations,
throughput reaches 1k+ pods/s when the number of nodes belows 1k

提供队列/作业管理的集中视图
完善的 Metrics

Provide views for queues/workloads, well-exposed metrics

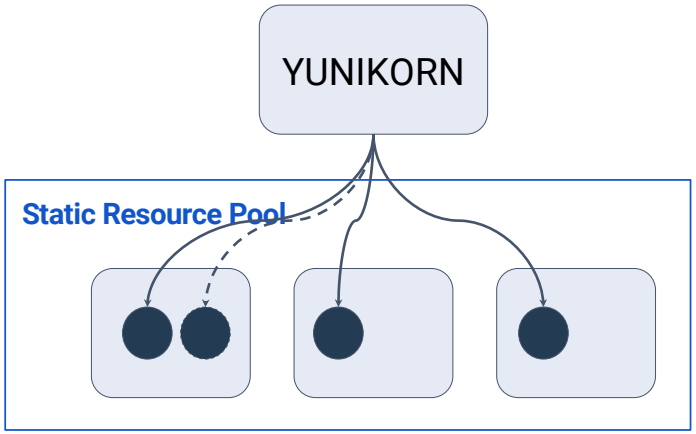
灵活的资源调度管理

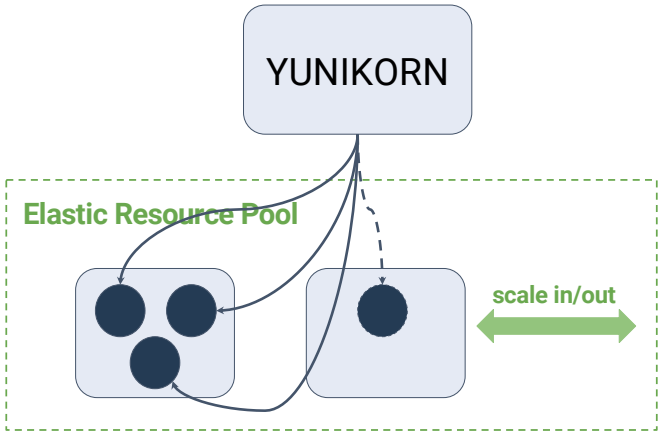
Flexible Resource Scheduling Management

调度策略

Scheduling policies

- **平铺 (Fair)** - 适用于静态资源池场景，如数据中心
  FAIR - applicable for static resource pool: data center etc.
- **聚拢 (Bin-packing)** - 适用于弹性资源池场景，如云服务器自动扩缩容
  Bin-packing - applicable for elastic resource pool: auto-scaling of servers on cloud etc.





平铺与聚拢调度策略

Evaluation of FAIR and BIN-PACKING Scheduling Policies

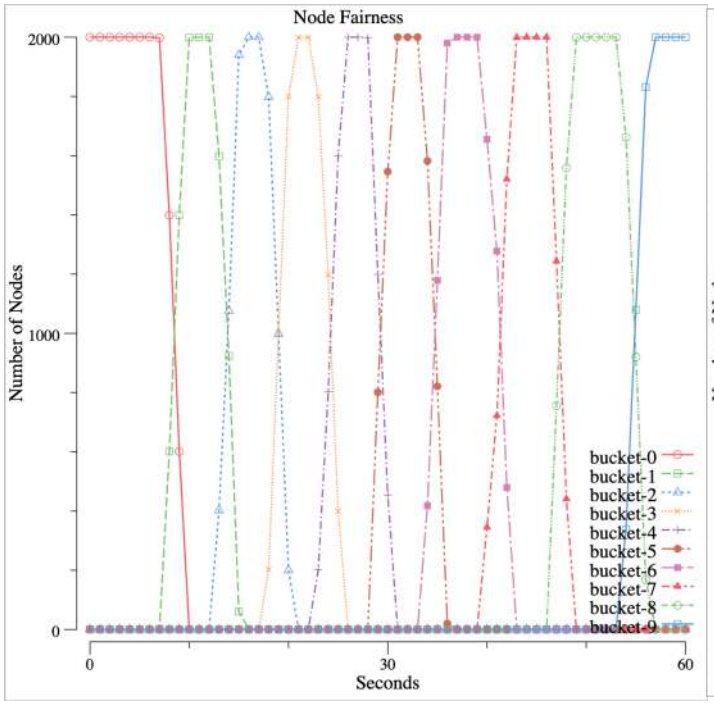
Prepare kubemark cluster with 2,000 hollow nodes(running on 20 real nodes), schedule 20,000 pods to use up all of the cluster's resource.

bucket-N metric represents the number of nodes whose resource usage ratio is in the range between 0.1*N and 0.1*(N+1).

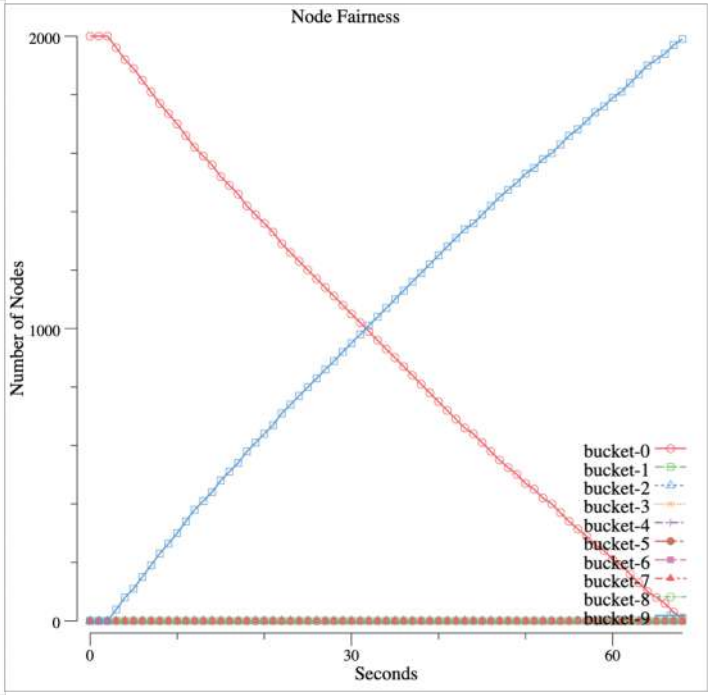
➔ FAIR - allocate resource evenly on nodes

➔ Bin-packing - allocate resource node by node

Fair



Bin-packing

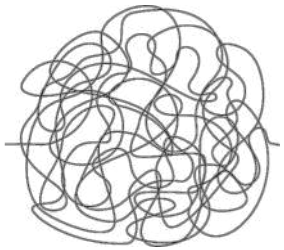


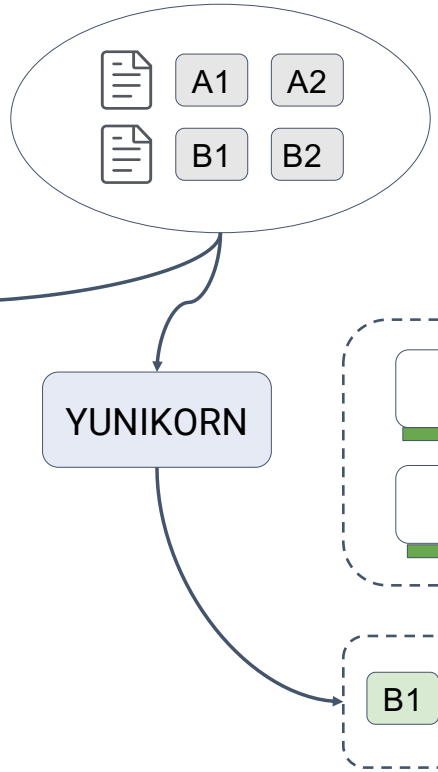
灵活的资源调度管理

Flexible Resource Scheduling Management

FIFO选取先调度的资源请求，未考虑作业，用户或者用户组信息，无公平性保证

choose pod request in FIFO order without considering workloads, user or user group, no fairness guarantee.



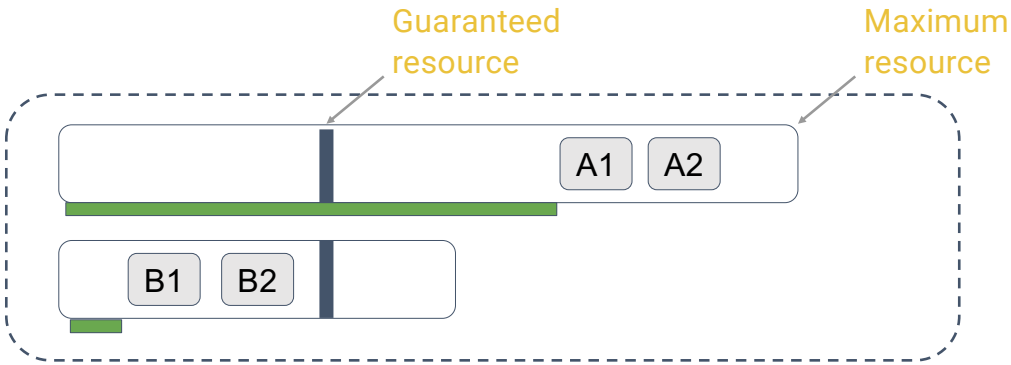


队列间公平性：优先分配给 Guaranteed 资源使用率最小的队列

Inter-queue fairness: allocate for queue with minimum usage ratio of guaranteed resource first

弹性资源管理：超分 + 抢占

Elastic resource management: over-allocation + preemption



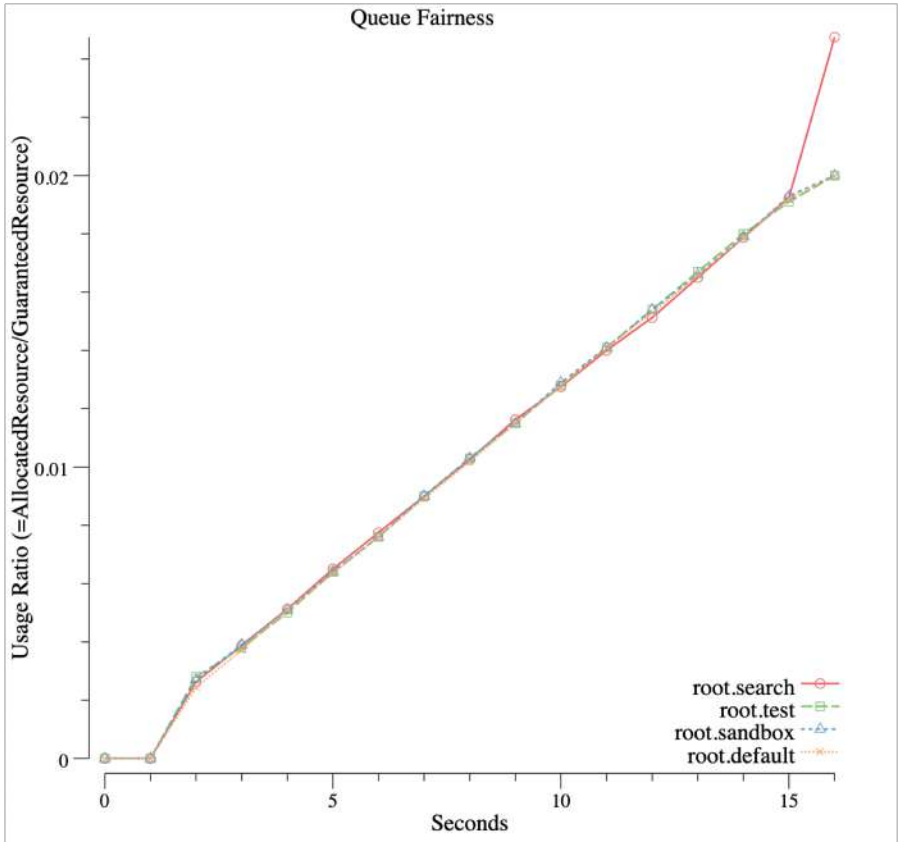
队列间资源公平性

Evaluation of Intel-queue Resource Fairness

Schedule workloads with different requests in 4 queues with different guaranteed resources.

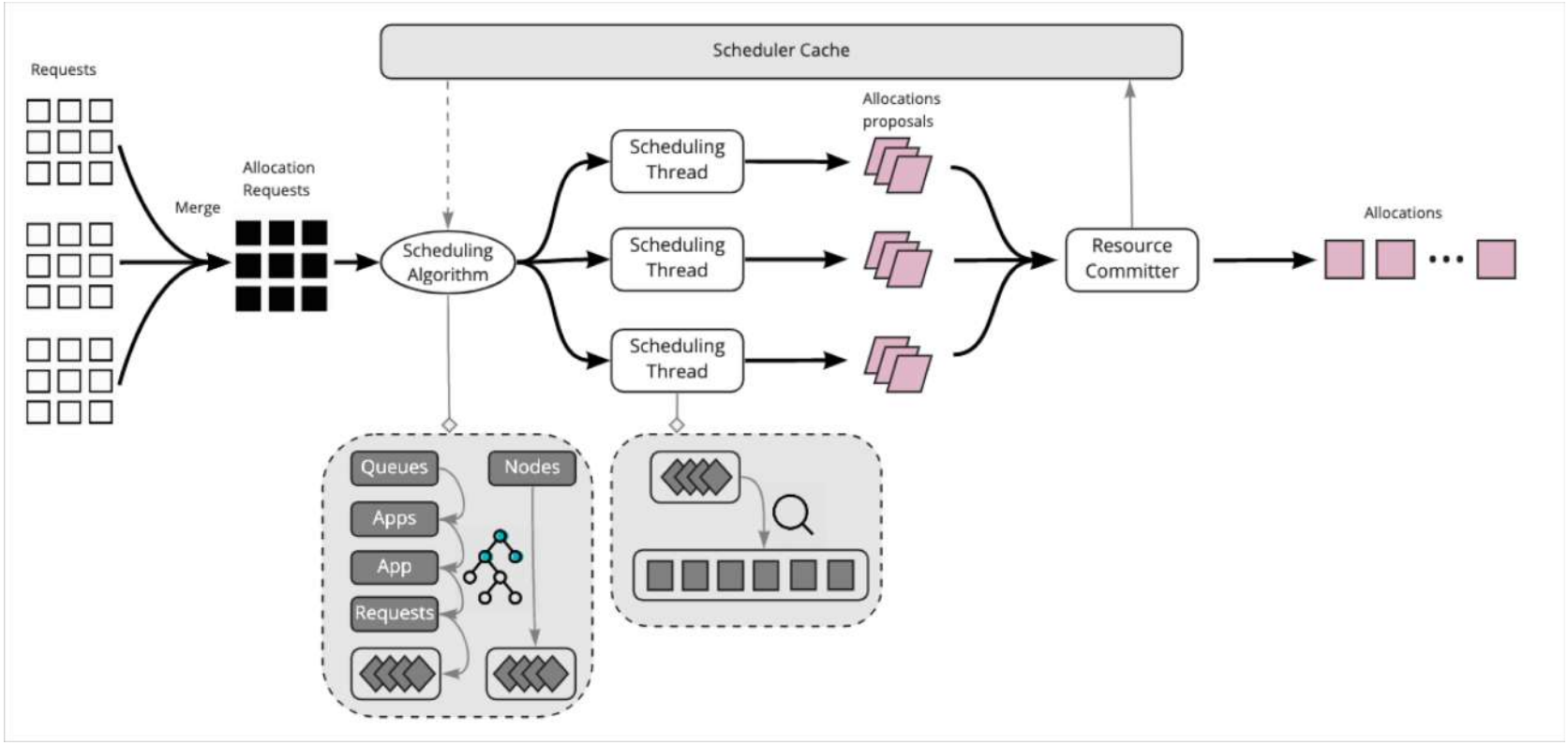
Queue	Guaranteed Resource (Mem)	Requests (NumOfPods * Mem)
root.default	500,000	1000 * 10
root.search	400,000	500 * 10
root.test	100,000	200 * 10
root.sandbox	100,000	200 * 50

Usage ratios of queues increased in the same proportion.



调度器性能优化

Optimize Scheduler Performance



调度器吞吐

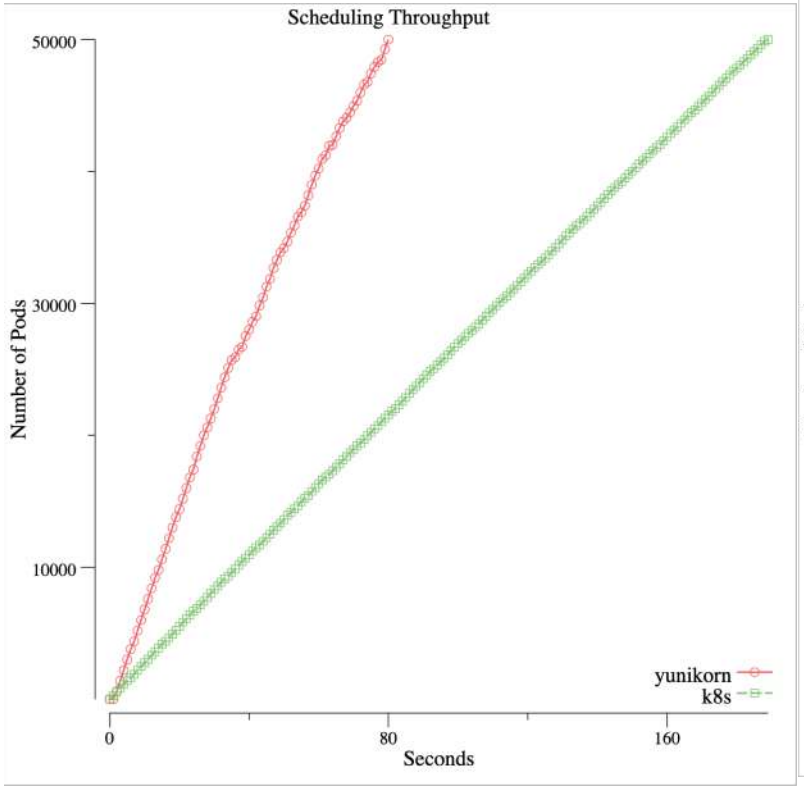
Scheduler Throughput

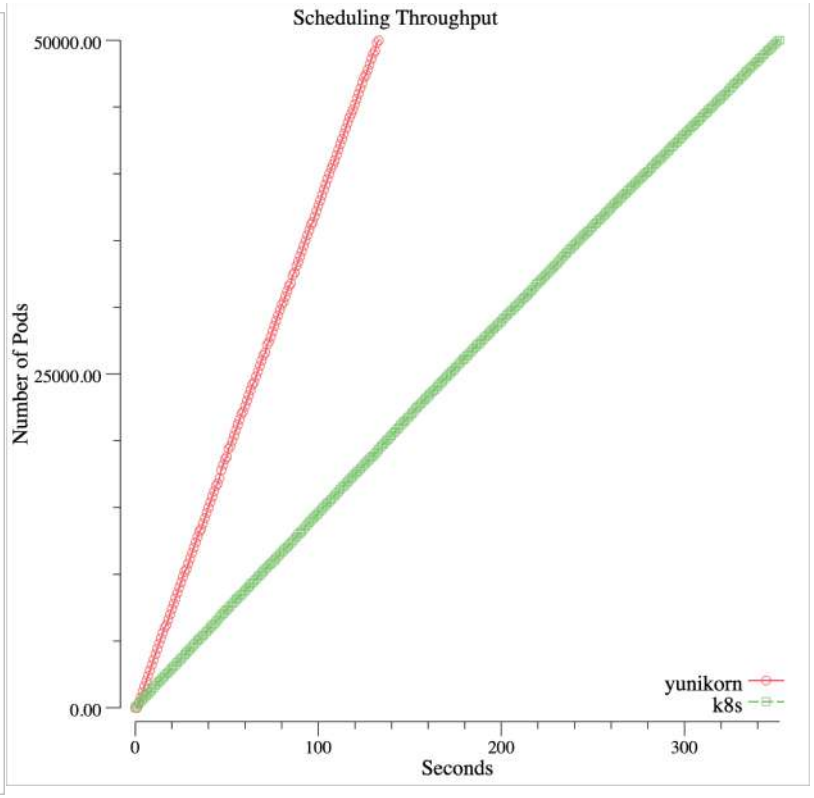
Schedule 50,000 pods on 2,000/4,000 nodes

Throughput (Pods per second)

617 vs 263 ↑ 134%

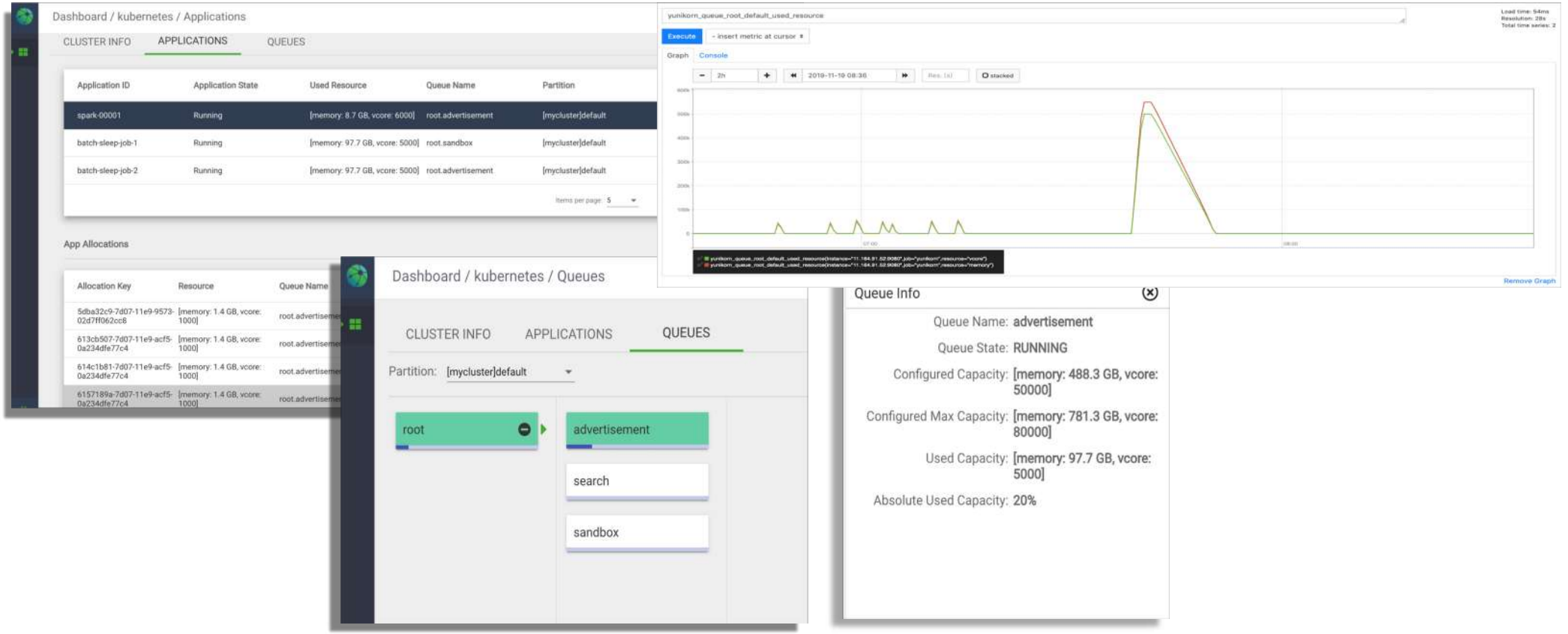
373 vs 141 ↑ 164%





作业管理及资源监控

Job management and resource monitoring



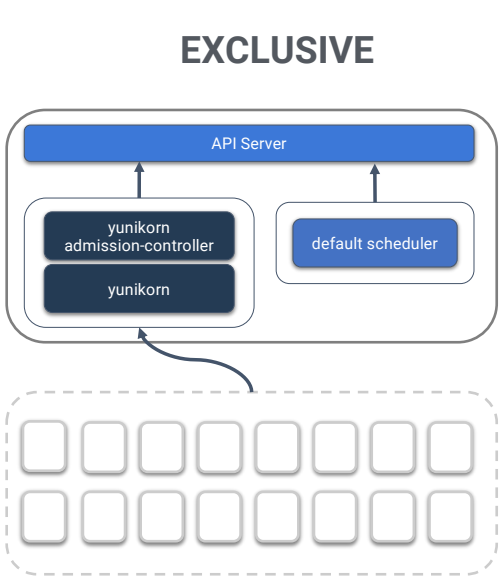
深入了解 YuniKorn

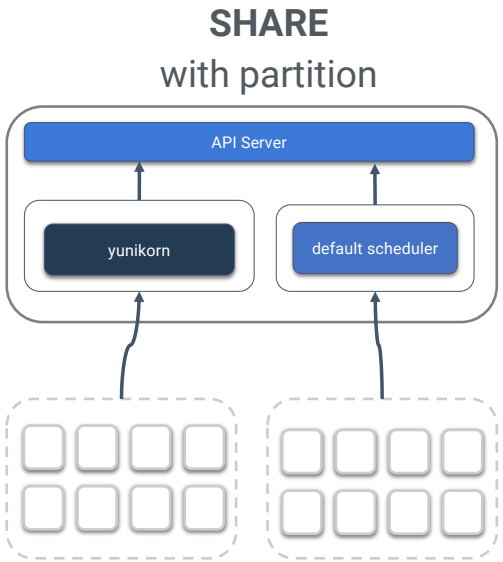
YuniKorn Deep Dive

03

安装与部署

Deployment and Configuration





Two different modes

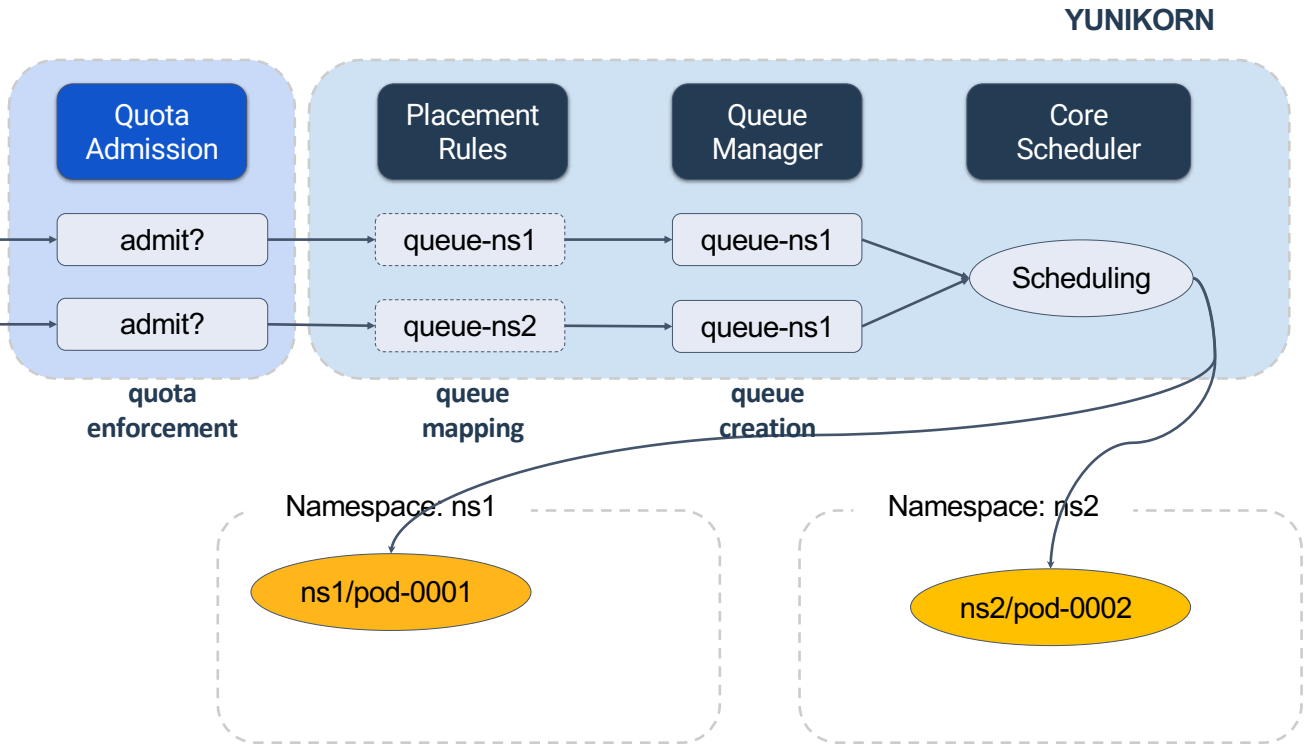
- **Exclusive** - Admission controller embedded
- **Share** - Without admission-controller

队列与命名空间

Work with queues and namespaces

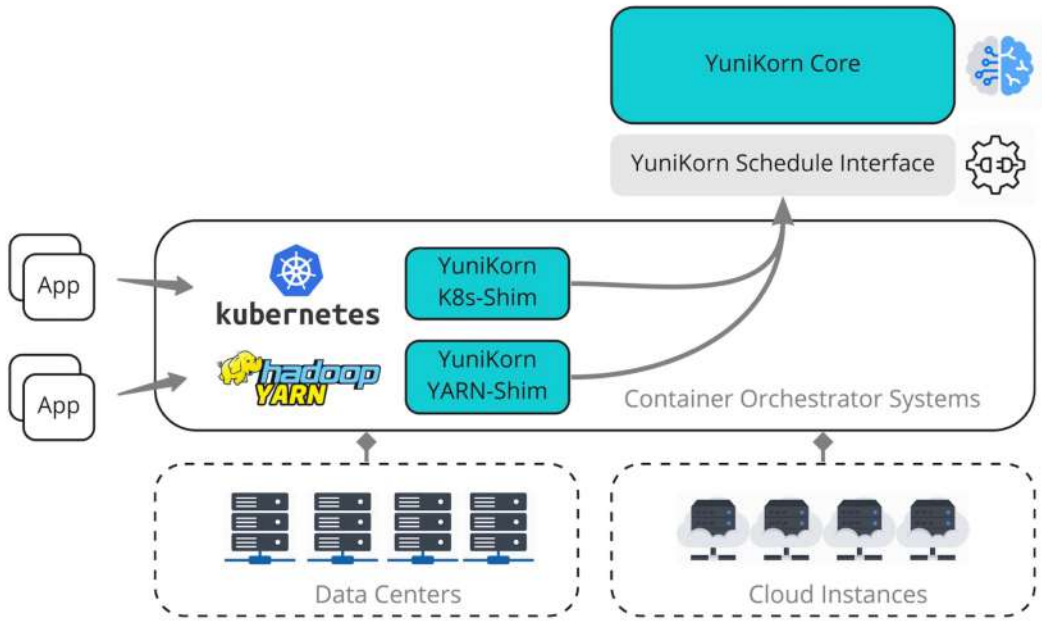
- Respect namespace resource quota
- Placement rule defines how to automatically map app to queues
- 1/1 mapping brings resource fairness to namespaces
- Pluggable placement policies

```
partitions:
  - name: default
    placementrules:
      - name: tag
        value: namespace
        create: true
    queues:
      - name: root
        submitacl: '*'
```



设计与架构

Design and Architecture

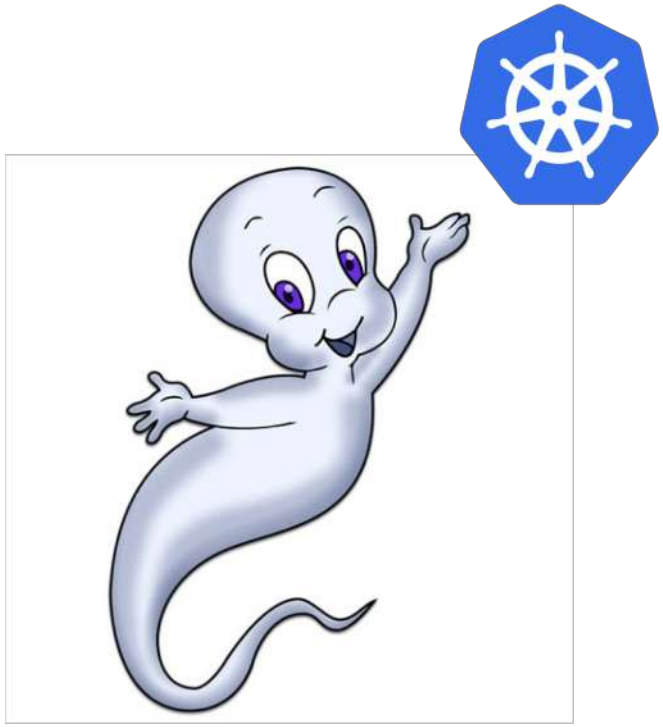


- YuniKorn is designed to serve common purpose, scheduler-core is **agnostic** about underneath platforms.
- **Abstract layer** for resource scheduling protocols, portable for any RM
- Written in Golang
- K8s-shim available, YARN shim WIP

YuniKorn Understands K8s Semantics

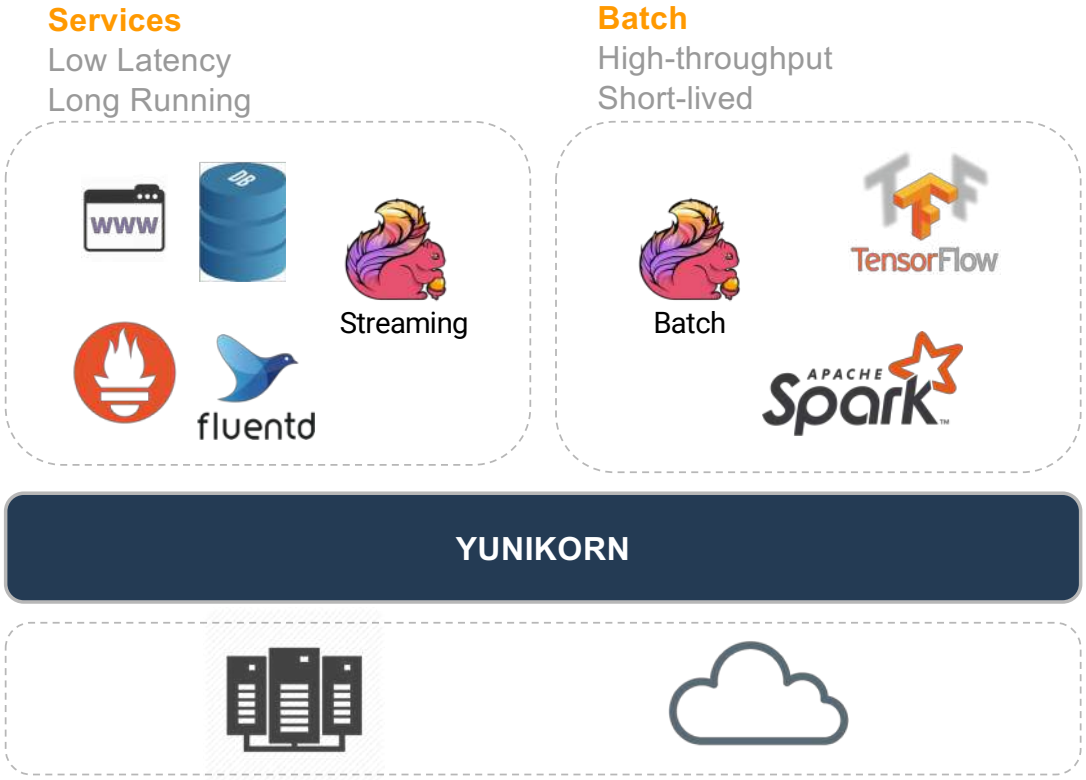
YuniKorn Understands K8s Native Semantics

- Support K8s **predicates**
  - Node selector
  - Affinity/Anti-affinity
  - Taints and toleration
  - …
- Support **PersistentVolumeClaim** and **PersistentVolume**
  - Volume bindings
  - Dynamical provisioning
- Publishes key **scheduling events** to K8s event system
- Support management commands
  - cordon nodes



YUNIKORN 适用于混部场景调度

YuniKorn is optimized for mixed-workloads scheduling



Guarantee Resource Fairness

Fairness and Priority [WIP] based preemption

Resource Quota Hard Limit

Predicates

PV/PVC

Taints & Tolerations

FLINK FORWARD

Open Source at July 17, 2019, Apache 2.0 Licence ❤️

Joining Apache Incubator 

Community effort by Cloudera, Alibaba and many more ...

Major contributors are experienced “scheduler” geeks

FLINK FORWARD

项目展望以及未来

Roadmap and Future

04

YuniKorn Roadmap

FLINK FORWARD

Current

- Hirechay queues
- Cross queue fairness
- Fair/Bin-packing scheduling policies
- Basic preemption
- placement rules & self queue management
- Metrics system and Prometheus integration

Upcoming

- Gang Scheduling
- Priority support (scheduling & preemption)
- Pluggable app discovery (support 3rd party app CRD/operators, e.g spark, flink operators)

未来展望

Future works.

- **支持更多的业务类型：搜索, 广告, 商业智能...**

  Support more usage scenarios: Search, Advertisement, BI...

- **支持Flink批任务运行**

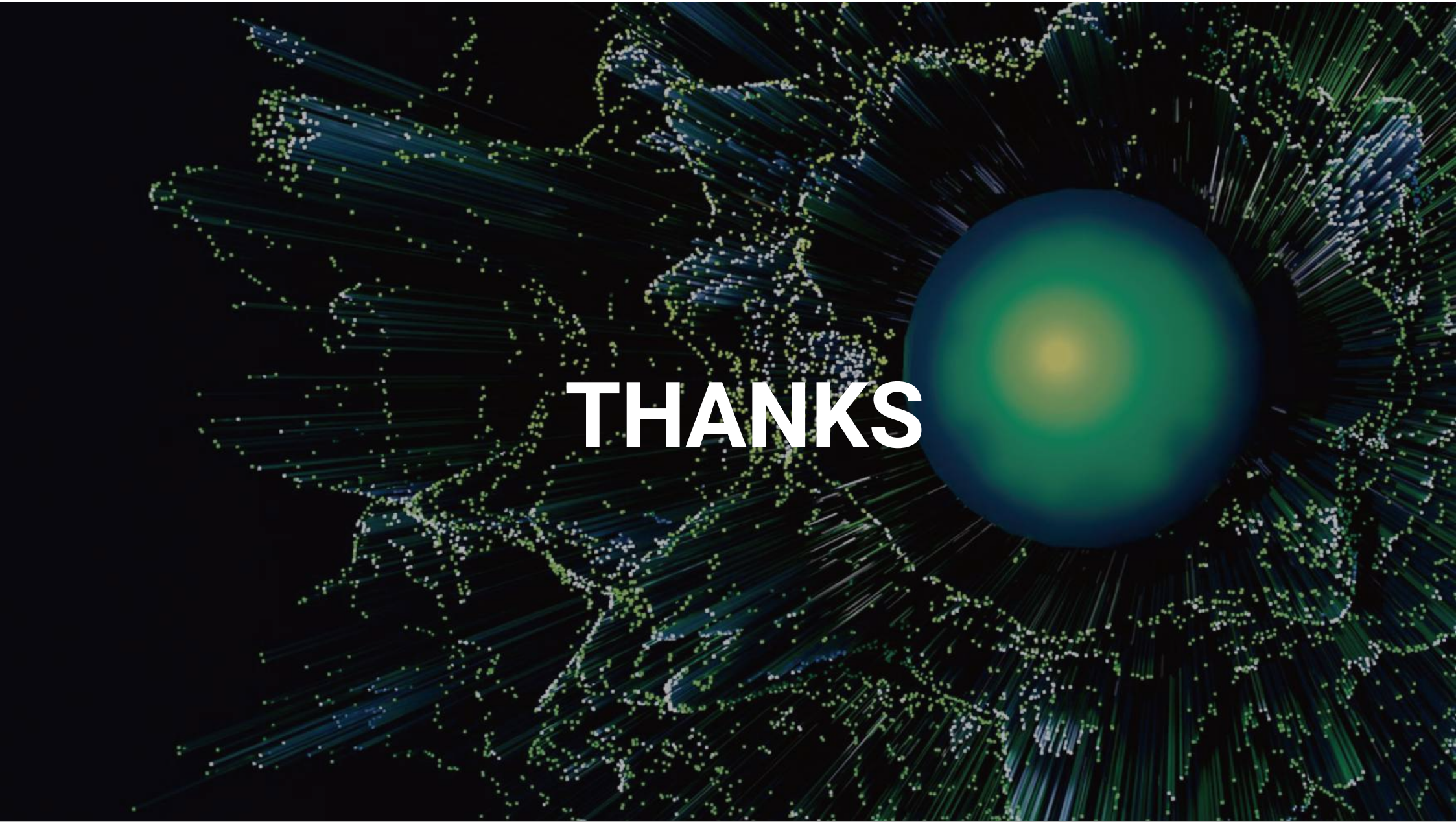
  Support flink batch workloads running on cloud native platform

- **与在线服务混部**

  Hybrid deployments with online services

- **支持无服务器架构**

  Support serverless architecture


THANKS