
SHANGHAI
KCD
GROWING CLOUD NATIVE TOGETHER
Kubernetes Community Days Shanghai 2024
Godel Scheduler:
字节开源内部超大规模在离线统一调度器
任玉泉 字节跳动

Content

目录

Part 01

研发背景

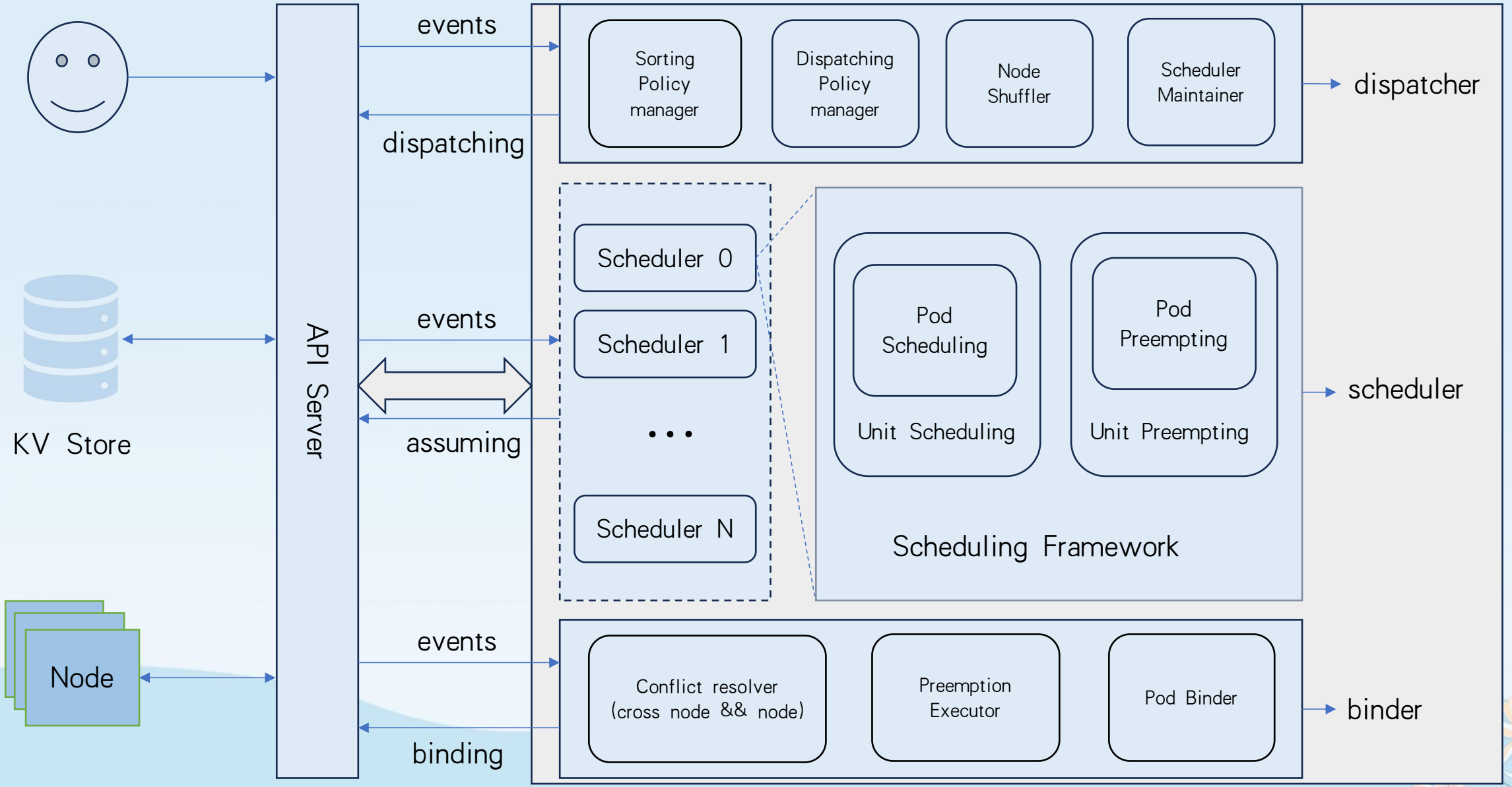
研发背景

- 业务需要
  - 成本：
    - 资源成本：在离线两套系统混部，利用率天花板低；
    - 开发成本：两套系统，重复工作多，适配成本高；
  - 效率：
    - 资源流转效率低；
    - 功能迭代效率低；
  - 运维：
    - 压力大；
- 开源产品不满足需求

Part 02

详细设计

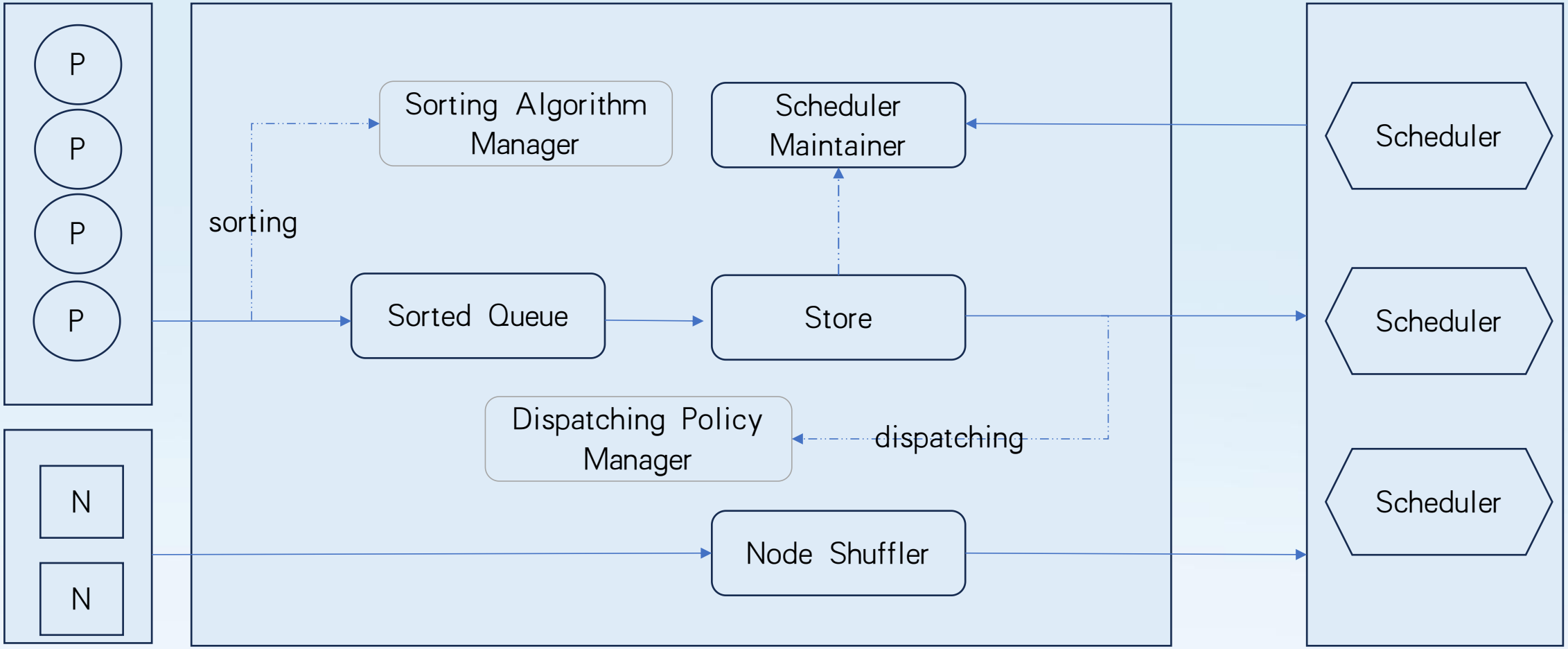
详细设计：整体架构


events
dispatching
Sorting Policy manager
Dispatching Policy manager
Node Shuffler
Scheduler Maintainer
dispatcher
API Server
KV Store
events
assuming
Scheduler 0
Scheduler 1
• • •
Scheduler N
Pod Scheduling
Unit Scheduling
Pod Preempting
Unit Preempting
Scheduling Framework
scheduler
Node
events
binding
Conflict resolver (cross node && node)
Preemption Executor
Pod Binder
binder

详细设计：整体架构

- 主要特点
  - 乐观并发： 把最耗时的操作抽出来，并发执行，提升吞吐天花板；
  - 两层框架：增强扩展性（“批”调度能力）的同时，进一步提升性能；
  - 统一调度：在线任务，离线任务，一个调度器；
  - 基于 k8s 系统：完全兼容 k8s 生态；

详细设计：dispatcher



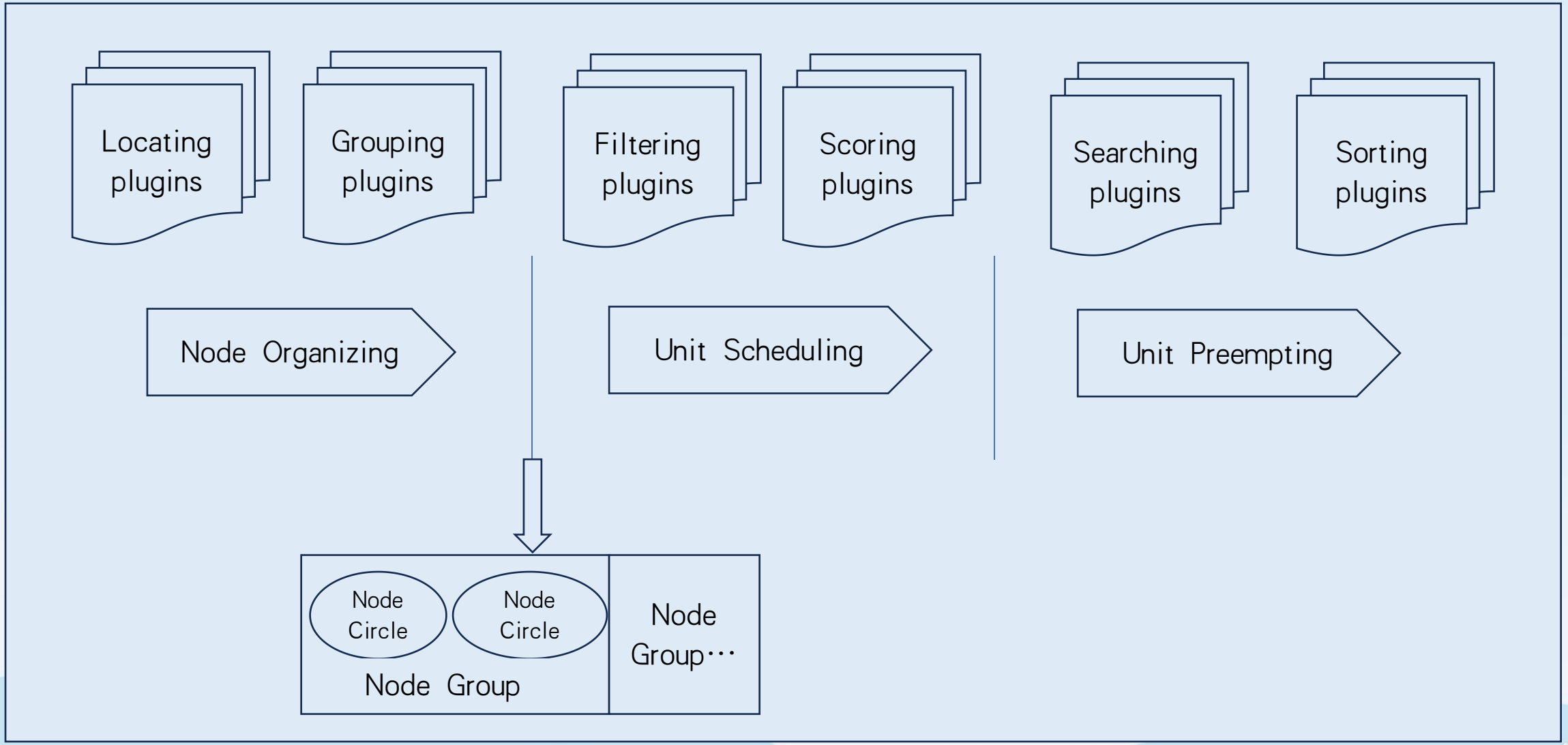
详细设计：dispatcher

Dispatcher 主要负责应用排队，应用分发，节点分区等工作。

它主要由几个部分构成： Sorting Policy Manager，Dispatching Policy Manager， Node Shuffler， Scheduler Maintainer 和 Reconciler。 其中：

- Sort Policy Manager： 主要负责对应用进行排队，现在实现了 FIFO， DRF/FairShare （没上线生产环境）排队策略，后面会添加更多排队策略，如： priority value based 等；
- Dispatching Policy Manager：主要负责分发应用到不同的 Scheduler 实例，现阶段是默认策略：LoadBalance，后面会增强该功能，做成插件化配置模式；
- Node Shuffler：主要负责基于 Scheduler 实例个数，对集群节点进行 Partition 分组，每个节点在一个 Partition 里面，每个 Scheduler 实例对应一个 Partition，Scheduler 调度的时候会优先选择自己 Partition 节点，没有合适的情况下，才会去找其他 Partition 的节点。如果 Node 增删或者 Scheduler 个数变化，会基于实际情况重新分配节点；Partition 规则现在是基于 Scheduler 个数平均分配，后面会增强，Partition 策略可配置；
- Scheduler Maintainer： 主要负责对 Scheduler 实例状态进行维护，包括 Scheduler 实例健康状况，负载情况，Partition 节点数等；

详细设计：scheduler



详细设计：scheduler

Scheduler 主要负责为应用做出具体的调度和抢占决策，但是不真正执行（执行者是 binder）

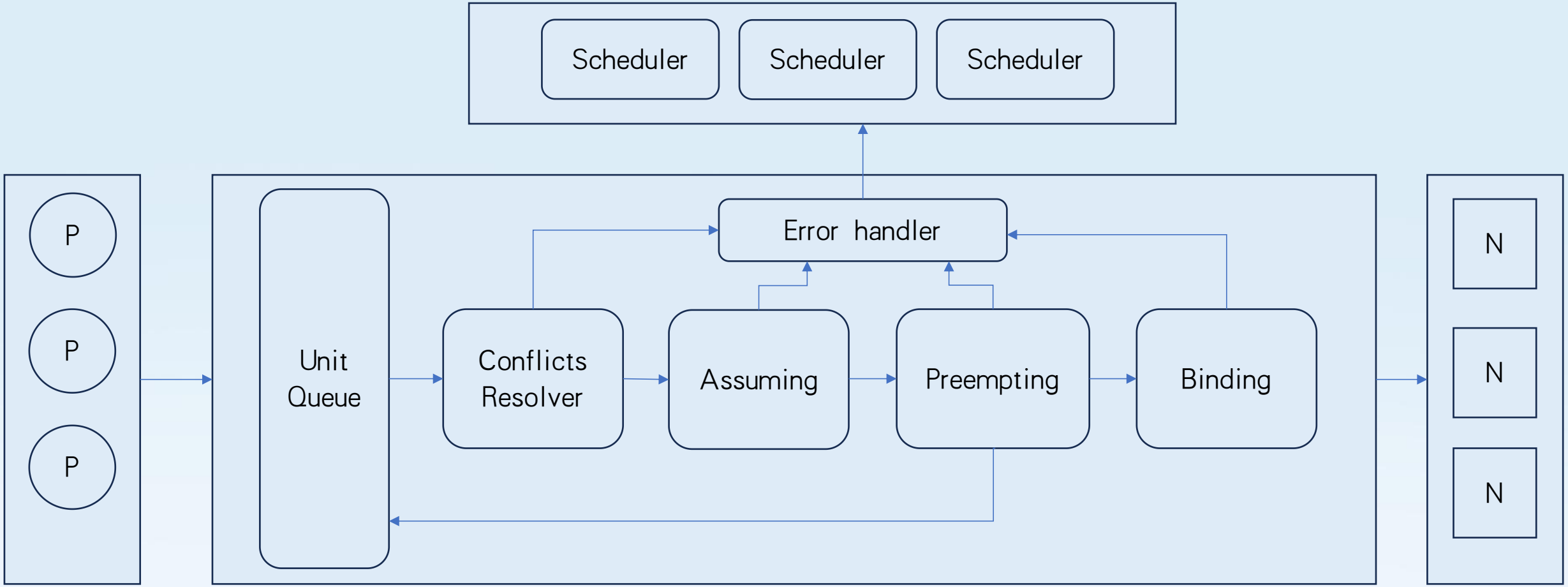
由两级框架组成： Unit scheduling framework 和 Pod scheduling framework。整个调度过程主要分为 3 大部分：Node Organizing， Unit Scheduling 和 Unit Preempting。

- Node Organizing： 过滤节点减少后续流程计算量，以及为节点进行排序。主要有两类插件：
  - Locating plugins： 基于应用，过滤掉不符合要求节点，比如： Local PV，DaemonSet Pods，Resource Reservation， Rescheduling 等，共同点是： 可以基于应用信息，过滤掉大部分节点，减少后面流程计算量，提升调度吞吐；
  - Node Grouping plugins： 为通过 Locating plugins 的节点进行分组，比如： 节点节点剩余资源量进行分组，或者基于 Job level affinity 里面的拓扑信息进行节点分组等，为的是能更快调度上或者获得更好的调度质量。

- Unit scheduling：基于应用请求，对通过 Node Organizing plugins 的节点进行匹配筛选和打分，类似 k8s Scheduler framework，Unit scheduling 阶段也有两类插件：
  - Filtering plugins：基于应用请求，过滤掉不符合要求的节点；
  - Scoring plugins：对上面筛选出来的节点进行打分，选出最合适的节点；

- Unit Preempting：上面阶段无法调度，则会进入抢占阶段，尝试为待调度应用去抢占正在运行的应用实例。该阶段也有两类插件：
  - Victim Searching： 遍历集群节点，尝试搜索 victims （被抢占应用），看是否能找到节点和 victims；
  - Candidates Sorting：如果上面步骤找到了合适的节点和 victims，则会为这些 victims 进行排序（节点粒度），选出最合适的节点和 victims；

详细设计：binder


Scheduler
Scheduler
Scheduler
P
P
P
Unit Queue
Error handler
Conflicts Resolver
Assuming
Preempting
Binding
N
N
N

详细设计：binder

Binder 主要负责乐观冲突检查，执行具体的抢占操作（删除 victims），进行应用绑定前的准备工作，比如动态创建存储卷等，以及最终执行绑定操作。

Binder 主要有 ConflictResolver, PreemptionExecutor 和 UnitBinder 三部分组成。

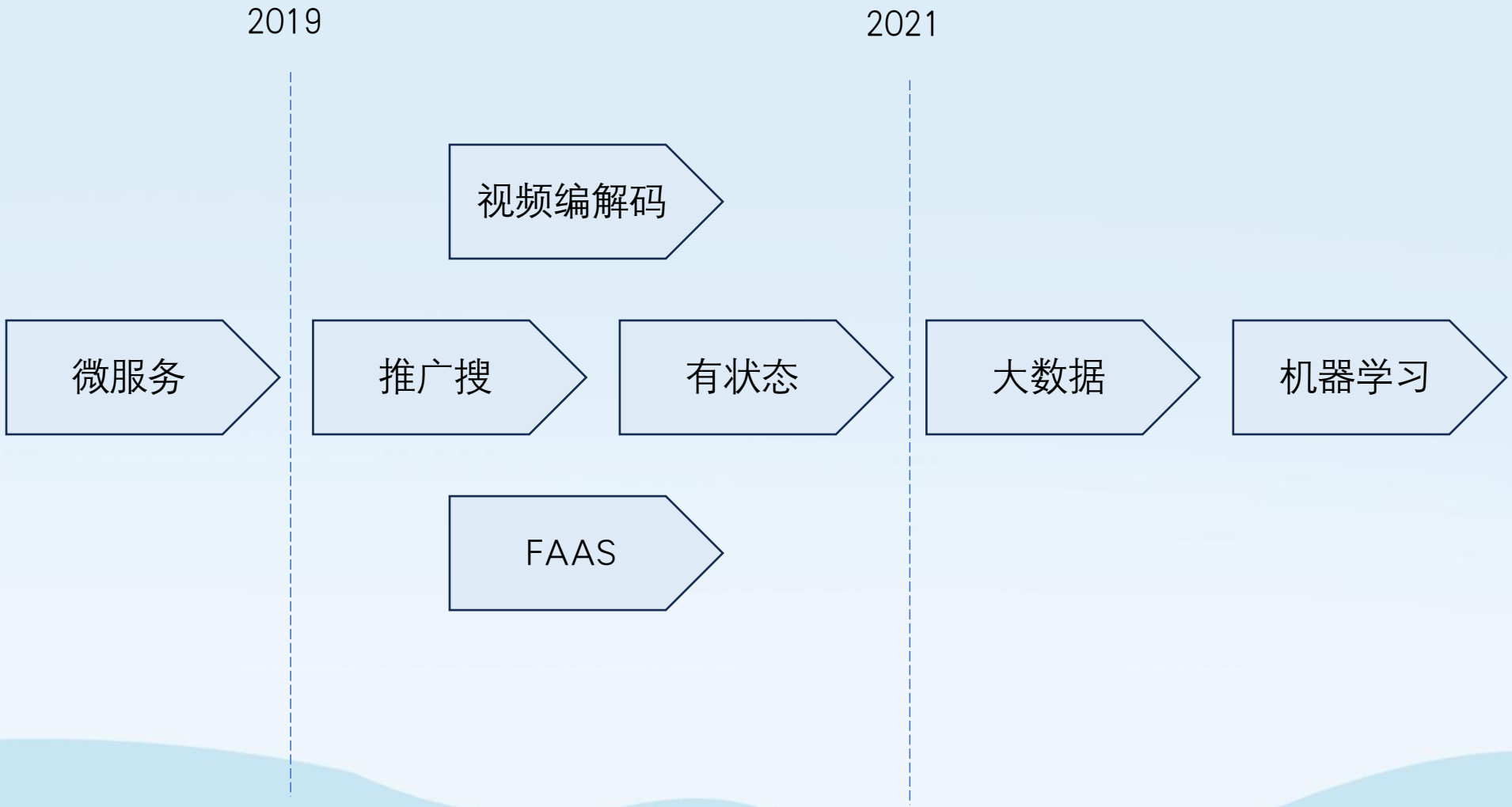
- ConflictResolver: 主要负责并发冲突检查，一旦发现冲突，立即打回，重新调度；Conflict resolver 有两大类： Cross node conflict resolver 以及 Single node conflict resolver
    - Cross node conflict resolver: 负责检查跨节点冲突，比如： 某个拓扑域调度限制是否仍然能满足等，由于该节点跨节点，Binder 必须串行执行；
    - Single node conflict resolver: 单节点内冲突检查，比如： 节点资源是否仍然足够等，该节点检查的逻辑限制在节点内部，所以不同节点的检查可以并发执行（unit 内 pods 调度到不同节点）；
- PreemptionExecutorperator: 如果没有冲突，同时应用需要抢占，则执行抢占操作，删除 victims，等待最终调度；
- UnitBinder: 主要负责绑定前准备工作，比如：创建 volume 等，以及执行真正的绑定操作。

现在的版本，Binder 里面还耦合了一个 PodGroup controller 实现，负责维护 PodGroup 的状态以及生命周期，后面会从 Binder 里面移除，独立成一个 Controller。

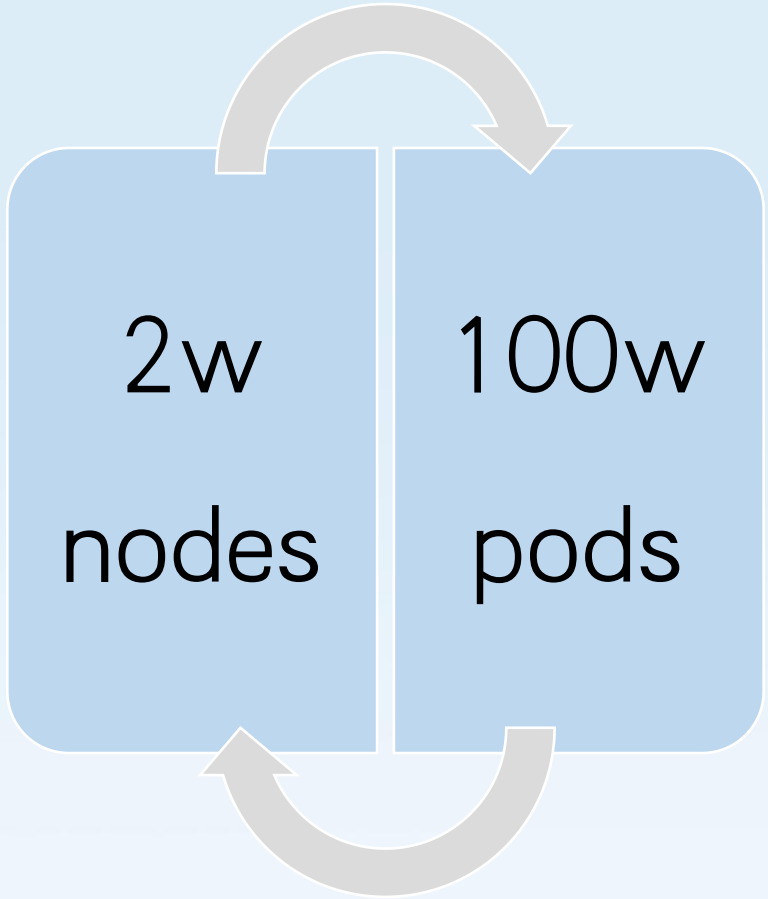
Part 03

内部实践

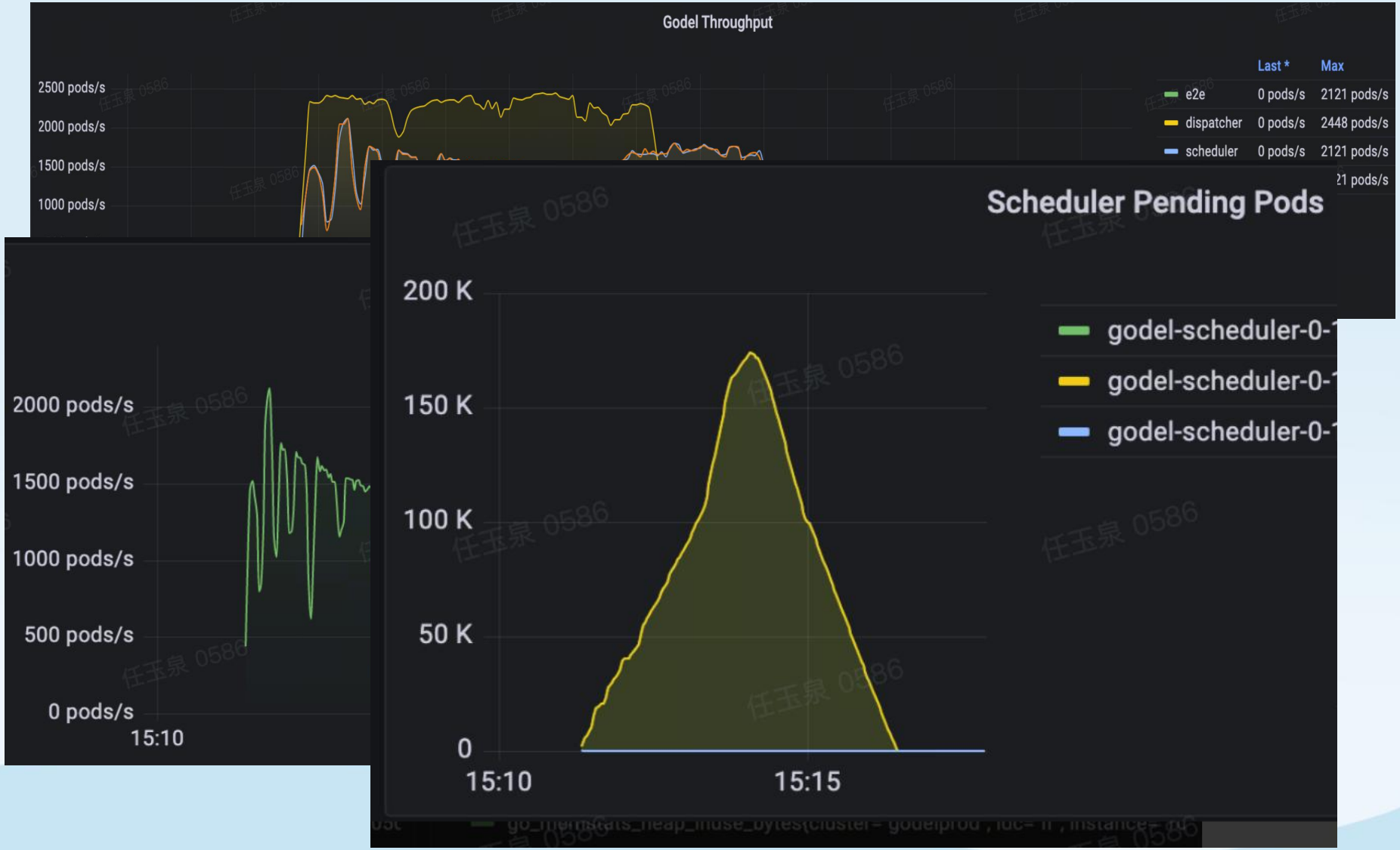
内部实践：支撑的业务



内部实践：单集群规模



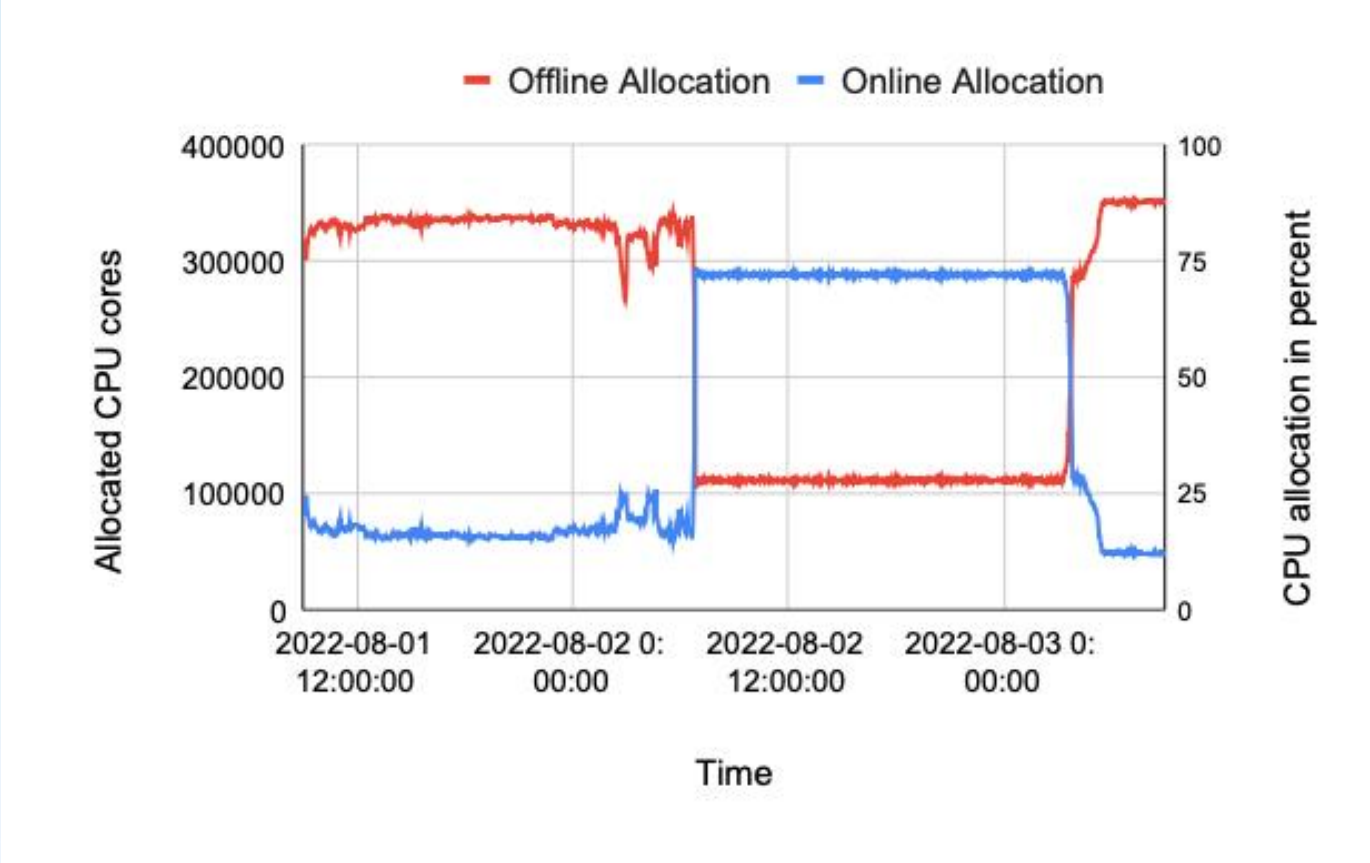
内部实践： 性能



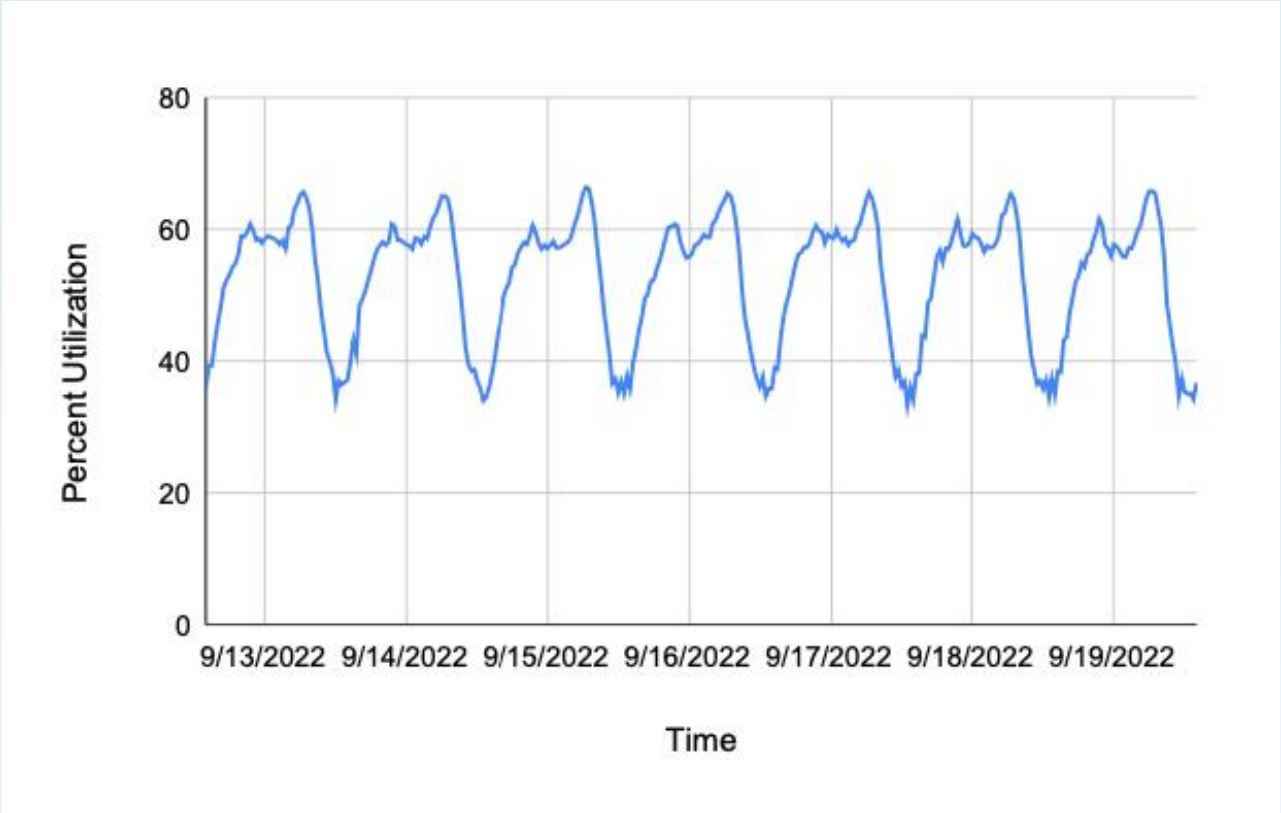
单 Scheduler 实例：
高峰调度吞吐： 2k+ pods/s

多 Scheduler 实例：
4k+ pods/s

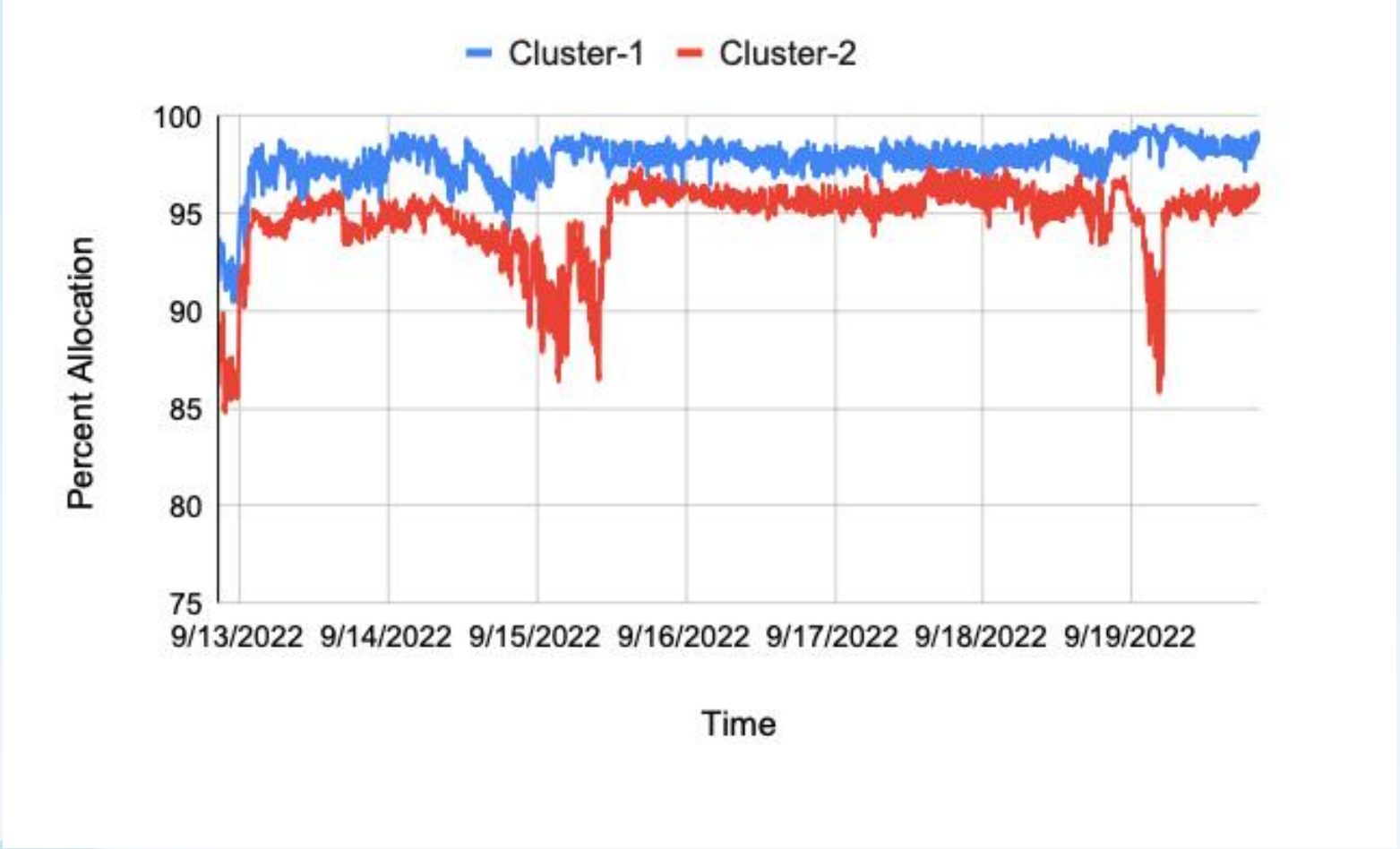
内部实践：在离线资源流转



内部实践：CPU 利用率


80
60
40
20
0
Percent Utilization
9/13/2022
9/14/2022
9/15/2022
9/16/2022
9/17/2022
9/18/2022
9/19/2022
Time

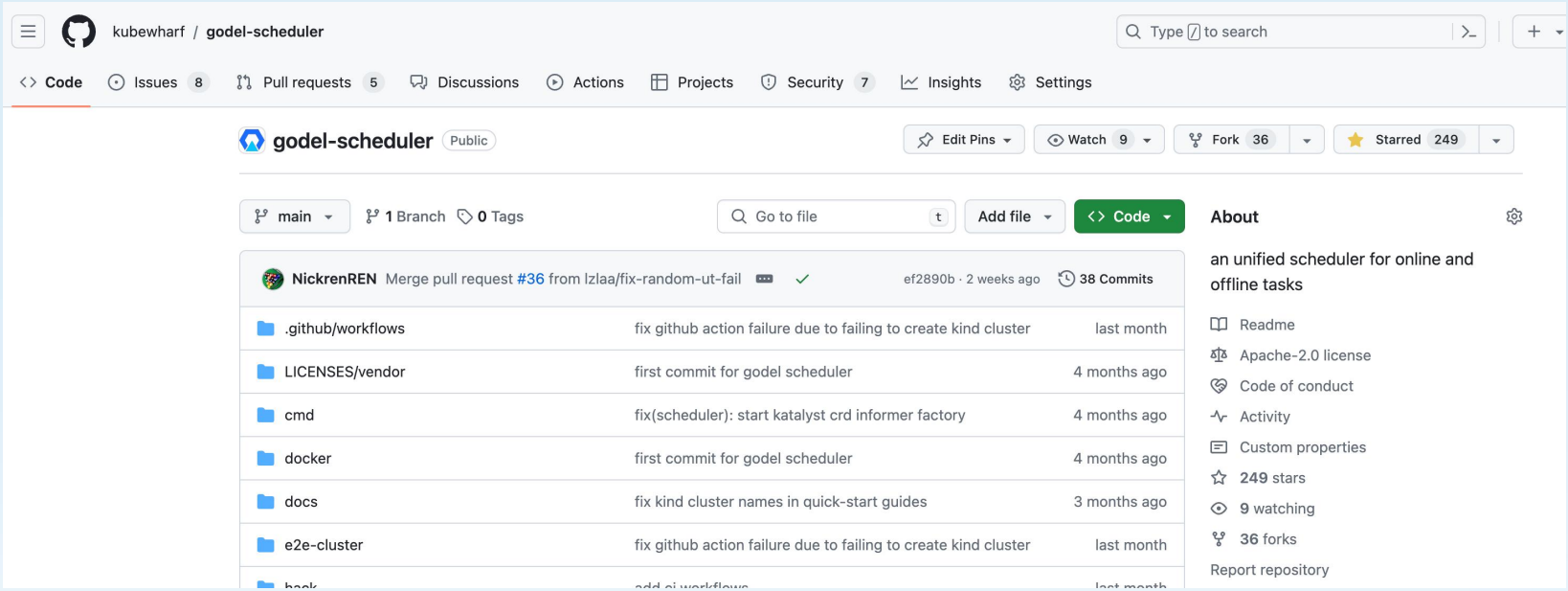
内部实践： GPU 分配率


Cluster-1
Cluster-2
100
95
90
85
80
75
Percent Allocation
9/13/2022
9/14/2022
9/15/2022
9/16/2022
9/17/2022
9/18/2022
9/19/2022
Time

Part 04

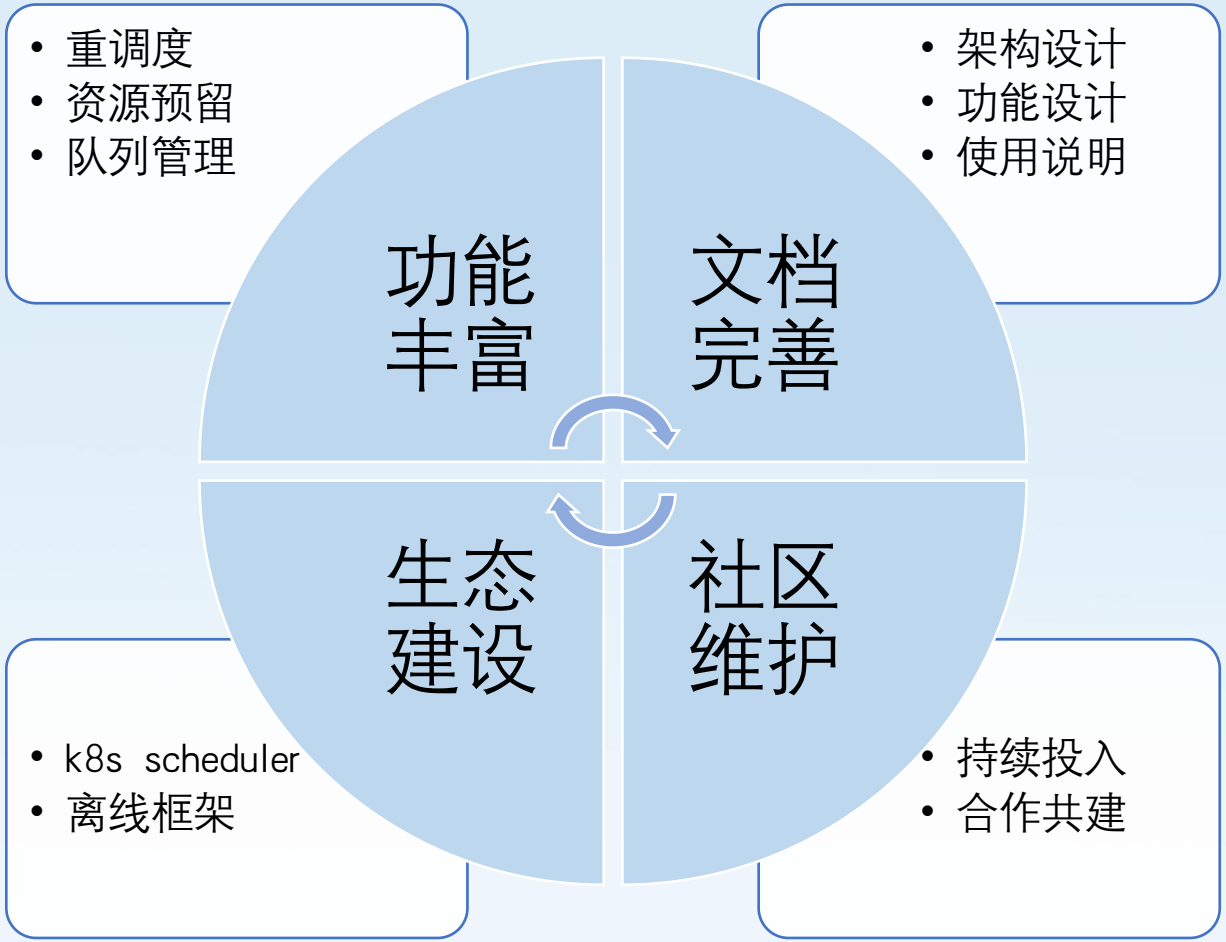
未来工作

未来工作: 已经开源



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

未来工作: 社区维护


• 重调度
• 资源预留
• 队列管理
• 架构设计
• 功能设计
• 使用说明
功能丰富
文档完善
生态建设
社区维护
• k8s scheduler
• 离线框架
• 持续投入
• 合作共建

未来工作:

时间	Releases（version）	新增重要功能	补充说明
2024 Q1	0.1	•实时数据接入 （load，usage ...）	•文档丰富 • 功能 Quickstart 文档 • 设计文档 •周边生态构建 • Flink • Spark • ML frameworks
2024 Q2	0.2	•资源预留	
2024 Q3	0.3	•重调度 •Unit 对象丰富 （除 PodGroup，支持 Deployment...）	
2024 Q4	1.0	•多队列资源管理 • Queue controller • DRF • FairShare • 结合 Queue 资源 抢占	

Thanks.