在离线混部在知乎的实践

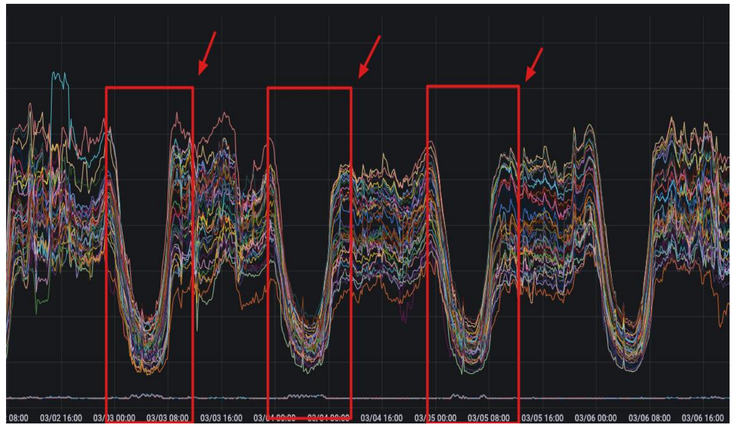
赵兵
知乎大数据基础架构工程师

CONTENT
目录 >>

Part 01 为何混部

1.1 资源使用现状

- 在线资源利用率低
- 离线资源紧张，无法保证核心任务资源
- 如何降本增效，在知乎备受关注
- 具有潮汐现象



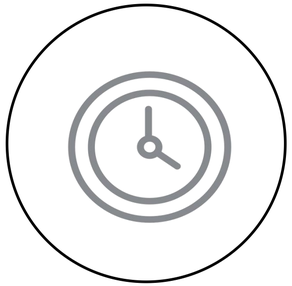
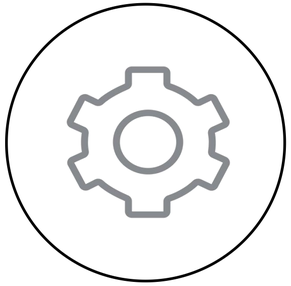
1.2 资源使用分析

用户对在线资源预估不准，资源申请量高于实际值，具有“占而不用”的现象

在线业务具有潮汐现象，按高峰值申请，低峰期资源浪费

夜间有大量的离线例行任务，任务资源申请不合理，资源紧张，产出不稳定

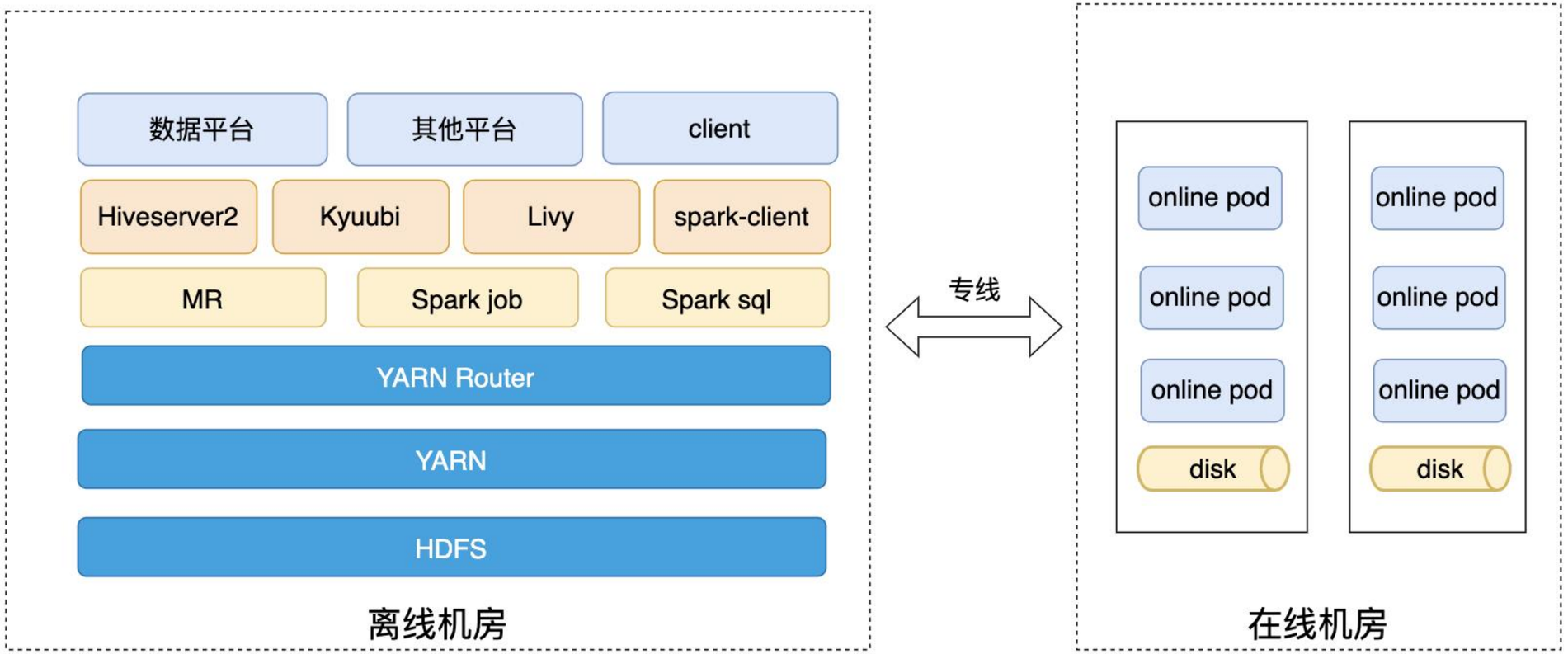
1.3 利用率提升

在离线混部	资源合理配置	容器化管理	降本增效
利用潮汐现象 在波谷时间段 运行离线任务	资源合理申请 自动扩缩容 运行更多的作业	集群化管理 合理规划资源 均衡调度	提升离线算力 提高产出稳定性 降低服务器成本 提升在线利用率

Part 02 如何混部

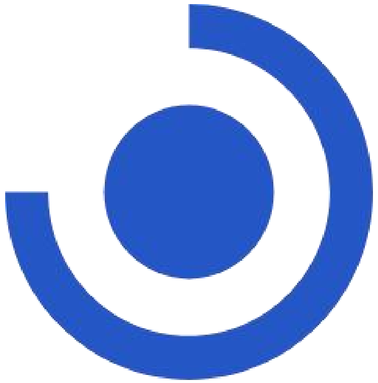
2.1 多云架构



问题

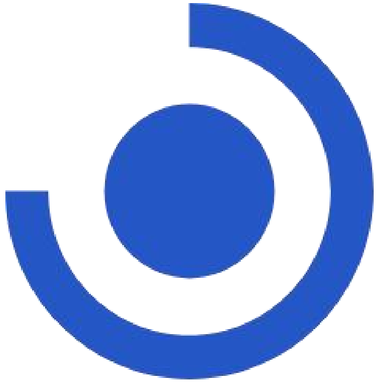
- 流量过大导致专线不可用
- 平台较多以及游离的客户机，如何实现调度
- 在线资源隔离

2.2 混部目标



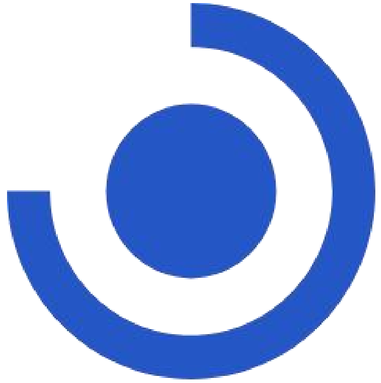
专线带宽控制

防止带宽打满，
影响在离线业务
合理的调度任务



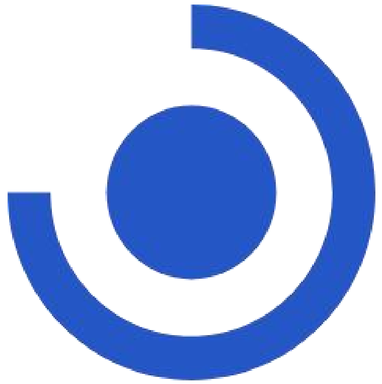
在线磁盘不足

容量小
无法满足shuffle



资源隔离

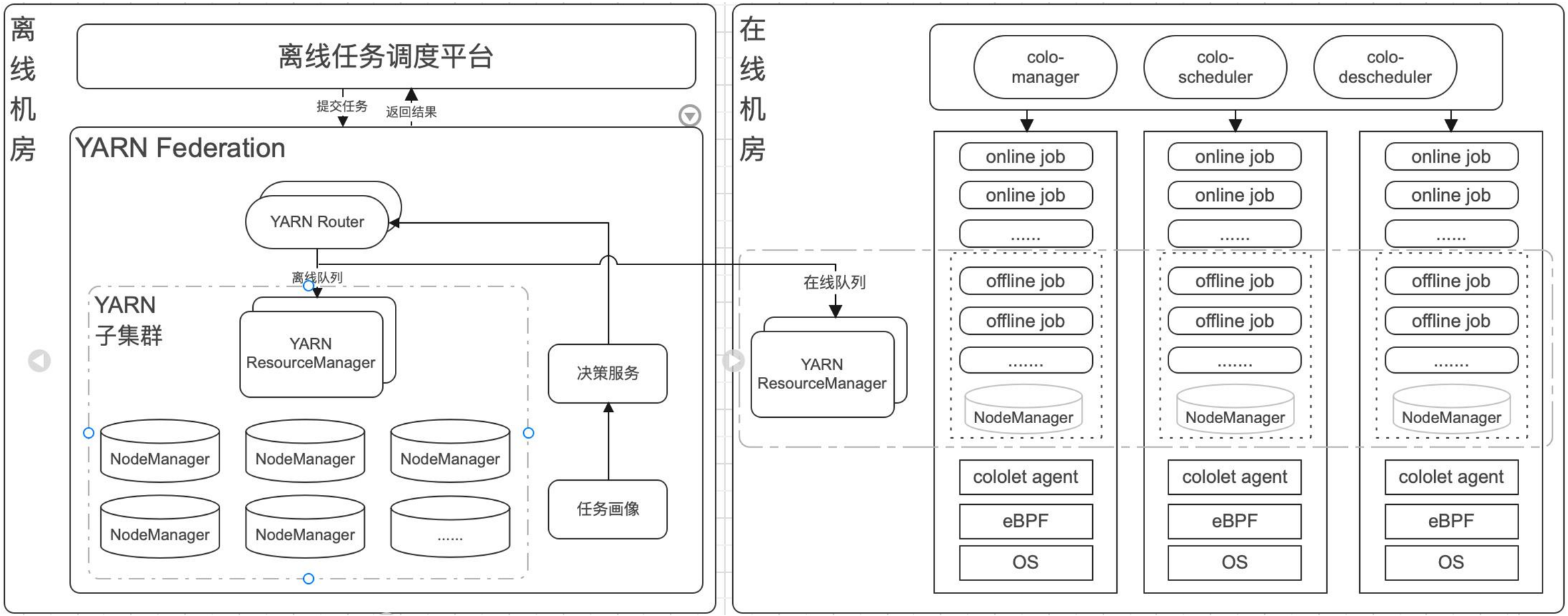
防止影响在线服务
cpu & mem



离线任务sla

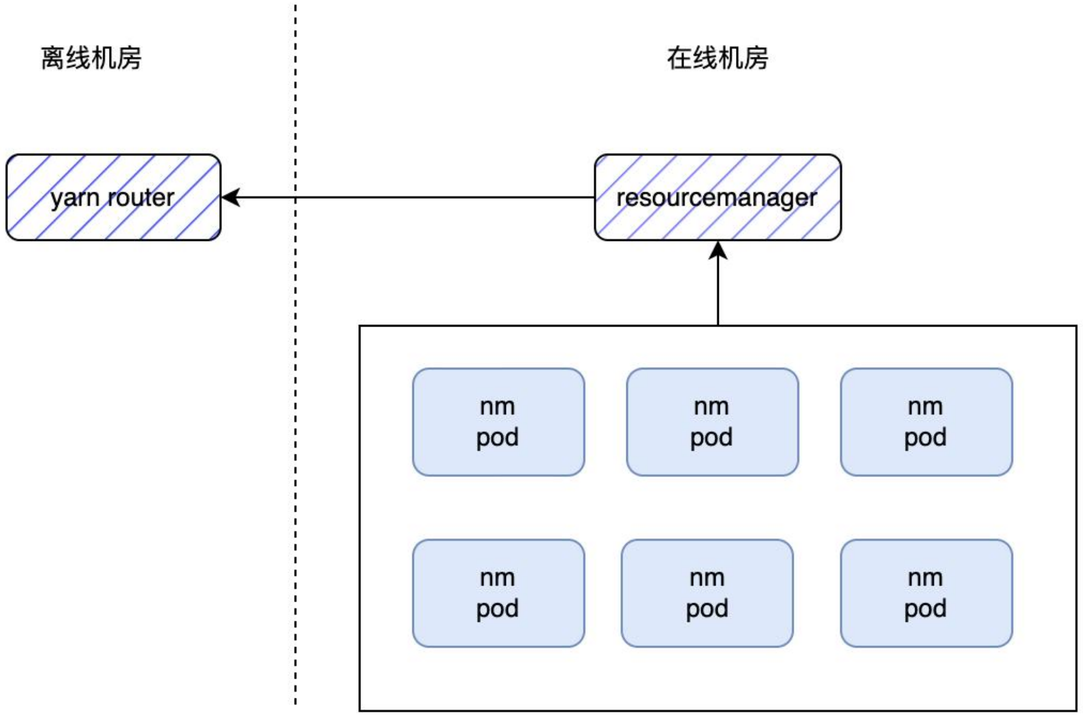
按优先级调度
保障任务稳定性
容错处理

2.3 架构设计


离线机房
离线任务调度平台
提交任务
返回结果
YARN Federation
YARN Router
离线队列
YARN
子集群
YARN
ResourceManager
NodeManager
NodeManager
NodeManager
NodeManager
NodeManager
.....
决策服务
任务画像
在线机房
colo-
manager
colo-
scheduler
colo-
descheduler
在线队列
YARN
ResourceManager
online job
online job
.....
offline job
offline job
.....
NodeManager
cololet agent
eBPF
OS

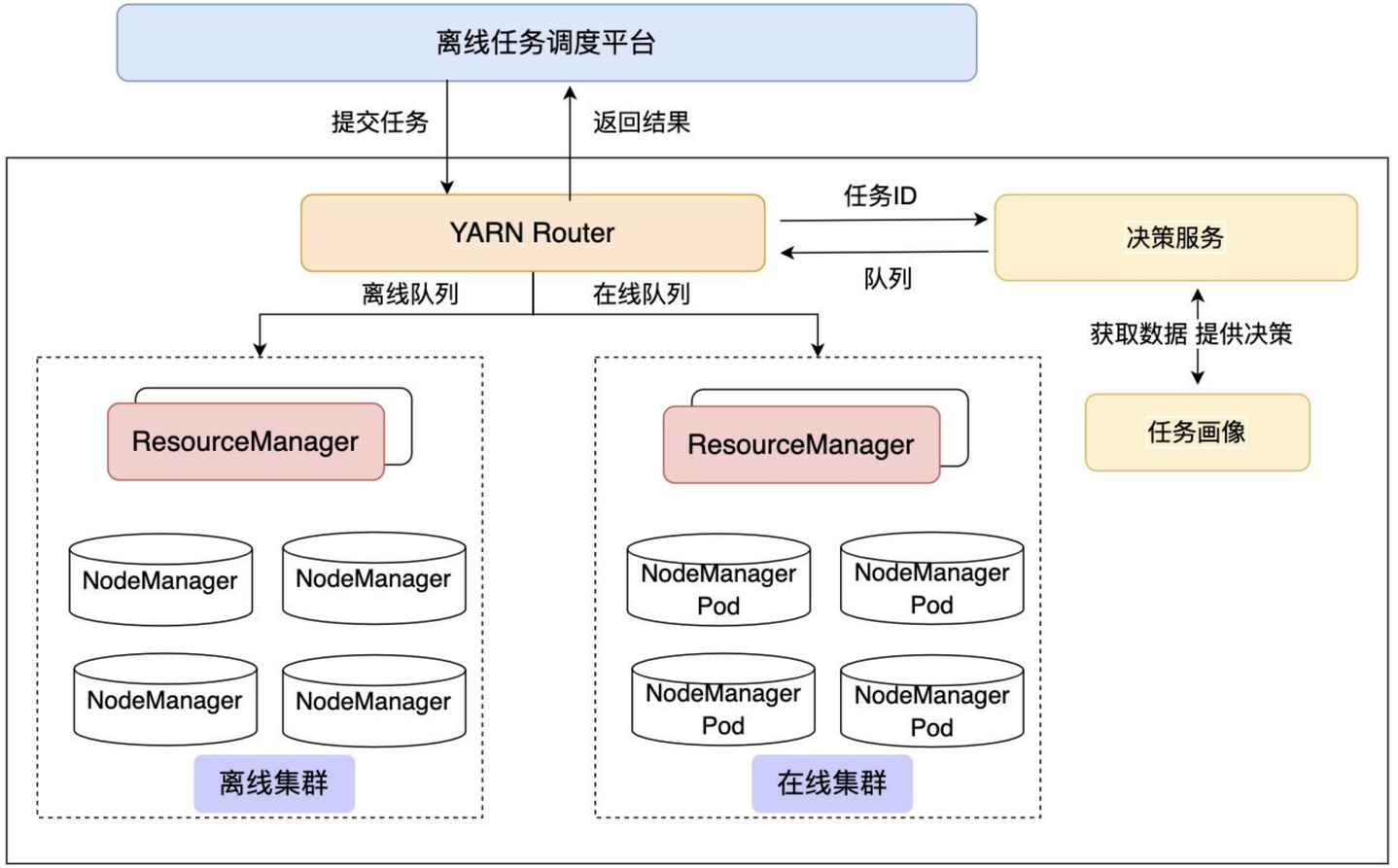
2.4 在线yarn部署

- 夜间 1 点对在线业务缩容，7 点扩容，物理机标记 label
- RM 物理机部署，稳定且可观测
- NM 容器化部署，DaemonSet 部署
- 主机网络 or 非主机网络
- hostname or ip

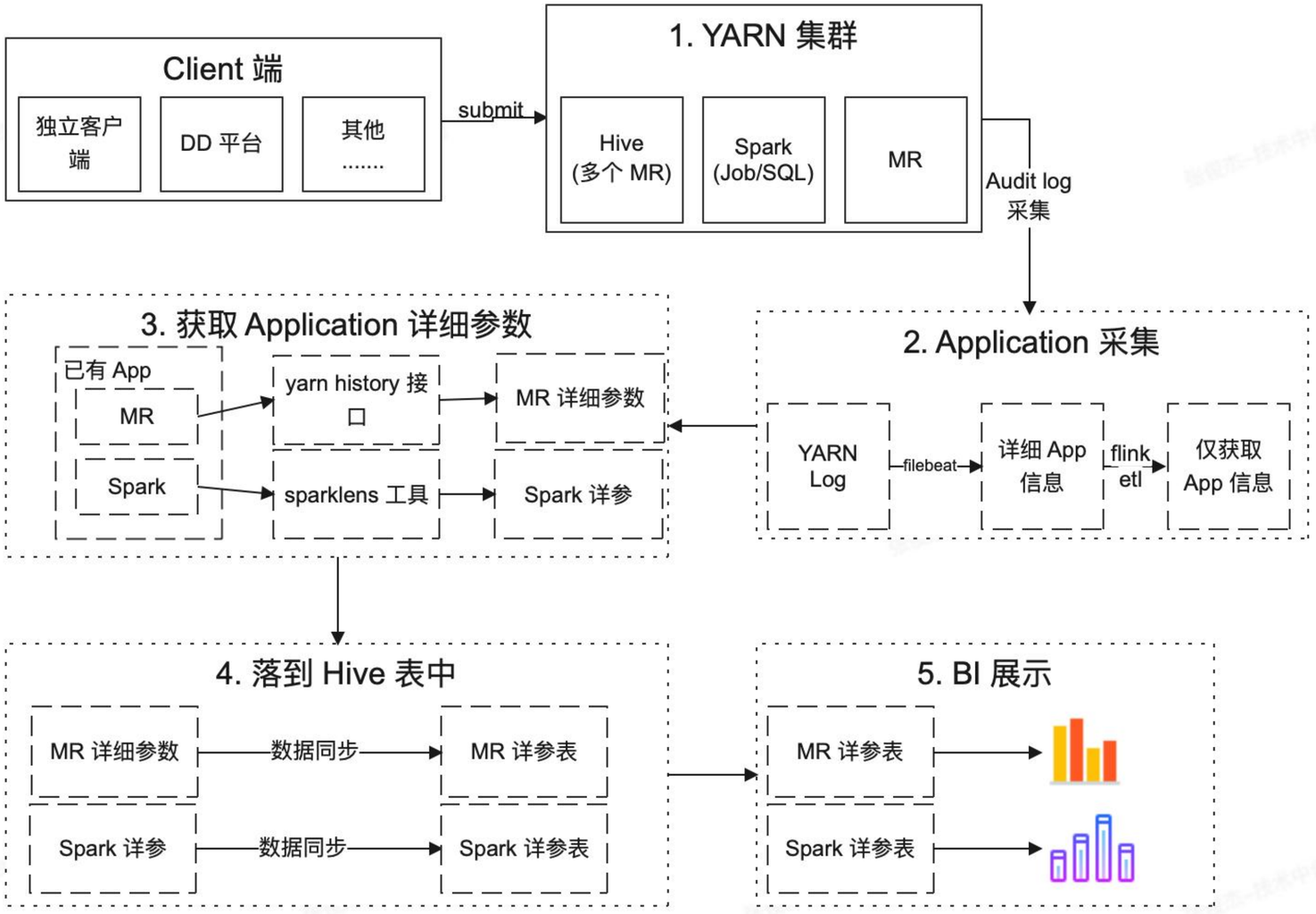


2.5 任务调度

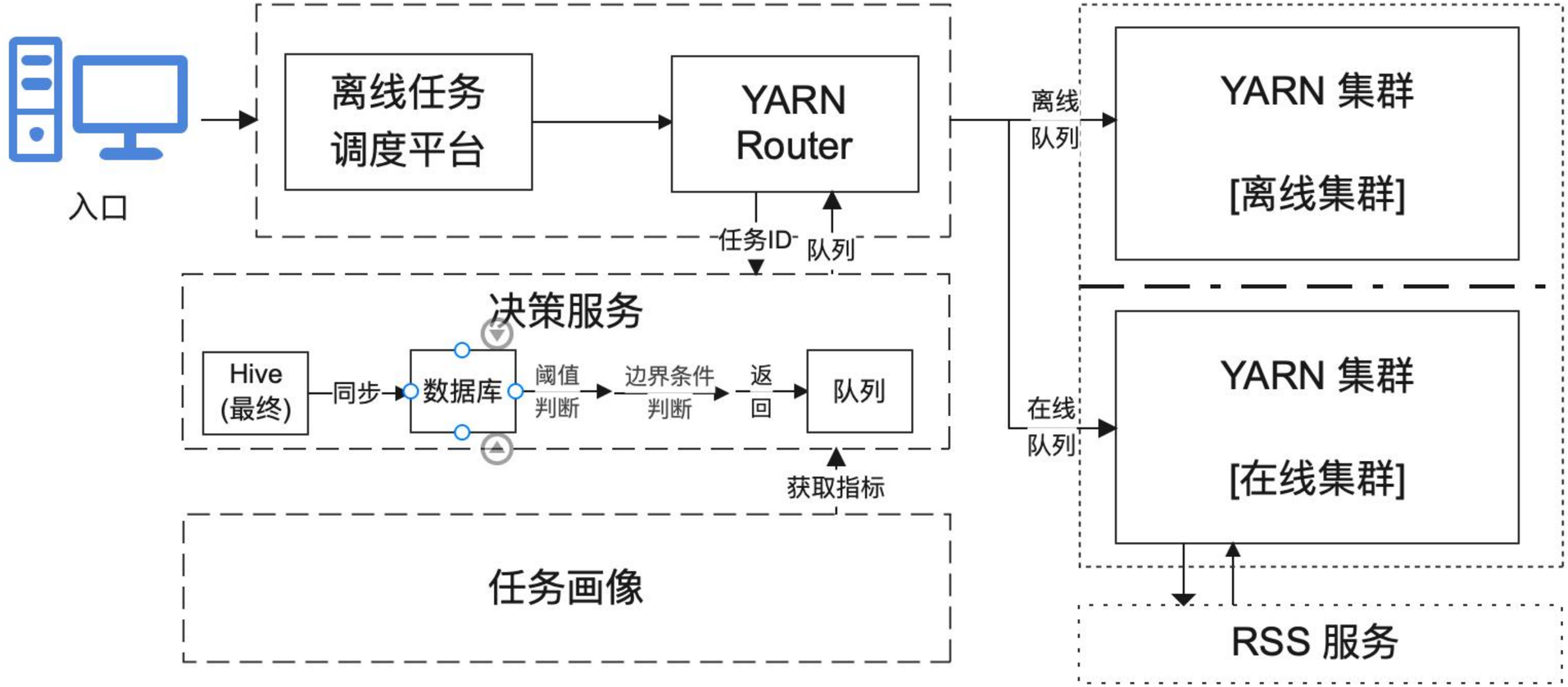
- 多个数据平台以及游离的客户机提交的任务均可以被决策
- 可控性高
- 业务无感知
- 改造成本低



2.6 任务画像


Client 端
独立客户端
DD 平台
其他
.....
submit
1. YARN 集群
Hive
(多个 MR)
Spark
(Job/SQL)
MR
Audit log
采集
2. Application 采集
YARN
Log
filebeat
详细 App
信息
flink
etl
仅获取
App 信息
3. 获取 Application 详细参数
已有 App
MR
Spark
yarn history 接口
sparklens 工具
MR 详细参数
Spark 详参
4. 落到 Hive 表中
MR 详细参数
数据同步
MR 详参表
Spark 详参
数据同步
Spark 详参表
5. BI 展示
MR 详参表
Spark 详参表

2.7.1 决策服务


入口
离线任务
调度平台
YARN
Router
任务ID
队列
决策服务
Hive
(最终)
同步
数据库
阈值
判断
边界条件
判断
返
回
队列
获取指标
任务画像
离线
队列
在线
队列
YARN 集群
[离线集群]
YARN 集群
[在线集群]
RSS 服务

2.7.2 任务稳定性

专线带宽

Map 的 input 数据大小
Reduce 的 output 数据大小
shuffle 数据大小

任务优先级

优先调度低优先级任务，
避免高优任务具有
被压制的风险

集群资源

在线 YARN 集群的可用资源
小于 100GB 时，
不在调度到在线集群

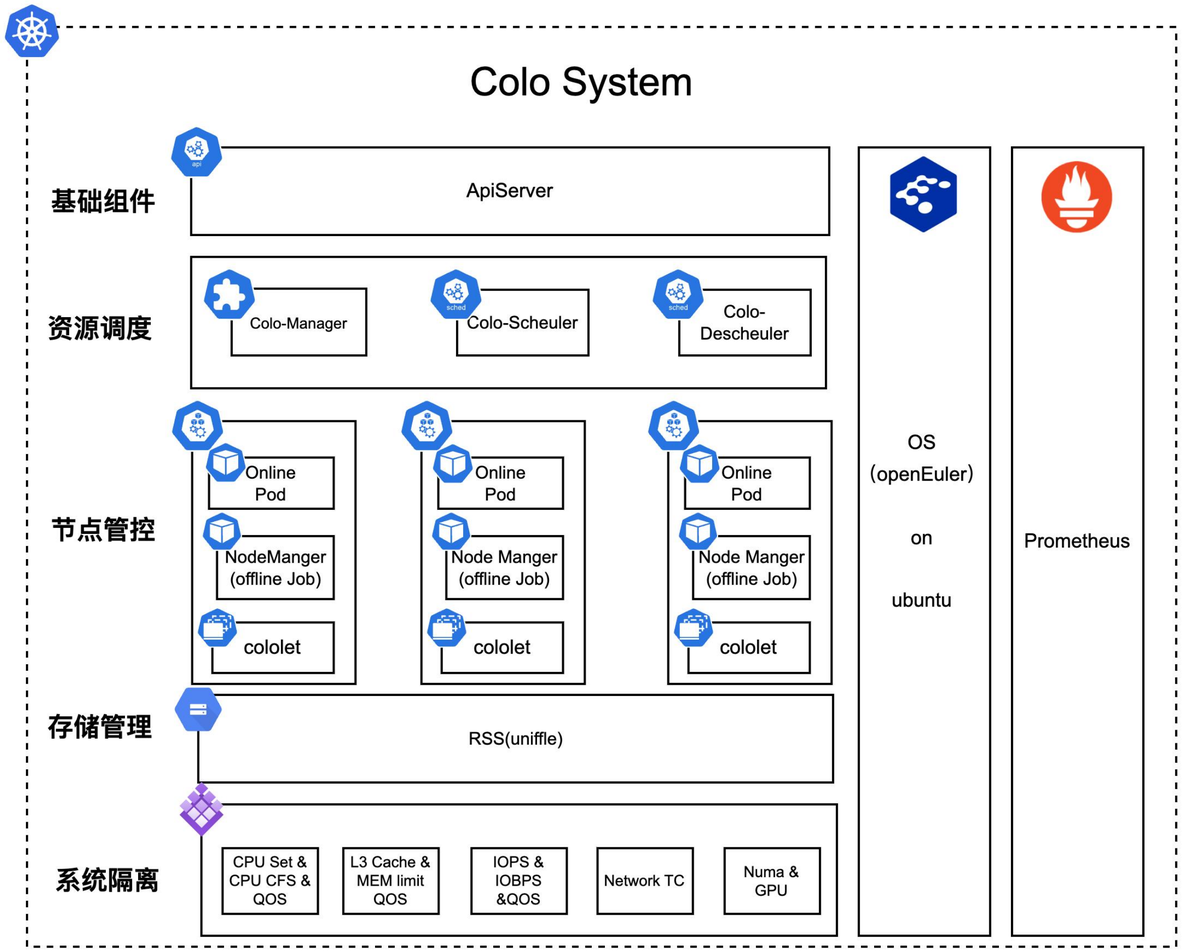
任务运行时常

当前时间加上任务的
耗时小于 7 点

2.7.3 容错处理

- 出现事件热点
- 在线环境出现异常时
- 近 n 天任务出现过异常

2.8.1 资源隔离



组件名称	功能
colo-scheduler	调度器，用于调度资源
cole-descheduler	重调度器，用于运行过程中平衡调度资源
cololet	用于采集指标，cpu/mem/io等；管理 Cgroup
colo-manager	同步nodemetrics，以及cololet资源配置等

2.8.2 cgroup隔离

IO 隔离

- device_read_bps
- device_read_iops
- device_write_bps
- device_write_iops

CPU 压制

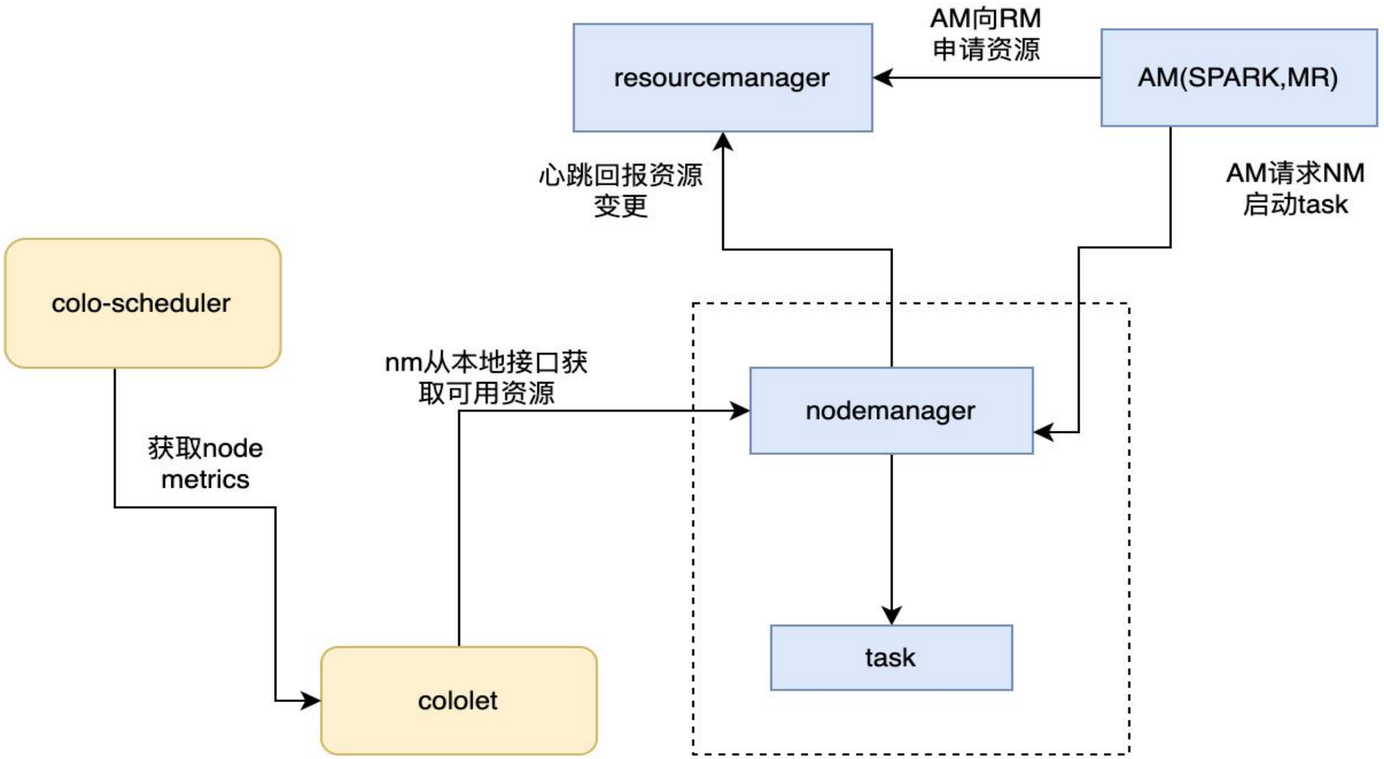
- cpu.qos_level：开启 CPU 优先级配置
- cpu.cfs_period_us
- cpu.cfs_quota_us

内存回收

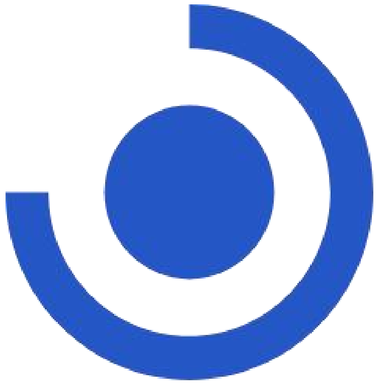
- memory.soft_limit_in_bytes
- memory.force_empty
- memory.limit_in_bytes
- /proc/sys/vm/drop_caches

2.9 资源管控

离线 Pod 资源不固定
如何不被压制
如何应对临界时间点

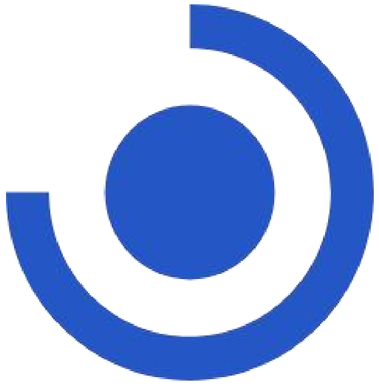


2.10 RSS 需求



支持现有的计算框架
MR & Spark

良好的性能



不依赖其他组件
对现有版本支持好

2.10.1 IO 隔离- Apache Uniffle

- 支持 MR/Spark/Tez
- 无依赖的组件
- 顺序写，顺序读
- 社区活跃度高

2.10.2 Apache Uniffle测试结果

变量	值
版本	社区版本 0.8
pod总内存	40GB
存储类型	MEMORY_LOCAL FILE
Xmx	36GB
副本数量	1
垃圾回收器	G1
JDK 版本	jdk8

shuffle数据大小	shuffle server节点	executor个数	ess耗时	rss耗时
400G	3	3	5803s	2301s
27G	3	3	414s	332s
7G	3	3	98s	116s

当shuffle数据达到一定量时，有明显的性能提升

2.10.3 uniffle线上效果

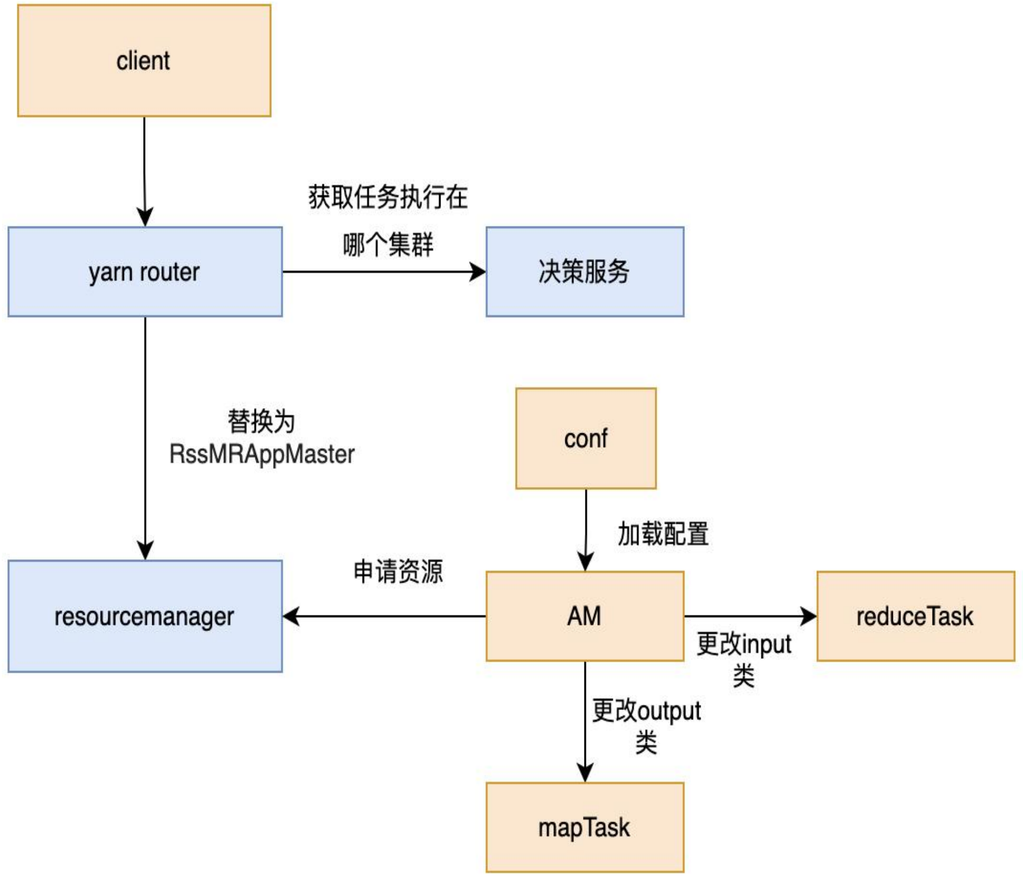
变量	值
容器镜像	社区版本 0.8
容器总内存	40GB
Xmx	36GB
存储类型	MEMORY_LOCALFILE
垃圾回收器	G1
客户端JDK 版本	JDK8
服务端JDK版本	JDK17
副本数量	3

- 10 台物理机,40 个 shuffle server 节点
- 峰值可处理 TB 级别的数据，均值在几百 G
- 被调度的任务平均耗时降低 8% 左右

2.10.4 线上优化

- MR task运行完，Container 不退出，资源不释放
  https://github.com/apache/incubator-uniffle/pull/882

- 客户端无法确定运行在哪个集群
  离线集群不支持 RSS，在线集群必须用 RSS
  在服务端控制是否使用 Uniffle



Part 03 混部收益

3.1 成本收益

每天调度近 6K+ 任务到在线
使用百 TB 内存和几万 core

调度到在线的任务
耗时降低 8% 左右

缓解离线资源紧张的压力
保障了任务产出稳定性
降低了几百台服务的成本

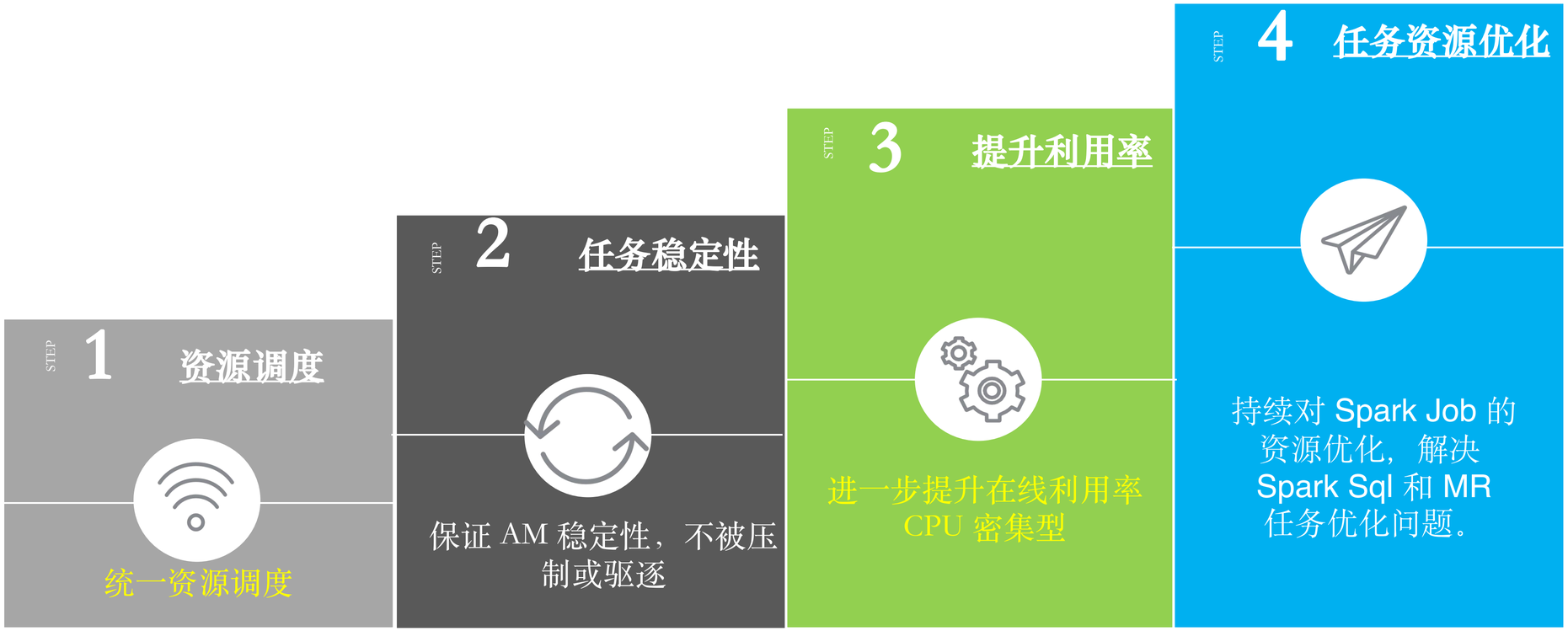
提升了在线服务器利用率
全天平均利用率提升 6%

3.2 技术收益

- 建设了完整的分时在离线混部能力及全时离在线混部能力的初步建设
- 在线调度器完成了基于实际负载的调度、均衡调度能力的增强，资源隔离等
- 离线建设了任务画像及调度决策能力，后续也可以进一步用于任务调度、资源分配的优化。
- RSS 服务建设与接入，完成大数据任务的存算分离架构实践，为后续进一步进行资源优化打下基础。

Part 04 未来规划

4 未来发展


STEP 1
资源调度
统一资源调度
STEP 2
任务稳定性
保证 AM 稳定性，不被压制或驱逐
STEP 3
提升利用率
进一步提升在线利用率
CPU 密集型
STEP 4
任务资源优化
持续对 Spark Job 的资源优化，解决 Spark Sql 和 MR 任务优化问题。

THANKS

软件正在重新定义世界

Software Is Redefining The World