FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

大规模作业的稳定性优化实践

邱从贤
Apache Flink Committer/腾讯高级开发工程师

FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

#1 平台介绍

#2 稳定性建设

#3 总结与展望

FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

#1 平台介绍

实时计算的应用

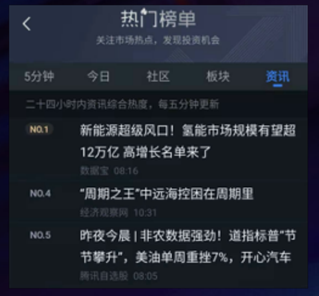
从Business Intelligence走向Continuous Intelligence

基金推荐(实时标签)

理财通实时活动投放

微加卡实时营销

基金实时交易看板



自选股实时热门榜单

基金实时浏览统计

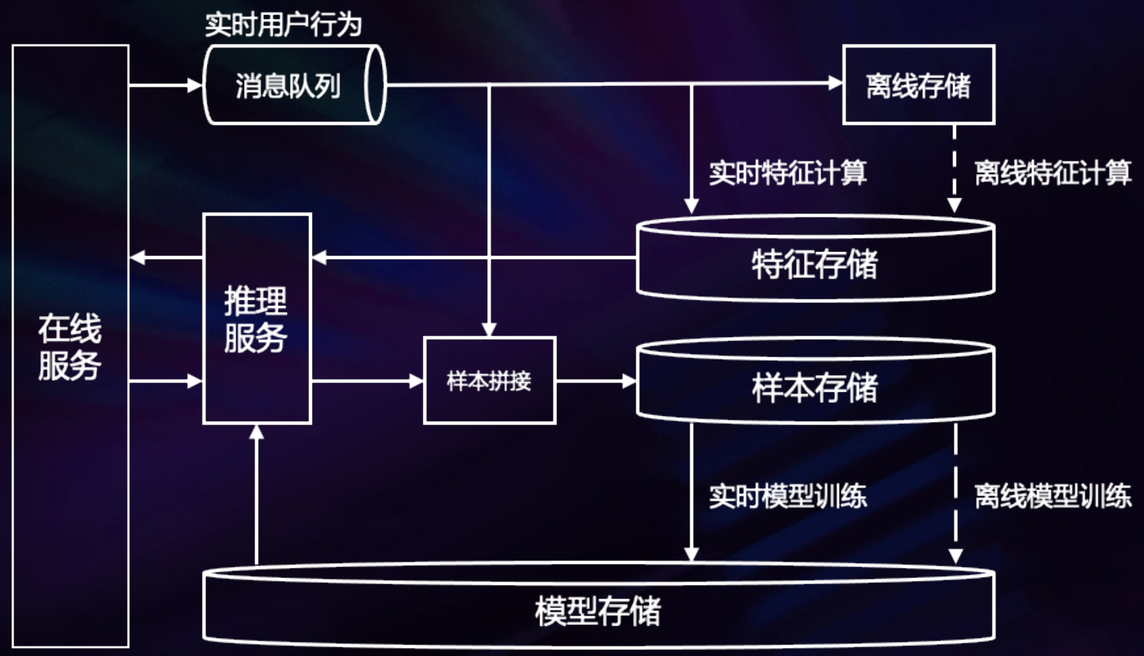
营销分析、指标提升

运营监控、商业决策

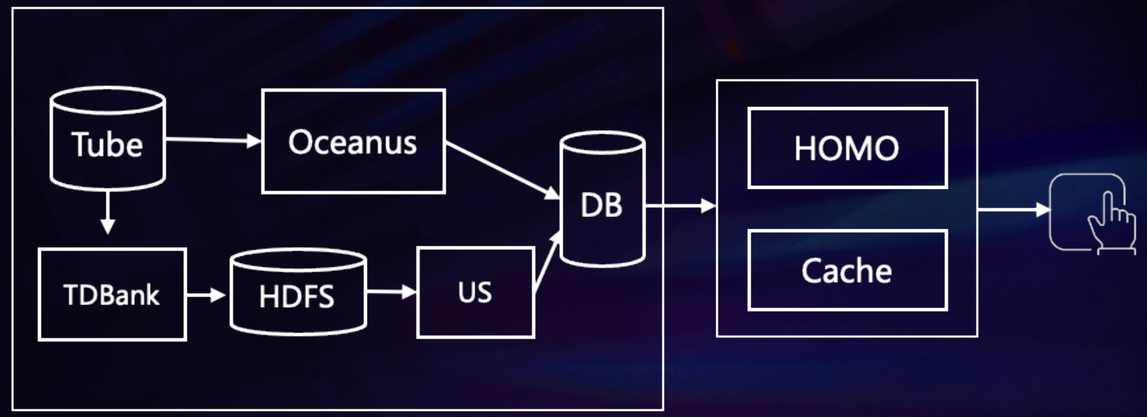
产品改进、体验升级

以财付通为例，实时计算在微加卡实时营销、实时智能推荐、基金实时营销分析、实时自助分析等项目中扮演重要的作用，理财通实时活动和信用卡营销效果取得了巨大的提升

实时计算的应用

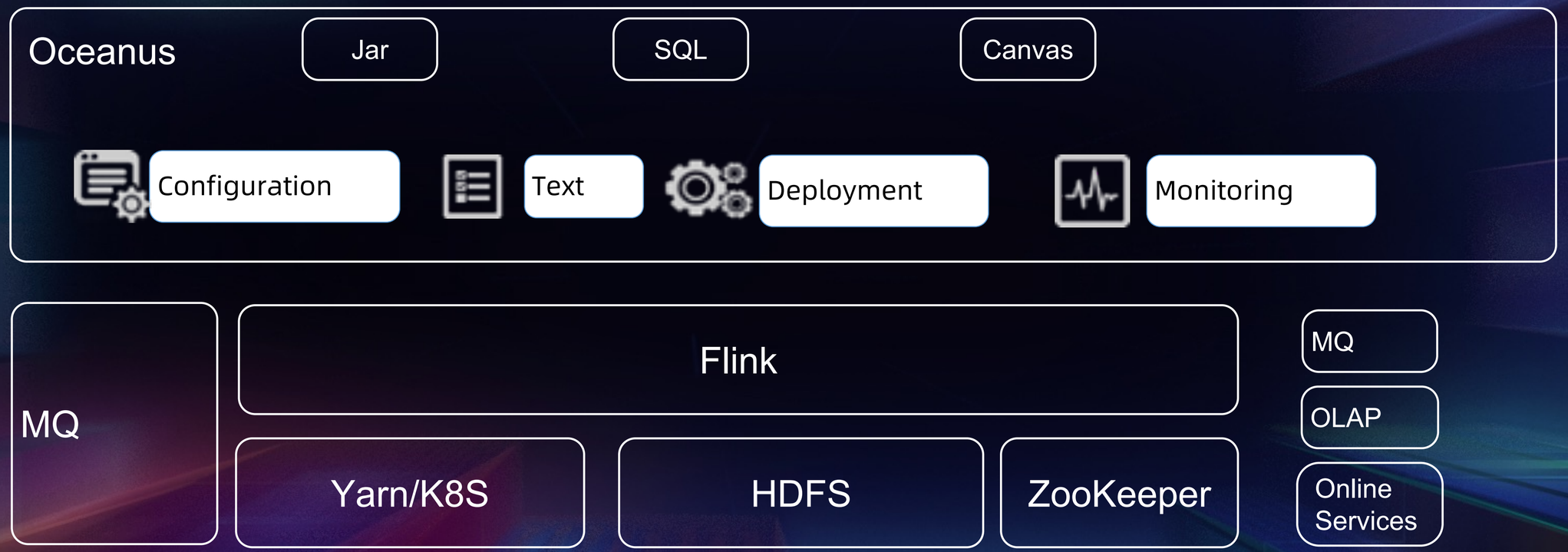


腾讯广告中的实时计算



腾讯视频中的实时计算

Oceanus 平台概况



FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

#2 稳定性建设

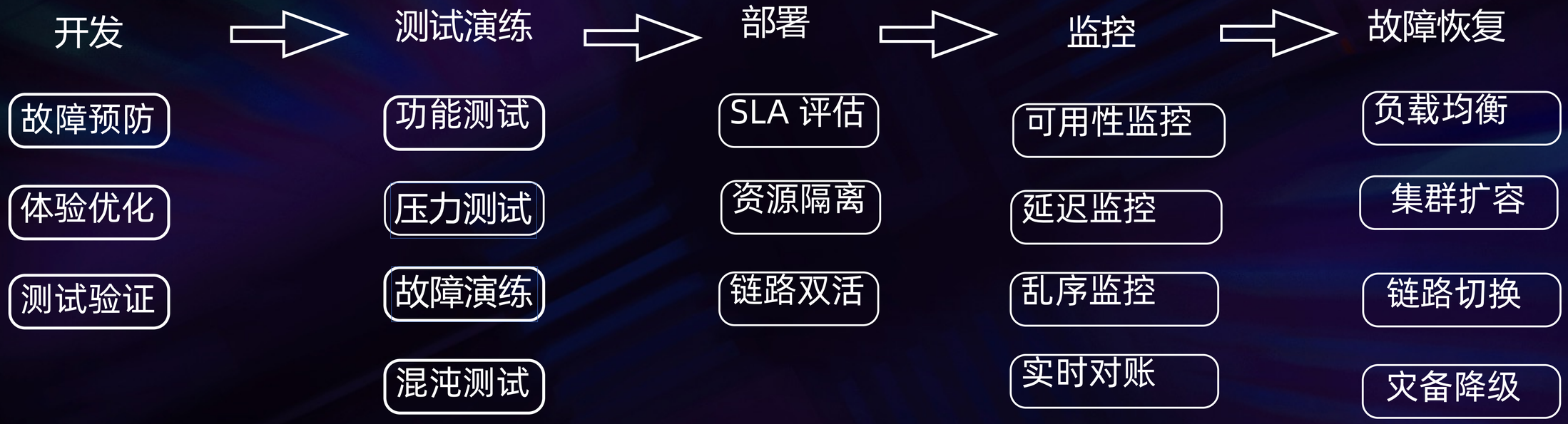
稳定性衡量要素

减少故障

空间上降低影响

时间上降低影响

稳定性保障方案


开发
故障预防
体验优化
测试验证
测试演练
功能测试
压力测试
故障演练
混沌测试
部署
SLA 评估
资源隔离
链路双活
监控
可用性监控
延迟监控
乱序监控
实时对账
故障恢复
负载均衡
集群扩容
链路切换
灾备降级

作业开发流程

FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

Jar 开发流程

```
DataStream<Tuple2<String, Integer>> counts =
    // split up the lines in pairs (2-tuples) containing: (word,1)
    text.flatMap(new Tokenizer())
    // group by the tuple field "0" and sum up tuple field "1"
    .keyBy( ...fields: 0).sum( positionToSum: 1);

// emit result
if (params.has( value: "output")) {
    counts.writeAsText(params.get("output"));
} else {
    System.out.println("Printing result to stdout. Use --output to specify output path.");
    counts.print();
}
```

Jar 代码 开发 ⇨ 测试 ⇨ 上线

作业开发流程

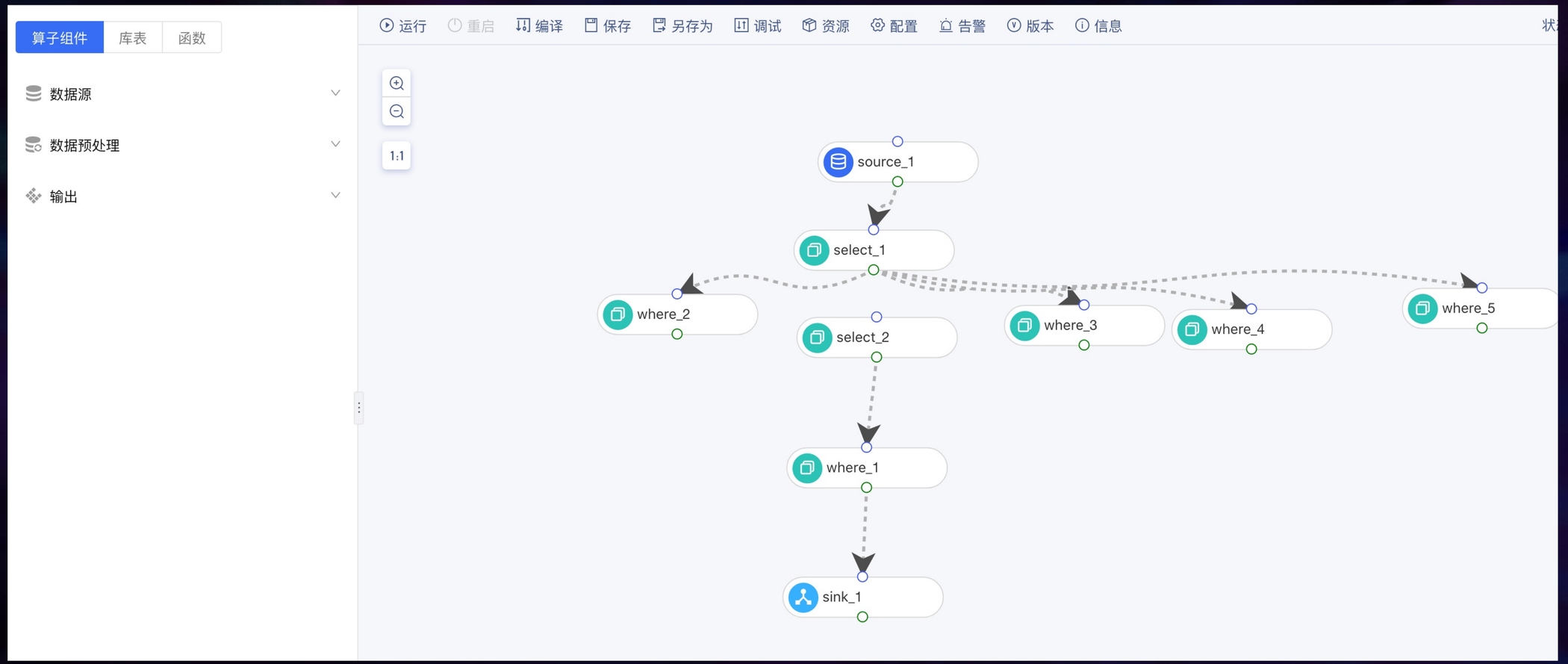
FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

SQL 开发流程

SQL 代码 开发 ⇨ 调试 ⇨ 上线

作业开发流程

Canvas 开发流程



Canvas 开发 ⇨ 调试 ⇨ 上线

作业开发流程

	开发方式	可能的问题	解决方案
Jar	Java 代码	开发调试困难 API 太底层 冲突较多	使用 SQL/Canvas 模式 提前检测 class 的冲突
SQL	SQL 代码	表结构不明显（字段信息不明显） 报错信息定位不方便	优化报错信息提示 提供 Debug 验证模式 使用 Canvas 模式
Canvas	图形化	语义不对齐	完善语义

故障演练

	故障类型	故障影响	如何恢复
server	server 故障/db 故障	影响作业启停 不影响作业运行	重启 server 或者 db
Yarn	RM 故障	无法启动	等待 RM 恢复
	NM 故障	NM 所在机器所有 container 作业重启	Flink 自动恢复
	资源不足	无法启动作业	扩容
HDFS	服务不可用	无法启动作业，运行中作业可能重启	等待 HDFS 恢复
	DN/NN 慢	作业无法启动，运行中作业可能重启	等待 HDFS 恢复
	容量满	无法启动作业，无法完成 checkpoint 无法自动 failover	扩容
Zookeeper	服务不可用 服务不稳定导致 leader 切换	无法启动作业，运行中作业可能 failover 无法正常恢复	等待 Zookeeper 恢复或者切换 zookeeper 集群
Flink 运行异常	Flink/用户逻辑异常	作业 failover	Flink 自动恢复

压力测试

通过不断增加压力了解各系统的能力边界

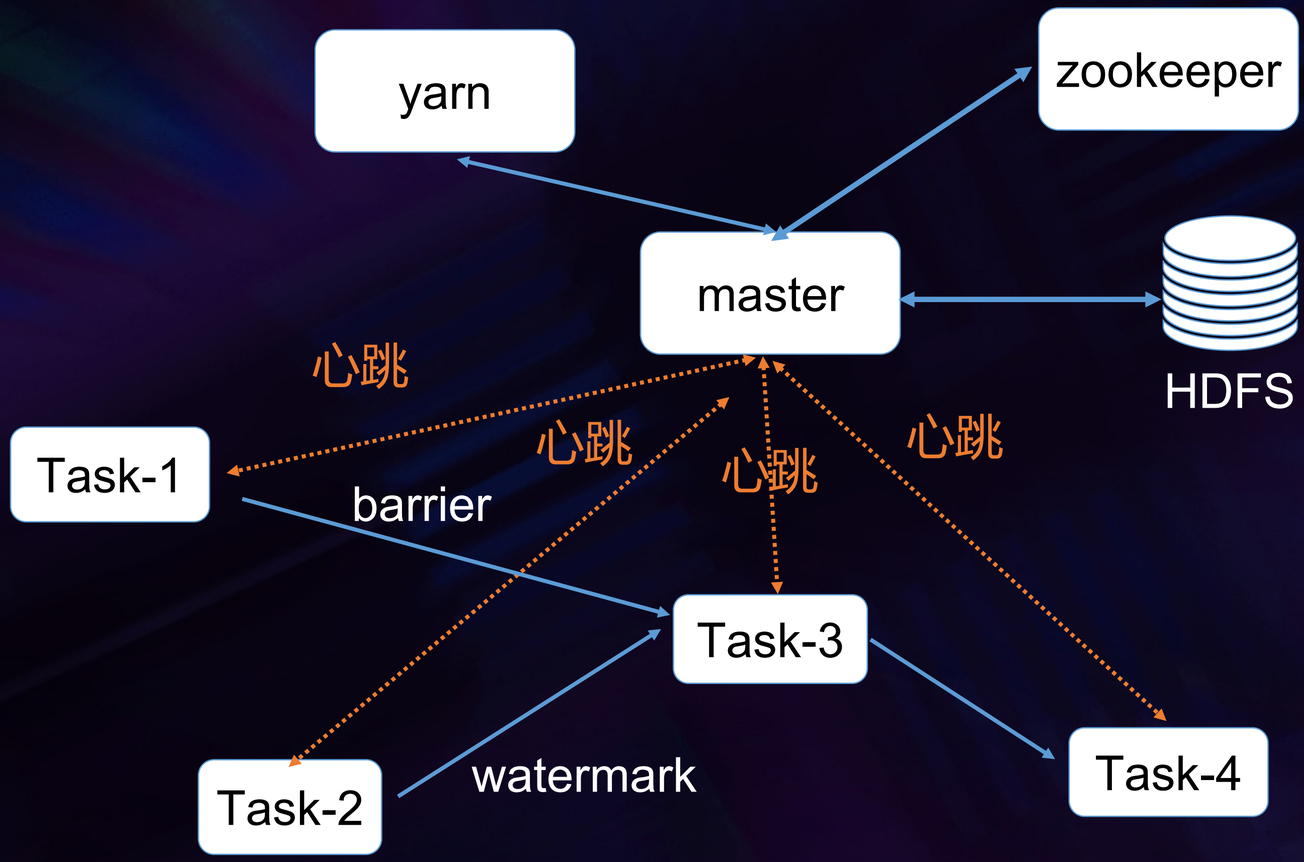
	测试项	优化	释义
server	不同 QPS 请求下，server 的请求成功率以及操作时延	水平扩容	掌握 server 的整体 SLA 以及扩容时机
Yarn	不同 application/container 规模下，集群的响应速度	资源隔离 降低硬件故障的影响	掌握不同规模集群能支撑的作业数
HDFS	给定集群规模下，能支持的 checkpoint 大小以及频率	小文件合并	掌握集群能支撑的 checkpoint 大小以及频率
Zookeeper	给定集群能支撑的作业数/TaskManager 数	降低 TM 对 Zookeeper 的连接依赖（TaskExecutor 启动或者 HeartbeatTimeout 后重新连接 zookeeper）	掌握集群能支撑的作业数/TaskManager 数

部署阶段

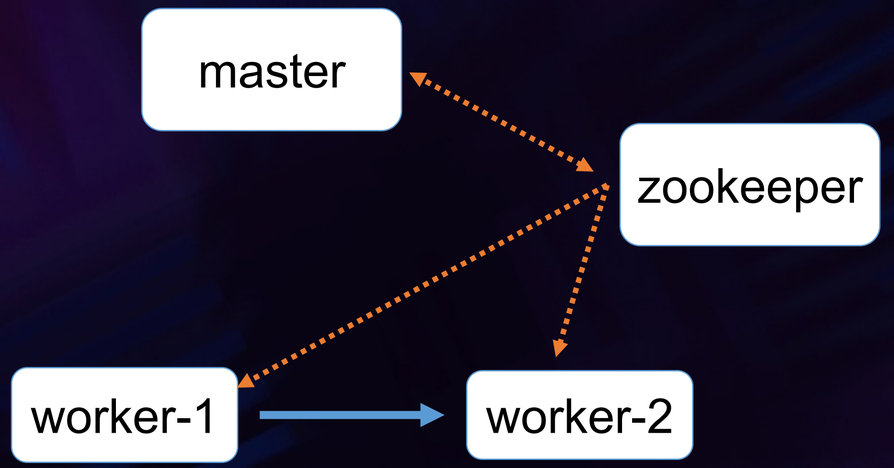
评估作业的接入，将可能的影响降到最低

	评估指标	处理措施
server	接入后能否满足 SLA 要求	水平扩容
Yarn	接入后集群能否支撑 application/container 的需求	集群扩容 接入新集群
HDFS	接入后集群能否支撑 checkpoint 的读写	集群扩容 接入新集群
Zookeeper	接入后集群能否支撑连接数/znode 数	Zookeeper 扩容/隔离

作业运行情况


yarn
zookeeper
master
HDFS
心跳
心跳
心跳
心跳
Task-1
barrier
Task-3
Task-2
watermark
Task-4

master/worker 的稳定性



可能的问题

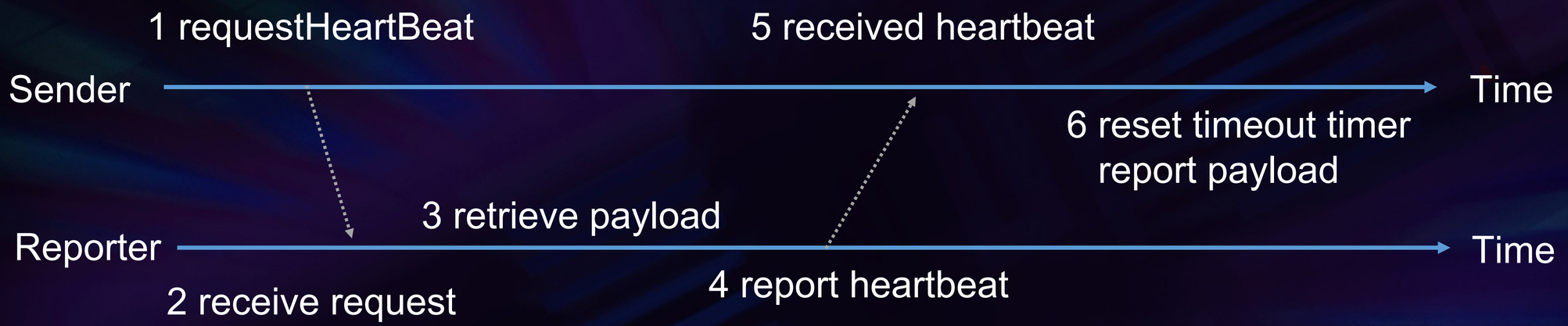
- 与Zookeeper 断开链接导致 lost leadership 作业 failover
- 资源超用被 kill
- 其他 container 使用资源过多影响
- GC 频繁导致

优化方案

- 容忍 Zookeeper suspended 事件
- 资源超用预警
- 资源隔离
- GC 告警

控制链路的稳定性

心跳模型

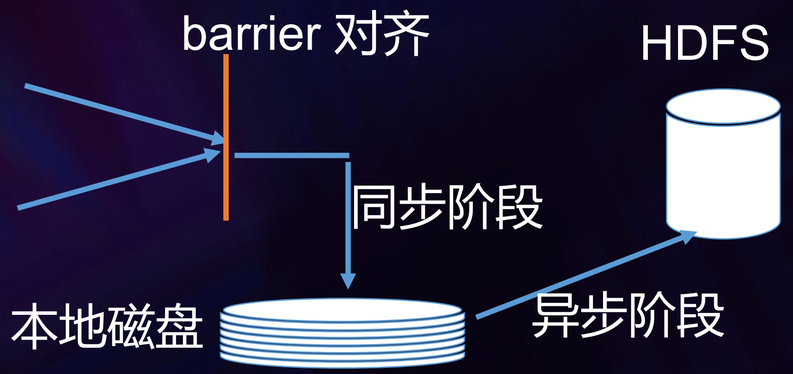


可能的影响因素
- 网络
- RPC 响应耗时（等待处理&实际处理耗时）
- 外部影响（GC 等）

相关指标
- 物理机的网络情况
- RPC 的响应耗时
- container 的 gc 情况

控制链路的稳定性

snapshot 流程



可能的问题
1. 上游反压导致 barrier 无法对齐（outPoolUsage 100%）
2. snapshot 同步
   A. 无法抢到锁
   B. Task 主线程太忙
3. 异步阶段的问题
   1. 磁盘 IO 高
   2. HDFS 不稳定

解决方案
1. 采集 task 相关指标进行告警
2. snapshot 同步
   A. 增加 snapshot 同步等待时间指标，并提供相关线程栈信息展示
   B. Task 主线程太忙 — 增加 CPU 使用率相关告警（container CPU 超用告警）
3. 异步阶段的问题
   1. 使用 SSD 提高 IO 性能
   2. 优化 Flink 使用 HDFS 的方式

数据链路的通畅

数量链路的通畅：是否反压；是否数据倾斜

	影响	指标
反压	数据不能及时处理	outqueue 100%
数据倾斜	数据可能无法及时处理	不同算子间处理数据量差异

作业监控

	关注指标	相关监控
container 稳定性	- master failover 次数 - GC 情况 - FD 数目 - 磁盘占用情况	- master fail 次数 - 作业 fail 次数 - container 资源使用情况监控（CPU，Memory，FD 等）
master/worker 控制链路稳定性	- 心跳相关指标（GC，RPC 等） - snapshot 相关指标（barrier 对齐，同步和异步耗时）	- RPC 耗时 - snapshot 各阶段耗时
worker 间数据链路稳定性	- 反压（outqueue） - 数据倾斜(inputqueue/input reocrds)	- inqueue/outqueue 使用率 - 不同并发间数据的处理情况

故障恢复

	切换条件	实施动作	依赖
负载均衡	集群间的压力不均衡，部分作业出现故障	作业迁移	checkpoint 跨集群迁移
链路切换	双跑链路其中一条链路出现故障	能自动切换	
集群扩容	集群资源不够导致作业故障	扩容集群	
灾备降级	集群资源不够，且暂时无法扩容	按优先级进行作业降级	
作业故障	作业 fail	自动 failover（无法自动 failover 的收到告警后人工干预）	

作业故障恢复

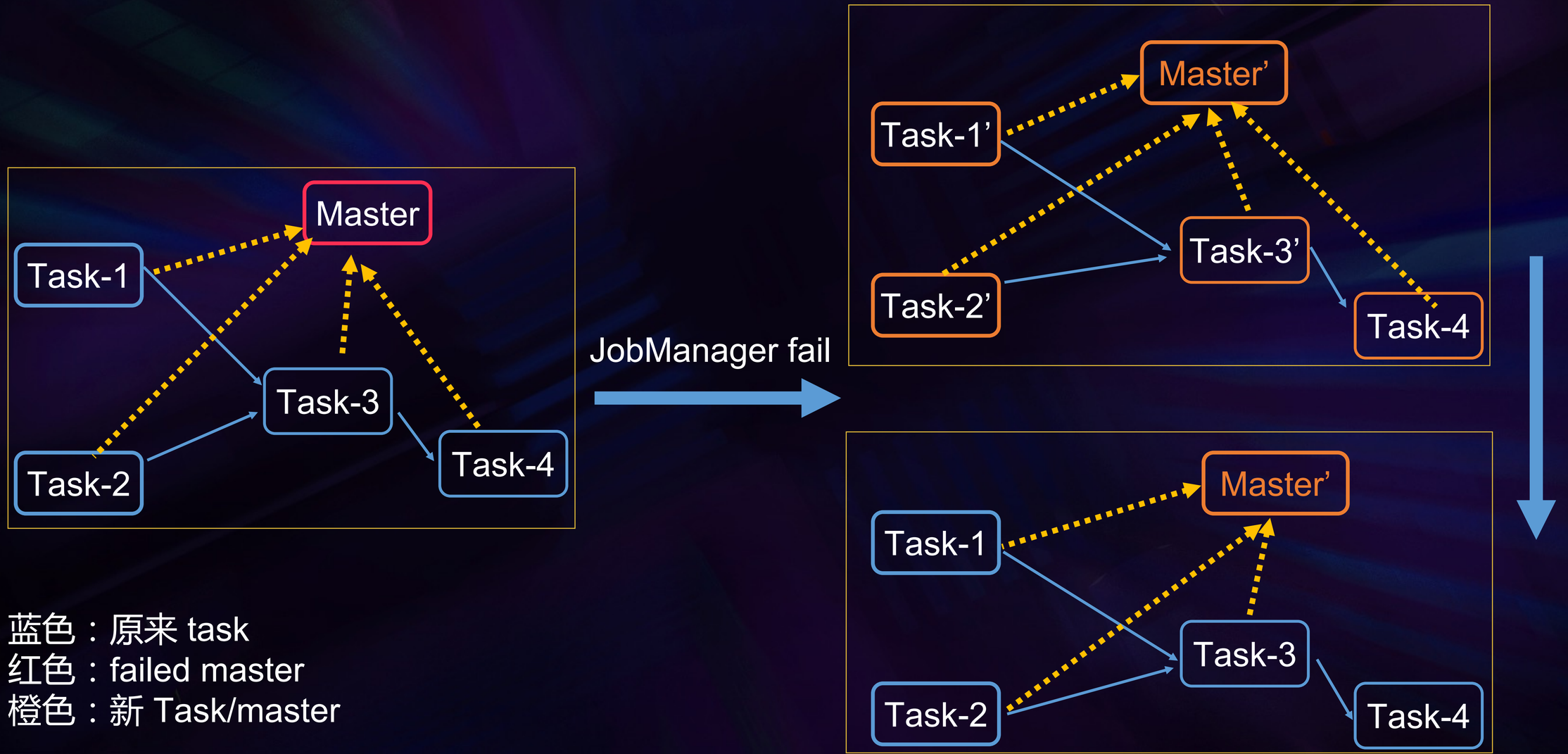
Flink 作业的故障恢复

1. Flink 检测到 fail
2. 按照 global/region failover 的策略重新调度 task

优化

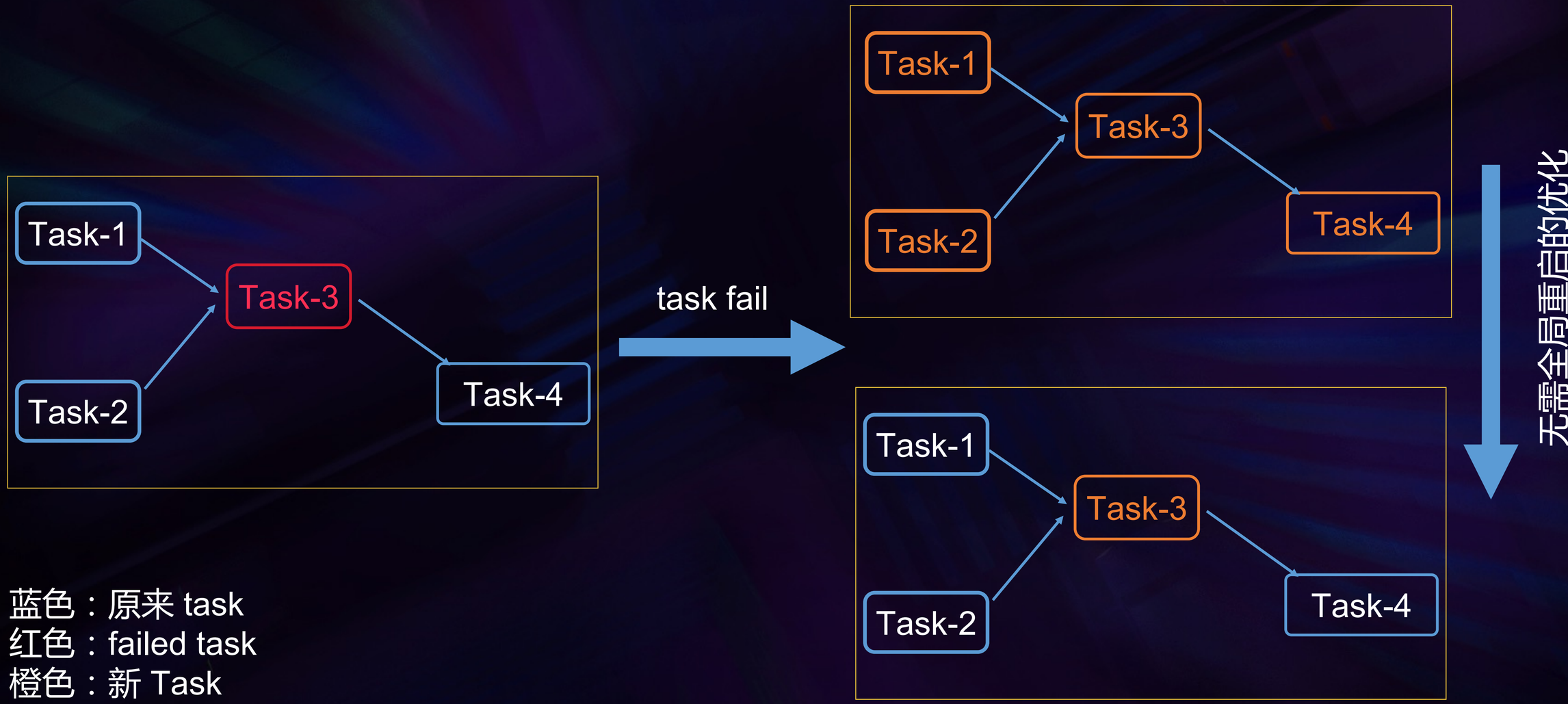
1. 从空间上降低影响
   - 无需全局重启的 master failover
   - 无需全局重启的 Task 故障恢复—单点重启
2. 从时间上降低影响
   - 优化 failover 速度

无需全局重启的 master failover



蓝色：原来 task
红色：failed master
橙色：新 Task/master

无需全局重启的 Task 故障恢复（有损）



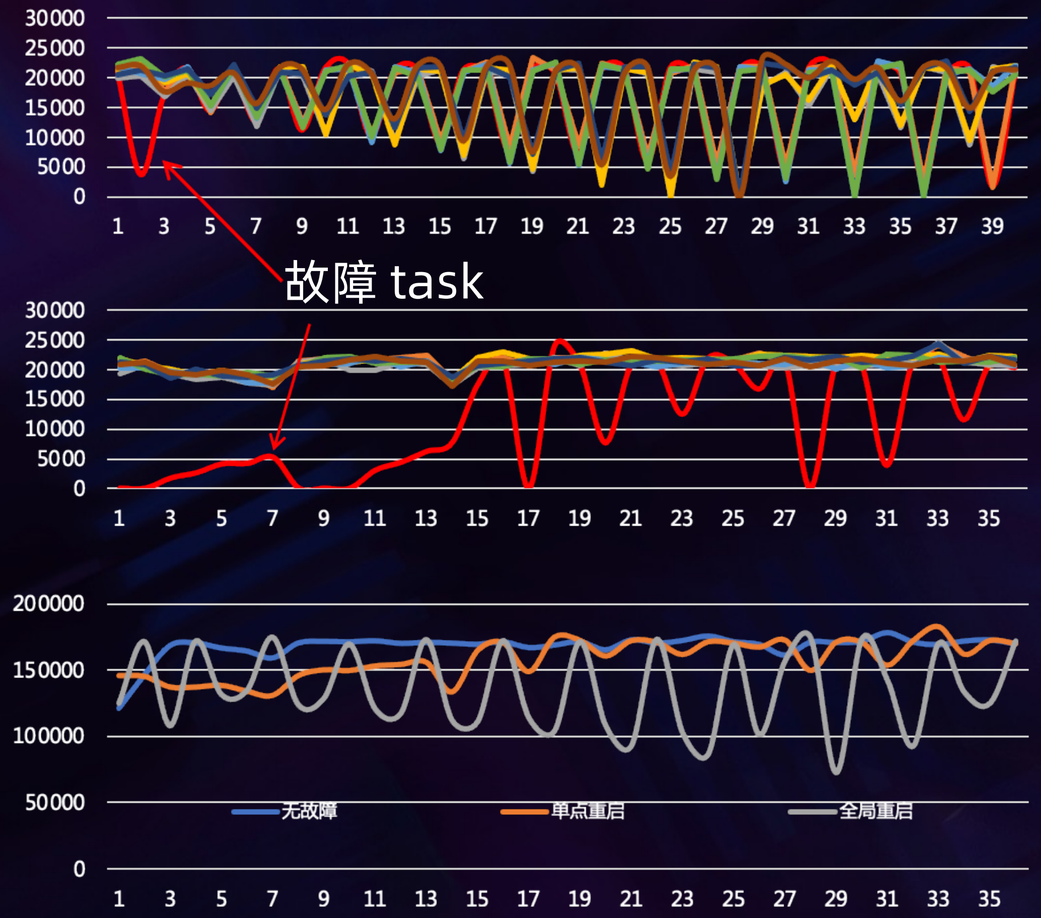
蓝色：原来 task
红色：failed task
橙色：新 Task

无需全局重启的 Task 故障恢复（有损）

全局恢复-Task粒度
单个Task故障导致
其他Task吞吐和延迟受到影响

单点恢复-Task粒度
单个Task故障几乎
不会影响其他Task的吞吐和延迟

作业粒度
单点重启保障作业
大部分数据仍然得到有效处理

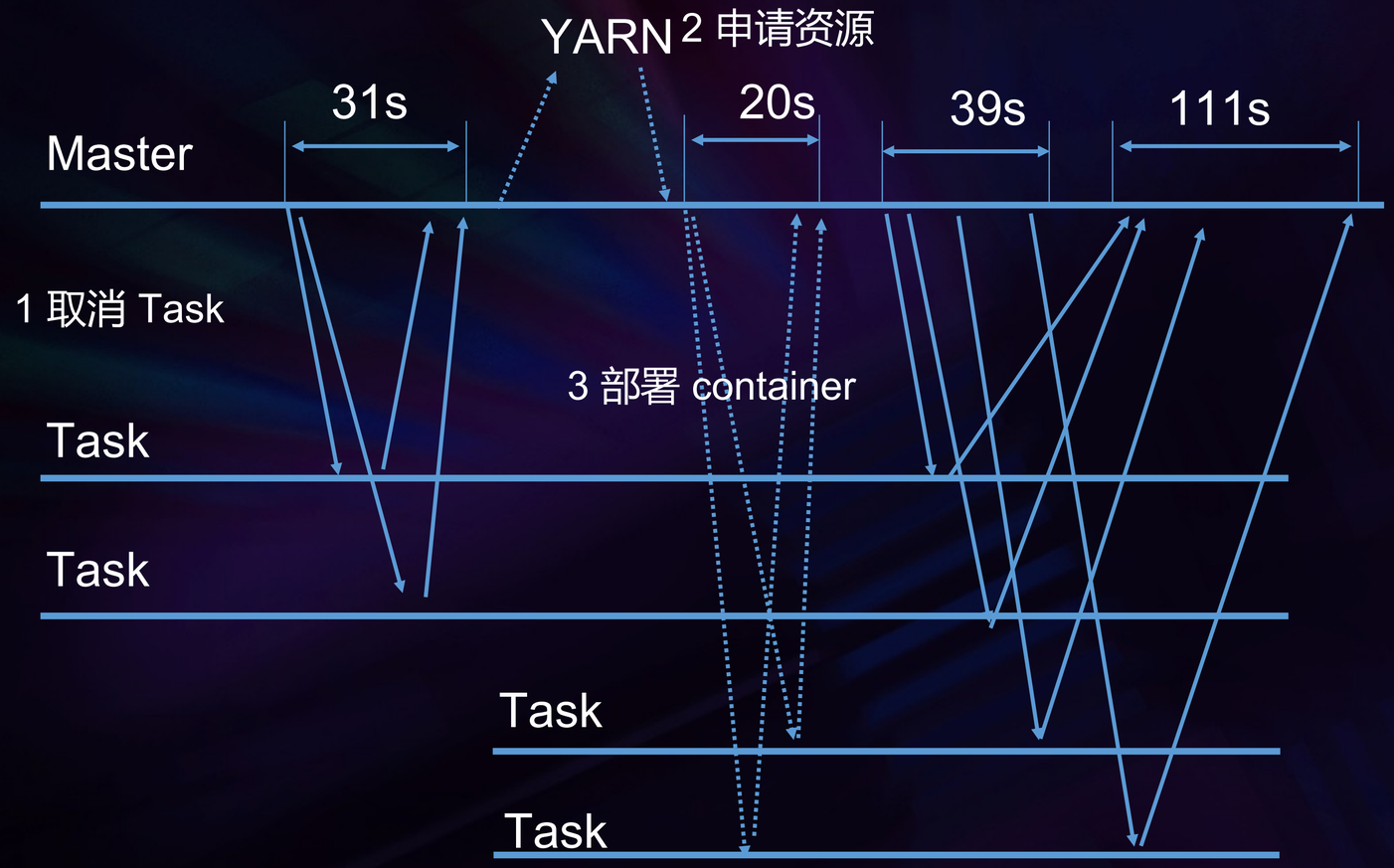


当Task发生故障时，仅重启发生故障的Task，
而不需要重启所有Task，
降低数据整体的平均延迟或计算整体的可用性

目前已在**腾讯广告太卜算法中台**上线，
支持作业规模超过**66万核**

广告模型业务，相对于数据完整性，
更关心数据的实时性

Task 故障恢复速度优化

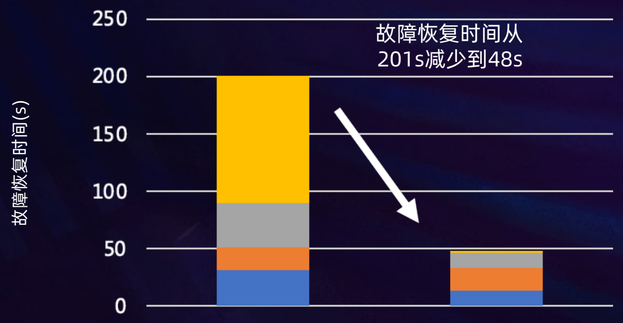


作业 Task 故障恢复流程（Container 故障，重新申请资源）

并发度 4761

主要瓶颈
master 主线程需要处理 Task 启动阶段大量 RPC 请求
Container 拉取文件慢
Container 需即时申请

解决方案
优化分布式协议，去除不必要的 RPC 请求
合并 Jar 包，减少 Container 启动所需的文件数目
允许额外的备份 Container，避免故障时再去申请部署



FLINK FORWARD #ASIA 2021 ONLINE 实时即未来 REAL-TIME IS THE FUTURE

#3 总结与展望

总结

开发部署

SQL 体验优化
Jar 冲突检测
Canvas 语义对齐

演练测试

功能测试
压力测试
故障演练

部署

SLA 评估
资源隔离
链路双活

监控

可用性
延迟监控
乱序监控

故障恢复

master failover
有损单点重启
优化恢复速度
localKeyBy/Backpressure-aware-Selector

展望

开发部署

智能 IDE

智能配置

演练测试

混沌测试

部署

在资源隔离的情况下提高使用率

监控

智能诊断和优化

故障恢复

无损单点重启

大状态的快速恢复


FLINK FORWARD
#ASIA 2021
ONLINE
实时即未来
REAL-TIME IS THE FUTURE
THANKS