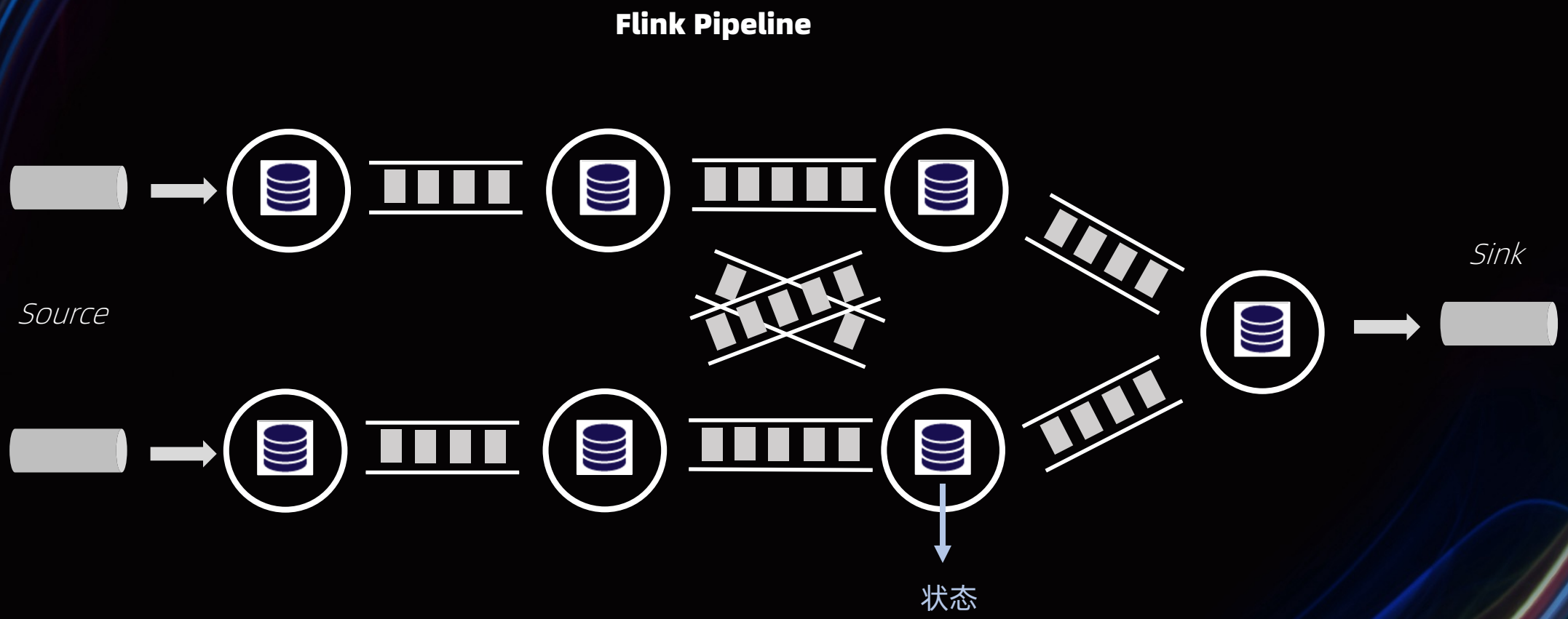
FLINK FORWARD # ASIA 2022 ONLINE | 实时即未来

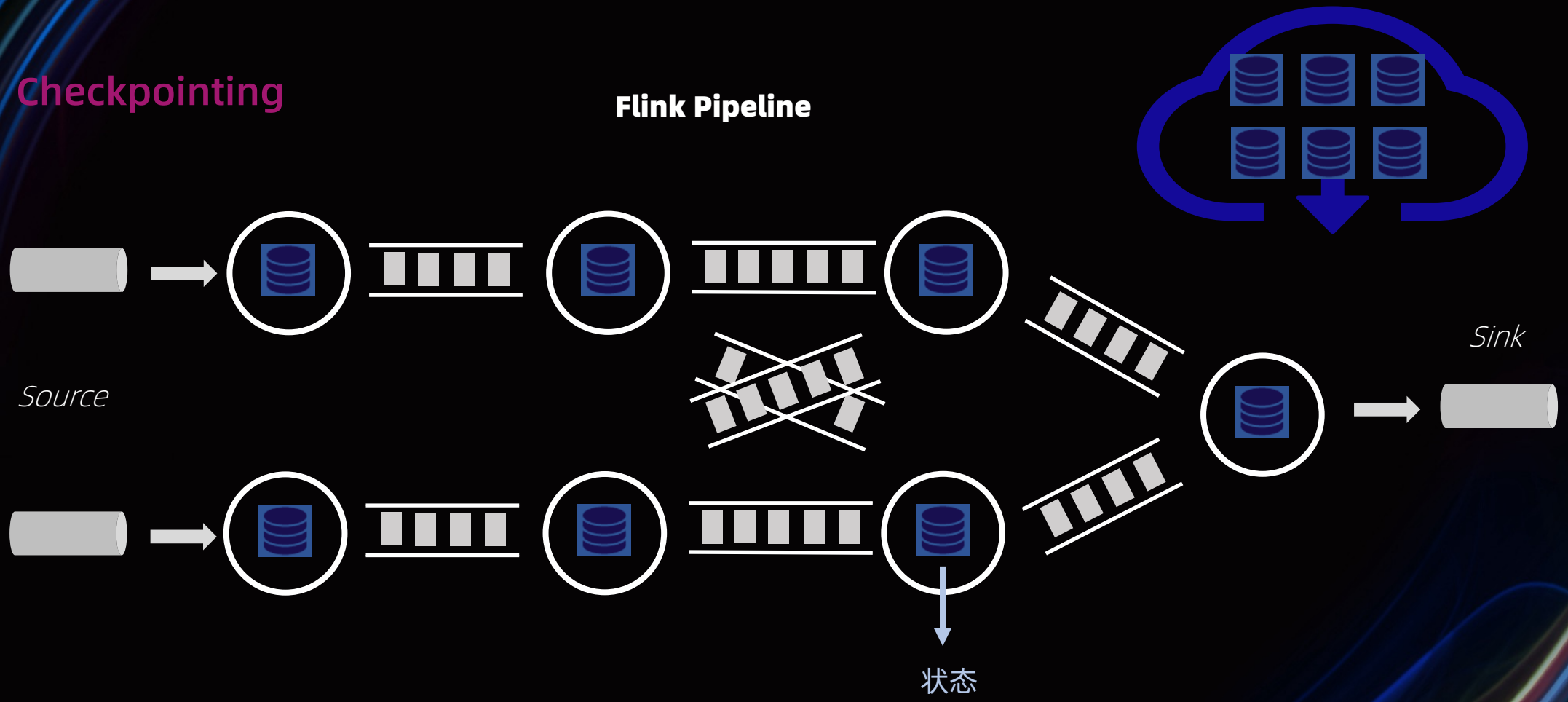
Flink容错恢复2.0最新进展

梅源
阿里云Flink存储引擎团队负责人
Apache Flink引擎架构师
Apache Flink PMC & Committer

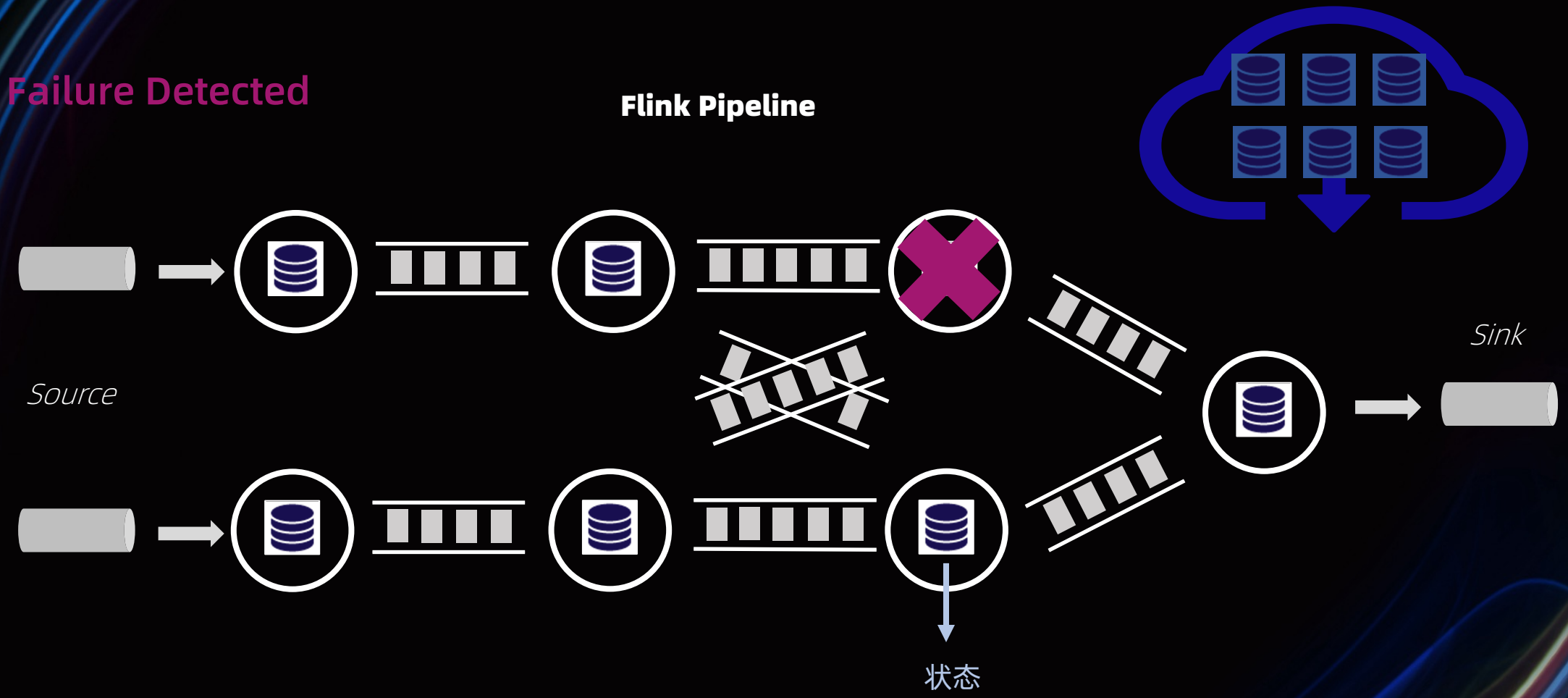
Flink 容错恢复



Flink 容错恢复

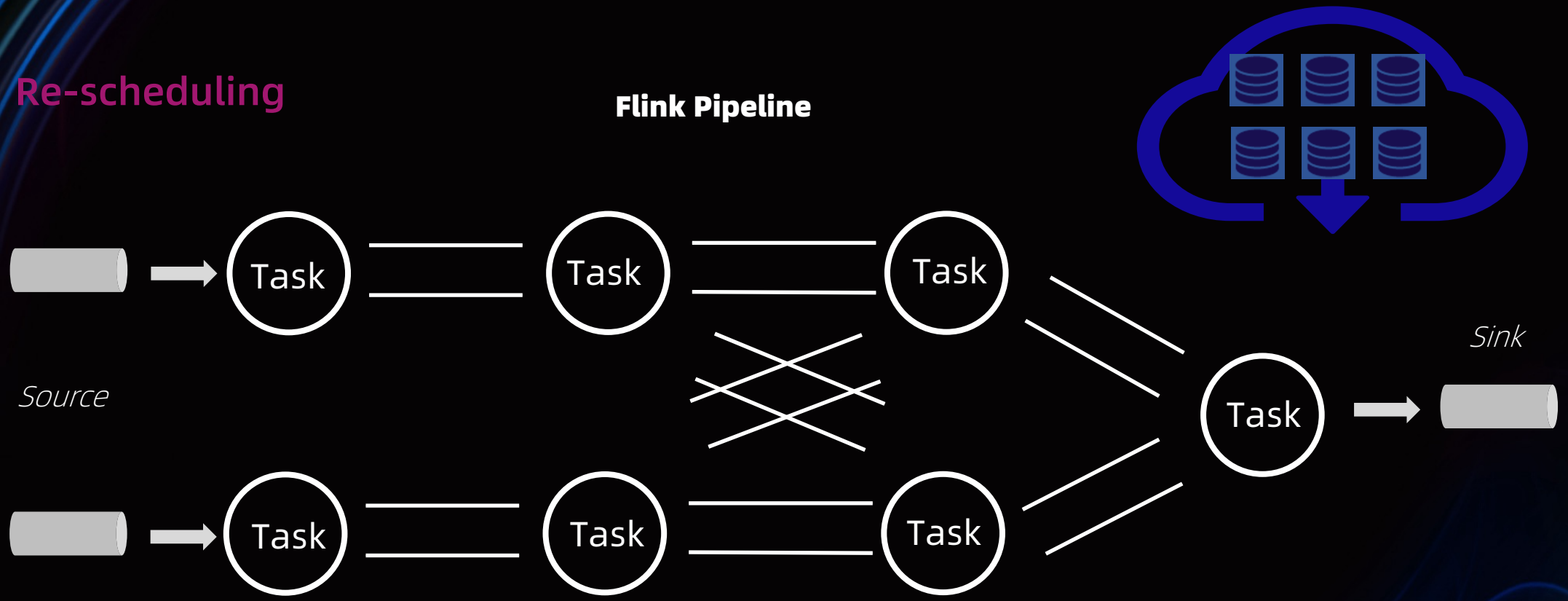


Flink 容错恢复



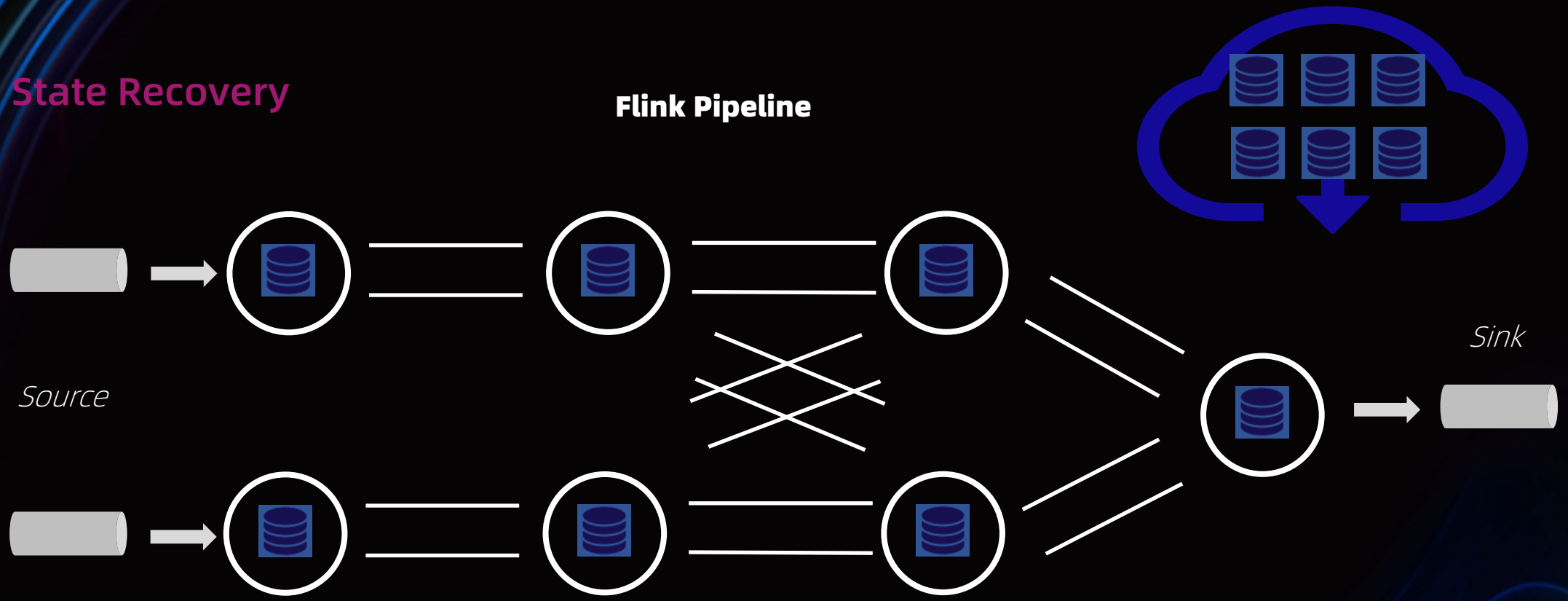
Flink 容错恢复

Re-scheduling



Flink 容错恢复

State Recovery



Flink 容错恢复 2.0

全链路

- → Checkpointing
- → Failure Detection
- → Re-scheduling
- → State Recovery

多维度

- → Processing Latency
- → Resource Cost
- → Data Consistency Level
- → Recovery Behavior

云原生

- → Fast Elasticity
- → Across Region Durability
- → Simple Dependency
- → Extendibility

Flink 容错恢复 2.0

全链路

- Checkpointing
- Failure Detection
- Re-scheduling
- State Recovery

多维度

- Processing Latency
- Resource Cost
- Data Consistency Level
- Recovery Behavior

云原生

- Fast Elasticity
- Across Region Durability
- Simple Dependency
- Extendibility

Checkpointing

Unaligned Checkpoint + Buffer Debloating

Generic Incremental Checkpoints

Incremental Native Savepoint

Scheduling

Approximate Task-Local Recovery

At-least-once Task-Local Recovery

Job Hot Update

State

State-Local Recovery + Working Directory

Rescale Improvement

Tiered State + Lazy Load

Flink 容错恢复 2.0

全链路

- Checkpointing
- Failure Detection
- Re-scheduling
- State Recovery

多维度

- Processing Latency
- Resource Cost
- Data Consistency Level
- Recovery Behavior

云原生

- Fast Elasticity
- Across Region Durability
- Simple Dependency
- Extendibility

Checkpointing

Unaligned Checkpoint + Buffer Debloating

Generic Incremental Checkpoints

Incremental Native Savepoint

阿里云实时计算企业级特性

Approximate Task-Local Recovery

At-least-once Task-Local Recovery

Job Hot Update

State

State-Local Recovery + Working Directory

Rescale Improvement

Tiered State + Lazy Load

→ 优化快照生成

升级分布式快照架构

快速稳定的 Checkpoints

→ 优化作业恢复和扩缩容

优化本地状态重建

云原生分层状态存储架构升级

简化重新调度的步骤

→ 优化快照管理

明确快照生命周期管理

增量 Native Savepoint

→ 优化快照生成

升级分布式快照架构

快速稳定的 Checkpoints

→ 优化作业恢复和扩缩容

优化本地状态重建

云原生分层状态存储架构升级

简化重新调度的步骤

→ 优化快照管理

明确快照生命周期管理

增量 Native Savepoint

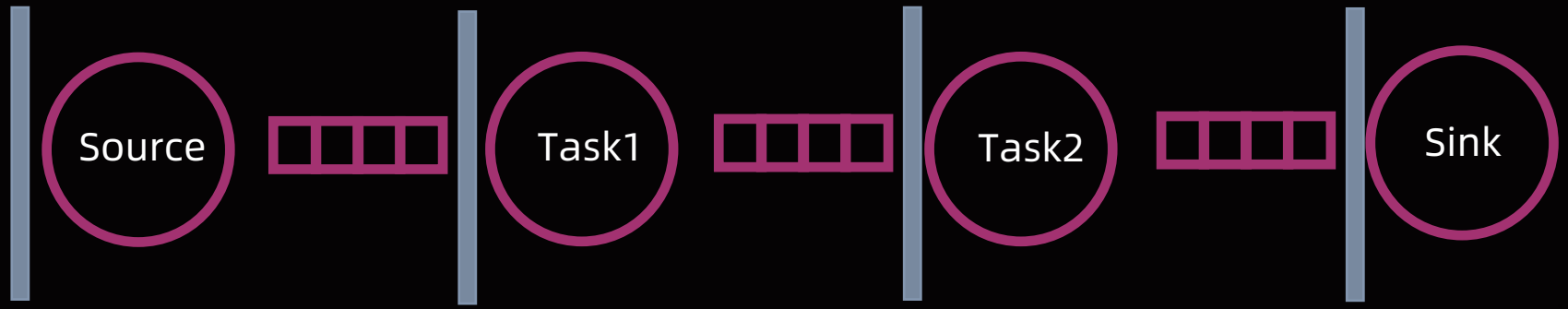
升级分布式快照架构

问题 1

对齐时间长，反压时被完全阻塞

Aligned Barrier

升级前



升级分布式快照架构

问题 1

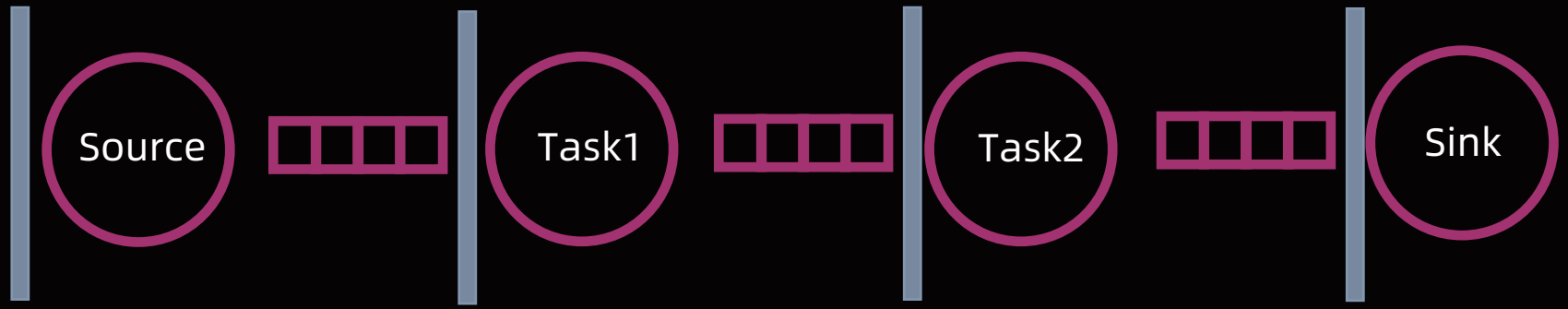
对齐时间长，反压时被完全阻塞

Aligned Barrier

问题 2

固定 Buffer 数目，多余的处理数据

升级前



升级分布式快照架构

升级前

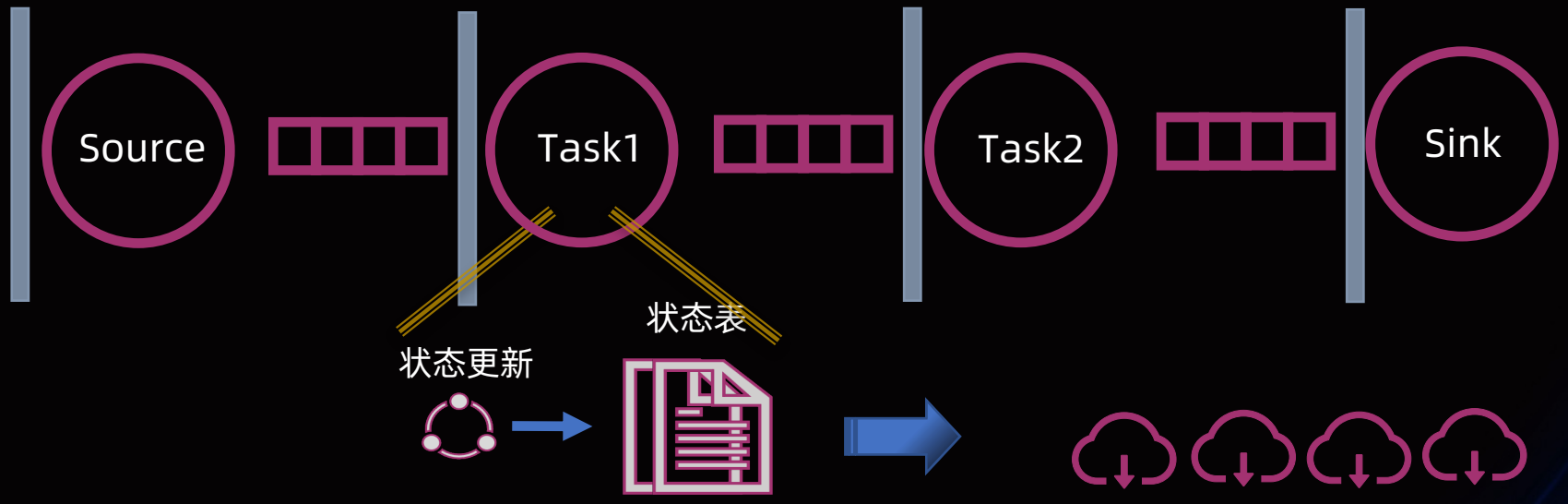
问题 1

对齐时间长，反压时被完全阻塞

Aligned Barrier

问题 2

固定 Buffer 数目，多余的处理数据



问题 3

快照异步上传时间较长且不可控

优化快照生成

影响的因素			
算子间缓存的中间数据	不被流动缓慢的中间数据阻塞	Unaligned Checkpoint	Flink 1.11, 1.12
	更少的中间数据	Buffer Debloating	Flink 1.13, 1.14
算子状态	更小更稳定的算子状态	Generic Log-based Incremental Checkpoints	Flink 1.15, 1.16

升级分布式快照架构

Aligned Barrier

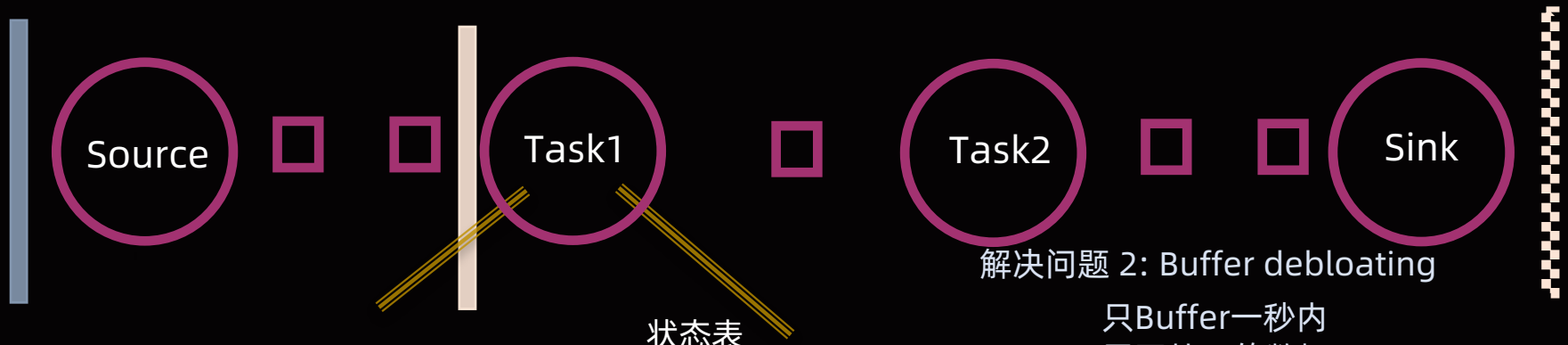
计时超时自动转换成
Unaligned Barrier

解决问题 1: Unaligned Checkpoint

允许透支Buffer解决在Buffer不足时不能及时响应UC的问题

启动全局计时超时机制有效改进超时触发的条件

瞬时推进
Unaligned Barrier



解决问题 2: Buffer debloating

只Buffer一秒内
需要处理的数据

状态表

状态更新

解决问题 3: 通用增量快照

解耦状态表和增量
日志异步上传过程

增量状态更新日志

持久存储 DFS

快照算子状态

原始设计

同步快照状态表 + 异步上传快照

快照算子状态

原始设计

问题1: 异步上传的文件大小依赖 State Backend 实现

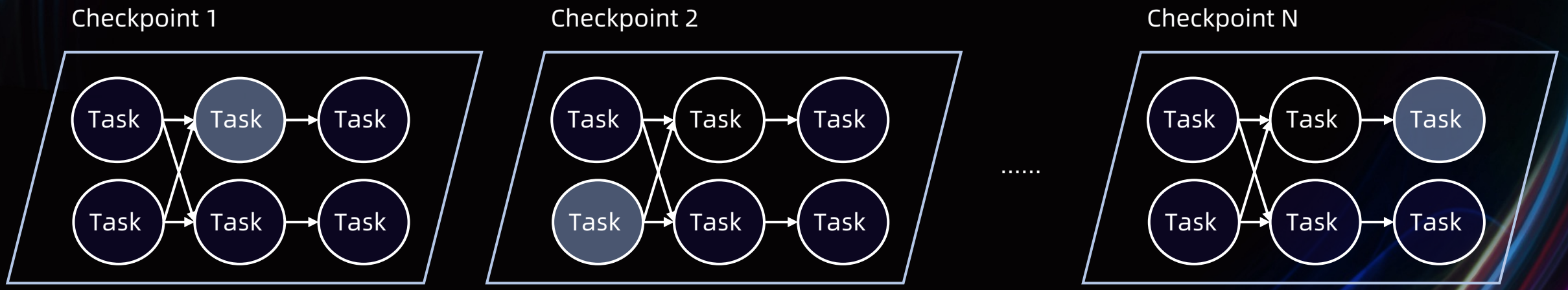
问题2: 在同步快照结束前无法开始异步上传过程

同步快照状态表 + 异步上传快照

快照算子状态

原始设计

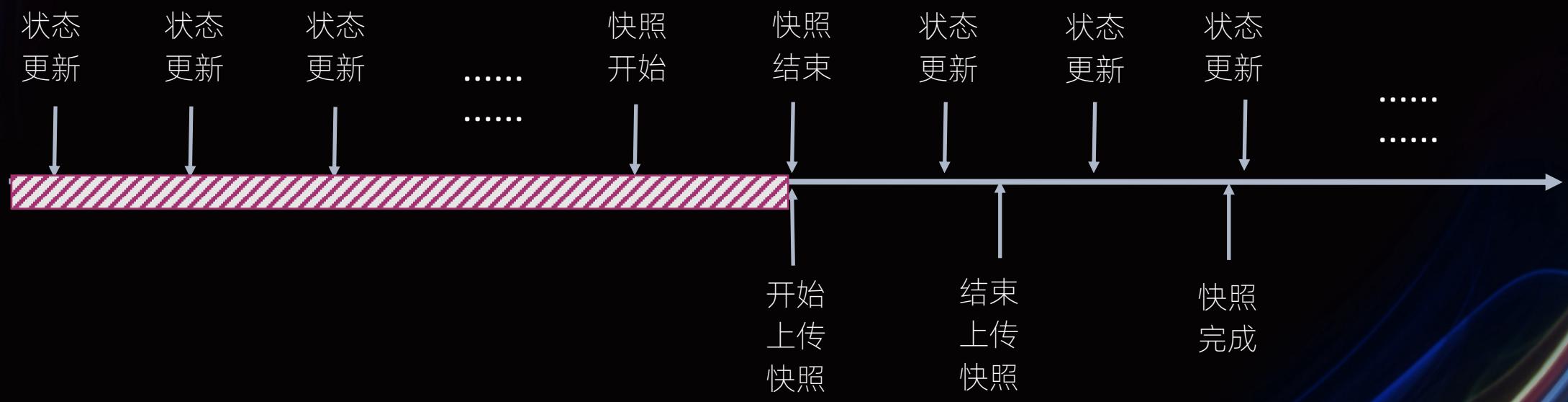
问题1：异步上传的文件大小依赖 State Backend 实现



快照算子状态

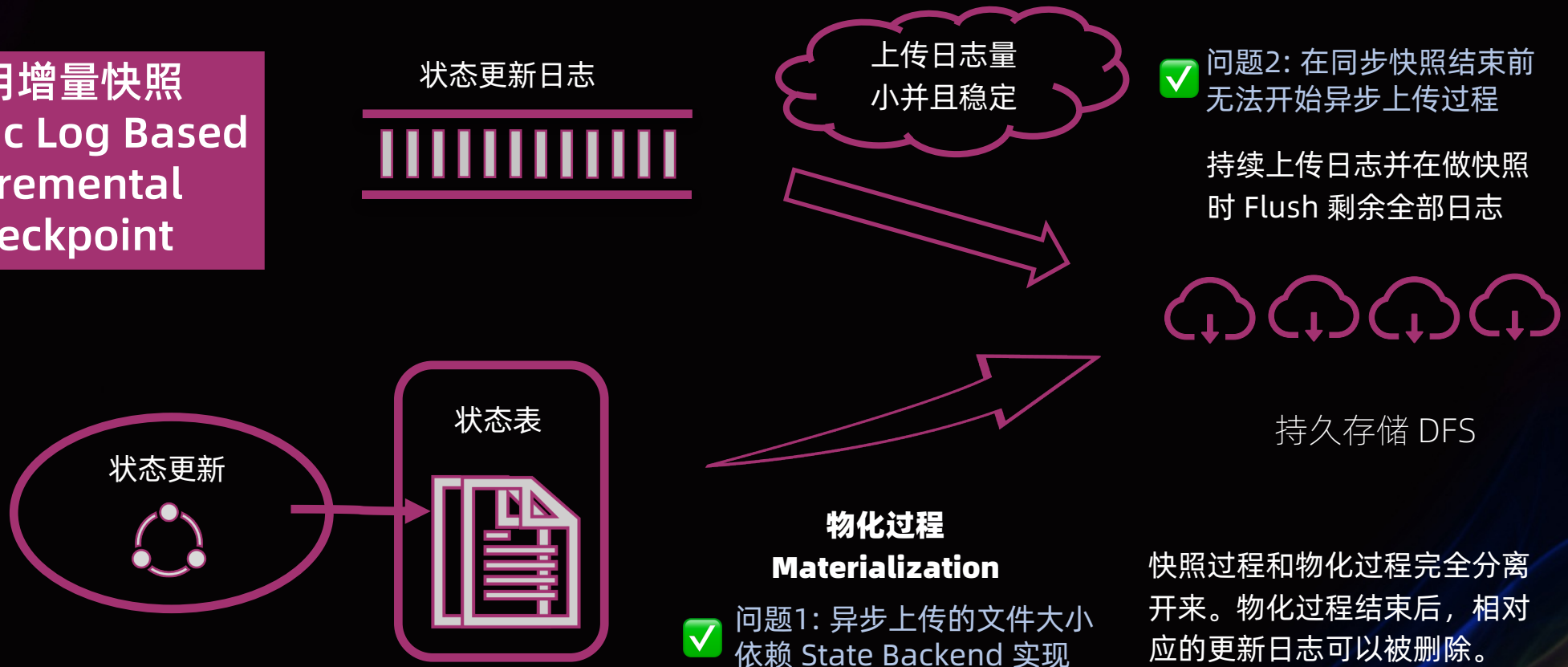
原始设计

问题2: 在同步快照结束前无法开始异步上传过程



通用增量快照

通用增量快照
Generic Log Based Incremental Checkpoint



✅ 问题2: 在同步快照结束前无法开始异步上传过程

持续上传日志并在做快照时 Flush 剩余全部日志

物化过程
Materialization

✅ 问题1: 异步上传的文件大小依赖 State Backend 实现

快照过程和物化过程完全分离开来。物化过程结束后，相对应的更新日志可以被删除。

通用增量快照

Checkpoint = 物化的状态表 (State Table) + 增量更新日志 (Changelog)

通用增量快照
Generic Log Based Incremental Checkpoint

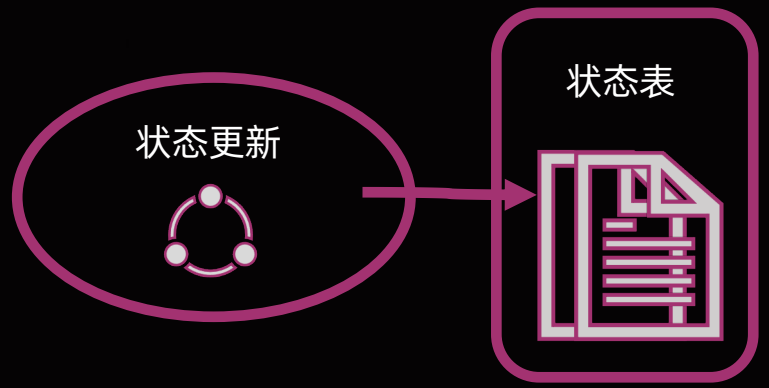
状态更新日志

上传日志量
小并且稳定

✅ 问题2: 在同步快照结束前无法开始异步上传过程

持续上传日志并在做快照时 Flush 剩余全部日志

持久存储 DFS



物化过程
Materialization

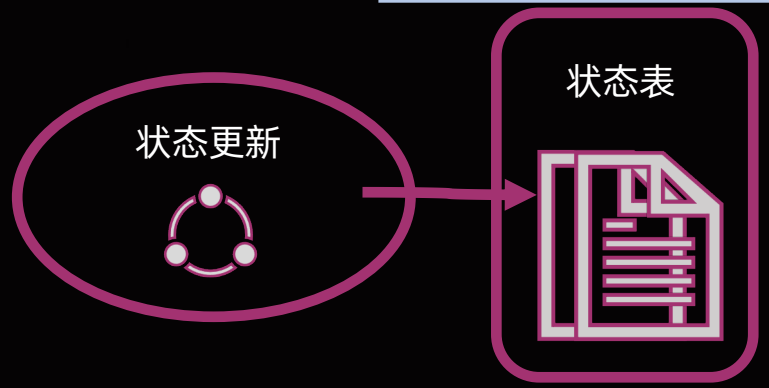
✅ 问题1: 异步上传的文件大小依赖 State Backend 实现

快照过程和物化过程完全分离开来。物化过程结束后，相对应的更新日志可以被删除。

通用增量快照

Checkpoint = 物化的状态表 (State Table) + 增量更新日志 (Changelog)

Changelog Storage (DSTL)



持久存储 DFS

物化过程
Materialization

✅ 问题1: 异步上传的文件大小依赖 State Backend 实现

快照过程和物化过程完全分离开来。物化过程结束后，相对应的更新日志可以被删除。

Changelog Storage (DSTL)

DSTL – Durable Short-term Log

持久化 Durability	**需要短期持久化增量日志，物化后即可删除**
高频写 Write-heavy	**纯 Append 写操作，仅在容错恢复时需要读取**
写延迟 Latency	**99.9% 的写请求需要在 1 秒内完成**
一致性 Consistency	**和现有 Checkpoint 机制提供同级别一致性保证**

通用增量快照 - Trade Off

更稳定的Checkpoint

防止 Checkpoint 耗时突增，
平滑 CPU 曲线，平稳网络流量使用

更快速的Checkpoint

减少 CP 时上传文件的大小
完成秒级Checkpoint

更小的端到端延迟

Checkpoint 越快，
Transactional Sinks 提交越频繁

更少的数据回追

通过设置更小的 Checkpoint 间隔，
提供更快速的容错

Checkpoint 放大

状态双写

通用增量快照

11 月 27 号下午专场 Talk：基于 Log 的通用增量 Checkpoint

→ 优化快照生成

升级分布式快照架构

快速稳定的 Checkpoints

→ 优化作业恢复和扩缩容

优化本地状态重建

云原生分层状态存储架构升级

简化重新调度的步骤

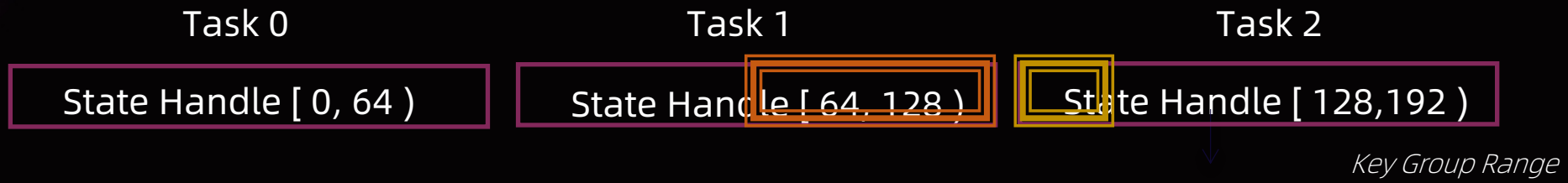
→ 优化快照管理

明确快照生命周期管理

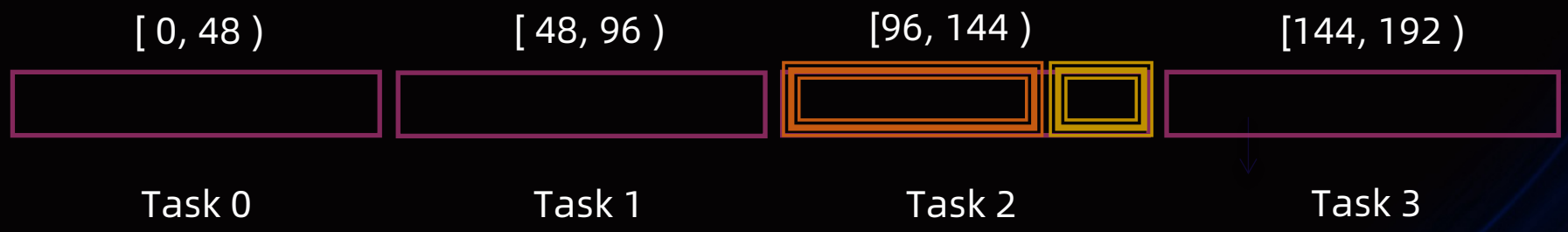
增量 Native Savepoint

扩缩容 Rescaling

Keyed State, Parallelism = 3



Keyed State, Parallelism = 4

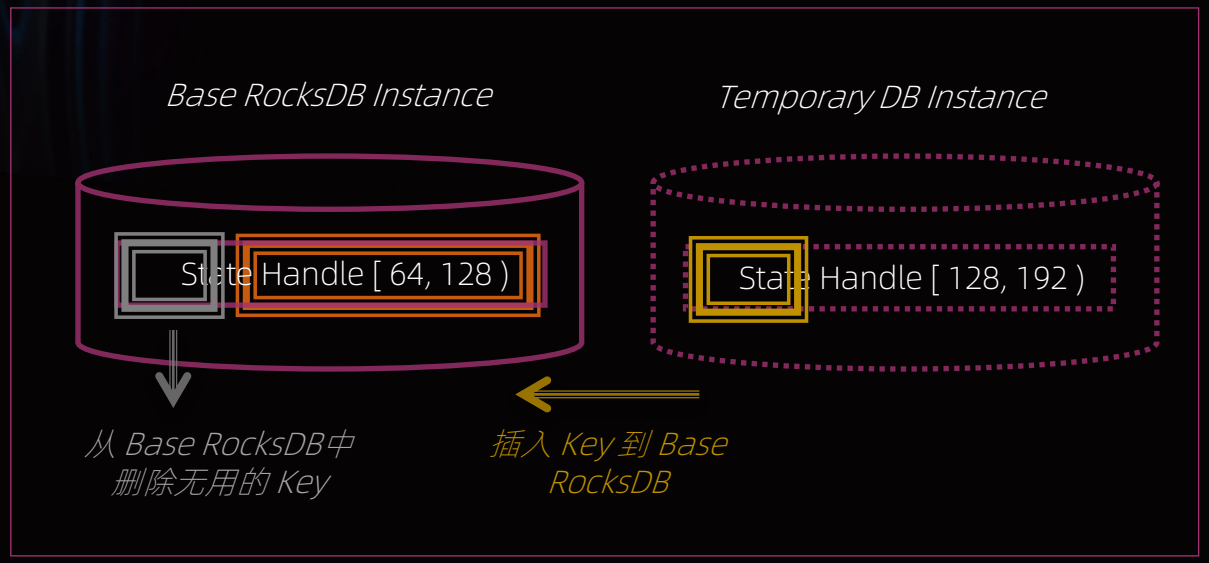


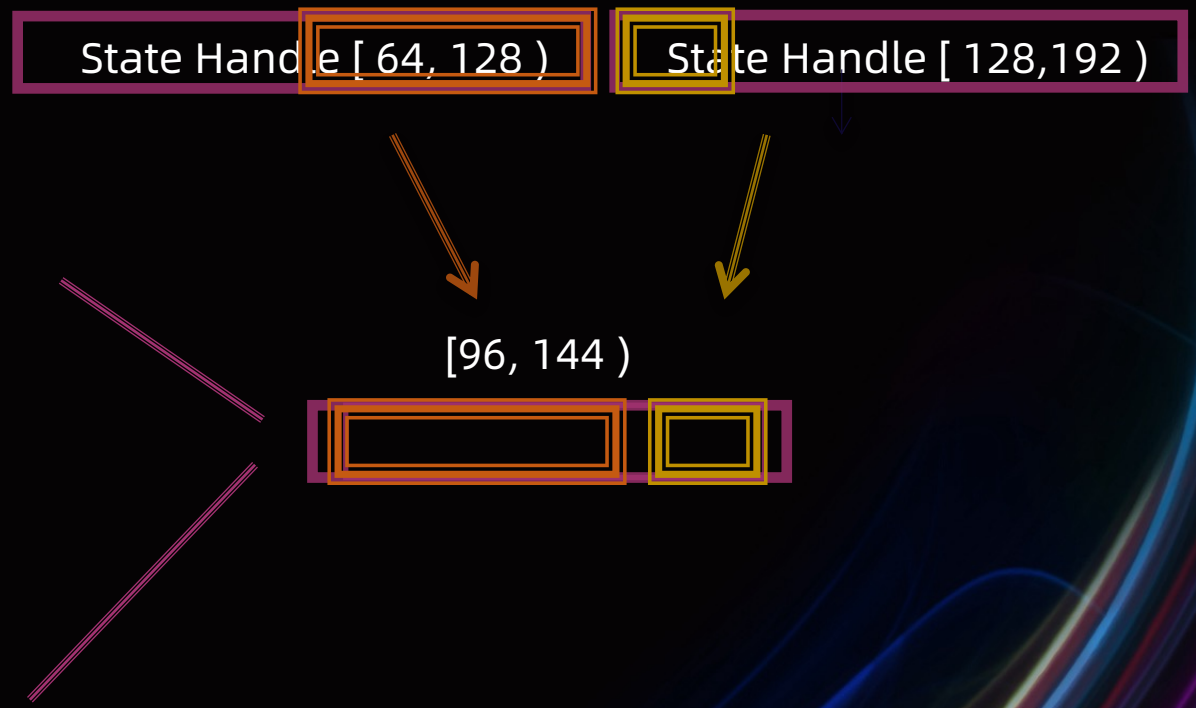
本地状态重建

1. 下载 State Handle 文件

2. 重建初始 RocksDB 实例，并删除对实例无用的 Key

3. 将临时 RocksDB 实例中的 Key 插入到重建的 RocksDB 中





优化本地状态重建 - Flink 1.16

1. 下载 State Handle 文件

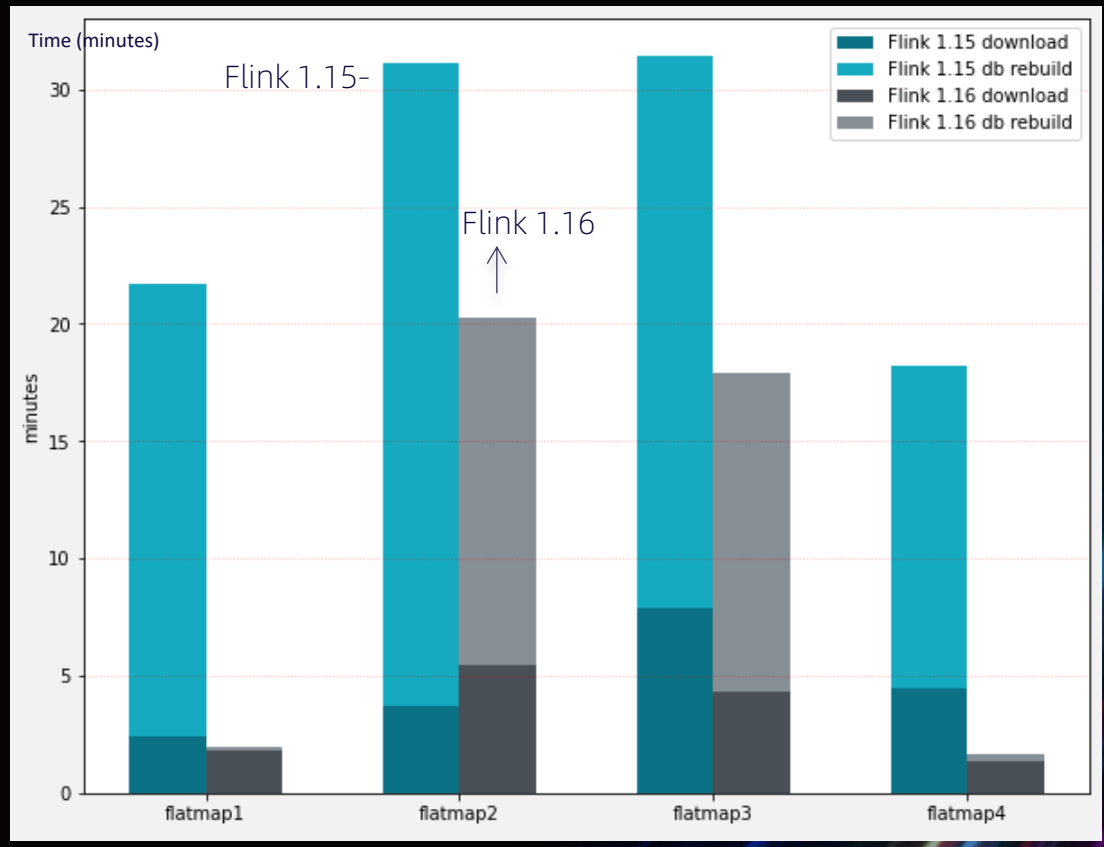
2. 重建初始 RocksDB 实例，并删除对实例无用的 Key

3. 将临时 RocksDB 实例中的 Key 插入到重建的 RocksDB 中

Flink 1.16 相关改进

1. 引入 `DeleteRange`, 整个删除 Key 的操作 → O(1), 对正常读写的影响极小
2. 基于第1点，保证重建的初始 RocksDB 实例不为空
3. 引入标准的 Rescaling Micro Benchmark

RocksDB Rescaling Time Flink 1.15- vs Flink 1.16



Word Count, Total State Size = 122 GB , Parallelism 3 → 4

扩容速度 2 - 10 倍提升

优化本地状态重建 - 企业级特性

1. 下载 State Handle 文件

2. 重建初始 RocksDB 实例，并删除对实例无用的 Key

3. 将临时 RocksDB 实例中的 Key 插入到重建的 RocksDB 中

阿里云实时计算企业级特性

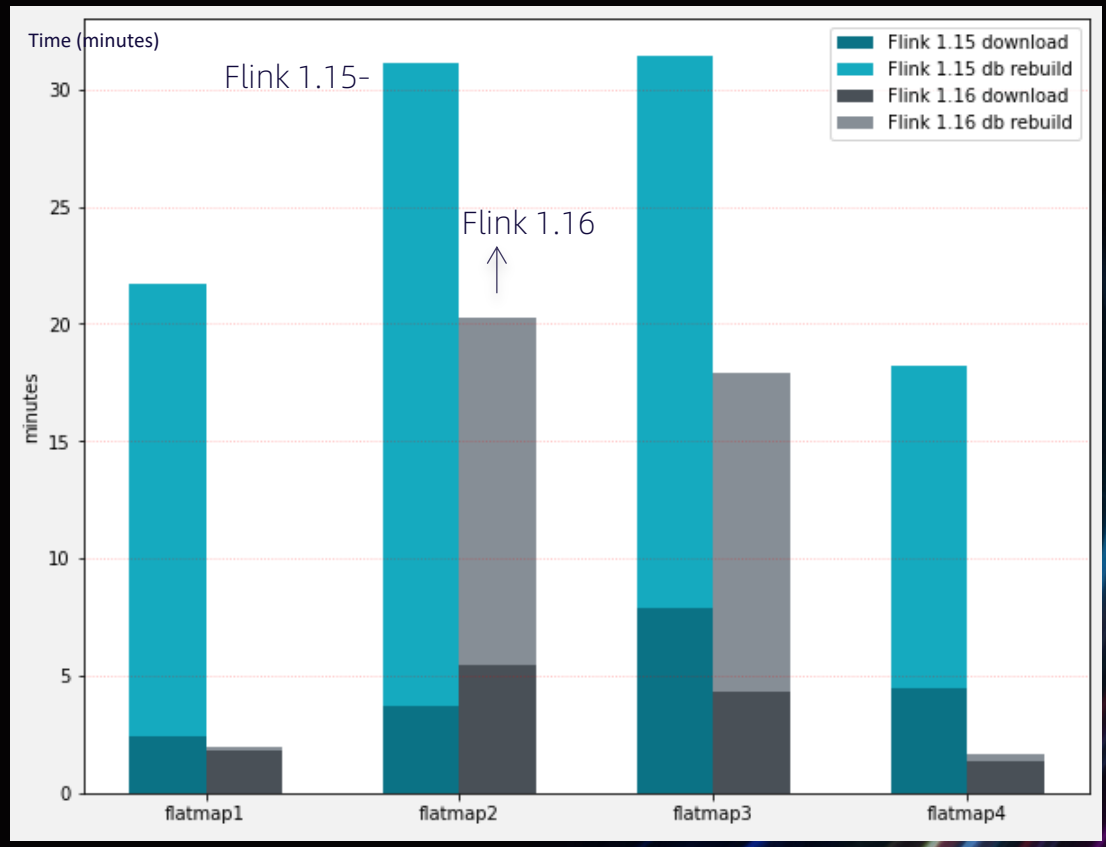
1. 只下载部分 State Handle 文件
2. 文件粒度直接合并，避免临时 DB 实例的创建

Flink 1.16 (ms/op)	阿里云实时计算
19366.421 ± 367.137	2626.339 ± 390.495

Flink Rescaling Micro Benchmark

缩容速度 7 倍提升

RocksDB Rescaling Time Flink 1.15- vs Flink 1.16



Word Count, Total State Size = 122 GB , Parallelism 3 → 4

扩容速度 2 - 10 倍提升

云原生分层状态存储架构

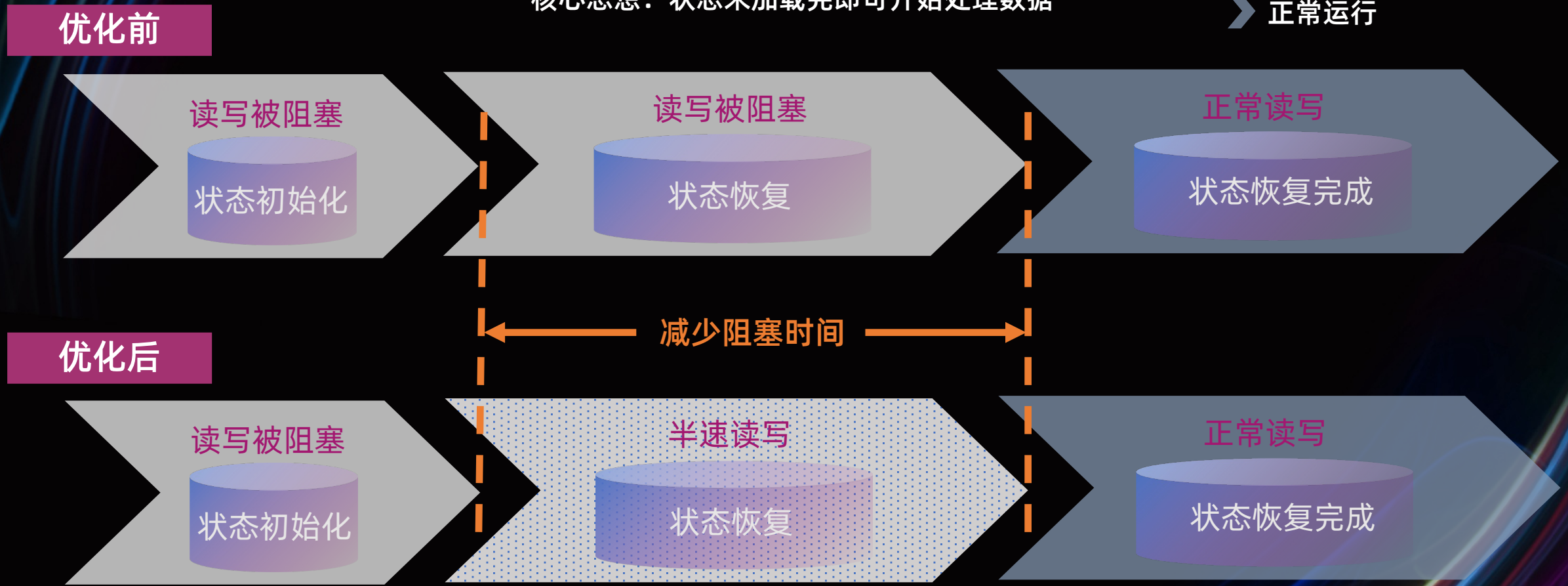
- 解决容器化部署本地磁盘大小受限的问题
- 解决小状态需要额外落盘的问题
- 解决外置状态成本高，数据一致性无法保障的问题
- 解决大状态访问速度慢的问题

可配置的状态加载策略

阿里云实时计算企业级特性

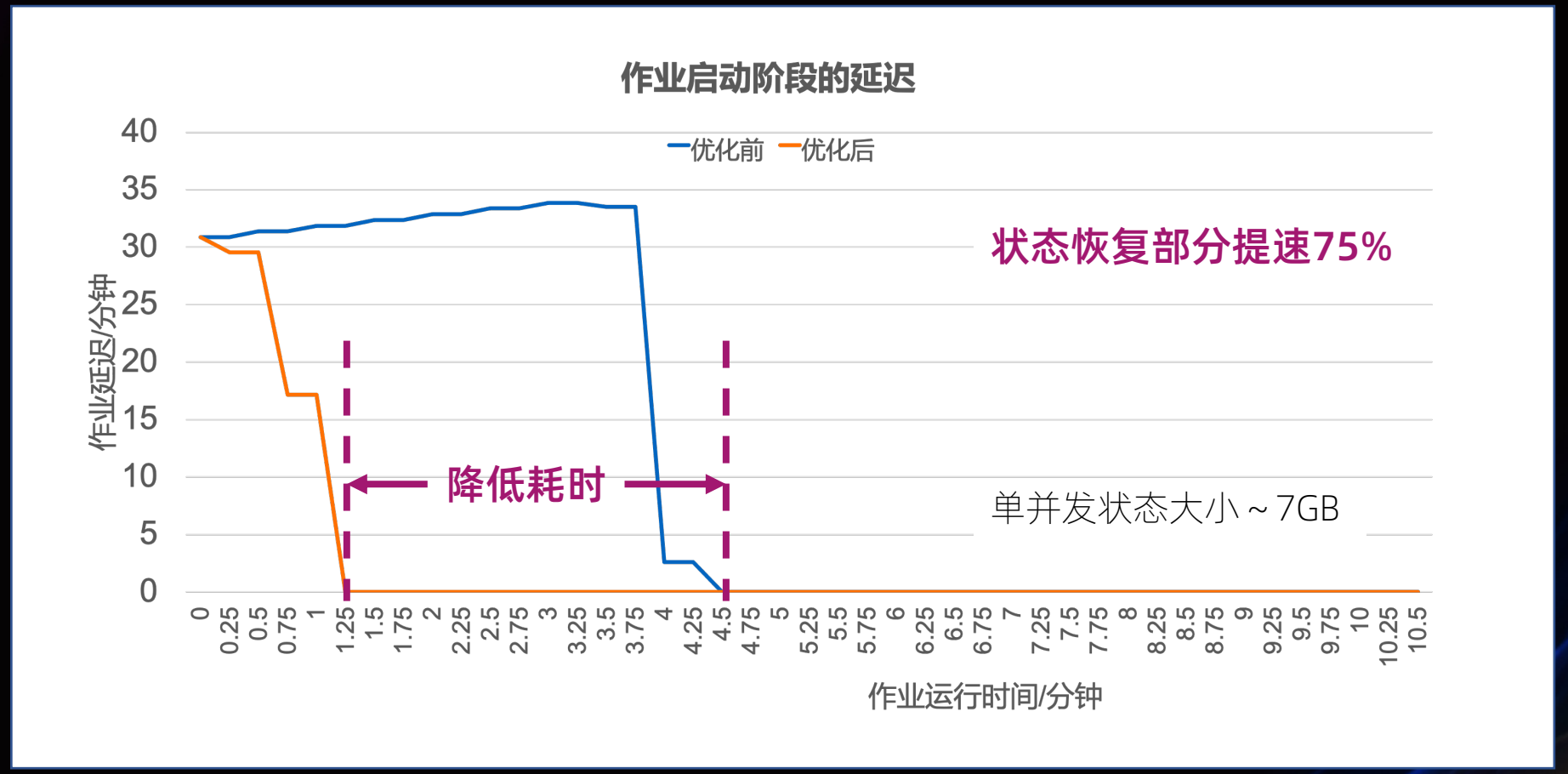
核心思想：状态未加载完即可开始处理数据

- 完全阻塞
- 半运行
- 正常运行



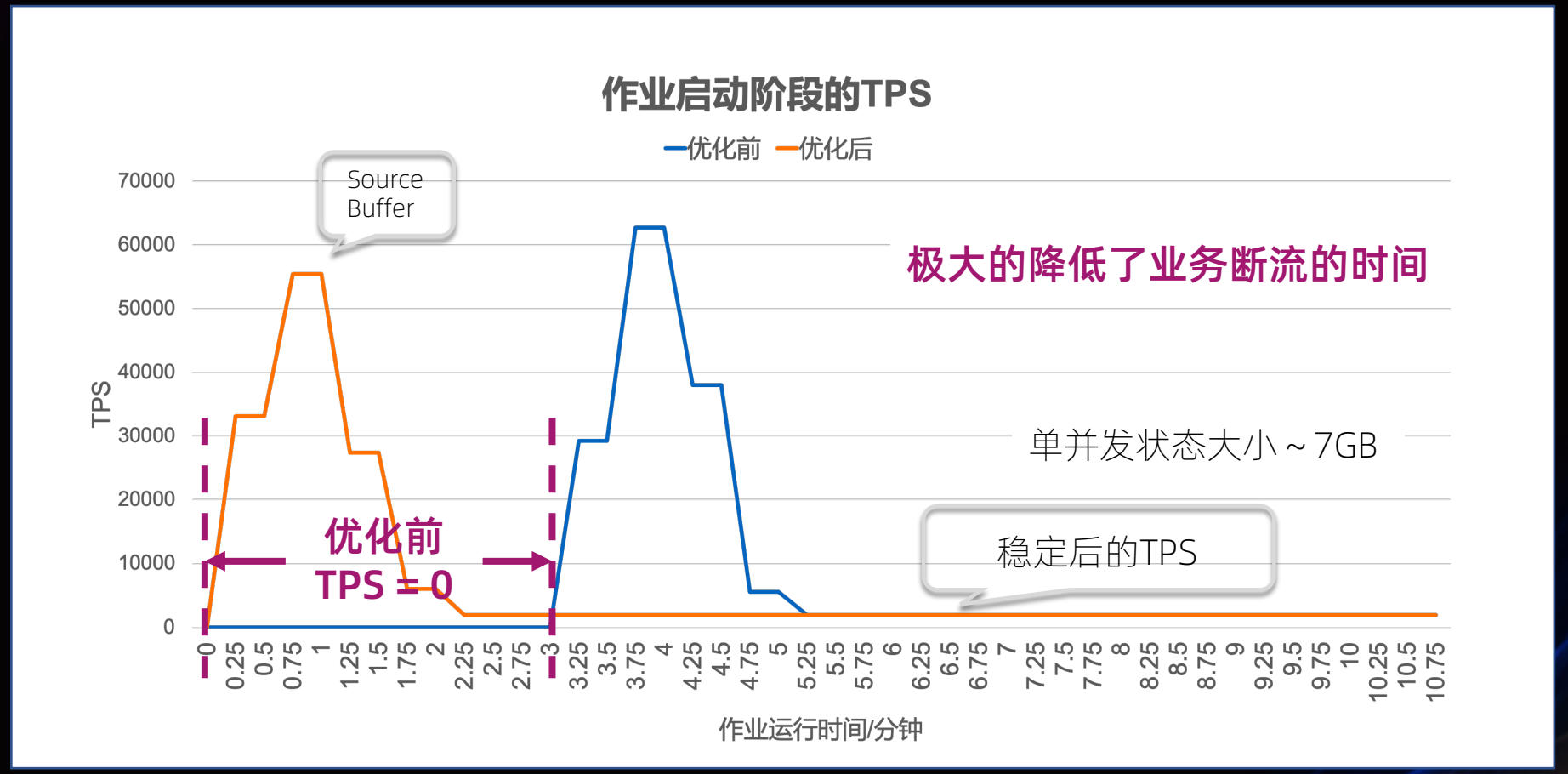
可配置的状态加载策略

测试结果来自阿里云实时计算平台服务，6.x 版本



可配置的状态加载策略

测试结果来自阿里云实时计算平台服务，6.x 版本



作业热更新

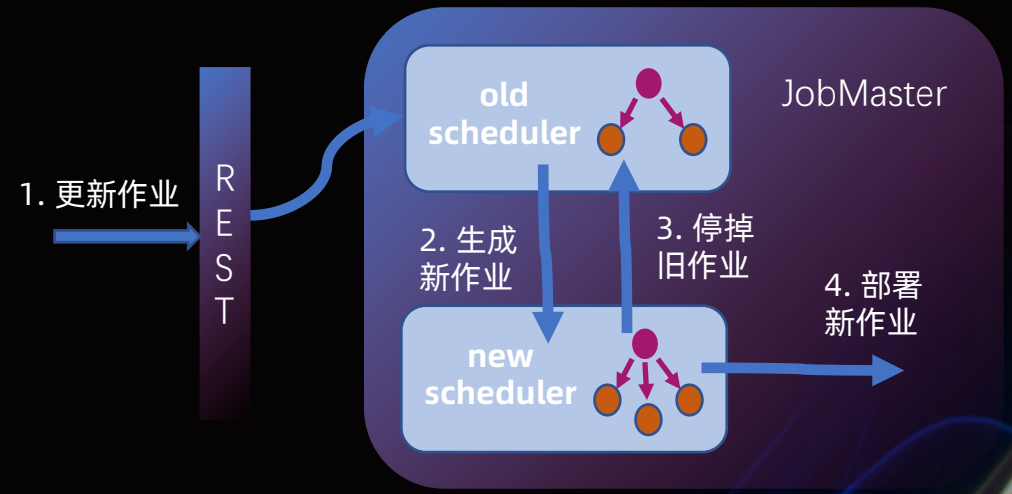
阿里云实时计算企业级特性

核心思想：简化作业重新调度的步骤

旧的流程

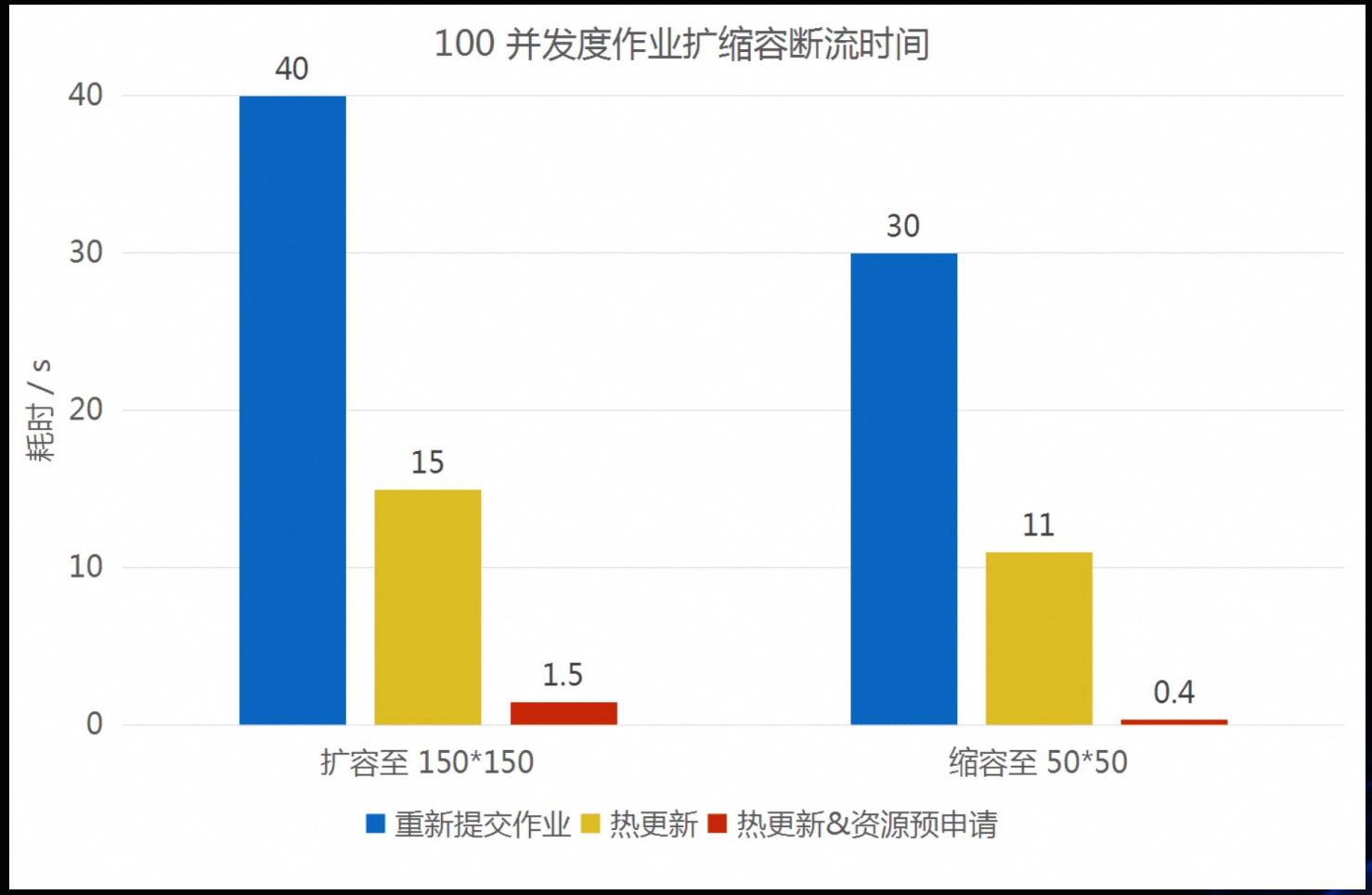
1. 停止和清理老作业	
2. 启动新作业	2.1 编译新作业
	2.2 申请、启动和初始化 Master 节点
	2.3 初始化作业
	2.4 申请、启动和初始化 Worker 节点
	2.5 调度部署任务
	2.6 初始化任务(下载 & 恢复 state)

热更新流程

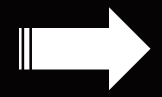


作业热更新

测试结果来自阿里云实时计算平台服务，6.x 版本



阿里云实时计算企业级特性

延迟状态加载策略 + 作业热更新 

扩缩容 Rescaling 无断流

→ 优化快照生成

升级分布式快照架构

快速稳定的 Checkpoints

→ 优化作业恢复和扩缩容

优化本地状态重建

云原生分层状态存储架构升级

简化重新调度的步骤

→ 优化快照管理

明确快照生命周期管理

增量 Native Savepoint

Savepoints Vs. Checkpoints

Savepoints

所属权归属用户

自包含的，不和 Flink 作业强绑定

不同 Flink 作业可以从同一个 Savepoint 启动

Checkpoints

所属权归属 Flink 引擎

非自包含，和生成 CP 的作业强绑定

Flink 引擎层按需清理 CP 文件

Savepoints Vs. Checkpoints

Savepoints

所属权归属用户

自包含的，不和 Flink 作业强绑定

不同 Flink 作业可以从同一个 Savepoint 启动

非常慢，状态稍大即不可用

Checkpoints

所属权归属 Flink 引擎

非自包含，和生成 CP 的作业强绑定

Flink 引擎层按需清理 CP 文件

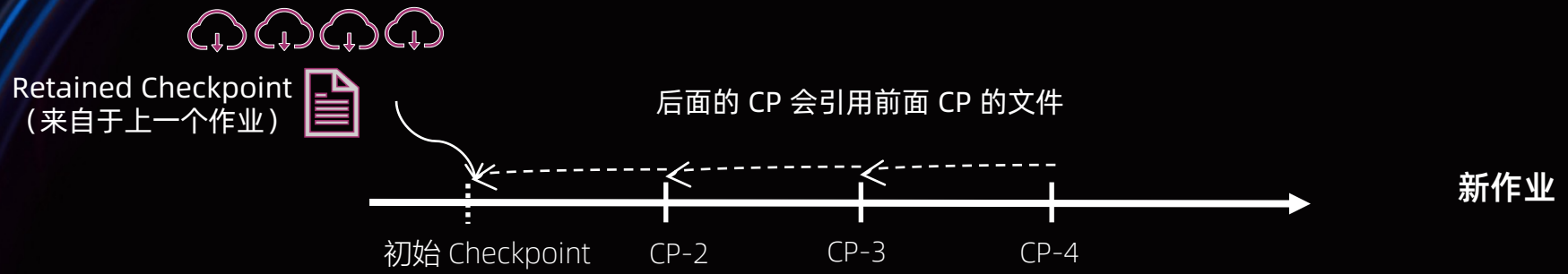
很快：增量 & 原生格式

Retained Checkpoints

Savepoints 和 Checkpoints 的混合体

- 用户负责删除 Retained Checkpoints
- 用户无法安全删除 Retained Checkpoints

Savepoints Vs. Checkpoints

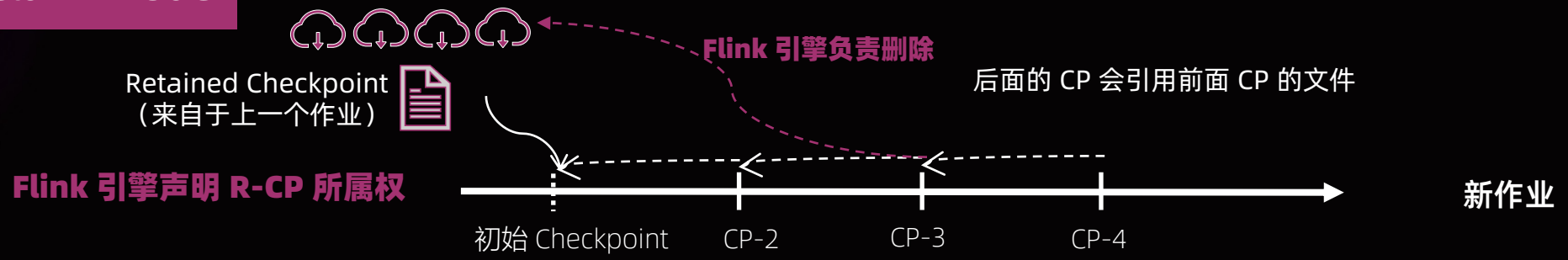


- 用户负责删除 Retained Checkpoints
- 用户无法安全删除 Retained Checkpoints

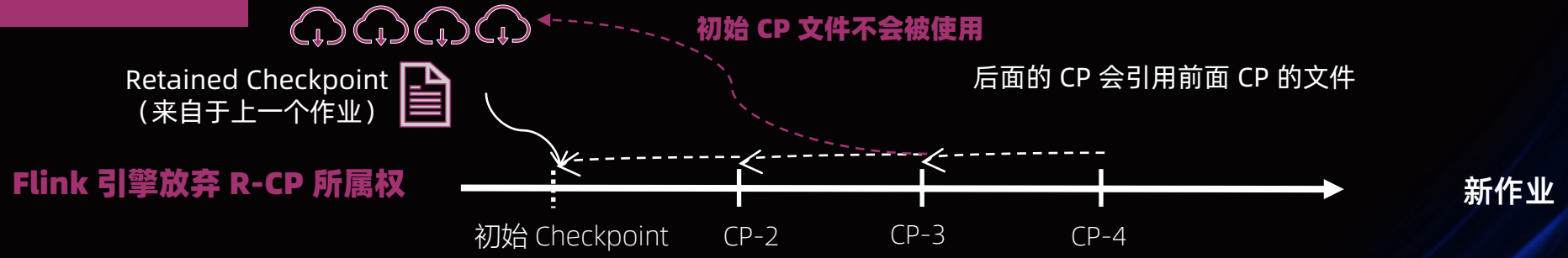
恢复模式（Restore Mode）：Claim Vs. No-Claim

Savepoints Vs. Checkpoints

Claim Mode



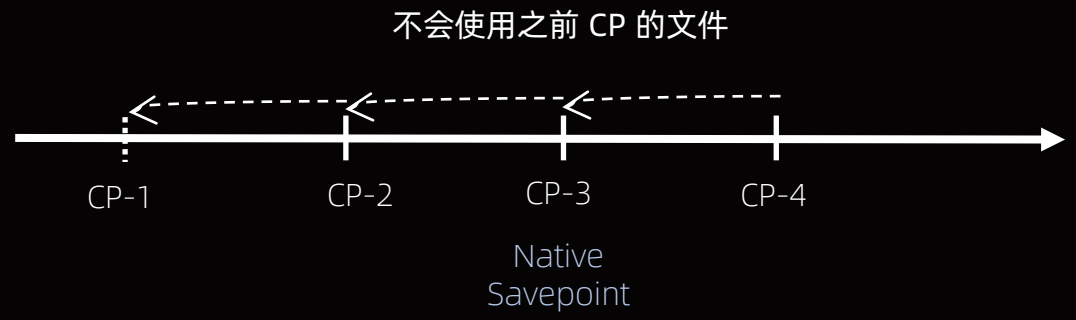
No-Claim Mode



Incremental Native Savepoints

	SP 时间	恢复时间	SP 增量数据	SP 总大小
< Flink 1.15	> 10 mins	950s	5.5GB	5.5GB
Flink 1.15	150s	160s	2.5GB	2.5GB
阿里云实时计算	5s	160s	100MB	2.5GB

Word Count　　Checkpoint Interval : 3 min　　Checkpoint Timeout : 10 mins



Flink容错恢复 2022 小结

- → 分布式快照架构升级，稳定快速的 Checkpoint
  - → Unaligned Checkpoint 自动切换生产可用
  - → 通用增量 Checkpoint 生产可用
- → 分层状态存储初步探索
  - → 扩缩容速度 2 - 10x 提升
- → 阿里云实时计算企业级特性
  - → 延迟状态加载策略 + 作业热更新 => 扩缩容无断流
- → 引入增量 Native Savepoint，全面提升 Savepoint 可用性

FLINK FORWARD # ASIA 2022 ONLINE | 实时即未来

谢谢大家