Flink中的两类新型状态存储
Introduction of Two New StateBackends

公司：阿里巴巴

职位：高级技术专家

演讲者：李钰（绝顶）

Alibaba Senior Expert

Apache HBase PMC

Yu Li (liyu@apache.org)

公司：阿里巴巴

职位：高级开发工程师

演讲者：唐云（茶干）

Alibaba Senior Software Engineer

Apache Flink Contributor

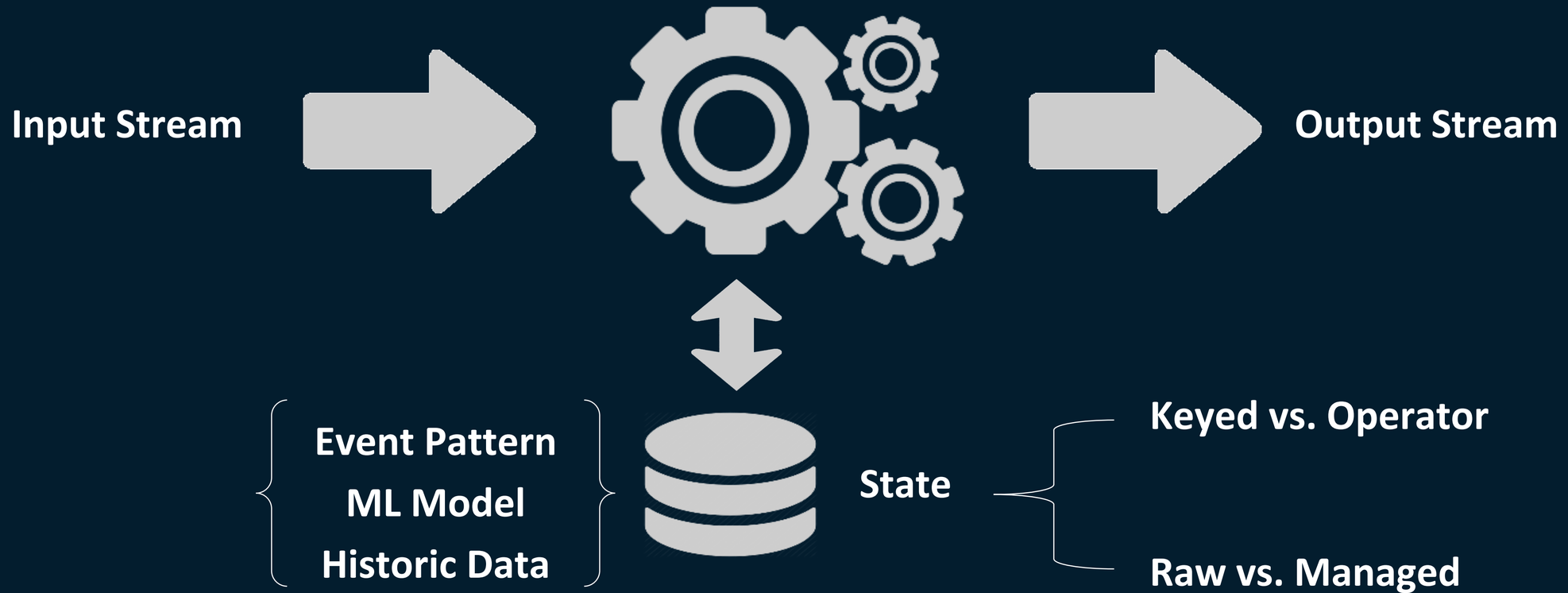
Yun Tang (chagan.ty@alibaba-inc.com)

FLINK FORWARD

大纲
Outline

- Flink中的状态数据及其存储回顾
  Review of Flink State and StateBackend
- 一种永不FullGC的内存状态存储解决方案
  A HeapKeyedStateBackend that Never FullGC
- 一种计算存储分离的状态存储解决方案
  Decouple Storage and Compute of State Management in Stream Computing

Flink中的状态数据
Review of Flink State



Flink的状态存储
Review of Flink State Backends

HeapKeyedStateBackend

- State lives as Java objects on the heap
- Data is de/serialized only during state snapshot and restore

- Highest Performance
- Memory overhead of representation
- State size limited by available heap memory
- Affected by GC overheads
- Currently no incremental checkpoints

RocksDBKeyedStateBackend

- State lives as serialized bytes in offheap memory and on local disk
- Data is de/serialized on every read and update

- Lower Performance (~order of magnitude)
- Relative low overhead of representation
- State size limited by available local disk space
- Not affected by GC overheads
- Naturally supports incremental checkpoints

字体

FLINK FORWARD

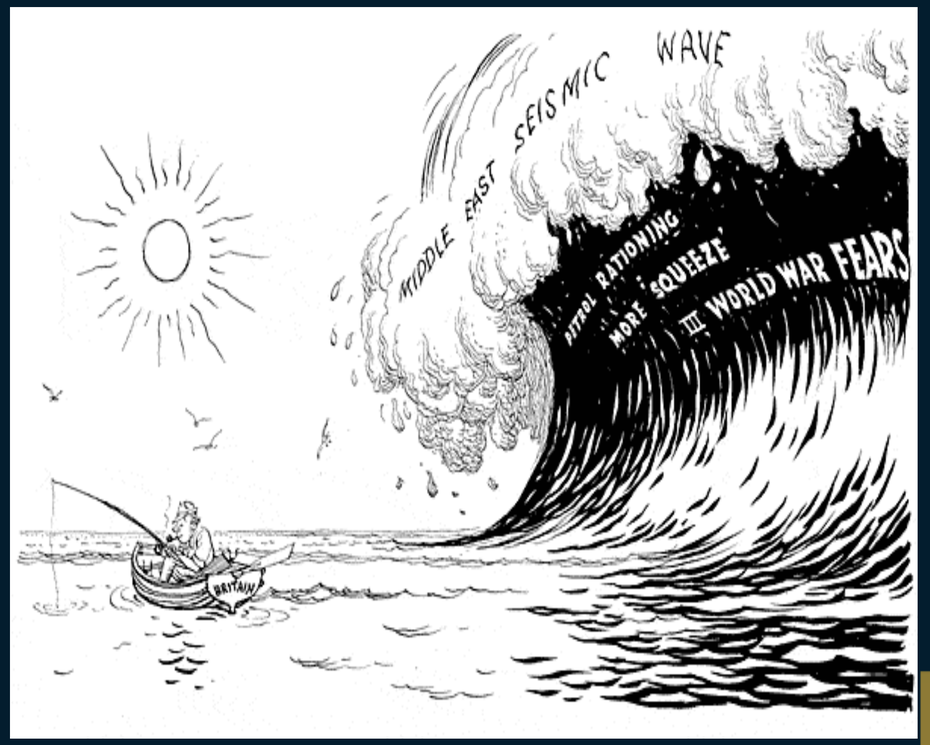
永不FullGC的状态存储解决方案

A HEAP KEYED STATEBACKEND THAT NEVER FULLGC

选择状态存储时尴尬的纠结
The Hesitation When Choosing Backends

为什么杀鸡却用牛刀?

Why using sledge-hammer to crack a nut?



99%的设计用于应对1%的危险

99% consideration for the 1% case

核心技术：内存状态监控
Key Tech: Heap Status Monitor

内存使用情况监控
Heap Status Monitor

- 对每个KG做内存统计（参考Luncene RamUsageEstimator）
  Do heap accounting for each KG (similar to Luncene's RamUsageEstimator)
- 定期通过Java MXBeans获取堆内存使用情况，GC次数和停机时间
  Get heap used, gc count and pause time with fixed interval
- 检查是否有old gc发生并且内存使用超过配置阀值
  Check whether heap used exceeds threshold after old gc
- 检查是否有gc停机时间超过配置阀值
  Check whether gc pause time exceeds threshold

核心技术：数据落盘和加载策略
Key Tech: Data Spill and Load Policy

数据落盘和加载策略
Data Spill and Load Policy

- 除内存统计外，对每个KG做访问频次统计
  Record and compute request rate for each KG
- 采取加权平均的方式计算不同KG的权重
  Compute the weighted average on the normalized KG size and request rate
- 落盘选择策略：优先落盘占用内存最多，请求最少的KG
  Spill policy: choose KG with more heap occupancy and lower request rate
- 加载选择策略：优先加载请求最频繁，占用内存最少的KG
  Load policy: choose KG with higher request rate and less heap occupancy

核心技术：落盘数据存储结构及增量checkpoint支持
Key Tech: Data Structure on Disk and Incremental Checkpoint

磁盘数据存储结构
Data Structure on Disk

- 使用一种紧凑的跳表结构
  A compacted SkipList structure for data on disk
- 通过delta-chain的方式支持写时复制，避免直接更新
  Use delta-chain instead of update in place to support copy-on-write
- 参考论文：Nitro from VLDB 2016
  Referring to the Nitro paper from VLDB 2016

增量checkpoint支持
Incremental Checkpoint

- 增加snapshot版本序号
  Add and record snapshot version
- 在数据中增加版本信息，使用delta-chain的方式保留snapshot中的数据版本
  Save version in value and reserve data in snapshot through delta-chain
- 写入和快照时对旧版本进行检查和清理
  Check data in delta chain for cleanup during put or snapshot

性能对比和开源计划
Performance and Upstreaming Plan

性能对比 Performance

- Word count job, state size = 566MB (Tested with PCIe-SSD)

	QPS (K/s)	Note
Heap (all in memory)	400	-Xmx=10GB
RocksDB (all in memory)	100	Cache=10GB
Heap (memory + disk)	160	-Xmx=3GB (spill ratio=57%)
RocksDB (memory + disk)	100	Cache=3GB

开源计划 Upstreaming Plan

- 局限性：checkpoint期间CPU消耗较高
  More CPU cost during checkpoint because of scan
- 将在近期贡献回社区
  Will start the upstreaming soon

字体

FLINK FORWARD

一种计算存储分离的状态存储解决方案

A SLOUTION TO DECOUPLE STORAGE AND COMPUTE OF STATE MANAGERMENT IN STREAM COMPUTING

FLINK FORWARD

Background

存储计算耦合的架构

The Compute-storage Coupled Arch

利用数据本地性（data locality）的存储与计算耦合的高容错架构（计算向数据迁移），是考虑到数据的网络传输开销在当时的环境下，远大于本地数据的磁盘读取开销的廉价高效的解决方案

Data locality was important when network is bottleneck, but things have already changed

Year	2003	2018
Network	100Mbps	10Gbps+

Background

存储计算分离的趋势
Trend to Decouple Compute and Storage

Alibaba Cloud

云计算时代，存储计算耦合已经不再适应弹性计算的需求
Decoupling storage resource is necessary for elastic computing

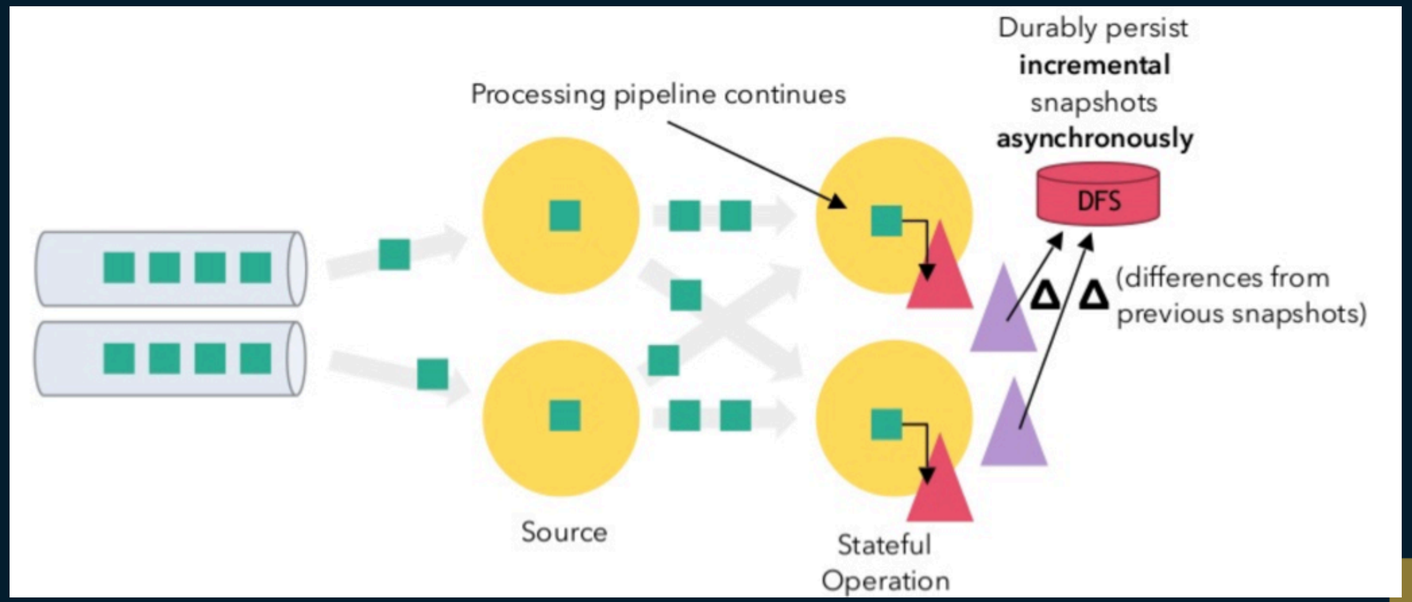
- 随着业务发展，需要不同的存储空间和计算能力配比时，机器的选型会比较复杂和繁琐
  Decoupling makes it easier for hardware budget
- 存储空间或计算资源不足时，只能同时对两者进行扩容，导致扩容的经济效率比较低
  Decoupling makes better resource utilization
- 因为数据耦合，不能随意关闭闲置的计算集群
  Decoupling makes better power saving

Background

目前Flink的流计算State管理机制

Current State Management In Apache Flink

Take RocksDB StateBackend for example

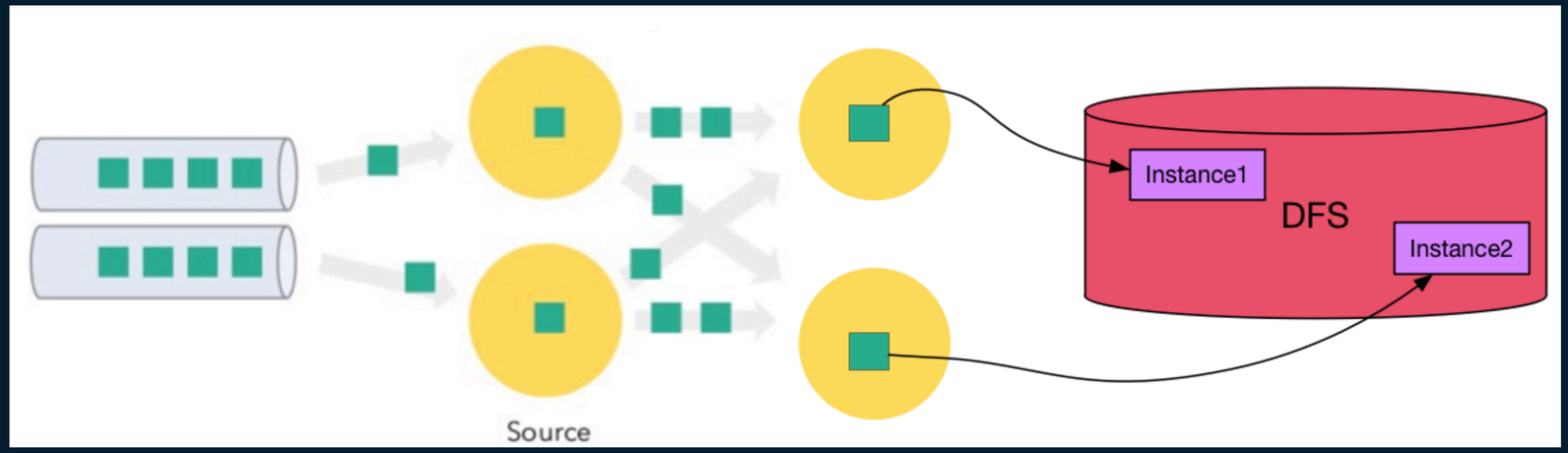


Overview

计算存储分离的State管理机制

Storage-Compute Decoupled Arch

Take Niagara* StateBackend for example

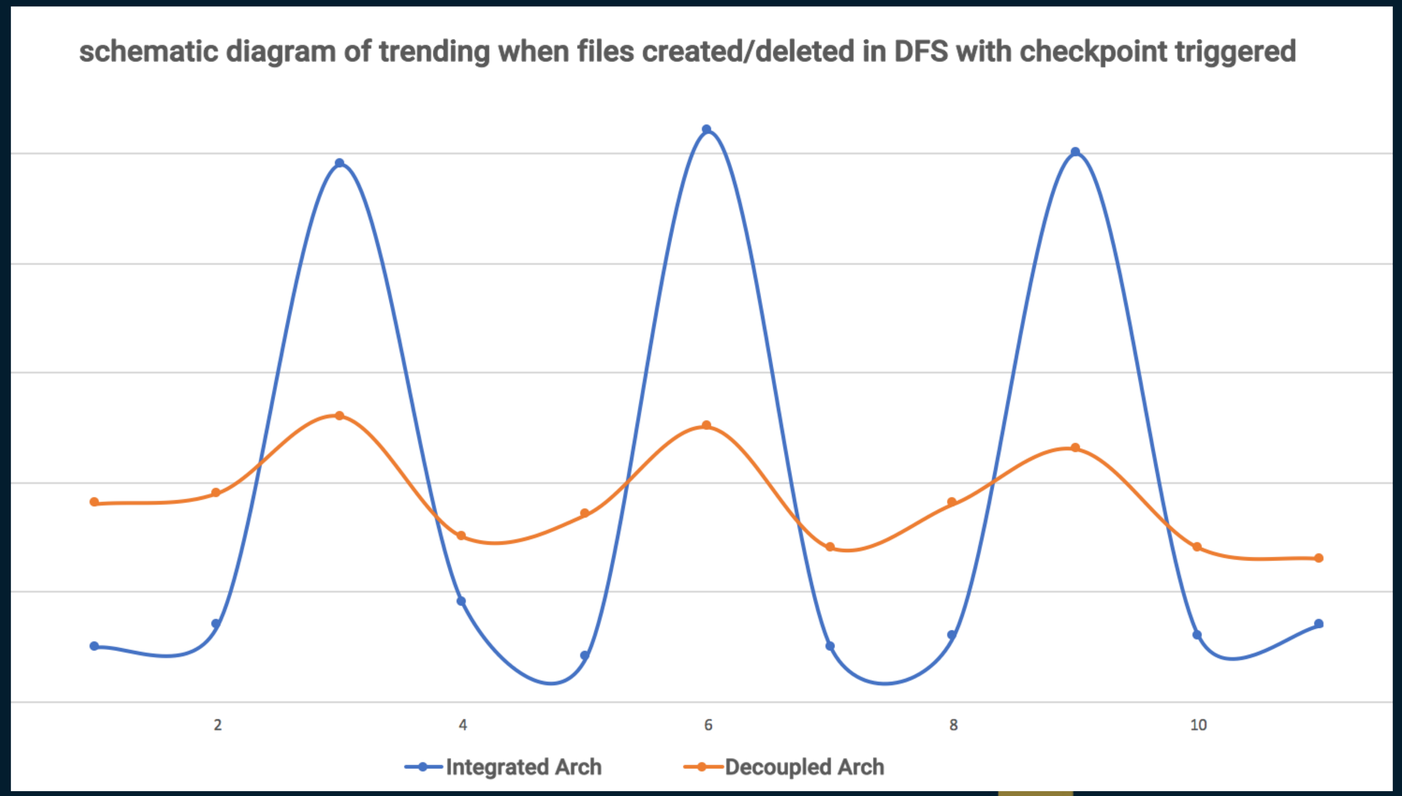


- Niagara is a self-developed, advanced KV store used in Alibaba

Implementation

存储计算分离的优点
Advantages

- 对分布式文件系统而言更平稳的秒级checkpoint创建和恢复
  Checkpoint created and restored more smoothly in seconds for DFS

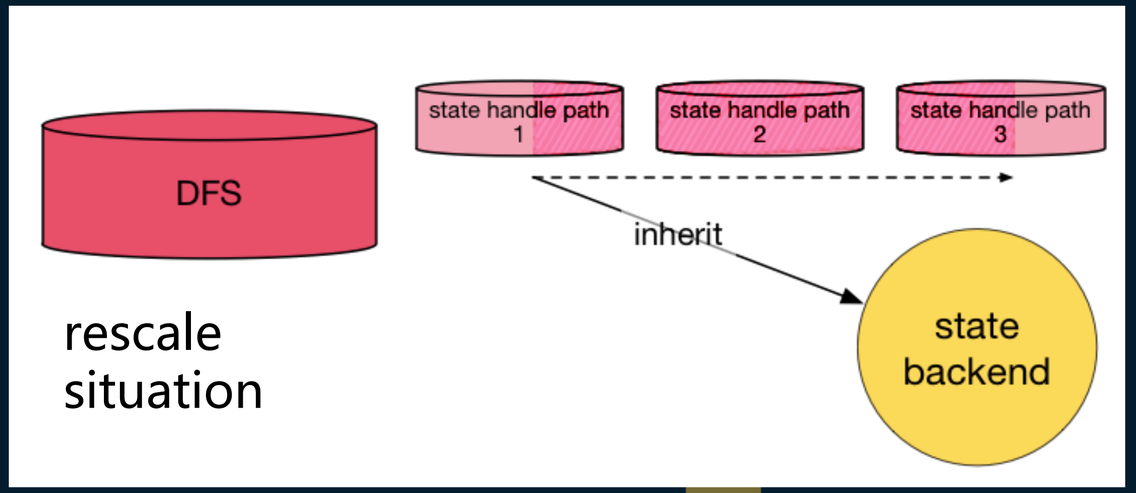


Implementation

存储计算分离的优点
Advantages

- 对分布式文件系统而言更平稳的秒级checkpoint创建和恢复
  Checkpoint created and restored more smoothly in seconds for DFS

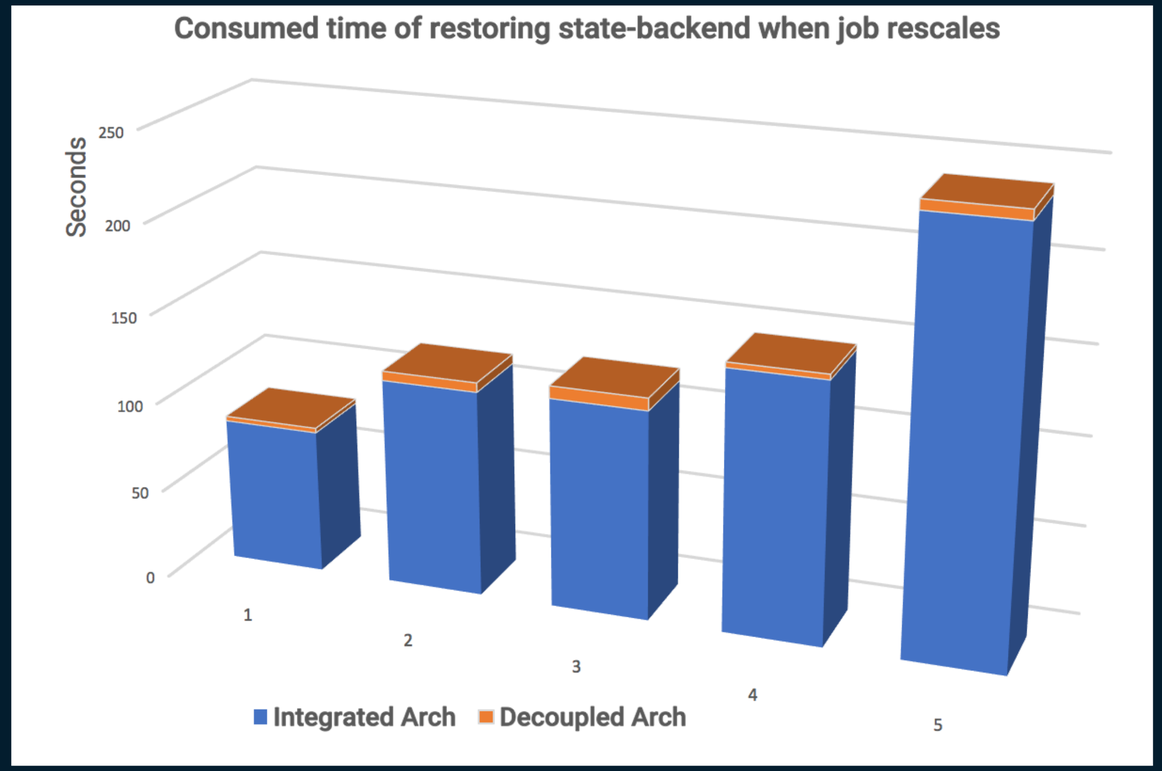
可以做到秒级的作业改并发，避免了目前Flink checkpoint架构下改并发时，每次本地大量的IO操作和缓慢的本地DB扫描更改数据的操作。
Restore backend in seconds when rescaling, avoid long consumed time of IO and DB scan in current Flink arch



Implementation

存储计算分离的优点
Advantages

- 对分布式文件系统而言更平稳的秒级checkpoint创建和恢复
  Checkpoint created and restored more smoothly in seconds for DFS



Implementation

存储计算分离的局限性
Constraints and Solution

- 计算存储分离面临着可能的写延迟问题

  Avoid write latency for decoupled storage arch

借助于Niagara在Pangu*上的前后端分离的异步IO架构，我们尽可能地降低了写延迟，数据直接写入Niagara内存中，后台异步再刷写到DFS上，尽量避免写延迟对作业造成的反压影响。

By means of async-arch of Niagara on Pangu*, data would be flushed to DFS asynchronously

* Pangu is a self-developed, advanced distributed file system used in Alibaba

Implementation

存储计算分离的局限性
Constraints and Solution

- 计算存储分离面临着可能的读延迟问题（痛点）
  Avoid read latency for decoupled storage arch

借助于DADI* 的SSD文件二级缓存的方案，我们尽可能降低了读延迟，在压测场景下获得了不逊于传统架构的性能表现。
By means of DADI*, a local fast second-level cache for LSM-like DB, we could guarantee the read latency nearly to traditional arch.

* **D**ata **A**cceleration for **D**isaggregated **I**nfrastructure，a high performance local second-level cache for LSM-like DB used in Alibaba.

Implementation

存储计算分离的局限性

Constraints and Solution

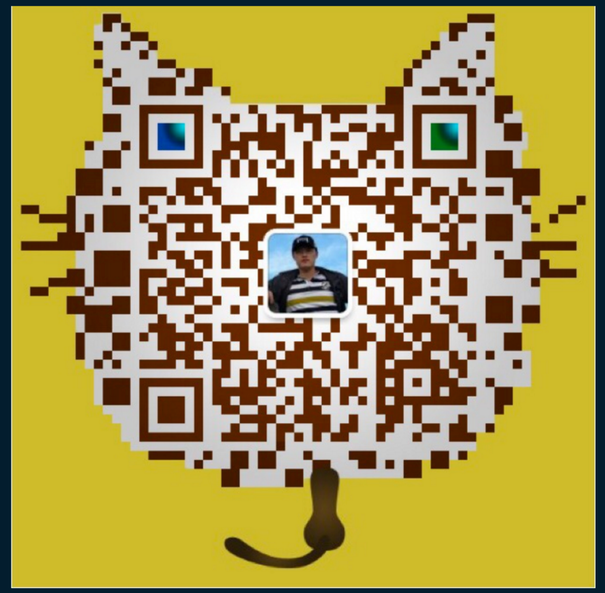
- 及时清理作业failover场景下在DFS上产生的无用目录与文件
  Clean useless directories/files on DFS eagerly

在JobManager中引入冗余无用目录、文件的清理机制，避免在DFS上的任何目录残留

Introduce checkpoint trash cleaner mechanism in JobManager

FLINK FORWARD

We are recruiting



微信
WeChat

钉钉
DingDing

THANKS