火山引擎 | DataFun.

字节跳动全域数据集成演进历程

李畅 字节跳动大数据工程师

个人简介

- 16年加入字节跳动开发套件团队，从0到1设计、研发了面向字节各业务线的数据集成服务
- 专注大规模数据的分布式计算和传输领域， 提供高效、可靠的全域数据集成解决方案

目录

01 数据集成背景

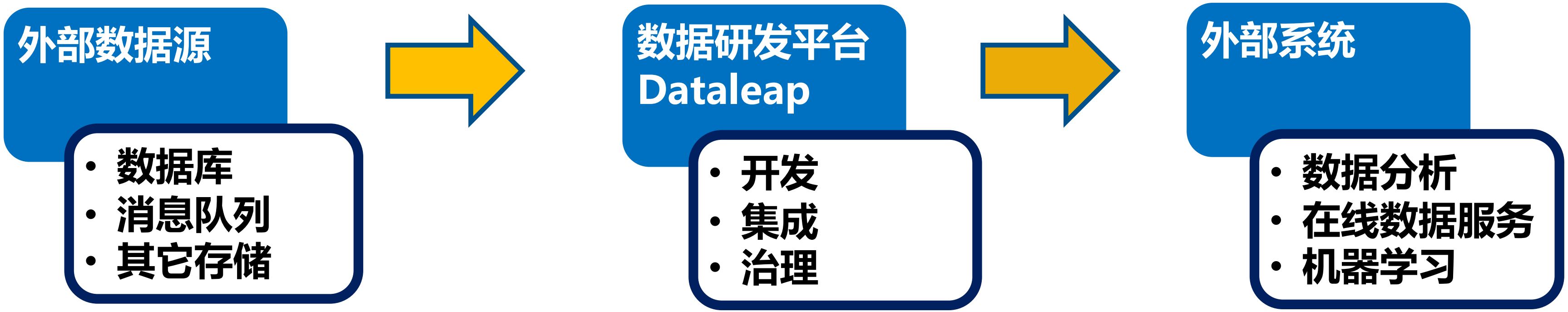
02 数据集成演进历程

03 通用能力改造

04 未来展望

01 数据集成背景

数据集成背景介绍



外部数据源

- 数据库
- 消息队列
- 其它存储

数据研发平台 Dataleap

- 开发
- 集成
- 治理

外部系统

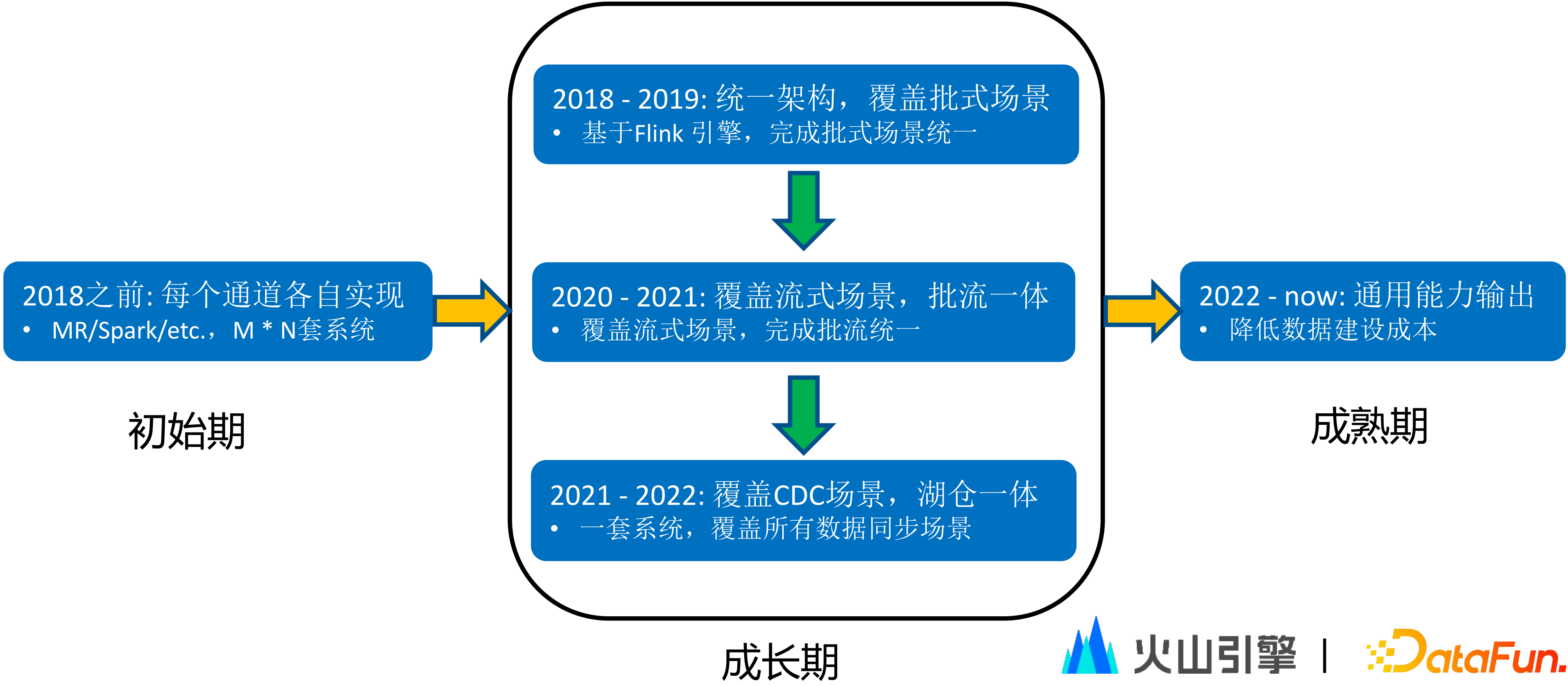
- 数据分析
- 在线数据服务
- 机器学习

- 数据集成是数据中台建设的基础，主要解决异构数据源间数据传输、加工和处理
- Dataleap是字节跳动自研的一站式数据中台套件，并服务字节内部各业务线数据建设场景

02

全域数据集成演进历程

全域数据集成演进历程



2018之前：每个通道各自实现
- MR/Spark/etc.， M * N套系统

初始期

2018 - 2019：统一架构，覆盖批式场景
- 基于Flink 引擎，完成批式场景统一

2020 - 2021：覆盖流式场景，批流一体
- 覆盖流式场景，完成批流统一

2021 - 2022：覆盖CDC场景，湖仓一体
- 一套系统，覆盖所有数据同步场景

成长期

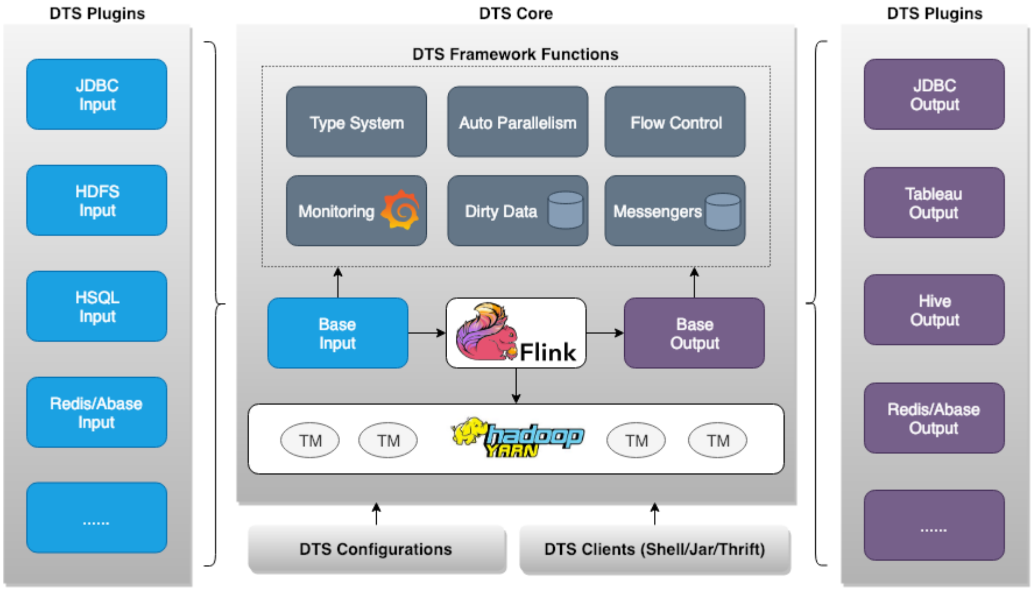
2022 - now：通用能力输出
- 降低数据建设成本

成熟期

基于Flink的异构数据源传输架构

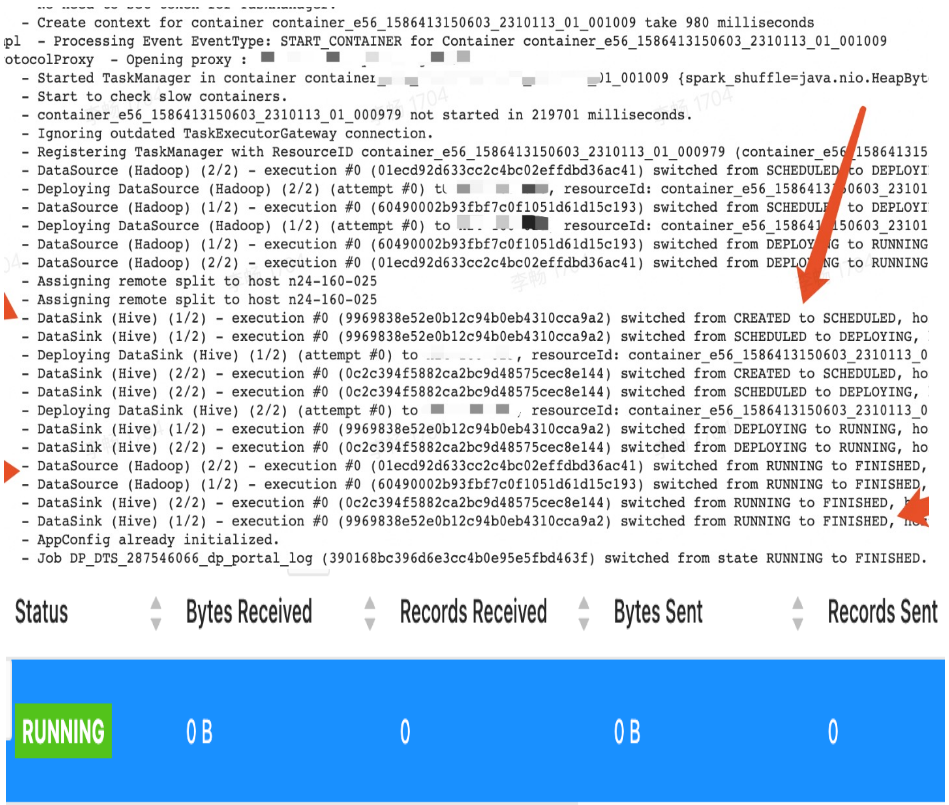
初始架构

- 基于Flink 1.5 DataSet API，只覆盖批式场景
- 提供抽象的BaseInput和BaseOutput，实现数据源种类线性扩展
- 框架层提供统一基础服务，包括类型系统、自动并发度、脏数据检测、流控等
- 支持Yarn部署，资源管理比较弹性



Flink Batch任务进度查询

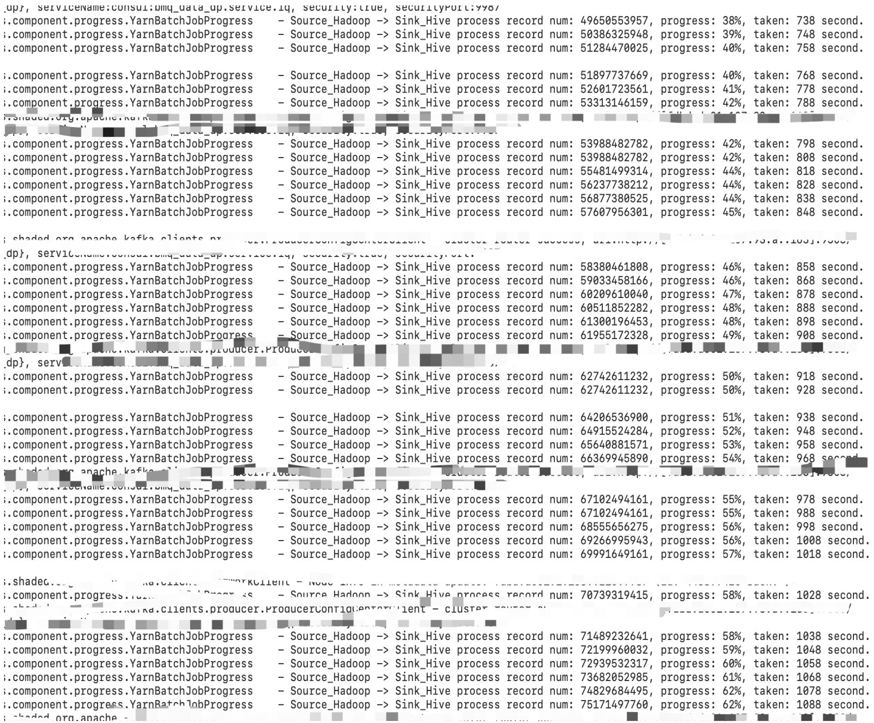
原始



```
 No need to set token for TaskManager.
 - Create context for container container_e56_1586413150603_2310113_01_001009 take 980 milliseconds
pl  - Processing Event EventType: START_CONTAINER for Container container_e56_1586413150603_2310113_01_001009
otocolProxy  - Opening proxy : [redacted]
 - Started TaskManager in container container[redacted]01_001009 {spark_shuffle=java.nio.HeapByt
 - Start to check slow containers.
 - container_e56_1586413150603_2310113_01_000979 not started in 219701 milliseconds.
 - Ignoring outdated TaskExecutorGateway connection.
 - Registering TaskManager with ResourceID container_e56_1586413150603_2310113_01_000979 (container_e56_158641315
 - DataSource (Hadoop) (2/2) - execution #0 (01ecd92d633cc2c4bc02effdbd36ac41) switched from SCHEDULED to DEPLOYI
 - Deploying DataSource (Hadoop) (2/2) (attempt #0) t[redacted], resourceId: container_e56_1586413150603_23101
 - DataSource (Hadoop) (1/2) - execution #0 (60490002b93fbf7c0f1051d61d15c193) switched from SCHEDULED to DEPLOYI
 - Deploying DataSource (Hadoop) (1/2) (attempt #0) to [redacted] resourceId: container_e56_1586413150603_23101
 - DataSource (Hadoop) (1/2) - execution #0 (60490002b93fbf7c0f1051d61d15c193) switched from DEPLOYING to RUNNING
 - DataSource (Hadoop) (2/2) - execution #0 (01ecd92d633cc2c4bc02effdbd36ac41) switched from DEPLOYING to RUNNING
 - Assigning remote split to host n24-160-025
 - Assigning remote split to host n24-160-025
 - DataSink (Hive) (1/2) - execution #0 (9969838e52e0b12c94b0eb4310cca9a2) switched from CREATED to SCHEDULED, ho
 - DataSink (Hive) (1/2) - execution #0 (9969838e52e0b12c94b0eb4310cca9a2) switched from SCHEDULED to DEPLOYING,
 - Deploying DataSink (Hive) (1/2) (attempt #0) to [redacted], resourceId: container_e56_1586413150603_2310113_0
 - DataSink (Hive) (2/2) - execution #0 (0c2c394f5882ca2bc9d48575cec8e144) switched from CREATED to SCHEDULED, ho
 - DataSink (Hive) (2/2) - execution #0 (0c2c394f5882ca2bc9d48575cec8e144) switched from SCHEDULED to DEPLOYING,
 - Deploying DataSink (Hive) (2/2) (attempt #0) to [redacted], resourceId: container_e56_1586413150603_2310113_0
 - DataSink (Hive) (1/2) - execution #0 (9969838e52e0b12c94b0eb4310cca9a2) switched from DEPLOYING to RUNNING, ho
 - DataSink (Hive) (2/2) - execution #0 (0c2c394f5882ca2bc9d48575cec8e144) switched from DEPLOYING to RUNNING, ho
 - DataSource (Hadoop) (2/2) - execution #0 (01ecd92d633cc2c4bc02effdbd36ac41) switched from RUNNING to FINISHED,
 - DataSource (Hadoop) (1/2) - execution #0 (60490002b93fbf7c0f1051d61d15c193) switched from RUNNING to FINISHED,
 - DataSink (Hive) (2/2) - execution #0 (0c2c394f5882ca2bc9d48575cec8e144) switched from RUNNING to FINISHED,
 - DataSink (Hive) (1/2) - execution #0 (9969838e52e0b12c94b0eb4310cca9a2) switched from RUNNING to FINISHED,
 - AppConfig already initialized.
 - Job DP_DTS_287546066_dp_portal_log (390168bc396d6e3cc4b0e95e5fbd463f) switched from state RUNNING to FINISHED.
```

Status	Bytes Received	Records Received	Bytes Sent	Records Sent
RUNNING	0 B	0	0 B	0

改进



```
_dp}, serviceName:consul:bmq_data_dp.service.lq, security:true, securityPort:9987
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 49650553957, progress: 38%, taken: 738 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 50386325948, progress: 39%, taken: 748 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 51284470025, progress: 40%, taken: 758 second.

;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 51897737669, progress: 40%, taken: 768 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 52601723561, progress: 41%, taken: 778 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 53313146159, progress: 42%, taken: 788 second.
[redacted]
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 53988482782, progress: 42%, taken: 798 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 53988482782, progress: 42%, taken: 808 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 55481499314, progress: 44%, taken: 818 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 56237738212, progress: 44%, taken: 828 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 56877380525, progress: 44%, taken: 838 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 57607956301, progress: 45%, taken: 848 second.

; shaded.org.apache.kafka.clients.pr[redacted]
_dp}, servi[redacted]
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 58380461808, progress: 46%, taken: 858 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 59033458166, progress: 46%, taken: 868 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 60209610040, progress: 47%, taken: 878 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 60511852282, progress: 48%, taken: 888 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 61300196453, progress: 48%, taken: 898 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 61955172328, progress: 49%, taken: 908 second.
[redacted]
_dp}, serv[redacted]
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 62742611232, progress: 50%, taken: 918 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 62742611232, progress: 50%, taken: 928 second.

;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 64206536900, progress: 51%, taken: 938 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 64915524284, progress: 52%, taken: 948 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 65640881571, progress: 53%, taken: 958 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 66369945890, progress: 54%, taken: 968 second.
[redacted]
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 67102494161, progress: 55%, taken: 978 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 67102494161, progress: 55%, taken: 988 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 68555656275, progress: 56%, taken: 998 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 69266995943, progress: 56%, taken: 1008 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 69991649161, progress: 57%, taken: 1018 second.

;.shaded.[redacted]
;.component.progres[redacted]    - Source_Hadoop -> Sink_Hive process record num: 70739319415, progress: 58%, taken: 1028 second.
[redacted]
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 71489232641, progress: 58%, taken: 1038 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 72199960032, progress: 59%, taken: 1048 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 72939532317, progress: 60%, taken: 1058 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 73682052985, progress: 61%, taken: 1068 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 74829684495, progress: 62%, taken: 1078 second.
;.component.progress.YarnBatchJobProgress    - Source_Hadoop -> Sink_Hive process record num: 75171497760, progress: 62%, taken: 1088 second.
;.shaded.org.apache [redacted]
```

Flink Batch任务进度查询

Flink Task执行过程

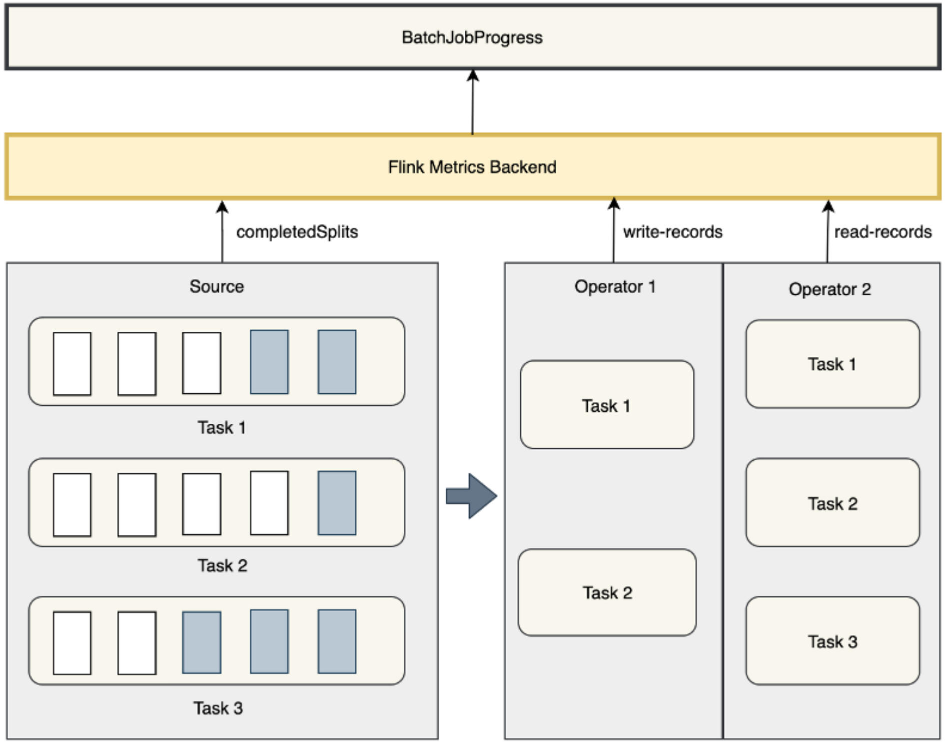
- Flink是以任务驱动，JM构建好Split，Task常驻，不断向JM请求新的Split
- 所有Split处理完Task才会退出

Source进度

- SourceProgress = CompletedSplits / TotalSplits

Operator进度

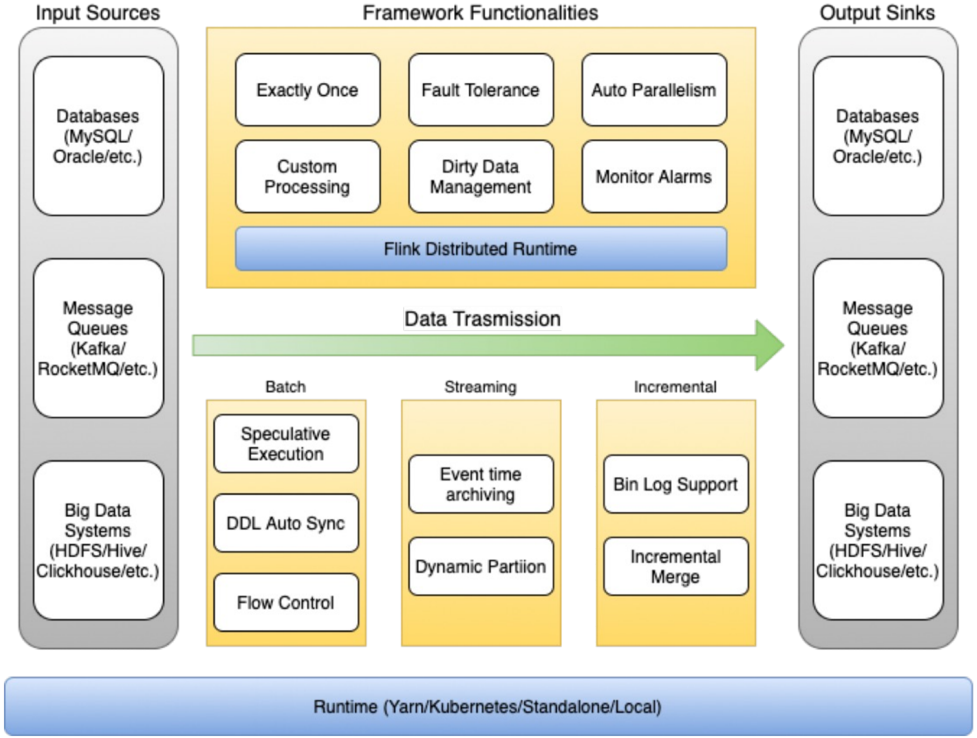
- CurrentProgress = Min(ParentProgress, Current-Read-Records / Parent-Write-Records)



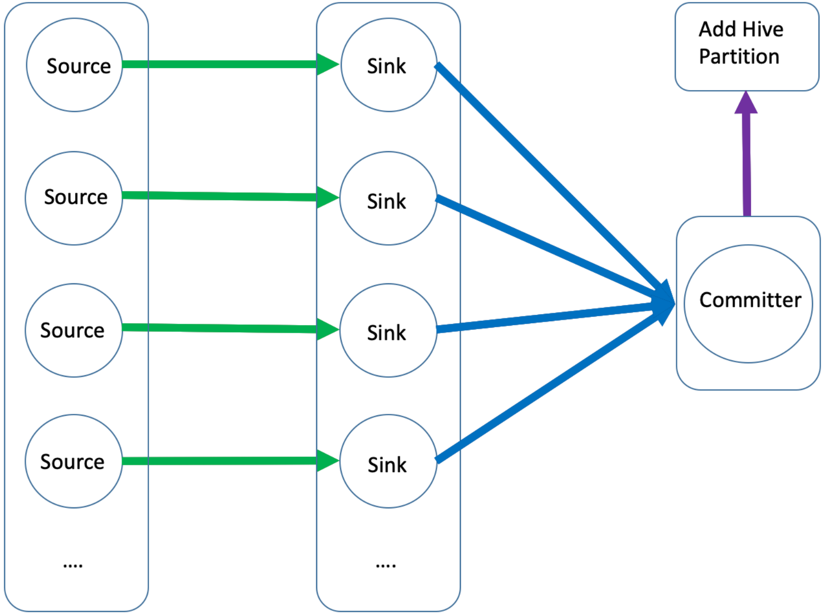
基于Flink批流一体的架构

主要升级点

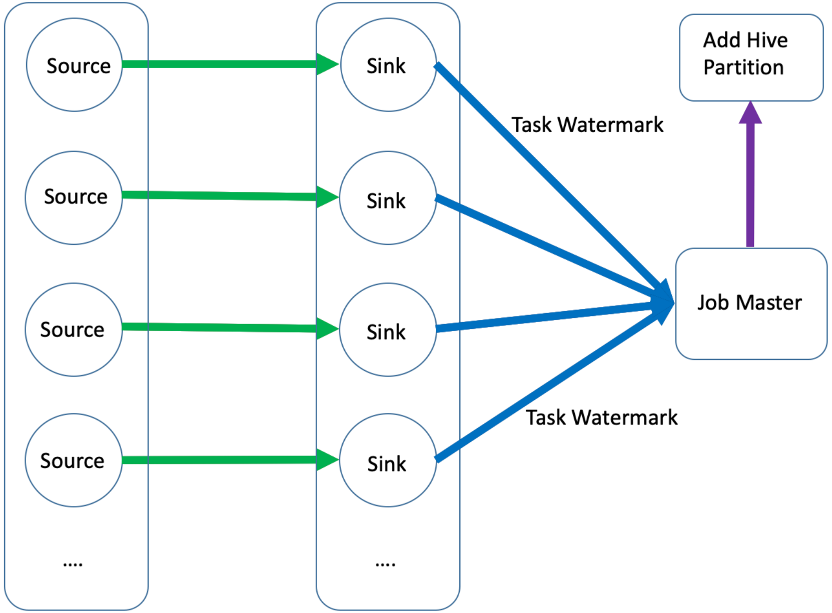
- Flink 1.5 -> Flink 1.9, API 统一到 DataStream API，支持批流一体架构
- 基础框架扩展，支持Exactly Once、Event Time、Auto DDL同步等特性
- 对Flink Core进行多项基础改进，支持推测执行、Region Failover
- Runtime升级，支持云原生架构


Input Sources
Databases (MySQL/ Oracle/etc.)
Message Queues (Kafka/ RocketMQ/etc.)
Big Data Systems (HDFS/Hive/ Clickhouse/etc.)
Framework Functionalities
Exactly Once
Fault Tolerance
Auto Parallelism
Custom Processing
Dirty Data Management
Monitor Alarms
Flink Distributed Runtime
Data Trasmission
Batch
Speculative Execution
DDL Auto Sync
Flow Control
Streaming
Event time archiving
Dynamic Partiion
Incremental
Bin Log Support
Incremental Merge
Output Sinks
Databases (MySQL/ Oracle/etc.)
Message Queues (Kafka/ RocketMQ/etc.)
Big Data Systems (HDFS/Hive/ Clickhouse/etc.)
Runtime (Yarn/Kubernetes/Standalone/Local)

MQ2Hive写入流程优化



Shuffle



Pipelined

基于Flink湖仓一体的架构

初始CDC同步架构

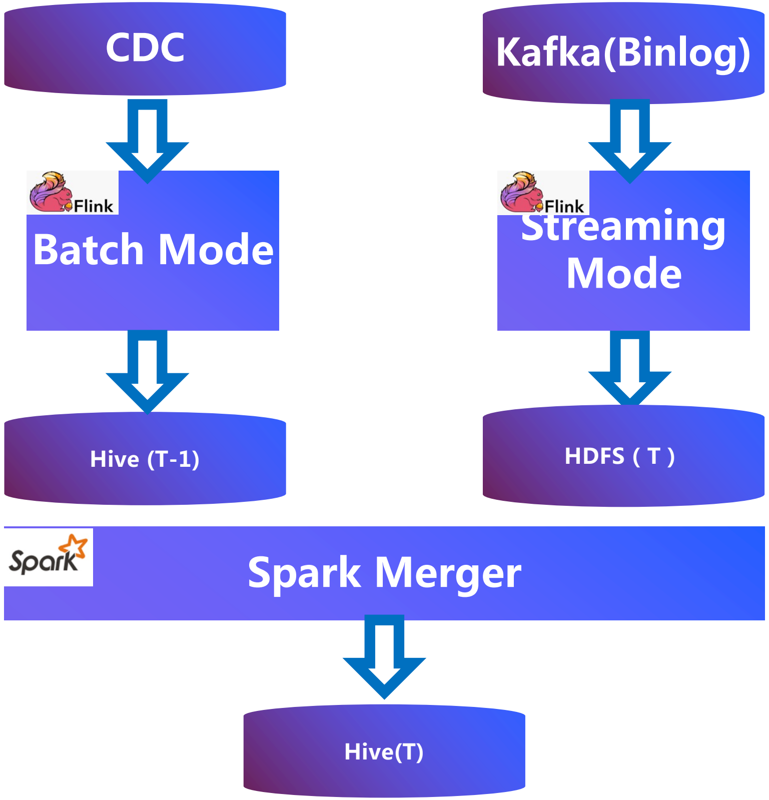
- 数据处理流程比较复杂
- 依赖Flink、Spark多种计算引擎

实时性

- T+1产出，最快小时级延迟，不支持近实时分析场景

效率

- 存储开销大，每个分区都是全量镜像
- 计算成本较高，Merge进行全局Shuffle

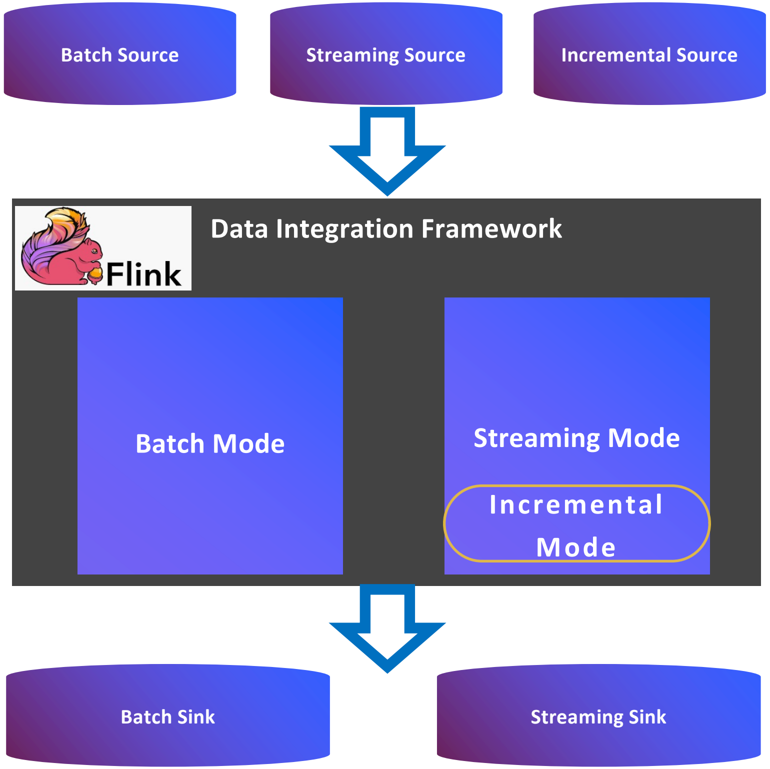


*CDC: Change data capture

基于Flink湖仓一体的架构

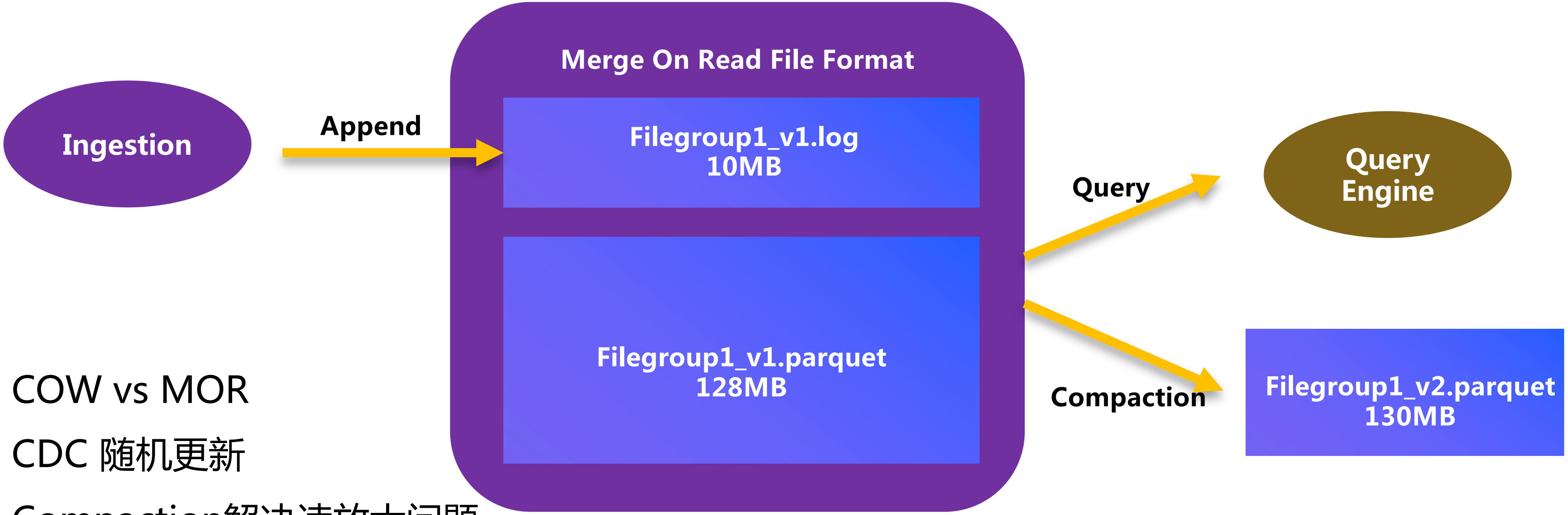
主要升级点

- Flink 1.9 -> Flink 1.11, 接入Hudi数据湖引擎，支持CDC数据变更同步
- 对Hudi引擎进行多项基础改进，以提高整体的写入效率和稳定性
- 近实时写入，延迟<=10min，综合性能提升70+%
- 完成架构统一，一套系统覆盖所有数据同步场景



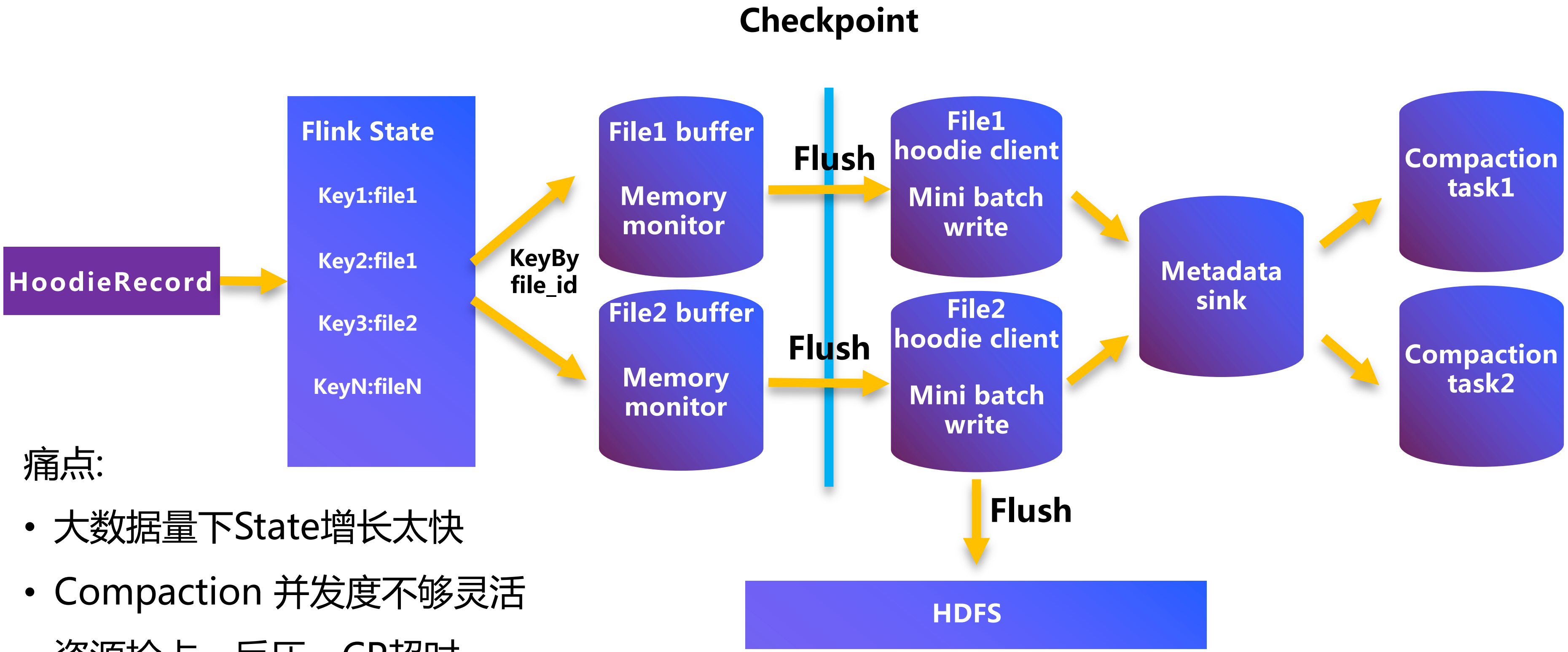
火山引擎 | DataFun.

Table Type选择



- COW vs MOR
- CDC 随机更新
- Compaction解决读放大问题

Hudi实时写入痛点分析

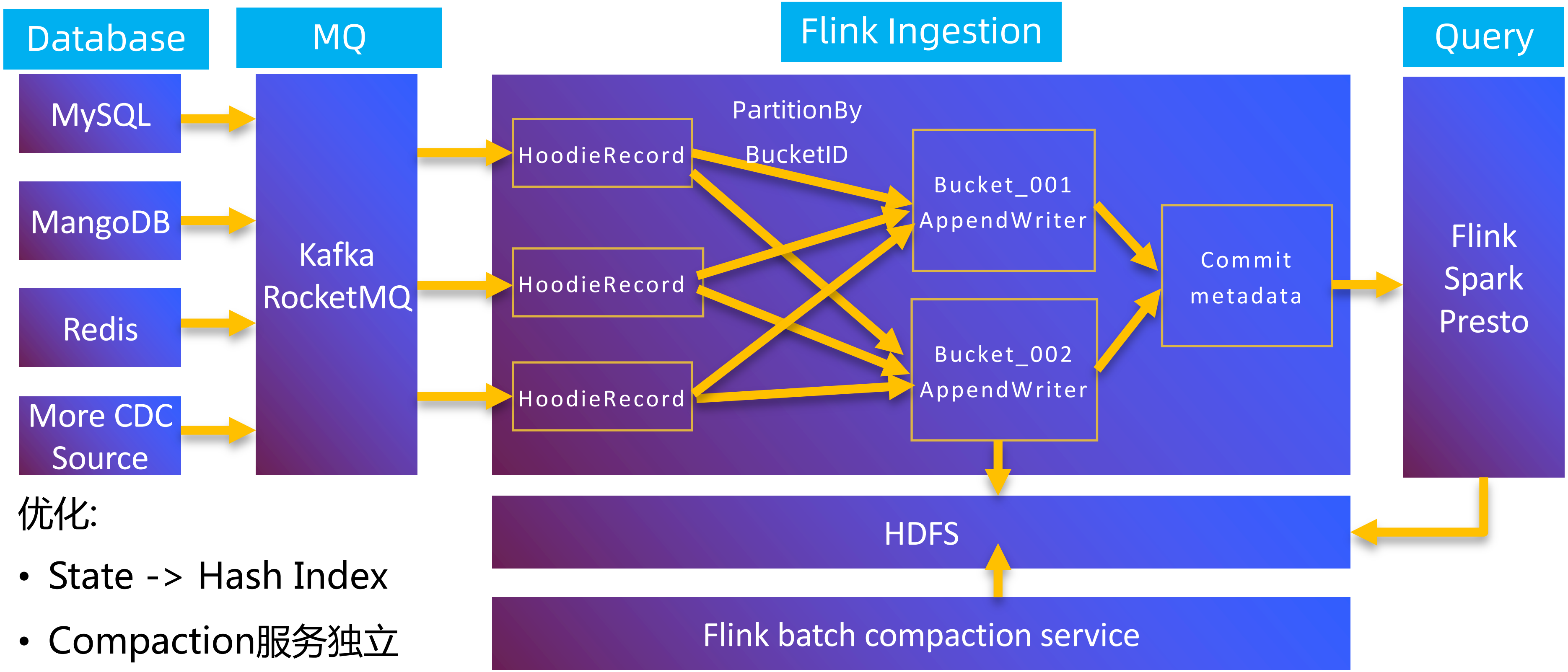


痛点:

- 大数据量下State增长太快
- Compaction 并发度不够灵活
- 资源抢占、反压、CP超时

火山引擎 | DataFun.

优化后Hudi实时写入流程



Database

MySQL

MangoDB

Redis

More CDC Source

MQ

Kafka RocketMQ

Flink Ingestion

HoodieRecord

HoodieRecord

HoodieRecord

PartitionBy BucketID

Bucket_001 AppendWriter

Bucket_002 AppendWriter

Commit metadata

HDFS

Flink batch compaction service

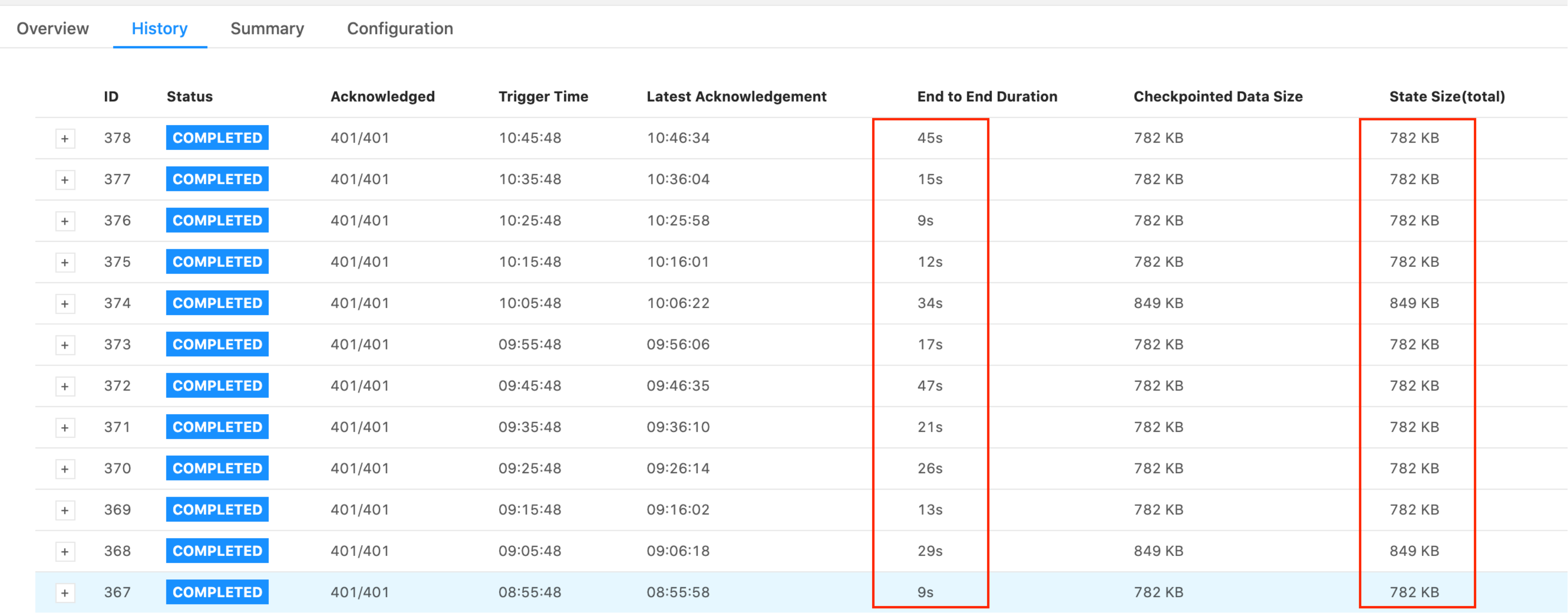
Query

Flink Spark Presto

优化:

- State -> Hash Index
- Compaction服务独立
- 缓存优化

写入效果



	ID	Status	Acknowledged	Trigger Time	Latest Acknowledgement	End to End Duration	Checkpointed Data Size	State Size(total)
+	378	COMPLETED	401/401	10:45:48	10:46:34	45s	782 KB	782 KB
+	377	COMPLETED	401/401	10:35:48	10:36:04	15s	782 KB	782 KB
+	376	COMPLETED	401/401	10:25:48	10:25:58	9s	782 KB	782 KB
+	375	COMPLETED	401/401	10:15:48	10:16:01	12s	782 KB	782 KB
+	374	COMPLETED	401/401	10:05:48	10:06:22	34s	849 KB	849 KB
+	373	COMPLETED	401/401	09:55:48	09:56:06	17s	782 KB	782 KB
+	372	COMPLETED	401/401	09:45:48	09:46:35	47s	782 KB	782 KB
+	371	COMPLETED	401/401	09:35:48	09:36:10	21s	782 KB	782 KB
+	370	COMPLETED	401/401	09:25:48	09:26:14	26s	782 KB	782 KB
+	369	COMPLETED	401/401	09:15:48	09:16:02	13s	782 KB	782 KB
+	368	COMPLETED	401/401	09:05:48	09:06:18	29s	849 KB	849 KB
+	367	COMPLETED	401/401	08:55:48	08:55:58	9s	782 KB	782 KB

火山引擎 | DataFun.

03

通用能力改造

通用能力改造

目标

- 对外能力输出，降低数据建设成本

能力构建

- 低成本共建能力
- 架构的兼容能力

低成本共建能力

思路1

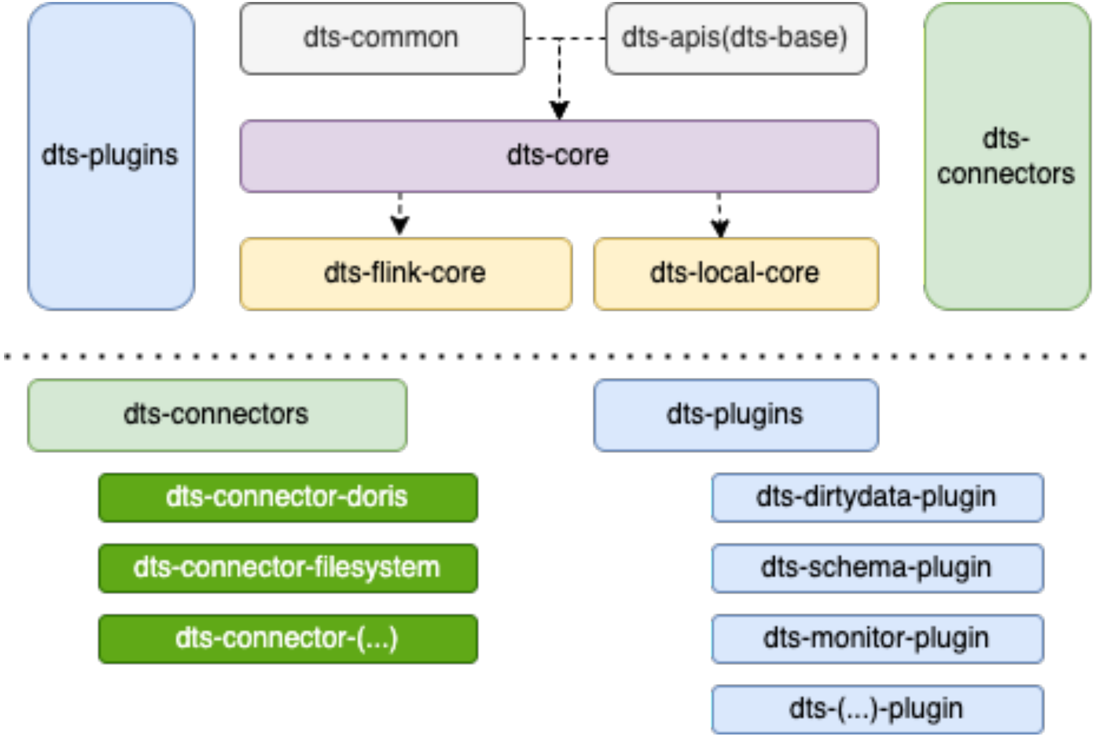
- 模块拆分

现状

- 大Jar包，模块间耦合较重
- 数据处理流程不清晰

解决方案

- 功能模块划分
- 组件可插拔



低成本共建能力

思路2

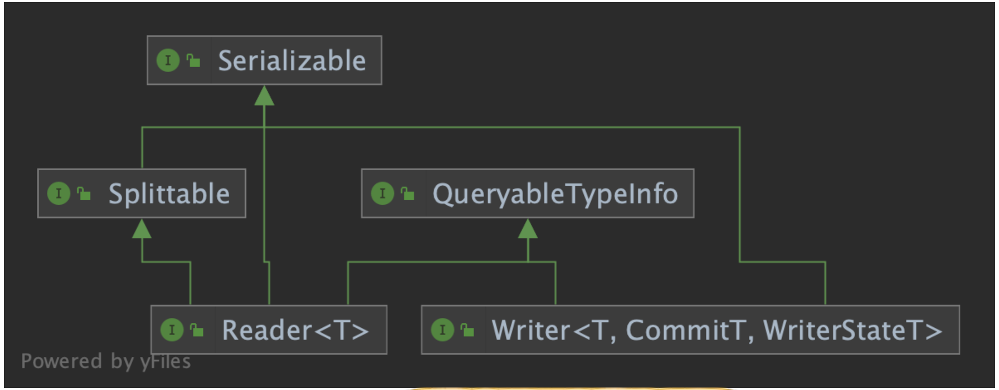
- 接口抽象

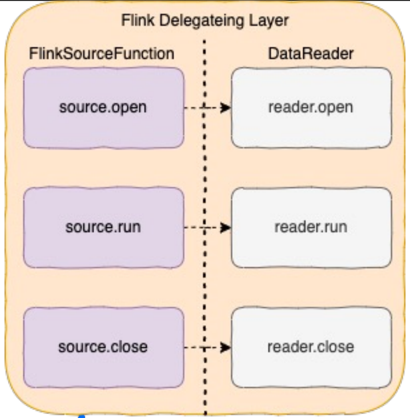
现状

- Flink API深度绑定，较为复杂
- Connectors接入成本高

解决方案

- 抽象新的API接口，与引擎无关
- 屏蔽引擎细节





架构兼容能力

思路1

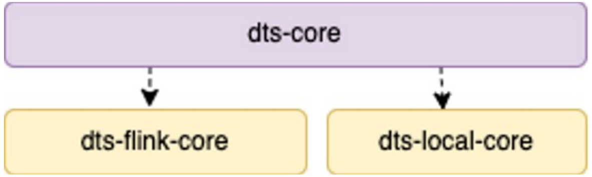
- 多引擎架构

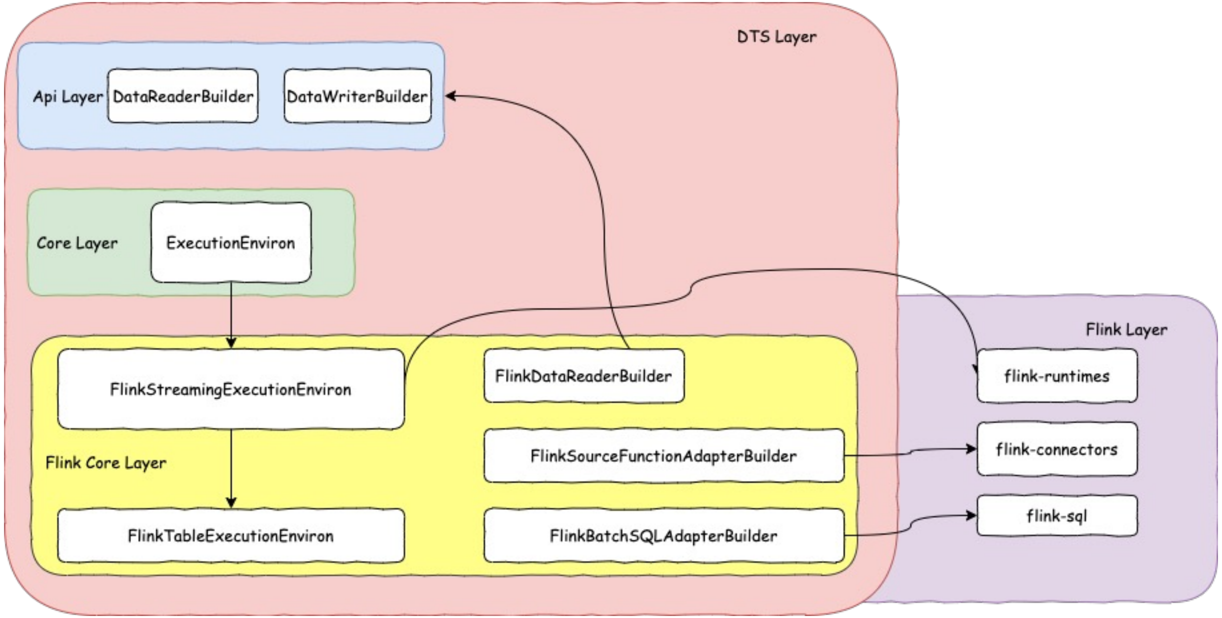
现状

- Flink深度绑定，场景受限制
- 依赖较重，简单场景资源浪费

解决方案

- 预留多引擎入口
- 执行环境抽象
- 探索Local本地执行方式





架构兼容能力

思路2

- 依赖隔离

现状

- 内部依赖
- 绑定公司大数据底座

解决方案

- 剔除内部依赖，采取通用解决方案
- 大数据底座Provided依赖，不绑定固定底座，运行时由外部指定，针对不兼容的场景，通过maven profile、 maven shade隔离
- 针对数据源多版本以及版本不兼容的问题，采取动态加载的策略

04

未来展望

未来展望

多引擎架构

- Local Engine 落地，支持本地执行，提高简单场景资源利用率
- 引擎智能选择策略，针对简单场景使用Local Engine；针对复杂场景复用大数据引擎的能力

通用能力建设

- 新接口推广，对用户屏蔽引擎细节，降低Connector开发成本
- 探索Connector多语言方案

流式数据湖

- 统一CDC数据入湖解决方案，稳定支撑千万级QPS
- 数据湖平台能力构建，覆盖批式、流式、增量使用场景

期待共建与交流

欢迎扫码加入微信群，获取更项目最新进展

非常感谢您的观看

火山引擎 | DataFun.