FLINK FORWARD #ASIA 2020 实时即未来

Flink如何实时分析Iceberg数据湖的CDC数据

阿里巴巴 李劲松/胡争

#1

常见的CDC
分析方案

#2

为何选择 Flink
+ Iceberg

#3

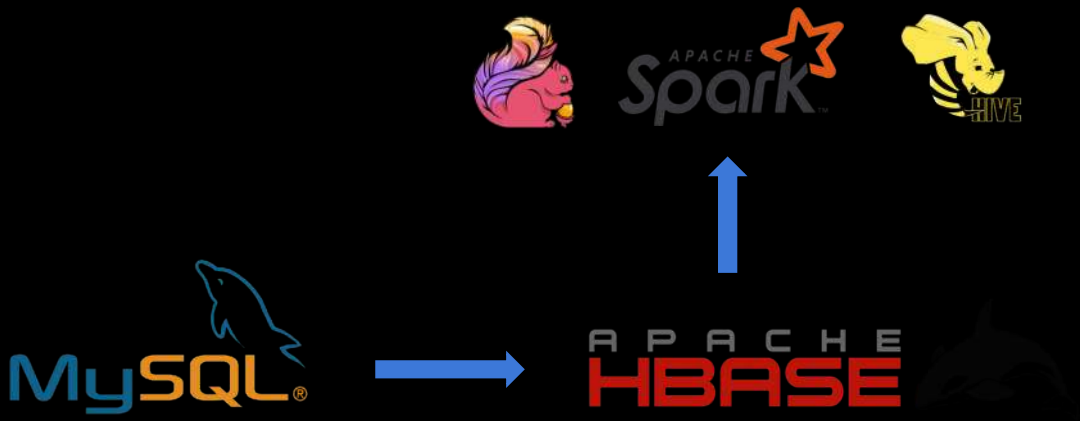
如何实时写
入读取

#4

未来规划

#1 常见的CDC分析方案

离线 HBase 集群分析 CDC 数据



方案评估

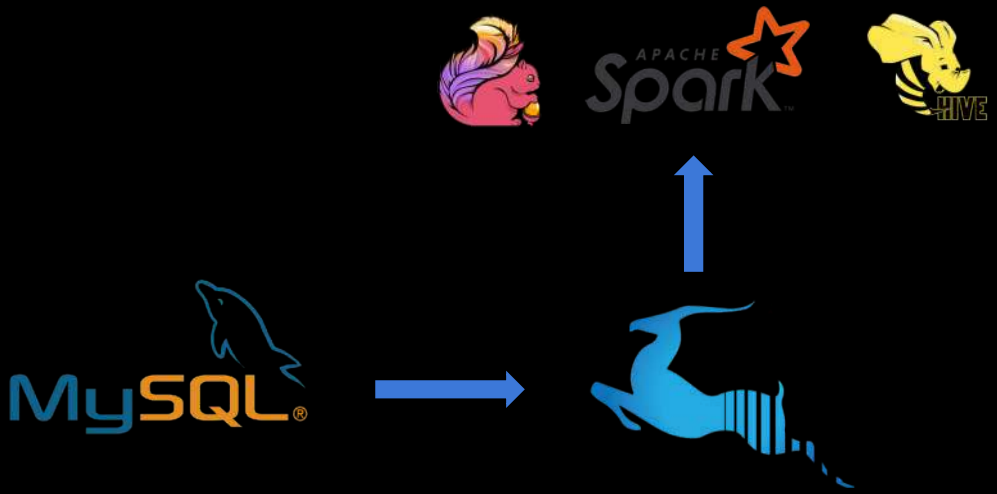
优点

1、CDC记录实时写入HBase。高吞吐 + 低延迟。
2、小范围查询延迟低。
3、集群可拓展

缺点

1、行存索引不适合分析任务。
2、HBase集群维护成本较高。
3、通过RegionServer定位HFile，Server的优化和缓存完全用不上。
4、数据格式绑定HFile，不方便拓展到Parquet、Avro、Orc等。

Apache Kudu 维护 CDC 数据集



方案评估

优点

1、支持实时更新数据，时效性佳。
2、列存加速，适合OLAP分析。

缺点

1、独立的Kudu集群，比较小众。维护成本高。
2、和 HDFS / S3 / OSS 等割裂。数据独立，且存储成本不如S3 / OSS。
3、Kudu的批量扫描不如parquet。
4、不支持增量拉取。

直接导入CDC到Hive分析

方案评估

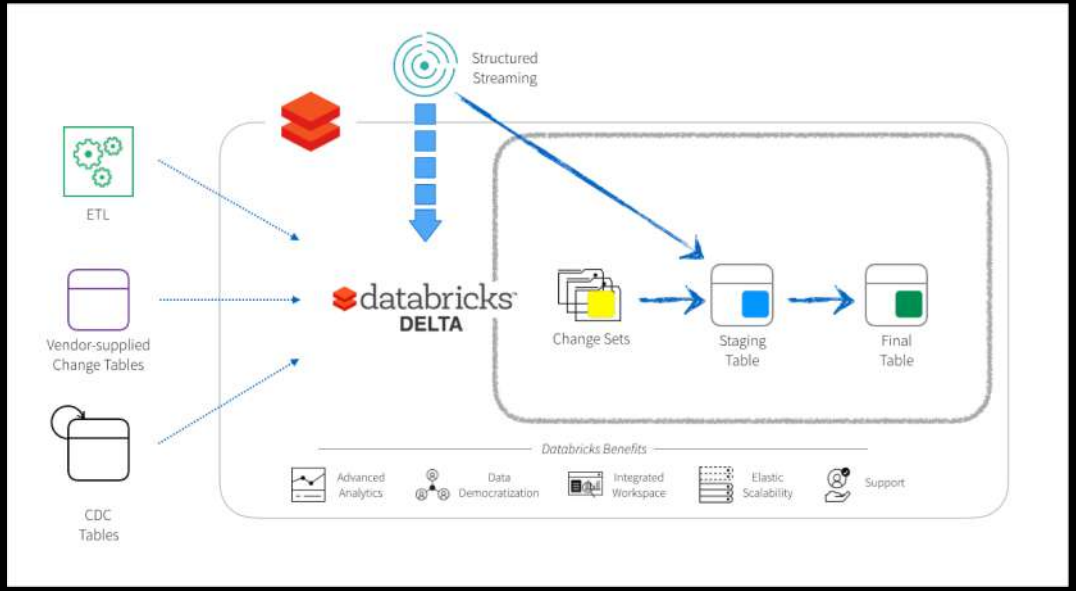
优点

1、流程能工作
2、Hive存量数据不受增量数据影响。

缺点

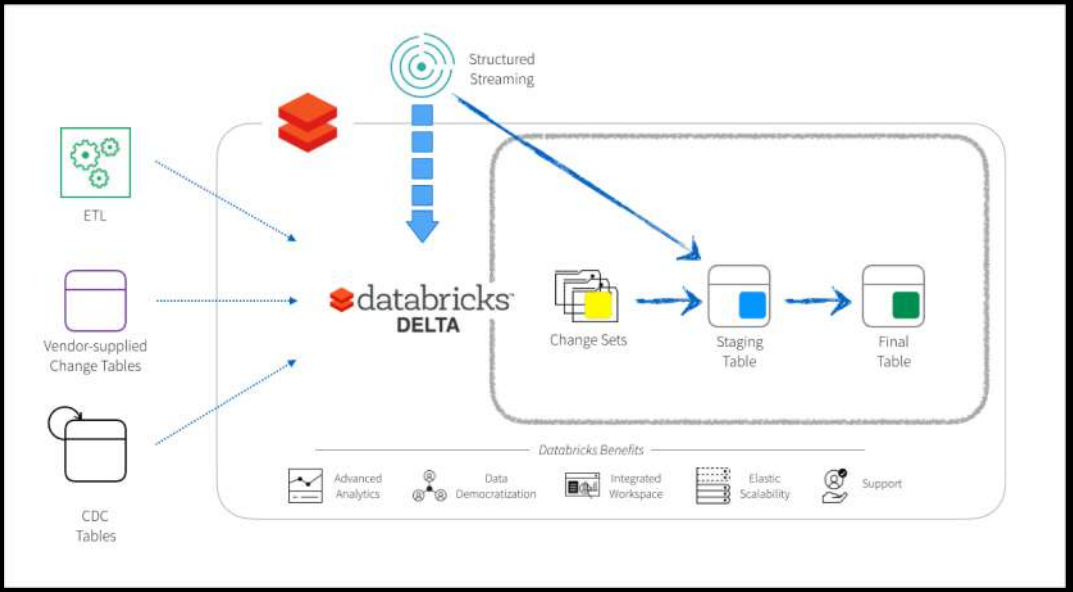
1、数据不是实时写入；
2、每次数据导致都要 MERGE 存量数据。T+1 方式更新，时效性差。
3、不支持实时upsert。

Spark + Delta 分析CDC数据




```
MERGE INTO users
USING changes ON users.userId = changes.userId
WHEN MATCHED AND FLAG='D' THEN DELETE
WHEN MATCHED AND FLAG<>'D'
THEN UPDATE address = changes.addresses
WHEN NOT MATCHED
THEN INSERT (userId, address)
VALUES (changes.userId, changes.address)
```

Spark + Delta 分析CDC数据



方案评估

优点

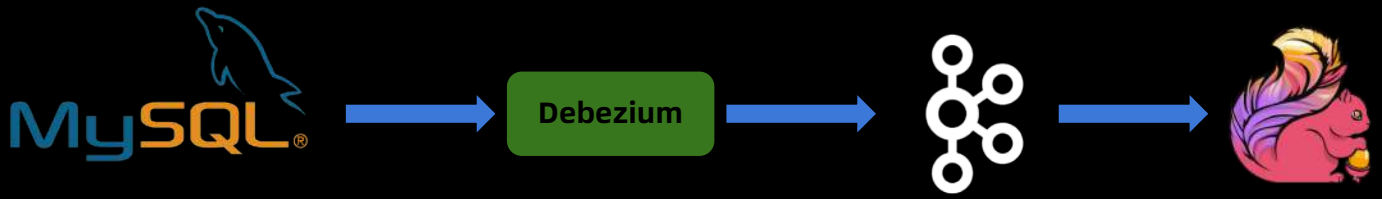
1、仅依赖 Spark+Delta，架构简洁。
2、无在线服务。维护和运行成本低。
2、列存存储，分析速度快。
3、方便上S3/OSS，超高性价比。

缺点

1、增量和全量表割裂，时效性不足。
2、设计和维护额外的Change Set表。
3、计算引擎并非原生支持CDC。
4、不支持实时Upsert。

#2 为何选择 Flink + Iceberg ?

Flink 原生支持 CDC 数据消费



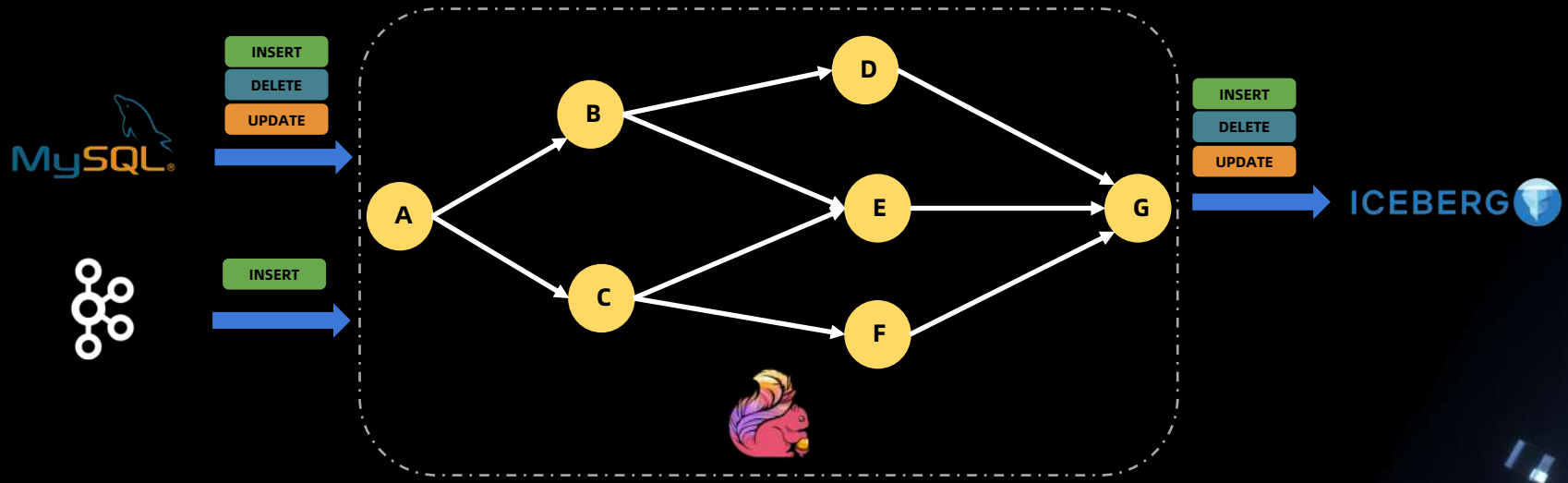
Flink 原生支持 CDC 数据消费

github.com/ververica/flink-cdc-connectors

```
-- creates a mysql cdc table source
CREATE TABLE mysql_binlog (
 id INT NOT NULL,
 name STRING,
 description STRING,
 weight DECIMAL(10,3)
) WITH (
 'connector' = 'mysql-cdc',
 'hostname' = 'localhost',
 'port' = '3306',
 'username' = 'flinkuser',
 'password' = 'flinkpw',
 'database-name' = 'inventory',
 'table-name' = 'products'
);

-- read snapshot and binlog data from mysql, and
-- do some transformation, and show on the client
SELECT id, UPPER(name), description, weight FROM mysql_binlog;
```

Flink 原生支持 Change Log Stream



Flink + Iceberg CDC导入方案

方案评估

优点

1、支持近实时导入和实时读取。
2、计算引擎原生支持CDC摄入，不需要额外的业务字段设计。
3、统一的数据湖存储，多样化的计算模型。
4、读取合并后的历史数据可充分利用列存加速。
5、云原生支持。
6、支持增量拉取。
7、架构足够简洁，无在线服务节点。

#3 如何实时写入读取?

批量更新场景 VS CDC写入场景

对比项	批量更新场景	CDC写入场景
典型场景	1. GDPR; 2. 批量删除data lake中某些共同特征的数据集。	1. Flink聚合结果实时upsert目标表; 2. Binlog实时导入data lake供数据分析。
示例	UPDATE test SET a = a + 1 WHERE a >100	UPDATE test SET (1,2) WHERE a=0 AND b=0
Query特点	1. 可携带任意过滤条件; 2. 不依赖key;	一般提供对应行的所有列旧值和新值;
数据量	单条Query更新大量数据集	单条Query仅更新一行数据
计算模型	长耗时的批作业	流式增量导入
更新频率	低频更新	高频更新

Apache Iceberg 设计CDC写入方案需要考虑的问题？

正确性

保证 Iceberg 表数据和
源表数据最终一致性

高效写入

支持并发写入
吞吐量足够大

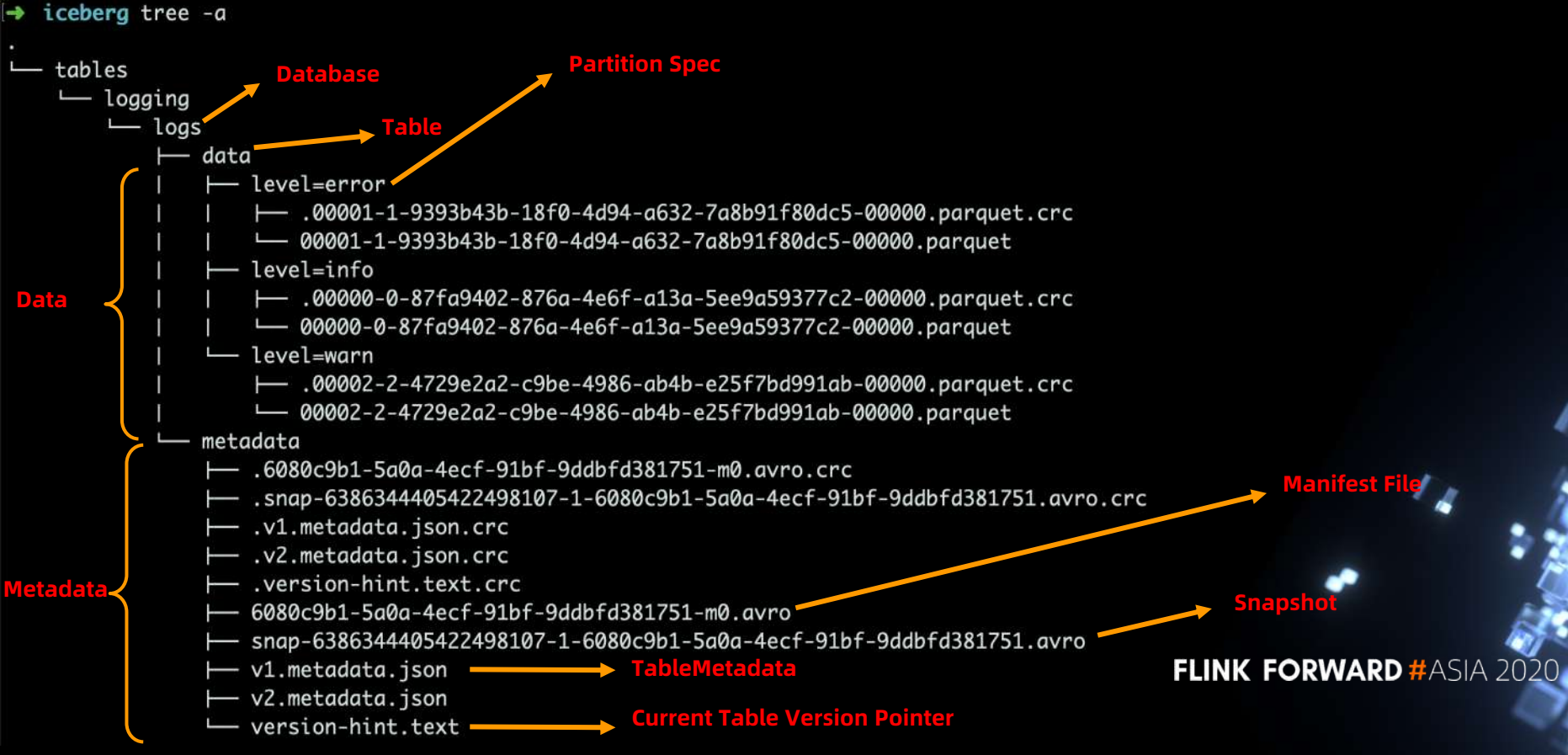
快速读取

Partition或Bucket级别
并发Merge-On-Read

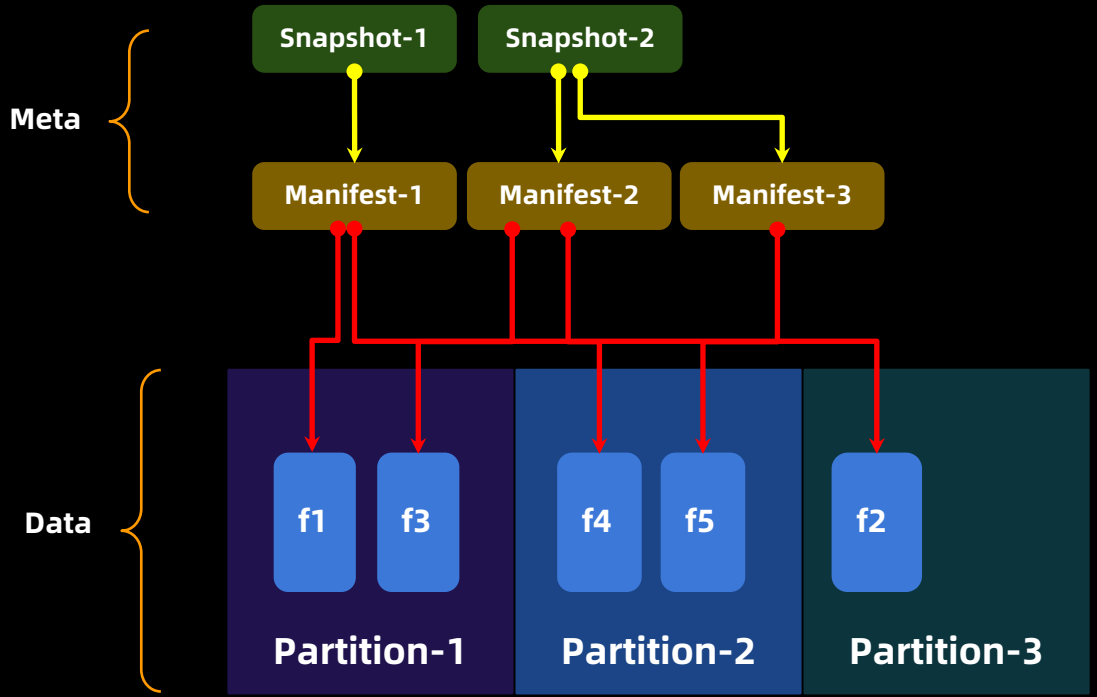
增量读

支持增量拉取便于进一
步数据Transform

Apache Iceberg Basic

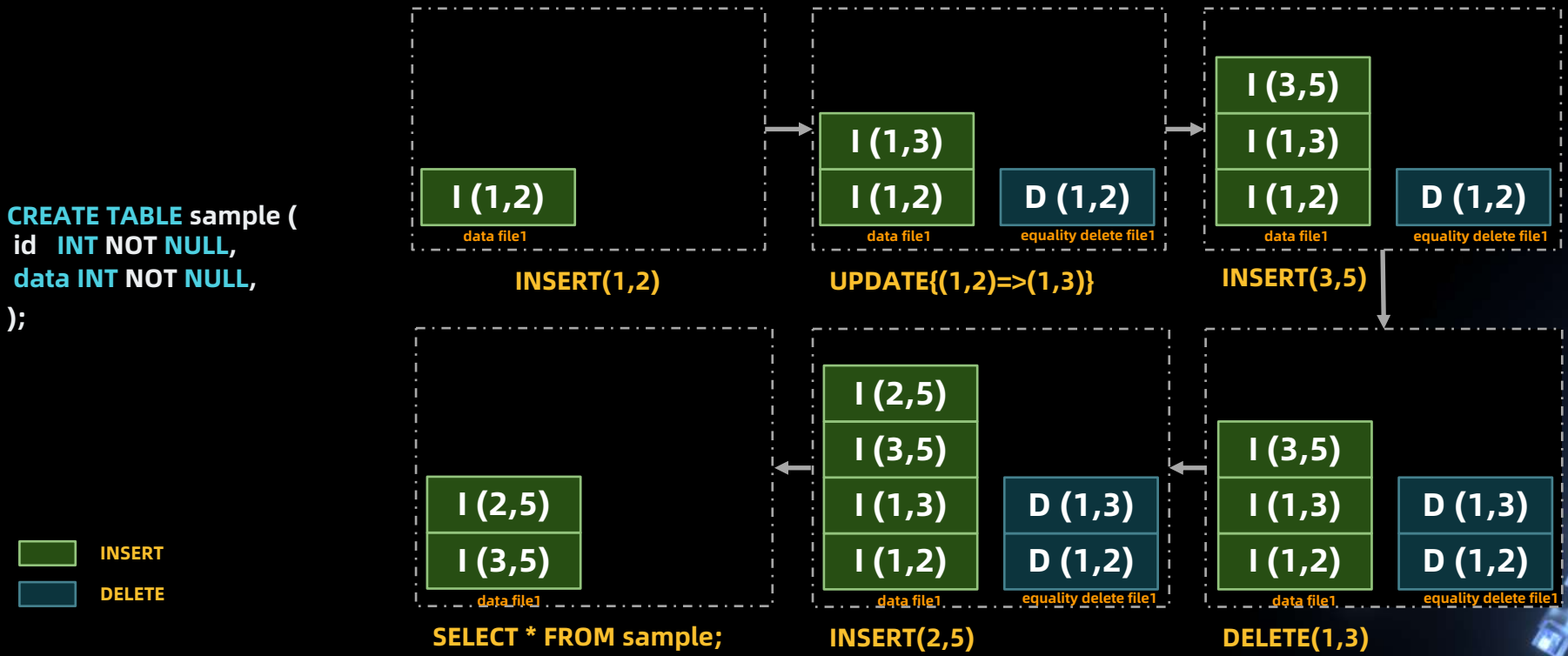


Apache Iceberg Basic

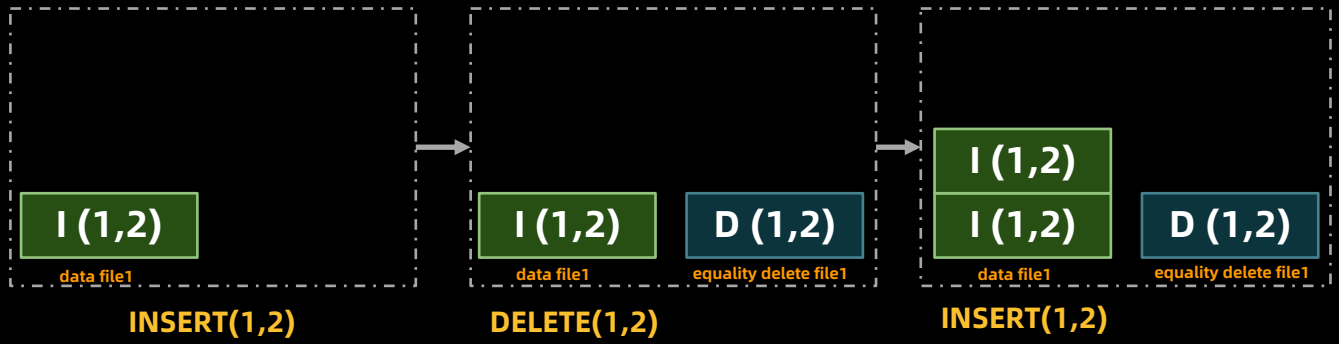


INSERT / UPDATE / DELETE 写入

```
CREATE TABLE sample (
 id   INT NOT NULL,
 data INT NOT NULL,
);
```



问题：同一行的多次更新导致错误语义？

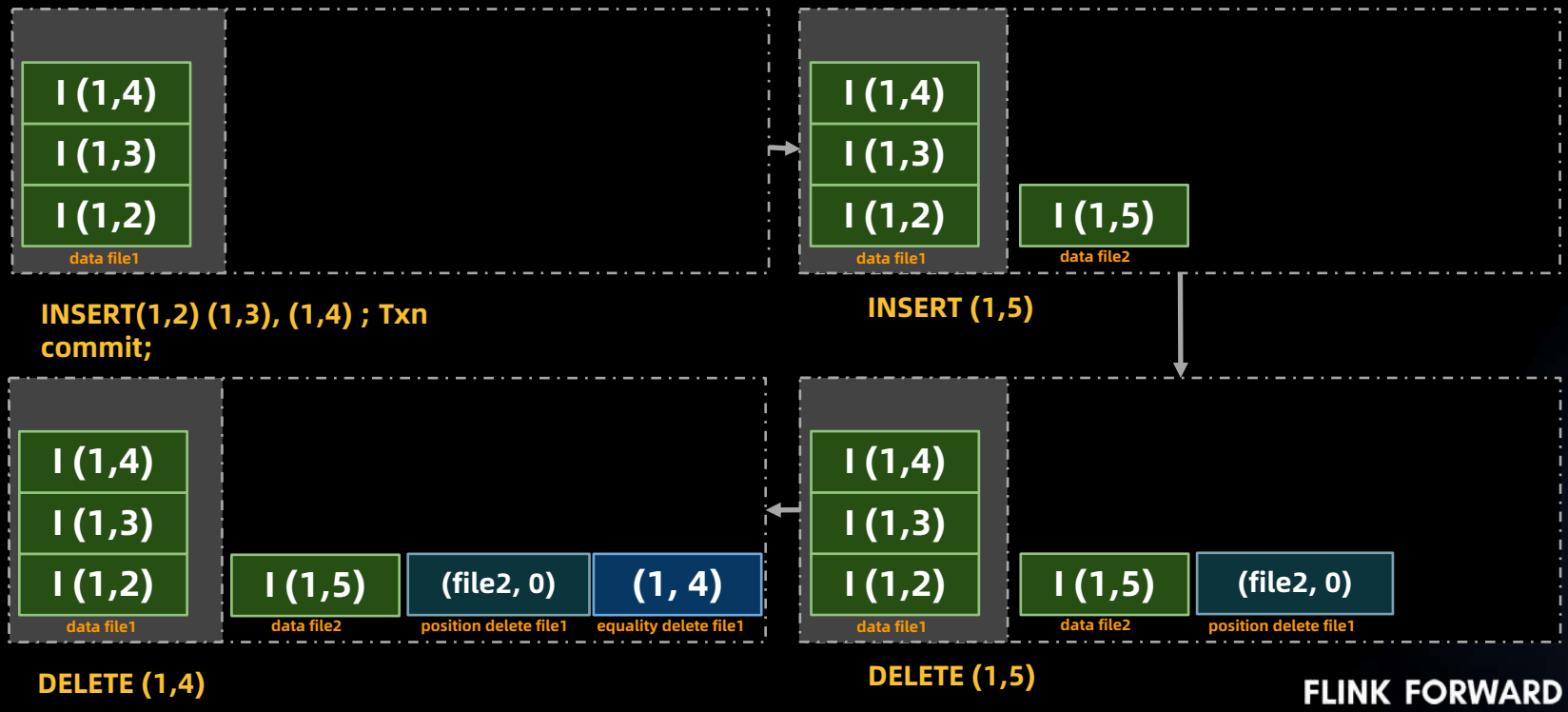


问题：Merge-On-Read最终读取结果为空集，实际应该返回 I(1,2)。

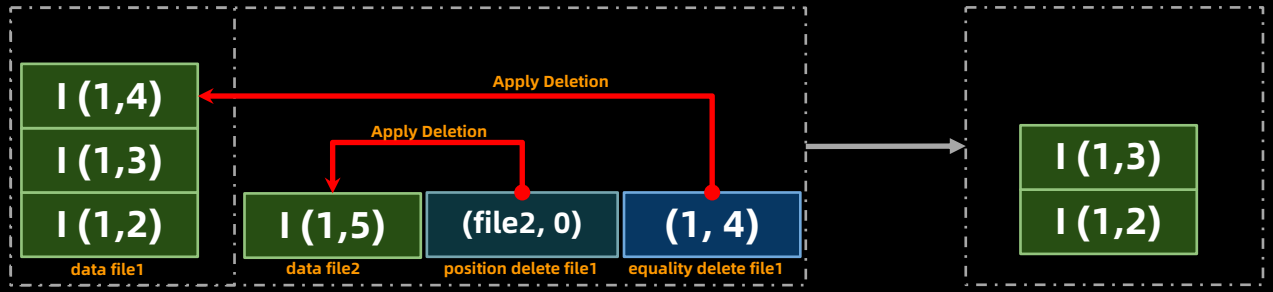
INSERT

DELETE

方案：Mixed pos-delete and equality-delete



方案：Mixed pos-delete and equality-delete



写入思路

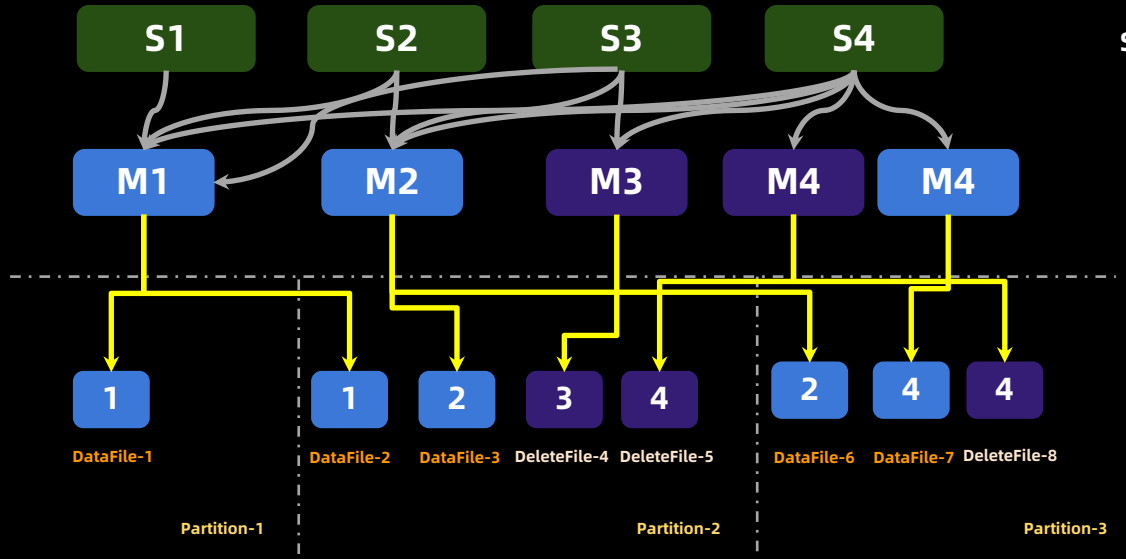
1、待Delete行在本次Txn内Insert过，则把Insert行的(file,pos)写Position Delete File；否则直接把Delete行写Equality delete file。

读取思路

1、Position Delete File和那些SeqNum不大于自己SeqNum的Data File做 JOIN；

2、Equality Delete File和那些SeqNum小于自己SeqNum的Data File做 JOIN。

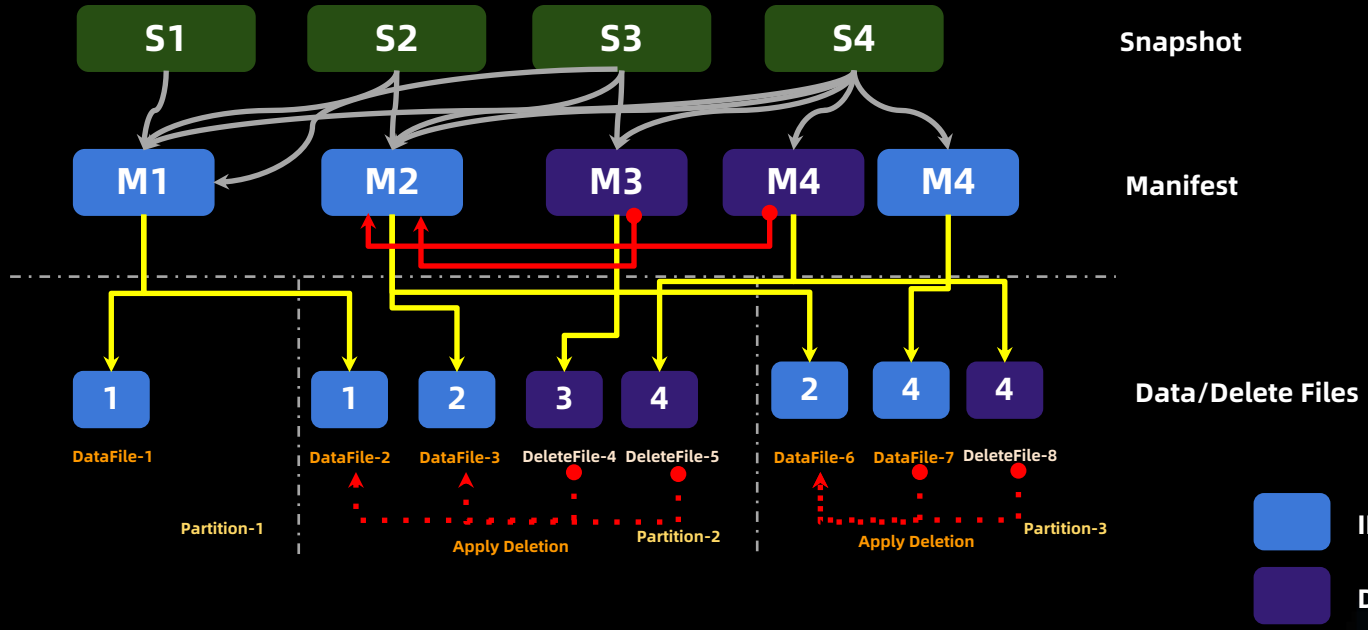
Manifest文件的设计



TXN.1: INSERT INTO click_events SET VALUES (...)
TXN.2: INSERT INTO click_events SET VALUES (...)
TXN.3: DELETE FROM click_events WHERE primary_key = XX
TXN.4: UPDATE click_events SET VALUES(...) WHERE primary_key = XX

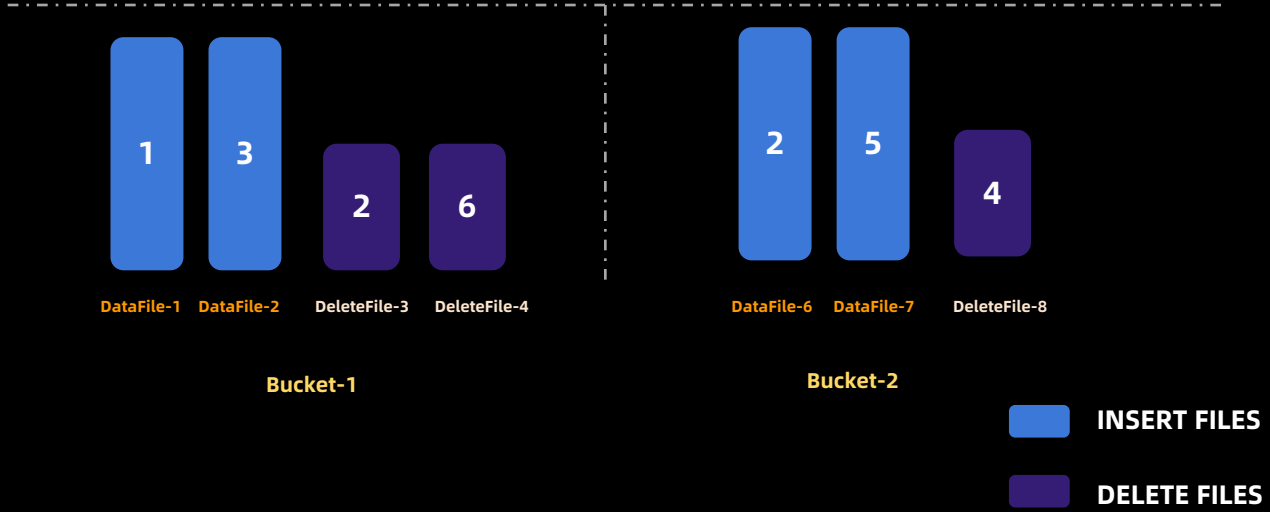
INSERT FILES
DELETE FILES

Manifest文件的设计

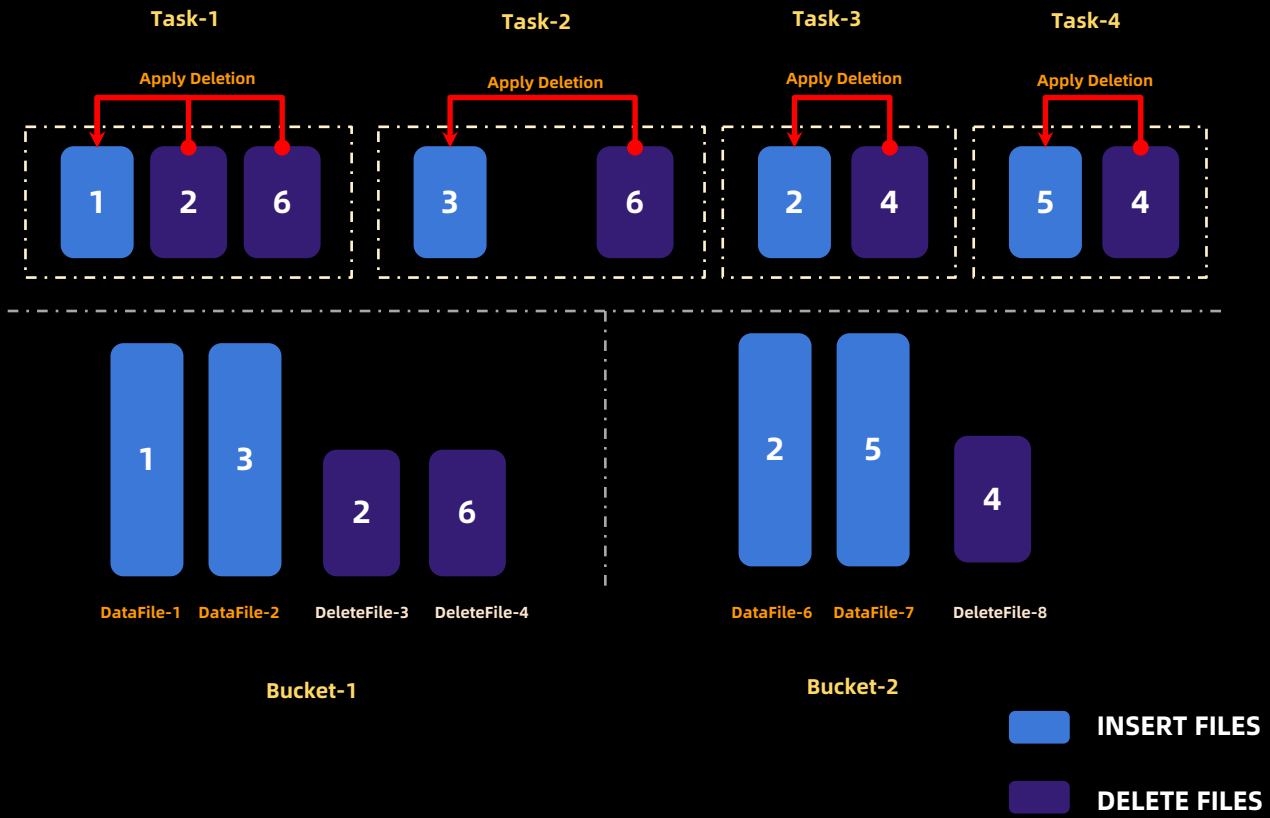


本质：拆分Data和Delete Manifests，快速为每一个DataFile找出对应的Delete File列表。

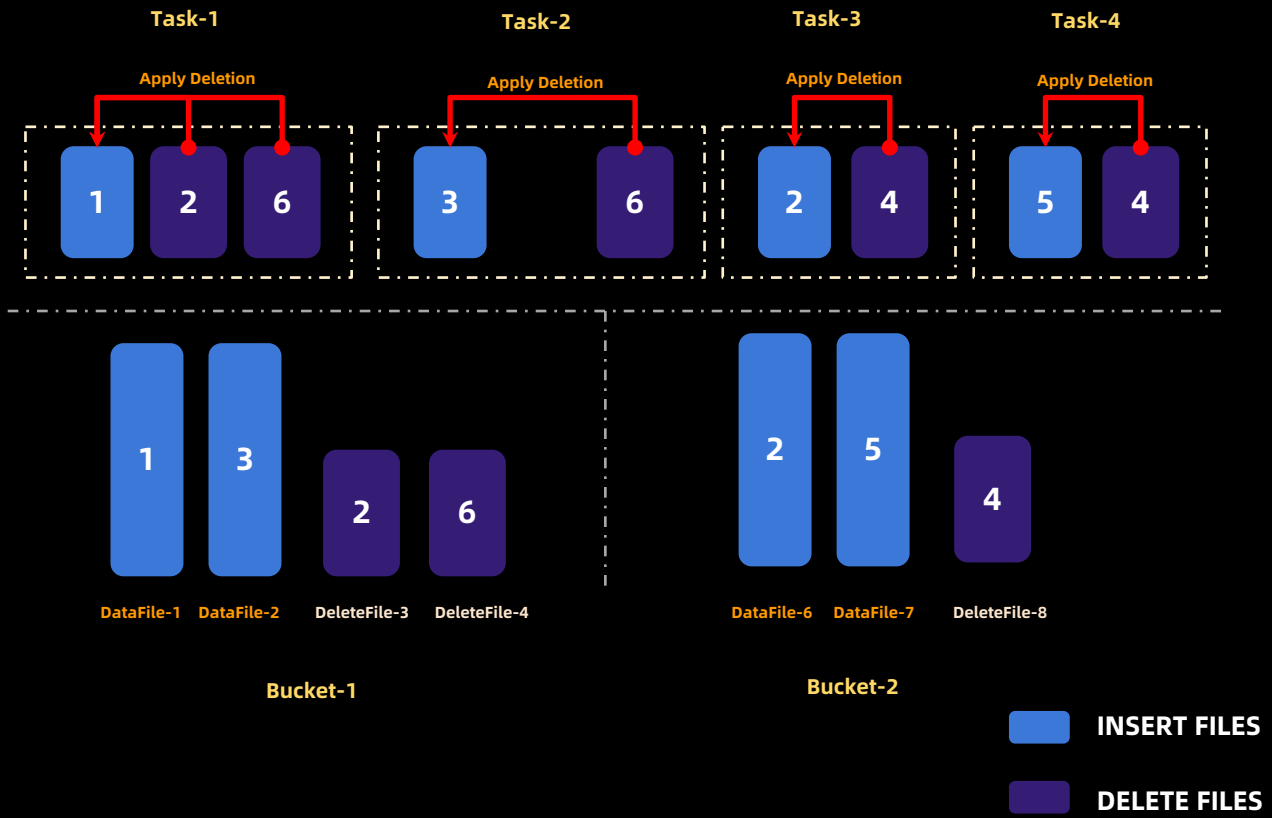
文件级别并发读取


1
3
2
6
DataFile-1
DataFile-2
DeleteFile-3
DeleteFile-4
Bucket-1
2
5
4
DataFile-6
DataFile-7
DeleteFile-8
Bucket-2
INSERT FILES
DELETE FILES

文件级别并发读取


Task-1
Task-2
Task-3
Task-4
Apply Deletion
Apply Deletion
Apply Deletion
Apply Deletion
1
2
6
3
6
2
4
5
4
1
3
2
6
2
5
4
DataFile-1
DataFile-2
DeleteFile-3
DeleteFile-4
DataFile-6
DataFile-7
DeleteFile-8
Bucket-1
Bucket-2
INSERT FILES
DELETE FILES

文件级别并发读取



方案总结

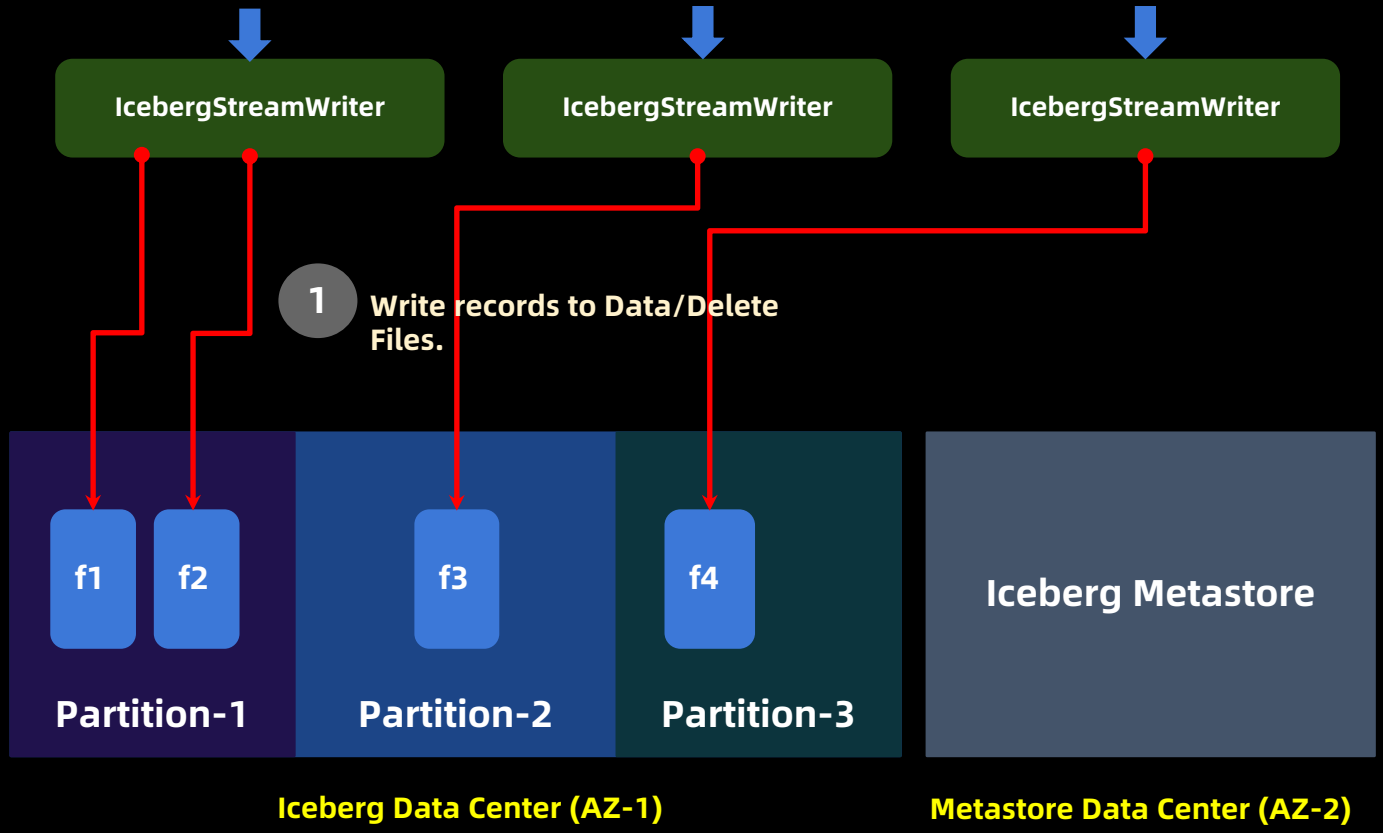
优点

1、满足正确性要求；
2、实现高吞吐写入；
3、满足并发高效读取；
4、可以实现snapshot级别的增量拉取。

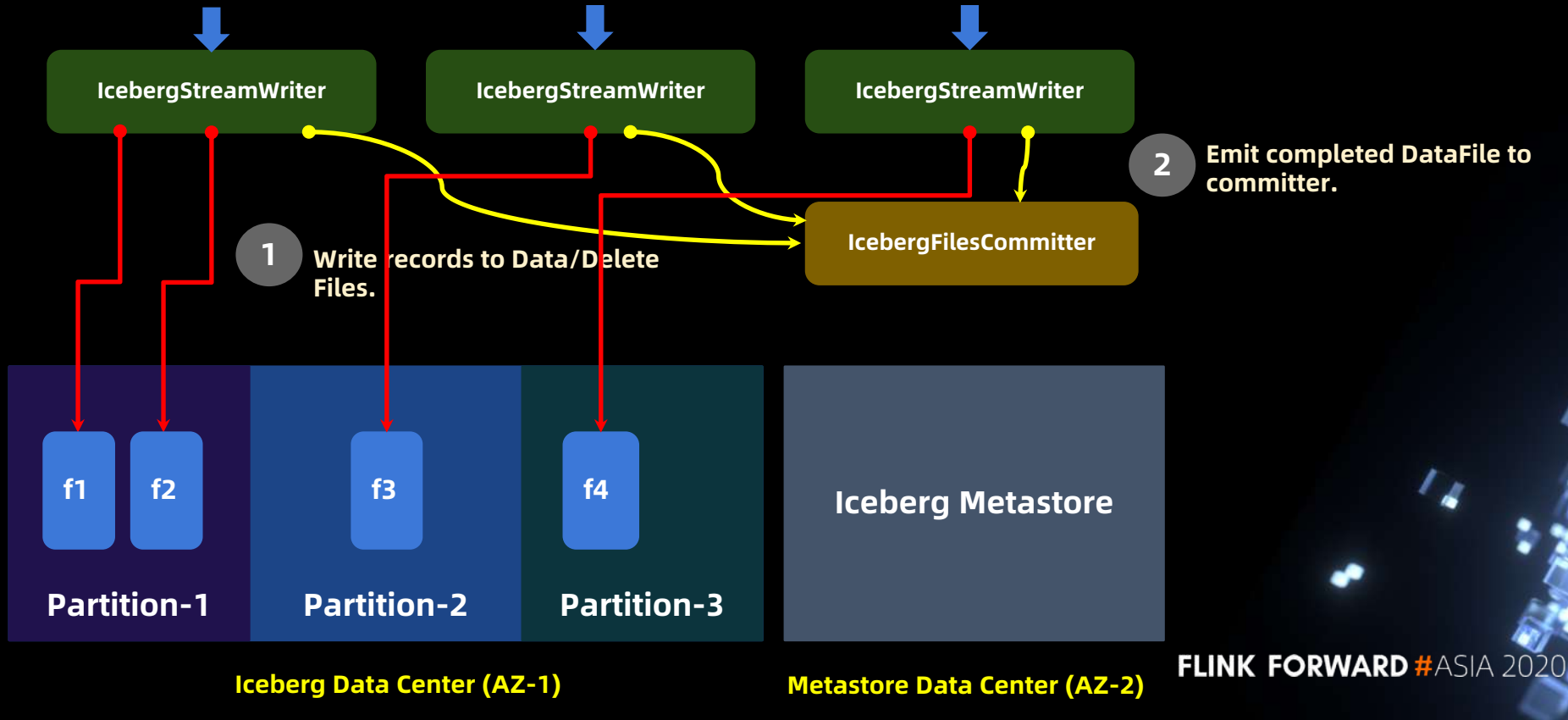
优化点

1、同一个Task内的重复Delete File可以缓存，加速 JOIN 效率；
2、对于DeleteFile溢出到Disk的情况，可考虑借助KV lib优化；
3、设计Bloom Filter过滤大量无效IO；
4、合理的Compact策略，以便控制Delete File的大小。

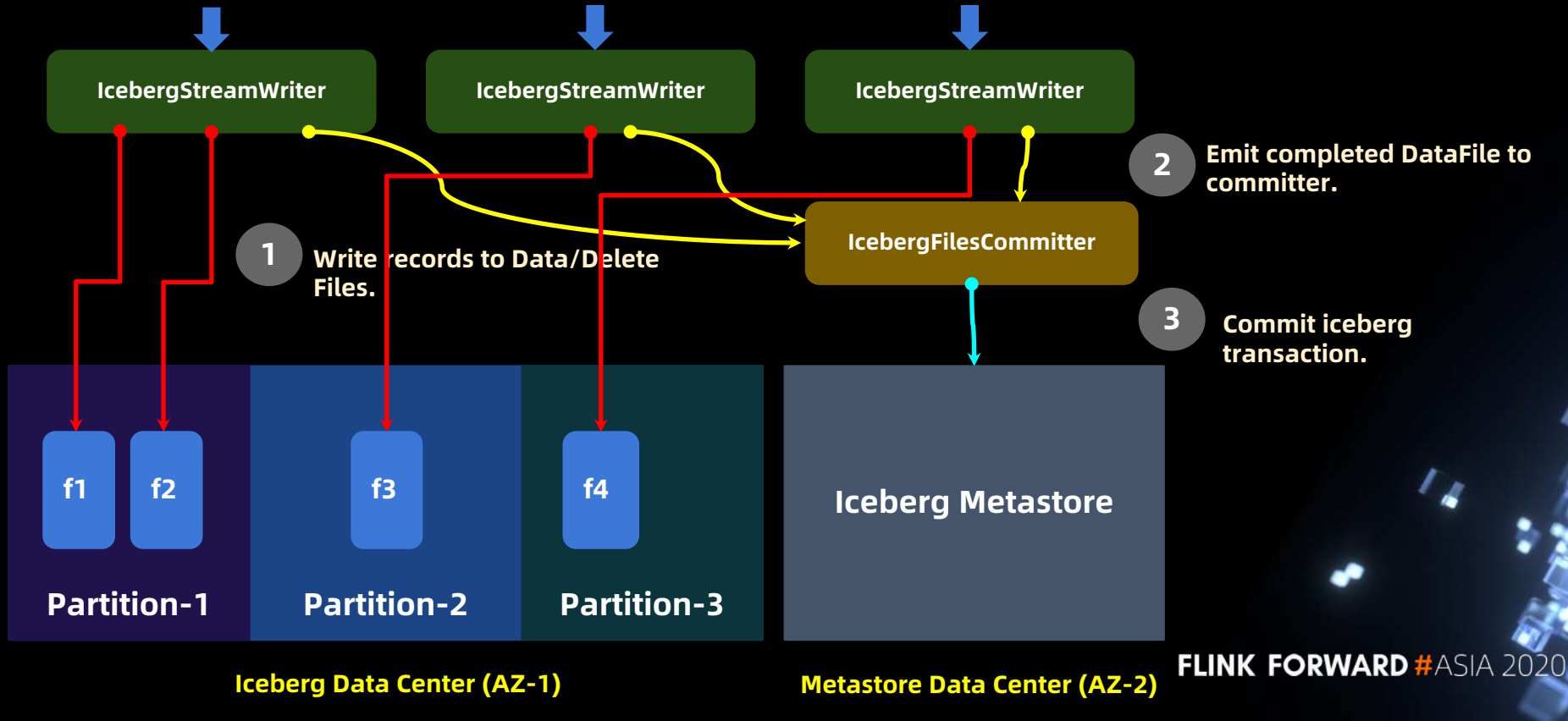
增量文件集的Transaction提交



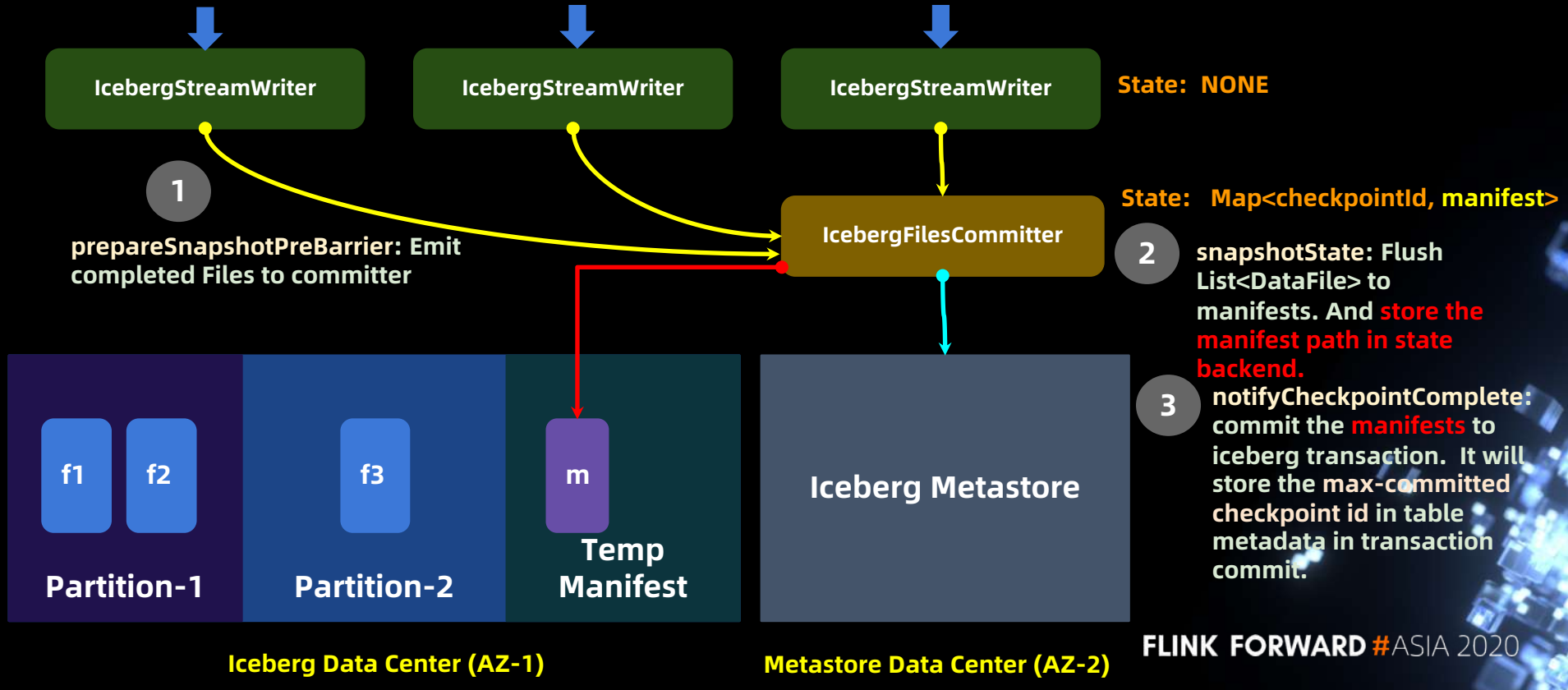
增量文件集的Transaction提交



增量文件集的Transaction提交



增量文件集的Transaction提交 - State设计



#4 未来规划

未来工作规划

Iceberg内核优化

1、大体量数据场景下的全链路测试，加强方案的整体链路稳定性。

2、如前面说述的分析性能优化。

3、提供CDC增量拉取相关Table API接口。

Flink集成

1、实现CDC数据自动合并和手动合并对接；

2、提供Flink增量拉取CDC数据的能力。

其他生态集成

1、Spark Streaming 对接CDC写入链路

2、Presto等引擎对接查询链路。

3、借助开源Alluxio加速数据查询。


FLINK FORWARD
#ASIA 2020
实时即未来
谢谢


FLINK FORWARD
#ASIA 2020
实时即未来
谢谢


FLINK FORWARD
#ASIA 2020
实时即未来
谢谢