HONGBO ZENG

Airbnb数据平台实践

airbnb

CNUTCon 2017
全球运维技术大会

上海·光大会展中心大酒店 | 2017.9.10–11

智能时代的新运维

大数据运维

DevOps

安全

SRE

Kubernetes

Serverless

游戏运维

AIOps

智能化运维

基础架构

监控

互联网金融

主办方 Geekbang. 极客邦科技 InfoQ

Agenda

- Data Platform at Airbnb
- Cluster Evolution
- Incremental Data Replication - ReAir
- Unified Streaming and Batch Processing - AirStream

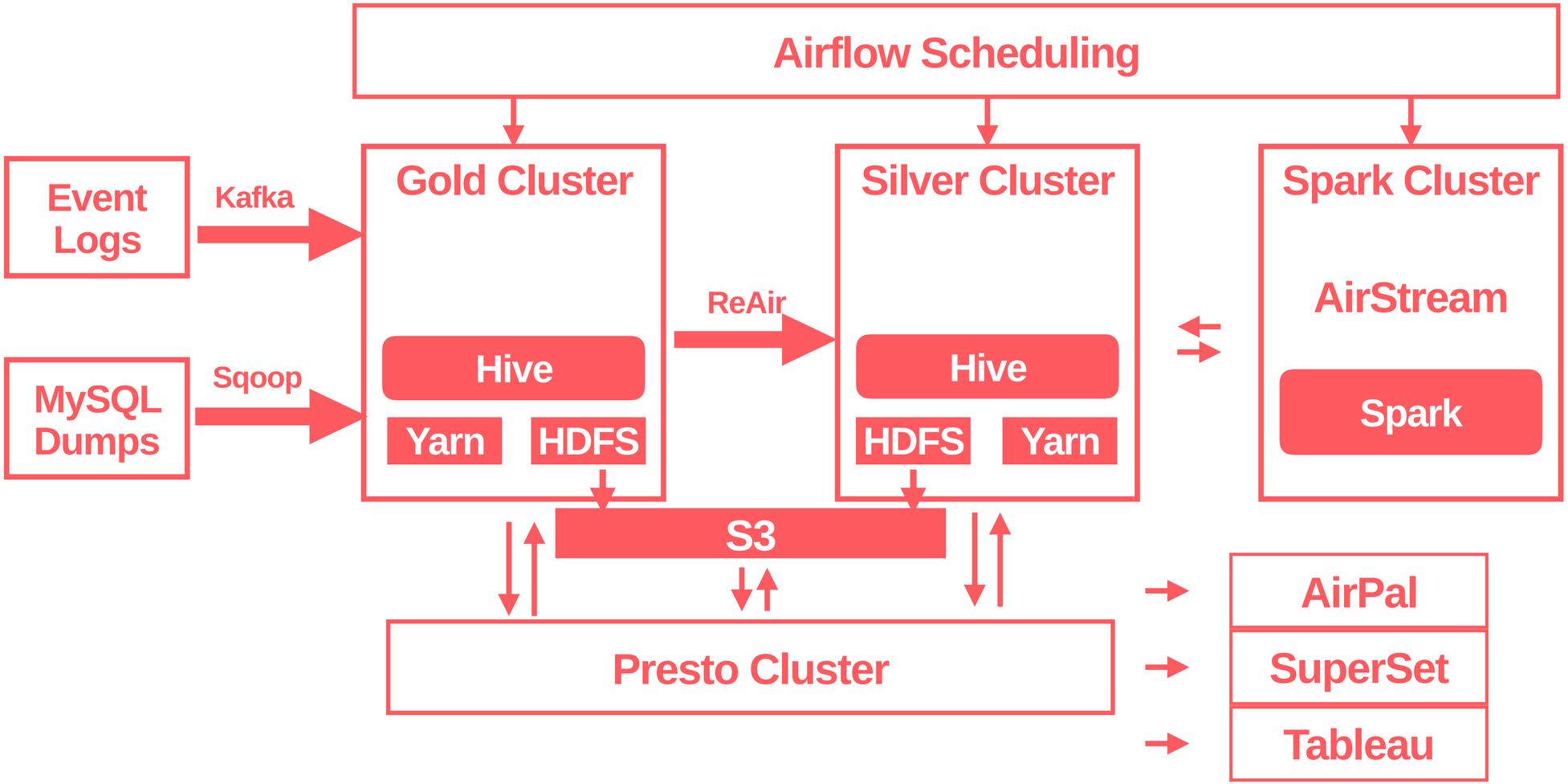
Agenda

- **Data Platform at Airbnb**
- Cluster Evolution
- Incremental Data Replication - ReAir
- Unified Streaming and Batch Processing - AirStream

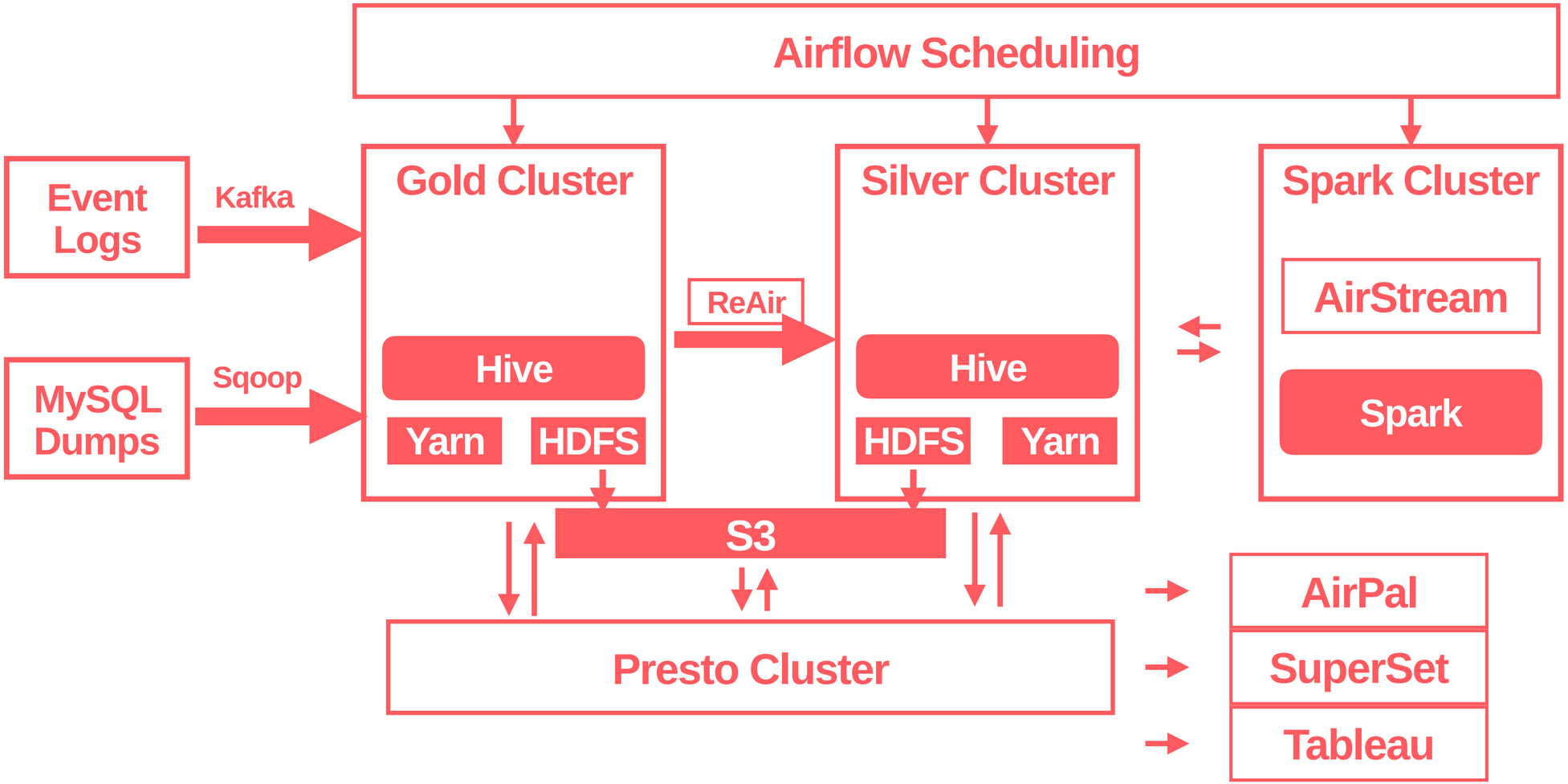
Scale of Data Infrastructure at Airbnb

>13B	>35PB	1400+	5x
#Events Collected	Warehouse Size	Machines Hadoop + Presto + Spark	YoY Data Growth

Data Platform



Data Platform



Agenda

- Data Platform at Airbnb
- **Cluster Evolution**
- Incremental Data Replication - ReAir
- Unified Streaming and Batch Processing - AirStream

Original Cluster

Setup

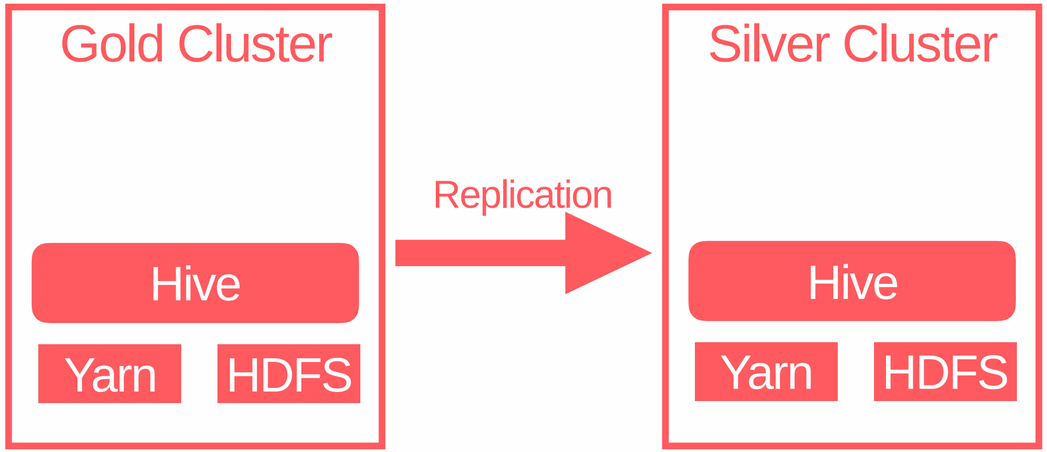
- Single HDFS, MR and Hive installation
- c3.8xlarge (32 cores / 60G mem / 640GB disk) + 3TB of EBS volume
- 800 nodes
- Tested DN on different AZ’s
- All data managed by Hive

Challenges

- Limited isolation between production / adhoc
- Adhoc
  - Difficult to meet SLA’s
  - Harder for capacity plan
- Disaster recovery
- Difficult roll outs

Two Clusters

- Two independent HDFS, MR, Hive metastores
- d2.8xlarge w/ 48TB local
- ~250 instances in final setup
- Replication of common / critical data - Silver is super of Gold
- For disaster recovery, separate AZ's



Multi-Cluster Trade-Offs

Advantages

- Failure isolation with user jobs
- Easy capacity planning
- Guarantee SLA’s
- Able to test new versions
- Disaster Recovery

Disadvantages

- Data synchronization
- User confusion
- Operational overhead

Multi-Cluster Trade-Offs

Advantages

- Failure isolation with user jobs
- Easy capacity planning
- Guarantee SLA’s
- Able to test new versions
- Disaster Recovery

Disadvantages

- **Data synchronization**
- User confusion
- Operational overhead

Agenda

- Data Platform at Airbnb
- Cluster Evolution
- **Incremental Data Replication - ReAir**
- Unified Streaming and Batch Processing - AirStream

Warehouse Replication Approaches

Batch

- Scan HDFS, metastore
- Copy relevant entries
- Simple, no state
- High latency

Incremental

- Record changes in source
- Copy/re-run operations on destination
- More complex, more state
- Low latency (seconds)

Incremental Replication

- Record Changes on Source
- Convert Changes to Replication Primitives
- Run Primitives on the Destination

Record Changes On Source

- Hive provides hooks API to fire at specific points
  - Pre-execute
  - Post-execute
  - Failure
- Use post-execute to log objects that are created into an audit log
- In critical path for queries

Example Audit Log Entry

```
mysql> select * from audit_log where id=31006102 \G
*************************** 1. row ***************************
              id: 31006102
     create_time: 2017-06-30 03:54:38
        query_id: airflow_20170630035353_c0cbbb70-5e32-46f2-8bac-d3462aa42be8
    command_type: QUERY
         command:


    INSERT OVERWRITE TABLE braavos_mtl_v01.mtlv3_summary_v02
    PARTITION(ds = '2017-06-21' , region = 'FL' , country = 'US' , coalition = 'US')
    SELECT date , amount_usd , payin_mtl_impact , payin_non_mtl_impact , payin_unallocated , payin_count , refund_amount_usd ,
 refund_mtl_impact , refund_non_mtl_impact , refund_host_resolution , refund_unallocated , refund_count , payout_amount_usd ,
payout_mtl_impact , payout_mtl_couponed , payout_unallocated , payout_count , collection_amount_usd , collection_mtl_impact ,
collection_unallocated , collection_count , mtl_as_of_balance
    FROM
      hive_compaction_staging.bcadff63f6cbb2a1cd42af177c0134754cb9b099

          inputs: {"tables":["hive_compaction_staging.bcadff63f6cbb2a1cd42af177c0134754cb9b099"]}
         outputs: {"partitions":["braavos_mtl_v01.mtlv3_summary_v02/ds=2017-06-21/region=FL/country=US/coalition=US"]}
        username: airflow
 chronos_job_name: NULL
chronos_job_owner: NULL
    mesos_task_id: NULL
              ip: 10.61.175.198
          extras: NULL
1 row in set (0.00 sec)

mysql>
```

Convert Changes to Primitive Operations

- 3 types of objects - DB, table, partition
- 3 types of operations - Copy, rename, drop
- 9 different primitive operations
- Idempotent

Primitive Example

```
CREATE TABLE srcpart (key STRING) PARTITIONED BY
(ds STRING)
```

- Copy Table

```
INSERT OVERWRITE TABLE srcpart PARTITION(ds='1')
SELECT key FROM src
```

- Copy Partition

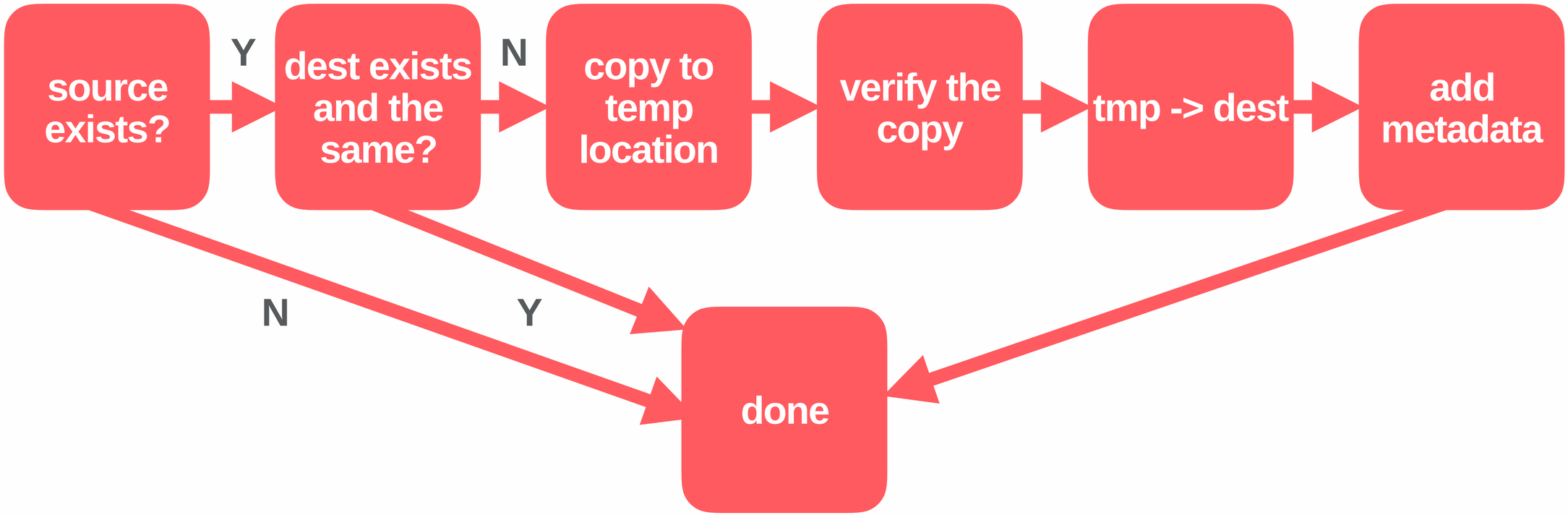
```
ALTER TABLE srcpart SET FILEFORMAT TEXTFILE
```

- Copy Table

```
ALTER TABLE srcpart RENAME to srcpart_old
```

- Rename table

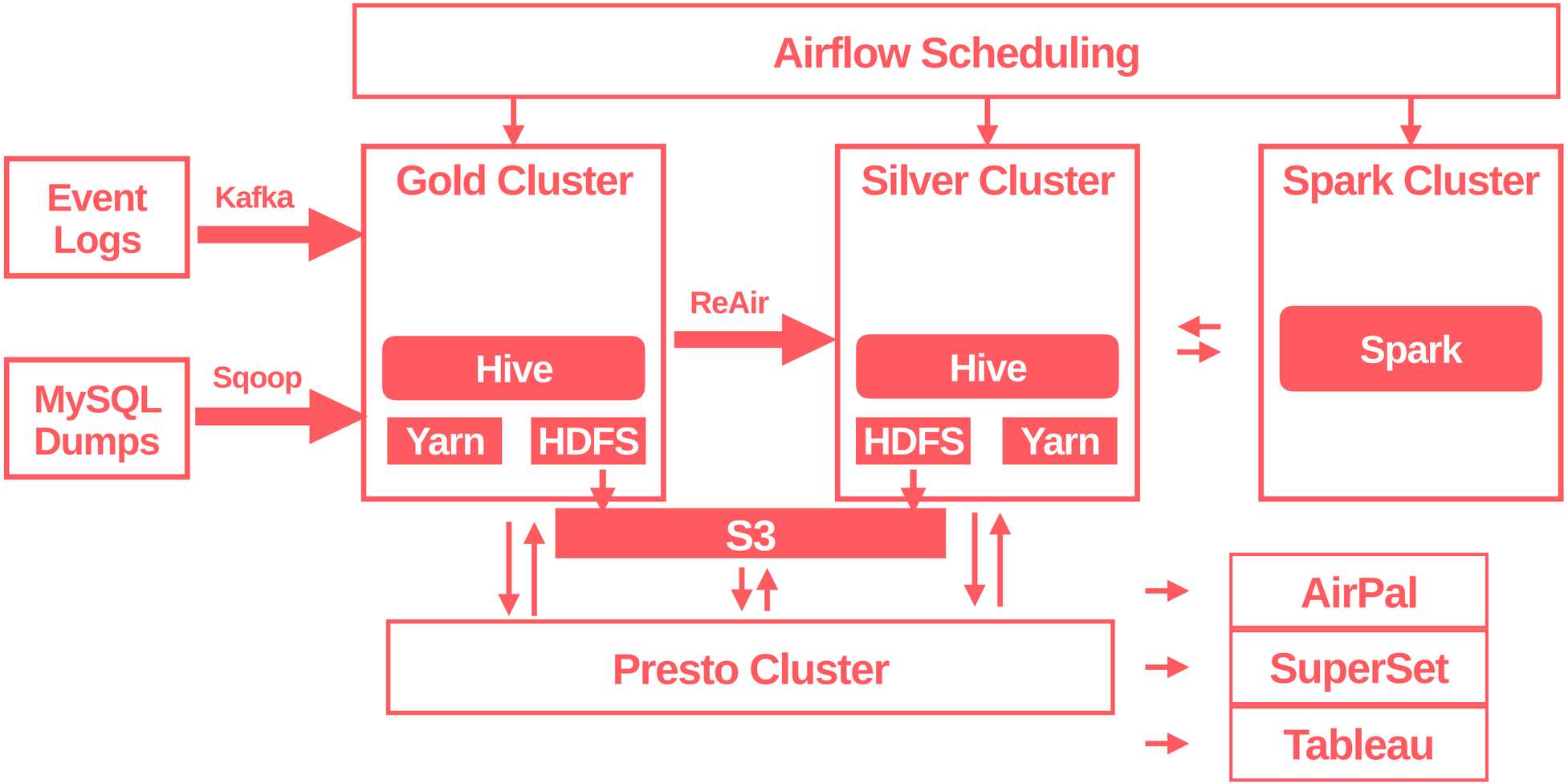
Copy Table Flow



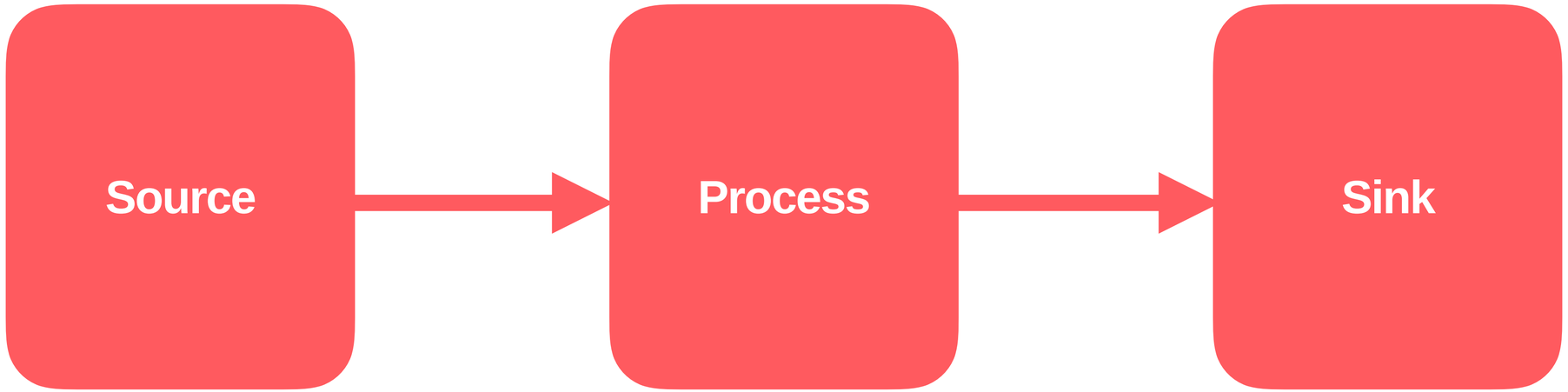
Agenda

- Data Platform at Airbnb
- Cluster Evolution
- Incremental Data Replication - ReAir
- **Unified Streaming and Batch Processing - AirStream**

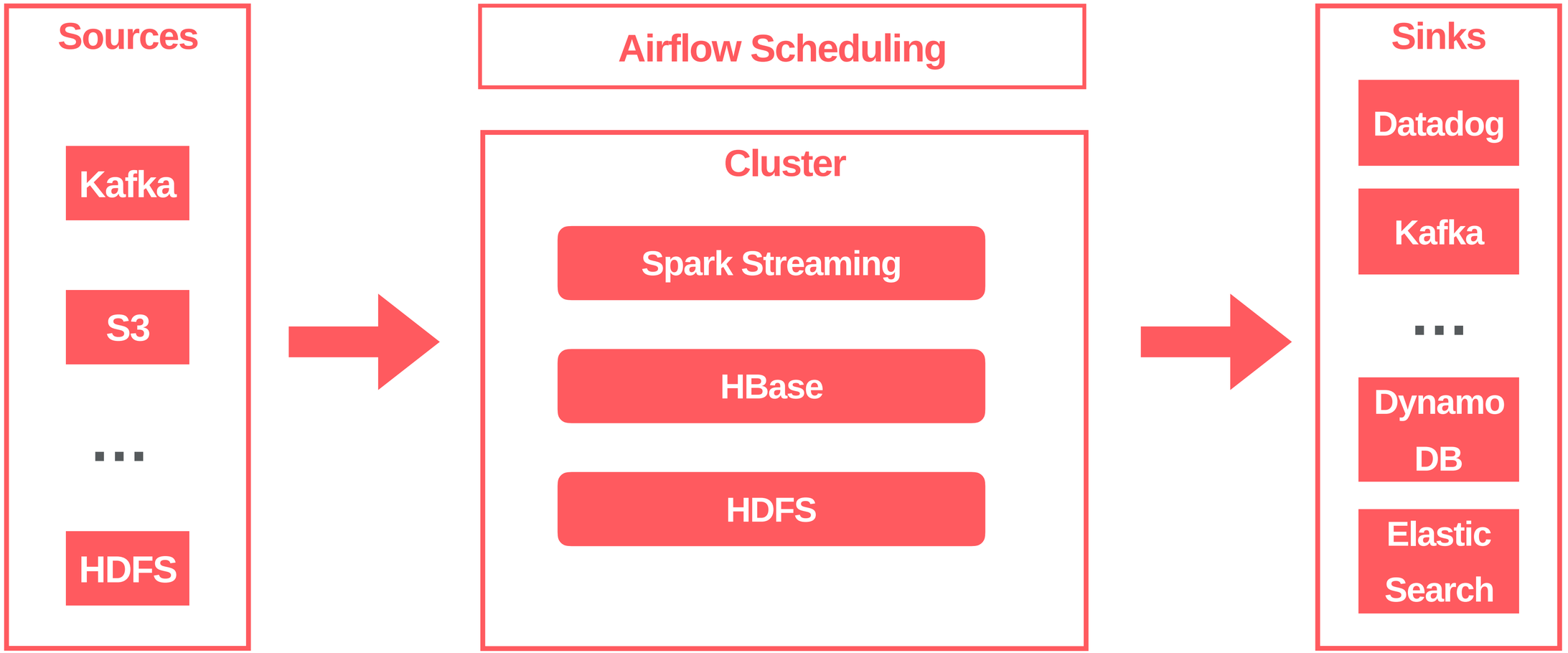
Batch Infrastructure



AirStream

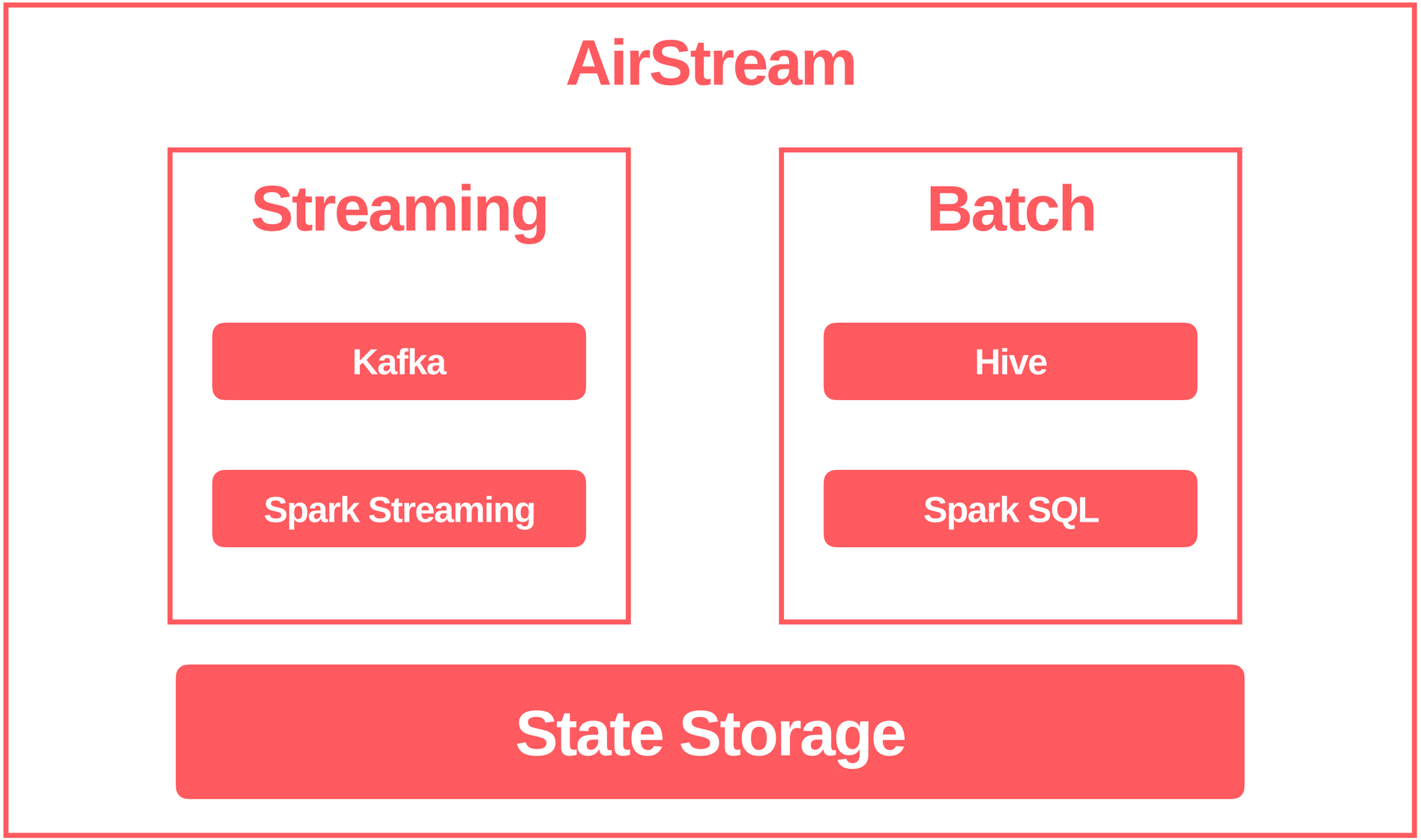


Streaming at Airbnb - AirStream



Lambda Architecture

Lambda Architecture



Sources

Streaming

```
source: [
 {
   name: source_example,
   type: kafka,
   config: {
     topic: "example_topic",
   }
 }
]
```

Batch

```
source: [
 {
   name: source_example,
   type: hive,
   sql: {
    select * from db.table where ds='2017-06-05';
  }
 }
]
```

Computation

Streaming/Batch

```
process: [{
   name = process_example,
   type = sql,
   sql = """
     SELECT  listing_id, checkin_date, context.source as source
     FROM  source_example
     WHERE  user_id IS NOT NULL   """
 }]
```

Sinks

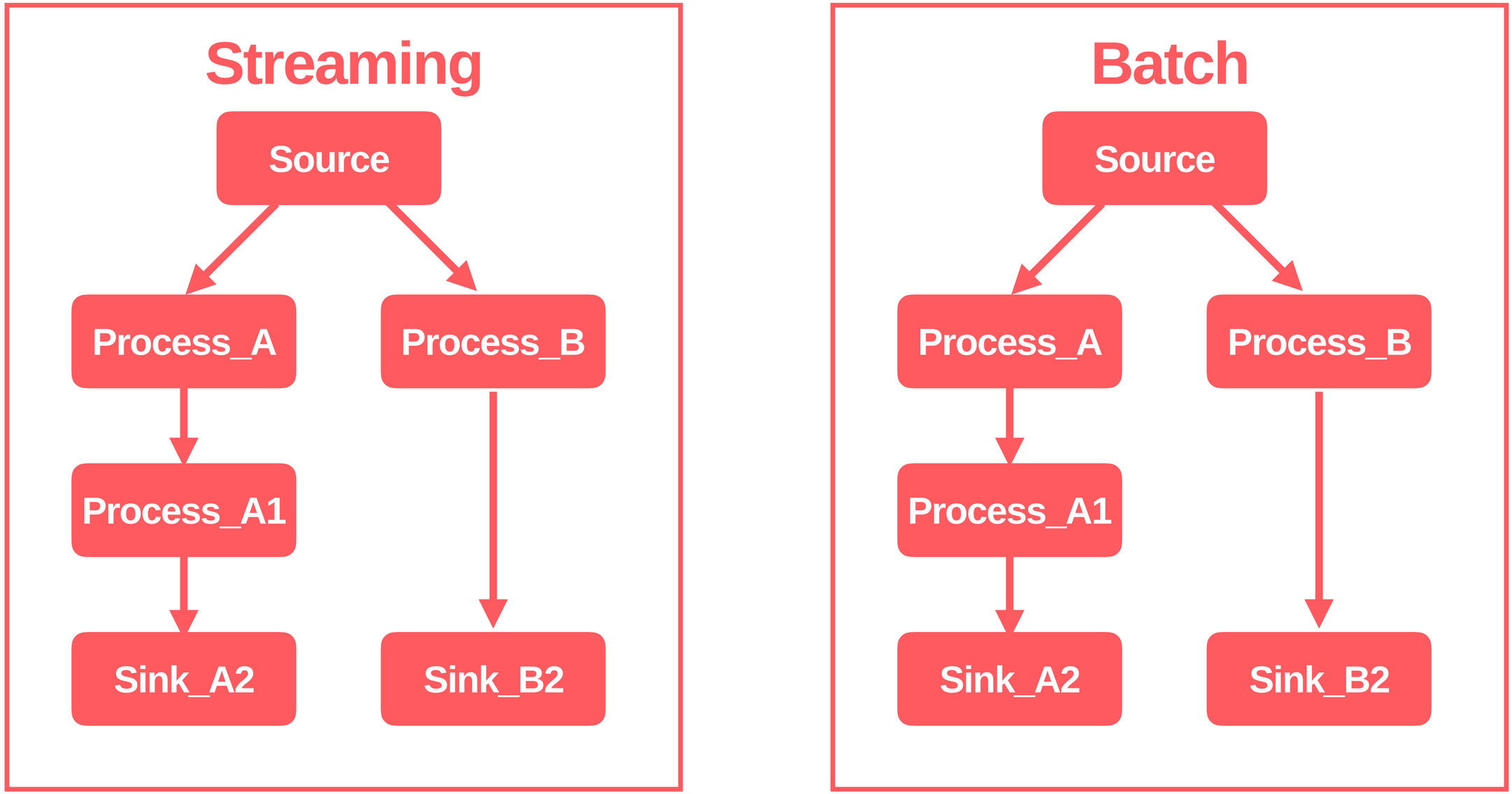
Streaming

```
sink: [
 {
   name = sink_example
   input = process_example
   type = hbase_update
   hbase_table_name = test_table
   bulk_upload = false
 }
]
```

Batch

```
sink: [
 {
   name = sink_example
   input = process_example
   type = hbase_update
   hbase_table_name = test_table
   bulk_upload = true
 }
]
```

Computation Flow

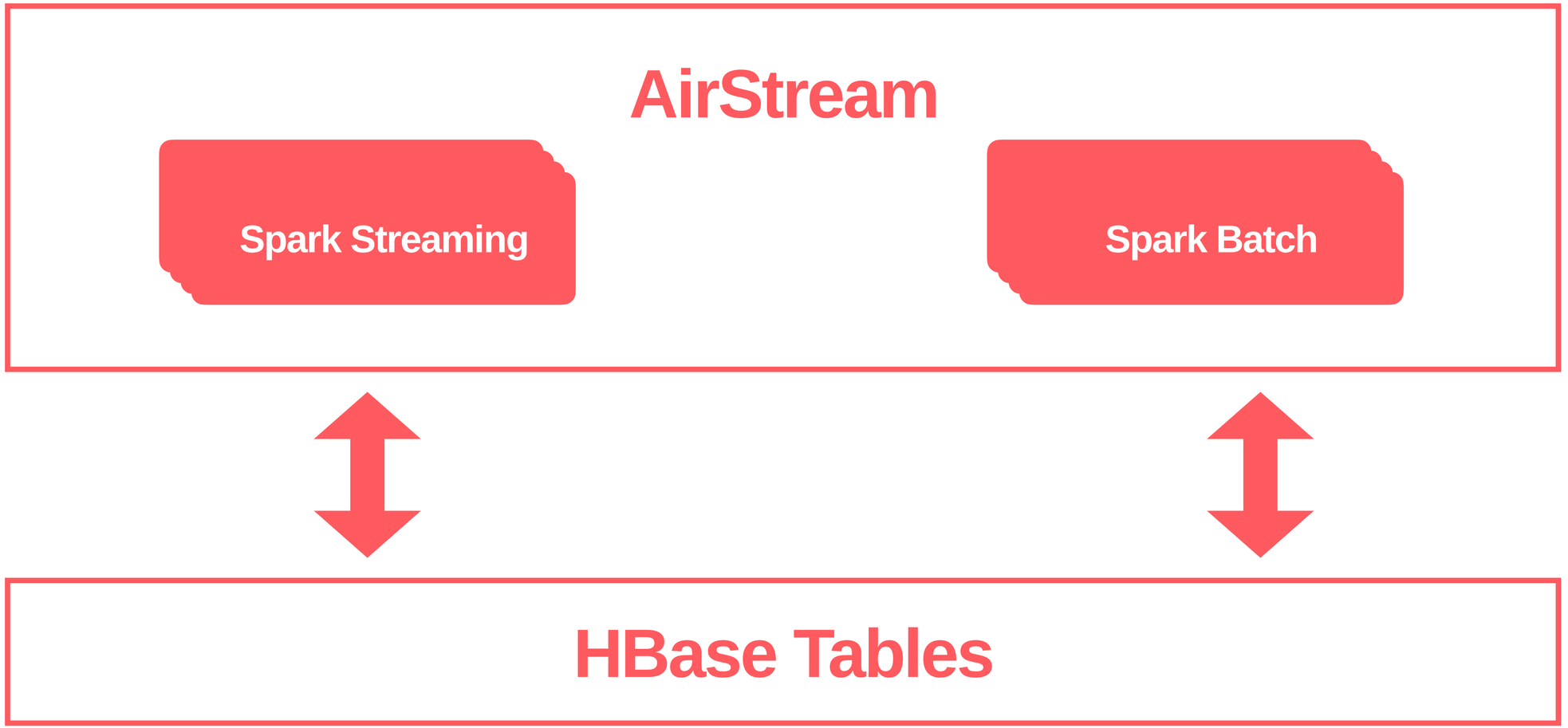


Unified API through AirStream

- Declarative job configuration
- Streaming source vs static source
- Computation operator or sink can be shared by streaming and batch job.
- Computation flow is shared by streaming and batch
- Single driver executes in both streaming and batch mode job

Shared State Storage

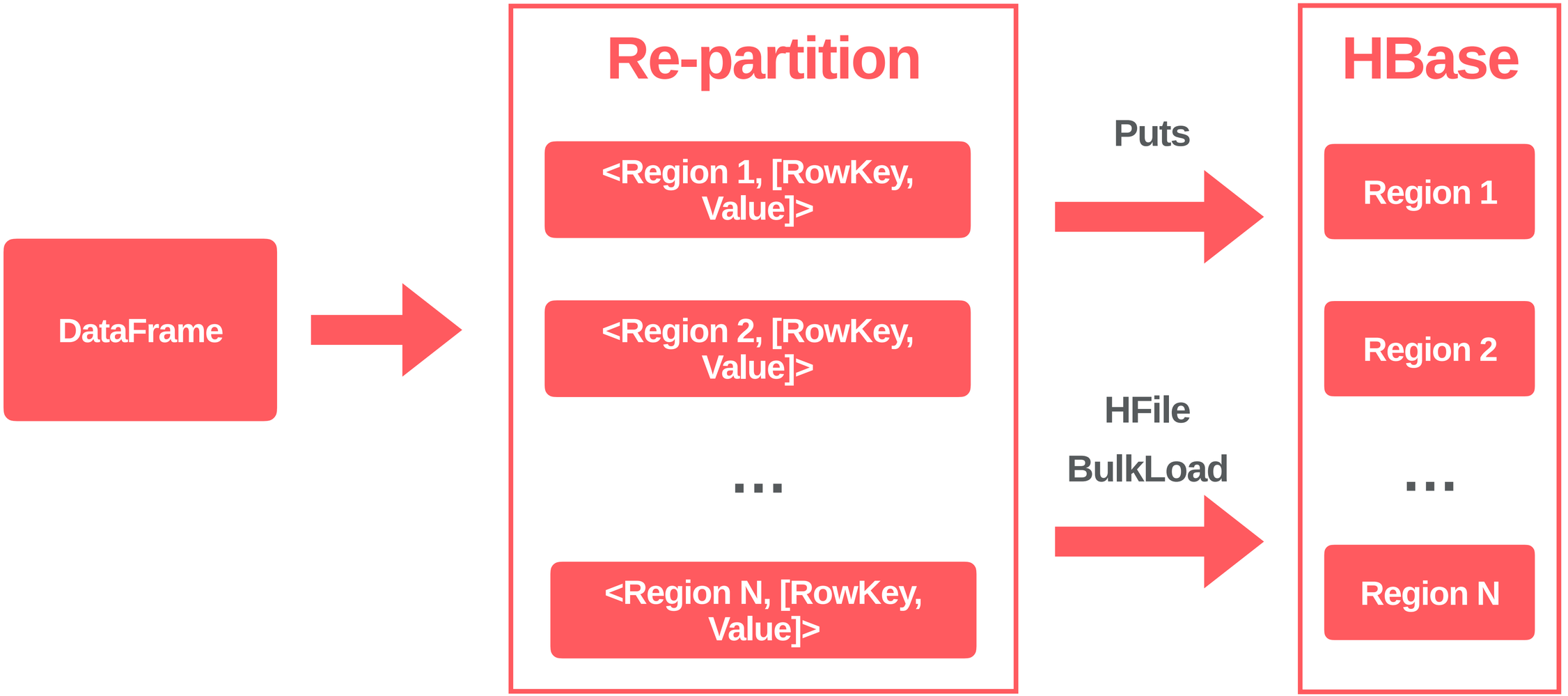
Shared Global State Store



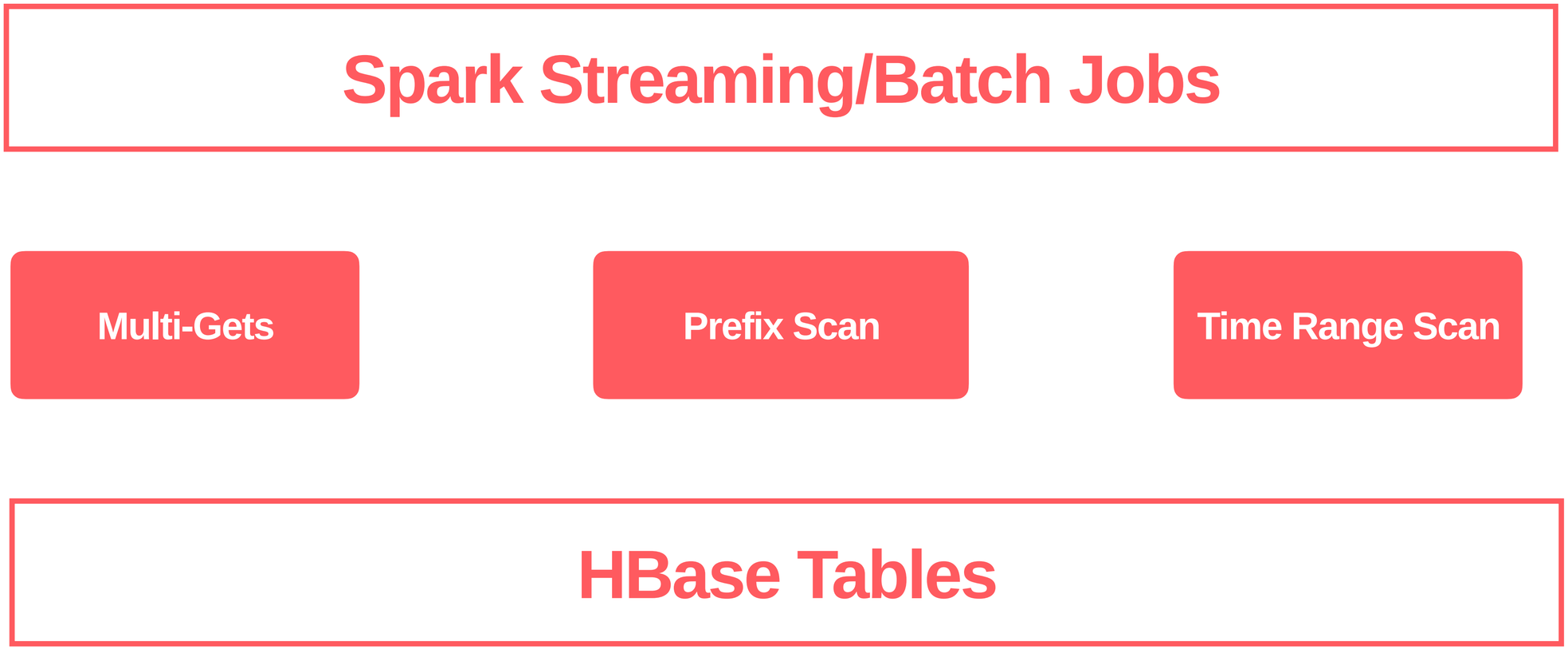
Why HBase

- Well integrated with Hadoop eco system
- Efficient API for streaming writes and bulk uploads
- Rich API for sequential scan and point-lookups
- Merged view based on version

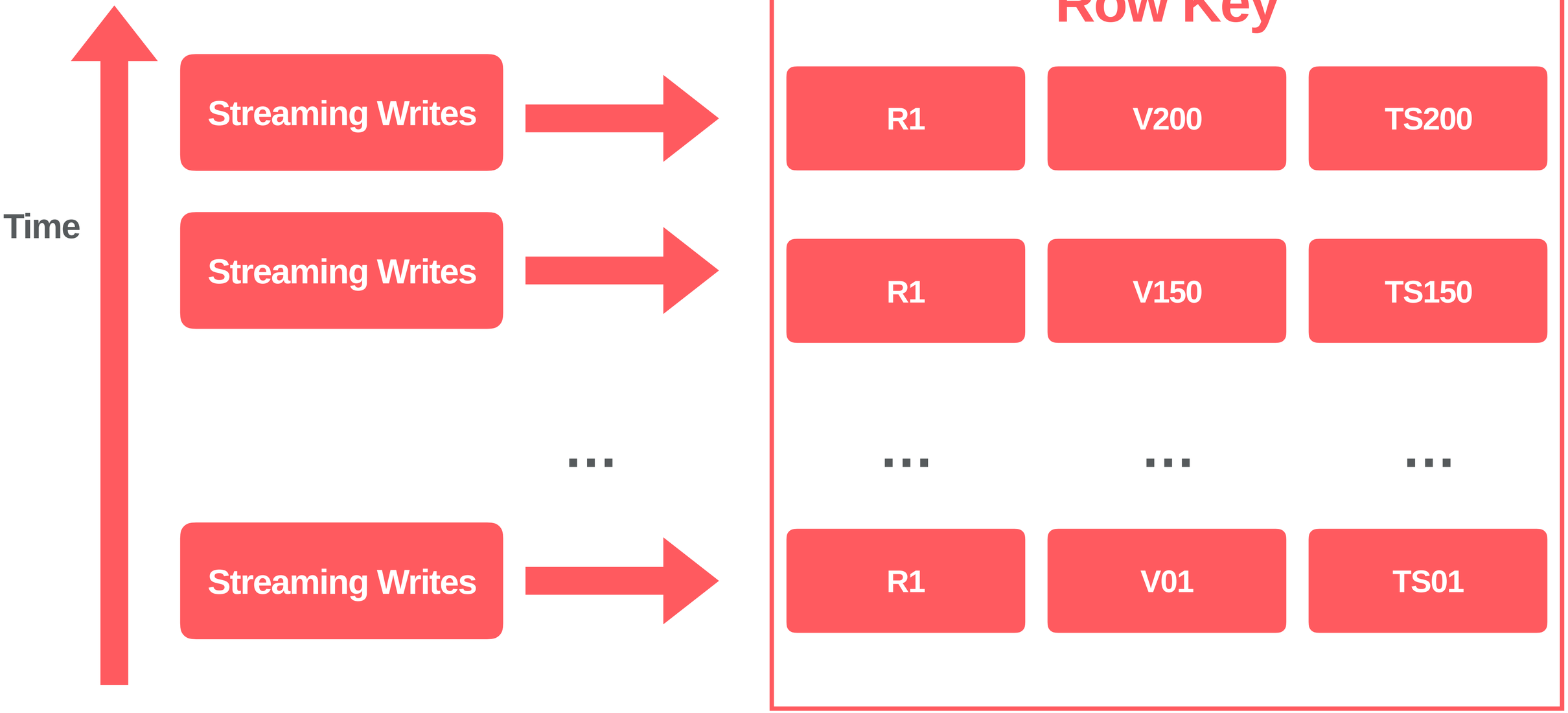
Unified Write API



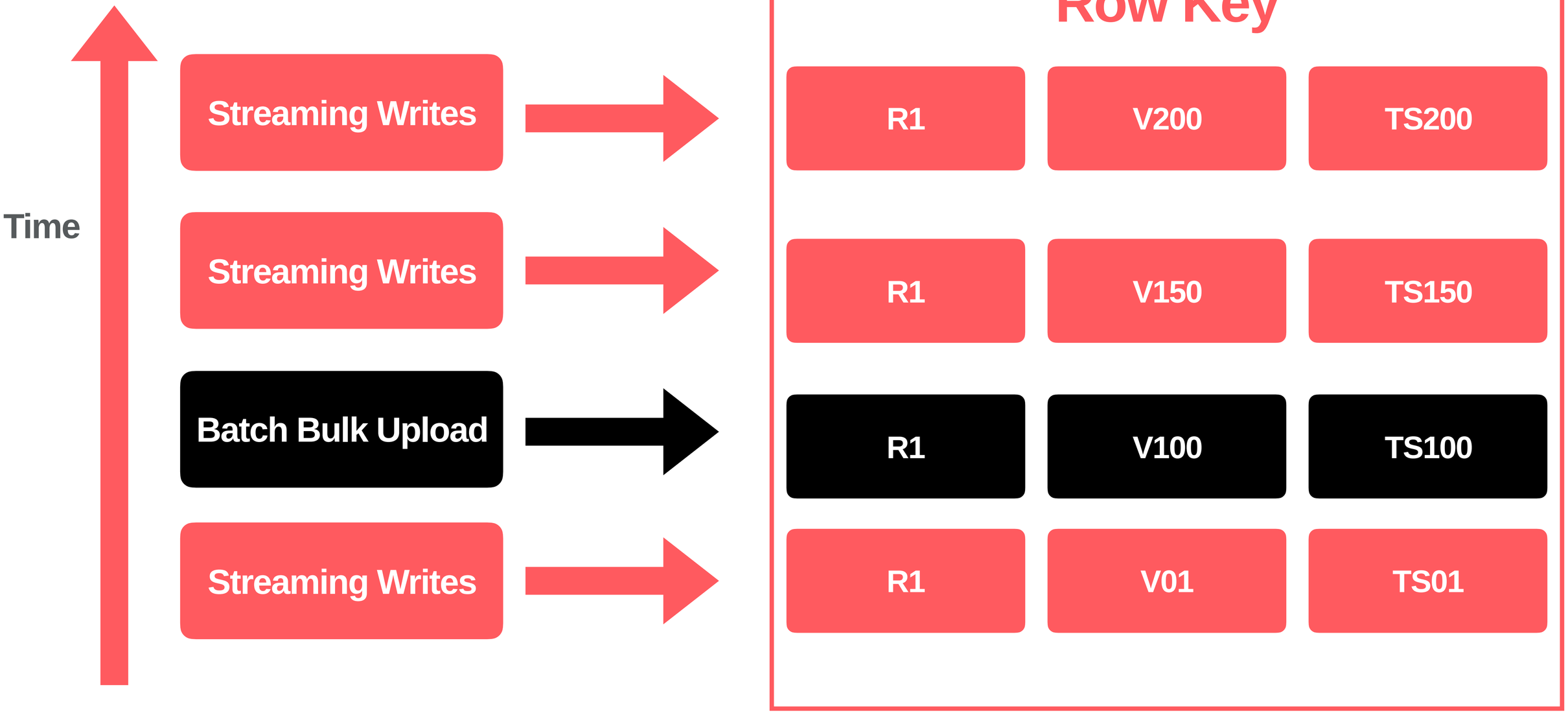
Rich Read API



Merged Views



Merged Views



Our Foundations

- Unify streaming and batch process
- Shared global state store

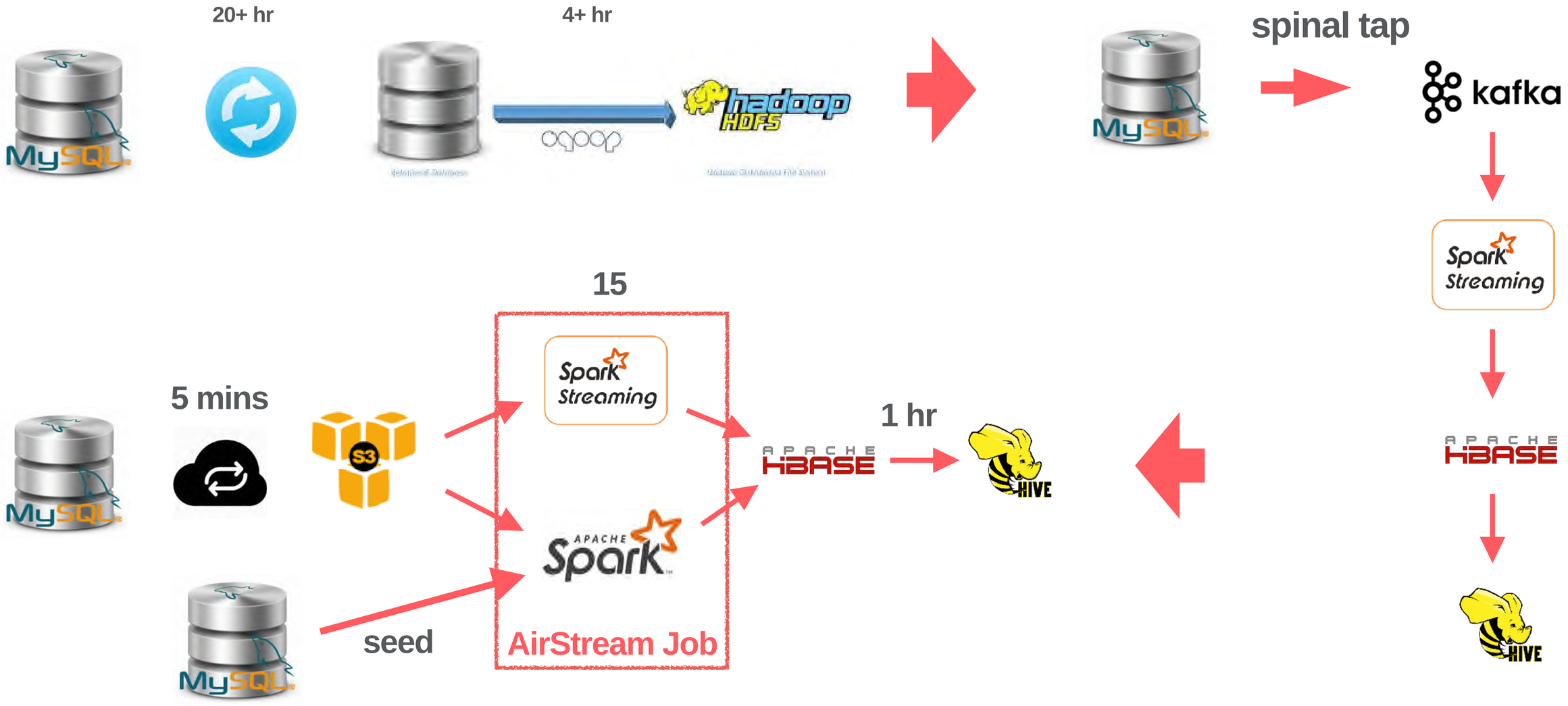
MySQL DB Snapshot Using Binlog Replay

Move Elephant

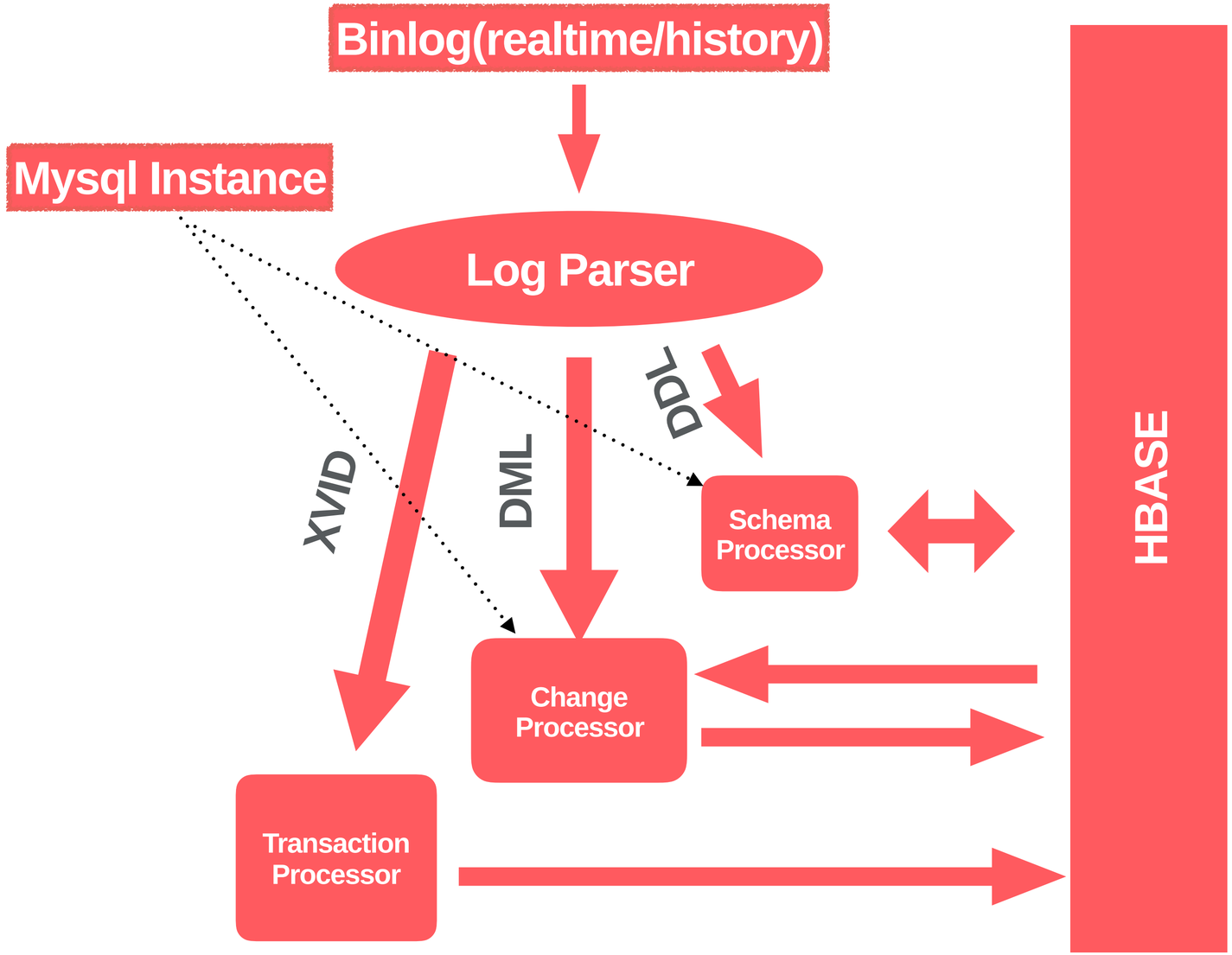
Database Snapshot

- **Large amount of data:** Multiple large mysql DBs
- **Realtime-ness:** minutes delay/ hours delay
- **Transaction :** Need to keep transaction across different tables
- **Schema change:** Table schema evolves

Binlog Replay on Spark



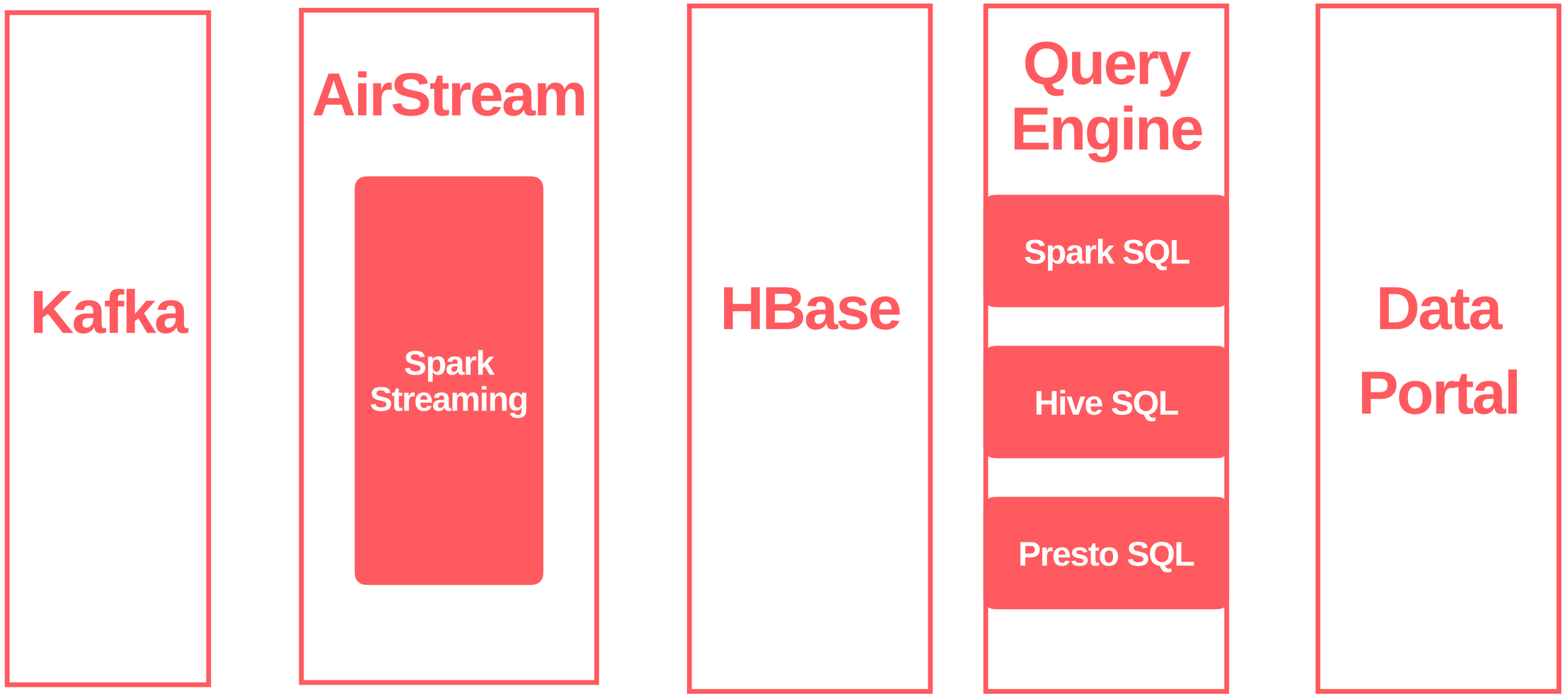
Lambda Architecture



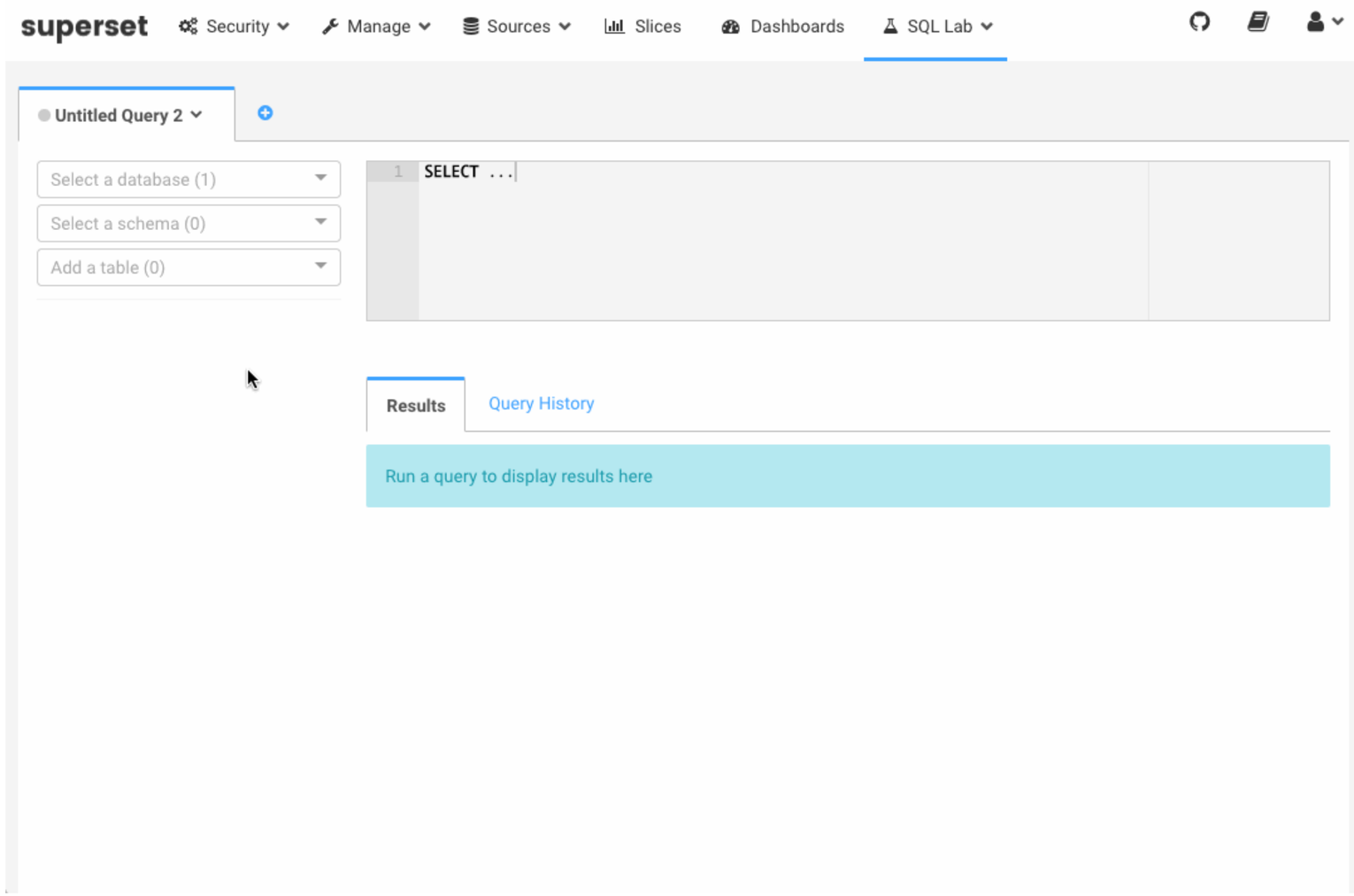
- **Streaming and Batch shares Logic:** Binlog file reader, DDL processor, transaction processor, DML processor.
- **Merged by binlog position:** <filenum, offset>
- **Idempotent:** Log can be replayed multiple times.
- **Schema changes:** Full schema change history.

Streaming Ingestion & Realtime Interactive Query

Realtime Ingestion and Interactive Query



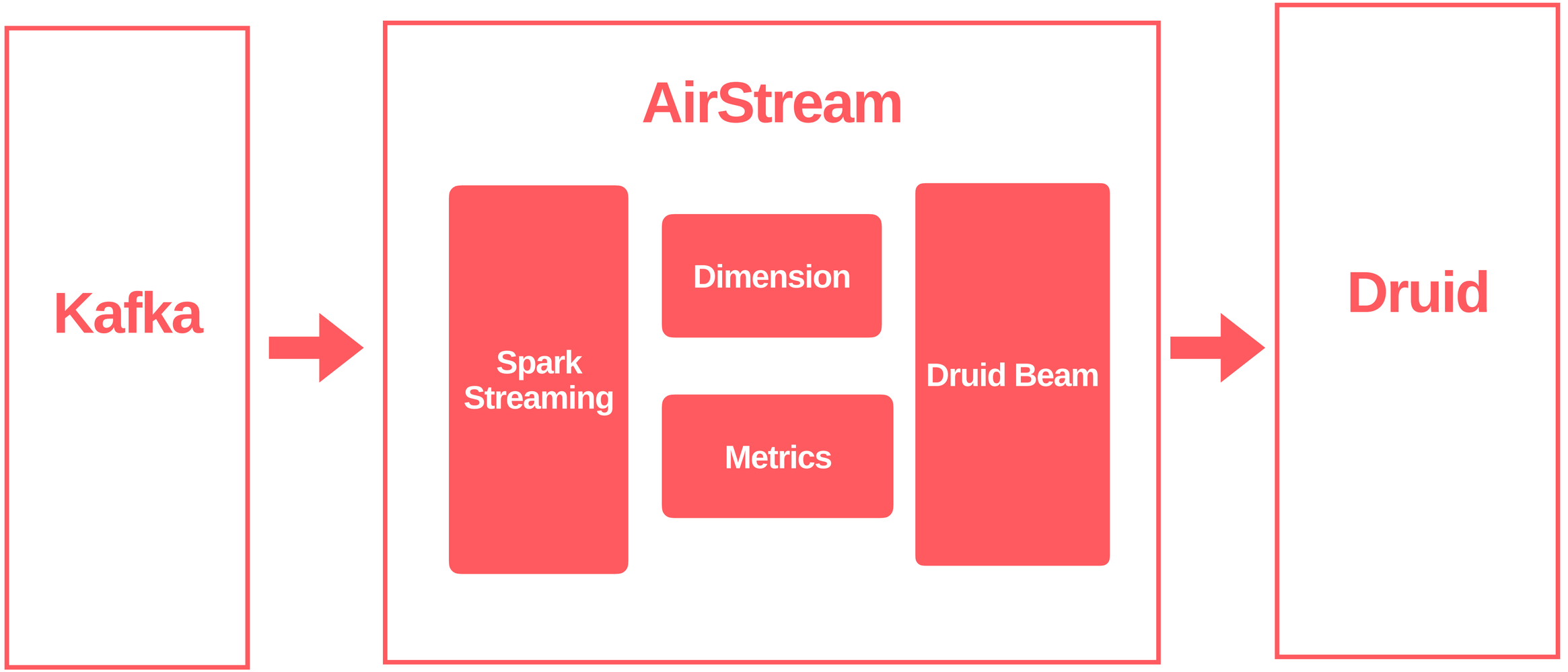
Interactive Query in SqlLab



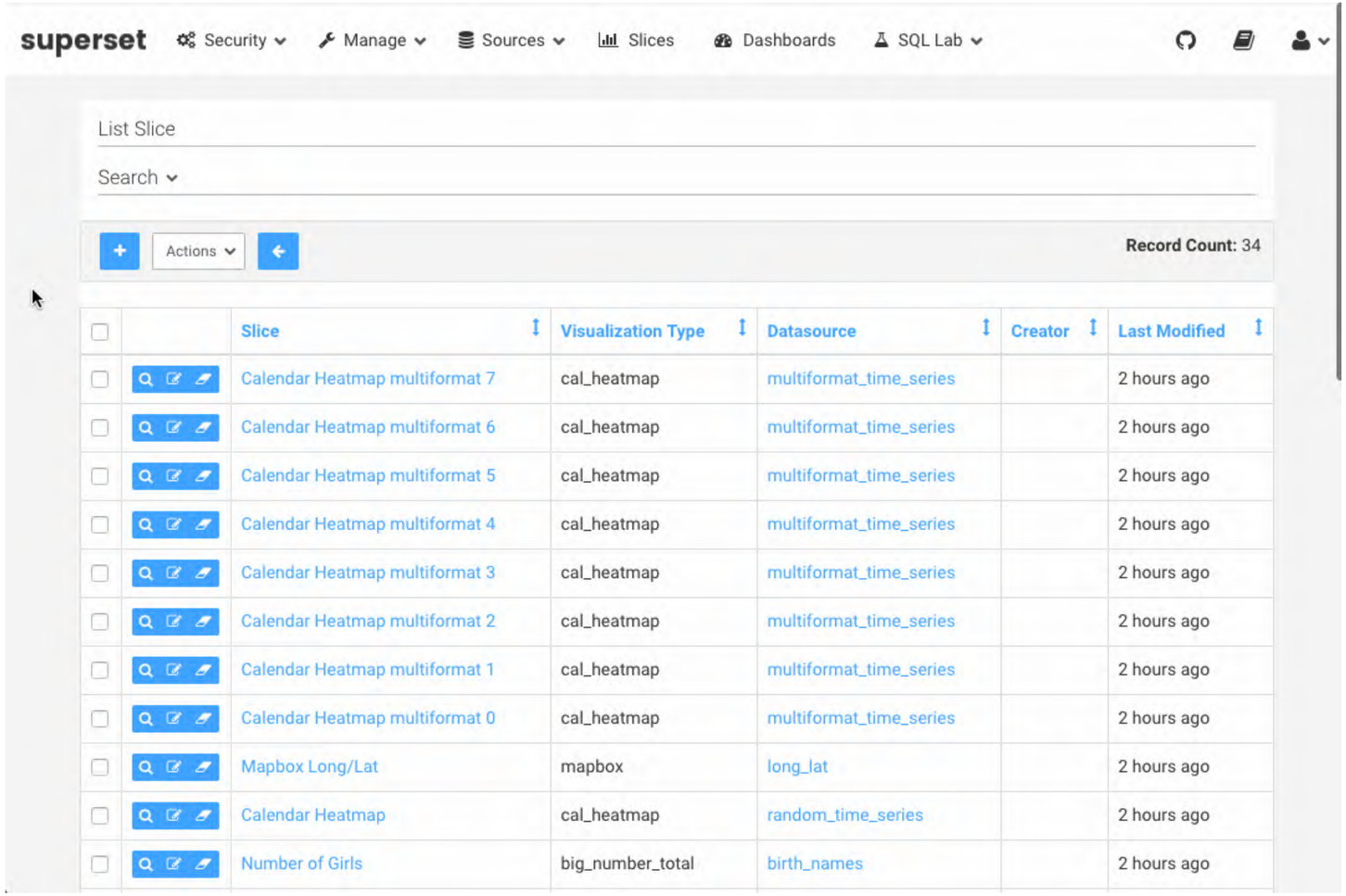
Thanks

Realtime OLAP with Druid

Realtime Ingestion for Druid

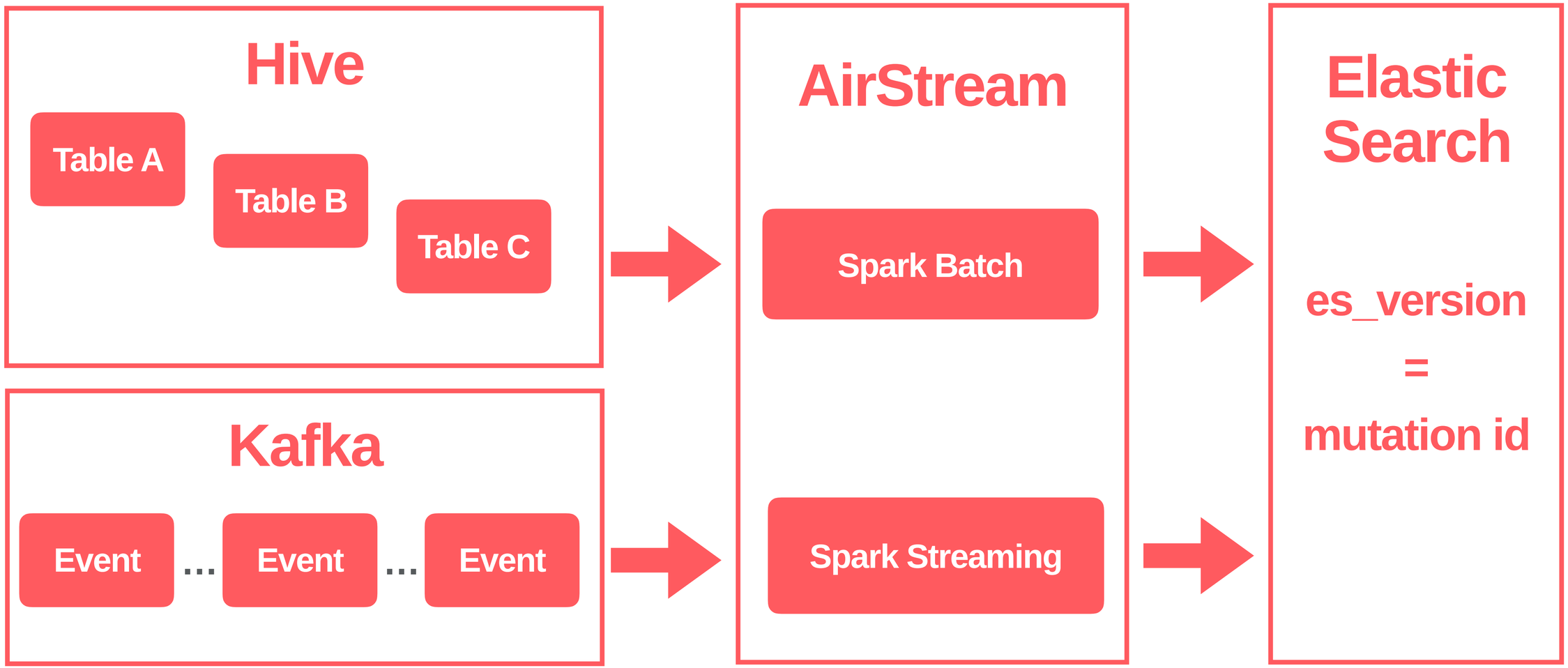


Superset Powered by Druid



Realtime Indexing

Realtime Indexing

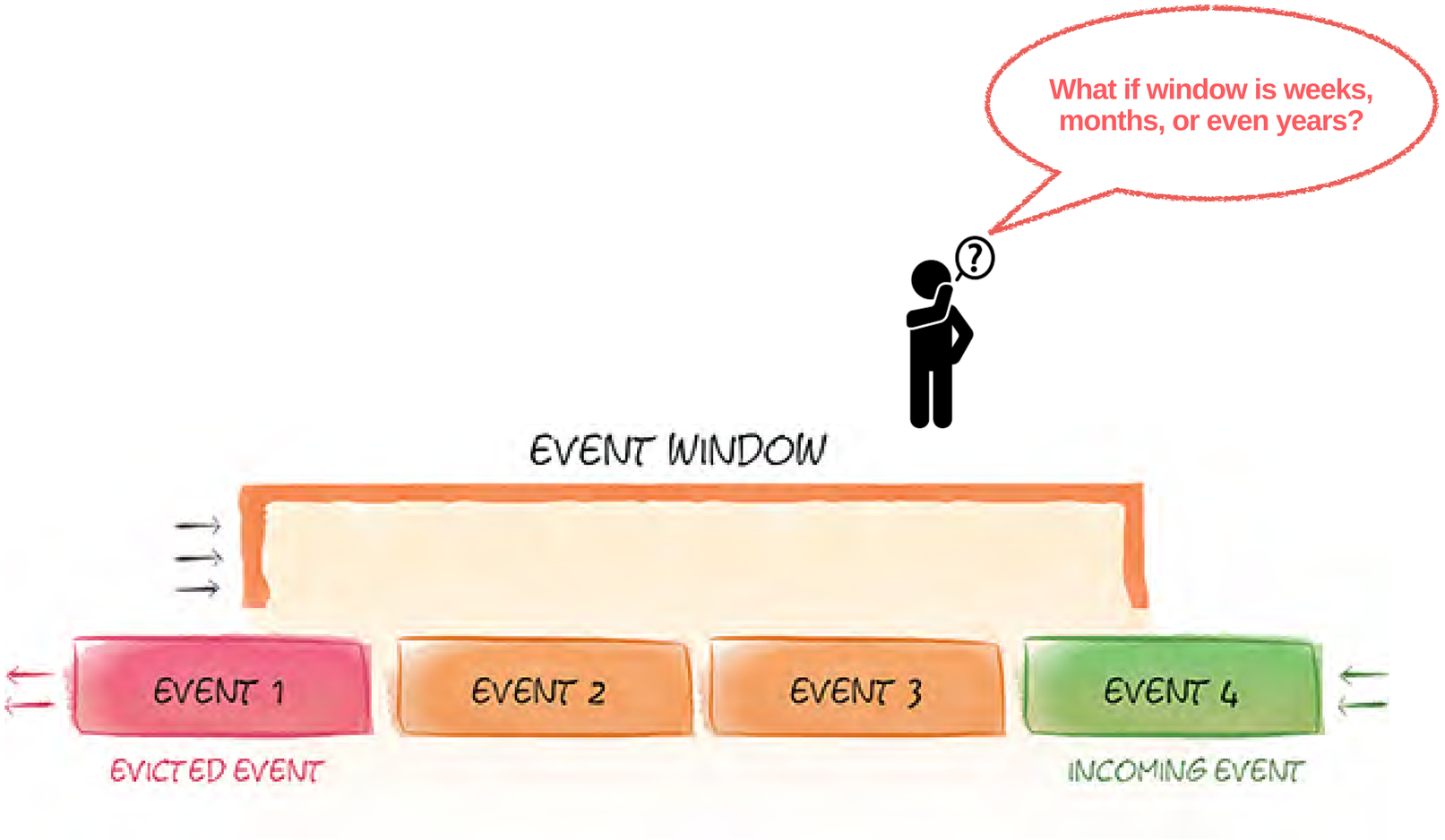


Backup Slides

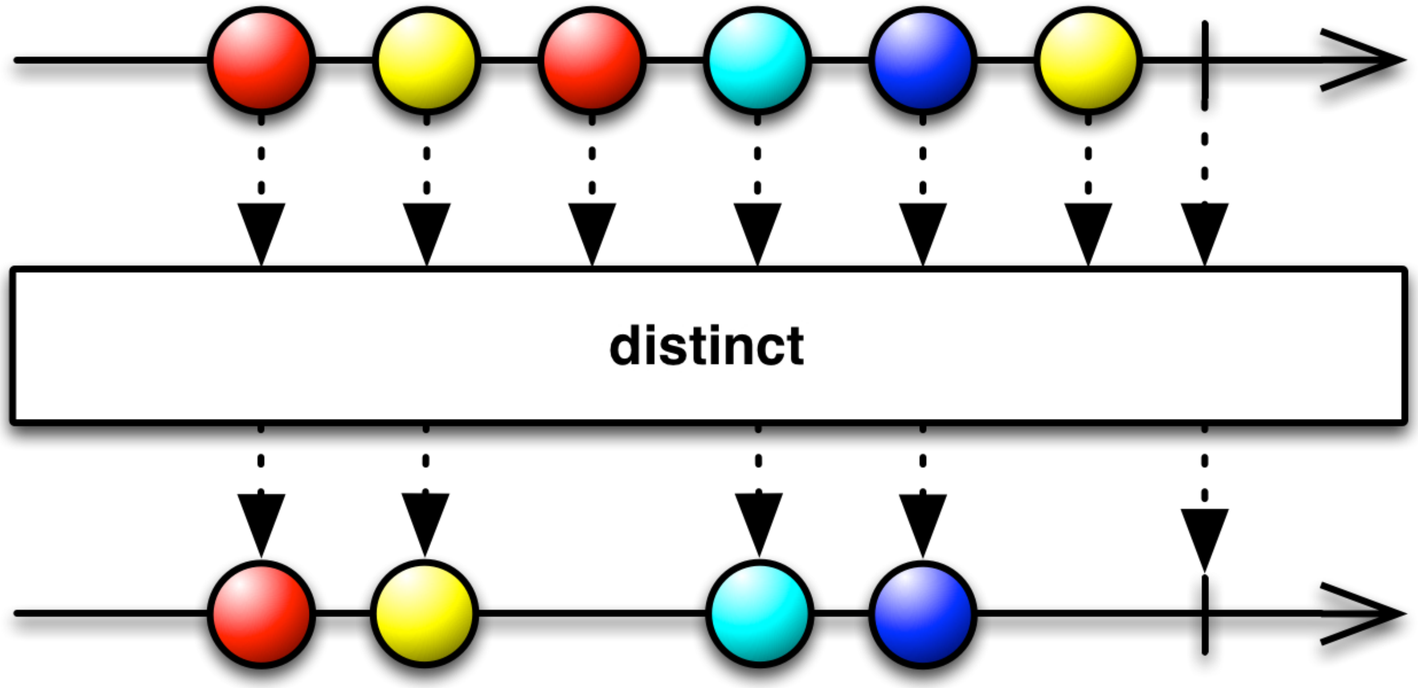
Tips

Moving Window Computation

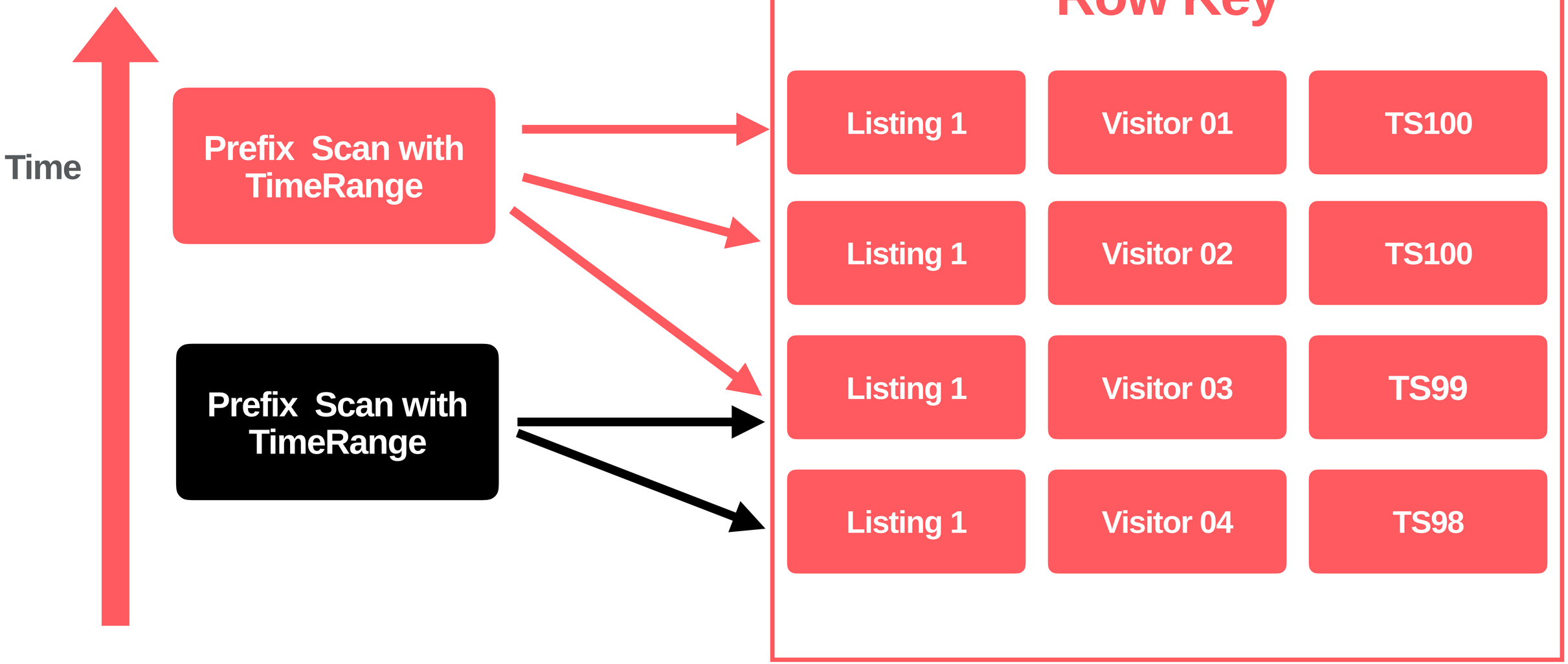
Long Window Computation



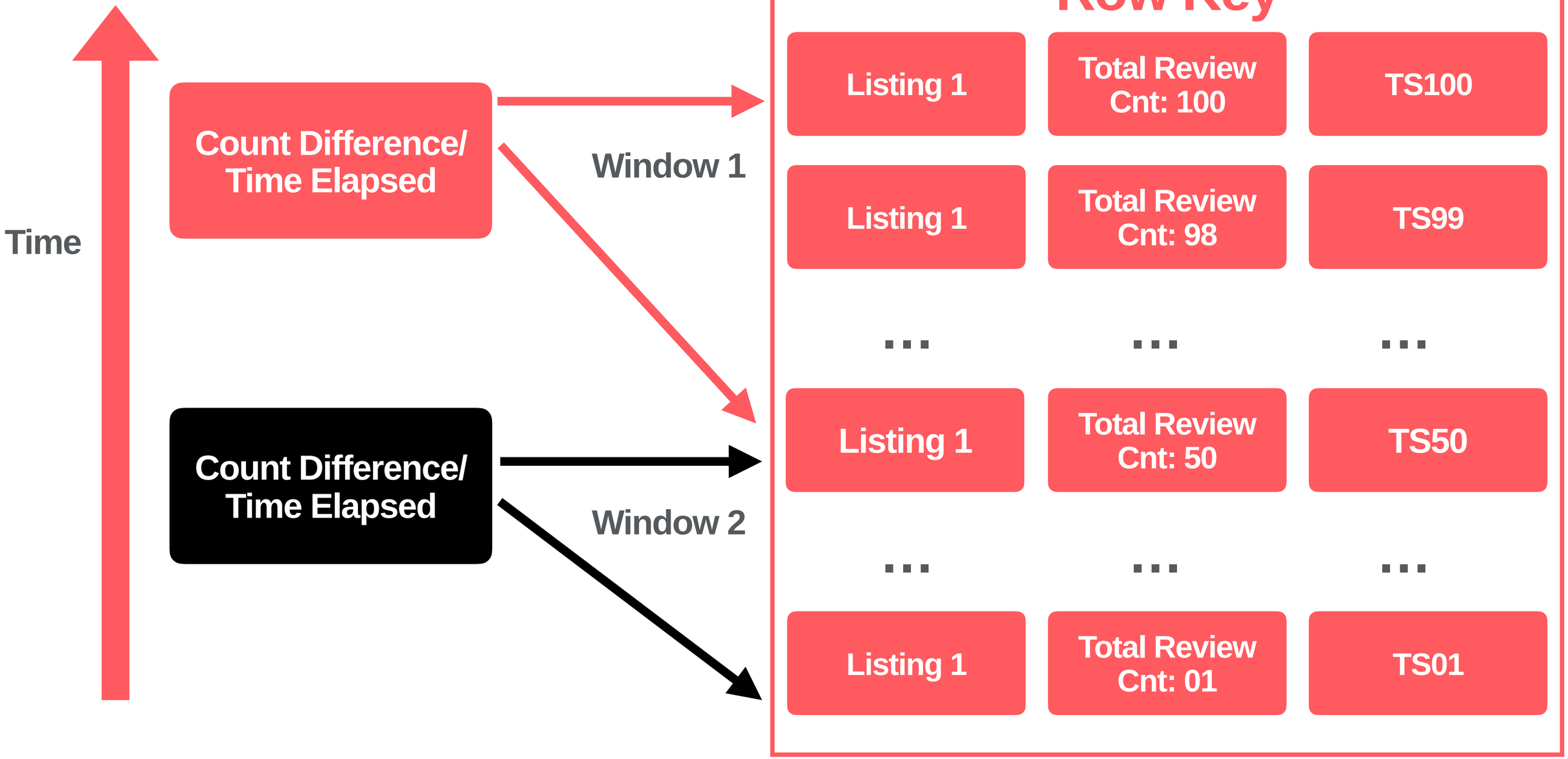
Distinct in a Large Window



Distinct Count

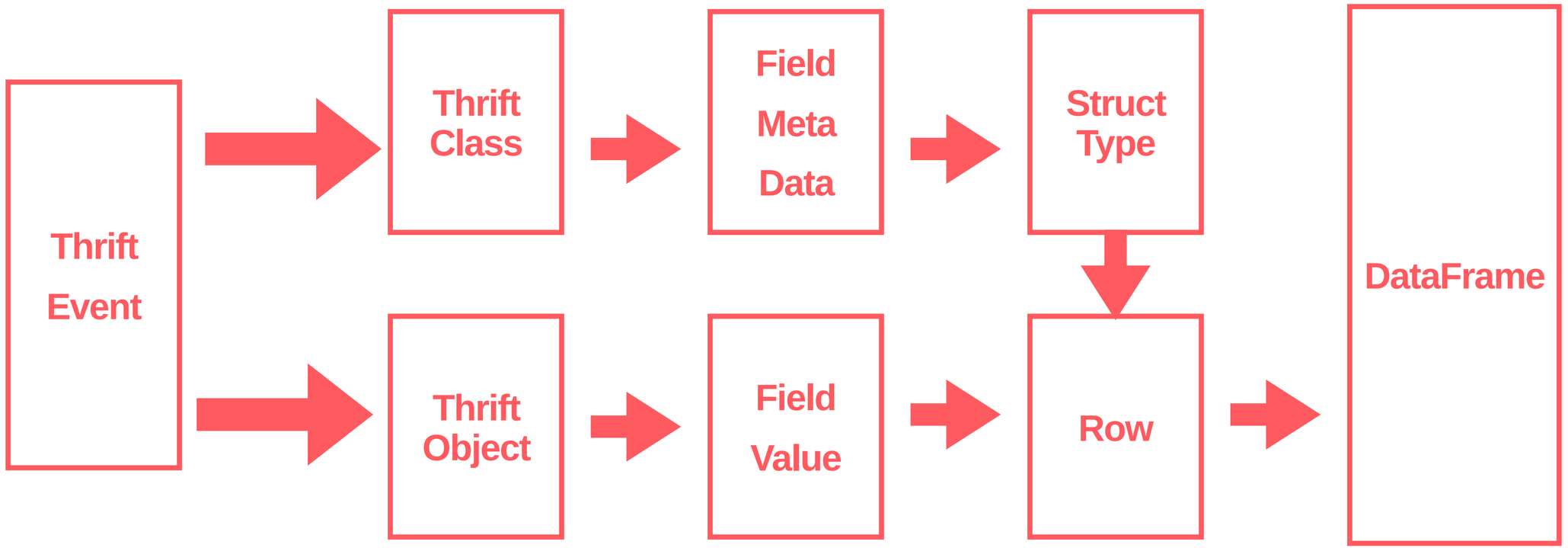


Moving Average



Schema Enforcement Streaming Events

Thrift -> DataFrame



https://github.com/airbnb/airbnb-spark-thrift

Summary

Unify Batch and Streaming Computation

Global State Store Using HBase

Run Primitives on Destination

- Serial execution
  - Easy to reason about operations
  - Very slow
- Parallel execution
  - Fast and scalable
  - Ordering is important: e.g. create table before copying a partition
  - DAG of primitive operations

让创新技术推动社会进步

HELP TO BUILD A BETTER SOCIETY WITH
INNOVATIVE TECHNOLOGIES

Geekbang.

极客邦科技

InfoQ

专注中高端技术人员的技术媒体

EGO EXTRA GEEKS' ORGANIZATION NETWORKS

高端技术人员学习型社交平台

StuQ 斯达克学院

实践驱动的 IT 教育平台

地址：北京市朝阳区洛娃大厦C座8层1801室　　网址：www.geekbang.org