Apache Flink

Production-Ready Flink and Hive Integration

- what story you can tell now

Bowen Li, Committer@Flink

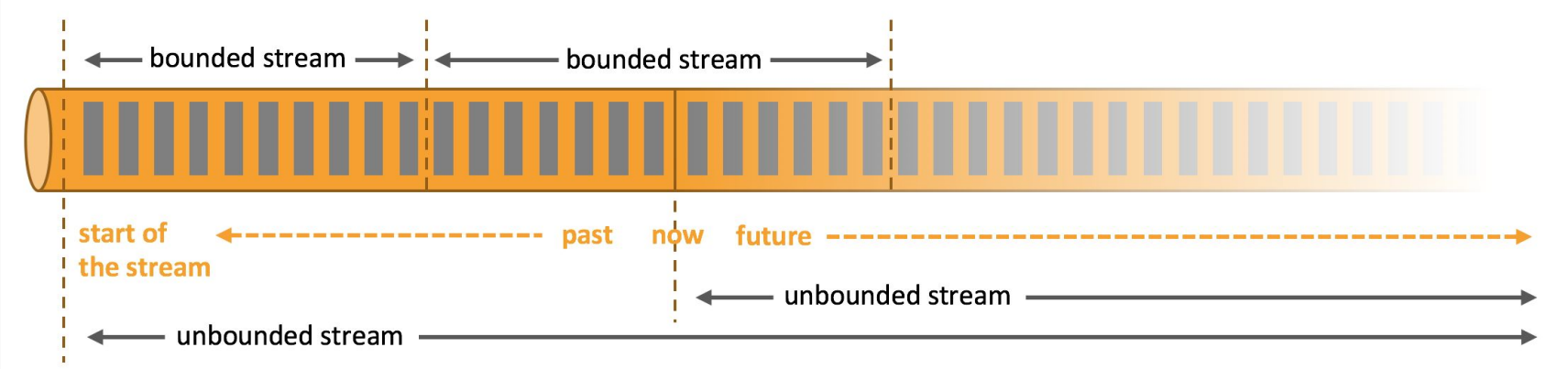
Flink Forward SF, April 2020

Apache Flink

Agenda

- Background
- Flink 1.10 - State of Union
- Flink 1.11 - Upcoming Features
- Stories You’d be able to tell now
- Q&A

Background



- Apache Flink is a framework and distributed processing engine for stateful computations over **unbounded (streaming)** and **bounded (batch)** data streams.

- Flink’s philosophy: Batch is a special case of Streaming

Apache Flink

Background

We annouced Flink + Hive integration as beta in Flink 1.9 to aim at

- Help to shift businesses to more real-time fashion
- Bring Flink’s cutting-edge Streaming capabilities to Batch
- Unify companies’ tech stack of online and offline data infrastructure
- Simplify deployments and operations
- Lower entry bar and learning cost for end users
  - developers, data scientists, analysts, etc

Apache Flink

Flink 1.10 - State of the Union

more on https://ci.apache.org/projects/flink/flink-docs-stable/dev/table/hive/

Apache Flink

Unified Metadata Management

Background

- Hive Metastore (HMS) has become the de facto metadata hub in the Hadoop
- Many companies have a single HMS to manage all of their metadata, either Hive or non-Hive metadata, as the single source of truth

Apache Flink

Unified Metadata Management

Flink’s HiveCatalog

- introduced in Flink 1.9
- production-ready in Flink 1.10
  - compatible with almost all HMS versions
  - can store Flink’s own metadata of tables, views, UDFs, statistics in HMS
  - also being the bridge for Flink to leverage Hive’s own metadata

Apache Flink

HiveCatalog E2E Example

Example to store a Flink's Kafka table thru HiveCatalog into HMS and later query the table with Flink SQL

Apache Flink

HiveCatalog E2E Example

Step 1: Configure the HiveCatalog in Flink SQL CLI

```
execution:
    planner: blink
    type: streaming
    ...
    current-catalog: myhive
    current-database: mydatabase

catalogs:
   - name: myhive
     type: hive
     hive-conf-dir: /opt/hive-conf  # contains hive-site.xml
```

Apache Flink

HiveCatalog E2E Example

Step 2: Create a Kafka table in HMS from Flink SQL CLI

```
Flink SQL> CREATE TABLE mykafka (name String, age Int) WITH (
   'connector.type' = 'kafka',
   'connector.version' = 'universal',
   'connector.topic' = 'test',
   'connector.properties.zookeeper.connect' = 'localhost:2181',
   'connector.properties.bootstrap.servers' = 'localhost:9092',
   'format.type' = 'csv',
   'update-mode' = 'append'
);
[INFO] Table has been created.
```

Apache Flink

HiveCatalog E2E Example

Step 3: Describe the Kafka table in HMS via Hive CLI

```
hive> describe formatted mykafka;
Database:               default
Table Parameters:
    flink.connector.properties.bootstrap.servers localhost:9092
    flink.connector.properties.zookeeper.connect localhost:2181
    flink.connector.topic  test
    flink.connector.type   kafka
    flink.format.type      csv
    flink.generic.table.schema.0.data-type VARCHAR(2147483647)
    flink.generic.table.schema.0.name         name
    flink.generic.table.schema.1.data-type INT
    flink.generic.table.schema.1.name         age
    ....
```

Apache Flink

HiveCatalog E2E Example

Step 4: Query the Kafka table in Flink SQL CLI

```
Flink SQL> select * from mykafka;
```

```
SQL Query Result (Table)
 Refresh: 1 s    Page: Last of 1
```

name	age
tom	15
john	21
kitty	30
amy	24
kaiky	18

Apache Flink

Supported Hive Versions

- Flink 1.9 - Hive 2.3.4 and 1.2.1
- Flink 1.10 - Most Hive versions include
  - 1.0, 1.1, 1.2
  - 2.0, 2.1, 2.2, 2.3
  - 3.1

Apache Flink

Reuse Hive Functions

- Flink 1.9: Supports reusing all Hive UDFs
- Flink 1.10: Support reusing all Hive **built-in functions**
  - Hive has developed a few hundreds of super handy built-in functions over the years
  - supported via Module mechanism, introduced in 1.10 to improve Flink SQL's extensibility
    - e.g. lay the road to custom data types in, say, geo and machine learning
  - provided by **HiveModule**

Apache Flink

Enhanced Read of Hive Data

- Flink 1.9
  - read non-partitioned and partitioned Hive tables
- Flink 1.10
  - read Hive views
  - vectorized reader of ORC files
  - enhanced optimizations:
    - partition-pruning
    - projection pushdown
    - limit pushdown

Apache Flink

Enhanced Write of Hive Data

- Flink 1.9
  - write to non-partitioned Hive tables
- Flink 1.10
  - write to partitioned tables, both static and dynamic partitions
  - support “INSERT OVERWRITE” syntax

Apache Flink

What's coming in Flink 1.11

- Hive near-real-time streaming sink
  - bring a bit more real-time experience to Hive
- Native Parquet reader for better performance
- Better interoperabilities
  - How about creating Hive tables, views, functions within Flink? :D

Apache Flink

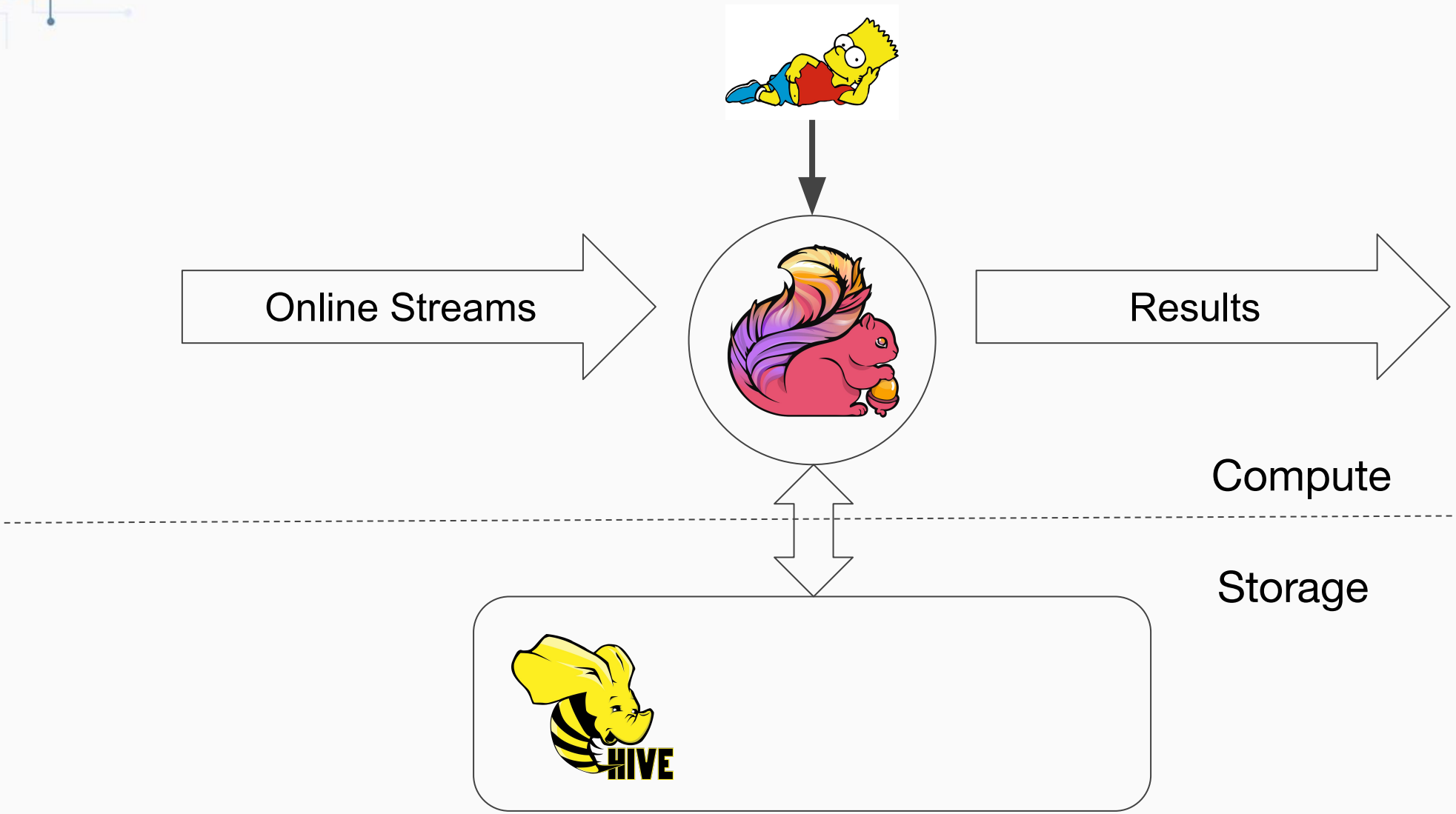
What's coming in Flink 1.11

- Better out-of-box dependency management
  - Yes, we are fully aware of the mess of Hive's dependencies
- JDBC Driver/Gateway so users can reuse existing tools to run SQL jobs on Flink
- Support a bit more Hive syntax

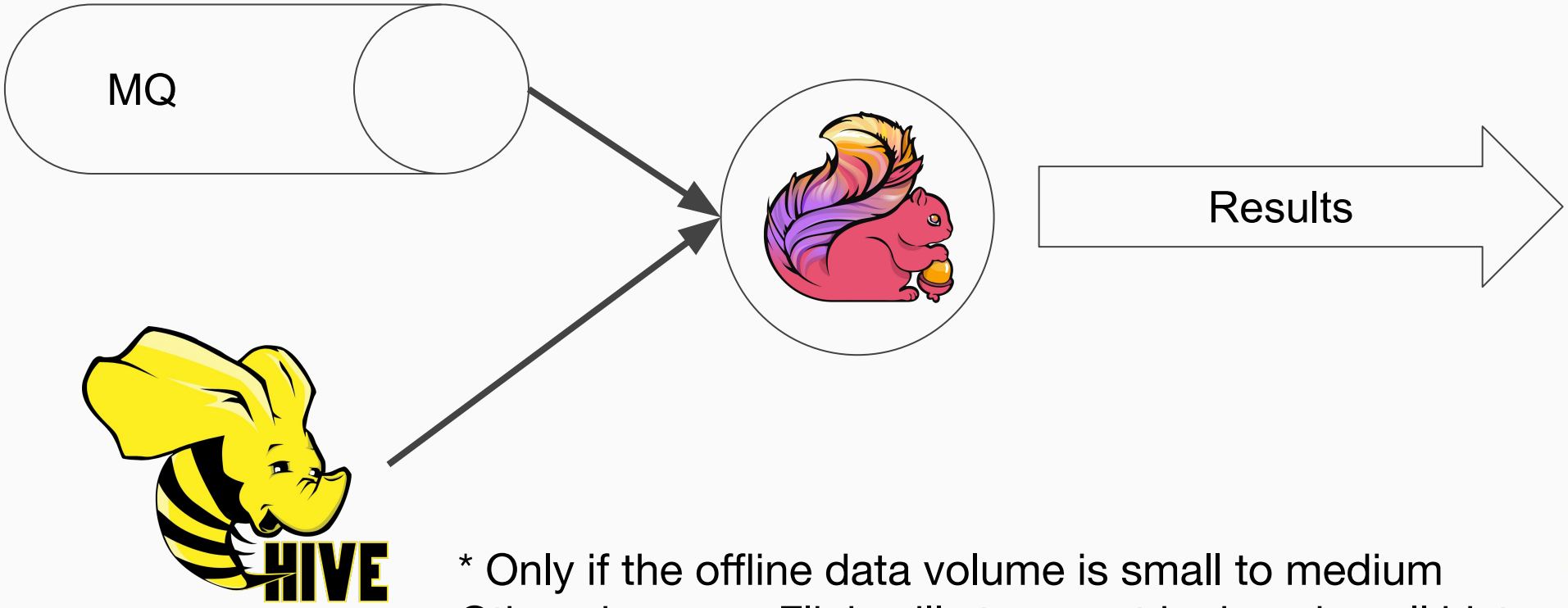
Apache Flink

What use cases do these Flink capabilities enable?

Story 1 - Unify Compute Infrastructure



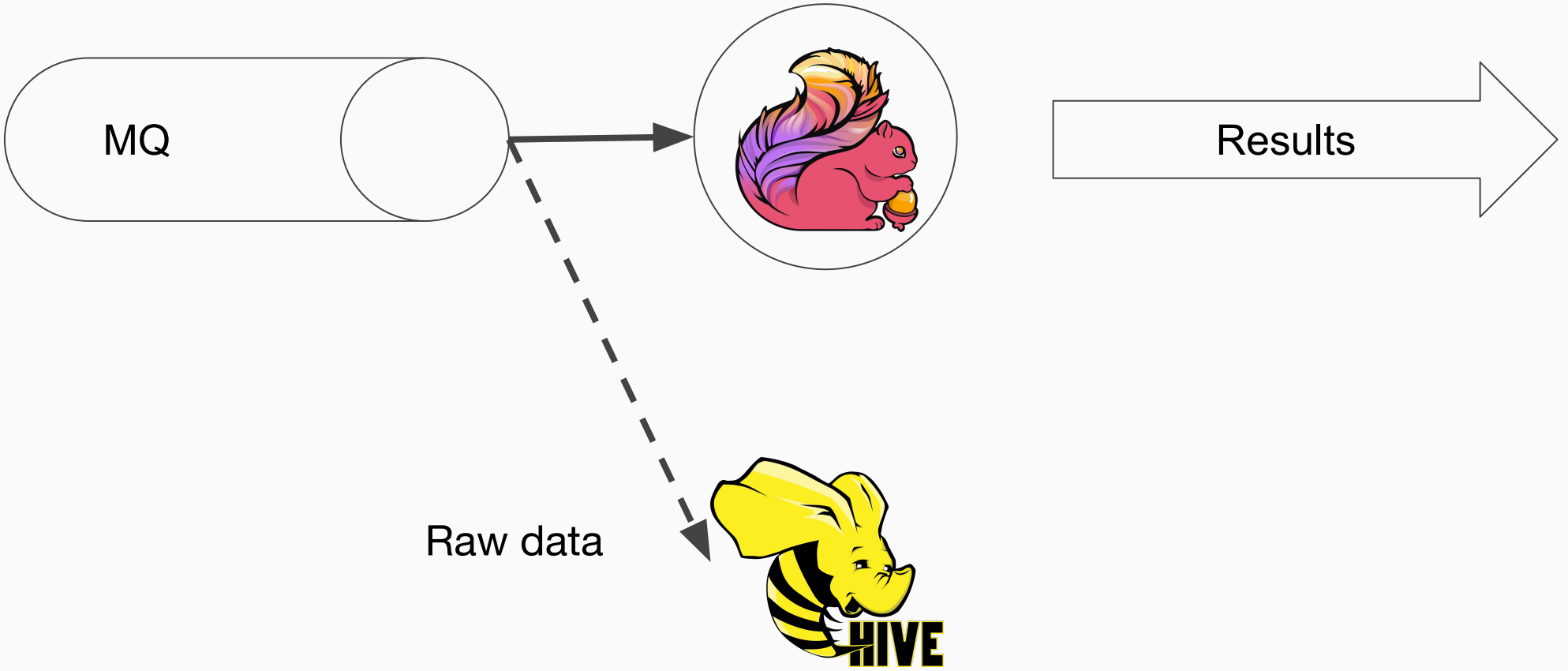
Story 2 - Join real-time data with offline data



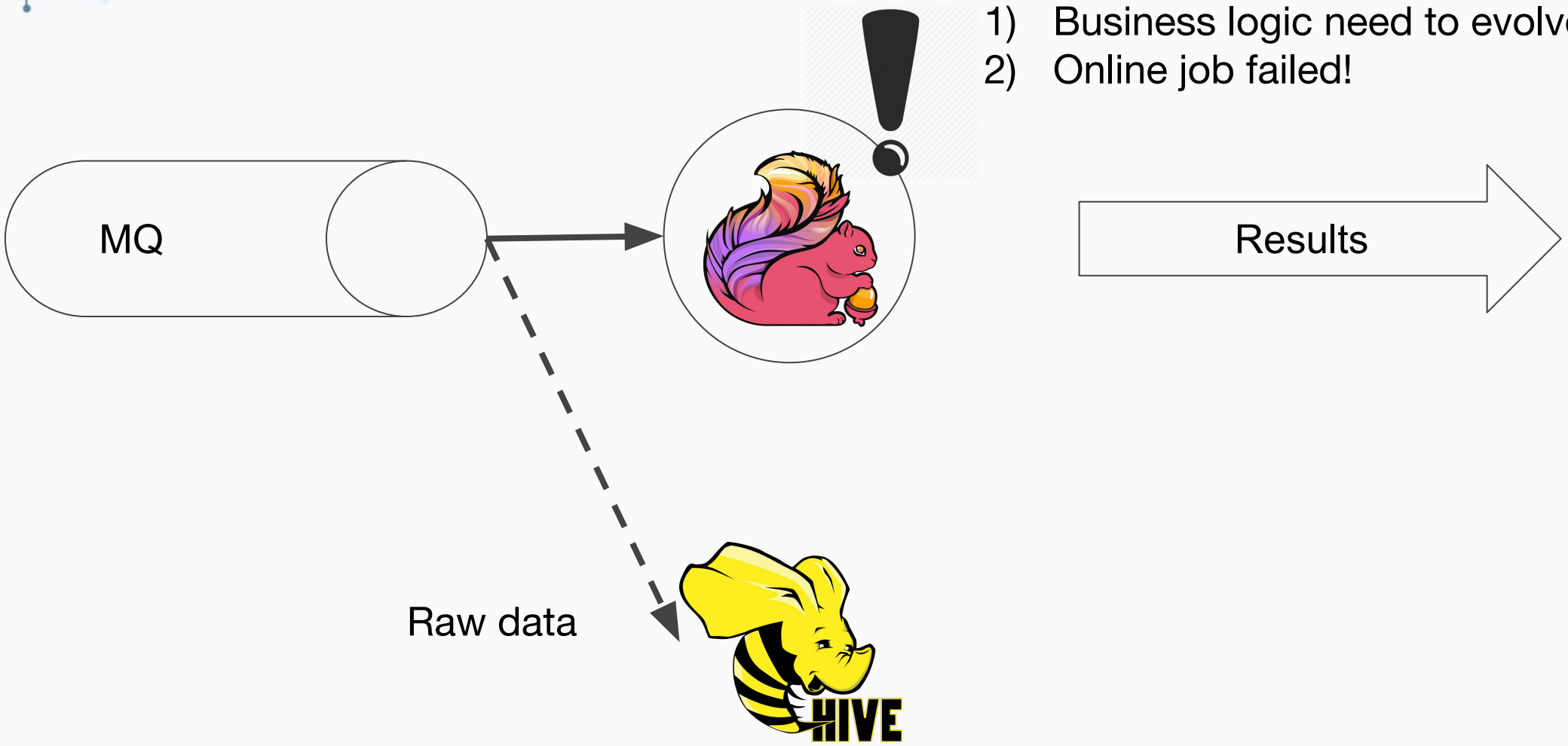
* Only if the offline data volume is small to medium
Otherwise, your Flink will stress out by keeping all history data in state

Apache Flink

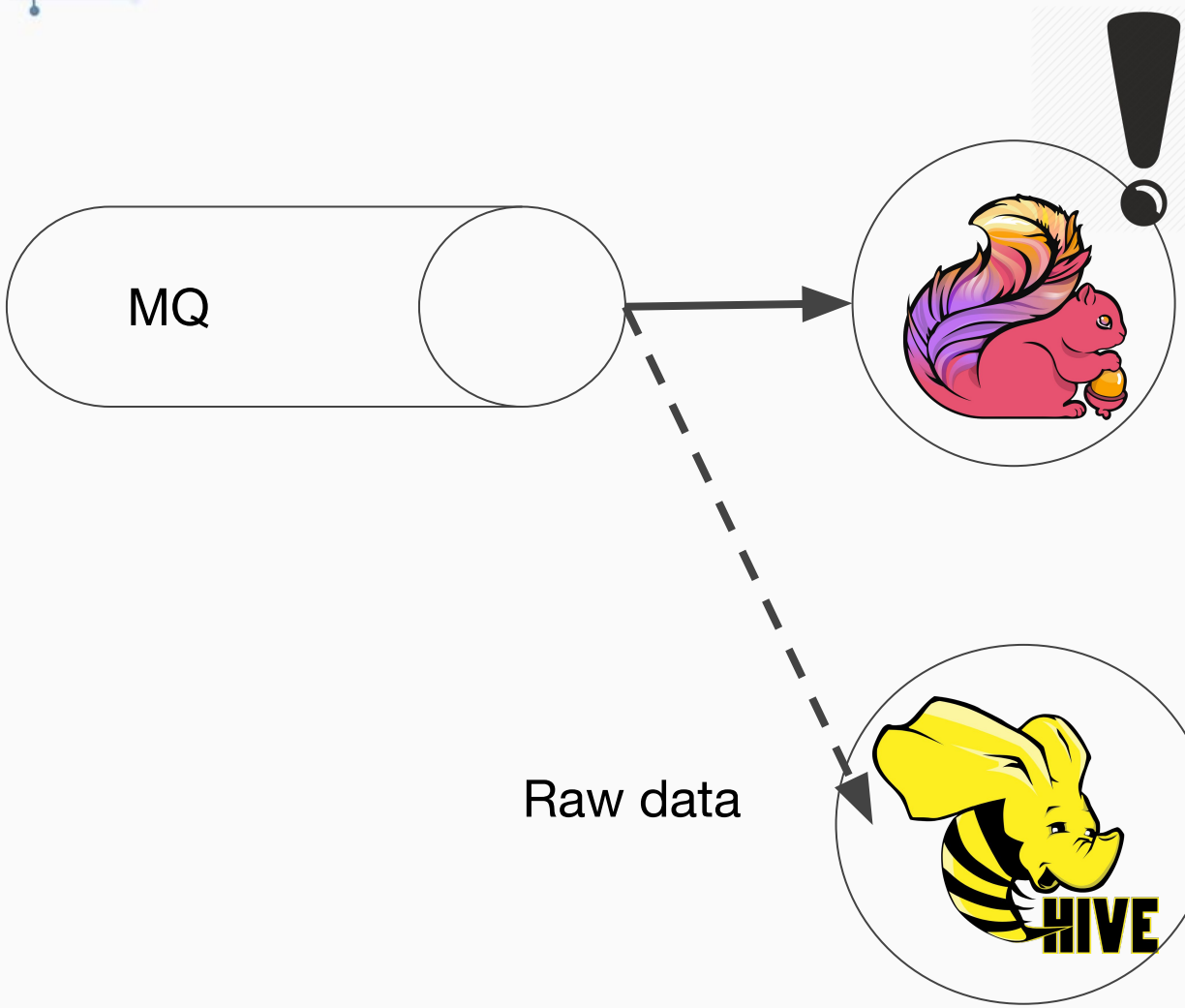
Story 3 - Backfill data on demand or on failures



1) Business logic need to evolve!
2) Online job failed!



1) Business logic need to evolve!
2) Online job failed!

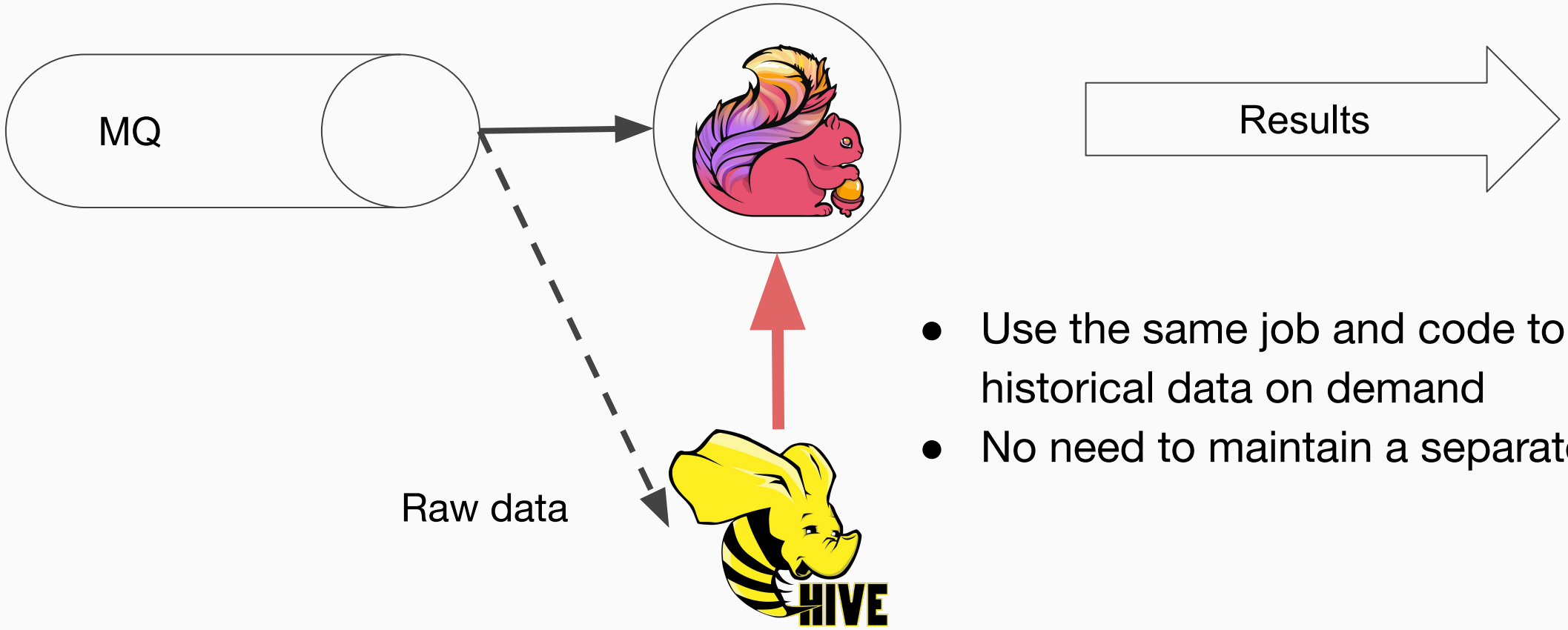


Results

In the past, you woud need a 2nd Hive job to do the backfill

Results

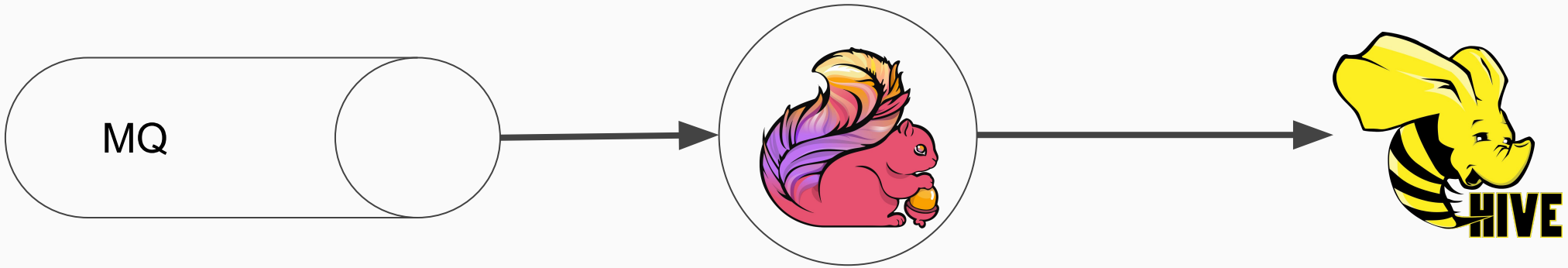
Online pipeline resumes and continue from the latest record because you don’t want to it to process piled-up data to increase pipeline latency and further sacrifice SLAs



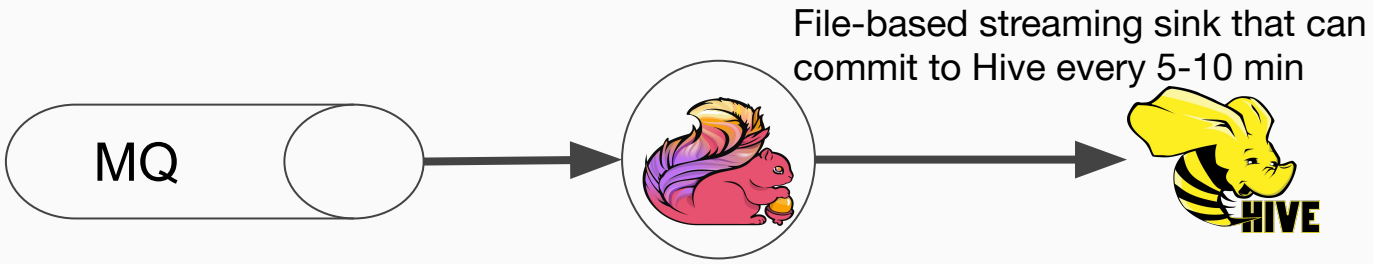
- Use the same job and code to reprocess historical data on demand
- No need to maintain a separate Hive job

Story 4 - Near-real-time data ingestion to Hive (upcoming)

File-based streaming sink that can commit to Hive every 5-10 min



Near-real-time data ingestion to Hive



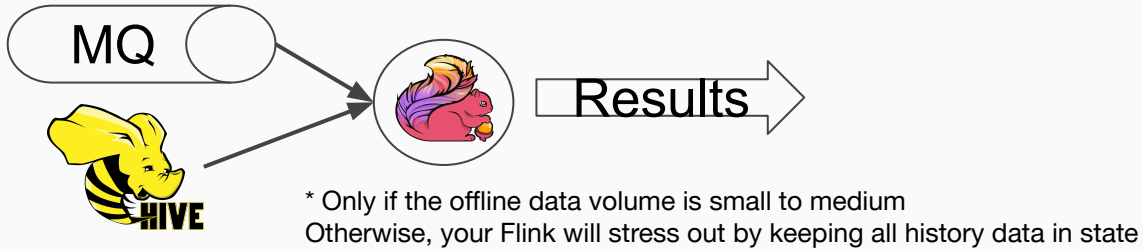
Is this too good to be true, or is there any pitfalls?

Apache Flink

A general question at the end though:

Is traditional data warehouses or data lake really a good fit for streaming-native engine like Flink?

A Few Unsolved Problems - Example 1



What if you want to join real-time data with huge amount of historical data?

- You can't put into Flink as the volume is too big for Flink state to handle
- Maybe point-lookup?
  - Well, traditional dw/data lake don't support point-lookup, they only scan
- Maybe bring in Cassandra/HBase/Redis/…..?
  - You're indeed going the opposite direction of unification by diligently adding one additional infra at a time

A Few Unsolved Problems : Example 2

Does near-real-time come without a limit and a price?

- Pipeline latency **cannot** go below **5-10mins**, otherwise too many small files are created
  - any latency you've saved in Kafka and Flink is caped by the file commit interval in dw/dl
- Still a lot of small files even Flink commits every **5-10mins**. How do you handle them?
  - maybe commit every 1h? please, how dare you call it near-real-time when there's 1h delay
  - maybe have some hourly + daily Airflow jobs? Good luck going back to batch world

Apache Flink

Is there a Solution?

Apache Flink

Is there a Solution?

Yes, we are launching a new purpose-built product **Hologres**

check out the session presented by Xiaowei Jiang

“Data Warehouse, Data Lakes, What’s Next?”

April 23 11am-11:40am PDT, Room5

Conclusions

- Flink 1.10 brings production-read integration with Hive on both metadata management and data handling sides
- There are still many unsolved challenges when integrating Flink with traditional data warehouse and data lakes

Thanks!

Q&A

Twitter: @Bowen__Li