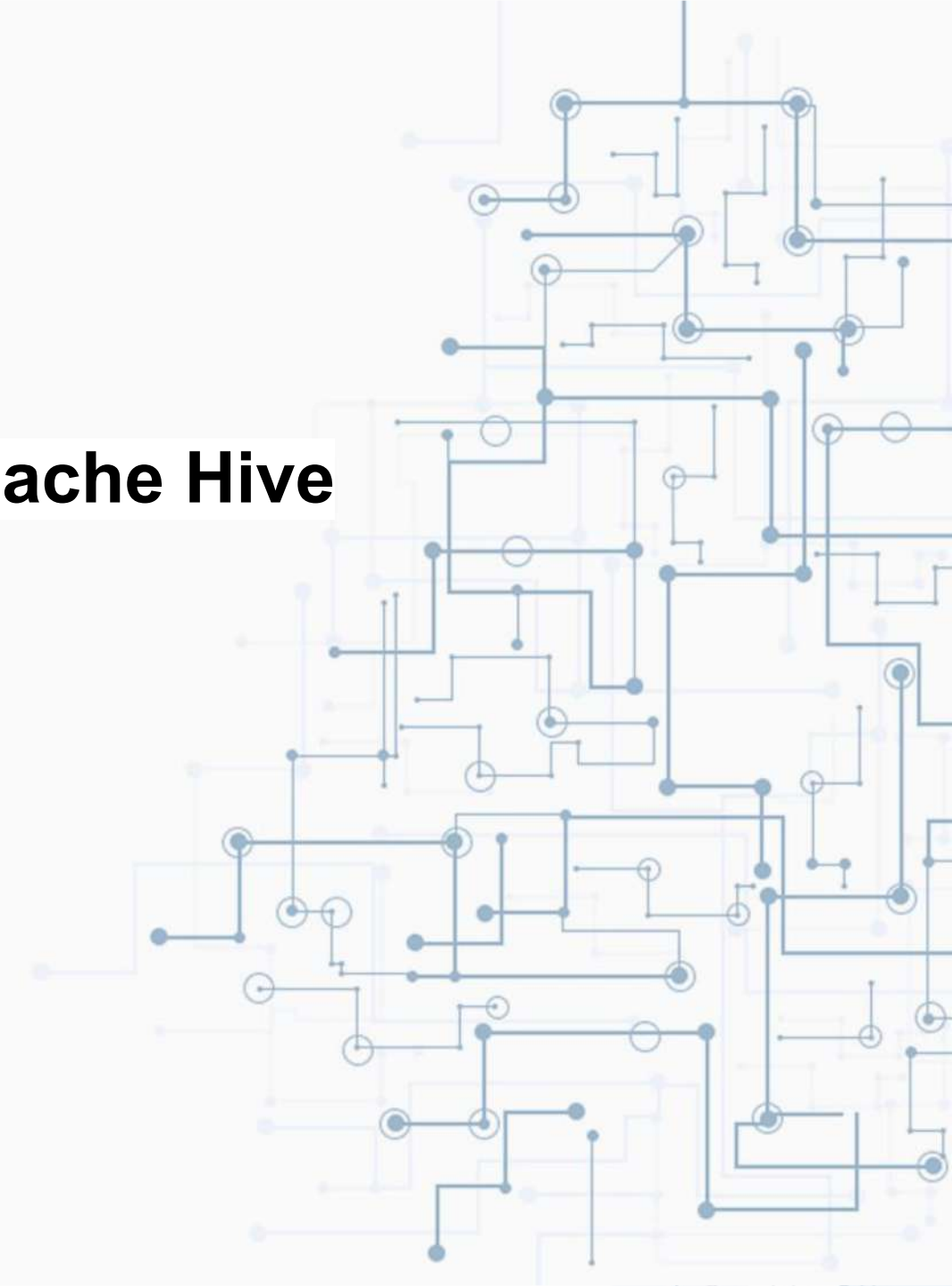
Integrating Apache Flink with Apache Hive

Xuefu Zhang, Senior Staff Engineer, Hive PMC
Bowen Li, Senior Engineer
王刚, Senior Engineer

Apache Flink Meetup 杭州 – 2019年03月02日



聚划算 淘宝网 Taobao.com 天猫 TMALL.COM 阿里云 aliyun.com

Alibaba Group

支付宝 ALIPAY CAINIAO 菜鸟 AliExpress

EB Total

PB Everyday

1T Event/Day

1.7B Events/sec

Subsecond Latency

Billions Events/sec

Ten Thousand Nodes

Ten Thousand Jobs

Apache Flink

Agenda

- Background
- Motivations
- Goals
- Architecture and Design
- Roadmap
- Current Progress
- Demo
- Q&A

Apache Flink

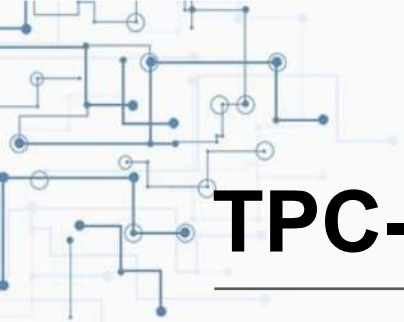
Background

- Stream analytic users usually have also offline, batch analytics
- Many batch users want to reduce latency by moving some of analytics to real-time
- AI is a major driving force behind both real-time and batch analytics (training a model offline and apply a model in real time)
- ETL is still an important use case for big data
- SQL is the main tool processing big data, streaming or batch

Apache Flink

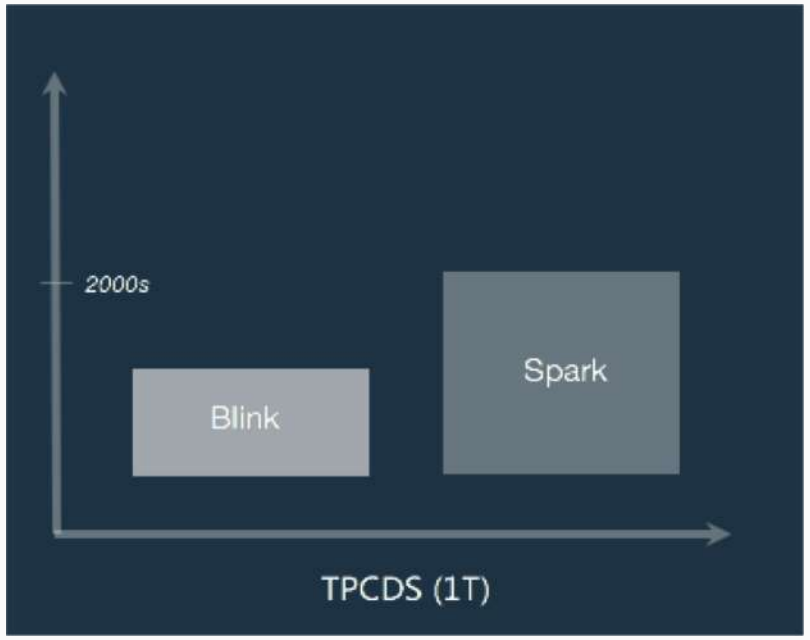
Background (cont'd)

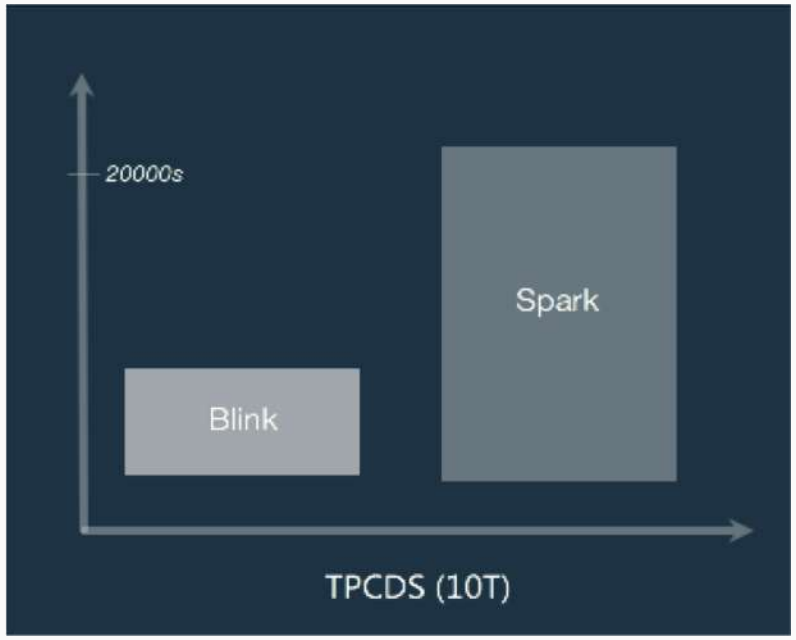
- Flink has showed prevailing advantages over other solutions for heavy-volume stream processing
- In Blink, we systematically explored Flink's capabilities in batch processing
- Flink shows great potential

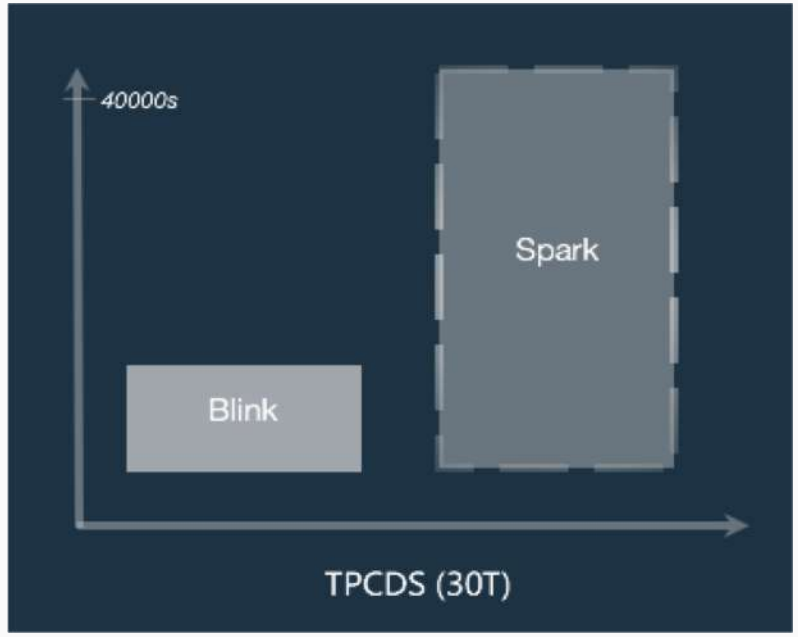


Apache Flink

TPC-DS: Blink v.s. Spark (the Lower, the Better)

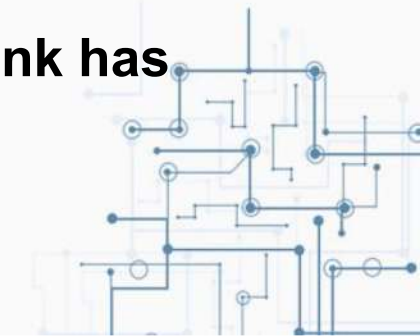


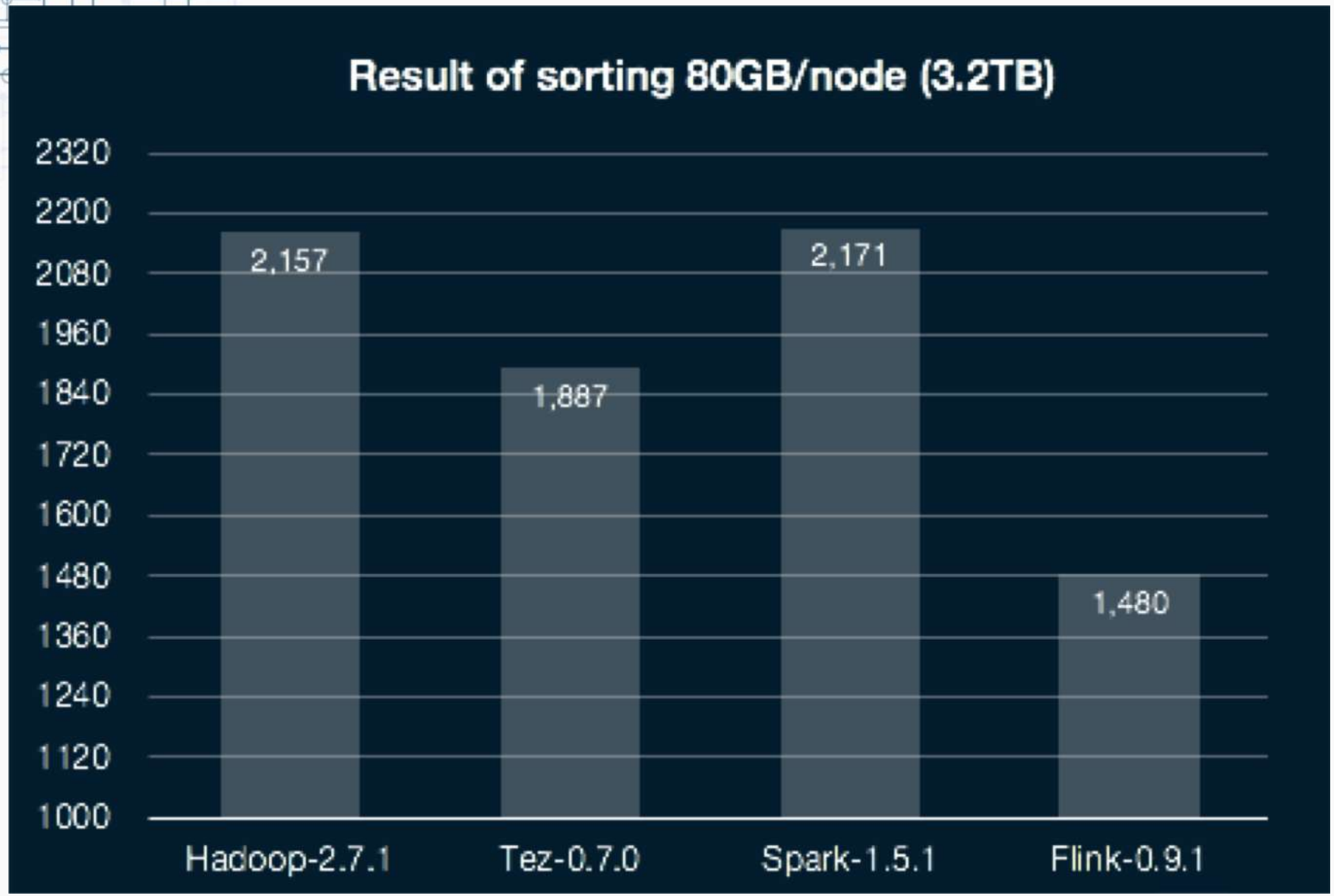




Observation: the larger the data size, the more performance advantage Flink has

Performance of Blink versus Spark 2.3.1 in the TPC-DS benchmark, aggregate time for all queries together.
Presentation by Xiaowei Jiang at Flink Forward Beijing, Dec 2018.





Apache Flink

Flink is the fastest due to its pipelined execution

Tez and Spark do not overlap 1st and 2nd stages

MapReduce is slow despite overlapping stages

A Comparative Performance Evaluation of Flink, Dongwon Kim, POSTECH, Flink Forward 2015

Apache Flink

Background (cont'd)

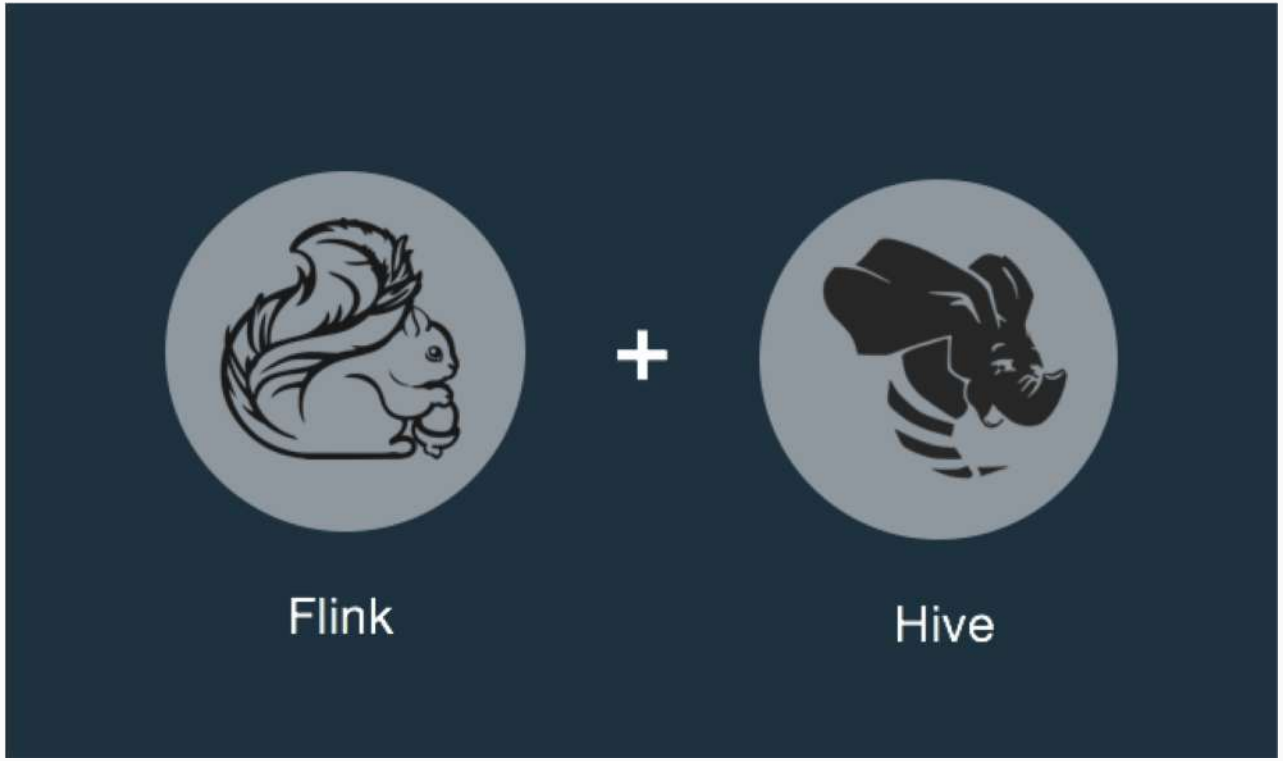
- Flink needs a persistent storage for its metadata

Apache Flink

Background (cont'd)

- Hive is the de facto standard for big data/batch processing on Hadoop
- Hive is the center of big data ecosystem with its metadata store
- Streaming users usually have Hive deployment and need to access data/metadata managed by Hive
- For Hive users, new requirements may come for stream processing

Integrating Flink with Hive



Apache Flink

Motivations

- Strengthen Flink’s lead in stream processing
- Unify a solution for both stream and batch processings
- Provide a unified SQL interface for the solution
- Enrich Flink ecosystem
- Promote Flink’s adoption

Apache Flink

Goals

- Access all Hive metadata stored in Hive metastore
- Access all Hive data managed by Hive
- Store Flink metadata (both streaming or batch) in Hive metastore
- Compatible with Hive grammar while abiding to SQL standard
- Support user custom objects in Hive continue to work in Flink SQL
  - UDFs
  - serdes
  - storage handlers

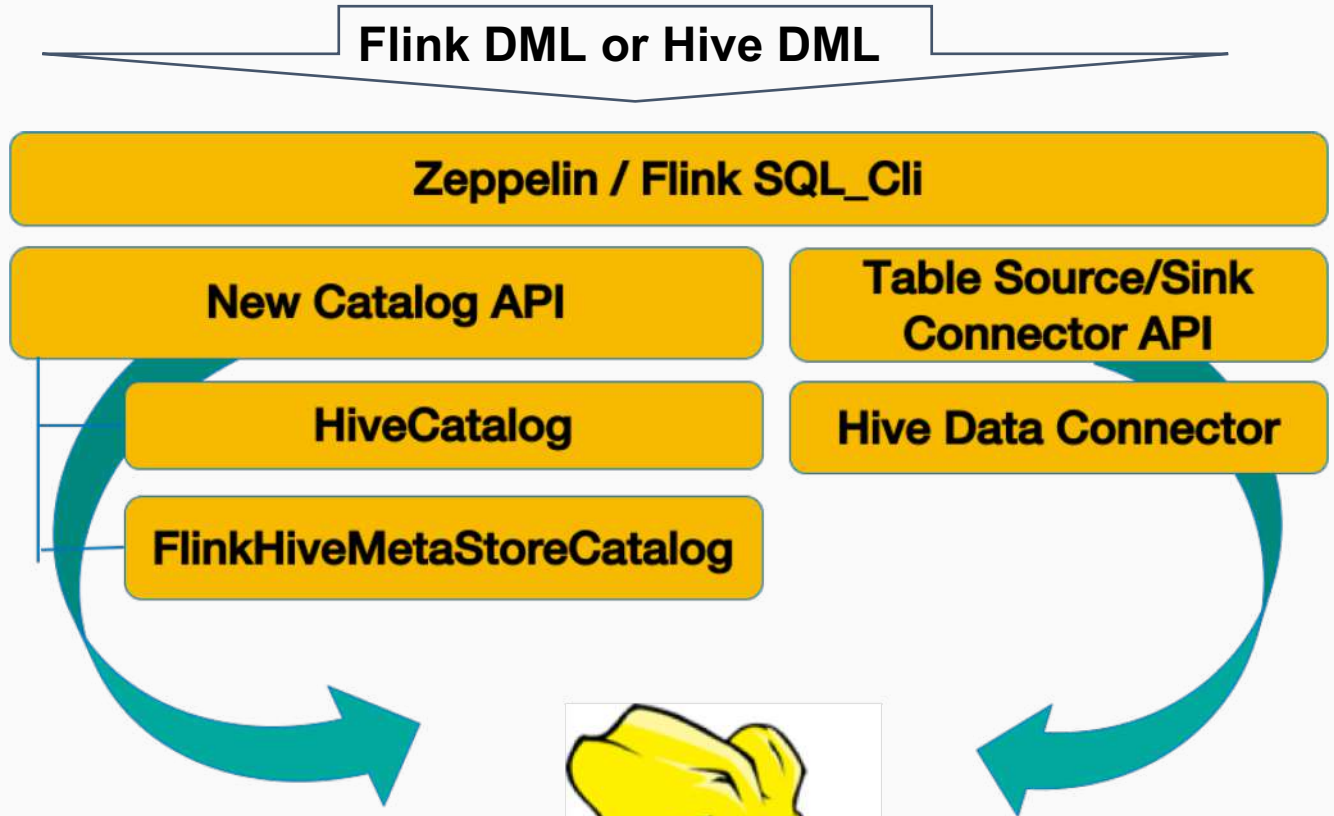
Apache Flink

Goals (cont'd)

- Feature parity with Hive (paritioning, bucketing, etc)
- Data types compatibility

- Make Flink an alternative, more powerful engine in Hive (longer term)

Integrating Flink with Hive - Goals

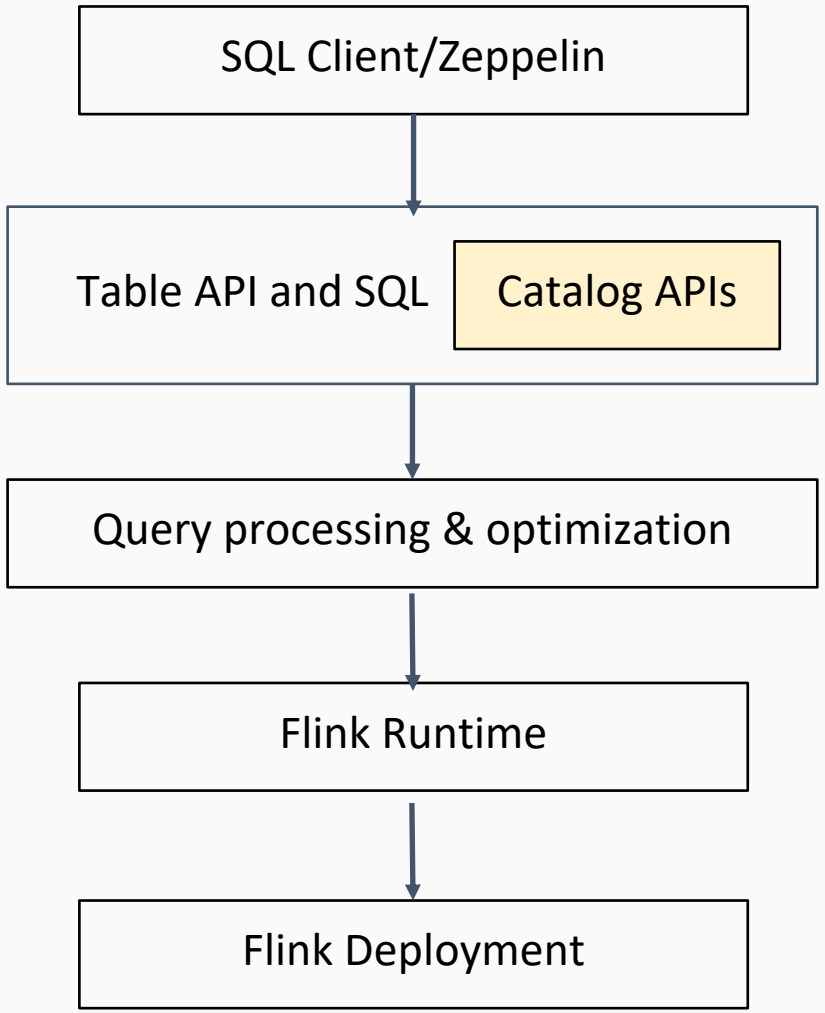


Hive Metadata is 100% compatible
(Data types, Tables, Views, UDFs)

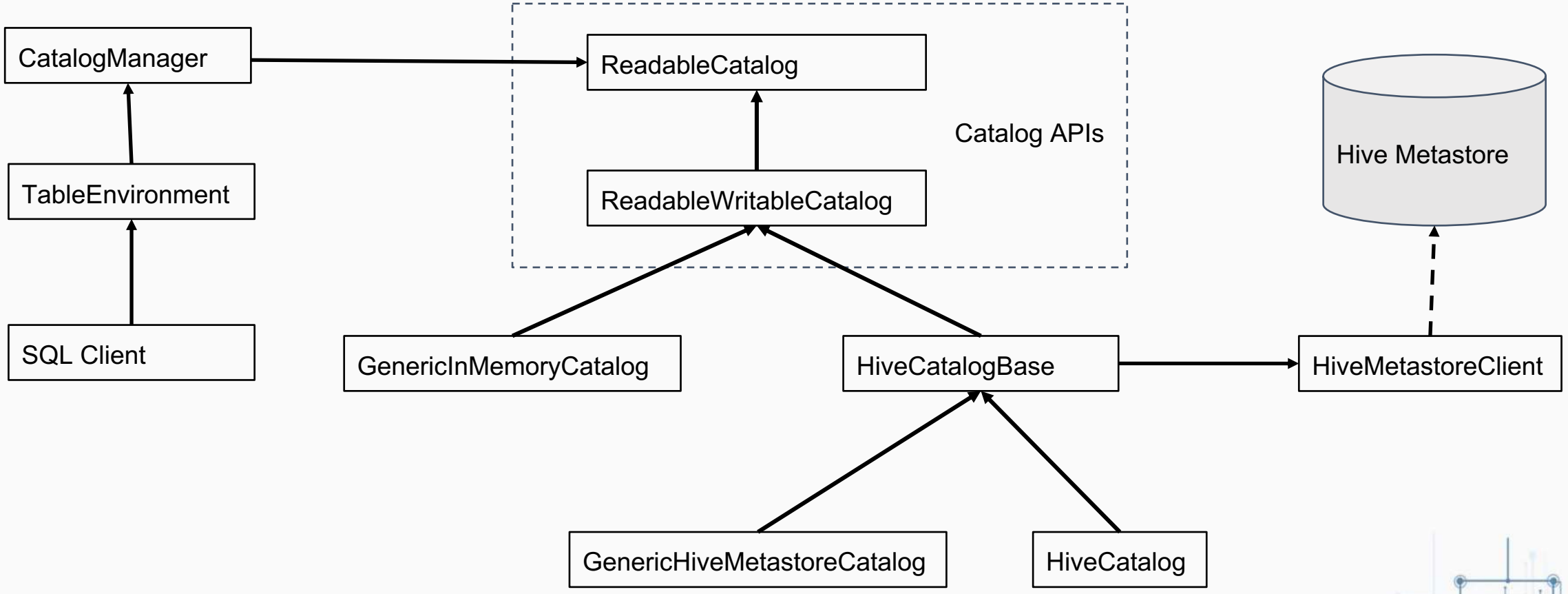
Hive Data is 100% accessible

Apache Flink

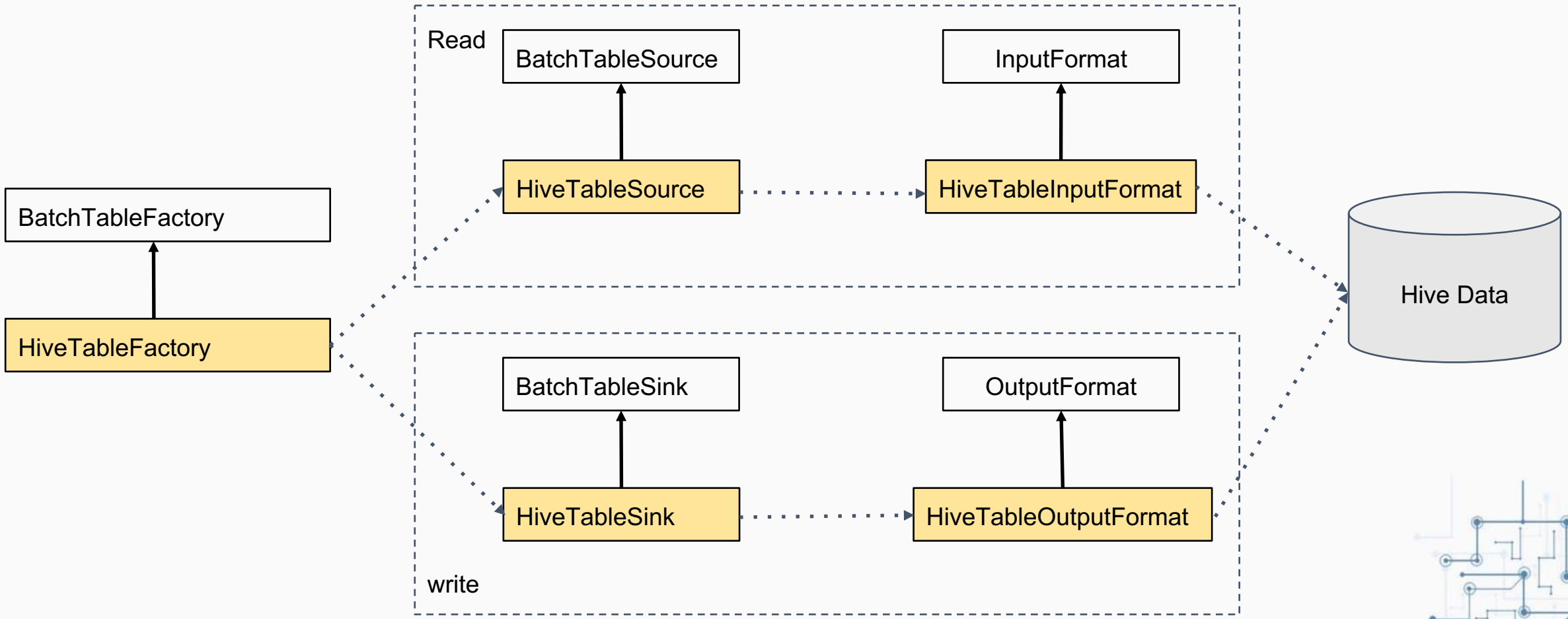
Arichtecture



Design


CatalogManager
ReadableCatalog
Catalog APIs
Hive Metastore
TableEnvironment
ReadableWritableCatalog
SQL Client
GenericInMemoryCatalog
HiveCatalogBase
HiveMetastoreClient
GenericHiveMetastoreCatalog
HiveCatalog
inheritance
reference

Design (cont'd)



Apache Flink

Roadmap

- Basic integration
  - Read access to Hive
  - Table metadata only
  - Simple data types
  - Demo version

Apache Flink

Roadmap (cont'd)

- Deep integration
  - Read/Write Hive metadata, data
  - Most data types
  - Basic DDL/DML
  - All meta objects (tables, functions, views, etc)
  - MVP version

Apache Flink

Roadmap (cont'd)

- Complete integration
  - Complete DDL/DML
  - All data types
  - Temporary meta objects (functions, tables)
  - Support all user custom objects defined in Hive (serdes, storage handlers)
  - Usability, stability
  - First release

Apache Flink

Roadmap (cont'd)

- Longer term
  - Optimizations
  - Feature parity
  - Regular maintenance and releases

- Hive on Flink

Apache Flink

Integrating Flink with Hive - Phases

This is a major change, work needs to be broken into steps

Phase 1: Unified Catalog APIs (FLIP-30, FLINK-11275)

Phase 2: Integrate Flink with Hive (FLINK-10556)

- for metadata thru Hive Metastore (FLINK-10744)
- for data (FLINK-10729)

Phase 3: Support SQL DDL in Flink (FLINK-10232)

Apache Flink

Phase 1: Unified Catalogs APIs

Flink current status:

Barely any catalog support; Cannot connect to external catalogs

What we have done:

Introduced new catalog APIs ReadableCatalog and ReadableWritableCatalog, and framework to connect to Calcite, that supports

- Meta-Objects
  - Database, Table, View, Partition, Functions, TableStats, and PartitionStats
- Operations
  - Get/List/Exist/Create/Alter/Rename/Drop

Status: Done internally, ready to submit PR to community

Phase 1: Unified Catalogs APIs

Flink current status:

No view support, currently views are tables.
Function catalog doesn't have well-managed hierarchy, and cannot persist UDFs.

What we have done:

- Added true view support, users can create and query views on top of tables in Flink
- Introduced in a new UDF management system with proper namespace and hierarchy for Flink based on ReadableCatalog/ReadableWritableCatalog

Status: Mostly done internally, ready to submit PR to community 

Apache Flink

Phase 1: Unified Catalogs APIs

Flink current status:

No well-structured hierarchy to manage metadata

Endless nested Calcite schema (think it as db, like db under db under db under …)

What we have done:

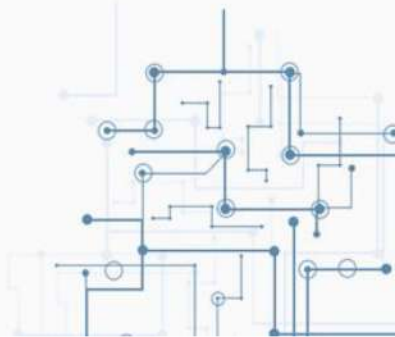
- Introduced two-level management and reference structure: **<catalog>.<db>.<meta-object>**
- Added CatalogManager:
  - manages all registered catalogs in TableEnvironment
  - has a concept of default catalog and default database to simply query

    `select * from mycatalog.mydb.myTable`

    === can be simplified as ===>>>

    `select * from myTable`

Status: Done internally, ready to submit PR to community



Apache Flink

Phase 1: Unified Catalogs APIs

Flink current status:

No production-ready catalogs

What we have done:

Developed three production-ready catalogs

- GenericInMemoryCatalog
  - in-memory non-persistent, per session, default
- HiveCatalog
  - compatible with Hive, able to read/write Hive meta-objects
- GenericHiveMetastoreCatalog
  - persist Flink streaming and batch meta-objects

Status: Done internally, ready to submit PR to community

Apache Flink

Phase 1: Unified Catalogs APIs

Catalogs are pluggable and opens opportunities for

- Catalog for MQ
  - Kafka(Confluent Schema Registry), RabbitMQ, Pulsar, RocketMQ, etc
- Catalog for structured data
  - RDMS like MySQL, etc
- Catalogs for semi-structured data
  - ElasticSearch, HBase, Cassandra, etc
- Catalogs for your other favorite data management system
  - ……

Apache Flink

Phase 2: Flink-Hive Integration - Metadata - HiveCatalog

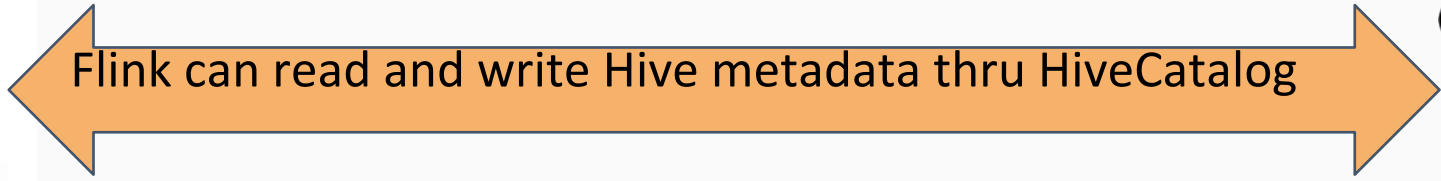
Flink current status:

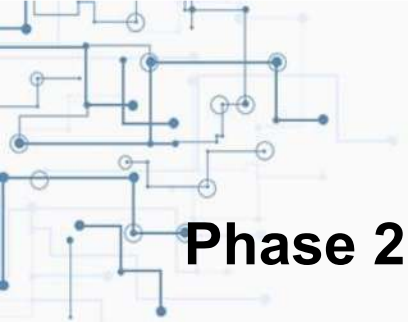
Flink's batch is great, but cannot run against the most widely used data warehouse - Hive

What we have done:

Developed HiveCatalog

- Flink can read Hive metaobjects, like tables, views, functions, table/partition stats,thru HiveCatalog
- Flink can create Hive metaobjects and write back to Hive via HiveCatalog such that Hive can consume





Apache Flink

Phase 2: Flink-Hive Integration - Metadata - GenericHiveMetastoreCatalog

Flink current status:

Flink’s metadata cannot be persisted anywhere

Users have to recreate metadata like tables/functions for every new session, very inconvenient

What we have done:

- Persist Flink’s metadata (both streaming and batch) by using Hive Metastore purely as storage

Apache Flink

HiveCatalog v.s. GenericHiveMetastoreCatalog

- for Hive batch metadata
- Hive can understand

- for any streaming and batch metadata
- Hive may not understand

Apache Flink

Phase 2: Flink-Hive Integration - Data

Flink current status:

Has flink-hcatalog module, but last modified 2 years ago - not really usable.

HCatalog also cannot to access all Hive data

What we have done:

Connector:

- Developed HiveTableSource that supports reading both partition and non-partition table and views, and partition-pruning
- Working on HiveTableSink

Data Types:

- Added support for all Hive simple data types.
- Working on supporting Hive complex data types (array, map, struct, etc)

Apache Flink

Phase 2: Flink-Hive Integration - Hive Compatibility

- Hive version
  - Officially support Hive **2.3.4** for now
  - We plan to support more Hive versions in the near future

- Above features are partially available in the Blink branch of Flink released in Jan 2019
  - https://github.com/apache/flink/tree/blink

Apache Flink

Phase 3: Support SQL DDL + DML in Flink

- In progress
- Some will be shown in demo

Apache Flink

Example and Demo Time!

Query your Hive data from Flink!

Apache Flink

Table API Example

```
BatchTableEnvironment tEnv = ...
tEnv.registerCatalog(new HiveCatalog("myHive1", "thrift:xxx"));
tEnv.registerCatalog(new HiveCatalog("myHive2", hiveConf));
tEnv.setDefaultDatabase("myHive1", "myDb");

// Read Hive meta-objects
ReadableCatalog myHive1 = tEnv.getCatalog("myHive1");
myHive1.listDatabases();
myHive1.listTables("myDb");

ObjectPath myTablePath = new ObjectPath("myDb", "myTable");
myHive1.getTable(myTablePath);
myHive1.listPartitions(myTablePath);

// Query Hive data
tEnv.sqlQuery("select * from myTable").print()
```

Apache Flink

SQL Client Example

// Register catalogs in sql-cli-defaults.yml

```yaml
execution: ...

deployment: ...

#==============================================================
# Catalog properties
#==============================================================
catalogs:
 - name: myhive
   catalog:
     type: hive
     connector:
       hive.metastore.uris: thrift://localhost:9083
     is-default: false
     default-database: default
 - name: mygeneric
   catalog:
     type: generic_hive_metastore
     connector:
       hive.metastore.uris: thrift://<ip>:<port>
     default-database: default
```

Apache Flink

SQL Client Example (cont')

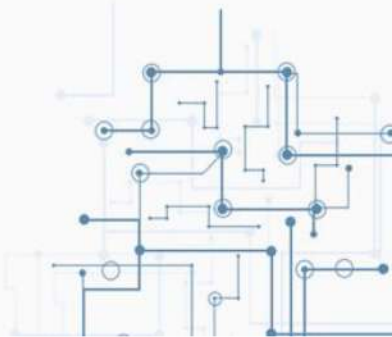
```
Flink SQL> SHOW CATALOGS;
myhive1
mygeneric

Flink SQL> USE myhive1.myDb;
Flink SQL> SHOW DATABASES;
myDb

Flink SQL> SHOW TABLES;
myTable

Flink SQL> DRESCRIBE myHiveTable;
...

Flink SQL> SELECT * FROM myHiveView;
...
```



Apache Flink

Happy Live Demo on SQL CLI!

```
62      Tarte au sucre  29      3       48 pies 49.3    17      0       0       0
63      Vegie-spread    7       2       15 - 625 g jars 43.9    24      0       5       0
64      Wimmers gute Semmelknödel       12      5       20 bags x 4 pieces      33.25   22      80      30      0
65      Louisiana Fiery Hot Pepper Sauce        2       2       32 - 8 oz bottles       21.05   76      0       0       0
66      Louisiana Hot Spiced Okra       2       2       24 - 8 oz jars  17.0    4       100     20      0
67      Laughing Lumberjack Lager       16      1       24 - 12 oz bottles      14.0    52      0       10      0
68      Scottish Longbreads     8       3       10 boxes x 8 pieces     12.5    6       10      15      0
69      Gudbrandsdalsost        15      4       10 kg pkg.      36.0    26      0       15      0
70      Outback Lager   7       1       24 - 355 ml bottles     15.0    15      10      30      0
71      Flotemysost     15      4       10 - 500 g pkgs.        21.5    26      0       0       0
72      Mozzarella di Giovanni  14      4       24 - 200 g pkgs.        34.8    14      0       0       0
73      Röd Kaviar      17      8       24 - 150 g jars 15.0    101     0       5       0
74      Longlife Tofu   4       7       5 kg pkg.       10.0    4       20      5       0
75      Rhönbräu Klosterbier    12      1       24 - 0.5 l bottles      7.75    125     0       25      0
76      Lakkalikööri    23      1       500 ml  18.0    57      0       20      0
77      Original Frankfurter grüne Soße 12      2       12 boxes        13.0    32      0       15      0
Time taken: 1.622 seconds, Fetched: 77 row(s)
hive> exit;

[admin@r101072041.sqa.zmf /home/admin/apache-hive-2.3.4]
$bin/hive
which: no hbase in (/home/admin/apache-hive-2.3.4/bin:/home/admin/hadoop-2.8.5/bin:/home/admin/hadoop-2.8.5/sbin:/home/admin/jdk1.8.0_191/bin/:/sbi
/usr/bin:/usr/local/sbin:/usr/sbin:/usr/X11R6/bin:/home/terry.wg/.local/bin:/home/terry.wg/bin)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/admin/apache-hive-2.3.4/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/admin/hadoop-2.8.5/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/home/admin/apache-hive-2.3.4/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) o
releases.
hive>
```

Apache Flink

This tremendous amount of work cannot happen without help and support

Shout out to everyone in the community and our team
who have been helping us with designs, codes, feedbacks, etc!

Apache Flink

Conclusions

- Flink is good at stream processing, but batch processing is equally important
- Flink has shown its potential in batch processing
- Flink/Hive integration benefits both communities
- This is a big effort
- We are taking a phased approach
- Your contribution is greatly welcome and appreciated!

THANKS