Presto on Spark

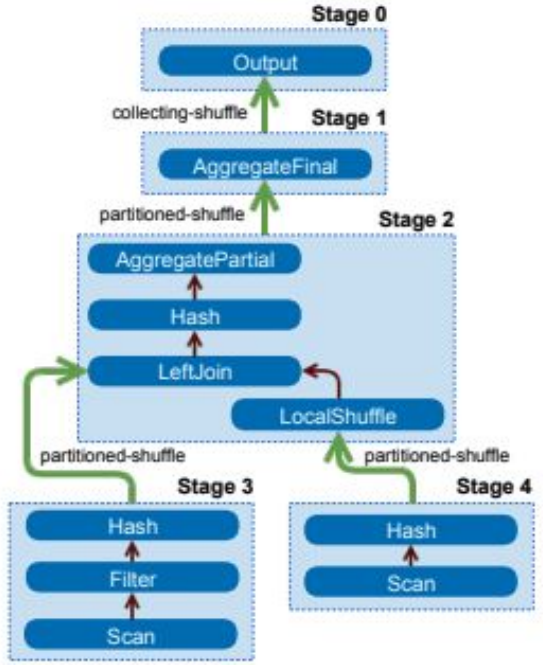
Andrii Rosa, Wenlei Xie

Agenda

- Motivation
- Design overview
- Early results
- Development status
- Q&A

Presto Architecture Overview

- Designed for Interactive use cases
- Classic MPP architecture
- In-memory streaming shuffle
  - Low latency
  - More operations can be done in parallel
- Standalone, multi-tenant service
  - Always “warm”, no “startup” delay
  - Better resource sharing (dynamic parallelism, shared memory)



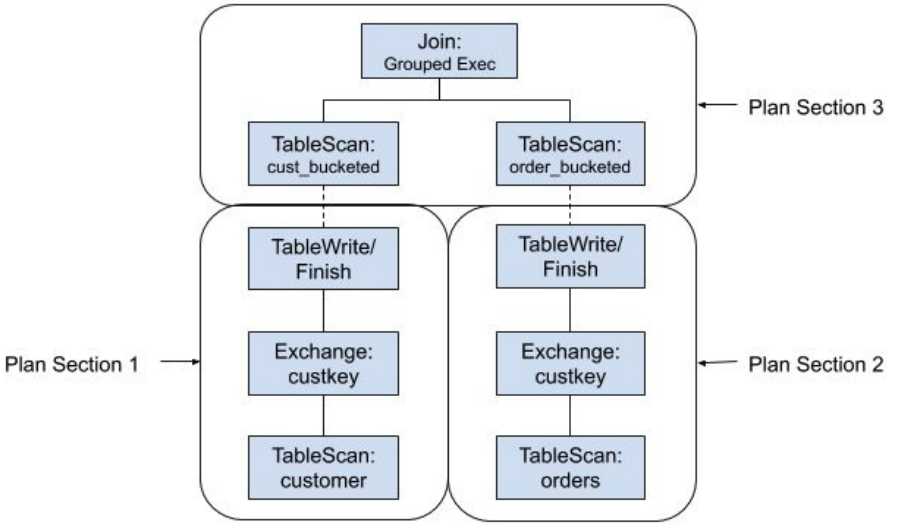
https://research.fb.com/publications/presto-sql-on-everything/

Scalability Limitations

- 5TB distributed memory limit
  - With streaming shuffle Aggregations and Joins have to be processed all at once
- Number of workers is limited to ~300 reducers and ~600 mappers
  - NxN streaming doesn’t scale
  - Single coordinator
- Queries that run more than 8 hours have ~50% of finishing
  - No failure recovery
  - Multi-tenant environment
  - No resource isolation
- At Facebook we run Presto for **Batch**
  - **5 times larger deployment** for Batch vs Interactive
  - Batch requires higher memory limits, parallelism, and reliability

Presto Unlimited

- Brings MapReduce style processing to MPP database
- Stores intermediate (shuffle) data on disk
- Allows more granular joins and aggregations processing
- Adds support to run large memory queries (>5TB)
- Increases reliability by allowing partial failure recovery
- Can be run on existing Presto deployments



https://prestodb.io/blog/2019/08/05/presto-unlimited-mpp-database-at-scale

Presto Unlimited Limitations

- Number of workers still limited to number of nodes in cluster (600)
- No resource isolation
  - Multitenant environment
  - Single ill behaving query can jeopardize the entire cluster
- No granular failure recovery
  - In some cases requires the whole stage to be restarted
- Hard to utilize off-peak capacities
  - Lack of resource allocation flexibility
  - Inability to easily add/remove worker nodes

Spark Architecture

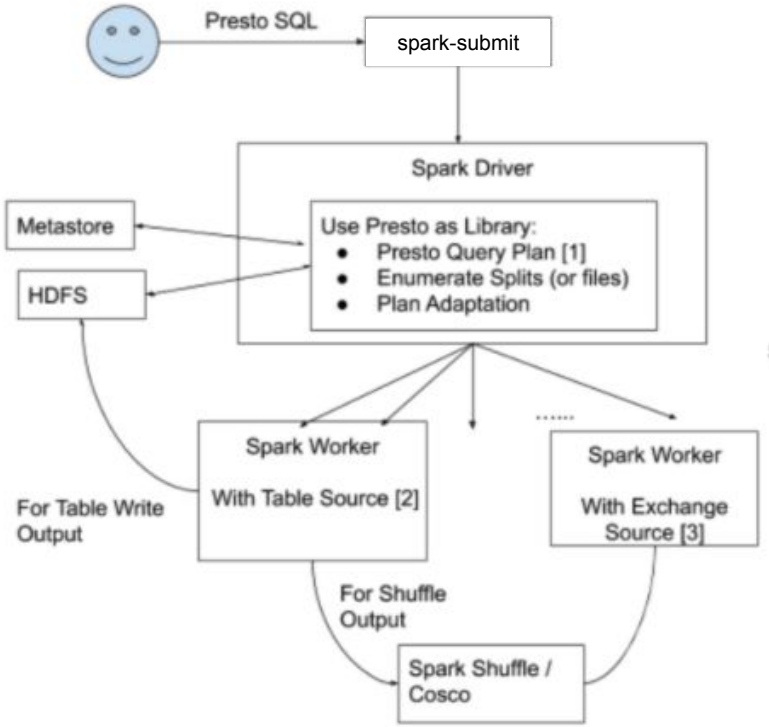
- Designed for large scale, reliable ETL processing
- Supports granular failure recovery
- Isolated containers (JVM) for each query
- Supports thousands of mappers/reducers in parallel
  - Driver (coordinator) is allocated per query
  - Shuffle algorithm is more scalable
  - Supports disaggregated shuffle
    - Apache Crail (https://crail.apache.org/)
    - Cosco (https://databricks.com/session/cosco-an-efficient-facebook-scale-shuffle-service)
- Flexible at resource allocation
  - Easy to add/remove worker nodes from the cluster

Why not simply use Spark?

- Presto SQL reusability and unification
  - Translation based solution doesn’t scale
  - Always hard to translate long tail
- UDFs
- Additional SQL Features
  - Lambdas
  - Powerful geospatial toolkit
- Query federation capabilities
- Efficient columnar evaluation engine
  - Project Aria

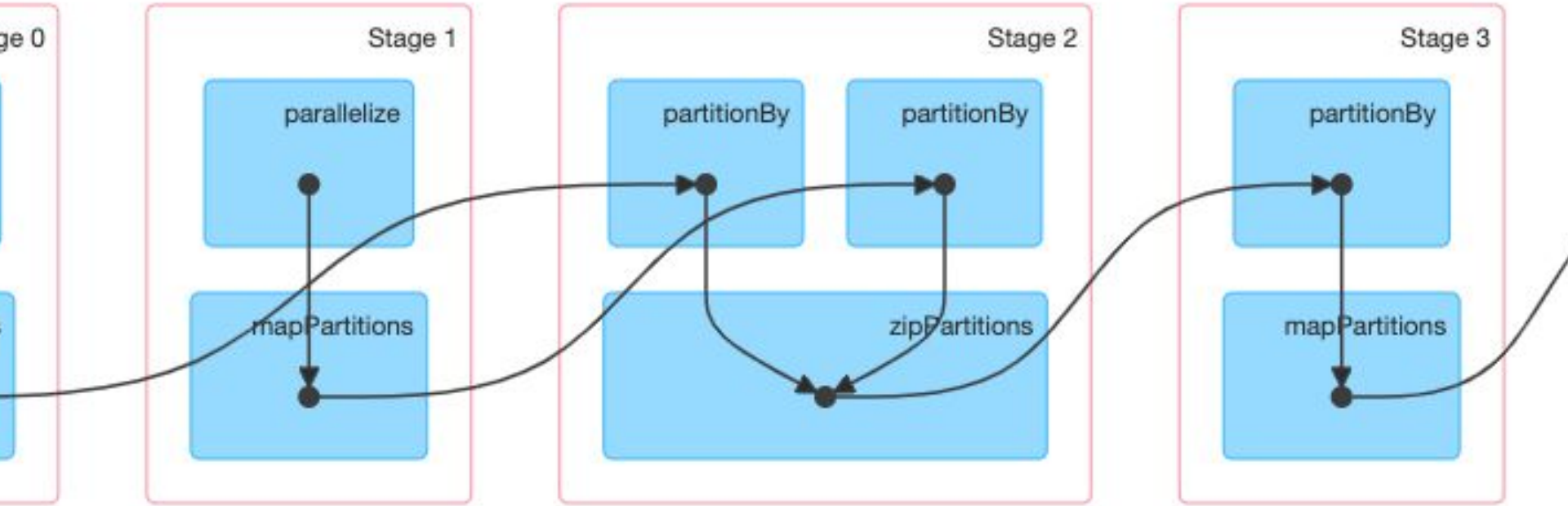
Presto on Spark Architecture

- Runs Presto as a library
  - No dependency on Presto deployment model
  - Custom batch job
- Transforms Presto distributed plan into Spark RDD
- Runs Presto evaluation code as an opaque “mapper” or “reducer”



Design Doc:
https://tinyurl.com/presto-on-spark-design

Spark DAG


Stage 1
parallelize
mapPartitions
Stage 2
partitionBy
partitionBy
zipPartitions
Stage 3
partitionBy
mapPartitions

spark-submit

```
# spark-submit \
   --master spark://spark-master:7077 \
   --class com.facebook.presto.spark.launcher.PrestoSparkLauncher \
   /presto/presto-spark-launcher/target/presto-spark-launcher-0.235-SNAPSHOT.jar \
   --package
   /presto/presto-spark-package/target/presto-spark-package-0.235-SNAPSHOT.tar.gz \
   --config /presto/etc/config.properties \
   --catalogs /presto/etc/catalogs \
   --catalog hive \
   --schema default \
   --file /tmp/query.sql
```

Early results

- **50TB+** distributed memory
- **3x** reduction in wall time
- **30%** - **50%** increase in CPU time
  - Shuffle requires sorting
  - Row oriented format is not very efficient

Development Status

- Initial version is merged to OSS
  - https://github.com/prestodb/presto/pull/13760
- Working on supporting more query shapes
  - N-way join
  - Broadcast Join
  - Bucketed tables support
- Working on improving CPU efficiency
  - More efficient row oriented format for shuffle
- Working on improving integration with Spark
  - Presto threading model is different
- Working on better integration with operational tooling
  - Event Listener
  - Execution statistics

Q&A