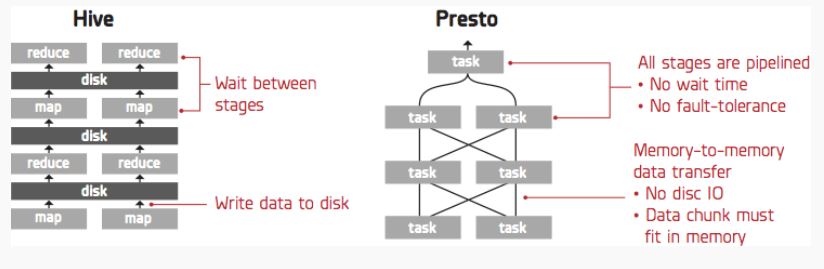
Presto vs Hive



* **What is Hive?**

**Hive** translates **SQL queries** into multiple stages of MapReduce and it is powerful enough to handle huge numbers of jobs (Although as Arun C Murthy pointed out, modern **Hive** runs on Tez whose computational model is similar to Spark’s). MapReduce is fault-tolerant since it stores the intermediate results into disks and enables batch-style data processing. Many of our customers issue thousands of Hive queries to our service on a daily basis. A key advantage of **Hive** over newer **SQL-on-Hadoop** engines is robustness: Other engines like Cloudera’s Impala and **Presto** require careful optimizations when two large tables (100M rows and above) are joined. **Hive** can join tables with billions of rows with ease and should the jobs fail it retries automatically. Furthermore, **Hive** itself is becoming faster as a result of the Hortonworks Stinger initiative.

-**Best of Hive:**

1) Large data aggregations

2) Large Fact-to-Fact joins

3) Large distincts (aka de-duplication jobs)

4) Batch jobs that can be scheduled

**-How to best use Hive:**

1) Optimized for: Throughput

2) SQL Standardized fidelity: HiveQL (subset of common data warehousing SQL)

3) Windows functions: Yes

4) Large JOINs: Very good for large Fact-to-Fact joins

* **What is Presto?**

In some instances simply processing **SQL queries** is not enough—it is necessary to process queries as quickly as possible so that data scientists and analysts can use Treasure Data for quickly gaining insights from their data collections. For these instances Treasure Data offers the **Presto query engine**. **Presto** is an in-memory distributed **SQL query** engine developed by Facebook that has been open-sourced since November 2013. **Presto** has been adopted at Treasure Data for its usability and performance.

-**Best of Presto:**

1) Interactive queries (where you want to wait for the answer)

2) Quickly exploring the data (e.g. what types of records are found in the table)

3) Joins with a large Fact table and many smaller Dimension tables

**-How to best use Presto:**

1) Optimized for: Interactivity

2) SQL Standardized fidelity: Designed to comply with ANSI SQL

3) Windows functions: Yes

4) Large JOINs: Optimized for star schema joins (1 large Fact table and many smaller dimension tables)

**Resume**

**Hive** **is optimized for query throughput, while Presto is optimized for latency.** **Presto** has a limitation on the maximum amount of memory that each task in a query can store, so if a query requires a large amount of memory, the query simply fails. Such error handling logic (or a lack thereof) is acceptable for interactive queries; however, for daily/weekly reports that must run reliably, it is ill-suited. For such tasks, **Hive** is a better alternative.

In terms of data-processing models, **Hive** is often described as a pull model, since its MapReduce stage pulls data from the preceding tasks. **Presto** follows the push model, which is a traditional implementation of DBMS, processing a **SQL query** using multiple stages running concurrently. An upstream stage receives data from its downstream stages, so the intermediate data can be passed directly without using disks. If the query consists of multiple stages, **Presto** can be 100 or more times faster than **Hive**.