Lecture 38:
- PrestoDB (wrap up)
- Dremel quick overview

Announcements:
- HW-6 out (due Fri)
- Make up Quiz Fri: ...

PrestoDB System Design

Scheduling:
- determine order stages are scheduled in (stage scheduling)
- determine how many and which nodes tasks placed on (task scheduling)

Stage Scheduling: two different modes
- all-at-once: stages execute concurrently as data becomes available
  - benefits latency-sensitive applications (wall-clock time)
- phased: stages executed in physical-plan topological order
  - benefits batch analytics (reduces memory use)

Task Scheduling:
- distinguishes leaf node (source read) scheduling
- ... from intermediate node scheduling
PrestoDB System Design

Leaf scheduling: ... leaf nodes process splits
• most CPU time is spent reading data – decompress, decode, filter, etc.
• work highly parallelizable, so leaf tasks scheduled on every worker node
• splits are assigned to leaf tasks lazily (as become available)
• allows for load balancing (local queue), allows results sooner

Intermediate scheduling:
• intermediate nodes can be placed on any worker node
• engine decides how many tasks should be schedule per stage
• data produced is stored in memory buffers for other workers
• workers request intermediate results from other workers

Engine monitors and adapts to execution: ... lots of details!
• e.g., adjusts parallelization based on buffer use (to improve resource use)

Fault Tolerance: “Presto does not have meaningful built-in fault tolerance for coordinator or worker node crash failures”
• coordinator failures cause the cluster to become unavailable
• worker node crash failure causes all queries running on that node to fail
• Presto relies on clients to automatically retry failed queries

However, suggest that node crashes are not very common in their data centers
• re-executing a query often faster than if fault tolerance built in (overhead)
• e.g., a 1-hour query executed twice is still faster than a 1-day query
• the exception is if a query is inherently long-running
PrestoDB: A Decade Later

Various improvements over the years (†)

• multiple coordinators (to remove single-point of failure)
• various query operator performance increases
• caching and spill memory to temporary storage
• recoverability to materialize intermediate data
• intergration with Spark (Presto as a library that runs on Spark)

Deprecating other systems in favor of Presto ... single system to:

• serve interactive, ad-hoc, ETL, and graph processing workloads
• for the entire data warehouses

And now Velox ... C++ lib for high performance single-node query execution

• takes DAG of operators ... a reusable/extensible component

© S. Bowers CPSC 324, Spring 2024 5

Back to History

Somewhat recent distributed “cluster-native” (cloud) OLAP data systems:

• Map Reduce  Google, circa 2004
• Hadoop  circa 2006
• Dremel  Google, circa 2006, shared nothing (interactive, in situ)
• Spark  circa 2010, Hadoop too slow
• Dremel  circa 2010, shared disk (GFS), shuffle service, etc.
• Apache Hive  Facebook, circa 2009
• Presto  Facebook, circa 2012
• ... many others (e.g., Snowflake, AWS RedShift, SparkSQL, Velox)
Dremel features / properties: ... note: PrestoDB is similar
• shared disk
• in-memory shuffle service (can spill to disk) ... hash joins
• columnar storage via Capacitor ... Google’s version of Parquet
• adaptive query optimization (change query plan as executing)

Query processing and optimization is also similar ...
• convert logical plan into stages (pipelines)
• each stage can have multiple parallel tasks
• coordinator and worker nodes, do the “normal” stuff
• try to reduce shuffle steps, etc.

Dremel: Distributed Shuffle Service

Shuffle Service:
• In-memory (key-value store)
• Can spill to disk (GFS/Colossus)
• Built on custom hardware
• Simplifies design / optimization
• Workers don’t maintain data

Scheduler:
• Assign tasks based on need
• ... adaptively scales up and down
• Stages can be pipelined

Note: Similar service used in Dataflow
Dremel: Additional Features

**Fault Tolerance**: redundant tasks
- ... if worker doesn't produce results within deadline

**Example Optimizations**: ... in addition to dynamic worker assignment

1. **Simplify Joins**: If a relation is smaller than expected after filter-step
   - For $R \bowtie S$, if $R$ smaller than expected
   - Just send it directly to each join stage worker "Broadcast Join"
   - Avoids partitioning step

2. **Dynamic Reshuffling**: ... if a partition starts to grow too large
   - Shuffle service can create additional partitions "on-the-fly"
   - Then repartition in the service itself (e.g., $P_2$ to $P'_2$, $P''_2$, etc.)

First "modern" lakehouse, component-oriented trend (storage, shuffle, exec, ...)

© S. Bowers CPSC 324, Spring 2024