Tuning Performance for SQL-on-Anything Analytics

Martin Traverso, Co-creator of Presto
Kamil Bajda-Pawlikowski, CTO Starburst

@prestosql @starburstdata
Presto: SQL-on-Anything

Deploy Anywhere, Query Anything
Why Presto?

- Community-driven open source project
- High performance ANSI SQL engine
  - New Cost-Based Query Optimizer
  - Proven scalability
  - High concurrency
- Separation of compute and storage
  - Scale storage and compute independently
  - No ETL or data integration necessary to get to insights
  - SQL-on-anything
- No vendor lock-in
  - No Hadoop distro vendor lock-in
  - No storage engine vendor lock-in
  - No cloud vendor lock-in
Project History

**FALL 2012**
4 developers start Presto development

**FALL 2013**
Facebook opensource Presto

**SPRING 2015**
Teradata joins the community, begins investing heavily in the project

**SUMMER 2017**
180+ Releases
50+ Contributors
5000+ Commits

**WINTER 2017**
Starburst is founded by a team of Presto committers, Teradata veterans

**WINTER 2019**
Presto Software Foundation established
Community

See more at our Wiki
Presto in Production

**Facebook:** 10,000+ of nodes, HDFS (ORC, RCFile), sharded MySQL, 1000s of users

**Uber:** 2,000+ nodes (several clusters on premises) with 160K+ queries daily over HDFS (Parquet/ORC)

**Twitter:** 2,000+ nodes (several clusters on premises and GCP), 20K+ queries daily (Parquet)

**LinkedIn:** 500+ nodes, 200K+ queries daily over HDFS (ORC), and ~1000 users

**Lyft:** ------ redacted due to the quiet period for the IPO ------

**Netflix:** 300+ nodes in AWS, 100+ PB in S3 (Parquet)

**Yahoo! Japan:** 200+ nodes for HDFS (ORC), and ObjectStore

**FINRA:** 120+ nodes in AWS, 4PB in S3 (ORC), 200+ users
Starburst Data

Founded by Presto committers:
- Over 4 years of contributions to Presto
- Presto distro for on-prem and cloud env
- Supporting large customers in production
- Enterprise subscription add-ons (ODBC, Ranger, Sentry, Oracle, Teradata)

Notable features contributed:
- ANSI SQL syntax enhancements
- Execution engine improvements
- Security integrations
- Spill to disk
- Cost-Based Optimizer

https://www.starburstdata.com/presto-enterprise/
Performance
Built for Performance

Query Execution Engine:

- MPP-style pipelined in-memory execution
- **Columnar** and **vectorized** data processing
- Runtime query **bytecode compilation**
- Memory **efficient** data structures
- Multi-threaded multi-core execution
- Optimized readers for **columnar formats** (ORC and Parquet)
- Predicate and column projection **pushdown**
- Now also **Cost-Based Optimizer**
CBO in a nutshell

Presto Cost-Based Optimizer includes:

- support for **statistics** stored in Hive Metastore
- **join reordering** based on selectivity estimates and cost
- automatic **join type** selection (repartitioned vs broadcast)
- automatic left/right side selection for joined tables

https://www.starburstdata.com/technical-blog/
Statistics & Cost

Hive Metastore statistics:
- number of rows in a table
- number of distinct values in a column
- fraction of NULL values in a column
- minimum/maximum value in a column
- average data size for a column

Cost calculation includes:
- CPU
- Memory
- Network I/O
Join type selection

From

- Distributed Join
  - reshuffle

Large table
  - Expensive to reshuffle

Small table
  - Cheap to broadcast

To

- Broadcast Join
  - no reshuffle!

Large table
  - Expensive to reshuffle

Small table
  - Cheap to broadcast

broadcast
Join left/right side decision

From

Distributed Join

Small table

Large table

Kept in distributed join operator memory

To

Distributed Join

Large table

Large table

Small table

Kept in distributed join operator memory
Join reordering with filter
Join tree shapes

Left deep

Join
Join
Join
Join

table A
table B

table C

table D

Bushy tree

Join
Join
Join
Join

table A

table B

table C

table D
Benchmark results

https://www.starburstdata.com/presto-benchmarks/
Benchmark results

- on average 7x improvement vs EMR Presto
- EMR Presto cannot execute many TPC-DS queries
- All TPC-DS queries pass on Starburst Presto

https://www.starburstdata.com/presto-aws/
Recent CBO enhancements

- Deciding on semi-join distribution type based on cost
- Support for outer joins
- Capping a broadcasted table size
- Various minor fixes in cardinality estimation
- ANALYZE table (native in Presto)
- Stats for AWS Glue Catalog (exclusive from Starburst)
Current and Future work
What’s next for Optimizer

● Stats support
  ○ Improved stats for Hive
  ○ Stats for DBMS connectors and NoSQL connectors
  ○ Tolerate missing / incomplete stats

● Core CBO enhancements
  ○ Cost more operators
  ○ Adjust cost model weights based on the hardware
  ○ Adaptive optimizations
  ○ Introduce Traits

● Involve connectors in optimizations
Involving Connectors in Optimization
History and Current State

- Original motivation: partition pruning for queries over Hive tables
- Simple range predicates and nullability checks passed to connectors. Modeled as `TupleDomain`

\[
((\text{col0 \ BETWEEN \ ? \ AND \ ?}) \ OR \ (\text{col0 \ BETWEEN \ ? \ and \ ?}) \ OR \ ...)) \\
\text{AND} \\
((\text{col1 \ BETWEEN \ ? \ AND \ ?}) \ OR \ (\text{col1 \ BETWEEN \ ? \ and \ ?}) \ OR \ ...)) \\
\text{AND} \\
... \\
\]
Partial evaluation of non-trivial expressions
  - Bind only known variables
  - Result in "true/false/null" or "can't tell". E.g.,

\[
f(a, b) := \text{lower}(a) \text{ LIKE} ‘\text{john}%’ \text{ AND } b = 1
\]

\[
f(‘Mary’, ?) \rightarrow \text{false} \rightarrow \text{can prune}
\]

\[
f(‘John S’, ?) \rightarrow b = 1 \rightarrow \_\_\_(ツ)\_/\_\_
\]
Beyond Simple Filter Pushdown...

- Dereference expressions. E.g., x.a > 5
- Array/map subscript. E.g., a[ ‘key’ ] = 10
- Complex filters and projections
- Aggregations
- Joins
- Limit: https://github.com/prestosql/presto/pull/421
- Sampling
- Others…

https://github.com/prestosql/presto/issues/18
Rule 1

Pattern

Result

B
C

C'
B'

A
B
C

D
E
F

A
C'

D
E
F
G
Rule 2

Pattern: B → E → ? → B’ → E’
Result: E’ → B’ → ?

Diagram:

A → C’ → D → E → G

A → C’ → E’ → B” → F → G
SELECT count(*)
FROM t
WHERE x.f > 5 AND y LIKE 'a%b'

Table t
x :: row(f bigint, g bigint)
y :: varchar(10)
New Connector APIs

`applyFilter(ConnectorTableHandle table, Expression filter)`

`applyLimit(ConnectorTableHandle table, long limit)`

`applyAggregation(ConnectorTableHandle table, List<Aggregation> aggregates)`

`applySampling(ConnectorTableHandle table, double samplingRate)`

...
Performance Benefits (?)

- Better support for sophisticated backend systems
  - Druid, Pinot, ElasticSearch
  - SQL databases
- Improved performance for columnar data formats (Parquet, ORC)
ORC Performance Improvements

https://github.com/prestosql/presto/pull/555
Project Roadmap

- Coordinator HA
- Kubernetes
- Dynamic filtering
- Connectors
  - Phoenix
  - Iceberg
  - Druid
- TIMESTAMP semantics
- And more… [https://github.com/prestosql/presto/labels/roadmap](https://github.com/prestosql/presto/labels/roadmap)
Getting Involved

● Join us on Slack
  ○ Invite link: https://prestosql.io/community.html

● Github: https://github.io/prestosql/presto

● Website: https://prestosql.io
Further reading

https://www.starburstdata.com/presto-newsletter/

https://fivetran.com/blog/warehouse-benchmark

https://www.concurrencylabs.com/blog/starburst-presto-vs-aws-emr-sql/


https://virtuslab.com/blog/benchmarking-spark-sql-presto-hive-bi-processing-googles-cloud-dataproc/
Thank You!

www.prestosql.io

www.starburstdata.com

@prestosql  @starburstdata