ONLINE , AROUND-THE-WORLD OPEN SOURCE DATABASE CONFERENCE

PERCONA LIVEONLINE

MAY 12 - 13th

2021

5 ways Facebook's ludicrous usage drives Presto innovation

Hello!

I am Ariel Weisberg

I am here because I forgot presentations are hard

You can find me at prestodb.slack.com

1.

What’s a Presto?

An ANSI SQL Compute engine

Presto TL;DR

- Apache 2.0 licensed distributed query engine
- Owned by the Linux Foundation
- Pluggable connectors allow you to query data where it already resides
- Consistent ANSI SQL interface over multiple connectors
- Horizontally and vertically scalable

Presto at FB

- Primarily internal usage
- Diverse workload
    - O(Tens of thousands) of users issuing queries (directly or indirectly)
    - O(Thousands) of query authors
    - O(Hundreds of thousands unique queries)
- Repeating “Batch” workload
    - Graph of data processing pipelines O(Tens of thousands)
    - Hourly, daily, monthly etc.
    - Must land the entire graph every day
- Adhoc/Interactive
    - Dashboards, alerts, Jupyter notebooks, CLI or similar
    - Other tools and systems

Presto economics

- Efficiency
  - Workload wants to grow (new use cases, organic growth)
  - Capacity growth costs money
  - Efficiency decreases required capacity growth
- Memory is at a premium
  - Presto originally “in-memory”
  - Workload grew to fit (and exceed) available memory
  - Oops, turns out ¼ memory hardware is more efficient
- Minimizing user impact
  - “Fix your query” as a last resort
  - Execute problematic queries without tuning

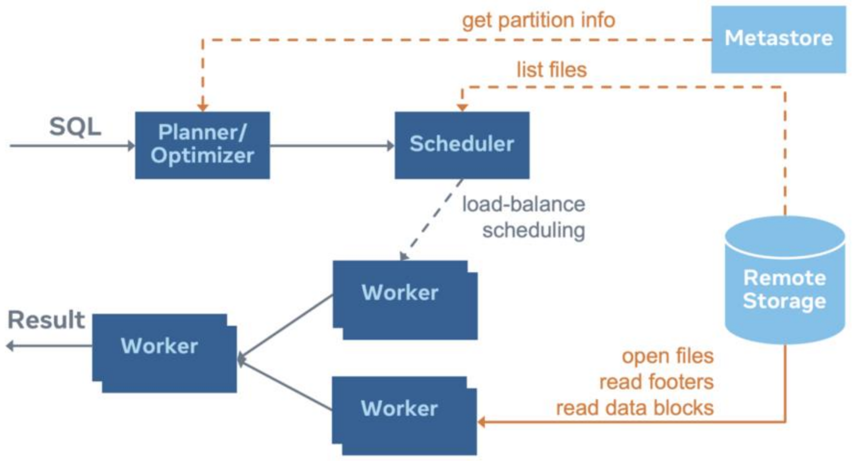
2.

RaptorX

Go real fast with caching

Presto Today: Disaggregated Storage and Physics!

- Data is growing exponentially faster than use of compute
- Resultant Industry trend towards scaling storage and compute independently e.g., Snowflake on S3, AWS EMR on S3, Big Query on Google Storage etc.
- Helps customers and cloud providers scale independently, reducing cost
- Data for querying and processing needs to be streamed from remote storage nodes
- New challenge for query latency as scanning huge amounts of data over the wire is going to be I/O bound when the network is saturated

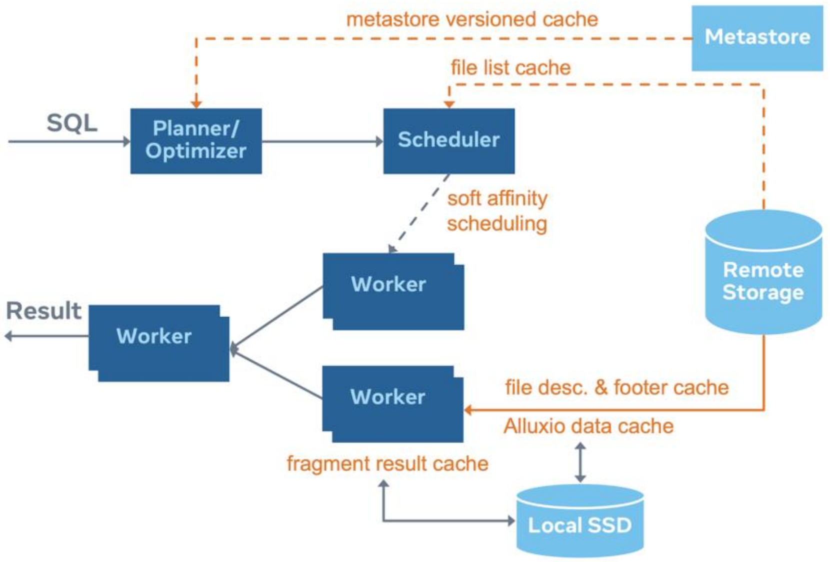


CAPTION: *Presto Servers need to retrieve data from remote storage*

Distance has increased between compute and storage and overcoming Physics is hard

RaptorX: Hierarchical Caching for Interactive Workloads!

- RaptorX's goal is to create a no migration query acceleration solution for existing Presto customers so that existing workloads can benefit seamlessly
- Challenge is to accelerate interactive workloads that are petabyte scale without replicating data
- Found top opportunities to increase performance by doing a comprehensive audit of query lifecycle
- Caching is obviously the answer and not new - however is a lot of work to manage e.g., cache invalidation etc.!
- What's new is '*true no-work*' query acceleration; Responses are returned upto 10x faster with no change in pipelines or queries

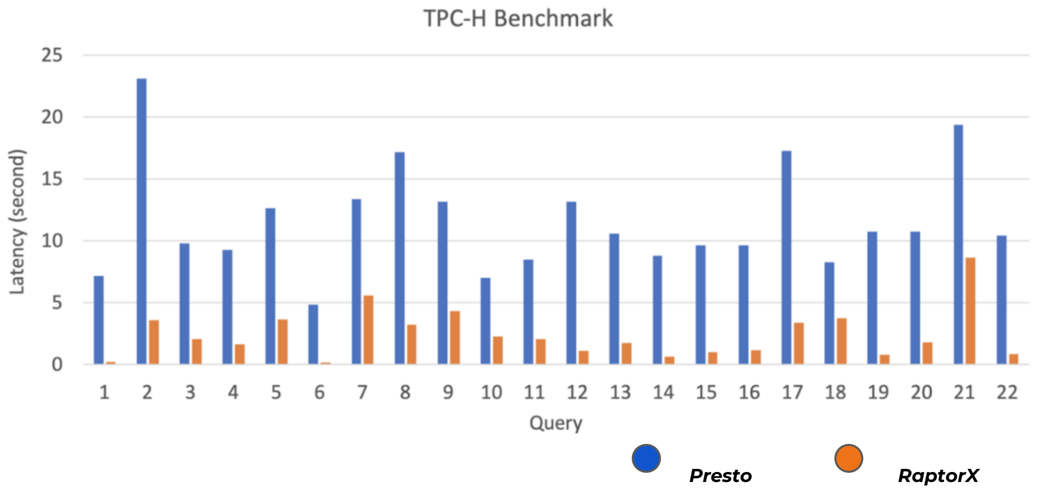


CAPTION: *Presto with RaptorX smartly caches at every opportunity*

Reduce distance between compute and storage intelligently!

RaptorX: 10X faster than Presto!

- We see more than 10X increase in query performance with RaptorX in production at Facebook
- TPC-H benchmark between Presto and RaptorX also confirms the performance difference!
- Test was run on a 114 node cluster with 1TB SSD and 4 threads per task
- TPC-H scale factor was 100 in remote storage
- Scan and aggregation heavy queries show 10X improvement (Q1, Q6, Q12-16, Q19 and Q22)
- Join heavy queries show between 3X and 5X improvement (Q2, Q5, Q10, or Q17)



CAPTION: *Presto + Cache i.e. RaptorX is on average 10X faster*

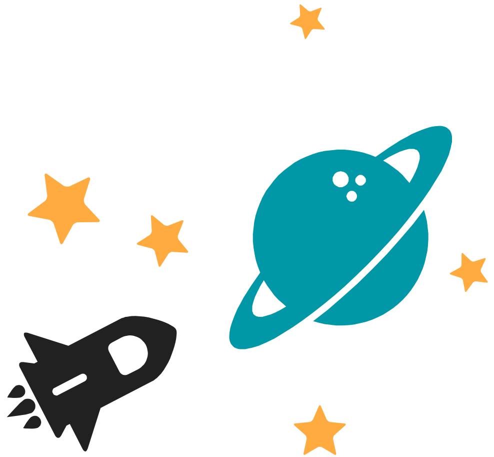
10X better performance with no change in pipelines!

RaptorX economics

- Replaces 4 other tools inside FB!
- In house development is incredibly expensive, redundancy increases cost, reduces quality
- Provides a single, popular, fully supported SQL dialect to more use cases
- Operational simplicity and efficiency

RaptorX

https://prestodb.io/prestoconday2021.html#RaptorX_Building_a_10X_Faster_Presto



3.

“Large Batch”

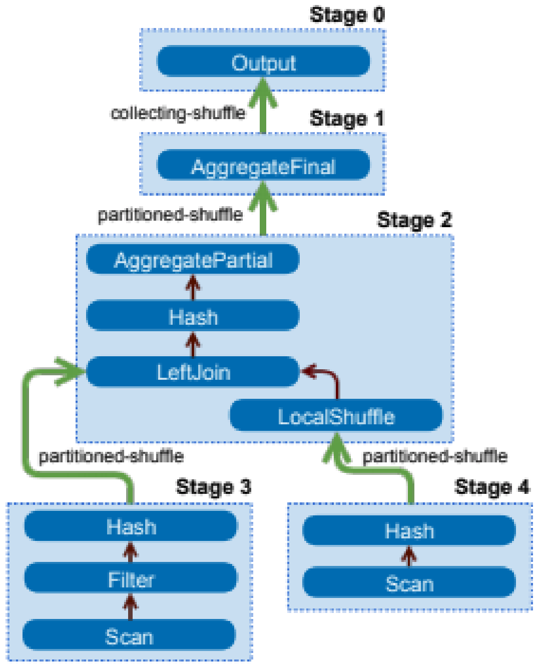
Presto bursting at the seams

Presto’s “Large Batch” approach

- Large Batch
  - Long running (hours to days)
  - CPU heavy (hundreds of CPU days to years)
  - High memory (>2.5tb)
  - Skewed (>5gb memory per node)
- Presto-on-Spark - Presto’s Java eval running on Spark as an RDD
- Presto Unlimited - MapReduce on Presto w/o full fault tolerance
- Operator Spilling - Local/remote disk to extend memory for skewed queries

Presto Architecture Overview

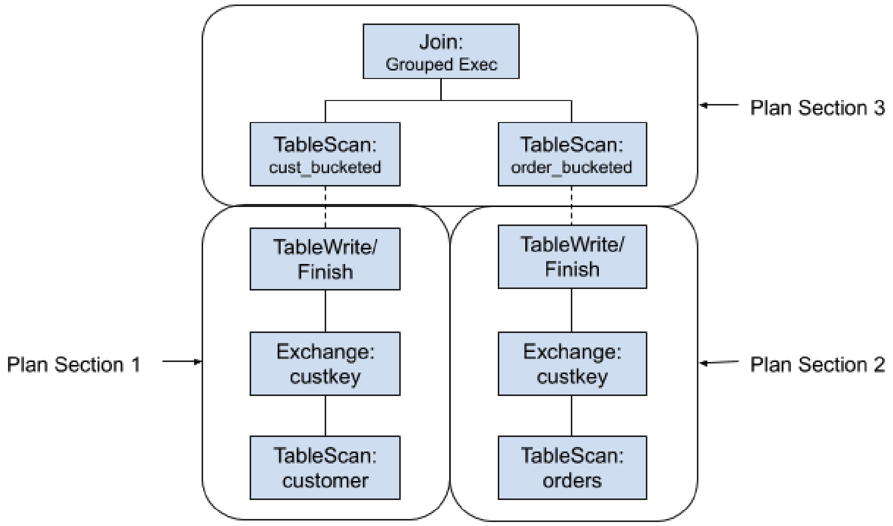
- Designed for interactivity
- Classic MPP architecture
- In-memory streaming shuffle
  - Low latency
  - More operations can be done in parallel
- Standalone, multi-tenant service
  - Always “warm”, no “startup” delay



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

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 (>2.5TB)
- Increases reliability by allowing partial failure recovery
- Can be run on existing Presto deployments



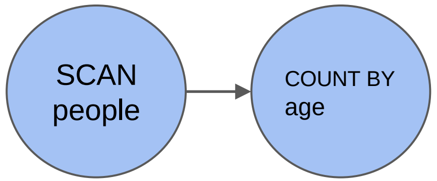
https://prestodb.io/blog/2019/08/05/presto-unlimited-mpp-database-at-scale

Presto-on-Spark 1000 feet view

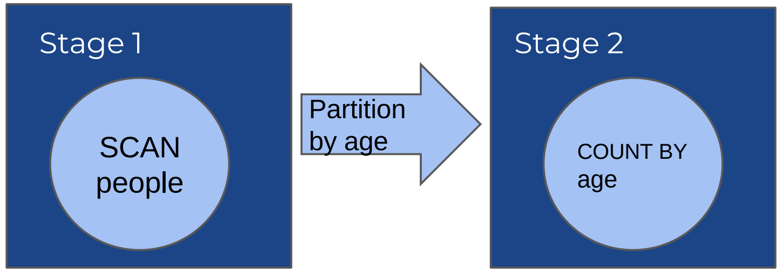
Presto SQL

```
SELECT
  age,
  count(*)
FROM people
GROUP BY age
```

Presto Logical Plan

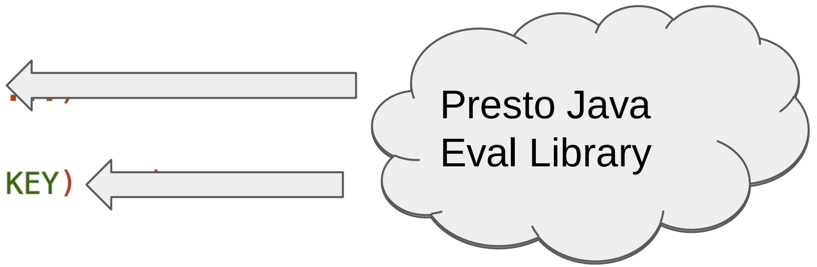


Presto Distributed Plan



RDD

```
// RDD
rdd
.mapToPair(... run Stage 1 operators (SCAN) ...)
.partitionByKey()
.reduce(... run Stage 2 operators (COUNT BY KEY)
.collect()
```



Focusing on Presto-on-Spark

- Obsoletes Presto Unlimited except for startup time
- Provides all the things
  - Presto SQL queries that scale instead of fail
  - Hardware fungibility between Presto and Spark (2.2x faster wall time!)
  - Isolation between queries via containerization and a dedicated Spark Driver per query
  - Fault tolerance
  - Fine grained resource allocation and scheduling
  - Operational simplicity
    - One cluster instead of many
    - Easy support for elastic capacity
  - Scale query execution beyond 600 nodes

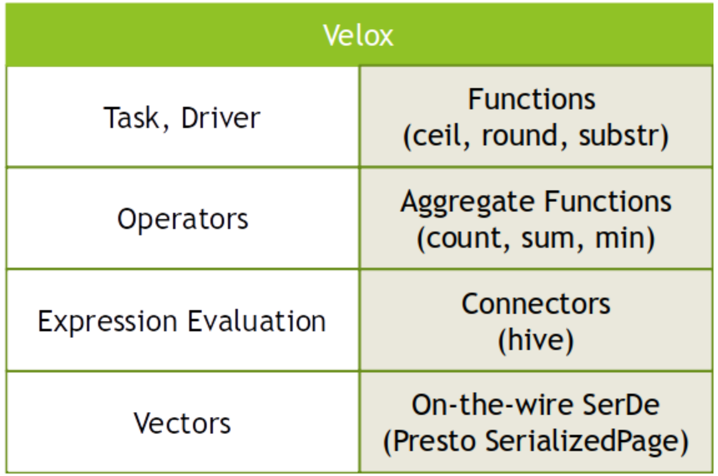
3.

Velox

Things you should never do, rewriting from scratch

Introducing Velox

- New C++ Vectorized execution engine
- No SQL parser
- No optimizer
- Inputs:
  - Single stage query plan
  - Expression tree
- Outputs:
  - Vectors
  - Serialized vectors



Velox	
Task, Driver	Functions (ceil, round, substr)
Operators	Aggregate Functions (count, sum, min)
Expression Evaluation	Connectors (hive)
Vectors	On-the-wire SerDe (Presto SerializedPage)

Velox Library

- **Not** intended to fully replace compute engines
- Provide state of the art and universal building blocks for compute
  - Embed in various products and services for SQL evaluation
  - Hybrid
- Why?
  - Efficiency and latency
  - Consistency
  - Reusability and Engineering Efficiency
- Goal is to partially or fully replace other eval engines
  - Presto
  - Spark
  - Stream processing
  - Monitoring engines
  - ML/AI
  - Custom applications

Velox economics

- Eval compatibility across engines
- Efficiency and stability
  - C++
  - Memory management
  - Benefits of a complete rewrite
- Efficiency wins shared across more use cases
- Faster wall time for queries

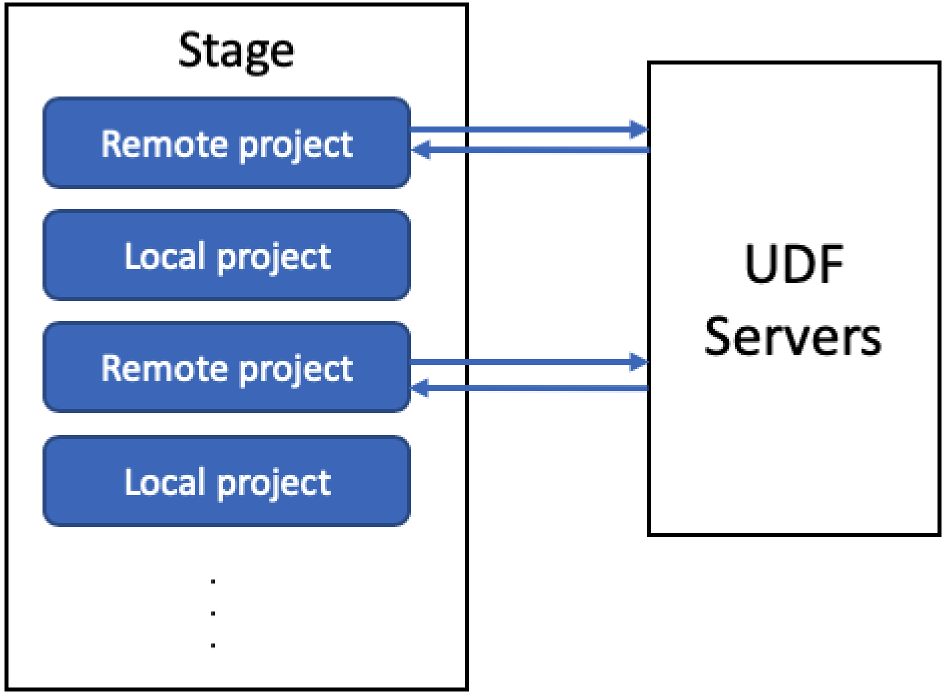
4.

Remote UDFs

Sandbox all the things

Existing UDFs

- Loaded at deployment time
- Run in process with limited isolation
- Blocking UDFs are impractical
- Don’t want to police UDF quality


Stage
Remote project
Local project
Remote project
Local project
UDF Servers

Remote UDF economics

- Shared pool of UDFs across multiple systems (Presto, Spark etc.)
- UDFs in multiple languages
- Scale disaggregated UDF capacity separately
- Design discussion issue #14053

5.

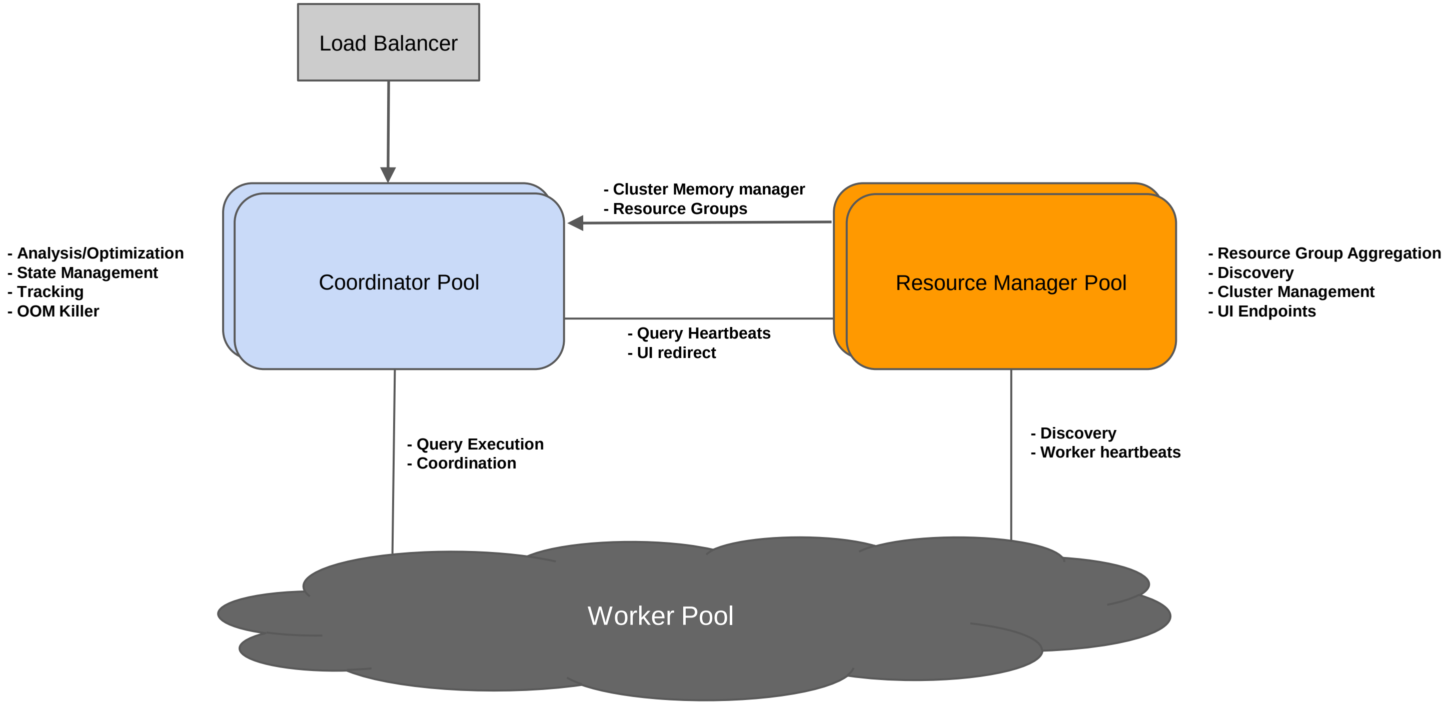
Fireball

Horizontally scale all the things

Presto cluster layout

- 1 coordinator
- 200-1000 nodes
- Coordinator runs many queries concurrently
    - Easily overloaded
    - Full GCs and timeouts fail all currently running queries
    - Retries and toxic workloads create large blast radius
- Need more capacity? Add more clusters

Fireball Architecture

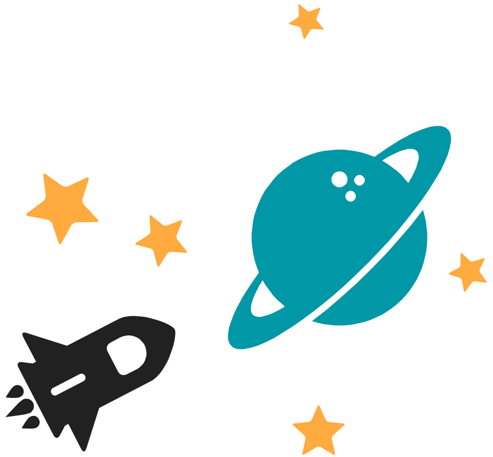


Fireball economics

- Operational simplicity
  - One cluster per region
  - Smaller blast radius for toxic workloads
- Efficiency
  - One big resource pool/less fragmentation
- No SPOF
- “Eliminate” coordinator bottleneck
- Support low CPU/Memory coordinators

Fireball

https://prestodb.io/prestoconday2021.html#Disaggregated_Coordinator



6.

Verification

Bonus thing!

Verifier

- Shadows production workload comparing two versions
- Runs nightly and as a release blocking process
- Suspected errors are semi-manually verified
- Not a complete solution
- Most similar to fuzz testing
- Finds weird things that only exist in a real deployment
    - Interactions between compute engines and different data formats
    - That one query in the entire workload that triggers that planner bug that is 5 years old

Thanks!

Any questions?

You can find me at:

- prestodb.slack.com

THANK YOU !

ONLINE, AROUND-THE-WORLD OPEN SOURCE DATABASE CONFERENCE

PERCONA

LIVEONLINE

MAY 12 - 13th

2021