facebook

INFRASTRUCTURE

Presto @ Facebook

Past, Present, and the Future

Nezih Yigitbasi

Looking Back at 2018

- 34 releases (0.192 to 0.215)
- ~2700 commits
- ~95 new contributors (total 358)
- ~600 new forks (total 2884)

Presto @ Facebook

- Multiple use cases
    - warehouse ETL and ad-hoc analytics
    - dashboards
    - analytics backend for A/B testing
    - analytics backend for user facing products
- 1000s of nodes across several data centers
- 100s of PBs and quadrillions of rows processed per day
- > 80% of new warehouse ETL workloads on Presto

Making Presto More Efficient
(a.k.a. Project Aria)

- Apply column store state of the art to Presto
  - CPU-friendly loops, vectorization, cache-consciousness, etc.
  - work with bounded memory
- Much better pushdown for complex types
- ~2.6x CPU efficiency win for basic star schema query

Breaking the Memory/Duration Barrier
(a.k.a. Presto Unlimited)

- Support running queries above distributed memory limits
  - bucket-by-bucket execution helps, but needs bucketed tables
  - materialize exchanges before joins & aggregations
- Partial recovery support for long running queries
  - retry failed “lifespans” in a task
    - is the output data consumed?
    - reschedule failed lifespans to a different task
    - cleanup partial output

Coordinator Scalability

- Dispatcher
- Transport improvements
    - HTTP/2 (RFC 7540)
    - SMILE
    - Afterburner

Coordinator Scalability: Dispatcher

- Coordinator does a lot today
  - parse, analyze, queue, manage workers & work, etc.
- Pull out the queueing & resource management
- Offload the coordinator
- Potentially better resource management decisions
  - multiple clusters per data center

Coordinator Scalability: Transport Improvements

- HTTP/2 (RFC 7540)
  - binary encoding, header compression, session multiplexing, etc.
- SMILE
  - Jackson’s binary encoding
  - no change needed in application
- Afterburner
  - Jackson module for code generation

* all disabled by default

Memory Management

- System pool is gone!
- Better visibility into tracked memory
    - /v1/cluster/memory and /v1/memory/{pool_name} endpoints
- Leak detector & more resilient OOM killer

- Reserved pool is next
    - major source of inefficiency
    - can already be turned off

Cost Based Optimization

- Initially contributed by Teradata/Starburst

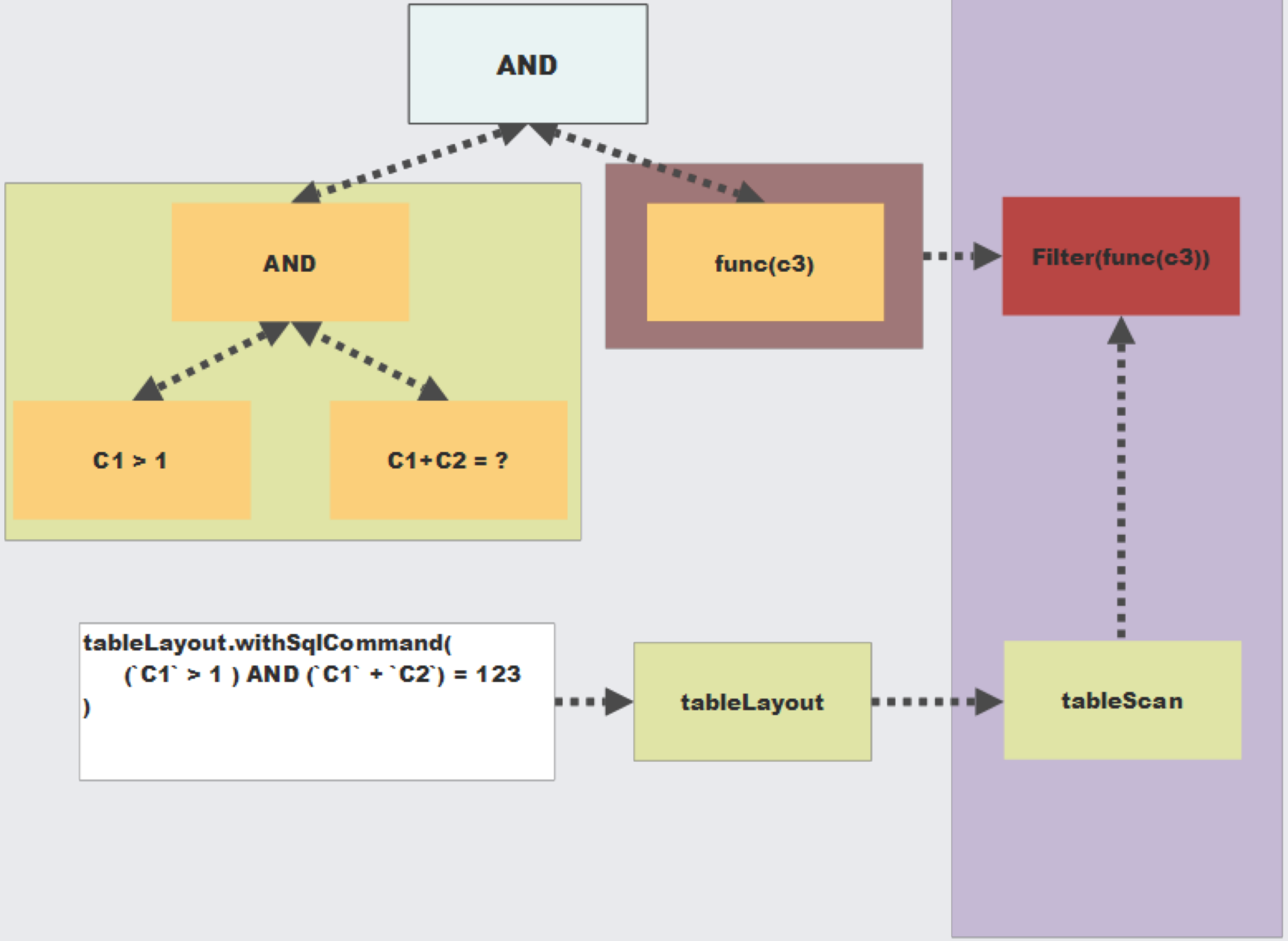
- Broadcast or distributed join
- Order of relations in a join
  - improves memory usage significantly for certain joins
- Reorder inner joins stacked on top of each other

Function Support

- Share SQL for common functions
- Call external services
- Custom functions a.k.a. UDFs
    - performance & isolation concerns

Connectors to Participate in Optimization

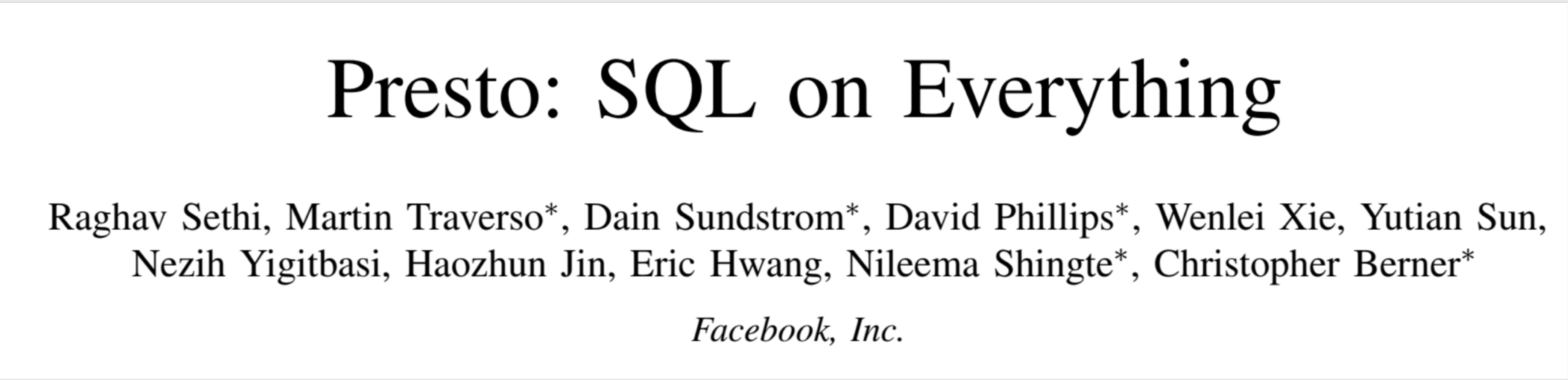
- Today connectors are like simple data/metadata sources
  - why not better utilize connector capabilities?
- Ask the connectors about the rules they support
- Let them rewrite subtrees of the plan
- Push down filter/project/aggregation to connectors



```
tableLayout.withSqlCommand(
    (`C1` > 1 ) AND (`C1` + `C2`) = 123
)
```

And Many More ...

- Warnings framework
- Improvements to geospatial functionality
  - distributed spatial join, performance optimizations, support for WKB/EntGeoPolygon formats, etc.
- Improvements to coordinator web UI
- Raptor V2 [WIP]
- Elasticsearch connector
- Kudu connector
- S3 Select support



Presto: SQL on Everything

Raghav Sethi, Martin Traverso[*], Dain Sundstrom[*], David Phillips[*], Wenlei Xie, Yutian Sun, Nezih Yigitbasi, Haozhun Jin, Eric Hwang, Nileema Shingte[*], Christopher Berner[*]

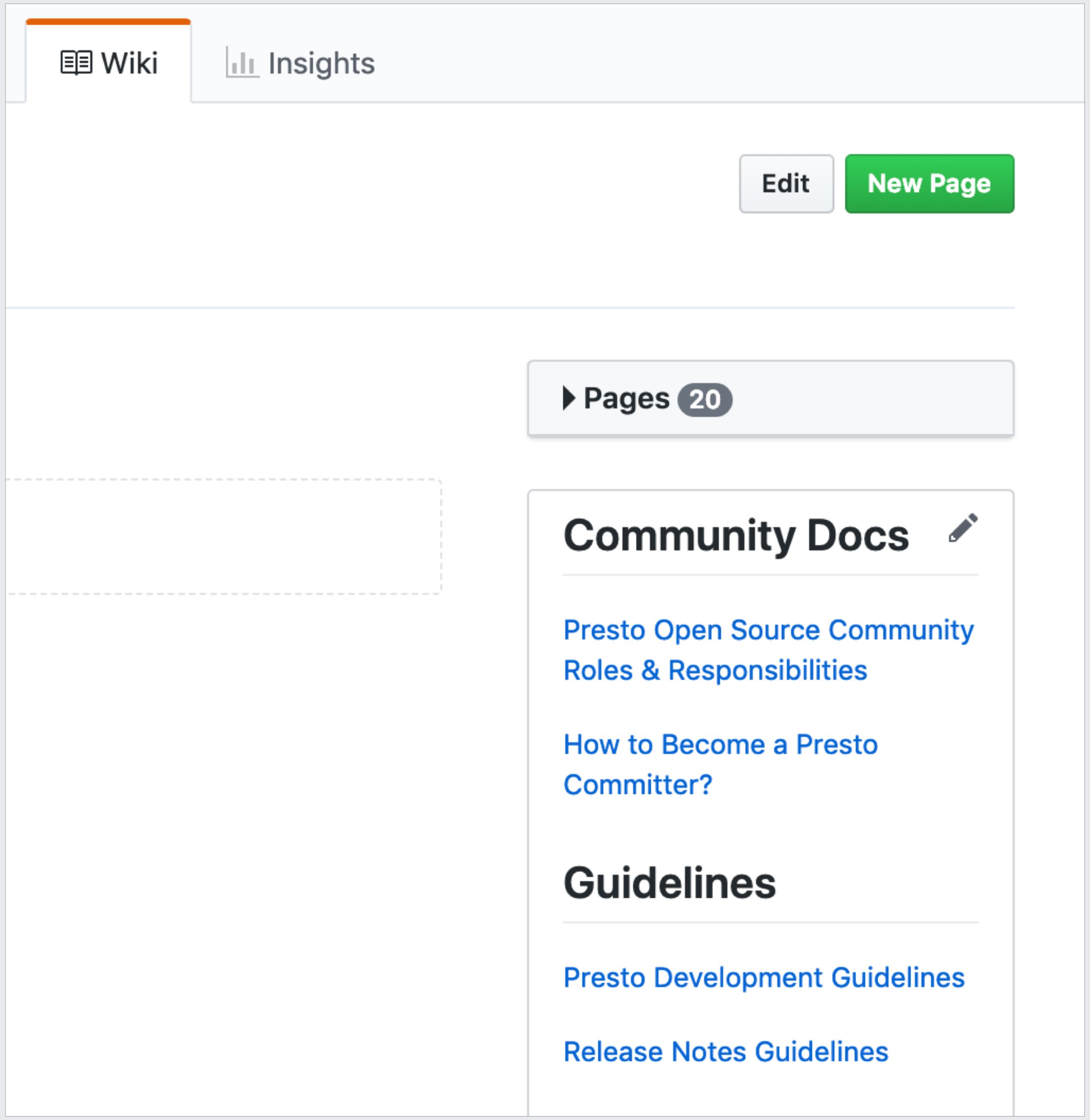
Facebook, Inc.

http://tinyurl.com/presto-paper

Releases

- Each release is verified extensively @ scale
- Improve the verifier tool
- Performance & reliability testing
- Branch-based release model

Community



Community Docs

Presto Open Source Community Roles & Responsibilities

How to Become a Presto Committer?

Guidelines

Presto Development Guidelines

Release Notes Guidelines

facebook

Thank You