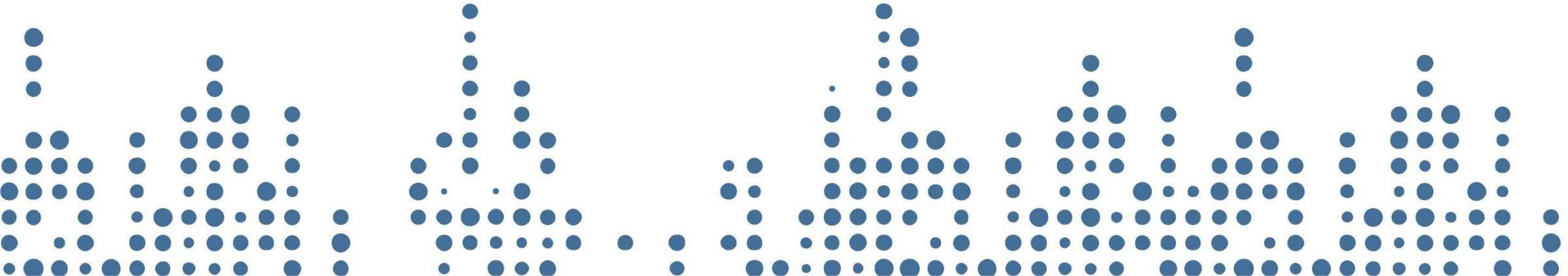
posit™

[DRAFT] The Future Roadmap for the Composable Data Stack

Wes McKinney

Data Council Austin 2024



Agenda

- My background
- Posit and Me
- Composable data systems: an overview
- Active growth areas in 2024
- Opportunities and Predictions

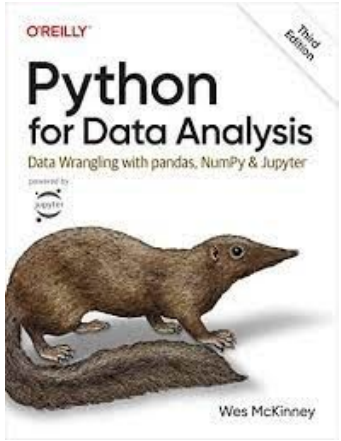
Me

posit™

Principal Architect

Composed Ventures

General Partner



Author

pandas

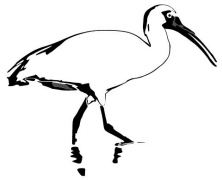
Creator

APACHE ARROW

Co-creator

VOLTRON DATA

Co-founder, Advisor



Ibis

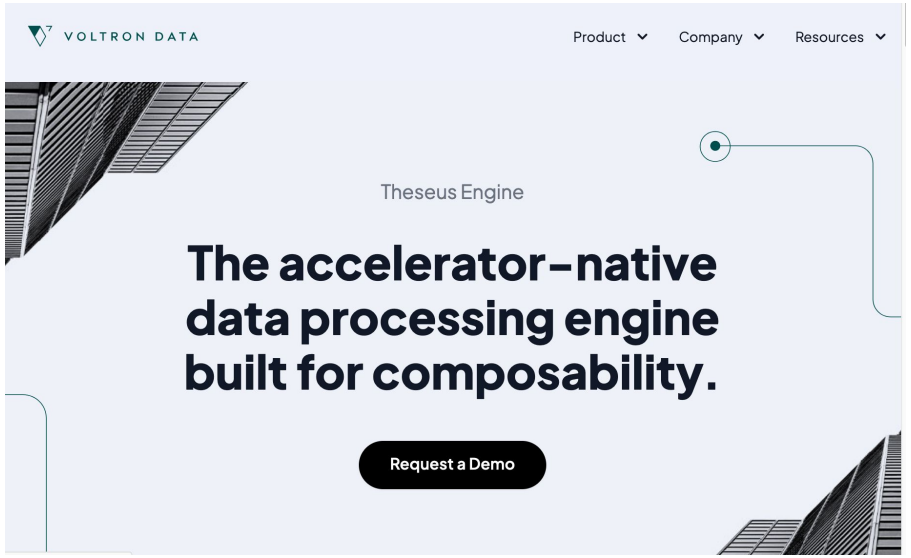
Creator

LanceDB

Advisor, Investor

Voltron Data (2021 —)

- Unlocking the potential of GPU accelerated analytics for large scale workloads
- Enterprise support for Apache Arrow
- Open source partnerships (Meta, Snowflake, others)
- $115M raised, 130 headcount and growing



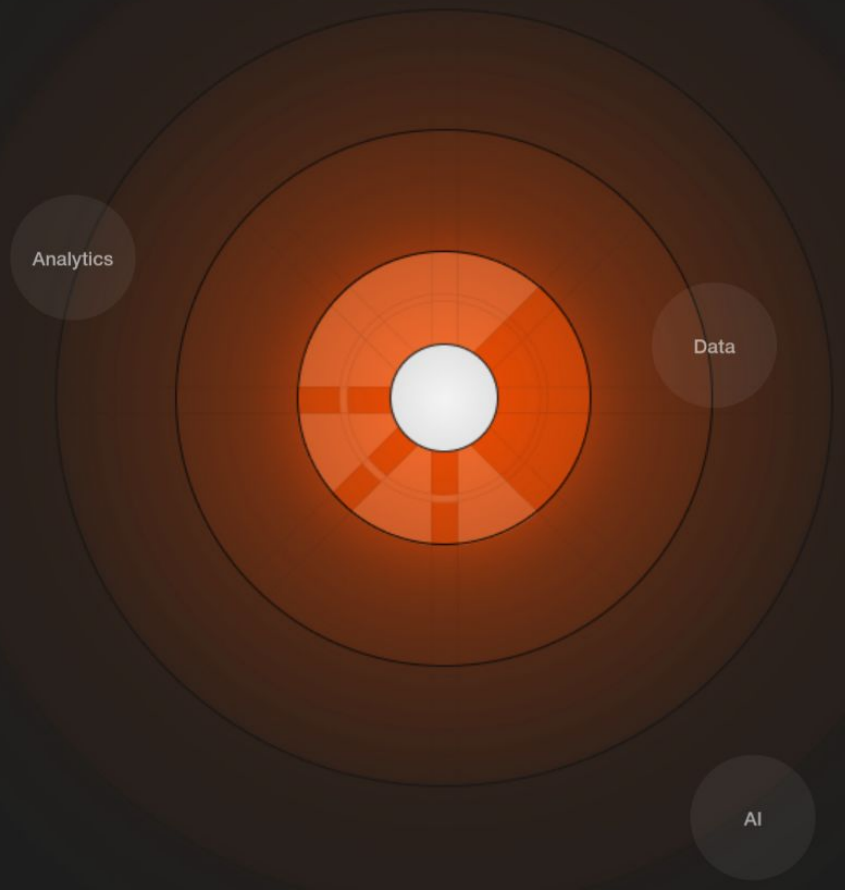
Composed Ventures

Portfolio

Investing in early-stage data infrastructure, AI, and ML companies.

Accelerating the growth of the Composable Data Stack

View Portfolio



Posit PBC

- Founded 2009, originally as RStudio
- 300-person, remote-first company
- Open source software for polyglot data science and technical communication
- Certified B corp, no plans to go public or be acquired
- Designing for long-term resiliency, creating a 100-year company

How Posit Makes Money

- Making open source work in the enterprise
- Products for managing the data science lifecycle
  - **Posit Workbench**: Managed IDEs and Jupyter Notebooks with enterprise-grade security & compliance
  - **Posit Connect:** Publish and share interactive data apps, notebooks, and reports
  - **Posit Package Manager:** securely manage deploy open source and proprietary Python and R packages

My History with Posit

- 2016: Hadley Wickham and I make Feather file format
- 2018: I partner with RStudio to create non-profit Ursa Labs for Arrow development
- 2020: Ursa team spins out into startup, becomes Voltron Data in 2021
- 2022: RStudio becomes Posit, embracing polyglot future
- 2023: I rejoin Posit as a principal architect

The reality is that Hadley and I think the “language wars” are stupid when the real problem we are solving is human user interface design for data analysis. … The programming languages are our medium for crafting accessible and productive tools. It has long been a frustration of mine that it isn’t easier to share code and systems between R and Python.

…

I found that we share a passion for the long-term vision of empowering data scientists and building a positive relationship with the open source user community. Critically, RStudio has avoided the “startup trap” and managed to build a sustainable business while still investing the vast majority of its engineering resources in open source development.

https://ursalabs.org/blog/announcing-ursalabs/

The reality is that Hadley and I think the “language wars” are stupid when the real problem we are solving is human user interface design for data analysis. … The programming languages are our medium for crafting accessible and productive tools. It has long been a frustration of mine that it isn’t easier to share code and systems between R and Python.

…

I found that we share a passion for the long-term vision of empowering data scientists and building a positive relationship with the open source user community. Critically, RStudio has avoided the “startup trap” and managed to build a sustainable business while still investing the vast majority of its engineering resources in open source development.

https://ursalabs.org/blog/announcing-ursalabs/

Sponsors

TWO SIGMA

NVIDIA

Bloomberg

RESEARCH

intel

RStudio

Creative Solutions for Funding OSS Work

The Composable Data Management System Manifesto

Pedro Pedreira
Meta Platforms Inc.
pedroerp@meta.com

Orri Erling
Meta Platforms Inc.
oerling@meta.com

Konstantinos Karanasos
Meta Platforms Inc.
kkaranasos@meta.com

Scott Schneider
Meta Platforms Inc.
scottas@meta.com

Wes McKinney
Voltron Data
wes@voltrondata.com

Satya R Valluri
Databricks Inc.
satya.valluri@databricks.com

Mohamed Zait
Databricks Inc.
mohamed.zait@databricks.com

Jacques Nadeau
Sundeck
jacques@sundeck.io

VLDB 2023

What is a “Composable Data System”?

- Builds with open standards and protocols
- Designed around modularity, reuse, and interoperability with other systems that use shared interfaces
- Resists “vertical integration”, builds in a virtual cycle with relevant open source ecosystem projects

“We envision that by decomposing data management systems into a more modular stack of reusable components, the development of new engines can be streamlined, while reducing maintenance costs and ultimately providing a more consistent user experience. By clearly outlining APIs and encapsulating responsibilities, data management software could more easily be adapted, for example, to leverage novel devices and accelerators, as the underlying hardware evolves. By relying on a modular stack that reuses execution engine and language frontend, data systems code could provide a more consistent experience and semantics to users, from transactional to analytic systems, from stream processing to machine learning workloads."

Pedrera et al. 2023

“We envision that by decomposing data management systems into a more modular stack of reusable components, the development of new engines can be streamlined, while reducing maintenance costs and ultimately providing a more consistent user experience. **By clearly outlining APIs and encapsulating responsibilities, data management software could more easily be adapted, for example, to leverage novel devices and accelerators, as the underlying hardware evolves.** By relying on a modular stack that reuses execution engine and language frontend, data systems code could provide a more consistent experience and semantics to users, from transactional to analytic systems, from stream processing to machine learning workloads."

Pedrera et al. 2023

“We envision that by decomposing data management systems into a more modular stack of reusable components, the development of new engines can be streamlined, while reducing maintenance costs and ultimately providing a more consistent user experience. By clearly outlining APIs and encapsulating responsibilities, data management software could more easily be adapted, for example, to leverage novel devices and accelerators, as the underlying hardware evolves. **By relying on a modular stack that reuses execution engine and language frontend, data systems code could provide a more consistent experience and semantics to users, from transactional to analytic systems, from stream processing to machine learning workloads."**

Pedrera et al. 2023

Why now?

- 1st-Gen Big Data / Hadoop Era: 2006 - 2013
  - MapReduce popularizes disaggregated storage + compute
- 2nd-Gen 2013 - 2021
  - Vendors shift from proprietary software to delivery of services
  - Open source standards emerge and are popularized
  - Rapid progress in storage, networking, computing performance
- 3rd-Gen 2021 - … ?
  - Open standards for composability become widely accepted
  - Next-gen components emerge, existing systems start retrofitting with composable pieces

Why now?

“…We foresee that composability is soon to cause another major disruption to how data management systems are designed. We foresee that monolithic systems will become obsolete, and give space to a new composable era for data management.”

Pedrera et al. 2023

The Great Data Tool Decoupling™

- Thesis: over time, user interfaces, data storage, and execution engines will decouple and specialize
- In fact, you should really want this to happen
  - Share systems among languages
  - Reduce fragmentation and “lock-in”
  - Shift developer focus to usability
- Prediction: we’ll be there by 2025; sooner if we all get our act together

NYC R Conference 2015
https://www.youtube.com/watch?v=stIxbC7uIzM&t=264s

Scalability! But at what COST?

Frank McSherry
Unaffiliated

Michael Isard
Unaffiliated*

Derek G. Murray
Unaffiliated†

Abstract

We offer a new metric for big data platforms, COST, or the Configuration that Outperforms a Single Thread. The COST of a given platform for a given problem is the hardware configuration required before the platform outperforms a competent single-threaded implementation. COST weighs a system's scalability against the overheads introduced by the system, and indicates the actual performance gains of the system, without rewarding systems that bring substantial but parallelizable overheads.

We survey measurements of data-parallel systems recently reported in SOSP and OSDI, and find that many systems have either a surprisingly large COST, often hundreds of cores, or simply underperform one thread for all of their reported configurations.

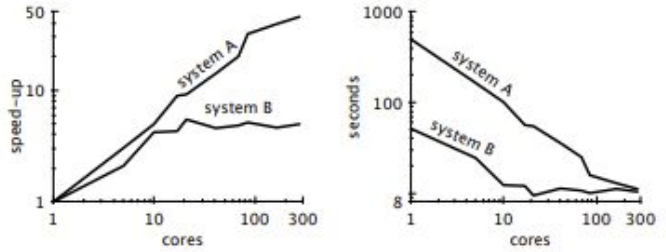


Figure 1: Scaling and performance measurements for a data-parallel algorithm, before (system A) and after (system B) a simple performance optimization. The unoptimized implementation "scales" far better, despite (or rather, because of) its poor performance.

While this may appear to be a contrived example, we will argue that many published big data systems more closely resemble system A than they resemble system B.

McSherry, Isard, Murray 2015

“The published work on big data systems… detail[s] their systems’ impressive scalability, [but] few directly evaluate their absolute performance against reasonable benchmarks. To what degree are these systems truly improving performance, as opposed to parallelizing overheads that they themselves introduce?”

- McSherry, Isard, Murray 2015

Glimpse of the Composable Landscape

Optimization

Apache Calcite™ ...?

Engines

DuckDB Velox DataFusion Voltron Data

Protocols

Apache Arrow

ADBC, Flight

Substrait

OpenLineage

Query Interface

Ibis malloy SQL dplyr

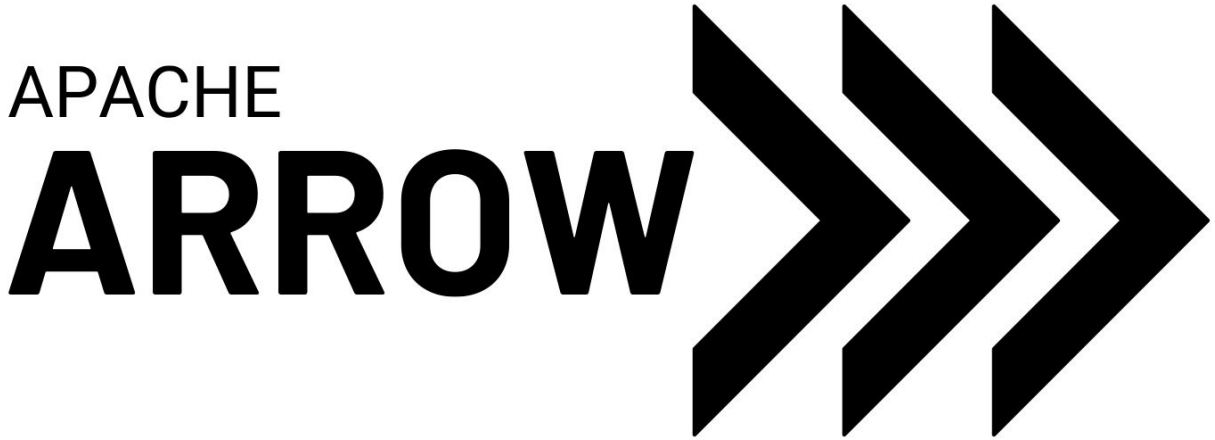
Storage

Iceberg Apache ORC Parquet LanceDB

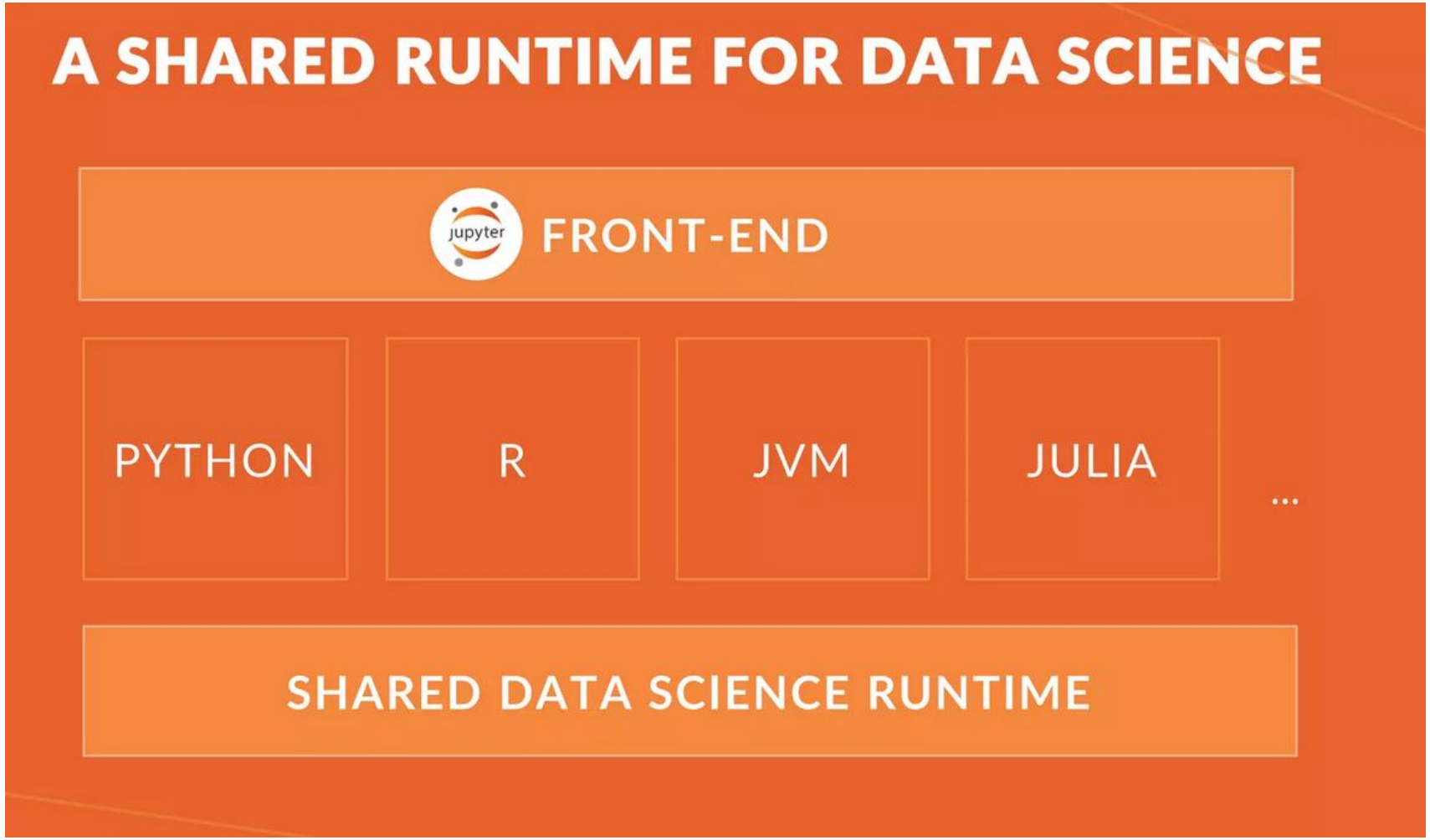
Missing pieces

- Distributed computing (Spark, Dask, Ray, “Serverless”)
- ETL modeling (dbt and others)
- Workflow / Infra Orchestration (Airflow, Dagster, Flyte, …)
- Development Environments
- Metadata Catalogues

2016: A Cross-Language Fast In-Memory Data Format



2017



https://www.slideshare.net/wesm/data-science-without-borders-jupytercon-2017

Don’t Hold My Data Hostage – A Case For Client Protocol Redesign

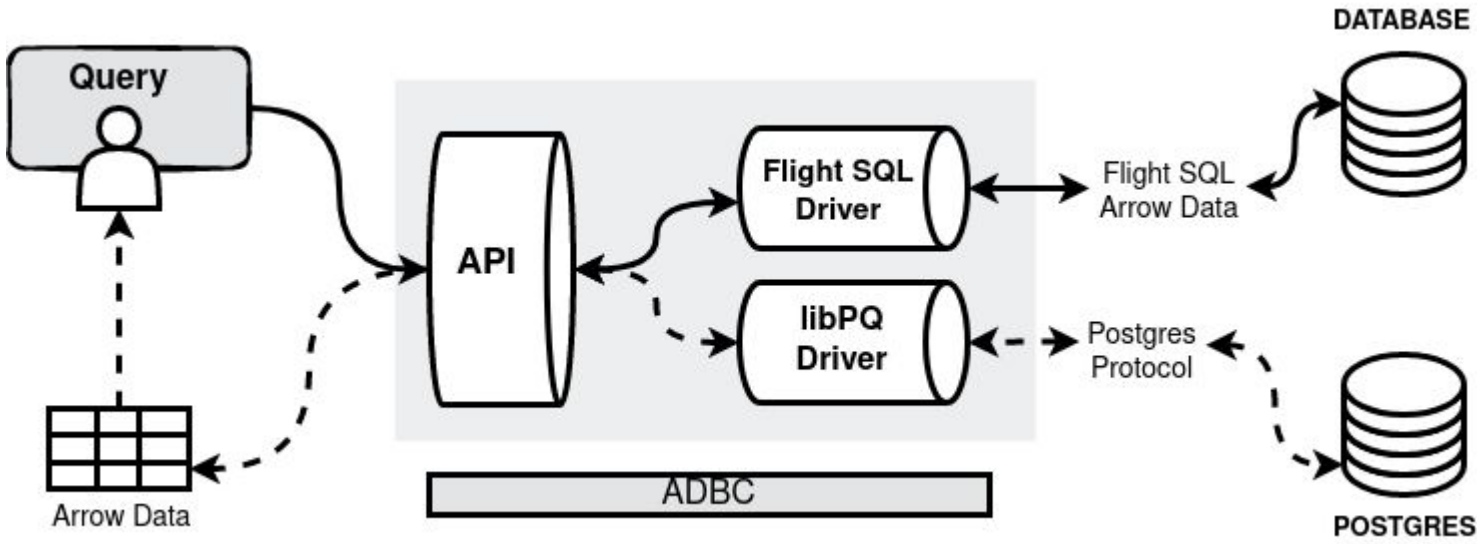
Mark Raasveldt
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
m.raasveldt@cwi.nl

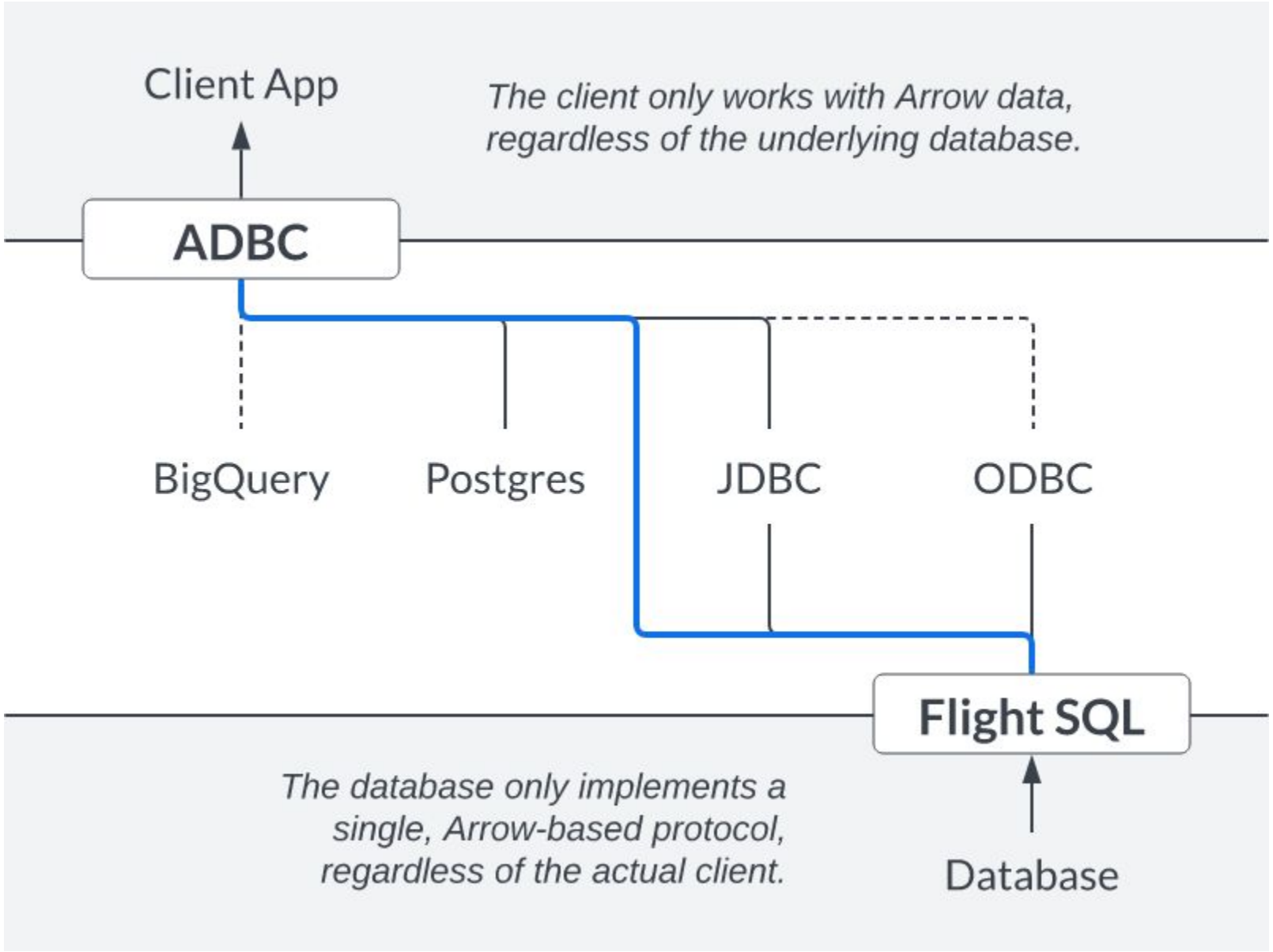
Hannes Mühleisen
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
hannes@cwi.nl

2017

Databases Equipped for the Composable Era

- ADBC : Arrow DataBase Connectivity
- FlightSQL: An Arrow-native wire protocol for SQL systems




Client App
The client only works with Arrow data, regardless of the underlying database.
ADBC
BigQuery
Postgres
JDBC
ODBC
Flight SQL
The database only implements a single, Arrow-based protocol, regardless of the actual client.
Database

DuckDB: an Embeddable Analytical Database

Mark Raasveldt
m.raasveldt@cwi.nl
CWI, Amsterdam

Hannes Mühleisen
hannes@cwi.nl
CWI, Amsterdam

2018

Layers of the Composable Cake

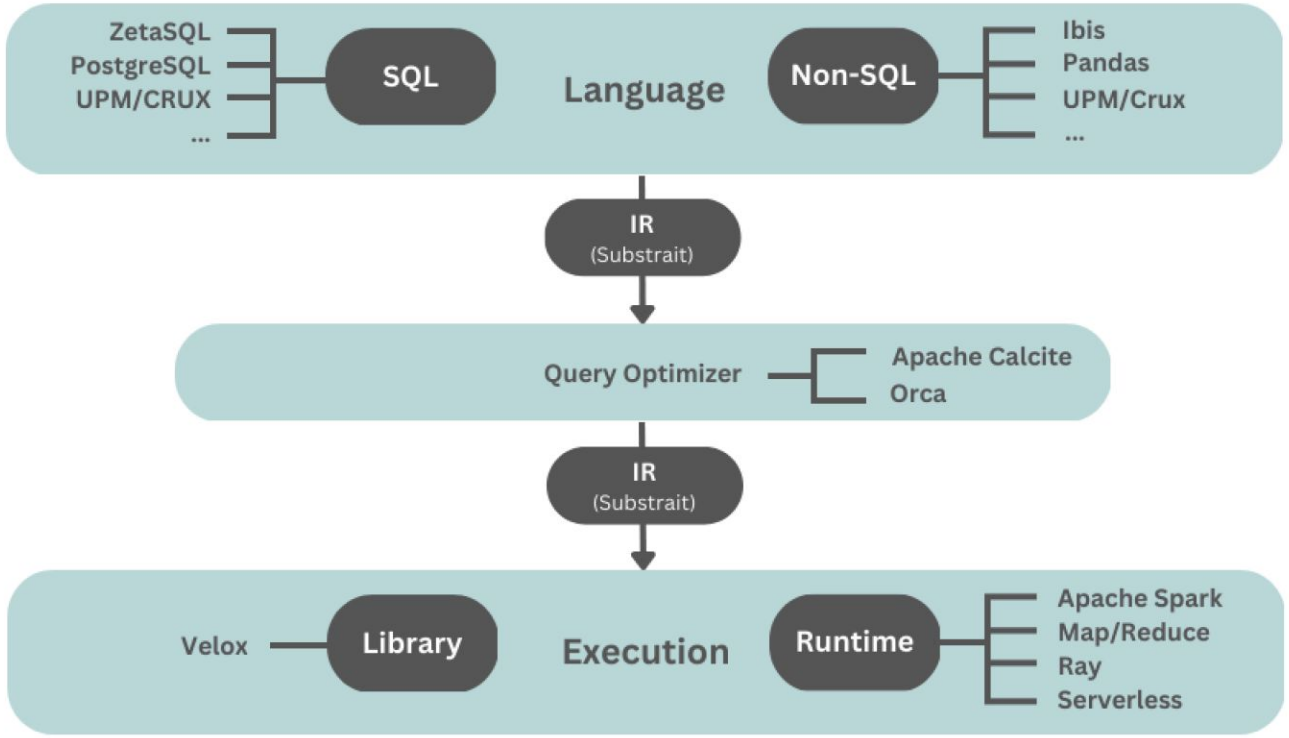


Figure 1: Open source modular data stack outline.

Modular Execution Engines

- Apache DataFusion (Rust)
- DuckDB (C++)
- Velox (C++)
- Theseus (C++ / CUDA)

Connecting Execution with Frontend + Optimizer

- Intermediate Representation (IR) for Queries

Substrait

https://substrait.io/

Modular Acceleration Projects

- Prestissimo (Velox in Presto)
- Apache DataFusion Comet (DataFusion in Spark)
- Apache Gluten (incubating) (Velox in Spark)

A Multi-Engine Data Stack?

- Why be locked into one full-stack execution engine (like Spark)?
- Cost/performance/latency varies greatly across workload shapes and sizes
- You can do a lot with DuckDB, but actual big data does still exist

← GO BACK TO BLOG

BIG DATA IS DEAD

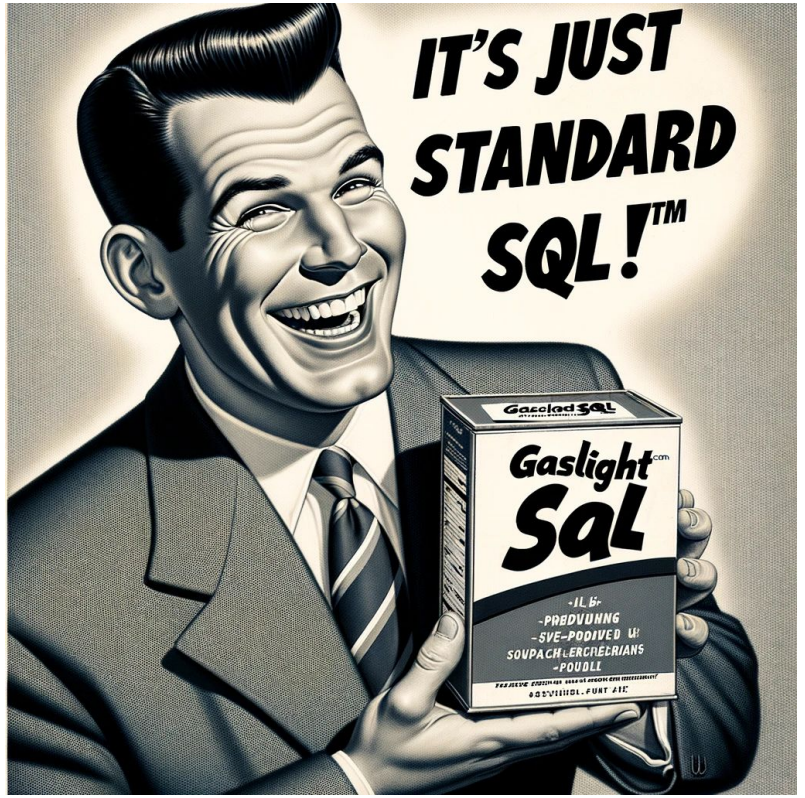
2023/02/07

BY JORDAN TIGANI

Barriers to a multi-engine data stack

- SQL dialects are non-portable and feature a wide spectrum of supported features
  - sqlglot is helping with this!

- Deciding which engine-to-use is non-trivial
- Diverse orchestration and infrastructure requirements



Magpie: Python at Speed and Scale using Cloud Backends

Alekh Jindal
Gray Systems Lab, Microsoft
alekh.jindal@microsoft.com

K. Venkatesh Emani
Gray Systems Lab, Microsoft
k.emani@microsoft.com

Maureen Daum*
University of Washington
mdaum@cs.washington.edu

Olga Poppe
Gray Systems Lab, Microsoft
olga.poppe@microsoft.com

Brandon Haynes
Gray Systems Lab, Microsoft
brandon.haynes@microsoft.com

Anna Pavlenko
Gray Systems Lab, Microsoft
annapa@microsoft.com

Ayushi Gupta*
Apple
ayushi.iiit@gmail.com

Karthik Ramachandra
Microsoft Azure Data
karam@microsoft.com

Carlo Curino
Gray Systems Lab, Microsoft
carlo.curino@microsoft.com

Andreas Mueller
Gray Systems Lab, Microsoft
andreas.mueller@microsoft.com

Wentao Wu
Microsoft Research
wentao.wu@microsoft.com

Hiren Patel
Microsoft
hirenp@microsoft.com

CIDR 2021

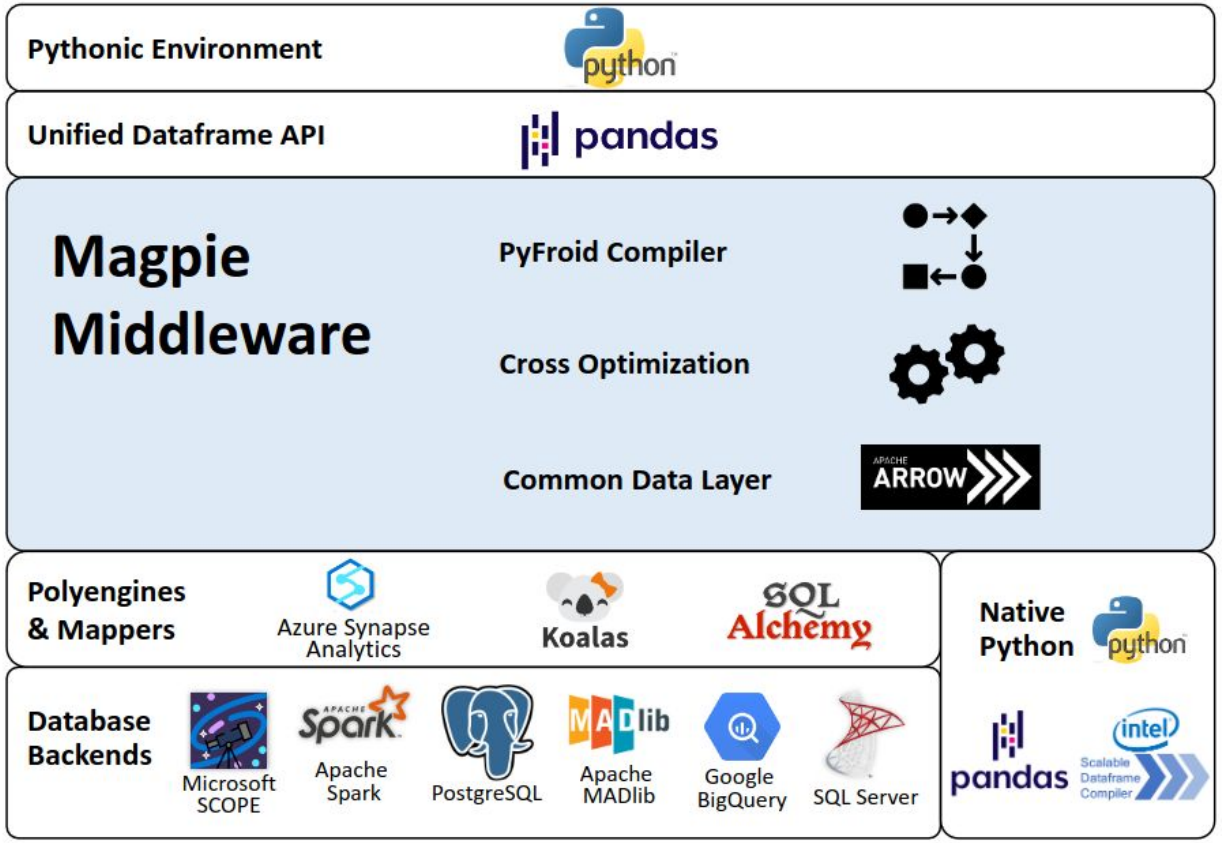
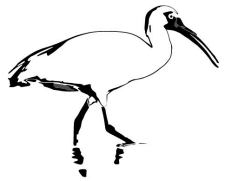


Figure 3: Our vision for a more simplified, unified, and efficient data science stack.

Enter the Bird  **ibis-project.org**

- An effort to harmonize the best of modern SQL with Pythonic fluent data frames
- Fixing a long list of shortcomings in the pandas API
- Leverage the benefits of a modern PL when creating complex analytical SQL queries
- Bringing portability (> 20 backends supported) to provide a unified Python API for a multi-engine data stack

```
left = ents.filter(_.known_for_titles.length() > 0).limit(10_000)
right = left.view()
shared_titles = (
    left
    .join(right, left.nconst != right.nconst)
    .select(
        s.startswith("known_for_titles"),
        left_name="primary_name",
        right_name="primary_name_right",
    )
    .filter(_.known_for_titles.intersect(_.known_for_titles_right).length() > 0)
    .group_by(name="left_name")
    .agg(together_with=_.right_name.collect())
    .mutate(together_with=_.together_with.unique().sort())
)
shared_titles
```

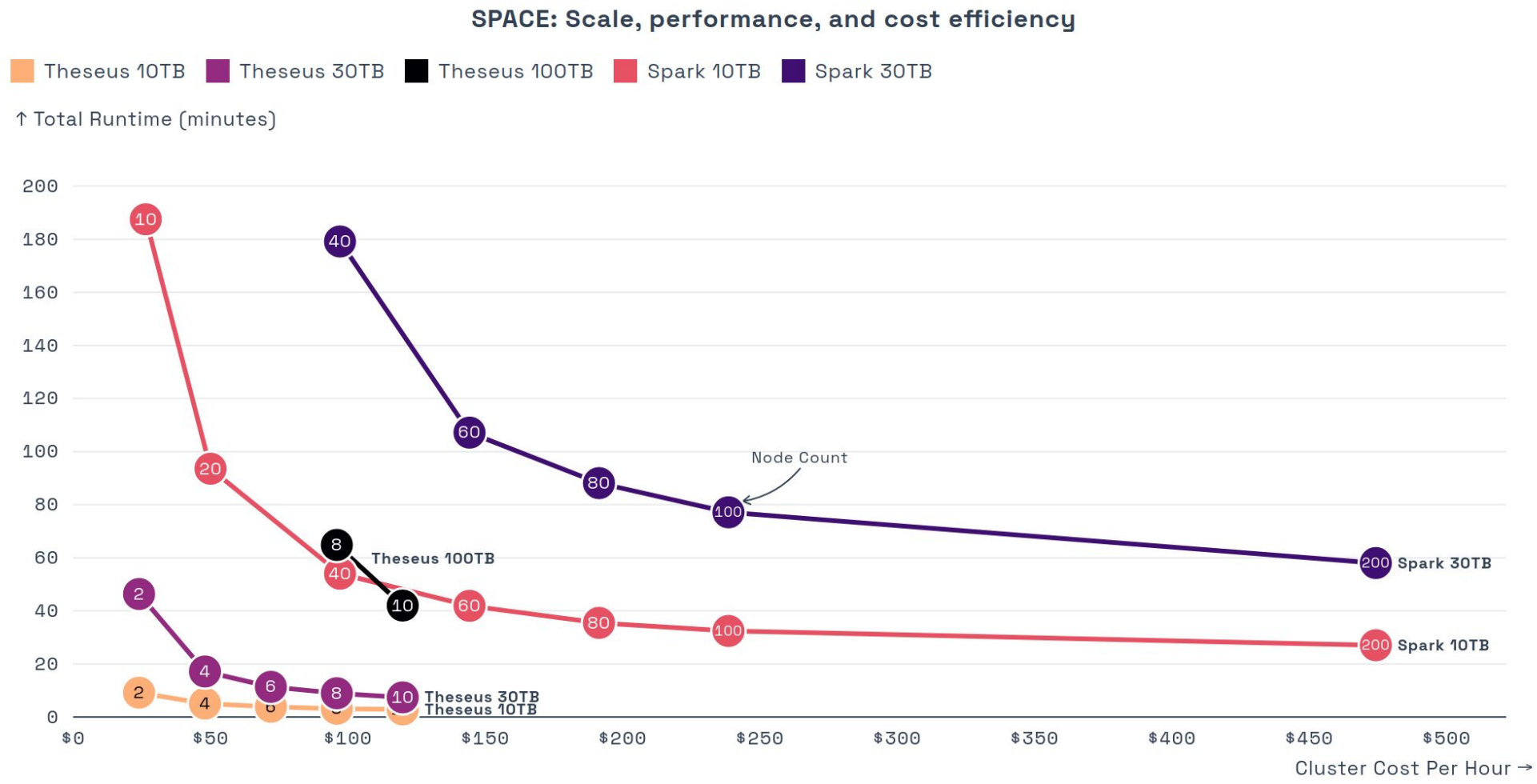
https://ibis-project.org/posts/bigquery-arrays/

Other Ibis Notes

- Created in 2015 at the same time as Arrow
- Strong focus on deep SQL support
- Helps automating the drudgery of tedious multi-step DDL workflows in databases

Why this matters

https://voltrondata.com/theseus



A Future Multi-Engine Stack

- An execution engine tailored for different data scales
  - <= 1TB: DuckDB and Friends
  - 1 - 10TB: Spark, Dask, Ray, etc.
  - > 10TB: Hardware-accelerated Processing (e.g. Theseus)
- A portable language front end (Ibis, Malloy, PRQL, or a standard transpilable SQL)
- Arrow-native API and wire transport

Closing Thoughts