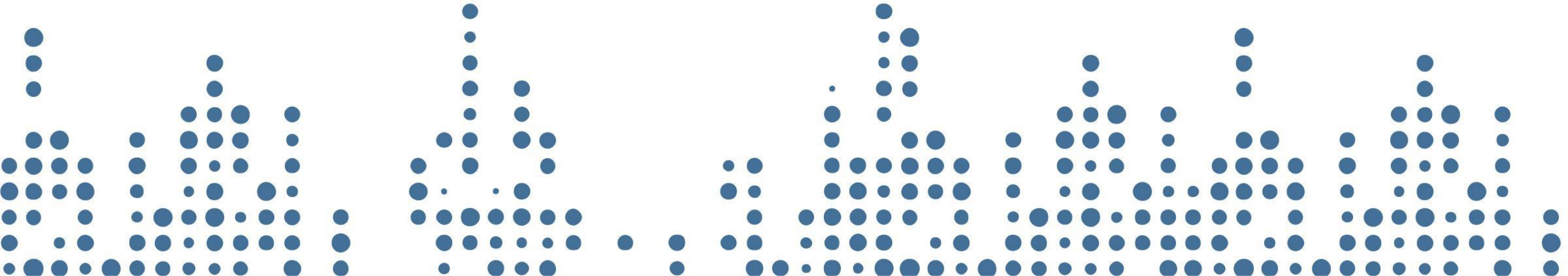
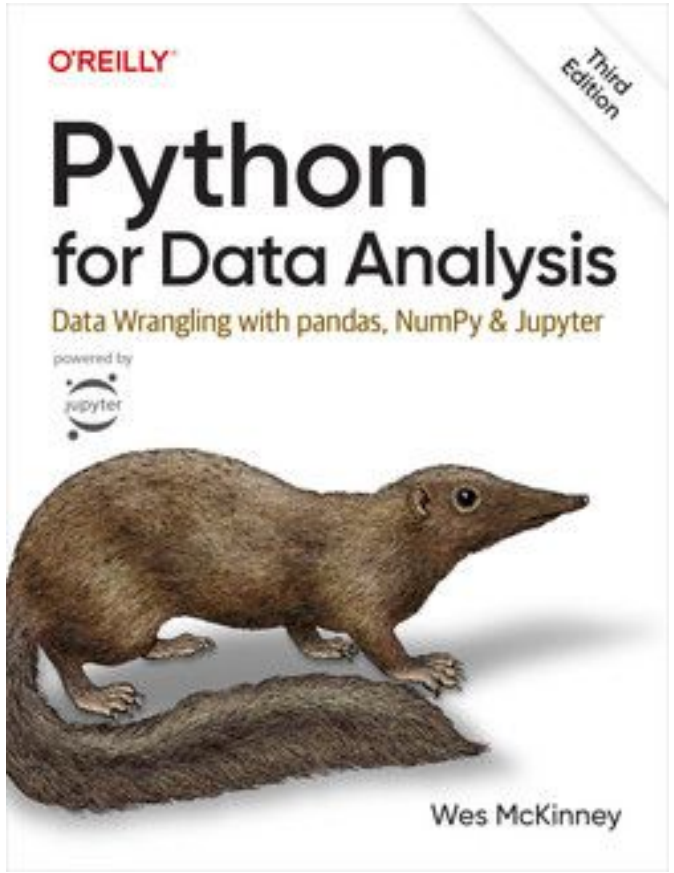
Brief Overview of My Path to Composable Data Systems

Wes McKinney
June 2024



Me



pandas

APACHE ARROW

Parquet

Ibis

Me

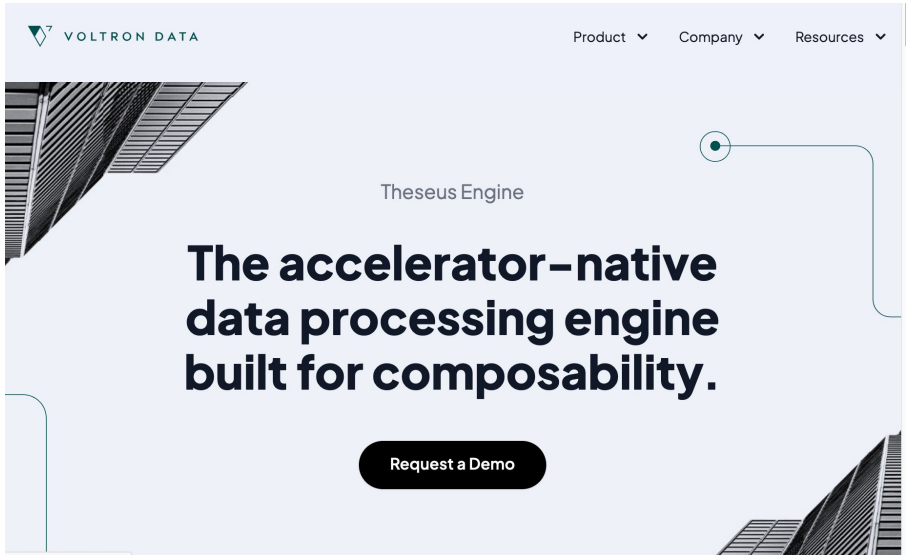
Principal Architect

Co-founder, Advisor

General Partner

Voltron Data (2021 —)

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



Posit PBC

- Founded 2009, originally as RStudio
- ~300 person, remote-first company
- Open source software for code-first 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

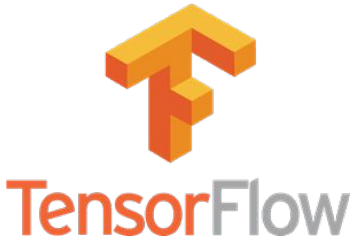
Brief History of pandas

- Development started in 2008 at AQR, a quant hedge fund
- Open sourced late 2009
- ***Python for Data Analysis, 1e*** published in August 2012
- Became a community-owned open source project in 2013

pandas in the Python data stack

- Essential data access (CSV, SQL, etc.) and in-memory data structures
- Data cleaning and manipulation
- Interactive / exploratory analysis
- First step in larger workflows before scikit-learn or other modeling libraries

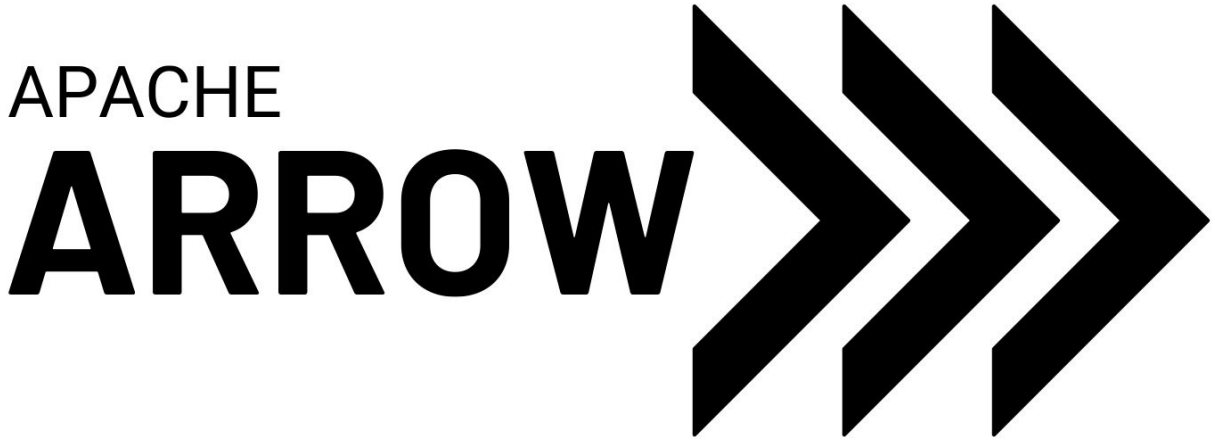
How did Python become so popular?



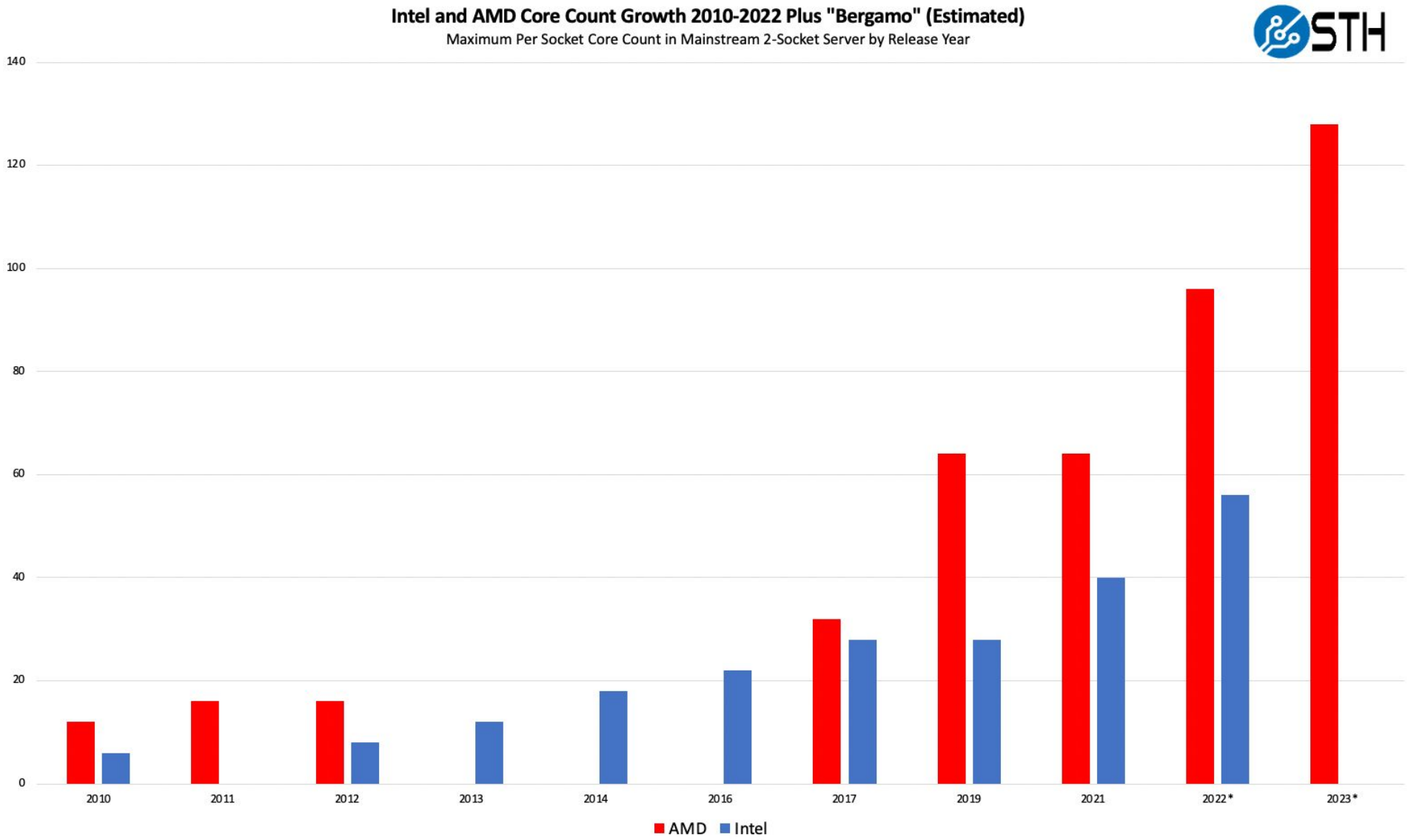
Python and Big Data

- Initial wave of Big Data systems were Java/JVM centric
- Python support was largely an afterthought
- Circa 2014: difficult for R/Python to access data originating in the Hadoop ecosystem

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



How has computing hardware changed over the last 20 years?



https://www.servethehome.com/amd-and-intel-2p-server-core-count-growth-2010-2022/

AMD EPYC™ 9654

Regional Availability: Global, China, NA, EMEA, APJ, LATAM

Platform: Server

Product Family: AMD EPYC™

Product Line: AMD EPYC™ 9004 Series

# of CPU Cores: 96

# of Threads: 192

Max. Boost Clock: Up to 3.7GHz

All Core Boost Speed: 3.55GHz

Base Clock: 2.4GHz

L3 Cache: 384MB

1kU Pricing: 11,805 USD

Default TDP: 360W

AMD Configurable TDP (cTDP): 320-400W

CPU Socket: SP5

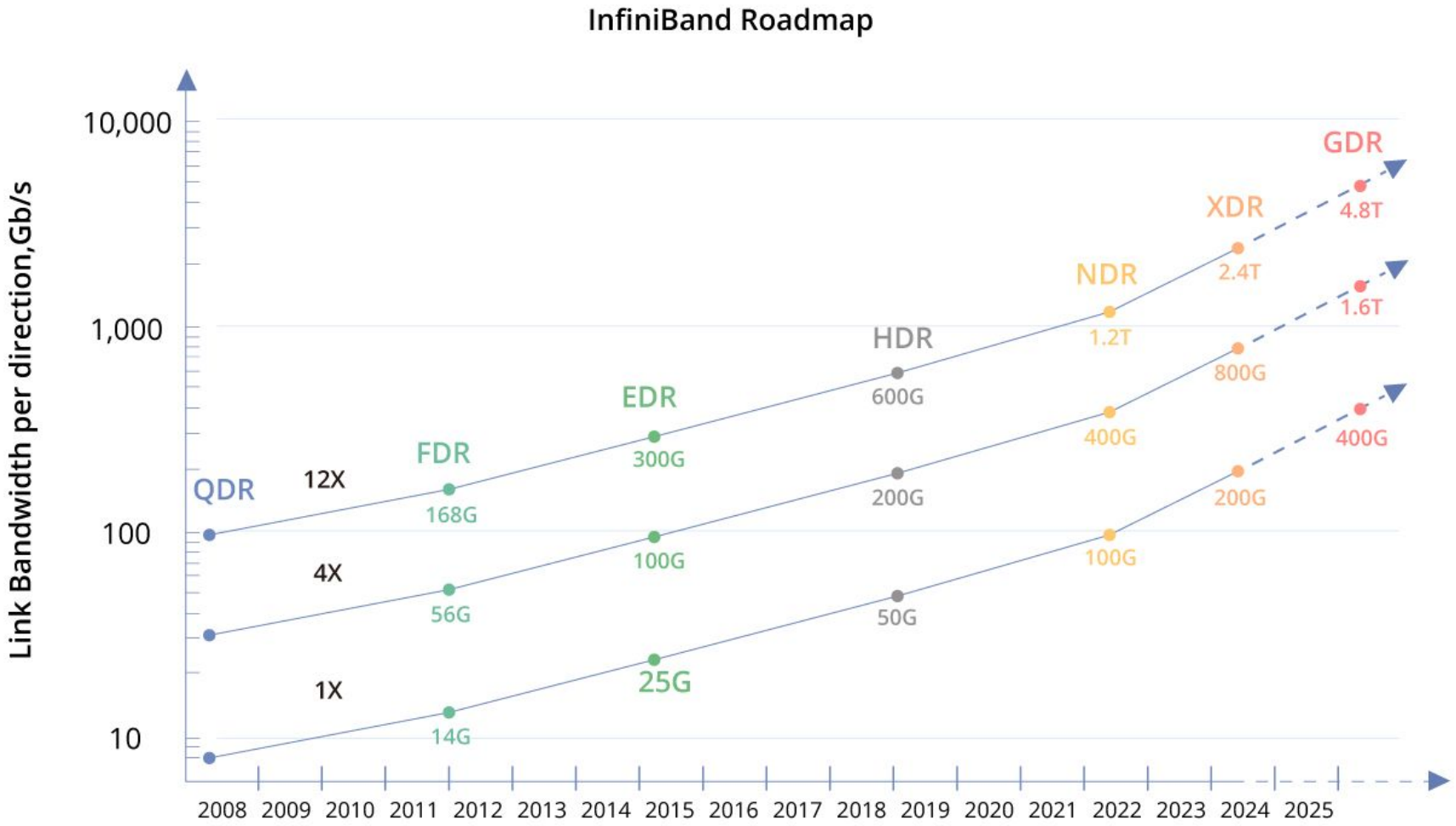
Socket Count: 1P / 2P

Launch Date: 11/10/2022

Trends in Disk Performance

Speed Statistic	HDD (Hard Disk Drive)	SSD (Solid State Drive)	NVMe m.2 (Nonvolatile Memory Express)
Read Speed	80 MB/s	200MB/s	5000 to 7300 MB/s
Write Speed	160 MB/s	550 MB/s	5000 to 6350 MB/s
Capacity Available	From 250GB to 14TB	250GB to 4TB	500GB to 4TB
Seeks	**~10ms**	**~0.1ms**	**~0.1ms**

Trends in Networking Performance



Python

```python
import pandas as pd

df = pd.read_csv('flights.csv')

df2 = pd.read_sql("select * from db.flights", con)

result = (
    df2.mutate(dep_minutes=df2['dep_delay'] / 60)
    .groupby('orig_code')
    ['dep_delay'].mean()
)
```

R

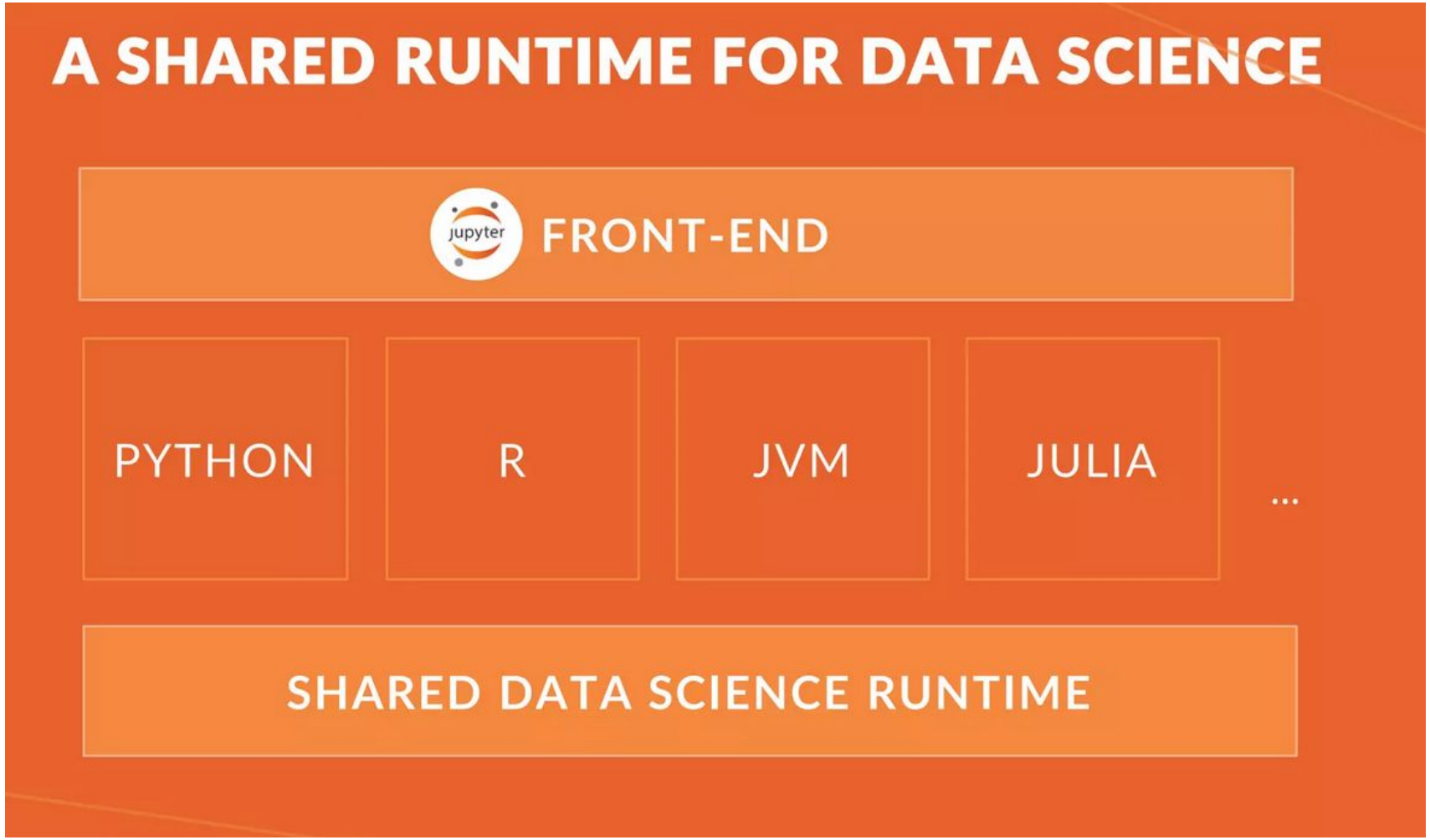
```r
flights <- read_csv('flights.csv')

flights %>%
    mutate(dep_minutes=dep_delay / 60) %>%
    group_by("orig_code") %>%
    aggregate(avg_delay=mean(dep_delay))
```

SQL

```sql
SELECT orig_code, avg(dep_delay) as avg_delay
FROM flights
GROUP BY 1
```

Mid-2010s Vision



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

Apache Arrow Development Group

Sponsors

TWO SIGMA

NVIDIA

Bloomberg

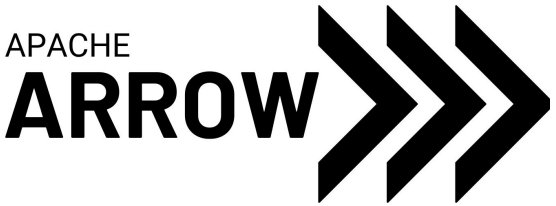
RESEARCH

intel

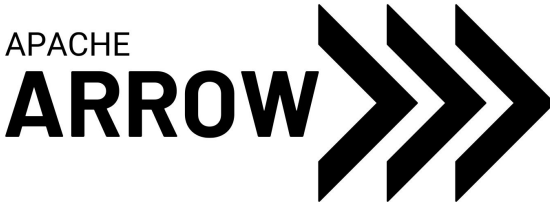
posit

Why now?

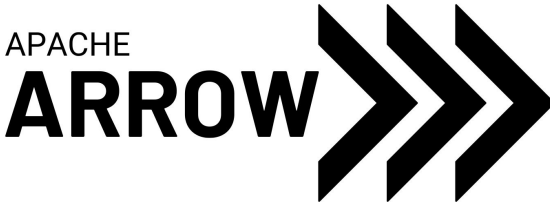
- 1st-Gen Big Data / Hadoop Era: 2006 - 2013
  - MapReduce popularizes disaggregated storage + compute
- 2nd-Gen 2013 - 2021
  - Vendors shift from proprietary software to delivery of services
  - Open source standards emerge and are popularized
  - Rapid progress in storage, networking, computing performance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


APACHE
ARROW

Data Lake Access



Data Lake Access



Language Interoperability

Data Lake Access

Parquet

Apache orc

APACHE ARROW

Fast Embeddable Execution

DuckDB

Velox

APACHE DATAFUSION

Language Interoperability

R

Java

GO

Data Lake Access

Parquet

Apache orc

Accelerated Database Connectivity

ADBC

snowflake

FlightSQL

Google BigQuery

APACHE ARROW

Fast Embeddable Execution

DuckDB

Velox

APACHE DATAFUSION

Language Interoperability

R

Java

GO

Layers of the Composable Cake

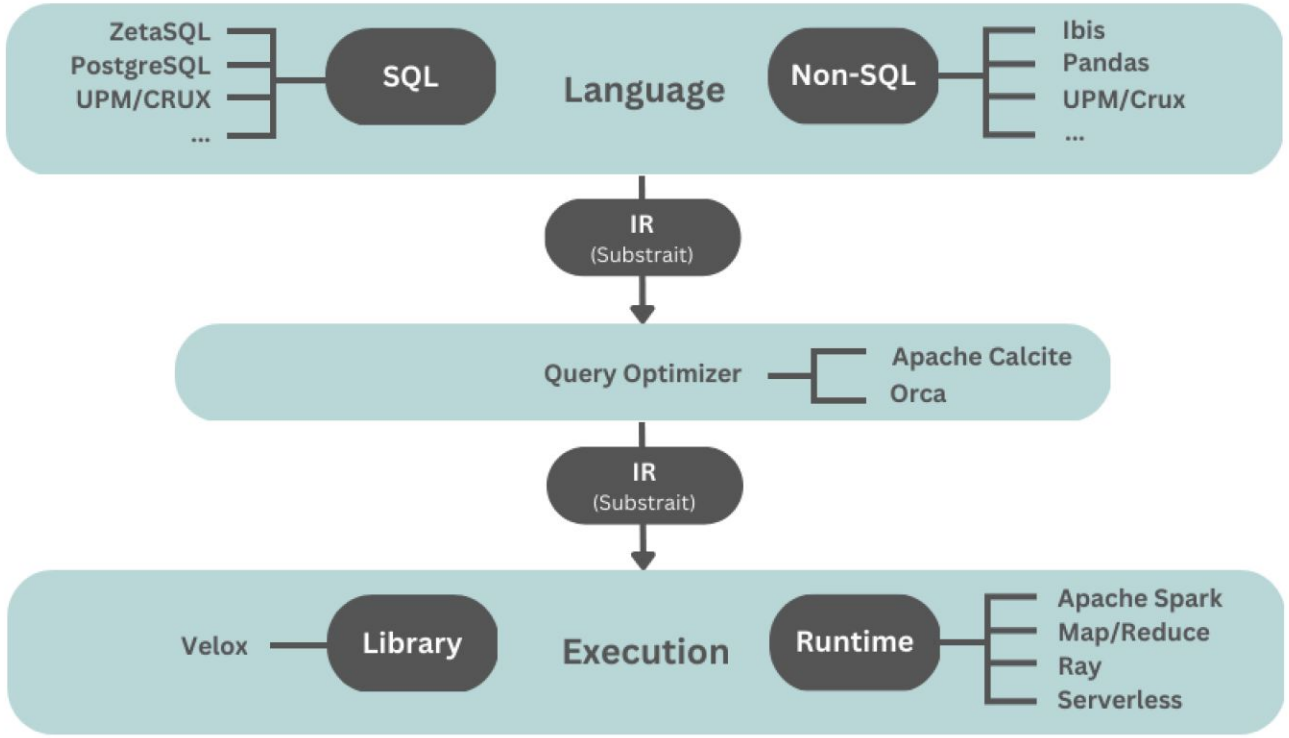


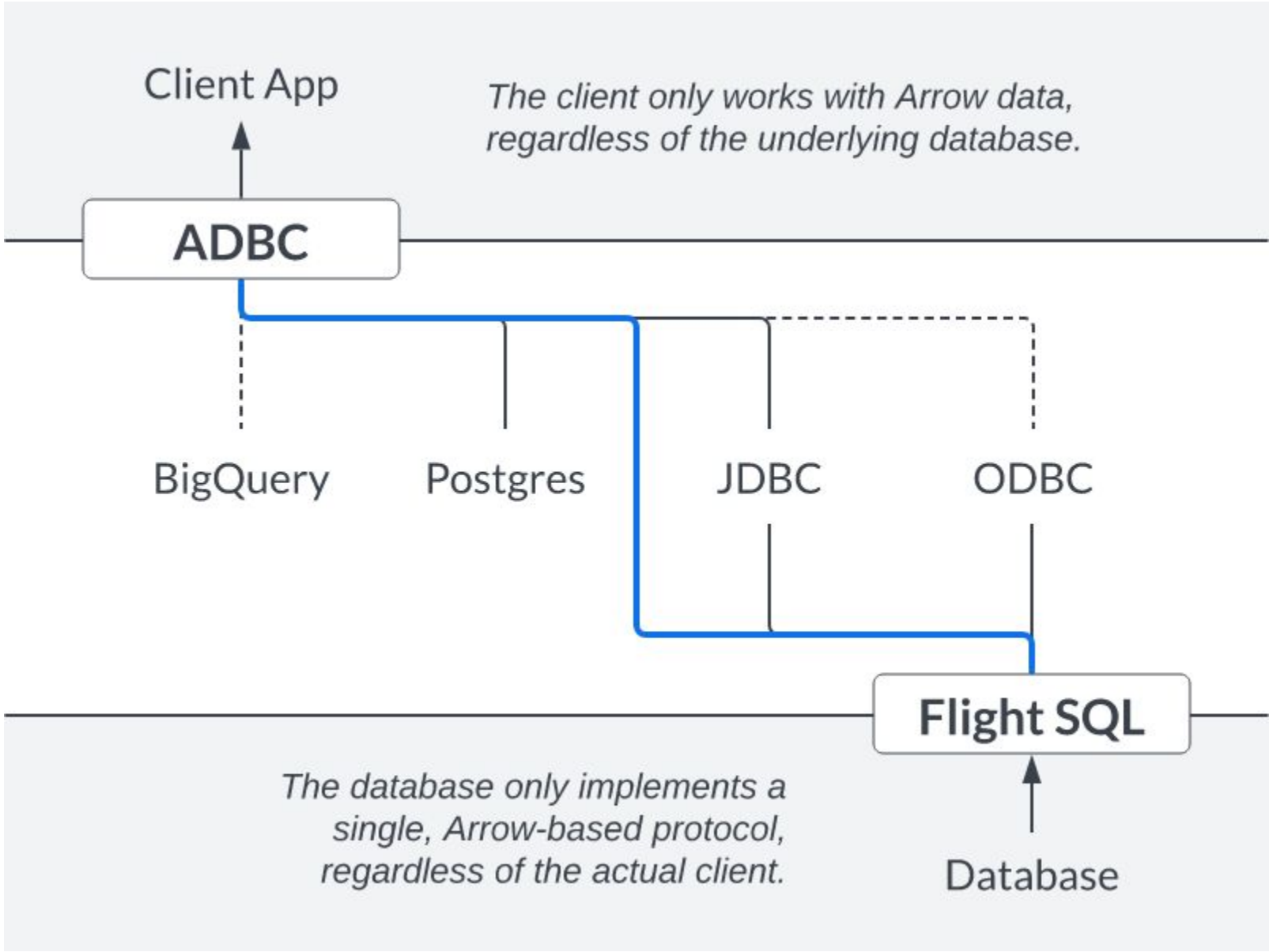
Figure 1: Open source modular data stack outline.

Modular Execution Engines

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

Modular Acceleration Projects

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


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

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

A Multi-Engine Data Stack?

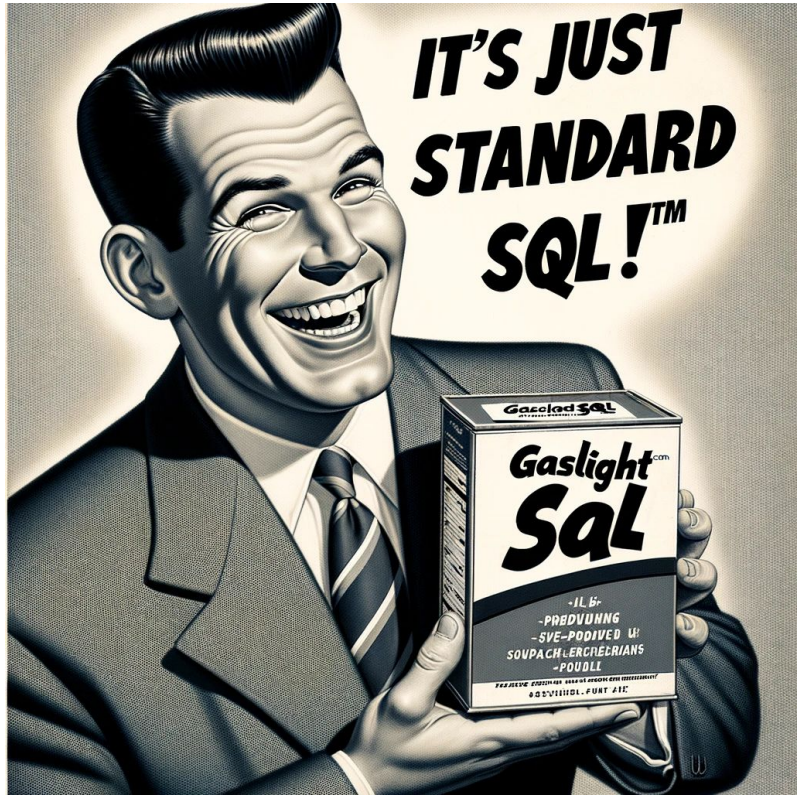
- Why be locked into one full-stack execution engine (like Spark)?
- Cost/performance/latency varies greatly across workload shapes and sizes
- You can do a lot with DuckDB, but actual big data does still exist

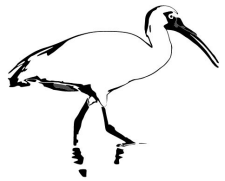
HT to Julien Hurault ! https://juhache.substack.com/p/multi-engine-data-stack-v0

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



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/

A Future Multi-Engine Stack

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