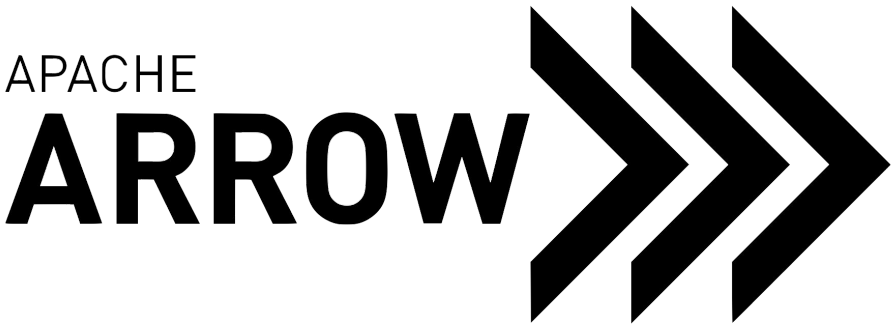
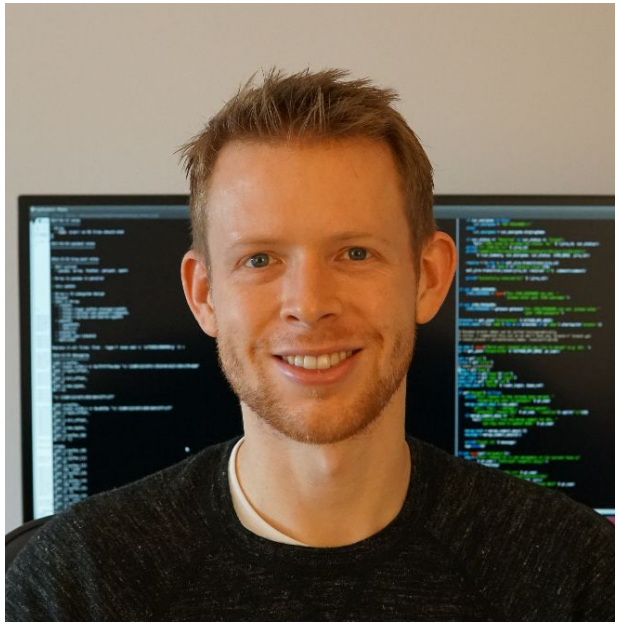
“A sharper Arrow” Accelerating Columnar Analytics with Apache Arrow

2020-10-21 Global STAC

Wes McKinney @wesmckinn



Me



- Director of **Ursa Labs**, open source dev group working on **Apache Arrow**
- Created Python **pandas** project
- PMC Apache {Arrow, Parquet}, ASF Member
- Wrote ***Python for Data Analysis***
- Formerly: Two Sigma, Cloudera, DataPad, AQR

Many columnar data tools have significant inefficiencies

- High % of compute spent on **serialization** (converting between data formats)
- **Inefficient in-memory computing** that **fails to fully utilize modern hardware capabilities**
- Much developer time spent building **data connectors** and maintaining **glue code.**

The Tabular Data “Tower of Babel”

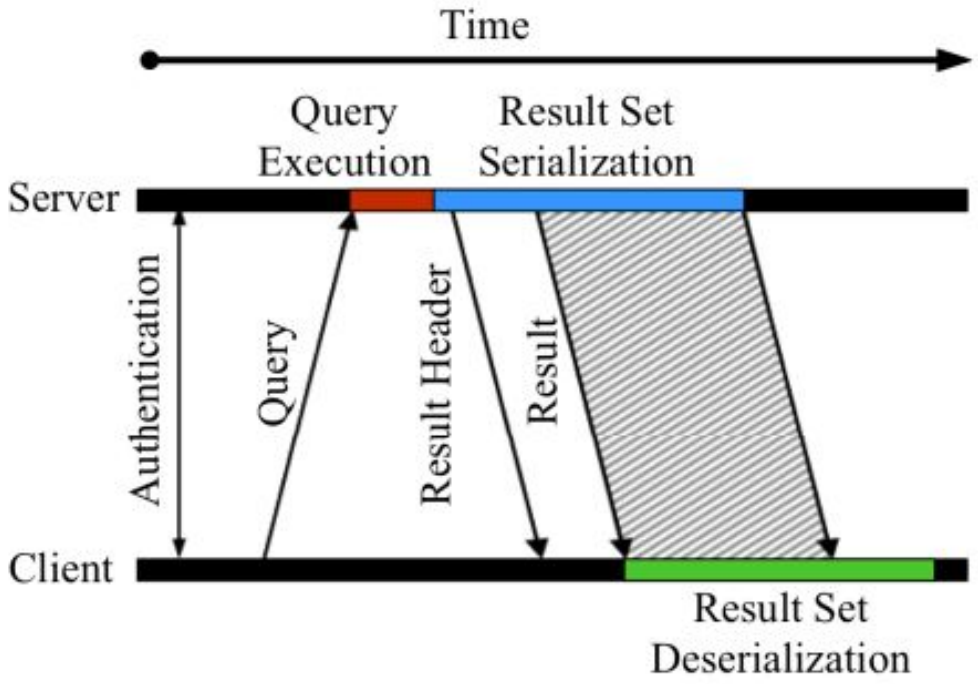
- Projects and language ecosystems utilize a multitude of in-memory data formats and file storage formats
- Analytics, Data Eng, ML / AI workloads often dominated by serialization overhead (can be 80-90+%)
- Complex and inefficient data access by data science tools in Python and R, etc.

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

VLDB '17

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

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



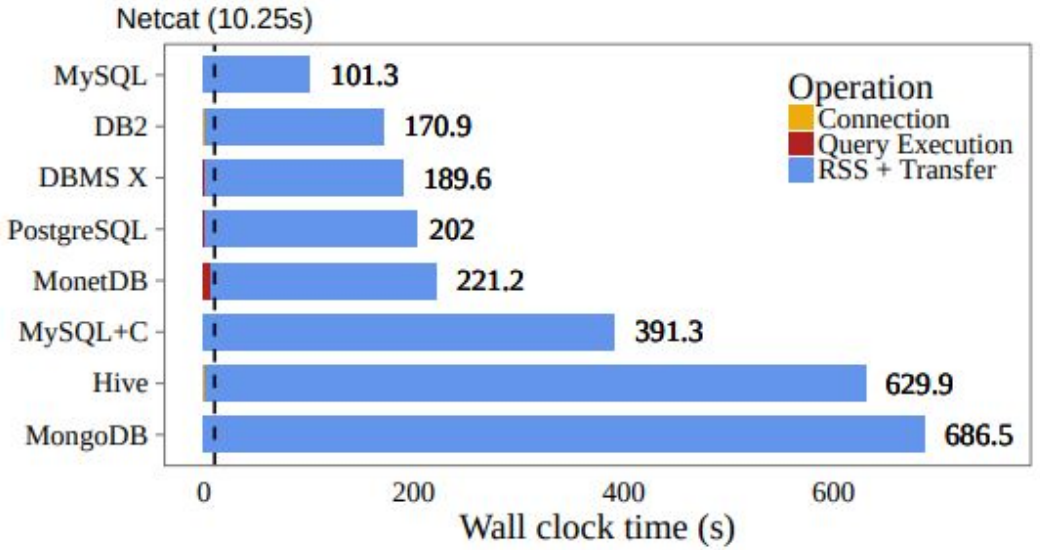


Figure 1: Wall clock time for retrieving the `lineitem` table (SF10) over a loopback connection. The dashed line is the wall clock time for netcat to transfer a CSV of the data.

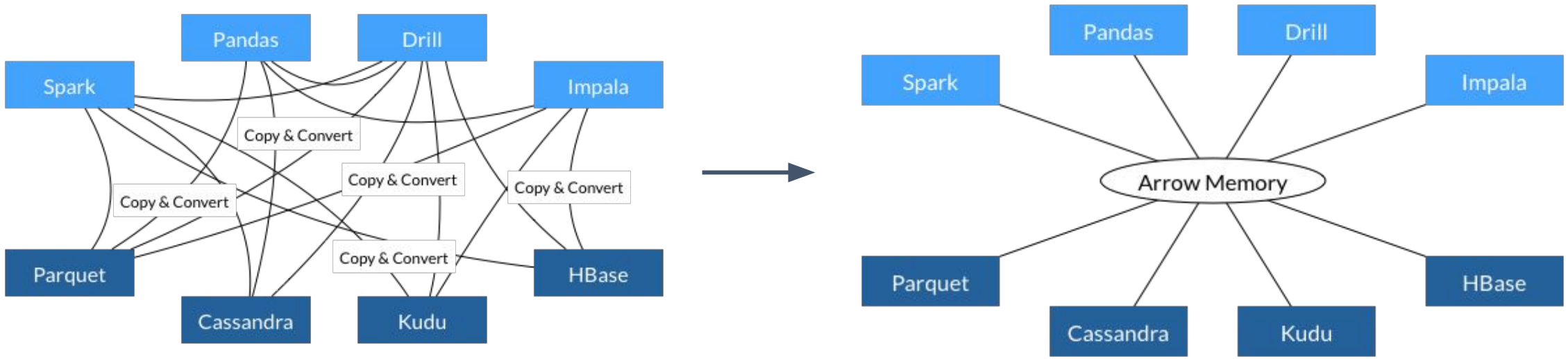
Objectives for an “interoperability utopia”

- Minimize / eliminate serde costs
- Reduce dev time spent on custom “glue code”
- Fast off-the-shelf client libraries for every language
- Transport efficiently from point to point without using intermediate file storage
- Avoid “coordinator bottlenecks”

APACHE
ARROW

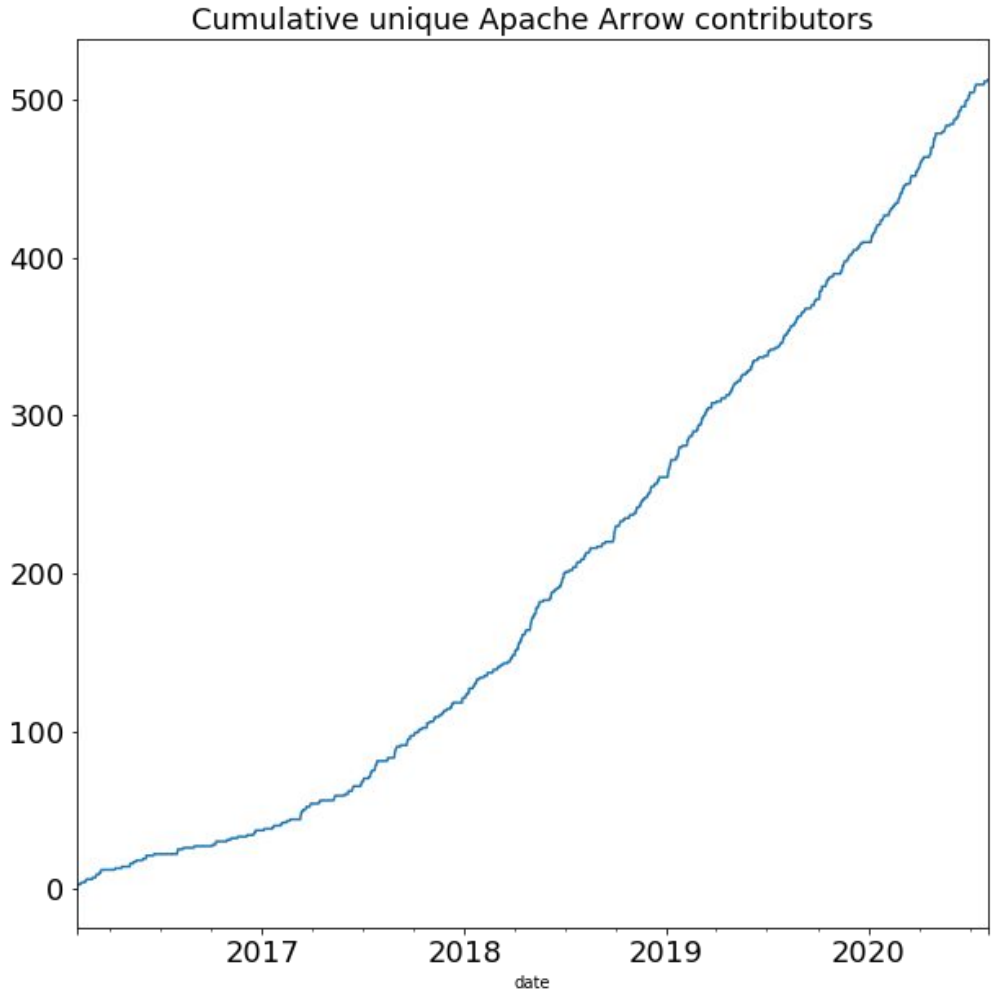
- ASF open source project started February 2016
- Provides
  - **Language-agnostic, standardized columnar data format** for efficient, parallel CPU/GPU-based data-frame-like processing
  - **Messaging protocol** that minimizes or eliminates **data serialization costs** at system or process boundaries
  - **Flight**, a state-of-the-art framework for building services that transport Arrow-based datasets over TCP
  - An in-development multi-language **computation platform**

Defragmenting Data

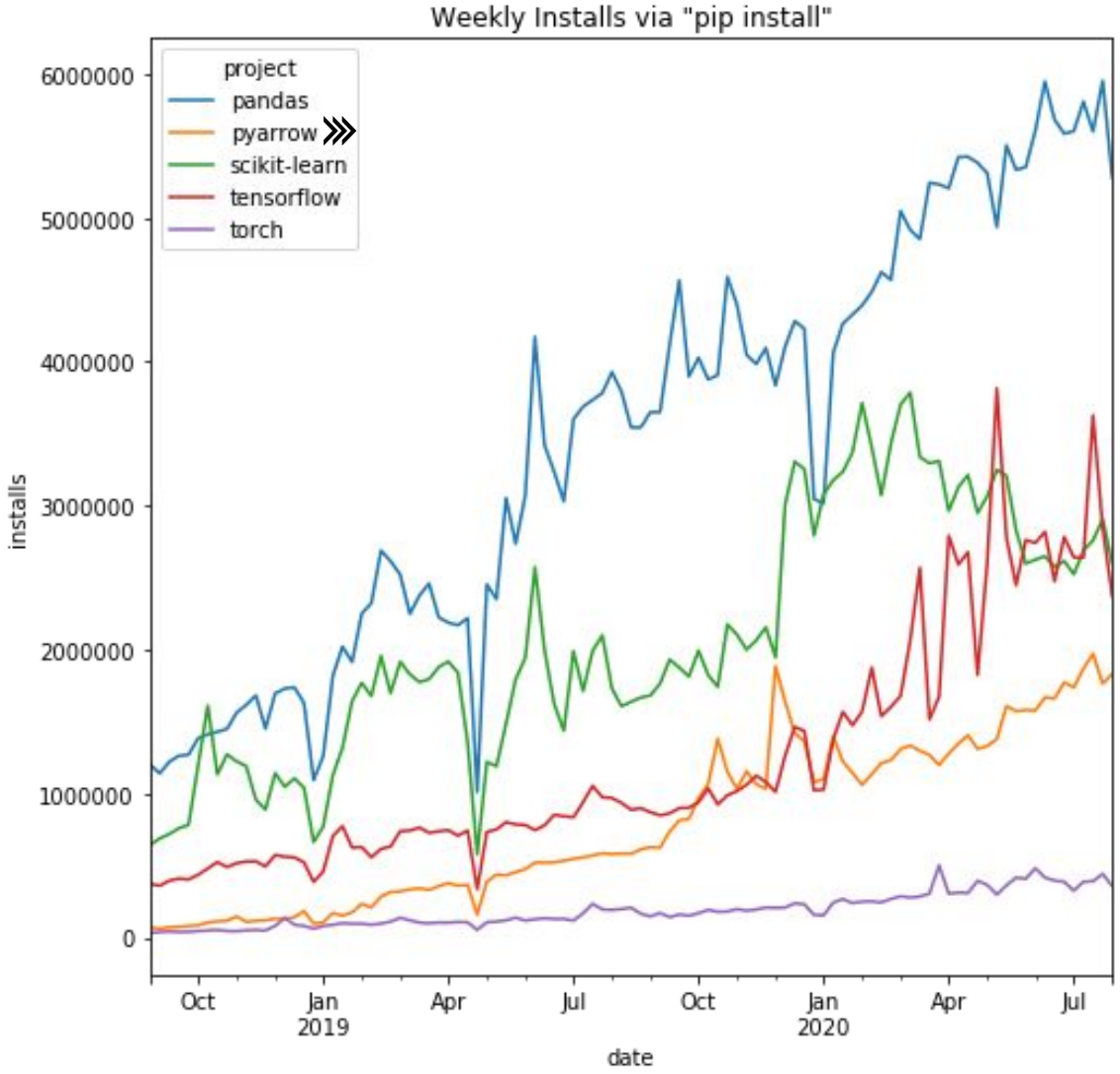


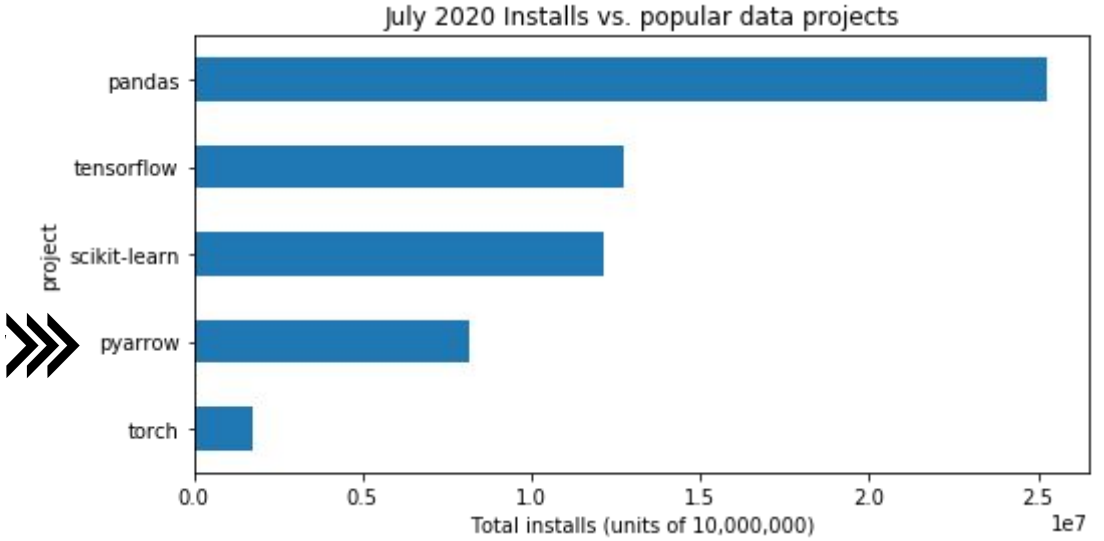
A Thriving Open Source Community

- Over 500 unique contributors since inception in February 2016
- **1.0.0 Release** July 2020 first “format stable” release
- 11 programming languages represented
- In use by: MSFT, GOOG, AMZN, NVDA, INTC, BABA, IBM

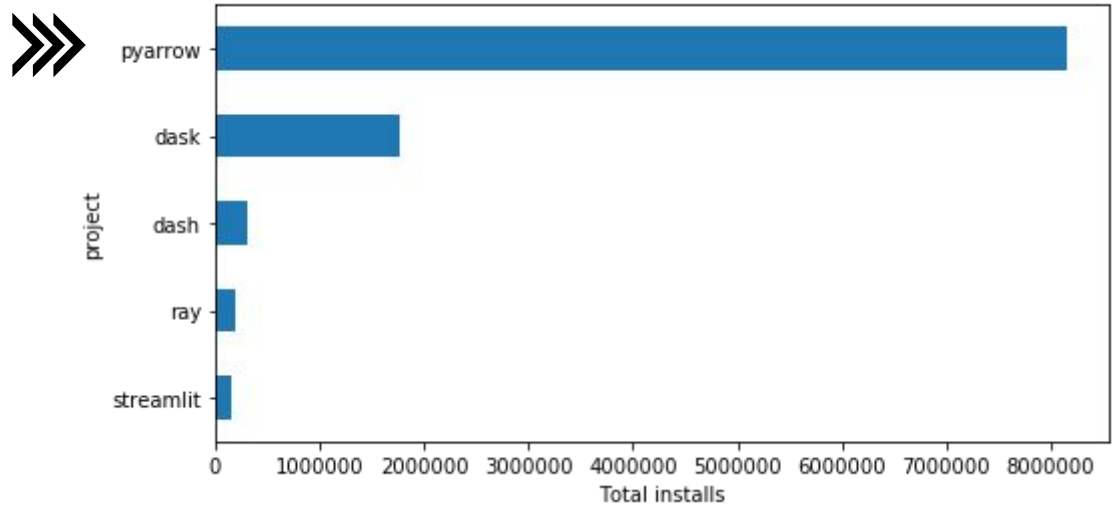


Python install metrics





vs some other rising Python projects



Some Example Arrow Uses

Native In-Memory Format for Columnar SQL Execution + Arrow Flight for fast data access

Accelerating Client Protocol Throughput

Accelerating Client Protocol Throughput and Python Parquet support

Parquet & Feather File Import + Extension Types

Accelerated User-Defined Functions for Python and R

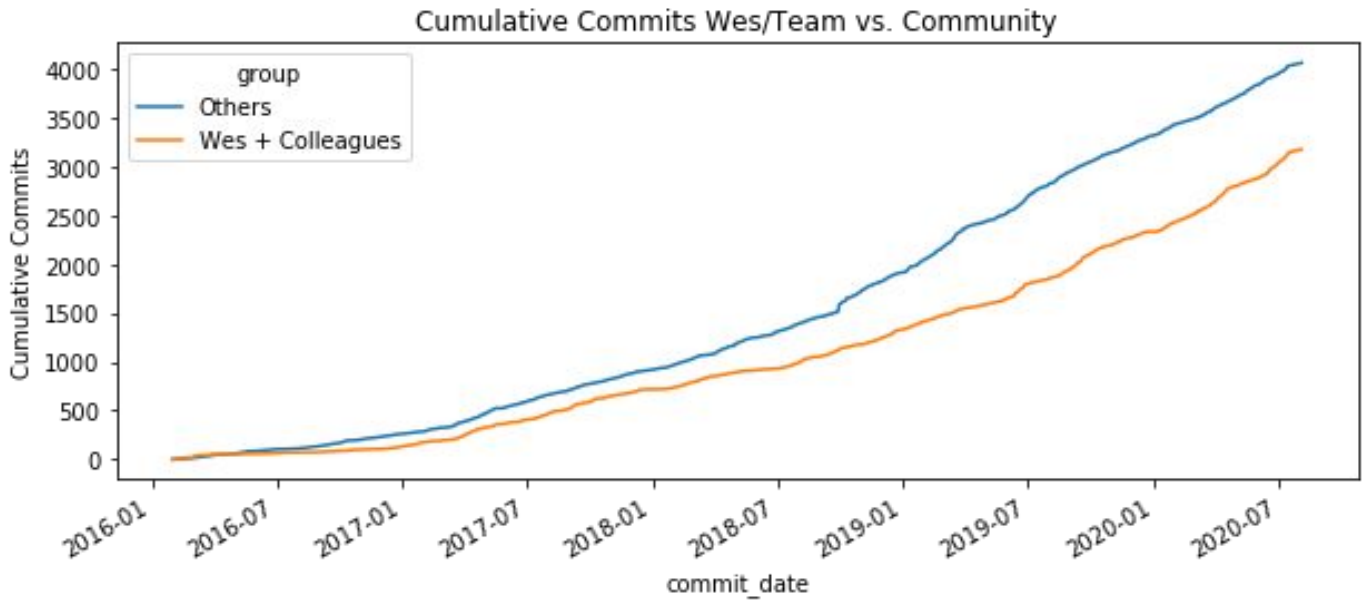
Arrow standard input format for TensorFlow Datasets

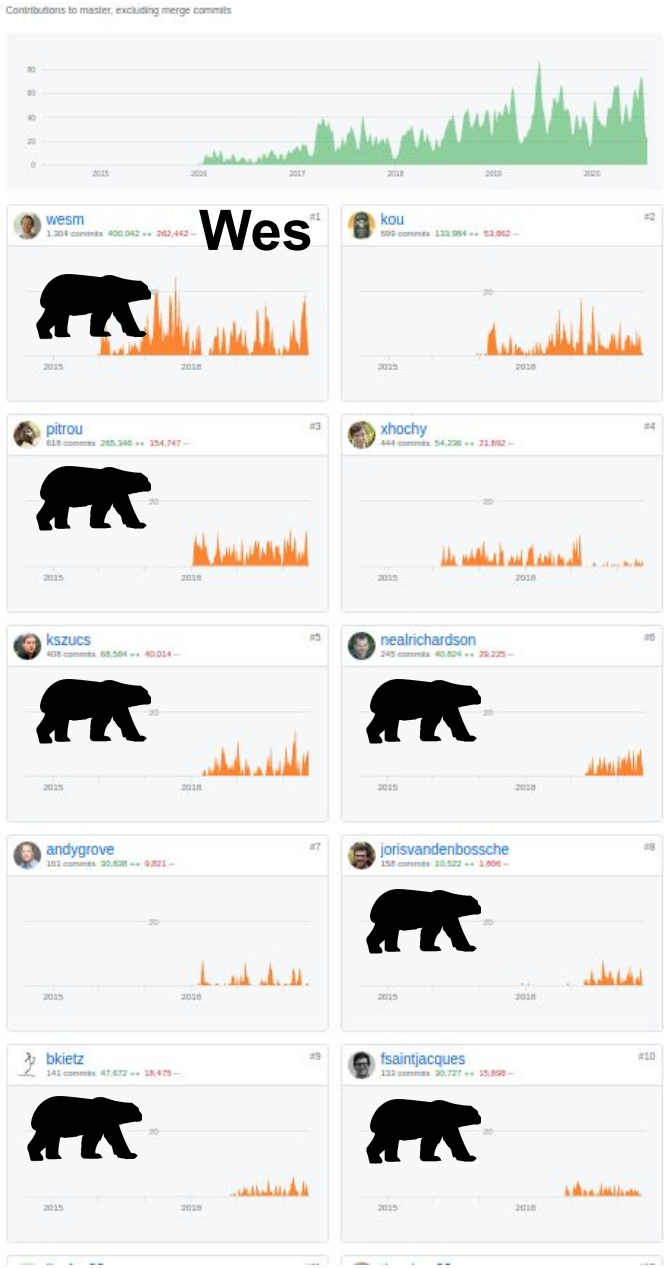
Data format for CUDA-based GPU Data Frame processing

Data Streaming for Low-latency Real Time Pivot Tables (from JP Morgan)

Ursa Labs has been the project's biggest driving force

- As a team, we are the largest single source of contributions in the open source project
- We have been responsible for a large fraction the project's operational tooling (CI, packaging, release management, developer tools)





Top 10 GitHub Contributors

Apache Arrow 1.0.0

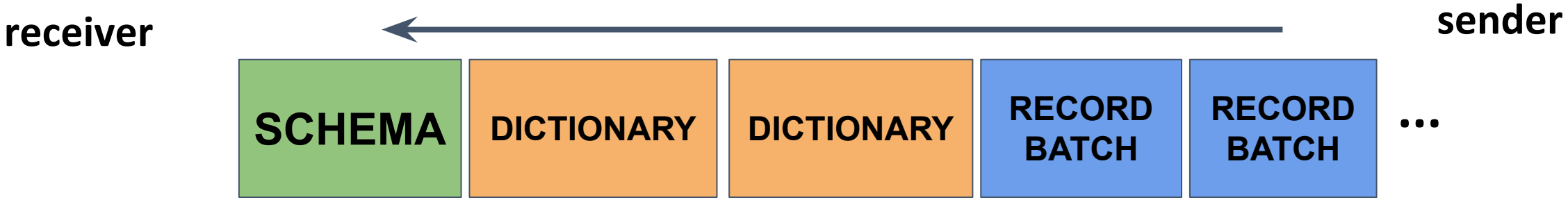
- Released July 2020
- “Formally” stable protocol, closed to breaking
- Backward and forward compatibility guarantees for the binary protocol
- Move to Semantic Versioning

Arrow’s Columnar Memory Format

- Runtime memory format for analytical query processing
  - Companion to serialization tech like Apache {Parquet, ORC}
- “Fully shredded” columnar, supports flat and nested schemas
- Organized for cache-efficient access on CPUs/GPUs
- Optimized for data locality, SIMD, parallel processing
- Accommodates both random access and scan workloads

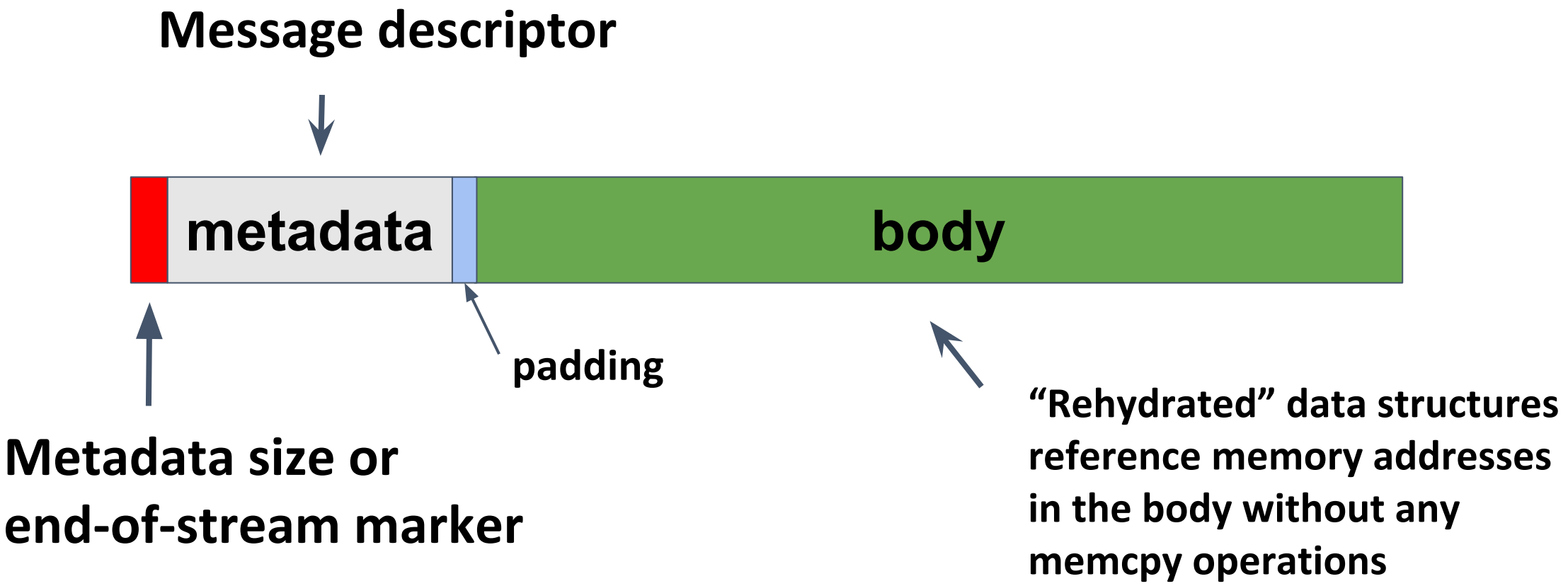
Arrow Binary Protocol

- **Record batch:** ordered collection of named arrays
- Streaming wire format for transferring record batch metadata and data between address spaces
- Often called “IPC protocol” since suitable for zero-copy shared memory interprocess communication (IPC)
- Sequence of “encapulated IPC messages”



Encapsulated protocol (“IPC”) messages

- Serialization wire format suitable for stream-based parsing



Flight - High Speed Network Transport for Arrow

- A gRPC-based framework for implementing clients and servers that send and receive Arrow columnar data natively
- Uses Protocol Buffers v3 for client protocol
- Pluggable command execution layer, authentication
- Low-level gRPC optimizations
  - Write Arrow memory directly onto outgoing gRPC buffer
  - Avoids typical copying or deserialization cycles

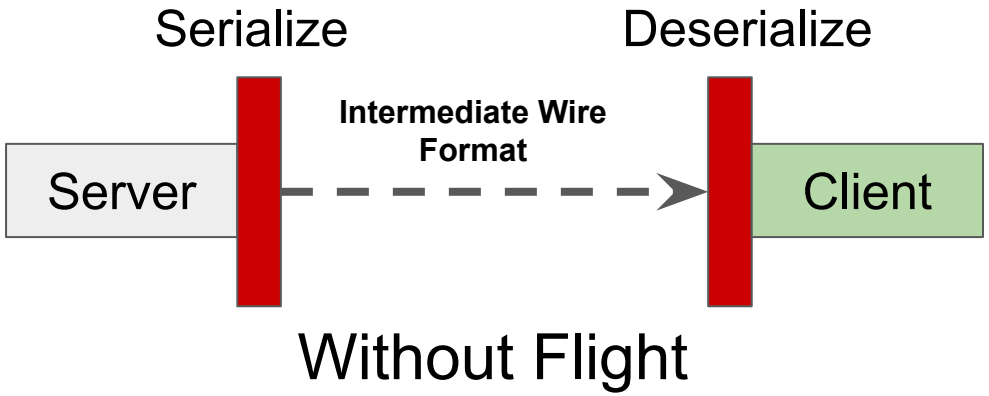
Key Early Developers

Some design requirements

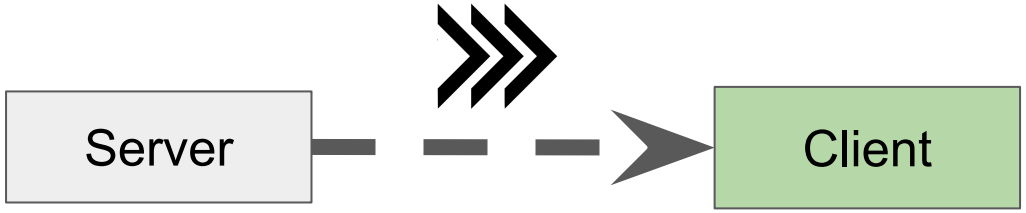
- Straightforward horizontal scalability and parallelization
- Can be used without knowledge of gRPC or Protocol Buffers
- “Dumb” clients with only the gRPC service definition can use without knowledge of Arrow columnar format

Advantages of moving to Arrow Flight

- Serialization-free transport over Arrow datasets over TCP
- Up to 10-100x faster end-to-end than common database network protocols like ODBC, JDBC
- Systems may still reap > 10x performance benefits by using Flight even if they are not internally “Arrow-native”



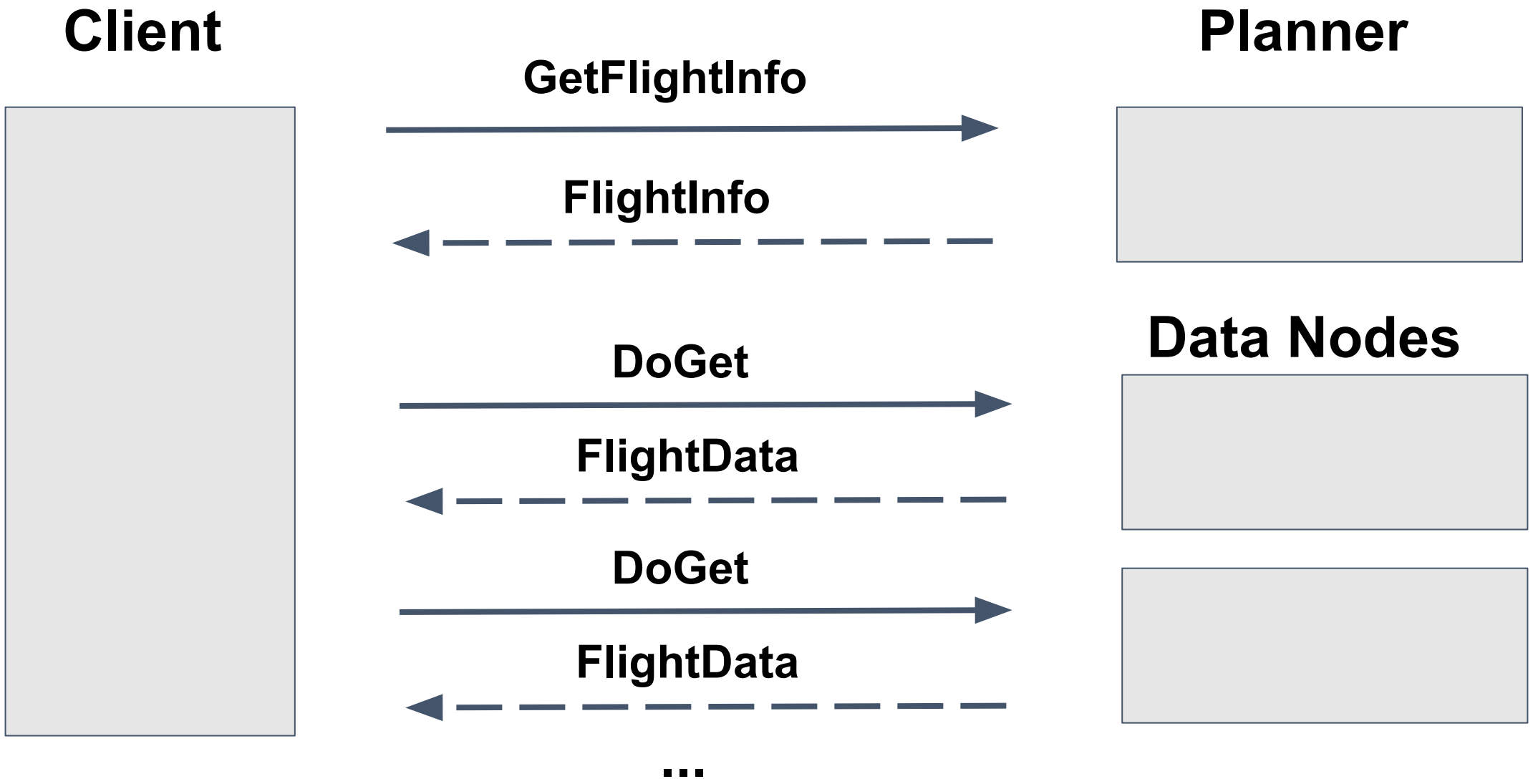
Without Flight



With Flight

NOT STAC BENCHMARKS

Arrow Flight - Parallel Get



Some Flight benchmarks

- 1 GBit ethernet (at home)
  - **112.24 MB/s** theoretical max
  - Achieves **92.17 MB/s**
- On localhost
  - **4 - 8 GB/s**

NOT STAC BENCHMARKS

Toward an Arrow-native World

- More reusable software components for data access and data processing
- Cross language boundaries without paying as high a price
- Fewer inefficiencies from data serialization

Getting involved

- Join dev@arrow.apache.org
- PRs to https://github.com/apache/arrow