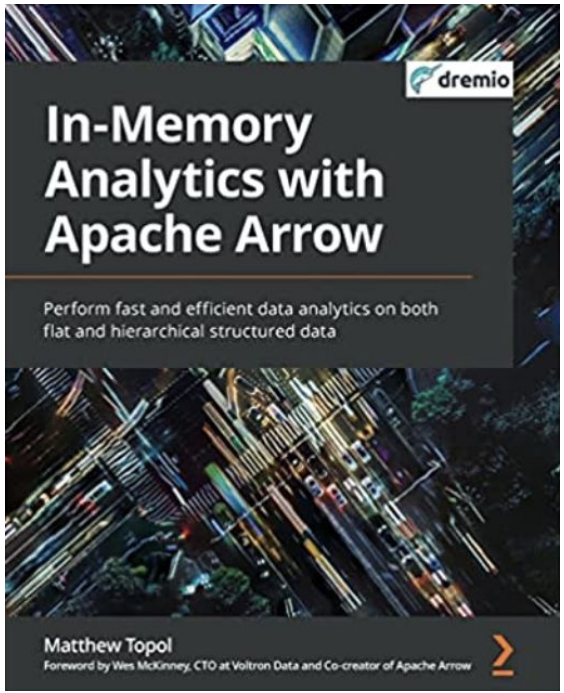
Apache Arrow Flight SQL

High performance, simplicity, and interoperability for data transfers

Jason Hughes

Director of Product Management @ Dremio

dremio


dremio
In-Memory Analytics with Apache Arrow
Perform fast and efficient data analytics on both flat and hierarchical structured data
Matthew Topol
Foreword by Wes McKinney, CTO at Voltron Data and Co-creator of Apache Arrow


APACHE
ARROW

Agenda

- What if we could have for our data transfers...
- What options do we have?
- What is Apache Arrow? Is it enough?
- What is Arrow Flight? Is it enough?
- What is Arrow Flight SQL? Is it enough?
- What are the Arrow Flight SQL ODBC & JDBC Drivers?

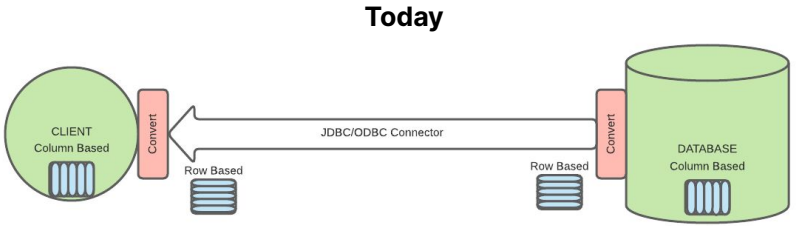
What if we could have for our data transfers...

- ✓ Standardized APIs
- ✓ No need for many different drivers
- ✓ High performance transfers
- ✓ Parallel data transfers (1:many, many:many)
- ✓ Easy to implement on server side, easy to use on client side

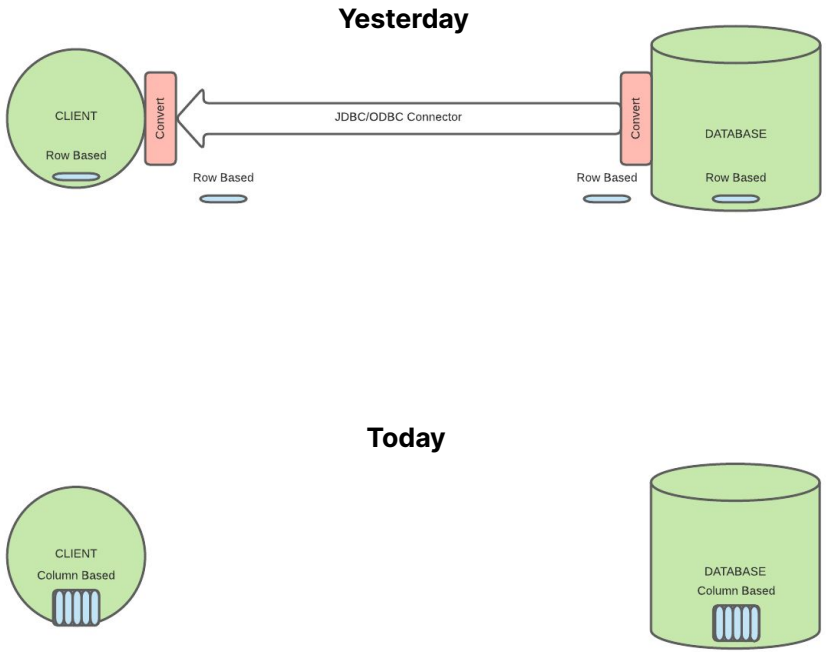
Why don’t ODBC & JDBC fit in today’s analytics world?

- Built in 1992 & 1997, respectively
  - Inherently row-based
  - Prevented the many:many of client-to-database-APIs
  - But resulted in one:many of API-to-database-protocols

- Today: heterogeneous big data workloads
- Most tools involved with OLAP workloads today are columnar
  - But there is no columnar transfer standard



Increasingly, columnar → columnar is the standard



What options do we have to keep things columnar?

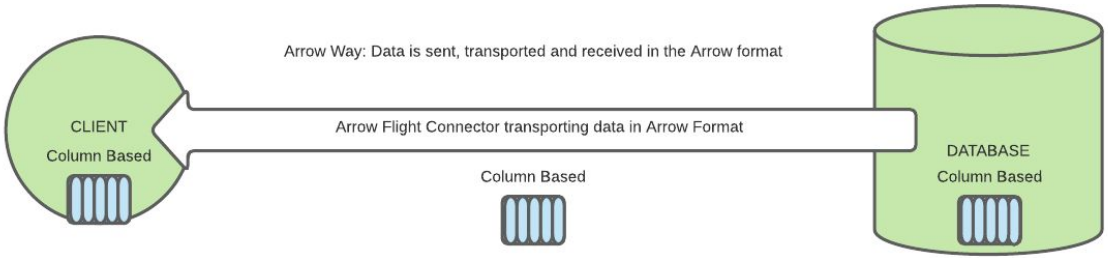
- Custom protocols
  - One per server
  - Implemented individually by each client

Pick your poison

	JDBC/ODBC	Custom protocol
Standardize client interface	✅	❌
Standardize client database access interface	✅	❌
Standardize server interface	❌	❌
Prevent unnecessary serialization	❌	✅

Enter: Apache Arrow Flight

What if?

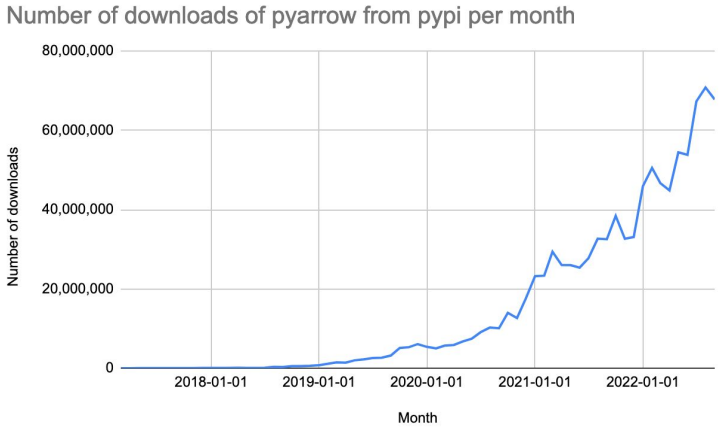


Enter: Arrow Flight

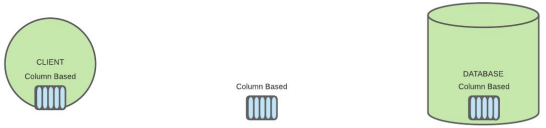
Before we go there: Apache Arrow

Introduction to Apache Arrow

- **An in-memory columnar format**
  - Includes libraries for working with the format
    - E.g., computation engine, IPC, serialization / deserialization from file formats.
  - Support for many languages (12 currently)
- **Co-created by Dremio and Wes McKinney**
- **Rapid adoption, popularity, and capability growth**



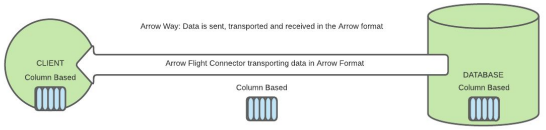
Isn’t Arrow enough?



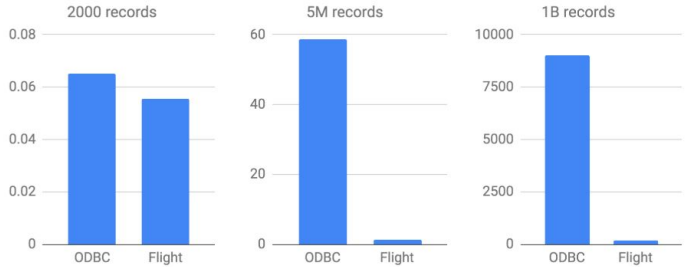
- Arrow is primarily geared towards operations on data in a single-node
  - No specification for cross-network transfers
    - Within a cluster
    - Client-server

Enter: Arrow Flight

What is Arrow Flight?

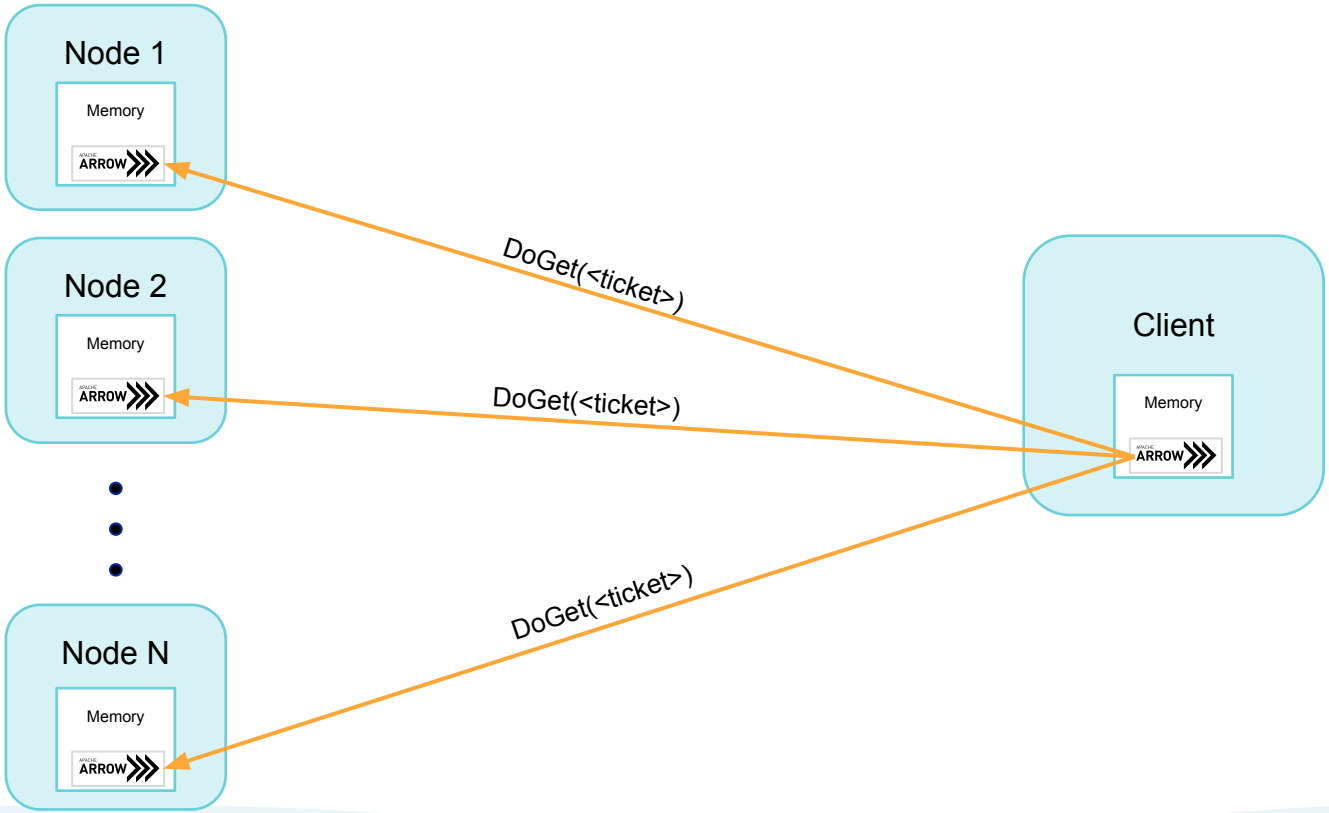


- A general-purpose client-server framework to **simplify high performance transport of large datasets over network interfaces**
- Designed for data in the modern world
  - **Column-oriented**, rather than row-oriented
  - Arrow's columnar format is designed for **large numbers of rows and high compressibility**
- Makes including the client-side in distributed computing easier
- Enables **parallel data transfers**



dremio

Parallel data transfers with Arrow Flight



Isn’t Arrow Flight Enough?

	JDBC/ODBC	Custom protocols	Arrow Flight
Standardize client interface	✅	❌	✅
Standardize client database access interface	✅	❌	❌
Standardize server interface	❌	❌	✅
Prevent unnecessary serialization	❌	✅	✅

Isn’t Arrow Flight Enough?

- Client sends an arbitrary byte stream, server sends a result
  - The byte stream only has meaning for a particular server that’s chosen to support it
  - Example – Dremio interprets the byte stream to be a UTF-8 encoded SQL query string
- Flight is meant to serve any tabular data, not databases in particular
  - E.g., catalog information is not part of Arrow Flight’s design
- Haven’t we seen these problems before?
  - ODBC and JDBC standardize database activities (e.g., query execution, catalog access)

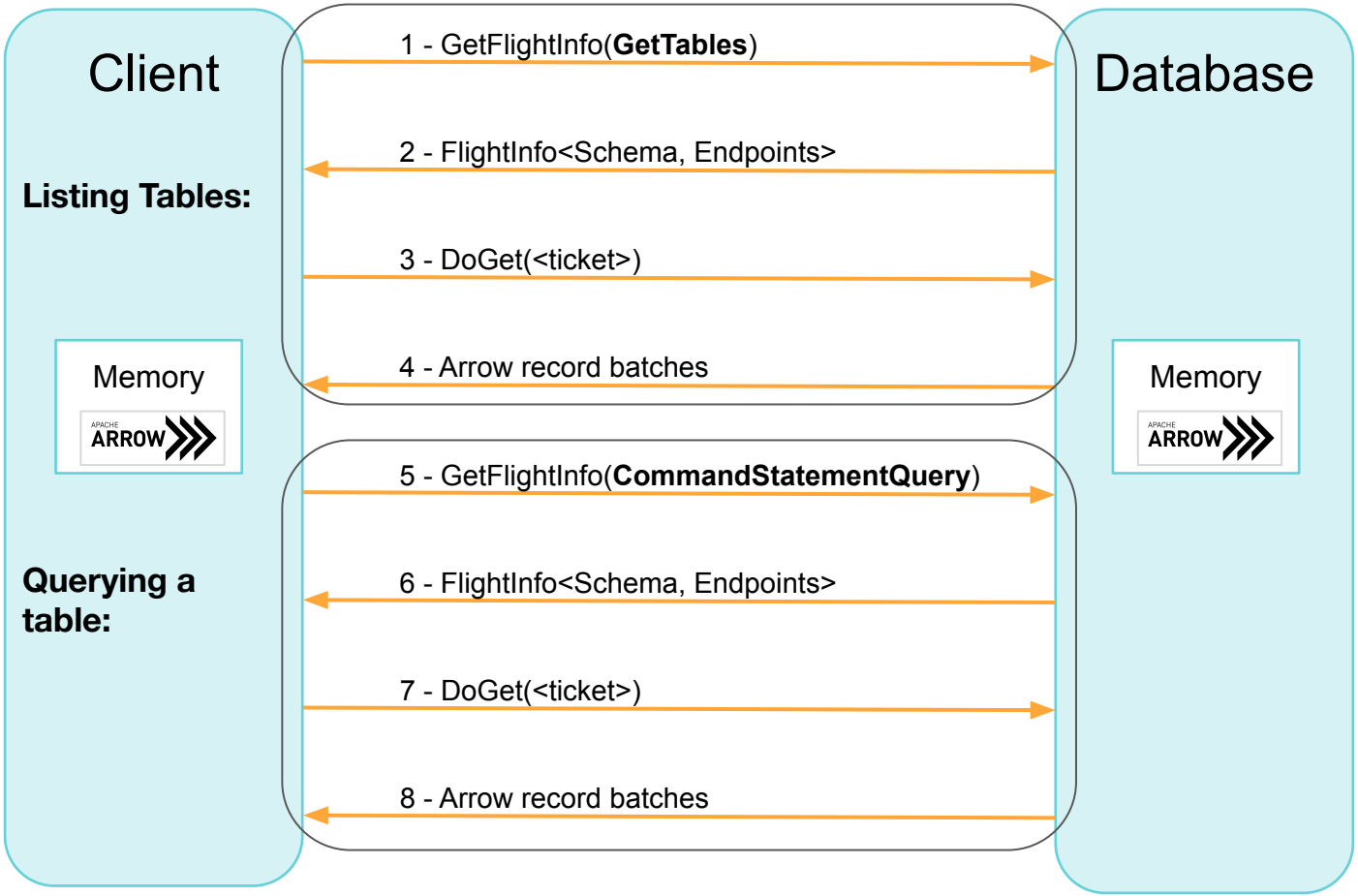
Enter: Arrow Flight SQL

What is Arrow Flight SQL?

- **Client-server protocol for interacting with SQL databases built on Arrow Flight**
  - Allows databases to use Arrow Flight as the transport protocol
  - Leverage the performance of Arrow and Flight for database access

- **Set of commands to standardize a SQL interface on Flight:**
  - Query execution
  - Prepared statements
  - Database catalog metadata (tables, columns, data types).
  - SQL syntax capabilities

- **Language and database-agnostic**
  - A single set of client libraries to connect to any Flight SQL server
  - Clients can talk to any database implementing the protocol
  - No need for a client-side driver per database


Client
Database
Listing Tables:
1 - GetFlightInfo(GetTables)
2 - FlightInfo<Schema, Endpoints>
3 - DoGet(<ticket>)
4 - Arrow record batches
Memory
APACHE ARROW
Memory
APACHE ARROW
Querying a table:
5 - GetFlightInfo(CommandStatementQuery)
6 - FlightInfo<Schema, Endpoints>
7 - DoGet(<ticket>)
8 - Arrow record batches

Arrow Flight SQL Commands

Metadata requests:

- CommandGetCatalogs
- CommandGetCrossReference
- CommandGetDbSchemas
- CommandGetExportedKeys
- CommandGetImportedKeys
- CommandGetPrimaryKeys
- CommandGetSqlInfo
- CommandGetTables
- CommandGetTableTypes

Queries:

- CommandStatementQuery
- CommandStatementUpdate

Prepared statements:

- ActionClosePreparedStatementRequest
- ActionCreatePreparedStatementRequest
- CommandPreparedStatementQuery
- CommandPreparedStatementUpdate

Isn’t Arrow Flight SQL Enough?

	JDBC/ODBC	Custom protocols	Arrow Flight	Arrow Flight SQL
Standardize client interface	✅	❌	✅	✅
Standardize client database access interface	✅	❌	❌	✅
Standardize server interface	❌	❌	✅	✅
Prevent unnecessary serialization	❌	✅	✅	✅

Isn’t Arrow Flight SQL Enough?

- Arrow Flight SQL is fairly new, will take time to be widely adopted by existing client tools (especially the enterprise BI ones)
  - Most BI tools already support ODBC or JDBC

Enter: Arrow Flight SQL ODBC & JDBC Drivers

Arrow Flight SQL ODBC & JDBC Drivers

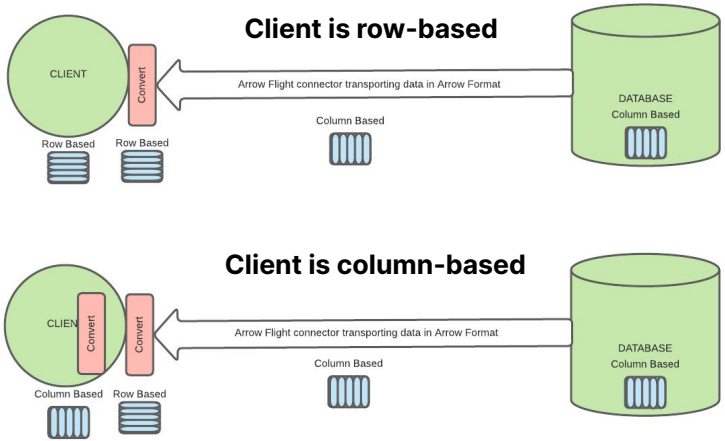
- ODBC & JDBC Drivers built on top of Flight SQL client libraries
  - A single driver to connect to any Flight SQL server
  - Supports arbitrary server-side options as connection properties
- **The drivers will work with any ODBC/JDBC client tool without any code changes**
- Completely open source

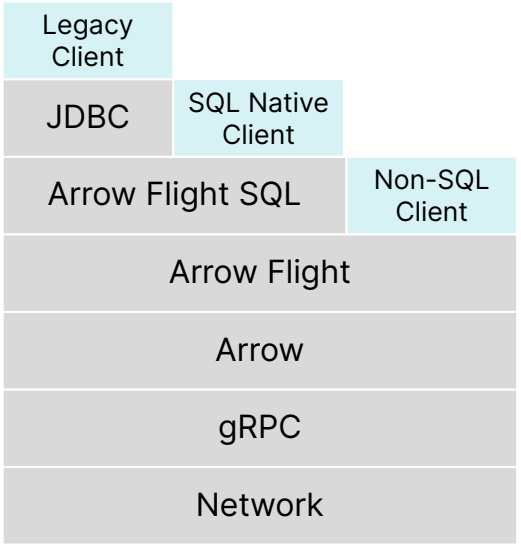
Traditional ODBC/JDBC vs Arrow Flight SQL ODBC/JDBC

	Traditional ODBC/JDBC	Arrow Flight SQL ODBC/JDBC
Driver Management	install and manage a driver **for each database**	install and manage **a single driver** for all supported databases
Performance	**row-based transfer**, so not great for large amounts of data	**column-based transfer**, so benefit from the usual columnar benefits

When wouldn't I use Arrow Flight ODBC/JDBC Drivers?

- Generally preferable to have native Flight SQL applications
  - Easier to harness future features like multiple endpoints
  - Better performance if your client is column-based

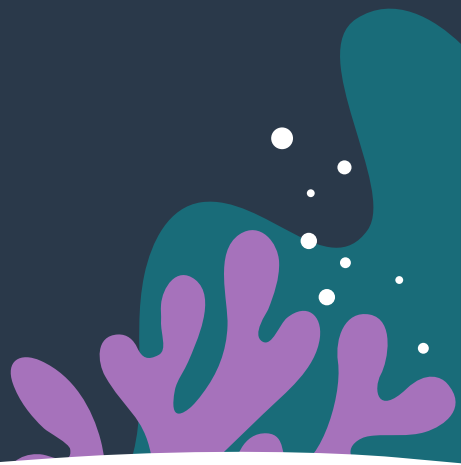




What if we could have for our data transfers...

- ✓ Standardized APIs
  - Arrow Flight for arbitrary tabular transfers, Arrow Flight SQL for transfers from databases
- ✓ No need for many different drivers
  - Both client and server side APIs are standardized
- ✓ High performance transfers
  - Transfers via highly optimized Arrow format
- ✓ Parallel data transfers (1:many, many:many)
  - Lays the foundation for parallel data transfers
- ✓ Easy to implement on server side, easy to use on client side
  - Server-side implementation is as easy as implementing a few RPC request handlers
  - A single set of client libraries to connect to any Flight SQL server

Q&A



Thanks!

jason@dremio.com