Voltron Data is a company that focuses on the advancement of Apache Arrow. The Data Thread of 2022 gave valuable information regarding the development, and future expansion of the Apache Arrow platform. Apache Arrow is a platform that can analyze memory in a server’s RAM. Most importantly it can work in any language and defines columnar memory formatting standards which allows faster processing of data. It can create libraries for software and allows for data to transfer without having to translate it into a different format first.

The keynote speakers of The Data Thread were Jacques Nadeau and Wes McKinney. Jacques Nadeau is the Co-Founder & CEO, Sundeck and Co-founder of Apache Drill. Meanwhile, Wes McKinney is the CTO/Co-Founder at Voltron Data and creator of Python pandas. Apache was created based off their shared issues and experiences. Wes came from the data science ecosystem in the Pandas environment and constantly ran into issues with memory while running large datasets. He was interested in building a bridge between the python ecosystem and data science. There was an issue with data and connectivity which is how he became interested in this problem. Jacques had a similar experience but was more focused on the creation of a good open source project and its construction.

Apache is open-source, so the project was paid for and there were many companies that backed the beginning. Funding for Apache can be seen as one of its points of success. Part of what makes open source successful is the willingness to share with others. There is a focus on building a community so that others can feel that they have ownership over parts of a project. Apache is a project that is language agnostic so it allows for everyone to have ownership in their own way. For example, most developers prefer Java, but not all during the creation. Below is a detailed overview of the software: Apache Arrow and its frameworks.

The ”Deconstructed Database” of Apache can be split into three categories: In-Memory Computing, Core Specification, and Connectivity. In-Memory Computing includes compute primitives, execution engines, expression compilation, and distributed scheduling. Primitives are the basic interfaces of code that can be used to build more advanced programming elements or interfaces. Execution engines is a computer program which can interpret a clinical guideline represented in a computerized format and perform actions towards the user of an electronic medical record. A guideline execution engine needs to communicate with a host clinical information system. Distributed scheduling involves software solutions that are capable of launching unattended scheduled jobs or workloads across multiple servers.

Core specification includes Columnar In-Memory Format, C Data Interface, and IPC Protocol. Columnar In-Memory Format can reduce the amount of data gathered from disk by allowing access to only the columns that are relevant for the specified query. C Data Interface highlights the cross-language platform of Apache Arrow. The C Data Interface carries no dependencies except a shared C ABI between binaries which use it. C ABIs are platform-wide standards which are followed by all compilers. To interact with the C Data Interface at the C or C++ level, the only thing you have to include in your code are two struct type declarations. The C Data Interface passes Arrow data buffers through memory pointers.IPC Protocol stands for inter-process communication. It is a mechanism that allows processes to communicate with each other and sync their actions. The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other through both: Shared Memory.[fragment]

Lastly, connectivity includes Flight RPC + FlightSQL, Database APIs (ADBC), Filesystem Interfaces, and File Format Interfaces. Flight RPC + FlightSQL is a new client-server protocol used for interacting with SQL databases that makes use of the Arrow in-memory columnar format and the Flight RPC framework. Flight SQL means database servers can implement a standard interface that is designed around Apache Arrow and columnar data from the start. APIs like ODBC provide bulk access to result buffers, this data must still be copied into Arrow arrays for use with the broader Arrow ecosystem, as implemented by projects like Turbodbc. Flight SQL makes full use of the Flight RPC framework and its extensibility, defining additional request/response messages via Protobuf. FlightSQL defines methods to query database metadata, execute queries, or manipulate prepared statements. Database APIs stands for DataBase Application Programming Interface and it is a library that lets Python connect to the database server.

While Apache was the biggest star of The Data Thread, it was not the only one. Another interesting program was Ibis. Ibis is an open-source data analysis framework that enables a 100% Python user workflow on top of big data systems. It is a new Apache licensed project from Cloudera. Ibis is a Python analytics library designed to provide the convenience of pandas’ APIs with the scalability of analytic SQL engines like BigQuery. It does this in a type-safe way, letting you build analytics expressions that compile to SQL and run on your favorite large-scale SQL engine. Some current features of Ibis ~~can~~ include the following:

* Comprehensive support of Impala functionality.
* Interoperability with pandas.
* Instruments that simplify interactions with HDFS.
* A pandas-like semantically complete data expression system that covers even such relational data concepts as self-joins, window functions, correlated and uncorrelated subqueries.
* High level analytics tools like bucketing, top-k, histogram, and value\_counts.

When you execute Ibis expressions, they turn into pandas DataFrames, which gives you access to the ecosystem of Python data libraries once you don’t need the scale that distributed SQL provides. Ibis proposes is one single API for data manipulation, and a core engine that will translate this high-level API in lower-level code, depending on the back-end. So, if we use a pandas back-end, Ibis will translate to pandas python code. If we use a BigQuery back-end, Ibis will translate to SQL. The user writes code using the Ibis API and just swaps the back-end between training and prediction.

Lenesha Williams