What *is* Apache Arrow?

Apache Arrow is a software development platform for building high performance applications that process and transport large data sets. It is designed to both improve the performance of analytical algorithms and the efficiency of moving data from one system (or programming language to another).

A critical component of Apache Arrow is its **in-memory columnar format**, a standardized, language-agnostic specification for representing structured, table-like datasets in-memory. This data format has a rich data type system (included nested and user-defined data types) designed to support the needs of analytic database systems, data frame libraries, and more.

The project also contains implementations of the Arrow columnar format in many languages, along with utilities for reading and writing it to many common storage formats. These official libraries enable third-party projects to work with Arrow data without having to implement the Arrow columnar format themselves. For those that want to implement a small subset of the format, the Arrow project contains some tools, such as a C data interface, to assist with interoperability with the official Arrow libraries.

The Arrow libraries contain many software components that assist with systems problems related to getting data in and out of remote storage systems and moving Arrow-formatted data over network interfaces. Some of these components can be used even in scenarios where the columnar format is not used at all.

Lastly, alongside software that helps with data access and IO-related issues, there are libraries of algorithms for performing analytical operations or queries against Arrow datasets.

Why define a standard for columnar in-memory?

Traditionally, data processing engine developers have created custom data structures to represent datasets in-memory while they are being processed. Given the “custom” nature of these data structures, they must also develop serialization interfaces to convert between these data structures and different file formats, network wire protocols, database clients, and other data transport interface. The net result of this is an incredible amount of waste both in developer time and in CPU cycles spend serializing data from one format to another.

The rationale for Arrow’s in-memory columnar data format is to provide an out-of-the-box solution to several interrelated problems:

- A general purpose tabular data representation that is highly efficient to process on modern hardware while also being suitable for a wide spectrum of use cases. We believe that fewer and fewer systems will create their own data structures and simply use Arrow.
- Supports both random access and streaming / scan-based workloads.
- A standardized memory format facilitates reuse of libraries of algorithms. When custom in-memory data formats are used, common algorithms must often be rewritten to target those custom data formats.
- Systems that both use or support Arrow can transfer data between them at little-to-no cost. This results in a radical reduction in the amount of serialization overhead in analytical workloads that can often represent 80-90% of computing costs.

- The language-agnostic design of the Arrow format enables systems written in different programming languages (even running on the JVM) to communicate datasets without serialization overhead. For example, a Java application can call a C or C++ algorithm on data that originated in the JVM.