I am sure the voltron session was great! For me, I was unable to get access to the session. I also had technical difficulties for the first hour of the session which was a three hour session. Then after that, I had difficulties retaining the information on Discord because I was watching from someone elses screen and they had connectivity issues from time to time. So gathering information for the seminar was actually very interesting! I had to go on youtube and rewatch all the recordings and hope I gained enough knowledge to write about the experience. I did take notes from what I watched on Youtube. I noticed that all the videos weren’t accessible to the public so I hope I do not leave out critical information due to not being able to actually attend the session. So for starters I watched a video about Ibis. Ibis is an open source library from what was told on the video. Ibis simplifies reading SQL as well as making it easier to use and understand for someone who is a new programmer, graduate student, or employee. Another benefit that was said on the Ibis section was saying that Ibis can be paramertized. By parameterizing Ibis, it can make it so much more easier to see errors that get displayed in the output from the code that is present. It is also easier to fail in Ibis. It is also noted that Ibis can be installed with a pip installer. There are drawbacks which are spoken upon in Ibis. These drawbacks include performance, abstraction, and features. For performance, there can be some overhead. For abstraction, there can be an additional layer. There are also execution phases in Ibis. These execution phases includes expression graph construction, compilation, and execution. Then from there the next session was about arrow and substrait. Arrow is defined as a universal standard for representing tabular data. Substrait is a universal standard for representing relational operations. Next I really honed in on TorchArrow. TorchArrow is a Python DataFrame library inspired by Pandas. The presenter said, “There is seamless handoff with PyTorch or other model authoring”. There are multiple execution runtines that are supported. These include the high performance CPU backend via Velox, “Graph Mode” which is Trace TorchArrow program for lowering operations, and GPU backend that will hopefully be implemented in the future. There was also criteo preprocessing which simplifies the operations done on tables within databases. There were also design aspects talked about at the seminar. In order to design an Arrow Library to be asynchronous, it was said that columnar bytes need to correspond to a new row, and row bytes correspond to a new column. There were also talks about Arrow2. Arrow2 is a re-write of the official Rust Arrow Library. In this library, it implements the full Arrow specification. The library also passes all integration tests against C++ and Java. There was also mention of the core design of Arrow2. For every format, there is a separate API to “read” and to “deserialize”. The read API is asynchronous but has the potential to be synchronous in the case there for local file systems. All the above information was in regards to the Libraries track for the Voltron Seminar. The next portion I will combine talking about the Languages Track and Use Cases Track. The first section I will discuss is the introduction to arrow for python programmers. Python does not utilize dealing with cores that well. As it is known, Arrow deals with cores in order to do its processing. Arrow stores data column by column. Python, however, has to access data in the form of bits. That is all for Arrow and Python. Now Arrow Flight SQL. Arrow Flight SQL is a wire protocol for SQL databases. It also executes queries as well as getting database metadata which can be very essential for operations. Arrow Flight SQL was built on top of Arrow Flight. Flight SQL is formatted in a way in which it is columnar. Acero is an Arrow native open source streaming query engine written in C++. When it comes to Arrow for R users, Apache Arrow is a multi-language toolbox for accelerating data interchange and in-memory processing. This is good for the language of R given it is used for statistical analysis and other data processing purposes. Speaking of R, there are many ways one can contribute to the Arrow R package. Some of the ways include contributing to the documentation, cookbook, add bug reports, and code contributions. The code to contribute to Arrow R package is done in C++ code. Apache Arrow Flight SQL has high performance, simplicity, and interoperability for data transfers. It comes in to offer standardization. Arrow Flight SQL allows databases to use Arrow Flight as the transport protocol. This leverages the performance of Arrow and Flight for database access. There are comparisons between the traditional ODBC/JDBC and Arrow Flight SQL ODBC/JDBC. When it comes to drive management, the traditional way installs and manages a driver for each database, while the Arrow Flight SQL version installs and manages a single driver for all databases. In regards to performance, the traditional way is a row-based transfer, so not great for large amounts of data, while the Arrow Flight SQL version is column-based, so benefit from compression of data over the wire. There were also microkernal notebooks that were talked about. Some benefits of these notebooks is that one can have versions of the notebook be referenced to at that cell’s point in time. Some other features include better error-handling and automatically re-run stale cells. However there are some downfalls which include migrating state between kernels. There was also mention of GraphQL with Apache Arrow. Here GraphQL and Apache Arrow can be used to have consistent new data points, multiple internal consumers, multiple different data stores, and non-standard schemas. Some benefits of dynamic schema generation (which is invoked with GraphQL and Apache Arrow) includes faster time to market for new data points, ease of automation extension, data source abstraction, reduced engineering effort necessary for maintenance, and consistent query patterns allow for templated query generation. Apache Arrow on the Web and Beyond included an Arrow JS Library. According to the presenter, “Arrow JS Library is completely written in TypeScript, available for Node and the browser, contains ESM and CJS support, and is treeshakable”. Some benefits include values being stored in binary format, which reduces parsing overhead and more compact. It is also multi-thread capable which results in a table with a large number of rows. There is also the potential to have hardware acceleration, meaning data from columns can be placed in the gpu very quickly. NVIDIA RAPIDS is a use case that Arrow JS has. The above was the languages track. The remaining portion of this two page report will overview some use cases. Ceph provides three types of storage interfaces. These include File, Object and Block. There is no central point of failure. Uses CRUSH maps that contain Object -ODS mapping. There is also an extensible object storage layer via the Ceph Object Classes SDK. Arrow Flight can be used for live batch data streaming. This means fast serializaiton which means low write to read latency. Snorkel Flow unlocks the complete data-centric AI development workflow. In the “Developers Using Arrow with Tableau” session, the presenter created a library called HyperArrow. This consists of a C++ core with a Python C++ extension. This includes a much cleaner type system, much improved null handling, easy to extend support for reading batches of data and a user experience very similar to pantab. There were also use cases for maximizing the performance of DNA analysis using apache arrow. In conclusion, ArrowSAM significantly improves runtime and efficiency of DNA analysis. Apache Arrow solves important limitations including in disk storage, analysis of tabular row-oriented data, and use of multiple programming languages. ArrowSAM is five times faster in difficult workflow stages and is four times more memory efficient than custom in-memory workflows. Apache Arrow and DataFusion has helped with Cube.js / Cube Store, InfluxDB IOx, FLOCK, VegaFusion and many more programs! Velox is a generic C++ database acceleration library that takes a fully optimized physical plan as input. There is no SQL parser, no Dataframe layer, and no optimizer. However, there is a lot of adaptability. Some use cases of Velox is its analytics, realtime infrastructure, transactional uses and machine learning uses. The Velox Library contains types, vectors, expression evaluation, functions, operators, Input/Output, and Resource Management. There was also talk about VAST. VAST is an embeddable engine for high-volume security telemetry and security operations.
Overall from the notes I took, the Voltron seminar was very informative! Glad I was able to gather some new information!