In-Memory Computing SUMMIT | EUROPE 2018

Predicting Share Prices in Real-Time with Apache Spark and Apache Ignite

MANUEL MOURATO

Summary

- What is the stock market?
- Making a profit on volatility: Scalp trading
- Looking at first hour price swings
- The need for an in-memory driven architecture
- Proposed Architecture
- Data Source
- Data Ingestion
- Data Processing
- In Memory Storage
- Persistent Storage
- Equity Classification
- Tableau: Visualizing the data
- Future Work
- Questions
- Annex

What is the stock market?

- When companies require more capital to grow their business , they may decide to “go public”.

- By making an initial public offering (IPO), companies receive money from institutional investors, based on the value of the company itself and the number of shares they make available.

- Then, in the secondary markets, individual market players also enter the “game”, by buying and selling these shares/stock, between themselves and also with institutional investors.

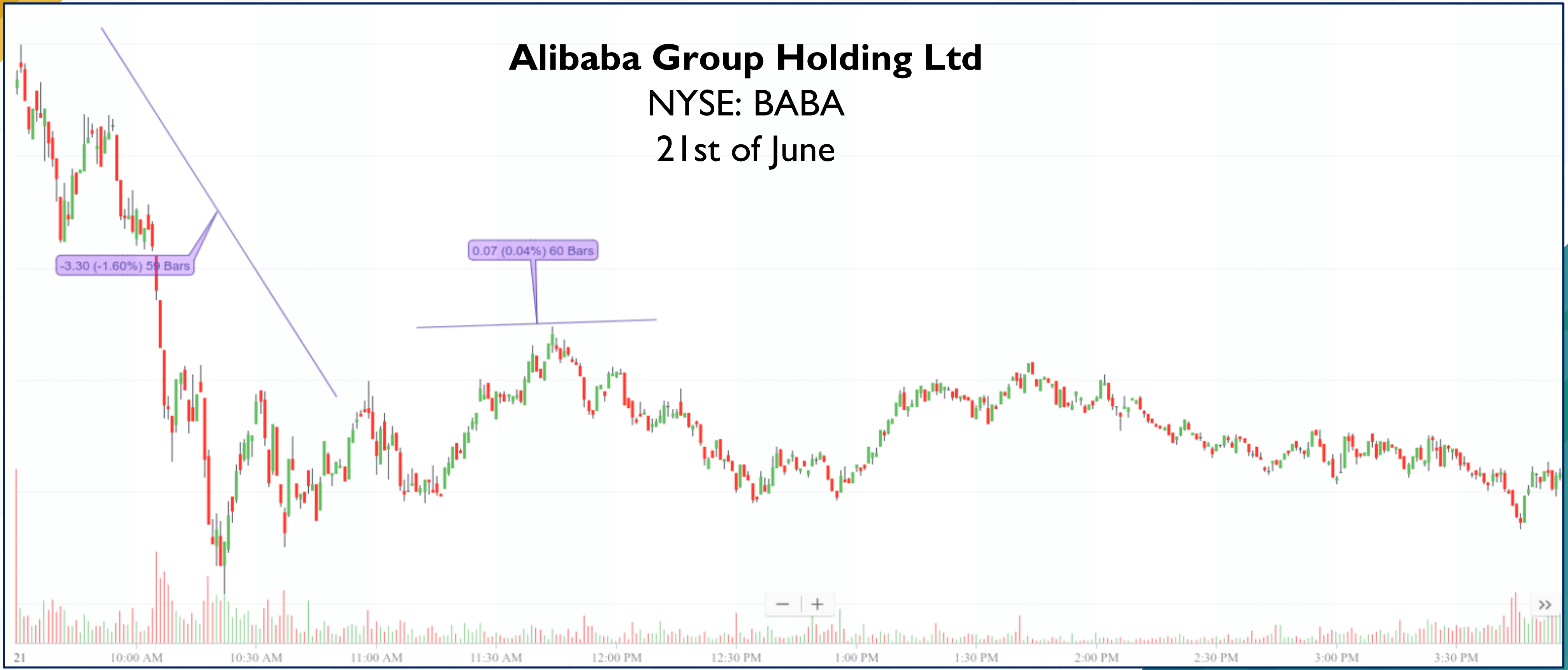
What is the stock market?

Main types of market players	
Investors	**Traders**
Keep stocks for large periods of time (months to years)	Keep stock for a few seconds, minutes or hours
Does not require a minimum financial amount	Require at least 25000 dollars to trade daily stocks
Can invest with no time constraints	Need to be active when the market is open and active
Gains compound slowly (10% return on initial capital per year)	Gains compound quickly (3% return on initial capital per day

Making a profit on volatility: Scalp Trading

- Scalp trading specializes in taking profits on small price changes, generally soon after a trade has been entered and has become profitable.
- Scalp traders must have a high win/loss ratio.
- Stop loss strategy should be of around 0.1% from your entry price.
- Traders place anywhere from 100 to a couple thousand trades in a single day.
- An interesting approach to scalping is to take advantage of the up-and-down price fluctuations between the open and close of a trading session (**stock's intraday volatility**).
- Buying and selling by individual investors is especially heavy in the minutes immediately after the market opens in the U.S. at 9:30 a.m. Eastern time, when the chances of getting the best price for a stock are lower and swings tend to be bigger.
- The difference between the bid and ask prices of shares in the S&P 500 was 0.84 percentage point in the first minute of trading, according to data from ITG.
- That gap shrinks to 0.08 percentage point after 15 minutes and to less than 0.03 percentage point in the final minutes of the trading day.

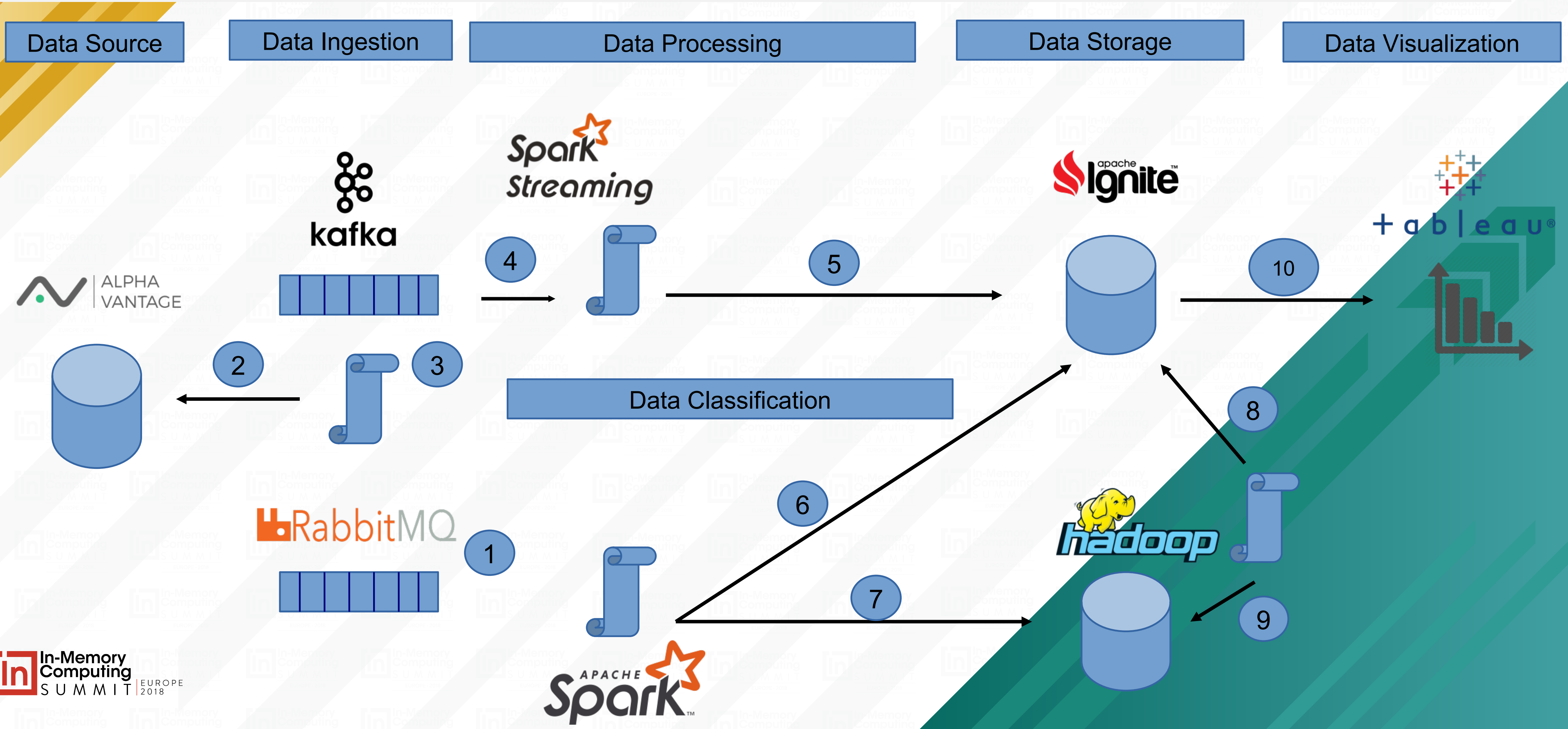
Looking at first hour price swings



The need for an in-memory driven architecture

- Scalp traders require a solution to help them make decisions in a matter of a few minutes.
- It should provide data from multiple company equities, so that a lot of trading can be done.
- Real prices should be available on a minute to minute basis.
- Identify trends from equity prices and determine if equities should be bought or sold in that minute.
- Provide an intuitive visualization for traders and investors.
- Queries to data should return immediate results.
- Historic data should be stored for an a posteriori analysis.
- As more data sources are added, the architecture should be able to seamlessly scale.

Proposed Architecture



Data Source

- Alpha Vantage Inc. is a leading provider of free APIs for realtime and historical data on stocks, physical currencies, and digital/cryptocurrencies.
- It contains a Time Series Intraday API with minute to minute equity data updates.
- Equity info is retrieved either in JSON or CSV format.

Downsides:

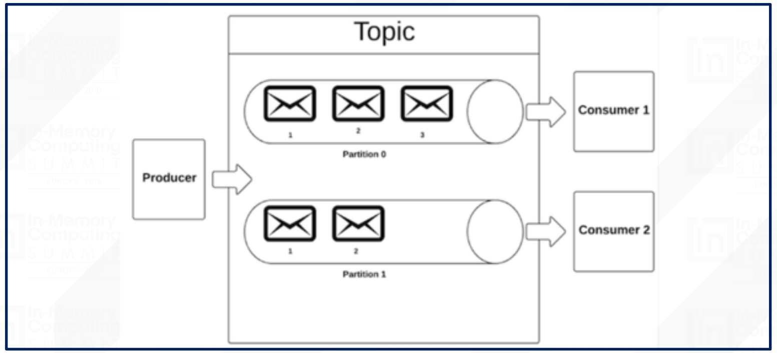
- Single point of failure: If the Alpha Vantage server becomes unavailable, the whole architecture that follows becomes meaningless.
- Allows a maximum of 3 calls per second using an API key.

ALPHA VANTAGE

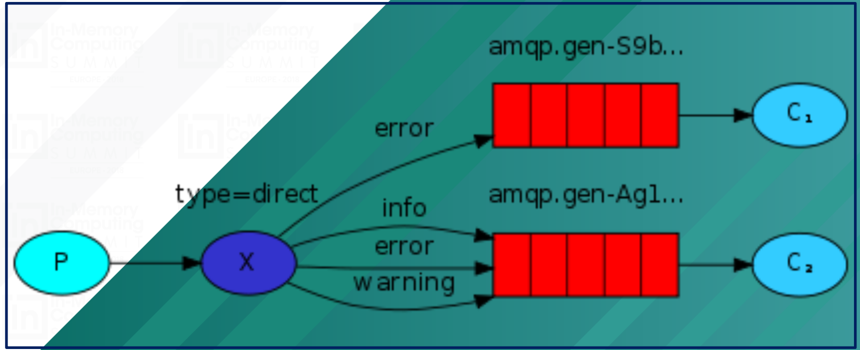
```
{
   "Meta Data": {
      "1. Information": "Intraday (1min) prices and volumes",
      "2. Symbol": "MSFT",
      "3. Last Refreshed": "2018-06-15 16:00:00",
      "4. Interval": "1min",
      "5. Output Size": "Compact",
      "6. Time Zone": "US/Eastern"
   },
   "Time Series (1min)": {
      "2018-06-15 16:00:00": {
         "1. open": "100.3500",
         "2. high": "100.3500",
         "3. low": "100.1000",
         "4. close": "100.1300",
         "5. volume": "27615036"
      } (...)
```

Data Ingestion
Kafka and RabbitMQ

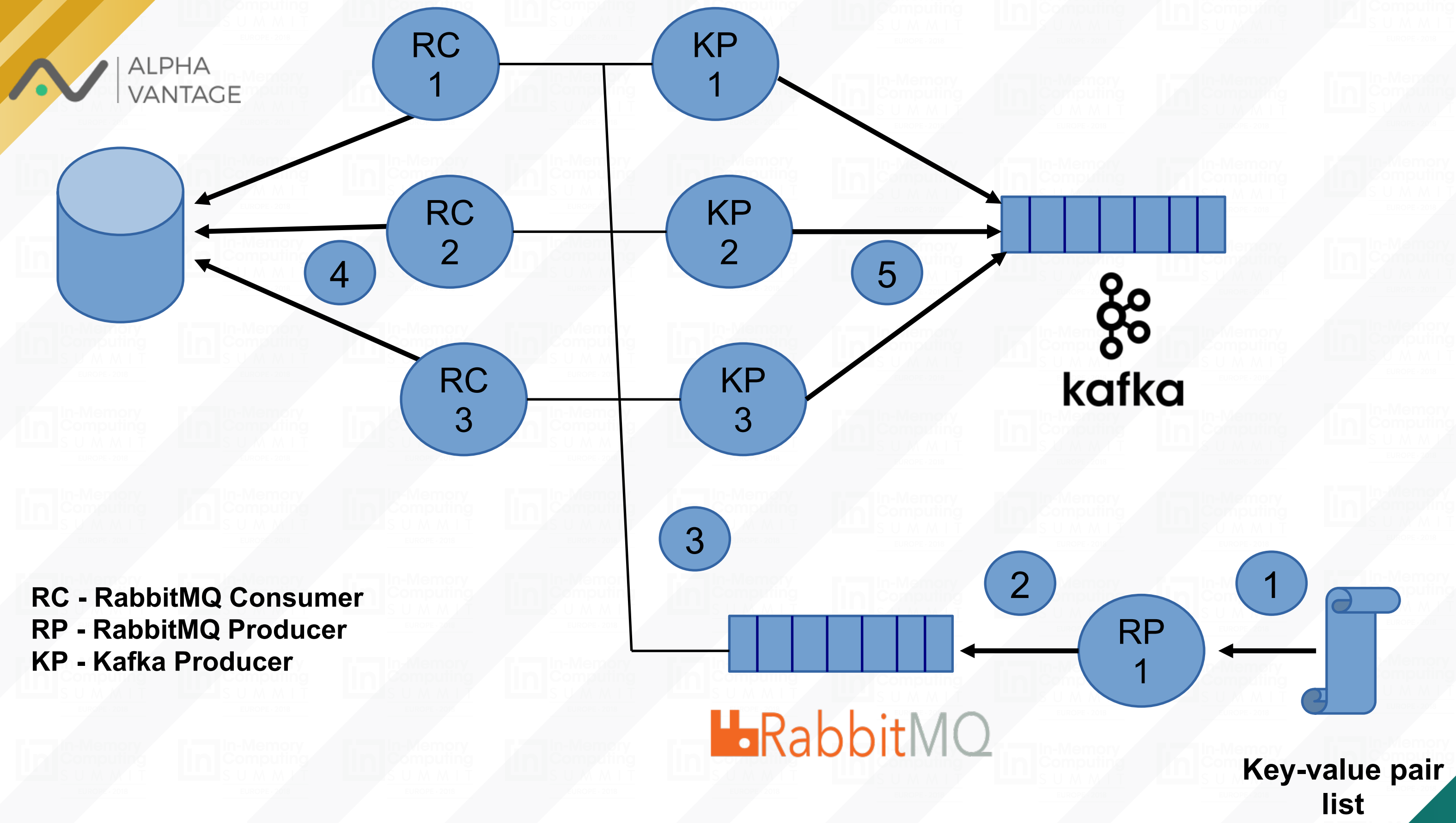
- Apache Kafka is a distributed streaming platform.
- It allows for the publishing and subscription to streams of records.
- It allows for the storage of records in a reliable manner.
- Each record consists of a key, a value, and a timestamp.



- RabbitMQ is a messaging broker - an intermediary for messaging.
- It gives your applications a common platform to send and receive messages, and your messages a safe place to live until received.
- Suited for short message TTLs.



Data Ingestion
Load Balancing and Fault Tolerance

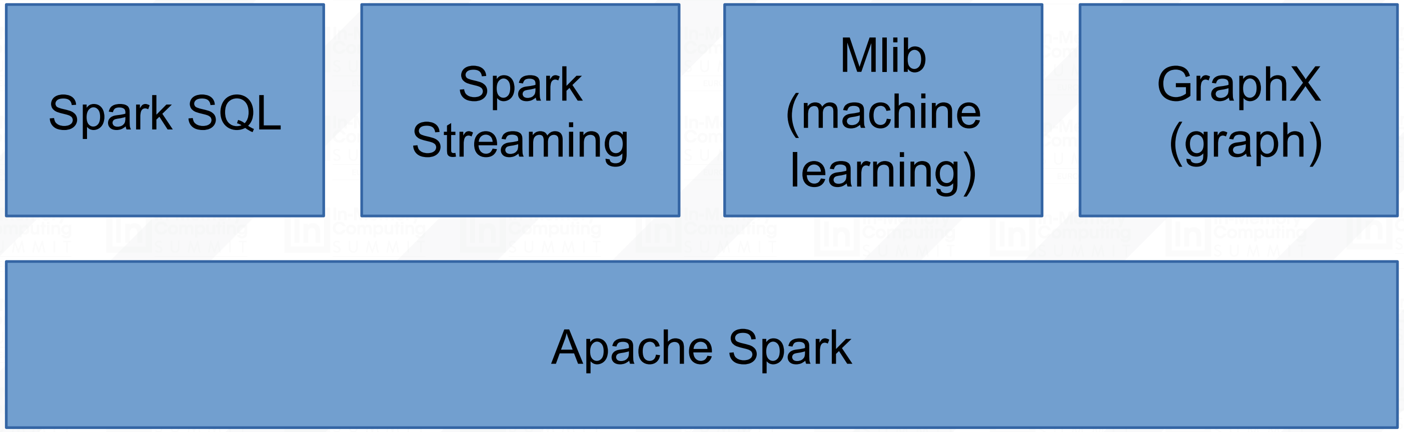


RC - RabbitMQ Consumer
RP - RabbitMQ Producer
KP - Kafka Producer

- There are three Kafka Producers in separate machines.
- There is a RabbitMQ server in another separate machine.
- Every minute, a Rabbit queue is supplied with multiple key pairs: **API_Key-Equity_Symbol.**
- Each Kafka producer will then consume a batch of key pairs, and perform calls to the Alpha Vantage server based on the received parameters.
- If one or more producers goes down for any reason, the other two will still consume key pairs from the Rabbit queue.
- This minimizes data loss, with the only impact being the increase in latency of data retrieval.

Data Processing
Apache Spark

- Apache Spark is a fast and general-purpose cluster computing system.
- It allows for the distribution of tasks, in a parallel fashion, among different machines/executors.
- There are currently 4 modules that expand Spark’s functionality.



Spark SQL

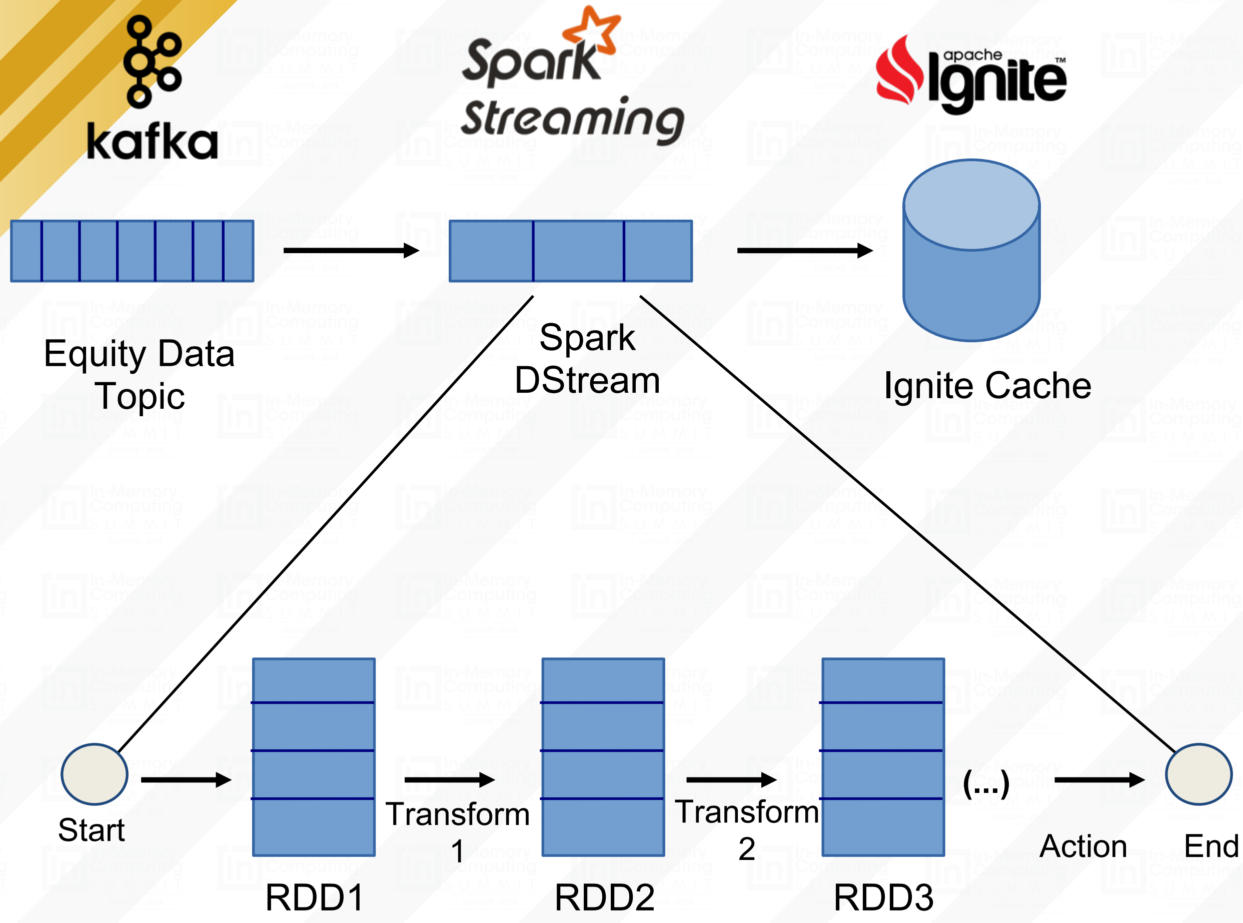
Spark Streaming

Mlib (machine learning)

GraphX (graph)

Apache Spark

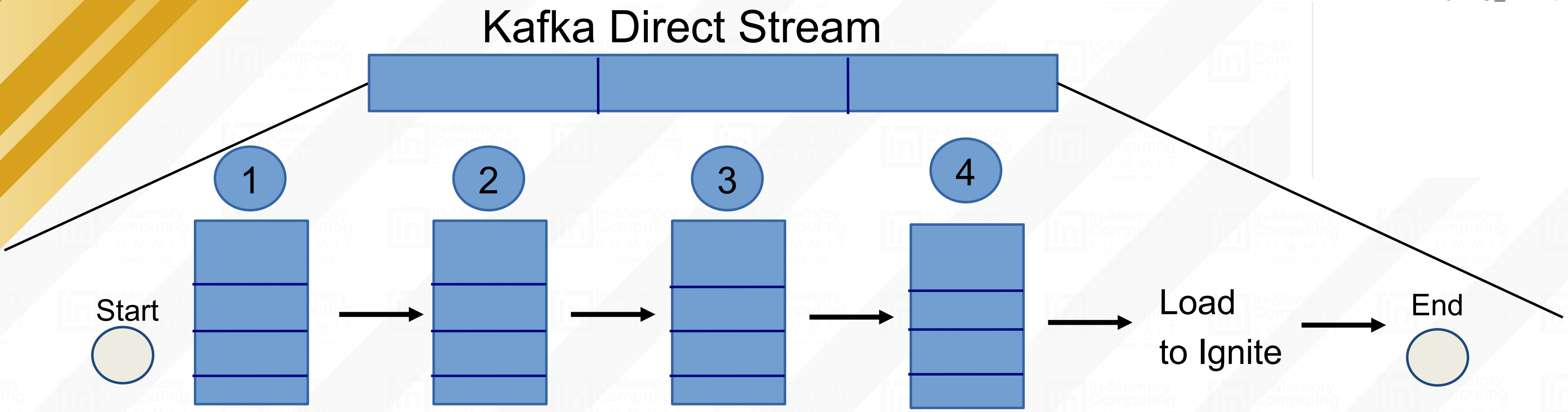
Data Processing
Spark Streaming

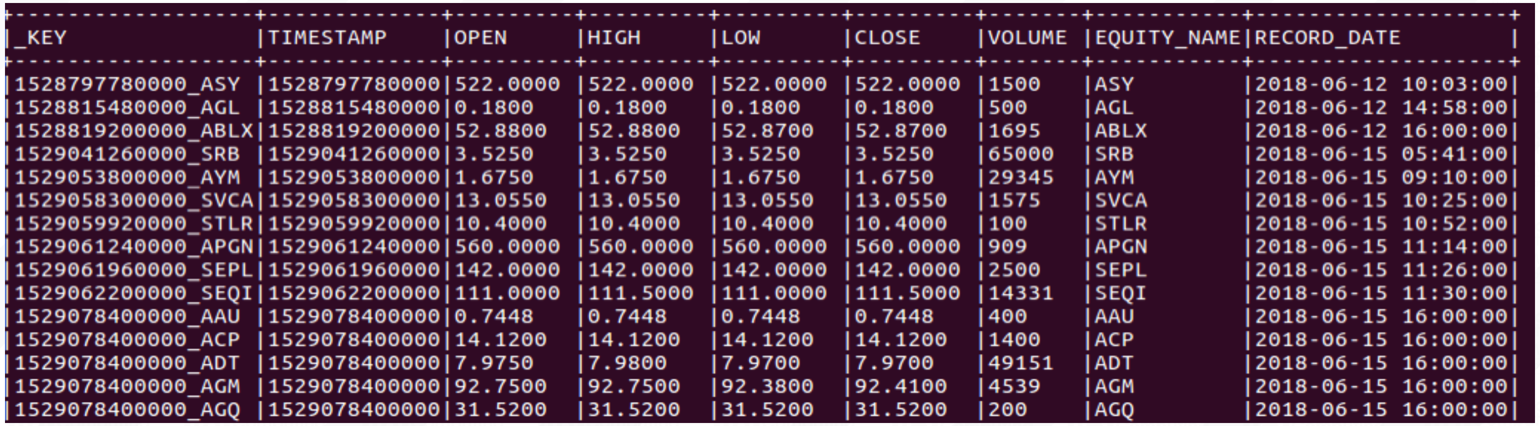


- Traditionally, Spark was used solely as a batch processing tool for great volumes of data, in hourly to daily intervals.
- Its main abstraction is an RDD (Resilient Distributed Dataset), which is divided into partitions that are processed in parallel.
- The Spark Streaming module is an attempt to adapt Spark to near real time scenarios, by using the concept of micro batching.
- A Spark Streaming job is a long running task, which receives and processes data in a fixed time interval.
- Its main abstraction is a DStream, which is a sequence of RDDs from different context executions.

Data Processing
This use case

```
case class Equity_Data(
                          @(QuerySqlField @field)(index = true)   Timestamp: String,
                          @(QuerySqlField @field)(index = false)  open: String,
                          @(QuerySqlField @field)(index = false)  high: String,
                          @(QuerySqlField @field)(index = false)  low: String,
                          @(QuerySqlField @field)(index = false)  close: String,
                          @(QuerySqlField @field)(index = false)  volume: String,
                          @(QuerySqlField @field)(index = true)   equity_name: String,
                          @(QuerySqlField @field)(index = true)  record_date: String
```

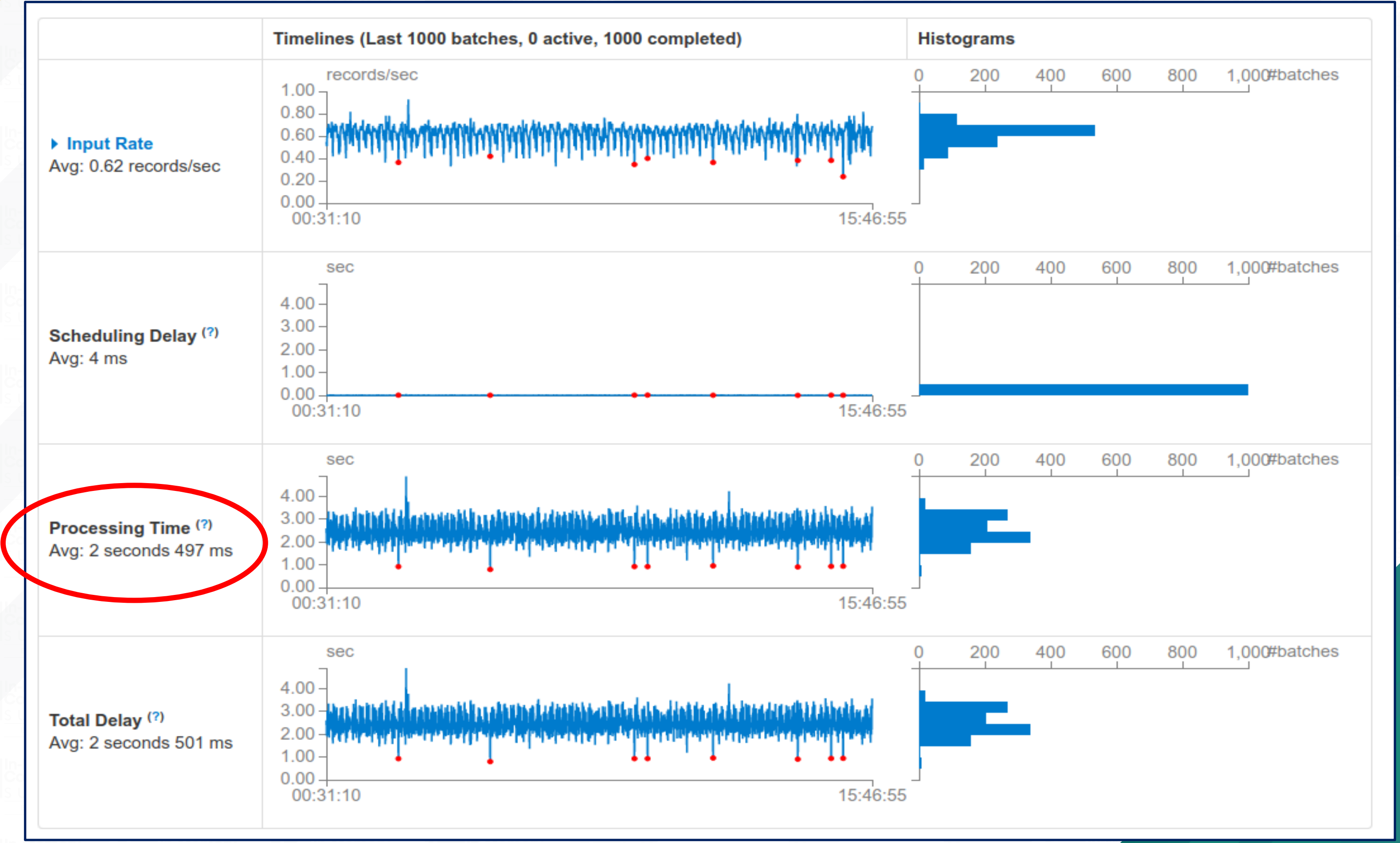




```
+------------------+-------------+---------+---------+---------+---------+-------+-----------+-------------------+
|_KEY              |TIMESTAMP    |OPEN     |HIGH     |LOW      |CLOSE    |VOLUME |EQUITY_NAME|RECORD_DATE        |
+------------------+-------------+---------+---------+---------+---------+-------+-----------+-------------------+
|1528797780000_ASY |1528797780000|522.0000 |522.0000 |522.0000 |522.0000 |1500   |ASY        |2018-06-12 10:03:00|
|1528815480000_AGL |1528815480000|0.1800   |0.1800   |0.1800   |0.1800   |500    |AGL        |2018-06-12 14:58:00|
|1528819200000_ABLX|1528819200000|52.8800  |52.8800  |52.8700  |52.8700  |1695   |ABLX       |2018-06-12 16:00:00|
|1529041260000_SRB |1529041260000|3.5250   |3.5250   |3.5250   |3.5250   |65000  |SRB        |2018-06-15 05:41:00|
|1529053800000_AYM |1529053800000|1.6750   |1.6750   |1.6750   |1.6750   |29345  |AYM        |2018-06-15 09:10:00|
|1529058300000_SVCA|1529058300000|13.0550  |13.0550  |13.0550  |13.0550  |1575   |SVCA       |2018-06-15 10:25:00|
|1529059920000_STLR|1529059920000|10.4000  |10.4000  |10.4000  |10.4000  |100    |STLR       |2018-06-15 10:52:00|
|1529061240000_APGN|1529061240000|560.0000 |560.0000 |560.0000 |560.0000 |909    |APGN       |2018-06-15 11:14:00|
|1529061960000_SEPL|1529061960000|142.0000 |142.0000 |142.0000 |142.0000 |2500   |SEPL       |2018-06-15 11:26:00|
|1529062200000_SEQI|1529062200000|111.0000 |111.5000 |111.0000 |111.5000 |14331  |SEQI       |2018-06-15 11:30:00|
|1529078400000_AAU |1529078400000|0.7448   |0.7448   |0.7448   |0.7448   |400    |AAU        |2018-06-15 16:00:00|
|1529078400000_ACP |1529078400000|14.1200  |14.1200  |14.1200  |14.1200  |1400   |ACP        |2018-06-15 16:00:00|
|1529078400000_ADT |1529078400000|7.9750   |7.9800   |7.9700   |7.9700   |49151  |ADT        |2018-06-15 16:00:00|
|1529078400000_AGM |1529078400000|92.7500  |92.7500  |92.3800  |92.4100  |4539   |AGM        |2018-06-15 16:00:00|
|1529078400000_AGQ |1529078400000|31.5200  |31.5200  |31.5200  |31.5200  |200    |AGQ        |2018-06-15 16:00:00|
```

1 - Original Data
2 - Processed data: JSON
3 - Processed data: Java Class
4 -Timestamp_Symbol-Java Class Pair

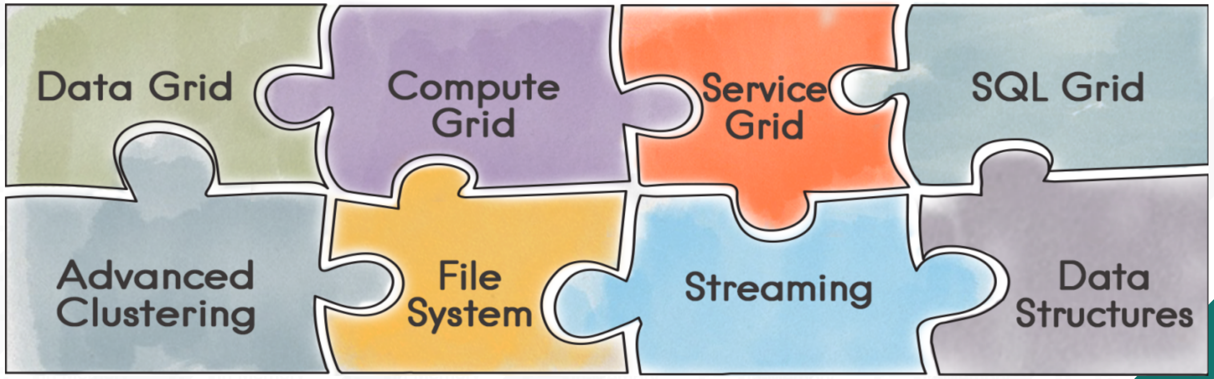
Data Processing Performance



	Timelines (Last 1000 batches, 0 active, 1000 completed)	Histograms
▸ Input Rate Avg: 0.62 records/sec		
Scheduling Delay (?) Avg: 4 ms		
Processing Time (?) Avg: 2 seconds 497 ms		
Total Delay (?) Avg: 2 seconds 501 ms		

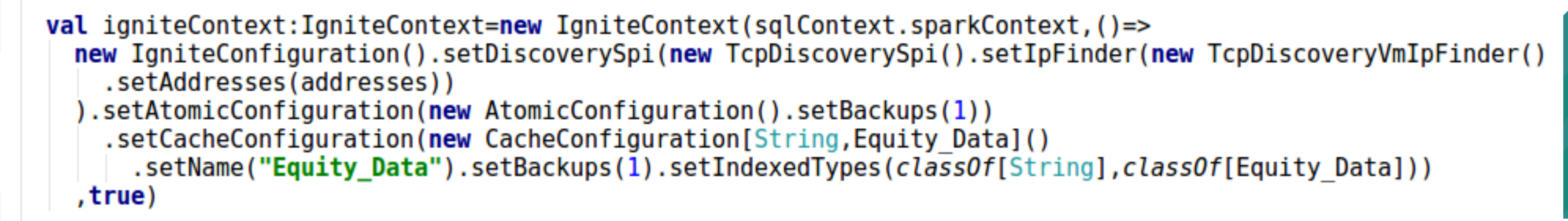
Cache Storage
Apache Ignite

- Apache Ignite is a memory-centric distributed database, caching,
- and processing platform.
  for transactional, analytical, and streaming workloads.
- Extremely simple to scale, using the concept of self discovering nodes.
- Provides a Native Persistence option for full cluster “crash scenarios”.
- Comes with an ANSI-99 compliant, horizontally scalable and fault-tolerant distributed SQL database.
- Allows for different data partitioning strategies based on different cache keys.
- Integrates with multiple visualization tools.

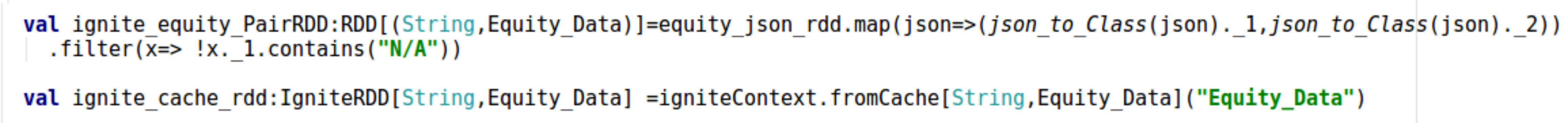


Cache Storage
Ignite-Spark Integration

- With the Ignite Spark integration, RDD’s from a Spark application can be directly mapped into an Ignite cache.
- It provides a shared, mutable view of the same data in-memory in Ignite across different Spark jobs, workers, or applications.
- While Apache SparkSQL supports a fairly rich SQL syntax, it doesn't implement any indexing. With Ignite, Spark users can configure primary and secondary indexes that can bring up to 1000x performance gains.



```
val igniteContext:IgniteContext=new IgniteContext(sqlContext.sparkContext,()=>
  new IgniteConfiguration().setDiscoverySpi(new TcpDiscoverySpi().setIpFinder(new TcpDiscoveryVmIpFinder()
    .setAddresses(addresses))
  ).setAtomicConfiguration(new AtomicConfiguration().setBackups(1))
    .setCacheConfiguration(new CacheConfiguration[String,Equity_Data]()
      .setName("Equity_Data").setBackups(1).setIndexedTypes(classOf[String],classOf[Equity_Data]))
  ,true)
```

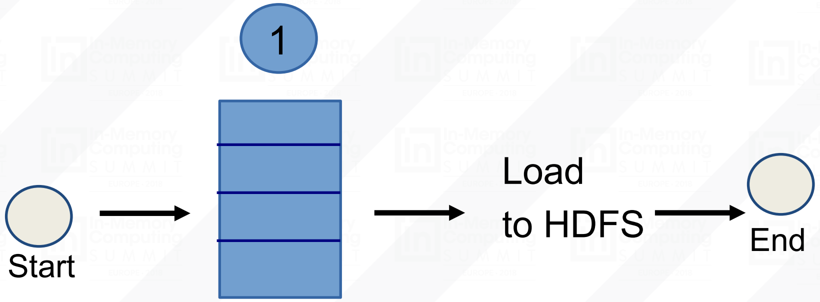


```
val ignite_equity_PairRDD:RDD[(String,Equity_Data)]=equity_json_rdd.map(json=>(json_to_Class(json)._1,json_to_Class(json)._2))
  .filter(x=> !x._1.contains("N/A"))

val ignite_cache_rdd:IgniteRDD[String,Equity_Data] =igniteContext.fromCache[String,Equity_Data]("Equity_Data")
```

Persistent Storage
HDFS

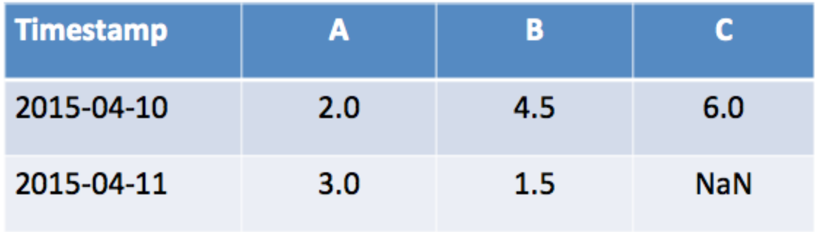
- The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware.
- HDFS is highly fault-tolerant.
- Suited for large files.
- Allows for data to be organized in a directory like structure.
- Integrates with Apache Ignite.



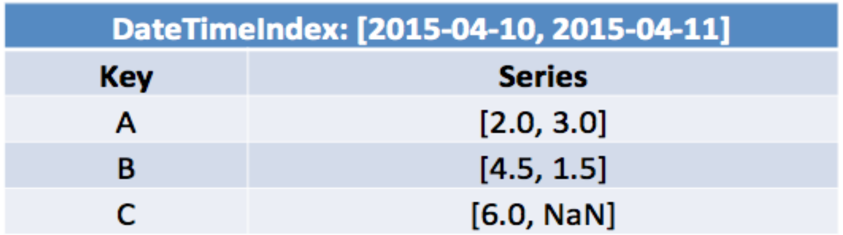
1 - Ignite Dataframe

Equity Classification
Spark-ts

- Time Series for Spark (spark-ts) is a Scala / Java / Python library for analyzing large-scale time series data sets.
- It offers a set of abstractions for manipulating time series data, as well as models, tests, and functions that enable dealing with time series from a statistical perspective.



Timestamp	A	B	C
2015-04-10	2.0	4.5	6.0
2015-04-11	3.0	1.5	NaN



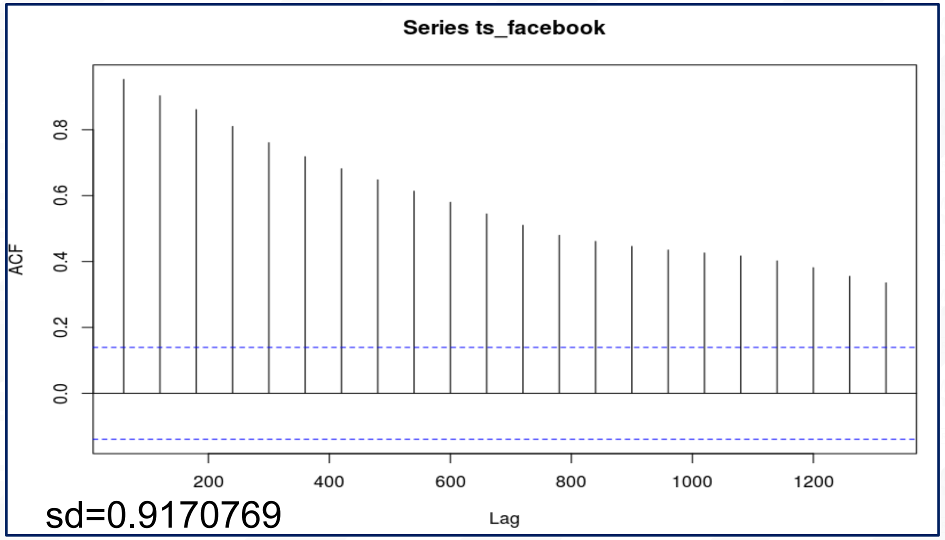
DateTimeIndex: [2015-04-10, 2015-04-11]	
Key	Series
A	[2.0, 3.0]
B	[4.5, 1.5]
C	[6.0, NaN]

- Each equity prices correspond to a vector, and each vector can be processed in a different machine/thread.
- Data from the last two weeks is loaded into spark to create these vectors.
- N/A values are handled by using a nearest neighbour approach.

Equity Classification
ARIMA

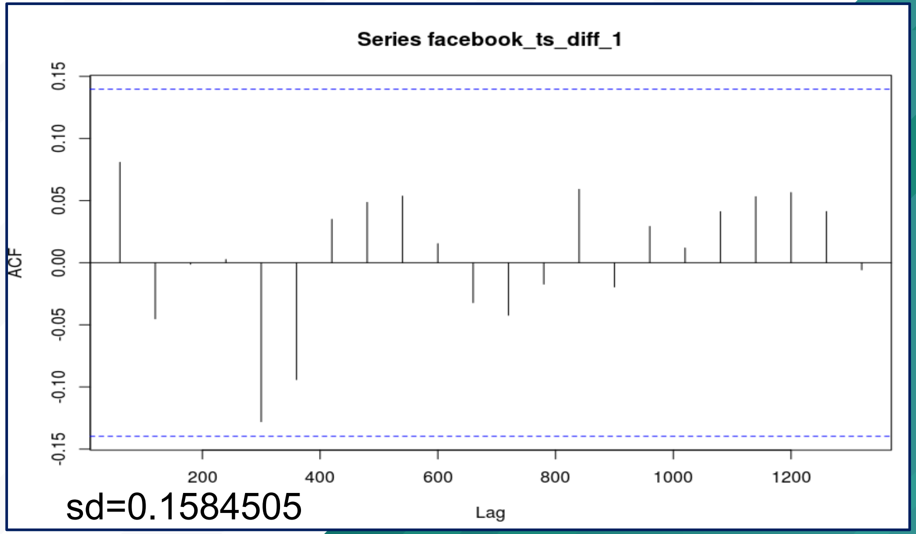
- ARIMA is used to forecast the value for the next 5 minutes.
- ARIMA is an **autoregressive integrated moving average model.**
- It is suited for time series data either to better understand the data or to predict future points in the series.
- Non-seasonal ARIMA models are generally denoted ARIMA(*p*,*d*,*q*).

- **d** is the degree of differencing (the number of times the data have had past values subtracted).
- It should be chosen such that the timeseries becomes stationary, fluctuates around a well-defined mean value and whose autocorrelation function (ACF) plot decays fairly rapidly to zero.



sd=0.9170769

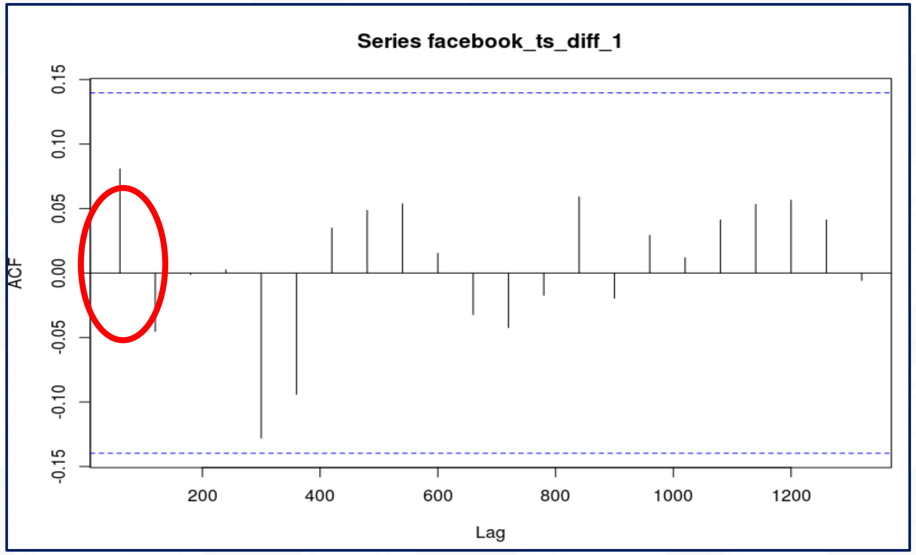
d = 1

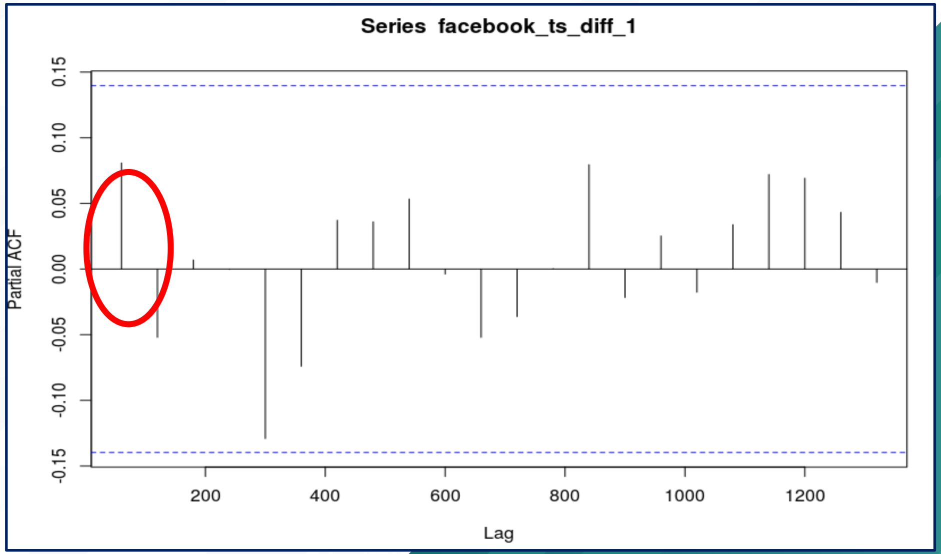


sd=0.1584505

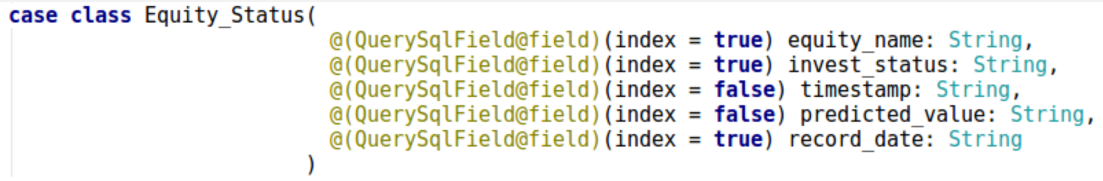
Equity Classification
ARIMA

- **p** is the order (number of time lags) of the autoregressive model.
- It should be chosen such that the PACF of the differenced series displays a sharp cutoff and/or the lag-1 autocorrelation is positive.
- **q** is the order of the moving-average model.
- It should be chosen such that the ACF of the differenced series displays a sharp cutoff and/or the lag-1 autocorrelation is negative.
- After trial and error , the values chosen for the ARIMA were **(1,1,0).**

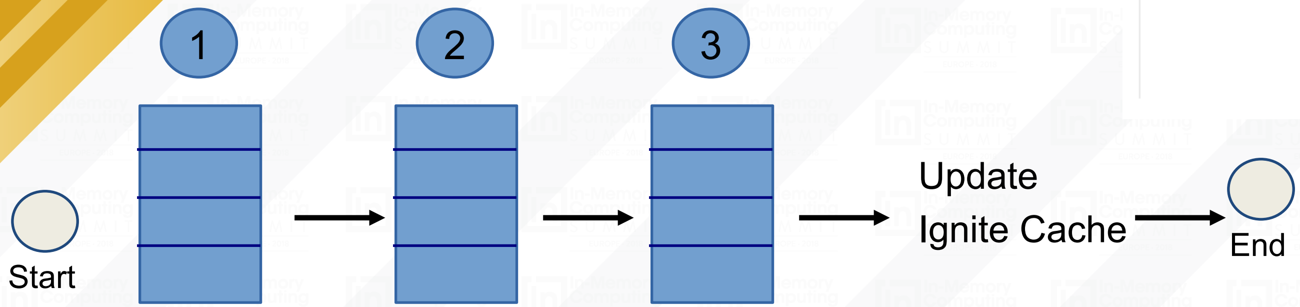




Equity Classification
Results and Performance

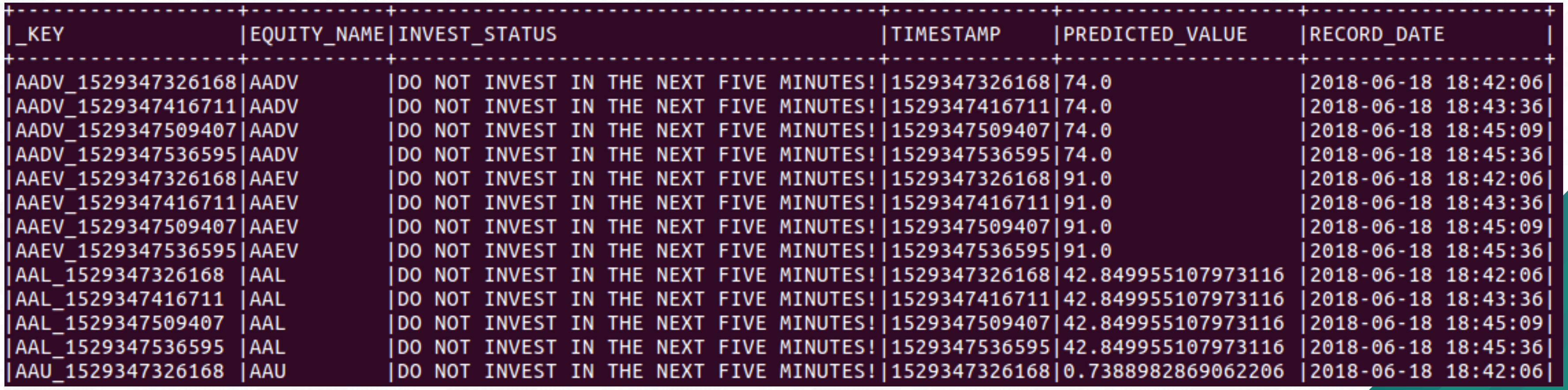


```
case class Equity_Status(
                        @(QuerySqlField@field)(index = true) equity_name: String,
                        @(QuerySqlField@field)(index = true) invest_status: String,
                        @(QuerySqlField@field)(index = false) timestamp: String,
                        @(QuerySqlField@field)(index = false) predicted_value: String,
                        @(QuerySqlField@field)(index = true) record_date: String
                       )
```



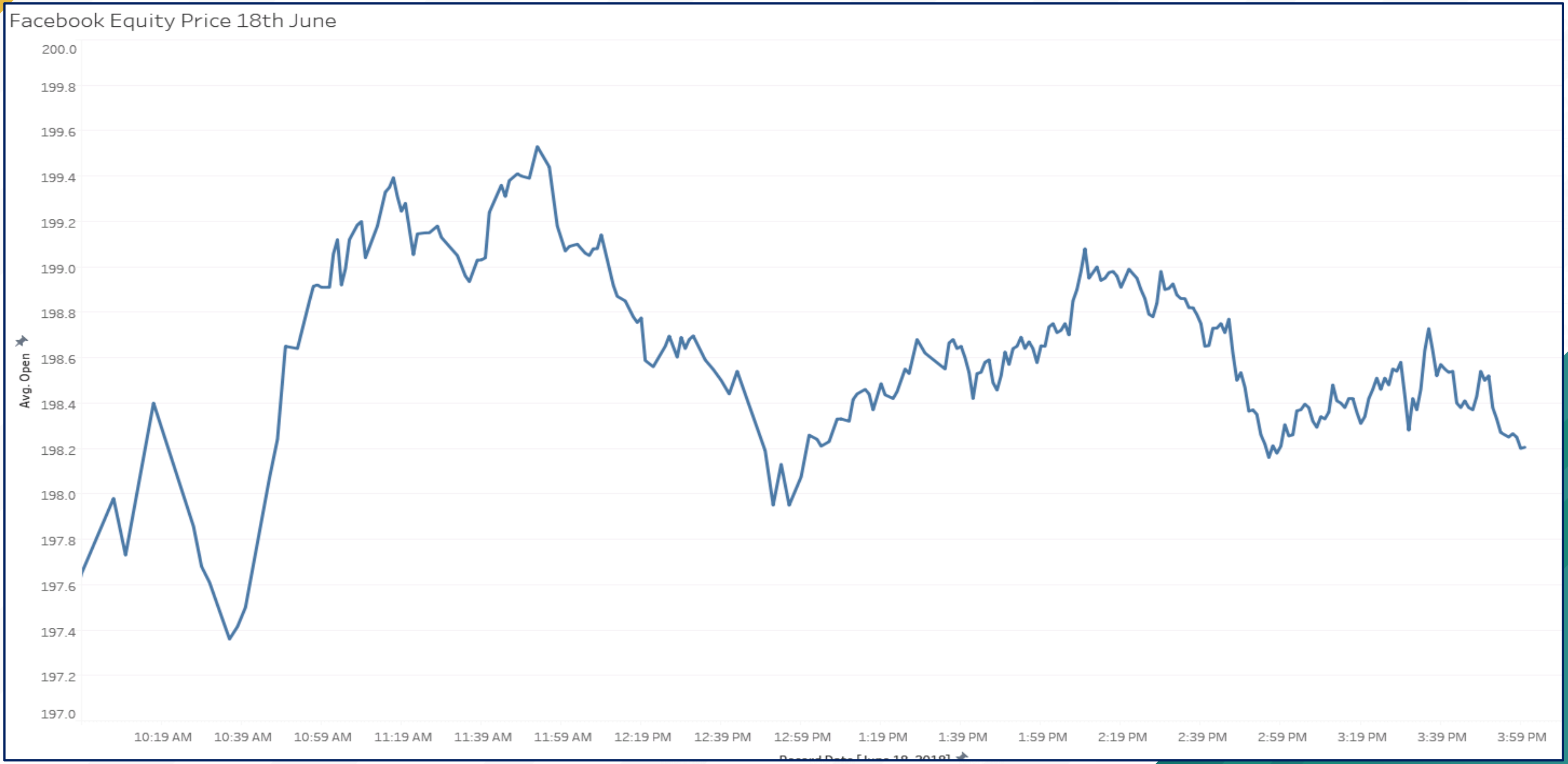
1 - Equity Data
2 - Timeseries Data
3 - Predictions Data

Time for Classification: 13 seconds.



```
+------------------+-----------+---------------------------------------+-------------+-------------------+-------------------+
|_KEY              |EQUITY_NAME|INVEST_STATUS                          |TIMESTAMP    |PREDICTED_VALUE    |RECORD_DATE        |
+------------------+-----------+---------------------------------------+-------------+-------------------+-------------------+
|AADV_1529347326168|AADV       |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347326168|74.0               |2018-06-18 18:42:06|
|AADV_1529347416711|AADV       |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347416711|74.0               |2018-06-18 18:43:36|
|AADV_1529347509407|AADV       |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347509407|74.0               |2018-06-18 18:45:09|
|AADV_1529347536595|AADV       |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347536595|74.0               |2018-06-18 18:45:36|
|AAEV_1529347326168|AAEV       |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347326168|91.0               |2018-06-18 18:42:06|
|AAEV_1529347416711|AAEV       |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347416711|91.0               |2018-06-18 18:43:36|
|AAEV_1529347509407|AAEV       |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347509407|91.0               |2018-06-18 18:45:09|
|AAEV_1529347536595|AAEV       |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347536595|91.0               |2018-06-18 18:45:36|
|AAL_1529347326168 |AAL        |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347326168|42.849955107973116 |2018-06-18 18:42:06|
|AAL_1529347416711 |AAL        |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347416711|42.849955107973116 |2018-06-18 18:43:36|
|AAL_1529347509407 |AAL        |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347509407|42.849955107973116 |2018-06-18 18:45:09|
|AAL_1529347536595 |AAL        |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347536595|42.849955107973116 |2018-06-18 18:45:36|
|AAU_1529347326168 |AAU        |DO NOT INVEST IN THE NEXT FIVE MINUTES!|1529347326168|0.7388982869062206 |2018-06-18 18:42:06|
```

Tableau
Visualizing the data



Facebook Equity Price 18th June

Tableau
Visualizing the data

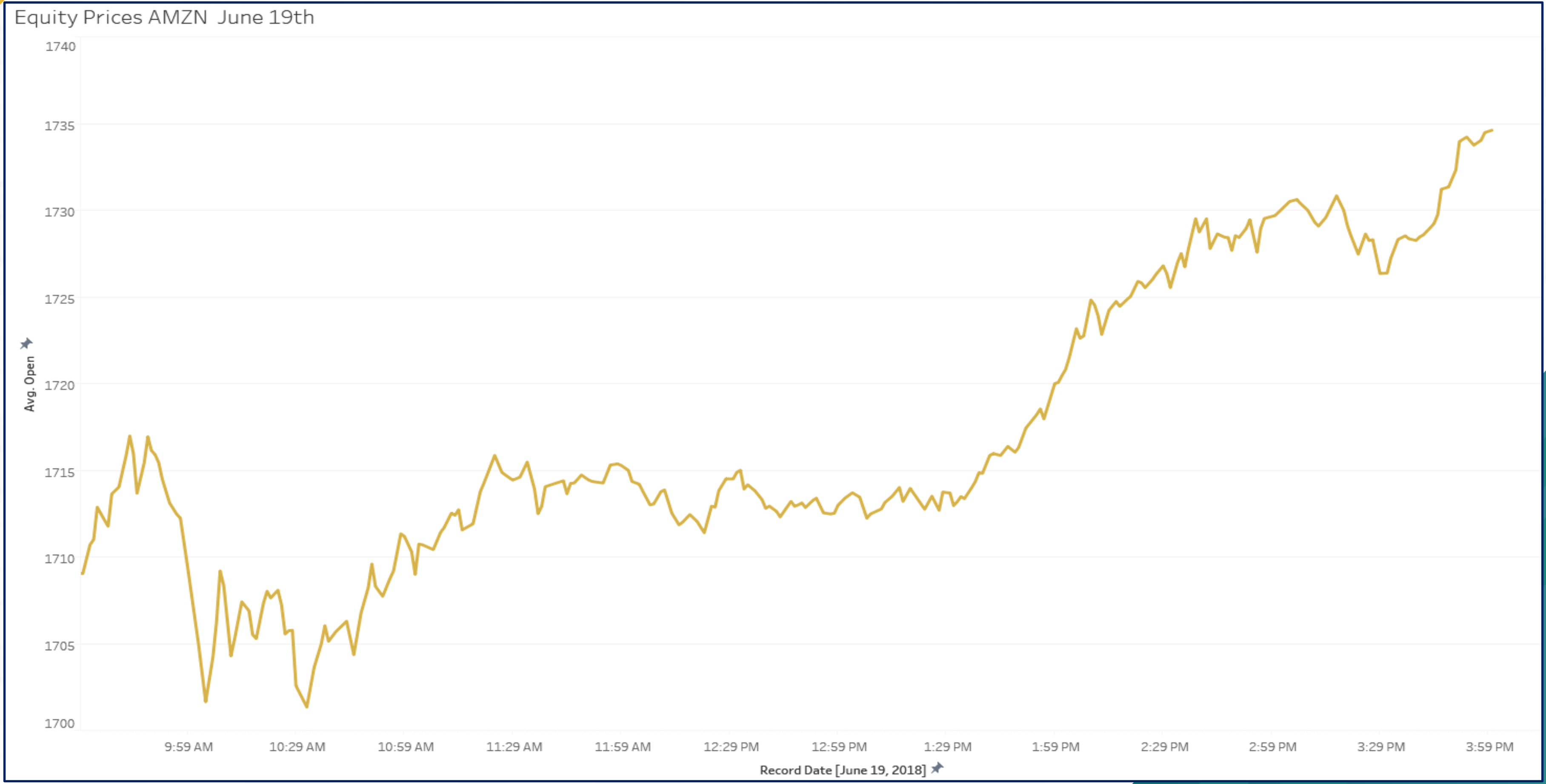
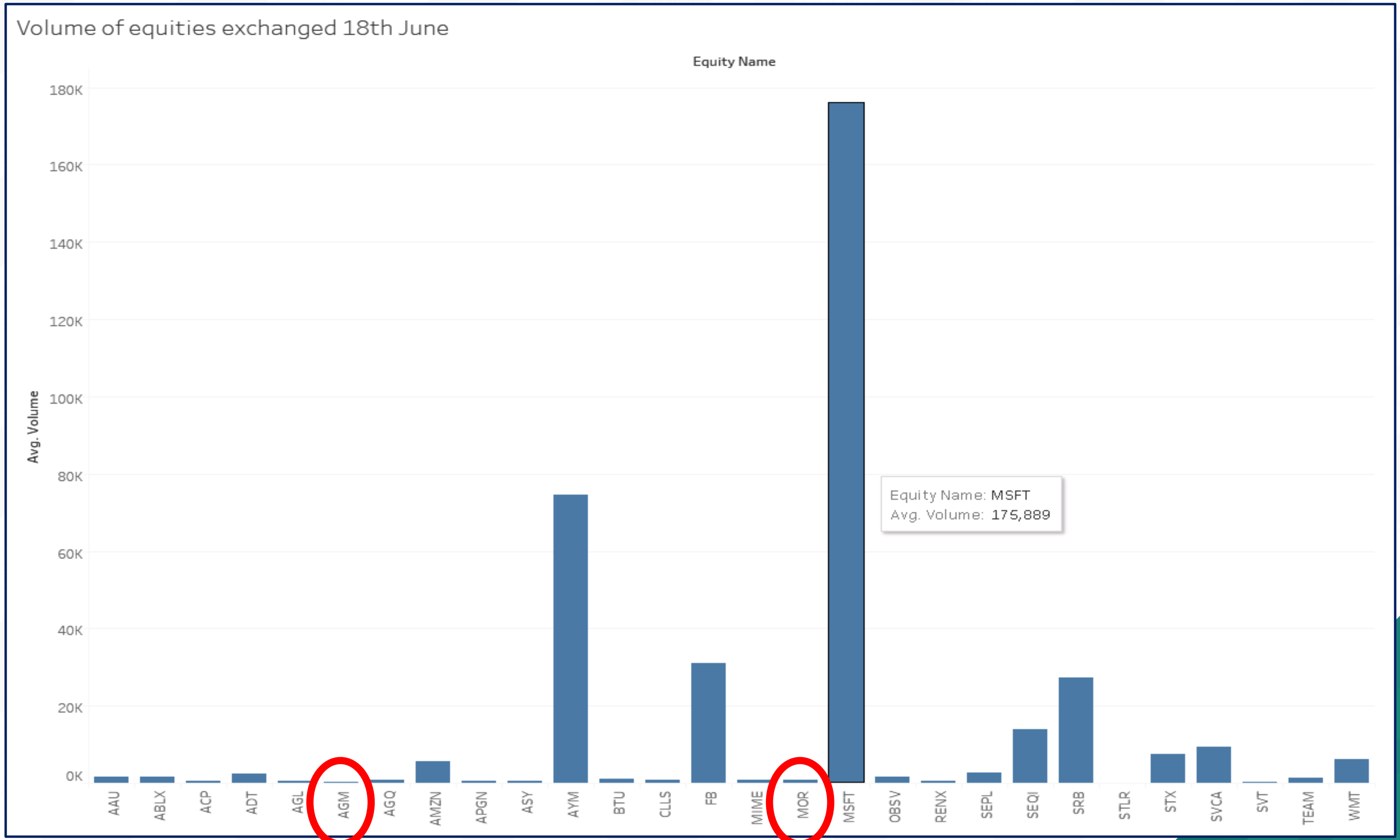


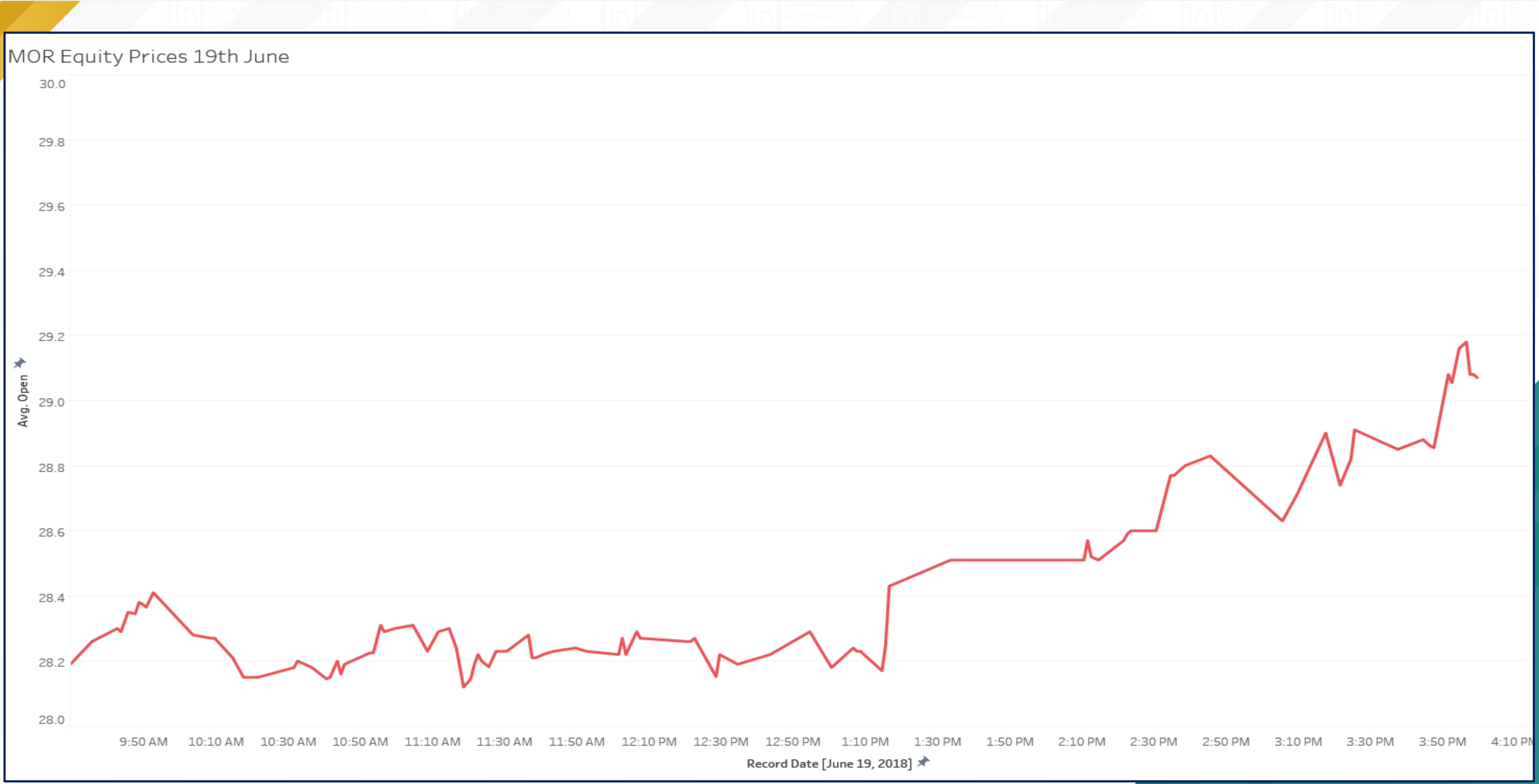
Tableau
Visualizing the data



Volume of equities exchanged 18th June

Equity Name: MSFT
Avg. Volume: 175,889

Tableau
Visualizing the data



In-Memory Computing SUMMIT EUROPE 2018

Tableau
Visualizing the data

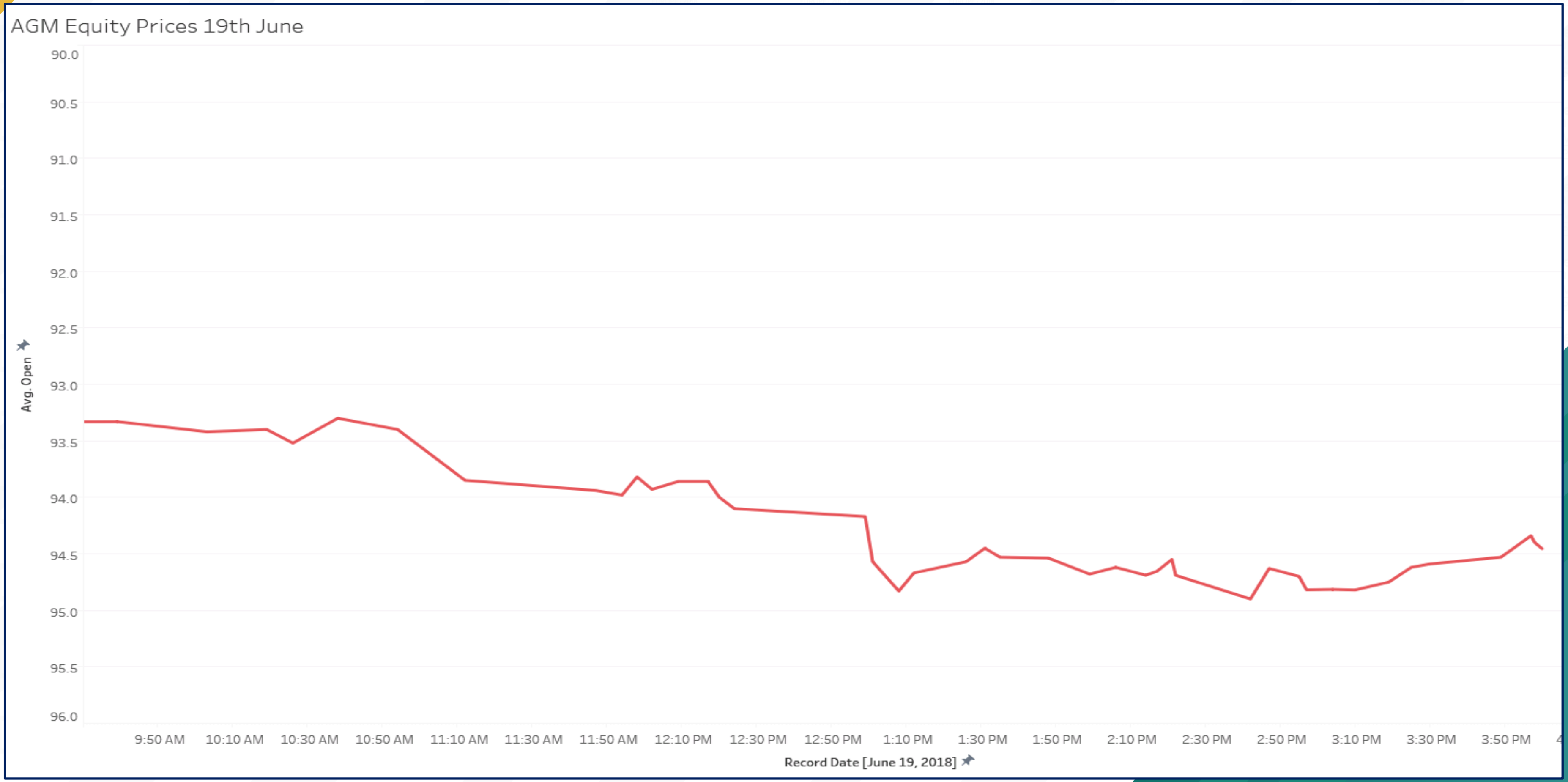


Tableau
Visualizing the data

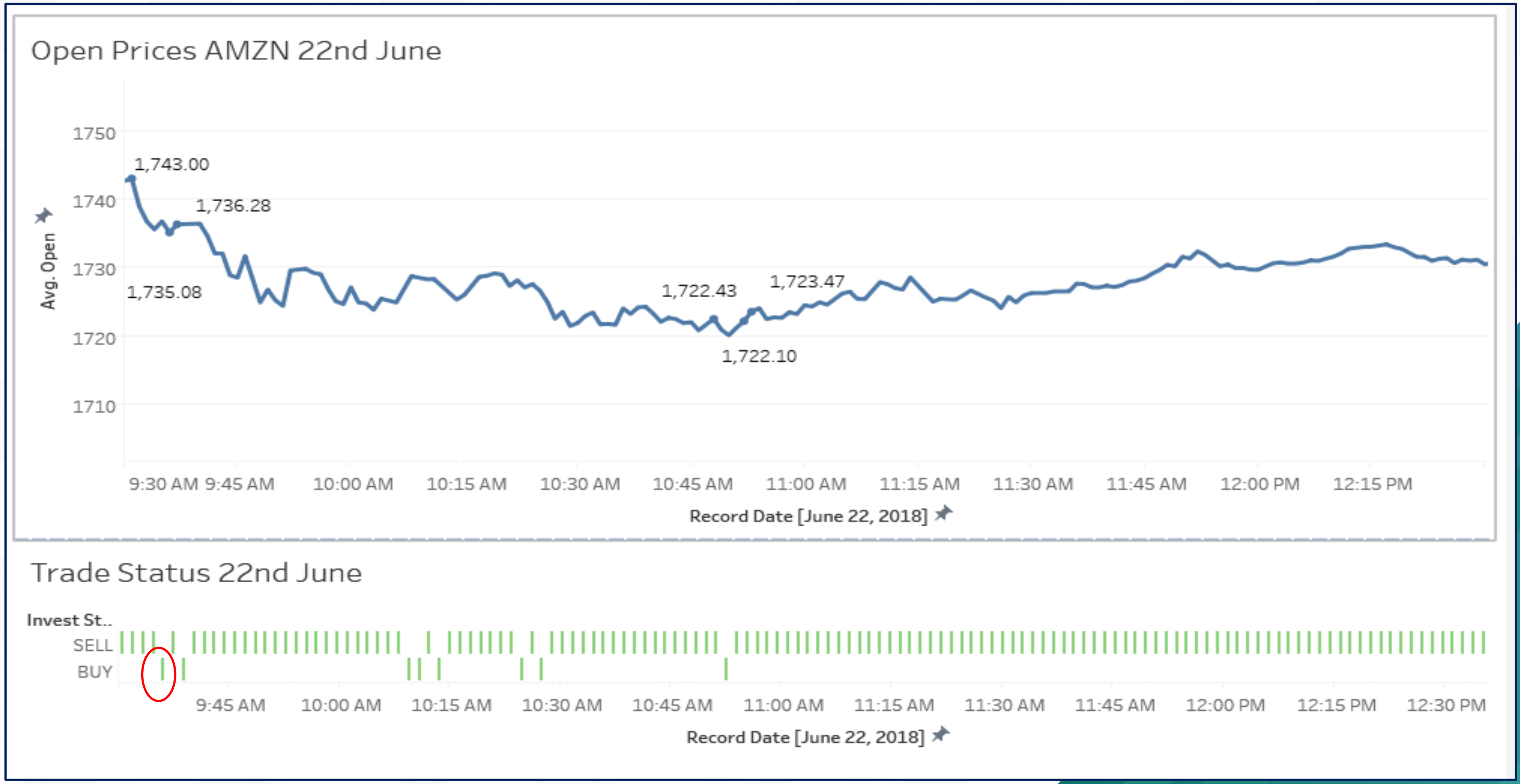
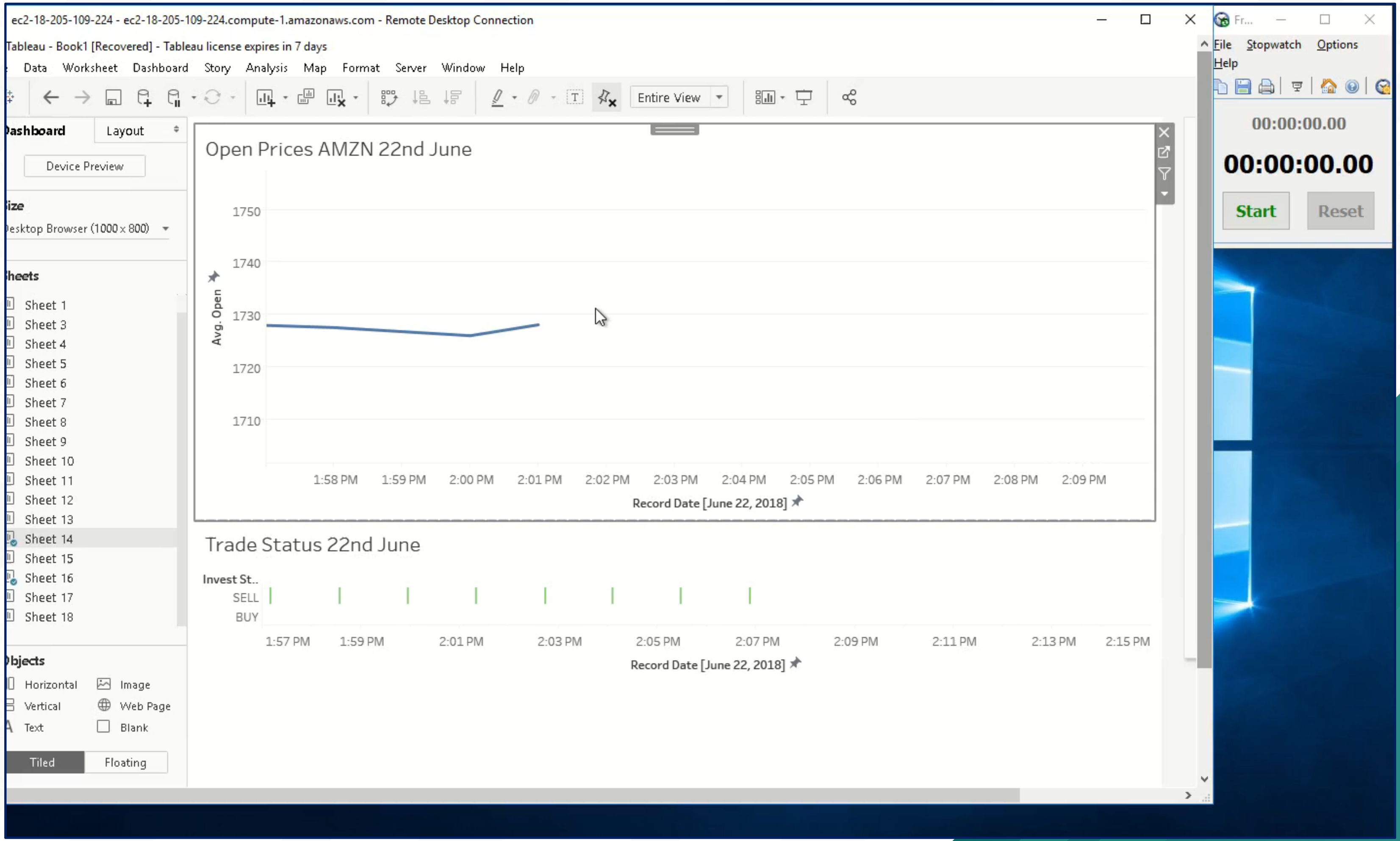


Tableau
Visualizing the data



Future Work

- Improving the ARIMA algorithm: as it stands, the proposed algorithm's predictions almost identically match the current price values.
- Increasing the number of data sources: currently only 100 different equities are being processed, do to source limitations. Increasing this number will truly validate this solution for Big Data scenarios.
- Update Ignite-Spark module, in order to support dataframe direct ingestion to a cache.
- Adapt Tableau to automatically refresh its graphs for real time feed.
- Implement an alert system for specific trading conditions.
- Explore more Spark and Ignite configurations to improve performance.
- Implement monitoring and security tools.

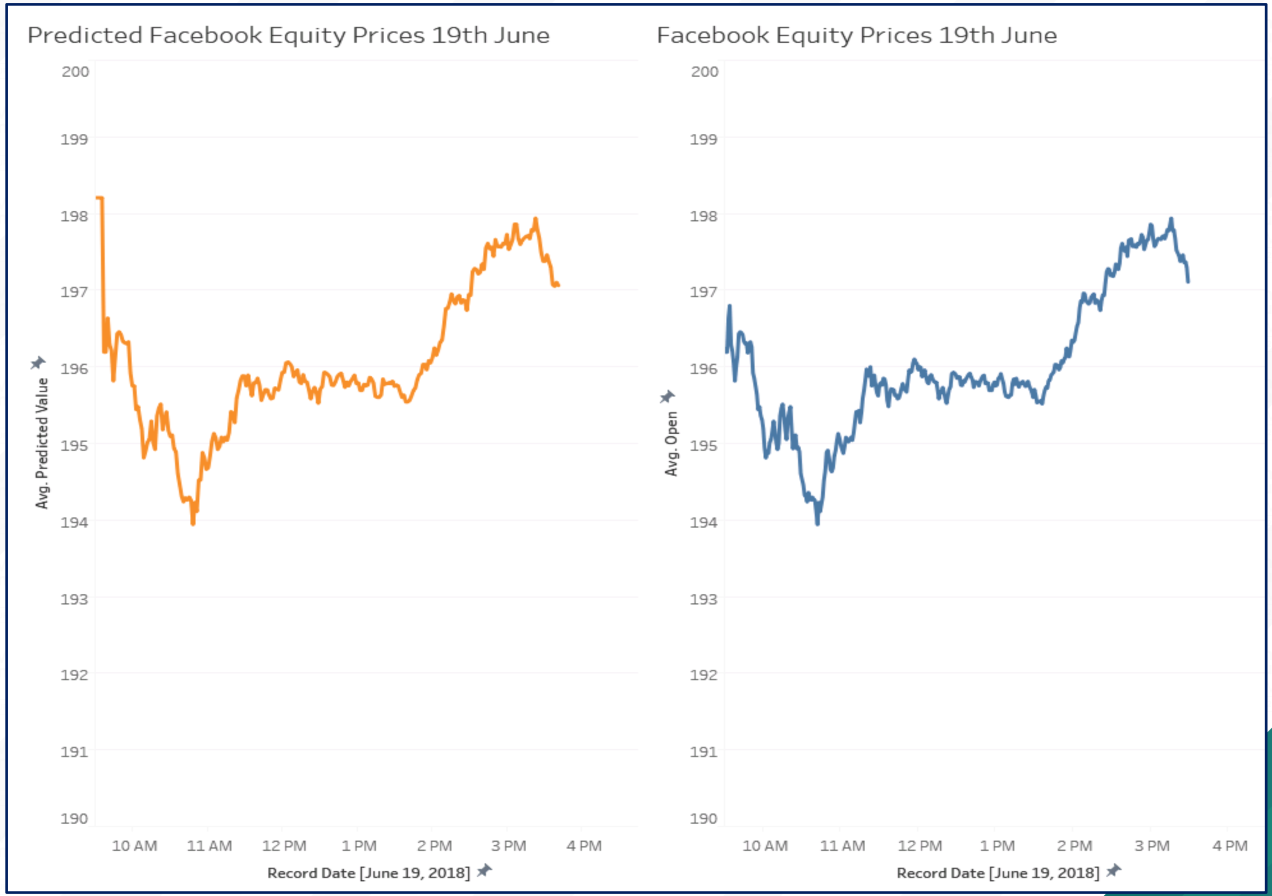
Questions

Thank You!

Q&A

?!

Annex



Predicted Facebook Equity Prices 19th June

Facebook Equity Prices 19th June