**OVERVIEW**

Design a system to process and enrich the heavy load of events coming from different systems at ~100k events/s (e.g. market data updates, prices and trades).

**Components used**:

- Kafka/Avro  
- Schema Registry/ Kafka Connect  
- Spark Streaming with SQL  
- Hive and HBase

**Proposed Solution:**

- Kafka can be used as messaging system and the systems generating events can send structured messages to Kafka topics, if the different events have the same id to relate and if they need to be combined for further processing, all the system can send events to the same Kafka topic using same schema. Kafka can also be used as database for storing events when the event source systems does not support back-pressure. If the systems generating events does not use Kafka, we can either write custom connect plugins to export the events coming from different sources to Kafka, or write custom receivers to read the events in the streaming systems.  
- We can use Avro for messages, it is fast binary protocol, compact format, in combination with schema registry it is more efficient.  
- When writing events to the Kafka topic, all the events related to the same id can be sent to the same partition. Hence the events belonging to the same ID will be processed by the same worker of the streaming application.  
- The metadata/inventory data coming from different systems for enriching the events can be store in some data warehouse like hive.  
- Spark streaming can be used with the micro batch interval size of either 10/20 seconds, and use spark streaming Kafka client to process messages in Kafka topic.  
- Spark streaming with Kafka client can be used for processing the events on Kafka topic, the events of the same ID group by key(id, timestamp?) and can be combined, so the combined record will have key and value as other events columns(market data updates, prices and trades)  
- The structured events can be converted to a DataSet and use the DataSet API to do the transformations. We can also store the DataSet as temporary tables in spark session and perform transformations with HQL by enabling hive support.  
- Using spark SQL we can read the metadata/inventory data stored in hive and with it we can enrich the input data, this can be driven by an HQL.  
- During processing of every micro batch, it is possible to have multiple records with same id and different timestamp, we can use the DataSet API withColumn() method and we can fill the column values with the last not null value. And at the end filter the DataSet to keep the latest event for every ID with filled values and use that for the next micro batch processing. Storing the latest events per ID in memory and persisting should be easy and feasible as the number of IDs are limited to 100s/1000s.  
- In the next micro batch processing combine the previous micro batch dataset having latest events for all IDs, with the current batch data and do the fill operation, it is like incremental operation for every micro batch.   
- At the end of every micro batch post the results of all the events of all IDs to the Kafka topic. And the dependent systems can process these filled events.  
- The filled events for IDs can be stored in HBase, by using the custom Kafka HBase sink connector (developed with Phoenix/HBase client)?  
- And the real time events can be fetched from HBase and shown to the users based on selection, for this we can write an NBI application to service the latest filled events for ID/IDs.

**Pros:**

- Using Kafka helps to store the data when the source systems have no back-pressure.  
- Spark streaming with Kafka also guarantees exactly once semantics.  
- Spark streaming guarantees ordering between the micro batches.  
- Spark SQL with DataSet API has good support for doing transformations and running SQL queries.

**Cons:**

- High latency with spark streaming, based on the micro batch interval size.