Overall implementation approach:

I have used Apache storm framework for this problem. Reason behind using Apache Storm is that it is focused on stream processing or complex event processing. It implements a fault tolerant method for performing a computation multiple computations on an event as it flows into a system.

To improve it more for this problem there are also other possibilities like integration of Storm with Apache Kafka and furthermore with kafka we can use NIFI processor which is an enterprise integration and dataflow automation tool that allows a user to send, receive, and other options.

Questions:

How many times a particular product has been launched during this time period?

For this task I I took "device_id" as a product and other field "type". If type of any event is 'launch' then with its device_id I store them. For holding this data I used HashMap because time-complexity of hashmap is O(1) for get(), contain(),size(). Also we don't need synchronization in this so it was efficient.

How many first-time launches can you detect for this product?

For this task I was using the same device_id and stores in HashSet. It offers constant time performance for the basic operations (add, remove, contains and size) and the main reason using this is that it guarantee duplicate-free collection of elements, no duplication and we need first time launches.

Can you detect duplicate events? How?

In this task I used two Hashset's. Any new event (different) was added to first hashset, but if the same events comes again then it was being added to second hashset (by compairing to first hashset) which assures that this event was already in first hashset so it means it's being repeated. Field "event_id" was used because it was unique among all events.

Do you observe anything weird in timestamps?

Yes, there was some error in epoch time (timestamps). In some events "send_timestamp" was greater than "received_timestamp". Mean field "received_timestamp" time was before "send_timestamp".

Which device has longest 'activity time' max('first event - latest event')

In this task I used two fields "create_timestamp" & "timestamp". Timestamp field and received_timestamp field had same time that's why is used timestamp to find the time interval for each device. To hold this data I used hashmap. I ignored all the events in which time was not correct, just used the right time events. To find the longest activity time I used java 8 stream() functionality because of time-complexity. There was another option, treemap in which order is maintained but its time complexity is O(log n) or if I use any other function to find MAX, time-complexity would be O(n). So I used hashmap and then stream().max() functionality.

What kind of statistics do you think are interesting? How could you visualize this?

Interactive analysis is the best way to really figure out what is going on in a data set. One way to do this is to analyze the whole data set at once using tools like Hive, Hadoop, or Pig. But an easier, better, and

more cost effective approach is to use random sampling. The following fields are interesting to analyze, (type, timestamps, geo, tenant_id, device_id, event_id).

How would you store the data so that further processing and/or analysis would be easy?

Storage of data should be in way that it can handle very large amounts of data and keep scaling to keep up with growth, and that it can provide the input/output operations per second (IOPS) necessary to deliver data to analytics tools. So it would be something like NoSQL (like MongoDB), Cassandra…

If you should prepare a maintenance time for the processor, when would you do it in order to cause minimal impact to products?

I am not clear about this question but I think parallel computing technique would be very beneficial in this case. A problem is broken into discrete parts that can be solved concurrently. Many problems are so large and complex that it is impractical or impossible to solve them on a single computer, especially given limited computer memory. Now a lot of frameworks are available for this kind of tasks.