**Beyond Batch Processing: Towards Real-Time and Streaming Big Data**

­­­­The paper has broadly explained about the strengths and shortcomings of MapReduce and Hadoop, real-time processing solutions and stream processing solutions. Firstly, the paper gives a brief introduction of big data, MapReduce and Hadoop. The author also speaks about how Hadoop is not suitable for handling interactive jobs, real-time queries and stream data. Due to these shortcomings, other solutions have been proposed. The author primarily focuses on two of these categories: real-time processing and stream processing of big data. The two solutions of the real-time processing discussed in the paper are: in-memory computing and real-time queries over big data. The in-memory computing is based on using a distributed main memory system to store and process big data in real-time. There are few in-process solutions available like spark and GridGain. Additionally, a distributed pool of memory which caches the frequently used data would help improve efficiency a lot. Apache Spark and GridGain support the caching mechanism. On the other hand, there are some solutions available for the real-time queries as well like Dremel and Apache Drill. Among these, Dremel is the prominent one which uses a novel columnar storage format for nested structures and also scalable aggregation algorithms for computing query results in parallel. In the stream-processing sector, there are two popular frameworks: Storm, and S4. Both have their own programming model, strengths and weaknesses. The author explains about both the frameworks and their superiority to MapReduce-based systems for stream processing.

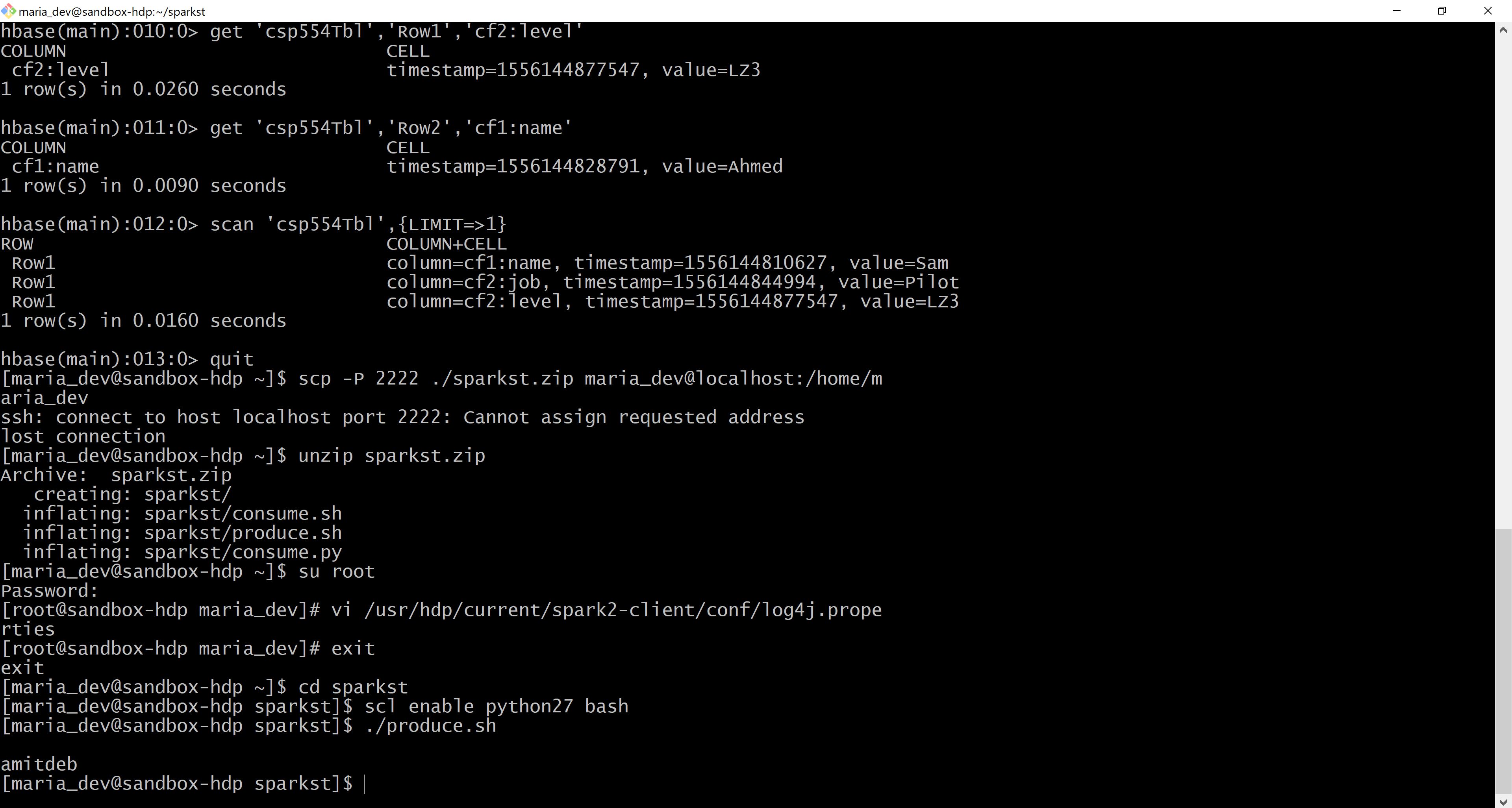
The author thinks that in-memory computing can handle both real-time and stream requirements very well and feels that Spark is the good example that justifies his opinion as it supports in-memory computing using RDD’s, real-time and interactive querying using Shark, and stream processing using fast micro-batching.

**Real-time stream processing for Big Data**

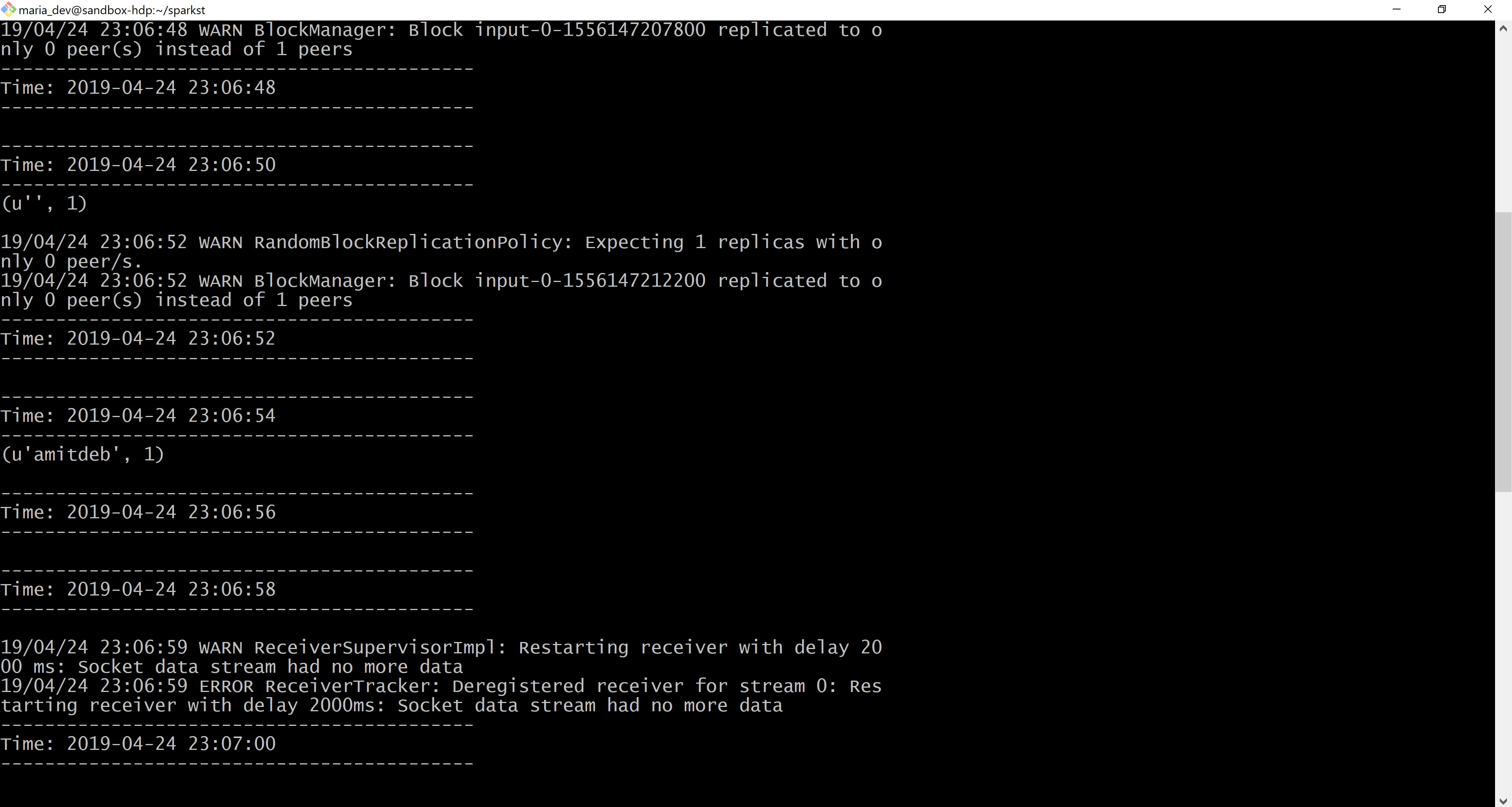
The article gives a detailed explanation about the state of the art of stream processors for low-latency Big Data analytics. This paper also consist of a qualitative comparison between Storm and its abstraction layer Trident, Samza and Spark Streaming. The author feels that batch-oriented systems cannot handle various types of data produced in real-world applications. The author specifies in the paper that stream-oriented applications are better than batch-oriented applications as they process data as it arrives. Handling data items immediately as they arrive would greatly decrease the latency at the cost of high per-item overhead. On the other hand, buffering and processing the data items in batches results in good efficiency. Some very good stream-oriented systems such as, Storm and Samza provide very low latency and relatively high per-item cost. Storm provides low latency, but does not offer ordering guarantees and is often deployed providing no delivery guarantees. Stateful exactly-once processing is available in Trident through idempotent state updates, but has notable impact on performance and even fault-tolerance in some failure scenarios. On the other hand, Samza focuses more on providing rich semantics. The author also discusses about the Kappa Architecture, where Samza and Kafka are tightly integrated and share messaging semantics which helps samza to use the ordering guarantees provided by kafka. There are many production deployments that implement lambda architecture and systems like Dataflow/Beam, Flink or Apex document, which have tried to take batch and stream-processing to the same level but have not enjoyed lot of success. While giving the conclusion ,the author strongly argues that horizontally scalable stream processors are gaining popularity because of their low latency level.

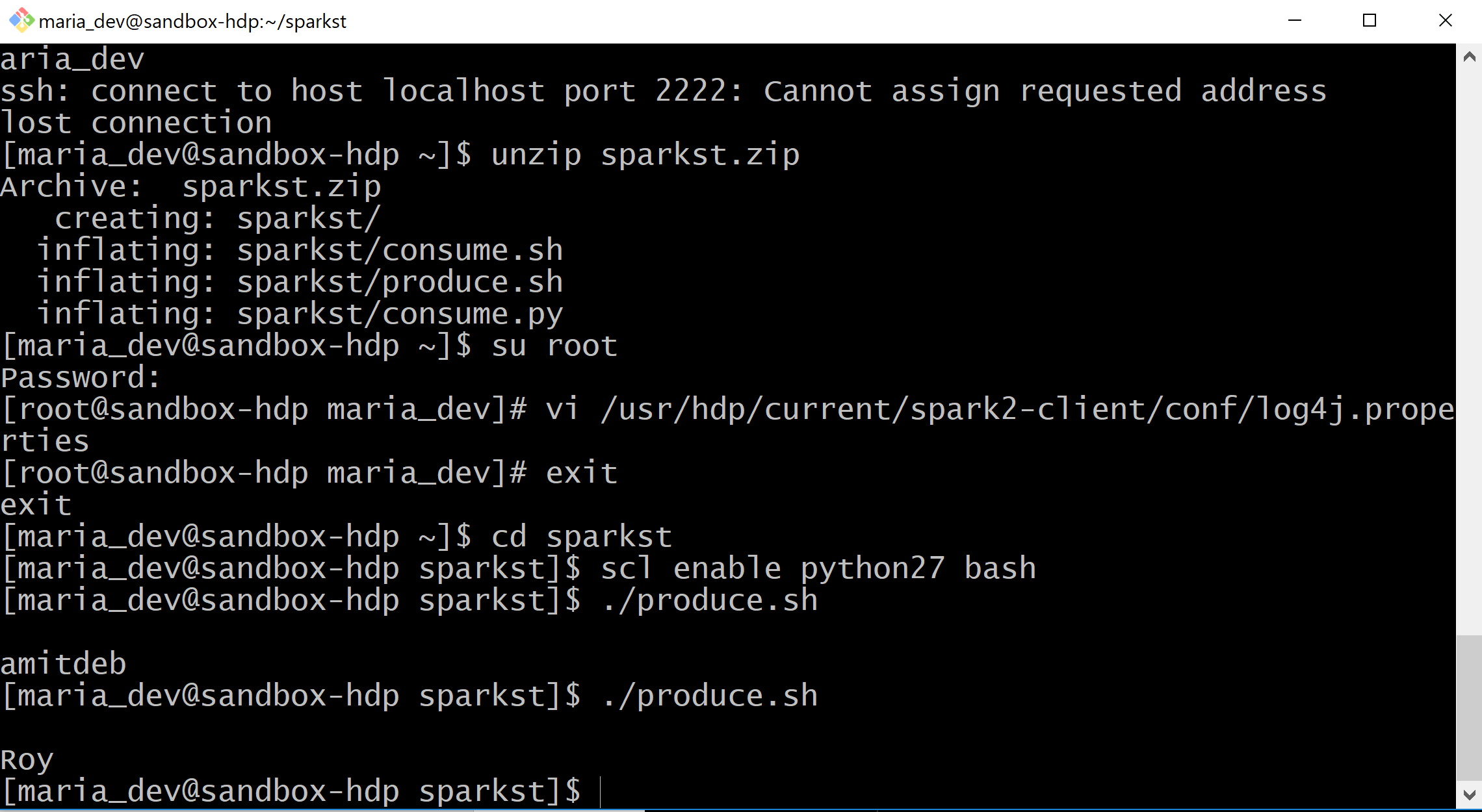
3) a)

i) Input: amitdeb

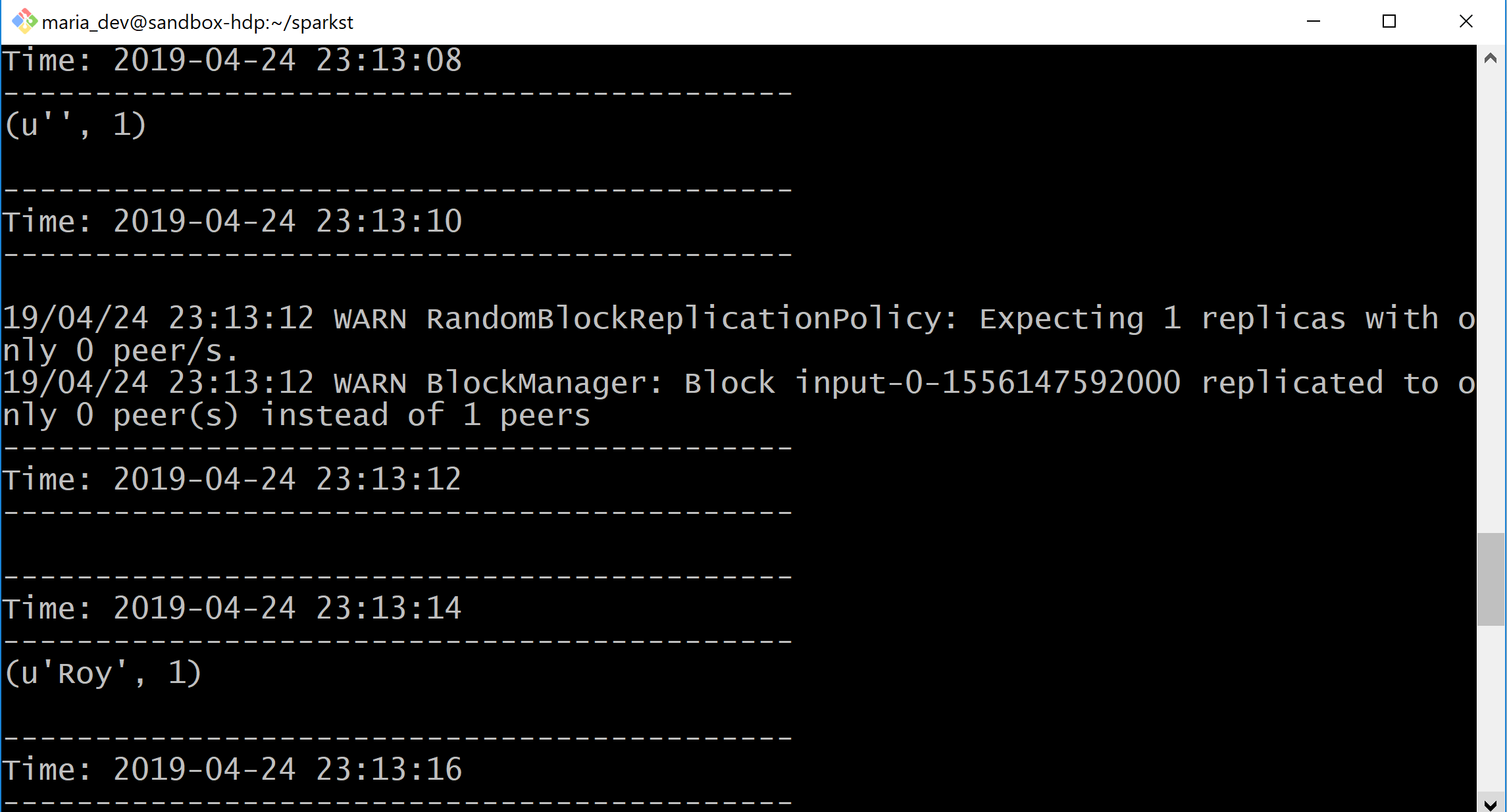


Output:

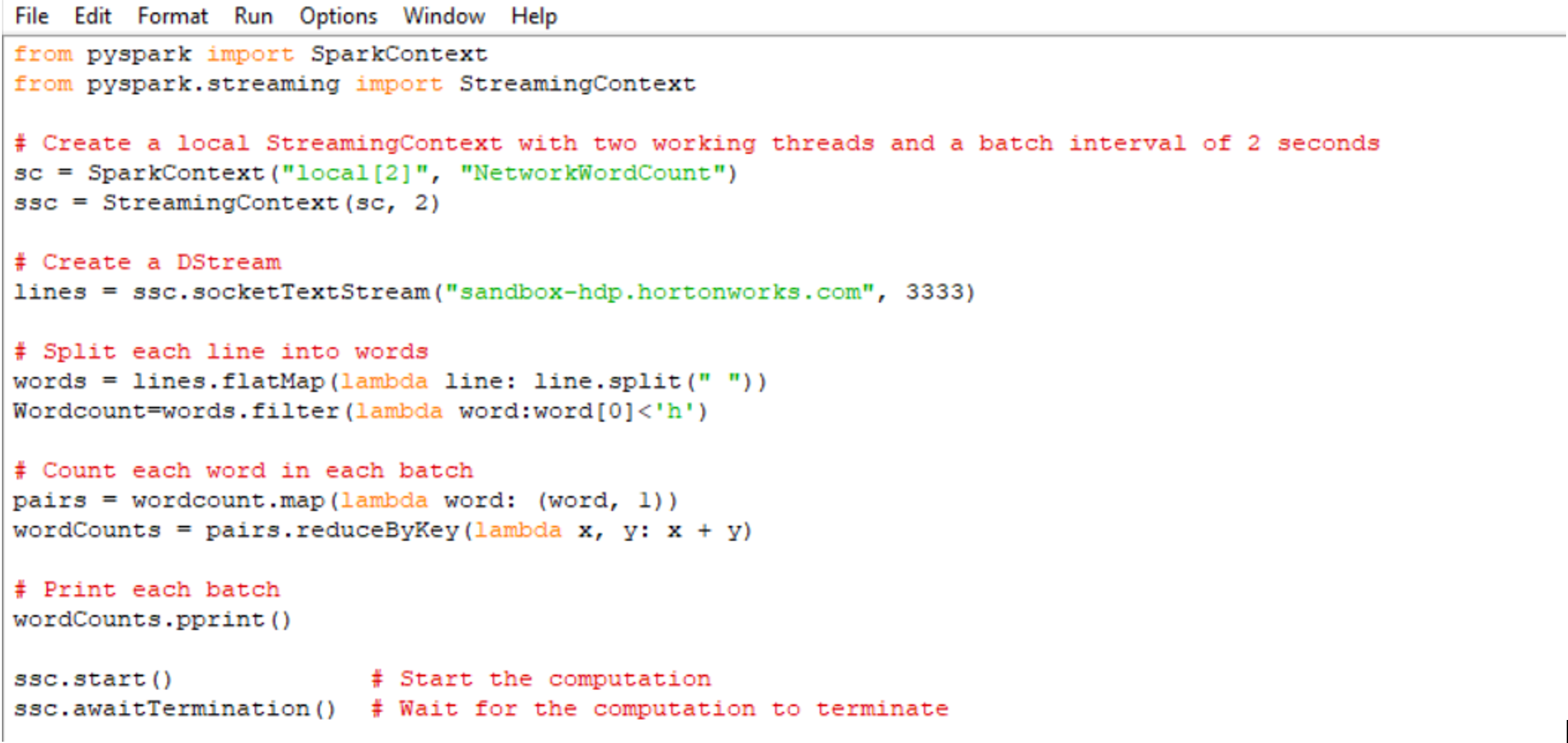


iI) Input: Roy 

Output:



3) b) Code for consume.py



1. Input: like apples

