
Xhafa, F. (2015) Processing and analytics of big data streams with Yahoo!S4. *IEEE 29th International Conference on Advanced Information Networking and Applications, Gwangju, South Korea, March 25-27, 2015: proceedings*. [S.l.]: IEEE, 2015. Pp. 263-270. Doi: http://dx.doi.org/10.1109/AINA.2015.194.

Xhafa, F. (2015) Processing and analytics of big data streams with Yahoo!S4. *IEEE 29th International Conference on Advanced Information Networking and Applications, Gwangju, South Korea, March 25-27, 2015: proceedings*. [S.l.]: IEEE, 2015. Pp. 263-270. Doi: http://dx.doi.org/10.1109/AINA.2015.194.

Processing and Analytics of Big Data Streams with Yahoo!S4

Fatos Xhafa, Victor Naranjo
Universitat Politècnica de Catalunya
Barcelona, Spain
EMail: {fatos,vnaranjo}@cs.upc.edu

Santi Caballé
Open University of Catalonia
Barcelona, Spain
EMail: scaballe@uoc.edu

***Abstract*—Many Internet-based applications generate huge data streams, which are known as Big Data Streams. Such applications comprise IoT-based monitoring systems, data analytics from monitoring online learning workspaces and MOOCs, global flight monitoring systems, etc. Differently from Big Data processing in which the data is available in databases, file systems, etc., before processing, in Big Data Streams the data stream is unbounded and it is to be processed as it becomes available. Besides the challenges of processing huge amount of data, the Big Data Stream processing adds further challenges of coping with scalability and high throughput to enable real time decision taking. While for Big Data processing the MapReduce framework has resulted successful, its batch mode processing shows limitations to process Big Data Streams. Therefore there have been proposed alternative frameworks such as Yahoo!S4, TwitterStorm, etc., to Big Data Stream processing. In this paper we implement and evaluate the Yahoo!S4 for Big Data Stream processing and exemplify through the Big Data Stream from global flight monitoring system.**

Keywords: Big Data, Stream Processing, Parallel Processing, Data Mining, Yahoo!S4, Scalability, Global Flight Monitoring System.

I. Introduction

Event-based systems are known for generating streams of data, which require fast processing of the incoming data. However, with the fast development of mobile, pervasive and sensor-based technologies, data streams have become a common pattern to many applications. Such data streams show new features when compared to traditional data streams. On the one hand, such data streams are unlimited in time (i.e. there is a continuous data stream generation) and on the other, the data size is huge. Thus, many Internet-based applications generate huge data streams, which are known as Big Data Streams. Such applications comprise IoT-based monitoring systems, data analytics from monitoring online learning workspaces and MOOCs (Massive Online Open Courses), global flight monitoring systems, etc.

- *IoT-based monitoring systems*: Current Internet architecture is able to support connectivity of hundreds of millions of IoT smart devices (Internet-of-Things). It is estimated that there are currently about 20.4 billion connected things worldwide, which is expected to grow to 29.7 billion by 2020. This number is at billions of connected devices, if included consumer devices like mobile phones, TVs, tablets, laptops, sensors, etc. In such systems, huge data streams are continuously generated. Consider for example, a wireless sensor network. It can be used for monitoring the environment, a logistics company, parkings in a city, wells of a petroleum company and a smart grid, to name a few. But this is in fact just one part of the IoT connectivity because the other part is the Internet of Everything (IoE), which includes not only devices but also people together with their devices and all the ways they connect and communicate (including machine-to-machine connections). The IoE based applications give rise to even bigger data streams, consider for example the Twitter data stream, requiring real time processing.
- *Monitoring online learning workspaces and MOOCs*: Online learning systems, differently from face-to-face learning, require continuous monitoring of learning workspaces so that the teachers and students can know at all times what's going on in the online workspaces. The monitoring is indeed crucial to shorten response time, to detect early dropouts, for scaffolding, etc. Even in its web-based implementation, the monitoring of online working spaces generates huge data streams (or when recorded in log files for batch processing). With the extensions of Virtual Campus to support mobile learning (through mobile phones and tablets), the size of data streams increases significantly due to the ability of connecting, working and being aware, anytime and anywhere (also known as A3 paradigm: Anytime, Anywhere, Awareness).
- *Global flight monitoring systems*: Global flight monitoring systems enable access to real-time Worldwide Flight Data of thousands flights tracked simultaneously. For instance, FlightRadar[1] is a flight tracking service that provides users with real-time information about thousands of aircrafts around the world.
- *Transaction/credit card fraud detection*: Business Analytics from transaction data are becoming a cornstone application to online banking. Recently, due to the

[1] http://www.flightradar24.com/

increase of mobile transactions, bancs are seeking ways to detect fraud in online transactions. This requires processing the data stream behind transactions and spotting in real time any fraudulent transaction and thus increasing security in online transactions.

- *Security threat detection and prediction*: With ever increasing sophisticated cyber-attacks, there is an imperative need to implement security threat detection and prediction systems. While traditionally this has been done by off-line processing of log data, the issue now is to process the log data in real time for security threat detection and prediction such as by detecting rare events, unexpected online user bahavior, etc.
- *Network fault prediction from sensor data*: With the Internet applications significantly growing in number of users, ensuring network stability and autonomic behavior is a real challenge to nowadays networks. The aim of autonomic computing is to detect failures in real time, their root cause and recover from such fault. Such features of autonomic computing require real time processing of data streams generated by "sensors" of the system.

In all these new applications, which as can be seen span across various domains, there is required real-time or near real-time response on Big Data Stream for faster decision making.

Batch processing, and therefore, MapReduce framework [4] is not suitable for those applications due to the need to process real time data stream on the fly to predict, for example, if a given transaction is a fraud, if the system is developing a fault, or if there is a security threat in the network or if a student interacting with a MOOC needs real-time scaffolding. The decisions need to be taken in real time, analytics are to be performed on short time windows in order to correlating and predicting events streams generated during that window time, otherwise the opportunity to address a solution is lost and might have no effect at a later time (as obtained by batch processing) [1], [2].

Therefore there have been proposed alternative frameworks such as Yahoo!S4 [8] , TwitterStorm [7], etc., to Big Data Stream processing. In this paper we implement and evaluate the Yahoo!S4 for Big Data Stream processing and exemplify through the Big Data Stream from global flight monitoring system.

The rest of the paper is organized as follows. In Section II we briefly overview some concepts and features of Big Data Streams. We discuss several frameworks from Big Data Stream Processing in Section III. Then, in Section IV, we present the implementation of the Big Data Stream Processing using Yahoo!S4 for global flight monitoring data streams as well as an experimental evaluation using real data from FlightRadar24 system. We end the paper in Section V with some conclusions and remarks for future work.

II. Big Data Streams and Challenges

There is an increasing interest in *online* data mining. Indeed, many Internet-based applications generate data streams that could be of interest to mine, especially for patterns discovery. For example, online banking applications might be interested to detect in real time failed transactions, or monitoring in real time flights information globally, mining streaming of tweets in Twitter, etc. What does really make the data stream "BIG DATA STREAM"? As in the case of Big Data, the answer comes from the definition of the main Vs:

1) ***Volume***: The amount of the data within the stream is large/very large for conventional computing systems.
2) ***Velocity/Rate***: The rate at which is generated the data stream is continuously growing. Indeed, data streams gather data from different sources, the amount of connected IoT devices and people is continuously growing resulting in a signinficant amount of data streams being generated.
3) ***Variety***: This feature refer to data heterogeneity, as there are heterogeneous data sources, data could come in structured/unstructured and can have various formats (text, multimedia,...). Data streams each time more combine data from different sources.
4) ***Veracity***: Data is checked against potential biases, noise, missing values, errors, etc.
5) ***Volatility*** defined as *data life time*: The data streams is to be processed as soon as it enters the system, however, the extracted information can be also persisted in databases, especially in NoSQL databases to cope with incoming volumes of data streams.
6) ***Value***: The value is defined in terms of data quality, that is, useful "knowledge" that can be extracted from data stream, how can it support business processes, fast decision making, etc.

There is a growing number of examples of Big Data Streams scenarios arising in every field of science and human activity. We briefly describe next a few of them.

Big Data Streams and Internet of Things: The Internet of Things is spreading out everywhere and thus Internet of Things Companies may be suppliers of Big Data Streams and analytical software that can help extract meaningful information from the enormous flows of data coming from many large scale applications (smart grids, smart cities, smart buidlings, "smart world"...)

Big Data Streams and Health Care: Big Data, in general and Big Data Streams more specifically are expected to make a revolution in healthcare, where data sets to be stored and analysed go far beyond traditional patient records. For example, there is a growing interest in continuous patient monitoring at hospitals, care centres or even at home, leading to Big Data Streams. In fact, remote patient monitoring is seen by large as potential way to reduce the burden to

premature or long time patient institutionalization (patients in hospital or care centres) due to caring of elderly, and growing population of patients that require long term care (e.g. patients of dementia, to be monitored 24x7 at home).

Big Data Streams, Virtual Campuses & MOOCs: Virtual Campuses are a widespread form of online learning and teaching. For instance, the Open University of Catalonia (Barcelona, Spain) accounts for more than 50.000 students and all academic and administrative activity is performed online! Log data files recording students activity can range from 15 to 20 Gb daily. It is of course of interest to know whats going on in the Campus (in virtual classrooms) in order to improve the online academic activity and learning outcomes as well as to improve the usage of resources, efficiency and security of the Virtual Campus, etc. Log data files are in this context considered as an important source of information. This has led to definition of learning analytics (also referred to as educational data mining). One recent example is that of MOOCs (Massive Open Online Courses), which unlike traditional online courses, can admit an "unlimited" number of students (there are some examples of courses exceeding 100.000 registrants). Learning analytics here can be useful to students monitoring, developing personalized MOOCs, designing personalized learning paths, etc. In such context it is desirable and useful to have a real time processing of the log data (as it is generated by online users' activity) due to the interaction of students with MOOCs system, so that the teachers and students can know at all times what's going on in the online workspaces. The monitoring is indeed crucial to shorten response time, to detect early dropouts, for scaffolding, etc.

Challenges of Processing Big Data Streams: The processing of Big Data is challenging *per se* due to the need of addressing the many Vs features. It becomes even more challenging in the case of Big Data Streams, especially due to the Velocity/Rate at which the data is generated. Assuming that the underlying infrastructure is able to collect the generated data into the data stream, conventional parallel processing systems, in most Big Data Streams, would not be able to cope with streams generated at high rates of data. Additionally, the challenges arise in mining Big Data Streams. Differently from mining large data sets already stored in databases, data warehouses or (distributed) file systems, online data mining has to cope with the incoming flow of data, requiring window-based sampling, chain sampling techniques to deal with the incoming flow of data. Considering short windows might imply loss of important correlated information, while large window time would cause serious memory issues.

III. Frameworks for Big Data Stream Processing

Batch processing, and therefore, MapReduce framework is not suitable for all scenarios arising in Big Data Streams due to the need to process real time data stream on the fly to predict, for example, if a given transaction is a fraud, if the system is developing a fault, or if there is a security threat in the network or if a student interacting with a MOOC needs real-time scaffolding. The decisions need to be taken in real time, analytics are to be performed on short time windows in order to correlating and predicting events streams generated during that window time, otherwise the opportunity to address a solution is lost and might have no effect at a later time (as obtained by batch processing).

Therefore there have been proposed alternative frameworks such as Yahoo!S4, TwitterStorm, etc., to Big Data Stream processing. We briefly describe them next.

Yahoo!S4: S4 is a general-purpose, distributed, scalable, fault-tolerant, pluggable platform that allows programmers to easily develop applications for processing continuous unbounded streams of data [8]. It is claimed that S4 fills the gap between complex proprietary systems and batch-oriented open source computing platforms. The aim is to develop a high performance computing platform that hides the complexity inherent in parallel processing system from the application programmer. The core platform is written in Java. The implementation is modular and pluggable, and S4 applications can be easily and dynamically combined for creating more sophisticated stream processing systems. S4 has been deployed in production systems at Yahoo! to process thousands of search queries per second. A first look and evaluation have been reported in [3], [6].

The main features of S4 are described as follows[2]:

- *Decentralized*: All nodes are symmetric with no centralized service and no single point of failure. This greatly simplifies deployments and cluster configuration changes.
- *Scalable*: Throughput increases linearly as additional nodes are added to the cluster. There is no predefined limit on the number of nodes that can be supported.
- *Extensible*: Applications can easily be written and deployed using a simple API. Building blocks of the platform (message queues and processors, serializer, checkpointing backend) can be replaced by custom implementations.
- *Cluster management*: S4 hides all cluster management tasks using a communication layer built on top of ZooKeeper, a distributed, open-source coordination service for distributed applications.
- *Fault-tolerance*: When a server in the cluster fails, a stand-by server is automatically activated to take over the tasks. Checkpointing and recovery minimize state loss.

Storm: Apache Storm is a free and open source distributed real time computation system. Storm makes it easy to reliably process unbounded streams of data, doing for

[2]http://incubator.apache.org/s4/

real time processing what Hadoop did for batch processing. Storm is simple, can be used with any programming language. Some evaluation of Storm can be found in [7].

A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed. Storm implements a set of characteristics that define it in terms of performance and reliability, fault tolerance and management.

The main features of Storm are described as follows[3]

- *Topologies*: Realtime computation on Storm is achieved by means of topologies (a topology is a graph of computation). Each node in a topology contains processing logic, and links between nodes indicate how data should be passed around between nodes.
- *Streams*: The core abstraction in Storm is the stream. A stream is an unbounded sequence of tuples. Storm provides the primitives for transforming a stream into a new stream in a distributed and reliable way. The basic primitives Storm provides for doing stream transformations are ”spouts” and ”bolts”. Spouts and bolts have interfaces that when implemented enable to run the application-specific logic (a spout is a source of streams, while a bolt consumes any number of input streams, does some processing, and possibly emits new streams).
- *Data model*: Storm uses tuples as its data model. Every node in a topology must declare the output fields for the tuples it emits.
- *Stream groupings*: Storm uses stream grouping to tell a topology how to send tuples between two components, while spouts and bolts execute in parallel as many tasks across the cluster.

Other frameworks for big data stream processing have been reported including Apache Kafka[4], Spark[5] and Apache Samza[6].

IV. Implementation of Yahoo!S4 for Big Data Stream Processing

A. The Actor Model

The Actor model [5], described by Carl Hewitt in the early 70s, is a model of concurrent computation for distributed systems. An ”actor” is defined as the smallest unit of concurrent computation. Each actor is an autonomous object that operates concurrently and asynchronously. Thus, in response to a received message, an actor can make local decisions, create more actors, send messages, and determine how to respond to the next message received. The Actor model extends the features of object-oriented paradigm by

[3]https://storm.apache.org/documentation/Tutorial.html
[4]http://kafka.apache.org/
[5]https://spark.apache.org/
[6]http://samza.incubator.apache.org/

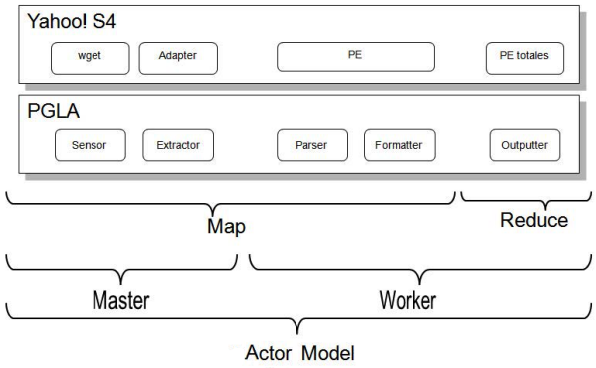

Figure 1. The Use of Actor Model for Implementing Yahoo!S4 (PGLA stands for Parallel Generic Log Adapter of IBM [9]; PE stands for Processing Elements in S4.)

providing the separation of control (where and when) from the logics of computation. One should pay special attention to this location transparency, and the characteristics of encapsulation and atomic execution of methods (defined as computational method response to a message).

The actor model can be implemented under different paradigms, for example, using a Master-Worker paradigm (see Fig. 1) under a Round-Robin scheduling algorithms (used in this work).

B. Experimental Evaluation

1) The input data set: To experimentally evaluate the Yahoo! S4 system, we have used a real source of data generated from real-time updating a large number of international flights from FlightRadar24[7]. Many commercial airliners that operate today, are equipped with a transponder, which is used by the control towers at airports to complement the information received by the radar. A worldwide network of radio receivers that use the ADS-B (Automatic Dependent Surveillance-Broad) technology to capture signals from these transponders (squawk codes) and send the information to a centralized system that updates and publishes online aircraft information received from the squawk codes issued by these transponders. The network consists of about 500 ADS-B receptors distributed throughout the world, which are visible only to aircraft within range of one of these receptors. At present, this coverage is as follows: 90% of Europe, some areas of the USA, Canada, Australia, Brazil, Middle East and Japan, along with other parts of the world. In Europe, 70% of commercial aircraft are equipped with these devices, while in North America, the number of aircraft that installed it is around 30%.

Through a http request[8], we can download a file from 200 to 300 Kb with real time information of all aircrafts identified, and thus generates a flow data every X seconds

[7]Flightradar24: http://www.flightradar24.com
[8]http://www.flightradar24.com/zones/full_all.json

```
"BEL10D":["4492EC",50.8275,5.2051,"301","8775","257","5775","N-EBBR1","RJ1H","OO-DWL",1350064704,"LIN","BRU","SN3150","0","-2048"]
"TRA5133":["4843D9",48.1883,2.6035,"177","41000","436","2114","T-LFOJ1","B737","PH-XRV",1350064704,"AMS","BCN","HV5133","0","0"],
"AM681":["4CA4B5",30.1297,-95.2837,"213","38000","432","0000","F-34TS1","B738","EI-DRC",1350064704,"YUL","MEX","AM681","0","0"],
```


Figure 2. A snippet of json data received for a flight.

(at Flightradar24 it is set to 10 seconds) provides the updated information of flights (between 4000 and 6000 aircrafts) on their geographic locations along with other information such as the identifier of the flight departure airport information, destination airport information, etc. The result of the process is dumped on screen or to a file, so it is easily possible to have a set of scripts that simulate the behavior of continuous queries, allowing a user accessing the console, filter the output of the process queries etc.

The information received for each flight contains flying code, latitude, length, angle to the north, elevation in feet, speed in knots, squawk code, identification of radar that caught the plane, model of the aircraft, plane register code, IATA airport code source, IATA airport code destination, flight secondary code, and other unused information. This information comes in JSON format. For example the following snippet (see Fig. 2, where we can identify the fields described previously).

2) Output results: As a result of processing incoming data stream of flight information, we obtain (either via console or persisted to a file or database) geo information of international flights covered by ADS-B network, supplemented with other information that is calculated at the time of receiving the data.

For every flight, we compute in real time:

- Departure Airport Name
- Destination Airport Name
- Country of Departure
- Destination Country
- Flight Status
- Landing Track Optimum
- Distance to origin and destination
- List of the n nearest airports to the aircraft radius, for a specified n.
- Country over which flies the plane
- Country closer to the current position

For the last four abovementioned calculations we must take into account the sphericity of the earth, so the orthodromic distance is calculated instead of Euclidean distance.

3) The HPC Infrastructure: We used RDLab as distributed infrastructure[9] (see Fig. 3) for processing and analyzing the data. The RDLab infrastructure aggregates hardware resources for research and project development:

- Over 160 physical servers.
- Over 1000 CPU cores and more than 3 TBytes of RAM memory.

[9]http://rdlab.lsi.upc.edu/index.php/en/

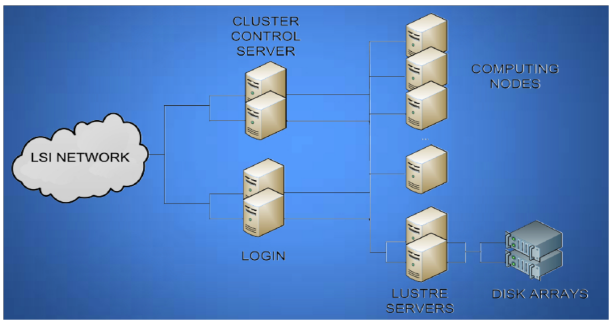

Figure 3. RDLab Cluster

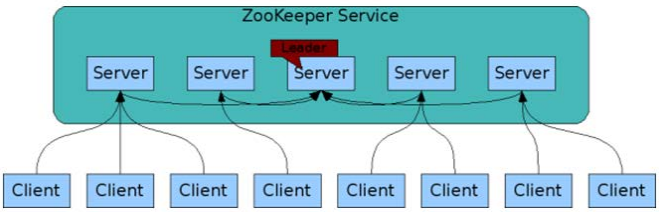

Figure 4. Zookeeper management services (source: ZooKeeper documentation).

- Over 130 TBytes of disk space.
- High speed network at 10Gbit.

The RDLab High Performance Cluster (HPC) offers several software packages such as Lustre High Performance Paralel filesystem, Hadoop support, SMP and MPI parallel computation, etc. We have used about 50-70 nodes in the Cluster managed by the ZooKeeper (see Fig. 4).

C. Parameter Configuration

In order to study the performance of the Yahoo!S4 implementation, the following parameters were considered:

- *Number of znodes*: the number of nodes in the cluster. In general the configuration of zclusters in Yahoo! S4 is managed by Zookeeper.
- *Chunk size*: The application is set to enter the chunk size as a parameter in input, and to show in output the corresponding reading and processing times for each of the chunks read and processed. Each reading request downloads small files of 200 to 300 Kb with information about approximately 6,000 flights. The reading rate is set to about 3 files per 2 seconds, approximately 10Mbps.
- *Waiting time* between consecutive readings: if necessary, due to lack of resources or to mitigate the limitations in computing capacity of slower nodes in

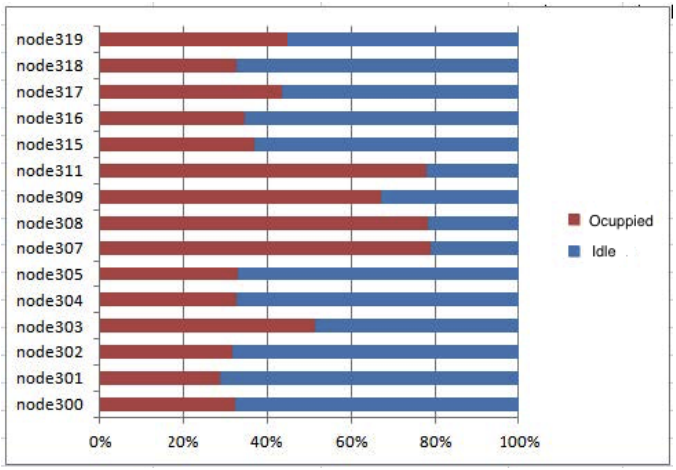

Figure 5. Busy *vs.* iddle time of the nodes in the cluster in proceesisng the incoming data stream.

the cluster, we can set up a short pause time for data reading process that generates the data stream. It should be noted here that while processing the data stream it is important to ensure that the output data stream (as a result of processing) is consistent with the incoming data stream in terms of order of events (flights in the case of flight monitoring system).

D. Computation Results

We measured the busy *vs.* iddle time for a sample of fifteen nodes in the cluster in proceesisng the incoming data stream (see Fig. 5).

As can be seen from Fig. 5, most of znodos are idle to start the next round (we remind here that we used the Round Robin scheduling strategy). Indeed, on average, during the overall processing time, 62% of the nodes are iddle. This suggests that there should be room to improve overall processing time by adjusting the parametrization of the processing elements (PEs). However, one must take into account that by adjusting the processing time per node so that a node can complete more PEs, it would jeopardize the ability to include in the znodes slower nodes. As a matter of fact, the znodes used by our Cluster are not homogenous in terms of their computing capacity. We could empirically observe that the processing time per chunk of the data stream could be four times larger than reading time of a chunk from the stream, as can be seen in Fig. 6.

This parameter adjusting was then crucial to get the best performance from the znodes of Yahoo!S4. As an example of the response time by processing the data stream, we computed the response time for some concrete flight. We selected a flight (at random) for which the response time along the processing of the data stream is about 3s in average (see Fig. 7).

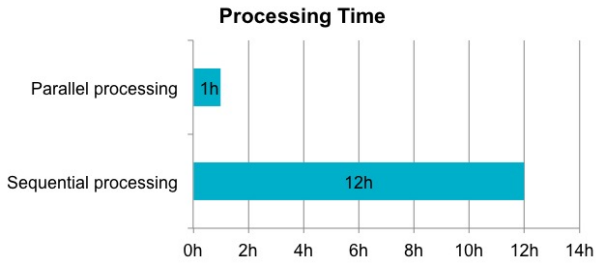

Figure 8. Sequential *vs.* parallel processing of the data stream.

Speedup of parallel processing: We observed that the parallel processing in the znodes of the cluster was 12 times faster than sequential processing (when using up to 70 znodes). That is, the Yahoo!S4 was able to process in 8 ms per flights (see Fig. 8 for a comparison of sequential *vs.* parallel processing of the data stream):

- 5.3 millons of flights
- 4.500 flights per file
- 1 file every 3 seconds

E. Issues from online mining of data streams

Differently from mining large data sets already stored in databases, data warehouses or (distributed) file systems, online data mining has to cope with the incoming flow of data.

The issues here however, as in the case of Big Data, are, on the one hand the *rate* at which data is generated and, on the other, the data variety (structured/unstructured, low/high volume, etc.) that can appear along the data stream. Window-based sampling, chain sampling techniques have been proposed to deal with the incoming flow of data. Memory issues arise if large window time are to be considered.

V. Conclusions

In this paper we have presented and evaluated the Yahoo!S4 architecture for real time processing of Big Data Streams. Yahoo!S4 is an alternative to batch mode processing –supported by MapReduce framework– for Big Data *Stream* processing in real time. We have implemented the Yahoo!S4 following the actor model for distributed computing and evaluated it with real data stream received from FlightRadar24 global flight monitoring system. The study was conducted in a Cluster environment using 50-70 znodes under zookeper servives and showed the efficiency of the implementation. We were able to achieve very fast processing time per flight (considering real data of about 6000 flights) from FlightRadar24. We also observed some issues with heteregoneity of computing znodes in the cluster that required adjusting the parameters to improve the busy vs iddle time of the znodes.

The output data from processing the data in our implementation was rather statistics as well as resummed tracking

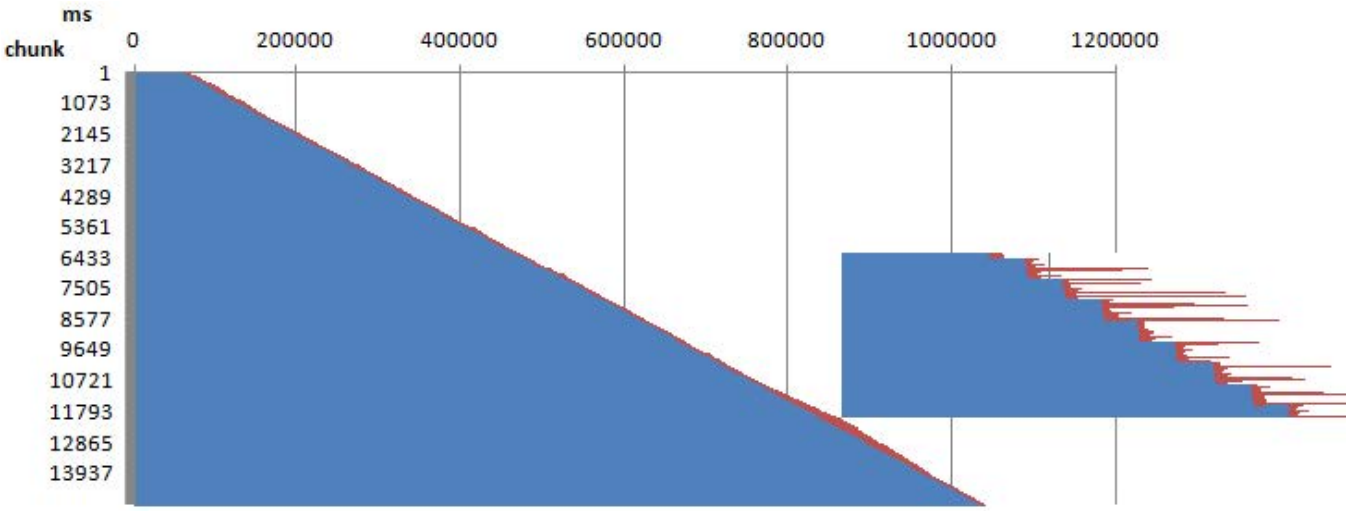

Figure 6. Reading *vs.* processing time of the znodes in the cluster. At some nodes the processing time could be four times larger than reading time.

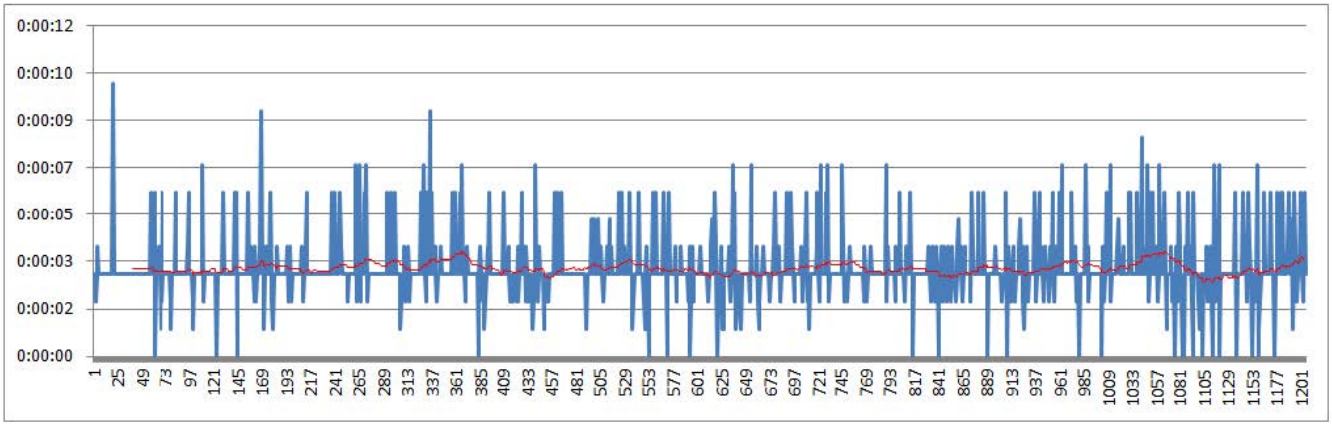

Figure 7. Response time for a flight along the processing of data stream.

data per flight. Obviuosly, it would be interesting to make more sophisticated computations (at Processing Elements) such as patterns of flight navigations, fuel consumption, alternative (secondary) routes, etc. However, challenges arise in mining data streams due to the high rate of data stream generated by online applications. Differently from mining large data sets already stored in databases, data warehouses or (distributed) file systems, mining data streams has to cope with the incoming flow of data, requiring window-based sampling, chain sampling techniques to deal with the incoming flow of data. Considering short windows might imply loss of important correlated information, while large window time would cause scalability and memory issues and even loss of information from the data stream. In our future work we plan to implement some mining functions as part of logics of PEs (Processing Elements) in Yahoo!S4 and evaluate the system under new scenarios.

ACKNOWLEGMENTS

This research has been supported by TIN2013-45303-P "ICT-FLAG" Enhancing ICT education through Formative assessment, Learning Analytics and Gamification and TIN2013-46181-C2-1-R "COMMAS" Computational Models and Methods for Massive Structured Data.

REFERENCES

[1] Caballé, S. and Xhafa, F.: Distributed-based massive processing of activity logs for efficient user modeling in a Virtual Campus. Cluster Computing 16(4): 829-844 (2013)

[2] Fernandez, R. C. and Pietzuch, P. Towards Low-Latency and In-Memory Large-Scale Data Processing. Doctoral Workshop of the 7th ACM International Conference on Distributed Event Based Systems (DEBS), 06/2013, Arlington, Texas, USA, (2013)

[3] Chauhan, J., Chowdhury, S.A. and Makaroff, D. Performance Evaluation of Yahoo! S4: A First Look. 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 12-14 Nov. 2012 Page(s): 58 - 65, Victoria, BC DOI: 10.1109/3PGCIC.2012.55, IEEE

[4] Dean, J. and Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. CACM, 51(1), 2008.

[5] Hewitt, C., Bishop, P., and Steiger, R. A Universal Modular Actor Formalism for Artificial Intelligence. IJCAI 235-245, 1973.

[6] Hoeksema, J., Kotoulas, S. High-performance Distributed Stream Reasoning using S4. In Ordring Workshop at ISWC, 2011.

[7] Mishne, G., Dalton, J. , Li, Z., Sharma, A. and Lin, J. Fast data in the era of big data: Twitter's real-time related query suggestion architecture. Proc. ACM SIGMOD Int. Conf. on Management of Data, pages 1147-1158, 2013.

[8] Neumeyer, L., Robbins, B., Nair, A., and A. Kesari. S4: Distributed Stream Computing Platform. Proc. 2010 IEEE International Conference on Data Mining Workshops, Page(s) 170-177, 2010.

[9] Xhafa, F., Claudi Paniagua, C., Barolli, L., and Caballé, S.: Using Grid services to parallelize IBM's Generic Log Adapter. Journal of Systems and Software 84(1): 55-62 (2011)