PROCEEDINGS OF SCIENCE

The City Fire Control Network System based on Storm for Real-time Data Processing Application

Susu Yang[1]
Institute of Information Engineering , Capital Normal University
Beijing, 100048, China
E-mail: `yangsusu112233@sina.cn`

Lijuan Zhou[2]
Institute of Information Engineering , Capital Normal University
Beijing, 100048, China
E-mail: `zlj87@139.com`

In the city fire control network system, the data of real-time information has become larger and larger. The traditional fire control system cannot deal with the problem of real time data with high efficiency. We analyse the requirements and performance of the open source Storm framework, and implement the improvement on the technical architecture and the characteristics of the fire system. We put forward a set of system architecture for real-time processing of data in high real-time and high scalability of the fire control network monitoring center. At the same time, the construction of the cloud computing platform, using the heartbeat detection mechanism to ensure the real-time performance of the monitoring unit. The research shows that, based on cloud computing and Storm platform, the architecture is fully applicable to the fire control system, which has the characteristics of high efficiency, high reliability, high performance and so on

ISCC 2015
18-19, December, 2015
Guangzhou, China

[1]Speaker

© *Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).*

PoS(ISCC2015)068

1. Introduction

In recent years, a variety of fire accident prone, there is traffic fire accident, production and life fire accident, industrial production fire accident. More than 80% of these fire accidents caused a very heavy loss. According to the analysis of the survey found that many fire accidents can be found in advance, and timely treatment and evacuation to reduce the loss. So it is very important to carry out early warning analysis and evaluation of fire risk. In order to solve the fire alarm, equipment fault timely detection and timely inspection, improve the city's fire disaster reduction ability, each city is trying to build a fire network platform. In particular, the data collection of fire network sensors for fire prevention and analysis is becoming increasingly important. The processing of large amounts of data from millions of devices in various regions is becoming more and more urgent. So the problem of real time processing of big data is becoming more and more prominent. But the existing fire alarm system is mainly to study the fire data collection, as well as the use of more advanced transmission equipment and more reliable transmission protocol for data transmission. However, we not only need the data real-time reliable transmission, we also need to analyze and forecast the data of sensors in real time and real-time monitoring, in order to achieve the fire timely warning, timely command, fire equipment failure timely detection and repair. Existing large data processing methods are divided into two types: stream processing and batch processing[1, 2] . The data batch processing is mainly aimed at the static data and some slow change data sets. For high real-time requirements system, especially to cope with the growing data, such as the abnormal data which detected by fire detection equipments and the status of the equipments. We need to pay more attention to real-time data stream processing. Today, the well-known stream processing systems are Kafka[3], Storm, S4[4] (scalable streaming system simple), Data Freeway and Puma[5], MillWheel[6], Spark Streaming and Photon, etc. Stream Data processing technology has also been integrated as part of traditional database product pipelines(e.g.[7,8]). In this paper, the system design is based on the cloud computing and Storm real-time data processing system, and the overall performance of the system is tested.

2. Cloud Computing Platform for City Fire Control Network System

With the rapid development of various sensors and electronic devices, human society has entered the era of big data[9] and the collection of various types of large data has become a reality, which also provides a variety of hardware and technical support for the collection and transmission of large amounts of fire detection data. City fire remote monitoring system is the automatic fire alarm system which is based on modern communication network, and integrated use of information technology, such as geographic information system, digital video monitoring and other information technology. At the same time, the national standard requirements: the cities (including prefecture level, the national total of 286), should set up a remote city fire monitoring system; At present ,in the face of a large number of monitoring units, the monitoring data is increasing, the fire center needs to collect and process massive data effectively and timely.

2.1 Limites of the Current Fire Control Network System

At present, China is still in the stage of the construction of the fire remote monitoring system, which is not yet mature. Monitoring unit store the data in the database, the fire control center query every the database of monitoring unit, and then process the data. Finally the information will be displayed on the client page. This approach has many drawbacks:

- Low efficiency: Most of the current city fire control network system, the monitor-ing unit will store their data in the database, the fire monitoring center frequently query the database of every monitoring unit to get the data, and then the data will be processed, and finally displayed on the client page. With the increase of the

PoS(ISC2015)068

amount of data in the monitoring units, the efficiency of the database retrieval will be reduced, resulting in low efficiency of the system.

- Bad real time: Due to the low efficiency of the current city fire network system, the time of the fire center getting the real-time data cannot be sure. In general, there will be at least a few seconds delay. A few seconds for fire warning, impacting the lives of thousands of people. So the real time of the data is the biggest problem to be solved for the fire monitoring center.
- Maintenance difficulty: In current city fire alarm network system, the fire control center continuous query the database of monitoring units, thus make each node server facing great pressure, easily lead server to crash. At the same time, the fault tolerance capability of the system is also poor, and it is difficult to maintain the system. Therefore, high real-time, high efficiency and high performance cloud computing cluster structure is the best way to solve the above problems. Building fire networking platform is also an urgent demand.

PoS(ISCC2015)068

2.2 Construction of Cloud Computing Platform for City Fire Control Network System

The system framework is shown in Figure 1. According to some actual conditions of the fire industry which has no direct data connection between the monitoring unit. Each monitoring unit directly connects to the Fire Center.

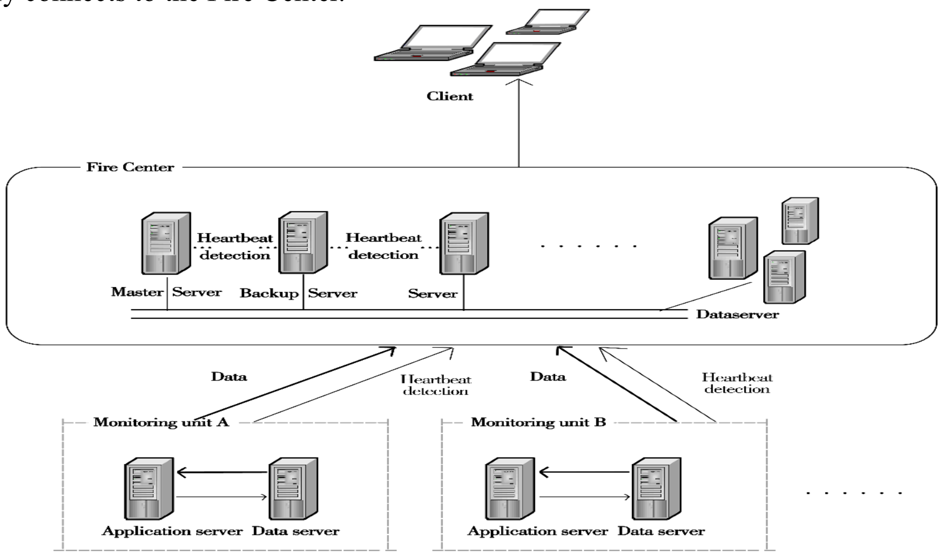

Figure 1: System Architecture Diagram

The fire center uses a heartbeat detection mechanism to protect the real-time monitoring of the monitoring unit, the unit send a status information to the fire center every few seconds. If the state information is not normal, then the fire center will notify the corresponding monitoring unit for timely maintenance, this mechanism also ensures a reliable connection between the fire center and monitoring unit. The fire center get the data timely and the server cluster analyze and calculate the data. Finally the data will be shown in the client page and stored in database.

2.3 High Real Time Performance of the Cloud Computing Platform for City Fire Control Network System

The fire control system which needs real-time monitoring the state information of each monitoring unit, and it needs to obtain the latest data from the monitoring unit. Therefore, the speed of data acquisition directly affect the real-time of the system. So the monitoring unit will store the data in database and at the same time in specific log file. The fire center continues to read the new data of monitoring unit log files. The old log file will be cleaned to control the size of the file effectively. At the same time, the fire center just reads the new data of the log file. Compared with the traditional way, the speed of the data will be ensured. Also, the cloud computing platform distribute computing tasks according to the load of each machine, it also ensures the efficiency of processing.

2.4 High Reliability of the Cloud Computing Platform for City Fire Control Network System

Fire control network system is a real-time service system, which needs to constantly calculate the data obtained from monitoring units. If fire network system has problems or outage, fire data are accumulated, and are not real-time feedback of fire data, this is unacceptable. Cloud computing platform is obtained by using the backup copy of the fault-tolerant, heartbeat detection and isomorphism interchangeable and other measures to ensure high reliability of service.

Server cluster in fire control center controlled by a master server, the cluster server will send a message to the master server at set intervals, once the server in the cluster failed, the master server can’t receive the heartbeat information of the server, the work of the server will assign to other server without delay. And the master server also has a standby server, once the master server failed, the standby will be put into use immediately.

2.5 The High Scalability of Cloud Computing Platform for City Fire Control Network System

The identification information of the monitoring unit in the system is stored in the master server of the fire center. At the same time, if the average load rate of the servers in the cluster is too high (more than 80%), it can be easily added to the cluster of new servers. In this paper, the system platform has been in the experiment, which provides a stable, efficient, reliable and convenient platform for the city fire control system.

3. Real-time Processing of Data in the Fire Monitoring Center System

In this paper, a real-time monitoring system based on flume, kafka and strom cluster is proposed for the real-time acquisition and processing of the mass data in the fire control system. Using the module design, each module completes its own specific functions. This chapter mainly introduces the components of the fire monitoring system, the design of the system structure, and the results of the system client.

3.1 Components of the System

Flume is a highly available, highly reliable, distributed mass log collection, aggregation and transmission system. This system is based on flume_agent, which is distributed in the monitoring unit. Kafka is a high throughput distributed publish subscribe message system. Because it is required to establish a buffer between the flume data and the storm processing data, the kafka accepts the data collected by flume, and the data is transferred to the storm in the form of message queue. Storm is an open source, big data processing system, which is designed for distributed real-time processing and has nothing to do with language. The use of storm is the core of the topology structure design, we deliver the message data in kafka to storm, and then according to the topology of the fire center server cluster to process the data, so the number of machines is the key to the whole system. Storm runs on a distributed cluster architecture. The storm cluster consists of a master node and multiple working nodes. The stream processing data can be stored in the mysql database.

The storm architecture of the firefighting system is shown in Figure 2. There are three important nodes: Nimbus node, Supervisor node and database node. Nimbus is equivalent to the “JobTracker” in Hadoop, is the key point of communication between users and storm system. Nimbus is responsible for the release and coordination of the topology of the task execution. Nimbus is stateless, if the nimbus service fails, the work node can continue to further work, but

PoS(ISCC2015)068

can not be carried out the task scheduling until nimbus is resurrection. Supervisor runs on all storm nodes, which accept the tasks assigned by nimbus and start and stop the work processes that belong to their own management. Supervisor is also a fast failure, all the states of nimbus and supervisor are stored in the Zookeeper, so the design is the key to the recovery of storm. After the analysis of the results, you can use mysql to store the data, and mysql uses master slave replication architecture, to avoid the loss of data.

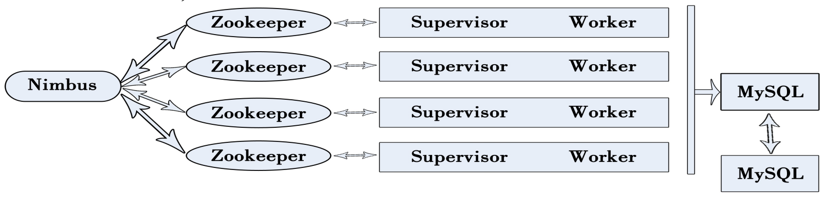

Figure 2: Structure Diagram of Storm in City Fire Control Network System

3.2 System Architecture Design

Based on storm, the fire monitoring center system is based on four layers: data acquisition layer, data access layer, data processing layer and data output layer. As shown in figure 3.

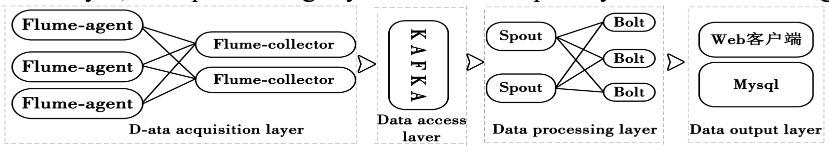

Figure 3. Architecture Diagram in City Fire Control Network System

3.2.1 Data Acquisition Layer

The data acquisition layer is responsible for collecting data from each probe device, using a specific log file to store information:

Since the conventional database is not suitable to be used in frequency searching large amounts of data, the system uses the log file to store the real-time information. The system uses flume to collect data. Flume uses a layered architecture: agent, collector, and storage. Agent is responsible for the log collection. The collector layer is responsible for receiving the agent layer, and the log is written to the corresponding store layer, the store layer is responsible for providing permanent or temporary log storage services, or the log stream to other servers.

Flume has the advantages of high reliability and easy to expand and manage. The system uses it as the main reason for the log collection tool. As shown in figure 4, high reliability is reflected in: in the fire monitoring system, when the monitoring unit server node fail (such as agent A), the log files on the data server can be transferred to the standby data server node (BackAgent1) and not lost. Easy to expand and manage: flume uses three layer architecture, respectively agent, collector and storage, each layer can be horizontally extended. In addition, the system uses the load balancing strategy when the agent layer send data to the collector layer. It sends the log to all the collectors to avoid the problem of a single point of failure.

PoS(ISCC2015)068

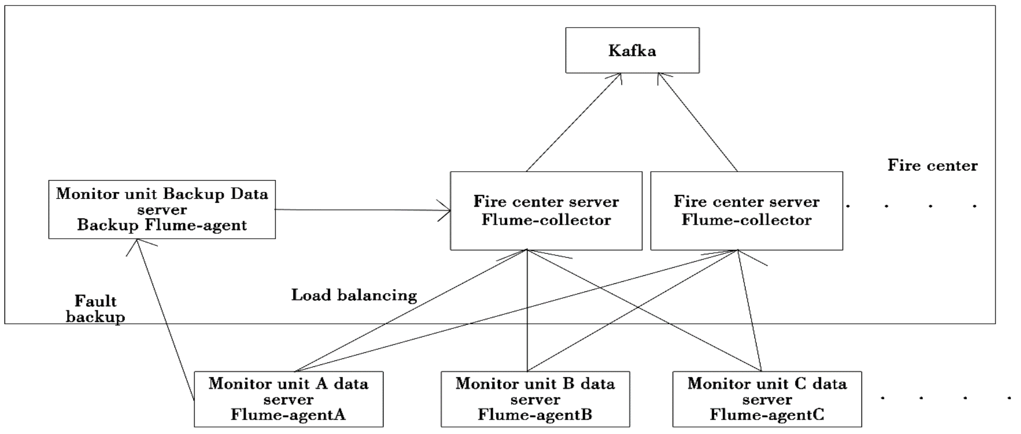
Kafka
Fire center
Monitor unit Backup Data server Backup Flume-agent
Fire center server Flume-collector
Fire center server Flume-collector
Fault backup
Load balancing
Monitor unit A data server Flume-agentA
Monitor unit B data server Flume-agentB
Monitor unit C data server Flume-agentC

Figure 4: Architecture Diagram of Flume

3.2.2 Data Access Layer

Since the speed of flume data acquisition and storm processing data is different, the system uses kafka as a buffer module. Kafka can collect feedback information in real time, and can support large amount of data, and provide the persistence of messages through the disk. This structure can be maintained for a long time, even if the data is up to TB+ level. The system uses kafka to receive data from flume and the message is submitted to the next module of the system.

3.2.3 Data Processing Layer

the Spout and Bolt components of the Storm framework is responsible for the data processing layer. Storm data processing architecture is composed of Bolt and Spout. Spout is the source of data, it will read data from external data sources, and send to Bolt, which is the processing path of Spout. If the Acker is not fully processed by the system and component, Acker will inform Spout that the message is processed. This Spout represents a complete processing of the entire tuple. So we can say that the fault tolerance principle of Storm guarantees the reliable processing of messages.

As shown in Figure 5, there are a lot of ways flowing to Bolt or Bolt stream from a data source of the Spout, which is defined by the stream packet mechanism, which mainly includes: random grouping, field grouping, direct grouping, global grouping, etc. In this system, we use the field grouping first and then the random grouping strategy according to the actual demand. The message type is divided into fire, fault, linkage, feedback, etc. The data stream is sent to a different Bolt by the message type field, the Bolt then sends the received data stream to the other Bolt tasks according to the random group. Finally, the results of the integration calculate by calculated Bolt.

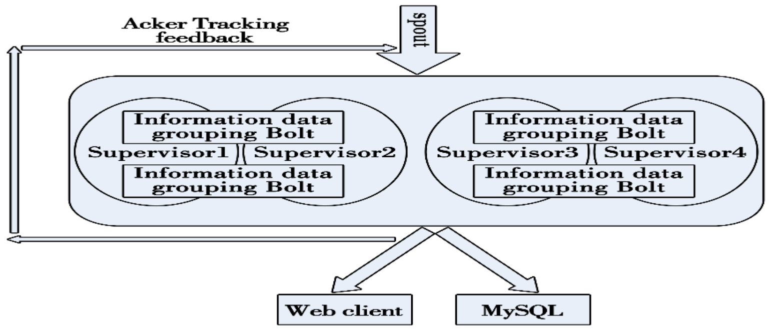
Acker Tracking feedback
spout
Information data grouping Bolt
Supervisor1
Supervisor2
Information data grouping Bolt
Information data grouping Bolt
Supervisor3
Supervisor4
Information data grouping Bolt
Web client
MySQL

Figure 5: Bolt Components Processing Diagram

City fire network system concludes thousands of various type of equipments and other security equipments. The processing of these data form equipments require relatively high timeliness. The data processing module bolt is mainly based on the data of the field grouping to filter out the normal state data in the mass data, processing high real-time requirements of the

PoS(ISCC2015)068

data, such as fire, fault, linkage and other information, and then the data is sent to the web client to display and insert into database. Finally, the tracking module Acker informs the storm framework that the task has been completed.

3.2.4 Data Output layer

The main emphasis of this paper is on the collection and processing of the real time data of the city firefighting system, and of course, the data is displayed on the client page and stored in database. You can use the mysql database system to store the data. Here no more details.

3.3 Displayed on the Client Page

We use express, socket. io interface to update the client information, for the technical more details are no longer described here.

4 The Test

This system is mainly to enhance the real-time and reliability of the city firefighting system in the process of large data processing, the following two aspects from these aspects to verify the performance of the system.

4.1 Real-time Performance Evaluation

Storm cluster configuration is as follows: The system uses 4 hosts to set up the storm cluster, CPU 4 * 2.2G, 2G memory, Gigabit Ethernet card, system environment: Java1.6, Kafka, storm0.9.1, flume1.5. There is one master server in the Cluster. The test case using 10 different size of the log file, which contains 50000 to 500000 lines of data to simulate different data. System each time deal with a file. Test process in real-time recording of the master server CPU, I/O and memory occupancy and other 3 node server hardware resource usage and the time to complete the processing process. Data processing diagram is shown in figure 6.

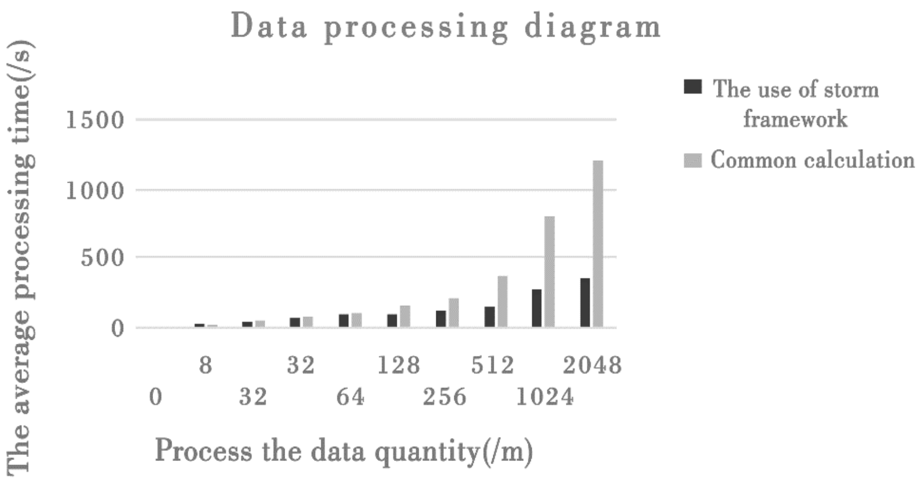

Figure 6: Data Processing Diagram

By contrast, we can see that using storm cloud computing architecture, data processing ability and the efficiency has improved significantly, and with the increase of test processing data, the storm cluster increased nearly doubled on the performance.

PoS(ISC2015)068

4.2 Reliability Performance Evaluation

In this section, our main goal is to examine the resiliency of storm and efficiency when faced with machine failures. In this experiment, we preallocated 8 hosts. The initial number of each component in the topology is as follows:

Component	#tasks
Spout	100
DistributorBolt	100
StatisticalBolt	150
AggregatorBolt	20

Table 1: The Initial Number of Tasks

The working process is set to 30, and has been maintained at 30. We run the topological task on 8 machines. Then, we waited 15 minutes, turn off one machine, and then turn off one of them every 15 minutes.

Finally, we monitor the number of tuple per minute to represent the throughput of the whole architecture, while also monitoring the average processing delay of a tuple. The throughput is measured as the number of tuples acked per minute. Test results of figure are as follows below:

As shown in Figure 7, when we remove a group of machines, there will be a temporary drop in the spike, but then it will come back soon. At the same time, we mainly focus on the throughput every fifteen minutes down once, which means that the same topology is treated with less machines. As shown in the graph, the throughput is also very fast and stable.

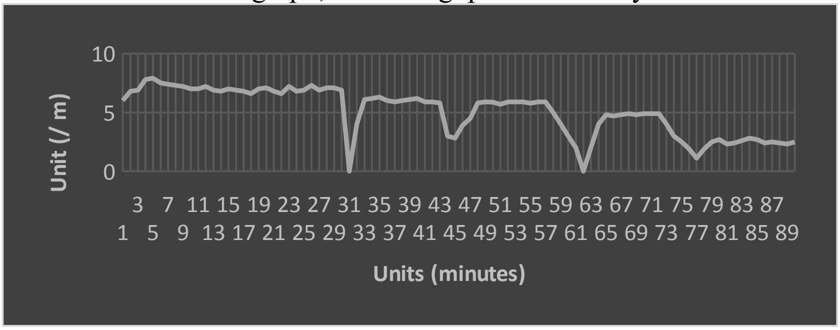

Figure 7: Throughput Measurements

As shown in Figure 8, the average response delay is also increased after shut down one machine. But we also note that in the first few 15 minutes, the delay is very short, but in the last 15 minutes, the delay is relatively large, but the system can also be quickly stabilized processing tasks.

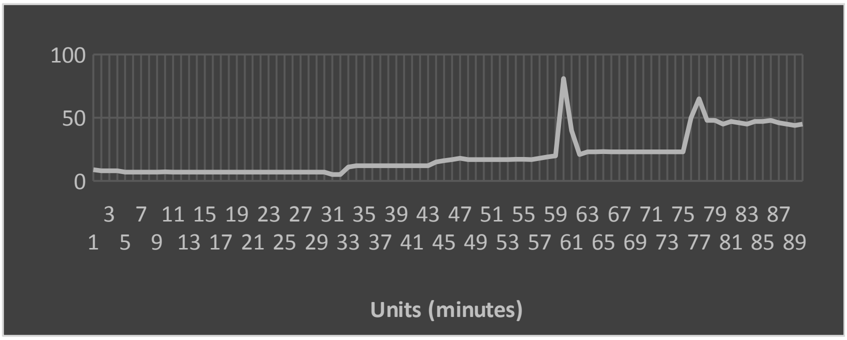

Figure 8: Latency Measurements

In general, as shown in this experiment, Storm has a good ability to recover from machine failures. And the performance of the system can be effectively stabilized when the machine is in trouble.

PoS(ISCC2015)068

5. Conclusion

This paper is to study the fire monitoring center of the monitoring unit of the data for real-time processing, the purpose is to monitor the status of the units of real-time monitoring, timely detection, as well as the failure analysis for future work to lay a good foundation. By analyzing the fire department's corresponding demand situation, and investigating the situation of the existing fire units, we found that the storm cluster system based on cloud computing platform can effectively solve the problem of fire data processing delay. But to provide storm data source of the fire units server or network failure, the fire center is easy to false alarm for this problem, we use the heartbeat detection mechanism to protect the monitoring center in real time to understand the situation of fire monitoring unit, but also to ensure the accuracy of fire data. Storm architecture has a good performance in solving a large amount of data stream real-time processing, but also the future work of the research focus, in the future work, we will be more to improve the visualization method, improve the reliability of the relevant parts, will be more consideration of real-time processing and batch processing of the problem.

References

[1] Hari B, Magdalena B, Donald C, Ugur C, Michael S, Richard T, Stanley B. Zdonik. *Retrospective on Aurora*. VLDB J. 13(4): 370-383 (2004)

[2] Arvind A, Brian B, Shivnath B, Mayur D, Keith I, Rajeev M, Itaru N. *STREAM: The Stanford Stream Data Manager*. IEEE Data Eng. Bull. 26(1): 19-26 (2003)

[3] Apache Kafka, *A high-throughput distributed messaging system*. 2013. http://kafka.apache.org/design.html

[4] Neumeyer L, Robbins B, Nair A, Kesair A. S4: Distributed stream computing platform. In: Proc. Of the 10th IEEE Int'1 Conf. On Data Mining Workshops(ICDMW 2010). Sydney: IEEE Press, 2010. 170-177.

[5] Borthakur D, Sarma JS, Gray J, Muthukkaruppan K, Spigeglberg N, Kuang HR, Ranganathan K, Molkov D, Mennon A, Rash S, Schmidt R, Aiyer A. *Apache Hadoop goes realtime at Facebook.* In: Proc. Of the ACM SIGMOD Int"1 Conf. on management of Data(SIGMOD 2011 and PODS 2011). Athens: ACM Press,2011. 1071-1080.[doi:10.1145/1989323.1989438]

[6] Tyler A, Alex B, Kaya B, Slava C, Josh H, Reuven L. *MillWheel: Fault-Tolerant Stream Processing at Internet Scale*. PVLDB 6(11): 1033-1044 (2013)

[7] Rajagopal A, Venkatesh B, Sumit D, Ashish G, Haifeng J, Tianhao Q. *Photon: fault-tolerant and scalable joining of continuous data streams.* SIGMOD Conference 2013: 577-588

[8] Sankar S, Srikanth B, Hua-Gang L, Vince L, Lei S, Wayne S. *Continuous Queries in Oracle*. VLDB 2007: 1173-1184

[9] Lynch C. Big data: How do your data grow? Nature, 2008, 45(7209): 28-29.

PoS(ISC2015)068