ICIC Express Letters
Volume **11**, Number **11**, November **2017**

 ISSN 1881-803X
pp. **1627–1633**

A STORM-BASED DISTRIBUTED FRAMEWORK FOR BINARY BERNOULLI SAMPLING

Wonhyeong Cho, Sanghun Lee, Siwoon Son and Yang-Sae Moon

Department of Computer Science
Kangwon National University
1 Kangwondaehak-gil, Chuncheon-si, Gangwon-do 24341, Korea
{ whcho; sanghun; ssw5176; ysmoon }@kangwon.ac.kr

Received May 2017; accepted August 2017

Abstract. *In recent years, a large volume of data streams have been rapidly produced in many applications such as SNS (social network service), smart devices, and IoT (internet of things), and accordingly, there have been a lot of needs for sampling techniques on those rapid data streams. In this paper, we deal with the performance improvement of binary Bernoulli sampling which performs sampling in the multiple-input environment. Previous binary Bernoulli sampling, however, has a significant performance degradation problem if the number of input sites is large or the data stream explodes. To solve this problem, we first propose a distributed processing model for improving the performance and scalability of binary Bernoulli sampling. We then implement this model on Apache Storm, a distributed real-time computation system. We also compare its performance with the existing binary Bernoulli sampling on the stream environment. Experimental results show that the proposed Strom-based binary Bernoulli sampling largely improves the performance compared with the existing one by up to 1.8 times. To our best knowledge, this is the first significant attempt that solves the performance degradation problem of sampling algorithms by extending a single processing system to a distributed processing system.*

Keywords: Binary Bernoulli sampling, Data stream, Apache Storm, Distributed processing model, Real-time computation

1. **Introduction.** In recent years, a large volume of data streams have been rapidly generated in manufacturer, network, transportation, SNS (social network service), IoT (Internet of things), etc. [1]. In general, the data stream is generated very fast and continuously. Therefore, for efficient analysis of the data stream, we need to extract samples by using appropriate sampling algorithms that reflect the characteristics and patterns of data streams [2-5]. Sampling is an extraction method in which a part of data representing a population is selected as a sample. Typical sampling algorithms are reservoir sampling [6], priority sampling [7], cluster sampling [8], etc. These sampling methods are not suitable for the real world applications, in which the data streams come from multiple sources. Therefore, we focus on binary Bernoulli sampling [9] that is designed for the multisource environment, i.e., distributed streaming environment [10]. Thus, we mainly discuss the improvement of the processing performance of binary Bernoulli sampling. More precisely, we present how to process the binary Bernoulli sampling in the distributed environment for a large volume of data streams.

A binary Bernoulli sampling framework consists of several sites and a coordinator [9]. Each site receives a data stream from a data source and sends the candidate sample data to the coordinator. Coordinator selects final samples based on the candidate sample data received from the sites. However, as the number of sites increases or the data stream explodes rapidly, the performance of binary Bernoulli sampling significantly decreases. This is because the site sends excessive candidate sample data to the coordinator, which

causes a severe bottleneck in the coordinator. Furthermore, the binary Bernoulli sampling has the duplex communication structure between coordinator and each site, which can lead to an excessive communication overhead because the sites may be connected through different networks. These bottleneck and communication overhead are major problems that degrade the performance of binary Bernoulli sampling. Therefore, in this paper we first propose a distributed processing model of binary Bernoulli sampling to alleviate the bottleneck and communication overhead problems. We then implement this model on Apache Storm [11-14], which is a distributed real-time computing system.

For the distributed processing of binary Bernoulli sampling, we design two major structures. The first one is to design a multiple coordinators structure to alleviate the bottleneck problem. Only one coordinator has a physical limitation to process all the candidate sample data if the number of data sources increases or data explosion occurs. Thus, we try to reduce the bottleneck so that we create several coordinators, and those coordinators process the excessive candidate sample data in a distributed processing manner. The second one is to adopt a single framework structure so that multiple sites and multiple coordinators operate in the same framework to reduce the communication overhead. In addition, we apply a shared memory structure to storing the samples and share the common parameters. Since our single framework has a local area network and the simplex communication, the communication overhead between sites and coordinators becomes much smaller than the existing structure.

In this paper, we implement the proposed distributed model of binary Bernoulli sampling on Apache Storm to verify its practicality and efficiency. Storm is a typical distributed processing system and specialized in data stream processing. To apply our distributed model to Storm, we first implement data sources as spouts and sites and coordinators as bolts [12]. We then use Redis [15], in-memory-based database, as a shared memory for rapid communication between nodes. Experimental results show that, according to the number of nodes, the proposed Storm-based binary Bernoulli sampling improves the performance by up to 1.8 times over the existing binary Bernoulli sampling. We believe that the proposed Storm-based binary Bernoulli sampling is a very meaningful approach that not only extends the multiple-input environment on the distributed model but also improves the performance by adopting Apache Strom as a real distributed processing system.

The organization of this paper is as follows. Section 2 describes related work. Section 3 proposes the distributed model of binary Bernoulli sampling and applies it to Apache Storm. In Section 4, we confirm the superiority of the proposed distributed model through experiments. Finally, Section 5 summarizes and concludes this paper.

2. **Related Work.**

Binary Bernoulli sampling [9] extracts samples with the same probability when data streams are given from multiple sources. Figure 1 shows a working framework of binary Bernoulli sampling. As shown in the figure, we note that binary Bernoulli sampling consists of n sites and one coordinator. Here is a brief description of the binary Bernoulli sampling procedure. First, each site receives a data stream from a data source. Second, all sites receive the *round* from the coordinator and send candidate sample data which can be selected as a sample. That is, each site performs pre-sampling in advance to reduce the overhead of the coordinator. Third, the coordinator selects the samples from the candidate sample data. If the round is updated, the coordinator shares the updated current round with the sites. Binary Bernoulli sampling is performed by repeating this procedure. Please refer to [9] for the detailed explanation.

Binary Bernoulli sampling has a performance degradation problem occurring when a large number of sites or a huge volume of stream data. Binary Bernoulli sampling has an n:1 structure where multiple sites send candidate sample data to a single coordinator. Therefore, if a large number of sites send excessive data to the coordinator, the coordinator

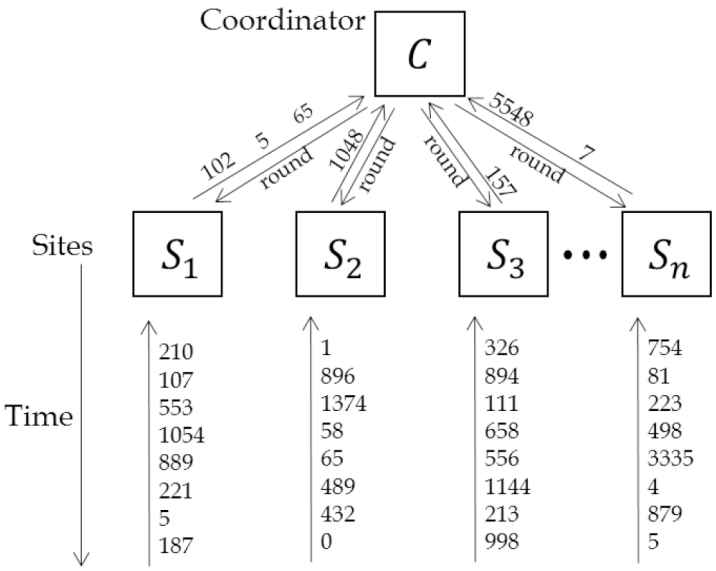

FIGURE 1. A working framework of binary Bernoulli sampling

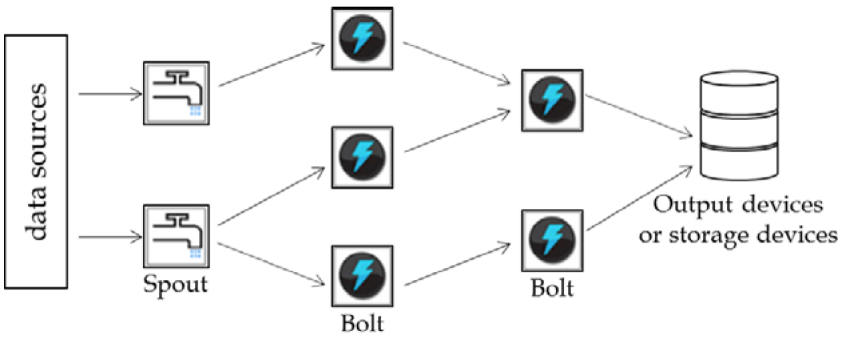

FIGURE 2. A working framework of Apache Strom

shows a bottleneck problem and the sampling performance is significantly degraded. In addition, since the site and coordinator use the duplex communication structure to share the round or candidate sample data while they are located in different networks, frequent communication overheads occur during data transmission which degrades sampling performance significantly. To alleviate these bottleneck and communication overhead problems, in this paper we first propose a distributed processing model of binary Bernoulli sampling with the multiple coordinators structure. We then design multiple sites and coordinators working on a single framework structure.

Apache Storm [11-14] is a typical distributed processing system for handling a large volume of real-time data streams. Storm is specialized in the data stream environment since it constructs multiple servers as a cluster and processes data in real time. In addition, Storm constructs a series of tasks from input to output of stream data as a topology [12], and the topology consists of a number of spouts and bolts. Each spout receives a data stream from data source, converts the data into a tuple [14], and sends it to bolts. Each bolt processes the data received from spouts or other bolts and sends the processed results to other bolts, output devices, or storage devices. Figure 2 shows a working framework of Strom, and readers are referred to [11] for the detailed explanation.

3. **A Storm-Based Binary Bernoulli Sampling.** In this section, we present a Storm-based distributed processing model for binary Bernoulli sampling. Figure 3 shows an overall working framework of the proposed model. As shown in the figure, the proposed

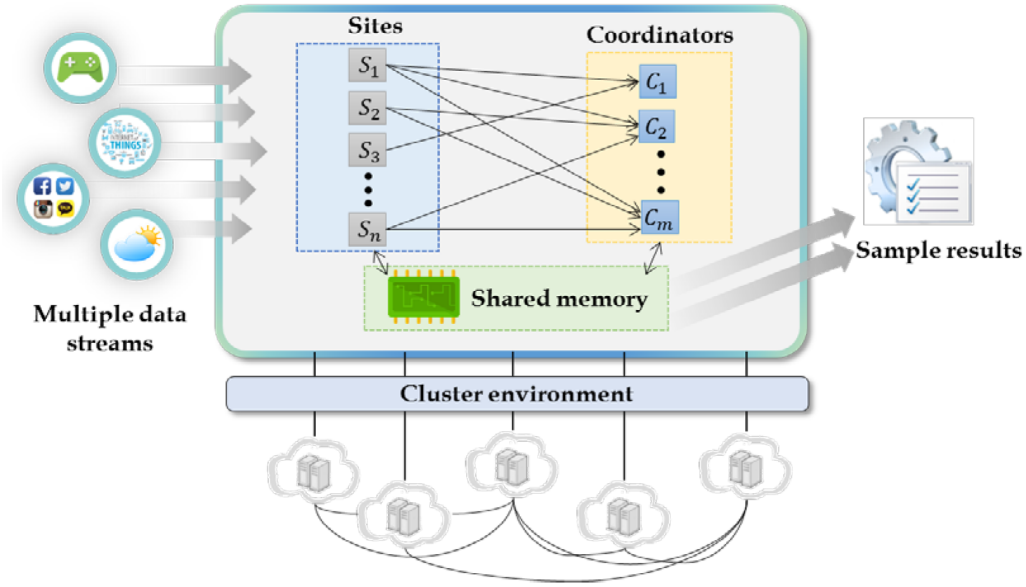

FIGURE 3. A working framework of a distributed binary Bernoulli sampling

distributed model consists of input data streams, sites, coordinators, shared memory, and sample results. Unlike Figure 1, our model has an n:m structure where multiple sites send data to multiple coordinators. Working steps of the distributed binary Bernoulli sampling are as follows. Firstly, data streams come from multiple data sources in the same way as the existing binary Bernoulli sampling. Secondly, each site performing the pre-sampling reads the current round from the shared memory. Thirdly, the site computes whether the input is the candidate sample data that can be selected as a sample in the current round, and if so, it sends the candidate sample data to the coordinator. Here, according to the many-to-many structure, a site may send a sampling candidate to any coordinator. Fourthly, the coordinators select final samples by comparing the candidate sample data with the current samples stored in the shared memory. If a coordinator increases the current round, the coordinator updates the round to the shared memory to share it with other sites and other coordinators. Fifthly, since the sample results are stored in the shard memory, we can use the sample results by accessing the shared memory.

The proposed distributed binary Bernoulli sampling alleviates the bottleneck and excessive communication overhead problems by adopting a single framework structure with multiple coordinators. Firstly, by adopting multiple coordinators structure, we can efficiently handle a large volume of candidate sample data received from the multiple sites as a distributed processing strategy. However, if coordinators are created in different nodes, sample results, round, and other information should be shared among coordinators and sites. To solve this problem, we use a shared memory structure and share these common information through the structure. Secondly, by constructing a single framework structure, we can reduce the communication costs between coordinators and sites. In the binary Bernoulli sampling, since sites and coordinators use duplex communication over different networks, it may show frequent and excessive communication overhead. To alleviate this problem, we first construct sites and coordinators in a single distributed framework. We then change the duplex communication structure to the simplex communication structure through a shared memory. In the existing sampling architecture, coordinators send sampling parameters to sites continuously, which results in frequent duplex communications between coordinators and sites. To solve this problem, we use the simplex communication structure where coordinators store the sampling parameters in the shared memory, and sites get the parameters by simply referring the shared memory. That is, by utilizing a

shared memory in a single framework, the proposed distributed model not only eliminates the communication costs from the coordinator to the site but also reduces the external network overhead.

We now explain how we apply the proposed model to a Storm-based architecture. Figure 4 shows an overall working framework of the Storm-based binary Bernoulli sampling. We here implement the proposed distributed model by fully exploiting Apache Storm functionalities. Comparing Figures 3 and 4, there are three different points as follows. Firstly, we implement the multiple data streams of Figure 3 as spouts. Secondly, we implement the sites and coordinators as bolts. Thirdly, to share the information in real time, we implement the shared memory as Redis [15], an in-memory-based DBMS (database management system). As shown in Figure 4, each site bolt receives a data stream from a spout and sends its candidate sample data to any coordinator bolt. Each coordinator bolt selects samples from the candidate sample data. The round is shared through Redis, and the sampling results also share the user through Redis.

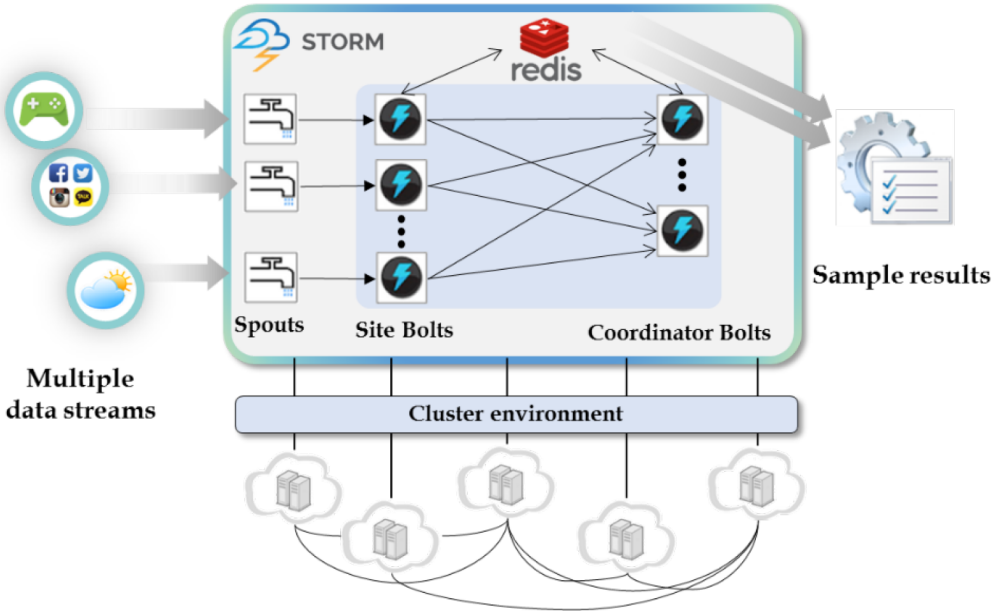

FIGURE 4. A working framework of a Storm-based binary Bernoulli sampling

4. **Performance Evaluation.** In the experiment, we compare the execution times of the existing binary Bernoulli sampling with the proposed Storm-based binary Bernoulli sampling. The hardware platform consists of one master node (Xeon E5-2630V3 2.4GHz, 8 Core) and eight slave nodes (Xeon E5-2630V3 2.4GHz, 6 Core), each of which has 32GB RAM and 256GB SSD. The software platform of all nodes is CentOS 7.2.1511 Linux operating system. As an input data stream, we randomly generate 1,000,000 text streams in each spout. For the existing binary Bernoulli sampling, we use a single node structure with one coordinator in Storm.

Table 1 shows the execution times measured at a single node (existing sampling model) and distributed nodes (proposed sampling model), respectively, and Figure 5 shows the execution times as a graph. Note that "1-2 nodes" means one master-two slaves. Likewise, "1-4 node" and "1-8 node" mean one master-four slaves and one master-eight slaves, respectively. As shown in the table and the figure, we note that, as the number of nodes increases, the execution time of the existing sampling rapidly increases while that of the proposed model increases gradually. In particular, as the number of spouts increases, that is, as the amount of input stream data increases, the performance improvement is much larger than the existing sampling model. In the distributed processing model,

TABLE 1. Execution times of single and distributed nodes

(*ms*)

No. of spouts	Single node	Distributed nodes (master-slaves)		
		1-2	1-4	1-8
1	181,859	176,573	160,778	166,669
2	189,636	176,583	167,592	173,680
4	229,482	207,696	179,689	176,551
8	357,927	238,171	220,631	193,482

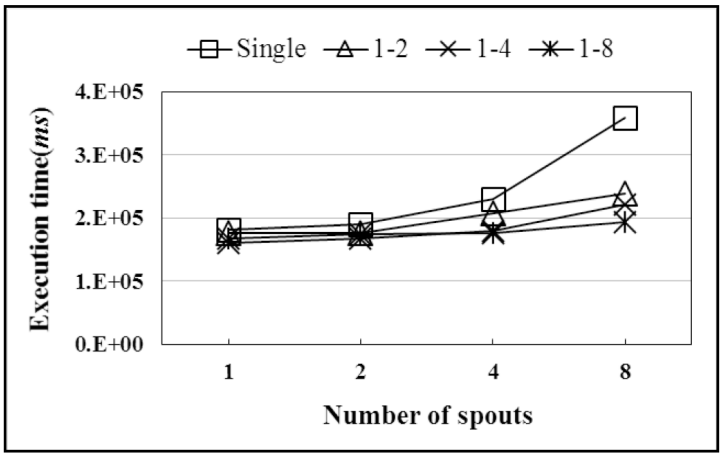

FIGURE 5. Change of execution times of single and distributed nodes

the throughput increases as the number of nodes increases. If the input data size is small, the performance of the single model and that of the distributed model may be similar. However, as the input stream increases, the distributed model with relatively high throughput will perform much faster than the single model. This indicates that the proposed Storm-based model beats the existing single model for the large and fast data stream environment. In summary, the proposed Storm-based binary Bernoulli sampling improves the performance by up to 1.8 times compared with the existing one. In addition, as the number of nodes increases, this performance difference becomes much bigger. It means that the proposed distributed processing model is suitable for a large volume of data stream environment.

5. **Conclusions.** In this paper, we proposed a distributed processing model for improving the performance of binary Bernoulli sampling and implemented the model on Apache Storm. Since the existing binary Bernoulli sampling works on a single node, it has a performance degradation problem for a large volume of stream data. To solve this problem, we first proposed a distributed processing model where binary Bernoulli sampling operates on multiple nodes. We then implemented the model on Storm which was specialized in data steam processing. Experimental results indicated that the proposed Storm-based binary Bernoulli sampling improved the performance compared with the existing one by up to 1.8 times. We also note that, as the number of nodes or a volume of data increases, the performance improvement significantly increases. By these experimental results, we believe that the proposed distributed binary Bernoulli sampling is an excellent approach that not only extends the existing binary Bernoulli sampling to the distributed structure but also improves the performance through the Storm-based actual implementation. As the future work, firstly we will consider the InfiniBand [16] network to minimize communication costs between sites and coordinators. Secondly we will apply the sliding window mechanism [17] to binary Bernoulli sampling.

Acknowledgment. This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. R7117-17-0214, Development of an Intelligent Sampling and Filtering Techniques for Purifying Data Streams) and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B4008991).

REFERENCES

[1] G. D. F. Morales, A. Bifet, L. Khan, J. Gama and W. Fan, IoT big data stream mining, *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, pp.2119-2120, 2016.

[2] J. Leskovec, A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*, Cambridge University Press, 2015.

[3] *Batch vs. Real Time Data Processing*, http://www.data sciencecentral.com/.

[4] G. Cormode and N. Duffield, Sampling for big data, *Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, USA, p.1975, 2014.

[5] X. Wu, X. Zhu, G. Q. Wu and W. Ding, Data mining with big data, *IEEE Trans. Knowledge and Data Engineering*, vol.26, no.1, pp.97-107, 2014.

[6] J. S. Vitter, Random sampling with a reservoir, *ACM Trans. Mathematical Software*, vol.11, pp.37-57, 1985.

[7] E. Cohen, Stream sampling for frequency cap statistics, *Proc. of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Sydney, Australia, pp.159-168, 2015.

[8] F. J. Fowler, *Survey Research Methods*, Thousand Oaks, 2015.

[9] G. Cormode, S. Muthukrishnan, K. Yi and Q. Zhang, Optimal sampling from distributed streams, *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Indianapolis, Indiana, pp.77-86, 2010.

[10] R. Ranjan, Streaming big data processing in datacenter clouds, *IEEE Cloud Computing*, vol.1, no.1, pp.78-89, 2014.

[11] *Apache Storm*, http://storm.apache.org/.

[12] Z. Yu, Y. Liu and X. Yum, Scalable distributed processing of k nearest neighbor queries over moving objects, *IEEE Trans. Knowledge and Data Engineering*, vol.5, no.5, pp.1383-1396, 2014.

[13] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal and D. Ryaboy, Storm@Twitter, *Proc. of the 21st ACM SIGMOD International Conference on Management of Data*, Snowbird, Utah, pp.147-156, 2014.

[14] J. Xu, Z. Chen and J. Tang, T-Storm: Traffic-aware online scheduling in Storm, *Proc. of the IEEE 34th International Conference on Distributed Computing Systems*, Madrid, Spain, pp.535-544, 2014.

[15] M. Xu, A forensic analysis method for redis database based on RDB and AOF file, *Journal of Computers*, vol.9, no.11, pp.2538-2544, 2014.

[16] S. Gugnani, X. Lu and D. K. Panda, Performance characterization of Hadoop workloads on SR-IOV-enabled virtualized InfiniBand clusters, *Proc. of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, Shanghai, China, pp.36-45, 2016.

[17] V. Braveman, R. Ostrovsky and C. Zaniolo, Optimal sampling from sliding windows, *Journal of Computer and System Sciences*, vol.78, no.1, pp.260-272, 2012.