ICIC Express Letters ICIC International ©2021 ISSN 1881-803X
Volume **15**, Number **3**, March **2021** pp. **265–272**

PERFORMANCE IMPROVEMENT OF DISTRIBUTED PROCESSING IN BINARY BERNOULLI SAMPLING

Wonhyeong Cho and Yang-Sae Moon*

Department of Computer Science
Kangwon National University
1 Gangwondaehakgil, Chuncheon-si, Gangwon-do 24341, Korea
whcho@kangwon.ac.kr; *Corresponding author: ysmoon@kangwon.ac.kr

Received September 2020; accepted November 2020

Abstract. *This paper addresses the problem of distributed processing in binary Bernoulli sampling (BBS in short). BBS is a stochastic sampling for the multi-source stream environment, and we need to use distributed processing of BBS to process large volumes of data streams generated from multiple input sources. Accordingly, a recent work proposed a distributed BBS model based on Apache Storm having the multiple coordinator structures. However, this technique has limitations in improving performance due to the coordinator waiting problem. In this paper, we solve the coordinator waiting problem by introducing multi-distribution and distributor separation structures. The multi-distribution structure minimizes the waiting time by participating multiple coordinators rather than only one in the distribution. The distributor separation structure maximizes processing performance by separating the distribution function from the coordinator. To experimentally evaluate the performance improvement, we apply the improved structures to Storm-based distributed BBS. Experimental results show that the multi-distribution and distributor separation structures improve the performance up to 90 times compared to the single distribution structure.*

Keywords: Binary Bernoulli sampling, Data stream, Distributed sampling, Apache Storm

1. Introduction.

As a stochastic sampling technique suitable for multi-source stream environments, binary Bernoulli sampling (hereinafter referred to as *BBS*) uses a single coordinator to process continuous streams coming from multiple input sites [1]. As the number of sites increases and big data streams are generated, however, there will be a coordinator bottleneck problem that is difficult to process with one coordinator. To deal with this problem, a recent work [2] proposed a distributed BBS model based on Apache Storm [3, 4] exploiting multiple coordinators. Figure 1 shows the structure of the recent distributed BBS model using multiple coordinators. As shown in the figure, we can solve the coordinator bottleneck problem by operating several coordinators in an integrated framework [2].

The distributed BBS model [2] of the multiple coordinator structure solves the coordinator bottleneck problem, but additionally incurs the *coordinator waiting* problem. This problem occurs because while one coordinator performs sample distribution, other coordinators should wait for data processing. According to the experiment using Storm, the data processing becomes faster as the number of coordinators increases, but the sample distribution is nearly constant regardless of the increase in coordinators. This coordinator waiting problem is a big limitation of improving the overall performance of the distributed BBS model.

To solve the coordinator waiting problem, in this paper we present the concept of *multi-distribution* structure and *distributor separation* structure. First, the multi-distribution

DOI: 10.24507/icicel.15.03.265

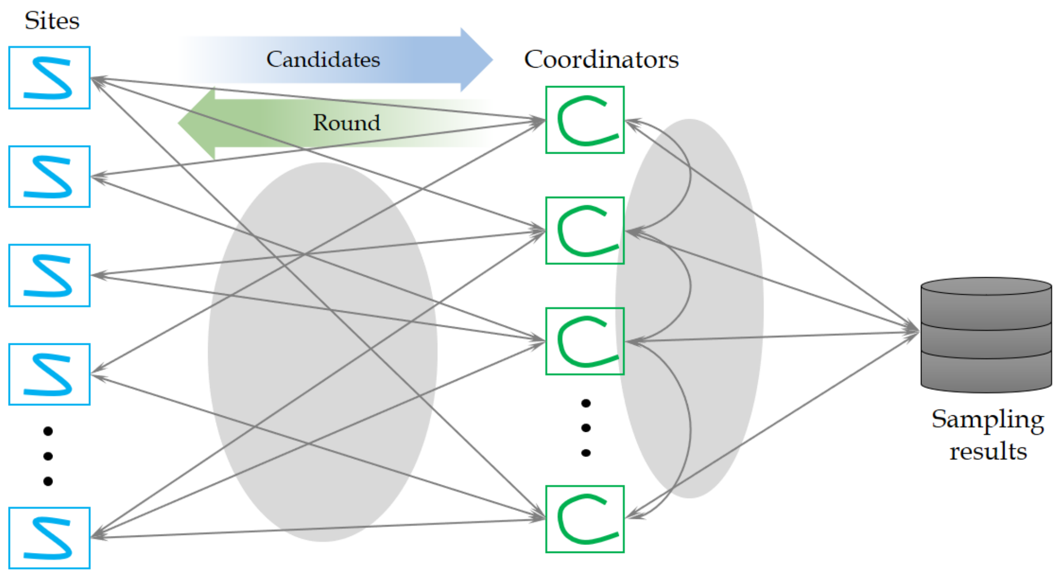

FIGURE 1. Example of applying stratified sampling to the stream environment

structure provides a novel parallelism in which multiple coordinators instead of one coordinator perform distribution at the same time by placing extra samples. Using such a multi-distribution structure we can solve the coordinator waiting problem because multiple coordinators can perform sample distribution at the same time. Next, the distributor separation structure adopts additional distributors that manages the sample distribution function separated from the original coordinators. Through this separation structure, the coordinator performs the data processing function only while the distributor manages the distribution function, thereby maximizing the overall performance of distributed processing.

In this paper, we implement the proposed multi-distribution and distributor separation structures in the Storm-based BBS [2] and perform multi-source sampling experiments in an actual distributed environment. Experimental results show that the proposed multi-distribution structure improves the performance up to 90 times compared to the single distribution structure. In addition, we experimentally prove that the more number of coordinators we use, the higher the performance we can achieve.

The rest of the paper is organized as follows. Section 2 describes the related work on distributed processing of binary Bernoulli sampling. Section 3 presents the proposed improvement solution to the distributed BBS model. Section 4 shows the results of experimental evaluation. Finally, Section 5 concludes the paper.

2. Related Work.

BBS is a representative probability-based sampling algorithm that samples data with the same probability for data streams [5, 6] generated from multiple sources [1]. The general structure of BBS consists of one coordinator and several input stream sites. BBS performs sampling using an infinite *binary string* consisting of 0s and 1s that are input with stream data. A brief description of the procedure for performing the BBS sampling is as follows [2]. First, each site receives a *round* from the coordinator, selects candidate data in the current round using a binary string, and then transmits the candidates to the coordinator. Next, the coordinator selects an actual sample from the candidate samples delivered from the site. If the current round changes during the sampling process, the coordinator shares the changed round with all sites. The coordinator and sites continuously repeat these candidate/sample selection and round change/transmission processes to perform sampling for multiple data streams continuously generated from multiple sources.

For efficient sampling of large-volume data streams, a distributed BBS with the structure of Figure 1 was recently proposed [2]. As shown in Figure 1, the distributed BBS model solves the coordinator bottleneck by providing the multiple coordinator structure. That is, since several coordinators participate in sampling at the same time, the distributed model can efficiently process the BBS for large volumes of stream data. In the distributed BBS model, however, the coordinator waiting problem in Figure 2 occurs when the samples are being distributed. Table 1 shows pre-experimental results for data processing time and sample distribution time, respectively, in distributed BBS. In the table, we note that while the data processing time decreases as the number of coordinators increases, there is no significant change in the sample distribution time due to the coordinator waiting problem. In this paper, we present an improvement technique for solving this coordinator waiting problem. Previous distributed sampling methods include MapReduce-based sampling technique [7] and Apache Spark-based distributed graph sampling [8], but these methods do not cover the distributed BBS model.

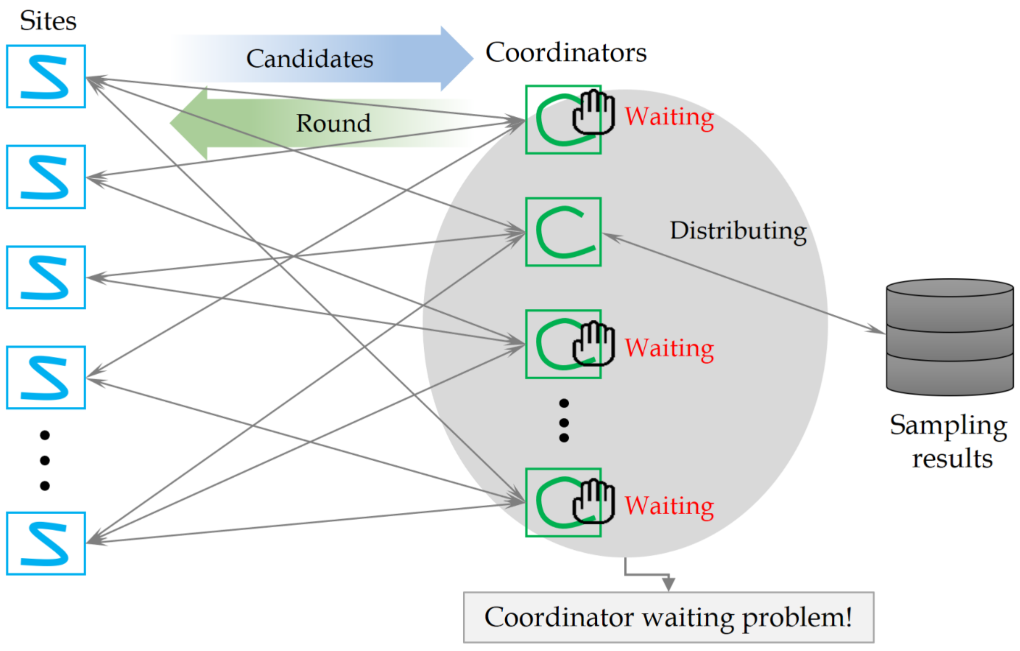

FIGURE 2. Coordinator waiting problem when distributing samples in the distributed BBS model

TABLE 1. Pre-experimental results of data processing time and sample distribution time in the distributed BBS model

Number of coordinators	Data processing time (ms)	Data distribution time (ms)
1	–	–
2	26,362	44,177
3	19,681	44,988
4	15,769	44,486

3. Improvement of the Distributed BBS Model.

In this section, we present a multi-distribution structure and a distributor separation structure for optimizing the distributed BBS model [2]. First, the *multi-distribution structure* is to alleviate the coordinator waiting problem, which is a major performance limitation of the distributed BBS model. In this paper, we design a multi-distribution structure by adding samples, and we call such added samples *extra samples*. The reasons for using the extra samples are as follows.

First, by using the extra samples, multiple coordinators can proceed with distribution at the same time. By this parallel distribution, we can significantly reduce the speed of processing the sample distribution. Second, there is no need to guarantee logical synchronization that occurs as the existing model [2] uses the shared memory structure such as Redis [9]. Since the multi-distribution structure performs distribution at the same time, there is no need for logical synchronization to wait for the coordinator in the existing model. As a result, the multi-distribution structure greatly reduces the coordinator waiting overhead of the previous multiple coordinator structure.

The second concept for optimizing the distributed BBS model is the *distributor separation structure*. In the distributed BBS model, the coordinator manages the *main samples* provided to the user and the *supplementary samples* additionally used when the main samples are insufficient. While a coordinator is distributing in the BBS, the coordinator cannot add data to main and supplementary samples due to its distribution work. Thus, all of the data that will belong to the main and supplementary samples are accumulated in the buffer, which may cause a serious decrease in sampling performance. In order to resolve this buffering phenomenon, the distributor separation structure functionally divides a coordinator into a coordinator and a distributor that process the main samples and the supplementary samples, respectively. Figure 3 shows how this distributor separation structure works. As shown in Figure 3, unlike the existing structure of Figure 2, which has only coordinators, there are coordinators that manage supplementary samples and distributors that manage main samples. The coordinator adds data to the supplementary samples, and the distributor adds data to the main samples. And, the distributor does the sample distribution work performed when the main samples are saturated. By adopting this separation structure, we can continuously add the data to supplementary samples. That is, when distributing samples, the existing model piles up data in a buffer, but the distributor separation structure can process the data to be added to supplementary samples without buffering or clogging, ensuring high processing efficiency.

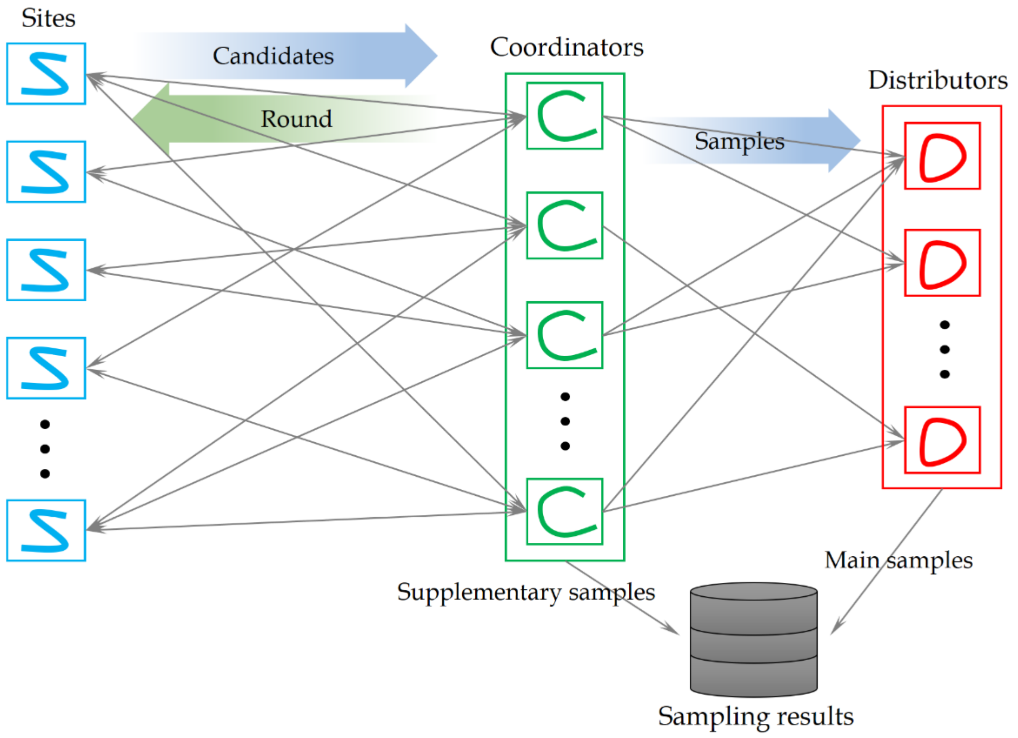

FIGURE 3. Operation concept of the distributor separation structure in the distributed BBS

Figure 4 shows the final structure of the distributed BBS model optimized with such multi-distribution and distributor separation structures. Compared to the basic distributed BBS model, it has a structure with separate distributors and extra samples. In addition, we change the functions of coordinators and distributors so that a coordinator adds data to the supplementary samples, and a distributor adds data to the main samples. We also change the distribution function so that all distributors proceed with multiple distribution of samples. In this paper, we implement the distributed BBS and the proposed optimization functions using *Spouts* and *Bolts* of Apache Storm [3] as in the previous work of [2].

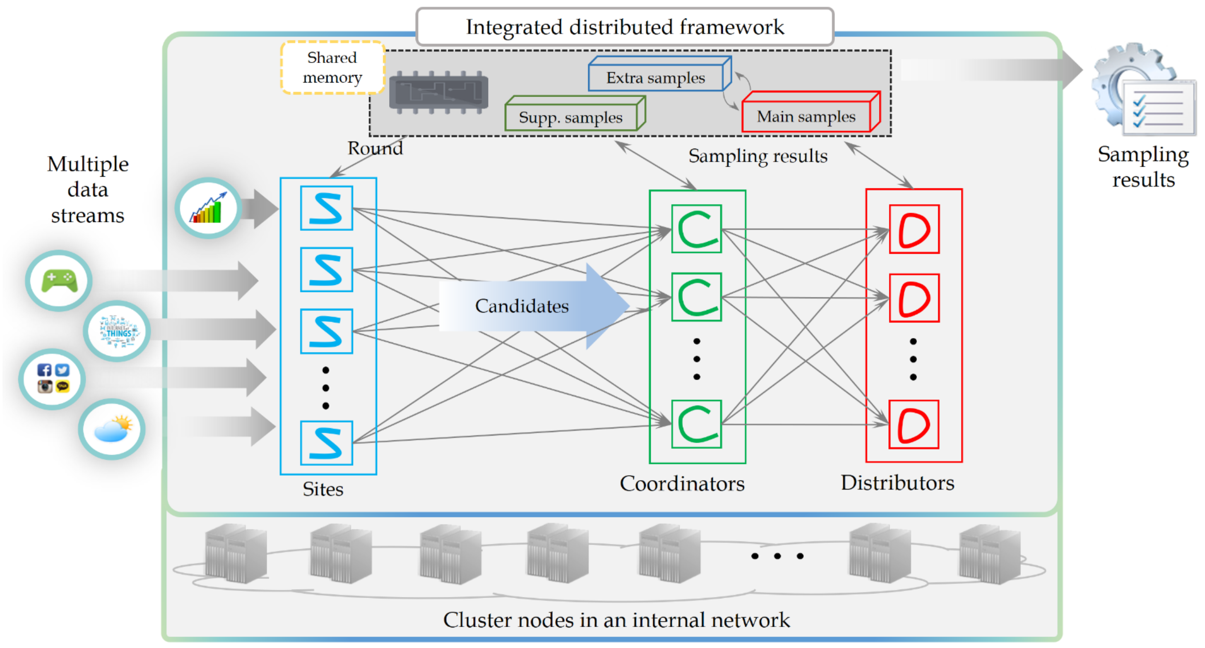

FIGURE 4. Improved structure of the distributed BBS through adopting multi-distribution and distributor separation structures

4. **Experimental Evaluation.** The hardware platform for the experiment consists of one master node (Xeon E5-2630V3 2.4GHz, 8Core, 32GB RAM, 256GB SSD) and eight slave nodes (Xeon E5-2630V3 2.4GHz, 6 Core, 32GB RAM, 256GB SSD). The software platform is CentOS Linux 7.2.1511, and the network bandwidth is 1Gbps. We implement the distributed BBS model using the Java language. The data used in the experiment is a randomly generated text stream, and 1,000,000 data items are transmitted into one Spout of Storm. We use the basic functions of Storm for load balancing and failure recovery for nodes [10].

In the first experiment, we compare the data processing time of single distribution and multiple distribution. Table 2 shows the experimental results of single and multiple distributions, and Figure 5 depicts the results as a comparative graph. As shown in the table and figure, when the number of coordinators is the same, the multi-distribution structure significantly shortens the data processing time by up to 23 times compared to the single distribution structure. In the single distribution structure, the processing time increases by only 1.2 times and 1.4 times as the number of coordinators increases, but in the multi-distribution structure, the performance largely improves by 2.1 times and 2.6 times. This is because when the number of coordinators increases, the sample distribution time is constant in the single distribution structure, but the distribution time gradually decreases in the multi-distribution structure. Thus, we can conclude that the multi-distribution structure is much more efficient than the single distribution structure in handling large volumes of multiple data streams.

TABLE 2. Data processing times of single and multiple distributions in the distributed BBS model

Number of coordinators	Data processing time (*ms*)	
	Single distribution	Multiple distribution
1	–	–
2	21,228	3,081
3	17,381	1,444
4	12,577	551

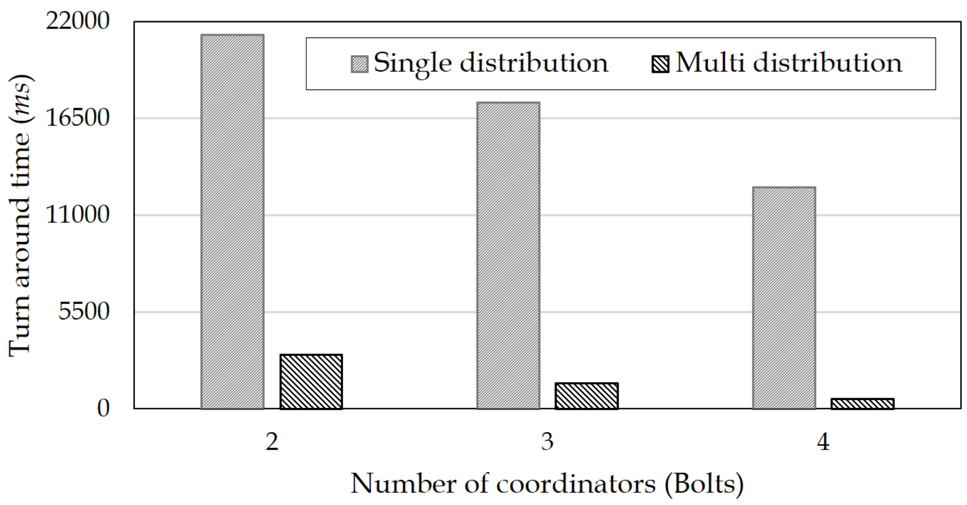

FIGURE 5. Comparison of data processing times of single and multiple distributions in the distributed BBS model

TABLE 3. Data processing time of the distributed BBS model exploiting both multi-distribution and distributor separation structures

Use of distributors or not	Number of distributors	Data processing time (*ms*)
No use of distributors	Single distribution	21,228
	Multiple distribution	3,081
Use of distributors	1	1,066
	2	663
	3	492
	4	235

Next, we experiment the data processing time of the distributed BBS model applying the multi-distribution structure together with the distributor separation structure. In the experiment, we fix the number of coordinators to two, and conduct the experiment while changing the number of distributors to which multiple distributions are applied. (Precisely speaking, we implement coordinators and distributors as Bolts of Apache Storm [2, 3].) Table 3 shows the experimental results of the multi-distribution and distributor separation structures, and Figure 6 depicts the results as a graph. The table and graph show that the data processing performance improves as the number of distributors increases. This is because coordinators and distributors manage main samples and supplementary samples separately, and apply the multi-distribution structure so that the distributors simultaneously perform the distribution.

In this section, to confirm the performance improvement of the distributed BBS model, we have experimented the enhanced model by adopting multi-distribution and distributor separation structures. Summarizing the experimental results, we confirm that the

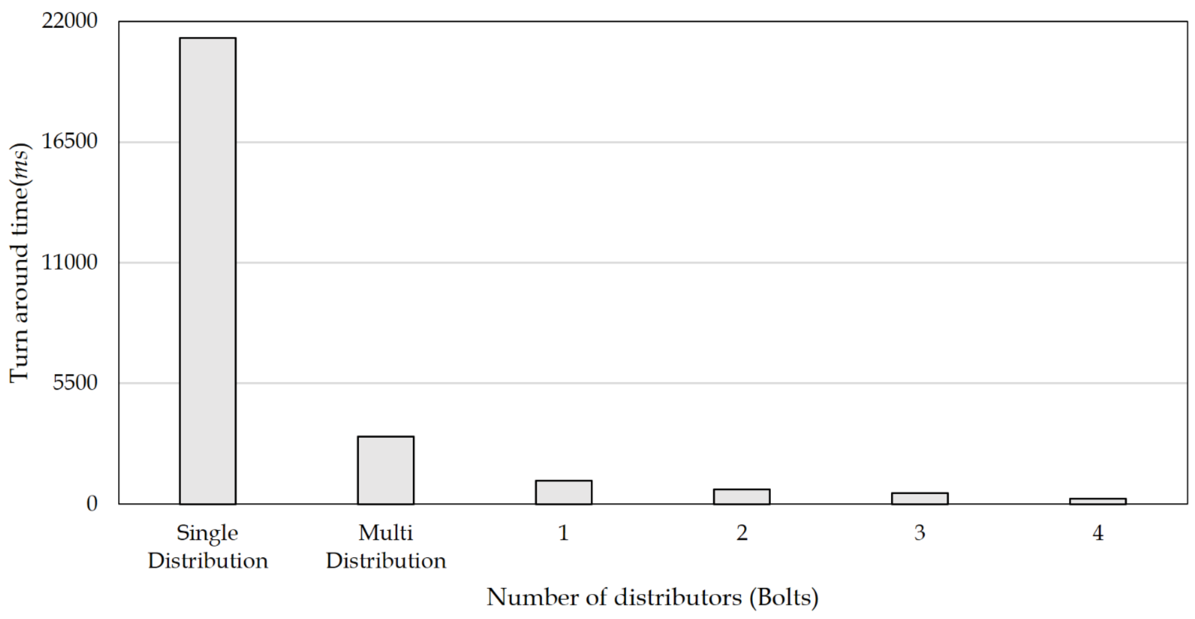

Figure 6. Comparison of data processing times of the distributed BBS model varying the number of distributors

proposed multi-distribution structure improves the data processing time up to 90 times compared to the *original* multiple coordinator structure. In addition, the distributed BBS model exploiting both multi-distribution and distributor separation structures finally shows the fastest data processing time. Through these experimental results, we believe that the proposed multi-distribution and distributor separation structures optimize the performance of the distributed BBS model.

5. **Conclusions.** In this paper, we figured out the coordinator waiting problem, which is a disadvantage of the distributed BBS model, and resolved the problem by introducing the concept of multi-distribution and distributor separation structures. Multi-distribution structure has the effect of greatly reducing the coordinator waiting overhead by distributing the coordinator's distribution function. The distributor separation structure can maximize the effect of distributed processing by separating the distribution function from the coordinator to the distributor newly adopted. According to the experimental results, the distributed BBS model, which solved the coordinator waiting problem, improved the data processing time by up to 90 times compared to the basic model [2]. As future work, we have a plan to minimize the communication cost between the coordinator and the source site by utilizing the high-speed network InfiniBand [11].

Acknowledgment. This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. 2019R1A2C1085311) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (No. 2020-0-00077, Core Technology Development for Intelligently Searching and Utilizing Big Data based on DataMap).

REFERENCES

[1] G. Cormode, S. Muthukrishnan, K. Yi and Q. Zhang, Optimal sampling from distributed streams, *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems*, Indianapolis, IN, pp.77-86, 2010.
[2] W. Cho, M.-S. Gil, M.-J. Choi and Y.-S. Moon, Storm-based distributed sampling system for multi-source stream environment, *International Journal of Distributed Sensor Networks*, vol.14, no.11, 2018.
[3] *Apache Storm*, http://storm.apache.org/, 2017.

[4] 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 International Conf. on Management of Data*, Snowbird, UT, pp.147-156, 2014.

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

[6] Z. Shou, F. Zou, S. Li and X. Lu, Outlier detection based on density of hypercube in high-dimensional data stream, *International Journal of Innovative Computing, Information and Control*, vol.15, no.3, pp.873-889, 2019.

[7] A. Haque, S. Chandra, L. Khan and C. Aggarwal, Distributed adaptive importance sampling on graphical models using MapReduce, *Proc. of 2014 IEEE International Conf. on Big Data*, Washington, D.C., pp.597-602, 2014.

[8] F. Zhang, S. Zhang and C. Lightsey, Implementation and evaluation of distributed graph sampling methods with spark, *Electronic Imaging*, DOI: 10.2352/ISSN.2470-1173.2018.01.VDA-379, 2018.

[9] 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.

[10] Z. Yu, Y. Liu, X. Yu and K. Q. Pu, Scalable distributed processing of k nearest neighbor queries over moving objects, *IEEE Trans. Knowledge and Data Engineering*, vol.27, no.5, pp.1383-1396, 2015.

[11] 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 Conf. on Big Data Computing Applications and Technologies*, Shanghai, China, pp.36-45, 2016.