Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 601868, 7 pages
http://dx.doi.org/10.1155/2014/601868

Research Article

Hadoop-Based Distributed Sensor Node Management System

In-Yong Jung, Ki-Hyun Kim, Byong-John Han, and Chang-Sung Jeong

Department of Electrical Engineering, Korea University, Seoul 136-713, Republic of Korea

Correspondence should be addressed to Chang-Sung Jeong; csjeong@korea.ac.kr

Received 30 August 2013; Accepted 6 March 2014; Published 30 March 2014

Academic Editor: Hwa-Young Jeong

We present a new architecture of HDSM (Hadoop-based distributed sensor node management system) for distributed sensor node management using Hadoop mapreduce framework and distributed file system (DFS). It offers various efficient ways for collecting sensor data and managing multiple sensor nodes by launching specific mapreduce applications on sensor nodes which upload data of sensor nodes to DFS and retrieve sensor data periodically from DFS. Additionally, it provides a flexible management scheme for sensor node by reconfiguring firmware or updating configurations and data formats of sensor nodes based on mapreduce framework. Also, it achieves a transparent and fault-tolerant architecture by exploiting various crucial features of Hadoop. Our experimental results show that it has efficient and stable performance.

1. Introduction

In ubiquitous computing environment with wired/wireless sensor networks, efficient sensor data collection, monitoring, and management are very significant for various useful data analyses [1]. Various sensor nodes such as CCTVs, temperature sensors, or Wi-Fi hotspots can be deployed with additional processor and storage devices for saving sensing data locally and connected to each other through networks such as Ethernet. These sensor networks need sophisticated approaches to achieve massive sensor data collection and cost-effective management of distributed sensor nodes.

Mapreduce is a useful framework in a wide range of applications including distributed parallel searching, indexing, clustering, and sorting for various data types and replaces old ad hoc software that performs the various analyses for large data sets [2]. Moreover, mapreduce programming model offers an easier way to build distributed parallel application running on multiple computing nodes for processing big data. Therefore, it has been adapted to several computing environments such as multiprocessor systems and cloud environments. So far, there exist various implementations for mapreduce algorithm and its execution framework such as Hadoop. They may be integrated with several distributed file systems (DFS) and offer overall management of the whole process for launching mapreduce application, integrating multiple servers, and managing communications and data transfers between components. They provide redundant and fault-tolerant architecture via their distributed architecture, data replication, and heartbeat mechanism.

However, the previous approaches are focused on accumulating log data of applications generated from other servers connected through wired network onto repository based on DFS and processing them by using mapreduce. In our research, we regard sensor network as a large scale mapreduce cluster and present a new sensor node management platform, called HDSM (Hadoop-based distributed sensor node management system) for distributed sensor node management by exploiting mapreduce framework and DFS and launching various mapreduce applications for executing sensor data uploading, retrieving, flushing, monitoring, and updating configurations. It offers various efficient ways for collecting sensor data and managing multiple sensor nodes by launching specific mapreduce applications on sensor nodes which upload data of sensor nodes to DFS and retrieve sensor data periodically from DFS. Additionally, it provides a flexible management scheme for sensor node by reconfiguring a firmware or updating configurations and data formats of sensor nodes based on mapreduce framework. Also, it achieves a transparent and fault-tolerant architecture by exploiting various crucial features of Hadoop. Our

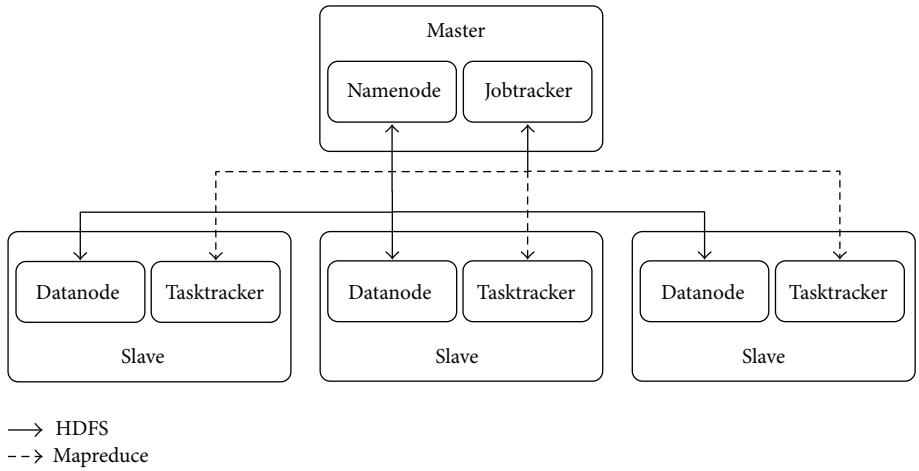

Figure 1: Architecture of Hadoop deployed on multiple nodes; HDFS and mapreduce and interaction between components.

experimental results show that it has efficient and stable performance.

The rest of the paper is organized as follows. First, in Section 2, we describe related works. Then, in Section 3, we describe an architecture of HDSM. Section 4 shows its experimental results, and Section 5 summarizes the conclusions of our research.

2. Related Works

Mapreduce application is comprised of two stages: mapper for processing fine-grained unit tasks such as filtering, counting, or sorting and reducer for summarizing operations [2]. There are various implementations for mapreduce algorithm and its execution framework such as Google mapreduce [3], Apache Hadoop [4], and Twister [5], and inputs and outputs of mapreduce are usually stored in a distributed file system such as GFS [6] and HDFS [4]. They offer orchestration of the whole management processes for launching mapreduce application, monitoring multiple nodes, data transferring, and managing communications between components.

Apache Hadoop is an open-source implementation of mapreduce framework that supports the running of applications on large clusters of commodity hardware and distributed file system named Hadoop distributed file system (HDFS) that stores large data on the computing nodes [2, 4]. It is written in the Java programming language. A Hadoop cluster includes a single master node and multiple slave nodes as described in Figure 1. For HDFS, Namenode is a master component which manages file locations and metadata of all files on HDFS. Namenode checks liveness of all slave nodes by heartbeat mechanism based on periodical notification with simple messages and offers command line interface and API layer for clients. Real data files on HDFS are divided into multiple chunks and managed by slave component named Datanode. It stores chunks into local disk of each worker node. Mapreduce engine for executing mapreduce application is comprised of a single Jobtracker on master node as a job coordinator process and multiple Tasktrackers on slave nodes as worker demons. After receiving job request for launching a mapreduce application, Jobtracker schedules and allocates map and reduces tasks to multiple Tasktrackers with awareness of the data location and makes them executed in parallel way. Mostly results of mapreduce jobs are saved into the HDFS. Both mapreduce engine and the HDFS are designed so that node failures are automatically handled by the framework. They provide redundant and fault-tolerant architecture via their expandable architecture and unique monitoring scheme named heartbeat mechanism. Therefore, their architecture is useful to construct distributed sensor node management system.

There exist a number of researches for processing collected sensor data using distributed parallel computing [7–14]. Also, there are various approaches for constructing large data collection systems on grid and cloud environments [15–17]. In particular, there are some approaches which are integrated with Hadoop mapreduce framework. Apache Flume [18] is a distributed service for efficiently collecting, aggregating, and transferring large amounts of log data from many different sources to HDFS. It offers a flexible architecture based on streaming data flows and uses a simple extensible data model that allows for online analytic application. Honu [19] is a large scale streaming data collection and processing pipeline built using Hadoop. It collects and processes all the unstructured and structured log events generated from various applications running on Amazon EC2 [20] or EMR [21] and makes them available to user through Hive [22].

However, these approaches are focused on accumulating log data of applications generated from other application servers connected through wired network onto repository based on DFS and processing them by using mapreduce. In our research, we regard sensor network as a large scale mapreduce cluster and present a sensor node management platform by exploiting mapreduce framework and DFS and launching various mapreduce applications for executing sensor data uploading, retrieving, flushing, monitoring, and updating configurations.

3. Distributed Sensor Node Management System Based on Mapreduce

In this section, we present an architecture of HDSM (Hadoop-based distributed sensor node management system) for managing distributed sensor nodes based on Hadoop mapreduce and distributed file system.

3.1. Key Features. HDSM has several key features as follows.

(1) HDSM offers an easier way to build distributed sensor node management system. HDSM uses basic architecture and components of Hadoop framework without correcting its functions or developing specific component for manipulating data transfers or communications. By integrating multiple sensor nodes with limited hardware resources via network and Hadoop technology, HDSM builds a huge logical sensor node cluster on distributed sensor network easily.

(2) HDSM offers automated management of the whole process for integrating multiple sensor nodes, launching mapreduce applications, and managing communications and data transfers between DFS and other components via Hadoop mapreduce framework. So it can save costs for building and maintaining management system for distributed sensor nodes.

(3) HDSM can focus on analyzing collected sensing data without being concerned about fault failure, since Hadoop has internal mechanism for handling system failure.

(4) HDSM can easily update software of sensor nodes such as sensor device drivers, firmware, data format, and node configurations by launching exclusive mapreduce application without much effort for manipulating each of sensor nodes in complex progress.

3.2. Architecture. HDSM consists of node control manager (NCM), Hadoop master, and slave. NCM and Hadoop master function as a HDSM manager which controls the overall operations of HDSM. A sensor network is considered as a large scale mapreduce cluster. Hadoop slave is deployed on each of sensor nodes. It consists of Tasktracker and Datanode and interacts with the Hadoop master which is composed of Jobtracker and Namenode. NCM manages and controls Hadoop master by using Hadoop commands and mediates interaction between HDFS and external systems such as repository or data consumer services. Retrieving accumulated sensing data in HDFS and flushing HDFS are done by executing Hadoop commands. Particularly, it offers effective way for retrieving of big data accumulated on sensor node cluster since data traffic is spread across all sensor nodes (Datanodes) in reading operation of HDFS. NCM requests Hadoop master to execute various operations including uploading sensing data to HDFS, flushing HDFS, and patching sensor node firmware or configurations by launching proper mapreduce applications. In each operation, a single map task is allocated on each sensor node, respectively, by assigning a number of map tasks as number of total sensor nodes, since Hadoop scheduler tends to allocate tasks to idle worker nodes which can afford to be assigned new tasks [2]. Therefore, all map tasks process their role independently on sensor nodes in parallel. Figure 2 shows an overall architecture of HDSM.

The overall operations of HDSM are described below as shown Figure 3.

(1) NCM requests Hadoop master to launch "Uploader" mapreduce application to make all sensor nodes upload their local sensing data to HDFS. Uploader application has only mapper class without reducer and creates a number of map tasks identical to the number of sensor nodes. Each map task is allocated and executed on each node and copies its local sensing data to HDFS with distinguishable file naming. Each local sensing data is divided into a single or multiple data chunk and saved by Datanode in HDFS. In this way, each sensor node uploads its local data to its Datanode, and location of all data is identified by Namenode. After uploading to HDFS, sensing data can be ready for retrieval by Hadoop master and its clients.

(2) NCM requests Hadoop master to retrieve uploaded sensing data from HDFS and store them to local disk, other repository servers, or data consumer services. All accumulated sensing data is read from multiple Datanode at a time periodically by reading multiple chunks from multiple Datanodes using Hadoop command.

(3) NCM requests Hadoop master for flushing HDFS storage to retain available disk space of sensor nodes due to the limited storage size of sensor node.

(4) NCM requests Hadoop master to update advanced sensor device driver or sensor node configurations and launch "Updater" mapreduce application. Updater application has no reduce stage, and the number of map tasks is identical to that of sensor nodes. NCM provides multiple commands lines to map task, which in turn executes them as child tasks such as updating sensor driver or firmware codes to newer version, changing data formats, editing sensor node configurations, or even executing commands such as reboot, calibrating, and initializing of sensor devices.

HDSM performs the above operations periodically according to the NCM configuration. Additionally, it offers a flexible way for users to add more operations by developing their corresponding mapreduce applications such as simple data analyzers or data converters. If computing power of sensor nodes is not enough, user can execute data migration to other repositories for back up, or data processing systems for high performance data analysis.

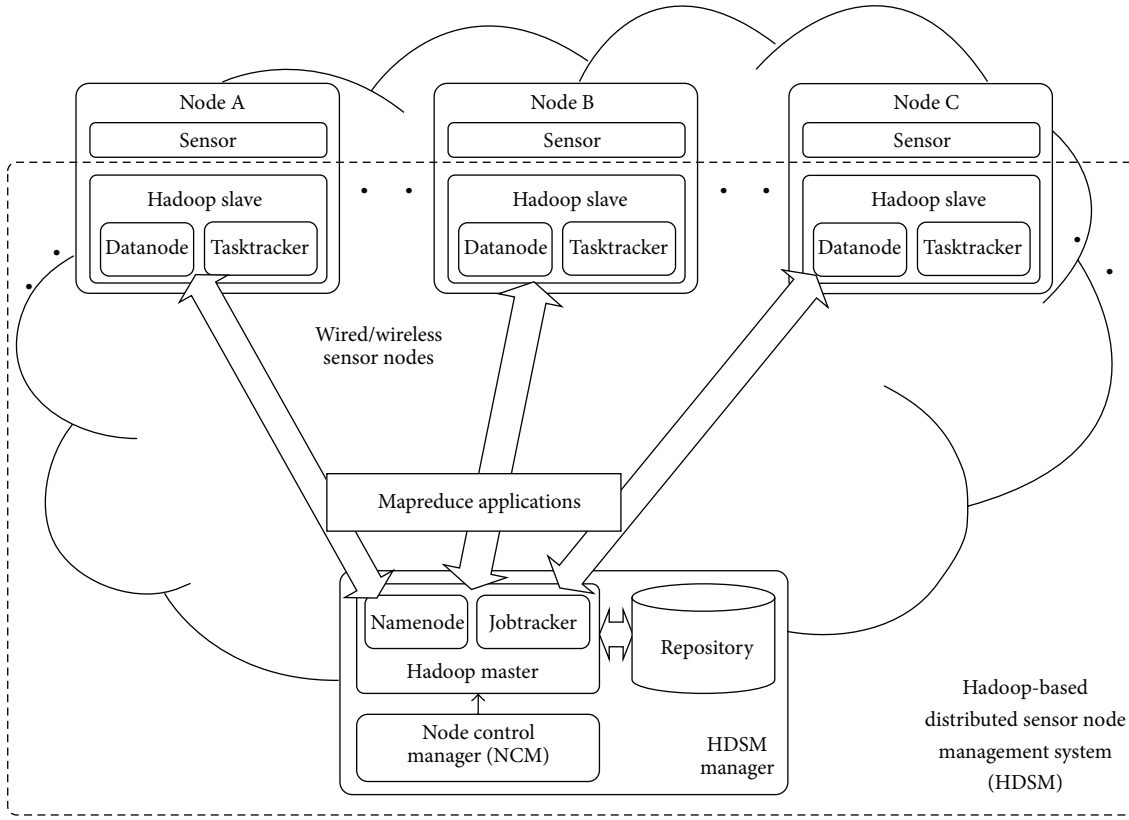

Figure 2: An overall architecture of HDSM.

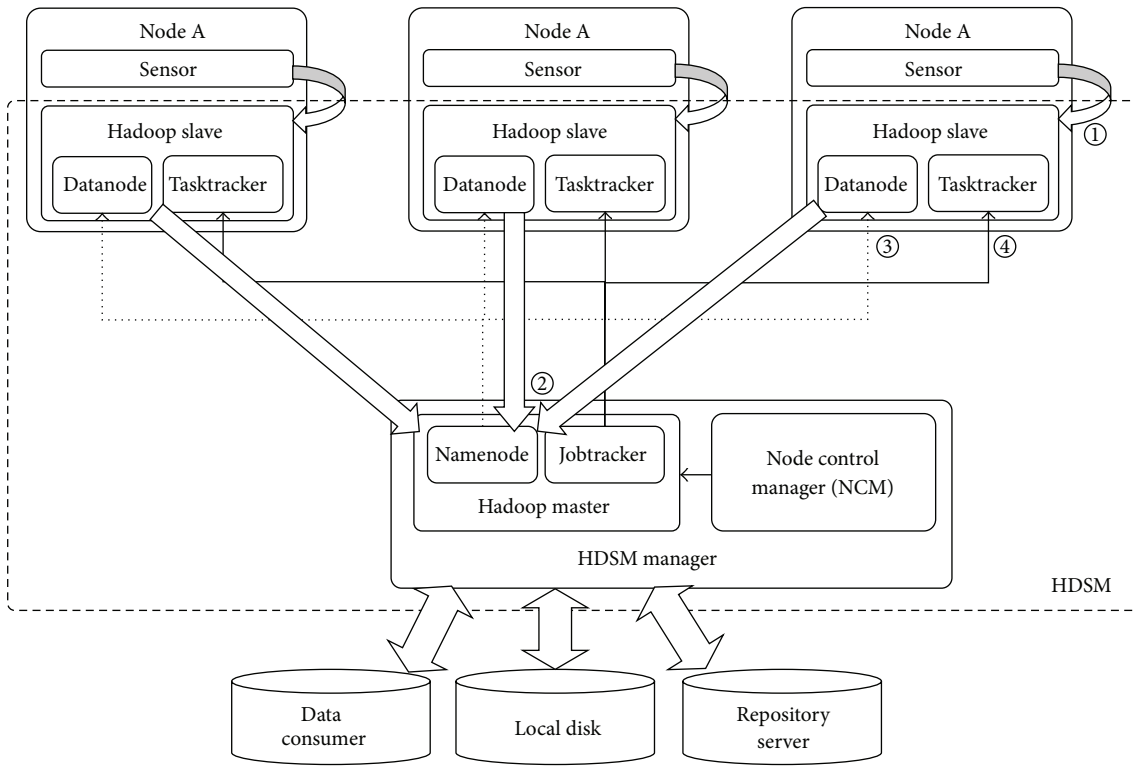

Figure 3: The key operations on HDSM requested by NCM. (1) Uploading local sensing data to HDFS by launching mapreduce application. (2) Retrieving accumulated sensing data from HDFS by using Hadoop command. (3) Flushing HDFS storage to retain available disk space by using Hadoop command. (4) Updating sensor node configurations by launching mapreduce.

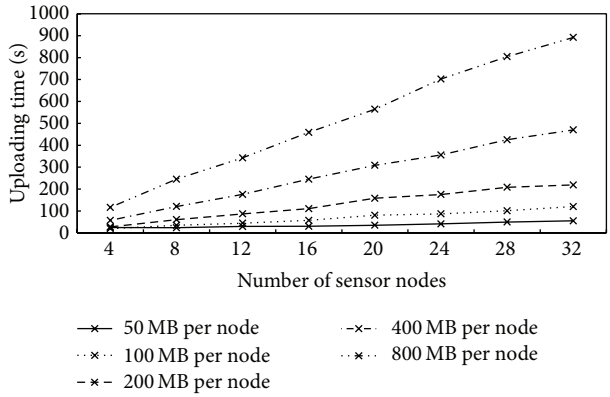

Figure 4: Average time of uploading sensing data with respect to the number of sensor nodes with various data sizes.

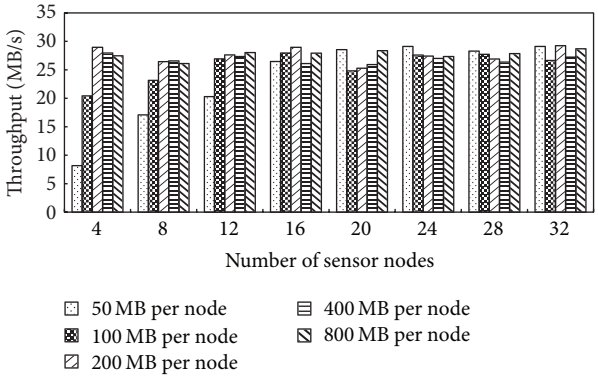

Figure 5: Average throughput of uploading sensing data with respect to the number of sensor nodes with various data sizes.

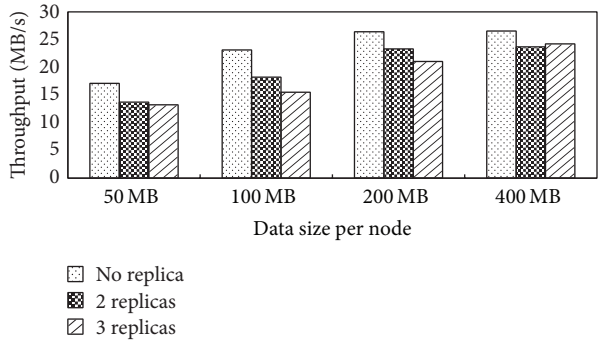

Figure 6: Average throughput of uploading data on 8 nodes with respect to different data size with various replication levels.

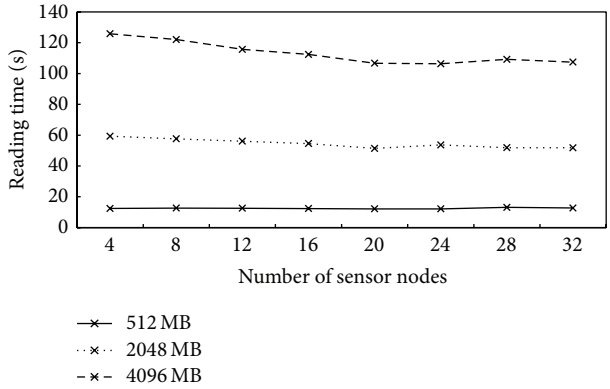

Figure 7: Average time of retrieving sensor data from HDFS with respect to number of sensor nodes with various data sizes.

4. Experiments and Results

We have implemented a prototype of HDSM and several mapreduce applications for key operations using Java programming language and Hadoop API [4]. HDSM is deployed on several sensor nodes, each of which has wired webcam, Core2Duo 2.33 GHz processor, 2 GB memory, and 320 GB disk. Each sensor node which imitates DVR (digital video recorder) for CCTV records video data (sensing data) captured by camera and saves it to its local disk storage periodically. Video files have various sizes from 50 MB to 800 MB depending on recording time. Unique IP address is assigned to each sensor node, and all sensor nodes are connected to HDSM manager via gigabit Ethernet.

We evaluate the I/O performance of HDSM for collecting and retrieving sensor data by measuring its processing time and throughput in Figure 4. It shows the average time of uploading sensing data with various sizes to HDFS on several sensor nodes. Block size of data chunk is 64 MB, and there is no replication in this experiment. Increase of uploading time depends on the number of sensor nodes, since amount of collectable sensor data in a period is proportional to the number of sensor nodes. However, average throughputs are relatively stabilized, regardless of number of sensor nodes or data size as shown in Figure 5. Mean of throughputs in all cases is 26.230 MB/sec, and standard deviation is 3.80.

Also, we measure the performance of uploading operation with various replication levels in Figure 6. It shows average throughput of uploading data on 8 nodes with various replication levels. The throughput of uploading data is decreased with increasing replication level, since, although data replication of Hadoop is performed to provide fault tolerance and availability, it may cause more data traffic on network and lower I/O performance of HDFS.

We evaluate retrieving performance of sensor data accumulated on HDFS in Figures 7 and 8. Block size of data chunk is 64 MB, and there is no replication in this experiment. Figure 7 shows that average retrieving time decreases as the number of sensor nodes increases. Figure 8 shows that the average throughput of retrieving data is higher than that of uploading operation and related to the amount of data size. Also, it shows that mean of throughput in all cases is 37.79 MB/sec, but throughput decreases with increasing data size.

In real environment, sensor node may have lower performance than the one in our experimental environment. So, we evaluate uploading time on low performance sensor nodes by using virtual machines to compare with high performance ones additionally as shown in Figure 9. Virtual machine has single core 2.13 GHz processor, 512 MB memory, and 10 GB disk and simulates cheaper low performance DVR

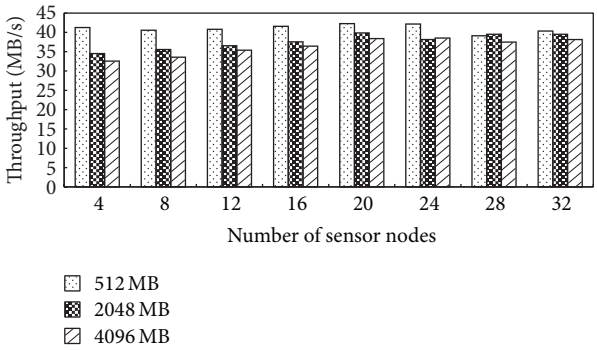

Figure 8: Average throughput of retrieving sensor data from HDFS with respect to number of sensor nodes with various data sizes.

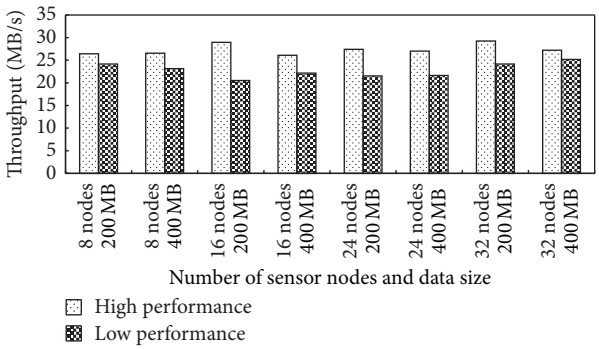

Figure 9: Average throughput of uploading sensor data with respect to number of sensor nodes and various data sizes on two types of node with low and high performance.

devices lower than $200 dollars. Experimental conditions are identical to the ones in Figure 5. Figure 9 shows that the mean of throughputs is 22.804 MB/sec on low performance nodes, while 27.362 MB/sec on high performance nodes with 16.65% performance degradation.

5. Conclusion

In this paper, we have presented HDSM, the Hadoop-based distributed sensor node management system for distributed sensor node management scheme, by exploiting Hadoop mapreduce framework and distributed file system and launching various mapreduce applications for executing sensor data uploading, retrieving, flushing, monitoring, and updating configurations. It offers various efficient ways for collecting sensor data and managing multiple sensor nodes by launching specific mapreduce applications on sensor nodes which upload data of sensor nodes to DFS and retrieve sensor data periodically from DFS. Additionally, it provides a flexible management scheme for sensor node by reconfiguring a firmware or updating configurations and data formats of sensor nodes based on mapreduce framework. Also, it achieves a transparent and fault-tolerant architecture by exploiting various crucial features of Hadoop. Our experimental results show that it has efficient and stable performance. As a future work, we are planning more researches for additional improvements and functionality with fault tolerance and high throughput.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2013-H0301-14-1001) supervised by the NIPA (National IT Industry Promotion Agency), Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the MSIP (NRF-2013043678), and Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research & Development Program (R2012030096).

References

[1] K. Chorianopoulos, "Collective intelligence within web video," *Human-Centric Computing and Information Sciences*, vol. 3, no. 10, pp. 1–16, 2013.

[2] T. White, *Hadoop: The Definitive Guide*, O'Reilly Media, 2012.

[3] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in *Proceedings of 6th Symposium on Operating Systems Design and Implementation (OSDI '04)*, pp. 137–149, 2004.

[4] Apache Hadoop, http://hadoop.apache.org/.

[5] J. Ekanayake, H. Li, B. Zhang et al., "Twister: a runtime for iterative MapReduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10)*, pp. 810–818, June 2010.

[6] S. Ghemawat, H. Gobioff, and S. Leung, "The google file system," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 29–43, October 2003.

[7] C. Jardak, J. Riihijärvi, F. Oldewurtel, and P. Mähönen, "Parallel processing of data from very large-scale wireless sensor networks," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10)*, pp. 787–794, June 2010.

[8] Y. Bao, L. Ren, L. Zhang, X. Zhang, and Y. Luo, "Massive sensor data management framework in cloud manufacturing based on Hadoop," in *Proceedings of the 10th IEEE International Conference on Industrial Informatics (INDIN '12)*, pp. 397–401, Beijing, China, July 2012.

[9] H. Chen, S. Jiang, and G. Chen, "Design and implement wireless temperature sensor Network based on distributed platform," *Applied Mechanics and Materials*, vol. 303–306, pp. 114–119, 2013.

[10] P. Zhang, H. Sun, and Z. Yan, "A novel architecture based on cloud computing for wireless sensor network," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE '13)*, pp. 472–475, 2013.

[11] B. Yu, R. Sen, and D. H. Jeong, "An integrated framework for managing sensor data uncertainty using cloud computing," *Information Systems*, vol. 38, no. 8, pp. 1252–1268, 2013.

[12] S. Yerva, H. Jeung, and K. Aberer, "Cloud based social and sensor data fusion," in *Proceedings of the 15th International Conference on Information Fusion (FUSION '12)*, pp. 2494–2501, Singapore, July 2012.

[13] Y. Pan and J. Zhang, "Parallel programming on cloud computing platforms—challenges and solutions," *Journal of Convergence*, vol. 3, no. 4, pp. 23–28, 2012.

[14] B. Yu, A. Cuzzocrea, D. Jeong, and S. Maydebura, "On managing very large sensor-network data using bigtable," in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 918–922, Ottawa, Canada, May 2012.

[15] F. Xhafa, "Processing and analysing large log data files of a virtual campus," *Journal of Convergence*, vol. 3, no. 3, pp. 1–8, 2012.

[16] X. Kui, Y. Sheng, H. Du, and J. Liang, "Constructing a CDS-based network backbone for data collection in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 258081, 12 pages, 2013.

[17] T. W. Wlodarczyk, C. Rong, C. I. Nyulas, and M. A. Musen, "An efficient approach to intelligent real-time monitoring using ontologies and Hadoop," in *Proceedings of the International Conference on High Performance Computing and Simulation (HPCS '10)*, pp. 209–215, Caen, France, July 2010.

[18] Apache Flume, http://flume.apache.org/.

[19] Honu, https://github.com/jboulon/Honu/.

[20] Amazon EC2, http://aws.amazon.com/ec2/.

[21] Amazon Elastic MapReduce (Amazon EMR), http://aws.amazon.com/elasticmapreduce/.

[22] Apache Hive, http://hive.apache.org/.