International Journal of Multimedia and Ubiquitous Engineering
Vol. 9, No. 12 (2014), pp. 61-72
http://dx.doi.org/10.14257/ijmue.2014.9.12.06

HDSW: Semantic Sensor Network System Based on Hadoop

Dongfeng Wang, Xiaoming Zhang* and Hongbin Gao

School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang, 050018
wangdongfenghb@gmail.com, zxm1975@gmail.com, gao_hb@hebust.edu.cn[1]

Abstract

Ontology is used to annotate sensor network data and enhances its semantic, which is the motivation of Semantic Sensor Network (SSN). With the development of SSN, massive RDF datasets have been produced, so SSN is up against a big challenge now, i.e., the integration and application of data. In this paper, we present a Hadoop-based Semantic Sensor Network System (named HDSW) to achieve efficient data management for sensor data in RDF. The architecture of HDSW is composed of sensor network layer, persistent layer, Hadoop layer, function layer and UI layer. As one of the core component of HDSW, the RDF storage module is described in detail. The RDF storage module uploads RDF dataset to Hadoop by MapReduce, and stores BigRDF file by key/value mode. Then BigRDF is mapped to HBase table schema by using MapReduce. Finally, we evaluate HDSW system, and the experimental results show that it has good scalability and efficiency.

Keywords: *Semantic Sensor Network, Hadoop, HBase, RDF, Cloud Computing*

1. Introduction

Sensor network has been widely used in all walks of life, which brings great convenience to human life. The sensor network data has the characteristic of heterogeneity, magnanimity and streaming, but it faces the problems of data storage and management. Now W3C has proposed SSN (semantic sensor network) [1] which makes use of ontology to solve data interconnection. With the development of sensor network, massive sensor data have been generated and cloud computing brings the hope for solving the problem of massive sensor data management at the same time. Hadoop [2] is the middleware and uses the virtualization technology of server deploying cloud computing platform. Therefore, we propose a system of distributed semantic sensor network based on Hadoop named HDSW which is to store and manage the data of semantic sensor network. We summarize the main contributions of this work as follows:

- The architecture of HDSW is composed of sensor network layer, persistent layer, Hadoop layer, function layer and UI layer, and each layer plays different role in HDSW system.
- MapReduce [3] and HBase [4] are used for massive semantic sensor data storage.
- We implement the RDF storage module, which uploads RDF files to HDFS by MapReduce, and stores BigRDF file by key/value mode. Then BigRDF is mapped to HBase table schema by using MapReduce.
- We do experiments for HDSW system. The results show our system has the characteristics of feasibility, scalability and efficiency.

[1] *Corresponding Author

ISSN: 1975-0080 IJMUE

The rest paper is organized as follows. Section 2 describes HDSW system and its architecture. Section 3 describes the storage module of RDF. Section 4 describes the experiments and results. Section 5 describes the related work. Finally, the summary of this paper and some possible future works are discussed in Section 6.

2. HDSW System

2.1. Overview of HDSW

In order to solve the semantic sensor network distributed management, we propose HDSW system which mainly includes data integration and sensor data applications. We use the SSN ontology [5] that come from W3C Semantic Sensor Network Incubator Group, and make use of RDF to store sensor data. Kno.e.sis [6] has provided LOD (linked open data) [7] for Standardized ontology (*e.g.*, SSN ontology) which comes from weather data at Mesowest. In order to implement data integration, we create a novel schema to store LOD by HBase. In brief, HDSW is a kind of cloud computing platform based on Hadoop that has characteristics of distributed system, so it works for sensor data applications.

2.2. HDSW Architecture

Figure 1 shows the architecture of HDSW， which contains sensor network layer, persistent layer, Hadoop layer, function layer and UI (User Interface) layer.

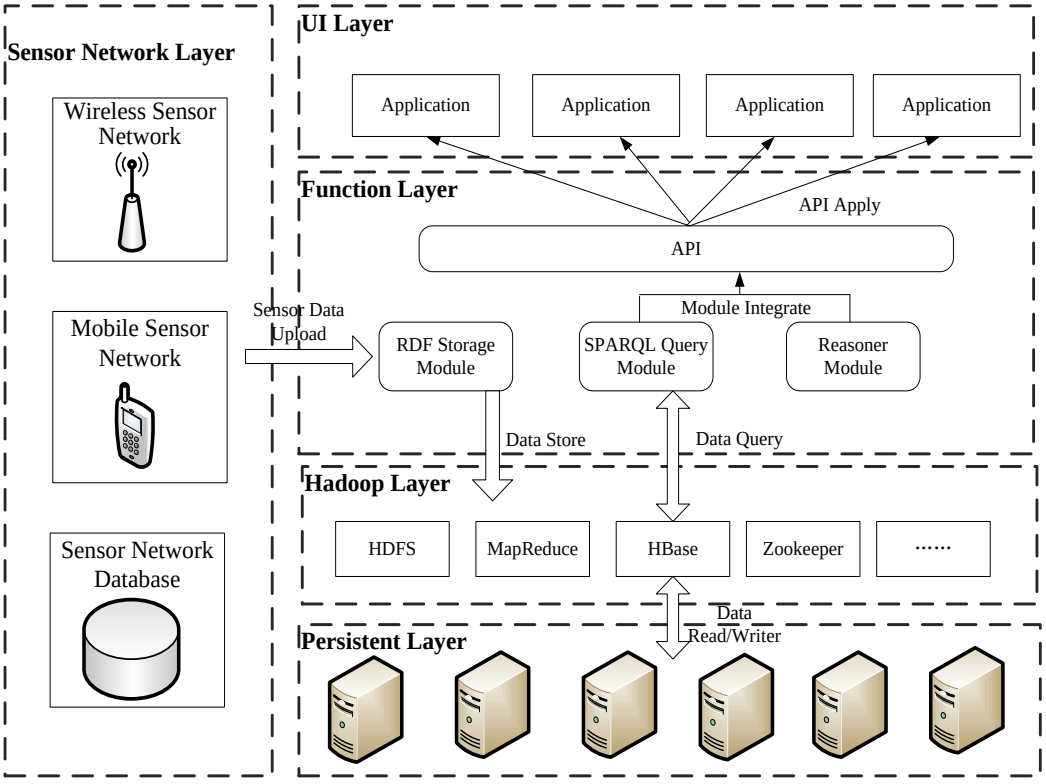


Figure 1. HDSW Architecture

The sensor network layer has a large number of sensor devices in the sensor network whose task is to collect sensor information and observation data. This layer maps heterogeneous data of sensor network to the form of RDF datasets by the use of SSN ontology. Because the Semantic Sensor Network has realized the technology of data acquisitions, this layer isn't acted as our research emphasis.

The Hadoop layer is the middleware of HDSW, including HDFS (Hadoop File System), MapReduce, HBase, Zookeeper, Pig, Hive, Mahout and so on. HDFS implements the management of distributed files on Hadoop, which mainly include the log files and data files. MapReduce is a kind of parallel computing model to achieve efficient data processing. HBase is a kind of NoSQL Database for the storage and management of massive data. Zookeeper [8] serves as the role of system coordinator that efficiently implements the load balance of distributed system. Then, this layer is built on these software tools and provides support and security for function layer.

The persistent layer consists of a large number of server storage devices for storing huge amounts of data permanently. The data of this layer is managed and scheduled by Hadoop layer uniformly.

The function layer is the core component of HDSW system, which has implemented the requirements of SSN. This layer consists of the RDF storage module, SPARQL query module, Reasoner and HDSW API (Application Programming Interface). The aim of the RDF storage module is to store the massive RDF dataset of SSN based on MapReduce computation model and HBase NoSQL Database. Since MapReduce is quite well suited for writing stencil-style data of parallel programs, we implement the manipulation of RDF datasets by MapReduce efficiently in HDSW. HBase Schema is designed to store RDF dataset logically, which is indeed stored in the persistent layer in the form of block eventually. SPARQL is a semantic query language for RDF, but it can't query HBase files directly. Therefore we create SPARQL query module that implements the query function of this system and use the Hadoop ecosystem programming tools (*e.g.*, Pig, Hive) to implement semantic query function on HBase files. While the data of sensor network is real-time, we need to implement continuous query function for HBase in HDSW. Reasoner, which implements inference based on axioms or rules, can make the information from HDSW API more accurately. HDSW API provides the data service for UI layer, which is implemented by invoking the query module and Reasoner. HDSW API provides a series of standardized operations which are convenient to the development of specific function in application.

The UI layer provides programs for specific areas or problems and satisfies the requirements by users. Users can develop application programs of distributed semantic sensor network system by HDSW API.

2.3. HDSW Process

The process of HDSW mainly includes data integration and data applications for SSN. The process of data integration for SSN is depicted in Figure 2.

1) We collect data from the Sensor Network and store it into Sensor Data.
2) Sensor data is transferred into RDF Data according to SSN Ontology. Then, these heterogeneous sensor data are unified into RDF dataset.
3) The RDF data are mapped to HBase by RDF Storage Module.
4) In this step, the data of HBase is stored into HDFS by the form of Hfile.

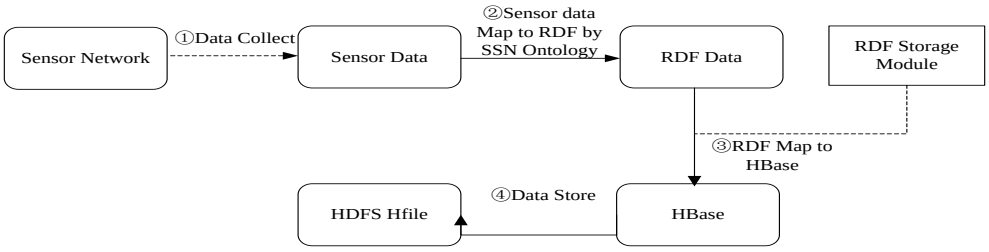


Figure 2. Data Integration Process

The process of data applications for SSN is depicted in Figure 3.

1) Users send request to the application including keywords and contents of the request.
2) The application sends task to SPRQAL Query Module and Reasoner Module respectively.
3) SPRQAL Query Module sends task of query to HBase Database, including continuous query conditions and fields. Then, Reasoner Module sends task to HBase Database.
4) HBase makes use of HMaser/HRegion to read Hfile, including primary key, and column family data.
5) HDFS gets corresponding data back to HBase Database by use of MapReduce.
6) HBase gets query results back to the SPARQL Query Module and Reasoner Module by HBase API.
7) HDSW API checks the task results from SPARQL Query Module and Reasoner Module. If the results are correct, just returning to the application and displaying to the user. Otherwise, HDSW will continue to search.
8) Finally, the query result returns from application to end user.

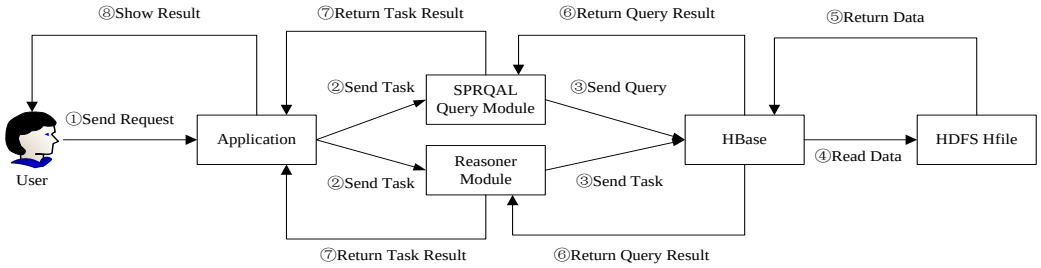


Figure 3. Process of Data Applications

3. RDF Storage Module

3.1. Overview of the Module

RDF storage module, which is based on MapReduce and HBase, is mainly designed for the storage of massive RDF data from SSN. For example, a temperature sensor device produces RDF data which describes spatial (*e.g.*, Beijing city), temporal (*e.g.*, 8:10AM GMT, 10-05-2014) and unit (*e.g.*, Celsius) attributes. Due to the temporal attribute of RDF data from SSN, it is a kind of stream data. In order to store and manage this kind of stream data, timestamp should be properly utilized. In addition, the design of HBase table schema and how to map the RDF sensor data to the HBase table are the main issues should be considered in RDF Storage Module.

In this paper, we present a kind of HBase table schema to store massive RDF of SSN, and this schema is named as HRDF. In order to achieve the effective management and operation of

the RDF data, we create an index mechanism that can improve the efficiency of query [9]. HRDF schema contains {Row-key(S), Column Family (I_P, I_O, I_{SO}, I_{SP}, I_{PO}), TS}, where Row-key represents S (subject attribute of RDF), Column Family represents P (Predicate attribute of RDF), O (Object attribute of RDF), SO, SP, PO and TS represents temporal attribute of RDF.

MapReduce computation model is used to map stream RDF to HRDF. RDF dataset of SSN contains a lot of small files, while Hadoop is more suitable for processing large files than a large number of small files. Therefore, RDF data files are merged into BigRDF that is one large file storing in HDFS. MapReduce technology deals with BigRDF in the form of key/value, so BigRDF is stored with key/value in our work. Finally, BigRDF is mapped into HRDF based on MapReduce and HBase API.

3.2. RDF Storage Module Process

RDF storage module is the process of RDF mapping to HRDF. We take advantage of linked open data by Kno.e.sis [6] to introduce the process of storage (In this version of RDF storage module we use LOD dataset, and rdf dataset based on SSN ontology will be used in the next version). Figure 4 describes temperature observation data of System_A01 on August 9, 2004 12:15:00.

```
sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00
    a       weather:TemperatureObservation ;
    om-owl:observedProperty       weather:_AirTemperature ;
    om-owl:procedure sens-obs:System_A01 ;
    om-owl:result "99.0"^^xsd:float , weather:fahrenheit .
    om-owl:samplingTime sens-obs:Instant_2004_8_9_12_15_00 .
```

Figure 4. Sample of RDF Triple

The process of RDF storage module is depicted in Figure 5.

1) In the phase of split, RDF data is distributed into the map function randomly.
2) We make use of the string matching algorithm to identify stream RDF triples' properties. Then S of RDF is acted as the *key*, and P, O of RDF as the *value*. Map function outputs data in form of *key/value*, *e.g.*, sens-obs: Observation_AirTemperature_A01_2004_8_9_12_15_00/P*1*a (where P is the abbreviation of the predict for a RDF statement, * is a linking identifier, 1 is an identifier for the relation of P and O for the same RDF triples, and the letter *a* is the name of the corresponding predict).
3) In the phase of shuffle, MapReduce orders data by key/value pairs, and assigns the data for the same *key* to the Reduce function.

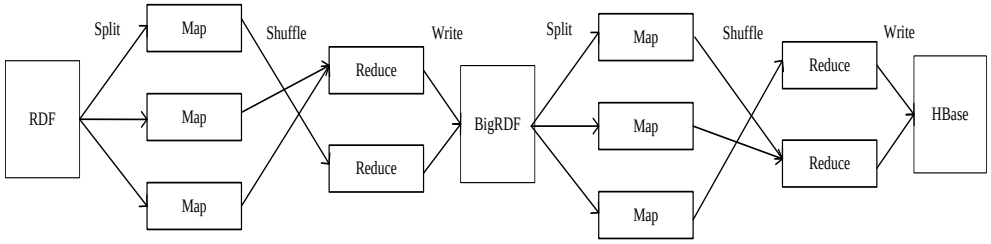


Figure 5. Process of RDF Storage Module

4) In the phase of Reduce, we receive the key/value data from shuffle and store key/value pairs into BigRDF by use of write function which comes from HDFS API. For example, the content of the BigRDF is shown in Figure 6.

Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	P*1*a
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	O*1*weather:TemperatureObservation
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	P*2*om-owl:observedProperty
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	O*2*weather:_AirTemperature
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	P*3*om-owl:procedure
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	O*3*sens-obs:System_A01
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	P*4*om-owl:result
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	O*4*"99.0"^^xsd:float
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	O*4*weather:fahrenheit .
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	P*5*om-owl:samplingTime
Sens-obs:Observation_AirTemperature_A01_2004_8_9_12_15_00	O*5*sens-obs:Instant_2004_8_9_12_15_00

Figure 6. Content of the BigRDF

5) MapReduce splits BigRDF data into the map function.
6) In the phase of map, we divide BigRDF into key-value pairs by tokenizer function and output key/value pairs to shuffle.
7) In the phase of shuffle, MapReduce orders data by key/value pairs, and assigns the data for the same key to the HBaseReduce function.
8) HBaseReduce creates HRDF connection using HBase API. Firstly we identify S, P and O from key/value. The variable key is the attribute of S, the variable value is the attribute of P and O. Secondly we identify P and O by the identifier from the key/value. If the value contains identifier P, we split value to array P[i] by tokenizer. P [0] is the identifier of attribute P, P[1] is the number for corresponding to the triple and P [2] is the value of attribute. We identify the timestamp attribute by temporal attribute of SSN ontology which stores timestamp into variable TS. Finally we store RDF attribute to HRDF by use of the function put (Key, P, P [0]: P [1], P [2], TS). The content of HRDF is shown in Table 1.
9) HBase divides HRDF data into blocks automatically with the form of Hfile storing in HDFS.

Table 1. Content of HRDF

Row-key	Timestamp	I_P	I_O	I_{SP}	I_{SO}	I_{PO}
sens-obs:Observation_AirTemperature_A01	1092024900	P:1=>A	O:1=>1*weather:TemperatureObservation	SP:a=>weather:TemperatureObservation	SO:weather:TemperatureObservation=>a	PO:a*weather:TemperatureObservation=>sens-obs:Observation_AirTemperature_A01
	1092024900	P:2=>Om-owl:observedProperty	O:2=>weather:_AirTemperature	SP:Om-owl:observedProperty=>weather:_AirTemperature	SO:weather:_AirTemperature=>Om-owl:observedProperty	PO:Om-owl:observedProperty*weather:_AirTemperature=>sens-obs:Observation_AirTemperature_A01
	1092024900	P:3=>om-owl:result	O:3=>"99.0"^^xsd:float	SP:om-owl:result=>"99.0"^^xsd:float	SO:"99.0"^^xsd:float=>om-owl:result	PO:om-owl:result*"99.0"^^xsd:float=>sens-obs:Observation_AirTemperature_A01
	.....	.....	……	……	……	…..

4. Performance Evaluation

4.1. Experimental Approach

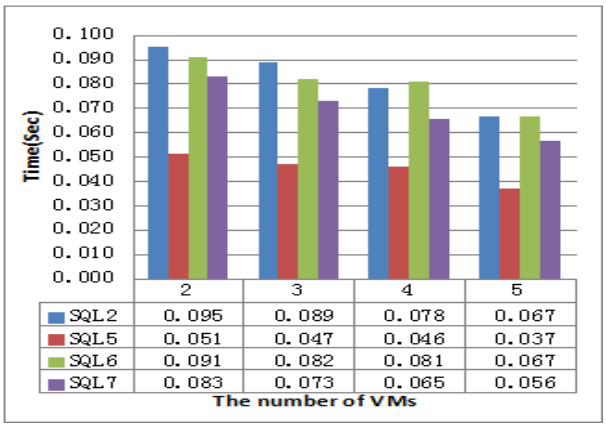
Experimental Dataset: Kno.e.sis has already provided massive RDF datasets of SSN [6, 7]. Datasets for sensors and sensor observations come from weather data at Mesowest. It contains descriptions of 20,000 weather stations and 160 million observations. The size of dataset is 1.7 billion triples.

Cluster Configuration: Our experimental equipment is cluster of 5 VM containers on a Dell PowerEdge R710 (Intel Xeon 5620*2, 16G ECC DDR3, 4*146G disks). Each container has 2-core CPU, 2G of RAM, 80G of storage space based on VM workstation 7.0. The cluster for our evaluation consists of a variable number of VM (2 to 5) in the role of the HDFS, MapReduce and HBase. Each VM runs 3 mappers and 3 reducers consuming 256MB of RAM. We use Hadoop 0.20.02 and HBase 0.92.03 respectively.

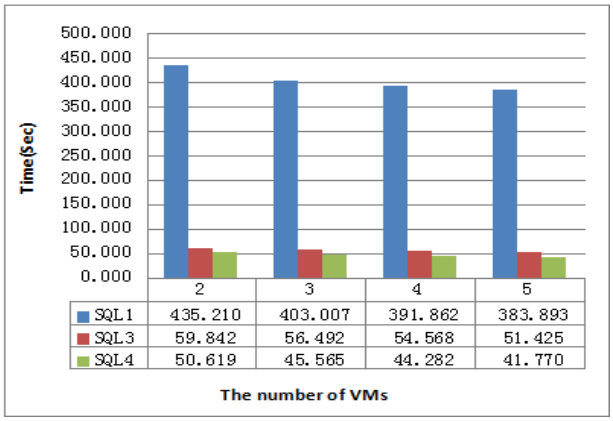
Experimental Task: SQL1 is the statement of query the entire table. S is the query condition of SQL2. P is the query condition of SQL3. O is the query condition of SQL4. S and P are query conditions of SQL5. S and O are query conditions of SQL6. S, P and O are query conditions of SQL7. Each task performs 6 times and eventually we take the average time of task as the final result for this experiment.

4.2. Experiment I

This experiment contains 10 million triples. We run 7 tasks on the different numbers of VM cluster and record the time of each task. Experiment result is shown in Figure 7. Firstly the time of SQL2, SQL5, SQL6 and SQL7 is under 1 second in Figure 7(a). The time of SQL1, SQL3, SQL4 are 383-400 seconds, 50 to 60 seconds, between 40 and 50 seconds respectively in Figure 7(b). The reason is that when the quantity of task results is small, the processing efficiency of HDSW is much higher. Otherwise it spends more time to scan entire table and display the query results. Secondly as the number of cluster nodes increasing, query time of each task decreases. The reason is that when the number of cluster nodes increasing, the parallel processing ability of Hadoop is much stronger. Therefore the system processing efficiency of the query task improves gradually. This experiment shows that HDSW has higher ability for parallel processing and scalability.



(a) Result of SQL2, SQL5, SQL6, SQL7

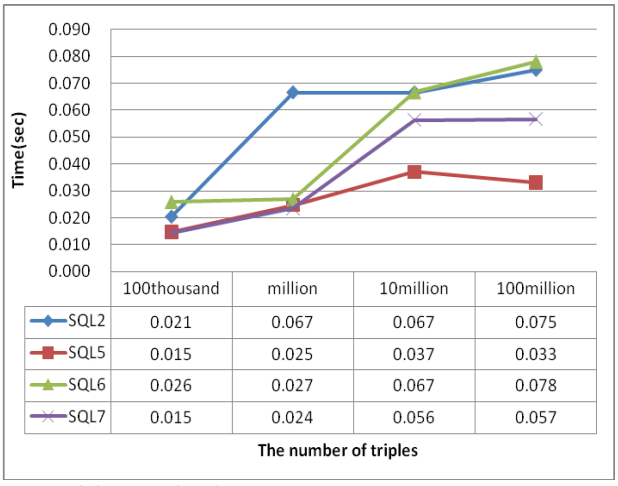


(b) Result of SQL1, SQL3, SQL4

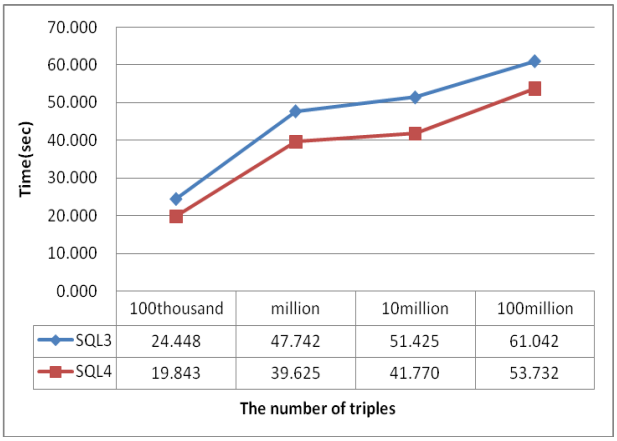
Figure 7. Result for Experiment I

4.3. Experiment II

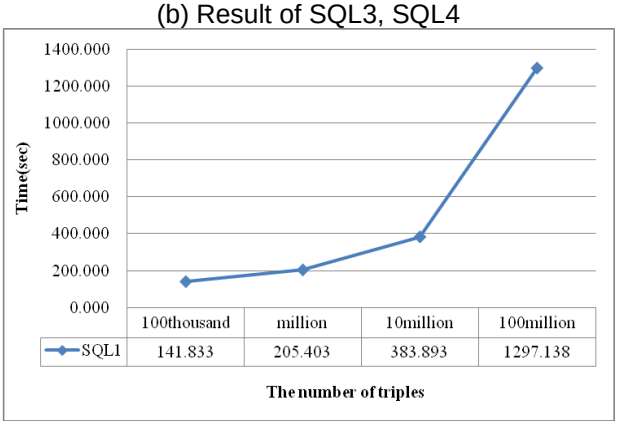
Experiment runs on cluster of 5 VMs. Each number of dataset is 100 thousand triples, 1 million triples, 10 million triples and 100 million triples respectively. We run 7 tasks on the different numbers of dataset and record the time of each task. The experimental results are shown in Figure 8. Firstly with the numbers of dataset increasing, the query time of SQL2, SQL5, SQL6 and SQL7 is under 1 second and rises slightly in Figure 8(a). Secondly with the numbers of dataset increasing, the query time of SQL3, SQL4 is between 20 to 60 seconds and increases only a little in Figure 8(b). Thirdly with the numbers of dataset increasing, the query time of SQL1 raises remarkably in Figure 8(c). The reason is that the program makes use of I/O for the displaying of query results, and with the number of dataset increasing, the query time rises remarkably. This experiment shows that storage and access ability of HDSW is extensible and the system of HDSW is feasible.



(a) Result of SQL2, SQL5, SQL6, SQL7



(b) Result of SQL3, SQL4



(c) Result of SQL1

Figure 8. Result for Experiment II

5. Related Work

The focus of HDSW is storage and integration for semantic sensor data. W3C Semantic Sensor Network Incubator Group has developed the SSN ontology which describes the sensor, observation and related concepts. In the meanwhile, some methods of RDF distributed storage and query has emerged. For example, Amazon provides a method of storage called Simple DB by the form of key/value on the foundation of AWS (Amazon Web Services). Simple DB has presented multiple index methods that improve the query efficiency of RDF datasets [10].

RDF Storage methods can be divided into two categories which are unstructured file storage and structured database storage. The first method uses distributed log files and establishes index files to store massive RDF data in the distributed system. The second method uses distributed database to create index files which can store and organize massive RDF data.

The method of unstructured file storage is mainly based on file in HDFS indexes that stores the RDF index files and datasets. It uses the HDFS API to query the index file, for example HadoopRDF [11] and H2RDF [12]. HadoopRDF [11] stores datasets to Hadoop in the form of n-triples, which creates index files based on predicate of triples. This method adopts the mechanism of greedy algorithm, which takes advantage of MapReduce implementing semantic query. This type of methods stores RDF in system by the form of triple, thus it only suits for the query of indexed attribute. H2RDF [12] uses MapReduce technology to design the storage and query algorithm for RDF based on cloud computing. This method stores RDF file in Hadoop system and uses HBase to establish 6 indexes model (S_PO P_SO, P_OS, O_PS, O_S and

S_OP). For example, P_OS is defined as <P, Count, Average> where P is the attribute of predicate, Count is the number of related subjects, and Average is the number of related objects. H2RDF combines hybrid indexing mechanism with hash algorithm implementing the mixed SPARQL query. This type of methods stores every RDF triple into a block file. If the size of block is set too huge, storage space isn't able to be fully taken advantage of.

The storage method based on database can be divided into two types. One is based on relational database storage, *e.g.*, the method of storing RDF by MySQL [13, 14]. The other one is based on the NoSQL database storage, *e.g.*, Jena-HBase [15]. The MySQL-based method [13, 14] combines Hadoop with the database MySQL to implement the storage of RDF dataset. R2RML is used to maps subject, predicate and object to primary key, the column of predicate and the column of object respectively. Although this type of methods have implemented the storage of RDF, it is hard to realize the query of semantics. Jena-HBase [15] uses Jena SDB to store and query RDF triples in HBase database. Jena-HBase combines hybrid indexing mechanism (*e.g.*, vertical index, horizontal index and hash index) to implement the storage of RDF. It uses the Jena framework and index mechanism to accomplish SPARQL query rapidly. This type of methods [16] can achieve extensible storage and efficient query.

The above methods provide some ways to store RDF data on Hadoop platform，but they don't specially designed for RDF data of SSN. In order to implement the semantic sensor network distributed management, we present HDSW system which mainly includes data integration and sensor data applications. The RDF data of SSN is stream data, so we utilize timestamp in HBase database to store the time of RDF data. Through the establishment of reasonable HBase data schema saves a lot of space for RDF storage and makes data query more convenient. We have realized the RDF data mapping to HBase by parallel processing of MapReduce technology.

6. Conclusions and Future Work

HDSW is designed for the storage and query for massive data of semantic sensor network based on Hadoop. The architecture of HDSW consists of sensor network layer, persistent layer, Hadoop layer, function layer and UI layer. The RDF storage module of function layer is detailed in this paper, and it store RDF file to HBase by using MapReduce. Experimental results have shown that our system has good scalability and efficiency.

In the future, we will explore the following areas of research. Firstly we will design semantic query module and API for HDSW efficiently so that we can employ our system in practical. Secondly we will extend our experimental evaluation with other query benchmarks, and HDSW will be compared with Jena-HBase, H2RDF *etc*. Additionally, we will also develop search engine for SSN applications.

Acknowledgments

This work is supported in Hebei Natural Science Foundation under Grant No.F2013208107, National Natural Science Foundation of China under Grant No.51271033 and 71271076, science fund project of Hebei Education Department under Grant No.QN20131138.

References

[1]. A. Sheth, C. Henson and S. S. Sahoo, “Semantic Sensor Web”, Internet Computing, IEEE, vol. 12, Issue 4, **(2008)**, pp. 78-83.
[2]. K. Shvachko, H. R. Kuang, S. Radia and R. Chansler, “The hadoop distributed file system”, Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, IEEE, **(2010)**, pp. 1-10.

[3]. J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool", Communications of the ACM, vol. 53, Issue 1, **(2010)**, pp. 72-77.
[4]. L. George, "HBase: the definitive guide", O'Reilly Media, Inc., Los Angeles, **(2011)**.
[5]. M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. David Kelsey, D. L. Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth and K. Taylor, "The SSN ontology of the W3C semantic sensor network incubator group", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 17, **(2012)**, pp. 25-32.
[6]. H. Patni, C. Henson and A. Sheth, "Linked sensor data", Collaborative Technologies and Systems (CTS), 2010 International Symposium on, IEEE, **(2010)**, pp. 362-370.
[7]. C. Bizer, T. Heath and T. Berners-Lee, "Linked data-the story so far", International journal on semantic web and information systems, vol. 5, no. 3, **(2014)**.
[8]. P. Hunt, M. Konar, F. P. Junqueira and B. Reed, "ZooKeeper: wait-free coordination for internet-scale systems", USENIX Annual Technical Conference, vol. 8, **(2010)**, pp. 9.
[9]. P. Cudré-Mauroux, I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, F. L. Keppmann, D. Miranker, J. F. Sequeda and M. Wylot, "NoSQL databases for rdf: An empirical evaluation", The Semantic Web–ISWC 2013, Springer Berlin Heidelberg, Los Angeles**,** vol. 8219, **(2013)**, pp. 310-325.
[10]. J. Murty, "Programming amazon web services: S3, EC2, SQS, FPS, and SimpleDB", O'Reilly Media, Inc., Los Angeles, **(2008)**.
[11]. J. H. Du, H. F. Wang, Y. Ni and Y. Yu, "HadoopRDF: A scalable semantic data analytical engine", Intelligent Computing Theories and Applications, Springer Berlin Heidelberg, Los Angeles, vol. 7390, **(2012)**, pp. 633-641.
[12]. N. Papailiou, I. Konstantinou, D. Tsoumakos, P. Karras and N. Koziris, "H2RDF+: High-performance distributed joins over large-scale RDF graphs", Big Data, 2013 IEEE International Conference on, IEEE, **(2013)**, pp. 255-263.
[13]. M. Hert, G. Reif, and H. C. Gall, "A comparison of RDB-to-RDF mapping languages", Proceedings of the 7th International Conference on Semantic Systems, ACM, **(2011)**, pp. 25-32.
[14]. C. Franke, S. Morin, A. Chebotko, J. Abraham and P. Brazier, "Distributed semantic web data management in HBase and MySQL cluster", Cloud Computing (CLOUD), 2011 IEEE International Conference on, IEEE, **(2011)**, pp. 105-112.
[15]. V. Khadilkar, M. Kantarcioglu and B. Thuraisingham, "Jena-HBase: A Distributed, Scalable and Effcient RDF Triple Store", Proceedings of the 11th International Semantic Web Conference Posters & Demonstrations Track, ISWC-PD, vol. 12, **(2012)**, pp. 85-88.
[16]. M. N. Vora, "Hadoop-HBase for large-scale data", Computer Science and Network Technology (ICCSNT), 2011 International Conference on, IEEE, vol. 1, **(2011)**, pp. 601-605.

Authors

Dongfeng Wang, he is born in 1991. He is master in School of Information Science and Engineering at Hebei University of Science and Technology. His main research interests include Semantic Sensor Network and Distributed System.

Xiaoming Zhang, he was born in 1975, received the PhD degree in computer application from University of Science and Technology Beijing of China in 2009, and the Master degree in computer application from Hebei University of China in 2002. Now, he is an associate professor in the School of Information Science and Engineering at Hebei University of Science and Technology, China. His main research interests include Semantic Web and domain-specific information integration.

Hongbin Gao, he is born in 1964. He is a professor and master supervisor in School of Information Science and Engineering at Hebei University of Science and Technology. His main research interests include IoT (Internet of things) and system integration.

