Journal of Information Hiding and Multimedia Signal Processing ©2016 ISSN 2073-4212
Ubiquitous International Volume **7**, Number **1**, January **2016**

Secondary Index to Big Data NoSQL Database - Incorporating Solr to HBase Approach

Bao-Rong Chang

Department of Computer Science and Information Engineering
National University of Kaohsiung
700, Kaohsiung University Rd., Nanzih District, Kaohsiung 811, Taiwan
brchang@nuk.edu.tw

Hsiu-Fen Tsai

Department of Marketing Management
Shu-Te University, Taiwan
59, Hun Shang Rd., Yen Chao, Kaohsiung County 824, Taiwan
soenfen@mail.stu.edu.tw

Hung-Ta Hsu

Department of Computer Science and Information Engineering
National University of Kaohsiung
700, Kaohsiung University Rd., Nanzih District, Kaohsiung 811, Taiwan
hushunghung@gmail.com

Received October, 2014; revised September, 2015

Abstract. *This paper introduces the combination of Big Data NoSQL database HBase and enterprise search platform Solr so as to tackle the problem of the secondary indexing function with fast query. In order to verify the effectiveness and efficiency of the proposed approach, the assessment using C-P ratio has been done for several competitive benchmark databases and the proposed one. As a result, our proposed approach outperforms the other databases and fulfills secondary indexing function with fast query in NoSQL database.*
Keywords: Big Data NoSQL Database, Secondary Index, HBase, Solr, Cost-Performance Ratio.

1. **Introduction.** Regarding big data storage[1, 2], the way of fast and easy data query is a concerned issue in NoSQL database. In general, NoSQL scheme[3, 4] is capability of supporting various data format to process the storage, yet it sacrifices the index searching function. HBase is of a NoSQL database as part of Hadoop ecosystem. It is known as the scheme of key-value and usually stores the results coming out of MapReduce execution. HBase features high scalability and high flexibility, delivering a high IO performance of big data. Solr is of a blazing fast open source enterprise search engine that can quickly create index and proceed with powerful full-text search. In this paper, we are able to combine HBase and Solr to enhance the secondary indexing function for HBase. After the success of this combination, we go for a series of stress test using several testing items and then make the performance comparison between the proposed one and the other benchmark databases. Finally, a cost effectiveness evaluation called C-P ratio has been done for a variety of databases. As a result, the assessment about C-P ratio will be analyzed and

discussed for all of databases mentioned in this paper.

2. **Combination of NoSQL database and enterprise search platform.** This paper studies how the combination of HBase and Solr runs in big data environment based on cloud computing platform. All of application programs were installed in a Linux-based operating system. Hbase is placed over Hadoop HDFS system. Thus HBase can be attached to Hadoop after the core parts of Hadoop have been installed in a physical machine such as MapReduce and HDFS. Solr can operate independently without any support from any other applications. With the corporation with Solr, Hbase can easily create index. On the other hand, Solr is able to provide GUI interface for users operation. The procedure to establish the combination of two applications can be listed as follows.

(1) Install Linux O/S on every host, connect them together via SSH and deploy JVM to every host to achieve a Linux cluster environment.
(2) Establish master and slave nodes, and start them up. Master node shall deploy Hadoop to slave nodes. This have Hadoop done in every host in a cluster environment [5, 6, 7].
(3) After deploying Hadoop and ZooKeeper to cluster, we need to confirm the start-up of Hadoop and ZooKeeper services. We are able to give jps instruction at terminal to check whether or not the services are running normally. After that, we establish HBase service [8, 9, 10, 11] within Hadoop.
(4) When the procedure #3 has done, web browser is used to view the start-up of Hadoop and HBase services. Key in http://localhost:50030, 50040, 50070, and 60010 to check each node if operating normally.
(5) Before we get Solr started, we need to modify the execution parameters in solrconfig.xml, which is a configuration file within./solr-version/examples/solr/collection1/conf/. We have to determine the solr whether or not setting input word string act as an index, content storage, and data format. Apache Solr needed http web-container to get it started, for example either Apache Tomcat or Apache Jetty. Here, we chose Jetty because of the default setting. After setting up, we key in "java -jar start,jar" to start up Solr in terminal. Finally, we got Solr's address, which is http://localhost:8983/
(6) Since HBase can't support automatically generated row-key, several big data files shall be modified in advance. We need to design a unique and complex rowkey which corresponds to a large number of rows (up to ten million rows.) In this study, we chose the American Yellow Page as data source. Our data combination is denoted "rowkey", "category", "shop-name", "telephone", "province", and "address" with a total of 6 columns. These data files have to translate into CSV format, and "," symbols are used to separate each column.
(7) The CSV file is uploaded to Hadoop file system, and these files are imported to HBase as full-text input via the special tool - "bulk load tool" [12]. We need to check the data integrity in HBase after data importing.
(8) Then, we use HBase output API and Apache HTTP API to transfer the document to Solr from HBase [13, 14, 15]. After the transmission, the indexes are created and the content is saved in disk in Solr. We can use web browser to check the amount of documents in Solr. Data in arow represent a document. We can use query function to search our keyword (Secondaryindex, or more) and reversely to search the primary index in Solr. We may able to applyfilter function to improve the precision of search results.

(9) After finishing the setup of the proposed system, we chose some other benchmarks to compare with the proposed one in the experiment. After the experiment, we are able to give a kind of assessment on those, for instance a cost effectiveness evaluation.

3. **System assessment.** In terms of the performance evaluation, we have initially tested the time for data read/write to a variety of databases, such as Apache HBase, Cassandra, Huawei HBase, Solandra, and Lily Project. Next, the time for data transfer to Solr from the databases mentioned above has to be recorded. Finally, the response time for the query function performed in Solr needed to be measured as well. In order to develop the assessment of the proposed approach, the necessitated equations are derived from first measuring a single datum access time for a certain database on Eq. 1, next calculating average access time based on a variety of data size on Eq. 2, then inducing a normalized performance index among the databases on Eq. 3, and finally resulting in a performance index according to different tests on Eq. 4. After that, evaluating the total cost of ownership is done for a certain period on Eq. 5 and then it turns out the cost-performance ratio on Eq. 6. In these equations we denote the subscript i the index of data size, j the index of database, and k the index of test category as well as the subscript s indicates a single datum. According to four tests on data write, data read, document transfer, and query/response to any of databases as mentioned above, first of all we have to measure a single datum access time taking a number of different data size as shown in Eq. 1, where $t_{s_{ijk}}$ represents a single datum access time, for a single run t_{ijk} stands for measured total time for a specific data size at a certain database, and N_{ik} means a specific data size. In Eq 2, $\bar{t}_{s_{ijk}}$ represents average time of a single datum access and w_i stands for the respective weight factor for $t_{s_{ijk}}$. A normalized performance index for a specific database at a certain test can be obtained as shown in Eq. 3, where $\bar{PI}_{jk}$ represents a normalized performance index. After that, we have evaluated the weighted average of normalized performance index and it turned out to be the performance index [16] for each database as shown in Eq. 4, where PT_j represents performance index, SF_1 stands for scale factor #1, W_k is the respective weight and $\bar{PI}_{jk}$ means a normalized performance index. In order to assess the cost effectiveness evaluation, we need to calculate total cost of ownership [17] in Eq. 5, showing the expenditure of money in the implementation of secondary indexing function for NoSQL database, where HC_a presents hardware cost, S_b stands for software cost, $RCAW_c$ means repairing cost after the warranty, DTC_d is down-time cost, and EUC_e explains extra upgrade cost. The monetary value of total cost of ownership may vary with location, market, and tax. Thus, a higher cost, for example, might be obtained in U.S., and a lower cost in Taiwan. In the system assessment, a typical cost effectiveness evaluation called C-P ratio has been introduced here to do the assessment in Eq. 6, where CP_{jg} is C-P ratio, SF_2 stands for scale factor #2, and TCO_{jg} means total cost of ownership.

$$t_{s_{ijk}} = \frac{t_{ijk}}{N_{ik}}, where\ i = 1, 2, ..., l, j = 1, 2, ..., m, k = 1, 2, ..., n \tag{1}$$

$$\bar{t}_{s_{ijk}} = \sum_{i=1}^{l} w_i \cdot t_{s_{ijk}}, where\ j = 1, 2, ..., m, k = 1, 2, ..., n, \sum_{i=1}^{l} w_i = 1 \tag{2}$$

$$\bar{PI}_{jk} = \frac{\frac{1}{\bar{t}_{s_{jk}}}}{\max\limits_{h=1,2,...,m}(\frac{1}{\bar{t}_{s_{hk}}})}, where\ j = 1, 2, ..., m, k = 1, 2, ..., n \tag{3}$$

$$PI_j = SF_l \cdot (\sum_{k=1}^{n} W_k \cdot \bar{PI}_{jk}), where\ j = 1, 2, ..., m, k = 1, 2, ..., n, SF_l = 10^2, \sum_{k=1}^{n} W_k = 1 \tag{4}$$

$$TCO_{jg} = \sum_a HC_a + \sum_b S_b + \sum_c RCAW_c + \sum_d DTC_d + \sum_e EUC_e, \quad where\ j = 1, 2, ..., m, g = 1, 2, ..., o \tag{5}$$

$$CP_{jg} = SF_2 \cdot \frac{PI_j}{TCO_{jg}}, where\ j = 1, 2, ..., m, g = 1, 2, ..., o, SF_2 = 10^4 \tag{6}$$

In order to examine the stability and reliability of NoSQL database secondary indexing function, a stress test of data retrieval in Solr has been taken in a big data environment. Technically speaking, this test generated up to 20 threads (20 windows) to respond 10 to 1000 queries and we had checked the latency (time interval) simultaneously. The key index in every query was different as shown in Fig. 1. Clearly the result would indicate the response time for the query in Solr and explain what the correlation between the amount of windows and the latency was found.

4. **Experimental results and discussion.** There are several experiments and a discussion presented in the following sub-sessions.

4.1. **Data transfer and data integrity checking.** In regard to implementation procedure as shown in Fig. 2, which indicated data transfer from HDFS to HBase and/or from HBase to Solr, there are risks of losing data during the transition. We have to verify the data integrity in HBase inner table and the amount of input documents in Solr. For examining HBase, we checked inner table using the command "scan table-name" in CLI as shown in Fig. 3. In Fig. 4, the document transfer from HBase to Solr has been done using the command in CLI. For examining Solr, we checked our input document amount in Solr using web interface as shown in Fig. 5. Furthermore, in terms of the performance evaluation, the time for data writing/reading in every database has been measured. Time for data transfer to Solr from every database has been recorded. Speaking of data import to HBase, we adopted a bulk-load tool with MapReduce computing to transfer the original file into HBase because this tool is capable of handling a large amount of data in the way of fast and smoothly transferring. For Solr, a program with specific port at Solr and designated HBase API has activated to quickly transfer documents to Solr from HBase where a java client to access Solr called Solrj have logged into the http server, that is Solr, to respond swiftly to the connection and on-line the document deliver to http server. This also demonstrates an efficiency way to realize a fast document transfer based on a client-server model for a huge amount of data. Alternatively, the other choice is that HBase coprocessor may launch a process to do the batch update frequently. However HBase coprocessor is not stable because it is still in the developing phase.

4.2. **Querying function and performance index.** Once the document transfer from Hbase to Solr, has done completely, the data are available in Solr and we could check the amount of document in Solr as shown in Fig. 5. In order to verify the secondary indexing function in the combination of HBase and Solr, we launched the query test in Solr as shown in Fig. 6, where we can check the information about the related operations on the

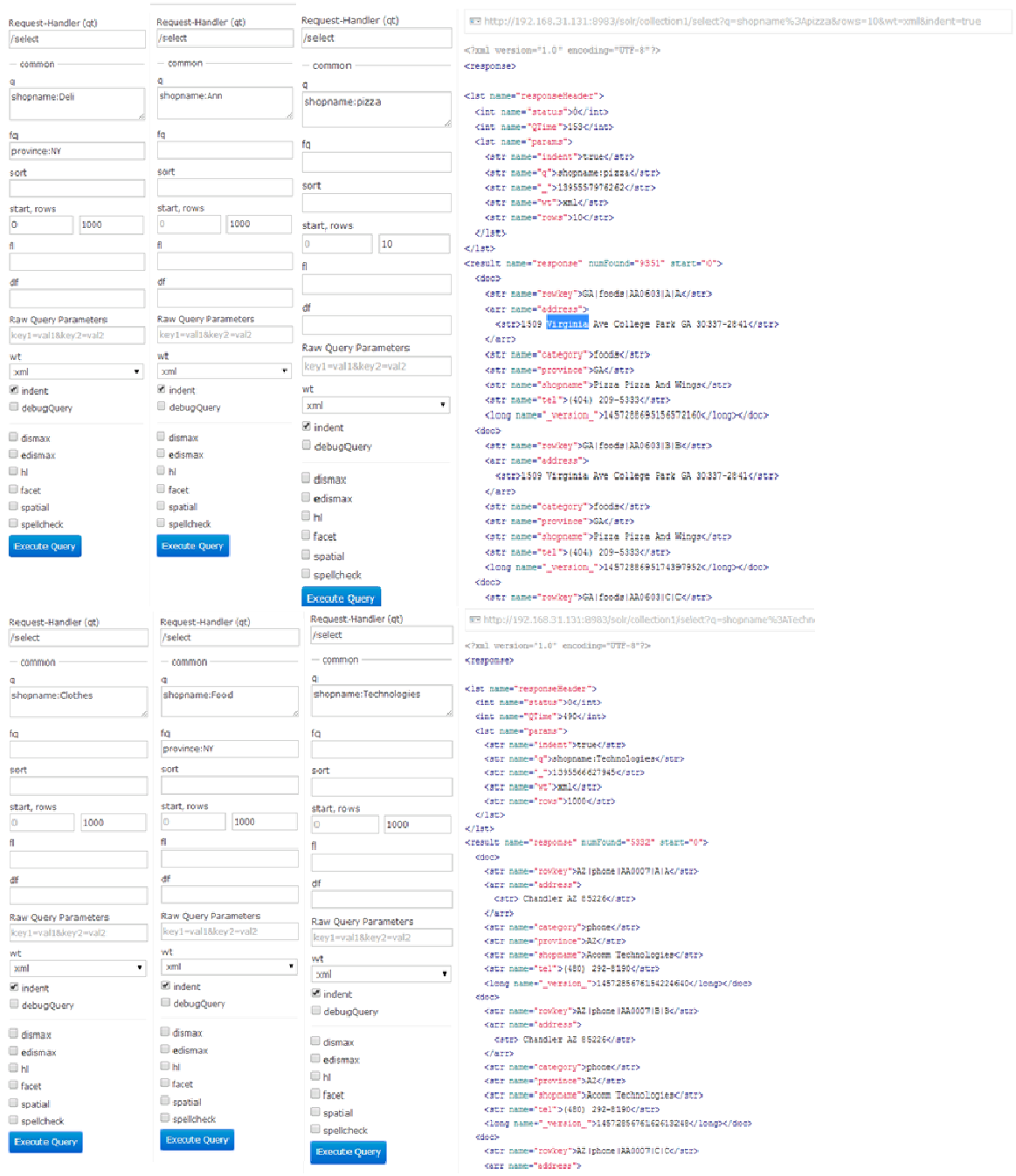

FIGURE 1. Latency under stress test for Solr (presenting 6 windows).

TABLE 1. Average time of a single datum access (unit: sec.)

Operation	HBase+Solr	Cassandra	Huawei HBase	Solandra	Lily Project
Write	0.000673563	0.011164768	0.0006735	0.011680475	0.00067955
Read	0.0025971	0.002637825	0.0026819	0.0027547	0.002608625
Transfer	0.011005125	0.01146665	0.011289425	0.011620125	0.011306375
Query	0.00000575	0.000136603	0.002028425	0.00007275	0.000041125

web. Solr provides normal search, filtering search, spatial search, and other more search

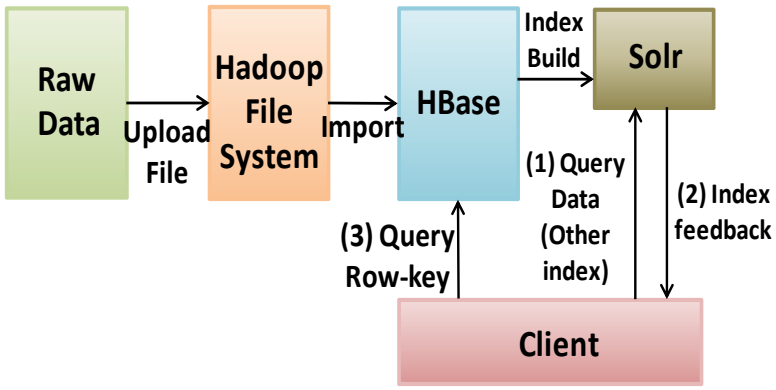

FIGURE 2. Implementation procedure.

```
92                                      column=addr:address, timestamp=1380369573826, value="58 Harvard Ave  Allston  MA 02134-1706"
92                                      column=category:s_c, timestamp=1380369573826, value="Restaurant"
92                                      column=province:s_prov, timestamp=1380369573826, value="MA Boston"
92                                      column=shop:s_n, timestamp=1380369573826, value="Boston Kaju Tofu Restaurant"
92                                      column=tel:s_ph, timestamp=1380369573826, value="(617) 208-8540"
93                                      column=addr:address, timestamp=1380369573826, value="50 Park Plz  Boston  MA 02116-400"
93                                      column=category:s_c, timestamp=1380369573826, value="Restaurant"
93                                      column=province:s_prov, timestamp=1380369573826, value="MA Boston"
93                                      column=shop:s_n, timestamp=1380369573826, value="Boston Park Plaza Restaurant"
93                                      column=tel:s_ph, timestamp=1380369573826, value="(617) 423-9560"
94                                      column=addr:address, timestamp=1380369573826, value="729 Boylston St # 3  Boston  MA 02116-2639"
94                                      column=category:s_c, timestamp=1380369573826, value="Restaurant"
94                                      column=province:s_prov, timestamp=1380369573826, value="MA Boston"
94                                      column=shop:s_n, timestamp=1380369573826, value="Boston Restaurant Group"
94                                      column=tel:s_ph, timestamp=1380369573826, value="(617) 587-9800"
95                                      column=addr:address, timestamp=1380369573826, value="315 Huntington Ave  Boston  MA 02115-4444"
95                                      column=category:s_c, timestamp=1380369573826, value="Restaurant"
95                                      column=province:s_prov, timestamp=1380369573826, value="MA Boston"
95                                      column=shop:s_n, timestamp=1380369573826, value="Boston Shawarma"
95                                      column=tel:s_ph, timestamp=1380369573826, value="(617) 670-0460"
96                                      column=addr:address, timestamp=1380369573826, value="Boston  MA 02101-0210"
96                                      column=category:s_c, timestamp=1380369573826, value="Restaurant"
96                                      column=province:s_prov, timestamp=1380369573826, value="MA Boston"
96                                      column=shop:s_n, timestamp=1380369573826, value="Bostone Pizza"
96                                      column=tel:s_ph, timestamp=1380369573826, value="(617) 267-3663"
97                                      column=addr:address, timestamp=1380369573826, value="N & Blackstone  Boston  MA 02109-0210"
97                                      column=category:s_c, timestamp=1380369573826, value="Restaurant"
97                                      column=province:s_prov, timestamp=1380369573826, value="MA Boston"
97                                      column=shop:s_n, timestamp=1380369573826, value="Bostonian Hotel-Seasons Restaurant"
97                                      column=tel:s_ph, timestamp=1380369573826, value="(617) 720-0379"
98                                      column=addr:address, timestamp=1380369573826, value="40 Fay St  Boston  MA 02118-4318"
98                                      column=category:s_c, timestamp=1380369573826, value="Restaurant"
98                                      column=province:s_prov, timestamp=1380369573826, value="MA Boston"
98                                      column=shop:s_n, timestamp=1380369573826, value="Bostonian Market And Caf?"
98                                      column=tel:s_ph, timestamp=1380369573826, value="(617) 778-0417"
99                                      column=addr:address, timestamp=1380369573826, value="907 Boylston St Ste 21  Boston  MA 02115-3139"
99                                      column=category:s_c, timestamp=1380369573826, value="Restaurant"
99                                      column=province:s_prov, timestamp=1380369573826, value="MA Boston"
99                                      column=shop:s_n, timestamp=1380369573826, value="Boylston Restaurant"
99                                      column=tel:s_ph, timestamp=1380369573826, value="(617) 236-1767"
758 row(s) in 2.2610 seconds
```

FIGURE 3. Scanning a table in HBase using CLI.

TABLE 2. Performance index

Database	Performance Index
HBase+Solr	99
Cassandra	51
Huawei HBase	73
Solandra	50
Lily Project	77

functions. For example, we did a search using the shop-name field that included "Food" as its keyword, and 1000 results appeared filtering the province tag with "NY". We keyed in "shopname:Food" in "q" field, inputted "province:NY" in "fq" field, and gave 1000 in

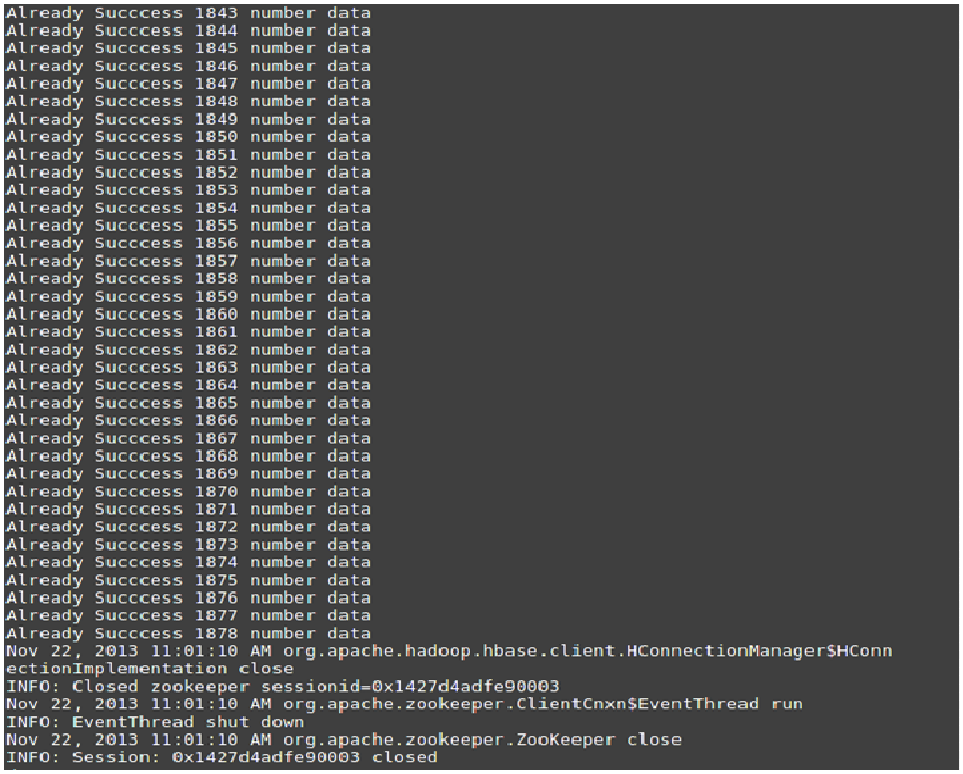

FIGURE 4. Importing data to Solr from HBase.

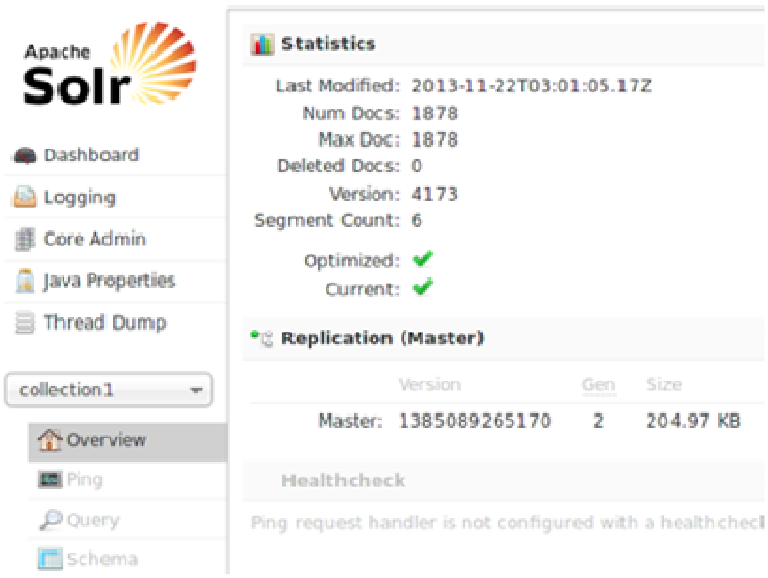

FIGURE 5. Presentation of Imported data in Solr using GUI.

rows field. Fig. 6 has shown the operation of query. The response time for the query function performed in Solr has also been marked. Besides average time-consuming on data read/write, document transfer, and query function is eventually obtained as listed in Table 1. After that, according to Eq. 4, we are able to evaluate the performance index for each database over a 5-year period of time as shown in Table 2.

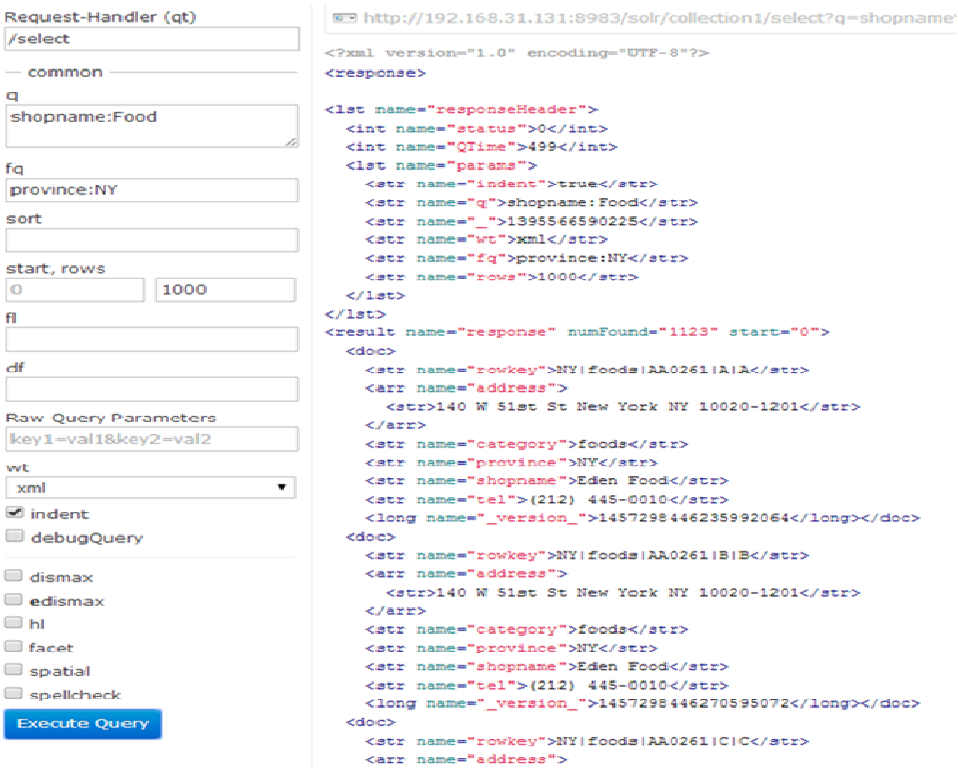

FIGURE 6. Response to a query in Solr using GUI

TABLE 3. Total cost of ownership over a 5-year period (Unit: USD)

Database	1^{st}Year	$2^{nd}Year$	3^{rd}Year	4^{th}Year	5^{th}Year
HBase+Solr	16393.3	13726.7	13726.7	13804.1	13877.9
Cassandra	16020	13353.3	13353.3	13430.8	13504.6
Huawei	16040	13373.3	13373.3	13450.8	13629.9
Solandra	13040	10373.3	10373.3	10450.8	10524.6
Lily Project	16393.3	13726.7	13726.7	13804.1	13877.9

TABLE 4. C-P ratio over a 5-year period

Database	1^{st}Year	$2^{nd}Year$	3^{rd}Year	4^{th}Year	5^{th}Year
HBase+Solr	61.00	72.85	72.85	72.44	72.06
Cassandra	31.94	38.32	38.32	38.10	37.89
Huawei	45.92	55.07	55.07	54.76	54.04
Solandra	38.85	48.84	48.84	48.48	48.14
Lily Project	47.27	56.46	56.46	56.14	55.84

4.3. **Assessment.** In the system assessment, we first analyzes total cost of ownership (TCO) according to several items such as hardware cost, staff cost, software cost, repair cost after warranty, down time cost, and extra upgrade cost. A summary of TCO has shown in Table 2. Here we estimated that hardware cost for two computers is $2666. Then, we assumed that the maintenance bill is $13000 every year for Hadoop together with HBase, Solr maintenance cost is approximately $300 per year, and for Cassandra

TABLE 5. Latency under stress test (unit: sec) (Win. =Window)

Query	Win. #1	Win. #2	Win. #3	Win. #4	Win. #5	Win. #6	Win. #7	Win. #8	Win. #9	Win. #10
10	0.15	0.1	0.2	0.2	0.1	0.15	0.16	0.15	0.2	0.2
100	1	1	0.8	1	1	0.8	1	1	1	1
1000	3	4	3	4	3	4	3	4	5	4

Query	Win. #11	Win. #12	Win. #13	Win. #14	Win. #15	Win. #16	Win. #17	Win. #18	Win. #19	Win. #20
10	0.15	0.15	0.15	0.15	0.2	0.2	0.16	0.15	0.2	0.2
100	1.2	0.8	1.1	1	1.1	1	1.2	1	1.1	1.2
1000	3	4	3	4	5	4	4	4	5	5

it would be $10300 every year. Accordingly we do the same maintenance estimation as the above-mentioned applications for Solandra and Lily Project because they are just the combination for the above applications. All of software cost is totally free due to open source. For hardware maintenance after warranty, we assumed that all the devices had the same risk of break-down, thus the chance of device break-down in the 4^{th} year was about 25%, while in the 5^{th} year it will be 50% chance. For the software upgrade cost, there is no charge because of open source. Regarding down time cost, we assumed that one application will cost $20 per year and the total cost would depend on the amount of software. Table 3 gives a summary of the total cost of ownership for this study. As for the system assessment, C-P ratio evaluation according to Eq. 6 for all of databases will yield a summary of those over a 5-year period of time as listed in Table 4.

4.4. **Stress Test and Discussion.** The issue about the stability and reliability of NoSQL database secondary indexing function has been concerned and hence a stress test of data retrieval in Solr has been taken in a big data environment. In this test, there are up to 20 threads (20 windows) used to accept the number of queries from 10 to 1000 and in the meantime the latency (time interval) has been counted. The key index in every query was different as shown in Fig. 1. Table 5 has listed the summary of latency and we have examined the results afterward. In the test from the statistics point of view the amount of opening windows obviously didn't affect the length of latency occurred in the query in Solr. The stability and reliability of NoSQL database secondary indexing function has verified because all of queries had responded in 5 seconds during the stress test.

5. **Conclusion.** This paper introduces the combination of NoSQL database HBase and enterprise search platform Solr to realize the secondary indexing function with fast query. In the assessment, a cost effectiveness evaluation called C-P ratio has been done among several competitive benchmark databases and the proposed one. As a result, our proposed approach outperforms the other databases and fulfills secondary indexing function with fast query in NoSQL database. Besides, a stress test has been taken to verify the stability and reliability of the proposed approach.

Acknowledgements. This work is fully supported by the Ministry of Science and Technology, Taiwan, Republic of China, under grant number **MOST 103-2221-E-390-011.**

REFERENCES

[1] D. Howe and M. Costanzo and P. Fey and T. Gojobori and L. Hannick and W. Hide and D. P. Hill and R. Kania and M. Schaeffer and S. S. Pierre and S. Twigger and O. White and S. Y. Rhee, Big data: The Future of Biocuratio, *Nature*, vol. 455, pp. 47-50, 2008.

[2] A. Jacobs, The Pathologies of Big Data, *Communications of the ACM - A Blind Person's Interaction with Technology*, vol. 52, no. 8, pp. 36-44, 2009.

[3] R. Cattell, Scalable SQL and NoSQL Dta Sores, *ACM SIGMOD Record*, vol. 39, mo. 4, pp. 12-27, 2010.

[4] J. Pokorny, NoSQL Databases: A Step to Database Scalability in Web Environment, *International Journal of Web Information Systems*, vol 9, no. 1, pp. 69-82, 2013.

[5] P. Zhou and J. Lei and W. Ye, Large-Scale Data Sets Clustering Based on MapReduce and Hadoop, *Journal of Computational Information Systems*, vol. 7, no. 16, pp. 5956-5963, 2011.

[6] J. K. Chiang, Authentication, Authorization and File Synchronization for Hybrid Cloud–The Development Centric to Google Apps, Hadoop and Linux Local Hosts, *Journal of Internet Technology*, vol 14, no. 7, pp. 1141-1148,2013.

[7] J. Leverich and C. Kozyrakis, On the Energy (in) Efficiency of Hadoop Clusters, *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, pp. 61-65, 2010.

[8] T. White, Hadoop: the Definitive Guide, *O'Reilly Media, Inc.*, Sebastopol, CA, USA, 2009.

[9] N. Dimiduk, HBase in Action, *Manning Publications*, Greenwich, UK, 2012.

[10] Y. Jiang, HBase Administration Cookbook, *Packt Publishing*, Birmingham, UK, 2012.

[11] C. Boja and A. Pocovnicu and L. Batagan, Distributed Parallel Architecture for Big Data, *Informatica Economica*, vol. 16, no. 2, pp. 116-127, 2012.

[12] J. Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, *Communications of the ACM - 50th Anniversary Issue*, vol. 51, no. 1, pp. 107-113, 2008.

[13] M. Hausenblas and J. Nadeau, Apache Drill: Interactive Ad-Hoc Analysis at Scale, *Big Data*, vol. 1, no. 2, pp.100-104, 2013.

[14] R.Kuc, Apache Solr 4 Cookbook, *Packt Publishing*, Birmingham, UK, 2013.

[15] T. Grainger and T. Potter, Solr In Action, *Manning Publications*, reenwich, UK, 2014.

[16] B. R. Chang, H.-F. Tsai and C.-M. Chen, Assessment of In-Cloud Enterprise Resource Planning System Performed in a Virtual Cluster, *Mathematical Problems in Engineering*,2014.

[17] B. R. Chang, H.-F. Tsai and C.-M. Chen, Evaluation of Virtual Machine Performance and Virtualized Consolidation Ratio in Cloud Computing System, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 3, pp. 192-200, 2013.