Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 560714, 10 pages
http://dx.doi.org/10.1155/2015/560714

Research Article

Implementation of Secondary Index on Cloud Computing NoSQL Database in Big Data Environment

Bao Rong Chang,[1] Hsiu-Fen Tsai,[2] Chia-Yen Chen,[1] Chien-Feng Huang,[1] and Hung-Ta Hsu[1]

[1] *Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 81148, Taiwan*
[2] *Department of Marketing Management, Shu-Te University, Kaohsiung 82445, Taiwan*

Correspondence should be addressed to Chien-Feng Huang; cfhuang15@nuk.edu.tw

Received 4 September 2014; Accepted 12 December 2014

Academic Editor: Gianluigi Greco

This paper introduces the combination of NoSQL database HBase and enterprise search platform Solr so as to tackle the problem of the secondary index function with fast query. In order to verify the effectiveness and efficiency of the proposed approach, the assessment using Cost-Performance 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 index function with fast query in NoSQL database. Moreover, according to the cross-sectional analysis, the proposed combination of HBase and Solr database is capable of performing an excellent query/response in a big data environment.

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 capable 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 index function for HBase. After the success of this combination, we go for a series of stress tests 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 Cost-Performance ratio (C-P ratio) [5] 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. Based on the cross-sectional data analysis [6], it will explore the performance of data access in NoSQL database in a big data environment as well.

For key-value database, it allows the application to store its data in a schema-less way. The data could be stored in a data type of a programming language or an object. There is no need for a fixed data model. Key-value storing divides many categories, like eventually consistent (always keeps the newest result if there is no update), hierarchical (can use the parent's attributes), cache in RAM (key value stored in memory, hash stored in cache, and hash used to present key-value index; time complexity is O(1)), solid state or rotating disk (like Google Bigtable which is used in solid state disk to enhance IO access speed), and ordered (with key-value pairs which can sort keys or values). For tabular database, it is a database that is structured in a tabular form. It arranges data elements in vertical columns and horizontal rows. Each cell is formed by the intersection of a column and row. Each row and column are uniquely numbered to make it orderly and efficient. This type of database has a virtually infinite range for mass data storage. Structuring data in tabular form may be the oldest method used. It is also simple. Tabular

Table 1: NoSQL database benchmark on 5 criteria.

Database	Performance	Scalability	Flexibility	Complexity	Functionality
Key-value store	High	High	High	Low	Variable
Column store	High	High	Moderate	Low	Minimum
Document store	High	Variable	High	Low	Variable
Graph database	Variable	Variable	High	High	Graph theory
Relational database	Variable	Variable	Low	Moderate	Relational algebra

database has several properties. They share the same set of properties per record. This means that every row has the same set of column titles. They access records through identifiers. Each table in a tabular database contains a particular set of related information that is connected with the database subject through key fields, which describe each record (row) so that, in the event of a query, the system can quickly locate the record. There are several famous databases of this type, like Google Bigtable, Apache Accumulo, Apache HBase, and Hypertable. For column store database, it stores data tables as sections of columns of data rather than rows of data. For RDBMS, rows are commonly used; the column store database has the advantage of aggregating computed data over large numbers of similar data items. Column store is used in data warehouse and CRM system. Using column store database, the system can evaluate which columns are being accessed and retrieved only if values are requested from the specific columns. For NoSQL database, each mechanism has different uses, and a famous database can have many properties, like Google Bigtable. It owns solid state disk key-value type and tabular type. For this study, HBase is a column-store database. It has an easy method to use, and its performance as well as the scalability is better than the others. Table 1 explains the performance of each type of database over 5 criteria.

The following paragraphs of this paper are arranged as follows. In Section 2, combination of NoSQL database and enterprise search platform will be described. The way to system assessment is given in Section 3. The experimental results and discussion will be obtained in Section 4. Finally, we drew a brief conclusion in Section 5.

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 user's 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 has Hadoop done in every host in a cluster environment [7–9].

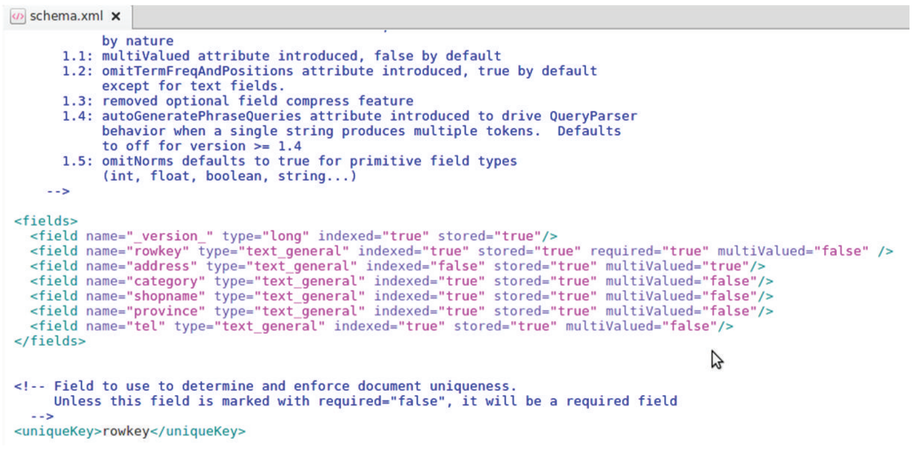
(3) After deploying Hadoop and ZooKeeper to cluster, we need to confirm the start-up of Hadoop and ZooKeeper services. We are able to give jps instruction at terminal to check whether or not the services are running normally. After that, we establish HBase service [10–13] within Hadoop.

(4) When procedure #3 has been done, web browser is used to view the start-up of Hadoop and HBase services. Key in http://localhost:50030/, 50040, 50070, and 60010 is used 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 to 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 cannot 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 "rowkey-category-shop name-telephone-province-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" [14]. We need to check the data integrity in HBase after data importing.



```
           by nature
       1.1: multiValued attribute introduced, false by default
       1.2: omitTermFreqAndPositions attribute introduced, true by default
           except for text fields.
       1.3: removed optional field compress feature
       1.4: autoGeneratePhraseQueries attribute introduced to drive QueryParser
           behavior when a single string produces multiple tokens.  Defaults
           to off for version >= 1.4
       1.5: omitNorms defaults to true for primitive field types
           (int, float, boolean, string...)
     -->

<fields>
  <field name="_version_" type="long" indexed="true" stored="true"/>
  <field name="rowkey" type="text_general" indexed="true" stored="true" required="true" multiValued="false" />
  <field name="address" type="text_general" indexed="false" stored="true" multiValued="true"/>
  <field name="category" type="text_general" indexed="true" stored="true" multiValued="false"/>
  <field name="shopname" type="text_general" indexed="true" stored="true" multiValued="false"/>
  <field name="province" type="text_general" indexed="true" stored="true" multiValued="false"/>
  <field name="tel" type="text_general" indexed="true" stored="true" multiValued="false"/>
</fields>


<!-- Field to use to determine and enforce document uniqueness.
     Unless this field is marked with required="false", it will be a required field
  -->
<uniqueKey>rowkey</uniqueKey>
```

FIGURE 1: Apache Solr configuring file for indexing.

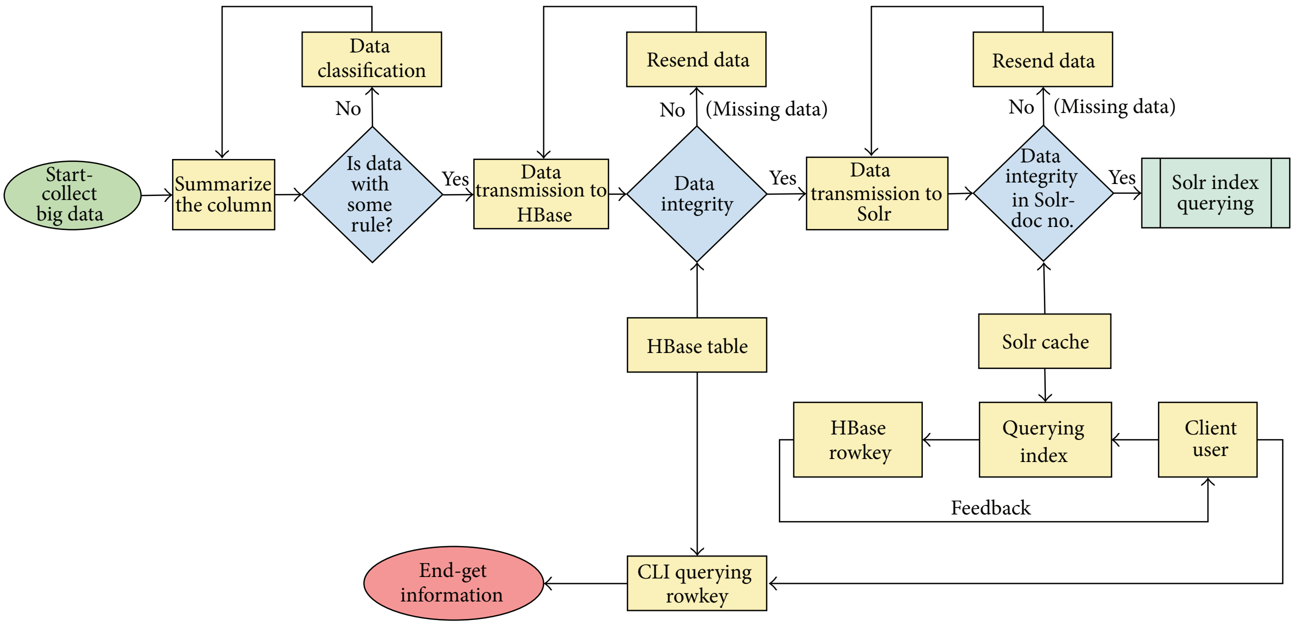


FIGURE 2: Flowchart of HBase together with Solr to implement secondary index operation.

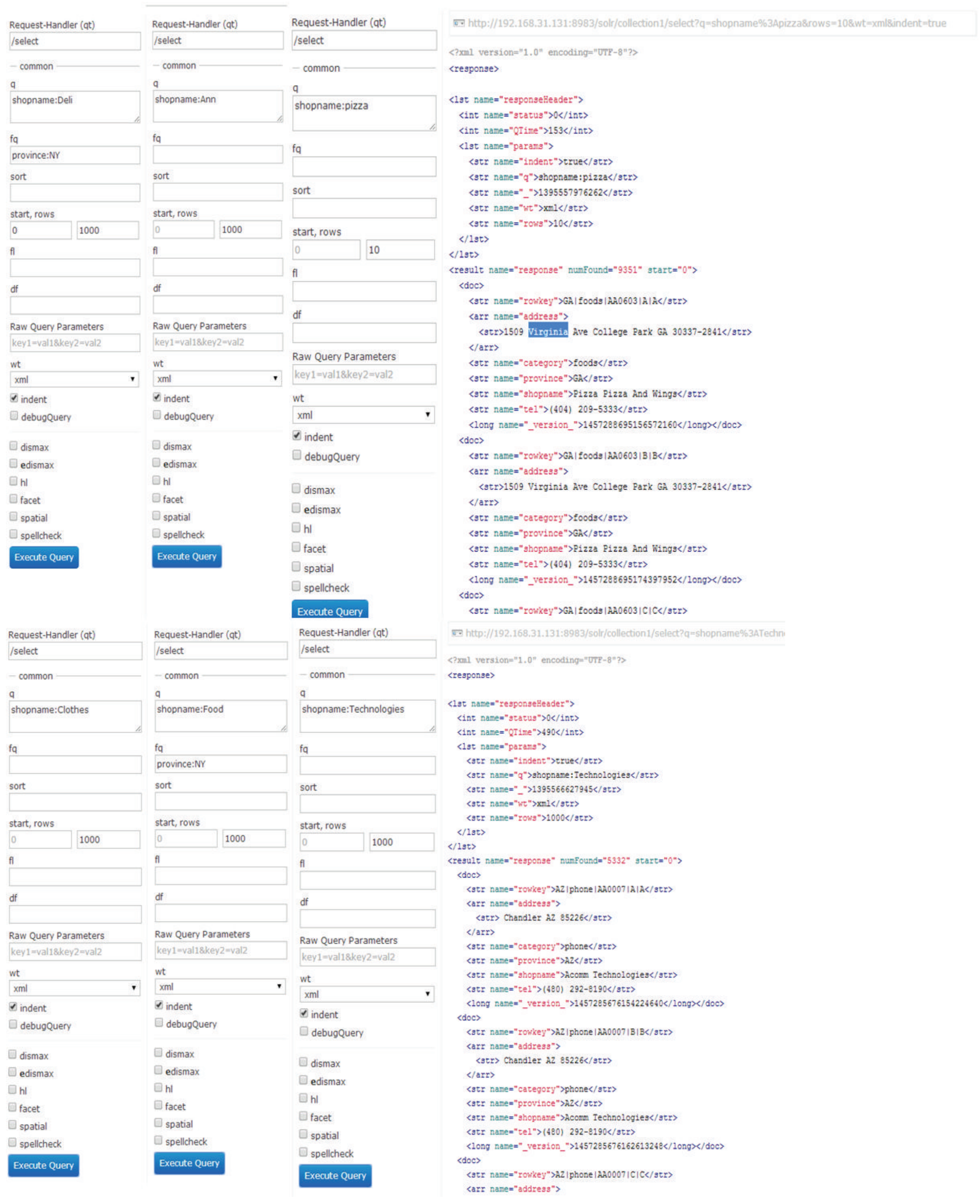
(8) Then, we use HBase output API and Apache HTTP API to transfer the document to Solr from HBase [15–17]. After the transmission, the indexes are created and the content is saved in memory in Solr, that is, the schema as defined and shown in Figure 1. We can use web browser to check the amount of documents in Solr. Data in a row represent a document. We can use query function to search our keyword (Secondary index or more) and reversely to search the primary index in Solr. We may be able to apply filter 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.

In Figure 2, a flowchart represents HBase together with Solr to implement secondary index operation.

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. 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 (1), where $t_{s_{ijk}}$ represents a single datum access time, for a single run t_{ijk} stands for



```
Request-Handler (qt)
/select
common
q
shopname:Deli
fq
province:NY
sort

start, rows
0        1000
fl

df

Raw Query Parameters
key1=val1&key2=val2
wt
xml
✓ indent
☐ debugQuery
☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck
Execute Query
```

```
Request-Handler (qt)
/select
common
q
shopname:Ann
fq

sort

start, rows
0        1000
fl

df

Raw Query Parameters
key1=val1&key2=val2
wt
xml
✓ indent
☐ debugQuery
☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck
Execute Query
```

```
Request-Handler (qt)
/select
common
q
shopname:pizza
fq

sort

start, rows
0        10
fl

df

Raw Query Parameters
key1=val1&key2=val2
wt
xml
✓ indent
☐ debugQuery
☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck
Execute Query
```

```
http://192.168.31.131:8983/solr/collection1/select?q=shopname%3Apizza&rows=10&wt=xml&indent=true

<?xml version="1.0" encoding="UTF-8"?>
<response>

<lst name="responseHeader">
  <int name="status">0</int>
  <int name="QTime">153</int>
  <lst name="params">
    <str name="indent">true</str>
    <str name="q">shopname:pizza</str>
    <str name="_">1395557976262</str>
    <str name="wt">xml</str>
    <str name="rows">10</str>
  </lst>
</lst>
<result name="response" numFound="9351" start="0">
  <doc>
    <str name="rowkey">GA|foods|AA0603|A|A</str>
    <arr name="address">
      <str>1509 Virginia Ave College Park GA 30337-2841</str>
    </arr>
    <str name="category">foods</str>
    <str name="province">GA</str>
    <str name="shopname">Pizza Pizza And Wings</str>
    <str name="tel">(404) 209-5333</str>
    <long name="_version_">1457288695156572160</long></doc>
  <doc>
    <str name="rowkey">GA|foods|AA0603|B|B</str>
    <arr name="address">
      <str>1509 Virginia Ave College Park GA 30337-2841</str>
    </arr>
    <str name="category">foods</str>
    <str name="province">GA</str>
    <str name="shopname">Pizza Pizza And Wings</str>
    <str name="tel">(404) 209-5333</str>
    <long name="_version_">1457288695174397952</long></doc>
  <doc>
    <str name="rowkey">GA|foods|AA0603|C|C</str>
```

```
Request-Handler (qt)
/select
common
q
shopname:Clothes
fq

sort

start, rows
0        1000
fl

df

Raw Query Parameters
key1=val1&key2=val2
wt
xml
✓ indent
☐ debugQuery
☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck
Execute Query
```

```
Request-Handler (qt)
/select
common
q
shopname:Food
fq
province:NY
sort

start, rows
0        1000
fl

df

Raw Query Parameters
key1=val1&key2=val2
wt
xml
✓ indent
☐ debugQuery
☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck
Execute Query
```

```
Request-Handler (qt)
/select
common
q
shopname:Technologies
fq

sort

start, rows
0        1000
fl

df

Raw Query Parameters
key1=val1&key2=val2
wt
xml
✓ indent
☐ debugQuery
☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck
Execute Query
```

```
http://192.168.31.131:8983/solr/collection1/select?q=shopname%3ATechn

<?xml version="1.0" encoding="UTF-8"?>
<response>

<lst name="responseHeader">
  <int name="status">0</int>
  <int name="QTime">490</int>
  <lst name="params">
    <str name="indent">true</str>
    <str name="q">shopname:Technologies</str>
    <str name="_">1395566627945</str>
    <str name="wt">xml</str>
    <str name="rows">1000</str>
  </lst>
</lst>
<result name="response" numFound="5332" start="0">
  <doc>
    <str name="rowkey">AZ|phone|AA0007|A|A</str>
    <arr name="address">
      <str> Chandler AZ 85226</str>
    </arr>
    <str name="category">phone</str>
    <str name="province">AZ</str>
    <str name="shopname">Acomm Technologies</str>
    <str name="tel">(480) 292-8190</str>
    <long name="_version_">1457285676154224640</long></doc>
  <doc>
    <str name="rowkey">AZ|phone|AA0007|B|B</str>
    <arr name="address">
      <str> Chandler AZ 85226</str>
    </arr>
    <str name="category">phone</str>
    <str name="province">AZ</str>
    <str name="shopname">Acomm Technologies</str>
    <str name="tel">(480) 292-8190</str>
    <long name="_version_">1457285676162613248</long></doc>
  <doc>
    <str name="rowkey">AZ|phone|AA0007|C|C</str>
    <arr name="address">
```

Figure 3: Latency under stress test for Solr (presenting 6 windows).

measured total time for a specific data size at a certain database, and N_{ik} means a specific data size. In (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 (3), where $\overline{\mathrm{PI}}_{jk}$ 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 [18] for each database as shown in (4), where PI_j represents performance index, SF_1 stands for scale factor #1, W_k is the respective weight, and $\overline{\mathrm{PI}}_{jk}$ means a normalized performance index. In order to assess the cost effectiveness evaluation, we need to calculate total cost of ownership [19] in (5), showing the expenditure of money in the implementation of secondary index 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 downtime 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 US 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 (6), where CP_{jg} is C-P ratio, SF_2 stands for scale factor #2, and TCO_{jg} means total cost of ownership as well as subscript j that represents various data center and g that stands for a certain period of time. Consider the following:

$$t_{s_{ijk}} = \frac{t_{ijk}}{N_{ik}}, \qquad (1)$$

where $i = 1, 2, \ldots, l$, $j = 1, 2, \ldots, m$, and $k = 1, 2, \ldots, n$,

$$\bar{t}_{s_{jk}} = \sum_{i=1}^{l} w_i \cdot t_{s_{ijk}}, \qquad (2)$$

where $j = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n$, and $\sum_{i=1}^{l} w_i = 1$,

$$\overline{\mathrm{PI}}_{jk} = \frac{1/\bar{t}_{s_{jk}}}{\mathrm{Max}_{h=1,2,\ldots,m}\left(1/\bar{t}_{s_{hk}}\right)}, \qquad (3)$$

where $j = 1, 2, \ldots, m$ and $k = 1, 2, \ldots, n$,

$$\mathrm{PI}_j = \mathrm{SF}_1 \cdot \left(\sum_{k=1}^{n} W_k \cdot \overline{\mathrm{PI}}_{jk}\right), \qquad (4)$$

where $j = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n$, $\mathrm{SF}_1 = 10^2$, and $\sum_{k=1}^{n} W_k = 1$,

$$\mathrm{TCO}_{jg} = \sum_a \mathrm{HC}_a + \sum_b S_b + \sum_c \mathrm{RCAW}_c + \sum_d \mathrm{DTC}_d + \sum_e \mathrm{EUC}_e, \qquad (5)$$

where $j = 1, 2, \ldots, m$ and $g = 1, 2, \ldots, o$,

$$\mathrm{CP}_{jg} = \mathrm{SF}_2 \cdot \frac{\mathrm{PI}_j}{\mathrm{TCO}_{jg}}, \qquad (6)$$

where $j = 1, 2, \ldots, m$, $g = 1, 2, \ldots, o$, and $\mathrm{SF}_2 = 10^4$.

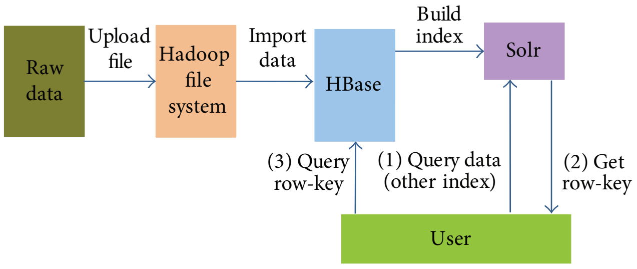


Figure 4: Implementation procedure.

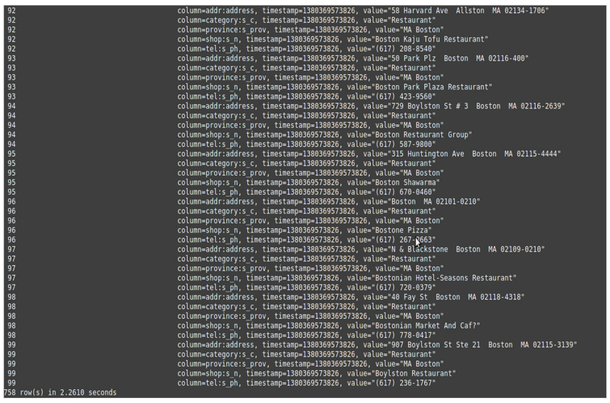

```
92                                  column=addr:address, timestamp=1380369573826, value="58 Harvard Ave  Allston  MA 02134-1706"
92                                  column=category:s_c, timestamp=1380369573826, value="Restaurant"
92                                  column=province:s_prov, timestamp=1380369573826, 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) 267-2663"
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 5: Scanning a table in HBase using CLI.

In order to examine the stability and reliability of NoSQL database secondary index 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 to 10 to 1000 queries and we had checked the latency (time interval) simultaneously. The key index in every query was different as shown in Figure 3. Clearly, the result would indicate the response time for the query in Solr and explain what correlation between the amount of windows and the latency was found.

4. Experimental Results and Discussion

There are a few experiments and a discussion presented in the following subsections.

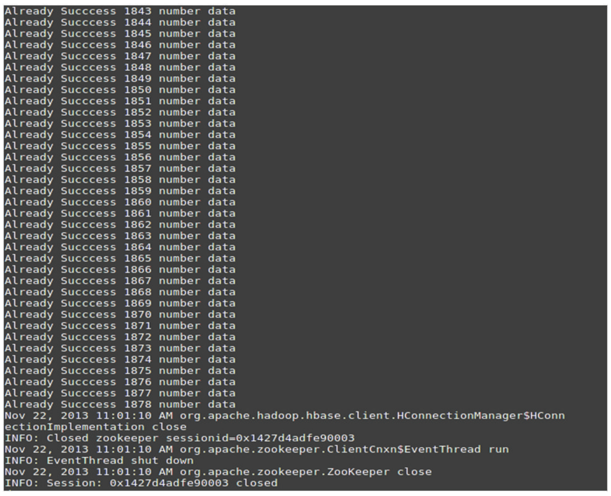
4.1. Data Transfer and Data Integrity Checking. In regard to implementation procedure as shown in Figure 4, which indicated data transfer from HDFS to HBase and/or from HBase to Solr, there are risks of losing data during the transition. We have to verify the data integrity in HBase inner table and the amount of input documents in Solr. For examining HBase, we checked inner table using the command "scan table-name" in CLI as shown in Figure 5. In Figure 6, 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 Figure 6. Furthermore, in terms of the performance evaluation, the time for data writing/reading in every database has been measured as listed in Tables 2 and 3.

Table 2: Time for writing data to database (unit: sec.).

Data size	HBase + Solr	Cassandra	Huawei HBase	Solandra	Lily Project
10^4	23	110.4	23	120	23
10^5	23.2	1109.2	23	1215	24
10^6	123.4	11211.3	125	11253	137
10^7	388.5	113157.7	390	113189	412

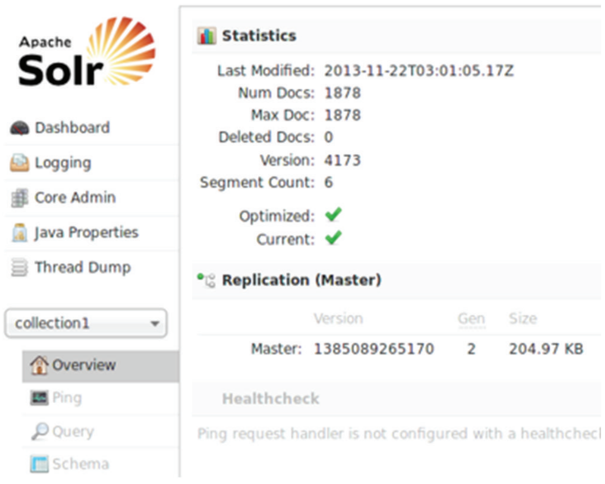
Table 3: Time for reading data from database (unit: sec.).

Data size	HBase + Solr	Cassandra	Huawei HBase	Solandra	Lily Project
10^4	27.2	27.6	30	29	27
10^5	266.5	270.7	269	288.5	273
10^6	2572.2	2614.2	2589.7	2735	2566
10^7	24312	24701	24479	24988	24385



```
Already Succcess 1843 number data
Already Succcess 1844 number data
Already Succcess 1845 number data
Already Succcess 1846 number data
Already Succcess 1847 number data
Already Succcess 1848 number data
Already Succcess 1849 number data
Already Succcess 1850 number data
Already Succcess 1851 number data
Already Succcess 1852 number data
Already Succcess 1853 number data
Already Succcess 1854 number data
Already Succcess 1855 number data
Already Succcess 1856 number data
Already Succcess 1857 number data
Already Succcess 1858 number data
Already Succcess 1859 number data
Already Succcess 1860 number data
Already Succcess 1861 number data
Already Succcess 1862 number data
Already Succcess 1863 number data
Already Succcess 1864 number data
Already Succcess 1865 number data
Already Succcess 1866 number data
Already Succcess 1867 number data
Already Succcess 1868 number data
Already Succcess 1869 number data
Already Succcess 1870 number data
Already Succcess 1871 number data
Already Succcess 1872 number data
Already Succcess 1873 number data
Already Succcess 1874 number data
Already Succcess 1875 number data
Already Succcess 1876 number data
Already Succcess 1877 number data
Already Succcess 1878 number data
Nov 22, 2013 11:01:10 AM org.apache.hadoop.hbase.client.HConnectionManager$HConn
ectionImplementation close
INFO: Closed zookeeper sessionid=0x1427d4adfe90003
Nov 22, 2013 11:01:10 AM org.apache.zookeeper.ClientCnxn$EventThread run
INFO: EventThread shut down
Nov 22, 2013 11:01:10 AM org.apache.zookeeper.ZooKeeper close
INFO: Session: 0x1427d4adfe90003 closed
```

Figure 6: Importing data to Solr from HBase.



Statistics

Last Modified: 2013-11-22T03:01:05.17Z
Num Docs: 1878
Max Doc: 1878
Deleted Docs: 0
Version: 4173
Segment Count: 6
Optimized: ✓
Current: ✓

Replication (Master)

	Version	Gen	Size
Master:	1385089265170	2	204.97 KB

Healthcheck

Ping request handler is not configured with a healthchecl

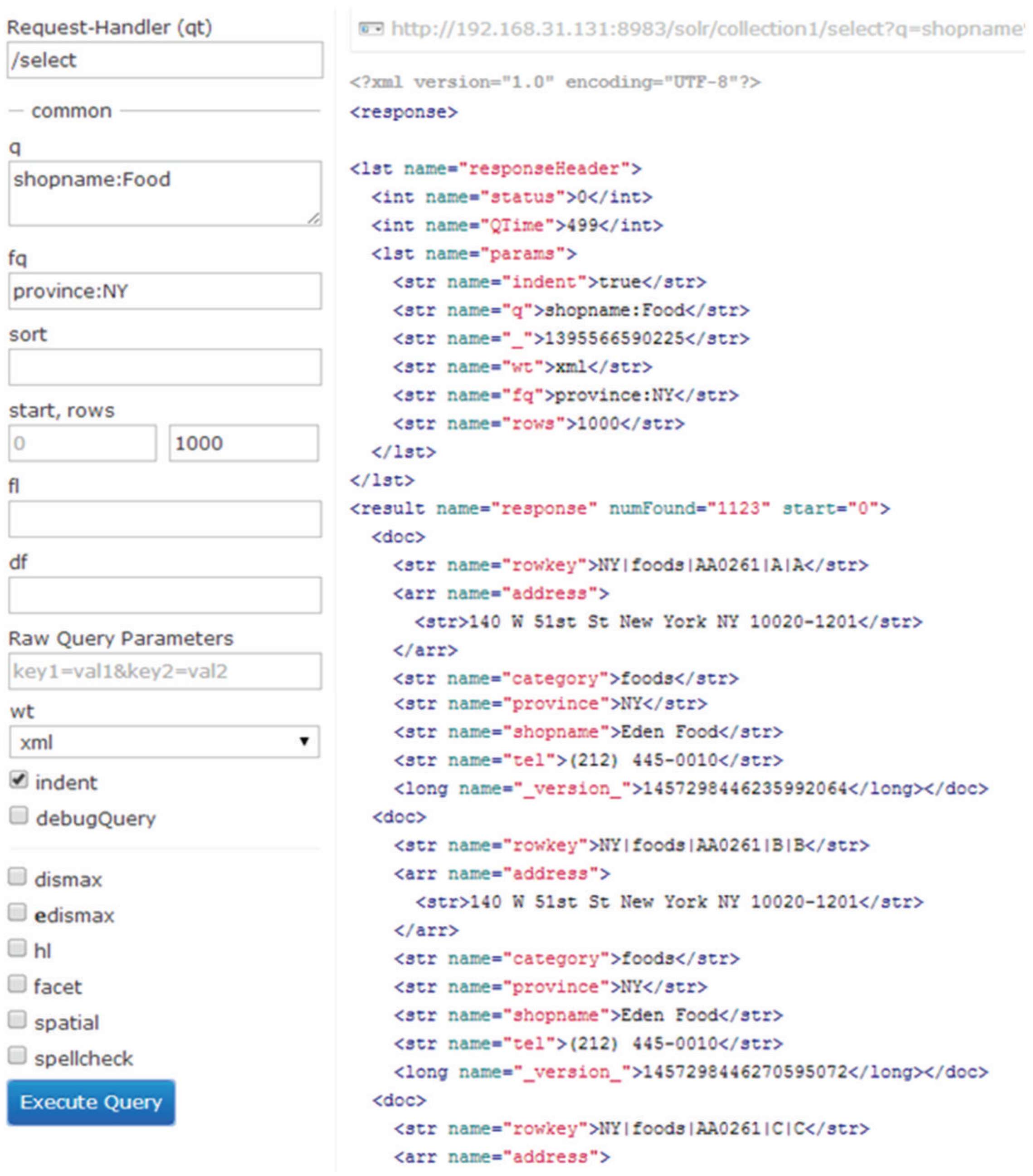
Figure 7: Presentation of Imported data in Solr using GUI.

Time for data transfer to Solr from every database has been recorded as listed in Table 4.

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 has logged into the http server, that is Solr, to respond swiftly to the connection and deliver the on-line document to http server. This also demonstrates an efficient 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 been done completely, the data are available in Solr and we could check the amount of document in Solr as shown in Figure 7. In order to verify the secondary index function in the combination of HBase and Solr, we launched the query test in Solr as shown in Figure 8, where we can check the information about the related operations on the web. Solr provides normal search, filtering search, spatial search, and other more search 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 rows field. Figure 8 has shown the operation of query. In Table 5, 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 6. After that, according to (4), we are able to evaluate the performance index for each database over a 5-year period of time as shown in Table 7.

4.3. Assessment. In the system assessment, we first analyze total cost of ownership (TCO) according to several items such as hardware cost, staff cost, software cost, repair cost



Request-Handler (qt): /select

common

q: shopname:Food

fq: province:NY

sort:

start, rows: 0 1000

fl:

df:

Raw Query Parameters: key1=val1&key2=val2

wt: xml

[✓ indent]

debugQuery

dismax

edismax

hl

facet

spatial

spellcheck

Execute Query

```
http://192.168.31.131:8983/solr/collection1/select?q=shopname

<?xml version="1.0" encoding="UTF-8"?>
<response>

<lst name="responseHeader">
  <int name="status">0</int>
  <int name="QTime">499</int>
  <lst name="params">
    <str name="indent">true</str>
    <str name="q">shopname:Food</str>
    <str name="_">1395566590225</str>
    <str name="wt">xml</str>
    <str name="fq">province:NY</str>
    <str name="rows">1000</str>
  </lst>
</lst>
<result name="response" numFound="1123" start="0">
  <doc>
    <str name="rowkey">NY|foods|AA0261|A|A</str>
    <arr name="address">
      <str>140 W 51st St New York NY 10020-1201</str>
    </arr>
    <str name="category">foods</str>
    <str name="province">NY</str>
    <str name="shopname">Eden Food</str>
    <str name="tel">(212) 445-0010</str>
    <long name="_version_">1457298446235992064</long></doc>
  <doc>
    <str name="rowkey">NY|foods|AA0261|B|B</str>
    <arr name="address">
      <str>140 W 51st St New York NY 10020-1201</str>
    </arr>
    <str name="category">foods</str>
    <str name="province">NY</str>
    <str name="shopname">Eden Food</str>
    <str name="tel">(212) 445-0010</str>
    <long name="_version_">1457298446270595072</long></doc>
  <doc>
    <str name="rowkey">NY|foods|AA0261|C|C</str>
    <arr name="address">
```

Figure 8: Response to a query in Solr using GUI.

after warranty, downtime cost, and extra upgrade cost. A summary of TCO has been shown in Table 7. 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 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 of 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 breakdown, and thus the chance of device breakdown in the 4th year was about 25%, while in the 5th year it will be 50% chance. For the software upgrade cost, there is no charge because of open source. Regarding downtime cost, we assumed that one application will cost $20 per year and the total cost would depend on the amount of software. Table 8 gives a summary of the total cost of ownership for this study. As for the system assessment, C-P ratio evaluation according to (6) for all of databases will yield a summary of those over a 5-year period of time as listed in Table 9.

4.4. Stress Test and Discussion. The issue about the stability and reliability of NoSQL database secondary index function has been concerned and hence a stress test of data retrieval

Table 4: Document transfer time from database to Solr (unit: sec.).

Data size	HBase + Solr	Cassandra	Huawei HBase	Solandra	Lily Project
10^4	109	120	115	123	115
10^5	1121	1130	1125	1154	1130
10^6	11105	11286	11173	11330	11186
10^7	108055	112806	112347	113105	112395

Table 5: Response time for querying function performed in Solr (unit: sec.).

Data size	HBase + Solr	Cassandra	Huawei HBase	Solandra	Lily Project
10^4	0.15	0.91	45	2	1
10^5	0.5	11.12	288	7	5
10^6	2	143.1	547	15	10
10^7	10	2011.13	1867	60	45

Table 6: Average time of a single datum access (unit: sec.).

Operation	HBase + Solr	Cassandra	Huawei HBase	Solandra	Lily Project
Data write	0.000673563	0.011164768	0.0006735	0.011680475	0.00067955
Data read	0.0025971	0.002637825	0.0026819	0.0027547	0.002608625
Document transfer	0.011005125	0.01146665	0.011289425	0.011620125	0.011306375
Query/response	0.00000575	0.000136603	0.002028425	0.00007275	0.000041125

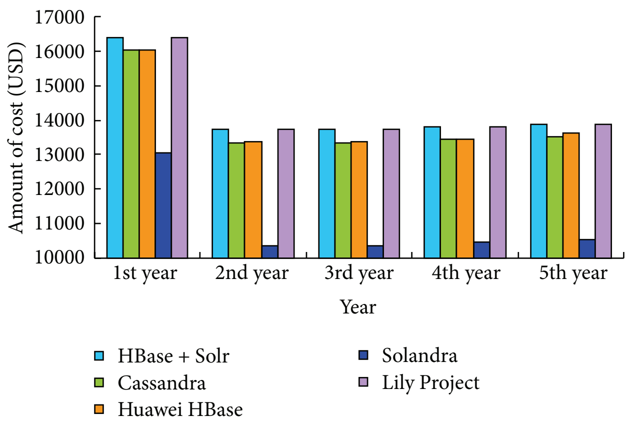


Figure 9: Total cost of ownership among various databases over a 5-year period.

Table 7: Performance index.

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

Table 8: Total cost of ownership over a 5-year period (unit: USD).

Database	1st year	2nd year	3rd year	4th year	5th 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 9: C-P ratio over a 5-year period.

Database	1st year	2nd year	3rd year	4th year	5th 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

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 Figure 3. Table 10 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 did not affect the length of latency occurring in the query in Solr. The stability and reliability of NoSQL database secondary index function have been verified because all of queries had responded in 5 seconds during the stress test.

It noted that performance indexes for five databases have been listed in Table 7 and they are time-invariant. In Figure 9, the total cost of ownership for our proposed approach has varied from year to year where it goes down dramatically and goes up slowly over a 5-year period. Accordingly, C-P ratio of the proposed approach goes up abruptly and almost

Table 10: 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
	#11	#12	#13	#14	#15	#16	#17	#18	#19	#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

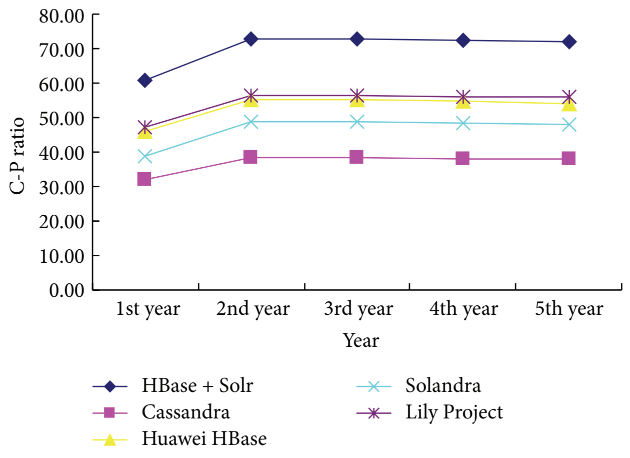


Figure 10: C-P ratio among various databases over a 5-year period.

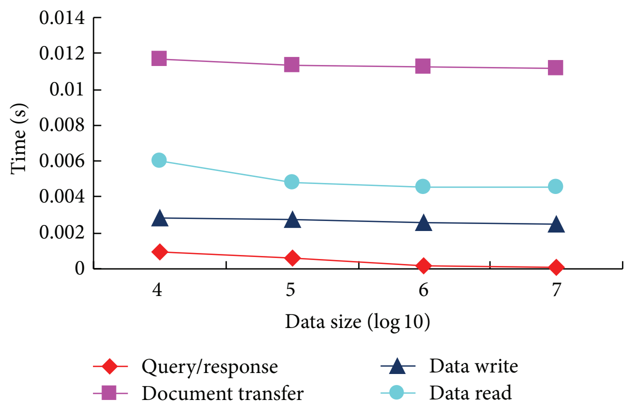


Figure 11: Average time of a single datum access in a certain database.

maintains the same level afterward as shown in Figure 10. Consequently, according to C-P ratio, our proposed approach outperforms the others during this period, as listed in Table 9. This has verified that our proposed approach has been realized successfully and performed significantly for a NoSQL secondary index function and fast query.

There are four tests about the function of data read, data write, document transfer, and query/response, as mentioned above in this paper to measure the average time of a single datum access in a certain database as listed in Tables 2 to 4. What we are interested in is to figure out whether the average time of a single datum access may be varied with data size or not for these functions. As shown in Figure 11, the cross-sectional data analysis [6] gave that it takes least time on the function of query/response when comparing with the other functions. The average time reduces dramatically as data size increases because the hit rate of data retrieval goes up rapidly in memory cache and concurrently the response time shrinks sharply. This figure illustrates that NoSQL database with secondary index function can achieve an excellent performance in query/response of a certain database, especially in a big data environment.

5. Conclusion

This paper introduces the combination of NoSQL database HBase and enterprise search platform Solr to realize the secondary index 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 index function with fast query in NoSQL database. Besides, a stress test has been taken to verify the stability and reliability of the proposed approach. Finally, according to the cross-sectional analysis, the proposed combination of HBase and Solr database is capable of performing an excellent query/response in a big data environment.

Conflict of Interests

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

Acknowledgment

This work is supported by the Ministry of Science and Technology, Taiwan, Republic of China, under Grant no. MOST 103-2221-E-390-011.

References

[1] D. Howe, M. Costanzo, P. Fey et al., “Big data: the future of biocuration,” *Nature*, vol. 455, no. 7209, 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 data stores," *ACM SIGMOD Record*, vol. 39, no. 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] B. R. Chang, H.-F. Tsai, C.-M. Chen, and C.-F. Huang, "Analysis of virtualized cloud server together with shared storage and estimation of consolidation ratio and TCO/ROI," *Engineering Computations*, vol. 31, no. 8, pp. 1746–1760, 2014.

[6] C.-C. Lee and C.-P. Chang, "Energy consumption and economic growth in Asian economies: a more comprehensive analysis using panel data," *Resource and Energy Economics*, vol. 30, no. 1, pp. 50–65, 2008.

[7] P. Zhou, 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.

[8] 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.

[9] 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.

[10] T. White, *Hadoop: The Definitive Guide*, O'Reilly Media, Sebastopol, Calif, USA, 2009.

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

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

[13] C. Boja, A. Pocovnicu, and L. Batagan, "Distributed parallel architecture for big data," *Informatica Economica*, vol. 16, no. 2, pp. 116–127, 2012.

[14] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[15] M. Hausenblas and J. Nadeau, "Apache drill: interactive ad-hoc analysis at scale," *Big Data*, vol. 1, no. 2, pp. 100–104, 2013.

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

[17] T. Grainger and T. Potter, *Solr in Action*, Manning Publications, Greenwich, UK, 2014.

[18] B. R. Chang, H.-F. Tsai, and C.-M. Chen, "High-performed virtualization services for in-cloud enterprise resource planning system," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 5, no. 4, pp. 614–624, 2014.

[19] 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.