A Simulator for the Execution Efficiency Measurement of Distributed Multi-Database Virtualization

Daichi Kano*, Hiroyuki Sato*, Jun Sawamoto*, and Yuji Wada**

* Graduate School of Software and information Science, Iwate Prefectural University, Japan
** Department of Information Environment, Tokyo Denki University, Japan
sawamoto@iwate-pu.ac.jp

Abstract –In database virtualization technology, the database of a different kind can be used as if it were a kind of database. However decline of execution efficiency is left as one of the research subjects. In improving the execution efficiency, it is necessary to measure the execution performance of the virtualization processes, especially in a distributed environment where multiple databases are connected via a network. In this study, we have designed and implemented the simulator for the execution efficiency measurement. This simulator measures the execution efficiency by calculating the processing time of virtualization processes, database processes and communication processes, and totaling them.

Keywords: Distributed database, Multi-database virtualization, Simulator, Performance evaluation and improvement.

1 INTRODUCTION

Today, it is important to discover and analyze the knowledge and trends which are hidden in large collections of data on ubiquitous network environment using data mining technology, and to use them for decision making of business, etc. However, since those data exists in various types of distributed databases, an appropriate database has to be chosen from a variety of databases and accessed properly. The work of the preparation process of data mining of acquiring appropriate data is needed, and it becomes a burden for the data analysis engineer who performs data mining in the distributed database environment.

To reduce this burden, the multi-database virtualization technology which enables a user to access various types of databases as if accessing a single type of databases has been studied [1-3]. The usefulness has been shown when database virtualization technology is used to perform data mining.

However, some research issues are pointed out. Degradation of the execution efficiency by virtualization processing among the research issues remain by the previous work as one of the main subjects to be solved. Since virtualization processing is performed in addition to normal database processing, it causes execution degradation. Virtualization processing transforms commands and the processing result based on the schema, and when especially the processing result becomes extensively large, virtualization processing becomes a burden. An improvement can be expected by using load sharing technology and parallel processing technology for this issue. While each load decreases by distributing data processing and parallel processing, we anticipate the generation of network delay by low line speed, congestion, etc. Therefore, factors about the network, such as communication time and transmission speed, become important as well as processing of databases.

In improving the execution efficiency, it is necessary to measure the execution performance of the virtualization processing, especially in a distributed environment where multiple databases are connected via a network. But it takes a lot of databases and large-scale network structure, and preparation of actual measurement environment is costly and very difficult. Therefore, the measurement environment using a simulator is considered.

In this study, we have designed and implemented the simulator for the execution efficiency measurement. This simulator measures the execution efficiency by considering the processing time of virtualization processes, database processes and communication comprehensively. And we aim to contribute to quantitative verification and evaluation of the execution efficiency improvement technique of virtualization processing.

The rest of this paper is organized as follows: In section 2, we describe related works. In section 3, we present our proposed solution for database virtualization. In section 4, details of the design of the proposed simulator are described. In section 5, we report the process and some results of acquiring reference parameters for the time of virtualization processing. Finally, the paper is concluded in section 6.

2 RELATED WORKS

The performance estimation of database system is an active research area. They mainly approach this subject by building performance models of database servers and running the models for the simulation [1]-[3].

Garcia [1] presents a simple model based on the queuing network paradigm using fixed distribution for the service times of the queues. The parameters used in the model are adjusted using measurements taken from real servers. This work demonstrates that extreme simple model is capable of predicting the performance of metrics of real database servers with high accuracy and capturing the essential performance aspects of database servers.

Wu, et al [2]-[3] propose a method for predicting query execution time for concurrent and dynamic database workloads. Their approach is based on analytic model rather than machine-learning model. They use optimizer's cost model to estimate the I/O and CPU operations for each individual query, and then use a queuing model to combine

these estimates for concurrent queries to predict their execution times. A buffer pool model is also used account for the cache effect of the buffer pool.

These related works are all targeted for real database servers. On the other hand, our target is virtualized distributed multi-database system. And we have designed and implemented the simulator for the execution efficiency measurement by considering the processing time of virtualization processes, database processes and communication processes. Our main goal is to discover the bottlenecks of the database virtualization processing.

Some earlier reports [4]-[6] have described the study of database virtualization technology.

Mori et al. [4] proposed development of a system to disseminate information actively to all users in a mobile computing environment. They implemented an experimental system using the meta-level active multi-database system as the platform in a mobile computing environment. By mapping the data of the local database group to a meta-database through the basic search and build operations, the system intends to combine data and include different types of local database group.

The data integration technique, Teiid [5], enables virtualization of various types of databases; through such virtual databases, one can access such data sources as relational databases, web databases, and application software such as ERP and CRM, etc. in real time. They can all be integrated for use. In fact, Teiid has a unique query engine. Furthermore, the real-time data integration is accomplished by connecting business application software through the JDBC/SOAP access layer with data sources which are accessed through the connector framework.

In [6], they similarly describes a module known as a wrapper that allows accessing and integrating data from various sources such as RDBs, the Web, and Excel files.

In our previous study [7], we considered the metadata, UML, ER model, and the XML schema as candidates for use to accomplish database virtualization. Thereby, ubiquitous databases can be used as if they were a single database. We then compared the advantages and disadvantages of each to analyse them as follows.

In our previous studies [7]-[10], we examined XML schema advantages and proposed a virtualization method by which such ubiquitous databases as relational databases, object-oriented databases, and XML databases are usable, as if they all behaved as a single database.

3 DATABASE VIRTUALIZATION [7]

Databases of many kinds exist in terms of their associated data model differences and vendor differences. Regarding differences among data models, each has different data representation, and unique associated manipulation. Some typical examples include the table type of relational databases (RDB), XML-representation type of XML databases (XMLDB), and object-oriented databases (OODB). Even the same model database might have different features among vendors. Regarding RDB for example, there might be some differences in SQL and/or data type representation. The typical example is that we have MySQL, PostgreSQL, and SQLServer from different vendors.

These differences according to the model and vendor bring some undesired results. For example, we might end up spending more time and labor during application system development because of the different data models that must be confronted. For example, we might need to acquire the right API to handle data of every different type of database. Virtualization of such different types of modelled databases to unify the procedures for all of them would probably impart less of workload and cost, and facilitate their management in a more flexible manner. Consequently, virtualization of databases, if it could be done, would facilitate application system design and database management as well.

To have a virtualization feature, we will consider the inclusion of features to manage distributed databases of similar types, the distributed databases of different types, and provide location transparency for users, such that they notice no differences of database structure or location and become able to use databases of all kinds in a flexible fashion. Fig. 1 portrays an example view of the database virtualization technique.

For virtualization of ubiquitous databases in our study, we will describe the schema information of the real databases, of which more than one always happens to exist, by creating and using one common XML schema. We also provide functionality of data search and update with the XML-based common data manipulation API.

3.1 XML Conversion Program

We will use an XML schema that provides a flexible representation capability and a high transparency capability.

To do so, we will produce such a virtualization concept in which the user would feel as if he or she were locally manipulating the remote site RDB from a local RDB process environment. That can be accomplished by converting the schema information and data information of the local RDB into the XML schema, and then storing that information into the RDB that the user would like to operate.

We developed an XML conversion program, XML Export/Import, as depicted in Fig. 2. We then used such different vendor RDBs as MySQL, PostgreSQL, and SQLServer2005 because they are available in the RDB virtualization system creation environment. We have to rebuild the XML tree with our XML conversion program when the distributed database is redefined.

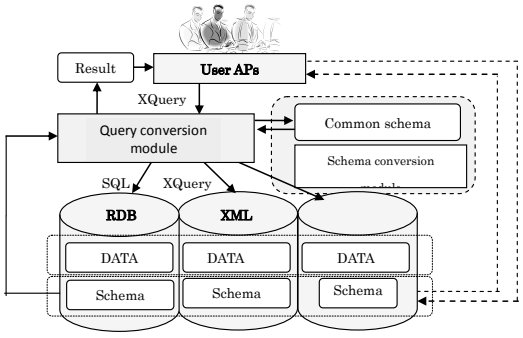

Figure 1: An example view of database virtualization.

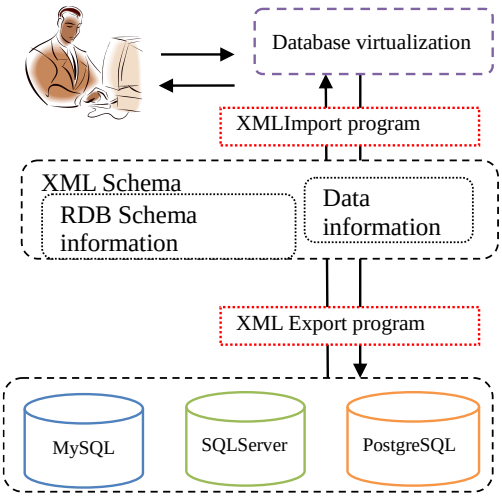

Figure 2: Virtualization technique for RDB databases.

3.2 RDB Schema Conversion into XML

The following describes how the RDB schema is converted into XML. Fig. 3 presents results of reading the schema information from the RDB and converting it into XML. The RDB schema information that is converted into an XML format includes "table names", "field names" (associated data types and default values), and "constraints" (primary key constraint, unique constraint, check constraint, NOT NULL constraint, and foreign key constraint) capability.

Regarding the XML tree structure, we described the table information in the table structure node with its elements of Field="column name", Type="data type", Null="TRUE or FALSE" (NOT NULL constraint). We described the schema information in the schema node with its elements of TYPE= "constraint name", Table= "table name", Column= "column name", ReTable= "referenced table name", ReColumn= "referenced column name", and Check= "rule".

3.3 RDB Data Conversion into XML

The manner in which the RDB data are converted into XML is described next. Fig. 4 portrays results of reading the data information from the RDB and conversion into XML. Because of the XML tree structure, we had dbname="database name", tblname="table name", and the actual data columns succeed.

3.4 Virtualization of Databases

We discuss the virtualization of modelled DBs of different types. For virtualization of different types of modelled DB, we describe the schema information of each model using a single common schema. The common schema we will use is an XML Schema. Around it, we will perform virtualization. Fig. 1 shows a virtualization method for different database types. To accomplish schema conversion from a different modelled database, we first get the schema information from an RDB to work on. Then we convert it into the correct XML schema for that RDB. We currently have to re-build

```
<?xml version="1.0" encoding"UTF-8" standalone="yes"

 <root>

<rdb Name="mysql">

   <database Name="questionnaire" >
      <table_structure Name="member">
      <field Field="samplenum"
              Type="integer" Null="FALSE"  Default="  />
   <field Field="answerday" Type="text"
     Null="FALSE"  Default="  />

   ….

 </table_structure>

 <schema>
  <constraint Type="PRIMARY KEY"
      Table="member"  Column="samplenum"  />

….

      <constraint Type="FOREIN KEY" Table="questionnaire"
      Column="samplenum" Retable="member"
      ReColumn="samplenum" />

….

 </schema>
</database>
</rdb>
</root>
```

Figure 3: Example of RDB schema information conversion into XML.

Code	Name	Latitude	Longitude
47401	Wakkanai	45.25	141.41
47404	Haboro	44.22	141.42
…	…	…	…

RDB

```
<?xml version="1.0" encoding="utf-8" ?>
<dataset dbname="chihou">
    <data tblname="AreaInfo ">
      <Code>47401</Code>
      <Name>Wakkanai</Name>
      <Latitude>45.25</Latitude>
      <Longitude>141.41</Longitude>
   </data>
     <data tblname=" AreaInfo ">
      <Code>47404</Code>
      <Name>Haboro</Name>
      <Latitude>44.22</Latitude>
      <Longitude>141.42</Longitude>
   </data>
   <…>
```

Figure 4: Example of actual RDB data conversion into XML.

the XML tree with our schema conversion module when the distributed database is redefined.

Table 1 presents schema conversion correspondences between the two. Because any XML DB is already described in the XML format, we extract the schema information without conversion. On the other hand, when the data are manipulated, our query conversion module automatically transfers the access results to the application program.

3.5 Techniques of Execution Efficiency Improvement

Methods of the execution efficiency improvement of virtualization processing (improvement in the speed) are as follows.

- The place of virtualization processing

In order to accelerate, the virtual database environment which uses load sharing technology and parallel processing technology is shown in Fig. 5, and we use both user side virtual DBMS and data side virtual DBMS.

Since the database is distributing through a network and communication time influences the whole processing time greatly, it becomes important to reduce the amount of data transfer for the improvement of the processing speed. Under the virtualization processing the data volume changes. Even if the same data is processed, data volume differs by the schema expression, RDB schema or XMLDB schema. Therefore, the place where the virtualization processing is performed could be changed, so that the amount of data transferred is reduced, and communication time is reduced.

- Database selection

When the same table and data are stored in different databases, it could be considered to make the load of each database uniform by acquiring data from a database with little load. In database virtualization technology, since virtual processing is added in addition to processing of the usual database, balancing of the database load becomes important.

4 DESIGN OF THE SIMULATOR

In this section, we design a simple model based simulator based on the database virtualization technique described in Section 3. Only two types of DBs, e.g. RDB and XMLDB, are considered here.

In improving the execution efficiency, it is necessary to measure the execution performance of the virtualization processing, especially in a distributed environment where multiple databases are connected via a network. But it takes a lot of databases and large-scale network structure, and preparation of actual measurement environment is costly and very difficult. Therefore, the measurement environment using a simulator is considered.

Two of the followings are the basic requirements needed by the simulator.

- Measurement for discovering the causes (bottlenecks) of delay of database virtualization processing can be performed.
- Measurement when the number of databases connected or the volume of each database becomes large on the virtual database environment using load sharing technology and parallel processing technology can be performed.

Table 1: SQL and associated XML

	SQL	XML
Table definition	CREATE TABLE table name…	<xsd: element name="table name"…
Column definition	CREATE TABLE… column name...	<xsd: element name="column name"…
Data type definition	CREATE TABLE… data type..	<xsd: element… type="data type"…
Default values	CREATE TABLE… column name DEFAULT value	<xsd: element… default="value"…
Primary key constraint	PRIMARY KEY	<xsd: key…
Unique constraint	UNIQUE	<xsd:unique …
Foreign key constraint	FOREIGN KEY	<xsd: keyref … refer =…
NOT NULL	NOT NULL	<xsd:… nillable="false"...
Method	CREATE METHOD	
Inheritance	CREATE TABLE… UNDER upper level table name	<xsd: complexType …

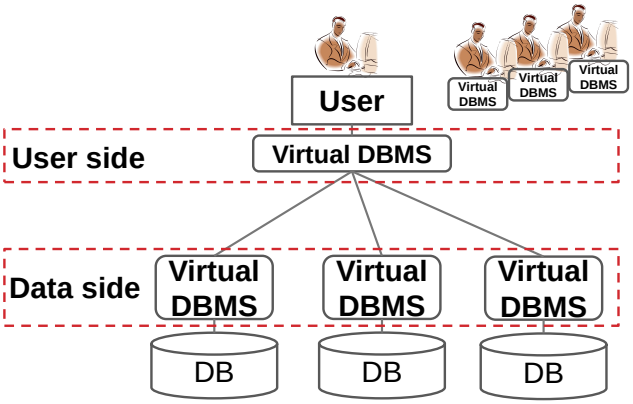

Figure 5: Virtual database environment which uses load sharing technology and parallel processing technology.

4.1 Outline of the Simulator

The main purpose of the simulator is the bottleneck discovery of database virtualization processing. For the purpose of this bottleneck discovery, actual processing, such as virtualization processing, database processing and communication processing, are not needed and actual processing is not performed in the simulator. The execution efficiency is computed simulating and integrating each processing time. Random elements such as network

congestion, user's command input timing are simulated and computed repeatedly to obtain average and variance.

Prerequisites for database access for the simulation are specified as follows. The data mining and distributed database environments are considered in the simulator and it assumes a limited range of database operations here. For example, database updating and join operations are excluded in the simulator.

By realizing each component such as database processing of the simulator as a process and performing inter-process communication with TCP protocol, the simulator can be implemented on a single PC or on two or more PCs. Followings are prepared as an item which can be changed by setup.

- Number of users
- Number, scale, and kind of databases
- Network line speed

4.2 Measurement Items

The following measurement is performed for the overhead identification of virtualization processing. About the reference parameters for the simulation, some preliminary simple virtual processings are performed beforehand and they are determined from the result at the time of implementation.

- Time of virtualization processing

This mainly considers time of conversion such as query conversion from XQuery to SQL and result conversion from RDB result into XML format. The measuring method computes and converts the processing time according to the length of a query, the data volume of the result, etc. based on the reference parameters.

- The change in the data volume after virtualization processing

The data volume fluctuated by virtualization processing of query result is measured.

- Processing time of a database

The processing time of a database is computed from a query. For example, in 'Selection', processing time changes by the existence of indexes. Processing time is changed also by the timing of the database usage and the number of users. If there are some database processing performed during system usage of a user, the wait time of the database processing will be added to the processing time for the user.

- The amount of data transfer

The data transfer rate is adjusted by changing the network utilization factor according to the number of users, users' usage timing, etc. of databases. The system determines the amount of data volume by what kind of query is issued to which database by each user, then decides the amount of data transfer by which network is used for the data transfer.

- Communication time

Communication time is computed using the following formulas.

$$Communication\ time = \frac{Amount\ of\ data\ transfer \times (1 + Rate\ of\ control\ data)}{Line\ speed \times Network\ utilization\ factor}$$

Since the network of a database is classified to class 3 in Network Quality of Service (QoS) of Y.1541 of ITU, delay by congestion is generated in the probability of 10^{-3} based on the class 3 of QoS. Time to be delayed in this case, being unspecified in the class 3 of QoS and not restricted, we make it the interval of the retransmission-of-message packet. The process on the data reception side performs the measurement of communication time.

4.3 Size of Packet

Packet size is needed for the determination of the rate of control data or the number of times of communication. The maximum size (MSS: Maximum Segment Size) of the packet changes with MTU (Maximum Transmission Unit) of the data link assuming that the database uses TCP.

The main current data links are Ethernet and PPPoE, and assuming the protocol uses TCP, MTU of Ethernet is used.

The maximum data volume per packet is set to 1460 bytes, and the number of times of communication is (Amount of data transfer /1460) and the rate of control data is (1-1460 / 1518).

4.4 System Configuration

Each component is realized by a process so that the each component, such as virtual DBMS, can be executed concurrently. Each component performs inter-process communication with TCP protocol, and the simulator is run on a single PC or two or more PCs. Development language is C and execution environment is Linux.

In order to decide to implement virtualization on user side or data side depending on the measurement result, virtualization process could be performed on both sides. Although designed supposing virtualization of RDB and XMLDB at this time, when adding virtualization of other DB kinds, it is made to be easy to extend. By saving the last setting environment in a file, and calling it easily, the time and effort for the setup for every simulator use is reduced.

The system configuration of the simulator based on Fig. 5 is shown in Fig. 6. And the component processes of the simulator are classified into following three.

- Interface process for the simulator user

Processing of a simulator user's interface and management of the whole simulator are performed. The setup of the simulator and directions of a simulation start are performed.

- User's process

Processing corresponding to each user using a database is performed. Execution of XQuery, reference of an XML schema, etc. are performed and processing time is sent to the interface process for the simulator user. In a communication module, calculation and conversion of communication time are performed from the data volume of the received result. In a virtualization process module, calculation and conversion of time of virtualization processing from the data volume of a result are performed.

- Handling process of each database

Processing of data side virtual DBMS and database accesses are performed. The processing time for processing of a database and virtualization processing according to a setup of the number of data etc. is computed and converted.

In a communication module, calculation and conversion are performed for the communication time of query reception based on the received query. By DB module, calculation and conversion of the processing time concerning query execution are performed and data size or the number of data of the result data are determined. In the virtualization process module, calculation and conversion of time for virtualization of data from the number of result data, etc. are performed.

In a communication module, since a transmitting side process does not need to consider the existence of delay, such as a collision, about measurement of a communication time, the communication module of the receiving side process measures communication time. Specifically, measurement of communication time in case a command is sent to data side virtual DBMS from user side virtual DBMS is performed by the database side communication module and in case a result is sent to user side virtual DBMS from data side virtual DBMS, measurement is performed by the user side module.

5 REFERENCE PARAMETERS FOR THE TIME OF VIRTUALIZATION PROCESSING

In this section, we determine necessary reference parameters for the simulator model in Section 4. The parameters are determined using measurements taken from real virtualization processing and database access.

Simple and preliminary virtualization processing was performed and the reference parameters of the processing time of virtualization processing and the fluctuation of the data volume after virtualization processing were determined. Although implementation was carried out in Java by the previous work [7], since Java operates on a virtual machine and delay by insufficient memory occurs, we re-implemented the system in C.

At this stage, since database virtualization of only RDB and XMLDB is assumed, only the reference parameters of these virtual processings are obtained. Moreover, execution using an actual database is not performed about processing of a database, but the function which returns dummy data is prepared. The execution environment of preliminary virtualization processing is as shown in Table 2.

The reference parameters obtained in this section are the references only for the environment shown in Table 2. The reference parameters should be reconsidered and modified under other environments.

About the composition of a database, RDB 'Chihou' assumes the database with the table and column shown in Table 3, and assumes the XMLDB database 'Tenkou' which is shown in Fig. 7.

The XQuery used for the execution is as follows.

```
for $A in fn:doc('Tenkou')//Item let $B :=
fn:doc('Chihou')//areainfo[@Code=$A/Station/Code]
let $C := fn:doc('Chihou')//observ[@Code=$A/Station/
Code] return <result>{$B/@Code, $B/Area, $B/Kana,
$C/Observ, $A//Precipitation, $A//Precipitations}
</result>
```

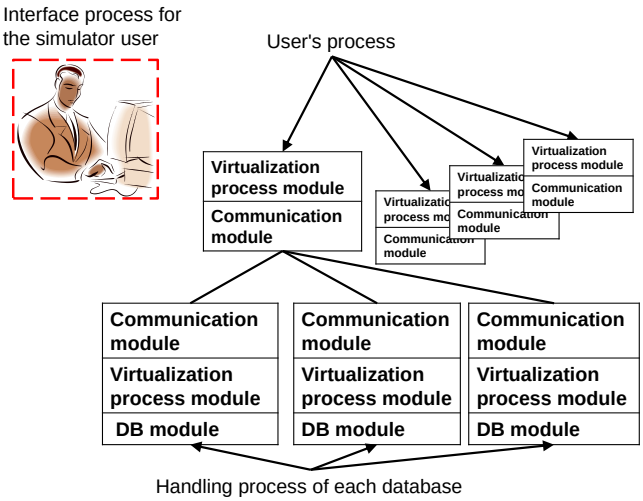

Figure 6: System configuration.

Table 2: Preliminary virtualization process execution environment

OS	Windows 8.1 pro 64bit
CPU	Core i5-3317U1.70GHz 2threads
Memory	4GB

Table 3: Structure of RDB 'Chihou'.

Table name	Column name
Areainfo	Code, Area, Kana
Observ	Code, Observ

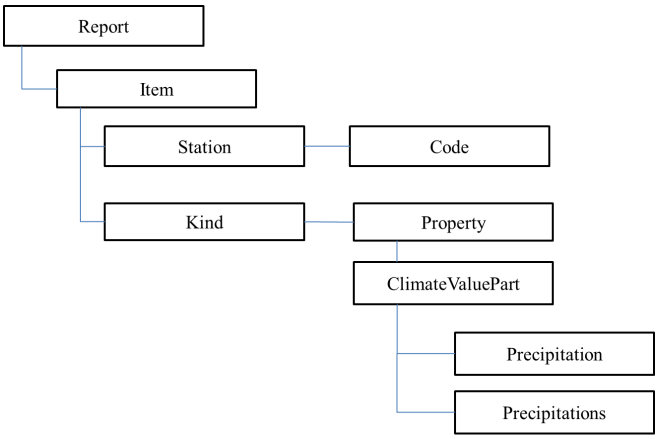

Figure 7: Structure of XMLDB 'Tenkou'.

Table 4: The virtualization processing time of the execution result of the query of RDB (microseconds).

	Number of columns		
Number of result data	1	3	5
500,000	657,763	1,660,663	2,658,075
1,000,000	1,280,225	3,270,225	5,205,550
2,000,000	2,501,350	6,092,450	10,225,800

Table 5: The virtualization processing time of the execution result of the query of XMLDB (microseconds).

	Number of items		
Number of result data	1	3	5
500,000	855,000	2,285,700	3,721,778
1,000,000	1,714,250	4,554,200	7,202,556
2,000,000	3,398,000	8,741,800	14,397,000

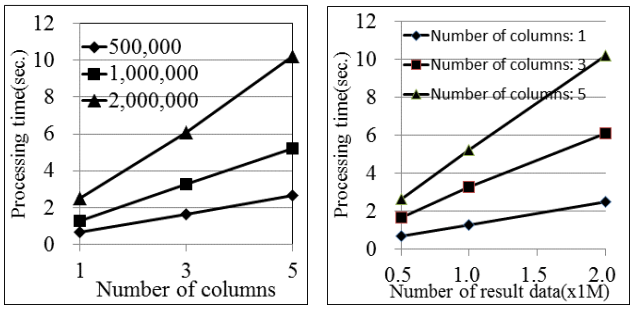

Figure 8: Graph of the virtualization processing time of the execution result of the query of RDB

This XQuery is a query which acquires data from RDB named 'Chihou' and XMLDB named 'Tenkou'. It is the query of returning the result which acquired from 'Chihou' of the 'let' phrase based on the result of 'Tenkou' acquired with the 'for' phrase, in the form described after 'return' phrase. From the simple execution result of virtualization processing, virtualization processing of a query execution result has measured time.

To determine the reference parameters, queries for above mentioned processing which return 500,000, 1,000,000 or 2,000,000 result data, are created and executed multiple times. From the execution results, reference parameters are determined as shown in Table 4, 5.

For the virtualization processing time of the execution result of the query of RDB, it is proportional to the number of result data and the number of columns, as shown in Fig. 8. Moreover it can be expressed by a linear equation of 0.953 microseconds of inclination and 0.208 microseconds of intercept of the number of columns. Value by these parameters and actual measurement are shown in Fig. 9.

For the virtualization processing time of the execution result of the query of XMLDB, it is proportional to the number of result data and the number of items, like RDB. Therefore, it can be expressed by a linear equation of 1.393 microseconds of inclination and 0.317 microseconds of intercept of the number of item.

The determined reference parameter of each processing time is shown in Table 6. As mentioned before, these reference parameters are the references only for the environment shown in Table 2. But, the main purpose of our simulator is the bottleneck discovery of database virtualization processing. So, we do not need to know absolute virtualization processing time. We need to know the relative ratio between the virtualization processing time and the database processing time. Although the database processing time is not shown yet in this paper, it should be measured in the same environment as this time, and we could use it.

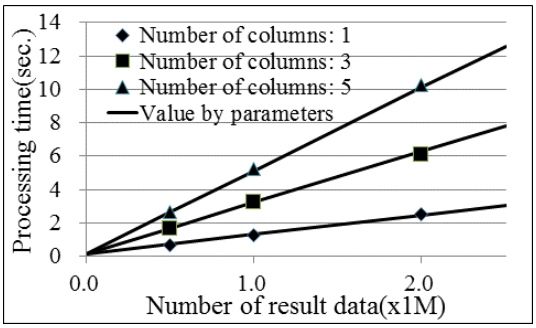

Figure 9: Value by reference parameters and actual measurement of the query of RDB

Table 6: Reference parameter of processing time (microseconds).

Processing	Processing time
Virtualization processing time of the execution result of the query of RDB	(0.953 x Number of columns +0.208) x Number of result data
Virtualization processing time of the execution result of the query of XMLDB	(1.393 x Number of items +0.317) x Number of result data

6 CONCLUSION

In this research, the design and implementation of the simulator which measure the execution efficiency of the database virtualization processing in the distributed environment where multiple heterogeneous databases were connected with the network have been performed.

However, verification and evaluation of this simulator itself is left yet. Therefore, it is necessary to advance to the next stage of performing verification and evaluation of the simulator, and perform quantitative measurement of database virtualization processing. From the result, we discover the bottleneck of database virtualization processing, and plan to accelerate the bottleneck parts in the future.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Number 24500122.

REFERENCES

[1] D. F. Garcia, "Performance Modeling and Simulation of Database Servers." The Online Journal on Electronics and Electrical Engineering Vol.2, No.1, pp.183-188 (2010).

[2] W. Wu, et al., "Predicting query execution time: Are optimizer cost models really unusable?." IEEE 29th International Conference on Data Engineering (ICDE), pp.1081-1092 (2013).

[3] W. Wu, et al., "Towards predicting query execution time for concurrent and dynamic database workloads." Proceedings of the VLDB Endowment, Vol.6, No.10, pp.925-936 (2013).
[4] K. Mori, S. Kurabayashi, N. Ishibashi, and Y. Kiyoki, "An Active Information Delivery Method with Dynamic Computation of Users' Information in Mobile Computing Environments." DEWS2004 1-A-04, (2004). (in Japanese)
[5] Teiid: http://www.jboss.org/teiid, Red Hat
[6] DB2: Information Integrator V8.1, http://www.jpgrid.org/documents/pdf/WORK4/sugawara_ws4.pdf
[7] Y. Wada, Y. Watanabe, K. Syoubu, H. Miida, J. Sawamoto, "Virtual Database Technology for Distributed Database in Ubiquitous Computing Environment," American Journal of Database Theory and Application, Vol. 1, No.2, pp.13-25 (2012).
[8] Y. Wada, Y. Watanabe, K. Syoubu, J. Sawamoto, and T. Katoh, "Virtualization Technology for Ubiquitous Databases," Proc. 4th Workshop on Engineering Complex Distributed Systems (ECDS), pp.555-560 (2010).
[9] Y. Wada, Y. Watanabe, K. Syoubu, J. Sawamoto, and T. Katoh, "Virtual Database Technology for Distributed Database," Proc. IEEE 24th International Conference on Advanced Information Networking and Applications Work-shops (FINA2010), pp.214-219 (2010).
[10] Y. Wada, Y. Watanabe, K. Syoubu, H. Miida, J. Sawamoto and T. Katoh, "Technology for Multi-database Virtualization in a Ubiquitous Computing Environment," International Workshop on Informatics (IWIN2010), pp. 89-96 (2010).

(Received October 23, 2014)
(Revised February 9, 2015)

Daichi Kano received M.S. degree in 2015 from Iwate Prefectural University, Japan. His research interests include distributed parallel processing and simulation. He is currently working for Tokyo Computer Service Co., LTD.

Hiroyuki Sato is currently a Professor of Faculty of Software and Information Science, Iwate Prefectural University, Japan. He received the B.E. in information engineering from Tsukuba University in 1982. He joined Mitsubishi Electric Corporation in 1982. He received his PhD degree from Tsukuba University in 2003. His research interests include parallel processing, and high performance computing. He is a member of IPSJ, IEICE and IEEE-CS.

Jun Sawamoto is currently a Professor of Faculty of Software and Information Science, Iwate Prefectural University, Japan. He received the B.E. and M.E. in mechanical engineering from Kyoto University in 1973 and 1975. He joined Mitsubishi Electric Corporation in 1975. He received his PhD degree from Tokyo Denki University in 2004. His research interests include ubiquitous computing, human-interface system, multi-agent systems, and cooperative problem solving. He is a member of IPSJ, IEEE-CS, ACM.

Yuji Wada received the B.E. and the M.E. in electrical engineering from Waseda University in 1974 and 1976, respectively. He joined Mitsubishi Electric Corporation in 1976. He received the PhD degree in computer science from Shizuoka University of Japan in 1997. He is currently a Professor in the Department of Information Environment, Tokyo Denki University. His research interests include database systems, data mining, and recommendation. He is a member of the IPSJ, the IEICE, the JSAI, the JSSST and the DBSJ.