ISSN 2277-3061

Performance Evaluation of Distributed Database Systems

Khaled Saleh Maabreh
Assistant Professor, Computer Information System Department, College of Science and Information Technology, Zarqa University, Jordan

ABSTRACT

Distributed database management systems manage a huge amount of data as well as large and increasingly growing number of users through different types of queries. Therefore, efficient methods for accessing these data volumes will be required to provide a high and an acceptable level of system performance. Data in these systems are varying in terms of types from texts to images, audios and videos that must be available through an optimized level of replication. Distributed database systems have many parameters like data distribution degree, operation mode and the number of sites and replication. These parameters have played a major role in any performance evaluation study. This paper investigates the main parameters that may affect the system performance, which may help with configuring the distributed database system for enhancing the overall system performance.

Keywords:

Distributed database; Locking; Concurrency control; Performance evaluation; Transaction.

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS AND TECHNOLOGY
Vol. 13, No. 9
editorijctonline@gmail.com
www.ijctonline.com, www.cirworld.com

INTRODUCTION

Recently, there has been an increasable interest in distributed database systems, in which the databases are distributed among multiple processing units. The remote access to some data objects that are distributed over the system is provided by predefined techniques. Since the access of non local data is expensive in terms of the communications overheads, data for example, can be replicated to avoid remote access for some types of queries such as read only query

Distributed databases are defined as a collection of multiple logical interrelated databases, distributed over a computer network [7, 16, 19]. Distributed database management systems manage large volumes of data and large number of users that increasingly grows. Users have accessed the data through different types of queries transparent from lower level of details. Data contained in a distributed database system must be organized in such a way that is satisfying the user requirements within an acceptable time [2, 16, 20]. There are many system parameters that are affecting the performance in terms of decreasing the overall waiting time and enhancing the execution time in order to increase the user satisfaction. Some of these parameters are; the number of sites that the distributed database system consists of, degree of data replication and the mode of the queries that are contained within the user transaction [4, 15, and 17].

This paper is an extended work of [12] that is investigating such of these parameters by evaluation of 3 sites [12], which represent a limited number of sites that may produce insufficient results. In order to obtain more significant results and produce real interpretations and conclusions, the number of sites is extended in this paper to *m* sites as well as the number of database objects is increased. Moreover precise evaluation means more helping in configuration the distributed database system for enhancing the overall system performance.

Transaction in a distributed database consists of several participants to execute over all requested sites; these participants must guarantee that the changes to data will be saved in order to commit the transaction. . If any of the participants fails, the entire transaction aborts. As a two-phase locking policy, data items must be locked before use, this strategy can be accomplished by a lock manager. There are many approaches according to where the lock management is performed: one of them is the centralized locking, which is used in this paper for simplicity of implementation and management. In this approach, there is one site responsible for granting locks thus, the central site has a lock table for the entire database. Communications among other sites are performed via transaction manager at the initiated site of the transaction, the lock manager at the central site, and the processor at other sites that is participating to run the operations [16]. In terms of the update operation, the effect of local commits extends to distributed transactions by involving all sites that contain a database item to agree before committing the main transaction.

To investigate the performance evaluation system under the distributed two-phase locking with different workloads, a discrete event and full parameterized simulation program is developed by using Java technology, to analyze the incidence of the main parameters that are selected in this paper: number of sites, operation mode and degree of replication. The simulation program will generate the database objects randomly. These objects will be then distributed into a multiple sites with different replication degrees and different operating modes. Finally, data produced by the simulation program will be gathered and analyzed to evaluate the system behavior. The rest of this paper is organized as follows: Section 2 presents the preliminary studies and related works. The proposed approach will be presented in Section. Discussion and analysis of a collected data will be listed in section 4 and section 5 is the conclusion.

BACKGROUND

Some of the system parameters are studied by several researchers for evaluating the system performance. The performance of a real time distributed database is proposed by Blesa and Zambardino [3] through introducing an abstract model for analyzing the most important methods of concurrency control. Their model was evaluated many parameter effects in the concurrency control environment like the number of items and the number of copies of each item and how they affect the average waiting time of each transaction. Number of data items and control messages appeared very high and causes heavy network traffic as it analyzed by their model. Studying the influences of the degree of replication on the distributed database system performance through analytical model is reported in [5, 14]. Those studying a model show that the degree of replication has the most effect by enhancing the system performance especially with read-only transactions. The tradeoff of replicated data in a distributed database environment is also presented by Ciciani et al., [6], by implementing and developing an analytical model against some of concurrency control protocols which is proving the significant improvement in the response time with additional time requirements to maintain consistency among the replicates. Yadav and Agarwal [21] examine the conflicts detection and resolution among replicated database and how these conflicts affect the system performance. By comparing the four concurrency control algorithms, their analysis data shows that the optimistic protocol could behave better in a replicated database.

The performance of centralized, partitioned and replicated database architecture by using TPC-C benchmark have been proposed and implemented to simulate a business and commercial environments by shipping et al., [18]. Their study includes a comparison details between centralized and distributed environments which provides useful information in determining the appropriate database architecture which may help administrators in moving from centralized to distributed environments. A design requirement for a Peer-to-Peer distributed database is addressed by George and Balakrishnan [8] by framing architecture named Flexi Peer cluster. Their architecture facilitates the data processing by addressing the fragmentation and allocation phases of database design. So, the sites can effectively store and produce results for queries instead of wasting time by waiting for unavailable data on that particular site. Hababeh et al., [9] proposed a method for grouping and allocating fragments into site cluster based on their communication cost for minimizing the communication cost among sites, and enhancing the system performance. Their method of clustering increases the availability and reliability of replicas which enhances the overall system performance. Analyzing of a distributed database system main parameters are presented in [12] which shows the incidence of these parameters on the overall system performance, but this study evaluates a limited number of sites. An advance work to this study is the main goal of this paper

PROPOSED APPROACH

A homogeneous environment of a distributed database system which consists of *m* database servers (sites) interconnected by a computer network is proposed in this study. Sites are assumed to be equivalent in which they have the same arrival rate, and each site has a subset of data that is accessed by the users locally or remotely. To evaluate an update operation, a primary copy approach is used in which there one copy considered as a master copy is located at one site. The details of the system parameters are summarized in Table 1.

Table1: (Simulation Parameters)

Parameter	Description	Values
Num-of-sites	Number of sites in the system	M
Num-of-DB	Number of databases in each site	1
Num-of-obj	Number of database objects for each site	10000
Num-of-replicas	Degree of replication	0.2, 0.4, 0.6, 0.8
Num-of-tables	Number of tables in a database	100
Num-of-trans	Number of transactions in the system	1000
Min-trans-size	Minimum number of operations	1
Max-trans-size	Maximum number of operations	25
Mode	Operation mode	R, RW, W
Queue-length	Maximum queue length	30
Chk-time	Mean time to check a lock	1 *ms*
Set-set	Mean time to set a lock	1 *ms*
Rel-time	Mean time to release a lock	1 *ms*
Acc-time	Mean time to access a data object	10 – 100 *ms*

The database tables are partially replicated over *m* sites as one-dimensional partial replication (some objects to all sites) [14]. Because the database is assumed to be virtually organized and to make the analysis simple and more practical, 50 sites are chosen to be studied in this research paper. The simulation program will fill these tables randomly with 500 rows for each table, and then distributes these tables across the system as one copy for each object as a master copy located randomly at exactly one site and the other copies are secondary or replicas. This will be done through three stages according to the degrees of replication mentioned in Table-1. The transactions are also generated randomly with different DML modes, (R means that all operations of the transactions will be read, RW means that mixed of read and write operations will be considered and W for write operation).

➢ Lock Manager

As a part of the simulation program, the lock manager will be implemented through three phases as follows: First, build a single key hash table containing locks with resource identifier for checking the conflicts; a hash table is used because of fast content based retrieval [1]. Second, build a queue attached to the same resource for managing the waiting transactions, because multiple locks can be held for the same resource (i.e. Shared locks), and multiple lock requests can be waiting for that resource. Finally, build and implement a transaction control block to determine if all locks belong to the same transaction, in order to release all locks held by that transaction at once.

➢ Deadlock Detection

Deadlock is a problem occurs among a set of two or more transactions when each transaction T in a set is waiting for an item locked by other transaction T’ in the set [7]. When a deadlock occurs in the systems, the overall system performance will be degraded until is resolving the deadlock. So as long as the deadlock persists in the system, the system performance and throughputs are decreased [10, 11]. Then, any performance study must avoid a deadlock problem during the execution in order to get more precise data.

To make the analysis easy and simple, the timeout approach of predefined value for deadlock detection will be used [13]. In this paper, the check for an available database item is assumed to take one millisecond, and one millisecond is needed between two successive trials. If the lock is granted, a random number between 10 and 100 milliseconds is chosen as a processing time; so 51 trials for acquiring a lock is sufficient to determine if the deadlocked problem is occurred.

DISCUSSION

Simulation program was run to evaluate the performance of distributed database system. The simulation setup consisted of 50 interconnected sites. Database objects are distributed over these sites by different degrees of replication. The simulation involved 150 trials into three stages, 50 trials in each stage with a different set of randomly generated numbers. Each trial is involving 200 transactions for each stage (read-only, write-only and mixed of read and write transactions). After execution of the simulation program by changing the parameters mentioned in Table 1, data are collected for each stage by changing the degree of replication from 0.2 to 0.8 and also changing the number of sites from 2 to 50. Data collected for read-only transactions are presented in Table 2. The simulation results show that average execution time depends on the degree of replication and the number of sites, because when the degree of replication becomes high, most of the generated transactions are working locally due to availability of data, this behavior can be shown in figure 1. So, the degree of replication 0.8 makes the average execution time for the running transactions be less than the average execution time for the degree of replication 0.6. The same thing happens when the degree of replication 0.4 or 0.2.

Tables 2 and 3 present the simulation results for write-only and mixed of read and write transactions respectively, the average execution time becomes higher than producing in a read-only transaction because of update propagation problem in order to make the database consistent. So, when the degree of replication is chosen to be 0.8, higher average execution time is produced, thus, the difference between these times is cleared by comparing the write-only and the mixed of read and write transaction. Figure 2, shows the system performance under write-only transactions which also indicate the effect of the system by both of the number of sites as well as the degree of replication. By considering a constant value for the degree of replication and varying values for a number of sites, we can notice that the average execution time is linearly increased because of complexity of update in case of write operation. The same notice can be shown in figure 3 when using mixed of read and write transactions. So, the number of sites has a direct promotion effect to the system performance when read-only operations are executed. Degree of replication has the most effect, especially in writing or mixed of read and write operation in an inverse proportionality manner. By inspecting the figures 2 and 3 that shows the system performance of write and mixed of read and write transactions, it is obvious that the system behaves on the degree of replication 0.4 better than on degree 0.6 and 0.8. So, the degree of replication recommends not exceeding 40% of total objects in order to preserve better system behavior.

Table2: (Read-only Transactions)

Number of sites	**Average Execution Time According to the Degree of Replication**			
	Degree=0.2	**Degree=0.4**	**Degree=0.6**	**Degree=0.8**
2	779	655	598	432
3	1430	1101	990	640
4	1889	1538	1200	820
5	2518	2000	1580	1080
6	3400	2430	1750	1140
7	3777	2780	1930	1170
8	4300	3200	2280	1200
9	5042	3450	2400	1240
10	5490	3830	2450	1350
15	6280	4300	2750	1500
20	6941	4760	2920	1540
25	7570	5290	3160	1500
30	7822	5630	3340	1580
35	7880	5660	3380	1546
40	7950	5740	3380	1521
45	8030	5800	3400	1496
50	8100	5800	3450	1500

Table3: (Write-only Transactions)

Number of sites	Average Execution Time According to the Degree of Replication			
	Degree=0.2	Degree=0.4	Degree=0.6	Degree=0.8
2	664	817	679	663
3	742	1068	1328	1790
4	963	1314	1995	2500
5	1012	1455	2651	3310
6	1158	1890	3322	4060
7	1490	2640	3980	4800
8	1890	3310	4636	5800
9	2600	4000	5400	6800
10	3250	5070	6581	8110
15	4000	6150	7530	9400
20	4670	7400	8600	10680
25	5340	8450	9720	12000
30	6200	9623	10970	13310
35	6900	10480	11800	14520
40	7510	11254	12800	15510
45	8000	12045	13520	16555
50	8520	13125	14470	17650

Table4: (Mixed of Read and Write Transactions)

Number of sites	Average Execution Time According to the Degree of Replication			
	Degree=0.2	Degree=0.4	Degree=0.6	Degree=0.8
2	708	964	1370	2041
3	1180	1394	2027	2704
4	1451	2076	2686	3368
5	1900	2722	3342	4025
6	2490	3375	4000	4684
7	2940	4029	4660	5730
8	3500	4710	5320	6340
9	3965	5374	5976	7200
10	4540	6033	6970	8450
15	5680	7783	8400	9721
20	6800	9102	9974	11460
25	7940	10800	12400	13428
30	8900	12100	13940	15110
35	9600	12895	14750	15899
40	10158	13555	15290	16600
45	10490	14260	16000	17200
50	10970	14710	16555	17680

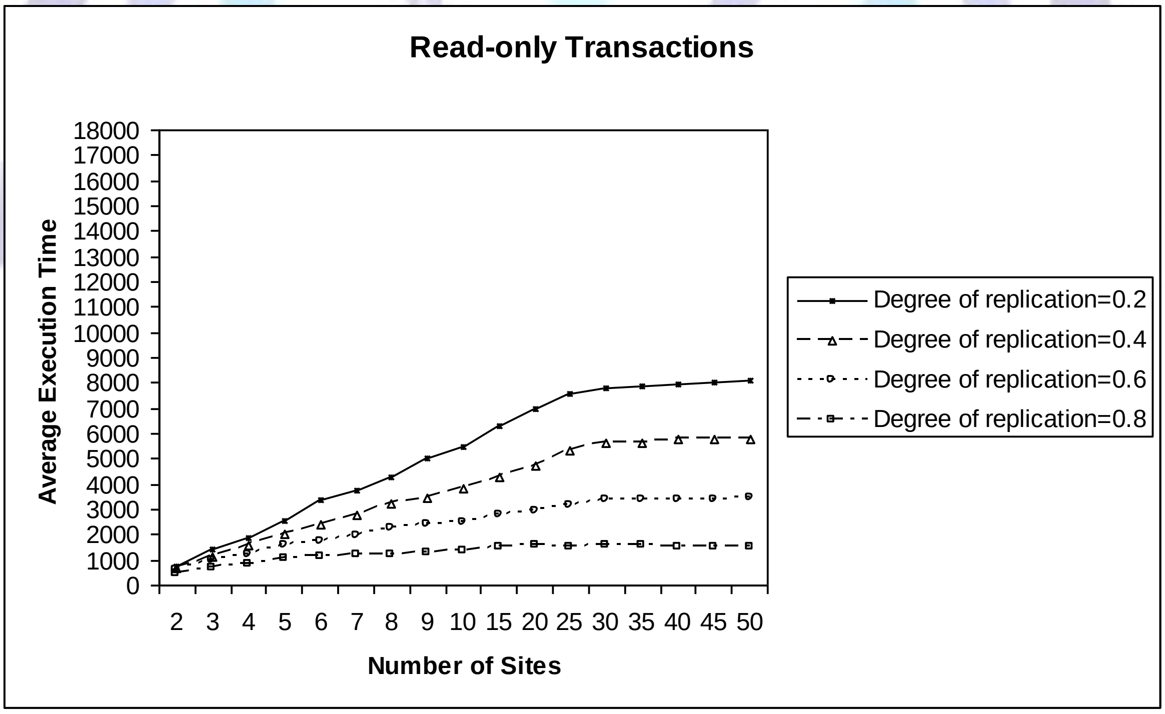

Figure1: (Read-only Transactions)

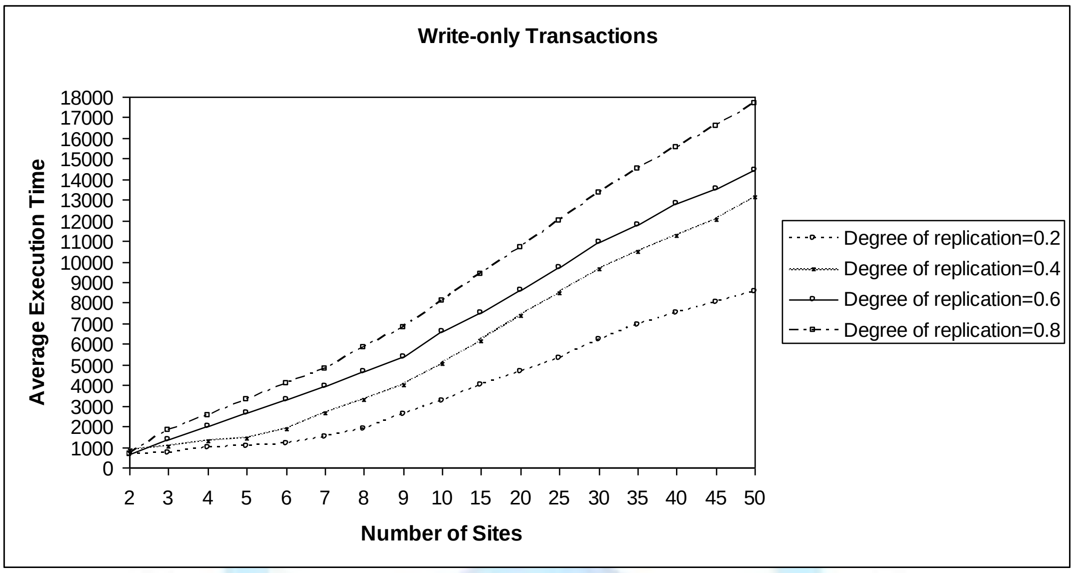

Figure2: (Write-only Transactions)

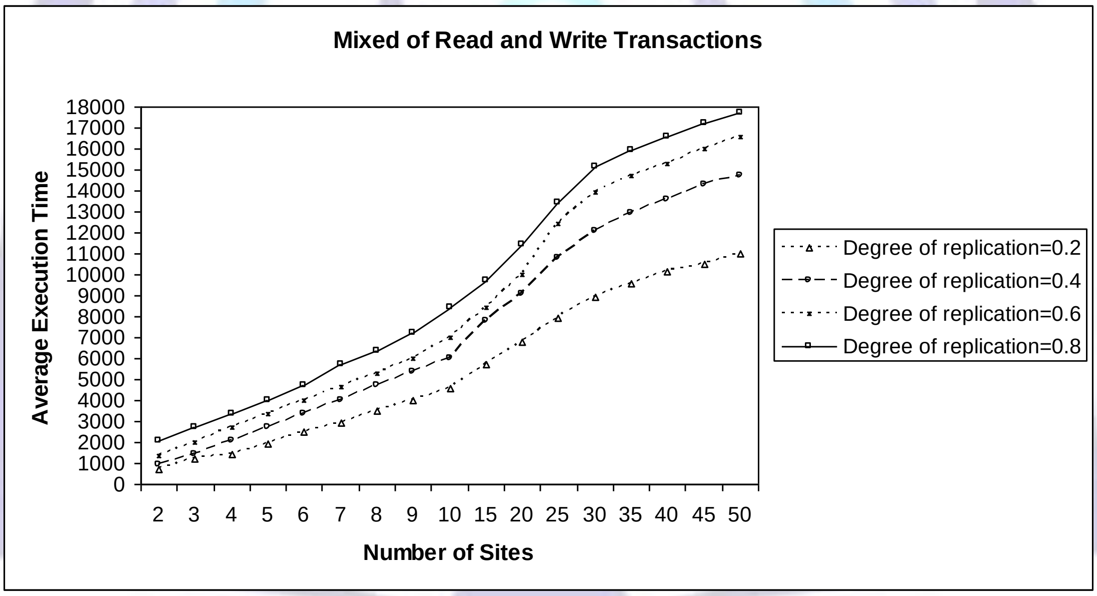

Figure3: (Mixed of Read and Write Transactions)

CONCLUSION

Discrete events and full parameterized simulation program were conducted to evaluate the performance of distributed database system. Data containing in these systems are varying in types from texts to images, audios and videos that must be available through an optimized level of replication. This paper investigates the main parameters that may affect the system performance, which may help in configuring the distributed database system for enhancing the overall system performance. The parameters that are evaluated, in this paper, are the number of sites that the system may consist of, the number of replicas allocated and operation mode. The simulation results for read-only transactions indicate that average execution time depends on the degree of replication and the number of sites, because when the degree of replication becomes high, most of the generated transactions are working locally due to availability of data. So, the system behaves better when the degree of replication becomes high. But, the degree of replication has the most effect, especially in writing or mixed of read and write operation in an inverse proportionality manner. To preserve better system performance in the systems in which most of transactions request write or mixed of read and write operations, the degree of replication is recommended to not exceed 40%, because of update complexities to preserve database consistency.

ACKNOWLEDGEMENT

This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan.

REFERENCES

[1] Bernstein P. A., Hadzilacos V., and Goodman N. "Concurrency Control and Recovery in Database Systems", Addison-Wesley, 1987.

[2] Bernstein P. and Newcomer E., "Principles of Transaction Processing for the Systems Professional", Bentham Press, 2004.

[3] Blesa P. and Zambardino R. “Incidence of parameters in the performance of a distributed database”, Annual Review in Automatic Programming, Vol. 15, Part 1, pp 41–46, Elsevier Science, 1990.

[4] Boral H. and DeWitt D. J., "A Methodology for Database System Performance Evaluation" on the proceedings of the International Conference on Management of Data, ACM SIGMOD, New York ,USA 1984.

[5] Burger A., Kumar V. and Hines M., "Performance of Multiversion and Distributed Two- Phase locking Concurrency Control Mechanisms in Distributed Databases", Information Science, Vol. 96, Issue 12, P129, 24P, 1997.

[6] Ciciani B., Dias D. M. and Philip S. "Analysis of Replication in Distributed Database Systems", IEEE Transactions on knowledge and data engineering. Vol. 2, No. 2. June 1990.

[7] Elmasri R. and Navathe S.B. "Fundamentals of Database Systems", 6th edition, Pearson Addison Wesley (Boston), 2011.

[8] George A. and Balakrishnan C.,"A Study on Performance Evaluation of Peer-to-Peer Distributed Databases", IOSR Journal of Engineering, Vol. 2(5) pp: 1168-1176, 2012.

[9] Ismail O. Hababeh, Nicholas B. and Muth R.,"A Method for Fragment Allocation Design in the Distributed Database Systems", the Sixth Annual U.A.E. University Research Conference, 2005.

[10] Krivokapi N., Kemper A. and Gudes E., "Deadlock detection in distributed database systems: a new algorithm and a comparative performance analysis", The VLDB Journal Volume 8, Issue 2, New York 1999.

[11] Ling Y., Chen S. and Chiang C.J., "On Optimal Deadlock Detection Scheduling", IEEE Transactions on Database, Vol. 55, Issue: 9, PP: 1178 – 1187, 2006.

[12] Maabreh K., “An Analyzing Study of the Distributed Database System Parameters”, On the proceedings of International Arab Conference on Information Technology (ACIT'2011). Riyadh, Saudi Arabia, 2011.

[13] Maabreh K. and Al-Hamami A., " Implementing New Approach for Enhancing Performance and Throughput in a Distributed Database " The International Arab Journal of Information Technology, vol. 10, no. 3, 2013.

[14] Matthias N. and Matthias J., "Performance Modeling of Distributed and Replicated Databases", IEEE transactions on knowledge dataengineering, v. 12 n4, p 645-672, July 2000.

[15] Mukkamala R., "Measuring the Effects of Data Distribution Models on Performance Evaluation of Distributed Database Systems", IEEE Transactions On Knowledge And Data Engineering, VOL. I. NO. 4, December 1989.

[16] Ozsu M. T. and Valduriez P., "Principles of distributed database systems", 3rd edition, Springer, New York, 2011.

[17] Raj Jain, "Database Systems Performance Evaluation Techniques", A project report written under the guidance of Prof. Raj Jain, 2008.

[18] Shiping Chen, Alex Ng and Paul Greenfield “A performance evaluation of distributed database architectures”, Concurrency and Computation: Practice and Experience, Volume 25, Issue 11, pp 1524–1546, 2013.

[19] Silberschatz A., Korth H.F. and Sudarshan S. "Database System Concepts", 6th edition, McGraw-Hill, New York, 2010.

[20] Weikum G. and Vossen G., "Transactional Information Systems, Theory, Algorithms and the Practice of Concurrency Control and recovery", Morgan Kaufman Publishers, 2002.

[21] Yadav A. K. and Agarwal A. "An Approach for Concurrency Control in Distributed Database System", International Journal of Computer Science and Communication, Vol. 1, No. 1, PP. 137-141, 2010.