Improving Database Storage Usability with the Cloud-based Architecture

Cuicui Su*, Yongzhi Wang*†, Yulong Shen*, Ke Cheng* and Jiawen Ma*
*School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China, 710071
†Key Laboratory of Grain Information Processing and Control (Henan University of Technology), Ministry of Education
Email: ccsu.hannah@gmail.com, yzwang@xidian.edu.cn

***Abstract*—Database is widely used for information storage and management. With the explosion of the data size, the requirement of the storage capacity is growing dramatically. Cloud offers clients a scalable solution to meet the demand of the increasing space. A cloud service, if used and managed properly, can increase the resource usability and provide more secure services.**

In this paper, we propose a cloud-based database architecture that increases the database storage usability meanwhile ensuring the data security. In this architecture, we move the database storage into a shared cloud-based server and leave the database engine at user's domain. The transmission of database physical files between the cloud and the database engine is achieved through Network File System. To avoid information leakage incurred by attacks on the cloud, the physical files stored in the cloud were encrypted by the database engine. To verify our idea, we used MySQL as our study case and evaluated the performance of this new architecture. A series of experiments indicate that the proposed architecture is promising in improving storage sharing, meanwhile guaranteeing the data security.

I. Introduction

Database is widely used for the data storage and processing [1], [2], [3], [4], [5]. As data grows, organizations need to consume more on traditional physical storage. In order to improve storage utilization, deal with the capacity limitation of traditional physical storage to database server and cut costs, we can move the physical files to a shared cloud-based server. By sharing storage space with many other users, it is possible for other users to access your data, sometimes because of criminal intent. Security of stored data and data in transit may be a concern when storing sensitive data at a cloud-based server. The risk of having data read during transmission can be mitigated through encryption technology. Encryption in transit protects data as it is being transmitted to and from the cloud service [6]. Encryption at rest protects data that is stored on the cloud.

When an organization selects to store data on the cloud, potentially sensitive data is at risk from insider attacks. Cloud users need to ensure that all critical data are masked or encrypted and that only authorized users have access to data in its entirety. To avoid the information leakage, we offer users an option to encrypt the data on the cloud by database engines. We propose a new architecture that introduces Network File System [7] to implement the transmission of physical files between the DB engine and the cloud. The research attempts to evaluate the performance of the new architecture by using benchmarks.

The architecture is presented in Fig. 1. The clients obtain information by sending queries to local database engine. The database engine has two major components: the storage engine and the query processor. The storage engine writes data to and retrieves data from stable media (e.g., disks). The query processor accepts, parses, and executes SQL commands. We replace the disks with the cloud where encrypted data files store in the architecture. The data transmission between database engine and the cloud is implemented by NFS server. Because the data on cloud is encrypted by the database engine and the encryption key is stored on the database engine side, people cannot retrieve information by attacking the cloud directly. Meanwhile, different users share the cloud storage service without interfering each other. This system is suitable for organizations that require large and secure storage and different access privileges for different departments, such as the military information management system [8] and the bank information management system, the model proposed in this paper not only offers the space to store physical data, but also protects the security of data.

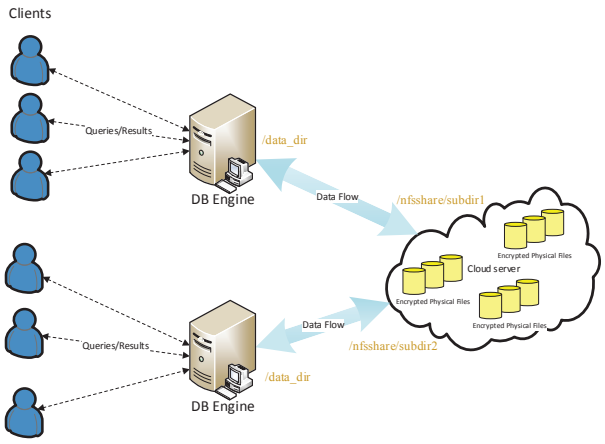

Fig. 1. The DB model with encrypted data on NFS.

We can create multiple subdirectories that will be mounted to *data_dirs* of different database engines respectively in the NFS shared directory on the cloud, which increases the storage usability. But for one specific database engine, there is only one subdirectory to mount it on. In this case, all the database

978-1-5386-1600-0/17 $31.00 © 2017 IEEE
DOI 10.1109/DSC.2017.102

engines share the cloud-based server as a storage platform.

We expect the access to remote data files on the cloud could be comparable to access to local physical files in speed. So, the feasibility of the new architecture will be verified by testing its performance. We took MySQL as our study case and proposed the proof-of-concept environment using a physical machine as the NFS server in this paper. The private cloud model is closer to the more traditional model of individual local access networks (LANs) used in the past by enterprise but with the added advantages of virtualization. So the proof-of-concept environment is reasonable for the architecture.

TPS, transactions per second, is an important indicator to MySQL performance evaluation. In this respect, a "Transaction" is a unit of execution that a client application invokes against a database [9]. And it was calculated as the total number of events divided by the total time of execution. This paper analyzed the performance of MySQL from the perspective of TPS obtained from the benchmark used and evaluated the decline in TPS due to remote data transmission. And the result of the research shows that the performance loss of the new architecture is moderate. So the new architecture in this paper may be a promising approach to increasing the storage usability to deal with massive data.

The contributions of the paper are twofold: (i) the introduction of the architecture that increases the database storage usability meanwhile ensuring the data security, and (ii) the performance evaluation of a study case of the new architecture.

The remaining of the paper is organized as follows. Related works are described in Section II. The insights into the architecture implementation and configuration of the system are given in Section III and IV. System evaluation are introduced in Section V. Finally, conclusion is drawn in Section VI.

II. Related Works

Intels Software Guard Extensions (SGX) [10] is a set of extensions to the Intel architecture that aims to provide integrity and confidentially guarantees to security-sensitive computation performed on a computer where all the privileged software is potentially malicious. Leveraging such a technology, Schuster et al. [11] proposed a solution to protect the execution integrity of MapReduce. We can apply SGX in DBMS to guarantee the system security. In order to protect the confidentiality and integrity of computation and data, we can upload them into Intel SGX the trusted hardware environment. While, because of the limited memory, we cannot upload all data into it. We can store the data files on the cloud, and install NFS service on the cloud server to implement the data transmission between the local database engine and the cloud. Then we can ensure the security of computation by using trusted hardware, and security of data by using MySQL InnoDB Tablespace Encryption. Users can connect with the MySQL engine and send queries to it. If security of the connections between MySQL engines and MySQL clients can be guaranteed, the whole system is secure. In order to estimate the availability of the system proposed above, this paper reports a study that probes into the influence that the introduction of NFS server exerts on MySQL performance.

The authors in [17] discussed the performance of MySQL and also highlighted the database elements that could be measured and adjusted during a benchmark. The authors in [18] analyzed MySQL performance on a limited resource server: Fedora and Ubuntu respectively. Two performance tests were conducted on the same hardware, but different distributions of Linux. The result of the paper presented that Ubuntu gave better performance than Fedora even though both were open source operating system. In [19], the authors evaluated performance of MySQL Cluster 7.0 in distributed data storage approach. The result of the paper illustrated that when the number of data storage nodes where data was stored increased, the number of successful transactions was improved gratefully and the average number of successful transactions per second was also. None of the papers above focused on the transmission of data files between the MySQL engine and remote physical files. Our research evaluated the performance of MySQL server with data files stored in remote NFS server, and analyzed the availability of the model proposed in this paper.

An orthogonal problem related to our work is to protect runtime integrity and control flow confidentiality of the cloud computing. Researchers have proposed a wide range of solutions from different perspectives [12], [13], [14], [15], [16]. As a complementary work, this paper addresses the data confidentiality problem.

III. Architecture Implementaion

As shown in Fig. 1 above. We introduced NFS server to implement the transmission of MySQL physical files between the database engine and the remote NFS server and MySQL InnoDB Tablespace Encryption to guarantee security of data files.

NFS is the abbreviation of Network File System, and it was originally developed by the Sun Microsystems in 1984. NFS was designed to achieve the sharing of file system resources in a network. By NFS, the native NFS client application can transparently read and write files located on the remote NFS server, just as if it were a local file. The introduction of NFS provides a way of making remote data files on the cloud available to local MySQL engine. In NFS system, NFS server stores the data on its disks, and application servers installed on the NFS client side request data through well-formed protocol messages. NFS structure is presented in Fig. 2.

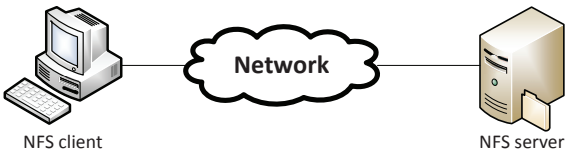

Fig. 2. The NFS Structure.

We can achieve a multi-level access control by defining the specific databases or tables that a specific user can access through the MySQL Access Privilege System [20]. MySQL Access Privilege System provides privileges that apply in different contexts and at different levels of operation. Database privileges apply to a database and to all objects within it. These privileges can be granted for specific databases, or globally so that they apply to all databases. *Innodb_file_per_table* is default in MySQL 5.6+ [21]. InnoDB storage engine supports data encryption for InnoDB tables stored in file-per-table tablespaces from MySQL 5.7. This feature provides at-rest encryption for physical tablespace data files. InnoDB Tablespace Encryption uses a two tier encryption key architecture, consisting of a master encryption key and tablespace keys. To enable encryption for a new InnoDB table or an existing InnoDB table, specify the ENCRYPTION option in a CREATE TABLE statement or an ALTER TABLE statement simply [22]. The third party cannot get the master encryption key stored in a *keyring* file specified by the *keyring_file_data* configuration option on the local MySQL engine side to decrypt the tablespace key. Therefore we can ensure the security of the informantion, even after the third party got the encrypted data files located on the remote NFS server.

Benchmarking refers to a set of standard test procedures against a database server to test and predict the performance of the system. A number of benchmarking tools are available to evaluate performance of a database system. The performance tests were all performed using Sysbench 1.0 [23] benchmarking software in this paper.

First, we installed MySQL 5.7.13 on the NFS client host. During MySQL's installing, program *mysql_install_db* initialized the MySQL data directory *"/var/lib/mysql"* by default, and created the system tables that it contains. Then we moved the entire directory *"/var/lib/mysql"* to the new created directory *"/nfsshare"* at the NFS server side. To setup NFS mounts, we installed NFS packages on both the NFS server and the NFS client machines. Then start the services on both machines. After installing packages and starting services on both the machines, we configured them for file sharing. To share a directory with NFS, we made an entry in *"/etc/exports"* configuration file at NFS server side. We created a new directory named *"nfsshare"* in *"/"* partition to share with the applications at the end of the client, then made an entry defining privilege in file *exports* for the directory to make the new created directory shareable in the network. For installation details, see [7].

Make sure there is a MySQL user and group on application server side and NFS server side with the same UID and GID. Make sure that the directory being exported and all the contents are owned by mysql:mysql. Make sure mysql:mysql has read and write access to the shared directory.

After configuring the NFS server, we mounted that shared directory or partition to the client server where MySQL server is. We mounted that shared directory to the local MySQL data directory on the client server. Then restart the MySQL service on the NFS client.

The hardware configuration in the paper is shown in Table I. We decided to use Ubuntu 14.04 LTS as our operating system for the system. In this research, we use MySQL 5.7.13 as our DBMS which is the latest version when I prepared this research last year. In this research, we had used 3 PC or laptop computers that were connected via a switch as illustrated in Fig. 3. The specification of the NFS server is a HP desktop with four 3.20GHz processing cores, 4GB-2133MHz of RAM, 47.1GB SATA of hard drive and on boarded 1000Mbps of network interface. The specification of the HostA is a ASUS labtop with four 2.30GHz processing cores, 2GB-1300MHz of RAM, 55.4GB SATA of hard drive and on boarded 1000Mbps of network interface. Whereas the HostB is a HP Desktop with eight 3.60GHz processing cores, 4GB-1600MHz of RAM, 50GB SATA of hard drive and on boarded 1000Mbps of network interface. All the computers were connected using a four-port 100Mbps switch with CAT5e wired cable connections.

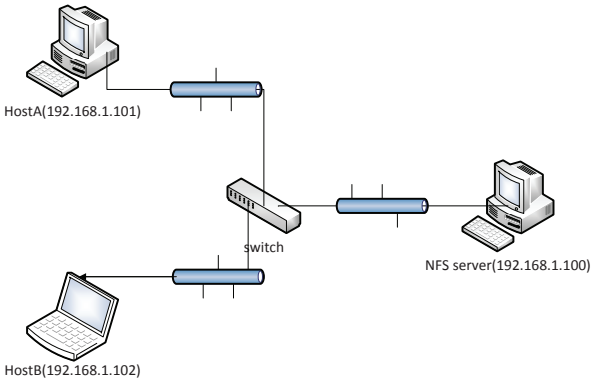

Fig. 3. Test case in this research.

Sysbench is a modular, cross-platform and multi-threaded benchmark tool for evaluating OS parameters that are important for a system running a database under intensive load. The value of TPS is directly related to the type of transactions being performed. In this experiment, change the number of threads from 4 to 128, incremented in multiples of 2. The performance variables that could be tweaked in Sysbench are varied, such as type of transaction (read-only or read-write) and size of each table to be processed (50000 rows or 10000 rows per table). The experiment was performed based on the variables mentioned previously. Each test will be executed on the same MySQL server host with different types of database transactions as well as different volume of data. For each case, each test was performed several times then the average would be the final result. Comparisons using statistical charts clearly identified differences of performances in term of TPS. Microsoft Excel was used to analyze and illustrate results statistically using graphs as demonstration tool. The MySQL and Sysbench detailed configuration are given in Table II and Table III.

On the MySQL client side, Sysbench tool started to send

TABLE I
HARDWARE SPECIFICATION

Server Specification(192.168.1.100)	
Model	HP Desktop
OS	Ubuntu 14.04 LTS
CPU	3.20GHz 4
Processor Model	Intel(R) Core(TM) i5-6500 CPU
Hard Disk	47.1GB SATA Hard Drive
Ram	4GB 2133MHz
Ethernet	100/1000M
HostA Specification(192.168.1.101)	
Model	ASUS Labtop
OS	Ubuntu 14.04 LTS
CPU	2.30GHz 4
Processor Model	Intel(R) Core(TM) i5-2410M CPU
Hard Disk	55.4GB SATA Hard Drive
Ram	2GB 1300MHz
Ethernet	100/1000M
HostB Specification(192.168.1.102)	
Model	HP Desktop
OS	Ubuntu 14.04 LTS
CPU	3.60GHz 8
Processor Model	Intel(R) Core(TM) i7-4790 CPU
Hard Disk	50GB SATA Hard Drive
Ram	4GB 1600MHz
Ethernet	100/1000M
Network Specification	
Switch	10/100M
Cable	CAT5e

TABLE II
MYSQL CONFIGURATION

Parameters	**Values**
Version	mysql Ver 14.14 Distrib 5.7.13
max_connextions	151
max_connext_errors	100
binlog_cache_size	32K
thread_cache_size	9
innod_thread_concurrency	4, 8, 16, 32, 64, 128

requests to the MySQL engine. The connection between the MySQL client program and the MySQL server can be done on the same host, or between different hosts. In this experiment, the two programs were on the same host, namely on the NFS client host.

IV. SYSTEM EVALUATION

As mentioned in the previous section, the experiment was performed using increasing number of threads in each mode with two different table sizes, namely 10000 and 50000 rows per table respectively, and the same number of tables, namely 10 tables. All the tests were performed under the identical setup. In the tables below, the negative value in the performance loss column represents for the improvement of performance, and positive values for performance loss.

TABLE III
SYSBENCH 1.1.0 PARAMETERS

Parameters	**Values**
Test Script	oltp_read_only.lua/oltp_read_write.lua
—tables	10
—table-size	10000/50000
—threads	4, 8, 16, 32, 64, 128
—time	300
—report-interval	20
—percentile	99

Table IV and table V depict TPS values for read-write mode where table size is 10,000. In the Sysbench read-write tests for the MySQL server with encrypted data files on HostA and the common MySQL below, TPS values for MySQL server with encrypted data on HostA was lower than that of common MySQL. For instance, TPS value of common MySQL server for 128 concurrent user connections (CUC) is 579.25 whereas TPS value of MySQL server with encrypted data on HostA is 372.96 for the same number of concurrent users, decreasing by 35.61%.

The differences between common MySQL model and the MySQL model with encrypted data on HostB are presented in table V. To our surprise, in some cases where CUC is relatively low, the TPS of MySQL model with data on HostB is higher than that of common Mysql. It means that when we have low concurrency of MySQL clients, the new model shows better performance than the common MySQL. But with increasing in number of concurrent user connections, the performance advantage of MySQL server with encrypted data on HostB over common MySQL is reducing, until when number of threads comes to 128, performance of MySQL server with encrypted data on HostB 15.56% lower than that of common MySQL server.

TABLE IV
TPS VALUES FOR HOSTA READ-WRITE, TABLE SIZE: 10,000

No. of Threads	**TPS common MySQL**	**TPS data on HostA**	**Performance loss**
4	39.68	35.19	11.32%
8	85.64	84.34	1.52%
16	167.55	148.84	11.17%
32	286.39	234.43	18.14%
64	391.27	317.75	18.79%
128	**579.25**	**372.96**	**35.61%**

When the table size increases to 50,000, the similar results are given as shown in table VI and table VII. With the increase in CUC, the downgrade of performance is increasing, that is to say the performance difference between the two is growing as shown in table VI. In case of HostB, on identical configuration with the test in table VI, the performance advantage of the model with encrypted data on HostB over common MySQL server has reduced, until the concurrent user connections come to 16, with TPS value of the MySQL model with encrypted

TABLE V
TPS VALUES FOR HOSTB READ-WRITE, TABLE SIZE: 10,000

No. of Threads	TPS common MySQL	TPS data on HostB	Performance loss
4	39.68	53.11	-33.85%
8	85.64	114.74	-33.98%
16	167.55	188.07	-12.25%
32	286.39	306.60	-7.06%
64	391.27	429.38	-9.74%
128	**579.25**	**489.12**	**15.56%**

data on HostB 1.32% lower than that of common MySQL server. Then with the increase in CUC, the downgrade of performance is increasing as shown in table VII.

TABLE VI
TPS VALUES FOR HOSTA READ-WRITE, TABLE SIZE: 50,000

No. of Threads	TPS common MySQL	TPS data on HostA	Performance loss
4	43.28	37.09	14.31%
8	85.62	72.67	15.12%
16	152.89	113.78	25.58%
32	258.24	158.97	38.44%
64	369.56	198.00	46.42%
128	**507.04**	**309.53**	**38.95%**

TABLE VII
TPS VALUES FOR HOSTB READ-WRITE, TABLE SIZE: 50,000

No. of Threads	TPS common MySQL	TPS data on HostB	Performance loss
4	43.28	44.82	-3.54%
8	85.62	101.85	-18.96%
16	**152.89**	**150.87**	**1.32%**
32	258.24	229.23	11.23%
64	369.56	300.88	18.58%
128	**507.04**	**389.67**	**23.15%**

Next, we focus on the tests in terms of read-only mode. The results of Sysbench Testing with read-only operations are presented in table VIII, IX, X, and XI. The MySQL model with encrypted data files on NFS server gets peak performance with 8 CUC as shown in tables below, while performance of common MySQL does not show regular changes with the increase in CUC. But the MySQL with encrypted data on NFS server shows little advantages over common MySQL server in general, and performance loss caused by the new model is on the rise.

From all the results given, the MySQL model with encrypted data on cloud-based server shows lower performance than common MySQL server but the downgrade is moderate in most cases, with MySQL engine on the same PC but data files in different places. In some cases where the CUC takes a lower value, there is the possibility of getting comparable performance, even improving the performance, requiring relatively high configuration of network and cloud-based server.

TABLE VIII
TPS VALUES FOR HOSTA READ-ONLY, TABLE SIZE: 10,000

No. of Threads	TPS common MySQL	TPS data on HostA	Performance loss
4	3465.16	3480.39	-0.44%
8	3592.26	**3592.89**	-0.02%
16	3500.26	3533.70	-0.96%
32	3555.52	3451.83	2.92%
64	3592.07	3343.20	6.93%
128	3519.89	2974.90	15.48%

TABLE IX
TPS VALUES FOR HOSTB READ-ONLY, TABLE SIZE: 10,000

No. of Threads	TPS common MySQL	TPS data on HostB	Performance loss
4	3465.16	3498.64	-0.97%
8	3592.26	**3597.78**	-0.15%
16	3500.26	3543.72	-1.24%
32	3555.52	3447.50	3.04%
64	3592.07	3358.13	6.51%
128	3519.89	2938.76	16.51%

When moving NFS server from physical PC to the cloud-based server, the similar conclusion would be drawn. So increasing the storage usability meanwhile ensuring the data security leads to moderate performance degradation. In conclusion, the new architecture is a promising way to expand the storage.

V. CONCLUSION

In this paper, we have proposed an architecture that improves database storage usability meanwhile guaranteeing the data security, using NFS to implement the data transmission between database engine and the remote physical files on the cloud-based server. And we do a series of tests to evaluate the availability of the new architecture in our research.

We can conclude from the study that the introduction of NFS server downgrades the performance of the system in high concurrency of MySQL clients, but the performance loss is moderate. So the MySQL model with encrypted data files on NFS server is a promising approach in terms of performance. And when the number of concurrent user connections is less than one certain value, the system proposed in this paper is available. That is to say, the new architecture in this paper is a promising approach to improving database storage usability meanwhile guaranteeing the data security.

ACKNOWLEDGMENT

This paper is supported in part by the Open Fund of the Chinese Key Laboratory of the Grain Information Processing and Control (No. KFJJ-2015-202), the Fundamental Research Funds for the Central Universities (No. XJJS15002, XJS16042, JBG160303, JB160312 and BDY131419), and the NSFC (No. U1536202, 61571352, 61373173, 61602364 and 61602365).

TABLE X
TPS VALUES FOR HOSTA READ-ONLY, TABLE SIZE: 50,000

No. of Threads	TPS common MySQL	TPS data on HostA	Performance loss
4	3527.36	3485.18	1.20%
8	3611.52	**3588.61**	0.63%
16	3554.69	3531.13	0.66%
32	3606.78	3458.56	4.11%
64	3572.62	3322.41	7.00%
128	3593.28	2930.31	18.45%

TABLE XI
TPS VALUES FOR HOSTB READ-ONLY, TABLE SIZE: 50,000

No. of Threads	TPS common MySQL	TPS data on HostB	Performance loss
4	3527.36	3448.61	2.23%
8	3611.52	**3563.81**	1.32%
16	3554.69	3517.56	1.04%
32	3606.78	3396.36	5.83%
64	3572.62	3321.86	7.02%
128	3593.28	2933.60	18.36%

REFERENCES

[1] Bajaj, S., and Sion, R. (2014). Trusteddb: A trusted hardware-based database with privacy and data confidentiality. IEEE Transactions on Knowledge and Data Engineering, 26(3), 752-765.

[2] Bater, J., Elliott, G., Eggen, C., Goel, S., Kho, A., and Rogers, J. (2017). SMCQL: secure querying for federated databases. Proceedings of the VLDB Endowment, 10(6), 673-684.

[3] Poddar, R., Boelter, T.,and Popa, R. A. (2016). Arx: A Strongly Encrypted Database System. IACR Cryptology ePrint Archive, 2016, 591.

[4] Wong, W. K., Kao, B., Cheung, D. W. L., Li, R., and Yiu, S. M. (2014, June). Secure query processing with data interoperability in a cloud database environment. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data (pp. 1395-1406). ACM.

[5] Arasu, A., Eguro, K., Joglekar, M., Kaushik, R., Kossmann, D., and Ramamurthy, R. (2015, April). Transaction processing on confidential data using cipherbase. In Data Engineering (ICDE), 2015 IEEE 31st International Conference on (pp. 435-446). IEEE.

[6] Nancy Messieh. "Publishers beware: Is CodexCloud the Grooveshark for ebooks?" October 2011.

[7] Christopher Smith , "Setting Up an NFS Server." http://nfs.sourceforge.net/nfs-howto/ar01s03.html.

[8] Dulłk, M., and Junior, M. D. (2016). Security in Military Cloud Computing Applications. Science and Military Journal, 11(1), 26.

[9] MySQL Performance Analysis and Tuning Tips on HPUX, Prepared By ViSolve Database Performance Team.

[10] Costan, Victor, and Srinivas Devadas. "Intel SGX Explained." IACR Cryptology ePrint Archive 2016 (2016): 86.

[11] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado,G. Mainar-Ruiz, and M. Russinovich, Vc3: Trustworthy data analytics in the cloud using sgx, in Proceedings of the 2015 IEEE Symposium on Security and Privacy, ser. SP 15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 38C54. [Online]. Available: http://dx.doi.org/10.1109/SP.2015.10

[12] Y. Wang; Y. Shen; H. Wang; J. Cao; X. Jiang, "MtMR: Ensuring MapReduce Computation Integrity with Merkle Tree-based Verifications," in IEEE Transactions on Big Data, doi: 10.1109/TBDATA.2016.2599928

[13] Yongzhi Wang, Jinpeng Wei, Shaolei Ren, Yulong Shen, Toward integrity assurance of outsourced computing - a game theoretic perspective, Elsevier Journal of Future Generation Computer Systems (FGCS), 2016, 55:87-100.

[14] Yongzhi Wang, Yulong Shen, RIA C An Audition-based Method to Protect the Runtime Integrity of MapReduce Applications, 23rd ACM Conference on Computer and Communications Security (ACM CCS 2016), Oct 24-28, 2016, Vienna, Austria. Poster Paper.

[15] Yongzhi Wang, Jinpeng Wei, and Mudhakar Srivatsa. "Result Integrity Check for MapReduce Computation on Hybrid Clouds". Proceedings of the 6th IEEE International Conference on Cloud Computing (IEEE CLOUD 2013), IEEE Computer Society, Santa Clara, CA, June 27-July 2, 2013, pages 847-854.

[16] Yongzhi Wang, Jinpeng Wei. Towards Protecting Control Flow Confidentiality on Cloud Based Computation. Elsevier Journal of Computers and Security (Computers and Security).Vol. 52, pp.106-127, July 2015. doi:10.1016/j.cose.2015.04.005.

[17] MySQL Performance Benchmarks, Measuring MySQLs Scalability and Throughput, A MySQLTechnical White Paper, May 2005.

[18] Ahmed, M., Uddin, M. M., Azad, M. S., and Haseeb, S. "MySQL performance analysis on a limited resource server: Fedora vs. Ubuntu Linux." Proceedings of the 2010 Spring Simulation Multiconference. Society for Computer Simulation International, 2010.

[19] Pukdesree, Sorapak, V. Lacharoj, and P. Sirisang. "Performance evaluation of distributed database on PC cluster computers." Wseas Transactions on Computers 10.1(2010):21-30.

[20] Security in MySQL, Oracle and/or its affiliates.

[21] MySQL Internals Manual, Oracle and/or its affiliates.

[22] MySQL 5.7 Reference Manual Including MySQL NDB Cluster 7.5, Oracle and/or its affiliates.

[23] Kopytov, Alexey. "SysBench manual." MySQL AB (2012).