Distributed Relational Database Performance in Cloud Computing: an Investigative Study

Full Paper

Alan Litchfield
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Alan.litchfield@aut.ac.nz

Awadh Althwab
Service and Cloud Computing Research Lab
Auckland University of Technology
Awadh.althwab@aut.ac.nz

Chandan Sharma
Service and Cloud Computing Research Lab
Auckland University of Technology
chandan.sharma@aut.ac.nz

Abstract

Weak points in major relational database systems in a Cloud Computing environment, in which the nodes were geographically distant, are identified. The study questions whether running databases in the Cloud provides operational disadvantages. Findings indicate that performance measures of RDBMS' in a Cloud Computing environment are inconsistent and that a contributing factor to poor performance is the public or shared infrastructure on the Internet. Also, that RDBMS' in a Cloud Computing environment become network-bound in addition to being I/O bound. The study concludes that Cloud Computing creates an environment that negatively impacts RDBMS performance for RDBMS' that were designed for n-tier architecture.

Keywords

Public Cloud, Distributed RDBMS performance, network latency, database, performance loss

Introduction

This paper presents the results of a study focused specifically on what significant performance issues are encountered and highlights specific areas where performance is negatively affected. The starting point for the study is with the premise that any distributed Relational Database Management System (RDBMS) requires a network and nodes (typically the n-tier architecture), that applications of RDBMS' are normal in an organizational context, and that the performance of RDBMS has been a concern for researchers for many years. The n-tier architecture operates on a client/server model and includes a database system and at least one server application or middleware, through which users gain access to the database system (Frerking, Blanton, Osburn, Topham, DelRossi, and Reisdorph 2004; Eriksson 2015). The Cloud Computing (CC) architecture differs from n-tier architecture, where CC is built on the public Internet and typically abstracts the physical architecture through virtualization (Ivanov 2013; Khajeh-Hosseini, Greenwood, and Sommerville 2010). It is understood that there are impacts on the performance of an RDBMS when it is run in a CC instance but just what those impacts are have not been well expressed.

In CC, the user may obtain direct access to data or via a Service-Oriented Architecture (SOA), for example through an Application Programming Interface (API) made available by a service provider. In n-tier architecture, programmatic controls exist within a data center and amongst its networks and servers as nodes and hooks but unlike the public network infrastructure, these have significant bandwidth (Benson, Akella, and Maltz 2010). This situation, of a shared and limited bandwidth, constrains the performance of applications that run in CC (Moens and De Turck 2015). Thus, for example, query optimization in a distributed RDBMS n-tier architecture is likely to suffer from performance issues (Chaudhuri 2012; Liu and Yu 1993; Mullins 1996; "Query optimization strategies in distributed databases" 2013). Therefore, whereas RDBMS' normally operate on n-tier architecture, we investigate RDBMS performance operating on a cloud-based architecture to determine where break points exist. What we have found is that all the systems up for testing failed at least one of the tests but that they failed them in different ways. To do this, we ran a series of experiments or tests using common query structures on databases in a cloud network. The experiments compare the performance of factors that allowed us to see where drops in performance indicated where potential program problem exist. We then investigated network and server logs to find exactly what the databases were doing at that point when performance dropped.

Related Work

CC is an abstraction that encompasses an assortment of technologies and practices. Attempts to define CC typically focus on some feature set, classes of technology, protocols, patterns of use, and so on. Both Geelan (2009) and Buyya, Yeo, and Venugopal (2008) include economies of scale in definitions of CC, for example reduction of the overall cost of cloud infrastructure consumption by service consumers compared with the equivalent resource requirements off-cloud. The authors focus on Service Level Agreement (SLA) provision to define the level of quality of service expected from either party. Additionally, scalability and the ability to optimize resource use empower users to have full control over their spending on IT services (Vaquero, Rodero-Merino, Caceres, and Lindner 2008). These resources are typically consumed on a pay-per-use basis and service providers are responsible for guaranteeing infrastructure to an agreed SLA.

One important feature of CC is a high level of availability for services, data and other IT resources (Litchfield and Althouse, 2014). However, merely moving a company's computing resources to a cloud platform without sufficient feasibility assessment for accessibility and performance may lead to bottlenecks. Technical bottlenecks or network insufficiency can in turn lead to data unavailability, especially when data move between cloud nodes over networks with limited bandwidth. Database locking, lack of storage capacity (such as in Thakar, Szalay, Church, and Terzis 2011) and cache flushing can also cause bottlenecks in Cloud-based systems.

This study is focused on the effect of Public Clouds (PuCs) on the performance of RDBMS' in CC. Li, Yang, Kandula, and Zhang (2010) conduct a comparison between PuCs and conclude that considerable differences exist between PuC providers and this makes it difficult to choose which provider to use. Further, Iosup et al. (2011) suggest that PuCs appear to suit small databases but that they demonstrate deficiencies when heavy workloads (such as scientific analytics) are involved, but Thakar et al. (2011) and Hashem et al. (2015) disagree, stating that PuCs such as Amazon Elastic Cloud Computing (EC2) and Microsoft SQL Azure are suited to scientific tasks. Gunarathne, Wu, Qiu, and Fox (2010) add that PuCs can be used for big data tasks performed on high dimensional data residing on heterogeneous databases, where complex queries consume intensive computational resources. But I/O performance inconsistencies occur on PuCs due to the existence of shared infrastructure (the Internet) and the potential for improperly tuned VMs on hypervisors.

The RDBMS is a mainstay of enterprise information systems management but in a study considering the uptake of NOSQL systems, McKendrick (2012) states that 77% of the study's sample consider structured data as central to their daily business activities, and that 92% use RDBMS' compared to 11% that have deployed NOSQL databases. The likelihood of the RDBMS being fully replaced by other types of database is not significant, however performance issues still need to be resolved.

In a distributed environment, query optimization presents persistent challenges (Chaudhuri 2012). The distributed environment adds significant complexity for query optimization where data are moved between locations to enable intermediate operations that may be intended to optimize the local processing of a query (Chaudhuri 1998). The consequence is the addition of more variables when choosing optimization plans (Mullins 1996; Khan and Khan 2013). Liu and Yu (1993) recommend that more investigation is needed in order to determine whether or not the inefficient implementation or unsuitable execution plans chosen by the RDBMS cause long processing queries. Their findings suggest that network overhead is a major influence. However, the issue remains unsolved and there is a need to revisit the traditional optimization methods. The introduction of CC has shifted the focus to alternative database

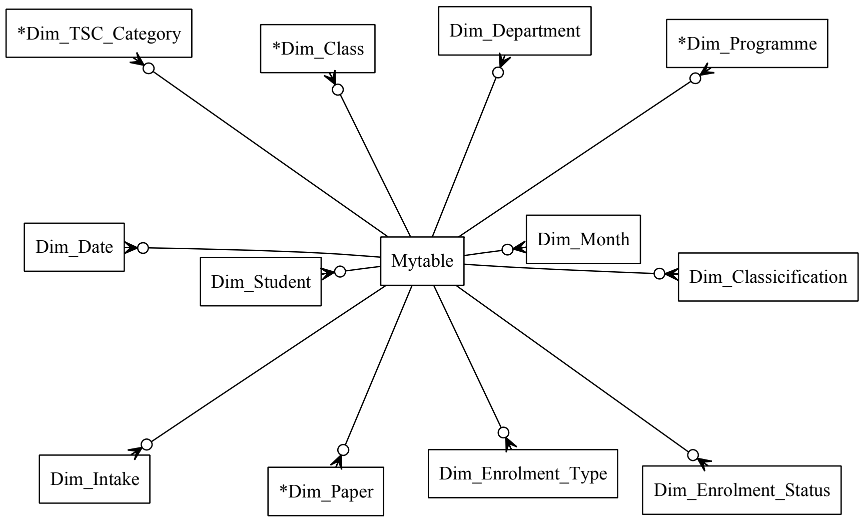

Figure 1: Experiment star schema

architectures, for example new approaches for query optimization in NOSQL databases appear to provide improved performance with large dataset processing (Zhang, Yu, Zhang, Wang, and Li 2012; Chaudhuri 2012).

Conflicting views about the use of RDBMS in the era of large datasets exist and indicate that it is architectural issues with the relational data model that prevents it from being effective in combating large datasets in CC, and it is these issues that erode RDBMS benefit (Litchfield and Althouse, 2014). Durham, Rosen, and Harrison (2014) suggest that large datasets pose challenges and the data model can be a significant limiting factor. They claim that RDBMS do not perform efficiently with big data and pulling data across a network impacts performance. In support of this, Minhas, Yadav, Aboulnaga, and Salem (2008) examine the performance of a RDBMS and conclude that running a database over a virtualized environment creates I/O disk overhead but that such an overhead does not have significant impact on the runtime of queries. Thakar et al. (2011) agree that CC affects RDBMS I/O performance. However, Bose, Mishra, Sethuraman, and Taheri (2009) conclude that CC is capable of handling database-intensive workloads. That although RDBMS' perform better off-cloud, they achieve between 85% and 92% of native performance in CC. Similarly, Ivanov, Petrov, and Buchmann (2012) conclude that CC is more suitable for read-mostly work but for other purposes such as OLTP and data-intensive workloads, CC poses challenges.

The use of benchmarking tools to measure network latency, CPU time, I/O, and memory hierarchy is useful to gather performance data for performance evaluation of PuC (for example, Iosup et al. 2011; Jackson et al. 2010; Lloyd et al. 2013). Other measures include query runtime and the number of tuples returned (Kohler and Specht 2014; Thakar et al. 2011; Baccelli and Coffman, 1982), transaction throughput (Anderson, Breitbart, Korth, and Wool 1998; Born, 1996), transaction response time (Born, 1996), and wait time (Gray, Helland, O'Neil, and Shasha 1996; Bouras and Spirakis 1996). In this study, these measures have been employed to assess the problem.

Method

The study attempts to replicate a real world situation. A dataset from a university data warehouse is used, because it is large, relational, and contains anonymized student records. The database consists of thirteen tables as a star schema (Figure 1). The tables are received as Comma Separated Value (csv) files. The data are converted into insert scripts and then split using a Unix split command so that groups of 10000 records could be inserted per file. Given the large number of files produced, the insertion process is automated by scripts. It was found that the database systems would run short of available memory if too many records were inserted at once. So while `MYTABLE` initially contains 400 million tuples, with a raw data size of 80GB, the dataset was reduced to 100 million tuples, a size of 18GB.

The study involves two Cloud Distributed Database (CDD) systems, SQL Server 2012 and Oracle 11g, running in VMs and connected through the Internet. The systems are set up in Amsterdam (Netherlands) and Auckland (New Zealand), as geographically distant locations and to be as far from each other as possible. Each system has two VMs (four in total). For comparable results all four VMs have identical configurations, however there is a difference in CPU speed (Table 1). The two VMs in Amsterdam contain the fact table and all experiments are run from Amsterdam. The other two VMs in Auckland contain the dimension tables. The experiments include nine queries (Table 2) that any RDBMS should be able to execute. The queries range from simple to complex, from simple select statements to inner and outer joins, sort, and aggregation functions.

The experiments were designed to capture data during their execution:

Duration: The time taken by a query to complete and is calculated as the sum of processing time and communication cost. This measure is an indicator that reflects if an experiment suffers from issues, and as general rule, the shorter it is, the fewer issues can be involved. However, a longer duration indicates otgerwise and further investigation is required to determine the cause(s) (Baccelli and Coffman 1982; Born 1996; Kohler and Specht 2014).

CPU Time: Time reported by RDBMS that indicates CPU consumption for the duration of the experiment's execution. This is an especially important measure since it plays a central role once data arrive from disk and determines what RDBMS choose as join operators to join datasets. High CPU time means long duration (Lloyd et al. 2013; Anderson et al. 1998; Iosup et al. 2011).

Disk Operation: The number of physical reads and writes that occur when experiments are undertaken. Since a RDBMS is known to be I/O bound, this measure is employed to observe such effect (Anderson et al. 1998).

Average I/O Latency: The measure tells the average time each I/O operation takes to finish. Since the study is conducted on a PuC environment, it is expected to reflect whether there is an effect of disk operation on performance (Lloyd et al. 2013; Anderson et al. 1998; Thanos, Bertino, and Carlesi 1988; Iosup et al. 2011).

Logical Read: Represents number of reads that take place in memory. This measure may be partly associated with CPU time so that when there is high CPU consumption, it is ac- companied by a large number of logical reads, although it is not always necessarily the case. Note, this measure is used when experiments revel special cases (Anderson et al. 1998).

Network Traffic: This measure means the amount of data that travels network for each experiment (Born 1996; Jackson et al. 2010).

Wait Events: This measure shows the events the systems wait for some operations to finish. As an example can be when the system waits for the cloud network to complete I/O operation (Born, 1996; Bouras and Spirakis 1996; Gray et al. 1996; Anderson et al. 1998; Thanos et al. 1988).

Data collection is conducted in two ways, suited to each RDBMS. The data are then analyzed using comparisons and statistical methods. The comparisons are between the RDBMS' on each node to: 1. Explain and compare both Amsterdam and Auckland execution plans. 2. Compare runtime between both systems. 3. Compare CPU time and explain its relevance to the chosen execution plans. 4. Number of logical reads used for comparison, if execution plans create a large number of logical reads. 5. Number of physical operations used for comparison and their effects on runtime, quantified by measuring the average I/O latency. 6. Wait events to explain the wait time that occurs during experiment runtime.

A correlation test is applied to study the effect of network traffic on runtime. In cases such as CPU time and run time correlation, the test is not suitable because CPU time is part of the runtime, so there is always a causal relationship. The normal data distribution is checked and a skewness test (Martin and Bridgmon, 2012) is performed on duration, CPU time, physical reads, and network traffic. Table 3 shows the results for the skewness test where Z is skewness in standard units.

In order to statistically analyse a dataset the Z values must be from -1.96 to +1.96 (Cramer and Howitt 2004). As shown in Table 3, the standard values in the study for skewness are between 3 and 6. This means that data are positively skewed, therefore the logarithm of data is undertaken to remove skewness so that pragmatic tests such as regression analysis can be carried out.

Server Location	Virtualization	VMs configurations
Amsterdam (Local)	Xenon, Quad Core x 2.13Ghz	Microsoft Windows 7 x64 Operating System, 4 CPU cores, 8 GB RAM, 200 GB of disk space
Auckland (Remote)	Xenon, Quad Core x 2.26Ghz	Microsoft Windows 7 x64 Operating System, 4 CPU cores, 8 GB RAM, 200 GB of disk space

Table 1: Experiment environment configurations

EXP	TJ	No.T	No.CS	No.CP	QC
1	Right Outer Join	2	3	2	Low
2	Inner Join	4	9	6	High
3	Inner Join	2	4	3	Medium
4	Inner Join	2	2	3	Low
5	Left Join	2	3	3	Low
6	Inner Join	2	4	5	High
7	Inner Join	3	5	6	High
8	Full Outer Join	2	2	4	High
9	Inner Join	2	2	1	Medium

KEY:

TJ = Type of Join	No.T = Number of Tables involved in the query
No.CS = Number of Columns involved in Selection	No.CP = Number of Columns involved in Projection
QC = Query Complexity	

Table 2: Summary of Experiments

Measure	Skewness Z value
Duration	(1.226/.378) = 3.24
CPU time	(2.097/.378) = 5.54
Physical reads	(1.793/.378) = 4.74
Network traffic	(1.882/.564) = 3.33

Table 3: Skewness test

The data sets are generated for two RDBMS', hence variables such as CPU time for the two databases are independent of each other. An independent sample test is suitable for such cases as it checks if there is any difference between the sample means. The samples are visualized using scatter plots, however due to space limitations in this paper, these are not all included.

Results

Due to space limitations, all data are not provided in the paper. Notable points in the data indicate that in EXP1, Oracle's NESTED LOOPS caused delays and affected CPU time, but the use of the SORT operator by the Auckland SQL Server VM eliminated a similar situation and provided a performance gain. Disk operations on the Auckland instance contributed to additional delays, for example SQL Server needed nearly six minutes to run EXP2 versus just over four minutes for Oracle. In EXP3 there is a significant variation in the runtime and CPU time for both systems. SQL Server took less than two minutes to finish EXP3, compared with Oracle's 4.8 hours. Oracle consumes more CPU time in the Amsterdam VM than the Auckland VM.

In EXP4, both systems execute in a nearly identical manner, but SQL Server runs for 6 hours whereas Oracle runs for 7.2 hours. Looking at Oracle in Amsterdam, there is a significant consumption of CPU time because it uses the NESTED LOOPS join operator. One row from DIM_STUDENT is selected, and then the operator looks for the matching row among 100 million tuples. However, In EXP5 Oracle consumes less CPU time than SQL Server. The latter's choice of MERGE JOIN operator creates the need for a SORT operator. This pattern also appears in EXP4, which indicates that while the MERGE JOIN operator is the best choice from the optimizer's point of view, it needs a SORT operator, which consumes more CPU time than a HASH JOIN operator.

The Amsterdam VMs appear to suffer from high I/O, which reflects the reality of being in a PuC environment where cloud infrastructure is shared among many users. SQL Server performed more physical reads in the Auckland VM in EXP3 (45ms) than it did in EXP2 (27ms). A similar pattern is observed in Oracle but this leads to a different result; the average I/O latency in EXP3 for the Auckland VM (14ms) is higher than EXP2 (8ms). Therefore, these latencies indicate that disk activity may be instrumental in poor performance of RDBMS' in a cloud environment.

Thus, the PuC appears to be a factor in the poor performance of RDBMS' in a CDD. Sorting large data volumes, in particular, is an expensive task, but SQL Server chose the MERGE JOIN operator and this requires the data to be sorted. This outcome occurred in six out of seven experiments. The experiments conducted on Oracle indicate that the use of the HASH JOIN operator requires less time than the MERGE JOIN operator, suggesting that HASH JOIN is more time efficient than MERGE JOIN. This was most noticeable in EXP7 where Oracle appears slower than SQL Server, needing 20 hours to run the experiment compared 10 hours for SQL Server, despite that SQL Server consumed more CPU time than Oracle in both VMs.

EXP8 aims to fully join two large datasets over the network and then perform an ORDER BY operation. Oracle did not complete due to a network issue and because of this, the execution plans are lost, especially in most of the Amsterdam instance data. However, SQL Server runs for the longest time so far and consumes the highest CPU time. The results show that there is a significant increase in resource consumption. For SQL Server, logical reads in the Amsterdam instance reached nearly 500000 reads and this is reflected in CPU time. Despite the Auckland instances performing more logical reads than Amsterdam, they consume less CPU time. The Amsterdam Oracle instance has zero logical reads because no data was carried forward.

In a second attempt of Oracle's EXP8, the data were reduced from 18GB with 100 million tuples to 10 million tuples. The second attempt finishes in 1 hour and 15 minutes. Based on this, the same query with 100 million tuples may have taken take 140 hours to run, or more than five days. Analysis of the execution plan shows that there is extensive disk activity to sort data, which influences the runtime. The Auckland instance experiences a high average I/O latency per read, whereas the Amsterdam instance (which reads only the index Table) needs 16ms per I/O read. Each disk write takes an average of 39ms and for the read, it takes an average of 9ms to finish.

In EXP9, SQL Server stopped after running for 24 hours and consequently data were lost. Oracle in EXP9 executed in the Auckland instance, instead of Amsterdam. An analysis of the execution plans demonstrate that the maintenance of data integrity can be an issue when a large dataset is involved, which suggests that an RDBMS becomes less effective in such a situation. However, Oracle behaves differently and its approach to EXP9 causes fewer complexities than with SQL Server. The Amsterdam SQL Server appears to perform a considerable number of logical reads and consumed 166 seconds of CPU time. While it is

Network traffic Log		
Duration	Pearson Correlation	.928
	Sig.(2-tailed)	.000
	N	16

= Correlation is significant at the level 0.01 level (2-tailed)

Table 4: Correlation between Duration and Network traffic

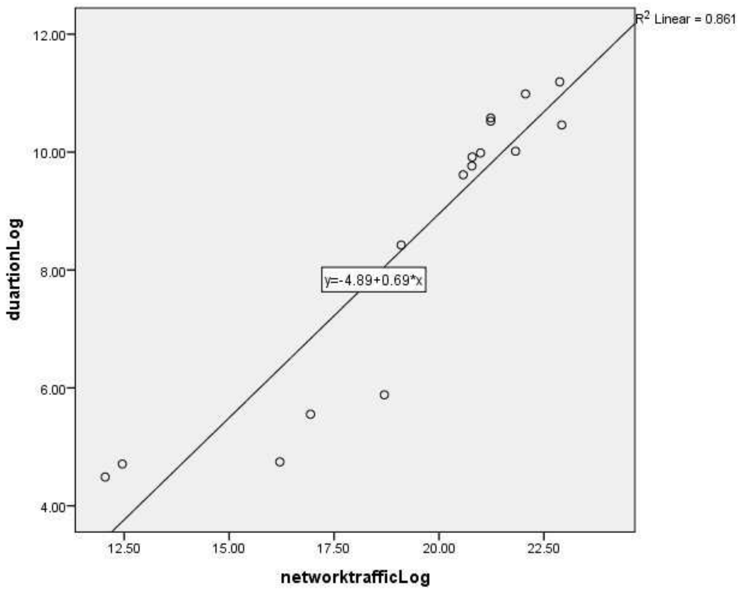

Figure 2: Duration vs Network traffic

normal practice for physical write to occur when an update happens, it is difficult to explain why SQL Server's Amsterdam instance has to perform them. Further, physical writes that occur in the Oracle Auckland instance are a result of the update statement. The same applies to SQL Server when it performed 630075 physical writes before cancellation.

As further evidence to indicate that the PuC environment is implicated in performance loss, SQL Server in EXP3 waits 12.43% of the runtime for I/O operations to complete. Similarly, Oracle in EXP2 and EXP9 waits 27.46% and 45% of runtime respectively for I/O reads to finish. Furthermore, the Amsterdam SQL Server instances experience a high average I/O latency, although they do not process as large a dataset as the Auckland VM. In EXP3, Oracle requires the data to be brought over the network before it processes the query. That creates performance issues for RDBMS' because of the lack of network capacity. Table 4 shows a significant correlation between duration and network traffic. This relationship is demonstrated in Figure 2 where the scatter plot shows a linear trend between duration and network traffic, that duration time increases as the network traffic increases. The plot also shows outliers in the lower left below the diagonal that represents duration time where the network traffic begins to exceed 100MB.

Table 5 indicates that the slope parameter is significantly different from zero at the level of 0.01. Thus, this is an important relationship between network traffic and duration. At 95% confidence, with every unit increase in network traffic, there is an increase in the duration between 0.533 and 0.851. Additionally, there is an interesting pattern in Figure 3 that displays a near sigmoid relationship. This suggests a growth pattern in the relationship between network traffic and duration that is affected by a complex of other unknown factors.

Conclusion

In this study, the intended outcome was to discover the break points when operating a RDBMS in a PuC. The experiments using common queries found in any RDBMS environment compare the performance of

Coefficients a

Model		Ustd Coeff.		Std Coeff.	t	Sig.	Conf. Interval	
		B	Std. Error	Beta			LB	UB
1	C	-4.887	1.462		-3.344	0.005	-8.002	-1.753
	NTL	0.692	0.074	0.928	9.325	0.000	0.533	0.851

a = Dependent Variable: durationLog

Key:

Ustd Coeff. = Unstandardized Coefficients	Std Coeff. = Standardized Coefficients
Conf. Interval = 95% Confidence Interval for B	LB = Lower Bound
NTL = network traffic log	UB = Upper Bound
C = Constant	

Table 5: Simple regression test

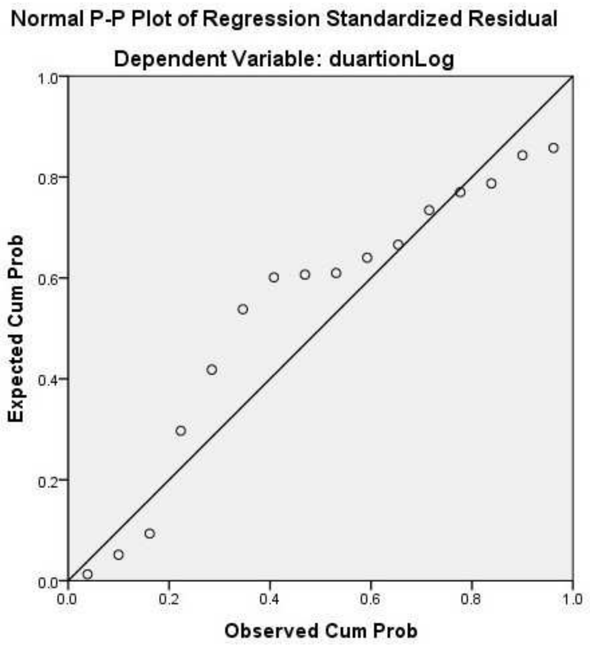

Figure 3: Normality of simple regression test

factors that allowed us to see where drops in performance indicated where potential program problem exist. We then investigated network and server logs to find exactly what the databases were doing at that point when performance dropped. It was found that the n-tier software architecture does not operate effectively in a PuC. While simple operations are not overly affected, more complex queries are shown to produce excessive delays. What is of note is that each of the CDDs applied in the experiments illustrate break points at different places. In particular, Oracle in EXP8 crashes at the 44364th second from the execution start. This is because Oracle downloads tables for executing each query and timed out in EXP8. More than 90% of both VMs time is spent waiting for data to move via the network. For completing EXP8 for Oracle (OSA) the data size was reduced from 100 million tuples to 10 million tuples so that smaller amount of data went through the network, minimizing the effect of the network. Similarly, SQL Server executes EXP9 in a manner that causes a long-running query and with the highest CPU time of all nine experiments. The steps in EXP9 that SQL Server undertakes indicates that including a sub-query in update statements in SQL Server causes significant performance issues. More importantly, SQL Server performs physical writes where it does not seem to need to, especially when the query does not intend to update the fact table and a cascade update is not required. Certainly, sub-queries in update statements are not always an issue if they proceed in a manner that fits CDD, as in Oracle's approach. However, SQL Server's approach to EXP9 does not fit CDD and this is evident when the sub-query is removed.

The study faces multiple limiting factors. The research originally aimed to use a larger dataset for the experiments, however most of the experiments took a long time to run and often they did not finish due to network failures. Therefore, experiments could not be easily compared. Also, the experiments were not conducted on n-tier architecture as a control. Since the study is conducted without any performance enhancements, the opportunity to redo the stated experiments including performance enhancements, both on and off the cloud, exists. More importantly, once the study is redone, then results would be useful in determining the architectural issues with the relational data model. Additional comparisons with NOSQL DB's can be considered. In the cloud environment, it may be argued that in graph databases, query processing is more efficient when compared to RDBMS'. This is because a join in a RDBMS is equivalent to an edge traversal between nodes in a graph database. Joins in graph databases represent a graph traversal. A graph database is optimized for graph traversals because elements (vertices and edges) maintain direct reference to their adjacent elements, and this makes traversing a graph structure within a graph database fast and efficient (Rodriguez and Neubauer, 2010).

References

Anderson, T., Breitbart, Y., Korth, H., and Wool, A. (1998). "Replication, consistency, and practicality: are these mutually exclusive?" ACM SIGMOD Record, (27:2), pp 484–495.

Baccelli, F., and Coffman, E. G. (1982). "A data base replication analysis using an m/m/m queue with service interruptions." In ACM Sigmetrics Performance Evaluation Review. (11:1). pp. 102–107.

Benson, T., Akella, A., and Maltz, D. (2010). "Network traffic characteristics of data centers in the wild." In Proceedings of the 10th ACM SIGCOMM conference on internet measurement. pp. 267–280.

Born, E. (1996). "Analytical performance modelling of lock management in distributed systems." Distributed Systems Engineering, (3:1), p 68.

Bose, S., Mishra, P., Sethuraman, P., and Taheri, R. (2009). "Benchmarking database performance in a virtual environment." In Performance evaluation and benchmarking. Berlin Heidelberg: Springer. pp. 167–182.

Bouras, C., and Spirakis, P. (1996). "Performance modeling of distributed timestamp ordering: Perfect and imperfect clocks." Performance evaluation, (25:2). pp 105–130.

Buyya, R., Yeo, C., and Venugopal, S. (2008). "Market-oriented cloud computing: Vision, hype, and reality for delivering its services as computing utilities." In 10th IEEE international conference on high performance computing and communications. pp. 5–13).

Chaudhuri, S. (1998). "An overview of query optimization in relational systems." In Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems. pp. 34–43).

Chaudhuri, S. (2012). "What next?: a half-dozen data management research goals for big data and the cloud." In Proceedings of the 31st symposium on principles of database systems. pp. 1–4.

Cramer, D., and Howitt, D. (2004). "The sage dictionary of statistics: a practical resource for students in the social sciences." Thousand Oaks: Sage.

Durham, E., Rosen, A., and Harrison, R. (2014). "Optimization of relational database usage involving big data a model architecture for big data applications." In 2014 IEEE symposium on computational intelligence and data mining. pp. 454–462

Eriksson, P. (2015). "A new approach for enterprise application architecture for financial information systems: An investigation of the architectural implications of adopting serialization and rpc frameworks, nosql/hybrid data stores and heterogeneous computing in financial information systems." (Masters thesis, KTH Royal Institute of Technology).

Frerking, G., Blanton, P., Osburn, L., Topham, J., DelRossi, R., and Reisdorph, K. (2004). "U.S. patent application 10/935,514" [Patent]. Washington, DC: U.S. Patent and Trademark Office.

Geelan, J. (2009). "Twenty-one experts define cloud computing." Cloud Computing Journal, (1:4), pp 1–5.

Gray, J., Helland, P., O'Neil, P., and Shasha, D. (1996). "The dangers of replication and a solution." ACM SIGMOD Record, (25:2), pp 173–182.

Gunarathne, T., Wu, T., Qiu, J., and Fox, G. (2010). "Mapreduce in the clouds for science." In 2010 IEEE second international conference on cloud computing technology and science (pp. 565–572).

Hashem, I., Yaqoob, I., Anuar, N., Mokhtar, S., Gani, A., and Khan, S. (2015). "The rise of 'big data' on cloud computing: review and open research issues." Information Systems (47:1), pp 98–115.

Iosup, A., Ostermann, S., Yigitbasi, M., Prodan, R., Fahringer, T., and Epema, D. (2011). "Performance analysis of cloud computing services for many-tasks scientific computing." IEEE Transactions on Parallel and Distributed Systems, (22:6), pp 931–945.

Ivanov, I. (2013). "The impact of emerging computing models on organizational socio-technical systems." In Software and data technologies. pp. 3–19. Berlin Heidelberg: Springer.

Ivanov, T., Petrov, I., and Buchmann, A. (2012). "A survey on database performance in virtualized cloud environments." International Journal of Data Warehousing and Mining, 8(3), pp 1–26.

Jackson, K., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., and Wright, N. (2010). "Performance analysis of high performance computing applications on the Amazon Web Services cloud." In IEEE Second international conference on cloud computing technology and science. pp. 159–168.

Khajeh-Hosseini, A., Greenwood, D., and Sommerville, I. (2010). "Cloud migration: A case study of migrating an enterprise IT system to IaaS." In IEEE 3rd international conference on cloud computing. pp. 450–457.

Khan, M., and Khan, M. (2013). "Exploring query optimization techniques in relational databases." International Journal of Database Theory and Application, (6:3).

Kohler, J., and Specht, T. (2014). "Vertical query-join benchmark in a cloud database environment." In Second world conference on complex systems. pp. 581–586.

Li, A., Yang, X., Kandula, S., and Zhang, M. (2010). "CloudCMP: comparing public cloud providers." In Proceedings of the 10th ACM SIGCOMM conference on internet measurement. pp. 1–14

Litchfield, A., and Althouse, J. (2014). "A systematic review of cloud computing, big data and databases on the cloud." In Twentieth Americas conference on information systems. AIS.

Liu, C., and Yu, C. (1993). "Performance issues in distributed query processing." IEEE Transactions on Parallel and Distributed Systems. (4:8), pp 889–905.

Lloyd, W., Pallickara, S., David, O., Lyon, J., Arabi, M., and Rojas, K. (2013). "Performance implications of multi-tier application deployments on Infrastructure-as-a-Service clouds: towards performance modeling." Future Generation Computer Systems, (29:5).

Martin, W. E., and Bridgmon, K. D. (2012). "Quantitative and statistical research methods: From hypothesis to results" (42:1). John Wiley and Sons.

McKendrick, J. (2012). "Big data, big challenges, big opportunities: 2012 IOUG big data strategies survey" (Technical Report). Murray Hill, New Providence, NJ: Unisphere Research.

Minhas, U., Yadav, J., Aboulnaga, A., and Salem, K. (2008). "Database systems on virtual machines: How much do you lose?" In IEEE 24th international conference on data engineering workshop, 2008. (pp. 35–41).

Moens, H., and De Turck, F. (2015). "Shared resource network-aware impact determination algorithms for service workflow deployment with partial cloud offloading." Journal of Network and Computer Applications, (49:1), pp 99–111.

Mullins, C. S. (1996). "Distributed query optimization." Technical Support. Retrieved from http://craigsmullins.com/dist_qry.pdf

"Query optimization strategies in distributed databases." (2013). International Journal of Advances in Engineering Sciences. (3:3), pp 23–29.

Rodriguez, M. A., and Neubauer, P. (2010). "Constructions from dots and lines." Bulletin of the American Society for Information Science and Technology. (36:6), pp 35–41.

Thakar, A., Szalay, A., Church, K., and Terzis, A. (2011). "Large science databases, are cloud services ready for them?" Scientific Programming. (19:2-3), pp 147–159.

Thanos, C., Bertino, E., and Carlesi, C. (1988). "The effects of two-phase locking on the performance of a distributed database management system." Performance Evaluation. (8:2), pp 129–157.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). "A break in the clouds: towards a cloud definition." ACM SIGCOMM Computer Communication Review. (39:1), pp 50–55.

Zhang, Y., Yu, L., Zhang, X., Wang, S., and Li, H. (2012). "Optimizing queries with expensive video predicates in cloud environment." Concurrency and Computation: Practice and Experience. (24:17), pp 2102–2119.