2020

# Factors That Influence Throughput on Cloud-Hosted MySQL Server

Eric Brown
*Walden University*

# Walden University

College of Management and Technology

This is to certify that the doctoral study by

Eric Brown

has been found to be complete and satisfactory in all respects,
and that any and all revisions required by
the review committee have been made.

Review Committee
Dr. Steven Case, Committee Chairperson, Information Technology Faculty
Dr. Charlie Shao, Committee Member, Information Technology Faculty
Dr. Jodine Burchell, University Reviewer, Information Technology Faculty

Chief Academic Officer and Provost
Sue Subocz, Ph.D.

Walden University
2020

Abstract

Factors That Influence Throughput on Cloud-Hosted MySQL Server

by

Eric Brown

MS, Capella University, 2006

BS, Western Illinois University, 1996

Doctoral Study Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Information Technology

Walden University

December 2020

## Abstract

Many businesses are moving their infrastructure to the cloud and may not fully understand the factors that can increase costs. With so many factors available to improve throughput in a database, it can be difficult for a database administrator to know which factors can provide the best efficiency to maintain lower costs. Grounded in Six Sigma theoretical framework, the purpose of this quantitative, quasi-experimental study was to evaluate the relationship between the time of day, the number of concurrent users, InnoDB buffer pool size, InnoDB Input/Output capacity, and MySQL transaction throughput to a MySQL database running on a cloud, virtual, database server. Data were collected from Debian Linux virtual machines (VMs) on Amazon Web Services, Google Cloud Platform, and Microsoft Azure using HammerDB database benchmarking software. The results of the one-way ANOVA were not significant. A key recommendation is to study further other factors and a more in-depth investigation into each cloud provider's performance. The implications for positive social change include the potential for database administrators to make informed decisions on how to configure MySQL to run in a VM and choose the best cloud provider so that nonprofits may serve their clients more efficiently.

# Factors That Influence Throughput on Cloud-Hosted MySQL Server

by

Eric Brown

MS, Capella University, 2006

BS, Western Illinois University, 1996

Doctoral Study Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Information Technology

Walden University

December 2020

# Dedication

I dedicate this work to my wife, Carla, for helping protect my time in the completion of this program and motivating me to continue during the difficult times. I could not have made it through this program without her support.

# Acknowledgments

I would like to acknowledge Dr. Christos Makregorgis, my original chair, who passed away during the editing process. The first two sections of this doctoral study are a reflection of his mentoring work.

I would also like to acknowledge Dr. Steven Case, who took over as the chair of my committee and assisted and helped to motivate me to finish this study.

Table of Contents

List of Tables

List of Figures

# Section 1: Foundation of the Study

Enterprise software often requires scalable speed and computing capacity, coupled with the business realities of curtailing costs (Garg, Singla, & Jangra, 2016). In meeting these needs, many organizations have been migrating their technology infrastructure to a private cloud or a commercial cloud provider, with an estimated $241 billion in cloud investments by the year 2020 (Gholami, Daneshgar, Low, & Beydoun, 2016). The cloud offers high availability, scalable speed, and high performance, allowing many businesses to realize that the cloud meets many of the same goals set forth for databases themselves (Sakr, 2014). An increasing number of clients can affect the performance of a database in the cloud (Januzaj, Ajdari, & Selimi, 2015), which can detract from the desired speed and performance. In this study, I compared the performance of MySQL running on virtual servers at different times of the day with varying compositions of load on several public cloud providers.

## Background of the Problem

A recent survey of technical professionals by RightScale (2019) found that 91% of respondents were utilizing public cloud providers. The cloud allows businesses to take advantage of a nearly unlimited pool of computing resources (Gholami et al., 2016). However, with larger workloads and processing demand variations comes unpredictable usage patterns (Gharbaoui, Martini, Adami, Giordano, & Castoldi, 2016). For this reason, it can be challenging to manage the quality of service (QoS) in the cloud. The topic of QoS assessment and management has been a matter of increasing interest in

business and academic circles (Abdelmaboud, Jawawi, Ghani, Elsafi, & Kitchenham, 2015).

While working as a database administrator (DBA) for a large organization, I often reviewed the usage of different systems: the database, servers, and the network. In nearly all cases, the usage graph patterns for all systems were similar. As the workday started, the usage graphs would show more and more activity, peaking in the midmorning hours. As midday approached, the graph would dip during the lunch break, and slowly rise again during the midafternoon hours. The activity would gradually decline as the end of the workday approached and would return to near zero by the early evening hours.

Amazon Web Services (AWS) is the most popular public cloud provider (RightScale, 2019), more popular means more users, which in turn could mean a smaller share of resources, particularly at times of day when business activity is at its peak. While the database workload may be outside of the control of the DBA, other factors, such as the InnoDB buffer pool and the InnoDB input/output (I/O) capacity, are under the control of the DBA. With this knowledge, I used a statistical experiment technique to model the database performance of the MySQL cloud database on several public cloud providers at different times of day, subject to a series of controllable experimental factors.

## Problem Statement

Today's businesses are moving on-premises database technologies to cloud, virtual, database servers to reap the benefits of scalability and flexibility to rapidly changing business needs, such as big data, increased throughput, and accessibility

(Nedelcu, Ionescu, Ionescu, & Vasile, 2014). Despite this movement, many cloud databases are not configured optimally by administrators in a way that could potentially reduce processor utilization by 30% per VM (Reddy & Shyamala, 2016). The general information technology (IT) problem was that a DBA may not understand the controllable factors available that can affect the performance of a cloud-based database server. The specific IT problem was that some DBAs lack information on the relationship between the time of day, the number of concurrent users, InnoDB buffer pool size, and the InnoDB I/O capacity to increase transaction throughput to a MySQL database running on a cloud, virtual, database server.

## Purpose Statement

The purpose of this quantitative, quasi-experimental study was to evaluate the relationship between the time of day, the number of concurrent users, InnoDB buffer pool size, InnoDB I/O capacity, and MySQL transaction throughput to a MySQL database running on a cloud, virtual, database server. The four independent variables considered in this experimental study were the time of day, the number of concurrent users, InnoDB buffer pool size, and the InnoDB I/O capacity. I tested each factor at two levels, referred to as high (+) and low (-). The dependent or response variable was the throughput of the MySQL as measured by the number of transactions per second (see Transaction Processing Performance Council [TPC], 2010). The public cloud computing platforms that constituted the population for this study were Amazon Web Services, Google Cloud, and Microsoft Azure, the top three public cloud providers based on market share (see Sikeridis, Papapanagiotou, Rimal, & Devetsikiotis, 2017). The location was the data

centers located in the United States, so the distance to the data centers was not a factor. This study contributes to social change because the identification of the combination of controllable factors that maximize throughput renders reduced costs for nonprofit organizations using the cloud, allowing such organizations to serve their clients more quickly and efficiently.

## Nature of the Study

I selected a quantitative methodology for this study. Quantitative methods utilize numerical data to test a hypothesis to answer the research question through careful measurement and evaluation of variables (Park & Park, 2016). A thorough analysis of the literature has informed me of the variables and the values to use to test the hypotheses to determine which factor combination provided the most efficient throughput in MySQL hosted on a cloud-based VM. Qualitative methods are exploratory by nature and gather expert opinions and experiences as data utilizing unstructured or semistructured techniques, the results of which are analyzed to surface themes to gain a deep understanding of the underlying nature of a phenomenon (Houghton & Casey, 2013). Expert opinions or experiences do not assist in quantifying the impact of the combination of measurable factor levels on a quantifiable response and, therefore, were not suitable to achieve my research objective. Mixed methods are employed to attempt to gain a fuller understanding of complex issues using a combination of quantitative and qualitative methods, which may include surveys, case studies, and interviews (Sommer & Subramanian, 2013). Because this study did not involve humans in the experiment, there was no need for surveys, case studies, or interviews or any other qualitative approaches,

so mixed methods were not appropriate to use. I developed the research question to focus on identifying the target levels for various factors that affect MySQL performance in a hosted cloud environment. This narrow research question required a classic, quasi-experimental design because it was essential to control both the factor levels and measure their impact; therefore, a quantitative method was necessary, and any qualitative method would not have addressed the research question.

I selected a full-factorial, quasi-experimental design for this study. A researcher applies a full-factorial design by changing each factor from a low to a high level until all combinations of levels are achieved (Reddy & Shyamala, 2016). Although there may be many real-world factors and their combinations that can influence a response outcome, in practice, a researcher only focuses on those factors and their combinations that have a high potential to influence the response and can be quantifiably measured (Sanchz & Wan, 2015). Due to the expense of running and recording continuous experiments with databases on public, cloud-hosted VMs, I decided to reduce the number of trials to $2^k$ using only a high and low level for each of the $k$ variables or factors. When using descriptive designs, researchers first collect data and then attempt to draw the meaning or conclusion of the general population from the data (Fisher & Marshall, 2009). Descriptive designs do not affect a treatment on the population but are used to attempt to measure dispersion or determine a point of central measure (Fisher & Marshall, 2009). Since the purpose of this study was to find an optimal combination of factors that I implemented, the definition of a descriptive study failed to meet the goals of this study. Researchers use correlational studies to determine how variables are correlated and where

the average differences in one variable relate to the average differences in another variable (Gabbiadini & Greitemeyer, 2017). In essence, correlational studies are conducted to determine the strength and direction of the relationship between two continuous variables (Chen & Popovich, 2002). However, the purpose of this study was not to determine how incremental changes in one variable relate to changes in another variable or the strength of the relationship between two variables. Quasi-experimental research attempts to measure the effects of treatment with the intervention of a researcher (Bärnighausen, Røttingen, Rockers, Shemilt, & Tugwell, 2017). The quasi-experimental design was appropriate for this study because I directly implemented a series of treatments on the database and did not randomly assign treatments between a control group and a treatment group. The real difference between experimental and quasi-experimental is the random assignment of subjects to treatment, and therefore, this study was not genuinely experimental due to the lack of randomness (see Abramson et al., 2018). DBAs have many specific factors that they can alter to improve database performance, and seeking the optimal combination of these particular factors would not qualify this approach as a quasi-experimental study. Ruling out descriptive, correlational, and experimental designs for this study left a quasi-experimental model as the appropriate option.

## Quantitative Research Question and Hypotheses

The overarching research question for this quasi-experimental study was:

What are the optimal levels of the time of day, number of users, InnoDB buffer pool size, and InnoDB I/O capacity that maximizes the throughput of MySQL running on a cloud server as measured by transactions per second (TPS)?

Having $k = 4$ factors (i.e., $F_1$, $F_2$, $F_3$, and $F_4$, representing the time of day, the number of concurrent users, InnoDB buffer pool size, and the InnoDB I/O capacity, respectively) and two possible experimental levels (i.e., a high and low value for each factor) led to a full-factorial design with $2^4$ or 16 possible experiments and responses or outcomes for each replication of the experiment. With replication in consideration, I replicated each experiment three times and used the mean value of each response for 48 trials (see National Institute of Standards and Technology/Semiconductor Manufacturing Technology [NIST/SEMATECH], 2012a). The pairs of main and interaction effect null and alternative hypotheses are as follows:

- Main Effect Hypotheses:
  - $H_{0a}$: The main effect $F_i$ of factor $i$ is not significant on the outcome.
  - $H_{1a}$: The main effect $F_i$ of factor $i$ is significant in the outcome.
- Two-Factor Interaction Effects Hypotheses:
  - $H_{0b}$: The interaction effect of $F_iF_j$ of the pair of factors $i$ and $j$ are not significant on the outcome.
  - $H_{1b}$: The interaction effect of $F_iF_j$ of the pair of factors $i$ and $j$ is significant on the outcome.
- Three-Factor Interaction Effects Hypotheses:

- $H_{0b}$: The interaction effect of $F_iF_jF_k$ of the triplet of factors *i*, *j*, and *k* is not significant on the outcome.
- $H_{1b}$: The interaction effect of $F_iF_jF_k$ of the triplet of factors *i*, *j*, and *k* is significant on the outcome.

The data for the four experimental factors and response variables were all assessed experimentally, and there was no survey role in this study. In other words, I did not solicit survey respondent input on the five (i.e., 4+1) variables and instead performed experiments from scratch.

## Theoretical Framework

Six Sigma is a statistical methodology used to reduce variation in a process to improve desired outcomes by making data-driven decisions (Maleyeff & Kaminsky, 2002). William Smith developed Six Sigma at Motorola in the late 1980s through realizing that variation in the individual components being assembled led to defective products and setting out to reduce the variations in components to reduce problems with the final product (LeMahieu, Nordstrum, & Cudney, 2017). The Six Sigma methodology involves five steps to follow in solving a problem: define, measure, analyze, improve, and control (DMAIC; LeMahieu et al., 2017).

Six Sigma consists of a set of statistical process control techniques to assess whether a production process or an output is out of control or not (Antony, Snee, & Hoerl, 2017). When a process is out of control, researchers use other methods to determine the variables contributing to the out-of-control process (Antony, Snee, & Hoerl, 2017).

In this study, I aimed to find the optimal levels for $k = 4$ specific two-level factors that affect database throughput measured as the number of TPS. The adopted methodology used to tackle this problem is a statistical method called a $2^k$ full factorial experiment. Because Six Sigma utilizes statistical methods to assess out-of-control output processes and the variables that need to be adjusted to help a process get back in control, I considered the Six Sigma as an appropriate framework to frame this research.

In this case of the current study, the first step in the Six Sigma process, to define the problem, was to improve the throughput of the database. The second step of the Six Sigma process is to measure, in which the researcher determines the inputs that have the most significant effect on database throughput (LeMahieu et al., 2017). In the literature, the three major areas that have been identified as bottlenecks for database systems in the cloud are buffer pool management, disk I/O operations (Ding, Shan, & Jiang, 2016), and processing capabilities (Bonthu, Thammiraju, & Murthy, 2014). For this experiment, these three areas had different levels tested to determine the optimal combination of InnoDB buffer pool size and InnoDB I/O capacity for the first two constraints identified. In the cloud, a user cannot see the other processes running on the same server that may or may not be affecting the processing capability of a virtual server hosted in the cloud. I tested the processing capacity at two different times of day, with the idea that more processing power may be available outside of regular business hours. The remainder of this experiment continued with the final three steps of the Six Sigma process, by analyzing the data to determine the best combination of factors to improve and control the throughout at optimal levels.

## Definition of Terms

*Infrastructure-as-a-service*: A service model that is essentially a virtualized server hosted on a publically available cloud service. This service model provides a virtualized server, including networking and storage services (Abourezq & Idrissi, 2016).

*InnoDB buffer pool size*: A parameter within the MySQL configuration that can set a block-level cache for caching indexes and data in memory (Tapdiya & Xue, 2014).

*InnoDB I/O capacity*: A parameter within the MySQL configuration that controls the maximum number of input and output operations per second (Kong, 2012).

*Platform-as-a-service*: A service model providing middleware services, such as database management systems, upon which developers can build application components (Kaltenecker, Hess, & Huesig, 2015).

*Service-as-a-Service*: A service model offering direct access to software uncoupled from the underlying technology. Users pay for this service model on a subscription basis, and the provider is responsible for software updates as well as backups of user data (Kaltenecker et al., 2015).

*Throughput*: In reference to databases, the measurement of the average operations per unit of time (Hwang et al., 2016).

## Assumptions, Limitations, and Delimitations

### Assumptions

The definition of the word assumption is a fundamental statement given or not given about facts in the study and how the facts may be related (Foss & Hallberg, 2017). Researchers typically make assumptions about the following: (a) the underlying theory,

(b) the phenomenon being investigated, (c) the instrument being used to measure the variables, (d) the methodology being used to address the problem being investigated, (e) the statistical analysis, (f) the statistical procedures used, (g) the population under study, and (h) the ability of the results to be generalized beyond the sample being studied (Dusick, 2015).

Concerning (a) the underlying theory, I assumed that there is a single underlying theoretical framework that fits the study (see Dusick, 2015). As discussed in the Theoretical Framework section, Six Sigma is the theoretical framework that was most suitable for this study because I used the same statistical methods to control the output processes of a database. I also assumed the parameters chosen for this study (i.e., number of users, InnoDB I/O capacity, and InnoDB buffer pool sizing) are the factors most likely to affect the throughput. Another factor, the available processing power, is affected by running the experiment at different times of day, with the idea that other users of the cloud system may be affecting the available processing power available for this experiment (see Bonthu et al., 2014).

When testing the phenomenon under investigation (b), I assumed that the phenomenon under scrutiny was measurable and defined (see Dusick, 2015). In examining the performance of databases, many researchers make use of the database benchmarking standards from the TPC, which measures the throughput of a database when the benchmarking software applies a simulated workload to the database (Ding et al., 2016; Sakr, 2014; Tapdiya & Xue, 2014). Because of a large number of examples using the TPC benchmarking standard, I assumed this to be a valid tool for measuring

database performance. For this test, I chose the TPC-C benchmarking, which simulates an online transaction processing.

A researcher must also make assumptions about the instrument (c) being used to measure the phenomenon (Dusick, 2015). The software performing the TPC-C benchmarking test was HammerDB, which supports the TPC-C benchmarking specifications. HammerDB (2018a) allows for creating the number of virtual users and measures the throughput as the transactions are applied.

The researcher also makes assumptions about the methodology (d) chosen and that it is the appropriate methodology for the given study (Dusick, 2015). As outlined in the Nature of the Study section, the purpose of this study was not to find out the how or why to a particular question but to determine the optimal levels of specific factors that provide the best throughput on a cloud-based VM hosting MySQL. Based on the facts presented in that section, I assumed that quantitative analysis was the proper approach for the study.

All statistical procedures have requirements for use, so researchers must make assumptions about the analysis (e; Dusick, 2015). For this study, I assumed that the independent and dependent measures chosen meet the criteria for a full-factorial analysis. Based on the facts presented in the Nature of the Study section, I assumed that a full-factorial analysis was the most appropriate choice for this study.

Along with assumptions about the analysis, a researcher must also make assumptions about the ability to detect any meaning in the relationships under observation (f; Dusick, 2015). I made an assumption that the values used for the factors

in the full-factorial analysis would show variations in throughput; however, this assumption was determined after the experiments were concluded and the results were calculated.

The relevance of the population (g) is another feature of a study that a researcher must address (Dusick, 2015). Since different public cloud providers use different virtualization platforms (Bernstein, 2014), I assumed that the underlying hypervisor does not play a role in experiments performed. I also assumed that the cloud providers gave the VMs used in these experiments equal treatment with any other VMs that may be working on the underlying hardware as well as identically performing hardware (see Xavier, Matteussi, Lorenzo, & De Rose, 2016). Comparable hardware or virtualized hardware can have an impact on database performance because storage, caching, and processing is so critical to database operations (Bonthu et al., 2014). With the use of several different public cloud providers, I assumed that I would find valid results that would not be skewed by underlying hypervisors, hardware, or unknown loads on the systems. The information provided in the Nature of the Study section justifies the assumption for the full-factorial analysis.

Finally, a researcher needs to assume that the results (h) are generalizable beyond the participants of the study (Dusick, 2015). Given that the statistical testing model outlined by Six Sigma is accurate and that a full-factorial quantitative analysis was a proper approach for this study, I assumed that the values determined in this experiment should be able to guide other MySQL DBAs in getting a superior throughput for their systems.

**Limitations**

Limitations are the realistic descriptions of the weaknesses in the research presented, provide a useful understanding of the context of the study, and outside of a researcher's ability to control (Brutus, Aguinis, & Wassmer, 2013). A researcher should consider the limitations of the study in the areas of (a) the theory under investigation, (b) the phenomenon being investigated, (c) the instruments used in the experiment, (d) the methodology, (e) the analysis, (f) the ability to detect any significance, (g) the participants, and (h) the results of the study (CITE).

The theoretical foundation of Six Sigma (a) limited this study as well as the modeling capabilities of Six Sigma in accurately modeling the events under observation. Six Sigma is used when seeking to improve results by looking at the factors that can affect results and systematically adjusting the inputs to achieve an optimal outcome (Psomas, 2016). In this study, I only looked at the factors affecting throughput and no other factors or outcomes.

Regarding (b) the phenomenon under investigation, the fundamental limitation was that the phenomenon of how the levels of certain measurable factors affect the throughput as measured by TPS is somewhat complicated by the dimension of time. In other words, a given software or application, such as a database and the cloud environment that it operates within, are never really static and can change when the application or environment is improved or updated. Therefore, the phenomenon itself is well defined, but the factors that influence throughput are not static.

The key limitation of the instrument (c) used for this study was that the software application, HammerDB, in the implementation of the TPC-C benchmarking standard measured the outcome of this experiment measurement. The TPC-C standard simulates transactions of users in an order-entry type of software. Other TPC (n.d.) benchmarks, like the TPC-DI, simulate the extraction, transformation, and loading of data into a database or the TPC-H, which simulates a database supporting a decision support system. While there remains a variety of benchmarking standards available, this study was limited to the TPC-C benchmarking standard.

I applied the quantitative (d) methodology in this study, so it was limited to the data collection and analysis associated with the quantitative approach. I did not conduct an analysis of opinions or perceptions by individuals on any aspects of cloud computing, databases, or VMs. The factors being manipulated and observed in this study were limited to quantitative analysis only. Another limitation of this study related to the data collection and analysis was in having enough samples to detect statistical power. As shown on p. 59, 50 samples will give power of 0.907, but in this case, it was advised that 48 samples will be sufficient to meet a power of 0.90 (C. Marigeorgis, personal communication, June 7, 2017).

The ability of the (e) statistical analysis to detect a significant difference in throughput using the factor levels chosen also limited this study. As previously mentioned in the hypotheses, the factors in this study were limited to two levels of the number of simulated users, the size of the InnoDB buffer pool, the size of the InnoDB I/O capacity, and the time of day.

The results of this experiment were limited to the (f) ability of the analysis to detect statistical differences or relationships that exist based on the factors and factor levels chosen. While there should be some improvement with increasing or decreasing at least one factor, the experiment may not show any statistical significance. For example, changing the number of virtual users from 10 to 100 may reduce throughput; however, with a sufficiently provisioned VM, the increased usage may be adequately handled with the existing virtual hardware.

As I outlined in the Purpose Statement section, the (g) population of this study was limited to the top three public cloud providers as defined by market share: AWS, Microsoft Azure, and Google Cloud (see Sikeridis et al., 2017). Other public cloud providers may have been suitable for this study, but due to the effort in setting up this experiment on each platform, I limited the trials to the three major platforms.

The (h) results of this study were limited to how generalizable they may be to other cloud platforms (see Dusick, 2015). Cloud providers use different methods of virtualization (Babcock, 2016), and each may be configured differently or guarantee various levels of service. The generalizations of the results in this study were limited to cloud providers that match the population under investigation.

**Delimitations**

Delimitations can facilitate the decision-making process as well as help eliminate biases (Argilaga, 2003). Delimitations are the factors of the experiment that the researcher can include and control (Dusick, 2015) and may span (a) the theoretical foundation, (b) the phenomenon being researched, (c) the instruments measuring the

phenomenon, (d) the methodology, (e) the analysis, (f) the ability to find significance in the experiment, (g) the population, and (h) the results.

The (a) theoretical foundation for this study was Six Sigma. While researching the topic of database efficiency, the literature enumerates other foundational theories, such as queueing theory (Srivastava, 2018). For this study, the theoretical lens of Six Sigma best represented the effect of multifactor manipulation on optimal throughput.

As mentioned in the Limitations section, researchers can use other methods to measure the efficiency of a database. Through the support of the literature and the tools used, throughput, as measured by TPS, was the (b) phenomenon chosen as the outcome for optimization in this study.

Costs and common usage helped in the selection of the instruments (c) for this study. The VMs in this study used the latest production version of Debian Linux, and the database used was the most recent production version of MySQL. The tool used to implement a simulated load on the database and measure throughput was the newest production version of HammerDB. On each cloud platform, I deployed equally provisioned VMs, and costs and availability of the configuration options determined the virtual hardware used.

The (d) methodology chosen for this study was quantitative. As discussed in the Nature of the Study section, quantitative experimentation was most appropriate to achieve the quantitative objectives of this experiment.

The (e) analysis for this study was a full-factorial analysis, as outlined in the Nature of the Study section.

I determined the (f) ability to detect statistical significance once the data had been collected and analyzed.

As outlined in the Limitations section, the (g) population I chose for this study, AWS, are the three public cloud platforms with the highest market share (see Sikeridis et al., 2017). I chose data centers located in North America to reduce the possibility of network lag affecting the outcome of the study,

The (h) results of this study should be generalizable to the three public cloud providers given that all other factors are equal, such as the operating system version, the resources available to the VM, and the database management system deployed.

## Significance of the Study

### Contribution to Information Technology Practice

As I mentioned in the Theoretical Framework section, there have been many studies conducted on ways to cut costs in cloud computing by improving performance in cloud databases. However, my broad literature search did not reveal a published experimental study that compares the performance of a single database on identical cloud servers at different times of the day. I suspected that, with all factors and cloud servers being equal, most cloud providers would perform about the same with some decrease or increase during the late morning or early afternoon hours, which is what I observed while working in an enterprise environment. The performance differences at different times of day could be due to the workload of a particular cloud provider or could affected by the backend virtualization architecture in use. It may be that Amazon's implementation of Xen is better suited for database queries than Microsoft's Hyper-V implementation

(Babcock, 2016) when all other factors are equal. Ideally, this study might help those working in IT make an informed decision when configuring and using MySQL on a cloud server.

## Implications for Social Change

In my years working in IT, I have donated significant amounts of my time applying my skills to a few select nonprofit organizations. When I have had the resources, I have also donated hardware. Initial equipment costs may be prohibitive for nonprofits, but in today's world, organizations still need computing resources to fulfill their mission. Cloud computing can offer cash-strapped organizations a way to meet their computing needs at a lower cost by removing the requirements for significant up-front investments (Mann, 2015). I hope this study will help organizations that help others make better use of their resources and give nonprofits more opportunity to help others.

# A Review of the Professional and Academic Literature

## Introduction

The literature review begins with a discussion of the foundational theory of Six Sigma, its history, and how other theories have built up to and worked around Six Sigma. Examples of the design of experiments (DOEs), which I used in the experiment for this study and researchers commonly use in conjunction with Six Sigma. I also discuss other theories frequently found in the literature that may pertain to this study and a discussion of why alternate theories were not selected. A review of the literature on the independent variables chosen for this study and the instrument used to measure the dependent variable is then provided.

The searching process for the literature review took many iterations. In general terms, I started using the search terms *database performance* and *database efficiency* in Google Scholar and the databases accessible through the Walden University Library. As I learned more about different approaches to database efficiency, various factors that affect database efficiencies, such as InnoDB and disk I/O, were added to the search terms. Six Sigma and related theories on efficiency were also consistent search terms. Many studies in the literature included both physical and VMs, so the search terms based on virtual performance were also included.

Seventy-six academic articles comprise this literature review, with 72.4% of the articles published in the last 5 years, and 86.8% of the articles deriving from peer-reviewed journals. In the literature review, I cover the theoretical framework for this study, Six Sigma, as well as comparative frameworks and the statistical methods commonly used for Six Sigma. The second portion of this section includes a discussion of what the literature has to report on the independent and dependent variables for this study and the instruments commonly used to collect the data for the experiments used in this study. The overreaching goal of this literature review was to identify key factors in the performance of database systems, namely MySQL, running on VMs in a cloud environment and how to measure that performance. Once the tool had also been identified for measuring database performance, the literature showed how Six Sigma and DOEs use these elements to answer the primary research question for this study: What are the optimal levels of the time of day, number of users, InnoDB buffer pool size, and

InnoDB I/O capacity that maximizes the throughput of MySQL running on a cloud server as measured by TPS?

## Foundational Theory and Design of Experiments

The foundational theory for this experiment, Six Sigma, does not exist in a vacuum and has evolved and overlaps with other theories in quality improvement or management. In this section, I summarize the literature that explains the underlying foundational theory of Six Sigma and the underpinning experimental approach that I applied in finding the optimal combination of factors to improve the performance of the MySQL database running on a cloud-based VM. I also discuss other theories that preceded Six Sigma and other theories used in conjunction with Six Sigma.

**Six Sigma.** Six Sigma is an improvement doctrine that focuses on continuous improvement by measuring existing processes and statistically measuring the impact of changing critical factors in that system to reach a defined goal (Hsieh, Lin, & Manduca, 2007). Other definitions in the literature mention the goal of improving performance by reducing variation on a system (LeMahieu et al., 2017). This second definition comes closer to matching the name of Six Sigma since the Sigma here represents the statistical notation for standard deviation and the goal of Six Sigma in keeping variation very small in meeting and exceeding customer expectations (Hsieh et al., 2007).

Six Sigma is a framework for improvement based upon the philosophy of total quality management (TQM; Antony et al., 2017; Maleyeff & Kaminsky, 2002). In TQM, both management and employees participate in improving the quality of the products or services to create long-term customer satisfaction (Sreedharan, Sunder, & Raju, 2018).

Nguyen and Nagase (2019) varied the definition slightly, describing TQM as an organization-wide and top-down philosophy that strives for customer satisfaction and continuous customer satisfaction. Where TQM is a management philosophy about making progress for the business as a whole, Six Sigma focuses more on removing defects by concentrating on specific and measurable outputs of a process (Sreedharan et al., 2018). It was out of the TQM frame of thinking that led William Smith of Motorola to develop Six Sigma around 1987 (Antony et al., 2017). The literature varies on the official start date of Six Sigma at Motorola. Many researchers agree that Jack Welch of General Electric pushed the concept of Six Sigma from an academic manufacturing concept into the limelight by announcing Six Sigma as the main business strategy for the corporation (Antony et al., 2017; Hsieh et al., 2007; LeMahieu et al., 2017; Reosekar & Pohekar, 2014). To prove excellence and knowledge in Six Sigma, practitioners in the area can earn certifications of green-belt, black-belt, and master black-belts (Antony, Gupta, Sunder, & Gijo, 2018), similar to the martial arts.

Six Sigma focuses on measurable data in a scientific, objective, and repeatable way with the end goal of improving the financial performance of an organization (Hsieh et al., 2007). The academic literature shows that the manufacturing field most often uses Six Sigma; however, Six Sigma also has uses in healthcare (Reosekar & Pohekar, 2014), education (LeMahieu et al., 2017), and IT (Hsieh et al., 2007). The implementation of Six Sigma works in different industries because nearly any professional domain can make use of its five-step process of DMAIC. The first stage in Six Sigma is to define and identify the problem that needs to be solved (Hsieh et al., 2007). Defining the problem is

one of the most critical stages of the entire process because the problem should be well defined and well understood by those attempting to solve the problem (Antony et al., 2017). In the second stage, the Six Sigma practitioner needs to decide what to measure and the best way of measuring performance (Hsieh et al., 2007). The third stage is analyzing the collected data to determine the problematic part of the process (LeMahieu et al., 2017). The fourth stage, improvement, involves using the output of the analysis phase and deciding on a course of action and how to implement and monitor the solution (LeMahieu et al., 2017). The control stage is the final step wherein the Six Sigma practitioner monitors and measures the defined problem for reduced variation or that the process is now under control (Hsieh et al., 2007).

The Six Sigma practitioner uses the DMAIC process to improve existing processes. There is a slight variation of these stages when approaching a new process or in working with software that uses the same first three words as DMAIC (i.e., design, measure, and analyze), but the last two words are design and validate (E. V., Antony, & Sunder, 2019). In these last two phases, the Six Sigma practitioner designs a solution to the new problem, and the end-customer may have the last word in validating that the solution does meet the requirements (Mouaky, Benabbou, & Berrado, 2018). However, in the initial phase of defining the variables to measure, the factors that most affect quality may not be easily identifiable. If the initial qualify factors are not obvious, a Six Sigma practitioner may resort to more subjective approaches, such as brainstorming, to identify root causes of variation (Cox et al., 2016). In these cases, a Six Sigma practitioner can use the process variation diagnostic tool as a more objective measure in

determining key factors affecting the variation, which requires sampling a small number of products on several factors and applying the improvement tool (Cox et al., 2016).

Not all organizations have fully embraced the Six Sigma philosophy, even with the proven results and the endorsements of successful corporations such as Motorola and General Electric. One common hindrance is buy-in from upper management as well as an understanding by the general workforce that reduction of variation can help the organization as a whole (Psomas, 2016). Another weak area for Six Sigma is the lack of cooperation between academics and fields outside of manufacturing, although there have been recent improvements (Reosekar & Pohekar, 2014). It is also challenging to use Six Sigma in disruptive innovation, and some organizations have even found that Six Sigma stifles innovation and creativity (Sony & Naik, 2019). One study found that over 60% of companies that initiated Six Sigma methodologies failed to develop the desired results (Antony et al., 2019).

**Lean manufacturing.** Entangled in Six Sigma is the concept of lean manufacturing, which focuses less on data collection and focuses more on applying known and tested methods of improving performance or reducing errors (Antony et al., 2017). Another concept of the lean philosophy is that a business focuses only on those activities that address customer needs and strip away anything that does not add value to the customer (Anthony & Antony, 2015). In the United States, Womack, Jones, and Roo defined the lean movement in 1990 in their book titled, *The Machine that Changed the World*, in which the authors wrote about the idea based upon their observations of the Toyota Production System for manufacturing automobiles in Japan (Anthony & Antony,

2015). Even though the Toyota Motor Company had been using their approach since their inception in the 1930s, Toyota never documented their process until the 1960s because they felt the procedure was merely common sense and too basic to bother codifying (Anthony & Antony, 2015).

From these two approaches was derived yet another methodology in improving manufacturing and services known as Lean Six Sigma, which attempts to synergize between lean and Six Sigma by making changes more quickly to improve the output (Sreedharan et al., 2018). Another article identified Six Sigma as being top-down and management-driven compared to Lean, which is more bottom-up, shop-floor driven, making the two philosophies all-encompassing and well suited together to bring value to both the company and customer (Anthony & Antony, 2015). In one case study, E. V. et al. (2019) found that Lean Six Sigma reduced average complaint resolution time by about 30% and reduced variation in solution times by nearly half. Searching more recent academic articles for Six Sigma, it becomes challenging to find an article on one of these continuous improvement philosophies without some mention of the others.

**Design of experiments.** One advanced statistical method used in the analysis phase of Six Sigma is the DOEs. DOE also lends itself well to the field of IT because DOE allows for the testing of many factors at once using specific combinations of patterns representing the configuration of services, such as databases (Hsieh et al., 2007). DOE allows researchers to observe the effects of individual factors as well as the interaction between combinations of factors (Hancock & McNeish, 2017). There are four stages in evaluating the performance of a system using DoE: declare the objective of the

experiment, design the experiment, conduct the experiment, and analyze the data (Reddy & Shyamala, 2016).

Once the Six Sigma practitioner defines the goal of the investigation, much like the initial stage of the Six Sigma process, the practitioner then needs to identify factors to study in designing the experiment (Ustinova & Jamshidi, 2015). The identified factors must be controllable by the researcher. Once such factors are determined, the researcher experiments with different levels of values, usually one high and one low value, on each factor to see which factors have the most significant effect on the system (Ustinova & Jamshidi, 2015). Choosing two levels for each factor also reduces the number of times an experiment is run, which reduces costs, time, and complexity it takes to run every continuous combination of factors (Jia et al., 2017). The commonly used, two-level, full-factorial design has all combinations of factors ran at two levels in the experimentation (Ustinova & Jamshidi, 2015). The change in outcome based on a single factor is known as the main effect, and changes based on combinations of multiple factors is known as the interaction effects; a full-factorial analysis is a method used to capture both of these effects (Jia et al., 2017). With two levels for each factor chosen and the number of factors commonly referred by the variable *k*, then the number of experiments performed is denoted as $2^k$ (CITE). For example, if four factors are chosen ($k = 4$), then the number of experiments performed in a full-factorial analysis would be $2^4 = 16$ experiments.

The order of experiments should be randomized to ensure extraneous factors do not play a part in the experimentation (NIST/SEMATECH, 2012a). Furthermore, there should be multiple runs of each combination of factors to determine that there is

consistency within the results (NIST/SEMATECH, 2012a). Replication also provides pure, error-free degrees of freedom in estimating the error variance (Jones & Montgomery, 2017). In continuing with the above example with four factors and two levels, for 16 runs with three trials each, I performed 48 experiments and used the average of the three results for the final analysis of variance.

Full factorial analysis has found its way into the field of IT as well. Reddy and Shyamala (2016) used factors of the number of clients, the type of system virtualized (i.e., web server, application server, or database server), and hypervisor software (i.e., VMWare, Xen, or Kernel-based Virtual Machine [KVM]) to develop a scoring system to rank the performance of hypervisor software. The measure used to determine the outcome was the mean central processing unit (CPU) utilization in each combination of experiments (Reddy & Shyamala, 2016). In a similar approach, a different team of researchers used a combination of algorithms to efficiently find outliers in sets of data (Orair, Teixeira, Meira, Wang, & Parthasarathy, 2010). One common thread between these two experiments is that they did not choose discrete numbers for all their factors. Reddy and Shyamala used different software as two of their factors, and Orair et al. (2010) turned different algorithms on and off for each of their experiments. These experiments go on to show that researchers can perform the analysis while adjusting the factors to values other than just numbers but discrete categories as well.

**Define, measure, and analyze.** For this experiment, it is appropriate to discuss the first stages of Six Sigma as it applies to database efficiency. LeMahieu et al. (2017) identified that the first step is defining the outcome of the experiments. While several

researchers attempted to improve the cost efficiency in databases, usually by way of reducing the power consumption of servers running database management systems (DBMSs; Ferretti, Pierazzi, Colajanni, & Marchetti, 2014; Han, Ghanem, Guo, Guo, & Osmond, 2014; Zhao et al., 2016), most of the articles in the literature measured database performance by throughput (Loghin, Tudor, Zhang, Ooi, & Teo, 2015; Narasayya et al., 2015; Xiang, Li, Chen, Guo, & Yang, 2016). Response time was another metric presented in the literature, but this metric was related to how users felt about the database performance rather than an objective metric (Liaqat et al., 2017). While the actual costs of database performance could have been the dependent variable here, one group of researchers presented the difficulties in trying to determine the actual costs of running a database server. The researchers cited the costs of hardware, software licenses, and power consumption as uncertain variables (Tak, Urgaonkar, & Sivasubramaniam, 2013). The elimination of other metrics left throughput, as defined by operations per second, as the dependent variable for this experiment. Furthermore, nearly all of the articles used the TPC benchmarking standard as the tool to produce and measure throughput on the databases, giving an academically supported instrument to use for these experiments.

Part of what makes the TPC benchmarking standard work is how the benchmark defines the number of users on a database system. Since the TPC standard is the proper instrument to use for the experiment, the number of users, virtualized users, in this case, was one of the independent variables. Remote systems could be responsible for query execution, but in the end, those remote systems would be a reflection of end-user action.

In typical business settings, users make use of systems in the course of their work. In the United States, "typical" business hours are from 9 a.m. to 5 p.m., or within an hour or 2 of this window. While working in an enterprise-sized environment, I would commonly see database usage start very low, then rise to a peak at midmorning, dipping down around noon, rising to a second peak during the midafternoon, before dropping down very low for the rest of the day. From this experience, I have identified the time of day as one of the independent variables for this experiment.

The other two independent variables for this experiment, the database settings of the InnoDB buffer pool and InnoDB I/O capacity, are critical choke points for database performance cited in the literature. Narasayya et al. (2015) identified buffer pool memory as essential to database performance and identified shared buffer pool memory in a multitenant environment like the cloud as problematic. The complication of buffer pool memory in multitenant environments that lead me to identify buffer pool memory as an essential factor, and why I investigated multiple cloud providers to see how each manages this particular factor. Among several teams of researchers, Sharma, Nelson, and Singh (2016) started their article by stating disk I/O is the most common problem facing DBAs. As with the buffer pool memory, multi-tenant environments may cause contention when two VMs are experiencing heavy read/write jobs (Dean et al., 2016). I discuss how the literature has approached the factors indicated above in later sections of this literature review.

**Analysis of similar theories.** Six Sigma was not the only theory found in the literature in the discussion of database efficiency. Several other theories were mentioned

in the literature and were potential candidates for this study. In the following sections, I discuss and define alternate foundational theories for this study and how they did not work for this experiment.

***Queueing theory.*** One other theory encountered in the review of the literature is queueing theory. Danish engineer Anger Erlang first introduced queueing theory in 1909 (Mandelbaum & Hlynka, 2009). While working for the Danish telephone company, Erlang proposed that a Poisson distribution best modeled the number of calls arriving at a telephone exchange during a given time interval (Kingman, 2009). Over the years, Erlang expanded upon is initial research to cover more complicated circumstances (Kingman, 2009). Many national post offices and telephone networks made use of Erlang's queueing theory (Kingman, 2009).

Queueing theory is used in computing to calculate and predict performance measures, such as finding the length of a line or predicting wait times (Yang, Cayirli, & Low, 2016). In the real world, production systems can experience a wide variation in workload demand, by a varying number of users. This changing workload can make mathematical modeling of a queue very difficult (Yang, Cayirli, et al., 2016). Essentially, queueing theory is the study of waiting in line (Hilier & Lieberman, 2015). One feature of a queue is an input source that feeds a queue or line. A service mechanism or server handles each member of the series (Hilier & Lieberman, 2015).

While I could make an argument to fit the independent variables for this experiment into queueing theory, I chose to pursue Six Sigma as the theoretical foundation. As presented in the Theoretical and Foundational Frameworks earlier,

because of the high and low values used for the independent variables fit in better with the design of experiments and, consequently, Six Sigma. Furthermore, queueing theory makes use of complex mathematical models to predict or measure performance, as usually measured by customer wait time (Yang, Cayirli, et al., 2016). With this fact in mind, I determined that the queueing theory was not the best choice for this experiment.

***Theory of constraints.*** Another theory mentioned in the literature, particularly in conjunction with Six Sigma, is the theory of constraints (TOC). Author Eliyahu Goldratt first presented TOC in 1984 (Hudson, 2017). Where Six Sigma proposes to improve the system process by reducing variability, lean works to improve performance, TOC focuses on improving performance by removing bottlenecks in processes (Hudson, 2017). The bottlenecks, in this case, can be physical, like in a manufacturing production line, or procedural, like in a software system. The improvement process under TOC involved five steps. For the first step, a researcher identifies a constraint. Then the researcher decides on how to reduce the impact of the constraint. From here, the researcher focuses on all aspects of only the individual constraint. Next, the researcher makes a change to the constraint. Finally, the researcher observes the overall system for new restrictions (Aryanezhad, Badri, & Rashidi Komijan, 2010). If all goes well, the identified constraint is no longer restricting process flow, and a new bottleneck becomes apparent and corrected using the same five steps (Aryanezhad et al., 2010). Since disk I/O was identified as a constraint to a database system, TOC seemed like a potential candidate for the theoretical foundation of this study. However, disk I/O is not the only factor under

consideration, and TOC focuses on a single factor at a time, which would not make TOC the best choice for this study.

**Big data and Six Sigma.** Big data often means data that comes in large volumes and at high velocity (Abadi et al., 2016). For performance improvement, some businesses keep the data locally to handle the high throughput of data. With large amounts of data, it can also be cost-prohibitive to transfer this data in and out of the cloud (Assunção, Calheiros, Bianchi, Netto, & Buyya, 2014). It is with this volume and velocity of data that DBAs must find ways to improve performance to keep the data pipeline flowing at the desired rate. Multiple authors looking at the future of database research agreed about the importance of database performance in this era of big data (Abadi et al., 2016; Assunção et al., 2014).

Six Sigma practitioners are using big data to assist with the measuring and analyzing steps of the DMAIC process. With large volumes of data coming from multiple sensors at a high velocity, engineers can use all of this data within the Six Sigma framework to make decisions more quickly (Duarte, 2017). One group of researchers proposed using Six Sigma, along with large amounts of customer data to help drive business processes. Jha, Jha, and O'Brien (2016) identified improved customer service as a goal and identified factors that should improve that goal: customer reviews, searches for similar products, weblogs, and images. The article concluded with the idea that businesses should not wait to analyze data until the system moves the data to a data warehouse but to use Six Sigma and big real-time data to drive business improvements. Diverse fields such as higher education (Laux, Li, Seliger, & Springer, 2017) and oil

fields (Xu, Wang, Wu, Shi, & Lu, 2017) are all using Six Sigma in conjunction with big data to make improvements in their systems.

## Analysis of Potential Themes

In reading the academic literature, several common themes emerged. After finding similar software, concepts, and approaches to database efficiency, the literature informed me about how researchers have been considering database efficiency. It was these common themes that lead me to choose the factors and instruments for this study. In the upcoming sections, I will review these themes that have assisted in preparing me for this experiment.

### The TPC Benchmarks

In deciding the database performance measures to use, and the tool to make the measurements, researchers favored the benchmarking standard created by the TPC. The literature may vary on the version of the benchmarking standard, but the literature also discusses different TPC benchmarks used in multiple experiments. Many of the articles presented in this literature review used variations of the TPC benchmark and explained why some researchers used more than one TPC benchmark to measure performance under different circumstances. The most widely used standard was the TPC-C benchmark standard that provides an intensive workload that emulates an online transaction processing system (Loghin et al., 2015). The TPC-C emulates read, write, and update queries that would commonly run in a commercial business where goods are ordered and shipped, with queries that would describe stock levels, order status, payment processing, or create a new order (Ferretti, Colajanni, et al., 2014). Users can alter

several variables at the start of TPC-C testing, such as the simulated number of warehouses used in this business emulation, the number of simulated users, and the volume of transactions submitted by the simulated users (Tian, Huang, Mozafari, & Schoenebeck, 2018).

The second most popular benchmarking specification was the TPC-H, which simulates a decision support system database environment comprised of only two update queries and 22 read-only queries (Barata, Bernardino, & Furtado, 2014). In their pursuit of determining energy usage, Xu, Tu, and Wang (2015) measured the differences between their proposed model of energy usage against the actual energy usage for each of the 22 read-only queries of the TPC-H. Loghin et al. (2015) pointed out that it is crucial to flush the file system cache each time these 22 read-only queries execute so that the data comes from the database rather than cached data when using the TPC-H benchmark standard. The fact that a variety of hardware, from cloud-based hardware, ARM processors, or solid-state storage, is a commonality in the literature in the discussion of the TPC-H benchmark.

**Hardware Methods of Improving Database Performance**

Throughout the research on database efficiency, researchers often cite disk I/O as one of the main bottlenecks for database performance. Researchers have attempted to improve this bottleneck by using caching methods, as well as various disk configurations to improve performance. Focusing on the input-output, some researchers have tried numerous algorithms to improve the input or output of a system.

**Caching methods.** One popular area of research into improving database research was different caching methods. In most computer systems, the main bottleneck is processing data lies in writing and reading data from the disks (Sharma et al., 2016). Database systems keep data stored in the cache to avoid disk I/O, which is an area of the main system memory allocated for this very purpose (Tailor & Morena, 2017). DBMSs employ several popular algorithms in determining which data to keep in the database cache. Least recently used (LRU) is the most popular method for removing data from the database and is used in most commercial databases (Tailor & Morena, 2017). With LRU, the database software keeps track of the age of pages in the cache and removes the oldest pages when more space is needed (Tailor & Morena, 2017). In their review, Tailor and Morena (2017) stated that Least frequently used is the second most popular algorithm in cache management. Least frequently used is a method that keeps the most popular buffer pages in the system memory.

Cache size is a variable in most database systems that can be adjusted by a DBA. The size of the cache may depend on the amount of available memory and the size of the database. In reviewing the literature, Nanda, Chande, and Sharma (2017a) found that researchers have varied on the optimal size of the database cache, with most researchers reporting 10% to 20% of the size of the database being optimal. Even in their experiments with adjusting cache sizes, Nanda et al. (2017a) found that a cache of 15% of the size of the database was optimal in cloud-based systems. This group performed multiple experiments involving the cache in cloud databases. Since the overall size of the cache can be unlimited in the cloud due to the elastic nature of resources in cloud

services, more cache can be better. In a separate experiment, Nanda, Chande, and Sharma (2017b) determined that placing the entire database in the cache usually outperformed entirely disk-based databases. Memory-based databases struggled when the types of queries were very different, and the database system had to pull unique sets of data each time rather than deliver a response already held in memory (Nanda et al., 2017b).

While most authors have focused on scaling-out in the database literature, others considered scaled-up systems. With today's technology, multicore processors are becoming more and more inexpensive. With more cores comes more worker threads hitting the database with queries. Even with sufficient cache sizing, contention can occur on the data pages in the buffer (Ding et al., 2016). In their work, Ding et al. (2016) introduced a unique buffer management algorithm using batching requests and prefetching data called BP-Wrapper. The authors went on to prove that BP-Wrapper can dramatically increase data throughput in both physical and virtual systems when implemented (Ding et al., 2016).

**Disk and storage variations.** As discussed in the previous section, there have been many approaches to improving the speed of database systems by studying the cache. Other authors have looked at making improvements on the other end of the disk I/O system by considering variations in storage options for database systems.

In his review, Richardson (2014) considered several approaches to improving database performance at the disk level. Even though a typical serial advanced technology attachment disk can transfer data at 750 megabytes (MB) per second, seek times for the

disk limit performance, which can locate data only at speeds of 250 times per second (Richardson, 2014). Solid-state drives (SSDs) can perform about 60 times faster than the standard hard drives, but SSDs also have a fixed number of reads and writes before they fail. Chrószcz, Łukasik, and Lupa (2016) confirmed these findings in their experiments with different database management systems on standard disks and SSDs. Using SSDs can load spatial data 4 times faster and can return query results up to seven times faster (Chrószcz et al., 2016). Proper implementation of indexes can improve performance, as well as using the right database management system for the type of data being stored (Richardson, 2014). For example, researchers found the document-oriented database MongoDB to be less effective in handling spatial data when compared to standard relational databases (Chrószcz et al., 2016).

In a different approach to hard drive considerations, Sharma et al. (2016) discussed different striping options for databases. With striping, data is written to multiple disks simultaneously, which improves the I/O rate (Sharma et al., 2016). If the database is capable of striping, there may be a processing overhead associated with separating and finding the data (Sharma et al., 2016). The operating system layer can also manage striping, which would move the processing overhead away from the database software and onto the operating system (Sharma et al., 2016). Using a redundant array of independent disk (RAID) hardware technology, striping at the physical layer removes any processing overhead away from the database or operating system. However, striping may make it more challenging to monitor the status of the

disks or the I/O (Sharma et al., 2016). In all cases, separating the data from the indexes and avoiding chained rows can also reduce disk contention (Sharma et al., 2016).

A different approach to improving data retrieval is the concept of hot-data and cold-data. In this approach, frequently accessed data is kept in an area where the system can locate it quickly, and less-frequented data remain in the regions that are slower to respond. One approach made use of the system's main memory as the storage for hot data. While effective, it can increase costs to provide enough main memory for adequate storage of hot data (Afify, El Bastawissy, & Hegazy, 2015). In their work, Afify et al. (2015) introduced an algorithm to determine the differences between hot and cold data, and when to move data between the main memory and the hard drives. Using a slightly different model, Saharan and Kumar (2015) introduced the idea of keeping frequently accessed data at the edge of a network rather than sending the data to a cloud-based database. In Fog computing, multiple nodes within a network share the data, keeping the data only one hop away (Saharan & Kumar, 2015). Fog computing can provide low latency access to data, while still allowing for the data to be geographically dispersed, provided fog nodes are in the same location where the data is collected and accessed (Saharan & Kumar, 2015).

**Software Methods of Improving Database Efficiency**

In looking at different ways of improving database efficiency, researchers have considered the different ways that the InnoDB engine functions and altering some of the variables of the engine. Other researchers have focused on different approaches for handling buffer management. Many of the approaches taken by researchers consider

ways in which the MySQL database software functions in their attempt to improve on the existing product. Because I am using one of the buffer configuration settings as a factor in this study, it is only appropriate to discuss a review of the InnoDB engine, as well as how buffers can affect database performance.

**The InnoDB database engine.** One focus of attention among researchers was the InnoDB engine that controls the buffer pool, indexes, and other essential database operations (Yu & Pradel, 2018). Most of the authors focused on methods to alleviate locking and contention, while others concentrated on the InnoDB I/O to the disk. One commonality among all but one of the articles discussed in this section was the use of the TPC benchmark previously discussed.

The most basic article compares MySQL to Microsoft's SQL server. In their work, Almeida, Furtado, and Bernardino (2015) examined the performance of the two database management systems in handling progressively larger decision support systems in a star schema. In the experiments, both database management systems performed roughly the same with smaller databases. MySQL began performing worse with the increase of the size of a database, with differences becoming significant with database sizes around 6 gigabytes (GB) and larger (Almeida et al., 2015). The authors suggested that it is the columnar indexing present in Microsoft SQL Server that allows it to outperform MySQL in this testing. However, the authors fail to mention any variables of the InnoDB engine could have been any values other than the default values, which may also explain the differences in performance.

Two different articles approached the disk I/O, but from slightly different perspectives. One researcher looked at the buffer pool on different types of media and compared performance from each. In his work, Kong (2012), considered buffer pools stored on a striped RAID array, an SSD, and on the flash cache of the RAID controller. Overall, the solid-state storage for the buffer pools outperformed the standard disk-based storage, with the flash cache on the RAID controller performing better than SSD (Kong, 2012). Taking the data closer to the final storage solution, other researchers looked at the performance of the compression the InnoDB engine uses to store the data. In his experiments, Lee (2014) looked at the performance of non-compressed data, the default compression of the InnoDB storage engine, and an experimental compression proposed by the author. The proposed method uses a data compression method that puts less processing overhead on MySQL, which allows the overall system to perform much better than the default data compression (Lee, 2014).

**Buffer management.** Database management systems use buffers to improve the speed of data between the memory and the CPU and avoiding the slower I/O to the disk (Guo, Yu, Liao, Yang, & Lu, 2017). Researchers have focused on buffers in their search for improving performance. In general terms, bigger buffers can improve performance, but making the buffer too large can affect other parts of the system requiring memory (Yang, Jin, Yue, & Yang, 2016). Researchers also considered the algorithms used to decide when data is written to the disk and removed from the buffer.

There are several variables in the configuration of MySQL that can be adjusted to improve database performance, such as *query_cache_size*, *key_buffer_size*, and

*innodb_buffer_pool_size*. Vilaplana et al. (2014) used the open-source tool MySQLTuner to modify several variables in the MySQL configuration to improve performance for their application. Most researchers focused on online transaction processing (OLTP) database models, but Vilaplana et al. (2014) focused on databases supporting applications used for metabolic pathway reconstruction. By increasing the values of five different variables, the researchers reported they were able to improve performance by 30%.

Of course, if you could have an unlimited buffer, performance could improve significantly. Guo, Yuan, Sun, and Yue (2015) proposed an infinite buffer in the form of virtual tables used in a new approach to the job of extracting, transforming, and loading (ETL) data. In a normal ETL process, a query retrieves data to a temporary table where it can be manipulated to meet the users' needs, then loaded into another database. The transform, extract, and load process introduced by the researchers transform the data during the load process into "virtual tables" which exist in memory, or the cloud, much like an unlimited buffer. The researchers did find a little performance improvement, but they also focused on disk efficiency in reporting the results of the experiments. The authors admitted that their approach does have limited use cases and that the focus is more on reducing disk I/O and reducing storage needs in the ETL process (Guo et al., 2015).

But increasing the values of the buffer variables is limited by the amount of memory available in the system. While additional memory is one option, there are limits and other costs incurred, such as increased energy usage. In their work, Guo et al. (2017)

focused on the optimal values for the buffer variables to improve performance while keeping energy usage down, as additional physical memory or memory usage boosts power usage. Guo et al. created a system that measured energy usage and CPU activity and varied the buffer variables to reduce energy consumption while providing acceptable database performance. The researchers suggested that cloud providers should report slightly diminished performance in the service level agreement (SLA; Guo et al., 2017).

Other researchers suggested that the SLA be used to outline diminished performance for some customers. In multitenant database systems, multiple users have databases hosted on the same database server and are usually unaware of the other users in the system. Narasayya et al. (2015) outlined an SLA metering technique that would allow for overprovisioning of a buffer pool in a multi-tenant database system, giving favor to the higher paying customers by taking away available memory from the lower-paying customers. In their approach, Narasayya et al. outline a multitenant buffer page replacement algorithm named MT-LRU. Many researchers suggested different algorithms to replace buffer pages to improve performance. One approach frequently referenced is the work done by Jiang and Zhang (2005) in their algorithm low inter-reference recency set (LIRS) as an improvement to LRU. While LRU considers the recently used data, LIRS also uses the frequency that the data was accessed to predict future data requests. LIRS was used to improve database performance without interfering with system performance (Jiang & Zhang, 2005).

One team considered the storage media when designing a buffer algorithm. As previously mentioned in the Disk and Storage Variation section, databases can get a

performance increase using SSD's. However, SSD's are not efficient with random writes (Yang, Jin, et al., 2016). The Clean-First and Dirty-Redundant-Write algorithm reduces the number of small random writes and marks buffer pages that have already been written to disk, called clean pages, for removal before the buffer pages that will require writing to the disk, known as dirty pages. The dirty pages are then written in bulk to the SSD when no clean pages are left to remove from the buffer (Yang, Jin, et al., 2016). Experiments showed the Clean-First and Dirty-Redundant-Write algorithm reduced the number of random writes to the disks as well as improved performance over standard buffer algorithms (Yang, Jin, et al., 2016).

**Complications in Database Performance in Multitenant Environments**

With database management systems and virtualized systems, there is a distinct possibility that there may be other users on the system. Weather at the database level, or the virtualization layer, conflicts can arise in I/O, CPU utilization, or memory utilization. Researchers considered these limitations in several directions. It is these considerations that lead me to include the time of day as a factor. I started with the idea that most VMs in the cloud in an American data center would experience larger loads during a traditional American workday, from 9 a.m. to 5 p.m., since this was my observations having administrated hundreds of servers during my career. The following articles speak to the complications that could be happening on the cloud provider platform that may interfere with database throughput on a VM in the cloud.

**Virtual machines.** The consensus seems to be that MySQL runs slower on a VM than on a physical machine. In testing the performance of MySQL on Microsoft's

Hyper-V server, Ahmed (2013) found that the response time was slower, and the CPU usage when idle was higher for a VM running MySQL. Chung and Nah (2017) found similar results with lower disk utilization and lower CPU utilization on the physical server when experimenting with a MySQL backed web application. The MySQL-backed web application did perform a little better in CPU utilization when provided with more processing cores (Chung & Nah, 2017). In a virtualized environment, researchers have found more issues with disk I/O in database applications than other areas of shared hardware such as memory or CPU. One team compared the performance of MySQL, a disk based DBMS, and DB2, with in-memory databases, on both physical and virtualized systems (Tajbakhsh, Dehsangi, & Analoui, 2017). Experiments showed DB2 performing slightly better than MySQL in the areas of CPU utilization in all tests, and faster response times and more TPS in DB2 as the number of users increased on a physical machine. MySQL did do slightly better with TPS and response times with smaller numbers of users. For both DBMSs, performance in these areas was worse when virtualized on the Xen platform (Tajbakhsh et al., 2017).

Researchers reported similar results in testing MySQL with the Xen virtualization platform in a clustered environment. Tapdiya and Xue (2014) ran experiments with a MySQL front end and DBMS on a single VM, and again with the MySQL front end on a separate VM from the DBMS. The results from the two different VMs were usually less consistent, with spikes in response time and CPU utilization on one or more nodes of the cluster. The explanation for these spikes was the inadequacy of the Xen network I/O system in communicating between the front end and back end of the

MySQL server (Tapdiya & Xue, 2014). When the same experiments were ran using virtualized servers in Amazon's public cloud in the same configuration, the results were very similar between the single and dual VMs on a three-node cluster. However, the dual VM setup would stop responding when the number of simulated users increased. The dual VM setup would timeout with about half as many users as the single VM experiments (Tapdiya & Xue, 2014).

In recognition of how all the different moving parts of a DBMS on a virtualized system can be complicated, one team developed a tool to detect bottlenecks at the virtualization layer. In their work, Dean et al. (2016) created a tool called PerfCompass, which monitors the VM's CPU, memory, buffers, and dropped network packets. The team then tested PerfCompass with several different applications on a VM: Apache web server, MySQL, Tomcat, Cassandra, and Hadoop (Dean et al., 2016). In the tests, PerfCompass was able to successfully alert the researchers to I/O interference, CPU overutilization, memory spikes, and packet loss when testing MySQL in a virtualized environment (Dean et al., 2016).

**Cloud considerations.** Since this study took place on VMs in the cloud, it is only appropriate to review the literature concerning cloud considerations for VMs and databases. The cloud is appealing to administrators due to the elastic nature of storage, networking, memory, and processing power (Hummaida, Paton, & Sakellariou, 2016). While the administrator and the cloud provider can tweak many factors, the final determining factor for QoS is the SLA, and the amount of money the customer is willing to pay for improved performance (Whaiduzzaman, Haque, Chowdhury, & Gani, 2014).

Cloud users will need to ensure they have sufficient internet speeds to access the cloud, as well as understand the fact that they will be running databases in a multi-tenant environment (Januzaj et al., 2015).

There is much in the academic literature reviewing the many approaches to provisioning cloud computing services and attempts to optimize the services. In their work, Whaiduzzaman et al. (2014) consolidated these articles into categories of strategies for cloud provisioning. The literature has taken many different approaches towards cloud provisioning, such as the areas of objectives for migrating to the cloud, requirements for cloud services, metrics required of the cloud services, and the approaches to cloud provisioning like SLA or statistical-based approaches (Whaiduzzaman et al., 2014). In a slightly different overview, Silva Filho, Monteiro, Inácio, and Freire (2018) aggregated and categorized academic articles on a VM placement and migration. Areas such as resource usage, optimizing migration, SLA fulfillment, reduced energy and costs, and clustering considerations are groupings in the literature for VM migration and placement (Silva Filho et al., 2018). In yet a third angle in categorizing the literature, another research team looked at the critical features needed in cloud adaptation. In their review of the literature, Hummaida et al. (2016) categorized the needs into the areas of resources, objectives, techniques, engagement, decision architecture, and type of managed infrastructure.

Amply available in the literature were articles that fell in the categories listed above. One team developed and tested a process that used software-defined networking (SDN) to determine a location for a VM, selected resources to support the VM, and

monitored the status of the VMs (Gharbaoui et al., 2016). With their proposed SDN model, Gharbaoui et al. (2016) were able to realize higher traffic flow and more efficient resource utilization. Similarly, Henneberger (2016) developed a stochastic mathematical model that was proven to reduce peak demand costs and total overall costs of hosting VMs in the cloud. Researchers also reported on the many ways of approaching performance improvements. Working on the assumption that cloud providers desired to move active VMs to physical servers with less active VMs, one team developed a methodology to decide where to move the active VM. In their work, Tseng, Chen, Chou, Chao, and Chen (2015) use machine learning to observe the behavior of VMs in deciding which physical machine to migrate highly active VMs. Looking more deeply into VM migration, Kumar and Saxena (2015) explained the process of VM migration down to the memory page level. Their experiments showed the performance of live migration of a VM depended on the amount of memory assigned to a VM, bandwidth of the environment, and the rate at which the application hosted on the VM is writing to the memory (Kumar & Saxena, 2015).

Other researchers didn't look at such minute parts of the virtualized systems in the cloud but chose to look at the bigger picture. Lang et al. (2016) proposed a system of using historical data to determine the best location for VM placement. Similarly, Tseng et al. (2015) proposed a system that emphasized overprovisioning the physical machines as much as possible without violating the SLA. Lang et al. also went so far as to propose releasing only a few high-performance SLAs and more low-performance SLAs. One team specifically studied how the cloud would perform for scientific computing. Iosup et

al. (2011) tested the performance of four different public clouds using high-performance computing tasks, high throughput computing using database benchmarking, and many-task computing. The conclusion, at least at the time of the article, was that cloud computing is enough for temporary solutions, but may not be financially viable for long-term scientific computing solutions (Iosup et al., 2011).

One newer technology has come into more substantial use over the past few years that deserves consideration. Some companies are beginning to use container-based virtualization over hypervisor-based virtualization. From the user perspective, there is little difference between the two. Containers are like VMs in that they do run an operating system and software services. However, containers provide only the bare essential libraries and supporting software (Kozhirbayev & Sinnott, 2017). In their experiments, Kozhirbayev and Sinnott (2017) found that databases use about the same amount of memory and processing power in containers as VMs. However, containers perform poorly in I/O operations, which are critical for database management systems. Since containers are stateless, DBAs should not store data in a container, but store the data on persistent storage (Bhimani et al., 2016).

**Gaps in the Literature**

Many layers can affect database efficiencies such as the hardware, operating system, virtualization software, DBMS, database structure, data types, and query organization. Furthermore, there can be several factors at each level that can also impact database performance. With so many areas of focus, one theme missing from the literature was consistency in focus. In much of the literature, the researchers would come

upon a novel approach to addressing one particular factor and compare the proposed approach to a standard database installation. Another gap in the literature was a comparison of the most popular relational DBMSs with each other in how they performed using the researcher's approach. When researchers compare multiple DBMSs, researchers tended to include not only structured query language (NoSQL) databases. Developers designed NoSQL databases to function differently, and with different types of data, as opposed to relational databases (Chandra, 2015). As such, a comparison of relation to NoSQL databases is not an equal comparison.

Furthermore, there are differences between the types NoSQL databases (document, columnar, graph, key-value) in their intended use case and how they handle data (Chandra, 2015), and the literature fails to address comparing NoSQL database systems of similar types. Since I began down this path of research, there has been a significant rise in the offerings of database-as-a-service (DBaaS), where cloud providers now offer many different standard relational databases and many different NoSQL databases where the cloud provider manages the DBMS. Because cloud providers have only recently provided these services, there is currently a lack of research in comparing the efficiency of DBaaS services.

## Transition and Summary

As shown in the review of the literature, database performance is affected by hardware and software configurations, some of which can be set by the DBA. When the hardware is virtualized and shared in a multitenant environment, there can be factors that affect database performance that is not only outside of the control of a DBA but outside

of a DBA's ability to view. Most DBMS manufacturers have recommended settings for optimizing databases, both on-premises and in the cloud. Because of these limiting factors, this research focuses on the InnoDB buffer pool size and the InnoDB I/O capacity, which are factors that can be controllable by the DBA and see how these factors interact with the number of users and the time of day.

In Section 2, I will explain the role of the researcher, and explain the use of the design of experiment methodology needed to quantify the various possible factor levels for the InnoDB buffer pool, InnoDB I/O capacity, simulated number of users, and the time of day of the measurements. I will also justify the use of the TPC-C benchmarking standard as implemented by HammerDB, which measured the dependent variable for this experiment.

# Section 2: The Project

In Section 2, I restate the purpose of the research, overview my role as a researcher, and review the methodology and tools that I applied to the research. The research method was quasi-experimental in nature and was focused on the manipulation of MySQL database instances running on MVs hosted in public cloud environments. In the review of the tools and methodology that I used to run the experiments, I focus on justifying why I chose these tools to help me explain the analysis of the data and the validity of the selected method for analyzing the data for this experiment. In closing out this second section of the study, I also justify the validity of the experiments and analysis performed here.

## Purpose Statement

This experiment was about finding the optimal combination of four factors that affect MySQL database performance. Specifically, I controlled four factors: time of day, the number of concurrent users, the InnoDB buffer pool size, and the InnoDB I/O capacity at a high and low level. The dependent variable was the throughput of MySQL as measured by the number of TPS. I conducted these experiments on similarly provisioned VMs on the public cloud platforms of AWS, Google Cloud, and Microsoft Azure provisioned in datacenters in the United States to avoid latency. I selected these platforms because they are the top three public cloud providers (see Sikeridis et al., 2017). All three cloud providers also allow the provisioning of MySQL for free for small scale instances. This study contributes to social change because it informs DBAs on identifying the combination of controllable factors that maximize throughput, improve

query effectiveness and overall database utilization, and reduce costs for organizations utilizing the cloud. These efficiencies can help nonprofit organizations and schools provide better services more efficiently to their clients and make better use of their resources.

## Role of the Researcher

In experimental research, the researcher's role is to determine if a cause-effect relationship exists between one or more factors by selecting and manipulating the independent variables and observing the effects on the dependent variables in an unbiased way (Ellis & Levy, 2009). For research to be credible, the researcher must have a persistent and prolonged engagement in the subject area and spend sufficient time engaging in case studies and field observations (Houghton & Casey, 2013).

I have been actively working with databases at a professional level for 20 years now. While observing database usage in the enterprise, I would notice database usage would follow the same patterns over time. The database usage would tend to be higher during the late morning and midafternoon of the workday. This experience led me to choose the time of day and the number of simulated users as two of the key factors included in this experimental research. If a database were experiencing slow query responses, some hardware-related approaches an individual could take would be to increase the size of the cache or increase the amount of memory available to the DBMS. These performance-boosting approaches, along with the academic literature, led me to choose the InnoDB buffer pool size as well as the InnoDB I/O capacity as two additional factors to evaluate in this experiment. For the number of simulated users, the low and

high values chosen represent a light or heavy load that may be experienced by an enterprise-level database. The low values for the I/O capacity and buffer pool were the default values given in a MySQL installation, and the high values were recommended by MySQL for the specifications of a VM running the DBMS. I chose the times of day chosen based upon the peak business hours and off-peak business hours as experienced in the United States.

The database selected for this experimental research was MySQL. MySQL is an open-source DBMS and was chosen partly due to the free nature of open-source software. I also chose MySQL because of its frequent use in the academic literature in experiments involving relational databases (see Luo, Zhou, & Guan, 2015; Raza, Kumar, Malik, Anjum, & Faheem, 2018; Tapdiya & Xue, 2014). Other options for open-source databases are MariaDB and PostgreSQL, but the academic literature did not mention these DBMSs as frequently as MySQL. Microsoft offers a freely available version of its SQL Server under the developer edition; however, there would have been extra costs involved in the licensing of the Windows operating system in addition to the lack of academic literature on this DBMS.

I used HammerDB as the tool to simulate the users and user activity. HammerDB (2018a) implements the TPC-C benchmarking standard against the database. The TPC-C benchmark was chosen as the appropriate measure here because this is the benchmark mentioned frequently throughout the literature. HammerDB reported the resulting activity of the database in TPS. I performed the final analysis of variance using the statistical software of Statistical product and Service Solutions (SPSS), which is a

popular software used for statistical analysis (see Marshall & Jonker, 2010). Just as with MySQL, HammerDB is an open-source product and freely available. Other benchmarking tools are capable of implementing the TPC-C benchmark, but there are licensing costs associated with them. Another benchmarking tool mentioned less frequently in the literature is sysbench, which is freely available. Sysbench is capable of emulating OLTP workloads, but there is no mention of implementing the TPC-C benchmark standard, which researchers frequently used in the literature.

In performing experiments, a researcher must engage in ethical behavior at multiple levels. In general, a researcher must show respect, make efforts for the well-being of those involved in the study and avoid injustice to the participants (CITE). Those participating must grant their informed consent to be involved with the study and that they understand the ramifications in volunteering for the study (Department of Health, Education, & Welfare, 1979). Since this study only involved changing settings for a DBMS or benchmarking tool on a virtual server in a public cloud environment, no human subjects were involved in any part of this study. Therefore, my role as a researcher was contained in the experimental database settings alone and was not related to the choice and treatment of sample participants.

## Participants

As stated in the preceding section, there were no human participants in this study. This experiment involved changing DBMS settings at different times of day by simulating queries to the database originating from simulated users employing a commonly used, freely available, database-benchmarking tool called HammerDB

(2018a), which implements the TPC-C benchmarking standard. It was necessary to use simulation software like HammerDB to maintain a static number of users at high and low levels. It would have been resource prohibitive to coordinate large numbers of real users simultaneously and have all actual participants utilizing the database in the same way for every testing instance.

## Research Method and Design

In this study, I considered whether a combination of a specific number of variable factors determines the optimal throughput of data as measured by TPS. The research question was focused on finding the optimal combination of various factors that will positively affect MySQL performance hosted in a cloud environment. The research method used in this study was quantitative in an attempt to quantify the relationships in the numerical data (see Albers, 2017). Specifically, I applied an experimental design using a full-factorial analysis, in which I changed each factor from a low to a high level until all combinations of levels were attempted (see Reddy & Shyamala, 2016).

### Method

The three methods used in research are quantitative, qualitative, and mixed methods (Ellis & Levy, 2009). Quantitative research methods are essential in justifying the research, while qualitative methods are excellent tools for discovery, testing, and revising (Ellis & Levy, 2009; Park & Park, 2016). In the academic literature, most of the research in finding optimal database performance uses quantitative experimentation (see Chang & Lin, 2016; Raza et al., 2018; Shmueli, Vaisenberg, Gudes, & Elovici, 2014). In this study, my intention was to justify the optimal combination of factors that can lead to

optimal database performance. Qualitative studies are more exploratory, seeking to understand why a phenomenon occurs (Barnham, 2015; Ellis & Levy, 2009; Park & Park, 2016). In this quantitative, quasi-experimental research, the goal was to quantify the optimal outcome based on a combination of measurable-factor levels but not why this combination was optimal or the opinions or experiences of users who implement this combination. Since mixed-methods research is a combination of qualitative and quantitative approaches (Kansteiner & König, 2020) and this experiment did not involve qualitative methods, the mixed-methods approach was not suitable.

**Research Design**

In the realm of quantitative methods, the research design used depends on what the researcher is trying to determine (Abramson et al., 2018). For example, there are many correlational designs in the academic literature in the study of the performance of databases (Raza et al., 2018; Xu et al., 2015; Zhou, Taneja, Qin, Ku, & Zhang, 2017) in which the relationship between variables is studied (Gabbiadini & Greitemeyer, 2017). As I mentioned in the Nature of the Study section, since the relationship between two variables was not under investigation or was how changes in one variable affect the other, a correlational study was not appropriate in this experiment. Since a casual-comparative design does not involve a researcher controlling the independent variable, as I did in this experiment, a casual-comparative design was not appropriate either (see Ellis & Levy, 2009).

On the other hand, the use of descriptive designs, in which the researcher uses the data to draw a general conclusion from the data (Fisher & Marshall, 2009), was also not

in common in the explored literature. Drawing general conclusions from the data, as a researcher would do with a descriptive design, would not have been useful in answering the research question posed in this study. In terms of experimental treatment and control types of designs, there were many articles in which the researchers measured the effects of a proposed intervention on a database or system to improve performance (CITE). Since this study lacked randomness in the selection of subjects or the application of a treatment, an experimental design was not appropriate (see Abramson et al., 2018). Quasi-experimental designs are used when the researcher intervenes with a treatment (Bärnighausen et al., 2017). Since I intervened by way of applying all combinations of high and low values for the factors in this experiment, the quasi-experimental design was the most appropriate for this study.

In terms of quasi-experimental designs, two groups of researchers discussed the use of factorial designs in studying the effects of different factors on database performance (Bizarro, 2015; Gonçalves, Guimarães, & Souza, 2014); however, neither group chose a similar sets of factors. Bizarro (2015) used DBMS platforms as one of the main factors, along with task characteristics, user characteristics, and database representations as additional factors. Gonçalves et al. (2014) used query algorithms, query shapes, and the quantity of table joins as factors in their experiments. In a literature review and research spanning over 2 years, I was unable to find research that used a full-factorial analysis to optimize throughput on a database using the combination of factors used in this study. This combination of MySQL running on a VM on a public

cloud provider meets all of the criteria as a platform to address the research question for this study.

## Population and Sampling

The population for this study was the public cloud providers with the capability to host the MySQL DBMS. My sample selection from this population was the three largest public cloud service providers measured by market share: AWS, Microsoft Azure, and Google Cloud Platform (see RightScale, 2019; Sikeridis et al., 2017). Each of these platforms offers free introductory trial periods for the small instances used in these experiments. I chose these platforms so that my research would be the most applicable to more DBAs. By definition, this sampling method was purposive, nonprobabilistic sampling, where the researcher deliberately selects the population for important information (see Taherdoost, 2016). I provisioned the VMs on all cloud platforms to enable uniformity in platform testing and comparisons. I installed the latest production version of Debian Linux on each VM, and the DBMS installed was the latest production version of MySQL as provided by the Debian package manager.

The benchmarking software, HammerDB (2018a), implements the TPC-C benchmarking standard, which outlines the method for emulating OLTP and measures throughput as the benchmark. During the throughput testing, HammerDB samples and averages the throughput every 10 seconds during the scheduled testing time. As previously mentioned in the Hypothesis and Design of Experiments sections, this full-factorial design involved four factors at two levels, each for $2^4 = 16$ factor combinations. I performed the TPC-C benchmarking test for all 16 factor combinations with three

replications (see NIST/SEMATECH, 2012a), for a total of 48 samples across all factor levels. Replications at the same factor combination help produce more precise regression coefficients, which allows for the study of variation in the outcomes, and are used to estimate the errors for statistical tests on the effect of factors and insure against bad runs or measurements (Minitab, 2018).

One tool used that calculates sample sizes based on desired effect and power is G*Power (Hancock & McNeish, 2017). The power of a statistical test is the probability that a researcher will correctly reject the null hypothesis (Hancock & McNeish, 2017). Given that the factorial experiment was analyzed in SPSS using the Generalized linear model (GLM), I can select a multiple linear regression a priori power analysis in G*Power (Faul, Erdfelder, Lang, & Buchner, 2007). The most common level for avoiding a Type I error is $\alpha = 0.05$ (Smith, 2012). Acceptable power levels to avoid a Type II error are above 0.8 (Dien, 2017), so for this experiment, I selected $\beta = 0.1$, leading to a power of $1 - \beta = 0.9$. A smaller effect size is required to detect smaller differences between the groups (Sullivan & Feinn, 2012), so for this experiment, I chose a smaller effect size of $f^2 = 0.35$. With four factors, G*Power calculates that 50 samples will give an actual power of 0.907, as shown in Figure 1. Fifty samples divided among $2^4 = 16$ different factor combinations averages to 3.1 samples per factor combination, which I rounded down to 3 samples or three replications per factor combination. In all, I collected 48 samples from each of the three public cloud providers, with the results for each provider reported separately. Justifications for the values used above are discussed

further in the Validity and Reliability section in the explanation on how I will address Type I and Type II errors for this experiment.

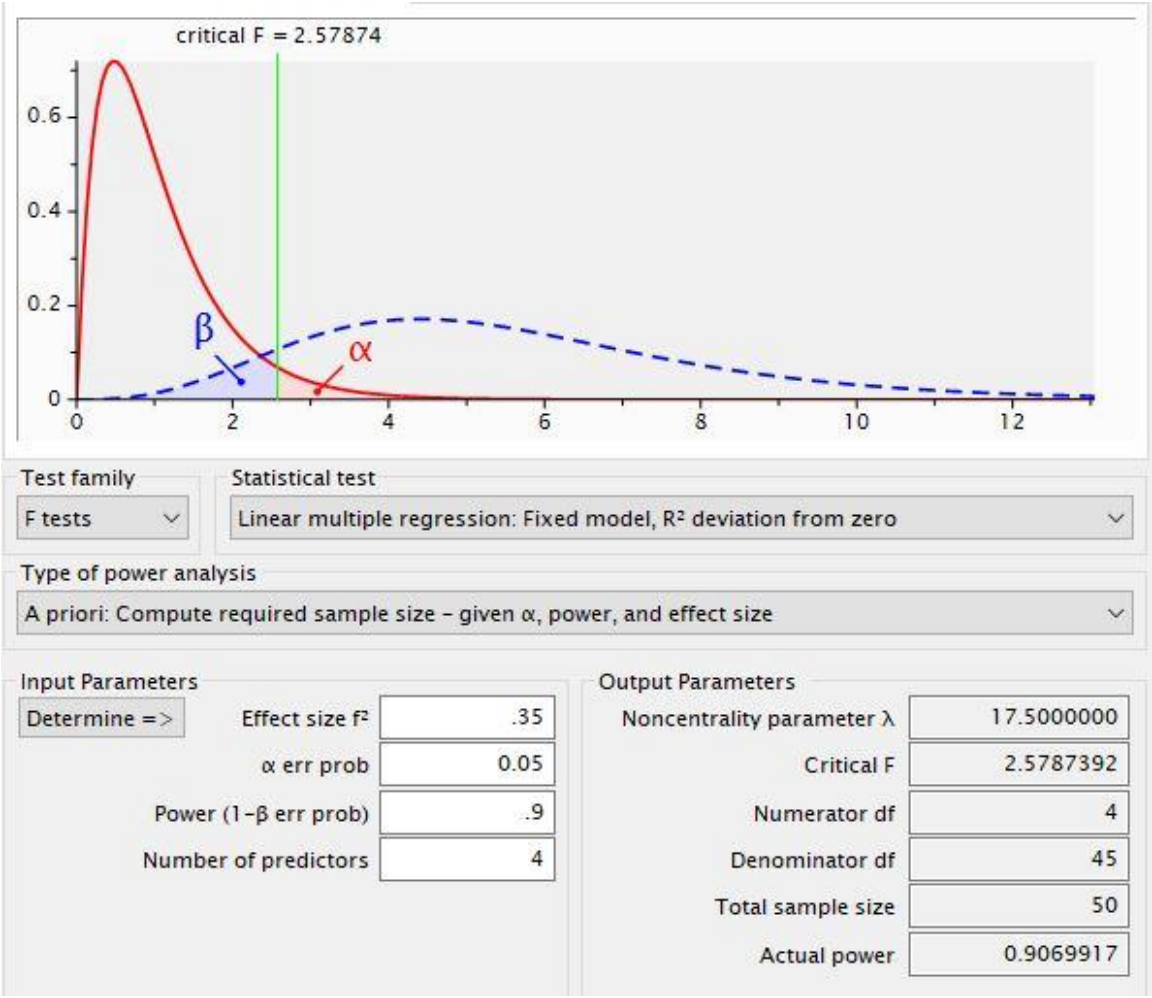


*Figure 1*. G*Power calculations determining sample size and power.

## Ethical Research

Since there are no humans involved in this testing, there is no one to consent to these experiments. With no participants, there is no process for withdrawing from the study since there will be no person to withdraw. I did not provide any incentive to any participant. With no participants, there are no agreement documents are necessary. With

no participants, there is no risk of loss of privacy, emotional or psychological distress, or economic loss to any individual or organization. The data written to the database by the benchmarking software HammerDB contains randomized text, for example, FKaak9ZBgtJr3Tr6gESW (HammerDB, 2018c). Since the data are random, there is no personal data to protect contained in the database. The dependent variable of throughput does not relate to any person or organization. I am not an employee or customer of the companies whose cloud services used for these experiments, and therefore, have no vested interest in the outcomes as they may compare to each other. I conducted these experiments within the parameters stated in the acceptable user agreements and terms of use for each cloud provider. Because any researcher or DBA has access to public cloud platforms to repeat this research, nor is any private company information collected, there is no need to keep the organizations confidential.

On the conclusion of the experiments, I will store the data on a thumb drive and compact disk, with copies saved in a safety deposit box for 5 years. The Walden Institutional Review Board approval number is 06-25-0623603, and I have provided a copy of the National Institute of Health Training Certificate of Completion in Appendix C.

## Instrumentation

The benchmarking software HammerDB is the primary instrument that generated queries simulating an OLTP database that will create the results of the $2^k$ factorial experiment representing the $k = 4$ factors each at two levels tested and the operationalization of the instrument.

The primary instrument used for data collection in this study is HammerDB. HammerDB is an open-source database benchmarking tool used for benchmarking databases such as MySQL, Oracle, Microsoft SQL, MariaDB, DB2, PostgreSQL, and Redis (HammerDB, 2018a). Steve Shaw of Intel leads the HammerDB project team in the development of HammerDB (HammerDB Blog, n.d.). HammerDB applies the TPC-C benchmarking standard to the DBMS by creating a database schema specified by TPC-C, as well as adding sample data that is also defined by the TPC-C benchmarking standard. HammerDB allows the user to select the number of virtual users committing transactions against the database, and report the results as transactions per minute (TPM). To get the TPM, HammerDB applies the types of queries specified in the TPC-C standard, which simulate new orders placed on an OLTP database (TPC, 2010).

The response variable, throughput as measured by TPS, is measured using a ratio scale. The TPC-C specification defines a transaction as any business transaction that is successfully committed within the database and has the result reported back to the user (TPC, 2010). As supported in the literature review, throughput is the defining measure provided by the TPC-C standard, as established by the TPC.

Before starting a benchmarking test, the user must instruct HammerDB on the number of simulated users to create. In choosing the number of simulated users, I intend to have a significant difference between the high and low values to change the amount of workload on the database. On the low level, there should be enough users to see activity, so I decided on ten users for the low value. After some testing with HammerDB, I settled on 100 users for the high value. With 100 users, HammerDB takes quite a bit of time to

create the simulated uses as well as conducting the benchmarking test. Many more users may be time prohibitive, causing the benchmarking analysis to venture outside of the time window specified. Higher numbers of users would also incur more costs. The number of people is considered a discrete value. Still, with high and low values used in this factorial study, the number of simulated users will be ordinal in this experiment.

For the time factor, I manually initiated the tests between 10 a.m. and 11 a.m. Central Standard Time on a standard workday, Monday through Friday, representing the low value for the time factor in these experiments. The high value for the time factor will be between 10 p.m. and 11 p.m. Central Standard Time on a standard workday, Monday through Friday. During my time as a DBA for an enterprise organization, the usage patterns for most databases would peak later in the morning as most people came to work and began working. The usage would taper off during the standard lunch hours, and the usage pattern would have a second smaller peak in the midafternoon. I chose the window of 10 a.m. to 11 a.m. as the earlier window to mimic the usage peak I observed in an actual working environment. Similarly, I chose the latter window because this time frame was usually one of the least busy times for databases. While time is continuous by nature, in this factorial study, I will be using time as an ordinal data type.

The setting for InnoDB buffer pool size is a MySQL variable that I can set using MySQL commands. The values used for this variable are the low default value of 128 MB and the high recommend value of 80% of the VM's physical memory (Oracle Corporation, 2019a). The developers of MySQL version 8.0 have configured the DBMS to use 512 MB of system memory (Oracle Corporation, 2019b), but this value is not

much greater than the default value of 128 MB. I allocated 2 GB of memory for the VMs so that there may be sufficient difference between high and low values for the buffer pool size. Consequently, the high value for the InnoDB buffer pool size was 80% of 2 GB or 1.664 GB. To manually set the low value of the InnoDB buffer pool size, I issued the SQL command: `SET GLOBAL innodb_buffer_pool_size=134217728;` This value is 128 MB in bytes. For manually setting the high value of the InnoDB buffer pool size, I executed the following command in MySQL: `SET GLOBAL innodb_buffer_pool_size= 1744830464;` which is 1,664 MB in bytes. While the buffer pool size can be set as a discrete data type, I treated this variable as an ordinal type with high and low values only.

Similarly, for the InnoDB I/O capacity, I set the factor levels to the low default value of 200 input-output operations per second (IOPS) and the high value of 1,000 IOPS recommended for faster storage (Oracle Corporation, 2019a). I set the low value for the InnoDB I/O capacity using the MySQL command: `SET GLOBAL innodb_io_capacity=200;` and the SQL command to set the high value `SET GLOBAL innodb_io_capacity=1000.` As with the buffer pool size, in practice, the I/O capacity is discrete by nature but was treated as an ordinal data type in this study.

## Data Collection Technique

On each of the three public cloud providers, I provisioned identical VMs. The latest production version of Debian Linux was installed and fully updated on each of the VMs on the public cloud provider. To ensure an identical environment and database management system on all three VMs on each public cloud platform, a bash script named

*db_install.sh*, found in Appendix A, was uploaded and executed. This script first downloads the latest production version of MySQL Community Server (8.0) from the MySQL repositories by adding the MySQL repositories to the server. After the installation of MySQL, the script creates an empty database named *tpcc* that will eventually contain the schema for testing. After this, the script downloads and installs HammerDB on the same cloud-based VM and registers the appropriate libraries to allow HammerDB to interact with the MySQL Server. Once the script *db_install.sh* is completed, the VMs, MySQL instance, and HammerDB is prepared to run the tests for each factor combination on the cloud-based VM. This initial step only happened once on each of the three VMs before any benchmark tests take place.

For each test with differing factor combinations, I used HammerDB to execute a TLC script named *sqlrun.sh*, listed in Appendix B. This script is based heavily on the example provided by HammerDB (2018b) in the documentation for scripting in the command-line interface for HammerDB. The command-line script configures HammerDB to use the MySQL instance located on the local cloud-based virtual server using the standard MySQL port of 3,306. Next, the script directs HammerDB to implement the TPC-C benchmarking standard, which specifies the schema for the *tpcc* database and the types of queries that will run against the database. The script goes on to configure HammerDB to take 2 minutes to ramp up before HammerDB begins tracking the average data throughput in the implementation of the TPC-C benchmark. The script also specifies that the entire test lasts for 5 minutes, including the 2 minute ramp-up time. During the final 3 minutes of the testing, HammerDB averages the throughput in 10

second intervals and reports the final averaged throughput in the resulting logfile. The script also sets the details for the log file produced during testing that will contain the results, the dependent variable for these experiments, after each test. The script sets the option for HammerDB to give the log file a unique name so that no log file gets overwritten during testing. At this point in the second script, the database is an empty shell with no tables or data, and the script has configured HammerDB to execute the TPC-C benchmarking standard.

The script then commands HammerDB to generate the SQL code necessary to create the database schema as indicated by the loadscript command in Appendix B. Next, the build schema command found in Appendix B executes the SQL script which builds the schema in the specified database and loads sample data as dictated by the TPC-C benchmarking specification. Figure 2 shows the TPC-C specified schema that is built by HammerDB. HammerDB loads the data into the tables in the database that meet the TPC-C specification with random data that meets the data types specified for each field, and other constraints specified in the database such as foreign key relationships. For example, the TPC-C specification calls for the field W_CITY in the Warehouse table to have up to 20 variable characters (TPC, 2010), to which HammerDB may enter text like FKaak9ZBgtJr3Tr6gESW (HammerDB, 2018c). These text fields do not need to be human-readable or understandable, as the TPC-C specification is only testing the throughput simulating an OLTP database, and these transactions do not need to be humanly actionable.

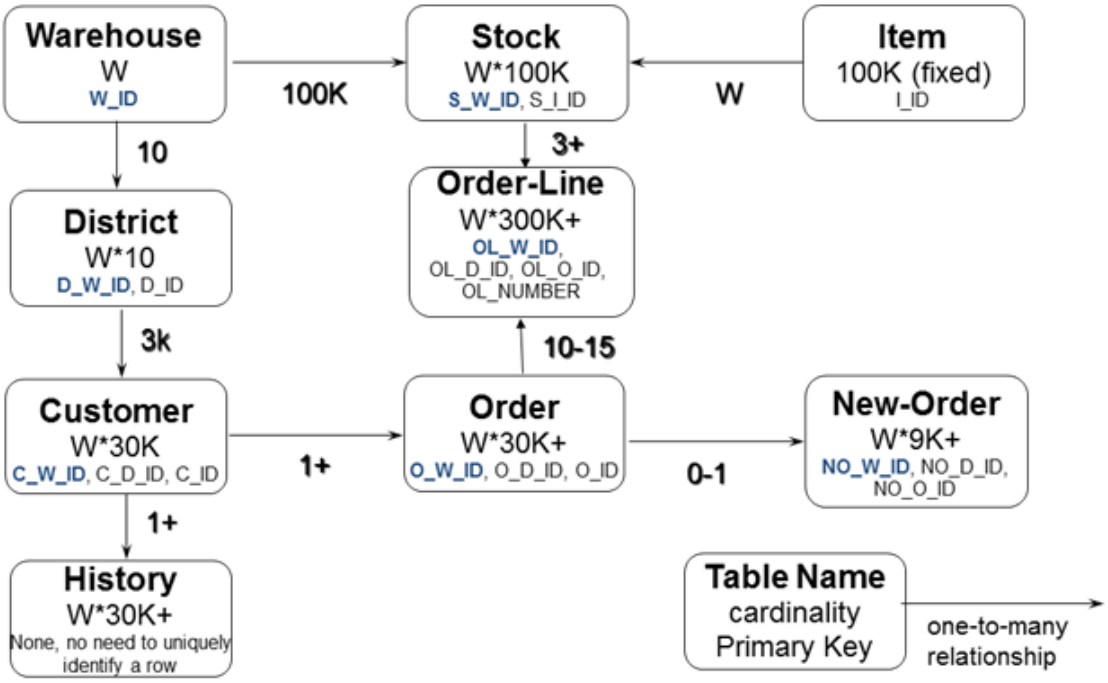


*Figure 2*. TPC-C Schema. From "Understanding the TPC-C workload," by HammerDB, 2018d (https://www.hammerdb.com/docs/ch03s05.html). Copyright 2018 by HammerDB. Reprinted with permission.

Depending on the factor level combination, the script instructs HammerDB to create either 10 or 100 virtual users for the database, with 10 virtual users being the low level and 100 virtual users the high level. HammerDB records the average number of TPM to a unique log file after each 5-minute test. It is the TPM listed at the end of the log file, shown in Appendix C, that is the data for the dependent variable in the analysis for this experiment.

On each of the three VMs, there were three replications of the experiment at each combination of factor levels. With $2^4 = 16$ factor level combinations, and three replications (NIST/SEMATECH, 2012a) of each combination on each VM will be 48 test runs on each VM.

I adjusted each of these factor levels until all combinations of high and low levels for all factors are tested, for a total of three replications on each cloud platform. One disadvantage of this technique is that it does require proper timing to ensure that the tests are performed within the allotted window, mainly since many of the factors must be manually applied. Another disadvantage is the fact that I had to be proficient on multiple cloud providers, and I had to manage accounts on each platform. One significant advantage of this testing process is that the operating system, the DBMS, and the testing instrument are all freely available open-source software with thorough documentation. I have cited the open-source documentation often throughout this document.

As the testing completes for each combination of factors, I downloaded the log files resulting from the 48 runs of the experiment and backed up the log files to multiple locations. This log file contains the time and date stamp of the test, the number of users simulated during the testing, and the average TPM reported by HammerDB. I have provided a sample of the output log from HammerDB in Appendix C, with the last line stating the TPM. I read each of these values from the result log files and manually enter the data into SPSS.

## Data Analysis Technique

As a reminder, the research question I am asking is: what are the optimal levels of the number of users, time of day, InnoDB buffer pool size, and InnoDB I/O capacity that will maximize throughput of MySQL, measured by TPS, running on a cloud-based VM. The hypothesis and null hypothesis for the main and interaction effects are:

- Main Effect Hypothesis:

$H_{0a}$: The main effect $F_i$ of factor *i* is not significant on the outcome.

$H_{1a}$: The main effect $F_i$ of factor *i* is significant in the outcome.

- Two-Factor Interaction Effects hypotheses:

$H_{0b}$: The interaction effect of $F_iF_j$ of the pair of factors *i* and *j* are not significant on the outcome.

$H_{1b}$: The interaction effect of $F_iF_j$ of the pair of factors *i* and *j* are significant on the outcome.

- Three-Factor Interaction Effects hypotheses:

$H_{0c}$: The interaction effect of $F_iF_jF_k$ of the triplet of factors *i*, *j*, and *k* is not significant on the outcome.

$H_{1c}$: The interaction effect of $F_iF_jF_k$ of the triplet of factors *i*, *j*, and *k* is significant on the outcome.

For repeatability, I replicated each experiment three times at each factor level combination, on each cloud platform, with the results for each cloud platform reported individually. In total, there are three cloud platforms with three replications each for the $2^4 = 16$ factor combinations with all experiments facilitated via the scripts and HammerDB. Table 1, located below, shows the design required to specify all feasible combinations of the high and low factor levels. For any missing data, I ran the experiment for the combination of factors needed again. If any results appeared to be significantly different from other similar results, I re-ran the benchmarking test to verify that the results.

Abramson et al., (2018) defined quasi-experimental design as one in which the researcher controls the independent variables, and analyzes the results. Since I changed each factor one at a time and measuring the results, this experiment meets the definition of quasi-experimental design. This study involves four of the many possible variables that may affect database throughput. A correlational analysis typically studies the strength of the relationships between two variables (Chen & Popovich, 2002), which would not suffice for the research questions in this study. Experimental reasearch involves a random assignment of treatment (Abramson et al., 2018), which was not done here. In the realm of factorial designs, an alternative would be a fractional factorial design, where a researcher uses a portion of the possible combinations of high and low factors in the experiment (Collins, Dziak, & Li, 2009). With only sixteen possible factor level combinations, it wasn't too much effort to perform all combinations of factor levels, and fractional factorial design is unnecessary. What follows is an outline of how I performed factorial ANOVA calculations in SPSS.

In Variable View in SPSS (Version 25), I added the four independent variables (time of day, number of simulated users, InnoDB buffer pool size, and InnoDB I/O

capacity) as shown in Figure 3.

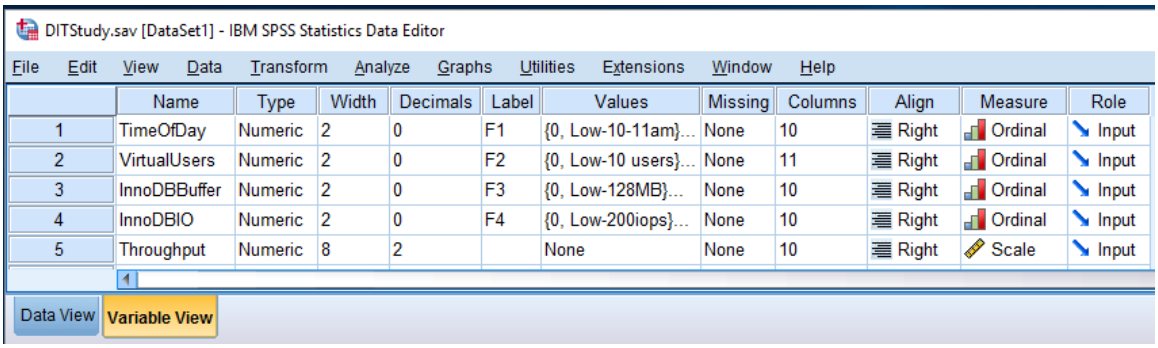


*Figure 3*. Variable view in SPSS with the independent and dependent variables.

The low level for each factor has a value of zero and the high level of each factor as one corresponding to the "-" or "+" signs in Table 1. I entered the dependent variable of throughput as the variable *Throughput*. Once I register the variables with SPSS, I added the data on the datasheet in SPSS. Sixteen different combinations of 0 and 1 for the independent variables and these 16 rows were repeated three times for each of the replications of the dependent variable from the three trials at each combination of factor levels.

Table 1

*Database Performance Factor Level Combinations*

| Run ID | Treatment Combination | Level of Factor | | | | Replicate | | |
|---|---|---|---|---|---|---|---|---|
| | | *a* | *b* | *c* | *d* | I | II | III |
| 1 | 1 | - | - | - | - | Y(1, A-, B-, C-, D-) | Y(2, A-, B-, C-, D-) | Y(3, A-, B-, C-, D-) |
| 2 | a | + | - | - | - | Y(1, A+, B-, C-,D-) | Y(2,A+, B- ,C-, D-) | Y(3, A+, B-, C-, D-) |
| 3 | b | - | + | - | - | Y(1,A-, B+, C-, D-) | Y(2, A-, B+, C-, D-) | Y(3, A-, B+, C-, D-) |
| 4 | ab | + | + | - | - | Y(1, A+,B+ ,C-, D-) | Y(2, A+,B+ ,C-, D-) | Y(3, A+,B+ ,C-, D-) |
| 5 | c | - | - | + | - | Y(1, A-, B-, C+, D-) | Y(2, A-, B-, C+, D-) | Y(3, A-, B-, C+, D-) |
| 6 | ac | + | - | + | - | Y(1, A+, B-, C+, D-) | Y(2, A+, B-, C+, D-) | Y(3, A+, B-, C+, D-) |
| 7 | bc | - | + | + | - | Y(1, A-, B+, C+, D-) | Y(2, A-, B+, C+, D-) | Y(3, A-, B+, C+, D-) |
| 8 | abc | + | + | + | - | Y(1, A+, B+, C+, D-) | Y(2, A+, B+, C+, D-) | Y(3, A+, B+, C+, D-) |
| 9 | d | - | - | - | + | Y(1, A-, B-, C-, D+) | Y(2, A-, B-, C-, D+) | Y(3, A-, B-, C-, D+) |
| 10 | ad | + | - | - | + | Y(1, A+, B-, C-, D+) | Y(2, A+, B-, C-, D+) | Y(3, A+, B-, C-, D+) |
| 11 | bd | - | + | - | + | Y(1, A-, B+, C-, D+) | Y(2, A-, B+, C-, D+) | Y(3, A-, B+, C-, D+) |
| 12 | abd | + | + | - | + | Y(1, A+, B+, C-, D+) | Y(2, A+, B+, C-, D+) | Y(3, A+, B+, C-, D+) |
| 13 | cd | - | - | + | + | Y(1, A-, B-, C+, D+) | Y(2, A-, B-, C+, D+) | Y(3, A-, B-, C+, D+) |
| 14 | acd | + | - | + | + | Y(1, A+, B-, C+, D+) | Y(2, A+, B-, C+, D+) | Y(3, A+, B-, C+, D+) |
| 15 | bcd | - | + | + | + | Y(1, A-, B+, C+, D+) | Y(2, A-, B+, C+, D+) | Y(3, A-, B+, C+, D+) |
| 16 | abcd | + | + | + | + | Y(1, A+, B+, C+, D+) | Y(1, A+, B+, C+, D+) | Y(1, A+, B+, C+, D+) |

*Note*: $a = F_1$ = Time of Day, $b = F_2$ = Number of simulated users, $c = F_3$ = InnoDB Buffer Pool Size, $d = F_4$ = InnoDB I/O Capacity

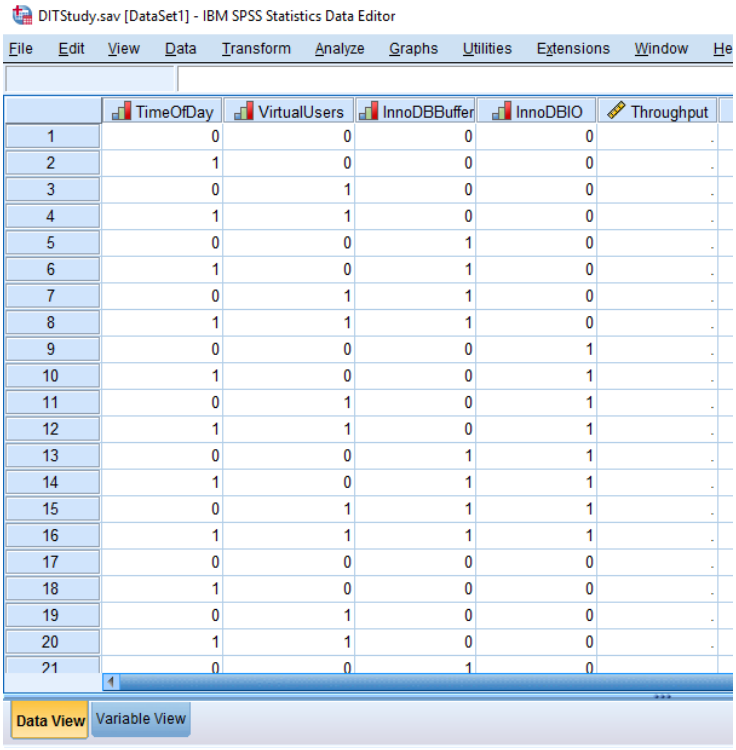


*Figure 4*. Data view from SPSS with factor combinations in Yates order.

In the Data View, the combination of factor levels were entered in the factor columns in Yates order, the same as Table 1, where the first factors alternate more frequently than the latter factors (NIST/SEMATECH, 2012b). Figure 4 shows the resulting data sheet before the execution of the experiments with the 16 factor-combinations possible, repeated three times for replication, for a total of 48 rows.

To initiate the analysis in SPSS, I chose Univariate from the GLM available under the Analyze menu. I added the four independent variables to the fixed factor(s) group, and the throughput was placed under the dependent variable, as shown in Figure 5.

The model for this experiment is full factorial, which was specified by clicking on the Model button shown in Figure 5, and ensuring that the correct combination of build terms was used for each combination of factors. Full factorial is the model defined, as

shown in Figure 6. After the full factorial model is confirmed, I clicked continue button seen in Figure 6, returning to the univariate dialog in Figure 5. At this point, I clicked the OK button in Figure 5 to begin the calculation of the results.

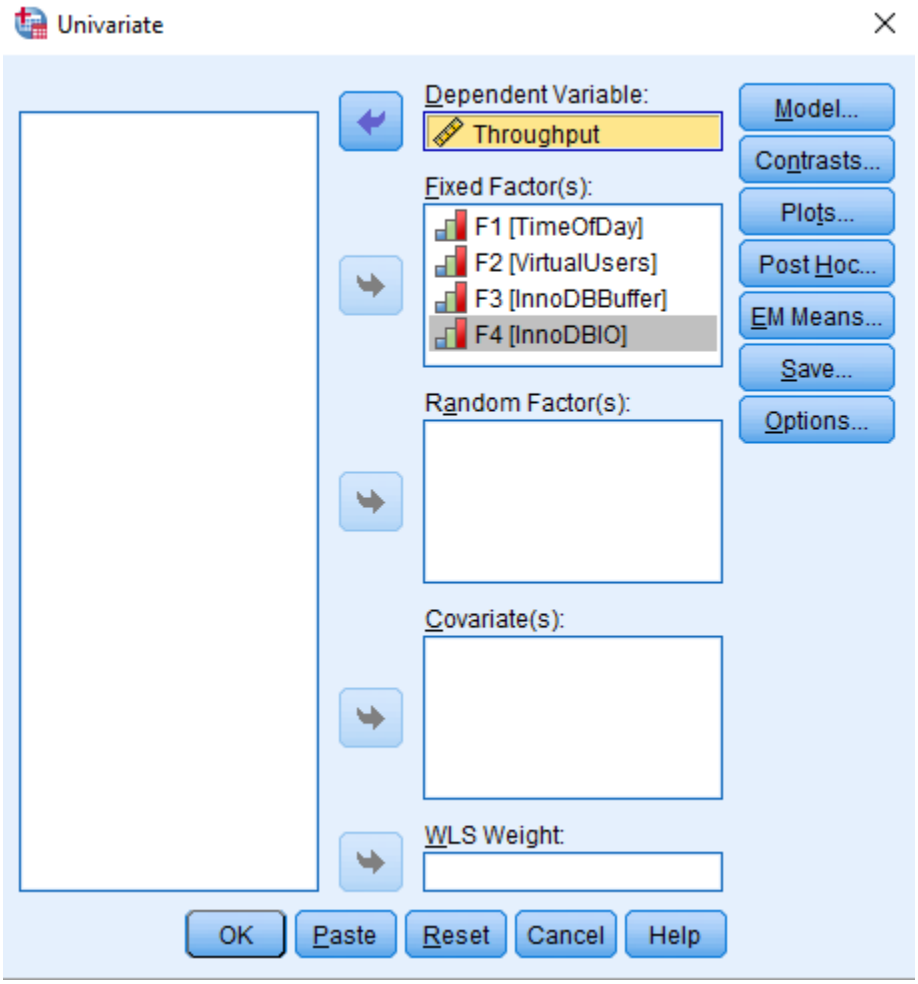


*Figure 5*. Populating the factors and dependent variables for analysis.

SPSS is a powerful tool, and running a full factorial ANOVA analysis with four factors is asking a lot. There are four factors in the SPSS model at all times. There are also six two-factor interaction terms possible four three-factor interactions possible, and a

single four-way interaction using all factors. These factor combinations can be found below in Table 2.

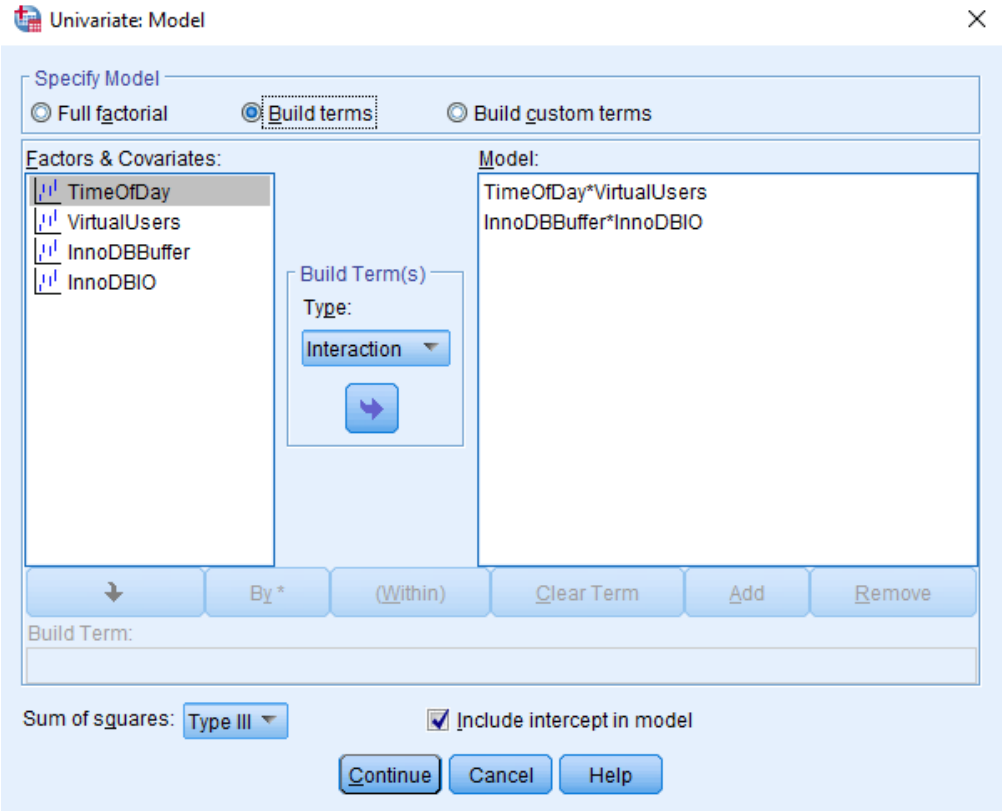


*Figure 6*. Specifying a full factorial model and interaction effects.

Due to professional experience and being informed by the literature, I do not expect many of the multi-factor interactions to produce significant results. In general, I am interested in the main effects (the significance of the coefficients of $F_1$, $F_2$, $F_3$, and $F_4$). Assuming that so-called "higher-order interactions" such as interactions of the second, third, and possibly fourth-order may be significant, a researcher may be tempted to include such higher-order interactions for SPSS to consider. The factor levels and interaction justifications are shown below in Table 2. However, such higher-order interactions are more difficult to interpret, tend to be less significant, and do not answer

the original research question and hypothesis. Thus, in this research, I chose only to include a subset of two-factor interactions, namely the combinations of time of day and number of users, and the combination of InnoDB buffer pool size and InnoDB I/O capacity. The selection of these specific factor pairs is based on my experienced-backed assumptions to expect a significant effect between $F_1$ representing the number of users and $F_2$ representing the time of day. From my professional observations, more users increased the demands on the DBMS. This demand would taper off near the end of the workday. The number of users being higher, and the overall increased demands of the workday suggest that increased users early in the day may show lower performance, and decreased users in the evening may experience higher throughputs. As suggested by the literature, an increase in the buffer pool size, represented by $F_3$, combined with an increase in the input-output to the DBMS, represented by $F_4$, should also result in higher throughput.

For each single and multi-factor combination selected, SPSS calculates the sum of squares using the GLM (IBM, 2017). GLM uses linear regression modeling variance involving a continuous dependent variable and categorical or discrete input variables representing the groups, or in this case, the combination of factor levels (Pennsylvania State University, 2018). In this case, I calculated a factorial ANOVA using GLM with four factors operating at two levels each. SPSS solves the GLM model in the form of the linear equation $y = b_0 + b_1F_1 + b_2F_2 + b_3F_3 + b_4F_4$, with $F_i$ representing the $i$th factor operating at two levels. Thus each $F_i$ is a binary instead of a continuous variable.

Table 2

*Justification of Factor Level Interactions*

| Interaction Type | Possible Factor Interactions | Relevant Interactions |
|---|---|---|
| 1-Factor interactions | $F_1$, $F_2$, $F_3$, $F_4$ | All single factor interactions are significant. |
| 2-Factor interactions | $F_1F_2$, $F_1F_3$, $F_1F_4$, $F_2F_3$, $F_2F_4$, $F_3F_4$ | The number of users and time of day ($F_1F_2$) interaction should give higher throughput with a lower load on the DBMS. The academic literature also suggests that a larger buffer pool and I/O pipeline ($F_3F_4$) should give improved throughput. |
| 3-Factor interactions | $F_1F_2F_3$, $F_1F_2F_4$, $F_1F_3F_4$, $F_2F_3F_4$ | Some combination of time of day, reduced numbers of users in conjunction with either a larger buffer pool ($F_1F_2F_3$) or I/O throughput ($F_1F_2F_4$) may give significant results, but only if one of the last factors has more of an effect than the other. |
| 4-Factor interactions | $F_1F_2F_3F_4$ | Not relevant. |

Models such as $F_1F_2$ which would consider the interactions of the first two factors, or $F_3F_4$, modeling the interactions between the third and fourth factors, and use more complicated GLM models of the form $y = b_0 + b_1 F_1 + b_2 F_2 + b_3 F_3 + b_4 F_4 + b_5 F_1 F_2 + b_6 F_3 F_4$. Continuing in this line of examples, models considering interactions of three factors such as $F_1F_0F_3$ may take an even more compliated GLM form such as $y = b_0 + b_1 F_1 + b_2 F_2 + b_3 F_3 + b_5 F_1 F_2 F_3$. Once SPSS calculates the sum of squares for that factor or factor combination, SPSS calculates the mean square for that factor or factor combination by dividing the sum of squares by the degrees of freedom for that factor combination. This final calculation gives us the *F* score for the factor combination (Sajid, 2016). The *F* score indicates if a combination of factors has a statistically

significant effect on the outcome (Glen, 2013). If this *F* score is significantly higher than the critical *F* value, then I can reject the null hypothesis, which in this case, would be that the factor or factor combination does not have a significant effect on the outcome. SPSS calculates the critical *F* value by a combination of the degrees of freedom for single factor or factor combination, the degrees of freedom for the entire experiment, and the desired alpha value for the experiment, in this case, 0.05 (Sajid, 2016). The *p* value is the probability that the *F* score is not due to randomness. If the *p* value is below the significance level of 0.05, then I have further confirmation that I can reject the null hypothesis (Glen, 2014).

For each factor and factor combination test where the *p* value, displayed as Sig. in the SPSS output, where the significance is below 0.05 for any factor or combination of factors, I rejectred the null hypothesis for that particular factor or combination of factors. This data analysis procedure was repeated and reported separately for each public cloud provider.

In the event of a significant result, researchers often perform a post hoc analysis to determine if one of the groups differs from one of the other groups. For this analysis, I also included a post hoc analysis using the estimated marginal means. The estimated marginal means calculates the marginal means for each factor adjusted for the other variables in the GLM, and in the case of a significant interaction, tells us if one of the two or more factors is still significant (Grace-Martin, 2019). SPSS can perform the calculations for the estimated marginal means by clicking on the EM Means button shown in Figure 5, which brings up the factors available for inclusion, as seen below in

Figure 7. I included any factor combination that had significant results. Clicking the continue button brings the user back to Figure 4 so that the user can begin the analysis chosen for the data.

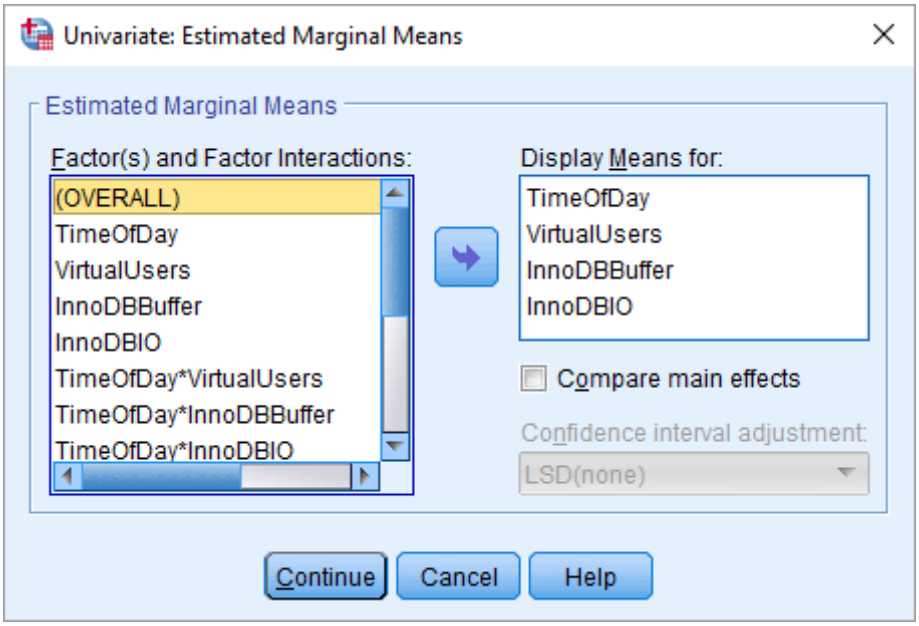


*Figure 7*. Estimated marginal means for SPSS.

When conducting a factorial ANOVA, I am making four basic assumptions about the data: variables are of the correct scale, the dependent variable data is normally distributed, the dependent data has the same variance, and the data is independent (Statistics Solutions, n.d.). In this experiment, the dependent variable is a ratio, as measured in megabytes per second, and the independent variables, being high and low values, are nominal. The correct scales for the independent and dependent variables required for ANOVA are inherent in the experiment, so this assumption is satisfied (Statistics Solutions, n.d.). Using SPSS, I confirmed that the dependent variable is normal by running a Kolmogorov-Smirnov (K-S) test for normality. The null hypothesis

of the K-S test is that the dependent data is normally distributed (Van den Berg, 2020). If the significance of the K-S test is less than 0.05, then I reject the null hypothesis, and therefore, the data is not considered normally distributed (Van den Berg, 2020). If the data is not normally distributed, I applied a log transformation to the dependent data, which changes the data to an index that will meet the assumption (Statistics Solutions, n.d.). The third assumption is that the dependent data has the same error variances, otherwise known as homogeneity of variance. Levene's Test of Homogeneity of Variance is one test that can be used to meet this assumption. Levene's Test is performed on the dependent variable over each of the four factors. A significance above 0.05 indicates that we can reject the null hypothesis that the variances are not equal (NIST/SEMATEC, 2012c). If $p < 0.05$, then we can accept the null hypothesis that the variances are not equal. However, the ANVOA test is considered robust and unequal variances shouldn't violate the underlying assumptions of ANOVA, particularly if the sample sizes are equal for all factor levels (Pennsylvania State University, 2020). In these experiments, $n = 8$ for each factor level, so the sample sizes are balanced. For the last point, the easiest way to avoid violating the assumption of independence is to ensure that the throughput is not measured too closely in time (McDonald, 2014). By avoiding repeated runs of the benchmarking test with the same factor combinations, and not performing more than measurement at the same time, I can assure that the observations are independent of each other, and not affected by other factors such as query results remaining in the database cache.

## Validity

Validity is critical for the quality of a measure and indicates that the interpretations of the results of a test are reliable (Kimberlin & Winterstein, 2008). As discussed in the literature review, each of the independent variables chosen for this experiment affects database performance in some way. Yu and Pradel (2018) spoke to the importance of the InnoDB engine in controlling the flow of data between the database and the disks. Other articles supported the importance of buffer pools and disk I/O in database performance (Kong, 2012; Lee, 2014). Tajbakhsh et al. (2017) showed that more virtual users lead to less throughput. Other studies introduced various methods to handle different workloads in a multi-tenant environment (Henneberger, 2016; Tseng et al., 2015). These studies support the inclusion of the independent variable of time of day for this study, to find potential differences in and out of the standard workday in the United States. Since higher processing power and more memory can positively affect database performance (Hwang et al., 2016), these factors will be controlled by provisioning VMs with identical specifications across all public cloud providers. It is with the supporting literature that I have attempted to address the internal validity of the variables as well as the interaction of external variables that may affect the outcome measured.

A Type I error occurs when a researcher incorrectly concludes that the null hypothesis when they should not (Sedgwick, 2014). The smaller the acceptable level of error gives a lower chance of encountering a Type I error. The most common level acceptable is 5%, or $p < 0.05$, although the researcher can fix this value at any level

(Smith, 2012). For these calculations, I followed the literature in using $p < 0.05$ as acceptable in avoiding a Type I error. Working to avoid a Type I error should yield a valid conclusion and avoid irrelevant results (Jackson & Brashers, 1994).

Conversely, a Type II error occurs when a researcher fails to reject a false null hypothesis (Oehlert, 2010, p. 150). The higher the power of a statistical test will reduce the chances of a Type II error (Faul et al., 2007). As explained in the Population and Sampling section, I used 48 samples, which will give a power of $1 - \beta = 0.9$. Typical acceptable rates of 0.80 are the minimum generally accepted level for power (Dien, 2017).

The key instrument in measuring the dependent variable used in this experiment is the TPC-C benchmarking standard. It was after an extensive review of the literature that I chose this benchmarking standard for this study (Ferretti, Colajanni, et al., 2014; Loghin et al., 2015; Tian et al., 2018). This instrument also has the added benefit of being able to alter the number of simulated users, which is one of the factors selected for an independent variable. As mentioned above, I controlled other external variables, such as hardware, that are likely to affect the dependent variable by running the experiments on identically provisioned VMs so that resources, so that hardware wasn't a factor in this experiment. I performed the benchmarking tests on the same VM as the database, so any delays in networking did not affect the outcomes because the test data was only transmitted locally within the VM, and not over a network. Due to the extensive support of the TPC-C benchmark in the literature and HammerDB's implementation of the TPC-

C standard in addition to controlling one of the factors for these experiments makes HammerDB a valid benchmark for this test.

## Transition and Summary

In this section, I have outlined the role of the researcher and described the participants for this study. After justifying the research methods and designs, I reviewed and supported the population for the study, and how I ethically completed this research. In the discussion of the data collection, I have outlined and supported the instruments and exact processes that I used to collect and analyze the data. This section concludes with a discussion of the validity of the data analysis process and the instruments chosen for this experiment. In the following sections, I will describe the outcomes of the study and how the experiments apply to professional practice and promote social change.

Section 3: Application to Professional Practice and Implications for Change

On the big three public cloud providers, Microsoft Azure, AWS, and Google Cloud Platform, I created three equally provisioned VMs with Debian 10 as the operating system. I have included complete specifications for the VMs in Appendix E. Using a script, I installed Version 8.0 of the MySQL database management system on each of the servers. I used the benchmarking software HammerDB to implement the TPC-C benchmarking standard on each of the VMs. I ran each benchmark test at combinations of high and low levels for the factors of time of day, the number of virtual users, InnoDB buffer pool size, and InnoDB I/O capacity for a total of 16 different combinations of trials ran on each VM. For replication, I ran each factor combination trail three times for a total of 48 trails on each virtual server.

What follows is the outcomes of the experiments, the statistical analysis, and the overall findings. I will support these findings from the peer-reviewed literature and explain how the results fit into the theoretical framework of Six Sigma. I also discuss how this study applies to professional practice and its implications for social change. This section concludes with recommendations for future actions and research.

## Overview of Study

The purpose of this quantitative, quasi-experimental study was to evaluate the relationship between the time of day, the number of concurrent users, InnoDB buffer pool size, InnoDB I/O capacity, and transaction throughput to a MySQL database running on a cloud, virtual, database server. I could not find any statistically significant results on any of the cloud providers at any factor levels.

I also noticed significant differences in throughput and costs on each of the cloud providers. While finding the optimal factor combination for throughput was the focus of this study, AWS consistently provided the fastest throughput at all factor levels compared to the other two cloud providers. Google was slightly slower than AWS but more inconsistent with results, occasionally having throughput at one third of the average speeds. Microsoft Azure was consistently slower than the other two cloud providers, with performance averaging just over one 10th of AWS results. For the total costs of these experiments, Google's were $0.80, AWS costs were $1.19, while Microsoft's total costs were $9.35.

## Presentation of the Findings

For this study, I performed a full-factorial ANOVA analysis in SPSS. The explanation of how SPSS calculates the sum of squares using the GLM to determine an *F* score for each factor and factor combination can be found on p. 76. As shown in Figure 4, the high values of each factor are represented by 1 and the low values are represented by a zero. The low value for the time of day was between 10 a.m. and 11 a.m., and the high value was 10 p.m. to 11 p.m. Ten virtual users were the low value for virtual users, and 100 virtual users were the top value. I used the default value of 200 IOPS for InnoDB I/O capacity for the low value, and 1,000 IOPS for the high value because it was Oracle's recommended value for faster storage (Oracle Corporation, 2019a). Finally, I used the default value of 134,217,728 bytes (i.e., 128 MB) for the low setting of InnoDB buffer pool size. I used 80% of the available memory; Oracle's recommended highest value (Oracle Corporation, 2019a). For these experiments, I provisioned the VMs with 2

GB of memory each, so I used 1,664 MB or 1,744,830,464 bytes in the MySQL configuration.

For repeatability, I ran each test three times at each factor level combination on each cloud platform. With each test, I ran a script that recorded the date, time, InnoDB buffer pool size, and InnoDB I/O capacity into a text file. Each time I ran the benchmarking test, I configured the benchmarking software to use a unique file name. The benchmarking software recorded the time of day, the number of virtual users, and the throughput measured in TPM. I recorded the throughput in TPM in a spreadsheet for all three trials, averaged the results, and divided the average by 60, which gave me TPS. I recorded this final calculation in SPSS in the empty column labeled Throughput in Figure 4. I have included the data used for the calculations in Appendices F, G, and H.

Overall, there were differences in each cloud provider's mean values, as shown in Table 3. There were no significant outliers for any of the cloud platforms under any factor combinations; however, there was a distinct difference in the throughput on the Google Cloud Platform, most likely stemming from two of the primary factor levels, as discussed further later in this section. There were no missing values from any of the trials.

Table 3

*Descriptive Statistics for Throughput on Each Cloud Provider*

| | *N* | Minimum | Maximum | *M* | *SD* |
|---|---|---|---|---|---|
| Azure | 16 | 37.86 | 44.39 | 42.2594 | 1.84825 |
| Amazon | 16 | 296.57 | 329.35 | 314.1912 | 8.47406 |
| Google | 16 | 96.19 | 263.66 | 178.7033 | 45.68022 |

As discussed in the Data Analysis Technique section, most of the ANOVA testing assumptions are inherent in the experiments. The independent variables are all nominal in practice, either high or low in value, and the dependent variable is ratio in scale, all of which are an assumption of ANOVA (Statistics Solutions, n.d.). In running the experiments, I met the assumption of independence by ensuring that I did not run the same combination of factor levels sequentially (see McDonald, 2014), verified by the time and date stamps on the output logs. Another assumption in ANOVA testing is that the dependent variable approximates a normal curve (Statistics Solutions, n.d.). One way to test for normality is to use the K-S test (Van den Berg, 2020). If the significance of the K-S test is $p < 0.05$, it can be assumed that the data are significantly different from a normal distribution (Van den Berg, 2020). For AWS, K-S indicated that the data are not significantly different from normally distributed data, with $D(16) = 0.115$, $p = 0.200$. For Microsoft Azure, $D(16) = 0.192$, $p = 0.119$, indicating that this set of throughput data are not significantly different from normally distributed data. And the K-S test for Google Cloud showed $D(16) = 0.202$, $p = 0.080$, indicating that the throughput data for Google Cloud are not significantly different from normally distributed data.

The final assumption for ANOVA testing is that the data has the same error variances or homogeneity of variance (Statistics Solutions, n.d.). One test to show homogeneity of variance is to use the Levene's Test of Homogeneity of Variance. This test is performed on the dependent variable over each of the four factors. A significance above 0.05 ($p > 0.05$) means that it can be concluded that the variances are equal (NIST/SEMATECH, 2012c). In performing this test across all factors on all platforms, I found that SPSS was unable to calculate Levene's statistic due to lack of degrees of freedom. With only a single sample for each cloud provider, degrees of freedom = $k - 1$, with k representing the number of factors. In this case, DF = 1 – 1 = 0, lacking the degrees of freedom necessary to calculate Levene's statistic. However, I can ignore this assumption because the ANOVA test is robust, particularly with balanced sample sizes for all factors (see Pennsylvania State University, 2020).

After completing the calculations, I found that there were no significant results. In all cases, $p > 0.05$, showing that none of the main factor or factor combinations had a significant effect on the throughput. Only significant factors should be used for the GLM model to show the relationship between the significant factors and the dependent variable (Šoltés, Zelinová, & Bilíková, 2019). Without significant factors, there is no GLM model to create.

**Main Effect Hypotheses**

$H_0$a: The main effect $F_i$ of factor i is not significant on the outcome.

$H_1$a: The main effect $F_i$ of factor i is significant in the outcome.

**Amazon Web Services.** As previously stated in the Population and Sampling section, the most common level for avoiding a Type I error is $\alpha = 0.05$ (see Smith, 2012). In using this level, I found no main factors where $p > 0.05$, so I failed to reject the null hypothesis and conclude that none of the main effects significantly affect the throughput. The results for the main factor effects on AWS can be found in Table 4.

Table 4

*Statistical Analysis Results for Main Factor Effects on AWS*

| Factor | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| Time of day | 319.87323 | 1 | 319.87323 | 17.6262 | 0.149 | 0.946 |
| Virtual users | 421.48090 | 1 | 421.48090 | 23.2252 | 0.130 | 0.959 |
| InnoDB buffer | 81.72160 | 1 | 81.72160 | 4.5032 | 0.280 | 0.818 |
| InnoDB I/O | 46.78560 | 1 | 46.78560 | 2.5781 | 0.355 | 0.721 |

**Microsoft Azure.** None of the main effects on Microsoft Azure were found to have statistical significance, as shown in Table 5. Therefore, I failed to reject the null hypothesis that the main factors do not significantly affect the throughput of the database.

Table 5

*Statistical Analysis Results for Main Factor Effects on Microsoft Azure*

| Factor | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| Time of day | 2.53606 | 1 | 2.53606 | 1.62964 | 0.330 | 0.449 |
| Virtual users | 0.32206 | 1 | 0.32206 | 0.20695 | 0.694 | 0.094 |
| InnoDB buffer | 12.33766 | 1 | 12.33766 | 7.92803 | 0.106 | 0.799 |
| InnoDB I/O | 6.77301 | 1 | 6.77301 | 4.35225 | 0.172 | 0.685 |

**Google Cloud Platform.** On Google Cloud, as with the other platforms, none of the main factor effects achieved $p > 0.05$, meaning that I failed to reject the null hypothesis that the main factors do not significantly affect the throughput on Google Cloud. The results of ANOVA testing can be found in Table 6.

Table 6

*Statistical Analysis Results for Main Factor Effects on Google Cloud*

| Factor | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| Time of day | 1182.844 | 1 | 1182.844 | 0.42521 | 0.632 | 0.298 |
| Virtual users | 5484.513 | 1 | 5484.513 | 1.97159 | 0.394 | 0.663 |
| InnoDB buffer | 1291.504 | 1 | 1291.504 | 0.46427 | 0.619 | 0.317 |
| InnoDB I/O | 14953.010 | 1 | 14953.010 | 5.37536 | 0.259 | 0.843 |

**Two-Factor Interaction Effects Hypotheses**

$H_0$b: The interaction effect of $F_iF_j$ of the pair of factors *i* and *j* are not significant on the outcome.

$H_1$b: The interaction effect of $F_iF_j$ of the pair of factors *i* and *j* is significant on the outcome.

**Amazon Web Services.** None of the combinations of two factors showed significant interactions on AWS, as shown in Table 7. I failed to reject the null hypothesis that any pair of factors have significant interaction effects on the throughput of the database.

Table 7

*Statistical Analysis Results for Two-Factor Effects on AWS*

| 2-Factor Combination | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| TimeofDay ✻ VirtualUsers | 0.65610 | 1 | 0.65610 | 0.0362 | 0.880 | 0.035 |
| TimeofDay ✻ InnoDBBuffer | 56.85160 | 1 | 56.85160 | 3.1327 | 0.327 | 0.758 |
| VirtualUsers ✻ InnoDBBuffer | 20.93062 | 1 | 20.93062 | 1.1534 | 0.477 | 0.536 |
| TimeofDay ✻ InnoDBIO | 4.24360 | 1 | 4.24360 | 0.2338 | 0.713 | 0.190 |
| VirtualUsers ✻ InnoDBIO | 1.55002 | 1 | 1.55002 | 0.0854 | 0.819 | 0.079 |
| InnoDBBuffer ✻ InnoDBIO | 59.98502 | 1 | 59.98502 | 3.3054 | 0.320 | 0.768 |

**Microsoft Azure.** One two-way interaction effect, InnoDB I/O capacity and InnoDB buffer pool size, came close to showing significance with $F(1,11) = 12.51130$, $p = 0.071$, $\eta_p^2 = 0.862$, as shown in Table 8; however, this combination still failed to fall below the significance level of $p < 0.05$. The result here is that I failed to reject the null hypothesis and conclude that none of the two-factor combinations have a significant effect on the throughput of the database.

Table 8

*Statistical Analysis Result for Two-Factor Effects on Microsoft Azure*

| 2-Factor Combination | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| TimeofDay ✻ VirtualUsers | 1.48231 | 1 | 1.48231 | 0.95251 | 0.432 | 0.323 |
| TimeofDay ✻ InnoDBBuffer | 0.94576 | 1 | 0.94576 | 0.60773 | 0.517 | 0.233 |
| VirtualUsers ✻ InnoDBBuffer | 0.07981 | 1 | 0.07981 | 0.05128 | 0.842 | 0.025 |
| TimeofDay ✻ InnoDBIO | 2.22756 | 1 | 2.22756 | 1.43140 | 0.354 | 0.417 |
| VirtualUsers ✻ InnoDBIO | 0.20476 | 1 | 0.20476 | 0.13157 | 0.752 | 0.062 |
| InnoDBBuffer ✻ InnoDBIO | 19.47016 | 1 | 19.47016 | 12.51130 | 0.071 | 0.862 |

**Google Cloud Platform.** As with AWS and Azure, none of the combinations of two factors showed significant interaction in the Google Cloud Platform, as displayed in Table 9. Therefore, I failed to reject the null hypothesis that any pair of factors have a significant interaction effects on the throughput.

Table 9

*Statistical Analysis Results for Two-Factor Effects on Google Cloud*

| 2-Factor Combination | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| TimeofDay ✻ VirtualUsers | 2172.259 | 1 | 2172.259 | 0.78089 | 0.539 | 0.438 |
| TimeofDay ✻ InnoDBBuffer | 0.107 | 1 | 0.107 | 3.86e-5 | 0.996 | 0.000 |
| VirtualUsers ✻ InnoDBBuffer | 11.577 | 1 | 11.577 | 0.00416 | 0.959 | 0.004 |
| TimeofDay ✻ InnoDBIO | 2.038 | 1 | 2.038 | 7.33e-4 | 0.983 | 0.001 |
| VirtualUsers ✻ InnoDBIO | 1875.107 | 1 | 1875.107 | 0.67407 | 0.562 | 0.403 |
| InnoDBBuffer ✻ InnoDBIO | 519.726 | 1 | 519.726 | 0.18683 | 0.740 | 0.157 |

**Three-Factor Interaction Effects Hypothesis**

$H_0$c: The interaction effect of $F_iF_jF_k$ of the triplet of factors *i*, *j*, and *k* is not significant on the outcome.

$H_1$c: The interaction effect of $F_iF_jF_k$ of the triplet of factors *i*, *j*, and *k* is significant on the outcome.

No significant interaction effects were found for all three public cloud providers for any combination of three factors. Consequently, I failed to reject the null hypotheses and conclude that any combination of the three factors will not significantly affect throughput on any of the three cloud providers. Results for three-way interaction effects for AWS can be seen in Table 10, Microsoft Azure results are listed in Table 11, and Google Cloud Platform results are listed in Table 12.

Table 10

*Statistical Analysis Results for Three-Factor Effects on AWS*

| 3-Factor Combination | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| TimeofDay ✻ VirtualUsers ✻ InnoDBBuffer | 24.45302 | 1 | 24.45302 | 0.78089 | 1.3475 | 0.574 |
| TimeofDay ✻ VirtualUsers ✻ InnoDBIO | 0.00302 | 1 | 0.00302 | 1.67e-4 | 0.992 | 0.000 |
| TimeofDay ✻ InnoDBBuffer ✻ InnoDBIO | 11.93702 | 1 | 11.93702 | 0.00416 | 0.6578 | 0.397 |
| VirtualUsers ✻ InnoDBBuffer ✻ InnoDBIO | 8.52640 | 1 | 8.52640 | 7.33e-4 | 0.4698 | 0.320 |

Table 11

*Statistical Analysis Results for Three-Factor Effects on Microsoft Azure*

| 3-Factor Combination | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| TimeofDay ✻ VirtualUsers ✻ InnoDBBuffer | 2.89851 | 1 | 2.89851 | 13.55036 | 0.169 | 0.931 |
| TimeofDay ✻ VirtualUsers ✻ InnoDBIO | 1.08681 | 1 | 1.08681 | 5.08076 | 0.266 | 0.836 |
| TimeofDay ✻ InnoDBBuffer ✻ InnoDBIO | 0.66016 | 1 | 0.66016 | 3.08619 | 0.329 | 0.755 |
| VirtualUsers ✻ InnoDBBuffer ✻ InnoDBIO | 0.00181 | 1 | 0.00181 | 0.00844 | 0.942 | 0.008 |

Table 12

*Statistical Analysis Results for Three-Factor Effects on Google Cloud*

| 3-Factor Combination | Sum of Squares | *df* | Mean Square | *F* | *p* | $\eta^2_p$ |
|---|---|---|---|---|---|---|
| TimeofDay ✻ VirtualUsers ✻ InnoDBBuffer | 64.762 | 1 | 64.762 | 0.02328 | 0.904 | 0.023 |
| TimeofDay ✻ VirtualUsers ✻ InnoDBIO | 265.120 | 1 | 265.120 | 0.09531 | 0.809 | 0.087 |
| TimeofDay ✻ InnoDBBuffer ✻ InnoDBIO | 621.879 | 1 | 621.879 | 0.22356 | 0.719 | 0.183 |
| VirtualUsers ✻ InnoDBBuffer ✻ InnoDBIO | 73.917 | 1 | 73.917 | 0.02657 | 0.897 | 0.026 |

**Research Question Answer**

The final answer to the research question on what is the optimal levels of time of day, number of users, InnoDB buffer pool size, and InnoDB I/O capacity maximizes the throughput of MySQL running on a cloud server is that none of these factors, individually or in combination, have a significant effect. The biggest factor that seemed to affect throughput was the cloud provider, and by extension, the underlying hypervisor. Amazon uses the Xen hypervisor (Badola, 2019), Microsoft uses their product Hyper-V as the underlying hypervisor for Azure (Microsoft, 2019), and Google's Cloud is supported by the KVM hypervisor (Honing & Porter, 2017). The underlying hypervisor software may have more to do with the performance differences than the individual companies themselves.

**Amazon Web Services.** Of the three cloud providers, AWS consistently gave the highest throughput at all factor level combinations. One of the initial assumptions was that the three cloud providers would perform about the same, and these experiments have shown that this was an incorrect assumption. While Google came close in performance overall, Google was more inconsistent with the throughput. Of all the factors, the number of users and time of day was the most influential factors, but these factors still did not reach significance. I had expected these two factors to have more value based on my experiences. From my observations as a DBA, I would have thought there to be a synergy between a few users and less overall system usage at night.

**Microsoft Azure.** Since the throughput for MySQL running on a Linux VM on Microsoft Azure was the lowest, on average, nearly twice as slow as AWS, I would not recommend any DBA use Microsoft Azure in the manner used in this study. The analysis of the data shows that no factor or factor combination will provide optimal throughput. However, tweaking the InnoDB buffer pool size and InnoDB I/O capacity may improve performance since this combination had the most significant levels, although not statistically significant. The poor performance of MySQL running on a VM on Microsoft Azure was suggested by Ahmed (2013), who found that MySQL running on Hyper-V suffered slower response times than running on a physical computer. Chung and Nah (2017) found similar results on the Xen hypervisor, but their results were not reflected in AWS's results as profoundly as in Microsoft Azure.

**Google Cloud.** As with the other cloud providers, Google Cloud showed no significant results. Google had the least significant results overall at all factor levels. I

expected InnoDB I/O capacity to be a significant factor when I observed consistent differences on this factor level when all other factors were equal. Disk I/O was also one of the biggest performance bottlenecks for databases in multitenant cloud databases (Xavier et al., 2016). When I set the InnoDB I/O capacity to a lower value, the throughput was generally higher and more consistent. Since I/O is recognized as a bottleneck, I casually observed better performance by sending shorter bursts of information to the disks.

Unexpected with Google was the inconsistency with results. In 18 out of the 48 trials performed on Google Cloud, I saw significantly lower throughput. The lower results were slightly higher than Azure, but roughly one third of the speed was found in the other 30 trials. In the other 30 trials, the throughput on Google was marginally lower than Amazon. In their experiments, Reddy and Shyamala (2016) also found that KVM used slightly more memory and processing power than Xen, which may explain the differences between AWS and Google.

**Theoretical famework.** For these experiments, Six Sigma provided a sound framework to test the different factors. One unexpected outcome was that I would find that none of the identified factors seemed to make a significant difference in throughput, but that the cloud providers themselves appeared to have the most significant difference. One goal of Six Sigma is to reduce variation in a system (LeMahieu et al., 2017). It is easiest to see the reduction in a variation on the Google Cloud Platform. Even though I found no factor to be significant, the InnoDB I/O capacity was the factor that seemed to

have the most significant difference. The throughput changes are more subtle on the other two cloud providers and not as easily observed.

Another benefit provided by Six Sigma was the reduced costs of the overall experiment. I ran the same number of trails on each cloud provider and shut down the VMs between testing windows to keep costs down. Both AWS and Google costs remained less than $1.20 US total. Microsoft's expenses for the same number of trails was over $9.00 US. If I had run ongoing trials, observing throughput by the second for an extended time, all providers' costs would have been significantly higher since cloud providers charge by the amount of processing power. By selecting the time of day as a factor and only using processing power for 2 hours each weekday, Six Sigma helped keep the cost of testing to a minimum.

## Applications to Professional Practice

The most significant application to professional practice is the performance of each of the cloud providers. This fact is contrary to my initial assumptions that each of the cloud providers, and the underlying hypervisors, would perform similarly under identical circumstances. Averaging the trials on each of the cloud providers, AWS averaged 314 TPS, Google averaged 179 TPS, and Microsoft averaged only 42 TPS. Looking at the optimal trials implemented on Google, which occasionally had low results, a DBA could see averages above 200 TPS. This result indicates that if a DBA wishes to run MySQL on a VM and get better overall performance, their best option would be to use AWS. Implementing the optimal factors of fewer users and running more intensive queries at night, a DBA could achieve the best performance given these

factors. The experiments also indicate that running MySQL on a Linux VM is not the best use case on Microsoft Azure's platform, and DBAs should consider other approaches for processing MySQL data if a company is locked into the Azure framework.

From my personal experiences as a DBA, the factor of the time of day is most likely evident to many DBA. Most of the significant ETL jobs were scheduled at night to avoid contention with user queries and optimal throughput. My experiments have failed to confirm that this practice will yield optimal performance, allowing the ETL jobs to process more quickly.

## Implications for Social Change

As mentioned in the previous section, there are substantial cost and performance implications for nonprofit organizations. First, if a nonprofit organization plans to run MySQL on a Linux VM, Microsoft's Azure platform would not be the best cloud provider for this kind of architecture. Not only from a performance consideration but cost considerations as well. From a cost perspective, there is not a significant difference between Google and AWS. Still, a nonprofit will likely see the best combination of steady performance and lower costs if they run a Linux VM hosting MySQL. From my experiments, it would also be helpful if such an organization runs any major ETL jobs at night, as my results have shown that database performance is better at night and with fewer users on AWS. A nonprofit can process more data more efficiently and at a lower cost allowing the organization to dedicate its dollars and processing power to help more people.

## Recommendations for Action

Based on these experiments' results, any DBA who is running MySQL on a VM on Microsoft Azure should consider migrating to another cloud platform or use a DBMS that functions better on the Azure platform. As suggested in the research reported by Almeida et al. (2015), a DBA may want to consider running Microsoft's SQL server instead of using Azure's SQL database-as-a-service.

The second recommendation for DBA would be to run intensive queries at night or when there are fewer users, particularly if they are running MySQL on a Linux VM on AWS or Google. The results of my experiments show that these two factors will help improve throughput. Finally, I recommend keeping the InnoDB I/O capacity at the default value. The InnoDB I/O capacity was shown to have significant results on Google, and there were observable improvements in throughput on AWS. However, the results on AWS were not found to be significant.

In the short term, I intend to present these results to a local developer's group, where I have presented in the past. The steering committee for that group has expressed interest in hearing the results. Since I currently teach a database course at a small university, I intend to include my findings as part of the course. If I were to attempt to publish these results, I might consider the journal *Proceedings of Very Large Databases* (PVLDB). Several articles were published by this journal cited in this paper, and I found many more articles of interest published by the PVLDB.

## Recommendations for Further Study

When I started down this research path, AWS had just begun a relational database as a service. Since then, all three public cloud providers have multiple types of databases as a service. I hope to follow one research path to determine any significant differences in performance between the relational database as a service across the three platforms. All three public cloud platforms have also started offering different types of NoSQL databases as a service. I would also like to study how different types of NoSQL database service performs on each cloud platform.

My professional experience with working on the cloud has been minimal before these experiments. One possible reason for the vast differences in performance between the three cloud providers is that there may be minor aspects that need to be tweaked on the operating system or at the cloud control panel on each cloud provider to improve performance. For this experiment, I only chose the default options in creating each VM. It may be worthwhile to study the operating system's overall performance on these platforms and implement improvements at this level before repeating the experiments described above. In a similar vein, another path of exploration may be to use a database profiler to see where the choke points are when the databases are under stress to understand where each of the cloud platforms. By extension, the hypervisors may be having issues.

## Reflections

Since I have not used any cloud platform before this study, I had no bias towards any platform. One of my initial assumptions was that all cloud providers would perform

similarly. This assumption proved to be incorrect. AWS has consistently had the most significant market share since I started this study, and Microsoft has gained a lot of market share since it enhanced its offerings a few years ago. I initially, and wrongly, assumed that I would get roughly the same performance from each of the platforms. I also hoped that consistent results from the three cloud providers showed one or more factors or factor interactions as significant on all platforms. I feel my results could be more conclusive if I could show that one or two factors stand out. While Azure had a poor showing in my experiments, I believe that the environment used in these trials was not the use case for the Azure cloud platform. There is likely a different environment in Azure that will run queries much faster. My results also hint at the notion that the servers on Google's platform may be overprovisioned, which would lead to the inconsistent throughput found in my experiments.

In the academic literature, there is such a vast array of methods researchers have used to improve some database performance aspects. Many of the approaches used were novel systems developed by the researchers. With all the minor settings available to DBAs, I do not see unique systems being of much use to those responsible for databases' day-to-day operations. If I were to continue with this research, it would be interesting to use the Six Sigma approach and find other database settings that may significantly improve database performance.

## Summary and Study Conclusions

This research's main takeaway is that Microsoft Azure is not the right platform to run MySQL on a Linux VM, both from a cost and performance perspective. However, I

do not think this advice would be new to a DBA. When I worked as a DBA, most ETL jobs were scheduled at night, mostly to avoid table locks during the daytime. Surprisingly, this factor was not significant on any of the platforms. While disk I/O was cited throughout the literature as the bottleneck for database throughput, I could not find significance in changing the I/O settings. However, I did see minor improvements by keeping the InnoDB I/O capacity at a lower level. The academic literature has supported the TPC specifications. More DBAs should use this benchmarking standard to test their DBMS and settings to ensure they are getting the expected performance from their systems.

With the proliferation of databases-as-a-service, I feel this is the next area for exploration. While Azure had a poor showing in my experiments, I believe there is an opportunity to find the services where Azure can excel. Similarly, it would be interesting to try different Google services to find one that would provide more consistent results than I found in these experiments. Since I started this program, all three cloud providers have greatly expanded the services they offer. If they continue adding services simultaneously, a researcher could be busy continuing my work on each new service, trying to find the most efficient service for the lowest price.

# References

Abadi, D., Franklin, M. J., Gehrke, J., Haas, L. M., Halevy, A. Y., Hellerstein, J. M.,… Doan, A. (2016). The Beckman report on database research. *Communications of the ACM*, *59*(2), 92–99. doi:10.1145/2845915.

Abdelmaboud, A., Jawawi, D., Ghani, I., Elsafi, A., & Kitchenham, B. (2015). Quality of service approaches in cloud computing: A systematic mapping study. *Journal of Systems and Software, 101*, 159–179. doi:10.1016/j.jss.2014.12.015.

Abourezq, M., & Idrissi, A. (2016). Database-as-a-service for big data: An overview. *International Journal of Advanced Computer Science and Applications*, *7*(1), 157–177. doi:10.14569/IJACSA.2016.070124.

Abramson, E. L., Paul, C. R., Petershack, J., Serwint, J., Fischel, J. E., Rocha, M.,… Li, S.-T. T. (2018). Conducting quantitative medical education research: From design to dissemination. *Academic Pediatrics, 18*(2), 129–139. doi:10.1016/j.acap.2017.10.008.

Afify, G. M., El Bastawissy, A., & Hegazy, O. M. (2015). A hybrid filtering approach for storage optimization in main-memory cloud database. *Egyptian Informatics Journal*, *16*(3), 329–337. doi:10.1016/j.eij.2015.06.007.

Ahmed, M. (2013). Physical server and virtual server: The performance trade-offs. *European Scientific Journal*, *9*(12), 222–232. Retrieved from http://www.eujournal.org/index.php/esj/article/view/1009.

Albers, M. J. (2017). Quantitative data analysis—In the graduate curriculum. *Journal of Technical Writing and Communication*, *47*(2), 215–233. doi:10.1177/0047281617692067.

Almeida, R., Furtado, P., & Bernardino, J. (2015). Performance evaluation MySQL InnoDB and Microsoft SQL Server 2012 for decision support environments. In *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering - C3S2E '15* (pp. 56–62). New York, NY: ACM Press. doi:10.1145/2790798.2790808.

Anthony, S., & Antony, J. (2015). Academic leadership and Lean Six Sigma. *International Journal of Quality & Reliability Management*, *33*(7), 1002–1018. doi:10.1108/IJQRM-03-2015-0047.

Antony, J., Gupta, S., Sunder, V. M., & Gijo, E. V. (2018). Ten commandments of Lean Six Sigma: A practitioners' perspective. *International Journal of Productivity and Performance Management*, *67*(6), 1033–1044. doi:10.1108/IJPPM-07-2017-0170.

Antony, J., Snee, R., & Hoerl, R. (2017). Lean Six Sigma: Yesterday, today and tomorrow. *International Journal of Quality & Reliability Management*, *34*(7), 1073–1093. doi:10.1108/IJQRM-03-2016-0035.

Antony, J., Sony, M., Dempsey, M., Brennan, A., Farrington, T., & Cudney, E. A. (2019). An evaluation into the limitations and emerging trends of Six Sigma: An empirical study. *The TQM Journal*, *31*(2), 205–221. doi:10.1108/TQM-12-2018-0191.

Argilaga, M. (2003). Observational methods (general). In R. Fernández-Ballesteros (Ed.), *Encyclopedia of psychological assessment* (Vol. 1, pp. 633-638). London, England: SAGE Publications. doi:10.4135/9780857025753.n136.

Aryanezhad, M. B., Badri, S. A., & Rashidi Komijan, A. (2010). Threshold-based method for elevating the system's constraint under theory of constraints. *International Journal of Production Research, 48*(17), 5075–5087. doi:10.1080/00207540903059505.

Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A. S., & Buyya, R. (2014). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, *79*, 3–15. doi:10.1016/j.jpdc.2014.08.003.

Babcock, C. (2016). AWS, Microsoft Azure top Gartner's magic quadrant for IaaS. Retrieved from http://www.informationweek.com.

Badola, V. (2019). AWS AMI virtualization types: HMV vs PV. Retrieved from https://cloudacademy.com/blog/aws-ami-hvm-vs-pv-paravirtual-amazon.

Barata, M., Bernardino, J., & Furtado, P. (2014). YCSB and TPC-H: Big data and decision support benchmarks. In *2014 IEEE International Congress on Big Data* (pp. 800–801). doi:10.1109/BigData.Congress.2014.128.

Barnham, C. (2015). Quantitative and qualitative research: Perceptual foundations. *International Journal of Market Research, 57*(6), 837–854. doi:10.2501/IJMR-2015-070.

Bärnighausen, T., Røttingen, J.-A., Rockers, P., Shemilt, I., & Tugwell, P. (2017). Quasi-experimental study designs series—Paper 1: Introduction: Two historical lineages. *Journal of Clinical Epidemiology, 89,* 4–11. doi:10.1016/j.jclinepi.2017.02.020.

Bernstein, D. (2014). Containers and cloud: From LXC to Docker to Kubernetes. *IEEE Cloud Computing*, *1*(3), 1–84. doi:10.1109/MCC.2014.51.

Bhimani, J., Yang, J., Yang, Z., Mi, N., Xu, Q., Awasthi, M.,… Balakrishnan, V. (2016). Understanding performance of I/O intensive containerized applications for NVMe SSDs. In *2016 IEEE 35th International Performance Computing and Communications Conference* (pp. 1–8). doi:10.1109/PCCC.2016.7820650.

Bizarro, P. A. (2015). Effect of different database structure representations, query languages, and task characteristics on information retrieval. *Journal of Management Information and Decision Sciences*, *18*(1), 27–53. Retrieved from https://www.abacademies.org/journals/journal-of-management-information-and-decision-sciences-home.html.

Bonthu, S., Thammiraju, S., & Murthy, Y. (2014). Study on database virtualization for database as a service (DBaaS). *International Journal of Advanced Research in Computer Science*, *5*(2), 31–34. Retrieved from http://www.ijarcs.info.

Brutus, S., Aguinis, H., & Wassmer, U. (2013). Self-reported limitations and future directions in scholarly reports: Analysis and recommendations. *Journal of Management*, *39*(1), 48–75. doi:10.1177/0149206312455245.

Chandra, D. G. (2015). BASE analysis of NoSQL database. *Future Generation Computer Systems, 52,* 13–21. doi:10.1016/j.future.2015.05.003.

Chang, H.-T., & Lin, T.-H. (2016). A database as a service for the healthcare system to store physiological signal data. *PLOS ONE, 11*(12), 1–27. doi:10.1371/journal.pone.0168935.

Chen, P., & Popovich, P. (2002). *Correlation*. Thousand Oaks, CA: SAGE Publications, Inc.. doi:10.4135/9781412983808.

Chrószcz, A., Łukasik, P., & Lupa, M. (2016). Analysis of performance and optimization of point cloud conversion in spatial databases. *IOP Conference Series: Earth and Environmental Science*, *44*(5), 1-6. doi:10.1088/1755-1315/44/5/052011.

Chung, H., & Nah, Y. (2017). Performance comparison of distributed processing of large volume of data on top of Xen and Docker-based virtual clusters. In *Database Systems for Advanced Applications: 22nd International Conference, DASFAA 2017, Suzhou, China, March 27-30, 2017, Proceedings, Part I* (pp. 103–113). Cham, Switzerland: Springer International Publishing. doi:10.1007/978-3-319-55753-3_7.

Collins, L. M., Dziak, J. J., & Li, R. (2009). Design of experiments with multiple independent variables: A resource management perspective on complete and reduced factorial designs. *Psychological Methods*, *14*(3), 202–224. doi:10.1037/a0015826.

Cox, S., Elton, V., Garside, J. A., Kotsialos, A., Marmo, J. V., Cunha, L.,… Gill, C. (2016). A new method to improve the objectivity of early Six Sigma analysis. *International Journal of Quality & Reliability Management, 33*(9), 1364–1393. doi:10.1108/IJQRM-02-2015-0023.

Dean, D. J., Nguyen, H., Wang, P., Gu, X., Sailer, A., & Kochut, A. (2016). PerfCompass: Online performance anomaly fault localization and inference in infrastructure-as-a-Service clouds. *IEEE Transactions on Parallel and Distributed Systems*, *27*(6), 1742–1755. doi:10.1109/TPDS.2015.2444392.

Department of Health, Education, & Welfare. (1979). *The Belmont Report*. Retrieved from https://www.hhs.gov/ohrp/sites/default/files/the-belmont-report-508c_FINAL.pdf.

Dien, J. (2017). Best practices for repeated measures ANOVAs of ERP data: Reference, regional channels, and robust ANOVAs. *International Journal of Psychophysiology, 111*, 42–56. doi:10.1016/j.ijpsycho.2016.09.006.

Ding, X., Shan, J., & Jiang, S. (2016). A general approach to scalable buffer pool management. *IEEE Transactions on Parallel and Distributed Systems*, *27*(8), 2182–2195. doi:10.1109/TPDS.2015.2484321.

Duarte, J. (2017). Data disruption. *Quality Progress*, *50*(9), 20–24. Retrieved from https://asq.org/quality-progress/.

Dusick, D. (2015). Assumptions and limitations. Retrieved from http://bold-ed.com/barrc/assumptions.htm.

Ellis, T., & Levy, Y. (2009). Towards a guide for novice researchers on research methodology: Review and proposed methods. *Issues in Informing Science and Information Technology, 6*, 323–337. doi:10.28945/1062.

E. V., G., Antony, J., & Sunder M., V. (2019). Application of Lean Six Sigma in IT support services –A case study. *The TQM Journal*, *31*(3), 417–435. doi:10.1108/TQM-11-2018-0168.

Faul, F., Erdfelder, E., Lang, A., & Buchner, A. (2007). G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior Research Methods, 39*(2), 175–191. doi:10.3758/BF03193146.

Ferretti, L., Colajanni, M., & Marchetti, M. (2014). Distributed, concurrent, and independent access to encrypted cloud databases. *IEEE Transactions on Parallel and Distributed Systems*, *25*(2), 437–446. doi:10.1109/TPDS.2013.154.

Ferretti, L., Pierazzi, F., Colajanni, M., & Marchetti, M. (2014). Performance and cost evaluation of an adaptive encryption architecture for cloud databases. *IEEE Transactions on Cloud Computing*, *2*(2), 143–155. doi:10.1109/TCC.2014.2314644.

Fisher, M. J., & Marshall, A. P. (2009). Understanding descriptive statistics. *Australian Critical Care, 22*(2), 93–97. doi:10.1016/j.aucc.2008.11.003.

Foss, N. J., & Hallberg, N. L. (2017). Changing assumptions and progressive change in theories of strategic organization. *Strategic Organization*, *15*(3), 410–422. doi:10.1177/1476127016671099.

Gabbiadini, A., & Greitemeyer, T. (2017). Uncovering the association between strategy video games and self-regulation: A correlational study. *Personality and Individual Differences*, 104, 129–136. doi:10.1016/j.paid.2016.07.041.

Gharbaoui, M., Martini, B., Adami, D., Giordano, S., & Castoldi, P. (2016). Cloud and network orchestration in SDN data centers: Design principles and performance evaluation. *Computer Networks*, 108, 279–295. doi:10.1016/j.comnet.2016.08.029.

Gholami, M., Daneshgar, F., Low, G., & Beydoun, G. (2016). Cloud migration process—A survey, evaluation framework, and open challenges. *Journal of Systems and Software*, 120, 31–69. doi:10.1016/j.jss.2016.06.068.

Glen, S. (2013). F statistic/F value: Simple definition and interpretation. Retrieved from https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/f-statistic-value-test/.

Glen, S. (2014). P-value in statistical hypothesis testing: What is it? Retrieved from https://www.statisticshowto.datasciencecentral.com/p-value/.

Gonçalves, F. A. C. A., Guimarães, F. G., & Souza, M. J. F. (2014). Query join ordering optimization with evolutionary multi-agent systems. *Expert Systems with Applications*, *41*(15), 6934–6944. doi:10.1016/j.eswa.2014.05.005.

Grace-Martin, K. (2019). Why report estimated marginal means in SPSS GLM? Retrieved from https://www.theanalysisfactor.com/why-report-estimated-marginal-means-in-spss-glm.

Guo, B., Yu, J., Liao, B., Yang, D., & Lu, L. (2017). A green framework for DBMS based on energy-aware query optimization and energy-efficient query processing. *Journal of Network and Computer Applications*, *84*, 118-130. doi:10.1016/j.jnca.2017.02.015.

Garg, N., Singla, S., & Jangra, S. (2016). Challenges and Techniques for Testing of Big Data. *Procedia Computer Science*, *85*, 940–948. doi.:10.1016/j.procs.2016.05.285.

Guo, S.-S., Yuan, Z.-M., Sun, A.-B., & Yue, Q. (2015). A new ETL approach based on data virtualization. *Journal of Computer Science and Technology*, *30*(2), 311–323. doi:10.1007/s11390-015-1524-3.

HammerDB. (2018a). About. Retrieved from https://www.hammerdb.com/about.html.

HammerDB. (2018b). CLI scripting. Retrieved from https://www.hammerdb.com/docs/ch08s08.html.

HammerDB. (2018c). Creating the schema. Retrieved from https://hammerdb.com/docs/ch04s04.html.

HammerDB. (2018d). Understanding the TPC-C workload. Retrieved from https://www.hammerdb.com/docs/ch03s05.html.

HammerDB Blog. (n.d.). About. Retrieved from https://www.hammerdb.com/blog/hammerdb/.

Han, R., Ghanem, M. M., Guo, L., Guo, Y., & Osmond, M. (2014). Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. *Future Generation Computer Systems*, *32*(1), 82–98. doi:10.1016/j.future.2012.05.018.

Hancock, G. R., & McNeish, D. M. (2017). More powerful tests of simple interaction contrasts in the two-way factorial design. *The Journal of Experimental Education*, *85*(1), 24–35. doi:10.1080/00220973.2015.1065220.

Henneberger, M. (2016). Covering peak demand by using cloud services – An economic analysis. *Journal of Decision Systems*, *25*(2), 118–135. doi:10.1080/12460125.2016.1141275.

Hilier, F., & Lieberman, G. (2015). Introduction to operational research. In *Introduction to Operational Research* (10th ed., pp. 731–784). New York, NY: McGraw Hill Education. doi:10.2307/2077150.

Honing, A., & Porter, N. (2017). 7 ways we harden our KVM hypervisor at Google Cloud: Security in plaintext. Retrieved from https://cloud.google.com/blog/products/gcp/7-ways-we-harden-our-kvm-hypervisor-at-google-cloud-security-in-plaintext.

Houghton, C., & Casey, D. (2013). Rigour in qualitative case-study research. *Nurse Researcher*, *20*(4), 12–17. https://rcni.com/nurse-researcher.

Hsieh, C.-T., Lin, B., & Manduca, B. (2007). Information technology and Six Sigma. *Journal of Computer Information Systems*, *47*(4), 1–10. doi:10.1080/08874417.2007.11645975.

Hudson, J. (2017). CL6 allows three shots at better improvement. *ISE: Industrial & Systems Engineering at Work*, *49*(10), 43–48. Retrieved from https://iise.org/ISEmagazine/.

Hummaida, A. R., Paton, N. W., & Sakellariou, R. (2016). Adaptation in cloud resource configuration: A survey. *Journal of Cloud Computing*, *5*(1), 7. doi:10.1186/s13677-016-0057-9.

Hwang, K., Bai, X., Shi, Y., Li, M., Chen, W.-G., & Wu, Y. (2016). Cloud performance modeling with benchmark evaluation of elastic scaling strategies. *IEEE Transactions on Parallel and Distributed Systems*, *27*(1), 130–143. doi:10.1109/TPDS.2015.2398438.

IBM. (2017). ANOVA method. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSLVMB_25.0.0/statistics_casestudies_project_ddita/spss/tutorials/varcomp_anova_method.html.

Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., & Epema, D. H. J. (2011). Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, *22*(6), 931–945. doi:10.1109/TPDS.2011.66.

Jackson, S., & Brashers, D. (1994). *Random factors in ANOVA* (Vol. 17). Thousand Oaks, CA: SAGE Publications, Inc. doi:10.4135/9781412985567.

Januzaj, Y., Ajdari, J., & Selimi, B. (2015). DBMS as a cloud service: Advantages and disadvantages. *Procedia - Social and Behavioral Sciences*, *195*, 1851–1859. doi:10.1016/j.sbspro.2015.06.412.

Jha, M., Jha, S., & O'Brien, L. (2016). Combining big data analytics with business process using reengineering. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)* (pp. 1–6). IEEE. doi:10.1109/RCIS.2016.7549307.

Jia, X., Li, Y., Sharma, A., Li, Y., Xie, G., Wang, G., … Ding, X. (2017). Application of sequential factorial design and orthogonal array composite design (OACD) to study combination of 5 prostate cancer drugs. *Computational Biology and Chemistry*, *67*, 234–243. doi:10.1016/j.compbiolchem.2017.01.010.

Jiang, S., & Zhang, X. (2005). Making LRU Friendly to weak locality workloads: A novel replacement algorithm to improve buffer cache performance. *IEEE Transactions on Computers*, *54*(8), 939–952. doi:10.1109/TC.2005.130.

Jones, B., & Montgomery, D. C. (2017). Partial replication of small two-level factorial designs. *Quality Engineering*, *29*(2), 190–195. doi:10.1080/08982112.2016.1254797.

Kaltenecker, N., Hess, T., & Huesig, S. (2015). Managing potentially disruptive innovations in software companies: Transforming from on-premises to the on-demand. *The Journal of Strategic Information Systems*, *24*(4), 234–250. doi:10.1016/j.jsis.2015.08.006.

Kansteiner, K., & König, S. (2020). Role of qualitative content analysis in Mixed Methods research designs. *Forum: Qualitative Social Research*, *21*(1), 221-242. Retrieved from http://www.qualitative-research.net.

Kimberlin, C. L., & Winterstein, A. G. (2008). Validity and reliability of measurement instruments used in research. *American Journal of Health-System Pharmacy*, *65*(23), 2276–2284. doi:10.2146/ajhp070364.

Kingman, J. F. C. (2009). The first Erlang century—and the next. *Queueing Systems*, *63* (1–4), 3–12. doi:10.1007/s11134-009-9147-4.

Kong, X. (2012). Research on performance optimization of OLTP systems based on InnoDB. *Journal of Theoretical and Applied Information Technology, 46*(2), 638–642. Retrieved from http://jatit.org/.

Kozhirbayev, Z., & Sinnott, R. O. (2017). A performance comparison of container-based technologies for the cloud. *Future Generation Computer Systems, 68,* 175–182. doi:10.1016/j.future.2016.08.025.

Kumar, N., & Saxena, S. (2015). Migration performance of cloud applications - A quantitative analysis. *Procedia Computer Science, 45,* 823–831. doi:10.1016/j.procs.2015.03.163.

Lang, W., Ramachandra, K., DeWitt, D. J., Xu, S., Guo, Q., Kalhan, A., & Carlin, P. (2016). Not for the timid. *Proceedings of the VLDB Endowment, 9*(13), 1245–1256. doi:10.14778/3007263.3007264.

Laux, C., Li, N., Seliger, C., & Springer, J. (2017). Impacting big data analytics in higher education through Six Sigma techniques. *International Journal of Productivity and Performance Management, 66*(5), 662–679. doi:10.1108/IJPPM-09-2016-0194.

Lee, K.-H. (2014). Performance improvement of database compression for OLTP workloads. *IEICE Transactions on Information and Systems, E97-D*(4), 976–980. doi:10.1587/transinf.E97.D.976.

LeMahieu, P. G., Nordstrum, L. E., & Cudney, E. A. (2017). Six Sigma in education. *Quality Assurance in Education, 25*(1), 91–108. doi:10.1108/QAE-12-2016-0082.

Liaqat, M., Chang, V., Gani, A., Hamid, S. H. A., Toseef, M., Shoaib, U., & Ali, R. L. (2017). Federated cloud resource management: Review and discussion. *Journal of Network and Computer Applications, 77,* 87–105. doi:10.1016/j.jnca.2016.10.008.

Loghin, D., Tudor, B. M., Zhang, H., Ooi, B. C., & Teo, Y. M. (2015). A performance study of big data on small nodes. *Proceedings of the VLDB Endowment*, *8*(7), 762–773. doi:10.14778/2752939.2752945.

Luo, Y., Zhou, S., & Guan, J. (2015). LAYER: A cost-efficient mechanism to support multi-tenant database as a service in cloud. *Journal of Systems and Software*, *101*, 86–96. doi:10.1016/j.jss.2014.11.038.

Maleyeff, J., & Kaminsky, F. C. (2002). Six Sigma and introductory statistics education. *Education + Training*, *44*(2), 82–89. doi:10.1108/00400910210419982.

Mandelbaum, M., & Hlynka, M. (2009). History of queueing theory in Canada prior to 1980. *INFOR: Information Systems and Operational Research*, *47*(4), 335–353. doi:10.3138/infor.47.4.335.

Mann, Z. Á. (2015). Allocation of virtual machines in cloud data centers—A survey of problem models and optimization algorithms. *ACM Computing Surveys*, *48*(1), 1–34. doi:10.1145/2797211.

Marshall, G., & Jonker, L. (2010). An introduction to descriptive statistics: A review and practical guide. *Radiography*, *16*(4), e1-e7. doi:10.1016/j.radi.2010.01.001.

McDonald, J. (2014). *Handbook of biological statistics* (3rd ed.). Baltimore, MD: Sparky House Publishing. Retrieved from http://www.biostathandbook.com/independence.html.

Microsoft. (2019). Isolation in the Azure public cloud. Retrieved from https://docs.microsoft.com/en-us/azure/security/fundamentals/isolation-choices.

Minitab. (2018). Mulligan? How many runs do you need to produce a complete data set? Retrieved from https://blog.minitab.com/blog/mulligan-how-many-runs-produce-complete-data-set.

Mouaky, M., Benabbou, L., & Berrado, A. (2018). DMADV approach to evaluate the Adaptive Kanban performance for inventory management process: The case of Moroccan public pharmaceutical supply chain. *Supply Chain Forum: An International Journal*, *19*(3), 178–190. doi:10.1080/16258312.2018.1484249.

Nanda, R., Chande, S., & Sharma, K. (2017a). Determining appropriate cache-size for cost-effective cloud database queries. *International Journal of Computer Applications*, *157*(6), 29–34. doi:10.5120/ijca2017912651.

Nanda, R., Chande, S., & Sharma, K. (2017b). Determining the effect of similar queries in cache-based cloud datastores. *International Journal of Advanced Research in Computer Science*, *8*(3), 906–911. Retrieved from http://www.ijarcs.info/.

Narasayya, V., Menache, I., Singh, M., Li, F., Syamala, M., & Chaudhuri, S. (2015). Sharing buffer pool memory in multi-tenant relational database-as-a-service. *Proceedings of the VLDB Endowment*, *8*(7), 726–737. doi:10.14778/2752939.2752942.

Nedelcu, B., Ionescu, A. M., Ionescu, A. M., & Vasile, A. G. (2014). Reshaping smart businesses with cloud database solutions. *Database Systems Journal*, *V*(4), 21–38. Retrieved from http://dbjournal.ro/.

Nguyen, T. L. H., & Nagase, K. (2019). The influence of total quality management on customer satisfaction. *International Journal of Healthcare Management*, *12*(4), 277–285. doi:10.1080/20479700.2019.1647378.

National Institutes of Standards and Technology/Semiconductor Manufacturing Technology. (2012a). 5.3.3.3.2. Full factorial example. In *e-Handbook of Statistical Methods*. Retrieved from http://www.itl.nist.gov/div898/handbook/pri/section3/pri3332.htm.

National Institutes of Standards and Technology/Semiconductor Manufacturing Technology. (2012b). 1.3.5.18. Yates algorithm. In *e-Handbook of Statistical Methods*. Retrieved from https://www.itl.nist.gov/div898/handbook/eda/section3/eda35i.htm.

National Institutes of Standards and Technology/Semiconductor Manufacturing Technology. (2012c). 1.3.5.10. Levene Test for equality of variances. In *e-Handbook of Statistical Methods*. Retrieved from https://www.itl.nist.gov/div898/handbook/eda/section3/eda35a.htm.

Oehlert, G. (2010). *A first course in design and analysis of experiments*. Kluwer Academic Publishers. Retrieved from http://users.stat.umn.edu/~gary/book/fcdae.pdf.

Oracle Corporation. (2019a). 15.13 InnoDB startup options and system variables. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/innodb-parameters.html.

Oracle Corporation. (2019b). 8.12.3.1 How MySQL uses memory. Retrieved from https://dev.mysql.com/doc/refman/8.0/en/memory-use.html.

Orair, G. H., Teixeira, C. H. C., Meira, W., Wang, Y., & Parthasarathy, S. (2010). Distance-based outlier detection. *Proceedings of the VLDB Endowment*, *3*(1–2), 1469–1480. doi:10.14778/1920841.1921021.

Park, J., & Park, M. (2016). Qualitative versus quantitative research methods: Discovery or justification? *Journal of Marketing Thought*, *3*(1), 1–7. doi:10.15577/jmt.2016.03.01.1.

Pennsylvania State University. (2018). 6.1 - Introduction to generalized linear models. Retrieved from https://newonlinecourses.science.psu.edu/stat504/node/216/.

Pennsylvania State University. (2020). 10.2.1 – ANOVA assumptions. Retrieved from https://online.stat.psu.edu/stat500/lesson/10/10.2/10.2.1.

Psomas, E. (2016). The underlying factorial structure and significance of the Six Sigma difficulties and critical success factors. *The TQM Journal*, *28*(4), 530–546. doi:10.1108/TQM-04-2015-0049.

Raza, B., Kumar, Y. J., Malik, A. K., Anjum, A., & Faheem, M. (2018). Performance prediction and adaptation for database management system workload using case-based reasoning approach. *Information Systems*, *76*(2018), 46–58. doi:10.1016/j.is.2018.04.005.

Reddy, P. V. V., & Shyamala, K. (2016). New scoring formula to rank hypervisors' performance complementing with statistical analysis using DOE. *Future Generation Computer Systems*, 61, 54–65. doi:10.1016/j.future.2016.02.012.

Reosekar, R., & Pohekar, S. (2014). Six Sigma methodology: A structured review. *International Journal of Lean Six Sigma*, *5*(4), 392–422. doi:10.1108/IJLSS-12-2013-0059.

Richardson, R. (2014). Disambiguating databases. *Communications of the ACM, 58*(1), 54–61. doi:10.1145/2687880.

RightScale. (2019). *RightScale 2019 state of the cloud report from Flexera*. Retrieved from https://www.rightscale.com/press-releases/rightscale-2019-state-of-the-cloud-report.

Saharan, K., & Kumar, A. (2015). Fog in comparison to cloud: A survey. *International Journal of Computer Applications*, *122*(3), 10–12. doi:10.5120/21679-4773.

Sajid, M. (2016). Two way ANOVA calculation by hand (ANalysis Of VAriance). Retrieved from https://stepupanalytics.com/two-way-anova-calculation-by-hand-analysis-of-variance.

Sakr, S. (2014). Cloud-hosted databases: Technologies, challenges and opportunities. *Cluster Computing*, *17*(2), 487–502. doi:10.1007/s10586-013-0290-7.

Sanchez, S. M., & Wan, H. (2015). Work smarter, not harder: A tutorial on designing and conducting simulation experiments. In 2015 Winter Simulation Conference (WSC) (Vol. 2016-Febru, pp. 1795–1809). IEEE. doi:10.1109/WSC.2015.7408296.

Sedgwick, P. (2014). Pitfalls of statistical hypothesis testing: Type I and Type II errors. *British Medical Journal*, *349*, 1-2. doi:10.1136/bmj.g4287.

Sharma, H., Nelson, S., & Singh, S. (2016). Tuning I/O subsystem: A key component in RDBMS performance tuning. *Database Systems Journal*, *7*(1), 3–11. Retrieved from http://dbjournal.ro/.

Shmueli, E., Vaisenberg, R., Gudes, E., & Elovici, Y. (2014). Implementing a database encryption solution, design and implementation issues. *Computers & Security*, *44*(2), 33–50. doi:10.1016/j.cose.2014.03.011.

Sikeridis, D., Papapanagiotou, I., Rimal, B. P., & Devetsikiotis, M. (2017). A comparative taxonomy and survey of public cloud infrastructure vendors, 1–20. Retrieved from http://arxiv.org/abs/1710.01476.

Silva Filho, M. C., Monteiro, C. C., Inácio, P. R. M., & Freire, M. M. (2018). Approaches for optimizing virtual machine placement and migration in cloud environments: A survey. *Journal of Parallel and Distributed Computing*, *111*, 222–250. doi:10.1016/j.jpdc.2017.08.010.

Smith, C. J. (2012). Type I and Type II errors: What are they and why do they matter? *Phlebology: The Journal of Venous Disease*, *27*(4), 199–200. doi:10.1258/phleb.2012.012j04.

Šoltés, E., Zelinová, S., & Bilíková, M. (2019). General linear model: An effective tool for analysis of claim severity in motor third party liability insurance. *Statistics in Transition New Series*, *20*(4), 13–31. doi:10.21307/stattrans-2019-032.

Sommer, T., & Subramanian, R. (2013). Implementing cloud computing in small & mid-market life-sciences. *Journal of International Technology and Information Management*, *22*(3), 55–76. Retrieved from https://scholarworks.lib.csusb.edu/jitim/.

Sony, M., & Naik, S. (2019). Six Sigma with C-K theory for innovations in operational excellence: A case study. *Benchmarking: An International Journal, 26*(7), 2105–2121. doi:10.1108/BIJ-08-2018-0241.

Sreedharan, R. V., Sunder, V. M., & Raju, R. (2018). Critical success factors of TQM, Six Sigma, Lean and Lean Six Sigma. *Benchmarking: An International Journal, 25*(9), 3479–3504. doi:10.1108/BIJ-08-2017-0223.

Srivastava, R. (2018). Exploration of in-memory computing for big data analytics using queuing theory. In *Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications - HP3C* (pp. 11–16). New York, NY: ACM Press. doi:10.1145/3195612.3195621.

Statistics Solutions. (n.d.). Assumptions of the factorial ANOVA. Retrieved from https://www.statisticssolutions.com/assumptions-of-the-factorial-anova/.

Sullivan, G. M., & Feinn, R. (2012). Using effect size—or why the P value is not enough. *Journal of Graduate Medical Education*, *4*(3), 279–282. doi:10.4300/JGME-D-12-00156.1.

Taherdoost, H. (2016). Sampling methods in research methodology; How to choose a sampling technique for research. *International Journal of Academic Research in Management*, *5*(2), 18–27. Retrieved from http://elvedit.com/journals/IJARM/.

Tailor, P., & Morena, R. (2017). A survey of database buffer cache management approaches. *International Journal of Advanced Research in Computer Science*, *8*(3), 409–414. Retrieved from http://www.ijarcs.info/.

Tajbakhsh, H., Dehsangi, M., & Analoui, M. (2017). Performance profiling of database systems in Xen. In *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)* (Vol. 2017–Janua, pp. 90–97). IEEE. doi:10.1109/ICCKE.2017.8167935.

Tak, B. C., Urgaonkar, B., & Sivasubramaniam, A. (2013). Cloudy with a chance of cost savings. *IEEE Transactions on Parallel and Distributed Systems*, *24*(6), 1223–1233. doi:10.1109/TPDS.2012.307.

Tapdiya, A., & Xue, Y. (2014). Performance variations in profiling MySQL server on the Xen platform: Is it Xen or MySQL? *International Journal of Computer Science and Information Technology*, *6*(2), 1–19. doi:10.5121/ijcsit.2014.6201.

Tian, B., Huang, J., Mozafari, B., & Schoenebeck, G. (2018). Contention-aware lock scheduling for transactional databases. *Pvldb*, *11*(5), 648-662. doi:10.1145/3177732.3177740.

Transaction Performance Council. (n.d.). Active TPC benchmarks. Retrieved from http://www.tpc.org/information/benchmarks.asp.

Transaction Performance Council. (2010). TPC benchmark C. Retrieved from http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf.

Tseng, F.-H., Chen, X., Chou, L.-D., Chao, H.-C., & Chen, S. (2015). Support vector machine approach for virtual machine migration in cloud data center. *Multimedia Tools and Applications*, *74*(10), 3419–3440. doi:10.1007/s11042-014-2086-z.

Ustinova, T., & Jamshidi, P. (2015). Modelling multi-tier enterprise applications behaviour with design of experiments technique. In *Proceedings of the 1st International Workshop on Quality-Aware DevOps - QUDOS 2015* (pp. 13–18). New York, NY: ACM Press. doi:10.1145/2804371.2804374.

Van den Berg, R. (2020). SPSS Kolmogorov-Smirnov test for normality. Retrieved from https://www.spss-tutorials.com/spss-kolmogorov-smirnov-test-for-normality.

Vilaplana, J., Solsona, F., Teixido, I., Usié, A., Karathia, H., Alves, R., & Mateo, J. (2014). Database constraints applied to metabolic pathway reconstruction tools. *The Scientific World Journal*, *2014*, 1–12. doi:10.1155/2014/967294.

Whaiduzzaman, M., Haque, M. N., Chowdhury, R. K., & Gani, A. (2014). A study on strategic provisioning of cloud computing services. *The Scientific World Journal*, *2014*, 1–16. doi:10.1155/2014/894362.

Xavier, M. G., Matteussi, K. J., Lorenzo, F., & De Rose, C. A. F. (2016). Understanding performance interference in multi-tenant cloud databases and web applications. In *2016 IEEE International Conference on Big Data (Big Data)* (pp. 2847–2852). IEEE. doi:10.1109/BigData.2016.7840933.

Xiang, T., Li, X., Chen, F., Guo, S., & Yang, Y. (2016). Processing secure, verifiable and efficient SQL over outsourced database. *Information Sciences*, *348*, 163–178. doi:10.1016/j.ins.2016.02.018.

Xu, B., Wang, W., Wu, Y., Shi, Y., & Lu, C. (2017). Internet of things and big data analytics for smart oil field malfunction diagnosis. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)* (pp. 178–181). IEEE. doi:10.1109/ICBDA.2017.8078802.

Xu, Z., Tu, Y.-C., & Wang, X. (2015). Online energy estimation of relational operations in database systems. *IEEE Transactions on Computers*, *64*(11), 3223–3236. doi:10.1109/TC.2015.2394309.

Yang, C., Jin, P., Yue, L., & Yang, P. (2016). Efficient buffer management for tree indexes on solid state drives. *International Journal of Parallel Programming*, *44*(1), 5–25. doi:10.1007/s10766-014-0340-7.

Yang, K. K., Cayirli, T., & Low, J. M. W. (2016). Predicting the performance of queues-A data analytic approach. *Computers and Operations Research*, *76*, 33–42. doi:10.1016/j.cor.2016.06.005.

Yu, T., & Pradel, M. (2018). Pinpointing and repairing performance bottlenecks in concurrent programs. *Empirical Software Engineering*, *23*(5), 3034–3071. doi:10.1007/s10664-017-9578-1.

Zhao, Y., Li, S., Hu, S., Wang, H., Yao, S., Shao, H., & Abdelzaher, T. (2016). An experimental evaluation of datacenter workloads on low-power embedded micro servers. *Proceedings of the VLDB Endowment*, *9*(9), 696–707. doi:10.14778/2947618.2947625.

Zhou, Y., Taneja, S., Qin, X., Ku, W.-S., & Zhang, J. (2017). EDOM: Improving energy efficiency of database operations on multicore servers. *Future Generation Computer Systems, 2017*, 1-14. doi:10.1016/j.future.2017.02.043.

Appendix A: Script db_install.sh

```
#!/bin/bash

#get file to add official MySQL repository to apt-get
wget https://dev.mysql.com/get/mysql-apt-config_0.8.9-1_all.deb

#install mysql repositories
dpkg -i mysql-apt-config_0.8.9-1_all.deb

#Tell Debian to refresh repositories
apt-get update

#Request MySQL Community get installed
apt-get -y --force-yes install mysql-community-server

#create the database for testing

mysql --user=root --password=db-test1 -e "create database tpcc;"
mysql --user=root --password=db=test1 -e "show databases;"

#install the connector library for HammerDB
apt-get -y --force-yes install libmysqlclient20

#Download HammerDB 3.1
wget "https://sourceforge.net/projects/hammerdb/files/HammerDB/HammerDB-
3.1/HammerDB-3.1-Linux-x86-64-Install"

#Change to executable
chmod +x HammerDB-3.1-Linux-x86-64-Install

#Run the self installer-sending yes and path to answer installer prompts
echo -e "Y /usr/local/HammerDB-3.1" | ./HammerDB-3.1-Linux-x86-64-Install
```

Appendix B: sqlrun.sh

```
#!/bin/tclsh

puts "SETTING CONFIGURATION"
#sets the correct database
dbset db mysql
#sets the connection location of MySQL
diset connection mysql_host localhost
#Sets the port of the MySQL connection
diset connection mysql_port 3306
#Sets the root password for MySQL
diset tpcc mysql_pass db-test1
#declares a timed TPC-C test on MySQL
diset tpcc mysql_driver timed
#ramps test up for two minutes before recording TPM
diset tpcc my_rampup 2
#sets test to run for five minutes
diset tpcc my_duration 5

#logging details
#turns on the log file
vuset logtotemp 1
#makes logfile name unique so they aren't overwritten
vuset unique 1
#includes timestamps in log file
vuset timestamps 1
#loads the SQL script to create tables and load values into tables
loadscript
#executes the script
buildschema
#remove the virtual user that built the script
vudestroy


#indicates the testing has started
puts "SEQUENCE STARTED"
#sets the number of users
vuset vu 10
#command to create users
vucreate
#command to run TPC-C benchmark
vurun
```

```
#Indicates the test setup is done
puts "TEST STARTUP COMPLETE, TRANSACTIONS STARTED"
```

# Appendix C: Sample HammerDB Log

```
Hammerdb Log @ Wed Jan 01 20:41:06 CST 2020
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
Timestamp 1 @ Wed Jan 01 20:41:25 CST 2020
Vuser 1:Beginning rampup time of 2 minutes
Timestamp 2 @ Wed Jan 01 20:41:26 CST 2020
Vuser 2:Processing 1000000 transactions with output suppressed...
Timestamp 3 @ Wed Jan 01 20:41:26 CST 2020
Vuser 3:Processing 1000000 transactions with output suppressed...
Timestamp 4 @ Wed Jan 01 20:41:27 CST 2020
Vuser 4:Processing 1000000 transactions with output suppressed...
Timestamp 5 @ Wed Jan 01 20:41:27 CST 2020
Vuser 5:Processing 1000000 transactions with output suppressed...
Timestamp 6 @ Wed Jan 01 20:41:28 CST 2020
Vuser 6:Processing 1000000 transactions with output suppressed...
Timestamp 7 @ Wed Jan 01 20:41:28 CST 2020
Vuser 7:Processing 1000000 transactions with output suppressed...
Timestamp 8 @ Wed Jan 01 20:41:29 CST 2020
Vuser 8:Processing 1000000 transactions with output suppressed...
Timestamp 9 @ Wed Jan 01 20:41:30 CST 2020
Vuser 9:Processing 1000000 transactions with output suppressed...
Timestamp 10 @ Wed Jan 01 20:41:30 CST 2020
Vuser 10:Processing 1000000 transactions with output suppressed...
Timestamp 11 @ Wed Jan 01 20:41:31 CST 2020
Vuser 11:Processing 1000000 transactions with output suppressed...
Timestamp 1 @ Wed Jan 01 20:42:25 CST 2020
Vuser 1:Rampup 1 minutes complete ...
Timestamp 1 @ Wed Jan 01 20:43:25 CST 2020
Vuser 1:Rampup 2 minutes complete ...
Timestamp 1 @ Wed Jan 01 20:43:25 CST 2020
Vuser 1:Rampup complete, Taking start Transaction Count.
Timestamp 1 @ Wed Jan 01 20:43:26 CST 2020
Vuser 1:Timing test period of 5 in minutes
Timestamp 1 @ Wed Jan 01 20:44:26 CST 2020
Vuser 1:1 ...,
Timestamp 1 @ Wed Jan 01 20:45:26 CST 2020
Vuser 1:2 ...,
Timestamp 1 @ Wed Jan 01 20:46:26 CST 2020
Vuser 1:3 ...,
Timestamp 1 @ Wed Jan 01 20:47:26 CST 2020
Vuser 1:4 ...,
Timestamp 6 @ Wed Jan 01 20:47:45 CST 2020
Vuser 6:mysqlexec/db server: Lock wait timeout exceeded; try restarting
transaction
Timestamp 1 @ Wed Jan 01 20:48:26 CST 2020
Vuser 1:5 ...,
Timestamp 1 @ Wed Jan 01 20:48:26 CST 2020
Vuser 1:Test complete, Taking end Transaction Count.
```

```
Timestamp 1 @ Wed Jan 01 20:48:26 CST 2020
Vuser 1:10 Active Virtual Users configured
Timestamp 1 @ Wed Jan 01 20:48:26 CST 2020
Vuser 1:TEST RESULT : System achieved 4849 MySQL TPM at 1606 NOPM
```

# Appendix D: NIH Training Certificate

**Certificate of Completion**

The National Institutes of Health (NIH) Office of Extramural Research certifies that **Eric Brown** successfully completed the NIH Web-based training course "Protecting Human Research Participants".

Date of completion: 09/09/2016.

Certification Number: 2154779.

Appendix E: Virtual Machine Specifications

| | AWS | Azure | Google Cloud |
|---|---|---|---|
| # of CPUs | 1 | 1 | 1 |
| CPU Type | Xeon 2.3 GHz | Xeon 2.4 GHz | Xeon 2.3 GHz |
| CPU Cache | 46080 kb | 30720 kb | 46080 kb |
| Memory | 2043412 kb | 1956444 kb | 2043648 kb |
| OS | Debian 10 | Debian 10 | Debian 10 |
| Hard Drive | 8.3 GB | 32 GB | 11 GB |
| Region | us-east-2c | Central US | us-central1-a |

Appendix F: Final Data for Amazon Web Services

| Time | Users | Buffer Pool | I/O | Trail 1 | Trial 2 | Trial 3 | Mean TPS |
|---|---|---|---|---|---|---|---|
| 10am | 10 | 134217728 | 200 | 19549 | 18662 | 19242 | 319.18 |
| 10am | 10 | 134217728 | 1000 | 19120 | 18307 | 19116 | 314.13 |
| 10am | 10 | 1744830464 | 200 | 18145 | 18796 | 19427 | 313.16 |
| 10am | 10 | 1744830464 | 1000 | 18059 | 18762 | 19363 | 312.13 |
| 10am | 100 | 134217728 | 200 | 18783 | 17895 | 18672 | 307.50 |
| 10am | 100 | 134217728 | 1000 | 17000 | 17963 | 18420 | 296.57 |
| 10am | 100 | 1744830464 | 200 | 17579 | 18252 | 18856 | 303.82 |
| 10am | 100 | 1744830464 | 1000 | 18940 | 18207 | 18881 | 311.27 |
| 10pm | 10 | 134217728 | 200 | 19675 | 19457 | 19133 | 323.69 |
| 10pm | 10 | 134217728 | 1000 | 18870 | 19294 | 18685 | 315.83 |
| 10pm | 10 | 1744830464 | 200 | 19677 | 19976 | 19630 | 329.35 |
| 10pm | 10 | 1744830464 | 1000 | 19632 | 19848 | 19402 | 327.12 |
| 10pm | 100 | 134217728 | 200 | 18027 | 19113 | 19010 | 311.94 |
| 10pm | 100 | 134217728 | 1000 | 18446 | 18531 | 18213 | 306.61 |
| 10pm | 100 | 1744830464 | 200 | 19218 | 19159 | 18965 | 318.57 |
| 10pm | 100 | 1744830464 | 1000 | 18959 | 19210 | 18745 | 316.19 |

Appendix G: Final Data for Microsoft Azure

| Time | Users | Buffer Pool | I/O | Trail 1 | Trial 2 | Trial 3 | Mean TPS |
|---|---|---|---|---|---|---|---|
| 10am | 10 | 134217728 | 200 | 2406 | 2652 | 2729 | 43.26 |
| 10am | 10 | 134217728 | 1000 | 2502 | 2358 | 2412 | 40.40 |
| 10am | 10 | 1744830464 | 200 | 2628 | 2475 | 2679 | 43.23 |
| 10am | 10 | 1744830464 | 1000 | 2556 | 2659 | 2775 | 44.39 |
| 10am | 100 | 134217728 | 200 | 2399 | 2836 | 2530 | 43.14 |
| 10am | 100 | 134217728 | 1000 | 2502 | 2620 | 2310 | 41.29 |
| 10am | 100 | 1744830464 | 200 | 2499 | 2617 | 2464 | 42.11 |
| 10am | 100 | 1744830464 | 1000 | 2547 | 2676 | 2596 | 43.44 |
| 10pm | 10 | 134217728 | 200 | 2645 | 2343 | 2683 | 42.62 |
| 10pm | 10 | 134217728 | 1000 | 2568 | 2230 | 2215 | 38.96 |
| 10pm | 10 | 1744830464 | 200 | 2393 | 2661 | 2417 | 41.51 |
| 10pm | 10 | 1744830464 | 1000 | 2507 | 2571 | 2585 | 42.57 |
| 10pm | 100 | 134217728 | 200 | 2503 | 2589 | 2741 | 43.52 |
| 10pm | 100 | 134217728 | 1000 | 2168 | 2456 | 2190 | 37.86 |
| 10pm | 100 | 1744830464 | 200 | 2396 | 2806 | 2698 | 43.89 |
| 10pm | 100 | 1744830464 | 1000 | 2583 | 2629 | 2701 | 43.96 |

Appendix H: Final Data for Google Cloud

| Time | Users | Buffer Pool | I/O | Trail 1 | Trial 2 | Trial 3 | Mean TPS |
|---|---|---|---|---|---|---|---|
| 10am | 10 | 134217728 | 200 | 16619 | 14864 | 4032 | 197.31 |
| 10am | 10 | 134217728 | 1000 | 12703 | 9654 | 10120 | 180.43 |
| 10am | 10 | 1744830464 | 200 | 13998 | 13227 | 5960 | 184.36 |
| 10am | 10 | 1744830464 | 1000 | 5989 | 14198 | 6052 | 145.77 |
| 10am | 100 | 134217728 | 200 | 15716 | 13952 | 13910 | 242.10 |
| 10am | 100 | 134217728 | 1000 | 3815 | 3569 | 9930 | 96.19 |
| 10am | 100 | 1744830464 | 200 | 13648 | 6077 | 12739 | 180.36 |
| 10am | 100 | 1744830464 | 1000 | 6497 | 5830 | 11851 | 134.32 |
| 10pm | 10 | 134217728 | 200 | 17312 | 14708 | 15439 | 263.66 |
| 10pm | 10 | 134217728 | 1000 | 12128 | 11033 | 10464 | 186.81 |
| 10pm | 10 | 1744830464 | 200 | 15330 | 14429 | 10294 | 222.52 |
| 10pm | 10 | 1744830464 | 1000 | 14819 | 14472 | 6148 | 196.88 |
| 10pm | 100 | 134217728 | 200 | 15915 | 4789 | 13992 | 192.76 |
| 10pm | 100 | 134217728 | 1000 | 11062 | 4391 | 10151 | 142.24 |
| 10pm | 100 | 1744830464 | 200 | 14546 | 6931 | 12924 | 191.12 |
| 10pm | 100 | 1744830464 | 1000 | 6520 | 6077 | 5838 | 102.42 |