

12-2018

# Analysis of SSD's Performance in Database Servers

Venkatesh Kandula  
vkandula@stcloudstate.edu

Follow this and additional works at: [https://repository.stcloudstate.edu/msia\\_etds](https://repository.stcloudstate.edu/msia_etds)

---

## Recommended Citation

Kandula, Venkatesh, "Analysis of SSD's Performance in Database Servers" (2018). *Culminating Projects in Information Assurance*. 71.  
[https://repository.stcloudstate.edu/msia\\_etds/71](https://repository.stcloudstate.edu/msia_etds/71)

This Starred Paper is brought to you for free and open access by the Department of Information Systems at theRepository at St. Cloud State. It has been accepted for inclusion in Culminating Projects in Information Assurance by an authorized administrator of theRepository at St. Cloud State. For more information, please contact [rswexelbaum@stcloudstate.edu](mailto:rswexelbaum@stcloudstate.edu).

**Analysis of SSD's Performance in Database Servers**

by

Venkatesh Kandula

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science in

Information Assurance

December, 2018

Committee members:  
Mark Schmidt, Chairperson  
Lynn Collen  
Balasubramanian Kasi

## Abstract

Data storage is much needed in any type of device and there are multiple mechanisms for data storage which vary from the device to device but at the end it's a magnetic drive which holds the data and stored in the form of digital format. One predominant data storage device is hard disk drive also called as HDD. Hard disk drives are used in a wide range of systems like computers, laptops and netbooks etc., it has magnetic platter which is used for reading and writing operations. (Hard disk drive, n.d.) With the emerging technologies and modularization of web application design architecture created a need for different kind of operating system and system architecture based on the functionality. If we want a server where files need to be placed it should be designed in such a way that it needs to be good at input and output operations (I/O). (How does a hard drive work?, 2018) If we want to store videos and stream, that server should be good at asynchronous streaming functionality. If we need to store the structured/un-structured data which can be pertained to any educational institution or an organization, we can use a database server to store this data in tables and it can be used. In general, we use hard disk drives to store any kind of data in all the servers, but there will be only changes in the system architecture. The concept of HDD utilization has been constant from past 20 years. There was a huge growth in the architectural design of operating systems used for hosting database servers, but when it comes to storage HDD's have been used for many years. With the need for speed and faster operations from the perspective of storage, solid state drives come in to picture. (SSD Advantage, n.d.) They have a different kind of architecture when compared to HDD and they are called as SSD. This paper discusses the idea of using SSD's instead of HDD's in database servers. We created multiple database instances for SSD's and HDD's and also created multiple web applications using JAVA and connected to each of these database servers to access data via REST API's. We have run multiple tests to compare the load time of all the different database instances and generated some visual analytics how it behaves when multiple/series of get operations made on the database with the REST API. This analysis will help in finding if there are any anomalies in the behavior with increase in throughput of read and write operations.

## Table of Contents

	Page
List of Tables .....	5
List of Figures .....	6
Chapter	
I. Introduction .....	9
Introduction .....	9
Problem Statement .....	10
Nature and Significance of the Problem .....	11
Objective of the Study .....	12
Research Questions .....	12
Definition of Terms .....	12
Summary .....	14
II. Background and Literature Review .....	15
Introduction .....	15
Background Related to Problem .....	15
Literature Related to the Problem .....	16
Literature Related to the Methodology .....	23
III. Methodology .....	26
Introduction .....	26
Design of the Study .....	26
Data Collection .....	26
Tools and Techniques .....	27

Chapter	Page
Hardware and Software Environment .....	27
Timeline .....	28
IV. Analysis of Results .....	29
Introduction .....	29
Environment Setup .....	30
Data Presentation .....	35
Data Analysis .....	46
Summary .....	50
V. Introduction, Results, and Conclusion .....	51
Introduction .....	51
Discussion and Results .....	51
Conclusion .....	54
Future Work .....	55
References .....	56
Appendix .....	60

**List of Tables**

Table	Page
1. Comparison of HDD's and SSD's .....	11
2. Solid-State Storage Terms .....	22
3. Details of Databases Configured .....	36
4. Results Obtained by Running Tests on HDD's .....	51
4. Results Obtained by Running Tests on SSD's .....	51

## List of Figures

Figure	Page
1. MySQL client/server model .....	16
2. Client/server model .....	17
3. Internal structure of hard disk drive .....	20
4. Screenshot of option to select type of hard drive when creating a virtual machine in Google Cloud Platform .....	21
5. Simple flash cell design .....	22
6. Blocks in solid state drivers .....	23
7. Ecommerce system interacting with multiple systems to get the data .....	30
8. Status codes representing HTTP responses .....	31
9. HTTP methods and their meaning .....	31
10. Trends showing growth of web APT's .....	32
11. Spring boot adoption trends .....	33
12. REST web service flow .....	34
13. Working of JPA repository .....	35
14. Creating SQL instance .....	37
15. Created all the MySQL instances on GCP as mentioned in Table 3 .....	37
16. Created app ms-reviews using IntelliJ IDE with Spring boot and Java .....	38
17. Configuring application with cloud database details .....	38
18. Created multiple configurations with the same app to hit different databases of different sizes .....	39

Figure	Page
19. Run dashboard showing running apps with different database servers	
back-ends for ms-reviews application .....	39
20. Run dashboard showing running apps with different database servers	
back-ends for e-cart application .....	40
21. Swagger UI with the all operations in e-cart .....	40
22. Swagger-UI for all operations in reviews app .....	41
23. Calling reviews API from e-cart API using REST template for integration .....	41
24. Details of each of the database in the cloud platform .....	42
25. Configurations needed to connect with database in cloud platform .....	42
26. Cloud console to check the table details in cloud platform .....	43
27. Design flow of e-cart application .....	43
28. Design flow of reviews application .....	44
29. Integration of ms-ecart and ms-reviews applications .....	45
30. Dashboard of Jmeter with HDD vs SSD test plan .....	46
31. Creating a HTTP request to hit a web application .....	47
32. Results obtained by sending request to application with 6000 records (HDD) .....	48
33. Results obtained by sending request to application with 6000 records (SSD) .....	48
34. Results obtained by sending request to application with 8000 records (HDD) .....	48
35. Results obtained by sending request to application with 8000 records (SSD) .....	49
36. Results obtained by sending request to application with 10000 records (HDD) .....	49
37. Results obtained by sending request to application with 10000 records (SSD) .....	49
38. Results obtained by sending request to application with 12000 records (HDD) .....	50



Figure	Page
39. Results obtained by sending request to application with 12000 records (SSD) .....	50
40. Bar graph displaying performance differences of HDDs and SDD's .....	53
41. Line chart representing trends in changing of load time of HDD's and SDD's with change in the number of records .....	54

## **Chapter I: Introduction**

### **Introduction**

With the growing IT revolution there was a huge growth in the number of transactions that happened on the internet every day, typically meta-data and actual transactional information about this data is stored in relational databases which has tables containing rows and columns where each row is representing one transaction (What is a database server?, 2018). Increasing in number of transactions on a single database server demands faster read and write operations to the database, because one client tries to do read operation on the database at the same time some other client tries to do write operation which may cause latency issues. Apart from the database server configurations the storage media used for storing data in the form of tables is also reason for latency. To avoid latency and increase speed of read and write operations it is suggested to use SSD in place of HDD based on the business demand.

Usage of SSD's increased a lot in the recent past because of various advantages like performance, durability and power consumptions, etc. (SSD advantage, n.d.) Traditional hard drives have moving components which act as read/write head and whereas solid state drives are made up of static components which reduces latency in seek time. SSD's also have non-volatile behavior in which if the flash memory is made of 3D TLC NAND-based flash memory, this technology is used in case if data persistence is needed even after power loss. Whereas on the other side we have DRAM based SSD's which are capable of providing very fast access, but they need continuous power supply. They also have battery inside to copy data to back-up storage from RAM in case of power failures and this data will be restored back to RAM when power gets restored. DRAM is more expensive than NAND-based flash memory SSD's (SSD advantage, n.d.)

In recent past there was a huge growth in the usage of number of business transactions that happened over internet. It leads to the need of storing of all these huge transactional or business-related information in databases and retrieve them back whenever needed. Traditionally different types of storage media like Microsoft Excel, Microsoft Access etc., was used but with the outburst of data they got shifted to web based applications containing database server as back end to store the data and retrieve them back when needed. To be faster apart from making changes to system level configurations we can also deploy SSD's in database servers instead of HDD's from the storage perspective.

### **Problem Statement**

Performance is a much-needed aspect in any kind of servers to serve continuous requests without any latency, either it is a video streaming server or a database server. There are many ways which we can increase performance of a database like upgrading infrastructure and scaling application etc. In case of upgrading infrastructure, we can itemize it by multiple things like changing CPU capacity, RAM size etc., apart from the other upgrade we can do data storage media from standard disks to solid state drives.

Retrieving data from databases by the applications on demand will become difficult with increase in size of data in the databases. To address this problem one thing we can do is use fast SSD's which will be helpful to retrieve data quickly. Since many databases already have lot of data and we need to find ways to move the data from HDD to SSD safely. This paper discusses what can be achieved with respect to performance of database by migrating storage infrastructure from HDD to SSD by making some read and write operations on MySQL database, which was created on google cloud platform. This paper also demonstrates a comfortable way to migrate storage infrastructure, i.e., HDD to SSD of already existing MySQL database without loss of

data by creating images and setting up a new instance of the image with SSD on google cloud platform.

### Nature and Significance of the Problem

With the huge growth of in the database servers it is tough to do read and write operations to databases. In order to address this problem, there should be some kind of upgrade need to be done in all aspects of system configuration and upgrade of storage media from HDD to SSD is one of the notable upgrade we can make.

The reason behind marking this problem as significant is HDD's can make 3,000 input/output operations per second on a database where as SSD's can make 15,000-40,000. Below table gives clear information about IOPS (Input / Output operation per second) and throughput comparisons between SSD's and HDD's (What is SSD (solid-state drive)? - definition from whatis.com, 2018).

**Table 1:** Comparison of HDD's and SSD's (Storage options, n.d.)

	Standard persistent disks	SSD persistent disks
<b>Maximum sustained IOPS</b>		
Read IOPS per GB	0.75	30
Write IOPS per GB	1.5	30
Read IOPS per instance	3,000	15,000 - 40,000*
Write IOPS per instance	15,000	15,000 - 30,000*
<b>Maximum sustained throughput (MB/s)</b>		
Read throughput per GB	0.12	0.48
Write throughput per GB	0.12	0.48
Read throughput per instance	180	240 - 800*
Write throughput per instance	120	240 - 400*

## Objective of the Study

The main objective of this study is to analyze the components of solid-state drives and what kind of impact they can make when the storage infrastructure in database servers are upgraded to solid state drives from traditional hard disk drives, make some visual analytics with the results obtained by two database servers one with SSD and other with HDD and keeping rest of the configuration same. This study will also give glance about upgrading the storage device of existing database servers from HDD to SSD without loss of any data in it on google cloud platform.

## Research Questions

The study questions include:

1. What made SSD's work faster than HDD's with better performance and low latency?
2. How far performance of database servers can be enhanced by deploying SSD's in them?
3. How to migrate the storage infrastructure of existing database servers from HDD's to SSD's in the context of database server is deployed in Google Cloud Platform?

## Definition of Terms

**Hard disk drive:** This is a kind of storage media which uses magnetic storage mechanism which is used for reading (retrieve) and write (store) are accomplished using a device called platter, which is a rotating disk coated finely with magnetic material. Data access is performed in random-access manner which uses sequential or non-sequential fashion (Hard disk drive, n.d.). These are pre-dominant secondary storage devices and are non-volatile which retains data even power is off.

***Solid state drive:*** Solid state drives are also kind of storage media which is comprised of static devices it has no movable devices like hard drives. It has a circuit board at a base level which is comprised of memory chips. SSD's have very low rate of failure. These are more expensive than HDD's hence these have more performance and less latency when compared to the previous (SSD advantage, n.d.).

***Database sever:*** A database server comprises of both hardware and software together. Typically, they act as a back-end software for a web application in client-server model (What is a database server?, 2018). It can also referred to as a high-end physical computer acts as host for database, it is also called as instance. All the machine level configurations of database server are irrespective relational database or non-relational database hosting on top of it.

***Google Cloud Platform:*** It is a public cloud platform in which google is offering computing services. It provides wide range of services like storage, computing and building applications (Google compute engine documentation | compute engine | Google cloud, 2017). Storage comes under infrastructure as a service, where it provides services like google cloud storage which is used to store large amount of structured and unstructured data. It also provides platform for storing non-relational data as well.

***Throughput:*** It is the measure of how much quantity of information can be processes by a system in a respective time period. It is the measure of speed in which amount of work that can be accomplished in a specific time, this time is also called as response time. It is desired to have high throughput with less response time (SearchNetworking, 2018).

***Latency:*** Latency is the measure time taken to access specific data inside a server or it can be a measure of response time. Low latency is desired in any kind of server or operations (SearchNetworking, 2018).

## **Summary**

In a brief, this chapter discusses about factors that make solid state drives stand out of other storage media. Performance with respect to IOPS (input/output operations per second) and latency issues with respect to response time were discussed here. How these parameters impact the performance of database servers installed with solid state drives. Challenges in upgrading storage infrastructure of database servers from hard disk drives to solid state drives and how we can upgrade storage infrastructure in google cloud platform from traditional hard drives to SSD's. High level description few technical terms stressed in this paper. Finally, we have discussed about throughput and latency parameters as well. In the next chapter we will discuss about challenges with database servers installed with HDD's, Google Cloud Platform and More functional and operational details about components of hard disk drives and solid-state drives.

## **Chapter II: Background and Literature Review**

### **Introduction**

There was a huge growth in the data in the enterprise domain, all this data is stored in database servers and it will be retrieved whenever needed. Growth in quantity of data leads to performance and latency issues. These issues can be handled by upgrading infrastructure of database servers, and the upgrade of storage media from hard disk drive to solid state drive is notable upgrade. One more challenge in this is to avoid any loss of data in case of existing upgrading or old database servers. To find out what made solid state drives work faster than hard disk drives? And why high performance and low latency is desired in database servers? We will examine the internal working structures of both solid-state drives and hard disk drives. This chapter discusses more about operational and functional aspects of components of both disks and need and importance of performance in database servers as well (SSD advantage, n.d.).

### **Background Related to Problem**

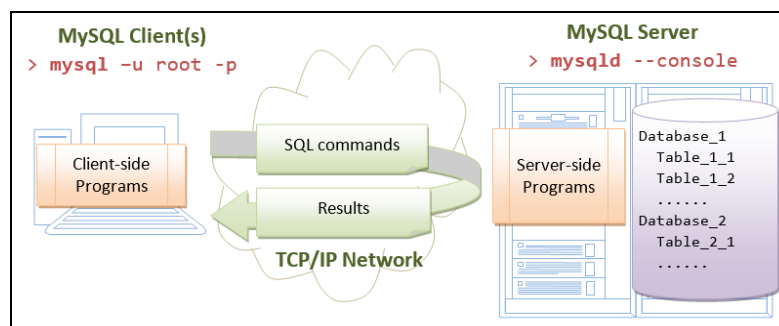
Upgrade of database servers is much needed as data retrieval and storing operations will be challenging. There will be notable increase in the performance and less latency when we upgrade storage from hard disk drive to solid state drive. The reason for high performance in the state drives is they don't have any movable components internally, that's why it is called as solid-state drive. They are built on a top of single circuit board which is comprised of memory chips (What is SSD (solid-state drive)? - definition from whatis.com, 2018). Therefore, going forward we will discuss more about internal components and working of SSD's, HDD's and storage media of database servers.



## Literature Related to the Problem

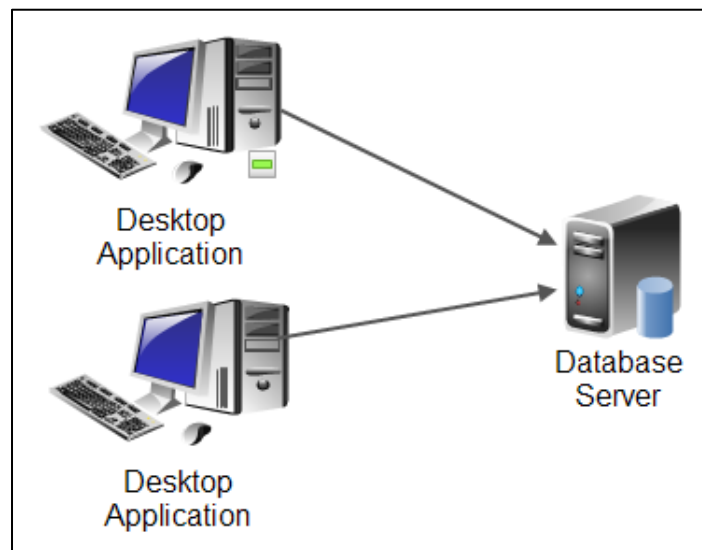
**Database servers:** It can be considered as a warehouse where a website saves and process data for different kinds of operations like generating analytics, populating dashboards and pulling reports etc. Typically, it can be considered as a system that is connected to internet which can be used for information storage or retrieval. In any client server application when a request is sent by any web client to database server and response is sent back to the client. In general database servers have dedicated operating system which is more compatible with read and write operations. Server configurations are based on the range of operations for which database server was deployed. But the database server is irrespective of kind of database has been deployed on top of the database server. Apart from data, we can also store files/images in a relational database in the form BLOB storage but for better performance it is advised to use dedicated file server (What is a database server?, 2018).

Database server takes SQL requests from web client and executes those requests on top of database like MySQL residing inside of the server. For a better performance, multiple instances of database can be used and at the same time scaling can be done by increasing multiple instances of the databases with growth request throughput to the server. Database requests can be done in multiple ways through triggers, functions and database queries.



**Figure 1:** MySQL client/server model (MySQL client/server model, n.d.)

We can use security functions as well inside which can be used to limited access to specific data inside the database (What is a database server?, 2018). There are some kind of database servers where we can use them as centralized database servers which can be used for identity management, where we can store credentials for giving access to websites, if we consider a use case of university which has multiple applications like course registrations, learning management system, student enrollment and student worker timesheet submission etc. Since all these applications are related to similar stakeholders, they can use same credentials to log in to all these applications and log in platform of this kind of systems is called as single sign on system. Where we can store login credentials in a single database server called as identity server and authorization can be done by validating data with this identity server. Since authentication to the applications will happen using the same identity server, this server should work with high performance and low latency. For this server to work efficiently, it should be deployed with high system configurations and apart from this data storage needed to be upgraded from hard disk drive to solid state drive.



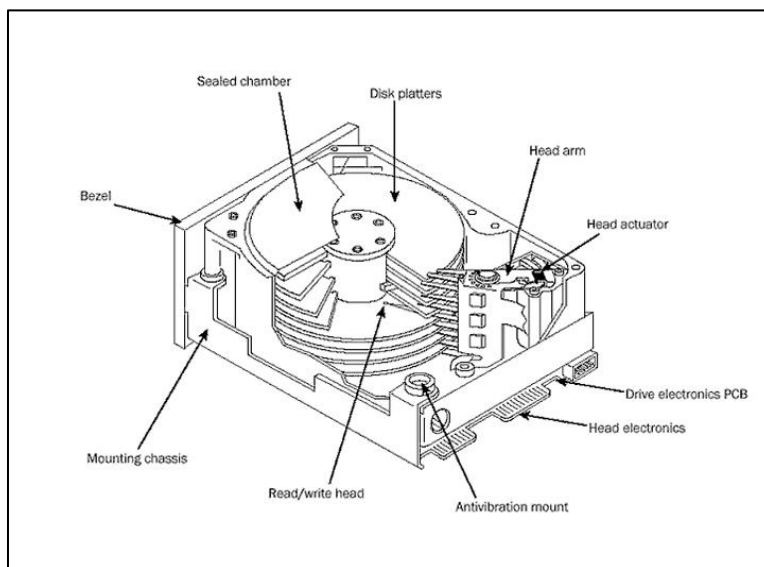
**Figure 2:** Client/server model (Software-architecture, n.d.)

**Hard disk:** A hard drive is a storage media which uses magnetic mechanism to store data/ information. Ideally it is a secondary storage media whereas RAM (Random Access Memory) is a primary storage memory (Hard disk drive, n.d.). Today there are wide range of hard drives available with wide range variants with speed, size, etc., and the very first hard drive was introduced by IBM in 1956 which was named as RAMAC. RAMAC stands for Random Access Method of Accounting and Control, the size of RAMAC about two refrigerators. Since it is big to operate it almost need entire room. Amount of data the very first RMAC can hold is 5 megabytes of data and IBM used to lease this storage systems for a price \$3200 per month. Later IBM introduced “Disk Packs” that increase capacity and decrease size to the extent they can, these were plugged with IBM main frame computers and the size of platters in this device was reduced from 24 inches to 14 inches. Each disk pack can hold about a size of 2 MB of data and it weighs about 9-pounds. Then after few years IBM introduced concept of inter connect which allows compatibility of storage systems of other vendors and then the outburst of hard disk manufacturing started by wide range of vendors.

Computers were mostly used by people or organizations who can afford them before 1970s. After that in early 1970s personal computers were released into market, with release of personal computers into consumer market there was a huge need in microchips which are needed to build systems and then slowly the prices got dropped. But the hard drive capacity of these computers is much less. Then in 1980s a startup company introduced hard drives about a size of 5 MB for a price of \$1500 which can be fit into any kind of personal computers and the name of this company is Shugart Technology, which was later changed to Seagate and Seagate well known in storage systems market till today.

**RAID technology:** This concept was introduced in SIGMOD conference by few scientists from U.C. Berkley. The full form of RAID is “Redundant Array of Inexpensive Disks”. Main concept behind development of RAID is dividing read and write components into separate logical units so that a single logical unit can work productively with loose coupling and this makes data operations faster. There will be very less possibility of loss of data as well (What is RAID (redundant array of independent disks)?, 2018). RAID employs the concept of data mirroring and data stripping. Mirroring helps in copying the data from one disk to other disk and stripping is the concept of dividing hard drive into small parts called as sectors typically each sector has a size of 512 bytes. There will be RAID Controller between operating system and physical hard disk. RAID controller helps in protecting data from system crashes and increases performance as well.

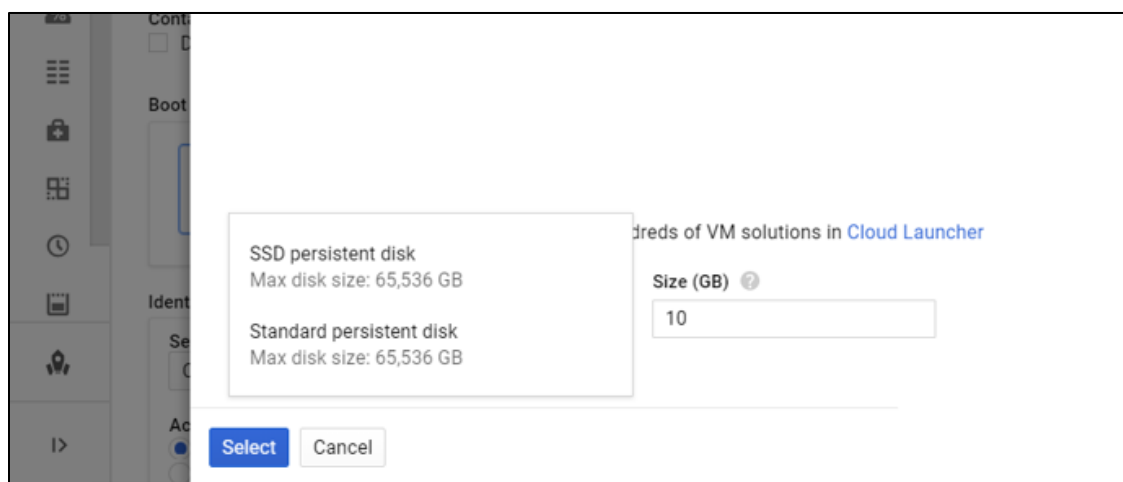
**Working of hard disk drive:** A hard drive is a sealed chamber which contains a spindle in middle of the sealed pack which is surrounded by disk platters. There will be a motor which is used to rotate the disks and there will be a second motor which is used to handle read and write head that stores and read data from the tracks of each platters. It has an actuator which will act as a stepper motor rotating read-write heads. Anti-vibration mount can be used to protect and avoid contact of hard drive from other components. Bezel is a cover which can be used to protect the chassis it acts as a mechanism to protect. There are few cases where entire hard drive is cover by a cabinet and few bezels accompanied with LED's which will blink in cases when hard drive is in use (How does a hard drive work?, 2018). Platters rotate with a speed of 4500 to 7200 rpm and the parameter used to measure disk access time is milliseconds. Physical location of a magnetic disk is located using cylinder, track and sector locations and logical address block can be used for address mapping.



**Figure 3:** Internal structure of hard disk drive (Internal structure of hard disk drive, 2017)

**Solid state drive:** A solid state drive stores data in solid-state flash memory and it is a non-volatile form of storage. Solid state drives will not have any movable components inside it, on the other side hard disk drive has a spindle and read/write head which handled using actuator. Solid state drives are comprised of an array of semiconductor memory organized in the form of disk using integrated circuits. Since it has no movable components and it comprises of array of semiconductors hence it is called as a solid state drive. The rapid growth in development and using solid state drives is need for high read/write performance. Apart from this SSD's low latency in read access, the reason behind this is SSD's use flash memory to store data and it allows reading data directly from flash SSD cell location. In recent past many organizations adopting solid state drives technology in their infrastructure and it includes servers with high-performance and availability, laptops and any other kind of storage infrastructure (What is SSD (solid-state drive)? - definition from whatis.com, 2018).

The features SSD's make it suitable for building database servers as well which need to serve lot of read/write operations in parallel. This technology is widely adopted and provided by cloud services as well, when we configure a virtual machine in a cloud infrastructure it provides option to us whether to create a virtual machine with hard disk drive or solid-state drive. There are wide range of cloud service providers who provide solid state drive infrastructure in the market like a\Amazon Web Services, Google Cloud Platform and Rackspace etc.,



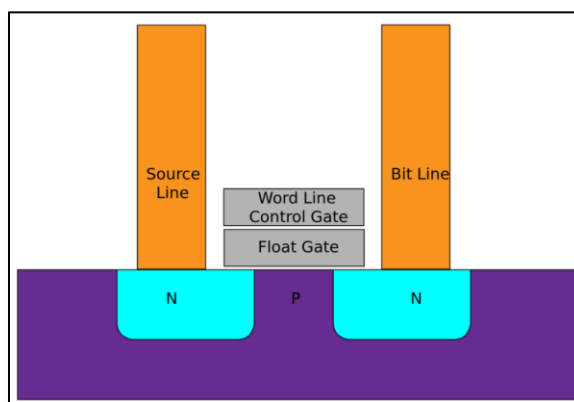
**Figure 4:** Screenshot of option to select type of hard drive when creating a virtual machine in Google Cloud Platform

**NAND flash cell:** Solid state drives comes with three form factors which makes easier for organizations to quickly flip from hard drives to solid state drives, it includes SSD can fit into the same HDD slots. SSD's use NAND flash memory concept which use 8-piece pin to access the data. SSD's has NAND flash memory containing single-level cell or multi-level cell and it stores 1-bit of data per cell and the recent solid drives are triple-level NAND flash cells (Zambelli, Micheloni, & Luca, 2018).

**Table 2:** Solid-State Storage Terms

Single-level Cell (SLC)	This is a type NAND flash chip that store single bit of data in single chip cell. It is one of the fastest, highly reliable, most expensive and long-lasting kind of NAND flash memory
Multi-level Cell (MLC)	In this type of NAND flash chip, it stores two bits in a chip cell. It is not long lasting as SLC, slower and less expensive.
Enterprise Multi-level cell (eMLC)	It is sophisticated version of Multi-level Cell (MLC) it has a controller and software inside which is used to overcome some problems of MLC. In enterprise eMLC is highly recommended in database servers, application servers etc.

**Working of solid-state drive:** In hard disk drives data is stored in movable disks called spinning disks, whereas in solid state drives data is stored in pool of NAND flash drives. These NAND flashes are made up of gate transistors and these flashes are designed to retain charges even if power supply is not available.

**Figure 5:** Simple flash cell design (Simple flash cell design, n.d.)

Above diagram illustrates design of simple flash cell, electrons are stored in float gate whereas 0 means charged and 1 means non-charged it different from what we think about electron charges in general whereas 0 means not charged and 1 as charged. All these flash cells are arranged on top of a grid and this grid is called as a block and there is other parameter called as page which is a row of a block. The size of pages varies and available as 2K, 4K, 8K, and 16K, etc., and the size block depends on the size of pages. In general, the size of block typically varies between 256KB to 4MB. Read and write operations works at page level in a grid and whereas overwrite works at block level, Hence, it requires high level voltage to remove the data at block level. While writing to a single cell in a page it copies entire page to a buffer and update the page again with old content and new content (Hu, 2012).



**Figure 6:** Blocks in solid state drives (Structure of SSD block , 2015)

### Literature Related to the Methodology

The problem statement of this paper is to determine how performance of database server increased with upgrade of storage infrastructure from hard disk drive to solid state drive. This paper illustrates how to upgrade the storage infrastructure of existing database servers from hard



drive to solid state drive in the context of Google Cloud Platform (Google compute engine documentation | compute engine | Google Cloud, 2017).

**Functional analysis of database servers with hard drive and with solid state drive:**

Setting up of database servers on Google Cloud Environment can be done in multiple ways, we can use different services provided by Google to setup working environment for this study. This can be done by creating a virtual machine using Google computing engine which is a service provided by Google Cloud through its service model Infrastructure as a Service, we need to create a virtual machine on Google Cloud Platform and install MySQL as a service in it manually (How to set up MySQL on Google compute engine | solutions | Google Cloud, 2017). Google compute engine offers wide range of options to scale and manage performance of these virtual machines. Google cloud provides another database service called Cloud SQL which is a database service managed end-to-end by Google (Khan & Jan, 2011). It is easy to setup and maintain using simple dashboards, when we are setting up a Cloud SQL service on Cloud it will ask for type of database we want. It offers two database services one is MySQL service and PostgreSQL, for this study we need to select MySQL database service (Cloud SQL | Cloud SQL | Google Cloud, 2017). Other way we can deploy infrastructure needed for our study is to use Google Cloud Launcher, it is an amazing service offered google cloud, where we will get complete setup needed for the study as a single package. We just need to give type of storage we need and capacity of hard drive rest all will be managed by the system itself.

**Which service need to be chosen:** Cloud SQL service provides wide range of features apart from database service, other features includes maintaining timely backups, auto scaling and maintenance etc., but the limitations here are we cannot setup SUPER users on this and we can't even use user-defined functionalities. If we want a custom image of MySQL which is provided

by third party, we can use google compute engine to deploy the custom service and do our study.

Third option is we can use google cloud launcher which installs MySQL service as soon as we deploy virtual machine using cloud launcher (Google compute engine documentation | compute engine | Google Cloud, 2017).

## **Chapter III: Methodology**

### **Introduction**

In this chapter we will briefly discussing about how research will be done and what are the steps need to be taken to find how performance metrics change by flipping SSD and HDD in database servers and find if are any anomalies and saturation points in the behavior, setting up google cloud platform as well. Apart from we will be discussing hardware and software components needed to accomplish this research.

### **Design of the Study**

The standard way to start a research is to adopt a specific framework which be either qualitative or quantitative. Sometimes it can be hybrid which involves both, we are using quantitative framework for this research. The goal of the project is to setup two database servers one with solid state drive and other hard drive, maintaining rest of the configuration same. Taking results of performance or response time of database operation in both servers and comparing them (Wiseman, n.d.). Analyzing the results and check when the performance actually changes and this research helps to check if a specific has to adopt solid state drive or not in their database servers. This research performed in cloud infrastructure: google cloud platform. We will be working on migrating storage infrastructure to solid state drive from hard disk drive as well, which is useful to flip from hard drive to solid state drive when the application is busy and we can flip back to hard drive when the application traffic less. In this way money spent on infrastructure can be saved.

### **Data Collection**

In this research we use couple of database servers deployed on google cloud platform one with hard drive and the other with solid state drive with maintaining same configurations in both

the server instances. Once setup servers got completed, MySQL database need to be installed in both servers. Google Cloud Platform is an incentive of Google, Inc., which provides wide range of cloud services on-demand. We will perform multiple database operations (read/write) and both the database servers and take results for multiple throughputs and compare the latencies.

### **Tools and Techniques**

Once data collected by following data collection process mentioned in the above, we will get the dataset and thus dataset is fed to data analysis tool called Tableau. Tableau has numerous special and outstanding features. Its effective information disclosure and investigation application enables you to answer important inquiries in seconds. You can utilize Tableau's intuitive interface to visualize data, investigate datasets perspectives, and even consolidate various databases effectively. It doesn't require any complicated scripting.

### **Hardware and Software Environment**

This research involves setting up database servers on top of google cloud platform and we need to install MySQL database software in those servers.

#### **Software requirements:**

- Linux operating system
- MySQL database software
- Tableau reporting tool
- Google Cloud Platform Console

**Hardware requirements:** Since we are using google cloud platform there is no need of physical solid state drive and hard drive we just need a laptop or desktop to login into google cloud platform and manage research infrastructure.

**Work in progress:** In this chapter we will be discussing about the effort needed for successful completion of implementation aspect of the research. In the previous chapters from the beginning discusses about the brief and high-level idea what is going to be implemented at the end of research and then in literature survey we have identified what are all the concepts needed to be studied in depth and information need to gathered is mentioned, followed by implementation details in theory was discussed. In the coming chapters we will be discussing more about the practical implementations of theory discussed in the previous chapters.

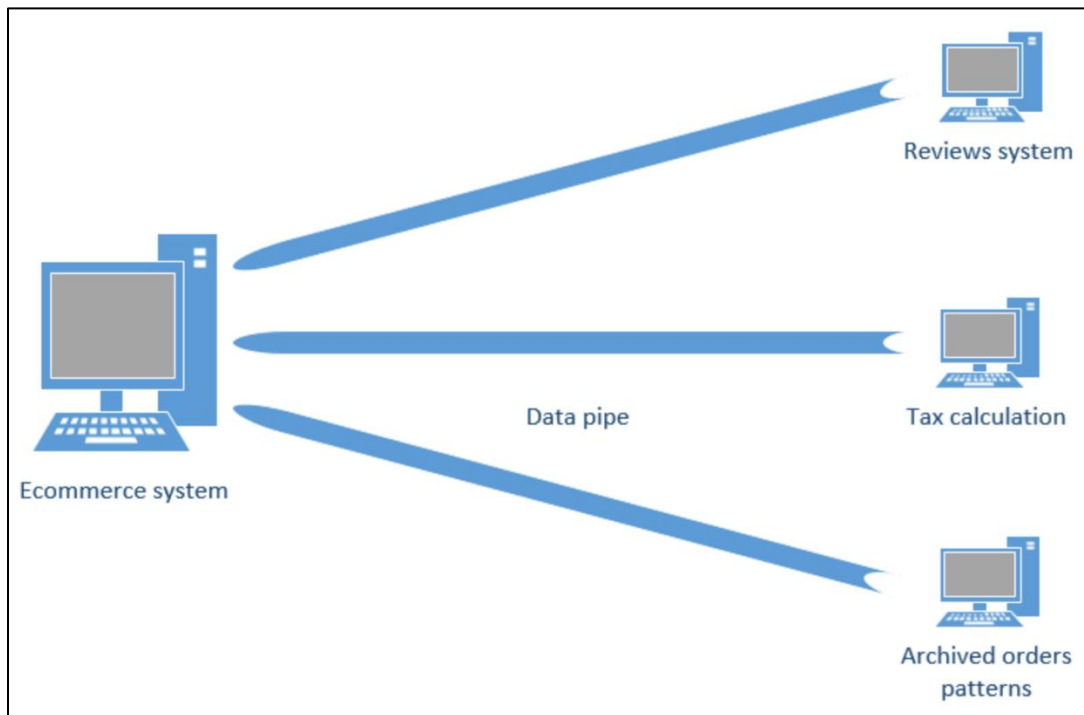
## Chapter IV: Analysis of Results

### Introduction

In this chapter we will be discussing about the use case of the implementation, setting up multiple database servers with data sets of different sizes, collection data which need to be stored in the database servers. On the other side building web applications to use database servers with different sizes of datasets and finding the load time for the each of the different datasets using Postman for only one user and Jmeter for mimicking multiple users at the same time and check the performance scenarios with putting up much load (multiple users trying to access the data at same time).

The use case we have chosen is a part of ecommerce system where the main ecommerce head is fetching the reviews from some other system. Most of the ecommerce sites pull reviews from third parties who maintain reviews of all the products and give the data based on the API call which include product details, ecommerce site will be having product details and it will call the reviews system with the product details and get the reviews and aggregate the data and provide it to the user.

It looks like single system but ideally in the back end there will be multiple systems like tax calculation, inventory management, and reviews handling etc., behind scenes. We have used swagger UI which is an open source API documentation tool for testing the entire system and developed small application using spring boot and java. Multiple instances of same applications configured and integrated with multiple database systems with difference in the size of datasets by changing the properties of database servers. Applications were set up in the local environment and database servers were installed on the google cloud platform.



**Figure 7:** Ecommerce system interacting with multiple systems to get the data

### Environment Setup

As shown in the figure above we have considered a use case where an ecommerce system will pull data from multiple systems from the back end and aggregate together to make a complete end to end system. If any of the data pipe get slow down entire call will get delayed and hence these systems need to be built in such a way that they should ensure high availability and security as well since they will be fulfilling requests from multiple clients. This platform was platform using java and spring boot framework, Jmeter tool was used to search for the performance testing and to data collected will analyzed using tableau software. All the individual systems in the use case were built as REST API's and the communication among them was built using REST template.

**REST:** REST stands for representational state transfer and it is architecture using which web services are built. Web services are used in connecting one system with other system. It is

stateless mechanism. These days REST is used in large scale because of it is light weight, easy to maintain and highly scalable (Vasyl, 2016). These services can be used by third parties or any system using authentication like using api keys, oAuth mechanism, etc., and the usage can be captured and monetized using proxy layers or api gateways on top of API's. Request to REST Api can be send either xml or json format we have used json format in this use case. Api will respond with HTTP status codes and responses. REST Api's have multiple methods which can be used for communication which includes get, post, put, delete, patch and these operations will respond using response body and HTTP status codes based on the type of operation.

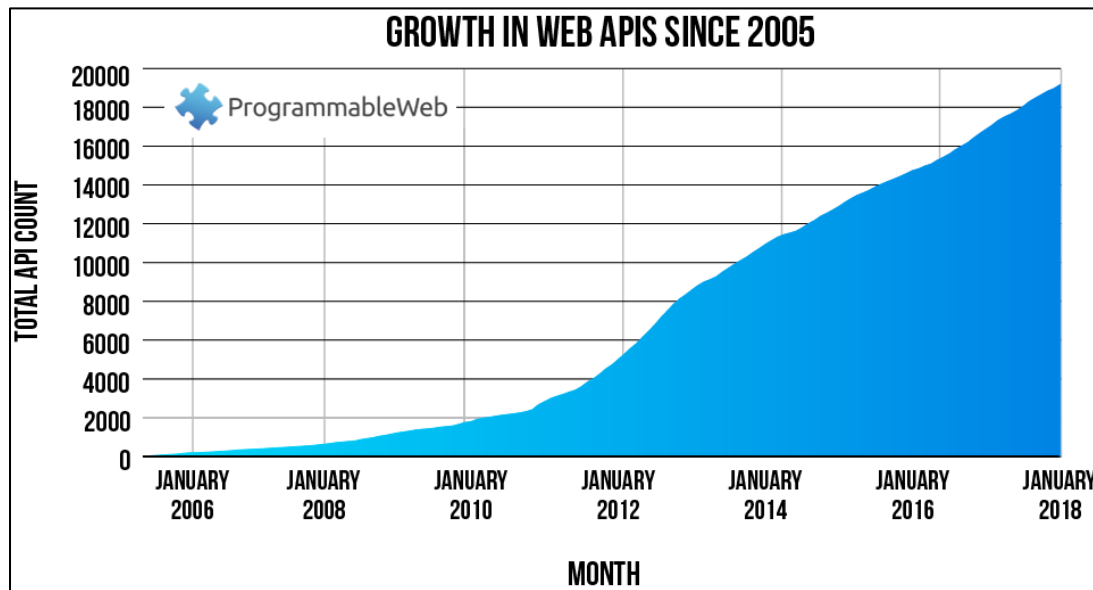
HTTP Status Codes		
<b>Level 200 (Success)</b> 200 : OK 201 : Created 203 : Non-Authoritative Information 204 : No Content	<b>Level 400</b> 400 : Bad Request 401 : Unauthorized 403 : Forbidden 404 : Not Found 409 : Conflict	<b>Level 500</b> 500 : Internal Server Error 503 : Service Unavailable 501 : Not Implemented 504 : Gateway Timeout 599 : Network timeout 502 : Bad Gateway

**Figure 8:** Status codes representing HTTP responses (Working with HTTP status codes, 2017)

Command	Meaning
GET	Return the requested item
HEAD	Request only the header information of an item
OPTIONS	Request communications options of an item
POST	Supply input to a server-side command and return the result
PUT	Store an item on the server
DELETE	Delete an item on the server
TRACE	Trace server communication

**Figure 9:** HTTP methods and their meaning (Server-sandbox, n.d.)





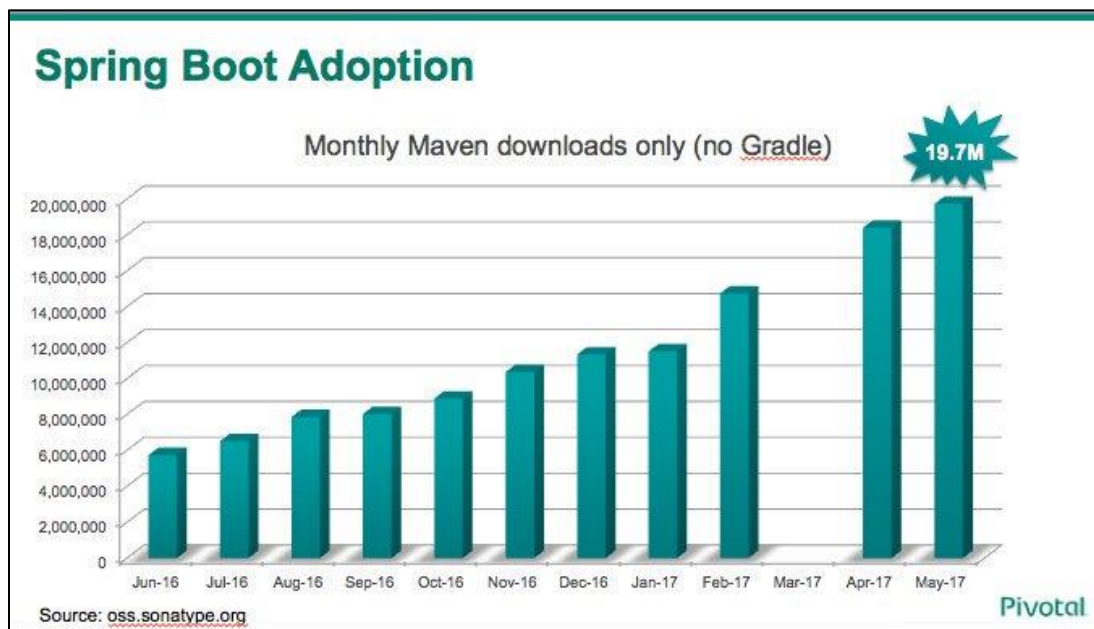
**Figure 10:** Trends showing growth of web API's (Growth in web api's since 2005, n.d.)

These days there was a huge growth in the usage of api's in any kind of domain. In the use case all the four systems were built as a REST api's and will be communicated using REST template and data will be exchanged in JSON format for individual system API documentation open-source swagger was used.

**Spring boot:** Spring boot is a java-based framework and it is open source as well. It is highly used in micro service development using REST and Java. There was huge growth in the adoption of spring boot in micro service development by organization because it is light weight nature and loose coupling (Spring boot tutorial, 2018). It has an inbuilt web server within the framework which avoids the extra over head of maintaining web server infrastructure. Its compatibility with spring cloud framework and other cloud infrastructure helps a lot in developing cloud native web applications.

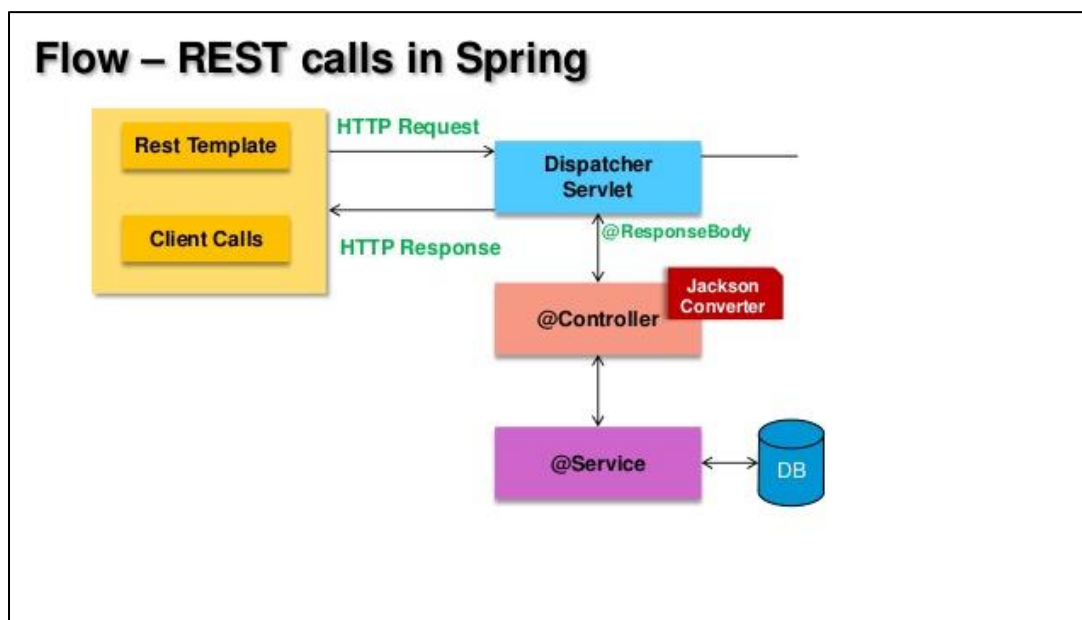
***Advantages of spring boot:*** It made web development easy by automating the lot of common things in the programming development and reduces the amount of time needed for building applications.

- It has lot of annotations which can be helpful in reducing lot of boilerplate code
- Spring boot has lot of compatibility with other frameworks from spring, which helps a lot during integration of multiple systems.
- It has lot of plugins which helps in increase the productivity of developers.
- It has an embedded server and in-built databases like H2 which makes configurations very easy.
- Only drawback with spring boot is it is bit confusing and complicated while migrating a legacy old project to spring boot, all the configurations need to be look into carefully. But new projects can be started very easily by using spring initializer.



**Figure 11:** Spring boot adoption trends (Trends showing adoption of Spring boot, n.d.)

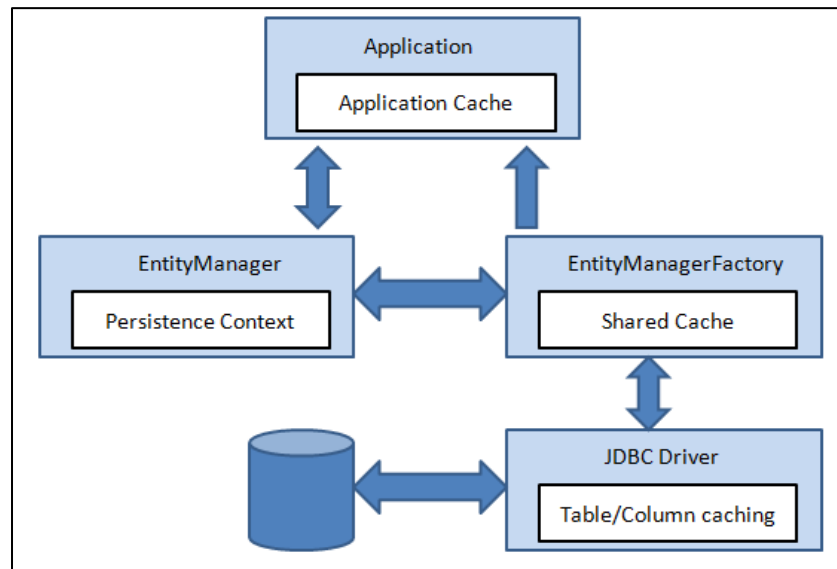
**Rest template:** Rest Template is a one of the component from spring-web and it can be used for communicating among restful web services. During this call authentication need to be included which can be used for authentication during API calls. It will return a response entity which contains HTTP status and body contains the response body based on the method of the API call (How to use spring resttemplate client for consuming restful webservice, 2018). In some cases, response will not be there based on the architecture or design of application. In such cases just call the api and forget. Proper logging needs to be implemented to track failures in the communication.



**Figure 12:** REST web service flow (Rest presentation documents, n.d.)

**Java and JPA:** Java is a high-level programming language used for developing enterprise-level web applications. There are multiple configurations available in java like J2SE, J2ME, and J2EE which is used based on the business need. It has one of best feature write once and run anywhere. Features of java include object oriented, platform independent, secure, robust, portable, multi-threaded, high performance, and distributed. We have used J2EE to develop our

applications mentioned in the use case. Java web applications will run inside a web container, which is configured inside a web server (Master microservices with spring boot and spring cloud, 2017). We have used JPA (Java Persistence API) for mapping database tables with java business objects and our databases deployed in google cloud platform. JPA has wide range of annotations needed for configuring java applications with database servers in declarative style. Java objects pertaining database objects are called as entities and each field in the entity represents a column in the table. This kind of mapping between java objects and database tables is called as object-relational mapping.



**Figure 13:** Working of JPA repository (Data caching in JPA, n.d.)

### Data Presentation

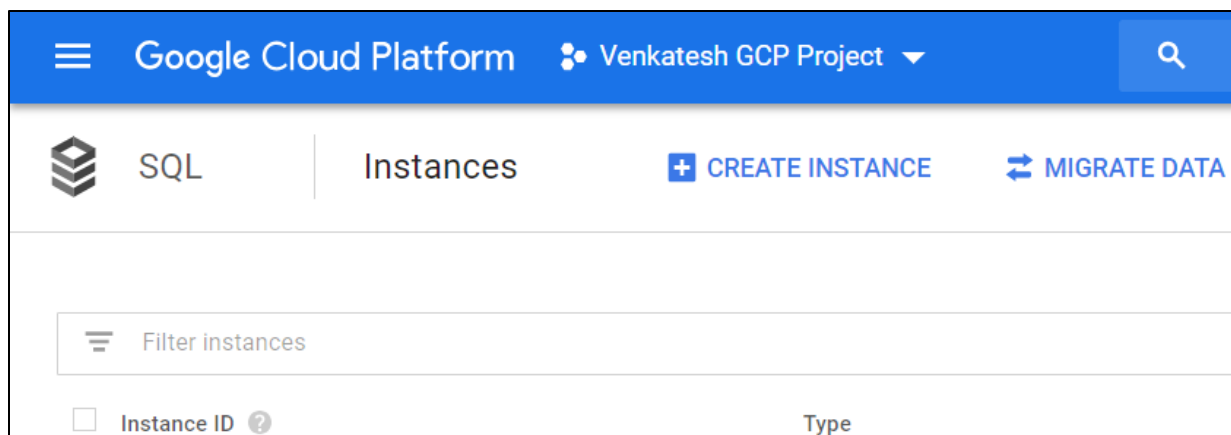
Around 12,000 records containing review details of different products has been collected and inserted into database. Different database instances have been setup on the cloud with 6,000, 8,000, 10,000, 12,000 records in the reviews table of each of the instance. Similar kind of instances has been built with same configuration with only change as storage mechanism. At the same time 2 different database instances with hard disk drive and solid-state drive has been set

for e-cart system as well, which is the main head of the application. Infrastructure details related to the use case were mentioned below. After setting up the instances on the google cloud platform. All the instances have been started and records has been inserted to the each of the instances using insert database scripts. Below table states the infrastructure details related to database instances.

**Table 3:** Details of Databases Configured

Database instance name	Number of records	Description
reviews-hdd-db-6000	6000	Instance with hard disk drive and reviews table containing 6000 records.
reviews-hdd-db-8000	8000	Instance with hard disk drive and reviews table containing 8000 records.
reviews-hdd-db-10000	10000	Instance with hard disk drive and reviews table containing 10000 records.
reviews-hdd-db-12000	12000	Instance with hard disk drive and reviews table containing 12000 records.
reviews-ssd-db-6000	6000	Instance with solid state drive and reviews table containing 6000 records.
reviews-ssd-db-8000	8000	Instance with solid state drive and reviews table containing 8000 records.
reviews-ssd-db-10000	10000	Instance with solid state drive and reviews table containing 10000 records.
reviews-ssd-db-12000	12000	Instance with solid state drive and reviews table containing 12000 records.
ecart-hdd-db	NA	Instance with ecart basic tables and storage as hard disk drive
ecart-ssd-db	NA	Instance with ecart basic tables and storage as solid state drive

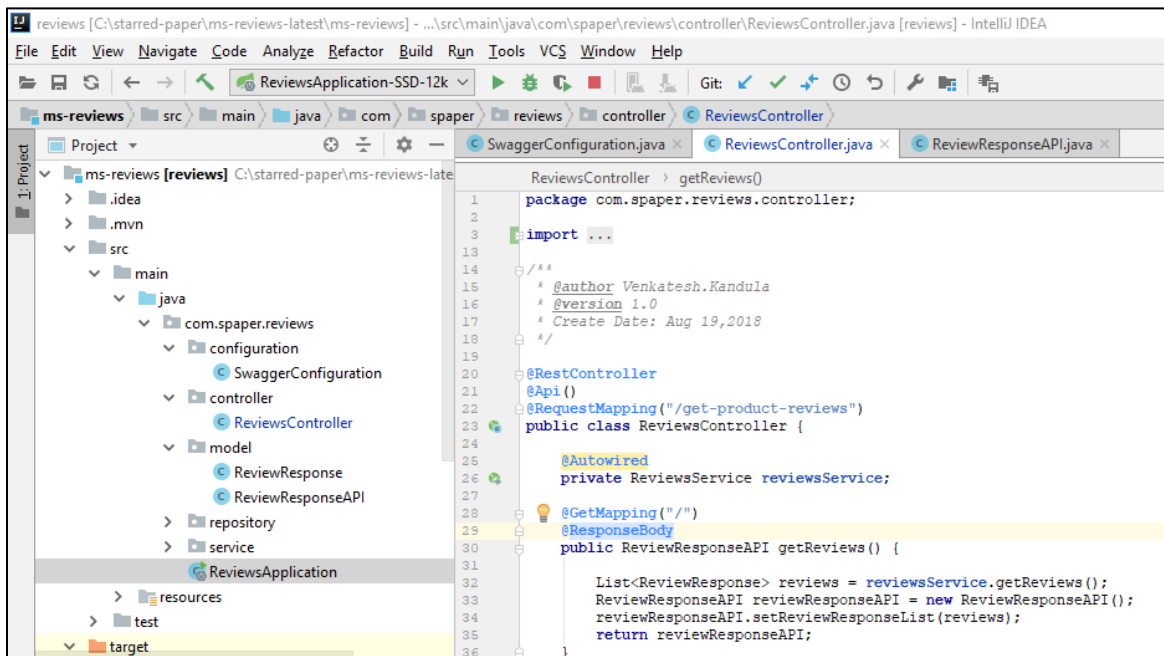
*Setting up databases on cloud environment:*



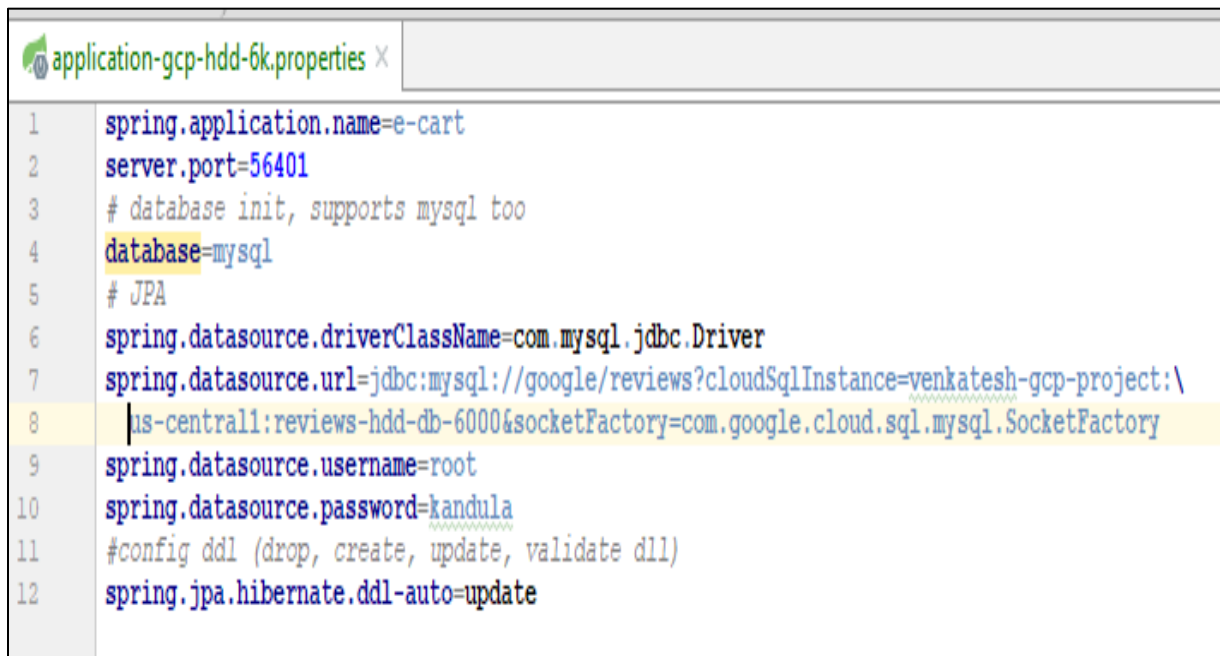
**Figure 14:** Creating SQL instance

Google Cloud Platform Venkatesh GCP Project			
SQL	Instances	CREATE INSTANCE	MIGRATE DATA
Filter instances			
Instance ID	Type	High availability	
<input type="checkbox"/> <input checked="" type="checkbox"/> ecart-hdd-db	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> ecart-ssd-db	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> reviews-hdd-db	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> reviews-hdd-db-10000	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> reviews-hdd-db-6000	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> reviews-hdd-db-8000	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> reviews-ssd-db	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> reviews-ssd-db-10000	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> reviews-ssd-db-6000	MySQL 2nd Gen 5.7	Add	
<input type="checkbox"/> <input checked="" type="checkbox"/> reviews-ssd-db-8000	MySQL 2nd Gen 5.7	Add	

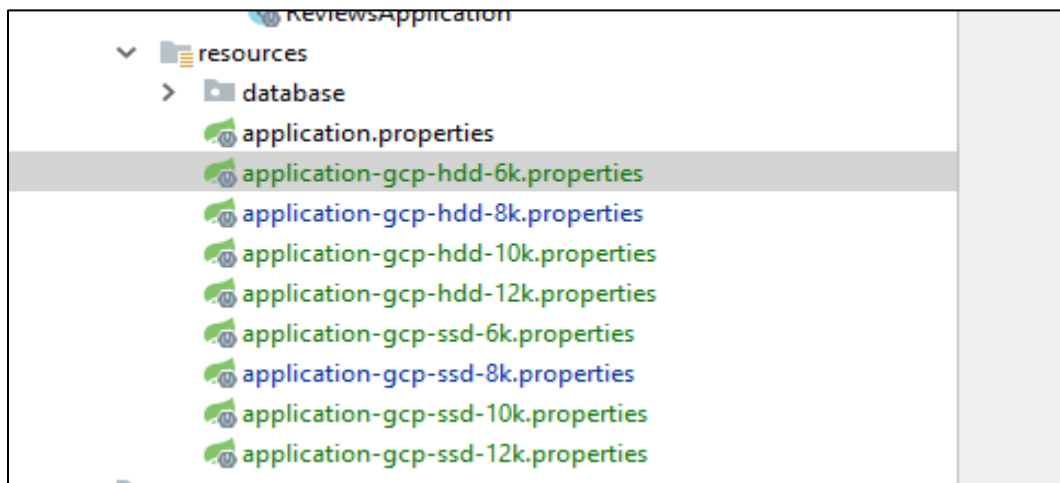
**Figure 15:** Created all the MySQL instances on GCP as mentioned in Table 3



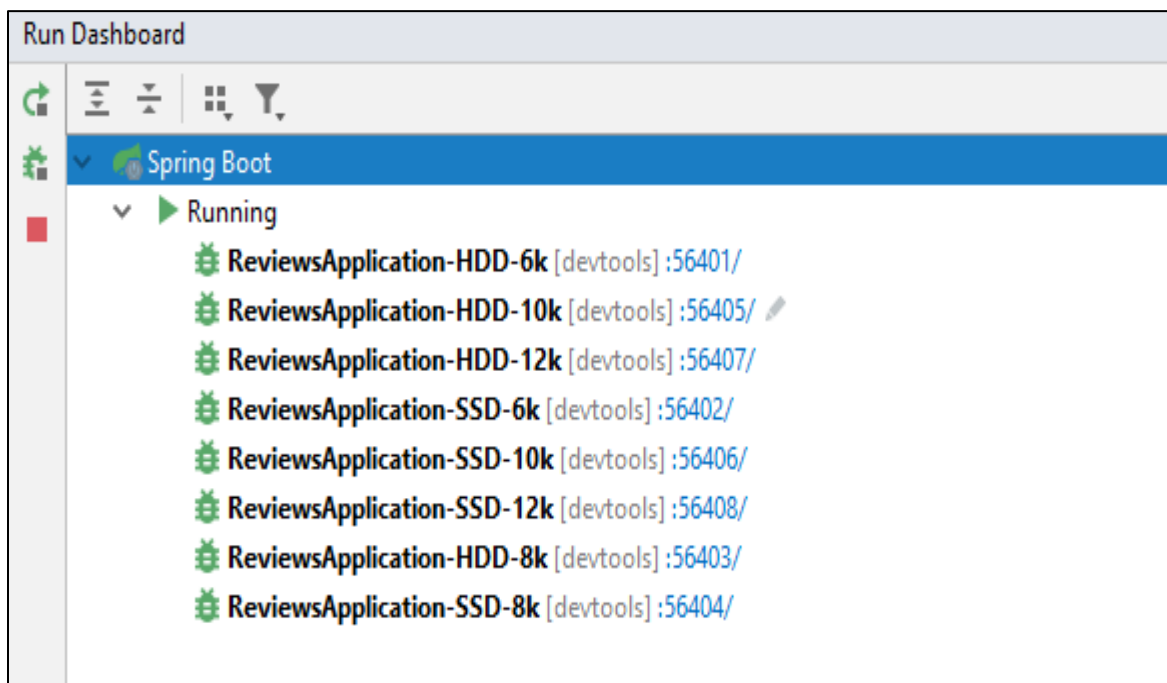
**Figure 16:** Created app ms-reviews using IntelliJ IDE with Spring boot and Java



**Figure 17:** Configuring application with cloud database details

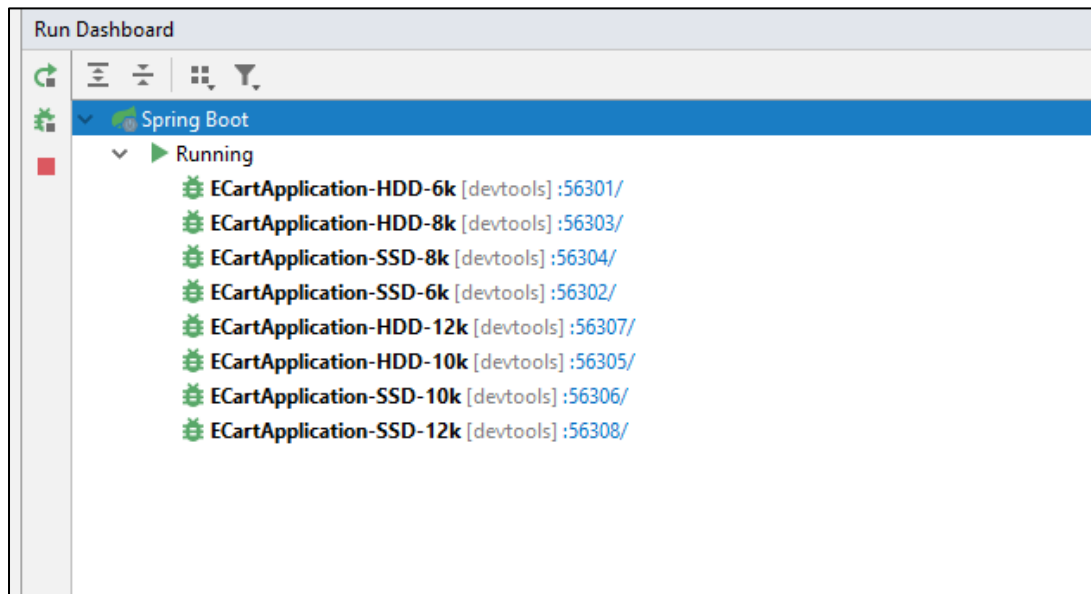


**Figure 18:** Created multiple configurations with the same app to hit different databases of different sizes

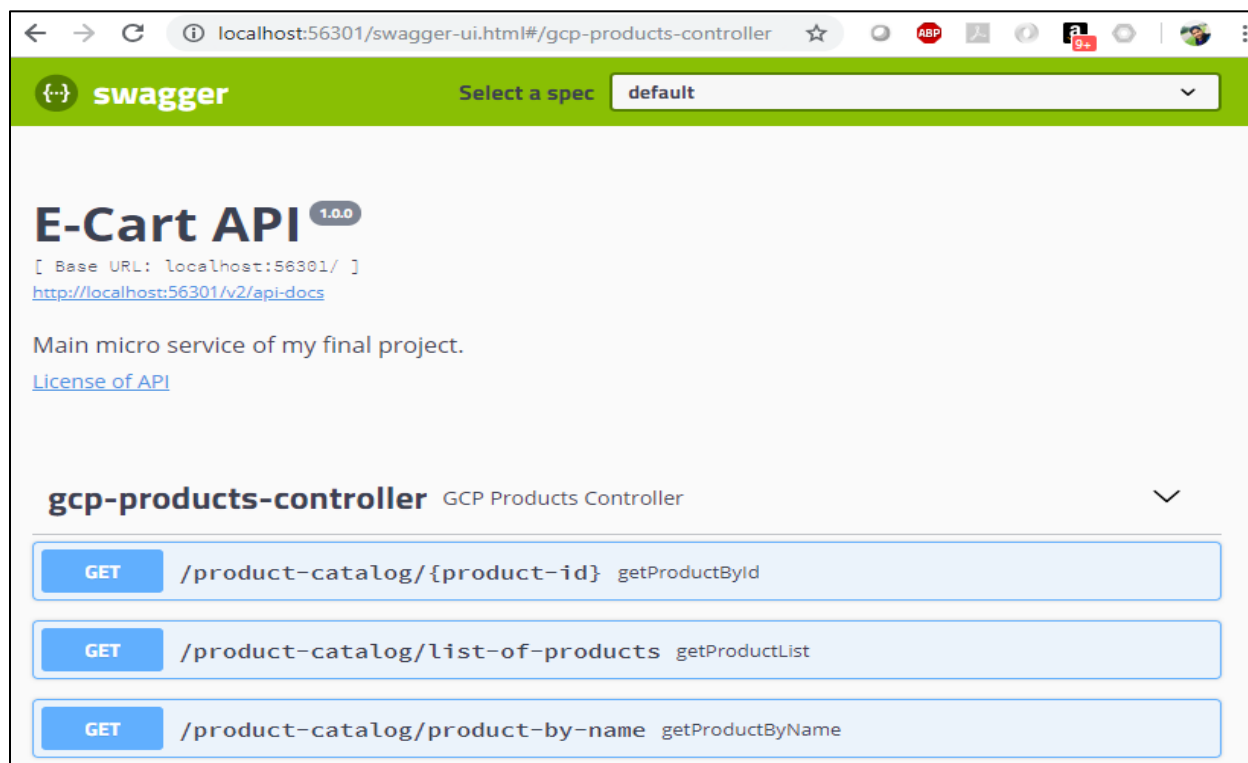


**Figure 19:** Run dashboard showing running apps with different database servers back-ends for ms-reviews application

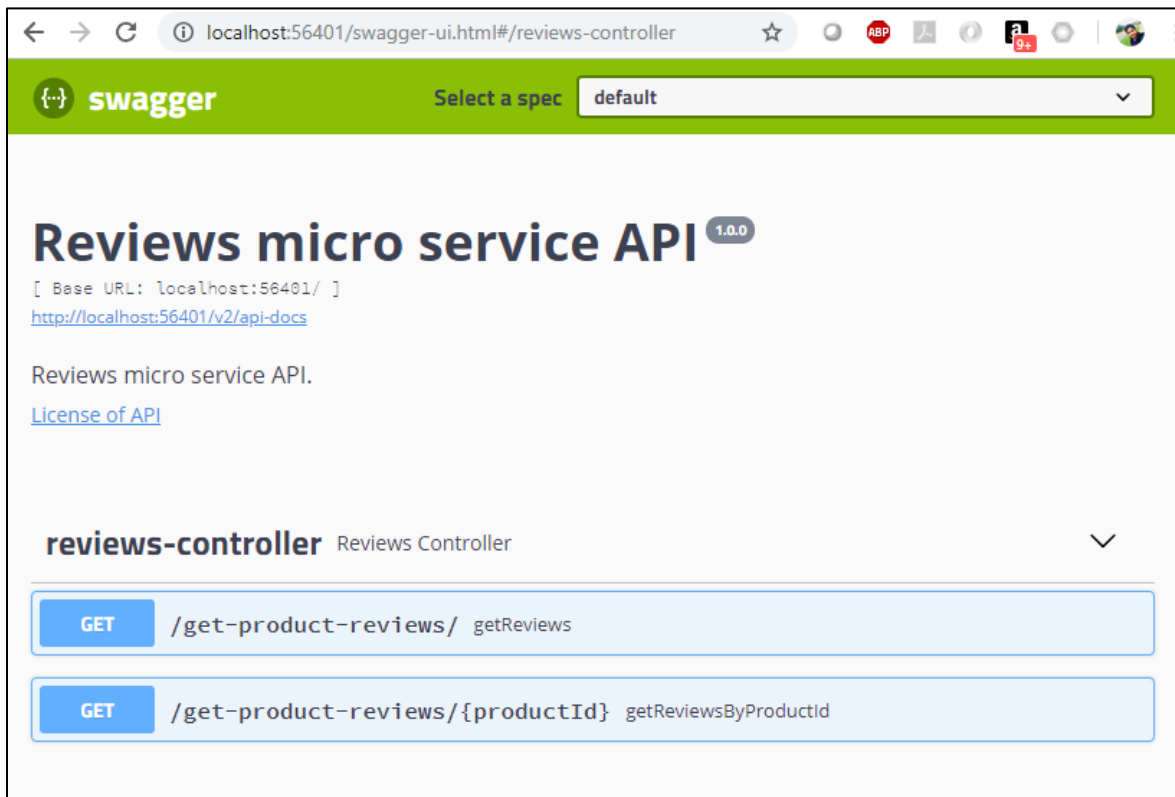




**Figure 20:** Run dashboard showing running apps with different database servers back-ends for e-cart application



**Figure 21:** Swagger UI with the all operations in e-cart



**Figure 22:** Swagger-UI for all operations in reviews app

```

/**
 * @author Venkatesh.Kandula
 * @version 1.0
 * Create Date: Aug 20,2018
 */

@Service
public class ProductsService {

    @Autowired
    private ProductsRespository productsRespository;

    @Autowired
    private MsReviewsClient client;

    public GCPProductResponseAPI getProductByName(String productName) {

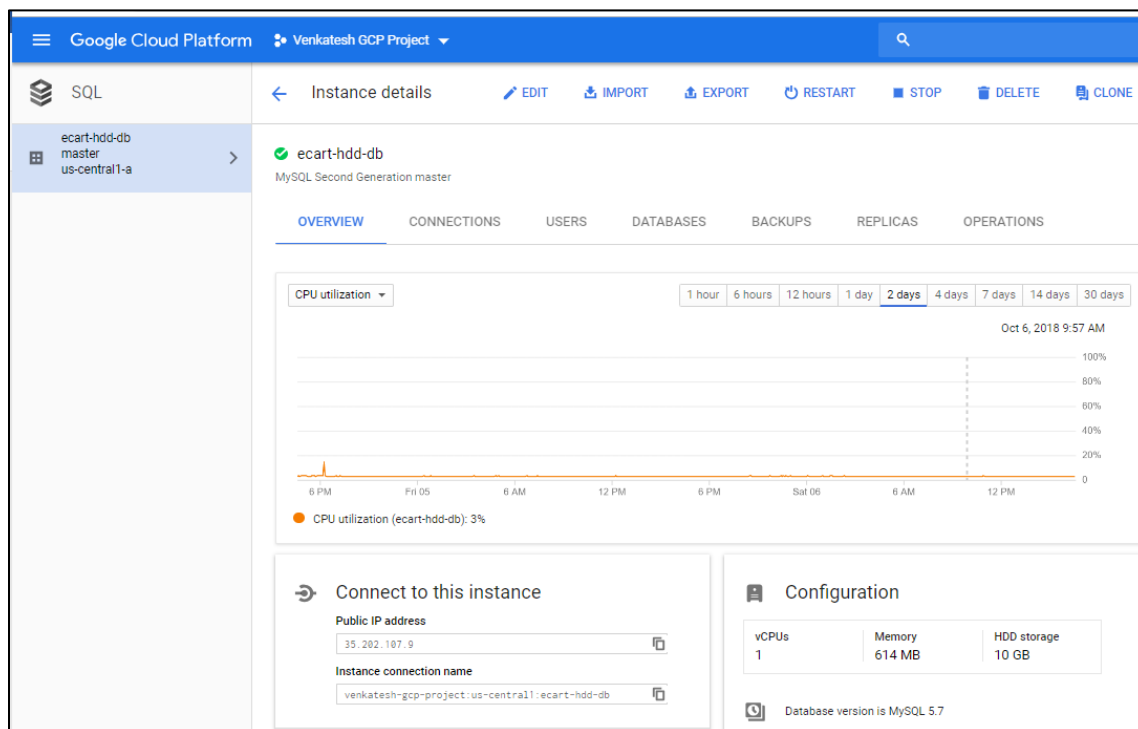
        GCPProductResponseAPI gcpProductResponseAPI = new GCPProductResponseAPI();
        ProductsResponse productsResponse = productsRespository.findByProductName(productName);

        //calling ms-reviews API using REST Client
        ReviewResponseAPI reviewResponseAPI = client.getReviews();

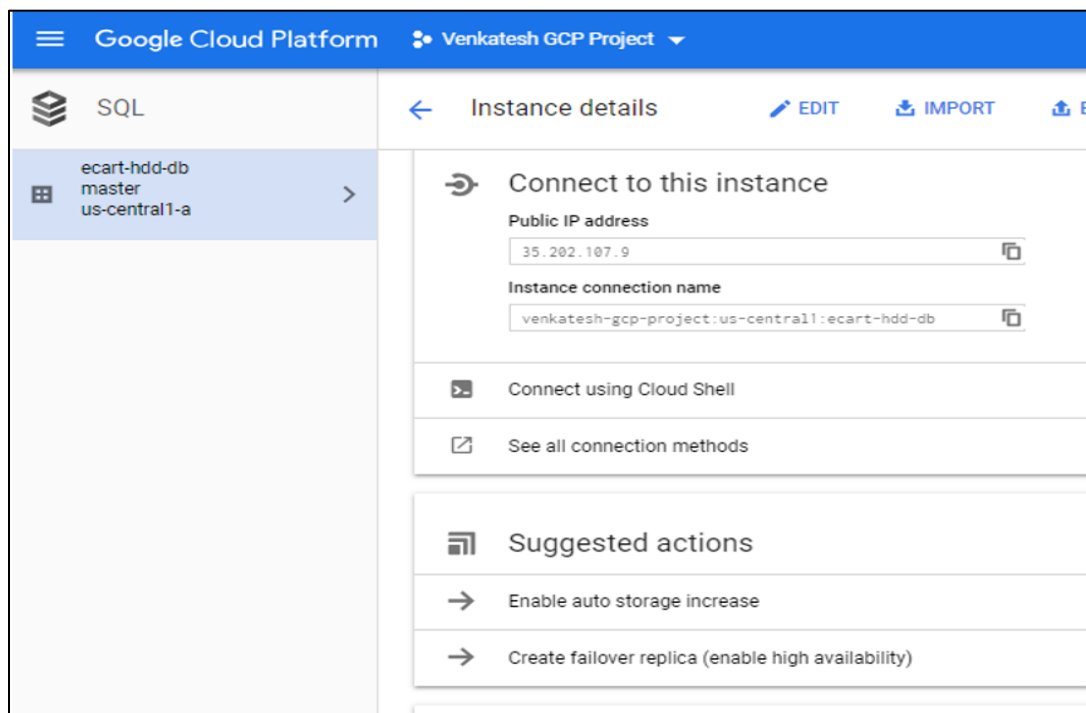
        gcpProductResponseAPI.setProductsResponse(productsResponse);
        gcpProductResponseAPI.setReviewResponseList(reviewResponseAPI.getReviewResponseList());
        return gcpProductResponseAPI;
    }
}

```

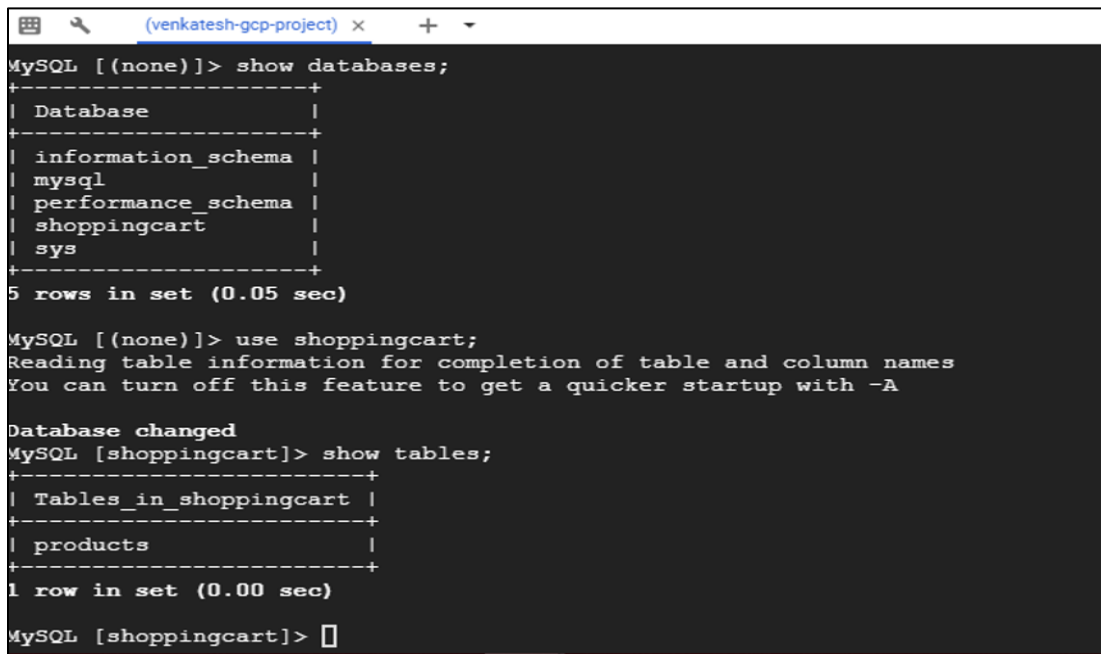
**Figure 23:** Calling reviews API from e-cart API using REST template for integration



**Figure 24:** Details of each of the database in the cloud platform



**Figure 25:** Configurations needed to connect with database in cloud platform



```

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| shoppingcart |
| sys |
+-----+
5 rows in set (0.05 sec)

MySQL [(none)]> use shoppingcart;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

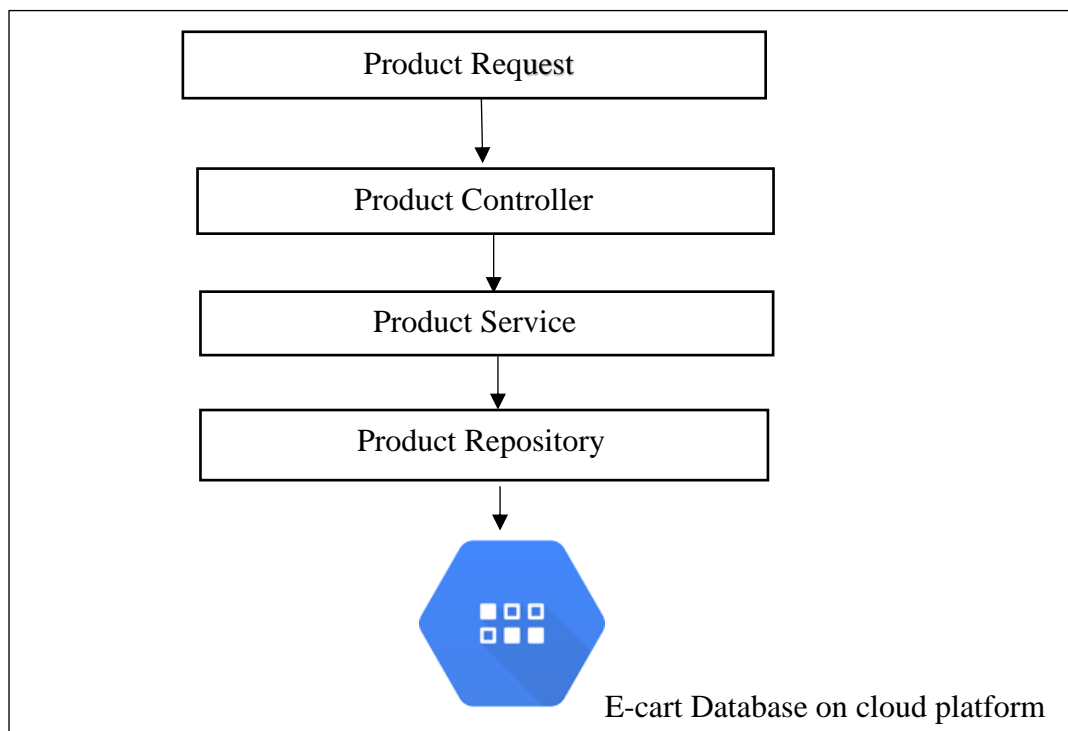
Database changed
MySQL [shoppingcart]> show tables;
+-----+
| Tables_in_shoppingcart |
+-----+
| products |
+-----+
1 row in set (0.00 sec)

MySQL [shoppingcart]> 

```

**Figure 26:** Cloud console to check the table details in cloud platform

**E-cart application code flow: (Micro-service 1).**

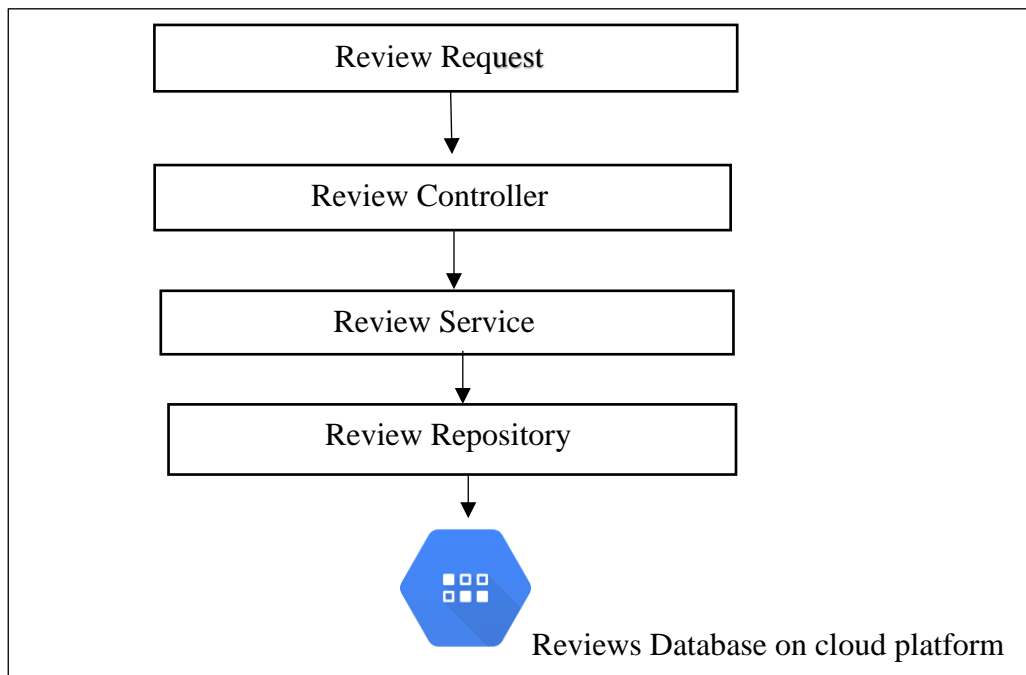


**Figure 27:** Design flow of e-cart application

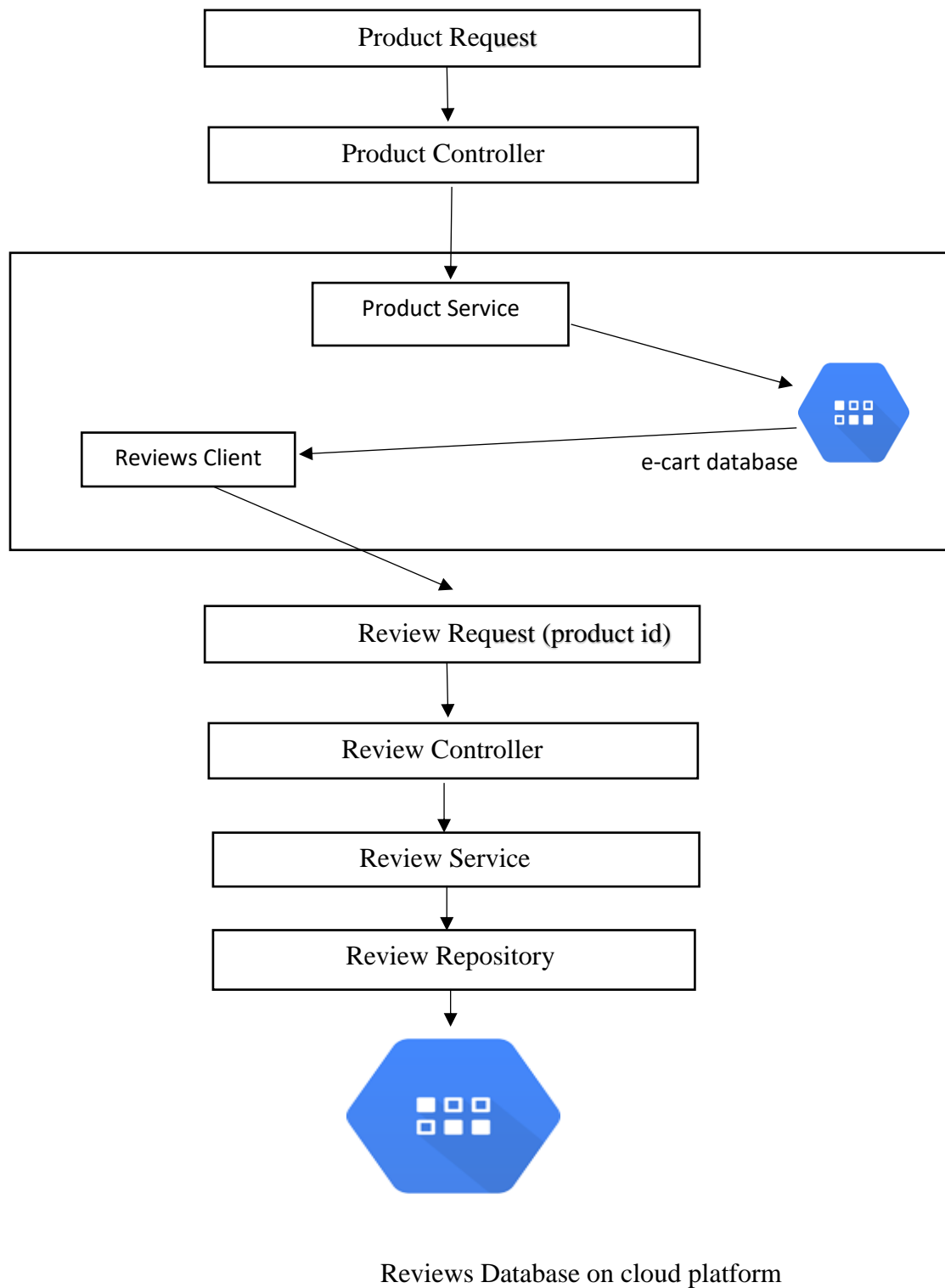
Above figure illustrates the flow or the design of e cart application, it will take a request from user and pass it to the controller, controller will make needed validations and pass it to the service, orchestration takes place here. Service will call the e cart database to get the products once it gets the details it set all the product details from database to the response object.

After that it will make a call to the review application or reviews micro service with the product details from the e cart application. It will take product id from the product details and make a call to the review application using rest template call. It will collect all the reviews information from both databases and aggregates together as a single response and gives to the user.

**Review application code flow: (Micro-service 2).**



**Figure 28:** Design flow of reviews application

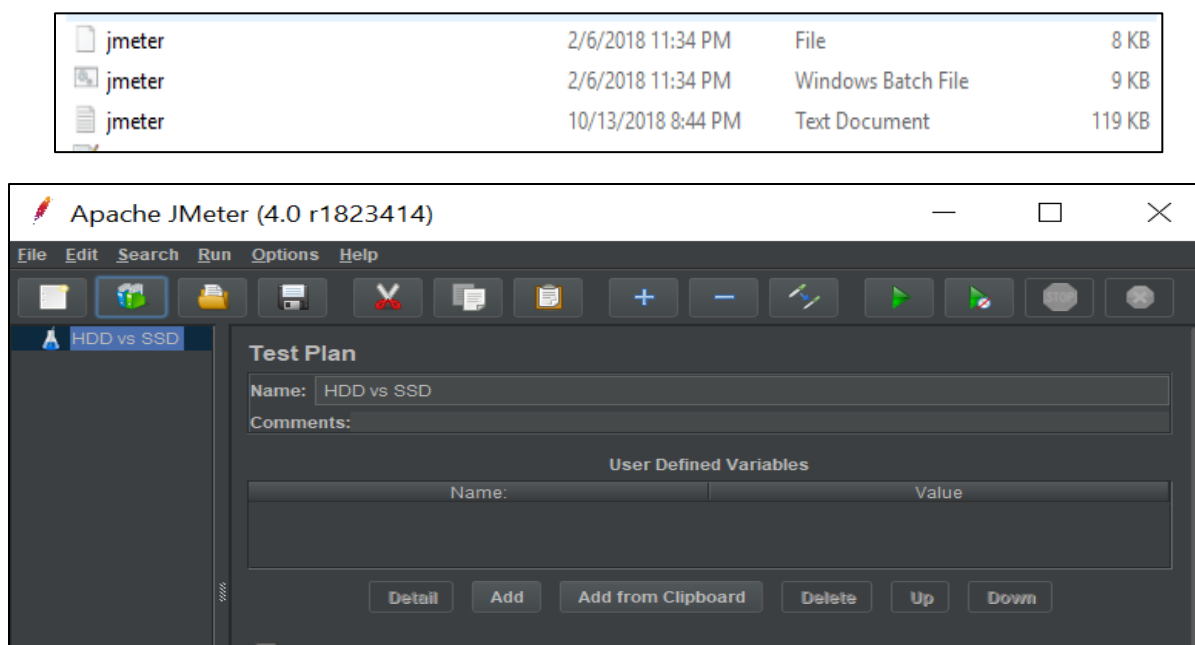


**Figure 29:** Integration of ms-ecart and ms-reviews applications

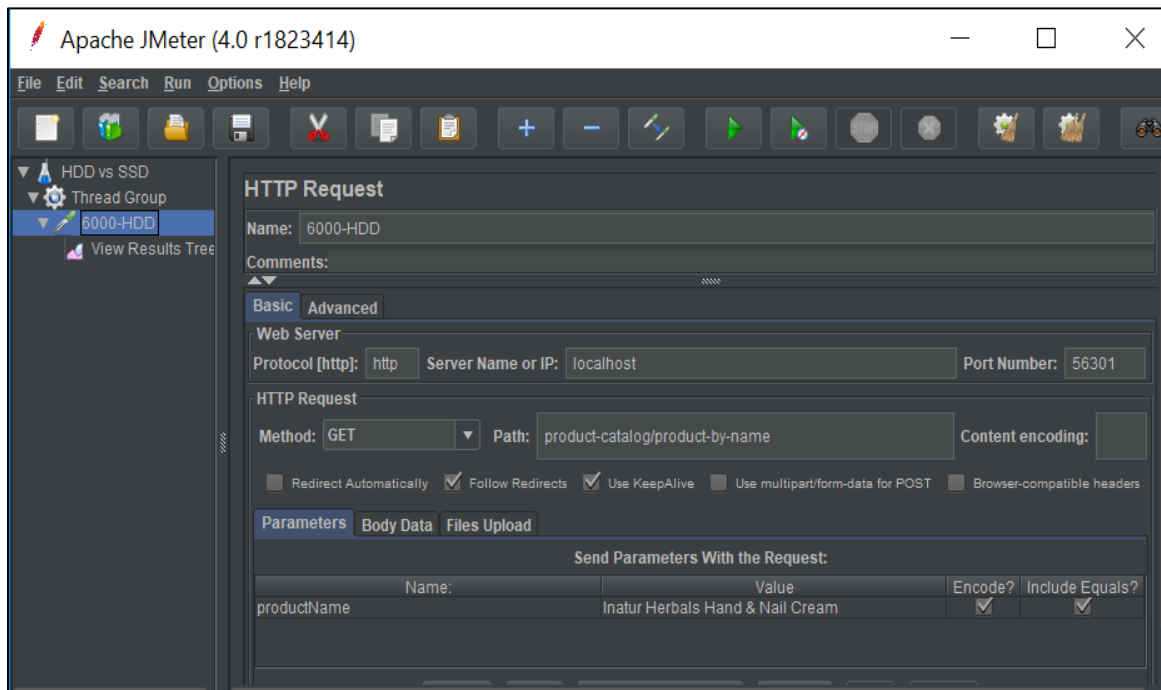
## Data Analysis

After setting up the environment, running apps and connecting them with databases in cloud environment, we need to do test runs and analyze the results. The tool we use for running tests is Jmeter.

**Testing the applications using Jmeter:** Jmeter is used to check load and performance capabilities of web applications. It is an open-source tool from apache. We need to have Java in the system for Jmeter to work. For running this tool, download the application from apache.org website, after downloading successfully navigate to bin folder of the downloaded folder and run the Jmeter.bat file. After running the bat file Apache Jmeter will be opened and give test plan name as HDD vs SSD.



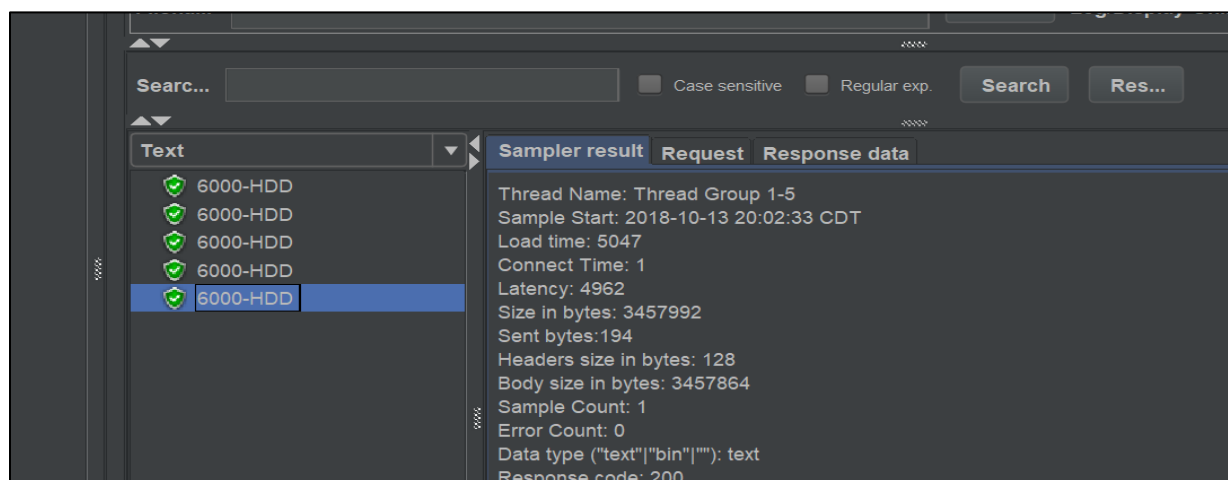
**Figure 30:** Dashboard of Jmeter with HDD vs SSD test plan



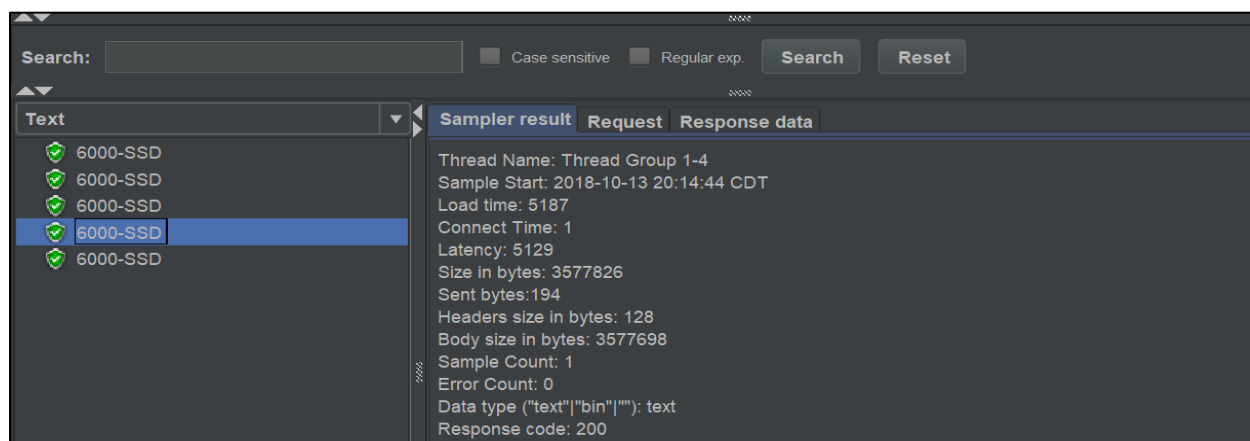
**Figure 31:** Creating a HTTP request to hit a web application

After creating test plan create a HTTP request in test plan to hit e-cart application which was connected to one the databases hosted in cloud environment. Request can be created by giving host name, protocol type, and port in which application was hosted and other request parameters in the request. Hit the run button on the menu bar at the top to run the tests and the results will get captured in the result tree (REST API testing, 2017). Same configuration needs to be prepared for all the different scenarios for different applications hosted. Below are the results after running it with different Configurations.

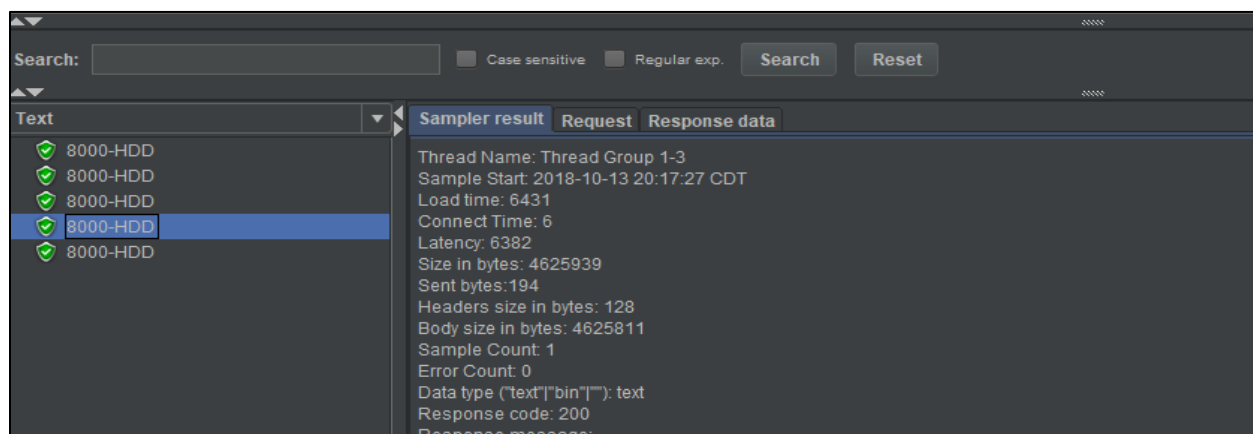




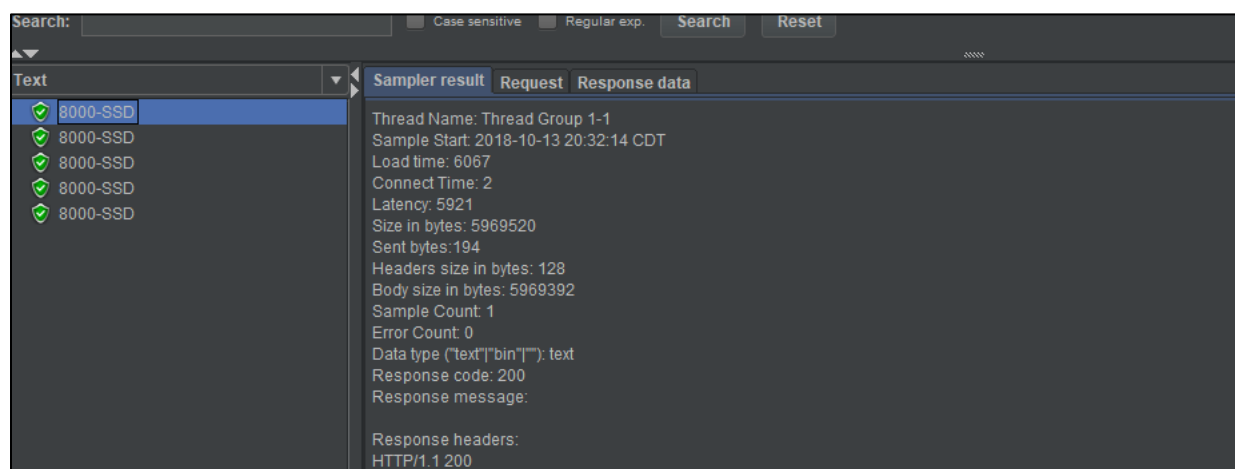
**Figure 32:** Results obtained by sending request to application with 6000 records (HDD)



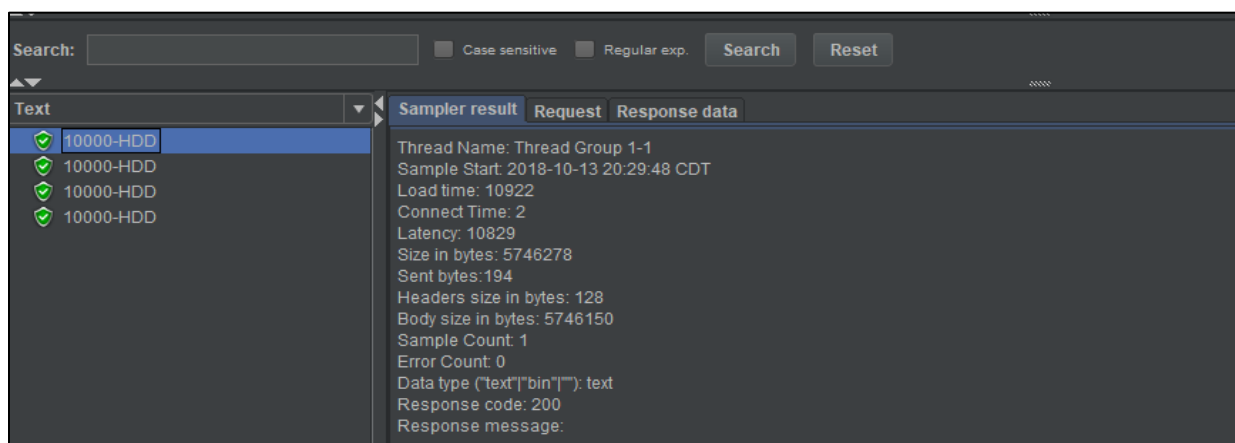
**Figure 33:** Results obtained by sending request to application with 6000 records (SSD)



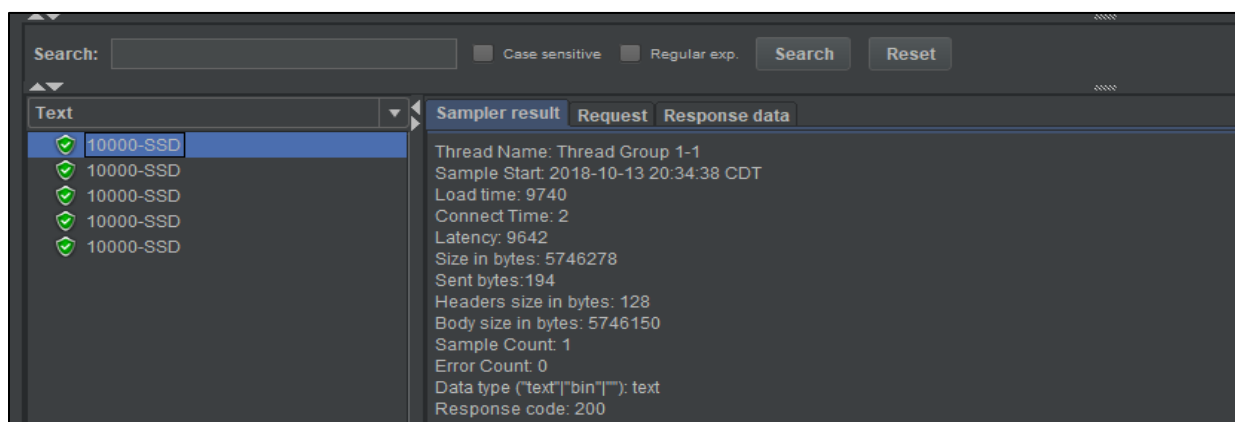
**Figure 34:** Results obtained by sending request to application with 8000 records (HDD)



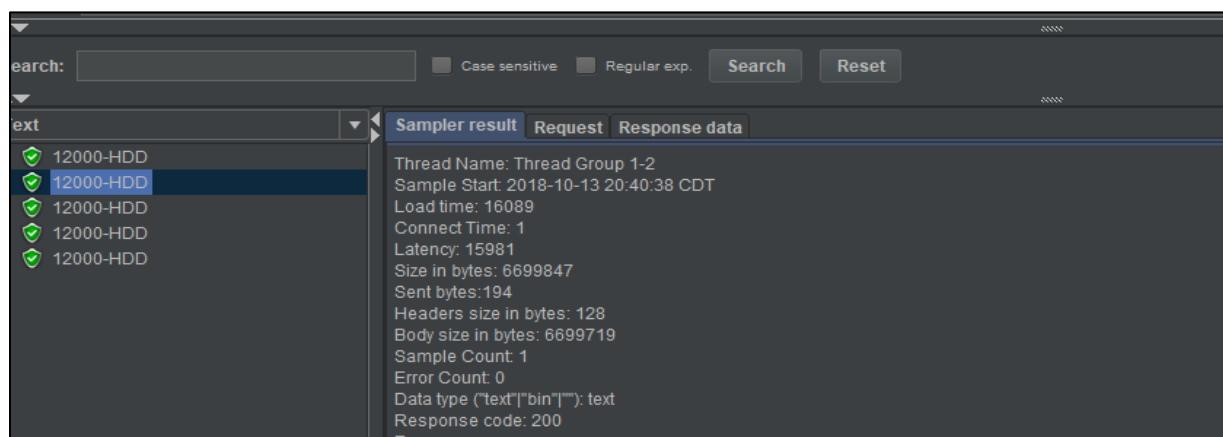
**Figure 35:** Results obtained by sending request to application with 8000 records (SSD)



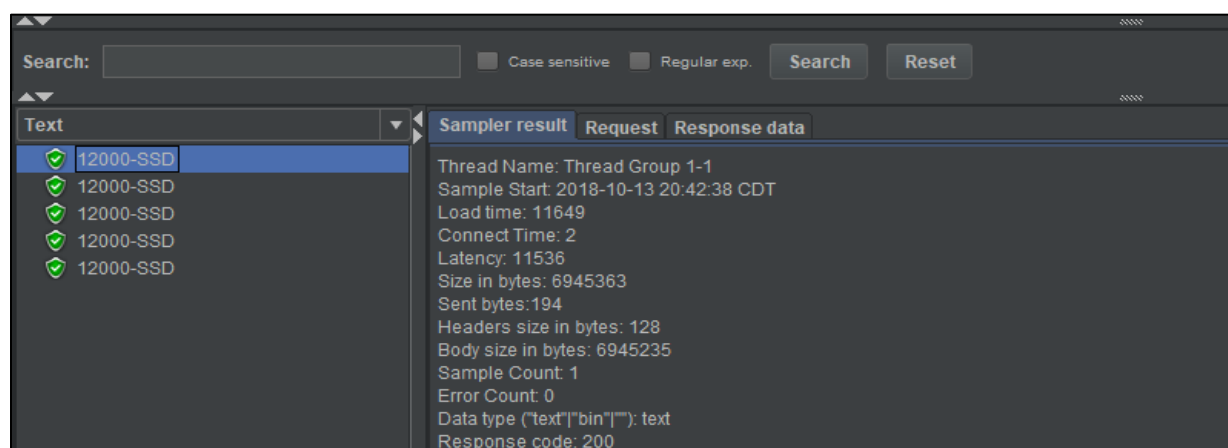
**Figure 36:** Results obtained by sending request to application with 10000 records (HDD)



**Figure 37:** Results obtained by sending request to application with 10000 records (SSD)



**Figure 38:** Results obtained by sending request to application with 12000 records (HDD)



**Figure 39:** Results obtained by sending request to application with 12000 records (SSD)

## Summary

In this chapter, we have created all the setup needed for our analysis, which includes creating web applications, setting up MySQL databases, adding needed configurations to connect applications with databases. After set-up is done, we have run the Load tests to find the load time for each of the application. Load tests were run on each of the application with the same request and captured the load time using Jmeter tool.

## Chapter V: Introduction, Results, and Conclusion

### Introduction

In this chapter, we will analyze the result obtained from Jmeter. After running the load tests the results are analyzed and generated graphs using Tableau tool. All the tests were ran using the same request only change will be the amount of data processing to generate the results. When the results of test runs are is uploaded to Tableau it will give interactive visual dynamics comparing hard disk drives (HDD's) and solid-state drives (SSD's).

### Discussion and Results

**Table 4:** Results Obtained by Running Tests on HDD's

Number of records	Load time in HDD's (in ms)
6000	5047
8000	6431
10000	10922
12000	16089

**Table 5:** Results Obtained by Running Tests on SSD's

Number of records	Load time in SSD's (in ms)
6000	5187
8000	6067
10000	9740
12000	11649

We can notice that with less number of records the amount of load time does not making much differences. But with increase in the size of database it is there was a spike differences of load times in the HDD's and SSD's. These days, web applications are designed with idea of loose coupling, independent functionality and ease of integration. Each of the application is setup and backed by its own database instance working solely for its one purpose following the principle of single responsibility. If we take systems like banking platform it can be divided

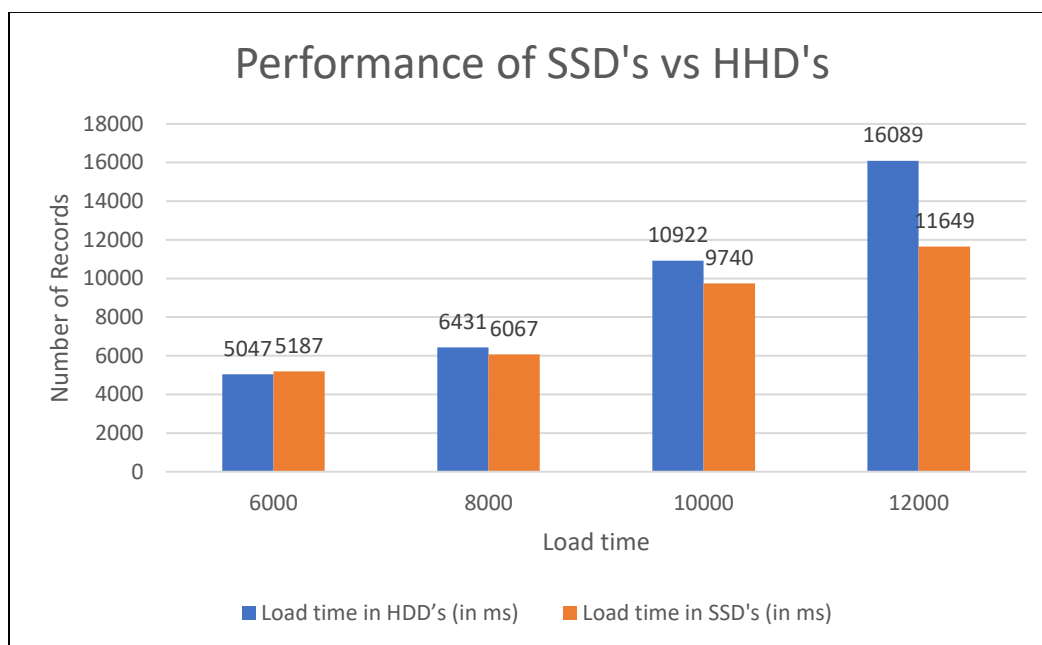
small number of applications for credit card operations, debit card operations, personal loans, mortgage loans, payment operations, promotional offers, teller operations, third party integrations like credit bureau operations and instant operations like instant credit cards and instant loans etc., for both prospective customers and current customers as well. This will end up creating lot of micro services backed by its own database all the services need to be worked with less processing time to complete the request and generate the required response.

If we take a scenario of a prospective customer applying for credit card loan it may include multiple back transactions which includes capturing all the details of a personal applying credit card, job details and calling credit bureau check from third party site, calling promo codes service to find the all the possible offers for the person and pull them from database, Processing requests from archives if the applicant applied it very recently. If everything is fine aggregate all the details and call the APR engine to get the cash APR and purchase APR details. All these business operations need to be done in fraction of minutes and instant decision need to be provided.

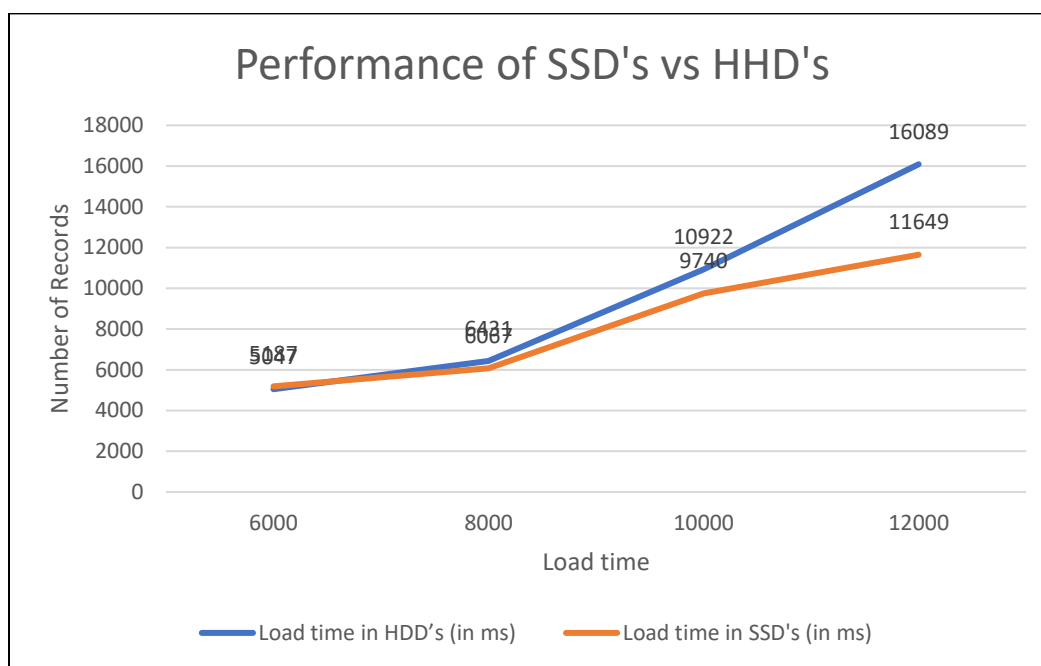
There are many things which need to concentrate to make these kinds of requests with less response time. Apart from code quality we need look into infrastructure and storage media as well, which will help in contributing to faster processing of requests and generating responses quickly. Building small applications with its own independent databases makes the design and architecture simple.

It also helps a lot in maintenance as well. If all the functionalities are built into a single application it is very tough to make even a small change and scope of testing and quality assurance will increases.

**Graphical representation of HDD's vs SSD's:** Bar graph representing the comparison of results obtained through Jmeter. We can notice from the bar graph the performance of can be identified well with growth of amount of data under process. If the data is very less and if the processing data is archived very frequently, we can continue using HDD's. But if the frequency of transactions more it will result in increase of input and out operations (IOPS) making the application to work slow and eventually it make ramp down the system completely.



**Figure 40:** Bar graph displaying performance differences of HDD's and SSD's



**Figure 41:** Line chart representing trends in changing of load time of HDD's and SSD's with change in the number of records

## Conclusion

After setting up environments with SSD's and HDD's, both of them were tested by hitting with same HTTP requests using Jmeter. The response remains same irrespective of environment and size of database, only difference is number of records need to be processed for obtaining results in the database. More ever we can clearly identify the differences in performance in terms of load time has been increasing exponentially with increase in the number of records need to be processed.

We have discussed about the key features of solid-state drives and its advantages over hard disk drives. We have also discussed about role of solid-state drives in the database servers which will be used as back end for web applications to store data. This paper also discussed about integrating independent micro services which are backed with database servers with

storage media as solid-state drives and finally deduced a graph representing the trends in performance in terms of load time.

### **Future Work**

With the help of this experiment we get to know the variation in the load time of responses with medium size datasets. This research can be extended by how by increasing the parallel requests at the same time and running the tests and analyze the results. At the same time this type of research can be done by setting up huge databases with series of continuous read and write operations where we can analyze the performance of caches as well.



## References

- Cloud SQL / Cloud SQL / Google Cloud.* (2017). Retrieved from google: <https://cloud.google.com/sql/docs/>.
- Data caching in JPA.* (n.d.). Retrieved from <http://www.thejavageek.com>: <http://www.thejavageek.com/wp-content/uploads/2014/05/jpa-caching.png>.
- Google compute engine documentation / compute engine / Google Cloud.* (2017). Retrieved from google: <https://cloud.google.com/compute/docs/>.
- Growth in web api's since 2005.* (n.d.). Retrieved from <https://www.programmableweb.com/>: [https://www.programmableweb.com/sites/default/files/growth-in-web-apis-since-2005\\_0.png](https://www.programmableweb.com/sites/default/files/growth-in-web-apis-since-2005_0.png).
- Hard disk drive.* (n.d.). retrieved from wikipedia: [https://en.wikipedia.org/wiki/hard\\_disk\\_drive](https://en.wikipedia.org/wiki/hard_disk_drive).
- How does a hard drive work?* (2018). Retrieved from explain that stuff: <http://www.explainthatstuff.com/harddrive.html>.
- How to set up MySQL on google compute engine / solutions / google cloud.* (2017). Retrieved from google: <https://cloud.google.com/solutions/setup-mysql>.
- How to use spring resttemplate client for consuming restful webservice.* (2018, September). Retrieved from <https://grokonez.com>: [c/java-integration/use-spring-resttemplate-client-consuming-restful-webservice](https://grokonez.com/c/java-integration/use-spring-resttemplate-client-consuming-restful-webservice).
- Hu, J. (2012). *Improving performance of solid state drives in enterprise*. Lincoln.
- Internal structure of hard disk drive.* (2017, January). Retrieved from <https://thetechhacker.com>: <https://i0.wp.com/thetechhacker.com/wp-content/uploads/2017/01/hdd-components.jpg>.
- Khan, K., & Jan, A. (2011). *Evaluating google app engine for enterprise application development*. Karlskrona.

*Master microservices with spring boot and spring cloud.* (2017, 12). Retrieved from

[https://www.codejava.net: https://www.codejava.net/frameworks/hibernate/java-  
hibernate-jpa-annotations-tutorial-for-beginners.](https://www.codejava.net/frameworks/hibernate/java-hibernate-jpa-annotations-tutorial-for-beginners)

*MySQL client/server model.* (n.d.). Retrieved from <http://softwsp.com/>: [ttp://download.softwsp.  
com/sites/13/2015/10/mysql-database-server-win-005.png](http://download.softwsp.com/sites/13/2015/10/mysql-database-server-win-005.png).

*Rest api testing.* (2017, 05). Retrieved from <https://www.blazemeter.com>: [https://www.](https://www.blazemeter.com/blog/rest-api-testing-how-to-do-it-right)

[blazemeter.com/blog/rest-api-testing-how-to-do-it-right.](https://www.blazemeter.com/blog/rest-api-testing-how-to-do-it-right)

*Rest presentation documents.* (n.d.). Retrieved from <https://image.slidesharecdn.com/>:

[https://image.slidesharecdn.com/rest-presentation-121213012311-phpapp02/95/rest-  
presentation-15-638.jpg?cb=1355361849.](https://image.slidesharecdn.com/rest-presentation-121213012311-phpapp02/95/rest-presentation-15-638.jpg?cb=1355361849)

*Searchnetworking.* (2018). Retrieved from what is throughput: <http://searchnetworking>

[.techtarget.com/definition/throughput.](http://searchnetworking.techtarget.com/definition/throughput)

*Server-sandbox.* (n.d.). Retrieved from <http://kikobeats.github.io>: [http://kikobeats.github.io/](http://kikobeats.github.io/server-sandbox/03.%20services/http/01.%20introduction.html)

[server-sandbox/03.%20services/http/01.%20introduction.html.](http://kikobeats.github.io/server-sandbox/03.%20services/http/01.%20introduction.html)

*Simple flash cell design.* (n.d.). Retrieved from <https://eeherald.s3.amazonaws.com>:

[https://eeherald.s3.amazonaws.com/uploads/ckeditor/pictures/oldarticleimages/flash\\_cell  
\\_structure.jpg.](https://eeherald.s3.amazonaws.com/uploads/ckeditor/pictures/oldarticleimages/flash_cell_structure.jpg)

*Software-architecture.* (n.d.). Retrieved from <http://tutorials.jenkov.com>: [http://tutorials.](http://tutorials.jenkov.com/images/software-architecture/client-server-2.png)

[jenkov.com/images/software-architecture/client-server-2.png.](http://tutorials.jenkov.com/images/software-architecture/client-server-2.png)

*Spring boot tutorial.* (2018, June). Retrieved from <https://www.baeldung.com>: [https://www.](https://www.baeldung.com/spring-boot-start)

[baeldung.com/spring-boot-start.](https://www.baeldung.com/spring-boot-start)

*SSD advantage.* (n.d.). Retrieved from insight: [https://www.insight.com/content/dam/insight-web/en\\_us/article-images/whitepapers/partner-whitepapers/the%20ssd%20advantage.pdf](https://www.insight.com/content/dam/insight-web/en_us/article-images/whitepapers/partner-whitepapers/the%20ssd%20advantage.pdf).

*Storage options.* (n.d.). Retrieved from <https://cloud.google.com/compute/docs/disks/>.

*Structure of SSD block .* (2015, 07). Retrieved from <https://www.extremetech.com/https://www.extremetech.com/wp-content/uploads/2015/07/diagram-1.png>.

*Trends showing adoption of Spring boot.* (n.d.). Retrieved from <https://pbs.twimg.com/media/dbxadjyv0aaqb2h.jpg:large>.

Vasyl, R. (2016, August). *A beginner's tutorial for understanding restful api.* Retrieved from <https://mlsdev.com/https://mlsdev.com/blog/81-a-beginner-s-tutorial-for-understanding-restful-api>.

*What is a database server?* (2018). Retrieved from techopedia: <https://www.techopedia.com/definition/441/database-server>.

*What is RAID (redundant array of independent disks)?* (2018). retrieved from searchstorage: <http://searchstorage.techtarget.com/definition/raid>.

*What is SSD (solid-state drive)? - definition from whatis.com.* (2018). Retrieved from searchstorage: <http://searchstorage.techtarget.com/definition/ssd-solid-state-drive>.

Wiseman, T. (n.d.). *The effects of an ssd on sql server performance.* Retrieved from timothyawiseman: <https://timothyawiseman.wordpress.com/2012/07/29/the-effects-of-an-ssd-on-sql-server-performance/>.

*Working with HTTP status codes*. (2017, 06). Retrieved from <https://codetteddy.com>:

<https://codetteddy.com/2017/06/06/create-api-with-asp-net-core-day-3-working-with-http-status-codes-in-asp-net-core-api/>.

Zambelli, C., Micheloni, R., & Luca, C. (2018). Impact of the nand flash power supply on solid state drives reliability and performance. *IEEE*.

## Appendix

Sample code for used for developing the web application for connecting it to the database deployed in cloud level is deployed here.

ProductController.java

```
@RestController
@Api()
@RequestMapping("/product-catalog")
public class GCPPProductsController {

    @Autowired
    private ProductService productService;

    @GetMapping("/list-of-products")
    @ResponseBody
    public List<ProductsResponse> getProductList() {

        return productService.getProductCatalog();
    }

    @GetMapping("/product-by-name")
    @ResponseBody
    public GCPPProductResponseAPI getProductByName(@RequestParam(name =
"productName") String productName){

        return productService.getProductByName(productName);
    }

    @GetMapping("/{product-id}")
    @ResponseBody
    public GCPPProductResponseAPI getProductById(@PathVariable(name = "product-
id") String productId) {

        return productService.getProductCatalogById(productId);
    }
}
```

ProductService.java

```
@Service
public class ProductService {
```

```

    @Autowired
    private ProductsRepository productsRepository;

    @Autowired
    private MsReviewsClient client;

    public List<ProductsResponse> getProductCatalog(){

        return productsRepository.findAll();
    }

    public GCPPProductResponseAPI getProductCatalogById(String productId) {

        GCPPProductResponseAPI gcpProductResponseAPI = new
        GCPPProductResponseAPI();

        ProductsResponse productsResponse =
        productsRepository.findById(productId);

        ReviewResponseAPI reviewResponseAPI = client.getReviews();

        gcpProductResponseAPI.setProductsResponse(productsResponse);

        gcpProductResponseAPI.setReviewResponseList(reviewResponseAPI.getReviewRespon
        seList());
        return gcpProductResponseAPI;

    }

    public GCPPProductResponseAPI getProductByName(String productName) {

        GCPPProductResponseAPI gcpProductResponseAPI = new
        GCPPProductResponseAPI();
        ProductsResponse productsResponse =
        productsRepository.findByName(productName);
        ReviewResponseAPI reviewResponseAPI = client.getReviews();
        gcpProductResponseAPI.setProductsResponse(productsResponse);

        gcpProductResponseAPI.setReviewResponseList(reviewResponseAPI.getReviewRespon
        seList());
        return gcpProductResponseAPI;
    }
}

public class GCPPProductResponseAPI {

```

```

@ApiModelProperty(name = "Product details")
private ProductsResponse productsResponse;

@ApiModelProperty(name = "List of reviews")
private List<ReviewResponse> reviewResponseList;

public ProductsResponse getProductsResponse() {
    return productsResponse;
}

public void setProductsResponse(ProductsResponse productsResponse) {
    this.productsResponse = productsResponse;
}

public List<ReviewResponse> getReviewResponseList() {
    return reviewResponseList;
}

public void setReviewResponseList(List<ReviewResponse> reviewResponseList) {
    this.reviewResponseList = reviewResponseList;
}
}

```

ReviewResponse.java

```

public class ReviewResponse {

    private String reviewerId;
    private String productId;
    private String reviewerName;
    private String reviewText;
    private String overall;
    private String summary;

    public String getReviewerId() {
        return reviewerId;
    }

    public void setReviewerId(String reviewerId) {
        this.reviewerId = reviewerId;
    }

    public String getProductId() {
        return productId;
    }
}

```

```

    }

    public void setProductid(String productid) {
        this.productid = productid;
    }

    public String getReviewername() {
        return reviewername;
    }

    public void setReviewername(String reviewername) {
        this.reviewername = reviewername;
    }

    public String getReviewtext() {
        return reviewtext;
    }

    public void setReviewtext(String reviewtext) {
        this.reviewtext = reviewtext;
    }

    public String getOverall() {
        return overall;
    }

    public void setOverall(String overall) {
        this.overall = overall;
    }

    public String getSummary() {
        return summary;
    }

    public void setSummary(String summary) {
        this.summary = summary;
    }
}

```

ProductsRespository.java

```

@Repository
public interface ProductsRespository extends JpaRepository<ProductsResponse, Integer>
{

```



```

    ProductsResponse findByUniqId(String id);

    ProductsResponse findByProductName(String productName);
}

ReviewController.java

@RestController
@Api()
@RequestMapping("/get-product-reviews")
public class ReviewsController {

    @Autowired
    private ReviewsService reviewsService;

    @GetMapping("/")
    @ResponseBody
    public ReviewResponseAPI getReviews() {

        List<ReviewResponse> reviews = reviewsService.getReviews();
        ReviewResponseAPI reviewResponseAPI = new ReviewResponseAPI();
        reviewResponseAPI.setReviewResponseList(reviews);
        return reviewResponseAPI;
    }

    @GetMapping("/{productId}")
    @ResponseBody
    public List<ReviewResponse> getReviewsByProductId(@PathVariable(name = "product
id") String id) {

        return reviewsService.getReviewsByProductId(id);
    }
}

```

ReviewsService.java

```

@Service
public class ReviewsService {

    @Autowired
    private ReviewRepository reviewRepository;

    public List<ReviewResponse> getReviews() {

        return reviewRepository.findAll();
    }
}

```

```

    }

    public List<ReviewResponse> getReviewsByProductId(String id) {

        return reviewRepository.findByProductid(id);
    }
}

```

ReviewRepository.java

```

@Repository
public interface ReviewRepository extends JpaRepository<ReviewResponse,Integer> {

    List<ReviewResponse> findByProductid(String productId);
}

```

SwaggerConfiguration.java

```

@Configuration
@EnableSwagger2
public class SwaggerConfiguration {

    @Bean

    public Docket api() {

        return new Docket(DocumentationType.SWAGGER_2)

            .select()

            .apis(RequestHandlerSelectors.any())

            .paths(Predicates.not(PathSelectors.regex("/error")))

            .build()

            .apiInfo(apiInfo());

    }

    private ApiInfo apiInfo() {

        return new ApiInfo(

            "Reviews micro service API",

```

```
"Reviews micro service API.",  
"1.0.0"  
new Contact("Venkatesh Kandula", "", ""),  
"License of API", "API license URL", Collections.emptyList());  
}  
}
```