Analysis of Data Oriented Web Application Systems

Anubhav Dinkar[1], Prakash Biswagar[2]

[1]*Student, Dept of Electronics and Communication, R.V. College of Engineering, Bangalore, INDIA -560102*

[2]*Professor, Dept of Electronics and Communication, R.V. College of Engineering, Bangalore, INDIA -560102*

[1]*anubhavdinkar05@gmail.com*, [2]*prakashbiswagar@rvce.edu.in*

D.O.I - 10.51201/JUSST/21/05-199

http://doi.org/10.51201/JUSST/21/05199

Abstract: *The purpose of this paper is to study and analyse the various tools that are used in modern day web application systems, which include but are not limited to Flask, Django, PostgreSQL, MongoDB, Docker containers, virtual machines, and so on. The main aim is to allow users of these technologies to be able to choose the right technology based on their needs and the scale of their applications. This is done with the help of sysbench and Docker and Linux based containers, along with basic Flask and Django web applications. Flask could be preferred for simpler web applications over Django. Docker and LXD do perform similarly for the most part, but due to its low storage footprint (only essential libraries are installed in the container, not an entire OS), and its ease of configurability in almost all operating systems, Docker is generally preferred over the others. PostgreSQL seemed to perform 2 times better than MongoDB in terms of the number of queries it handled*

Keywords: **Containers, Docker, Python, Flask, Django, Database, MongoDB, PostgreSQL, Virtualization, Linux, LXC, LXD**

1. INTRODUCTION

Web application systems are those sets of services that communicate with each other in order to be able to give functionality to the application it is being used to develop. Servers and clients serve as the backbone of application development over the internet. Websites have a framework (usually for both frontend and backend) on a remote server, and a user's browser (either on a computer or a mobile device), acts as a client to that server.

Broadly, the architecture can be divided into two parts - frontend and backend. This setup is usually present on the server, which is accessed with the help of APIs on the client side.

There is an increased shift from static websites, which make use of HTML, CSS and JS to dynamic websites, which make use of web services like REST APIs, dynamic JS libraries like jQuery, and so on, which have lower loading times due to the API based systems that are used. Servers need only send the required information in XML or JSON format which can then be consumed by the client to generate the website. [6]
Further, containerized development environments are being preferred over a simple development environment as containers are proven to be more flexible and portable. The code will run the same way irrespective of the platform it runs on - a physical workstation, or a cloud based server. [3]

Web architectures are constantly replacing the older method of static website development, and most companies are migrating to these methods. Most companies that build their applications for the internet make use of these technologies. YouTube, Spotify and Pinterest are some examples of websites that make use of Django. On the other hand, Netflix and Reddit are built with the help of Flask [14]. Companies like JP Morgan Chase and ThoughtWorks make use of Docker in their applications while KPMG, Toyota, et cetera use the non-relational database MongoDB. PostgreSQL is the second most used database technology in the world right now, and is used by companies like BMW in their applications. [22] However, it is important to understand the exact use cases and scenarios of the application so that the correct technology is chosen for the requirements, so as to

achieve the highest efficiency. Incorrect tool usage can cause a significant computation cost as well as an increase in latency, which is not ideal.

Several such tools which are used in modern day web systems are discussed and analyzed here, which will provide the users of such services with quantifiable data to fall back on while choosing the services they need for their applications.

2. LITERATURE REVIEW

The purpose of the study done in [1] was to study the several features of the two popular web development frameworks in Python - Django and Flask. It presents the practical comparison of the two frameworks, with the help of a basic social networking application built on both Flask and Django for the frontend, while using Python, SQLite, HTML, CSS, JS and Ajax as well. It showed that the main points of advantage of Flask over Django was that it was simple, flexible and was very much under the developer's control. On the other hand, the extensive features and scalability of Django makes it also a popular choice. It found that the best fit for large scale applications was Django, while for smaller, less complex applications, Django can be very cumbersome and Flask would be a better choice.
As per [1], Flask is a micro-service which is very flexible in terms of the freedom of selecting the database for the website. In fact, migration can also be achieved very easily, with the help of the *Alembic* library in Python, which can generate a migration file from one kind of database to another. Also, the security provisions of Flask are high, as the entire application is sandboxed. The database cannot be accessed outside the application instance, however other measures within the application itself need to be implemented by the developer. [13]
Flask also has a functionality to verify the login information into a website, if applicable. It makes use of JWT (JSON Web Tokens) to verify and authenticate the users who login to the website. It can also be used to protect certain resources and APIs from a certain type of user, which is done with the help of access tokens. [15]

On the other hand, Django is a full fledged web application framework. The databases that can be used here are limited "out of the box" - the developer can choose among PostgreSQL, MariaDB, MySQL and SQLite. In terms of security, Django - being open source- is highly secure directly. CSRF, XSS, SQL injection, clickjacking, host header validation and SSL (HTTPS) are all taken care of here, inherently. When encryption is used with HTTP, it is known as HTTPS. HTTPS makes use of SSL to authenticate the identity of the client making the request, and this is an essential feature of any website.

With the rise in containers in various cloud architectures, users must choose their container technology wisely keeping in mind the various overheads associated with each of them. In [2], various TCP services were run to measure the server and system performances of each container, compared to a native system. While in terms of system performance, LXD was better than LXC and Docker, when it comes to the services, the performances are marginally different. Docker, however, is preferred for its ease of use and low resource footprint, as it only installs the necessary libraries on top of a basic OS in the container, rather than installing a full fledged OS. Further, [3] deals with comparing LXC and Docker and it found that the use-case is the main point of difference between the 2 technologies, and that performance is almost identical. Thus, for a lower to medium load server, Docker would be preferable due to its low latency and speed.

It was concluded by Sampathkumar [9] that LXC is better for those infrastructures that do not require a lot of isolation, and for more dynamic setups. On the other hand, Docker is useful where developers need a set of applications and libraries rather than an entire OS, thus making it a very lightweight and inexpensive solution. Gupta et al [10] concluded that LXD performed better than VMs and Docker containers in terms of computation speed, however the difference was marginal.

In [4], the comparison is made in terms of the response time in a 5 node cluster, which showed that PostgreSQL outperformed MongoDB in all cases. The speedup of PostgreSQL over MongoDB was on average above 2.6, clearly showing the superiority of PostgreSQL when it comes to query handling. On the other hand, [20] finds that the NoSQL database "performs better by a factor of 25x which increases exponentially as the data size increases". However, as PostgreSQL also has a NoSQL implementation, it is preferred over the other kinds of databases. [20]

In this paper, all results and tests conducted were on the following device specifications:

- Operating System: Mac OS X - Big Sur
- RAM: 16GB DDR4 3733 MHz
- 2.3 GHz Quad core Intel i7 processor

3. BACKEND TOOLS

PostgreSQL and MongoDB

Databases are the means of storage used by applications, either on physical devices, or on the cloud through websites. This is one of the most crucial parts of a website, as all kinds of data - ranging from the products shown on a website like Amazon to user data on any website that uses a user login system.

Broadly, there are two kinds of databases - relational and non-relational. In a period of about 5 years, non-relational databases are on a constant increase in popularity [16]. Relational databases are those in which data can be represented in terms of a table with rows and columns, with a schema that specifies what kind of data each of those columns (and hence their tables) will store. Examples include MySQL, PostgreSQL, Oracle SQL, and so on. Non-relational databases need not have any fixed structure, and even unrelated data can be stored and managed. With an increase in complexity in web applications, non-relational databases are growing in popularity. There are different kinds of non-relational databases - search engines like Elastic Search, Graph databases like Neo4J and document stores, which store data in JSON form, like MongoDB.

This section aims to compare and contrast the real world scenarios in which the two kinds of databases can be used. Here, the comparison is made between a scalable document based NoSQL database - MongoDB and an open source object relational database system- PostgreSQL. [18][19]

In this paper, the methodology employed to test the performance of the two kinds of databases is shown in the below schematic diagram.

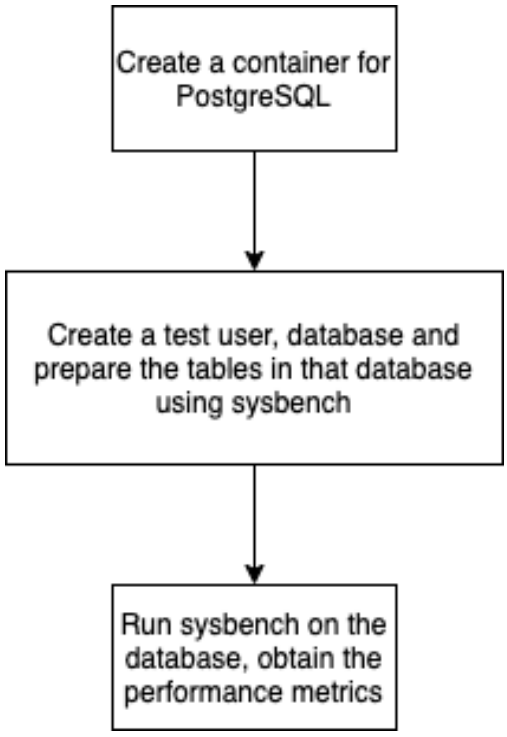

Figure 1. Flow of the Testing methodology for PostgreSQL

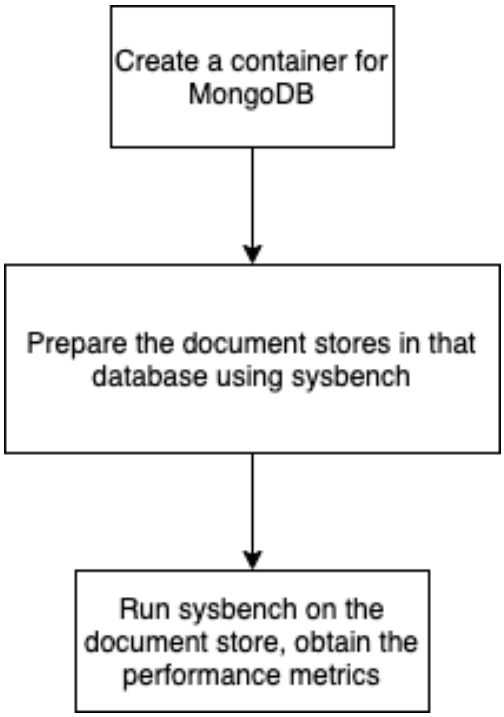

Figure 2. Flow of the Testing methodology for MongoDB

4. CONTAINERIZATION TOOLS

Virtualization is of great importance in the growth of cloud computing and is regarded as the foundation of cloud. There are widely 2 types of virtualization solutions - hypervisors and containers. Containers are newer solutions in the cloud infrastructure. This increases the capability of data centres with the help of virtualization on the server side. [8]

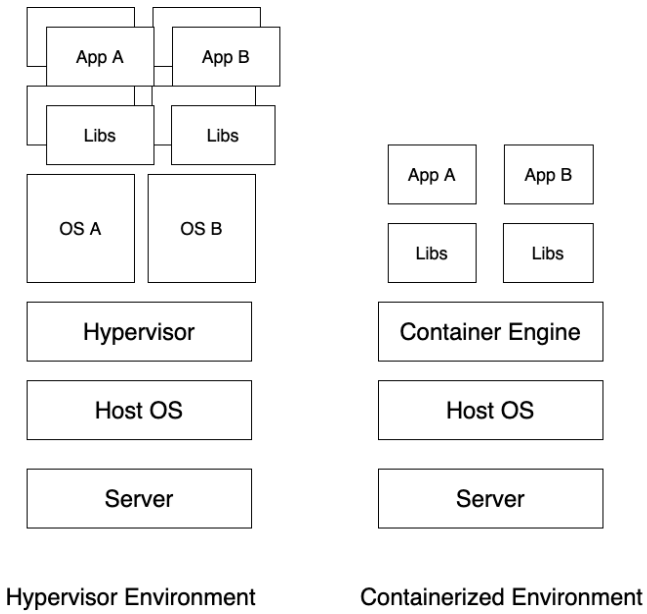

Figure 3. Containerization vs Virtualization

It has been shown that containerized environments are capable of achieving higher performance when compared to traditional hypervisor based virtualization. [7]
The more commonly used containerization engines are Docker, LXC and LXD. All of them use the same concept of running a Linux container, the performance of each of these containers differs depending on the service that is running on top of it. Currently, Docker's market share in the containerization market is constantly increasing, and is only behind LXC, and is on a constant growth since 2013. [21]

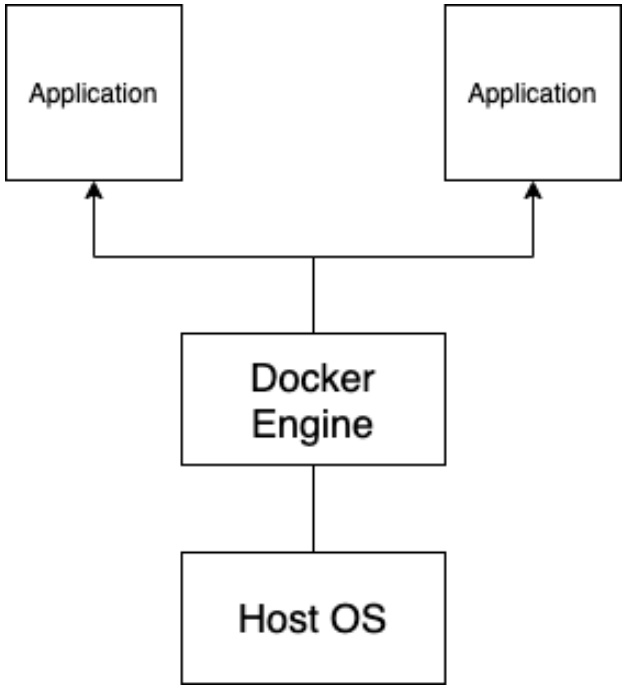

Figure 4. Schematic of a Docker container setup

In this paper, the method of evaluation is that of analysing the CPU and memory utilization of an operating system image running on top of the container. As LXD is simply an extension of LXC, the comparison here is done only between Docker and LXD.

The parameters for evaluation were:

- CPU performance
- Memory performance

For LXD, an Ubuntu VM was used for the entire setup, with the help of multipass.run. Firstly, lxd was installed on the VM and then the container was set up, with a CentOS 8 image being installed in it. Finally, sysbench was run on that container. For the Docker setup, Docker was installed on the same Ubuntu VM, and then a CentOS 8 image was deployed on that container. Then, sysbench was performed on the Docker image as well.

A flow diagram showing the testing procedure is shown in the below figure:

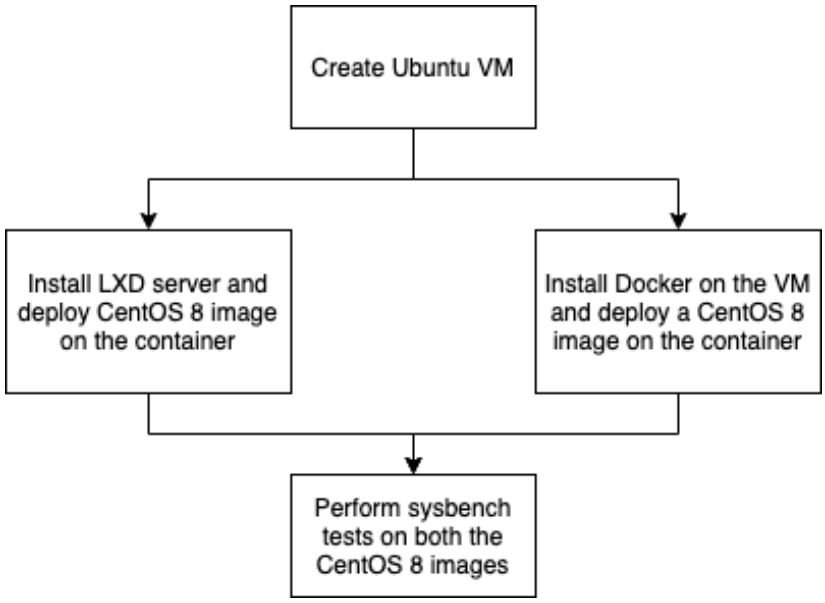

Figure 5. Flow of the tests conducted on LXD and Docker

5. FRONTEND TOOLS

Django and Flask

This section of the paper aims to compare and analyze the Django and Flask frameworks, which are the most commonly used Python frameworks in today's websites.[11]

The testing methodology employs the deployment of a simple web application built in Flask and Django. A simple web application that can do 4 of the basic operations in a database - CRUD (Create, Retrieve, Update, Delete) was stress tested using Locust, a load testing tool built entirely in Python. Locust swarms the website with requests, and returns statistics about the performance of the website while responding to both GET and POST requests. In this paper, the update and delete queries were combined into one testing environment.

A flow diagram of the same is shown below:

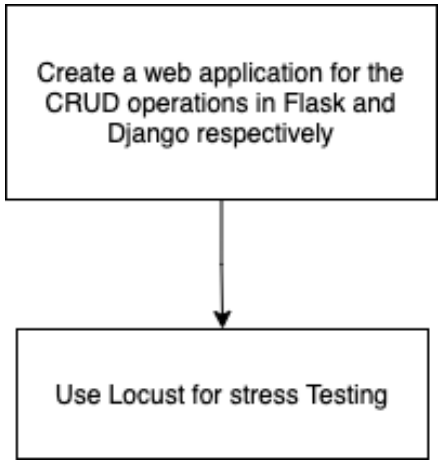

Fig 6. Flow of tests comparing Flask and Django

6. RESULTS

A. PostgreSQL vs MongoDB

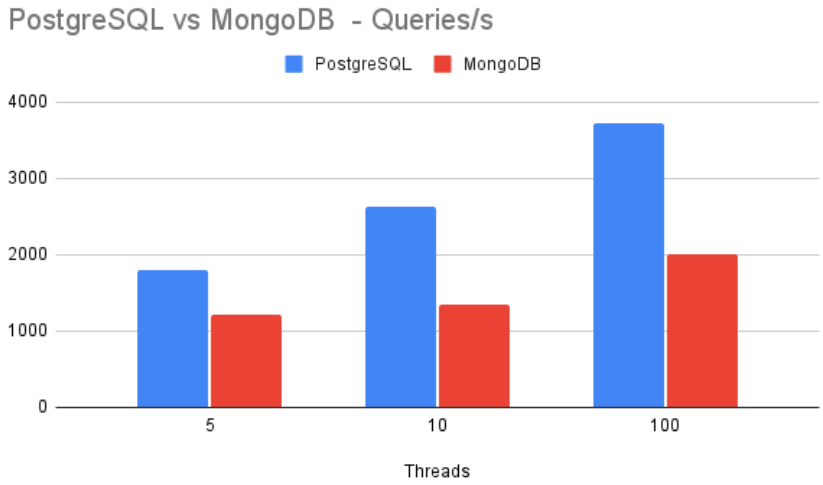

Figure 7 Number of Queries Handled per second in PostgreSQL vs MongoDB

The above figure clearly shows the disparity in the speed with which queries are executed per second. The number of threads were increased as well in the test, and it shows that with the higher number of threads, the performance is boosted by a large margin, however, PostgreSQL is still able to perform much better than MongoDB.

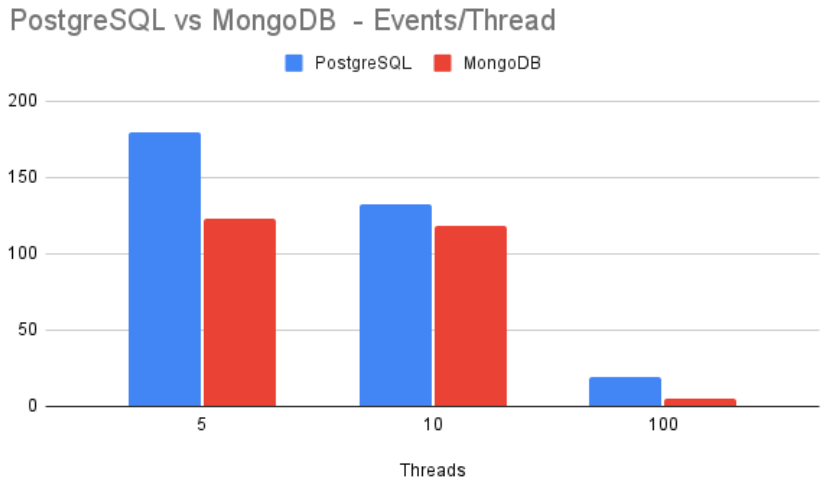

Figure 8 Number of Events per Thread in PostgreSQL vs MongoDB

We can see from the above graph that MongoDB starts to perform almost as well as PostgreSQL in terms of the number of Events executed in each thread, as the difference is almost negligible as the number of threads reaches 100. Also, we can see the saturation of events per thread in both kinds of databases when the number of threads goes to a very high number as is seen above.

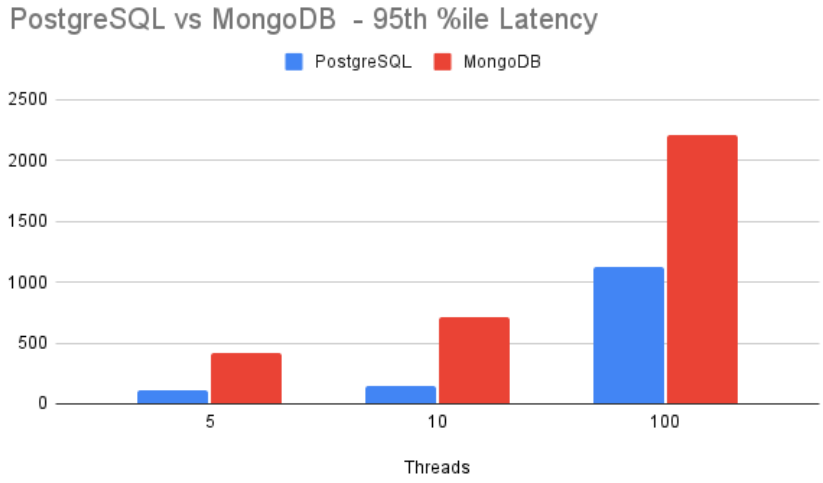

Figure 9. 95th Percentile Latency in PostgreSQL vs MongoDB

A similar expected result is seen here as well in the 95th %ile latency graph, as with an increase with the number of threads. 95th percentile latency is the measure at which 95% of users are experiencing latency lower than it. We can see that a high 95th %ile latency is not ideal. Another conclusion that can be drawn from the above graphs is that as the number of concurrent users (threads, in this case) increases, the MongoDB performance gets better. This is what is expected, based on the literature survey conducted.

B. Docker vs LXD

a. CPU tests

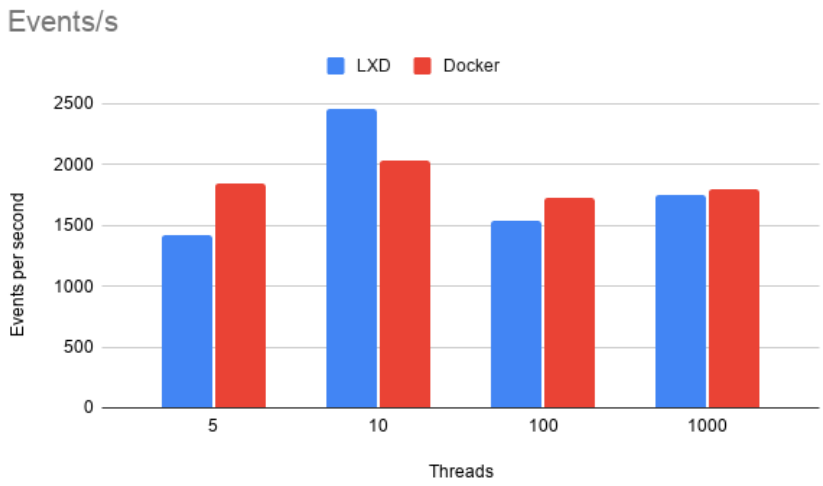

Figure 10. Events executed per second in LXD vs Docker

In the above graph we can see that LXD performs better when the number of threads is lower, but as that number increases, Docker's performance starts to converge to that of LXD's, as the number of threads reach 1000

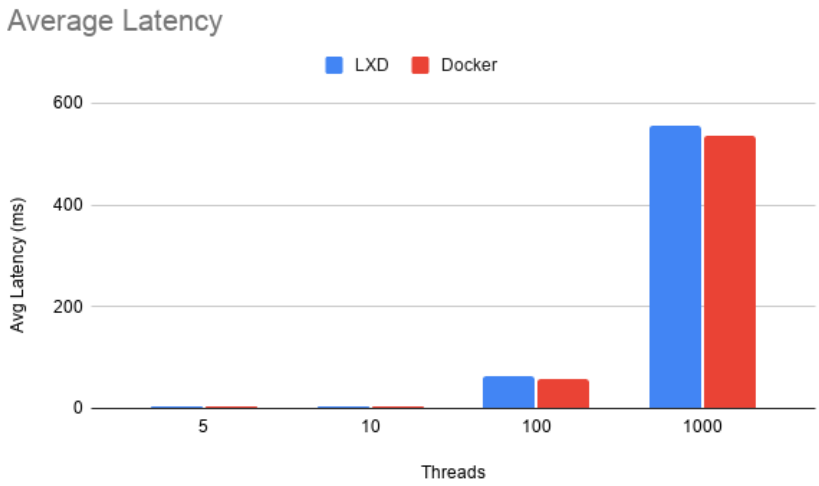

Figure 11. Average Latency in LXD vs Docker

Here, the average latency is negligible in both LXD and Docker, when the number of threads is low. However, LXD performs better by a considerable margin compared to Docker as the number of threads increases. This is consistent with the findings in the literature review.

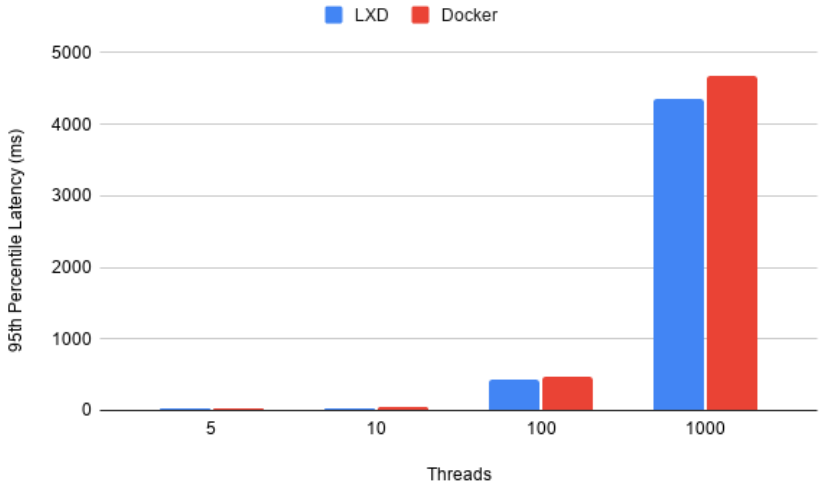

Figure 12. 95th Percentile Latency in LXD vs Docker

Similar to the average latency stat, LXD performs marginally better than docker in terms of 95th percentile latency as well, as is seen in the above graph.

b. Memory Tests

Consistent with the findings in the literature, it is seen that Docker has a higher rate of data transfer compared to LXD, for both read and write operations.
The results for the same are shown in the below figures.

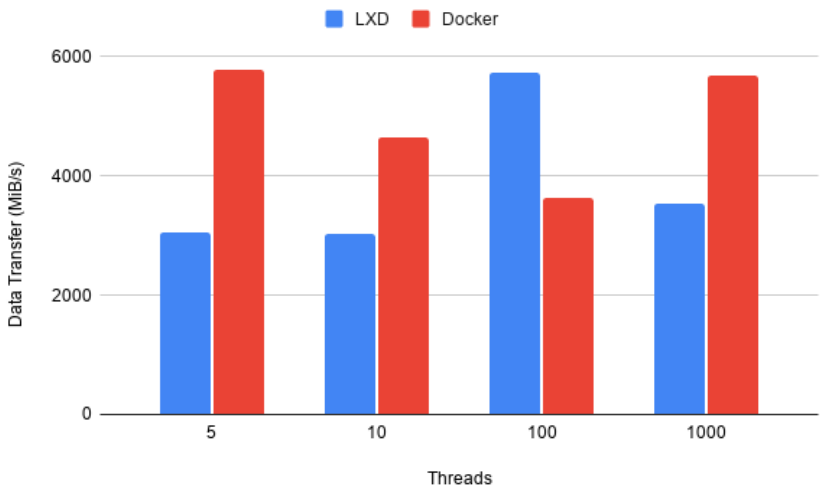

Figure 13. Data Transferred per second while Reading from Memory – Docker vs LXD

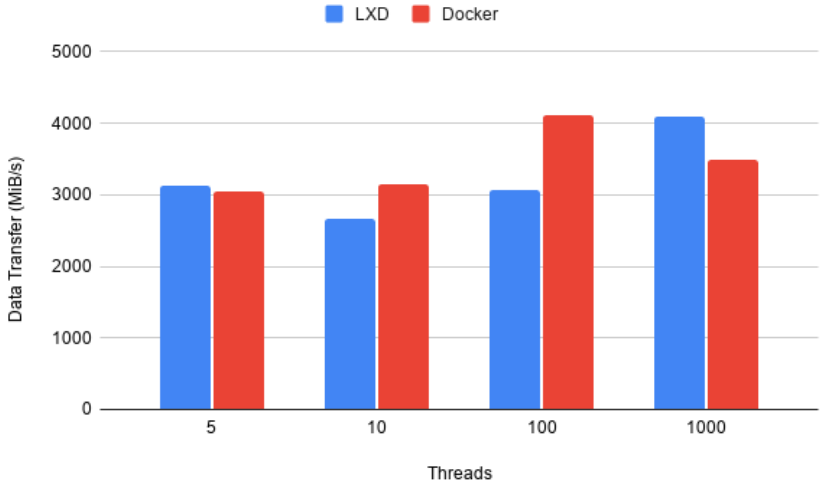

Figure 14. Data Transferred per second while Writing to Memory – Docker vs LXD

C. Flask vs Django

We can see in the below figure that when a record is to be inserted into the database (CREATE), the response time is nearly identical in both Flask and Django.

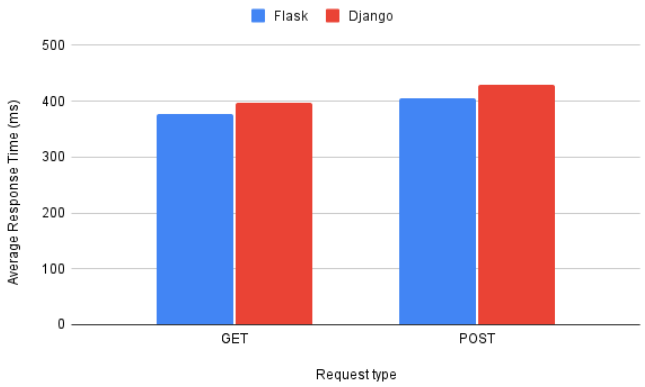

Figure 15. Average Response time of the requests that were sent = Flask vs Django.- CREATE Operation

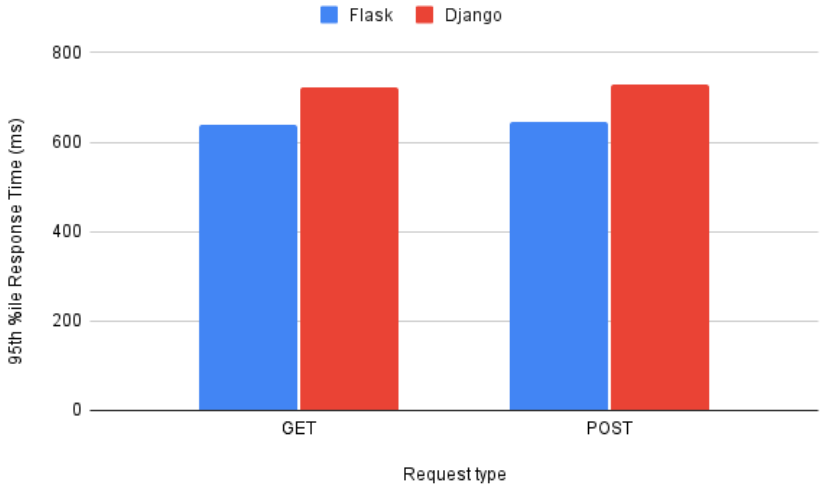

Figure 16. 95th Percentile Response Time in Flask vs Django – CREATE Operation

In the above graph, clearly, the 95th percentile latency is lower in Flask than in Django, which is an indicator of good performance.

Similarly, for the other operations - RETRIEVE and UPDATE:

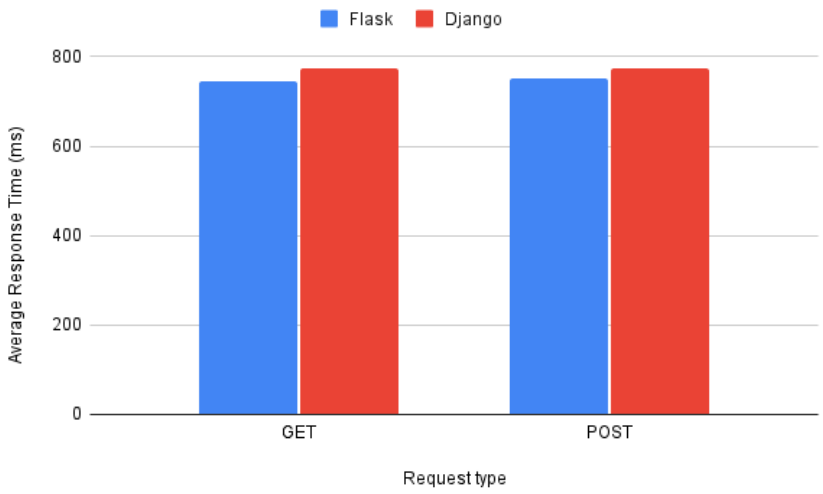

Figure 17. Average Response Time for Flask vs Django – RETRIEVE Operation

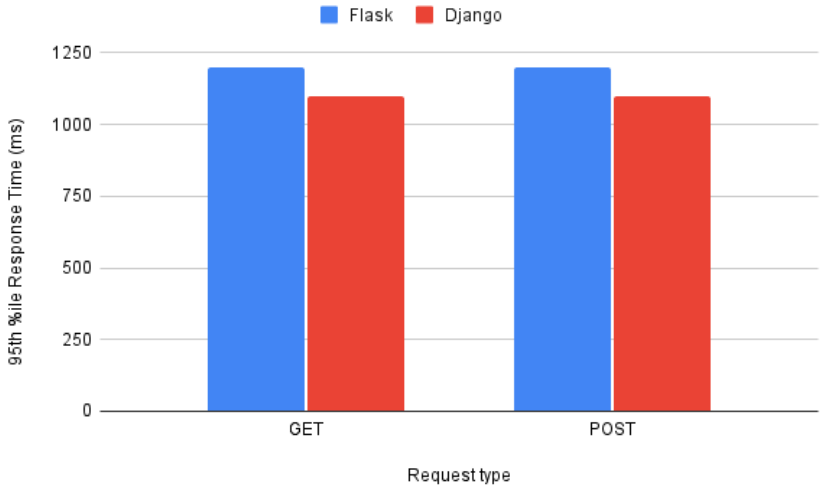

Figure 18. 95th %ile Response Time for Flask vs Django – RETRIEVE Operation

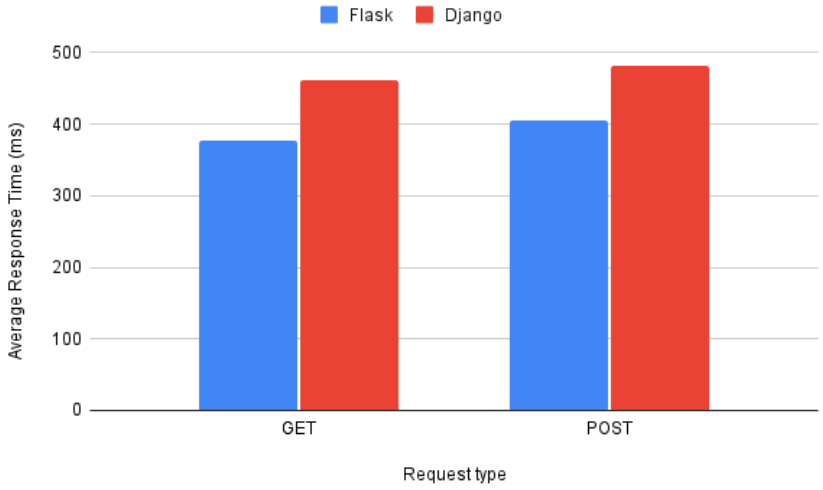

Figure 19. Avg Response Time for Flask vs Django – UPDATE Operation

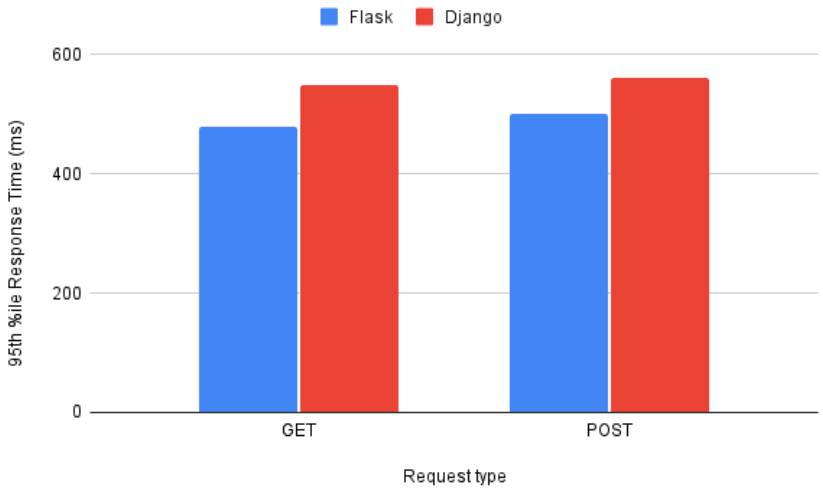

Figure 20. 95th %ile Response Time for Flask vs Django – UPDATE Operation

7. CONCLUSIONS AND FUTURE SCOPE

With the kind of server hardware that is present today, the marginal gains LXD may have in terms of the data transfer rate does not prove to be a huge factor in deciding which tool should be used. Docker has an advantage over LXD as it is much easier to configure and use in every operating system, while LXD can only be set up inside a Linux machine, and as Docker only installs necessary libraries instead of the entire OS, it would be a better option.

For website app development which needs to be done very quickly, Django is proven to be a better alternative. However for simpler needs, it is necessary to look at the various performance parameters as shown in the above sections, in which Flask almost consistently outperformed Django. While the differences are marginal, Flask is much less cumbersome to set up and is generally more flexible, as is already discussed in the previous sections.

For almost all business scenarios, PostgreSQL would be a better choice compared to MongoDB, as is shown in the above results. PostgreSQL executes more queries per second and also has a lower latency as well. It also seems to perform twice as well as MongoDB when the number of threads is higher, in terms of queries executed per second, while

There are several complex database designs, like polygons, which is one of the only scenarios in which MongoDB seems to outperform PostgreSQL. [17] So in the future, this work could be expanded to more complex designs in database, web applications and so on.

REFERENCES

[1] D. Ghimire, “Comparative study on python web frameworks: Flask and django,” 2020.
[2] A. R. Putri, R. Munadi, and R. M. Negara, “Performance analysis of multi services on container docker, lxc, and lxd,” Bulletin of Electrical Engineering and Informatics, vol. 9, no. 5, pp. 2008–2011, 2020.
[3] M. Moravcik, P. Segec, M. Kontsek, J. Uramova, and J. Papan, “Comparison of lxc and docker technologies,” in 2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA), IEEE, 2020, pp. 481–486.
[4] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, and D. Anagnostopoulos, “Mongodb vs postgresql: A comparative study on performance aspects,” GeoInformatica, pp. 1–26, 2020.

[5] S. Agarwal and K. Rajan, “Performance analysis of mongodb versus postgis/postgresql databases for line intersection and point containment spatial queries,” Spatial Information Research, vol. 24, no. 6, pp. 671–677, 2016
[6] R. Morabito, J. Kj¨allman, and M. Komu, “Hypervisors vs. lightweight virtualization: A performance comparison,” in 2015 IEEE International Conference on Cloud Engineering, IEEE, 2015, pp. 386–393
[7] D. Swersky. (2018). “What is Docker, and why is it so popular?” [Online]. Available: https://raygun.com/blog/what-is-docker/.
[8] A. Sampathkumar, “Virtualizing intelligent river®: A comparative study of alternative virtualization technologies,” 2013.
[9] S. Gupta and D. Gera, “A comparison of lxd docker and virtual machine,” International Journal of Scientific & Engineering Research, vol. 7, no. 9, pp. 1414–1417, 2016.
[10] A. Johari. (2019). “Python and Netflix: What Happens When You Stream a Film?” [Online]. Available: https://medium.com/edureka/how-netflix-uses-python1e4deb2f8ca5#:~:text=Flask%5Cbackslash%3A,Jupyter)%20on%20a%20large% 20scale..
[11] D. Engines. (2021). “DB engines Ranking trend,” [Online]. Available: https://dbengines.com/en/ranking.
[12] StackExchange. (2017). “Intersecting two polygon-layers in PostgreSQL,” [Online]. Available: https : / / gis . stackexchange . com / questions / 215945 / how - to - intersect-two-polygon-layers-in-postgresql-postgis