MCBENCH: A MULTI-CLOUD BENCHMARKING SYSTEM

by

ABDULAZIZ ALABDULJALIL

A THESIS

Presented to the Department of Computer Science
and the Division of Graduate Studies of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Master of Science

March 2024

THESIS APPROVAL PAGE

Student: Abdulaziz Alabduljalil

Title: MCBench: A Multi-Cloud Benchmarking System

This thesis has been accepted and approved in partial fulfillment of the requirements for the Master of Science degree in the Department of Computer Science by:

Ram Durairajan	Chair
Joe Li	

and

Krista Chronister	Vice Provost for Graduate Studies

Original approval signatures are on file with the University of Oregon Division of Graduate Studies.

Degree awarded March 2024

THESIS ABSTRACT

Abdulaziz Alabduljalil

Master of Science

Department of Computer Science

March 2024

Title: MCBench: A Multi-Cloud Benchmarking System

In today's climate, there is a trend of enterprises moving their systems and applications to the cloud, with systems working within multiple cloud providers. However, as the trend continues, there remains a lack of a benchmarking system to adapt benchmark applications to multi-cloud paths. We introduce MCBench, a benchmarking system able to seamlessly work with any application which uses microservices to containerize for easier usability. We also study the performance of different applications in inter-region and intra-region multi-cloud paths, measuring latency and throughput. We show MCBench's performance is consistent whether running a single or many sequentially run applications, and is affected slightly by cross-traffic.

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

In the ongoing “Cloudification” of the Internet, the rapid adoption of multi-cloud strategies by modern enterprises has become a defining trend. Instead of building infrastructure around local servers, companies have sought to connect servers across the world using the cloud. This trend is driven by a several compelling advantages, including competitive pricing, mitigating vendor lock-in risks, achieving global reach, and addressing the imperative need for data sovereignty. A recent industry report underscores the magnitude of this trend, revealing that over 85% of enterprises have already embraced multi-cloud strategies, marking a paradigm shift in how organizations approach their digital infrastructure [10].

However, as the multi-cloud trend continues, enterprises encounter a challenge in the complexity associated with configuring and managing diverse cloud application across multiple cloud providers. The complexity is born out of the heterogeneous compute and network infrastructure across different cloud providers, with the convenience of mixing cloud providers underlined by the difficulty of configuring applications to perform according to different provider standards.

One of the duties of a network administrator is to continuously benchmark their network and identify any anomalies to fix and repair. Running benchmarking applications is one form of measuring a network’s performance. However, the heterogeneous nature of multi-cloud networks makes it difficult to benchmark these networks while navigating the complexity of constructing paths from one cloud provider to another. There is a lack of a universal standard or general system that works with different cloud providers, since such a system needs to conform to each cloud provider's rules. Current benchmark systems have been built for homogeneous networks and cannot measure multi-cloud paths [13].

What is needed is a benchmarking system that works universally with any multi-cloud network and is extensible to allow developers to run their benchmarking applications on it. The system needs to be lightweight, unaffected by cross-traffic, and perform well when under stress running multiple application to save time for developers.

In this work, we will introduce MCBench, a containerized multi-cloud benchmarking system capable of running any application developers wish to benchmark with. The system addresses the issues faced by state-of-the-art, and is lightweight, extensible, and robust against cross-traffic. We also conduct a characterization study that observes MCBench benchmarking many multi-cloud networks through two analyses: (1) A longitudinal report of multi-cloud networks through MCBench; (2) A before/after analysis of introducing cross-traffic to MCBench.

After conducting our analyses, we find that MCBench is able to run one benchmarking application or several sequentially with almost no loss in performance. While there are many outliers in inter-region multi-cloud paths, we find that MCBench does not lose latency or throughput in both intra and inter-region networks. Finally we find that MCBench is able to withstand cross-traffic that consumes 20% of the total bandwidth with only up to 10% increase in latency.

CHAPTER II

BACKGROUND AND MOTIVATION

In this chapter, we will discuss the background of network measurement in both mono and multi-cloud networks, as well as the challenges faced building a multi-cloud benchmark system

2.1 Challenges

Many benchmarking systems have been built to measure a network's performance and pinpoint issues to the network's administrators. The systems strive to be timely and easily manageable for developers to use, with varying degrees of success. However, as enterprises shift their infrastructure from private to cloud to multi-cloud networks, the complexity of measuring networks rises. The difficulty of working with multiple cloud providers while avoiding traffic from multiple sources has born a lack of benchmarks that work with multi-cloud networks. Heterogeneous computing and network infrastructure has created more requirements are needed from these benchmarking systems.

Due to the fast changing climate of network infrastructure, benchmarking systems need be lightweight and easily switch from one system to the next. Systems need to be easily installable, movable, and quick to use. Next, with new benchmarking applications being developed to address the rapid change, the system also needs to be extensible for developers to quickly swap in their applications and run their benchmarks. Lastly, as more multi-cloud networks are used, increased traffic, chances of congestion, and interfering noise occur. New benchmarking systems need to address the problem by being robust against cross-traffic so as to keep benchmarking results pure.

2.2 State-of-the-Art

Current state-of-the-art systems that tried to address benchmarking multi-cloud networks try to simplify the cloud infrastructure rather than work with it. One system, Invisinets [9], has tried to reduce the complexity for cloud operators by removing the abstractions set by the cloud providers. A developer only need to deal with a set of computing parameters through Invisnets API which then translates to the different building blocks of cloud providers. While Invisinets works well with mono-cloud networks, not only does the API require a relationship between the cloud provider and many enterprises, when multi-cloud networks are considered, configuring for thousands of enterprises all with their own set of desired parameters will be unnecessary work for cloud providers.

Other benchmarking systems [4, 5, 6, 7, 11] perform well with multiple multi-cloud networks. easily installable and lightweight. However, they tend to serve one benchmarking purpose and do not allow developers to use their own applications. These systems also require a noiseless path to test and perform poorly when cross-traffic is introduced.

2.3 Requirements of An Ideal Solution

As discussed in the challenges faced (§ 2.1), what is critically lacking in the cloud climate is a benchmarking system able to work across different cloud providers. The benchmarking system has to be lightweight and user-friendly to avoid adding to the complexity inherent in using different cloud providers. The system must also be easily extensible by developers who wish to use their own application benchmarks and accommodate their diverse needs. Lastly, an absence of a benchmarking capability aggravates the critical gap in understanding multi-cloud paths, necessitating a thorough characterization study.

CHAPTER III

OVERVIEW OF MCBENCH

In this paper we introduce MCBench[1]: a user-friendly benchmarking system capable of running on different multi-cloud paths. The system seeks to revolutionize the benchmarking landscape by providing a unified and streamlined approach to managing and configuring application benchmarks in a multi-cloud environment. To build this approach, MCBench leverages micro-services using Docker [2] to significantly lower difficulty barriers, while making the benchmarking system lightweight and extensible. Using Docker empowers any developer to seamlessly add their preferred application of choice. The lightweightness of MCBench can also save enterprises time with the system's quick installation and fast use, and save on cost as Docker is free

3.0.1 Characterization Study. This work also addresses the aforementioned characterization study by conducting two types of analyses. The first involves presenting a longitudinal performance report of multi-cloud paths derived from our benchmarking system, offering valuable insights for enterprise developers. The second analysis introduces cross-traffic considerations, studying the impact of cross-traffic on multi-cloud paths through a before-after style analysis. This holistic approach provides a nuanced understanding of multi-cloud dynamics, enabling organizations to make informed decisions in their pursuit of optimized multi-cloud strategies.

[1]The code and current implementation for this project is in https://gitlab.com/onrg/mcbench

CHAPTER IV
MCBENCH: DESIGN AND IMPLEMENTATION

In this chapter, we describe the details of MCBench's implementation and the methodology of our evaluations, including the tools used and steps taken to conduct our analyses.

4.1 Implementation

To abide by the three standards we set for a capable benchmarking system (lightweight, extensible, and robust), we built MCBench around Docker to be both lightweight and extensible.

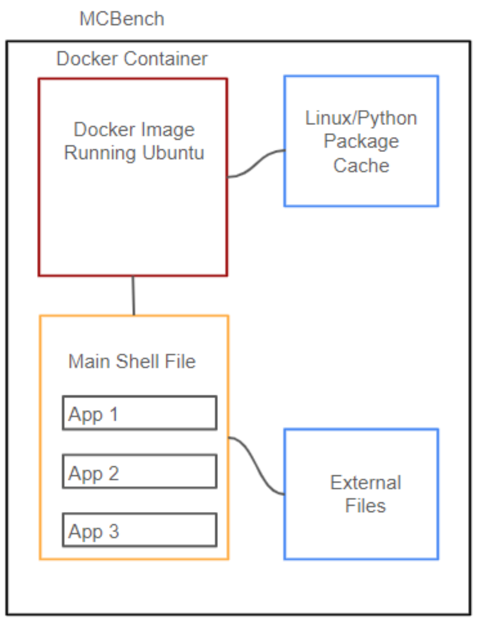

Figure 1. MCBench's architecture

The system's architecture in Figure 1 shows how MCBench is extensible. The Docker image, running Ubuntu Linux, acts as the heart of the system, installing all necessary Linux and Python packages needed to run a developer's applications. Although the first build time proportionally long to the amount of packages installed and their memory, all packages are then cached in the system's storage to enhance subsequent build times and allow developers to tweak their applications between runs without long wait times. The cache is implemented using Docker's code in the dockerfile, telling Docker to install packages, cache them in storage, and remove any unnecessary installation-helping packages to save on memory.

MCBench requires some pre-configuration in knowing the credentials of both endpoints of the path to be benchmarked, including SSH usernames, IP addresses, or whatever an application might need. The Docker image will then read the credentials collected as arguments and insert them into the applications. Because of this structure, developers only need to run the image with the necessary arguments for their applications without needing to interact further with MCBench during runtime.

The main shell file includes the applications themselves, running them based on the arguments inserted while running the docker container. The shell file also imports any external data from files necessary to run the developers' applications. An example would be our PyTorch model application training and testing in separate .PY files before their contents imported into the shell file and sent over an SSH connection to begin benchmarking.

4.2 Measurement Setting

In order to evaluate the performance of MCBench, we have to consider its versatility and endurance. Thus, two analyses were conducted: (1) Performance of multi-cloud paths bench-marking applications; (2) Comparison of impact before/after cross-traffic introduction. To conduct our experiments, we deploy MCBench on pairs of Virtual Machines (VMs) on different cloud providers. Each pair acts as sender and receiver while benchmarking the connection the pair shares.

We conduct our experiments in a series of collective batches, with each pair run bidirectionally to account for path asymmetry, to performance measurements including latency and throughput.

4.3 Scenarios and Cloud Regions

As MCBench is meant to perform on multi-cloud networks, several cloud operators have been chosen to conduct tests on. The evaluations were done in intra-region and inter-region fashions to showcase MCBench's performance in close and far linked networks. For intra-region tests, each cloud provider is connected to a different cloud provider in the same region, and connected to different regions for inter-region tests.

The choices for the regions shown in Table 1 were done due to the cost of renting VMs and close proximity when conducting intra-region tests. Each VM runs on at least 2 vCPU cores, 1 GB of memory, and Ubuntu server 22.04 LTS. We also cap the VMs interface to 100 Mb/s to reduce inconsistent results to a minimum.

4.3.1 Scenarios.

The tests revolve around using three applications/benchmarks: (1) Wrk, an HTTP benchmarking tool; (2) Training and transferring a PyTorch model over SSH; (3) Sending MySQL data using HammerDB. The benchmarks are then run in three different scenarios:

- Single Scenario, where each benchmark is run in a separate evaluation and acts as a baseline.

Cloud Provider	East Region	West Region
GCP	Virginia	Oregon
AWS	Virgina	Oregon
Azure	Virgina	Arizona

Table 1. Cloud providers and their regions

– Sequential Scenario, where the three benchmarks are run one after the other.

– Cross-Traffic Scenario, where cross-traffic is introduced while the Sequential Scenario is running, possibly impacting performance.

4.3.1.1 Sequential Scenario. We run the three applications one after the other in this scenario by including all three applications in MCBench. The trio's performance is compared to the Single Scenario's performance to estimate MCBench's ability to run multiple applications in the same session without much loss in performance.

4.3.1.2 Cross-Traffic. For cross-traffic, we generate traffic through Cisco T-Rex [1], an open-source simulated network traffic generator, and introduce the traffic while MCBench runs its sequential scenario (3 applications sequentially). We simulate simple HTTP traffic during the entire run limiting the cross-traffic throughput to 20 Mb/s, 20% of the experiments' total throughput. The threshold of 20% was decided by testing increasing amounts of cross-traffic on MCBench's performance, with lower thresholds having no effect on the system, and higher thresholds halting the system's performance to unacceptable levels.

4.4 Data Collection

We conduct our tests over a 2 month long period, where we run each VM pair bidirectionally 50 times averaged over both directions. The time conducted for each run is 3.5 minutes on average for Single Scenario runs and 12 minutes on average for Sequential and Cross Scenario runs. Each run is measured to collect latency and throughput measurements. We report the latency using 10 ping probes over 1-second intervals. For packet loss rate and throughput, we use the iperf3 [3] tool configured to transmit data over TCP connections in 5-second intervals.

CHAPTER V

RESULTS

In this chapter, we will use our discussed methodology to observe MCBench's performance in the three different scenarios via intra-region and inter-region paths. We examine the RTT, packet loss rate, and throughput of both single and sequential runs, where sequential runs are three applications run one after the other, and compare them for performance differences. We will also observe the impact of introducing cross-traffic while running a sequential run of MCBench.

5.1 Single vs Sequential Scenarios

We run each application alone and measure their performance. We then run them sequentially and measure their performance collectively. The single applications and sequential runs are compared observe the performance difference when using many applications with MCBench.

5.1.1 RTT. There seems to be no difference in latency in Figure 2 when running the benchmarks on their own versus sequentially in order. As MCBench acts as a container system for running benchmark applications, the lack of difference could be attributed to the applications themselves cleaning and closing connections before the next benchmark is run. The higher average of HammerDB runs could be the result of sending test packets before establishing a connection to decide whether said connection is safe and stable.

The choice of path does not seem to have an effect on the latency of these benchmarks. An assumption could either be the clean construction of these benchmark applications or the lack of a substantial congestion in any data center while collecting data.

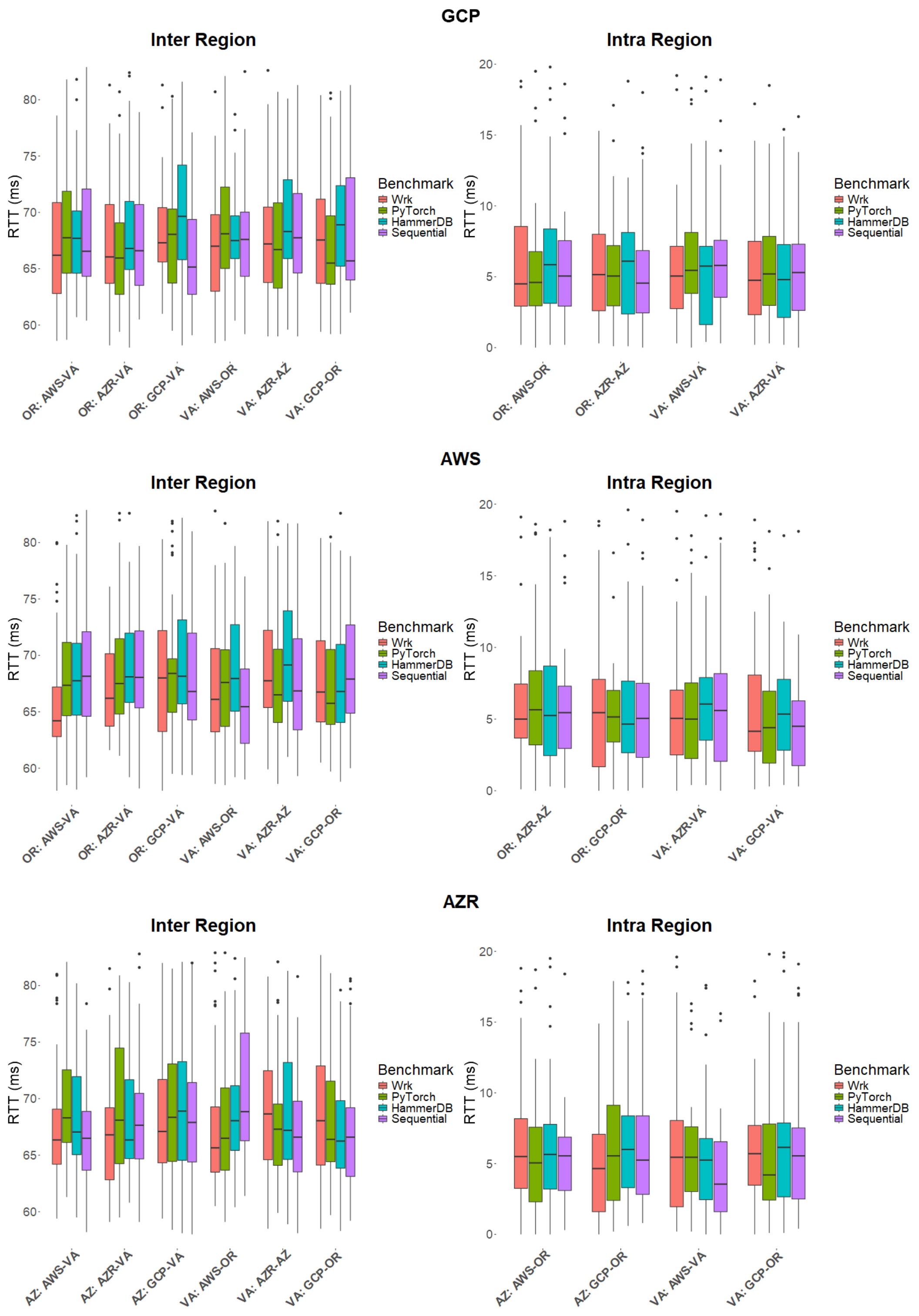

Figure 2. RTT of single application runs vs. sequential application runs

5.1.2 Packet Loss Rate.

Figure 3 similarly exhibits little performance deviations between single and sequential runs. We do notice however many outliers in inter-region paths, with upwards of 5% and 8% packet loss rate, while intra-region paths entertain few outliers. While switching cloud providers does not seem to be a contributor to packet loss, the *distance* of the path can be observed as proportional to packet loss. An assumption could be the longer the distance, the more possible interfering connections along the way, especially with multi-cloud paths.

5.1.3 Throughput.

Figure 4 show the same observations as Figure 2 and Figure 3 with little to no performance difference. Like with packet loss, throughput sees a lot of low outliers in inter-region paths. We can assume the cause is the same as for packet loss, long distances allow for more chances for the path to be hit with noise and congestion.

5.2 Cross-Traffic Scenario

Figure 5 shows that there is little difference in performance runs with no cross-traffic is on average better by 5 ms for inter-region paths and 0.5 ms for intra-region paths. The results show MCBench is relatively robust against cross-traffic, the benchmarks performing without an issue. Cross-traffic consuming 20% of the available bandwidth while seeing only a 7% increase in latency in inter-region paths can be chalked up to everyday traffic increase. The 10% increase in intra-region paths however can be challenging to ignore depending on the application used. Lowering the network consumption of MCBench by establishing longer wait times between applications and make closing connections mandatory after each application runs can probably address the issues see here.

5.3 CP & Region Trends

During the first month of data collection, we observed a congestion in each path connecting to Azure's Virginia data center that caused high latency and low throughput. We tried to change the day and time we conducted our experiments, yet the irregularity persisted for an entire month before resolving on its own. Another trend we observed during the second month was the relatively higher latency in the Arizona region, again an Azure data center. While the difference in latency between regions was much lower than the Virginia congestion, it was still interesting to see both problems occur within Azure's data centers.

5.4 Summary of Key Results

Our work gives us several key observations for multi-cloud networks. The portability of MCBench in a Docker container allows for quick installation and easy handling, making the system lightweight while solving the management complexity faced with multi-cloud networks. Figure 2, 3, and 4 show that MCBench has no performance loss when running one application vs. several at the same time, confirming MCBench is extensible to other applications with no issue. Lastly, Figure 5 shows no influence of cross-traffic on MCBench, showcasing that MCBench is robust against cross-traffic.

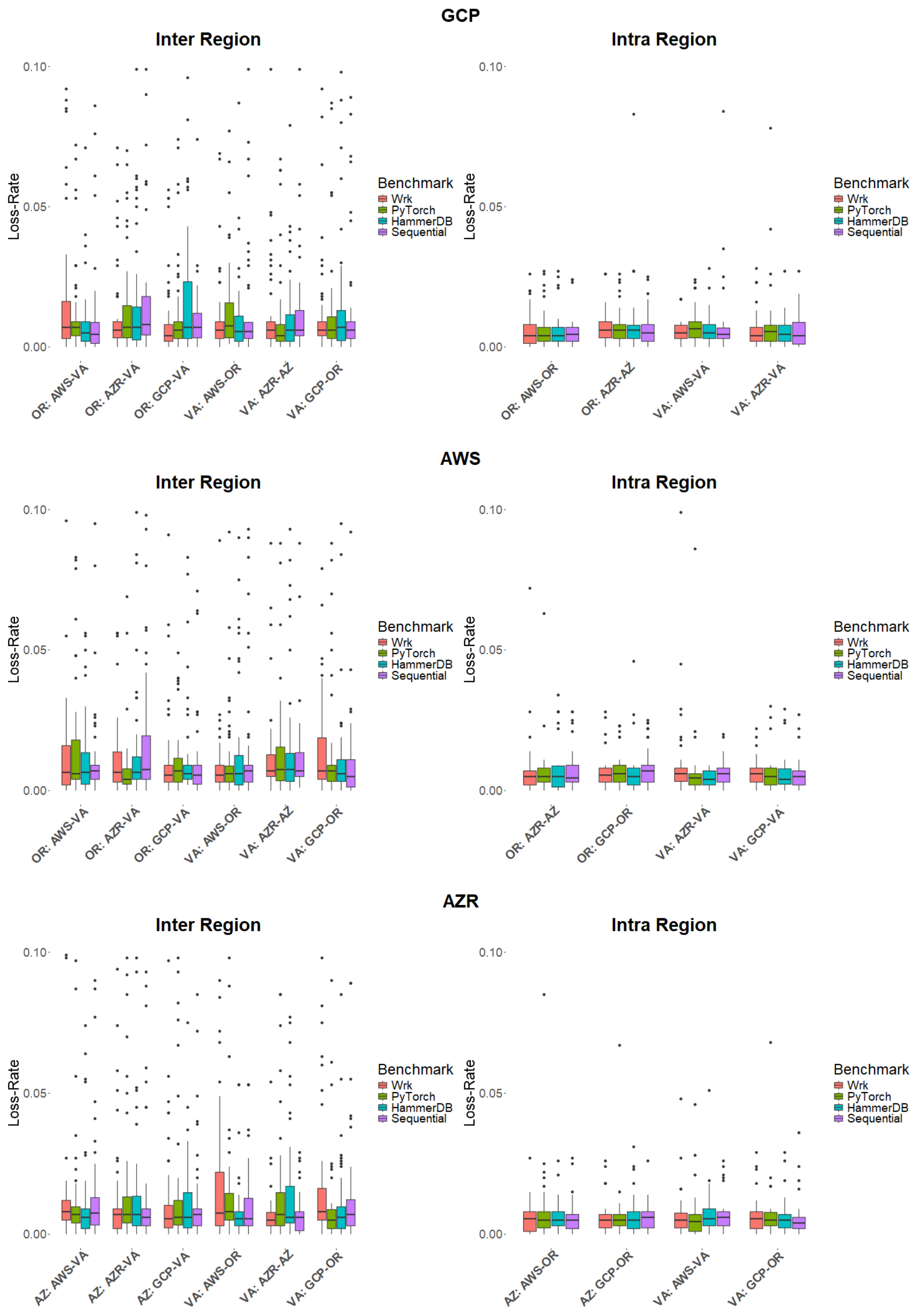

Figure 3. Packet Loss Rate of single application runs vs. sequential application runs

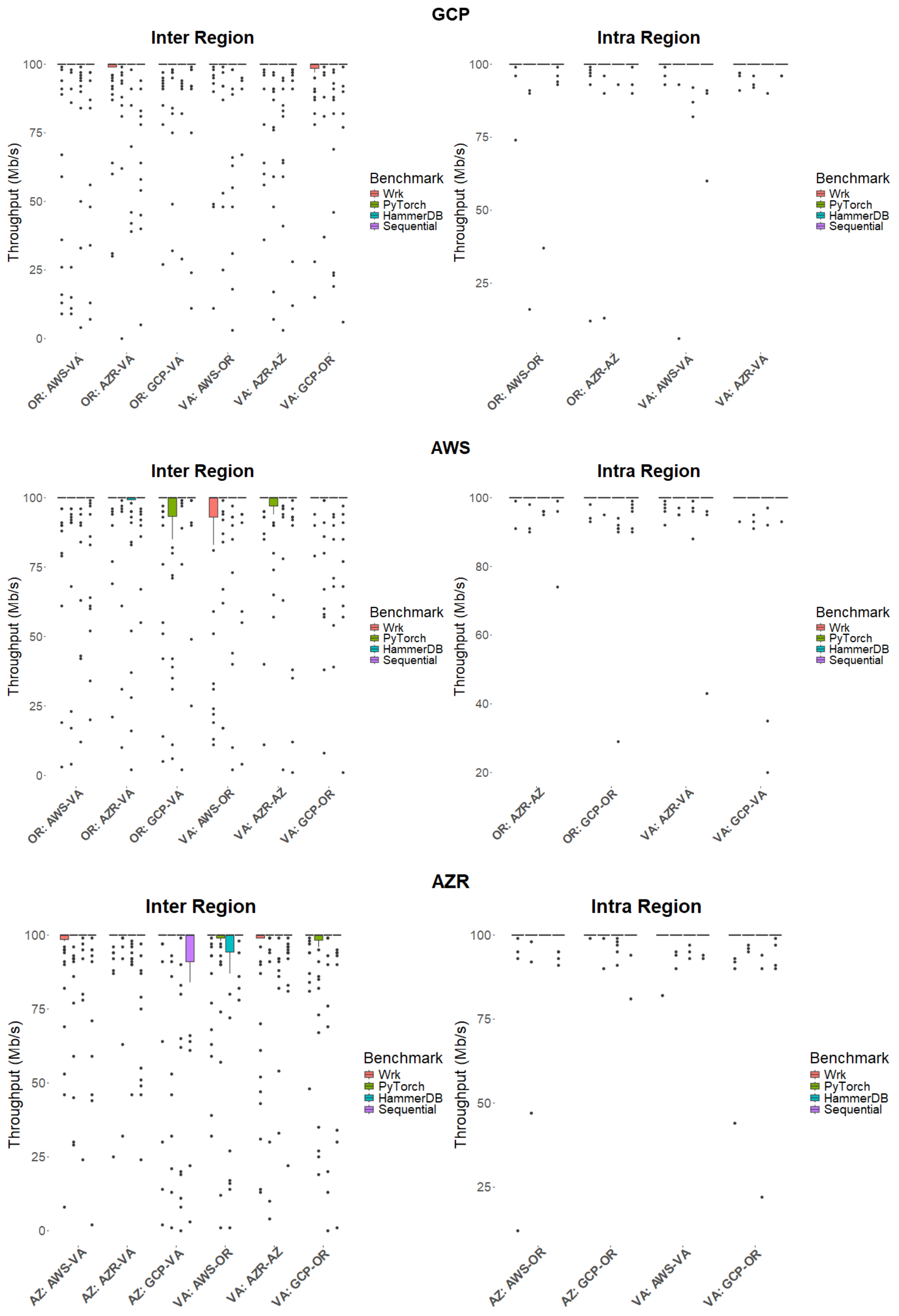

Figure 4. Throughput of single application runs vs. sequential application runs

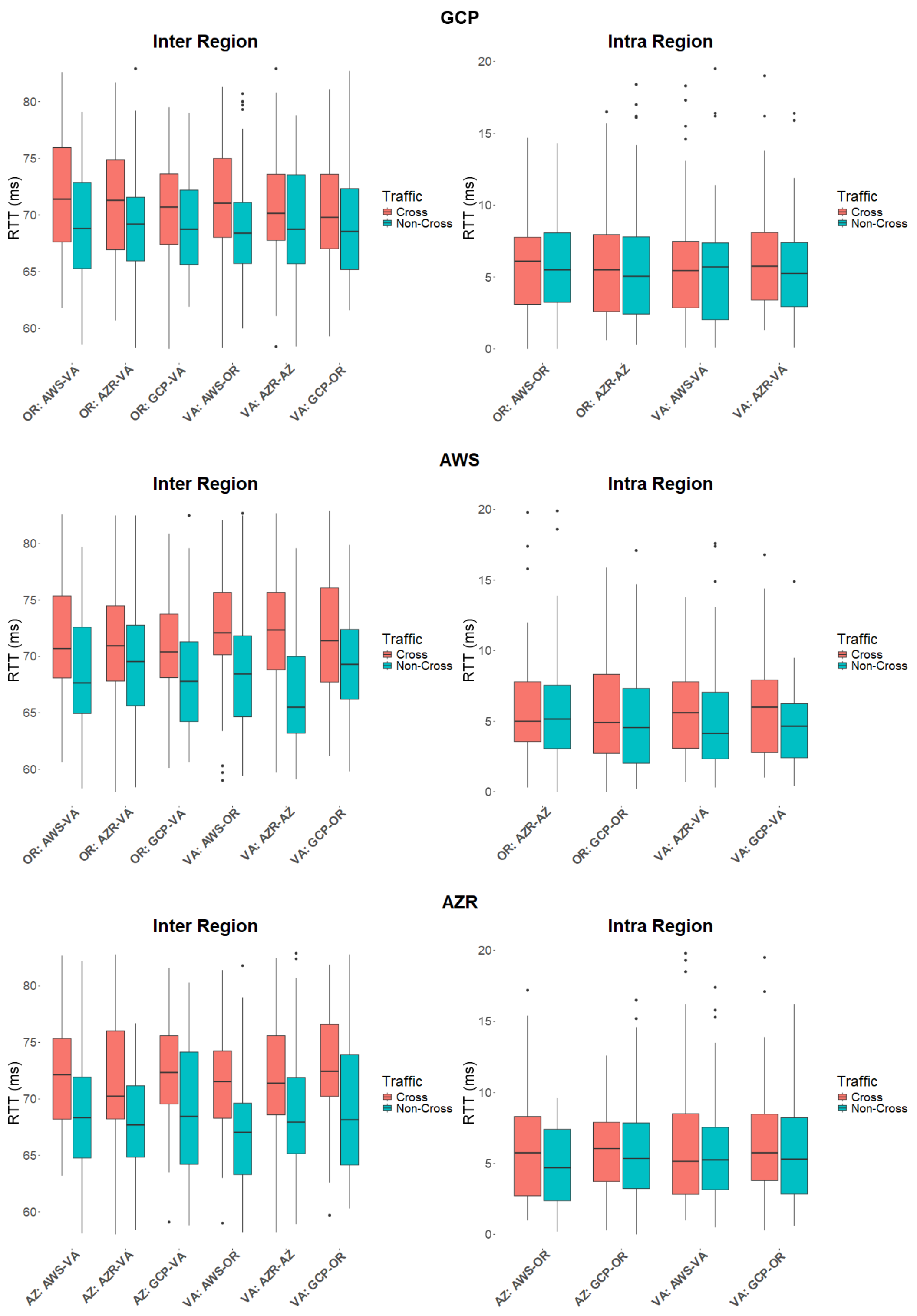

Figure 5. RTT of sequential application runs with cross-traffic vs. no traffic

CHAPTER VI

DISCUSSION AND OUTLOOK

In this chapter we will first discuss an outline of introducing path optimization to MCBench, using the most optimal path in the multi-cloud network. We will then discuss further studies on different methods for path optimization to integrate into MCBench, as well future work that can improve MCBench from a user-friendliness standpoint.

While MCBench is able to run benchmarking applications over multi-cloud networks, the system is currently unable to switch to another network path if it faces congestion or a large packet loss rate. Even if path hopping is currently out of this work's scope, we would like to mention works that are exploring the subject and how their solutions could integrate nicely with MCBench.

6.0.1 Path Optimization. A recent characterization study [12] has been exploring the differences between public, private, and cloud internet routes in terms of latency and throughput. While conducting their evaluations, the authors use different path discovery methods to "maximize the amount of responsive hops" along the way using both *scamper* [8] and ICMP probes. The inclusion of these tools within MCBench's initialization can help find an optimized path for benchmarking before running its applications. Developers who wish to test a clean noiseless path or their network at its current best could use a path optimization path. A yet unpublished paper currently being written, *Stratocore*, is researching different path optimization methods that change networks in real-time to meet demand. The techniques described within the paper can possibly be integrated into MCBench so benchmarks change to an optimized path their studying during runtime without disruption to avoid cross-traffic and growing congestion.

6.0.2 Enterprise Cross-Traffic Thresholds. In our analysis, we determined MCBench has little performance loss with a 20% bandwidth of cross-traffic. However, that 20% threshold only acts as a baseline for MCBench's performance. The next to take would be discussing with developers in the industry on what is the average tolerable performance loss a company will accept for their network. Those discussions could be used to test MCBench's limits by increasing the cross-traffic threshold and observing if the system's performance falls within the discussed acceptable levels.

6.0.3 Docker Optimization. As MCBench is built right now, a developer needs to install their application to the system and any packages it requires to run. These packages are cached when installed to prevent a long build time after the first build, saving developers time when making changes to their application. MCBench's current flaws lie in the first build time being too long and the amount of memory the image takes when built. We used three applications (1) Wrk, (2) PyTorch, and (3) HammerDB for our evaluations, including the packages needed to run them. Using all three applications, it took a staggering 12 minutes to build the docker image the first time and 9 GB of space. While later build times averaged 2.5 seconds, the 12 minutes might be too much to ignore for developers.

A proposed solution to this problem could be looking at pulling Docker images containing those packages and copying them over to local storage. These Docker images must be prepared ahead of time, but if the most used packages were already installed in them, time would be saved by copying these images rather than installing the packages again in MCBench. Another solution is to use Docker Slim to produce smaller containers and an image with less layers.

CHAPTER VII

CONCLUSION

In this thesis we built a user-friendly benchmarking system MCBench using micro-services that is able t navigate the complexity of measuring the performance of multi-cloud networks. The system is lightweight, extensible to other applications, and robust against interfering cross-traffic. The system is made to run using simple commands, and allows developers to use their own benchmarking applications. We ran evaluations of MCBench that found its consistent performance across multiple cloud providers, regions, and applications. Finally we proposed several studies and ideas to improve the system including possible optimized path detection and a lower Docker first build time.

REFERENCES CITED

[1] Cisco t-rex. `https://trex-tgn.cisco.com/`. Accessed: 2024-02-25.

[2] Docker. `https://www.docker.com/`. Accessed: 2024-02-25.

[3] iperf. `https://iperf.fr/`. Accessed: 2024-02-25.

[4] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):242–253, aug 2011.

[5] Theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. Cloudnaas: a cloud networking platform for enterprise applications. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SOCC '11, New York, NY, USA, 2011. Association for Computing Machinery.

[6] Michael Dalton, David Schultz, Jacob Adriaens, Ahsan Arefin, Anshuman Gupta, Brian Fahs, Dima Rubinstein, Enrique Cauich Zermeno, Erik Rubow, James Alexander Docauer, Jesse Alpert, Jing Ai, Jon Olson, Kevin DeCabooter, Marc De Kruijf, Nan Hua, Nathan Lewis, Nikhil Kasinadhuni, Riccardo Crepaldi, Srinivas Krishnan, Subbaiah Venkata, Yossi Richter, Uday Naik, and Amin Vahdat. Andromeda: performance, isolation, and velocity at scale in cloud network virtualization. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, NSDI'18, page 373–387, USA, 2018. USENIX Association.

[7] Daniel Firestone. VFP: A virtual switch platform for host SDN in the public cloud. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 315–328, Boston, MA, March 2017. USENIX Association.

[8] Matthew Luckie. Scamper: a scalable and extensible packet prober for active measurement of the internet. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, page 239–245, New York, NY, USA, 2010. Association for Computing Machinery.

[9] Sarah McClure, Zeke Medley, Deepak Bansal, Karthick Jayaraman, Ashok Narayanan, Jitendra Padhye, Sylvia Ratnasamy, Anees Shaikh, and Rishabh Tewari. Invisinets: Removing networking from cloud networks. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 479–496, Boston, MA, April 2023. USENIX Association.

[10] Pluralsight. State of cloud 2023. Technical report, Pluralsight, 2023.

[11] Priya Sethuraman and H. Reza Taheri. Tpc-v: A benchmark for evaluating the performance of database applications in virtual environments. In Raghunath Nambiar and Meikel Poess, editors, *Performance Evaluation, Measurement and Characterization of Complex Systems*, pages 121–135, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[12] B Yeganeh, R Durairajan, R Rejaie, and W Willinger. A First Comparative Characterization of Multi-cloud Connectivity in Today's Internet. *Lecture Notes in Computer Science*, 12048:193–210, Mar 2020.

[13] B. Yeganeh, R. Durairajan, R. Rejaie, and W. Willinger. A case for performance- and cost-aware multi-cloud overlays. In *2023 IEEE 16th International Conference on Cloud Computing (CLOUD)*, pages 560–566, Los Alamitos, CA, USA, jul 2023. IEEE Computer Society.