Head-to-Head: Which is the Better Cloud Platform for Early Stage Start-up? Docker versus OpenStack

Yi-Zong Ou
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: ou9@illinois.edu

Jung-Chen Chen
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: jchen186@illinois.edu

***Abstract*—The cloud platform has been the top choice when technology startup companies choose the best platforms to deploy their systems. This paper presents a series of decision procedures for choosing the Infrastructure as a Service (IaaS) in cloud computing from the perspective of technology startup companies. We target startup companies as potential users. We have developed a decision tree for assisting entrepreneurs and developers to choose the IaaS according to performance and usability of Docker versus OpenStack. We use the Yahoo! Cloud Serving Benchmark (YCSB) as a benchmark for key-value storage. The target databases include MongoDB and Cassandra. The decision assists the technology startup companies to choose the cloud platform with the best fit while developing and deploying the web services.**

I. Introduction

The decision to choose a platform is hard. Choosing from two popular and outstanding cloud platforms is especially hard. Cloud computing technologies have been noticed since the year 2000. Today, more and more cloud technologies are being developed, ranging from container based virtualization to distributed data streaming processing for social network and big data. Legacy classification of technologies falls into three categories, from bottom to top they are: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). When a developer starts to develop and deploy a service. A major question comes to mind is where and what platform to deploy the service. The decision is critical because the platform enables a good service even batter in terms of good response time from uses' perspective and time, and budget saving from companies' perspective. On the hand, choosing a wrong fit technology might potential lead a catastrophe for the company including shortage manpower for maintaining the service, over budget for cloud service, etc.

Start-up companies have been a main thrust for adopting the latest technologies. For most start-up companies, they need to make decision about the system implemented from the very beginning. Choosing the platform for their service is essential for providing a successful service. The reason is that start-up companies are typically lack of resources, especially for the early-stage start-up companies. Developers would put more time on the service itself instead of putting time on constructing, tuning and maintaining the infrastructure. A variety of cloud services provide attractive services to run the service application in the cloud, such as Amazon EC2, Microsoft Azure and Google Cloud Platform. However, some developers would want to build their private cloud to fine-grain customized configuration for their demands. It can achieve by leveraging private cloud technology, such as OpenStack.

This paper presents a decision tree for assisting developers to choose the best fit cloud platform technologies according to their needs. We select two emerging platforms, OpenStack and Docker, as rivals for comparison. We adopt Yahoo! Cloud Serving Benchmark (YCSB) to evaluate the performance on key-value store. In addition, we use the metrics from software engineering perspective to evaluate the usability of the two targets platform.

Our contributions are:

- A decision tree for start-up companies to select the cloud platform
- Performance evaluation on MongoDB and Cassandra on OpenStack and in a Docker container based on YCSB benchmark.
- Identify the potential needs for start-up companies to develop web services.

The paper is organized as follows. Section II examines the related work of cloud platform, benchmark and decision tree technologies. Section III provides our motivation scenario. Section IV presents our research methodology of conducting experiments. Section V presents experiments configuration and experiment data. In Section VI, we present the decision tree for choosing cloud platform, and Section VIII shows our conclusion.

II. Background and Related Work

How to utilize cloud infrastructure to run an application has become an important decision for building successful cloud application. Modern cloud infrastructure uses virtualization to isolate applications and on-demand allocate the resources. With system hypervisor, users can create virtual machines and allocate the resource. However, it is expensive for a hypervisor to run multiple OS on a single physical machine to satisfy the isolation between each applications [1]. By contrast, a container is a light weight process level virtualization which improves performance and reduces the start-up time [2]. Many papers have studied the performance of virtual machines compared with the Linux container [1], [3]–[7]. Containers perform better than or equal to virtual machine in almost all cases [4]. The issues with virtual machines are the start-up speed and running a whole system in order to get isolation [6]. Docker is an open source platform and relies on Linux containers (LXC) with high level APIs, which provides a faster way for developers to deploy applications inside containers [8]. Since Docker images do not need a complete boot of a new operating system, it provides a lightweight approach to run application on shared compute resources [9]. The research [7] has shown a significant difference between boot speed for a KVM hypervisor and a Docker container. Furthermore, Docker Machine can help developers to set up Docker on a variety of cloud platforms quickly [10]. Docker SWARM [11], a native clustering tool for Docker, is recently released for gathering several Docker Engines into a single, virtual host. It allows swapping in powerful backends for scaling production deployments . Similar to Docker, OpenStack is an open source software that allows developers to deploy cloud infrastructure and provide tools for managing virtual machines. In addition, OpenStack is scalable and equipped hypervisors to support visualization [12]. This makes OpenStack an important role as a cloud provider [13], [14]. Since OpenStack is one of the most popular open source projects as well as Docker, it is hard to determine which platform is better [15]. The decision can be done by the users' preference of storage style and how the infrastructure is to be deployed [16]. The performance evaluation of OpenStack with Hadoop shows the results that virtual network performance of the multi-host plan has been greatly improved compared to the single-host plan [17]. The performance between Eucalyptus and OpenStack are compared by the BYTE UNIX benchmark suit [18].

There are numerous benchmarks for examining the cloud platform from different perspectives. Among of them, YCSB [19] is one of the well-known benchmark of key-value data store. YCSB supports most of key-value store databases. It provides multiple types of workload which represent different scenarios of the key-value store. It also provides users to write script for customized workload. We use YCSB as the benchmark in our experiments.

Choosing one among candidates is always hard. Using decision tree is one method to support the decision making process. Many research areas has this common demand. For example, the paper [20] is about whether to use P2P technology for the service. The paper [21] uses the concept from decision tree to construct a robust face detection. We use a decision tree to support developers to select the cloud platform for their developed services.

1) Docker: Docker is an open platform which is designed to develop, ship, and run applications faster by combining a lightweight container virtualization platform with workflows. There are two main components of Docker. The first one is Docker which is the open source container virtualization platform. The second one is Docker Hub which is the Software-as-a-Serive Platform used for sharing and managing Docker containers. Docker is written in Go and the architecture of Docker consists of a client and a server as shown in Figure 1. The Docker client is the user interface to Docker and user interact with the Docker daemon through the Docker client. Docker client can run on the same host with the Docker daemon, or connect a Docker client to a remote Docker daemon via sockets or a RESTful API. A Docker image is used to create Docker containers and consists a series layers which Docker uses union file system to combine these layers into a single and coherent image. The union file systems can building blocks for containers and the variants used for this task including AUFS, btrfs, vfs, and DeviceMapper. As a result, Docker can be a lightweight container by adding or updating the specific layers rather than replacing the whole image. Docker registries are the distribution component of Docker for holding images. A Docker container is created from an image and includes an operating system, user-added files, and meta-data. In addition, it is an isolated and secure application platform which can run an application.

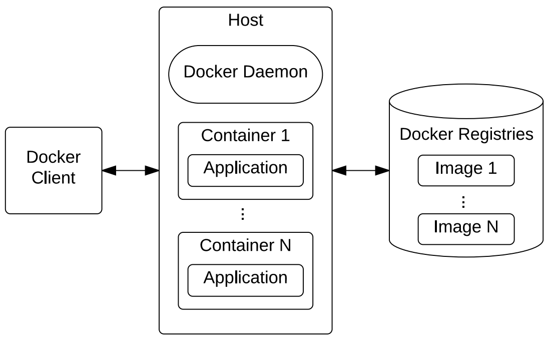


Fig. 1: The architecture of Docker

2) OpenStack: OpenStack is an open source cloud computing platform. In the legacy classification of cloud computing technologies, OpenStack falls into the category of Platform as a Service (PaaS). Openstack is currently managed by OpenStack Foundation and developed by developers world-wide. OpenStack is a collection of tools to manage the cloud computing environment. In this section, we introduce a few major components of OpenStack. OpenStack is organized by nine key component. Nova is also called computing node. It is for hosting the virtual machines. Swift is dedicated for objects and files storage. Cinder is for block storage management. Neutron node is related to network service. Horizon is the web interface for managing the whole system, such as creating virtual machine, configuring access group. Glance provides the snapshot services of the virtual machine. Users can use the snapshot of a machine to create another virtual machine with same environment. This is similar to the user of Docker which can ship and run the container on other Docker system.

A. MongoDB

MongoDB is an open-source document-oriented database with a data structure composed of field and value pairs. A record in MongoDb is a document which is similar to a JSON object with dynamic schemas. MongoDB supports for embedded data models so that it reduces I/O activity to reach high performance. MongoDB uses sharding to deal with large data sets and provide high throughput operations. Sharding is to scales horizontally that divides the data sets and then distributes the data across servers. Each server is an independent database and all of the server gather together to be a single logical database [22].

B. Cassandra

The Apache Cassandra database is a row-oriented database [23]. Cassandra is designed to support large data sets in an efficient way. Cassandra is capable of running on multiple transparently. Cassandra stores data as multidimensional hash table. It is useful to add or change features without disrupting service. This schema-free characteristic can support an Agile development. Cassandra uses virtual nodes, which enables to rebuild a dead faster and improves the heterogeneous machines in a cluster. That is, users can decide a a certain number of vnodes to smaller or larger machines [24].

III. Motivation and Problem Statement

It is difficult for developers or technology start-up entrepreneurs to choose the appropriate system because of the large variety when they consider virtualization technology platforms for hosting their web services. Successful start-up companies such as Facebook and Twitter have millions of users and comprehensive cloud infrastructure to sustain their growth. For those startups who do not have sufficient human and financial resources, it is necessary for them to select the most suitable cloud infrastructure technology to create and deploy their cloud services. Furthermore, those early-stage start-ups may face shortage of manpower with relatively stringent development time. In order to choose a feasible cloud platform, it is important to consider the performance, feasibility, usability, scalability and maintainability. Application developers have to match their workload requirements to the best suited cloud data serving system. Besides, they needs to examine the trade-off between various conditions. These motivates us to evaluate the performance of candidate cloud infrastructures: Docker and OpenStack. Moreover, a decision tree can help the users to choose the most suitable cloud infrastructure. By following the proposed decision tree, developers and entrepreneurs can select the best-fit cloud infrastructure.

IV. Methodology

In this paper, we present the research methodology to construct the decision tree. We examine the different architectural decisions that are made by cloud system. Then, we decide the degree of evaluation and focus on its effect on performance and usability. After benchmarking the systems, we explain the benchmark results. Finally, we choose the metrics for evaluating the usability to make our decision tree completely well organized.

A. Degrees of Evaluation

We choose two representative degrees for evaluation: performance and usability. For start-up companies, they may not have sufficient traffic to sustain a huge private infrastructure at the early stage. Though, they can use the public cloud, companies would have their own private cloud under the control of the IT department. With private cloud, it can maximize and optimize the utilization of existing tools behind the firewall. In this case, they needs to make decision to choose a cloud platform for their services. For start-up companies that devote into the social network and data processing, they also use key-value store as their database to provide their service. On the other hand, usability is essential for a successful service. By classifying a series of metrics, it can choose the candidates of the components based software development [25]. We will give a set of metrics to construct the decision tree in the following section.

B. Performance Evaluation

We choose the YCSB benchmark for evaluating the performance of OpenStack and Docker. The YCSB is the open source benchmark tool that facilitates performance comparisons of the cloud systems which provide online read and write access to data. The YCSB Client, an extensible workload generator, which can be used to load datasets and execute workloads across a variety of systems. Besides, YCSB offers a set of Core workloads that are basic benchmarks for cloud systems and these workloads can be executed by the YCSB Client. Table I shows the configuration of YCSB Core workloads [19].

C. Usability

Usability can be divided into five categories according to ISO 9126 including Understandability, Learnability, Operability, Attractiveness, and Usability Compliance. Each of them is defined in the ISO 9126. Based on the research work [25], we focus more on the metrics especially for small scale start-up companies. Fully understanding the demand of software development for start-up companies that are beyond the scope of this paper. We use heuristic approach to decide the possible metrics to measure cloud platform for the small companies. The chosen metrics include:

- Quality of Documentation: The quality metrics include contents of manuals, effectiveness of manuals, content of demos, contents of help system, and effectiveness of help

TABLE I: Workloads of YCSB Benchmark

Workload	Description	Ratio	Purpose
A	Update heavy workload	Read/update ratio: 50/50	Session store records recent actions
B	Read mostly workload	Read/update ratio: 95/5	Photo tagging; add a tag is an update, but most operations are to read tags
C	Read only	Read/update ratio: 100/0	User profile cache
F	Read-modify-write	Read/read-modify-write ratio: 50/50	User database

system. We will evaluate these metrics based on the materials provided in the relative websites, demos, and community forums.

- Complexity of the Design: The complexity metrics include ease of building the platform on the physical machine, requirement of physical machine, and maintainability.

D. Decision Tree

Decision tree has been used in many decision support system [26], [27]. For example, a factory uses decision tree to decide the investment on assets. Some decision trees use probability to decide the favor branches̃citeyam2001intelligent. By using the probability module, users measure the expectation values and makes the decision based on the measured values. Rather than using the probability model on the branches to create the decision tree, we aim to develop the generic decision tree for start-up companies to choose their suitable platform. Besides, a leaf of the decision tree is a choice to select one of the best cloud platforms. As a result, users can simply follow the decision tree to decide which cloud platform is suitable for their needs and enable them to perform their service in the future.

As Figure 2 shown, the procedures for constructing the decision tree are maded by analyzing a variety of the software development requirement. The requirements facilitate the metrics for both performance and usability. In addition, we decide the both performance and usability metrics in order to construct the decision tree later. After we create the metrics, we run a set of experiments based on the selected benchmark on each platform. We examine it whether the collected data is sufficient to construct the decision tree. If it is insufficient to construct the tree based on the experiment results. We examine what the missing information is. Finally, we address how to construct the decision tree in Section VI.

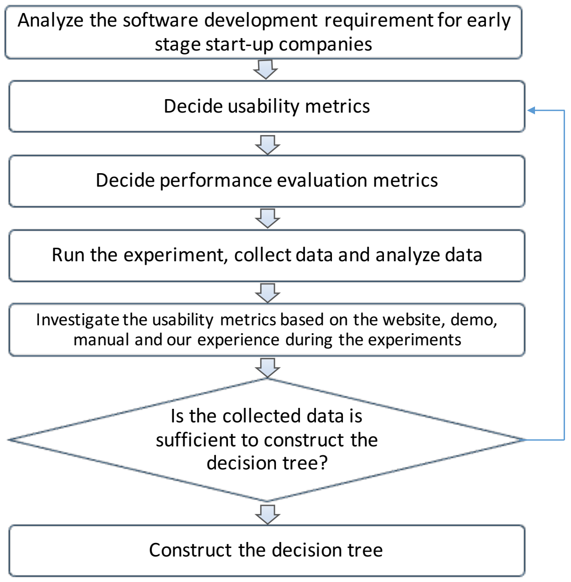

Analyze the software development requirement for early stage start-up companies
Decide usability metrics
Decide performance evaluation metrics
Run the experiment, collect data and analyze data
Investigate the usability metrics based on the website, demo, manual and our experience during the experiments
Is the collected data is sufficient to construct the decision tree?
Construct the decision tree

Fig. 2: Decision Tree Construction Procedure

V. Experiments

A. Configuration

We use YCSB to evaluate the performance running both Cassandra and MongoDB on OpenStack and Docker. In order to fairly evaluate the performance of both databases on both cloud platform, we choose a physical server to fix the variations of hardware device. During the duration of running benchmark, we guarantee that there is only one benchmark that runs on our system because disks are usually the bottleneck in the system, especially for disk measurement. We deploy our system on a single machine to design a decision tree for an early-stage startups. The hardware and software configurations of the experiments are shown as follow:

- Server: Dell Precision Tower 7810
- Processors: Intel Xeon Processor E5-2630 v3 (8C, 2.4GHz, Turbo, HT, 20M, 85W, with Intel-VT enable
- Memory: 32GB DDR4 at 2133MHz
- Hard Disk: 2TB 3.5” Serial-ATA (7,200 RPM)
- Operation System: Ubuntu Serever 14.04.0 64 bit
- Cloud platform: OpenStack Horizon
- VM OS: CentOS 7 64 bit
- Cloud platform: Docker version 1.5
- Container: CentOS 7 64 bit
- Key-Value Store: MongoDB, version 2.4.9
- Key-Value Store: Cassandra version 1.2.19
- Benchmark: YCSB version 0.1.4

B. Benchmark Workloads

We design six sets of experiments for the performance evaluation. Table II shows the details configurations of each experiment. The first four sets of experiments are named group one and the last two sets of experiments are named group two. In group one experiments, we explore the performance of each key-value store on each platform under different workloads provided by YCSB. The difference between the first four experiments is that the workloads for benchmarking. For the group 2 experiments, we explore flavors of a virtual machine that affects performance of each key-value on OpenStack. A flavor in OpenStack represents the capabilities of the virtual machine. For example, for VM with default large flavor, the VM is equipped with four virtual processors, 8 GB RAM and 80 GB disk. Since the overall hardware resource is limited, the number of virtual machine on the OpenStack will depend the rest of available resource. For each experiments on group 1, we specify six throughput values ranging from 5000, 10000, 15000, 20000, 25000 and 30000. For each experiment in group 2, we specify six throughput values ranging from 1000, 2000, 3000, 4000 and 5000 because of the limited hardware resource assigned to the VM. We

TABLE II: Experiment Configurations

Experiment	Workload	Database	Platform
1	A	Both	OpenStack & Docker Large flavor.
2	B	Both	OpenStack & Docker Large flavor.
3	C	Both	OpenStack & Docker Large flavor.
4	F	Both	OpenStack & Docker Large flavor.
5	A	MongoDB	OpenStack: Large, medium and small flavor.
6	A	Cassandra	OpenStack: Large, medium and small flavor.

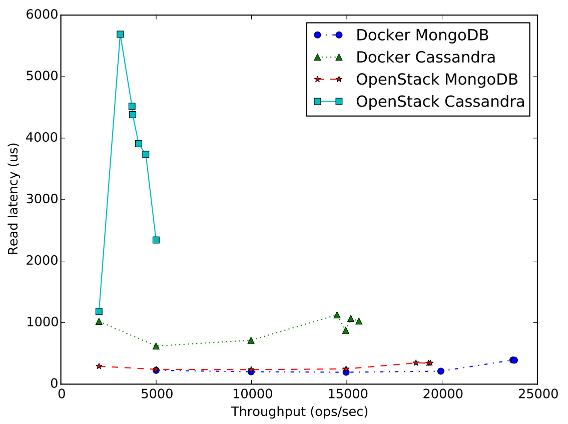
mark each data point of each line sequentially based on the given throughput. Finally, we use the overall throughput field from the individual log file to mark the actual throughput during the experiment. The operation count and record count are both set at 1,000,000 to have longer testing time.

C. Experiment 1

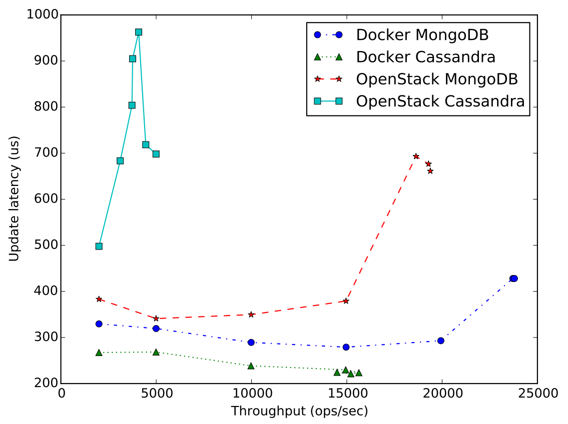
In the experiment 1, we examined Workload A with 50 percent reads and 50 percent updates. We generate latency versus throughput curves by trying different target throughput to measure the resulting latency for each cloud serving system. As the Figure 3 shows, MongoDB in a Docker container achieved best throughput and the lowest latency for reads. MongoDB on OpenStack has low latency for reads and it reaches the second highest throughput. The update operations show that MongoDB in a Docker container provides the highest throughput though the latency for updates is not the lowest one. Cassandra in a Docker container achieved the lowest latency for updates.

D. Experiment 2

We ran Workload B with 95 percent reads and 5 percent updates. As the results shown in Figure 4, MongoDB on OpenStack peaked at 29835 operations/sec with lowest latency for reads. On the other hand, MongoDB in a Docker container reached 23016 operations/sec. Both of them have stable latency for reads. For update operations, MongoDB



(a) Read operations



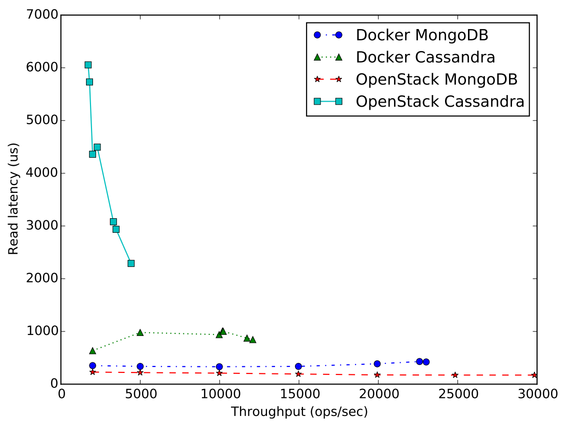
(b) Update operations

Fig. 3: Experiment 1 on Workload A–update heavy

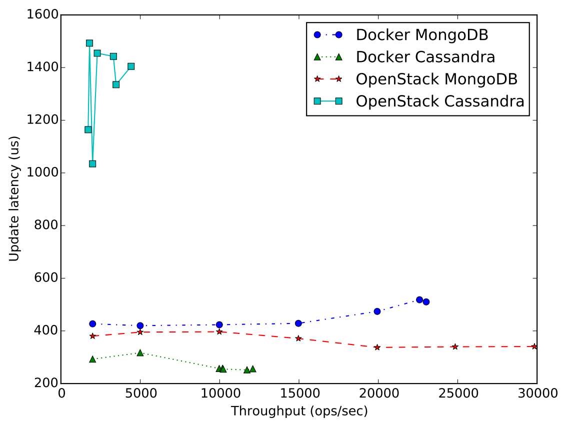
on OpenStack still reached best throughput and the latency decreased as offered throughput increased.

E. *Experiment 3*

Workload C is 100 percent read. Either MongoDB in a Docker container or on OpenStack outperformed Cassandra in Docker and OpenStack. As the Figure 5 shown, Cassandra's latency for reads with same configuration on both OpenStack and Docker had different results. Cassandra in a Docker container reached 12182 operations/sec but Cassandra on OpenStack performed 4424 operations/sec.



(a) Read operations

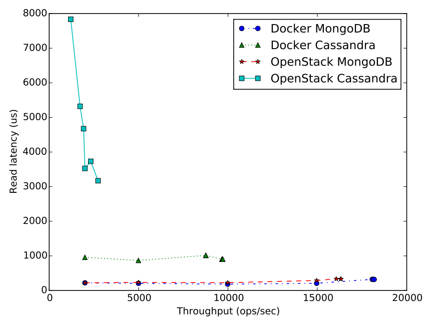


(b) Update operations

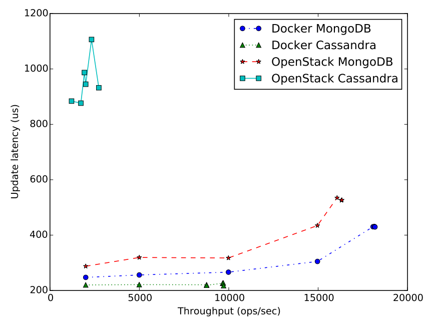
Fig. 4: Experiment 2 on Workload B–read heavy

F. *Experiment 4*

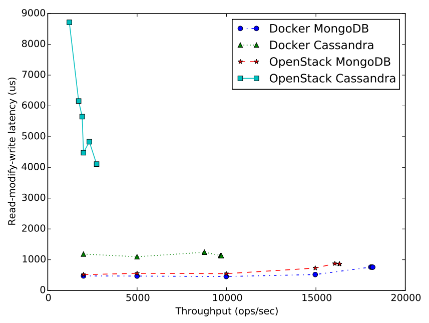
We ran workload F which is the read-modify-write workload with 50 percent reads, 50 percent read-modify-write. In this workload, the YCSB client reads a record, modify it, and write back to the database. The results are shown in Figure 6. As the offered throughput increased, the operation latency of MongoDB on both Docker and OpenStack increased. MongoDB on a Docker container achieved the best throughput for reads, updates, and writes and the lowest latency for reads and writes.



(a) Read operations



(b) Update operations



(c) Write operations

Fig. 6: Experiment 4 on Workload F–read-modify-write

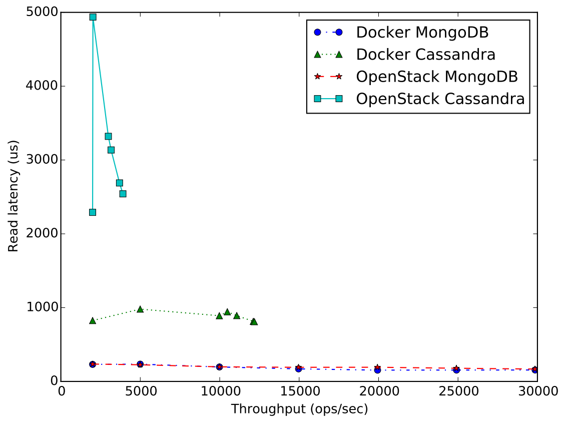


Fig. 5: Experiment 3 on Workload C–read only

G. *Experiment 5 and Experiment 6*

For group two experiments, Figure 7 and Figure 8 depict the different operation latency versus throughput on different flavor virtual machines on MongoDB and Cassandra respectively. In both group two experiments we use the default number of virtual processor and amount of RAM from the default flavors including large, medium and small. We change the disk amount of all flavors to 80 GB in order to have consistent and enough size to store the large record counts of the YCSB benchmark.

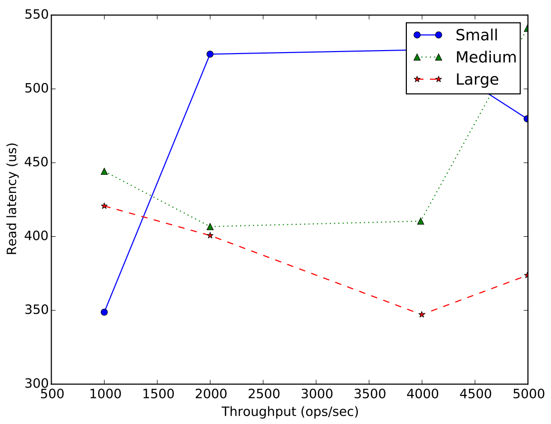
From both Figure 7 and Figure 8, we can see flavors affect the read and update latency dramatically. For VM with small flavor, given the throughput of 5000, YCSB can only run at less than 3000 operations per seconds. The VM with better hardware resources can significant lower the Read and Write latency. This is true on both Experiment 5 and Experiment 6. Usually, flavors or also called size of virtual machine affect the operation cost. Public cloud providers such as Amazon EC2 or Microsoft Azure provide higher price for a better equipped virtual machine.

H. *summary*

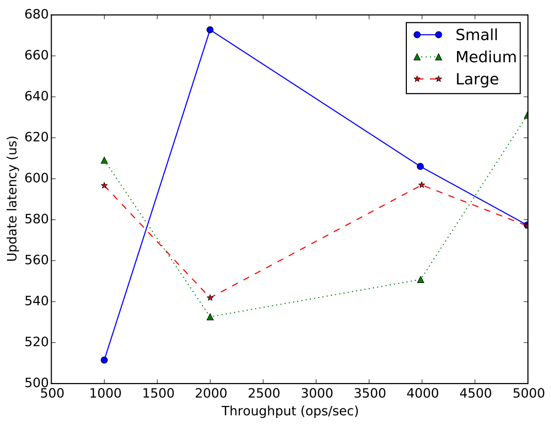
Figure 3, Figure 4, Figure 5 and Figure 6 show the results of experiment 1 to 4 respectively. For MongoDB on group one experiment, MongoDB performs better on the Dokcer than the OpenStack. All the VM used in group one are large flavor. These experiment results would be affect by the insufficient hardware resources or the performance degradation of the virtual machine. We observe the similar results of Cassandra on the group 1 experiments. Moreover, for Cassandra on OpenStack platform, YCSB cannot achieve the assigned throughout on most of the data points. To address this issue, we decrease the assigned throughput range on the group two experiments.

VI. Decision Tree Development

The proposed decision tree enables developers to choose the suitable cloud technology. We follow the criteria of performance and usability to construct the decision tree. For example, on the performance criteria, we use key-value store benchmark YCSB on

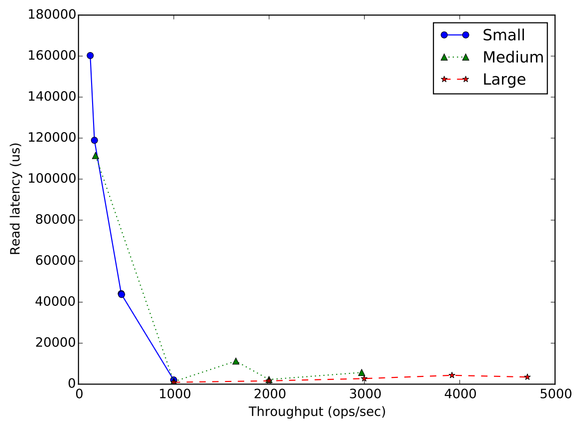


(a) Read

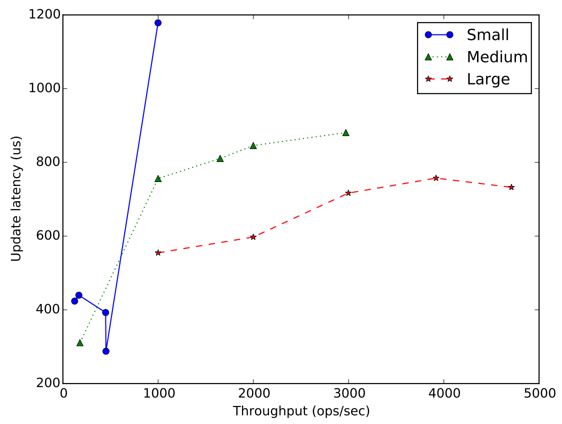


(b) Update

Fig. 7: Experiment 5: Flavors of VM on MongoDB



(a) Read



(b) Update

Fig. 8: Experiment 6: Flavors of VM on Cassandra

both platforms. In addition, we will analyze what are the demands of start-up companies for their cloud services. Based on this demand, we analyze these metrics of each criteria and derive the decision tree. We present the usability analysis and decision tree in the following sections.

A. Usability Analysis

We analyze on the the quality of documentation and the complexity of operability of Docker and OpenStack. We examine the online documentations [28] [29] and official online forum [30]- [31] of both Docker and OpenStack for investigation. According to the [25], we choose the metrics listed in Table III. Each metrics is scored that ranges from one to three. One represents the positive to arguments of the metric. Table III presents the scores of both Docker and OpenStack on each metric of our investigation.

We give Docker higher rank than OpenStack. For the two metrics on manual, although Docker is relatively young and simpler in terms of product and design in comparison to OpenStack, the documentations of Docker are readable and effective. Considering the design complexity and development age, OpenStack has completely official documentations than Docker. However, it is harder for the first time reader to decide what they need on OpenStack documentation than Docker. On the contents of demo,

TABLE III: Qualities of documentation

Metrics	Docker	OpenStack
Contents of manual	3	2
Effectiveness of manual	3	2
Contents of demo	3	1
Contents of help system	3	3
Effectiveness of help system	3	3

we rank Docker higher than OpenStack. Users can simply follow the guides on Docker documentation and have a workable Docker system. However, it is difficult for developers to set up a workable multiple or single node OpenStack system. It is the reason why we rank OpenStack lower than Docker in terms of contents of demo. Docker provides a series of intuitive and understandable tutorials for developers to get hands on quickly and painless. According to our experience, documentation on Docker is more self-contained and does not require user to spend extra efforts on finding the relevant details.

For operability, we choose the ease of setup, requirements of hardware resource and maintainability. We ranks Docker higher than OpenStack in terms of the installation process. It is easy to get Docker run on a server or simply a laptop on Linux system. It is difficult to install OpenStack even for the single node installation. At first time, we had issues on installing OpenStack on Google cloud platform. We assume that it is not practical to setup another cloud environment on the Google cloud platform. Then, we switch to deploy on a physical server. It is difficult to install OpenStack based on the official manual. During the installation, we also found an installation bug and then reported to the official.

OpenStack is different from Docker on the level of virtualization. OpenStack uses hardware level virtualization in comparison to the process level virtualization for Docker. For the hardware resource part, OpenStack relies on decent hardware for installation. For single installation from the Ubuntu Cloud Installer, it recommends the minimum installation should have 8 CPUs, 12 GB memory (hard constraints) and 100GB capacity of disk. There is no official minimum requirement on hardware for Docker. In the experiment, we can deploy Docker on a virtual machine with 512MB memory. On the maintainability, both platforms provide command line tools. OpenStack provides official web based GUI dashboard for managements. It would be easier for most of the management tasks. As a result, we rank OpenStack higher on maintainability.

TABLE IV: Operability

Metrics	Docker	OpenStack
Ease of setup	3	1
Level of hardware	1	3
Ease of Maintainability	2	3

B. Decision Tree Construction

We have evaluated the usability based on qualities of documentation and operability. Besides, we use YCSB to benchmark the key-value store: MongoDB and Cassandra on both Docker and MongoDB. We use the above mentioned analysis and experiments to design the seven questions for construction the decision tree. Based on answering these questions sequentially, developers have a clear picture to make a decision between Docker or OpenStack.

As Figure 9 shown, the decision tree is to choose the cloud platform between Docker or OpenStack. We provide a skew tree because there are two platforms to be chosen: Docker or OpenStack. The question nodes are arranged by importance of the decision. If the question is more important, then the user gets the decision earlier. For example, for the first question, users are asked whether the application runs on Windows platform. If the answer is yes, then the user chooses OpenStack. The reason is that Docker is based on the LXC technologies on Linux kernel. As a result, there is not easily to run Docker on Windows operating system. Though the Docker designers provide a helper application called Boot2Docker which creates a Linux virtual machine on Windows to run Docker, we got some bugs during the experiments.

The answer of each question does not inherit the decision characteristics from previous questions. For example, the left most and right most leaf are

OpenStack. For the left most leaf on OpenSatck, it does not contradict to the right most leaf on OpenStack. For example, the left most leaf does not necessary need to take the left branch of the first question.

The second question is about hardware resource. Based on the Table IV, Docker almost has no hardware limitation. The third question is about the performance, Docker has better performance due to its design of light weight virtualization. For the fourth question, Docker is easier to maintain. For the sixth question, OpenStack is chosen since it has low level virtualization technology. The final decision is the uniqueness of Docker that the Docker container can be shipped and runs where it is on another Docker. Although OpenStack has functions to take the snapshot of the VM and shares the snapshot with others to replicate the service and environment, it is more heavy weight than the Docker container. Hence, we choose Docker when developers require build-and-ship frequently.

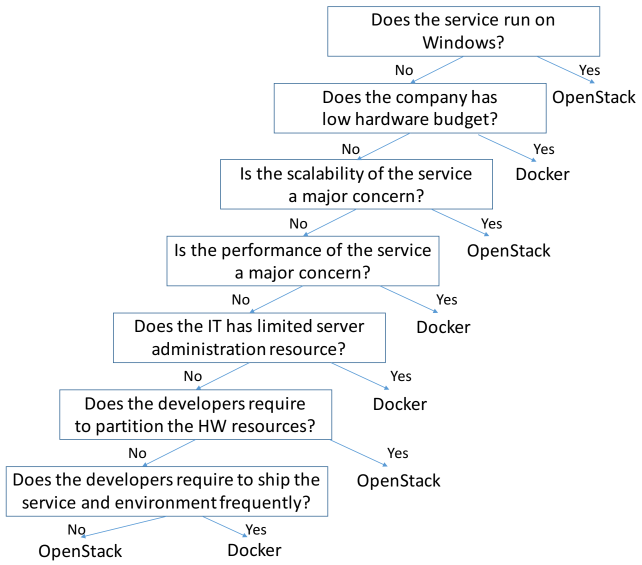

Does the service run on Windows?
No
Yes
OpenStack
Does the company has low hardware budget?
No
Yes
Docker
Is the scalability of the service a major concern?
No
Yes
OpenStack
Is the performance of the service a major concern?
No
Yes
Docker
Does the IT has limited server administration resource?
No
Yes
Docker
Does the developers require to partition the HW resources?
No
Yes
OpenStack
Does the developers require to ship the service and environment frequently?
No
Yes
OpenStack
Docker

Fig. 9: Decision Tree of Docker versus OpenStack

VII. Discussion

We design the decision tree based on the analysis and experiments. In this section, We discuss a few issues during the analysis and experiments and potential improvement.

Docker and OpenStack use different level of virtualization which process level virtualization and hardware level virtulization respectively. In this paper, we do not compare two technologies on different levels. That is, we do not evaluate OpenStack versus VMWare or QEMU versus Xen Hypervisor. The reason we compare Docker versus OpenStack is that both platforms are emerging and have potential to dominate on each domain. Therefore, it is intuitive and interesting to compare both technologies.

We evaluate the performance on Docker and OpenStack from the aspect of key-value store performance. We choose YCSB benchmark for MongoDB and Cassandra. MongoDB outperforms Cassandra on the chosen four workloads based on the experiments results. There are a variety of benchmarks such as Google PerfKit; however, we cannot investigate all the potential benchmarks due to the limited time. As a result, the experiments are sufficient to understand the performance difference between Docker and OpenStack with the database aspects. Future works include using other benchmark from different aspects and run all workloads provided in YCSB. In addition, we can fully discover the potential on OpenStack and Docker on a fully distributed environment. We can benchmark Docker containers under different resource allocations as well.

For the usability analysis, we analyze each platform based on the official website, user forum and our user experiences. There are software engineering metrics which can be used for the potential improvements. One difficulty during the experiment is that having an objective score for each metrics. For example, it is infeasible for us to count the number of unanswered questions on the user forum. However, this might be an index for evaluating how active the forum. If the forum does not provide sufficient statistics, it is hard to evaluate the system.

VIII. Conclusion

Start-up companies are important thrusts of innovation. To become a successful start-up company, entrepreneurs and developers need to make tons of important decisions. Among all the important decisions, choosing the deployed platform is one of the hardest and crucial decision when develop the service. We present a decision tree for developers to choose the cloud platform based on the experiments to benchmark key-value store performance and analysis on the usability. Based on the evaluation and the analysis, we design the decision tree with seven questions to assist developers to make a right decision to meet their needs.

Acknowledgment

We appreciate Professor Indranil Gupta for discussion the theme and scope of the paper, Muntasir Raihan Rahman for the discussion on virtualization backup mechanisms and Faraz Faghri for discussion on levels of virtualization and comparisons between each virtualization technology. We thank all of the reviewers and students at CS 525 Advanced Distributed System, and Professor Indranil Gupta for their time and constructive suggestions.

References

[1] M. J. Scheepers, "Virtualization and containerization of application infrastructure: A comparison," 2014.

[2] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support paas," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 610–614.

[3] K.-T. Seo, H.-S. Hwang, I.-Y. Moon, O.-Y. Kwon, and B.-J. Kim, "Performance comparison analysis of linux container and virtual machine for building cloud," 2014.

[4] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," *technology*, vol. 28, p. 32, 2014.

[5] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*. IEEE, 2013, pp. 233–240.

[6] D. Strauss, "The future cloud is container, not virtual machines," *Linux Journal*, vol. 2013, no. 228, p. 5, 2013.

[7] Kvm and docker lxc benchmarking with openstack. [Online]. Available: http://bodenr.blogspot.com/2014/05/kvm-and-docker-lxc-benchmarking-with.html

[8] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, no. 3, pp. 81–84, 2014.

[9] Do i need openstack if i use docker? [Online]. Available: https://www.mirantis.com/blog/need-openstack-use-docker/

[10] Docker launches its container orchestration tools. [Online]. Available: http://techcrunch.com/2015/02/26/docker-launches-its-container-orchestration-tools/

[11] Docker swarm. [Online]. Available: https://docs.docker.com/swarm/

[12] R. Nasim and A. J. Kassler, "Deploying openstack: Virtual infrastructure or dedicated hardware," in *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*. IEEE, 2014, pp. 84–89.

[13] Openstack. [Online]. Available: http://www.openstack.org/

[14] R. Kumar, N. Gupta, S. Charu, K. Jain, and S. K. Jangir, "Open source solution for cloud computing platform using openstack," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 5, pp. 89–98, 2014.

[15] The new stack and linux foundation survey: Openstack and docker are the most popular open source projects. [Online]. Available: http://thenewstack.io/the-new-stack-and-linux-foundation-survey-openstack-and-docker\penalty-\@M-are-the-most-popular-open-source-projects/

[16] A. Barkat, A. D. d. Santos, and T. T. N. Ho, "Open stack and cloud stack: Open source solutions for building public and private clouds," in *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2014 16th International Symposium on*. IEEE, 2014, pp. 429–436.

[17] Q. Xu and J. Yuan, "A study on service performance evaluation of openstack," in *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2014 Ninth International Conference on*. IEEE, 2014, pp. 590–593.

[18] D. Steinmetz, B. W. Perrault, R. Nordeen, J. Wilson, and X. Wang, "Cloud computing performance benchmarking and virtual machine launch time," in *Proceedings of the 13th annual conference on Information technology education*. ACM, 2012, pp. 89–90.

[19] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010, pp. 143–154.

[20] M. Roussopoulos, M. Baker, D. S. Rosenthal, T. J. Giuli, P. Maniatis, and J. Mogul, "2 p2p or not 2 p2p?" in *Peer-to-Peer Systems III*. Springer, 2005, pp. 33–43.

[21] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[22] The mongodb 2.4 manual. [Online]. Available: http://docs.mongodb.org/v2.4/

[23] Cassandra architecture in brief. [Online]. Available: http://docs.datastax.com/en/cassandra/1.2/cassandra/architecture/architectureIntro_c.html

[24] E. Hewitt, *Cassandra: the definitive guide*. " O'Reilly Media, Inc.", 2010.

[25] M. Bertoa and A. Vallecillo, "Usability metrics for software components," in *8th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE2004), Oslo, Norway*, 2004.

[26] R. Sharda, S. H. Barr, and J. C. MCDonnell, "Decision support system effectiveness: a review and an empirical test," *Management science*, vol. 34, no. 2, pp. 139–159, 1988.

[27] R. J. Kuo, C. Chen, and Y. Hwang, "An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network," *Fuzzy sets and systems*, vol. 118, no. 1, pp. 21–45, 2001.

[28] Docker documentations. [Online]. Available: https://docs.docker.com/

[29] Openstack documentations. [Online]. Available: docs.openstack.org/

[30] Docker forum. [Online]. Available: https://forums.docker.com/

[31] Openstack forum. [Online]. Available: https://ask.openstack.org/zh/questions/