Performance Comparison for Mining Large Data from the Internet and Learning using ID3 Algorithm in a Docker versus Virtual Machine Environment

Abishek Ravichandran
B.E Computer Science and Engineering.
S.S.N College of Engineering
Kalavakam, Chennai, Tamil Nadu, India

Aishwarya Sundararajan
B.E Computer Science and Engineering.
S.S.N College of Engineering
Kalavakkam, Chennai, Tamil Nadu, India

V. Balasubramanian
Assistant Professor ME. (CSE)
S.S.N College of Engineering
Kalavakam, Chennai, Tamil Nadu, India.

ABSTRACT
Every day, 2.5 quintillion bytes of data are generated. A sizeable portion of the data is available through the internet. The efficacy of the decisions being made revolves around the extent to which analysis is performed on the procured data. Containers provide Operating System Virtualization and Linux Containers present secure execution environments by independently executing processes.[1]This paper aims at proving that the performance of Docker Container in mining large data from the internet and learning using ID3 algorithm to generate a decision tree to predict useful results is much better than the performance in a Virtual Machine Environment.

General Terms
Virtualization

Keywords
Docker, Container, Virtual Machine.

1. INTRODUCTION

1.1 What is a Virtual Machine?
A Virtual Machine is an operating System or an application environment that is installed as software and imitates dedicated hardware. Software called Hypervisor emulates multiple virtual hardware platforms that are isolated from each other (CPU, Memory hard disk, network and other hardware resources). A Virtual Machine that is running on a physical host can consume unequal resource quantities. Virtualization requires more bandwidth, storage and processing capacity than a traditional desktop or a server.[2]

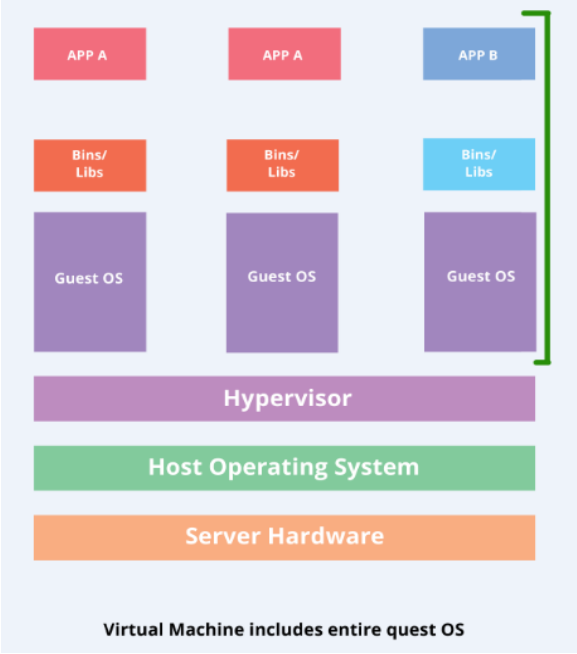
1.2 What is a Container?
Containers have a long history in computing. Unlike Virtualization through the use of hypervisors, in which several independent virtual machines run virtually on the physical hardware through an intermediate layer, containers run user spaces on top of the system's operating system.[3] Hence, container virtualization is often dubbed operating-system level virtualization. It is thus required that the container must be developed on the same kernel as the operating system of the host. Containers allow multiple isolated user spaces to be run in a single host.

Containers are popular for large-scale deployments of multi-client services for light-weight sandboxing and as process isolation environments. chroot jail is a common example of a container that creates an isolated directory environment for running processes.[4] Hackers that illegally find their way inside the environment, often find themselves trapped in the container, posing no threat to the actual host itself. Containers do not require an emulation layer or a hypervisor layer to run and instead use the operating system's normal system call interface. This reduces the overhead required to run containers. Containers can be complex, extremely hard to setup and difficult to manage.

1.3 Advantages of Container over Virtual Machine
Containers run on the host Operating System, instead of running a separate guest Operating System on top of the host OS. It can hence be concluded that Hypervisors are not used in Container Virtualization.

Both Virtual Machines and Containers achieve the same objective – to achieve platform independence. But the primary advantage of Containers over Virtual Machines is that the implementation of the former does not involve the usage of two Operating Systems. Containers are thus faster than Virtual Machines.

The architectural comparison between a Virtual Machine and a Container is shown in Figure 1.



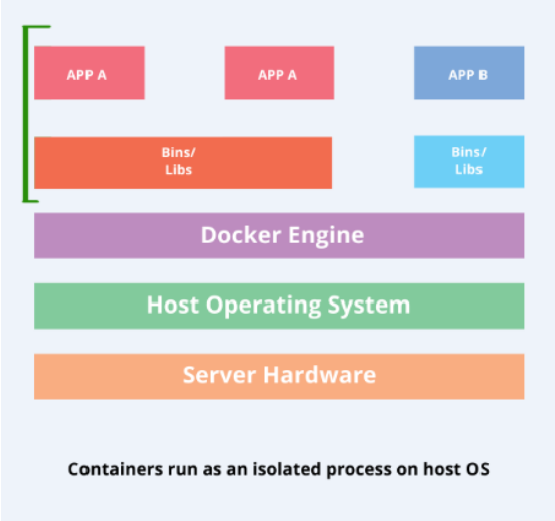


Fig 1 Virtual Machines versus Containers

1.4 What is a Docker Container?

A Docker is a tool that aims to remove all the drawbacks of a Container, as a Container can be complex and hard to manage. A Docker has 4 components, as shown in Figure 2, namely:

1.4.1 The Docker client and server

Docker has a client and server. The Docker client communicates with the Docker server or daemon, which, in turn, is made to do the necessary work. Docker is equipped with a command line interface, and also a full RESTful API. The Docker daemon and client can be run on the same host or can be connected to your local Docker client to a remote daemon running on a different host. [4]

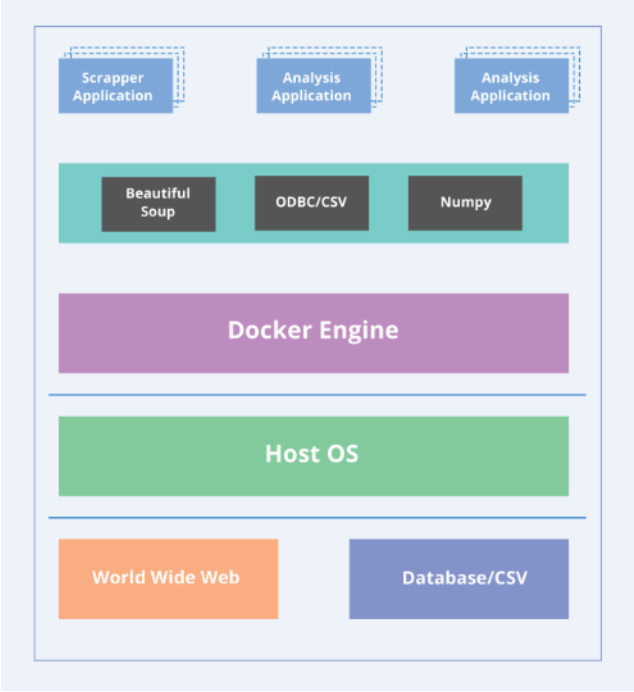


Fig 2 Docker Architecture

1.4.2 Docker Images

Images form the building blocks or foundation of the Docker universe. Containers are launched from images. Images are basically compose the build phase of the Docker lifecycle. They have a layered format, using Union file systems, that are built step-by-step using a set of instructions executed in series.

Images can be considered as the source code for your containers. They have high portability and can be shared, stored, and updated.[4]

1.4.3 Registries

The built images are housed in the registries. The two key types being: public and private .Docker operates the publish registry for images, called the Docker Hub. An account can be created on the Docker Hub to share and store images. A user can also run his own private registry. This enables users to store images behind their respective firewalls, which may be a primary requirement for certain organizations.[4]

1.4.4 Docker Containers

Docker helps to build and deploy containers within which you can package the required applications and services. Containers are launched from images. They may contain one or more running processes. [4]

2. ID3 ALGORITHM

ID3 (Examples, Target_Attribute, Attributes)

Step 1 Create a root node for the tree

Step 2 if all examples are positive,

2.1 Return the single-node tree Root, with label = +.

Step 3 if all examples are negative,

3.1 Return the single-node tree Root, with label = -.

Step 4 if number of predicting attributes is empty,

4.1 Return single node tree Root with label = most common value of the target attribute in the examples.

Step 5 Otherwise Begin

5.1 A = The Attribute that best classifies examples.

5.2 Decision Tree attribute for Root = A.

5.3 For each possible value, v_i, of A,

5.3.1 Add a new tree branch below Root, corresponding to the test A = v_i.

5.3.2 Let Examples (v_i) be the subset of examples that have the value v_i for A.

5.3.2.1If Examples (v_i) is empty,

5.3.2.1.1Below the new branch add a leafnode of label = most common target value in the examples.

5.3.2.2Else below this new branch add the subtree ID3 (Examples (v_i), Target_Attribute, Attributes – {A})

Step 6 End

Step 7 Return Root

Fig 3 ID3 Algorithm

The ID3 (Iterative Dichotomiser 3) algorithm, as shown in Figure 3 [5], is a non-incremental algorithm that is used to construct a decision tree from a dataset. The ID3 classes are inductive. This means that all classes created by the algorithm for a set of training samples are expected to work for all future instances.

2.1 Entropy

A statistical property - Information Gain - is used in the ID3 algorithm. Information Gain measures how well the attribute separates the data set into targeted classes. Information Gain is derived from another property from Shannon's Information Theory - Entropy. Entropy is generically defined as a measure of disturbance. In the Information Gain Theory, Entropy measures the amount of information that rests in an attribute in a given data set. [5]

Entropy H(S) is defined as follows: [6]

$$H(S) = -\sum_{x \in X} p(x) \log p(x) \cdots\cdots\cdots (1)$$

S - The current (data) set for which entropy is being calculated (changes every iteration of the ID3 algorithm)

X - Set of classes in S

p (x) - The proportion of the number of elements in class x to the number of elements in set S.

2.2 Information Gain

The Information Gain is calculated as follows: [6]

$$IG(A,S) = H(S) - \sum_{t \in T} p(t) H(t) \cdots\cdots\cdots (2)$$

H(S)-Entropy of set S.

T-The subsets created from splitting set S by attribute A such that, $S = \bigcup_{t \in T} t$

p(t)-The proportion of the number of elements in t to the number of elements in set S.

H(t) – Entropy of subset t.

2.3 Data Description

The sample data used by ID3 has certain requirements [7], which are:

2.3.1 Attribute-value description

The attributes describing each example should be the same and must have a fixed number of values.

2.3.2 Predefined classes

The attributes of an example must already be defined, that is, they are not learned by ID3.

2.3.3 Discrete classes

Classes must be sharply delineated.

2.3.4 Continuous classes

Classes broken up into vague categories such as a metal being "hard", "flexible" or "soft" are suspect.

2.3.5 Sufficient examples

There must be enough test cases to distinguish valid patterns from chance occurrences as inductive generalization is used. (i.e not provable).

2.4 Attribute Selection

A statistical property, called information gain, is used to decide which attribute is the best. Gain measures how well a given attribute separates training examples into targeted classes. The one with the highest information is selected. Entropy measures the amount of information in an attribute.

Given a collection S of c outcomes,

$$Entropy(S) = S - p(I) \log p(I) \cdots\cdots\cdots (3)$$

P (I) is the proportion of S belonging to class I. S is over c.Note that S is not an attribute but the entire sample set. [7]

3. HDFS AND HADOOP

Hadoop Distributed File System works by fragmenting large files into smaller pieces called blocks which are stored on data nodes. The entire collection of all the files in the cluster is sometimes also known as the file system namespace. The management of this namespace is performed by the NameNode.

The data nodes can communicate among themselves so as to cooperate during normal file system operations. This is essential because blocks for one file are more likely to be stored on multiple data nodes. Because the NameNode is so crucial for accurate operation of the cluster, it can and should be duplicated as a precaution in case of single point failure.

HDFS addresses the challenges of big data by splitting files into a related collection of smaller blocks. These blocks are then distributed among the data nodes in the HDFS cluster. These are managed by the NameNode. Block sizes are customizable and are usually 128 megabytes (MB) or 256MB, which implies that a 1GigaByte file takes up eight 128MB blocks for its storage needs.

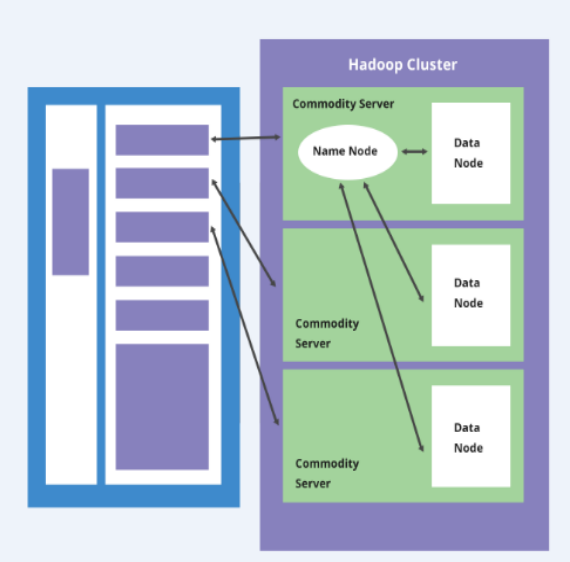


Fig 4 Hadoop Cluster

HDFS is quite resilient. Hence these blocks are replicated throughout the cluster to guard against a server failure. HDFS keeps track of all these fragments or pieces through file system metadata. Metadata are defined as data about data.

A Hadoop cluster has been shown in Figure 4 . The figure has three commodity servers and a NameNode with a set of DataNodes in each server.

Imagine HDFS metadata to be a template for providing a comprehensive description of the following points:

1) When the file was created, modified, deleted and accessed. Within the cluster, where the file blocks are stored.

2) Who holds the rights to modify or view the file .

3) How many different files are stored on the cluster.

4) The number of data nodes that exist within the cluster.

5) The location where the transaction log is present for the cluster.

HDFS metadata is quite often stored in the NameNode. The metadata is loaded into the physical memory of the NameNode server while the cluster is in operation. For a larger cluster, a larger metadata footprint can be expected.

The Hadoop MapReduce process includes several stages, each of which involves important operations, which helps you to reach your goal of getting the required answers from the Big Data set. The process is initiated with a request from the user to run a MapReduce program and this continues till the results are written back to the Hadoop Distributed File System (HDFS).[8]

4. IMPLEMENTATION

The main aim of this paper is to show the advantages of container virtualization over Hypervisor virtualization (using Virtual Machines) in terms of probability as well as performance. This is achieved by running a Python program that uses Hadoop clustering, MapReduce and ID3 algorithm on a sample data set for both the aforementioned setups. The performance metric in each case is the time taken for the developed sequence of programs to execute in each environment.

Five datasets containing 10,000 to 50,000 records, each having 10,000 records more than the previous set, is utilized in our test environment. These datasets are a set of grades obtained in all courses for each student in a particular semester. The grade ranges from 'A' to 'E'. The prediction (column) is the Grade Point Average (GPA). The number of training examples is unknown.

The functionality of the program is to obtain a dataset from a web page, parse the data obtained using HDFS and Hadoop. Then, the parsed values of data, obtained from the previous step, are trained using ID3 algorithm, which provides the predicted value of GPA as a result.

Firstly, the data is obtained through the usage of the API – beautifulsoup in python. The web page is parsed and converted into a csv file to reduce the complexity in accessing the dataset.

The second step is carried out using HDFS and Hadoop MapReduce. The main purpose as to why HDFS and Hadoop is incorporated in this step, is to implement an optimal way to arrive at the probabilities of each grade per subject depending on the total number of training samples by making use of Hadoop Clustering and MapReduce. The data in each column is placed in separate directories in the HDFS. The Hadoop algorithms are run for the data in each of those input directories present in the HDFS (in this case, for each subject) successively and the count of each grade is obtained for each column/subject by parsing the contents of the output files generated in HDFS once MapReduce is completed at every iteration. The total number of training examples can be obtained by summing up the counts of all grades for any one subject. Using this data, the probability of grade in each subject is estimated.

Finally, the data obtained from MapReduce and the csv dataset is fed to the ID3 algorithm for training. Once the training is complete and the Decision Tree generated, the algorithm would be able to predict the resultant Grade Point Average of a student given a sample set of grades in each subject.

For performance measurement in both container and virtual machine environment, the total time consumed for mining the HTML file from the Internet followed by performing Hadoop MapReduce in the HDFS as well as the ID3 data training time are calculated and noted.

The implementation is run on both Linux Container as well as a Virtual Machine and the performance cost is obtained and compared for both the cases. The results obtained from our analysis depict that the performance is faster in a Linux Container environment than in a Virtual Machine environment.

5. PERFORMANCE COMPARISON

The performance metrics used to analyze the comparison between the container environment and the virtual machine environment are the total time consumed for mining the HTML file from the Internet followed by performing Hadoop MapReduce in the HDFS, training time of the ID3 algorithm.

Five large data sets comprising thousands of records are run in both the environments and the comparison parameters are computed for each case and tabulated. The following points can be inferred from Table 1:

Table 1. Performance Comparison

Dataset Size	Container Environment		VM Environment	
	Execution Time (in seconds)	Training Time (in seconds)	Execution Time (in seconds)	Training Time (in seconds)
10000	150	0.366416	247	0.511873
20000	155	0.819863	269	1.031174
30000	165	1.220207	291	1.644953
40000	175	1.752751	579	2.117370
50000	184	2.275225	1084	2.630092

1) The total time consumed for mining the HTML file from the Internet followed by performing Hadoop MapReduce in the HDFS by the Virtual Machine (known as the Execution Time in Seconds(s)) is greater than the time consumed by the containers and there is a drastic increase in the execution time consumed by the Virtual Machine as the dataset grows larger.

2) The training time of the ID3 (measured in Seconds(s)) algorithm is greater in a Virtual Machine than in a Container for a given dataset**.**

Graphical plots of each performance metric versus the dataset size are plotted below for both the containers and the Virtual Machine.

From Figure 5a), it is inferred that as the dataset size increases, there is a drastic increase in the Execution time for the Virtual Machine whereas there is only a small linear increase in the Execution Time for the Container.

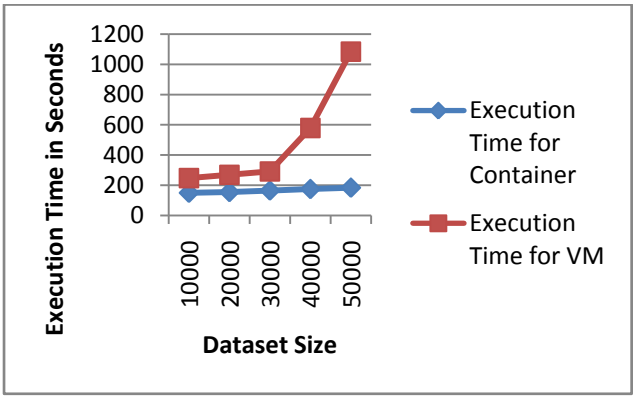


Fig 5a) Dataset size versus Execution Time

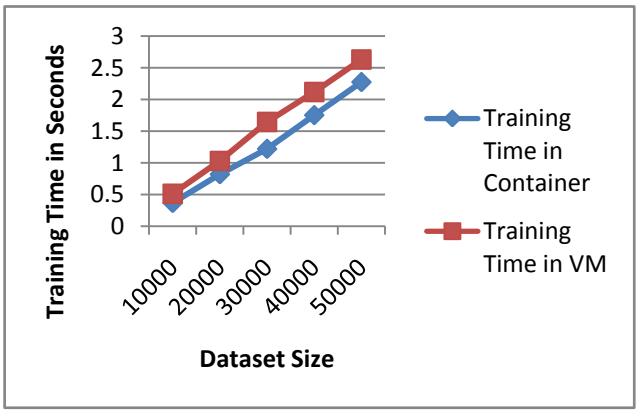


Fig 5b) Dataset size versus Training Time

From Figure 5b), it is inferred that the training time for the ID3 Algorithm is higher in a Virtual Machine environment than in the Container environment.

6. CONCLUSION

Containers have secured their place in the technology world owing to their ability to scale up or down and security isolation. In addition, there is no hassle of managing physical servers. Mining from the World Wide Web and analysis in a Desktop Operating System takes lesser time than that in the Container Environment. But Containers are portable unlike Desktop Operating Systems.

This paper contains an effective implementation of Container-based Virtualization and shows that the performance of a Container is much better than that of a Virtual Machine for scraping large data from the internet and performing analysis on the data scraped using the ID3 algorithm. It can also be concluded that the training time of the ID3 algorithm is lesser in the Container environment as against that in the Virtual Machine environment.

Containers have a lot of scope for future work and improvement, which is discussed under the next heading.

7. SCOPE FOR FUTURE WORK

Containers are getting popular by the day. Thanks to this emerging technology, Container-based solution exists for Virtualization. The application is encapsulated along with all the dependencies and packages into a system and language-independent module known as the 'container'. It is within the container that the application runs in isolation from the other containers. However, containers share the resources of the host machine. The paper takes advantage of the isolated process execution within containers.

Docker is a lightweight virtualization solution to implement containers. It is, in fact, monopolizing the container industry with over 78 percent market share. Despite not offering the security that full hypervisor-based virtualization offers, Docker is widely used because of its performance and flexibility. Stalwarts in the industry agree to the following as the major advantages of containers – faster/easier deployment, flexibility in deployment, process/memory isolation, scalability and cost savings compared to Virtual Machines. Docker is about 600 times faster than Virtual Machines during startup and about 100 times faster during shutdown. In absolute reference, Docker takes only about 50ms for both startup and shutdown.

The paper can be further improved with a more intuitive user interface and a provision for remote migration. The application can also be made to automatically fill up forms to fetch data. The image of the container can be uploaded to an online repository so that the container can be pulled from any machine. [9]

Containers are a great path to better applications, both in the cloud and on-premises. Containers make the lives of developers and testers easy. There will be a day when platforms are no longer a hindrance to effective application development and deployment owing to containers.

8. REFERENCES

[1] Major Hayden, "Securing Linux Containers", GIAC (GCUX) Gold Certification, July 26, 2015.

[2] Samuel T. King, George W. Dunlap, Peter M. Chen, "Operating System Support for Virtual Machines," Proceedings of the 2003 USENIX Technical Conference, 2003.

[3] Young, C.J., "Extended Architecture and Hypervisor Performance," Proceedings IEEE Computer Society Conference, Boston, MA, September 1971.

[4] James Turnbull,2014,'The Docker Book'

[5] http://faculty.simpson.edu/lydia.sinapova/www/cmsc310/LN310_AIMA/L13-ML-ID3.pdf

[6] ID3Algorithm,https://en.wikipedia.org/wiki/ID3_algorithm

[7] https://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm

[8] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Google,Inc.,OSDI 2004.

[9] Carl Boettiger, An introduction to Docker for reproducible research, with examples from the R environment, (2015) ACM SIGOPS Operating Systems Review, Special Issue on Repeatability and Sharing of Experimental Artifacts. 49(1), pp.71-79.