CRAWL BOT MINING AND ANALYSIS OF DATA FROM THE WORLD WIDE WEB THROUGH CONTAINERS

A PROJECT REPORT

Submitted By

VIGNESH N 312212104118

ABISHEK R 312212104005

MUKUNDRAM M 312212104060

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SSN COLLEGE OF ENGINEERING

KALAVAKKAM 603110

ANNA UNIVERSITY :: CHENNAI - 600025

April 2016

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report titled **"CRAWL BOT Mining and Analysis of Data from the World Wide Web through Containers"** is the *bonafide* work of "**VIGNESH N. (312212104118)**, **ABISHEK R. (312212104005)**, and **MUKUNDRAM M. (312212104060)**" who carried out the project work under my supervision.

DR. CHITRA BABU
HEAD OF THE DEPARTMENT
Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

V.BALASUBRAMANIAN
SUPERVISOR
Assistant Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Place:
Date:

Submitted for the examination held on. . . . . .

Internal Examiner **External Examiner**

ACKNOWLEDGEMENTS

I thank GOD, the almighty for giving me strength and knowledge to do this project.

I would like to thank and deep sense of gratitude to my guide **V.BALASUBRAMANIAN**, Assistant Professor, Department of Computer Science and Engineering, for his valuable advice and suggestions as well as his continued guidance, patience and support that helped me to shape and refine my work.

My sincere thanks to **Dr. CHITRA BABU**, Professor and Head of the Department of Computer Science and Engineering, for her words of advice and encouragement and I would like to thank our project Coordinator **Dr. S. SHEERAZUDDIN**, Associate Professor, Department of Computer Science and Engineering for his valuable suggestions throughout this first phase of project.

I would like to extend my sincere thanks to all the teaching and non-teaching staffs of our department who have contributed directly and indirectly during the course of my project work.

Finally, I would like to thank my parents and friends for their patience, cooperation and moral support throughout my life.

Vignesh N. **Mukundram M.** **Abishek R.**

ABSTRACT

Every day, we generate 2.5 quintillion bytes of data. A sizeable portion of the data we generate is available through the World Wide Web. The efficacy of the decisions that we make revolves around the extent to which analysis is performed on the data procured. This project aims at improving the decisions that people make by providing the entropy associated with the attributes of automatically fetched data.

Information Security is the greatest concern pertaining to the cyber world. The data collected from the World Wide Web must not only be mined securely, but also be analyzed and stored keeping in mind the growing security concerns of the target audience. Linux Containers present secure execution environments by independently executing processes. Containers also break the platform barrier by providing Operating System Virtualization. This project also aims at providing the secure environment for securely mining and analyzing data and presenting the results to the user.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

”I never guess. It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.”, said Sir Arthur Conan Doyle once. This quote is the reflection of what exists today in the World Wide Web. Data today is synonymous to renewable energy. There is no dearth of data, but tapping of data is minimal. There is thus a need to tap the data. In this era of data deluge, it is impossible to manually perform mining and analysis on the data. The aim of this project is to automate the process of mining and analysis while ensuring the security required through container virtualization.

1.1 Containers

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. 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.[3]

Since containers are technically considered just ’guests’ running on a ’host’, they are seen as less flexible. This is somewhat true because they can only run an

operating system environment that is based on the same kernel as the operating system of the host. For instance, CentOS Linux could be run on an Ubuntu server, but Microsoft Windows cannot be.

Containers are alleged to be less secure than the hypervisor virtualization because full in virtualization, each instance is a fully-blown system with its own operating system with restricted interactions amongst themselves. [3] Countering this is the fact that containers, being lightweight, lack the larger attack surface that full operating system exhibits. This is besides the vulnerabilities of the hypervisor where system call translations take place.

Despite all these probable limitations, containers have been deployed in several use cases. Containers are popular for large-scale deployments of multi-client services, for lightweight sandboxing and as process isolation environments. chroot jail is a common example of a container that creates an isolated directory environment for running processes. Hackers that illegally find their way inside the environment often find themselves trapped in the container posing no threat to the actual host itself.

OpenVZ, Solaris Zones, Linux Containers (like lxc) etc. can look like full-blown hosts by themselves rather than just execution environments. Docker has modern Linux features like control groups and namespaces. This means that containers can have very strong memory and process isolation, individual network and storage stacks, as well as resource management capabilities to allow co-existence of multiple containers within a host.

Containers are generally considered a lean technology because they require limited overhead. Unlike traditional virtualization or paravirtualization technologies, they 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 and can allow a greater density of containers to run on a host.

Containers in general, have not achieved large-scale adoption. This is largely because of the fact that containers can be complex, extremely hard to set up and difficult to manage. Docker is a tool that aims to remove all the above drawbacks. In fact, it is justifiable to say that Docker has, to a large extent, succeeded in achieving that. Today, Docker is a tool that is commonly used in the industry with many major players 'dockerizing' their applications.

1.2 Docker

Docker is an open-source tool that automates the deployment of application into containers. A team at Docker Inc. (earlier known as dotCloud Inc.) developed Docker as a Platform-as-a-Service (PaaS) and released it under the Apache 2.0 license.

Docker is special in a unique way. It provides an application deployment engine atop a virtualized container execution environment. Docker provides a lightweight, fast environment not just to run an application, but also provide a container that contains all the required dependencies to run the application.

Developers can now easily share their code; testers and developers can amicably work together regardless of the platform each is working in. Docker is remarkably simple. In fact, one could get started by simply running a Docker binary on a compatible Linux kernel. Docker's mission is to provide the following -

- AN EASY AND LIGHTWEIGHT WAY TO MODEL REALITY
- A LOGICAL SEGREGATION OF DUTIES
- FAST, EFFICIENT DEVELOPMENT LIFE CYCLE

1.3 Managing Big Data with Hadoop: HDFS and MapReduce

Hadoop is an open-source software framework that uses the Hadoop Distributed File System (HDFS) and MapReduce to analyze big data in a distributed environment.

The Hadoop Distributed File System was developed to easily manage large volumes of data in a simple way. MapReduce decomposes big problems into smaller elements to compute results quickly and effectively. Each of the smaller elements can be executed on a different system in a distributed environment. MapReduce is a software framework that enables writing of programs to process massive amounts of unstructured data in a distributed environment. It was

designed by Google to efficiently execute a set of functions on a large chunk of data in batch mode.

The ’map’ component of MapReduce distributes the problem in a grid and handles the allocation of tasks in a manner that balances the load on the systems and manages recovery from failures. Upon the completion of the several subtasks assigned to the systems in the distributed environment, the aggregation function ’reduce’ puts back together all the individual results to provide the overall outcome.

HDFS is not the final destination for files. Instead, it is a data service that offers a unique set of capabilities needed when data volumes and velocity are high.

CHAPTER 2

VIRTUALIZATION VS CONTAINER BASED VIRTUALIZATION

Container Virtualization is a contemporary to Virtual Machines. But, the main difference between Virtual Machines and Container Virtualization is that, containers run on the host Operating System, instead of running a seperate guest Operating System on top of the host OS. It can hence be concluded that Hypervisors [1] are not used in Container Virtualization.

Both Virtual Machines and Containers achieve the same objective - to achieve platform independence. But, the primary advantage of Container Virtualization over Virtual Machines is that the implementation of the former does not involve the usage of two Operating Systems. Containers are thus extremely faster and efficient as compared to Virtual Machines. The following is a diagrammatic comparison of Container with Virtual Machines.

The main problem faced by system administrators and developers was that it was working on dev and qa but it wasn't working on production environment. Well the problem most of the times can be a version mismatch of some library or few packages not being installed etc. This is where Docker steps in, and solves the aforementioned problem forever, by making an image of an entire application, with all it's dependencies and ship it to your required target environment / server. So, it is easy to conclude that if the application worked in your local system, it should work anywhere in the world (as the entire thing is being shipped).[3]

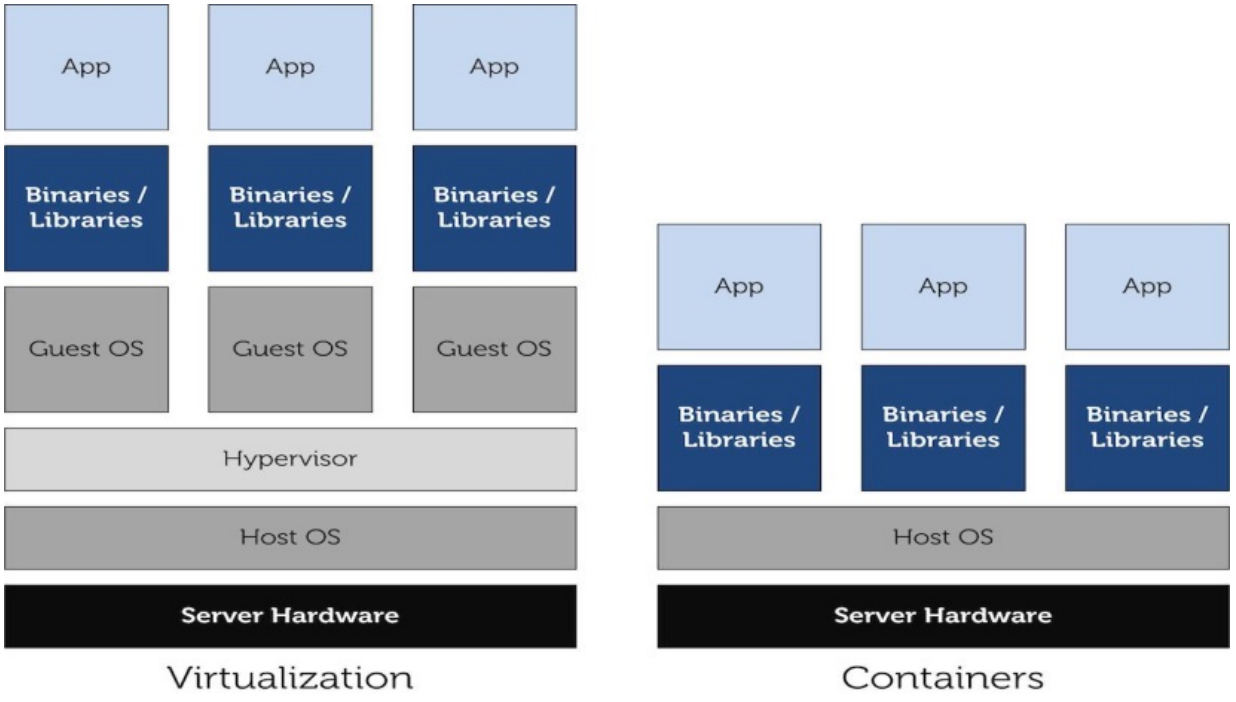

FIGURE 2.1: Virtualization vs Container Based Virtualization, source :http://cloudacademy.com/blog/wp-content/uploads/2015/09/12.jp

It is easy for us to think that Hypervisor based Virtualiztion can solve the problem of was working in dev and qa but not in production, by taking an image of an entire virtual host and launching a new virtual instance from it(the thing which we usually do in aws, or openstack). Alas, this can be done. But a container is so light weight that we need not go through such trouble of setting up a new host just for your application. In fact, it just takes a matter of seconds to pull a container image from a registry and start it. This is the main advantage of using Docker Container Virtualization. We will not be discussing Docker here, simply because it need special attention and requires a series of posts to cover it.

Definition 2.1. Its nothing but a method or technique used to run an operating system on top of another operating system. So the hardware resources are fully

utilized and will be shared by each of the operating system running on top of the base operating system .

The basic concept behind a hypervisor based virtualization is to emulate the underlying physical hardware and create virtual hardware(with your desired resources like processor and memory). And on top of these newly created virtual hardware an operating system is installed(Guest OS). To put it in simple words, the basic role of the hypervisor is that it translates the system calls of the guest OS to that of the host OS. So this type of virtualization is basically operating system agnostic. In other words, you can have a hypervisor running on a linux system create a virtual hardware and can have windows installed on top of that virtual hardware, and vice-versa.

Definition 2.2. A Hypervisor is also called as a virtual machine Monitor(VMM), This is because the hypervisor sits in between the guest operating system and the real physical hardware. Hypervisor controls the resource allocation to the guest operating system running on top of the physical hardware.

What is Container Virtualization?

The one common thing which found while discussing hosted and bare metal virtualization is that both of them are based on a hardware level(basically they are virtualizing hardware resources). But container virtualization is done is Operating System / Kernel level, rather than the hardware level.
Now it is obvious to identify the main advantage of container based virtualization over hosted and bare metal. As each containers are sitting on top of the same kernel, and sharing most of the base operating system. Thus, containers are much smaller and lighter as compared to a virtualized guest OS. Due to its light

weighted nature, an operating system can have many containers running on top of it, as compared to the limited number of guest operating system which can be run on top of the host OS.

Advantages of using Docker :

- The main advantage of using docker for container based virtualiztion is because of it's ability to easily build, and ship containers. Yes, it is possible for you to ship your applications to anybody or a remote server without any hassle i.e The entire container is shipped(So, it is guaranteed that whatever worked on your local system will surely work on your target environment).
- There is GIT like version controlling feature for container. It is possible for us to make changes to your applications and commit it and upload it your repository or ship the latest version to your target environment.
- There is an excellent community shared repository for containers, called the Docker Hub, where you can get ready made containers with your required open source app on top of it. Containers can be pulled easier and faster by just using a shell command, like yum repositories ,for packages.
- Reuse : Any existing container built by others can be used to modify and make your own version of the container. Also, there are concepts such as base images, on top of which we can have our own configuration and required environment for your custom application.

CHAPTER 3

COMPONENTS AND ARCHITECTURE

Docker has 4 core components, namely :

- The Docker client and server
- Docker Images
- Registries
- Docker Containers

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 client binary, 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. Dockers architecture is shown here:

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.

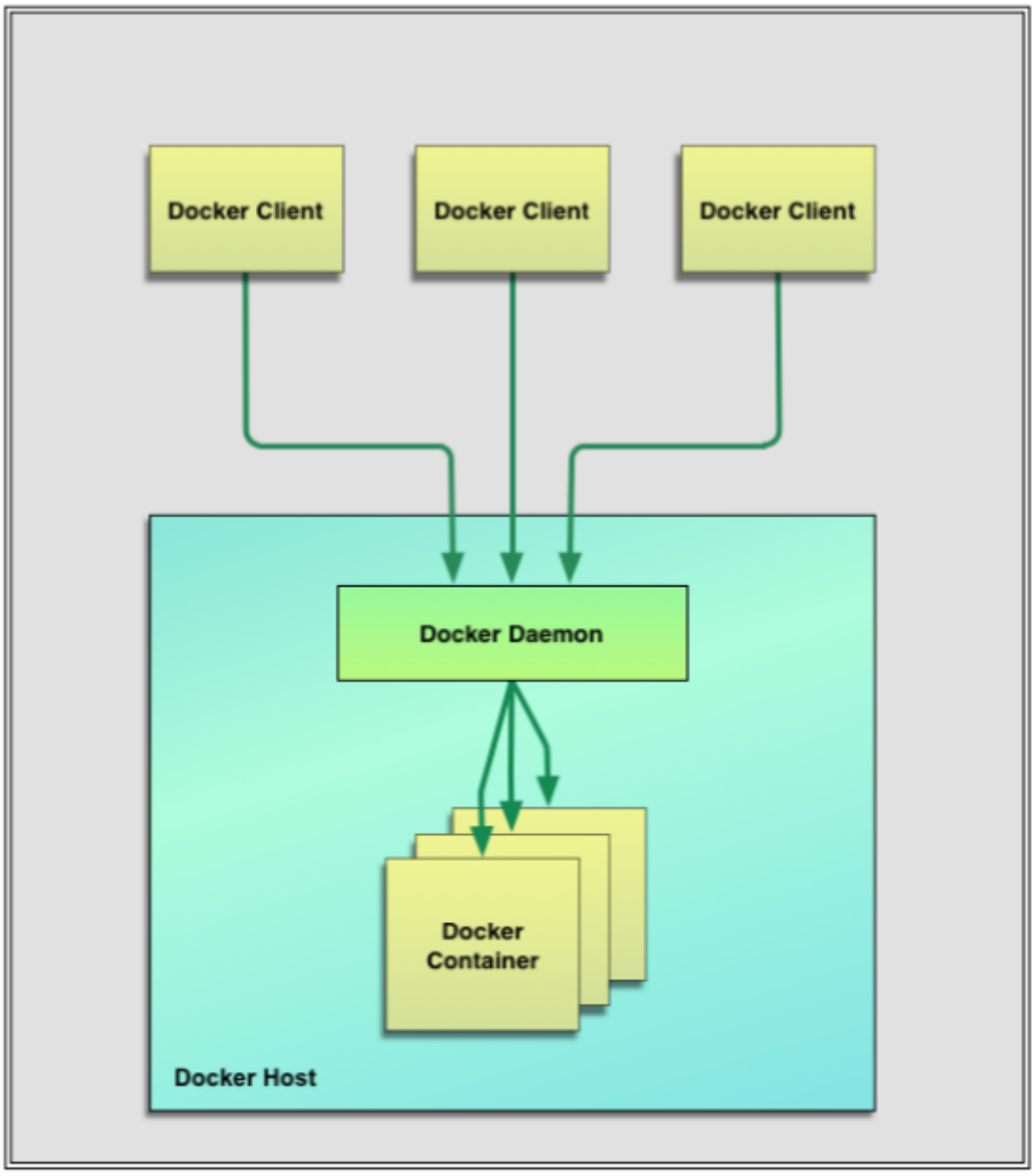

FIGURE 3.1: Docker Architecture
[3]

Registries

The built images are housed in the registries. The two key types of registries are : private and public registries.

Docker operates the public registry for images, called the Docker Hub. You can

create an account on the Docker Hub and use it to share and store your own images.

People have built and shared over ten thousand odd images on the hub. Docker images for various open source applications are available. Users can also store images that they want to keep private on the Docker Hub. These images might contain source code or other proprietary information they want to keep secure or only share with other members of your team or organization.

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 and institutions.

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. Images can be considered as the building or packing aspect of Docker. The containers can be viewed as the running or execution aspect of Docker.

A Docker container is:

- An image format.
- A set of standard operations.
- An execution environment.

Docker mirrors the concept of the standard shipping container, which is used to transport goods globally. But here, instead of shipping goods, Docker containers are used to ship software.

Each container contains a software image which is nothing but the cargo and, like its physical counterpart, allows a series of operations to be performed. For instance, it can be created, started and restarted. When necessary it can also be stopped and destroyed.

Just like a shipping container, the contents of the container don't matter to Docker when performing these actions; for instance, whether a container is a web server or a database, or an application server. Each container is loaded the same way just as any other container would be loaded.

The destination to which the container is shipped doesn't matter to Docker. It can be built on your laptop, uploaded to a registry and then downloaded to a physical or virtual server. It can then be tested and deployed to a cluster of a dozen Amazon EC2 hosts, and can be run. Just like a normal shipping container, it is generic, interchangeable, portable and as stackable as possible.

With Docker, an application server can be quickly built, along with a message bus, a utility appliance and a CI test bed for an application. It can build local, self-contained test environments or replicate complex application stacks for production or development purposes. The different possible uses of Docker are infinite.

3.1 System Architecture

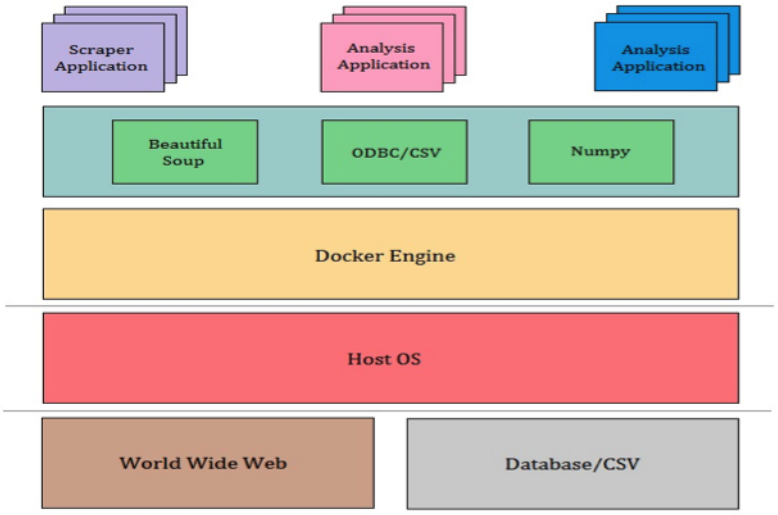

FIGURE 3.2: System Architecture

CHAPTER 4

HADOOP DISTRIBUTED FILE SYSTEM (HDFS) FOR BIG DATA PROJECTS

The Hadoop Distributed File System is a versatile, resilient, as well as a clustered approach to managing files in a big data environment. HDFS is never the final destination for files. Rather, it is used as a data service that offers a unique set of functionalities needed when the data volumes as well as the velocity are high. Since the data is usually written once and then read many times after that, HDFS is a very good option for supporting big data analysis in comparison to the constant read-writes of other file systems.

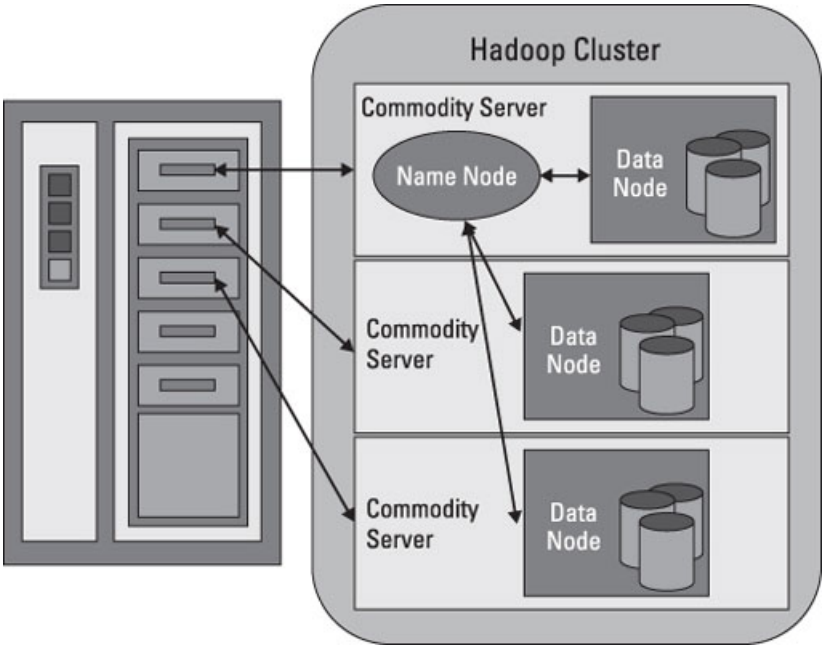

FIGURE 4.1: Docker Architecture
[3]

Big data NameNodes

HDFS works by fragmenting large files into smaller pieces known as blocks. These blocks are stored on data nodes, and it is the duty of the NameNode to know what blocks on which data nodes respectively constitute the complete file. The NameNode may also act as a traffic cop as it manages all access to the files.

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.

Despite a strong relationship existing between the NameNode and the data nodes, they usually function in a loosely coupled fashion. This enables the cluster elements to behave in a dynamic fashion, thereby adding more servers as the demand increases. In a typical configuration, you would find one NameNode and possibly a data node running on one physical server in the rack. The other servers usually run only data nodes.

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.

Big Data Nodes

Though Data nodes are not smart, they are quite resilient. Within the HDFS cluster, the data blocks are replicated across various data nodes and access to

these nodes is managed by the NameNode. This mechanism is designed for optimal efficiency when the entire collection of nodes of the cluster are collected into a rack. In fact, the NameNode makes use of a rack ID to keep track of the data nodes within the cluster.

Usually , the data nodes also provide heartbeat messages to check and verify connectivity between the NameNode and the data nodes. When a heartbeat is no longer present, the NameNode will unmap the data node from the cluster and continues operating as though nothing ever happened. When the heartbeat resurfaces, it is then added to the cluster transparently with respect to either the user or the application.

Data integrity is another key feature. HDFS supports a number of functionalities designed specifically to provide data integrity. When files are usually broken into several blocks and then distributed across the different servers in the cluster, any variation in the operation of even a single element could affect the integrity of the data. HDFS makes use of transaction logs and checksum validation in order to ensure integrity within and across the cluster.

Transaction logs are used to keep track of every operation and are most effective in auditing the file system. They are also used to rebuild the file system in case of some unknown or unwanted operation.

Checksum validations are often used to guarantee and ensure the contents of files in HDFS. When a file is requested by a client, its contents can be verified by examining its checksum. If there is a match in the checksum, the file operation can proceed. If not, an error will be reported. Checksum files are kept hidden to

ensure that tampering is avoided.

Data nodes make use of local disks in the commodity server for persistence. All the data blocks are stored locally, mainly for reasons pertaining to performance. Data blocks can be replicated across several data nodes, so that the failure of a single server may not necessarily corrupt a file. The degree to which it is replicated and the number of data nodes as well as the HDFS namespace are established when the cluster is first implemented.

HDFS for big data

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 1GB file takes up eight 128MB blocks for its storage needs.

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 is defined as data about data. Imagine HDFS metadata to be a template for providing a comprehensive description of the following points :

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

- Who holds the rights to modify or view the file
- How many different files are stored on the cluster
- The number of data nodes that exist within the cluster
- 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.

CHAPTER 5

HADOOP MAPREDUCE FOR BIG DATA

For us to completely understand the capabilities and functionality of the Hadoop MapReduce, its important to know the difference between MapReduce (the algorithm) and one of the implementations of MapReduce(Hadoop MapReduce). Hadoop MapReduce is an implementation of the MapReduce algorithm which is developed, maintained and monitored by the Apache Hadoop project.

To help in the understanding of it's working, It is easier to think about this implementation of MapReduce as an engine, as that is exactly how it works. You provide input(fuel) to the "engine", the "engine" converts the input provided, into the required output quickly and efficiently, and you get the answers you seek.

The Hadoop MapReduce process includes several stages, each of which involves important set of 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 Distibuted File System(HDFS).

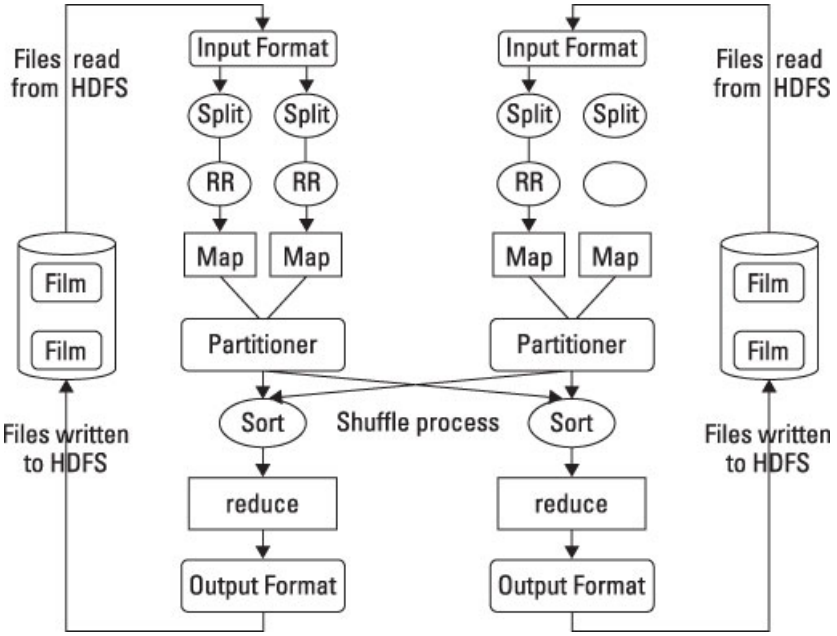

FIGURE 5.1: Docker Architecture
Source: docker-for-dummies.com

The work done by HDFS and MapReduce are performed on nodes in a cluster which is hosted on numerous racks of commodity servers. To simplify the discussion, the diagram shows only two nodes.

Get the big data ready

When a client initiates a MapReduce program to be executed, the first step is to locate the input file and to read the raw data contained in it. The format of the file can be completely arbitrary, but the data which is being processed must be converted to something the MapReduce program can process. This is taken care of by the RecordReader and the InputFormat. The job of the InputFormat is to decide as to how the input file is going to be broken into smaller pieces for it to be processed by a method called InputSplit.

After the InputFormat, it then assigns a RecordReader to change the raw data for processing by the map. There are Several types of RecordReaders which are supplied with Hadoop, providing us with a wide variety of conversion options. The ways that Hadoop manages the huge variety of data types found in big data problems is provided by the aforementioned feature.

Once the big data map begins, the data is now in a suitable form which is acceptable to map. A distinct instance of map is usually called upon to process the data for each input pair.

What happens to the processed output, and how can one keep track of them? Map has two additional functionalities to address the above questions. Since map and reduce need to work hand in hand to process the data, the program will need to collect the output from the various independent mappers and then pass it on to the reducers. This task is usually performed by an OutputCollector. A Reporter function is used to provide necessary information gathered from map tasks so that the user will get to know in case the map tasks are complete.

All of this work is being divided and performed on multiple nodes in the Hadoop cluster concurrently. There may be some cases in which the output from certain mapping processes needs to be augmented before the reducers can begin. Or, in some cases, some of the intermediate results may have to be processed before the reduction process.

In addition to this, some of the output may be on a node which is different

from the node where the reducers for the specific output will be running. A partitioner and a sort handles the gathering and shuffling of the intermediate results. The results obtained by the map tasks will be fed to a specific partition as input to the reduce tasks.

Once all the map tasks are finished, all the intermediate results are accumulated in the partition and then a shuffling occurs, sorting the output for the optimal processing by reduce tasks.

Reduce and combine for big data

For each of the output pair obtained in map, reduce is called to perform its task. In similarity to map, reduce accumulates its output while all the tasks are being processing. Reduce cannot begin until all the mapping is finished. The output of the reduce is also of a key and a value pair. While this is necessity for reduce to complete its work, it may not be the most effective output format for the application being made.

Hence, Hadoop provides an OutputFormat feature, which works just like the InputFormat. OutputFormat takes all the key-value pairs and organizes the obtained output for writing it to the HDFS. The final task is to actually write the data onto the HDFS. This operation is performed by RecordWriter, and it performs very much like the RecordReader except it's in reverse. It takes the OutputFormat data and writes it onto HDFS, in the form which is necessary for the requirements of the program.

In earlier versions of Hadoop, the coordination of all these activities was managed by a job scheduler. This scheduler was naive and rudimentary, and as the mixture of jobs changed and grew in size, it was clear that a different approach was necessary to be adopted. The main drawback in the old scheduler was the lack of management of resources. The latest version of Hadoop incorporates this new capability.

The Hadoop MapReduce is the heart of the Hadoop system. It provides all the capabilities and functionalities needed to break big data into manageable smaller chunks, process the fragmented data in parallel on the distributed cluster, and then make the output data available for user consumption or additional processing. Also, it does all this work in a highly fault-tolerant and resilient manner. This is just the beginning of something great.

CHAPTER 6

DATA ACQUISITION, PROCESSING AND REPRESENTATION

6.1 BeautifulSoup

”You didn’t write that awful page. You’re just trying to get some data out of it.” Beautiful Soup is a parser for HTML and XML documents available as a Python package. A parse tree is created for the pages that are parsed and using this tree, data can be extracted from HTML and XML documents. This process is commonly dubbed as ’web scraping’.

Beautiful Soup consists of simple methods to navigate and search through the parse tree. It also supports making modifications to the tree where necessary. Beautiful Soup can be termed as a toolkit used to dissect a document to extract the essential portions of the document. It is extremely simple to use Beautiful Soup as it automatically converts input documents to Unicode and output documents to UTF-8. Beautiful Soup supports most common encodings. Popular Python parsers like lxml and html5lib can be used with Beautiful Soup as Beautiful Soup sits on top of these parsers. Further, a combination of these can be used to try out different parsing techniques in order to balance speed and flexibility. Beautiful Soup is available for Python 2.6+ and Python 3.

6.2 iPython Notebook

iPython is a command shell that can be used for interactive computing. It initially supported Python programming language only but has grown to accept several other scripts. iPython supports code, text, rich media, mathematical expressions, inline plots and other media. The iPython notebook is a browser-based notebook that can be used to display the content that iPython deals with.

The iPython Notebook is an agile tool used for computation and for data analysis. It consists of two components -

- A web application in which text, mathematics, computations and rich media can be showcased. Here, input and output are both stored in persistent cells that can be edited in-place.
- A notebook app that is used to automatically save the current state of the computation in the web browser.

iPython uses a standard text file with the extension .ipynb. Despite the notebook documents being plain text files, they carry a unique extension because they use the JSON format to store a complete, reproducible copy of the current state of computation within the Notebook app. This means that changes made to the project are automatically updated real-time.

6.3 Jupyter Notebook

Jupyter Notebook is an application that allows editing and executing notebook documents easily within a web browser. It is a client-server application where the local desktop acts as the client that is connected to a remote server from which the data can be fetched.

Apart from displaying, editing and executing notebook documents, Jupyter Notebook also has a dashboard known as the 'Notebook Dashboard'. The dashboard is a control panel used to show and open local files and also to shut down the kernel.

The Jupyter Notebook used to be called the iPython Notebook. However, from iPython 4.0, the format of the notebook, the message protocol, the notebook web application etc. have been moved to a new project under the name of 'Jupyter'. iPython, by itself, focuses on interactive Python only. It is also worthwhile to note that the interactive Python capabilities of the Jupyter Notebook are imparted by the Python kernel that iPython provides.

6.4 Numpy

Numpy (pronounced 'Numb-Pie' or 'Numb-Pee') is a package that can be used with the Python programming language. It is used to add support for huge, multi-dimensional arrays and matrices. Numpy also contains a large library of high-level mathematical functions that can used on the huge multi-dimensional arrays. Numpy was derived from a similar yesteryear

tool known as 'Numeric'. Numpy was created in 2005, when its creator, Travis Oliphant incorporated several compelling features from Numeric's main competitor 'Numarray' into Numeric, with extensive modifications. Numpy is open source and freely available to all.

6.5 Pandas

Pandas is a package used with the Python programming language for data manipulation and analysis. It offers data structures and operations on the data structures for manipulating numerical tables and time series. Pandas is open source and is freely available for use by all. Surprisingly, the name 'Pandas' is derived from the term 'panel data', a term used for multidimensional data sets that are structured.

he Wikipedia page for Pandas lists the following as its features -

- DataFrame object for data manipulation with integrated indexing
- Tools for reading and writing data between in-memory data structures and different file formats
- Data alignment and integrated handling of missing data Reshaping and pivoting of data sets
- Label-based slicing, fancy indexing, and subsetting of large data sets
- Data structure column insertion and deletion
- Group by engine allowing split-apply-combine operations on data sets
- Data set merging and joining

- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure
- Time series-functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging

6.6 Matplotlib

Matplotlib is a 2-dimensional plotting library used with the Python programming language to produce various ways of data representation and provide interactive environments across platforms for users to access the data. Matplotlib can be natively used with iPython, python scripts and MATLAB. In this project, we generate plots of the data using Matplotlib and visualize it using the iPython Notebook within a browser.

Matplotlib is open source and is free to use by all. It has a very active community that was used several times during the development of this project. As of October 2015, matplotlib 1.5.x supports Python versions 2.7 - 3.5.

6.7 ID3 Algorithm

A decision tree is a format of representing a model of decisions and its possible consequences. It also includes the probability of each outcome and the associated result.

Decision trees are often used in operations research where it is required to analyze the various decisions and to identify a path that will most likely reach the goal. They are extremely popular in the field of Machine Learning. Decision trees are fairly simple to construct and extremely simple to understand.

Each internal node in the decision tree represents a 'test' on one of the attributes of the data. For instance, whether it is sunny or rainy or overcast at a given time could be values of the weather attribute. Decision trees can consist of decision nodes (represented by squares), chance nodes (represented by circles) and end nodes (represented by triangles).

The ID3 (Iterative Dichotomiser 3) algorithm 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. The more advanced C4.5 algorithm was developed from the ID3 algorithm and is commonly used in machine learning and natural language processing applications to generate decision trees today. Most digital personal assistants like Cortana, Siri and Google Now use an improved version of the C4.5 algorithm to make decisions.

The ID3 algorithm chooses a different attribute that is tests at each level of the decision tree. The cardinal aim of the ID3 algorithm is to choose the order of attributes to be tested along the different levels of the decision tree in a manner that rests most information on the attribute chosen the first (at the first level) and least information on the attribute chosen the last (at the last level).

A statistical property - Information Gain - is used to achieve the above. 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 information that rests in an attribute in a given data set.

This can be elucidated using an example. Thousands of trains run successfully from their departure stations to their arrival stations successfully. Very very rarely, a train undergoes an accident and does not successfully travel from its departure station to its arrival station. However, despite the probability of such an accident being extremely low. the importance associated with such an incident is extremely high.

Entropy H(S) is defined as follows

$$H(S) = - \sum_{x \in X} p(x) \log_2 p(x)$$

where,

- 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

Information Gain is calculated as follows

Inline image 1

where - H(S) - Entropy of set S

- T - The subsets created from splitting set S by attribute A

- 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

(Formulae ref - https://en.wikipedia.org/wiki/ID3$_{algorithm}$)

Data Description

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

- Attribute-value description - the same attributes must describe each example and have a fixed number of values.
- Predefined classes - an example's attributes must already be defined, that is, they are not learned by ID3.
- Discrete classes - classes must be sharply delineated.
- Continuous classes broken up into vague categories such as a metal being "hard, quite hard, flexible, soft, quite soft" are suspect.
- Sufficient examples - since inductive generalization is used (i.e. not provable) there must be enough test cases to distinguish valid patterns from chance occurrences.

Attribute Selection

How does ID3 decide which attribute is the best? A statistical property, called information gain, is used. Gain measures how well a given attribute separates

training examples into targeted classes. The one with the highest information (information being the most useful for classification) is selected. In order to define gain, we first borrow an idea from information theory called entropy. Entropy measures the amount of information in an attribute.

Given a collection S of c outcomes

Entropy(S) = S -p(I) log2 p(I)

where p(I) is the proportion of S belonging to class I. S is over c. Log2 is log base 2.

Note that S is not an attribute but the entire sample set.

CHAPTER 7

UML DIAGRAMS OF PROPOSED SYSTEM

7.1 Class Diagram

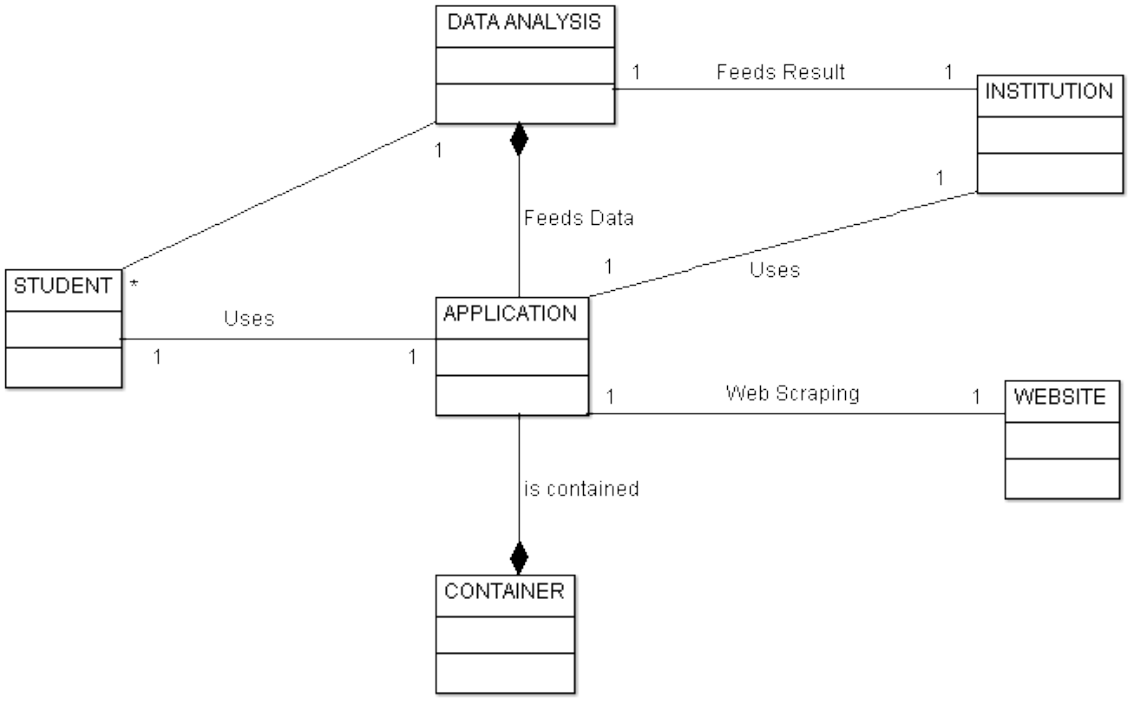

FIGURE 7.1: Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

7.2 Dataflow Diagram

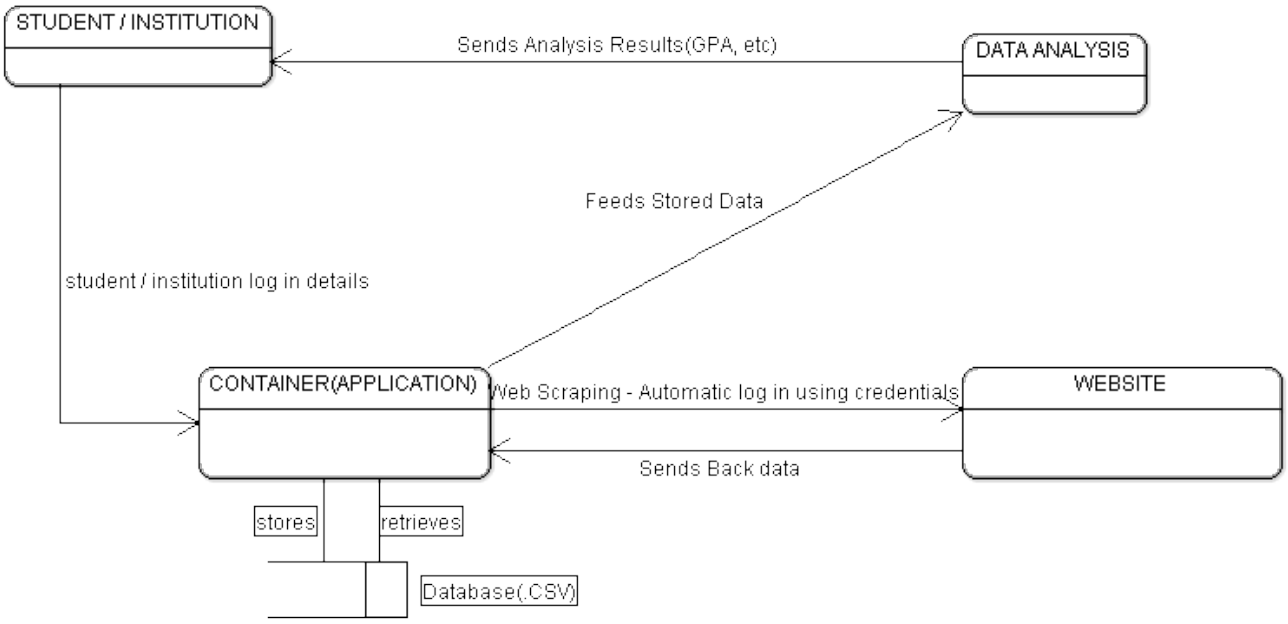

FIGURE 7.2: Dataflow Diagram

A data flow diagram (DFD) is a graphical representation of the ”flow” of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

7.3 Entity-Relationship Diagram

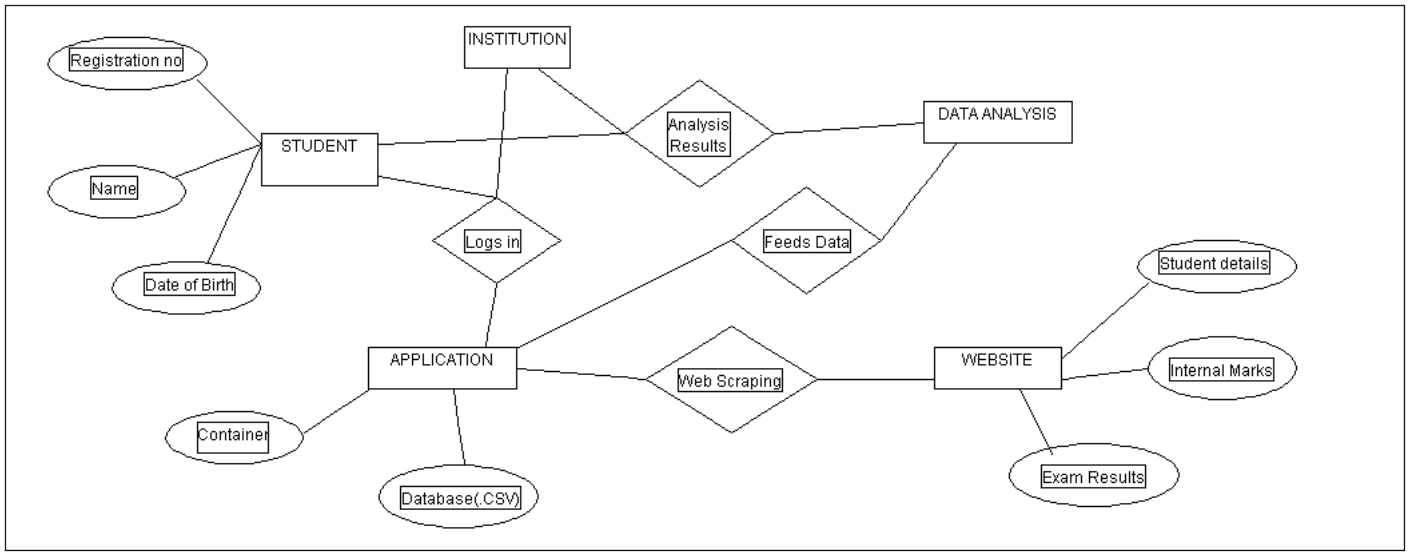

FIGURE 7.3: ER Diagram

Diagrams created to represent these entities, attributes, and relationships graphically are called entityrelationship diagrams. An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity.

7.4 Usecase Diagram

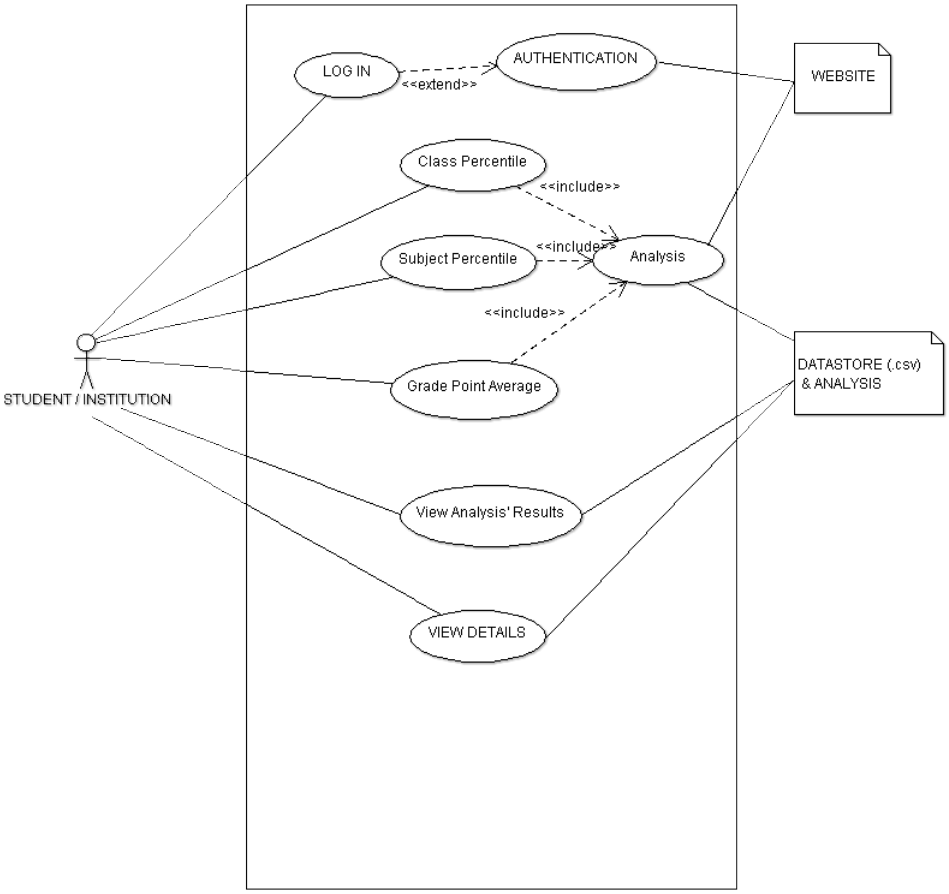

FIGURE 7.4: Usecase Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

CHAPTER 8

RESULTS

The application was effectively deployed on multiple platforms without any change in the underlying source code. This is possible because the project's container image contains all dependencies within the execution environment regardless of the platform.

The application scrapes examination results from a HTML page and converts them into a CSV file. Using Hadoop, the most probable grade for each subject can be predicted. Students are classified into 6 groups based on their past performance. Using ID3 algorithm, the application can predict the group to which a student belongs to.

Plots that depict the performance of the students are generated for each subject using matplotlib and plotly.The application also supports shared volumes, live migration (nascent implementation).

With the ability to scale up or down and security isolation without the hassles of managing physical servers, Containers have secured their place in the technology world. The popularity of Containers are on the rise with several big players in the industry adopting and supporting Dockers. The most recent announcement on that front is the support for native Docker containers for Windows 10 in the coming months.

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

development and deployment. Though we are not there yet, we are closer than we ever have. And Container Virtualization is a huge leap in the right direction.

CHAPTER 9

CONCLUSION AND FUTURE WORK

Containers are getting popular by the day. Thanks to this emerging technology, we now have a container-based solution 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 other containers. However, containers share the resources of the host machine. The project 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.

Docker is reliant on some features of the Linux Kernel - namely namespaces and cgroups. Namespaces are used to abstract a resource and build an isolated workspace. Logistics about the resource are only visible to those processes that are a part of the namespace that the resource forms a part of. Cgroups help to distribute the available hardware resources to the processes and effectively manage them.

The project contains an effective implementation of container-based virtualization. It can be used to scrape data from the internet and perform analysis on the data scraped. The project can be further improved with a more intuitive user interface and 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.

With the ability to scale up or down and security isolation without the hassles of managing physical servers, Containers have secured their place in the technology world. The popularity of Containers are on the rise with several big players in the industry adopting and supporting Dockers. The most recent announcement on that front is the support for native Docker containers for Windows 10 in the coming months.

Containers are a great path to better applications, both in the cloud and on-premises. Containers make the lives of the developers and testers easy. There will be a day when platforms are no longer a hindrance to effective application development and deployment. Though we are not there yet, we are closer than we ever have. And Container Virtualization is a huge leap in the right direction.

REFERENCES

1. Alex Williams (2014) 'The New Stack: The Docker and Container Ecosystem eBook Series'

2. Felter,W., Ferreira,A., Rajamony,R., Rubio,J. (2014) 'An Updated Performance Comparison of Virtual Machines and Linux Containers', IBM Research, Austin, TX

3. James Turnbull (2014) 'The Docker Book'

4. Liu J.H., Li N.(2011) ' Optimized ID3 algorithm based on attribute importance and convex function'

5. Sarath Pillai (2014) 'http://www.slashroot.in/difference-between-hypervisor-virtualization-and-container-virtualization'

6. Simplivity, White Paper (2015) 'Accelerating DevOps with Docker and SimpliVity OmniStack Hyperconverged Infrastructure'