Docker in support of Big Data Applications and Analytics

Anand Sriramulu
Indiana University
107 S Indiana Ave
Bloomington, Indiana, USA 47405
asriram@iu.edu

ABSTRACT

Different uses cases on how docker can improve the performance of Big Data applications.

KEYWORDS

i523, hid338, Data Science, Docker, Containers, Big Data Analytics, Cloud Computing

1 INTRODUCTION

The rapid acquisition of big data and development of computationally intensive analysis has led to need for novel approaches to software deployment. Containers are a solution that allow software to run from one computing environment to another, whether it be from a developer's computer to a test environment or a physical machine in data center to a virtual machine (VM) in a private or public cloud.

Docker is a newer type of container technology and an open source platform for developers and sysadmins to build, ship, and run applications.

Docker has the same concept as a container used for cargo ships: a standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery. In between, containers can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another. [**?**]

Docker enables any payload to be encapsulated as a lightweight, portable, self-sufficient container that can be manipulated using standard operations and run consistently on different systems.

Docker containers are isolated, but share the Linux OS kernel, and where appropriate, bins/libraries. This results in significantly faster deployment, much less overhead, easier migration and faster performance. You share the host OS services, so unlike a Virtual Machine, you're not replicating the OS

2 DOCKER BENEFITS

In a seemingly constant state of maturation, the benefits of using Docker increase on a regular basis. Here, I'll outline the top five benefits of using the ever-growing platform. [**?**]

2.1 Continuous Deployment and Testing

Docker is gaining a lot of traction in the development and devops world for its consistency across environments. There are always minor differences between environments in development and release lifecycles, unless you have your own private repository environment with tight checks in place. These differences may be because of different package versions or dependencies. Nevertheless, Docker can address that gap by ensuring consistent environments from development to production. Docker containers are configured to maintain all configurations and dependencies internally. As a result, you can use the same container from development to production making sure there are no discrepancies or manual intervention.

With Docker containers, you can also ensure that developers don't need an identical production environment set up. Instead, they can use their own system to run Docker containers on VirtualBox. If you need to perform an upgrade during a product's release cycle, you can easily make the necessary changes to Docker containers, test them, and implement the same changes to your existing containers. This sort of flexibility is a key advantage of using Docker. Just like standard deployment and integration processes, Docker allows you to build, test and release images that can be deployed across multiple servers. Even if a new security patch is available, the process remains the same. You can apply the patch, test it and release it to production.

2.2 Multi-Cloud Platforms

One of Docker's greatest benefits is portability. Over last few years, all major cloud computing providers, including Amazon Web Services (AWS) and Google Compute Platform (GCP), have embraced Docker's availability and added individual support. Docker containers can be run inside an Amazon EC2 instance, Google Compute Engine instance, Rackspace server or VirtualBox, provided that the host OS supports Docker. If this is the case, a container running on an Amazon EC2 instance can easily be ported between environments, say to VirtualBox, achieving similar consistency and functionality. This grants you a level of abstraction from your infrastructure layer. In addition to AWS and GCP, Docker works very well with various other IaaS providers like Microsoft Azure, and OpenStack. [**?**]

2.3 Environment Standardization and Version Control

As discussed above, Docker containers ensure consistency across multiple development and release cycles, standardizing your environment. On top of that, Docker containers work just like GIT repositories, allowing you to commit changes to your Docker images and version control them. Suppose you perform a component upgrade that breaks your whole environment. It is very easy to rollback to a previous version of your Docker image. This whole process can be tested in a few minutes. When compared to VM backup and image creation processes, Docker is fast, allowing you to quickly make replications and achieve redundancy. Additionally, launching Docker images is as fast as running a machine process.

2.4 Isolation

Docker ensures your applications and resources are isolated and segregated. Docker containers are as good as VM hypervisors when it comes to isolating resources, but there is still work to be done in terms of management and administration. [**?**]

Consider a scenario where you are running multiple applications on your VM. These applications can be team collaboration software (e.g., Confluence), issue tracking software (e.g., JIRA), centralized identity management systems (e.g., Crowd) and so on. Seeing as all of these applications run on different ports, you would have to leverage them on Apache and Nginx as a reverse proxy. So far, everything is in good shape, but as your environment moves forward, you will also need to configure a content management system (e.g., Alfresco) into your existing environment. Bear in mind that it requires a different version of Apache Tomcat, which will cause a problem. In order to fix this, you can either move your existing applications to another version of Tomcat or run your content management system (Alfresco) on your currently deployed version.

Fortunately, with Docker, you don't have to do this. Docker makes sure each container has its own resources that are isolated from other containers. You can have various containers for separate applications running completely different stacks. Aside from this, effectively removing applications from your server is quite difficult and may cause conflicts with dependencies. However, Docker helps you ensure clean app removal since each application runs on its own container. If you no longer need an application, you can simply delete its container. It won't leave any temporary or configuration files on your host OS.

On top of these benefits, Docker also ensures that each application only uses resources (CPU, memory and disk space) that have been assigned to them. A particular application won't hog all of your available resources, which would normally lead to performance degradation or complete downtime for other applications.

2.5 Security

Docker is evolving at a fast pace, which Gartner even acknowledges, as mentioned above. From a security standpoint, Docker ensures that applications that are running on containers are completely segregated and isolated from each other, granting you complete control over traffic flow and management. No Docker container can look into processes running inside another container. From an architectural standpoint, each container gets its own set of resources ranging from processing to network stacks.[**?**]

As a means of tightening security, Docker uses host OS sensitive mount points (e.g., '/proc' and '/sys') as read-only mount points and uses a copy-on-write filesystem to make sure containers can't read each other's data. It also limits system calls to your host OS and works well with SELinux and AppArmor. Additionally, Docker images that are available on Docker Hub are digitally signed to ensure authenticity. Since Docker containers are isolated and resources are limited, even if one of your applications is hacked, it won't affect applications that are running on other Docker containers.

3 BIGDATA AND DOCKER

Big Data is one of the big trends in IT of recent years. The vast majority of CIOs are collecting and managing more business information than they did two years ago.

CIOs and IT Operations have a common goal: prepping their IT infrastructure to manage the data deluge and growing revenue by making better use of the data they collect.

They also share some common frustrations. Often the right systems are not in place to gather the information they need, and many struggle to give their business managers access to pertinent information.

Arming an organization with the appropriate technology, staff, and systems/processes needed to optimize information for true business intelligence can help manage the data deluge. The following approaches can be applied to increase the chances of a successful outcome.

3.1 Use Docker To Avoid Dependency Issues

Each developer might have different set of big data tools and not to mention all the dependencies required, which then must be distributed to each machine in a cluster.

Companies assume this situation is manageable, but get enough developers on the same cluster and it doesn't take long for one tools requirements to break another. This will cause all the dependencies issues.

In this situation, there are two choices - get an entire development team to standardize on a common toolset, or use Docker. Docker allows each tool to be self contained, along with all of its dependencies. This means that a application can have different jobs use different versions of the same tool without a conflict.

This frees up your DevOps team to use the best tools for the data processing job, or set up entirely new systems and drive incredible scale and efficiency.

3.2 Reduce Reliance On MapReduce Experts With Pachyderm

For sysadmins that have a large amount of data to analyze, the go-to method has typically been to run MapReduce queries on Hadoop. This typically requires specialist programmers who specialize in writing MapReduce jobs, or hiring a third party such as Cloudera.

This typically means that Big Data initiatives require a lot of co-ordination internally and require resources that are beyond the reach of even large enterprises who do not have that kind of expertise on tap.

Alternatively, Pachyderm is a tool that allows programmers to implement a http server inside a Docker container, then use Pachyderm to distribute the job. This has the potential to allow sysadmins to run large scale MapReduce jobs quickly and easily to make product level decisions, without knowing anything about MapReduce.[4]

Pachyderm has the ambition of replacing Hadoop entirely - whether it achieves that remains to be seen, but it certainly looks like it will be a significant player in the next generation of data processing

3.3 Run Scheduled Analytics Using Containers With Chronos

By reading the above section, it's evident that the containers are a great way of deploying services at scale and giving isolation to services that run on the same host and improving utilization, but Docker can also be used for batch processing as well.

The latest release of the Chronos job scheduler for Mesos allows you to launch Docker instances into a Mesos cluster. This provides developers and sysadmins with the ability to run scheduled analytics jobs using containers.[1]

Chronos allows you to schedule Docker containers to run ETL, batch and analytics applications without manual setup on your cluster nodes. One of the neat features of Chronos is that it will also produce a dependency graph between scheduled jobs that depend on each other, so they only run if the previous job is successful.

Chronos and Marathon combine really nicely to provide orchestration for a container infrastructure.

3.4 Provision A Big Data Dev Environment Using Ferry

Ferry allows you to create big data clusters on the local machine (and AWS). The beauty of Ferry is that it allows anyone to define a big data stack using YAML, and then share it with other developers using a Dockerfile.[3] Setting up a Hadoop cluster is as simple as:

```
backend:
- storage
personality: "hadoop"
instances: 2
layers:
- "hive"
Connectors:
- personality: "hadoop-client"
```

Get started by typing

```
ferry start hadoop
```

This will create a two node Hadoop cluster and a single Linux client. This can be customized at runtime or defined using a Dockerfile. Ferry is great for developers who want to get up and running with a big data environment using a test AWS box, developers that need a local big data dev environment, or users that want to share Big Data applications.

Running Ferry on AWS also has several advantages over something like Elastic MapReduce, such as not tying you to a single cluster of a single type (such as Hadoop).

3.5 Run Big Data As Microservices With Coho

When we talk to enterprise customers about Big Data processing, there are one or two recurring themes. For example, in healthcare, there are frequent workflows where new data triggers a new action.

Taking transcoding as an example. When a new image is pushed to the storage system, a transcoding workflow will take place, reading the data back to a client machine or VM, transcoding it, and then writing the results back to storage. This can mean that the data has to cross the network three times!

In Big Data environments, data might be pushed out to a separate HDFS-based analytics system, only to be pushed back to the enterprise system when the job has been run.[2]

Coho has worked on a storage-integrated tool that allows developers and DevOps teams to think specifically about workflows as operations on data, and for them to be embedded in the storage system.

These resulting extensions can then run efficiently and transparently at scale as the system grows. This theoretically allows presentation layers to be built on top of existing data, for the system to be extended with audit and compliance functionality and for complex, environment based access controls to be built.

4 CONCLUSIONS

The complex nature of big data and the tools used to analyze these data sets makes efficient processing difficult with standard environments. As noted above, the use of emerging technologies such as Docker in combination with automated workflows may significantly improve the efficiency of data processing in data analytics. With the growing number of open data projects, use of these techniques will be necessary to take advantage of available computational resources.

While performance and pipeline efficiency were key components of this implementation, Docker containers also allow for application isolation from the host operating system. Since many big data tools have complex sets of dependencies and are difficult to build from source, the ability to deploy containers with different operating systems and dependency versions to the same host decreases the amount of effort needed to being analysis. For example, the cgDownload utility is distributed as a compiled binary for use on CentOS 6.7, but can only be deployed on CentOS 7 when built from source, which requires a significant amount of manual configuration. With the use of containers allowed the deployment of each utility on its natively supported operating system, which improves stability and decreases the potential for dependency conflicts among software applications.

Several other tools exist for the orchestration of containerized applications, such as Kubernetes and Docker Swarm. For complex platforms, these tools can be used to deploy containers across hardware clusters and to integrate networking and storage resources between containers. However, these applications work strictly at the container level and do not inherently provide application-level workflows as presented here. Additional implementation experience about the use of these tools within high-performance clusters may provide valuable insights about the scalability of these tools within data analytics workflows.

Because of the subsequent increase in analysis throughput, use of these tools means that big data analyses can be done even with limited local computational capacity. Finally, use of container technology can improve pipeline and experimental reproducibility since preconfigured applications can be readily deployed to nearly any host system. While many factors can impact reproducibility, the use of containers limits variability due to differences in software environment or application configuration when appropriately deployed. The continued use of emerging technology and novel approaches

to software architecture has the potential to increase the efficiency of computational analysis in big data.

REFERENCES

[1] [n. d.]. Chronos. ([n. d.]). https://mesosphere.com/blog/docker-on-mesos-with-chronos/
[2] [n. d.]. Coho. ([n. d.]). http://www.cohodata.com/blog/2015/04/09/docker-data-services-microservices/
[3] [n. d.]. Ferry. ([n. d.]). http://drydock.readthedocs.io/en/latest/
[4] [n. d.]. Pachyderm. ([n. d.]). http://www.pachyderm.io/open_source.html

5 BIBTEX ISSUES

Warning–I didn't find a database entry for "About-Docker"

Warning–I didn't find a database entry for "Docker-Benefits"

Warning–I didn't find a database entry for "Multi-Cloud-Platform"

Warning–I didn't find a database entry for "Container-VM"

Warning–I didn't find a database entry for "Docker-Security"

Warning–no key, author in pachyderm

Warning–no author, editor, organization, or key in pachyderm

Warning–to sort, need author or key in pachyderm

Warning–no key, author in Chronos

Warning–no author, editor, organization, or key in Chronos

Warning–to sort, need author or key in Chronos

Warning–no key, author in Ferry

Warning–no author, editor, organization, or key in Ferry

Warning–to sort, need author or key in Ferry

Warning–no key, author in Coho

Warning–no author, editor, organization, or key in Coho

Warning–to sort, need author or key in Coho

Warning–no key, author in Chronos

Warning–no key, author in Chronos

Warning–no key, author in Coho

Warning–no key, author in Coho

Warning–no key, author in Ferry

Warning–no key, author in Ferry

Warning–no key, author in pachyderm

Warning–no key, author in pachyderm

Warning–no key, author in Chronos

Warning–no author, editor, organization, or key in Chronos

Warning–empty author in Chronos

Warning–empty year in Chronos

Warning–no key, author in Coho

Warning–no author, editor, organization, or key in Coho

Warning–empty author in Coho

Warning–empty year in Coho

Warning–no key, author in Ferry

Warning–no author, editor, organization, or key in Ferry

Warning–empty author in Ferry

Warning–empty year in Ferry

Warning–no key, author in pachyderm

Warning–no author, editor, organization, or key in pachyderm

Warning–empty author in pachyderm

Warning–empty year in pachyderm

(There were 41 warnings)

6 ISSUES

DONE:

Example of done item: Once you fix an item, change TODO to DONE

6.1 Uncaught Bibliography Errors

Missing bibliography file generated by JabRef - what did you use? you're missing a lot of data

Citations in text showing as [?]: this means either your report.bib is not up-to-date or there is a spelling error in the label of the item you want to cite, either in report.bib or in report.tex

6.2 Formatting

Other formatting issues - abstract missing several sentences

6.3 Writing Errors

Errors in title, e.g. capitalization

Errors - in tone and voice; this is not a manual

6.4 Citation Issues and Plagiarism

It is your responsibility to make sure no plagiarism occurs. The instructions and resources were given in the class

Claims made without citations provided

Need to quote directly cited material - e.g. http://www.iamondemand.com/blog/5-key-benefits-of-docker-ci-version-control-portability-isolation-and-security/

6.5 Character Errors

Erroneous use of quotation marks, i.e. use "quotes" , instead of " "

6.6 Structural Issues

Acknowledgement section missing