Projects in Big Data Software and Applications

Spring 2017

Bloomington, Indiana

Editor:
Gregor von Laszewski
Department of Intelligent Systems
Engeneering
Indiana University
laszewski@gmail.com

Contents

S17-IO-3004/report/report.pdf not submitted

S17-IO-3009/report/report.pdf not submitted

Prototyping a Virtual Robot Swarm with ROS and Gazebo

MATTHEW LAWSON[1] AND GREGOR VON LASZEWSKI[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: laszewski@gmail.com*

project-000, March 27, 2017

Our virtual robot swarm prototype accomplishes a *TBD task* by allocating portions of the task to each virtual robot (VR). As each VR works on its piece of the task, it communicates relevant information back to the master VR. Upon completion, the master VR collates the results and creates a human-readable report. The virtual swarm utilizes the *Robot Operating System* to control the virtual robots, *Gazebo* to simulate the task completion and *RVIZ* to visualize the process. We use *Ansible* to deploy the software to a distributed computing environment. Th importance of our effort centers on some super-special conclusion I do not yet grasp.

Keywords: Cloud, I524, ROS, Gazebo, Robot, Swarm

https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IO-3010/report/report.pdf

1. INTRODUCTION

Stating the Problem: Simulating a single robot's actions and responses to its environment prior to real-world deployment mitigates risk and improves results at a relatively low cost. It follows that simulating the actions and responses of a group of robots, e.g., a swarm, will also improve results at a low cost. However, deployment of an interconnected swarm of virtual robots requires much more time and effort than a single virtual robot. Collecting the results from a swarm also requires additional effort.

Our Contribution: We create a cross-platform system to quickly and relatively easily deploy and manage a swarm, as well as evaluate the swarm's operational effectiveness. Or maybe something else...I'm not sure, yet. Automate the deployment of a virtual robot swarm that will accomplish some arbitrary task; capture data from the swarm as it completes its task; report back the results in a human-readable format.

2. VIRTUAL ROBOT SWARM COMPONENTS

2.1. Robot Operating System (ROS)

TBD; will include a discussion of a) how to obtain and install ROS and b) ROS graph concepts. The latter topic will introduce ROS' core components, namely a node, publications, suscriptions, topics and services. This section should probably cover ROS packages and ROS client libraries (primarily C++ and Python)

2.2. Gazebo

TBD; again, introducing core Gazebo concepts. In this case, will include a) world files, b) model files and c) its client-server model. It should also include non-obvious limitation examples, e.g., a gripper arm driven by a single screw instead of multiple screws. In addition, it should allude to any limitations that affect our simulation.

2.3. Ansible

TBD; briefly describe Ansible - what it is, salient features, etc.

2.4. Testing Environment

TBD; briefly describe cloudmesh

3. VIRTUAL ROBOT SWARM PROJECT IMPLEMENTATION

3.1. VR Swarm task

TBD; discuss the task to be accomplished by the swarm, as well as how the information collected during task completion will be communicated back to the master node for collation and reporting.

3.2. Deployment

TBD; document the Ansible steps needed to successfully deploy ROS and Gazebo on multiple computers; will include references for obtaining major components, including adding new repositories if needed.

3.3. Modifications, Pitfalls

TBD; discuss any obstacles encountered with deployment due to dependency problems, connecting ROS and Gazebo, etc.

3.4. Initializing the Swarm

TBD; starting ROS and Gazebo to create the virtual environment; testing swarm interconnectivity; designating master node, etc.

3.5. Begin Task and Monitor Swarm's Progress

TBD; discuss the steps to initiate task completion and monitor the swarm's progress;

3.6. Information Acquired

TBD; discuss the iformation obtained from the swarm wrt the task at hand as well as each node's vital signs, e.g., battery level;

3.7. Updating Software

TBD; discuss the methods used to implement software updates on each node; remain cognizant of battery levels *I have no idea how I might accomplish an over-the-air update of ROS. This point intimidates me.*

4. VR SWARM PROJECT CONCLUSIONS

TBD; present the data collected in some visualization format; discuss why this project advances robotics forward by utilizing distributed computing;

5. EXAMPLES OF ARTICLE COMPONENTS

The sections below show examples of different article components.

6. FIGURES AND TABLES

It is not necessary to place figures and tables at the back of the manuscript. Figures and tables should be sized as they are to appear in the final article. Do not include a separate list of figure captions and table titles.

Figures and Tables should be labelled and referenced in the standard way using the `\label{}` and `\ref{}` commands.

6.1. Sample Figure

Figure 1 shows an example figure.

6.2. Sample Table

Table 1 shows an example table.

Table 1. Shape Functions for Quadratic Line Elements

local node	$\{N\}_m$	$\{\Phi_i\}_m$ $(i = x, y, z)$
$m = 1$	$L_1(2L_1 - 1)$	Φ_{i1}
$m = 2$	$L_2(2L_2 - 1)$	Φ_{i2}
$m = 3$	$L_3 = 4L_1L_2$	Φ_{i3}

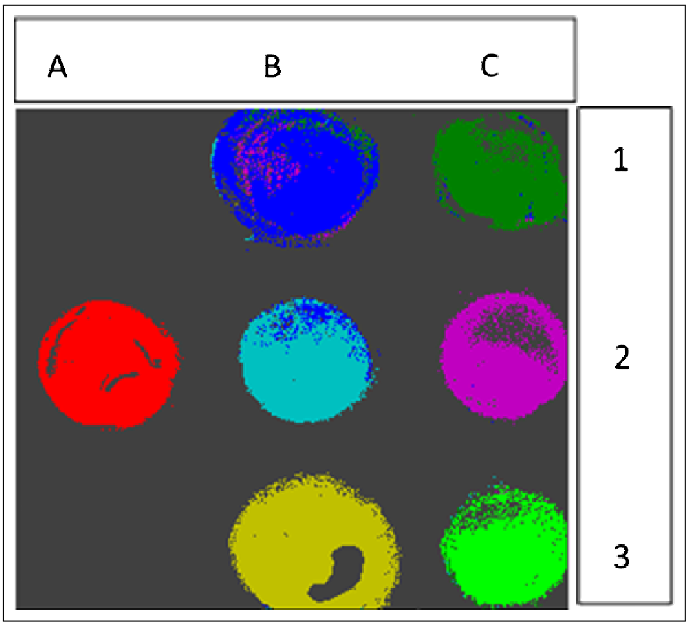


Fig. 1. False-color image, where each pixel is assigned to one of seven reference spectra.

7. SAMPLE EQUATION

Let $X_1, X_2, \ldots, X_n$ be a sequence of independent and identically distributed random variables with $\mathrm{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2 < \infty$, and let

$$S_n = \frac{X_1 + X_2 + \cdots + X_n}{n} = \frac{1}{n}\sum_{i}^{n} X_i \qquad (1)$$

denote their mean. Then as n approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converge in distribution to a normal $\mathcal{N}(0, \sigma^2)$.

8. SAMPLE ALGORITHM

Algorithms can be included using the commands as shown in algorithm 1.

Algorithm 1. Euclid's algorithm

```
1: procedure EUCLID(a, b)          ▷ The g.c.d. of a and b
2:     r ← a mod b
3:     while r ≠ 0 do              ▷ We have the answer if r is 0
4:         a ← b
5:         b ← r
6:         r ← a mod b
7:     return b                    ▷ The gcd is b
```

Algorithm 2. Python example

```
for i in range(0,100):
  print i
```

9. REFERENCE MANAGEMENT

The best programs to manage your references is jabref or emacs. You can edit the references and verify them with them for format errors. To cite them use the citation key. You can add multiple bib files to the bibliography command separated by comma. Add citations with the cite command. See [1] for an example on how to use multiple clouds. In [2] we list the class content.

Here a test of a citation with an underscore in the url [3].

10. SUPPLEMENTAL MATERIAL

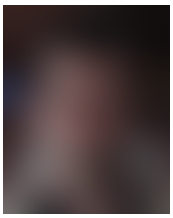
REFERENCES

[1] G. von Laszewski, F. Wang, H. Lee, H. Chen, and G. C. Fox, "Accessing Multiple Clouds with Cloudmesh," in *Proceedings of the 2014 ACM International Workshop on Software-defined Ecosystems*, ser. BigSystem '14. New York, NY, USA: ACM, 2014, pp. 21–28. [Online]. Available: http://doi.acm.org/10.1145/2609441.2609638

[2] Gregor von Laszewski and Badi Abdul-Wahid, "Big Data Classes," Web Page, Indiana University, Jan. 2017. [Online]. Available: https://cloudmesh.github.io/classes/

[3] Web Page. [Online]. Available: http://www.google.com/some_underscore

AUTHOR BIOGRAPHIES



John Smith received his BSc (Mathematics) in 2000 from The University of Maryland. His research interests include lasers and optics.

Alice Smith received her BSc (Mathematics) in 2000 from The University of Maryland. Her research interests also include lasers and optics.

Bruce Wayne received his BSc (Aeronautics) in 2000 from Indiana University. His research interests include lasers and optics.

A. WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Matthew Lawson. Designed the project in collaboration w/ Gregor von Laszewski, researched the material and implemented the project. Slept far too little.

Gregor von Laszewski. Provided invaluable insights at key points during the process.

B. REPORT CHECKLIST

- □ Have you written the report in word or LaTeX in the specified format?
- □ Have you included the report in github/lab?
- □ Have you specified the names and e-mails of all team members in your report. E.g. the username in Canvas?
- □ Have you included the HID of all team members?
- □ Does the report have the project number added to it?
- □ Have you included all images in native and PDF format in gitlab in the images folder?
- □ Have you added the bibliography file in bibtex format?
- □ Have you submitted an additional page that describes who did what in the project or report?
- □ Have you spellchecked the paper?
- □ Have you made sure you do not plagiarize?
- □ Have you made sure that the important directories are all lower case and have no underscore or space in it?
- □ Have you made sure that all authors have a README.rst in their HID github/lab repository?
- □ Have you made sure that there is a README.rst in the project directory and that it is properly filled out?
- □ Have you put a work breakdown in the document if you worked in a group?

Charge Detection Mass Spectrometry

SCOTT MCCLARY[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: scmcclar@indiana.edu

project-001, March 27, 2017

A Charge Detection Mass Spectrometry research application is used to show the benefits of using Ansible Galaxy. Previously, this properitery research application was installed by hand on local servers or Supercomputers. Transfering the input data to remote systems as well as aggregating/visualizing the results is difficult. Improving this research workflow by automating the deoployment of the necessary software subsystems assists in building an efficient, reproducible and scalable Charge Detection Mass Spectrometry research workflow.

Keywords: Chemistry, Cloud, HPC, I524, Parallel Computing

https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IO-3011/report/report.pdf

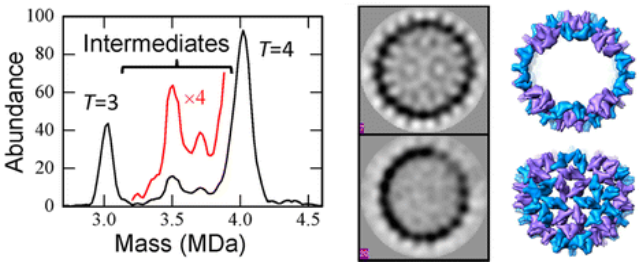


Fig. 1. The chart to the left displays an accurate measurement of the Hepatitis B virus (HBV) created by the research group's CDMS application [**?**]. This detailed mass information is used to create the images shown in the middle and to the right, which show 2-D and 3-D models of HBV.

1. INTRODUCTION

The Martin F. Jarrold research group studies Charge Detection Mass Spectrometry. Their workflow consists of conducting scientific experiments using a Mass Spectrometer. This instrument creates raw data throughout each experiement. They have build an application a Fast Fourier based application written in Fortran that processes the output files to determine detailed mass information of the substance used in the aformentioned experiment. The detailed mass information outputted from the application can be used to solve important research topics such as the measure of the the Hepatitis B virus, shown in figure 1.

2. EXECUTION PLAN

The following subsections act as a timeline regarding how I broke the project up week-by-week in order to complete the entire project by the desired deadline. The project execution plan is simply a guide and was followed diligently; however, some items were pushed forwards/backwards as technological challenges were faced.

Scott McClary: need to update the plan as timing information changes throughout the semester.

2.1. March 6, 2017 - March 12, 2017

This week I installed Cloudmesh on my local machine, created my first Virtual Machine on the Chameleon Cloud and tested Ansible Galaxy on remote systems such as one or more Chameleon Cloud VM's. I also wrote the project proposal, which will eventually become the project reoprt.

2.2. March 13, 2017 - March 19, 2017

This week I tested the deployment of the Intel Compiler on one or more Chameleon Cloud VM's using Ansible Galaxy. I did not expect significant progress to be made during this weeek given that I was out of town for Spring Break.

2.3. March 19, 2017 - March 26, 2017

This week I configured the Intel Compiler to use the Indiana University Intel license server. This required connecting to Indiana University's VPN from the command line.

2.4. March 27, 2017 - April 2, 2017

This week I deployed the Charge Detection Mass Spectrometry along with the required input data on one or more Chameleon Cloud VM's using Ansible Galaxy.

Fig. 2. CDMS Pipeline

2.5. April 3, 2017 - April 9, 2017

This week I benchmarked both the deployment and the analysis on at least one cloud (i.e. Chameleon Cloud). I also created a method to aggregate the output from one or more VM's and locally visualize the results.

2.6. April 10, 2017 - April 16, 2017

This week I wrote the majority of the project report.

2.7. April 17, 2017 - April 23, 2017

This week I ensured the reproducibility of my source code as well as revised the final version of the report.

3. ANSIBLE GALAXY

Ansible Galaxy was leveraged in order to automate the deployment of the required software subsystems, user code and data.

3.1. Software Subsystems

The CDMS application relies on the Math Kernel Library (MKL) to leverage efficient Fast Fourier Computations. The application also leverages the OpenMP parallel framework in order to divide the work amongst available CPU's. Therefore, in order to compile and run the application, the Intel compiler is required, which provides the MKL and OpenMP functionality.

3.2. User Code

The Martin F. Jarrold Group has written a Fast Fourier Based application written in Fortran in order to conduct their CDMS research. This application is approximately 15,000 lines of code. Depending on the input, about 60% to 70% of the compute time is spent within external MKL libraries conducting FFT calculations.

3.3. Data

The CDMS application inputs a set of raw 2 MB files. In order to develop and test the efficiency of the deployment, a small and large dataset was used. The small test dataset (i.e. 200 files) has a total size of 400 MB and the large dataset (i.e. 4,506 files) has a total size of 9.012 GB. A typical dataset for the research group is approximately the size of the large dataset. In a single day, 7 to 10 datasets are created and need to be processed. When an algorithmic change occurs to the research application, a large batch of archived data requires reprocessing. In this case, terabytes of data may be processed. This is why the parallelization and therefore the scalability of the application is critical to the Martin F. Jarrold research group.

Working to figure out where I will store the data and how to partition the data amongst available VMs.

4. CDMS RESEARCH PIPELINE

5. LICENSING

The Intel Compiler requires a license in order to complete the installation. A student license is obtainable; however, the leveraginng the Indiana University Intel license would provide a more complete solution. In order to use the Indiana University Intel license, the Virtual Machines have be in the Indiana University IP address space or have to have to be connected to Indiana University's Virtual Private Network (i.e. VPN). In order to connect to the VPN, one must first connect via DUO Authentication (i.e. use a phone to validate as well).

Scott McClary: need to continue working to connect to IU's VPN (and DUO) from the command line. So far this solution is not yet complete. Need to figure out licensing information for Jarrold Group code.

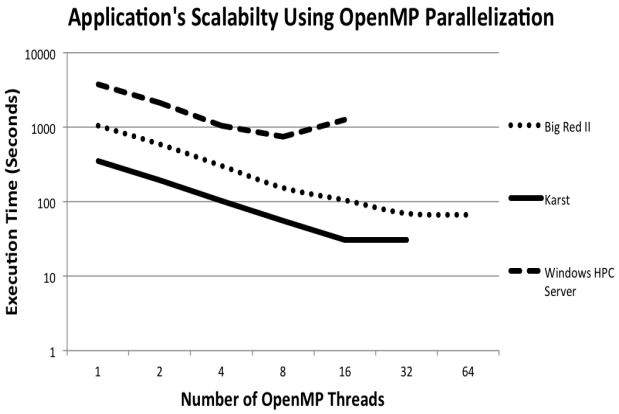


Fig. 3. The figure above shows the scalability (i.e. reduction in time-to-solution) as the number of OpenMP threads increase on local servers, Supercomputers and Clouds.

6. BENCHMARK

As discussed in section 3.2, the application is parallelized using OpenMP. Therefore, this application utilizes the avaialable computational power available. Figure 3 compares the performance of the application on difference compute resources (i.e. local servers, Supercomputers and clouds).

The time required to deploy and run the application in the cloud is shown in the figure (TBD). This benchmark includes the time required for the installation of the software subsystems as well as the time required to run the application.

Add timing information for one or more clouds (i.e. Chameleon) to Figure 3 and compare the results (if possilbe).

6.1. OpenMP Scalability

7. CONCLUSION

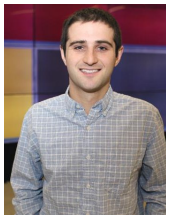
The use of Ansible Galaxy to run the Charge Detection Mass Spectrometry application in the Cloud (e.g. Chameleon Cloud) imporved the efficiency, reproduciblity and scalability. In comparison to running on the Indiana University HPC clusters (e.g. Karst and Big Red II), the application time-to-solution diminished sigificantly. The most important and useful tool that was developped as a result of this project was the automation of the deployment of the necessary software subsystems, the application itself, the necessary input data and aggregation/visualization of the output. The use of Ansible Galaxy within this research workflow will allow the Martin F. Jarrold

research group to focus on the details of their specific research rather than on the details of managing the software subsystems, running the application and managing the input/output data.

ACKNOWLEDGEMENTS

The authors would like to thank the School of Informatics and Computing for providing the Big Data Software and Projects (INFO-I524) course [1]. This project would not have been possible without the technical support & edification from Gregor von Laszewski and his distinguished colleagues.

AUTHOR BIOGRAPHIES



Scott McClary received his BSc (Computer Science) and Minor (Mathematics) in May 2016 from Indiana University and will receive his MSc (Computer Science) in May 2017 from Indiana University. His research interests are within scientific application performance analysis on large-scale HPC systems. He will begin working as a Software Engineer with General Electric Digital in San Ramon, CA in July 2017.

WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Scott McClary. He completed all of the work for this paper including researching and testing Apache Airavata as well as composing this technology paper.

REFERENCES

[1] Gregor von Laszewski and Badi Abdul-Wahid, "Big Data Classes," Web Page, Indiana University, Jan. 2017. [Online]. Available: https://cloudmesh.github.io/classes/

Automated Sharded MongoDB Deployment and Benchmarking for Big Data Analysis

MARK MCCOMBE[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: laszewski@gmail.com

S17-IO-3012, March 27, 2017

Using Python, Ansible, and Cloudmesh Client scripts, a fully automated process is created for deploying a configurable MongoDB sharded cluster on Chameleon, FutureSystems, Jetstream, and (possibly) Amazon Web Services (AWS) cloud computing environments. A user runs one program, which configures and deploys the environment based on input parameters for Config Servers, Mongos Instances, Shards, and a Replication Factor. Additionally, the automated process will run multiple benchmarking tests for each deployment, capturing statistics in a file as input for python (STILL EVALUATING OPTIONS) visualization programs, the results of which are displayed in this report. As background, technologies and concepts key to the deployment and benchmarking, such as MongoDB, Python, Ansible, Cloudmesh Client, and Openstack are examined.

Keywords: MongoDB, Cloud Computing, Ansible, Python, Cloudmesh Client, Openstack, I524

https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IO-3012/report/report.pdf

1. INTRODUCTION

As the final project for I524, Big Data Software and Projects, Spring 2017, a Python program invoking Ansible playbooks has been created to fully automate a configurable deployment of a MongoDB sharded cluster on various clouds. Chameleon Cloud, FutureSystems, and Jetstream are the currently supported clouds. The scripts have been developed and tested on an Ubuntu 16.04 LTS (Xenial Xerus) Virtual Machine running in Virtual Box. In addition to Ansible, Cloudmesh Client is used for key interaction between the client and cloud environment. Options can be specified for deployment cloud, replication degree of the Config Server servers, number of Mongos instances, number of Data Shards, and the degree of replication within the Shards.

As part of the deployment, automated benchmarking tests are also run. Tests were performed with various sharding and replication configurations to assess their impact on performance. Performance results are captured and graphed using Python's matplotlib, the results of which are displayed and analyzed in this report.

2. EXECUTION PLAN

A rough execution plan is shown below. These steps are not necessarily sequential so many will be worked on concurrently. All functionality mentioned may not be added to final project and additional features may be added during development.

1. Convert my existing MongoDB deployment ksh prototype to Python cmd and Ansible
2. Test and utilize cloudmesh clusters
3. Test new version on Chameleon Cloud
4. Test new version on FutureSystem
5. Port to Jetstream and test
6. Depending on Cloudmesh functionality available, xplore porting to AWS. May do some testing of this early on to allow for building in to program since non OpenStack may work differently.
7. Explore adding x.509 security option to security option to existing keyfiles
8. Add Map/Reduce test to existing mongoimport and find benchmarking tests
9. Consider rewriting visualization originally done with python using another language or tool
10. Complete paper

2.1. Update 3/26/2016

1. Project approved by Professor based on proposal and execution plan.
2. Successfully tested Chameleon and FutureSystems cloud and cloudmesh access and commands.
3. Tested JetStream access, but unable to connect via cloudmesh. Question to Professor on Piazza.
4. Completed all Ansible lessons/assignements as preparation for project work. Beginning to work on incorporating to project.
5. Completed cloudmesh cluster tutorial. Beginning to work on incorporating to project.
6. Made modifitions to report.

3. INFRASTRUCTURE

Will make major modifications

Two clouds were selected for deployment: Chameleon Cloud and Futuresystems. In our automated deployment and benchmarking process, the -d option determines which cloud the deployment will be run on.

3.1. OpenStack

Both Chameleon Cloud and FutureSystems use OpenStack. OpenStack is a free, open source cloud computing platform, primarily deployed as IaaS. [1] Openstack was created in 2010 as joint project between NASA and Rackspace that is currently managed by the OpenStack Foundation. [1] Open Stack is open source software released under the Apache 2.0 license. [2]

Open Stack has various components, also known by code names [1]. Examples of Openstack components (and code names) are Compute (Nova), Networking (Neutron), Block Storage (Cinder), Identity (Keystone), Image (Glance), Object Storage (Swift), Dashboard (Horizon), Orchestration (Heat), Workflow (Mistral), Telemetry (Ceilometer), OpenStack Telemetry (Ceilometer), Database (Trove), Elastic Map Reduce (Sahara), Bare Metal (Ironic), Messaging (Zaqar), Shared File System (Manila), DNS (Designate), Search (Searchlight), and Key Manager (Barbican) [1].

3.2. Chameleon Cloud

Chameleon is funded by the National Science Foundation and provides computing resources to the open research community. The Chameleon testbed is hosted at the Texas Advanced Computing Center and the University of Chicago. Chameleon provides resources to facilitate research and development in areas such as Infrastructure as a Service, Platform as a Service, and Software as a Service. Chameleon provides both an OpenStack Cloud and Bare Metal High Performance Computing Resouces. [3]

If *chameleon* is specified for the -d parameter, the automated deployment and benchmarking process will be run on Chameleon Cloud.

3.3. FutureSystems

FutureSystems is a computing environment run by Indiana University that supports educational and research activities. [4] FutureSystems is directed by Geoffrey C. Fox and Gregor von Laszewski, both of Indiana University. [5] For our deployment, we utilized the OpenStack Kilo Cloud, running on the India machine.

If *kilo* is specified for the -d parameter, the automated deployment and benchmarking process will be run on Kilo Cloud.

3.4. Jetstream

3.5. Amazon Web Services

3.6. Cloud Hardware Comparison

Will make major modifications

Figure 1 shows a comparison of several key computing resources on Chameleon, FutureSystems, and Jetstream cloud environments.

Need to add Jetstream info. https://jetstream-cloud.org/technology.php

	FutureSystems	Chameleon	Jetstream
CPU	Intel Xeon E5-2670 v3	Intel Xeon X5550	Dual Intel E-2680v3 "Haswell"
CPU cores	1024	1008	7680
CPU speed	2.66GHz	2.3GHz	2.5GHz
RAM	3072GB	5376GB	placeholder
RAM speed	1333Mhz	unknown	unknown
switches	Juniper/Dell EX Force 10	Dell S6000	
storage	335TB	1.5PB	2 TB

Fig. 1. Cloud Hardware Specification Comparison
[6] [7]

MAYBE CHANGE THIS TO PER VM OBVIOUSLY REFORMAT, probably flip columns and rows

4. ANSIBLE

5. PYTHON/CMD

6. CLOUDMESH CLIENT

The Cloudmesh Client toolkit is an open source client interface that standardizes access to various clouds, clusters, and workstations. [8] Cloudmesh Client is a python based application developed by Gregor von Laszewski and others at Indiana University.

In the deployment, Cloudmesh Client is used to handle most interaction with the Virtual Machines in the clouds. The following tasks are performed via Cloudmesh Client:

1. Uploading a Public Key - Cloudmesh's key add and upload commands simplify key management.
2. Uploading Security Rules - Cloudmesh's secgroup commands allow new security rules to be added and uploaded to the cloud.
3. Booting Virtual Machines - Cloudmesh's vm boot command allow easy creation of virtual machines.
4. IP Address Management - Cloudmesh's functionality to assign and capture Floating IP Addresses is utilized
5. Deleting Virtual Machines - Cloudmesh simplifies deletion of VMs created in the deployment.

Cloudmesh Client simplifies and standardized interaction with the cloud for these tasks. This allows us to more easily port the deployment to additional clouds. Furthermore, by

encapsulating the logic necessary to perform these tasks we are shielded from changes in interfaces made by individual clouds.

More details of the use Cloudmesh Client in the automated deployment are included in Appendix B.

7. MONGODB

MongoDB is a popular open source, document oriented noSQL database. It stores documents in JSON-like schema-less formats called collections. [9] DBEngines ranks MongoDB as the most popular noSQL store and as the fifth most popular Database Management System overall. [10]

7.1. Architecture

A sharded cluster in MongoDB has three main components, all of which will be implemented in our deployment:

- Config Servers - hold configurations setting and metadata
- Mongos - a query routing interface between applications and the cluster
- Shards - subsets of data

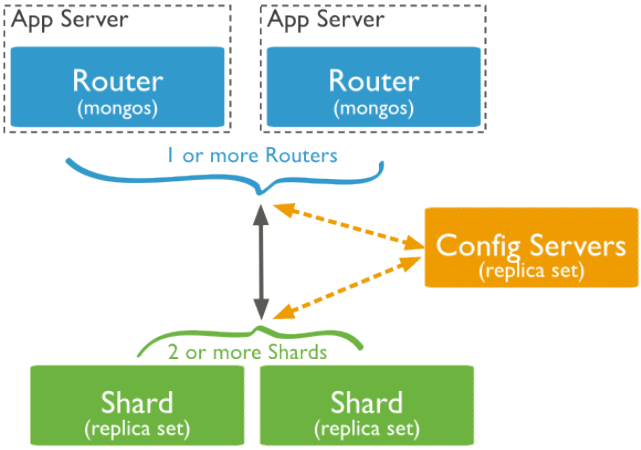


Fig. 2. Sharded MongoDB Architecture [11]

Figure 2 depicts a sharded MongoDB environment with two Mongos instances and two data Shards. The replica sets shown for both Config Servers and Shards may have any number of replicas within the set.

7.2. Config Servers

Config Servers stored metadata for sharded MongoDB clusters. This metadata includes information about the state and structure of the data and components of the sharded cluster. [12]

Config Servers also contain authentication information. For example, information about the keyfiles used for internal authentication between the nodes (described in detail in the Security subsection that follow) is stored in the Config Servers. [12]

In production deployments, it is recommended for Config Servers to be deployed in 3 member replica sets. [11] The rationale behind a 3 member set is discussed in more detail in the Replication subsection that follows.

In our deployment and benchmarking automation, the degree of replication in the Config Server Replica Set is controlled by the -c parameter. For example, specifying 1 for -c will create a Replica Set with 1 Config Servers, specifying 1 for -c will create a Replica Set with 2 Config Server, and so on.

7.3. Mongos Routers

Mongos is a query routing service used in sharded MongoDB configurations. Queries from applications go through Mongos, which locates the data in the sharded cluster. The Mongos instances accomplish this by reading and caching data from the Config Servers. [13]

For applications with high performance or availability requirements multiple Mongos instances may be ideal. In a high volume application, spreading the routing load over multiple Mongos instances can benefit performance. Additionally, multiple Mongos instances may increase availability in the case where a Mongos instance fails. [12]

In our deployment and benchmarking automation, the number of Mongos instances created is controlled by the -m parameter. For example, specifying 1 for -m will create 1 Mongos instance, specifying 2 for -m will create 2 Mongos instances, and so on.

7.4. Shards

Sharding, or distributing data across machines, is used by MongoDB to support large data sets and provide high throughput. [14]. Our deployment and benchmarking will test the performance of various numbers of shards, measuring the performance improvements associated with sharding in MongoDB.

Documents are distributed among the shards using a shard key. A sharded collection must have one, and only one, shard key which must have a supporting index. [14] Since the shard key is critical to performance and efficiency, particular care must be given to shard key selection. [15] In our performance testing a key was chosen that would distribute data relatively evenly across the shards, but was not used in retrieving the data as the more costly retrieval of not using an index provided a better test case.

In our deployment and benchmarking automation, Sharding is controlled by the -s parameter. For example, specifying 1 for -s will cause 1 shard to be created, specifying 2 will cause 2 shards to be created, and so on.

7.5. Replication

In databases, replication provides data redundancy leading to greater availability and fault tolerance. Replication in MongoDB is achieved via Replica Sets. [16] Replica Sets were implemented in our deployment for both Config Servers and Shards.

Two key benefits provided by replication are redundancy and fault tolerance.

- Redundancy - Each Replica in a Replica Set provides another copy of data. Higher redundancy means more nodes can be lost without data being lost.
- Fault Tolerance - Higher numbers of Replicas in a set increase fault tolerance, which leads to increased availability. As a general rule, a set will be able to tolerate faults while the majority of its nodes are still available

As shown in Figure 3, odd numbers of members in a replica set are a better choice for fault tolerance. For example, both a 3 and 4 replace set can only tolerate one member failing while maintaining availability. This is because a majority of the members must be available to maintain availability. In a 3 replica

Replica Members	Majority Needed	Fault Tolerance
2	1	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3
8	5	3

Fig. 3. Fault Tolerance by Replica Set Size [17]

set the majority is 2, so it can tolerate 1 member failing. In a 4 replica set, the majority is 3, so it can still only tolerate 1 member failing. Increases in fault tolerance only occur when the next odd numbered member of a replica set is added. [17]

For production systems, a standard deployment is a 3 Replica Set. [17]. A 3 replica set provides 3 copies of the data for redundancy and fault tolerance if 1 member of the set were to fail. In a situation where availability was of higher concern, a 5 replica set would provide 4 copies of the data for redundancy and fault tolerance if 2 members of the set were to fail.

In our automated deployment and benchmarking process, the degree of replication for Shards is controlled by the -r option. For example, specifying 1 for -r will create a replica set per shard with only 1 copy of data (essentially no replication, although technically we create a 1 member replica set), specifying 2 will cause a replica set of 2 to be created, and so on.

7.6. Security

There are two levels of security to consider in a sharded MongoDB deployment: internal and external authentication.

In our deployment the various MongoDB components (config servers, mongos instances, shards, and replicas) all reside on separate Virtual Machines. These machines must be able to communicate with each other. Two steps were necessary to enable this internal authentication. First, the ports (27017, 27018, 27019, 28017) used by MongoDB needed to be opened for communication. This was accomplished by adding appropriate security group rules to the clouds through Cloudmesh client. Second, MongoDB requires the internal authentication to be done by either keyfiles or x.509 certificates. [18] In our deployment, authentication is done by keyfiles. CONSIDERING ADDING X.509 SECURITY OPTION

For external authentication, the three users were created.

- *admin* - The user *admin* is created with the role of *userAdminAnyDatabase*. This user performs administrative functions such as creating the other users.
- *cluster_admin_user* - The user *cluster_admin_user* is created with the role *clusterAdmin*. This user performs sharding functions such as sharding the collection and checking its data distribution.
- *user1* - This is a standard user with readWrite permissions. This user performs the benchmarking tests and other functions not requiring administrative privileges.

8. DEPLOYMENT

Will make major modifications

The automated process will fully deploy a sharded MongoDB environment with the number of Config Servers, Mongos Instances, Shards, and degree of Replication specified as input parameters. It will also automatically created a sharded collection, run benchmarking tests agains it, and capture the output in CSV files as input for python visualization programs.

8.1. Deployment Process

The deployment consists of Ansible playbooks and Cloudmesh Client scripts and commands. It is run on the user's local Ubuntu 16.04 instance and executes commands both locally and on virtual machines in the cloud over ssh. A high level overview of the process is provided here. An inventory and overview of the code used to automate the deployment process is provided in Appendix B.

The automated deployment is run by executing the top level script, mongo_deploy.sh [19] [20] The script takes several parameters, described below. The script will fully deploy the environment as specified.

THIS WILL DRASTICALLY CHANGE BUT LEAVING AS PLACEHOLDER TO MAKE SURE EVERYTHING IS INCLUDED AFTER CONVERSION TO ANSIBLE

- -d - the Deployment Cloud (*chameleon* or *kilo*)
- -c - a number, the size of Config Server Replica Set
- -s - number of Shards
- -r - a number, size of Shard Replica Set

After validating the input parameters the deployment is run as follows.

Security

1. Add and upload the user's public key (this functionality is currently disabled to prevent throwing an error as it is assumed the user has already done this).
2. Add security rules to the user's default security profile port opening ports 27107, 27018, 27019, and 28017 for communication.
3. Upload the security group profile.
4. Create a new random keyfile (described in more detail in the security section) for each deployment.

Config Servers

1. Boot the appropriate number of Virtual Machines based on the -c parameter.
2. Check if there are available Floating IP Addresses in the pool. If not, create a new one.
3. Assign a Floating IP to each Virtual Machine and record it in a file (the Floating IP is needed for communication between the servers and ssh from the client to the cloud).
4. Modify /etc/hosts to fix a warning message generated when running as sudo
5. Install MongoDB software on all servers.

6. Transfer the mongo.key file to the Virtual Machine using Secure Copy (scp)
7. Transfer a configuration file for the Config Server to the Virtual Machine using Secure Copy (scp)
8. Over ssh, stop the running mongod process on each server and start a new mongod processes using the configuration file sent in the previous step.
9. Over ssh, on one Config Server, run an initiate command including the IP addresses of the others to start the Replica Set.

Mongos Instances

1. Boot the appropriate number of Virtual Machines based on the -m parameter.
2. Check if there are available Floating IP Addresses in the pool. If not, create a new one.
3. Assign a Floating IP to each Virtual Machine and record it in a file (the Floating IP is needed for communication between the servers and ssh from the client to the cloud).
4. Modify /etc/hosts to fix a warning message generated when running as sudo
5. Install MongoDB software on all servers.
6. Transfer the mongo.key file to the Virtual Machine using Secure Copy (scp)
7. Transfer the configuration file for the mongos instance to each Virtual Machine using scp. Prior to transferring, update the template configuration file to include the IP Address of the Configuration Server created earlier.
8. Over ssh, stop the running mongod process on each server and start a new mongos processes using the configuration file sent in the previous step.
9. Create several users (detailed in security section)

Shards

1. For each -r Replica Set, boot the appropriate number of Virtual Machines based on the -s parameter.
2. Check if there are available Floating IP Addresses in the pool. If not, create a new one.
3. Assign a Floating IP to each Virtual Machine and record it in a file (the Floating IP is needed for communication between the servers and ssh from the client to the cloud).
4. Modify /etc/hosts to fix a warning message generated when running as sudo
5. Install MongoDB software on all servers.
6. Transfer the mongo.key file to the Virtual Machine using Secure Copy (scp)
7. Transfer the configuration file for the Shard to each Virtual Machine using scp. Prior to transferring, update the template configuration file to include the correct replica set name.
8. Over ssh, stop the running mongod process on each server and start a new mongod processes using the configuration file sent in the previous step.
9. Over ssh, on one shard in the replica set, run an initiate command including the IP addresses of the other shards in the set to start the Replica Set.
10. On a mongos instance, run a command to add the shard replica set.

After deployment and benchmarking (described in detail in a later section) are complete, all Virtual Machines that have been created in the current deployment are deleted.

8.2. Deployment Timing

Will make major modifications

The configuration parameters and total deployment time is captured in a file for each deployment (benchmarking timings are later added to this file as well). Total run time for a few interesting configurations are shown in Figure 4.

Deployment A shows a simple deployment with only one of each component being created. This deployment may only be suitable for a development or test environment. Deployment A takes 517 seconds.

Deployment B shows a more complex deployment with production like replication factors for Config Servers and Shards and an additional Mongos instance. This deployment may be suitable for a production environment as it has greater fault tolerance and redundancy. Deployment B takes 2533 seconds.

Deployment C shows a deployment focused on high performance. It has a high number of shards, 9, but no fault tolerance or redundancy. The deployment may be suitable where performance needs are high and availability is less critical. Deployment C takes 1968 seconds.

More detailed run times for these deployments are shown in Appendix B, along with benchmarking results.

	Config Servers -c	Mongos -m	Shards -s	Replicas -r	Time in Seconds
A	1	1	1	1	*517*
B	3	2	3	3	*2533*
C	1	1	9	1	*1968*

Fig. 4. Deployment Times on Chameleon Cloud in Seconds

The total number of virtual machines is highly correlated with deployment time as booting the machines and installing the software, tasks that occur for all nodes, take the most time. he additional steps to configure Config Servers, Mongos Instances, Replicas, and Shards run in relatively similar times, so the specific type of component created has little impact on the deployment time. For example, holding all other deployment variables at 1, a deployment with 5 Config Servers took 1258 seconds, one with 5 Mongos Instances took 1253 seconds, one with 5 Shards took 1292 seconds, and one with a 5 Shard Replica set took 1277 seconds.

Figure 5 shows this empirically, as it takes a very similar time to launch configurations with the same total number of nodes, but extremely different mixed of Config Servers, Mongos Instances, Replicas, and Shards.

The total number of nodes in a deployment can be calculated by the following equation involving the parameters to the deployment script.

c + m + (s * r) = total nodes

Config Servers -c	Mongos -m	Shards -s	Replicas -r	Time in Seconds
5	1	1	1	*1258*
1	*5*	1	1	*1253*
1	1	*5*	1	*1292*
1	1	1	*5*	*1277*

Fig. 5. Deployment Times on Chameleon Cloud in Seconds

Note: Due to the floating IP issues in the Kilo environment, all deployment times shown are for Chameleon environment. Professor von Laszewski is aware of these infrastructure issues and they cannot be resolved by the project team. Fugang Wang is currently working on resolving them.

9. BENCHMARKING

After the sharded MongoDB instance has been fully deployed, a benchmarking process is run to assess performance of the configuration. This process has also been fully automated. It is automatically invoked after the deployment when running mongo_deploy.sh. An inventory of the code used to automate the benchmarking process is provided in Appendix B.

9.1. Data Set

Add Map/Reduce

The data set used in the benchmarking testing and analysis was Major League Baseball PITCHf/x data obtained by using the program Baseball on a Stick (BBOS). [21] BBOS is a python program created by *willkoky* on github which extracts data from mlb.com and loads it into a MySQL database. While it would be possible to convert this program to populate the MongoDB database directly, collecting all of the data is a time consuming process. Therefore, the data was captured locally to the default MySQL database and then extracted to a CSV file. This file contains 5,508,014 rows and 61 columns. It is 1,588,996,075 bytes in size uncompressed.

9.2. Methodology

Will make major modications

The primary benchmarking goal of the project was to assess the impact of sharding on performance in MongoDB. Since replication was also built into the deployment process, a secondary goal was to assess the impact of replica sets on performance. The benchmarking tests are design to assess performance of the following situations.

MAY ADD MAP REDUCE TEST SCENARIO

https://docs.mongodb.com/manual/core/map-reduce/ https://docs.mongodb.com/manual/core/map-reduce-sharded-collections/

1. Impact of Sharding Configuration on Reads
2. Impact of Sharding Configuration on Writes
3. Impact of Replication Configuration on Reads
4. Impact of Replication Configuration on Writes

To access the impact of different configurations on writes, we use MongoDB's mongoimport command. Mongoimport is a command line tool capable of loading JSON, CSV, or TSV files. [22] In this case, we load a CSV file to the pitches collections in the mlb database.

To assess the impact of different configurations on reads, we use MongoDB's find command. We read the data previously loaded by the mongoimport command to the pitches collection. The find command retrieves documents that meet a specified criteria. In this case, we search for pitches with a speed over 100 mph, a relatively rare event in baseball. To limit the information sent back over the network, we only return a count of these events. 3,632 is the count returned of 5,508,014 total documents. The column we search on does not have an index, as the goal is to test the impact of sharding on a long running query.

For each test, timing results were captured in file benchmark_datetime.csv. This file included the configuration the MongoDB configuration the test was run under (cloud, config server replication factor, mongos instances, number of shards, and shard replication factor) along with the run times of the find and mongoimport commands. After all tests were run, a shell script, combine_benchmark.sh was run to combine all files into one file, benchmark_combined.csv.

The graphical depictions of the test results show in the next section were created by running python programs to average the run times across the shard and replication configurations shown. For consistency, config server replication and mongos instances were both kept at 1 for all benchmarking tests. Additionally, replication was kept at 1 for sharding tests and sharding at 1 for replication tests. This methodologies allows us to isolate the variable we are assessing.

A compressed version of file has been placed in an Amazon Web Services S3 directory. The benchmarking script retrieves the file via wget, uncompresses it, and loads it to a collection named *pitches* in MongoDB using mongoimport before running the find command.

9.3. Benchmarking Process

Before the benchmarking process can be run, a sharded collection must be created. The script mongo_collection.sh creates and shards a collection for the data set to be loaded to. As with other steps in the process, this is automated and runs out of mongo_deploy.sh.

Create Sharded Collection

1. Ssh to a mongos instance. Authenticating as user1 via the mongo shell, create the pitches collection.
2. Ssh to a mongos instance. Authenticating as cluster_admin_user via the mongo shell, run a command to enable sharding in the mlb database the pitches table is located in.
3. Ssh to a mongos instance. Authenticating as cluster_admin_user via the mongo shell, run a command to shard the collection.

The shard key is set to pitchID. PitchID is a unique key to each pitch document. Selecting pitchID as the shard key should cause the data to be reasonably evenly distributed amound the shards. Data distribution will be analyzed in a subsequent section.

Once the sharded collection has been created, benchmarking test can be run. The script mongo_benchmark.sh runs the benchmarking process. It adds the benchmarking times for both find and mongoimport to the file created during the deployment containing the configuration parameters and deployment time.

Benchmarking

1. Ssh to a mongos instance. Using wget retrieve the pitches file from a Amazon Web Services (AWS) Simple Storage Service (S3) folder and uncompress the file.
2. Connect to a mongo shell on the mongos instance and run a mongoimport of the file to the sharded collection, capturing the run time.
3. Connect to a mongo shell on the mongos instance and run a find command against the sharded collection, capturing the run time.

9.4. Data Distribution

To explore how data was allocated among the shards, the getShardDistribution() command, which reports on how data and documents are distributed among shards [23], is run as part of the automated process after the data is populated in the collection by mongoimport. Figure 6 show the results of tests with 1 to 5 shards. The results clearly show the data is well distributed, although interestingly, in all cases there is some skew toward the final shard having the most data.

	1	2	3	4	5
1	100				
2	45.83	54.16			
3	31.17	31.05	37.76		
4	23.72	23.98	24.14	28.14	
5	19.27	18.88	18.86	18.89	24.08

Fig. 6. Data Distribution among Shards

9.5. Benchmarking Analysis

9.5.1. Impact of Sharding on Reads

Will make major modications

Figure 7 depicts the impact on performance of various numbers of shards on a find command in both Chameleon and Kilo Clouds. The sharp decline in run time as the number of shards is increased shows the positive effect of sharding on performance. For example, while the run time for 1 shard in both Chameleon and Kilo is over 120 seconds, the time drops to around 15 seconds for 6 shards. This is a significant gain in performance.

For small numbers of shards, performance gains are almost exact proportion to the number of shards. 2 shards yields 1/3 the run time of 1 shard. 3 shards yields 1/2 the run time of 1 shard. From 3 to 6 chards, we still see still see significant improvement, but incrementally less than for the first 3 shards. After 6 shards, we see only slight performance gains.

From the closeness of the Chameleon and Kilo lines we can see that performance in the two clouds is very similar for this find test. This is an interesting observation as for both deployment and mongoimport, performance was much better on

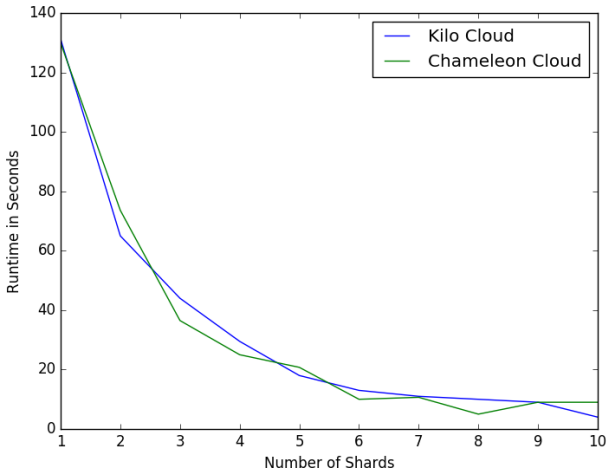


Fig. 7. Find Command - Sharding Test

Chameleon Cloud than Kilo. One difference from the mongoimport test is that much less data is being sent over the network. Network speeds could be a factor in this discrepency.

Figure 7 can be recreated by running the program benchmark_shards_find.py passing the file benchmark_combined.csv as a parameter. It plots the average run time for each configuration as shown using matplotlib.

9.5.2. Impact of Sharding on Writes

Will make major modications

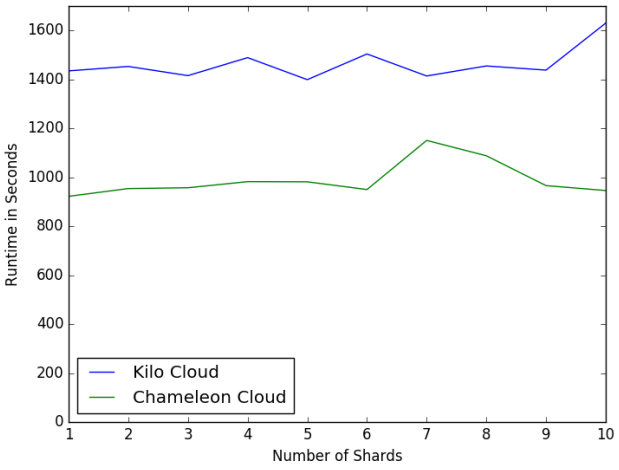


Fig. 8. Mongoimport Command - Sharding Test

Figure 8 depicts the impact on performance of various numbers of shards on a mongoimport command in both Chameleon and Kilo Clouds. For both clouds run time of the mongoimport command in our tests does not appear to be impacted by the number of shards. Since the same amount of data is written with more computing resources available when there are more shards, we might expect to see a performance gain. However, there are possible explanations for performance not improving. First, the mongoimport command may not write data in parallel. This is not indicated in the documentation, but it seems likely that it

reads the file serially. Second, resources on the server the data is written to may not be the bottleneck in the write process. Other resources like the network time are likely to be the bottleneck. Since we are always going over the network from the mongos instance to a data shard, regardless of the number of shards, a bottleneck in the network would impact all shard configurations equally.

While sharding did not benefit a single threaded mongoimport command, it is likely it would benefit heavy write operations, particularly coming through multiple mongos instances. In a non-sharded environment, this would lead to a heavy load on the single data shard. In a sharded environment, the load on each shard would drop as the number of shards increased.

While performance on Chameleon and Kilo was very similar for the find command, performance of the mongoimport command was significantly better on Chameleon than on Kilo. We see approximately 50% better performance on Chameleon Cloud than on Kilo. Although not formally plotted due to environmental issue in Kilo Cloud, similar results were observed for deployment times. Overall, Chameleon Cloud is a faster environment that FutureSystems Kilo Cloud.

Figure 9 can be recreated by running the program benchmark_shards_import.py passing the file benchmark_combined.csv as a parameter. It plots the average run time for each configuration as shown using matplotlib.

9.5.3. Impact of Replication on Reads

Will make major modications

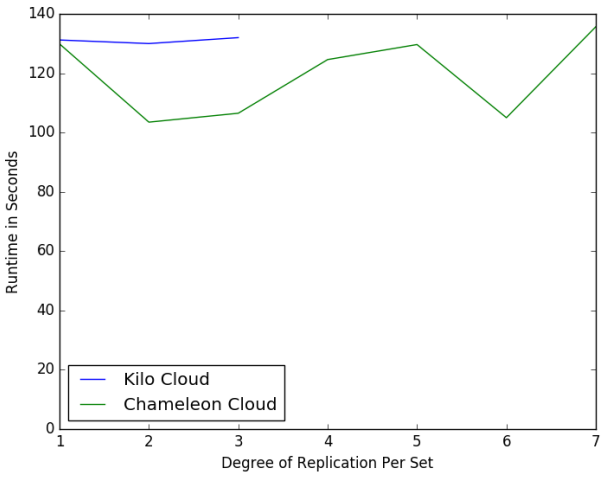


Fig. 9. Find Command - Replication Test

Figure 9 depicts the impact on performance of various numbers of replicas on a find command in both Chameleon and Kilo Clouds. (Note: Due to environmental issues with floating IPs in the India/Kilo environment, tests were only able to be run from 1, 2, and 3 replica configurations on Kilo cloud. This is an infrastructure problem beyond the control of the project team that Professor von Laszewski is aware of.) While it is clear that replication does not have the same performance impact on the find command that sharding does, it appears that there may be a slight performance penalty to high degrees of replication. Replica sets up to 3 did not show this penalty, but 4 and 7 replica sets both had increased run times. Without a larger sample size it cannot be determined if this is a real effect or random variation. We would not expect replication to have a significant performance degradation on the the find command since it only needs to read one copy of the data, but the increased communication necessary in a replica set may cause a small performance penalty in some cases.

Similarly to our sharding mongoimport test, performance on Chameleon was better than on Kilo for all test runs in the find replication test. The difference was similar to the 50% improvement noted earlier.

Figure 9 can be recreated by running the program benchmark_replicas_find.py passing the file benchmark_combined.csv as a parameter. It plots the average run time for each configuration as shown using matplotlib.

9.5.4. Impact of Replication on Writes

Will make major modications

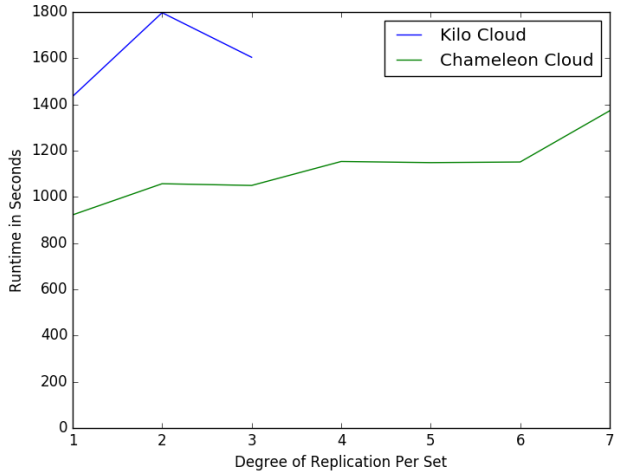


Fig. 10. Mongoimport Command - Replication Test

Figure 10 depicts the impact on performance of various numbers of replicas on a mongoimport command in both Chameleon and Kilo Clouds. (Note: Due to environmental issues with floating IPs in the India/Kilo environment, tests were only able to be run from 1, 2, and 3 replica configurations on Kilo cloud. This is an infrastructure problem beyond the control of the project team that Professor von Laszewski is aware of.) The test results show negative impact on the mongoimport command for increase levels of replication. On Chameleon, a replication factor of 2 leads to approximately a 20% performance penalty and a replication factor of 7 leads to 50% worse performance. Given that an extra copy of data is written with each increase in the replication factor, this performance hit is expected.

Similarly to our sharding mongoimport and find replication tests, performance on Chameleon was better than on Kilo for all test runs in the mongoimport replication test, again by a similar percentage.

Figure 10 can be recreated by running the program benchmark_shards_import.py passing the file benchmark_combined.csv as a parameter. It plots the average run time for each configuration as shown using matplotlib.

10. SUMMARY

Will make major modications

We have created, tested, and demonstrated a fully automated program to configure and deploy a sharded MongoDB cluster to three cloud environments. The cluster can be deployed with a selected number of Config Server Replicas, Mongos Routers, Shards, and Shard Replicas. An automated benchmarking process to show the impact of well distributed data across shards of a large data set has been created and run for various configurations to assess the impact of Sharding and Replication on performance. A key finding of these tests is that read performance, typically a high priority for noSQL data stores, increases significantly as shards are added. Testing also showed a predictable performance penalty is associated with replication.

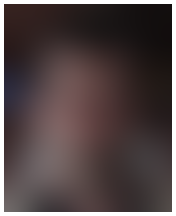
ACKNOWLEDGEMENTS

The authors thank X, Y, Z for technical support.

REFERENCES

[1] Wikipedia, "Openstack," web page, Nov. 2016, 11/9/2016. [Online]. Available: https://en.wikipedia.org/wiki/OpenStack
[2] OpenStack, "Openstack community q&a," web page, Nov. 2016. [Online]. Available: https://www.openstack.org/projects/openstack-faq/
[3] Chameleon, "About," web page, Nov. 2016. [Online]. Available: https://www.chameleoncloud.org/about/chameleon/
[4] FutureSystems, "About," web page, Sep. 2014. [Online]. Available: https://portal.futuresystems.org/about
[5] ——, "Staff," web page, Dec. 2016. [Online]. Available: https://portal.futuresystems.org/staff
[6] Chameleon, "Hardware description," web page, Nov. 2016. [Online]. Available: https://www.chameleoncloud.org/about/hardware-description/
[7] G. von Laszewski, "Hardware," web page, Jan. 2013. [Online]. Available: http://futuregrid.github.io/manual/hardware.html
[8] ——, "Cloudmesh client toolkit," web page, Sep. 2016. [Online]. Available: http://cloudmesh.github.io/client/
[9] Wikipedia, "Mongo," web page, Sep. 2016. [Online]. Available: https://en.wikipedia.org/wiki/MongoDB
[10] DB-Engines, "Bdb-engines ranking," web page, Nov. 2016. [Online]. Available: http://db-engines.com/en/ranking
[11] MongoDB, "Sharded cluster components," web page, Sep. 2016. [Online]. Available: https://docs.mongodb.com/v3.2/core/sharded-cluster-components/
[12] ——, "Config servers," web page, Nov. 2016. [Online]. Available: https://docs.mongodb.com/manual/core/sharded-cluster-config-servers/
[13] ——, "Mongos," web page, Nov. 2016. [Online]. Available: https://docs.mongodb.com/manual/reference/program/mongos/
[14] ——, "Sharding," web page, Sep. 2016. [Online]. Available: https://docs.mongodb.com/manual/sharding/
[15] ——, "Shard keys," web page, Sep. 2016. [Online]. Available: https://docs.mongodb.com/manual/core/sharding-shard-key/
[16] ——, "Replication," web page, Sep. 2016. [Online]. Available: https://docs.mongodb.com/manual/replication/
[17] ——, "Replica set deployment architectures," web page, Nov. 2016. [Online]. Available: https://docs.mongodb.com/v3.2/core/replica-set-architectures/
[18] ——, "Enable internal authentication," web page, Nov. 2016. [Online]. Available: https://docs.mongodb.com/v3.0/tutorial/enable-internal-authentication/
[19] ——, "Deploy a sharded cluster," web page, Sep. 2016. [Online]. Available: https://docs.mongodb.com/v3.2/tutorial/deploy-shard-cluster/
[20] M. Sanaulla, "Setting up sharded mongodb cluster in localhost," web page, Feb. 2015.
[21] willkoky, "Baseball on a stick," web page, Apr. 2016. [Online]. Available: https://sourceforge.net/projects/baseballonastic/
[22] MongoDB, "mongoimport," web page, Sep. 2016. [Online]. Available: https://docs.mongodb.com/manual/reference/program/mongoimport/
[23] ——, "db.collection.getsharddistribution()," web page, Nov. 2016. [Online]. Available: https://docs.mongodb.com/manual/reference/method/db.collection.getShardDistribution/

AUTHOR BIOGRAPHIES



Mark McCombe received his B.S. (Business Administration/Finance) and M.S. (Computer Information Systems) from Boston University. He is currently studying Data Science at Indiana University Bloomington.

A. WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Mark McCombe. Completed all work.

Proposal for Music Predictive Analysis Project based on Lyrics

LEONARD MWANGI[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: lmwangi@iu.com

project-01, March 27, 2017

Being certain that lyrics of your song will lead to the next greatest hit would boost confidence to a lot of amateur artists who are faced with fears of never making it thus never attempting to make good their creativity. With Machine Learning (ML) this can be a thing of the past, these artists would have the ability to let ML models determine the viability of their lyrics becoming the next hit based on history of other songs that have made it to top. Through training, the model can certainly determine the outcome of different songs which will be depicted in this project.

Keywords: Cloud, I524

https://github.com/lmundia/sp17-i524/tree/master/project/S17-IO-3013/report/report.pdf

1. INTRODUCTION

When faced with the decision to forward their song to a recording company, amateur artists find it daunting due to uncertainty of whether their song would be recorded and if it is if it will make them wealthy. Having ability to run the lyrics through a predictive analysis process that would determine the viability of the song making it would be a huge win and confidence booster to many artists. That prediction is achievable by use of machine learning and creating a model that takes already greatest his and trains it to determine what makes the song successful. This would be done by analyzing the lyrics, the locality and time of release.

In this project, we will utilize machine learning to help determine the viability of a song becoming the next greatest hit based on the lyrics, time of production, locality and the artist. The project will utilize the greatest hits of all time [1] to train a model which will then be used to analyze larger dataset of random songs [2] and provide an in-depth analysis of the next possible hit. The project will utilize a Hadoop cluster deployed on Chameleon Cloud using CloudMesh to accomplish this analysis.

The following components will be utilized to accomplish the project:

- Ansible
- Apache Mesos
- Apache Spark
- MongoDB
- Million Song Dataset
- Billboard charts
- Python Scripts

2. COMPONENTS ROLES

ANSIBLE

Will be used to install software packages and define roles to different nodes in the cluster.

APACHE MESOS

Will act as the scheduler for the environment.

APACHE SPARK

Due to Sparks ability to parallel process, we'll utilize it to process the dataset to achieve the required performance while providing in-depth analysis.

MONGODB

MongoDB will be used as the repository for the dataset.

3. MILLION SONGS DATASET

This is a freely-available community maintained dataset [**?**]. The dataset will be used by ML as the source of random songs that will be analyzed for results. This project will utilize a subset of the dataset due to time and size of our development environment.

BILLBOARD CHARTS

In conjunction with Million Songs Dataset, Billboard charts [**?**] will be used to determine the greatest hits of all time, which will be used to train the model on how to determine a great hit.

PYTHON SCRIPTS

Scripts will be used to train the model and determine the next greatest hit.

4. CONCLUSION

Ability for amateur artists, artists and record labels to quick determine the viability of a hit is paramount to their success and missed chances due to inexperience, fear of unknows, bad song or acting when time is not ripe can be costly. Machine learning has the ability to change these outcomes, a well-trained model can help determine with high accuracy where the song will end up.

REFERENCES

Deploying CouchDB Cluster

RIBKA RUFAEL[1,*,+]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: rrufael@umail.iu.edu*
[+] *HID: S17-IO-3016*

project-000, March 27, 2017

This project focuses on deployment of CouchDB Cluster using Ansible playbook on Ubuntu Chameleon Cloud VMs and benchmarking of the deployment.

Keywords: CouchDB

https://github.com/cloudmesh/classes/raw/master/docs/source/format/report/report.pdf

1. INTRODUCTION

CouchDB [1] is a no sql database management system under Apache. Data is stored as documents in CouchDB. In this project CouchDB cluster of one or more Chameleon cloud VMs is deployed using Ansible playbook and benchmarking is done to measure the time it took for deployement using a TBD benchmarking tool.

2. EXECUTION PLAN

This section depicts week by week execution plan for the project

2.1. Week 1

I was able to develop an initial Ansible script that will deploy CouchDB on Ubuntu 16.04 VM , upon successul installation the script will start CouchDB and then it will stop CouchDB on the remote VM. I booted Chameleon VM using Cloudmesh and then run Ansible playbook from my local machine. The tasks in my playbook run successfully.

2.2. Week 2

Run Ansible playbook to deploy CouchDB on Chameleon Cloud VM. Benchmark time it takes to deploy CouchDB on Chameleon Cloud VM

2.3. Week 3

Extend the Ansible script developed in Week 1 to deploy CouchDB into two Chameleon Cloud VMs.

2.4. Week 4

Run Ansible playbook to deploy CouchDB on two Chameleon Cloud VMs. Benchmark time it takes to deploy CouchDB on Chameleon Cloud VMs.

2.5. Week 5

Analysis of the benchmark results for deployment of CouchDB cluster. Document results in report and finalize report.

3. TECHNOLOGIES USED

- Ansible
- Cloudmesh
- Other technologies TBD

4. DEPLOYMENT

TBD

5. BENCHMARK RESULTS

TBD

6. CONCLUSION

TBD

ACKNOWLEDGEMENTS

TBD

REFERENCES

[1] Apache Software Foundation, "Technical Overview — Apache CouchDB 2.0 Documentation," Web Page, Mar. 2017, accessed: 2017-03-11. [Online]. Available: http://docs.couchdb.org/en/2.0.0/intro/overview.html

Analysis of USGS Earthquake Data

NANDITA SATHE[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: nsathe@iu.edu*

project-001, March 27, 2017

Geo-spatial data fits into definition of Big Data as it has all three 'V's viz. high-velocity, high-volume and high-variety. Big Data Analytics tools now allow us to analyze the huge volumes of geo-spatial data. Data of earthquakes that take place globally is a major part of crucial geo-spatial data. This application analyzes data related to earthquake which can be utilised in further research. US Geological Survey's (USGS) Earthquake Hazards Program monitor and report earthquakes, assess earthquake impacts and hazards, and research the causes and effects of earthquakes [1].

Keywords: I524, geospatial, MongoDB, D3.js, Apache Spark, Python, USGS, Ansible

https://github.com/nsathe/sp17-i524/blob/master/project/S17-IO-3017/report/report.pdf

1. INTRODUCTION

The USGS estimates that several million earthquakes occur in the world each year, although many go undetected because they occur in remote areas or have very small magnitudes [2]. Thus earthquakes pose significant risk globally to the mankind. USGS collects volumes of geospatial data pertaining to earthquakes and makes it available for analysis. This project intends to analyze this data. The application will be deployed on cloud. Deployment will be automated using Ansible.

2. TECHNOLOGIES USED

Technologies used for development and deployment of this project are listed below.

1. Cloudmesh - For connecting to different cloud environments.
2. Ansible -For deploying software and associated packages.
3. Python - Writing script for data analysis and data processing
4. Apache Spark - For data processing
5. Mongo-DB - For storing Geo-spatial data
6. D3.js - As a visualization tool

3. EXECUTION PLAN

This is how I intend to execute the project on week-by-week basis. Although my intention is to follow the plan deligently, it is possible that because of technical and other un-foreseen challenges deadlines may be pushed ahead.

1. **6 Mar 2017 - 12 Mar 2017** Create virtual machines on Chameleon cloud using Cloudmesh and submit the project proposal.
2. **13 Mar 2017 - 19 Mar 2017** Deploy Mongo DB to Chameleon cloud using Cloudmesh and develop initial Ansible playbook to install the required software packages.
3. **20 Mar 2017 - 26 Mar 2017** Write script in Python for downloading USGS data at run-time. Write Python and Spark scripts for data analysis and processing.
4. **27 Mar 2017 - 02 Apr 2017** Implement visualization using D3.js. Update Ansible playbook to install D3js package.
5. **03 Apr 2017 - 09 Apr 2017** Test on different cloud systems. Define quantitative benchmarks. Tentatively benchmarks will be for data insertion time and data processing time.
6. **10 Apr 2017 - 16 Apr 2017** Create deployable software package in Python.
7. **17 Apr 2017 - 23 Apr 2017** Update and finalize the Project Report

4. BENCHMARK

As mentioned in Section 3, benchmarks are tentatively for data insertion time and data processing time on different clouds.

5. ACKNOWLEDGEMENTS

This project is undertaken as part of the I524: Big Data and Open Source Software Projects course at Indiana University. The author would like to thank Prof. Gregor von Laszewski and his

associates from the School of Informatics and Computing for providing all the technical support and assistance.

6. LICENSING

TBD

7. CONCLUSION

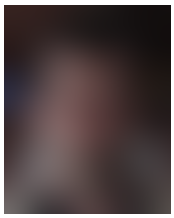
TBD

REFERENCES

[1] USGS, "Earthquake hazards program," Web Page. [Online]. Available: https://earthquake.usgs.gov/

[2] ——, "About us - program overview," Web Page. [Online]. Available: https://earthquake.usgs.gov/aboutus/

AUTHOR BIOGRAPHIES



Nandita Sathe is PMP certified project manager by profession. She will obtain MS in Data Sciences from Indiana University in May 2018. Her interests are in data analytics and machine learning.

8. WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Nandita Sathe. She completed all the work related to development of this application including research, testing and writing the project report.

S17-IO-3018/report/report.pdf not submitted

Twitter sentiment analysis of the Affordable Care Act in 2017

MICHAEL SMITH[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: mls35@iu.edu

project001,March 27, 2017

The mission of this project is to utilized technologies and cloud computing to perform a successful sentiment analysis through software deployment written in python. The software deployment will encompass data mining, analysis of big data, and comparison of this deployment across a variety of cloud computing services. The sentiment analysis will use the social media platform twitter and python libraries that effectively extract relevant data to the project goal.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IO-3019/report

1. INTRODUCTION

The current political climate of the United States is divided on many important issues. There is a disconnect between the motivations of the politicians of today and what is deemed important to the American people. The affordable care act(ACA) also known as Obamacare has been a target of the GOP, however it is uncertain if that sentiment is shared by most Americans. With a law that affects millions of Americans it is often difficult to gauge how the American people feel about the current healthcare law. Twitter is one of the biggest social medias on the internet with 67 million active users in the United States as of Q4 2016.[1] It is the goal of the project to gauge how Obamacare is viewed by twitter users who reside in the United States.

2. TECHNOLOGIES

Cloudmesh is an open source toolkit that allows the user to work across a variety clouds, virtual machines and clusters. This facilitates the ease of porting deployments to different clouds enabling the capability to benchmark cloud performance on a particular deployment. [2] The project was developed by Gregor von Laszekski and his colleagues at Indiana University. This will be the primary interface used to port the software to clouds such as chameleon cloud and futuresystems.

Ansible is open source software used for automation of provisioning of software deployments. This will be used when deploying on virtualization and cloud environments.

Chameleon cloud and futuresystems to be discussed here.

3. PYTHON

Early scripts have already been developed utilizing Twitters API and the python library tweepy to mine tweets that contain information relevant to Obamacare. Currently, the code authenticates with the twitter API, mines the tweets that contain a keyword of interest and finally a sentiment analysis which will rate a tweet by its polarity and subjectivity. For the sentiment analysis, the TextBlob library is used for its natural language processing(NLP) functionality. The final part of code outputs the tweet content and sentiment analysis into two columns into a csv file. This code will evolve to include data visualization through matplotlib and deeper analysis as the project moves closer to completion. Other python libraries will likely be added as well.

4. BENCHMARKS

Software will be deployed and benchmarked on various clouds.

5. LICENSING

TBD

6. WORK BREAKDOWN

Michael Smith is reponsible for all aspects of this project.

7. CONCLUSION

The software once finalized will be deployed across various clouds with the help of cloudmesh and ansible. Benchmark performance of the various clouds as well as analysis of the

twitter sentiment data will encompass most the final project report.

8. AUTHOR BIOGRAPHIES

Michael Smith is a senior quality control peptide chemist at Creosalus Inc. in Louisville, Kentucky. Michael possesses a MS in pharmaceutical sciences and a BS in Biology from the University of Kentucky. He will obtain his MS in Data Sciences program from Indiana University in May 2018. His current interests are python programming, data analytics, and spending time with his children.

REFERENCES

[1] Statista, "Number of monthly active twitter users in the united states." [Online]. Available: https://www.statista.com/statistics/274564/monthly-active-twitter-users-in-the-united-states/

[2] Cloudmesh, "Cloudmesh," Webpage. [Online]. Available: https://cloudmesh.github.io/

Detection of street signs in videos in a robot swarm

SUNANDA UNNI[1,*] **AND GREGOR VON LASZEWSKI**[1,**]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: suunni@indiana.edu*
[**] *Corresponding authors: laszewski@gmail.com*

project-1: Data analysis of Robot Swarm data, March 27, 2017

Extracting and identifying traffic signals from the videos captured by Robot swarms to help in recognizing the pattern and benchmarking the performance of the setup.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IO-3022/report/report.pdf

1. INTRODUCTION

For test purpose we created some mobile videos of traffic in a simulated traffic setup. All saved video files are uploaded on the Hadoop HDFS [1]. Batch processing is enabled on the input video files to search for key images, namely the red, green and yellow signals in the images using the OpenCV [2] library's Template matching functionality. Hadoop Map reduce [1] is used for processing and analysis of the images in the videos and getting a count of the how many red or green or yellow signals are encountered.

collectd [3] is used for benchmarking of the setup with Apache Hadoop using various sized data sets and number of nodes.

2. TECHNOLOGY USED

tables need a begin table end table

Technology Name	Purpose
Hadoop [1]	map reduce
OpenCV [2] Pattern matching in video	
ansible [4]	Automated deployment
collectd [3]	Collection of statistics of setup for benchmarking

3. PLAN

tables need a begin table end table

Week	Work Item	Status
week1	Ansible deployment script for Hadoop setup	planned
week2	Ansible deployment script for OpenCV setup	planned
week3	Creating sample videos	planned
week4	OpenCV template matching script	planned
week5	Deployment and test of basic setup	planned
week6	Ansible deployment of collectd	planned
week7	Performance measurement of setup and report creation	planned
week8	Exploring different setup	planned

4. DESIGN

TBD

5. DEPLOYMENT

TBD

6. BENCHMARKING

TBD

7. DISCUSSION

TBD

8. CONCLUSION

TBD

9. ACKNOWLEDGEMENT

REFERENCES

[1] Apache Software Foundation, "Apache hadoop," Web Page, 2014. [Online]. Available: http://hadoop.apache.org/

[2] itseez.com, “Opencv- open source computer vision,” Web Page, 2017. [Online]. Available: http://opencv.org/
[3] “collectd - the system statistics collection daemon,” Web Page. [Online]. Available: https://collectd.org/
[4] “Ansible, deploy apps. manage systems. crush complexity,” Web Page. [Online]. Available: https://www.ansible.com/

S17-IR-2002/report/report.pdf not submitted

On-line advertisement click prediction

SAHITI KORRAPATI[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: sakorrap@iu.edu, S17-IR-2013

March 13, 2017

This project aims at predicting the most suitable advertisements to be displayed on the web pages based on relevance by ranking each ad based on the liklihood of clicking. Data is obtained as CSV files from Kaggle Datasets and is stored in Hadoop Data File system(HDFS). In this project, various Bigdata tools and softwares are used to carry out the analytical computations efficiently. And Ansible is used to deploy all the necessary softwares on a cloud.

Keywords: Hadoop, Ad click Prediction, BigData, Ansible, Chameleon cloud

https://github.com/sakorrap/sp17-i524/tree/master/project/S17-IR-2013/report.pdf

1. INTRODUCTION

It has been analyzed that an average American spends about 23 hours per week surfing on-line [1]. This on-line user activity is being captured by companies to perform analysis for advertisements or recommendations or any other purpose. This has given rise to the field of "Web Analytics" and one such application is Ad Click prediction. Many measures are available to assess the ad performance. One such measure to assess the immediate ad response is click-through rate (CTR) of the advertisement [2] which is defined as the ratio of a number of clicks on an ad to the number of times the ad is shown, expressed as a percentage [3]. The user activity data that is used for prediction is enormous. To handle such large volumes of data Big data technologies come handy. The dataset that we are dealing with in this project is released by Outbrain which is 2 Billion page views and 16,900,000 clicks of 700 Million unique users, across 560 sites [4]. The data is anonymized and is in CSV file format. Paraquet compressing along with impala/drill to be decided to query and analyze the data compressed in files that are stored in HDFS. Programming language for analyzing the data is yet to be decided between JAVA and Python. Ansible is used to deploy the software and chameleon cloud for running virtual machines.

2. BACKGROUND

The dataset contains a sample of users' page views and clicks, as observed on multiple publisher sites in the United States between 14-June-2016 and 28-June-2016. Each viewed page or clicked recommendation is further accompanied by some semantic attributes of those documents [4]. The dataset contains numerous sets of content recommendations served to a specific user in a specific context. Each context (i.e. a set of recommendations) is given a display_id. In each such set, the user has clicked on at least one recommendation. Our task is to rank the recommendations in each group by decreasing predicted likelihood of being clicked [4].

2.1. Data fields description

Each user in the dataset is represented by a unique id (uuid). A person can view a document (document_id), which is simply a web page with content (e.g. a news article). On each document, a set of ads (ad_id) are displayed. Each ad belongs to a campaign (campaign_id) run by an advertiser (advertiser_id). Figure 1 shows the fields in our dataset. Metadata about the document is also provided, such as which entities are mentioned, a taxonomy of categories, the topics mentioned, and the publisher [4].

3. SETUP AND CONFIGURATION

TBD

4. WORK FLOW

TBD

5. EXPERIMENTS AND RESULTS

TBD

6. LICENSING

TBD

7. CONCLUSION

TBD

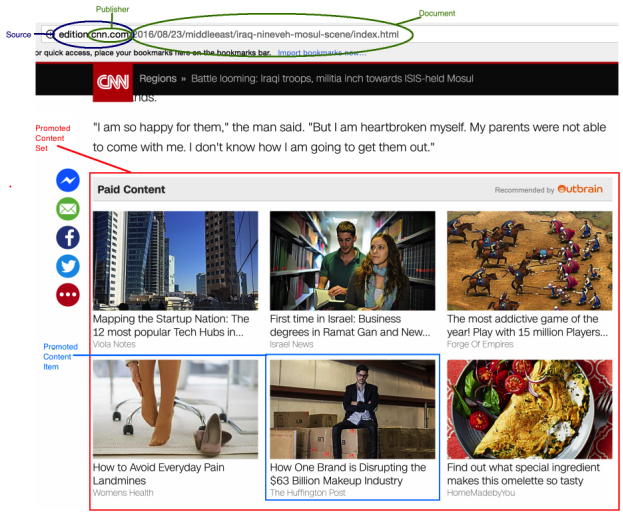


Fig. 1. Displaying Source, Publisher, Document, Promoted content set and items [4]

8. EXECUTION SUMMARY

1. Feb 23 - Mar 6, 2017 Exploring Python, Ansible and Chameleon cloud
2. Mar 10 - Mar 13, 2017 Exploring the datasets available and come up with the project proposal
3. Mar 14 - Mar 20, 2017 Decide on the architecture and workflow
4. Mar 21 - Mar 24, 2017 Configure and setup the workbench
5. Mar 25 - Mar 30, 2017 Prepare high level design
6. Mar 31 - Apr 6, 2017 Implementation and Testing
7. Apr 7 - Apr 11, 2017 Automate the deployment process using Ansible
8. Apr 12 - Apr 17, 2017 Optimizing and work on benchmarking
9. Apr 18 - Apr 22, 2017 Project review and completing any pending work
10. Apr 23 - Apr 24 Complete the report and commit the code

ACKNOWLEDGEMENTS

The authors thank Professor Gregor Von Laszewski and all the AIs of big data class for the guidance and technical support.

REFERENCES

[1] D. Mielach, "Americans spend 23 hours per week online, texting," Web-page, accessed mar-13-2017. [Online]. Available: http://www.businessnewsdaily.com/4718-weekly-online-social-media-time.html
[2] marketingterms.com, "Clickthrough rate definition," Web-page, accessed mar-13-2017. [Online]. Available: http://www.marketingterms.com/dictionary/clickthrough_rate/
[3] Wikipedia, "Click-through rate," Webpage, accessed Mar-13-2017. [Online]. Available: https://en.wikipedia.org/wiki/Click-through_rate
[4] Outbrain, "Data introduction," Webpage, accessed Mar-13-2017. [Online]. Available: https://www.kaggle.com/c/outbrain-click-prediction/data

AUTHOR BIOGRAPHIES

Sahiti Korrapati is pursuing her MSc in Data Science from Indiana University Bloomington

S17-IR-2016/report/report.pdf not submitted

Machine Learning for Customer churn prediction using big data analytics

YATIN SHARMA[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: yatins@indiana.edu*

project-001, March 27, 2017

This project involves use of machine learning algorithms to identify customers who are most likely to discontinue using the service or product.

Keywords: Prediction, Bigdata, Apache Spark, MLlib, Hadoop, Analytics

https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IR-2034/report/report.pdf

CONTENTS

1. INTRODUCTION

We will use Apache Spark[1] machine learning library for fitting a predictive model on a massive dataset. Detailed analysis and modeling will be carried out in Python Programming language.

2. EXECUTION SUMMARY

The tentative schedule for this project has been outlined below:

1. March 13-March 19, 2017: Create virtual machines on Chameleon, FutureSystems and Jetstream clouds
2. March 13-March 19, 2017: Deploy Hadoop cluster to the clouds and install the required software packages to the clusters and also finalize data.
3. March 20-March 26, 2017: Data Preprocessing and applying transformation to extract features from the data.
4. March 27-April 09, 2017: Use MLlib to train and evaluate various machine learning algorithms and choose best based on various performance metrics.
5. April 10 - April 16, 2017: Create deployable software packages in Python.
6. April 17-April 23, 2017: Complete Project Report.

3. WORKFLOW

The project will make use of the following four components.

1. Apache Spark
2. Hadoop
3. Spark MLlib

4. DEPLOYMENT

We will deploy our application using Ansible[2] playbook. Deployment of Master/slave nodes will be done hadoop/spark distributed cluster environment. Different cloud systems that will be used in the project include Chameleon,FutureSystems and JetStream.

5. BENCHMARKING

Performance of the Hadoop/Spark clusters deployed on different clouds will compared for benchmarking.

6. CONCLUSION

TBD

7. ACKNOWLEDGEMENT

TBD

REFERENCES

[1] A. S. Foundation, “Overview - spark 2.1.0 documentation,” Web Page, accessed: 03-12-2017. [Online]. Available: http://spark.apache.org/docs/latest/index.html

[2] “Ansible Documentation,” Web Page. [Online]. Available: http://docs.ansible.com/ansible/index.html

S17-IR-2039/report/report.pdf not submitted

Machine Learning for Customer churn prediction using big data analytics

DIKSHA YADAV[1,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: yadavd@umail.iu.edu*

project-001, March 27, 2017

This project involves use of machine learning algorithms to identify customers who are most likely to discontinue using the service or product.

Keywords: Prediction, Bigdata, Apache Spark, MLlib, Hadoop, Analytics

https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IR-2044/report/report.pdf

CONTENTS

1. INTRODUCTION

We will use Apache Spark[1] machine learning library for fitting a predictive model on a massive dataset. Detailed analysis and modeling will be carried out in Python Programming language.

2. EXECUTION SUMMARY

The tentative schedule for this project has been outlined below:

1. March 13-March 19, 2017: Create virtual machines on Chameleon, FutureSystems and Jetstream clouds
2. March 13-March 19, 2017: Deploy Hadoop cluster to the clouds and install the required software packages to the clusters and also finalize data.
3. March 20-March 26, 2017: Data Preprocessing and applying transformation to extract features from the data.
4. March 27-April 09, 2017: Use MLlib to train and evaluate various machine learning algorithms and choose best based on various performance metrics.
5. April 10 - April 16, 2017: Create deployable software packages in Python.
6. April 17-April 23, 2017: Complete Project Report.

3. WORKFLOW

The project will make use of the following four components.

1. Apache Spark
2. Hadoop
3. Spark MLlib

4. DEPLOYMENT

We will deploy our application using Ansible[2] playbook. Deployment of Master/slave nodes will be done hadoop/spark distributed cluster environment. Different cloud systems that will be used in the project include Chameleon,FutureSystems and JetStream.

5. BENCHMARKING

Performance of the Hadoop/Spark clusters deployed on different clouds will compared for benchmarking.

6. CONCLUSION

TBD

7. ACKNOWLEDGEMENT

TBD

REFERENCES

[1] A. S. Foundation, “Overview - spark 2.1.0 documentation,” Web Page, accessed: 03-12-2017. [Online]. Available: http://spark.apache.org/docs/latest/index.html

[2] “Ansible Documentation,” Web Page. [Online]. Available: http://docs.ansible.com/ansible/index.html

Real-time Visualization of Happiness Quotient across English regions based on Twitter data

SOWMYA RAVI[1,*], SRIRAM SITHARAMAN[2], AND SHAHIDHYA RAMACHANDRAN[3]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[2] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[3] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: sowravi@iu.edu, srirsith@iu.edu, shahrama@iu.edu

project-001, March 27, 2017

This project involves development of a real-time system which streams live data from twitter to visualize the "Happiness index" across the English-speaking regions in the world. Live data from twitter is injected into the system using streaming API in spark. All possible tweets are taken into consideration for analyzing the overall happiness level of people tweeting from different locations. Suitable classifier will be built to identify if the tweet is positively biased. The results of the Language processing algorithm will be visualized in real-time using d3.js.

Keywords: Real-time streaming, data visualization, Twitter, Natural Language Processing

https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IR-P001/report/report.pdf

CONTENTS

1. INTRODUCTION

In 1969, Drs. Jerry Boucher and Charles E. Osgood, psychologists at the University of Illinois, proposed the 'Pollyanna Hypothesis' which asserts that "there is a universal human tendency to use evaluatively positive words more frequently and diversely than evaluatively negative words in communicating" [1]. Such theories were hard to validate due to the absence of significant data and the lack of generality. With Social media turning into the primary platform where people express on a day-to-day basis these claims can be analysed by sampling a portion of the data. Twitter generates nearly 200,000 tweets in less than a minute. Big data technologies prove to be particularly useful in in storing, processing and analysing such large data. It is also possible to setup real-time systems that can output results with a latency of very few seconds.

2. EXECUTION SUMMARY

The approximate schedule for completion of this project has been outlines in the section below:

1. Mar 6 - Mar 12, 2017 Create virtual machines on Chameleon, FutureSystems and Jetstream clouds using Cloudmesh and submit the project proposal.
2. Mar 13-Mar 19, 2017 Deploy Hadoop cluster to the clouds using Cloudmesh and create Ansible playbook to install the required software packages (Cassandra,D3.js,Kafka etc.) to the clusters and to upload the twitter data.
3. Mar 20-Mar 26, 2017 Pre-processing of the tweets to create required features for using in the Natural Language Processing algorithm. Building a language model to estimate the Happiness quotient
4. Mar 27-Apr 02, 2017 Develop an interactive visualization of the analysed data in D3.js
5. Apr 03-Apr 09, 2017 Continuing with the D3.Js visualization and connecting with streaming data from twitter to convert it into a live dashboard.
6. Apr 10 - Apr 16, 2017 Create software package that can be readily deployed in Python

7. Apr 17-Apr 23, 2017 Complete the partially done Project Report

3. WORKFLOW

The project will make use of the following four components.

1. Apache Spark
2. Apache Kafka
3. Apache Cassandra
4. D3.js

The Architecture of the system is shown in Fig.1

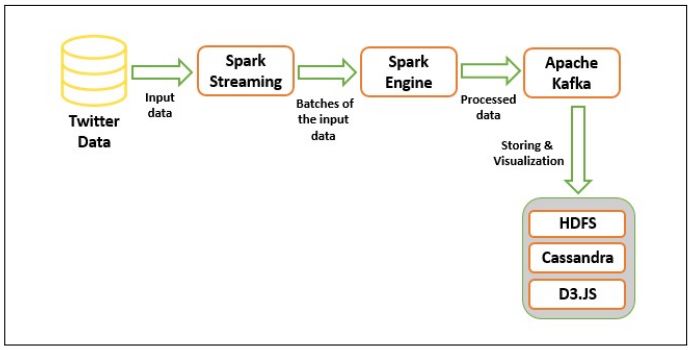


Fig. 1. Architecture

3.1. Phase 1: Streaming Twitter data using Apache Spark

Spark is a high speed in-memory data engine which is specialized to perform tasks such as streaming or requiring repeated access to datasets. The objective is to obtain the happiness index of tweets from different English speaking countries around the world. Thus it will require a fairly large amount of tweets collected over a time frame(TBD). Spark's framework provides the facility to work with a variety of data formats including text. Spark streaming which is the extension of the core spark API aids in the streaming process and delivers it to the core engine. The data is then passed to Apache Kafka which helps in pipeline processing

3.2. Phase 2: Kafka

Kafka, a queueing system serves as an ingestion backbone to Apache spark. It is a super-fast, low-latency, distributed and partitioned stream processing service. Kafka being highly reliable and scalable, is perfect for integrating the huge stream of twitter data to a data sink.

3.3. Phase 3: Apache Cassandra

Cassandra was chosen as the database because of it's high scalability and reliability. Cassandra used along with spark streaming and kafka forms an excellent base for real time analytics. Cassandra being a NoSQL database is well suited to store unstructured textual data. A feature that makes Cassandra stand out is that it is a column oriented database which makes it horizontally scalable too.

3.4. Phase 4: D3.js

Real time visualization of the processed data streamed from Kafka message queuing service would be created to view the results from the analytics performed on the twitter data.

4. CONCLUSION

Put in some conclusion based on what you have researched

Acknowledgement Put in the information for this class and who may sponsor you. Examples will be given later

REFERENCES

[1] J. Boucher and C. E. Osgood, "The pollyanna hypothesis," *Journal of Verbal Learning and Verbal Behavior*, vol. 8, no. 1, pp. 1 – 8, 1969. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022537169800022

Flight Price Prediction

HARSHIT KRISHNAKUMAR[1,*] AND KARTHIK ANBAZHAGAN[2]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[2] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: harkrish@iu.edu, kartanba@iu.edu

project-001, March 27, 2017

This project aims at tracking live flight status and flight pricing in the US. Live flight data streams are obtained using Python APIs and stored in Big Data Hadoop Distributed File Systems. This paper explores the use of Apache Hive to store data streams and analyse the data. The analyses will be presented in real time using d3.js.

Keywords: big data, apache hive

https://github.com/cloudmesh/sp17-i524/project/S17-IR-P002/report/report.pdf

CONTENTS

INTRODUCTION

Air travel is getting increasingly popular with the airlines providing cheaper fares and better services. More often, customers tend to look for flights in the last minute, which is exploited by third party vendors who look to gain more profits in the rush hour. Skyscanner is a travel fare aggregator website and travel metasearch engine which helps users find the lowest rates from multiple travel sites, as well as instant comparisons for hotels and car hire removing the need for customers to search across different airlines for prices [1].

A metasearch engine (or aggregator) is a search tool that uses another search engine's data to produce their own results from the Internet [2]. Metasearch [3] engines take input from a user and simultaneously send out queries to third party search engines for results. Sufficient data is gathered, formatted by their ranks and presented to the users. The Skyscanner Live Pricing allows developers to access live pricing information on prices for different flights, by making requests to the Live Pricing API.

In this project, we would be querying the Skyscanner Live Pricing API using Apache HIVE and deploying the data on cloud (1-TBD & 2-TBD). Cloudmesh would be used for cloud management and the software stack deployment would be done through Ansible. We would benchmark performance of our analysis across mutilple clouds. We would be presenting a real-time visualization of the cheapest air fare and the most likely travel destination analysis in D3.js.

WORKFLOW

The project will make use of Python APIs to retrieve live flight prices information from Skyscanner and dump it in Apache HIVE database [4]. SQL Analyses are performed on this data and the results of analyses are stored in HIVE and presented in an interactive dashboard or website. The dashboard will take the onward and return journey locations, and the date of travel as inputs from users and show different price ranges for different dates commencing from the next available flight, for a period of three months. This aims to provide the users an idea as to when is the safe time to book flight tickets and beyond which date will the prices shoot up.

EXECUTION SUMMARY

The schedule for completion of this project has been outlined below:

1. Mar 06-Mar 12, 2017 Creating virtual machines on Chameleon cloud using Cloudmesh and coming up with a project proposal

2. Mar 13-Mar 19, 2017 using cloudmesh to set up Hadoop clusters and installing the required software packages

3. Mar 20-Mar 26, 2017 Fetching the data from Skyscanner API and adding it to our HIVE database

4. Mar 27-Apr 02, 2017 Running few data mining/time series models to predict the ticket prices

5. Apr 03-Apr 09, 2017 Review the work done and find out scopes for improvement and creating a benchmark report
6. Apr 10-Apr 16, 2017 Presenting the work in D3.Js in real-time as a visualization of the analysis
7. Apr 17-Apr 23, 2017 Complete the Project Report

ACKNOWLEDGEMENTS

The author thanks Professor Gregor Von Lazewski for providing us with the guidance and topics for the Project. The author also thanks the AIs of Big Data Class for providing the technical support.

REFERENCES

[1] B. J. Jansen, A. Spink, and C. Ciamacca, "An analysis of travel information searching on the we," Pennsylvania State University, Paper 10(2), 101-118, 2018, accessed: 2017-3-14. [Online]. Available: https://faculty.ist.psu.edu/jjansen/academic/jansen_travel_searching.pdf

[2] S. Berger, *Sandy Berger's Great Age Guide to Online Travel*, 1st ed. Greenwich, CT, USA: Que Publishing, 2005. [Online]. Available: https://www.amazon.com/Great-Guide-Internet-Sandy-Berger/dp/0789734427

[3] E. J. Glover, S. Lawrence1, W. P. Birmingham, and C. L. Giles, "Architecture of a metasearch engine that supports user information needsarchitecture of a metasearch engine that supports user information needs," in *Eigth International Conference on Information Knowledge Management*. ACM New York, NY, USA, 1999, pp. 210–216. [Online]. Available: http://www.researchgate.net/publication/2596239_Architecture_of_a_Metasearch_Engine_that_Supports_User_Information_Needs/file/d912f5131046ecac21.pdf

[4] L. Leverenz, "Gettingstarted with hive, apache software foundation," Web Page, Sep. 2016. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/GettingStarted

AUTHOR BIOGRAPHIES

Harshit Krishnakumar is pursuing his MSc in Data Science from Indiana University Bloomington
Karthik Anbazhagan is pursuing his MSc in Data Science from Indiana University Bloomington

Detecting Street Signs in Videos in a Robot Swarm

RAHUL RAGHATATE[1,*] AND SNEHAL CHEMBURKAR[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: rraghata@iu.edu, snehchem@iu.edu

S17-IR-P003, March 29, 2017

The aim of this project is to deploy a software package to detect different street signs in a video stream. This will be a scalable system over Hadoop based cloud ecosystem to incorporate multiple video feeds and parallel real-time processing of the feeds. A comparative benchmark will be developed based on the performance of package on multiple cloud systems.

Keywords: Street Signs, Video Streams, OpenCV, Spark, Cloud

https://github.com/cloudmesh/sp17-i524/raw/master/project/S17-IR-P003/report/report.pdf

INTRODUCTION

Detecting objects in images has always been keen area of interest in the field of computer vision. There are many applications developed based on this simple idea like auto tagging pictures (e.g. Facebook, Phototime), counting the number of people in a street(e.g. Placemeter), classifying pictures, detecting vehicles, etc.On the similar grounds, we are building a software package which can be deployed easily on cloud infrastructure and establish a platform to detect different street signs in a video stream. A benchmark will be developed based on performance of this software on different cloud systems. The database of street signs will be restricted to US street signs. The video streams used for this project are simulated or captured using mobile camera.

REQUIREMENT ANALYSIS

We are using following technologies for complete project development and deployment:

1. Cloudmesh - For connecting to different cloud environments.

2. Ansible -For deploying software from master node on virtual slave nodes.

3. Python - Writing script for data analysis and data processing in Spark [1] engine over Hadoop.

4. Hadoop -Required for uploading our data set of images on distributed data storage platform as well as for video streams and its processing in Spark Stream. Using pre-build Hadoop and Spark in Cloudmesh so as to focus on video data distribution and analysis and perform optimization testing on cluster.

5. OpenCV [2] -Perform video analysis for street sign detection using open source computer vision libraries of video/image analysis algorithms.The OpenCV library provides several features to manipulate images(apply filters, transformation), detect and recognize objects in images.

METHODOLOGY

1. Data gathering for street signs and video streams

2. Deploy Hadoop clusters on cloud using Cloudmesh

3. Develop Ansible script to install OpenCV on cloud

4. Build a model to detect or track street signs using OpenCV. We plan on implementing two programs-

   - Read an image and run the Haar cascade classifier to detect the signs in the image and
   - use the video stream and detect signs in real time.

5. To detect street signs, we will be using Haar based cascade classifier which detect objects in an image. As detecting signs and categorizing them are two different problems and use two different approaches. Hence, we will benchmark detection first and will work on categorization as future development.

6. Test the performance of software package on 3 different clouds or on the same cluster with multiple nodes.

7. Create benchmarks based on the above results

EXECUTION SUMMARY

This section specifies the week by week timeline for project completion.

1. Mar 6 - Mar 12, 2017 Create virtual machines on Chameleon cloud using Cloudmesh and submit the project proposal.
2. Mar 13-Mar 19, 2017 Deploy Hadoop cluster to Chameleon cloud using Cloudmesh and develop Ansible playbook to install the required software packages to the clusters (OpenCV, etc)
3. Mar 20-Mar 26, 2017 Train data to detect or track street signs using OpenCV. Develop Ansible playbook to setup database and connectivity among multiple nodes.
4. Mar 27-Apr 02, 2017 Capture the results of street sign tracking in video streams.
5. Apr 03-Apr 09, 2017 Test on different cloud systems and define benchmarks.
6. Apr 10 - Apr 16, 2017 Create deployable software package in Python.
7. Apr 17-Apr 23, 2017 Write Project Report

USE CASES

1. Street Sign Detection for autonomous vehicles.
2. Analysis of traffic signs in Google Street View to estimate all signs ahead hence, useful in ambulance , fire brigade services, simplest path finder etc.

BENCHMARK

Benchmarks will be created based on the performance of the software in different cloud environments. The initial analysis will be done on a single short video stream and then on video streams distributed across 2 or 3 nodes. The different cloud systems used for the purpose of benchmarking are Chameleon, FutureSystems and JetStream.

ACKNOWLEDGEMENTS

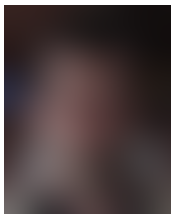
This project is undertaken as part of the I524: Big Data and Open Source Software Projects coursework at Indiana University. We would like to thank our Prof. Gregor von Laszewski, Prof. Gregory Fox and the Associate Instructors for their help and support

REFERENCES

[1] A. S. Foundation, “Overview - spark 2.1.0 documentation,” Web Page, accessed: 03-12-2017. [Online]. Available: http://spark.apache.org/docs/latest/index.html

[2] “Home - opencv/opencv wiki,” Code Repository, accessed: 03-12-2017. [Online]. Available: https://github.com/opencv/opencv/wiki

AUTHOR BIOGRAPHIES



Rahul Raghatate will receive his Masters (Data Science) in 2018 from The Indiana Univeristy Bloomington. His research interests include Big Data and Machine Learning.

Snehal Chemburkar will receive her Masters (Data Science) in 2018 from Indiana University Bloomington. Her research interests also include Big Data and Machine Learning.

WORK BREAKDOWN

Will be updated in later phases of project.

FOLLOWING TOPICS ARE YET TO BE INCLUDED

2.2. Shell Access If applicable comment on how the tool can be used from the command line

3. Licensing Often tools may have different versions, some fre, some for pay. Comment on this. For example while a tool may offer a comercial version this version may be too costly for others. Identify especially the difference between features for free vs commercial tools.

Sometimes you may need to introduce this also in the introduction as there may be a big difference and without the knowledge you do not provide the user an adequate introduction.

4. Ecosystem Some technologies have a large ecosystem developed around them with extensions plugins and other useful tools. Identify if they exists and comment on what they can achieve

provide potentially a mindmap or a figure illustrating how the technology fits in with other technologies if applicable.

5. Educational material Put information here how someone would find out more about the technology. Use important material and do not list hundreds of web pages, be selective.

Predicting Readmission of Diabetic patients

KUMAR SATYAM[1,*], PIYUSH SHINDE[1,**], AND SRIKANTH RAMANAM[1,***]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: ksatyam@indiana.edu
[**] Corresponding authors: pshinde@iu.edu
[***] Corresponding authors: srikrama@iu.edu

project-000, March 27, 2017

We are trying to predict whether a diabetic patient will be readmitted to the hospital, using several features representing patient and hospital outcomes. We will use Hadoop/Spark distributed architecture on multiple clouds as the core infrastructure and machine learning classification algorithms for data analysis.

Keywords: Hadoop, Spark, Ansible, Python

https://github.com/cloudmesh/classes/blob/master/project/S17-IR-P004/report/report.pdf

CONTENTS

1. INTRODUCTION

We will use Hadoop to split the dataset and transfer the data chunks to different data nodes. We will use Ansible to install pre-requisite softwares and push configurations on different machines. The data chunks would then be analyzed using machine learning techniques and the results would be aggregated predicting whether a patient would be readmitted or not. This information would help hospitals to be better prepared for readmitting patients.

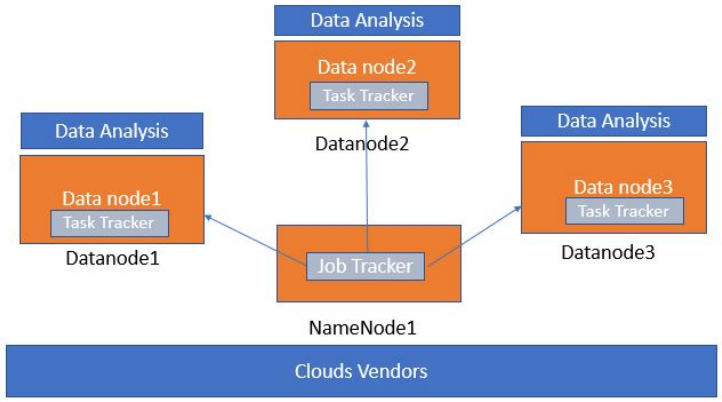


Fig. 1. Deployment Architecture

2. TIMELINE

Week	Target
1	Finalizing Technologies, Data Cleansing
2	Hadoop/Spark Deployment on Chameleon Cloud
3	Troubleshooting
4	Data Analysis
5	Deployment on other cloud using Ansible
6	Benchmarking
7	Report Preparation

3. TECHNOLOGIES

Technology	*Usage*
Hadoop[1]/**Spark** [2]	Distributed Data Storage
Python[3]/**Java**[4]/**Scala**[5]	Development
Ansible [6]	Application Deployment & Configuration Management
TBD	Benchmarking
LaTex [7]	Document Preparation

4. DEPLOYMENT

We will deploy a master & multiple slave nodes in the Hadoop/Spark distributed cluster environment.

We will use **Ansible** as an automated application and configuration deployment tool. This will enable us to install softwares and push configurations simultaneously from master node to the respective target nodes.

5. BENCHMARKING

We will assess the performance of the Hadoop/Spark clusters deployed on different clouds. The parameters for benchmarking would be memory usage, storage size and IO throughput.

6. RESULTS

Results of data analysis and benchmarking will be showcased in this section.

7. CONCLUSION

Using the 130-US hospitals dataset [8] for years 1999-2008, we should be able to analyze factors pertaining to readmission of patients with diabetes.

8. ACKNOWLEDGMENTS

This project was a part of the Big Data Software and Projects (INFO-I524) course. We would like to thank Professor Gregor von Laszewski and the associate instructors for their help and support during the course.

REFERENCES

[1] "Welcome to Apache™ Hadoop®!" Web Page, accessed: 2017-03-12. [Online]. Available: http://hadoop.apache.org/

[2] "Apache Spark: Lightning-fast cluster computing," Web Page, accessed: 2017-03-12. [Online]. Available: http://spark.apache.org/

[3] "python," Web Page, accessed: 2017-03-12. [Online]. Available: https://www.python.org/

[4] "java," Web Page, accessed: 2017-03-12. [Online]. Available: https://www.java.com/en/

[5] "Scala," Web Page, accessed: 2017-03-12. [Online]. Available: https://www.scala-lang.org/

[6] "ANSIBLE," Web Page, accessed: 2017-03-12. [Online]. Available: https://www.ansible.com/

[7] "The LATEX Project," Web Page, accessed: 2017-03-12. [Online]. Available: https://www.latex-project.org/

[8] "Diabetes 130-US hospitals for years 1999-2008 Data Set," Web Page, accessed: 2017-03-12. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008#

Analysis Of People Relationship Using Word2Vec on Wiki Data

ABHISHEK GUPTA[1,*] AND AVADHOOT AGASTI[1,**]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: abhigupt@iu.edu
[**] Corresponding authors: aagasti@iu.edu

project-1: Data mining for a wiki url , March 27, 2017

Given a wiki URL of a person, find out his details like School, Spouse, Coaches, language, alma-meter etc Typically, the wiki page has all this information available but in the free form text. We need to converting it into structured data format so that it can help us analyze the people, from the networks etc We can create a network by navigating the people mentioned in the wiki page.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IR-P005/report/report.pdf

CONTENTS

1. INTRODUCTION

Use spark [1] to load the wiki data and create word vectors. Train it using spark ML [2] and then use the model for analytics and prediction. The training set will use Word2Vec. Word2vec [3] is a group of related models that are used to produce word embeddings. Word2Vec is used to analyze the linguistic context of the words. In this project, we created Word2vec model using Wikipedia data. Our focus is people and organization names occurring in the Wikipedia data and to see if Word2vec can be used to understand relationship between people. Typically Wikipedia page for people and celebraties contain the entire family and friends, colleagues information. Our idea is to use Word2vec to see if using a smaller training set of known relationships whether we can derive similar relationship for anyone who has presence on Wikipedia. This mechanism can be then used to convert the data hidden in textual format to more structured data.

Technology Name	Purpose
spark [1]	data analysis
sparkML [2]	machine learning
python [1]	development
ansible [4]	automated deployment
collectd [5]	statistics collection for benchmarking

2. PLAN

Following table gives a breakdown of tasks in order to complete the project. Assuming week1 starts after submission of the proposal. These work items are high level breakdown on the tasks and may changes if needed.

Week	Work Item	Status
week1	Basic POC of Word2Vec using Python	planned
week2	Scripts to download Wiki data	planned
week3	Word2Vec Spark program	planned
week4	Training and measuring accuracy	planned
week5	Ansible Deployment script for Spark	planned
week6	Deployment and test on 2 clouds	planned
week7	Performance measurement	planned
week8	Report Creation(parallel)	planned

3. DESIGN

TBD

4. DEPLOYMENT

Solution will be deployed using Ansible [4] playbook. Automated deployment should happen on two or more nodes cluster. Deployment script should install all necessary software along with the project code to the cluster nodes.

5. BENCHMARKING

Solution will use collectd [5] to collect statistics. Once the solution is deployed to the cluster. We should benchmark parameters like

- cpu
- memory
- throughput reads/writes

Benchmarking will be done for one or more cloud providers. The deployment scripts should be agnostic to the cloud provider.

6. DISCUSSION

TBD

7. CONCLUSION

Using this wiki analysis we should be able to build a network based on wiki data.

8. ACKNOWLEDGEMENT

We acknowledge our professor Gregor von Laszewski and all associate instructors for helping us and guiding us throughout this project.

9. APPENDICES

TBD

REFERENCES

[1] "Spark Python API (PySpark)," Web Page, accessed: 2017-02-26. [Online]. Available: https://spark.apache.org/docs/0.9.1/python-programming-guide.html

[2] "Spark ml programming guide," Web Page, accessed: 2017-02-26. [Online]. Available: https://spark.apache.org/docs/1.2.2/ml-guide.html

[3] "Word2Vec, learning vector representation of words," Web Page, accessed: 2017-02-26. [Online]. Available: https://en.wikipedia.org/wiki/Word2vec

[4] "Ansible, deploy apps. manage systems. crush complexity," Web Page, accessed: 2017-02-26. [Online]. Available: https://www.ansible.com/

[5] "collectd - the system statistics collection daemon," Web Page, accessed: 2017-02-26. [Online]. Available: https://collectd.org/

cloudmesh cmd5 extension for AWS

MILIND SURYAWANSHI[1] **AND PIYUSH RAI**[2]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[2] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*

project-006, March 27, 2017

The cludmesh client will be extended to support cluster deployment on AWS using cmd5.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IR-P006/report/report.pdf

1. INTRODUCTION

We are going to look at cloudmesh client [1] man pages, understand what features are already supported for cloud like chameleon and implement similar support for AWS. The new features will be provided under aws subcommand. We will study the technologies currently in-use by the project and analyze how they can be used further to achieve our objective.

2. DESIGN

TBD

3. TECHNOLOGY USED

- Cloud Mesh
- AWS
- Cmd5
- libcloud

4. STEPS

- Understand cloudmesh client.
- Propose changes and get reviewed.
- Open aws account and test basic commands.
- Make changes and test.
- Benchmark.

ACKNOWLEDGEMENTS

TBD

REFERENCES

[1] Web Page. [Online]. Available: https://github.com/cloudmesh/client

AUTHOR BIOGRAPHIES

Milind Suryawanshi received his BE (Electronics and Telecommunication) in 2010 from The University of Pune. His research interests also include Big Data analytics for intelligence and research.
Piyush Rai received his BE (Computer) in 2011 from The University of Pune. His research interests also include Big Data analytics for military intelligence and frinancial markets.

A. WORK BREAKDOWN

The work on this project was distributed as follows between the authors:

Milind Suryawanshi. TBD

Piyush Rai. TBD

Identifying spam messages using R and Pandas over Docker Swarm

SAGAR VORA[1,*] AND RAHUL SINGH[1,**]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: vorasagar7@gmail.com
[**] Corresponding authors: rahul_singh919@yahoo.com

project-1, March 27, 2017

A classification model shall be built by working on a training data set of 5574 text messages, each marked as a spam or a legitimate message. The model shall be used to correctly predict the class of any new incoming text message as a spam or a legitimate one. The application shall be deployed using [1] Docker Swarm while data manipulation and classification shall be done using Pandas and R in conjunction.

Keywords: Docker, Swarm, R, Pandas

https://github.com/cloudmesh/sp17-i524/raw/master/project/S17-IR-P007/report/report.pdf

test

1. INTRODUCTION

To address the problem of incoming spam messages, a model shall be developed using the Bayesian Classification technique to correctly classify each incoming email/text message as a spam or a legitimate one. The model aims at developing a message filter that shall correctly classify messages based on word probabilities that are extracted from the training dataset.The training dataset to build the model consists of 5574 message records. Dataset taken from [2]. The training process shall use the cross-validation feature provided by R to build the classification model and use Bayes theorem of conditional probability to predict the class of each incoming message.

2. DESIGN

2.1. Building the Classification model

2.1.1. CrossValidation for the training data

To develop an efficient training model, we shall partition the data into 2 subsets - training data and classification data. We shall choose one of the subsets for training and other for testing. In the next iteration the roles of the subsets shall be reversed, i.e the training data becomes the classification one and vice versa. This operation shall be carried out until each individual record is used both as a classification and training record. We shall use the cross validation feature provided by R for this subsampling. This subsampling technique handles the underfitting problem and guarantees an effective classification model.

2.1.2. Training process

Content of each of the spam marked messages shall be processed through Naive Bayes Classifier. The classifier shall maintain a bag of words along with the count of each word occuring in the spam messages. This word count shall be used to calculate and store the word probability in a table that shall be cross-referenced to determine the class of the record on classification data [3].

A selected few words have more probability of occuring in a spam messages than in the legitimate ones. Eg: The word "Lottery" shall be encountered more often in a spam message. The classifier shall correlate the bag of words with spam and non-spam messages and then use Bayes Theorem to calculate a probability score that shall indicate whether a message is a spam or not. The results shall be verified with the results available on the training dataset and the classifier accuracy shall be calculated. The classifier shall use the Bayesian theorem over the training dataset to calculate probabilities of such words that occur more often in spam messages and later use a summation of scores of the occurence of these word probabilities to estimate whether a message shall be classified as spam or not. After working on several samples of the training dataset, the classifier shall have learned a high probability for spam based words whereas, words in legitimate message like family member or friends names shall have a very low probability of occurence.

2.2. Classifying new data

Once the training process has been completed, the posterior probability for all the words in the new input email is computed using Bayes theorem. A threshold value shall be defined to classify a message into either class. A message's spam probability is

computed over all words in its body and if the sum total of the probabilities exceeds the predefined threshold, the filter shall mark the message as a spam [4].

A higher filtering accuracy shall be achieved through filtering by looking at the message header i.e the sender's number/name. Thereby if a message from a particular sender is repeatedly marked as spam by the user, the classifier need not evaluate the message body if it is from the same sender.

3. DISCUSSION

TBD

4. DEPLOYMENT

Our application will be deployed using Ansible [5] playbook. Automated deployment should happen on two or more nodes clouds or on multiple clusters of a single cloud. Deployment script should install all necessary software along with the project code to the cluster nodes.

5. CONCLUSION

TBD

6. ACKNOWLEDGEMENT

We acknowledge our professor Gregor von Laszewski and all associate instructors for helping us and guiding us throughout this project.

7. APPENDICES

TBD

REFERENCES

[1] "Docker swarm," Web Page, accessed: 2017-03-10. [Online]. Available: https://docs.docker.com/engine/swarm/

[2] "SMS spam collection dataset," Web Page, accessed: 2017-03-10. [Online]. Available: https://www.kaggle.com/uciml/sms-spam-collection-dataset

[3] J. Provost, "Naive-Bayes vs. Rule-Learning in Classification of Email," in *Artificial Intelligence Lab*. The University Of Texas at Austin: The University Of Texas, 1999, accessed: 2017-03-10. [Online]. Available: http://mathcs.wilkes.edu/~kapolka/cs340/provost-ai-tr-99-281.pdf

[4] Wikipedia, "Naive bayes spam filtering," Web Page, January 2017, accessed: 2017-03-10. [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering

[5] "Ansible, deploy apps. manage systems. crush complexity," Web Page, accessed: 2017-03-12. [Online]. Available: https://www.ansible.com/

Big data Visualization with Apache Zeppelin

NAVEENKUMAR RAMARAJU[1,*] AND VEERA MARNI[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: naveenkumar2703@gmail.com, narayana1043@gmail.com

project-008, March 27, 2017

Apache Zeppelin is an open source notebook for data analytics and visualization. In this project we deploy Apache Zeppelin in cluster and visualize data stored in Spark across cluster using Apache Zeppelin interpreter that employs Python and Scala in same notebook.

Keywords: Zeppelin, Apache, Big data, Visualization

https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IR-P008/report/report.pdf

1. INTRODUCTION

Apache Zeppelin[1] is an interactive notebook that is used for data ingestion, discovery, analytics, visualization and collaboration. It has built in Spark integration and supports multiple language backends like Python, Hadoop HDFS, R etc. Multiple languages can be used within same Zeppelin script and share data between them. In this project we aim to deploy Zeppelin 0.7 along with in built Spark and backend languages R and Python across cluster using Ansible. Then install additional visualization packages provided by Apache Zeppelin Helium APIs.

We also aim to load a large data set into Spark across cluster and perform data analytics and visualization in cloud using Zeppelin. We have not decided about data set at this point.

2. EXECUTION PLAN

Deploy Spark, Zeppelin, Helium, R and Python using ansible by March 31.

Find a data set by March 31.

Data set found.

Tamilnadu Electricity Board Hourly Readings Data Set

This data can be effectively produced the result to fewer parameter of the Load profile can be reduced in the Database

Data Set Characteristics: Multivariate
Number of Instances: 45781
Attribute Characteristics: Real
Number of Attributes: 5
Associated Tasks: Classification, Regression, Clustering
Source:K.Kalyani ,kkalyanims@gmail.com,T.U.K Arts College,Karanthai,Thanjavur.

Relevant Papers: Efficient Electricity Utilization By IHBMO
Attribute Information: forkva,forkw,type,sector,service.

Benchmark the deployment times of individual and all items by April 7.

Load the data distributed across machines using Spark by April 7.

Perform Visualization on loaded data with Zeppelin using Spark, Scala and Python in same environment and validate the collaboration across cluster by April 14.

See, if configuration of Apache clusters inside Zeppelin could be done employing Ansible at deployment time by April 17.

Finish report and submit project on April 21.

3. DEPLOYMENT

TBD

4. BENCHMARKS

TBD

5. VISUALIZATION WITH ZEPPELIN

TBD

6. SUPPLEMENTAL MATERIAL

TBD

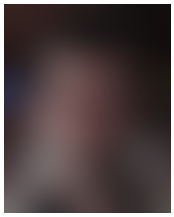
ACKNOWLEDGEMENTS

TBD

REFERENCES

[1] Apache Zeppelin, "Zeppelin 0.7.0 Documentation," Web Page, Apache Software Foundation, Mar. 2017. [Online]. Available: https://zeppelin.apache.org/docs/0.7.0/

AUTHOR BIOGRAPHIES



Naveenkumar Ramaraju yet to update his Bio

Veera Marni yet to update his bio

A. WORK BREAKDOWN

TBD

Naveenkumar Ramaraju TBD

Veera Marni TBD

B. REPORT CHECKLIST

- □ Have you written the report in word or LaTeX in the specified format?
- □ Have you included the report in github/lab?
- □ Have you specified the names and e-mails of all team members in your report. E.g. the username in Canvas?
- □ Have you included the HID of all team members?
- □ Does the report have the project number added to it?
- □ Have you included all images in native and PDF format in gitlab in the images folder?
- □ Have you added the bibliography file in bibtex format?
- □ Have you submitted an additional page that describes who did what in the project or report?
- □ Have you spellchecked the paper?
- □ Have you made sure you do not plagiarize?
- □ Have you made sure that the important directories are all lower case and have no underscore or space in it?
- □ Have you made sure that all authors have a README.rst in their HID github/lab repository?
- □ Have you made sure that there is a README.rst in the project directory and that it is properly filled out?
- □ Have you put a work breakdown in the document if you worked in a group?

C. POSSIBLE TECHNOLOGY PAPER OUTLINE

The next sections are just some suggestions, your may want to add sections and subsections as you see fit. Images and references do not count towards the 2 page length. Please use the `\section`, `\subsection`, and `\subsubsection` commands in your paper. do not introduce hardcoded numbers. Use the `\ref` and `\label` commands to refer to the sections.

Abstract Put in the abstract a summary what this paper is about

1. Introduction Introduce the technology and provide general useful information.

2. Architecture If applicable include a description about architectural details. This may include a figure. Make sure that if you copy a figure you put the **[?]** in the caption also. Otherwise it is plagiarism.

2.1. API comment on the API which could include language bindings

2.2. Shell Access If applicable comment on how the tool can be used from the command line

2.3.Graphical Interface If applicable comment on if the technology has a GUI

3. Licensing Often tools may have different versions, some fre, some for pay. Comment on this. For example while a tool may offer a comercial version this version may be too costly for others. Identify especially the difference between features for free vs commercial tools.

Sometimes you may need to introduce this also in the introduction as there may be a big difference and without the knowledge you do not provide the user an adequate introduction.

4. Ecosystem Some technologies have a large ecosystem developed around them with extensions plugins and other useful tools. Identify if they exists and comment on what they can achieve

provide potentially a mindmap or a figure illustrating how the technology fits in with other technologies if applicable.

4. Use Cases

4.1. Use Cases for Big Data Locate and describe major usecases that demonstrate the technology while focussing on big data related use cases. Make sure you do proper references with the [**?**] command. Do not put URLs in the text.

4.2. Other Use Cases Some technologies may not just be used for big data, find other makor use cases from other areas if applicable. Make sure you do proper references with the [**?**] command. Do not put URLs in the text.

5. Educational material Put information here how someone would find out more about the technology. Use important material and do not list hundrets of web pages, be selective.

6. Conclusion Put in some conclusion based on what you have researched

Acknowledgement Put in the information for this class and who may sponsor you. Examples will be given later

Cloudmesh Docker Extension

KARTHICK VENKATESAN[1] AND ASHOK VUPPADA[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.

S17-IR-P009, March 27, 2017

This project will be to create a cmd5 based command line interface to docker with commands similiar to those currently supported in cloudmeshclient .The client will also be enhanced to integrate to docker swarm.A Benchmark suite will be created to benchmark the deployment of application using the client.

Keywords: Cloud, I524

https://github.com/cloudmesh/classes/blob/master/docs/source/format/report/report.pdf

replace the url with your URL on github

1. INTRODUCTION

The project will be to create a cmd5 based command line interface to docker with commands similiar to those currently supported in cloudmeshclient .The client will also be enhanced to integrate to docker swarm.A Benchmark suite will be created to benchmark the deployment of application using the client. Ansible scripts will be created for deployment of Docker into the virtual machines and also to deploy docker images for Application/Services

2. TECHNOLOGY USED

- Cloud Mesh
- Docker
- Docker Swarm
- Ansible

3. DEVELOPMENT

- Task 1: Develop Ansible Script for Docker Install on Virtual Machine
- Task 2: Develop Ansible Script for Installing an service(TBD) on a Docker Swarm
- Task 3: Develop Docker interface to create and provision single containers leveraging Docker Rest API
- Task 4: Develop commands in cloud mesh using cmd5 to manage Docker Swarm leveraging Docker Rest API
- Task 5: Develop benchmark script for bechmarking the docker container management options and deployment of application into containers.

4. EXECUTION PLAN

- Step 1: Create a a group of Virtual Machines in Chameleon cloud using Cloud Mesh
- Step 2: Run Ansible script to install Docker on all the Virtual Machines provisioned
- Step 3: Create a Docker Swarm with multipe Nodes in each Virtual Cluster using Cloud Mesh
- Step 4: Deploy Application (TBD) Docker Images into the Docker nodes using Ansible
- Step 5: Run Bechmark scripts to bechmark both Application Deployment and Provisioning .

5. DOCOPTS MANUAL PAGE FOR CLOUDMESH DOCKER

TBD

6. DATA ANALYSIS APPLICATION TO BE DEPLOYED IN DOCKER

TBD

7. BENCHMARK PROCESS

TBD

8. BENCHMARK RESULTS

TBD

9. STEPS TO EXECUTE

TBD

Add a single reference at least to see if the refernce management works

REFERENCES

Deployment of Vehicle Detection application on Chameleon clouds

ABHISHEK NAIK[1,*] **AND SHREE GOVIND MISHRA**[2,*]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[2] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*
[*] *Corresponding authors: absnaik810@gmail.com, shremish@indiana.edu*

project-001, March 27, 2017

This project focuses on the deployment of Vehicle Detection application on multiple Chameleon clouds using Ansible playbook. It also focuses on the benchmarking of the deployment results and its analysis.

Keywords: Vehicle detection, Ansible, Cloudmesh, OpenCV, Haar Cascades, Cloud, I524

https://github.com/absnaik810/sp17-i524/blob/master/project/S17-IR-P010/report.pdf

1. INTRODUCTION

Vehicle Detection forms an integral part of the development of new technologies like fully self-driving cars, etc. One of the techniques to perform such detection is by using Haar Cascades [1]. This technique has been applied to vehicle detection by creating a haar-cascade cars.xml file which has been trained using 526 rear-end images of cars. In this project, we would be extending this vehicle detection approach to enable it to run on multiple clouds. This deployment would initially be done on the localhost, followed by deployment on 1, 2, 3 and 4 clouds. Cloudmesh client would be used for cloud management and Ansible scripts would be used for software stack deployment. Appropriate benchmarking would be carried out at each iteration using some benchmarking technique or tool (TBD).

2. SOFTWARE STACK

- Ansible
- Cloudmesh
- TBD

3. EXECUTION PLAN

The execution plan that would be followed is as under:

3.1. Week 1

Deployment of the vehicle detection application on the localhost, using Ansible playbook. Benchmarking of the important factors that are chosen for observation. Analysis of the benchmarking results thus obtained.

3.2. Week 2

Reservation of a single Chameleon cloud using cloudmesh client. Deployment of the vehicle detection application on the cloud, using Ansible playbook. Benchmarking of the important factors that are chosen for observation. Analysis of the benchmarking results thus obtained.

3.3. Week 3

Reservation of 2 Chameleon clouds using cloudmesh client. Deployment of the vehicle detection application on the clouds, using Ansible playbook. Benchmarking of the important factors that are chosen for observation. Analysis of the benchmarking results thus obtained.

3.4. Week 4

Reservation of 3 and 4 Chameleon clouds using cloudmesh client. Deployment of the vehicle detection application on the clouds, using Ansible playbook. Benchmarking of the important factors that are chosen for observation. Analysis of the benchmarking results thus obtained.

3.5. Week 5

Final analysis of the benchmarking results obtained so far. Studying the effect of software stack deployment on multiple clouds. Publishing the final report and conclusion.This project focuses on the deployment of Vehicle Detection application on multiple Chameleon clouds using Ansible playbook. It also focuses on the benchmarking of the deployment results and its analysis.

4. DEPLOYMENT

As per the current plan, we would first identify the software stack that would be required to run the vehicle detection application on the localhost, which is a Ubuntu 16.04.02 machine

with 3 GB RAM. This software deployment would be done using an Ansible script [2]. Ansible is an automation engine which we would use to orchestrate the software deployment via playbooks using the inventory.txt file. Important factors like the time required for deployment, the 'ease' of deployment (TBD), etc. would be benchmarked using some benchmarking technique or tool (TBD).

The next step would be to deploy this software stack on a single cloud. We would be using the cloudmesh client [3], a command line based client, to reserve a cloud and enable access to it. We would consider the feasibility of deploying the software stack on the cloud, by careful observation of the benchmarking results obtained in the previous step of deploying it on the localhost. Depending upon the outcomes, we would be able to identify the minimum system requirements that would be typically required for the deployment of the software stack. This deployment would again be achieved by using Ansible scripts [2]. These results would again we benchmarked for further analysis.

As the next step, we would carry out this deployment on two clouds. Ansible would again be used for deployment of the software stack and cloudmesh client for the management of the clouds. The results obtained would again be benchmarked and analysis performed. We would iterate this for 3 and 4 clouds. More clouds might be considered depending upon the time constraints of the project.

Finally the observations and analyses would be published as a report.

5. BENCHMARKING RESULTS AND ANALYSIS

TBD

6. CONCLUSION

TBD

7. ACKNOWLEDGEMENTS

TBD

REFERENCES

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*. Cambridge, MA: Institute of Electrical and Electronics Engineers (IEEE), Dec. 2001. [Online]. Available: http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf

[2] Wikipedia, "Ansible (software)," Web page, online; accessed 10-Mar-2017. [Online]. Available: https://en.wikipedia.org/wiki/Ansible_(software)

[3] G. von Laszewski, "Cloudmesh/client," Code Repository, accessed: 2017-3-10. [Online]. Available: https://github.com/progrium/buildstep

Head Count Detection Using a Docker Swarm Cluster

ANURAG KUMAR JAIN[1], PRATIK SUSHIL JAIN[1], AND RONAK PAREKH[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*

project-000, March 27, 2017

By deploying our face detection application on Docker cluster, we will try to achieve high throughput by parellalizing processing of images for head count task on multiple nodes each having thousand of pictures.

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/tree/master/project/S17-IR-P011

1. INTRODUCTION

Counting the number of people in a image has been a challenge in the field of computer vision [1]. Given a huge number of images, finding the number of people in each image can become a cumbersome task. We try to solve this issue using our distributed approach using a docker swarm cluster which is a cluster of Docker [2] engines, or nodes, where services have to be deployed. We run the OpenCV [3] face detection algorithms on smaller data in multiple nodes and obtain the head count of the people in each picture.

ACKNOWLEDGEMENTS

This project is undertaken as part of I524: Big Data And Open Source Software Projects at Indiana University, Bloomington. We would like to Prof. Gregor von Laszewski and Assosiate Instructors for their help.

REFERENCES

[1] Wikipedia, "Face detection," Web Page, Feb. 2017, online; accessed 09-March-2017. [Online]. Available: https://en.wikipedia.org/wiki/Face_detection

[2] Wikipedia, "Docker(software)," Web Page, Feb. 2017, online; accessed 09-March-2017. [Online]. Available: https://en.wikipedia.org/wiki/Docker_(software)

[3] Wikipedia, "OpenCV," Web Page, Feb. 2017, online; accessed 09-March-2017. [Online]. Available: https://en.wikipedia.org/wiki/OpenCV

Optical Character Recognition

SABER SHEYBANI[1] AND SUSHMITA SIVAPRASAD[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: sheybani@umail.iu.edu,sushsiva@umail.iu.edu

project-000, March 27, 2017

Optical Character Recognition is a technology for converting images into machine encoded text format. In this project, the input data is in PNG format and our goal is to recognize the words/letters in the image as accurately as possible and convert the dataset into TXT format. The heart of OCR is a classification algorithm which will be implemented using Python programming language. The algorithm will be deployed using the Ansible technology [1] to remote virtual clusters.

Keywords: OCR,ansible,classification

https://github.com/SushmitaSivaprasad/sp17-i524/tree/master/project/S17-IR-P012/report/report.pdf

1. 1.INTRODUCTION

This project proposal provides an overview on how we plan on implmenting the OCR technology. It gives a background on the kind of technology that has been used , delving into some of the basic concepts used in the implentation process.We have also discussed some important applications of this technology in the real world.

2. 2.BACKGROUND

2.1. 2.1 OCR Technology

Optical Character Recognition is a technology which is used to convert different types of documents that can be in the form of scanned papers (raster images) or PDF into an editable and searchable form [2]. The images can be in either the basic black & white or multicolored. The technology first analyzes the structure of the document and divides it into smaller segments. Finally, individual characters are singled out one by one and fed to a classification algorithm which will return the closest letter that the individual character could possibly be identified with.

2.2. 2.2 Ansible

Ansible is an IT automation tool. It uses YAML in order to issue the state of the server [2]. Ansible implements the internal command that is required to reach that state which depends on the operating system. The ansible playbook which consists of these internal commands can be applied across any server or service. There is no requirement to instal an additional software on the target system as the commands are run over an SSH session.

2.3. 2.3 Feed Forward Neural Networks

Artificial Neural Network is a paradigm in computing, inspired by the structure of biological nervous systems. It consists of a network of processing units, where the output of each unit is a nonlinear function of its weighted inputs that come from other units. Such network can be trained to solve different kinds of problems, including classification and clustering. A feed forward neural networks is one in which the neurons are organized in a number of layers and each layer only feeds to the next one, but not to the previous one (no feedback). However, in a back-propagation process, the errors from one iteration of classification will be fed back from the output to the network, in order to modify and improve the network for next iterations.

3. 3.ANSIBLE DEPLOYMENT

We will be using Ansible [1] for running the OCR algorithm. The jobs will be collected and organized in a Playbook [3] and run on virtual clusters provided by Chameleon Cloud [4]. The tasks will include Installing the essential libraries on the remote machine and running the program.

4. 4.OCR IMPLEMENTATION

Optical Character Recognition have already been developed in numerous ways, focusing on different goals. For our purpose, various classification algorithms such as K-Nearest Neighbor and Neural Network (multilayer perceptron) can be used. For this project, a feedforward, back-propagation Neural Network will be used. The steps are as follows: Preprocessing: The input images need to be segmented into units that each of them keep only one glyph (symbol). Also, the colored or grayscale images will be binarized. Feature extraction: The glyphs will be decomposed into features like lines, closed loops, line direction, and

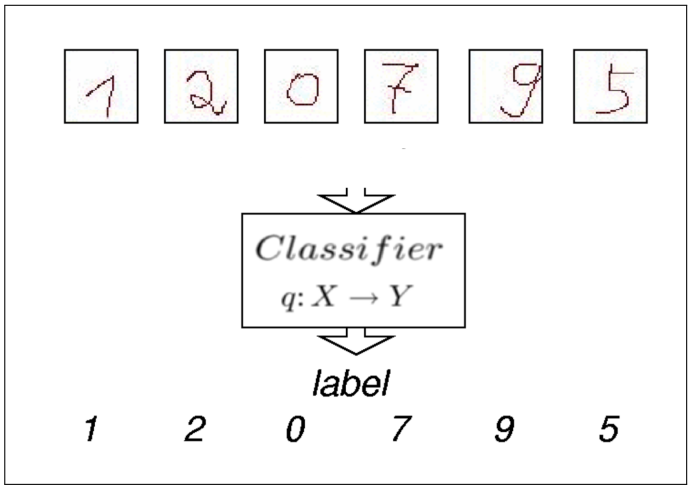


Fig. 1. Illustration of an OCR system [5]

line intersections. Character recognition: The image features will be fed to the neural network and they will be compared with stored glyph features and the nearest match will be chosen, after multiple iterations of classification by the network.

5. 5.APPLICATION

OCR converts images to machine-readable text. That will make it the initial tool that needs to be used for processing any documents or simply any written material in a digital image, which has been captured by a camera[6]. It's output can be stored significantly more compact than scanned images. But beyond that, it enables us to process the output information for numerous applications. Examples of these applications include creating a narrator machine to help the visually impaired read nondigital documents and signs, or automatic recognition of automobile number plates.

6. ACKNOWLEDGEMENT

A very special thanks to Professor Gregor von Laszewski and the teaching assistants Miao Zhang and Dimitar Nikolov for all the support and guidance. This project proposal is written during the spring 2017 semester course I524: Big Data and Open Source Software Projects at Indiana University Bloomington.

REFERENCES

[1] "Ansible Documentation," Web Page, Mar. 2017. [Online]. Available: http://docs.ansible.com/ansible/index.html
[2] "What is OCR and OCR Technology," Web Page, 2017. [Online]. Available: https://www.abbyy.com/en-us/finereader/what-is-ocr/
[3] "Playbook," Web Page, Mar. 2017. [Online]. Available: http://docs.ansible.com/ansible/playbooks.html
[4] "A configurable experimental environment for large-scale cloud research," Web Page, Jan. 2017. [Online]. Available: https://www.chameleoncloud.org/
[5] "Demo: Optical character recognition."
[6] "Optical Character Recognition," Web Page, Mar. 2017. [Online]. Available: https://en.wikipedia.org/wiki/Optical_character_recognition

Weather Data Analysis

VISHWANATH KODRE[1], SABYASACHI ROY CHOUDHURY[1], AND ABHIJIT THAKRE[1]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[1] Corresponding authors: sabyasachi087@gmail.com, vkodre@gmail.com, athakre@gmail.com

project-000, March 27, 2017

The project aims to analyze any relationship between change in climate, geo- magnetic field and natural disasters with focusing on use of Hadoop Framework for data analysis and Ansible for automating deployment and monitoring.

Keywords: Cloud, I524

https://github.com/cloudmesh/classes/blob/master/docs/source/format/report/report.pdf

1. INTRODUCTION

The study of environmental science and climatic changes around has been done for decades, the study has always been predictive based on the past experiences and forecasting of the weather conditions around us. With use of modern days technologies it determining the climatic changes and with analysis done around it has helped human being to prepare and face the natural calamities. Though with current equipment weather department has strengthen their arms but has not been able to be full proof and many time its not been able to predict/ forecast the climatic changes effectively. The study of the whether data and geo graphical changes is ongoing evolving process. Thus more and more researcher needs modern days tools and technologies to leverage it and forecast more accurately.

1.1. Objective

The goal of this is to study the weather data and analyze the relationship between the geo graphical changes such change in geo magnetic field and/or natural disaster. With use of Hadoop for distributed data analysis aims to finds any pattern that might exists between these parameters. The course of the analysis will also provides visualization of these parameters in order to identify any pattern in a more intuitive way. By leveraging the power ansible for application deployment over cluster and monitoring the application performance to determine scalability and throughput. The conclusion will be determine by establishing any existing pattern, analysis done over it and by visualizing it.

2. DATA SOURCES

Weather data has been recorded since 19th century. This data can be used to estimate climate changes and forecasting. The same data can be can be used to find any existing pattern with natural disasters. Following sources has been compiled for weather, natural disaster and geo magnetic fields.

- Weather-Data[1]
- Natural Disaster[2]
- Geo Magnetic Field[3]

3. HIGH LEVEL DESIGN

The design of the application is thought of leveraging power of Hadoop as main processing unit of analysis with deployment on the cluster environment where application requires multiple processing units for execution, database for persistence and visualization tools for graphical outputs. The project is divided into following steps:

- Data cleaning and persistence - The raw data cannot be use directly for analysis. First data has to be parsed and required parameters will be extracted. Then this extracted data will be dumped into a NoSql database.
- Core Analysis Program - Core analysis program will be responsible for figuring out any hidden patterns between aforesaid parameters. Program will compare natural disasters occurred, geo-magnetic orientation and climate data set on a given location and duration and compute relationship between them. The program will be an MapReduce implementation and is the heart of the application. The program will be executed through Hadoop framework. Hadoop will execute the program in a distributed manner.
- Deployment and Monitoring - The application needs multiple processing units and monitoring system. Ansible will be used for deployment and manage nodes for program execution.Ansible will be responsible for following tasks i) Deployment and configuration of Hadoop on the multiple nodes. ii) Starting Hadoop servers, inserting/reading data. iii) Execution of the commands to run the analysis using

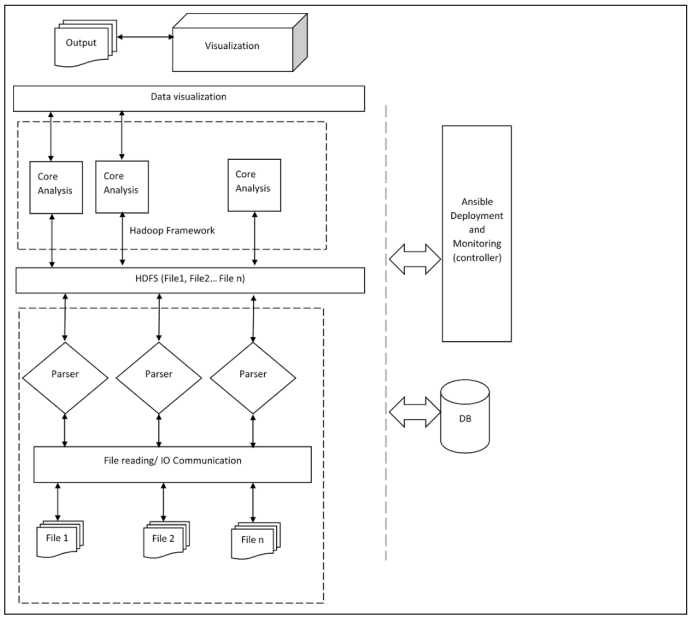


Fig. 1. Architecture

Hadoop to filter the input data and write response to HDFS or some output file. iv) This output can be then passes to the visualization step as the input data.

- Visualization - Finally once the programs completes execution, using the scikit-tool or other visualization tool kit and the output file, graphs and patterns depicting the relationship can be plotted more intuitive representation.
- BenchMarking - The application can be benchmarked for the scalability by addition more nodes and checking the performance for strong scaling. The report will be represented in tabular format.

REFERENCES

[1] "Weather data," Web page. [Online]. Available: https://www.ncdc.noaa.gov/
[2] "Natural disaster," Web page. [Online]. Available: http://www.emdat.be/
[3] "Geo magnetic field data," Web page. [Online]. Available: https://geohazards.usgs.gov/mailman/listinfo/geomag-data

S17-IR-P014/report/report.pdf not submitted

S17-IR-P015/report/report.pdf not submitted