Big Data Technologies

Editor: Gregor von Laszewski

March 13, 2017

0.1 Contributors

Name	HID	Title
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
Scott McClary	TBD	Charge Detection Mass Spe
Mark McCombe	S17-IO-3012	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
Ribka Rufael	S17-IO-3016	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
Rahul Raghatate, Snehal Chemburkar	S17-IR-2026, S17-IR-2006	TBD
Kumar Satyam, Piyush Shinde, Srikanth Ramanam	S17-IR-2031,S17-IR-2035,S17-IR-2028	TBD
Abhishek Gupta, Avadhoot Agasti	S17-IO-3005, S17-IO-3000	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
Sagar Vora	S17-IR-2041	TBD
TBD	S17-IR-2017 and S17-IR-2029	TBD
Karthick Venkatesan,Ashok Vuppada	S17-IO-3023,S17-IO-3024	TBD
Abhishek Naik Shree Govind Mishra	S17-IR-2022 S17-IR-2021	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD
ENTER YOUR NAME HERE	ENTER YOUR HID HERE	TBD

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 11, 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/classes/blob/master/docs/source/format/report/report.pdf

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.

VIRTUAL ROBOT SWARM COMPONENTS

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)

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.

Ansible

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

Testing Environment

TBD; briefly describe cloudmesh

VIRTUAL ROBOT SWARM PROJECT IMPLEMENTATION

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.

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.

Modifications, Pitfalls

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

Initializing the Swarm

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

Begin Task and Monitor Swarm's Progress

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

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;

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.*

VR SWARM PROJECT CONCLUSIONS

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

EXAMPLES OF ARTICLE COMPONENTS

The sections below show examples of different article components.

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.

Sample Figure

Figure 1 shows an example figure.

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}

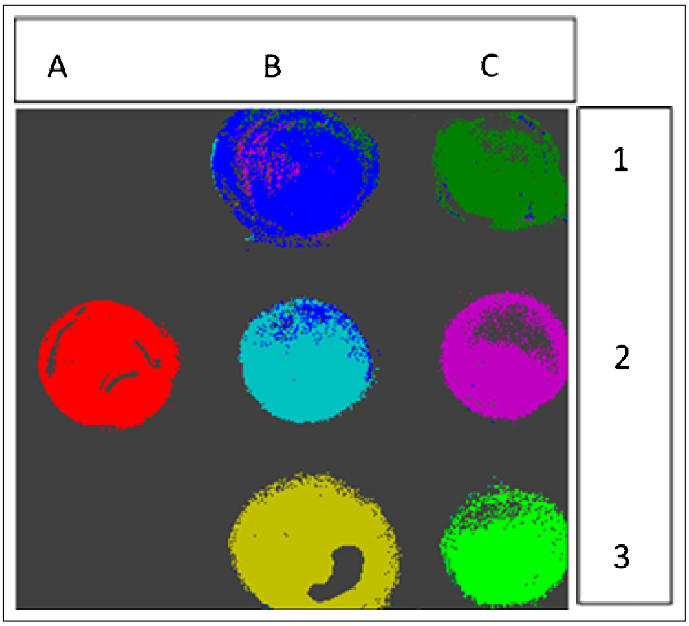


Fig. 1. False-color image, where each pixel is assigned to one of seven reference spectra.

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

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

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

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

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

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].

SUPPLEMENTAL MATERIAL

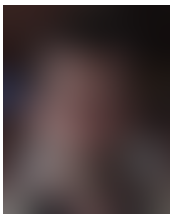
REFERENCES

[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.

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.

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 9, 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

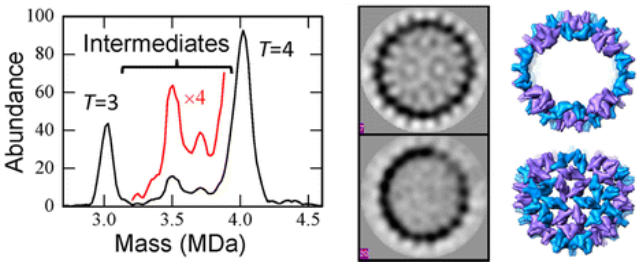


Fig. 1. The chart to the left displays an accurate measurement of the Hepatitis B virus (HBV) created by the research group's CDMS application [**?**]. This detailed mass information is used to create the images shown in the middle and to the right, which show 2-D and 3-D models of HBV.

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.

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 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.

2.4. 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.

2.5. April 10, 2017 - April 16, 2017

This week I wrote the majority of the project report.

Fig. 2. CDMS Pipeline

2.6. 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.

4. CDMS RESEARCH PIPELINE

5. LICENSING

TBD

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.

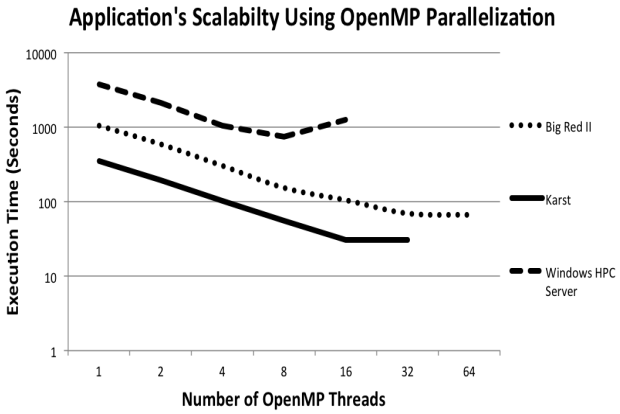


Fig. 3. The figure above shows the scalability (i.e. reduction in time-to-solution) as the number of OpenMP threads increase on local servers, Supercomputers and Clouds.

6.1. OpenMP Scalability

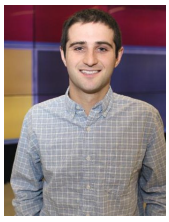
7. CONCLUSION

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/

S17-IO-3012/report/report.pdf not submitted

S17-IO-3013/report/report.pdf not submitted

Deploying CouchDB Cluster

RIBKA RUFAEL[1,*]

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

project-000, March 12, 2017

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

Keywords: Cloud, I524

https://github.com/cloudmesh/classes/blob/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

Develop Ansible script to deploy CouchDB on Ubuntu 16.04

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

S17-IO-3017/report/report.pdf not submitted

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

S17-IO-3019/report/report.pdf not submitted

S17-IO-3022/report/report.pdf not submitted

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

S17-IR-2004/report/report.pdf not submitted

S17-IR-2013/report/report.pdf not submitted

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

S17-IR-2019/report/report.pdf not submitted

S17-IR-2034/report/report.pdf not submitted

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

S17-IR-2044/report/report.pdf not submitted

S17-IR-P001/report/report.pdf not submitted

S17-IR-P002/report/report.pdf not submitted

Project Report: 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

project-001, March 13, 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, I524

https://github.com/rahulraghatate/sp17-i524/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

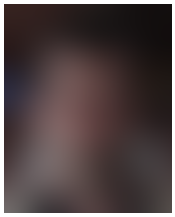
This project is undertaken as part of the I524: Big Data and Open Source Software Projects coursework at Indiana University. We would like to thank our Prof. Gregor von Laszewski, Prof. Gregory Fox and the Associate Instructors for their help and support

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 13, 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.

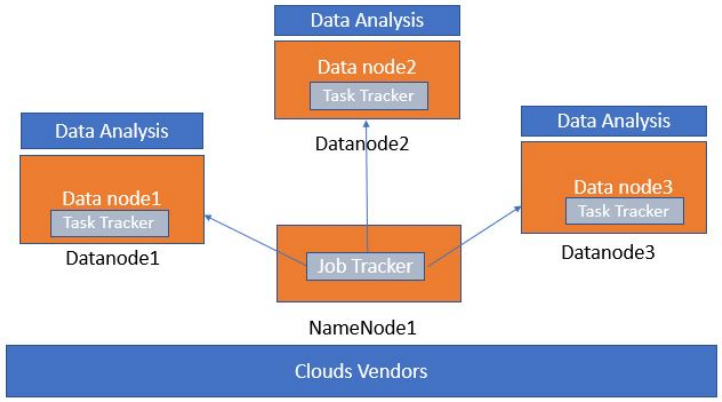


Fig. 1. Deployment Architecture

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 10, 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/

S17-IR-P006/report/report.pdf not submitted

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 13, 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/tree/master/project/S17-IR-P007/report/report.pdf

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.

DESIGN

Building the Classification model

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.

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.

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.

DISCUSSION

TBD

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.

CONCLUSION

TBD

ACKNOWLEDGEMENT

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

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 13, 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.

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

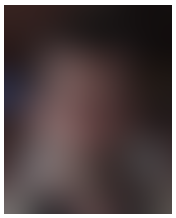
ACKNOWLEDGEMENTS

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

Project Report:Integrate Docker into cloudmesh and demonstrate its use

KARTHICK VENKATESAN[1] AND ASHOK VUPPADA[1]

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

project-000, March 10, 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

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

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 13, 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

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).

SOFTWARE STACK

- Ansible
- Cloudmesh
- TBD

EXECUTION PLAN

The execution plan that would be followed is as under:

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.

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.

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.

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.

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.

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.

BENCHMARKING RESULTS AND ANALYSIS

TBD

CONCLUSION

TBD

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

S17-IR-P011/report/report.pdf not submitted

S17-IR-P012/report/report.pdf not submitted

S17-IR-P013/report/report.pdf not submitted

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