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.*

This template can be used to prepare a research article for I524. Note that this template can be run from your own TEX system or within the cloud-based Overleaf system or Sharelatex systems.

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 [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).

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