Application of Edge-to-Cloud Methods Toward Deep Learning

Khushi Choudhary*‖, Nona Nersisyan*‖, Edward Lin*‖, Shobana Chandrasekaran*¶, Rajiv Mayani*
Loïc Pottier*, Angela P. Murillo†, Nicole K. Virdone*, Kerk Kee‡, Ewa Deelman*
*Information Sciences Institute, University of Southern California, CA, USA
{khushich, nnersisy, eddiel}@usc.edu {schand, mayani, lpottier, virdone, deelman}@isi.edu
†School of Informatics and Computing, Indiana University, IN, USA
apmurill@iu.edu
‡Texas Tech University, TX, USA
kerk.kee@ttu.edu

2022 IEEE 18th International Conference on e-Science (e-Science) | 978-1-6654-6124-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ESCIENCE55777.2022.00065

***Abstract*—Scientific workflows are important in modern computational science and are a convenient way to represent complex computations, which are often geographically distributed among several computers. In many scientific domains, scientists use sensors (e.g., edge devices) to gather data such as CO2 level or temperature, that are usually sent to a central processing facility (e.g., a cloud). However, these edge devices are often not powerful enough to perform basic computations or machine learning inference computations and thus applications need the power of cloud platforms to generate scientific results. This work explores the execution and deployment of a complex workflow on an edge-to-cloud architecture in a use case of the detection and classification of plankton. In the original application, images were captured by cameras attached to buoys floating in Lake Greifensee (Switzerland). We developed a workflow based on that application. The workflow aims to pre-process images locally on the edge devices (i.e., buoys) then transfer data from each edge device to a cloud platform. Here, we developed a Pegasus workflow that runs using HTCondor and leveraged the Chameleon cloud platform and its recent CHI@Edge feature to mimic such deployment and study its feasibility in terms of performance and deployment.**

***Index Terms*—Scientific Workflows, Workflow Management Systems, Edge Computing, Pegasus, Zooplankton, Machine Learning.**

I. Introduction

This paper proposes a method of carrying out experiments that rely on edge and cloud computing in order to make the experimental process more streamlined and efficient. To experiment with the system, we have chosen an existing application [1] that classifies lake zooplankton.

By analyzing plankton, it is possible to learn about the health of ecosystems and how they may be affected by environmental changes. This is important in today's world as we face many climate-related challenges. The experiment run will classify plankton images into 35 different classes that will reveal information about the state of the surrounding ecosystem.

In this work, we build the data classification workflow and show how the execution of such a workflow can be automated by leveraging existing technologies. Running an experiment on both edge and cloud devices provides automation and improves performance. When programs are only run in the cloud, latency exists because data processing is happening far from the user. With edge devices, data processing or running workflows can happen on devices that are located closer to the user, thereby improving performance, but only if the edge devices are powerful enough for a given application.

‖Undergraduate student, equal contribution
¶Graduate student, equal contribution

II. Systems Used

In this work, we used the Chameleon testbed, the Pegasus workflow management system, and HTCondor for job management.

Chameleon Cloud Site. The Chameleon Cloud Site [2] is an NSF-funded testbed system primarily used for computer science experimentation. This testbed is designed to be reconfigurable. Chameleon utilizes a Python library that allows users to work with the testbed and Jupyter notebooks. For our project, we utilized CHI@TACC, CHI@Edge, and the Jupyter Interface. CHI@TACC was used for the cloud site and CHI@Edge was used for the edge site. We used Jupyter notebooks to configure our projects, create execution sites, and execute the workflow created.

Pegasus Workflow Management System. Pegasus [3] is a workflow management system. Scientific workflows allow users to execute computational tasks that often have multiple steps. A workflow is "an abstraction used by scientists to express an ensemble of complex, computational operations" [3]. Pegasus is responsible for managing the execution of the workflow (see Figure 1).

HTCondor. HTCondor [4] is a job management system that allows reliable execution of many jobs on the behalf of the user. By doing so, HTCondor is able to do large amounts of work and computation without any user intervention.

III. Approach

Cloud Site. First, the cloud site was established. We configured a CHI project and set the site to CHI@TACC. We then used

978-1-6654-6124-5/22/$31.00 ©2022 IEEE
DOI 10.1109/eScience55777.2022.00065

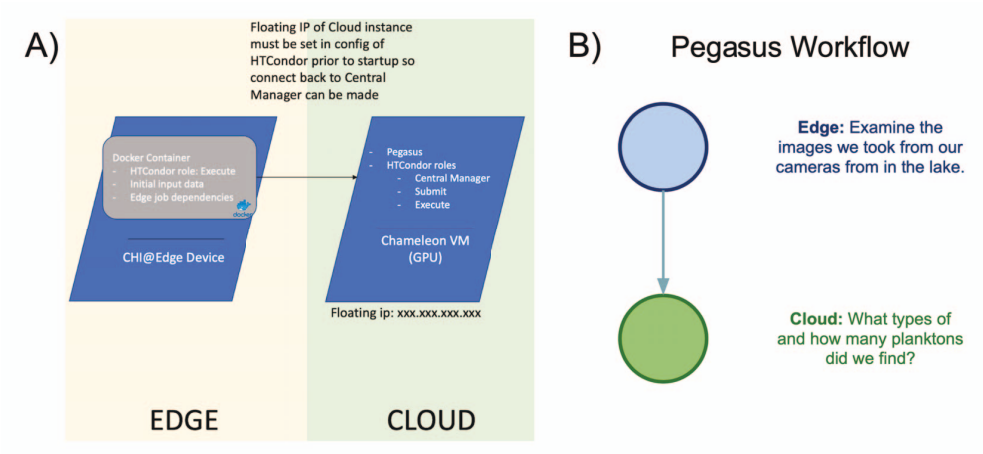

Figure 1: Global architecture of the project.

Chameleon's Python API to create a lease with a reservation for 1 node and 1 floating IP. Once the lease was created, we launched a bare metal instance to start a virtual machine. To do so, a server was created with the "Ubuntu 18.04" image. On this virtual machine, a floating IP address was then associated. We then installed and configured the HTCondor and Pegasus software. The software was installed by creating a .sh file where we wrote a script that installed HTCondor, set the submit, execute, and central manager roles, and installed Pegasus. Once the software was installed and configured, we performed checks to ensure it was installed and configured correctly and using the correct versions of the software. They were configured correctly if it was able to successfully show 96 available computing slots.

Edge Site. We then moved to CHI@Edge to begin creating the edge site. Once again, the Chameleon Python API was utilized to create a lease with a reservation for one Raspberry Pi. Once the lease was created, a container containing the plankifier ML model (rajivmayani/condor8-arm64-plankifier-worker) was started. After, the software was configured. This was done through another .sh file. Within this .sh file, HTCondor's execute node role was configured. Once again, checks were performed. Specifically, the command `condor_status` was used. We knew it was configured correctly if this command showed that there were now 97 compute slots available.

Workflow and Execution. To write the workflow, we took the original bash script and converted it to a Pegasus workflow using the Pegasus Python API. For each execution site, a job was created. Jobs are used for commands that need to be run. This includes command line arguments that need to be passed, input files, or even output files that will be outputted after the job is done. After each job, a transformation catalog entry was also created. Transformation catalogs are used to find locations of the executable that are used by the jobs in the workflow. First, a job for the edge site was created. For the edge job, arguments that were used were added and stored in an output file. Then, a transformation catalog entry for the edge job was created. For the cloud job, arguments that were needed were added alongside an input file and an output file. The input file took the output file that was created in the edge job. Once again, a transformation catalog entry was created for the cloud job. Lastly, the workflow was executed, and we were able to successfully output the results.

IV. Conclusion

As a result of our project, we found 26 types of zooplankton. This could have been done manually, but it would have taken a lot of time and effort and would be error-prone. Modern-day technology helps us classify things faster than ever before. This was done by using a cloud site and an edge site. We established a connection between the two, and as a result of the workflow, we were able to find the different types of zooplankton. In the future, we plan to apply this workflow to data collected from other lakes, retraining the models for new zooplankton classification. This would enable other ecologists to benefit from these automated tools.

Acknowledgments

This project was conducted as part of the CI Compass Student Fellowship Program supported by the National Science Foundation award #2127548.

References

[1] S. Kyathanahally, T. Hardeman, E. Merz, T. Kozakiewicz, M. Reyes, P. Isles, F. Pomati, and M. Baity-Jesi, "Data for: Deep learning classification of lake zooplankton," 2021. [Online]. Available: https://opendata.eawag.ch/dataset/deep-learning-classification-of-zooplankton-from-lakes

[2] K. Keahey *et al.*, *Chameleon: A Scalable Production Testbed for Computer Science Research*. CRC Press, 05 2019, pp. 123–148.

[3] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus: a Workflow Management System for Science Automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015, funding Acknowledgements: NSF ACI SDCI 0722019, NSF ACI SI2-SSI 1148515 and NSF OCI-1053575. [Online]. Available: http://pegasus.isi.edu/publications/2014/2014-fgcs-deelman.pdf

[4] D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: the Condor Experience," *Concurrency and computation: practice and experience*, vol. 17, no. 2-4, pp. 323–356, 2005.