

INTRODUCTION

- Propose a method of carrying out experiments that rely on edge and cloud computing to make an experimental process more streamlined and efficient.
- Existing use case that classifies lake zooplankton, which will infer knowledge of surrounding ecosystem health [1].
- 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.
- Running experiments on both edge and cloud devices improves performance:
 - When jobs run only in the cloud there exists latency because the processing of data is happening far away from the user.
 - With edge devices, the processing of data happen on devices that are located closer to the user thereby improving performance.

SOFTWARE USED

Chameleon Cloud Site

- An NSF-funded cloud testbed primarily used for computer science experimentation and is designed to be reconfigurable.
- Provides a Python library that allows users to work with the testbed and Jupyter notebooks.
- We utilized CHI@TACC for the cloud site, CHI@Edge for the edge site, and a Jupyter notebook.

Pegasus Workflow Management System

- A workflow management system
 - *“an abstraction used by scientists to express an ensemble of complex, computational operations”*
- Scientific workflows allow users to execute computational tasks that often have multiple steps. Pegasus is responsible for managing the execution of the workflow.

HTCondor

- HTCondor systems allow for the submission of many jobs and are able to handle large amounts of work and computation without any user intervention.

APPROACH

Cloud Site

- Located at CHI@TACC site
- Used Chameleon's API to create a lease and start a virtual machine with a floating ip address
- Installed and configured Pegasus and HTCondor, set the submit, execute, and central manager machines
- Performed checks to ensure the software was configured correctly

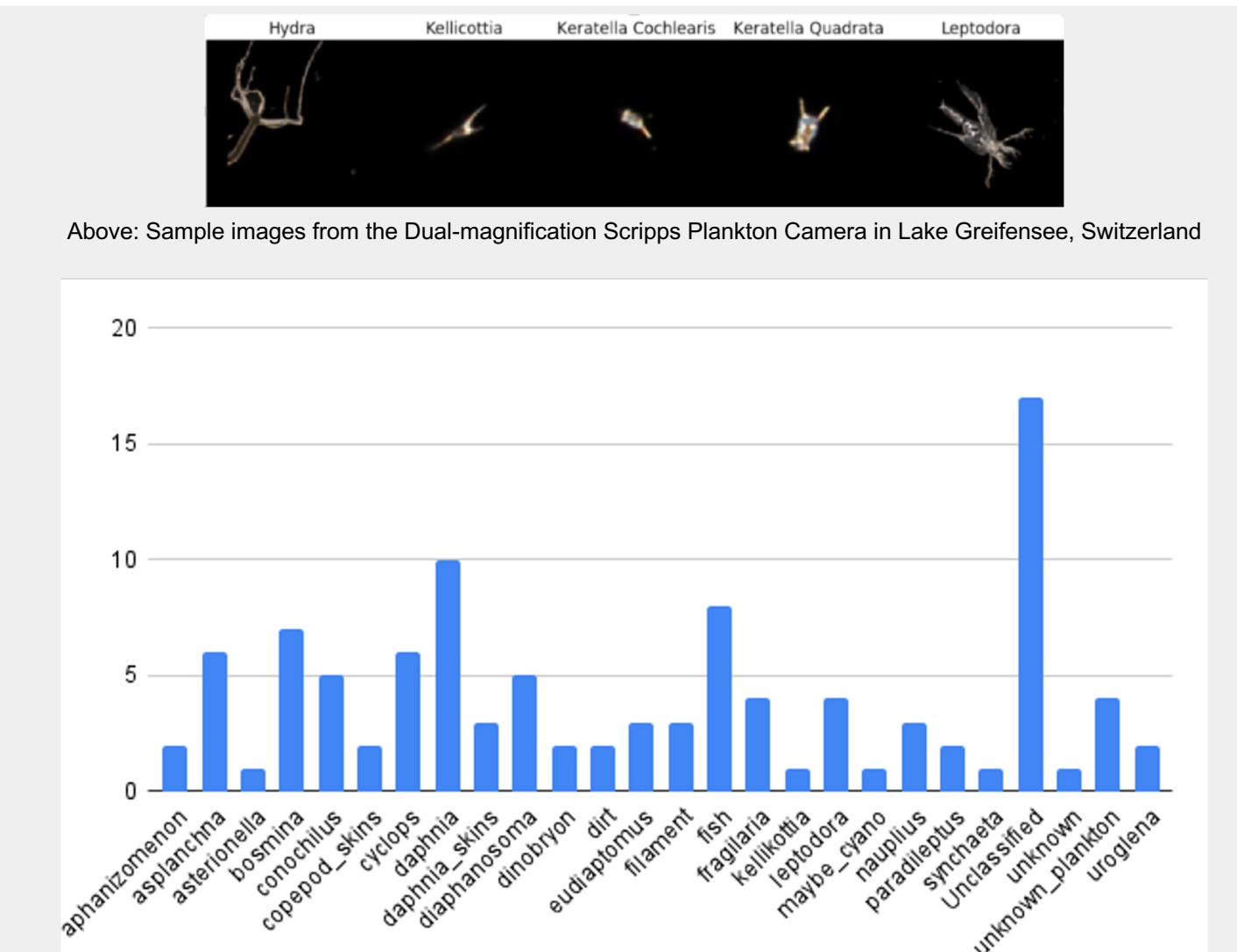
Edge Site

- Moved to the CHI@Edge site
- Installed and configured HTCondor via a container running on one Raspberry Pi [2]
- Performed checks by using the command `condor_status` to ensure there were 97 spots

Workflow and Execution

- From bash scripts we converted them into a workflow using the Pegasus Python API
- A job encapsulates the executable that runs, allows command line arguments to be passed to the executable at runtime. It also takes input files and produces output files
- For the edge job, we added arguments that were needed and stored everything in an output file
- For the cloud job, we added arguments, and an input and output file
- The input file was the output file created from the edge job
- For each job, we created a transformation catalog used to discover locations of executables that are involved by the jobs in the workflow

RESULTS



CONCLUSION

We found 26 types of zooplankton. This could have been done manually, but it would have taken more time and effort, and would be error-prone. Modern-day technology helps us classify faster. 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.

REFERENCES

- [1] S. Kyathanahally, *et al.* “Data for: Deep learning classification of lake zooplankton,” 2021. [Online]. Available: <https://opendata.eawag.ch/dataset/deep-learning-classification-of-zooplankton-from-lakes>
- [2] Docker image: “rajivmayani/condor8-arm64-plankifier-worker”

ACKNOWLEDGEMENTS

This project was supported by the National Science Foundation Grant no: [2127548](#) .
We would like to thank the Chameleon project for providing their support and expertise for this project.

