Localizing CyberGIS-Compute through Containers

Mit Kotak
mitak2@illinois.edu
University of Illinois Urbana-Champaign
Urbana, Illinois, US

Zimo Xiao
zimox2@illinois.edu
University of Illinois Urbana-Champaign
Urbana, Illinois, US

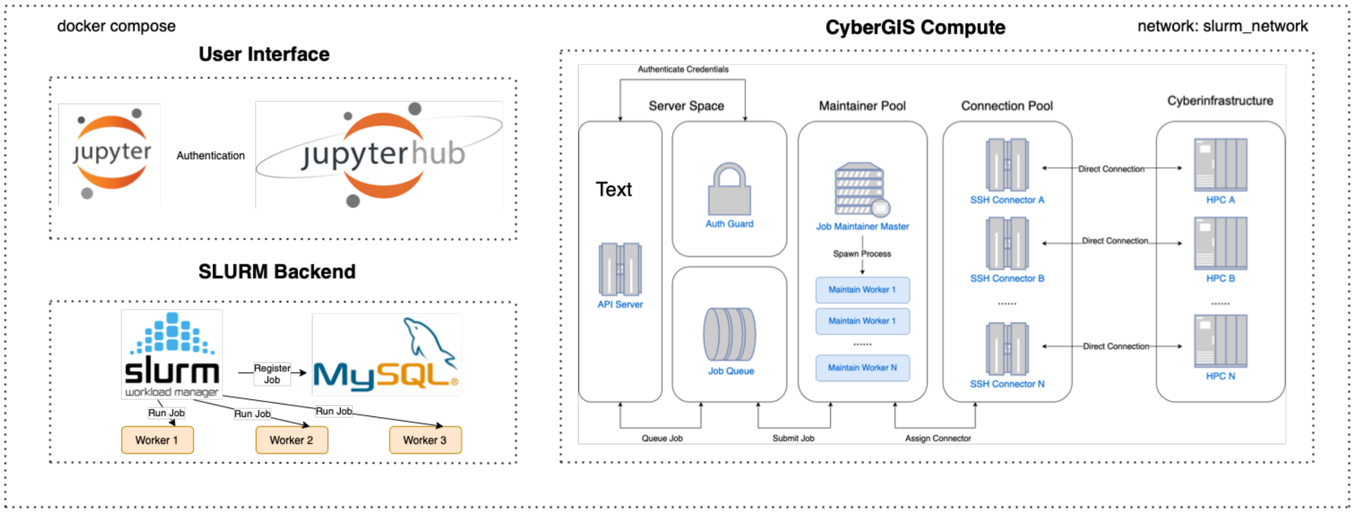

Figure 1: Overview of Local CyberGIS-Compute Container

ABSTRACT

In recent years there has been a push towards solving complex geospatial problems using advanced cyberinfrastructure (CI) systems, broadly defined as CyberGIS. While this allows the geosciences to tackler larger and more complex problems, adoption of cyberGIS techniques are slowed by the technical barriers associated with CI. Within the CyberGIS ecosystem, CyberGIS-Compute has emerged as a promising middleware for bridging access to High Performance Computing Resources (HPC) by providing a simple, easy-to-use Jupyter interface. However, its integration with external HPC resources makes it difficult for model contributors and developers to develop, debug, and test. We present a self contained Dockerized framework that can independently run the full CyberGIS-Compute stack on local machines. This streamlines the development process for model contributors and developers in the CyberGIS-Compute community.

CCS CONCEPTS

• **Applied computing → Earth and atmospheric sciences**; • **Information systems → Geographic information systems**.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

Containers, HPC, CyberGIS

ACM Reference Format:
Mit Kotak and Zimo Xiao. 2022. Localizing CyberGIS-Compute through Containers. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

CyberGIS, defined as geographic information science and systems based on advanced cyberinfrastructure (CI) (e.g., High Performance Computing (HPC) resources), has transformed how large-scale geospatial problem solving is conducted [7]. However, leveraging cybergis capabilities and CI resources is challenging, both due to the steep learning curve. CyberGIS-Compute [4] has emerged as a promising middleware that provides access to HPC resources through CyberGIS-Jupyter[5], an easy-to-use Jupyter interface. Applications of CyberGIS-Compute involve remote sensing data fusion [2], spatial accessibility [1], and hydrological modeling.

CyberGIS-Compute also provides a unique model contribution mechanism that allows community members to easily share their models with the users. This not only ensures reproducibility, but also compresses the model's complexity into a simple user interface where the model contributor can preconfigure the user-facing parameters. Model contributors package their models as Github repositories with metadata describing how CyberGIS-Compute should run the model. While this mechanism greatly simplifies the process of using these models, model contributors have to still test and debug their models on HPC resources in order to integrate them into the CyberGIS-Compute framework. This is not ideal since HPC resources cannot be securely shared across the model contributor community and often provides limited control over the model environment. The integration with HPC also complicates development and testing of CyberGIS-Compute.

We propose a novel framework that runs CyberGIS-Compute on local machines. This greatly simplifies the development process since model contributors: (a) can quickly configure the dependencies needed for the model by trying out different Singularity images, (b) can ensure that their model is able to pass through different layers of CyberGIS-Compute and successfully run on the desired back-end resource, and (c) can simulate different HPC configurations to see how their model might get parallelized across multiple nodes. We use Docker microservices [3] to create separate layers within the same network, thus providing a self-contained local deployment of all the components of CyberGIS-Compute.

2 ARCHITECTURE

The CyberGIS-Compute architecture can be broadly classified into three components: JupyterHub, CyberGIS-Compute and HPC infrastructure. These components are distributed making it tedious to deploy and debug CyberGIS-Compute, especially for community members without direct access to the infrastructure. On top of that, CyberGIS-Compute consists of multiple layers as shown in Figure 1 which makes it difficult to trace bugs through different stacks.

Our approach involves creating separate Docker containers for all of these services and then running them on the same machine and network. Instead of relying on CyberGIS-Compute logs, model contributors and developers have direct access to docker logs which helps breakdown their workflow into manageable chunks.

The various Docker containers can be grouped into three sections: user interface, Cybergis-Compute and SLURM backend. The user interface consists of a JupyterHub which spawns individual user containers. The user container can be configured to match the user's preferred Jupyter environment whereas the JupyterHub container manages user authentication for CyberGIS-Compute. These settings are be helpful to users who might be interested in setting up CyberGIS-Compute deployments for their research group or department. The Cybergis-Compute library is packaged into a separate container which provides RESTful services. The HPC backend is replaced by a set of SLURM containers [6] which can be configured to match different HPC machine configurations.

We faced two main challenges while creating this framework. The first one was networking. We had to map out all of the different SSH and REST calls that were happening across different services onto a single network without modifying any of these services since we wanted to replicate CyberGIS-Compute's behavior on HPC resources. The next challenge was to install all of the dependencies typically found on HPC machines while ensuring that the deployment was light enough to run on user machines.

3 CONCLUDING DISCUSSION

We envision this framework will empower the CyberGIS-Compute community to take a more active role in its development. Model contributors can integrate this framework into their model testing workflow to ensure long term compatibility with CyberGIS-Compute. Community developers can use this tool to locally test their contributions of new features and submitting bug reports. Lastly, this framework will greatly simplify deploying CyberGIS-Compute for those who wish to support their own instance. This will help realize CyberGIS-Compute's vision towards democratizing access to HPC resources.

ACKNOWLEDGMENTS

The authors would like to thank Alexander Michels, Anand Padmanabhan and Shaowen Wang for providing feedback on the abstract. This material is based upon work supported by the Institute for Geospatial Understanding through an Integrative Discovery Environment (I-GUIDE) that is supported by the National Science Foundation (NSF) under award No. 2118329. The material is also based in part upon work supported by NSF under grant No. 1833225. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF. Our computational work used Virtual ROGER, which is a geospatial supercomputer supported by the CyberGIS center for Advanced Digital and Spatial Studies and the School of Earth, Society and Environment at the University of Illinois Urbana-Champaign.

REFERENCES

[1] Jeon-Young Kang, Alexander Michels, Fangzheng Lyu, Shaohua Wang, Nelson Agbodo, Vincent L Freeman, and Shaowen Wang. 2020. Rapidly Measuring Spatial Accessibility of COVID-19 Healthcare Resources: A Case Study of Illinois, USA. *International journal of health geographics* 19, 1 (2020), 1–17.

[2] Fangzheng Lyu, Zijun Yang, Zimo Xiao, Chunyuan Diao, Jinwoo Park, and Shaowen Wang. 2022. CyberGIS for Scalable Remote Sensing Data Fusion. In *Practice and Experience in Advanced Research Computing (PEARC '22)*. Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/3491418.3535145

[3] Dirk Merkel et al. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux j* 239, 2 (2014), 2.

[4] Anand Padmanabhan, Zimo Xiao, Rebecca Vandewalle, Furqan Baig, Alexander Michels, Zhiyu Li, and Shaowen Wang. 2021. CyberGIS-Compute for Enabling Computationally Intensive Geospatial Research. In *SpatialAPI'21: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on APIs and Libraries for Geospatial Data Science*. https://doi.org/10.1145/3486189.3490017

[5] Anand Padmanabhan, Zimo Xiao, Rebecca Vandewalle, Alexander Michels, and Shaowen Wang. 2021. Enabling Computationally Intensive Geospatial Research on CyberGIS-Jupyter with CyberGIS-Compute. In *Proceedings of Gateways 2021*. Zenodo. https://doi.org/10.5281/zenodo.5570056

[6] Giovanni Torres. 2013. Project Title. https://github.com/giovtorres/slurm-docker-cluster.

[7] Shaowen Wang. 2010. A CyberGIS Framework for the Synthesis of Cyberinfrastructure, GIS, and Spatial Analysis. *Annals of the Association of American Geographers* 100, 3 (June 2010), 535–557. https://doi.org/10.1080/00045601003791243