Department: Head
Editor: Name, xxxx@email

Scalable Scientific Interactive Research Computing with Project Scinco

J. Stubbs
Texas Advanced Computing Center, Austin, TX, USA

A. Jamthe
Texas Advanced Computing Center, Austin, TX, USA

N. Freeman
Texas Advanced Computing Center, Austin, TX, USA

M. Packard
Texas Advanced Computing Center, Austin, TX, USA

G. Curbelo
Texas Advanced Computing Center, Austin, TX, USA

C. Hammock
Texas Advanced Computing Center, Austin, TX, USA

***Abstract*—Interactive computing with Jupyter notebooks has transformed the state-of-the-art of scientific research computing. Users can perform a multitude of computational tasks in real time, including data cleansing, analysis, visualization, and post-processing, with Jupyter notebooks. Additionally, the ability to write and execute code and include supporting text and images in the same document has made it popular for use in scholarly articles and teaching. These capabilities complement the batch computing services provided by HPC centers exposed through traditional science gateways. However, integrating Jupyter into science gateways and other advanced computing ecosystems introduces new challenges related to scalability, collaboration, and reproducibility. In this paper, we discuss Project Scinco (Scalable Interactive Computing), an open-source platform for scalable, reproducible, interactive scientific computing, designed to be run in academic computing centers and incorporated into science gateways. We describe the prime features, architecture, deployment choices, and challenges of project Scinco.**

Traditionally, web applications or science gateways have enabled users to run analyses asynchronously on remote systems. The Science Gateways Community Institute's Software Catalog lists over 600 gateway projects [1]. More recently, as the growing number of disciplines bring big data techniques to bear on fundamental problems, interactive computing modes such as Jupyter Notebooks have gained tremendous popularity.

Interactive computing enables users to execute code incrementally and analyze intermediate results before running the next steps in a program. This paradigm simplifies many tasks, particularly data analysis, which is often exploratory in nature. With Jupyter notebooks, users can create interactive documents that contain both code and text, thus allowing for *literate computing*. Providing explanations of program logic next to executable code and the generated results can make the analyses in scholarly articles more understandable and reproducible. Additionally, individuals can write Jupyter notebooks for classroom learning or self-paced tutorials.

These interactive computing tools complement the traditional batch computing services provided by academic computing facilities and HPC centers, where programs are submitted to a scheduler and executed in the future. While the utility and popularity of projects such as Jupyter necessitate their inclusion into research computing ecosystems, science gateways face several challenges incorporating these tools into their environment, including 1) scalability challenges related to a fundamentally different resource utilization profile; 2) identity, authorization and security challenges that impact collaboration; and 3) challenges related to the reproducibility of the computations contained within the interactive computing environment itself.

Project Scinco (Scalable Interactive Computing) aims to tackle these challenges by providing the national research community with a hosted, secured, scalable Jupyter service that augments and complements the batch computing capabilities that are provided by HPC centers and is made available within science gateways. Scinco builds on several open-source technologies, including Docker, Kubernetes, JupyterHub, and Tapis [2] to deliver a platform capable of supporting thousands of users and dozens of projects running across hundreds of servers. Scinco provides a flexible authentication and authorization module based on Tapis and OAuth2, allowing academic institutions to use their local identity system. Scinco integrates with advanced storage and computing resources available in existing science gateways, allowing users to interactively compute on the same personal and community datasets used in batch compute jobs. Finally, Scinco provides a powerful administrative portal where project owners and their delegates can configure many of the run-time aspects of the Jupyter service.

The primary contributions of this paper are as follows:

- We provide a high-level, qualitative description of the requirements and desirable attributes of a scalable interactive computing system which could integrate into and complement the batch computing capabilities provided by science gateways.
- We provide architecture and implementation details of the Scinco project, and show how Scinco enables interactive computing capabilities to be integrated into gateways platforms, thus solving the three primary challenges faced by science gateways when integrating interactive computing listed above. With the architectural and design details of Scinco, science gateways can either integrate with Scinco directly or build their own JupyterHub instances, leveraging similar ideas as described in this paper.
- We describe some existing use cases of the Scinco project, establishing its viability in a number of research domains.

In the eight years since the original system was developed, thousands of researchers have authored tens of thousands of Python notebooks on Scinco as part of research efforts across a broad range of science and engineering disciplines, including astronomy, machine learning, neuroscience, oceanography, synthetic design and more. Numerous university-level courses across several institutions have been taught using Scinco, helping to train thousands of students making up the next generation work force and guaranteeing

a lasting impact for the project.

SCINCO USE CASES

Scinco builds upon the experience and lessons learned from developing a custom JupyterHub at the Texas Advanced Computing Center beginning in 2015. Today, high-profile projects such as DesignSafe-CI [3] and the Hobby Eberly Telescope Dark Energy Experiment (HETDEX) [4] depend on Scinco for interactive computing functionality.

As part of the initial release of the DesignSafe Natural Hazards Engineering Cyberinfrastructure project in 2015, TACC deployed a customized version of JupyterHub to provide users with interactive computing capabilities that integrated with and enhanced the capabilities of the core DesignSafe-CI portal. To date, DesignSafe-CI researchers leverage JupyterHub at TACC to easily share, manage, and access their critical natural hazards datasets and computational workflows.

Scinco's JupyterHub instance for project HETDEX, the first major experiment to search for dark energy, is extensively used. HETDEX leverages a giant Hobby-Ebberly Telescope at McDonald Observatory and a set of spectrographs to map the three-dimensional positions of millions of galaxies. Scinco offers a quick and easy way for HETDEX to access a complex data model. Many visualizations and interactive widgets used for data analysis for HETDEX are developed using Scinco.

Several other projects including the Synergistic Discovery and Design Environment (SD2E), Estimating the Circulation and Climate of the Ocean (ECCO), and 3-D Electron Microscopy have used Scinco in the recent past.

BACKGROUND

Interactive Computing: State of the Art

Interactive computing with Jupyter has gained popularity among data analysts, scientists, machine learning developers, instructors, and students. It has become the de facto standard for teaching computer science courses across numerous universities and institutions. The total number of Jupyter notebooks available in public repositories on GitHub has increased exponentially over the last several years, and at the time of this writing, there are more than 8.6 million Jupyter notebooks in such repositories [5].

A number of academic and industrial projects offer web-based, interactive computing capabilities. These include Pangeo cloud for geoscience research, MyBinder, Google Colaboratory, Kaggle, etc. Various competitions hosted on Kaggle provide solutions to challenging real-world problems in the form of Jupyter Notebooks. However, most web-based interactive computing platforms, particularly those developed by commercial cloud companies, tend to stand alone and lack easy integration into science gateways and academic HPC ecosystems.

Gateway Integration Challenges

Although interactive computing tools such as Jupyter notebooks have become extremely popular due to the ease with which one can perform a range of analysis tasks in real-time, science gateways must address several challenges to incorporate the technology into their platforms.

First, resource utilization in interactive computing differs fundamentally from that of traditional batch computing where the goal is to maximize the utilization of the available CPU. With interactive computing, investigators execute bits of code and then analyze the results before proceeding. This usage mode produces spikes of CPU usage in between long idle periods. At the same time, unlike batch compute jobs, users expect notebook servers to start up quickly and to be available throughout the workday. As a result, JupyterHub requires entirely different resource utilization and scaling properties to those available to gateways from traditional HPC centers.

Secondly, academic computing facilities and science gateways rely on identity and access management systems for authentication and to authorize users for access to data and compute resources. Ideally, an interactive computing platform would easily integrate with these existing systems to allow the same authorization schemes to be applied.

Finally, given the prominent use of notebooks for data analysis, an interactive computing resource must integrate into the same storage resources made available to users via the academic facility or science gateway. However, such an integration is a nontrivial task, requiring a com-

mon identity system as previously mentioned, but also requiring the interactive computing service to provide direct access to data the way that an HPC system does for a batch job.

Reproducibility Challenges

While Jupyter notebooks simplify some aspects of the computational reproducibility challenges, they also suffer from similar issues inherent in all software and, to some extent, introduce new problems. Like all programs, code executed in a notebook often depends on specific versions of programming languages, libraries and other software packages. Unless the notebook service makes this larger computing environment available with the notebook, computational reproducibility will be hard to achieve.

Additionally, notebooks commonly depend critically on datasets contained in files that are separate from the notebook document itself. To reproduce results contained within a notebook, the same datasets must be made available. Tracking whether the datasets are available in each notebook becomes challenging for the science gateway or the resource provider.

In this paper, we will discuss how Scinco addresses these issues to provide an open-source, production-grade, scalable interactive computing solution integrated deeply with an academic cyberinfrastructure ecosystem.

SCINCO ARCHITECTURE

Scinco architecture utilizes a customized JupyterHub, an open-source, cloud-based Jupyter project that allows users to access notebook servers running on remote machines. Scinco incorporates *multitenancy* or logically separated instances of the platform so that individual projects can configure various aspects of the notebook servers and data made available to the users based on their individual project needs. The basic subsystems of Scinco are as shown in Figure 1.

Notebook Server and JupyterHub

Jupyter notebook server implements a client-server architecture in which a user makes HTTP requests via their browser to a server, which could be running on a separate machine. JupyterHub provides user notebook servers as-a-service by managing authentication and user accounts and coordinating a single-user notebook server spawning on-demand. The authentication and notebook spawning mechanisms of JupyterHub can be customized via a plugin's architecture.

OAuth-based Authenticator Plugin

Scinco implements a custom JupyterHub *authenticator* plugin to utilize the Tapis OAuth2-based authentication module. This flexible framework allows projects to configure various types of identity stores, including LDAP, Active Directory, third-party OAuth solutions such as Google and GitHub, and even identity federations such as Globus Auth. Scinco's implementation builds on top of Tapis with an additional function, which can be customized per tenant, to assign each user a unique Unix UID and one more of the GIDs.

Custom Kubespawner Plugin

Once authenticated, the *spawner* plugin launches a notebook server for the user on the available resources. Scinco's spawner builds on top of the open-source Kubespawner project to launch notebook servers as Docker containers on a Kubernetes cluster. Unlike batch schedulers, Kubernetes launches containers instantly and scales to hundreds of nodes. Scinco's spawner executes the container using the UID and GIDs associated with the user described previously, ensuring that permissions on shared file systems will be honored by all actions taken by the user within the notebook server container.

Kubespawner Hooks

Scinco further extends the community Kubespawner with additional notable features by implementing two "hooks" as shown in 1. The first and most notable hook supports the Spawner Options screen, which allows users to select different container images for their notebook servers. These images come pre-loaded with different types of software (e.g., libraries for astronomy, biology, civil engineering, machine learning, etc.) and are versioned, allowing Scinco to maintain support for older notebooks while simultaneously being able to provide access to the latest versions of libraries.

To support the Spawner Options screen, the Scinco hook determines the images available to the user by retrieving its "dynamic configuration";

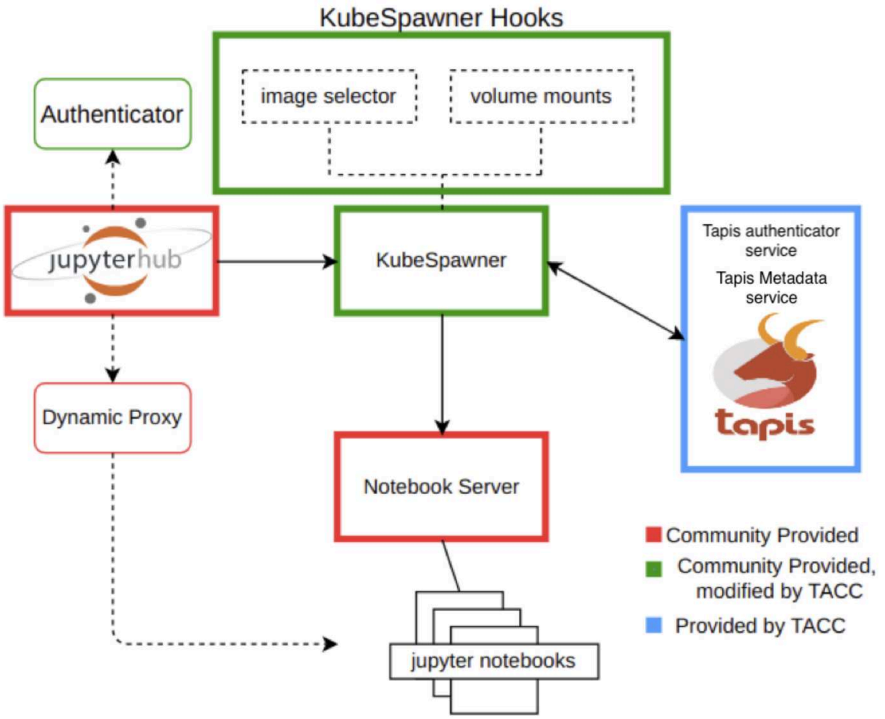

Figure 1. Scinco Architecture

i.e., JSON data stored with the Tapis Metadata API describing various configurable run-time aspects of Scinco. Each Scinco project (or "tenant") maintains its own dynamic configuration, which can be updated at any time via a simple Tapis API call. Because the Scinco spawner retrieves its metadata before spawning each notebook server, the latest metadata configuration gets applied each time a user starts a new server. Therefore, users see changes to the configuration by simply stopping and restarting their notebook server; in general, redeployment of the cluster is not necessary.

Another hook implemented in the Scinco spawner retrieves a list of remote data sources available to the authenticated user. Each data source corresponds to a directory on a file system available via a network protocol (e.g., NFS, Luster, etc.) as well as an access level – either read-only or read-write. The Scinco spawner mounts each data source into the user's container using Kubernetes persistent volume claims (PVCs) as it gets spawned at the access level indicated.

An Administrative portal developed recently supports modifying some of these image and storage configurations on-the-fly by project admins, which we will discuss in detail in the later sections.

DEPLOYMENT

The Scinco design builds on top of the Kubernetes orchestration platform and utilizes a customized Kubespawner plugin as discussed in the previous section. A Scinco instance can be deployed to Kubernetes within a single unprivileged namespace, using a few YAML files that define Kubernetes Deployment and Service objects. One of the primary ways to deploy these to the cluster is the `kubectl apply -f *.yml` method.

Scinco itself has no persistent storage requirements within Kubernetes. The JupyterHub server configuration is handled during deployment using the YAML files. The additional configurations that control aspects such as the available container images for notebook servers, the amount of memory and CPU resources available to each user and the available file mounts for each user are managed using the Tapis Metadata service. Each Scinco instance must be configured with a Tapis tenant that has been set up with the Scinco

metadata schema and permissions. Each Scinco instance identifies a set of project admins that are authorized to modify this metadata as per their project requirements, using either the Scinco Admin Portal or by making API calls to Tapis directly.

Deployment Considerations

Sizing the cluster resources can be a challenge. Variables like the total number of users, amount of memory/CPU per user, and variances in use between users may not be known at the time of deployment. Total CPU and RAM requirements depend entirely on the nature of work the users are doing in the notebooks. For example, projects performing data curation or visualization tend to consume significantly fewer resources than projects training and utilizing heavy ML/AI models.

As of today, TACC Scinco Kubernetes clusters contains approximately 220 CPU cores and 1,200 GB of memory across 20 servers. The use of dynamic metadata and Kubernetes allows the cluster to be scaled and modified as resource needs change.

Kubernetes has an optional metrics server (which must be installed by the cluster administrator) [6] that collects CPU and memory statistics for nodes and pods. CPU usage is measured in milliunits of a CPU core. Scinco administrators and/or Kubernetes cluster administrators can monitor the number of resources consumed by the users using the `kubectl top pod` command, and they can add or remove resources from the cluster to accommodate usage. Administrators may also choose to periodically shut down containers that have been running for a long time and/or appear to be idle.

Persistent Storage for Notebooks

Each user notebook running in the cluster is a single container with no inherent persistent storage. In the event of a container restart, files or data created within the container will be lost permanently. In a Kubernetes environment, there is no assumption of up-time or that the container will continue to run until the user no longer needs it. Therefore it is highly beneficial to have a shared or long-running storage space where users can save their critical work. Two out of several ways to achieve this are described below.

Kubernetes presents objects called Persistent Volume Claims (PVCs), which are data volumes managed by Kubernetes and mounted into containers at run time. Scinco can be configured to use the underlying PVC system to create data volumes for each user and mount them at run time.

Cluster administrators may also have existing shared storage systems (for example, NFS servers). These servers may export directories to the Kubernetes cluster that may mount inside individual user containers. A common way to setup is to have one “work” directory per user, so each user may have their own personal space to store files. In the event of a notebook restart, files within that directory will be retained and remounted on the next notebook server startup.

Another approach is to use the NFS file system and simultaneously present Read-Only data directories to many users. If there is a large dataset that many users wish to analyze, it may be exported simultaneously to every running container in the Scinco instance.

Custom Notebook Images to Cater to Specific Research Use Cases

One of the significant advantages of deploying stand-alone JupyterHub environments for different groups is that we can exercise control over the packages users can access. This paradigm allows us to customize JupyterHub deployments to cater to each scientific use case. Projects within Scinco can also choose to organize users into groups and subgroups. Each group can provide access to different resources, including notebook server images with pre-installed packages, CPU and memory resources, and remote data sources that will appear as mounts within their running notebook servers. Users can choose from the available docker images that best suit their use case and make their experiments more reproducible.

SCINCO FEATURES

CI/CD with Tapis Workflows

Project Scinco enables its projects and individual users to define and deploy their own custom notebook server images. Manually managing the build, verification, and release of notebook images with standard command line utilities

is labor-intensive and time-consuming. Additionally, the large size of the notebook images makes it difficult and cost-ineffective to use traditional continuous integration tools (CI) such as GitHub Actions or GitLab CI. For this reason, this project has chosen to leverage Tapis Workflows as the primary CI technology for its users.

Tapis Workflows is an official Tapis API and workflow executor that enables users to build and execute research computing pipelines in the Tapis ecosystem. Users can configure source control platforms (e.g., GitHub, GitLab, etc.) to send webhook notifications to the Tapis Workflows API to trigger their workflows. This API was initially designed to facilitate continuous integration efforts. Since then it has been adapted to serve more generic computing workflow use cases, enabling users to execute arbitrary code in the language and runtime environment of their choice, run containerized applications, and send HTTP requests. Figure 2 demonstrates a typical workflow for the SCINCO project.

The fundamental unit of execution in a Tapis workflow is called a task. A single image build task is required for continuous integration-based workflows, such as those of the Scinco project. This task is defined in four parts; the image builder, context, destination, and archive.

The *image builders* available with Tapis Workflows are Singularity and Kaniko [7]—a container-based software, which is capable of container-in-container image construction without the need for root access on the host machine, ensuring a more isolated and secure build environment.

The *context* is the origin or source of the image to be built. This could be a DockerHub registry or source code in a GitHub repository. When defining a task with a source code context, users specify the path to the build file—Singularity or Dockerfile—and the path to the build context. If these contexts are private, users can provide the credentials in the task definition. These credentials are stored securely in the Tapis Security Kernel—a role-based authentication and secrets management Tapis service— and only fetched for the Workflow Executor when their workflow is triggered.

The *destination* of an image build task is the specific location of the image, where it will be sent after the build completes. Examples of destinations include image registries such as DockerHub, or local storage in the workflow executor file system. Just as with the context, users can provide the credentials required to push to private registries. Users can also store the resultant Singularity (SIF) or compressed (tar.gz) file in the local file system of the Workflow Executor. However, this is an ephemeral storage solution intended only for storing images and files for the duration of the execution of a workflow. Once the workflow terminates, all artifacts produced during its execution gets deleted. Tapis Workflows allows us to define archives as a permanent storage solution.

Archives are permanent storage mediums to which the results, files, and images produced during the run of a workflow are persisted. Currently, the only supported archive is a Tapis System; a server or set of servers—virtual or physical—that are part of a Tapis deployment.

Reproducibility Features

To ensure that the results computed in Jupyter notebooks created on the Scinco platform are reproducible, the project implemented an on-save hook mechanism that stores two custom attributes within an individual notebook file's metadata the first time it gets saved. The first attribute contains the *container image name* and *tag* used for the notebook server in which the notebook file gets saved. Scinco injects the image name and a tag as environment variables at the notebook server startup. The on-save hook reads them and saves them to the notebook file metadata. With this attribute saved in the notebook file itself, a future process – including one independent of Scinco – can determine the correct container image to use to open the file. The second attribute is a unique identifier attached to a notebook file when created. We use a UUID v4 for the identifier. The UUID attribute gets created when one does not already exist, providing a way to track which notebooks are copies of other notebooks. If a notebook is a copy, it will have the same UUID.

Administrative Portal

Supporting five JupyterHub clusters with more than 1600 total users running at TACC has required about two to three full-time devel-

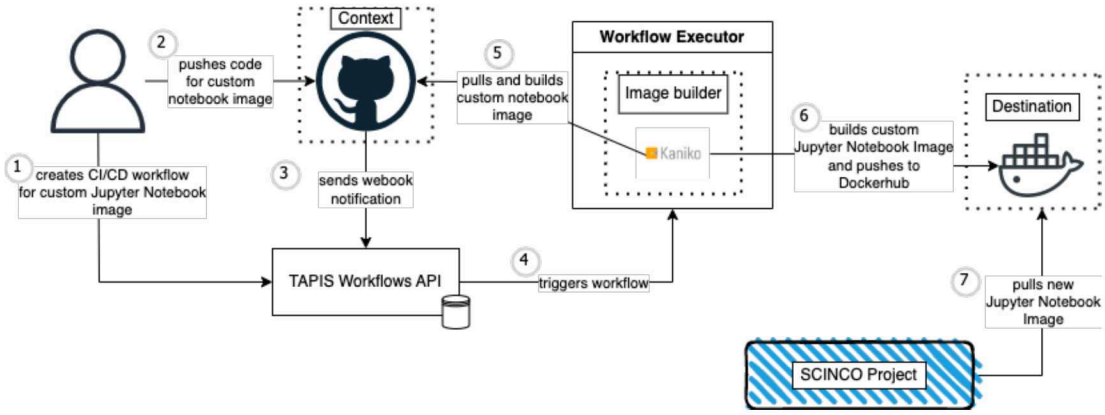

Figure 2. Tapis Workflows and SCINCO

opers, working directly with the project teams, to define requirements, configure options and preferences, and evolve the Jupyter cluster to meet the needs of each project. In order to make this process scale to hundreds of independent projects, with project Scinco, we developed an administrative portal that is a one-stop shop for all configuration, reporting, and administrative matters. The administrative portal empowers administrators from each project to manage their own JupyterHub clusters without direct support from TACC's Jupyter staff. The Scinco administrator portal allows project (i.e., "tenant") administrators to dynamically modify user and identity management configuration. They can customize resource utilization on a per-project/user basis by submitting basic HTML forms, which in turn make corresponding POST requests to the Tapis Metadata service, where the project metadata is stored as a MongoDB document. They can create different user groups and allow particular notebook images to be used per user group.

Within the portal, project administrators can configure and manage the available storage locations and the corresponding path names that will appear in the notebook server, such as "MyData", "MyProjects", "PublishedData", "CommunityData", etc. Users can be assigned to specific groups and associated with specific GIDs as well. At the time of starting the notebook server, Scinco will use these configurations to determine which directories to mount for a user notebook server. Additionally, the admin portal provides a graphical user interface for managing the available notebook images.

Metrics and Monitoring

Scinco supports monitoring metrics such as CPU utilization and memory usage for individual users, as illustrated in Figure 3. Scinco leverages the following open-source projects for its monitoring functionality: Prometheus [8], a time series database to scrape the instantaneous CPU and memory usage of each notebook server; Thanos [9] to provide long-term storage for metrics collected by Prometheus; Grafana [10] for visualizing the monitoring data in a dashboard and cAdvisor [11], a container advisor that provides metrics for resource usage and performance of the running containers. Scinco also generates high-level project usage data such as how many unique users have logged in during a specified time window, median notebook server launch time and run time, total daily user count, etc. These performance and usage metrics are good indicators of system reliability and can be leveraged by project administrators to generate annual reports for their projects. A new feature to generate and email automated weekly usage reports to project administrators is implemented receently and will be available to all the Scinco admins in early Summer 2023.

CONCLUSION

Jupyter notebooks enable investigators to efficiently tackle tasks common to computational research projects, compelling academic computing facilities and science gateways to incorporate them into their offerings. Yet, integrating Jupyter into existing ecosystems presents challenges with respect to scalability, collaboration and repro-

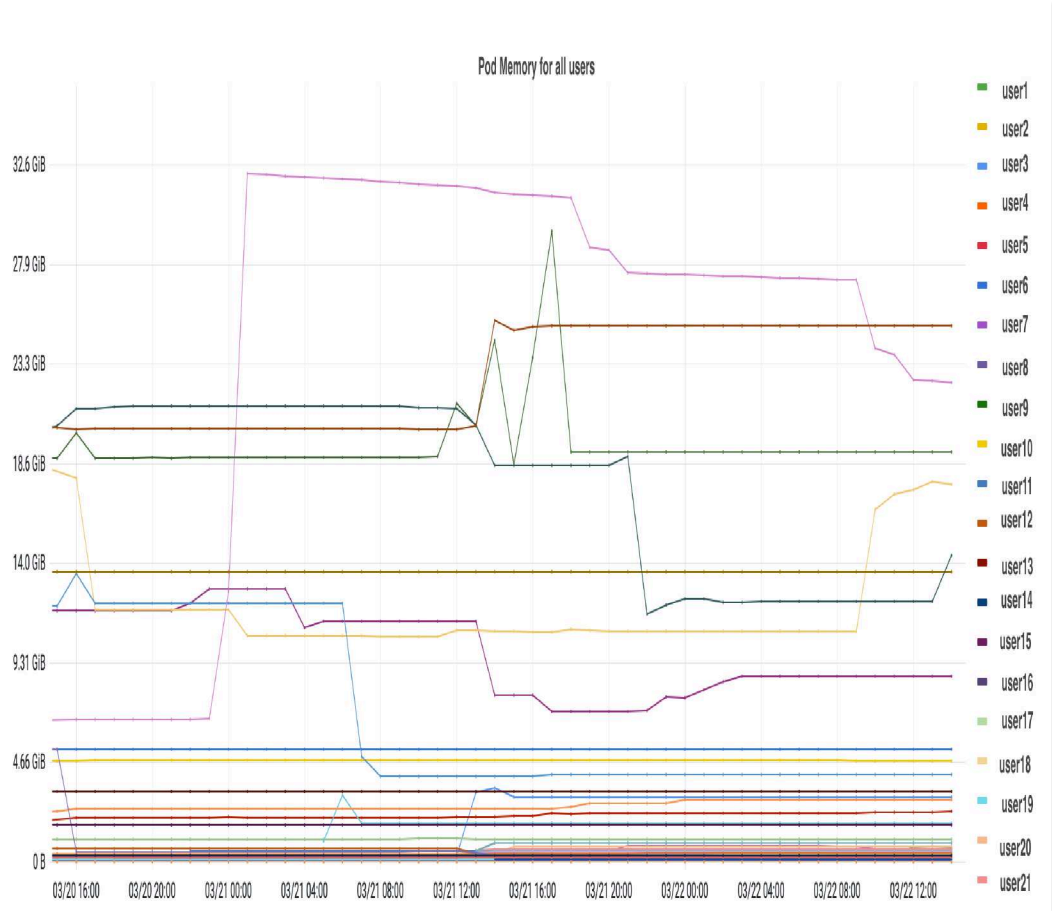

Figure 3. Memory utilization per user

ducibility. Project Scinco addresses these challenges by providing an interactive computing platform based on open-source technologies such as JupyterHub, Docker, Kubernetes and Tapis. Scinco is utilized by several major cyberinfrastructuxre projects spanning a number of scientific domains. Scinco staff at TACC is working on exciting features for usage metrics reporting that came in as a requirement from project administrators. An update to Scinco's JupyterHub was recently completed to provide users with JupyterHub 3.0 version and keep up with the JupyterHub releases.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation Office of Advanced CyberInfrastructure, the Tapis Framework:[1931439 and 1931575]

REFERENCES

1. "SCGI Software Catalog," [Online]. Available: https://catalog.sciencegateways.org/#/home. [Accessed: 10-Dec-2022].
2. J. Stubbs et al., "Tapis: An API Platform for Reproducible, Distributed Computational Research," In: Arai, K. (eds) Advances in Information and Communication. FICC 2021. Advances in Intelligent Systems and Computing, vol. 1363. Springer, Cham. https://doi.org/10.1007/978-3-030-73100-7_61
3. E.M. Rathje et al.," DesignSafe: New cyberinfrastructure for natural hazards engineering," Natural Hazards Review, 18(3), 06017001.https://doi.org/10.1061/(asce)nh.1527-6996.0000246
[Online]. Available: https://govtribe.com/opportunity/federal-contract-opportunity/shade-darpapa210403 [Accessed: 20-Jun-2022].
4. "Hobby Ebberly Telescope Dark Energy Experiment," [Online]. Available: https://hetdex.org [Accessed: 20-Jun-

2022].

5. Parente P., “Estimate of Public Jupyter Notebooks on GitHub,” [Online]. Available: https://nbviewer.jupyter.org/github/parente/nbestimate /blob/master/estimate.ipynb. [Accessed: 20-Jun-2022]
6. “Metrics For Kubernetes System Components,” [Online]. Available: https://kubernetes.io/docs/concepts/cluster-administration/system-metrics/. [Accessed: 2-Dec-2022].
7. “Kaniko: Containerized Image Builder, ” [Online]. Available: https://github.com/GoogleContainerTools/kaniko [Accessed: 29-Nov-2022]
8. “Prometheus,” [Online]. Available: https://prometheus.io. [Accessed: 20-Jun-2022].
9. “Thanos,” [Online]. Available: https://thanos.io. [Accessed: 20-Jun-2022].
10. “Grafana,” [Online]. Available: https://grafana.com [Accessed: 20-Jun-2022].
11. “Container Advisor,” [Online]. Available: https://github.com/google/cadvisor [Accessed: 20-Jun-2022].

Dr. Joe Stubbs is a Research Associate and leads the Cloud and Interactive Computing (CIC) group at the Texas Advanced Computing Center at the University of Texas at Austin. Dr. Stubbs is the Principal Investigator of two NSF-funded projects- Tapis and Abaco and has played a fundamental role in developing numerous national-scale cyberinfrastructure systems for various scientific and engineering communities used by thousands of researchers.

Dr. Anagha Jamthe is a Research/Engineering Scientist Associate in the Cloud and Interactive Computing group (CIC) at the Texas Advanced Computing Center at the University of Texas at Austin. Dr. Jamthe is one of the technical contributors to the NSF funded Tapis Project and manages the JupyterHub project at TACC.

Nathan Freeman is an Engineering Scientist Associate in the the Cloud and Interactive Computing (CIC) group at the Texas Advanced Computing Center at the University of Texas at Austin. Nathan manages the development of the Tapis Workflows API and related services, libraries, and UI.

Mike Packard is a System Administrator at the Texas Advanced Computing Center at the University of Texas at Austin. He provides devops and automation support for several cloud infrastructure and research projects.

Gilbert Curbelo is a Software Developer at the Texas Advanced Computing Center at the University of Texas at Austin. Gilbert is in charge of updating different components of the JupyterHub project and contributes to the Tapis project.

Cody Hammock is a System Administrator at the Texas Advanced Computing Center at the University of Texas at Austin. He provides monitoring support for the Scinco Project, and manages or contributes to several cloud infrastructure projects at TACC.