Common Resource Descriptions for Interoperable Gateway Cyberinfrastructure

Joe Stubbs
TACC,
University of Texas, Austin
Texas, USA
jstubbs@tacc.utexas.edu

Suresh Marru
Cyberinfrastructure Integration Research Center,
Indiana University
Indiana, USA
smarru@iu.edu

Daniel Mejia
Network for Computational Nanotechnology,
Purdue University
Indiana, USA
dmejiapa@purdue.edu

John-Paul Navarro
University of Chicago,
Argonne National Lab
Illinois, USA
navarro@anl.gov

Eric Franz
The Ohio Supercomputer Center
Ohio, USA
efranz@osc.edu

Steve Black
TACC,
University of Texas, Austin
Texas, USA
scblack@tacc.utexas.edu

Dimuthu Wannipurage
Cyberinfrastructure Integration Research Center,
Indiana University
Indiana, USA
dwannipu@iu.edu

Sudhakar Pamidighantam
Cyberinfrastructure Integration Research Center,
Indiana University
Indiana, USA
pamidigs@iu.edu

Claire Stirm
San Diego Supercomputer Center
University of California, San Diego
California, USA
cstirm@ucsd.edu

Maytal Dahan
TACC
University of Texas, Austin
Texas, USA
maytal@tacc.utexas.edu

Marlon Pierce
Cyberinfrastructure Integration Research Center
Indiana University
Indiana, USA
marpierc@iu.edu

Michael Zentner
San Diego Supercomputer Center,
University of California, San Diego
California, USA
mzentner@ucsd.edu

ABSTRACT

Science gateway projects face challenges utilizing the vast and heterogeneous landscape of powerful cyberinfrastructure available today, and interoperability across technologies remains poor. This interoperability issue leads to myriad problems: inability to bring multiple heterogeneous specialized resources together to solve problems where different resources are optimized for different facets of the problem; inability to choose from multiple resources on-the-fly as needed based on characteristics and available capacity; and ultimately a less than optimal application of nationally-funded resources toward advancing science. This paper presents version 1.0 of the Science Gateways Community Institute (SGCI) Resource Description Specification – a schema providing a common language for describing storage and computing resources utilized by science gateway technologies – as well as an Inventory API and software development kits for incorporating resource definitions into gateway projects. We discuss multiple gateway integration design options, with trade offs regarding robustness and availability. We detail the adoption to date of the SGCI Resource Specification by several prominent projects, including Apache Airavata, HUBzero®, Open OnDemand, Tapis, and XSEDE. The XSEDE adoption is worth highlighting explicitly as it has led to a new API within the XSEDE Information Services architecture which provides SGCI resource descriptions of all active XSEDE resources. Additionally, we show how the use of the SGCI Resource Specification provides interoperability across resource providers and projects that adopt it. Finally, as a proof of concept, we present a multi-step analysis that runs Quantum ESPRESSO and visualizes the energy band structures of a Gallium Arsenide (GaAs) crystal across multiple resource providers including the Halstead cluster at Purdue University and the Stampede2 supercomputer at TACC.

PEARC '21, July 18–22, 2021, Boston, MA, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8292-2/21/07…$15.00
https://doi.org/10.1145/3437359.3465576

CCS CONCEPTS

• **Computer systems organization** → **Distributed architectures**; • **Information systems** → *Computing platforms*; • **Computing methodologies** → *Distributed computing methodologies.*

KEYWORDS

Cyberinfrastructure, interoperability, resource description, science gateways community institute

ACM Reference Format:
Joe Stubbs, Suresh Marru, Daniel Mejia, John-Paul Navarro, Eric Franz, Steve Black, Dimuthu Wannipurage, Sudhakar Pamidighantam, Claire Stirm, Maytal Dahan, Marlon Pierce, and Michael Zentner. 2021. Common Resource Descriptions for Interoperable Gateway Cyberinfrastructure. In *Practice and Experience in Advanced Research Computing (PEARC '21), July 18–22, 2021, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437359.3465576

1 INTRODUCTION

Science gateways [15] provide online interfaces to advanced storage and computing cyberinfrastructure, allowing investigators to more easily utilize these resources for their research. Given the complexity and heterogeneity of cyberinfrastructure today, science gateways are regarded as important tools by thousands of practitioners. Nevertheless, gateway projects face difficult challenges leveraging the various storage and computing resources available, and interoperability across these technologies remains poor. This interoperability problem manifests itself at two levels: on one level, individual science gateways would ideally be able to interchange and integrate multiple advanced systems together, and on another level, researchers could ideally interoperate across multiple science gateways. The lack of interoperability leads to inefficient use of resources: specific components of an analysis initially built to use one resource cannot be redirected to another resource better fit for the task – for example, a resource with more applicable capabilities or one that has more available cycles, etc. Similarly, a lack of interoperability often prevents researchers from conducting analyses on new datasets stored in different resources.

The Science Gateways Community Institute (SGCI) [31] is sponsoring a multi-institutional project to produce a Resource Description specification, together with an Inventory API and client tooling, to provide a common language for describing storage and computing resources. Unlike past efforts to provide common resource descriptions, the SGCI Resource Description specification explicitly targets science gateway projects, focusing on their use cases and the information needed to achieve them. The project adopted an open and inclusive governance model, where all work is hosted in public repositories on GitHub, and the team solicited feedback and contributions from the community at conferences and regularly scheduled meetings. This community driven approach has led to promising results, as several prominent projects have made significant contributions to the specification and have either completed or made substantial strides towards adoption.

This paper announces and presents version 1.0 of the SGCI Resource Description specification, the Inventory API, and associated tooling. The paper also offers initial evidence of the project's viability. We detail the process by which resource descriptions are published to the inventory, and how adoption by the XSEDE project [29] has led to the publication of a complete set of resource descriptions of all currently active XSEDE systems. We describe the different options science gateways and gateway frameworks have for incorporating SGCI resource descriptions into their projects, and we give accounts of the adoptions in the Open OnDemand [9], Tapis [27], and Apache Airavata [18] gateway frameworks. Finally, we describe how utilizing SGCI resource descriptions and the associated tooling leads to data exchange and interoperability across cyberinfrastructure, and we walk through a proof-of-concept Jupyter notebook [14] that computes and visualizes the energy band structures of a Gallium Arsenide (GaAs) crystal by executing a multi-step Quantum ESPRESSO analysis across different geographically distributed resources.

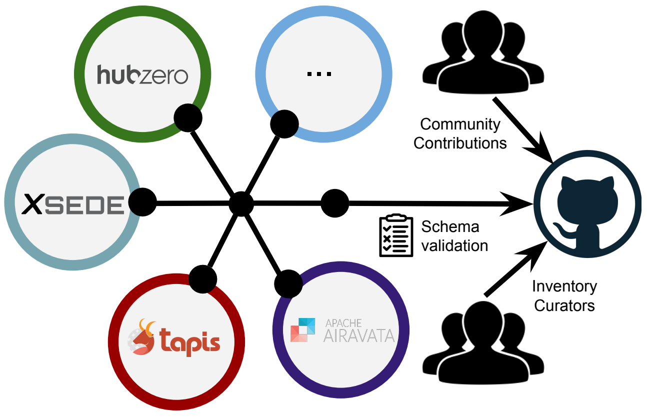

Figure 1: Overview of SGCI Resource Inventory with planned integrations with Apache Airavata, HUBzero®, Tapis frameworks and XSEDE.

In summary, the primary contributions of this paper are:

- An overview of version 1.0 of the SGCI Resource Description specification and mechanisms leading to resource description publication, including a complete set of resource definitions for all active XSEDE systems.
- An overview of the Inventory API and language SDKs as well as the different design options available to science gateways for integrating SGCI resource descriptions into their projects. Details of the adoption to date in the Airavata, Open OnDemand and Tapis systems are also included.
- A description of how the adoption of the specification leads to interoperability and a proof of concept demonstration of a multi-step Quantum ESPRESSO analysis packaged as a Jupyter notebook that runs across different geographically distributed resources.

2 SGCI RESOURCE DESCRIPTION SCHEMA

2.1 Overview

The SGCI Resource Specification is implemented as a JSON Schema document describing the required and optional attributes available for describing resources. Each resource description is itself a JSON document conforming to the SGCI schema. Using JSON Schema and the associated ecosystem of tooling, one can automate tasks related to JSON data, including validation and deserialization, in all major programming languages. For example, all documentation and examples are automatically updated on readthedocs using the sphinx-jsonschema [11] as new changes are pulled to GitHub. In

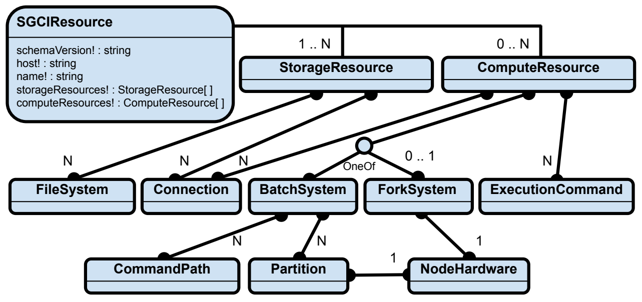

Figure 2: Overview architecture of the SGCI Resource Description specification schema

conjunction with the prominence of JSON data and APIs across the modern web, this ecosystem made JSON Schema a compelling choice for the SGCI Resource Descriptions project.

Fundamentally, a resource represents a logical collection of storage and/or computing capabilities made available at a primary network address and, optionally, one or more secondary addresses. Examples of resources include nationally funded supercomputers such as Bridges, Comet, or Stampede2, as well as campus clusters, file servers large and small, and cloud computing resources such as object stores and virtual machines. Resource descriptions include attributes of a resource that apply to all or most users and exclude attributes specific to individual users or groups, such as accounting information (e.g., allocation numbers) or identity information. In particular, the resource definition is publicly available and therefore cannot include security sensitive information.

Every resource definition must include a *host* attribute whose value provides the primary network address of the resource; for example, the host attribute for the Stampede2 supercomputer has the value "stampede2.tacc.utexas.edu". An essential design in the schema is that the host attribute must be unique across the inventory: there can only be one resource definition with a given host value. This uniqueness property guarantees that all clients utilizing a particular resource have the same definition of the resource's properties and is a key ingredient in solving the interoperability problem.

The schema organizes a resource's capabilities into storageObjects and computeObjects. Details on these objects are provided in the subsequent sections.

2.2 Storage Resource Objects

A storage capability is described in a storageObject whose primary attributes are a storageType, one or more connectionObjects, and a fileSystem object. Version 1.0 of the schema supports describing POSIX, S3 and iRODS storage capabilities, and the storageType captures which category of storage is being described. Multiple storageObjects can be included in a single resource definition to allow for systems that provide multiple storage capabilities, such as the Corral storage system at TACC, which has both POSIX and S3-compatible APIs. The required connectionObject defines the protocols used to connect to and authenticate with the resource and additional information such as the port to use when connecting. It is worth noting that the serviceHost can optionally be supplied within the connectionObject to specify a different network address than the primary host attribute to use for a particular capability. This feature allows for use cases such as where data transfer capabilities attached to computing resources are accessed via a separate network address, as can be found on many HPC systems. Finally, one or more fileSystem attributes optionally included to describe paths for different storage areas, such as "scratch", "work" and "home" commonly found on HPC systems.

2.3 Compute Resource Objects

Compute definitions that define a batchSystemDefinition must specify the jobManager, such as SLURM, SGE or PBS and may specify the scheduler host, commandPaths to various scheduler client binaries, and hardwareProfiles which allows defining the different ways jobs can request compute resources. Many HPC centers use scheduler partitions or queues, while other sites may employ constraints, node features, or other scheduler submission arguments to partition compute resource requests. By optionally supplying the submitArgs array within the hardwareProfiles object the minimum required submission arguments for that compute resource partition may be specified. Features of the nodes provided by the hardwareProfile are defined using the totalNodes property and nodeHardware object with properties such as gpuCount, memorySize and cpuCount. The computeQuotas object is optionally included to define the restrictions on jobs submitted using this hardwareProfile, such as minMemoryPerJob and maxMemoryPerJob. Restrictions like these enable addressing use cases like at OSC where when requesting a large memory node the job must specify the memory request between 363GB and 744GB.

2.4 Related Work

Computational resource discovery was a topic of widespread interest during the Grid Computing era. The Open Grid Forum (OGF) pioneered many standards including the Resource descriptions. The Distributed Resource Management Application API (DRMAA) [30] defined a generalized API to Distributed Resource Management (DRM) systems in order to facilitate the development of portable application programs and high-level libraries. The Job Submission Description Language (JSDL) [5] was used to describe the requirements of computational jobs for submission to resources in Grid environments. The Grid Laboratory Uniform Environment (GLUE) [4] is a conceptual information model for Grid entities described using the natural language and UML Class Diagrams. Globus Resource Specification Language (RSL) [8] described grid resources including computational job information. The Simple API for Grid Applications (SAGA) [12] aimed to provide high-level interfaces for common grid components (e.g., transfer and scheduling). RADICAL-SAGA [20] is a current implementation of the SAGA specification, including Python bindings, and is used by the RADICAL-Cybertools suite of tools.

Perhaps the most similar approaches are the directory services used to maintain state about distributed computing environments. The Globus Monitoring and Discovery Service [6] builds upon the LDAP protocol to address the distributed resource selection problem. It maintains not only static configuration information about distributed grid resources but also low level information recorded by participating components. It implements a decentralized structure designed to enable it to scale to represent large grids and frequent data updates.

The XSEDE Information services discussed in Section 5.1 internally uses the GLUE2 schema. Though, we will remain focused on the goal to create interoperable CI components for which adoption and maintainability are key factors. The initial stakeholders (the co-authors of this paper) represent a diverse set of projects that are committed to consume and maintain the proposed descriptions and we fully intend to engage with the wider CI developer community.

3 PROJECT AND INVENTORY GOVERNANCE

Our goals with the SGCI resource description schema, resource inventory and reference implementations are community efforts. We encourage code and other community contributions, resulting in more diverse communities coalescing around this valuable effort. We describe our community governance, essentially the rules that guide the interactions intending to encourage participation with a discussion that lead to resolutions and decisions. [17] provides a general discussion of various governance approaches.

3.1 Open Governance model

We will adopt the Open Governance [24] models that provide well-defined mechanisms executed through open communications, allowing individuals from diverse and even competing organizations to interact in neutral forums. The Open Governance model enables the contributors to collaborate to encourage growth and transform passive users into active project members.

We will leverage the tooling available in the GitHub ecosystem to implement the Open Governance approach. Technical Decisions as much as possible will be made asynchronously on Pull Requests and Issue discussions. We plan to periodically meet with an open meeting invitation sent to all active contributors. Issue resolution will be done by active stakeholder vote, although the weighting of the votes may not be equal. We have not defined Veto mechanisms, but these will be implicit in the voting process (that is, consensus may be a prerequisite). We will follow the examples in large open source foundations such as the Apache Software Foundation [7]. Most importantly, the project membership will not be limited to a particular organization. All but a few decisions are made by voting on publicly available, archived mailing lists; discussions of new candidate git write access will be the main exception.

3.2 Contributions and Project Sustainability

We believe the Resource Schema and the associated implementations have to be actively maintained and continually enhanced with changing computational landscape. The sustainability of this effort will heavily depend on the communities who are invested in the project's success. This project was initially seeded by SGCI partners from Apache Airavata, HUBzero®, and Tapis projects. As a first principle, the schema and inventory reference implementations must obviously fulfill a need for a community of users to build a community. Open Ondemand and XSEDE Information Services groups joining this effort validates the usefulness.

4 THE INVENTORY API AND LANGUAGE SDK

4.1 Overview

The SGCI Resource Descriptions project provides science gateways and gateway frameworks with three possible design options for integration. First, all resource definitions are publicly available from the GitHub repository, and projects are free to download or clone the repository to a local file system for direct use. This approach is perhaps the most straightforward technique, but it introduces the potential for local definitions to get out of sync with the definitions on GitHub. To avoid the sync issue, projects can opt to use the GitHub API to pull resource definitions in real time. This is a good choice for many projects, as the GitHub API has very high availability, and using the API means always working with the latest definition. However, usage of the GitHub API is subject to rate limits, per the documentation [1]. For projects requiring updated resource definitions with no rate limits, an official Inventory API has been developed (see 4.2 for details). Projects can choose to deploy an instance of the Inventory API or use an existing instance. Finally, software development kits (SDKs) in Python and Java provide support for all three integration approaches.

Additionally, the Resource Description specification, Inventory API and languages SDKs have been designed to accommodate pulling a resource definition from multiple sources. We anticipate that over time a number of large projects and organizations may choose to deploy an instance of the Inventory API. While the ultimate source of truth for a given resource definition lives in GitHub, some instances of the Inventory API may have a more up-to-date description for a short period of time. The Resource Description schema includes an optional attribute for specifying a location (i.e., a URL) of an Inventory API containing the most up-to-date definition for the resource. Projects can choose to retrieve resource definitions from the most up-to-date source if needed. This is particularly useful for more dynamic information, such as available software modules and outage periods. The language SDKs provide a merge function for automatically combining multiple definitions of a given resource.

4.2 Inventory API Design

The inventory is designed with goals of having a centralized resource inventory, ability to add new resources, modify, and maintain existing information as changes are made to the underlying resources (e.g., additional nodes added to a queue on a resource) and the inventory itself must be highly available because this impacts the availability of the science gateways using it and finally, the physical cost of building and maintaining the inventory must be kept to a minimum, as this is critical to its sustainability.

We build an intermediary data store to mitigate potential concern of GitHub's availability, performance and API rate limiting (which limits the number of queries a project could make against GitHub within a given time period). We introduce an inventory cache. The inventory cache service pulls data from GitHub and persists in a Mongo Database. The reference implementation provides a GRAPHQL [28] and REST API's to allow fetching and searching of resources descriptions from the inventory. GraphQL provides a more flexible, filtered and controlled search of the inventory data with familiar REST-like operations.

4.3 Language SDKs

As part of the SGCI Resource Inventory effort, we have developed a Python SDK — an installable Python library called SCGICatalog [26], for working with resource descriptions. SGCICatalog is currently published on GitHub, but we are planning to publish it to pypi and/or other public package repositories. SCGICatalog allows Python programs to extract resource descriptions from different locations: GitHub, given a Token or User/Password credentials; XSEDE, accessing the XSEDE Information Services API; or Local, using a local copy of the resources. SCGICatalog allows clients to list all the resources available from that location, and get the JSON description of a specific resource from the location. SCGICatalog validates the resource against the SCGI Inventory schema and warns users of inconsistencies of the resource.

SCGICatalog also allows clients to search for specific information from the resource definition. Search is implemented using JMES [13] Path syntax, so complex and multiple queries can be performed on the same search, and clients can merge different descriptions of the same resource loaded from different locations.

Similarly, we have developed a convenient Java client [25] to fetch JSON resource descriptions from within Java programs. As adoption and use of the SGCI resource inventory increases, we anticipate existing clients will be enriched and new clients will emerge.

5 PUBLISHING RESOURCE DESCRIPTIONS INTO THE INVENTORY

5.1 XSEDE API

The XSEDE program federates 20 compute, storage, and cloud resources allocated on a merit basis to 2,500 projects and 11,000 users, plus another 20 un-allocated resources [33]. XSEDE has several information systems that manage resource information of interest to science gateways. XSEDE operators manually enter information in the Resource Description Repository (RDR) and enter resource outages into the XSEDE Central Database (XCDB). They automatically publish batch scheduler information every few minutes by running an Information Publishing Framework (IPF) service on each resource. Both manually entered and automatically collected information is continuously aggregated into XSEDE's Central Information Services [16] [32] where it is cross-referenced and stored in an information warehouse. XSEDE's User Portal, Research Software Portal, and other services access resource information using public Central Information Services RESTful APIs to provide users and software with authoritative up-to-date resource information to enable resource discovery and access [21].

XSEDE's collaboration with SGCI's Resource Description project started by comparing the proposed schema to XSEDE's existing resource information. The comparison identified schema information XSEDE already has, information XSEDE is missing, and information XSEDE already has that wasn't in the schema that might be of interest to science gateways.

XSEDE already has basic descriptive information about resources in pre-production, production, or post-production phases available to science gateways; information about the GridFTP and OpenSSH connections available for science gateways to login, manage jobs, and transfer data to or from resources; and about the batch scheduler (job manager) and node hardware partitions available through the scheduler.

Information that XSEDE already had and has proposed as additions to the SGCI schema include information on the current operational status of resources with associated start and end dates, current and future outages, and available software modules. These are being explored as possible additions to the schema because they are of interest to science gateways, though they introduce a more dynamic aspect of resource descriptions.

Through this collaboration XSEDE learned that science gateways are interested in file-system information and in the paths to important tools and commands which it currently does not collect. XSEDE will need to work with resource operators to manually or automatically collect this information.

Leveraging XSEDE's Central Information Services which aggregates manually entered and automatically collected resource information from multiple information systems, XSEDE developed an initial SGCI Schema version 1.0 compatible API that returns active allocated XSEDE resource information plus the proposed resource status and outages information additions [34].

5.2 Publishing Existing Descriptions From Frameworks

5.3 TAPIS

Tapis is an open source, domain-agnostic API framework funded by the National Science Foundation for distributed, reproducible computational research, hosted at the Texas Advanced Computing Center at the University of Texas at Austin, the Hawaii Datascience Institute at the University of Hawaii and other partnering institutions [27] . Using Tapis, researchers can work with storage and computing systems across the United States and internationally to manage data and metadata and execute software using a common API. Thousands of users across dozens of projects use Tapis for their research.

The Tapis team developed a script for automatically converting a Tapis system to an SGCI resource. One interesting challenge is that while every Tapis system includes a host attribute having the same meaning as in SGCI resources, the host attribute is not required to be unique in Tapis systems. This lack of uniqueness enables users to "customize" the system definition Tapis uses for them for a given host in various ways – for example, the rootDir attribute on a Tapis system is a path that acts like chroot so that Tapis prepends it to any absolute path against that system. The conversion script merged some attributes from different system definitions for the same host while discarding others.

5.4 Open OnDemand

Open OnDemand is an NSF-funded open source general purpose web HPC portal that enables users to manage files, submit jobs, and use many different graphical applications. OnDemand has been downloaded by more than 200 HPC facilities around the world.

SGCI resource descriptions will be used for OnDemand's cluster configuration. To encourage publishing to the inventory catalog, a script for publishing and a web IDE for editing these resource descriptions may be developed and provided through the OnDemand interface.

5.5 Airavata

Apache Airavata [18] is an open-source science gateway framework that powers the multi-tenanted Science Gateways Platform as a service (SciGaP) [22]. As of this writing, SciGaP platform operates 45 Science Gateways executing over 200 scientific applications on over 70 supercomputing resources across the world. Apache Airavata internally curates resource descriptions of these resources in an application catalog registry accessible with a Thrift based API. A java library [2] was written to extract compute and storage resource descriptions from Airavata registry and translate to SGCI resource schema. These resulting JSON documents were manually committed to the SGCI resource inventory. We plan to enhance this description extraction and publishing to inventory process by adding checks to verify existence of a given resource within the inventory and merge the information. From our early experience, the translation of the Airavata internal format and the SGCI scheme was straight forward and no information field mismatches were identified.

5.6 Manual Curation of Missing Details and Resources

As discussed in the Introduction, our contributions in this paper highlight both the Schema as well as curated resource descriptions in the SGCI inventory. Above we described how gateway frameworks and XSEDE bootstrap resources descriptions. However we realize and expect the inventory will never be comprehensive enough to cover all publicly describable computational resources. As we gather experience in how the missing descriptions will be handled, we will share our findings in the future publications.

As discussed in the governance section, our current plan is to engage the community at large and encourage publications through Github pull requests. As discussed more in the Future Work section, we will consider providing utilities which can be installed on resources to query scheduler and publish to the inventory. This will be a software distribution and administrators have to install the utility to install on a given resource. Alternatively, an administrator can manually populate the JSON document and submit a pull request.

6 USING RESOURCE DESCRIPTIONS IN SCIENCE GATEWAYS AND FRAMEWORKS

In Section 5 we discussed how gateway frameworks and XSEDE can contribute resource descriptions and publish into a common SGCI resource inventory. In this section we discuss how gateway frameworks can consume the descriptions from the inventory.

6.1 Open OnDemand

OnDemand's interactive app plugins define user inputs and the job template required for starting interactive apps such as RStudio. Each plugin may contain duplicate information about a cluster's node properties, partitions, and other details required to properly submit the batch jobs. This duplication imposes a maintenance burden when maintaining an OnDemand instance and makes it more difficult for other sites to copy OSC's OnDemand plugins for their use. The SGCI schema provides similar configuration options that OnDemand's "cluster config" provides while also supporting the detailed hardware configuration in the form of hardwareProfiles. By adopting the SGCI schema for OnDemand's new cluster config, sites running OnDemand will be able to shift most of the hardware configuration out of the numerous plugins into a centralized location, simplifying the plugin code and making the plugins more portable.

OnDemand will support both JSON and YAML cluster configs that validate against the schema. The YAML version will make schema adoption easier for sites not interested in the inventory catalog but wanting to enjoy the benefits the schema confers. YAML allows inline comments and is the format of other OnDemand configuration.

Many sites are running OnDemand where the OnDemand operator is not a sysadmin. The use of the SCGI inventory may reduce the maintenance burden for these sites. System admins at these sites can publish resource definitions for their clusters to the SGCI inventory. Then the OnDemand operators at those sites can pull the resource definitions into OnDemand. As new clusters come online and old clusters are taken offline, OnDemand may automatically update the new or updated resource definitions.

6.2 Tapis

The Tapis Framework uses systems as one of its fundamental abstractions, and several other higher-level features including files (and associated data management), apps, and jobs rely on them. The Tapis system abstraction closely resembles an SGCI resource. One difference between SGCI resources and Tapis systems is that SGCI resources can have multiple storage capabilities, such as POSIX and S3-compliant storage endpoints, while Tapis assumes each system provides one type of storage. The Tapis team developed an endpoint within the Systems API for importing an SGCI resource description as one or more systems: the import will produce one system for each storageResource defined in the resource definition. An additional development effort currently underway will add the ability to import information contained within a computeResource object as one or more capabilities of the systems being created.

Beyond their use as a source of system definitions, Tapis plans to enable datasets to be described in terms of an SGCI resource, where by a dataset we simply mean a file or folder on some storage resource. Tapis already recognizes a special syntax comprised of the system ID and path for referencing datasets on Tapis systems. This syntax appears in many places throughout the Tapis API, such as sources for data transfers or inputs to jobs. An analogous syntax comprised of the SGCI host and path will be used to reference a file or folder on an SGCI resource. Because the host attribute uniquely defines an SGCI resource, Tapis only must find a system definition with a matching host attribute that the user has access to. It is worth noting that Tapis systems can be created on behalf of large groups of people, even an entire project (or "tenant"). Therefore, it will be possible for the typical Tapis user to work exclusively with SGCI hosts in API requests involving data, meaning those data could have been generated outside of Tapis, for example, through another gateway or gateway framework technology.

6.3 Airavata

The Apache Airavata [18] science gateway framework API to describe computational resources is based on Apache Thrift, which gives Airavata a strongly typed, programming language independent way of defining its interfaces. From the API definitions, Airavata generates client packages in multiple languages including Java, Python, PHP and C++. Client gateways access Airavata through the API Server through a secure channel (SSL sockets or HTTPS). Airavata allows administrators to define and describe computational resources and application descriptions in its App Catalog component. Airavata and its eco system of components will integrate with the SGCI resource inventory through an intermediate bridge service. The service will translate SGCI schema into Airavata internal compute and storage resource descriptions. The service will subscribe to changes to SGCI schema and will run integration tests, once passed the SGCI inventory descriptions will be synchronized with Airavata application catalog descriptions. Airavata integration with the schema to build downstream components to configure user compute and storage preferences is available from [3].

7 INTEROPERABILITY AND PROOF-OF-CONCEPT JUPYTER NOTEBOOK

7.1 Overview and Interoperability

This section presents a demonstration that utilizes SGCI resource descriptions to conduct a multi-step analysis across three machines at three different institutions. The workflow, packaged as a Jupyter app, runs Quantum ESPRESSO to compute and visualize the band structure of a Gallium Arsenide (GaAs) crystal. In addition to being a realistic analysis, distributing such a workflow across systems is useful because each step requires substantial compute which may only be available at a certain site at any given time. Each step may require a large set of input files that may reside on only one machine.

Notably, the demo obtains all information about the resources it needs to execute the workflow from the SGCI resource descriptions with one exception: information about the multi-factor authentication system on each is "hard-coded" into the app. (Adding support for multi-factor authentication metadata is planned future work). Additionally, the demo app suggests another area currently planned for future work – defining a common language for describing applications. The demo uses an ad hoc definition of applications to describe Quantum ESPRESSO. Still, the SGCI project plans to ultimately provide a formal specification for applications (see Future Work).

7.2 Scientific Background

Band structure is one of the most essential concepts in solid state physics, as it defines a schematic way of visualizing the electronic configuration of a system. Visualizing the bands helps in understanding the characteristics of different materials and their differences. Usually, to calculate band-structures for a material, users

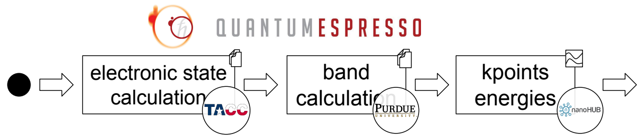

Figure 3: Diagram of the three-step workflow to generate band structures

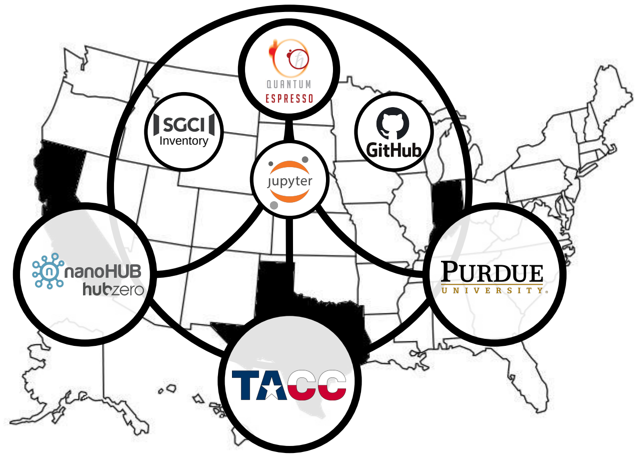

Figure 4: Overview of geographically distributed resources used by the Jupyter notebook

have to follow a simple three-step workflow to generate these bands. A high-level overview of the steps involved follows.

The electronic state calculation step consists of obtaining a potential energy surface, and requires solving the electronic Schrödinger equation over a range of nuclear coordinates on a Self-Consistent Field calculation (SCF) process. The electronic Schrödinger equation solved numerically can calculate approximated wavefunctions. The SCF method is an iterative method that involves selecting an approximate Hamiltonian, calculating a more accurate set of orbitals, and iterating until results converge. The SCF method requires a set of pseudopotentials for each atom type in the system, and the particular k-points of interest.

The bands calculation step requires a non-SCF calculation with the desired k-points grid and the number of bands to be calculated. The calculation requires a preset potential, usually as output of the result of an electronic state calculation.

Finally, the k-points and energies step requires post processing the results. All output data from previous calculations are distributed across multiple files and output logs. The visualization of the bands requires the path for each band to be reconstructed based on energies and k-points.

7.3 Demo Application

We present a Jupyter App [19], a Jupyter notebook that can be executed in AppMode or running on a Voila server[23], that distributes the band structure visualization workflow on three different HPC providers: Halstead/RCAC at Purdue University, Stampede2 at TACC, and nanoHUB/HUBzero®hosted at SDSC. The notebook takes advantage of the Python SDK (SCGICatalog) and the SGCI

Resource Inventory to get access to different resource descriptions and defines templates based on JMESPath queries to extract relevant information of each resource. Additionally, the notebook defines the notion of an “application" to be executed, in this case Quantum ESPRESSO, as the path to the executable file, a list of modules required, and a set of inputs, and these data are referenced by the templates.

After extracting the information from the definitions, the notebook creates connections using dual factor authentication (2FA) to each server; however, each server implements different dual authentication methods – Stampede2 uses a time-based one-time authentication, while Halstead uses a push-based authentication. As the notebook has to handle these connections independently, a resource's dual authentication implementation should be part of the schema for password based connections. When connections are successfully established, the notebook guides users to submit the electronic state calculation to Stampede2, moves the results to Halstead, submits the bands calculation on Halstead, copies the results back to nanoHUB, extracts the relevant information from files, and visualizes the band structure of Gallium Arsenide (GaAs) as an interactive widget using Plotly [10].

nanoHUB and HUBzero®are looking to integrate SCGICatalog as part of the tools included in the core services to interact with external providers. Submit jobs to community clusters using the Tapis API or submit jobs to other HPC providers registered on Airavata.

8 FUTURE WORK

The SGCI Resource Description project plans to expand the present work with several future efforts. First, while version 1.0 already includes the language to describe several prominent resource types, including storage and computing resources commonly found in major academic datacenters, support for more modern and less traditional computing resources, including some cloud computing capabilities, is somewhat lacking. As community usage of these types of resources increases and experience with them grows, the project expects that the motivation and knowledge needed to expand the schema will result in community contributions to add and support them.

The project will engage in several efforts to grow the catalog of resource descriptions currently available. XSEDE can expand the resource descriptions it publishes by encouraging the 20 unallocated XSEDE federated resources that aren't required to use its tools to do so and thus enhance access by science gateways. These XSEDE tools could also be made available to resource operators not federated with XSEDE. XSEDE is also exploring whether to contribute to the community a RESTful API that caches all SGCI resource descriptions from GitHub in XSEDE's Central Information Services. Two potential benefits would be that the RESTful API could provide query capabilities. The cache is more well suited for handling frequently changing resource descriptive information that GitHub.

Beyond new resource types, the project recognizes the inevitable need to support additional information on resources. Some information, such as metadata about the multi-factor authentication systems present on a resource, are already known to the project. In contrast, others are unknown and will come by way of community usage. Dynamic information in particular, such as the available software modules or real-time outage information, presents both an opportunity and a challenge. A fundamental issue that future work will address pertains to the life cycle of resource definitions and the frequency and mechanisms by which the definitions are updated. Currently, all changes to project artifacts, including resource definitions, are made via public pull requests (PRs) to the GitHub repository. The team welcomes PRs from the community to improve any aspect of the project, including updates to resource definitions, documentation, or code improvements to the Inventory API or language SDks. The project will be reviewing all PRs to the GitHub repository during regularly scheduled meetings on a monthly interval, and these meetings are open to community participation. The meeting schedule with connection information will be posted to the Resource Description project page within the SGCI website. Over time, as the project's adoption hopefully increases, we will learn the community's needs and adopt additional policies and mechanisms for updating resource description information as needed.

Descriptions of resources form a foundation upon which the project plans to develop an application description schema and associated tooling. The idea is to enable users to invoke applications across cyberinfrastructure resources and science gateways seamlessly, we require a common language for describing the applications. However, applications, in turn, have requirements on the resources where they run (for instance, a dependency on a hardware type, a software module or a container runtime). Thus, an application description language depends on having a common language to describe the resources where they will run.

Ultimately, the goal of the project is to improve interoperability across science gateways and the cyberinfrastructure they leverage. As new use cases emerge through increased community adoption, the project will evolve to better meet that end.

9 CONCLUSION

This paper presents version 1.0 of the SGCI Resource Description project, a common language and set of tools for describing the attributes needed by science gateways to make use of a storage or computing resource. In collaboration with the XSEDE project, the SGCI Resource Inventory contains detailed, up-to-date information about all active XSEDE systems. The paper describes additional integrations and collaborations with the Airavata, OnDemand and Tapis projects, and it provides multiple design options to science gateways wanting to use SGCI Resource descriptions. Finally, the paper tackles the interoperability problem. Includes a description of a proof-of-concept Jupyter notebook that leverages the Inventory API and Python SDK to run a multi-step Quantum ESPRESSO analysis across resources at different institutions.

ACKNOWLEDGMENTS

Authors JS, SM, DM, SB, DW, SP, CS, MD, MP and MZ are supported by NSF Award #1547611. Author EF supported by NSF Award #1835725.

REFERENCES

[1] 2021. *GitHub Rate Limiting*. https://docs.github.com/en/rest/overview/resources-in-the-rest-api#rate-limiting Last access: 2021-03-08.

[2] Apache Airavata. 2020. *Data Publish*. Retrieved March 9, 2021 from https://github.com/SciGaP/scigap-resource-inventory

[3] Apache Airavata. 2020. *Resource Descriptions*. Retrieved March 9, 2021 from https://github.com/apache/airavata/tree/storage-resource-profile/modules/resource-profile

[4] Sergio Andreozzi, Stephen Burke, Felix Ehm, Laurence Field, Gerson Galang, Balazs Konya, Maarten Litmaath, Paul Millar, and JP Navarro. 2009. GLUE Specification v. 2.0. In *Open Grid Forum Recommendation Documents*. Open Grid Forum.

[5] Ali Anjomshoaa, Fred Brisard, Michel Drescher, Donal Fellows, An Ly, Stephen McGough, Darren Pulsipher, and Andreas Savva. 2005. Job Submission Description Language (JSDL) Specification, Version 1.0. In *Open Grid Forum, GFD*, Vol. 56.

[6] Karl Czajkowski, Steven Fitzgerald, Ian Foster, and Carl Kesselman. 2001. Grid Information Services for Distributed Resource Sharing. In *Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing (HPDC)*. 181–194.

[7] Roy T Fielding. 1999. Shared leadership in the Apache project. *Commun. ACM* 42, 4 (1999), 42–43.

[8] Ian Foster and Carl Kesselman. 1998. The Globus project: A status report. In *Proceedings of Seventh Heterogeneous Computing Workshop (HCW'98)*. 4–18.

[9] Dave Hudak, Doug Johnson, Alan Chalker, Jeremy Nicklas, Eric Franz, Trey Dockendorf, and Brian L. McMichael. 2018. Open OnDemand: A web-based client portal for HPC centers. *Journal of Open Source Software* 3, 25 (2018), 622. https://doi.org/10.21105/joss.00622

[10] Plotly Technologies Inc. 2015. *Collaborative data science*. Montreal, QC. Retrieved March 9, 2021 from https://plot.ly

[11] Inoor. 2020. *sphinx-jsonschema*. Retrieved March 9, 2021 from https://github.com/lnoor/sphinx-jsonschema

[12] Shantenu Jha, Hartmut Kaiser, Andre Merzky, and Ole Weidner. 2007. Grid Interoperability at the Application Level using SAGA. In *Third IEEE International Conference on e-Science and Grid Computing (e-Science 2007)*. 584–591.

[13] JMESPath. 2020. *Specification*. Retrieved March 9, 2021 from https://jmespath.org/specification.html

[14] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. 2016. *Jupyter Notebooks-a publishing format for reproducible computational workflows*. Vol. 2016.

[15] Katherine A Lawrence, Michael Zentner, Nancy Wilkins-Diehr, Julie A Wernert, Marlon Pierce, Suresh Marru, and Scott Michael. 2015. Science gateways today and tomorrow: positive perspectives of nearly 5000 members of the research community. *Concurrency and Computation: Practice and Experience* 27, 16 (2015), 4252–4268.

[16] Lee Liming, John-Paul Navarro, Eric Blau, Jason Brechin, Charlie Catlett, Maytal Dahan, Diana Diehl, Rion Dooley, Michael Dwyer, Kate Ericson, et al. 2009. TeraGrid's integrated information service. In *Proceedings of the 5th Grid Computing Environments Workshop*. 1–10.

[17] M Lynne Markus. 2007. The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management & Governance* 11, 2 (2007), 151–163.

[18] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, Ross Gardler, et al. 2011. Apache airavata: a framework for distributed applications and computational workflows. In *Proceedings of the 2011 ACM workshop on Gateway computing environments*. 21–28.

[19] Daniel Mejia. 2021. *SGCI Resource Schema Demo Jupyter App*. Retrieved March 9, 2021 from https://github.com/denphi/sgci-demo

[20] Andre Merzky, Ole Weidner, and Shantenu Jha. 2015. SAGA: A Standardized Access Layer to Heterogeneous Distributed Computing Infrastructure. *SoftwareX* 1 (2015), 3–8.

[21] John-Paul Navarro and Amy Hovious. 2019. Breaking Resource Discovery Barriers. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*. 1–2.

[22] Marlon Pierce, Suresh Marru, Eroma Abeysinghe, Sudhakar Pamidighantam, Marcus Christie, and Dimuthu Wannipurage. 2018. Supporting science gateways using apache airavata and scigap services. In *Proceedings of the Practice and Experience on Advanced Research Computing*. 1–4.

[23] QuantStack. 2020. *And voilà!* Retrieved March 9, 2021 from https://blog.jupyter.org/and-voil%C3%A0-f6a2c08a4a93

[24] Eric Raymond. 1999. The cathedral and the bazaar. *Knowledge, Technology & Policy* 12, 3 (1999), 23–49.

[25] SGCI. 2020. *Resource Clients*. Retrieved March 9, 2021 from https://github.com/SGCI/sgci-resource-clients

[26] SGCI. 2020. *Science Gateways Community Institute Catalog*. Retrieved Feb 17, 2020 from https://catalog.sciencegateways.org/

[27] J Stubbs, R Cardone, M Packard, A Jamthe, S Padhy, S Terry, J Looney, J Meiring, S Black, M Dahan, S Cleveland, and G Jacobs. 2021. Tapis: An API Platform for Reproducible, Distributed Computational Research. In *Proceedings of the 2021 Future of Information and Communication Conference (FICC)*. Springer.

[28] Ruben Taelman, Miel Vander Sande, and Ruben Verborgh. 2018. GraphQL-LD: linked data querying with GraphQL. In *ISWC2018, the 17th International Semantic Web Conference*. 1–4.

[29] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D Peterson, et al. 2014. XSEDE: accelerating scientific discovery. *Computing in science & engineering* 16, 5 (2014), 62–74.

[30] Peter Tröger, Roger Brobst, Daniel Gruber, Mariusz Mamonski, and Daniel Templeton. 2012. Distributed Resource Management Application API Version 2 (DRMAA). Technical report, Open Grid Forum.

[31] Nancy Wilkins-Diehr, Michael Zentner, Marlon Pierce, Maytal Dahan, Katherine Lawrence, Linda Hayden, and Nayiri Mullinix. 2018. The science gateways community institute at two years. In *Proceedings of the Practice and Experience on Advanced Research Computing*. 1–8.

[32] XSEDE. 2021. *Info Services*. Retrieved March 9, 2021 from https://info.xsede.org/info

[33] XSEDE. 2021. *Service Providers*. Retrieved March 9, 2021 from https://www.xsede.org/ecosystem/service-providers

[34] XSEDE. 2021. *XSEDE SGCI v1 API*. Retrieved March 9, 2021 from https://info.xsede.org/wh1/warehouse-views/v1/resources-sgci/v0.1.0/?format=json