Tapis: An API Platform for Reproducible, Distributed Computational Research

Joe Stubbs*, Richard Cardone*, Mike Packard*, Anagha Jamthe*, Smruti Padhy*, Steve Terry*, Julia Looney*, Joseph Meiring*, Steve Black*, Maytal Dahan*, Sean Cleveland†, Gwen Jacobs†
*Texas Advanced Computing Center, Austin, TX, USA
jstubbs, rcardone, mpackard, ajamthe, spadhy, sterry1, jlooney, jmeiring, scblack, maytal@tacc.utexas.edu †University of Hawaii, Manoa, HI, USA
seanbc, gwenj@hawaii.edu

Abstract—Modern computational research increasingly spans multiple, geographically distributed data centers and leverages instruments, experimental facilities and a network of national and regional cyberinfrastructure (CI). In this paper we introduce Tapis, an open-source API platform funded in 2019 by the National Science Foundation for distributed computational experiments. We describe the motivation and core features of the Tapis platform, including data management and code execution, with support for streaming data and real-time as well as batch workflows. A fine-grained permissions system underlies all Tapis objects enabling data to be stored privately, shared with individuals or "published" to a community, and provenance endpoints expose the detailed history logs Tapis collects on analyses, enabling workflows to be repeated and results reproduced. We describe how Tapis builds upon prior NSF investments in CI, discuss some of the early use cases driving its design, and conclude with the roadmap for future work.

1 Introduction

Modern computational experiments are becoming increasingly distributed, particularly those designed by interdisciplinary teams. These efforts span geographically distributed data centers and leverage instruments, experimental facilities, and national and regional cyberinfrastructure to address fundamental problems in science and engineering. For example, investigators might first apply machine learning techniques to raw data from remote sensors, or they might perform genetics analysis against sequencing data produced in a robotics wet lab. Next, they might pass those results to a simulation that runs on a traditional HPC supercomputer. The simulation results in turn help to calibrate the remote instruments and processes for the next iteration, and the cycle repeats. Programming the execution of such experiments, much less in a scalable, reproducible way, presents a formidable challenge to investigators and prevents many such experiments from ever being run. Furthermore, the operation of the advanced cyberinfrastructure to accommodate such experiments requires specialized knowledge and experience, creating another obstacle for many regional academic providers.

Tapis is an open source, NSF funded Application Program Interface (API) platform for distributed computation that will provide production-grade capabilities to enable researchers to 1) securely execute workflows that span geographically distributed providers, 2) store and retrieve streaming/sensor data for real-time and batch job processing, with support for temporal and spatial indexes and queries, 3) leverage containerized codes to enable portability, and reduce the overall time-to-solution by utilizing data locality and other smart scheduling techniques, 4) improve repeatability and reproducibility of computations with history and provenance tracking built into the API, and 5) manage access to data and results through a fine-grained permissions model, so that digital assets can be securely shared with colleagues or the community at large.

Tapis is itself distributed: a hosted RESTful API platform, deployed at various institutions including the Texas Advanced Computing Center (TACC) at the University of Texas at Austin, the University of Hawaii (UH), Manoa, and others. End users and applications interact with Tapis by making authenticated HTTP requests to Tapis's public endpoints. In response to requests, Tapis's network of microservices interact with a vast array of physical resources on behalf of users: high performance and high throughput computing clusters, file servers and other storage systems, databases, bare metal and virtual servers, etc. The goal of Tapis is to provide a unified, simple to use API enabling teams to accomplish computational and data intensive computing in a secure, scalable, and reproducible way so that domain experts can focus on their research instead of the technology needed to accomplish it. Working alongside researchers from various fields to drive real-world use cases, Tapis aims to be the underlying cyberinfrastructure for a diverse set of research projects: from large scale science gateways built to serve entire communities, to smaller projects and individual labs wanting to automate one or more components of their process.

Tapis is motivated by lessons learned from developing and operating the Agave API platform [1] for over six years, and draws on NSF investments in the Abaco (Actor Based Containers) [2] functions-as-a-service project as well as CHORDS (Cloud-Hosted Real-time Data Services for the

Geosciences) [3] platform. Tapis is made up of a set of API services that together form the Tapis platform. The primary APIs include the Files API for managing data on remote storage, the Systems, Apps and Jobs APIs for registering and executing software and research codes, the Actors API for registering small executables that run with very low latency in response to messages sent over HTTP, the Meta API, a high performance document store for scaling research data collections to billions of documents serialized using formats such as JSON or XML, and the Streams API for storing and retrieving sensor data. Underlying all of these services is a set of authorization APIs which comprise the Tapis Security Kernel. The Tapis security kernel is a unique, decentralized solution to authorization and security management with APIs that enable trust federation across physical and institutional boundaries.

A recurring technique leveraged throughout the platform is the use of container technology to enhance portability and reproducibility, not just for the end user's research computations, but for all Tapis execution. Each Tapis microservice is packaged into a Docker image, and the official Tapis deployment tooling targets the Kubernetes platform. This approach simplifies operations across datacenters while simultaneously increasing the uniformity of deployed Tapis components, whether they run at academic institutions, nationally funded cloud providers or the commercial cloud.

The rest of the paper is organized as follows: in section 2 we discuss in detail the background, motivation and related work; in 3 we describe the primary capabilities Tapis will provide; in 4 we highlight some projects that have agreed to be early adopters of the platform, and describe their use cases and requirements; in 5 we provide a high-level overview of the Tapis architecture; finally, in 6 we conclude with a road map for the five funded project years.

2 Background and Related Work

2.1 API Platforms

Microservice architectures, JSON and OAuth2 have greatly reduced barriers to distributed application development and have enabled new usage patterns across industry and academia. Microservices typically are HTTP-based APIs built on open standards such as REST. All leading cloud providers including Amazon AWS, Google Cloud Platform and Microsoft Azure provide such services. At the Texas Advanced Computing Center, the Agave science-as-a-service API enabled 14 different official projects to manage data, run jobs on HPC and HTC systems, and track provenance and metadata about computational experiments. The more recently funded Abaco API supports functional programming models and event driven architectures on cloud infrastructure. Abaco's novel use of containers and the Actor Model of concurrent computation support several projects after less than two years in production. Official collaborations include projects across a wide range of scientific/engineering domains and NSF directorates, including CyVerse [4] (formerly, the iPlant Collaborative) [5] [6], DesignSafe [7], Ike Wai [8], 3DEM Hub [9], the Science Gateways Community Institute [10], and VDJServer [11]. Non-NSF projects include the iReceptor portal for immune genetics research; the CNAP Center of Biomedical Research Excellence at Kansas State University; and Planet Texas 2050, funded by Texas and the University of Texas, Austin. An analysis of usage data suggests that many more projects leverage these platforms in an unofficial capacity, with approximately 20,000 OAuth client applications having been registered.

2.2 Containers and Distributed Computations

The number of scientific workflows making use of distributed computational experiments is increasing. Advances in technologies related to small devices, instruments, robotics and other forms of experimental facilities have led to an explosion in large, real-time data sources. Examples include genomic sequencing facilities, survey telescopes, climate sensors, shake tables, wind tunnels, and laser light sources. Computations often leverage a mix of AI/Machine Learning techniques in combination with model simulations to derive new insights. As a result, these experiments require a mix of computational resources, data access methods and management techniques.

Researchers analyze sensor data using batch and real-time processing. With batch processing, investigators write programs to analyze a static set of data defined at the start of the analysis. These programs may utilize traditional HPC machines scheduled via a batch scheduler such as SLURM, or high-throughput and/or cloud resources, and they may be written in a variety of languages and frameworks. Such frameworks include traditional model simulations that leverage MPI, AI/ML code that depend on CUDA or higher-level libraries such as TensorFlow, or other data-intensive frameworks like Hadoop and Spark. With real-time processing, codes analyze new streaming data points as they arrive. In the simplest cases, these analyses check basic conditions to determine if interesting or unusual criteria are met; in the general case, real-time processing can involve many of the same kinds of analysis as batch processing.

Finding a mechanism to enable computational portability can be a great simplification if not an out right requirement for successfully orchestrating such workflows. Steps in the workflow that analyze streaming data often have a real-time, low-latency requirement that can only be achieved by running code near the data. Traditional simulation steps often require running codes across large clusters that may only be available in world-class HPC centers. For other steps with less resource requirements, minimizing time-to-solution may involve leveraging high throughput systems and machines with shorter queue times than highly-demanded supercomputers. If the workflow application (or individual step components) can be easily moved to different computing resources, different steps can leverage the most appropriate resource available at the time of execution.

Over the last five years or so, software teams have started adopting Linux container technologies, chiefly the Docker platform, to improve application portability. Container best practices encourage practitioners to bundle all application assets, including software libraries and other dependencies, into the container image to produce an independent package to minimize dependencies on the execution environment. The goal is to enable containers to be executed

from the image on any machine where the container runtime is available.

While container technology significantly improves the portability and reproducibility of general applications, adoption within the scientific computing community has lagged behind industry. The primary reason is that containers cannot easily encapsulate all dependencies of scientific code running in customized, heterogeneous environments. Scientific computations rely on specialized hardware and custom software installed across a computing infrastructure, including MPI, networking, mathematical and GPU libraries, and specialized workflow managers, that cannot be captured within a container. Additionally, the container runtime itself becomes a dependency of containerized applications, which may rely on specific versions, configurations, plugins or optional features of that runtime.

Our strategy with Tapis therefore is to augment container technology with metadata captured by the platform that describes requirements of applications as well as capabilities of different execution environments. These abstractions, described in more detail in 3.1, are based on our experience running containerized workloads in a variety of contexts. The Singularity [12] runtime is available on all major HPC systems at TACC, where over 4,400 unique containerized applications are available for use. In its first two years of operation, nearly 25,000 Docker images representing actor functions were registered with the Abaco platform. TACC manages many other containerized applications across its various cloud offerings such as its custom JuptyerHub clusters.

A growing body of evidence suggests that time-to-solution can be reduced using these techniques. For instance, a 2018 paper entitled Virtualizing the Stampede2 SuperComputer with Applications to HPC in the Cloud [13] describes methods for building a virtual HPC resource in the JetStream cloud that shares many properties of the Stampede2 supercomputer. The performance of a number of popular scientific applications was measured on both the virtual and real Stampede2 clusters. The paper presents a profile for scientific applications where performance in the virtual cluster is similar to that on the real Stampede2 cluster, suggesting that such applications are amenable to cloud bursting, i.e., scheduling on alternative resources to bypass potentially long queues on HPC systems.

2.3 Distributed Security

Distributed applications are often constrained by rigid authentication, authorization and secrets management requirements. Some applications, for instance, build in their authentication mechanisms, making integration into enterprises with existing security infrastructures and policies difficult. The practice of inventing ad-hoc access control is even more pervasive, where the introduction of administrative users, permissions, groups and roles occurs incrementally as user requirements evolve. Distributed applications almost always have to manage passwords, keys and other secrets; these secrets often get embedded in configuration files, databases, build scripts, deployment scripts, container definition files, etc. All of these practices make application integration and management more difficult and less secure.

The security subsystem is among the most basic and critical of all modules in a distributed application, one on which almost all other subsystems depend, and one that does not depend on other subsystems. Many distributed applications face the same security design issues as Tapis, yet there is not an off-the-shelf package that delivers a robust, lightweight, flexible, high performance solution.

The goal is for the Tapis Security Kernel to provide a complete security API for distributed applications with the following characteristics:

- Easy integration with any authentication/identity manager.
- Fast, fine-grained authorization checking that scales to millions of objects.
- A secure, highly available, multi-tenant secrets store.
- Management of all secrets through the store.
- Support for on-premises or remote installation.

Our approach incorporates hardened open-source packages into a portable system unified by a simple, high-level API. A key simplification is that we use signed tokens created outside of our system as proof of identity. Any authentication system that can create and properly sign a token with a trusted key can be integrated into Tapis - no further specification required. The community version of HashiCorp's Vault [14] product provides a mature, robust secrets store with numerous management features. Apache Shiro [15] provides the basis for a simple, powerful authorization mechanism that we extend (a) with hierarchical roles for improved user management and (b) with a permissions model that allows file path names to be succinctly represented for scalability.

2.4 Software Comparison

We briefly survey the landscape of software systems available for computational research.

2.4.1 Gateway Frameworks

Gateway frameworks provide components for building web-based computational science portals with intuitive user interfaces to advanced cyberinfrastructure. Apache Airavata [16], Agave [1], Galaxy [17], Globus [18], HubZero [19] and WS-PGRADE/gUSE [20] are among the most widely used, domain agnostic projects, and while all have enjoyed undeniable success, none currently attempt to address the aforementioned challenges of distributed experiments. For example, none of these frameworks provide APIs for sensor data. With the exception of Galaxy, which provides support for tools packaged as Docker images, none provide first-class support for containerized application workloads. With Dockerized Galaxy tools, the abstractions do not include notions of capabilities such as specialized hardware and system libraries, required to achieve portability of high-performance codes. Additionally, among these major gateway frameworks, only Globus Auth attempts to provide a decentralized security system; however, Globus Auth does not provide a secrets store for securing arbitrary data such as keys, certificates, database passwords, etc. Moreover, it is neither open source nor free to use, and the charge model isn't publicly available. This is a very different model from that implemented for Tapis.

2.4.2 Gateway Tools

Gateway tools comprise another class of comparable software including workflow managers such as Pegasus [21] and Taverna [22], real-time data mediation services like Brown Dog [23] and SciServer [24], and infrastructures for streaming data such as CHORDs and Data Turbine [25]. We assessed each project for features that would be of potential benefit for distributed workflows and discovered that most would require a heavy development investment to utilize. A detailed discussion of the assessment is beyond the scope of this manuscript.

2.4.3 Commercial Platforms

Tapis draws comparison to a number of commercial cloud offerings. Amazon Web Services (AWS) provides a suite of offerings for storing and processing streaming data including IoT, Kinesis, SQS, Lambda, etc. as well as IAM and secrets management offerings. Google Cloud Platform and Microsoft Azure provide similar if less mature offerings; However, in each case, usage is restricted to the specific commercial platform where costs can be prohibitive for researchers. Moreover, these services lack any significant integration into the national cyberinfrastructure provider fabric, and the closed, proprietary nature of these platforms makes the prospect of future integrations unlikely.

2.4.4 Application Security Software

Distributed applications implement security using different underlying technologies, each offering its own mix of features, compliance, performance, availability, cost, deployment options, etc. Technologies such as OAuth2, Kerberos, PERMIS and Amazon Key Management Service focus on one aspect of security, where others such as Apache Fortress and Microsoft Active Directory incorporate several aspects [Sec1-6]. As a group these implementations assume sole control of the security fabric, which makes integration and management in existing environments difficult.

In contrast, Tapis decouples application code from authentication method, using robust open source authorization and secrets management tools, and packaging those tools as an easily deployable system accessible through a unified API available to any application that wishes to incorporate it. This enables distributed implementation and management while offering fine grained controls for administrators and flexibility for developers and researchers.

3 Primary Capabilities

In this section we detail the primary capabilities of the Tapis framework. While Tapis is a new framework under active development and targeting an initial public release during the summer of 2020, it draws upon architecture, ideas, lessons learned, and, in some cases, actual code, developed under a number of other projects.

3.1 Data Management and Code Execution

Fundamental to all computational research is the ability to manage data and execute codes or *apps* to analyze such data. The Tapis Files API enables users to manage data on remote storage including SFTP and iRODS storage servers and S3-compatible object stores. Synchronous endpoints exist for listing, uploading, downloading and renaming data, and an asynchronous endpoint provides a managed transfer capability between two storage resources. Entire storage resources as well as individual files and directories can be registered as private to a single API user or shared with one or more users.

The Tapis Apps API provides a catalog of executable software which, like the resources within the Files API, can be private to a specific user or shared. The Tapis Jobs API is then used to execute an instance of an app on a remote execution resource. These execution resources can themselves be scheduled using a traditional HPC scheduler such as SLURM, in which case Tapis can inject the necessary scheduler directives based on metadata provided in the job request and in the app definition; otherwise, the Jobs service can start the app by directly forking a process on the underlying operating system.

The app model centers around containers to improve portability and reproducibility, and to enable flexible scheduling of computational workloads across geographically distributed providers. We achieve this flexibility by introducing execution system *capabilities* and application *requirements*. Based on our experience running both native and containerized applications, we have identified an initial list of capabilities required for determining whether a specific execution system can support a given application. The initial list of capabilities can be grouped into the following types:

- *Container runtime capabilities.* These capabilities include type and version of the runtime as well as optional features of the runtime (e.g., Singularity bind mounts) that may be enabled or disabled through configuration by the system administrator.
- *MPI capabilities.* The capabilities include the MPI version as well as the associated networking technology (e.g., Mellanox Infiniband, Intel Omni-Path, etc.)
- *GPU capabilities.* These capabilities include the GPU API type and version available (e.g., CUDA, OpenGL, etc).

Additionally, the Jobs service will use these enhanced app definitions to execute jobs in a distributed manner to take advantage of data locality and, optionally, to schedule jobs on underutilized systems. An instance of the Tapis Jobs service running at a given institution can be configured to utilize a local security kernel for system credentials, enabling a datacenter to keep all sensitive credentials on premise; see 3.2 for more details.

3.2 Identity, Authorization, Security and Tenancy

Tapis provides a modular authentication subsystem with the goal of achieving sufficient flexibility to enable institutions to integrate their existing identity providers and related systems. Fundamental to the platform is the notion of *tenancy*; a *tenant* in the Tapis framework represents a logical separation of Tapis entities (i.e., apps, jobs, actors, etc.) as well as a high-level authentication and security configuration. A key simplification in our approach is to leverage signed JSON Web Tokens (JWT, [26]) as the single, sole mechanism

for proving identity to any Tapis API. As each tenant is configured with its own public key for token signatures, the entire authentication system can be customized on a per-tenant basis.

The Tapis authentication subsystem is comprised of the following components:

- *Tokens API.* A stateless microservice and reference implementation for generating a properly formatted, signed JWT.
- *Authentication Server.* An OAuth2 and OIDC compliant web server capable of integrating with LDAP servers for authenticating end users and generating signed JWTs using a token API.
- *Tenants API.* Administrative API for managing the registry of tenants globally in a distributed Tapis installation.
- *API Router.* A load balancer and "edge router" capable of routing requests to back-end services, potentially across sites.

Tapis administrators manage the registry of tenants via the Tenants API, and it provides public (even anonymous) endpoints for discovering fundamental properties of a tenant, such as the public key used to sign JWTs for the tenant and the location of the tenant's security kernel. The rest of the components can be viewed as optional, but are used, for example, to provide an OAuth2-compliant authentication system against the primary TACC LDAP server.

The Tapis Security Kernel is a distributed subsystem comprised of open-source software components tied together by a unifying API. Figure 1 shows the main components of the subsystem. The security kernel starts with its inclusion of a secrets store, Vault [Vault], and an authorization service, Shiro [Shiro] and, significantly, its exclusion of an authentication component. The security kernel provides an API that the platform's other microservices use to securely interact with users' storage and execution resources and with each other. Vault manages all secrets in the system, including all passwords and keys, using a fault-tolerant, scalable, highly available cluster of VMs.

The Tapis Security Kernel builds upon Shiro's security framework to create a scalable, fine-grained authorization facility by extending its permissions model with representations of file path names. This extension, along with proper caching, provides a scalable solution to the problem of fine-grained authorization checking across a virtually unlimited, distributed namespace.

The security kernel's API presents a unified interface to secrets and permissions management. Secrets are only accessible to a microservice if the user on behalf of whom the microservice is acting is authorized via Shiro. The API operates on users in multiple tenants, all of which have previously authenticated to the microservice. Multiple instances of the security kernel can exist in the system. Each tenant is assigned a kernel, which may be shared with other tenants or be exclusive to itself. In addition, a kernel instance can reside in a tenant's data center and be locally administered while the rest of the system runs and is administered in a central location. By configuring a local kernel, organizations have the option to keep and administer their secrets in-house.

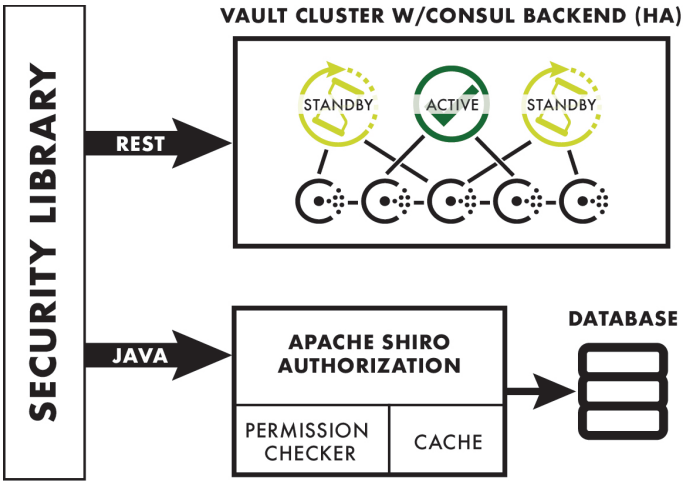

Fig. 1. Tapis Security Kernel

3.3 Support for Streaming Data

The Tapis Streams API provides a production-quality service that builds on top of the CHORDS project for real-time data services and extends the primary data models including *site*, *instrument* and *variable*, with additional metadata including adding spatial indexing and permissions. The Streams API also integrates Tapis event-driven functions (see 3.4) and data management and code execution capabilities (see 3.1) to provide an analysis capability on streaming data sources. Support for streaming data includes the following capabilities:

- Storing and retrieving streaming data for batch job processing, with support for temporal and spatial indexes and queries.
- Automated, event-driven data stream processing workflows with integration into Tapis functions as well as other streaming frameworks.
- Automated data management and scheduled archiving based on programmable policies.

Batch job processing is supported by allowing a Tapis stream, specified as a query, to be the input to a job. In such a case, the Jobs service will schedule the stream data to be transferred to the execution system, manifested as a JSON, CSV or similarly formatted file, prior to launching the app.

Real-time, event-driven workflows can also be supported by the Streams API. Tapis provides this capability through integration with its functions-as-a-service, jobs service and other systems or services that support web-hooks. For these use cases, analysis of incoming data in real time is critical to the success of the experiment. The Streams API will provide the following capabilities to support various levels of analysis based on their computational needs:

- *Alerts as Events* Third-party processing engines will receive real-time notifications when measurement data hits predefined criteria. Subscribers will receive an HTTP POST request containing JSON data detailing the event that triggered the alert.
- *Processing Events with Tapis Functions* - Developers will be able to register an Abaco function with an

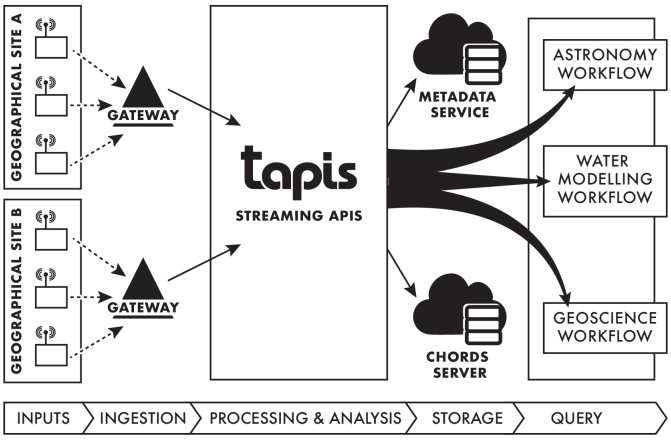

Fig. 2. Tapis Streaming APIs

alert to perform scalable processing with no infrastructure to maintain.

- *Scheduled Relays to Third-Party Systems* - Analogous to the transfers support for batch applications, the streaming data API will be capable of supporting scheduled relays for fixed time intervals to remote services, such as, message brokers and streaming frameworks capable of handling large quantities of data.

In order to support science use cases involving vast amounts of generated data (e.g., many GB/sec), the streaming data API will also provide endpoints for robust data management and archiving policies. For each stream, the data management APIs will support an archiving threshold and a policy. Thresholds will include time-to-live and max sizes (in bytes), and policies will include compressed transfer to a remote storage system and physical delete.

3.4 Functions-as-a-service and Events-Driven Workloads

Tapis will adopt and evolve the NSF funded Abaco (Actor Based Containers) project to provide functions-as-a-service to support event-driven workloads. Abaco is based on the Actor Model of concurrent computation and Docker; users define computational primitives called *actors* with a Docker image, and Abaco assigns each actor a unique URL over which it can receive messages. Users send the actor a message by making an HTTP POST request to the URL. In response to an actor receiving a message, Abaco launches a container from the associated image, injecting the message into the container. Typically, the container execution is asynchronous from the message request, though Abaco does provide an endpoint for sending a message to an actor and blocking until the execution completes, providing synchronous execution semantics. Abaco maintains a queue of messages for each actor, and is capable of launching containers in parallel for a given actor when the actor is registered as *stateless*.

A primary use case is to allow users to develop and register actors to process events in Tapis. Examples of events include new files arriving on a storage system, a job completing on an execution system, either normally or abnormally, or alerts from the streaming data service indicating measurements have hit predefined thresholds, as described in 3.3.

3.5 Container Registries and Advanced Job Scheduling

One of the chief accomplishments of the Docker platform has been to build a catalog of millions of containerized applications freely available for download from its Docker Hub. Other community efforts such as BioContainers and Singularity Hub have followed in Docker's footsteps to provide registries of containerized software for research computing. The Tapis Apps and Jobs services will capitalize on these efforts so that applications stored in these third-party registries will be available for use through the platform.

Leveraging the computational portability features of containers and the system capabilities and application requirements described in 3.1 , the Jobs service provide an optional smart scheduling capability to users interested in minimizing time to solution for a given workload. The smart scheduling option, when enabled for a specific job, will examine the execution systems available to the user whose capabilities meet the requirements of the application. Additionally, the user can provide a whitelist or blacklist of systems to consider or ignore, respectively. For systems meeting the requirements, the Jobs service will initially consider two factors as part of its scheduling decision: the queue time for the execution system and the time to transfer any input data. A mix of real-time data (current queue lengths) and historical data (average transfer rates to the execution system from the storage systems) will be used to estimate the job start time on each system, and the job will be scheduled on the system with the earliest start time. We will consider evolving the scheduling algorithm to use additional information over time.

3.6 Highly Scalable Document Store and Metadata API

A growing number of research projects curate and analyze large collections of highly structured data, with performance requirements regarding time to access, search and move the data. The Tapis Meta API provides a scalable storage solution for structured documents serialized with formats such as JSON or XML. The service aims to support a wide variety of data collections and usage patterns by providing users with a high degree of customization of the backend data store structure. Within a tenant, *collections* can be registered that define their own schema and indexes. Support for geospatial and hashed indexes is included.

In addition to basic create, read, update and delete functionality for documents, the Meta API integrates with the Tapis Security Kernel (see 3.2) to provide a permissions system over the documents themselves. Like other Tapis objects, Meta documents can be private to a single API user or shared with one or more users. Backing the Meta API is a managed MongoDB cluster running on dedicated hardware with high-speed attached storage. The service scales to support large collections - a research group at UT Southwester Medical School is planning to scale an adaptive immune repertoire sequencing data repository to

10B records as part of the iReceptor Plus project [27], a consortium of institutions funded by EU and the Canadian Institutes of Health (see 4.2 for more details).

3.7 DevOps Tooling for Automated Platform Management

Tapis is providing automated deployment and system management to simplify the administration of components at a given site. Based on experience developing the TACC Deployer project [28], [29], the Tapis DevOps tools will:

- Deploy and configure the security kernel.
- Deploy and configure individual Tapis services.
- Provide detailed, actionable logging information.
- Monitor the status of system components.
- Provide an audit trail for security-sensitive operations.

Each Tapis service is packaged into a Docker image, and the official deployment tooling targets the open source Kubernetes container orchestration system. Kubernetes provides a number of important features, including: service deployment scheduling onto one or more compute nodes, networking and discovery between services in the same Kubernetes namespace, service configuration management via Kubernetes Configmaps, persistent storage via Kubernetes Persistent Volume Claims (PVCs), and basic service health monitoring and restart policies. Moreover, Kubernetes serves as a common denominator across different datacenters and deployment environments, and is emerging as the dominant container orchestration technology. As an open source project, Kubernetes is easy to install in an academic datacenter, but at the same time, all major commercial cloud platforms, including AWS, Azure and Google Cloud Platform, provide a "Kubernetes-as-a-service" solution. The Tapis project is planning to utilize a small cluster in the Azure cloud for testing our provisioning tools in a sandbox environment.

4 Early Science Adopters

4.1 Real Time Climate Data for the Hawaiian Islands.

Climate observations in Hawaii are critically important to understand past, current, and future climate of the nation and the globe. Despite Hawaii's importance, national climate data compilations and analyses include only sparse and inadequate coverage of the Hawaiian Islands. As the only US state within the tropics, Hawai'i also provides a model system for studying the functional responses of tropical terrestrial ecosystems, including tropical rain forests, dryland forests, and agriculture, to changes in atmospheric CO2 and climate. Up to date, high quality, reliable climate information in Hawaii is of great value to the nation.

Tom Giambelluca's research group at the University of Hawaii is working on automated workflows to provide ready access to a spatially comprehensive, high quality, reliable climate data set and data analysis products covering the Hawaiian Islands. The climate researchers will use Tapis to: (1) gather and centralize climate data from all known sources and new monitoring stations using Tapis data streams; (2) update the data archive to maintain near-real-time status with Tapis triggered events for Tapis functions and containerized process workflows; (3) automate basic data screening, homogeneity testing, and gap filling using Tapis functions; (4) automatically update statistical analysis of available data by site and produce daily near-real-time, high-resolution, gridded digital map products and reports of climate variables (e.g., precipitation, temperature, humidity, wind speed, solar radiation) using Tapis containerized workflows scheduled on HPC or cloud resources; (5) share data and products with the wider community through Tapis data management and sharing services.

4.2 iReceptor Plus: Human Immunological Data Storage, Integration and Controlled Sharing

Increasingly, scientists depend on next generation sequencing (NGS) data to understand the immune response in autoimmune and infectious disease as well for developing vaccines. The iReceptor Plus platform provides a science gateway linking to federated, geo-distributed repositories of NGS data conforming to adaptive immune receptor repertoire (AIRR) community standards. Co-funded by the European Union and the Canadian Institutes of Health Research, iReceptor Plus constitutes an international consortium of institutions providing AIRR datasets as well as an API for access.

The Cowell Lab at the University of Texas Southwestern Medical Center is leveraging the Tapis Meta service for storage and management of their AIRR dataset. An initial prototype effort in which 350 million records were created in the Tapis Meta service has yielded promising results. The Cowell group plan to grow the collection to ten billion records during the four year iReceptor Plus project. They are also considering using Tapis for high-throughput analysis jobs at TACC and Compute Canada.

4.3 Planet Texas 2050

The Planet Texas 2050 group wants to leverage spatially dense and ever-increasing Geoscience temporal datasets generated by Arduino-based microcontrollers as ground-truth inputs for integrated models of water-land-atmosphere-urban systems [30]. Deployments collect hydrological and atmospheric measurements to feed models describing various processes related to flooding and aquifer recharge.

Work is already underway to leverage the Tapis Streams service for real-time data from the Adruino-based microcotrollers using the microcontroller API associated with Particle Argon and Xenon mesh series devices. Future work involves hardening this preliminary effort into a production-grade service and scaling up the size of the sensor deployment. Additionally, a workflow will be built to automate the process of converting raw stream data to dataframes for integrated modelling analysis.

4.4 Ocean Wave Fingerprinting and Automatic Wave Forecasting

Oceanic and atmospheric processes leave their imprint, much like a fingerprint, on the ocean surface. These fingerprints can be used to understand air-sea interaction, ocean wave propagation, wave-ice interaction, and small-scale

atmospheric weather phenomena. Justin Stopa, UH Ocean Engineering uses high resolution, synthetic aperture radar (SAR) from the Sentinal-1 satellites to study sea surface roughness at very fine spatial resolution (>10 m). SAR provides high-resolution data regardless of environmental conditions and provides information at a global scale capturing approximately 200-400 Gb per platform/day. Manual classification is impractical for the S1 database (120,000 images per month) and machine learning techniques and efficient data workflows are required to make best use of this data. The Stopa team is also developing a real-time wave forecasting system for mitigation of damaging events such as tropical cyclones or large swells, trans-oceanic commerce, and recreation activities. This system requires information from satellites, in-situ sensors, and models.

In-situ sensors include buoy observations that measure wind speeds, wave spectra (energy as a function of frequency and direction), or air-sea temperature difference. Model information includes wave forecast sea state parameters which are usually gridded in time and space. Tapis will be used to support both the classification of satellite images via containerized ML workflows and the automatic wave forecasting workflow. The forecast model is launched when new data is posted by the major weather forecast offices that run global atmospheric and oceanic simulations. Wind speeds, ice concentrations, air-sea temperature differences, and sometimes ocean currents are required forcing fields that drive a spectral wave model such as WAVEWATCH3 (WW3). Regional (e.g. state of Hawaii) and local (e.g. Oahu, island-scale) information, is needed to resolve the wave transformations around the intricate bathymetry of Hawaii (e.g. [31], [32]). WW3 (> V5.0) is extremely efficient at linking and nesting multiple wave grids internally (e.g. global and regional models). The Simulating Waves Nearshore (SWAN) spectral wave model is more efficient in coastal environments (island scales 1-100 km) when the spatial scales become small due to its implicit numerical scheme to solve the governing equation. The spectral wave information and forcing fields must be passed from WW3 to SWAN. Since the wave model data could miss details of the wave field, the observations provide complementary information. Tapis functions can be triggered to pull auxiliary datasets from models, other satellite observations or sensors as additional inputs to the appropriate algorithm based on the earlier classification. The results of these calculations can become secondary Tapis data streams that provide geophysical parameters ([33]) that will be used to build automated algorithms using Tapis containers and HPC jobs to move the data from the storage systems to the compute system to generate new products such as swell fields by back-propagating swell components (direction and wavenumber) and the forward associated swell components with a storm source (e.g. [34]; [35]).

5 System Architecture

Tapis capabilities are organized into a set of microservices, where each service provides an independent API over HTTP, designed in a RESTful style. API contracts are codified using OpenAPI v3 definitions, and these definitions are used to generate live docs and initial client libraries in Python and Java. Most services also communicate with one or more databases and/or message queues using direct connections, and with other Tapis services over HTTP. All service requests - both external requests and requests from other Tapis services - are authenticated using a JSON Web Token (JWT). As part of initialization, a Tapis service retrieves a JWT representing itself from the Tapis Tokens API.

Tapis supports deployment of its components to multiple sites or physical locations. The components at these sites "work together" to comprise a single distributed Tapis installation. Each distributed installation will have a single *primary site* and zero or more *associate sites*. The primary site has the following roles and responsibilities:

- The primary site runs one or more instances of every (primary) Tapis service.
- The primary site runs a copy of each database needed by the primary Tapis services. The Tapis services running in the primary site communicate with these databases and not any other databases running at external associate sites.
- The primary site runs the Tenants API (see description of multi-tenancy below); no other site in the installation runs the Tenants API.

In addition to one or more Tapis services, each site runs a Tapis API Router (or "edge router") component. The Tapis API Router is responsible for forwarding requests to specific Tapis microservices.

Tapis also supports multi-tenancy: a *tenant* is a logical separation of Tapis objects for a specific project or group. Tapis objects (e.g. systems, apps, jobs, streams, etc.) in one tenant cannot be accessed from another tenant. Tenants have the following properties:

1) A universe of user accounts, ultimately linked to an LDAP query or some other identity store.
2) A base URL for all (public) API requests (e.g., https://api.sd2e.org).
3) A base URL for the location of any services running at the primary site (e.g., https://sd2e.api.tapis.io).

Tenants and associate sites are related through a hub and spoke model (see Figure 3) based on which site is hosting the tenant's base URL. There are two cases: If the primary site hosts the tenant's base URL, then all Tapis services for that tenant are hosted at the primary site. Alternatively, an associate site can host the tenant's base URL. In this case, the associate site's API Router is responsible for routing service requests either to a service running locally at the associate site or to a service running at the primary site. To route to the primary site, the associate site configures its API Router to route requests to the tenant's primary site base URL, defined in 3) above. Thus, all services for a given tenant are hosted either exclusively at the primary site, or between the primary site and one associate site. No cross-service requests or forwarding between associate sites occurs.

6 Tapis Road Map

Tapis is funded by a five year Cyberinfrastructure for Sustained Scientific Innovation (CSSI) Framework grant from the National Science Foundation, which began September 1,

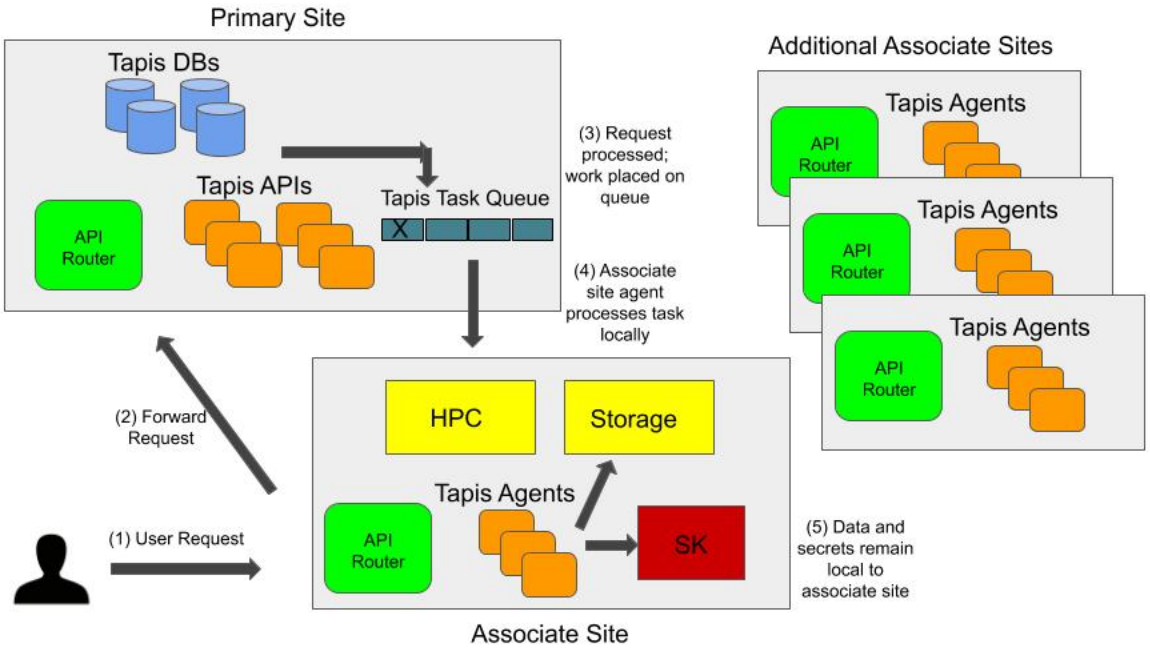

Fig. 3. Multi-Site Tapis Deployment.

Each site represents a physical datacenter. (1) A user's request is first resolved to a site based on the *base URL*. Based on configuration, the site's API Router routes the request to the Primary Site (2) which processes the request and puts work on a shared Task Queue (3). An agent at the Associate Site takes the work unit (4) and consults its own Security Kernel while processing the task (5).

2019. The project team has defined specific milestones to be delivered across the five project years, summarized in the following table:

Year	**Real-Time Capabilities**	**Batch Capabilities**	**Security and Authz**
Year 1	Production release of streaming API	System, Application and Job abstractions	Security Kernel initial release.
Year 2	Data management and archiving features	Support for third-party registries	Automated devops tools for security kernel
Year 3	Support for Alerts as Events	Gateway in a Box release	Local caching solution for security kernel
Year 4	Support for Abaco functions	Smart Jobs scheduling, initial release	Automated devops tools for hybrid and HA deployment
Year 5	Support for scheduled relay streams	Expanding support and features for existing services	Devops for monitoring and log aggregation

Additionally, an Early Adopters workshop is planned for the end of the first project year, targeting the July, 2020 time frame. During the workshop, Tapis core teammembers will present talks on the system capabilities developed, and early adopters will be invited to present their use cases, data sets, and adoption of the platform to date.

7 Acknowledgments

This material is based upon work supported by the National Science Foundation Office of Advanced CyberInfrastructure, award numbers 1931439 and 1931575.

References

[1] R. Dooley *et al.*, "Software-as-a-service: The iplant foundation api," IEEE. 5th IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS), 2012.

[2] J. Stubbs *et al.*, "Rapid development of scalable, distributed computation with abaco," Science Gateways Community Institute. 10th International Workshop on Science Gateways, 2018.

[3] B. Kerkez *et al.*, "Cloud hosted real-time data services for the geosciences (chords)." Geoscience Data Journal, 2016, pp. 2–4.

[4] N. Merchant *et al.*, "The iplant collaborative: Cyberinfrastructure for enabling data to discovery for the life sciences," *PLOS Biology*, 2016.

[5] N. Merchant, E. Lyons, S. Goff, M. Vaughn, D. Ware, D. Micklos, and P. Antin, "The iplant collaborative: cyberinfrastructure for enabling data to discovery for the life sciences," *PLoS biology*, vol. 14, no. 1, p. e1002342, 2016.

[6] S. A. Goff, M. Vaughn, S. McKay, E. Lyons, A. E. Stapleton, D. Gessler, N. Matasci, L. Wang, M. Hanlon, A. Lenards *et al.*, "The iplant collaborative: cyberinfrastructure for plant biology," *Frontiers in plant science*, vol. 2, p. 34, 2011.

[7] E. M. Rathje, C. Dawson, J. E. Padgett, J.-P. Pinelli, D. Stanzione, A. Adair, P. Arduino, S. J. Brandenberg, T. Cockerill, C. Dey *et al.*, "Designsafe: new cyberinfrastructure for natural hazards engineering," *Natural Hazards Review*, vol. 18, no. 3, p. 06017001, 2017.

[8] S. B. Cleveland *et al.*, "The 'ike wai gateway- a science gateway for the water future of hawai'i"," Science Gateways Community Institute. Proceedings of Science Gateways 2018, Austin TX, USA September 2018, 2018.

[9] E. Litvina, A. Adams, A. Barth, M. Bruchez, J. Carson, J. Chung, K. Dupre, L. Frank, K. Gates, K. Harris, and H. Joo, "Brain initiative: Cutting-edge tools and resources for the community." *Journal of Neuroscience*, vol. 39, no. 42, pp. 8275–84, 2019.

[10] N. Wilkins-Diehr, M. Zentner, M. Pierce, M. Dahan, K. Lawrence, L. Hayden, and N. Mullinix, "The science gateways community institute at two years," in *Proceedings of the Practice and Experience on Advanced Research Computing*. ACM, 2018, p. 53.

[11] (2019) Vdjserver. Last access: 2019-10-30. [Online]. Available: http://vdjserver.org

[12] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PloS one*, vol. 12, no. 5, p. e0177459, 2017.

[13] W. C. Proctor, M. Packard, A. Jamthe, R. Cardone, and J. Stubbs, "Virtualizing the stampede2 supercomputer with applications to hpc in the cloud," in *Proceedings of the Practice and Experience on Advanced Research Computing*, 2018.

[14] (2019) Hashicorp vault. Last access: 2019-10-30. [Online]. Available: https://www.vaultproject.io/

[19] (2019) hubzero. Last access: 2019-10-30. [Online]. Available: https://hubzero.org/

[15] (2019) Apache shiro. Last access: 2019-10-30. [Online]. Available: http://shiro.apache.org/

[16] (2019) Apache airavata. Last access: 2019-10-30. [Online]. Available: https://airavata.apache.org/index.html

[17] (2019) Galaxy community hub. Last access: 2019-10-30. [Online]. Available: https://galaxyproject.org/

[18] (2019) Globus. Last access: 2019-10-30. [Online]. Available: https://www.globus.org/

[20] T. Gottdank, *Introduction to the WS-PGRADE/gUSE Science Gateway Framework*. Cham: Springer International Publishing, 2014, pp. 19–32. [Online]. Available: https://doi.org/10.1007/978-3-319-11268-8_2

[21] E. Deelman *et al.*, "Pegasus: a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015. [Online]. Available: http://pegasus.isi.edu/publications/2014/2014-fgcs-deelman.pdf

[22] K. Wolstencroft *et al.*, "The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud," *Nucleic Acids Research*, vol. 41, Jul 2013.

[23] S. Padhy, G. Jansen, J. Alameda, E. Black, L. Diesendruck, M. Dietze, P. Kumar, R. Kooper, J. Lee, R. Liu *et al.*, "Brown dog: Leveraging everything towards autocuration," in *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 493–500.

[24] (2019) Sciserver. Last access: 2019-10-30. [Online]. Available: http://www.sciserver.org/

[25] (2019) Dataturbine. Last access: 2019-10-30. [Online]. Available: http://dataturbine.org/

[26] (2018) Linux pam. Last access: 2018-03-17. [Online]. Available: https://github.com/linux-pam/linux-pam

[27] (2019) ireceptor plus. Last access: 2019-10-30. [Online]. Available: https://www.ireceptor-plus.com/

[28] J. Stubbs *et al.*, "Tacc's cloud deployer: Automating the management of distributed software systems." The 2nd Industry/University Joint International Workshop on Data Center Automation, Analytics, and Control (DAAC). Supercomputing., 2018. [Online]. Available: https://drive.google.com/file/d/1oORwQdQEWTHLpARVJPzQqR_0OY5SOrfg/view

[29] S. B. Cleveland *et al.*, "Building science gateway infrastructure in the middle of the pacific and beyond: Experiences using the agave deployer and agave platform to build science gateways." Proceedings of the Practice and Experience on Advanced Research Computing, PEARC 2018, 2018.

[30] J. Powell, J. Stubbs, S. Cleveland, S. Pierce, and M. Daniels, "Streamed data via cloud-hosted real-time data services for the geosciences as an ingestion interface into the planet texas science gateway and integrated modeling platform." Science Gateways Community Institute. Proceedings of Science Gateways 2019, San Diego, CA, USA September 2019, 2019.

[31] J. Stopa, K. F. Cheung, and Y.-L. Chen, "Assessment of wave energy resources in hawaii." *Renewable Energy*, vol. 36, no. 2, pp. 554–567, 2011.

[32] N. Li, K. Cheung, J. Stopa, F. Hsiao, Y.-L. Chen, L. Vega, and P. Cross, "Thirty-four years of hawaii wave hindcast from downscaling of climate forecast system reanalysis." *Ocean Modelling*, no. 100, pp. 78–95, 2016.

[33] J. E. Stopa and A. Mouche, "Significant wave heights from sentinel1 sar: Validation and applications," *J. Geophys. Res. Oceans*, vol. 122, pp. 1827–1848, 2017.

[34] F. Collard, F. Ardhuin, and B. Chapron, "Monitoring and analysis of ocean swell fields from space: New methods for routine observations." *JGR-Oceans*, vol. 114, no. C7, 2009.

[35] J. Stopa, F. Ardhuin, R. Husson, H. Jiang, C. B., and F. Collard, "Swell dissipation from 10 years of envisat asar in wave mode." *GRL*, 2016.