Federated Galaxy: Biomedical Computing at the Frontier

Enis Afgan,
Department of Biology, Johns Hopkins University, Baltimore, MD, USA

Vahid Jalili,
Department of Biomedical Engineering Oregon Health and Science University Portland, OR, USA

Nuwan Goonasekera,
Melbourne Bioinformatics University of Melbourne Melbourne, VIC, Australia

James Taylor,
Departments of Biology amd Computer Science Johns Hopkins University Baltimore, MD, USA

Jeremy Goecks
Department of Biomedical Engineering Oregon Health and Science University Portland, OR, USA

Abstract

Biomedical data exploration requires integrative analyses of large datasets using a diverse ecosystem of tools. For more than a decade, the Galaxy project (https://galaxyproject.org) has provided researchers with a web-based, user-friendly, scalable data analysis framework complemented by a rich ecosystem of tools (https://usegalaxy.org/toolshed) used to perform genomic, proteomic, metabolomic, and imaging experiments. Galaxy can be deployed on the cloud (https://launch.usegalaxy.org), institutional computing clusters, and personal computers, or readily used on a number of public servers (e.g., https://usegalaxy.org). In this paper, we present our plan and progress towards creating Galaxy-as-a-Service—a federation of distributed data and computing resources into a panoptic analysis platform. Users can leverage a pool of public and institutional resources, in addition to plugging-in their private resources, helping answer the challenge of resource divergence across various Galaxy instances and enabling seamless analysis of biomedical data.

Keywords

cloud bursting; data federation; service computing

I. Introduction

The Galaxy application [1] is an open-source web framework that integrates existing data analysis tools and provides biologists with a comprehensive and scalable system to perform computational analysis. It is one of the most popular platforms for biomedical data analysis in the world, with 5,500 available tools and over 5,000 academic papers referencing the

enis.afgan@jhu.edu .

platform (https://bit.ly/gxyStats). In addition to tool integration and provenance tracking for reproducibility, Galaxy includes a workflow system for automated multi-step analyses, a visualization framework including visual analytics capabilities, and facilities for sharing and publishing analyses [2]. These features are coupled with extensive, publicly available documentation and training materials [3]. A unique feature of Galaxy is its web-based user interface, which enables anyone, regardless of their informatics expertise, to run complex biological data analyses at scale using only a web browser.

Many institutions and labs around the world have taken on the effort of deploying and maintaining instances of Galaxy, often freely available for use not only to their local researchers but also to the world-wide community. Currently, there are approximately 100 such free and public servers. The most popular such server is known as Galaxy Main, available at usegalaxy.org. Hosted by the Galaxy project using XSEDE resources [4], it contains nearly 1,000 installed tools, which were used by over 125,000 registered users to run over 18M jobs generating in excess of 2PB of data since the service was made available in year 2006. Additionally, Galaxy supports execution in the cloud where, through a self-service model, researchers and groups can instantiate their own, preconfigured, and dynamically scalable instances of Galaxy through a web browser [5]. Fig. 1 shows various Galaxy infrastructure deployment options.

While this ecosystem of public servers has fostered a vibrant and growing community of users, it is also increasingly apparent that, in its current form, no individual Galaxy server can meet the significant computational demand (storage and jobs) and desire for customization (variable toolset), leading to a multiserver, quota-based system that apportions available resources among its users. This has led to: (a) a non-trivial maintenance burden on institutions that host a Galaxy server, mainly because Galaxy depends on a complex ecosystem of software, such as a job scheduling engine (e.g. Slurm [6], PBS [7]), a database, a high-performance file system, etc.; (b) the fragmentation of the Galaxy ecosystem, with some servers running outdated versions of Galaxy; (c) a profusion of choice for the user, who must be cognizant of available data, toolset, and usage quotas available on a given instance; and (d) the necessity for a user to install their own instance if no public server is suitable for their current analysis needs (see Figure 1). This is at odds with the goal of accessibility, one of the core pillars that underpin Galaxy (in addition to reproducibility and transparency) [8].

With this in mind, we have been working on unifying the Galaxy ecosystem. Instead of multiple Galaxy instances being independently hosted and managed, we envision fewer instances (perhaps just a single public one) that dynamically integrates distributed storage and compute resources, installing tools, reference-data, and other dependencies, on-the-fly (see Figure 1). The unique feature in this vision, that differs from traditional SaaS and cloud-bursting models, is that we intend to federate resources from a variety of providers (e.g., clouds, department clusters, supercomputing centers) and run the user's analysis on these resources, allowing researchers to extend the capacity of an existing Galaxy instance by bringing additional (i.e., their own) resources. This diverse, user-centric bursting model will be available to all users through Galaxy's web-based user interface and will not require any technical expertise to run distributed analyses. Another key advantage of this model is that sensitive data (e.g., genomic data used in medical diagnoses) will be processed in place behind institutional firewalls. Such a federated execution approach will also improve job performance by leveraging data locality. In this paper, we describe the core components in this vision and present our progress to date.

II. Galaxy-as-a-Serivce Vision

Like with many other *-as-a-Service* notions, our aim is to provide a higher level of abstraction for interacting with Galaxy while maintaining much of the flexibility Galaxy is known for. Currently, to use Galaxy, a researcher can: (a) utilize a public Galaxy instance; (b) deploy a private server on a cloud provider; or (c) install their own server (see Figure 1). These three options provide increasing control over the server but also increased maintenance; for a public instance, a user must conform to the imposed usage quotas and available toolset, but there is no maintenance required. Installing one's own instance yields maximum flexibility over tools installed and hardware used but at the expense of instance installation, configuration, and maintenance. The cloud option falls somewhere in between.

With the Galaxy-as-a-Service (GaaS) concept, we are creating a federated Galaxy that enables distributed and cloud-centric analyses through the Galaxy user interface that maximizes usability, minimizes required software and infrastructure maintenance, and enables data-local computing with already available resources. This will be achieved by offering an always-on instance that will exhibit all the benefits of a managed software service where the researcher can simply access the service via URL and start interacting with the application, data, and tools without having to maintain the underlying software or hardware. In the background, Galaxy will—in addition to any free resources offered by the server—acquire additional compute and storage resources on the user's behalf and automatically integrate them into the resource pool available to the specific user. Internally, Galaxy will route only the specific user's jobs to the newly allocated resources (and allow the user to share the given resource pool with collaborators). This additional resource allocation will be based on a user's session during which the user can explicitly and easily instruct Galaxy to acquire the additional capacity (see Fig. 2). Once the session ends, Galaxy will release the computing resources and keep only the data storage resources needed to store analysis results. The resource acquisition will be performed using the user's credentials, with the ability to acquire resources from a variety of infrastructure providers (see Section III.C). Initially, we are focusing on cloud providers but gradually we will add support for linking hosted infrastructure as well (e.g., institutional clusters), supporting multi-cloud scenarios [9]. Overall, this will have the benefit of a single Galaxy instance being able to expand its capacity beyond what a typical public resource can supply to the community while maintaining a central location for data access and sharing.

The described GaaS concept supports several appealing use cases. The most recognizable use case is the ability to use a managed, public Galaxy instance without the resource limitations of a shared server. Further, the ability to acquire additional resources allows for more efficient computing. Biomedical data is highly distributed so acquiring and using additional compute resources near the data—including private data—presents an opportunity to reduce runtime and cost.

III. Required Architectural Components

Realizing the GaaS vision requires a significant technological investment not only within the Galaxy application but also increasingly in the surrounding ecosystem. We present three critical architectural layers required for GaaS: compute and storage infrastructure and their integration via state-of-the-art authentication and authorization mechanisms (Fig. 3.A). The three components act as abstractions through which the Galaxy application can reason and interact with the highly heterogeneous pool of resources and providers (Fig. 3.B). The following subsections describe those abstraction layers.

A. Authentication and Authorization

We are adding means of federating users identities (single sign-on), such that users can login to any of the Galaxy instances or resources using a common identity. Once logged in, users will have access to a project-centric view of Galaxy, with access to projects and resources governed by a project administrator. By default, users will be enrolled into a default project, with capabilities and quota restrictions mirroring that of Galaxy Main. However, users will be able to create their own projects and plug-in custom resources into their project, such as public cloud storage or institutional compute resources (see Sections III.B and III.C). Project owners will be able to invite other users to their project, and invited users will be eligible to utilize these extra resources transparently, with no configuration required, just as if they were accessing an institutional Galaxy server.

When users submit a job, their identity will be used to determine the resources that the user is eligible to access, and the job will be routed to the appropriate destination. At each point of access to compute or data, the federated identity will be used to determine access rights, ensuring that users only access compute and storage resources for which have been authorized.

As an initial step towards this goal, we are developing the CloudAuthz library (https://github.com/galaxyproject/ cloudauthz), which leverages the popular authentication and authorization (authnz) protocols OAuth2 and OpenID Connect to enable user logins using third-party identity providers such as their institution or major identity providers (e.g., Google, Microsoft, Amazon). Later, we will add a project management layer to Galaxy, which will integrate with the CloudAuthz library to provide system-wide authnz for resource access.

B. Compute

We have recently integrated NSF's Jetstream cloud (https://jetstream-cloud.org/) [10] with Galaxy Main as a pilot for validating multi-cloud computing in Galaxy. Galaxy's analysis jobs are routed to Jetstream while the data transfer and job management are managed by the Galaxy framework. In the initial 2 years Galaxy Main has run ~225,000 Jetstream jobs from >13,000 distinct users, serving as a successful proof of concept for remote cloud job execution at scale.

To generalize this work to a wide variety of target compute infrastructures, we have prototyped a software stack to interface with multiple cloud and infrastructure providers in a uniform fashion and provide high-level runtime environments. Collectively called CloudVE (Virtual Environment), this stack acts as the foundation for infrastructure-agnostic computing for Galaxy and is composed of the following layers:

- **CloudBridge** [11] is a Python library with a uniform and well tested API over popular IaaS cloud providers so that the same API calls (e.g. create a file, start a virtual machine) will perform the same operations across all supported clouds (unlike Apache Libcloud (https://libcloud.apache.org/) or T erraform (https://www.terraform.io/), which require conditional, vendor-specific code);
- **CloudLaunch** [12] leverages CloudBridge, and has the ability to deploy arbitrary applications onto cloud infrastructures or hosted machines in a consistent manner. This is achieved by capturing the application deployment process in a simple, portable manner;
- **CloudMan** [13] manages a runtime environment on the underlying infrastructure by creating structure (e.g., a cluster) over individual machines. This includes the ability to dynamically scale up or down to ensure the optimal utilization of resources. A new version of CloudMan in development builds on Docker and Kubernetes, which are mature, proven technologies well-suited for writing portable and versioned applications;
- **HelmsMan** is envisioned as an application manager (e.g., start, stop, upgrade applications). It builds on Helm, a Kubernetes package manager, and can readily deploy suites of applications as a unit. For GaaS, HelmsMan will deploy and manage Pulsar as a remote job manager for Galaxy, a cluster job manager (e.g., Slurm, HTCondor), a Postgres database, and a web server.

Jointly, these layers will facilitate deployment of a runtime environment for Galaxy on a variety of computational resources. While we are initially focused on consuming IaaS cloud resources (because of the high degree of control), our approach makes it readily possible to integrate compute resources from managed infrastructures, such as hosted servers or clusters. In such cases, a remote client (i.e., an instance of CloudMan) is launched as a local, standalone application, which will orchestrate the required services for interacting with Galaxy (e.g., Pulsar) and configure it to attach to the local resources.

C. Storage

In addition to authnz and compute, data storage is a crucial component of the overarching GaaS concept. Traditionally, Galaxy has required a centralized view of the data with all the data being local to the current cluster, available over a shared file system. However, genomic data is becoming increasingly large and geographically distributed as DNA sequencing technologies make it easy for anyone to generate large datasets. For this reason, a suitable data model for GaaS needs to support federated data stores. Further, a public Galaxy instance cannot supply sufficient storage capacity for a large community. To address these issues, we are working on a distributed and pluggable mechanism to handle user data within Galaxy. Called the *Galaxy Object Store,* this is a data virtualization layer that allows data to be physically disconnected from a Galaxy instance. Through the Object Store interface, Galaxy can readily access data on local or cloud-based storage resources. Currently, we have implemented interfaces for local or network disk, hierarchical disk (as a rule-based composition of multiple disks), and all major cloud storage providers (AWS S3, Azure Blob, Google Storage, and OpenStack Swift implemented via the CloudBridge library). This implementation allows a Galaxy instance to offload data storage onto remote resources while the internal middleware will automatically retrieve required datasets and place the new ones into the appropriate storage location.

The current Galaxy Object Store supports operations on remote datasets from a single location controlled exclusively by Galaxy, but the GaaS concept requires enabling individual users to associate any number of personal storage resources from a variety of storage providers with their Galaxy account. This will allow users to access and store data on self-provisioned resources, thus overcoming any storage restrictions imposed by instance quotas. In combination with the remote job execution described in Section III.B, this model will allow Galaxy to run jobs local to the data, allowing analysis of private data, reducing data transfer, and improve job performance by leveraging data locality during job scheduling. When the data needs to be moved to a job site that differs from the current location of the data, the Object Store module will either download the data via the storage API endpoints or use a data transfer service such as Globus Transfer [14] or Aspera (http://asperasoft.com/).

IV. Discussion

The components presented in Section III interact with each other to deliver the vision of GaaS. Fig 3.B captures their interactions: after a user has associated identities with their account, they can (1) add storage resources and (2) attach transient compute infrastructure associated with their identity to their active data analysis session. This process involves (3) creating any number of job environments, managed by CloudMan, which ensures a suitable job runtime environment exists that Galaxy can interface with (by default, using Galaxy's remote job execution engine Pulsar). Once the user submits jobs, (4) Galaxy routes them to the appropriate environment where jobs, via the Galaxy Object Store interface, interact with the job data. (5) Job runtime environment is conducive to the tool requirements and, upon job completion, (6) the output data is persisted in user's storage. A user will be able to associate multiple environments with their session and also share their resources with collaborators. The sharing option also makes it possible to have community resources shared between all the users of the system while using the same deployment principles.

This federation model fits extremely well with publicly shareable data, but potential issues arise when working with sensitive data, such as patient health information. While a federated model can ensure that private data does not leave institutional resources, it is not possible to have a centralized Galaxy service which does not have access, at the very least, to some metadata, because one of Galaxy's primary functions is to store history and provenance records of an analysis to ensure reproducibility. Therefore, if an institution has strict data movement policies where even metadata leakage is a serious security risk, then the GaaS model can no longer be applied in such cases. The alternative in such cases is to deploy an institutional GaaS—which will provide similar federation capabilities within an institution, alleviating the potential security concerns of a public, global, GaaS.

Acknowledgment

This project is supported through NIH grants HG006620, HG005133, HG004909 and HG005542 as well as NSF grants DBI 0543285, 0850103, 1661497, and Australian National Data Service grant number eRIC07.

References

[1]. Afgan Eet al., "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update," Nucleic Acids Res, vol. 44, no. W1, pp. W3–W10, 2016. [PubMed: 27137889]

[2]. Goecks J, Coraor N, Nekrutenko A, and Taylor J, "NGS analyses by visualization with Trackster," Nature Biotechnology, vol. 30, no. 11. pp. 1036–1039, 2012.

[3]. Batut Bet al., "Community-driven data analysis training for biology," bioRxiv, p. 225680, 2017.

[4]. Towns Jet al., "XSEDE: Accelerating scientific discovery," Comput. Sci. Eng, vol. 16, no. 5, pp. 62–74, 2014.

[5]. Afgan Eet al., "Harnessing cloud computing with Galaxy Cloud," Nat. Biotechnol, vol. 29, no. 11, pp. 972–974, Nov. 2011. [PubMed: 22068528]

[6]. M. G. ABYJette Morris A., "SLURM: Simple Linux Utility for Resource Management."

[7]. Feng H and Rubenstein D, "PBS : A Unified Priority-Based Scheduler," Electr. Eng, pp. 203–214, 2007.

[8]. Goecks Jet al., "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," Genome Biol, vol. 11, no. 8, 2010.

[9]. Nadjaran Toosi A, Calheiros RN, and Buyya R, "A Interconnected Cloud Computing Environments: Challenges, Taxonomy and Survey," ACM Comput. Surv. V, N, Artic. A, vol. 47, no. 1, pp. 1–47, 2013.

[10]. Stewart CAet al., "Jetstream: a self-provisioned, scalable science and engineering cloud environment," in Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE '15, 2015, pp. 1–8.

[11]. Goonasekera N, Lonie A, Taylor J, and Afgan E, "CloudBridge: A simple cross-cloud python library," in ACM International Conference Proceeding Series, 2016, vol. 17–21-July.

[12]. Afgan E, Lonie A, Taylor J, and Goonasekera N, "CloudLaunch: Discover and Deploy Cloud Applications," Futur. Gener. Comput. Syst.

[13]. Afgan E, Baker D, Coraor N, Chapman B, Nekrutenko A, and Taylor J, "Galaxy CloudMan: delivering cloud compute clusters," BMC Bioinformatics, vol. 11 Suppl 1, p. S4, 2010.

[14]. Liu B, Sotomayor B, Madduri R, Chard K, and Foster I, "Deploying bioinformatics workflows on clouds with galaxy and globus provision," in Proceedings - 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, SCC 2012, 2012, pp. 1087–1095.

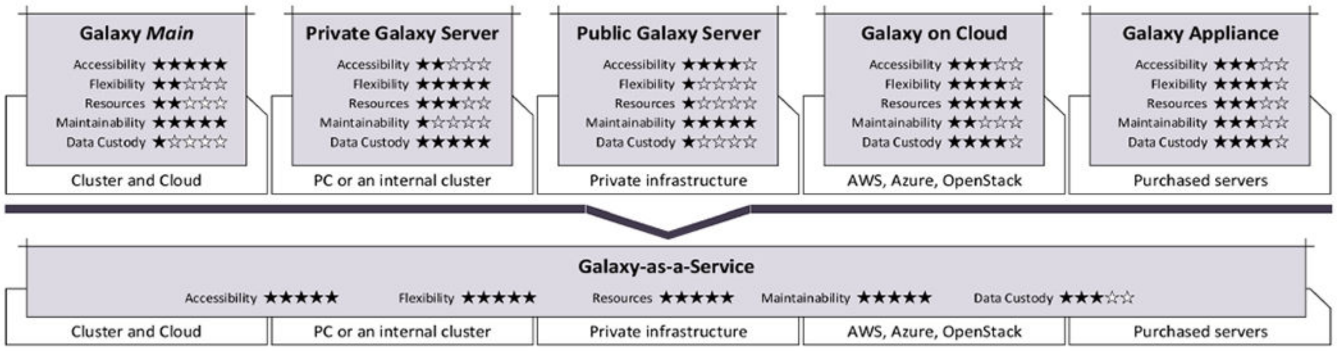

Fig. 1.
The planned evolution of Galaxy from a set of fragmented instances to a service-based solution with dynamically pluggable infrastructure.

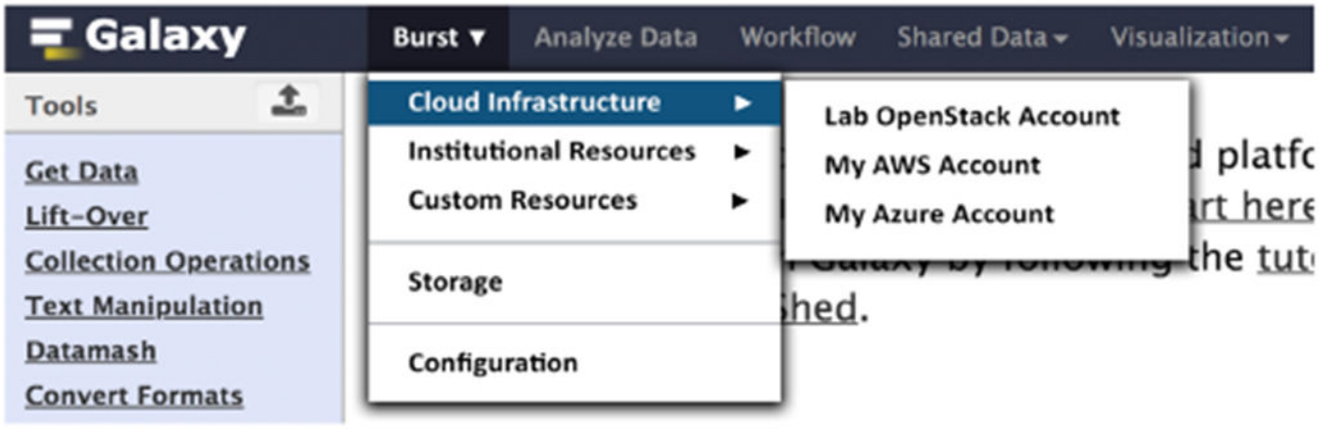

Fig. 2.
A tentative interface of linking additional resources by a user.

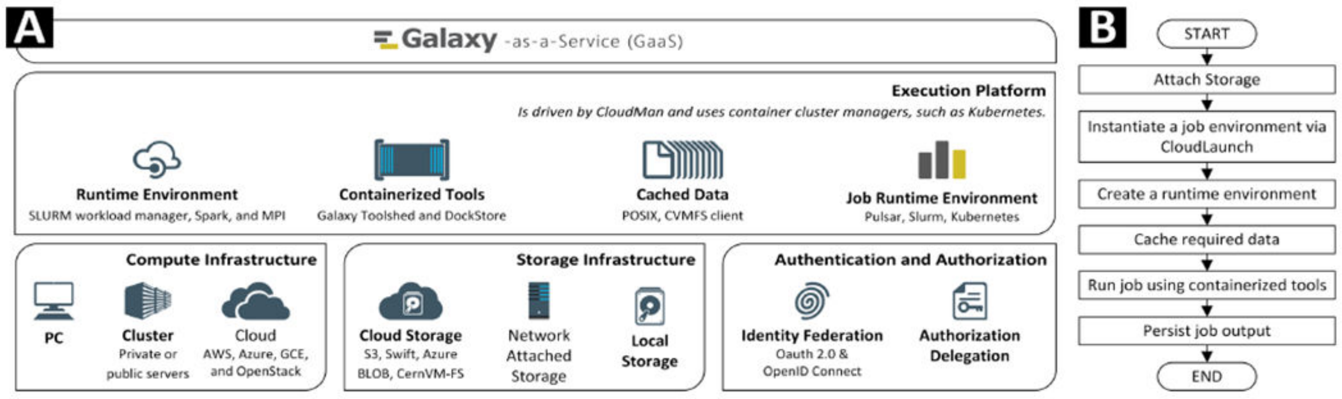
A
Galaxy -as-a-Service (GaaS)
Execution Platform
Is driven by CloudMan and uses container cluster managers, such as Kubernetes.
Runtime Environment
SLURM workload manager, Spark, and MPI
Containerized Tools
Galaxy Toolshed and DockStore
Cached Data
POSIX, CVMFS client
Job Runtime Environment
Pulsar, Slurm, Kubernetes
Compute Infrastructure
PC
Cluster
Private or public servers
Cloud
AWS, Azure, GCE, and OpenStack
Storage Infrastructure
Cloud Storage
S3, Swift, Azure BLOB, CernVM-FS
Network Attached Storage
Local Storage
Authentication and Authorization
Identity Federation
Oauth 2.0 & OpenID Connect
Authorization Delegation
B
START
Attach Storage
Instantiate a job environment via CloudLaunch
Create a runtime environment
Cache required data
Run job using containerized tools
Persist job output
END

Fig. 3.

(A) Conceptual architecture of components making up GaaS, their matching technologies, and (B) simplified invocation flow.