DOI 10.12694/scpe.v16i1.1058
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 19–35. http://www.scpe.org

SELECTED APPROACHES AND FRAMEWORKS TO CARRY OUT GENOMIC DATA PROVISION AND ANALYSIS ON THE CLOUD

PHILIP C. CHURCH AND ANDRZEJ M. GOSCINSKI*

Abstract. While High Performance Computing clouds allow researchers to process large amounts of genomic data, complex resource and software configuration tasks must be carried out beforehand. The current trend exposes applications and data as services, simplifying access to clouds. This paper examines commonly used cloud-based genomic analysis services, introduces the approach of exposing data as services and proposes two new solutions (HPCaaS and Uncinus) which aim to automate service development, deployment process and data provision. By comparing and contrasting these solutions, we identify key mechanisms of service creation, execution and data access required to support non-computing specialists employing clouds.

Key words: Bio-informatics, Software as a Service, HPC

AMS subject classifications. 68M14, 92C37

1. Introduction. High Performance Computing (HPC) has enabled new discoveries in disciplines such as biology and medicine by providing computer facilities to perform complex algorithms and process big data within reasonable time frames. However, HPC requires powerful and expensive computational hardware, data storage, advanced middleware, and sophisticated discipline oriented applications. Due to their high initial purchase and maintenance costs, HPC resources are only affordable by rich institutions. Furthermore, these resources are shared by many researchers, which leads to long waiting times for application execution. Thus, many researchers cannot access HPC infrastructures when needed; they often scale down their applications to reduce waiting times.

In response to these problems, public cloud vendors such as Amazons Elastic Compute Cloud (EC2) [1] have started to provide solutions specifically designed for running HPC applications. These clouds provide the ability to scale on demand as the users requirements change, accelerating the discovery of new knowledge in various fields of research. Thus, discipline specialists now have access to on-demand, scalable and pay-as-you-go HPC facilities that can provide users faster turnaround times on their experiments.

However, while these clouds alleviate the costs of procuring required IT resources, the cost and time of learning how to prepare a HPC cloud and its applications remain a problem to many users [2]. Thus, if discipline scientists want to use HPC clouds for scientific discovery, they must also become system administrators and computer specialists performing time consuming resource management and software configuration activities [3]. Once the HPC cloud has been set up, additional time must be spent transferring data from local machines to the cloud before analysis can be carried out. This process differs depending on the area of discovery; when working with genomic data, users take advantage of both publicly available and privately collected data. As a result, a significant amount of the scientists time is spent to build an understanding of the HPC cloud, acquiring system management skills and managing data instead of conducting research on these HPC clouds.

Discipline specialists are used to accessing software tools through user friendly discipline oriented interfaces. Therefore, we propose to use the lessons learnt from using software tools and hide from a discipline specialist all the operations that require administrator and computing expert knowledge and skill to exploit clouds. This paper examines currently used solutions in the area of bioinformatics to carry out research in the cloud, as well as our work on building frameworks to simplify development and deployment of sequential and HPC application exposed as services to be executed in a Software as a Service (SaaS) cloud.

In terms of genomic data, the current approach taken in the bioinformatics area is that data is shared through public data servers (also known as biological databases). Three geographically separated databases (American, European and Asian) store genomic data which is mirrored on a daily basis; this is supplemented by smaller public databases which store specialized data (for example cancer data [4]). Research groups will often have a local server in which they maintain copies of the data they are interested in (public and privately

*School of Information Technology, Faculty of Science and Technology, Deakin University, Geelong VIC 3127, Australia (philip.church@research.deakin.edu.au, ang@deakin.edu.au)

collected) as well as analysis tools [5]. Since data is stored not only on private data servers but mainly on public data servers, there is a natural opportunity to expose them as Data as a Service (DaaS).

By comparing and contrasting these solutions, we identify the cloud access procedures that are carried out by discipline specialists, and cloud service components that are commonly provided by developers but hidden from these specialists. We demonstrate that building frameworks that automate these cloud service components opens the way to build SaaS and DaaS clouds that support genomic data analysis and make them easy to use by discipline, non-computing specialists.

In brief, the contributions of this paper are as follows;

- A review of current cloud service approches to analyse genomic data.
- An extention of the Uncinus framework to publish and access genomic databases.
- The identification of key components required by cloud services to support non-computing specialists.

The rest of this paper is as follows. Section 2 describes current approaches to carry out genomic research on the cloud. Sections 3 presents developed by us two cloud service solutions (HPCynergy and Uncinus). Section 4 presents the path towards DaaS in biology and describes an extension to the Uncinus framework to support current genomic data resources. Section 5 compares and contrasts cloud solutions using a range of criteria. Section 6 describes the key components required by cloud services to support non-computing specialists. Finally, section 7 presents the conclusion and future work.

2. Current Approches. The use of HPC clouds to support genomic analysis is of great interest to the bioinformatics community. HPC clouds promise the ability to access HPC resources cheaply and on-demand. While any software can run on IaaS clouds, software exposed as a service can be accessed by a majority of researchers working in the area of bioinformatics and therefore it is inherently more useful.

Popular tools used to carry out genomic research on the cloud include: Cloud BioLinux, Galaxy and the Tuxedo Suite.

2.1. Cloud BioLinux. One of the early attempts to simplify the deployment and execution of bioinformatics software on the cloud was Cloud BioLinux [6]. Cloud BioLinux is a virtual machine (VM) configured for high-performance bioinformatics using cloud platforms. At time of writing [7], over 135 bioinformatics tools have been deployed and configured on the virtual machine. As applications have been taken from a large number of sources, there is no standard interface mechanisms, instead users access and execute installed applications through unique graphical interfaces or command line. However documentation of each installed application is made available through a centralized website.

Cloud BioLinux is compatible with IaaS clouds that implement the Amazon model, such as Amazon EC2 [1], OpenStack [8] and Eucalyptus [9]. Alternatively the Cloud BioLinux VM can be executed on a desktop computer using virtualization software such as Virtualbox [10]. The operation to deploy Cloud BioLinux on Amazon EC2 is shown in Fig. 2.1.

1. First, the user selects and configures the Cloud BioLinux VM. This process requires that the user create or select a key pair (a security credential), which will be used by the user to securely connect to the VM instance after it is running. The user must also create or select a security group, which defines firewall rules for the VM instance. Lastly, they specify the number of machines in which to deploy Cloud BioLinux. Running VMs are shown in the Amazon EC2 management console.

2. Once the VM has been launched, the user can access Cloud BioLinux. Users can access the Cloud BioLinux command line through a SSH client (secure shell) and the security keys generated in step 1. Alternately they can use a virtual desktop client to access Cloud BioLinux graphically. Once connected, users transfer files to Cloud BioLinux from a local or remote server before running applications.

3. Once a user has finished their analysis, they can terminate their Cloud BioLinux virtual machine. For this purpose, in the Management Console, they locate the instance(s) in the list of instances on the Instances page, and confirm to terminate the instance(s).

Currently Cloud BioLinux does not provide HPC support; instead each user runs distinct VMs that encapsulate an operating system, analysis software and data. With this approach, users receive distinct and pre-defined computational resources, since each user initializes their own Cloud BioLinux server. While Hadoop/MapReduce [11] [12] is available on Cloud BioLinux, computation is limited to the single virtual machine. Future work will allow end-users to easily provision Hadoop clusters on any cloud. This will facilitate

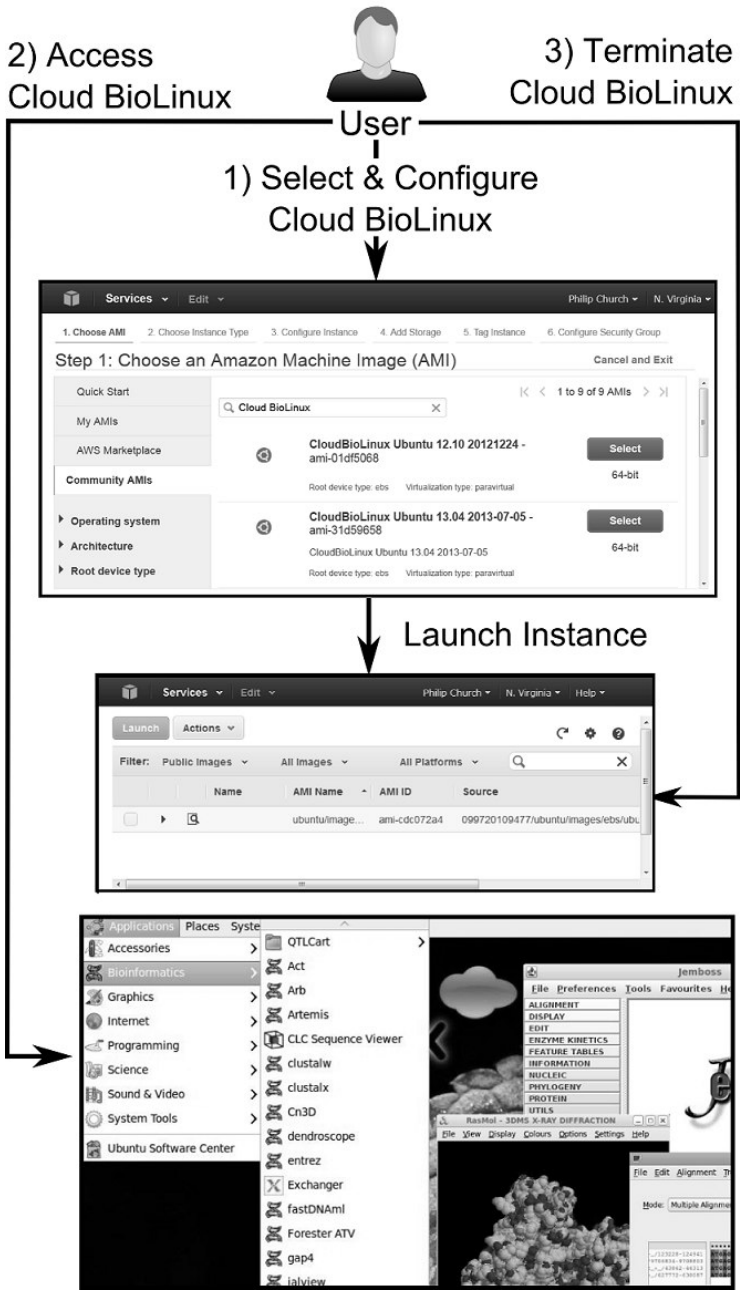

FIG. 2.1. *Cloud BioLinux Deployment Process*

running large-scale bioinformatics data processing pipelines.

2.2. Galaxy. Galaxy [13] is an web-based framework for genomic research. Due to the flexibility and easy-to-use nature of Galaxy, it has become a popular method for carrying out genomic analysis. Galaxy is deployed on a server and accessed through a web browser. Using provided web interfaces, users can share data and applications (as tools) and execute tools as workflows (see Fig. 2.2). Users using Galaxy first upload data from local or public data servers (biological databases) using a range of Galaxy tools. Uploaded data is transfered via FTP and stored on the Galaxy server. Tools which provide analysis capabilities are then used to

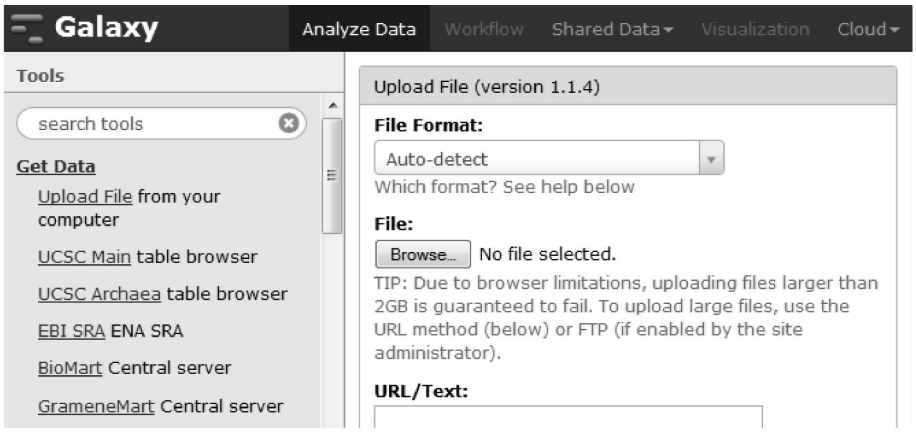

FIG. 2.2. *Galaxy Framework Interface*

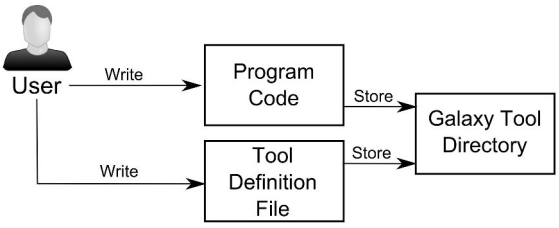

FIG. 2.3. *Galaxy Tool Deployment Process*

process data uploaded to the Galaxy server. Galaxy ensures that this analysis is reproducible by automatically generating metadata for each analysis step. Galaxys metadata includes: input datasets, tools used, parameter values, and output datasets. Galaxy also provides a workflow system which facilitates analysis repeatability, allowing users to carry out analysis based on stored metadata.

Adding a tool to Galaxy requires the user to provide an application to execute and a graphical interface (see Fig. 2.3). Applications can be written in a number of different programming languages as long as the Galaxy server has a valid interpreter installed; common languages include Python and Perl. Regardless of the language utilized, code is placed in the tools directory of the Galaxy installation. Each tool in this directory must be exposed through a graphical interface. To simplify this process, Galaxy users define a GUI for their tool using a Tool Definition File (TDF). A TDF is written in XML and consists of four parts; input controls, output data, test inputs (for validation purposes) and user help.

A key feature of Galaxy is workflow generation, where users can link together installed tools into a single pipeline. Through a pipeline Galaxy can run multiple tools concurrently, reducing the time taken for analysis. To simplify the construction of these pipelines, Galaxy provides an editor (see Fig. 2.4) from which users access a graphical interface for creating and modifying workflows. A user drags and drops tools onto the workflow canvas and configures each step in the workflow. Users can add tags to a workflow, annotating each step of the workflow. Workflows are run in Galaxys analysis workspace; like all tools executed in Galaxy, Galaxy automatically generates history items and provenance information for each tool executed via a workflow.

Galaxy supports the use of the Amazon EC2 cloud through the CloudMan service [14] (see Fig. 2.5). CloudMan automates the process of constructing a cluster using Amazon EC2 resources. Through a web interface, users must first specify their AWS Secret Key ID, and Secret Key. Once these details have been validated, a user can select the size of the cluster required and the location of a persistent data volume. CloudMan utilizes a modified Cloud BioLinux image which contains Sun Grid Engine (SGE) [15]. This image is automatically deployed on the Amazon EC2 cloud as both the head node and workers. Data stores are then linked to the

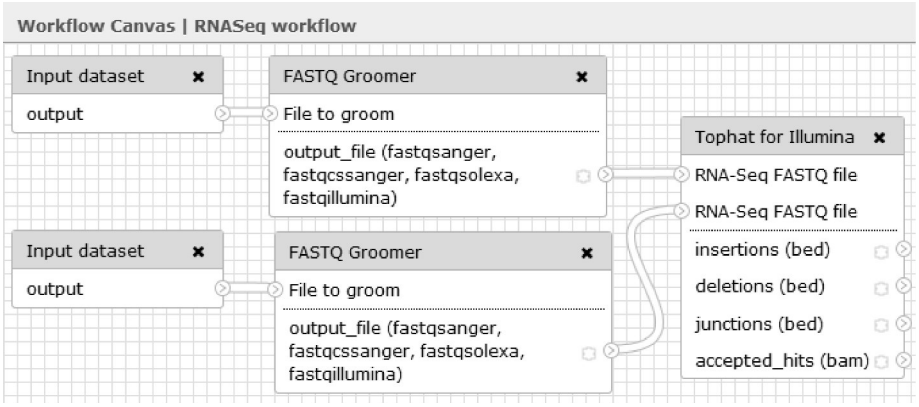

FIG. 2.4. *Galaxy Workflow Deployment Interface*

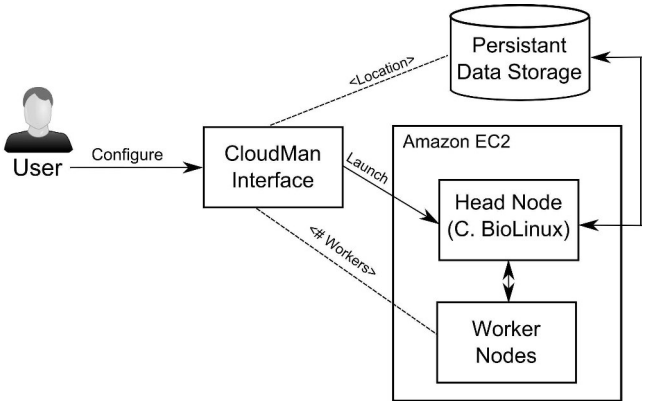

FIG. 2.5. *Galaxy Cloud Deployment Process*

constructed cluster. Once deployed, users can access Cloud BioLinux tools through command line and Galaxy through framework interfaces (via a web browser). By providing user with standard Galaxy interfaces, the cloud is made transparent to the user, in other words Galaxy running on the cloud is indistinguishable from Galaxy running on a local server.

2.3. Tuxedo Suite. The Tuxedo suite brings together commonly used bioinformatics tools for analyzing genomic data generated from high throughput sequencing machines. Some of these tools provide support for HPC platforms in order to reduce the time taken to process genomic data. Two HPC enabled tools in this suite, Crossbow and Myrna, have been made available as cloud services.

Crossbow [16] is a scalable application for genome sequencing aimed at performing alignment between the small fragments produced by the current generation of high-throughput sequencing machines. Crossbow has been integrated into Amazon EC2 using a map and reduce strategy [12]. A graphical interface (see Fig. 2.6) allows users to deploy Crossbow on the Amazon EC2 cloud. This interface requires user provide their AWS Secret Key ID and Secret Key to access their Amazon Account, input any optional arguments accepted by Crossbow and select the number and type of cloud instances they wish to use.

The operation of Crossbow is shown in Fig. 2.7, where a virtual cluster is first created on the Amazon EC2 cloud. Sequence fragments are then uploaded from the users desktop to Amazon storage (S3). Next crossbow code is uploaded to the master/head node, compiled and run. Output data is compressed and sent back to the user.

Myrna [17] is cloud-scale software developed for the purpose of analysing RNA-seq data. Using this software, it is possible to identify the number and type of genes present in a biological sample. To accomplish this, Myrna

FIG. 2.6. *Crossbow Service Interface*

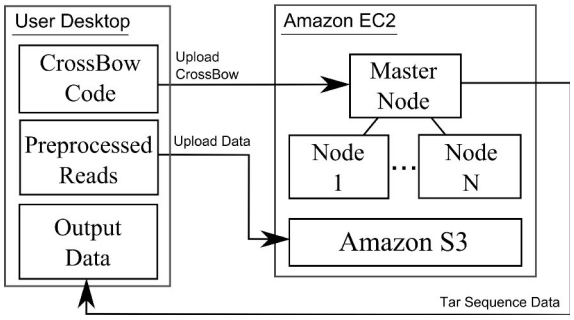

FIG. 2.7. *Crossbow Cloud Workflow*

aligns sequence data, normalizes and carries out statistical modeling in a single computational pipeline. Like Crossbow, Myrna also runs on Amazon EC2 using a map reduce strategy. During execution, each step in the Myrna pipeline is mapped and reduced. A map stage takes a stream of input data, analyses and returns results in the form of a stream. A sort/huffle phase is carried out that sorts data according to data similarity. Lastly, the reduce stage performs computation on data. Myrna is accessed as a service through a web interface in the same manner as Crossbow. Through this interface, users provide their Amazon ID and Secret Key and configure software settings. Upon execution of Myrna as a service, a virtual cluster on Amazon EC2 is created before deploying and executing Myrna.

3. Proposed Approches For Bioinformatic Services. Popular approaches to carry out bioinformatics on the clouds utilize a combination of packaged virtual machines and graphical interfaces to expose applications

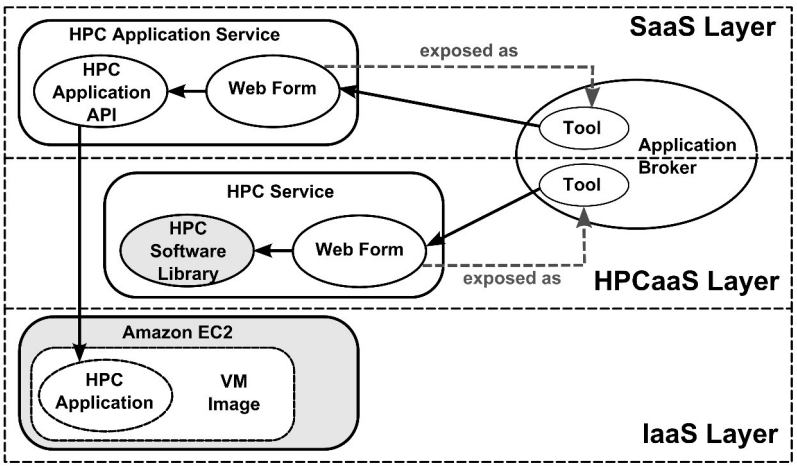

FIG. 3.1. *Implementation overview of HPC cloud framework*

as services. In this way, tools such as Galaxy and Tuxedo Suite (and to a lesser extent Cloud BioLinux) allows users to take advantage of scalability and clouds without understanding cloud architecture. While these solutions are simpler to use, they are difficult to develop and are not flexible (users limited to the services made available). In response we present two approaches for exposing bioinformatics applications as services, HPCaaS and Uncinus. HPCaaS brings targeted cloud computing to Galaxy, while Uncinus supports the development and deployment of services through a cloud middleware.

Our frameworks offer these features through HPC-based publicly accessible virtual machine (IaaS) abstraction, automating the process of exposing HPC biology and discipline applications as services, making these services visible through a broker, and evocable through user friendly discipline oriented interfaces.

3.1. HPCaaS: Galaxy Plugin. We have implemented a Galaxy plugin that allows users to deploy software packages on the EC2 cloud. This plugin was developed by integrating three components: (i) the EC2 service (public IaaS cloud) for providing HPC infrastructure, (ii) a HPC service software library for accessing high level HPC resources obtained from a IaaS cloud, and (iii) the Galaxy software application used as a web-based platform for exposing and accessing HPC application services.

The overview of the prototype design, demonstrating the relationships among the Amazon EC2 service, the HPC software library, and the Galaxy software application is shown in Fig. 3.1. Also shown in Fig. 3.1 is our view of the cloud service stack and where different cloud services would be found. At the bottom (IaaS layer), Amazon EC2 provides cloud infrastructure services. Supported HPC applications are installed in virtual machines and their images were saved and stored in EC2.

In the middle (HPCaaS layer), a HPC software library [18] was used to expose and access Amazon EC2 services in order to provide users a higher level of HPC services such as constructing and managing computer clusters. By using a specific feature of the Galaxy software application, a web-form of the HPC service was generated to be used as a simple but effective interface for users to access the HPC service. Finally, such HPC service was exposed as a tool in a Galaxy server.

On the top (SaaS layer), a HPC application service for a supported HPC application was developed as follows. First, an API of the HPC application was constructed. This API acts as a program stub for its corresponding HPC application, which had been installed in a virtual machine and had been stored in the Amazon EC2 service. Second, a web-form of the HPC application service was generated using the Galaxy application software. Finally, such HPC application service was exposed as a tool in a Galaxy server. It should be noted that each HPC application service would access the HPC services in the HPCaaS layer and the HPC application installed and stored in a VM image at the bottom IaaS layer through its web-form.

Using the HPCaaS Galaxy plugin, users can expose existing virtual machines stored on the Amazon cloud. For each VM, HPCaaS's Tool Definition File (TDF) needs to be modified. In this process, the location of the

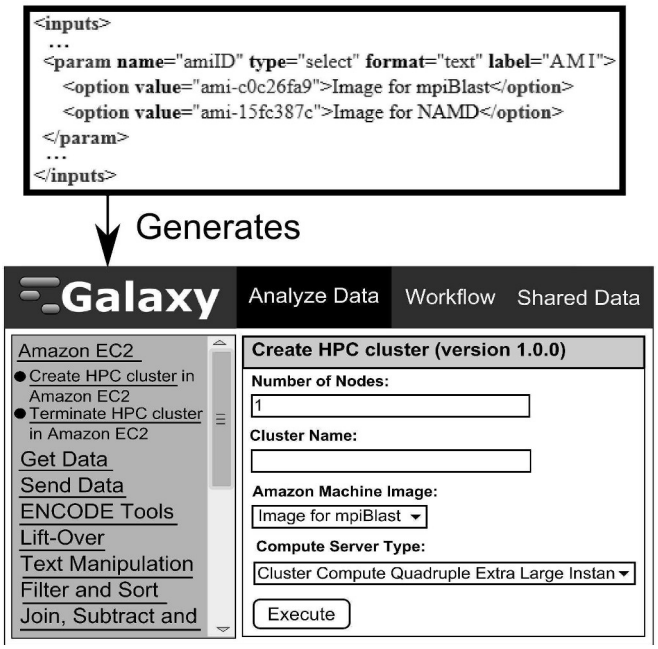

FIG. 3.2. *HPCaaS Galaxy Deployment Interface*

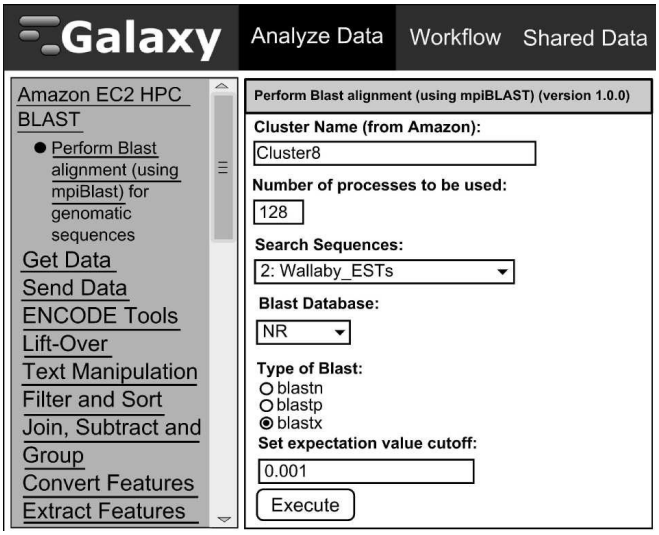

FIG. 3.3. *HPCaaS mpiBLAST service Interface*

Amazon Machine Image (defined as an ID given by the cloud provider) is added to the dropdown menu of the graphical interface (see Fig. 3.2). Lastly, an interface for the VM is generated; this is performed by defining a TDF in the same manner as other Galaxy tools.

The steps taken to utilize this interface is similar to that of the Tuxedo suite (see section 2.3), in that users can create a virtual cluster in the cloud by specifying the required number and type of cloud resources. HPCaaS will automatically construct a virtual cluster with the specified resources, each node consisting of a copy of the virtual machine image. However, HPCaaS is more flexible then the Tuxedo Suite as it allows multiple VM to be exposed through the same interface. In this way, the time taken to build HPC cloud services is reduced.

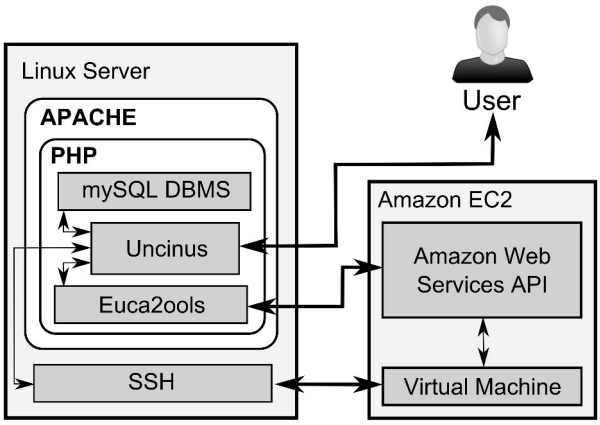

FIG. 3.4. *Uncinus Server Overview*

Once the virtual machine has been deployed on cloud resources, it is made available as a Galaxy tool. An interface for mpiBLAST [19], a commonly used distributed sequence alignment tool, is shown in Fig. 3.3. It consists of six input controls;

- Cluster Name: the name given to the cloud cluster deployed using HPCaaS.
- Number of processes: The number of instances of mpiBLAST that should be run. This also effects the number of fragments that the mpiBLAST database is divided into.
- Search Sequences: the file containing the sequence fragments in which to align.
- The database type: the database in which to carry out BLAST.
- The type of blast: the version of BLAST to run. This option depends on the type of data being analyzed
- The expectation cutoff: a text control which allows a user to choose a value in which to filter results.

The mpiBLAST interface is defined such that it mirrors similar tools such as the Tuxedo Suite and those provided through Galaxy. In this way, a user can access services in a uniform manner regardless of where the software package is being run.

3.2. Uncinus. Uncinus is an environment which takes advantage of existing computer and software resources. Users are provided with a web based interface in which they can share and access a range of cloud, resource and software services. Through use of a broker, Uncinus supports a modular approach to deploying applications on HPC clusters and IaaS clouds, users constucting services (consisting of software and hardware) on demand [20]. These on-demand services are made possible through automated application deployment and job submission provided by the Uncinus middleware.

Deployment of Uncinus on a Linux server is shown in Fig. 3.4. Both end-users and developers access Uncinus through a series of web pages (HTML/PHP) hosted by an APACHE server. Through these web pages, users can access services to publish applications and request compute resources. To provide these services, Uncinus interacts with a mySQL database (to store user credentials and service deployment information) and Euca2ools [9] (to interact with Amazon like clouds). Through euca2ools, users can access the Amazon Web Service API and thereby request cloud resources. Once cloud resources are deployed SSH is used to interact directly with virtual machines.

Users accessing Uncinus for the first time must create a user account. Through this account creation process, a user provides a unique username, password and cloud provider account details. For security reasons, cloud provider account details must be provided in the form of access and secret keys. Once logged in, users can access the components that make up Uncinus (mySQL DBMS, cloud middleware and Euca2ools) through graphical interfaces, shown in Fig. 3.5. The mySQL database is used to store end-user credentials and information about deployed applications/resources. Cloud Deployment Middleware sits above the IaaS cloud and provides a number of services including secure data transfer, cloud resource allocation, HPC environment setup, job submission systems and automated application deployment. Communication between the middleware services

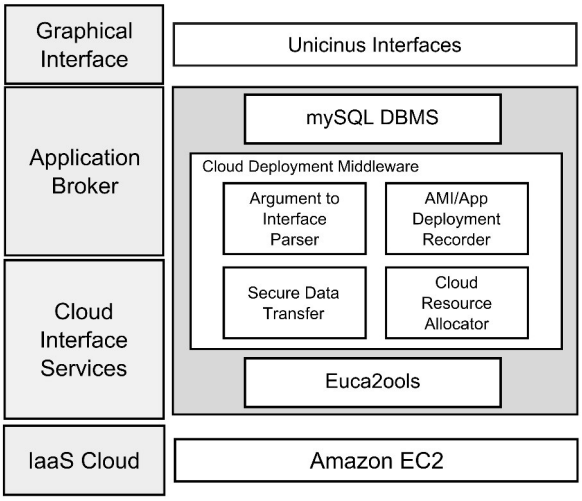

FIG. 3.5. *Uncinus Software Overview*

and the cloud is performed through Euca2ools.

Users access Uncinus through a series of web interfaces that expose the Cloud Deployment Middleware. Unlike other commonly used cloud solutions (described in section 2) which require construction of virtual machines, Uncinus creates services at the application level allowing users with limited cloud experience to develop software services. During the process of exposing an application, users specify a variety of attributes [21] including the service name, software files, installation scripts, software I/O and hardware requirements.

Fig. 3.6 presents the attributes required to deploy mpiBLAST as a service. Publication of this service begins by assigning the service a name (Application Name). Next, a compressed file containing the mpiBLAST source code was uploaded to the broker (Files). An installation script (Install Script) was provided to specify that the openMPI service be loaded as a prerequisite before compiling mpiBLAST. The services interface (Arguments) is specified by defining the type of data required. For this service, a control to upload sequence data and three text controls that determine the type of database and BLAST to be run were specified. The commands to execute mpiBLAST were provided (Running script) in the form of a shell script that formats the selected database and executes mpiBLAST over the selected number of processes. The output of the mpiBLAST service (Result) is a file called blast_results.txt which is to be downloadable by the user. Lastly, requirements for running mpiBLAST are provided; Linux (Operating System), a cluster with 8 nodes each with 8 core running at 2.66 GHz (CPU) and 2 GB of RAM per node (RAM).

The published mpiBLAST service was deployed using the interface shown in Fig. 3.7. Users begin by selecting from available computational resources (cloud and non-cloud) and from a list of published services. When this job is submitted, Uncinus carries out automated resource selection (using published attributes such as CPU and RAM) to identify how to deploy services on the selected resources. This resource selection process calculates a distance metric based on the service requirements and resource specifications.

Once the mpiBLAST service was deployed, a web interface is automatically generated using the published attributes. The mpiBLAST service interface presented to the user is shown in Fig. 3.8. Through this interface, users can: specify mpiBLAST argument, upload sequence data, execute mpiBLAST, download results, and terminate the job (freeing all cloud resources). Also provided through this interface is a tool for transferring files to and from the cloud; uploaded files are mirrored across each node in the cluster.

Through the current implementation of Uncinus, users can take advantage of published software and cloud resources, accessing resources without the need to carry out complex configuration tasks such as setting up a virtual cluster. However in the example above, users are still required to upload sequence data from their local machine. By exposing data as a service, Uncinus can be extended to allow software services to take advantage

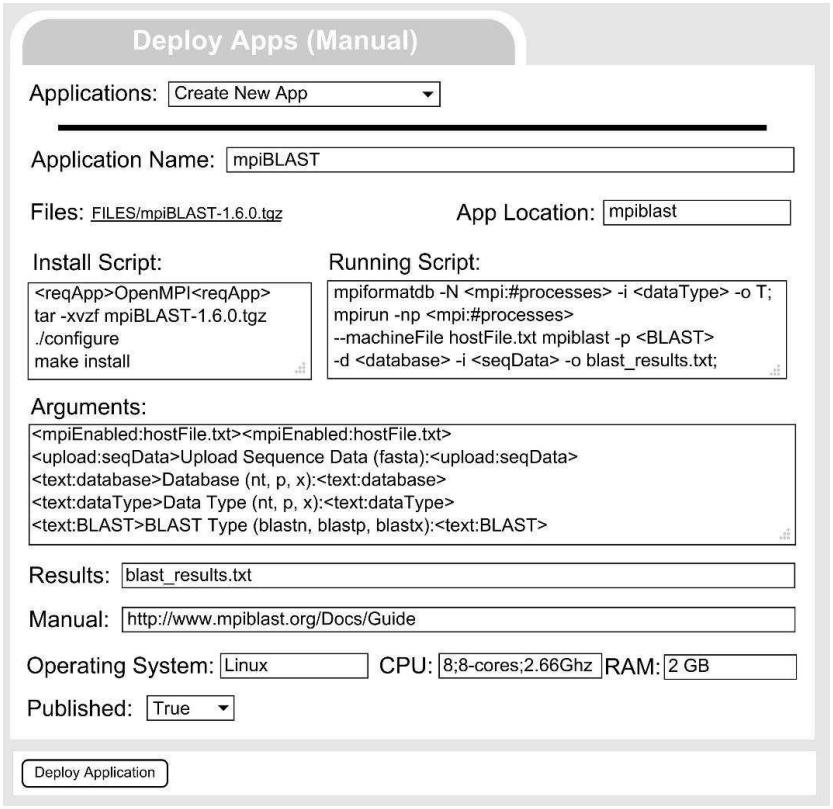

FIG. 3.6. *Uncinus mpiBLAST Publication*

of publically available data.

4. Moving Toward Data as a Service. Data as a Service is an area not highly explored in the area of genomics. The only commercial example is Amazon which provides two DaaS services: S3 which provides users with extendable storage and the Elastic Block Storage (EBS) which provides users the ability to create virtual hard-drives. Through these two services, Amazon provides access to genomic data from the 1000 genome project. An academic attempt at DaaS is being led by the University of Chicago [26], who have recently announced the creation of a petabtye scale open science data cloud which will allow researchers access to manage, analyze and share their data.

4.1. Toward DaaS in Biology. Currently the approach taken in the bioinformatics area is that data is shared through public data servers (also known as biological databases). Three primary databases (NCBI [22], EBI [24] and NGI [23]) store DNA sequence data and are mirrored on a daily basis, this is supplemented by many smaller databases which target a subset of data (for example specific diseases [4] and collection platforms [25]). Research groups working the area will often have a local server in which they maintain copies of the data they are interested in (public and privately collected) as well as analysis tools (as seen in solutions such as Galaxy) [5]. Data stored on these local servers exist in the form of human readable ASCII files which are often not indexed or easily searchable.

While these methodologies are satisfactory for small amounts of data, when analysing the amount of genomic data generated by current sequencing machines, there arises a number of issues.

- Database Discovery: Smaller specialized databases can be difficult to find, currently discovery of new databases is reliant on paper publication and word-of-mouth.
- Lack of Data Management: Research groups maintain local data servers which store data. Data is often stored in a human readable ASCII format which takes up more disk space compared to binary formats.

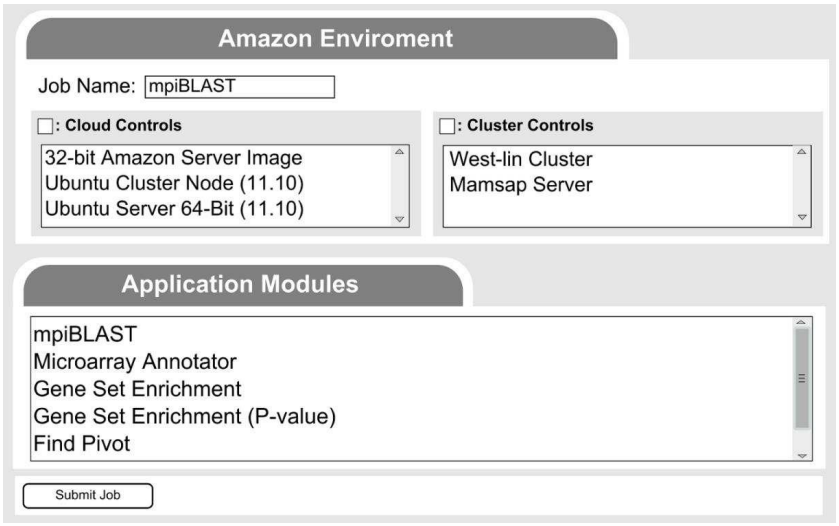

FIG. 3.7. *Uncinus Service Development Interface*

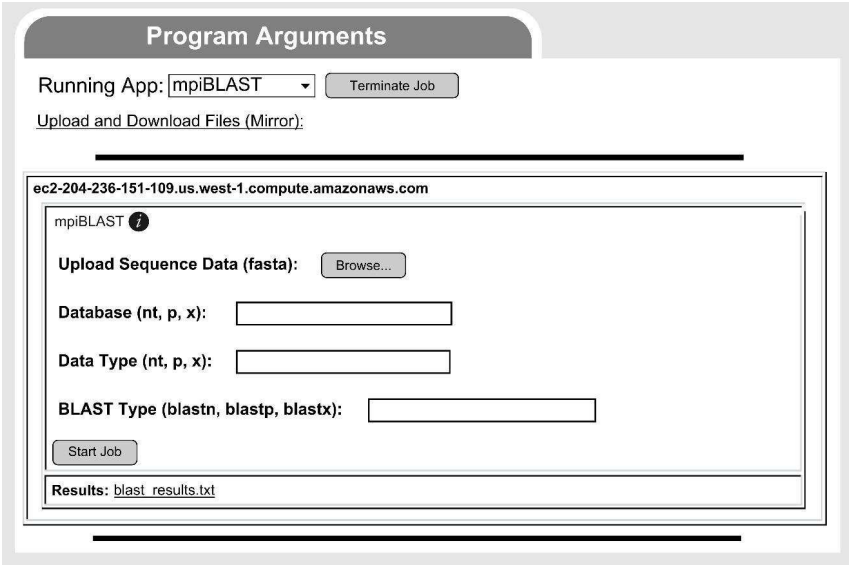

FIG. 3.8. *Uncinus mpiBLAST Service Interface*

Data is difficult to find as there are limited search/discovery mechanisms implemented on these local data servers.

- Database Reliability: Local data servers often have minimal/no backup or redundancy (due to the cost of storage). These systems are inflexible, which jeopardizes availability and reliability.
- Data Transfer: In context of utilizing cloud resources, most cloud tools require that data is moved from public databases to a user's local machine and then back to the cloud. This method is wasteful as data is transferred to a local machine where it is not needed.

Since data is stored not only on private data servers but mainly on public data servers, there is a natural opportunity to expose them as Data as a Service. The data could be stored on any of the server without the user involvement; the data attributes could be stored on a data broker, which becomes a major tool used by other user who wish to search for specific data. The data copies, for reliability and availability, could be stored

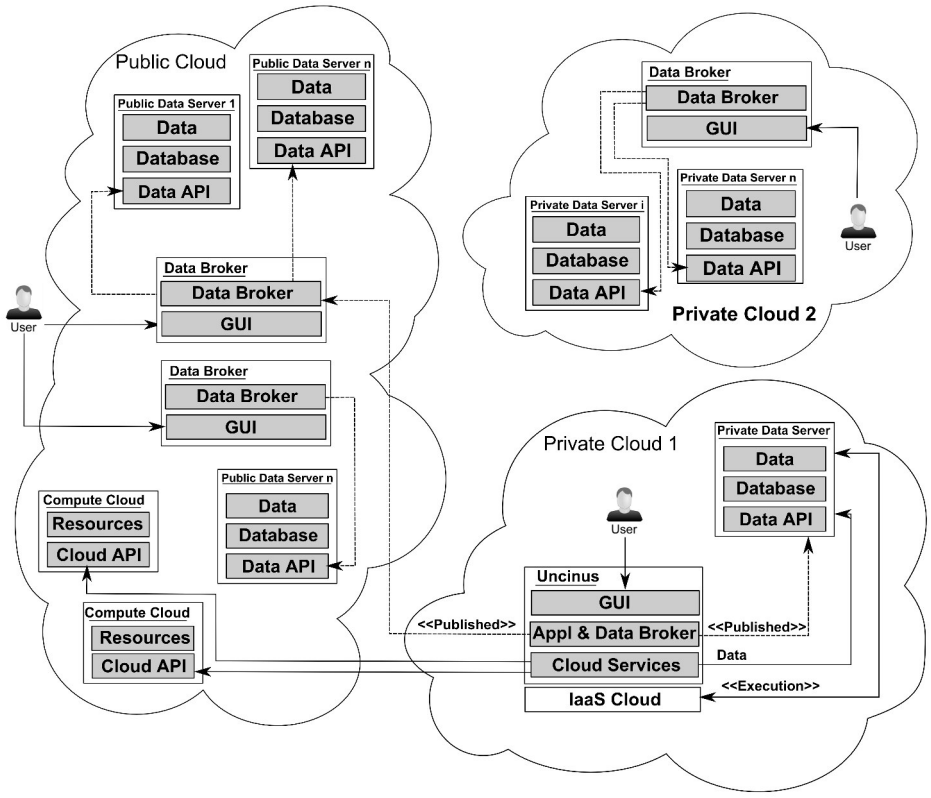

FIG. 4.1. *DaaS Overview*

on a number of servers; their location could be hidden from the user. The selection of a copy of the requested data could be carried out by the broker taking into consideration user location and money limits. A user oriented interface could offer an easy access to the data broker that supports data selection for the application of interest. Data are managed by cloud service providers; that relieves discipline specialists from many activities of computing nature. As shown in Fig. 4.1, the user can carry out discovery on their local resources, within the research group/laboratory private cloud, and/or a public or hybrid cloud. The last option lowers the costs of resources, supports computations on demand, and allows cooperation to be exploited widely.

4.2. DaaS in Uncinus. In this proposed solution we extend the Uncinus application broker to publish data severs. Users specify the location of the server as well as methods to download files (GET), upload files (PUT) and get directory listings (LIST). In this way Uncinus can support a range of public biological databases by abstracting their unique API. Public data brokers are also deployed which users can access to discover databases and share their own data. The data broker will also be able to upload data to published biological databases. By linking the Uncinus broker to a public broker, Uncinus users will be able to access private data servers and public biological databases. Through the proposed solution we are convinced that in many cases the performance of discovery in biology and medicine could be dramatically improved if data sharing (as described above) is applied widely.

5. Comparison of Solutions. Our paper investigates a number of popular bioinformatics SaaS cloud solutions. These solutions differ in how they are developed, deployed and used. We compare and contrast these solutions through the use of six criteria; 1) usability/access, 2) development, 3) deployment, 4) service discovery 5) cloud utilization and 6) data intergration. Each platform and their adherence to the five criteria are shown in Table 5.1.

The first criterion focuses on how the presented solutions are accessed and used. In general, all of the

TABLE 5.1
Popular solutions compared to defined criteria.

	Access	Develop.	Deploy.	Discovery	Utilization	Data
BioLinux						Amazon
Tuxedo						Amazon
Galaxy						Tools
HPCaaS						Tools
Uncinus						Data Broker

bioinformatics cloud software described in this paper allows users to take advantage of scalability and clouds without understanding cloud architecture. Through graphical interfaces, users can access HPC clouds without needing to carry out the time consuming task of setting up middleware to create a virtual cluster in the cloud. Cloud BioLinux exposes applications through an OS interface, while Tuxedo Suite, Galaxy, HPCaaS and Uncinus provide web based controls.

The second criterion focuses on comparing the effort taken to develop services. In general, development of bioinformatics cloud services is difficult and time consuming, requiring skills in genomics, HPC computing, programming and cloud computing. This is seen in Cloud BioLinux and HPCaaS, which depend on the creation and packing of a VM through the use of command line tools. In the case of HPCaaS, users must also modify the HPCaaS tool to recognize the VM. The Tuxedo Suite required the development of graphical interfaces and software API to communicate with the cloud application. Recognizing the difficulty of cloud service development, we have seen a gradual shift from individual services to frameworks such as Galaxy and Uncinus, which attempt to simplify service development. Galaxy makes the development process easier by simplifying construction of graphical interfaces. Uncinus takes this one step further and allows users to define attributes, which are used to generate graphical interfaces and better utilize resources.

The third criterion examines how solutions are deployed on the cloud; currently this ranges from difficult to deploy to no deployment required. Traditionally, HPC cloud solutions have been offered at an IaaS level, which is targeted at computing experts. Early bioinformatics solutions which made use of these IaaS cloud platforms (such as Cloud BioLinux) are difficult to deploy. Other solutions attempt to simplify deployment through automation of this deployment process. Galaxy, HPCaaS and Uncinus as frameworks, provide functions to deploy services on clouds. The Tuxedo Suite automates all aspects of deployment; users require only their Amazon ID and Secret Key.

The fourth criterion examines how these services are discovered by users. Individual services such as Cloud BioLinux and the Tuxedo Suite do not have any inbuilt form of discovery. The Tuxedo Suite consists of a small number of tools and therefore has no need for discovery mechanisms; as these suites get larger, the lack of service discovery can limit their use. Cloud BioLinux contains hundreds of different applications; however service discovery is not supported. Each application provided by Cloud BioLinux has different access methods. Frameworks such as Galaxy, HPCaaS, and Uncinus, which also make available large amounts of tools, use service repositories. By centralizing and indexing available services, users can find software easier.

The fifth criterion examines how well these solutions take advantage of the cloud during execution. The Tuxedo Suite is built directly for the cloud; this solution takes advantage of cloud scalability to increase performance. CloudBioLinux and Galaxy (via CloudMan) are based around duplicating the environment, and unless parallelism is built into the hosted tools, can only ensure a consistent level of performance (each user having their own instance of Galaxy or Cloud BioLinux). Our approaches aimed to expose individual distributed applications like the Tuxedo Suite while taking advantage of the framework features to simplify service development. The first approach, HPCaaS, extended Galaxy based analysis to the cloud through a HPC software library that exposed Amazon EC2 services. Using HPCaaS, users could access VM stored on the EC2 cloud and execute distributed applications through Galaxy interfaces. Uncinus focused on service level development and deployment on clouds. Instead of developing a VM, users could publish installation steps which would be automatically carried out.

The sixth criterion examines how well these solutions are integrated with data. While all package can be

integrated with data, some are more complex to set-up then others. Packages that use Amazon EC2 can use genomic data (such as data from the 1000 genome project) stored using Amazon Cloud Storage (S3) and the Elastic Block Store (EBS). Some variants of CloudBioLinux will automatically mount the virtual EBS drives, and Tuxedo Suite (while not providing native access to this data) could be configured to use the genomic data stored on Amazon S3. Galaxy provides tools which can be used to discover and download genomic data which is stored on the Galaxy Server. HPCaaS (as a Galaxy tool) can also utilize the data downloaded in this way. The method proposed by Uncinus uses a data broker, allowing for discovery of data across multiple up-to-date genomic databases, as a result Uncinus is able to access a wider range of data when compared to the Amazon based solutions. Furthermore Uncinus stores only the link and description of the data, instead of the actual data as seen in the Galaxy approach, this minimizes the need for local storage and prevents unnecessary transfer between the genomic database, the Uncinus server and the computational resources.

In conclusion, the cloud solutions presented in this paper fall under two categories, stand-alone services and frameworks. Individual solutions include Cloud BioLinux and the Tuxedo Suite, while frameworks include Galaxy, HPCaaS, and Uncinus. The benefits of utilizing individual services are that they are often built for and can fully utilize the cloud, in addition they often have minimal to no deployment requirements. On the other hand, individual services are difficult to develop (requiring specialized GUI and API) and are difficult to discover. For a user looking to carry out research on the cloud, picking the right service from hundreds is difficult. Popular frameworks solve this problem with large repositories of services managed by brokers. This model can identify and therefore reduce the overlap in the types of cloud software services being made available. Additionally, frameworks simplify development by automating key steps in the service development process such as for example construction of graphical interfaces. While these features clearly benefit discipline researchers, popular frameworks (Galaxy) do not take full advantage of distributed and cloud computing platforms. Our solutions attempted to fill this gap to some success, proposing frameworks which provided access to clouds in a manner similar to individual services.

6. Discussion. While originally developed for business, cloud and service computing can be used for research. Cloud resources on-demand can enhance local resources, reducing the turn-around time of analysis and/or allowing for bigger problems to be solved. However clouds do not fit all applications; depending on the research being carried out, HPC hardware may be required. Fortunately, a range of cloud providers now offer HPC cloud solutions. Services built on top of these HPC cloud solutions could offer researchers access to HPC on demand without the need to understand and carry out complex deployment methods. It is by combining service and cloud computing technologies that solutions have been devised to simplify genomics analysis. These solutions fall into two categories; stand-alone services which are built from the ground up to utilize the cloud, and cloud frameworks which simplify the development of cloud services.

Despite the advantages of frameworks such as Galaxy, the bioinformatics cloud software service area is still dominated by individual, isolated applications (as seen in the Tuxedo Suite and Cloud BioLinux). Development of these cloud services are time consuming and often repetitive. A review of these approaches shows that the most utilized operations during IaaS application deployment fall into four categories: cloud security, resource allocation, application deployment, and data transfer (see Table 6.1). When deploying HPC applications, operations in these categories must be repeated for each computational node utilized. For this reason, steps such as installation and data transfer (particular in large data scenarios) can greatly increase the time spent to utilize cloud resources. In addition, a user wishing to provide services through the cloud (a service provider) is required to provide a further layer of abstraction on top of the IaaS cloud, exposing the deployed cloud application through a graphical interface.

TABLE 6.1
Most Utilized Operations during IaaS deployment.

Cloud Security	Creating security keys and groups
Application Deployment	Installing software applications
Data Transfer	Transfer files to and from the cloud
Resource Allocation	Launch and Terminate Instances

TABLE 6.2
Components required for SaaS development.

Software API	Allows execution of cloud applications
Graphical Interface	Allows communication with the API
Service Publication (Optional)	Stores and allows users to find service

Once software has been deployed on the cloud, it is possible to expose it as a service. Software services can be discovered and utilized by a larger number of discipline specialists but require development of a software API and graphical interface which are then (optionally) published (see Table 6.2). A software API defines the instructions used to run the application. A graphical interface allows users to execute and pass application arguments to the API. These services can be published; cloud application, API and interfaces stored and indexed through a broker. Service publication supports users, allowing then to discover services that fulfill their goals.

We propose that future cloud solutions should look to automate as much of these operations as possible in order to make service development as simple as possible. When building clouds aimed at carrying out research, cloud specific operations should be logically separated from data processing and analysis.

7. Conclusion. In conclusion, this paper presents a survey of approaches currently used in genomic analysis. Trends in the area show a move from IaaS clouds to SaaS frameworks. Early cloud solutions (Cloud BioLinux) packaged software tools into a single virtual machine. Due to the difficulty of maintaining this type of monolithic solution, developers moved to individual cloud services (Tuxedo Suite). More recently, frameworks such as Galaxy, which provide usability similar to cloud services while simplifying development, have become popular. However Galaxy's support for clouds is still primitive, reliant on mirroring the current state of the Galaxy server.

As a response we developed and presented two SaaS cloud frameworks that draw upon concepts from SaaS clouds, Galaxy and bioinformatics cloud software, in order to solve known cloud usability issues (the difficultly of cloud development and flexibility of individual tools). These frameworks addressed cloud usability by providing researchers the tools to access the cloud and run distributed applications. Users with a background in programming, system administration and cloud computing can develop HPC software and publish VM resources. Users with knowledge of the software applications (but limited programming skills) can define required attributes through the service publication interfaces and become SaaS providers. Lastly, users with a background in biology and minimal computing knowledge can access the applications and resources (published by the broker) that are required to perform analysis. For such users, clouds are made completely transparent and HPC applications are exposed as services. In this way discipline specialists with different levels of computing expertise can take advantage of cloud resources. By incorperating the ability to publish data resources, we further simplfy the genomic analsis process, users will be able to access up-to-date genomic data while removing the need for locally maintained genomic databases.

Through this successful integration of cloud and service computing, the analysis, interpretation and computation of genomic mammalian data was made easier, to be carried out by non-computing discipline specialists, and cheaper. By comparing current approaches and our cloud frameworks we identify the cloud access procedures that are commonly carried out by users, and cloud service components that are commonly provided by developers. Operations carried out by users during IaaS application deployment fall into four groups: cloud security, resource allocation, application deployment, and data transfer. They require a computing expert to be carried out; discipline specialists should be relieved from learning clouds and acquiring skills to carry out these operations. Instead they should concentrate on their discipline problems using clouds, in particular SaaS clouds. The major components provided by cloud services fall into three categories: Software API, Graphical Interface and Service Publication. The development of these components should be carried out to satisfy discipline specialists requirements to allow them to directly benefit from SaaS clouds.

Future work should aim to automate these procedures in such a way that clouds are made transparent to the user. Research into graphical methods to create virtual machines could widen the use of IaaS clouds, while natural language could be applied to service development to automatically generate graphical interfaces from software manuals and source code. A number of technical improvments could also be made: to take advantage

of exisiting software repositories such as Galaxy, there is a need to standardize interface mechanisms, there is also a need to enable seamless application scalability to improve cloud performance.

REFERENCES

[1] Amazon, *Amazon Elastic Compute Cloud: Getting Started Guide*, LLC AWS Amazon, 25 (2010).
[2] Goscinski A, Brock M, Church P, *High Performance Computing Clouds.*, CRC, Taylor & Francis group, June 2011.
[3] Yelick K, Coghlan S, Draney B, Canon RS, *The Magellan Report on Cloud Computing for Science*, U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), December 2011.
[4] D. Fenstermacher, C. Street, T. McSherry, et. al, *The Cancer Biomedical Informatics Grid* , in Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the, 2005, pp. 743-746.
[5] L. Dai, X. Gao, Y. Guo, J. Xiao, and Z. Zhang, *Bioinformatics clouds for big data manipulation.*, Biology Direct, vol. 7, p. 43, 2012.
[6] Cloud Research Group, *Cloud BioLinux: pre-configured and on-demand high performance computing for the genomics community*, 2010.
[7] Krampis K, Booth T, Chapman B, Tiwari B, Bicak M, Field D, Nelson K, *Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community.*, BMC Bioinformatics 2012, 13(1):42.
[8] OpenStack, *Open source software for building private and public clouds.*, http://www.openstack.org/
[9] Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D, *The Eucalyptus Open-Source Cloud-Computing System*, Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. IEEE Computer Society 2009.
[10] Oracle VirtualBox, http://www.virtualbox.org
[11] Dean J, Ghemawat S, *MapReduce: simplified data processing on large clusters*, Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6; San Francisco, CA. 1251264: USENIX Association 2004: 10-10.
[12] Apache Hadoop, http://hadoop.apache.org/
[13] Goecks J, Nekrutenko A, Taylor J, Team TG, *Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences*, Genome Biol 2010, 11(8):R86.
[14] Afgan E, Baker D, Coraor N, Chapman B, Nekrutenko A, Taylor J, *Galaxy CloudMan: delivering cloud compute clusters*, BMC Bioinformatics 2010, 11(Suppl 12):S4.
[15] Gentzsch W, *Sun Grid Engine: Towards Creating a Compute Power Grid*, Proceedings of the 1st International Symposium on Cluster Computing and the Grid. IEEE Computer Society 2001.
[16] Langmead B, Schatz M, Lin J, Pop M, Salzberg S, *Searching for SNPs with cloud computing*, Genome Biol 2009, 10(11):R134.
[17] Langmead B, Hansen K, Leek J, *Cloud-scale RNA-sequencing differential expression analysis with Myrna*, Genome Biol 2010, 11(8):R83.
[18] Wong AKL, Goscinski AM, *A unified framework for the deployment, exposure and access of HPC applications as services in clouds*, Future Generation Computer Systems 2013, 29(6):1333-1344.
[19] Darling A, Carey L, Feng W, *The Design, Implementation, and Evaluation of mpiBLAST.*, ClusterWorld 2003: 2002.
[20] Church P, Wong A, Brock M, Goscinski A, *Toward Exposing and Accessing HPC Applications in a SaaS Cloud.*, Web Services (ICWS), 2012 IEEE 19th International Conference on: 24-29 June 2012 2012. 692-699.
[21] Brock M, Goscinski A, *Attributed Publication and Selection for Web Service-Based Distributed Systems*, Services - I, 2009 World Conference on: 6-10 July 2009 2009. 732-739.
[22] National Center for Bio-informatics, http://www.ncbi.nlm.nih.gov/
[23] National Genomics Institute, http://www.nig.ac.jp/
[24] European Bioinformatics Institute, http://www.ebi.ac.uk/
[25] A. Brazma, H. Parkinson, U. Sarkans, et. al, *ArrayExpress–a public repository for microarray gene expression data at the EBI.*, Nucl. Acids Res., vol. 31, pp. 68-71, January 1, 2003 2003.
[26] Open Science Data Cloud - University of Chicago, https://www.ci.uchicago.edu/research-centers/open-science-data-cloud

Edited by: Jesus Carretero
Received: September 8, 2014
Accepted: January 21, 2015