Towards a Science Gateway for Bioinformatics: Experiences in the Brazilian System of High Performance Computing

Kary Ocaña, Marcelo Galheigo
National Laboratory of Scientific Computing
Petrópolis, Brazil
{karyann, galheigo}@lncc.br

Carla Osthoff, Luiz Gadelha
National Laboratory of Scientific Computing
Petrópolis, Brazil
{osthoff, lgadelha}@lncc.br

Antônio Tadeu A. Gomes
National Laboratory of Scientific Computing
Petrópolis, Brazil
atagomes@lncc.br

Daniel de Oliveira
Institute of Computing
Fluminense Federal University
Niterói, Brazil
danielcmo@ic.uff.br

Fabio Porto
National Laboratory of Scientific Computing
Petrópolis, Brazil
fporto@lncc.br

Ana Tereza Vasconcelos
National Laboratory of Scientific Computing
Petrópolis, Brazil
atrv@lncc.br

***Abstract*—Science gateways bring out the possibility of reproducible science as they are integrated into reusable techniques, data and workflow management systems, security mechanisms, and high performance computing (HPC). We introduce BioinfoPortal, a science gateway that integrates a suite of different bioinformatics applications using HPC and data management resources provided by the Brazilian National HPC System (SINAPAD). BioinfoPortal follows the Software as a Service (SaaS) model and the web server is freely available for academic use. The goal of this paper is to describe the science gateway and its usage, addressing challenges of designing a multiuser computational platform for parallel/distributed executions of large-scale bioinformatics applications using the Brazilian HPC resources. We also present a study of performance and scalability of some bioinformatics applications executed in the HPC environments and perform machine learning analyses for predicting features for the HPC allocation/usage that could better perform the bioinformatics applications *via* BioinfoPortal.**

Keywords—science gateway, bioinformatics, high performance computing

I. Introduction

Nowadays, genomics research shows an unprecedented effort in sequencing and categorizing genomes produced by new-generation high-throughput DNA sequencing [1]. The capacity for the biological data generation has led to an explosive growth of the complexity, heterogeneity, volume, and geographic dispersion of this biological "big data" [2]. Considering the annual growth of the generated data, it is estimated that the biological big data will reach 44 zettabytes in 2020 [3]. Thus, analyzing this volume of data is far from trivial. The integration of the latest breakthroughs in biomedical technology from one side and High Performance Computing (HPC), Scientific Workflow Management Systems (SWfMS) [4], and Database Management Systems (DBMS) [5] from another side, enables remarkable advances in the fields of healthcare, drug discovery, genome research, computational biology, system biology, data science (management, sharing, and execution) and so on.

Computational biology and bioinformatics are interdisciplinary fields that deal with the development of computational, mathematical, and biostatistics methods to analyze large biological datasets to infer hypotheses or discover new solutions. They have emerged as a very promising area in the analysis of genome sequences. Latin America has a very active research community interested in developing and using bioinformatics approaches for supporting academic, scientific, and industrial demands. The Brazilian Bioinformatics Network (RNBio) aims at strengthening the bioinformatics research projects in Brazil in a multi-institutional format with the training of specialized human resources in thematic studies involving bioinformatics and computational biology. RNBio has scientific collaborations with the Brazilian National System for High Performance Computing[1] (SINAPAD), which offers to users several heterogeneous and geographically distributed resources with high performance/throughput computing (HPC/HTC) capabilities and customized security models.

A science gateway [6] is defined as "a community-developed set of tools, applications, and data that is integrated *via* a portal or a suite of applications" [7]. SINAPAD applies grid computing [8] using the middleware CSGrid[2] that offers the single access points through web interfaces (gateways) for the submission of scientific applications for the use of HPC resources. CSGrid offers two key entry points for users, a desktop Java client and a service bus (called OpenBus). OpenBus allows managing services as jobs (OpenDreams) and files (HDataService), provided as APIs. Other CSGrid services that can be accessed by science gateways are the mc2toolset, a RESTful web service (APIs to develop gateways), and a CLI (Command Line Interface).

CSGrid has been used in different projects, as the mc2toolset that supports the prototyping of several scientific portals[3] at SINAPAD for physics, meteorology, chemistry, complex networks, mathematical, bioinformatics, medicine, *etc* [9]. However, although CSGrid and their tools represent a step forward, it may be complex to the regular users to execute

[1] https://www.lncc.br/sinapad
[2] https://jira.tecgraf.puc-rio.br/confluence/display/CN/CSGrid+Home
[3] https://www.lncc.br/sinapad/portais.php

their experiments, since it may require advanced knowledge of several tools, frameworks, applications, filesystems, etc. BioinfoPortal invokes the middleware CSGrid for managing the requests of users for executing applications in the HPC/HTC resources of SINAPAD. At the end of the execution, a link pointing to the scientific results is sent to the e-mail of the users.

The main goal of this paper is to address the problems and features of designing a multiuser computational platform for parallel/distributed executions of large-scale bioinformatics applications using the Brazilian HPC resources. A grid-based architecture for the science gateway is introduced. This architecture allows scientists to design and integrate components of a heterogeneous architecture for their experiments. Other goals of the Project called BioinfoPortal are: 1) to integrate HPC, SWfMS, and DBMS technologies; 2) to dispatch, manage, and execute processes in a transparent and friendly manner for (non)expert users; 3) to evaluate the performance of applications; and 4) to manage the provenance data tracking of records of executions at Bioinfo-Portal.

The results (scientific/performance) of HPC applications executions at BioinfoPortal were analyzed with the support of the provenance databases, by submitting high level database analytical queries. These results show that BioinfoPortal is functional and capable of processing up to the datasets required for each one of the bioinformatics applications, especially HPC applications as RAxML with multithreads and MPI, which improved the performance. We analyzed the general features of the application executions (input size, software parameters, efficiency of machines capacity) using machine learning techniques to explore the allocation/usage of computational resources for the gateway. Decision trees generated with regression models, based on a historic of the dataset, provided a promisor learning module and proved that choosing the platform configuration for performing executions is valuable for exploring the better usage of the HPC infrastructure in science gateways.

The science gateway BioinfoPortal mediates the execution of a suite of bioinformatics applications using the computational resources of SINAPAD. It follows the Software as a Service (SaaS) delivery model and is built on top of the middleware CSGrid, which allows managing the bioinformatics applications (programs, tools, or workflows) with HPC/HTC. BioinfoPortal is currently supported by the team of RNBio, National Laboratory of Scientific Computing (LNCC[4]) and SINAPAD and it is freely available for the academic use at https://bioinfo.lncc.br/.

This paper is organized as follows. Section II presents related works. Section III describes the specification of the architecture of the science gateway BioinfoPortal and presents its implementation using CSGrid and the HPC resources of SINAPAD. Section IV shows the experimental results and discussion and Section V concludes the paper and points out future work.

II. Related Work

Many science gateways offer sharing possibilities within a community using technologies based on the reusability of methods and reproducibility of diverse fields of science [7]. There are several initiatives of bioinformatics research groups for developing web interfaces, portals or migrating workflows or software for the community. We highlighted for discussion some main science gateways that covered HPC technologies.

MoSGrid [10] is a web-based science gateway for structural bioinformatics that provides an intuitive user interface to several applications. The security concept applies SAML (Security Assertion Markup Language) and allows for trust delegation from the user interface layer, middleware layer, and Grid middleware layer with HPC facilities.

Galaxy [11] is an intuitive science gateway that supports a large number of communities with overlapping research fields. However, Galaxy lacks the support of grid-based Distributed Computing Infrastructures (DCIs) and requires the installation of a Galaxy instance per underlying DCI. Thus, the migration of Galaxy workflows to WS-PGRADE workflows allows for flexibly using existing Galaxy workflows for various DCIs.

myExperiment [12] is an online research environment that supports the social sharing of bioinformatics workflows. As a public repository of workflows, myExperiment allows every user to discover those that are relevant to their research, which can then be reused and repurposed to their specific requirements. Although myExperiment represents a step forward, it does not allow users to execute their workflows.

The CIPRES[5] Science Gateway is a public resource for inference of large phylogenetic trees. It is designed to provide access to NSF XSEDE's large computational resources through a simple browser interface. It released a RESTful API to allow integration of CIPRES capabilities into other desktop software and web applications. The CIPRES Science Gateway is designed to manage data much like an e-mail client. The data is then used to stage individual jobs.

III. Proposed Approach: BioinfoPortal Gateway

The architecture of BioinfoPortal gateway is composed of four layers: User Interface Layer, Management Layer, Data Layer, and Resource Layer, as depicted in Fig. 1.

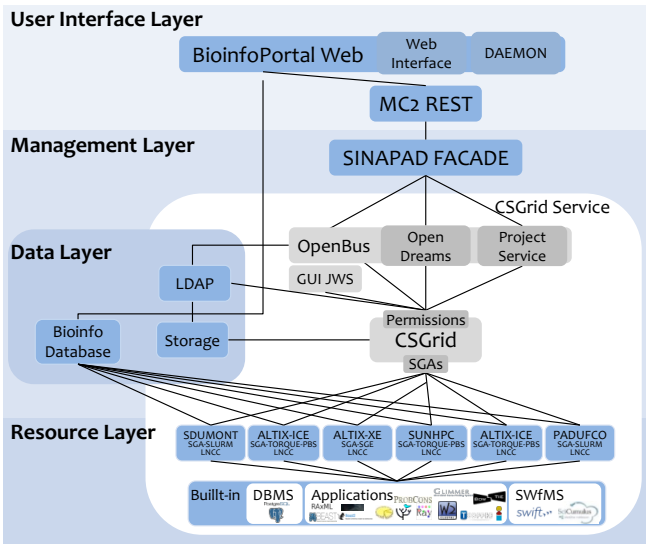

Fig. 1 BioinfoPortal Architecture

The *User Interface Layer* dynamically implements the BioinfoPortal interface; *i.e.,* it is a front-end where the user can submit executions of standalone applications or scientific workflows. The User Interface Layer is also responsible for

[4]http://www.lncc.br/

[5]http://www.phylo.org

returning the results by the users' e-mail. The *Management Layer* relies on CSGrid to access the data and job management services and provides features for job and workflow scheduling, sharing files, restricting anonymous access, and tracking provenance data. The *Resource Layer* is composed of the computational resources available for usage (in the context of this paper, the clusters, and supercomputers of SINAPAD) and hosts the bioinformatics software, libraries, SWfMS, and DBMS. The *Data Layer* provides access to the data storage in the relational databases of BioinfoPortal and authenticates the users using the repository Lightweight Directory Access Protocol (LDAP[6]). As aforementioned, BioinfoPortal relies on the middleware CSGrid and the computational resources of SINAPAD that provides functionalities of parallelism, distribution, and management of the executions using HPC/HTC capabilities. Table 1 presents the main features of the SINAPAD network.

TABLE 1 THE FEATURES AND FUNCTIONALITIES OF SINAPAD

Features	Functionalities
HPC/HTC	- Scalability, performance, optimization, distribution
Shared Architecture	- Shared memory - Types: SMP, UMA, NUMA, GPU, etc. - Parallelization: OpenMP6, Threads, PThreads, CUDA
Distributed Architecture	- Distributed memory, distributed resource management application API, storages, parallelization (MPI) - Heterogeneous resources: OGE, SGE-Oracle, Sun Grid, LoadLeveler, PBS, PBS Pro, TorquePBS, SLURM, LSF, - Clusters connected via HPC net (InfiniBand, fibrechannel) - Distributed file system: Lustre, NFS, pNFS
Hybrid architecture	- Shared + distributed memory
Cloud computing	- Virtualization - Pay-*per*-use resources - Low scale Brazilian academic resources - Data providers
Grid environment	- Types: HPC/HTC and opportunistic (volunteers) - Grid tools: Globus, Maui-Moab, Condor, CSGrid, Ourgrid - Data provider - Typically heterogeneous resources

A. The User Interface Layer (front-end)

It allows users to access several services and portals[7] of SINAPAD through two sub-layers: services and applications. The sub-layer of services is composed of the MC2 services [9], such as MC2 REST, which dynamically interacts with other services in the grid environment of SINAPAD.

The sub-layer of applications is composed of web clients, such as our BioinfoPortal Web that presents two components, Web interface and DAEMON. The Web Interface is used as front-end of users and has an authentication strategy that categorizes the type of access (public or private) assigned to users, which need to be previously registered at SINAPAD through an electronic form. The *public access* (for guest users) is provided with no authentication, but the service administrator can restrict options for using the computing resources. The *private access* (for private users) is provided with authentication as the service administrator provides more flexibility or no restriction. DAEMON is a robust fault tolerance mechanism for dispatching and scheduling jobs/workflows. It is connected to the Data Layer at Bioinfo Database for providing an entry for accessing the provenance data information (records of executions or specific domain-data provenance of bioinformatics applications).

The actual version of BioinfoPortal has public access that allows users to choose software, parameters, and input data for submitting their jobs/workflows. Finally, BioinfoPortal collects requirements, manages submissions, executes tasks, and finally report results sent to the browser and trigger sends on the individual e-mail results record.

B. The Management Layer

It is composed of three sub-layers: Facade, Bus Services, and CSGrid. The Facade sub-layer enables the configuration for connecting the services (Bioinfo-Portal, SINAPAD's services, HPC resources); searches the information needed for applications, clients, and users' authentication; manages files; and submits/monitors jobs/resources. The sub-layer for Bus Services[8] are implemented as a service-oriented software called OpenBus that allows for searching and publishing services by the client applications and for controlling the authentication/authorization of clients and services using internal governance management systems, digital certifications, and LDAP.

The sub-layer CSGrid is a client application of the OpenBus services (OpenDreams, ProjectService) that executes and manages data in the grid environment. It manages computational resources, clients, users, submissions, data, applications, and user databases (locally or LDAP) of BioinfoPortal at SINAPAD. CSGrid uses the Java-based graphical user interface (GUI) that allows SINAPAD to manage and implement internal applications.

The module SGA of CSGrid remotely manages the access of computational resources (Resource Layer). SGA is responsible for managing, submitting, scheduling, deleting, associating, and executing computational resources and jobs. SGA uses a pattern nomenclature, which is specific for each application executed in each cluster (*sga-<sinapadName>-<schedulerType>-<resourceName>_<queueName>*). SGA defines the best combination of settings (name, scheduler type, queue submission, parameters) to maintain the effective connectivity between SINAPAD's computational resources.

The "Concept Sandbox" is a mechanism associated with the execution of the computational resources. It provides the availability, reliability, and security of the data using two stages: the *stage-in* allows users to submit the input data sets and the *stage-out* returns the results of the executions to the users of a Project at CSGrid.

The "Concept Algorithm" (Fig. 2) is the structure at CSGrid where the applications are available. This term describes the non-interactive applications (using information as input/output data, parameters) that are dispatched directly by the clients of CSGrid. The Algorithm requires the information of applications – name, version, installation and compilation features in computational resources – to configure the client interface. For instance, as the application Align-m is installed in clusters (*sun.hpc, altix-xe, ice, sdumont*); then the Algorithm Align-m can be created at CSGrid and consequently, the interfaces of BioinfoPortal are able to access the Algorithm Align-m to call and execute the application Align-m in *sun.hpc, altix-xe, ice,* or *sdumont*.

The computational resources of SINAPAD are selected using two modes of requisitions: by *users* who need to choose the parameters, configurations, and clusters or by *schedulers*

[6]https://tools.ietf.org/rfc/rfc4511.txt
[7]https://www.lncc.br/sinapad/gateways.php?pg=gateways
[8]https://jira.tecgraf.puc-rio.br/confluence/display/OPENBUS020/Home

which automatically decide which resource is eligible for submissions. The metric used for schedulers is the number of free Central Processing Units (CPUs) in each cluster. Other metrics as the amount of free memory, disk space, network latency, or job submissions can be included by new plugins. The configuration of the actual version of BioinfoPortal uses public access (no user authentication) with automatic submissions using the mode of the automatic mode by schedulers, but it can be re-configured.

C. The Data Layer

It presents four sub-layers LDAP, Storage, Bioinfo Database, and Scientific Workflow Management Systems (SWfMS) Databases. The LDAP sub-layer manages the authentication of users for the portals of SINAPAD. The Storage sub-layer stores the information of the user data, application credentials, algorithm configurations, user/application permissions, and execution history proceeding of the application portals and CSGrid. The Bioinfo Database sub-layer stores the information (job monitoring, tasks executions, provenance data) proceeding of the applications executed in SINAPAD resources.

The SWfMS Database sub-layer includes the provenance databases of SWfMS SciCumulus [13] and Swift [14]. The management of the provenance data [15] records the history of executions and supports the analysis of scientific experiments. The SciCumulus database follows the data model PROV-Dff and W3C PROV and uses the PostgreSQL Relational Database Management System (RDBMS) to manage at runtime the provenance data information consisting of the performance of executions, structure of workflows, and domain-data. Swift is a script programming language for scientific workflows. It uses the server-less SQLite[9] relational database for storing the provenance data.

The decision-making or fault-tolerance predictions are possible by extracting the provenance information from the databases Bioinfo and SWfMS coupled to data mining and machine learning techniques. For instance, obtaining information, based on the history of records, about which tasks are running in which clusters and if the capacity of processing time and memory is enough can be used to predict/prevent errors of execution, which could affect all the services and portals of SINAPAD (including Bioinfo-Portal).

D. The Resource Layer

The computational resources of SINAPAD are formed by clusters, grids, and public/private clouds. A heterogeneous HPC cluster environment can contain processors and devices with different bandwidth and computational capabilities. Due to that reason, the installation or compilation of the applications and dependencies is made manually by developers/clients, following some requirements: (i) using parameters that optimize the compilation and installation of applications, libraries, and dependencies; (ii) using, as possible, mechanisms and technologies to distribute and parallelize tasks, *i.e.*, graphics processor unit (GPU), message passing interface (MPI), or threads; (iii) using mechanisms for optimizing the submissions of schedulers Torque/PBS, SGE, SLURM, Load Leveler; (iv) optimizing the use/configuration of parameters for SWfMS; and (v) coupling to Data Layer, resources as DBMS to register the provenance.

The *Resource Layer* presents two sub-layers. The sub-layer of applications with 35 software deployed. The sub-layer of SWfMS with SciCumulus [13] and Swift/T [14], which greatly reducing the complexity of managing experiments by providing features for scalable execution, scalable data management, fault-tolerance, and provenance data tracking [16]. The integration of SWfMS to CSGrid and computational resources of SINAPAD required the following steps. For SciCumulus, we implemented: (i) a network proxy that allows accessing to the SciCumulus provenance database; (ii) modification of the scripts of configuration (XML) and execution of applications (bash scripts) of SciCumulus, (iii) new bash scripts for mining labels ("tags" in the XML scripts of SciCumulus); and (iv) a connection between the databases of SWfMS and Bioinfo sub-layers. For Swift, we implemented bash scripts to (i) configure Swift at the HPC resources of SINAPAD; (ii) connect the database SQLite to CSGrid and SINAPAD; and (iii) configure the environment to manage execution jobs from a submission node to the clusters.

IV. Results and Discussion

This section presents the experimental evaluations for supporting the functionality of Bioinfo-Portal. We analyzed the performance and scalability of the applications RAxML, SciPhy, and SwiftGecko in HPC clusters. Analyses of the results were supported by the use of machine learning techniques for predicting the allocation/usage of computational resources for the gateway, based on the features dataset size of input data, software parameters, and efficiency.

The open source software RAxML is based on Maximum Likelihood (ML) algorithms for statistical calculations to construct phylogenetic trees used for inferring evolutionary life and phylogenetic relationships between genomes. The phylogenetic workflow SciPhy is managed with the SWfMS SciCumulus and aims at producing phylogenetic trees from input DNA, RNA and amino acid sequences. The workflow for parallel genome comparison SwiftGecko used the Swift parallel scripting system to identify blocks of large rearrangements, starting with the simple collection of ungapped local alignments. In Bioinformatics, comparative and evolutionary analyses of genomes is a traditional problem with high memory and CPU time requisites. Those requisites can be reduced using HPC techniques in its development.

The Orange Data Mining Framework [17] was used for data mining and machine learning analyses. Decision trees were generated with regression models using the provenance historic dataset of real application executions of BioinfoPortal that were obtained by querying databases from the Layer Data. In terms of the amount of data in the BioinfoPortal executions, we have queried the sub-layer Bioinfo database that contains the execution of 766 applications (performed by Brazilian and collaborator users) until 01/02/2019.

A. The Architecture of the Science Gateway Bioinfo-Portal

CSGrid provides the central infrastructure for developing the Algorithms and managing executions using the computational resources of SINAPAD. Each Algorithm presents scripts of configuration (".xml") and execution (".sh"), which are modified/implemented following the requirements of the applications or computational resources. The main scripts are *portal.xml* that configures the main web interface of Bioinfo-Portal; *config.xml* that configures the

[9] http://www.sqlite.org

parameters used by each Algorithm; *execute.sh* that executes the Algorithm based on the parameters of *config.xml*; and *prescript.sh* that presents the parameters to configure the environment for the computational resources. Fig. 2 presents the structure of CSGrid for the Algorithm RAxML, the scripts, and the available computational resources.

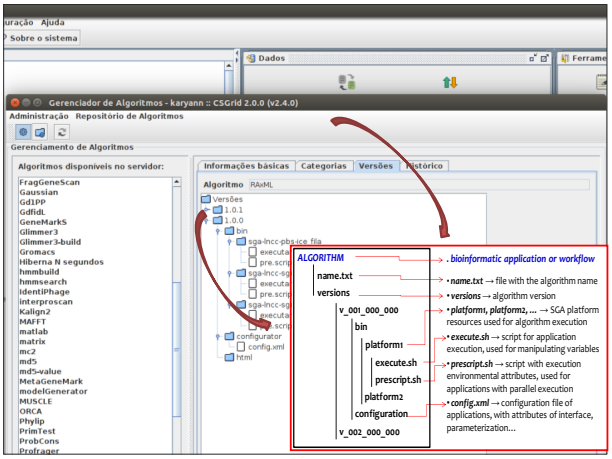

Fig. 2 The management of the Algorithm RAxML at CSGrid

1) The script *portal.xml* connects the web interface of BioinfoPortal to the Project area of the Algorithms. CSGrid defines which Algorithm will be executed by the gateway. The user must select the Algorithm; then, the web interface is dynamically built in the gateway and provides the fields of input/output and parameters that need to be filled by the user. The *portal.xml* allows that multiple versions of the same Algorithm be available at the gateway (XML attribute multiple-versions); then users can execute several versions of the applications. Alg. 1 shows an example of *portal.xml* for Algorithms RAxML [18] and SciPhy [19].

Alg. 1 The *portal.xml* configuration file of Algorithms RAxML and SciPhy

```
<portal-config multiple-versions='true'
  resource-choice='true'
  auto-generate='true'>
  ...
  <acronym-name>Bioinfo</acronym-name>
  <full-name>Bioinfo-Portal</full-name>
  <csgrid-proj-name>Bioinfo</csgrid-proj-name>
  <algorithms-config>
    <algorithm>
      <name>RAxML</name>
      <version>1.0.0</version>
    </algorithm>
    <algorithm>
      <name>SciPhy</name>
      <version>1.0.0</version>
    </algorithm>
  </algorithms-config>
  ...
</portal-config>
```

2) The script *config.xml* describes the input/output arguments required by each Algorithm. It is similar in purpose to the files of the JSON Service Description Language (JSDL) of the middleware UNICORE [20] and the Common Tool Description (CTD) of the OpenMS[10] library. Nevertheless, *config.xml* provides more details about how the arguments of the Algorithm can be rendered in a user interface *e.g.*, to the conditional attributes: "for the Algorithm RAxML, the fields A (amino acid type) and B (nucleotide type) are mutually exclusive". These attributes allow users to hide/disable field(s) at the web interface determined by one specific value of the flag. The *config.xml* is used by CSGrid for dynamically assembling the web interfaces and by OpenDreams service for invocating commands of scripts that reify the Algorithm in the underlying computational resources. Alg. 2 shows an example of *config.xml* for Algorithms RAxML and SciPhy.

Alg. 2 The *config.xml* configuration file of Algorithm RAxML

```
<?xml version='1.0' encoding='UTF-8'?>
<algorithm='execute.sh' code ='yes' id='yes'>
  <format>$NAME_PARAM=$VALUE_PARAM </format>
  <group='Input Parameters (Select)'>
  <name='INTYPE' label='Input' clue='Input' st='dna'>
      <item id='aa' val='aa' clue='aa file'/>
      <item id='dna' val='dna' clue='dna file'/>
    </name>
  </group>
  <group='Input Parameters'>
      <name='numCPU' label='CPU' min='1' max='16'
clue='threads' st='16' opt='true'/>
    <infile='INPUT_AA' label='aa' clue='aa'/>
    <infile='INPUT_DNA' label='dna' clue='dna'/>
     <name='BP' min='1' max='2000' clue='bp' st='100'
opt='true'/>
      <text='MODEL_AA' label='Mod aa' clue='Mod aa'
st='PROTGAMMAWAG' opt='true'/>
     <text='MODEL_DNA' label='Model dna' clue='Model
dna' st='GTRGAMMA' opt='true'/>
  </group>
  <group='Output Parameters'>
          <outfile='OUTPUT' label='dir' clue='dir'
category='directory'/>
  </group>
  <display>
    <parameter='INPUT_DNA'/>
    <parameter='MODEL_DNA'/>
    <condition='INTYPE' value='dna'/>
  </display>
  display>
    <parameter='INPUT_AA'/>
    <parameter='MODEL_AA'/>
    <condition ='INTYPE' value='aa'/>
  </display>
</algorithm>
```

3) The script *execute.sh* describes the specific arguments used for the execution of Algorithms for each one of the bioinformatics applications. Arguments can be extracted from the command line used to execute the application. Alg. 3 shows an example of *execute.sh* for the Algorithm RAxML.

Alg. 3 The *execute.sh* execution file of Algorithm RAxML

```
#!/bin/bash
. ~/setclasspath.sh
. /hpc/modulos/bash/openmpi-1.8.5-gcc47.sh
function parseArgs {
  for arg in $* ; do
    typeset name=`echo $arg | sed "s/=.*//"`
    typeset value=`echo $arg | sed "s/.*=//"`
    export $name=$value
  done
}
parseArgs $*          ###CSGrid Arguments###
echo "Id SGE: "$JOB_ID
echo $INTYPE, $INPUT_DNA, $INPUT_AA, $BOOTSTRAP,
$MODEL_AA, $MODEL_DNA, $OUTPUT
cmdId=`echo $cmdId | sed
's/\(.*\)\@\(.*\)\.\(.*\)/\1_\2_\3/'`
cd $OUTPUT            ###Execute RAxML DNA###
numCPU=$NSLOTS
CMD="/usr/bin/time -f "%e" -o ${OUTPUT}/.time.log 2
$ALG_DIR/RAxML/v_1/raxmlHPC-MPI-SSE3"
CMDmpi="/usr/bin/time -f "%e" -o ${OUTPUT}/.time.log
mpirun -np $numCPU -hostfile $TMPDIR/hostfile
$ALG_DIR/RAxML/v_1/raxmlHPC-MPI-SSE3"
if [ "$INTYPE" == "dna" ]
then
```

[10]https://www.openms.de

```
21   EXEC1="$CMD -m $MODEL_DNA -p 112233 -s $INPUT_DNA -
n $cmdId.best -c 4 -f d"
23   $EXEC1
24   EXEC2="$CMD -m $MODEL_DNA -p 112233 -s $INPUT_DNA -b
223344 -# $BP -n $cmdId.bp -c 4 -f d"
25   $EXEC2
26   EXEC3="$CMD -m $MODEL_DNA -s $INPUT_DNA -f b -t
best.$cmdId.best -z bp.$cmdId.bp -n $cmdId.bestML.bp"
27   $EXEC3
28   fi
29   if [ "$INTYPE" == "aa" ]
30   then
31   EXEC1="$CMD -m $ MODEL_AA -p 112233 -s $INPUT_AA -n
$cmdId.best -c 4 -f d"
32   $EXEC1
33   EXEC2="$CMD -m $MODEL_AA -p 112233 -s $INPUT_AA -b
223344 -# $BP -n $cmdId.bp -c 4 -f d"
34   $EXEC2
35   EXEC3="$CMD -m $MODEL_AA -s $INPUT_AA -f b -t
Tree.$cmdId.best -z bp.$cmdId.bp -n $cmdId.ML.bp"
36   $EXEC3
37   fi
38   if [ $? != 0 ]
39    then
40       echo "RAxML returned an error."
41       exit 1
42   fi
```

4) The script *prescript.sh* describes arguments used for the configuration of the environment. Alg. 4 shows an example of *prescript.sh* for the Algorithm RAxML.

Alg. 4 The *prescript.sh* execution file of Algorithm RAxML

```
1    #$ -N JOB_RAXML
2    #$ -cwd
3    #$ -V
4    #$ -pe mpi 16
5    #$ -l h_rt=216000
```

B. Implementation and Analyses of Performance

The experiments of RAxML, SciPhy, and SwiftGecko were executed in the cluster Altix ICE (Altix [11]) and supercomputer Santos Dumont (SDumont[12]). Altix consists of 25 diskless machines (execution nodes), each node is given by an SGI ICE 8400 server with 2 Intel Xeon X5650 2.67GHz Hexa Core processors (totaling 12 cores) and 48 GB DDR3 DIMMs of memory. The cluster has one login node with a gross storage capacity of 5TB. SDumont consists of 16 TB RAM, storage totaling 1,7 PetaBytes (Seagate 1.5 Buster), 10.692 cores, 1.1 PetaFlops, Intel Xeon E5-2695v2, 30 MB cache, 12 cores – 3.2 GHz. Machine learning analyses using regression models with decision trees were performed using results of executions, as will be detailed in Sub-section IV (C).

a) The Software RAxML. The compilation of the code of RAxML was adapted to the capabilities of the CPU(s) infrastructures. Depending on the processor features, RAxML supports three instructions used to substantially accelerate the likelihood and parsimony computations: the Streaming SIMD Extensions 3 (SSE3), the faster Advanced Vector Extensions (AVX), and the even faster AVX2 vector. RAxML presents four options for execution using multi-core shared memory systems; one sequential and three parallel using MPI, PThreads, or Hybrid (MPI + PThreads) instructions. The sequential version is used to process small to medium datasets. The parallel efficiency of the PThreads version depends on the alignment length. However PThreads works well for very long alignments, the performance is extremely hardware-dependent. The efficiency depends on the number of states of the data; the more states the data have (4 for DNA, 20 amino acid), the fewer site patterns are needed for an efficient parallel execution *per* thread/core. The parallel efficiency also depends on the rate of heterogeneity of the model; the GAMMA model entails more computations than the CAT model, which needs approximately 1/4 of the computations than the GAMMA model requires.

Fig. 3 shows (A) the web interface of the application RAxML at BioinfoPortal and (B) the layout of the Algorithm RAxML at CSGrid. The web interface of RAxML is automatically generated by the configuration script *config.xml* (Alg. 2) and the execution scripts *execute.sh* (Alg. 3) and *prescript.sh* (Alg. 4). The arguments required to execute RAxML through BioinfoPortal are defined by users, which must upload the input file (alignment in PHYLIP format) and fill options as input file type (nucleotide or amino acid), bootstrap value (default 100), substitution model (GTRGAMMA for nucleotide or PROTGAMMAWAG for amino acid), *e-mail*, and the output directory.

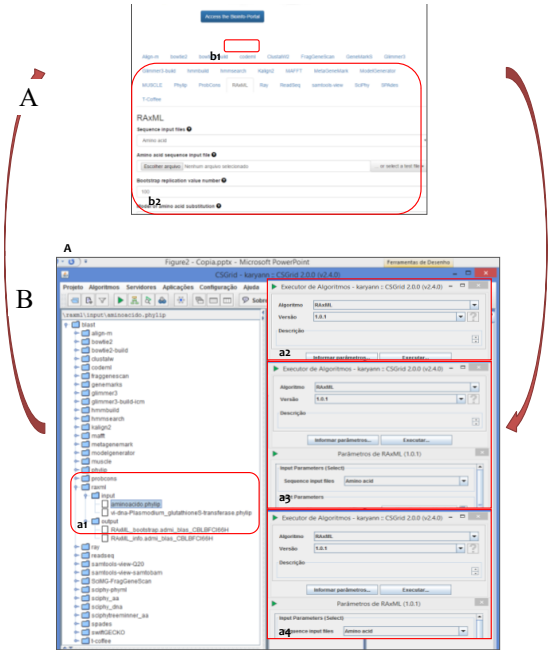

Fig. 3The Algorithm RAxML of BioinfoPortal

Three experiments with RAxML were performed in Altix and SDumont. The maximum number of cores *per* node for Altix is 12 and for SDumont is 24. RAxML was compiled with SSE3 in Altix and with AVX in SDumont.

1. The total execution time of the RAxML versions. RAxML HPC Serial was executed in one single core, RAxML PThreads in 12 cores, RAxML MPI in 120 cores, and RAxML Hybrid in 120 cores. Table 2 presents the features of the superalignments used as input data.

TABLE 2 FEATURES OF THE SUPERALIGNMENTS

Name	Features			
	Category	*Number of Taxa*	*Number of Amino Acid*	*Size in KB*
D1 (TotalAlignConc)	Large	74	21,260	2,091
D2 (TrimmedAlignConc)	Large	74	12,807	1,260
D3 (C1Total)	Medium	26	22,906	792
D4 (C1Trimmed)	Medium	26	16,068	553
D5 (C2Total)	Small	12	4,941	79
D6 (C2Trimmed)	Small	12	4,481	72
D7 (C3Total)	Medium	36	3,490	168
D8 (C3_trimmed)	Medium	36	3,270	157

[11] http://www.lncc.br/ice/

[12] http://sdumont.lncc.br/

The input file is one superalignment of amino acid sequences in format PHYLIP (called D5). The superalignment has a size of 79 KB and is formed of 31 concatenated universal orthologous (UO) genes belonging to 12 protozoan genomes. The parameters used for setting RAxML versions are JTT as an evolutionary model, GAMMA as the rate of model heterogeneity, and 100 as the bootstrap value of replications.

Fig. 4 presents the Total Execution Time (TET) in minutes obtained after the execution of the versions of RAxML. The TET decreases using the parallel versions PThreads, MPI, and Hybrid of RAxML in comparison to the version RAxML HPC Serial and using SDumont in comparison to Altix. For Altix, the RAxML MPI was the version that presented the best performance, with the TET reduced from 285,57 minutes (one single core) to 4.59 minutes (using 120 cores). For SDumont, the RAxML Hybrid was the version that presented the best performance, with the TET reduced from 161,29 minutes (using one single core) to 1.88 minutes (using 120 cores).

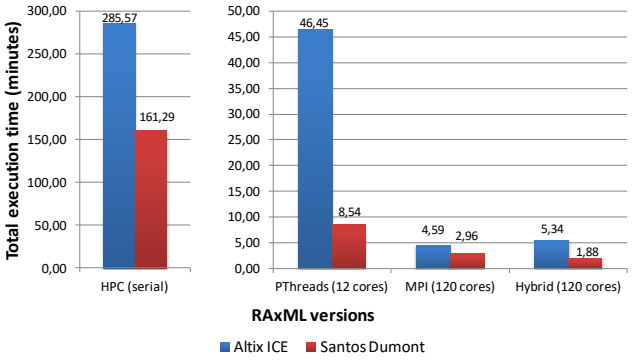

Fig. 4 The TET of the versions of RAxML executed in Altix and SDumont - superalignment D5

2. The scalability of RAxML MPI (Altix) and RAxML Hybrid (SDumont) using one superalignment. This experiment aims to evaluate the performance gains of the versions of RAxML that outperformed Altix (RAxML MPI) and SDumont (RAxML Hybrid) according to the number of cores in minutes, using one superalignment. The performance of RAxML was measured on a single processor machine (one core) to analyze the local optimization before scaling up the number of cores. After that, the performance and scalability of RAxML were measured using from 2 up to 120 cores. The superalignment used as input data is D5. Fig. 5 presents the scalability in minutes obtained after the execution of RAxML MPI in Altix and RAxML Hybrid in SDumont, using the superalignment D5.

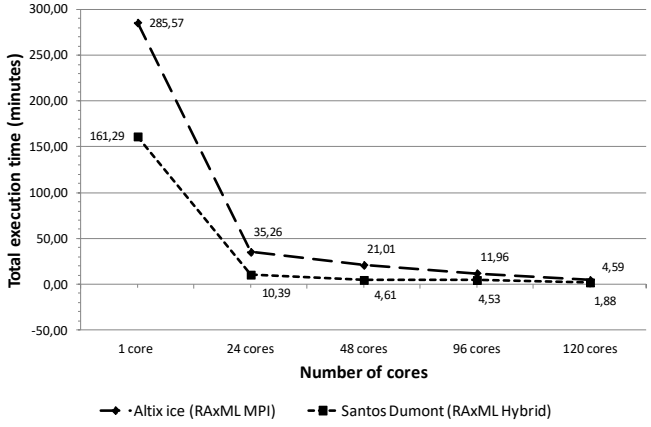

Fig. 5 The scalability of RAxML MPI in Altix and RAxML Hybrid in SDumont - superalignment D5

The TET decreases, in both cases, when more cores were provided to use. The RAxML Hybrid (SDumont) presented the best performance, when it processes one superalignment, the TET was reduced from 161.29 minutes (using one single core) to 10.39 minutes (using 24 cores) with a performance improvement of up to 93.56% and to 1.88 minutes (using 120 cores) with a performance improvement of up to 98.83%. Using one core, RAxML Hybrid at SDumont outperforms RAxML MPI at Altix; the TET was reduced from 285.57 minutes to 161.29 minutes with a performance improvement of up to 43.52%.

3. The TET of RAxML MPI (Altix) and RAxML Hybrid (SDumont) using eight superalignments. The executions of the RAxML versions used as input files eight superalignments, varying features of the number of taxa, number of amino acid, and size (KB). The parameters used for setting RAxML versions are JTT as the evolutionary model, GAMMA as the rate of the heterogeneity of the model, and 100 as the bootstrap value of replications.

Fig. 6 presents the TET in hours of RAxML MPI and RAxML Hybrid in Altix using eight superalignments. The TET decreases, for all superalignments, when the RAxML parallel versions were executed. The eight superalignments files were categorized as Large (D1, D2), Medium (D3, D4, D7, D8), and Small (D5, D6), depending on the features Number of Taxa, Number of Amino Acid, and Size in KB, as depicted in Table 2.

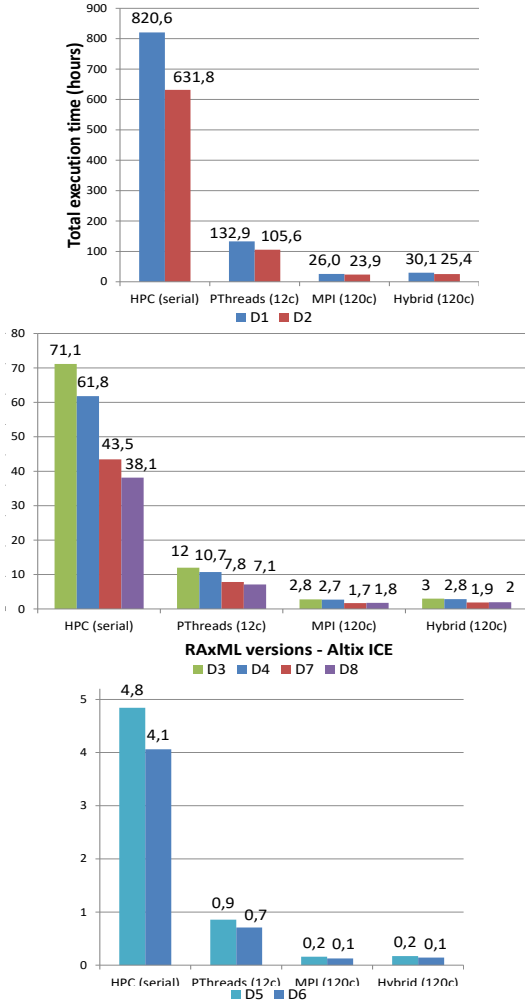

Fig. 6 TET of RAxML MPI and RAxML Hybrid in Altix - eight superalignments

The versions RAxML MPI and RAxML Hybrid outperformed the versions RAxML HPC Serial and RAxML PThreads. With RAxML Hybrid, the TET of the largest file D1 was reduced from 820.60 hours (using one single core) to 30.10 hours (using 120 cores) and the TET of the smallest file D6 was reduced from 4.10 hours (using one single core) to 0.10 hours (using 120 cores). A relation was observed between the increment of the size of the superalignments files and the improvement of performance of approximately 90% between 1 and 120 cores. D1: 2,091 KB and 96.33%; D2: 1,260 KB and 95.99%, D3: 792 KB and 95.78%; D4: 553 KB and 95.47%; D7: 168 KB and 95.63%; D8: 157 KB and 94.75%; D5: 95.83 KB and 97.83%; D6: 72 KB and 97.50%. These values indicate that parallelization benefits from larger files and that have more taxa and/or longer sequence lengths.

b) The Workflow SciPhy is composed of four activities: construction of multiple sequence alignment (MSA) with MAFFT; format conversion of MSA with ReadSeq; election of the evolutionary model with ModelGenerator; and (4) construction of trees with RAxML. The provenance database of SciCumulus stores the information of the workflow execution and data of SciPhy; and it is connected to the database Bioinfo that provides the access to the information of all Bioinfo-Portal's applications and all SINAPAD' services/portals. Queries to the provenance database of SciCumulus can be performed at runtime, which allows to the web interface report automatically the messages of the status of executions of SciPhy *e.g.*, which activity is finished or if an error is presented (Fig. 7).

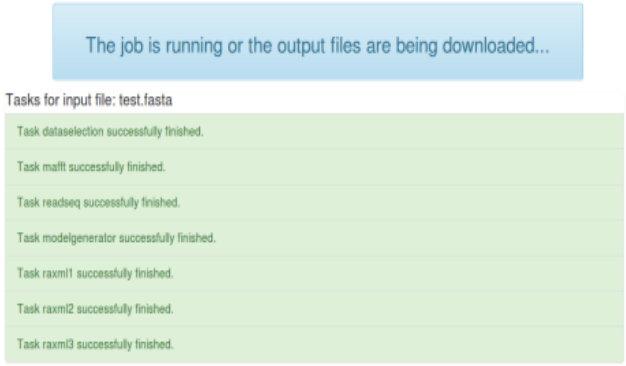

Fig. 7 The output of SciPhy executions at Bioinfo-Portal

The arguments used as input files and parameters required to execute SciPhy through BioinfoPortal are: to upload the input file (alignment in FASTA format) and to fill the following options, input file type (*e.g.*, amino acid), e-mail, and output directory. Fig. 7 shows print messages at the final of each activity execution of SciPhy at Bioinfo-Portal, obtained by querying the SciCumulus provenance database. The activities that were successfully finished were printed in the web interface in green; on the other hand, those ones that reported an error in the execution were printed in red.

We have evaluated the performance of the parallel execution of SciPhy in Altix. SciPhy was executed with 200 multifasta input files of amino acid sequences and three different configurations: on a single processor cluster (16 cores) through Bioinfo-Portal, on a single processor cluster (16 cores) directly at the cluster through CSGrid, and on three processor clusters (16 cores each) directly at the cluster through CSGrid. Fig. 8 presents the TET in minutes of SciPhy, which presents the best performance with the third configuration *i.e.,* when it is executed directly at CSGrid with three clusters (in green). The TET was reduced from 374.75 minutes (using one single cluster through Bioinfo-Portal) to 180.19 minutes (using three clusters through CSGrid).

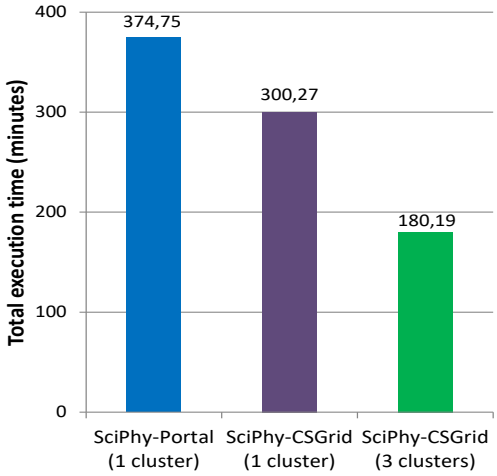

Fig. 8 The TET of the workflow SciPhy in Altix

c) The workflow SwiftGecko: maps the application GECKO [21] to Swift for providing parallelization and provenance data tracking. GECKO was designed to identify collections of high-scoring segment pairs (HSPs) by genome comparisons. The conceptual view of the workflow SwiftGecko is composed of three modules (M1 to M3) formed of 10 activities. (M1) The dictionary calculation: creates dictionaries for the sequence input and is composed for the activities 1-4; (M2) the detection of high-scoring segment pairs (HSP): detects the hits that identify the HSP locations and is composed for the activities 5-9; and (M3) The post-processing: performs the statistical calculations and functional annotations and is composed by the activity 10.

The provenance database of Swift stores the provenance data information as the TET and the scientific domain data information. By querying the provenance database of Swift, specialists can infer about the evolutionary and taxonomic relationship of genomes based on the number of fragments, also crossing with computational information of CPU consumed. The input arguments required by SwiftGecko at BioinfoPortal are one input file (genomes in FASTA format), one selection option for GECKO parameters (length, similarity, word length, FixedL), and an e-mail.

The executions (resulting in 135 task executions) were performed in a cluster with 72 nodes, 16GB of RAM, and eight computing cores *per* node. The input data is formed by five genomes of bacteria completely annotated. The TET was based on three metrics: a sequential execution directly at the cluster (in red); a parallel execution directly at the cluster (in blue); and a parallel execution through CSGrid via Bioinfo-Portal (in green). Fig. 9 presents the TET in minutes of SwiftGecko, which presents the best performance with the second configuration *i.e.,* using a parallel execution directly at the cluster (in blue). The TET was reduced from 8.2 minutes (sequential execution directly at the cluster) to 4.23 minutes (parallel execution directly at the cluster).

It is worth mentioning that SwiftGecko applications use an out-of-core strategy and are I/O intensive. Then, scalability could be improved by using higher throughput storage systems or more efficient data management strategies. Finally, the parallel execution through CSGrid via BioinfoPortal measures the time required for submission, execution, and file

transference between CSGrid/SINAPAD and the cluster, which increment the time of response via Bioinfo-Portal.

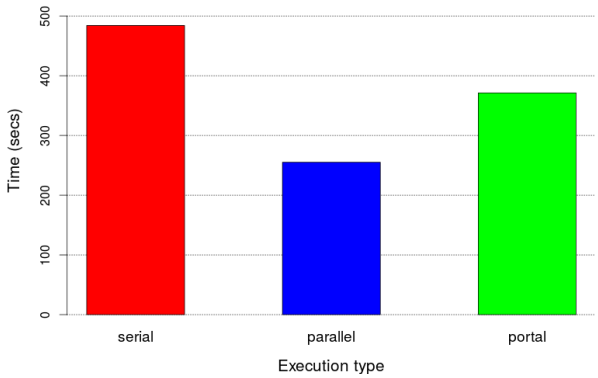

Fig. 9 The TET of the workflow SwiftGecko

C. *Statistics of the Gateway*

The database of Bioinfo and provenance databases of the SWfMS were integrated into the statistical tools of Google Analytics. The information of jobs executed *per* month, jobs executed *per* applications, or session *per* country can be accessed to Bioinfo-Portal. Fig. 10 shows the TET in hours *per* applications executed until 01/11/2018. The applications most accessed were FragGeneScan, Align-m, and Phylip. Until 01/02/2019, the portal was accessed 2,812 times and 766 applications were executed.

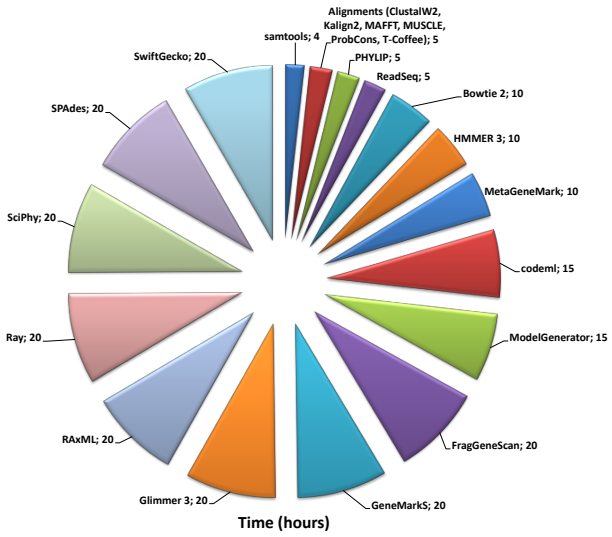

Fig. 10 TET of the applications of BioinfoPortal

An exploratory data analysis using classification trees was performed with the available data of the database of Bioinfo to infer knowledge about the most adequate computational resource to execute the applications. Data mining algorithms were used to discover patterns; we apply regression models with decision trees [22] using the Orange data mining tool for statistical analyses. Orange implements the core algorithm ID3 and employs a top-down, greedy search through the space of possible branches with no backtracking.

Nevertheless, before generating the decision tree, we had to evaluate the statistics that each attribute used in the model (*i.e.*, *threads*, *datasize*, and *node* attributes). The main idea of the attribute analysis is to identify potential problems with the chosen attributes and decide if an action needs to be taken that may require collecting more data. In Fig. 11(a), Fig. 11(b), Fig. 11(c), we can state that there is no one dominant attribute value and the distribution is not even; and for our analyses, results indicate that attributes can be used in the predictions.

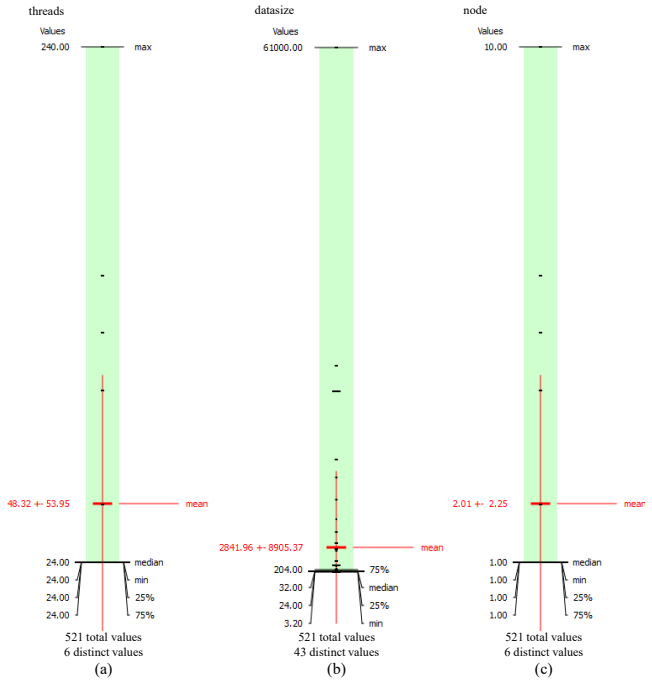

Fig. 11 The attribute statistics: (a) threads (number of threads), (b) datasize (data size of the alignment in KB), and (c) node (number of nodes)

In Fig. 11(a) evaluating the *threads* attribute, we can observe that 75% of the number of threads used was 24, from an interval of 24 to 240. Fig. 11(b) shows that 75% of data input presented size of 204 KB, from an interval of 3.2 to 610,000. Moreover, evaluating the *node* attribute in Fig. 11(c), we observed that 75% of the number of nodes used was 1, from an interval of 1 to 10. This information can assist users to distinguish outlier points to find anomalies or specific biological characteristics. By using these attributes to build estimation models, we can discover, using classification or regression algorithms, the relation of biological input data (size in KB) and the number of threads that can be used for the executions that generate maximum values of efficiency. Now we observed that the applications of BioinfoPortal consume low computational resources, which is due to the restriction for size and number of inputs assigned to the Projects of users.

Fig. 12 presents the inferred rules for exploring the efficiency of the executions of applications at Bioinfo-Portal. In this analysis, we can state that the efficiency of computational resources of BioinfoPortal applications is determined by two parameters: the number of threads (*threads*), the size of the alignment in KB (*datasize*), and the number of nodes (*node*) *i.e.,* the combination of values of these three parameters defines the efficiency of the executions.

For example, Fig. 12 presents that the efficiency that consumes the number of threads between 100 and 81 with less than 36,000 of size in KB of data size and will obtain, on average, 100 of efficiency. The machine-learning strategies appoint, for the actual scenario, that the best machine setup in a heterogeneous environment for executing applications presented at least 75% of efficiency. However, results obtained are interesting and provide an idea about the behavior of application executions in HPC computational resources *via* BioinfoPortal, more refined experiments with more data must to be reevaluated in future analyses.

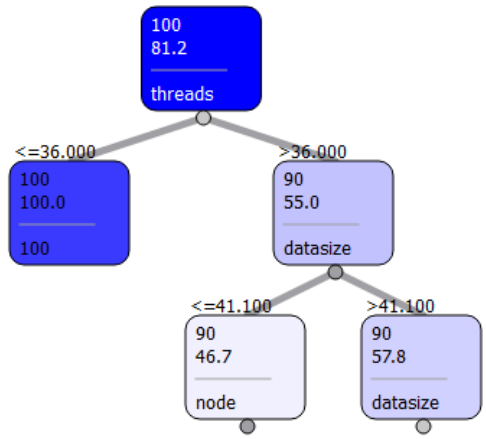

Fig. 12 The decision tree associated with the efficiency of the executions of the BioinfoPortal applications

V. Conclusions

We introduce our science gateway BioinfoPortal as a user-friendly interface that manages executions of bioinformatics applications with HPC/HTC technologies in computational resources, including clusters of supercomputers. BioinfoPortal is implemented over the grid middleware CSGrid and uses the computational resources of the Brazilian network SINAPAD, which bring-out grid computing management services. The experiences of SINAPAD at developing science gateways demonstrated the feasibility of the use of the technologies HPC/HTC [9].

Data analytics in science gateways are essential to support the exploratory nature of science. Large-scale experiments can benefit from data analytics facilities to evaluate the results, reduce the incidence of errors, decrease the total execution time, and sometimes reduce the financial cost. The data analysis process in science gateways needs to explore the statistics of applications execution, performance issues, and the content of data files. Each application execution in science gateways may consist of hours or days of processing, thus, it is unfeasible to perform an analysis without automatic semantic computational database support.

This paper evaluates the architecture of Bionfo-Portal, coupled with technologies of management systems and HPC resources to explore the processes of bioinformatics applications in an efficient manner. We are also concerned of coupling to BioinfoPortal with the best configurations for the efficient use of the computational resources, especially for the MPI and multithreading applications RAxML, SPAdes, FragGeneScan, MAFFT, Ray, Bowtie, and HMMER. However BioinfoPortal is free available and functional, there are several open challenges concerning to the maintenance of security in science gateways and coupling management systems for databases and workflows, which can be supported by machine learning technologies.

Acknowledgment

The funding for this research was provided by the Brazilian Advanced Network on Computational Biology (Grant no. CAPES 051/2013), Brazilian Bioinformatics Network (Grant no. CNPq 456644/2013-0), CNPq/Universal (Grant no. 429328/2016-8), and FAPERJ/JCNE (Grant no. 232985/2017-03). We are also grateful to the comments made by the anonymous referees.

This is a post-peer-review, pre-copyedit version of an article published in the Proceedings of the 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). The final version is available online at: https://doi.org/10.1109/CCGRID.2019.00082.

References

[1] D. C. Koboldt, K. M. Steinberg, D. E. Larson, R. K. Wilson, and E. R. Mardis, 'The next-generation sequencing revolution and its impact on genomics', *Cell*, vol. 155, no. 1, pp. 27–38. 2013.

[2] E. Bertino *et al.*, 'Challenges and Opportunities with Big Data', 2012.

[3] L. Dai, X. Gao, Y. Guo, J. Xiao, and Z. Zhang, 'Bioinformatics clouds for big data manipulation', *Biology Direct*, vol. 7, no. 1, p. 43, 2012.

[4] W. Aalst and K. Hee, *Workflow Management: Models, Methods, and Systems*. The MIT Press, 2002.

[5] R. Ramakrishnan and J. Gehrke, *Database management systems*, Third edition, International edition. New York: McGraw-Hill, 2003.

[6] S. Gesing *et al.*, 'Gathering requirements for advancing simulations in HPC infrastructures via science gateways', *Future Generation Computer Systems*. 2017.

[7] S. Gesing, J. Krüger, R. Grunzke, S. Herres-Pawlis, and A. Hoffmann, 'Using Science Gateways for Bridging the Differences between Research Infrastructures', *Journal of Grid Computing*, vol. 14, no. 4, pp. 545–557. 2016.

[8] I. Foster, C. Kesselman, and S. Tuecke, 'The Anatomy of the Grid: Enabling Scalable Virtual Organizations', *Lecture Notes in Computer Science*, vol. 2150, 2001.

[9] A. Gomes, B. F. Bastos, V. Medeiros, and V. M. Moreira, 'Experiences of the Brazilian national high-performance computing network on the rapid prototyping of science gateways', *Concurrency and Computation: Practice and Experience*, vol. 27, no. 2, pp. 271–289. 2015.

[10] S. Gesing *et al.*, 'A Single Sign-On Infrastructure for Science Gateways on a Use Case for Structural Bioinformatics', *Journal of Grid Computing*, vol. 10, no. 4, pp. 769–790. 2012.

[11] J. Goecks, A. Nekrutenko, and J. Taylor, 'Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences', *Genome Biology*, vol. 11, no. 8, p. 86, 2010.

[12] C. A. Goble *et al.*, 'myExperiment: a repository and social network for the sharing of bioinformatics workflows', *Nucleic Acids Res.*, vol. 38, no. Web Server Issue, pp. 677–682. 2010.

[13] D. De Oliveira, E. Ogasawara, F. Baião, and M. Mattoso, 'SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows', in *International Conference on Cloud Computing*, Washington, DC, USA, 2010, pp. 378–385.

[14] L. Gadelha, M. Wilde, M. Mattoso, and I. Foster, 'MTCProv: a practical provenance query framework for many-task scientific computing', *Distrib Parallel Databases*, vol. 30, no. 5, pp. 351–370, 2012.

[15] L. Moreau *et al.*, 'Special Issue: The First Provenance Challenge', *Concurrency and Computation: Practice and Experience*, vol. 20, no. 5, pp. 409–418, 2008.

[16] M. Mattoso *et al.*, 'Towards supporting the life cycle of large scale scientific experiments', *International Journal of Business Process Integration and Management*, vol. 5, no. 1, pp. 79–92, 2010.

[17] J. Demšar *et al.*, 'Orange: Data Mining Toolbox in Python', *Journal of Machine Learning Research*, vol. 14, pp. 2349–2353, Aug. 2013.

[18] A. Stamatakis, 'RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies', *Bioinformatics*, vol. 30, no. 9, pp. 1312–1313, 2014.

[19] K. Ocaña, D. Oliveira, E. Ogasawara, A. Dávila, A. Lima, and M. Mattoso, 'SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes', in *Adv. in Bioinformatics and Computational Biology*, Angra dos Reis, Brazil, 2011, pp. 66–70.

[20] K. Benedyczak, M. Wroński, A. Nowiński, K. S. Nowiński, J. Wypychowski, and P. Bała, 'UNICORE as Uniform Grid Environment for Life Sciences', in *Adv. in Grid Computing - EGC 2005*, vol. 3470, P. Sloot *et al*, Eds. Springer Berlin Heidelberg, 2005, pp. 364–373.

[21] O. Torreno and O. Trelles, 'Breaking computational barriers of pairwise genome comparison', *BMC Bioinformatics*, vol. 16, no. 1. 2015.

[22] J. Han and M. Kamber, *Data Mining: Concepts and Techniques, Third Edition*, 3 edition. Burlington, MA: Morgan Kaufmann, 2011.