Bio-UnaGrid: Easing Bioinformatics Workflow Execution Using LONI Pipeline and a Virtual Desktop Grid

Mario Villamizar, Harold Castro, David Mendez
Department of Systems and Computing Engineering
Universidad de los Andes
Bogotá D.C., Colombia
{mj.villamizar24, hcastro, dg.mendez67}@uniandes.edu.co

Silvia Restrepo, Luis Rodriguez
Department of Biological Sciences
Universidad de los Andes
Bogotá D.C., Colombia
{srestrep, luisrodr}@uniandes.edu.co

Abstract—**Bioinformatics researches use applications that require large computational capabilities regularly provided by cluster and grid computing infrastructures. Researchers must learn tens of commands to execute bioinformatics applications, to coordinate the manual workflow execution and to use complex distributed computing infrastructures, spending much of their time in technical issues of applications and distributed computing infrastructures. We propose the Bio-UnaGrid infrastructure to facilitate the automatic execution of intensive-computing workflows that require the use of existing application suites and distributed computing infrastructures. With Bio-UnaGrid, bioinformatics workflows are easily created and executed, with a simple click and in a transparent manner, on different cluster and grid computing infrastructures (line command is not used). To provide more processing capabilities, at low cost, Bio-UnaGrid use the idle processing capabilities of computer labs with Windows, Linux and Mac desktop computers, using a key virtualization strategy. We implement Bio-UnaGrid in a dedicated cluster and a computer lab. Results of performance tests evidence the gain obtained by our researchers.**

Keywords; Bio-UnaGrid; grid computing; cluster computing; bioinformatics; BLAST; mpiBLAST; desktop grid; UnaGrid; LONI Pipeline

I. INTRODUCTION

Today genomic projects involve the use of two complementary approaches: cluster computing which allows to aggregate homogeneous computational resources for a specific research group or organization, and grid computing which aggregates heterogeneous computational resources of different organizations to support larger computational capabilities in e-Science projects.

High performance computing (HPC) infrastructure, such as cluster or grid computing, provide results in shorter time, however when bioinformatics researchers want to execute applications, they face several consuming time problems: a) there are many application suites to run different analysis, so researchers must learn command line syntax (options, input and output files, distributed execution environment, etc.) for hundreds of applications. b) Researchers must learn commands to manage and use distributed computing infrastructures. c) Applications require specific and complex configurations to operate in distributed environments. d) Researchers frequently execute a set of applications in a sequential and/or coordinated manner, called pipelines or workflows. Workflows are regularly manually executed by researchers, so they must wait an application finishes, before executing the next. e) Very often researchers want to execute applications that require larger processing capabilities than those provided by dedicated computing infrastructures, so they must wait weeks or months before getting their results.

Different approaches have been developed for solving these problems partially on dedicated computing infrastructures. In this work we propose and evaluate an integral infrastructure that allows bioinformatics researchers, requiring large computing capabilities, to focus in bioinformatics analysis and not on technical computing issues of distributed computing infrastructures. The infrastructure, called Bio-UnaGrid, allows researchers to define workflows using graphical user interfaces (GUIs) and drag and drop tools provide by the LONI Pipeline [1] application. Workflows are executed and distributed in a transparent manner on a grid infrastructure. To support larger processing capabilities to those provided by dedicated infrastructures, Bio-UnaGrid also uses the UnaGrid [2] desktop grid infrastructure. UnaGrid permanently takes advantage of the idle processing capabilities available in desktop computers, while students do their daily activities.

Several existing bioinformatics application suites have been ported into Bio-UnaGrid. Performance tests were executed using the BLAST algorithm and its distributed implementations using *query segmentation*, provided by NCBI BLAST [3], and *database segmentation*, provided by mpiBLAST [4]. Test results show that Bio-UnaGrid allows to define workflows and to execute them on dedicated and desktop grid infrastructures, providing speedups 10 times higher than the speed on a desktop computer.

This paper is organized as follows: section 2 presents the related works. Section 3 describes the BLAST application, the UnaGrid infrastructure and the LONI Pipeline application. Section 4 details the Bio-UnaGrid architecture, including its integration with MPI (Message Passing Interface) applications. Section 5 describes the implementation deployed on a university campus. Results and performance tests are described in Section 6. Section 7 concludes and presents future work.

II. RELATED WORKS

Bioinformatics projects require the use of application suites for analysis that regularly require large computing capabilities. An analysis executed like a sequential computational job on a personal computer, may take weeks or months. Cluster computing infrastructures solve this problem aggregating the computational capabilities of several homogeneous computers, a cluster, to parallel execute a set of computational jobs. In a computational cluster there is a cluster master, a computer that submits a subset of jobs to other cluster computers: the slave nodes. Computational clusters are created by individual research groups to support their computational requirements.

Grid computing emerged as a technology that allows different organizations with common goals to create a virtual organization (VO) to share resources such as data, hardware and software. A grid computing infrastructure can group a great number of heterogeneous resources to support the computational requirements of the VO. In a grid infrastructure a set of computational jobs can be distributed among clusters belonging to different administrative domains. A drawback of this approach is that grid infrastructures require complex management processes.

To execute a bioinformatics analysis on a cluster or grid infrastructure, the application must be wrapped or designed to operate on these infrastructures. Wrapped application adapt an existing standalone application so it can be executed in parallel like a set of smaller and independent jobs, this approach is known as bags-of-tasks (BoT). In applications designed to operate on distributed infrastructures, a single job is automatically executed using several processes executed coordinately in a computational cluster or grid, using, regularly a MPI implementation.

Like BLAST, other bioinformatics applications are frequently used by researchers to execute different analysis in a coordinated and dependent manner, called workflows or pipelines, which require HPC infrastructures. The manual and command-based workflow execution requires bioinformatics researchers to spend most of their time in: configuring application parameters, managing HPC infrastructures, managing scientific data, and linking partial results from one application to another. Several projects have been developed to facilitate the automatic workflow execution in other scientific fields. Projects like Khoros [5], 3D Slicer [6], SCIRun/BioPSE [7] and Karma2 [8] for image processing; MAPS [9] for brain images; Trident [10] for oceanography; Kepler [11] and Swift [12] for agnostic area; MediGRID [13] for biomedical; Pegasus [14], OpenDX [15], and Triana [16] for heterogeneous applications; Taverna [17] is a framework to executed bioinformatics applications in distributed environments using MyGrid [18] middleware.

Most of workflow tools have limitations for bioinformatics applications such as: they require applications to be recompiled or modified, they support internal data sources with specific data structures; they operate with specific platforms, and cluster or grid middlewares; they are designed for solving needs of specific scientific fields; and they regularly require researchers to use complex commands involving consuming-time tasks.

Although dedicated cluster and grid infrastructures provides large computational capabilities, in a university or enterprise campus there are tens or hundreds of desktop computers that are under-utilized. Some clusters or grid solutions take advantage of the idle computing capabilities for e-Science projects, these systems are called Desktop Grid and Volunteer and Computing Systems (DGVCSs). Several DGVCS solutions has been developed using different approaches, solutions and architectures, including SETI@home [19], BOINC [20], OurGrid [21], Integrade [22] and UnaGrid [2].

From the conducted survey and our experience working on the field, we concluded that an infrastructure allowing existing bioinformatics applications to be easily incorporated without modifications in workflows created through GUIs and executed on different HPC infrastructures is needed. The infrastructure also must allow the use of different data sources (internal and external), and the incorporation of MPI applications commonly used in bioinformatics projects. For getting results faster the infrastructure should operate on dedicated computing infrastructures and on DGVCS that take advantage of the idle processing capabilities of tens of desktop computers with different operating systems. Bio-UnaGrid integrates the capabilities of LONI Pipeline and UnaGrid to provide these features.

III. LONI PIPELINE, UNAGRID AND BLAST

A. *LONI Pipeline*

LONI Pipeline [1] is a free framework used to execute neuroscience workflows (see [23]); however its design and implementation allows any command line application (like most bioinformatics applications) to be incorporated. From a computational perspective, LONI differs from other workflow tools in several features: it does not require external application being recompiled, it supports external data storage sources, it is hardware platform independent, and it can be installed on different cluster or grid middlewares using the Pipeline plugin Application Programming Interface (API) [1]. From a research user perspective LONI was designed having in mind features like usability, portability, intuitiveness, transparency and abstraction of cluster or grid infrastructures.

LONI Pipeline uses a client/server model. The server is installed on the master computer of a computational cluster, managed by a distributed resource management (DRM) system such as Oracle Grid Engine (OGE). LONI uses standard execution commands, so any applications executed through command line can be incorporated without requiring recompilations or new developments. An application is defined and loaded in the server through a GUI, defining an XML file, called a LONI Module that contains information about path application executable, and number and types of input and output parameters.

LONI client is a lightweight and standalone Java application that can be executed on Windows, Linux or Mac desktops. A researcher connects through the LONI client with the LONI server, using a customizable authentication protocol (LDAP, Database, etc.). The LONI modules are downloaded into the client and the researcher can begin to create workflows using simple drag and drop tools provided by an intuitive GUIs. Once a workflow is created, the researcher can begin its validation and execution. The workflow is executed (no commands are required) on the HPC infrastructure available for the master of LONI Pipeline. During its execution the researcher can disconnect of the LONI Server, and reconnect to query the partial results and to monitor the workflow state.

B. UnaGrid solution

UnaGrid is a DGVCS, developed at Universidad de los Andes, which provides the processing capabilities required by applications of different research areas at a university through the use and deployment of Customizable Virtual Clusters (CVCs) composed dynamically on demand, and executed on conventional desktop machines with Linux, Windows or Mac operating systems. The UnaGrid solution does not require applications to be recompiled or rewritten, and it has been used in projects of different research areas for executing BoT applications [2].

UnaGrid is a DGVCS that takes advantage of the idle processing capabilities of desktop computers within computer labs, in a non-intrusive manner, through the execution of Customized Virtual Clusters (CVCs). A CVC is a set of commodity and interconnected physical desktops computers executing virtual machines (VMs). While a student do his/her activities, a VM, playing a slave role of the CVC, is executed as a low-priority and background process on each desktop computer used by a student. A dedicated machine for the CVC plays the role of the cluster master. All of these VMs in execution make up a CVC, which has the operating system (mainly Linux), applications, and middleware required by the research group.

This model allows researcher to continue executing applications within their native environments, guaranteeing high usability of the infrastructure. Users access a CVC through a SSH connection to the CVC master. The use of virtualization tools such as VMware, Oracle Virtual Box, Citrix or Microsoft System Center, allows adding and taking advantage of the capabilities of tens or hundreds of machines in computer labs that have Windows, Linux or Mac operating systems, as well as the faculty to assign and limit the resources consumed by the VMs.

When a research group requires its CVC, they can deploy it on demand using a novel Web application called GUMA (Grid Uniandes Management Application). GUMA allows deploying on demand a previously configured CVC. A researcher securely access GUMA (using a Web browser) and defines the size (number of VMs) of the CVC he/she requires. GUMA automatically deploy the VMs on selected desktops, hiding the complexities associated with the location, distribution and heterogeneity of computing resources, and providing an intuitive graphical interface. GUMA also provides services for selection, shutdown and monitoring of physical computers and VMs. GUMA offers high usability to the UnaGrid solution, using the on-demand approach.

C. BLAST algorithmn

BLAST is a heuristic based application widely used to search for similarities between a set of biological sequences *S* and sequences in a database *D*. Wrapped applications using the NCBI BLAST implementation use the *query segmentation* approach. In this case a sequence query set *S* is divided in *n* subsets of sequences S_i. Each subset S_i is compared with the complete database *D*. A number *n* of NCBI BLAST independent jobs are executed on a computational cluster. Each independent job, executed by a cluster slave, compares a subset S_i with the database *D*. After the *n* jobs have been executed, the files generated by each job are merged in a single file containing all results.

Query segmentation offers great performance when the complete database *D* can be stored on the RAM memory of each cluster node. mpiBLAST adds *database segmentation*. mpiBLAST divides a sequence database *D* in *m* subsets D_j, and the sequence set *S* is compared with each subset D_j in a coordinated manner. A search using mpiBLAST is executed through a single mpiBLAST job executed coordinately on *m* slave nodes (logical MPI cluster) of a physical computational cluster. Each logical slave node compares the sequence set *S* with a subset D_j, and partial results of all logical slave nodes are stored in a unique shared file.

Several BLAST variations modify BLAST algorithm to execute relevant specific analysis and accelerate searches [24] such as PSI-BLAST, PHI-BLAST, Mega-BLAST MPBLAST, WU-BLAST2 and BLASTZ. Other BLAST solutions adapt or implement BLAST to operate efficiently on specific computing infrastructures [24]. BLAST solutions such as HGBS and FPGA-based BLAST require specialized hardware. BLAST implementation for dedicated cluster computing infrastructures are BeoBlast, NBLAST, Soap-HT-BLAST, mpiBLAST, Hyper-BLAST, dBLAST parallelBLAST, BLAST.pm, Parallel BLAST++, ScalaBLAST, pioBLAST and avaBLAST. These solutions use processing scheduling systems like Condor, PBS, OGE or Torque, and distributed storage systems like NFS or PVFS.

Solutions such as TurboBLAST, GBTK, GridBLAST CloudBLAST, G-BLAST, PackageBLAST and mpiBLAST-PIO, operates on dedicated grid computing infrastructures with standard middleware like Globus. W.ND BLAST [25], BOINC BLAST [26], and BLAST on BitDew [27], use the processing capabilities of DGVCSs to execute searches. W.ND BLAST only operates on Windows desktops. BOINC and BitDew require every application being modified.

IV. Bio-UnaGrid Infrastructure

We propose the Bio-UnaGrid infrastructure, which was designed to facilitate to bioinformatics researchers the use of bioinformatics applications, the automatic execution of workflows, and the use of HPC infrastructures. With Bio-UnaGrid bioinformatics researchers can use the processing capabilities of dedicated computational clusters, and the idle processing capabilities provided by the UnaGrid DGVCS.

Using the LONI features, Bio-UnaGrid allows that existing and new bioinformatics application can be easily integrated and used by researchers without been recompiled or modified. Current version of LONI Pipeline does not support the execution of MPI applications such as mpiBLAST, however, Bio-UnaGrid support the execution of this type of applications.

A. *Bio-Unagrid Architecture*

Bio-UnaGrid is based on the integration of two main solutions: LONI Pipeline and UnaGrid. The Bio-UnaGrid architecture is shown in Figure 1.

The *LONI Pipeline Client* is the main entry point of bioinformatics researchers to the Bio-UnaGrid infrastructure. Researchers use the LONI client to connect with the *LONI Pipeline Server*, through an authentication process against a *User Database*. Each researcher receives a username and a password. After authentication the researcher can view the bioinformatics applications installed on the *LONI Pipeline Server* through the *LONI Pipeline Client*, and he/she can proceed to create the workflows, using the bioinformatics *LONI Module Library*, drag and drop, and other tools.

For each *LONI Module* (a bioinformatics application), the researcher must define the input and output parameters (which can take values such as strings, path files, path directories, numbers and related lists), and how each *LONI Module* is connected with other LONI Modules of the workflow. A single *LONI Module* may be used to execute several jobs, for example if the BLASTn *LONI Module* of the NCBI BLAST suite, receives as input parameters ten sequence query files and a database, like *nt*, the *LONI Pipeline Server* will execute ten independent BLASTn jobs, each one comparing a sequence query file with the *nt* database. Once a workflow has been created, researchers can execute it, sending it to the *LONI Pipeline Server*. Researcher can monitor and visualize the status of the workflow, and download partial results, through GUIs of the *LONI Pipeline Client*.

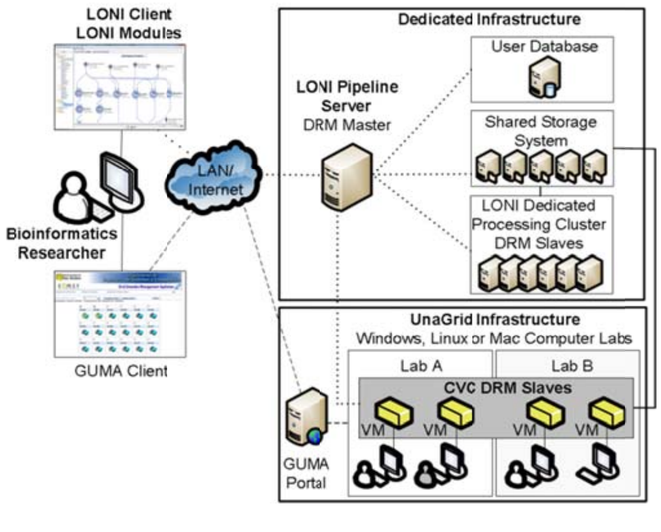

Figure 1. Bio-UnaGrid architecture.

The *LONI Pipeline Server* is a dedicated server with the master component of a DRM such as OGE installed locally. The *LONI Pipeline Server* receives the workflows sent by researchers; divide the workflows in individual jobs and send them, in an order manner, to the *DRM Master*. The *DRM Master* distributes the jobs to the *DRM Slaves* which can be running on a *Dedicated Processing Cluster* or on a CVC of UnaGrid, composed of VMs executed on heterogeneous desktop computers, which we will call *CVC DRM Slaves*.

The *Dedicated Processing Cluster* is permanently available for researchers; however its capabilities are limited. To provide more processing capabilities, the *CVC DRM Slaves* can be deployed on demand using the *GUMA Portal* of the UnaGrid infrastructure. Before a researcher execute a workflow or during the workflow execution, he/she can access the *GUMA Portal* using a username and a password. Through *GUMA Portal* researchers define the execution time and number of virtual machines (*VMs*), previously configured by the UnaGrid support team, he/she required for the workflow execution. *GUMA Portal* deploys bioinformatics *VMs* on different desktop computers distributed along computers labs in the university campus. When the *CVC DRM Slaves* are turn on by GUMA, they contact the *DRM Master*. The *DRM Master* begins to submit individual jobs to *CVC DRM Slaves*.

A *Shared Storage System* is used to store the *LONI Modules*, the binaries of bioinformatics applications, and input and output data required for the workflow execution, including genomic databases. The *DRM Master*, the *Dedicated Processing Cluster* and the *CVC DRM Slaves* access the *Shared Storage System* during workflow executions. During workflow creation researchers can select local files of the personal computer where the *LONI Pipeline Client* is executed, and these files are transferred automatically to the *LONI Pipeline Server* and stored in the *Shared Storage System*.

For executing workflows bioinformatics researchers do not have to worry about applications' commands or complex HPC infrastructures. They only need to define workflows using drag and drop tools provided by *LONI Pipeline Client*, and run workflows using a single click. Workflows are executed faster and transparently on dedicated clusters, and on tens or hundreds of commodity desktop computers provided by UnaGrid. When a workflow is finished, researchers can download the results of all *LONI modules* using GUIs, and they are ready to do their bioinformatics analysis.

B. *Bioinformatics applications on Bio-UnaGrid*

The process to incorporate existing and new bioinformatics application suites, like NCBI BLAST, into Bio-UnaGrid infrastructure are summarized in five steps.

1) *Installation and configuration of the application suite on the Shared Storage System:* This installation is executed from the *LONI Pipeline Server*.

2) *Creation of a LONI Pipeline Module using the LONI Pipeline Client for each executable of the suite:* for this it is

necessary to identify the input and output parameters of the executable, specifying its dependences and data types.

3) Individual tests for each LONI Pipeline Module of the suite from the LONI Pipeline Client.

4) Storage of the LONI Pipeline Module in the LONI Pipeline Server.

5) Using the modules from workflows created by bioinformatics researchers in the LONI Pipeline Client.

. This process is executed by the HPC infrastructure's system administrator only once for each bioinformatics application. After a *LONI Module* for a bioinformatics application is created, all researchers can use it in their workflows. This process allows new applications to be agilely incorporated, facilitating the creation of more complex workflows. Our experience in the deployment of several application suites show that any application that can be used from the command line can be integrated into the Bio-UnaGrid solution.

C. MPI applications on Bio-UnaGrid

At the time of this development, LONI did not support MPI applications. To allow the execution of MPI applications, we developed a wrapper, called *mpiJobManager*, which uses the Distributed Resource Management Application API (DRMAA). DRMAA allows sending and monitoring jobs executed on computational cluster using a Distributed Resource Manager (DRM) such as the Oracle Grid Engine (OGE). To execute an MPI Application, such as mpiBLAST, a researcher uses the *MPI LONI Module* of the MPI application to specify the number of MPI processes to be used. *MPI LONI Modules* are received by the *LONI Pipeline Server* and executed on a *DRM Slave* (a dedicated server or desktop computer) using the *mpiJobManager* wrapper.

When an *mpiJobManager* job is executed, it executes a Linux script, called *launcherMPIJob*. The *launcherMPIJob* script sends an MPI job to the *LONI Pipeline Server*, specifying the process number of the job. The *LONI Pipeline Server* receives the MPI job and proceeds to execute it on the *DRM slaves*. During its execution, the MPI job is monitored by the *LONI Pipeline Server* using the *mpiJobManager*. Because of the design of the LONI Pipeline Server, researchers can query the results of a whole MPI job but not the status of its individual processes.

V. IMPLEMENTATION

Bio-UnaGrid was implemented at Universidad de los Andes, involving the current bioinformatics dedicated processing cluster from the Department of Biological Sciences, some dedicated servers and a computer lab with Windows desktop computers from the Department of Systems and Computing Engineering. Bio-UnaGrid was implemented to support several genomic projects related to coffee, potato and cassava, which seek genomic analysis to improve coffee, potato and cassava production affected by different biological organisms that decrease their production [28] [29]. The implementation is illustrated in Figure 2.

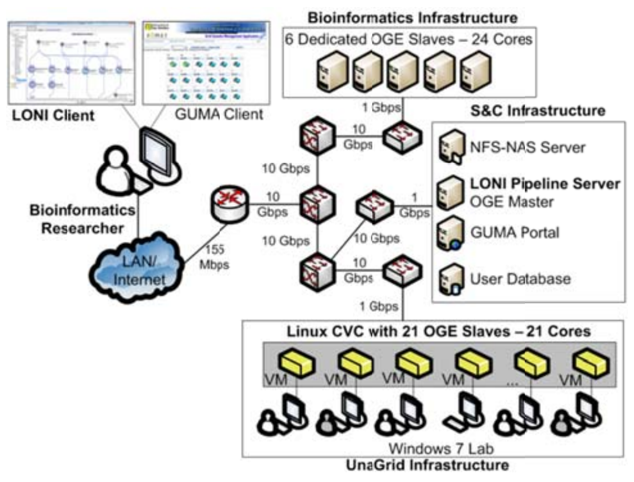

Figure 2. Bio-UnaGrid implementation.

The *LONI Pipeline Server* version 5.0.2 and the *DRM Master* of OGE version 6.2 update 5, were installed on a dedicated server. We use an NFSv3-NAS (Network File System version 3 - Network Attached Storage) solution like the *Shared Storage System*. The *Database User* was installed on a dedicated server using MySQL version 5.1.5. On the dedicated cluster used by the Department of Biological Sciences composed by 6 servers each one with an Intel Xeon X5560 Quad Core processor of 2.8 GHz and 4 GB of RAM, was installed the Slave component of OGE, so these servers now can process jobs sent from *LONI Pipeline Clients*.

To test the performance of bioinformatics applications on the desktop computers, a CVC with 21 VMs was deployed on a computer lab with Windows 7 desktops computers, which have an Intel i5 processor of 3.46 GHz and 8 GB of RAM. On the CVC we installed Debian 4.0 and the Slave component of OGE, so these VMs can also process jobs sent from *LONI Pipeline Clients*. VMs were configured with a CPU core and 4 GB of RAM. All nodes are interconnected through 1 GbE and 10GbE links and fault tolerance mechanisms were configured for BoT applications using the OGE capabilities.

Four bioinformatics application suites have been installed on the Bio-UnaGrid implementation: NCBI BLAST version 2.2.20, HMMER version 2.3.2, InterProScan version 4.6 and mpiBLAST version 1.6.0. Because mpiBLAST requires the use of an MPI implementation, the MPICH-2 implementation version 1.2.7 was installed. 21 *LONI Modules* were created for the application suites: 6 for NCBI BLAST (BLASTn, BLASTp, BLASTx, tBLASTn, tBLASTx and MEGABLAST); 3 for HMMER (Build, Search and Calibrate); 11 for InterProScan (BlastProdom, Coils, Gene3D, PIR, Panther, Pfam, SEG, SMART, SuperFamily, TIGRfam and fPrintScan) and 1 for mpiBLAST.

Researchers now can execute these bioinformatics applications from workflows. An example of a workflow created through the *LONI Pipeline* Client with these applications, and executed on the HPC infrastructure is shown in Figure 3.

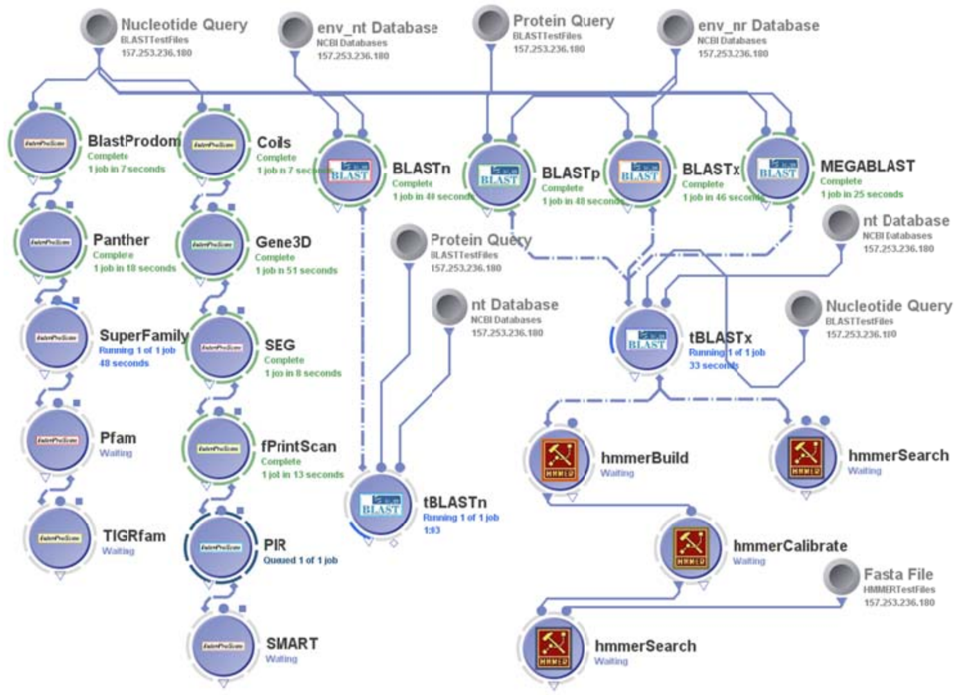

Figure 3. A bioinformtics workflow executed on Bio-UnaGrid.

During the workflow execution bioinformatics researchers can see the status of the complete workflow, and of each individual *LONI Module* (Completed, Running, Queued, and Waiting). In the workflow shown in Figure 3, all files (query, databases and FASTA) are stored in the *Shared Storage System*. This workflow is being executed on the configured distributed infrastructure and researchers do not have to use any command; all workflow operations are executed using GUIs.

VI. PERFORMANCE TESTS AND RESULTS

Bio-UnaGrid support dedicated clusters and desktop machines; we executed performance tests in both environments, in the dedicated cluster and in the UnaGrid CVC. We selected the high popular BLAST algorithm to execute these tests. A detailed time-performance characterization for the execution of BLAST (using *query segmentation*) and mpiBLAST in cluster and grid infrastructures can be found in [30] and [4] respectively.

For testing the performance of Bio-UnaGrid for executing BoT applications we used the NCBI BLASTn application, varying the number of CPU cores and the size of the sequence set. We executed BLASTn searches between nucleotide sets of 480 (487 KB) and 960 (954 KB) sequences, and the *nt* database (28 GB). The searches were executed using 5, 10, 15 and 20, BLASTn independent processes (BoT BLASTn). In both environments each independent process was executed using a CPU core to compare a sequence set with the complete *nt* database. We use *query segmentation*, using a LONI Module, for dividing the sequence sets in the same number of processes; subsets of 96, 72, 48 and 24 sequences were used for the 480 sequence set.

All tests were executed 3 times and we calculated the average time. Two metrics were used in the performance tests, the execution time and the speedup. The speedup is a measure that indicates the improvement in the execution time of a parallel algorithm when it is compared with same algorithm executed in a sequential manner [31]. The sequential searches were executed in a desktop of the UnaGrid CVC and in a server of the dedicated cluster, using a standalone BLASTn search. Sequential searches were executed using a single CPU core. The execution times of the sequential searches with the sets of 480 and 960 sequences on the dedicated server were 6,436 and 13,140 seconds respectively, and on the desktop were 12,288 and 19,488 seconds.

The execution times of the searches on the dedicated cluster and the UnaGrid CVC, using BoT BLASTn, are shown in figure 4.

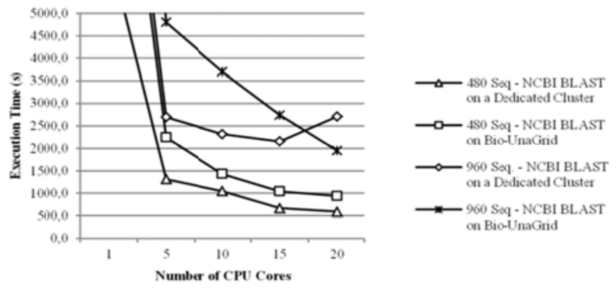

Figure 4. Execution time of BoT BLASTn searches between sets of 480 and 960 sequences, and the *nt* database.

Results in Figure 4 show that the execution time of the searches in both environments, decrease when the process number (CPU cores) is increased (except in the search with 960 sequences on the dedicated cluster using 20 processes). Taking as reference the execution time of the sequential searches on each environment, we calculated the speedups shown in Figure 5. The execution time of the search with 480 sequences on the dedicated cluster was reduced from 6,436 to 589 seconds (using 20 processors), providing results 10.9 times faster (10.9x). On the Bio-UnaGrid CVC the execution time was reduced from 12,288 to 936 seconds, using 20 processors (13.1x faster). For the search with 960 sequences, the execution time on the dedicated cluster was reduced from 13,140 to 2,152 seconds when 15 processors were used (6.1x faster), and on the Bio-UnaGrid CVC from 19,488 to 1,946 second using 20 processors (10x faster).

For testing the execution of MPI applications on both environments, we executed the same tests using mpiBLAST. In these tests we executed MPI processes coordinated on 5, 10, 15 and 20 CPU cores. In each test, the additional process called *mpiJobManager* was executed in another CPU core (this is the reason to show 21 OGE Slaves on Figure 2). We used *database segmentation* to divide the database in 5, 10, 15 and 20 partitions using the *mpiformatdb* installed with mpiBLAST, using a LONI Module. The execution times of these searches using mpiBLAST are shown in Figure 6. The speedups for these searches are shown in Figure 7.

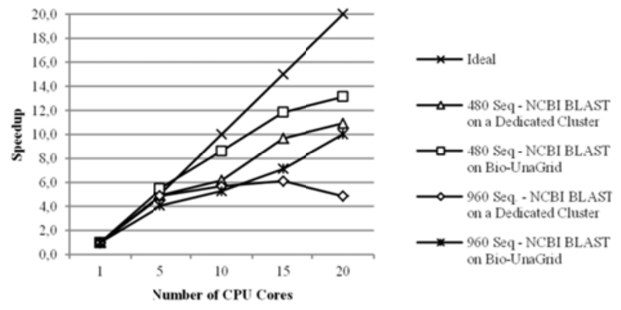

Figure 5. Speedups of BoT BLASTn searches between sets of 480 and 960 sequences, and the *nt* database.

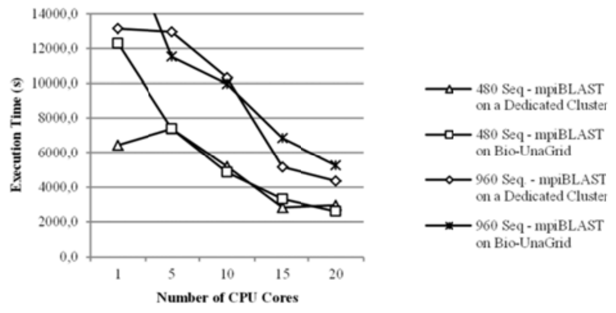

Figure 6. Execution times of mpiBLAST searches between sets of 480 and 960 sequences, and the *nt* database.

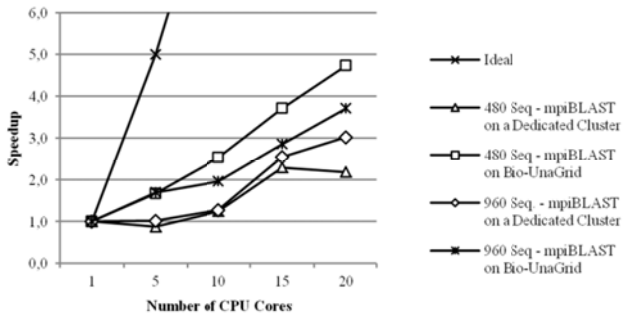

Figure 7. Speedups of mpiBLAST searches between sets of 480 and 960 sequences, and the *nt* database.

Similar to previous tests, using mpiBLAST on both environments, in most tests the execution time is reduced when the number of processor is increased. On the dedicated cluster the execution time for the search with 480 sequences was reduced from 6,436 to 2,815 seconds when 15 processors were used (2.3x faster). On the Bio-UnaGrid CVC the execution time was reduced from 12,288 to 2,597 seconds, using 20 processors (4.7x faster). For the search with 960 sequences, the execution time on the dedicated cluster was reduced from 13,140 to 4,154 seconds when 20 processors were used (3x faster), and on the Bio-UnaGrid CVC from 19,488 to 5,255 second using 20 processors (3.7x faster).

Results from Figures 5 and 7 show that Bio-UnaGrid can provide speedups similar or higher to those provided by a dedicated cluster, decreasing the result generation time when the number of processes is increased. Using the idle processing capabilities, Bio-UnaGrid provides additional processing capabilities to those provide by dedicated computational clusters, decreasing more than 13.1 times (13.1x) the execution time of bioinformatics workflows.

VII. CONCLUSIONS AND FUTURE WORK

We proposed Bio-UnaGrid, an infrastructure that allows bioinformatics researchers to easily define workflows using GUIs and execute them on different cluster and grid computing infrastructures with a simple click. Bio-UnaGrid is highly extensible as existing bioinformatics applications can be incorporated without modifications. With Bio-UnaGrid, researchers focus on analysis of application results, not on technical issues of distributed computing infrastructures. To take advantage of more processing capabilities, besides using dedicated computing infrastructures, Bio-UnaGrid also use the idle processing capabilities of tens or hundreds of desktop computers commonly available in computer labs.

We implemented Bio-UnaGrid in a dedicated cluster composed of 6 servers, and a computer lab with several bioinformatics application suites such as NCBI BLAST, HMMER, InterProScan and mpiBLAST. Performance tests with NCBI BLASTn and mpiBLAST show that Bio-UnaGrid can reduce the sequential execution time up to 13.1 times faster. These results show promising opportunities for bioinformatics researchers to get results in shorter times using the idle processing capabilities available in computer labs using Windows, Linux or Mac desktops.

As future work, we will incorporate more bioinformatics applications suites in Bio-UnaGrid and we will execute new performance tests with other applications such as HMMER, MPI-HMMER, ClustalW and ClustalW-MPI, in a larger grid deployment involving hundreds of heterogeneous desktop computers in different administrative domains. We will implement a shared storage system more scalable than NFSv3. We also plan to implement a fault tolerance mechanism for MPI applications.

REFERENCES

[1] I. Dinov et al., "Neuroimaging Study Designs, Computational Analyses and Data Provenance Using the LONI Pipeline," PLOS ONE, vol. 5, Sep. 2010, doi:10.1371/journal.pone.0013070.

[2] H. Castro, E. Rosales, M. Villamizar, and A. Miller, "UnaGrid - On Demand Opportunistic Desktop Grid," Proc. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, June 2010, pp. 661-666, doi:10.1109/CCGRID.2010.79.

[3] S. Altschul et al., "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," Nucleic Acids Research, vol. 25, July 1997, pp. 3389-3402, doi:10.1093/nar/25.17.3389.

[4] A. Darling, L. Carey, and W. Feng, "The design, implementation, and evaluation of mpiBLAST," Proc. ClusterWorld 2003, June 2003.

[5] S. Kubica, T. Robey, and C. Moorman, "Data parallel programming with the Khoros data services library," Lecture Notes in Computer Science, vol. 1388, 1998, pp. 963-973, doi:10.1007/3-540-64359-1_762.

[6] S. Pieper, M. Halle, and R. Kikinis, "3D SLICER," Proc. IEEE International Symposium on omedical Imaging: Nano to Macro, April 2004, pp. 632-635, doi:10.1109/ISBI.2004.1398617.

[7] R. MacLeod, D. Weinstein, D. Germain, D. Brooks, C. Johnson, and S. Parker, "SCIRun/BioPSE: integrated problem solving environment for bioelectric field problems and visualization," Proc. IEEE International Symposium on Biomedical Imaging: Nano to Macro, April 2004, pp. 640-643, doi:10.1109/ISBI.2004.1398619.

[8] Y. Simmhan, B. Plale, and D. Gannon, “Karma2: Provenance Management for Data Driven Workflows,” International Journal of Web Services Research, vol. 5, April 2008, pp. 1-22, doi:10.4018/jwsr.2008040101.

[9] B. Lucas, B. Landman, J. Prince, and D. L. Pham, “MAPS: A Free Medical Image Processing Pipeline,” Proc. 14th Annual Meeting of the Organization of Human Brain Mapping, June 2008.

[10] Y. Simmhan, R. Barga, C. Ingen, E. Lazowska, and A. Szalay, “Building the Trident Scientific Workflow Workbench for Data Management in the Cloud,” Proc. Third International Conference on Advanced Engineering Computing and Applications in Sciences, Oct. 2009, pp. 41-50, doi:10.1109/ADVCOMP.2009.14.

[11] B. Ludäscher et al., “Scientific workflow management and the Kepler system,” Concurrency and Computation: Practice & Experience, vol. 18, Aug. 2006, pp. 1039-1065, doi:10.1002/cpe.v18:10.

[12] Y. Zhao et al., “Swift: Fast, Reliable, Loosely Coupled Parallel Computation,” Proc. IEEE Congress on Services, IEEE, July 2007, pp. 199-206, doi:10.1109/SERVICES.2007.63.

[13] D. Krefting et al., “MediGRID: Towards a user friendly secured grid infrastructure,” The international journal of grid computing-theory methods and applications, vol. 25, March 2009, pp. 236-336, doi:10.1016/j.future.2008.05.00.

[14] E. Deelman et al., “Pegasus: Mapping Scientific Workflows onto the Grid,” Lecture Notes in Computer Science, vol. 3165, 2004, pp. 131-140, doi:10.1007/978-3-540-28642-4_2.

[15] D. Thompson, J. Braun, and R. Ford, “OpenDX: Paths to Visualization,” Proc. VIS, Inc.

[16] D. Churches et al., “Programming scientific and distributed workflow with Triana services,” Concurrency and Computation: Practice & Experience, vol. 18, Aug. 2006, pp. 1021-1037, doi:10.1002/cpe.v18:10.

[17] T. Oinn et al., “Taverna: lessons in creating a workflow environment for the life sciences,” Concurrency and Computation: Practice and Experience, vol. 18, Aug. 2006, pp. 1067-1100, doi:10.1002/cpe.v18:10.

[18] MN Alpdemir et al., “Contextualised workflow execution in MyGrid,” Lecture Notes In Computer Science, vol. 3470, 2005, pp. 444-453, doi:10.1007/11508380_46.

[19] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, “SETI@home An Experiment in Public-Resource Computing,” Communications of the ACM, vol. 45, Nov. 2002, pp. 56-61, doi:10.1145/581571.581573.

[20] D. Anderson, “BOINC: A System for Public-Resource Computing and Storage,” Proc. in 5th IEEE/ACM International Workshop on Grid, IEEE, Nov. 2004, doi:10.1109/GRID.2004.14.

[21] B. Francisco and M. Rodrigo, “The OurGrid Approach for Opportunistic Grid Computing,” Proc. First EELA-2 Conference, Feb. 2009.

[22] A. Goldchleger, F. Kon, A. Goldman, M. Finger, and G. Bezerra, “InteGrade: object-oriented Grid middleware leveraging the idle computing power of desktop machines,” Concurrency and Computation: Practice and Experience, vol. 16, March 2004, pp. 449-459, doi:10.1002/cpe.824.

[23] LONI Pipeline, "Laboratory of Neuro Imaging", [Online], http://pipeline.loni.ucla.edu.

[24] M. Sousa, A. Melo, and A. Boukerche, “An adaptive multi-policy grid service for biological sequence comparison,” Journal of parallel and distributed computing, vol. 70, Feb. 2010, pp. 160-172, doi:10.1016/j.jpdc.2009.02.009.

[25] S. Dowd, J. Zaragoza, J. Rodriguez, M. Oliver, and P. Payton, “Windows.NET network distributed basic local alignment search toolkit (W.ND-BLAST),” BMC Bioinformatics, vol. 6, April 2005, doi:10.1186/1471-2105-6-93.

[26] S. Pellicer, G. Chen, K. Chan, and Y. Pan, “Distributed Sequence Alignment Applications for the Public Computing Architecture,” IEEE Transactions on NanoBioscience, vol. 7, March 2008, pp. 35-43, doi:10.1109/TNB.2008.2000148.

[27] H. He, G. Fedak, B. Tang, and F. Cappello, “BLAST Application with Data-aware Desktop Grid Middleware,” Proc. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE, May 2009, pp. 284-291, doi:10.1109/CCGRID.2009.91.

[28] A. Vargas, et al., “Characterization of Phytophthora infestans Populations in Colombia: First Report of the A2 Mating Type,” Phytopathology, Sep. 2009, pp. 82-88, doi:10.1094/PHYTO-99-1-0082.

[29] S. Restrepo et al., “Computational Biology in Colombia,” PLOS Computational Biology, vol. 5, Oct. 2009, doi:10.1371/journal.pcbi.1000535.

[30] E. Afgan, and P. Bangalore, "Performance Characterization of BLAST for the Grid," Cluster Computing, vol. 13, Feb. 2010, pp. 385–395, doi:10.1007/s10586-010-0121-z.

[31] A. Shah, G. Folino, and N. Krasnogor, “Toward High-Throughput, Multicriteria Protein-Structure Comparison and Analysis,” IEEE Transactions on NanoBioscience, vol. 9, June 2010, pp. 144-155, doi:10.1109/TNB.2010.2043851.