Challenges and Opportunities of HealthGrids
V. Hernández et al. (Eds.)
IOS Press, 2006

Secured Distributed Service to manage Biological Data on EGEE grid

Christophe Blanchet [a,1], Rémi Mollon [a], and Gilbert Deléage [a]
[a] *Institut de Biologie et Chimie des Protéines (IBCP UMR 5086); CNRS; Univ. Lyon 1; IFR128 BioSciences Lyon-Gerland; 7, passage du Vercors, 69007 Lyon, France*

Abstract. Biological data are most times published and then become public ones. They, then, do not need to be isolated or encrypted. But, in some cases, these data stemed from patients or are analyzed with, for instance, pharmaceutical or agronomics goals. Also in simple ways , these data, before to become public, have to be kept confidential while researchers haven't been able to publish their work or to register them. So they are a lot of cases where the integrity and the confidentiality of biological data have to be protected against unauthorized accesses. But, as these private data are also large datasets, they need high-throughput computing and huge data storage to processed, such as ones produced by complete genome projects. These requirements are enhanced in the context of a Grid such EGEE, where the computing and storage resources are distributed across a large-scale platform. We have developed a secured distributed service to manage biological data on grid: the EncFile encrypted files management system. We have deployed it on the production platform of the EGEE grid project. Thus we provided grid users with a user-friendly component that doesn't require any user privileges. And we have integrated into a bioinformatics grid portal associated to encrypted representative biological resources: world-famous databases and programs.

Keywords. Biological data, grid computing, secured data, encryption

Introduction

Bioinformatics needs high-throughput computing and huge data storage to understand datasets such as ones produced by complete genome projects [1][2]. But these data are linked to patients, and used in scientific or industrial processes such as drug design and gene function identification. These use cases need to have a certain level of confidentiality and integrity to preserve the patient privacy or the patent secret. Obviously important in a local computing context such as supercomputer or cluster, these requirements are exacerbated in the context of a Grid such EGEE, where the computing and storage resources are distributed across a world-wide platform.

Biomedical applications are pilot ones in the EGEE project and manage a devoted virtual organization: the "biomed" VO. Thus, biomedical science has specific security requirements such as electronic certificate system, fine grain access to data, encrypted storage of data and anonymity. Certificate system provides biomedical entities (like users, services or Web portals) with a secure and individual electronic certificate for

[1] Corresponding author: Institut de Biologie et Chimie des Protéines (IBCP UMR 5086), 7 passage du Vercors, 69007 Lyon, France; Christophe.Blanchet@ibcp.fr

authentication and authorization management. One key quality of such a system is the capacity to renew and revoke these certificates across the whole grid. Biomedical applications also need fine grain access (with Access Control Lists, ACLs) to the data stored on the grid: biologists and biochemists can then, for example, share data with colleagues working on the same project in other places. This data also need to be gridified with a high level of confidentiality because they can concern patients or sensitive scientific/industrial experiments. The solution is then to encrypt the data on the Grid storage resources, but to provide authorized users with transparent and unencrypted access.

In this paper we describe the security requirements from biomedical applications about file encryption in the EGEE project. We give some examples of biological databases and applications for protein sequence analysis that are a representative model for encrypted file use cases. We present the model we have built for encrypted-file management, and the system we have deployed and are using today on the production platform of the EGEE grid, over the LCG2 middleware.

1. Biological data and protein sequence analysis applications

Biological data and bioinformatics programs have both special formats and behaviors, which are highlighted especially when they are used into a distributed computing platform such as grid [3].

Biological data represent very large datasets of different nature, from different sources, with heterogeneous models: protein three-dimensional structures, functional signatures, expression arrays, etc. They are currently more than 700 biomolecular available databases [4]. These worldwide databases are available on Internet through for example HTTP or FTP transfers. Moreover, biological data are not static, every day, new ones are published and existing ones may be updated, like with Swiss-Prot, a world-famous protein database [5]. Thus, biological databases need to be periodically updates, and classified as major or minor releases. But scientists using these databases need to access them exactly on the same way than before the last release [6]: under the same filename, under the same index in a database management system, …

Bioinformatics experiences use and produce very different and numerous methods and algorithms to analyze whole biological data, which are available to the community [7]. For each scientific domain of Bioinformatics, they are very often several different high-quality available programs for computing the same dataset in as many ways. Most bioinformatics programs are not adapted to distributed platforms. One of largest disadvantage is certainly that they are only accessing data with local file system interface to get the input data and to store their results, and that these data must be unencrypted to be read.

In Figure 1, a simple model, centered round the program, describe the kind of I/O of most bioinformatics programs, The processed data during the computation could be provided by the user or extracted from databanks such as these described in Table I. The input data can be (i) one or more parameters, (ii) user data, and (iii) databanks. The program always received input data through the standard command line and file system interfaces. The main differences between user data and databases are size of these data and their location on distributed platform. The user data are transferred at job submission from the user interface to the remote computing node. And databases are

available at public uniform resource locators (URL), which can be FTP, HTTP, or specific to the distributed platform

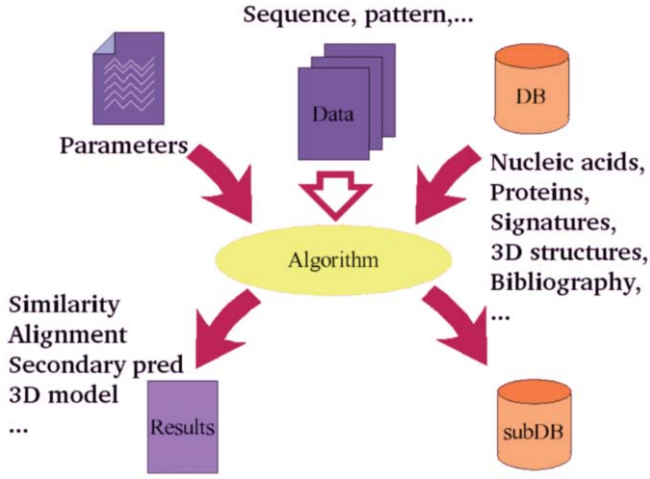

Figure 1. Algorithm-centered model of protein sequence analysis data and software. Simple execution model where user gives as input to the algorithm suitable kind of data and files, including large reference databases, and get the files containing the results of the biological analysis and/or subset of these input database.

Among the numerous set of available bioinformatics programs, we have chosen several ones for their representativeness: FastA [8], SSearch [9] and BLAST [10]. We have token them as models along this work, because of their special requirements for input or output of data. Indeed, they need to have access to large dataset as reference to compute their analyses, to protein sequences databases. They also produce large data sets as result of this computation, which can be a subset of the input protein sequences. This sub-database could then be pipelined to other bioinformatics program.

Table 1. Bioinformatics Resources Deployed on the Egee Grid

Resource	Grid Descriptor
Swiss-Prot	lfn://genomics_gpsa/db/swissprot/swissprot.fasta
And Blast indexes	lfn://genomics_gpsa/db/swissprot/swissprot.fasta.phr lfn://genomics_gpsa/db/swissprot/swissprot.fasta.pin lfn://genomics_gpsa/db/swissprot/swissprot.fasta.psq
TrEMBL	lfn://genomics_gpsa/db/trembl/trembl.fasta
PROSITE	lfn://genomics_gpsa/db/prosite/prosite.dat lfn://genomics_gpsa/db/prosite/prosite.doc
ClustalW	ESM tag “genomics_gpsa_clustalw”
SSearch	ESM tag “genomics_gpsa_ssearch”

Examples of biological databases and bioinformatics programs we have registered and deployed onto the EGEE grid. Database files have been encrypted and registered as logical files into the replica manager system, with their own logical filename (LFN, lfn://), and programs with an tag of the experiment software manager (ESM tag)

2. Grid computing

Grid computing concept defines a set of information resources (computers, databases, networks, instruments, etc.) that are integrated to provide users with tools and applications that treat those resources as components within a « virtual » system [19][20][21]. A grid middleware provides the underlying mechanisms necessary to create such systems, including authentication and authorization, resource discovery, network connections, and other kind of components.

2.1. The European EGEE grid

The Enabling Grids for E-sciencE project (EGEE [11]), funded by the European Commission, aimed to build on recent advances in grid technology and to develop a service grid infrastructure. The EGEE consortium involves 70 leading institutions in 27 countries, federated in regional Grids, with currently a combined capacity of 17,000 CPUs and 5 petabytes of storage. The platform is built on the LCG-2 middleware, inherited from the EDG middleware developed by the European DataGrid Project (EDG, FP5 2001-2003). The middleware LCG-2 is based upon the Globus toolkit release 2 (GT2) and the Condor middleware. A new middleware gLite [11], is being developed to improve the performances and the services provided by the future EGEE platform.

The EGEE middleware provides grid users with an “user interface” (UI) to launch a job. The main commands to run a job are: job submission (edg-job-submit), progress report (edg-job-status) and results download (edg-job-get-output). There are several important components into the EGEE grid: the “workload management system” (WMS) responsible of the jobs scheduling on the platform. The scheduler (or “resource broker”) determines where and when to compute a job: (i) using one “computing element” (CE) near one “storage element” (SE) containing the data in case of simple jobs, or (ii) several CEs an SEs in case of larger jobs. A computing element is a gatekeeper to a cluster of several CPUs, the worker nodes (WN) managed by a batch scheduler system. The “information system” that centralize all parameters raised by the grid components (CPUs, storage, network,…). A other key-component of a grid is the “data management system” (DMS), especially when you plan to run bioinformatics applications.

2.2. Distributed storage on the EGEE grid

The “data management system” (DMS) on the EGEE grid is a key service for our bioinformatics applications. Having efficient usage of it will be synonymous of good distribution of our protein sequence analysis applications. The main component involved into this mechanism is the storage element, which stores the file on different kind of medium: disk or tape. These data may have metadata attached to, with access restricted to the system or allowed to the application/user. Following this design, the data manager in EGEE has different functionalities: providing user with command for data registration and replication, and sharing data information with the WMS for scheduling job near the given data.

On the EGEE grid, the replica manager system provide user with data management functionalities such as: data registration (lcg-cr), data replication (lcg-rep) and data suppression (lcg-del). All these commands are available through command line

interface (CLI) or through application programming interface (API). The data are stored as file on the storage elements. These files are identified in a logical namespace with GUID (Grid Unique IDentifier) and LFN (Logical File Name) that are pointing to local occurrences on the SE: the SFN (Storage File Name), absolute pathway on the given storage element.

Despite there are tools to get the SFN from LFN or GUID, these are no automatic substitution mechanisms on the application command line. A legacy bioinformatics application launched on the EGEE grid won't be able to access the remote data stored on SEs, except if we download it on the given worker node before to execute the program.

3. Biomedical requirements about file encryption

Biomedical applications are pilot in the EGEE project and have strong requirements about data security, integrity, authentication and authorization. One key point is encrypted storage to prevent from storage system failure or cracking, and malicious system administrator. Indeed, nowadays, nobody is able to affirm that his computer, or his computer network, is totally secure. Moreover, in a context as distributed as the grid one, a lot of system administrators take part of the grid management. So, it would be a too strong assumption to trust all of them.

But encrypting needs cryptographic keys, and these ones have also to be stored somewhere. And if they would be in a centralized key server, the storage element related problems would only be moved to this server. Moreover, end users do not be able to get the cryptographic keys, even if they are authorized to access the corresponding file.

Another requirement is to use the bioinformatics programs as "black boxes", what means that their source code can't be modified. The main reasons for this are they are too numerous, and their source code is very often not available. The consequence is that it's the encrypted file management system that has to be adapted to applications, and not the opposite. In other words, accesses to encrypted gridified files must be completely transparent for biomedical applications. And then, users don't need keys. So to grant them access to such information is useless, and could endanger the system if one of them would distribute these keys to others – intentionally or not.

An acceptable solution would obviously be to store files in a encrypted format, to not reveal corresponding keys to users, and to not centralized them in an unique server, which would be vulnerable to a system failure or compromising. The last requirement is that it must be able to work with the current production platform of EGEE project: the one using LCG-2 middleware.

One of the EGEE workgroups is developing an encrypted file management system, partially fulfilling biomedical requirements. Indeed, gridified files accesses - encrypted or not – aren't transparent for applications: they have to use an API. Moreover, cryptographic keys are currently stored in a single key-store, which is so a single point of failure of the system, and a very interesting target for an attacker. Finally, this system is available on the pre-production platform of the EGEE project, the one built upon the gLite middleware, and not on the production platform. Its advantage is that it inherits the ACL support of the gLite middleware, a functionality that is not available on the production platform for the moment.

4. EncFile, securing biological data on Grid

The Grid context is very distributed, and so data security is a tough problem. Files are stored on specific component, the storage elements, but also on worker nodes for computation, and system administrators can read them at this moment.

To deal with such a problem, we encrypt sensitive biological files on the EGEE storage elements. We use the AES algorithm (Advanced Encryption Standard, FIPS 197) with keys of 256 bits. Indeed, AES has good security properties and performances: decrypting a 200 MB files takes 22 seconds on a computer with two PentiumIII 1.0 GHz and 1GB RAM. Each file is encrypted with one different key stored into a key service (see Figure 2).

Grid services are distributed, and the main reason is to bring fault tolerance properties to the platform. Another reason is to have only limited trust on system administrators. Thus, we have also applied this distribution principle and have used, to store keys, M-of-N technique from Shamir secret sharing algorithm [14]. We split a key into N shares, each stored in a different server. To rebuild a key, exactly M of the N shares are needed. With less than M shares, it is impossible to deduce several bits or even one of them. In this way, we can consider that the key servers don't need to be secured more than one installed with good practices. Indeed, an attacker need to compromise at least M servers in order to be able to reconstruct the key, and then decrypt the encrypted data.

The "EncFile" system is composed of the N key servers and one client (see Figure 1). Each key server is a PostGreSQL server storing one of the N shares of all the keys in its tables. EncFile client is doing the decryption of the file, and is the only component able to rebuild the keys and guarantee their confidentiality. It must be almost impossible for users to retrieve them.

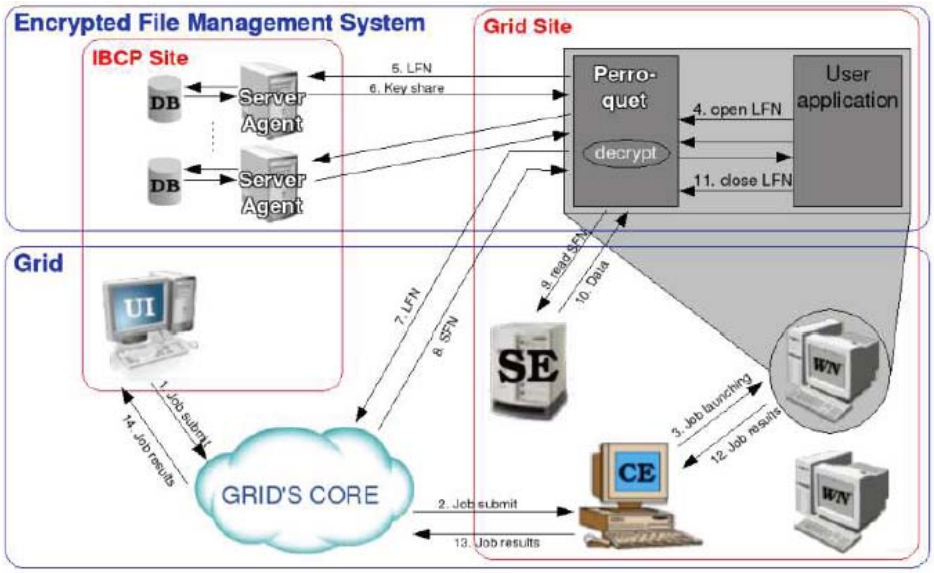

Figure 2. Integration of the encrypted file management system in the EGEE platform (UI: user interface, CE: computing element, SE: storage element, WN: worker node, DB: database, LFN: logical file name).

The transfer of the keys between the servers and the client is very important. We secure them by using the OpenSSL library with encryption and mutual authentication. In order to determine user authorization, the client uses his grid certificate to authenticate himself. Nonetheless, to avoid that a malicious person creates a fake EncFile client (e.g. to retrieve keys), a double authentication is done, once with the specific certificate of the EncFile client, and once with the user one.

It's important to note that this system enforces data confidentiality and integrity, but it doesn't protect files from a malicious attacker who would success to erase them. For that, it's necessary to implement a replication mechanism, which ensures that files are always replicated on a minimum number of storage elements.

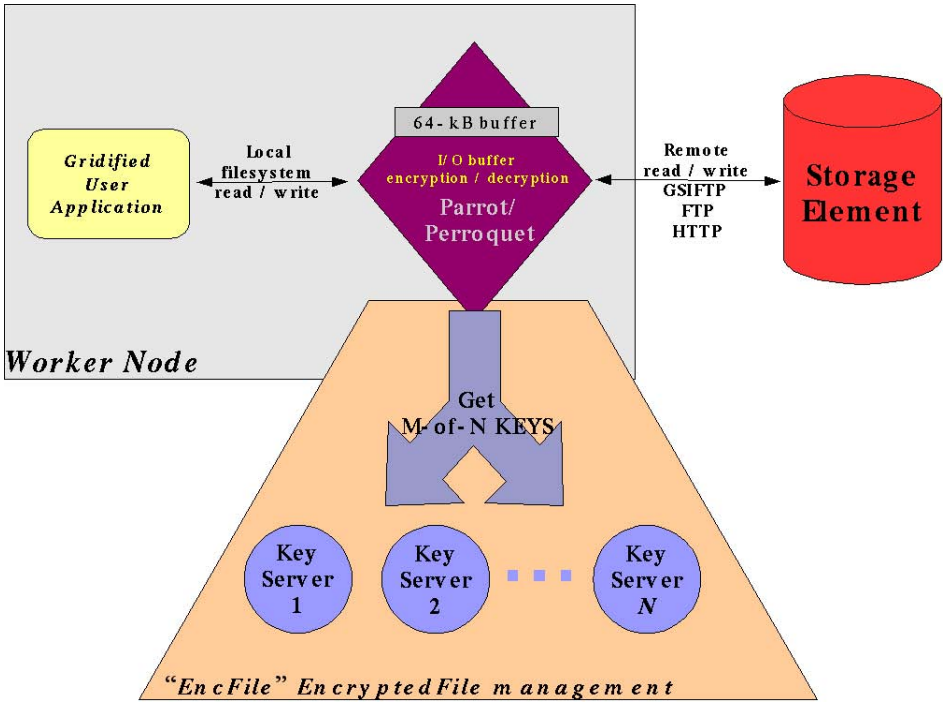

Figure 3. Architecture of the EncFile client integrated into Perroquet. It is catching the local I/O and forwarding them through network to remote storage element. Several transfer protocols are available such as FTP, HTTP, and especially GSI-FTP which is authenticated, crypted and fully compatible with the EGEE grid middleware, component and services. Perroquet recognizes EGEE LFNs , like lfn://genomics_gpsa/db/Swissprot.fasta, and is encrypting and decrypting data "on the fly"

As explained previously, most of the bioinformatics programs are only able to access their data to local file system interface, and also to work only on plain data. To answer to these 2 strong issues, we have combined the EncFile client and the Parrot software [15]. This EncFile client (called Perroquet in Figure 2), which acts as a launcher for applications, catches all their standard input/output calls, and replaces them with equivalent remote calls to handle several remote file protocols. Perroquet, is based on Parrot for catching the I/O calls, but we have modified it to made it compliant with the EGEE production platform and to be able to work with the logical file name (see Table 1) of our biological resources onto the EGEE grid. Currently the supported protocols are: http, ftp, gsiftp, lfn. Moreover, to encrypt and decrypt files, we have

integrated an EncFile client in it. This enables on-the-fly decryption, and so decrypted file copies aren't needed. This has mainly two consequences: (i) higher security level because decrypted file copies could endanger data, (ii) better performances because files aren't read twice to locally copy and to decrypt.

Thus, the EncFile client permits any applications to transparently read and write encrypted, or not, remote files as if they were local and plaintext. The EncFile system is used to secure sensitive user data on the EGEE resources to be used by the selected protein sequence analysis programs FastA, SSearch and BLAST. Programs, which run on the grid worker nodes, can transparently store files into the replica management system of EGEE (RMS) with EncFile, and access them, as well as other remote files (see Figure 3).

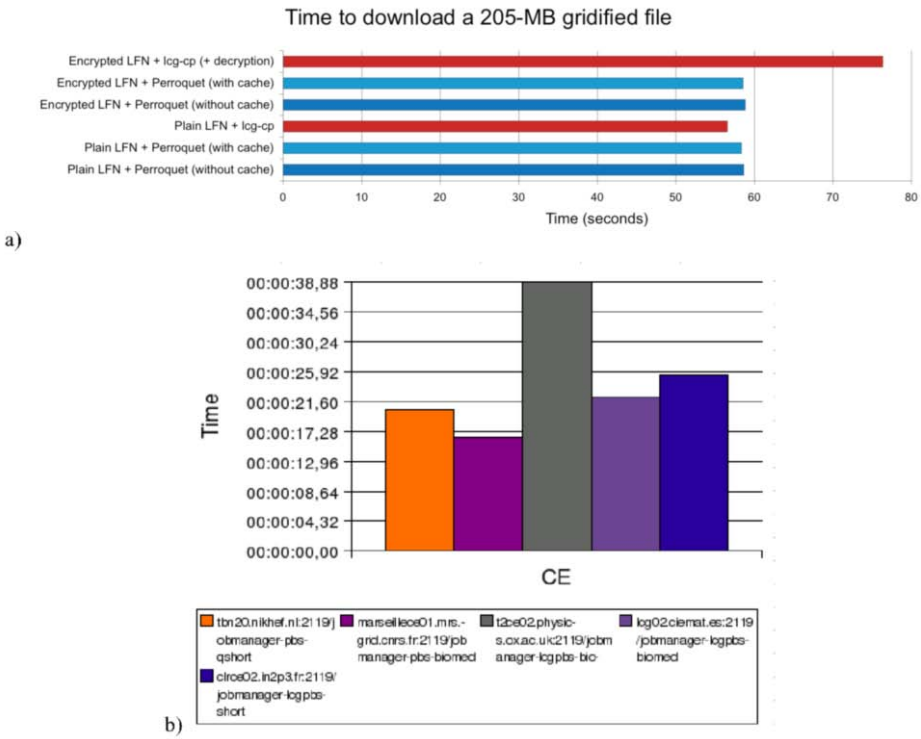

Figure 4. Download time on the EGEE platform of a 500,000 sequences file (around 200 Mbytes). We have compared the use of the lcg-cp and the perroquet programs, with plain and encrypted file. In a), The file is downloaded from a remote storage element (not the near-SE) to a given worker node. In b), the file is download from the nearSE of the given worker node, and decrypt on-the-fly

We have done some benchmarks with the selected biological data and protein sequence algorithms to study the effect of the cryptographic mechanism on file access. The performances of EncFile system were quite good (see Figure 4). Indeed, despite the cryptographic overhead – the cost for manipulating encrypted data (decryption overhead) was estimated over large files (200 MB) – the jobs using EncFile client to access encrypted files are faster than the same jobs but with middleware commands and plain-text files. And, when we use Perroquet, time difference between jobs with encrypted files, and the ones with plain-text files is very close to zero. Comparing a) and b) in Figure 4 shows that the ratio of time to get and decrypt 200MB file is of 3 if

the file is get from any storage element than from the near storage element. Enhancing the importance of an efficient replication mechanism of the encrypted biological data that will be used by bioinformatics programs on the EGEE grid.

5. Integration of secured biological data into a bioinformatics grid portal

GPS@ grid portal (Grid Protein Sequence Analysis, http://gpsa.ibcp.fr) is a bioinformatics integrated portal devoted to protein sequence analysis on the EGEE grid [1][2]. This grid portal is the grid port of the Network Protein Sequence Analysis (NPS@, [7]). GPS@ portal provides the biologist with a user-friendly interface for the EGEE grid resources, computing and storage.

GPS@ acts as a high-level grid user interface hiding to biologist the mechanism of the EGEE workload management system. All steps involved for the execution of bioinformatics jobs on the grid infrastructure are numerous and complex. Grid user can submit a job through command line interface on a EGEE user interface. We have integrated all the calls to the grid middleware getting status of the submitted job. GPS@ user has only to submit. The bioinformatics algorithms and biological databases have been distributed and registered on the EGEE grid and GPS@ runs its own EGEE interface to the grid.

The given biological databases have been distributed on EGEE with the lcg commands. lcg-cr to register the database file and to copy it on the grid. Despite the replica management system have no file tree, we have decided to create one by given logical filename according to the following model *lfn://genomics_gpsa/db/<dbname>/<dbfiles>* (see Table 1). These LFNs have been replicate with the lcg-rep commands on several sites without applying any particular model of replication.

Bioinformatics programs have also been integrated to the grid services (see Table 2) with the same goals, but in a slightly different way. We have used the experiment software management service (ESM) that permit to distribute package on remote sites of the grid with the agreement of the site administrator. We have also define a namespace to distinguish our bioinformatics applications from the other field ones. Our ESM tags follow this model: *genomics_gpsa_<appname>*.

Table 2. Example of Bioinformatics Algorithms Gridified into the GPS@ Portal

Algorithm	Class	Input databank	Gridified
BLAST	Similarity	Sequence	EGEE
FASTA	Similarity	Sequence	EGEE
SSearch	Similarity	Sequence	EGEE
ClustalW	MSA	no	EGEE
Multalin	MSA	no	EGEE
PattInProt	Pattern/Profile	Sequence, Pattern, profile	EGEE/DIET
Pfsearch/ pfscan	Profiles	Sequence, Pattern, profile	EGEE
GOR4	PSSP	no	EGEE

Algorithm	Class	Input databank	Gridified
SIMPA96	PSSP	no	EGEE
SOPMA	PSSP	no	EGEE

(PSSP: Protein secondary structure prediction; MSA: Multiple Sequence Alignment)

Both the deployed LFNs and ESM tags play a key-role into the scheduling of our future jobs. Indeed, the jobs submitted, with LFN and/or ESM tag used into the job description file, will be send according to a match-making between these symbols and the free sites hosting one physical replicate of them.

All the described biological databases, even though they are public ones and so not confidential, have been gridified on the EGEE grid in an encrypted way for the purpose of the demonstration. Then we have applied on these encrypted biological databases all the gridified algorithms available on the GPS@ portal (see Table 2). All these world-famous legacy programs, such as BLAST, ClustalW or pfsearch, are only able of local file access and on plaintext, non-encrypted, files. With the help of the EncFile system, the perroquet client has provided these bioinformatics job with remote and encrypted data, as if they were local and unencrypted ones.

6. Conclusion

We have deployed several representative biological data in encrypted state onto the worldwide EGEE grid. These resources have been registered with canonical logical names, and thus are available to all biologists and bioinformaticians participating to the EGEE "biomed" virtual organization. We have developed the EncFile encrypted files management system, and deployed it on the production platform of the EGEE project. Thus we provided grid users with an user-friendly component that doesn't require any user privileges. To demonstrate, we have integrated this secured data service into the GPS@ Web portal on the EGEE production platform. Users of the Web portal have been able to submit standard bioinformatics analyses with world-famous legacy applications on encrypted biological data in a transparent way.

Future works will be done on the adaptation of our EncFile system to other distributed system as EncFile is not linked to the EGEE grid components. This will bring access to other biological resources stored into different storage system, and to others bioinformatics methods integrated into other distributed platform.

7. Acknowledgement

This works was partially supported by the European Union (EGEE project, ref. INFSO-508833). Authors express thanks to Douglas Thain for the interesting discussions about the Parrot tool.

References

[1] Jacq, N., Blanchet, C., Combet, C., Cornillot, E., Duret, L., Kurata, K., Nakamura, H., Silvestre, T., Breton, V. : Grid as a bioinformatics tool. , Parallel Computing, special issue: High-performance parallel bio-computing, Vol. 30, (2004).

[2] Breton, V., Blanchet, C., Legré, Y., Maigne, L. and Montagnat, J. : Grid Technology for Biomedical Applications. M. Daydé et al. (Eds.): VECPAR 2004, Lecture Notes in Computer Science 3402, pp. 204–218, 2005.

[3] Desprez, F., Vernois, A., Blanchet, C.: Simultaneous Scheduling of Replication and Computation for Bioinformatic Applications on the Grid. ISBMDA 2005, Lecture Notes in Computer Science 3745: 262-273

[4] Galperin, M.Y.: The Molecular Biology Database Collection: 2005 update. Nucleic Acids Research 33 (2005) National Center for Biotechnology Information and National Library of Medicine and National Institutes of Health.

[5] Bairoch, A, Apweiler, R : The SWISS–PROT protein sequence data bank and its supplement TrEMBL in 1999. Nucleic Acids Res. 27 (1999) 49-54

[6] Perriere, G, Combet, C, Penel, S, Blanchet, C, Thioulouse, J, Geourjon, C, Grassot, J, Charavay, C, Gouy, M, Duret, L, Deleage, G..: Integrated databanks access and sequence/structure analysis services at the PBIL. Nucleic Acids Res. 31, (2003) 3393-9.

[7] Combet, C., Blanchet, C., Geourjon, C. et Deléage, G. : NPS@: Network Protein Sequence Analysis. Tibs, 25 (2000) 147-150.

[8] Pearson W.R. ; Searching protein sequence libraries: comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. PNAS (1988) 85:2444-2448

[9] Smith T.F., Waterman M.S. : Identification of common molecular subsequences. J. Mol. Biol. (1981) 147:195-197

[10] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J., Basic local alignment search tool. J. Mol. Biol. 215 (1990) 403–410.

[11] Enabling Grid for E-sciencE (EGEE) www.eu-egee.org

[12] Foster, I. And Kesselman, C. (eds.) : The Grid 2 : Blueprint for a New Computing Infrastructure, (2004).

[13] Vicat-Blanc Primet, P., d'Anfray, P., Blanchet, C., Chanussot, F. : e-Toile : High Performance Grid Middleware. Proceedings of Cluster'2003 (2003).

[14] Shamir., A. "How to share a secret". Communications of the ACM , 22(11):612–613, Nov. 1979.

[15] Thain, D. and Livny, M.: Parrot: an application environment for data-intensive computing. Scalable Computing: Practice and Experience 6 (2005) 9-18

[16] Desmedt Y. and Jajodia S.. "Redistributing secret shares to new access structures and its applications". Technical Report ISSE TR-97-01, George Mason University, Fairfax, VA, July 1997.