FROM GRIDS TO CLOUD
The Pathway for Brain dMRI Cloud Services

Tarik Zakaria Benmerar and Fatima Oulebsir-Boumghar
ParIMed Team, LRPE, USTHB, BP32, El Alia, Bab-Ezzouar, Algiers, Algeria

Keywords: Cloud Computing, Grid Computing, dMRI Brain, Cloud Services, SaaS, PaaS.

Abstract: In this paper, we present the actual architecture of Acigna-G, our Cloud-oriented Grid Computing platform and the ongoing deployment of a MAS algorithm for brain segmentation. Also, we discuss three important improvements for this platform to allow the deployment of brain dMRI cloud services : HTTP/Restful oriented computing services for the management of user's service requests, application-level virtualization coupled with distributed computing models, and separation of user request management and computing tasks execution as found on actual PaaS Cloud Services. Such architecture would offer a convenient deployment and use of brain dMRI PaaS/SaaS Cloud Services onto a computing grid.

1 INTRODUCTION

For more than a decade, Grid Computing has fueled the scientific research by offering an access to a huge number of federated computing ressources of different organizations. Many experiments have been undertaken, from oil industry to finance while passing by Particles Physics and Medical Imagery, more particularly, in neuro-imagery and dMRI images processing and analysis.

Neverthless, in such infrastructure the complexity of deployment and development of grid applications remain a present hurdle, and as pointed by Gabrielle Allen *et al.* there exists a shortage of *real grid users* (Allen et al., 2003). Also, as mentioned by Christian Vecchiola *et al.* (Vecchiola et al., 2009), some issues are bureaucratic : as grids are shared worldwide, a proposal should be submit by research groups describing the type of research carried out. This approach has lead to a competitive use of scientific grids, and minor research projects couldn't get access to them.

Cloud Computing has permitted to avoid such issues, with virtualized computing ressources offered as an IaaS Cloud Service. Small research groups can harness these resources, and pay only for usage. In such approach, these groups manage also the underlying computing infrastructure, and not only the design of computing applications.

To avoid infrastructure management, PaaS Cloud Services can be used for application deployment only. They offer application-level services that harness distributed computing common models like Map/Reduce, without any prior expert knowledge. Unfortunately, actually no PaaS cloud service offers the same features for high performance computing, more particularly for brain dMRI processing and analysis.

Acigna-G project is an ongoing effort to offer a convenient deployment and use of brain dMRI PaaS/SaaS Cloud Services onto a computing grid. In this paper, we present the actual architecture of this Cloud-oriented Grid Computing platform and the ongoing deployment of a MAS algorithm for brain segmentation. Also, we discuss three important improvements for this platform to allow the deployment of brain dMRI Cloud Services : HTTP/Restful oriented computing services for the management of user's service requests, application-level virtualization coupled with distributed computing models, and separation of user request management and computing tasks execution as found on actual PaaS Cloud Services.

The remaining parts of the paper are organised as follows : In section 2 we present brain dMRI softwares and features needed for a cloud service. In Section 3, we discuss computing grids architectures, and in Section 4, PaaS Cloud Services architecture. In Section 5, we present the actual Acigna-G architecture and the ongoing deployment of a MAS algorithm for brain segmentation. In Section 6, we discuss the improvements to achieve for the platform. Section 7 concludes the paper.

Benmerar T. and Oulebsir-Boumghar F..
FROM GRIDS TO CLOUD - The Pathway for Brain dMRI Cloud Services.
DOI: 10.5220/0003960401410146
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 141-146
ISBN: 978-989-8565-05-1

2 BRAIN dMRI CLOUD SOFTWARES AND SERVICES

2.1 Introduction

In this section, we introduce the actual Brain dMRI Softwares. After that, we discuss the important features that are needed for a Cloud Service for Brain dMRI.

2.2 Actual Brain dMRI Softwares

Medical Imagery softwares such as MedInria (MedInria, 2012) and FSL (FSL, 2012) propose different kinds of dMRI processing and visualization like diffusion tensor field estimation and visualization, and white fibers tractography. We should note that tools such as FSL provide a low level command lines to launch certain processings. This allows the deployment of grid tasks using FSL, and commands such as fsl_sub provide a parallelization for these processings.

Providing a Cloud Service for such processing is important for broad access, and for providing a flexible software upgrades.

2.3 A Brain dMRI Cloud Service

A Brain dMRI Cloud Service should deliver the same features provided by the mentionned softwares, but accessible using a web browser to provide a convenient and broad access. In fact, three types of users will interact with such cloud : Researchers will provide the state of the art optimised and parallelized algorithms, developers will provide the back-end PaaS applications and front-end SaaS applications, and finally the clinicians who will authenticate and use the provided SaaS for medical diagnosis.

We should note that the client web browser interacts with the cloud through a Restful/HTTP interface. Also, all the back-end infrastructure is composed of grid ressources that span different organisations.

3 TASK-BASED AND SERVICE-BASED GRID ARCHITECTURE

3.1 Introduction

In the grid computing world, two strategies exist for the infrastructure deployment : Task based strategy and service based strategy (Glatard et al., 2008). In this section, we present both strategies and the relevant pros and cons for latter comparisons.

3.2 Task-based Architecture

In the *task based* strategy or *global computing*, computing tasks to be executed are defined by the user by specifying the executable code file, the input data files, and the command line parameters to invoke the execution. This strategy is known for using batch systems for task execution. The best known Grid Middleware adopting this strategy is Globus (Foster, 2006).

Although these systems allow the deployment of a rich set of grid applications, they are known for being highly centralized, and alternative P2P Grid platforms such as JaceP2P (VUILLEMIN, 2008) have been proposed for problems having interdependant computing units. Also, prior knowledge of the Grid API is mandatory.

3.3 Service-based Architecture

In the *service based strategy* or *meta computing*, the application codes are wrapped into standard interfaces, and only the invocation interface is known. DIET (Amar et al., 2008) adopts this strategy, and enables the deploment of NES (Network Enabled Server System) (Matsuoka et al., 2000), using an RPC-style (Remote Procedure Call). It offers access to a set of servers offering specific computing services, for a specific problem set, using a web browser, a compiled program or a PSE (Problem Solving Environment) like Matlab or Scilab.

This strategy is a simpler alternative, as input parameters are the only mendatory elements needed for service invocation. Neverthless, the client is constrained to available services, and custom grid application deployment is impossible. Also, the user must be familiar with Grid API for service interaction using a custom code. Furthermore, for several dependant service invocations, data are sent back and forth, resulting to unnecessary communications. Recent researches have proposed solutions such as data persistence and redistribution to solve this issue (Desprez and Jeannot, 2004).

4 PaaS CLOUD SERVICES ARCHITECTURE

4.1 Introduction

Microsoft Windows Azure (Chappell, 2012) and Google App Engine (Google, 2012) are among the

earliest and the major PaaS Cloud Providers. We review in this section some archicture points that can be used for the construction of a PaaS Cloud Service for a brain dMRI.

4.2 Application-level Virtualization

Actually, PaaS applications are run in a virtualized sandboxed environment, for a secure execution. The sandboxing is ensured by the VM executing the application. It restricts the application execution to a limited set of standard libraries and PaaS application-level services (datastore, mail, memcache etc.).

Neverthless, as this approach restricts access to the underying OS, binary programs written in languages such as C/C++ or Fortran are not supported.

4.3 Common Distributed Computing Model

Today, Distributed Computing Models such as Map Reduce (Dean and Ghemawat, 2004) or Memcache (Dormando, 2012) allow us building scalable web applications. Web giants such as Facebook and Google actually use them for their web portals and applications, and have opensourced some of them for broad use.

From a PaaS standpoint, these models are provided as application-level services. No detailed implementation knowledge is required, and access is done in a convenient way.

4.4 Separation of User Request Management and Background Tasks Management

Some user requests to a PaaS web application cannot be handled in a timely fashion. These requests require intensive tasks to be executed.

To solve such issue, PaaS platforms provide a way to place a task in a queue, and be executed in the background. In this case, the web application recieves the request and delegates it to a background task, so that it can deliver a response in a short time.

5 Acigna-G ARCHITECTURE AS A SOLUTION

5.1 Introduction

Previous works on Acigna-G (Benmerar and Oulebsir-Boumghar, 2011) introduced an application-level virtualization layer inspired by actual PaaS platforms, and a particular architecture named multi-level services architecture. The initial objectif was to provide an experimental cloud service for the submission and use of grid applications without prior expertise of grid protocols.

Three levels of hierarchy exist represented by the Master Server, the local server, and the compute ressource. The master server manages all the grid ressources and tasks, the local server manages ressources and tasks at a given site. The compute ressource manages the execution of tasks at its level. A given application is decomposed into application instances or tasks, each one executing different codes, sandboxed using Plash (Seabon, 2012), and interacting with each other through the virtualization layer. For a convenient interaction with tasks, we provide a virtual web terminal as illustrated in Figure 1.

Figure 1: Virtual web terminal (Benmerar and Oulebsir-Boumghar, 2011).

In the remaining parts of this section, we introduce the multi-level services architecture, and we present the ongoing deployment of a MAS algorithm for brain segmentation.

5.2 Multi-level Services Architecture

Multi-level Services Architecture is the main architecture contribution of Acigna-G. The Multi-level Services (see Figure 2) architecture implements web services which are at different hierarchical levels represented by the Compute Ressources, the Local Servers and the Master Server. Each service can call services implemented on nodes (i.e Master Server, Local Servers or Compute Ressources) located at the same hierarchical level with the same parent, on the parent node or on the child nodes.

This architecture allows a flexible services calls propagation with interresting consequences : Firstly, some service calls originated from applications can avoid upward propagation of calls, to decrease loads on Local and Master Server. Secondly, the possibil-

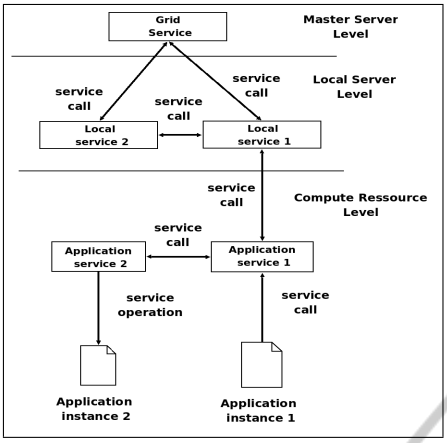

Figure 2: Acigna-G Multi-level Services architecture.

ity of P2P inter-nodes communication allows an efficient deployment of applications with interdependant computing units. Finally, this architecture is a flexible way to interpret application instance service calls into an interaction with the grid infrastructure.

5.3 Deploying a MAS Algorithm for Brain Segmentation

Haroun *et al.* have proposed a new hybrid segmentation algorithm (Haroun et al., 2005). It is a MAS algorithm combining different methods such as K-Means, FCM and contextual methods such as Region growing and produces better results than non-hybrid ones. A Python implementation of this algorithm (Laguel, 2010) using the SPADE environment (Spade, 2012) have been deployed onto the BrainVisa platform (BrainVISA, 2012).

Currently, there is an ongoing deployment of this algorithm onto Acigna-G. Before that, we are currently adapting the python application to the platform by following two main steps (see Figure 3) :

1. Creating pseudo agents and many spade servers. The communication between pseudo agents is done with socket.
2. Using socketd application service for pseudo agents communication. Socketd allows socket use for communication between two instances even if executed in different compute ressources. It gives an illusion to both of them that they are on the same ressource.

To conclude this section, we should note that additional efforts are required to integrate any existing platform such as Spade to a virtualized multi tenant environment, as they have been originally designed for non-virtualized single tenant one.

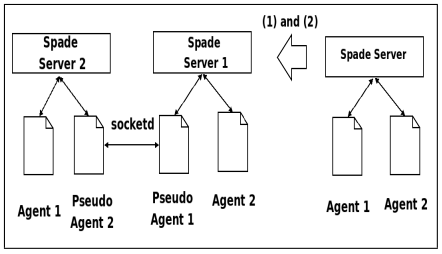

Figure 3: Adapting the MAS algorithm to Acigna-G.

6 Acigna-G IMPROVEMENTS TO ACHIEVE

6.1 Introduction

In the present section, we discus the future improvements of our Acigna-G platform, towards building a SaaS/PaaS Brain dMRI Cloud Service. We discuss three important improvements : HTTP/Restful oriented computing services for the management of user's service requests, application-level virtualization coupled with distributed computing models, and separation of user request management and computing tasks execution as found on actual PaaS Cloud Services.

We also discuss the use of a convenient programming language for the development of PaaS applications, and describe briefly the integration of workflow management in Acigna-G.

6.2 Managing User Service Requests

We have seen that PaaS platforms separate background tasks management from user requests management. By analogy and from a grid computing standpoint, task management would refer to the grid application execution management. HTTP/Restful user request management actually doesn't exist in a grid environment.

To improve our platform, we propose adding HTTP/Restful cloud services support, separated from the task management. These cloud services would allow external non-grid users to be authenticated and invoke certain computing services. Hence, we have two type of users : Grid users, that have access to computing ressources and service users, that have only access to the different HTTP/Restful services. The user authentication is ensured at the Restful/HTTP Master Service level.

As for master, local and application computing services found in our Multi-level Services Architecture, there would exist a whole hierarchy of Rest-

ful/HTTP services. For any invocation requiring a task management, it should be delegated to a computing service as illustrated in Figure 4.

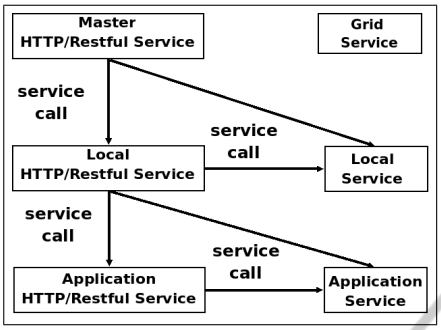

Figure 4: HTTP/Restful Services.

In the context of a brain dMRI cloud service, non-grid users are doctors that can submit a dMRI image to a Restful Service using an external web interface. The Restful Service would submit a new task to the grid by invoking a grid-level service.

6.3 Application-level Virtualization and Common Parallel/Distributed Computing Models

We have observed that PaaS common distributed models, conveniently accessible as application-level services permit users the built of scalable web applications without detailed implementation knowledge. The actual multi-level services architecture allows a virtualized interaction between different application instances, but it still gives a low-level control of the application to the grid user.

Keeping the same architecture but making the virtualized environment more abstract, we envision some application services as Parallel/Distributed computing services that can be invoked from the application. From the grid computing standpoint, this can be seen as equivalent to a service based strategy where both server and client codes are on the same infrastructure. The latter point allows the resolution of dependant service invocation issue we have seen with such strategy.

If we restrict the application to use only these services, the load on a given node is due to services code execution rather than applications code execution, leading to a better management of the infrastructure. For example, depending on the available nodes and their computing loads, a given service can be executed on a different node, or be split into a different number tasks, to load balance the multiple nodes. As the execution detail is left to the service, it can be optimised for nodes containing GPUs (*Graphic Processing Units*) or CPUs (*Central Processing Units*), depending on the task.

In the brain dMRI cloud service context, these computing services can be seen as common processing tasks such as Diffusion Tensor Field estimation, parallelized and optimized for a grid execution.

6.4 Workflow Management

As part of the future improvments, a workflow can be invoked at the application service level. This will help implementing complexe services constitued from smaller unit tasks. Where each one, can be a service invocation or an executable task.

In the common parallel/distributed models context, some models can be implemented as workflows for an even better conveniency at the application service level.

6.5 Convenient Programming Languages

Use of common parallel/distributed models for application-level services in this new architecture, has led us to propose the support of specific VM powered programming languages, as the real computing load is on the service execution, rather than on the application itself. Hence, the user has a high-level control on the application execution, and a convenient programming would make the use of application-level services easier. For such purpose, we have choosen the Python programming language, for different reasons (Caia et al., 2005) :

- Python is a very convenient programming. MIT school has proposed it in an introductory course to programming.
- Python can be optimised for scientific computing and parallel programming, thanks to modules such as numpy.
- In fact, for maximum optimisation, specific parts that are commonly used in scientific computing are implemented and compiled in C and can be called from python, thanks to the C-API interface.

7 CONCLUSIONS

Additional efforts and improvements are needed to adapt grids for providing PaaS/SaaS brain dMRI Cloud Services. This is reflected in our experimental and ongoing Acigna-G project, and as a first step

towards such goal the introduction of the Multi-level Services Architecture, that was proposed in a previous work is necessary, as it provides for the applications, a more convenient access to their grid environment, in a virtualized manner. Actually, there is an ongoing deployment of a MAS algorithm implemented in Python/Spade, and we can argue that one of the difficulties of such paradigm shift is that existing platforms such as Spade need additional efforts to adapt them to a virtualized multi tenant environment.

Deep analysis of the current PaaS platforms architectures gave us interesting insights for the future improvements of the platform to achieve our objectif. Firstly, a whole new hierarchy of HTTP/Restful services will be built, that are equivalent in architecture to the existing Acigna-G computing services, but for non-grid users service requests management. It will allow to such users a convenient access to different computing services without direct grid ressources access. And secondly, we have proposed the integration of parallel/distributed models such as optimized and parallelized Diffusion Tensor Field estimation for the platform. With such an architectural point introduced, we'll avoid the existing platform adaptation issue, and have a better infrastructure management as the real computing load is on the service execution.

REFERENCES

Allen, G., Davis, K., Dolkas, K. N., Doulamis, N. D., Goodale, T., Kielmann, T., Merzky, A., Nabrzyski, J., Pukacki, J., Radke, T., Russell, M., Seidel, E., Shalf, J., and Taylor, I. (2003). Enabling applications on the grid a gridlab overview.

Amar, A., Bolze, R., Boix, E., Caniou, Y., Caron, E., Chouhan, P. K., Combes, P., Dahan, S., Daila, H., Delfabro, B., Frauenkron, P., Hoesch, G., Isnard, B., Jan, M., L'Excellent, J.-Y., Mahec, G. L., Christophe, P., Cyrille, P., Alan, S., Cédric, T., and Antoine, V. (2008). Diet user's manual. inria, ens-lyon, ucbl. Retrieved January 27, 2012. http://graal.ens-lyon.fr/DIET/download/doc/UsersManualDiet2.4.pdf.

Benmerar, T. Z. and Oulebsir-Boumghar, F. (2011). Toward a cloud architecture for medical grid applications : The acigna-g project. In *Proceedings of the 10st International Symposium on Programming and Languages ISPS '2011*.

BrainVISA, T. (2012). Brainvisa official website. Retrieved January 27, 2012. http://www.brainvisa.info.

Caia, X., Langtangen, H. P., and Moea, H. (2005). On the performance of the python programming language for serial and parallel scientific computations. *Scientific Programming*, 13:31–56.

Chappell, D. (2012). *Introducing Windows Azure*. David Chappell and Associates. Retrieved January 27, 2012. http://www.davidchappell.com/OnBeingIndependent– –Chappell.pdf.

Dean, J. and Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *Proceedings of the OSDI'04: Sixth Symposium on Operating System Design and Implementation*.

Desprez, F. and Jeannot, E. (2004). Improving the gridrpc model with data persistence and redistribution. In *Proceedings of the Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (ISPDC/HeteroPar'04)*.

Dormando (2012). What is google app engine ? Retrieved January 27, 2012. http://memcached.org.

Foster, I. (2006). Globus toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology*, 21(4):513–520.

FSL (2012). Fsl website. Retrieved January 27, 2012. http://www.fmrib.ox.ac.uk/fsl/.

Glatard, T., Montagnat, J., Lingrand, D., and Pennec, X. (2008). Flexible and efficient workflow deployment of data-intensive applications on grids with moteur. *International Journal of High Performance Computing Applications*.

Google (2012). What is google app engine ? Retrieved January 27, 2012. http://code.google.com/appengine/docs/whatisgoogleappengine.html.

Haroun, R., Oulebsir-Boumghar, F., Hassas, S., and Hamami, L. (2005). A massive multi agents system for brain mri segmentation.

Laguel, H. (2010). *Déploiement sur une plateforme de visualisation 3D, d'un algorithme coopératif pour la segmentation d'images IRM, autour d'un système multi-agents. Computer Sciences P. F. E., 12 Oct. 2010, directed by F. Oulebsir-Boumghar., FEI, USTHB Alger*. USTHB.

Matsuoka, S., Nakada, H., Sato, M., and Sekiguchi, S. (2000). Design issues of network enabled server systems for the grid. grid forum, advanced programming models working group whitepaper. volume 1971, pages 4–17.

MedInria (2012). Mediniria website. Retrieved January 27, 2012. http://med.iniria.fr.

Seabon, M. (2012). Plash's sandbox environnment. Retrieved January 27, 2012. http://plash.beasts.org/environment.html.

Spade (2012). Spade2 - smart python agent environment. Retrieved January 27, 2012. http://code.google.com/p/spade2/.

Vecchiola, C., Pandey, S., and Buyya, R. (2009). High-performance cloud computing: A view of scientific applications. In *ISPAN 09: Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, pages 4–16.

VUILLEMIN, P. (2008). *Calcul itératif asynchrone sur infrastructure pairàpair : la plateforme JaceP2P*. PhD thesis, Université de Franche-Comté UFR Sciences et Techniques Laboratoire d'Informatique de l'Universit de Franche-Comté.