Recent advances of MedIGrid PSE in an LCG/gLite environment

V. Boccia[1], L. Carracciuolo[2], L. D'amore[1], G. Laccetti[1], M. Lapegna[1] and A. Murli[3]
[1]University of Naples Federico II, vania.boccia@unina.it, luisa.damore@dma.unina.it, giuliano.laccetti@dma.unina.it, marco.lapegna@dma.unina.it
[2]CNR, luisa.carracciuolo@cnr.it
[3]SPACI, almerico.murli@dma.unina.it

***Abstract*—In the context of the S.Co.P.E. national project, here we are concerned with improvements and enhancements of a medical imaging grid-enabled framework, named MedIGrid, oriented to the transparent use of resource-intensive applications for managing, processing and visualizing biomedical images. In particular we describe an implementation of the MedIGrid PSE in the environment of the S.Co.P.E. project, based on the integration of the LCG/gLite and SCOPE-toolkit middlewares, reporting some planning strategies aimed to obtain more efficiency in data management and job submission. First we focus our attention on how to exploit the features of the new middleware environment to improve the PSE efficiency and services reliability: in particular our choices allowed to avoid:**

- **bottlenecks due to multiple and unuseful Resource Broker (WMS) transits;**
- **bound to data sizes, imposed by WMS.**

Further, we spent also some work to modify, extend and improve the underlying numerical components to introduce, at the application level, mechanisms to sustain a given level of performance and to survive to a fault during execution flow. Finally, we also devote some comments and suggestions on possible new middleware enrichments and/or integrations.

I. Introduction

THE activity, described in this paper, is partially supported by PON S.Co.P.E. project (Italian acronym for high Performance, Cooperative and distributed System for scientific Elaboration). This is a research project with two aims: the developement of applications in fundamental research fields and the implementation of an open and multidisciplinary Grid infrastructure, involving several departments of the University of Naples Federico II distributed in metropolitan scale in the city of Naples. In this context we were involved in improving and enhancing a medical imaging grid-enabled PSE, named MedIGrid, oriented to the transparent use of resource-intensive applications for managing, processing and visualizing biomedical images (see [20], [15]). The reason of the choice of the LCG/gLite middleware as reference environment is twofold: it provides some general purpose and high level services (Data and Metadata Catalogue, Resource Broker, ...) and it is the de-facto choice for the most of the Italian (S.Co.P.E. [9], PI2S2 [8], CYBERSAR [2], CRESCO [1], SPACI [10], IGI [5], ...) and European (EGEE [3], EGI [4], ...) grid projects.

This paper is organized as follows: in Sec. II we report some details on the GRID middleware infrastructure, in Sec. III we describe the MedIGrid implementation in this environment, focusing on some planning strategies aimed to obtain more efficiency in data management and job submission; in Sec. IV are reported two MedIGrid case study numerical kernels focusing on recent modifications we have done to introduce fault tolerance and a checkpointing system at application level. Finally in Sec. V we report conclusions, future works and some

suggestions on possible middleware enrichment and/or integration aimed to support HPC applications.

II. The GRID infrastructure

As cited before, the testbed we used to develop our work is mainly the SCoPE infrastructure.

From the hardware point of view the S.Co.P.E. infrastructure is constituted by up to two thousands computational cores linked by means of infiniband connection and about 100 TB of disk space. These resources are distributed among some Departments of the University of Naples Federico II and are connected by a Metropolitan Area Network implemented by a 1 Gbps fibre channel tecnology.

From the software point of view it is essentially based on the integration of the LCG/gLite and SCOPE-toolkit middlewares. The low level middleware LCG/gLite provides the user with high level services for scheduling and running computational jobs, accessing and moving data, and obtaining information on the Grid infrastructure, all embedded into a consistent security framework [16]. To make easy for delevopers integrate their applications into the GRID environment the low level middleware has to be enriched with a set of consolidated libraries on which applications rely, to form the so called “applicative middleware”. In the context of S.Co.P.E. project, the *SCOPE-toolkit* fills the existing gap between low level middleware and applications: libraries (i.e. BLAS, LAPACK e ScaLAPACK for linear algebra and PETSc and FFT for scientific computation, other scientific software (i.e. GROMACS for molecular dynamics), compilers and so on, are collected in a self contained package that automizes (by means of standard installation/validation procedures) the dissemination of nedeed software on the computing resources, making applications easily integrated into the S.Co.P.E. GRID infrastructure.

III. The new PSE design

MedIGrid is oriented to the transparent use of resource-intensive applications for managing, processing and visualizing biomedical images. The applications used by MedIGrid are denoising of 3D echocardiographic images sequence and segmentation of 2D echocardiographic images.

In Fig. 1, is shown the web page to access the PSE: after a simple authentication phase, the user has the chance to:

- select the action (denoising, segmentation, etc.) to be executed on input images, configure some parameters and start the execution (see fig. 2);
- monitoring the execution (see fig. 3);
- use some tools, based on standard library as ImageJ [6], to visualize and post process output image data (see fig. 4)

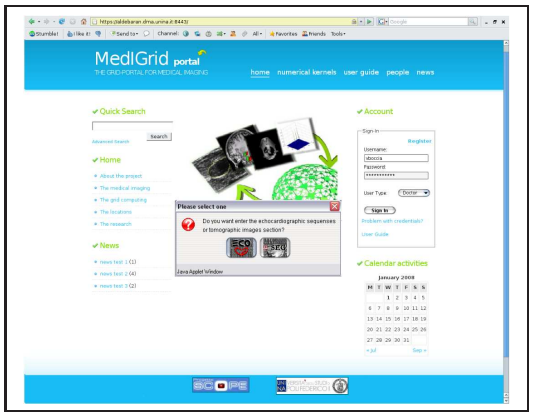

Fig. 1. MedIGrid Portal

In our previous works [**?**], [15], [20], some details are reported about a first PSE implementation in Globus environment, paying particular attention to some rilevant aspects as fault tolerance and performance maintenance. Due to a change in the underlying GRID environment (gLite instead of Globus) (see Fig. 5), our work on MedIGrid PSE involved:

- A server side modification of the GRID Portal by means of calls to functions of LCG/gLite Java library to interface the PSE with some new collective services:
  - LFC, for a more efficient data management,

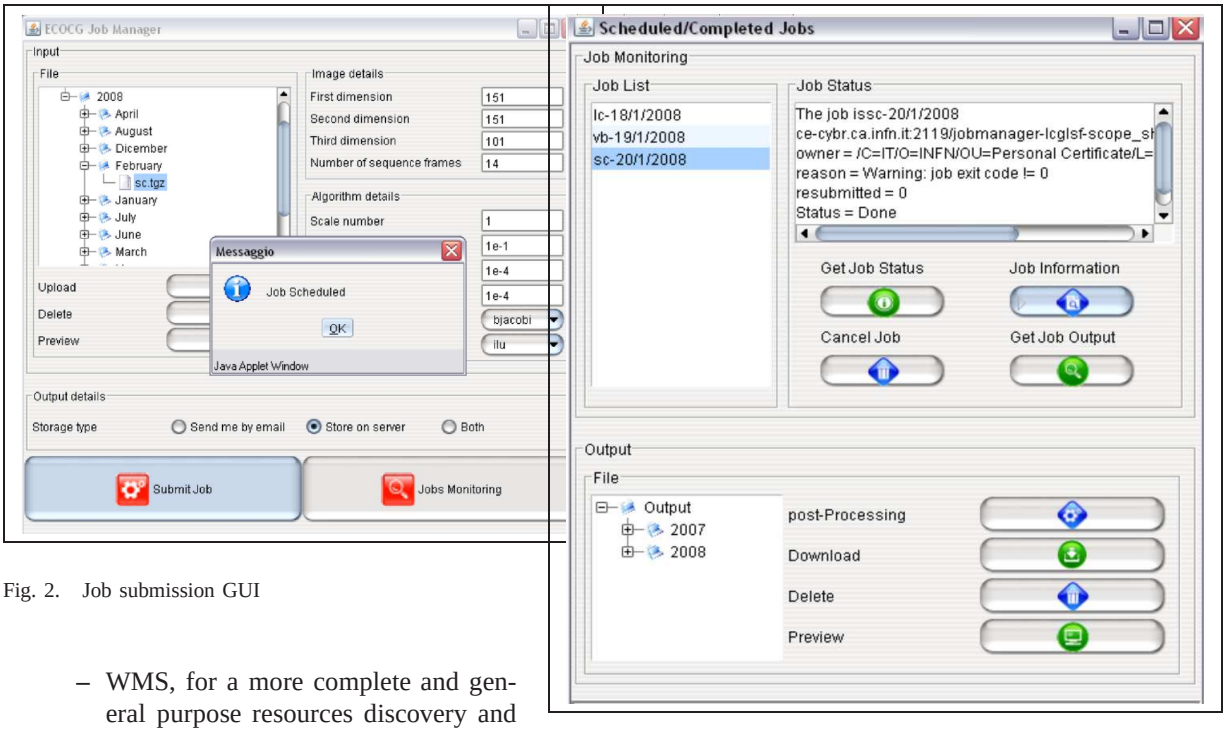

Fig. 2. Job submission GUI

Fig. 3. Job Monitoring GUI

  - WMS, for a more complete and general purpose resources discovery and selection,
  - VOMS, for the management of VO access policies to the resources and
  - MyProxy to automatic credential renew,
- The integration of some middleware features and additional code to implement a such kind of job monitoring,
- The introduction of an embryonal checkpointing system implemented at application level.

While we had to modify the "hidden" part of the PSE, we left unmodified the GUI, already complete of a satisfating number of consolidated tools for image processing parameter configuration, image visualization and post-processing.

Here our attention is focused on some planning strategies aimed to obtain more efficiency in data management and job submission.

In particular our choices allowed us to avoid bottlenecks, due to multiple and unuseful Resource Broker (WMS) transits, and limitation imposed from WMS to data sizes.

To give an idea of problem size we can observe that an input sequence of only 14 ecographic small images is large more than 20*MB* and the output of the process is $20MB \times nscales$ (where $nscales$ is an input parameter related to the output quality). As the images dimension and the number of problems to be solved encrease, we have to avoid unuseful paths in data transferring and redundance storage on WMS, to provide an efficient data management.

In MedIGrid job execution schema, application binaries and data are stored on Storage Resources and registered on a Logical File Catalogue with symbolic names. On the UI are present only text files containing image processing parameters.

At execution time, parameter files reach computational resources passing through the Resource Broker (WMS), while binaries and input data are transferred directly from Storage to Computational Resource, reducing the execution bottleneck. To reach this aim, we had to

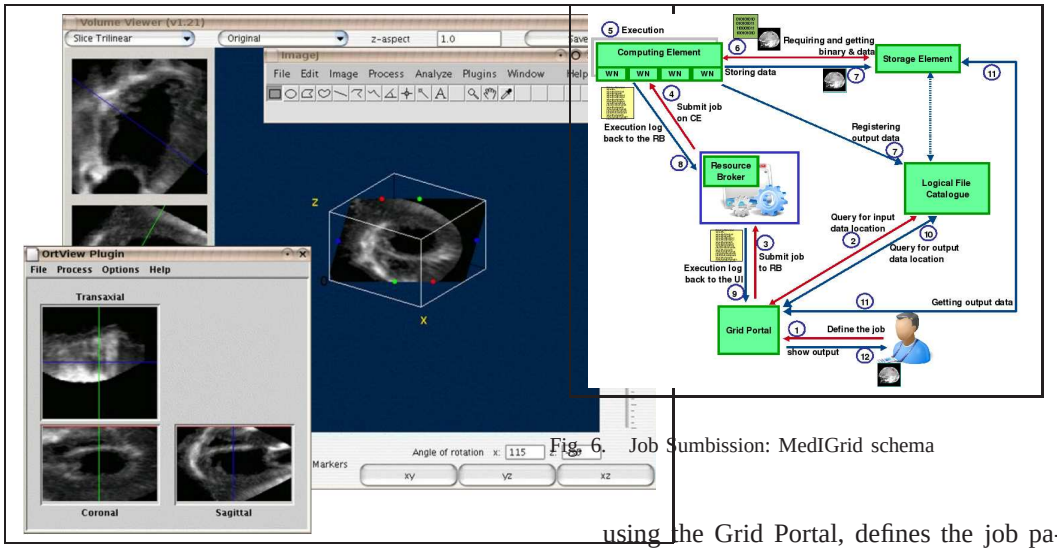

Fig. 6. Job Sumbission: MedIGrid schema

Fig. 4. Visualization GUI

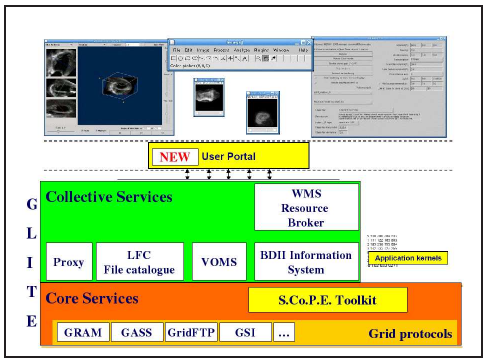

Fig. 5. New PSE architecture

modify our numerical kernels, by the means of calls to LFC/GFAL library functions: if LFC functions let us to access data transparently, by means of their logical names, GFAL tools allow efficient data transfer by multistream gridFTP low level protocol.

At the end of the execution output log file reaches the UI through the WMS, while the output image is directly moved on the Storage and registered on the File Catalogue.

Summarizing, looking at the schema in Fig. 6 some main steps are performed:

- Job definition (steps 1-2): The user, by using the Grid Portal, defines the job parameters (by chooising image processing algorithm, by browsing the File Catalogue to select input data, by fixing the output data logical name, ...) through the Grafical Interface;
- Job sumbission (steps 3-4): The submission service on Grid Portal, submits the defined job to the Resource Broker that choices a Computing Element;
- Job execution (steps 5-6): The first step of the job script asks to LFC for application binary and input data and then gets them directly from the Storage Element;
- Data archiviation (step 7): At the end of the process the job script copies the output data directly on the SE and, at the same time, register files on the File Catalogue with the symbolic name chosen at step 1 from the user.
- Output retrieval and visualization (steps 8-12): The execution log file moves from the Working Node to the User Interface through the Resource Broker to be visualized by the user; the user can require output files that, by means of their symbolic names, are moved from the SE directly on the Grid Portal and then visualized.

As reported above, at step 1 (see Fig. 6), the User defines, through the GUI, some requirements and parameters; on server side these one will be used from the *job service* to construct

the job in a Job Description Language format (JDL file). The job service is implemented by using objects and methods from native gLite standard library and from extra WMS-Proxy library [7].

```
Type="Job";
 JobType="MPICH";
 NodeNumber=6;

 Executable="ECOCG.sh";
Arguments="6 scope scopelfc01.dsf.unina.it
            scoperb01.dsf.unina.it:2170
            scopese01.dsf.unina.it
            2008/February/
            sc 1e-1 1e-4 1 1e-4 150 150
            100 14";
StdOutput="ECOCG.err_out";
StdError="ECOCG.err_out";
 InputSandbox="ECOCG.sh";
 OutputSandbox="ECOCG.err_out","Denoise3d.out";
```

Fig. 7. The JDL File generated by using WMS-PROXY Java API

In Fig. 7 we show an example of JDL file generated. We highlight (see text in bold face) some rilevant attributes:

- `JobType` and `NodeNumber` used to identify a parallel job (based on MPI)
- `InputSandbox` that contains only the script file with parameters setting
- `OutputSandbox` that contains only the job execution log files.

In Fig. 8, we report the script file code. We highlight (see text in boldface) some rilevant lines:

- `PROGARGS` variable that contains all user defined parameters
- the download of input data and binaries, from the storage to the computing resource, using `lcg-cp` command
- the execution of parallel application, using `mpirun` command
- the output data transferring on the storage and its registration on file catalogue, using `lcg-cr` command

Job monitoring system is implemented by combining two mechanisms:

- the *perusal* feature provided by WMS broker, and accessed by means of Java API, to retrive the job execution log files at runtime
- a custom service that retrives partial output saved by the application during its execution (see checkpointing mechanism described later on).

```
#!/bin/sh
 # ECOCG.sh
CPU_NEEDED=$1
VO=$2
LFC_HOST=$3
LCG_GFAL_INFOSYS=$4
VO_DEFAULT_SE=$5
NAME_PATH=$6
NAME=$7
FILEN=$NAME_PATH/$NAME
# algorithm parameter setting:
# N1, N2, N3, SIGMA, SCALESTEP,
# NSCALES, RCONVERGENCE, NOF.
OUTFILE=$NAME_PATH/$16
 PROGARGS=-filename $NAME
          -scalestep $SCALESTEP
          -sigma $SIGMA -nscales $NSCALES
          -iz 1 -rconvergence $RCONVERGENCE
          -n1 $N1 -n2 $N2 -n3 $N3
          -ksp_type gmres -pc_type bjacobi
          -sub_pc_type ilu -ksp_max_it 300
          -nof $NOF
echo "Downloading binaries and input data"
echo "  ================================="
 lcg-cp --vo $VO
    lfn:/grid/$VO/MEDIGRID/SOFTWARE/Denoise3d
    file:'pwd'/Denoise3d
 lcg-cp --vo $VO
    lfn:/grid/$VO/MEDIGRID/ECOCG/Input/$FILEN.tgz
    file:'pwd'/$NAME.tgz
tar xzvf $NAME.tgz
echo "  ================================="

echo "Executing mpirun"
echo "  ================================="
 mpirun -np $CPU_NEEDED -machinefile
          $HOST_NODEFILE 'pwd'/Denoise3d
          $PROGARGS >  Denoise3d.out 2> &1
echo "  ================================="

echo "Uploading Output files"
echo "  ================================="
tar czvf $NAME.OUT.tgz Stato.txt $NAME*_*
 lcg-cr --vo $VO -d $VO_DEFAULT_SE
    -l lfn:/grid/$VO/MEDIGRID/ECOCG/Output
    /$OUTFILE.tgz file:'pwd'/$NAME.OUT.tgz
echo "  ================================="
```

Fig. 8. An example of execution script

IV. PSE NUMERICAL COMPONENTS

As cited in the previous section, both applications are based on the *Portable, Extensible Toolkit for Scientific Computation* (**PETSc**) library [11], [12], [13], a suite of data structures

and routines that provide the building blocks for the implementation of large-scale codes concerning the scalable parallel solution of scientific applications modeled by partial differential equations. Details can be found in [17], [18].

With the aim of modify, extend and/or improve the numerical components based on PETSc, we are working on the introduction, at *application level*, of a some kind of checkpointing methods, that will be combined with fault-tolerance mechanisms.

With an *application level* approach, the mechanisms enabling the fault-tolerance features in the grid appplications, are implemented in the algorithms of computational kernels and they are not demanded to the middleware, delegating the application developer to deal with the matter of choosing and saving data appropriately. On the other hand, with this strategy, the application is able to gain an higher level of efficiency in storing and retrieving the data from which performing restarting tasks.

Both the denoising and segmentation algorithms are based on iterative schemes and, in order to recover from a fault, they can restart from those ”points” corresponding to the last ”correct” computed data. So we can use a kind of *“disk based checkpointing”* method, registering those data to the LFC service, at each iteration, so that we could be able to restart with the last computed data; at fault occurrence, application is automatically re-submitted on alternative resource accessing LFC service to obtain last computed data from which perform restarting task. The disk-based checkponting strategy in an iterative scheme collects the following key steps:

- the last ”correct” computed data is retrieved from remote storage resource by LFC service;
- the current data is computed;
- the computed data is stored on remote storage resource and registered on the LFC.

We can observe that, such kind of *disk-based* checkpointing, give us the chance to replicate data on more remote storages (as allowed by LFC/GFAL system): data are always available, even if storage and computing resources can become unavailable. This implement a storage service reliability, also paying for overhead in data transfer.

To find and handle faults in Message Passing system, we have to modify appropriately both the application and the “high level” middleware. Thus, to use WMS authomatic re-scheduling, the application has to execute the following tasks:

1) checking the presence of fault after each communication phase, and,
2) if fault is found, terminating execution with an exit non-zero code.

V. Conclusions and Future Works

This paper describes some work done to implement the MedIGrid PSE in an LCG/gLite environment. Things are much evolved, starting from our preliminary results, and, although this experience has been realized in economy of thought, features and tools now available in the LCG/gLite middleware open new scenarios for the medical community to collaborate and share resources, information and knowledge.

As mentioned in previous section, to modify, extend and/or improve the underlying numerical components, we have to introduce some checkpointing techniques, to be combined with fault-tolerance mechanisms. To introduce fault-tolerance mechanisms, we have to modify appropriately both the application and the “high level” middleware components (i.e. PETSc).

Actually our work involves the implementation of mechanisms that handle all communication faults, at level of .

As described in this document, we used the high level middleware mechanisms to manage in efficient way data transfer, but no much work, until now, has been devoted on data security and simple metadata management (very relevant problem if we think to medical field). In the future, we will take part to the development of middleware tools and services aimed to handle this matter in a standard way (i.e. by introducing DICOM support both at application and at middleware level).

REFERENCES

[1] *CRESCO Project* Home Page. http://www.cresco.enea.it/

[2] *CYBERSAR Project* Home Page. http://www.cybersar.com.

[3] *EGEE Project* Home Page. http://www.eu-egee.org/.

[4] *EGI Project* Home Page. http://web.eu-egi.eu/.

[5] *IGI Project* Home Page. http://grid.infn.it/igi/partners.html.

[6] *ImageJ: Image Processing and Analisys in Java.* http://rsbweb.nih.gov/ij/index.html.

[7] *Java Library for WMProxy* https://grid.ct.infn.it/twiki/bin/view/GILDA/ApiJavaWMProxy.

[8] *PI2S2* Project Home Page. http://www.pi2s2.it/.

[9] *S.Co.P.E. Project* Home Page. http://www.scope.unina.it.

[10] *SPACI Project* Home Page. http://www.spaci.it/.

[11] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. *PETSc users manual* Technical Report ANL- 95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[12] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. *PETSc Web page*, 2001. http://www.mcs.anl.gov/petsc.

[13] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. *Efficient management of parallelism in object oriented numerical software libraries.* In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, Modern Software Tools in Scientific Computing, pages 163-202. Birkhäuser Press, 1997.

[14] G. B. Barone, D. Bottalico, V. Boccia, L. Carracciuolo, S. Pardi, M. Scognamiglio, and F. Serio. *Middleware Applicativo: lo S.Co.P.E. Toolkit*. In Universit'a degli Studi di Napoli "Federico II", editor, Proceedings of Italian National Conference of e-Science 2008, Napoli, 2008. CRUI.

[15] V. Boccia, L. D'Amore, M. Guarracino, and G. Laccetti. *A Grid Enabled PSE forMedical Imaging Applications: Experiences on MedIGrid.* In Proceedings of The 18th IEEE Symposium on Computer-Based Medical Systems, pages 529-536, Boston, 2005. Springer-Verlag.

[16] S. Burke, S. Campana, P. M. Lorenzo, C. Nater, R. Santinelli, and A. Sciaba. *GLITE 3.1 USER GUIDE* 2008.

[17] L. Carracciuolo, L. D'Amore, and A. Murli. *Towards a parallel component for imaging in PETSc programming environment: A case study in 3-D echocardiography.* Parallel Computing, 32:67-83, 2006.

[18] L. D'Amore, D. Casaburi, L.Marcellino, and A.Murli. *Segmentazione di sequenze di immagini digitali mediante previsione del moto.* TR 62, Dip. di Matematica e Applicazioni, Università degli Studi di Napoli Federico II, 2006.

[19] E. Laure and alt. *Programming the Grid with gLite.* Computational Methods in Science and Technology, 12:33-45, 2006.

[20] A. Murli, V. Boccia, L. Carracciuolo, L. D'Amore, G. Laccetti, and M. Lapegna. *Monitoring and Migration of a PETSc-based Parallel Application for Medical Imaging in a Grid computing PSE*. In P. Gaffney and J. Pool, editors, Grid-Based Problem Solving Environments, pages 421-432, Boston, 2007. Springer-Verlag.

[21] V. Boccia, L. Carracciuolo, L. D'Amore, G. Laccetti, M. Lapegna, A.Murli *The MedIGrid PSE in an LCG/gLite environment* International Symposium on Parallel and Distributed Processing with Applications International 2008 (ISPA '08), Workshop on High Performance and Grid Computing in Medicine and Bioinformatics (HiPGCoMB) Sydney, Australia 10-12 Dicembre 2008, ISBN: 978-0-7695-3471-8, pp.829-834.