J Grid Computing
DOI 10.1007/s10723-012-9223-6

HPC on the Grid: The Theophys Experience

Roberto Alfieri · Silvia Arezzini · Alberto Ciampa · Roberto De Pietri · Enrico Mazzoni

Received: 27 January 2012 / Accepted: 6 July 2012

Abstract The Grid Virtual Organization (VO) "Theophys", associated to the INFN (Istituto Nazionale di Fisica Nucleare), is a theoretical physics community with various computational demands, spreading from serial, SMP, MPI and hybrid jobs. That has led, in the past 20 years, towards the use of the Grid infrastructure for serial jobs, while the execution of multi-threaded, MPI and hybrid jobs has been performed in several small-medium size clusters installed in different sites, with access through standard local submission methods. This work analyzes the support for parallel jobs in the scientific Grid middlewares, then describes how the community unified the management of most of its computational need (serial and parallel ones) using the Grid through the development of a specific project which integrates serial e parallel resources in a common Grid based framework. A centralized national cluster is deployed inside this framework, providing "Wholenodes" reservations, CPU affinity, and other new features supporting our High Performance Computing (HPC) applications in the Grid environment. Examples of the cluster performance for relevant parallel applications in theoretical physics are reported, focusing on the different kinds of parallel jobs that can be served by the new features introduced in the Grid.

Keywords HPC · Grid · Theoretical physics

R. Alfieri (✉) · R. De Pietri
INFN Parma and Parma University,
Viale G.P. Usberti 7/A, Parma, Italy
e-mail: roberto.alfieri@fis.unipr.it

S. Arezzini · A. Ciampa · E. Mazzoni
INFN Pisa, Polo Fibonacci Largo B. Pontecorvo 3,
Pisa, Italy

1 Introduction

Since early 2000s, Grid [1] has been the emerging paradigm for the setting up of distributed computational infrastructures and services. This paradigm has been mainly used for serving the needs of High Throughput Computing (HTC), but it was scarcely used for parallel applications. The INFN has been always at the forefront on the development and deployment of Grid enabled computing farms and storage systems, but it was relaying on small local clusters or dedicated HPC facilities for all of its needs for parallel resources. Recently, the general availability of multi-core architectures have pushed the demand for parallel and multi-threaded resources, not only in communities like the Theophys one, involved in compute-intensive simulations.

This trend has brought the Theophys community to consider the opportunity to undertake a project towards a common framework, able to

Published online: 05 August 2012

integrate all its resources in the EGEE distributed infrastructure [2], which was widely used by the community for the management of serial jobs. The project would have carried out important (and obvious) benefits in terms of resources management, exploitation and usability. The need to install new resources for the whole Theophys community brought to the deployment of a national cluster designed to support heterogeneous applications (serial, parallel and multi-thread) and to be accessed as a Grid service.

Various scientific communities have developed proposals or projects addressed to provide a uniform interface for users, or to enable the submission of applications requiring multi-scale resources. These objectives have been investigated using very different approaches.

A significant example of this trend is given by the effort made by the Computational Chemistry initiative [3], led by the University of Kentucky and involving other Universities and supercomputer centers in USA, with the aim to provide a common infrastructure for the chemistry community. The approach has been the development of a dedicated Grid infrastructure, CCG (Computational Chemistry Grid), consisting of basic Grid middleware components, such as the Globus Toolkit (see Section 3) and CGI scripts, providing core functionalities to the client applications.

Other communities, like ours, are pursuing the idea of performing multi-threaded or HPC computation into standard Grid infrastructure, like the example in the field of Cosmological simulations [4]. Although the work demonstrated that the EGEE infrastructure can be successfully used to run numerical simulations, the authors claim that the possibility to select WNs on the basis of the memory, number of cores and local disk space would simplify the porting and the management of their parallel jobs.

A similar approach has been pursued in the field of the Earth Science [5], where has been done an analysis of the requirements for the porting of MPI applications in the EGEE environment. Despite the improvements in the MPI support, thanks to the work of specific MPI Working Groups [6] promoted by EGEE, the infrastructure was not considered a production tool suitable for their applications.

Last related work analyzed here is the ShareGrid Peer-to-Peer Desktop Grid [7] that is a Grid infrastructure aiming at the federation of resources provided by a set of small research laboratories. The infrastructure is designed to have good scalability and performance for different kind of jobs, including serial, parallel and SMP ones. What is worth of mention in this work is a complementary technique adopted for the multicore support, based on the use of multiple virtual machines per physical machine. With this solution, the users can agree with the site administrators on a specific hardware and software environment, which can be deployed on clusters composed by heterogeneous hosts. Disadvantages are the overhead introduced in the resources usage and the requirement of relevant configuration efforts.

This paper is organized as follow. In Section 2 we give a short description of the users community, its computational needs and the project objectives. In Section 3 we analyze the actual HPC support in the most common Grids. In Section 4, that is the main part of the present work, we discuss problems encountered and solutions adopted in the design and deployment of the project. In Section 5 there is a description of two applications in theoretical physics executed in the Grid environment, that we used to test the new features introduced by the project. The conclusions of our work are drawn in Section 6.

2 The INFN Theoretical Physics Community, Computational Needs and Project Motivations

The INFN theoretical physics community consists of more than one thousand researchers distributed over 28 sites and involved in more than 60 research projects. In this community, beside several activity that require computation based on serial jobs, there are research fields, such as Lattice Quantum Chromo-Dynamics (QCD), Fluid Dynamics and Numerical Relativity, which relies on the execution of massive parallel applications. These applications require HPC resources ranging from large PC clusters up to specialized supercomputers.

In the past these computational needs have been fulfilled through very different solutions.

The EGEE Grid infrastructure has been widely used for serial calculation. During the year 2010 Theophys VO has been the sixth major CPU consumer, with about 1 Million of executed jobs (see Fig. 1).

In theoretical physics there is the request of numerical simulations, such as Lattice QCD, requiring the computational power provided only by massively parallel supercomputers. Since early 80s INFN has developed a own supercomputers family, named APE [9], dedicated to such problems. QCD simulation jobs have been executed also at Cineca and in general on PRACE facilities (see Section 3).

For smaller parallel jobs ("medium-sized") several independent small clusters have been installed and accessed using standard methods and local policies. These resources are used for different levels of parallelism, ranging from applications which require the exclusive access to a single whole-node for multi-threaded applications, or single process applications requiring the exclusive usage of the whole memory of the node, up to complex hybrid MPI-openMP applications (see Section 5) requiring hundreds of cores.

Often research groups have multi-scale applications requiring access to different kind of resources (see Fig. 2). Consequently, users have to interact with different resource providers and to manage different access credentials and methods. The use of independent resources leads to a waste of computational power and administration efforts. The need of a single shared platform was considered increasingly important and Grid appeared to be the natural solution since it has been successfully used by the community for serial applications.

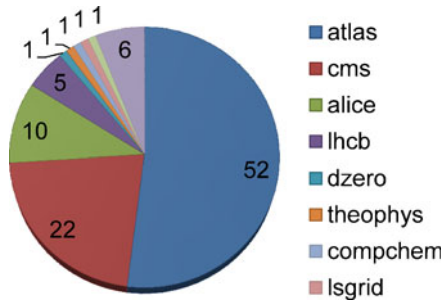

Fig. 1 Theophys Grid usage. This chart, derived from the EGI Accounting Portal [8] shows the usage in 2010 of the Grid infrastructure by the Theophys VO in comparison with the other EGI/EGEE VOs, in terms of Normalized CPU time

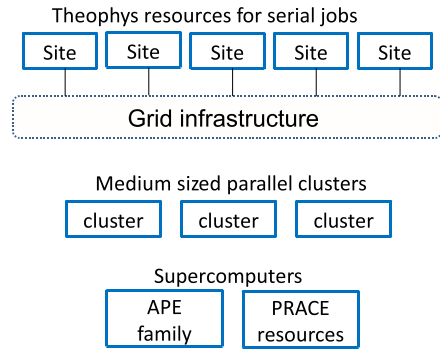

Fig. 2 Theophys resources. In 2010 there were 3 types of resources: EGEE infrastructure for serial jobs, small parallel clusters dedicated to medium-sized jobs and supercomputers for LQCD simulations

For these reasons, in early 2010, we decided to evaluate the reliability of the sites, the flexibility of the support and in general whether the middleware was suitable for our "medium-sized" applications. The evaluation has been realized through a series of probe jobs submitted to all sites supporting MPI and the Theophys VO. These probe jobs execute a shell script that collects environmental information from WNs (LRMS type, flavor and version of the MPI implementations) and estimates network and CPU performance through the execution of basic bench-marking programs. The results, presented in Table 1, show that there was a general lack of opportunity to use the Grid to perform even small scale parallel computation on the already active resources.

An important step towards a better support for HPC has been the release in June 2010 of a document proposed by the EGEE MPI Working Group [6]. The document recommended a method for deploying MPI support that should work for both users and site administrators and it proposed new middleware features to support the upcoming multi-core architectures.

Table 1 The table presents the situation of the MPI support among sites supporting Theophys VO in early 2010, prior to the start of the project

MPI/LRMS	Status	Network (μs-MBytes/s)	CPU (GFlops)
MPICH/PBS	OK	114-54	0.7
MPICH/PBS	OK	109-56	0.7
MPICH/PBS	OK	114-54	0.7
MPICH/LSF	OK	98-108	1.2
MPICH/LSF	OK	94-101	1.3
MPICH/PBS	OK	1287-10	0.4
MPICH/LSF	mpich misconf.	–	1.3
MPICH/LSF	mpich misconf.	–	1.5
MPICH/LSF	mpich misconf.	–	1.5
MPICH/LSF	mpich misconf.	–	1.5
MPICH/LSF	ssh misconf.	–	0.8
MPICH/LSF	ssh misconf.	–	0.7
MPICH/LSF	ssh misconf.	–	0.8
MPICH/PBS	aborted	–	–
MPICH/PBS	aborted	–	–

The numbers presented in the Network column specify Latency and Bandwidth measured among 2 nodes of the cluster. Almost half of the resources were working for serial jobs, but not for parallel ones, due to software (ssh or MPICH) misconfigurations

This recent evolution has convinced the INFN Theoretical Physics community to start a project with different objectives and different time-scales:

- to provide to the community a single large resource able to satisfy all the small and medium-sized computational needs described in Section 2. This objective has clear advantages in terms of simplification of the user interface, administrator tasks, resource exploitation, and transparency on the user's activity through a detailed web accounting service. Of course, this result can be achieved with a traditional cluster too.
- to integrate the medium sized activity in the existing Grid infrastructure implemented for serial jobs, as mentioned earlier. Here the need for a Grid-enabled HPC cluster has evident benefits for both users and administrators point of view: architectural components and functionalities (such as AAA services), job life-cycle management, resources integration and management can be reused for the new resources.
- to realize the first step toward a single, global infrastructure for the whole community, based on Grid services (see Fig. 3 for a view of the envisioned resources architecture). The infrastructure should be able to integrate old and new parallel clusters, so the design of the first parallel cluster, we are describing here, must be scalable and reliable. Moreover, as we will see later on (see Section 3), PRACE, that is actually an important supercomputing provider for the Theoretical physics community, could be able in a near future to extend its services to Grid users. This integration can be endorsed by the scientific communities, especially if every other computational activity of the communities is already performed inside the infrastructure.

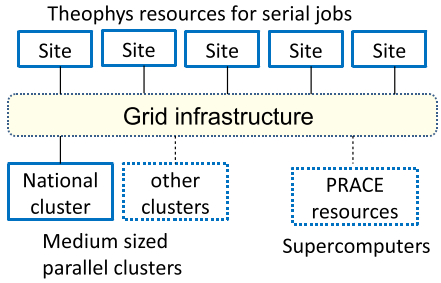

Fig. 3 The desired Theophys infrastructure

3 HPC Support in the Grids

A widely used middleware is the Globus Toolkit (GT) [10]. It includes software modules for security (GSI), data services (GridFTP), Information services (MDS), Execution Management Services (GRAM) and communication, which are often taken by other projects to build new middlewares.

For non-trivial jobs, the user has to write a Globus job script, which is written using the Resource Specification Language (RSL). In case of parallel jobs the RSL parameter needed to specify the CPUs number is "count". The following is an

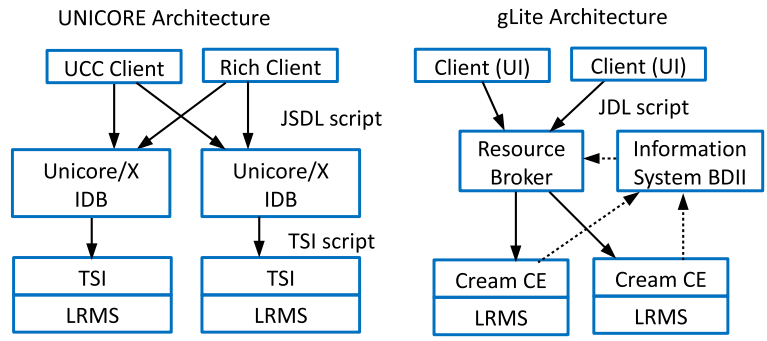

Fig. 4 Comparison between gLite and UNICORE architectures from the job generation process point of view

example of a RSL file (GT4 syntax) requiring 8 CPUs:

```
<job>
  <executable>mpi_test</executable>
  <stdout>mpitest.stdout</stdout>
  <stderr>mpitest.stderr</stderr>
  <count>8</count>
  <jobType>mpi</jobType>
...
</job>
```

Some middlewares, such as gLite [11] and ARC [12] (Advanced Resource Connector), provide users with high level services for scheduling and running computational jobs, accessing and moving data, and obtaining information on the Grid infrastructure. They still use selected Globus Toolkit libraries, chiefly in the security context (GSI). Other middlewares have been developed to support specialized services such as dCache [13] which is designed for data management, and UNICORE (Uniform Interface to Computing Resources) [14] which is specialized in supercomputing applications.

In 2010 gLite, ARC, UNICORE and dCache joined forces to develop and unify their middlewares in a common project called EMI (European Middleware Initiative) [15].

Considering the purpose of this work, we analyze here a comparison between the parallel support in gLite, which was the middleware in use by Theophys for serial jobs at the moment of the project start up (see Section 2), and in UNICORE, which was specifically created for HPC applications [16] (Fig. 4).

UNICORE supports resource specifications through a Job description file, written in JSDL (OGF standard) [17] with the addition of extensions for parallel applications and the support for the Execution Environments. The Execution Environments, specified by the administrator by means of the IDB file, provides to the users a high level of abstraction, hiding resource details and providing simple execution environments to the user [18].

The user can specify resources and environment requirements through a command line script or a Java web portal (Rich client). The main resource requirements supported by UNICORE are the total number of CPUs, number of nodes, CPUs per node and RAM per node. Through the Execution Environment it is possible to specify the software environment and other options, such as the MPI flavor and the number of requested processes (see Fig. 5 for an example).

The parallel support in gLite is derived from the recommendations of specific MPI Working Groups promoted by EGEE. It is achieved through the Job Description Language (JDL) attribute `CPUNumber`, which specifies the number of Job Slots that will be requested to the Resource Manager (see Fig. 6 for an example).

```
{
 Executable: "./hello.mpi",
 Imports: [ {From: "/myfiles/hello.mpi",
             To: "hello.mpi" }, ],
 Resources:{ CPUsPerNode: 2, Nodes: 2, },
 Execution environment: { Name: OpenMPI,
         Arguments: { Processes: 12, }, },
}
```

Fig. 5 UNICORE Command-line client example

```
CPUnumber = 8;
Executable = "mpi-start-wrapper.sh";
Arguments = "my-mpi-prog OPENMPI";
InputSandbox = "mpi-start-wrapper.sh my-mpi-prog";
OutputSandbox = {"std.out", "std.err"};
StdOutput     = "std.out";
StdError      = "std.err";
Requirements  =
member("MPI-START",
other.GlueHostApplicationSoftwareRunTimeEnvironment)
&& member("OPENMPI",
other.GlueHostApplicationSoftwareRunTimeEnvironment)
;
```

Fig. 6 MPI job in gLite: JDL script example

MPI-Start [19] is an abstraction layer, located between the middleware and the underlying LRMS and MPI flavors, that offers a unique interface to start parallel jobs with different implementations of the execution environment. The architecture is organized in three main frameworks: The scheduler framework provides support for different Local Resource Management Systems (LRMS). It detects the availability of a given scheduler and generates a list of machines that will be used to execute the application. The Execution framework prepares the environment needed to start the particular MPI implementation. The user has only to specify which MPI flavor will be used to execute the application. The Hooks framework is responsible for the files distribution in case of file system not shared between WNs and it manages customizable pre-run and post-run hooks.

The example reported in Fig. 6 shows the JDL file of a job demanding 8 processors and the OpenMPI flavor. The Requirements attribute makes the brokering service to select sites publishing that they support MPI-Start and OpenMPI.

A step towards a better support for HPC in gLite has been the release in June 2010 of a document proposed by the second EGEE MPI Working Group [6] which proposed the introduction of specific JDL Tags. The new features are designed to allow users to specify how cores should be distributed over the cluster, whether full nodes should be used or not and how many nodes should be involved in the allocation itself.

In the comparison between the MPI supported methods in UNICORE and gLite we can distinguish two separate stages: the resource selection and the job environment setup.

The resource selection in gLite has a higher abstraction level thanks to the Resource Broker, which permits the integration of large sets of serial and parallel resources. In UNICORE users have a higher level of flexibility, since they can specify a more detailed hardware environment, such as CPUs per node and RAM per node, that are needed to support multi-thread and hybrid applications. This feature was specified in gLite as a recommendation document, but it was not implemented.

The job environment setting is an advanced feature that allows users to configure the way jobs are executed in a more detailed and user-friendly fashion. A common scenario is the configuration of an environment for parallel jobs, such as MPI. UNICORE explicitly supports this feature in the UNICORE/X Execution Environment. The same feature is provided in gLite by MPI-Start.

Since the major mandate of EMI is the unification of the different middlewares, works are ongoing towards a common solution for the execution task. EMI-ES (EMI Execution Service) [20] is a specification for the common Execution Service, developed by the EMI project, that intends to define a joint web service interface to ARC (ARC-CE), gLite (CREAM) and UNICORE compute services, by incorporating good ideas from all three. The HPC capabilities supported by EMI-ES are NumberOfSlots, SlotsPerHosts, ExclusiveExecution (whether a host should be allocated for exclusive use by the job), ProcessPerHost and ThreadsPerProcesses. The deployment of EMI-ES (foreseen in EMI-2) will make possible the introduction of common clients and the interoperability among different Computing Elements.

3.1 Grid Infrastructures

Grid technologies have been introduced in Europe since early 2000s pushed by big scientific experiments and projects, starting from LHC at CERN [21]. In these years the European Union (EU) has funded several projects aiming at the setting-up of a continental infrastructure able to provide computational resources to the whole scientific community.

The European infrastructure in use by the Theoretical Physics community at the time of the project start was EGEE, based on the gLite middleware.

EGEE ended in 2010 and was substituted by the EGI project (European Grid Infrastructure) [22]; EGI is now based on the EMI middleware and is composed of a central coordinating body (called the EGI.eu) and more than 40 National Grid Initiatives (NGIs), representing the resource infrastructure providers. The users are organized in Virtual Organizations and Theophys is the Virtual Organization representing the Italian Theoretical Physics community, supported by IGI, which is the Italian NGI.

Grid technologies are now in use also in the most important supercomputing consortia in US and Europe:

XSEDE [23] is a US project supporting 16 supercomputers and data analysis resources across the country, replacing and expanding the former TeraGrid initiative [23]. The Grid interconnection is provided by OSG (Open Science Grid) [24] which is a Grid infrastructure based on Globus Toolkit with additional modules for security, storage and data management, work-flow and other higher level services.

The analogue European initiative is PRACE (Partnership for Advanced Computing in Europe) [25] which is a EU funded project established to create a permanent pan-European High Performance Computing service for research. The project incorporate the previous initiative, DEISA (Distributed European Infrastructure for Supercomputing Applications) [26], which was completed in April 2011. PRACE is composed by more than 20 partners, whose HPC facilities are organized in a hierarchical model. The higher level, Tier-0, includes 6 supercomputers located in Germany, France, Spain and Italy. The Tier-1 machines are national supercomputers made available from Finland, France, Germany, Ireland, Italy, Poland, Sweden, The Netherlands, Serbia, Switzerland, Turkey, and United Kingdom. PRACE has deployed a common services infrastructure, Grid based, to integrate and operate the Tier-0 and Tier-1 systems as a single distributed Research Infrastructure and to provide a transparent access to users. From the Grid perspective, an important feature is the support of UNICORE as a core service for both Tier-0 and Tier-1 facilities [25].

Recent scientific projects are growing in Europe relying on resources provided by both supercomputer centers, like PRACE, and HTC Grid infrastructures, like EGI. These projects share the common interest in multi-disciplinary multi-scale models and they require large scale or even extreme scale computing capabilities.

Among others, we mention GEMS [27] and MAPPER [28]. GEMS is an application software developed inside CompChem VO, that distributes a large quantity of independent tasks on a HTC platform (EGI). Their results are passed as a set of multiple inputs for a a final elaboration implemented on a HPC platform hosted at Cineca (the Italian PRACE member). The computational solution pursued in this project is the definition of a work-flow engine that combines together HTC and HPC elements. It will be based on EMI-ES, as soon as it will be available.

MAPPER is an EU-funded project aiming at the integration of heterogeneous infrastructures for programming and execution of multiscale simulations from different user communities: physiology, computational biology, fusion, hydrology and nano-material science. The new services and interoperability tools will be introduced by joint task-force between EGI, MAPPER and PRACE.

The scientific interest towards a convergence between HTC and HPC has been one of the main reason behind the development of our project and it follows the line of integrating, into the EMI middleware, the Grid components in use in Supercomputing Centers.

4 The Project

Although, even if the middlewares comparison (see Section 3) has highlighted that UNICORE was more suitable for HPC support, we decided to adopt the gLite middleware for the following reasons:

– GLite is a stable and robust middleware for HTC and is actually the middleware in use in the Italian NGI (IGI). This means that

the Theophys community of users and administrators has a deep experience with it and, moreover, a consolidated infrastructure based on gLite is in place, providing services that can be used for HPC resources too.
– Although the support and the usage of parallel computing in gLite was quite inadequate, the Recommendation document elaborated by the EGEE MPI-WG has accelerated the process towards an improvement of the support. As a consequence, an additional objective of our work has been the contribution towards a real start up of the usage of parallel computing in gLite, through a close collaboration with the middleware developers.

The key points we faced in the design of a resource able to support not only serial and MPI but also SMP and hybrid applications, as described earlier, have been the following:

– how to implement "WholeNodes support", which was one of the main targets of the project, since, when we decided to start, the support wasn't implemented yet. This point cover two distinct aspects: how to get the whole nodes allocated and how to distribute MPI ranks and/or threads on the reserved Job Slots.
– how to deal with serial and parallel jobs or, in other words, how to keep serial jobs out of parallel resources. Indeed serial jobs without any requirement may land on any resources, including parallel ones.
– An issue faced in this project, concerning any HPC resources (Grid and traditional ones), is the optimization in the cores utilization, by maximizing cores exploitation and minimizing parallel jobs starvation.
– often jobs in theoretical physics are HTC (running for several days) and data-intensive (see for example the Einstein Toolkit in Section 5.2), so a special point in this work is the design of a storage architecture optimized for such job types.

4.1 Cluster Characteristics

The Cluster had to be integrated into the existing Grid infrastructure and the deployment procedure active within our organization (INFN). They are standardized on the basis of the Scientific Linux (SL) distribution and the Yaim configuration tool. These requirements ruled out the possibility of consider any other cluster management solutions. Moreover, the choice of the Local Resource Management System (LRMS), in our case LSF, was a consequence of the existing infrastructure present at the PISA computing center and we did not examined other possible systems, in terms of performances or capabilities.

The main problem is that there are different software requisites that need to be harmonized. The requisites given by the local infrastructure management, the middleware and the User Communities are in general different and, in some aspect, conflicting with each other. We made the strategic choice of decoupling the handling of the requisites in the following way. On the WNs, we installed an Enterprise class operating system, that guarantees the hardware and software support. Inside that system, the middleware and user applications are run using the "chroot" mechanism.

In our case the version of the running operating system is the "Suse Linux Enterprise Server (SLES)", version 10sp2: this system guaranteed the support of the hardware characteristics, for instance the used HCA IB and the GPFS software. The middleware, instead, is installed on a specific "chroot" inside which runs a specific version of Scientific Linux along with all the libraries necessary for the correct middleware and user applications work. Hence, from the user point of view, the environment is Scientific Linux 5.5, while from the operative core system point of view it is SLES 10sp2. As an example of the benefits allowed by this kind of installation we can report a couple of cases:

– A bug in the AMD processors of the Cluster (AMD erratum 298) caused the effect to half the performances of the WN. The solution has been identified in the creation of an ad-hoc kernel, operation that has been possible and relatively simple thanks to the decoupling.
– Another case is the updating of the middleware and user libraries. With the use of this decoupling mechanism we can rollback and

extend to all the WN of the Cluster simply unpacking a tar.gz. As far as the performances is concerned, the "chroot" mechanism does not introduce any overhead, while this happens in the case of real virtualization solutions. We proved it running SPEC benchmark inside and outside the "chroot" environment obtaining very similar results.

The fabric of the Cluster has been realized using hardware based on Cisco components (switch SFS 7012, HCA SFS HCA320 A1 with Mellanox technology MT25204 chipset) managed by a software layer based on Open Fabric version OFED-1.5.1 that has a better and more continuous development.

4.2 Storage

An important objective of this project is to ease the data management [29]. The choice of the Grid storage services made available to the Cluster has been inherited from the existing Data Center infrastructure, which is based on the StoRM [30] Storage Element. The back-end of StoRM is composed by a couple of Enterprise Class storage system Data Direct Network S2A9900, for about 1PB of RAW disk space, with GPFS file system.

The file system is accessible from the cluster WNs via POSIX and from the Storage Element using the Storage Resource Manager (SRM) protocol (see Fig. 7).

The cluster leverages on the IB connection to increase the file system performance using the IP over IB protocol.

Parallel jobs have typically a long execution time (several days), while elaborating large amount of data-sets (>1 GB). In the standard Grid architecture such jobs need to spend considerable amount of time uploading the input data-set from the user's storage area to the WNs and, at the end of the calculation, from the WNs back to the user's storage area. The data transfer involves typically a single process, while other allocated cores remain unused. Moreover, parallel jobs often require a CPU time larger than the maximum time allowed by the queue, so the user, in principle, has to move checkpoint data-sets forward and backward several times. To ease up this problem we have exploited the possibility that, on parallel clusters, the same physical storage can be shared between WNs and Storage Elements. On the Cluster a storage space ("chkPoint") is shared for reading and writing between WNs and the Storage Element.

The job work-flow on the Cluster is the following:

- Input data-sets are uploaded from the user's SE to the resource SE via GridFTP or, in case of small amount of data, they are shipped along with the InputSandbox.
- The running job writes outputs (standard output, standard error and output data) to the "ChkPoint" Area.
- The user can monitor the status of the running job by perusing the output files via SRM requests.
- At the job completion the user can retrieve the output files via SRM, or continue the calculation on the same data-set through a subsequent submission performed on the cluster.

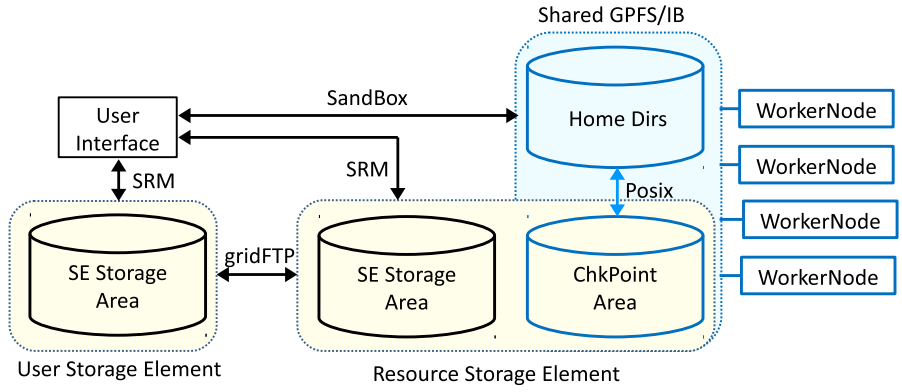

Fig. 7 Cluster storage architecture

4.3 Queues Organization

The Cluster is integrated in a wider Grid site that must provide resources also for serial jobs. The result is the fact that the biggest part of served jobs is of the serial type, for which there is no granularity issue: 1 job = 1 CPU (in reality 1 Core) and it is not important where the specific core is allocated. For parallel jobs this is not the case. For this reason in the CE, at the middleware level, a set of parameters have been introduced to allow users to specify their requests in terms of distribution of the used cores. These modifications to the middleware were included in the experimental version of the middleware we used for the Cluster. Now these modifications are a regular part of the normal middleware distribution (see next section).

This issue has two faces and can be viewed from two different point of view. From the user perspective is useful/necessary to know the granularity of the available systems and to be able to specify requests on the allocation of the Job Slots: we know that some types of applications can benefit from having "near" Job Slots (to use near core) and in other cases this is less important. So the user is happy for the opportunity to submit specific requests to the middleware.

From the site administrator point of view this can lead to inefficiencies: for instance it can happen that cores remain free due to the reservation by a job waiting the availability of other requested cores, impeding to maintain the cluster fully loaded. The experience matured in the last years with our cluster showed evidence of these issues:

- The coexistence of serial and parallel jobs on shared resources is not an easy issue for the dimension of our cluster. The best way to fill the holes of free cores would be the "short queue" but generally it doesn't work properly because it requests the collaboration by the users that, in our experience, not need this kind of resources.
- The second best choice to fill the unallocated cores is to allow the execution of serial jobs submitted by communities different from the main one. But in this case the jobs, not having specific requests for their collocation, can easily saturate the available resources making difficult the start of parallel jobs.
- The best solution would be the possibility to manage the "packing" of the serial jobs on the same Worker Node. This option is not possible in LSF (at least is not available in the version we are using), so that the only way to obtain this result is the migration of running jobs. We are exploring this solution but there are problems in the interaction with the middleware.
- Mechanisms like the Job Slot reservation and the back fill are used. They allow to mitigate the problems and to increase the exploitation of the available resources but they have the side effects of leading to long wait times for the execution of parallel jobs and/or to resource starvation.

4.4 Granularity and Multicore Support

The cluster model supported by the gLite middleware didn't consider the different levels of communication types among cores in the modern multi-core systems. Figure 8 displays the architecture of a modern cluster with multicore biprocessor nodes. In this example the inter-process communication performance (latency and bandwidth) depends on the position of the processes with respect to the cores. We measured the communication round-trip time (see Fig. 8, bottom left) and bandwidth (see Fig. 8, bottom right) using the NetPipe [31] utility. The bandwidth performance makes evidence of the Level 2 cache effect at a packet size of about 8 KBytes and the Level 3 cache effect for packet sizes up to 2 MBytes.

Grid middlewares should support multithreaded and hybrid OpenMP/MPI parallelism. They should also support the CPU and memory affinity control for the exploitation of cache effects and the regulation of the DDR access in order to limit the bottleneck introduced by the inter-processor links (QPI or HT).

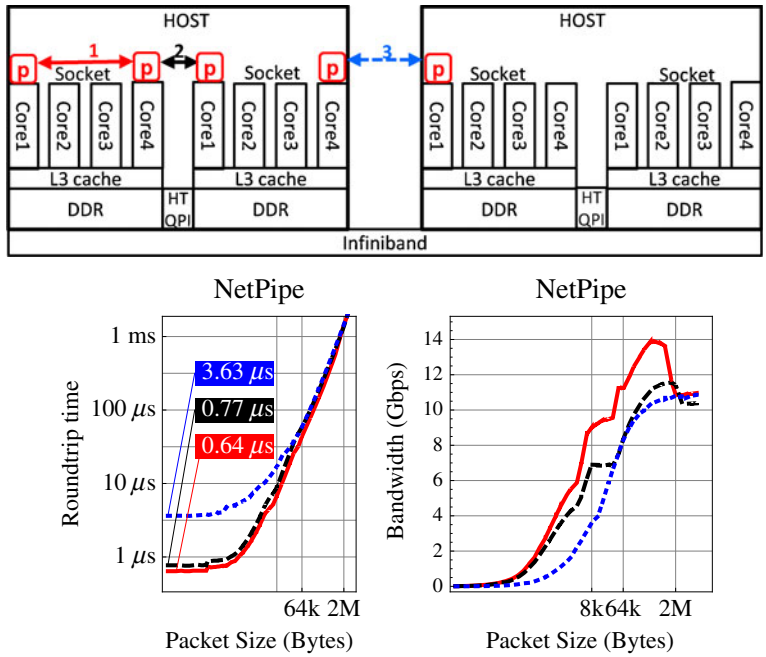

Fig. 8 Communication types patterns in multicore systems (*top*). Different communication patterns correspond to different performances. The actual performances, as measured using the Netpipe tool for the different communication patterns, are shown in the bottom-left box (round trip time) and bottom-right box (Bandwidth). *Continuous red line* for intra-socket communications (*1*), *black-dashed line* for intra-node communications (*2*) and *blue-dotted line* for extra-node infiniband communications (*3*)

The MPI-WG recommendation document, previously mentioned, introduces three new JDL attributes for the Granularity support (Fig. 9). These attributes allow users to specify:

- **SMPGranularity**: This value determines the minimum number of cores that should be allocated on any host.
- **WholeNodes**: Whether whole nodes (all the core present on a single node) should be exclusively allocated for the job.
- **HostNumber**: How many nodes should be used.

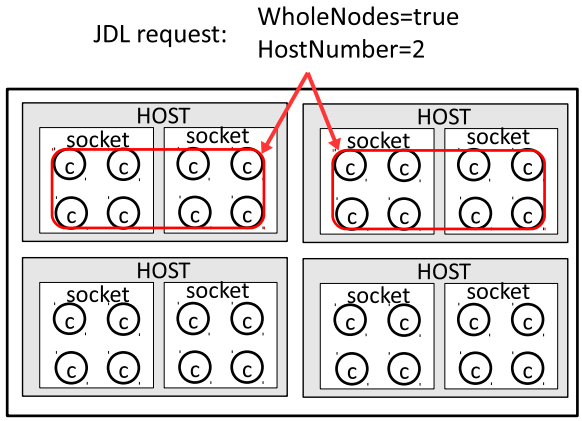

Fig. 9 Grid cluster model with granularity support

The use of these new attributes will allow users to better specify the execution environment needed by the application. In Fig. 10 are shown some specific usage examples.

These attributes were not implemented yet in EGEE so we started a collaboration with the gLite middleware developers aiming at the development and testing of a provisional patch for CREAM CE [32], including a preliminary support for the new attributes.

```
# Pure Multi thread job
WholeNodes = True;
SMPGranularity = 8;
Hostnumber=1;
```

```
# Hybrid Job
WholeNodes = True;
HostNumber = 4;
SMPGranularity = 8;
```

```
# MPI job with one process per node.
HostNumber = 16;
CPUNumber = 16;
```

Fig. 10 Examples of use of the new Granularity attributes. *Top example*: request for a multi-threaded job requiring a whole node with a minimum of 8 cores. *Middle example*: request of 4 exclusive nodes with at least 8 cores per node. *Bottom example*: request for an MPI job spread over as many nodes as possible

This patch is based on the ability of the CREAM CE to forward specific requirements (CeRequirements) [32], written by the user in the JDL, directly to the CE and there processed to properly instruct the local batch system. CeRequirements made possible the introduction of the attributes (with a different syntax, but with an equivalent semantics) on the production environment in a transparent way, preserving the standard functionality of the service. In Fig. 11 are reported the examples shown in Fig. 10 using the provisional syntax.

Unlike the JDL Requirements, CeRequirements don't involve the match-making process, so the attributes can be used only by means of direct submissions to the resource. This implies that the end-point must be explicitly specified in the Requirements. Of course this strategy doesn't scale but, in this first phase of the project where it exists only a single HPC resource, it is a reasonable choice.

4.5 MPI-Start

In many cases parallel jobs require to allocate Job Slots on the minor possible number of nodes to minimize the communication overhead. The idea is to bind each MPI process (and its threads) to a specific subset of processing resources (cores, sockets, etc.) in order to inhibit excessive process movement away from its data (L3 cache, physical memory bank, ...) and to improve inter-process communications efficiency.

```
# Pure Multi thread job
CeRequirements=
" hostsmpsize==8 && WholeNodes=\"true\" && && Hostnumber=1";
```

```
# Hybrid Job
CeRequirements=
"hostsmpsize==8 &&  WholeNodes=\"true\" && Hostnumber=4";
```

```
# MPI job with one process per node.
CeRequirements="Hostnumber=16";
CPUNumber = 16;
```

Fig. 11 Multicore support in gLite with the provisional granularity patch. Here are reported the same examples of Fig. 10 using the provisional syntax

At the start of the project, there was no standardized way of performing these optimizations and users needed to implement process binding on the starting shell script, in a way dependent on the target execution cluster. This approach is not suited to the whole idea of the Grid and needed to be overcome. For this reason, in collaboration with the MPI-Start developers, an experimental version of the tool has been released and installed on the Theophys cluster, providing the support for openMP programming and CPU affinity. By using this experimental release of MPI-Start, it is possible to start MPI processes at CPU socket level, preserving an affinity lock between processes and memory.

The importance of this tuning can be seen on the testing results shown in Table 2: performance decreases when increasing the number of threads per node ('-psocket' and '-pnode') from 4 to 8, due to the fact that, in the latter case, memory affinity between processes and data is not ensured. The last two lines show that, if MPI processes are not forced to completely fill the WNs ("whole nodes" is not set), the computation results have large variation (we reported the minimum and maximum measured time) and they are always greater than in the other case (first line marked with '-pcore').

The MPI-Start tool allows a flexible management of the execution environment for mixed MPI/openMP jobs, through the following command line options: (i) `-pcore`, one MPI process per CPU core; (ii) `-psocket`, one MPI process per CPU socket; (iii) `-pnode`, one MPI process per node. In Fig. 12 is reported the example of a hybrid MPI/openMP job that launches one MPI process per CPU socket and one openMP thread per core.

4.6 Authentication and Authorization

The gLite security architecture is based on well established work in the Grid community. On the authentication side a credential storage ensures proper security of credentials while proxy X.509 certificates enable single sign-on. The authorization functionality is obtained with the help of the VOMS [33] service, which provides support

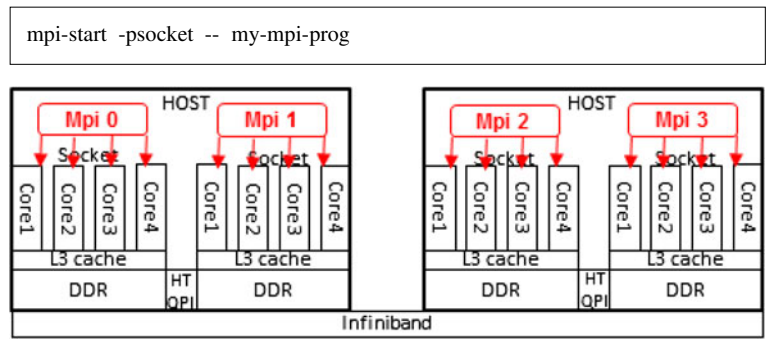

Fig. 12 This is an example of how to lunch a hybrid OpenMP/MPI program using MPI-Start with four MPI instances (each with 4 threads) on two computing nodes

for Roles and Groups membership. In a Grid infrastructure such as EGEE/EGI, where serial and parallel jobs coexist in the same environment, we need a way to avoid standard serial jobs being dispatched to parallel queues by the WMS match making process.

For this purpose we introduced the "**Role=Parallel**" which has to be assigned to HPC users. According to this method, the command to start a parallel session on the Theophys cluster is obtained by adding the option "**-voms theophys:/theophys/Role=parallel**", to the "**voms-proxy-init**" command.

To submit a serial job, the parallel role must not be specified, i.e., one should give the option "**-voms theophys:/theophys**".

This method is not user friendly since users have to generate different kind of VOMS proxies depending on the kind of jobs they need to execute, but the use of VOMS extensions (Role or Group) was the simplest way to implement this feature in the current Grid architecture. The problem of keeping generic jobs out of a given resource is common to other special resources, such as nodes equipped with GP-GPU cards or large amount of memory. This issue should and will be investigated by middleware designers in order to elaborate a common solution.

5 Parallel Applications

We report here the functionality test and the overall performance of the system, using two freely available scientific codes used by some of the groups that use the system.

The two production codes are the "Chroma" [34] code, which performs Monte Carlo simulation of lattice QCD, and the Einstein Toolkit [35–37], which performs the time evolution of matter coupled to the Einstein's equations (General Relativistic Hydrodynamics). The first one, the "Chroma" code, needs parallelization for reducing computation time on a small data-set. The second one, the "Einstein Toolkit", needs parallelization in order to distribute, among nodes, the memory allocation of evolved variables. While the first one is a pure-MPI application, the second one has a mixed MPI/OpenMP parallelization.

5.1 The Chroma Library

The first test is based on the freely available USQCD collaboration "Chroma" library. It refers to Hybrid Monte Carlo simulations of a Pure SU(3) Lattice Gauge Theory, using two different lattice Grid sizes ($32 \times 32 \times 32 \times 4$ and $16 \times 16 \times 16 \times 16$) and performing 2000 sweeps. The total allocated memory is very small: 36 MBytes, for a Grid size of $32 \times 32 \times 32 \times 4$, or 16 MBytes, when the Grid size is $16 \times 16 \times 16 \times 16$.

The parallelization is executed by distributing the data Grid on N-processes that communicate to each other using the MPI library. The timing has been measured by executing the simulation on 1, 2, 4, 8, 16, 32 nodes, i.e. on 8, 16, 32, 64, 128, 256 MPI processes, respectively. All the runs are executed with MPI processes bound to their core, by imposing memory affinity. The results are reported in Table 2, and in Fig. 13 is reported a plot of the efficiency against the number of nodes.

Table 2 Total execution time for two data Grid size

# nodes	1	2	4	8	16	32
# CPUs	8	16	32	64	128	256
SU(3) on a $32 \times 32 \times 32 \times 4$ Grid						
Time (min)	287	140	59.8	27.1	14.2	9.00
Efficiency	1.00	1.03	1.20	1.32	1.27	1.00
SU(3) on a $16 \times 16 \times 16 \times 16$ Grid						
Time (min)	167	70.5	31.1	15.4	10.6	6.57
Efficiency	1.00	1.18	1.34	1.35	0.98	0.79

The reported efficiency is computed by multiplying the execution time by the number of execution nodes, and scaled with respect to the execution on a single node. #CPUs is the number of MPI processes

5.2 The Einstein Toolkit

The test is performed by simulating the evolution of a stable general relativistic TOV-Star model using the Einstein Toolkit consortium code. We evolve the system on a cubic multi-Grid mesh with five levels of refinement (each of local size $40 \times 40 \times 40$) and we perform 800 time evolution steps. The total allocated memory for a scalar evolution is of the order of 1 GBytes. Since this application supports hybrid parallelization, it has given us the possibility to test the behavior of our system under different distributions of the computation between MPI processes and openMP threads. The results are reported in Table 3. In Fig. 14 is reported the total simulation time.

Table 3 Total execution time (in minutes) for a test evolution of a Relativistic Star on the same Grid, as a function of the number of computing cores used

# nodes	1	2	4	8	16
# CPUs	8	16	32	64	128
-pcore	127.2	79.6	57.6	49.1	47.2
-psocket	129.3	78.9	59.8	46.6	43.3
-pnode	172.5	123.5	81.1	66.3	52.9
No whole node allocation (pure MPI) and no affinity					
Min	135.1	90.8	60.7	52.8	101.9
Max	154.3	102.6	73.5	79.8	127.6

Times in the same column correspond to the same number of CPUs allocated for each job. "-pcore", "-psocket", "-pnode", correspond to the allocation of 1 MPI process (with 1 thread) for each core, 1 MPI process (with 4 threads) for each CPU socket and 1 MPI process (with 8 threads) per node, respectively. The last two lines refer to the minimum and maximum execution time obtained when the whole-node allocation is relaxed

6 Conclusions and Future Works

Even though the project is based on provisional components, the Cluster is fully operative and it is producing scientific results. The project

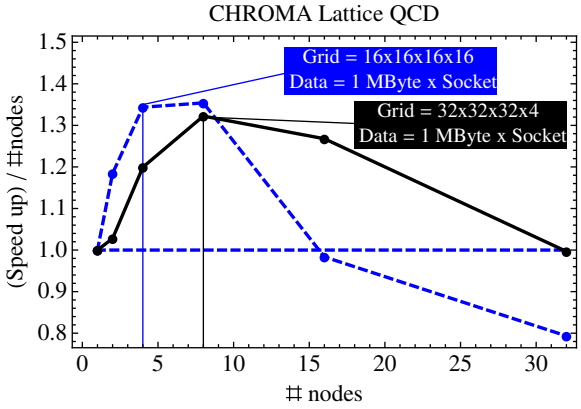

Fig. 13 Plot of the efficiency of the parallel execution as a function of the # s of nodes for the Hybrid Mote Carlo simulation of the pure gauge SU(3) Gauge theory performed using the Chroma library. The required simulation times are reported in Table 2. Please note that the maximum efficiency is reached when the data size assigned to a group of 4 MPI processes, which are executed on the same CPU socket, is approximately equal to the size of the L3 cache

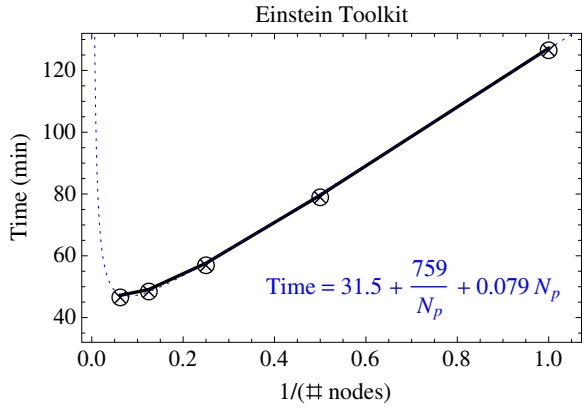

Fig. 14 Total execution time (in minutes) for the evolution of a Relativistic star for pure MPI parallelization as a function of the number of nodes on which the execution take place. Each MPI process is constrained on a fixed CPU core and all the cores on the same node are allocated for the execution. The time values are reported in Table 3 in the line labeled by "-pcore". The *dotted blue line* shows the fitted scaling law

achieved two important results for the Theophys community:

- the deployment of a centralized resource for parallel applications that shares the same platform with the other Grid resources available for serial jobs
- the design of a global infrastructure that will be able to fulfill all the computational needs of the community.

Besides these two primary objectives that were achieved for the Theophys VO, this work has produced other contributions to the whole computational community. A step towards a better support in gLite/EMI for jobs needing to exploit multi-socket and multi-core architectures. To show that, even if users generally consider (see Section 1 and [6]) the Grid not ready for parallel computations, the adoption of the new features described here enables a flexible and easy porting of HPC applications on the Grid.

After the experimental activity described here, the Granularity attributes have been included in the EMI middleware, which is now under deployment inside the EGI infrastructure. In the second phase of the project we intend to deploy a HTC/HPC infrastructure based on the new EMI middleware by involving a set of selected sites and users from different scientific communities. Beside the tests to verify resilience and scalability of the platform, which will be performed in close collaboration between users and sites administrators, a special attention will be dedicated to the usability, through the introduction of a Web portal with HPC support.

Acknowledgements We would like to thank: M. Sgaravatto, S. Monforte and A. Gianelle (INFN, Italy) for their efforts in the development of the CE CREAM patch supporting multicore architectures; E. Fernandez (IFCA, España) for the development of the support of CPU affinity and openMP in MPI-Start; and A. Feo for the testing activity with the Lattice QCD program "Chroma".

References

1. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: enabling scalable virtual organizations. Int. J. Supercomput. Appl. **15**(3), 200–222 (2001)
2. Ferrari, T., Luciano Gaido, L.: Resources and services of the EGEE production infrastructure. J. Grid Computing **9**(2), 119–133 (2011)
3. Dooley, R., Milfeld, K., Guiang, C., Pamidighantam S., Allen, G.: From proposal to production: lessons learned developing the computational chemistry Grid cyberinfrastructure. J. Grid Computing **4**(2), 195–208 (2006)
4. Becciani, U., Antonuccio-Delogu, V., Costa, A., Petta, C.: Cosmological simulations and data exploration: a testcase on the usage of Grid infrastructure. J. Grid Computing **10**(2), 265–277 (2012)
5. Vilotte, J.P., Moguilny, G.: Earth science: requirements and experiences with use of MPI in EGEE. In: EGEE09 Conference, Barcelona, 21–25 September 2009
6. Engelberts, J.: Towards a robust and userfriendly MPI functionality on the EGEE Grid. In: EGEE User Forum 2010, Uppsala, 12–15 April 2010. See also: http://www.Grid.ie/mpi/wiki/WorkingGroup
7. Anglano, C., Canonico, M., Guazzone, M.: The ShareGrid peer-to-peer desktop Grid: infrastructure, applications and performance evaluation. J. Grid Computing **8**(4), 543–570 (2010)
8. IGI accounting portal: http://accounting.egi.eu (2010)
9. Bodin, F., Boucaud, P., Cabibbo, N., Cascino, G., Calvayrac, F., Della Morte, M., Del Re, A., De Pietri, et al.: APE computers—past, present and future. Comput. Phys. Commun. **147**(1–2), 402–409 (2002)
10. Foster, I.: Globus toolkit version 4: software for service-oriented systems. In: IFIP International Conference on Network and Parallel Computing, LNCS 3779, pp. 2–13. Springer, Berlin (2005)
11. Laure, E., Fisher, S.M., Frohner, A., Grandi, C., Kunszt, P., Krenek, A., Mulmo, O., Pacini, F., Prelz, F., White, J., Barroso, M., Buncic, P., Hemmer, F., Di Meglio, A., Edlund, A.: Programming the Grid with gLite. Comput. Methods Sci. Technol. **12**(1), 33–45 (2006)
12. Ellert, M., et al.: Advanced resource connector middleware for lightweight computational Grids. Future Gener. Comput. Syst. **23**, 219–240 (2007)
13. Ernst, M., Fuhrmann, P., Mkrtchyan, T., Bakken, J., Fisk, I., Perelmutov, T., Petravick, D.: Managed data storage and data access services for data Grids. In: Computing in High Energy Physics and Nuclear Physics 2004 (CHEP04), Interlaken, Switzerland, p. 665, 27 Sept–1 Oct 2004
14. Streit, A., Bala, P., Beck-Ratzka, A., Benedyczak, K., Bergmann, S., Breu, R., Daivandy, J. M. , Demuth, B., Eifer, A. Giesler, A., Hagemeier, B., Holl, S., Huber, V., Lamla, N., Mallmann, D., Memon, A.S., Memon, M.S., Rambadt, M., Riedel, M., Romberg, M., Schuller, B., Schlauch, T., Schreiber, A., Soddemann, T., Ziegler, W.: UNICORE 6—recent and future advancements. Ann. Télécommun. **65**(11–12), 757–762 (2010)
15. Fuhrmann, P.: EMI, the introduction. In: Proceeding of CHEP 2010, Taipei, 18 October 2010
16. Fernández, E.: A unified user experience for MPI jobs in EMI. In: EGI User Forum 2011, Vilnius, 11 April 2011

17. Anjomshoaa, A., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A.: Job Submission Description Language (JSDL) specification, version 1.0. White Paper, 7 November 2005
18. Schuller, B.: MPI in UNICORE. In: EGI Technical Forum 2010, Amsterdam, 15 September 2010
19. Dichev, K., Stork, S., Keller R., Fernández, E.: MPI support on the Grid. Comput. Inform. **27**(2), 213–222 (2008)
20. Schuller, B., Konya, B., Konstantinov, A., Sgaravatto, M., Zangrando, L.: EMI-ES, a common interface to ARC, gLite and UNICORE computing elements. In: EGI User Forum, Vilnius, 11 April 2011
21. Carminati, F., Templon, J., et al.: Common use cases for a HEP common application layer. White Paper, LHC-SC2-20-2002
22. European Grid Infrastructure: An integrated sustainable Pan-European infrastrucute for researchers in Europe (EGI-InSPIRE). White Paper, 18 April 2011. Document Link: https://documents.egi.eu/document/201
23. XSEDE Production Baseline: Service provider software and services. White paper, released on 22 February 2012
24. Altunay, M., Avery, P., Blackburn, K., Bockelman, B., Ernst, M., et al.: A science driven production cyberinfrastructure—the open science Grid. J. Grid Computing **9**(2), 201–218 (2011)
25. Berg, A.: PRACE distributed infrastructure services and evolution. In: EGI Community Forum 2012, Garching, 28 March 2012
26. Gentzsch, W., Denis Girou, D., Kennedy, A., Lederer, H., Reetz, J., et al.: DEISA-distributed European infrastructure for supercomputing applications. J. Grid Computing **9**(2), 259–277 (2011)
27. Laganá, A., Costantini, A., Gervasi, O., Faginas Lago, N., Manuali, C., et al.: Compchem: progress towards GEMS a Grid empowered molecular simulator and beyond. J. Grid Computing **8**(4), 571–586 (2010)
28. Borgdorff, J., Falcone, J., Lorenz, E., Chopard, B., Hoekstra, A.: A principled approach to distributed multiscale computing, from formalization to execution. In: Proceedings of The Seventh IEEE International Conference on e-Science Workshops, Stockholm, Sweden, 5–8 December 2011, pp. 97–104. IEEE Computer Society, Washington, DC (2011)
29. Frohner, A., Baud, J.P., Garcia Rioja, R.M., Grosdidier, G., Mollon, R., Smith D., Tedesco, P.: Data management in EGEE. In: Journal of Physics, Conference Series 219 (2010)
30. Zappi, R., Magnoni, L., Donno, F., Ghiselli, A.: StoRM: Grid middleware for disk resource management. In: Proceedings of Computing in High-Energy Physics, 27 Sept–1 Oct 2004 Interlaken, Switzerland, (CHEP04), pp. 1238–1241 (2005)
31. Turner, D., Oline, A., Chen, X., Benjegerdes, T.: Integrating new capabilities into NetPIPE. Lect. Notes Comput. Sci. **2840**, 37–44 (2003)
32. Aiftimiei, C., Andreetto, P., Bertocco, S., Dalla Fina, S., Alvise Dorigo, A., Frizziero, E., Gianelle, A., Marzolla, M., Mazzucato, M., Sgaravatto, M., Traldi S., Zangrando, L.: Design and implementation of the gLite CREAM job management service. Future Gener. Comput. Syst. **26**(4), 654–667 (2010)
33. Alfieri, R., Cecchini, R., Ciaschini, V., dell'Agnello, L., Frohner, A., Lorentey, K., Spataro F.: From Gridmap-file to VOMS: managing authorization in a Grid environment. Future Gener. Comput. Syst. **21**(4), 549–558 (2005)
34. Edwards, R.G., (LHPC Collaboration), Joó, B., (UKQCD Collaboration): The chroma software system for lattice QCD. arXiv:hep-lat/0409003. In: Proceedings of the 22nd International Symposium for Lattice Field Theory (Lattice2004), Nucl. Phys. **B140** (Proc. Suppl), 832 (2005). See also: http://usqcd.jlab.org/usqcd-docs/chroma/
35. Goodale, T., et al.: The cactus framework and toolkit: design and applications. In: Vector and Parallel Processing—VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science. Springer, Berlin (2003).
36. Schnetter, E., Hawley, S.H., Hawke, I.: Evolutions in 3-D numerical relativity using fixed mesh refinement. Class. Quantum Grav. **21**, 1465–1488 (2004)
37. Baiotti, L., Hawke, I., Montero, P.J., Löffler, F., Rezzolla, L., Stergioulas, N., Font, J.A., Seidel, E.: Three-dimensional relativistic simulations of rotating neutron star collapse to a Kerr black hole. Phys. Rev. D **71**, 024035 (2005). See also: http://einsteintoolkit.org/