

CONTRACT NUMBER 508830

DEISA
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
SUPERCOMPUTING APPLICATIONS**

European Community Sixth Framework Programme
RESEARCH INFRASTRUCTURES
Integrated Infrastructure Initiative

SA4 Mid-term Activity Report

Deliverable ID: DEISA-DSA4-3.2
Due date: October, 30, 2005
Actual delivery date: November, 25, 2005
Lead contractor for this deliverable: IDRIS-CNRS, France

Project start date: May 1st, 2004
Duration: 4 years

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Content

Table of Content.....2

1. Executive Summary.....3

2. Introduction.....4

3. Common Production Environment.....5

 3.1 CPE software stack.....5

 3.2 Interface to access the software.....5

4. Software monitoring for the Common Production Environment.....8

5. User Documentation.....13

6. Support of scientific Joint Research Activities.....14

 6.1 JRA 1 (Material Sciences).....14

 6.2 JRA 2 (Cosmology Applications).....15

 6.3 JRA 3 (Fusion Research).....15

 6.4 JRA 4 (Life Sciences).....15

 6.5 JRA 5 (Industrial CFD).....16

 6.6 JRA 6 (Coupled Applications).....16

 6.7 DECI projects.....17

7. Other activities.....18

 7.1 Trouble Ticket System.....18

 7.2 DEISA Cluster Resource Management Package.....18

8. References and Applicable Documents.....19

9. List of Acronyms and Abbreviations.....20

1. Executive Summary

The main objective of this service activity is to perform all the actions required to allow scientific users to adopt and use the DEISA supercomputing infrastructure. This is largely done by providing and maintaining a *Common Production Environment* on all the platforms of the infrastructure, and also providing documentation on its usage, specialised support to help the porting and optimisation of applications and a decentralised *Help Desk* service.

This document gives a detailed overview of all the tasks achieved or currently undertaken in the *User Support and Applications Service Activity (SA4)* during the reporting period. This constitutes the report of the mid-term annual activities.

This document is publicly available.

2. Introduction

During the last six months, the *Applications and User Support Service* Activity has mainly concentrated its efforts on the following tasks:

- improvements in the implementation of the *Common Production Environment*, especially of the *Modules* interface which was heavily rewritten internally to enhance its generality and portability, both between equivalent platforms of different partners, between different platforms and between the DEISA and the local environments
- the full adaptation and deployment of the framework to monitor this *Common Production Environment* on all the core sites
- updates to the DEISA user documentation, according to both changes made in the user interface and to feedback received after the release of the initial versions, thereby improving and clarifying some issues
- provision of help to the users of the scientific *Joint Research Activities*.

3. Common Production Environment

The unified *Common Production Environment* (CPE), as with all distributed infrastructures of this kind, is a major feature of DEISA which defines a coherent set of software accessible on the various sites of the infrastructure. It offers both a common interface to the users, independent of the target platform really used, and the ability to migrate the jobs between different supercomputers of the same architecture. Of course, the level of coherence is not the same everywhere in the infrastructure, ranging from very high inside each subgroup of homogeneous computers to a lower level across the other subgroups.

Three main components define the CPE:

- a coherent set of software packages,
- a uniform interface to access the software,
- a framework to monitor the software (this will be described in section 4).

3.1 CPE software stack

The CPE is divided into six categories: environment (which does not include any software, but defines the environment needed by a DEISA user), shells, compilers, libraries, tools and applications. The initial CPE is the preliminary set of software components and is based on the basic software environment of every supercomputer (compilers, generic tools, libraries provided by the manufacturer, etc.) and the requirements of the applications used in the scientific JRAs. This initial CPE has been installed on the five core sites which first join the infrastructure (CINECA, CSC, FZJ, IDRIS, and RZG). The current status on these sites, as reported dynamically by the monitoring framework, is shown in Figures 1 to 3. As can be seen, only very few components are missing from place to place, mostly due to some licences not yet available for some commercial software, however, all the major components are in place.

It must be emphasized that this task has required a large amount of effort by all the sites, not only to install new software not originally available, but also in many cases to re-install existing ones to fulfil the specific DEISA requirements (specific release of the product, 64-bit version, etc.)

3.2 Interface to access the software

Having the required software installed and making them available to users is not enough. The local system administration policies define different places to install the software, especially those not included in the standard system installation, and a major requirement of DEISA is that a common interface hides these differences. Additionally, this is also a critical requirement for job migration between homogeneous computers.

As reported earlier [2, 4], the *Modules* [14] public domain tool was chosen to fulfil such a requirement, allowing the provision of a dedicated portable interface to each component of the CPE, based on the so-called *modulefiles*. Since the early autumn of 2004, various subsets of these *modulefiles* for the initial CPE on AIX systems have been developed and distributed by IDRIS, who were in charge of this task, and then evaluated and tested by the partners. Nevertheless, during the last few months major work has continued in this area, concentrating on two particular points. The

interfaces of all the components of the CPE have now been developed, but also modifications have been performed according to the various feedback received from the partners and to comply with some of their new expectations. The modifications have not affected the user interface (except for a minor change when several versions of the same component must be managed), but they have led to important internal rewriting. The major reason for all these changes and the effort required to make them is that it appears to improve greatly both the generality and portability of the DEISA Modules environment.

Firstly, as this is a rather heavy task, it appears better to try to reuse as much as possible of the work already performed for AIX systems for the porting and adaptation of the Modules environment to the operating systems of the other platforms which must soon be integrated into the DEISA infrastructure. Although some care was taken of portability issues since the early stages of development, largely by adapting and testing the first initial subset of the *modulefiles* on both an NEC Super-UX system and on an SGI Altix, it appeared that new efforts must be made in this area.

Secondly, although some sites already used the Modules environment for their local users it also appeared that several of the others plan to do the same in the near future. That is to say that the Modules environment will become a critical tool for most of the sites, illustrating how work performed in DEISA can have positive benefits outside of the project.

More precisely:

- The initialization process has been completely changed to separate the local and DEISA Modules environments for those sites which already use it for their local users.
- Each *modulefile* is now split between a generic and a local part. The local part only contains those definitions which are specific to a particular site (mainly the location of the files for the relevant software, which is completely dependent on the local system administration rules for software installation). This allows easier maintenance when internal technical changes are made in the writing of the modulefiles.
- The management of several versions of the same software now follows the standard way proposed by the *Modules* tool, which allows for full compatibility with the evolution of the tool itself. Recent changes to the Modules tool had illustrated some drawbacks with the preceding convention that we used (the two sites which used the old convention in their local installation have accepted to switch to the new one.)
- A higher level of customization has been recently introduced to allow, for each site, the ability to share the same DEISA *modulefiles* between the DEISA environment and the local one, thereby avoiding the duplication and maintenance of two different sets to interface the same software.

In parallel, the work to port the DEISA Modules environment on non-AIX platforms has already started. LRZ has already ported the AIX set of *modulefiles* for the SGI Altix system, with, of course, the required adaptations for the specific software of this system, especially the compilers and the scientific libraries provided by the manufacturer. They have also adapted it to the internal changes made in the last months, as described above, for compatibility purposes amongst systems. A technical meeting was organized in October to exchange the experiences and

feedback between the developers and in particular to describe and explain the choices made and the work to be done to the other partners (BSC and HLRS), which will now have to port the Modules environment to their own IBM Linux and NEC Super-UX systems.

To help to distribute all the required configuration files and to allow each partner to easily know the changes made since their last update, a CVS server has been set up and configured at IDRIS. A local area is managed by each partner in charge of the development and maintenance of the Modules environment on one particular operating system and allows the other partners to download only the modified files when updating their local installation.

4. Software monitoring for the Common Production Environment

It is an absolute necessity that each centre's staff have both an up to date overview and a detailed status of the software environment, especially in a distributed infrastructure like DEISA. The monitoring of such a distributed software environment has become a requirement in all important grid projects, even if the requirements are rather high and sometimes difficult to handle, especially in a heterogeneous context. The development of such a general framework is a huge task by itself.

The first feature required in such tools is the ability to verify the accessibility of the various software packages installed on the different computers, the expected version level of these packages and also, when possible, their current behaviour. It is also important that these tools allow the verification of the status after software upgrades and offer the administrators and the user support services an updated view of what is installed and available on the various computers of the distributed infrastructure, alerting them of possible deficient components.

As explained in the deliverable D-SA4-3 [4], the INCA tool (*Test Harness and Reporting Framework*) [13], developed by the San Diego Supercomputer Center and the Argonne National Laboratory for the TeraGrid project [17], has been chosen after an evaluation process. Although it was developed for the needs of this particular project, it was developed from the beginning with the idea in mind of not limiting it to the specific requirements of TeraGrid, but allowing for it to be easily adapted to other grid infrastructures.

INCA is specifically designed to periodically run a collection of validation scripts, called *reporters*, with the purpose of collecting two different kinds of information:

- the version of the software installed (*version reporters*),
- the availability and the correct operation of this software (*unit reporters*).

The collected information is then cached on the INCA server, and can be archived to produce a historical representation of the status of the resources of a grid.

The architecture of INCA is composed of three levels:

1. a centralized *server*, which itself includes a centralized controller, named the *collector*, and a *depot*
2. *clients*, installed on each resource to be validated, which themselves include a *distributed controller* and a *reporter suite*
3. *data consumers*, such as the Web-based one, to display the results.

The evaluation of the INCA framework was done at CINECA between December 2004 and the beginning of the spring of 2005, with adaptations required for both the interface provided by the Modules tool (as INCA originally used the SoftEnv tool [16]) and for a small subset of our CPE. After this time period, very positive conclusions were made about the suitability of the INCA framework to our needs and the relative ease with which it could be adapted to our context. A meeting was organized at the end of April to explain to all the partners the details of the INCA architecture, the work achieved and how to set up the client on their own systems. During the following months a full deployment has been performed on the core sites, and INCA can be considered to have been in full production since the summer. This deployment was lead by LRZ, who took charge of the setup and operation of the centralized server, particularly the depot populated by the local clients, and, with the help of several

partners, the development of the version reporters for all the components of the initial CPE. A service analogous to the one set up to distribute the updates in the Modules environment has been established to distribute the INCA reporters, using the same CVS server. Meanwhile, the development of some *unit* reporters has just begun. Unit reporters are used to check not only the existence of the expected version level of a particular software package, but that some of its major functionalities are working as planned (which may also involve other software layers too). These unit reporters will greatly improve the service given by the monitoring framework, allowing for the quick diagnosis and solution of some complex problems regarding some software packages.

Figures 1 to 3 show the current status on the core sites, as reported on the DEISA *data consumer* designed at LRZ. It shows a summary, the details for each of the monitored software packages on each platform, and gives various historical graphs of the past behaviour.



inca.deisa.org Common Production Environment (CPE) - Inca status page

[Summary view](#) | [Stack view](#) | [Status graphs](#)

Summary of DEISA CPE 1.0

Page generated by Inca: 11/21/05 16:31 CET

This page offers a summary of results for critical grid, development, and cluster tests ([view list of tests](#)). Details about a resource's test results are available by clicking on the resource name in the "Site-Resource" column of the table.

Site	Resource	Total Pass	Compilers	Libraries	Shells	Tools	Applications
CINECA	sp5	94%	100% (4 / 4)	92% (13 / 14)	100% (2 / 2)	91% (11 / 12)	100% (4 / 4)
CSC	sp4	97%	100% (4 / 4)	100% (14 / 14)	100% (2 / 2)	91% (11 / 12)	100% (4 / 4)
FZI	sp4	94%	100% (4 / 4)	100% (14 / 14)	100% (2 / 2)	83% (10 / 12)	100% (4 / 4)
IDRIS	sp4	97%	100% (4 / 4)	92% (13 / 14)	100% (2 / 2)	100% (12 / 12)	100% (4 / 4)
RZG	sp4	94%	100% (4 / 4)	100% (14 / 14)	100% (2 / 2)	83% (10 / 12)	100% (4 / 4)

Note: unknown or indeterminate results can occur if a test interacts with a batch queue. If the batch queue is busy, the test eventually times out. Time out results are marked as "Unknown" because they are not an indication of success or failure.

Figure 1: INCA Summary View

package	version	cne-sp5	csc-sp4	fzj-sp4	idr-sp4	rzg-sp4
cluster_aix_pwr4_compilers						
java	=1.4	<u>1.4.2</u>	<u>1.4.2</u>	<u>1.4.2</u>	<u>1.4.2</u>	<u>1.4.2</u>
xlC	=7.0	<u>7.0</u>	<u>7.0</u>	<u>7.0</u>	<u>7.0</u>	<u>7.0</u>
xlC	=7.0	<u>7.0</u>	<u>7.0</u>	<u>7.0</u>	<u>7.0</u>	<u>7.0</u>
xlF	=9.1	<u>9.1</u>	<u>9.1</u>	<u>9.1</u>	<u>9.1</u>	<u>9.1</u>
cluster_aix_pwr4_libraries						
blacssmp	=3	<u>3.2.0.0</u>	<u>3.1.0.1</u>	<u>3.2.0.0</u>	<u>3.2.0.0</u>	<u>3.1.0.2</u>
essl	=4	<u>4.2.0.2</u>	<u>4.1.0.1</u>	<u>4.2.0.2</u>	<u>4.2.0.2</u>	<u>4.1.0.1</u>
esslsmp	=4	<u>4.2.0.2</u>	<u>4.1.0.1</u>	<u>4.2.0.2</u>	<u>4.2.0.2</u>	<u>4.1.0.1</u>
mass	=4.1	<u>4.1.0.3</u>	<u>4.1.0.0</u>	<u>4.1.0.2</u>	<u>4.1.0.1</u>	<u>4.1.0.0</u>
pessl	=3	<u>3.2.0.0</u>	<u>3.1.0.1</u>	<u>3.2.0.0</u>	<u>3.2.0.0</u>	<u>3.1.0.2</u>
pesslsmp	=3	<u>3.2.0.0</u>	<u>3.1.0.1</u>	<u>3.2.0.0</u>	<u>3.2.0.0</u>	<u>3.1.0.2</u>
unix_applications						
cpmd	=3.9	<u>3.9.2</u>	<u>3.9.2</u>	<u>3.9.2</u>	<u>3.9.1</u>	<u>3.9.2</u>
cpmd2cube	=jan05	<u>jan05</u>	<u>jan05</u>	<u>jan05</u>	<u>jan05</u>	<u>jan05</u>
gopenmol	=2.32	<u>2.32</u>	<u>2.32</u>	<u>2.32</u>	<u>2.32</u>	<u>2.32</u>
torb	=1.22	<u>1.22</u>	<u>1.22</u>	<u>1.22</u>	<u>1.22</u>	<u>1.22</u>
unix_libraries						
blacs	=3	<u>3.2.0.0</u>	<u>3.1.0.1</u>	<u>3.2.0.0</u>	<u>3.2.0.0</u>	<u>3.1.0.2</u>
fftw	=2.1.5	<u>2.1.5</u>	<u>2.1.5</u>	<u>2.1.5</u>	<u>2.1.5</u>	<u>2.1.5</u>
hdf5	=1.6	<u>1.6.4</u>	<u>1.6.4</u>	<u>1.6.2</u>	<u>1.6.3</u>	<u>1.6.4</u>
lapack	=3.0	<u>3.0</u>	<u>3.0</u>	<u>3.0</u>	<u>3.0</u>	<u>3.0</u>
nag	=20,21	<u>20B</u>	<u>20B</u>	<u>21</u>	<u>20B</u>	<u>20B</u>
netcdf	=3	<u>3.6.0-p1</u>	<u>3.5.1</u>	<u>3.5.1</u>	<u>3.5.0</u>	<u>3.5.0</u>
scalapack	=1.7	<u>1.7.0</u>	<u>1.7.0</u>	<u>1.7.0</u>	<u>1.7.0</u>	<u>1.7.0</u>
wsmp	=4	error	<u>4.8.5</u>	<u>4.8.5</u>	error	<u>4.10.25</u>
unix_scripting						
perl	=5.8	<u>5.8.0</u>	<u>5.8.0</u>	<u>5.8.0</u>	<u>5.8.0</u>	<u>5.8.0</u>
python	=2.4	<u>2.4.1</u>	<u>2.4</u>	<u>2.2.2</u>	<u>2.4.1</u>	<u>2.4.2</u>
tcl	=8.4	<u>8.4b1</u>	<u>8.4.4</u>	<u>8.4.2</u>	<u>8.4.10</u>	<u>8.4.6</u>
tk	=8.4	<u>8.4b1</u>	<u>8.4.4</u>	<u>8.4.2</u>	<u>8.4.10</u>	<u>8.4.6</u>
unix_shells						
bash	=3	<u>3.00.0(1)</u>	<u>3.00.16(1)</u>	<u>3.00.16(1)</u>	<u>3.00.0(1)</u>	<u>3.00.0(1)</u>
tsh	=6.11	<u>6.11.00</u>	<u>6.11.00</u>	<u>6.11.00</u>	<u>6.13.00</u>	<u>6.11.00</u>
unix_tools						
emacs	=20,21	<u>21.1.1</u>	<u>21.1.1</u>	<u>21.2.2</u>	<u>21.1.1</u>	<u>20.2.1</u>
gmake	=3	<u>3.80</u>	<u>3.80</u>	<u>3.80</u>	<u>3.80</u>	<u>3.80</u>

Figure 2: INCA Stack View (excerpt for the compilers, applications and some libraries and tools)

Grid Resource Status Archive

Page generated by Inca: 11/21/05 16:42 CET

The following page can be used to query the pass/fail status history for a resource. The pass/fail status is collected every 10 minutes (via a reporter) using the code from the summary status page. To generate a graph:

- Select one or more resources.
- Select one or more test type categories. You can graph the test type categories on a single graph or individually.
- Select one or more periods of time (e.g., over the past day).

Resources	Test Type	Time Period
cne-sp5 csc-sp4 fzj-sp4 idr-sp4 rzg-sp4	Shells Tools Compilers Libraries total	day week month year
<input checked="" type="radio"/> individual graphs <input type="radio"/> single graph		

Graph

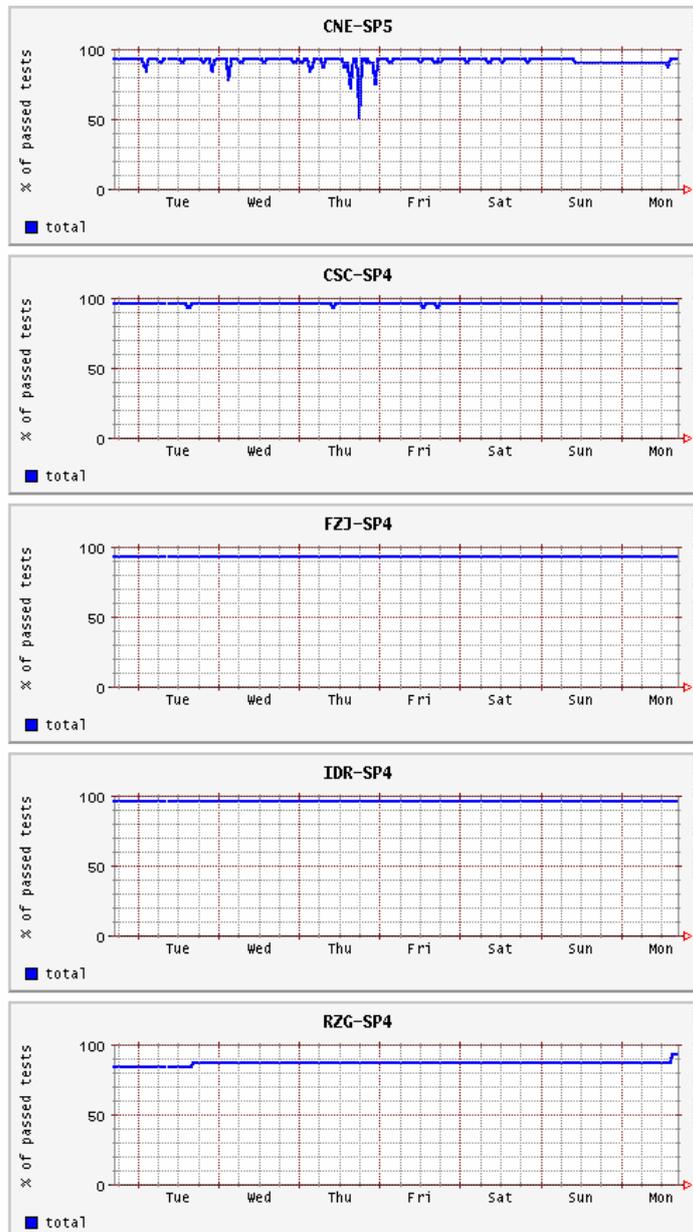


Figure 3: INCA Status graph over a week

In order to both give our feedback in a detailed way, and to re-enforce the collaboration previously achieved only through e-mail exchanges, a meeting was organized at the beginning of July with two of the main INCA developers at SDSC. This joint INCA DEISA/TeraGrid technical meeting gave us the opportunity to present our previous work carried out in the evaluation of INCA in DEISA, providing details of the problems encountered during the customisation and set-up phases, to explain our future plans, and also to provide some suggestions for the development of future versions of INCA. This was also the opportunity for the developers to present us their development roadmap, detailing the numerous evolutions and major improvements which will be introduced in the next major version, expected at the beginning of next year. These improvements include a new library to enable the development of reporters in any language, the switch to a database for the depot, a completely new security architecture, a new presentation layer, etc.

5. User Documentation

The preliminary DEISA documentation, prepared at the beginning of the year, as expected for the D-SA4-2 deliverable [3], was fully revised at the end of the summer. Both the *Primer* [12] and the FAQ [11] were updated, to introduce all the information which had changed and to take in account the feedback received after the release of the preliminary versions. In particular, all the paragraphs describing the CPE were updated, both for the software themselves and for their Modules interface; the description of the UNICORE initialisation process was heavily reviewed and enhanced, in particular for the management of the certificates; and the submission of jobs using the AIX batch scheduler was detailed in some areas to make the users aware of some known pitfalls.

6. Support of scientific Joint Research Activities

All of the projects included in these JRAs have had a direct and strong connection with one of the core partners (RZG for JRAs 1 and 3, IDRIS for JRAs 4 and 6, CINECA for JRA 5), except for JRA 2 which is connected with EPCC, but has in fact also carried out work and experiments on the systems of two of the core sites (IDRIS and RZG).

6.1 JRA 1 (Material Sciences)

The work of JRA1 was concentrated on the enhancement of the functionality of the UNICORE CPMD plug-in (input validation, pre-fill of GUI components) and the implementation of the materials science portal (see D-JRA1-3 [5]). Currently CPMD and WIEN2k are supported via this portal. The CPMD application can be invoked and controlled either via a UNICORE client or via a web-application, whereas the usage of WIEN2k is currently only supported by a UNICORE plug-in which was developed by JRA1.

With support of SA4, the recent patches of CPMD version 3.9 have been installed and tested at the DEISA sites RZG, IDRIS, CINECA, FZJ and CSC. The standard library of pseudo-potentials, provided by <http://www.cpmc.org>, is also available at all those sites for registered CPMD users.

A CPMD modulefile has been developed and integrated into the CPE. It defines environment variables allowing a consistent access to the CPMD resources:

```
$CPMD           is the path to the executable CPMD application
$CPMD_PPLIBRARY is the path of the standard PP-library
```

The CPMD application can be called by

```
$CPMD cpmd-input-file [$CPMD_PPLIBRARY] > output-file
```

Note that the name CPMD_PPLIBRARY has been chosen differently to the usual name PP_LIBRARY. The reason is that CPMD users have to be sure about what pseudo-potential data set is actually used. Although the CPMD application is provided by DEISA in a consistent manner, the consistency of additional data sets, such as the pseudo-potential library, cannot be guaranteed within the CPE. Therefore users who intend to employ the default pseudo-potential library must define the path variable PP_LIBRARY explicitly, e.g. by means of CPMD_PPLIBRARY.

In order to enable the access via UNICORE, most of the DEISA sites had to update relevant application sections in their UNICORE incarnation database (IDB).

For post-processing, the tools CPMD2CUBE (conversion of CPMD's Wannier-function files into cube files, <http://www.cpmc.org>) and gOpenMol 2.32 (post-processing for visualization) have been installed at most of the DEISA sites. Those post-processing tools are not demanding with regard to compute resources, but they reduce the amount of (raw) visualisation data significantly before this data is rendered on a remote front-end device.

The corresponding modulefiles *cpmd2cube* and *gopenmol* have also been developed with support of SA4. They define appropriate environment variables.

A modulefile for WIEN2k has also been developed and integrated into the CPE.

6.2 JRA 2 (Cosmology Applications)

During the last six months JRA 2 has mainly concentrated on work packages WP1 (Grid-enablement of Gadget for metacomputing environments), WP2 (Development of a Grid-enabled toolset) and WP3 (Grid enablement of Hydra-MPI through code migration).

The user support teams of the sites on which JRA 2 work helped by advising on the availability of different software packages on the DEISA infrastructure and helping to install critical software where necessary, namely PACX-MPI and MPICH-G2. Although this software is not part of the DEISA CPE, these packages have been installed on HPCx and several other EPCC machines for use by JRA 2. The results for using this software are reported in D-JRA2-1.2 [6].

In order to facilitate the work of this JRA, the user support teams at IDRIS and RZG also helped in setting up the necessary user accounts for JRA 2, and have also co-ordinated with their local operations teams the scheduling of jobs at each site for cross-site tests of a grid-enabled version of the cosmological application GADGET.

6.3 JRA 3 (Fusion Research)

In JRA 3, the activities during the reported period (see deliverable D-JRA3-3 [7]) have been focussed in three directions: to provide a portal for the ORB code suite, to study the scaling of the TORB code on AIX and Linux systems, and to develop the new ORB5 version of the code. The second task especially involved user support activities.

Firstly, the local adaptations of the Modules environment for the generic software included in the CPE, required for each site, were not enough here. As for JRA 1, the main application code used in this JRA, TORB, is itself integrated in the CPE, as well as a post-processing tool used by it. Consequently, the corresponding *modulefiles* for them had to be developed, and later adapted according to the internal changes made in the Modules environment, as explained in the section 3.

Secondly, the TORB code was evaluated during benchmark campaigns on several other sites, at EPCC up to 1024 processors, at ECMWF up to 2048 and at BSC up to 4096, with very satisfactory results, as reported in the deliverable previously mentioned. This work was done with coordination between the user support group of RZG and those of the other partners. The usage of the platforms was straightforward at EPCC and ECMWF, as the same kind of IBM SP4 systems were used, but more efforts were required for the MareNostrum supercomputer at BSC. This implied more interactions between the teams to port and successfully run the code on a huge number of processors on this IBM Power PC hardware platform, especially to solve the memory allocation problems encountered, and also to test the effect of various optimization options specific to the compiler.

6.4 JRA 4 (Life Sciences)

During the last six months, JRA 4 has concentrated its activities on the Genomics project (see deliverable D-JRA4-3 [8]), which itself deals with two different

applications. The first one, for the identification of new human mitochondrial proteins, is handled by a research team from INSERM. For this application the user support group at IDRIS installed the NCBI's BLAST code and the required databanks (human genome) on the SP4 supercomputer, then made efforts to optimize it. Two different aspects were heavily improved: the level of parallelization and the way in which the input and output data were managed. This allowed a decrease in the calculation time for the whole project, initially estimated to be 16 processors used during 50 days, to 26 days only.

The second application, under the responsibility of a research group from INRA, concerns the re-annotation of all known prokaryotic genomes, using the AGMIAL platform developed at the MIG INRA laboratory. During the reporting period, the user support team at IDRIS focused on the HMMER code used inside AGMIAL. It was ported to the SP4 platform but the performance tests showed that the input/output file operations were very poor. Consequently, some modifications were introduced to the code to store the whole database in memory. The elapsed time was divided by a factor of two after this improvement. The modifications made to the code were, of course, submitted to the authors to allow them to incorporate these improvements in their standard development branch.

6.5 JRA 5 (Industrial CFD)

JRA 5 is involved in computational aero-acoustic (CAA) modelling and simulations for some relevant automotive applications. The support tasks to this JRA during the first part of the projects are mainly related to:

1. Connectivity to the DEISA infrastructure. Set up of the minimal software environment at the CRF site and of the DEISA software environment at the CINECA site;
2. Installation, testing and tuning of the CFD++ package (the Metacomp Inc. CAA software) at CINECA, IDRIS and FZJ sites in order to assess the licensing issues and the environment set up. This has been installed on both an IBM SP and Linux cluster at CINECA;
3. Submission of test cases (mainly sizes of 32, 64 and 128 CPUs) in order to assess the resource needs and the simulation sizes;
4. Submission of "light" production cases for relevant simulations.

During the last six months (see deliverable D-JRA5-4 [9]), the activity has been focused on the debugging of the CFD++ software for large datasets and the tuning for IBM power 4 architecture. Two new CFD++ releases have been installed on the CINECA SP4 (now SP5) cluster.

6.6 JRA 6 (Coupled Applications)

For the previous six months, the JRA 6 focused its activities in exploiting the three coupled applications adapted to the DEISA environment (see deliverable D-JRA6-3 [10]). At the moment, it is still not easy to deploy coupled applications, even in the same site, as co-allocation capabilities are not yet available, and parts of coupled applications have to wait for each other or must be manually launched.

Several important technical problems were solved with the help of the user support team at IDRIS during this period:

- A specific study on pthread memory allocation (pthread stack) for the AIX system was done, when pthreads are allocated by OmniOrb, the CORBA implementation currently used.
- The asynchronous mechanisms, built with tools provided by the pthread library were more deeply tested. This functionality was implemented to avoid waiting for the end of a remote request, wasting cycles on dedicated resources. This piece of software is used in all of our coupled applications.
- There was a requirement in the environmental project to help to implement a new remotely accessible service. This service is devoted to a hydrological model in wet zones.
- The research teams helped to present a project to DECI, especially to calibrate a large coupled application and to make performance tests. The FOCUS project, coupling combustion and thermal radiation models, has been selected.

6.7 DECI projects

For the DECI initiative, managed by the *Applications Task Force*, the user support groups in all sites were deeply involved in the contacts with their local scientists who planned to send an Expression of Interest answering the Call for Proposals. Knowing their applications, they could give them various advices to prepare their proposal and to precisely estimate the required resources. And now, in order to support the initial DECI projects, the first of which were selected in September and launched in October, the *Applications Task Force* has set up a personalised support for each of the selected projects, composed of an *Ataskf* member of the Home site, a member from another one of the Execution sites (in most of the sites they belong to the user support groups) and a member of the operations group.

7. Other activities

7.1 *Trouble Ticket System*

As previously mentioned in the D-SA4-3 deliverable [4], a centralised *Trouble Ticket System* is a real requirement for a decentralised infrastructure such as DEISA. Although most of the partners already use such a system locally for their own needs, a common and centralised one was required for DEISA. After an analysis, during the spring of 2005, of the tools actually used by the partners, the setup of the DEISA configuration of that selected (Request Tracker [15]) was performed at RZG, which will operate this centralised service for the entire DEISA infrastructure. This is obviously a very important tool for the user support activities, but it will also be used for the other activities, especially by the operations group. It will allow to:

- give all sites a global view of the situation of the virtual platform by looking at all the problems opened, involving the various activities,
- allow the whole history of situations and problems reported to be kept,
- allow the generation of reports and statistics.

7.2 *DEISA Cluster Resource Management Package*

The *DEISA Cluster Resource Management Package* (DCRMP) will help the administrators of the DEISA sites in the installation and maintenance of their software environments, both for system and user oriented components. It will also allow the implementation of a powerful checking mechanism to guarantee the coherence of the installations (checking for dependencies between components, detection of incoherencies between expected versions of packages, etc.) This activity, lead by CINECA as an SA3 one, has already started and the first DCRMP distribution has been recently released, currently including UNICORE [18] and the *Resource Management Information System* (see deliverable D-SA3-3 [1]).

According to the technical choices already made, the SA4 group will use this framework to integrate in the future releases of the DCRMP various user oriented software packages, especially public domain scientific and graphics libraries, tools and applications. This will help the administrators in charge of the installations on the sites to perform them more easily, and will also to help to maintain a high level of coherence between the platforms themselves, as the version of the distributed packages will be guaranteed to be analogous. The setup of this task, for SA4 components, is currently under discussion and will be launched in the next few weeks.

8. References and Applicable Documents

- [1] DEISA D-SA3-3 deliverable: *First release of DCRMP*
- [2] DEISA D-SA4-1 deliverable: *SA4 Service Definition and Operation*
- [3] DEISA D-SA4-2 deliverable: *Basic DEISA Infrastructure Documentation*
- [4] DEISA D-SA4-3 deliverable: *First SA4 Annual Report*
- [5] DEISA D-JRA1-3 deliverable: *Full operation of Material Sciences Portal*
- [6] DEISA D-JRA2-1.2 deliverable: *Grid-enabled implementation of Gadget*
- [7] DEISA D-JRA3-3 deliverable: *Final report on simulation code TORB, progress report on new ORB5 code for extreme computing within DEISA*
- [8] DEISA D-JRA4-3 deliverable: *Final report on production status of initial applications*
- [9] DEISA D-JRA5-4 deliverable: *"Light" production cases results*
- [10] DEISA D-JRA6-3 deliverable: *Final report on technological and scientific impact of the three initial projects*
- [11] DEISA FAQ: <http://www.deisa.org/userscorner/faq.php>
- [12] DEISA Primer Documentation: <http://www.deisa.org/userscorner/primer.php>
- [13] INCA (Test Harness and Reporting Framework): <http://inca.sdsc.edu/>
- [14] Modules: <http://modules.sourceforge.net/>
- [15] Request Tracker: <http://www.bestpractical.com/rt>
- [16] SoftEnv: <http://www-unix.mcs.anl.gov/systems/software/msys>
- [17] TeraGrid: <http://www.teragrid.org/>
- [18] UNICORE (UNiform Interface to COmputing REsources): <http://unicore.sourceforge.net/>

9. List of Acronyms and Abbreviations

Ataskf	Applications Task Force
BLAST	Basic Local Alignment Search Tool
CPE	Common Production Environment
DCRMP	DEISA Cluster Resource Management Package
DECI	DEISA Extreme Computing Initiative
Home site	The site where a user usually works, logs in and submits jobs
INRA	<i>Institut National de la Recherche Agronomique</i>
INSERM	<i>Institut National de la Santé et de la Recherche Médicale</i>
IS	Information System
JRA	Joint Research Activity
MIG INRA	<i>Unité Mathématique, Informatique et Génome</i> of INRA
NCBI	National Centre for Biotechnology Information
Reference site	Same as <i>Home site</i>
RMIS	Resource Management Information System
SDSC	San Diego Supercomputer Center
TTS	Trouble Ticket System