Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)
Monterey, California, April 15-19, 2007, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2007)

EXPERIENCES APPLYING UNICORE GRID MIDDLEWARE FOR NUCLEAR SIMULATION TOOLS LIKE THE 3D REACTOR CORE MODEL QUABOX/CUBBOX

A. Geiger, J. Luther and G. Ohme
T-Systems Solutions for Research GmbH
Theodor Heuss-Straße 4, 38122 Braunschweig, Germany
Alfred.Geiger@t-systems-sfr.com, Johannes.Luther@t-systems-sfr.com,
Gert.Ohme @t-systems-sfr.com

S. Langenbuch and K. Velkov
Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH
Forschungsinstitute, 85748 Garching, Germany
Siegfried.Langenbuch@grs.de, Kiril.Velkov@grs.de

ABSTRACT

The progress of nuclear simulation techniques is characterized by more detailed modeling in neutronics and fluid dynamics and by multi-physics multi-scale coupling to fulfil the increased requirements on accuracy and fidelity of analysis. Therefore, advanced applications of reactor design and safety analysis need easy access to high performance computing resources. The vision of Grid computing is to provide these required computing resources at the engineer's desk. From the beginning SfR, the IT service provider of GRS, is participating in the development of Grid middleware like UNICORE (Uniform Interface to Computing Resources). The objective of this middleware is to establish a user friendly interface to the computing resources. UNICORE 5 is a production-ready and well tested Grid middleware. Meanwhile, the implementation is converted to internationally accepted Web Services standards in a similar way like the Globus Toolkit 4 leading to the new UNICORE 6 version. For testing the available Grid computing features and for getting early experiences for nuclear applications, the 3D reactor core model QUABOX/CUBBOX is chosen as an application case. This nuclear simulation code was fully integrated into a Grid infrastructure. The experiences obtained by installing the Grid middleware, by developing application clients and for job monitoring and data access services are presented and discussed.

Key Words: Grid computing, UNICORE, Nuclear simulation, QUABOX/CUBBOX

1. INTRODUCTION

Starting with the new millennium the vision of the future of computing services has become reality. Application services, managed services, and hosting are an increasingly common part of the computing landscape. Users are looking forward to access high performance computing resources as easy as other everyday appliances. Furthermore, the provision of computing services is increasingly driven by economies of scale and the effective utilization of resources which is strongly supported by Grid middleware.
In 2002 the Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH contracted the services of the whole IT infrastructure to T-Systems Solutions for Research GmbH referred to as SfR.

Looking at the scenario described above, we started to share the idea of making GRS applications ready for a usage within the new evolving Grid environments.
Assuming engineers could use remote high performing platforms as easy as local computers at the site of GRS, the high peaks of enormous computing power that are needed for the simulation of nuclear plants could be satisfied in a much more economic way. Some advanced models would even become reality for the first time. The external computers include e. g. the high performance systems of HWW (Höchstleistungsrechner für Wissenschaft und Wirtschaft Betriebsgesellschaft mbH) accessible to SfR. The goal of this partnership including the service provider T-Systems is to provide to its partners more powerful supercomputer capabilities than each of them could afford alone.
As an additional benefit the easy access to the GRS applications could lead to a much broader base of users, and it frees from the burden of distributing new code releases to their partners.
Last but not least, the vision of a new cooperation model could be realized between the end consumers, GRS as a provider of nuclear application codes and databases, and the IT expert SfR as a provider of computing and network resources as well as the Grid infrastructures.
A lot of applications of GRS were already developed in the 70s and 80s of the past century. They were continuously updated, but to be honest they didn't follow all the developments in software engineering. This means an interesting challenge, especially as the evolvement of the Grid is applying very advanced techniques of code management. Grids are in big parts still an objective of research and there isn't widespread experience with Grid middleware in productive use at all. Luckily with UNICORE 5 [1] a production-ready and well tested Grid middleware was already available.
Within the project UniGrids [2], financed by the EU within the 6th Framework Programme 2004-2006, we have chosen the 3D reactor core model QUABOX/CUBBOX (QCB) [3] as a pilot application of a nuclear simulation to investigate the needs and to perform an integration of such an application code into a UNICORE based Grid. Follow-up investigations have studied the more advanced features available for integrating application clients. In this paper we review the own activities, describe the reached state and give a glance at upcoming developments.

2. Description of UNICORE

2.1. History

Like with Globus Toolkit [4], the development of the middleware UNICORE started in 1997 even before the concept of "Grid Computing" [5] was coined by I. Foster and C. Kesselman in 1999. The main idea of UNICORE, financially supported by the German Ministry for Research and Education (BMBF), was to free scientists from the need to know about the details of different operating or batch systems before they could really start their own research. Therefore, the Grid principle of the abstraction of computer resources was born. Afterwards, in 2003 the release of UNICORE Plus followed, which is now called UNICORE 5. This is the version, which is the base for production today and which is also the main focus of the work presented in this paper.

2.2. State of the art and future developments

The standardization of the Open Grid Service Architecture OGSA [6] has caused a reformulation and reengineering of UNICORE 5 on the base of Web Services. The outcome is named UNICORE 6 with a proven high compatibility to all OGSA-based Grid middleware systems like Globus Toolkit 4 (GTK4) or Condor, only to name a few. The activities are continued in the Open Grid Forum [7]. Central concepts of UNICORE have become Grid Standards, for example JSDL, the job subscription definition language, which was in essence known as AJO, the abstract job object in UNICORE 5.
The software of UNICORE 6 [8] is right now available in the 6th alpha-release, a beta-release is planned for the middle of the year and a production release is foreseen at the end of 2007. The development and maintenance of UNICORE is coordinated and driven by the UNICORE Forum, an association of developers and users.

2.3 Production Version UNICORE 5

UNICORE 5 distinguishes between client and server components, an overview is given in Fig. 1. Both authenticate to each other with PKI-certificates of type X509v3. The communication between client and server after authentication uses encrypted secure socket layer SSL connections.
The client can be a simple command line interface. However, in most cases a graphical user interface GUI is used, that will be discussed later.
The basic server component named Vsite (Virtual Site) has four components:

- NJS
  The most important component is called Network Job Supervisor NJS. The NJS is mainly used for the communication with the client.

Two other components are used for the abstraction of a real computer:

- IDB
  The so called Incarnation DataBase IDB is used mainly for the abstraction of individual commands of the operating system
- TSI
  The Target System Interface TSI abstracts the different existing batch systems, like PBS (Torque), SLURM, LSF and so on.
- UUDB
  The UNICORE User DataBase UUDB is used to map between a user certificate and a local user account on the operating system. The UUDB is used for authorization.

A UNICORE client would be already able to talk to the NJS of an Vsite. In the real world several Vsites, i.e. different compute servers are separated from the internet by a firewall and form a UNICORE Site (USite). Another server component is needed to implement this concept.

- Gateway
  Tunnel the communication of a UNICORE client through a firewall into a Usite.

In UNICORE 5 the client and the server exchange messages forming an Abstract Job Object AJO (this name is historical and somewhat misleading, as not only jobs can be handled by the UNICORE Protocol Layer UPL).

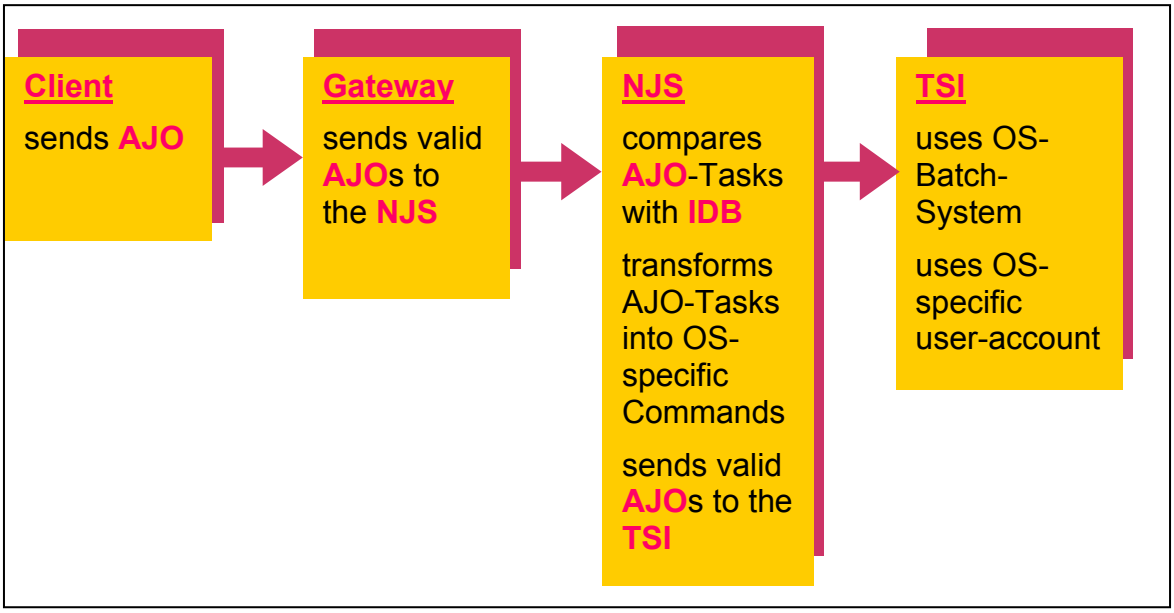

Fig. 1 Description of main UNICORE 5 components

In a real world scenario, it is often necessary that one job needs resources of different computers. For example one computer may be specialized in performing numerical calculations, another one

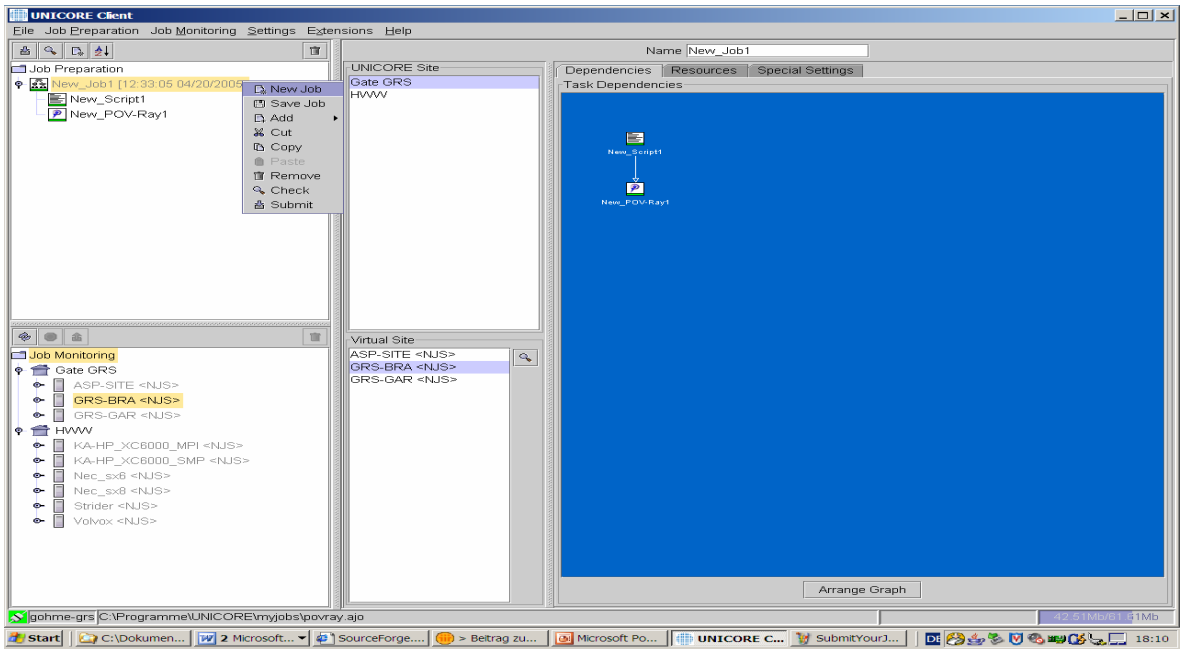

Fig. 2 UNICORE panel for job handling

is much more appropriate to do the visualization of results. In this situation, the NJS computing server can send its results to the NJS of the visualization server. This situation leads to the concept of a workflow, in which a NJS can behave even like a client.
An end user can use the UNICORE Client in all phases of job handling as can be seen on the panel of Fig. 2.
The main job phases are

- Job Preparation
- Workflow Description
- Job Monitoring
- Job Postprocessing

The upper left sub-window is used for the job preparation: for example local job input files can be specified as well as script files, which have to be executed on the Vsite.
The lower left sub-window shows all the Usites and their related Vsites that can be accessed with your certificates. In our case we see first the Usite of local computers at GRS and second the Usite of the HWW with several other computers.
In the two middle sub-windows the Usite (upper) and Vsite (lower) can be chosen, where the job or sub job should run on. In our case we choose the Usite of GRS and the Vsite GRS-BRA.
In the big blue window the workflow can be defined. In our case first a script file “New Script” shall be executed and after this as an example the application PovRay [9] (a well known ray tracing code to render photo realistic pictures) shall be started.
With the concept of plugins the UNICORE 5 client can be expanded to support the requirements of application specific tasks. You can formulate application specific input masks as well as output handling. The Fig. 3 shows an example outcome of the ray tracer PovRay. For more details and the code we refer to the UNICORE website.

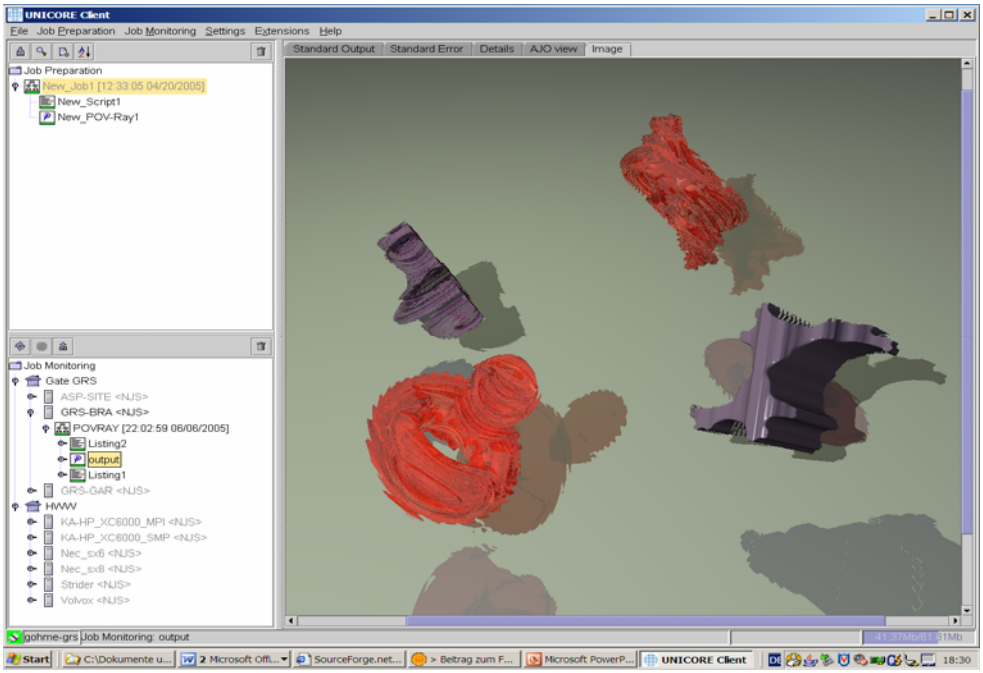

Fig. 3 UNICORE application for the exemplary PovRay case

3. Experiences with the UNICORE infrastructure

The client installation only takes a few minutes and a very good documentation for the usage is available at the UNICORE website.
The installation of UNICORE on the server side is also easy and a good documentation exists. The installation of the server component software for the three Usites at GRS, including the gateway and the handling of the firewall was done in only a few hours.

But there is a hurdle, which has to be mentioned:
The generation and management of X509v3 certificates is a prerequisite for the usage of UNICORE or most other Grid middleware. If no assistance of a security expert is available, then one has to invest quite some days or even more to get a public key infrastructure (PKI) certification authority (CA) fulfilling the requirements of UNICORE to work. It should be mentioned, that even the PNPCA (plug and play CA) offered at the UNICORE download site is still in a beta state.

In fact, one has to be able to generate or get at least two (better three) types of X509v3-certificates:

- User Certificates (necessary, extended key usage "clientAuth", SSL/TLS Web Client)
- Server Certificates (necessary, extended key usage "serverAuth, clientAuth". SSL/TLS Web Server)
- Code Certificates (recommended, extended key usage "codeSigning", Code Signing)

Of course, the user certificates are applied on the client side, the server certificates on the server side. Code certificates can assure the integrity of downloadable plugins and/or limit their usage.

The next needed knowledge has to do with the used keystore format. It is necessary to get familiar with the format JKS (Java Keystore) used by Java and second the PEM (PKCS12) keystore format, which is used by the opensource software openssh.

4. QUABOX/CUBBOX in a UNICORE application scenario

4.1. General software considerations

QUABOX/CUBBOX (QCB) is a legacy FORTRAN code, because it was developed in the 70s. This made several requirements and conditions evident:

- QCB needed to be ported to systems, which could be used as platforms for UNICORE, using flexible methods of automatic installation.
- QCB is in productive use. Model adoptions by the engineers could take place while concurrently work is performed on the code under web services aspects.
- No change must ever lead to different results.

To manage the concurrent changes by the physicists and the web service people, we modernized the source code management with a Subversion [10] based software repository. The integrity of the code after changes is steadily checked by Gump [11] and a continuous integration service CIS [12], which checks the output of the changed code against reference cases.

4.2. Development of a plugin for UNICORE 5

For UNICORE 5 a plugin was programmed for the reactor core model QCB, Fig. 4. As can be seen, the new element “QCB” was added to the list of available tasks of a workflow. If it is selected, an input mask appears, (Fig. 5), where the user can specify some selected parameters for the job. These parameters include a Jobname, the filename of the QCB-inputfile, the directory, where the results will be stored, when they come back from the Vsite. The three parameters MODE, TAVREF and RHOREF are typical input values for the reactor core model. They are used here as a demonstration of input handling. These values override the data for the corresponding parameters in the input-files. All parameters are handed over to QCB, when the job is submitted to the Vsite.

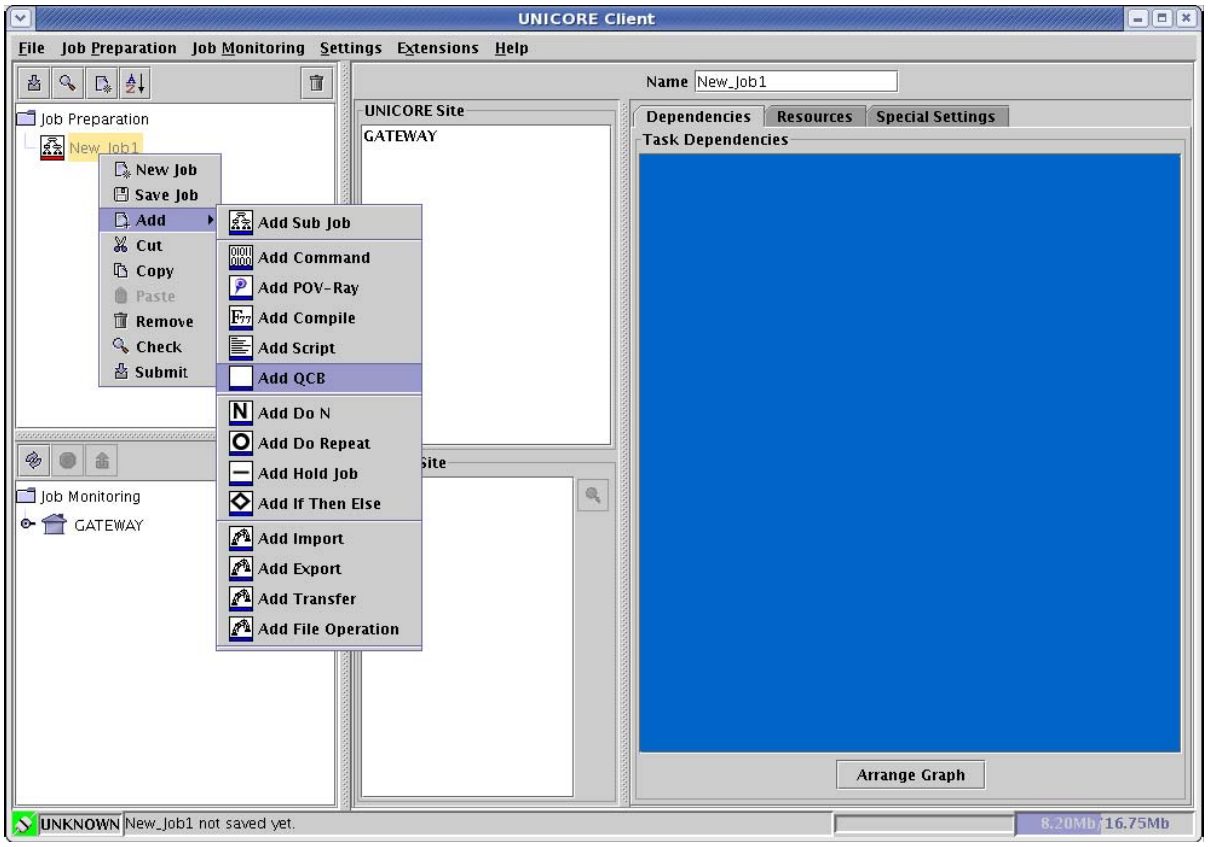

Fig. 4 UNICORE panel showing the plugin application client QCB

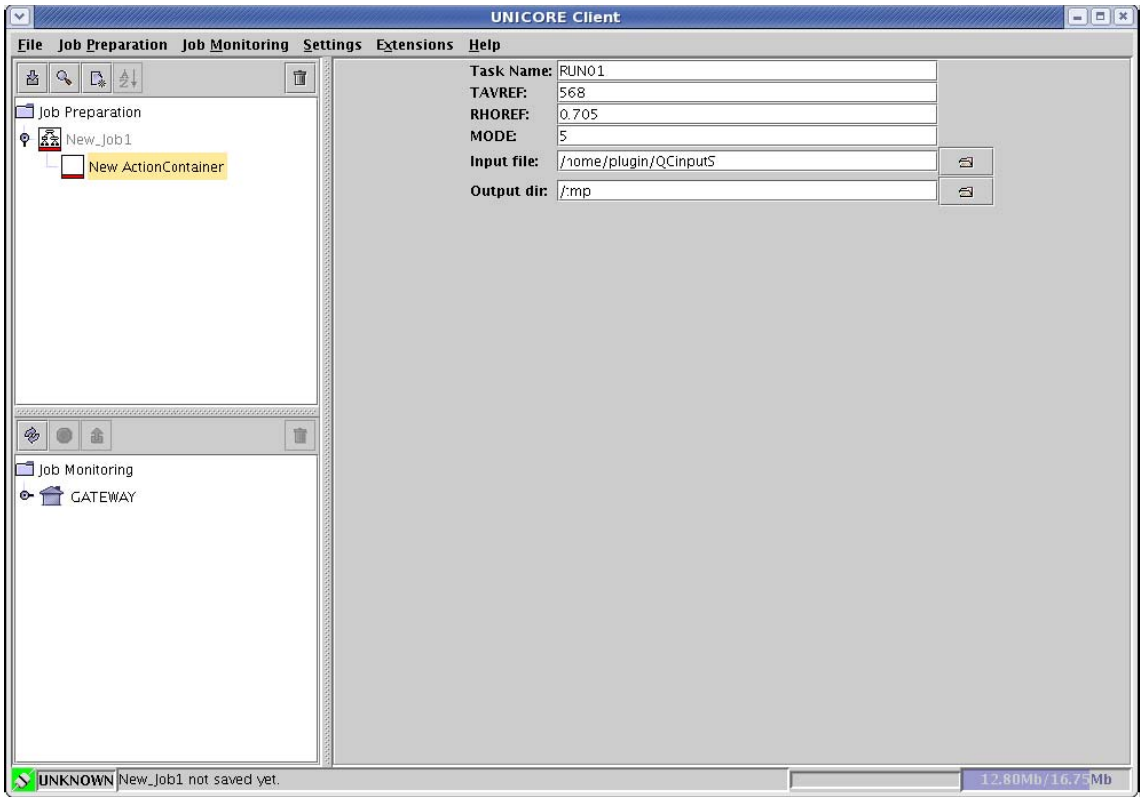

Fig. 5 UNICORE input panel of the QCB plugin

This implemention of a UNICORE 5 plugin for QCB for the job handling of nuclear calculations was used as a base for the next step applying the new version UNICORE 6.

4.4. Development of a GridBean for UNICORE 6

The new UNICORE 6 implementation provides the Grid Programming Environment (GPE). It makes available three types of clients:

- An expert client, which gives full access to the Grid for expert users and administrators.
- An application client, as a Java application for defining simple interfaces for specific applications.
- A portal client, as a portlet compliant to Java Specification Request (JSR) – 168 which can be applied by standard Web browsers.

A specific application is represented in this environment by a corresponding GridBean [13]. The adaptation to the new specifications was part of a diploma thesis work [14]. Such an application GridBean can be handled by each type of client and the format requirements are defined within this standard.

An application GridBean consists of three main software elements:

- A main class of GridBean
- A class to generate the GUI, and
- A description of the GridBean using XML.

The GridBean of the QCB application is described by the following:
The Java program QCBGridBean.java contains the declaration of the namespace and the variables and the constructor of the class QCBGridBean.

```
Public QCBGridBean() {
        Set (JOBNAME,"QCB");
        Set (TAVREF,"568.0"); /* example of a parametervalue */
        Set (SOURCE,sourceStub);
        Set (TARGET, new GPEFileStub("ex.out"));}
```

The QCBPlugin.java Program includes the constructor of the class QCBPlugin and further elements describing the panel functions.

```
Public QCBPlugin(Client client) {
        Super ("QCB");
        addInputPanel( new OptionsPanel(client)) ;
        addInputPanel( new SourceFilePanel(client)) ;
        addOutputPanel( new OutputPanel(client));}
```

Other programs like OutputPanel.java and SourceFilePanel.java were directly taken from the installation example.

The description of the QCBGridBean as XML file GridBean.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<gb:GridBeansInfo xmlns:gb="http:gpe.intel.com/gridbeans">
<gb:Name></gb:Name>
<gb:Author>Johannes Luther</gb:Author>
<gb:Version>1.0</gb:Version>
<gb:Application>QCB</gb:Application>
<gb :Description>QCB GridBean</gb :Description>
<gb :PluginVersion>1</PluginVersion>
<gb :GridBean>com.intel.gpe.gridbeans.qcb.QCBGridBean</gb :GridBean>
<gb :Plugin
type="gb :Swing">com.intel.gpe.gridbeans.qcb.plugin.QCBPlugin</gb:Plugin>
<gb:ApplicationName>QCB</gb:ApplicationName>
<gb :ApplicationVersion>1.0</gb :ApplicationVersion>
</gb :GridBeansInfo>
```

With these new Java programs it is possible to integrate the application QCB into the new release UNICORE 6. The application panels look very similar to those of the plugin option in UNICORE 5 presented in Fig. 4 and Fig. 5.

4.4. Introduction of XML output in QUABOX/CUBBOX

A specific aspect refers to the exchange of data between applications in the Grid. The I/O-statements in QCB have been changed by replacing important output statements by equivalent XML-output.
For example QCB uses the following quite common code (simplified) for the output of 3D arrays, which describes for example the energy production rates or thermal feedback parameters within a nuclear reactor.

```
      REAL*8 ARRAY(IIX,KKY,MMZ)
      . . .
      DO 9150 IISLOW=ISLOWA, ISLOWE
      . . .
      WRITE (NWR,9220) . . .  , . . .  ,  . . .
     1       (ARRAY(IIFAST,IIMEDI,IISLOW), IIFAST=IFASTA,IFASTE)
9220  FORMAT(1H0,2A3,I2,',',I2,')',1P10D12.5)
      . . .
9150  CONTINUE
```

It is easy to see, that most of the statements are declarative, only the DO and WRITE aside "=" mean functionality. However an external code in a workflow, which wants to work on this QCB output besides the data would need an additional knowledge about the declarations, which are hidden in QCB statements.
It is an advantage to replace this part of code by the following lines achieving a self declarative XML output as a result:

```
      WRITE(JNIO,   '("<?xml version=""1.0"" encoding=""UTF-8""?>")')
      WRITE(JNIO,   '("<ARRAY>')
      WRITE(JNIO,   '("  <NAME> ARRAY  </NAME>")')
      WRITE(JNIO,   '("  <TYPE> REAL*8 </TYPE>")')
      WRITE(JNIO,   '("  <DIMENSIONS>")')
      WRITE(JNIO,   '("    <DIMENSION>i6</DIMENSION>")') IIX
      WRITE(JNIO,   '("    <DIMENSION>i6</DIMENSION>")') KKY
      WRITE(JNIO,   '("    <DIMENSION>i6</DIMENSION>")') MMZ
      WRITE(JNIO,   '("  </DIMENSIONS>")')
      . . .
      WRITE(JNIO, '  <VARLIST>')
      . . .
      DO 9150 IISLOW=ISLOWA, ISLOWE
      . . .
      DO IIFAST=IFASTA,IFASTE
        WRITE(JNIO, '("    <ELEMENT>")')
        WRITE(JNIO, '("      <COUNTER>")')
        WRITE(JNIO, '("        <NUMBER>     1</NUMBER>")')
        WRITE(JNIO, '("        <INDEX>", i,"</INDEX>")')IIFAST
        WRITE(JNIO, '("      </COUNTER>")')
        WRITE(JNIO, '("      <COUNTER>")')
        WRITE(JNIO, '("        <NUMBER>     2</NUMBER>")')
        WRITE(JNIO, '("        <INDEX>", i,"</INDEX>")')IIMEDI
        WRITE(JNIO, '("      </COUNTER>")')
        WRITE(JNIO, '("      <COUNTER>")')
        WRITE(JNIO, '("        <NUMBER>     3</NUMBER>")')
        WRITE(JNIO, '("        <INDEX>", i,"</INDEX>")')IISLOW
```

```
        WRITE(JNIO, '("       </COUNTER>")')
        WRITE(JNIO, '("       <REPRESENTATION>")')
        WRITE(JNIO, '("         D12.5")')
        WRITE(JNIO, '("       </REPRESENTATION>")')
        WRITE(JNIO, '("       <VALUE>")')
        WRITE(JNIO, '("        ",D12.5)')ARRAY(IIFAST,IIMEDI,IISLOW)
        WRITE(JNIO, '("       </VALUE>")')
        WRITE(JNIO, '("     </ELEMENT>")')
      ENDDO
9150 CONTINUE
     WRITE(JNIO,     '("  </VARLIST>")')
     WRITE(JNIO,     '("</ARRAY>")')
```

The variable “JNIO” in this context points to the output file of the XML data. This XML file can be easily read in by a common XML parser of an external application. This was done to build a simple visualizer for the 3D data as a chain of 2D slices as shown in the Fig. 6 below.

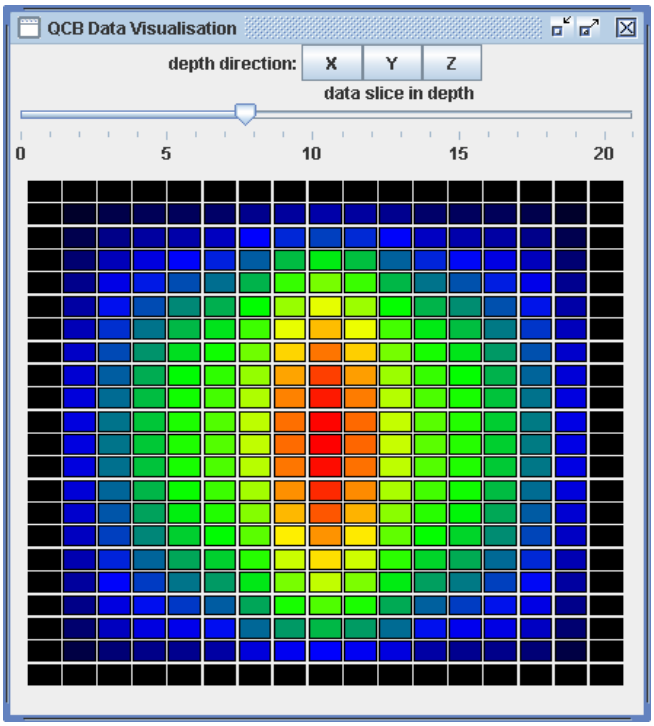

Fig. 6 Example of a simple visualizer based on XML output data

With the three buttons the direction of the data slices can be chosen. In addition, with the slider the specific planes can be chosen. In this way, one now can easily walk through the 3D data array.
The XML description means the addition of some overhead, but a big advantage is gained by the self description of the data set, which supports the transfer between different computers and applications.

5. CONCLUSIONS

The investigations performed in the framework of the UNICORE Grid middleware using the 3D reactor core model QUABOX/CUBBOX as a representative nuclear application have been successful. The objective was achieved to integrate the nuclear simulation code formulated in a legacy programming language into a modern Grid environment. The application client has been fully integrated into the UNICORE infrastructure in two options: a plugin for UNICORE 5 and a GridBean for UNICORE 6. Extended practical experiences have been gained by the implementations. In future, the generated prototypes can be applied for integrating other applications into the Grid. The advantage of data exchange between clients using XML formats has been demonstrated. It is mandatory to get support from experienced software engineers, because during the phase of installation and integration a lot of technical issues have to be solved. Only after reaching the goal, all the foreseen advantages of high performance computing in the Grid can be used for production simulations.

REFERENCES

1. Homepage of UNICORE http://www.unicore.eu
2. Homepage of project UniGrids http://www.unigrids.org
3. Homepage of Globus Toolkit http://www.globus.org
4. S. Langenbuch and K. Velkov, Overview on the Development and Application of the Coupled Code System ATHLET-QUABOX/CUBBOX, M&C, Avignon, Sept. 12-15, 2005
5. I. Foster, C. Kesselman: The Grid – Blueprint for a New Computing Infrastructure
6. The Open Grid Service Architecture (OGSA) http://www.globus.org/ogsa
7. Homepage of Open Grid Forum http://www.ogf.org
8. UNICORE, OGF19, Software Provider Forum, Chapel Hill, NC, USA, Jan 29 – Febr 2, 2007
9. Homepage of PovRay http://www.povray.org
10. Homepage of Subversion http://svn.tigris.org
11. Homepage of Gump http://gump.apache.org
12. Homepage of CIS-Server T-Systems-SfR : http://cis.t-systems-sfr.com
13. R. Ratering et al., GridBeans: Supporting e-Science and Grid Applications, Proc. of the 2nd IEEE Int.Conf. on e-Science and Grid computing, Amsterdam, Dec 4-6, 2006
14. J. Luther, Control of a simulation code in Grid environments, Diploma thesis at FH Landshut, February 2007