



CONTRACT NUMBER 508830

DEISA
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
SUPERCOMPUTING APPLICATIONS**

European Community Sixth Framework Programme
RESEARCH INFRASTRUCTURES
Integrated Infrastructure Initiative

Quickstep code from CP2K package for adequate usage in
DEISA

Deliverable ID: DEISA-DJRA1-4

Due date: April 30, 2006

Actual delivery date: May 17, 2006

Lead contractor for this deliverable: RZG, Germany

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 Contents

Table of Contents.....	1
1. Introduction.....	2
1.1 Executive Summary.....	2
1.2 References and Applicable Documents.....	2
1.3 Document Amendment Procedure.....	2
1.4 List of Acronyms and Abbreviations.....	3
2. Quickstep Grid-Enabling work.....	4
2.1 The Quickstep software requirements.....	4
2.2 Compiling and Testing.....	5
2.3 Things to be done.....	5
3. Design and Implementation of Quickstep Portal Plug-In.....	6
3.1 MDA approach to Quickstep.....	6
3.2 The Web Application Plug-In.....	7
3.2.1 The Quickstep Editor.....	7
3.2.2 Upload.....	9
3.2.3 The source editor.....	9
3.2.4 The design view.....	10
3.2.5 Inserting New Sections.....	11
3.2.6 Multiple open projects.....	12
3.2.7 Job submission.....	12
4. CPMD support by HLRS.....	12
5. Screening for further relevant codes.....	14
5.1 GROMACS.....	15
5.2 ESPResSo.....	15
5.3 LAMMPS.....	15
5.4 NAMD.....	15
5.5 NWCHEM.....	15
6. Conclusion and Outlook.....	15
7. Acknowledgement.....	16

1. Introduction

1.1 Executive Summary

Work was focussed on supporting the QUICKSTEP code in DEISA. The aim of the grid-enabling work was to ensure that recent versions of Quickstep can be compiled and run in the DEISA infrastructure without problems.

After analysis of the Quickstep software requirements, Quickstep was successfully compiled on various target architectures (Power4, Power4+, Power5, and PowerPC). In order to check the accuracy and integrity of the simulation results, the Quickstep regtest test suite was used.

A major task was then the design and the implementation of a Quickstep plug-in for the JRA1 portal solution, to give users a high level of support in creating and modifying Quickstep input files. State-of-the-art web publishing technologies present in most of modern browsers have been employed to achieve the goal of maximum usability.

The application plug-in now supports the full functionality required. This includes a tree explorer, an editor, uploads, project creation, syntax checks, and job submission.

Work on the JRA1 portal has been presented by T. Soddemann: "Science Gateways to DEISA. Motivation, user requirements, and prototype example" at GGF14, Chicago, IL, USA, June 26-30, 2005. Science Gateways: Common Community Interfaces to Grid Resources http://www.ggf.org/GGF14/ggf_events_schedule_Gateways.htm, jointly organized by: GGF Steering Group Community Council, NSF TeraGrid Project (US), HPC-Europa Project, Pragma. The corresponding paper has been accepted for publication.

Additionally the new DEISA site HLRS has contributed support for optimized use of CPMD on their NEC vector architecture.

1.2 References and Applicable Documents

[1] <http://www.deisa.org>

[2] <http://cp2k.berlios.de/>

1.3 Document Amendment Procedure

1.4 List of Acronyms and Abbreviations

API	Application Programming Interface
CP2K	Car-Parrinello molecular dynamics 2000
CPE	Common Production Environment
DESHL	DEisa Services for Heterogeneous management Layer
EJB	Enterprise Java Bean
GPFS	General Parallel File System
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	secured version of HTTP
IE	Microsoft Internet Explorer
Java EE	Java Enterprise Edition
JCP	Java Community Process
JRA	Joint Research Activity
JSDL	Job Services Description Language
JSR	Java Specification Request
MDA	Model Driven Architecture
MVC	Model-View-Controller pattern
NJS	Network Job Supervisor
O/R Mapping	Object/Relational Mapping
OASIS	Organisation for the Advancement of Structured Information Standards
OMG	Object Modelling Group
PDA	Personal Digital Agent
POJO	Plain Old Java Object
SA	Service Activity
SOA	Service Oriented Architecture
UML	Unified Modelling Language
UNICORE	UNiform Interface to COmputing REsources
WS	Web Service
XHTML	XML syntax compliant HTML
XLST	XML Style SheeT
XMI	XML container syntax for UML
XML	eXtensible Markup Language
XSD	XML Schema Definition

2. Quickstep Grid-Enabling work

CP2K is a freely available (GPL) program (see <http://cp2k.berlios.de>), written in Fortran 95, to perform atomistic and molecular simulations of solid state, liquid, molecular and biological systems. It provides a general framework for different methods such as density functional theory (DFT) using a mixed Gaussian and plane waves approach (GPW), and classical pair and many-body potentials. CP2K provides state-of-the-art methods for efficient and accurate atomistic simulations. Sources are freely available and actively improved.

The DFT part of CP2K is referred to as Quickstep. Quickstep is the more mature part of CP2K. Many of its modules are of production quality, so that several of our users have already decided to employ Quickstep for their research even in this early stage of the product.

The grid-enabling work of Quickstep consisted of two steps which are described in the following sections. The aim of the grid-enabling work is to ensure that recent versions of Quickstep can be compiled and run in the DEISA infrastructure without problem. It is not yet the aim of this task to have Quickstep integrated into the Common Production Environment (CPE) software stack at this point in time, since Quickstep is evolving too quickly. Bug fixes as well as new features appear every other week. The typical Quickstep user will currently use a private copy of a Quickstep version which suits his needs. This aspect will also be discussed in the next section which deals with the Quickstep portal plug-in.

2.1 The Quickstep software requirements

It had to be assured that the software stack defined in SA4 reflects the needs of Quickstep. The following table gives an overview of the software requirements of Quickstep and their availability in the current (at the time of writing this deliverable) Common Production Environment (CPE) software stack.

Software	(version)	CPE
gmake		yes
Fortran 90/95		yes
BLAS		yes*
LAPACK		yes*
math-atlas		no*
SCALAPACK		yes*
MPI		yes
FFTW	2.1.5	yes

* Vendor libraries available

For a part of the mathematical libraries proprietary vendor libraries such as the Intel Math Kernel Library can be used on some platforms. As it turned out, not every Fortran 90/95 compiler supports the language features being used within Quickstep. Fortunately, however, the main target platforms based on the IBM Power/PowerPC are fully compliant. This problem therefore does not yet have a direct effect on DEISA and is hence not discussed in the following.

2.2 *Compiling and Testing*

In a second step, the Quickstep code had to be compiled on the various target architectures (Power4, Power4+, Power5, and PowerPC). The compilation was successful on all those platforms.

In order to check the accuracy and integrity of the simulation results the Quickstep regtest test suite was used.

2.3 *Things to be done*

Since Quickstep is still in development, it would be unwise to prematurely freeze a version in order to make it available via the DEISA CPE. This will change in the future. Hence, SA4 will need to be prepared to support a version of Quickstep which will be marked as stable by the Quickstep authors. JRA1 will stay in contact with the Quickstep authors in order to find and check suitable release candidates. Furthermore, JRA1 will work together with SA4 on the provision of a CPE module for Quickstep.

3. Design and Implementation of Quickstep Portal Plug-In

Unlike many other codes, Quickstep creates an XML syntax and semantic description of its input files. Although it is not an XML schema definition, it is very well suitable to follow a Model Driven Architecture (MDA) approach and having large parts of the implementation work done automatically.

MDA can be seen as synonym for separating the model from its underlying implementation. It separates the business and application logic from the technological platform. We have used an MDA-like approach in order to have Java class definitions, and also HTML forms and form mapping/binding information, automatically generated from the given business logic which in this case is the Quickstep input file description. Details are described in the first subsection.

The development of the concept making this MDA approach possible is a result of the close cooperation of this JRA with the Parrinello group. This cooperation has already proved to be successful for CPMD and has now been continued with the developments around the CP2K package.

All parts of the application which do not directly deal with the representation of Quickstep input files have been developed using conventional design and implementation methods. A more detailed analysis is given in the second subsection. As in the case of all other already available plug-ins (see DEISA-DJRA1-3 and DEISA-DJRA1-2) we have made use of the underlying portal application's four tier design. By making extensive use of Javascript on the client side, the served HTML pages are a more rich client solution than just bare and quasi static content. The web application is naturally based on the cocoon framework and the infrastructure given by the main portal application. The Java Enterprise tier in this case currently mainly serves the purpose of providing services with respect to managing the persistence of project data. This is described in more detail in the last subsection.

3.1 *MDA approach to Quickstep*

Model Driven Architecture is a quite new approach to software design. Its principle is that the model which describes the software is independent of the underlying implementation technology. When applying MDA, a software architect will create a description of the software in a technology-independent language such as UML and later create the technology dependent parts using some kind of meta-compiler.

In the present case, we used the XML description of the Quickstep input file as our model. Of course, it only describes a part of the application, but it has all the information necessary for successfully dealing with Quickstep input files. Since the XML description file's syntax is proprietary (meaning it is not XSD or XMI), we first needed to develop a kind of meta-compiler which creates the necessary programme source code in the desired programming language (Java). This was achieved by developing a set of XML style sheets which transform the input file description into Java source files and furthermore into cocoon-form XML definitions, bindings and XHTML templates. This way, almost 300 classes and forms have been generated, reflecting the different possible input file sections currently supported by Quickstep.

Although a large part of the Java source code for dealing with Quickstep input file can be generated automatically, there are two Abstract Classes which need to be manually created: These are an abstract class for embedding code stubs for reading and outputting Quickstep input file segments (which is extended by each of the generated classes), and a parser class.

Due to its hierarchical structure, the Quickstep input file is internally represented as an XML DOM document, where each section is held in the `userData` field of an associated XML element object. This allows for easy navigation through the tree structure as well as the opportunity to check the structural correctness of the Quickstep input file at creation and modification time.

3.2 The Web Application Plug-In

The web application is initially supposed to give its users a high level of support in creating and modifying Quickstep input files. In the case of the CPMD application plug-in the focus was to complement the UNICORE rich client¹ solution with a comfortable web based approach. In the case of Quickstep no UNICORE client exists. Hence, the design of the web application, especially the view components, can be carried out free from any constraint in terms of look and feel. Nevertheless, this does not make the task of designing a useful user interface any easier.

In order to achieve the goal of maximum usability, we have employed state-of-the-art web publishing technologies present in most modern browsers such as Mozilla-like browsers (Mozilla, Firefox, Netscape), Opera and MS Internet Explorer.² Hence, the Quickstep web application plug-in can be seen as a rich client solution implemented in HTML with the help of CSS and Javascript.

Actually, we make extensive use of Javascript. A couple of Javascript objects play the role of a controller and respond directly to user input. In turn the user benefits from this design by working with an application which already responds from within the browser on the client side. It does not necessarily have to wait for a server response. Most of the server interactions happen asynchronously anyway, which underlines the design of a quickly responding rich client application.

3.2.1 The Quickstep Editor

When a user has navigated to the Quickstep input file editor page, he will see the familiar DEISA header and portal footer plus a web application interface which consists of a menu bar and different window-like areas. Among them are a tree explorer and a source editor pane which can be toggled between a source editor (source) and a form view (design).

¹ Naturally, this is seen from the perspective of an eager Web application development team.

² Nevertheless, it has to be noted that only Mozilla-like browsers have yet been thoroughly tested, and CSS problems may occur when pages are currently viewed with IE,

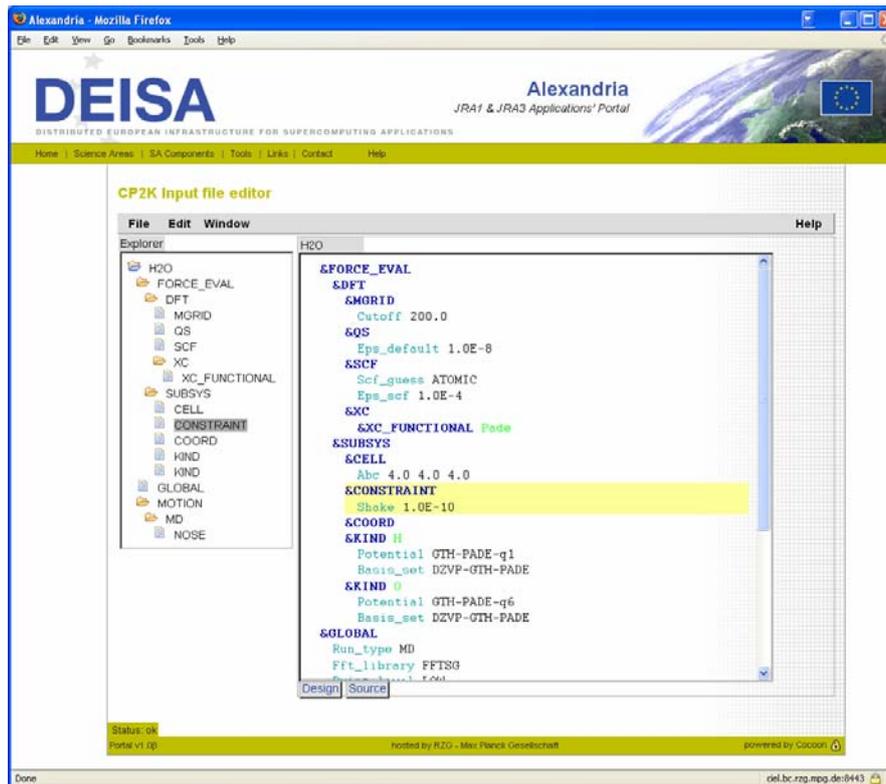


Figure 1: Selecting a section in the explorer tree highlights the appropriate section in the source editor. The source editor also shows syntax highlighting of the Quickstep input file.

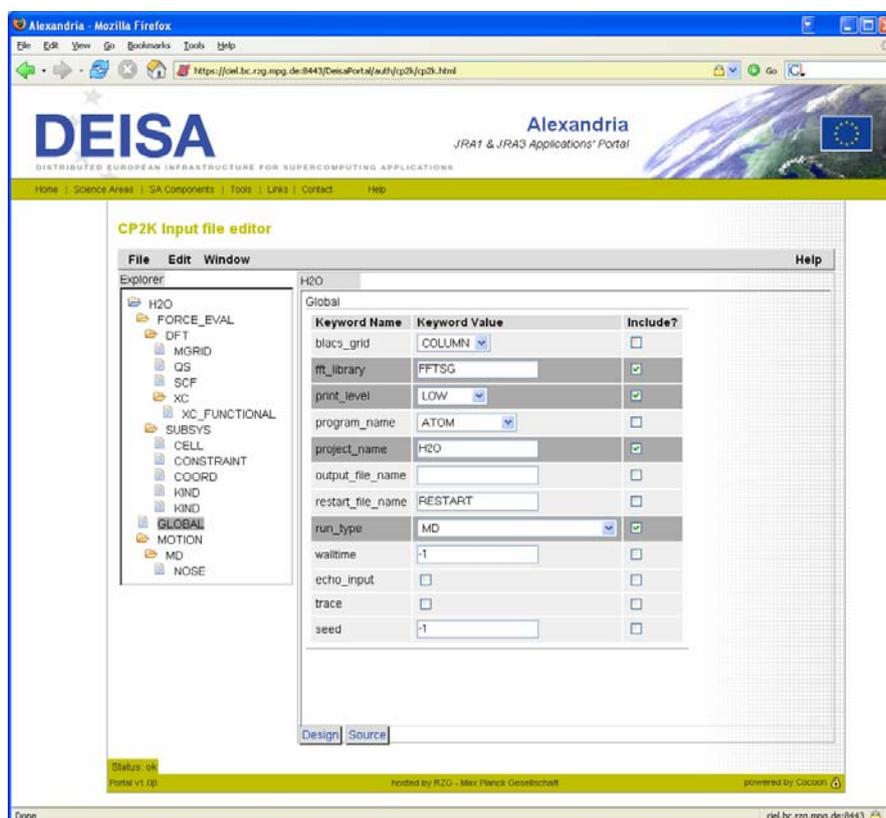


Figure 2: Design view: The selected section in the explorer is shown as a form with all its keyword possibilities. Dark grey lines highlight the keywords to be included in the Quickstep input file.

3.2.2 Upload

Usually, the user will start with an existing Quickstep input file. In order to do so, he selects "upload" from the "file" menu. A popup window is then shown asking the user for the file to upload and optionally for the project name to be used. In case no name is given, the name specified in the uploaded file is used. If that has also not been specified then the filename is used without its extension.

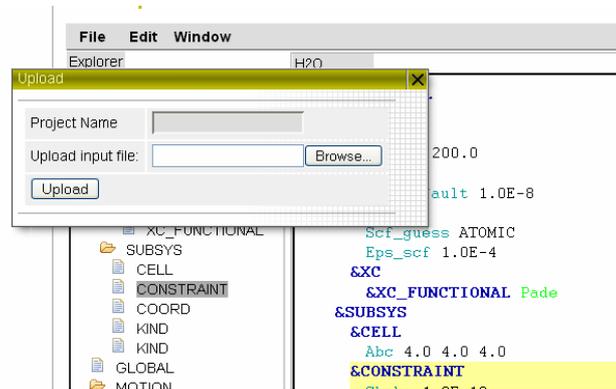


Figure 3: Uploading a new Quickstep project using the upload manager. The project name is optional since it may be contained in the Quickstep input file. If the name is not present, the input filename without extension is used.

Upon clicking the upload button the editor pages are reloaded from the server with the updated information. The Explorer window now contains the tree structure of the input file.

3.2.3 The source editor

In the source editor pane the source editor is shown with syntax highlighting enabled. The source editor is not a normal HTML text area but makes use of the rich text editor embedded in most modern browsers. Hence we can highlight sections and keywords in different colours. When a user modifies the input file in the source editor, his current modifications, which have not yet been synchronized with the server, can be identified by the black letters on the light red background. If the source editor loses focus or if the user presses <ctrl>-s on the keyboard the modification is sent to the server. There the modified input file is parsed again. The Javascript application in the client browser will then pull the modified tree and source from the server and make them visible at the appropriate places. In case of an error, the problem region is marked and the user will see an error message in the logs window. The editor further supports common actions as copy, cut, and paste. Its undo and redo features are currently limited to the last action carried out.

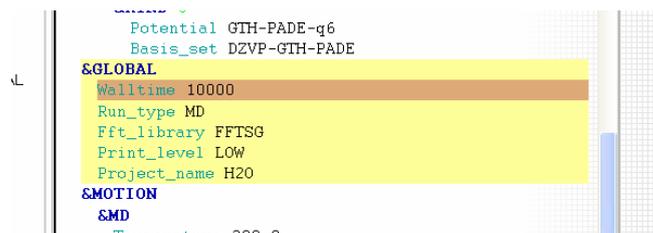


Figure 4: A light red background indicates ranges which have been changed in the editor but have not yet been updated on the server.

The nodes and leaves of the tree in the explorer are selectable. Upon selection, the appropriate section in the source editor is highlighted and the section form with the current values is asynchronously fetched from the server.

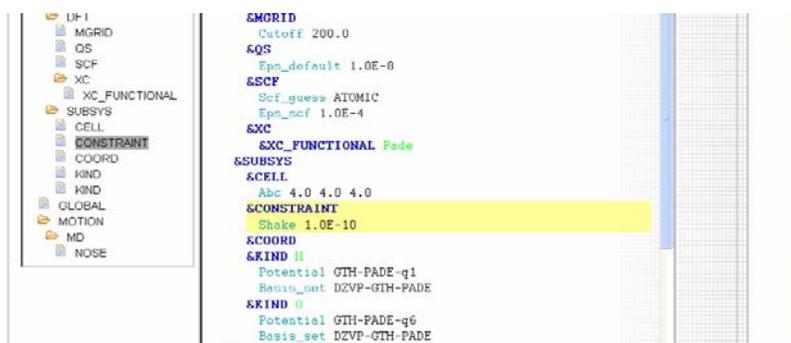


Figure 5: The nodes and leaves of the tree in the explorer are selectable. The appropriate section in the source editor is highlighted upon selection.

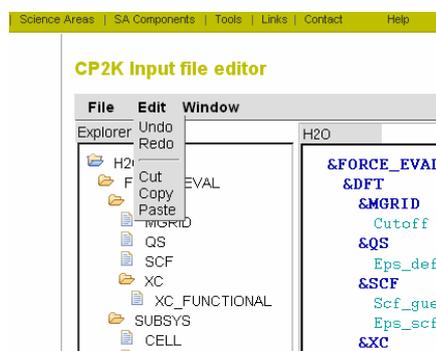


Figure 6: Common operations such as Cut, Copy, and Paste are supported for use in the source editor. The Undo and Redo functions are currently limited to the last action.

3.2.4 The design view

When the user switches to the design view, he can display those forms instead of the input file source. The forms contain all allowed keywords for each section. The keywords to be included in the input file are marked with a checkbox and highlighted in dark grey for better visibility. Changes are again reported back to the server upon pressing <ctrl>-s on the keyboard or using the *save* menu item from the *File* menu.

Figure 7: Upon selection of the inclusion checkbox in the forms view, its row is highlighted for better visibility.

3.2.5 Inserting New Sections

Insertion of new sections is achieved using the *new->section* menu item from the *File* menu. Depending on the selected section in the explorer tree, sections can only be inserted if they are allowed as sub-sections of the selected section. Similarly, the user can delete sections via the *cut* menu item of the *Edit* menu.

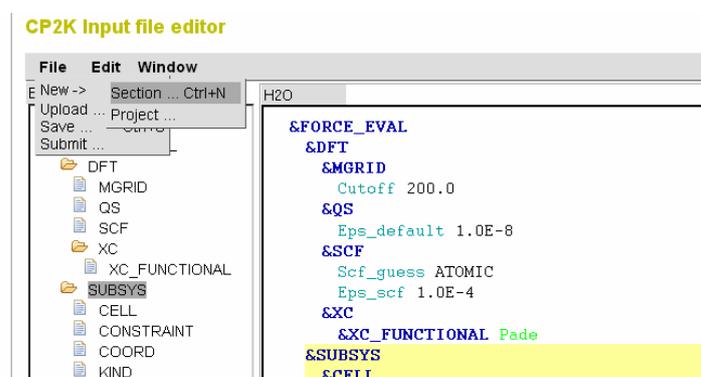


Figure 8: A new section is created by selecting *New->Section* from the *File* menu.

When the new section dialog is opened an appropriate choice of addable sections is shown. In cases where a section already exists and is only allowed to occur once in the parent section, the checkbox is disabled. We see this as an advantage for the user in comparison to not showing the not addable section at all. The user knows what sections can in principle be added and will directly understand why this section is not addable. This is a common behaviour of many commercial applications.

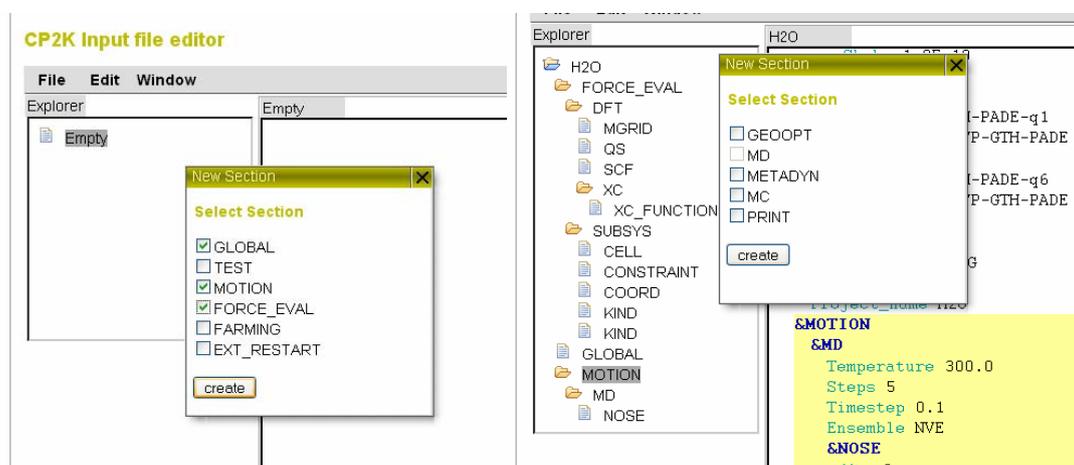


Figure 9: A new section can be created by using the *New->Section* menu and selecting the appropriate sections from the dialog. Selecting sections which are not allowed in the current context is disabled (right).

3.2.6 Multiple open projects

The Quickstep input file editor application is designed to support multiple open projects. Context switching takes place upon selecting the editor tab with the name of the project. The explorer and source editor panes always show the currently active project.

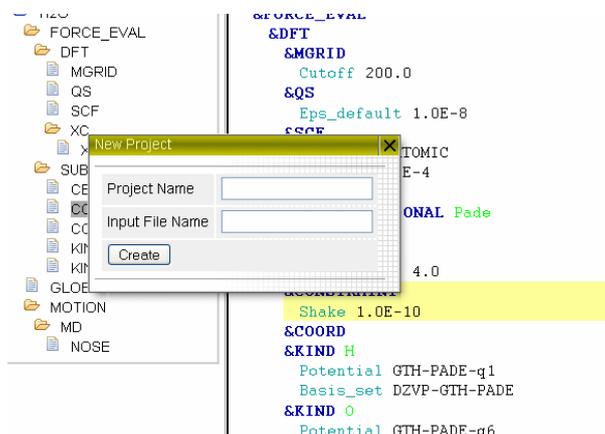


Figure 10: Opening a new empty project using the project create dialog.

3.2.7 Job submission

Once the user has an input file ready for submission, he does this with the submit menu item from the File menu. The job is then prepared and given to the Job Manager described in DJRA1-3 for submission and runtime handling.



Figure 11: A job submission is performed with the File->Submit menu item.

Using the Job Manager the user will have to take care of selecting the correct Quickstep executable for his job since, as mentioned in subsection 2.3, Quickstep does not yet belong to the DEISA CPE.

4. CPMD support by HLRS

This chapter refers to the contributions of the new DEISA site HLRS. HLRS has contributed support for optimized use of CPMD on their vector architecture, NEC SX-8.

Status

- CPMD v3.9.1 has been successfully ported to SX-8
- CPMD v3.9.1 (MPI version) has been installed on HLRS' SX-8 and is accessible for

users owning a valid license from www.cpmc.org
- The application is running well on this vector machine, and large cases especially profit from the SX-8 architecture

Work for SX version:

Porting the distribution of CPMD to the SX-8 basically involves adding special SX compilation rules for some subroutines to the Makefile. This is done by applying an AWK-script to the Makefile produced by the configure script of the distribution.

The special compilation rules enable inlining and higher optimization for some performance critical subroutines. Some BLAS subroutines are also inlined for some instances (original source code comes from www.netlib.org, also provided with the NEC MathKeisan-Library Distribution) in order to reduce the calling overhead for small routines.

Some performance- and scalability studies have been performed. Initial tests on SX-6+ and comparisons on SX-8 have been conducted with moderate sized test cases. In addition, pre-production studies have been performed with rather large test cases. Results are illustrated in Figures 12 and 13.

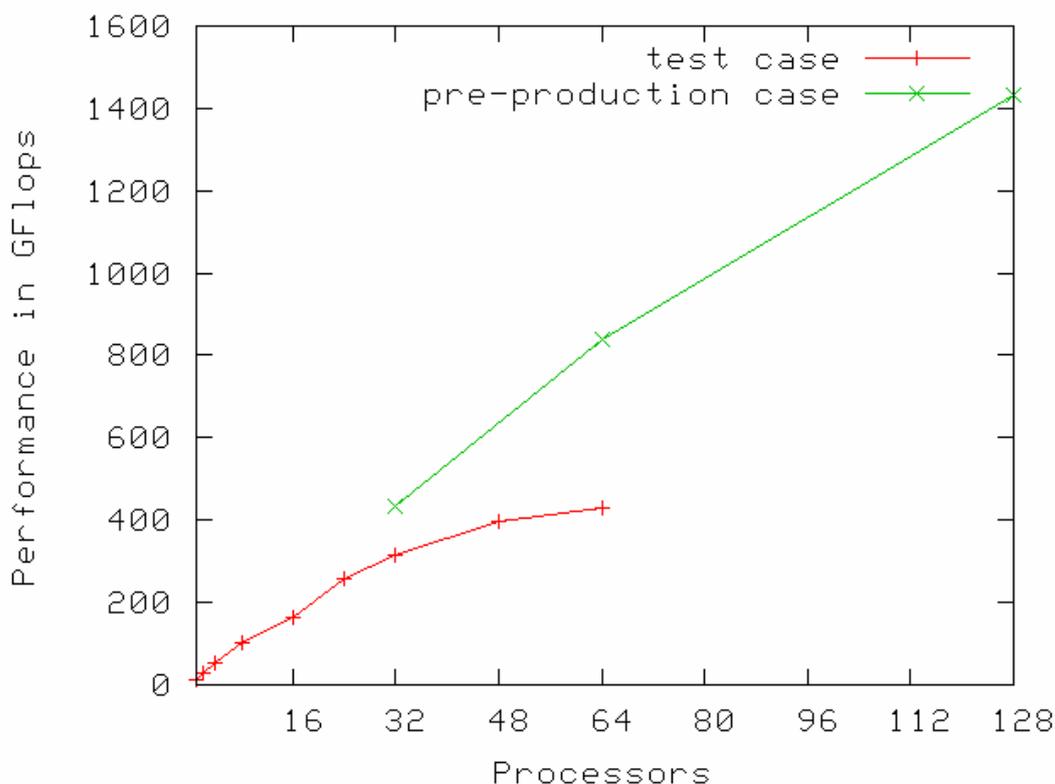


Figure 12: Performance results for CPMD on NEC SX-8 for small test cases and large (pre-production) cases

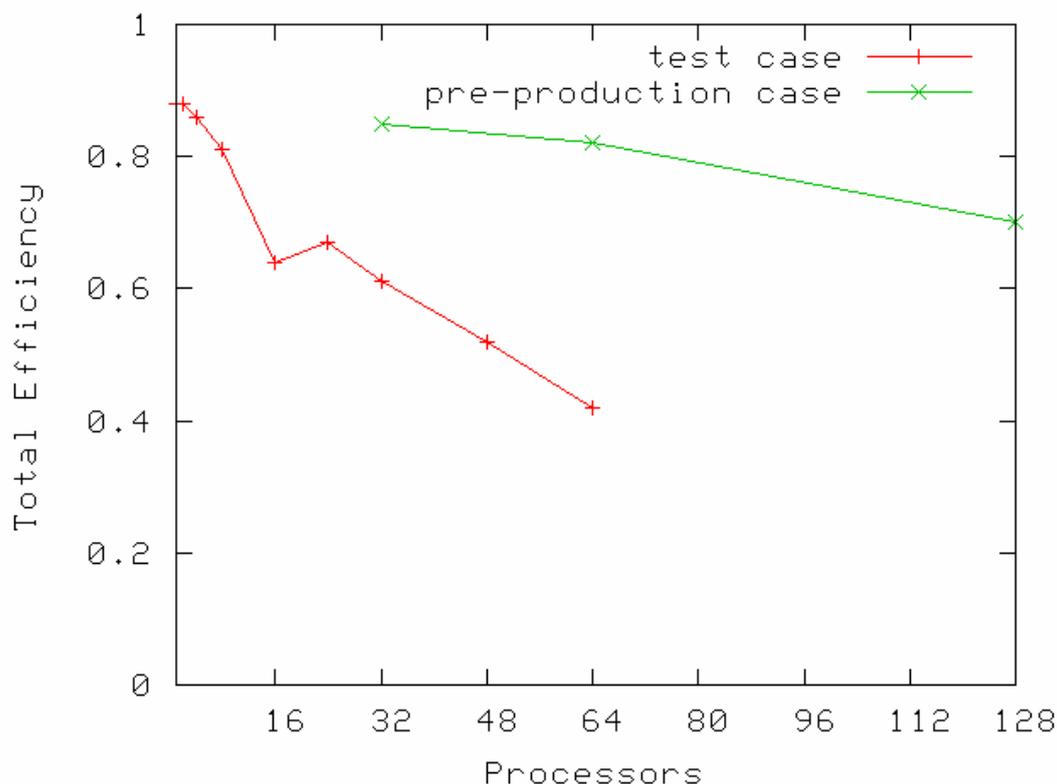


Figure 13: Performance results from Fig. 12 for CPMD on NEC SX-8 for small test cases and large (pre-production) cases, represented as total efficiencies with respect to the peak performance (total efficiency =1)

As CPMD performs large sequential I/O of a binary RESTART file, tests have been conducted to find good parameters for FORTRAN I/O buffersizes to reduce this bottleneck. This is especially important for big cases (latest test cases are in the range of 20-50 GB) on large numbers of processors. The 'Real Time' for reading was improved by 40%, and for writing by 65%, compared to the default system settings.

Initial tests have been conducted with the hybrid parallel version of CPMD using MPI and OpenMP. The current version shows better performance for large processor numbers compared to the pure MPI version, while for small and medium processor numbers the pure MPI version was found to be superior.

5. Screening for further relevant codes

A screening for further important materials science codes of superior relevance has been carried out, explicitly taking into account the demands for application enabling formulated in DECI proposals. As candidates the following codes have been marked:

- GROMACS (<http://www.gromacs.org>) (DECI project SNARE)
- ESPResSo (<http://www.espresso.mpg.de>) (DECI project POLYRES)
- LAMMPS (<http://www.cs.sandia.gov/~sjplimp/lammps.html>) (DECI project LIAMS)

- NAMD (<http://www.ks.uiuc.edu/Research/namd/>) (DECI project LIAMS)
- NWCHEM (<http://www.emsl.pnl.gov/docs/nwchem/nwchem.html>)

Some of the codes will be used by different DECI projects. In the following we describe the main purpose of the codes. A decision on work plan of making these applications generally available will be provided in month 30.

5.1 GROMACS

GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of particles. It is primarily designed for biochemical molecules like proteins and lipids that have a lot of complicated bonded interactions, but since GROMACS is extremely fast at calculating the non-bonded interactions (that usually dominate simulations) many groups are also using it for research on non-biological systems, e.g. polymers. GROMACS will be used in the DECI project SNARE.

5.2 ESPResSo

Espresso has already been described in JRA1-2 and is mentioned here only for completeness. It is one of the applications used in the DECI project POLYRES.

5.3 LAMMPS

LAMMPS is a classical molecular dynamics simulation code designed to run efficiently on parallel computers. It was developed at Sandia National Laboratories, a US Department of Energy facility, with funding from the DOE. It is an open-source code, distributed freely under the terms of the GNU Public License (GPL). LAMMPS will be one of the applications used in the DECI project LIAMS.

5.4 NAMD

NAMD, recipient of a [2002 Gordon Bell Award](#), is a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems. NAMD will be one of the applications used in the DECI project LIAMS.

5.5 NWCHEM

Like ESPResSo, NWChem has already been described in JRA1-2 and is mentioned here only for completeness.

6. Conclusion and Outlook

We have implemented a Quickstep plug-in for the JRA1/JRA3 Portal solution for ease of use of the Quickstep simulation code within DEISA. This plug-in initially serves the users as an aid for submitting Quickstep jobs to the DEISA computing infrastructure. Due to its novel design, the authors are optimistic that this plug-in will also help users to prepare new Quickstep projects.

The solution described here allows users to submit Quickstep jobs to all UNICORE enabled DEISA sites. With respect to porting and optimization tasks there is no need for action from JRA1 side since this is taken care of by the Quickstep development

team. As soon as signalled by the authors, Quickstep will be provided in the DEISA CPE.

The next 6 month will be dedicated to including all remaining parts of the CP2K computation suite into the plug-in. Furthermore, we intend to release a large part of this portal plug-in, together with a basic version of the JRA1/JRA3 portal framework, to the CP2K user community under the GPL open source license. The goal is to facilitate usage of CP2K in DEISA and to support users in coping with the new syntax of the input files.

7. Acknowledgement

We thank the developers of the CP2K suite for their support and the possibility of having such a fruitful exchange of ideas.