

# Code Migration on DEISA

Gavin J. Pringle\*, Odysseas Bournas, Elena Breitmoser, Terence M. Sloan, Arthur Trew  
EPCC, University of Edinburgh, UK

\* all correspondence: [g.pringle@epcc.ed.ac.uk](mailto:g.pringle@epcc.ed.ac.uk)

## Abstract

The Distributed European Infrastructure for Supercomputing Applications, or DEISA, is a consortium of leading national supercomputing centres that deploys and operates a persistent, production quality, distributed supercomputing environment with continental scope. Code Migration, where a large simulation begins running on one platform and finishes on another, is key for any large simulation which requires more time than any single batch job permits. The authors have successfully implemented Code Migration within DEISA, invoked using either UNICORE or the DESHL. The UNICORE Workflows GUI allows for automatic job submission and the necessary data staging, using a predetermined set of platforms. For the DESHL, which might be considered a command line interface to UNICORE, dependencies between tasks cannot be declared explicitly. Therefore, we employ DESHL-bash scripts, running locally, that manage job submissions, job monitoring and the staging of data.



## What is Code Migration?

Consider a very large simulation which starts running on one particular DEISA platform and finishes on another, different DEISA platform of a potentially different vendor. This process is referred to as Code Migration. This is not to be confused with Job Migration, where a job is submitted to one platform, and is migrated to another batch queue on another, separate platform.

Code migration allows simulations to migrate between DEISA platforms, which, considering the fact that many huge simulations take a long time to complete, is an important feature, particularly for they require more time to complete than any DEISA site permits within the scope of a single batch job.

When a simulation is submitted, the associated batch queues maximum time limit is passed to the code. At each time step, a check is performed at each iteration to measure how much time remains. If the remaining time is less than the time to run the next iteration, the code dumps the restart file and exits cleanly. The code is then automatically resubmitted to the DEISA infrastructure. The process repeats until the simulation is complete.

Here, we describe two methods of implementing Code Migration within DEISA, utilising elements of the DEISA infrastructure, namely the UNICORE Workflows or the DESHL, in conjunction with basic augmentations to the simulation code in question. This form of code migration has been successfully introduced to a large cosmological simulation, within the DEISA infrastructure, seamlessly utilising several of the DEISA platforms.

## Implementation of Code Migration

A Simulation Code should have the following four characteristics to be able to Migrate:

- Restart capability
  - The code must be able to create a restart files, where this file contains sufficient information for the same simulation to continue running from the point at which the restart file was created. These files are sometimes referred to as checkpoint files.
  - Typically, these files can be created at fixed intervals of: wall-clock seconds or at a fixed number of simulation time steps.
  - The code automatically employs the most recent restart file as its 'initial conditions'.
- Batch awareness
  - To be fully automated, the simulation must have a measure of how much time remains within its associated batch queue. This can be achieved quite simply: the code includes an internal timer which measures how long the code has been running. Then, by passing the associated wall clock limit to the code, as a runtime parameter, the code may check at every time step, say, to measure how much wall clock time remains.
- Portable Data files
  - Due the heterogeneous nature of modern supercomputing clusters, such as DEISA, the simulation must use utilise a portable data format, such as HDF5 or NetCDF.
- Pre-ported to all platforms to be employed.
  - The code is installed, by hand, on each of the participating platforms. This ensures that every invocation of the simulation will utilise an executable which has been tuned to run on the required number of processors.
  - Employing staged executables restricts the user to homogeneous platforms only and does not ensure that each executable is optimised nor hyperscaled. (In this context, to stage a file is to place it on a remote platform immediately prior to its use, and to hyperscale a code is to ensure that the code scales to thousands of processors.)

## UNICORE

The UNICORE Workflow feature allows for automatic job submission and the necessary data staging. The platforms utilised are currently predetermined, however, resource discovery can be implemented if brokering is available.

The user places all the necessary files required to run the simulation, such as the initial conditions, in a particular directory, namely the UNICORE `USPACE`. The executable, however, is automatically imported into the `USPACE` by UNICORE.

Once the simulation has completed its first pass on the first platform, the files associated with this initial run, which reside within `USPACE`, are automatically copied inside this second platform's `USPACE`, since this script is considered as a new job by the Workflow manager. The data is thus transported to the second platform automatically. This process continues until the simulation is complete, thereafter one final task (a bash script) exports the data out of `USPACE` to `$DEISA_HOME`, for instance.

This UNICORE Workflow described above was tested for a large cosmological simulation using HPCx and IDRIS as the first and second platforms of a two-platform code migration job. Since HPCx was not, at that time, part of the DEISA global file system, this latter experiment demonstrated that this works in the heterogeneous extended DEISA infrastructure.

## DESHL

From a user's point of view, DESHL might be considered as being a command line interface version of UNICORE, however, this is a significant over simplification. One major difference between DESHL and the UNICORE Workflow process is that dependencies between tasks cannot be declared explicitly. Thus we have created a bash script, which runs on a local workstation, that manages the job submissions, job monitoring and staging of data, using the DESHL as its engine. Using this script, we have successfully started the cosmological simulation at HPCx and completed it at IDRIS, where the initial input files and the final output files all reside at the local workstation.

The key stages of this particular script are now described in some detail.

- Stage the startup files in the HPCx `USPACE`, using the `deshl copy` command. As with the UNICORE example, the executable is installed at the target platform by hand. Note that, when employing the `deshl copy` command, wildcards can only be used when staging data into the `USPACE`. At present, wildcards cannot be employed when copying data out of `USPACE`. Therefore, we stage files via a bash script. This script creates a tar file of all the files to be staged, which is then moved to the target machine by DESHL and another script unpacks the tar file upon arrival. Indeed, this method may be faster than employing wildcards for a large number of large files.
- The executable is then submitted to HPCx on the local workstation using `deshl submit` command.
- The job's status is then determined by repeatedly calling the `deshl status` command at a fixed interval. This is currently set to one minute. A `sed` operation is performed on the output to isolate the 'state', where the state can be `Running` (which actually means running or pending), `DoneOK` or `DoneFailed`.
- Once the simulation has completed successfully, the restart file is taken back to the workstation using the `deshl fetch` command.
- The second job restart file is then staged to the IDRIS `USPACE` using the tar script and `deshl copy`.
- The executable is then submitted to run at IDRIS using the `deshl submit` command.
- Again, the status is monitored by polling the IDRIS batch queues with repeated calls to the `deshl status` command.
- Once the simulation has completed successfully, the output files are taken from the IDRIS `USPACE` back to the local workstation using `deshl fetch`.

## Current and Future Work

### Resubmit Suite

A Resubmission Suite is one where the simulation is launched and, if the queue time limit expires before the simulation has ended, the Suite repeatedly resubmits the job until the simulation is complete. This is simply a reduced form of Code Migration, where only one site is employed. A small template version of this in both UNICORE and DESHL formats are available from the authors.

### Simulation Workflows

A Simulation Workflow is a simulation which has many executable, run sequentially, where the output of one forms the input to the next. This is a common scenario for UNICORE users. A Simulation Workflow is a simple extension of the Code Migration suite and a small template version in DESHL format is available from the authors.

### DESHL Taskfarms

A DESHL taskfarm, for multiple, independent parallel jobs is currently under development at EPCC, using the DESHL Code Migration bash scripts as its basis.

Here, the user's workstation is the master and the groups of worker processors are DEISA platforms. As before, we assume that the executable has been installed, by hand, on each of the participating DEISA platforms. The master submits a number of parallel jobs to each of the participating platforms and then monitors each job. Once a job has completed, the output is retrieved to the user's workstation and the next job sent to that platform.

### Brokering

If a broker is introduced, then Code Migration could be employed to repeatedly submit a job to the next available platform, rather than a particular platform. This would significantly reduce the user's time-to-solution, as no time is spent waiting in batch.

Furthermore, if the requested batch queues are small, i.e. a small number of processors and/or a short length of time, then this method could be used to soak up any spare cycles being offered as national services drain their queues when switching from different modes of operation, i.e. switching between day and night modes, where a day mode typically has many smaller queues, whilst a night mode might disable all interactive queues. There are also modes when a whole machine must be drained when preparing for huge simulations which require the whole machine, i.e. preparing for Capacity Runs.

Soaking up these spare cycles is, effectively, cycle scavenging, and would contribute to higher utilisation figures of all participating sites.

## Conclusion

Allowing simulations to migrate between DEISA platforms is a very important feature for any large simulations which require more time to run than any DEISA site permits within a single batch job.

Thus, in the near future, users will submit multiple, very long/large simulations, from their workstation, which will run on any number of remote HPC platforms, without the need for knowledge of the local vagaries of any of the platforms nor the need to nurse the simulation suite to completion.

We thank the DEISA Consortium, co-funded by the EU, FP6 projects 508830/031513.

- [1] DEISA: <http://www.deisa.org>  
 [2] UNICORE: <http://www.unicore.eu>  
 [3] DESHL: <http://deisa-jra7.forge.nesc.ac.uk>

## What is DEISA?

The Distributed European Infrastructure for Supercomputing Applications [1], or DEISA, is a consortium of leading national supercomputing centres that currently deploys and operates a persistent, production quality, distributed supercomputing environment with continental scope. The purpose of this FP6-funded research infrastructure is to enable scientific discovery across a broad spectrum of science and technology, by enhancing and reinforcing European capabilities in the area of high performance computing. This becomes possible through a deep integration of existing national high-end platforms, tightly coupled by a dedicated network and supported by innovative system and grid software.

DEISA consists of a heterogeneous cluster of large HPC systems, including IBM Regatta clusters with Power4, Power4+ and Power5 processors, an IBM PowerPC cluster, an NEC SX-8, two SGI Altix platforms and a Cray XT4. The participating partners are BSC (Spain), CINECA (Italy), CSC (Finland), ECMWF (International Organisation), EPCC/HPCx (UK), FZJ (Germany), HLRS (Germany), IDRIS-CNRS (France), LRZ (Germany), RZG (Germany), BSC (Spain) and SARA (The Netherlands).

The Global Parallel File System, or GPFS, has been extended to work across DEISA's heterogeneous cluster. This presents the user with a single view of the distributed file system. Four file systems are defined and accessible: `$HOME`, `$DEISA_HOME`, `$DEISA_DATA` and `$DEISA_SCRATCH`. Accessing data in these file systems in jobs, consistently over all sites, must not use absolute path names but should employ these environment variables to address the location of data or programs.

## DEISA User Interfaces

There are several user interfaces to the DEISA infrastructure, including UNICORE and the DESHL.

UNICORE [2] provides a seamless interface for preparing and submitting jobs to a wide variety of computing resources. It has a three-tier design:

- A UNICORE client GUI is used for the preparation, submission, monitoring, and administration of jobs.
- The Gateway is a site's point of contact for all UNICORE connections. It also checks if the user's certificate is signed by a trusted CA. Site specific information on computing resources, including the availability of applications, is provided by a Network Job Scheduler (NJS). This server dispatches the jobs to a dedicated target machine or cluster, and handles dependencies and data transfers for complex workflows. It transfers the results of executed jobs from the target machine.
- A Target System Interface (TSI) is a server that runs on the target machine interfacing to the specific batch scheduler. TSI servers are currently running on 11 machines in DEISA interfacing various batch schedulers on different platforms: CINECA, CSC, FZJ, IDRIS and RZG employ the multi-cluster LoadLeveler (LL-MC) under AIX, ECMWF and EPCC run LoadLeveler (LL) under AIX, BSC runs LoadLeveler under Linux, HLRS employs NQS II under Super-UX, LRZ uses PBS Pro under Linux and SARA runs LSF under Linux.

The DESHL - DEISA Services for the Heterogeneous management Layer [3] - was developed by the DEISA JRA7 activity to complement the UNICORE GUI as a means of exploiting the DEISA infrastructure. The DESHL provides users with job and data management capabilities via an Open Group standards-based command line interface and a Java Application Programming Interface (API) based on the Open Grid Forum's (OGF) Simple API for Grid Applications (SAGA) standard. DESHL was the first publicly available Java implementation of SAGA and was the first to be used in an HPC production environment.

Through its use of UNICORE, the DESHL shields users from the underlying platform and batch scheduler heterogeneity of the DEISA infrastructure. It also even further insulates the users from the UNICORE middleware itself. The use of standards at the command line and API levels allows users to interact in a grid-independent manner at the application, command line and batch script level.