John von Neumann Institute for Computing

Submission Scripts for Scientific Simulations on DEISA

Gavin J. Pringle, Terence M. Sloan, Elena Breitmoser,
Odysseas Bournas, Arthur S. Trew

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr,
F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 705-711, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

http://www.fz-juelich.de/nic-series/volume38

Submission Scripts for Scientific Simulations on DEISA

Gavin J. Pringle, Terence M. Sloan, Elena Breitmoser, Odysseas Bournas, and Arthur S. Trew

EPCC, University of Edinburgh
Mayfield Road, Edinburgh, EH9 3JZ, UK
E-mail: {g.pringle, t.sloan, e.breitmoser, o.bournas, a.trew}@epcc.ed.ac.uk

The DEISA Services for the Heterogeneous management Layer, better known as the DESHL, allows users and their applications to manage batch jobs and data across a computing Grid in a uniform manner regardless of the underlying hardware and software on the various nodes. The DESHL provides a means for coordinating and integrating services for distributed resource management in a heterogeneous supercomputing environment. It provides users and their applications with a command line tool and application programming interfaces for job and data management in DEISA's UNICORE-based grid.

The DESHL employs the SAGA (Simple API for Grid Applications) and JSDL (Job Submission Description Language) emerging grid standards as well as the Open Group Batch Environment Services specification. Reliance on Grid standards will allow interoperability with other Grid environments.

Three DESHL-based bash scripts have been created which demonstrate how DEISA can be exploited directly from the user's workstation. These scripts are for Task Farms, Workflows, and for Code Migration/Resubmission.

1 Introduction

DEISA[2] - the Distributed European Infrastructure for Supercomputing Applications - is currently a Grid of eleven supercomputers spread across eight different European countries, where homogeneous access is given to this Grid of heterogeneous supercomputers.

There are several user interfaces to the DEISA infrastructure, including UNICORE[3] and the DESHL[1,4], or DEISA Services for the Heterogeneous management Layer.

A UNICORE GUI provides a seamless interface for preparing and submitting jobs to a wide variety of computing resources. An alternative command-line interface is provided by the DESHL.

The creation of the DESHL was driven by the fact that a large number of users prefer to use the command line, rather than a GUI, when managing their supercomputing simulations. These users are typically more accustomed to accessing HPC resource via the command line, however, some users simply cannot use the mouse for physical reasons and require command line access.

A light weight DESHL client is run on the users workstation, Linux/Solaris or Windows/DOS, where a user can transfer files to and from their workstation and any of the DEISA sites, and also submit and monitor DEISA batch jobs. The client sits on top of the existing UNICORE infrastructure, but is not dependent on the UNICORE graphical client.

Thus, the DESHL provides simplified and seamless access to DEISA resources, without needing to log into any individual site. Indeed, under the DEISA access model, direct login to any site other than the users home site is not permitted. Authentication/Authorisation is by the existing UNICORE mechanisms: any valid X.509 certificates for UNICORE are also valid certificates for DESHL.

A set of DESHL-based bash scripts have been created to permit users to launch supercomputing applications on DEISA from their workstation, without the need for multiple usernames and passwords, nor the need to learn the various flavours of compilers, batch systems or performance characteristics of each of the various platforms.

There are four bash script suites available, which can be employed for Code Resubmission, Code Migration, Simulation Workflows and Task Farms.

This paper presents the basics of the DESHL frameworks, and then contains a detailed discussion of each of the four suites.

2 The DESHL

The DESHL allows users and their applications to manage batch jobs and data across a computing grid in a uniform manner regardless of the underlying hardware and software on the various nodes of a grid. The DESHL employs emerging grid standards so that a user and their applications are not affected by the differences or changes in the hardware and software configuration on the grid nodes.

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 work in a grid-independent manner at the application, command line and batch script level.

The DESHL software has been developed by DEISA's seventh Joint Research Activity (JRA7). EPCC and ECMWF from the UK, FZJ from Germany and CINECA from Italy are the participants in this research activity. The DEISA JRA7 activity is using modern Grid standards to develop a means for coordinating and integrating services for distributed resource management in a heterogeneous supercomputing environment.

The DESHL employs the SAGA[5] (Simple API for Grid Applications) and JSDL[6] (Job Submission Description Language) emerging grid standards as well as the Open Group Batch Environment Services specification[7]. Reliance on Grid standards will allow interoperability with other Grid environments.

The DESHL command line tool job management and file transfer capabilities include: determining the DEISA sites to which a user can submit a batch job to; submitting a batch job to a DEISA site; terminating a batch job at a DEISA site; viewing the status of a batch job on a DEISA site; uploading a file to a DEISA site; down-loading a file from a DEISA site; deleting a file from a DEISA site; determining if a file exists on a DEISA site; listing the contents of a directory on a DEISA site; renaming a file on a DEISA site; and, copying/moving a file between DEISA sites.

The DESHL supports the command line tool operations through a SAGA-based API, the DESHL Client library, which simplifies access to heterogeneous computing and data resources. This SAGA API interfaces with the more powerful and complex Grid Access library (also known as Roctupus) that in turn interacts with the low-level UNICORE specific library, ARCON[8]. Any of these APIs can be used by Java applications developers as appropriate.

The DESHL also includes a GUI-based installer, an installation and user manual, a design description, and API documentation for the DESHL Client and Grid Access libraries.

The DESHL Client library was the first publicly-available Java-based implementation

of file and job management using the proposed SAGA grid standard. More importantly, it was the first SAGA implementation to be actively deployed, tested against and used with a production-level, continental grid infrastructure and thus represents an important contribution to global Grid computing.

2.1 UNICORE

The DESHL sits on top of the UNICORE-based DEISA Grid, where UNICORE has a three-tier design: the client GUI, the Gateway and the Target System Interface. 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 and is typically the user's so-called Home Site. It also checks if the user's certificate is signed by a trusted CA. Execution Site specific information on the computing resources, including the availability of applications, is provided by a Network Job Scheduler (NJS). This server dispatches the jobs to a set of dedicated target machines or clusters, known as the Execution Sites, 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 each of the Execution Sites, interfacing to each specific batch scheduler.

3 DESHL-Based Bash Scripts

Four suites of DESHL-based bash scripts have been developed to demonstrate the DESHL's ability to perform Code Resubmission, Code Migration, Workflow Simulations and Task Farms. All four suites are freely available from the authors.

Code Resubmission is where an application is launched and, if the queue time limit expires before the simulation has completed, the script repeatedly resubmits the job.

Code Migration is where an application is launched on one platform and completes on another. This is not to be confused with Job Migration, where a job is submitted to 'local' platform's batch system, and is migrated to another batch queue on a 'remote' platform where the job runs on this 'remote' platform.

Workflow Simulations are multi-step simulations which may also have many executables, which run in a given sequence, where the output of one forms the input to the next. This is a common scenario for UNICORE users.

Task Farms are a collection of independent parallel jobs, where the master coordinates groups of processors to concurrently execute each job.

3.1 Code Resubmission

A Resubmission Suite is one where the application is launched and, if the queue time limit expires before the simulation has ended, the Resubmission Suite repeatedly resubmits the job until the simulation is complete. Very large simulations may require several days to complete and, by default, no DEISA site currently permits jobs longer than 24 hours.

We have developed a DESHL-based bash script which the user can employ to submit such a large job. The job is submitted only once. The suite performs the work normally carried out by the user in such circumstances, namely job monitoring and simulation restarts.

The process is as follows: the batch queue time limit is passed to the application upon submission. The application itself then regularly checks how much time remains. If time is short, then the application dumps the restart file and exists cleanly. During this time, the Resubmission Suite has been monitoring the simulation and, once the application is found to have exited, then the simulation is submitted again. The process repeats until the simulation is complete.

A Simulation Code should have the following two characteristics to be able to migrate:

Restart capability: the application code must be able to create a restart file, 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 application automatically employs the most recent restart file as its 'initial conditions'.

Batch awareness: to be fully automated, the application code must have a measure of how much time remains within its associated batch queue. This can be achieved quite simply: the application code includes an internal timer which measures how long the simulation has been running. Then, by passing the associated wall clock limit to the application, as a runtime parameter, the application may check at, say, every time step to measure how much wall clock time remains. The batch queue time limit is passed to the application upon submission.

3.2 Code Migration

Code Migration is an extension of the Code Resubmission, where multiple sites are employed rather than just a single site. It is an important feature for simulations that require more resources than any DEISA site offers, such as disk space or computer cycles. We refer to this process 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.

As with Code Resubmission, the associated batch queues maximum time limit is passed to the application, and the application dumps the restart file automatically before the time limit is reached. However, in for Code Migration, the application is then automatically resubmitted to the *next* platform to be employed within the DEISA infrastructure. The process repeats until the simulation is complete.

As for Code Resubmission, the application code itself must have certain capabilities.

Restart capability: à la Code Resubmission.

Batch awareness: à la Code Resubmission.

Portable Data files: due to the heterogeneous nature of DEISA, the application must utilise a portable (binary) data format, for example, HDF5 or NetCDF. This will ensure that the restart file does not restrict the choice of platform.

Pre-ported application: the application is installed, by DEISA staff, on each of the participating platforms. This ensures each invocation will utilise a tuned application. Employing staged executables restricts the user to homogeneous platforms only and does not ensure that each executable is optimised nor hyper-scaled. (In this context, to stage a file is to place it on a remote platform immediately prior to its use, and to hyper-scale an application code is to ensure that the code scales to thousands of processors.)

3.3 Simulation Workflows

A Simulation Workflow is a simulation which has many executables, run in a prescribed order, where the output of one forms the input to the next. This is a common scenario for UNICORE users. A Simulation Workflow is a more simple form of the Code Migration Suite where different application codes run on each site, rather than the same code. The DESHL can be employed to implement all of the UNICORE workflow schema, such as 'do-while', 'repeat-until', 'if-then', etc.

If resubmission is not required, then the only restrictions are that portable binary data files must be employed and that executables should be pre-installed rather than staged.

3.4 Task Farms

A DESHL-based Task Farm Suite, for multiple independent parallel jobs, has been created. Here, the user's workstation assumes the role of the master and the each group of worker processors are individual DEISA platforms.

As before, employing portable (binary) data input files is required, so that we are not limited by our choice of platforms. Further, we also assume that the executable has been installed, by DEISA staff, on each of the participating DEISA platforms.

The process is as follows: 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 particular platform.

Note that DEISA platforms limit the number of jobs that can be submitted at any given time, where this limit can range from one to tens of jobs, depending on the each site's local configuration. This site-dependent variation is hidden from the user by the DESHL script and the user simply sees the Task Farm running to completion.

Currently, the Task Farm Suite assumes that each job runs within a single site, however, this restriction can be lifted if the job is actually two applications which are loosely-coupled, as DEISA supports co-scheduling.

It is worth noting that Task Farms are currently not possible with UNICORE GUI.

4 Pseudo-Code of the Code Migration Suite

This section contains a step-by-step discussion of how the Code Migration Suite proceeds. The pseudo-code of the other Suites are not presented as the other Suites are straightforward extensions of this Code Migration Suite.

We assume that the user runs the DESHL client on his local workstation, where this client manages the job submissions, job monitoring and staging of data. Further, we assume that only two platforms are required for a two part job. Lastly, all input and output files are stored locally, on the user's workstation.

Stage the startup files. The input files are sent from the user's workstation to the first platform's `USPACE`, using the `deshl copy` command. Here `USPACE`, or User SPACE, is a UNICORE construct and is a automatically created, hidden directory reserved for use by UNICORE and, hence, the DESHL, and is used as a temporary scratch directory. 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, which creates a tar file of all the files to be staged, which is then moved to the target machine by DESHL. Then, on the target machine, another script unpacks the tar file upon arrival. Indeed, this method may be faster than employing wildcards for a large number of large files.

Run first part of the job. The executable is then submitted to the first platform, at the local workstation, using the `deshl submit` command.

Monitor Job Status. 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), `Done` or `Failed`.

Once Complete, Retrieve Restart File. Once the application has closed successfully, the restart file is copied back to the workstation using the `deshl fetch` command.

Stage the startup files The restart files required for the second half of this job are then staged to the second platform's `USPACE` using the tar script and the `deshl copy` command. NB: the restart files can be staged from the USPACE of the first platform directly to the USPACE of the second platform, however, in this example, the user wishes to examine the state of the intermediate restart file.

Run last part of the job The executable is then submitted to run at the second platform using the `deshl submit` command.

Monitor Job Status Again, the status is monitored by polling the second platform's batch queues with repeated calls to the `deshl status` command.

Once Complete, Retrieve Results Once the simulation has completed successfully, the output files are taken from the second platform's `USPACE` back to the local workstation using the `deshl fetch` command.

5 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.

Further, if the requested batch queues are small, i.e. a small number of processors and/or a short length of time, then the Task Farm Suite 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 Capability Runs.

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

6 Conclusion

In this paper, we have described the basic form of the DESHL, or the DEISA Services for the Heterogeneous management Layer.

We then described four Suites, formed from DESHL-based bash scripts, namely Code Resubmission, Code Migration, Workflow Simulations and Task Farm, along with the necessary alterations required to the application code(s). All four suites are freely available from the authors.

Thus, users can now submit multiple, very long/large simulations, with complex workflows directly from their workstation, which will run on any number of remote platforms with DEISA's heterogeneous supercomputing Grid. What makes this more remarkable is that the user does not need knowledge of the local vagaries of any of the platforms, only requires a single DEISA Certificate, and does not need to monitor nor nurse the simulation to completion.

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

References

1. T. M. Sloan, R. Menday, T. Seed, M. Illingworth and A. S. Trew, *DESHL - standards-based access to a heterogeneous European supercomputing infrastructure*, in: Proc. 2nd IEEE International Conference on e-Science and Grid Computing - eScience 2006, 4th–6th December, Amsterdam, (2006).
2. DEISA: Distributed European Infrastructure for Supercomputing Applications, `http://www.deisa.eu`
3. UNICORE, `http://www.unicore.eu`
4. DESHL, `http://deisa-jra7.forge.nesc.ac.uk`
5. OGF (Open Grid Forum) SAGA (Simple API for Grid Applications) Research Group. `https://forge.gridforum.org/projects/saga-rg`
6. OGF (Open Grid Forum) JSDL (Job Submission Description Language) Working Group. `http://forge.gridforum.org/projects/jsdl-wg`
7. Open Group specification for Batch Environment Services. `http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap03.html`
8. UNICORE 5 ARCON Client Library documentation. `http://www.unicore.eu/documentation/api/unicore5/arcon`