The GDASH Utility for DASH Version 3.1

Introduction

GDASH is a program designed to speed-up the execution of *DASH* by allowing it to be run on a Univa UD Grid MP system of distributed computing resources. It is used, from a PC, to submit, monitor and retrieve *DASH* jobs to the Grid MP system. For example, consider a *DASH* job that consists of 200 SA runs, each using 1×10^7 SA moves. Such a job might take 24 hours to execute on a CPU with a single core e.g. Intel Core 2 Solo 2.4GHz. By running the same *DASH* job on a Grid MP system with 200 clients attached (each containing an Intel Core 2 Solo 2.4GHz chip) execution time is cut to ~10 minutes. Not only can results be obtained more quickly, but more challenging optimisations can be undertaken e.g. more degrees-of-freedom, multiple possible models.

Note that *GDASH* only controls the submission and retrieval of *DASH* jobs that execute on the Grid MP system. It has no utility without a licenced copy of *DASH*. Unless otherwise stated, all references to *DASH* in this manual should be taken to mean *DASH* version 3.1

Requirements

In order to correctly deploy *GDASH*, you require the following:

a. a working installation of GridMP and the help of your Grid MP 'system administrator'
b. a copy of *DASH* 3.1 with a valid site or demo licence

GDASH itself runs under MS Windows XP (SP2) and MS Windows Vista. The GDASHInstaller requires the MS .Net 2.0 Framework (or higher) to be installed. This is present by default in XP SP2 and Vista.

NB: If any of the computers on your Grid MP system have a version of *DASH* that pre-dates version 3.1, you are advised to remove *DASH* from those computers before running *DASH* 3.1 on the grid. This is because the licence files for the older versions of *DASH* will cause *DASH* 3.1, running under Grid MP, to fail with an 'invalid licence' message. Note that computers on the grid that have *DASH* 3.1 already installed with a valid licence will not cause any problems.

Installation steps

There are four steps to installing *GDASH*

a. Ensure that you have successfully installed *DASH* 3.1 on your PC.
b. Gather together some information about your Grid MP setup (see 'Grid MP pre-requisites').
c. Unpack the *GDASH Install Package.zip* file into some temporary directory from which the installation can then be run.
d. Use the *GDASHInstaller* program to install and register *DASH* 3.1 as a program on the Grid MP server(s).
e. Use the *GDASHInstaller* program to install the *GDASH* client on your PC in order to submit and retrieve jobs that you create with your local copy of *DASH* 3.1.

Grid MP pre-requisites

You will require the following information, all of which will be known to your Grid MP system administrator, in order to successfully install *GDASH*.

- Grid MP MGSI server URL
- Grid MP file server URL
- Grid MP username with 'Add Application' permission
- Password for the above username

Furthermore, you will also require access to the following files, which are supplied as part of the Grid MP SDK; again, your Grid MP system administrator will be able to provide them.

- buildmodule.exe
- buildpkg.exe
- loader.exe

Installation

Once you have gathered together all the required information, and have unzipped the *GDASH Install Package.zip* file, run the *GDASHInstaller* and select 'Install DASH on Grid MP'. Supply the required information and the installer will automatically create the necessary application / program on the Grid MP system.

Once *DASH 3.1* is installed on the Grid MP system, select 'Install *GDASH* on this computer'. Again, supply the requested information and *GDASH* will be installed on your local computer. The installer will setup the necessary environment variables in order to let you access both *DASH 3.1* and *GDASH* on your local machine from a DOS command prompt. A file, DASH.mpconfig, is created in the same directory as the *GDASH* executable; the role of this vital control file is explained at the end of this document.

Post-installation hint

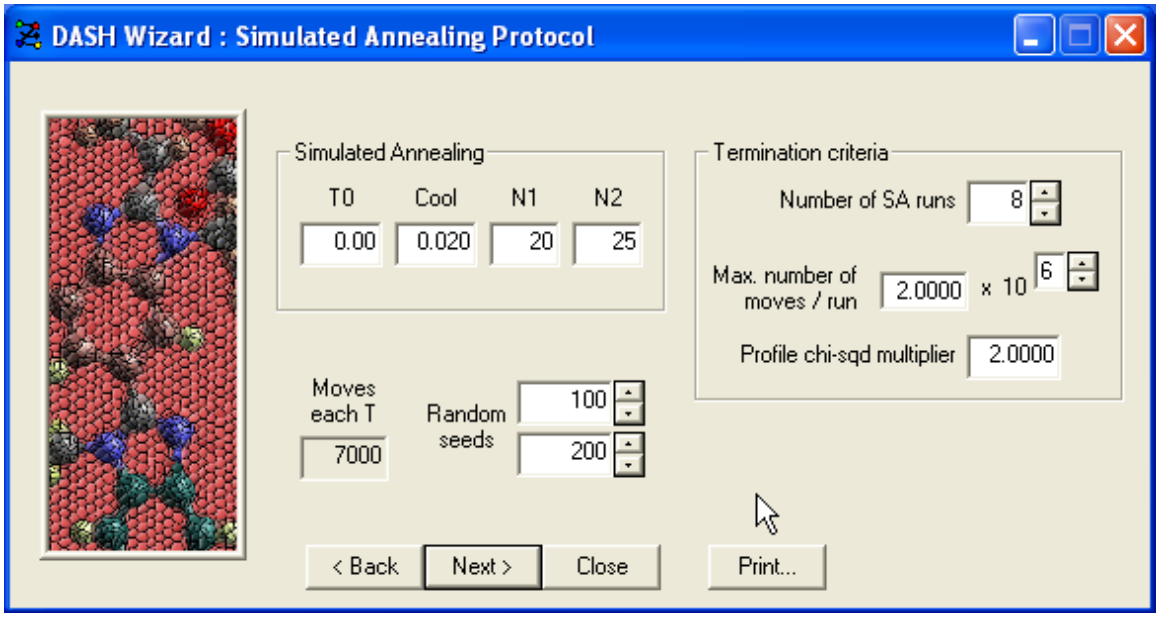
As *GDASH* is invoked from the command line, MS Vista users will find its 'Open command window here' facility extremely valuable to allow a command window to be opened in a specific directory. MS Windows XP users may wish to install the equivalent functionality provided as part of the 'PowerToys' supplied by Microsoft.
For more details, see:

http://www.microsoft.com/windowsxp/downloads/powertoys/xppowertoys.mspx

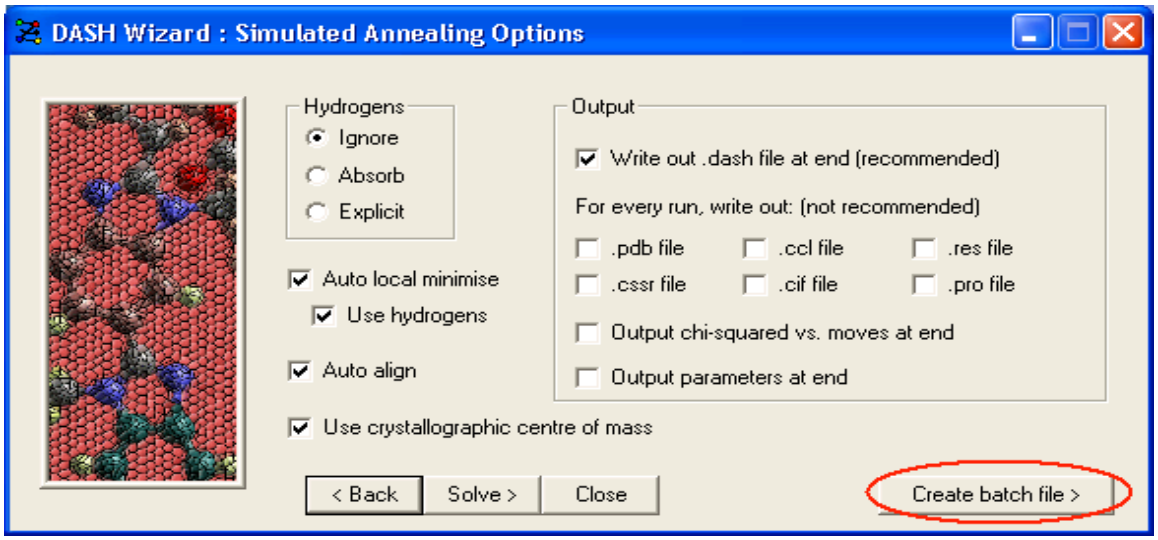
Setting up a *DASH* grid job

NB. This section assumes you are familiar with using DASH on a Windows PC and have already prepared the files necessary to start simulated annealing runs.

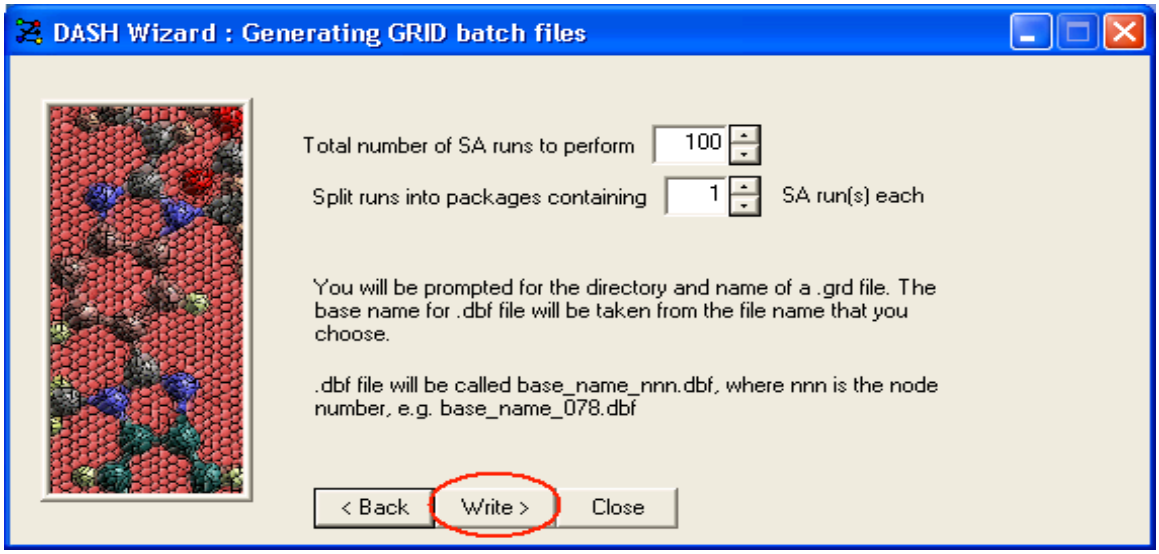
Start *DASH* in the normal way. At the 'Welcome to DASH' screen, click 'Simulated annealing structure solution'. Select your .sdi and .zmatrix files, click 'Next' to go to the next screen. Continue to set up your *DASH* run as usual until you come to the 'Simulated Annealing Protocol Page'



Set up your run as normal, ensuring you have set appropriate random seeds, number of moves / run and profile chi-squared multiplier. Click 'Next' to go to the next screen.

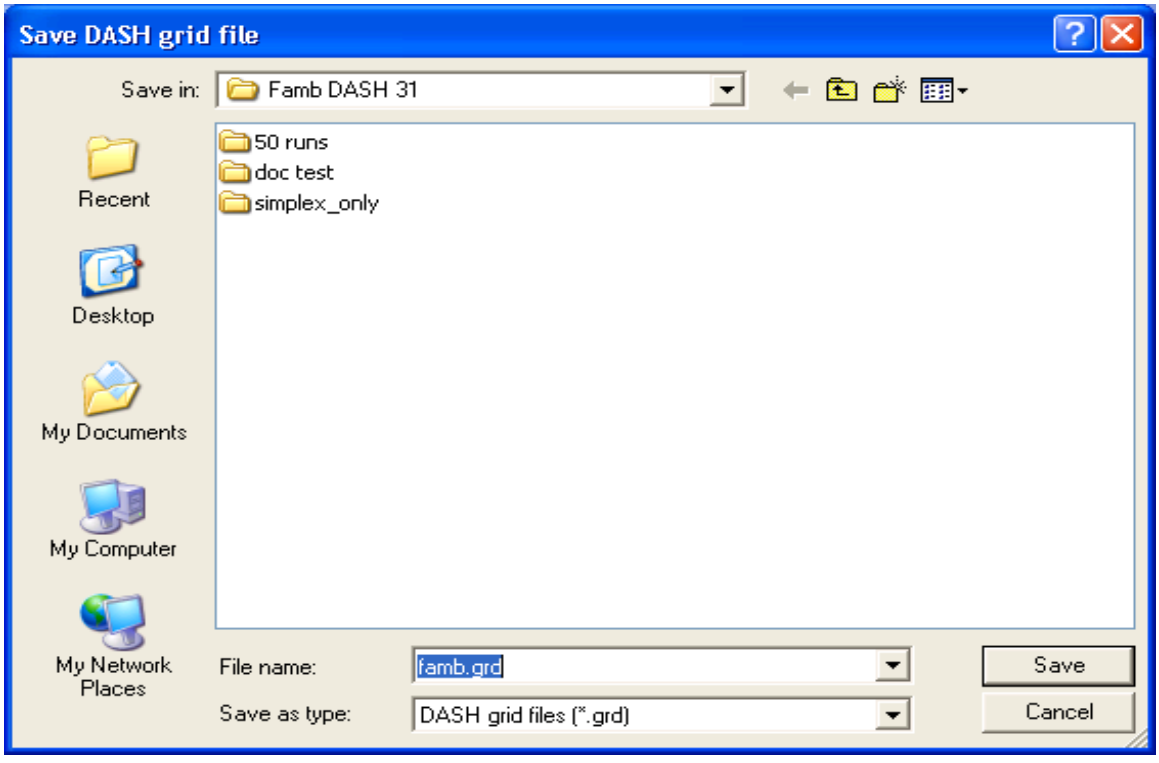


Set the options you require, ensuring 'Write out .dash file at end' is selected. Then click 'Create batch file…'

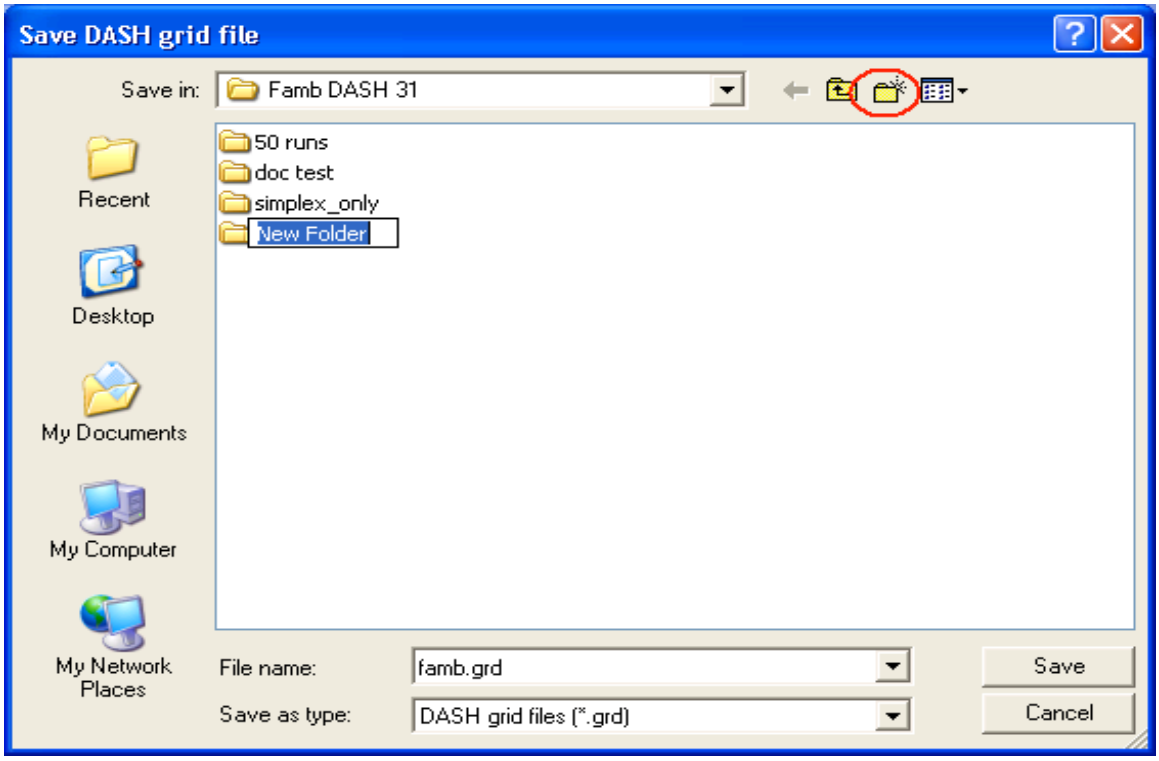


Enter the values you require for 'Total Number of SA' runs and 'number of packages'. Typically, the package size will be '1' i.e. each copy of *DASH* that is sent out to the grid clients will have one complete *DASH* SA run to execute. Make sure that the packages you create take a sensible amount of time to run - roughly 5 minutes to a few hours. See *Tips For Running DASH on Grid MP* at the end of this document for more details on package sizes.

Click 'Write' to create the grid files. A Save File Dialogue box will open.



To keep files separate, you may want to create a new folder to save the grid files in.



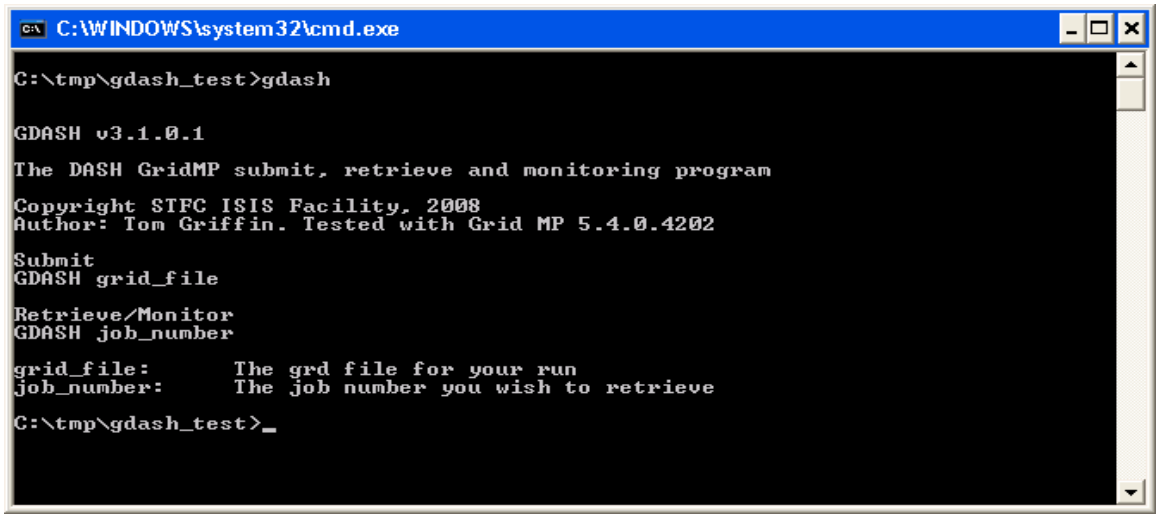
Give the new folder a name, then double-click to move into it, before pressing 'Save' to create the grid files. A message will inform you these have been written.

Click OK to return to the previous screen. You can now exit *DASH*. If you look in the directory you just created, you will find the following files.

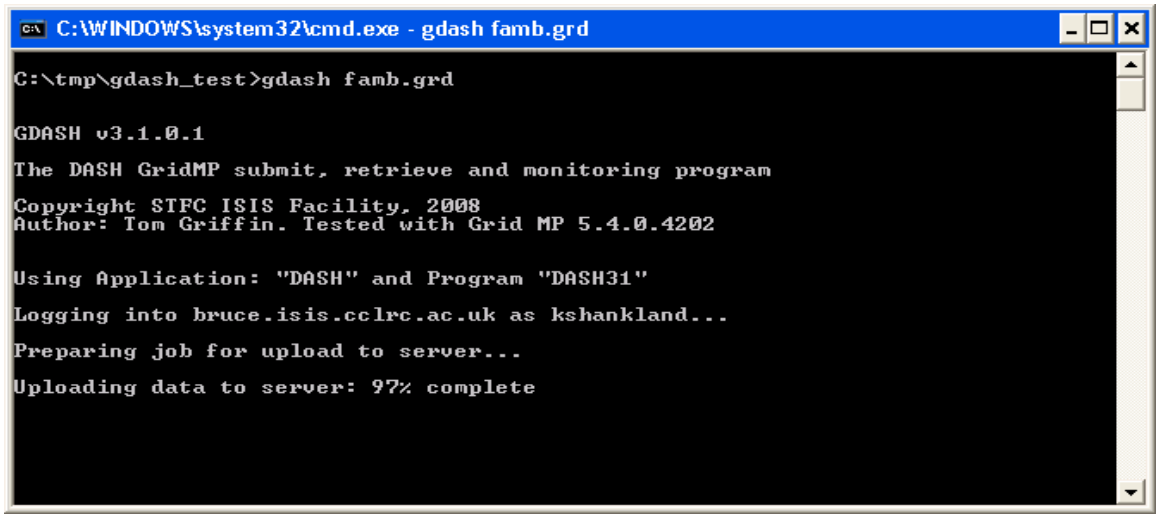
SDI, HCV, PIK, TIC, DSL: Files needed for *DASH* to operate on
GRD: A text file containing a list of the *DASH* .dbf files to be executed
DBF: Each DBF file is a text file containing a set of instructions to control *DASH* when it is running in command-line mode (as it must do on the grid)

Running *GDASH* to submit your DASH job

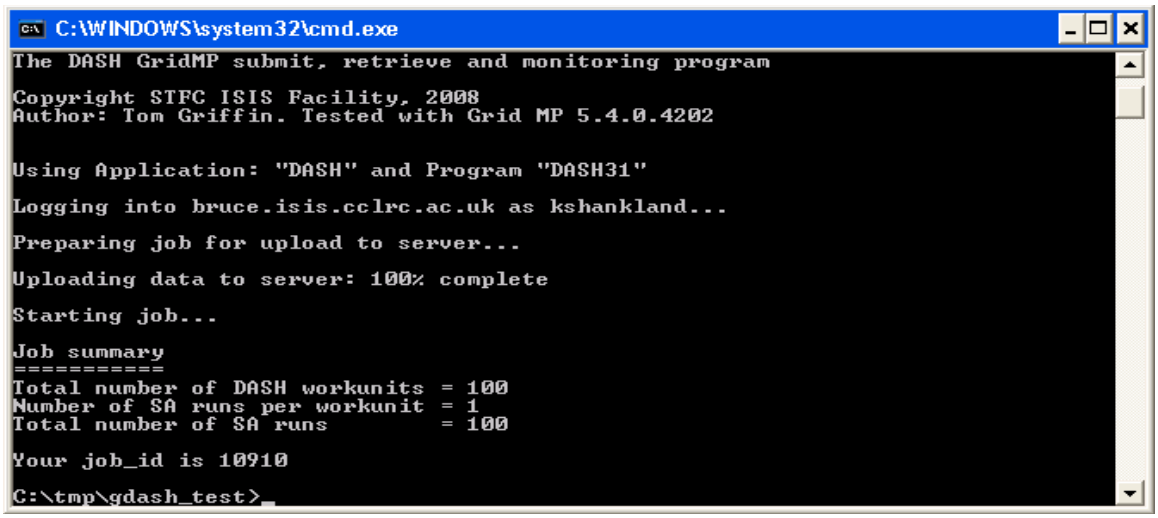
Firstly, open a command prompt and navigate to the directory where you saved the grid files created in *DASH* (you may find the 'Open command prompt here' functionality of Windows very useful in this regard – see the hint at the start of this manual). Typing 'gdash' at the command prompt returns a brief summary of how to run *GDASH*.



To submit the job you created, simply type 'gdash *filename*.grd', where *filename*.grd is the name of the .grd file you created using *DASH*.



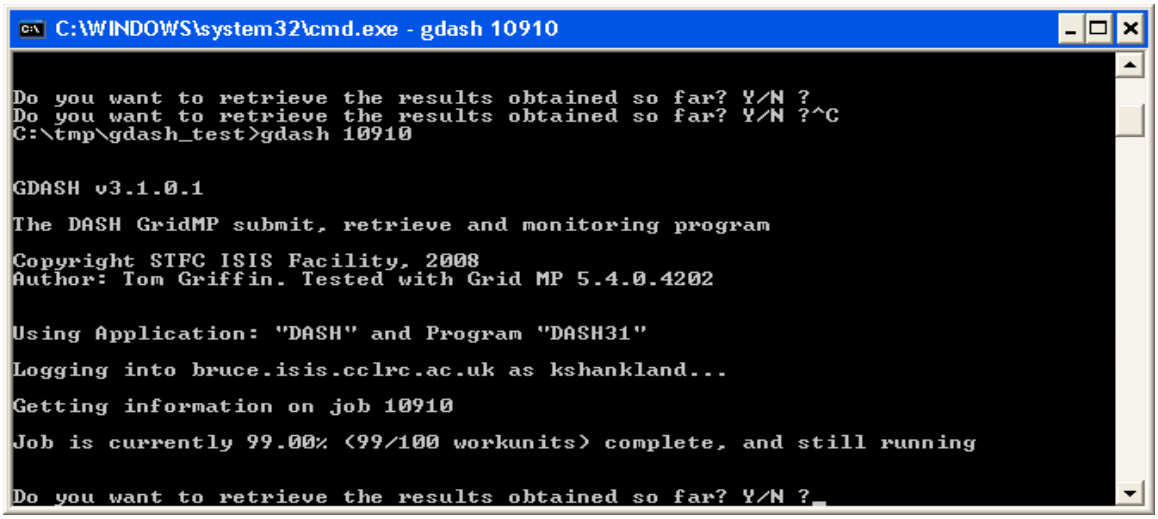
The program displays the progress of job submission; once all files have been uploaded to the server, the job is started on the grid and a summary of the job returned.



Note in particular the 'job_id'. You will need this number in order to monitor and retrieve your *DASH* job (NB: If you lose or misplace this, you can always find it by logging onto the Grid MP management console; see 'Monitoring your *DASH* Job using Grid MP Management Console').

Monitoring your *DASH* Job using *GDASH*

To monitor the progress of your job, type 'gdash' followed by the job number.



GDASH returns the status of the job. If 100% complete, it automatically retrieves the *DASH* results (see 'Retrieving your *DASH* Job using *GDASH*'). If not, it returns the '% completeness' and offers the ability to download those results already completed.

Monitoring your *DASH* Job using Grid MP Management Console

Your can monitor the progress of your *DASH* job by logging into the Grid MP Management Console (e.g. https://<*yourGridMPserver*>). To log on, enter the username and password; these are the same as the ones you used when setting up the *GDASH* client installation. They can also be found in the DASH.mpconfig file, C:\Program Files\CCDC\GDASH\DASH.mpconfig.

Clicking 'Manage Jobs' will show you a list of your jobs currently running on the Grid. Clicking the name of your job will bring up detailed information about its status.

The most important field in this screen is **Status**: this shows if your job is still running or if it is completed. When a job transitions to **Completed,** the entire set of *DASH* results may be downloaded and the job deleted from the Grid MP server(s) if desired.

Retrieving your *DASH* Job using *GDASH*

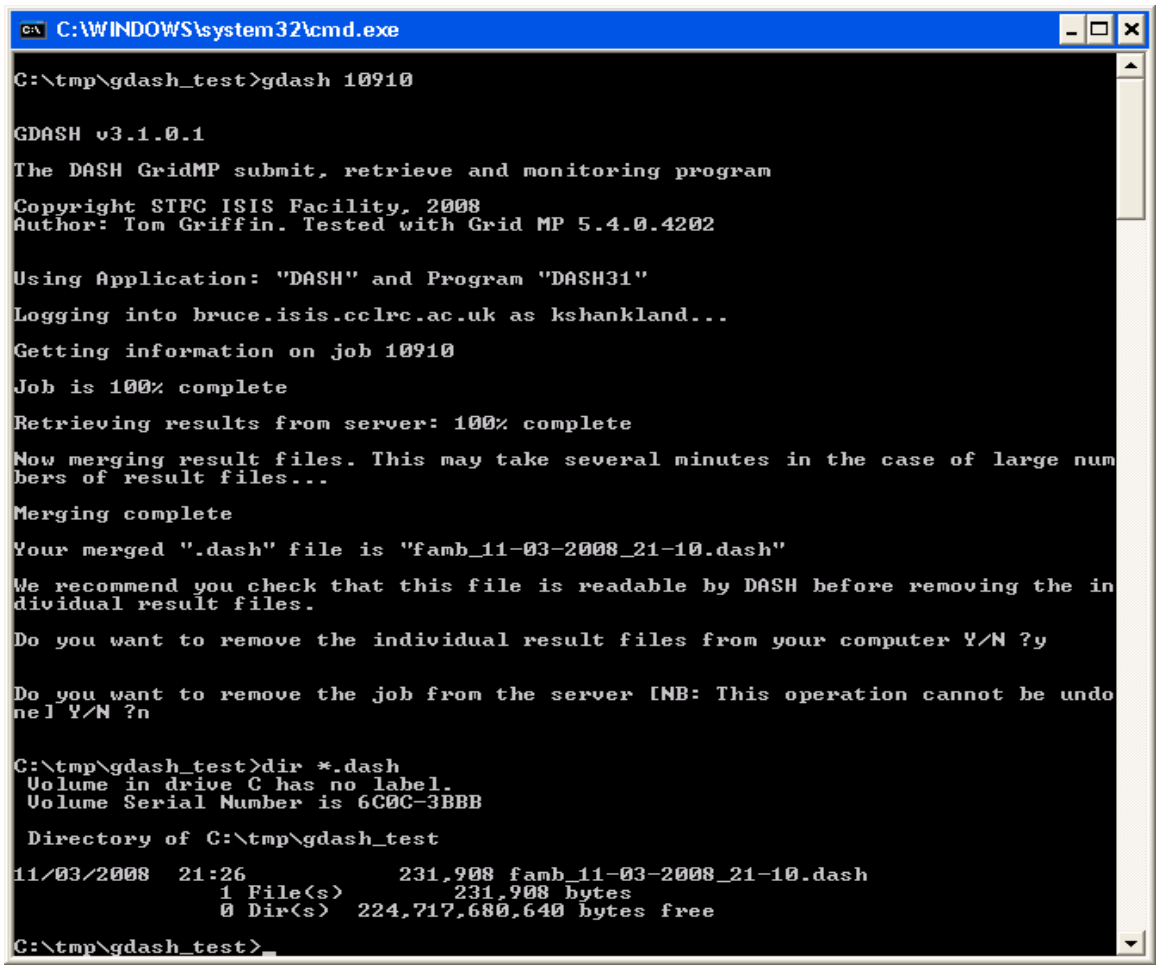
To retrieve the results of your job, type 'gdash' followed by the job number. The results (.dash) files are downloaded from the server to your PC and then merged into a single *DASH* file automatically.

You are given the option of deleting the individual results files from their temporary working sub-directory (recommended, to save disk space and keep your filesystem tidy), but are advised to do so only after checking that the merged .dash file is readable by *DASH* (simply double-click on the file to open it in *DASH*). You are then given the option of deleting the job from the grid server. If you do not delete the job, it remains on the server (occupying server disk space) until it is deleted either via the Management Console or by running *GDASH* to first download the results again.

As stated in 'Monitoring your *DASH* Job using *GDASH'*, if the job is not complete, you can still download a partial set of results. In such cases, *GDASH* does not give you the option of deleting the job from the server.

Note that the .dash file is assigned a name that incorporates the date and time of submission of the job.

NB: Jobs may not always run to 100% completion, depending upon a number of factors. You can always use *GDASH* to retrieve those results that are complete.


C:\WINDOWS\system32\cmd.exe
C:\tmp\gdash_test>gdash 10910
GDASH v3.1.0.1
The DASH GridMP submit, retrieve and monitoring program
Copyright STFC ISIS Facility, 2008
Author: Tom Griffin. Tested with Grid MP 5.4.0.4202
Using Application: "DASH" and Program "DASH31"
Logging into bruce.isis.cclrc.ac.uk as kshankland...
Getting information on job 10910
Job is 100% complete
Retrieving results from server: 100% complete
Now merging result files. This may take several minutes in the case of large numbers of result files...
Merging complete
Your merged ".dash" file is "famb_11-03-2008_21-10.dash"
We recommend you check that this file is readable by DASH before removing the individual result files.
Do you want to remove the individual result files from your computer Y/N ?y
Do you want to remove the job from the server [NB: This operation cannot be undone] Y/N ?n
C:\tmp\gdash_test>dir *.dash
Volume in drive C has no label.
Volume Serial Number is 6C0C-3BBB
Directory of C:\tmp\gdash_test
11/03/2008 21:26 231,908 famb_11-03-2008_21-10.dash
1 File(s) 231,908 bytes
0 Dir(s) 224,717,680,640 bytes free
C:\tmp\gdash_test>

Tips for running *DASH* on Grid MP

Avoiding grid job failures

- Always check that a single SA run can be completed in *DASH* without any errors. You only need to set a relatively small number of SA moves (e.g. 1 x 10^6) in order to perform this check.

- Start the *DASH* grid job on your own PC using *MDASH* (see *MDASH* documentation for how to do this) to quickly check for errors in the job before actually submitting the job to the grid.

- Make a note of how long your single SA run (see first point, above) takes to complete on your PC. This can help with setting the timeouts in the mpconfig file. For example, if 1 x 10^6 SA moves take 3 minutes to complete on your 'average specification' PC, and your grid job specifies that each workunit has 20 x 10^6 SA moves, you can estimate that a workunit will take around one hour to execute on an equivalent 'average specification' PC connected to the grid. If you estimate that a workunit might take longer to execute than the default timeout (10 hours) specified in the dash.mpconfig file, you may wish to either decrease the number of SA moves or increase the timeout. Remember that some machines on the grid may well be slower than the PC on which you run your timing tests.

Optimising package size for SA runs

- When creating a grid job in *DASH* you are asked how you would like to sub-divide your planned SA runs. Generally speaking, each package should contain only one SA run, as this enables maximum throughput to be obtained on a grid of machines of varying power.

- An exception to the above rule occurs when each individual SA run takes only a few tens of seconds or less to execute. In this scenario, the time taken by the grid server(s) to package up a workunit, send it out to the grid and then retrieve it can exceed the time taken to actually execute the SA code, leading to a net slowdown relative to single PC execution, rather than a speedup. It therefore makes sense to set the package size to be larger (say 10), such that each resultant workunit consists of at least several minutes of CPU time.

Possible GDASH error messages and associated solutions

Login

```
C:\example>GDASH example.grd
Logging into bruce.nd.rl.ac.uk as tom...
Error: -1 '(-1) Login credentials not valid. (login)'
```

Cause: Incorrect username / password combination

Solution: Check the details in DASH.mpconfig are correct.

Mpconfig file

```
Error reading DASH.mpconfig
```

Cause: GDASH was not able to read DASH.mpconfig correctly. DASH.mpconfig may be corrupt / incomplete.

Solution: Ensure DASH.mpconfig follows the rules in *The DASH.mpconfig* section.

File Not Found

```
Logging into bruce.nd.rl.ac.uk as tom...
Preparing job for upload to server...
Couldn't open: example_002.dbf
```

Where example_002.dbf is any file

Cause: A file which is required for the DASH run is not in the current directory.

Solution: Copy the required file into the directory, or re-run the DASH wizard to generate a new set of files

Network Error

```
Error: caught exception "103 'Connect failed'"
MGSI Error deleting job. Exiting anyway
```

or

```
Error: 0 'Could not resolve host name: bruce.nd.rl.ac.uk (MgsiClient:
tried 4 times)'
```

Cause: There is no network connection to the grid server.

Solution: Try connecting to the Web interface to your grid server. Resolve any network problems. Check that the server URLs in the DASH.mpconfig file are correct. If you can connect to the web server, but are still having problems with GDASH, contact your network or Grid administrator.

Application Not Found

```
Error:'(-4) Could not find specified Application record (DBC code:
DB_NO_DATA). (getApplicationByName)'
```

Cause: Unable to access the *DASH* program on the grid server. Either *DASH* is not on the server or the user does not have the necessary privileges to view applications and create jobs.

Solution: Install *DASH* on the grid server(s)

Solution: Ensure the user has the necessary privileges to read applications and create jobs

Known Issues

Issue: If a user's password is set to the keyword 'INTERACTIVE' in the mpconfig file, they will always be expected to type their real password before the job submission can proceed.

Workaround: Use a different password, or type it at the interactive prompt

Issue: GDASH uploads the same data files with each dbf file.

Workaround: This is by design: *GDASH* is designed to submit a single *DASH* job at a time, for submitting different *DASH* jobs, use multiple instances of *DASH*/*GDASH*

Frequently asked questions

I don't have Grid MP....can I use CONDOR instead?

Almost certainly yes, though we have not tried. Given that *DASH* is a relatively simple command-line driven program when operating in 'grid-mode', it should be fairly straightforward to adapt it to run under CONDOR. You will, of course, still require a site licenced version of *DASH*.

I submitted a job with 100 workunits and the job completed, but it returned '99/1/7' under the Results (succ/unsucc/Errors) column of the Jobs web page. What does this mean?

When a workunit is sent to a device there are three possible return types:

Successful (succ) – this means the workunit (i.e. the DASH program) returned an exit code indicating successful completion of the workunit.

Unsuccessful (unsucc) – this means the workunit (i.e. DASH program) returned an exit code indicating that the workunit failed to run correctly.

Both *Successful* and *Unsuccessful* results count towards the results for a workunit (i.e. an Unsuccessful result is not resent for execution on another machine)

Errors (errors) – this means that the workunit was not completed due to some problem with the grid agent on the client machine. Errors can be caused by a machine being switched off, the grid agent losing contact with the grid servers or some threshold / timeout being exceeded. When a workunit returns an error, it is resent to another machine, until such time as max_errors is exceeded)

In the above example

- 99 of the workunits completed successfully and returned .dash files for merging.
- 1 workunit was unsuccessful, and no .dash file was returned for this workunit.
- There were 7 errors, which the grid handled by resending these workunits for subsequent successful execution on other machines.

The DASH.mpconfig application configuration file

GDASH has the ability to control many of the grid parameters that are relevant to your DASH job; these can all be set in the DASH.mpconfig file, which is installed by default in directory

C:\Program Files\CCDC\GDASH

The file itself is a txt file such as shown below:

```
//The DASH Configuration file
//All lines must be intact.
Grid_Username: grid_power_user
Grid_Password:      power_user_password
Results_per_WU:     1
Max_Concurrent:     1
Max_Error:    3
Priority:     10
WU_Clock_Timeout:   36000
WU_CPU_Timeout:     36000
AppName: DASH
ProgName: DASH31
MGSI_FILESVR_URL:   https://bruce.isis.cclrc.ac.uk:28443/mgsi/filesvr.fcgi
MGSI_SOAP_URL:      https://bruce.isis.cclrc.ac.uk:18443/mgsi/rpc_soap.fcgi
```

It contains some fairly self-explanatory fields, such as the URLs for the grid servers and the username / password for connecting to the grid. Similarly, it is clear that the configuration file is for DASH31 jobs and that the DASH31 program belongs to an application group called DASH. The remaining fields are somewhat more cryptic to the casual user and are explained below.

Setting	**Example**	**Explanation**
Grid_Username	jsmith	Your username to log on to the Grid. This is the name under which DASH jobs will be submitted.
Grid_Password	pwd123	Your password to log on to the Grid. If you do not want you password to be stored in this plain-text file, use INTERACTIVE as the value, and you will be prompted for your real password each time GDASH is started.
Results_per_WU	1	The number of results you require from each workunit i.e. how many times you want that workunit to be run. A workunit is the small section of work sent out to a client and corresponds to a DASH run on a client. This value will normally be set to one. It would only ever be set to greater than one if you required multiple results per workunit. Note that as the random seed would be the same for each of these, the result would be identical, so this is only of use in an environment where you wish to check machines/results are not being tampered with
Max_Concurrent	2	The maximum number of copies of a workunit that you wish to run concurrently. This allows redundancy in the system, as one workunit can be executing on two (or more) machines at the same time. Therefore if one workunit fails due to a client being switched off, results will still be returned from other clients running the same workunit. Obviously, this redundancy comes at

		the expense of using additional CPU power.
Max_Error	3	The maximum number of errors you wish to tolerate per workunit. In this context, errors are not those generated by your program, but by problems on the grid, such as devices being switched off etc. Once a workunit has reached Max_Error, it will not be distributed to any more devices. Note that if a job fails to reach 100% completeness due to errors, you can always use GDASH to retrieve the completed results.
Priority	10	The priority of this job, relative to other jobs from the same user. 1 is Lowest priority, 100 highest. e.g. If you submit two jobs, 'A' and 'B', and you give 'A' priority 10 and B priority 60, workunits from job 'B' will be distributed to clients in preference to any of job 'A' workunits.
WU_Clock_Timeout	36000	The time (in seconds) a workunit is allowed to run for before it is considered to have failed. This is the elapsed time from a workunit entering a PC, regardless of how much CPU time it actually received. This is used to stop 'runaway' programs e.g. a program entering an infinite loop. It should be set to allow plenty of time for your program complete.
WU_CPU_Timeout	36000	The CPU time (in seconds) a workunit is allowed to consume before it is considered to have failed. This is the CPU time actually spent executing the workunit. This is used to stop 'runaway' programs e.g. a program entering an infinite loop. It should be set to allow plenty of time for your program complete.
AppName	DASH	The application on the Grid that your job calls. (Do not change unless you are sure)
ProgName	DASH31	The program within the application on the Grid that your job calls. (Do not change unless you are sure)
MGSI_FILESVR_URL	https://myserver.com:28443/mgsi/fsvr.fcgi	The location of the FileServer used by the Grid. (Do not change: this is fixed by the system administrator)
MGSI_SOAP_URL	https://myserver.com:18443/mgsi/rpc_soap.fcgi	The location of the SOAP Server used by the Grid. (Do not change: this is fixed by the system administrator)