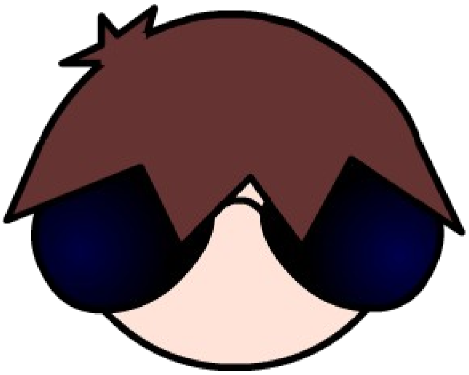


GridPilot
User's Guide

Frederik Orellana
Niels Bohr Institute
University of Copenhagen

May 2009

Table of Contents

Introduction

This guide covers the functionality of GridPilot -0.2.0.

GridPilot is a tool to facilitate various tasks related to grid computing. More precisely, GridPilot has a plugin architecture for running computing jobs on various execution back-end. Currently supported execution back-ends are:

- local forking of processes
- remote forking of processes via SSH
- forking of processes in a locally running virtual machine (via SSH)
- GridFactory
- Amazon Elastic Compute Cloud
- NorduGrid/ARC
- EGEE/gLite

In order to manage many jobs, bookkeeping information is kept in a database table. This table can be hosted on various job database back-ends. Supported job database back-ends are:

- a local in-memory Java SQL engine, HSQLDB

- a remote MySQL database with plain password or X509 authentication (using a personal grid certificate)

In order to manage the input and output of jobs, various file transfer systems are supported:

- the local file system (on both UNIX/Linux and Windows)
- HTTPS with X509 authentication of both client and server
- GridFTP
- SRM

Moreover, in order to catalog these files, various file and dataset catalogs are supported:

- a local HSQLDB catalog
- a remote MySQL catalog
- ATLAS DQ2
- LFC

The various databases can be searched and the results are displayed as tables. When appropriate, rows can be copy-pasted between these tables. This makes e.g. copying information from one file catalog to another a trivial operation for the user.

For GridPilot, a job is a script running on a machine somewhere. Typically a "production" of data will then involve one Linux shell script (calling calling one or several applications) running with different input parameters on many machines. Such a production will result in a number of output files, which together make up a *dataset*. The jobs are prepared and run by GridPilot, following the instructions of the user. These instructions are given by filling two records in database tables. The records to be fill in are:

- a **transformation record**: a record with fields like "script", "arguments" and "outputFiles", specifying the location of the script (on the local disk, on a web server or on a GridFTP server), the arguments to run this script (like e.g. `./myscript.sh my_input_file.txt 3`) and the names of the output files produced (and registered in a dataset/file catalog)
- a **dataset record**: a record with fields like "transformationName", "totalFiles" and "inputDataset", specifying the transformation used to produce this dataset, the number of files making up this dataset and the name of the dataset labeling the collection of input files

After this has been done, a number of **job definition records** are produced by GridPilot. These can be "submitted" to any of the execution back-ends, which will result in output files that can be registered as **file records**.

Getting started

License

GridPilot is provided under the terms of the GNU General Public License, available at http://www.gnu.org/licenses/gpl.html. This means that GridPilot is provided "as is" - the authors of GridPilot do not take any responsibility what so ever for any consequence of its use. The source code of GridPilot is available at http://www.gridpilot.dk/ or upon request.

Requirements

General

- SUN's Java Runtime Environment (JRE) 1.6.0_07 or higher. Get it here
- 500 MB of RAM or more

For using grid resources

- an X509 certificate with and accompanying secret RSA key

For using remote databases

- a fast and stable Internet connection: preferably 256 Mb/s or more, both ways

Installing/upgrading

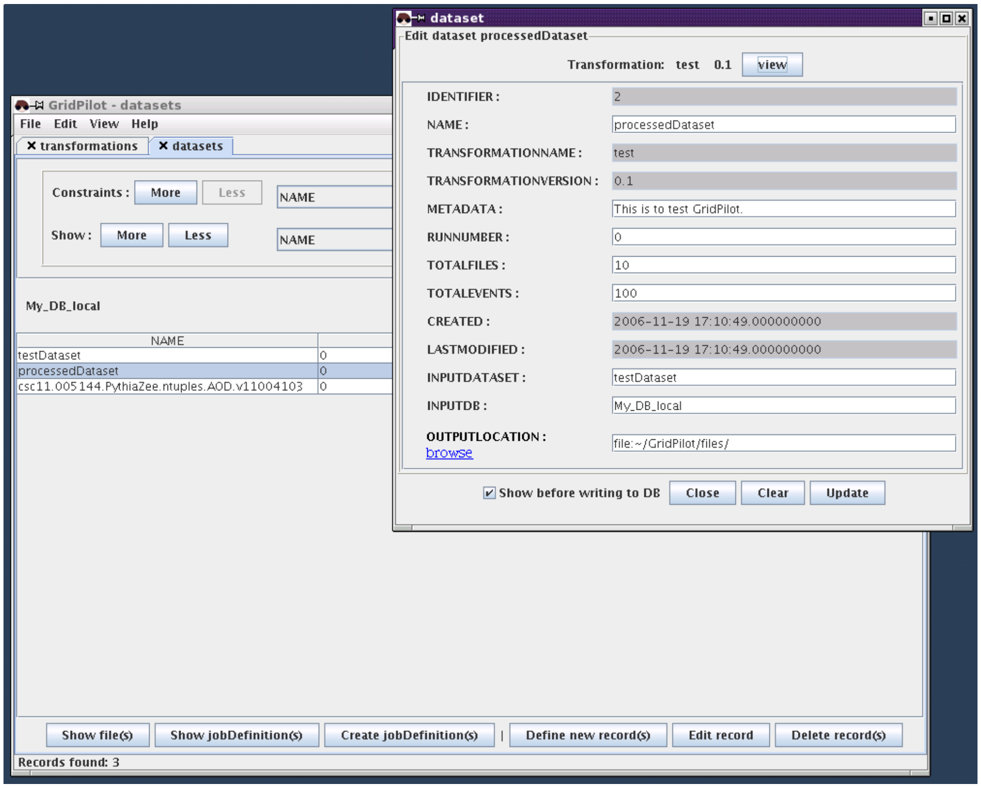
To install:

- download the appropriate installer, zip file or tarball from http://www.gridpilot.dk/
- run the installer
- run GridPilot – this will trigger a series of configuration questions

If you're upgrading from a previous version, you should first delete or rename the configuration file ".gridpilot" (under UNIX/LInux) or "gridpilot.conf" (under MS Windows) and the directory "GridPilot". After running the new version of GridPilot for the first time, you may then migrate your old configuration settings and/or files in the directory "GridPilot" by hand.

The user interface

The main window



NAME	
testDataset	0
processedDataset	0
csc11.005144.PythiaZee.ntuples.AOD.v11004103	0

IDENTIFIER :	2
NAME :	processedDataset
TRANSFORMATIONNAME :	test
TRANSFORMATIONVERSION :	0.1
METADATA :	This is to test GridPilot.
RUNNUMBER :	0
TOTALFILES :	10
TOTALEVENTS :	100
CREATED :	2006-11-19 17:10:49.000000000
LASTMODIFIED :	2006-11-19 17:10:49.000000000
INPUTDATASET :	testDataset
INPUTDB :	My_DB_local
OUTPUTLOCATION :	file:~/GridPilot/files/

The main window and a dataset editing window.

The main window holds tabs that each contain a search interface for one of the database tables of "runtimeEnvironments", "transformations", "datasets", "jobDefinitions" or "files". When GridPilot starts, this window holds one or several tabs (specified in the configuration file). Tabs can be closed by clicking the small cross. New tabs can be opened by selecting "View" → [database] → [table]. Tabs can be rearranged by dragging and dropping.

The upper part of a tab holds the query interface. Queries can be narrowed by adding more constraints (internally, the constraints are combined with a logical and).

The lower part of a tab holds a table displaying the search results. Clicking on the column names causes the list to be sorted according to the values of that column. Selecting one or several row and right-clicking will bring up a menu of actions that can be performed on these record(s). One possibility is to edit or view an individual record. This is done by double-clicking on the corresponding row, using the right-click menu or the button at the bottom of the tab, which will bring up a small window with full record.

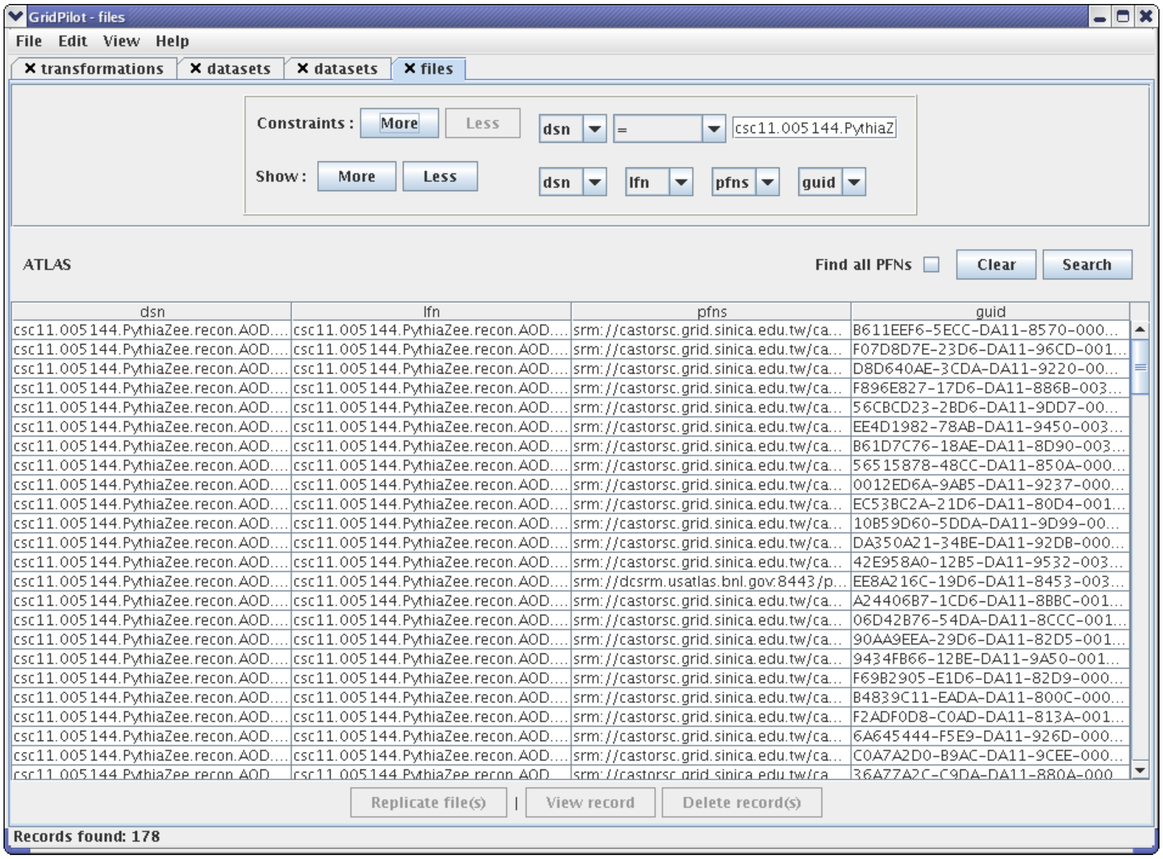
Of the 5 possible table types, only two should need manual editing: "transformations" and "datasets". The records of the "jobDefinitions" table are also generated by the user, but in an automatic fashion. These three tables will be discussed in turn in the section "Running jobs". The "runtimeEnvironments" table type and the "files" table type are not meant to, but *can* be edited by the user. Notice, however, that records from the "runtimeEnvironments" table that were generated automatically by GridPilot on startup, will be deleted when GridPilot exits.

A **runtime environment table** contains a list of runtime environments that can be selected when defining a transformation. This list is populated by the computing system plugins on startup. For example:

- the NorduGrid/ARC plugin queries the NorduGrid information system for installed runtime environments on the clusters where jobs can be executed with the active grid certificate
- the EGEE/gLite plugin queries the EGEE information system for installed runtime environments on the clusters where jobs can be executed with the active grid certificate
- the GridFactory, EC2 and *Fork* plugins query the defined runtime catalog URLs for installable runtime environments. You can add entries to a catalog with “Help” → ”Wizards: create software package”

Remark: The “runtime catalog URLs” is defined in the top-level of the configuration file. A runtime catalog is a text file in XML format containing information about runtime environments (software packages): where the software can be downloaded and on which other runtime environments it depends, etc. Currently, such catalogs are supported only by the GridFactory, EC2 and *Fork* plugins, but in the future also ARC sites will be able to subscribe to such catalogs .

A **file table** contains a list of files registered in the associated database back-end. Such a table is typically populated by GridPilot after determining that a job has finished successfully and its output files copied to a storage element. GridPilot can also register existing files that are not registered anywhere, replicate files and add the new locations to existing records or copy records from one database to another by simple copy-paste.

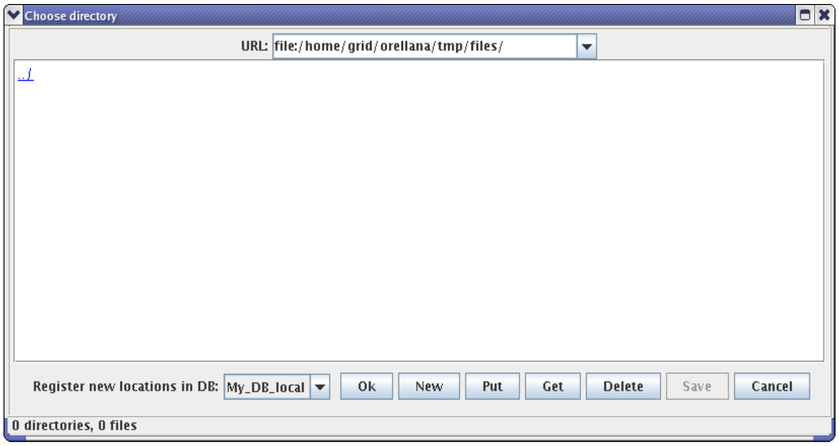


dsn	lfn	pfns	guid
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	B611EEF6-5ECC-DA11-8570-000...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	F07D8D7E-23D6-DA11-96CD-001...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	D8D640AE-3CDA-DA11-9220-00...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	F896E827-17D6-DA11-886B-003...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	56CBCD23-2BD6-DA11-9DD7-00...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	EE4D1982-78AB-DA11-9450-003...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	B61D7C76-18AE-DA11-8D90-003...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	56515878-48CC-DA11-850A-000...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	0012ED6A-9AB5-DA11-9237-000...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	EC53BC2A-21D6-DA11-80D4-001...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	10B59D60-5DDA-DA11-9D99-00...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	DA350A21-34BE-DA11-92DB-000...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	42E958A0-12B5-DA11-9532-003...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://dcsrm.usatlas.bnl.gov:8443/p...	EE8A216C-19D6-DA11-8453-003...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	A24406B7-1CD6-DA11-8BBC-001...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	06D42B76-54DA-DA11-8CCC-001...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	90AA9EEA-29D6-DA11-82D5-001...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	9434FB66-12BE-DA11-9A50-001...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	F69B2905-E1D6-DA11-82D9-000...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	B4839C11-EADA-DA11-800C-000...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	F2ADF0D8-C0AD-DA11-813A-001...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	6A645444-F5E9-DA11-926D-000...
csc11.005144.PythiaZee.recon.AOD....	csc11.005144.PythiaZee.recon.AOD....	srm://castorsc.grid.sinica.edu.tw/ca...	C0A7A2D0-B9AC-DA11-9CEE-000...
csc11.005144.PythiaZee.recon.AOD	csc11.005144.PythiaZee.recon.AOD	srm://castorsc.grid.sinica.edu.tw/ca	36A77A2C-C9DA-DA11-880A-000

Records found: 178

A file catalog tab in the main window.

Comment: technically, a "files" table is actually a "virtual table"; that is, it does not correspond to *one* table in a database. Instead, the file table can correspond to the physical tables "t_lfn", "t_pfn" and "t_meta"; or it is generated on the fly from the output files listed in the job definition table.

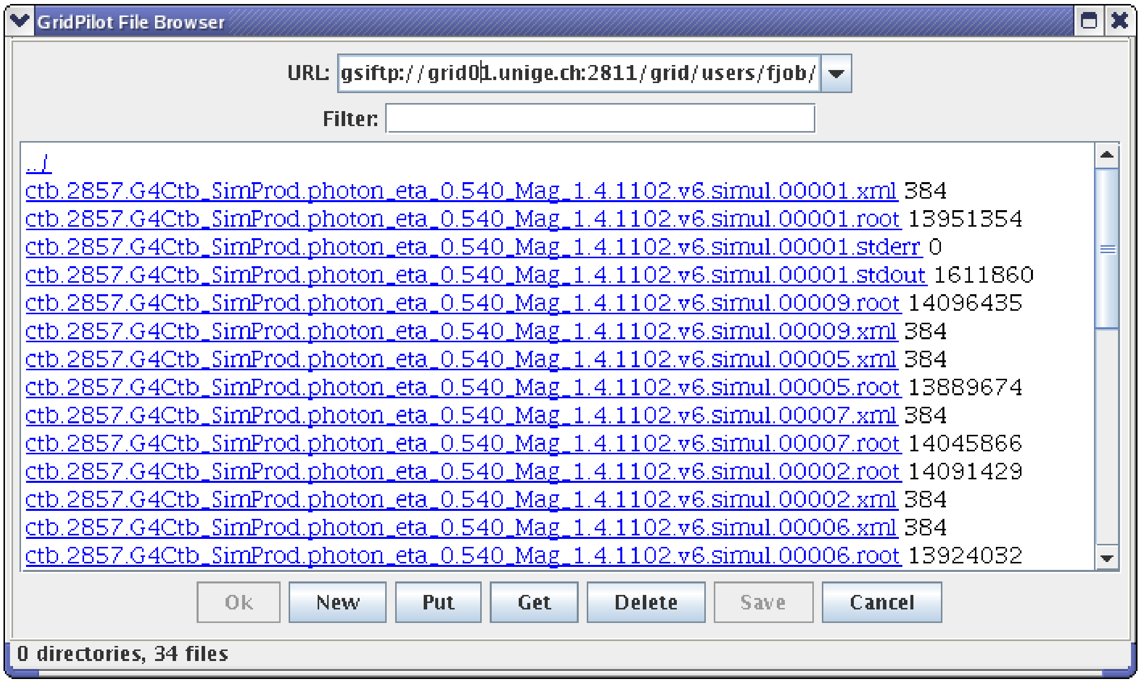

Choose directory
URL: file:/home/grid/orellana/tmp/files/
../
Register new locations in DB: My_DB_local
Ok
New
Put
Get
Delete
Save
Cancel

0 directories, 0 files

The directory chooser dialog presented when downloading/replicating files.

Files listed on a file table can be downloaded by clicking "Replicate file(s)" or from the right-click menu. The reason for using the word "replicate" instead of "download" is that the directory chooser that is popped up allows choosing a remote directory on a GridFTP server. If this is done, the file(s) will be copied using the third-party transfer capabilities of GridFTP. Another reason is that on the file chooser, one can also choose to have the new file locations registered in one of the available file catalogs.

The file browser



URL: gsiftp://grid01.unige.ch:2811/grid/users/fjob/

Filter:

```
../
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00001.xml 384
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00001.root 13951354
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00001.stderr 0
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00001.stdout 1611860
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00009.root 14096435
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00009.xml 384
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00005.xml 384
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00005.root 13889674
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00007.xml 384
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00007.root 14045866
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00002.root 14091429
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00002.xml 384
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00006.xml 384
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00006.root 13924032
```

0 directories, 34 files

The file browser window.

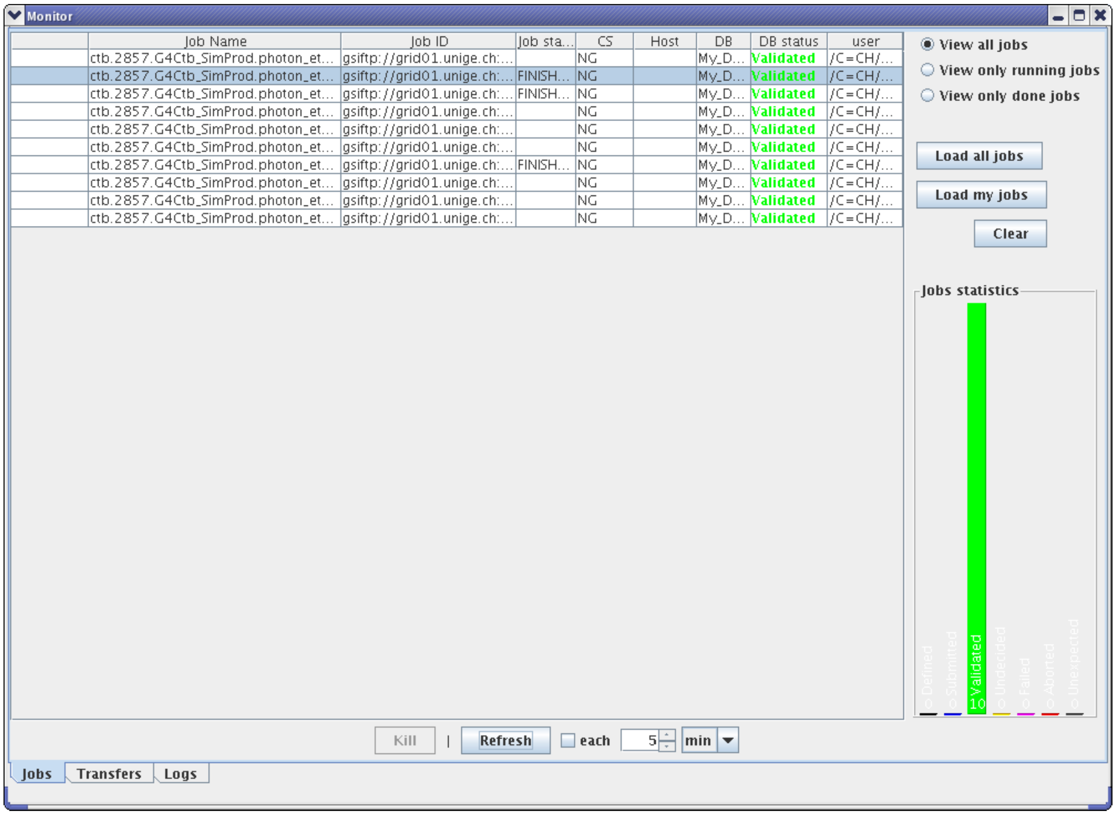
The GridPilot file browser is opened by selecting "View" → "New browser" or typing ctrl+o. It works much like a standard file/web browser (e.g. the Explorer on MS Windows or Konqueror on Linux) with (very) limited functionality and some quirks. The main difference is that apart from the local file system and the web, also GridFTP servers can be browsed. Moreover, files can be downloaded and uploaded and files and directories can be created. The download/upload functionality is meant only for quickly accessing single, small files. Larger files should be transferred via the functionality provided on the file catalog tabs.

Hint: After typing in or selecting a URL from the history drop-down menu, you must hit return. Otherwise the location will not be opened, and, when the file browser is used a file/directory chooser, this means that what you actually choose is not the location you see in the address field of the browser, but a default location.

The monitoring window

The monitoring window contains at least three tabs. 3 of these tabs hold: a job monitor, a file transfer monitor and a log file viewer. The monitoring window is shown when a job is submitted or a file transfer is started. It can be manually shown or hidden by checking or unchecking "View" → "Show monitor", or by typing ctrl+m. Depending on which computing system plugins are enabled (in the preferences), other tabs may be present in the monitoring window – displaying information on virtual machines.

The job monitor



	Job Name	Job ID	Job sta...	CS	Host	DB	DB status	user
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...		NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...	FINISH...	NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...	FINISH...	NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...		NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...		NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...		NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...	FINISH...	NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...		NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...		NG		My_D...	Validated	/C=CH/...
	ctb.2857.G4Ctb_SimProd.photon_et...	gsiftp://grid01.unige.ch:...		NG		My_D...	Validated	/C=CH/...

The job monitor tab in the monitor window.

If jobs were started with a previous launch of GridPilot, they can be retrieved by clicking "Load all jobs" or "Load my jobs". Alternatively, records can be selected in a "jobDefinitions" tab in the main window and chosen to be monitored with the button at the bottom or from the right-click menu.

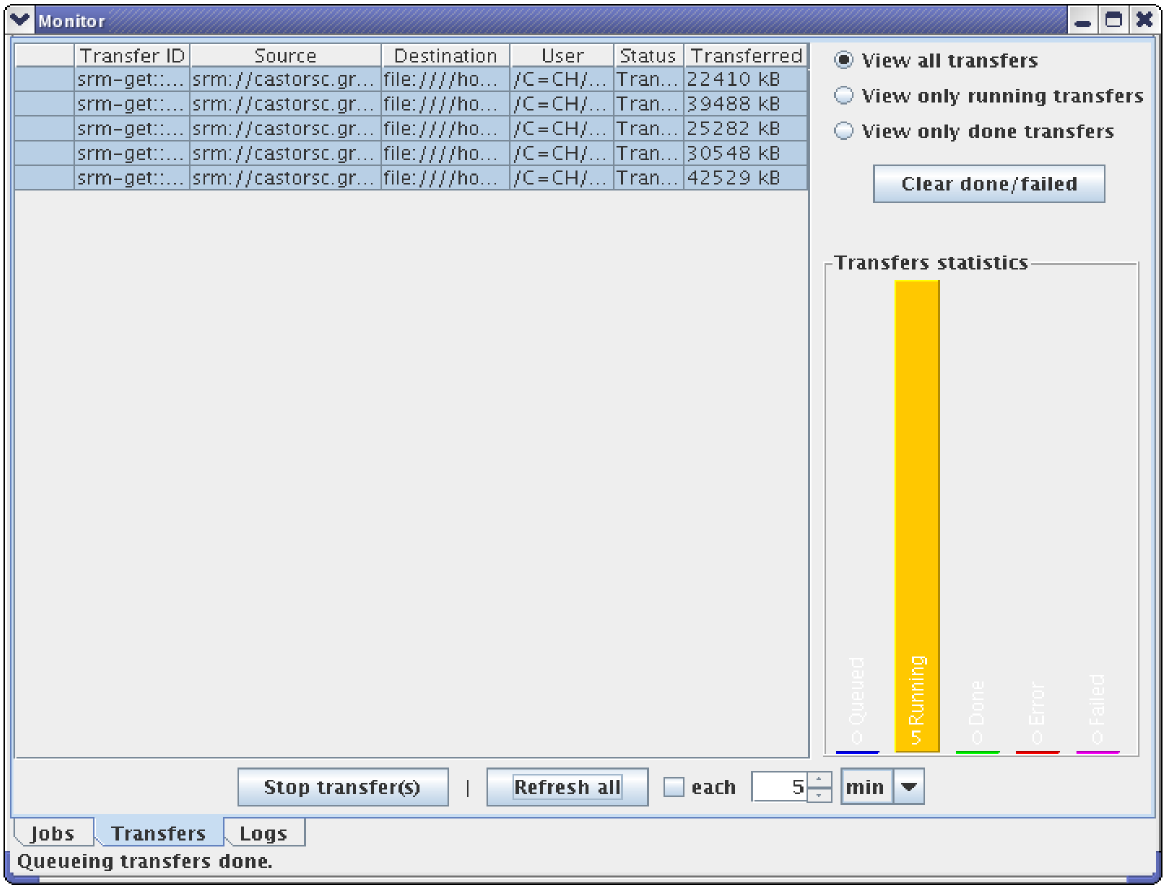
Clicking "Clear" will simply clear the monitor, but not affect neither the running of the jobs nor the database records of the jobs. To update the status of all monitored jobs by querying their computing systems, the button "Refresh" should be clicked. Checking the "each" checkbox will cause such an update to be performed automatically with regular intervals.

The statistics panel shows current number of jobs that are validated, failed, running, etc. Different views are obtained by clicking on the graphics.

Clicking on the column names causes the list of jobs to be sorted according to the values of that column.

Selecting one or several jobs and right-clicking will bring up a menu of actions that can be performed on these jobs. These actions include killing the jobs, changing the job status and resubmitting the jobs.

The file transfer monitor



	Transfer ID	Source	Destination	User	Status	Transferred
	srm-get:...	srm://castorsc.gr...	file:///ho...	/C=CH/...	Tran...	22410 kB
	srm-get:...	srm://castorsc.gr...	file:///ho...	/C=CH/...	Tran...	39488 kB
	srm-get:...	srm://castorsc.gr...	file:///ho...	/C=CH/...	Tran...	25282 kB
	srm-get:...	srm://castorsc.gr...	file:///ho...	/C=CH/...	Tran...	30548 kB
	srm-get:...	srm://castorsc.gr...	file:///ho...	/C=CH/...	Tran...	42529 kB

● View all transfers
○ View only running transfers
○ View only done transfers

□ each 5 min

Queueing transfers done.

The transfer monitor tab in the monitor window.

Initiating a file transfer from the "files" tab in the main window will cause information about this transfer to be displayed on the file transfer monitor. The functionality of the file transfer monitor is very similar to that of the job monitor, with the important difference that information about file transfers is lost when GridPilot is closed. GridPilot is thus not meant to be used for large-scale file replication.

Hint: When downloading from an SRM server, GridPilot may time out before the server returns "Ready". This is usually solved by simply retrying the transfer manually, or changing the values of one or both of the following configuration parameters: “`copy retries`” and “`copy retry timeout`”.

The log viewer

Error information and some other information is logged in the file gridpilot.log. Information added since GridPilot was launched is displayed on the log viewer tab. Right-clicking in this tab will bring up a menu with some choices on how to display newly added information.

Running jobs

For GridPilot, every job and every file belong to a dataset. So, before running jobs and producing files, a dataset record must be defined.

Moreover, since a dataset is produced by a transformation, defining a dataset involves choosing a transformation. If a suitable transformation is not found in the transformation table, a new one should be defined.

The following should be considered when planning a production:

- where to store job information: this is determined by the database chosen for dataset record
- where to get input files: this is determined by the input dataset chosen when defining the dataset record
- where to store output files: this is chosen when defining the dataset record
- where to register output files: they will be registered in the database holding the dataset record
- on which computing resources to run: this is chosen on submission time

Building a transformation

A test transformation is automatically created on startup. It can be opened by double-clicking on the record on a "transformations" tab.

It is seen that the runtime environment chosen is "Linux". Clicking on the drop-down box reveals a list of names. These have been filled in by the enabled computing system plugins. If the right environment is not available, two things can be done: 1) a new one can be defined. This involves setting up the corresponding software on one or several computing back-ends. For example, when running locally, it involves writing a setup script and placing it in the folder defined in the configuration file by e.g. `runtime directory = ~/GridPilot/runtimeEnvironments`. 2) "Linux" can be chosen and a tarball containing the necessary software can be specified in the "inputFiles" field of the transformation record.

It is also seen that "name", "version" and "comment" have been specified. These are simply strings labeling the transformation.

The field "arguments" specifies which arguments the transformation script must be given. It must be given as a space separated list of strings. Each string can be anything, but typically is a mnemonic name, labeling the particular parameter given to the script. If the string is not one of the 'special arguments' listed in the section "Generating job definitions" it will appear as a field to be filled in when defining the jobs. The test transformation takes two arguments: "multiplier" and "inputFileURLs". "inputFileURLs" is one of the 'special arguments' and will be filled in automatically, leaving only "multiplier" to be filled in (any integer can be given).

The field "script" specifies the physical location of the transformation script. It can be on the local disk, on the web or on a GridFTP server.

The field "inputFiles" specifies input files that will be downloaded and placed next to the transformation script for all jobs. As mentioned above, this could for example be a tarball containing software used by the transformation.

Notice that in order to save typing work, one can simply use the menu items "Edit" → "Copy" and "Edit" → "Paste" to copy a transformation and then edit only the fields that need to be changed.

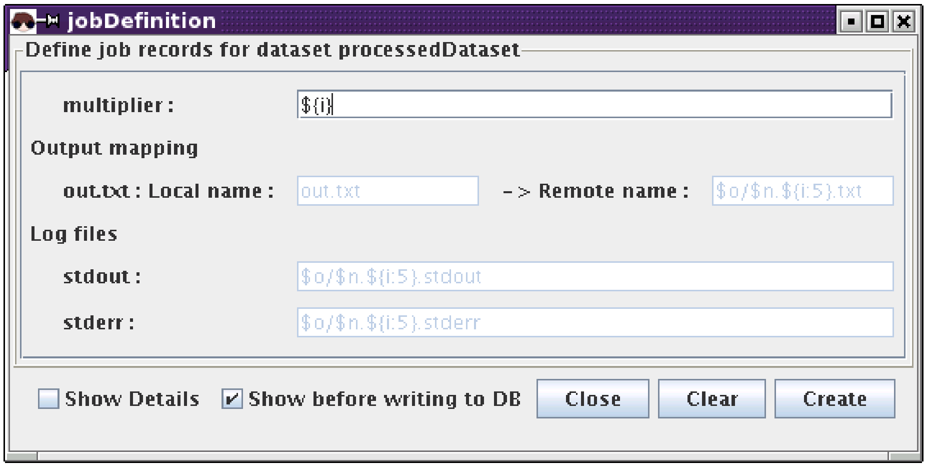
Building a dataset

The main consideration is whether or not the dataset to be defined has the files of another dataset as input files. If so, this other dataset should be found and selected on a "datasets" tab. Then, with this dataset record selected, the button "Define new record(s)" should be clicked. After clicking this button a window will pop up with fields to be filled in.

If an input dataset was selected, a drop-down box is presented, where the target database is chosen. In other words, it is allowed to define a dataset in one database with an input dataset from another. Once a choice has been made, most of the fields will be filled in automatically: fields that occur in both the source and target dataset records will be filled in with the values from the source. It is then up to one self to edit to one's liking. If several input datasets are selected, several new datasets will be defined. In this case, the fields are filled in with values from the first source dataset. Fields left empty will be filled with values from consecutive datasets.

The fields of the default schema that *must* be filled in are "name" and "outputLocation". If the data files to be produced are high energy physics data, they are likely to contain a number of so-called *events*. If the total number of events to be produced as output of the production is known, it can be specified in the field "totalEvents". The field "totalFiles" specifies the number of files to be produced. Optionally, one can fill in "metaData" with some information characterizing the data files in question.

Generating job definitions



jobDefinition

Define job records for dataset processedDataset

multiplier : ${i}

Output mapping

out.txt : Local name : out.txt -> Remote name : $o/$n.${i:5}.txt

Log files

stdout : $o/$n.${i:5}.stdout

stderr : $o/$n.${i:5}.stderr

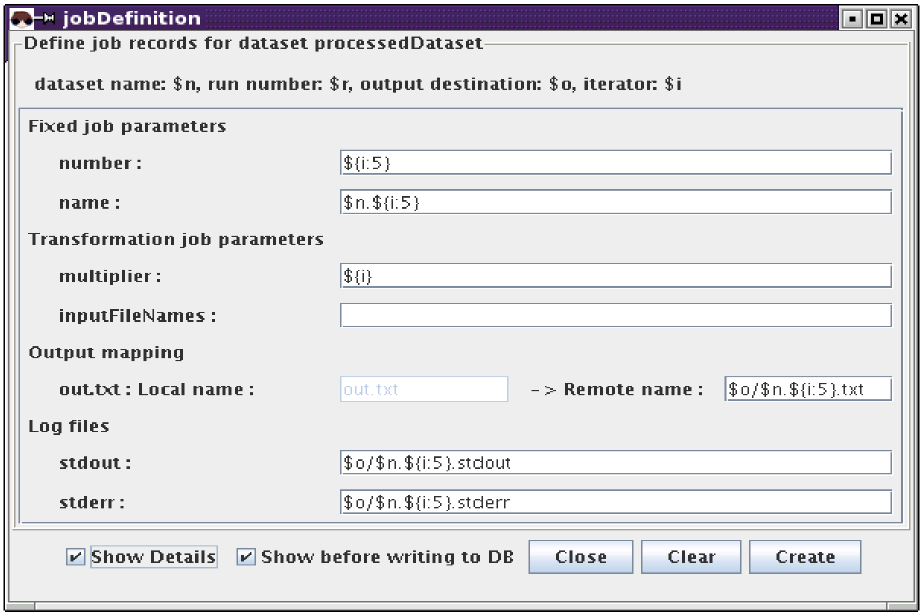
☐ Show Details ☑ Show before writing to DB

The job creation window.

Once a dataset record has been created, job definition records can be generated in an automatic way: on the "datasets" tab, select the created dataset record and click the button "Create job definitions". This will pop up a window that may or may not have fields to be filled in. If there are fields to be filled in, they will be unknown arguments of the transformation script. They may also be fields to be filled in if another schema than the default is used for the job definition table.

Usually it should not be necessary to fill in anything, or but a few static variables to be passed as arguments to the transformation script. For example, the output file(s) of the transformation are assigned a generated name to be used when uploaded to the final destination.

Clicking "Create" will pop up a confirmation window. Clicking "OK for all" will start the generation of the job definition records. This may take a few seconds, please leave the job definition window open to get the feedback that the generation of records has finished.



Define job records for dataset processedDataset

dataset name: $n, run number: $r, output destination: $o, iterator: $i

Fixed job parameters

number : ${i:5}

name : $n.${i:5}

Transformation job parameters

multiplier : ${i}

inputFileNames :

Output mapping

out.txt : Local name : out.txt -> Remote name : $o/$n.${i:5}.txt

Log files

stdout : $o/$n.${i:5}.stdout

stderr : $o/$n.${i:5}.stderr

☑ Show Details ☑ Show before writing to DB

The job creation window with "Show details" checked.

There may be cases where even more control is needed. For such cases, a number of variables are available. Some of these are listed when the check-box "Show details" is checked. Here follows a list of all available variables:

$n	the value of the field "name" of the dataset record
$r	the value of the field "runNumber" of the dataset record
$o	the value of the field "outputLocation" of the dataset record
${i}	the number of the generated job definition (1,2,3,...)
${i:n}	the number of the generated job definition (1,2,3,...), padded with zeros to take up n digits
$1, $2, $3, ...	the value of the dataset field number 1, 2, 3, ...

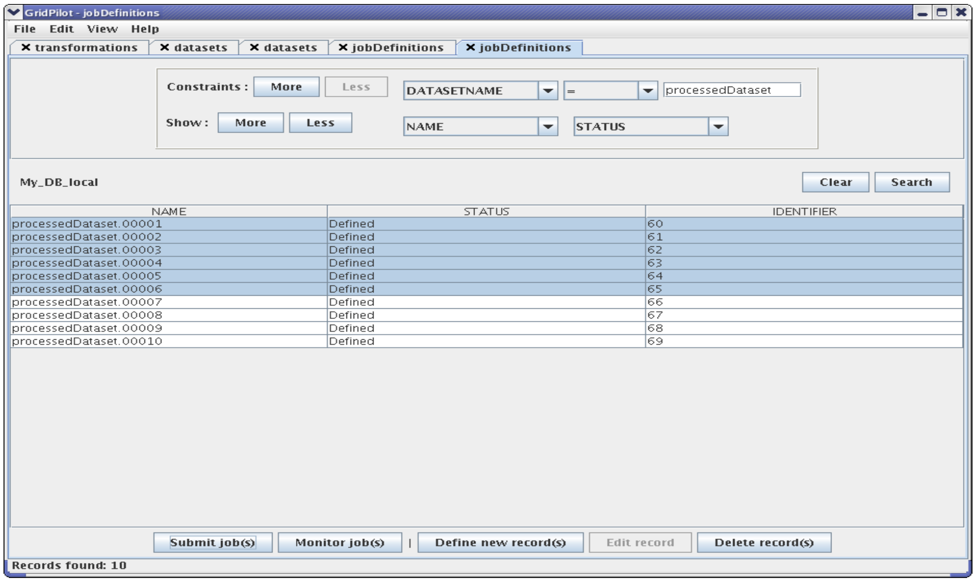
Moreover, standard arithmetics can be used within curly braces. E.g. if ${i} is 4, then ${i:4} is 0004 and ${(i-1)%20 + 1} is equal to 5.

In general, when a transformation has some argument names specified in the field "arguments", GridPilot will prompt for the value of these when defining jobs. They will then be filled in the "jobDefinition" field "transPars". However, as mentioned, certain names are known and will be filled in automatically. These name are:

inputFileURLs	this argument will be assigned the value of the field of the same name in the job definition record, which in turn is filled in automatically if an input dataset has been chosen
nEvents	this argument will be assigned the value of the field of the same name in the job definition record, which in turn is filled in automatically if an input dataset has been chosen or the fields "totalFiles" and "totalEvents" have been filled in in the dataset record
eventMin	-"-
eventMax	-"-
inputFileNames	if left empty and an input dataset has been selected, this argument will be assigned the value of the field of the same name in the job definition record, which in turn is filled in automatically on job creation

Finally, if a transformation argument of name `someName` is matched by a line in the dataset field **"metaData"** of the form `someName: some value`, `some value` is filled in as value for the argument `someName`.

Submitting jobs



Constraints : DATASETNAME = processedDataset

Show : NAME STATUS

My_DB_local

NAME	STATUS	IDENTIFIER
processedDataset.00001	Defined	60
processedDataset.00002	Defined	61
processedDataset.00003	Defined	62
processedDataset.00004	Defined	63
processedDataset.00005	Defined	64
processedDataset.00006	Defined	65
processedDataset.00007	Defined	66
processedDataset.00008	Defined	67
processedDataset.00009	Defined	68
processedDataset.00010	Defined	69

Records found: 10

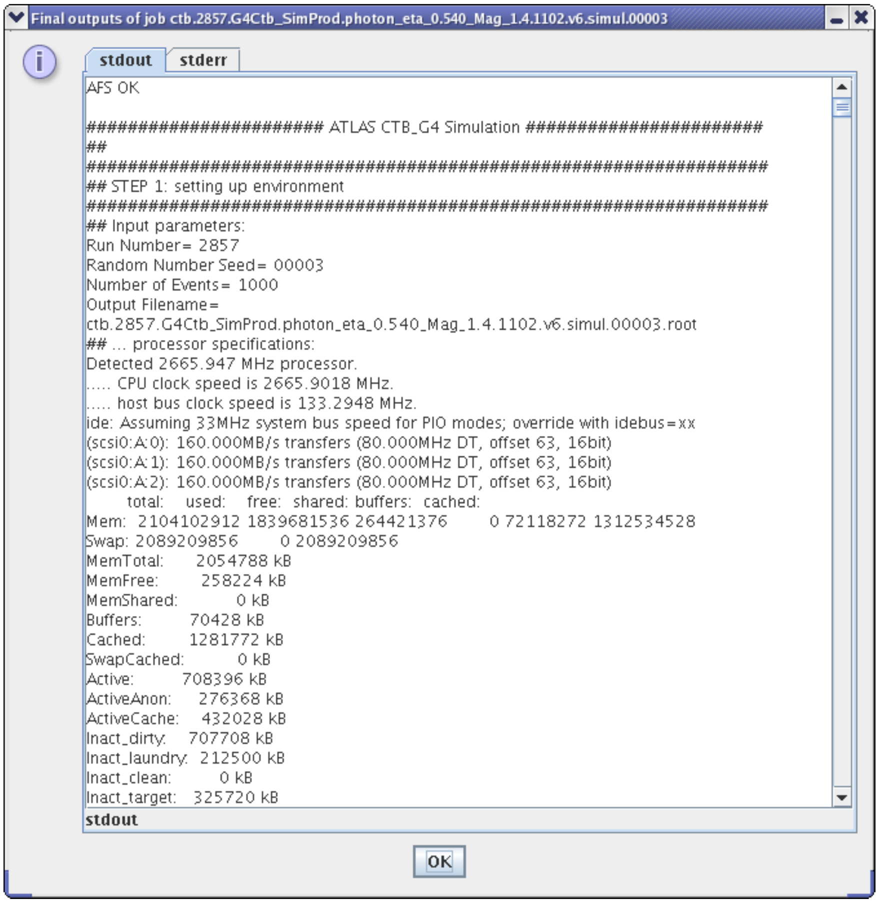
A job “definitions” tab in the main window.

Job submission is done by selecting jobs on a "jobDefintitions" tab clicking the button "Submit job(s) or using the right-click menu. The computing systems appearing are the ones defined in the configuration file.

Once submitted, the jobs will appear on the job monitoring tab in the "Monitor" window. Notice that this does not mean that they have actually been accepted on the computing system. Actual acceptance is signaled by a Job ID appearing in the first column.

After a job has been assigned a Job ID, one can click "Refresh" or use the right-click menu to get various information on selected job(s) as described in the section "The monitoring window".

Monitoring jobs and retrieving output files


Final outputs of job ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00003
stdout
stderr
stdout
OK

```
AFS OK

######################## ATLAS CTB_G4 Simulation ########################
##
########################################################################
## STEP 1: setting up environment
########################################################################
## Input parameters:
Run Number= 2857
Random Number Seed= 00003
Number of Events= 1000
Output Filename=
ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul.00003.root
## ... processor specifications:
Detected 2665.947 MHz processor.
..... CPU clock speed is 2665.9018 MHz.
..... host bus clock speed is 133.2948 MHz.
ide: Assuming 33MHz system bus speed for PIO modes; override with idebus=xx
(scsi0:A:0): 160.000MB/s transfers (80.000MHz DT, offset 63, 16bit)
(scsi0:A:1): 160.000MB/s transfers (80.000MHz DT, offset 63, 16bit)
(scsi0:A:2): 160.000MB/s transfers (80.000MHz DT, offset 63, 16bit)
        total:    used:    free:  shared: buffers:  cached:
Mem:  2104102912 1839681536 264421376        0 72118272 1312534528
Swap: 2089209856        0 2089209856
MemTotal:      2054788 kB
MemFree:        258224 kB
MemShared:           0 kB
Buffers:         70428 kB
Cached:        1281772 kB
SwapCached:          0 kB
Active:         708396 kB
ActiveAnon:     276368 kB
ActiveCache:    432028 kB
Inact_dirty:    707708 kB
Inact_laundry:  212500 kB
Inact_clean:         0 kB
Inact_target:   325720 kB
```

The stdout of a running job.

The status of jobs, can be tracked by regularly clicking the button "Refresh" and by using the items of the right-click menu to e.g. inspect the stdout. When GridPilot performs such a refresh and detects that a job has finished, certain actions will be performed:

- the stdout and stderr of the job will be downloaded to the local disk and checked for error messages. This is referred to as *validation* of the job
- if the job passes validation and if not already done by the computing system, output files (including stdout and stderr) will be copied to their final destination, as specified in the

dataset record

- if the first two points have gone well, the job will be set as "Validated". If not, it will be set as "Failed" or "Undecided"
- if the first two points have gone well and if the database holding the job information is a file catalog, the output file of the job will be registered in the file catalog. If the database is not a file catalog, the output file will always figure on the "files" tab, irrespective of whether it actually exists or not
- if the stdout contains lines of the form `GRIDPILOT METADATA: [someField] = [someValue]` and the field `[someField]` is one of the fields of the "jobDefinitions" table, the value `[someValue]` will be filled in in the jobDefinition record

Hint: a job may write something on stderr without actually failing. However, a non-empty stderr will cause the job to be flagged as "Undecided". To avoid this, you can have your job script redirect stderr to stdout. E.g. for Bash, this is done with `2>&1`.

If jobs have ended up in the state "Undecided", they can be set as "Validated" by hand from the right-click menu. This can also be done in an interactive way, by selecting "Decide" from the right-click menu.

In order to rerun jobs, they should first be set as "Failed", then as "Defined". This will trigger a cleaning up of possible output files. If such a clean-up is not desired (it may take some time, even if there were no output files produced), jobs can be set as "Aborted" and then "Defined". A shorter way to resubmit is to simply select "Resubmit" from the right-click menu. This will also trigger a clean-up of output files.

Failed jobs can be resubmitted from the right-click menu. This may make sense, if the failure was due to some temporary problem with the computing (grid) system.

Running jobs can be killed via the "Kill" button or from the right-click menu.

Examples

Notice: The jobs of these examples can run on any of the supported grid systems (currently, they are all Linux based), however, if you're on a Linux system you can also run these jobs *locally* with the “Fork” plugin. If you're on a different platform, e.g. MS Windows, you can run the jobs in a local virtual machine with the “VMFork” plugin.

Running a simple job

As a first simple example we will run a job that uses the uses the transformation “no_files_transformation”.

- on the “transformations” tab, click “Search” [1]
- double-click on the row with name “my_transformation” - you'll see that the corresponding script, “my_transformation .sh”, is an empty file
- write any Linux commands you like, e.g. ”`echo hello world`“ in this file. After doing this, save and close the script and close also the transformation
- on the “datasets” tab, click “Search”
- select “my_dataset” and click “Create jobDefinition(s)”, “Create”, “OK” and “Close”
- select “my_dataset” and click “Show jobDefinitions(s)”
- select the job you just created and click “Submit job(s)” → [your favorite system]
- the job monitoring window will open and you can follow the progress of your job
- after the job has finished, the first time you click on “Refresh” on the job monitoring panel, the stdout and stderr of the job will be copied to the “outputLocation” of the dataset - by default your “grid home URL” (as set in your preferences)

Running 10 simple jobs

As a second simple example we will run a job that uses the uses the transformation “test”. If you open the transformation record, you'll see that the corresponding script, “test.sh”, is a shell script calling standard GNU/Linux tools. “test.sh” takes a number, “multiplier”, as argument[2] and produces another number which it saves to a file, “out.txt”.

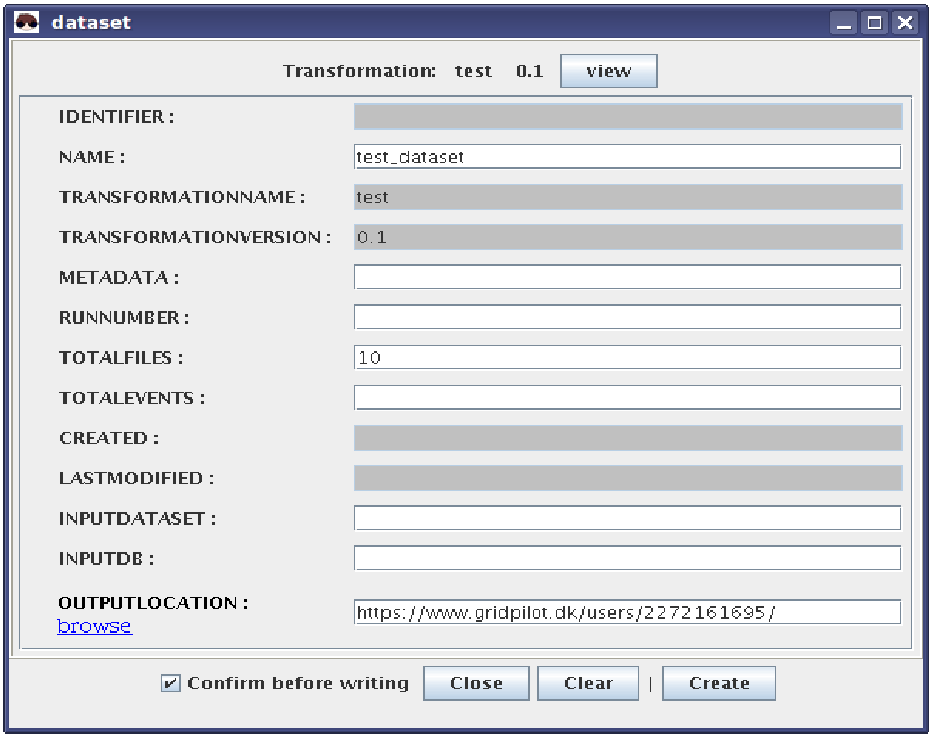
Here is how to run the script with numbers from 1 to 10 and save the 10 output files on a remote server as “test_dataset.01.txt”, “test_dataset .02.txt”, ..., “test_dataset .10.txt”:

- click on the "datasets" tab
- click on “Define new record(s)”
- fill in at lease the fields “NAME”, “TOTALFILES” and “OUTPUTLOCATION”. An example is given in the figure below
- click “Create” and “Close”

[1] If you've closed this tab, open a new on: "View" → "New tab with My_DB_Local"→ "transformations".

[2] The second argument, “inputFileNames”, is optional (it defaults to “data1.txt,data2.txt”) and we will not use it in this example.

- on the “datasets” tab: select the created dataset and click “Create jobDefinition(s)”
- fill in the field “multiplier” e.g. with the value ${i}
- select the dataset you just created and click “Show jobDefinitions(s)”
- select the jobs you just created and click “Submit job(s)” → [your favorite system]



Transformation: test 0.1

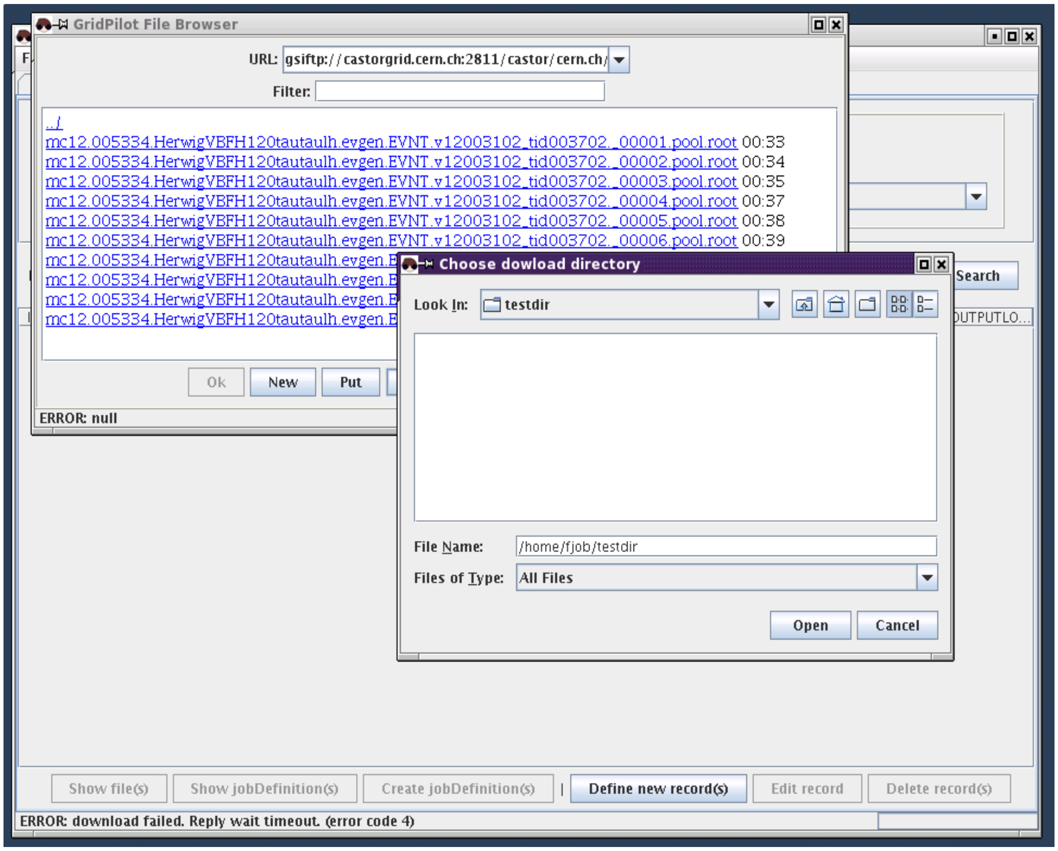
IDENTIFIER :	
NAME :	test_dataset
TRANSFORMATIONNAME :	test
TRANSFORMATIONVERSION :	0.1
METADATA :	
RUNNUMBER :	
TOTALFILES :	10
TOTALEVENTS :	
CREATED :	
LASTMODIFIED :	
INPUTDATASET :	
INPUTDB :	
OUTPUTLOCATION :	https://www.gridpilot.dk/users/2272161695/

☑ Confirm before writing

Finding and downloading a file from a GridFTP server

If you just want to find and download a few files quickly, you may start by looking for your favorite dataset name with Google. If you're lucky and find some URL that start with "gsiftp://", type ctrl+o in GridPilot, copy-paste a *directory* URL in the URL field and hit return. After waiting a few seconds for the SSL handshake, etc. you should see a list of the files in the directory.

Then, find the file you're interested in and click "Get". This will present you with a local directory chooser. After choosing a local directory, you will be prompted for a file name. Here you have to type in the name of the file (you could have copied the name from the browser window).



URL: gsiftp://castorgrid.cern.ch:2811/castor/cern.ch/

Filter:

```
../
mc12.005334.HerwigVBFH120tautaulh.evgen.EVNT.v12003102_tid003702._00001.pool.root 00:33
mc12.005334.HerwigVBFH120tautaulh.evgen.EVNT.v12003102_tid003702._00002.pool.root 00:34
mc12.005334.HerwigVBFH120tautaulh.evgen.EVNT.v12003102_tid003702._00003.pool.root 00:35
mc12.005334.HerwigVBFH120tautaulh.evgen.EVNT.v12003102_tid003702._00004.pool.root 00:37
mc12.005334.HerwigVBFH120tautaulh.evgen.EVNT.v12003102_tid003702._00005.pool.root 00:38
mc12.005334.HerwigVBFH120tautaulh.evgen.EVNT.v12003102_tid003702._00006.pool.root 00:39
mc12.005334.HerwigVBFH120tautaulh.evgen.E
mc12.005334.HerwigVBFH120tautaulh.evgen.E
mc12.005334.HerwigVBFH120tautaulh.evgen.E
mc12.005334.HerwigVBFH120tautaulh.evgen.E
```

ERROR: null

Look In: testdir

File Name: /home/fjob/testdir

Files of Type: All Files

ERROR: download failed. Reply wait timeout. (error code 4)

Browsing files on gsiftp://castorgrid.cern.ch/.

After this, the download should start. GridPilot will hang until the file has been downloaded. If it's a small file, you will not notice much. If it's a large file, it's a nuissance. *Therefore, you should not use the file browser for downloading anything but a few small files.*

Instead, you should locate files in a dataset/file catalog and initiate the transfer from the "files" tab on the main window.

However, some files on some GridFTP server may not be registered anywhere. Well, then you can register them!

Registering files on a GridFTP server in a file catalog

Let's take the example of a private production of ATLAS data that has been carried out and the files put on a GridFTP server.

These files are grouped in datasets that are registered in a dataset catalog on one of the database back-ends. The files are then registered in a file catalog on the same back-end.

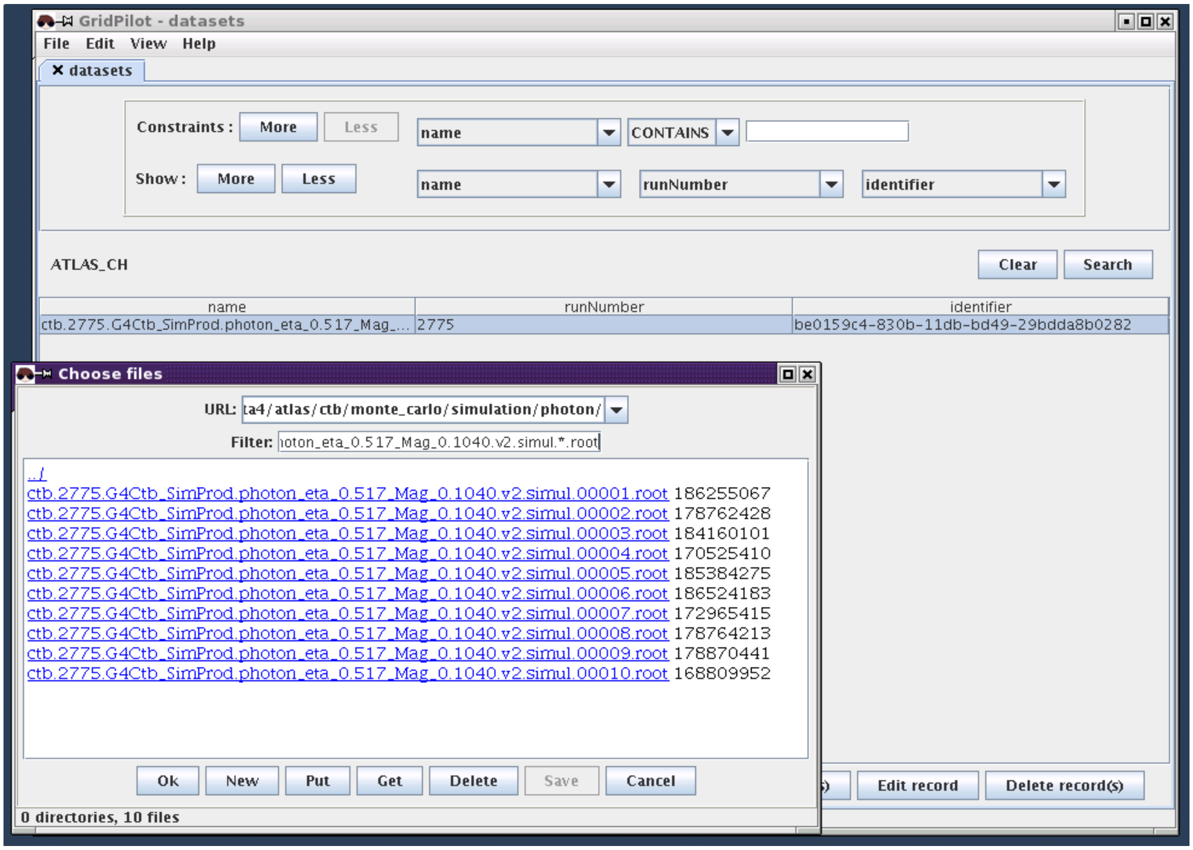
If the target audience is limited, both registrations can be done on any MySQL server. The interested parties can then all have this server listed in their GridPilot configuration file.

If the target audience is the whole ATLAS collaboration, the dataset catalog should be a central

DQ2 catalog and the file server one registered in the file "TiersOfATLAS".

Hint: Instead of immediately publishing datasets and files in a central catalog like the central ATLAS DQ2 dataset/file catalog, you can first publish them either in the default local database provided by GridPilot or in your own MySQL database. Then you can always re-publish them in a higher-level database later, by simple copy-paste between GridPilot tabs.

Here, we will go through how an ATLAS dataset and 10 associated files were registered in a MySQL catalog.



Constraints: name CONTAINS

Show: name, runNumber, identifier

ATLAS_CH

name	runNumber	identifier
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_...	2775	be0159c4-830b-11db-bd49-29bdda8b0282

URL: ta4/atlas/ctb/monte_carlo/simulation/photon/

Filter: hoton_eta_0.517_Mag_0.1040.v2.simul.*.root

```
../
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00001.root 186255067
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00002.root 178762428
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00003.root 184160101
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00004.root 170525410
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00005.root 185384275
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00006.root 186524183
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00007.root 172965415
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00008.root 178764213
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00009.root 178870441
ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00010.root 168809952
```

0 directories, 10 files

Registering files located on a GridFTP server in a MySQL dataset and file catalog.

- a "datasets" tab was opened with the database system to be used for registration. Then "Define new record(s)" was clicked and filled in as best possible. As a minimum, the "name" field must be filled in. The identifier field is filled in by GridPilot with a freshly generated UUID.
- the newly created dataset record was selected and from the right-click menu "Import file(s)" was chosen

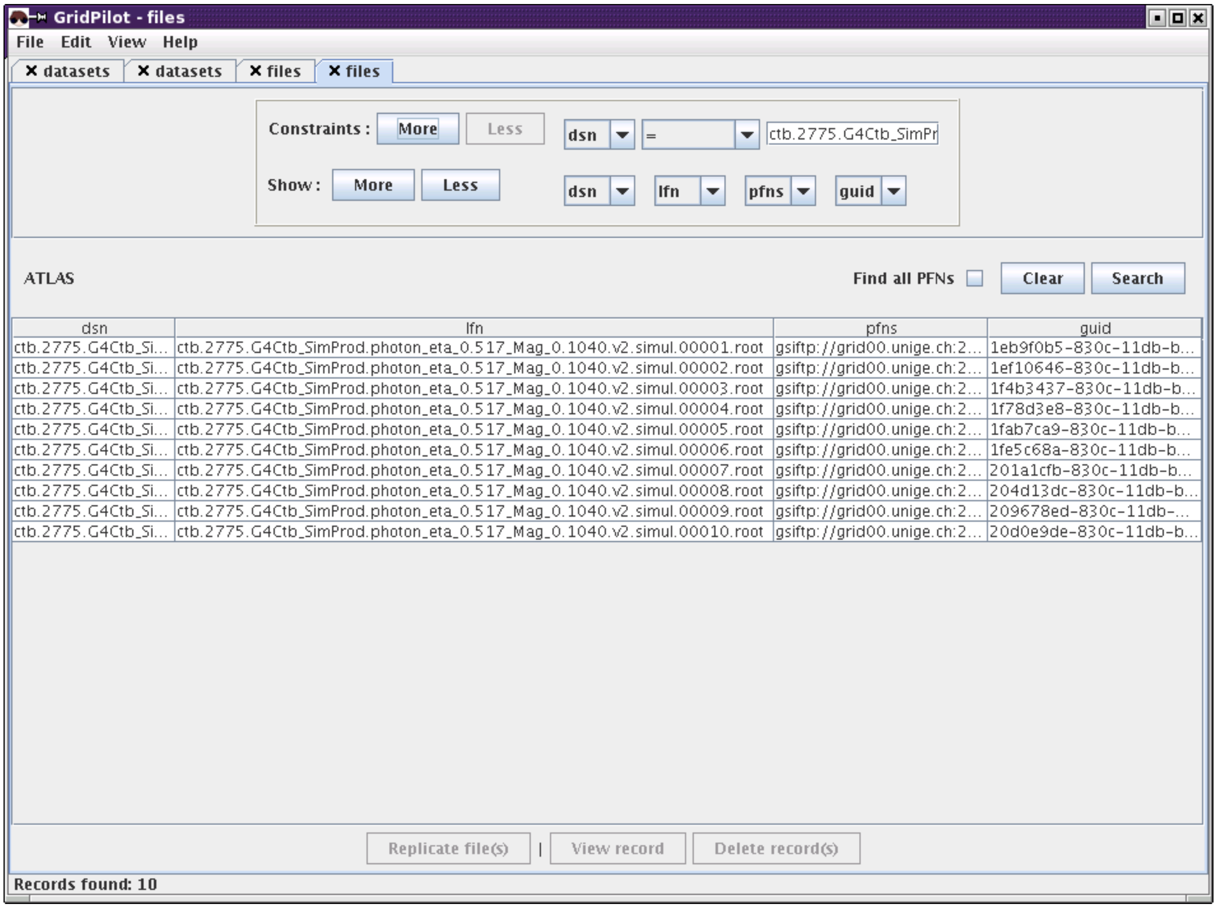
- with the file selector that popped up the physical files were selected. GridPilot registers all files present in the selector window, so one should, if necessary, limit the selection by filling in the "Filter" text field and hitting return. In the case at hand the string `ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.*.root` was used as filter. UUIDs were generated by GridPilot for all files
- Once the wanted files and none else were in the browser window, "OK" was clicked
- to verify that the files had been registered the dataset was selected and "Show files" was clicked

Publishing files registered in one file catalog in another

Let's see how the files just generated could be made available to the whole ATLAS collaboration.

In GridPilot logic, the files you want to register, must belong to a dataset, also in the new file catalog. So, first one has to create the dataset in the "ATLAS" database system and then register the files with this dataset:

- **Open a "datasets" tab with the "ATLAS" database system:** "View" → "ATLAS" → "datasets"
- **Verify that the name is not already taken:** type `ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul` in the search field
- **Copy the dataset record to the "ATLAS" database system:** the row on the "datasets" tab with the MySQL datase system can simply be copy-pasted into the "datasets" tab with the "ATLAS" database system
- **Open a "files" tab with the ATLAS" database system:** select the newly created dataset and click "Show file(s)"
- **Register the files with the "ATLAS" database system:** the rows on the "files" tab with the MySQL datase system can simply be copy-pasted into the "files" tab with the "ATLAS" database system



Constraints : dsn = ctb.2775.G4Ctb_SimPr

Show : dsn lfn pfns guid

ATLAS

Find all PFNs □

dsn	lfn	pfns	guid
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00001.root	gsiftp://grid00.unige.ch:2...	1eb9f0b5-830c-11db-b...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00002.root	gsiftp://grid00.unige.ch:2...	1ef10646-830c-11db-b...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00003.root	gsiftp://grid00.unige.ch:2...	1f4b3437-830c-11db-b...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00004.root	gsiftp://grid00.unige.ch:2...	1f78d3e8-830c-11db-b...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00005.root	gsiftp://grid00.unige.ch:2...	1fab7ca9-830c-11db-b...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00006.root	gsiftp://grid00.unige.ch:2...	1fe5c68a-830c-11db-b...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00007.root	gsiftp://grid00.unige.ch:2...	201a1cfb-830c-11db-b...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00008.root	gsiftp://grid00.unige.ch:2...	204d13dc-830c-11db-b...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00009.root	gsiftp://grid00.unige.ch:2...	209678ed-830c-11db-...
ctb.2775.G4Ctb_Si...	ctb.2775.G4Ctb_SimProd.photon_eta_0.517_Mag_0.1040.v2.simul.00010.root	gsiftp://grid00.unige.ch:2...	20d0e9de-830c-11db-b...

Records found: 10

Re-publishing file information from a MySQL dataset/file catalog to the central ATLAS catalog.

Notice: the "ATLAS" database system has a few quirks as compared to the local and MySQL database systems:

- When copy-pasting a dataset into a "datasets" tab with the "ATLAS" database system, the dataset identifier is not kept. The ATLAS system operates with both a DUID and a VUID; GridPilot identifies the dataset identifier of the other database systems with the VUID of ATLAS. Unfortunately, when creating a new ATLAS dataset, it is not possible to force a VUID - a new one is always generated
- When files are added to an ATLAS dataset, the dataset keeps its VUID as you would expect. However, when *deleting* files from a dataset, a new dataset (or dataset version in ATLAS terminology) is generated, with a new VUID

Replicating ATLAS data from an SRM server to a GridFTP server

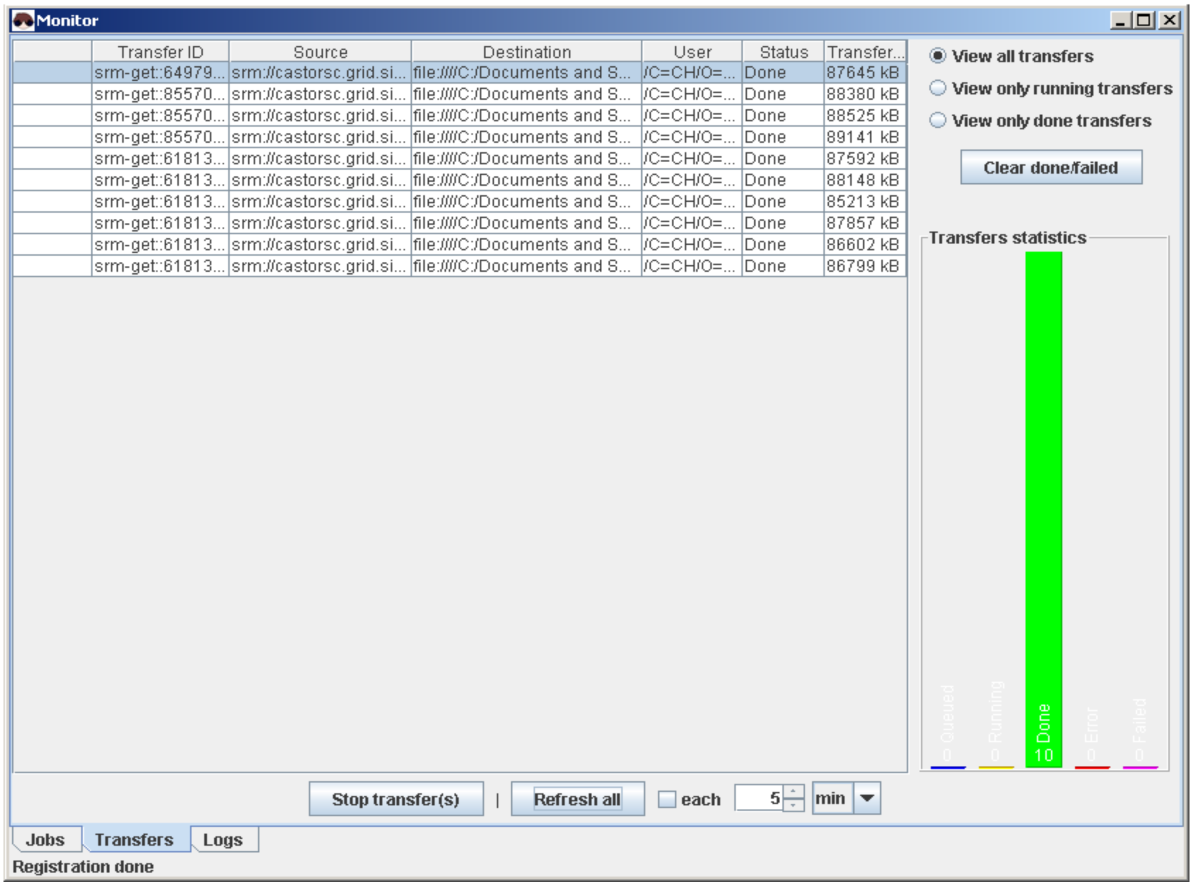
Task

- Locate the files of the dataset "csc11.005144.PythiaZee.recon.AOD.v11004103" in the central ATLAS dataset/file catalog

- Replicate them to a local GridFTP server - this includes registering the new file locations in a local MySQL dataset/file catalog

Notice

- GridPilot is **not** meant to be used for massive data movement. For this, the replication systems of the WLCG and/or ATLAS should be used. One reason is that GridPilot does not place the transfer requests with any external service and thus needs to run until the last transfer is completed.



	Transfer ID	Source	Destination	User	Status	Transfer...
	srm-get::64979...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	87645 kB
	srm-get::85570...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	88380 kB
	srm-get::85570...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	88525 kB
	srm-get::85570...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	89141 kB
	srm-get::61813...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	87592 kB
	srm-get::61813...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	88148 kB
	srm-get::61813...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	85213 kB
	srm-get::61813...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	87857 kB
	srm-get::61813...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	86602 kB
	srm-get::61813...	srm://castorsc.grid.si...	file:////C:/Documents and S...	/C=CH/O=...	Done	86799 kB

● View all transfers
○ View only running transfers
○ View only done transfers

□ each 5 min

Registration done

Replicating ATLAS CSC files to a GridFTP server and MySQL dataset/file catalog.

Steps

- Open a "datasets" tab with the "ATLAS" database system: "View" → "ATLAS" → "datasets"
- Type in `csc11.005144.PythiaZee.recon.AOD.v11004103` in the search field and hit return
- Select the dataset and click "Show files". GridPilot then first finds all the (unqualified) file

names and then starts querying the DQ2 server and the grid file catalogs for the locations of the files

- After a while, all files will appear in the results table. GridPilot may report "No response from ATLAS for select. Do you want to interrupt it?". In this case, simply click "No" (timeouts are configurable)
- Select some files and click "Replicate file(s)"
- With the file chooser, choose a download destination on a GridFTP server where you have write access, and from the drop-down list "Register new location in DB", choose to register the new locations in a MySQL database where you have write access. If you don't have write access on any MySQL database, you can always choose to register in "My_DB_local"
- After the transfers have started, click "Refresh all" to follow the progress of the transfers
- Wait for the transfers to finish

Hint: When clicking "Show files" on the ATLAS "datasets" tab, the "files" tab can take a long time in loading, because GridPilot queries the file catalog(s) for each single file to get the physical file name (URL). To interrupt this and show the remaining records without physical file names, click on the small cross next to the progress bar in the lower right corner of the main window.

Hint

- If the transfers fail to start transferring on a first try, it is probably because GridPilot times out waiting for a "Ready" message from the SRM server files. You can simply select them in the file transfer monitor and choose "Retry transfer(s)" from the right-click menu. Notice that timeouts are configurable
- If the transfers still fail, you can try downloading from more sources by repeating the search in the "Files" tab with the checkbox "Find all PFNs" checked and then retrying the replication

ATLAS Combined Test Beam photon simulation

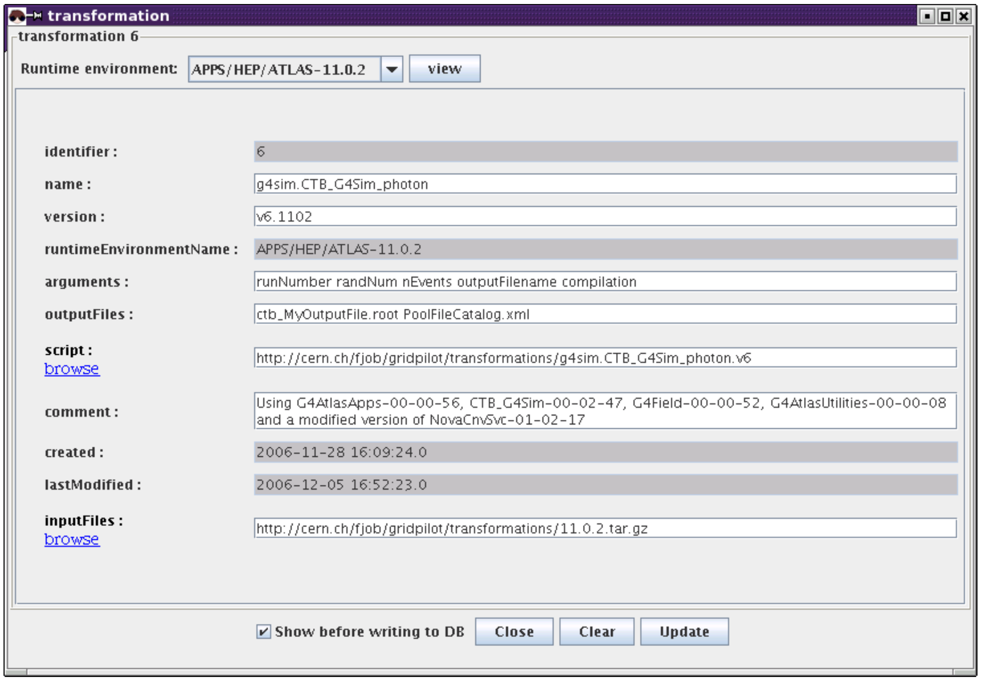
In this example we will re-simulate some ATLAS photon events from the Combined Test Beam exercise in 2004.

A suitable transformation script can be accessed at http://cern.ch/fjob/gridpilot/transformations/g4sim.CTB_G4Sim_photon.v6. To put this script to use, we have to create a record for it in GridPilot:

- open a "transformations" tab with any database system (where you have write access), e.g. "My_DB_local": "View" → "My_DB_local" → "transformations"

- click "Define new record(s)
- in the pop-up window, choose the runtime environment "ATLAS-11.0.2" from the drop-down list
- fill in the table in the pop-up window like in the figure below

Comment: If the runtime environment "ATLAS-11.0.2" is not available, try reconfiguring your NorduGrid/ARC computing system: e.g. to access Swiss resources you need to set `GIISes = ldap://odin.switch.ch:2135/Mds-Vo-name=Switzerland,o=grid`. To access only e.g. two computing resources you need to set `clusters = first.host.org second.host.org`



transformation 6

Runtime environment: APPS/HEP/ATLAS-11.0.2

identifier : 6

name : g4sim.CTB_G4Sim_photon

version : v6.1102

runtimeEnvironmentName : APPS/HEP/ATLAS-11.0.2

arguments : runNumber randNum nEvents outputFilename compilation

outputFiles : ctb_MyOutputFile.root PoolFileCatalog.xml

script : http://cern.ch/fjob/gridpilot/transformations/g4sim.CTB_G4Sim_photon.v6

comment : Using G4AtlasApps-00-00-56, CTB_G4Sim-00-02-47, G4Field-00-00-52, G4AtlasUtilities-00-00-08 and a modified version of NovaCnvSvc-01-02-17

created : 2006-11-28 16:09:24.0

lastModified : 2006-12-05 16:52:23.0

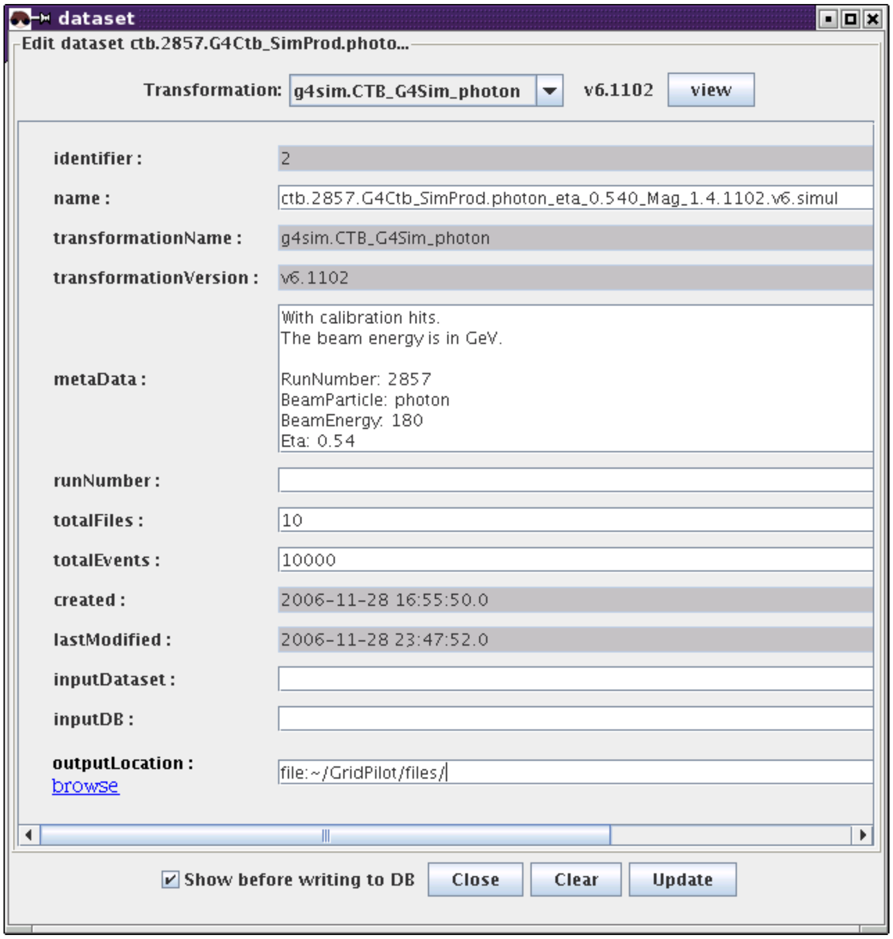
inputFiles : http://cern.ch/fjob/gridpilot/transformations/11.0.2.tar.gz

☑ Show before writing to DB

Creating an ATLAS transformation for photon simulation.

Next, we define a dataset:

- open a "datasets" tab with the same database system
- click "Define new record(s)
- in the drop-down in the pop-up window, **choose the transformation you just created**
- fill in the table in the pop-up window like in the figure below



Edit dataset ctb.2857.G4Ctb_SimProd.photo...

Transformation: g4sim.CTB_G4Sim_photon v6.1102

identifier : 2

name : ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul

transformationName : g4sim.CTB_G4Sim_photon

transformationVersion : v6.1102

metaData :

```
With calibration hits.
The beam energy is in GeV.

RunNumber: 2857
BeamParticle: photon
BeamEnergy: 180
Eta: 0.54
```

runNumber :

totalFiles : 10

totalEvents : 10000

created : 2006-11-28 16:55:50.0

lastModified : 2006-11-28 23:47:52.0

inputDataset :

inputDB :

outputLocation : file:~/GridPilot/files/

browse

☑ Show before writing to DB

Creating an ATLAS photon simulation dataset.

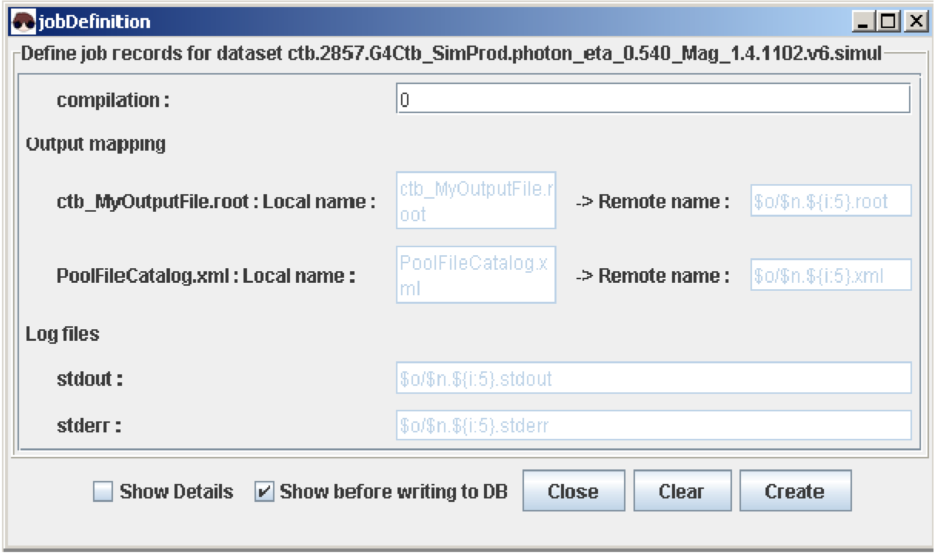
Comments

- as "name" you can choose anything you like, but bear in mind that ATLAS has a naming convention that you may want to follow
- the "metaData" should contain some description of the data this dataset will contain. Lines of the form `[field]: [value]` are special: the values can be accessed when creating jobs (see below)
- the "outputLocation" can be either a directory on a GSIFTP server or a directory on your local hard disk. In the last case you can use the symbol "~" for you home directory (on any platform). It is recommended to click "browse" instead of typing in text by hand

Creating the job definitions is now straightforward:

- perform a search on the "datasets" tab
- select the row with the dataset you just created
- click "Create job definition(s)"

- in the pop-up window that appears there is only one field to fill out: "compilation". You can type either "0" (to not have the code recompiled) or "1" (to have the code recompiled). It is recommended to type 0 (or leave blank)
- in the pop-up window, click "Create". This will pop up a confirmation dialogue, displaying all the information GridPilot is going to store about this job
- click " OK for all"



jobDefinition

Define job records for dataset ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul

compilation : 0

Output mapping

ctb_MyOutputFile.root : Local name : ctb_MyOutputFile.root -> Remote name : $o/$n.${i:5}.root

PoolFileCatalog.xml : Local name : PoolFileCatalog.xml -> Remote name : $o/$n.${i:5}.xml

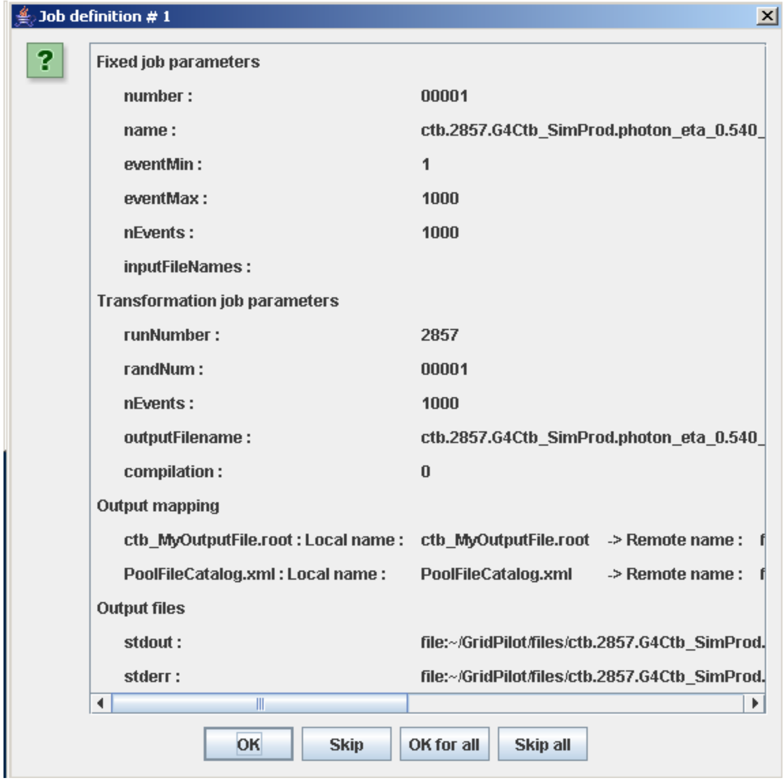
Log files

stdout : $o/$n.${i:5}.stdout

stderr : $o/$n.${i:5}.stderr

☐ Show Details ☑ Show before writing to DB

Creating job definitions.



Job definition # 1

Fixed job parameters

number : 00001

name : ctb.2857.G4Ctb_SimProd.photon_eta_0.540_

eventMin : 1

eventMax : 1000

nEvents : 1000

inputFileNames :

Transformation job parameters

runNumber : 2857

randNum : 00001

nEvents : 1000

outputFilename : ctb.2857.G4Ctb_SimProd.photon_eta_0.540_

compilation : 0

Output mapping

ctb_MyOutputFile.root : Local name : ctb_MyOutputFile.root -> Remote name : f

PoolFileCatalog.xml : Local name : PoolFileCatalog.xml -> Remote name : f

Output files

stdout : file:~/GridPilot/files/ctb.2857.G4Ctb_SimProd.

stderr : file:~/GridPilot/files/ctb.2857.G4Ctb_SimProd.

Confirmation dialog when creating job definitions.

Notice that the "Transformation job parameters" are filled out automatically by GridPilot: `randNum`, `nEvents` and `outputFilename` are recognized as standard parameters, `runNumber` is found using the "metaData" information of the dataset

You can now submit the jobs:

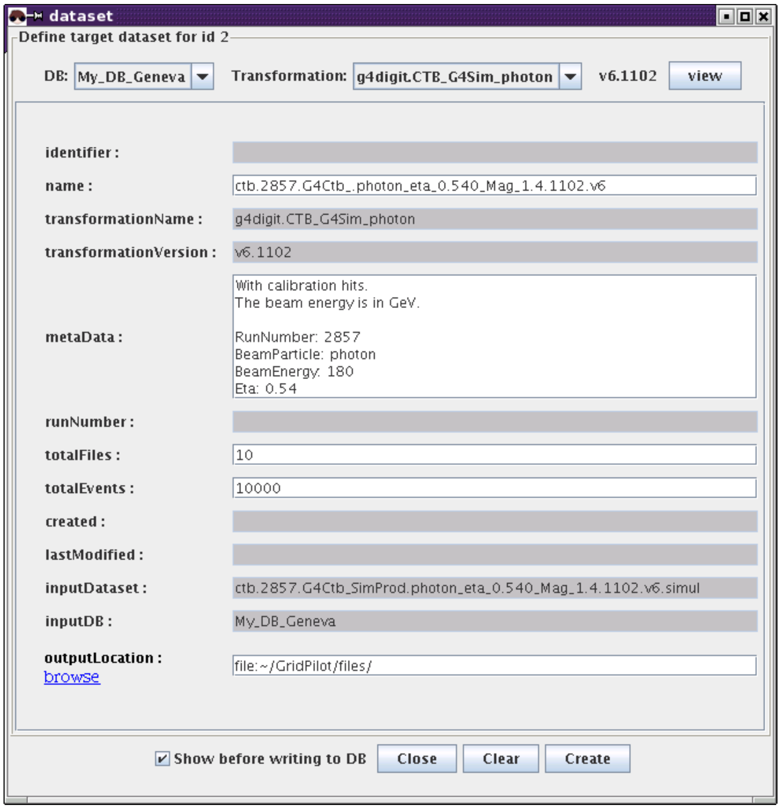
- on the "datasets" tab, select the row with the dataset you just created
- click "View job definition(s)"
- on the "jobDefinitions" tab that opens, select your job definitions and click "Submit", then choose one of the computing systems that appear
- on the job monitor you can now follow the progress of the jobs by clicking "Refresh" and/or using the right-click menu

ATLAS Combined Test Beam photon digitization

In this example we will run so-called "digitization" on the files we produced in the previous example. We will use the transformation script http://cern.ch/fjob/gridpilot/transformations/g4sim.CTB_G4Sim_photon.v6 in the simplest possible way: one input data file and one output data file. The GridPilot transformation record is produced like in the precedent example.

We define the dataset like above, but with a slight change:

- open a "datasets" tab with the same database system
- **select the dataset you created above**
- click "Define new record(s)
- in the pop-up window, **choose the transformation you just created**
- fill in the table in the pop-up window like in the figure below; notice that the fields are already filled with the values of the input dataset - change as appropriate



Define target dataset for id 2

DB: My_DB_Geneva Transformation: g4digit.CTB_G4Sim_photon v6.1102

identifier :	
name :	ctb.2857.G4Ctb_.photon_eta_0.540_Mag_1.4.1102.v6
transformationName :	g4digit.CTB_G4Sim_photon
transformationVersion :	v6.1102
metaData :	With calibration hits. The beam energy is in GeV. RunNumber: 2857 BeamParticle: photon BeamEnergy: 180 Eta: 0.54
runNumber :	
totalFiles :	10
totalEvents :	10000
created :	
lastModified :	
inputDataset :	ctb.2857.G4Ctb_SimProd.photon_eta_0.540_Mag_1.4.1102.v6.simul
inputDB :	My_DB_Geneva
outputLocation : browse	file:~/GridPilot/files/

☑ Show before writing to DB

Creating an ATLAS digitization dataset.

Now you can create and submit jobs like in the previous example.

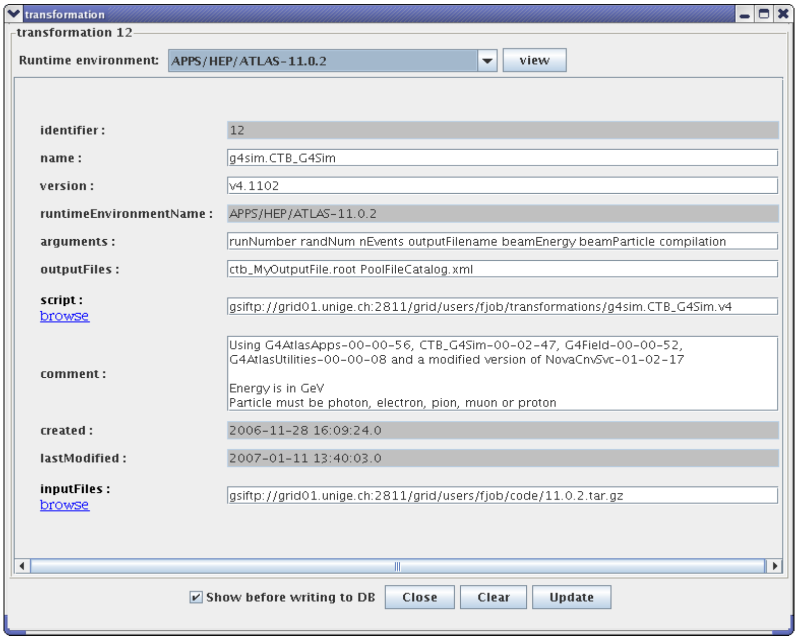
ATLAS Combined Test Beam electron (proton or muon) simulation

In this example we will simulate some ATLAS electron events from the Combined Test Beam exercise in 2004.

We will follow the same procedure as in the previous two examples.

A suitable transformation script can be accessed at http://cern.ch/fjob/gridpilot/transformations/g4sim.CTB_G4Sim.v4. To put this script to use, we have to create a record for it in GridPilot:

- open a "transformations" tab with any database system (where you have write access), e.g. "My_DB_local": "View" → "My_DB_local" → "transformations"
- click "Define new record(s)
- in the pop-up window, choose the runtime environment "ATLAS-11.0.2" from the drop-down list
- fill in the table in the pop-up window like in the figure below



transformation 12

Runtime environment: APPS/HEP/ATLAS-11.0.2

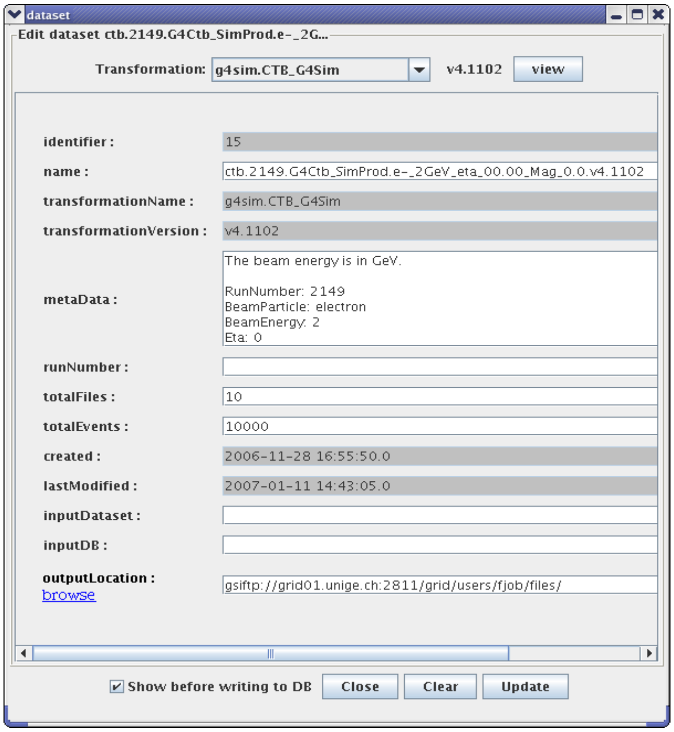
identifier :	12
name :	g4sim.CTB_G4Sim
version :	v4.1102
runtimeEnvironmentName :	APPS/HEP/ATLAS-11.0.2
arguments :	runNumber randNum nEvents outputFilename beamEnergy beamParticle compilation
outputFiles :	ctb_MyOutputFile.root PoolFileCatalog.xml
script : browse	gsiftp://grid01.unige.ch:2811/grid/users/fjob/transformations/g4sim.CTB_G4Sim.v4
comment :	Using G4AtlasApps-00-00-56, CTB_G4Sim-00-02-47, G4Field-00-00-52, G4AtlasUtilities-00-00-08 and a modified version of NovaCnvSvc-01-02-17 Energy is in GeV Particle must be photon, electron, pion, muon or proton
created :	2006-11-28 16:09:24.0
lastModified :	2007-01-11 13:40:03.0
inputFiles : browse	gsiftp://grid01.unige.ch:2811/grid/users/fjob/code/11.0.2.tar.gz

☑ Show before writing to DB

Creating an ATLAS transformation for CTB simulation.

Next, we define a dataset:

- open a "datasets" tab with the same database system
- click "Define new record(s)
- in the drop-down in the pop-up window, **choose the transformation you just created**
- fill in the table in the pop-up window like in the figure below



Edit dataset ctb.2149.G4Ctb_SimProd.e-_2G...

Transformation: g4sim.CTB_G4Sim v4.1102

identifier :	15
name :	ctb.2149.G4Ctb_SimProd.e-_2GeV_eta_00.00_Mag_0.0.v4.1102
transformationName :	g4sim.CTB_G4Sim
transformationVersion :	v4.1102
metaData :	The beam energy is in GeV. RunNumber: 2149 BeamParticle: electron BeamEnergy: 2 Eta: 0
runNumber :	
totalFiles :	10
totalEvents :	10000
created :	2006-11-28 16:55:50.0
lastModified :	2007-01-11 14:43:05.0
inputDataset :	
inputDB :	
outputLocation : browse	gsiftp://grid01.unige.ch:2811/grid/users/fjob/files/

☑ Show before writing to DB

Creating an ATLAS electron simulation dataset.

Comments

- as "name" you can choose anything you like, but bear in mind that ATLAS has a naming convention that you may want to follow
- the "metaData" should contain some description of the data this dataset will contain. Lines of the form `[field]: [value]` are special: the values can be accessed when creating jobs (see below)
- the "outputLocation" can be either a directory on a GSIFTP server or a directory on your local hard disk. In the last case you can use the symbol "~" for you home directory (on any platform). It is recommended to click "browse" instead of typing in text by hand

Creating the job definitions is now straightforward:

- perform a search on the "datasets" tab
- select the row with the dataset you just created
- click "Create job definition(s)"
- in the pop-up window that appears there is only one field to fill out: "compilation". You can type either "0" (to not have the code recompiled) or "1" (to have the code recompiled). It is recommended to type 0 (or leave blank)
- in the pop-up window, click "Create". This will pop up a confirmation dialogue, displaying all the information GridPilot is going to store about this job
- click " OK for all"

Notice that the "Transformation job parameters" are filled out automatically by GridPilot: `randNum`, `nEvents` and `outputFilename` are recognized as standard parameters, `runNumber, beamEnergy, beamParticle` are found using the "metaData" information of the dataset

You can now submit the jobs:

- on the "datasets" tab, select the row with the dataset you just created
- click "View job definition(s)"
- on the "jobDefinitions" tab that opens, select your job definitions and click "Submit", then choose one of the computing systems that appear
- on the job monitor you can now follow the progress of the jobs by clicking "Refresh" and/or using the right-click menu

The output files can be digitized by the same procedure as for the photons (see above), using the transformation script http://cern.ch/fjob/gridpilot/transformations/g4digit.CTB_G4Sim.v4.

ATLAS mSugra simulation

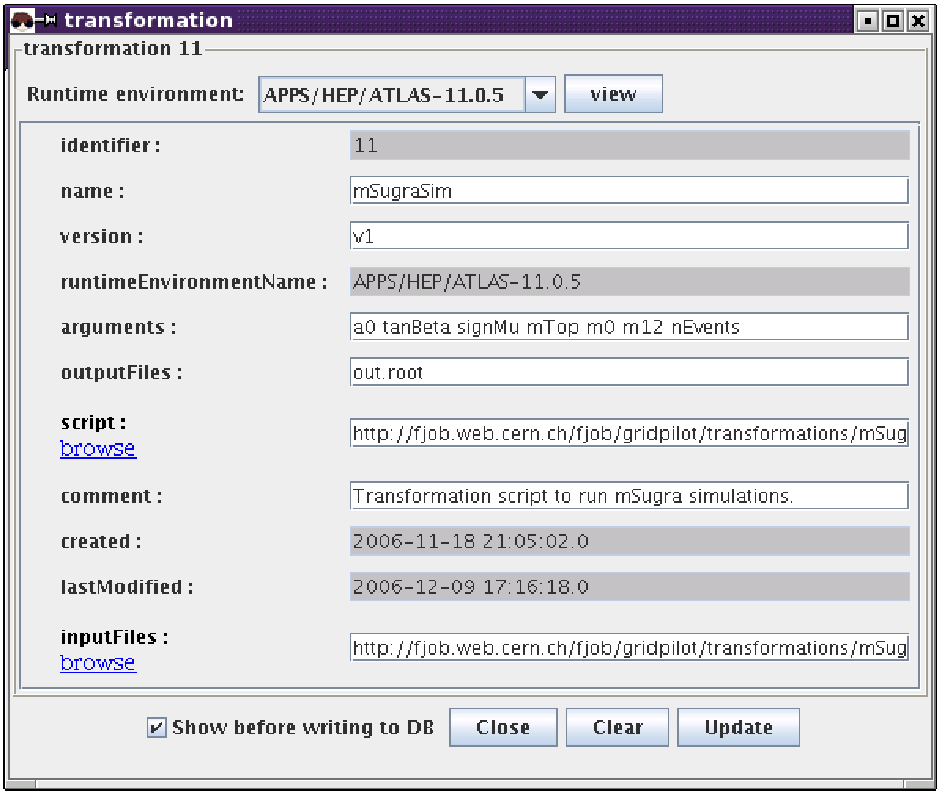
Task: Simulate 1'681 x 10'000 ATLAS mSugra events with 41x41 different values of the mSugra parameters *m0* and *m12*: 0,100,200,...,4000 x 0,25,50,...,1000.

Details

- the ATLAS software release to be used is 11.0.5
- the transformation script at http://cern.ch/fjob/gridpilot/transformations/mSugraSim.pl
- input file for the transformation script: http://cern.ch/fjob/gridpilot/mSugraInputs.tar.gz
- the script takes three parameters: *A0, tan(β), sign(μ) m_Top, m0, m1,* [number of events]
- of these, the first 4 are static for this dataset, i.e. do not change for the whole production
- output should be copied to a GridFTP server

Steps

- **Create a transformation:**



transformation

transformation 11

Runtime environment: APPS/HEP/ATLAS-11.0.5

identifier : 11

name : mSugraSim

version : v1

runtimeEnvironmentName : APPS/HEP/ATLAS-11.0.5

arguments : a0 tanBeta signMu mTop m0 m12 nEvents

outputFiles : out.root

script : http://fjob.web.cern.ch/fjob/gridpilot/transformations/mSug

comment : Transformation script to run mSugra simulations.

created : 2006-11-18 21:05:02.0

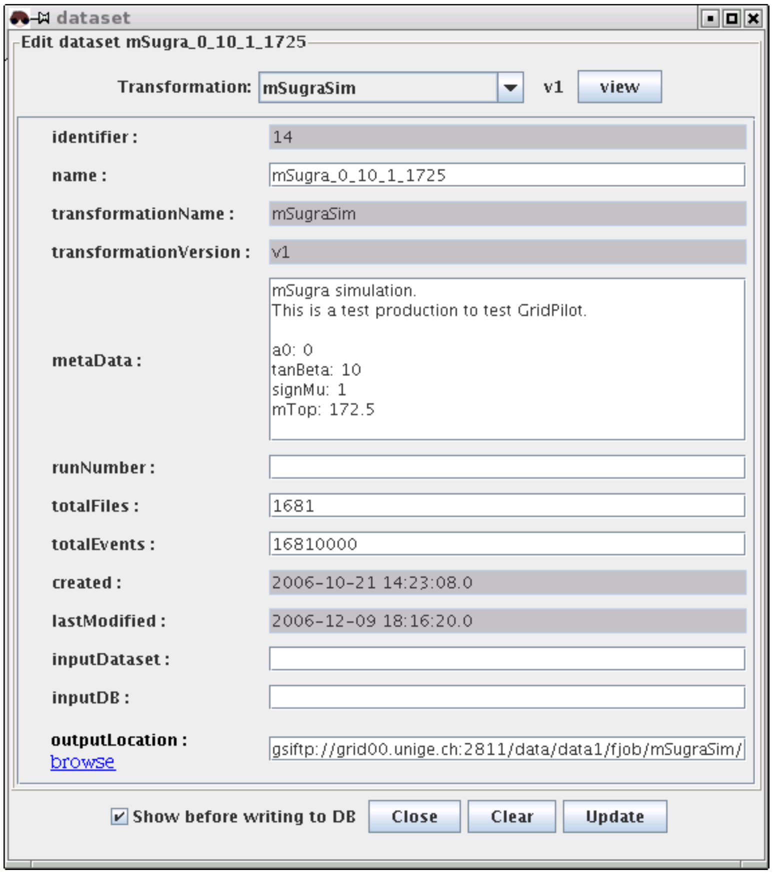
lastModified : 2006-12-09 17:16:18.0

inputFiles : http://fjob.web.cern.ch/fjob/gridpilot/transformations/mSug

☑ Show before writing to DB

Transformation for simulation of ATLAS mSugra events.

- **Create a dataset:** Notice that the static parameters have been given as metadata



Edit dataset mSugra_0_10_1_1725

Transformation: mSugraSim v1

identifier :	14
name :	mSugra_0_10_1_1725
transformationName :	mSugraSim
transformationVersion :	v1
metaData :	mSugra simulation. This is a test production to test GridPilot. a0: 0 tanBeta: 10 signMu: 1 mTop: 172.5
runNumber :	
totalFiles :	1681
totalEvents :	16810000
created :	2006-10-21 14:23:08.0
lastModified :	2006-12-09 18:16:20.0
inputDataset :	
inputDB :	
outputLocation : browse	gsiftp://grid00.unige.ch:2811/data/data1/fjob/mSugraSim/

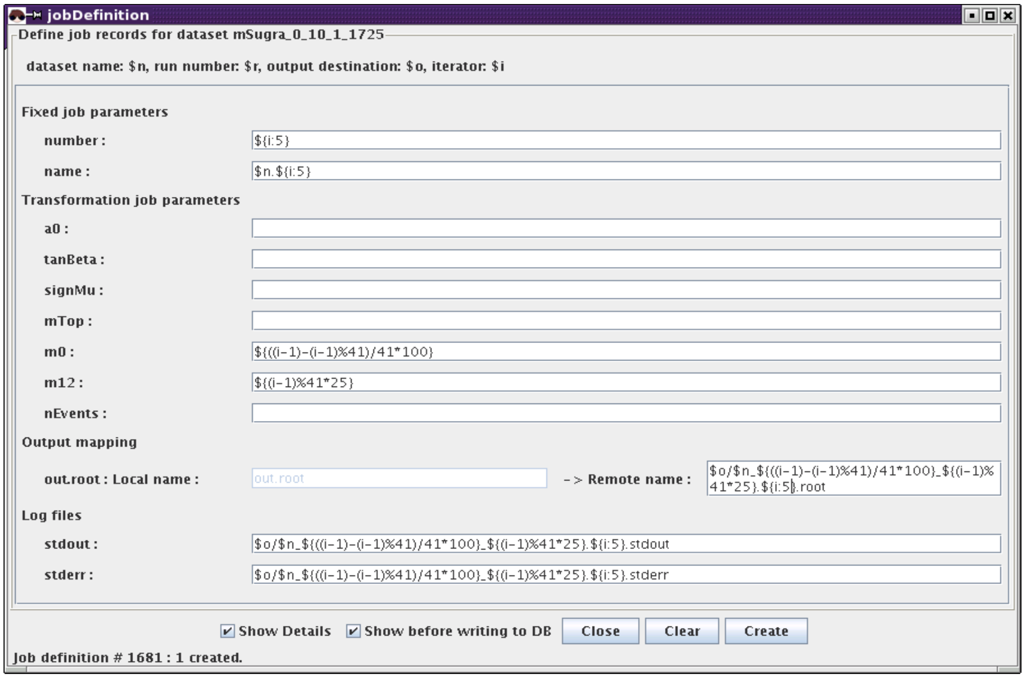
☑ Show before writing to DB

ATLAS mSugra dataset.

- **Create the job definitions:**

Comments

- we have to find an algorithm to map i = 1,2,3,...,1681 to the dublets (m0,m12) = (0,0), (0,25),..., (4000,1000). The answer is: ((i - mod(i, 41))/41*100, (mod(i, 41))*25), with i=0,1,...,1680
- in order to use this algorithm in the job definition fields, we check "Show details"
- then we fill in the fields, using `${((i-1)-(i-1)%41)/41*100}` for "m0" and `${(i-1)%41*25}` for "m12". Here we use the fact that basic arithmetics expressions enclosed by `${...}` in the job definition fields, are interpreted by GridPilot
- notice that the fields that only appear after clicking "Show details" can safely be left blank, as they are filled in automatically by GridPilot



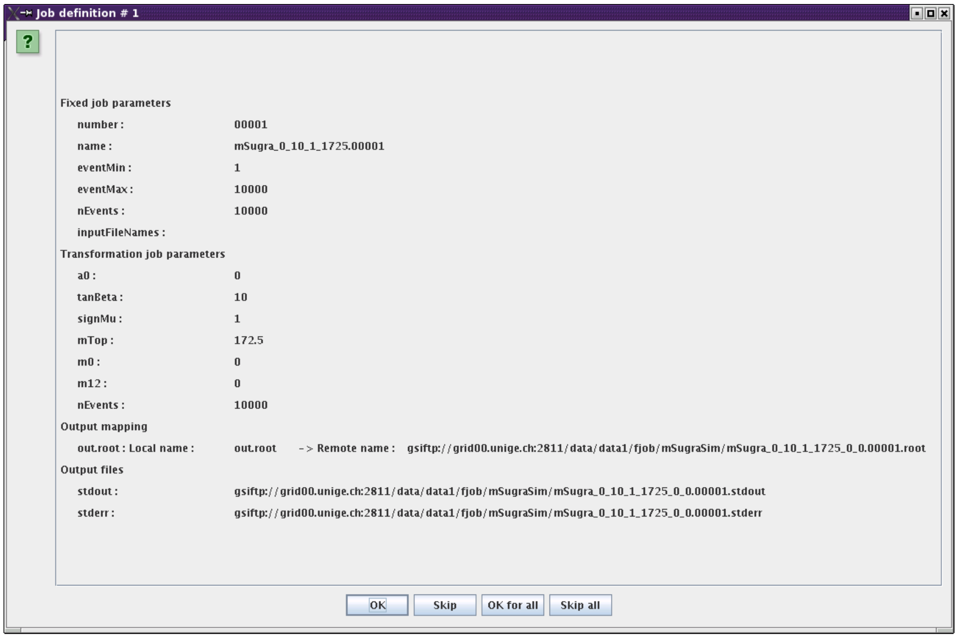
```
Define job records for dataset mSugra_0_10_1_1725

 dataset name: $n, run number: $r, output destination: $o, iterator: $i

Fixed job parameters
    number :              ${i:5}
    name :                $n.${i:5}
Transformation job parameters
    a0 :
    tanBeta :
    signMu :
    mTop :
    m0 :                  ${((i-1)-(i-1)%41)/41*100}
    m12 :                 ${(i-1)%41*25}
    nEvents :
Output mapping
    out.root : Local name :   out.root      -> Remote name :  $o/$n_${((i-1)-(i-1)%41)/41*100}_${(i-1)%
                                                              41*25}.${i:5}.root
Log files
    stdout :              $o/$n_${((i-1)-(i-1)%41)/41*100}_${(i-1)%41*25}.${i:5}.stdout
    stderr :              $o/$n_${((i-1)-(i-1)%41)/41*100}_${(i-1)%41*25}.${i:5}.stderr

☑ Show Details  ☑ Show before writing to DB
Job definition # 1681 : 1 created.
```

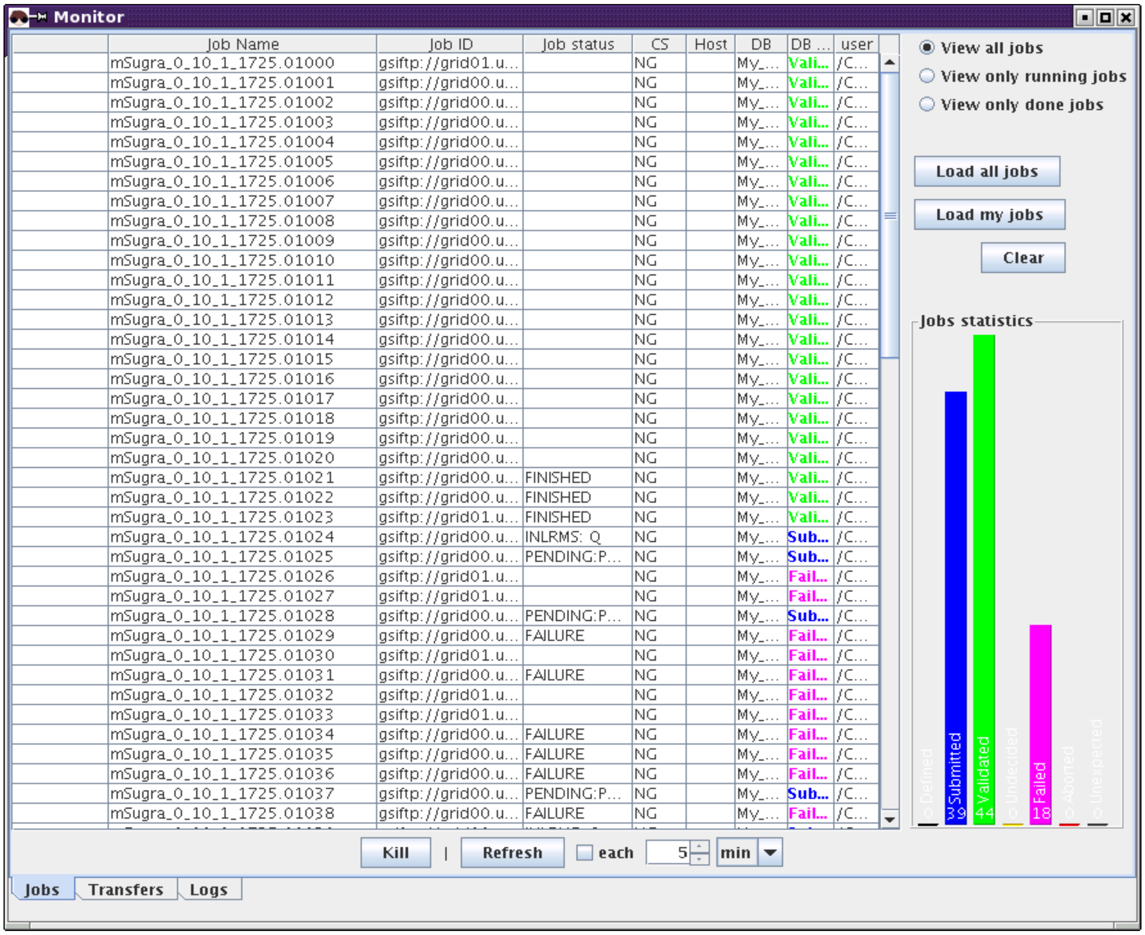
Creating ATLAS mSugra job definitions.



```
Fixed job parameters
    number :              00001
    name :                mSugra_0_10_1_1725.00001
    eventMin :            1
    eventMax :            10000
    nEvents :             10000
    inputFileNames :
Transformation job parameters
    a0 :                  0
    tanBeta :             10
    signMu :              1
    mTop :                172.5
    m0 :                  0
    m12 :                 0
    nEvents :             10000
Output mapping
    out.root : Local name :   out.root    -> Remote name :   gsiftp://grid00.unige.ch:2811/data/data1/fjob/mSugraSim/mSugra_0_10_1_1725_0_0.00001.root
Output files
    stdout :              gsiftp://grid00.unige.ch:2811/data/data1/fjob/mSugraSim/mSugra_0_10_1_1725_0_0.00001.stdout
    stderr :              gsiftp://grid00.unige.ch:2811/data/data1/fjob/mSugraSim/mSugra_0_10_1_1725_0_0.00001.stderr
```

Confirmation dialog when creating ATLAS mSugra job definitions.

- **Submit and monitor the jobs** as in the previous examples. With 1681 jobs involved, this is a larger production. It is recommended to submit batches of a few hundred jobs at a time. The figure below shows the situation a day after 100 jobs were submitted. A good deal of them have failed because the `(m0, m12)` parameter dublet was outside of the allowed range. The rest are either running or have finished correctly and been validated.



	Job Name	Job ID	Job status	CS	Host	DB	DB ...	user
	mSugra_0_10_1_1725.01000	gsiftp://grid01.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01001	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01002	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01003	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01004	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01005	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01006	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01007	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01008	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01009	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01010	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01011	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01012	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01013	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01014	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01015	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01016	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01017	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01018	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01019	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01020	gsiftp://grid00.u...		NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01021	gsiftp://grid00.u...	FINISHED	NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01022	gsiftp://grid00.u...	FINISHED	NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01023	gsiftp://grid01.u...	FINISHED	NG		My_...	Vali...	/C...
	mSugra_0_10_1_1725.01024	gsiftp://grid00.u...	INLRMS: Q	NG		My_...	Sub...	/C...
	mSugra_0_10_1_1725.01025	gsiftp://grid00.u...	PENDING:P...	NG		My_...	Sub...	/C...
	mSugra_0_10_1_1725.01026	gsiftp://grid01.u...		NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01027	gsiftp://grid01.u...		NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01028	gsiftp://grid00.u...	PENDING:P...	NG		My_...	Sub...	/C...
	mSugra_0_10_1_1725.01029	gsiftp://grid00.u...	FAILURE	NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01030	gsiftp://grid01.u...		NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01031	gsiftp://grid00.u...	FAILURE	NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01032	gsiftp://grid01.u...		NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01033	gsiftp://grid01.u...		NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01034	gsiftp://grid00.u...	FAILURE	NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01035	gsiftp://grid00.u...	FAILURE	NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01036	gsiftp://grid00.u...	FAILURE	NG		My_...	Fail...	/C...
	mSugra_0_10_1_1725.01037	gsiftp://grid00.u...	PENDING:P...	NG		My_...	Sub...	/C...
	mSugra_0_10_1_1725.01038	gsiftp://grid00.u...	FAILURE	NG		My_...	Fail...	/C...

Monitoring running mSugra jobs.

Analyzing ATLAS AOD registered in DQ2

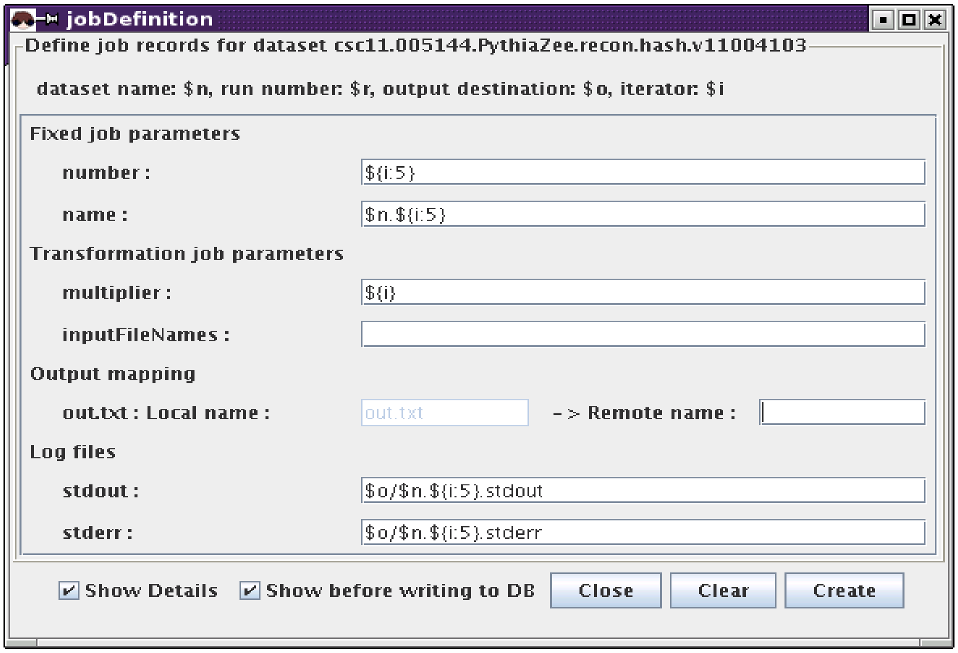
For simplicity we will use the test transformation, "test" that is shipped with GridPilot. The transformation script "test.sh" takes two arguments: `multiplier` and `inputFileNames` (a comma separated list of file names) and calculates a hash number from that, which it writes out to a files "out.txt". We will see how we can use this transformation on input files registered in DQ2. This example is for illustration only; once you've understood how this works, you should define your own transformation that does something sensible with the input files.

- open a "datasets" tab with the ATLAS database system; let's again search for the dataset `csc11.005144.PythiaZee.recon.AOD.v11004103`
- select the dataset record and click "Define new record(s)"
- select "My_DB_Local" from the drop-down list on the window that is popped up
- if more than one transformation is present in your local database, another drop-down list will appear from which you can choose the transformation "test"; otherwise you will simply see "Transformation: test 0.1"
- fill out the "NAME" field with a name of your choice, e.g. `csc11.005144.PythiaZee.recon.hash.v11004103`

- fill out the field "OUTPUTLOCATION" with a location of your choice, e.g. `file:~/GridPilot/files/`
- fill out the field "TOTALFILES" with e.g. `3`. If this field were left blank all 178 job definitions would be generated. Like this, only the first 3 will be generated, which is fine, since this is only a test
- click "Create" and "OK" in the confirmation dialog

- click on the "datasets" tab with "My_DB_local"
- click "Create job definition(s)"

- fill in some number in the field "multiplier", e.g. `${i}`
- click the checkbox "Show details"
- clear the field "Remote name". This causes the name to be generated automatically, by simply appending ".out" to the input file name
- click "Create"


jobDefinition

Define job records for dataset csc11.005144.PythiaZee.recon.hash.v11004103

dataset name: $n, run number: $r, output destination: $o, iterator: $i

Fixed job parameters

number :	${i:5}
name :	$n.${i:5}

Transformation job parameters

multiplier :	${i}
inputFileNames :	

Output mapping

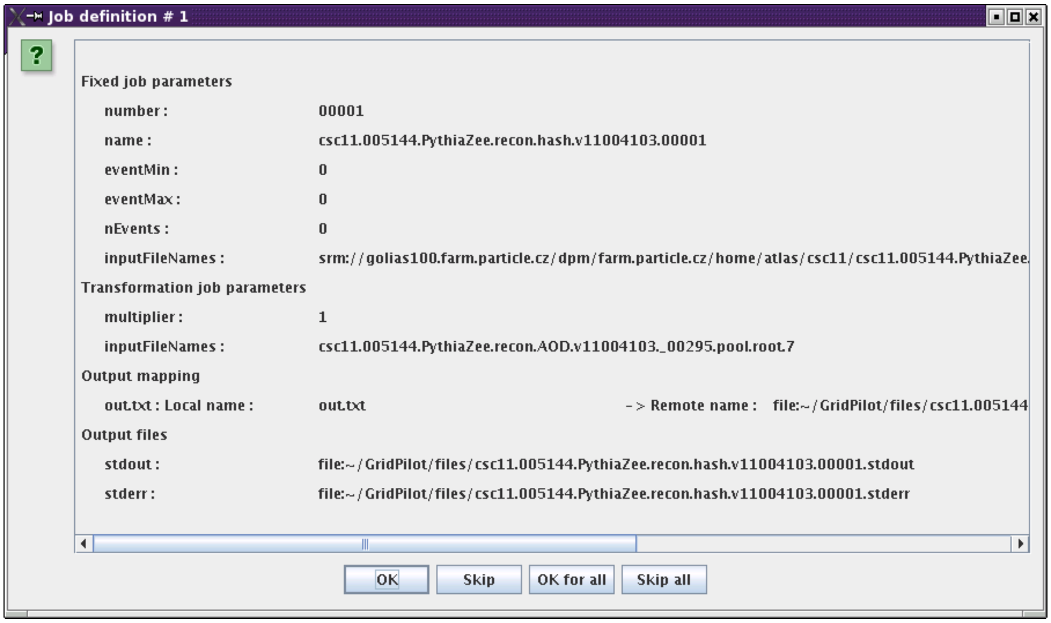
out.txt : Local name : out.txt -> Remote name :

Log files

stdout :	$o/$n.${i:5}.stdout
stderr :	$o/$n.${i:5}.stderr

☑ Show Details ☑ Show before writing to DB

Creating jobs with input files from the ATLAS data management system DQ2.



```
Fixed job parameters
    number :                 00001
    name :                   csc11.005144.PythiaZee.recon.hash.v11004103.00001
    eventMin :               0
    eventMax :               0
    nEvents :                0
    inputFileNames :         srm://golias100.farm.particle.cz/dpm/farm.particle.cz/home/atlas/csc11/csc11.005144.PythiaZee.
Transformation job parameters
    multiplier :             1
    inputFileNames :         csc11.005144.PythiaZee.recon.AOD.v11004103._00295.pool.root.7
Output mapping
    out.txt : Local name :   out.txt                          -> Remote name :   file:~/GridPilot/files/csc11.005144
Output files
    stdout :                 file:~/GridPilot/files/csc11.005144.PythiaZee.recon.hash.v11004103.00001.stdout
    stderr :                 file:~/GridPilot/files/csc11.005144.PythiaZee.recon.hash.v11004103.00001.stderr
```

Confirmation dialog when creating jobs with input files from the ATLAS datamanagement system DQ2. Notice that the destination file has been named after the input file.

- after the jobs have been created, you can submit them to "NG"

Remark

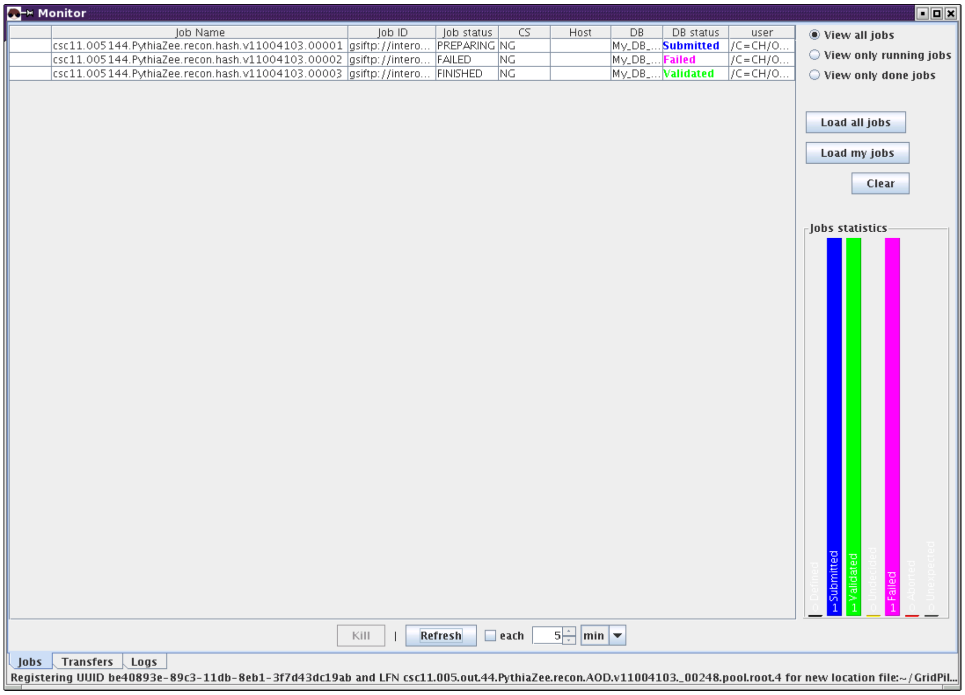
If you want to run the jobs on your local computer, you need to add the following two lines to your configuration file, in the section `[FORK]`:

```
remote copy command = ngcp
required runtime environment = ARC
```

and create a file "ARC" in your local runtime directory (configured by `runtime directory`)

The contents of this file could for example be:

```
currentDir=$PWD
cd ~/nordugrid-arc-standalone-0.5.57
source setup.sh
echo "[my password]" | grid-proxy-init -pwstdin
cd $currentDir
```



Monitor

	Job Name	Job ID	Job status	CS	Host	DB	DB status	user
	csc11.005144.PythiaZee.recon.hash.v11004103.00001	gsiftp://intero...	PREPARING	NG		My_DB_...	Submitted	/C=CH/O...
	csc11.005144.PythiaZee.recon.hash.v11004103.00002	gsiftp://intero...	FAILED	NG		My_DB_...	Failed	/C=CH/O...
	csc11.005144.PythiaZee.recon.hash.v11004103.00003	gsiftp://intero...	FINISHED	NG		My_DB_...	Validated	/C=CH/O...

● View all jobs
○ View only running jobs
○ View only done jobs

```
Registering UUID be40893e-89c3-11db-8eb1-3f7d43dc19ab and LFN csc11.005.out.44.PythiaZee.recon.AOD.v11004103._00248.pool.root.4 for new location file:~/GridPil...
```

Running ATLAS jobs. One has failed because getting the input file timed out. It should be resubmitted.

Appendix A: The configuration file

The configuration file *must* be present and must be in one of the following locations:

for UNIX/Linux:

- [home directory]/.gridpilot
- [gridpilot installation directory]/gridpilot.conf

Here [home directory] will typically be something like "/home/myusername".

for MS Windows:

- [home directory]\gridpilot.conf
- [gridpilot installation directory]\gridpilot.conf

Here [home directory] will typically be something like "C:\Documents and Settings\My Name"

When running GridPilot>0.3.0 for the first time, a configuration file will be created automatically. This file can then be edited later by choosing “Edit” → “Preferences”. In particular, you may want to change which computing, file transfer and database systems are enabled, which defaults are set for download locations (where finished jobs store their output), etc.

If changes are made to the configuration file while GridPilot is running, they will not be effective until after GridPilot has been restarted.

Below follows a list of all configuration parameters, together with a short description and suggestions for settings. Notice that each `[name] [value]` *must* be on *one* line.

Appendix B: Bug reports and feature requests

Before reporting a bug, please check if it has not already been reported and if it is not a know issue (see below).

Known issues

- On some platforms, in rare cases, the GUI may freeze in a non-reproducible manner. This seems to be a Swing issue. Any suggestions on what the exact cause may be are welcome. The 'old' (refactored several times without much change) class "Table" is under suspicion. The solution is to kill (under MS Windows, use the Task Manager) and restart GridPilot
- Tables with more than a few hundred rows may take a long time in sorting, when clicking on one of the column names. This is because the sorting algorithm of the "Table" class is the simplest possible. It would not be difficult to implement e.g. "bubble sort" and that would presumably improve the situation considerably
- When the MySQL connection is lost, e.g. due to a temporary network failure, it is not communicated to the user. In such a situation, the solution is to select "File" → "Databases" → "Reconnect"
- Transformations can depend on only one runtime environment
- Pasting using ctrl+v works only in tabs where a search has already been carried out. In "fresh" panels one has to use the menu: "Edit" -> "Paste"
- When the database "ATLAS" is enabled in the configuration file, GridPilot may hang on startup if the URL specified by `tiers of atlas = http://atlas.web.cern.ch/Atlas/GROUPS/DATABASE/project/ddm/releases/TiersOfATLASCache.py` is not available. This happens from time to time. The solution is to download the file at a time when it is available and replace the setting with e.g. `tiers of atlas = file:~/GridPilot/TiersOfATLASCache.txt`
- When pasting a dataset into the "ATLAS" "datasets" tab, it will no keep it's identifier (VUID); instead a new one will be generated. This is due to a limitation in the (DQ2) back-end
- The implementation of the ATLAS database system does not use caching. This means that performance is probably far below what it could potentially be

- The NorduGrid/ARC computing system has a very simplistic brokering algorithm: the first suitable cluster with free CPUs is chosen. If no suitable cluster with free CPUs is found, the cluster with the largest total number of CPUs is taken. The state of the clusters (free CPUs) is refreshed for each 10th submitted job. This simple approach has some limitations:
  - data proximity is not taken into account
  - in the (common) situation where all queues are full, one cluster will get all the jobs

Reporting bugs

Bugs can be reported here:

https://savannah.cern.ch/bugs/?func=additem&group=atcom

Please give as full a description as possible. This includes choosing the GridPilot release and specifying the operating system, Java version, etc.

To get full debug information from GridPilot, set `debug = 3` in the configuration file.

Feature requests are very welcome. In particular it would be interesting to know which features, if any, would be needed for GridPilot to be useful outside of high energy physics. Feature requests can be submitted the same place, by choosing "Severity" → "Wish".

Acknowledgements

The work on GridPilot was motivated by a desire to simplify small data productions with the ATLAS off-line software, thus enabling single individuals to carry out data analysis, using a large number of processors and accessing grid storage elements. To some extent, the code draws on previous work of the ATLAS production group at CERN and also has contributions from developers outside of CERN. In particular, I would like to thank: Cyril Topfel who contributed the authenticated DQ2 functionality, Marco Niinimaki who demonstrated how to build the code as an applet, and finally Luc Goossens and Vandy Berten who wrote the first GUI for ATLAS data production.