Simplifying use of the Grid

Robert Currie(1), Ulrik Egede(1), Johannes Elmsheuser(2), Robert Fay(3), Patrick Haworth Owen(1), Alexander John Richards (1), Mark William Slater (4), William Lawrence Sutcliffe (1), Matt Williams (4)

(1) Imperial College Sci., Tech. & Med. (GB), (2) Ludwig-Maximilians-Univ. Muenchen (DE), (3) University of Liverpool (GB), (4) University of Birmingham (GB)

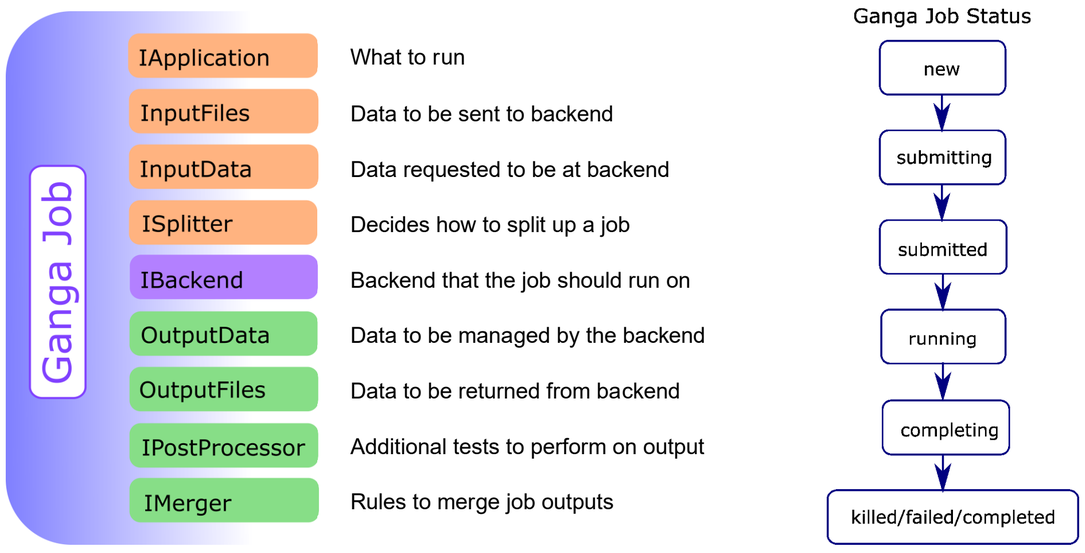
Introduction to Ganga

Ganga is an easy to use Job submission and management tool developed in python. The project has a large user-base with approximately 300 active daily users and is responsible for submitting and managing over 100,000 jobs per day. It has a proven track record and was heavily used by both the ATLAS and LHCb communities throughout Run1 of the LHC.
Ganga allows users to automate the tasks required to submit user programs varying from custom pyROOT applications to complex physics analyses developed around the Gaudi/Athena C++ frameworks.
Job submission in Ganga is commonly performed through either a user written Ganga python scripts or an interactive Ganga environment making use of an IPython console interface.

Ganga Job Management

Ganga makes use of a set of plugin interfaces which allows for jobs to be described using a minimal job script. The configuration of these jobs is saved to disk using an XML based repository storing a jobs configuration and its current state.
A Ganga job is constructed using 9 possible plugins which can be used to describe how a job is to be structured. These plugins make use the custom Ganga framework with objects inheriting from GangaObject class.

Ganga is able to package the chosen Application to be run and pass it to the chosen processing backend. Using the inputfiles and inputdata objects Ganaga is able to manage the data to be used by the Job. Additionally outputfiles and outputdata are used to manage any output data produced by the Job.



Once a job has finished running Ganga is then able to automatically post-process and merge the results for the user.
To provide a simple yet configurable job management system, Ganga assigns each job a status. This is derived from the status of the job at the backend. This provides a generic way of managing jobs across multiple backend and provides the user with a generic interface to control their jobs.

File Management

Ganga 6 makes use of a common file interface IGangaFile which allows for file objects to be easily managed and configured.
Using the new IGangaFile interface Ganga is able to determine the most efficient way to transfer data between different storage solutions and the backend where the job is to be run. This allows users to construct jobs which run over a given set of files without having to know the implementation details of the file management. Ganga then handles the steps required to make the data available to the job on the backend.

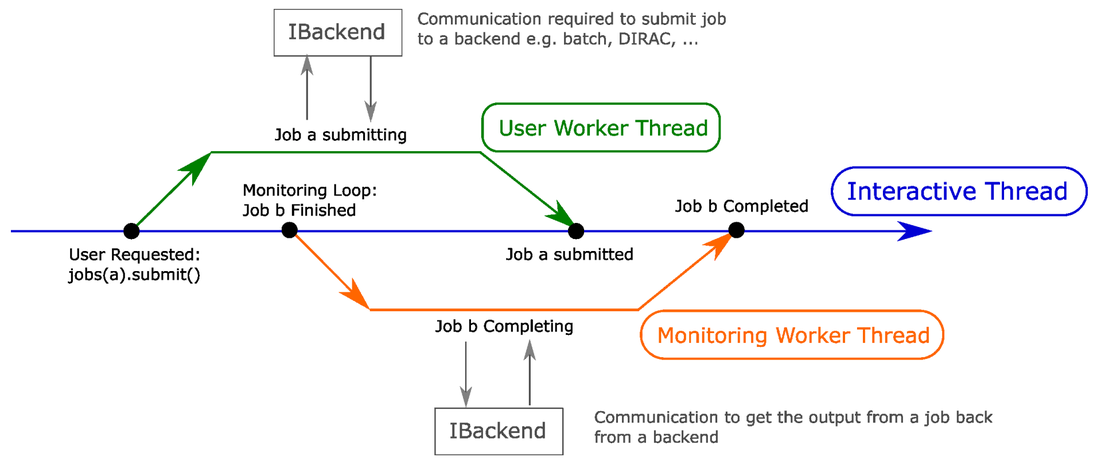
Ganga File Object	Description
LocalFile	Files stored locally on disk accessible through common *NIX commands.
DiracFile	Files stored on a Dirac managed Storage Element
MassStorageFile	Files stored in a locally accessible backup system.
GoogleFile	Files stored on a Google Drive.
LCGSEFile	Files stored on an LCG SE.
CernBoxFile	Files stored on the CERNBox file storage system.

Processing Queues

Ganga 6 includes a new queues interface, accessible from the interactive Ganga prompt. This feature allows for time consuming methods to be completed in separate worker threads to the main interactive thread.

A fixed number of worker threads are constructed on the launch of Ganga to perform User and Monitoring processing. This prevents the Ganga client from overloading the machine it's running on. In order to make the most of the resources available Ganga then queues additional processing requests until a Worker Thread instance becomes free.

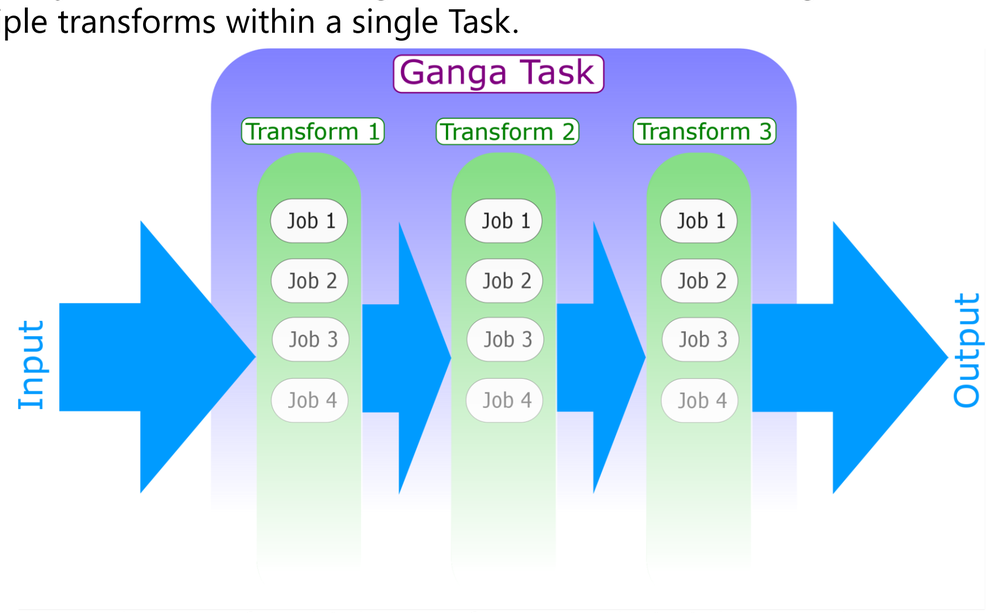
The diagram below demonstrates the new queues mechanism with worker threads. In this scenario the user is able to request for 'Job a' to be submitted to a backend making use of a User Worker Thread. This is then able to execute in a background python thread. Whilst this is running the Monitoring Loop has determined that 'Job b' has completed and begins the task of completing the job in Ganga using a Monitoring Worker Thread.



Through the use of the queues interface users are able to register many different commands to be executed in the background away from the main Interactive Prompt. This allows for users to continue to manage other jobs and data whilst waiting for processes to complete.

Tasks in Ganga

Ganga 6.1 introduces a new generic Tasks system to allow for a complex series of jobs to be chained together. This a achieved through the use of multiple transforms within a single Task.



Tasks can make use of multiple transforms to chain jobs together to process a large computational request. Ganga Transforms are capable of mapping as one-to-one or one-to-many making them useful for data quality checking. Ganga Tasks are capable of propagating data through various transforms as it becomes available. This allows for the whole Task to be completed more efficiently. As well as this, Tasks are able to exploit many features within Ganga such as queueing and auto-resubmitting Jobs.
Ganga provides some default preconfigured Tasks which allows for complex LHCb and ATLAS job chaining to be performed.

Future Developments

Ganga 6.0 has shown to be a stable platform on which to develop upon with Ganga 6.1 taking advantage of this. Looking toward Ganga 6.2 one of the areas of future development is an improved credential management system. One of the goals of this is that it will allow for passing additional user authentication information to jobs running on various backends.
Further improvements will also include additional work and improvements in file and task management features first released in Ganga 6.1.