Ganga user interface for job definition and management

D.L. Adams\textsuperscript{a}, Ph. Charpentier\textsuperscript{b}, K. Ciba\textsuperscript{c}, C. Collins-Tooth\textsuperscript{d}, U. Egede\textsuperscript{e}, K. Harrison\textsuperscript{f}, R.W.L. Jones\textsuperscript{g}, A. Maier\textsuperscript{b}, J.T. Moscicki\textsuperscript{b}, G.N. Patrick\textsuperscript{h}, A. Soroko\textsuperscript{i}, C.L. Tan\textsuperscript{j}

\textsuperscript{a} BNL, Upton, NY 11973-5000, USA
\textsuperscript{b} CERN, CH-1211 Geneva 23, Switzerland
\textsuperscript{c} Institute of Nuclear Physics, 31-342 Krakow, Poland
\textsuperscript{d} Department of Physics and Astronomy, University of Glasgow, G12 8QQ, UK
\textsuperscript{e} Department of Physics, Imperial College London, SW7 2AZ, UK
\textsuperscript{f} Cavendish Laboratory, University of Cambridge, CB3 0HE, UK
\textsuperscript{g} Department of Physics, University of Lancaster, LA1 4YB, UK
\textsuperscript{h} Rutherford Appleton Laboratory, Chilton, Didcot, OX11 0QX, UK
\textsuperscript{i} Department of Physics, University of Oxford, OX1 3RH, UK
\textsuperscript{j} School of Physics and Astronomy, University of Birmingham, B15 2TT, UK

Ganga is a component-based system providing a user interface for job definition and management in a distributed environment. It is being developed in the context of two high-energy physics experiments, ATLAS and LHCb, whose specific it addresses, but offers possibilities for extension and customisation that make it potentially interesting for a wide range of user communities. It is implemented in Python.

ATLAS and LHCb will investigate various aspects of particle production and decay in high-energy proton-proton interactions at the Large Hadron Collider at CERN, Geneva, due to start operation in 2007. Both experiments will require processing of data volumes of the order of petabytes per year, and will rely on computing resources distributed across multiple locations. The experiments’ data-processing applications, including simulation, reconstruction and physics analysis, are all based on the Gaudi C++ framework. This provides core services, such as message logging, data access and histogramming; and allows run-time configuration via options files, which may be written using a value-assignment syntax similar to that of C++ or in Python, and may have considerable complexity. The two experiments have developed different solutions for cataloguing the available data: the ATLAS Metadata Interface (AMI) and the LHCb Bookkeeping Database.

Several possibilities for exploiting distributed computing resources, and for processing distributed data, have been developed using Grid technologies. These include systems from national and international projects, such as EDG/LCG/EGEE, ARDA, Grid3 and NorduGrid, and experiment-specific solution, such as DIAL in ATLAS and DIRAC in LHCb. The situation for the user is confusing, in that different sites hosting data of interest may have adopted different systems, requiring different commands and job descriptions, and all of the systems continue to evolve. Also, a user will often wish to run test jobs on local resources, where debugging is simpler, before submitting to remote sites, and will prefer not to have to do different things in the two cases.

Ganga provides a single frontend for submission to multiple backends, shielding the user from the uncertainties and technical complications associated with data processing in a distributed environment. To further help ATLAS and LHCb users, Ganga integrates an editor tailored for working with Gaudi options files, and tools for querying the experiments’ catalogues. Ganga allows the user to specify the software to run, which may include user-supplied algorithms, to set values for any configurable parameters, to select data for processing, and to submit jobs to a variety of local batch queues and Grid-based systems. After jobs are submitted, Ganga monitors their status, and takes care of saving any output. Ganga offers a Command-Line Interface in Python (CLIP), which can be used for scripting, and a Graphical User Interface (GUI) (Figure 1), which significantly simplifies basic tasks.
The main functionality of Ganga is divided among three core components, corresponding to Application Manager, Job Manager and Job Registry, and is accessed through a fourth component, the Client, on top of which additional layers are added (CLIP and GUI) to enhance usability. The four components are defined in such a way that they could be placed on different hosts, although in the current implementation all components are installed on a single machine.

The Application Manager needs a host where the environment for running the user’s application is set up, and the required software is available. This component takes a user’s locally valid description of how an application should be run, and converts it into a site-independent job representation. This operation may include translating local path and environment settings, and identifying user files that must be shipped with the job.

The Job Manager needs a host that allows submission to one or more backends. This component accepts a job representation defined by the Application Manager, creates a wrapper script for running the specified application on the chosen backend, and generates any additional job-description files that the backend requires. It executes job-related commands – submit, pause, kill, and so on – on behalf of the Client, and will try to satisfy requests for job splitting and subsequent output merging.

The Job Registry is implemented as a server providing access to a database. It acts a a job monitor, storing details of defined jobs and recording changes in job status.

The Ganga core components define public interfaces in relation to their functionality, and manage plug-ins that implement these interfaces for particular cases. For example, the Application Manager has plug-ins for Gaudi-based reconstruction and analysis applications; the Job Manager has plug-ins for backends including LSF, PBS, LCG, gLite, DIAL and DIRAC; the Job Registry has a plug-in for accessing an ARDA lightweight metadata catalogue. Plug-ins for other types of application, backend and database are readily added, allowing Ganga to be extended to match new requirements.

This presentation gives details of Ganga’s design and functionality, describes the use of Ganga in ATLAS and LHCb, and outlines the possibilities for extension and customisation to meet the needs of other user communities.