

CMS AND THE GRID

The Compact Muon Solenoid (CMS) is one of four experiments that will operate at the Large Hadron Collider (LHC) at CERN. The LHC is a proton-proton collider, CMS is a general-purpose detector designed to detect a wide range of new physics, including the Higgs Boson, Supersymmetry and CP violation.

Some 10 Petabytes of data will be produced for each year that CMS operates. Analysis of these data would require around 70,000 of today's fastest computers. This problem is well-suited to the "Grid" paradigm first proposed by Foster and Kesselman.

The idea of the Grid is to integrate the computing power of sites worldwide. The Grid will allow a user to submit a computing job from anywhere in the world, for that job to run at a site somewhere else in the world on data that is not local to the user and for the results to be returned.

The Grid for the LHC is called the LHC Computing Grid (LCG). LCG is designed to fulfil the needs of all the LHC experiments and will be fully operational by the time CMS begins data taking in 2007.

In LCG the data from CMS will be stored at various sites throughout the world. There will be more than one copy of each file to avoid network bottlenecks and data loss. Each file will have two different names; a Logical File Name (LFN) that will be the same for all copies and a Physical File Name (PFN) that will indicate where each copy of the file is physically located. Most of the data in LCG will be "read only" to avoid synchronisation issues between replicas.

Currently CMS data is "published" at a site via a system called PubDB. PubDB allows a site to publish what collections of data it holds, it also points to a catalogue that contains the LFN/PFN mappings for that site, this is used by jobs running at the site.

DISTRIBUTED ANALYSIS USING GROSS

In order for a physicist to analyse data a simple automated tool is needed. The current CMS analysis program is ORCA (Object-oriented Reconstruction for CMS Analysis). This tool performs batch analysis on local data with no knowledge of the Grid. It was our aim to write a program that, while simple for the user, would allow their ORCA analysis code to be run on CMS data over LCG in a fully-distributed way with no alteration to ORCA or their code.

To this end we have developed GROSS (GRidified ORCA Submission System). The current functionality enables a physicist to submit their ORCA analysis code with a simple command and one file to describe their job to GROSS. Typically a user will wish to analyse a dataset, which may be divided into many hundreds of runs. GROSS takes the user's task and splits it into multiple smaller jobs, where each of these jobs will analyse one or more runs. The functionality and design are illustrated in Figure 1.

GROSS has been designed as a lightweight, fully object-oriented system built on top of existing CMS and LCG tools. CMS currently uses a system called BOSS (Batch Object Submission System) to submit jobs to local computer farms and provide monitoring. By building on top of existing technologies, GROSS avoids duplicating features and code.

GROSS takes care of all the steps necessary to run CMS analysis in a distributed way. It prepares multiple sub-jobs from a master task and archives them in a relational database for later use. GROSS can submit these jobs to either LCG or any other batch system. GROSS (via BOSS) is also able to monitor job status and output for all of the submitted jobs in real time. GROSS can automatically retrieve the output of a job once it has finished.

The user interacts with GROSS via a simple CLI (Command Line Interface) and a JDL (Job Description Language) file created by the user. JDL files follow a simple Classad design. GROSS takes this file, with its own keys, and uses it to create and submit the jobs.

GROSS utilises an abstract factory design in order to create fully objects of the same family. Different factory types are used for each different type of job that can be submitted. Currently there are two; ORCA jobs for LCG submission and ORCA jobs for local batch system submission.

Job creation and preparation can be achieved with one command. GROSS reads in information from the JDL file and creates the appropriate objects and files. This information is then saved into the database. A number of files are needed for job submission and to steer the job once it is submitted. These are stored with their relevant objects in the database. This allows jobs to resubmitted by the user at will. Each different factory type has an associated wrapper script stored in the database; this is used to set up the environment, deal with input/output files and to run the users' executable.

Functionality

The core GROSS commands include:

- OneStep – Prepares task, does job splitting and submits the jobs; and

