GROSS: AN END USER TOOL FOR CARRYING OUT BATCH ANALYSIS OF CMS DATA ON THE LCG-2 GRID.

H. Tallini, D. Colling, B. MacEvoy, S. Wakefield, Imperial College London, UK

Abstract

GROSS (GRidified Orca Submission System) is part of a CMS prototyping effort to develop and deploy analysis tools on the Grid. GROSS provides end users with a single interface for running batch analysis tasks over LCG-2. The tool carries out job splitting, preparation, submission, monitoring and archiving in a transparent way which is simple to understand for the end user. Central to its design has been the requirement for allowing multi-user analyses, and to accomplish this all persistent information is stored on a backend MySQL database. This database is additionally shared with BOSS (Batch Object Submission System), to which GROSS interfaces in order to provide job submission and real-time monitoring capability.

In this paper we present an overview of GROSS's architecture and functionality and report on first user tests of the system using CMS Data Challenge 2004 (DC04) data.

CMS AND THE GRID

The Compact Muon Solenoid (CMS) [1] is one of four experiments that will operate at the Large Hadron Collider (LHC) at CERN. The LHC is a proton–proton machine with a centre-of-mass energy of 14 TeV and a luminosity of $10^{34}cm^{-2}s^{-1}$. CMS is a general-purpose detector designed to detect a wide range of new physics, including the Higgs Boson, Supersymmetry and CP violation.

At the LHC, bunch crossings will occur every 25ns with each bunch containing approximately 10^{11} protons.

Figure 1: Output data rate from CMS [2].

The majority of the events produced will be of no interest to physicists, hence a dramatic reduction in data rate is required. An online trigger system determines which events to save and which to ignore. Figure 1 shows the performance of the trigger system.

After this huge reduction the amount of data from CMS to be analysed will still be enormous. The size of each raw event will be about 2MB. These raw events will generally not be analysed directly but will first be summarised into reconstructed events. Even with these data reduction measures 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 [3]. This problem is well-suited to the "Grid" paradigm first proposed by Foster and Kesselman [4].

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 user does not need to know how or where the job ran, only that it did.

LHC Computing Grid

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. Detailed information on LCG may be found in the User Guide [5].

LCG is organised as a hierarchy emanating from CERN, which is also known as Tier-0. Below this level will be large national centres (Tier-1's), followed by regional centres (Tier-2's) and then individual institutes (Tier-3's). As one descends the hierarchy the computing power at each site decreases but the number of such sites increases rapidly to offset this.

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 loss of data. 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. A user will refer to a file by its LFN and the LCG software will translate this into the PFN of the closest replica. The user can then use this PFN to access the file. Most of the data in LCG will be "read only" to avoid synchronisation issues between replicas.

DISTRIBUTED ANALYSIS USING GROSS

Introduction

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) [6]. 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 only a few simple commands 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 2.

GROSS has been designed as a lightweight, fully object-oriented system built on top of existing CMS and LCG data management tools [7]. CMS currently uses a system called BOSS (Batch Object Submission System) [8] to submit jobs to local computer farms and provide monitoring. By building on top of existing technologies, GROSS avoids duplicating features and is isolated from changes in data management and job submission.

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. Once GROSS indicates that the job has finished the output may be retrieved.

Architecture

During the design phase of GROSS the user requirements and use case scenarios concerned with distributed analysis were considered. Some of these were drawn from the existing LCG RTAG (Requirements and Technical Assessment Group) document [9]. One of the main requirements was that a simple solution was needed to allow physicists to concentrate on their analysis code rather than on how to submit it. Another major requirement was that the system should be as modular as possible in order to cope with the inevitable changes in LCG and ORCA that will occur over the lifetime of CMS.

GROSS has a modular design implemented in C++ along with a MySQL [10] RDBMS (Relational Database Management System) for persistency between sessions. More information on GROSS architecture can be found in the user guide [11]. BOSS uses a database to store information about jobs and to monitor their state; GROSS adds to this by also storing the information about analysis tasks, sub-jobs and even allows submission to be from a different computer from where the job was prepared as long as the same database is used.

GROSS interacts with the user via a simple CLI (Command Line Interface) and a JDL (Job Description Language) file created by the user. JDL files are used to submit jobs to the Grid and follow a simple Classad [12] design. The JDL file has some fields shared with LCG submission and some specific to GROSS. GROSS takes this file 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.

GROSS separates the tasks of preparing analysis jobs and submitting them. An object represents each task and each sub-job of that task is represented by a job object. Distinctions are also made between those objects that are created from the instructions in the JDL file and then stored in the database and those that are created from the database. When a user prepares a job, GROSS reads in information from the JDL file and creates the appropriate objects and files. This information can then be saved into the database for later use. Upon submission these data are then read from the database and the objects and files are recreated and submitted. 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.

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 ORCA.

Functionality

The four core GROSS commands are:

- Prepare – Prepares task and does job splitting – information saved to database;
- Submit – Submits prepared jobs;
- Query – Allows querying of job status and real-time output monitoring; and
- Output – retrieval of job output.

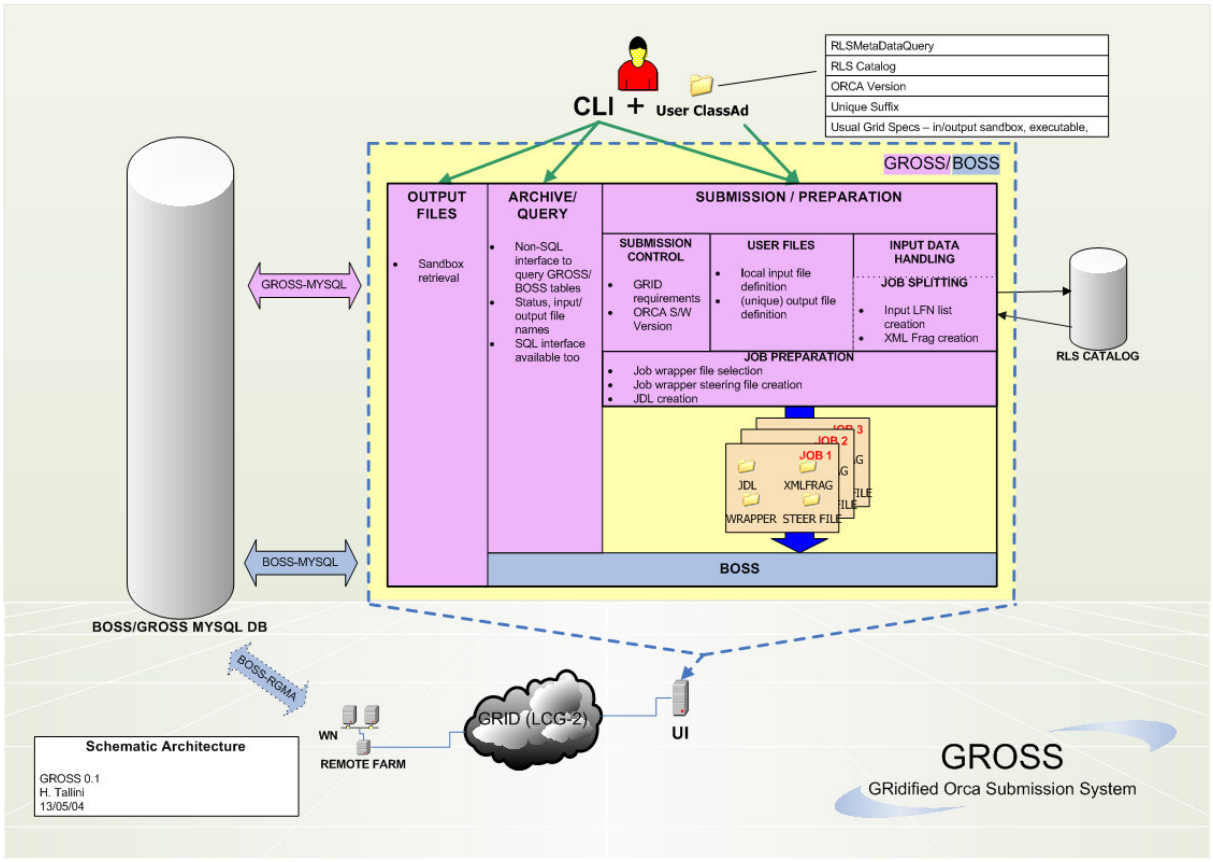

Figure 2: Functionality and design of GROSS.

The physicist user starts with a dataset they wish to analyse and their analysis code. From here they write their JDL file. It is likely that a number of options in the JDL will be common to all of their tasks (e.g. the files required to allow submission to LCG) and most users will therefore establish a common template for repeated use.

The user must specify their executable and state whether it is already resident within the ORCA installation, a local pre-compiled executable or whether they would like their code to be compiled at the start of their job. As well as specifying the dataset to be analysed, the user may also include a subquery on that dataset, e.g. only analysing a selected subset of the data.

In common with normal submission to LCG, the user must specify their input files (not event data but either libraries for their code or steering files for ORCA etc.) and output files. Output files can either be returned to the user when they run the GROSS 'Output' command or registered to LCG and saved to Grid storage. The former option is most appropriate for log files and other small output files. The latter is suitable for analysis where other physicists are likely to be interested in the results; they can then locate these files as they might any others in LCG.

Lastly, a user must specify which version of ORCA they require. This is done by specifying the version of ORCA they wish GROSS to set up when the job starts (sites are likely to have more than one version installed at any one time), and by specifying a search on the LCG tag that sites publish to advertise available software.

If the user wishes to submit their job to a batch system on a local farm they must provide slightly different information. They must specify where they wish their output to be saved locally and the setup files for ORCA.

Once the JDL file has been written the user may prepare their jobs using the "Prepare" command. GROSS reads through the JDL file and, from the dataset and optional subquery, it contacts RefDB [13], the CMS production database, to ensure that the dataset required is available. GROSS then uses the LCG tools to find which files correspond to the dataset. It saves the relevant information to the database and creates the necessary temporary submission files. These files include a steering file to direct the wrapper script, the wrapper script to be submitted and a catalogue file with a list of input files and their LFNs/PFNs in a form understood by ORCA.

Although these files are created they are not needed at submission time, as they are all either stored into the database or recreated dynamically. They simply allow the user to see what GROSS will be submitting and check for errors if they wish. At runtime these files will either be overwritten or recreated depending on whether or not they

still exist. Jobs therefore do not need to be submitted by the same person, or on the same computer, that prepared them as long as the database is available.

When a task is saved to the database an identifying number is allocated which is then used to refer to that task in the future. Each sub-job is also given an id number; all of the GROSS commands work with task ids and also, optionally, job ids. Generally, however, a user will wish to find the status or get the output of an entire task rather than of one sub-job.

To submit a task the user simply issues the appropriate GROSS "Submit" command with the task id and desired batch system type as arguments. GROSS then sends a confirmation message stating how many jobs were submitted.

Once jobs have been submitted the user may monitor their progress with the GROSS "Query" command. Some of the monitoring information is provided by BOSS, including the ability to monitor the output of each job and, if required, to send this information back to be stored in the database and retrieved via the Query command. This is possible because BOSS sends its own program to execute the wrapper script and this program monitors the output. This allows the user to pick up any problems quickly.

Once a job has finished the user may retrieve their output. Jobs submitted to local batch systems will simply store their output in an area specified by the JDL file. Jobs submitted to LCG will either have their output automatically saved to Grid storage or retrieved when the GROSS "Output" command is run. Before downloading the output files, GROSS first checks that the jobs are finished. The output of the completed jobs is downloaded via an LCG API to a separate directory for each job. The location of these files is then entered into the database, and printed on further instances of the Output command.

GROSS use in CMS Data Challenge 2004 (DC04)

In March 2004 CMS undertook a data challenge, the aim of which was to produce and distribute data at a rate comparable with CMS running at 25% of nominal event rate. Another objective was analysis of these data. The analysis that took place during DC04 was automatic online analysis using well-defined "official" analysis code. GROSS was not used officially in DC04 due to delays in development. However, towards the end of DC04, GROSS was used by researchers at Imperial College London to analyse tens of DC04 datasets with simple test analyses. These analyses were carried out on data stored in the UK, Italy and Taiwan.

Future Directions

GROSS currently has basic use case functionality but requires more evaluation in the wider community. It is hoped that the CMS physics analysis groups will use GROSS and provide feedback on features, ease-of-use and any bugs still present.

Some of the functionality of GROSS is not implemented in the most appropriate way. One example is that, when contacting RefDB, GROSS uses HTTP requests and parses the output. This will be rectified. GROSS will shortly make use of Web services using a CMS implementation called Clarens [14]. This service will be used as a layer between GROSS and external services. A new database implementation [15] has also just been released containing information about datasets ready for analysis. We anticipate that, by using Clarens, GROSS will easily integrate with this new system.

References

[1] "The Compact Muon Solenoid: Technical Proposal". The CMS Collaboration, CERN/LHCC 94-38, LHCC/P1, December 1994.

[2]http://wwwseminars.web.cern.ch/wwwseminars/2002-OtherFormats/t-20020925.ppt

[3] S. L. Lloyd, Physics World, October 2003.

[4] Foster, I. & Kesslman, C. (Eds). The Grid: Blueprint for a new Computing Infrastructure. Morgan-Kaufmann (1999).

[5] http://edms.cern.ch/file/454439/LCG-2-UserGuide.html

[6] http://cmsdoc.cern.ch/orca

[7] http://pool.cern.ch/

[8] http://www.bo.infn.it/cms/computing/BOSS/

[9] "Common use cases for a HEP common application layer for analysis", HEPCAL II, LHC-SC2-20-2002

[10] http://www.mysql.com

[11]http://www.hep.ph.ic.ac.uk/e-science/projects/downloads

[12] http://www.cs.wisc.edu/condor/

[13] "RefDB: The Reference Database for CMS Monte Carlo Production", talk given at Computing in High Energy and Nuclear Physics conference 2003, La Jolla, California.

[14] http://clarens.sf.net

[15]http://agenda.cern.ch/askArchive.php?base=agenda&categ=a043475&id=a043475s1t1/transparencies