Scalability Tests of R-GMA-Based Grid Job Monitoring System for CMS Monte Carlo Data Production

Daniele Bonacorsi, D. Colling, L. Field, S. M. Fisher, Claudio Grandi, Peter R. Hobson, Paul Kyberd, B. MacEvoy, J. J. Nebrensky, H. Tallini, and S. Traylen

***Abstract*—High-energy physics experiments, such as the compact muon solenoid (CMS) at the large hadron collider (LHC), have large-scale data processing computing requirements. The grid has been chosen as the solution. One important challenge when using the grid for large-scale data processing is the ability to monitor the large numbers of jobs that are being executed simultaneously at multiple remote sites. The relational grid monitoring architecture (R-GMA) is a monitoring and information management service for distributed resources based on the GMA of the Global Grid Forum. In this paper, we report on the first measurements of R-GMA as part of a monitoring architecture to be used for batch submission of multiple Monte Carlo simulation jobs running on a CMS-specific LHC computing grid test bed. Monitoring information was transferred in real time from remote execution nodes back to the submitting host and stored in a database. In scalability tests, the job submission rates supported by successive releases of R-GMA improved significantly, approaching that expected in full-scale production.**

I. Monitoring Architecture

THE management of a large Monte Carlo (MC) production or data analysis, as well as the quality assurance of the results, requires careful monitoring and bookkeeping. The batch object submission system (BOSS) [1] has been developed as part of the compact muon solenoid (CMS) suite of software to provide real-time monitoring and bookkeeping of jobs submitted to a compute farm system. Its original design assumed that jobs were submitted to a local batch farm. Individual jobs to be submitted are wrapped in a BOSS executable which, when it executes, spawns a separate process that extracts information from the running job's input, output, and error streams. Pertinent information (such as status or events generated) for the particular job is stored, along with other relevant information from the submission system, in a local database. BOSS allows an end user to define new jobtypes in order to store specific sets of information about those jobs in addition to the standard job details.

In order for the BOSS database to monitor jobs reliably in a grid environment, jobs executing on a remote compute element need to pass information back to the submitter's site in real time. However the current implementation of BOSS is unsuitable for wide-area deployment since each wrapper instance makes a direct connection back to the database management system (DBMS). This introduces security and firewall issues for the database host as well as being unusable with grid implementations that do not allow outgoing connectivity from worker nodes, and the sheer number of connections required in the grid context can overwhelm the DBMS. Hence, BOSS has been modified to use the relational grid monitoring architecture (R-GMA) as a transport mechanism to deliver information from the remotely running job to the centralized BOSS database at the user interface (UI) of the grid system, from where the job was submitted.

The R-GMA architecture is based on that of the grid monitoring architecture (GMA) [2] of the Global Grid Forum. The GMA, as shown in Fig. 1, consists of three components: consumers, producers, and a directory service, which we refer to as a registry as it avoids any implied hierarchical structure.

Producers of information register themselves with the registry when they are instantiated. The registry describes the type and structure of information the producers want to make available to the grid. Consumers of information query the registry to discover what type of information is available and locate producers that provide the required information. Once a consumer has this information, it contacts the producer directly to obtain the relevant data. Thus, in principle, the "Register" and "Discover" transactions between the clients and the registry (the dotted lines in Fig. 1) only need to occur once, irrespective either of how much data is actually transferred (the solid line) or of the lifetime of the producer or consumer. R-GMA [3] is a specific implementation of this general GMA architecture using Java servlet technology (Tomcat [4]), so that as well as the producer and consumer described above, producer and consumer servlets exist and communicate using HTTP or HTTPS.

Manuscript received November 14, 2003; revised April 20, 2004 and June 15, 2004. This work was supported in part by PPARC and in part by the European Union.

D. Bonacorsi and C. Grandi are with the Instituto Nazionale di Fisica Nucleare, Bologna, Italy (e-mail: Daniele.Bonacorsi@bo.infn.it; Claudio.Grandi@bo.infn.it).

D. Colling, B. MacEvoy, and H. Tallini are with the Department of Physics, Imperial College London, London, SW7 2BW, U.K. (e-mail: d.colling@imperial.ac.uk; b.macevoy@imperial.ac.uk; h.tallini@imperial.ac.uk).

S. M. Fisher and S. Traylen are with the Particle Physics Department, Rutherford Appleton Laboratory, Chilton, U.K. (e-mail: S.M.Fisher@rl.ac.uk; S.Traylen@rl.ac.uk).

L. Field was with the Particle Physics Department, Rutherford Appleton Laboratory, Chilton, U.K.

P. R. Hobson, P. Kyberd, and J. J. Nebrensky are with the Department of Electronic and Computer Engineering, Brunel University, Cleveland Road, Uxbridge, UB8 3PH, U.K. (e-mail: Peter.Hobson@brunel.ac.uk; Paul.Kyberd@brunel.ac.uk; J.Nebrensky@brunel.ac.uk).

P. R. Hobson, P. Kyberd, and J. J. Nebrensky are with the Department of Electronic and Computer Engineering, Brunel University, Cleveland Road, Uxbridge, UB8 3PH, U.K. (e-mail: Peter.Hobson@brunel.ac.uk; Paul.Kyberd@brunel.ac.uk; J.Nebrensky@brunel.ac.uk).

Digital Object Identifier 10.1109/TNS.2004.839094

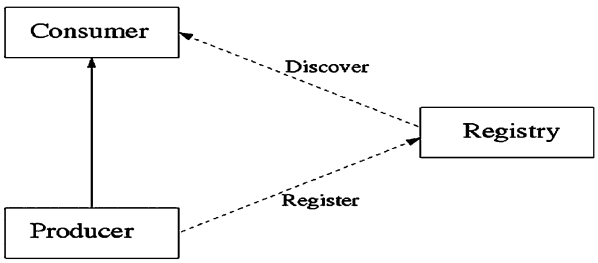

Fig. 1. GMA. The solid line shows the data transfer path, while the dotted lines show the "Register" and "Discover" transactions.

In our tests, R-GMA and BOSS were used in the following way (numbers in braces refer to entities in Fig. 2). At the UI, a receiver $\{3\}$ is created. This receiver sends its registration details via a locally running servlet $\{5b\}$ to the registry $\{6\}$. The registry stores details of the receiver (i.e., that it wishes to consume messages from a BOSS wrapper and the hostname of the DBMS) and sends back a list of any available matching producers. A job is submitted using the grid infrastructure—details of which are in principle irrelevant—from a UI $\{1\}$ and eventually arrives on a worker node (WN) $\{4\}$ at a remote compute element. When the job runs, the BOSS wrapper first creates a producer that sends its details, via a servlet $\{5a\}$ at that remote farm, to the registry $\{6\}$ which records details about the producer, including a description of the data but not the data itself. This description includes the output messages from the BOSS wrapper and the hostname of the DBMS at the submitting UI. The registry is thus able to notify the receiver $\{3\}$ of the new producer. The receiver then contacts the new producer directly and initiates data transfer, storing the information in the BOSS database $\{2\}$. As the job runs and monitoring data on the job are generated, the producer sends data into a buffer within the farm servlet, which, in turn, streams it to the receiver servlet.

Using BOSS and R-GMA in this way is a flexible and robust method for retrieving real-time job monitoring information. The use of standard Web protocols for data transfer allows straightforward operation through site firewalls and networks. Moreover, with only a single local connection required from the consumer to the BOSS database (rather than from a potentially large number of remote grid compute sites) this is a more secure mechanism for storing data.

In production deployment, the intention is that the receiver $\{3\}$ would be a specialized Java application based on the standard R-GMA consumer, storing data in a MySQL database [5]. The dashed arrows from the WN $\{4\}$ back to the UI $\{1\}$ in Fig. 2 indicate the BOSS journal file containing all messages sent, which is returned via the sandbox (a controlled environment in which potentially hazardous code can be safely run) after the job has finished and can thus be used to ensure the integrity of the BOSS DB (but not, of course, for online monitoring).

II. Test Setup and Initial Results

The monitoring architecture was initially tested on the European data grid (EDG) test bed [6]. An R-GMA producer servlet (v. 2.2.4) was installed on a single compute element at Imperial College, London, U.K., a consumer servlet was installed on the Imperial College UI, and a registry was installed at Brunel University. The BOSS database was installed on the Imperial College UI. Small numbers of CMS MC jobs were submitted from the UI to the EDG grid with the requirement for the resource broker to send the job to the Imperial College farm so that the appropriate R-GMA producer software would be found. It should be emphasized that standard CMS MC production software was used in submitting these jobs, and as such this represented a "real world" application test of R-GMA.

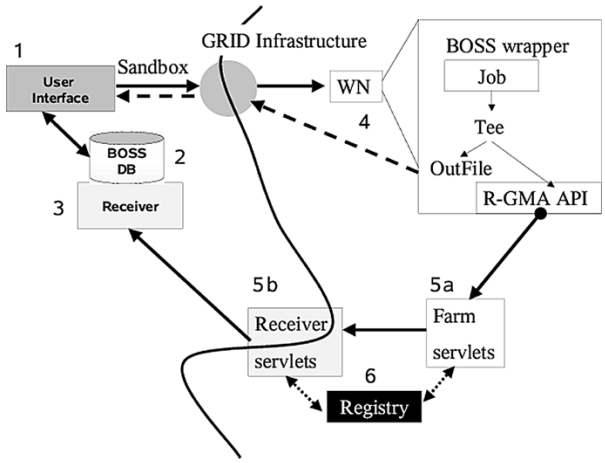

Fig. 2. Use of R-GMA in BOSS. Components labeled 3 and 5b form the R-GMA consumer while those labeled 4 and 5a are the producer. Components which are local to the submitting site lie to the left of the dividing curve, while those to the right are accessed via the grid infrastructure. Receiver servlets may be local to the UI or at other sites on the grid.

In this initial test, the MC jobs were configured to generate only a small number of events, such that they took about 10 min to complete execution. For each job, approximately 30 separate messages were required to be sent back to the BOSS database, via R-GMA. A comparison could then be made between the information in the database and the BOSS journal file to verify that no information had been lost or corrupted.

In initial tests, submitting small bunches of jobs such that no more than five jobs were running simultaneously, the monitoring system was shown to work successfully with all information being successfully relayed and stored in the BOSS database. When the submission rate was increased, however, information was lost. This was thought to be due to the use of a (now deprecated) producer that had a limited buffer size such that, at high transfer rates, information was overwritten before the consumer was able to retrieve it.

III. Test Description

R-GMA (version 3.3.28) was installed on machines (typically dual 1-GHz PIII with 1 GB of RAM) on a CMS-specific LHC computing grid [7] test bed. This provided a more appropriate producer and thus a more demanding test was required to provide a realistic stress on the system. It was not possible to dedicate all of the test bed machines to a test of the monitoring architecture, nor is it a suitable use of resources to involve software whose ability to perform is completely unknown in a data challenge. Finally, a test of the real system is limited to the performance provided by the current system; it is not possible to test

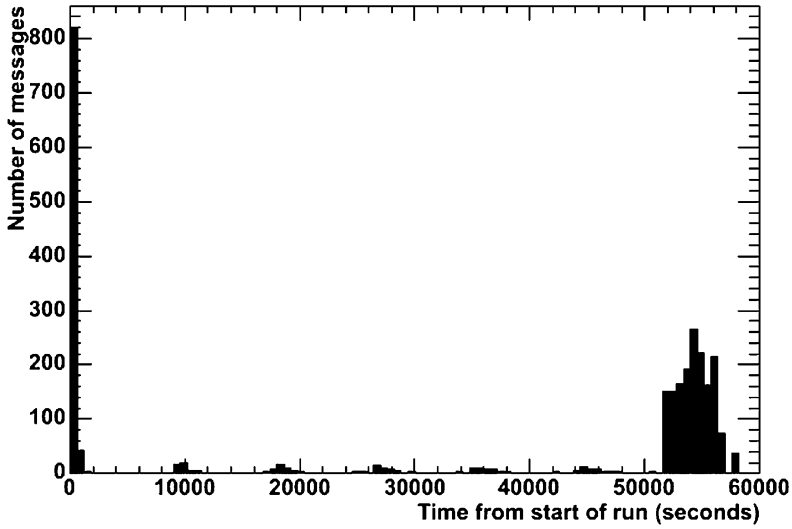

Fig. 3. Cumulative distribution of message arrival time for 44 CMSIM jobs.

the performance of a future, upgraded system. It was, therefore, decided to create a simulation of the production system, specifically of the output from the "CMSIM" component of the CMS Monte Carlo computation suite.

In the first instance, around 50 jobs were individually run. The times of their messages were recorded. From this information a message profile of a typical CMSIM job was created. The messages have a mean length of 35 characters.

It can be seen from Fig. 3 that the distribution of messages from jobs has three distinct phases. During the first second of the job, messages are sent equally spaced at a rate of around one every 50 ms and a single message occurs some 890 s later with an uncertainty of 170 s. This ends the first phase of the job. During the main phase of the job messages are generated about every 8800 s, with a Gaussian width of 480 s. In the final phase, as the job performs various housekeeping operations, 40 messages are generated in bursts over 100 s, including a burst of approximately ten messages in the final second that signals the termination of the job. With the distribution of completion times stretching over a period of 5000 s, *even* for jobs that start simultaneously, the probability of two jobs terminating together is less than 0.1%.

A test class using the R-GMA streamproducer [4] was written in Java, which created and sent messages at the measured rates. An instance of such a class is a realistic model of the performance of a single CMS Monte Carlo job as far as the R-GMA system is concerned. Since no processing is required between messages, a single CPU is capable of providing a load equivalent to many CPU's running the full job.

A single machine can generate messages at a rate of greater than fifty per second; thus it can provide a load greater than a typical grid cluster of 200 machines, up to a rate of 4/s per node.

Job initialization generates a similar number of messages and here the limiting factor is how fast a submission script is capable of starting jobs on the grid. Preliminary measurements indicate that the time for jobs to propagate through the workload management system and start executing varies between 3 and 10 s. Thus, the initial message bursts of submitted jobs will not overlap. Since we are interested in keeping a record of all messages that get through, rather than just a cumulative update as provided by BOSS, we also replaced the dedicated BOSS receiver with a consumer that uses the R-GMA Archiver [4] to store all received messages in a MySQL database. This makes it possible to detect the loss of messages even when they are superseded by a later, successfully transmitted, message.

We, thus, have a system that allows a single machine to generate an R-GMA load that is equivalent to several hundred machines at a single site. Adding a local machine to run the R-GMA servlets and with two machines at a site (representing the UI and the farm) we can simulate the full load on R-GMA. To complete the test system, we placed a registry at the Rutherford Appleton Laboratory and the consumer at Imperial College. Tests so far completed have involved running such a system at up to four geographically dispersed sites.

IV. Scalability Test Results

The greatest load on the monitoring system occurs during job initialization, as this is the period when messages are being passed with the greatest frequency, and so the number of simultaneously running jobs that can be handled depends on the interval between the different jobs starting. In the extreme case of all the jobs starting (essentially) simultaneously release 3.3.45 of RGMA can handle the equivalent of 400 CMSIM jobs with no message loss. However, introducing a delay of as little as 1 s between job starts more than doubles this. This staggered start provides a much more realistic simulation of an actual CMSIM production. Four hundred jobs represents approximately 20% of the predicted CMS production load.

When 500 jobs are started together, only about 440 transfer data successfully. When failure occurs, typically all the messages are lost from one or more jobs; this is generally because of a catastrophic failure of the StreamProducer servlet. Together with the developers of R-GMA, we have identified various problems including: initial use of under-powered machines (e.g., 733-MHz PII with 256 MB RAM) running servlets within the R-GMA infrastructure; the Java Virtual Machine (JVM) instance used by the servlets only having the Tomcat

default memory allocation available; the JVM instance used by the producer servlets requiring more than the default number (1024) of network sockets available; as well as other limits and flaws in the versions of R-GMA used. The developers of R-GMA have addressed these issues in newer releases. We have installed more powerful hardware (all machines with 1-GB RAM) and a new version of R-GMA (v. 3.4.13) with optimally configured JVM instances. We repeated the tests described in Section III and successfully received all the data from more than 4000 simulated jobs at multiple sites, a level of performance consistent with the needs of CMS. A report of these results is currently in preparation.

V. Summary

We have created a system to simulate the loads that R-GMA will experience during its use as a monitoring tool for the CMS data challenges on the EDG test bed. The original installation of R-GMA was unable to sustain the expected loads, failing at just over 400 jobs. By upgrading to more powerful machines running the servlets, improvements to R-GMA, and a more optimized configuration of the JVM, we have recently achieved a level of performance better than this by a factor of 10. This improved level of performance should meet the requirements of the running CMS experiment.

References

[1] C. Grandi and A. Renzi. Object based system for batch job submission and monitoring (BOSS). [Online]. Available: http://www.infn.it/cms/computing/BOSS
[2] B. Tierney *et al.*. (2002) A grid monitoring architecture. [Online]. Available: http://www.gridforum.org/
[3] A. Cooke *et al.*, "R-GMA: An information integration system for grid monitoring," in *Proc. 11th Int. Conf. Cooperative Information Systems*, 2003.
[4] [Online]. Available: http://jakarta.apache.org/tomcat/.
[5] [Online]. Available: http://www.mysql.com/.
[6] [Online]. Available: http://eu-datagrid.web.cern.ch/eu-datagrid/.
[7] [Online]. Available: http://lcg.web.cern.ch/LCG/.