Performance of R-GMA Based Grid Job Monitoring System for CMS Data Production

R. Byrom, D. Colling, S. M. Fisher, C. Grandi, P. R. Hobson, P. 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 CERN laboratory in Geneva, have large-scale data processing requirements, with stored data accumulating at a rate of 1 Gbyte/s. This load comfortably exceeds any previous processing requirements and we believe it may be most efficiently satisfied through Grid computing. Management of large Monte Carlo productions (~3000 jobs) or data analyses and the quality assurance of the results requires careful monitoring and bookkeeping, and an important requirement when using the Grid is the ability to monitor transparently the large number of jobs that are being executed simultaneously at multiple remote sites. R-GMA is a monitoring and information management service for distributed resources based on the Grid Monitoring Architecture of the Global Grid Forum. We have previously developed a system allowing us to test its performance under a heavy load while using few real Grid resources. We present the latest results on this system and compare them with the data collected while running actual CMS simulation jobs on the LCG2 Grid test bed.**

I. Introduction

HIGH Energy Physics experiments, such as the Compact Muon Solenoid (CMS) at the CERN laboratory in Geneva, have large-scale data processing requirements, with data accumulating at a rate of 1 GB s^{-1}. This load comfortably exceeds any previous processing requirements and we believe it may be most efficiently satisfied through Grid computing. Furthermore the production of large quantities of Monte Carlo simulated data provides an ideal test-bed for Grid technologies and will drive their development.

One important challenge when using the Grid for data analysis is the ability to monitor transparently the large number of jobs that are being executed simultaneously at multiple remote sites. BOSS (Batch Object Submission System) [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. Originally designed for use with a local batch queue, BOSS has been modified to use the Relational Grid Monitoring Architecture (R-GMA) as a transport mechanism to deliver information from a remotely running job to the centralized BOSS database at the User Interface (UI) of the Grid system, from whence the job was submitted. R-GMA [2] is a monitoring and information management service for distributed resources based on the Grid Monitoring Architecture of the Global Grid Forum.

We have previously reported on a system allowing us to test performance under heavy load whilst using few real Grid resources [3]. This was achieved using lightweight Java processes that simulate the content and timing of the messages produced by running CMS Monte Carlo simulation (CMSIM) jobs, but don't actually carry out any computation. Many such processes can be run on a single machine, allowing a small number of worker nodes to generate monitoring data equivalent to that produced by a large farm.

In this paper we discuss the final results from the scalability tests mentioned above, and describe our initial experiences when using R-GMA deployed on a real, production Grid (LCG2) [4].

II. Use of R-GMA in BOSS

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. BOSS 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. Individual jobs to be run 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

Manuscript received 27th October 2004.This work was supported in part by PPARC and by the European Union.

R. Byrom, S. M. Fisher and S. Traylen are with the Particle Physics Department, Rutherford Appleton Laboratory, Chilton, UK (*e*-mail R.Byrom@rl.ac.uk, S.M.Fisher@rl.ac.uk, S.Traylen@rl.ac.uk).

D. Colling, and B. MacEvoy are with the Department of Physics, Imperial College London, London, SW7 2BW, UK (*e*-mail d.colling@imperial.ac.uk, b.macevoy@imperial.ac.uk).

C. Grandi is with the Instituto Nazionale di Fisica Nucleare, Bologna, Italy (*e*-mail Claudio.Grandi@bo.infn.it).

P. R. Hobson, P. Kyberd and J. J. Nebrensky are with the School of Engineering and Design, Brunel University, Uxbridge, UB8 3PH, UK. (*e*-mail: Peter.Hobson@brunel.ac.uk, Paul.Kyberd@brunel.ac.uk, henry.nebrensky@physics.org).

H. Tallini was with the Department of Physics, Imperial College London, London, UK when this work was being done.

information from the submission system, in a database within a local DBMS (currently MySQL [5]).

Direct transfer of data from Worker Nodes (WN) back to the UI has some problems in a Grid context:

- the large number of simultaneous connections into the DBMS can cause problems – within CMS the aim is to monitor at least 3000 simultaneously running jobs;
- as the WNs are globally distributed, the DBMS must allow connections from anywhere. This introduces security risks both from its exposure outside any site firewall and from the simplistic nature of native connection protocols;
- similarly, the WNs must be able to connect to a DBMS located anywhere – but there is still debate over the nature and scope of the network connectivity that they should make available.

We are therefore evaluating the use of R-GMA as the means for moving data around during on-line job monitoring. R-GMA is a monitoring and information management service for distributed resources based on the Grid Monitoring Architecture (GMA) of the Global Grid Forum and originally developed within the EU DataGrid project [6]. As it has been described elsewhere ([2], [3], [7]) we discuss only the salient points here.

The GMA uses a model with producers and consumers of information, which subscribe to a registry that acts as a matchmaker and identifies the relevant producers to each consumer. The consumer then retrieves the data directly from the producer; user data itself does not flow through the registry.

R-GMA is an implementation of the GMA in which the producers, consumers and registry are Java servlets (Tomcat, [8]). R-GMA is **not** a general, distributed RDBMS system but a way to use the relational model in a distributed environment; that is, producers

- announce: SQL "`CREATE TABLE`"
- publish: SQL "`INSERT`"

while consumers

- collect: SQL "`SELECT`"

Fig. 1 shows how R-GMA has been integrated into BOSS (numbers in braces refer to entities in the figure). The BOSS DB {2} at the UI has an associated "receiver" {3} that registers – via a locally running servlet {5b} – with 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). 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 an R-GMA StreamProducer 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 that the output is BOSS wrapper messages 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.

Within LCG a servlet host {5a, 5b} is referred to as a "MON box", while the registry {6} is denoted an "Information Catalogue".

Each running job thus has a Producer that gives the host and name of its "home" BOSS DB and its BOSS jobId; this identifies the producer uniquely. The wrapper, written in C++, publishes each message into R-GMA as a separate tuple – equivalent to a separate "row".

The BOSS receiver, implemented in Java, uses an R-GMA consumer to retrieve all messages relating to its DB and then uses the jobId and jobType values to do an SQL `UPDATE`, by JDBC, of the requisite cell within the BOSS DB.

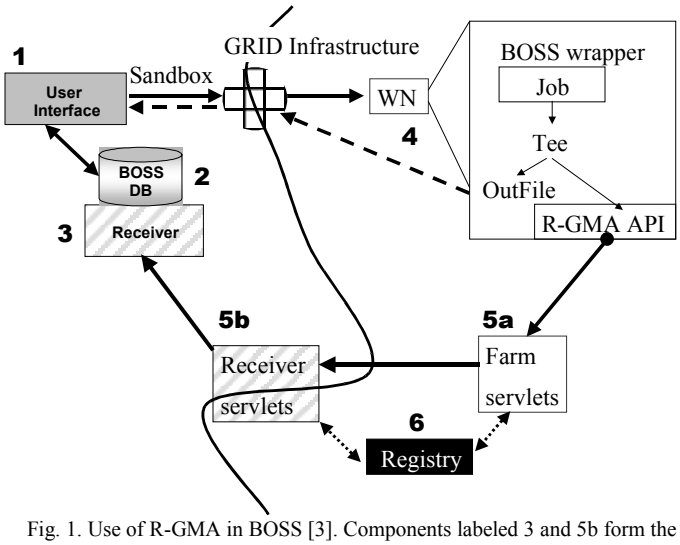

Fig. 1. Use of R-GMA in BOSS [3]. Components labeled 3 and 5b form the R-GMA consumer while those labeled 4 and 5a are the producer. Components which are local to the submitting site lie to left of the dividing curve, while those to the right are accessed (and managed) by the Grid Infrastructure. Receiver servlets may be local to the UI or at other sites on the Grid.

The use of standard Web protocols (HTTP, HTTPS) for data transfer allows straightforward operation through site firewalls and networks, and only the servlet hosts / MON boxes actually need any off-site connectivity. 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.

Using R-GMA as the data transport layer also opens new possibilities as not only can a consumer can watch many producers, but also a producer can feed multiple consumers. R-GMA also provides uniform access to other classes of monitoring data (network, accounting...) of potential interest.

Although it is possible to define a minimum retention period, for which published tuples remain available from a producer, R-GMA ultimately provides no guarantees of message delivery. The dashed arrows from the WN {4} back to the UI {1} in Fig. 1 indicate the BOSS journal file containing all messages sent, which is returned via the Grid sandbox mechanism after the job has finished and can thus be used to ensure the integrity of the BOSS DB (but not, of course, for on-line monitoring).

III. Scalability Testing

Before use within CMS production it is necessary to ensure R-GMA can cope with the expected volume of traffic and is scalable. The CMS MC production load is estimated at around 3000 simultaneous jobs, each lasting about 10 CPU hours.

Possible limits to R-GMA performance may include the total message flux overwhelming a servlet host; a farm servlet host running out of resources to handle large numbers of producers; or the registry being overwhelmed when registering new producers, say when a large farm comes on line.

To avoid having to dedicate production-scale resources for testing, it was decided to create a simulation of the production system, specifically of the output from the "CMSIM" component of the CMS Monte Carlo computation suite. A Java MC Simulation represents a typical CMS job: it emulates the CMSIM message-publishing pattern, but with the possibility of compressing the 10-hour run time. For simulation, CMSIM output can be represented by 5 phases:

1. initialization: a message every 50 ms for 1 s
2. a 15 min pause followed by a single message
3. main phase: 6 messages at 2.5 hour intervals
4. final: 30 messages in bursts, over 99 s
5. 10 messages in the last second

(for more details of intervals and variability see [3]). The MC Sim also includes the BOSS wrapper housekeeping messages (4 at start and 3 at end) for a total of 74 messages.

Obviously, there is no need to do the actual number crunching in between the messages, so one MC Sim can have multiple threads ("simjobs") each representing a separate CMSIM job – thus a small number of Grid jobs can put a large, realistic load on to R-GMA.

In order to analyse the results, an R-GMA Archiver and HistoryProducer are used to store tuples which have been successfully published and received. The HistoryProducer's DB is a representation of the BOSS DB, but it stores a history of received messages rather than just a cumulative update – thus it is possible to compare received with published tuples to verify the test outcome. The topology of our scalability testing scheme is shown in fig. 2.

In essence our procedure is to submit batches of simjobs to the Grid, and see

- if messages get back
- how many come back

By changing the number of MC Sims used and where they are run, we can focus stress on different links of the chain.

It should be noted that the results don't apply just to BOSS – any monitoring framework would have to transfer the same amount of data from the jobs back to the UI.

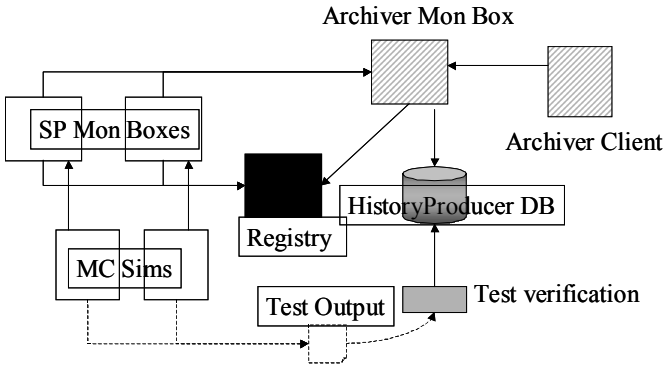

Fig. 2. Topology of scalability tests (shading as fig. 1).

For the first series of scalability tests the simjobs were compressed to only run for about a minute (the message-publishing pattern thus being somewhat irrelevant).

Initial tests, with R-GMA v. 3.3.28 on a CMS testbed (registry at Brunel University), only managed to monitor successfully about 400 simjobs [3]. Various problems were identified, including:

- various configuration problems at both sites (Brunel University and Imperial College) taking part in the tests, including an under-powered machine (733 MHz PII with 256 megabytes RAM) running servlets within the R-GMA infrastructure in spite of apparently having been removed from it
- limitations of the initial R-GMA configuration: for example, many "OutOfMemory" errors as the servlets only had the Tomcat default memory allocation available; or the JVM instance used by the Producer servlets requiring more than the default number (1024) of network sockets available
- other limits and flaws in the versions of R-GMA used.

We have since installed more powerful hardware (all machines with 1 GB RAM) and an updated version of R-GMA (v. 3.4.13) with optimally configured JVM instances. We have repeated the tests and successfully received all the data from 6000 simjobs across multiple sites, a level of performance consistent with the needs of CMS.

The developers of R-GMA have addressed these issues in newer releases, and indeed a modified version of these CMS tests now forms part of R-GMA performance test suite – providing feedback into R-GMA development. On the EDG WP3 test-bed (based at RAL) using R-GMA v. 3.4.13, 1 MC Sim creating 2000 simjobs and publishing 7600 tuples

was proven to work without any glitch, and successful monitoring demonstrated for 2 MC Sims each running 4000 simjobs (with 15200 published tuples).

As the simjobs were so short and only a couple of WNs were needed, the MC Sims were run remotely through SSH rather than submitted through a job manager. We have found that for reliable operation new simjobs should not be started at a sustained rate greater than one every second. For these tests the simjobs were time compressed to last only 50 s; thus the number of simultaneously running simjobs was much lower than the real case, but since the whole test took less than the typical run time of a CMSIM job the message flux was actually higher.

IV. BOSS, R-GMA AND LCG2

We still need to confirm that R-GMA can handle the stress of job monitoring under "real-world" *deployment* and *operation* conditions. As it will be a major vehicle for the running of CMS software, the LCG is an obvious platform for such work. Although R-GMA is part of the LCG 2.2.0 release it is not a mandatory component, and many sites have not yet installed it for their WNs. Even if the R-GMA infrastructure is in place and working, it may still not be able to support CMS applications monitoring, either intrinsically, because CMS' needs are too demanding, or simply because of the load on R-GMA from other users. In the first section below we consider the availability of R-GMA to Grid users; the second discusses the testing of R-GMA in a shared environment.

A. Deployment

So far there have been significant problems getting R-GMA jobs to run at all due to the misconfiguration of a large proportion of sites: e.g. on 13th October 2004 only 13 of 24 matching resources were able to run a simple test job and send data back to Brunel University.

- At 3 sites messages were published to their MON box, but didn't reach the consumer (all have since confirmed this was a firewall issue).
- At 3 sites the MON boxes simply refused connections from the WNs.
- 3 sites were "falsely advertising" an R-GMA environment advertised even though it was not installed or configured.
- 2 sites aborted the job due to other Grid problems.
- Even from 3 of the "working" sites, a few of the messages went astray (exact reasons still to be determined).

Successful roll-out of a complex infrastructure spanning the globe is difficult: most sites are run not by Grid middleware developers but by system administrators, with major non-Grid responsibilities and little specific knowledge. Confusing, missing or incorrect documentation – in particular regarding the manual MON box installation – has caused major headaches. At present R-GMA deployment is not yet ready for users though it is very close – tests since already suggest better than 90% resource validity. Similar issues will, of course, have to be faced by any other job monitoring scheme.

B. Operation

We plan to use initially the CMSIM emulator from the scalability tests described above, but now with a runtime of ~30 min (the main phase accelerated by 100*x*). This not only avoids the need to reserve resources on a Data Challenge scale, but also means we can use sites that don't have the full CMS software environment available. The 30 minute run time is long enough that all simjobs at multiple sites can expected to be running at the same time, even when submitted through the normal LCG Workload Management System, while still allowing problems to be identified on-line rather than the morning after. Although CMSIM has recently been withdrawn by CMS, the information to be monitored from its successor, OSCAR, is essentially the same and so the change is not expected to affect the significance of the results; however it will eventually become necessary to rewrite the MC Sim to simulate OSCAR simply because it will no longer be possible to run real CMSIM jobs for comparison.

In a simultaneous submission of 50-simjob MC Sims to 4 manually specified sites, all 14800 messages transferred successfully. Although this is encouraging, such a test cannot simply be scaled up without considering the possible side-effects for other Grid users. The nature of the MC Sim is that it uses minimal computing resources – 1 CPU – and requires no storage. The main Grid components at risk are those of R-GMA itself: the "Information Catalogue" (or registry) and the MON boxes. Although the consequences of bringing down the registry are very serious, the chances of this happening are very low: the registry does not store the data itself but merely pointers to the producers and consumers that use it, the storage requirements for which are trivial compared to modern computer memories. The rate at which producers are registered/cleared may be more important, but in our application is unlikely to exceed 10 s^{-1} – even real jobs will be dealt with serially by the queuing system. The R-GMA registry has already survived performance testing with up to 3200 simultaneous producers and with more than 100 producer registrations s^{-1} [7].

Bringing a farm's MON box down would have much less impact on other users, but is still unfriendly. Establishing the safe traffic limits for a MON box requires understanding the load caused both by producers and by consumers. The latter can be done fairly safely by distributing a number of small-scale producers over several sites: the aggregate thus only stresses the MON box at the consumer. The former is probably best determined by installing LCG MON and WN packages on machines not part of the LCG, and testing directly.

V. Conclusions

We have carried out tests of the viability of a job monitoring solution for CMS data production that uses R-GMA as the transport layer for the existing BOSS tool. By using a lightweight solution to simulate the output from the application, we have shown that the performance of R-GMA is consistent with CMS' production requirements.

R-GMA is currently being rolled out on the LCG and an initial test has been encouraging with 14800 messages transferred without loss to four sites on the LCG2 Grid test-bed. In the immediate future we will repeat the tests with the simjobs running in a much more realistic fashion in an environment where other things are happening.

In the medium term we will submit simjobs with their normal time span (circa 10 h to complete) and compare them with the data collected while running actual CMS production jobs on the LCG2 Grid. From this we will be able to identify the bottlenecks in the system and calculate at what point they become important. We will assess the reliability of the system for a production cycle lasting several weeks and conclude with the implications for full CMS production.

VI. Acknowledgment

This work has been funded in part by PPARC (GridPP) and by the EU (EU DataGrid).

VII. References

[1] C. Grandi and A. Renzi, "Object Based System for Batch Job Submission and Monitoring (BOSS)", *CMS Note 2003/005*; [Online]. Available: http://www.infn.it/cms/computing/BOSS

[2] A. Cooke *et al.*, "R-GMA: an information integration system for Grid monitoring" in *Proceedings of the Eleventh International Conference on Cooperative Information Systems,* 2003.

[3] H. Tallini *et al.*, "Scalability tests of R-GMA based Grid job monitoring system for CMS Monte Carlo data production" presented at IEEE NSS 2003; IEEE Trans. Nucl. Sci. accepted for publication.

[4] [Online]. Available http://lcg.web.cern.ch/LCG/

[5] [Online]. Available http://www.mysql.com/

[6] [Online]. Available http://eu-datagrid.web.cern.ch/eu-datagrid/

[7] A.W. Cooke *et al.*, "The relational grid monitoring architecture: mediating information about the Grid" submitted to *Journal of Grid Computing*.

[8] [Online]. Available http://jakarta.apache.org/tomcat/