Performance of R-GMA for Monitoring Grid Jobs for CMS Data Production

R. Byrom, D. Colling, S. M. Fisher, C. Grandi, P. R. Hobson, P. Kyberd, B. MacEvoy, J. J. Nebrensky 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 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. 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. 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 running on the LCG 2 Grid test bed using the LCG 2.6.0 middleware release. For a sustained load equivalent to 7 generations of 1000 simultaneous jobs, R-GMA was able to transfer all published messages and store them in a database for 98% of the individual jobs. The failures experienced were at the remote sites, rather than at the archiver's MON box as had been expected.**

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 which 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 merely simulate the content and timing of the messages produced by running CMS Monte Carlo simulation (CMSIM) jobs without actually carrying 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.

Unlike most assessments of monitoring middleware, which use dedicated, isolated testbeds (e.g. [3], [10]), we here discuss our experiences when using R-GMA deployed on a real, production Grid (the LCG, v. 2.6.0) [4]. Although CMSIM has recently been withdrawn by CMS, the information needing to be monitored from its successor, OSCAR, is essentially identical and so the change is not expected to affect the significance of the results.

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 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

Manuscript received 11th November 2005.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).

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 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 Grid sites may refuse to make the required network connectivity 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. It was originally developed within the EU DataGrid project [6] and now forms part of the EU EGEE project's gLite middleware [7]. As it has been described elsewhere ([2], [3]), 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 ... WHERE`"

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.

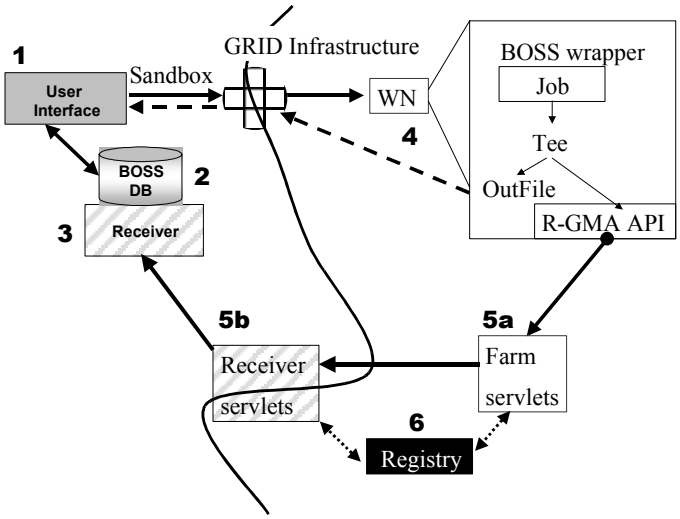

Fig. 1. Use of R-GMA in BOSS [3]. Components labeled 3 and 5b form the R-GMA consumer while those labeled 4 and 5a are the producer. Components which are local to the submitting site lie to left of the dividing curve, while those to the right are accessed (and managed) by the Grid Infrastructure. Receiver 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. Initial 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. The Java MC Sim code has been named *bossminj*.

In order to analyse the results, an R-GMA Archiver and HistoryProducer are used to store tuples that 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 and see

- if messages get back
- how many come back

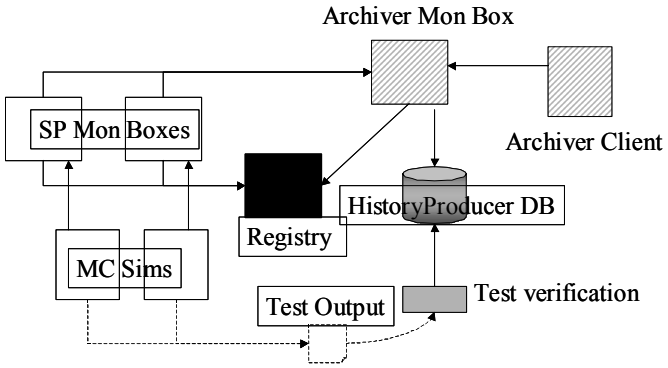

Fig. 2. Topology of scalability tests (shading as fig. 1).

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.

These tests were later repeated using 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. All the messages were successfully received from 6000 simjobs across multiple sites [9], a level of performance consistent with the needs of CMS.

As the simjobs were so short and only a couple of WNs were needed, the producers were run remotely through SSH rather than submitted through a job manager. We found that for reliable operation new simjobs should not be started at a sustained rate greater than one every second. For those 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. Job Monitoring on LCG 2.6.0

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. R-GMA is part of the LCG middleware; however, 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 essence our procedure is to submit batches of simjobs to the Grid (via a Resource Broker) and count the number of messages successfully transferred back to the database. This can be compared with the number of messages inserted into R-GMA, which is recorded in the output files returned via the Grid sandbox. By changing the number of MC Sims used and where they are run, we can focus stress on different links of the chain.

Each simjob was time-compressed by speeding up phase 3 by 100 times, for a run-time of just over 30 minutes. The MC Sims were limited to spawning 200-250 simjobs, in case several were sent to the same site. In initial testing we received every message from 1250 simjobs within a single MC producer at one site, but encountered problems with just 250 simjobs at another.

200-simjob MC producers were submitted to the Grid (LCG production zone) at ~5 minute intervals for a period of 6 hours. The only JDL requirements given were for LCG version (2.6.0) and for a sufficient queue wall-clock time – no site filtering or white-listing was used. If jobs were aborted or stuck in a queue, extra producers were submitted to try to have 1000 simjobs always active.

The archiver's MON box had an AMD Athlon XP 2600+ (model 10) CPU with 2 GB RAM and the LCG MON node software installed; this MON box was not shared with any other Grid resource. A second PC with an AMD Athlon XP 2600+ (model 10) CPU and 1.5 GB RAM hosted the MySQL DBMS used by the R-GMA HistoryProducer to store the received tuples, and also acted as the Grid User Interface. Both machines were running Scientific Linux (CERN) v. 3.0.5 [11] and the Sun Java SDK (v. 1.4.2_08) [12].

Overall 115 MC producers were submitted over the night of October 19th to 20th, of which 27 failed to start because R-GMA was not installed or working at the WN and 18 were aborted because of other middleware issues.

Another two MC producers were sent to one site where the MON box failed part-way through each, and at a further site the job was cut off in mid-publication with no sandbox returned to allow diagnosis.

Two of the successful MC producers had to wait in queues until long after all the others had finished.

Although 39% of the MC producers failed to start correctly, they only encountered problems at 13 out of the 45 Grid sites to which they were submitted. About half of those sites received and failed a series of Grid jobs, making the success rate by job much worse than that by site (the "black-hole effect"). While this is an improvement over our findings from one year previously, when 11 out of 24 sites failed to run an MC producer correctly [9], there clearly still remain a significant number of badly configured Grid sites that will have a disproportionately deleterious effect on LCG's user experience.

Of the 23000 simjobs submitted, 14000 (61%) ran at a remote site, of which 13683 (98%) transferred all of their messages into the database.

Every single one of the 1017052 individual messages logged as published into R-GMA was also transferred successfully. It thus appears that the failures were all associated with the remote sites' MON boxes, rather than problems with the archiver's MON box which was expected to be a bottleneck.

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.

An R-GMA archiver has been shown to receive all messages from a sustained load equivalent to over 1000 time-compressed CMSIM jobs spread across the Grid.

A single site MON box can handle over 1000 simultaneous local producers, but requires correct configuration and sufficient hardware (dedicated CPU with at least 1 GB RAM). Successful deployment of a complex infrastructure spanning the globe is difficult: most sites are run not by Grid developers but by sysadmins with major non-Grid responsibilities. Thus the testing of middleware solutions must include not only the intrinsic reliability of the software on some ideal testbed, but also the consequences of hardware and administrator limitations during installation and operation. We believe this highlights the importance of formal site functional testing to confirm that software is properly deployed and of providing users or RBs with a mechanism for white-listing, i.e. selecting only sites known to be properly configured for job execution.

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://boss.bo.infn.it/

[2] A.W. Cooke *et al.*, "The relational grid monitoring architecture: mediating information about the Grid" *Journal of Grid Computing* **2** (4) pp. 323-339 (2004)

[3] D. Bonacorsi *et al.*, "Scalability tests of R-GMA based Grid job monitoring system for CMS Monte Carlo data production" *IEEE Trans. Nucl. Sci.* **51** (6) pp. 3026-3029 (2004)

[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] [Online]. Available http://glite.web.cern.ch/gLite/

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

[9] R. Byrom *et al.*, "Performance of R-GMA Based Grid Job Monitoring System for CMS Data Production" *2004 IEEE Nuclear Science Symposium Conference Record* pp. 2033-2037 (2004)

[10] X.H. Zhang, J.L. Freschl and J.M. Schopf: "A Performance Study of Monitoring and Information Services for Distributed Systems" *12th IEEE International Symposium on High Performance Distributed Computing*, Seattle, USA pp. 270-282 (2003)

[11] [Online]. Available http://linux.web.cern.ch/linux/

[12] [Online]. Available http://java.sun.com/