Designing the Next Generation Grid Information System

Laurence Field
CERN, Geneva, Switzerland
E-mail: `Laurence.Field@cern.ch`

Paul Harvey
CERN, Geneva, Switzerland
E-mail: `harveyp@dcs.gla.ac.uk`

Tim Dyce
University of Melbourne, Vitoria, Australia
E-mail: `tjdyce@unimelb.edu.au`

Abstract. Grid information systems play a core role in today's production Grid infrastructures. They provide a coherent view of the Grid services in the infrastructure while addressing the performance, robustness and scalability issues that occur in dynamic, large-scale, distributed systems. As the number of services within these infrastructures continues to grow, it must be ensured that the Grid information system will meet the future scalability requirements. This paper discusses the various fundamental design changes that could be made to improve scalability, and presents preliminary results from the evaluation of various alternative technologies.

1. Introduction

Grid information systems enable users, applications and services to discover which Grid services exist in a Grid infrastructure and further information about their structure and state [**?**]. Information describing each Grid service is provided by the service itself. Hence, the Grid service, in terms of Grid computing, is the primary information source [5]. Grid information systems provide a mechanism for resolving queries that may need to take into consideration multiple information sources. These information sources are distributed to the same extent that Grid services are distributed. Given the large number of services in a typical Grid infrastructure, Grid information systems use methods to prefetch some of this information to improve performance. For this purpose, implementations of Grid information systems [4] employ caching [8] and/or quasi-copies [7] in order to address performance, robustness and scalability issues.

The Grid information system used in the Worldwide LHC Computing Grid (WLCG) [6] employs a three level caching model that is implemented using LDAP [9] databases. At the service-level, information about the service is periodically obtained from the service itself and stored in an LDAP database that resides alongside the service. At the site-level, information about all services provided by an institute are stored in an LDAP database located at that site and is populated by periodically querying the service-level databases. Finally, at the top-level, information about all services provided by all institutes are stored in an LDAP database, which is populated by periodically querying the site-level databases. The result is essentially a distributed LDAP model, where each lower level cache represents a branch. As of October 2010 the WLCG information system contained approximately 2000 service-level caches, 370 site-level caches and 100 top-level caches. A diagram that explains the WLCG topology can be seen in Figure 1.

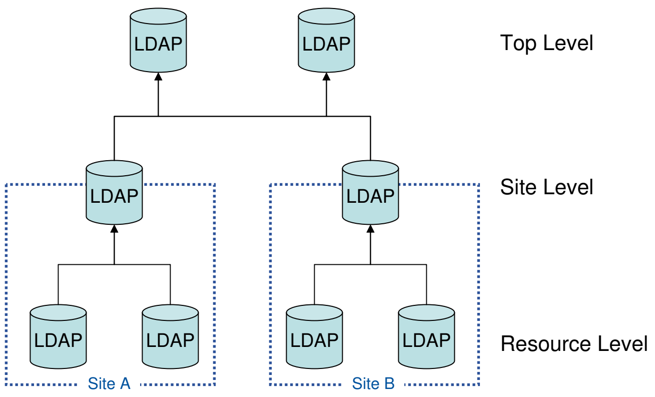

Figure 1. The topology of the WLCG information system.

The software used to maintain these caches, along with the caches themselves is known as the Berkeley Database Information Index (BDII) [2]. Essentially, this software obtains a snapshot by either querying the service or querying a lower-level cache. The snapshots obtained from the lower levels are merged and the result is compared against the current snapshot as stored in the LDAP database. New, deleted, or modified objects are grouped and the database is updated accordingly. The rational for using a differential update method is provided in a previous paper [3].

Although the differential update method only touches a subset of information in the cache, obtaining the snapshot for the comparison requires all the information to be transported over the network. Considering the transportation between the site- and top-level, 100MB of of data is being transported on every update cycle, which is 72GB per day per top level BDII instance. This unnecessarily consumes bandwidth and requires additional time to transfer the superfluous information, which increases the overall update time. Another issue is that the frequency at which the cache is updated does not necessarily match the frequency at which the information is changing. This mismatch can result in either information in the cache not being up-to-date or information being obtained unnecessarily. A recent paper [1] has shown that the rate at which the information is changing is not uniform across the different object types, in fact it is most dissimilar. This difference makes it difficult to chose an optimal value for periodically refreshing the cache. Any decision will result in either information being out-of-date or the unnecessary use of bandwidth. In the extreme case, where the transferring all information requires a significant amount of time, information in the cache could be out-of-date as soon as the cache has been updated.

In the current system, the difference between the snapshot and the cache in calculated at the higher-level. However, the lower-level would have already calculated the difference when it updates itself. The focus of this investigation is to understand if it is possible to transfer these changes rather than the compete set of information and the resulting affect on performance of the system.

This paper is outlined as follows. Section 2 describes the current state of the WLCG information system and gives a summary of the information provided along with the frequencies of change. Section 3 gives an overview of two new prototypes along with performance measurements. Section 4 compares these and gives performance measurements, with the theoretic limits along with thoughts on future enhancements. Some concluding remarks are given in Section 5.

2. Technology Choices

A previous investigation [1] showed that the frequency of change depends on the object type from the information model. For all object types, the number of new or deleted objects was less that 1% per day. However, for the number of modifications per day, there was more variation. For some object types, 80% of the objects experienced a change, while for other object types less than 0.1 % of them experienced a change. Looking at the changes in more detail, the object type which experienced 80% of changes over the day saw 50% of those objects being modified on each update cycle. These figures suggest that not all object types need to be updated at the same frequency.

Although the current version of the BDII already employs a differential update process, all information is currently transported over the network. This investigation focuses on two implementations where only that changes are transported, thus the bandwidth used and update time should be reduced.

2.1. Syncrepl

Syncrepl [11] is an overlay (plug-in) to the OpenLDAP [12] framework and is described in RFC 4533 [**?**]. The OpenLDAP database technology is used within the current information system implementation. The overlay itself provides synchronized replication between a number of different databases configurations, including 1xN, Nx1,and NxN. Replication is organized such that there is a master server to which additions, deletions of modifications are made. This server then replicates any operations to slave databases who have registered to the master. A master may have many slaves, a slave may have many masters and a master may be both a slave and a master, however operations may only be made to branches for which it is a master - i.e. the appropriate access permissions are required. Synchronization is achieved by the transportation of a Synchronization cookie (SyncCookie). This contains a Change Sequence Number (contextCSN), essentially a synchronization checkpoint (timestamp). For each entry an entry identifier (entryUUID) is sent with an empty record for no changes or a record describing the changes since the last checkpoint. Finally an entryUUID is sent for every entry which was deleted. As syncrepl offers all of the functionality desired for this work and is integrated with the existing technology, it seemed a natural choice.

2.2. Messaging

An alternative approach involved producing a new implementation of the transport mechanism, based on a messaging technology. A messaging system is one in which information is placed inside of a *message* by a client. The client then sends this message to a third party known as a broker. It is the responsibility of this broker to then forward this message to its recipient. The recipient may then do as they please with the contents of the message. The message itself may contain either text or binary data, and it is the decision of the users of such a system to decided what to place as the contents of the message. In order to coordinate the routing of messages, the broker offers two options: queues and topics.

Here the idea was similar to syncrepl, in that each operation (addition, deletion, modification) would produce an individual message to be passed on. The advantage that messaging technology has over syncrepl is that all messages are relayed by broker. This alleviates the load on the database machine and enables a publish subscribe mechanism to be used. In addition, it is decoupled from the underlying database technology, which could change in future.

3. Prototype Evaluation

In order to evaluate the two prototypes, it is required to replicate a real deployment scenario. To achieve this, information from the production system was used as input data. The top-level BDII at CERN was used as an information source. As information from each site is provided on a branch, it is trivial to split the information per site. However, doing this may or many not be necessary depending on the nature of the test. What is more important is that the changes between snapshots are captured. Again, this can be achieved by periodically querying the top-level BDII at CERN for all objects and using the identical algorithm as used by the BDII to calculate the difference between snapshots. In addition to evaluating the prototypes locally, the same test was carried out over the WAN. For this purpose an instance of the prototype hosted by the University of Melbourne in Australia was used as the CERN to Melbourne network path represents one of the more extreme use cases found in the production infrastructure.

3.1. Syncrepl

A slightly modified version of the BDII software version 5 was installed on a physical machine at CERN running Scientific Linux 5. The OpenLDAP version was updated to 2.4, as this version supports syncrepl. Syncrepl was enabled by modifying the slapd.conf file that is used by the BDII. The BDII was configured to use the top-level BDII at CERN as an information source.

An OpenLDAP database, version 2.4, was installed on a physical machine a CERN running Scientific Linux 5. It was configured to synchronize from the BDII using the syncrepl mechanism. This setup was replicated at the University of Melbourne. The complete setup is shown in Figure 2.

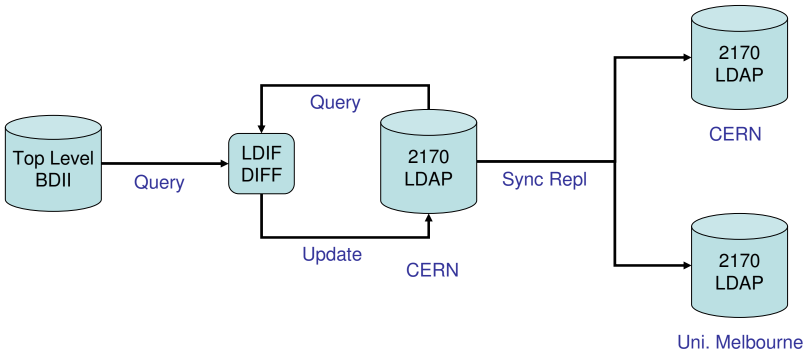

Figure 2. Syncrepl Testbed.

The content of the OpenLDAP databases were compared to the content of the top-level BDII. The local OpenLDAP database at CERN showed no sign of problems, the information was identical. However, the OpenLDAP database at the University of Melbourne periodically crashed with a segmentation fault, along with the BDII at CERN. The exact cause of this is as yet undetermined.

3.2. Messaging

A prototype of the BDII was produced that could be used with the ActiveMQ [10] messaging system. This resulted in two flavors of BDII: a producer BDII, which publishes changes to the messaging system and consumer BDII which retrieves these messages from the messaging system and updates the corresponding OpenLDAP database accordingly. The consumer BDII makes use of threads and is tolerant of ordering problems due to the inclusion of sequence numbers for each message. As messages are created on a per producer basis, no global sequencing is required. The mechanism makes use of topics within the message broker to ensure that messages are delivered to the correct address. The actual implementation was written in python, using the python implementation of the STOMP library. ActiveMQ was chosen, primarily due to existing experience, familiarity with the product and an already deployed broker network. The producer was installed on a physical machine a CERN running Scientific Linux 5 and configured to use the top-level BDII at CERN as the information source. The consumer was installed on a similar machine at CERN, with the setup being replicated at the University of Melbourne. The complete setup is shown in Figure 3.

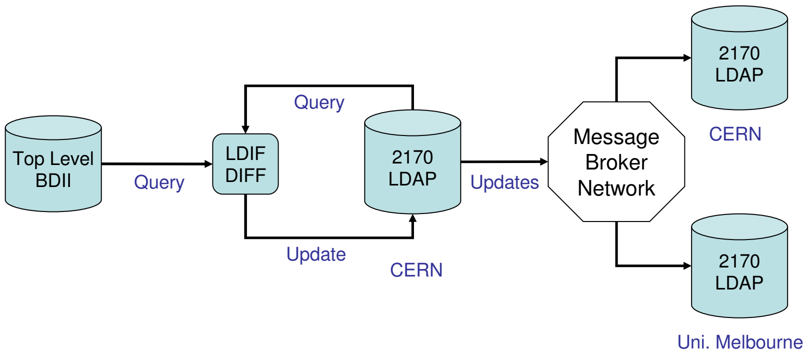

Figure 3. Messaging testbed.

This scenario was instrumented to verify the correctness of information, measure the amount of data transfered and measure the latency when processing messages. A graph showing the amount of data transported can be seen in Figure 4.

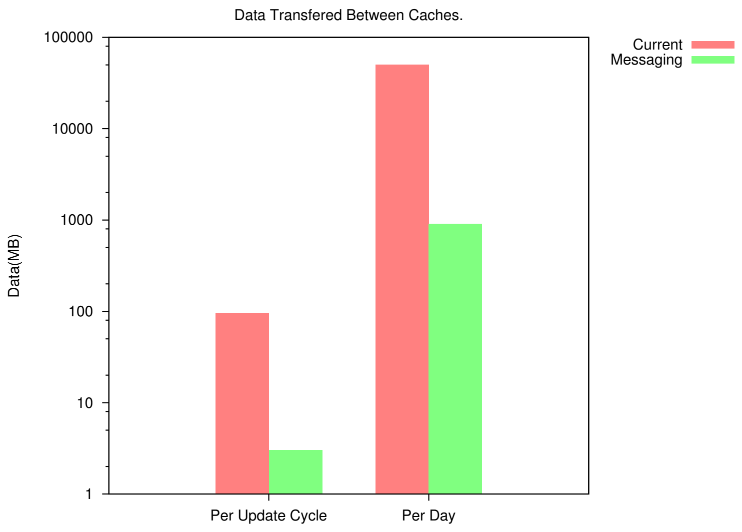

Figure 4. Amount of data transported by different mechanisms

Figure 4 shows that the use of the messaging system reduces the amount of data transferred per day by two orders of magnitude. The update time for the new method is defined as the time required from the beginning of an incoming message stream to the end of that message stream. Figure 5 shows a comparison of the update times per message stream for each transport mechanism, both for CERN and the University of Melborne.

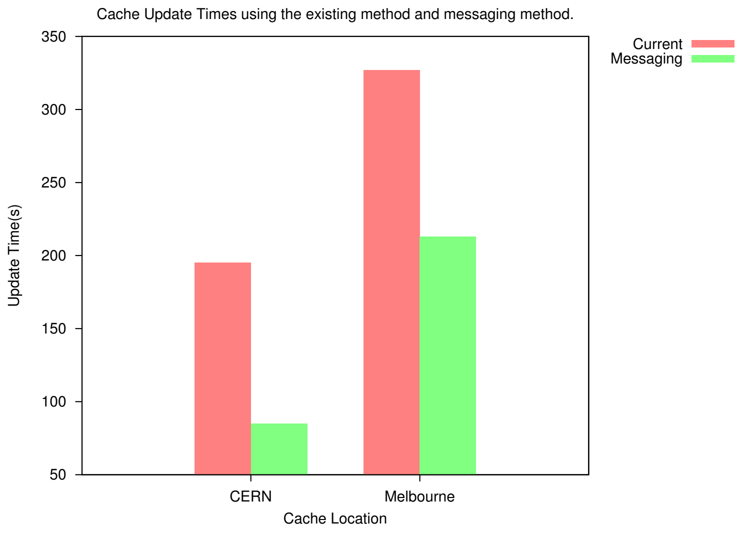

Figure 5. Average Data Transfer Times

It can be seen that the messaging system is faster in both the local (best case) and remote (worst case) situations when compared to the existing system. However, as the update time with messaging at CERN is approx 75% faster and the update time at Melbourne is only 50% faster, it suggests that network latency affects the performance.

4. Discussion of the Result

The failure of syncrepl during testing for the CERN to Melbourne University synchronization suggests that the technology is not ready to be used in production for such a scenario. Further investigation is required to understand the failure and to re-evaluate future versions of OpenLDAP. However the success locally suggests that it could be used to synchronize between the resource- and site-levels. This would not only reduce the latency of information at the site-level but also simplifies the deployment aspects.

The use of messaging in this evaluation was purely as a medium to transport information changes. This design decision proved to be successful in the sense that both the OpenLDAP database at CERN and the University of Melbourne were consistent with top-level BDII. In addition, this setup proved to be reliable for the duration of the evaluation (3 days). Transporting only the changes rather than the complete information resulted in a significant decrease in the amount of data transported over the network and the update time was also reduced. The need for the messaging infrastructure may result in this mechanism being too complex to be used for the resource- to site-level synchronization as an additional service would have to be provided.

In the prototype a synchronisation method is invoked in four scenarios: an existing consumer detects a new message from an unknown provider, a new consumer comes to life, a known provider is restarted, an existing consumer does not receive a subsequent message from the provider within a predefined time. This results in the consumer deleting the complete subtree in the database for that provider and request the full information for that branch. The synchronization method is critical to ensure smooth operation in a production deployment scenario and requires further investigation before it can be considered ready for production.

The prototype makes use of queues in the messaging infrastructure. The use of queues, the size of the message and the number of messages are already related and these parameters need to be understood in more detail to ensure that the optimal choices are made.

5. Conclusion

In this investigation, the use of syncrepl and ActiveMQ were both evaluated as replacement mechanisms for the transportation changes between the site- and top-level databases. It was found that the use of syncrepl over the WAN, CERN to Melbourne University, was not robust and would fail after a period of time. The use of the ActiveMQ messaging system proved to be reliable for the duration of the test. Transporting only the changes using the messaging system reduced the amount of information transported by two orders of magnitude. The update time was also reduced by 56.4% for the LAN deployment scenario and 34.8% for the WAN deployment scenario.

The simplicity of syncrepl deployment and that it was successful while working over the LAN, suggests that it would be a good candidate for transporting changes from the resource- to site-level. The use of messaging, while more complicated to setup as it requires the deployment of a messaging infrastructure, is effective over the WAN and hence would be a good candidate for transporting changes from the site- to top-level.

When transporting only changes, ensuring consistency becomes more important and as such a robust synchronization mechanism is required. Although this was not required for the evaluation and no inconsistency was observed, experience of production Grids suggests that inconsistencies will occur due to the unexpected behavior of a globally distributed infrastructure that spans multiple administrative domains. Before such a mechanism can be considered production ready, a robust synchronization method must be in place.

The use of compression either for change messages or the complete information could bring improvements and would be and area for further investigation.

In general, the use of messaging to transport change messages from the site- to top-level, does seem to be a direction for the future, however, more work is required before such a system could be ready for production deployment.

References

[1] Laurence Field and Rizos Sakellariou. How dynamic is the grid? towards a quality metric for grid information systems. In *11th ACM/IEEE International Conference on Grid Computing.*

[2] Laurence Field and Markus W. Schulz. Grid deployment experiences: The path to a production quality LDAP based grid information system. In *In the Proceedings of 2004 Conference for Computing in High-Energy and Nuclear Physics*, Interlaken, Switzerland, 2004.

[3] Laurence Field and Markus W Schulz. An investigation into the mutability of information in production grid information systems. *Journal of Physics: Conference Series*, 219(6):062046, 2010.

[4] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A directory service for configuring high-performance distributed computations. In *Proceedings. The Sixth IEEE International Symposium on High Performance Distributed Computing (Cat. No.97TB100183)*, pages 365–375, Portland, OR, USA.

[5] I. Foster, C. Kesselman, J.M. Nick, and S. Tuecke. Grid services for distributed system integration. *Computer*, 35(6):37–46, 2002.

[6] M Lamanna. The LHC computing grid project at CERN. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 534(1-2):1–6, 2004.

[7] Joachim Schmidt. Quasi-copies: Efficient data sharing for information retrieval systems. In *Advances in database technology*, volume 303, Venice, Italy, 1988.

[8] F. Stamatelopoulos and B. Maglaris. A caching model for efficient distributed network and systems management. In *Proceedings Third IEEE Symposium on Computers and Communications. ISCC'98. (Cat. No.98EX166)*, pages 226–230, Athens, Greece.

[9] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3), 1997.

[10] ActiveMQ. http://activemq.apache.org/.

[11] OpenLDAP 2.2 administrator's guide: LDAP sync replication. http://www.openldap.org/doc/admin22/syncrepl.html.
[12] OpenLDAP, url = http://www.openldap.org/, howpublished = http://www.openldap.org/.