An Evaluation of Information Consistency in Grid Information Systems

Laurence Field · Rizos Sakellariou

Received: 27 February 2016 / Accepted: 15 July 2016 / Published online: 25 July 2016

Abstract A Grid information system resolves queries that may need to consider all information sources (Grid services), which are widely distributed geographically, in order to enable efficient Grid functions that may utilise multiple cooperating services. Fundamentally this can be achieved by either moving the query to the data (*query shipping*) or moving the data to the query (*data shipping*). Existing Grid information system implementations have adopted one of the two approaches. This paper explores the two approaches in further detail by evaluating them to the best possible extent with respect to Grid information system benchmarking metrics. A Grid information system that follows the data shipping approach based on the replication of information that aims to improve the currency for highly-mutable information is presented. An implementation of this, based on an Enterprise Messaging System, is evaluated using the benchmarking method and the consequence of the results for the design of Grid information systems is discussed.

Keywords Gird · Information systems · Query processing · Service discovery

L. Field (✉)
CERN, Geneva, Switzerland
e-mail: laurence.field@cern.ch

R. Sakellariou
The University of Manchester, Manchester, UK
e-mail: rizos@manchester.ac.uk

1 Introduction

A fundamental aspect of Grid computing is the need to obtain information about the structure and state of Grid services which are widely distributed geographically. Information describing each Grid service is provided by the service itself and hence the Grid service is the primary information source. The information provided conforms to an information model which represents the main concepts of a Grid service and the relationships between them in order to clearly define their semantics. It is assumed that the contents of the information source are up-to-date, that is the values represent the real state of the Grid service at the source.

Queries need to be resolved that may consider some or all information sources in order to enable efficient Grid functions that may utilise multiple cooperating services. In general, Grid information is extremely distributed, i.e. each tuple (object instance) is found at a different location [8]. The queries are not complex in terms of structure, no more than one predicate and no joins, and hence the result is typically a subset of the total information (e.g. find the closest computing service to a storage service). The goal is to minimise the query response time in order to execute many queries, from many clients, for many information sources (i.e. to address a scalability problem).

In the absence of a central information system, the burden of information retrieval (i.e. discovering the information sources) and data retrieval (i.e. executing the query) falls on the client initiating the query. A *Grid information system* [17] aims to remove the burden of information retrieval and data retrieval from the client by providing a central point, an information service, to which a query can be sent and the result is returned. The provision of an information service transfers the burden from the client to the information service; however, the information sources still need to be discovered and the query still needs to be executed.

In terms of querying the information sources, the question is whether to move the query to the data (*query shipping*) or to move the data to the query (*data shipping*) [16]. This paper explores this question in further detail by evaluating to the best possible extent the different approaches using the benchmarking metrics described in [14]. In particular, the (α, β)-currency metric is used along with real data from a production Grid infrastructure, the Worldwide LHC Computing Grid, to identify the approach that provides the best *information consistency* taking into account the scalability challenges of a large-scale Grid. In this context, information consistency means that on query resolution, the result should accurately reflect the information source (in other words, it should be consistent) even if a time difference exists between when that information was extracted from the information source and when the query was returned to the client. Additionally, a novel approach for a Grid information system based on a publish-subscribe model is adopted to send changes as soon as they occur at the information source.

This paper is structured as follows. Section 2 describes the metrics that will be measured and discusses related results from the literature where available. A detailed evaluation of the main approaches using the benchmarking method in [14] is provided in Section 3 along with their suitability. The consequences of the results for the design of Grid information systems are discussed in Section 4 along with some concluding remarks in Section 5.

2 Querying Grid Information Systems

2.1 Background

The challenges of global information systems were outlined a long time ago, in a paper [18] which argued that scalability, autonomy, and the (un)availability of information sources would make brute force selection techniques, exhaustive searches, and global synchronisation (consistency) impossible for a large number of queries due to the long latencies and required throughput. It was also pointed out that unavailability is also an issue as scale grows, as the chance of failure in at least one component increases dramatically which means that at least some of information sources will be unavailable if there are many. Due to the challenge of global synchronisation, consistency requirements may need to be relaxed and in addition, as information sources can be unavailable, the results may be incomplete. Different trade-offs between all these issues may exist in different settings and environments.

In a typical architecture for query processing, information related to the location of information sources is stored in a *catalogue* [16]. In a Grid environment the catalogue is the service registry or index that provides the list of services along with their location in the form of a URL (the service record) [12]. Other attributes, such as the service type, can be used to avoid contacting services that may be irrelevant for the query, however for the global selection query, only the URL identifying the information source is required. It should be highlighted that information about which services should and should not be in the infrastructure does not originate from the service itself. The authoritative source for information about which services are participating in the infrastructure and to which administrative domain they belong is dependent on the management policies of the Grid infrastructure. The ownership and control of such information is of utmost importance in a global infrastructure and how this information is managed is a question of policy. Therefore, a prerequisite of a Grid information system is the existence of a catalogue in the form of a service registry which is a result of those management policies that can be used to discover all the information sources in a Grid infrastructure [12]. The details of how these policies are used to create the service registry is out-of-scope for this paper but it is an area that would benefit from further research.

After the information sources have been discovered, the query still needs to be executed. The question is whether to move the query to the information source (*query shipping*) or to move the information to the query (*data shipping*). Two approaches for data shipping may be used [16]; *caching* and *replication*. Caching is client-initiated and, in our context, is triggered by query execution; it typically uses a synchronous protocol that is based on invalidation [16], a *fault-in* approach. Replication is server-initiated and, in our context, is triggered by the information source itself and takes place even if no queries are processed; it aims to maintain the consistency of a *quasi-copy* [2] of the information using an asynchronous protocol. In general, replication tries to move the information close to a large group of clients so that they can access it cheaply the first time it is queried [20]. Caching enables a client to access the information cheaply when that information is used repeatedly [4]; the first query will experience a cost overhead while all other subsequent queries will benefit. In most cases, the cost overhead is much smaller than the potential benefit when there are many queries that can benefit.

Put simply, replication is required if there are more queries than updates (expansion test) and should be avoided if there are more updates than queries (contraction test)[22]. Data shipping scales well with the number of information sources, however, it can create very high communication costs if redundant data is shipped to the information service. The challenge is to anticipate what information needs to be replicated. As the number of queries increases, so does the probability that a specific query will occur in a given time period. This means that with enough queries in that period it is highly likely that a specific query will occur and the corresponding object or attribute will be requested, hence all information from all services needs to be transferred. In this scenario, with a sufficiently large number of queries, the caching approach would in any case ship all information from the information sources. Hence, the focus should be on efficiently replicating the information from the information source to the information service. In general, techniques for maintaining the consistency of a quasi-copy of the information source at the information service would be an area for further investigation.

Both caching and replication make it possible for queries to access data cheaply.

The first question that needs to be answered is do queries benefit from cheap access to information, i.e. in a Grid environment should a query shipping or a data shipping approach be used? The second question is that if a data shipping approach is employed, which method should be used to ensure consistency.

A method to benchmark Grid information systems was outlined in [14] and can be used to compare the two different approaches. This benchmarking method defines two metrics; the query response time and the (α, β)-currency metric. The (α, β)-currency metric provides a probability α that a randomly selected value from a quasi-copy is current after a given period of time β. It differs from definitions of freshness in terms of time since the last observation was made (e.g. [21]), as the age of information alone does not indicate how consistent it is with an information source; it also differs from binary approaches that assume overall knowledge of what is up-to-date (e.g. [23]).

In any case, to evaluate a scenario that resembles a production Grid infrastructure both query response time and the (α, β)-currency need to be taken into account. In our investigation, the reference environment is the Worldwide LHC Computing Grid (WLCG)[5], which is the largest Grid computing infrastructure in existence today [6].

2.2 Query Response Time

The query response time is the time taken from when the query is sent by the client to the information source to when the query result has been transferred back. As the query response time may be linked to specific design decisions, such as query language used and cache data structure, it has to be measured.

One factor that affects the performance of a query and its response time is the number of queries that may be executed concurrently; hence this aspect must also be explored. As a protection from queries that take too long or hang indefinitely, a timeout is usually specified for each query. Unavailability becomes more of an issue as scale grows; it is expected that several information sources may be unavailable if there are many. Hence, as the number of sources increases, so does the probability that a specific query will timeout in a given time period. This suggests that for a large number of information sources, the timeout value may significantly contribute to the performance. As a result, average query response time for parallel queries needs to be found empirically and will be measured for different scenarios.

The Metacomputing Directory Service (MDS) [15] from the Globus is an example of an implementation that adopted the query shipping approach. Its performance has been the focus of a number of studies [14, 24–26]. What these studies show is that when there is a large number of concurrent queries, the query response time degrades dramatically without the use of caching (that implies a data shipping approach). This opens up further questions with respect to the trade-offs between query shipping and data shipping, especially in relation to the scale of today's production Grid infrastructures.

2.3 (α, β)-currency

The quality of information returned can be quantified using the (α, β)-currency metric which provides a probability, α, that a randomly selected object value stored by the Grid information system is current with respect to a grace period β. The probability α can be calculated using (1), where β is the age of the information and λ is a constant for an object type.

$$\alpha = e^{-\lambda\beta} \tag{1}$$

In the case of query shipping, the value for the query response time can be used for the grace period β in the (α, β)-currency equation. In the case of data shipping, the value β in the (α, β)-currency equation is the latency between when the information source changed and when the information service was updated.

In the case of cache, where there is a limit to the time that data can live in the cache (Time To Live – TTL), the minimum value of α is predetermined by the value of TTL. Hence, the (α, β)-currency concept can be used to calculate the optimal TTL value so that α does not decrease below a desired value. For a given quality level α, the maximum TTL β can be calculated using (2).

$$\beta = \frac{ln\alpha}{-\lambda} \tag{2}$$

Following the work in [13], the values of λ can be calculated using (3), where f is the frequency of change and N is the number of instances of an object type.

$$\lambda = \frac{f}{N} \tag{3}$$

Based on the analysis of daily snapshots of the WLCG information system for March 2010, the values for the frequency of change f and the number of instances N for each object type in the Glue 1.3 information model [3] were provided in [13]. Using these values, the value for λ was calculated using (3), which was used in (2) to calculate the optimum value of β for each object type where α=0.999. The results are shown in Table 1.

The results in Table 1 suggest that a single value for the TTL of 30s, which was used in the original MDS deployment, may not be optimal. For many object types, the TTL could be increased to 3600s (1 hour) and α would still be greater than 0.999. However, for object types which have a high frequency of change, the TTL value would need to be reduced to 1-2 seconds. What also needs to be considered is the cost in terms of time required to create the cache as, if the time taken to obtain the information is greater than the TTL value, the cache will be immediately invalidated and hence it will not be possible to provide a cache with the desired quality level.

2.4 Network Environment

One of the main features of a Grid computing environment is that the information sources (Grid services) are widely distributed geographically. This suggests that the underlying network, which links the Grid services, provides physical limitations for data movement and hence may be a significant component of β. As such we will assert that the approach which requires the least amount of data to be transferred should be the

Table 1 The optimum TTL value of β for a *three nines* tolerance level

Object Type	λ (s^{-1})	β (s)
CE	1.91×10^{-03}	0.52
CESEBind	1.14×10^{-08}	88086.72
CESEBindGroup	5.71×10^{-09}	175240.31
Cluster	3.54×10^{-08}	28294.81
Location	7.66×10^{-08}	13069.01
SA	5.11×10^{-04}	1.96
SE	4.85×10^{-04}	2.06
SEAccessProtocol	2.99×10^{-08}	33423.77
SEControlProtocol	3.29×10^{-09}	303847.95
Service	4.86×10^{-04}	2.06
ServiceData	1.99×10^{-09}	503153.62
Site	3.63×10^{-08}	27583.25
SubCluster	2.28×10^{-06}	437.98
VOInfo	3.86×10^{-09}	259381.45
VOView	8.35×10^{-06}	0.68

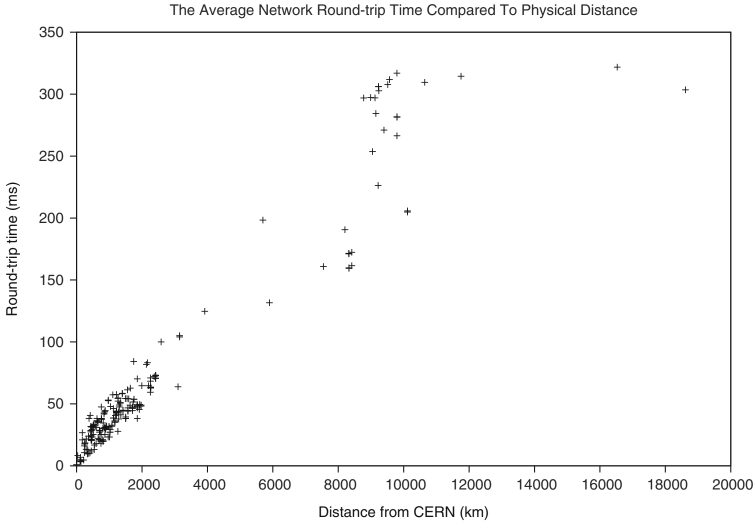

Fig. 1 Average network round-trip time in relation to physical distance from CERN (variance suppressed for clarity)

most efficient approach, with the lowest values for β, and hence α, and the query response time.

In order to understand the network infrastructure in WLCG, the Linux *ping* command was used to measure the round-trip time between a Virtual Machine hosted by the CERN data centre in Geneva and a Grid service at each of the 368 participating sites. As each site publishes its location, the distance between the two geographical locations can be calculated from their longitudes and latitudes. A graph plotting the average round-trip time in relation to physical distance for 64 bytes from CERN is shown in Fig. 1.

As expected, it can be seen that greater physical distance corresponds to higher round-trip times. The average round-trip time for the LAN to a CERN Grid service is less than 1ms. For a Grid service within the same country (Switzerland) or physically close (less than 200km), the average round-trip time was less than 10ms, and within Europe the average round-trip time was less than 100ms. Average round-trip times greater than 100ms correspond to Grid services in North America, South America and the Asia-Pacific region. A Grid service for the University of Melbourne in Australia had the highest average round-trip time, 321.8ms with a standard deviation of 1.7. At a physical distance of 16,500 km from CERN, it is the second furthest after the University of Auckland in New Zealand, which is 18,600 km away and had the ninth highest round-trip time of 303.5ms with a standard deviation of 0.4. The University of Melbourne is, therefore, a good site to select for evaluating the effects of high network latency.

3 An Evaluation Of The Different Approaches

3.1 Query Shipping

3.1.1 The Query Response Time

With the release of the EMI Resource Information Service (ERIS) from the EMI project [1], querying services directly has officially been supported in the WLCG infrastructure as opposed to enforcing site-level aggregation. As the OSG [19] does not offer either site-level or service-level interfaces, the subsequent tests do not include resources from this infrastructure. To discover the information services, the WLCG service registry (GOCDB) [11] was used to obtain the host names for each service and an LDAP URL was constructed from the hostname using the standard port (2170) and the standard bind point (mds-vo-name=resource,o=grid) for the ERIS. A service crawler was created to query all the ERISs and is analogous to a web crawler querying web pages. The maximum information returned by one service was 407KB and the minimum was 72 bytes, which represents the base entry object of the hierarchical database structure. The median size of information returned from all services was 7.4KB with the total information from all services being 40.7MB.

The time taken to query the Service object for each service was measured fifty times and the mean query response time was found to be 0.68 seconds with a standard deviation of 0.85. If the queries are executed using one thread (i.e. serialized), it would take 903

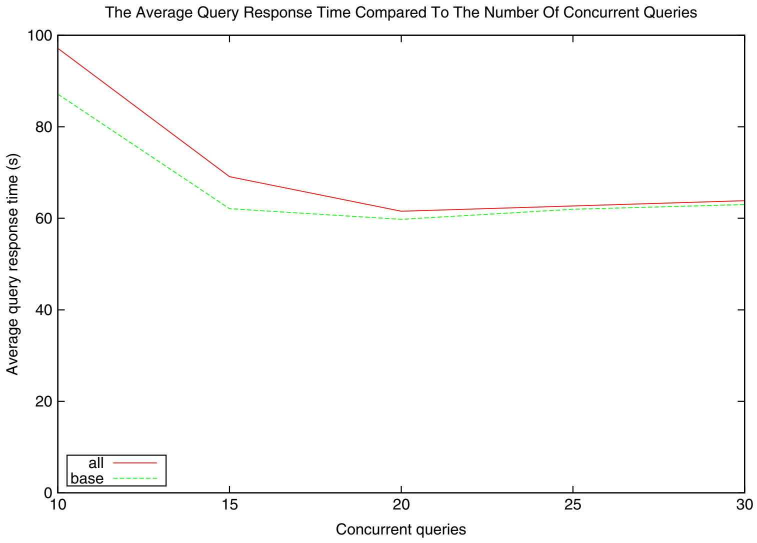

Fig. 2 The average total query response time for different numbers of concurrent queries

seconds to obtain the information from all services. The average query response time was measured using different values for the maximum number of concurrent queries and with a fixed timeout value of 5 seconds. The results are indicated by 'all' in Fig. 2. It can be seen that using a maximum of 20 concurrent queries returns the best results with the information from all services being obtained in 61.6 seconds with a standard deviation of 1.1.

To understand if there is an advantage gained by querying for only one attribute, the same test was repeated but using a query that only returned one attribute from the base entry (55 bytes). The results are indicated by 'base' in Fig. 2. It seems that using a maximum of 20 concurrent queries gives again the best result with an average total query time of 59.8 seconds and a standard deviation of 0.9. It can be seen that there is only a small advantage gained by querying for only one attribute. This suggests that the connection overhead for the query is much greater than the amount of data returned.

As the size of the data returned from the base entry is similar to the information used by the ping command (55 bytes and 64 bytes, respectively), the

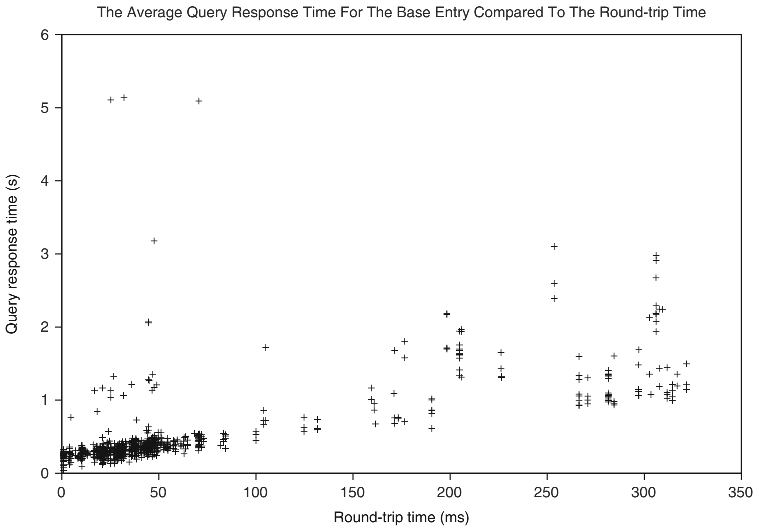

Fig. 3 The average query response time for the base entry compared to the ping time (variance suppressed for clarity)

average query response time can be compared to the average round-trip time. As can be seen in Fig. 3, the average round-trip time does not give an accurate indication of the average query response time, which suggests that the performance of the ERIS may be an important factor. However, there appears that there is a minimum threshold for the query response time whereby no query is less than 3 times the round-trip time and the median is 5.6 times.

The above results suggest that the overhead for obtaining all information is so small that unless network usage is a significant concern (7.4KB per service on average versus 55 bytes for one attribute), the information from a service should be considered as an atomic unit rather than handling each attribute individually.

3.1.2 (α, β)-currency

Taking the average total query time of 59.8s, found in the previous section, as the value for β, which was the best result when all services were contacted and one attribute was returned, the values of α for the top ten queries [10, 14] are shown in Table 2, where the value of λ for the object type can be found in Table 1.

From the results in Table 2 it can be seen that, even with a query shipping approach due to the length of time it takes to execute the query, seven of the ten query responses have a value of α less than 0.999. For Q7 (CE) and Q10 (VOView) α is as low as 0.892 and 0.915, respectively.

Table 2 The (α, β)-currency for the query shipping approach

Query	Object	λ (s^{-1})	β (s)	α
Q1	CESEBind	1.14×10^{-08}	59.8	0.999999
Q2	SA	5.11×10^{-04}	59.8	0.969904
Q3	SE	4.85×10^{-04}	59.8	0.971414
Q4	SE	4.85×10^{-04}	59.8	0.971414
Q5	CESEBind	1.14×10^{-08}	59.8	0.999999
Q6	Service	4.86×10^{-04}	59.8	0.971355
Q7	CE	1.91×10^{-03}	59.8	0.892063
Q8	SA	5.11×10^{-04}	59.8	0.969904
Q9	SubCluster	2.28×10^{-06}	59.8	0.999864
Q10	VOView	1.48×10^{-03}	59.8	0.915299

3.2 Data Shipping

3.2.1 The Query Response Time

Using a data shipping approach the query is resolved directly by the Grid information service and in WLCG an LDAP server is used to provide the query interface for the information service. The average query response time for the top ten queries [10] made to the WLCG infrastructure was measured in [14] along with other relevant metrics. This evaluation found that for the queries with the smallest response size (642 bytes) and largest response size (8.5Mbytes), the average query response times were 0.003 seconds and 0.41 seconds, respectively. The response time for the query which requests all information (e.g. `select *`) from all services was 4.8s with a standard deviation of 0.4 when querying for all objects using a quasi-copy of the information.

3.2.2 (α, β)-currency

The provision of a persistent information source enables alternative approaches for data shipping to be considered, in particular approaches where these information sources actively transfer all the information to the information service. Replicating via a simple copy, as implemented in current Grid information services, is not efficient as all object types are considered identical. This means that for some object types, redundant data is being shipped and for other object types there is a low value for α as data is not being shipped frequently enough. By introducing an active information source, changes can be identified at the information source and the updates can be shipped to the information service. In other words, the changes observed translate into the information that would be sent in such a scenario.

To implement a data shipping approach for transporting changes, three components are required. One at the information source that can detect when a change has occurred and take the appropriate action. A second to transfer this information from the information source to the information service, and a third at the information service to do the necessary updates. One constraint is that reverse discovery should not be necessary, i.e. a service should not have to discover the

Table 3 Cumulative changes as a percentage per object type

Object	Cumulative %
VOView	44.65
CE	86.27
SA	94.79
Service	98.19
SE	99.41
SubCluster	100

information service. Therefore the transfer mechanism has to ensure that information reaches the information service, even if the service does not specify the final location. For this investigation a data shipping approach based on enterprise messaging technology [7] will be used to send the updates directly to the information service. It is not possible on a reasonable time scale to deploy a test on each service in the WLCG Grid infrastructure to capture the updates of the information source [9]. In addition, due to the granularity for the maximum resolution of the snapshots available from querying the services directly, there is an undercount in the number of changes. The updates will, therefore, be simulated and a specific test will measure the best- and worst-case scenarios, respectively.

To simulate the changes, the LDAP DNs (Unique Identifiers) for all objects were harvested from the WLCG Grid information system and grouped by object type. Using the information on changes from Table II and Table III in [13], the changes as a percentage of the attributes of each object type were calculated cumulatively as shown in Table 3.

A random number was generated between 1 and 100 which was mapped to an object type using the information in Table 3. A random LDAP DN was then selected for that object type and an update message was constructed. The update message consisted of the LDAP DN, the relevant dynamic attributes for that object type and the values, which were set to the current time. The average size of the message generated for each object type is shown in the second column of Table 4. The message was then sent to a topic on a messaging broker hosted at CERN using a python client. A delay was added to enable the frequency of messages to match the expected frequency of changes.

To measure the latency, a message consumer was created that subscribes to the topic. This was run on a machine that has the time synchronized with the messaging client. When a message is received, the values for the attributes are compared with the current time to obtain the latency for that message.

Two tests were performed to measure the latency for updating information. One represents the best-case scenario where the information source and service are on the same LAN. The other represents the worst-case scenario where the information source and service are separated by a physical distance that spans half the globe. The first test evaluated the best-case LAN scenario with the message publisher hosted at CERN. The average round-trip time between the publisher and the broker for 64 bytes of data was 2.42ms with a standard deviation of 1.9. The second test evaluated the worst-case WAN scenario with a message publisher at the University of Melbourne in Australia. The average round trip time between the publisher and the broker for 64 bytes of data was 340ms with a standard deviation of 1.1. The latencies for each test and object type are shown in Table 4.

Table 5 shows the re-calculation for the frequencies of change considering the dynamic attributes only, using the same daily snapshots of the information system that were recorded during the month of June 2011. Due to the under-counting problem, the probability p

Table 4 Average message size and latency for the two tests

Object	Size (bytes)	β_{CERN} (ms)	$\beta_{Melbourne}$ (ms)
CE	562.6	2.60	220.10
SA	397.8	2.58	215.20
SE	278.2	2.13	218.27
Service	316.2	2.60	219.14
SubCluster	295.1	2.77	220.56
VOView	375.6	2.37	217.63

Table 5 Frequencies of change for dynamic attributes

Object	Total	Modifications	p	Changes	f (s^{-1})
CE	5081	3964	0.78	18024	0.2086
SA	2957	1012	0.34	1540	0.0178
SE	499	209	0.42	361	0.004174
Service	4150	989	0.24	1299	0.01503
SubCluster	755	210	0.28	292	0.003376
VOView	11081	7942	0.72	28033	0.3245

Table 6 Value of α for the local and remote tests

Object	$\lambda(s^{-1})$	α_{CERN}	$alpha_{Melbourne}$
CE	4.1×10^{-5}	0.9999999	0.9999910
SA	6.0×10^{-6}	0.9999999	0.9999987
SE	8.4×10^{-6}	0.9999999	0.9999982
Service	3.6×10^{-6}	0.9999999	0.9999992
SubCluster	4.5×10^{-6}	0.9999999	0.9999990
VOView	2.9×10^{-5}	0.9999999	0.9999936

that a modification occurs was used to calculate the actual number of changes.

The frequencies of change from Table 5 were then used along with the values for β from Table 4 to calculate the values for α. The results are shown in Table 6.

It can be seen that for the worst-case scenario (the CE object type published from the University of Melbourne) α has a value greater than 0.99999.

4 Discussion

4.1 Query Shipping or Data Shipping?

The results from this investigation reiterate the observation that the query response time is less for a data shipping approach than a query shipping approach. The best result achieved for the query shipping approach in this scenario was an average 59.8s with a standard deviation of 0.9 when using 20 concurrent query threads. This can be compared with a query for all information made to the WLCG infrastructure where the average query response time was 4.8s with a standard deviation of 0.4. Even when considering the (α, β)-currency, due to the time taken for the distributed query, query shipping does not necessarily offer any advantage.

4.2 Considerations for Query Shipping

An interesting observation for query shipping is the importance of the catalogue in the form of a service registry. As the query response time is reduced if less information sources are contacted, being able to use a service registry to exclude information sources will improve the performance. However, there was only a small advantage if the information from each source returned only one attribute. In addition, as each query could potentially contact all the relevant information sources, each information source is required to handle as many queries as they are initiated; hence, query shipping does not scale well if there are many queries as the information sources are potential bottlenecks in the system. The optimum value for the TTL value to be used in a fault-in caching approach shows that for some object types the information would be invalidated every 1-2 seconds. As it takes longer to obtain this information using a query shipping approach, the result is that the cache will always be invalid.

4.3 Feasibility of Data Shipping

Based on the analysis of the snapshots in [13], a data shipping approach would generate nearly 6 million updates a day (69 per second) and have a throughput of 7.6GB a day (88KB per second). Comparing this to a query shipping approach, the top ten queries made to the WLCG infrastructure, which together represent 90.8 % of the queries made, would result in 18.3MB of data being transferred for those 90.8 queries. According to [10], the CERN top-level information service in the WLCG information system experiences approximately 2 million queries per day. Taking the top ten queries as a sample, the total data transferred per day would be 1,748GB. Note that the total size of the data from all information sources is only 104MB. This represents both the volume for the query shipping approach and for the information service itself. This simulation suggests that using messaging is technically feasible as a data rate of 88KB per second is much less that the Gbps WAN connections that are provided.

4.4 Considerations for Data Shipping

Only sending information about a change when it occurs raises a bootstrapping issue with the initial population. This can be solved by obtaining the initial values directly from the information source. Periodically cross-checking with the information source directly may be required to maintain the consistency when experiencing temporary network failures. An alternative approach would be to periodically publish the information which could also act as a heartbeat for system monitoring and could also be useful from a service discovery perspective. The main difference with such an approach is with respect to the Grid infrastructure management policies used as these define the rules regarding which services should be in the infrastructure. How such management policies are implemented to control what services are allowed to join the infrastructure and, hence, control what can be published, or control what is consumed is an area for further investigation.

5 Conclusion

This paper revisited the fundamental concepts of processing queries in a Grid environment in order to identify the different approaches. The fundamental problem of resolving a query over many information sources is a data retrieval problem that has been extensively examined in the distributed query processing research literature. There are two main approaches for distributed query processing; query shipping (moving the query to the data) and data shipping (moving the data to the query). Not surprisingly, the query response time is lower for the data shipping approach than for the query shipping approach, although the actual performance difference may depend on other factors including the implementation. Even when considering (α, β)-currency, due to the time taken for the distributed query, it does not necessarily offer any advantage and, hence, a data shipping approach should be adopted.

If a fault-in approach is used to refresh the cache, some queries will experience higher response times due to the cache being invalidated as a query shipping approach is used to perform the update. A data shipping approach based on either an asynchronous cache update process or replication will give a predictable query response time for all queries. Due to the query scalability demands of Grid computing, with a sufficiently large number of queries, the caching approach would in any case ship all information from the information sources. Hence, the focus should be on efficiently replicating the information from the information source to the information service.

A simple method to maintain a replica of all information from the information source by transferring changes was considered and a prototype implementation based on a enterprise messaging system was evaluated. A simulation was used to evaluate the best- and worst-case scenarios, for shipping the changes from within a LAN and over a high-latency network connection, respectively. It was found that for the worst-case scenario (the CE object type published from the University of Melbourne) the value of α is greater than 0.99999. Considering the real information from the WLCG information system, this approach would generate nearly 6 million updates a day (69 per second) and have a throughput of 7.6GB a day (88KB per second). This novel approach represents a major change for Grid computing or related global information systems as it requires the information sources to be active and explicitly uses data shipping rather than query shipping. However, further work is required to understand how to address the bootstrapping issue and to ensure synchronisation in the event of failures. In addition, how management policies are implemented to control what can be published or consumed is another area for further investigation.

To conclude, the paper provides evidence that a data shipping approach based on an enterprise messaging system that sends changes represents an improvement with respect to the handling of dynamic information than existing approaches.

References

1. Aiftimiei, C., Aimar, A., Ceccanti, A., Cecchi, M., Meglio, A.D., Estrella, F., Fuhrmam, P., Giorgio, E., Konya, B., Field, L., Nilsen, J.K., Riedel, M., White, J.: Towards next generations of software for distributed infrastructures: the european middleware initiative. In: Proceedings of the 8th International Conference on E-Science, pages 1–10, Chicago, IL, USA (2012)
2. Alonso, R., Barbara, D., Garcia-Molina, H.: Data caching issues in an information retrieval system. ACM Trans. Database Syst. **15**(3), 359–384 (1990)
3. Andreozzi, S., Burke, S., Field, L., Litmaath, M.: GLUE schema version 1.3
4. Baeza-Yates, R., Ribeiro-Neto, B.: Modern information retrieval. Addison Wesley, Harlow, England (1999)
5. Bird, I.: Computing for the large hadron collider. Annu. Rev. Nucl. Part. Sci. **61**(1), 99–118 (2011)
6. Bird, I., Jones, B., Kee, K.F.: The organization and management of grid infrastructures. Computer **42**(1), 36–46 (2009)

7. Casey, J., Cons, L., Lapka, W., Paladin, M., Skaburskas, K.: A messaging infrastructure for WLCG. J. Phys. Conf. Ser. **331**(6), 062015 (2011)
8. Cooke, A.W., Gray, A.J.G., Nutt, W., Magowan, J., Oevers, M., Taylor, P., Cordenonsi, R., Byrom, R., Cornwall, L., Djaoui, A., Field, L., Fisher, S.M., Hicks, S., Leake, J., Middleton, R., Wilson, A., Zhu, X., Podhorszki, N., Coghlan, B., Kenny, S., O'Callaghan, D., Ryan, J.: The Relational Grid Monitoring Architecture: Mediating Information about the Grid. J. Grid Comput. **2**(4), 323–339 (2004)
9. David, M., Borges, G., Gomes, J., Pina, J., Plasencia, I.C., Castillo, E.F., Díaz, I., Fernandez, C., Freire, E., Simón, Ā., Koumantaros, K., Dreschner, M., Ferrari, T., Solagna, P.: Validation of Grid Middleware for the European Grid Infrastructure. J. Grid Comput. **12**(3), 543–558 (2014)
10. Ehm, F., Field, L., Schulz, M.W.: Scalability and performance analysis of the EGEE information system, vol. 119 (2008)
11. Ferrari, T., Gaido, L.: Resources and Services of the EGEE Production Infrastructure. J. Grid Comput. **9**(2), 119–133 (2011)
12. Field, L., Memon, S., Márton, I., Szigeti, G.: The EMI registry: Discovering services in a federated world. J. Grid Comput. **12**(1), 29–40 (2014)
13. Field, L., Sakellariou, R.: How dynamic is the grid? towards a quality metric for grid information systems. In: The proceedings of the 11th IEEE/ACM International Conference on Grid Computing, pages 113–120, Brussels, Belgium (2010)
14. Field, L., Sakellariou, R.: Benchmarking grid information systems. In: Proceedings of the 17th International Euro-Par Conference, volume 6852 of Lecture Notes in Computer Science, pages 479–490, Bordeaux, France (2011)
15. Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., Tuecke, S.: A directory service for configuring high-performance distributed computations. In: Proceedings of the Sixth IEEE International Symposium on High Performance Distributed Computing, pages 365–375, Portland, OR, USA (1997)
16. Kossmann, D.: The state of the art in distributed query processing. ACM Comput. Surv. **32**(4), 422–469 (2000)
17. Mastroianni, C., Talia, D., Verta, O.: Designing an information system for grids: Comparing hierarchical, decentralized p2p and super-peer models. Parallel Comput. **34**(10), 593–611 (2008)
18. Ordille, J.J., Miller, B.P.: Database challenges in global information systems. In: Proceedings of the ACM SIGMOD international conference on Management of data, pages 403–407, Washington, D.C., United States, p. 1993 (1993)
19. Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Wrthwein, F., Foster, I., Gardner, R., Wilde, M., Blatecky, A., McGee, J., Quick, R.: The open science grid. J. Phys. Conf. Ser., 78 (2007)
20. Rahman, R.M., Barker, K., Alhajj, R.: Replica Placement Strategies in Data Grid. J. Grid Comput. **6**(1), 103–123 (2008)
21. Sakagami, H., Kamba, T., Sugiura, A., Koseki, Y.: Effective personalization of push-type systems - visualizing information freshness. Comp. Netw. ISDN Systems **30**(1...7), 53–63 (1998). Proceedings of the Seventh International World Wide Web Conference
22. Wolfson, O., Jajodia, S., Huang, Y.: An adaptive data replication algorithm. ACM Trans. Database Syst. **22**(2), 255–314 (1997)
23. Zanikolas, S., Sakellariou, R.: An importance-aware architecture for large-scale grid information services. Parallel Process. Letters **18**(03), 347–370 (2008)
24. Zhang, X., Freschl, J.L., Schopf, J.M.: A performance study of monitoring and information services for distributed systems. In: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, pages 270–281, Seattle, WA, USA (2003)
25. Zhang, X., Freschl, J., Schopf, J.M.: Scalability analysis of three monitoring and information systems: MDS2, r-GMA, and hawkeye. J. Parallel Distrib. Comput. **67**(8), 883–902 (2007)
26. Zhang, X., Schopf, J.M.: Performance analysis of the globus toolkit monitoring and discovery service, MDS2. In: IEEE International Conference on Performance, Computing, and Communications, 2004, pages 843–849, Phoenix, AZ, USA (2004)