A Comparison of the Relative Performance of XML and SQL Databases in the Context of the Grid-SAFE Project

Joshua W. Green

The Grid-SAFE Team

October 2008

Abstract

The Grid-SAFE project will develop a software framework to support accounting, reporting and usage monitoring, and resource management on advanced computing facilities. Grid-SAFE is expected to allow its data to be queried using XPath. SAFE, Grid-SAFE's previous incarnation, uses a relational database to store its usage records. An XPath parser could be written to convert XPath queries into SQL queries, but an alternative solution would be to replace the underlying database with an XML database, allowing queries to be directly passed to the database without the need to analyse them. The key question then becomes one of performance — will an XML database provide acceptable levels of performance for the types of query required by the Grid-SAFE system. An analysis was made of the eXist and Sedna XML databases to see if their performance was sufficient for Grid-SAFE's needs. When compared to the MySQL database currently used by SAFE, the XML databases took between 40 and 400 times longer to gather information in situations Grid-SAFE will commonly find itself in. Both databases were found to be inadequate for Grid-SAFE. Considering that these two XML database are fairly advanced compared with their peers, the practicality of using XPath as the standard query language is called into question.

Contents

1 Introduction **1**

2 Background **2**
2.1 How SAFE Currently Uses its Database 2
2.2 XPath and XQuery 3
2.3 Indexing and Schema Awareness 3
2.4 The maturity of XML Databases 4

3 Databases **4**
3.1 Databases Selected 4
3.2 Other Possible Candidates 4

4 Method **5**
4.1 Sample Data 5
4.2 Dates 5
4.3 Queries and Time Sets 5
4.4 Database Configuration 6
4.4.1 MySQL 6
4.4.2 eXist 6
4.4.3 Sedna 7
4.5 Test Machine 10
4.6 A Note About XML Conversion 10

5 Results & Analysis **10**

6 Conclusions **13**

1 Introduction

As part of the JISC-funded Grid-SAFE project, EPCC is developing a software framework which provides accounting, and reports usage statistics and resource management on advanced computing facilities. EPCC has already developed a similar product called SAFE which has been used for several years now on many of the national HPC services EPCC maintains. The task now it to modify SAFE to conform to the specifications as recommended in the JISC report on accounting and monitoring frameworks [PKRC07].

One specification not currently implemented by SAFE is the ability to query the data held by Grid-SAFE using XPath, and have the results returned in an XML format. The feature implies Grid-SAFE would be best suited by using an XML database to store it's data. In terms of fulfilling the specification, this is most probably true. However, the quantity of data managed by Grid-SAFE is substantial. The development team have voiced concerns that an XML database would not be efficient enough to perform the tasks required of it.

This report attempts to ascertain whether the development team's concerns are well founded. Data has been taken from a SAFE implementation running on HPCx[1] and fed into several databases. Various queries of a format commonly used by SAFE were run on each database in order to compare their efficiency.

SAFE already uses a relational database at its core and it is unlikely that this will change during the initial development of Grid-SAFE. If it can be shown that XML databases are efficient enough, an appropriate XML database may be integrated into the core of Grid-SAFE during a later phase

[1]one of the larger computing resources managed by EPCC on behalf of the ESPRC

of development. In the event that the current state of the art of XML databases cannot provide the required performance, the relational database currently used by SAFE will be left in place. Attempts to support XPath will be made and the appropriate bodies who develop the specifications Grid-SAFE must adhere to will be informed of the problems and urged to provide suitable solutions in successive versions of the specifications.

2 Background

A full evaluation of the advantages and disadvantages of XML databases is beyond the scope of this report. However, a summary of the key benefits XML databases offer are as follows:

1. XML databases are good at handling unstructured data.
2. Many sets of data do not map well to relational databases but do map well to the hierarchical structure of XML databases.
3. The structure of data in XML databases is much more flexible and amenable to change compared to the data structures used in relational databases. This feature becomes more important as the products that uses the database evolve.
4. A lot of services already use XML, either as a native format or as a data transmission format. Since a lot of communication is now done with XML, it may be more efficient to use XML natively instead of continually encoding and decoding data.
    - This is particularly important if the database is being used purely to store and retrieve XML.

Let us consider these benefits in relation to Grid-SAFE. Grid-SAFE will store usage records which have a well defined structure. The structure maps well to the way relational databases store data, making benefits 1 and 2 irrelevant in this context.

Benefit 3 may have some value for Grid-SAFE; it is not inconceivable that the structure of usage records will change over time.

Any database will have to convert the usage record to its internal format. Good XML databases never store their data in raw XML, but the format is likely to be closely related to XML and easier to convert than the formats used by relational databases. However, this conversion is only necessary when data is being imported or exported. Most of the queries performed by SAFE are used to generate data for internal use. The data is not required in any particular format as SAFE's internals can be rewritten to handle any format the database produces. The conversion to XML is only required when data is leaving the system. While benefit 4 certainly has merit, it does not have major implications for Grid-SAFE.

2.1 How SAFE Currently Uses its Database

SAFE has a well defined pattern of interaction with its database and there is little reason to believe this pattern will change much as it evolves into Grid-SAFE. Almost all queries take the form of a summation of a particular entry from records which were created or finished between a particular date interval. An example can be seen in figure 1. As a result, date is the most important index used.

Most of the statistical information is used to create graphs. On average, these graphs contain between 10 and 30 points, with each point requiring a summation query over a particular time interval. The exact interval and value to be summed can vary dramatically so it is not always possible to produce the data in advance. When the user requests analysis, they must wait for all database queries to be executed and the graph and web page to be generated. To avoid potentially frustrating user-perceived

```
SELECT a, COUNT(*), SUM(b)
FROM job_usage_record
WHERE ‘StartedTimestamp‘> 1167609600
  AND ‘CompletedTimestamp‘> 1170316800
  AND ‘CompletedTimestamp‘< 1170435600
GROUP BY a;
```

Figure 1: An example of the most common format of a query submitted by SAFE. *a* and *b* vary but start and completed time are almost always used to filter entries. In its current implementation, SAFE uses UNIX timestamps as dates.

performance, the longest acceptable delay should be few seconds and preferably less (under a second would be ideal).

Based on SAFE's operation while monitoring the usage of HPCx and Hector[2], work load for Grid-SAFE is not anticipated to be very high[3]. While concurrency must be planned for, it is unlikely that many sets of queries will be executed simultaneously. Therefore, the performance of a database may be considered sufficient if it can perform 30 queries of the form shown in Figure 1 within a few seconds.

2.2 XPath and XQuery

The RUS specification that Grid-SAFE aims to adhere to mandates searches by XPath. XPath is a language that has been developed to extract parts of XML documents. XQuery is a full query language which supports the concept of variables, functions and a larger set of operations. XQuery includes XPath as part of the language. For more information, please see the W3C's documentation of XPath[W3C08b] and XQuery[W3C08c].

Grid-SAFE is not expected to support XQuery. However if it were to use an XML database, the database would almost certainly support it. Although the language is not mandated for use by XML databases, it is fairly ubiquitous among the more mature implementations. It is reasonable to assume XQuery will be available for use within Grid-SAFE, even if Grid-SAFE does not make such functionality externally available.

2.3 Indexing and Schema Awareness

The performance of any database improves dramatically when various elements are indexed. SAFE currently employs this tactic and indexes all date entries. This greatly improves the performance of the MySQL database it currently uses. Good XML databases support the notion of indexing and this technique was employed during testing.

Relational databases define the types of their elements when a table is created. Although XML offers an aritrary range of data types, they are not specified within the XML document. This information is stored in a separate document called the *schema*. To say that an XML database is *schema aware* implies that it may import schema and make use of them to optimise queries performed on documents to which the schema applies. This feature is an optional extension of XQuery and is currently not widely supported. Only a handful of commercial databases are schema aware and there are almost no free open source implementations that have this ability.

[2]Another large computing resource managed by EPCC
[3]Based on observations made by SAFE's principal architect and maintainer

2.4 The maturity of XML Databases

It should also be noted that XML databases are still evolving. Anecdotal observations as to their maturity compare them to the state of relational databases in the mid-Nineties. There is still much room for improvement. While still considered slower than relational databases, XML databases are closing the gap.

Grid-SAFE will not be finished for several years, at which time the performance of XML databases in comparison to relational databases may very well have improved. However, there is no guarantee that the relative performance improvement (if it occurs) will be significant. As a result, it is appropriate to rely on results measured today to determine the adequacy of XML databases for use with Grid-SAFE when it is finally released.

3 Databases

SAFE currently uses MySQL to hold its accounting and statistical analysis data. This database product will be used as a control for all results. An XML database must have at least comparative performance to MySQL if it is to be considered as a replacement.

The W3C XML Query home page[W3C08c] specifies 53 XML database implementations. To narrow the choice down, several limits and guides were employed:

1. Implementations must run on multiple platforms. This is a requirement for Grid-SAFE so it should be a requirement for databases that Grid-SAFE uses.

2. Implementations must be freely available. There are not enough resources available to pay for the use and testing of commercial databases. Most free trial versions do not have the required features, or prevent access to large enough data sets to be tested effectively. In any case, an efficient implementation of Grid-SAFE should not require extra financial resources to operate[4].

3. The W3C's official test suite report page[W3C08a]. As there is only enough time to test a handful of databases, picking ones that are more developed and feature complete seems prudent.

4. A thesis that measured the performance of several XML databases[Hal05] was used to weed out several implementations.

3.1 Databases Selected

The eXist Database EXist is a very popular and commonly used XML database. eXist only allows the existence of one database but does support multiple documents, collections of documents and collections of collections. It provides a graphical user interface for local administration as well as a command line interface.

The Sedna Database Another very popular and commonly used XML database. Among other things, Sedna is used by *wikixmldb*[5] to search for information on Wikipedia. Considering that the size of Wikipedia's XML documents reaches 24GB, Sedna should be able to scale to the sizes Grid-SAFE requires.

3.2 Other Possible Candidates

Several other databases were investigated and are listed below. Some were rejected because they did not meet the selection criteria. Others because a cursory look at their specification suggested they would not be as efficient as implementations already being tested.

[4]although they are entirely appropriate as optional extras

[5]`http://wikixmldb.dyndns.org/`

Mark Logic Apparently a very fast solution, and supports schema awareness. Unfortunately, Mark Logic is a very expensive commercial offering and was not available for testing. Due to its price, it has been ruled out from deployment with Grid-SAFE.

Berkeley DB XML An extension of the of the Berkeley DB written in C++ and designed mainly to be an embedded database rather than a stand alone server. The Berkeley DB has a strong reputation and is used extensively by Oracle. Berkeley DB XML sits on top of the Berkeley DB and allows XML data to be stored and queried using XQuery. It is pretty much the only free XML database which (at least partially) supports schema awareness. Unfortunately a lack of time meant that we were unable to test this database.

Saxon There was not enough time to test the basic version of this database. There was no reason to suggest it would perform significantly better than the ones chosen. A commercial version is schema aware.

Stylus Studio This only works on Windows.

X-Hive Supports data awareness but according to Hall's analysis[Hal05] it does not perform as well as eXist and Sedna.

4 Method

4.1 Sample Data

Data collected by SAFE from HPCx during 2007 was used. This is real data and is of the volume one would expect Grid-SAFE to be able to handle. There were 301,641 records in all, taking up just under 50MB when represented as text in columns, or just over 200MB when represented as XML of the format to be read by an XML database[6].

4.2 Dates

Almost all queries performed by SAFE filter their results primarily by date. The current date format used in the MySQL database is the UNIX time format. The RUS specification which Grid-SAFE targets specifies that XML queries must be XPath 1 compliant. XPath 1 does not support a date format which means that dates must be represented by strings. This limitation is potentially crippling as string comparison is far more time consuming than number comparison. Nevertheless, this limitation must be observed. Since none of the databases being tested are schema aware, and thus perceive dates as strings, this limitation is somewhat enforced by current XML database technology.

It should be noted that this limitation places a bias on the results as the final results cannot be truly said to represent a direct comparison between the performance of cutting edge XML databases and relational databases. If an XML database is used, it may be prudent to represent dates internally by a more appropriate method and convert the output accordingly. An alternative approach would be to seek to change the RUS specification to support XPath 2, which does support a date format. Making use of databases that are schema aware and using the XQuery date functions would also help.

4.3 Queries and Time Sets

The queries used in these tests take the form of the vast majority of queries performed by SAFE. These queries primarily filter entries by date. Query 1 is used to measure the performance of the

[6]The size this data takes up within a database can vary dramatically from database to database

Id	End Date - lower bound	End Date - upper bound	Number of Destin ct Records
1	2007-02-01 08:00:00	2007-02-02 17:00:00	1,321
2	2007-11-01 22:36:10	2007-11-05 22:36:10	2,776
3	2007-05-01 00:00:00	2007-05-31 23:59:59	23,077
4	2007-08-01 22:36:10	2007-10-01 12:00:10	45,767
5	2007-01-01 00:00:00	2007-11-01 23:30:00	248,819
6	2007-01-01 10:10:10	2007-12-20 15:15:15	294,014

Table 1: The six time intervals used for each query

databases for this critical function. Query 2 is closer to the kind of query performed by SAFE: a sum of some values over a set of entries within some time period. The queries were modeled on SQL queries taken from SAFE's usage logs.

Dates appear in two locations in the data: *Create Time* (XML element) or *StartedTimestamp* (SQLentry) denotes when the record was created. *End Time* or *CompletedTimestamp* timestamp denote when the job specified in the record finished. Queries of the standard form specify a create time jobs must have started after, and two end times between which the job must have finished.

For all measurements made, the start time was set to the earliest time in the data set. A set of six end time intervals was used to see how the size of the results set changes the query execution time. This provides a set of twelve queries: two basic queries, each with six end time intervals.

From now on, the end time intervals will be referred to by number. There range and number of distinct entries are shown in Table 1.

Intervals one, three and six contain roughly a day, a month and a year's worth of entries. These intervals are important because most queries performed by SAFE use these scale of interval. Day and month intervals are by far the most common so the performance of these queries are particularly important.

4.4 Database Configuration

The databases had to be modified in different ways to optimise the queries executed on them. The modification primarily took the form of marking start and end dates in the records as being entries for the databases to index. This modification improves performance by at least an order of magnitude for all databases. All databases were installed locally on the test machine to avoid network file system overhead. Their query execution time was measured using the UNIX *time* command.

4.4.1 MySQL

The queries executed on MySQL can be seen in figure 2. The queries were directly copied from SAFE's usage logs and only slightly modified to make sure they were exactly the same as the ones used on the XML databases. Any optimisations used by SAFE such as the *count(1)* instead of *count(*)* were retained.

StartedTimestamp and *CompletedTimestamp* were marked as elements to index in the table definition presented to MySQL

4.4.2 eXist

The queries executed on eXist can be seen in figure 3. The queries were based on the SQL queries taken from SAFE's usage logs. XQuery's FLOWR[7] syntax was avoided as it does not give databases

[7]FLOWR: shorthand for *For, Let, Order by, Where, Return*, the standard order of commands used within XQuery. Pronounced *flower*

```
SELECT UserName
FROM ‘HPCx-2007‘
WHERE ‘StartedTimestamp‘   > 1167609600
AND   ‘CompletedTimestamp‘ > 1170316800
AND   ‘CompletedTimestamp‘ < 1170435600
AND   1=1;
```

Query 1

```
SELECT Class,
       SUM(NumCPUs) AS ’Total CPUs’,
       COUNT(1) AS ’NJobs’
FROM ‘hpcx-2007‘
WHERE ‘StartedTimestamp‘> 1167609600
  AND ‘CompletedTimestamp‘> 1170316800
  AND ‘CompletedTimestamp‘< 1170435600
GROUP BY Class;
```

Query 2

Figure 2: MySQL Queries used

the chance to apply as many optimisations as the equivalent XPath statements[8]. This was tested and found to be true for eXist.

eXist provides the option of adding a configuration file which may contain elements (specified by qname) to be used as indices. Once this has been done, the collection must be indexed. This takes about five minutes for a collection of the size tested.

eXist allows queries to be sent through a web service. It was decided that this would be avoided as it would add an unnecessary overhead. An administration client written in Java could be used to execute queries on the database directly, but this prohibits the use of scripts to automate the measurement collection process. The command line interface provided by eXist requires the database to be started and shut down for each query. This would not happen if eXist were properly integrated into Grid-SAFE. The time taken for eXist to perform an extremely simple query was measured several times and the average was subtracted from all other measurements made. This should provide a fair value for the amount of time eXist requires to execute a query without adding bias to the measurements by including the server’s start up and shut down time.

Attempts were made to see if eXist’s performance could be improved by presenting data to it in one large file instead of 365 smaller ones. Unfortunately, eXist was unable to import a file so large (210MB).

As with the MySQL measurements, times were measured using the UNIX *time* command.

4.4.3 Sedna

The queries executed on Sedna can be seen in figure 4. As with the queries executed on eXist, these queries were based on the SQL queries taken from SAFE’s usage logs. The FLOWR expression of XQuery was avoided as with eXist’s queries but no test was performed on Sedna to determine if this was beneficial or not.

Sedna uses its own language extensions to achieve indexing. Indexes must be created with the *CREATE INDEX* command which is submitted to Sedna as a query. Unlike eXist, Sedna does not

[8]According to various documentation on eXist’s website: `http://exist-db.org/xmlprague06.html`

```
declare namespace ur="http://www.gridforum.org/2003/ur-wg";

let $input      := collection("hpcx-2007"),
    $startTime  := '2007-01-01T00:00:00.000Z',
    $endMinTime := '2007-02-01T08:00:00.000Z',
    $endMaxTime := '2007-02-02T17:00:00.000Z'

return $input//ur:UsageRecord[ur:StartTime gt $startTime]
                             [ur:EndTime gt $endMinTime]
                             [ur:EndTime lt $endMaxTime]
                             //ur:LocalUserId
```

Query 1

```
declare namespace ur="http://www.gridforum.org/2003/ur-wg";

let $input      := collection("hpcx-2007"),
    $startTime  := '2007-01-01T00:00:00.000Z',
    $endMinTime := '2007-02-01T08:00:00.000Z',
    $endMaxTime := '2007-02-02T17:00:00.000Z'

let $ur := $input//ur:UsageRecord[ur:StartTime gt $startTime]
                                 [ur:EndTime gt $endMinTime]
                                 [ur:EndTime lt $endMaxTime]

for $queue in distinct-values($ur/ur:Queue)
let $urq := $ur[ur:Queue=$queue],
    $count := count($urq),
$numCPUs :=sum($urq/ur:Processors)
order by $queue ascending
return (<result>{$queue}&#x9;{$numCPUs}&#x9;{$count}</result>)
```

Query 2

Figure 3: eXist Queries used

```
declare namespace ur="http://www.gridforum.org/2003/ur-wg";

let $startTime  := '2007-01-01T00:00:00.000Z',
    $endMinTime := '2007-02-01T08:00:00.000Z',
    $endMaxTime := '2007-02-02T17:00:00.000Z'

let $ursEndInterval := index-scan-between("EndTime",
                                          $endMinTime,
                                          $endMaxTime,
                                          "SEG")

return $ursEndInterval[ur:StartTime gt $startTime]//ur:LocalUserId
```

Query 1

```
declare namespace ur="http://www.gridforum.org/2003/ur-wg";

let $startTime  := '2007-01-01T00:00:00.000Z',
    $endMinTime := '2007-02-01T08:00:00.000Z',
    $endMaxTime := '2007-02-02T17:00:00.000Z'

let $ursEndInterval := index-scan-between("EndTime",
                                          $endMinTime,
                                          $endMaxTime,
                                          "SEG")

let $urs := $ursEndInterval[ur:StartTime gt $startTime]

for $queue in distinct-values($urs/ur:Queue)
let $urq := $urs[ur:Queue=$queue],
    $count := count($urq),
$numCPUs :=sum($urq/ur:Processors)
order by $queue ascending
return (<result>{$queue, $numCPUs, $count}</result>)
```

Query 2

Figure 4: Sedna Queries used. The *index-scan-between* function is discussed in section 4.4.3

allow the specification of indices by qname. Instead, each index is given a name and a collection on which it operates over. Inbuilt methods *index-scan* and *index-scan-between* can then be used to extract entries from the collection. This method of extraction involves modifying the queries in a way that makes them incompatible with other XML databases. Indexing on Sedna takes about one minute to create each index on a data set of the size used.

4.5 Test Machine

The machine used to perform all test was an Intel Quad Core 2.4GHz with 4KB cache and 8GB memory. It runs a version of Scientific Linux. The machine uses a network filesystem, however all test data was deliberately stored locally on the machine to remove the network overhead. For the same reason, all databases were also installed locally.

4.6 A Note About XML Conversion

The measurements for the XML databases include the time taken to convert the solution into XML. The MySQL measurements do not. If Grid-SAFE is to meet the RUS specification, this task must be performed whenever the data is to leave the system. As a result, there is an additional overhead associated with MySQL in this situation which has not been measured.

Conversely, any queries used internally only require the raw data and may be designed to read it in whatever format the database produces. In this instance, the overhead of XML creation by the XML databases is unnecessary. While a full set of measurements were not made to determine the size of this overhead, queries over time sets five and six (which have large result sets) were observed to take about 5 times longer when XML was generated. The delay on queries with small result sets sets (time sets zero and one) was about an factor of two. These values were observed due to a feature in eXist which allows one to set a maximum limit on the number of results displayed. The limit does not alter the result of the query but instead puts a limit on the number of results actually returned when the query completes.

5 Results & Analysis

The the execution time for query 1 can be seen in figure 5. The measurements performed on MySQL are substantially smaller than those performed on the XML databases. As a result, all graphs from now on will display the measurement taken from MySQL multiplied by 100 to aid comparison. The value was obtained by taking the time for one query and multiplying it by 100 (as opposed to timing 100 queries).

Results of the queries run on the eXist, MySQL (scaled $\times 100$) and Sedna databases can be seen in figures 6 and 7. At best, the XML databases take 35 times longer to perform their queries. Sedna achieved this result with eXist generally performing two orders of magnitude slower than MySQL. eXist seems to improve in comparison to Sedna as the query time interval gets larger.

Table 2 shows the time it would take to extract the information required to generate graphs from the data over various time intervals. Technically, it is possible to get all information with one query and calculate the individual points from the data returned. In practice, this is less efficient for MySQL. The effect on efficiency was not tested for eXist and Sedna. Although XQuery does lend itself more to this kind of query than SQL, it is unlikely that the XML databases would benefit from index acceleration while calculating the individual graph points[9], meaning a performance increase is my no means certain.

Sedna's performance comes close to what is required but the amount of time spent retrieving data to generate a graph over a year is far too long. It would also seem that for XML databases,

[9]Indexes usually operate over a fixed set of data. Indices cannot, in general, be used over an extracted subset of the indexed data.

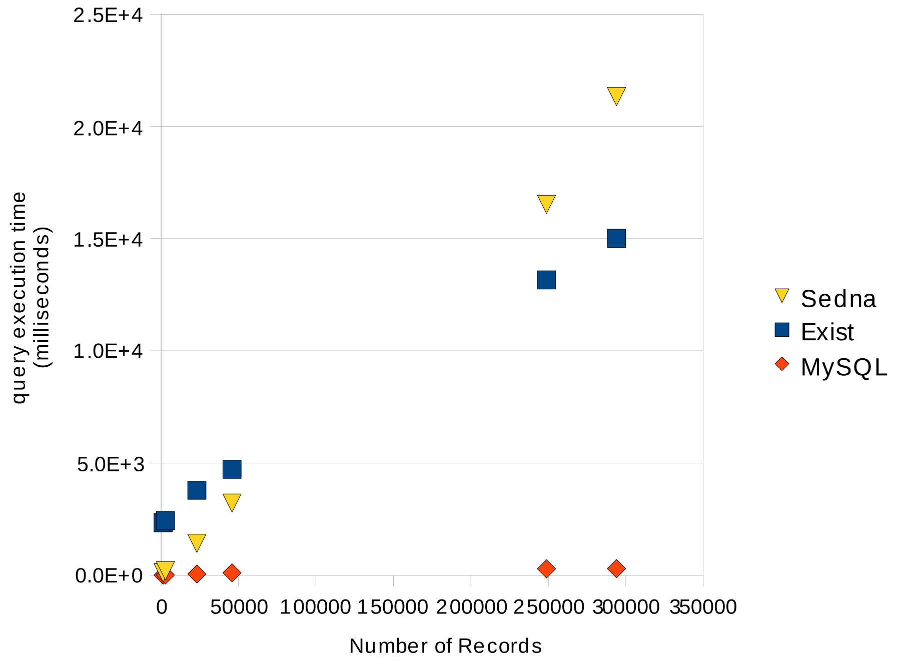

Figure 5: Time taken to execute query 1, using the 6 different time intervals. Results shown against the number of records whose completion time falls within the specified interval.

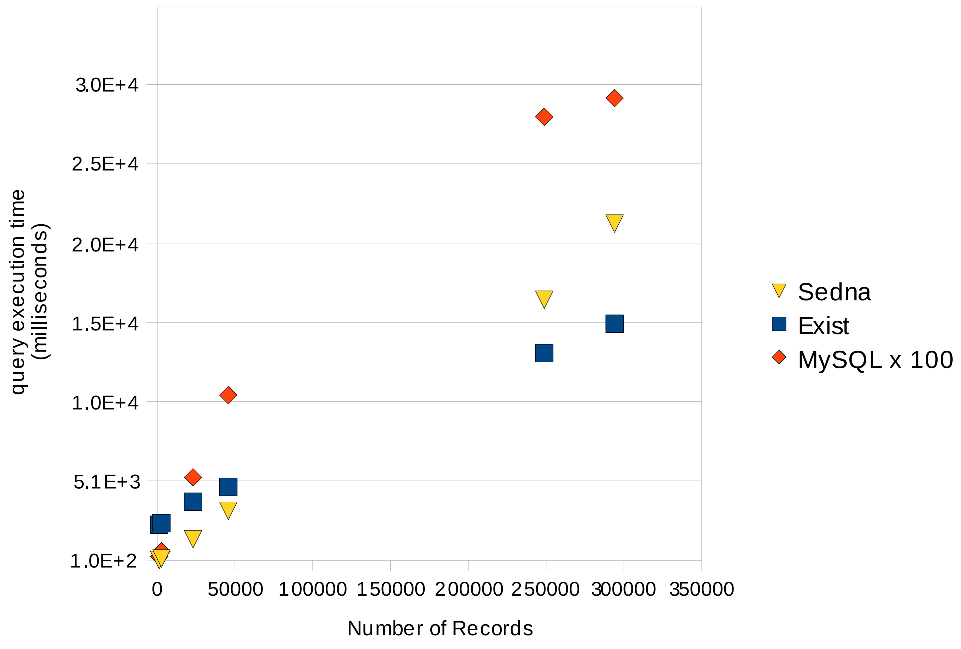

Figure 6: Time taken to execute query 1, using the 6 different time intervals. Results shown against the number of records whose completion time falls within the specified interval. Timings for MySQL have been multiplied by 100.

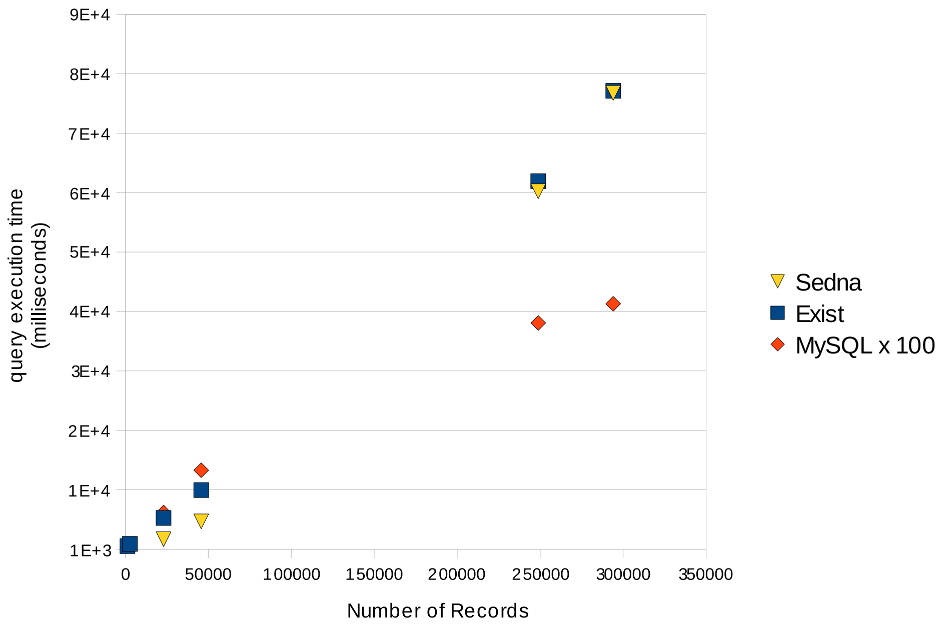

Figure 7: Time taken to execute query 2, using the 6 different time intervals. Results shown against the number of records whose completion time falls within the specified interval. Timings for MySQL have been multiplied by 100.

Graph Range	**Queries needed**	**MySQL**	**eXist**	**Sedna**
Week	7	0.03	10.6	1.05
Month	30	0.13	45.5	4.50
Year	12	0.86	75.1	32.23

Table 2: The time it would take to extract inforation needed to produce a graph with the specified ranges from the different databases

time increases linearly with the size of the results set. In contrast, MySQL's time increase is pseudo-logarithmic[10]. This is worrying as some queries will attempt to perform analysis over several years. It would appear that this kind of analysis is completely impractical using when XML databases.

It should be noted that tests were to performed to make sure the XML databases were indeed making use of their indices. Sedna was found to perform far better with the indices than without and an examination of eXist's trace file shows it too is making use of them.

One possible shortcoming of these measurements is lack of use of a schema. With such a schema, XML databases could determine the structure of the data more accurately and optimise themselves accordingly. As mentioned in section 2 most XML databases, including eXist and Sedna, cannot use schema. This investigation would have benefited from an analysis of schema aware databases. The benefit a schema would have brought would be type casting, allowing the date fields to be compared as dates rather than strings. XQuery features date comparison functions which could not be used during testing due to the prohibitive cost casting string dates in the data set to date objects at query

[10]MySQL timing measurements appears to be logarithmic but actually curve more steeply. This was discovered by plotting the exponent of query execution time against result set size. The graph is not included here as it shows little more than what has just been described

execution time.

One possible solution to the problems faced by the XML databases would be to maintain different collections with different sets of information in them. Collections would be smaller and so faster to search. While this would improve performance, it could also introduce a large storage overhead. When stored in the database, eXist takes up an order of magnitude more storage space to store the sample data set than MySQL does. Many types of graph can be generated by Grid-SAFE and creating collections of collections of collections of data would add an unwanted extra hierarchical layer which would be harder to write queries for[11]. Such an approach could hardly be called elegant and is more of a basic fix than a proper solution.

Due to time limitations, the performance of the databases as a function of the total usage record set was not examined. The use of indexes should make query execution time increase as the log of the size of the results set. This will certainly be true for MySQL, but whether or not it is true for eXist and Sedna was not investigated.

6 Conclusions

XML databases have come a long way in the last few years. They are, however, still noticeably slower than relational databases. While they may have their advantages when handling data ill-fitted to a table structure, this advantage is not useful in the context of Grid-SAFE's needs. Usage records are, as their name suggests, records and the relational database's data structure perfectly suited for storing records. Indeed, the relational data structure is ideally suited to Grid-SAFE's needs.

A loss of performance would not be so important if XML databases could perform well enough. The technology is still under heavy development after all and they will almost certainly improve over the next few years. However, the XML databases tested were not suitable for our particular purposes and there is no way of ensuring the technology will improve sufficiently for performance to reach acceptable levels by the time Grid-SAFE is complete.

Grid-SAFE needs to continue using its current relational database if it is to to perform adequately. This does pose the interesting question of how it will support XPath as required by the RUS specification. XML databases are a fascinating technology and may one day be able to replace relational databases in these situations. However, until the technology matures it cannot be considered practical or usable for the tasks Grid-SAFE requires it to perform.

References

[Hal05] David Hall. *An XML-based Database of Molecular Pathways.* PhD thesis, Linköping Universet, Sep. 2005.

[PKRC07] M. A. Pettipher, A. Khan, T.W. Robinson, and X. Chan. Review of accounting and usage monitoring – final report, July 2007.

[W3C08a] W3C. The offical test suite for databases reporting to support xquery, Oct. 2008.

[W3C08b] W3C. The world wide web consortium XML path language, Oct. 2008.

[W3C08c] W3C. The world wide web consortium XML query page, Oct. 2008. Last Modified: 2008/09/08 19:44:00 (Revision 1.146).

[11]If collections were ordered by date interval, one would have to know this when writing a query and select the appropriate collection. This could be done automatically by parsing the query but it would not be an elegant solution