SAMGrid Peer-to-Peer Information Service

Matthew Leslie[1,2], Siniša Veseli[2]

[1]Oxford University Computing Laboratory
[2]Fermi National Accelerator Laboratory

m.leslie1@physics.ox.ac.uk, veseli@fnal.gov

Abstract

SAMGrid presently relies on the centralised database for providing several services vital for the system operation. These services are all encapsulated in the SAMGrid Database Server, and include access to file metadata and replica catalogs, dataset and processing bookkeeping, as well as the runtime support for the SAMGrid station services. Access to the centralised database and DB Servers represents a single point of failure in the system and limits its scalability.

In order to address this issue, we have created a prototype of a peer-to-peer information service that allows the system to operate during times when access to the central DB is not available for any reason (e.g., network failures, scheduled downtimes, etc.), as well as to improve the system performance during times of extremely high system load when the central DB access is slow and/or has a high failure rate. Our prototype uses Distributed Hash Tables to create a fault tolerant and self-healing service. We believe that this is the first peer-to-peer information service designed to become a part of an in-use grid system.

We describe here the prototype architecture and its existing and planned functionality, as well as show how it can be integrated into the SAMGrid system. We also present a study of performance of our new service under different circumstances. Our results strongly demonstrate the feasibility and usefulness of the proposed architecture.

INTRODUCTION

The high energy physics community places stringent demands on its data handling systems. Experiments such as MINOS, and the D0[2] and CDF[1] detectors at Fermilab generate petabytes of data which must be stored and made available to physicists for analysis[3]. This is made more challenging by the international nature of the collaborations that analyse this data. The CDF experiment, for instance, has collaborators in 11 countries on three continents.

To meet these demands, the SAM-Grid[4] system has evolved to be both robust and fault tolerant. However, like many grid systems, it relies heavily on central services. While some of these services can easily be configured to failover to (possibly off-site) backups, no such possibility exists for the central database, which stores all information about the SAM-Grid system. This reliance on a single database creates two problems, a load bottleneck which limits scalability, and a single point of failure, which limits failure tolerance. Here we describe efforts to reduce this dependency through deploying a scalable and fault tolerant peer to peer information service.

In sections and we give a brief overview of the existing SAM-Grid information service, and describe why we feel a peer to peer replacement is appropriate. We describe recent advances in peer to peer software that power our new system in section , and how we incorporate them into SAM-Grid in section . In section , we investigate the performance of our implementation of this architecture. Finally, in section , we discuss the context of this work and offer concluding remarks.

EXISTING SAM-GRID ARCHITECTURE

The SAM-Grid system offers a wide variety of services for data transfer, cataloguing, data storage and process bookkeeping in a distributed environment. SAM-Grid users can create datasets of physics data files based on metadata attributes, then use the SAM system to manage the delivery and processing of these files, and finally the storage of the results.

Two of the main system components are the *Station* and the *DBServer*. It is the station that requests and logs the delivery of files to user projects, recording which files are stored on which disks, and managing storage space. To record and retrieve this data from a persistent store, the station uses CORBA method calls to communicate with the DBServer. All SAM-Grid information is stored in the central database, and so the DBServer must translate these requests into SQL and pass them on. The results the database returns are then processed and returned to the station. The DBServer hides stations from the underlying database schema, and provides a level of indirection between the station and the database that we have exploited in our information service architecture.

MOTIVATION

Although it is possible to run more than one DBServer, the SAM-Grid design does not allow for more than one database. This limits both the scalability and the fault tolerance of the entire system. Though the Oracle database has proven reliable, network outages can still bring all off-site processing to a halt. As eighty percent of the 50 stations currently running as part of the D0 experiment are hosted

off-site, there is a strong case for a more failure tolerant information system.

Additionally, the runtime of many SAM-Grid commands increases significantly during the nightly central database backups. This illustrates a need for a more scalable information system that offers more consistent performance.

To meet these goals, we looked at recent research into fault tolerant distributed systems, particularly Distributed Hash Tables(DHTs), which we discuss in detail in the next section.

DISTRIBUTED HASH TABLES

Distributed hash tables provide a solution to the lookup problem in distributed systems. Given the name of a data item stored somewhere in the system, the DHT can determine which node that data item should be stored on, often with time complexity logarithmic in the size of the network.

There are many DHT designs available, including Tapestry[13], CAN[12], Kademlia[10] and Chord[11]. We chose to implement our information system using Chord. In Chord, the keyspace can be thought of as being arranged in a ring. Nodes and data items are assigned IDs within this keyspace. Each node is responsible for storing the data it is the first clockwise successor of. In Figure 1, for instance, node 54 is responsible for data with keys between 40 and 53.

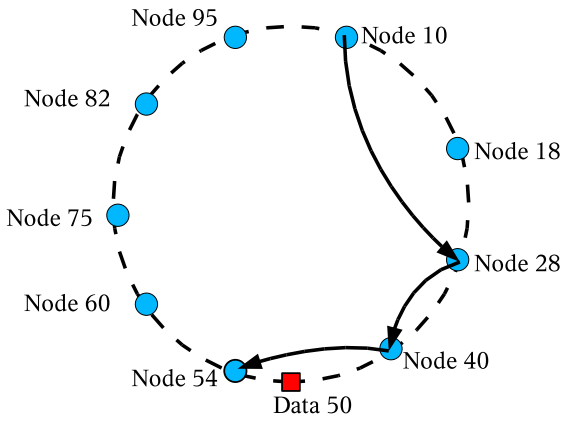

Figure 1: A Chord ring in which Node 10 is looking up key 50. For the purposes of this diagram, keys are between 0 and 100

To provide scalability, each node need only maintain knowledge of a small proportion of other nodes in the network, including a number of its clockwise successors, its immediate predecessor, and several *fingers* at distances one half, one quarter, one eighth, and so on of the way round the ring from it. It uses this knowledge to forward requests for keys it does not own to nodes which are closer to the requested key. Figure 1 shows how a lookup for key 50 originating at Node 10 might travel between nodes. As the distance is reduced by a constant fraction at each routing hop, lookup time are logarithmic in the number of nodes.

Nodes are allowed to join and leave the system at will, causing *churn* in the set of nodes in the system. Chord uses maintenance algorithms to repair routing tables despite node churn. Our information system also needs to recover from data lost when nodes leave the system. To do this, we store a number of replicas of all data.

To maintain reliability in the long term, we must replace the replicas of data that are lost when a node fails. Previous information system solutions have recommended that soft-state storage is used, and a central store periodically reinserts the data [5]. The bandwidth costs of such a scheme are high, however, and as SAM-Grid currently stores metadata on over a million files, it was felt that looking up and reinserting all of this data on a regular basis would be prohibitively expensive. Instead, our information service runs an additional data storage and replication algorithm to efficiently maintain replicas.

SYSTEM ARCHITECTURE

DHTs allow the lookup of data based on a name, such as a filename, a common use case for the current SAM-Grid information system. DHTs do not inherently support complex queries, another use case that SAM-Grid currently allows. In SAM-Grid, complex searches are triggered by dataset definitions.

A dataset definition is a set of criteria for file metadata that defines which files a physicist is interested in analysing for a particular project. Once a dataset definition is created, it is then possible to search for the relevant files and store the result as a *snapshot*. When this dataset definition is required in future, the snapshot can be referred to instead of carrying out the search a second time. This reduces load on the central database, but also increases the proportion of operations which can be supported by a DHT.

If snapshots are correctly used, the majority of SAM-Grid operations can continue without the availability of complex search. Because of this, we judge that a DHT based information service would significantly improve the reliability and scalability of SAM-Grid as a whole.

In order to integrate a Chord DHT with SAM-Grid, we provide an intermediary information service (IS) that takes requests from the station and selectively either passes them to the central DBServer, or uses the Chord Ring to look up the information.

The basic architecture is shown in Figure 2. Each station runs its own information service, and these form the Chord ring. Information such as file metadata is stored in the ring under a name based on its unique identifier. These names are then converted by a hash function into a 160 bit Chord ID. The associated data is then serialised and stored on the node responsible for that ID. If the data can also be associated exclusively with a single station (for instance information about that station's disks) the data is also stored on the information service belonging to that station. Lookups

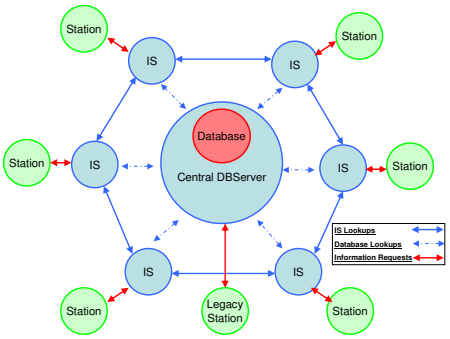

Figure 2: System architecture

for a specific piece of information can then be satisfied by querying the Chord ring.

When a station requests information, the information service will first consult the Chord ring. If the information is not found, the lookup is passed to the central DBServer, and the results will be both inserted into the ring and returned to the station. Any updates are sent to the ring and to the database. The central DBServer is able to send updates to the Chord ring if a station contacts it directly.

In order to maintain data consistency if the network is split, only limited write activity is permitted when the central database is not available. Decisions on what data can be changed or created without centralised coordination are made on a case by case basis, depending on the data type being modified. Changes are generally limited to data that a particular station controls exclusively, so that only a network fragment containing that station may modify that data. Where changes are made without central control, temporary updates are at first made in the Chord ring only, until the database is reactivated and the data can be committed.

As mentioned in the previous section, we maintain reliability by creating and maintaining a number of replicas of all objects stored. The number of replicas stored, and the rate at which data maintenance runs is critical to system reliability. Using information on the average uptime for a SAM-Grid station, and the model given in [9], we chose to configure our system with three replicas and set the data maintenance algorithm to run every five minutes.

This functionality should be sufficient to remove reliance on the central database or any specific information service for the most common station operations, and should also allow limited station autonomy, so that a station may continue to process local files without network access.

SYSTEM PERFORMANCE

Our initial implementation of this architecture has demonstrated excellent performance. To illustrate system scalability and fault tolerance, we present examples from a test deployment of a 12 node information service, comprised of nodes at Fermilab and in Oxford.

In figure 3, we show the results from scalability tests. We ran up to 10 simultaneous clients, and recorded the time to fetch an item that had previously been stored into the Chord Ring. We compare this with results obtained using the current centralised implementation at CDF. The times shown are averaged over 2000 fetches. The centralised implementation becomes significantly slower as the number of clients increases. The bottleneck in this test was database I/O, and not the DBServer. The Oracle database, must fetch the data to answer each query sequentially from a single disk array. Because of this, the centralised version shows poor scalability.

The peer to peer version shows only very slight variation in fetch times as load increases. Although our new version is slower than the centralised implementation with only one client, it is an order of magnitude faster when serving ten clients.

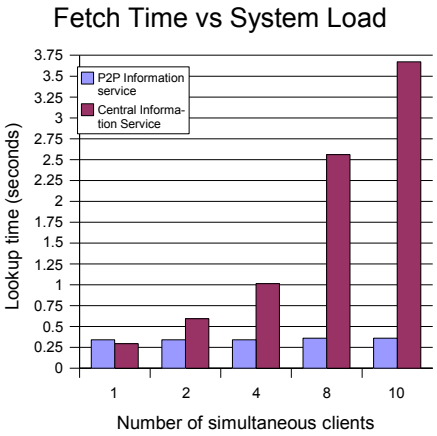

Figure 3: Fetch Performance Against System Load

The system used very little bandwidth, even when maintaining large quantity of data. Figure 4 shows the outgoing bandwidth from our information system under a workload of repeated fetches and stores. The maintenance bandwidth usage is initially around 300B/s, increasing to around 500B/s when data is first stored and the data maintenance algorithm starts to run. As would be expected, outgoing bandwidth peaks when data is being fetched. Routing data stores to the appropriate node also requires increased outgoing bandwidth.

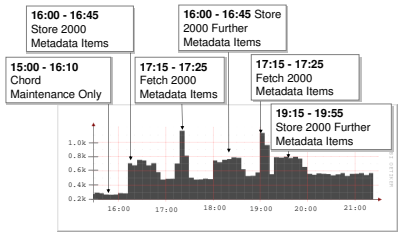

Figure 4: Bandwidth Usage During System Tests

To assess failure tolerance, monitor the fetch latency of

two clients while simulating the failure of a single node. In the majority of cases, the failure had no impact on either client. In other cases, a client was engaged in communication with the node that failed. This led to a communication error, triggering a back-off period followed by a retry - which was always successful.

DISCUSSION

Related Work

A peer to peer replica location service based on Chord is described in [5]. Our work builds on this by allowing the storage of all system information. We also offer improved reliability through the use of data maintenance algorithms, and propose a mechanism for consistent updates despite the possibility of network splits.

A great deal of work has also been done on complex search and range queries in DHTs[6, 8]. Such a system would make the full functionality of SAM-Grid available during database downtimes, however we do not currently consider search to be sufficiently critical to SAM-Grid operations to justify the implementation costs of these designs.

Further Work

Our current implementation of the P2P Information service allows for the storage and retrieval of immutable data. The architecture does allow for mutable data, however, and an implementation and evaluation of transactional updates would certainly be an interesting area for further work.

We currently maintain replicas using the DHash maintenance algorithm given by Cates [7], but in future versions, we would like to use a Dynamic data maintenance algorithm. Dynamic algorithms are more complex, but can offer faster lookup times and improved fault tolerance [9].

Conclusions

We have shown that a peer to peer information service is feasible, and can be integrated with existing Grid software. Initial evaluations of the system are promising, showing the scalability and fault tolerance properties that we hope to achieve. These very positive results indicate that further development would be worthwhile.

Acknowledgements

We would like to thank Fermilab Computing Division for its ongoing support of the SAMGrid project, and especially the CCF, CEPA, and Run II Departments. We would also like to thank the UK Particle Physics and Astronomy Research Council (PPARC) for it's support of e-science research. Finally, we would like to thank everyone at D0 and CDF who has contributed to this project.

REFERENCES

[1] The Collider Detector at Fermilab. *http://www-cdf.fnal.gov/*.

[2] The D0 experiment. *http://www-d0.fnal.gov/*.

[3] SAM data production plots. *http://cdfsam-prd.fnal.gov/sam_local/PlotsAndStats/ProductionPlots/SamProductionPlots.html*.

[4] A. Baranovski et al. SAM-GRID: A system utilizing grid middleware and sam to enable full function grid computing. *Nucl. Phys. Proc. Suppl.*, 120:119–125, 2003.

[5] M. Cai, A. Chervenak, and M. Frank. A peer-to-peer replica location service based on a distributed hash table. In *SC*, page 56, 2004.

[6] M. Cai, M. Frank, J. Chen, and P. Szekely. Maan: A multi-attribute addressable network for grid information services. In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, 2003.

[7] J. Cates. Robust and efficient data management for a distributed hash table. Master's thesis, Massachusetts Institute of Technology, May 2003.

[8] M. Harren, J. M. Hellerstein, R. Huebsch, B. T. Loo, S. Shenker, and I. Stoica. Complex queries in dht-based peer-to-peer networks. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 2002.

[9] M. Leslie. Reliable data storage in distributed hash tables (preprint submitted for publication). *http://arxiv.org/abs/cs.DC/0507072*.

[10] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of IPTPS02*, 2002.

[11] R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM SIGCOMM 2001*, San Diego, CA, September 2001.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.

[13] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, Jan 2004.