SAMGRID PEER TO PEER INFORMATION SERVICE

M. Leslie (Oxford University), S. Veseli (FNAL)

SAMGrid

SAMGrid is a large scale distributed system to deliver petabyte scale datasets for processing at the CDF, Minos and DØ experiments.

It does this by providing the following services in a single unified framework:

- Managing File Storage
  - Files are housed on tape and cached on disks around the world
- Managing File Delivery
  - Get files from tape or cache
  - Provide location transparency
  - Manage your local cache
  - Use a variety of file transfer mechanisms
- Managing File Metadata
  - The SAM database allows metadata based file retrieval
  - User does not need to know a filename
- Providing Analysis Bookkeeping
  - Which files you ran over, with which application.
- Managing Jobs
  - Choose an execution site, deliver job and data to it and store output

Existing SAMGrid Information System

Some of the main components of the existing SAMGrid Information Service are:

- **The Station:**
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 discover information on these objects, the station communicates with the DBServer.

- **The DBServer:**
The DBServer receives queries from the Station, contacts the database for the necessary information, and processes the results.

- **The Database:**
The Oracle database stores all information about the SAMGrid System. Ultimately, all requests to read or write information are fulfilled by the database.

Motivation for Change

The current information architecture, while effective, leaves the database as a load bottleneck, and as a single point of failure. Stations are unable to operate during network splits or database downtimes. Additionally, during operations which affect database performance, such as nightly backups, the performance of the entirety of SAM-Grid suffers.

Distributing the data seems a natural solution to these problems, however searching complex distributed metadata can be slow and consume large quantities of bandwidth. Additionally, it is difficult to guarantee that any search result is complete, since nodes may be missing from the system. This would be unacceptable for SAM-Grid.

Analysis of user logs however, shows that complex search in fact is a relatively rare use case. The vast majority of requests are simple lookups for specific pieces of information. Such lookups could easily be satisfied by a distributed lookup service, which would reduce the reliance on the central database. Most SAM-Grid operations could then continue during database downtimes, unaffected by high database load.

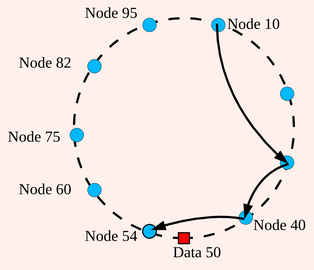
We chose to implement such a lookup system using a Distributed Hash Table

Distributed Hash Tables

Distributed hash tables (DHTs) are the focus of much current research in distributed computing. They allow information to be located in an efficient, fault tolerant manner.

Our new Information System uses the Chord DHT. Chord nodes automatically arrange themselves in a ring, similar to that shown on the right. Each node is assigned a key, and is responsible for data with keys between it and it's clockwise predecessor.

Information can be found by forwarding requests between nodes. Each node need only know about a small number of other nodes in the network, yet information can be found in time logarithmic in the number of nodes.



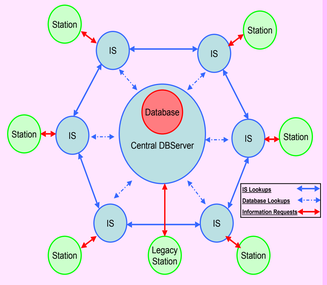
A Chord ring. The black arrows show a request for the data item with key 50 being routed from Node 10 to Node 54, which stores the data.

New Architecture

Our new information service architecture is illustrated in the figure to the right. 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 160 bit Chord ID based on its unique identifier. If the data can also be associated exclusively with a single station, the data is also stored on the information service belonging to that station.

When a station requests data, the information is looked up in the ring. If the data cannot be found in the ring, it is requested from the central DBServer, and then cached in the Chord ring. Updates are sent both to the chord ring and to the central database. Some information updates can also be queued in the chord ring alone during database downtimes.

Maintenance algorithms make sure that the routing structure is repaired when nodes fails, and replicas of all data are stored and maintained to provide extra failure tolerance.



Performance

Performance testing of our initial implementation of this architecture has revealed that such a system is not only feasible, but has excellent performance and transparent fault-tolerance. Lookup times in our 12 node test are significantly faster than those achieved with the previous system, when as few as two clients are simultaneously connected. The plot below-left shows response times are almost unchanged by increased client load, in stark contrast to the previous centralised implementation.

No user intervention is required despite regular node failures. The bandwidth plot below right shows an IS node recovering automatically from a neighboring node failure within 30 seconds. It then recreates the lost replicas of information, whilst continuing to serve clients as normal.