Elastic Quotas and Distributed Data Handling on the SAM-Grid

Charles Dowling
DePaul University
Chicago, IL
chas@fnal.gov

Andrew Baranovski
Fermi National Accelerator Laboratory
Batavia, IL
abaranov@fnal.gov

Abstract

CDF and DZero are the two largest collider experiments at Fermilab, and the largest currently running high-energy physics experiments in the world. In DZero the amount of data, the throughput of data processing, and the size of the collaboration present a unique challenge for the experiment's meta-computing system. With these considerations in mind Fermilab has developed the SAM-Grid system to allow globally distributed high-throughput data processing and analysis.[1] At the core of the solution is the data-handling system SAM. The goal of SAM is to provide a middleware between physics applications and low level storage and network resources to enable streaming of the analysis data in a transparent fashion. An important component of the SAM system is the cache management module. Physics applications often request the same data repetitively, and the cache management module was designed to take advantage of this behavior. In this article we discuss an administrative aspect of that module, user groups and quotas, which we use to efficiently allocate storage resources in the multi-user SAM-Grid. We begin by discussing the limitations of the current policy, and then present a proposed quota management system inspired by Leonard et al.'s [3] elastic quotas.

1. Introduction

A typical SAM Station's cache consists of several nodes and disks [2]. These devices are conventionally mapped to hard drives and computers specifically employed to deliver storage and network resources for the grid. Because a typical application runs multiple consumer processes, it expects data from one or more sources simultaneously. It is SAM's job to cache this data and maintain the status of the nodes.

Because disk space on the nodes is a limited resource, that resource must be managed. SAM has to balance replacing data files that are no longer in use with the risk of improving one application's performance at the expense of another. To keep any one job from getting too large SAM enforces quotas on disk storage space. These quotas are placed on user groups to assure that a group's total storage use is below the limit determined by SAM administrators.

Under the current policy, groups cannot replace files that belong to other groups. Although this policy is easy to implement it is not a viable long term solution. We have observed that some nodes may become inaccessible to new groups because they do not have files of their own on the node to replace. Since there are many user groups the cache becomes quickly fragmented. Furthermore, if the administrator wishes to reallocate quotas, he or she must often manually remove files to allow a quota to increase.

It is clear that a new quota management strategy is required. Below we describe a cache management policy inspired by Leonard et al.'s work on elastic quotas [3]. We divide each node into guaranteed space and elastic space and set respective quotas per group. By allowing users to replace other groups' files in the elastic space a group can no longer be locked out of a particular node.

2. Design

We begin (Section 2.1) by explaining a new dual quota system and its advantages. In (Section 2.2) we discuss how one can decrease administration by providing a mechanism that allows old files to be removed automatically via a Least Recently Used (LRU) policy. There are two primary requirements that influence all aspects of design. First, administration of file management must be minimal, and second, since this is a production grid all bookkeeping operations must be computationally inexpensive.

2.1. Quotas

Every user group is given two quotas: a Guaranteed Quota (GQ) and an Elastic Quota (EQ). The GQ assures users that at minimum they will have a certain amount of persistent file storage. Note that it is not the case that GQ must be greater than zero. For example, a test group might

only be allowed elastic storage out of deference to more important projects.

The EQ represents the amount of elastic space accessible to the group. If the group fills its GQ it may continue to store new files on the node until it's EQ has been reached (typically one-hundred percent of the elastic space). If we define the elastic space (ES) as:

$$ES = TotalDiskSpace - \sum_{i}^{no.ofgroups} GQ_i \qquad (1)$$

we can ensure that files within a group's GQ will never be compromised. Note that if

$$\sum_{i}^{no.ofgroups} EQ_i = ES \qquad (2)$$

we have merely implemented the static cache that we wanted to avoid! [1] It is thus desirable to set $EQ = ES$ and restrict EQs only if the administrator sees fit.

2.2. File Removal Policy

Since the elastic space is available to all user groups it is likely that storing files in the space will require removal of another group's data. It is thus necessary to implement a policy that is not only fair to all users, but easily administered.

Since we do not want the administrator to have to manually delete files, a basic LRU policy makes sense. This guarantees that the most space any group can monopolize is its GQ. When SAM needs to delete a file from ES, it looks for the least recently active *group* instead of least recently used file. This determination can be made in constant time via a hash table. If this group is over its GQ, SAM deletes whatever space it needs according to the group's pre-existing cache replacement policy. If the space needed is more than can be erased without compromising GQs, SAM moves on to the next least recently active group.

At this time we will not consider how malicious users might game the cache replacement policy because we consider it unlikely.

References

[1] http://www-d0.fnal.gov/computing/grid/.

[2] G. Garzoglio. A globally distributed system for job, data, and information handling for high energy physics. FERMILAB-THESIS-2005-32.

[3] O. C. Leonard, J. Nieh, E. Zadok, J. Osborn, A. Shater, and C. Wright. The Design and Implementation of Elastic Quotas: A System for Flexible File System Management. Technical Report CUCS-014-02, Computer Science Department, Columbia University, June 2002. www.cs.columbia.edu/ library.

[1] or worse if they write to the same space