U.S. Department of Energy Best Practices Workshop on File Systems & Archives
San Francisco, CA
September 26-27, 2011
Position Paper

First Author Name Affiliation e-mail address	**Second Author Name** Affiliation e-mail address

Philippe DENIEL
CEA/DAM
philippe.deniel@cea.fr

ABSTRACT / SUmmary

CEA/DAM manages two compute centers : TERA100 (first SC in Europe) which is dedicated to classified applications and TGCC which is an open compute center for institutional collaboration (see http://www-hpc.cea.fr/en/ for details). The management of produced data lead CEA's teams to deal with several specific issues, making them develop their own solutions and tools. This paper is focusing on fFiles's Llifetime, and metadata management.

INTRODUCTION

CEA/DAM has been involved in HPC for many years. Because the compute has widely increased, the amount of produced data as drastically increased as well, making it necessary to have dedicated systems and dedicated teams to handle the architecture in charge of storing the data. This situation leads to several challenges : keeping data available to end users is of course one of them, but not the only one. With a huge amount of data comes a huge amount of metadata records. Consideration haves to be taken to manage them. Last, the data kept areis not all of same value. When some files are criticals, others are not, but managing this aspect may be painful to the user who has thousands of files to delal with and sort. Tools have then to be made available to users to help them deal with information life cycle.

Quotas and retentions

Ian Fleming said “diamonds are forever”, but for such files are not forever. The main issue there comes from the users. They produce lots of data (a daily production of 30 to 100 TB a day is a very common situation at CEA/DAM), but they often done't care about what the data become. This leads to a perpetually growing storage system where less than 1% of the content is accessed. Finally a big amount of files will never be read and are even totally useless once the run of the code is over (checkpoint/restart files for example). But the truth is this : if not forced, a user will never delete his files. Two main reasons for this:

- Llack of time
- Afraid fear of accidentally deleteing useful data

I suggest two solutions to handle this. The first is an old-fashioned Unix paradigm : quotas. The second is more sophisticated and is based on extended attributes to implement files's retentions.

Quotas usually works on a "space used" and a "used inodes" basis. File's size is not that critical (modern FS and storage system are huge today), but consideration on inodes are more interesting because they depict well the numbers of metadata records owned by a user. This is interesting in today's situation where the meteadata footprint becomes the filesystem's limitation. Quotas are simple to set, manage and query (quotactl function in the libC, RQUOTAv2 protocol to be used jointly with NFS), but is has its inconvenientlimitations. One of them is the distributed nature of the filesystem used in the HPC world. In a massively distributed product where data areis spread across multiple data servers with parallel pattern, it becomes hard to efficiently keep a centralized place to keep user's information on quotas. Anyway, I suggest that when available, quotas are to be used because they are a simple way of setting limits to the users, making them aware of the amount of files and data that they own.

Files rRetentions is a an other promising another way. The idea is to associate a specific metadata record to every file and directory. This is done by using extended attributes (aka xattr), which makes the assumption that the underlying storage system's namespace handles such a feature. This xattr will contain an information on the object's lifetime. This can be something like "this file will stop being of interest after a given date" or "this file can be considered useless isf not read/written during a defined period". The key there is to have this metadata for every file (with users input). Specific tools will then audit the file system, produce a list of files to be deleted based on "retention policies". The user will be warned (mail...) when some of theirhis files are candidates for deletion. Finally files are purged. This approach can lead to a virtuous circle : when producing data, users will take the habit to set the parameters to tell how long they'll requireneed the files, giving to the administrator input on their file's lifetime. This is good for the sysadm that who will save space on his storage system, and this is good for the user can who can schedule the deletion of his files, avoiding the painful task of cleaning his directories when quota limit is reached.

Metadata management

Past challenges tofor filesystems wasere size : would the available resources be large enough to store everything I want to put in the system ? Then come performance consideration, and the idea that the users hate to wait to access their data. Right now, these aspects are addressed by modern filesystems (for example Lustre which is widely used at CEA) that are based on a distributed design relying on multiples data servers.

But many files means many metadata records and this can quickly be problematic, especially in a HPC environment. People who have once seen a single directory with hundreds of thousands of files in it know what I am speaking about. Beyond the technical consideration (big directories are an "edge" situation), the manageability of such exotic objects is a real problem : a single "ls -l" in it may last for hours.

Frequent filesystem audits (like those from CEA's RobinHood product (http://robinhood.sf.net)) helps in this : it becomes easy to identify "nasty" patterns in users directories and takes corrective actions. For example, the admin could decide to pack a big directory into a single tar file. Providing users with tools withusing "best practices enforcement" is also a way we follow. Copying Data copydata to the storage system goes is to delegated to a utilitytool that can decide to pack the data automatically.

Metadata volume is definitely an aspect to be seriously considered. Data volume issues have been solved by striping the data. It may not be so easy to stripe metadata because they carry internal dependencies (a file belongs to a directory and can exist with several names if hard links are available) which may limit the algorithms. I actually believe that the main challenge for the filesystems on exascale compute center will be metadata management. Starting into considering this issue today, by setting limits to users to prevent them for to creating “file systems's monsters” and by teaching them the good practices is definitely something to be done today.

ConclusionS

The Exascale systems are coming tomorrow. Beyond the compute power's revolution, there is an incredible technical gap for the storage system. Data management will not be the greatest challenge, but metadata management will. The systems we will have at this time will store data that are produced today or have been generated in the past years. If we are not careful today, we will come to an excruciating situation in the future. And for sure, tomorrow's issues can be smoothed today by setting metadata's useage limits (quotas, retentions) and by providing users with tools to reduce metadata production.