U.S. Department of Energy Best Practices Workshop on

File Systems & Archives

San Francisco, CA

September 26-27, 2011

Position Paper

HPC Data Storage Strategy at the UK Atomic Weapons Establishment (AWE)

Mark Roberts
AWE High Performance Computing
mark.roberts@awe.co.uk

Paul Tomlinson
AWE High Performance Computing
paul.tomlinson@awe.co.uk

ABSTRACT / SUMMARY

AWE has adopted a resilient and globally accessible storage model to support concurrent desktop and compute cluster access. We now provide a global persistent multi-tiered storage which has enhanced usability and reliability. Increasing pressure on budgets has recently focused efforts to reduce and consolidate the directly-attached parallel cluster file system into several global scratch file systems cross mounted on all compute clusters.

INTRODUCTION

Historically, AWE has focused on one or more compute clusters, each with its own local parallel scratch file system and global persistent storage provided by commodity Network Attached Storage (NAS) filers or in-house Network File System (NFS) servers.

Following a major facility issue a few years earlier, which resulted in the sole persistent data store, based on IBMs General Parallel File System (GPFS), being corrupted and had to be restored (very slowly) from backup, it was decided to upgrade to a resilient GPFS cluster. This was designed to provide multi-site resilience, protecting from loss of either site.

This has allowed us to maintain native multicluster GPFS access to the numerous cluster log-in nodes, visualisation clusters, and secure NFSv4 access direct to the desktop.

COMMON ENVIRONMENT

A common name space, providing a consistent view of file systems on desktops, compute and visualization clusters, helps users locate their data and has encouraged well structured work flow with respect to makefiles, shared libraries and code areas.

This, combined with other common environment features, has aided the trend towards more local prototyping, with easier scaling up onto larger platforms.

DESKTOP ENHANCEMENT

The Hierarchical Storage Managed (HSM) file systems are now exposed to desktops, running file managers and search utilities, that scan, index and try to determine type by content, all potentially triggering unwanted retrievals, which can block interactive access until the recall from tape completes.

To mitigate this, we modified the KDE3 file manager and GNU utilities (*ls, find, stat, file)* to be aware that a file is migrated based on the naive concept that a migrated file has a non-zero file size with zero data blocks. Users are given visual cues with syntax highlighting or icon overlays, along with extra options for handling such files. Preview operations that would generate multiple recalls are skipped. Obtaining the migration state via an API or extended attributes that could propagate through software and network protocols would be preferable. However the simple approach has worked well in our environment.

FILESYSTEM USAGE TRENDS

For many years, we recorded per-user GPFS file system usage with a simple POSIX *find* command in conjunction with the HSM *dsmls* command. This was highly inefficient, taking over 24 hours to complete a scan, as well as adding unnecessary CPU and I/O load on the Tivoli Storage Manager (TSM) server.

We now utilize the GPFS Information Lifecycle Management (ILM) interface to scan the file systems and output records containing the extended information such as name, size, access, modify and create time. The resulting data file is processed and imported into a MySQL database. This allows for fine granular analysis at the user and file system level day by day or over a time period.

Currently, with 35 million objects, the new method takes under 20 minutes. We believe that providing such granular information influences users' data storage behavior for the better, and lets us quickly identify unreasonable or unintended usage when problems occur. It also provides long-term trend information to aid future file system capacity planning and procurement.

We have also engaged with Lustre developers to see if future Lustre releases can incorporate a similar capability.

DECOUPLED PARALLEL FILESYSTEMS

Traditionally, as part of a compute cluster procurement, we purchased storage for a dedicated parallel file system to provide the localised fast bandwidth required by codes.

With the potential of cluster lifetimes becoming shorter due to the the ever increasing pace of technology releases, the cost of the disk infrastructure is now becoming a factor. Once our clusters were decommissioned the disk was also removed and disposed of. With a recent capability cluster procurement the local parallel file system hardware accounted for approximately 15% of the total expenditure. The demand for storage is increasing, so as the total of memory and cores on clusters the associated storage costs may have a larger impact.

We have decided to “decouple” the local parallel file system which allows the storage to be used by multiple clusters. This gives the freedom to either use the cost reduction by increasing the compute size or directing the saving elsewhere. One immediate advantage of no directly attached storage is more rapid initial cluster deployment and, possibly, regular scheduled maintenance without affecting access to the data via other clusters.

The community gain improved useability by not having to transfer the data between the local parallel file systems on the clusters and then to persistent storage.

By extending the concept of resilience to global scratch with a global scratch file system cluster in each facility (three planned) we can now factor in scheduled downtime and upgrades to a chosen global scratch file system cluster more effectively.

DATA AND FS AWARE SCHEDULING

It is our intention that each compute cluster should use by default the global scratch file system in its local facility.

Users may, however, wish to use another global scratch file system in a different building or use the “local” global scratch in conjunction with it. In order to prevent jobs failing due to an unavailable global scratch, we are investigating the concept of storage as a consumable resource in the same manner as a node or cores is used today.

By integrating awareness of global storage into the scheduler/resource manager, jobs may be prevented from failing prematurely. Also when global scratch or persistent storage clusters are scheduled for maintenance then scheduler system reservations can be placed on the file system preventing jobs from being dispatched, if the job requested that file system as a resource.

DATA EXPLOSION

With an increasing amount of scratch, persistent storage and upgraded network links, it becomes relatively easy for the user to copy everything everywhere. This leads to wasted bandwidth, disk storage and tape backups due to duplicated data. File system de-duplication on persistent storage may be possible but ultimately undesirable.

With early MPP clusters it was often quicker to move data from disk or recall from off-line storage than regenerate the data. With the large fast clusters available today it may, in some cases, be quicker and cheaper to save network, disk, and tape resources by regenerating data. This is ultimately a decision that only the user is best placed to make but having to weight up the impact on QA reproducibility and provenance.

Existing compute cluster parallel file systems were designed with the ability to hold four times the amount of system memory from a OS initiated checkpoint. The majority of the mainstream codes are now using restart dumps generated via an in-house I/O library. Code users can then choose whether to perform a full state restart or focus on only saving selected data within the run for analysis. Intelligent software-based restarts greatly reduce the amount and bandwidth required and could allow considerable cost savings, but non-restartable third party codes remain a barrier.

CONCLUSION

Implementation of a fast, secure, exported and resilient global parallel file system for persistent and archive storage has proved invaluable for unifying compute resources at all scales. However the ease of accessibility has created some additional problems and raised user expectations, requiring adaptation of the user interface. We are now exploring a similar approach for efficient global scratch storage.