U.S. Department of Energy Best Practices Workshop on

File Systems & Archives

San Francisco, CA

September 26-27, 2011

Position Paper

Nicholas P. Cardo
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
cardo@nersc.gov

ABSTRACT

Disk quotas are a useful tool for controlling file system space consumption. However, each file system type provides it's own mechanism for displaying quota usage. Furthermore, each file system could display the information differently. Unifying how quota information is reported would simplify the user's experience.

Also having quotas span multiple file systems would provide users some flexibility in storage usage.

INTRODUCTION

The use, management, and enforcement of disk quotas is often difficult to interpret at the user's level as well as being too rigid of an enforcement mechanism.

Identification of the issues

While disk quotas are extremely useful in managing disk space, they are often complicate, hard to understand, counter productive to the user community.

Lets first examine quota-reporting utilities. For IBM's General Parallel File System (GPFS), the command `mmlsquota` is used

```
nid00011:~> mmlsquota home1
                      Block Limits                               | File Limits
Filesystem type       KB    quota    limit in_doubt grace | files   quota   limit in_doubt grace Remarks
tlhome1     USR 29491548 41943040 41943040    34032  none |  7680 1000000 1000000      429  none fshost
```

to display quota limits and usage.

For Lustre, quota information is obtained with the command `lfs quota`.

```
nid00011:~> lfs quota -u juser /scratch
Disk quotas for user juser (uid 500):
     Filesystem  kbytes   quota   limit   grace   files   quota   limit   grace
       /scratch 2666948       0       0              83       0       0
sc-MDT0000_UUID     120               0              83               0
sc-OST0000_UUID       4               0
…
```

While standard Linux utilizes the quota command.

```
$ quota   Disk quotas for user juser (uid 500):
Filesystem blocks quota limit grace files quota limit grace
/dev/fs0        2   100   200           2    10    20
```

So now there are three different commands each with a different syntax showing different information. Instructing users how to interpret the results can be quite involved. This is especially true when the data is closely examined to exactly what the users really care about. At that level all that matters is what is being consumed and what the limit is. Lustre is quite detailed in its output and provides space consumption information down to the Object Storage Target (OST). This presents a case of information overload as file systems could have 100's of OSTs and each one represents one line of output. But the real question is do users really need to see this.

File system quota reporting also is highly dependant on the file system architecture, and provides details unique to that file system. GPFS provides the `mmrepquota` command producing:

```
                     Block Limits                    |           File Limits
 Name  type     KB     quota    limit in_doubt grace | files   quota    limit in_doubt grace
 fuser1  USR 17684 41943040 4194304        0  none |    72 1000000 1000000        0  none
 fuser2  USR   180 41943040 4194304        0  none |    32 1000000 1000000        0  none
```

Lustre provides no such reporting functionality. Standard Linux provides the `repquota` command for reporting operations.

```
# repquota /quota
*** Report for user quotas on device /dev/fs1
Block grace time: 7days; Inode grace time: 7day
                 Block limits          File limits
User        used soft hard grace used  soft  hard  grace
-------------------------------------------------------
fuser1   -- 1204    0    0           5     0   0
fuser2   --   10  100  200           9    10  20
```

The more file systems types that are present on a system, the bigger the problem becomes.

Along the same lines is that the actual underlying quotas are per file system and cannot be aggregated across multiple file systems. Users must be granted quotas on each individual file system and managed by that file systems quota utilities.

Statement of Position

Quota utilities should be externalized from the products where each vendor is encouraged to contribute to them to support their file system. Furthermore, each vendor should supply a standardized API call to retrieve or manipulate disk quotas. It is recognized that each file system may need to present details not applicable to other file systems. In this case, the utilities should use extended flags to control the operation.

The application of disk quotas needs to be externalized from file system. While the accumulation of accounting data needs to be within each file system, the enforcement of quotas can be externalized. This would allow for a single disk quota to span multiple file systems regardless of file system type. A kernel module could open the quota file, holding the file descriptor open for a system call to access directly from within the file systems.

SUPPORTING DOCUMENTATION

Many quota operations can be easily externalized. Each of the file systems mentioned already provide an API call that can be used to retrieve or manipulate disk quotas. GPFS provides `gpfs_quotactl()`, Lustre provides `llapi_quotactl()`, and standard Linux provides `quotactl()`. This shows that the underlying interface is already in place, but unique to that file system. Linux already can differentiate between the file system types. The mount table contains the field `mnt_type`, which identifies the underlying file system. So why can't a single form of `quotactl()` which utilizes the `mnt_type` to differentiate the file system types be put in place?

The answer is, it can.

User Quota Report

The first utility to make use of this capability essentially replaces `mmlsquota`, `lfs quota`, and `quota`, with a single utility that can display quota information to the users, regardless of file system type.

In this example, the scratch and scratch2 file systems are Lustre, while project, common, u1, and u2 are GPFS. This utility utilizes `getmntent()` to read the mount table in order to access `mnt_type` which is used to determine the file system type. Then the appropriate `quotactl()` system call is used to access the quota information for the file system. The data is then normalized to a consistent format and presented to the users.

```
Displaying quota usage for user fuser1:
           -------- Space (GB) --------   ----------- Inode -------------
FileSystem  Usage   Quota  InDoubt Grace   Usage    Quota   InDoubt Grace
---------- ------- ------- ------- ----- -------- -------- -------- -----
scratch          3       -       -     -       83        -        -     -
scratch2        24       -       -     -      334        -        -     -
project          0       -       0     -     1944        -        0     -
common           0       -       0     -       11        -        0     -
u1              28      40       0     -     7680  1000000      429     -
u2               0      40       0     -        2  1000000        0     -
```

File System Quota Report

Quota reporting at the file system level is very useful for determining the top consumers of the resources. The issue of different file systems reporting different information can be easily overcome. However, the lack of the capability to simply loop through all quota entries a major obstacle had to be overcome. The solution used was to loop on all users to get their usage information. The downside is that if a user is removed from the system and is consuming resources, it will never be reported.

In a similar manner as in the user quota reporting utility, `statfs()` is used to get the `f_type` of the file system. This is then used to determine the correct `quotactl()` to use for that file system. The user list is obtained simply by looping on `getpwent()`.

```
Filesystem: /scratch2
Report Type: Space
Report Date: Wed Sep  7 07:14:37 2011

          ---- Space (GBs) --- Inode ---
Username  Usage  Quota   Usage    Quota
--------- ------ ------ -------- --------
fuser1     8262      0   663560        0
fuser2     7824      0   225937        0
fuser3     5593      0   216674        0
fuser4     4548      0   111542        0
fuser5     2171      0   436872        0
```

The report can be sorted either by space or by inodes. Reported is a simple and easy to read output that is the same regardless of file system type.

Quotas Spanning File Systems

Enforcing file system quotas external to the file system opens up a flexibility to customize the effects when quotas are reached as well as the opportunity to span file systems. Normally quotas are set up with a soft limit that can be exceeded for some grace period while not exceeding a hard limit. The effect of reaching the hard limit is usually the I/O being aborted with the error `EDQUOT` (quota exceeded). Running a large-scale computation for several days that aborts due to quota limits being reached seemed a bit counter productive, not to mention the loss of valuable computational time. Rather than to terminate the run, a better solution would be to allow it to run to completion while preventing further work from starting. A simple check at job submission and another at job startup can prevent new work from being submitted or started without the loss of computational time.

In addition to the flexibility in how to enforce quota limits, the ability to combine usage information from multiple file systems is enabled allowing for a single quota to span file systems. The process is to simply retrieve the utilization from the desired file systems, accumulate it, and then evaluate it. For batch jobs, this can be performed in submit filters or prologues. This is in production at job submission time. If users are over their quota, they will receive a message:

```
ERROR: your current combined scratch space usage of 6 GBs exceeds
your quota limit of 4 GBs.

You are currently exceeding your disk quota limits. You will
not be able to submit batch jobs until you reduce your usage
to comply with your quota limits.
```

This change has improved the users experience on the system while keeping resource consumption in check.

Externally to the file system, an infrastructure was needed to support the ability to grant a quota that applies to all users, as well as exceptions. Some projects simply require more storage resources than is desired to grant to all users. Having a default quota is easy as it is a value that applies to all users. The challenge was the ability to override this while tracking those with extended quotas.

Another utility was created to manage a data file used to track quota extensions.

```
> chquota -R

           --------- Space Quota --------- --------- Inode Quota ------------
Username Q GigaBs Expiration    Ticket      Inodes   Expiration    Ticket     Filesystem
-------- - ------ ---------- ------------- --------- ---------- ------------- ----------
fuser1   U  10240 01/10/2012 110112-000033   5000000 01/10/2012 110112-000033 /scratch
fuser1   U  10240 11/15/2011 110714-000039         - --/--/----       -       /scratch
```

Not only are the new limits for space and inodes recorded, but also the expiration dates for the extension as well as the problem tracking ticket. From a single report, a clear understanding of all existing quota extensions can be ascertained. A feature of this utility is the ability to automatically remove expired quotas. Each not via cron, the command is run to evaluate all quota extensions and remove any that have expired.

Another feature that is targeted to improving the users experience is the ability to inform if a quota extension is about to expire.

```
chquota: your 6 GB space quota on /scr expires on 09/09/11
(110901-000001)
```

The number in the parenthesis is the trouble ticket number tracking the request. This can be placed in login scripts to inform users each time they login to the system.

CONCLUSIONS

Simplifying disk quotas improves usability, reporting, and the user's experience on the system all while controlling consumption of resources.

File system vendors should be encouraged to align their quota implementations into a single command set of tools that provide a consistent interface, regardless of file system type. Until that happens, centers should adopt a plan to develop such tools as they improve the user's experience. Taking this one step further, all centers should adopt the practice of putting these tools into service creating consistency across centers. Many users perform their calculations at several centers and having a consistent set of tools will enhance their ability to work effectively.

By externalizing disk quota enforcement to job submission, users are forced to keep their resource consumption in check without the risk of losing a run due to quota limits. As a result, the computational resources are much more effective as no time is lost due to calculations being cut short when quota limits are hit.