iVEC Petascale Data Store

The Petabyte Data Store is a scalable storage solution based on Sun's SAM-QFS (Storage and Archive Manager File System & Quick File System) software, and consists of a 70TB of disk cache backed by a 6500 slot automated tape library. The SAM-QFS software automates the migration and retrieval of files across multiple tiers of storage, from the online disk cache, to near-line tape storage, to offline tape archival. This system allows the user to see their complete directory hierarchy and have seamless access to their files. Each file that is placed on the system has a copy written to two separate tapes.

The system is managed via archival policies based on "file size" and "file last access time". The dual write policy ensures data is highly protected and available, and the archival policies allow for exclusion lists for "on disk only" or "always on disk", with high and low watermarks defined to ensure disk space is never completely filled. SAM-QFS uses tar archives when copying data to tape (many files are bundled), providing an easy mechanism for "bare-metal" recovery from tape to any Unix compatible system, with or without SAM-QFS.

SAM-QFS is capable of supporting an almost infinite capacity. By increasing the number of available tape slots in the library and/or by using higher capacity tape media, SAM-QFS can continue to grow.

Currently we are not in a position to create a tertiary offsite copy of all data; in the event of a catastrophic failure destroying both onsite copies, we will be unable to recover the data. A tertiary copy can be created for smaller amounts of non-recreatable data upon request; this would be at an alternate facility or to an alternate tape for offsite storage but this service will be evaluated on a case by case basis.

Please contact help@ivec.org for further information.

Accessing the Data Store

Access to the Petabyte Data Store is via pbstore.ivec.org.
You can access pbstore via scp (WinSCP), rsync, ftp, sftp, ssh, Gridftp.

A project directory is available on *pbstore.ivec.org; /pbstore/<project_name>*
You will need to create a folder for yourself under this location. You will now be able to transfer files to the Data Store.

Please Note: Home directories (/home) are only provided for shell based parameters and should not be used to store any data; home directories on pbstore.ivec.org are NOT backed up!
All data MUST be stored in the appropriate area;
/pbstore/<project_name>/<user_name>

SAM QFS Tools Overview

PATH: /opt/SUNWsamfs/bin

- archive - archive files to tapes
- stage - retrieve files from the tape to disk cache
- sls - similar to “ls”, with more file migration information
- sfind - SAM-QFS “find”
- release - release the file from disk caches
- ssum - set file checksum attributes
- sdu - “du” replacement size of archived directory/file.

SAM QFS File Attributes

```
$ sls -D /pbstore/zz00/rob/csi_muc_en.mp4
/pbstore/zz00/rob/csi_muc_en.mp4:
 mode: -rw-rw-r--  links:   1  owner: mol083    group: zz00
 length: 149921083  admin id:      0  inode:   58711632.1
 offline;  archdone;
 copy 1: ----- Feb  5 04:43     6d45b.2b45d87 li IV1104
 copy 2: ----- Oct 20  2009     43d8c.2935e58 li IV1011
 access:      Nov 25  2009  modification: Apr  8  2008
 changed:     Nov 30  2009  attributes:   Oct 20  2009
 creation:    Oct 20  2009  residence:    Jun  1 20:33
```

mode: The file’s mode and permissions
links: the number of hard links to the file
owner: the owner of the file
group: the group to which the owner belongs.

length: the file’s length in bytes
admin id: the file’s admin ID number
inode: the file’s inode number

damaged: the file is damaged.
offline: the file is offline.
archdone: indicates that the archiver has completed processing the file.

copy: Indicates the position of the file archive and the byte offset for each copy and the tape volume it is on.

access: the time the file was last accessed and modified.
changed: the time the file content and the file’s attributes were last changed.
creation: the time the file was created and became resident in the file system.

Migrating files from tape to the disk cache

NOTE: "-w" Wait for each file to be staged back on-line before completing.

This does not allow the system to sort the stage requests in the order that the files are archived on tape media. In order to get the best performance in this situation, do the following:

Basic Staging

```
$ stage /pbstore/zz00/rob/csi_*
$ stage -w /pbstore/zz00/rob/csi_*
```

Wild card expressions like "*" can be used, as seen above. The command supports directory recursion with the "-r" switch.

Advanced Staging

How many offline files do I have?

```
$ sfind /pbstore/zz00/rob -offline -exec echo {} \; | wc -l
     2
```

How much data is offline?

```
$ (sdu -sk /pbstore/zz00/rob | awk '{print $1/1024}' && du -sk /pbstore/zz00/rob | awk '{print -$1/1024}') | xargs | bc
285.089
```

How to find offline files and stage them to disk?

```
$ sfind /pbstore/zz00/rob -offline -exec stage {} \;
$ sfind /pbstore/zz00/rob -offline -exec stage -w {} \;
```

Interfacing with the Data Store

Protocols & Tools

- ssh – Secure Shell (PUTTY)
- sftp – Secure File Transfer Protocol
- scp – Secure Copy Protocol (WinSCP)
- rsync – synchronizes files and directories
- gsiftp – Grid FTP Protocol

PUTTY - http://www.chiark.greenend.org.uk/~sgtatham/putty/

WinSCP - http://winscp.net/

GridFTP - http://www.globus.org/toolkit/downloads/5.0.1/

Using SSH

NOTE: Replace <username> with your ivec username.

```
$ ssh <username>@pbstore.ivec.org
```

High Performance SSH

If you are running a "High Performance SSH/SCP - HPN-SSH" compiled version of ssh you will be able to take advantage of several performance enhancements. We run an HPN-SSH server by default on pbstore.ivec.org (port 22). *http://www.psc.edu/networking/projects/hpn-ssh/*

none-cipher

The "none-cipher" will encrypt the authentication process (password), but the data stream will not be encrypted. This will speed up data transfers and reduce the CPU load on the server (pbstore).

```
$ ssh -oNoneSwitch=yes -oNoneEnabled=yes <username>@pbstore.ivec.org
```

multi-threaded AES-CTR (MT-AES-CTR)

If using an un-encrypted data stream is not an option you can get increased performance by using multi-threaded AES-CTR (MT-AES-CTR) encryption. NOTE: This will create additional CPU load on the server, usage should be limited.

```
$ ssh -oCipher=aes128-ctr <username>@pbstore.ivec.org
```

or

```
$ ssh -caes128-ctr <username>@pbstore.ivec.org
```

NOTE: There is also a 192 and 256 bit encryption option, just change the 128 above

Using SCP

Standard SCP

```
$ scp -p <file_list> <username>@pbstore.ivec.org:/pbstore/<my_project>/<my_folder>
```

```
usage: scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
        [-l limit] [-o ssh_option] [-P port] [-S program]
        [[user@]host1:]file1 [...] [[user@]host2:]file2
```

Alternate SCP Wrapper (sscp)

The "sscp" command understands offline files contained within the Data Store; it is run from the remote host (ie. cognac.ivec.org). When run, it will stage all offline files to the disk cache before transferring them to the host.

```
$ sscp -p <file_list> <username>@pbstore.ivec.org:/pbstore/<my_project>/<my_folder>
```

NOTE: The "sscp" command requires ssh keys to be created and populated between hosts, this is automatically done when "sscp" is run for the first time. It will ask for a password several times, but all subsequent runs will only ask for a single password.
* Currently this command is only available on cognac.ivec.org.

```
usage: sscp [-prv] [-P port] [[user@]host1:]/file1 [[user@]host2:]/file2

OPTIONS:
        -h This help screen.
        -p Preserves modification times, access times, and modes from the original file.
        -r Recursively copy entire directories.
        -v Verbose mode.
        -P [ port number ] Specifies the port to connect to on the remote host.

NOTE: Multiple file lists must be surrounded by double quotes!
        "[file1] [file2] [file3] [...]"

To report bugs, please email help@ivec.org
```

Using RSYNC

NOTE: The rsync command is very efficient and has many options; make sure you understand what it is going to do before you proceed. Always refer to the "man" page.

```
$ rsync -a --size-only <source_files> <username>@pbstore.ivec.org:/pbstore/<my_project>/<my_folder>
```

Some available switches;

-a
This is equivalent to -rlptgoD. It is a quick way of saying you want recursion and want to preserve almost everything. The only exception to this is if --files-from was specified, in which case -r is not implied.

-v
This option increases the amount of information the daemon logs during its start-up phase. After the client connects, the daemon's verbosity level will be controlled by the options that the client used and the "max verbosity" setting in the module's config section.

-u
This forces rsync to skip any files which exist on the destination and have a modified time that is newer than the source file. (If an existing destination file has a modify time equal to the source file's, it will be updated if the sizes are different.) In the current implementation of --update, a difference of file format between the sender and receiver is always considered to be important enough for an update, no matter what date is on the objects. In other words, if the source has a directory or a symlink where the destination has a file, the transfer would occur regardless of the timestamps. This might change in the future (feel free to comment on this on the mailing list if you have an opinion).

--size-only
Normally rsync will not transfer any files that are already the same size and have the same modification time-stamp. With the --size-only option, files will not be transferred if they have the same size, regardless of timestamp. This is useful when starting to use rsync after using another mirroring system which may not preserve timestamps exactly.

--delete-after
Request that the file-deletions on the receiving side be done after the transfer has completed. This is useful if you are sending new per-directory merge files as a part of the transfer and you want their exclusions to take effect for the delete phase of the current transfer.

-e, --rsh=COMMAND
This option allows you to choose an alternative remote shell program to use for communication between the local and remote copies of rsync. Typically, rsync is configured to use ssh by default, but you may prefer to use rsh on a local network.

Using GridFTP

GridFTP is a relatively mature transfer protocol which has performance features such as support for parallel streams and pipelining for transfers that consist of many small files. It also supports "third party transfers" which means that you can initiate a server to server transfer from your client and the files will not be transferred through your client. In fact, once the transfer is initiated, you can shutdown (close) your client. Further information about GridFTP can be found;
http://www.globus.org/toolkit/docs/5.0/5.0.1/data/gridftp/

By default, GridFTP uses X.509 certificates for authentication. To use GridFTP you will need to have a certificate that has been signed by one of the trusted Certification Authorities. Currently the ARCS CA is the only supported Certification Authority. If you do not currently have a certificate issued by this CA and would like to use GridFTP, you can request a certificate by following the instructions;
http://www.arcs.org.au/index.php/services/security/290-grix

Once you have your certificate in place, you can generate a proxy credential which is necessary for GridFTP transfers with command below. Note that you will need to do this from a machine that has the GridFTP client tools available.

```
$ grid-proxy-init
```

This will prompt you for your private key passphrase and will generate a short-lived credential which can be used for subsequent transfers. You can then perform three different types of transfers: from a local filesystem to a GridFTP server, from a GridFTP server to a local filesystem, and from a GridFTP server to another GridFTP server. An example of each of these situations is given below, using the globus-url-copy command.

```
$ globus-url-copy file:///path-to-file/filename gsiftp://pbstore.ivec.org/destination-path/filename
$ globus-url-copy gsiftp://pbstore.ivec.org/path-to-file/filename file:///destination-path/filename
$ globus-url-copy gsiftp://pbstore.ivec.org/path-to-file/filename gsiftp://other.gridftp.server/destination-path/filename
```

NOTE: The triple slash /// when used with a file URL in the first two commands above.

On installations that support it, you can also use SSH for authentication with GridFTP. Note that the GridFTP servers '"and the client"' must support SSH authentication for you to use this method. This method does not require you to have a certificate and you do not need to issue the grid-proxy-init command before initiating transfers. The following commands illustrate the usage of SSH authenticated transfers.

```
$ globus-url-copy file:///path-to-file/filename sshftp://pbstore.ivec.org/destination-path/filename
$ globus-url-copy sshftp://pbstore.ivec.org/path-to-file/filename file:///destination-path/filename
$ globus-url-copy sshftp://pbstore.ivec.org/path-to-file/filename sshftp://other.gridftp.server/destination-path/filename
```