

A Software Facility for Persistent and Asynchronous Access to HPSS

M.L.Chen, R.D. Burris, D.L. Million, ORNL, Oak Ridge, TN 37831, USA;
 M.K. Gleicher, Gleicher Enterprises, San Diego, CA 92122, USA.

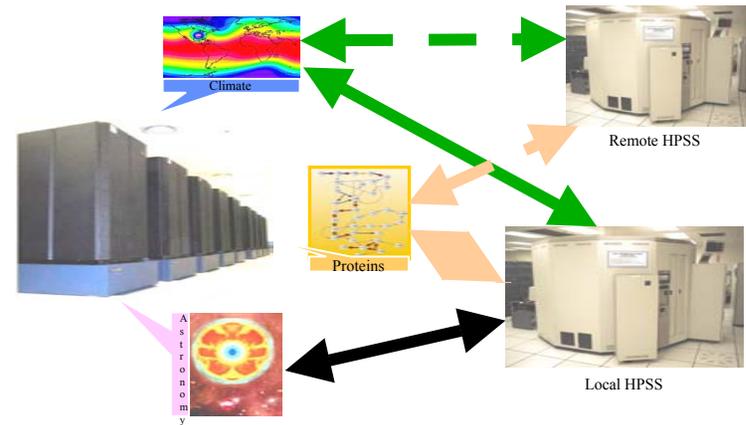
Introduction:

Two major concerns for transferring data between supercomputers and High Performance Storage System (HPSS) installations are HPSS downtime handling and relatively low Wide-Area Network (WAN) transfer rate.

If a file transfer request is made during HPSS downtime, the user gets an error return without any further information. The burden is on the user to find out the HPSS status and to resubmit the request. Furthermore, the nodes of the supercomputer where the files originate may hang because of HPSS unavailability.

If the transfer involves a remote HPSS, poor WAN transfer rates will result in extended periods of workstation or supercomputer resource use. And again, if there is an unexpected network problem, users only get an error return and have to keep resubmitting until success.

HPSSQ, a software facility with decoupling and persistence features, developed in the ORNL/Probe research facility, solves these problems. It saves computer resources significantly and frees users from the necessity of detecting errors and resubmitting.

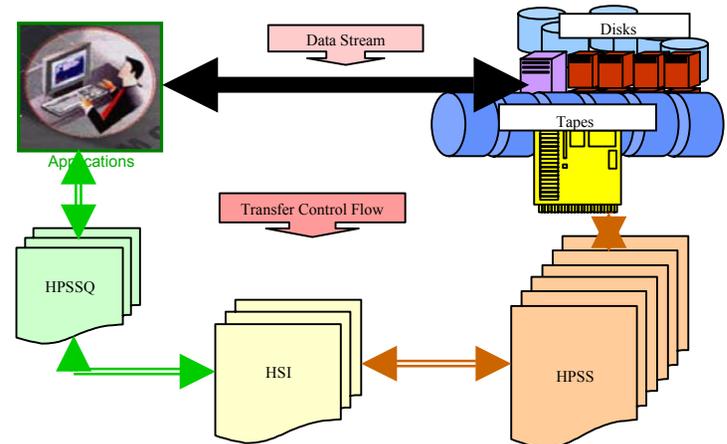


Architecture:

Users submit "hpssq" commands to request data transfers between a computer and an HPSS installation. The HPSSQ client transfers the user's request and environment information to the server, which runs as a daemon on an IBM supercomputer node. The client's workstation, or supercomputer node, is released promptly after receiving the server's acknowledgement.

The server queues requests in a dynamic list. One thread is dispatched for each new transfer. Transfers are implemented using the Hierarchical Storage Interface (HSI), a utility whose features include recursion, wildcarding, and direct transfers between HPSS installations.

A local server can be created upon user's request and runs in interactive mode to transfer data to HPSS installations in remote Kerberos realms.



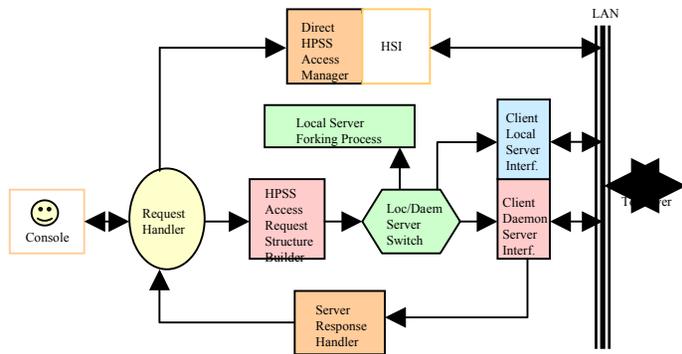
Major Features:

The server is a multithreaded process capable of accepting thousands of requests in seconds. Multiple transfer operations can proceed concurrently.

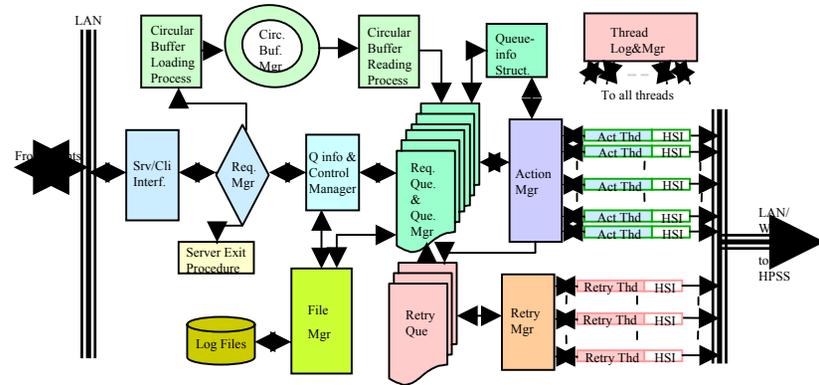
If a request fails, it will be automatically resubmitted at time intervals governed by environment parameters. When a request completes, a status report is emailed to the user. The request queue can be automatically recovered after an incident such as a power failure.

This facility maintains the integrity of all HSI security features. It provides a friendly interface to users, and supports requests submitted in interactive mode or through batch jobs. It unburdens users and saves computer resources significantly.

HPSSQ Client Software Structure



HPSSQ Server Software Structure



HPSSQ User Interface

HPSSQ can run either interactively or in batch jobs. For users' convenience, hpssq command arguments are the same as HSI's wherever it is possible.

Examples of interactive mode with local Kerberos authentication:

1. Online help (commands and simple examples)

```
> hpssq -help
```

2. To submit requests to server queue and to retry in case of failure

To put myFile into myDir of ORNL production HPSS and to use default log file [request record remains after completion]:

```
> hpssq -q "cd myDir; put myFile" [-keepList]
```

To get myStoredFile from myDir of ORNL production HPSS and to set log file path = myLogPath:

```
> hpssq -q "cd myDir; get myStoredFile" -O "myLogPath"
```

3. Queue control and information querying commands

To query current status of submitted requests and to specify log file for full [or request cliId#] information dumping:

```
> hpssq -feb "get status [cliId#]" -O "detailLogPath"
```

To remove records of requests submitted with -keepList flag:

```
> hpssq -feb "rm cliId#1 cliId#2 cliId#3"
```

4. Direct access to HPSS

```
> hpssq -local -q "cd myDir; ls"
```

Examples of batch job with local Kerberos authentication:

Batch job command file example

```
#!/bin/csh -fzv
#@ job_type      = serial
#@ class         = batch
#@ wall_clock_limit = 0:3:00
#@ output        = $(host).$(jobid).out
#@ error         = $(host).$(jobid).out
#@ queue
#
```

```
echo SLOADL_PROCESSOR_LIST
echo SKRB5CCNAME
# write/read a file to/from ORNL production HPSS
hpssq -q "cd Ssmssd; put $file_1" [-O "logpath"]
echo $Status (= returned cliId)
hpssq -q "cd Ssmssd; get $file_2" [-O "logpath"]
echo $Status (= returned cliId)
# Query current status of a submitted request
hpssq -feb "get status [cliId#]" [-O "logpath"]
echo $Status (= current status of request cliId#)
exit
```

Accessing HPSS with authentication in remote Kerberos realms:

1. Online help (commands and simple examples)

```
> hpssq -remote -help
```

2. To submit requests to server queue and to retry in case of failure

To put myFile into myDir of a remote site HPSS and to use default log file [request record remains after completion]:

```
> hpssq -remote -s remSiteName -q "cd myDir; put myFile" [-keepList]
```

To get myStoredFile from myDir of a remote site HPSS and to set log file path = myLogPath:

```
> hpssq -remote -s remSiteName -q "cd myDir; get myStoredFile" -O "myLogPath"
```

3. Queue control and information querying commands

To query status of submitted request and to specify log file for full [or specified request] information dumping:

```
> hpssq -remote -feb "get status [cliId#]" -O "detailLogPath"
```

To remove records of requests submitted with -keepList flag:

```
> hpssq -remote -feb "rm cliId#1 cliId#2 cliId#3"
```

HPSSQ Current Status

HPSSQ has been installed on Eagle (an IBM supercomputer at ORNL) and will be released to users at the end of November 2002.

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.