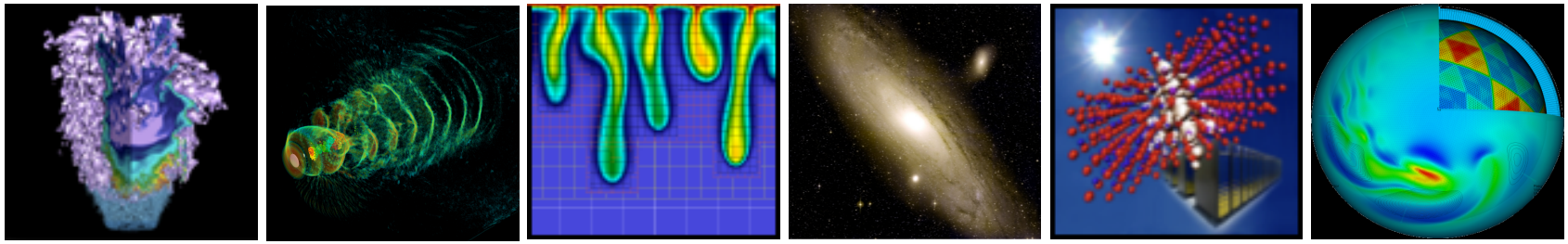


Introduction to Archival Storage at NERSC

Nick Balthaser
NERSC Storage Systems Group
nabalthaser@lbl.gov

NERSC User Training
March 8, 2011

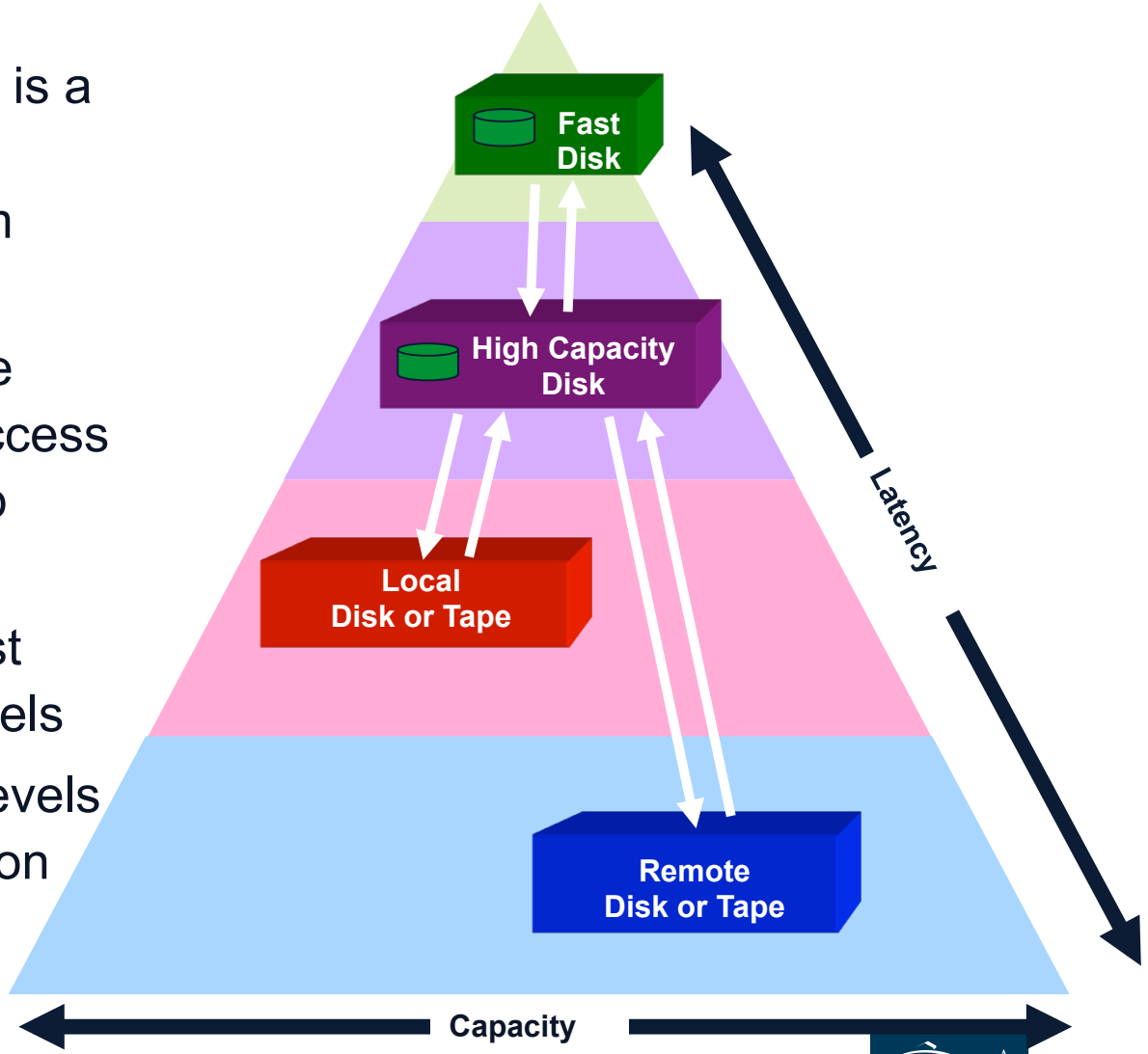
U.S. DEPARTMENT OF ENERGY | Office of Science
NERSC National Energy Research Scientific Computing Center
BERKELEY LAB Lawrence Berkeley National Laboratory

Agenda

- NERSC Archive Technologies Overview
- Use Cases for the Archive
- Authentication
- Storage Clients Available at NERSC
- Avoiding Common Mistakes
- Optimizing Data Storage and Retrieval

NERSC Archive Technologies

- The NERSC archive is a hierarchical storage management system (HSM)
- Highest performance requirements and access characteristics at top level
- Lowest cost, greatest capacity at lower levels
- Migration between levels is automatic, based on policies



Archive Technologies, Continued

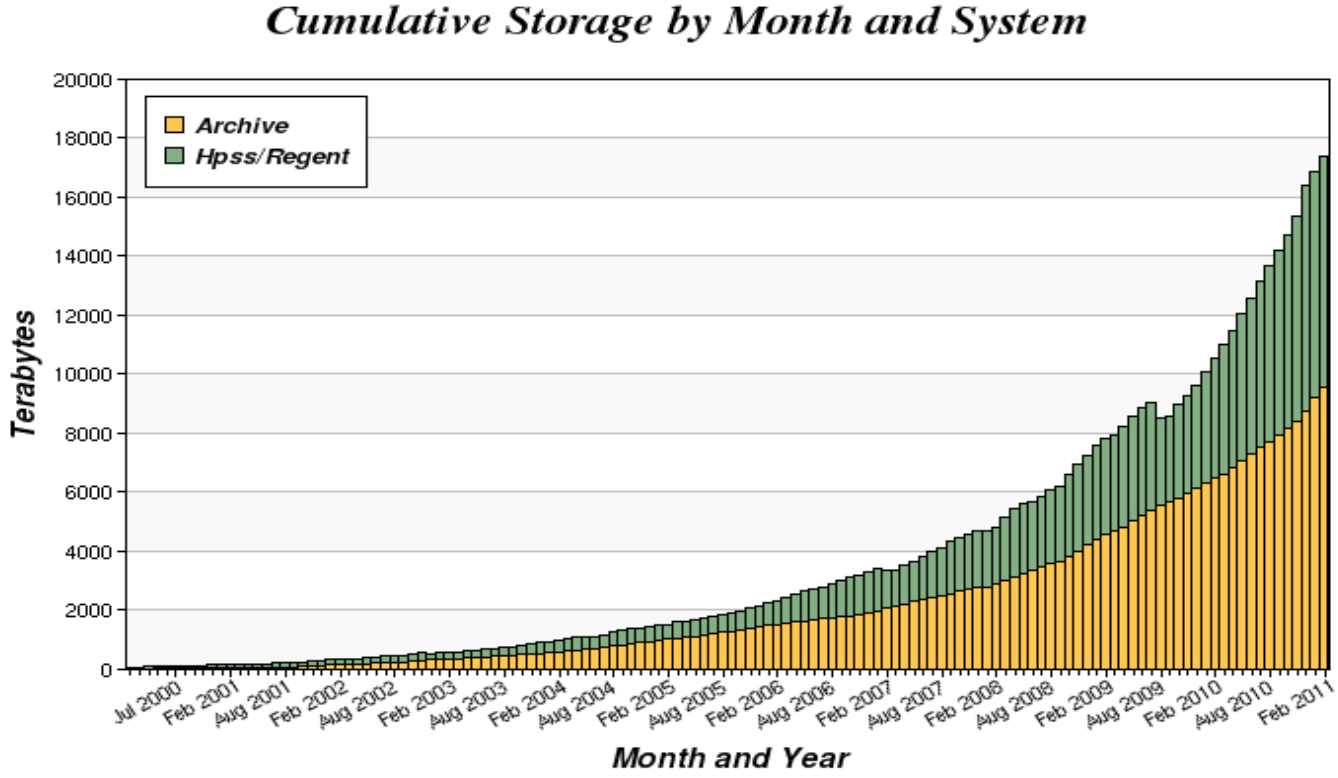
- **NERSC archive implements 2 levels of HSM based on fast access requirement—fast front-end disk cache and enterprise tape**
- **Permanent storage is magnetic tape, disk cache is transient**
  - 10PB data in 100M files written to 26k cartridges
- **Cartridges and tape drives are contained in robotic libraries**
  - Cartridges are loaded/unloaded into tape drives by sophisticated library robotics
- **75 tape drives in user (archive) system**
  - 2 cartridge and drive technologies in use: Oracle T10KB (1TB, high capacity) and 9840D (fast access, 80GB)

Archive Technologies, Continued

- **Front-ending the tape subsystem is 140TB fast-access disk**
  - Data Direct Networks and LSI Logic disk arrays
- **User system has 10 server nodes, IBM p5 running AIX**
  - 9 IO nodes called data movers: read/write to network, disk and tape devices
  - 1 core server: coordinates system activity and serves metadata
- **IBM/DOE HPSS Storage Application**
  - NERSC is a DOE development partner

Archive Technologies, Continued

- **Approximately 50% data growth per year**



- **Transfer rates of over 1GB/sec are possible**

Archive Technologies, Continued…

- **HPSS clients can emulate filesystem qualities**

  - FTP-like interfaces can be deceiving: the archive is backed by tape, robotics, and a single SQL database instance for metadata
  - Operations that would be slow on a filesystem, e.g. lots of random IO, can be impractical on the archive
  - It's important to know how to store and retrieve data efficiently
- **HPSS does not stop users from making mistakes**

  - It is possible to store data in such a way as to make it difficult to retrieve
  - The archive has no batch system. Inefficient use affects others.

Use Cases for the Archive

- **Typical use case: long-term storage of very large raw data sets**
  - Good for incremental processing
- **Long-term storage of result data**
- **Data migration between compute platforms**
- **Backups (e.g. /scratch purges on franklin)**

Authentication

- **NERSC storage uses a token-based authentication method**
  - User places encrypted authentication token in ~/.netrc file at the top level of the home directory on the compute platform
  - Token information is verified in the NERSC LDAP user database
  - All NERSC HPSS clients can use the same token
  - Tokens are username and IP specific—must generate a different token for use offsite

Authentication, Continued

- **Authentication tokens can be generated in 2 ways:**
  - Automatic – NERSC auth service:
    - Log into any NERSC compute platform
    - Type “hsi”
    - Enter NERSC password
  - Manual – https://nim.nersc.gov/ website
    - Under “Actions” dropdown, select “Generate HPSS Token”
    - Copy/paste content into ~/.netrc
    - chmod 600 ~/.netrc
- **Use NIM website to generate token for alternate IP address**

~/.netrc example

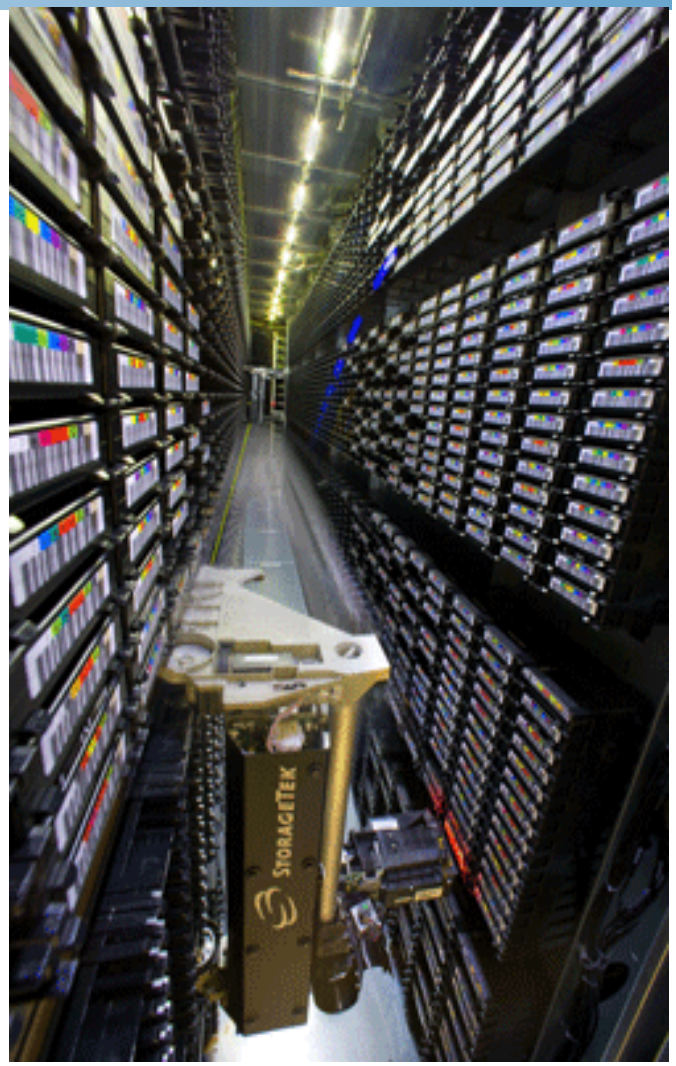
```
machine archive.nersc.gov
login joeuser
password 02UPMUezYJ/Urc7ypflk7M8KHLITsoGN6ZIcfOBdBZBxn+BViShg==


machine ftp.nersc.gov
login anonymous
password joeuser@nersc.gov
```

- **Remember to set permissions on this file**
  - The authentication token is a key, like an ssh key—treat accordingly

HPSS Clients

- **Parallel, threaded, high performance:**
  - HSI
    - Unix shell-like interface
  - HTAR
    - Like Unix tar, for aggregation of small files
  - PFTP
    - Parallel FTP
- **Non-parallel:**
  - FTP
    - Ubiquitous, many free scripting utilities
- **GridFTP interface (garchive)**
  - Connect to other grid-enabled storage systems



HSI

- **Most flexibility, many features and options**
- **Most easily abused**
- **Features:**
  - Parallel, high speed transfers
  - Interactive and non-interactive modes
  - Common shell commands: chown, chmod, ls, rm, etc.
  - Recursion
  - Command-line editing and history
  - Wildcards
- **Connecting to the archive: type “hsi”**

  bash-4.0$ **hsi**

  [Authenticating]

  A:/home/j/joeuser->

Interactive HSI

- **Transfer**

  A:/home/j/joeuser-> **put myfile**

  put 'myfile' : '/home/j/joeuser/myfile' (2097152 bytes, 31445.8 KBS (cos=4))

- **Retrieve**

  A:/home/j/joeuser-> **get myfile**

  get 'myfile' : '/home/j/joeuser/myfile' (2010/12/19 10:26:49 2097152 bytes,
  46436.2 KBS)

- **Full pathname or rename**

  A:/home/j/joeuser-> **put local_file : hpss_file**

  A:/home/j/joeuser-> **get local_file : hpss_file**

- **Wildcards**

  A:/home/j/joeuser-> **prompt**

  prompting turned off

  A:/home/j/joeuser-> **mput .bash***

Non-interactive HSI

- **One-line mode**

  bash-4.0$ **hsi "mkdir mydir; cd mydir; put myfile; ls –l"**

- **Command File**

  bash-4.0$ **cat mycommands.txt**

  put myfile

  ls -l

  quit

  bash-4.0$ **hsi "in mycommands.txt"**

- **Here Document**

  bash-4.0$ **hsi <<EOF**

  put myfile

  ls -l

  quit

  EOF

- **Standard Input**

  bash-4.0$ **echo 'mkdir mydir; cd mydir; put myfile; ls -l; quit' | hsi**

HTAR

- **Similar to Unix tar**
- **Parallel, high speed transfers, like HSI**
- **Recommended utility for archiving small files**
  - Faster/safer than running Unix tar via pipeline
  - Creates index for fast file retrieval
- **HTAR traverses subdirectories to create tar-compatible aggregate file in HPSS**
- **No staging space required**
- **Limitations:**
  - Aggregate file can be any size, recommend 500GB max
  - Aggregates limited to 5M member files
  - Individual HTAR member files max size 64GB
  - 155/100 character prefix/filename limitation

HTAR, Continued

- **Create archive**

  bash-4.0$ **htar –cvf /home/n/nickb/mytarfile.tar ./mydir**

  HTAR: a ./mydir/

  HTAR: a ./mydir/foofile

  HTAR: a /scratch/scratchdirs/nickb/HTAR_CF_CHK_50212_1297706778

  HTAR Create complete for /home/n/nickb/mytarfile.tar. 2,621,442,560 bytes written for 1 member files, max threads: 3 Transfer time: 11.885 seconds (220.566 MB/s)

- **List archive**

  bash-4.0$ **htar –tvf /home/n/nickb/mytarfile.tar**

- **Extract member file(s)**

  bash-4.0$ **htar –xvf /home/n/nickb/mytarfile.tar ./mydir/foofile**

PFTP and FTP

- **PFTP**
  - Standard FTP-like interface distributed with HPSS
  - Implements parallel transfers for performance
  - FTP-compatible syntax
  - Scriptable with some effort (Here doc or command file)

  bash-4.0$ **pftp –i < cmds.txt**

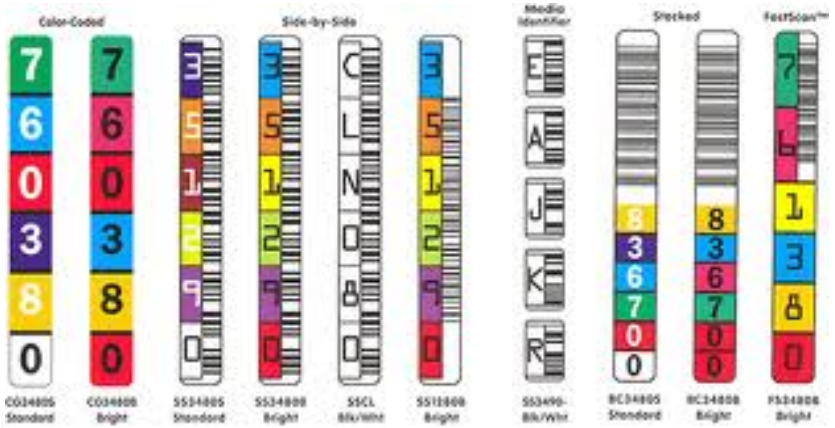
- **FTP**
  - Available everywhere, but non-parallel, low performance
  - Free utilities such as ncftp, curl, and Perl Net::FTP add flexibility for scripting
- **Both interfaces implement *ALLO64 <filesize>* for writing files to the correct COS (more later)**

GridFTP

- **GridFTP uses a certificate based authentication method—not ~/.netrc**
  - Users can use grid credentials to transfer data between other grid-enabled sites
- **GridFTP is the server**
  - Clients include uberftp and globus-url-copy
- **Clients often support user-tunable parameters for WAN transfer**

Avoiding Common Mistakes

- Small files
- Recursive/unordered requests
- Streaming data via Unix pipelines
- Massive pre-staging
- Large directories in HPSS
- Long-running transfers
- Session Limits



Small Files

- **Large tape storage systems do not work well with small files**
  - Tape is sequential media—must be mounted in tape drive and positioned to start of file for reads—SLOW
  - Storing large numbers of small files may spread them across dozens or hundreds of tapes
  - mounting dozens of tapes and then seeking to particular locations on tape can take a long time, and impair usability for others
  - Store small files as aggregates with HTAR
    - Large HTAR aggregates end up on fewer tapes
    - HTAR index speeds member file retrieval
  - Requests for large numbers of small files can be ordered to mitigate performance impact (next section)

Recursive/unordered Requests

- **Using HSI for recursive storage and retrieval is almost always non-optimal**
  - Recursively storing a directory tree is likely to store a lot of small files across a large number of tapes
  - Recursive file retrieval is likely to cause excessive tape mount and positioning activity
    - Not only slow, but ties up system for other users
- **Use HTAR instead of recursive HSI**
- **Order read requests for small files (next section)**

Streaming Data via Unix Pipelines

- **Unix pipelines often used to alleviate the need for spool area for writing large archive files**
  - Pipelines break during transient network issues
  - Pipelines fail to notify HPSS of data size
    - Data may be stored on non-optimal resources, and/or transfers fail
    - Retrieval can be difficult
- **Use global scratch to spool large archive files**
- **Use HTAR if spool space is an issue**
- **If streaming via pipe is unavoidable, use PFTP with *ALLO64 <bytes>* hint**

```
bash-4.0$ pftp archive <<EOF
> bin
> quote allo64 7706750976
> put "| tar cf - ./joeuser" /home/j/joeuser/joeuser.tar
> quit
> EOF
```

Pre-staging

- **HSI allows pre-fetching data from tape to disk cache**
- **With data pre-fetched into cache, hypothetically it should be available quickly for processing**
  - Problems:
    - The disk cache is shared, 1st-come, 1st-served basis
    - If cache is under heavy use by other users, data may be purged before use
    - If data read to cache is larger than cache, it will be purged before use
    - Either situation results in performance penalty as data is read twice from tape
- **Solution: pre-stage large data to global scratch, not disk cache**

Large HPSS Directories

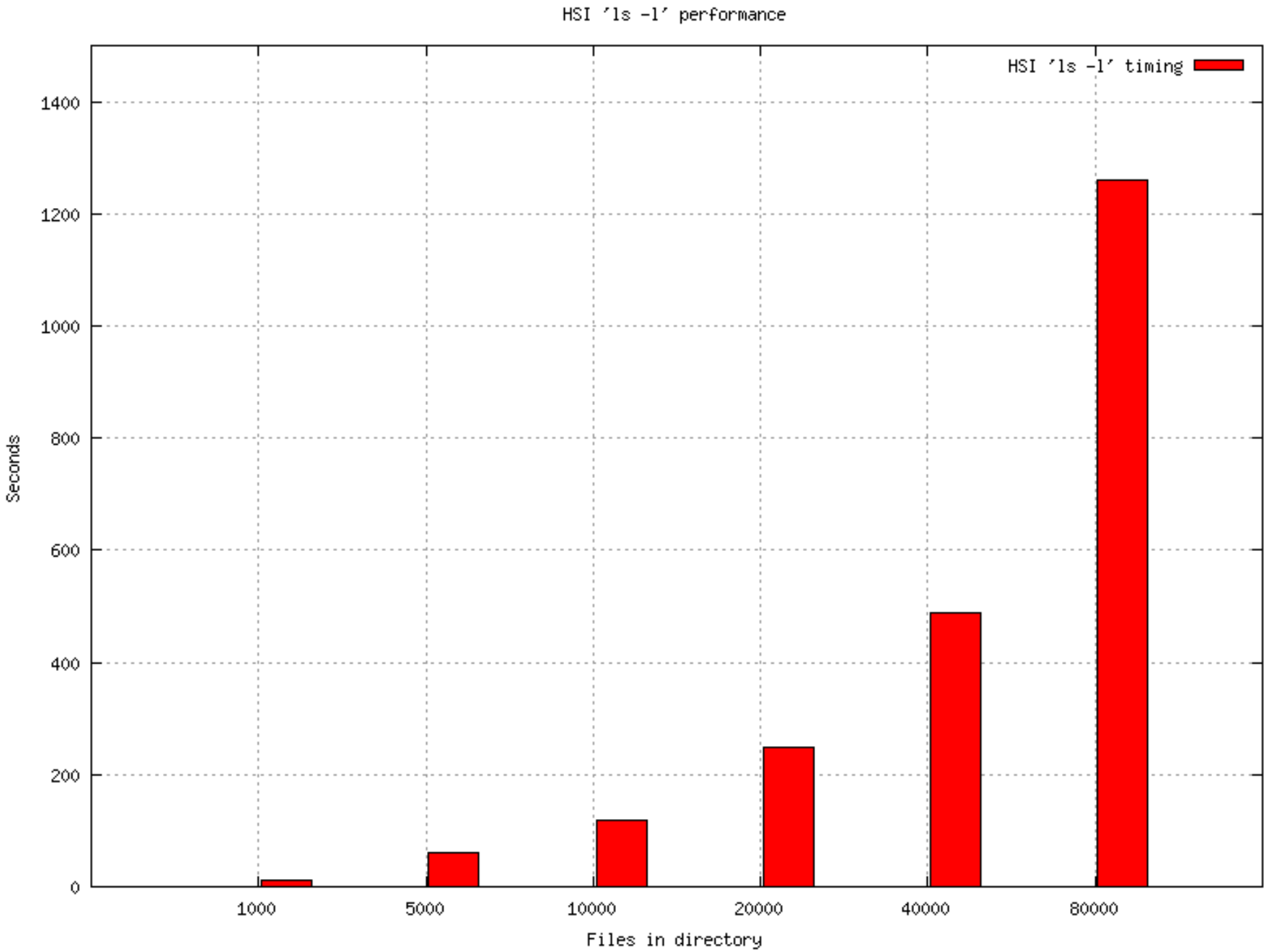
- **Each HPSS system is backed by a single database instance**
  - Every user interaction causes some database activity
  - One of the most database-intensive commands is HSI long file listing, "*ls –l*"
  - Directories containing more than a few thousand files may become difficult to work with interactively

bash-4.0$ **time hsi -q 'ls -l /home/n/nickb/tmp/testing/80k-files/' > /dev/null 2>&1**

real 20m59.374s

user 0m7.156s

sys 0m7.548s


HSI 'ls -l' performance
HSI 'ls -l' timing
Seconds
0
200
400
600
800
1000
1200
1400
1000
5000
10000
20000
40000
80000
Files in directory

Long-running Transfers

- **Can be failure-prone for a variety of reasons**
  - Transient network issues, planned/unplanned maintenance, etc.
- **HSI and PFTP do not have capability to resume interrupted transfers**
- **Data is not completely migrated to tape from disk cache until transfer is completed**
- **Recommend keeping transfers to 24 hrs/ under if possible**

Session Limits

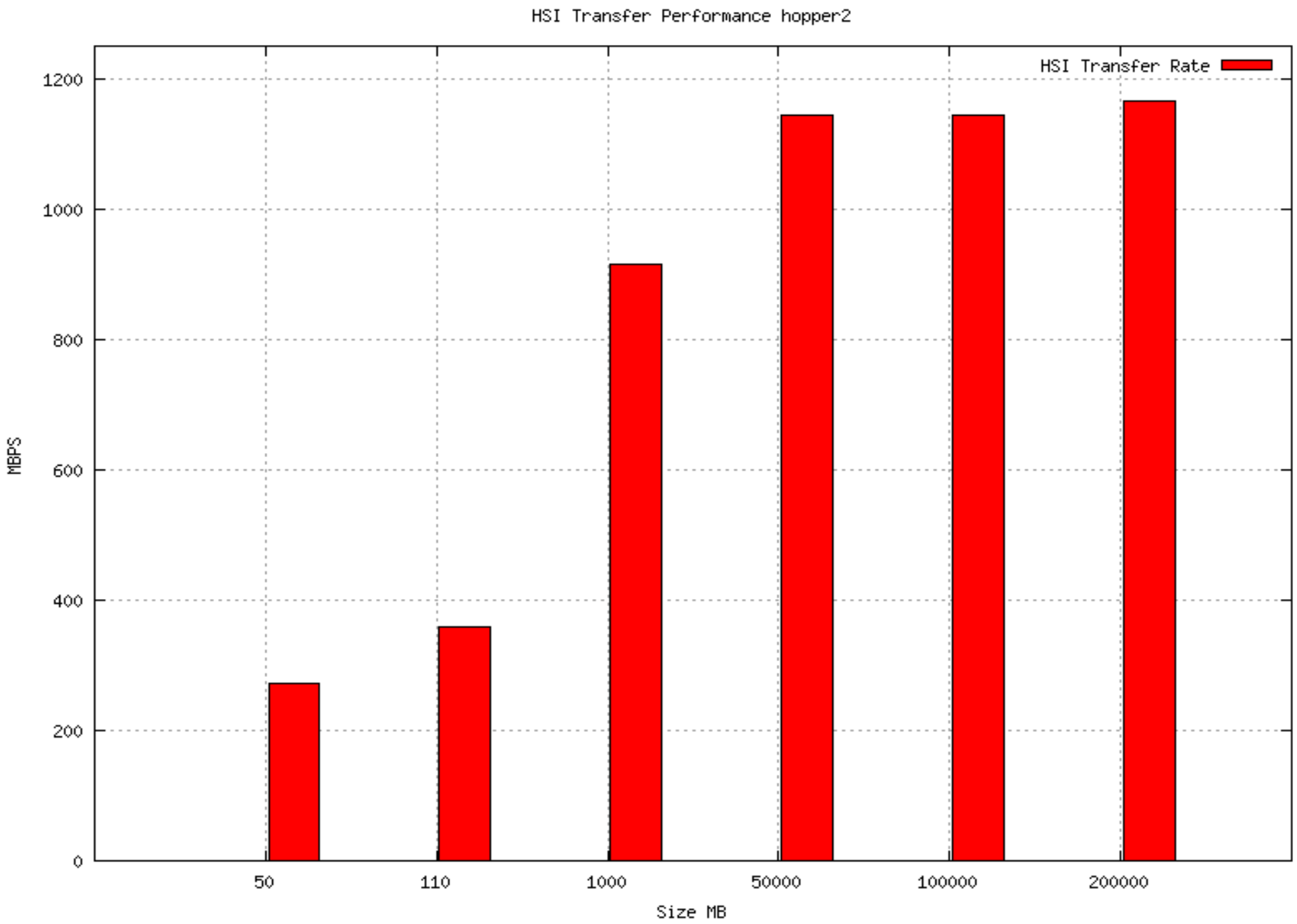
- Users are limited to 15 concurrent sessions
- This number can be temporarily reduced if a user is impacting system usability for others

Optimization

- Guidelines for successful storage
- Guidelines for successful retrieval

Storage Optimization

- **Guidelines for successful storage: store files on as few tapes as possible**
  - Ideally, store (and retrieve) files of optimum size, currently 200 – 300GB
  - Aggregate groups of small files with HTAR (or other aggregation method, e.g. tar, cpio, etc.)
  - Do not use Unix pipelines to store data—stage archive files to spool area first
    - If no spool space use HTAR
    - If pipe is unavoidable, use PFTP with *ALLO64 <filesize>*


HSI Transfer Performance hopper2
HSI Transfer Rate
MBPS
0
200
400
600
800
1000
1200
50
110
1000
50000
100000
200000
Size MB

Retrieval Optimization

- **Successful small file retrieval: minimize tape mounts and positioning**
  - Order requests by cartridge and position
    - Use HSI to list cartridge and tape position
    - We have a sample script to help with this: contact *consult@nersc.gov*

Tape Ordering Example

1. **List files to be retrieved in a text file**

bash-4.0$ **cat files.txt**

/home/j/joeuser/mydir/myfile00

/home/j/joeuser/mydir1/myfile01

/home/j/joeuser/mydir2/myfile02

2. **Generate cartridge and position list with HSI**

bash-4.0$ **for file in `cat files.txt`**

do

echo -n "$file" >> pv.list

hsi -q "ls -X $file" 2>&1 | grep 'PV List' >> pv.list

done

bash-4.0$ **cat pv.list**

/home/j/joeuser/mydir/myfile00 Pos: 3536 PV List: EA854100

Pos:140790 PV List: ED000100

/home/j/joeuser/mydir1/myfile01 Pos: 1 PV List: EM450200

/home/j/joeuser/mydir2/myfile02 Pos: 3 PV List: EM450200

Tape Ordering Example, Continued

3. Generate per-cartridge lists in position order

```
bash-4.0$ for vol in `awk '{print $6}' pv.list | sort -u`
do
   grep $vol pv.list | sort -n +2 -3 | awk '{print $1}' > ${vol}.list
done

bash-4.0$ cat EM450200.list
/home/j/joeuser/mydir1/myfile01
/home/j/joeuser/mydir2/myfile02
/home/j/joeuser/mydir3/myfile03
```

4. Convert per-cartridge lists to HSI command files

```
bash-4.0$ cat EM450200.cmd
get /home/j/joeuser/mydir1/myfile01
get /home/j/joeuser/mydir1/myfile02
get /home/j/joeuser/mydir2/myfile03
```

Tape Ordering Example, Continued

5. Finally, run HSI using command files

```
bash-4.0$ for i in “*.cmd”
do
   hsi –q “in ${i}.cmd”
done
```

Reporting Problems

- **Contact NERSC Consulting**
  - Toll-free 800-666-3772
  - 510-486-8611, #3
  - Email *consult@nersc.gov*.

Further Reading

- **NERSC Website**
  - http://www.nersc.gov/nusers/systems/HPSS/
- **NERSC Grid documentation**
  - http://www.nersc.gov/nusers/services/Grid/grid.php
- **HSI, HTAR, PFTP man pages should be installed on compute platforms**
- **Gleicher Enterprises Online Documentation (HSI, HTAR)**
  - http://www.mgleicher.us/GEL/
- **"HSI Best Practices for NERSC Users" – LBNL publication number pending**

National Energy Research Scientific Computing Center