Access Coordination of Tertiary Storage for High Energy Physics Applications

Luis Bernardo, Arie Shoshani,
Alex Sim, Henrik Nordberg

Scientific Data Management Group
Computing Science Directorate
Lawrence Berkeley National Laboratory

Outline

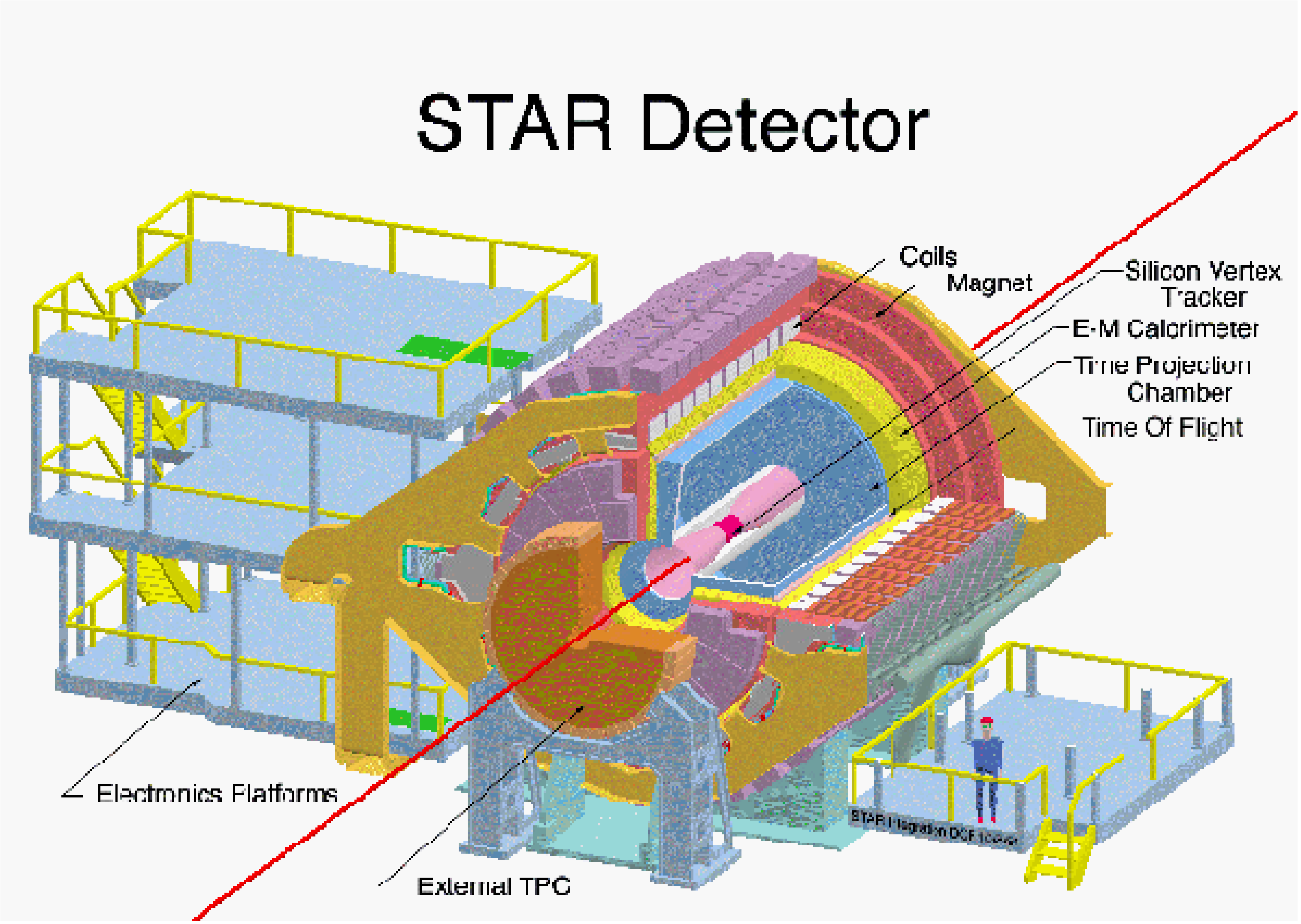
- Short High Energy Physics overview (of data handling problem)
- Description of the Storage Coordination System
- File tracking
- Tertiary storage coordination
    - queuing file transfers
    - file queue management
    - File clustering parameter
    - Transfer rate estimation
    - Query estimation - total time
    - Error handling

Data Volumes for planned HENP experiments

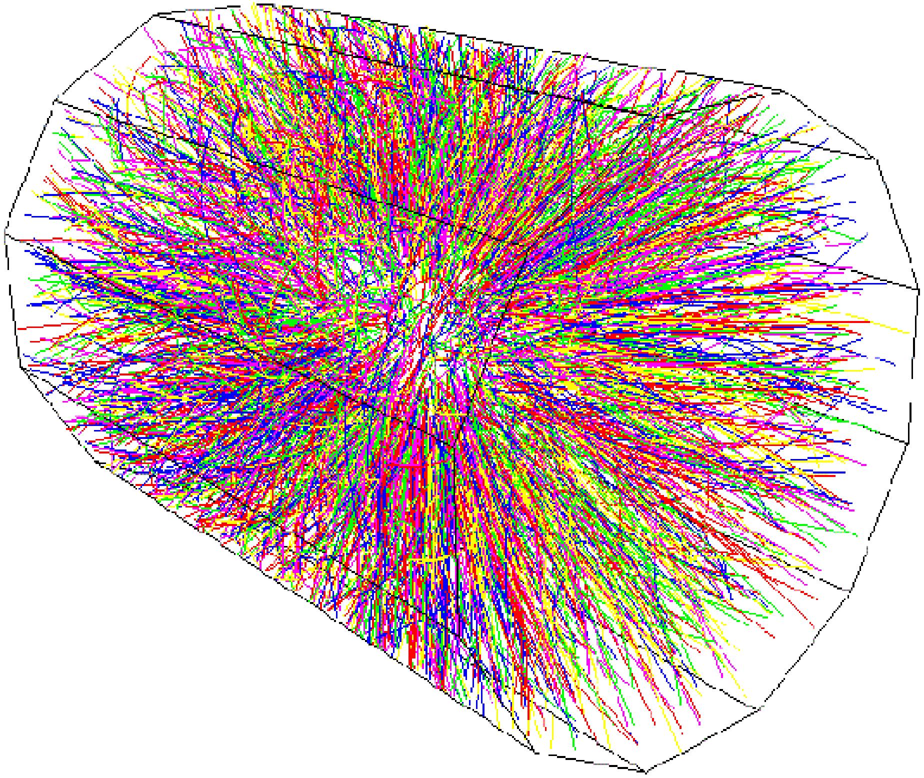
Collaboration	# members /institutions	Date of first data	# events/year	total data volume/year-TB
STAR	350/35	2000	10^7-10^8	300
PHENIX	350/35	2000	10^9	600
BABAR	300/30	1999	10^9	80
CLAS	200/40	1997	10^{10}	300
ATLAS	1200/140	2004	10^9	2000

STAR: Solenoidal Tracker At RHIC
RHIC: Relativistic Heavy Ion Collider

Physical Layout of the Detector



Result of Particle Collision (event)



Typical Scientific Exploration Process

- Generate large amounts of raw data
  - large simulations
  - collect from experiments
- Post-processing of data
  - analyze data (find particles produced, tracks)
  - generate summary data
    - e.g. momentum, no. of pions, transverse energy
    - Number of properties is large (50-100)
- Analyze data
  - use summary data as guide
  - extract subsets from the large dataset
    - Need to access events based on partial properties specification (range queries)
    - e.g. ((0.1 < AVpT < 0.2) ^ (10 < Np < 20)) v (N > 6000)
  - apply analysis code

Size of Data and Access Patterns

- STAR experiment
  - 10^8 events over 3 years
  - 1-10 MB per event: reconstructed data
  - events organized into 0.1 - 1 GB files
  - 10^{15} total size
  - 10^6 files, ~30,000 tapes (30 GB tapes)
- Access patterns
  - Subsets of events are selected by region in high-dimensional property space for analysis
  - 10,000 - 50,000 out of total of 10^8
  - Data is randomly scattered all over the tapes
- Goal: Optimize access from tape systems

EXAMPLE OF EVENT PROPERTY VALUES

I event 1
I N(1) 9965
I N(2) 1192
I N(3) 1704
I Npip(1) 2443
I Npip(2) 551
I Npip(3) 426
I Npim(1) 2480
I Npim(2) 541
I Npim(3) 382
I Nkp(1) 229
I Nkp(2) 30
I Nkp(3) 50
I Nkm(1) 209
I Nkm(2) 23
I Nkm(3) 32
I Np(1) 255
I Np(2) 34

I Np(3) 24
I Npbar(1) 94
I Npbar(2) 12
I Npbar(3) 24
I NSEC(1) 15607
I NSEC(2) 1342
I NSECpip(1) 638
I NSECpip(2) 191
I NSECpim(1) 728
I NSECpim(2) 206
I NSECkp(1) 3
I NSECkp(2) 0
I NSECkm(1) 0
I NSECkm(2) 0
I NSECp(1) 524
I NSECp(2) 244
I NSECpbar(1) 41
I NSECpbar(2) 8

R AVpT(1) 0.325951
R AVpT(2) 0.402098
R AVpTpip(1) 0.300771
R AVpTpip(2) 0.379093
R AVpTpim(1) 0.298997
R AVpTpim(2) 0.375859
R AVpTkp(1) 0.421875
R AVpTkp(2) 0.564385
R AVpTkm(1) 0.435554
R AVpTkm(2) 0.663398
R AVpTp(1) 0.651253
R AVpTp(2) 0.777526
R AVpTpbar(1) 0.399824
R AVpTpbar(2) 0.690237
I NHIGHpT(1) 205
I NHIGHpT(2) 7
I NHIGHpT(3) 1
I NHIGHpT(4) 0
I NHIGHpT(5) 0

54 Properties, as many as 10^8 events

Opportunities for optimization

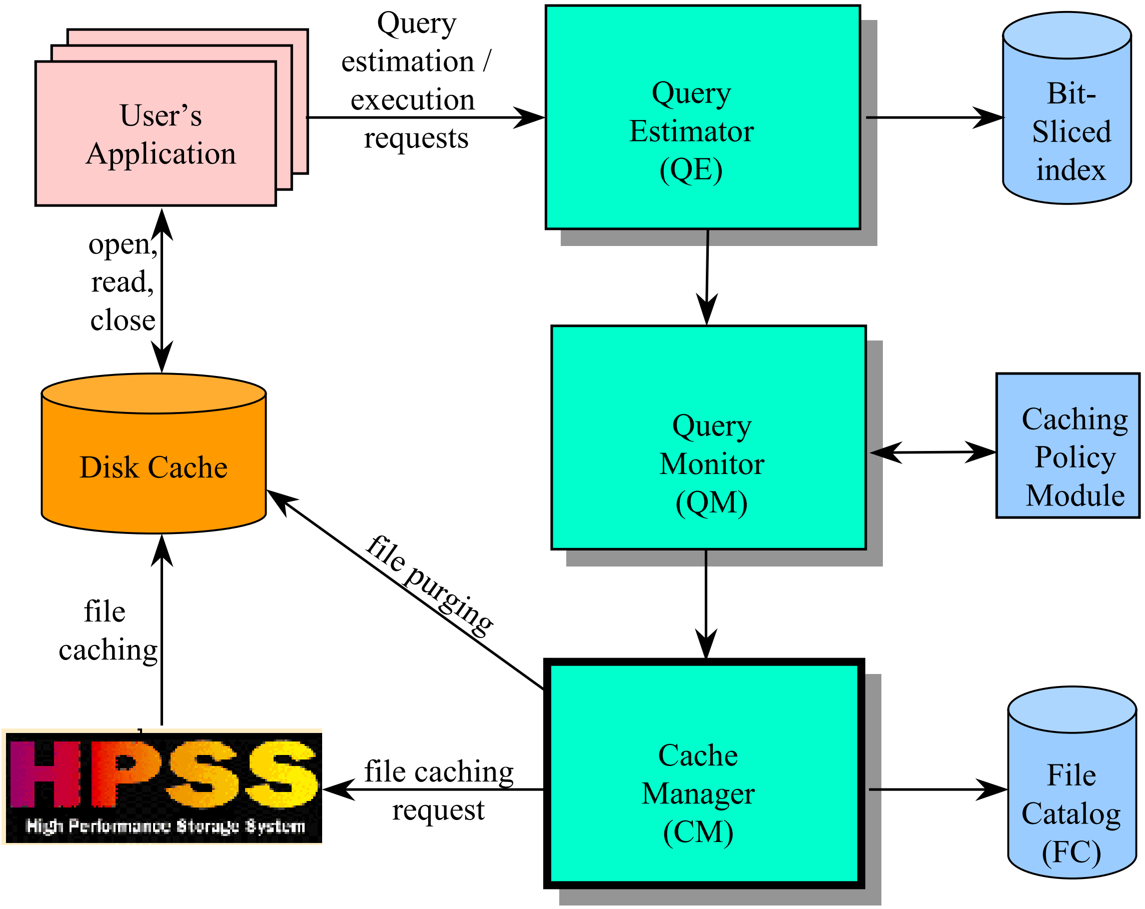
- Prevent / eliminate unwanted queries
  => query estimation (fast estimation index)

- Read only events qualified for a query from a file (avoid reading irrelevant events)
  => exact index over all properties

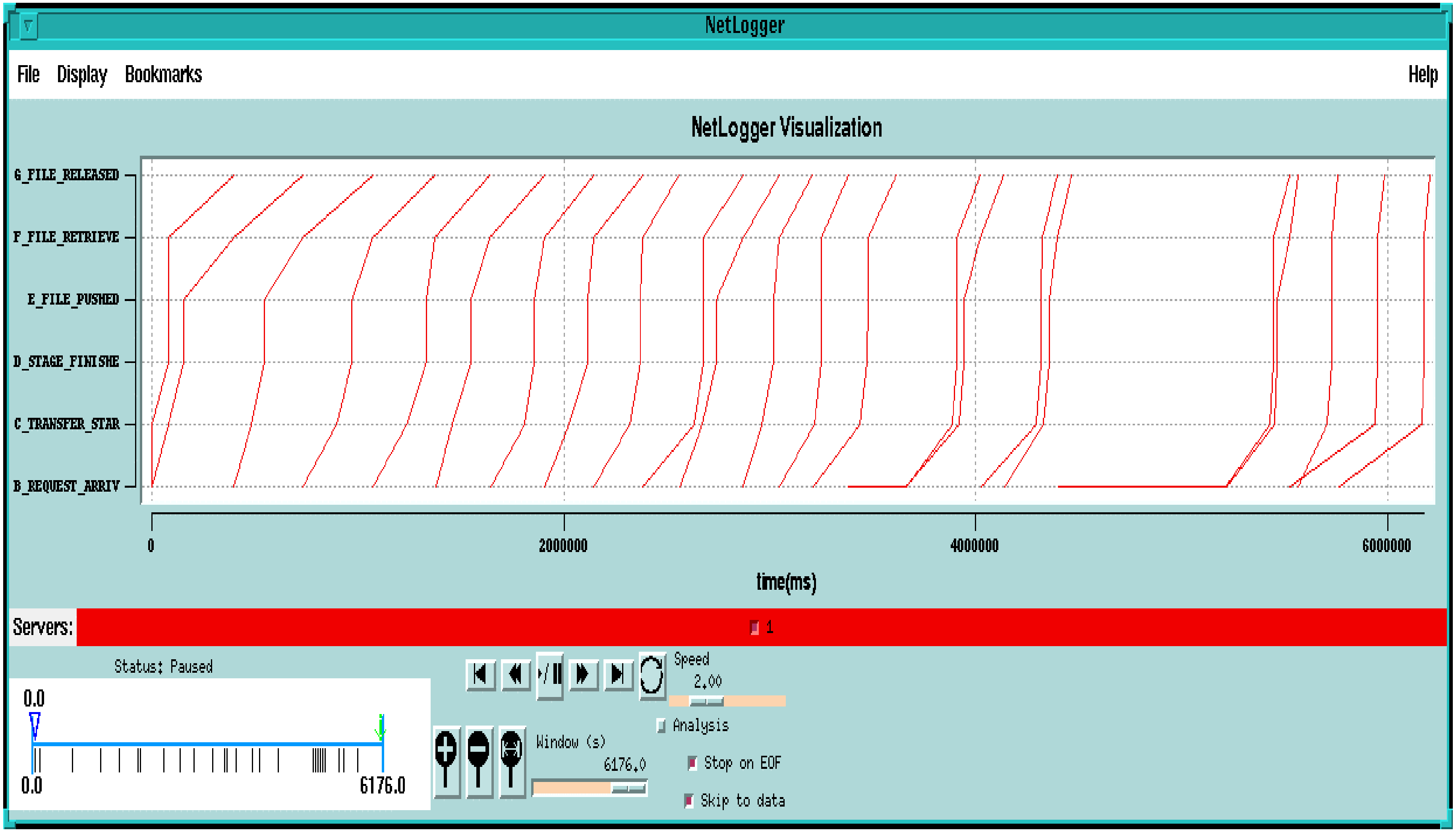
- Share files brought into cache by multiple queries
  => look ahead for files needed and cache management

- Read files from same tape when possible
  => coordinating file access from tape

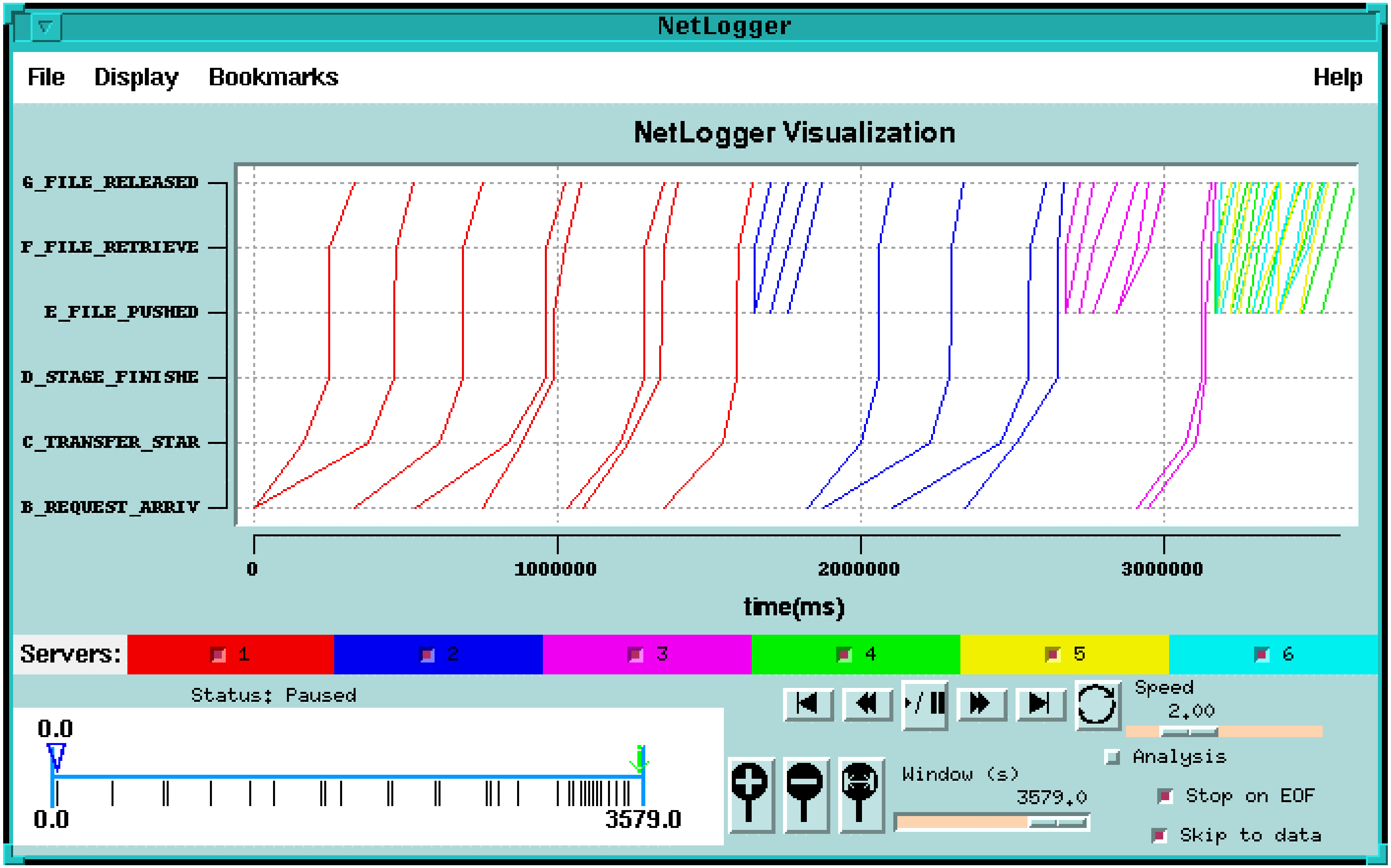
The Storage Access Coordination System (STACS)



File Tracking (1)



File Tracking (2)



Status: Paused

Speed 2.00

Window (s) 3579.0

☐ Analysis

☒ Stop on EOF

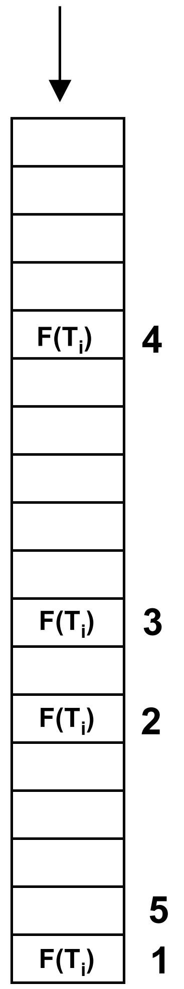
☒ Skip to data

Queuing File Transfers

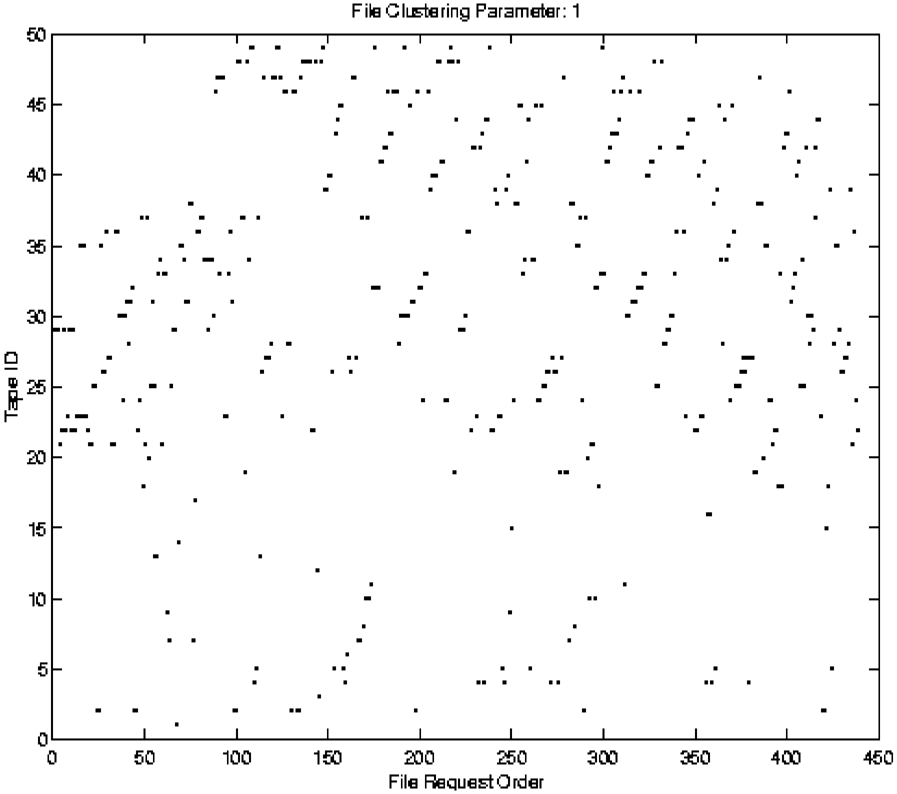
- Number of PFTPs to HPSS are limited
  - — limit set by a parameter - NoPFTP
  - — parameter can be changed dynamically
- CM is multi-threaded
  - — issues and monitors multiple PFTPs in parallel
- All requests beyond PFTP limit are queued
- File Catalog used to provide for each file
  - — HPSS path/file_name
  - — Disk cache path/file_name
  - — File size
  - — tape ID

File Queue Management

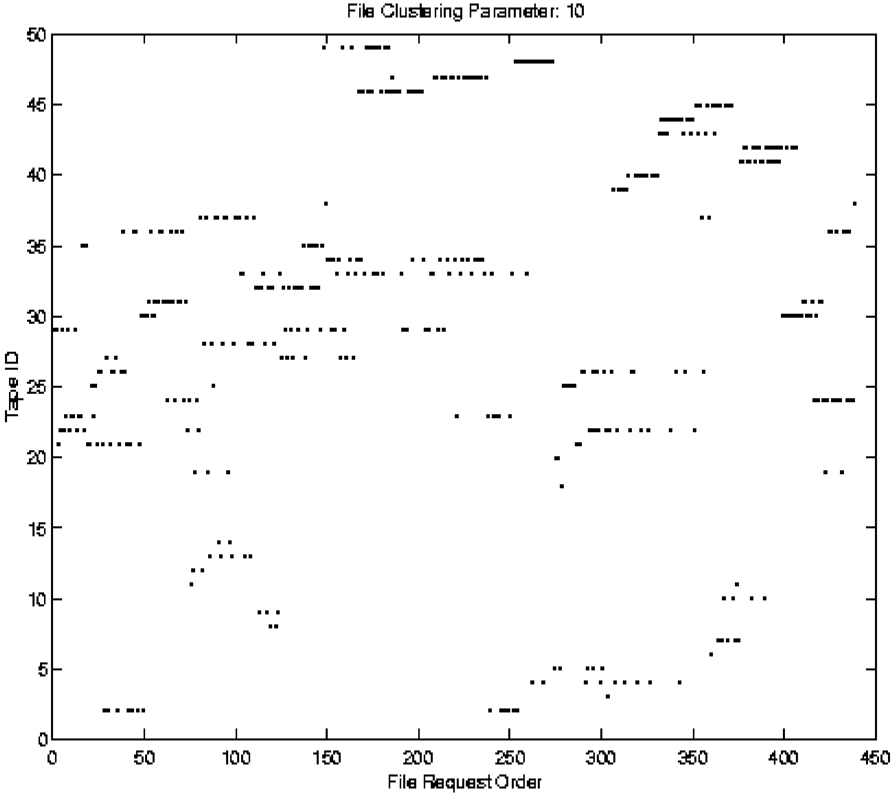
- Goal
  - minimize tape mounts
  - still respect the order of requests
  - do not postpone unpopular tapes forever
- File clustering parameter - FCP
  - If the file at top of queue is in Tape$_i$ and FCP > 1 (e.g. 5) then up to 4 files from Tape$_i$ will be selected to be transferred next
  - then, go back to file at top of queue
- Parameter can be set dynamically



File Caching Order for different File Clustering Parameters



File Clustering Parameter = 1

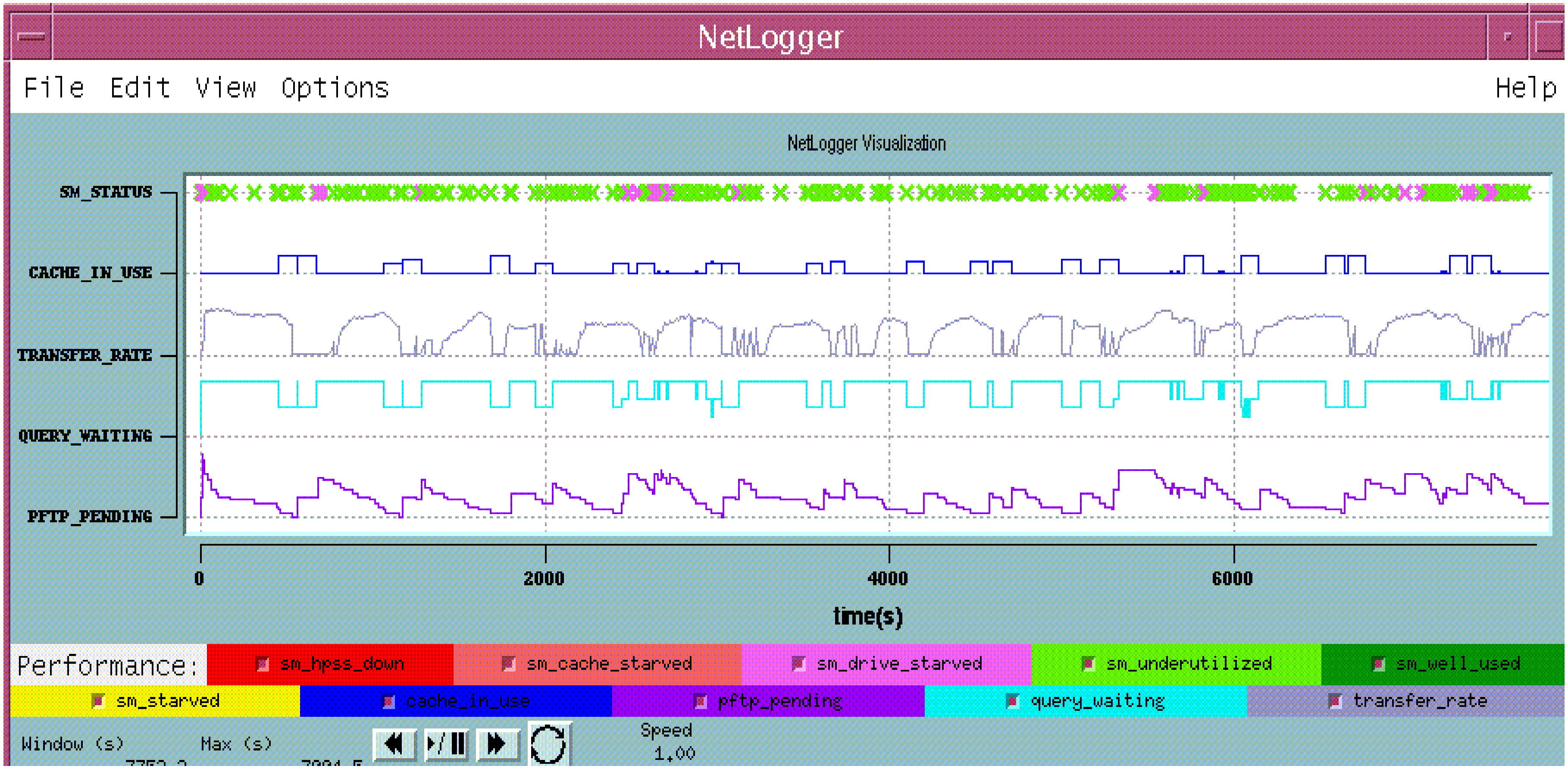


File Clustering Parameter = 10

Transfer Rate (Tr) Estimates

- Need Tr to estimate total time of a query
- Tr is average over recent file transfers from the time PFTP request is made to the time transfer completes. This includes:
  - mount time, seek time, read to HPSS Raid, transfer to local cache over network
- For dynamic network speed estimate
  - check total bytes for all file being transferred over small intervals (e.g. 15 sec)
  - calculate moving average over n intervals (e.g. 10 intervals)
- Using this, actual time in HPSS can be estimated

Dynamic Display of Various Measurements



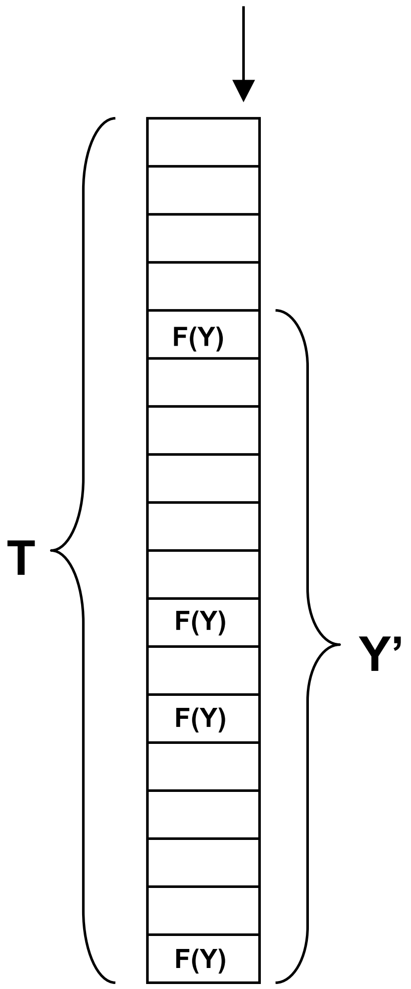
Window (s) 7752.2

Max (s) 7904.5

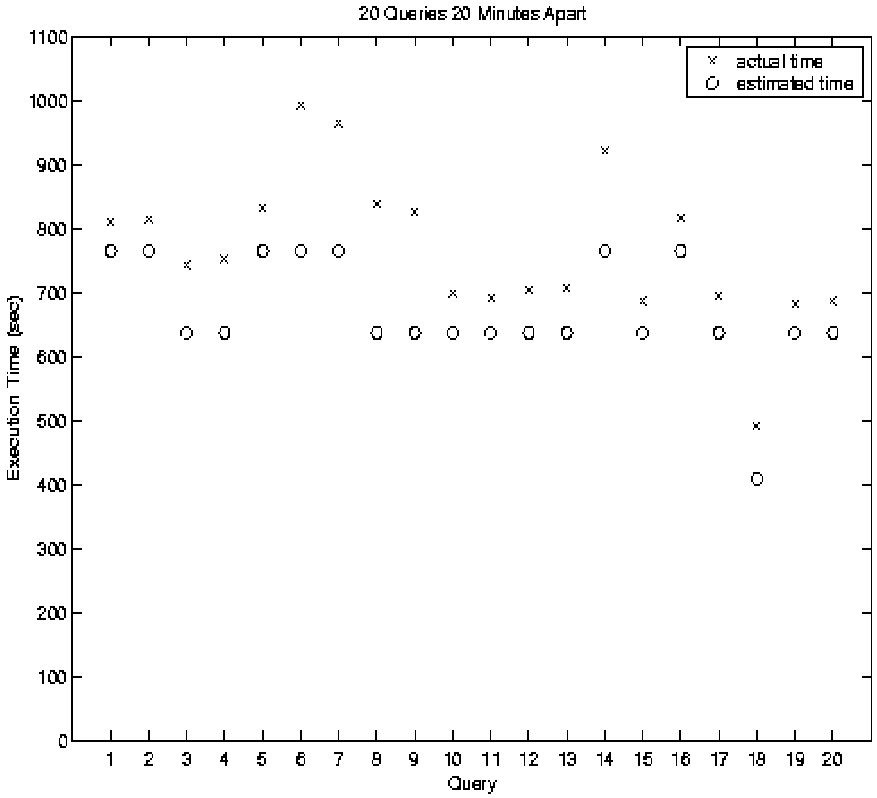
Speed 1.00

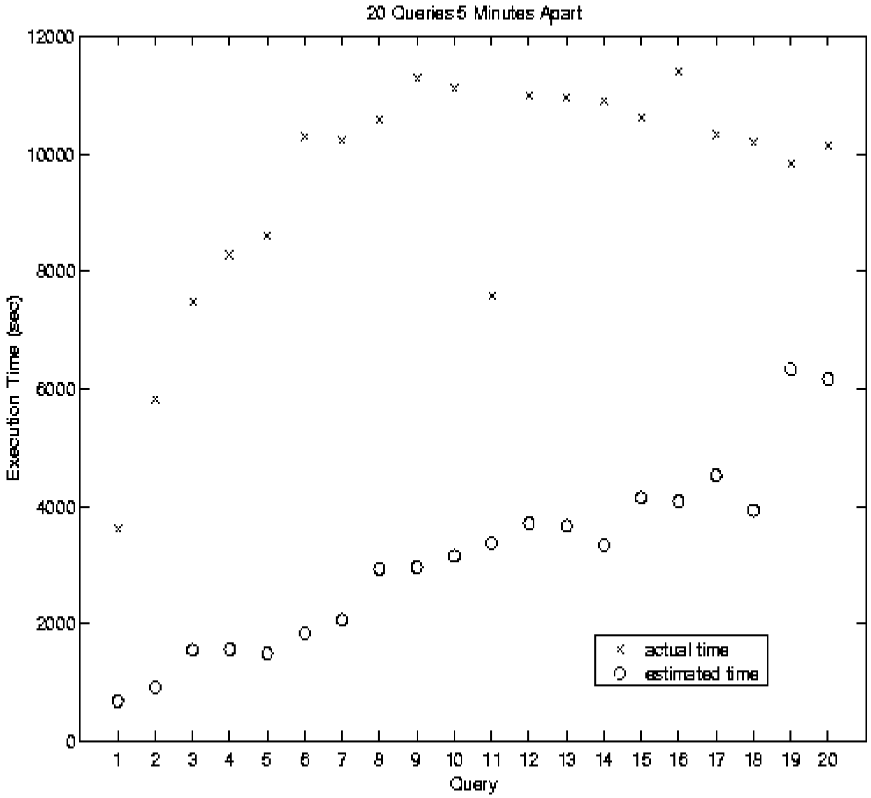
Query Estimate

- Given: transfer rate Tr.
- Given a query for which:
  - X files are in cache
  - Y files are in the queue
  - Z files are not scheduled yet
- Let s(file_set) be the total byte size of all files in file_set
- If Z = 0, then
  - QuEst = s(Y')/Tr
- If Z ≠ 0, then
  - QuEst = (s(T)+ q.s(Z))/Tr
    where q is the number of active queries



Reason for q.s(Z)





20 Queries of length ~20 minutes launched 20 minutes apart

Estimate pretty close

20 Queries of length ~20 minutes launched 5 minutes apart

Estimate bad - request accumulate in queue

Error Handling

- 5 generic errors
  - — file not found
    - return error to caller
  - — limit PFTP reached
    - can’t login
    - re-queue request, try later (1-2 min)
  - — HPSS error (I/O, device busy)
    - remove part of file from cache, re-queue
    - try n times (e.g. 3), then return error “transfer_failed”
  - — HPSS down
    - re-queue request, try repeatedly till successful
    - respond to File_status request with “HPSS_down”

Summary

- HPSS Resource Manager
  - — insulates applications from transient HPSS and network errors
  - — limits concurrent PFTPs to HPSS
  - — manages queue to minimize tape mounts
  - — provides file/query time estimates
  - — handles errors in a generic way
- Same API can be used for any MSS, such as Unitree, Enstore, etc.