A High Performance Hierarchical Storage Management System For the Canadian Tier-1 Centre @ TRIUMF

Denice Deatrich, Simon Xinli Liu, Reda Tafirout

CHEP 09, Prague

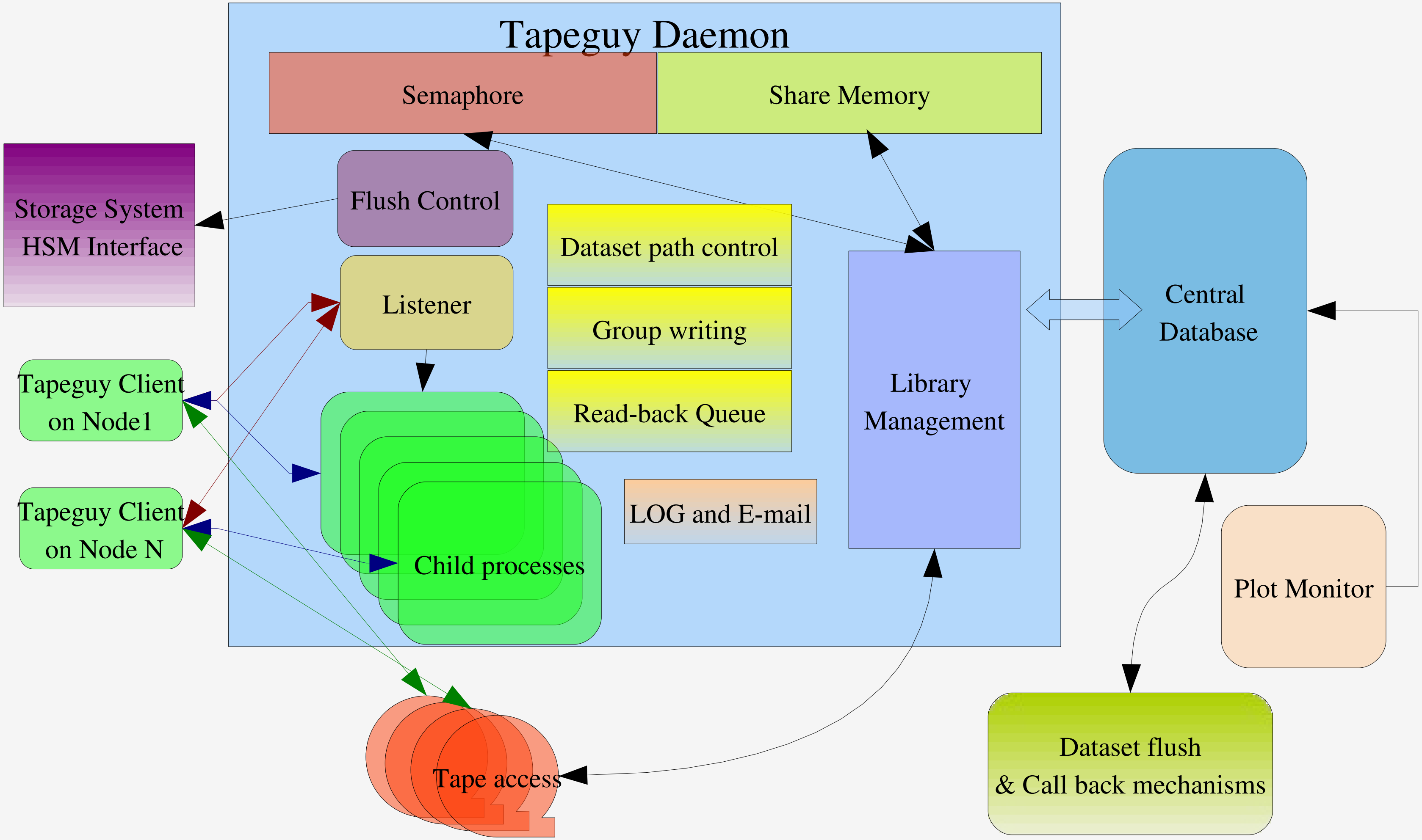
Outline

- Logical Architecture and Implementation
- Tape Library Management
- Group writing
- Read-back Queuing
- I/O balancing
- Performance and scalability
- Virtual Tape Library

Introduction

- **Background and Requirements**
  - **The ATLAS experiment at the LHC will generate a very large amount of data when recording proton-proton collision events**
  - **The great majority of the data will be stored on tertiary storage (tape systems) at CERN (Tier-0) and the Tier-1 centers**
- **An efficient HSM system is crucial for Tier-1 centers for the ATLAS computing model to function properly:**
  - **File grouping and tape grouping**
  - **Reordering of requests & prioritization**
  - **Scalability**
- **Known proprietary HSMs are not ideal**
- **Tapeguy: high performance non-proprietary HSM system**
  - **Designed and developed at TRIUMF**
  - **No proprietary library or drive access code is used**
  - **In production at the Tier-1 (interfaced with dCache)**

Logical Architecture

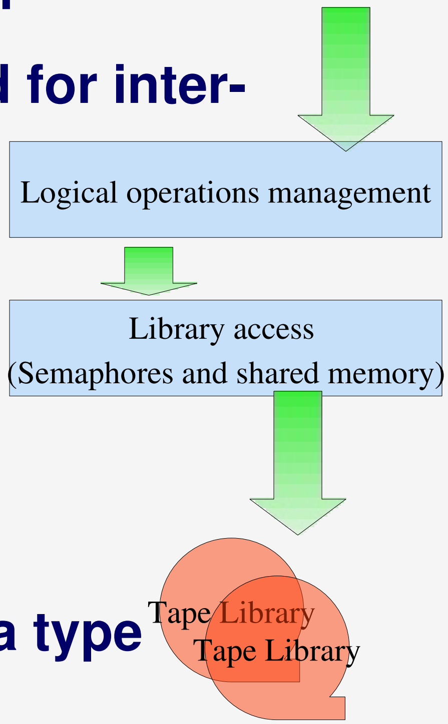


Tapeguy Implementation

- Implemented in Perl
  - TCP socket-based server daemon (can be easily expanded to multiple daemons if needed)
- Persistent naming of tape devices across all nodes
- MySQL is used as the database back-end
- Log and e-mail alert
- Requires ssh access to the dCache Admin Interface

Tape Library Management

- **Design**
  - **Supports for multiple libraries**
  - **Standard primitive utilities- mtx, mt and tar**
  - **Semaphores and shared memory are used for inter-process communication**
- **Logical operation management**
  - **Logical drive operation type**
    'write','read','mixed' (on demand)
  - **LRU algorithm for tape drive usage**
  - **Water mark protection for each tape media type**
- **Logs: tape library and drive activities**



How Tapeguy deals with HSM

For high HSM efficiency:

prioritization

reordering

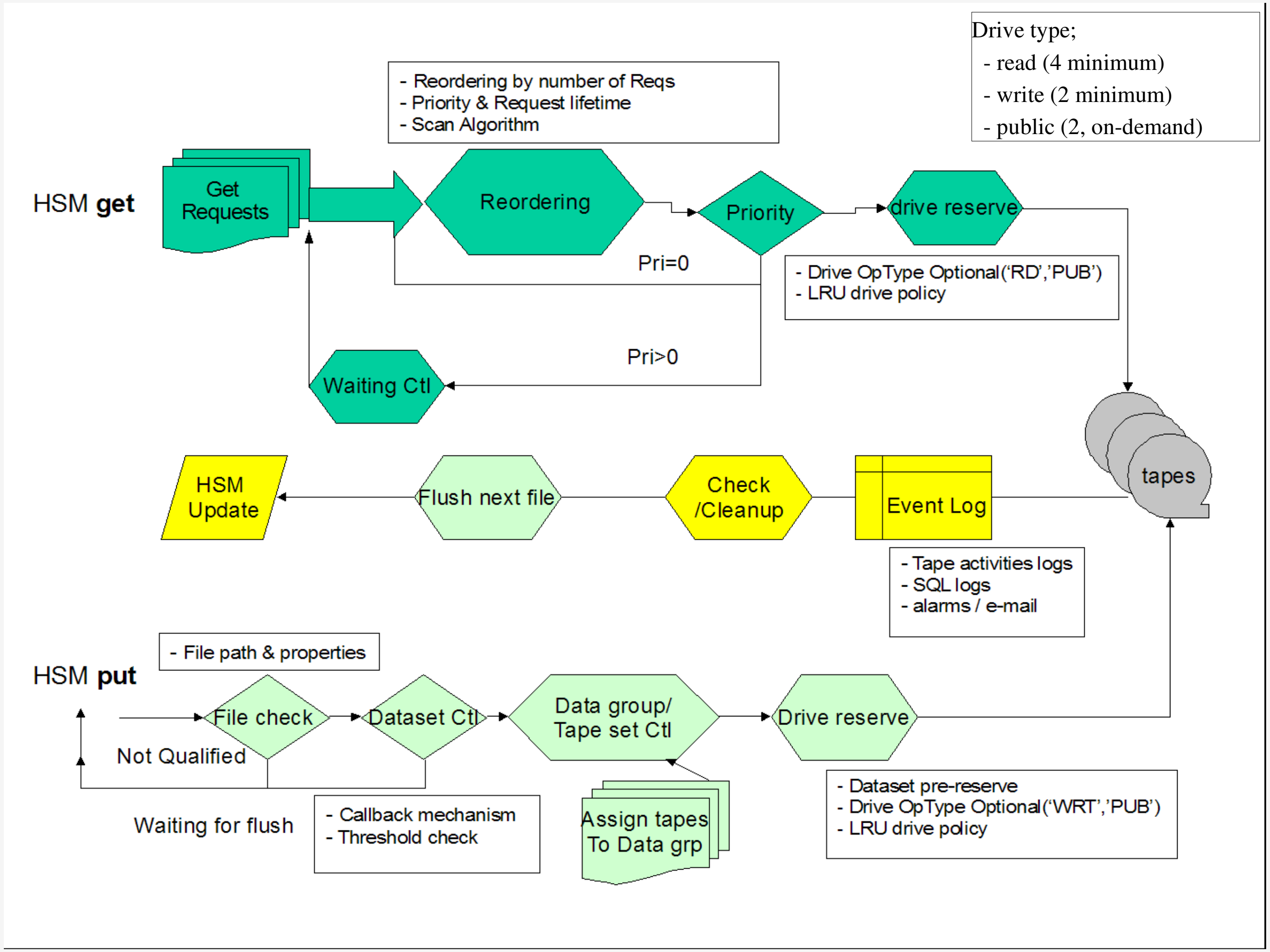
file grouping

tape grouping

minimizes tape mounts

maximizes reads per mount

Write pools set to rdonly while writing to tape on a rotation basis (to avoid contention on the same LUN during flush)



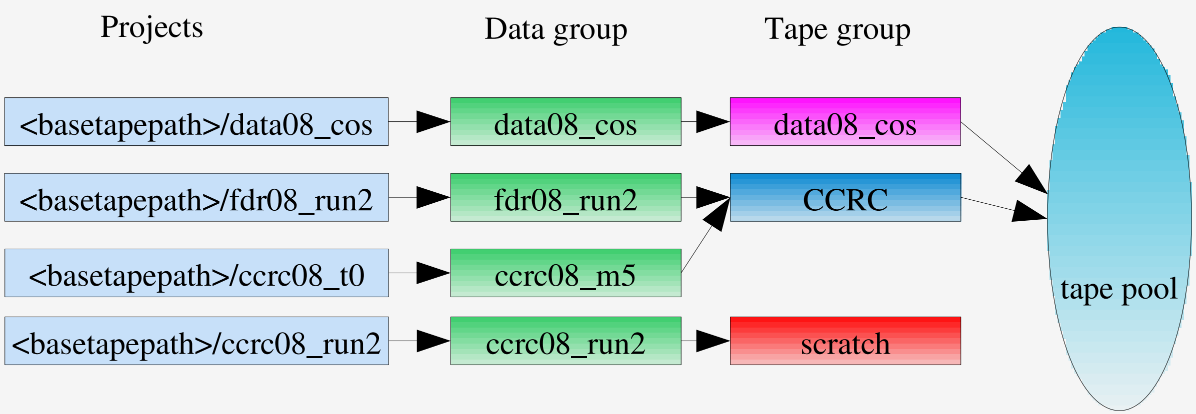
Group Writing (I)

- Data is organized on tape using the pathname
  - Dataset path control groups each file by physical pathname
  - Automatically creates datagroup for the first pathname. A tapegroup for it is created as well
  - A tape group can be manually shared by small data groups
  - Tapes are automatically allocated to tape group on demand

ATLAS path structure: <basetapepath>/project/datatype/datasetname/filename

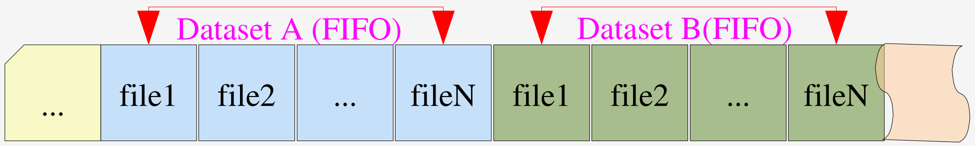
	Dir level	dir tree
	1	<basetapepath>
Project	2	data
	2	fdr
	2	ccrc
	2	prod
Datatype	3	RAW
	3	ESD
	3	HITS

EXAMPLE



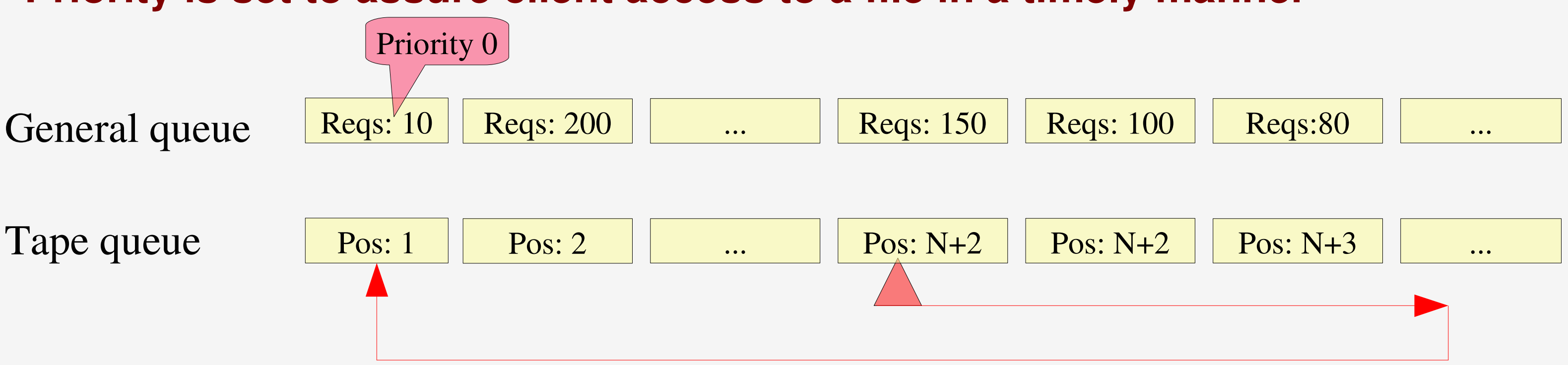
Group Writing (II)

- 'Dataset flush control' packs files in one dataset physically close on tapes
  - A file is registered in its dataset when it first comes to Tapeguy, it returns a code to HSM interface which lets the file stay in the disk buffer
  - Three thresholds for 'dataset flush control'
    - Lifetime of dataset processing (15 hours)
    - The data volume (800GB, A tape capacity)
    - Dashboard callbacks (not reliable)
  - FIFO flush order is used for dataset flushing



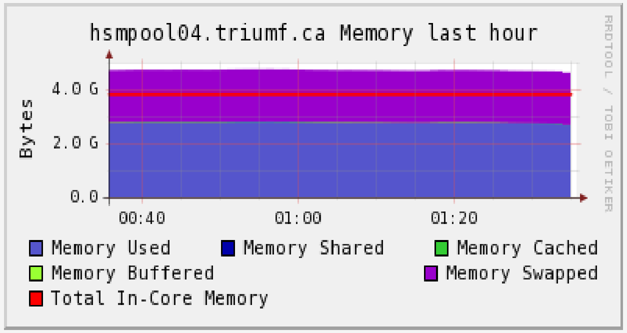
Read-back Queuing

- Two kinds of queues: 'general queue' and 'tape queue'; both are dynamic
- 'General queue' – for deciding the tape picking order
  - Picks the tape that has the most requests at that moment, providing high throughput and minimizing tape mounts
  - Remains open until all requests are served
- 'tape queue' – for reordering requests by file position on tape
  - Elevator algorithm is used to reorder files requests according to the file position on that tape and the tape head position
  - Every request receives its estimated wait time at that moment
- Priority is set to assure client access to a file in a timely manner



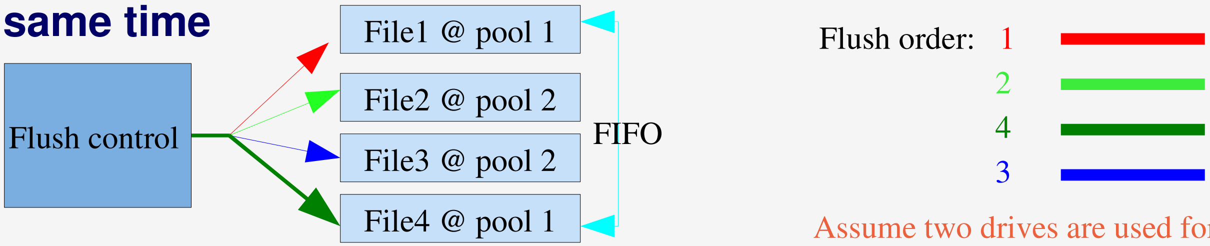
I/O balancing

- **Why I/O balancing ?**
  - **A few LTO-4 drives can easily saturate host I/O bandwidth**
  - **The tape speed adjusts to the available data stream within minimum and maximum streaming speeds**



Worse case: 6 drives reading on one host
At the same time, took 2.5 hours

- **I/O balancing at writing**
  - **Balances the tape I/O across multiple pools when flushing files to tape**
  - **'pool rotation' policy is used when disk and tape writing occur at the same time**



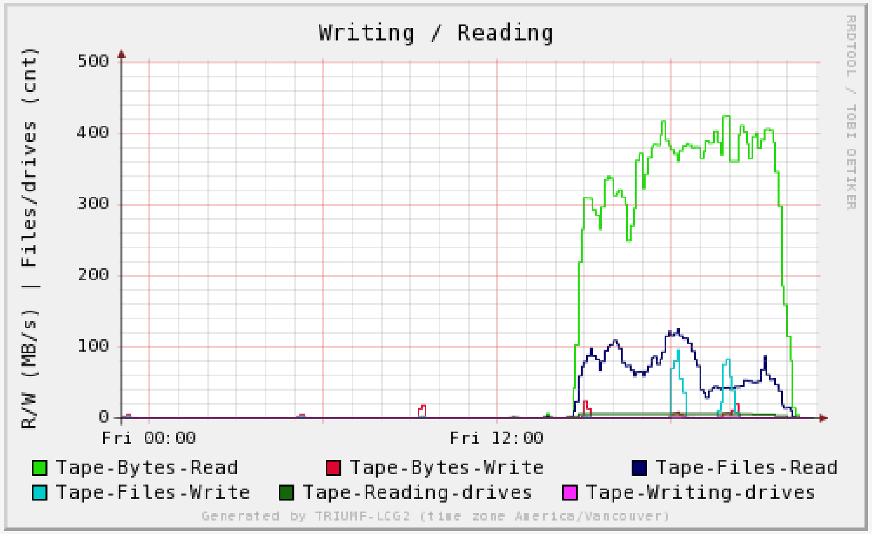
Assume two drives are used for this writing case

- **I/O balancing at reading**
  - **'sleeping' if tuning option 'max_reading_on_one_pool' is reached**
  - **It wastes resources, but makes tape reading stable and efficient**

TRIUMF

Performance and Scalability(I)

- **Bulk pre-stage test**
  - **35 FDR datasets(3172 files)**
  - **9 TB data volume(13 tapes)**
  - **~8 hours to pre-stage (up to 6 drives)**
- **Currently can do: > 1 TB/hour**

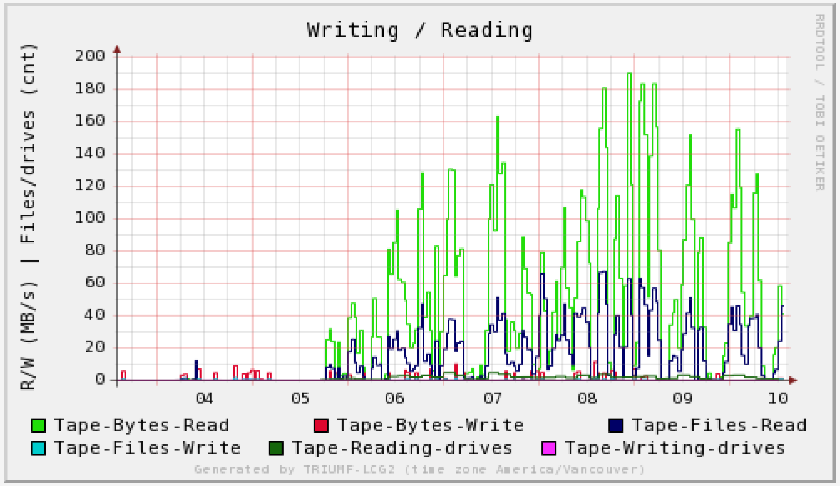


Mass Storage Efficiency (MSS)

Date	R_Rate(MB/s)	W_Rate(MB/s)	Avg_File_R_Size(MB)	Avg_File_W_Size(MB)	R_Per_Mnt(MB)	W_Per_Mnt(MB)	R_Rep_Mnts	W_Rep_Mnts
2009Feb09	65.5	52.14	3001	4160	849740.4	37440	1.00(Total:11)	1.00(Total:0)

Performance and Scalability(II)

- **March-09 reprocessing (data to March 10)**
  - **No file pre-stage in advance (not ideal scenario, but reading still got benefit from dataset level write grouping)**
  - **105 datasets, 13987 files**
  - **23 TB data volume (50 tapes involved)**



Mass Storage Efficiency (MSS)

Date	R_Rate(MB/s)	W_Rate(MB/s)	Avg_File_R_Size(MB)	Avg_File_W_Size(MB)	R_Per_Mnt(MB)	W_Per_Mnt(MB)	R_Rep_Mnts	W_Rep_Mnts
2009Mar09	50.04	52.94	1831	3600	332270.36	43200	1.14(Total:16)	1.00(Total:0)
2009Mar08	40.61	59.82	1380	4373	240637.22	118080	1.50(Total:24)	1.00(Total:0)
2009Mar07	24.82	88.42	1820	3733	170268.62	100800	1.75(Total:28)	1.00(Total:0)
2009Mar06	36.45	79.73	1873	3960	149904.37	95040	1.41(Total:24)	1.00(Total:0)
2009Mar05	39.32	107.93	1808	4560	95840.5	54720	1.00(Total:3)	1.00(Total:0)

Virtual Tape Library

- **Was targeted for Tapeguy functional tests**
- **Features**
  - **It virtualizes tape libraries using disk storage**
  - **Provides tape library and drive operations for integration with Tapeguy (load/unload, tape rewind and tape head seeking etc.)**
  - **Write and read are available to virtual tapes (real read/write, or simulated read/write using symbolic links)**
  - **Very flexible for Tapeguy testing (only 4 definition changes at Tapeguy)**
  - **Overwriting a tape is possible and is tracked**
- **Configurable settings**
  - **Multiple libraries and drives in the same instance**

Conclusion

- **Tapeguy has been in production at the TRIUMF Tier-1 Centre since 2007 (a prototype version was developed in 2005 for Tier-1 service challenges)**
- **Provides greater control and flexibility than proprietary HSMs do**
- **Performance is good, and is expected to be scalable in order to match an increasing throughput demand in the coming years**

QUESTIONS ?

We would like to acknowledge the contributions and support from:

Canada Foundation for Innovation (CFI)

British Columbia Knowledge Development Funds (BCKDF)

National Research Council (NRC)

National Science & Engineering Council (NSERC)

CANARIE

BCNET

HEPNET

TRIUMF

4004 Wesbrook Mall

Vancouver, B.C. Canada V6T 2A3

Tel: +1 604 222-1047   Fax: +1 604 222-1074

e-mail: storage @ lcg.triumf.ca

www.triumf.ca

Backup slides

TRIUMF

HSM communication (with dCache)

- As a plug-in to dCache HSM layer
  - Configured to be called by dCache HSM interface

    <cmd> put | get <pnfsid> <localfilename> -si=<storageInfo> [options]
  - A shell script (hsmcp.sh) does some elementary error checking, then calls Tapeguy client
  - Uses dCache-defined HSM return code range
- 'Talks' with dCache admin interface
  - Granted 'ssh' access to dcache Admin Interface
  - Allows Tapeguy to initiate file flush control
  - Rotates shift pool modes between 'read-only' and 'read-write' mode
  - A standalone cron script cleans up dead read-back requests (is not a part of Tapeguy, but it's one way to cleanup dead request for file)

Tapeguy Implementation

- **Implemented in Perl**
  - TCP socket-based server daemon (can be easily expanded to multiple daemons if needed)
  - Set of perl modules (common, database and device-access)
  - A shell script (hsmcp.sh from HSM) does elementary error checking, then call Tapeguy client
- **Persistent naming of tape devices across all nodes**
  - udev provides persistent naming and assigns ownership of the devices to the unprivileged users
  - stinit is used to configure tape drive initialization (scsi2logical etc.) and to avoid tape rewinding on any reboot
- **MySQL is used as the database backend**
  - Perl DBI layer and DBD::mysql driver are used
  - Client nodes are not allowed to access the database
- **Server and client tape library operation**
  - Uses mtx (mtx-driveinfo) and mt standard linux utilities to operate tape library
  - Uses typical tar to write/read data to tape (block factor 8192 was a good default)
- **Log and e-mail alert**
  - Both file system and database logs (for each tape device operation) are available
  - two different aliases (Tapeguy_info, Tapeguy) via OS e-mail system
- **Granted ssh access to the dCache Admin Interface**

Dark side of HSM (dCache)

- **HSM interface limited user return code is not enough to describe tape system**
  - **30 <= rc < 40 user-defined range deactivates the request**
  - **Reports Problem to PoolManager if receives user-defined range return code**
- **Live processes cost memory**
  - **Reading requests from HSM has to be alive during 'waiting for queued '**
  - **1MB memory cost for each hsmcp.sh**