TReqS : Scheduling for survival

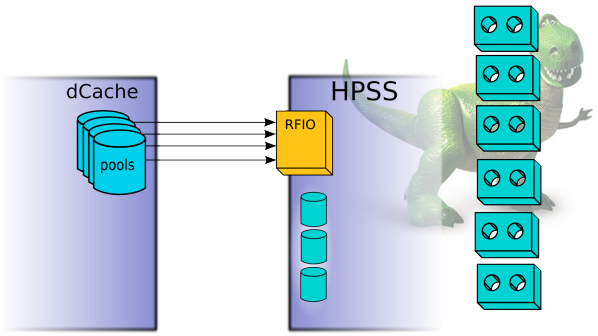
Jonathan Schaeffer

CC-IN2P3 CNRS
-
Storage team

27 October 2009

Notes

Notes



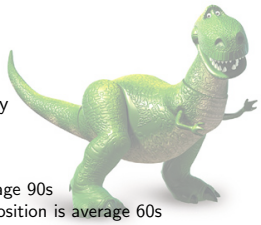
Notes

Experiment's needs

- The LHC experiments use all a tape backend
- They use dCache as a frontend to access it
- They plan to read together at 400MB/s
- Reprocessing campaigns will access several thousands of files

Notes

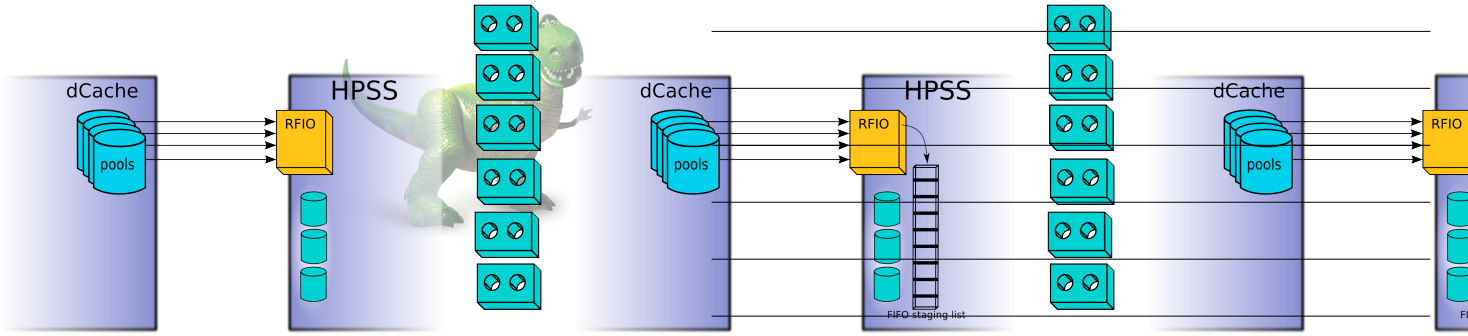
Our setting



- dCache restores a file from HSM blindly
- HPSS handles requests as a FIFO list
- File access latency is important :
  - Moving a tape in the library takes average 90s
  - Moving the reading head to the file's position is average 60s
- Asking for N files on M tapes can take N mounting operations

If the HSM is not smart enough, data access gets really chaotic

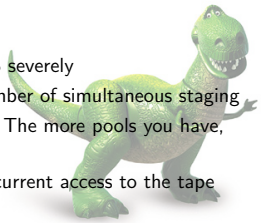
Notes



Notes

As a result...

The dramatics conclusions :



- Moderate dCache usage impacts HPSS severely
- The only control we have is on the number of simultaneous staging
- Staging is managed per dCache pools. The more pools you have, the more staging
- All experiments have uncontrolled concurrent access to the tape drives
- It is foolish to hope reaching the required rates like this

> *"Don't worry Mr B., I have a cunning plan to solve the problem."*
>
> Baldrick in Black Adder III, Episode 5

Notes

Until then...

Manual prestaging

- Before each big exercise, getting a list of files to be accessed
- Sorting the files using HPSS metadata
- Staging the files on HPSS disks
- Only then, the exercise can begin

Manual Prestaging implies a lot of preparatory work and proved to be painful for administrators and experiments

Notes

State of the art

- BNL uses a scheduler for the file access between dCache and HPSS
- It uses HPSS API to get the files metadata and order the file requests by tapes
- Thanks to David Yu, we were able to get their software
- We studied it and started to adapt it to our site

Notes

From BNLBatch to TReqS

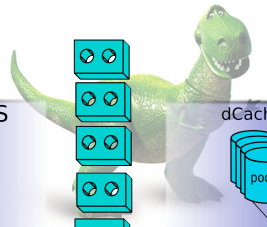
BNLBatch needed some adaptations to fit our local needs.

- Changing the job submission system
  - from file driven to database driven
- Interaction with ACSLS
  - Get the status of a drive from ACSLS database (from ACSLS v7)
- Adding metrics for monitoring and accounting
- Using our local messaging system for monitoring

TReqS hit our production system in july.
TReqS stands for Tape REQuest Scheduler

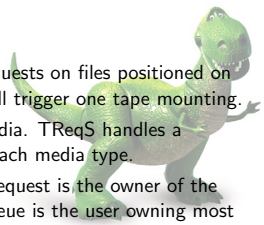
Notes

The big picture Corrected



Notes

Some key concepts



Queue Container for all current requests on files positioned on a same tape. One queue will trigger one tape mounting.

Resource It's a drive for a type of media. TReqS handles a maximum of resources for each media type.

Owner The user submitting a file request is the owner of the request. The owner of a queue is the user owning most of the filerequests in the queue.

Allocation A resource is allocated to the activated queue's owner. There is an allocation table giving a resource allocation for each media type to a user.

Notes

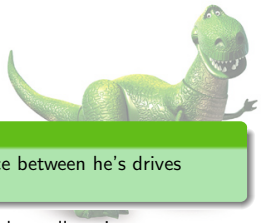
Resources (un)fair Share

- Experiments are concerned about sharing the same resources
- Dedicating drives is the good way to inefficiency and resource waisting

TReqS enables resource sharing and guarantees a minimal resource to a user

Notes

Choosing the best user



Definition

Allocation Score for a user is the difference between he's drives allocation and the used drives.

The best candidate is the one with the highest allocation score

Notes

Choosing the best queue

There can be several way of choosing a best queue :

- The largest in file numbers
- The largest in size
- The oldest
- Some cunning mix of all those parameters

Notes

Operation feedback

TReqS has modest needs:

- A Virtual machine with 512 MB RAM
- Hosting a local MySQL database

This configuration is scalable for thousands of clients

Notes

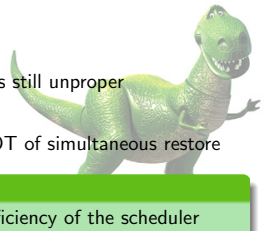
Benefits

From the few month of our experience with TReqS:

- Better resources usage (less mounting, more reading)
- Sharing resources between experiments, ability to guarantee a minimum of drives used
- Quicker file access implies less slow jobs
- HPSS experts less stressed (shiny hairs, shiny smiles, lovely people)

Notes

Still some drawbacks



dCache way of asking for file on a HSM is still unproper

- No cerntralized restore manager
- For best ordering efficiency, allow a LOT of simultaneous restore

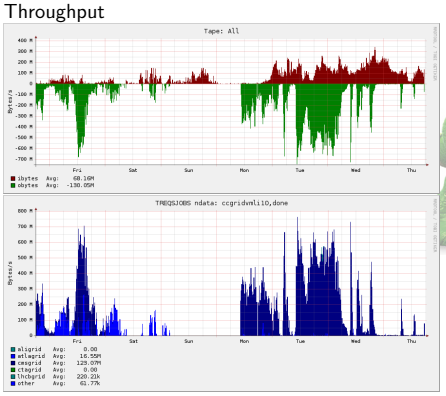
What about good practices ?

Some bad practices are masked by the efficiency of the scheduler

- reading small files
- running job accessing nearline files

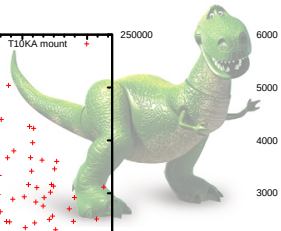
Notes

Throughput

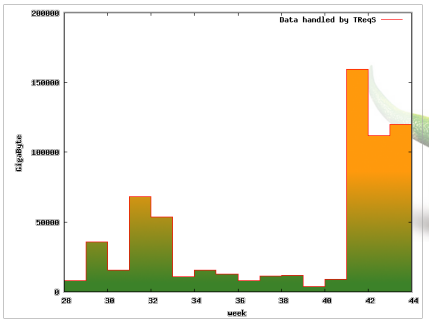


Notes

Tape mounts



Notes



Total data handled : 535 TB

Notes

Global view

1. HPSS migration
2. TReqS start

Notes

Future of the server

- Proof of concept is promising ...
- and stays in production until next release
- which is a completely rewritten system
- emphasizing :
  - scheduling algorithms
  - metrics and accounting
  - operation helpers
  - clean and documented code
  - modern coding practives (OO, Model Driven)
  - modularity (talking with other HSMs)
  - advanced HPSS interaction (querying DB2 metadata database)

Notes

On the client side

- Interfacing with other storage products (like XRootD)
- Interactive TReqS client would fit some users needs
- Having TReqS as only gateway to get files from our HSM

Notes

Conclusion

Scheduling tape requests helped us a lot:

- Achieving descent throughput between dCache and HPSS
- Gaining more serenity on day to day exploitation

"Yes Baldrick, let us not forget that you tried to solve the problem of your mother's low ceiling by cutting off her head."

Edmund Blackadder in Black Adder III, Episode 5

Thank you for your attention.

Notes

Notes

Notes

Notes