Handling Small Files in dCache

Karsten Schwank

15. Mai 2014
Content

1 Problem
   Motivation
   Constraints

2 Solution
   Basic Idea
   System Overview
   Reading and Writing
   Configuration
   Scaling
   Outlook
Problem

With large number of small files on tape drives the read time for one file is dominated by the seek time (tens of seconds)
Motivation

- Transparently optimise tape access for small files
Constraints and Requests

- We have no influence on the tape system
- Use with any recent dCache system (with NFS4.1)
- Keep load on dCache (i.e., Chimera) low
Bundle small files into container files using dCache’s already existing features.
Overview

File Source

Packer

Tape

/data/xx/tape/
/data/xx/
/data/zz/tape/
/data/arcs/a1/
/data/arcs/a2/
/data/cloud/arcs/

Small Files
Overview

Karsten Schwank

Small Files
Our solution makes strong use of dCache’s advantages

- Small files are directly read from dCache, potentially coming from multiple pools
- Container files are written into dCache, potentially being stored on multiple pools
- dCache acts as a cache for the small files and for the containers
- The containers are stored using dCache’s regular tape connection

- The service is integrated into dCache using its HSM mechanism
  → every small file triggers an hsm-script
- We use a special URI to logically connect small files with containers
- The bundling mechanism is file format agnostic, currently uses ZIP
A word of advice

Even though container files are regular dCache files, they should *not* be accessible by the users!
More details

- Problem
- Solution
- Basic Idea
- System Overview
- Reading and Writing
- Configuration
- Scaling
- Outlook

NFS Door

Packer

MongoDB

dCap Door

Pools (small files)

Pools (big files)

Karsten Schwank

Small Files
Writing a small file

What happens if a small file is written?
Client writes file

File Source

NFS Door

Packer

MongoDB

dCap Door

Pools (small files)

Pools (big files)

Karsten Schwank

Small Files
File is stored on pool
HSM script creates entry in DB
Packer sees entry in DB

1. File Source
2. NFS Door
3. MongoDB
4. Packer
5. Pools (small files)
6. dCap Door
7. Pools (big files)

Karsten Schwank
Small Files
Packer adds file to file container

1. File Source
2. NFS Door
3. MongoDB
4. Packer
5. Small Files
6. dCap Door
7. Pools (small files)

**Basic Idea**

- Problem
- Solution
- System Overview
- Reading and Writing
- Configuration
- Scaling
- Outlook

Karsten Schwank

Small Files
File container is written into dCache

Karsten Schwank

Small Files
File container is stored on pool

File Source

NFS Door

Packer

MongoDB

dCap Door

Pools (small files)

Pools (big files)

Small Files
Packer adds container URI to file entry in MongoDB
HSM script returns URI to dCache

Karsten Schwank

Small Files
What happens if a small file is read back?
Client requests small file

1. Client requests small file
2. NFS Door
3. dCap Door
4. Pools (big files)
5. Pools (small files)
6. MongoDB

System overview involves:
- Reading and Writing
- Configuration
- Scaling
- Outlook
Small file pool is triggered to deliver small file

Client

NFS Door

Pools (small files)

Packer

MongoDB

dCap Door

Pools (big files)
Small file pool requests container file

Client

NFS Door

Packer

MongoDB

dCap Door

Pools (small files)

Pools (big files)
Big file pool is triggered to deliver container file
Small file pool extracts small file from container
Small file is delivered to door
Small file is delivered to client
Configuration

What can be configured?
Define one or more *Packaging Classes*. A Packaging Class defines a group of small files that should be handled the same way and end up on the same tape set.

Matching Attributes:
- path and file pattern
- sGroup and storeName pattern
- Minimum file age
- Maximum file age

Storage Attributes:
- Maximum container file size
- Container target directories
Packaging Class Configuration Example

```
[Experiment-Tiffs]
patherExpression=~/data/experiments/xy/  
fileExpression=.*\.tiff?
archiveSize=20G
archivePath=filebundles/exp/

[Experiment-Others]
patherExpression=~/data/experiments/xy/  
fileExpression=.*\.(?!tiff?)
archiveSize=100G
archivePath=filebundles/exp/
```

**Please note:** You have to make sure every small file is matched by exactly one Packaging Class!
Packaging Class Packing Example

hsmInstance = dcache

/data/experiments/xy/
  file1.txt
data.dat
  image.tiff
  ...
/data/experiments/xy/sub1/
  image_1.tiff
  image_2.tiff
  image_3.tiff
  ...
/data/experiments/xy/sub2/
  imageX.tiff
  imageY.tiff
  file1.txt
  ...

hsmInstance = osm

/data/filebundles/exp/
  b1La4a2.darc  20G
  buPknuV.darc  100G
  bc35v1f.darc  20G
  bGA6DyA.darc  20G
Scaling

How does it scale?
Bottlenecks

- The packing script creates one archive at the time
- Stacked up pending small file entries in MongoDB might slow down the packing system
- use multiple instances of the script working on distinct sets of files (Multiple instances of the script can run on the same machine or on different machines)
Current Status

- Deploying the system for evaluation in a pre-production environment at DESY
Future Plans

- Will be made available after successful evaluation
- Implement to expand the whole container if one file from it is requested
Questions?