Small-File Aggregation for dCache Tape Interface

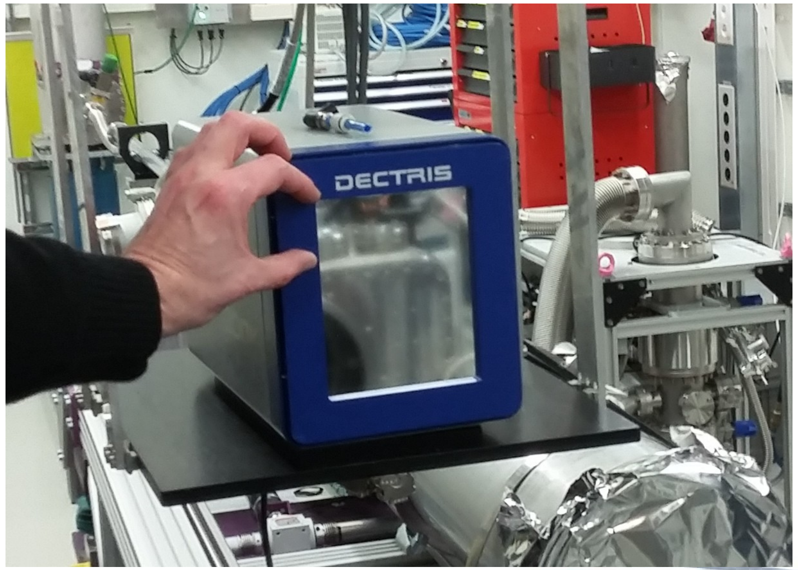
Packing small files for better tape performance

Svenja Meyer for dCache team
Hamburg, 17.03.2021

Agenda

About PETRA III

Ring-based x-ray radiation source at DESY since 2010

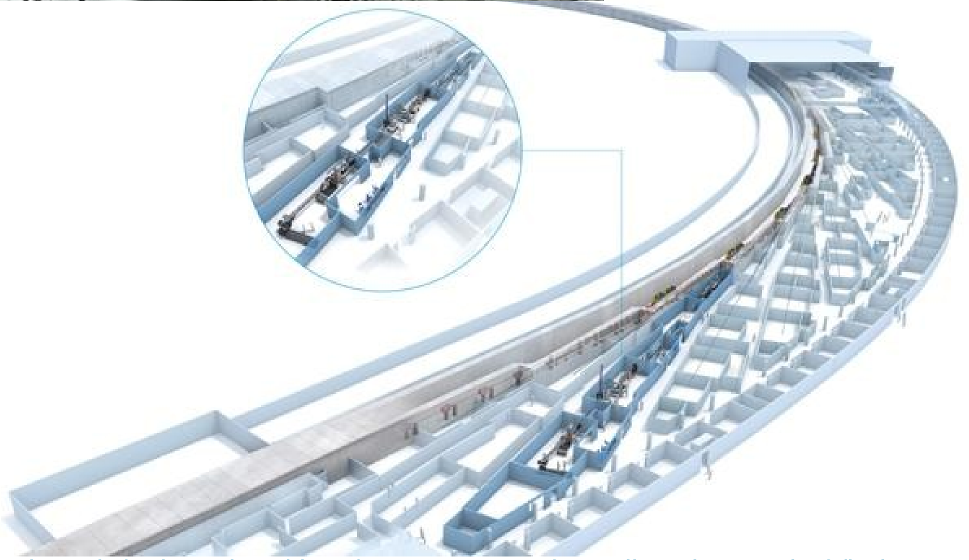


Some facts

- 2.3 km long
- First particle physics, later pre-accelerator for HERA, now x-ray radiation source
- 21 beamlines with 45 measuring stations

More than beamlines...

- We offer archiving and computing power, too
- Guaranteed data preservation for 20 years
- Sometimes data has to be stored twice on different tapes
- Not possible to repeat measurements easily



Picture taken from https://www.desy.de/ueber_desy/desy/grossgeraete_fuer_die_wissenschaft/index_ger.html

The problem with data taken from PETRA III

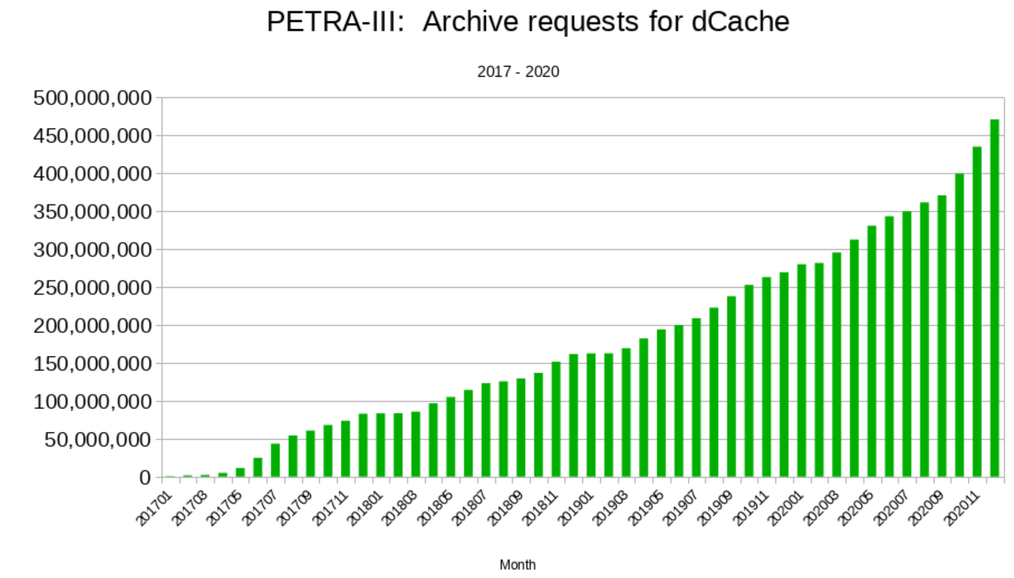
Large number of small files to archive

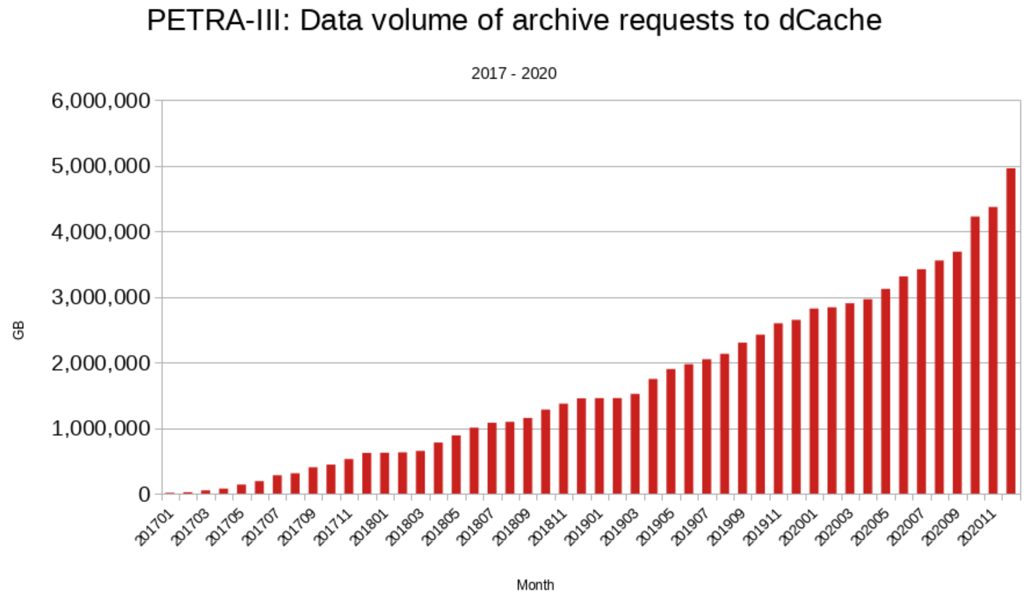
More and more small files

- PETRA III at DESY is producing many small files
- 5 PB, 450,000,000 files
  - That is an average of ~ 11 MB per file
- Sometime 2 million files per directory

Mount, fast-forward, write or read, unmount, repeat

- Average of 50 s before data can be written to tape
- Writing data rate is ideally 300 MBps
- Packing small files to bigger files to improve performance
- Files of one beamtime are usually read together





SmallFiles-Plugin

Current solution since 2014 to pack small files into bigger ones

How it works - flushing

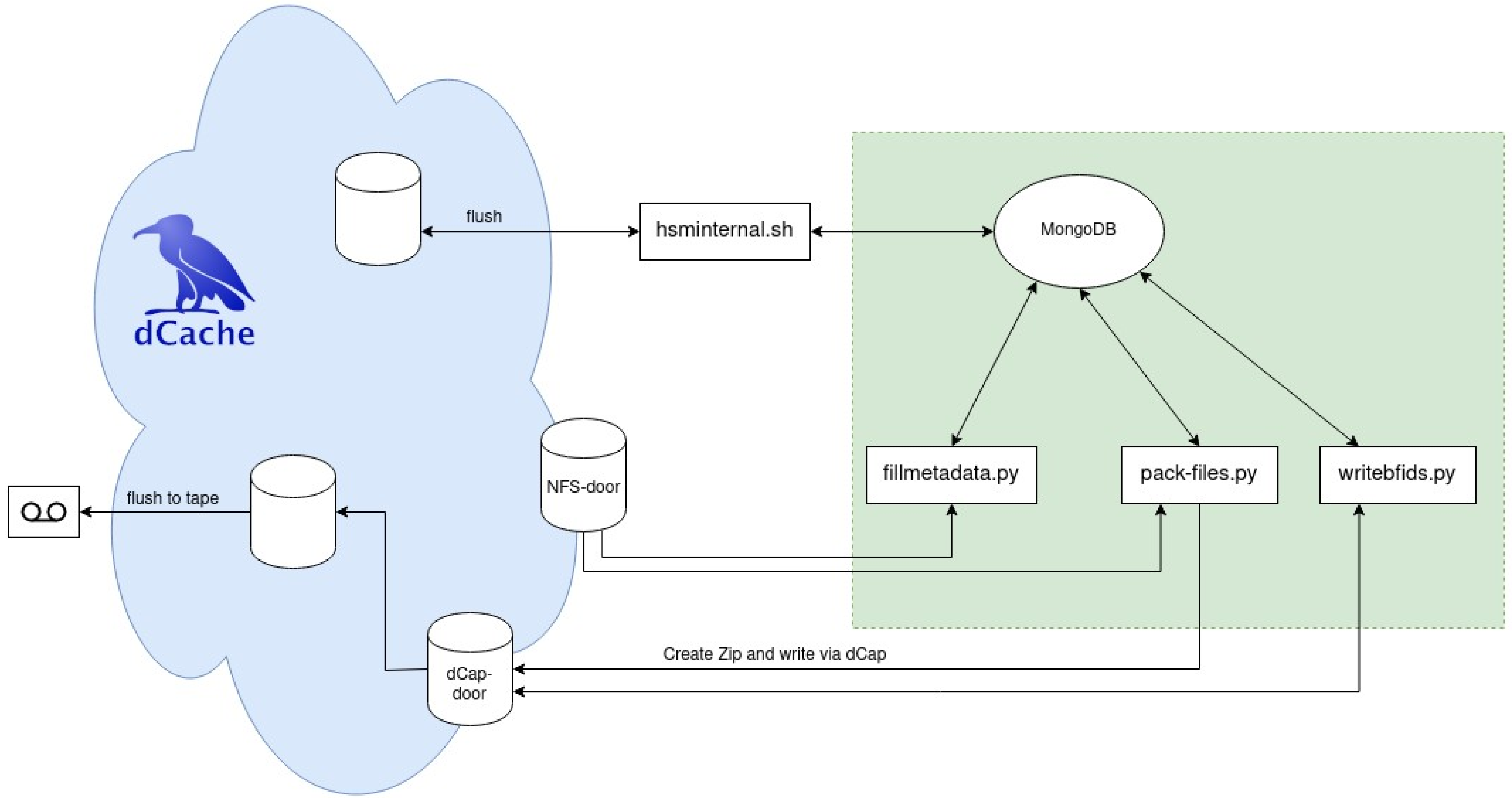
- dCache triggers a script that creates a minimal record in a MongoDB
- This record gets extended by another script
- With the large record the files get packed to zip-files
- The last step is to check if the files got packed correctly. The zip-file is then renamed and the database records of the files gets updated
- The first script recognizes this and sets the files in dCache to “cached”

How it works - staging

- dCache triggers a script that opens the zip-file via dCap
- This way, it returns the requested file back to dCache

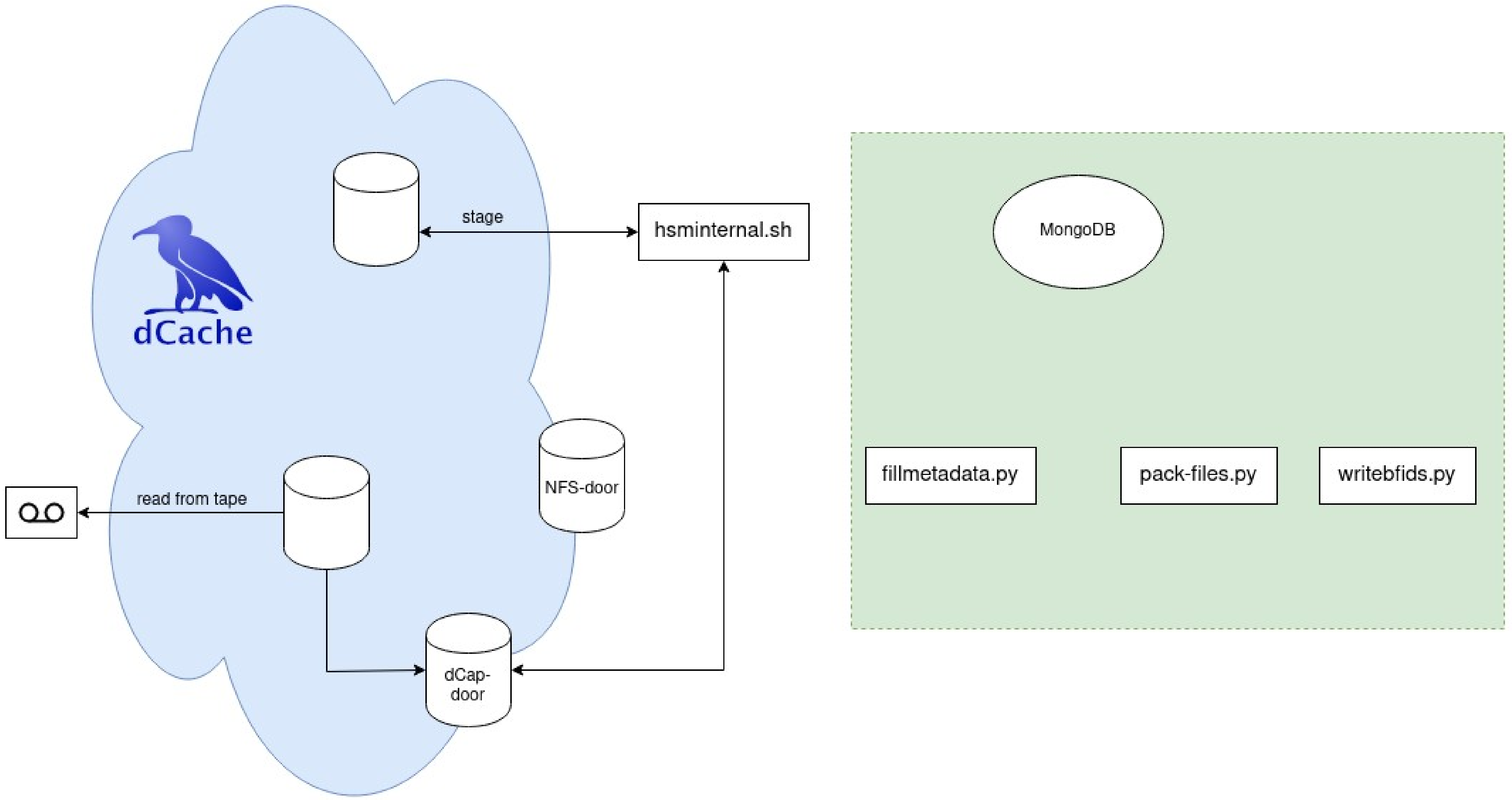
SmallFiles-Plugin

Current solution since 2014 to pack small files into bigger ones



SmallFiles-Plugin

Current solution since 2014 to pack small files into bigger ones

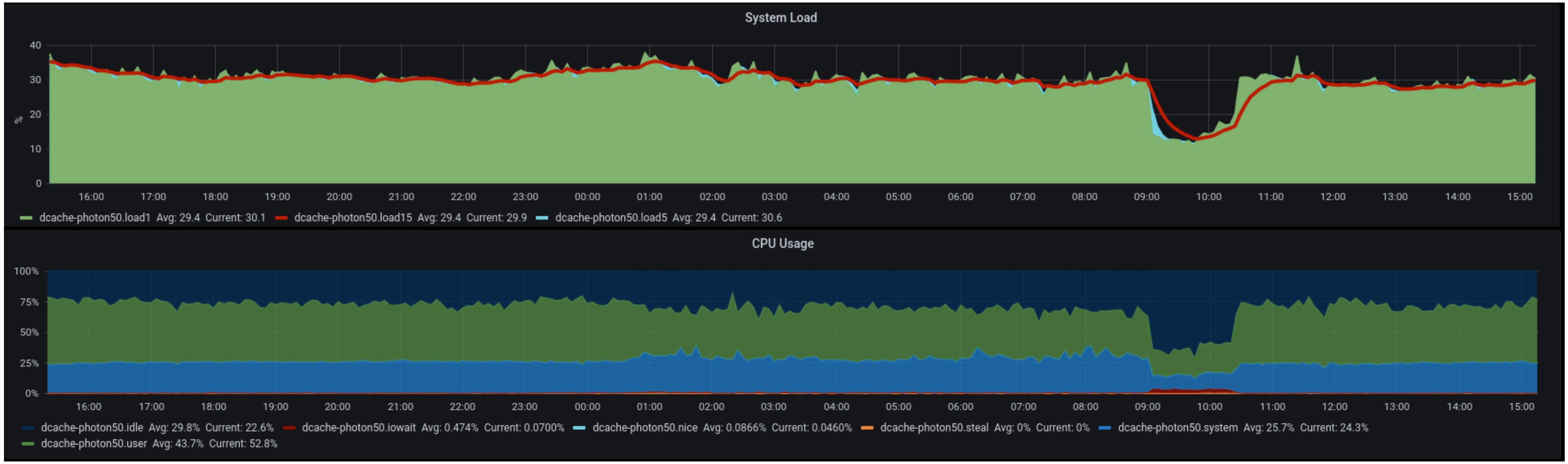


Problems with the SmallFiles-Plugin

It works, but too slowly...

High load on machine

- For every flush request a new connection is opened
- Every file retries flushing until it gets response that it is on tape
- MongoDB works best with one connection for all requests

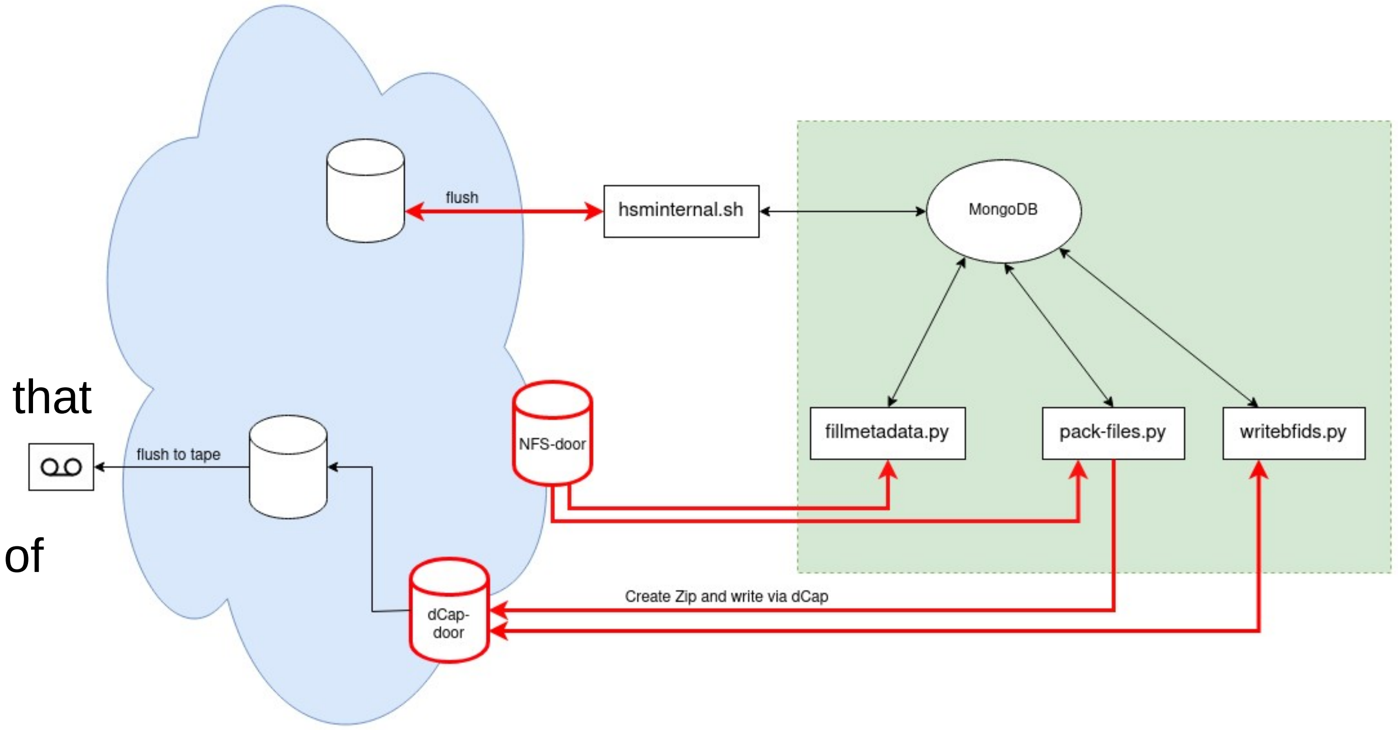


Problems with the SmallFiles-Plugin

It works, but too slowly...

“The dose makes the poison”

- Each request for a file to the NFS-door puts additional load to dCache
- Accessing a file via NFS means requesting metadata and getting pool information of files that we already know
- Problems appeared with increasing numbers of flushes, more data just makes it worse
- It is a little bit like a denial-of-service attack

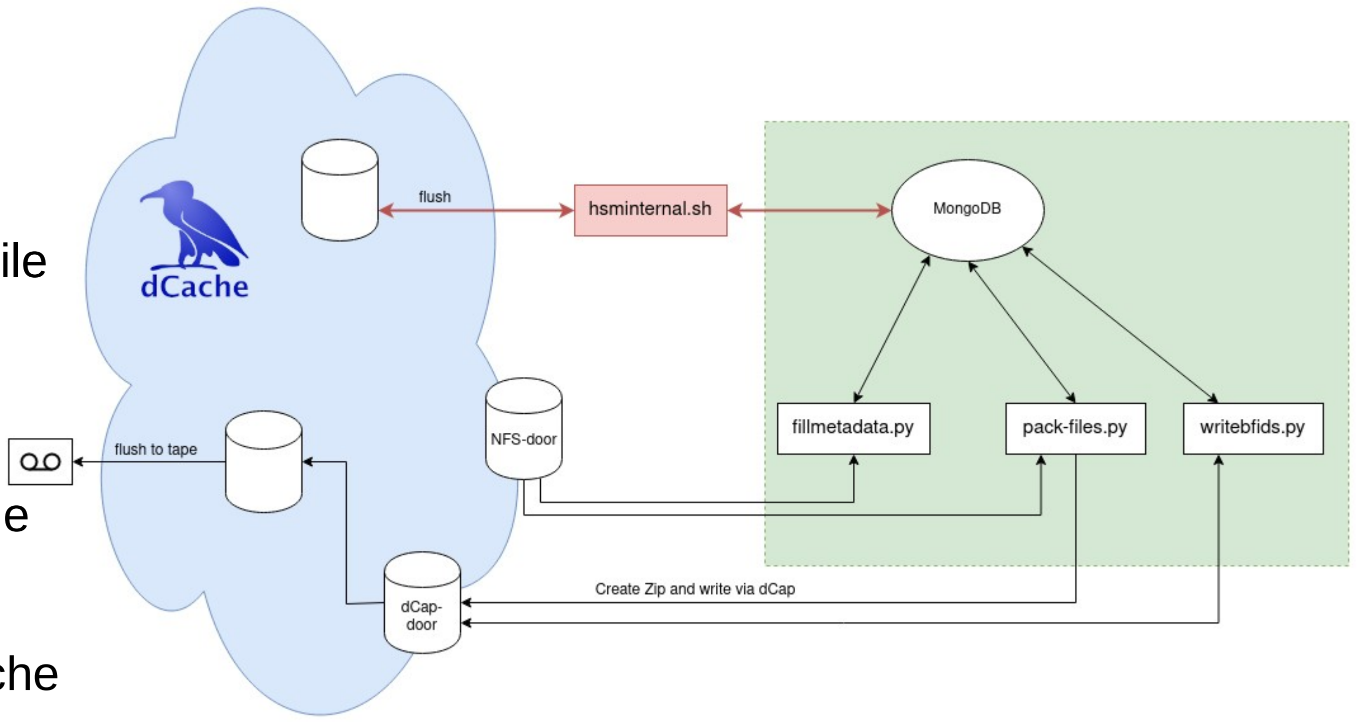


The future of packing small files - Sapphire

Getting faster avoiding NFS and dCap

Step 1

- Replacing hsminternal.sh with a dCache-native Java-Driver
- Makes it possible to replace one connection per file with one connection for all files
- Possibly decreasing load on pools
- With these changes, staging is not possible for the moment
- You can still use the old version for staging, dCache is flexible enough
- For us, stage is not that important for the moment: It doesn't use many resources, it is rarely used

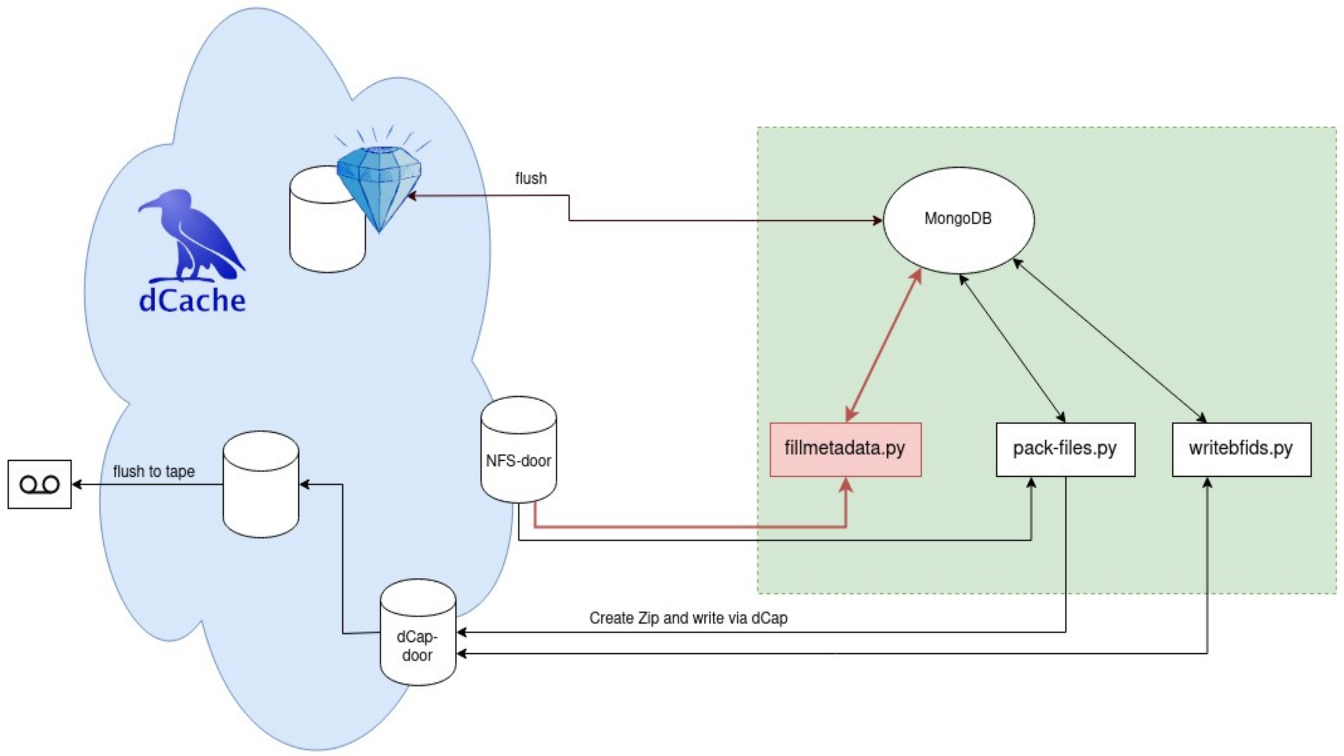


The future of packing small files - Sapphire

Getting faster avoiding NFS and dCap

Step 2

- Replacing fillmetadata.py with the Java-Driver
- The Java-Driver already gets all relevant information from dCache directly
- Could potentially lead to even less load
- One connection to NFS less for each file

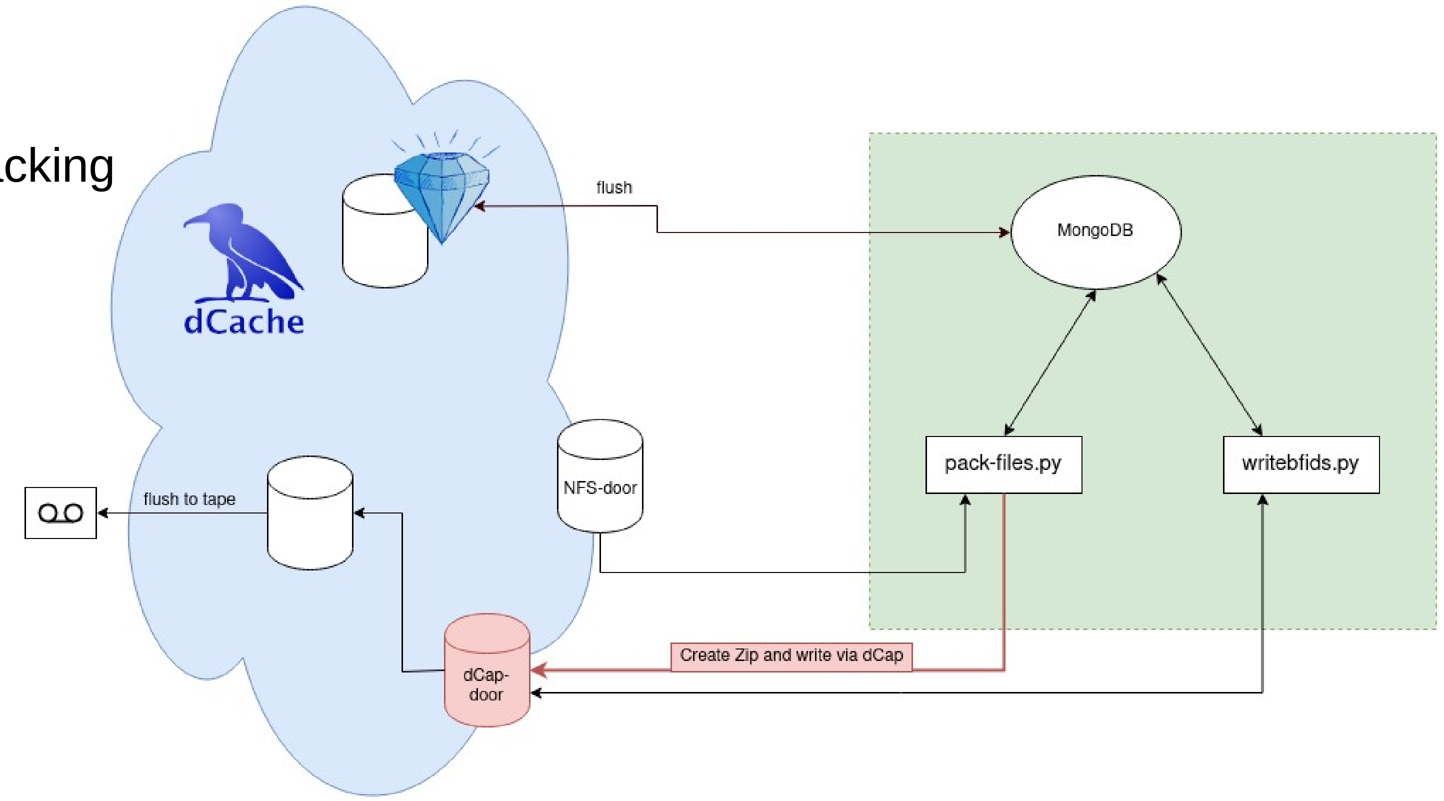


The future of packing small files - Sapphire

Getting faster avoiding NFS and dCap

Step 3

- Packing locally on the packing machine
- No need to use dCap anymore
- At this point staging is working again with downloading and unpacking locally on the packing machine

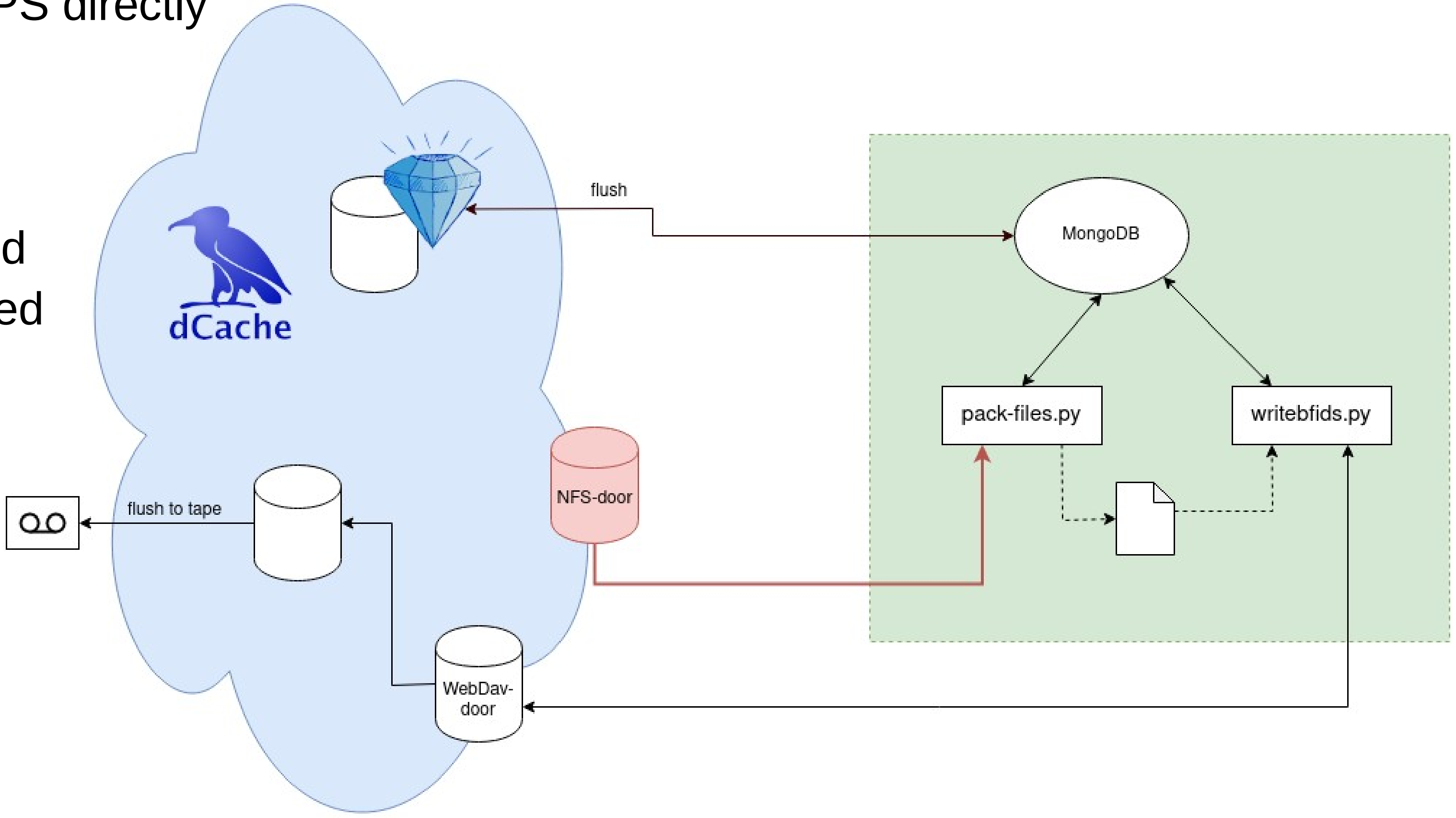


The future of packing small files - Sapphire

Getting faster avoiding NFS and dCap

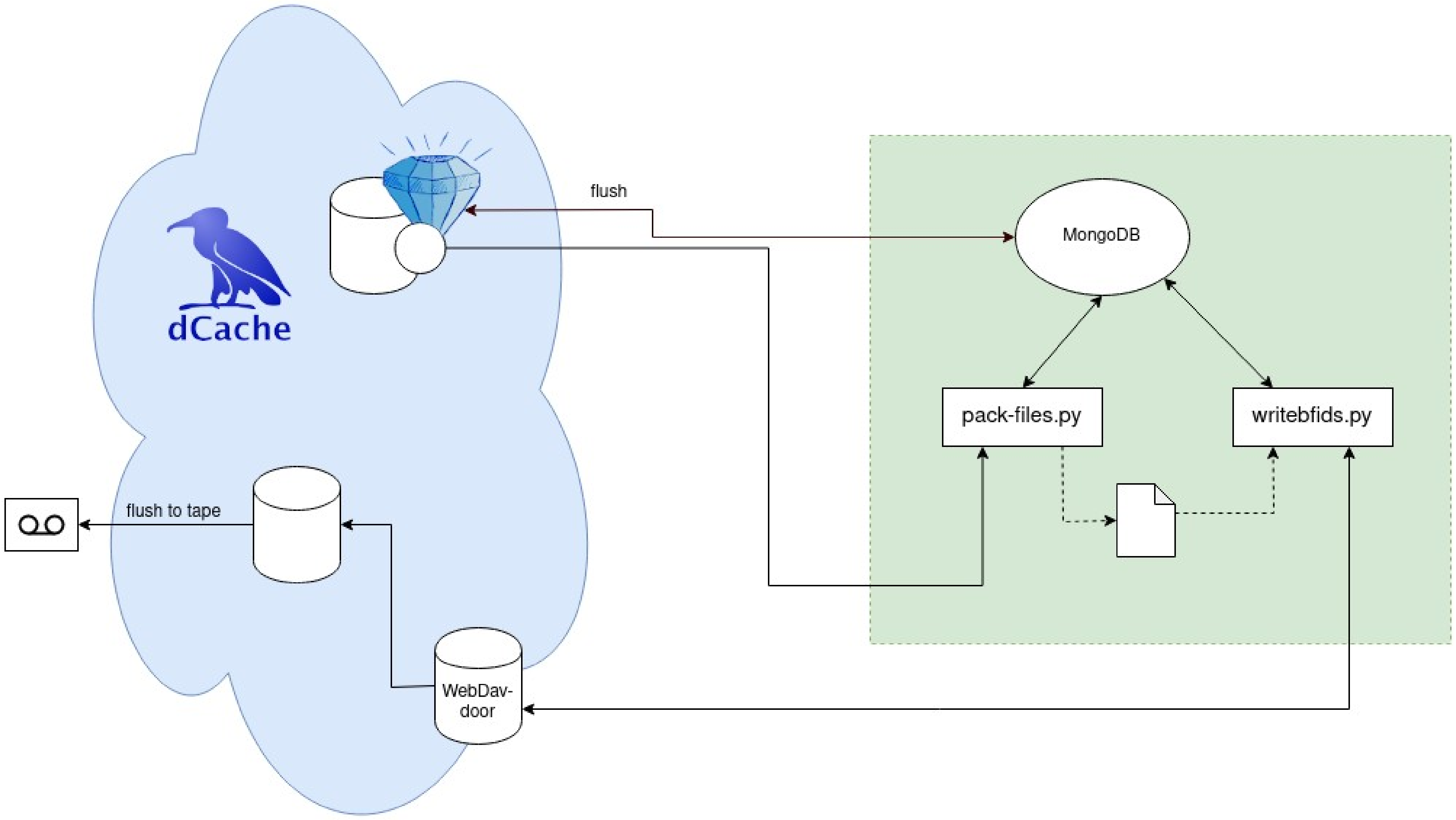
Step 4

- Download the files to be flushed via HTTPS directly out of Sapphire
- No NFS connections anymore
- Less load on dCache as the metadata and pool information doesn't have to be queried



The future of packing small files - Sapphire

Getting faster avoiding NFS and dCap

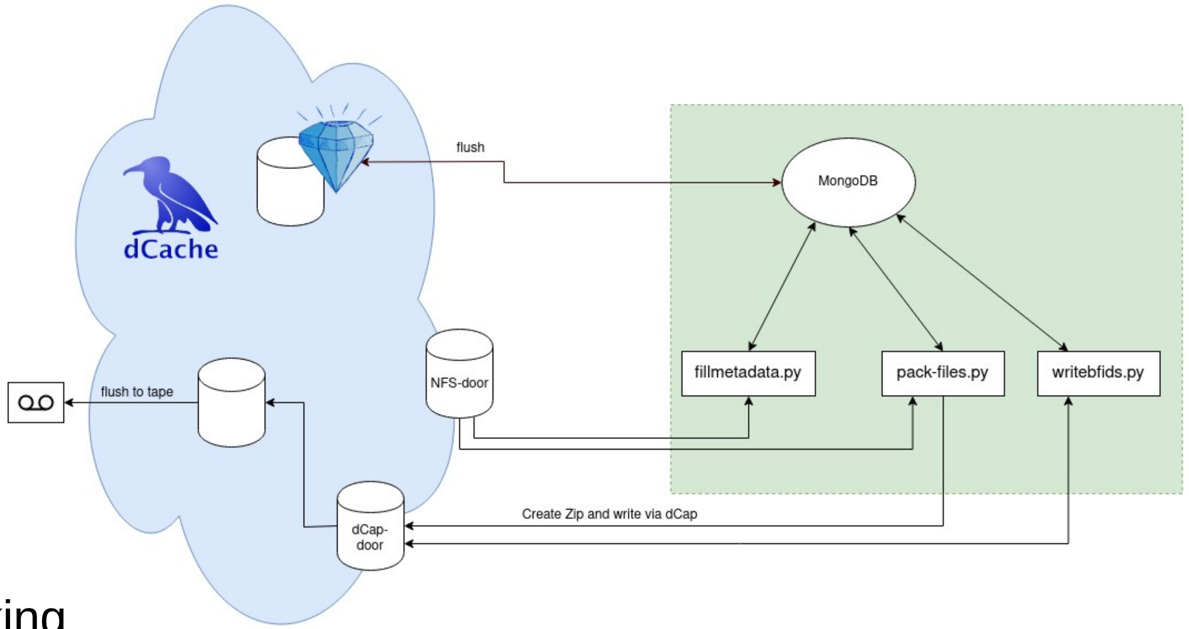


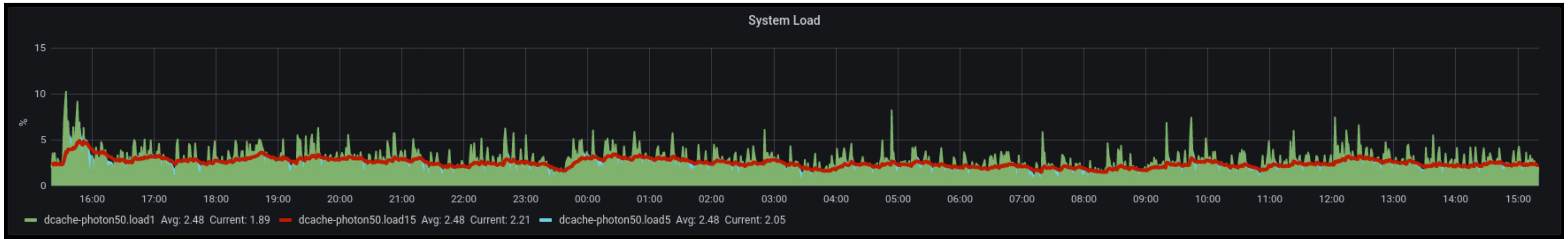
Present situation

First improvements

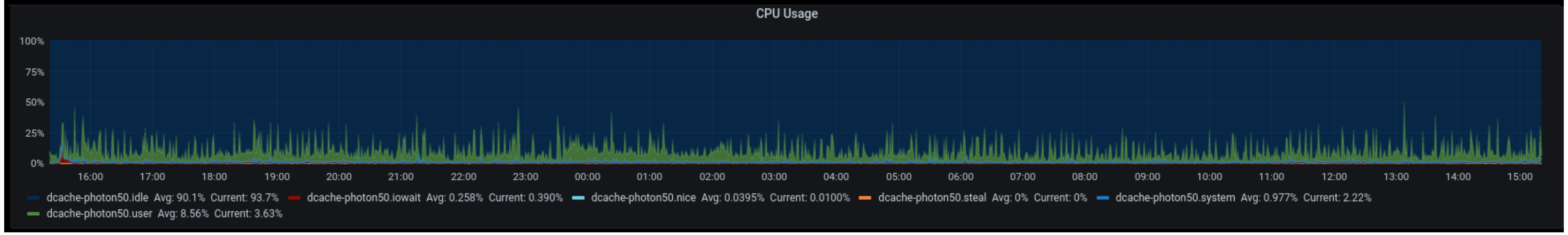
Step 1: reached

- Replace hsminternal.sh
- Installation on three productive pools
- Load decreased by a factor of 10
- Still need some improvement to speed up packing





🟩 dcache-photon50.load1 Avg: 2.48 Current: 1.89 🟥 dcache-photon50.load15 Avg: 2.48 Current: 2.21 🟦 dcache-photon50.load5 Avg: 2.48 Current: 2.05



🟦 dcache-photon50.idle Avg: 90.1% Current: 93.7% 🟥 dcache-photon50.iowait Avg: 0.258% Current: 0.390% 🟦 dcache-photon50.nice Avg: 0.0395% Current: 0.0100% 🟧 dcache-photon50.steal Avg: 0% Current: 0% 🟦 dcache-photon50.system Avg: 0.977% Current: 2.22%
🟩 dcache-photon50.user Avg: 8.56% Current: 3.63%

Future steps

Our roadmap

Step 2: Waiting for tests

- Drop fillmetadata.py
- (First) version finished development, now waiting for the deployment in step 1 to finish
- Prerequisite: pools need an update of dCache

Step 3: In development

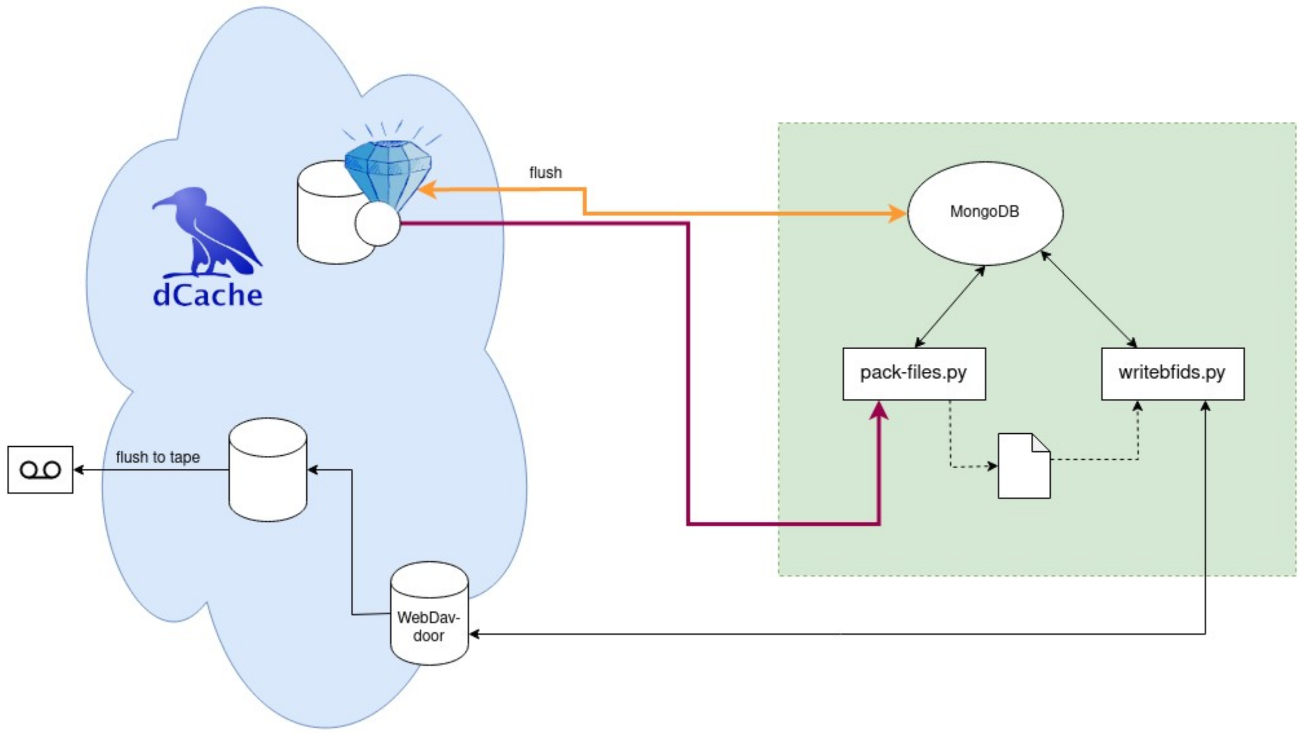
- Pack files locally
- Currently in development phase
- First files already got packed locally during local testing
- However, still a lot of things to be done

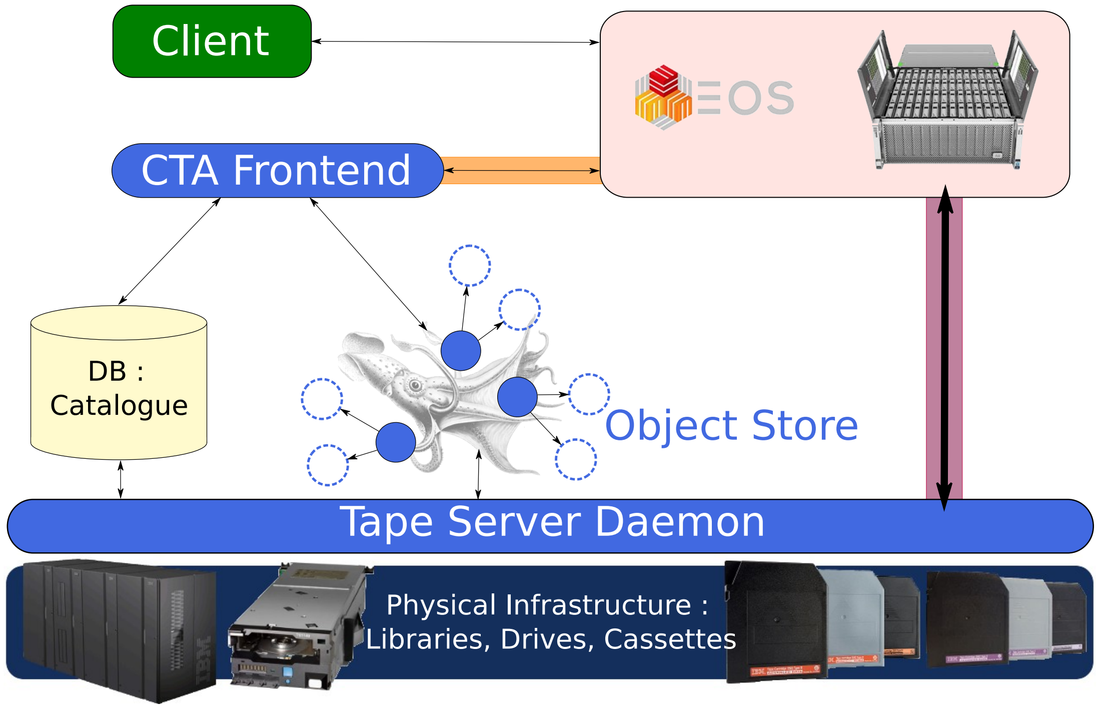
Step 4: Plan for May

- Request data directly through Java-Driver
- Development not started
- Plan is to finish this step in May to start first bigger tests

Sapphire for CTA

Base for plugin to connect dCache to CTA





Picture taken from
https://gitlab.cern.ch/cta/CTA/-/blob/master/doc/Presentations/20200722_GWDG/images/CTA_Arch3A.svg

Thank you

Next dCache (virtual) workshop: June 1- June 2 2021

Contact

DESY. Deutsches Elektronen-Synchrotron

www.desy.de

Svenja Meyer
IT-Scientific Computing
svenja.meyer[at]desy.de