

# Lessons learned (or to learn) & Architectural Evolution



**Andreas-Joachim Peters**  
CERN - IT  
Storage Group



# A self-critical review



- an attempt to look at
  - where do we have problems today?
    - how do we correct them?
  - which parts of the system did not mature enough?
    - how do we correct this?
  - what can we improve in the future process?

# EOS Architecture Aquamarine



 Open Source Storage

 Open Source Storage



Beryl Aquamarine  
**V 0.3.X**

all instances today!

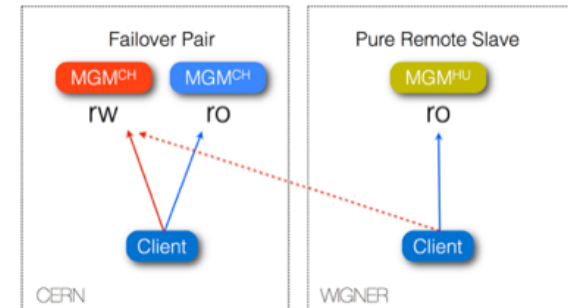
read/  
write

MGM  
Master

MGM  
Slave

read  
only

META DATA



FST

FST

FST

FST



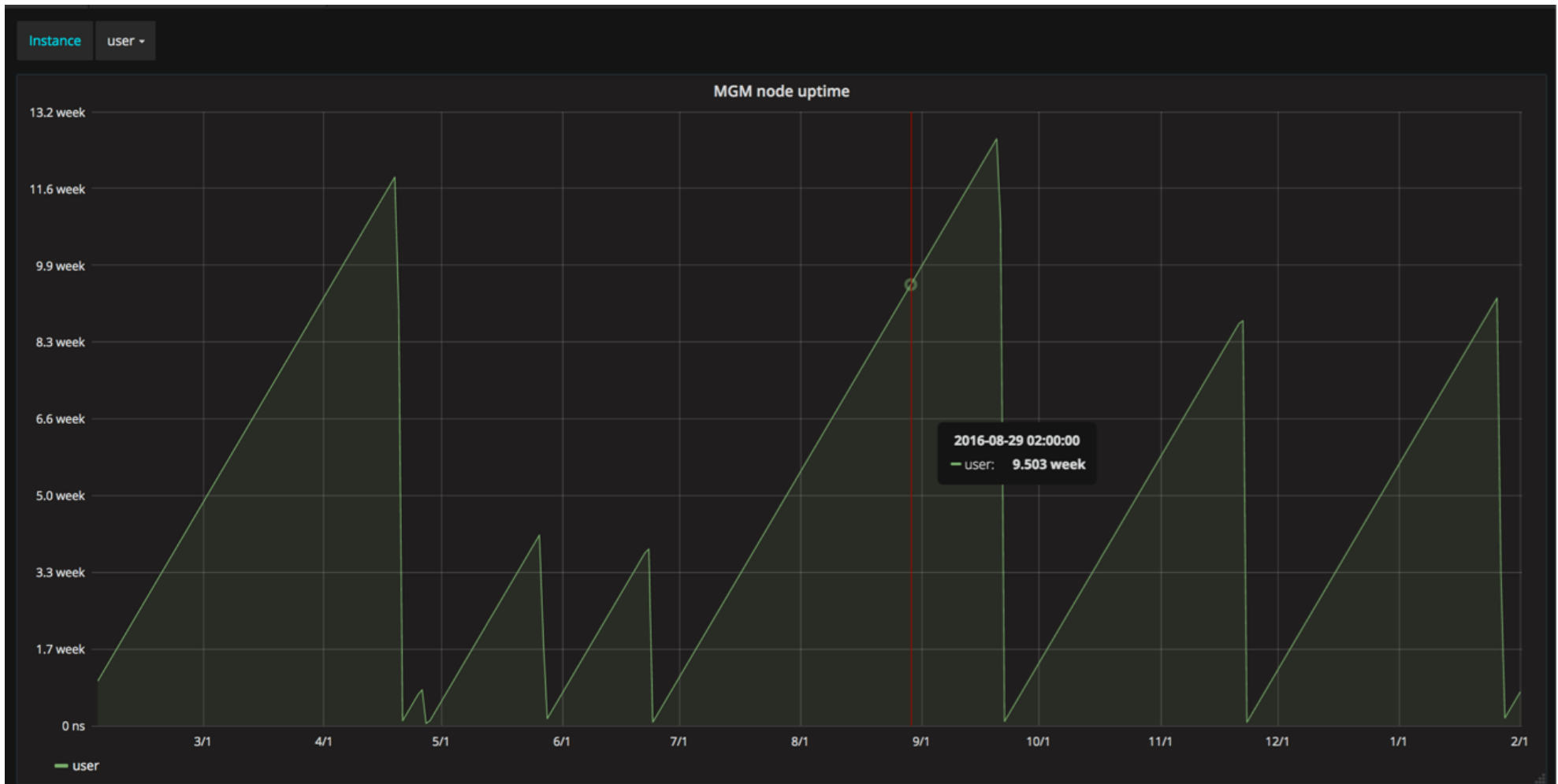


CERN Open Source Storage

CERN open source storage

# In-memory namespace

## Namespace Uptime in EOSUSER/Cernbox



8 starts in 1 year



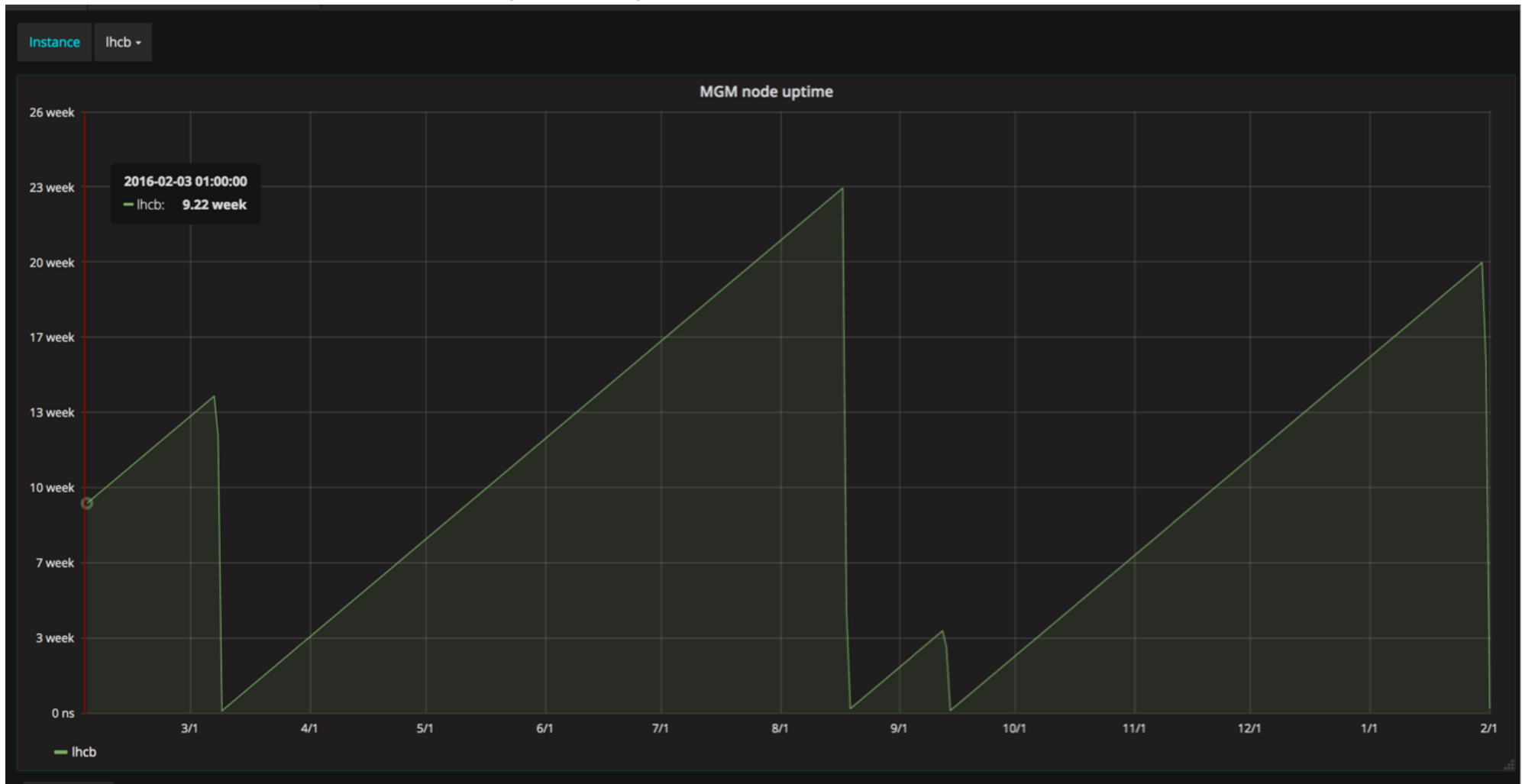


CERN Open Source Storage

Open Source Storage

# In-memory namespace

## Namespace Uptime in EOSLHCB



4 starts in 1 year

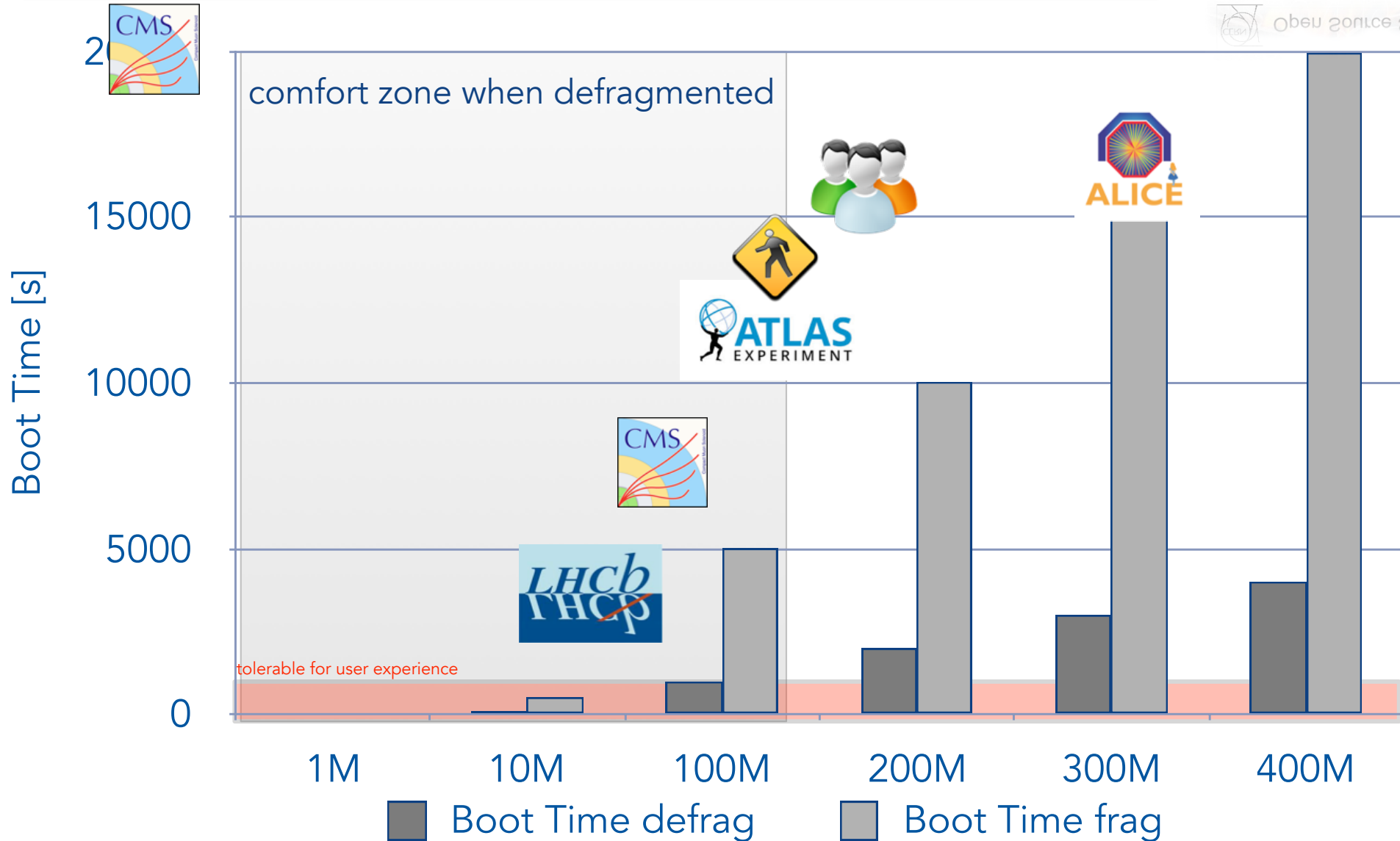




CERN Open Source Storage

CERN Open Source Storage

# In-memory namespace

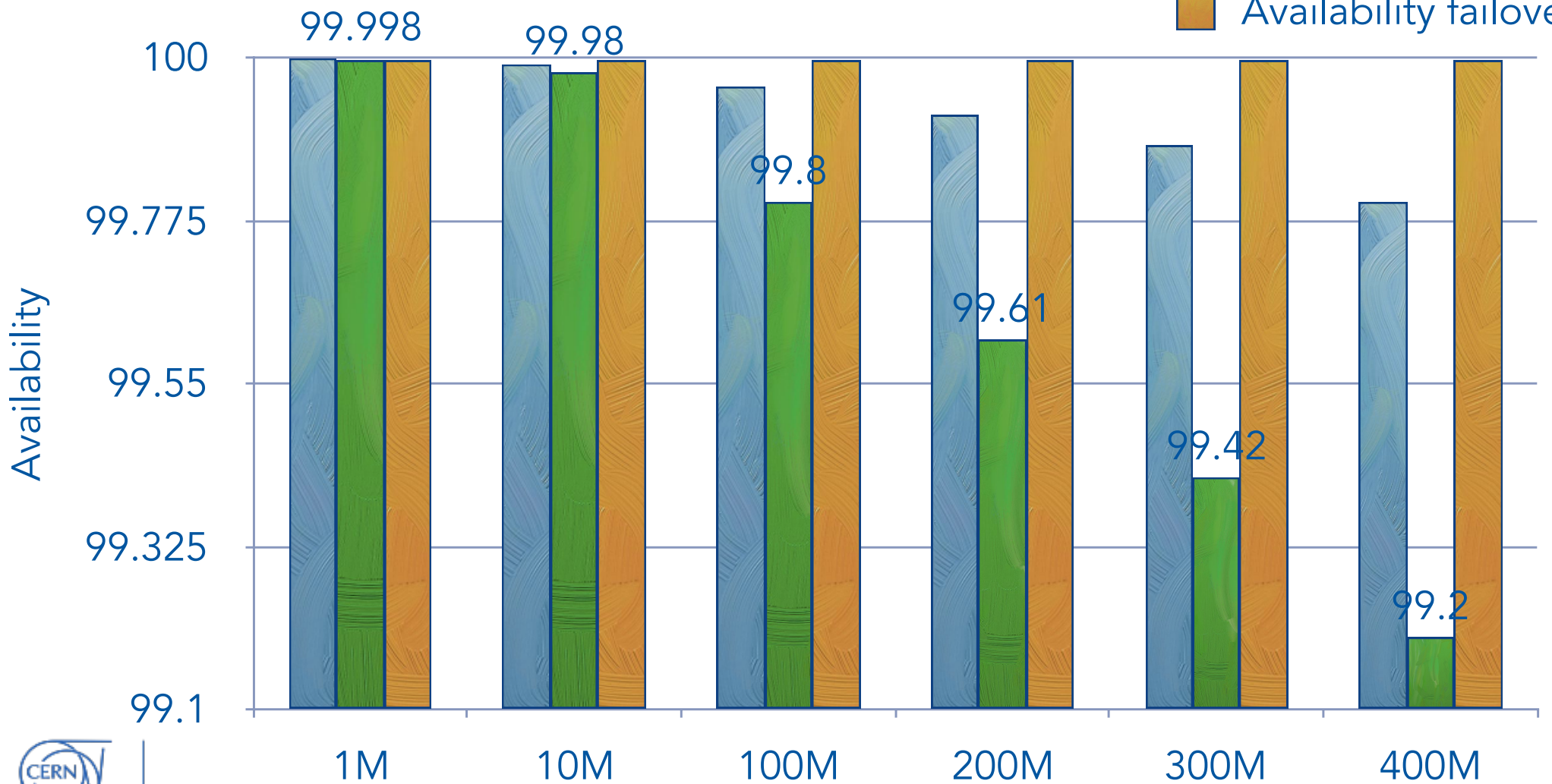




- Availability defrag
- Availability frag
- Availability failover

# In-memory namespace

Availability [ assuming uptime of 1 month ]





# In-memory namespace

- we crossed the ‘comfortable’ operation point with most instances already
- only acceptable way of operation is failover for instances  $> 100M$  entries
  - still until recently failover procedure was not mature to failover large namespaces - why?
    - requires systematic QA procedure to exercise transparent failover in a continuous testing/integration system
    - failover model was not complete
      - DNS alias change model is not ideal
        - load-balanced fronted redirector
        - virtual service IP failover
        - ....




# In-memory namespace



- a practical problem now at CERN
  - we push more and more people into the service before having solved this known issues
- a positive effect was that bugs in production get/got a lot of attention and fixes
- for the future version: need to put more effort in planning and exercising all possible failure scenarios

# Replica Consistency



 Open Source Storage

 error  $\sim 10^{-5}$

```
[root@eosuser-srv-m1 ~]# eos fsck report
timestamp=1485943737 tag="d_cx_diff" n=190 shadow_fsid=
timestamp=1485943737 tag="d_mem_sz_diff" n=2198 shadow_fsid=
timestamp=1485943737 tag="m_cx_diff" n=97 shadow_fsid=
timestamp=1485943737 tag="m_mem_sz_diff" n=102 shadow_fsid=
timestamp=1485943737 tag="orphans_n" n=86202 shadow_fsid=
timestamp=1485943737 tag="rep_diff_n" n=3747 shadow_fsid=
timestamp=1485943737 tag="rep_██████████" n=30 shadow_fsid=
timestamp=1485943737 tag="unreg_n" n=241 shadow_fsid=
```


error  $\sim 6\%$

```
[root@eosalice-srv-m3 ~]# eos fsck report
timestamp=1485942572 tag="adjust_replica" n=2090366 shadow_fsid=
timestamp=1485942572 tag="d_cx_diff" n=3492 shadow_fsid=
timestamp=1485942572 tag="d_mem_sz_diff" n=210017 shadow_fsid=
timestamp=1485942572 tag="file_██████████" n=118151 shadow_fsid=
timestamp=1485942572 tag="m_mem_sz_diff" n=957810 shadow_fsid=
timestamp=1485942572 tag="orphans_n" n=17429227 shadow_fsid=
timestamp=1485942572 tag="rep_diff_n" n=288461 shadow_fsid=
timestamp=1485942572 tag="rep_██████████" n=2098957 shadow_fsid=
timestamp=1485942572 tag="unreg_n" n=33344 shadow_fsid=
timestamp=1485942572 tag="zero_replica" n=144 shadow_fsid=
```



# Replica Consistency



 Open Source Storage

 error  $\sim 10^{-5}$

```
eos f[root@eoscms-srv-b1 ~]# eos fsck report
timestamp=1485943637 tag="adjust_replica" n=449869 shadow_fsid=21190
timestamp=1485943637 tag="d_cx_diff" n=23 shadow_fsid=21190
timestamp=1485943637 tag="d_mem_sz_diff" n=153 shadow_fsid=21190
timestamp=1485943637 tag="file_██████████" n=45 shadow_fsid=21190
timestamp=1485943637 tag="m_mem_sz_diff" n=2242 shadow_fsid=21190
timestamp=1485943637 tag="orphans_n" n=3651 shadow_fsid=21190
timestamp=1485943637 tag="rep_diff_n" n=22524 shadow_fsid=21190
timestamp=1485943637 tag="rep_██████████" n=450826 shadow_fsid=21190
timestamp=1485943637 tag="unreg_n" n=2875 shadow_fsid=21190
```

error  $\sim 1\%$

```
[root@eosatlas-srv-b1 ~]# eos fsck report
timestamp=1485942448 tag="adjust_replica" n=94568 shadow_fsid=25358
timestamp=1485942448 tag="d_cx_diff" n=299 shadow_fsid=25358
timestamp=1485942448 tag="d_mem_sz_diff" n=315 shadow_fsid=25358
timestamp=1485942448 tag="file_offline" n=279 shadow_fsid=25358
timestamp=1485942448 tag="m_cx_diff" n=3 shadow_fsid=25358
timestamp=1485942448 tag="m_mem_sz_diff" n=3064 shadow_fsid=25358
timestamp=1485942448 tag="orphans_n" n=6918 shadow_fsid=25358
timestamp=1485942448 tag="rep_diff_n" n=85118 shadow_fsid=25358
timestamp=1485942448 tag="rep_offline" n=94840 shadow_fsid=25358
timestamp=1485942448 tag="unreg_n" n=1806 shadow_fsid=25358
```



# Replica Consistency



- where do errors come from?
  - bugs in previous FST versions
    - use of FUSE plays a big role here
  - machine restarts
  - lost client messages
- no fully automatic procedure correcting this inconsistencies
  - leaving files with inconsistent layouts might lead to data loss with the next unreadable filesystem
- need to systematically cover all failure scenarios
- implement an automatic repair procedure - manually is not good
- also here: need continuous QA testing platform





# Service Scalability



- we have seen service overload due to limitations in GSI handshakes/s
  - last week EOSCMS 'slow' due to 35.000 batch clients doing GSI handshakes
- we have a solution to scale the authentication part, but we never even tested that in the pre-production service
- for the future: all service components have to be as stateless as possible to enable scale-out



Open Source Storage



Open Source Storage

# Replica, RAIN & Clustered Storage

- file replica storage is easier to understand and to recover
  - not optimal to get the maximum performance out of available resources when only few files are used
  - imposes large fluctuations in latency due to unbalanced load for low number of streams
- still: FST model is very practical for life-cycle management, geo-geographic distribution etc.
  - head towards clustered storage behind stateless FSTs for larger installation
    - similar to stateless Kinetic prototype
      - multi-path scheduling => Geoffray



# EOS Architectural Evolution



Beryl Aquamarine  
**V 0.3.X**



Citrine  
**V 4.X**

read/  
write

MGM  
Master

MGM  
Slave

read  
only

META DATA  
stateless

MGM

MGM

MGM

Persistency

FST

FST

FST

FST

DATA

FST

FST

FST

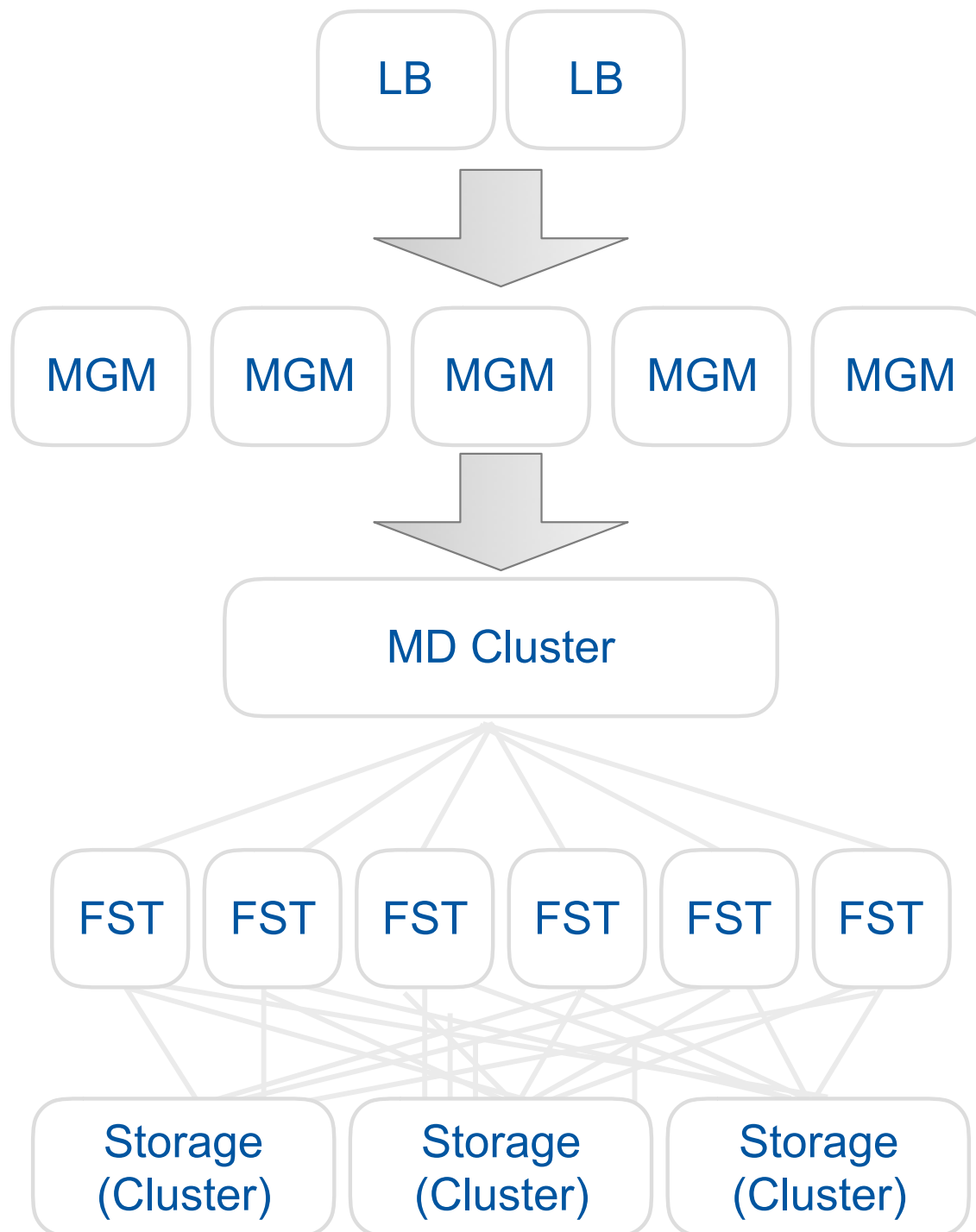
FST





 Open Source Storage

 Open Source Storage





2011

remote  
data  
store



Open Source Storage

Open Source Storage

# Interface Evolution

2017

remote  
data  
store

+

file  
transactional  
storage



remote  
data  
store

+

file  
transactional  
storage

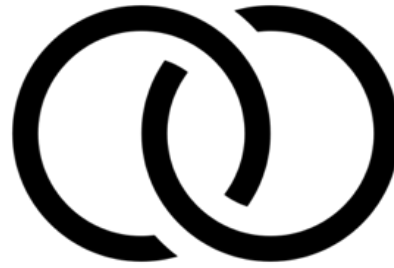
+

distributed  
filesystem  
behaviour



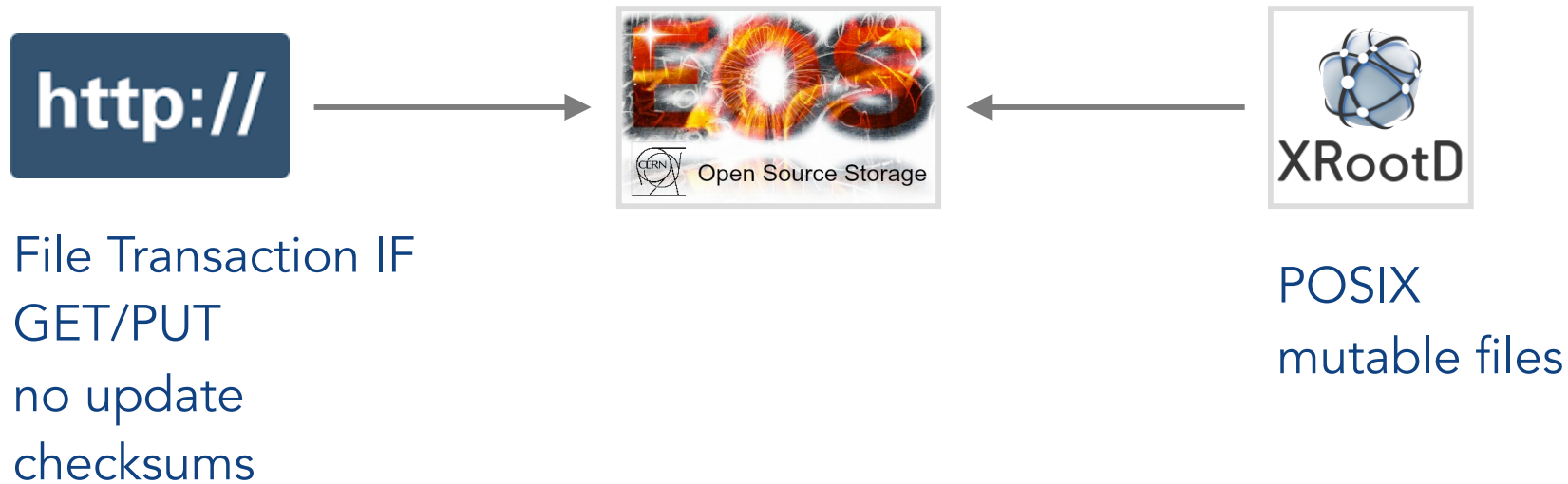
Evolution

**EOS has started 6 years ago as a remotely accessible data storage system with *posix-similar interface*. The interfaces has been extended to provide **file transaction functionality**. The most recent architectural change is to provide mounted filesystem semantics.**

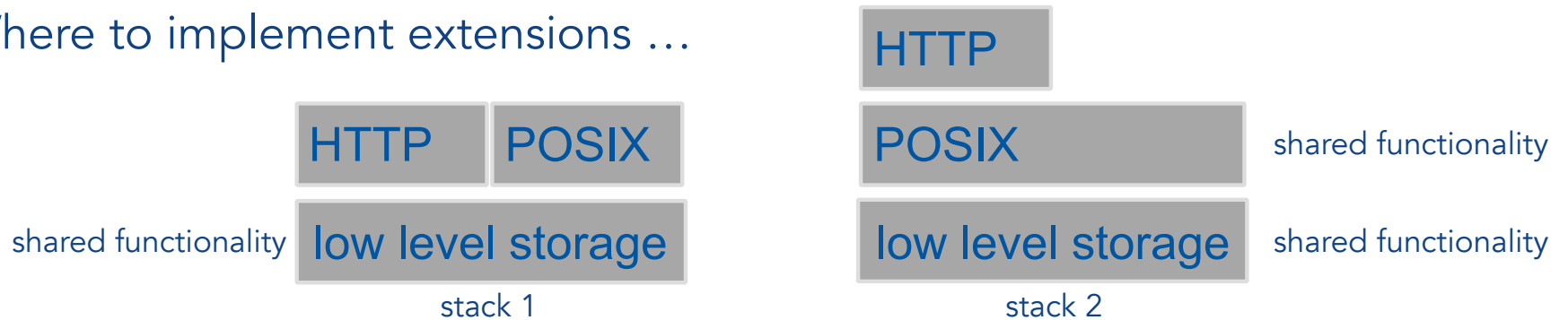


The challenge:

**How to marry two different worlds in the same storage system and make them visible to each other?**



Where to implement extensions ...





# Summary

- top priority items for evolution are
  - namespace scalability & high availability
  - storage backend scalability
  - replica consistency
  - fs-like behaviour
- => shared nothing everywhere
- absolutely need more QA/CI testing and fully automatic recovery for common error cases



[www.cern.ch](http://www.cern.ch)

Thank You!