AFS/OSD: massive production experience and work in progress

Hartmut Reuter
reuter@rzg.mpg.de

Hartmut Reuter, RZG, Germany
Felix Frank, DESY Zeuthen, Germany
Andrei Maslennikov, CASPUR

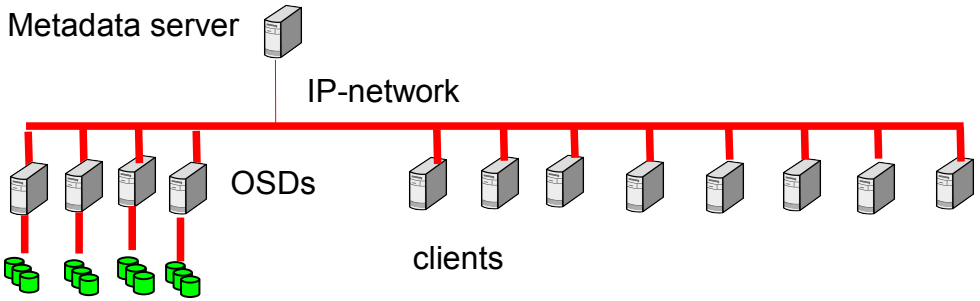
Overview

- OpenAFS + Object Storage
  - R&D project sponsored by CERN and ENEA
  - Addition of HSM features to replace MR-AFS
  - Policies
  - Experience at RZG (AFS cell ipp-garching.mpg.de)
- Work in Progress
  - Use of dCache as archival OSD
  - How to make use of cluster filesystems for AFS
    - Embedded GPFS
    - Embedded Lustre

What is object storage

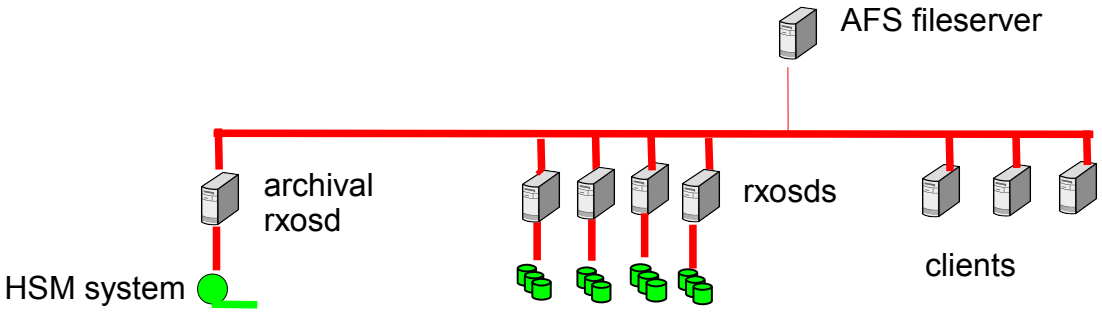
- Object storage systems are distributed filesystems which store data in object storage devices (OSD) and keep metadata in metadata servers.
- Examples of object storage systems are **Lustre**, **Panasas**
- Access on a client to a file in object storage consists in the following steps:
  - rpc to metadata server which checks permissions and returns a special ancrypted handle for each object the file consists of.
  - rpc to the OSDs to read or write the objects using the handle obtained from the metadata server
- The OSD can decrypt the handle by use of a secret shared between OSD and metadata server. The advantage of this technique is that the OSD doesn't need any knowledge about users and access rights.
- Files may be striped or mirrored over multiple OSDs

:



"AFS/OSD" or "OpenAFS + Object Storage"

- Is an extension to OpenAFS which allows files to be stored in object storage.
- The OSDs (object storage devices) for OpenAFS use the rx-protocol and are therefore called "rxosd".
- The AFS fileserver stores the metadata of files which are in object storage.
- A new ubik-database stores location and features of the OSDs.
- The decision which files should go into object storage can be based on policies.
- Files in object storage can be simple or striped over multiple OSDs, and/or have copies in multiple OSDs
- AFS/OSD allows to use HSM systems to migrate inactive files to tape

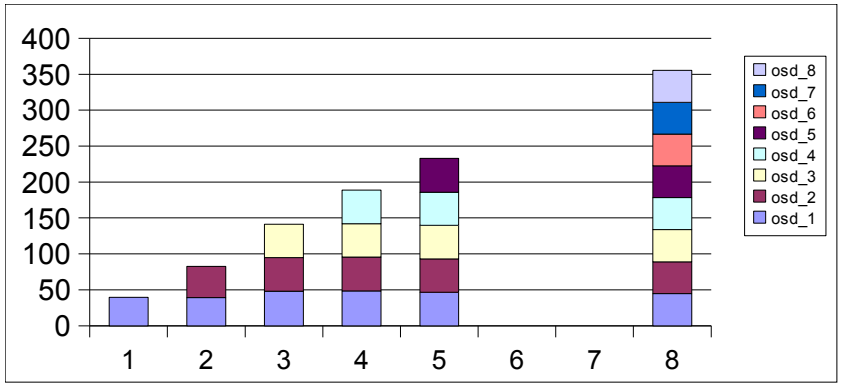


The R&D project

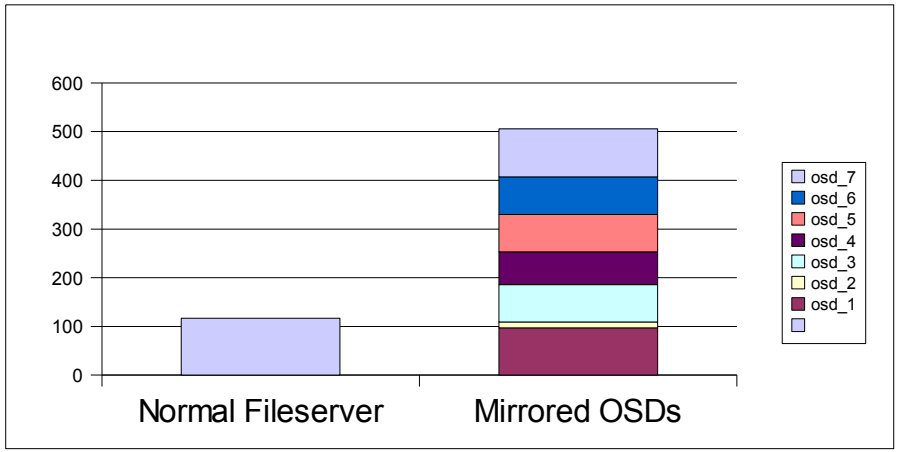
- In 2004 Rainer Többicke from CERN implemented a first version of AFS with object storage as a proof of concept. It was tested at the CASPUR StorageLab in Rome.
  - However, Rainer couldn't find the time to develop his idea any further
- In 2005 Andrei Maslennikov from CASPUR managed to bring the interested institutions and persons together to form a real project.
  - Design decisions were made on a meeting at CERN
  - Funds were raised from CERN, and ENEA to hire two system programmers to work at CASPUR:
    - Roberto Belloni and
    - Ludovico Giammarino
- Most of the development was actually done at RZG because there was the most experience with the AFS source.
- In March 2007 the project ended with a week of tests at CERN.

CERN Tests March 2007

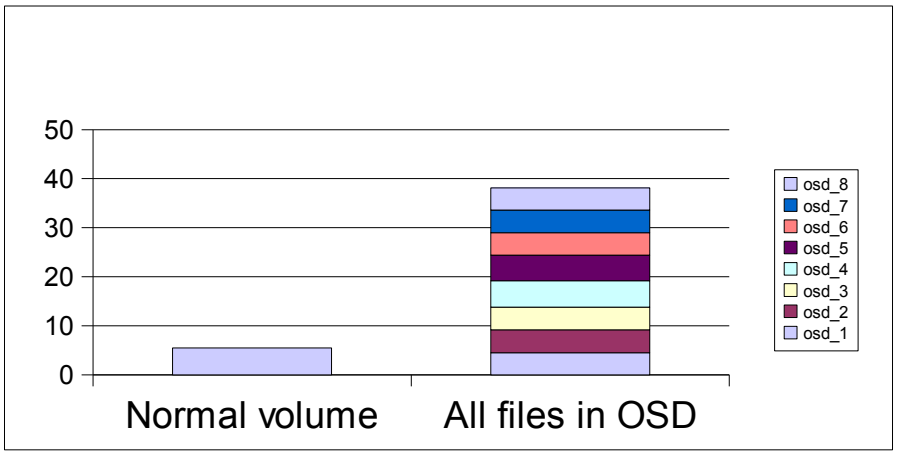
Read/Write 50 clients, variable number of OSDs



50 clients reading the same 1.3 GB file



~120 clients reading randomly in large RW-volume



Total Throughput (read+write) MB/s

State of AFS/OSD at the end of the R&D project

- In volumes with an “osd flag” files larger than a threshold value (defined in the OSD database) are automatically stored in object storage.

- Still missing from the original project:
  - Design and implementation of policies
  - Commands to replace OSDs and to backup data in OSDs

- Still missing to replace MR-AFS by AFS/OSD at RZG
  - Implementation of HSM features
    - Automatic migration and retrieval of files
    - Space control
  - Health check for data in object storage (salvage)

Development 2007-2008

- 2007: Implementation of HSM features at RZG
  - Archival OSDs which use underlying HSM systems (TSM-HSM)
  - Automatic creation of copies in archival OSDs
  - Space control: once a high water mark is reached
    - Find longest unused objects in OSDs
    - Check existence of archival copies
    - Remove from OSD to free space
  - Automatic or forced recall of files from archival OSDs
- 2008: Implementation of policies at DESY Zeuthen by Felix Frank

Policies

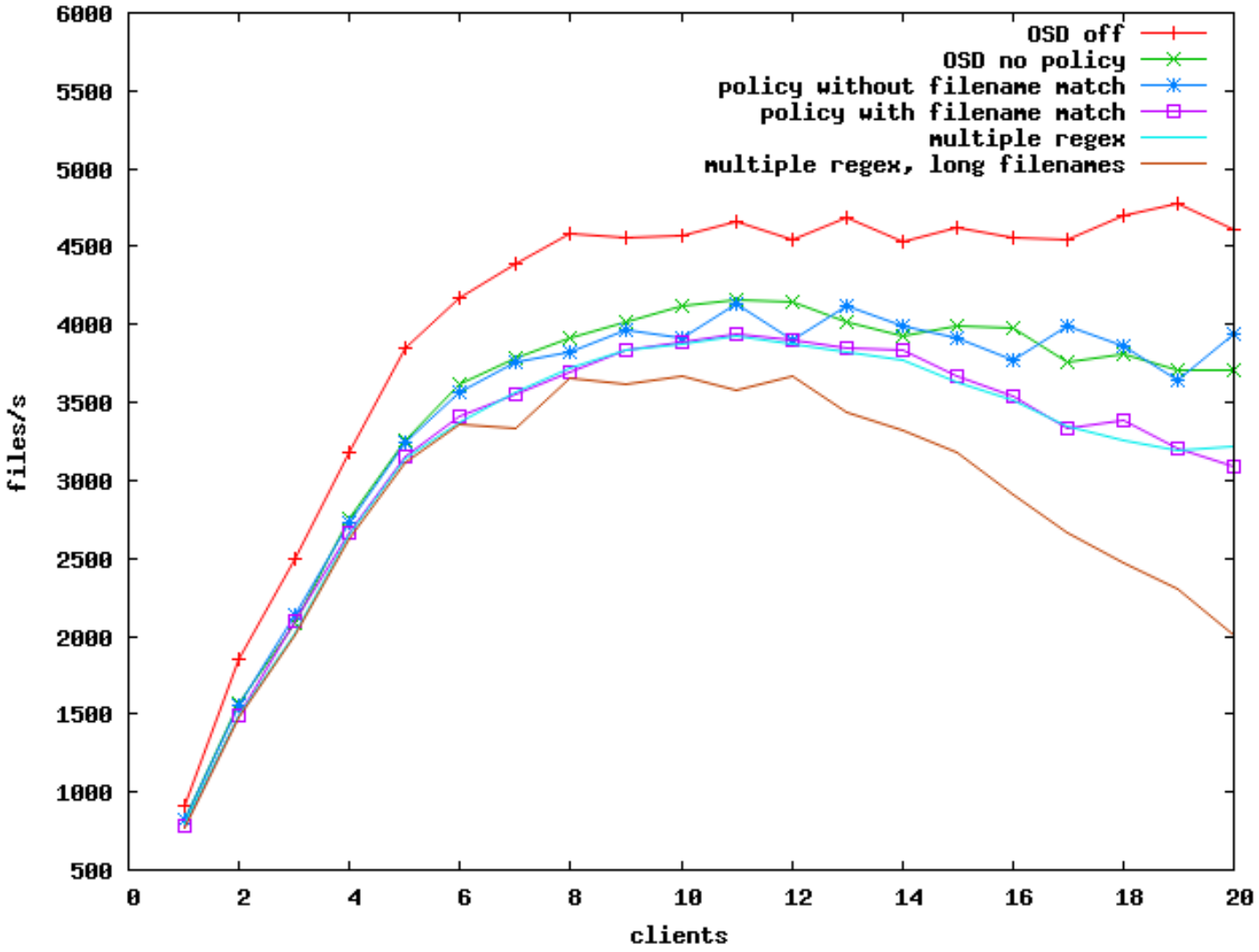
- Felix Frank at DESY Zeuthen designed and implemented the policies.
  - The policies control
    - which files go into object storage (based on name and size)
    - Whether and how they should be striped or mirrored
  - The policies are rules and are stored in the OSDDB

```
3 root

  ~'*.root' => location=osd, stop;
```

  - In the AFS volume a policy can be set to be used for the volume.
  - Additionally each directory in the volume can have a different policy number to apply for files created in this directory.
  - The AFS fileserver gets the policy-information from the OSDDB and applies it to all newly created files in volumes wich are allowed to have OSD files.

File creation rate



Evaluation of a policy has its price:

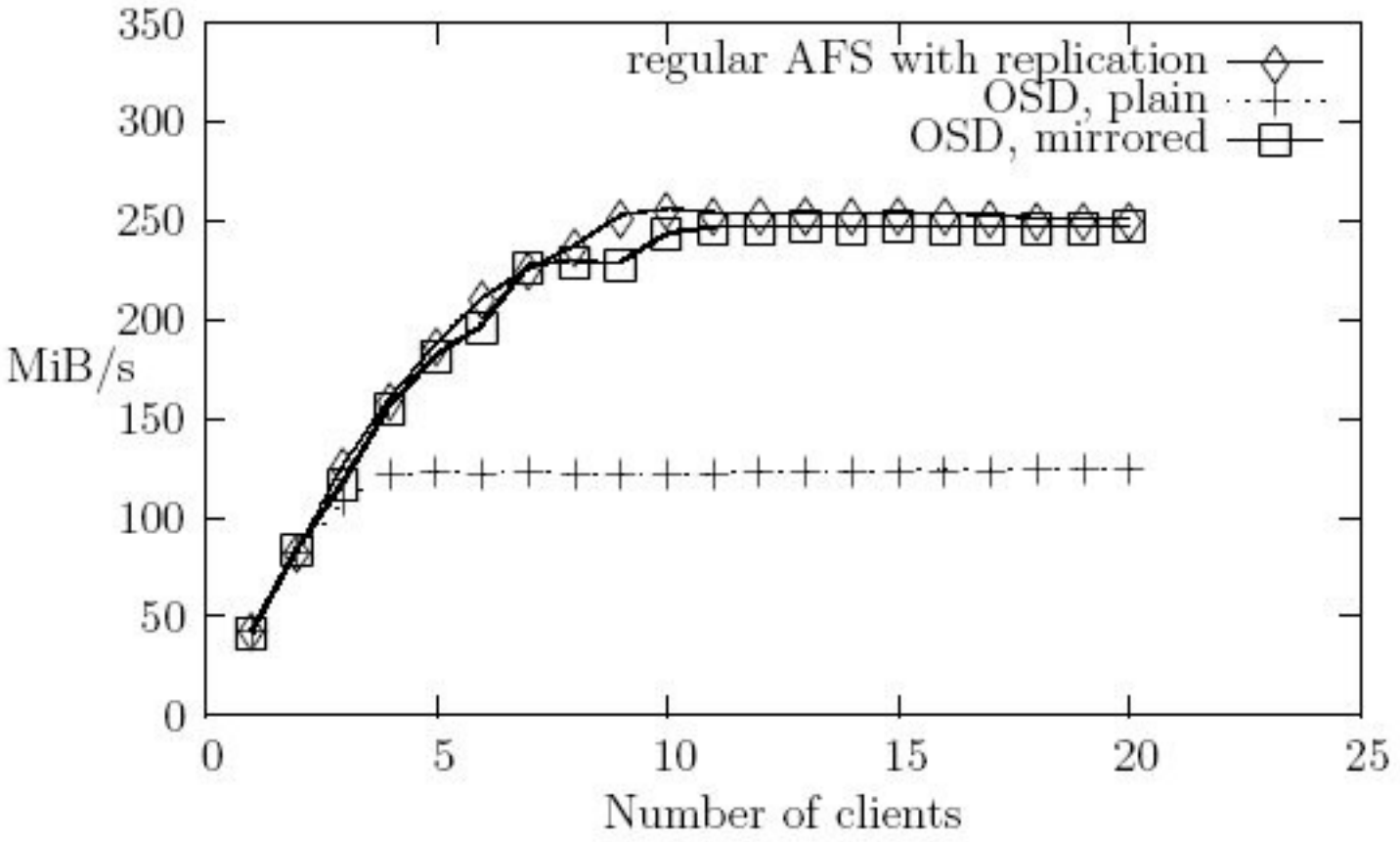
OSD off means normal AFS volume.

OSD no policy means files > 1 MB go into OSD

Most expensive is evaluation of regulare expressions on file names.

Measurement and graphic by Felix Frank.

Read throughput of a single file, either replicated or in OSD plain or mirrored

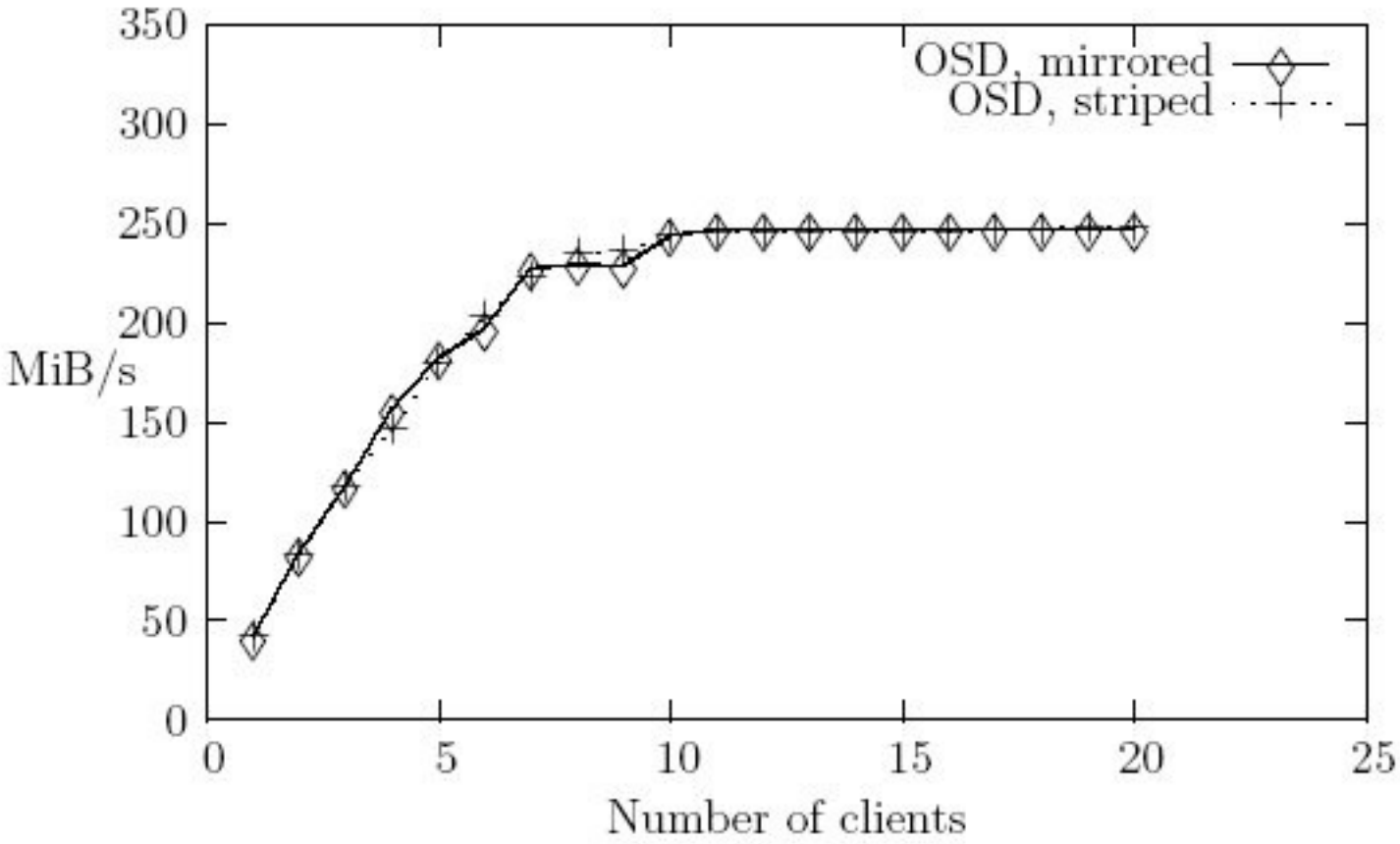


Tests with 2 server machines acting as fileservers and OSDs.

RO-replication and mirroring of OSD file are nearly equivalent for read, but mirrored OSD file can be overwritten.

Measurement and graphic by Felix Frank.

Reading a single file in OSD either mirrored or striped



Tests with 2 server machines acting as fileservers and OSDs.

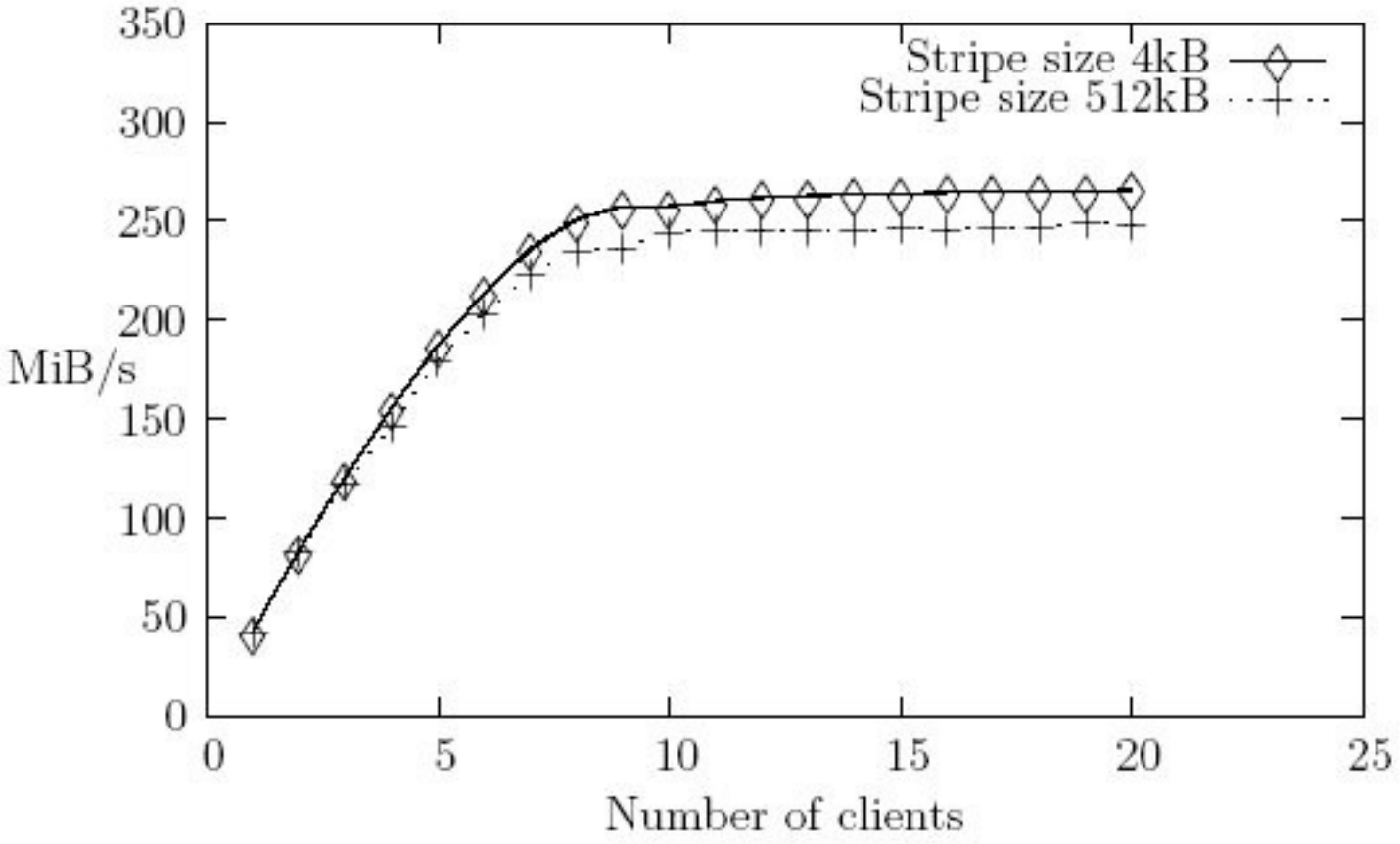
Mirroring and striping give same read throughput.

Mirroring has half write performance, but protects against loss of a disk system.

Measurement and graphic by Felix Frank.

Dependency on stripe size

Read throughput of a single striped file



For me it was surprising that read performance gets better with smaller stripe size!

Measurement and graphic by Felix Frank.

Deployment of AFS/OSD

- 2007 RZG moves all “normal” AFS volumes to AFS/OSD
  - AFS/OSD should be 100% compatible to OpenAFS
  - Some additional features available
- 2008 RZG moves all other volumes from MR-AFS to AFS/OSD
  - Data in HSM (TSM-HSM) remain where they are
  - Conversion of MR-AFS metadata to AFS/OSD metadata on the fly
- 2008 DESY Zeuthen starts evaluating AFS/OSD
- 2008 Test cell at DESY Hamburg to develop connection with dCache

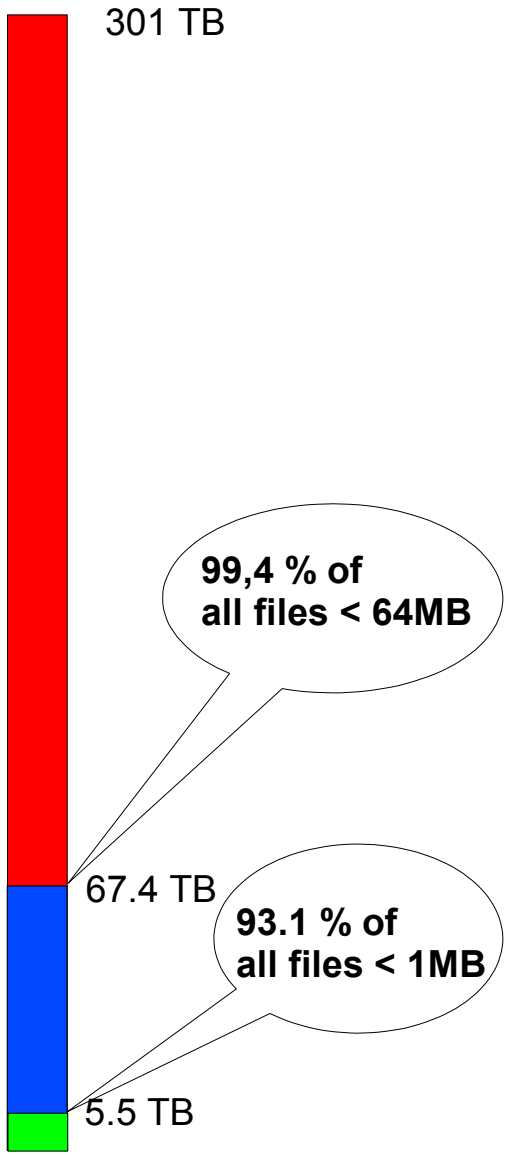
RZG's AFS cell „ipp-garching.mpg.de“

Real Production Environment for AFS/OSD

40	fileservers with 195 TB disk space
21	non-archival OSDs with 100 TB disk space
2	archival OSD with 2 independent HSM systems TSM-HSM, They will be replaced by HPSS by the end of the year
27000	AFS volumes
8700	users
300	TB total data
4	TB data written per day
8	TB data read per day

File Size Histogram

File Size Range	Files	%	run %	Data	%	run %
0 B - 4 KB	64853397	50.31	50.31	80.867 GB	0.03	0.03
4 KB - 8 KB	10368948	8.04	58.35	56.730 GB	0.02	0.04
8 KB - 16 KB	9157574	7.10	65.46	97.675 GB	0.03	0.08
16 KB - 32 KB	10300431	7.99	73.45	215.601 GB	0.07	0.15
32 KB - 64 KB	7942739	6.16	79.61	363.820 GB	0.12	0.26
64 KB - 128 KB	6006376	4.66	84.27	523.980 GB	0.17	0.43
128 KB - 256 KB	4005120	3.11	87.37	709.525 GB	0.23	0.66
256 KB - 512 KB	4360813	3.38	90.76	1.484 TB	0.49	1.16
512 KB - 1 MB	3067697	2.38	**93.14**	1.980 TB	0.66	**1.82**
1 MB - 2 MB	2051828	1.59	94.73	2.851 TB	0.95	2.76
2 MB - 4 MB	2033889	1.58	96.31	5.361 TB	1.78	4.55
4 MB - 8 MB	1830856	1.42	97.73	9.617 TB	3.20	7.74
8 MB - 16 MB	1112483	0.86	98.59	12.125 TB	4.03	11.77
16 MB - 32 MB	598374	0.46	99.05	12.370 TB	4.11	15.88
32 MB - 64 MB	474283	0.37	**99.42**	19.651 TB	6.53	**22.41**
64 MB - 128 MB	306544	0.24	99.66	25.633 TB	8.52	30.93
128 MB - 256 MB	165577	0.13	99.79	28.363 TB	9.43	40.36
256 MB - 512 MB	130065	0.10	99.89	46.954 TB	15.60	55.96
512 MB - 1 GB	125479	0.10	99.99	84.709 TB	28.15	84.11
1 GB - 2 GB	13216	0.01	100.00	18.967 TB	6.30	90.41
2 GB - 4 GB	2264	0.00	100.00	5.896 TB	1.96	92.37
4 GB - 8 GB	873	0.00	100.00	5.244 TB	1.74	94.11
8 GB - 16 GB	599	0.00	100.00	5.652 TB	1.88	95.99
16 GB - 32 GB	174	0.00	100.00	3.784 TB	1.26	97.25
32 GB - 64 GB	120	0.00	100.00	5.306 TB	1.76	99.01
64 GB - 128 GB	21	0.00	100.00	1.725 TB	0.57	99.59
128 GB - 256 GB	4	0.00	100.00	611.258 GB	0.20	99.78
256 GB - 512 GB	2	0.00	100.00	667.311 GB	0.22	100.00
Totals:	128909746 Files			300.938 TB		



Storage Usage


```
   Storage usage:
-------------------------------------------------------------------
                1 local_disk  125057252 files      58.750 TB
  arch. Osd     4 raid6         1081100 objects     3.362 TB
  arch. Osd     5 tape          3954903 objects   242.597 TB
        Osd     8 afs16-a         18731 objects     3.237 TB
        Osd     9 mpp-fs9-a      357897 objects     8.169 TB
        Osd    10 afs4-a          16587 objects     3.053 TB
        Osd    11 w7as            96750 objects     1.689 TB
        Osd    12 afs1-a         252390 objects     1.302 TB
  arch. Osd    13 hsmgpfs       1325048 objects   179.208 TB
        Osd    14 afs6-a           7791 objects   978.469 GB
        Osd    17 mpp-fs8-a       26663 objects     1.175 TB
        Osd    23 mpp-fs11-gj     32187 objects     2.625 TB
        Osd    24 mpp-fs12-a      54220 objects     3.487 TB
        Osd    25 mpp-fs13-a      49844 objects     3.597 TB
        Osd    34 afs17-gb       110104 objects     4.598 TB
        Osd    35 afs18-ga        83807 objects     4.490 TB
        Osd    36 afs21-ge        72709 objects     4.627 TB
        Osd    37 afs22-gf        54286 objects     4.600 TB
        Osd    38 afs19-gc        85792 objects     4.610 TB
        Osd    39 afs20-gd        65615 objects     4.629 TB
        Osd    40 afs23-gg        58011 objects     4.609 TB
        Osd    41 mpp-fs15-gi     48537 objects     3.319 TB
        Osd    42 mpp-fs14-gh     60169 objects     3.395 TB
        Osd    43 mpp-fs10-gk     48560 objects     3.020 TB
-------------------------------------------------------------------
Total                        133018953 objects   555.124 TB
```

Files in the Fileserver, not in OSDs

primary TSM-HSM

secondary TSM-HSM

Data without a Copy


```
Data without a copy:
-------------------------------------------------------------------
if !replicated: 1 local_disk  125057252 files      58.750 TB
  arch. Osd     4 raid6               1 objects     7.954 MB
  arch. Osd     5 tape           972784 objects    57.681 TB
        Osd     9 mpp-fs9-a         288 objects   145.075 GB
        Osd    12 afs1-a             52 objects   103.994 MB
  arch. Osd    13 hsmgpfs            22 objects    14.575 GB
        Osd    17 mpp-fs8-a          20 objects    10.266 GB
        Osd    23 mpp-fs11-gj        35 objects    17.455 GB
        Osd    24 mpp-fs12-a         34 objects    17.137 GB
        Osd    25 mpp-fs13-a         35 objects    17.802 GB
        Osd    34 afs17-gb          273 objects    17.461 GB
        Osd    35 afs18-ga          174 objects    15.614 GB
        Osd    36 afs21-ge          188 objects    21.597 GB
        Osd    37 afs22-gf          188 objects    32.217 GB
        Osd    38 afs19-gc          199 objects    56.445 GB
        Osd    39 afs20-gd          208 objects    72.576 GB
        Osd    40 afs23-gg          229 objects    86.408 GB
        Osd    41 mpp-fs15-gi        34 objects    17.171 GB
        Osd    42 mpp-fs14-gh        37 objects    18.355 GB
        Osd    43 mpp-fs10-gk        38 objects    19.715 GB
-------------------------------------------------------------------
Total                        126032091 objects   116.999 TB
```

Files not yet copied To secondary TSM-HSM

After bad experiences with TSM-HSM in February 2008 we decided to copy all data into a secondary TSM-HSM.

We have mostly longtime archives including „cultural heritage“.

Data Transfer

```
> xfer
---snip---
fs afs17.rzg.mpg:      72.64 gb  rcvd      43.16 gb  sent per day (3.428044 days)
fs afs18.rzg.mpg:      76.10 mb  rcvd       2.92 tb  sent per day (3.427303 days)
---snip---
vos afs17.rzg.mp:     119,74 gb  rcvd       3.25 gb  sent per day (3.428044 days)
vos afs17.rzg.mp:       2.23 gb  rcvd      43.41 mb  sent per day (3.427303 days)
---snip---
osd           42:      48.23 gb  rcvd     210.64 gb  sent per day (3.427488 days)
osd           43:      78.61 gb  rcvd     224.65 gb  sent per day (3.427812 days)
Fileserver xfer:        1.15 tb  rcvd       4.42 tb  sent per day
Volserver xfer:       433.71 gb  rcvd     422.21 gb  sent per day
Rxosd xfer:             2.41 tb  rcvd       3.31 tb  sent per day
Total transfer:         3.99 tb  rcvd       8.15 tb  sent per day
```

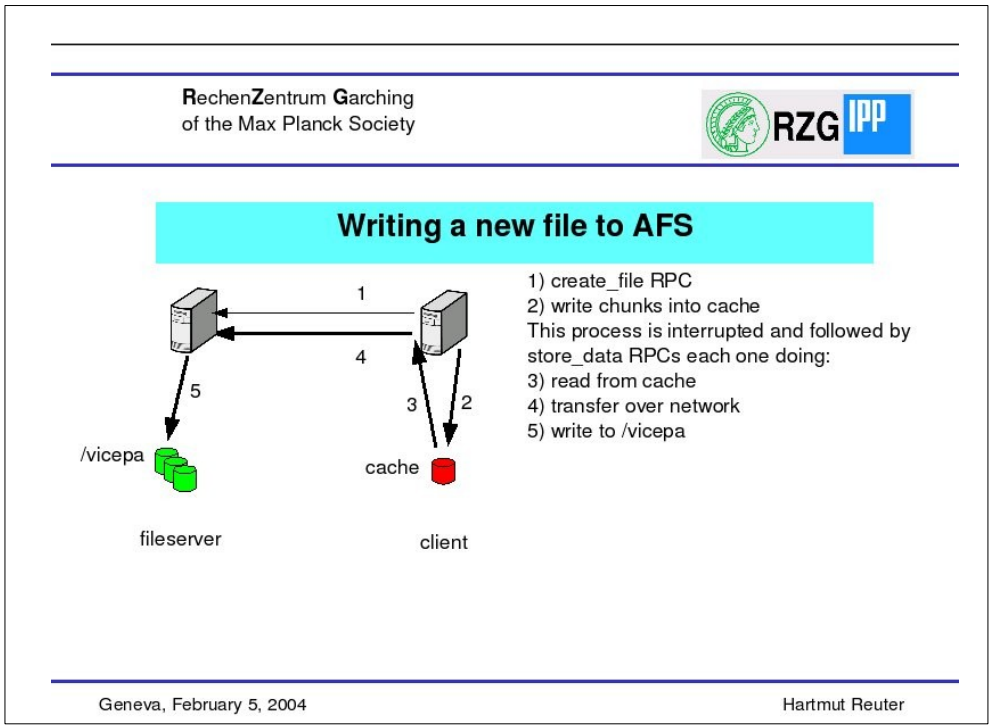
- Fileserver transfer
  - Very active non-OSD volumes with lots of small files
- Volserver transfer
  - Nightly „vos release“ to obtain actual RO volumes (backup)
- Rxosd transfer
  - Direct read/write from clients, but
  - contains also creating of copies on archival OSDs

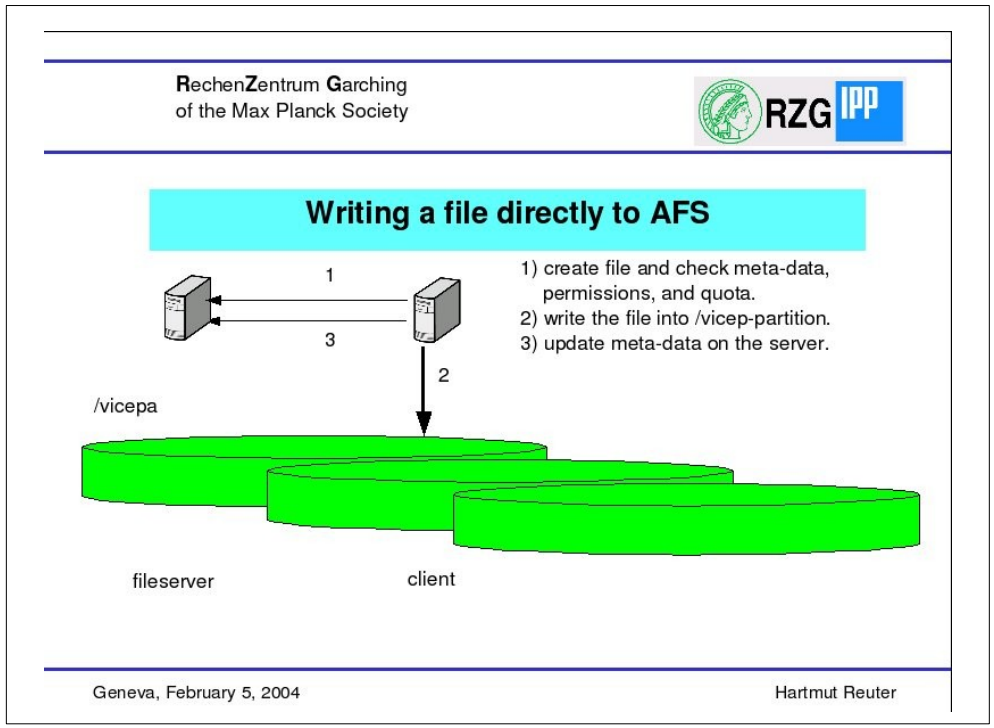
dCache as archival OSD

- Sites with **dCache** (DESY) could backup objects in OSDs into dCache
- rxosd must access data in dCache through dcap-library calls
- Special handling:
  - Files can be opened for write only once in dCache
- Support for both:
  - old **dCache** with /pnfs filesystem
  - **chimera** without /pnfs
- Running in a AFS-OSD-test cell in Hamburg
- Not yet ready for production (still some open issues with **chimera**)

AFS and Cluster Filesystems

- Cluster filesystems such as Lustre or GPFS are much faster than AFS, especially in connection with fast networks such as Infiniband.
- Already five years ago I gave a talk at CERN about using shared filesystems for AFS
  - At that time a prototype implementation which bypassed the cache.





AFS and Cluster Filesystems

- The actual implementation uses the cache (preferably memory cache).
- The rxosd stores the objects in a shared filesystem (Lustre or GPFS)
  - The client contacts as usual the fileserver to get the OSD location,
  - but reads or writes the objects directly through the shared filesystem
- You should, of course, export your AFS partitions only in trusted environments.
- The technique can also be used independently of AFS/OSD for AFS fileserver partitions,
  - but shared filesystems do not perform well for millions of small files and file creation and deletion are slow.
  - Therefore I would recommend to use it only for OSDs where you typically put not that many but large files.
- It makes your data in Lustre or GPFS accessible world-wide and on all platforms
- It can be used to add HSM capabilities to Lustre

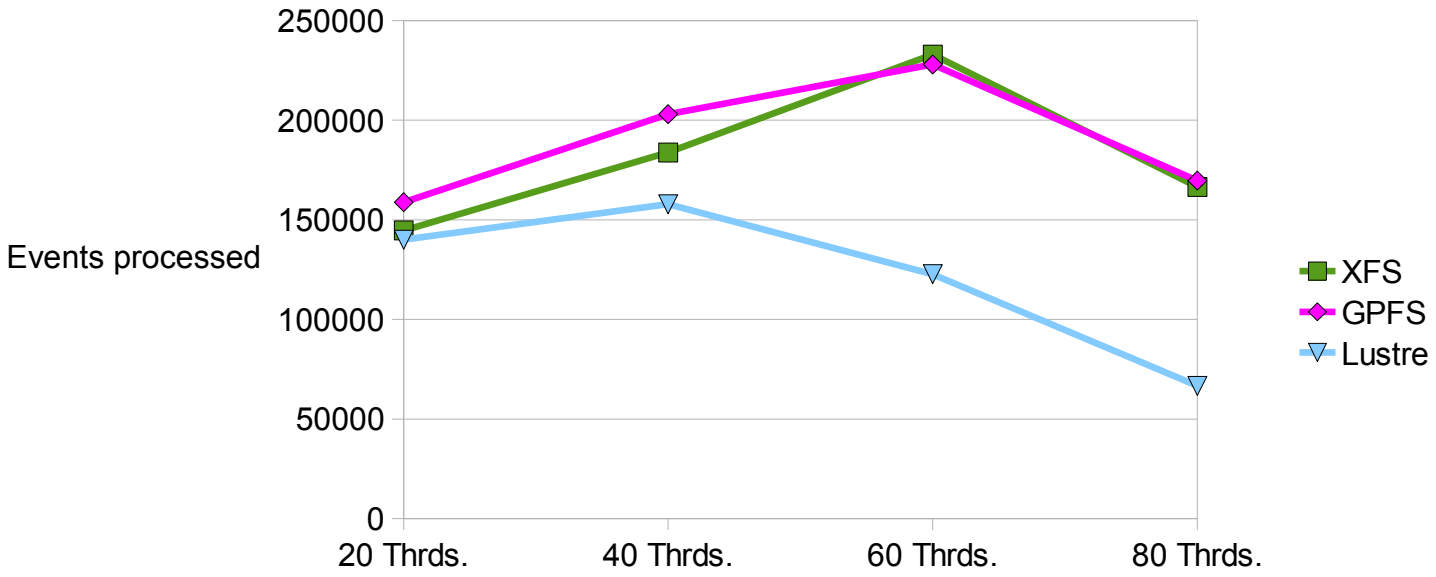
Tests at Karlsruhe (gridka.de)

- Andrei Maslennikov did all the tests
- 10 clients with GigE interface run ATLAS software
  - Different number of threads per client (2, 4, 6, 8)
  - Goal: get processed as many events as possible
- Server:

```
FEBRUARY/MARCH 2009 - CERN DISK SERVER - VERSION 2 (RAID-5)
  -  Intel E5405 @ 2.00GHz (2x4=8 cores in total)
  -  8 GB of RAM
  -  4 GigE outlets in bonding/alb (measured 450+ MB/sec aggregate)
  -  3W 9650SE 3xRAID-5(256k), 500+ MB/sec aggr
  -  RHEL 4.7 (64 bit)
  -  Lustre 1.6.6 (no checksumming)
```

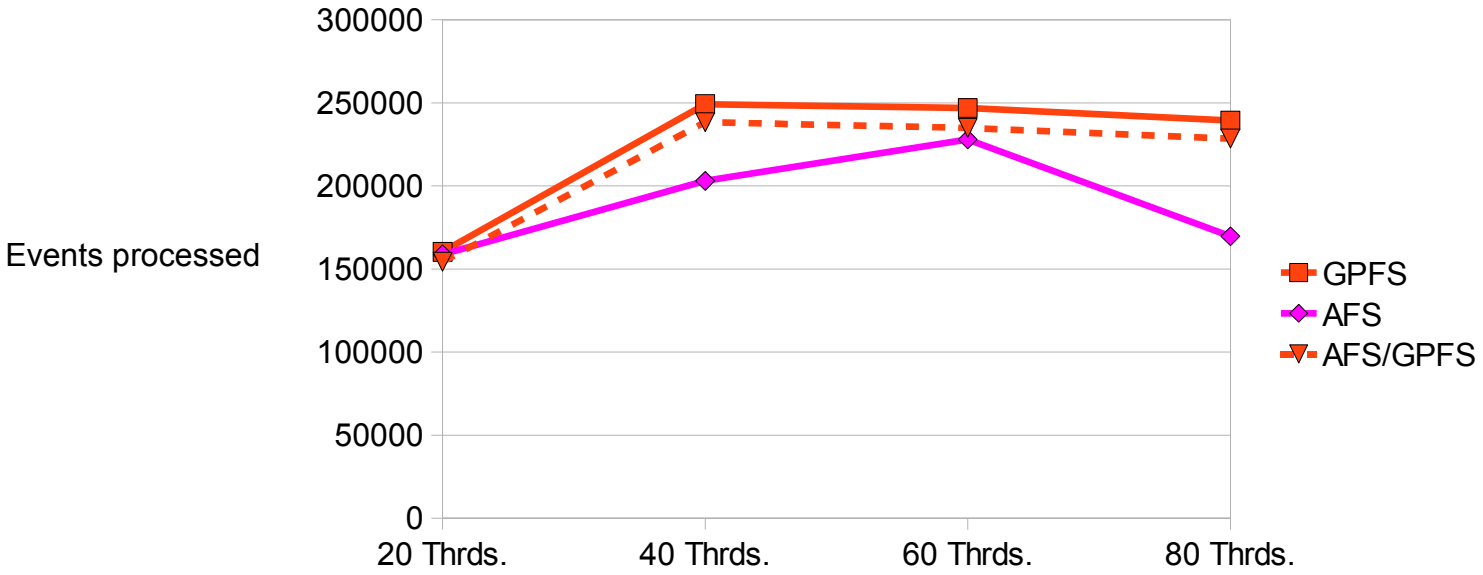
AFS server filesystem XFS, GPFS, Lustre

- The following filesystems were used for the /vicepa partition on the AFS server
  - XFS (is used in many AFS cells on Linux servers, also at RZG)
  - GPFS (Read/write performance good, file creation and deletion slow)
  - Lustre (Obviously not that good as a local AFS server filesystem)
- Tests with normal AFS client access through rx-protocol
  - Goal: get processed as many events as possible



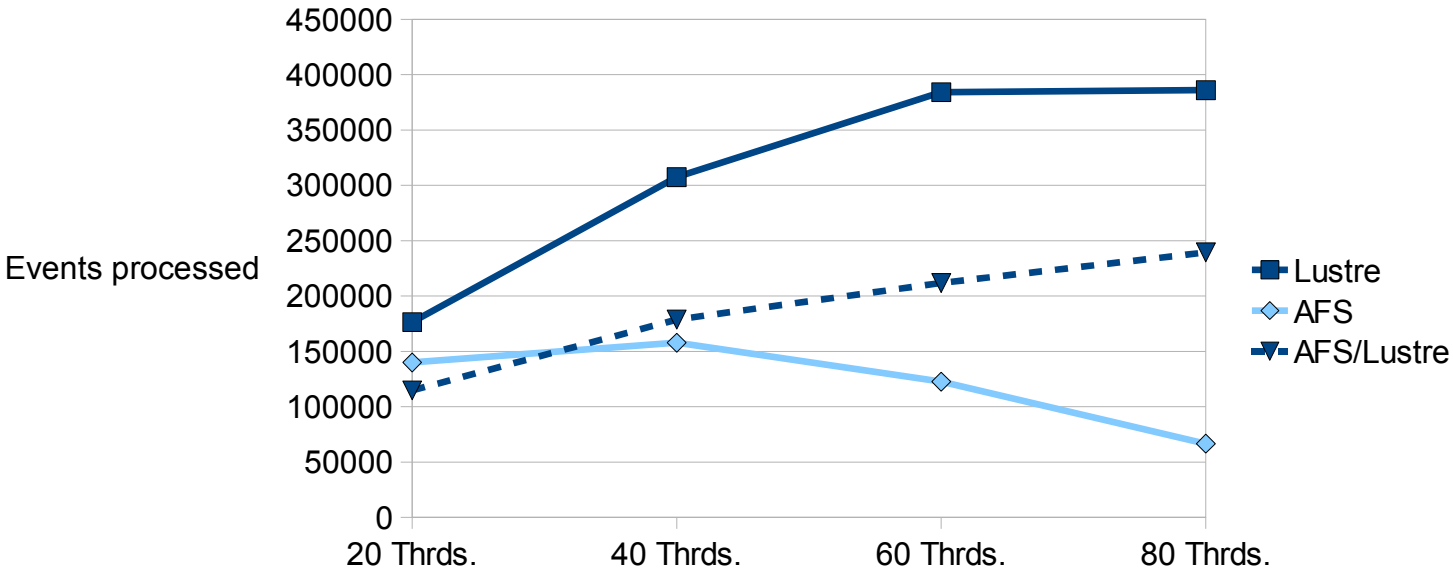
Embedded GPFS

- Test environment at Karlsruhe (gridka.de)
  - 1 server AFS-/vicepa-Partition with GPFS filesystem
  - 10 client machines, each running 2, 4, 6, 8 threads
  - Access either GPFS native or AFS native or AFS using GPFS inside client
- Tests done by Andrei Maslennikov
  - Goal: get processed as many events as possible



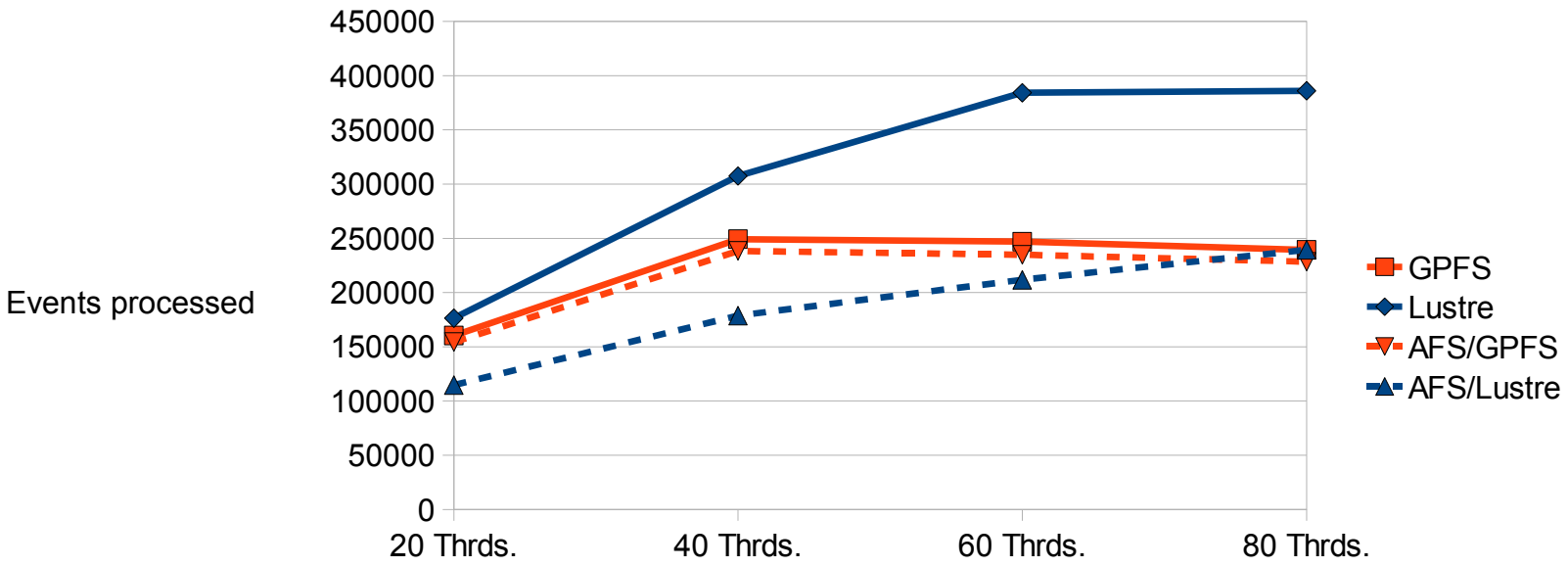
Tests with AFS and Lustre

- Test environment at Karlsruhe (gridka.de)
  - 1 server Lustre and AFS, /vicepa-Partition Lustre
  - 10 client machines, each running 2, 4, 6, 8 threads
  - Access either Lustre native or AFS native or AFS using Lustre inside client
  - Because of a problem with Lustre on the client the design had been changed to do open and close before and after reading of each AFS chunk
- Tests done by Andrei Maslennikov
  - Goal: get processed as many events as possible



AFS/GPFS compared to AFS/Lustre at gridka

- Native Lustre is clearly faster than native GPFS
- AFS/GPFS is faster than AFS/Lustre for low number of threads
  - No big performance difference between native GPFS and AFS/GPFS,
  - but data in AFS/GPFS are accessible all over the world and on all platforms
  - Results unfair against Lustre because of modified AFS-client



Embedded Lustre at DESY Zeuthen

- While testing AFS with Lustre at DESY Zeuthen Felix Frank found a solution for the Lustre problem seen at gridka getting rid of the many open/close calls. Basically same client as in gridka for GPFS
- Lustre (1.6.7 or 1.8) configuration:
  - OSS on Dell 1950, 2 2.33 Ghz CPUs 8 GB memory, SL 5.3
  - OST on RAID6 PERC6 controller (13+2) 1TB sata disks in Dell MD1000 enclosure connected with 2 Quadlane SAS cables.
  - No striping over OSTs, DDR Infiniband to clients

```
write of 34359738368 bytes took 151.388 sec.
close took 0.134 sec.
Total data rate = 221449 Kbytes/sec. for write
```

```
read of 34359738368 bytes took 127.308 sec.
close took 0.000 sec.
Total data rate = 263564 Kbytes/sec. for read
```

2 reads embedded Lustre at DESY Zeuthen

- Also reads on two clients at the same time are fast:

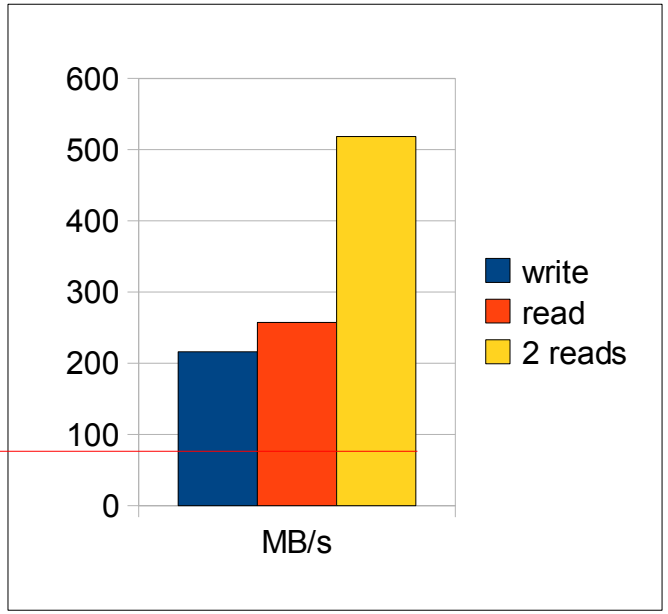
```
Total data rate = 270145 Kbytes/sec. for read
```

```
Total data rate = 260697 Kbytes/sec. for read
```

The total throughput of ~ 518 MB/s is not too far from native Lustre (537 MB/s)

- The AFS-clients had 256 MB memory cache with 1 MB chunk size.
  - The performance depends on the chunk size:
  - With 64 KB chunks it goes down to 200 MB/s

Typical speed of normal AFS client



Summary

- AFS/OSD
  - is now in full production including HSM and policies
- In AFS embedded shared filesystems (Lustre and GPFS) are not yet in production, but a promising technique
  - To hide GPFS and Lustre to the user without loosing much performance
  - To give worldwide secure access to GPFS and Lustre
  - To give other platforms (Windows, MacOS, Solaris…) acces to Lustre and GPFS
  - To add HSM to Lustre (GPFS can already have HPSS or TSM-HSM)

How to get AFS/OSD

- Right now AFS/OSD is still not in the official OpenAFS distribution and CVS
- AFS/OSD is maintained in the subversion server of DESY
  - You can view the source and patches at

```
https://svnsrv.desy.de/viewvc/openafs-osd/trunk/openafs
```

  - or check it out with

```
svn co https://svnsrv.desy.de/public/openafs-osd/trunk/openafs
```

  - To get full functionality configure with

```
--enable-namei-fileserver --enable-largefile-fileserver
--enable-object-storage --enable-vicep-access
```

Questions or comments?

Thank you