Update to OpenAFS + Object Storage

Hartmut Reuter
reuter@rzg.mpg.de

Agenda

- What is AFS/OSD?
- What is new this year?
- Current usage of AFS/OSD
- Integration of AFS/OSD into OpenAFS

What is AFS/OSD?

In few words because I talked about this already on many AFS workshops:

- AFS/OSD is an extension to OpenAFS which

  1. allows to store files in (object storage) instead of the fileserver's partition. The object storage consists in many disk servers running „rxosd“.
  2. brings HSM functionality to AFS if an archival “rxosd” uses an underlying HSM system. This feature offers „infinite“ disk space.
  3. gives fast access to AFS files in clusters with “embedded filesystems“. A shared filesystem such as GPFS or Lustre is used by an „rxosd“ and the clients in the cluster can access the data directly.

A talk describing AFS/OSD was given in Newark and Graz 2008:
http://workshop.openafs.org/afsbpw08/talks/thu_3/Openafs+ObjectStorage.pdf

A talk describing “Embedded Filesystems” was given in Stanford 2009:
http://workshop.openafs.org/afsbpw09/talks/thu_2/Embedded_filesystems_opt.pdf

A tutorial about AFS/OSD was given in Rome 2009:
http://www.dia.uniroma3.it/~afscon09/docs/reuter.pdf

What's new this year?

- There were no big changes in the 1.4 version of AFS/OSD during the last year
  - Some bug-fixes,
  - A new subcommand „fs listosdcand“ shows which OSDs a file could go to
  - svn checkout http://svnsrv.desy.de/public/openafs-osd/trunk/openafs/.
- The former openafs-1.5-osd tree has become the openafs-1.6-osd tree
  - It is in sync with the openafs 1.6 tree (now 1.6.0).
  - The RPC interface of the fileserver and rxosd has been redesigned to reflect the objections I got from the gerrit reviewers
  - It is kept is fully backward compatible to the 1.4-osd RPCs.
  - It is not yet in production except for the rxosd which is the bridge to our new HSM system HPSS.
  - svn checkout http://pfanne.rzg.mpg.de/svn/RZG-AFS/trunk/afs_kerberos/openafs-1.6-osd/
- Sine Nomine created a mailing list “rxosd-devel@sinenomine.net” which any interested developer should join.

Integration into git master

- After the conference in Pilsen I started to submit code to gerrit
  - As a 1st step all clients should support rxosd and vicep-access
  - As a 2nd step it should be possible to configure the server to support rxosd and vicep-access. But the default should remain no support.
- The code in the git master will be based on the 1.6-version of AFS/OSD
  - but not contain all the backward compatibility with our 1.4 tree we have in our openafs-1.6-osd tree needed at RZG to support older client versions
- During code review many changes got into this code and our 1.6-osd
  - Conflict with IANA about port numbers 7011 and 7012 solved by using in future 7006 for OSDDB and store OSD port numbers in the OSDDB
  - Non T10-based object description in new paramter lists (64bit numbers for Ids, uniquifiers...)
  - Unions to allow different versions in the same RPC

My problems with gerrit

- I know it's possible to define dependencies between patches when sending stuff to gerrit, but I never got this right.
  - Therefore my patches were seen as independent and build didn't work
- I now try to make a monster-patch which contains all necessary to have the client ready for AFS/OSD inclusive embedded filesystems.
  - This patch I sync constantly with the other changes in git master
- I compile this tree and try to test it once in a while
  - Because I want very everything works as expected before submitting it
- My client - build from master and my patches - crashes.
  - But the client build from master without my patches crashes, too!
- Tell me when the master is in a state to allow testing the client!!!

How to go ahead

- There were many objections from the reviewers against the code I already sent to gerrit
  - I accept: nearly all of these objections were reasonable
  - So I had to make updates to my changes to get them merged.
  - This takes long time and before all changes are in gerrit it's not easy – also for the reviewers – to see what they are for.
- Therefore I now would like to get things into the master as soon as it is clear
  - that it builds
  - the builded code runs as expected
  - all other things work as before
- And better do all – certainly necessary – improvements after this
  - because it's much easier and better to understand what is affected by them

Usage of AFS/OSD

- So, why could sites want to deploy AFS/OSD?
    1. Better distribution of data on on-line storage (the original idea from CERN)
    2. To have HSM for AFS (datamigration onto tapes)
    3. Fast access to AFS files in clusters with “embedded filesystems“
    4. Other goodies offered configuring without --enable-object-storage and without --enable-vicep-access
- DESY Zeuthen was interested in case 3 because they wanted to deploy Lustre
    - With the uncertain future of Lustre they stopped this project
- ENEA made last year some tests to 3 with GPFS as “embedded filesystem“
- PSI (Paul-Scherrer-Institut) made in June tests to 2 and 3
    - with SamFS as HSM system and GPFS as “embedded filesystem“
- Presently the only site using AFS/OSD in production is RZG with use case 2
    - Migrating from TSM-HSM to HPSS as HSM system

AFS cell „ipp-garching.mpg.de“

37 fileservers in 3 towns with 215 TB disk space

22 non-archival OSDs with 134 TB disk space

3 archival OSD two with TSM-HSM, the 3rd one with HPSS.

(TSM-HSM will comletely be replaced by HPSS during the next year)

31000 volumes

8700 users

>700 TB total data

4.8 TB data written per day

3.5 TB data read per day

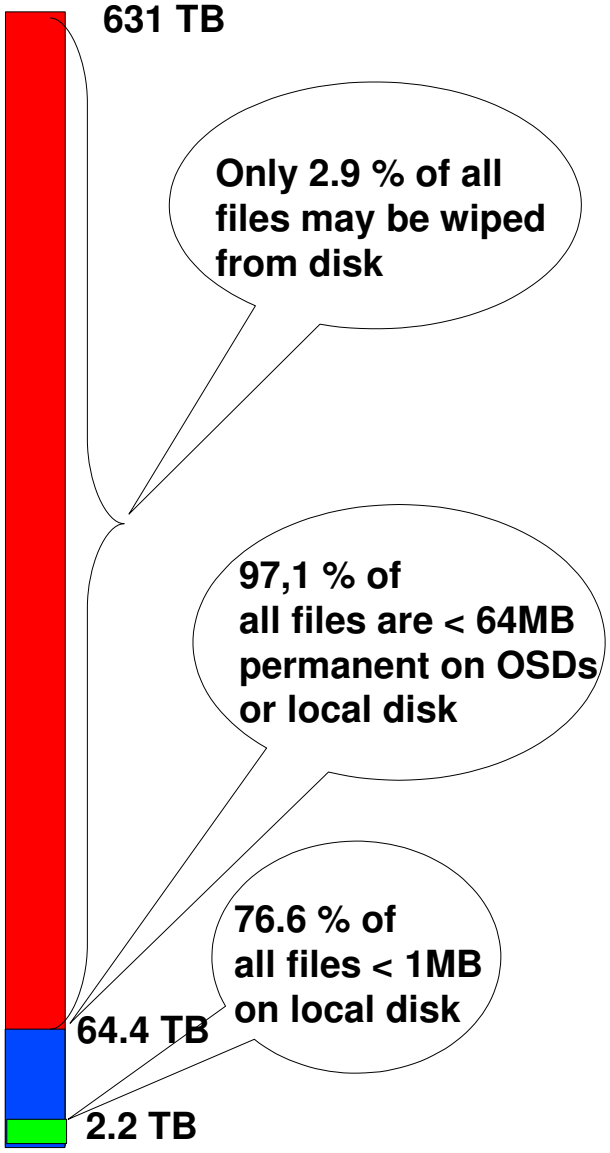
“classical” non-OSD-Volumes

File Size Range	Files	%	run %	Data	%	run %
0 B - 4 KB	75092720	52.70	52.70	95.622 GB	0.14	0.14
4 KB - 8 KB	12384532	8.69	61.39	68.120 GB	0.10	0.25
8 KB - 16 KB	9714265	6.82	68.21	105.963 GB	0.16	0.40
16 KB - 32 KB	10868161	7.63	75.84	227.009 GB	0.34	0.74
32 KB - 64 KB	8629520	6.06	81.89	395.656 GB	0.59	1.34
64 KB - 128 KB	7845625	5.51	87.40	681.917 GB	1.02	2.36
128 KB - 256 KB	4618266	3.24	90.64	809.247 GB	1.21	3.57
256 KB - 512 KB	4224098	2.96	93.60	1.446 TB	2.22	5.78
512 KB - 1 MB	3544773	2.49	96.09	2.190 TB	3.36	9.14
1 MB - 2 MB	1875861	1.32	97.41	2.561 TB	3.93	13.07
2 MB - 4 MB	1344520	0.94	98.35	3.506 TB	5.37	18.44
4 MB - 8 MB	1138671	0.80	99.15	6.122 TB	9.38	27.83
8 MB - 16 MB	441430	0.31	99.46	4.661 TB	7.14	34.97
16 MB - 32 MB	361043	0.25	99.71	7.635 TB	11.70	46.67
32 MB - 64 MB	258720	0.18	99.90	10.570 TB	16.20	62.87
64 MB - 128 MB	90520	0.06	99.96	7.163 TB	10.98	73.85
128 MB - 256 MB	33589	0.02	99.98	5.658 TB	8.67	82.52
256 MB - 512 MB	18601	0.01	100.00	6.319 TB	9.68	92.20
512 MB - 1 GB	4229	0.00	100.00	2.523 TB	3.87	96.07
1 GB - 2 GB	1040	0.00	100.00	1.522 TB	2.33	98.40
2 GB - 4 GB	113	0.00	100.00	315.388 GB	0.47	98.88
4 GB - 8 GB	45	0.00	100.00	250.664 GB	0.38	99.25
8 GB - 16 GB	46	0.00	100.00	424.992 GB	0.64	99.89
16 GB - 32 GB	3	0.00	100.00	75.739 GB	0.11	100.00
Totals:	142490391 Files			65.255 TB		

- The command “vos traverse” shows a file size histogram over all servers and/or servers you selected.
- With the option „-noosd“ it examins only non-OSD-volumes, with „-onlyosd“ only OSD-volumes.

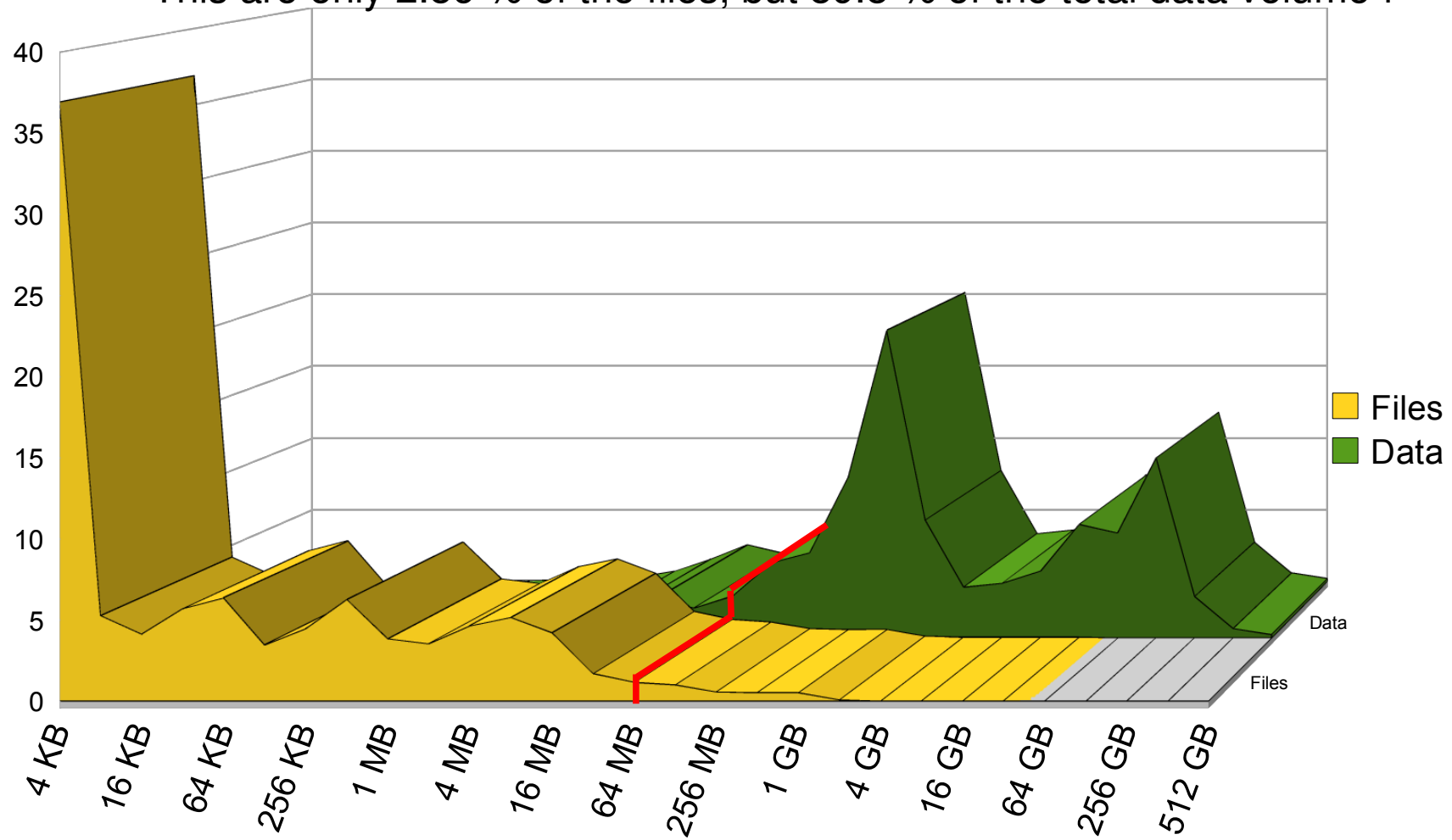
OSD-Volumes

File Size Range	Files	%	run %	Data	%	run %
0 B - 4 KB	12905742	36.93	36.93	14.299 GB	0.00	0.00
4 KB - 8 KB	1849181	5.29	42.22	9.801 GB	0.00	0.00
8 KB - 16 KB	1449922	4.15	46.37	16.127 GB	0.00	0.01
16 KB - 32 KB	2000494	5.72	52.09	44.678 GB	0.01	0.01
32 KB - 64 KB	2229522	6.38	58.47	98.103 GB	0.02	0.03
64 KB - 128 KB	1217728	3.48	61.96	104.545 GB	0.02	0.04
128 KB - 256 KB	1564760	4.48	66.43	279.069 GB	0.04	0.09
256 KB - 512 KB	2196850	6.29	72.72	754.536 GB	0.12	0.20
512 KB - 1 MB	1350153	3.86	76.58	943.227 GB	0.15	0.35
1 MB - 2 MB	1243395	3.56	80.14	1.800 TB	0.29	0.64
2 MB - 4 MB	1629340	4.66	84.80	4.433 TB	0.70	1.34
4 MB - 8 MB	1809733	5.18	89.98	9.753 TB	1.54	2.88
8 MB - 16 MB	1480290	4.24	94.22	16.123 TB	2.55	5.44
16 MB - 32 MB	604566	1.73	95.95	12.327 TB	1.95	7.39
32 MB - 64 MB	415553	1.19	97.14	17.275 TB	2.74	10.12
64 MB - 128 MB	360140	1.03	98.17	31.006 TB	4.91	15.03
128 MB - 256 MB	209233	0.60	98.77	35.250 TB	5.58	20.62
256 MB - 512 MB	183921	0.53	99.29	66.601 TB	10.55	31.16
512 MB - 1 GB	191070	0.55	99.84	127.436 TB	20.18	51.34
1 GB - 2 GB	37615	0.11	99.95	48.821 TB	7.73	59.07
2 GB - 4 GB	7603	0.02	99.97	20.971 TB	3.32	62.39
4 GB - 8 GB	3998	0.01	99.98	22.597 TB	3.58	65.97
8 GB - 16 GB	2733	0.01	99.99	27.669 TB	4.38	70.36
16 GB - 32 GB	2145	0.01	99.99	47.007 TB	7.44	77.80
32 GB - 64 GB	976	0.00	100.00	43.429 TB	6.88	84.68
64 GB - 128 GB	857	0.00	100.00	74.594 TB	11.81	96.49
128 GB - 256 GB	108	0.00	100.00	17.092 TB	2.71	99.20
256 GB - 512 GB	10	0.00	100.00	3.803 TB	0.60	99.80
512 GB - 1 TB	2	0.00	100.00	1.273 TB	0.20	100.00
Totals:	34947640 Files			631.492 TB		



File Size Histogram

- The diagram shows number of files and amount of data over the logarithm of the file size.
- All data right of the red line at 64 MB can be wiped from disk kept only on tape
  - This are only 2.86 % of the files, but 89.8 % of the total data volume !



Use of OSDs

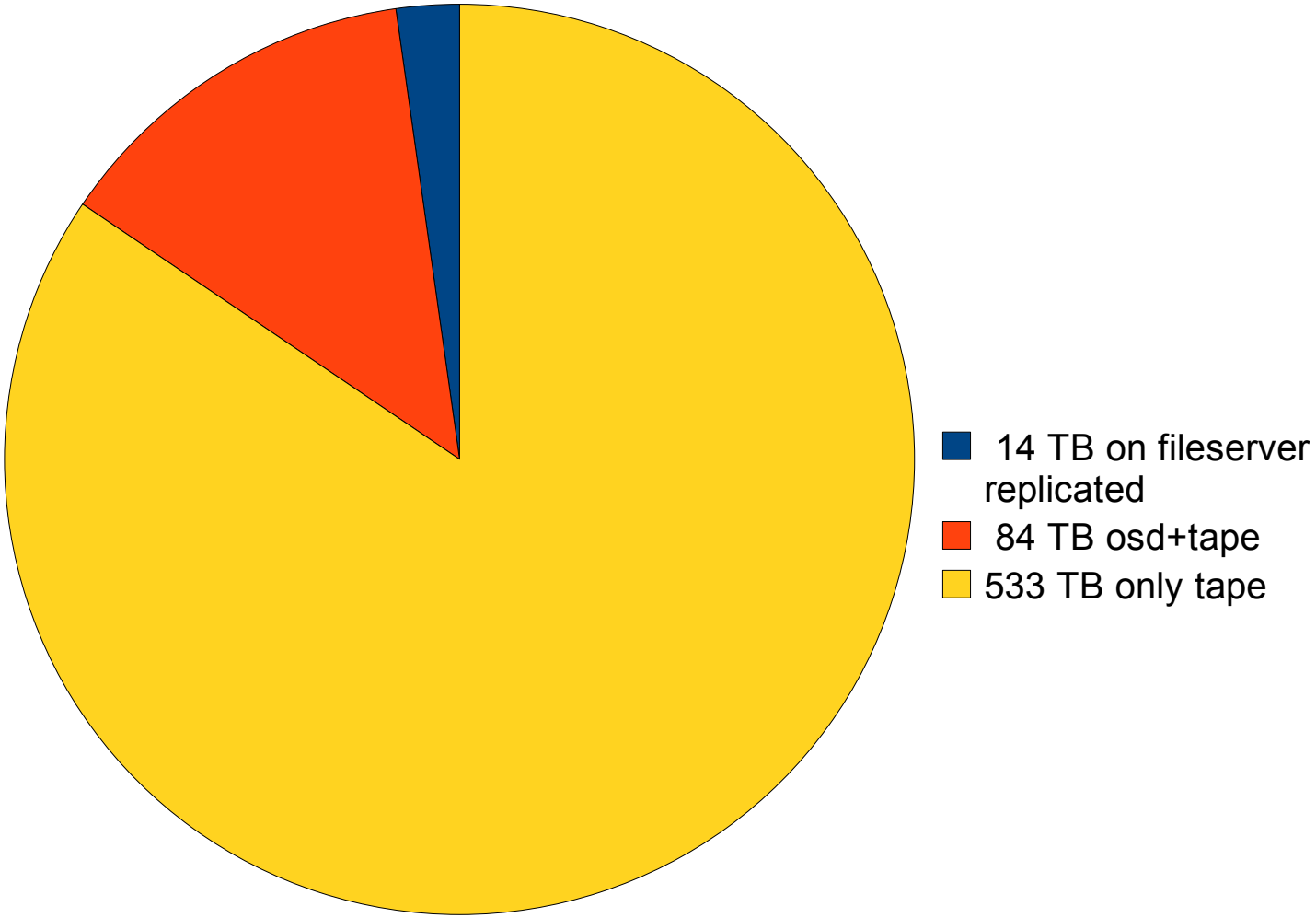
```
 Storage usage:
-----------------------------------------------------------------
                 1 local_disk   27427346 files      14.106 TB
  arch. Osd      3 hpss           264048 objects     2.219 TB
  arch. Osd      4 raid6         1721242 objects     5.461 TB
  arch. Osd      5 tape          7395474 objects   598.687 TB
        Osd      8 afs16-a        177998 objects     3.290 TB
        Osd      9 mpp-fs9-a      124203 objects     4.568 TB
        Osd     10 afs2-a              1 objects     4.760 MB
        Osd     11 w7as            98061 objects     1.922 TB
  arch. Osd     13 hsmgpfs       3764543 objects   581.278 TB
        Osd     23 mpp-fs11-gj    145967 objects     4.564 TB
        Osd     24 mpp-fs12-a     173188 objects     5.035 TB
        Osd     25 mpp-fs13-a     171805 objects     5.025 TB
        Osd     32 afs8-z              3 objects     2.799 GB
        Osd     34 afs17-gb       156619 objects     4.559 TB
        Osd     35 afs18-ga        88306 objects     4.347 TB
        Osd     36 afs21-ge       240487 objects     4.594 TB
        Osd     37 afs22-gf       224982 objects     4.302 TB
        Osd     38 afs19-gc       247277 objects     4.449 TB
        Osd     39 afs20-gd       117963 objects     1.976 TB
        Osd     40 afs23-gg       295137 objects     4.515 TB
        Osd     41 mpp-fs15-gi    152564 objects     4.557 TB
        Osd     42 mpp-fs14-gh    198844 objects     4.548 TB
        Osd     43 mpp-fs10-gk    150452 objects     4.572 TB
        Osd     44 sxbl19-z       793554 objects     5.587 TB
        Osd     47 afs26-a        144144 objects     4.118 TB
        Osd     49 afs1-b         333797 objects     2.243 TB
        Osd     50 afs28-z         53871 objects     4.183 TB
        Osd     51 lethe-z          3420 objects   475.879 GB
-----------------------------------------------------------------
Total                          44665296 objects     1.254 PB
```

„vos traverse“ tells you also where your data are

- 14 TB on filservers
- 83.1 TB on disk OSDs
- 1.16 PB on arch. OSDs (so much because we create two copies)

Where the data are

- All data in the local partitions of the fileservers are replicated to other fileservers
- All data in disk OSDs have copies in archival OSDs (tape)

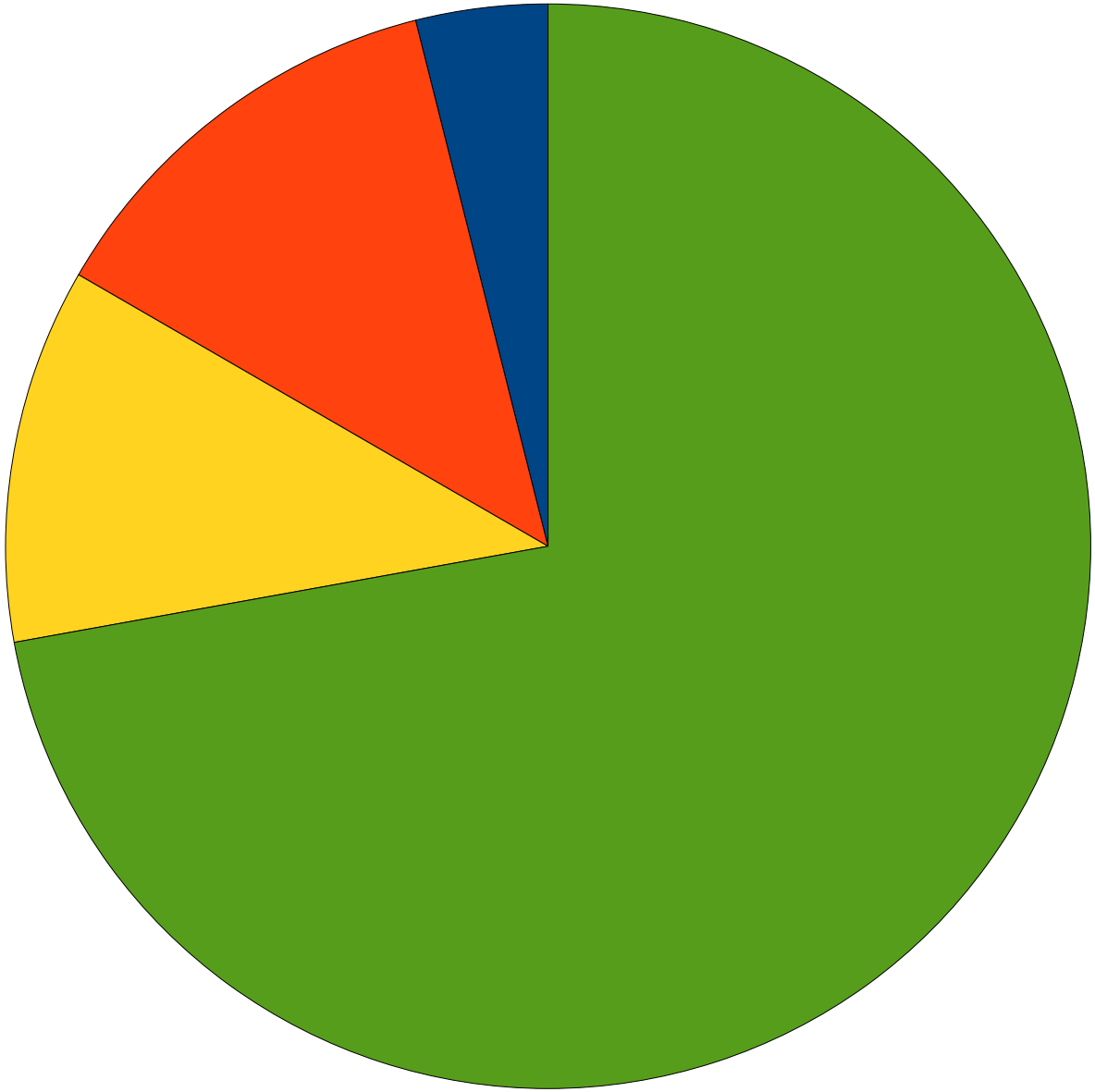


Data without a copy

```
Data without a copy:
-----------------------------------------------------------------
if !replicated: 1 local_disk   27427346 files      14.106 TB
  arch. Osd     5 tape             8388 objects    17.603 TB
        Osd     8 afs16-a           172 objects   164.701 MB
        Osd     9 mpp-fs9-a         154 objects    35.971 MB
  arch. Osd    13 hsmgpfs          2764 objects    18.174 TB
        Osd    23 mpp-fs11-gj       148 objects     4.287 GB
        Osd    24 mpp-fs12-a        148 objects    32.064 MB
        Osd    25 mpp-fs13-a        164 objects    38.566 MB
        Osd    32 afs8-z              2 objects     2.797 GB
        Osd    34 afs17-gb           28 objects     1.071 GB
        Osd    35 afs18-ga           29 objects   227.195 MB
        Osd    36 afs21-ge          213 objects   852.470 MB
        Osd    37 afs22-gf          221 objects    97.515 MB
        Osd    38 afs19-gc          208 objects   791.747 MB
        Osd    39 afs20-gd          159 objects    48.040 MB
        Osd    40 afs23-gg          178 objects   133.398 MB
        Osd    41 mpp-fs15-gi       157 objects    36.947 MB
        Osd    42 mpp-fs14-gh       193 objects    55.369 MB
        Osd    43 mpp-fs10-gk       149 objects     4.415 GB
        Osd    44 sxbl19-z         4830 objects   158.395 GB
        Osd    47 afs26-a           157 objects   504.347 MB
        Osd    49 afs1-b              8 objects   346.546 MB
        Osd    50 afs28-z            58 objects   707.855 MB
        Osd    51 lethe-z          1412 objects    58.805 GB
-----------------------------------------------------------------
Total                          27447286 objects    50.113 TB
```

- 14 TB on filserverers are replicated and therefore not really vulnerable
- 36 TB on OSD 5 and 13 have still to be copied to another archival OSD, but they have already 2 tape copies
- 234 GB of new data have yet not got their copies on archival OSDs.
- Really vulnerable are only the 234 GB, less than 0.04 % of the total data!

What is stored in our AFS cell?



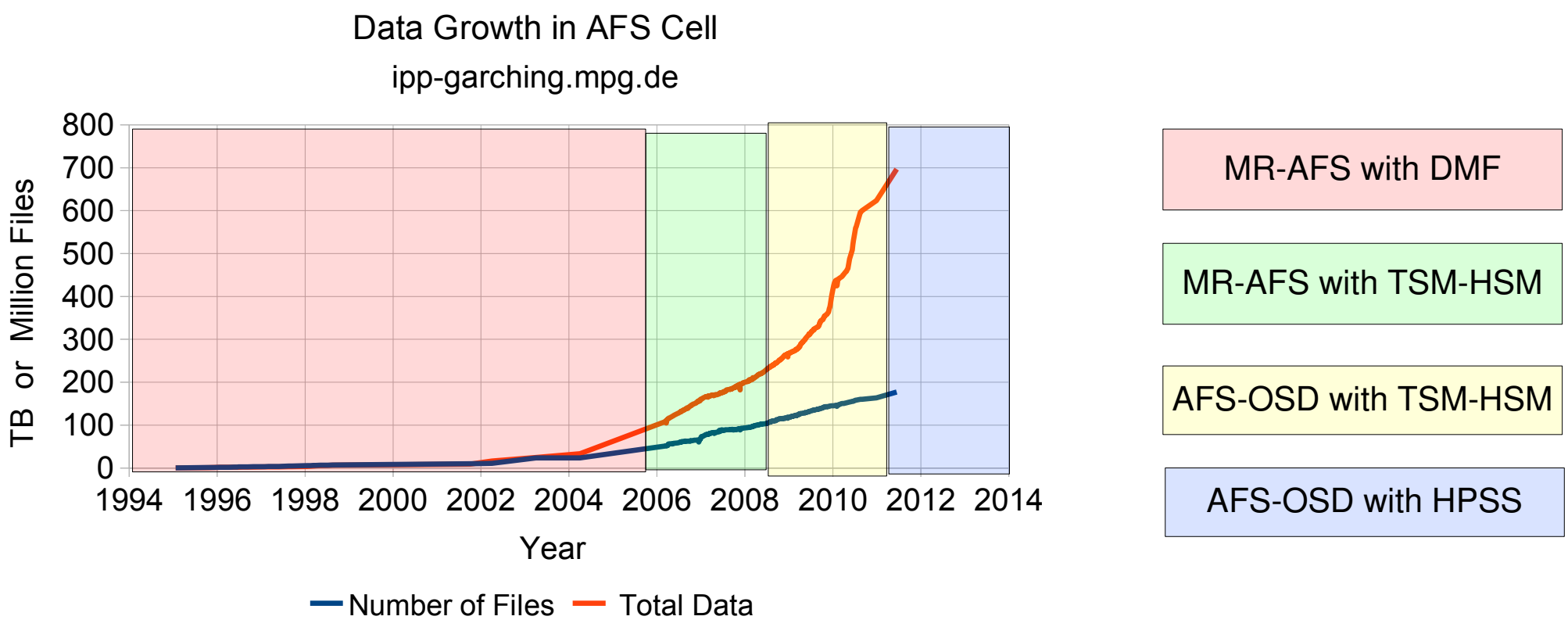
In 2009:

- home directories 13.7 TB
- small file exp. data (AUG, MPE) 44.4 TB
- m-tree 39.0 TB
- experiment data and other long time archives 251.7 TB

Questions or comments?

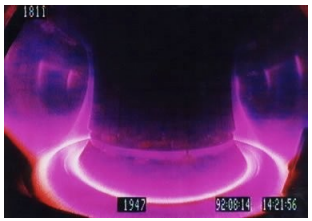
Thank you

Our Data Growth



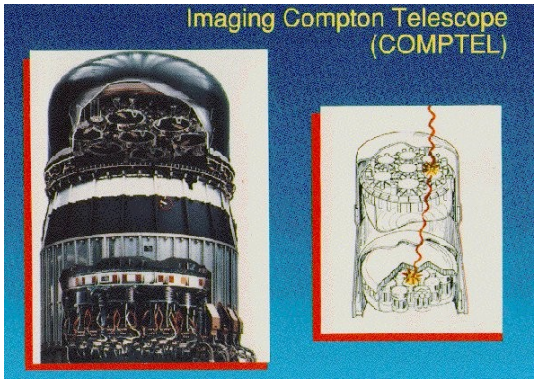
- Transparent to the user and the AFS-tree we had different HSM systems and AFS versions
- Data growth brought us this year to the limit of what TSM-HSM could ingest
- HPSS claimes to scale much better because you can add more data movers

Long time preservation for experiments at IPP



- Experiment Asdex at IPP (~1980-1990)
  - about 33500 shot files smaller 5 MB, totally ~83 GB
    - copied from tapes into AFS at the end of the 90-ies
- Experiment W7AS at IPP (1988-2002)
  - about 60000 shot files 1.5 - 90 MB, totally ~2.2 TB
    - 1988-1995 created under VM/CMS on IBM mainframes and stored in HADES then transfered into AFS
    - 1995-2002 created under VMS or UNIX(AIX) and stored in AFS
- Experiment Asdex Upgrade am IPP (since 1993)
  - ca. 20000 shots each wich many diagnostic files, totally ~92 TB
    - 1993-1995 stored in AMOS2 on IBM mainframe, then transfered into AFS
    - since 1995 in AFS

Gamma-Ray-Astronomy MPE



- COMPTEL (1991-2000)
  - nearly 2 TB in container files, each one ~ 100 MB
    - 1991-1995 stored in HADES, then transfered into AFS
    - 1995-2000 stored in AFS
- EGRET Energetic Gamma Ray Experiment (1991-1996)
  ~ 56 GB archived in AFS
- INTEGRAL (2001-2010 or longer)
  - up to now ~11 TB in small files in AFS, all on disks

Magic Projekt (MPP u.a.)

- Gamma-ray-telescope on the Canaric Ilands
  - since 2003 ~ 106 TB in AFS

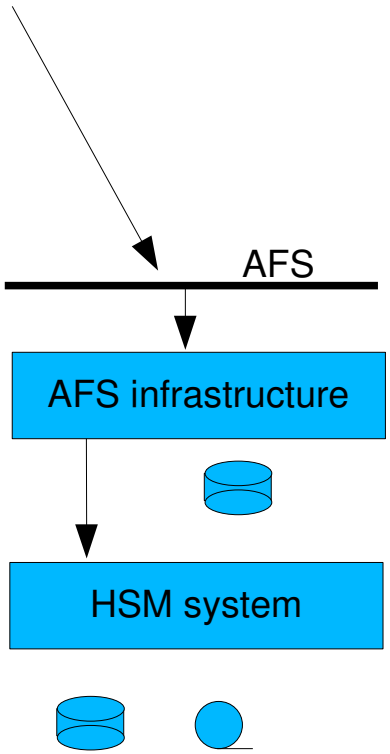
Cultural Heritage

- Video and audio documents from „MPI für Psycholinguistik“ in Nijmegen
  - since 2004 ~ 9.5 TB
    - copied in Nijmegen directly into AFS

- Phototek of „Biblioteca Hertziana“ in Rome
  - since 2004 ~ 3.7 TB
    - in the beginning USB disks sent by mail to RZG and here copied into AFS
    - now copied in Rome directly into AFS

- Phototek of „Kunsthistorisches Institut“ in Florence
  - since 2004 ~ 5 TB
    - in the beginning USB disks sent by mail to RZG and here copied into AFS
    - Now copied into AFS by „rsync“ from GWDG in Göttingen

Why do we use AFS for long time preservation?



- Available for nearly all platforms (Linux, Unix, Windows, MacOs)
- World-wide access
- Security by Kerberos authentication
- Granular granting of access rights by use of ACLs
- Full access to the source code
  - OpenAFS is open source
  - OpenAFS + Object Storage developed at RZG open source as well
- Scales perfectly:
  - Servers can be added at any time
  - Redistribution of data without service interruption
- AFS + Object Storage offers HSM features
  - Any underlying HSM system can be used
  - Transparent acccess to off-line data
- HSM system can be replaced invisibly to the users
  - Copying all data from one HSM system to another can take very long (months to years)

New (sub)commands for general users

- „vos examine“ shows for volumes which may use OSDs typically “osd policy 1“
- There are two new or extended „fs“ subcommands:
  - „fs ls“ gives an output similar to „ls -l“
  - „fs whereis“ shows also OSDs
- The new command „osd“ has also some subcommands for general users
  - „osd list“ shows a list of all OSDs known to the OSDDB database
  - „osd fetchqueue“ shows the fetch queue of files to be brought back on-line on all archival OSDs
  - „osd policies“ gives a list of all known policies in th OSDDB database
- The new command „afsio“ allows writing and reading of files bypassing the cache manager
  - data piped into „afsio write“ are stored in the specified AFS file
  - data piped into “afsio append” are appended at the specified AFS file
  - „afsio read“ reads an AFS file and writes the data to stdout

vos examine

- „vos examine“ shows for volumes which may use OSDs a field “osd policy“

```
~> vos ex muser.hwr
muser.hwr                          536879945 RW  276677045 K  On-line
    afs16.rzg.mpg.de /vicepy                          874 files
    RWrite  536879945 ROnly          0 Backup          0
    MaxQuota 1000000000 K, osd policy    1 1000 files
    Creation    Wed Sep 26 19:51:25 2001
    Copy        Thu Jan 29 16:39:29 2009
    Backup      Never
    Last Update Tue Oct 13 13:16:22 2009
    0 accesses in the past day (i.e., vnode references)

    RWrite: 536879945     ROnly: 536879946
    number of sites -> 3
       server afs16.rzg.mpg.de partition /vicepy RW Site
       server afs16.rzg.mpg.de partition /vicepy RO Site
       server afs4.bc.rzg.mpg.de partition /vicepy RO Site
~>
```

„fs ls“

- „fs ls“ gives an output similar to „ls -l“

```
~> fs ls
f rw-     hwr      550615 1998-07-01 13:04:21 arla-0.7.2.tar.gz
l rwx     hwr          11 2009-09-18 19:57:51 servers -> tmp/servers
m rwx     root       2048 private
o rwx     hwr    63128640 1999-03-17 12:23:03 unicosmk.cray-t3e.20443
d rwx     hwr        2048 1995-07-26 09:47:14 tmp
w rw-  daemon    79829905 1997-01-08 09:22:50 ymp_uni80.mrafs34a.wrk
~>
```

The character in column 1 tells you what it is:

d == directory

f == normal AFS file in fileserver partition

l == symbolic link

m == mount point

o == object file (on-line)

w == wiped object file (off-line)

„fs whereis“

- „fs whereis“ has been extended to show also the OSDs where the data are stored:

```
> fs ls ymp_uni80.mrafs34a.wrk
w rw-  daemon    79829905 1997-01-08 09:22:50 ymp_uni80.mrafs34a.wrk
> fs whereis ymp_uni80.mrafs34a.wrk
File ymp_uni80.mrafs34a.wrk is on host afs16.rzg.mpg.de  Osds: tape hsmgpfs
>
```

osd list

- „osd list“ shows all OSDs. („1 local_disk“ means the fileserver partition, not a real OSD)

```
~>  osd list
id name(loc)      ---total space---      flag  prior. own. server lun size range
 1 local_disk                                   wr  rd                 (0kb-1mb)
 4 raid6            5119 gb  69.0 % up   arch  64  64     afs15.rz  0 (0kb-8mb)
 5 tape             8924 gb  24.5 % up   arch  64  40     styx.rzg  0 (1mb-500gb)
 8 afs16-a          4095 gb  81.1 % up   hsm   80  80     afs16.rz  0 (1mb-100gb)
 9 mpp-fs9-a       11079 gb  84.9 % up   hsm   80  80 mpp mpp-fs9.  0 (1mb-500gb)
10 afs4-a           4095 gb  77.1 % up   hsm   80  80     afs4.bc.  0 (1mb-100gb)
11 w7as(hgw)        2721 gb  67.1 % up         80  80     afs-w7as  0 (1mb-100gb)
12 afs1-a           1869 gb  92.6 % up         70  80 tok afs1.rzg  0 (1mb-100gb)
13 hsmgpfs         31236 gb  14.5 % up   arch  64  30     hsmi.rzg 12 (8mb-500gb)
14 afs6-a           1228 gb  84.9 % up   hsm   70  80 tok afs6.rzg  0 (8mb-100gb)
23 mpp-fs11-gj      5580 gb  70.7 % up   hsm   80  80 mpp mpp-fs11 191 (1mb-100gb)
24 mpp-fs12-a       6143 gb  78.1 % up   hsm   80  80 mpp mpp-fs12  0 (1mb-100gb)
25 mpp-fs13-a       6143 gb  84.0 % up   hsm   80  80 mpp mpp-fs13  0 (1mb-100gb)
32 afs8-z           1023 gb   1.3 % up   hsm   12  80     afs8.rzg 25 (1mb-100gb)
34 afs17-gb         5585 gb  85.0 % up   hsm   80  80     afs17.rz 183 (1mb-500gb)
35 afs18-ga         5585 gb  85.0 % up   hsm   80  80     afs18.rz 182 (1mb-500gb)
36 afs21-ge         5585 gb  85.0 % up   hsm   80  80     afs21.rz 186 (1mb-500gb)
37 afs22-gf         5585 gb  84.9 % up   hsm   80  80     afs22.rz 187 (1mb-500gb)
38 afs19-gc         5585 gb  83.7 % up   hsm   89  80     afs19.rz 184 (1mb-500gb)
39 afs20-gd         5585 gb  82.8 % up   hsm   80  80     afs20.rz 185 (1mb-500gb)
40 afs23-gg         5585 gb  74.9 % up   hsm   80  80     afs23.rz 188 (1mb-500gb)
41 mpp-fs15-gi      5580 gb  84.7 % up   hsm   80  80 mpp mpp-fs15 190 (1mb-500gb)
42 mpp-fs14-gh      5580 gb  84.8 % up   hsm   80  80 mpp mpp-fs14 189 (1mb-500gb)
43 mpp-fs10-gk      5580 gb  84.1 % up   hsm   80  80 mpp mpp-fs10 192 (1mb-500gb)
44 sxbl19-z         6656 gb   3.5 % up   hsm   80  80 aug sxbl19.a 25 (1mb-500gb)
...
```

osd list -wipeable

- With -wipeable you get only the wipeabble osds, but with additional information
  - Newest wiped shows you how long unused files stay on-line

```
>  osd list -wipeable
Iid name(loc)        size     state    own    usage    limit   wipe >    newest wiped
 8 afs16-a          4095 gb up             81.1 %   85.0 %   64 mb    Jul 31 2008
 9 mpp-fs9-a       11079 gb up       mpp   84.9 %   85.0 %   64 mb    Oct 21 2008
10 afs4-a           4095 gb up             77.1 %   85.0 %   64 mb    Aug 14 2008
14 afs6-a           1228 gb up       tok   84.9 %   85.0 %   64 mb    May 19 2008
23 mpp-fs11-gj      5580 gb up       mpp   70.7 %   85.0 %   64 mb
24 mpp-fs12-a       6143 gb up       mpp   78.1 %   85.0 %   64 mb
25 mpp-fs13-a       6143 gb up       mpp   84.0 %   85.0 %   64 mb
32 afs8-z           1023 gb up              1.3 %   80.0 %   64 mb    Jul 31 2008
34 afs17-gb         5585 gb up             85.0 %   85.0 %   64 mb    Jun 30 2009
35 afs18-ga         5585 gb up             85.0 %   85.0 %   64 mb    Apr 24 2009
36 afs21-ge         5585 gb up             85.0 %   85.0 %   64 mb    May 31 2009
37 afs22-gf         5585 gb up             84.9 %   85.0 %   64 mb    Jun 14 2009
38 afs19-gc         5585 gb up             83.7 %   85.0 %   64 mb    Jun 25 2009
39 afs20-gd         5585 gb up             82.8 %   85.0 %   64 mb    Aug  2 2009
40 afs23-gg         5585 gb up             74.9 %   85.0 %   64 mb    Jun 19 2009
41 mpp-fs15-gi      5580 gb up       mpp   84.7 %   85.0 %   64 mb    Feb 25 2009
42 mpp-fs14-gh      5580 gb up       mpp   84.8 %   85.0 %   64 mb    Feb 25 2009
43 mpp-fs10-gk      5580 gb up       mpp   84.1 %   85.0 %   64 mb    Jun 21 2009
44 sxbl19-z         6656 gb up       aug    3.5 %   85.0 %   64 mb
...
```

osd policies

- „osd policies“ shows all policies defined in the database.

```
~> osd policies
     2 local_disk
        true => location=local, continue;
     3 root
        ~'*.root' => location=osd, stop;
     4 striped
        >8M => location=osd, stripes=2, stripe_size=12, continue;
~>
```

- Additionally exists policy number 1. It is hard coded and means files > max size for local_disk should go into OSDs

afsio

- 'afsio' is used to write or read files bypassing the cache manager. 'afsio' reads from stdin (for write) and writes to stdout (for read)
  - write — write date into an empty or new AFS file

```
~> afsio help write
afsio write: write a file into AFS
Usage: afsio write [-file <AFS-filename>] [-cell <cellname>] [-verbose] [-md5] [-help]
Where: -md5  calculate md5 checksum
~>
```

  -
  - append — append data at the end of an existing AFS file
  - read — read an AFS file
  - help — help information

- On some platforms afsio is much faster than I/O through the cache manager

```
/tmp> ls -l 1gb
-rw-------  1 hwr rzs 1073741824 2009-10-14 12:55 1gb
/tmp: cat 1gb | afsio write /afs/ipp/m/hwr/1gb -verbose -md5
801 MB transferred, present date rate = 26 MB/sec.
Transfer of 1073741824 bytes took 37.774 sec.
Total data rate = 27 MB/sec. for write
a6cdcbd4622366b4ba41618c3717a2fe  1gb
user=19.108 system=15.043 real=0:39.90
/tmp>
```

With -verbose it shows every 30 seconds the current data rate. -md5 slows down, of course.

Appendix
Commands for Administrators

- There are some new or extended „fs“ subcommands:
  - „fs [fid]vnode“ shows all vnode fields. For OSD files also the index in the volume's OSD metadata file
  - „fs [fid]osd“ shows the OSD metadata
- The command „osd“ has some subcommands to show what is stored in an OSD
  - „osd volumes“ shows the RW-voiume ids present in the OSD
  - „osd objects“ shows all objects in the OSD belonging to a specified volume
  - “osd examine” shows details about a single object
- The command “vos” got some new subcommands
  - “vos traverse” shows file size statistic and the number of objects per OSD
  - “vos listobjects” shows object-ids of all objects on a specified OSD
  - “vos salvage” does a health check for a volume checking sizes and link counts

„fs vnode“

- „fs vnode“ shows all vnode fields

```
>  fs vnode ymp_uni80.mrafs34a.wrk
File  536879945.290.46282
        modeBits          = 0644
        linkCount         = 1
        author            = 2
        owner             = 2
        group             = 4132
        Length            = 79829905      (0x0, 0x4c21b91)  76.130 MB
        dataVersion       = 3
        unixModifyTime    = 1997-01-08 09:22:50
        serverModifyTime = 2009-05-10 17:13:48
        vn_ino_lo         = 0    (0x0)
        lastUsageTime     = 2009-02-25 10:46:39
        osd file on disk = 0
        osdMetadataIndex = 75
        parent            = 1
>
```

  - Length shows also in hex notation vn_length_hi and length.
  - This file is in object storage and doesn't have therefore an inode number
  - vn_ino_hi is filled with the uniquifier in namei-fileservers. LastUsageTime is stored at this location instead.

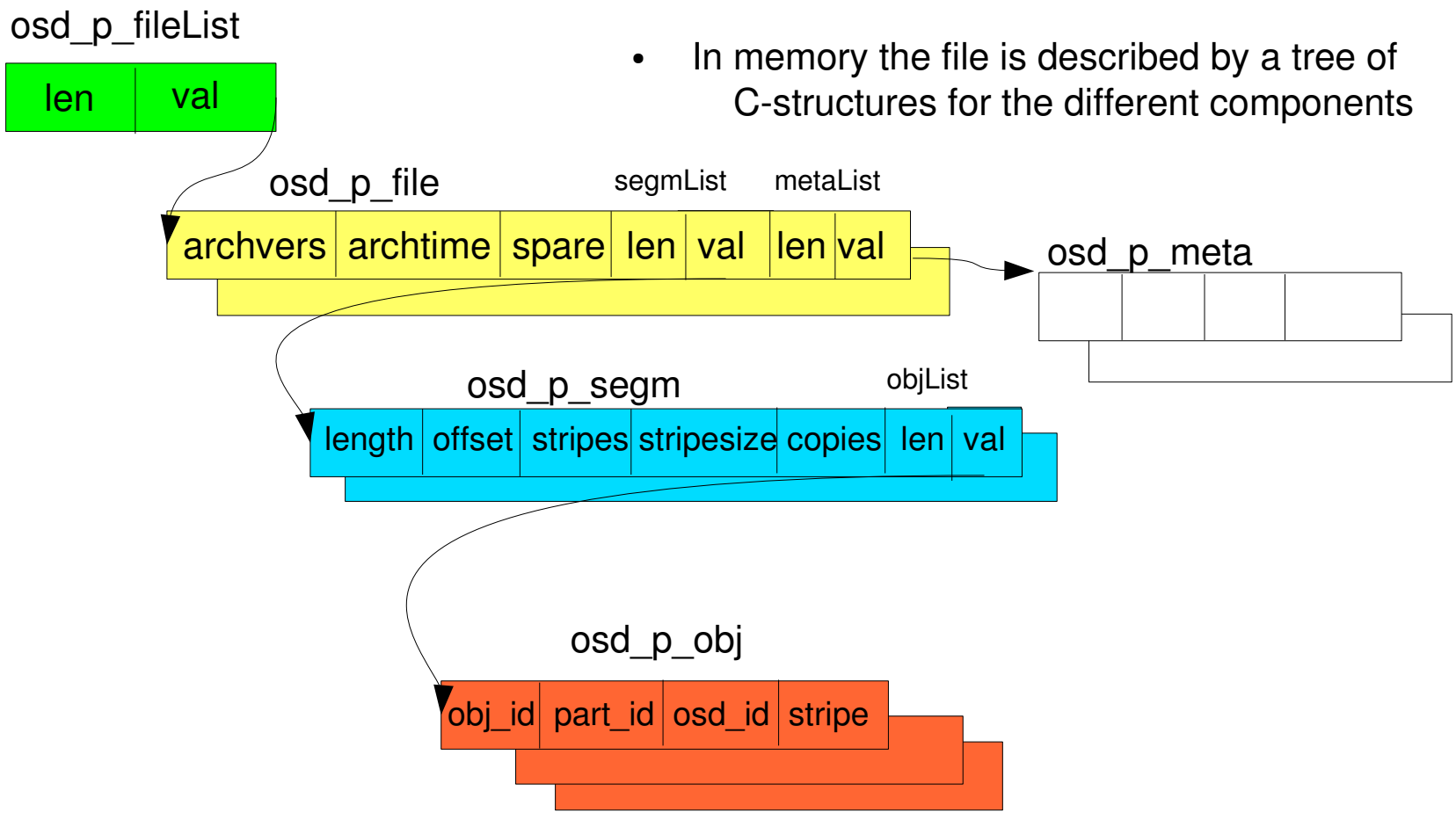
„fs osd“

- For an OSD file you can see the OSD metadata with „fs osd“:

```
> fs osd ymp_uni80.mrafs34a.wrk
ymp_uni80.mrafs34a.wrk has 312 bytes of osd metadata, v=3
Archive, dv=3, 2007-07-19 17:39:30, 1 fetches, last: 2009-02-25, 1 segm, flags=0x2
    segment:
        lng=79829905, offs=0, stripes=1, strsize=0, cop=1, 1 objects
        object:
            obj=536879945.290.46282.0, osd=5, stripe=0
    metadata:
        md5=6441333f2acdae8833898bebaf2041d2  as from  2007-07-19 17:39:30
Archive, dv=3, 2009-02-25 10:46:39, 1 segm, flags=0x2
    segment:
        lng=79829905, offs=0, stripes=1, strsize=0, cop=1, 1 objects
        object:
            obj=536879945.290.46282.0, osd=13, stripe=0
    metadata:
        md5=6441333f2acdae8833898bebaf2041d2  as from  2009-02-25 11:52:02
>
```

- This file is not on-line, only in two archival OSDs (5 and 13)
  - The file had been restored once from the 1st archival copy on Feb. 25
  - Both archive are form data version 3 and have therefore the same md5 sum
  - flag=0x2 means it has been checked that the file was copied to tape

osd metadata in memory



- In memory the file is described by a tree of C-structures for the different components

- These structures are serialized in net-byte-order by means of rxgen-created xdr-routines into slots of the volume special file “osdmetadata”.

„fs osd“

- For an OSD file you can see the OSD metadata with „fs osd“:

```
> fs osd ymp_uni80.mrafs34a.wrk
ymp_uni80.mrafs34a.wrk has 312 bytes of osd metadata, v=3
Archive, dv=3, 2007-07-19 17:39:30, 1 fetches, last: 2009-02-25, 1 segm, flags=0x02
    segment:
        lng=79829905, offs=0, stripes=1, strsize=0, cop=1, 1 objects
        object:
            obj=536879945.290.46282.0, osd=5, stripe=0
    metadata:
        md5=6441333f2acdae8833898bebaf2041d2  as from  2007-07-19 17:39:30
Archive, dv=3, 2009-02-25 10:46:39, 1 segm, flags=0x02
    segment:
        lng=79829905, offs=0, stripes=1, strsize=0, cop=1, 1 objects
        object:
            obj=536879945.290.46282.0, osd=13, stripe=0
    metadata:
        md5=6441333f2acdae8833898bebaf2041d2  as from  2009-02-25 11:52:02
>
```

- This file is not on-line, only in two archival OSDs (5 and 13)
  - The file had been restored once from the 1st archival copy on Feb. 25
  - Both archive are form data version 3 and have therefore the same md5 sum
  - flag=0x2 means it has been checked that the file was copied to tape

Salvaging OSD volumes

- “bos salvage” checks only consistency of the OSD metadata.
- To check also the data run „vos salvage <volume>“ (without options it just checks)

```
> vos salvage 1108472992.115754.200635
Salvaging volume 1108472992
Object 1108472992.115754.200635.0 has wrong length on 34 (449740800 instead of 512037786)
Object 1108472992.115828.200709.0: linkcount wrong on 5 (1 instead of 3)
Object 1108472992.115830.200711.0: linkcount wrong on 5 (1 instead of 3)
1108472992: 49697 local (10.243 gb) and 8985 in OSDs (2.493 tb), 3 errors ATTENTION
```

-
- In this example 1 object is too short. The low link count of 2 other objects is not necessarily an error.
  - Probably these are archival copies which have been created after the last replication. Before running „vos salvage -update“ we did a new „vos release“

```
> vos salvage 1108472992.115754.200635 -update
Salvaging volume 1108472992
Object 1108472992.115754.200635.0 has wrong length on 34 (449740800 instead of 512037786),
repaired
1108472992: 49697 local (10.243 gb) and 8985 in OSDs (2.493 tb), 1 errors ATTENTION
```

Damaged Partition

- Broken RAIDs happen about twice a year in our cell !
- If it's a fileserver partitions:
  - If you consequently released RO-volumes to other servers
    - You may run „vos convertROtoRW“ on the RO-volumes
    - Don't forget to create new RO-volumes after that !
  - Otherwise you will have to restore dumps (takes much longer)
- If it's an OSD partitions
  - run „vos listobjects“ for the lost OSD on all fileservers to get the object-ids
  - If it's a non-archival OSD
    - run „fs fidwipe“ for all these object-ids (the tag-suffix will be ignored)
      - will fail for newly created files without archival copy.
    - run „fs fidprefetch“ to bring the wiped files on-line again
  - If it's an archival OSD
    - run „fs fidreplace <obj-id> <osd-number> -1“ to eliminate the archival copy

AFS hanging or slow 1

If AFS seems to be slow or if files or directories seem to be blocked it makes sense to analyze what's going on on your server and client machines.

- Ask hanging clients with „cmdebug“ or „cmdebug -long“ to find which file is requested
- Use „rxdebug <client> 7001 -nodally“ to see active RPCs
- Use „fs threads -server <fileserver>“ to see load on a fileserver

```
~: fs threads -server afs16
3 active threads found
rpc FsCmd on 1108595910.590 from 130.183.9.5
rpc StoreData64 on 1108524165.480 from 134.107.107.11
rpc FsCmd on 0.15 from 130.183.2.114
~:
```

  - The last line is my „fs thread“ command itself (as I know from the IP address)
  - The other FsCmd line is a “fs fidarchive“ running on the database server
- Use „o

```
~: osd threads 13
rpc create_archive on 1108595910.590.8415.0 from 130.183.30.16
rpc threads on 0.0.0.0 from 130.183.2.114
~:
```

AFS hanging or slow 2

I don't have saved the output and steps of the analysis of such a hanger, but here is what happened

- There was a problem with GPFS and TSM-HSM on an archival OSD which let hang a write() to GPFS forever
- The write() belonged to an RXOSD_create_archive RPC for which a „fs fidarchive“ was waiting.
- „fs fidarchive“ has a WRITE_LOCK on the file's vnode because it must update the osdmetadata when the archiving is successful.
- This blocked many RXAFS_GetStatus and/or RXAFS_InlineBulkStatus RPCs
- Finally all threads of the server were blocked and the server didn't respond anymore
- A server restart only helped for a short time because the archiver script immediately started the next „fs fidarchive“ to that fileserver and the same thing happened again...

Server Statistcs

```
> fs stat afs14
Since  2009-09-17 05:08:22 (211842 seconds == 2 days, 10:50:42 hours)
Total number of bytes received     211137205106  196 gb
Total number of bytes sent          55456907335   51 gb
rpc 65538 StoreData64               8707980
rpc   132 FetchStatus               1563205
rpc   147 GiveUpCallBacks            612517
rpc   135 StoreStatus               1906959
rpc   137 CreateFile                 314794
rpc 65537 FetchData64               2063199
rpc   157 ExtendLock                  89790
rpc   140 Link                        11107
rpc   136 RemoveFile                  52155
rpc   156 SetLock                    103698
rpc   158 ReleaseLock                103694
rpc 65560 OsdPolicy                    7713
rpc   138 Rename                     277955
rpc 65536 InlineBulkStatus           673076
rpc 65542 GetStatistics64              8254
rpc   146 GetStatistics                8254
rpc   130 FetchData                   41994
rpc   133 StoreData                   41586
rpc   139 Symlink                      1697
rpc   141 MakeDir                      8973
rpc   134 StoreACL                     6528
rpc   155 BulkStatus                  31685
rpc 65539 GiveUpAllCallBacks             35
rpc   142 RemoveDir                     394
rpc 65566 Statistic                       9
```

- „fs statistic <server> [-verbose]“ gives a statistic about data traffic and RPCs with “-verbose” you get the transfer rates per 15 minute interval
- „osd statistic <OSD> [-verbose]“ gives the same for OSDs
- „vos statistc <server> [-verbose]“ gives the same for the volserver, but without RPC statistc.

These commands allow you to see hot spots and give an idea about the data traffic in your cell.

Server Statistcs

```
> vos statistic afs11 -v
/----- snip ------/
17:45-18:00     0 KB/s sent     0 KB/s received
18:00-18:15     0 KB/s sent  3133 KB/s received
18:15-18:30     0 KB/s sent  4756 KB/s received
18:30-18:45     0 KB/s sent  2874 KB/s received
18:45-19:00     0 KB/s sent  1511 KB/s received
19:00-19:15     0 KB/s sent  2173 KB/s received
19:15-19:30     0 KB/s sent  2176 KB/s received
19:30-19:45     0 KB/s sent   809 KB/s received
19:45-20:00     0 KB/s sent  1317 KB/s received
20:00-20:15     0 KB/s sent   371 KB/s received
20:15-20:30     0 KB/s sent  5660 KB/s received
20:30-20:45     0 KB/s sent   671 KB/s received
20:45-21:00     0 KB/s sent  4098 KB/s received
21:00-21:15     0 KB/s sent  8706 KB/s received
21:15-21:30     0 KB/s sent 10452 KB/s received
21:30-21:45     0 KB/s sent  3892 KB/s received
21:45-22:00     0 KB/s sent  5480 KB/s received
22:00-22:15     0 KB/s sent  7339 KB/s received
22:15-22:30     0 KB/s sent  8558 KB/s received
22:30-22:45     0 KB/s sent  5603 KB/s received
22:45-23:00     0 KB/s sent  6961 KB/s received
23:00-23:15     0 KB/s sent  5522 KB/s received
23:15-23:30     0 KB/s sent  2779 KB/s received
23:30-23:45     0 KB/s sent   946 KB/s received
23:45-24:00     0 KB/s sent     0 KB/s received
Since  Sep 17 05:00 (716186 seconds == 8 days, 6:56:26 hours)
Total number of bytes received     348728736471  324 gb
Total number of bytes sent                    0    0 bytes
~:
```

- This example shows volserver traffic during nightly „vos release“.
- This server only keeps RO-copies of volumes belonging to one of the experiments.
- The vos release script is started at 18:00 by CRON.
- All intervals before 18:00 don't show activities.

Command Reference 'osd' 1

- The „osd“ command has the following subcommands which talk to the OSDDB
  - createosd — create new osd entry in the OSDDB
  - setosd — change fields for existing osd entry in OSDDB
  - deleteosd — mark osd entry as obsolete (will not really delete it)
  - addpolicy — create a policy entry in the OSDDB
  - deletepolicy — delete a policy entry in the OSDDB
  - addserver — create server entry in the OSDDB
  - deleteserver — delete server entry in the OSDDB
  - list — list OSDs known in the OSDDB
  - policies — list policies known in the OSDDB
  - servers — list servers known in the OSDDB
  - osd — list all fields of the osd entries in the OSDDB
- Use 'osd help <subcommand>' to get syntax and parameters information.
- All modifying commands can only used by administrators (in UserList of the server)

Command Reference 'osd' 2

- The „osd“ has the following subcommands to analyze or manage data in an OSD
  - volumes — show IDs all RW-volumes having data in the OSD
  - objects — show object IDs (Fids) of all objects of a volume
  - examine — show details of an object
  - incrlinkcount — increment link count of an object
  - decrlinkcount — decrement link count of an object
  - read — read contents of an onbject
  - write — over-write contents of an object
  - md5sum — let OSD calculate md5sum of an object
- Use 'osd help <subcommand>' to get syntax and parameters information.
- All these commands can be used only by administrators (in UserList of the server)

Command Reference 'osd' 3

- Other subcommands of 'osd';
  - fetchqueue — show fetch requests on archival HSM OSDs
  - statistic — show RPC statistic and data flow
  - threads — show active RPCs
  - getvariable — show value of a variable
  - setvariable — set new value to a variable (e.g. LogLevel)
  - wipecandidate — get sorted list of longest unused obejcts
  - help — get help texts
- Use 'osd help <subcommand>' to get syntax and parameters information.
- Some of these commands can be used only by administrators (in UserList of the server)

Command Reference 'vos'

- New subcommands of 'vos';
  - archcand — get sorted list of files which need archival copies
  - statistic — show data flow statistic
  - listobjects — show objects on specified OSD
  - getvariable — show value of a variable
  - setvariable — set new value to a variable (e.g. LogLevel)
  - salvage — check size and linkcounts of all objects in a volume
  - traverse — show file statistic on server or volume
  - split — split a volume at a specified directory vnode
- New options for subcommand 'dump'

  -osd — dump OSD files as normal files (include data)

  -metadataonly — dump only directories and OSD metadata (for 'dumptool')

Command Reference 'fs'

- New display subcommands of 'fs';
  - statistic — show RPC statistic of data flow
  - threads — show active RPCs
  - [fid]vnode — show vnode fields (and relative path)
  - [fid]osd — show OSD metadata of a file
  - ls — shows directory in 'ls -l' style with info about osd files
  - getvariable — show value of a variable (e.g. LogLevel)
  - translate — translate namei-path to fid or vice versa
  - listlocked — shows locked vnodes

Command Reference 'fs' 2

- New 'fs' subcommands acting on files:
  - [fid]prefetch — bring wiped OSD-file back on-line (asynchronously)
  - [fid]archive — create archive copy of OSD file
  - [fid]wipe — wipe on-line copy, keep only archival copies
  - [fid]replaceosd — move object on specified OSD to another one
  - [fid]oldversion — restore older version of OSD-file
  - createstripedfile — preallocate OSD file
- Other new modifying subcommands:
  - setpolicy — set policy on directory level
  - setvariable — set new value to a variable (e.g. LogLevel)

Command Reference 'afsio'

- 'afsio' is used to write or read files bypassing the cache manager. 'afsio' reads from stdin (for write) and writes to stdout (for read)
  - write — write date into an empty or new AFS file

```
~: afsio help write
afsio write: write a file into AFS
Usage: afsio write [-file <AFS-filename>] [-cell <cellname>] [-verbose] [-md5] [-help]
Where: -md5  calculate md5 checksum
```

  - append — append data at the end of an existing AFS file

```
~: afsio help append
afsio append: append to a file in AFS
Usage: afsio append [-file <AFS-filename>] [-cell <cellname>] [-verbose] [-help]
~:
```

  - read — read an AFS file

```
~: afsio help read
afsio read: read a file from AFS
Usage: afsio read -file <AFS-filename> [-cell <cellname>] [-verbose] [-help]
~:
```

  - help — help information
- On some platforms 'afsio' is remarkably faster than I/O through the cache manager

Building openafs-1.4-osd

After checking out the source code from DESY'a subversion server by

svn checkout http://svnsrv.desy.de/public/openafs-osd/trunk/openafs/

Do as usual configure and make.

configure has additional options:

--enable-object-storage to build server and clients with OSD support

--enable-vicep-access to build server and clients with support for embedded filesystems

Building openafs-1.6-osd

After checking out the source code from our subversion server by

svn checkout http://pfanne.rzg.mpg.de/svn/RZG-AFS/trunk/afs_kerberos/openafs-1.6-osd/

Do as usual configure and make.

The client always is built with support for object storage and for Linux 2.6 also for embedded filesystems!

configure has additional options:

--enable-object-storage	to build **server** with OSD support
--enable-vicep-access	to build **server** with support for embedded filesystems
--enable-hpss-hsm	enable use of HPSS as HSM system for object storage
--with-hpss-path=path	where include and lib for HPSS can be found, typically /opt/hpss

Questions or comments?

Thank you