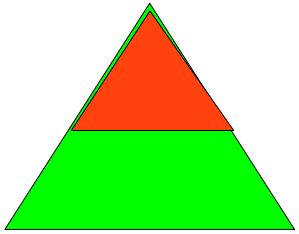
HPSS as HSM solution for AFS-OSD

Hartmut Reuter
reuter@rzg.mpg.de

HSM



HSM means Hierarchical Storage Managent

- Requires a hierarchy of storage systems, a storage pyramide:
  - with the fastest and most expensive storage at the top
  - and the slowest and and most unexpensive storage at the bottom.
- Unused files will be migrated after a while down to less expensive storage
  - Access to a file on slow storage triggers migration back to fast storage.
- In many cases just two different storage layers:
  - Disk (fast, expensive)
  - Tape (slow startup, unexpensive media)
- To run unattended HSM systems mostly requires a tape robot system (expensive!).

Use of HSM with AFS-OSD

OpenAFS-OSD allows you to use your site's HSM system for AFS space:

- Create an archival OSD on a partition under control of the HSM system
- Set some flags and variables for your disk OSDs in the OSDDB
  - „osd setosd <number> -wipeable“
  - „osd setosd <number> -highwatermark <per mille value>“
  - „osd setosd <number> -minwipesize <value>“
- Run the archiver and wiper scripts as bos instances on your database servers

This gives you HSM functionality for OSD-files in AFS. If the OSD partition reaches the highwatermark (typically 85 %) the wiper script removes some longest unused files from the OSD. So you get „unlimited“ disk space in the OSDs.

File Size Histogram

File Size Range	Files	%	run %	Data	%	run %
0 B - 4 KB	81038398	50.56	50.56	100.596 GB	0.02	0.02
4 KB - 8 KB	12936319	8.07	58.63	71.035 GB	0.01	0.03
8 KB - 16 KB	10906831	6.80	65.43	117.180 GB	0.02	0.05
16 KB - 32 KB	11559488	7.21	72.64	243.626 GB	0.04	0.09
32 KB - 64 KB	9806693	6.12	78.76	446.594 GB	0.07	0.16
64 KB - 128 KB	7557635	4.71	83.48	656.244 GB	0.11	0.27
128 KB - 256 KB	5344475	3.33	86.81	939.943 GB	0.15	0.42
256 KB - 512 KB	5566350	3.47	90.28	1.886 TB	0.31	0.73
512 KB - 1 MB	4077746	2.54	92.83	2.610 TB	0.44	1.17
1 MB - 2 MB	2604629	1.62	94.45	3.630 TB	0.61	1.77
2 MB - 4 MB	2502771	1.56	96.01	6.636 TB	1.11	2.88
4 MB - 8 MB	2343596	1.46	97.47	12.567 TB	2.09	4.97
8 MB - 16 MB	1572770	0.98	98.46	17.175 TB	2.86	7.84
16 MB - 32 MB	843902	0.53	98.98	17.271 TB	2.88	10.71
32 MB - 64 MB	603531	0.38	99.36	24.913 TB	4.15	14.86
64 MB - 128 MB	402034	0.25	99.61	33.962 TB	5.66	20.52
128 MB - 256 MB	217768	0.14	99.75	36.847 TB	6.14	26.66
256 MB - 512 MB	192038	0.12	99.87	70.386 TB	11.73	38.39
512 MB - 1 GB	171588	0.11	99.97	113.809 TB	18.96	57.36
1 GB - 2 GB	30779	0.02	99.99	40.698 TB	6.78	64.14
2 GB - 4 GB	5357	0.00	99.99	14.039 TB	2.34	66.48
4 GB - 8 GB	2897	0.00	100.00	17.071 TB	2.84	69.32
8 GB - 16 GB	1592	0.00	100.00	16.383 TB	2.73	72.05
16 GB - 32 GB	1776	0.00	100.00	39.667 TB	6.61	78.66
32 GB - 64 GB	933	0.00	100.00	41.606 TB	6.93	85.59
64 GB - 128 GB	782	0.00	100.00	68.721 TB	11.45	97.04
128 GB - 256 GB	98	0.00	100.00	15.214 TB	2.54	99.58
256 GB - 512 GB	6	0.00	100.00	2.524 TB	0.42	100.00
Totals:	**160292782 Files**			**600.150 TB**		

Filesize Histogram over
All 160 million files in
Our cell

50 % of the files <= 4K

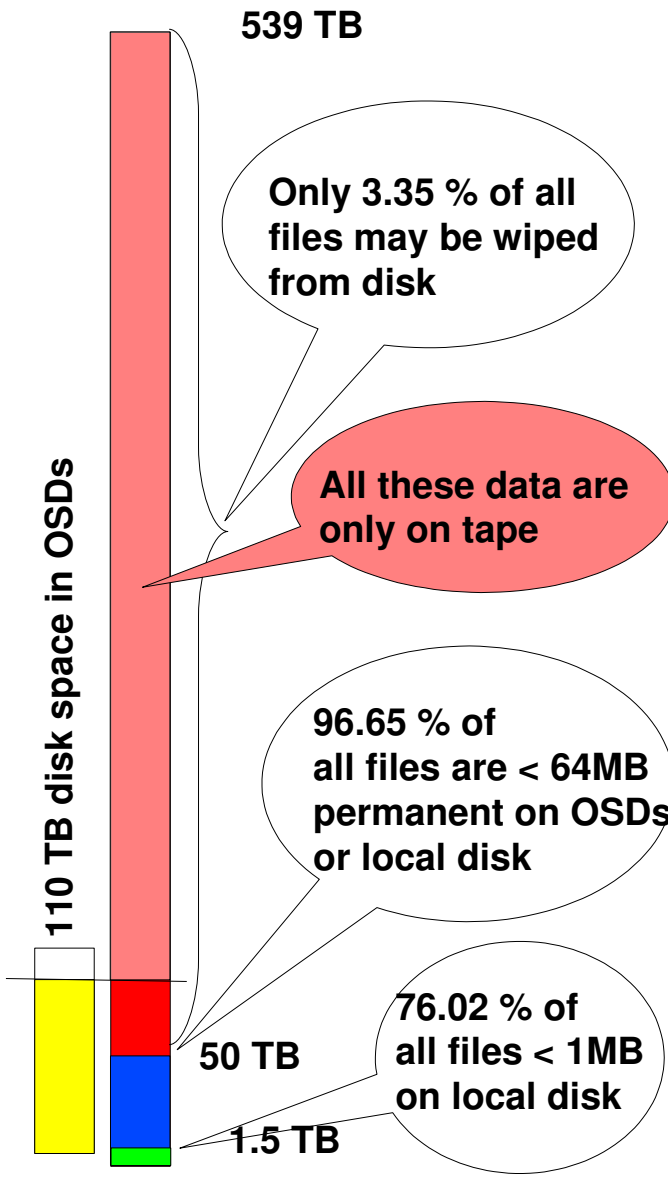
Total data 600 TB

~ 1/3 of the data in files > 2G

16.4 % of the files, but
89.9 % of the data in
volumes which may us OSD

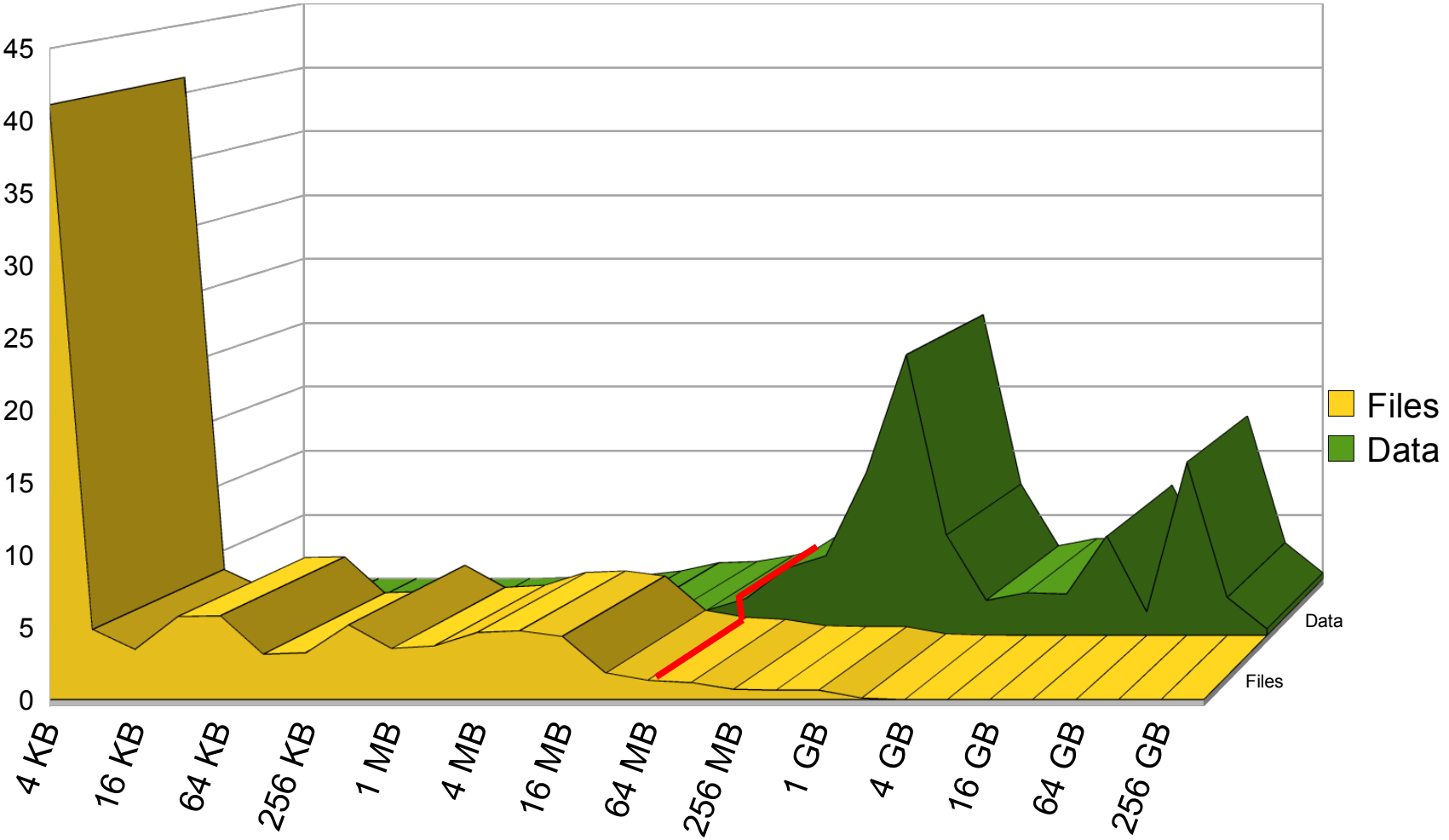
OSD-Volumes

File Size Range	Files	%	run %	Data	%	run %
0 B - 4 KB	10824005	41.13	41.13	10.495 GB	0.00	0.00
4 KB - 8 KB	1276485	4.85	45.98	6.781 GB	0.00	0.00
8 KB - 16 KB	909992	3.46	49.43	9.952 GB	0.00	0.00
16 KB - 32 KB	1508949	5.73	55.17	32.442 GB	0.01	0.01
32 KB - 64 KB	1522376	5.78	60.95	65.963 GB	0.01	0.02
64 KB - 128 KB	823253	3.13	64.08	69.613 GB	0.01	0.04
128 KB - 256 KB	850758	3.23	67.31	157.896 GB	0.03	0.06
256 KB - 512 KB	1361513	5.17	72.49	470.401 GB	0.09	0.15
512 KB - 1 MB	930117	3.53	**76.02**	672.312 GB	0.12	**0.27**
1 MB - 2 MB	974692	3.70	79.72	1.406 TB	0.26	0.53
2 MB - 4 MB	1219333	4.63	84.36	3.286 TB	0.61	1.14
4 MB - 8 MB	1250013	4.75	89.11	6.682 TB	1.24	2.38
8 MB - 16 MB	1146475	4.36	93.46	12.676 TB	2.35	4.73
16 MB - 32 MB	489300	1.86	95.32	9.792 TB	1.82	6.55
32 MB - 64 MB	350622	1.33	**96.65**	14.610 TB	2.71	**9.26**
64 MB - 128 MB	308325	1.17	97.82	26.592 TB	4.93	14.20
128 MB - 256 MB	186614	0.71	98.53	31.603 TB	5.86	20.06
256 MB - 512 MB	174590	0.66	99.20	64.450 TB	11.96	32.02
512 MB - 1 GB	167701	0.64	99.83	111.504 TB	20.69	52.71
1 GB - 2 GB	30242	0.11	99.95	39.973 TB	7.42	60.12
2 GB - 4 GB	5311	0.02	99.97	13.899 TB	2.58	62.70
4 GB - 8 GB	2881	0.01	99.98	16.997 TB	3.15	65.86
8 GB - 16 GB	1589	0.01	99.99	16.357 TB	3.04	68.89
16 GB - 32 GB	1774	0.01	99.99	39.602 TB	7.35	76.24
32 GB - 64 GB	933	0.00	100.00	41.606 TB	7.72	83.96
64 GB - 128 GB	782	0.00	100.00	68.721 TB	12.75	96.71
128 GB - 256 GB	98	0.00	100.00	15.214 TB	2.82	99.53
256 GB - 512 GB	6	0.00	100.00	2.524 TB	0.47	100.00
Totals:	**26318729 Files**			**538.973 TB**		

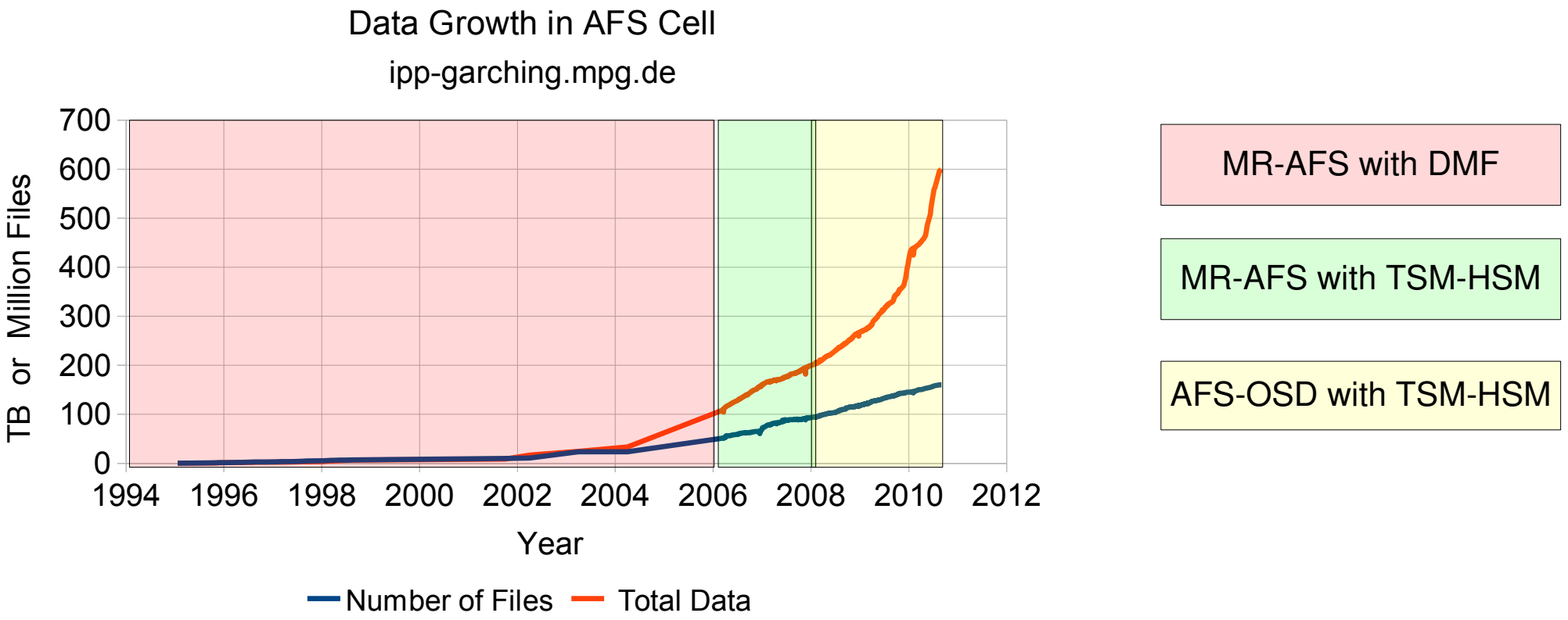


OSD Volumes

- 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 staying only on tape
  - This are only 3.35 % of the all files, but 90.74 % of the total data volume !



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

HSM systems

There are two classes of HSM systems:

1. Those you can see as a mounted filesystem, such as
   - DMF (SGI got it along with CRAY in 1996)
   - TSM-HSM (IBM)
   - SAMFS-QFS (SUN/ORACLE)
2. Those which can only be accessed through special libraries, such as
   - CASTOR (CERN)
   - DCACHE (DESY)
   - HPSS (IBM)

HPSS can also be mounted in LINUX, but they recommand for performance reasons the use through library calls.

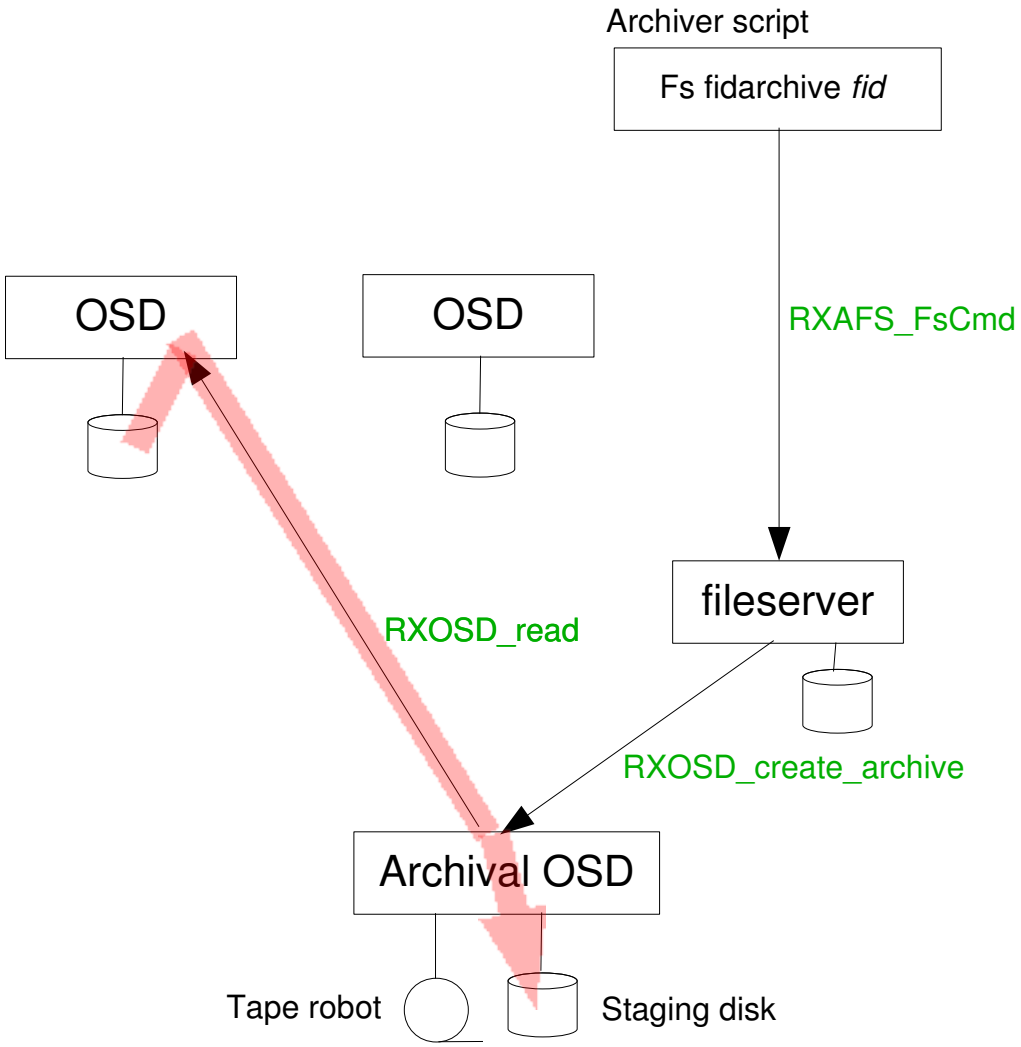
The 1st class doesn't require special support by AFS-OSD, of the 2nd class only DCACHE and HPSS are supported at the moment.

Non-mountable HSM systems

For the non-mountable HSM systems a special interface had to be provided:

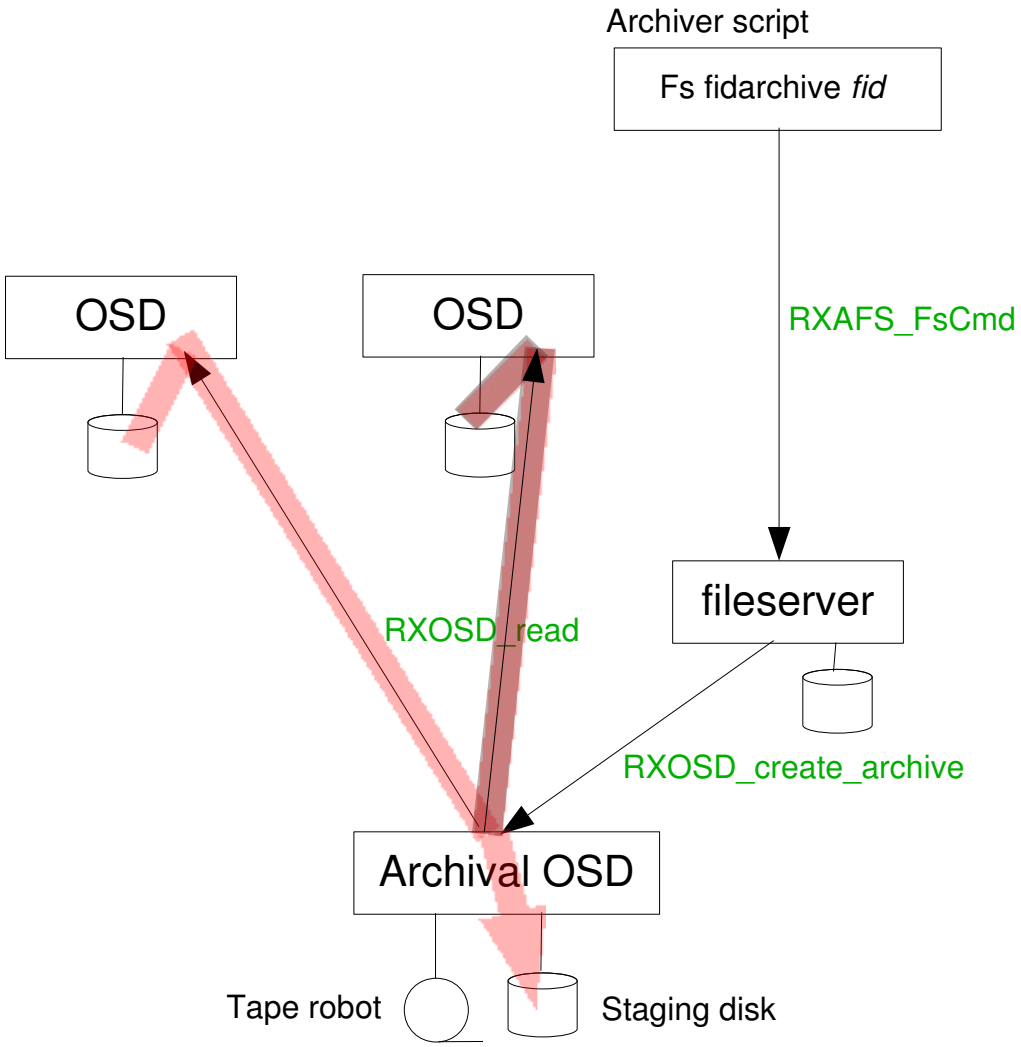
- The NAMEI-linktable should not reside in the HSM system to allow for fast increment and decrement of link counts (as needed with volume clone and release).
    - Therefore a small visible /vicep-partition is necessary which contains only the tree for the “special” directories.
- Instead of calling the C-library routines for I/O operations a pointer to an operations vector is supplied in IHandle_t. It can either point to the C-library routines or to the specific interface for the HSM system.
    - For normal partitions and for the linktable it points to the C-library
    - For files in HPSS or DCACHE it points to the interface routines.
- The base directory path where the NAMEI data structure begins in the non-mountable HSM system has to be supplied by a command line parameter to the rxosd.

Archiving a simple file into HSM



- Archiver script: „fs fidarchive *fid*“
- Fileserver does RPC RXOSD_create_archive to archival OSD
- Archival OSD does RPC RXOSD_read to OSD where the file is stored
- Archival OSD returns md5 checksum to fileserver
- Fileserver stores md5 checksum in file's OSD metadata.
- HSM-system migrates file from staging disk to tape.

Archiving a striped file



- Archiver script: „fs fidarchive <FID>“
- Fileserver does RPC RXOSD_create_archive to archival OSD
- Archival OSD does RPCs RXOSD_read to OSDs where stripes of the file are stored
- Archival OSD returns md5 checksum to fileserver
- Fileserver stores md5 checksum in file's OSD metadata.
- TSM-HSM migrates file from staging disk to tape.

Direct Archiving to HPSS

All LINUX-based OSDs can use the HPSS api-library to access files in HPSS directly. This allows to archive files more efficiently.

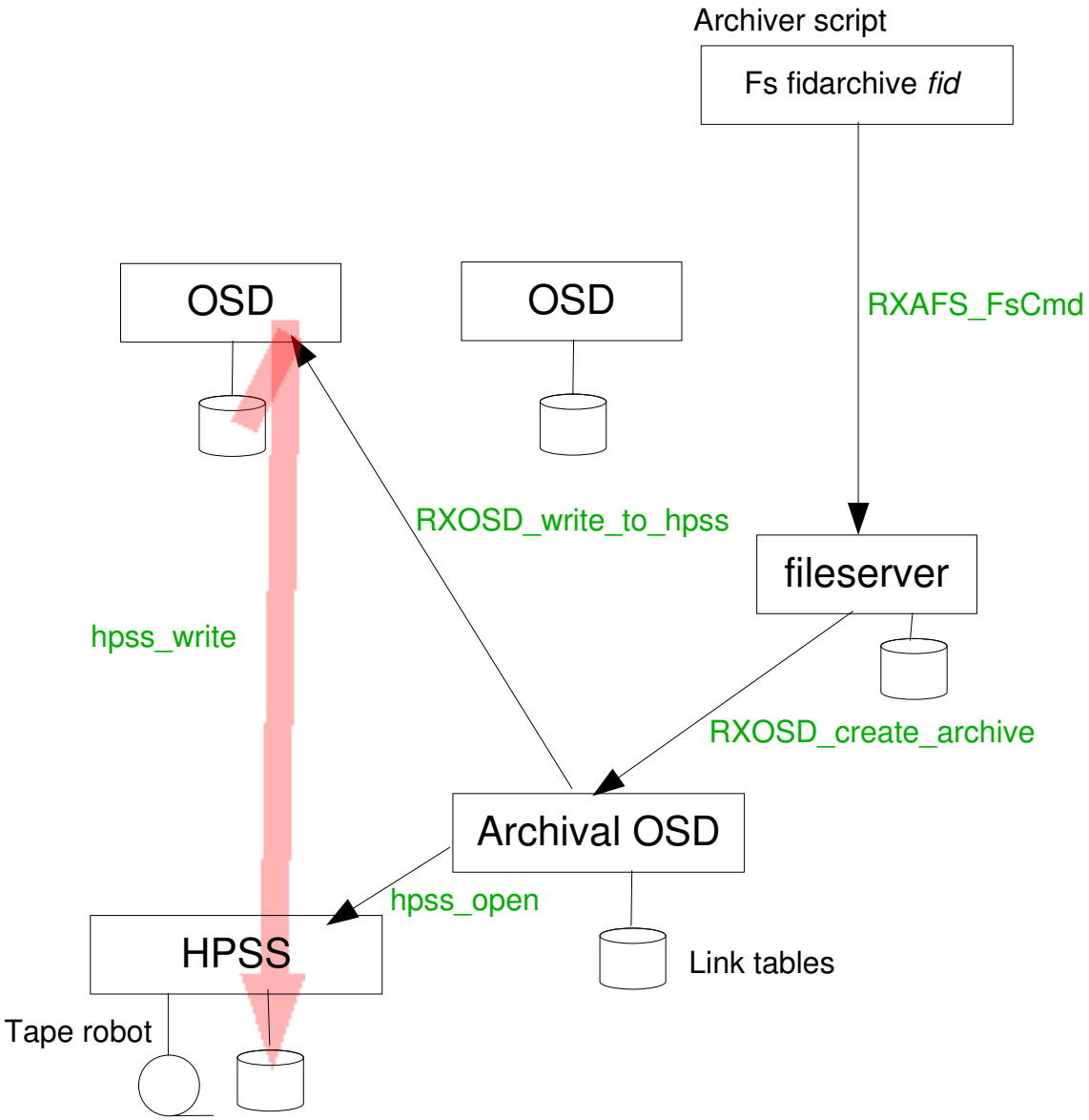
- The archival OSD which owns HPSS and keeps the link-table creates the archival file in HPSS and than starts a new RPC “write_to_hpss” to the OSD where the file resides.
- The OSD where the file resides sends directly its data to HPSS and calculates the md5 checksum on the fly.
- The archival OSD returns the md5 checksum to the fileserver.

The archival OSD owning HPSS can handle many those archiving requests in parallel because it only controls the archiving without actually transfering data.

This technique can be used only for files consisting in a single object. Striped files can still only be archived with the old method.

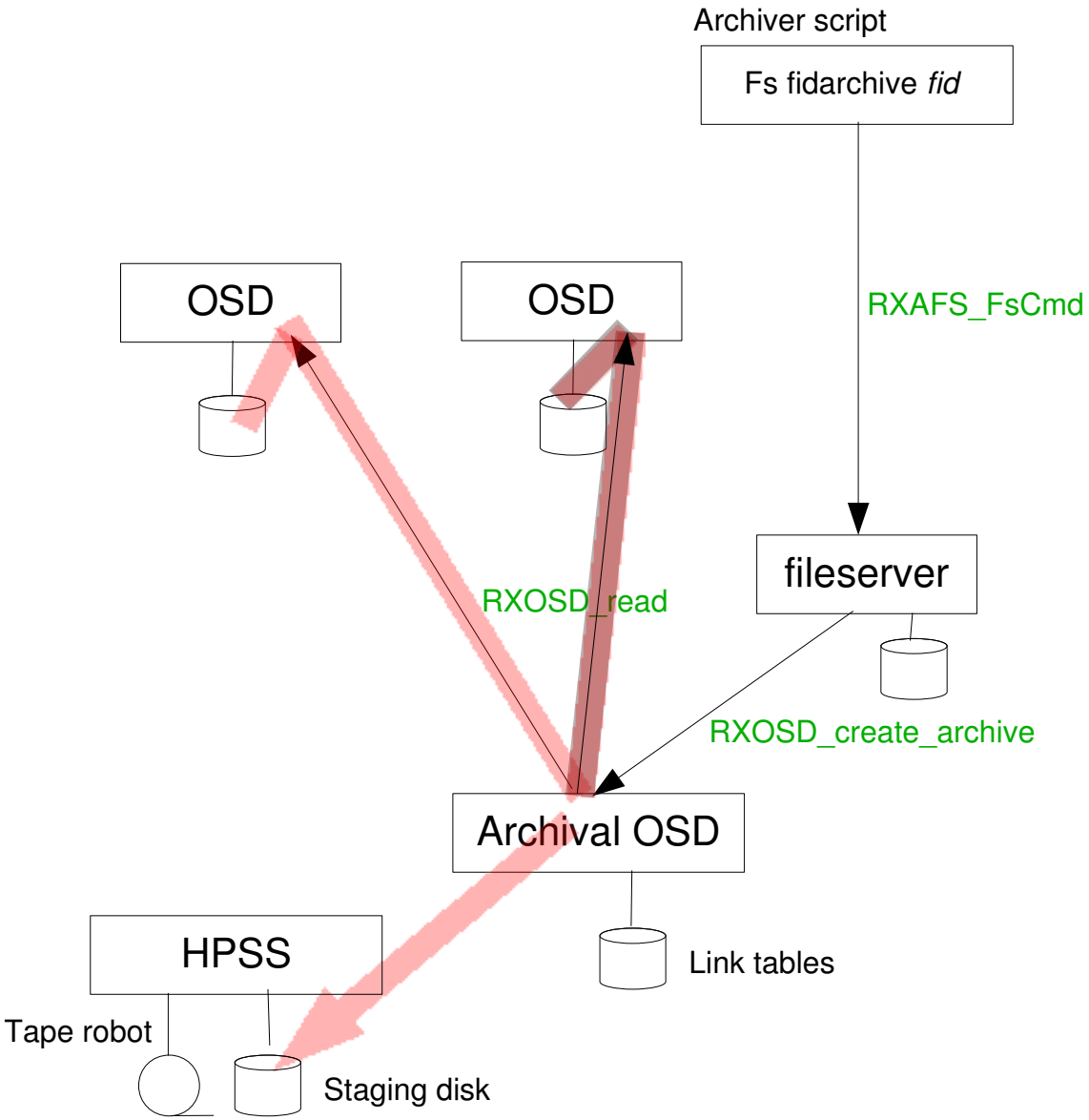
The OSDDB informs the archival OSD which OSDs can see HPSS directly (a new flag in the OSDDB to be set by the administrator).

Archiving simple files to HPSS



- Archiver script: „fs fidarchive <FID>“
- Fileserver does RPC RXOSD_create_archive to archival OSD
- Archival OSD creates file in HPSS
- Archival OSD does RPC RXOSD_write_to_hpss to OSD
- OSD sends data to HPSS mover and calculates md5 checksum
- Archival OSD returns md5 checksum to fileserver
- Fileserver stores md5 checksum in file's OSD metadata.
- HPSS migrates file from staging disk to tape.

Archiving a striped file to HPSS



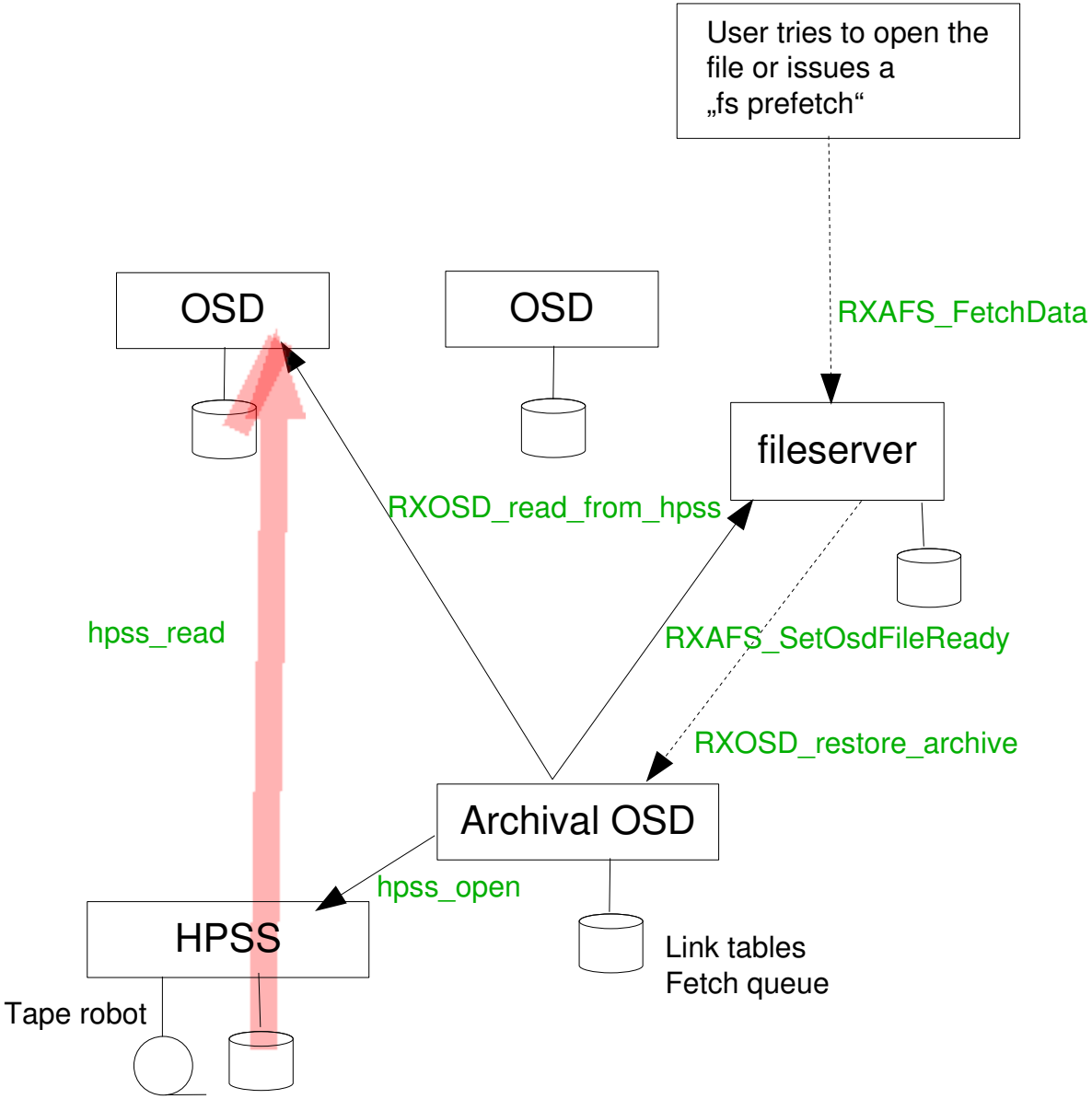
- Archiver script: „fs fidarchive <FID>“
- Fileserver does RPC RXOSD_create_archive to archival OSD
- Archival OSD does RPCs RXOSD_read to OSDs where stripes of the file are stored and sends data to HPSS mover
- Archival OSD returns md5 checksum to fileserver
- Fileserver stores md5 checksum in file's OSD metadata.
- HPSS migrates file from staging disk to tape.

Recall of files from HPSS

In analogy to the archiving also the recall can be delegated to the OSD where the file should reside (in case the OSD is able to access HPSS directly):

- The fileserver chooses an OSD, creates an object there and stores the information in the file's metadata. Then it issues an RXOSD_restore_archive RPC to the archival OSD which owns HPSS.
- The archival OSD queues the request in the fetch queue. It allows only a certain number of recalls to run in parallel.
- When the recall was started and the file came on-line inside HPSS the archival OSD does a RXOSD_read_from_hpss RPC to the target OSD.
- The target OSD gets the data from HPSS, calculates the md5 checksum and stores the data in the object.
- At the end the archival OSD does a RXAFS_SetOsdfileReady RPC to the fileserver which compares the md5 checksums and sets the file usable.

Recall of a file from HPSS



- User tries to open file
- Fileserver does RPC RXOSD_restore_archive to archival OSD, which returns „file on tape“
- Archival OSD queues request
- Archival does hpss_open to bring file on-line in HPSS
- Archival OSD does RXOSD_read_from_hpss to OSD
- OSD gets data from HPSS mover and calculates md5 checksum
- Archival sends md5 checksum with RXAFS_SetOsdFileReady to fileser
- Fileserver compares checksums and sets file usable

Building 1.4.12 for HPSS

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 new options:

--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

You need also, of course, --enable-object-storage

Building 1.5.76 for HPSS

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-1.5-osd/

Do as usual configure and make.

configure has new options:

--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

You need also, of course, --enable-object-storage

Questions or comments?

Thank you