Everything You Always Wanted to Know About AFM * But Were Afraid to Ask

jochen.zeller@sva.de

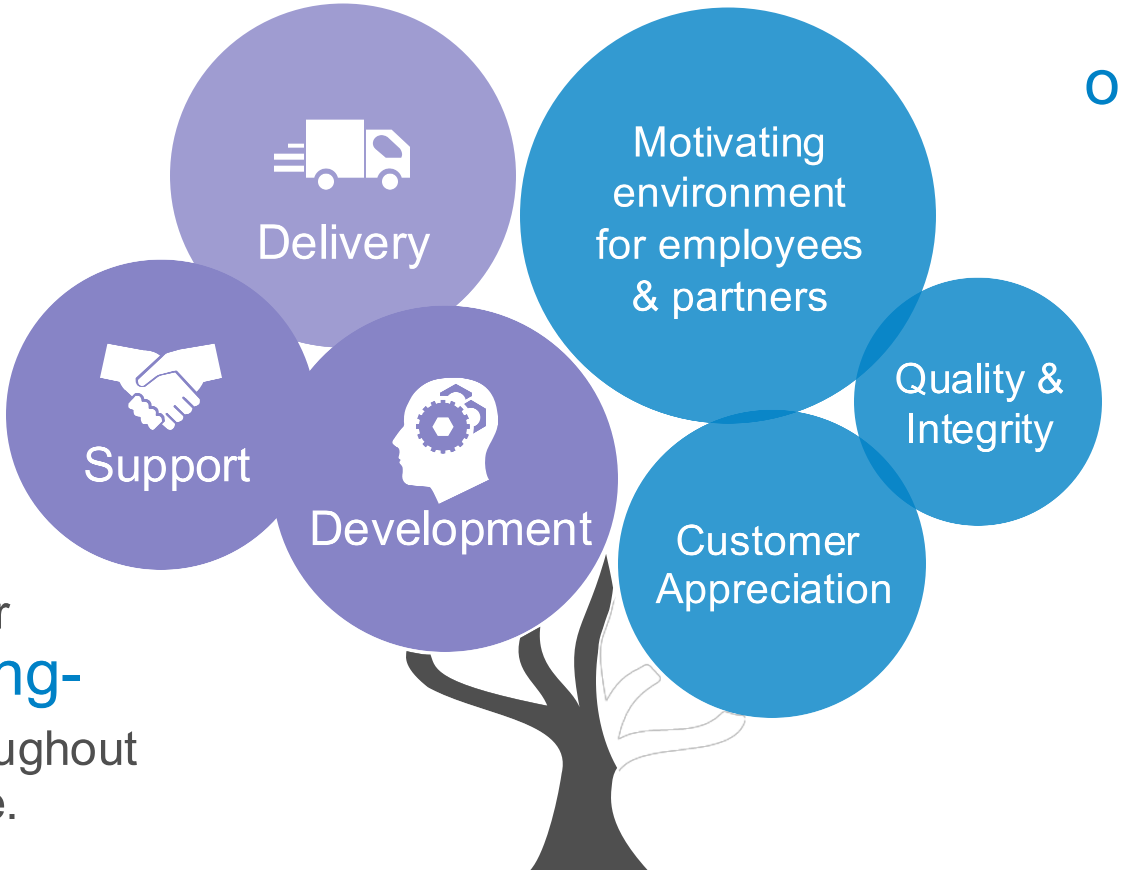
Profile and corporate objective

Biggest owner-operated system integrator in Germany

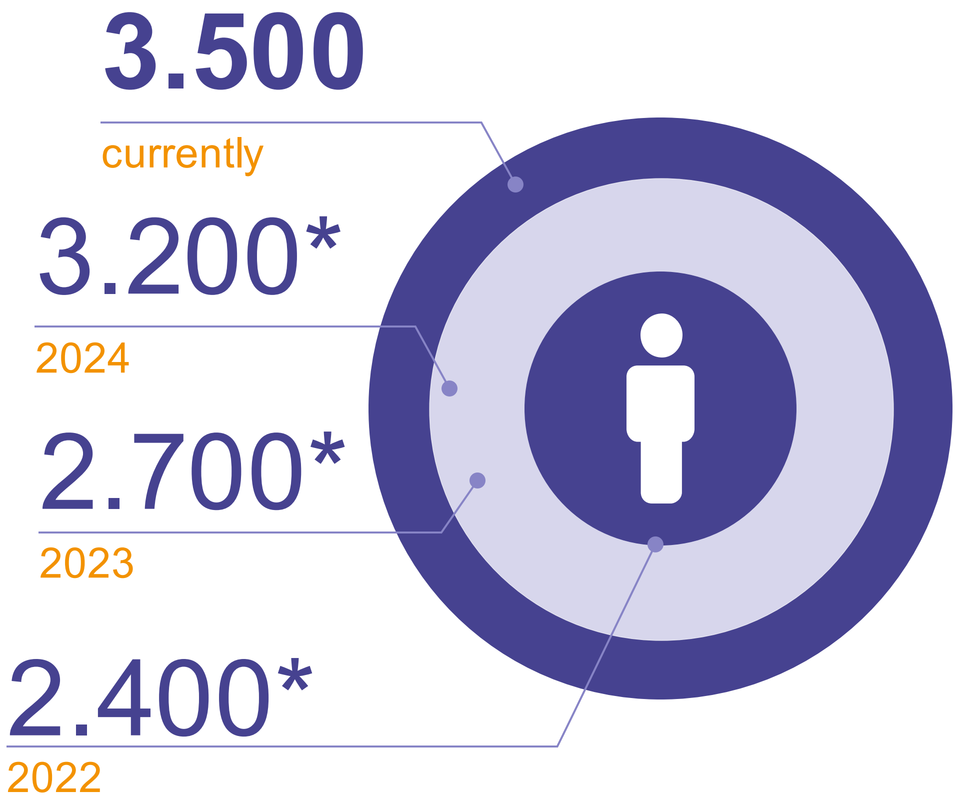
Steady growth with more than 3.500 employees in Germany

We accompany our customers on a long-term basis throughout the project lifecycle.

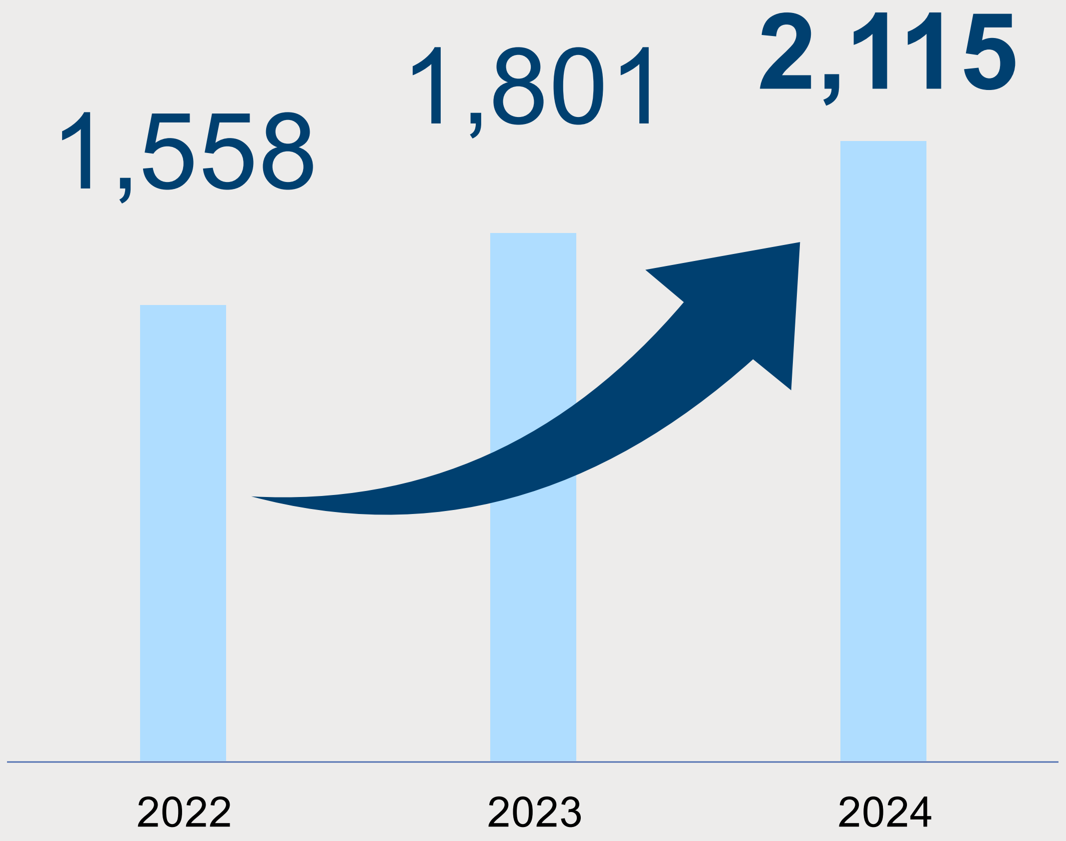
Company objectives



Employees



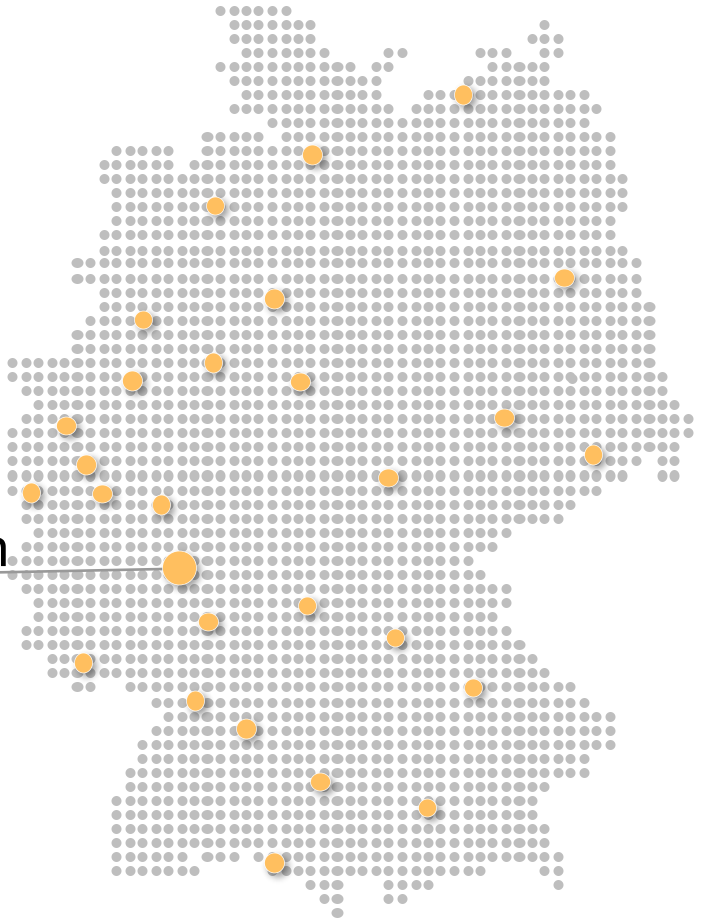
Sales volume in Mio €



*Average number of respective year

Company

28
Locations



INDUSTRIES
such as

AUTOMOTIVE

RETAIL

PUBLIC

MACHINE & PLANT CONSTRUCTION

FINANCE & INSURANCE

TELEKOMMU-NICATION

Disclaimer

⇢ Whenever I say GPFS, I mean IBM Storage Scale.

What to expect in my two sessions

- AFM
- AFM
- AFM
- AFM
- AFM
- AFM

- AFM
- AFM
- AFM
- AFM
- FCMs are cool
- Beware of FCMs

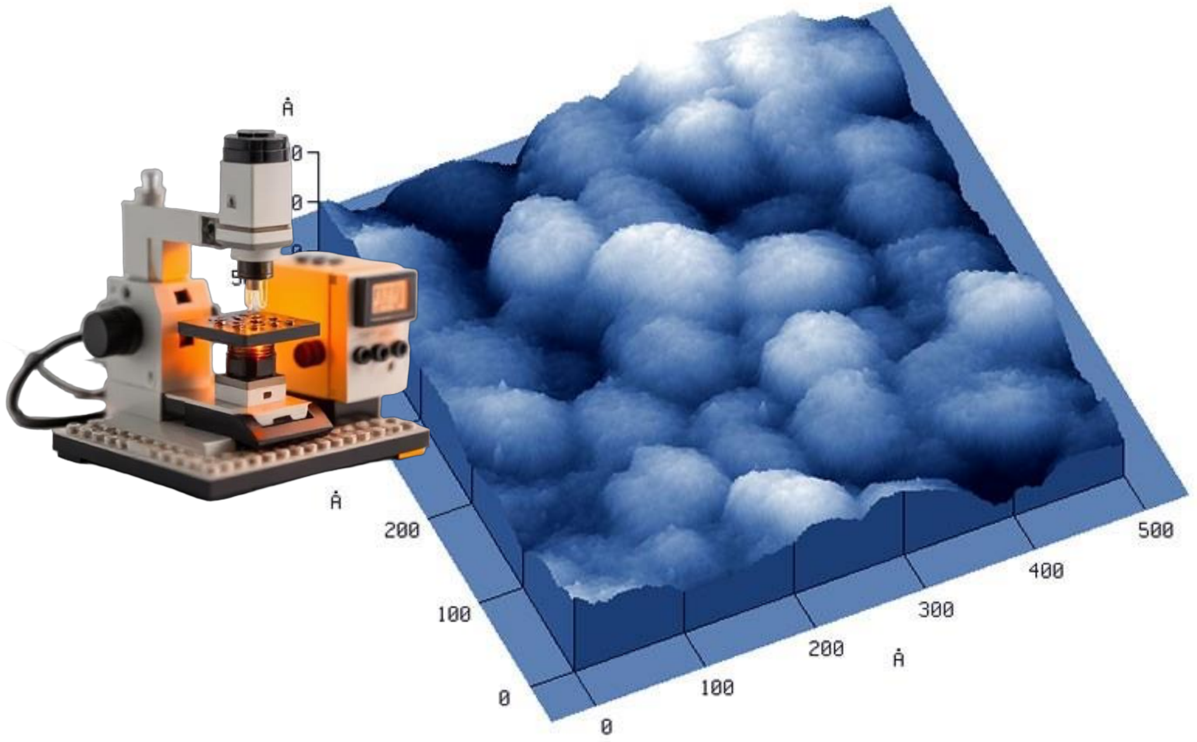
Before we start - What is AFM?

- What does Google answer when you enter "AFM"?
  - *AFM – **A**tomic **F**orce **M**icroscopy*

  ZONK! – here and now, false!
  - And if I ask Google "AFM Jochen Zeller"?

    First hit:

    Not enough money for an all flash HPC storage
  - That brings us closer to the point :)



- IBM Storage Scale AFM – **A**ctive **F**ile **M**anagement
  - But that doesn't make anyone smarter
- What does ChatGPT say about this?
  - *IBM Storage Scale AFM (Active File Management) is a feature of IBM Storage Scale (formerly known as IBM Spectrum Scale) that enables efficient management and synchronization of files across multiple locations. It is particularly beneficial for organizations that need to manage data in distributed environments and optimize access to that data…* ✓

Before we start - What is AFM?

AFM relation

- AFM HOME
  - Home of the data

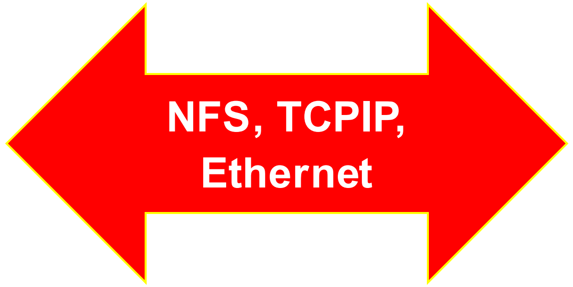
- AFM CACHE
  - Cache shows content of home
  - Reads data on access from cache
  - Is able to prefetch data
  - Can also write back changes

1st use case - simplify and accelerate

NFS Server

/home 30TB

/department 200TB



HPC Server

Automounter:
/home/<user>
/department/<dep>

Automounter:
/home/<user>
/department/<dep>

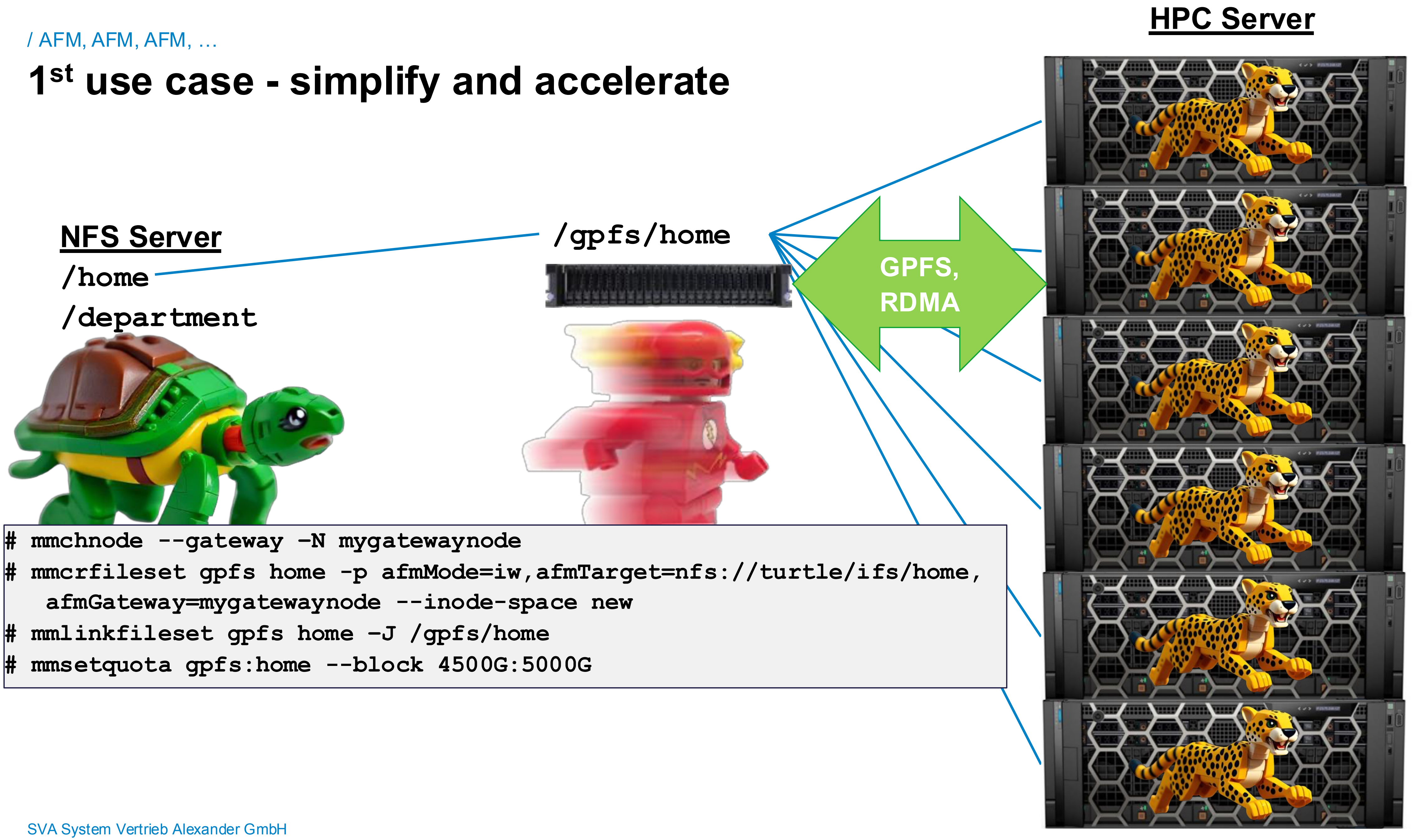
Automounter:
/home/<user>
/department/<dep>

Automounter:
/home/<user>
/department/<dep>

Automounter:
/home/<user>
/department/<dep>

Automounter:
/home/<user>
/department/<dep>

1st use case - simplify and accelerate



```
# mmchnode --gateway -N mygatewaynode
# mmcrfileset gpfs home -p afmMode=iw,afmTarget=nfs://turtle/ifs/home,
   afmGateway=mygatewaynode --inode-space new
# mmlinkfileset gpfs home -J /gpfs/home
# mmsetquota gpfs:home --block 4500G:5000G
```

1st use case - simplify and accelerate

- GPFS Client view:

```
# df -h /gpfs/home
Filesystem Size  Used Avail Use% Mounted on
gpfs         5T    0T    5T   1% /gpfs
# ls /gpfs/home
user1 user2 user3 user4 …
```

- How to use this home directory?
  - Example, SSSD, sssd.conf

```
override_homedir = /gpfs/home/%u
```

1st use case - simplify and accelerate

⇢ AFM Gateway node view:

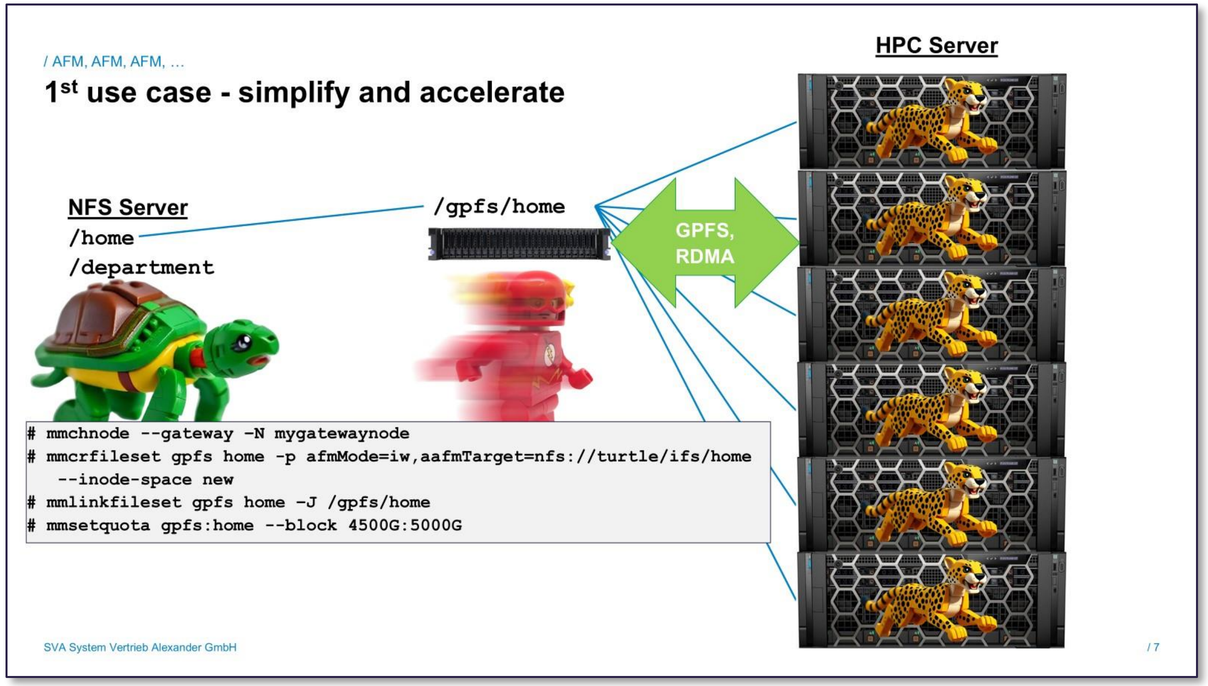
```
# df -h | grep afm
turtle:/ifs/home    100T  85T  15T   85% /var/mmfs/afm/datafs-1/mnt

# ls /gpfs/home
user1 user2 user3 user4 …
# ls /var/mmfs/afm/datafs-1/mnt
user1 user2 user3 user4 …

# mmlsfileset gpfs home
Filesets in file system 'gpfs':
Name                 Status    Path
home                 Linked    /gpfs/home

# mmafmctl gpfs getstate
Fileset Name    Fileset Target           Cache State         Gateway Node    Queue Length   Queue numExec
------------    --------------           -------------       ------------    ------------   -------------
home            nfs://turtle/ifs/home    Active              myGatewayNode   0              77
```

And what do we achieve with this?



```
# mmchnode --gateway -N mygatewaynode
# mmcrfileset gpfs home -p afmMode=iw,aafmTarget=nfs://turtle/ifs/home
   --inode-space new
# mmlinkfileset gpfs home -J /gpfs/home
# mmsetquota gpfs:home --block 4500G:5000G
```

- No NFS and no automounter in HPC cluster.

  (I don't need to explain that any further)
- All writes go directly to GPFS and are cached in GPFS
- For those who don't know: GPFS makes it faster!
- The fileset quota can be used to control how much data remains in GPFS. Soft and hard limits should be close to each other, e.g. 950GB : 1000GB.
- The AFM fileset evicts data before the hard limit is reached. Similar to HSM, the data is removed from the inode and then refers to the AFM source (= AFM HOME).
- I almost forgot. The writes in GPFS are transferred asynchronously to the NFS server. If there have been changes on the NFS server, AFM will detect this directly when a file or directory is opened, with a new lookup.

Is this AFM caching also possible with other storages???

- Yes, it is!
- Of course with GPFS, direct (*Not enough money for …*) or via remote cluster mount

```
# mmcrfileset gpfs hpc_cache -p afmMode=iw,afmTarget=gpfs:///gpfs/hpc_home --inode-space new
# mmlinkfileset gpfs home –J /gpfs/home
# mmsetquota gpfs:home --block 4500G:5000G
```

- And with object storage (AWS S3, IBM COS, Microsoft Azure Blob, Google Cloud Platform, Seagate Lyve Cloud)

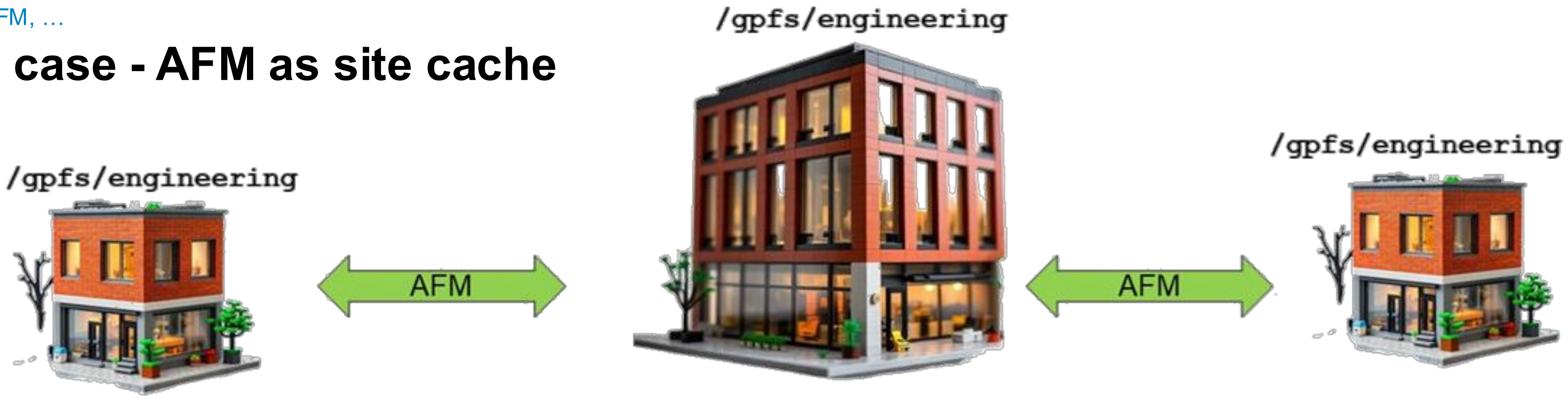
```
# mmafmcoskeys mybucket set myakey:mysecret
# mmafmcosconfig mygpfs mybucket --endpoint http://myOBJstorage --bucket mybucket --object-fs --uid
1000 --gid 1000 --dir /gpfs/mybucket --mode iw
# mmsetquota mygpfs:mybucket --block 450G:500G
```

… but more on that later …

Can I get started with AFM now?

- If you think: I've understood it, ready to go!
  - Stay down
  - Listen!
  - Could become exciting

2nd use case - AFM as site cache



- Three site “whatever parts” engineering company
- All engineers use the same applications, which are started from a local share
- Engineers work with the same data at all three locations, but the same projects / files are not processed at different sites at the same time
- Distance >200km between sites, >30ms latency

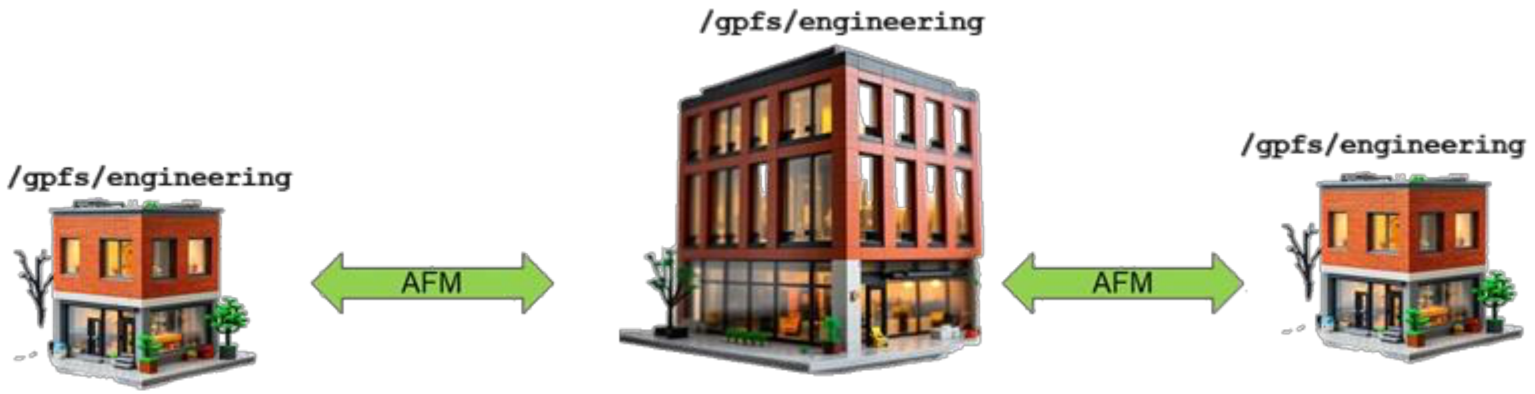
2nd use case - AFM as site cache

- Headquarter: 2PB ESS3500 + 4 CES nodes to export data by SMB and NFS
- Sites: 100TB 3-node cluster
  - Node 1+2 = NSD Server + CES node, node 3 = AFM gateway node
- AFM IW (independent writer) / RO (read only) filesets on sites, point to NFSv4 share in headquarter

AFM caches to NFS HOME
`/gpfs/engineering/sw`
`/gpfs/engineering/data`

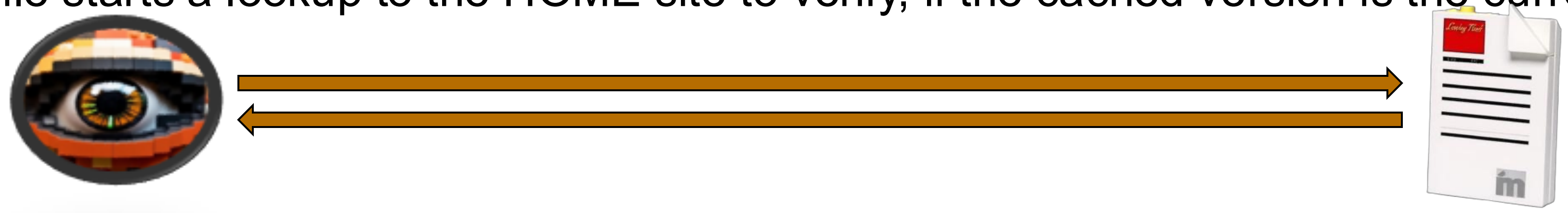
NFSv4 export
`/gpfs/engineering/sw`
`/gpfs/engineering/data`

AFM caches to NFS HOME
`/gpfs/engineering/sw`
`/gpfs/engineering/data`



2nd use case - AFM as site cache

- Latency challenge
  - When an engineering application starts, it loads hundreds of libraries in the background
  - Each read of a file starts a lookup to the HOME site to verify, if the cached version is the current



  - 500 sequential file lookup x 30 ms = 15 seconds additional latency, before the application starts
  - Since AFM works with a queue and does not always send a lookup immediately, in reality 15 seconds quickly become 30 - 60 seconds until the application is started
  - We tinkered, tested, called Venkat, tinkered again, destroyed a chache, repaired, tested again, I drove crazy, called Venkat again, we adjusted and tested again, and after 3 months:

2nd use case - AFM as site cache

```
Attributes for fileset sw:
===============================
Status                                  Linked
Path                                    /gpfs/engineering/sw
Id                                      6
Root inode                              59299
Parent Id                               0
Created                                 Fri Nov  11 12:34:12 2024
Comment
Inode space                             4
Maximum number of inodes                20971520
Allocated inodes                        6924288
Permission change flag                  chmodAndUpdateAcl
IAM mode                                off
afm-associated                          Yes
Permission inherit flag                 inheritAclOnly
Target                                  nfs://headquarter/gpfs/engineering/sw
Mode                                    read-only
File Lookup Refresh Interval            disable
File Open Refresh Interval              disable
Dir Lookup Refresh Interval             disable
Dir Open Refresh Interval               disable
Async Delay                             disable
Expiration Timeout                      disable (default)
Last pSnapId                            0
Display Home Snapshots                  no (default)
Number of Read Threads per Gateway      8
Number of Gateway Flush Threads         8
Prefetch Threshold                      50
Eviction Enabled                        no
IO Flags                                0x0
IO Flags2                               0x40 (afmSyncReadMount)
```

The fileset is read-only, because the shares are also read-only, so that we don't have to bother with protocol locking. Everything for speed.

One solution could be: disable lookups and prefetch updates once a day. But be careful, if lookups are disabled, you have to do the prefetch with --force, otherwise nothing will be updated:

```
# mmafmctl gpfs prefetch -j sw --prefetch-threads
32 --directory /gpfs/engineering/sw --force
```

Partial prefetches are also available, e.g. if only one application has received updates:

```
# mmafmctl gpfs prefetch -j sw --prefetch-threads
32 --directory
/gpfs/engineering/sw/mathCalc/v22.06 --force
```

Of course, eviction must be switched off and all files for the applications must be kept locally.

2nd use case - AFM as site cache

- Not enough space & fast access challenge for data
  - The caches have only 5% of the capacity of the home.
  - By the way, an “ls” or opening a directory in a Windows browser also triggers a lookup for the complete content (not recursive). There are also AD queries for users, groups and ACLs. That takes time.

```
# mmlsfileset gpfs data -X
...
Mode                                    independent-writer
File Lookup Refresh Interval            60
File Open Refresh Interval              60
Dir Lookup Refresh Interval             60
Dir Open Refresh Interval               60
Async Delay                             15
Last pSnapId                            0
Display Home Snapshots                  no
Number of Read Threads per Gateway      16
Number of Gateway Flush Threads         16
Prefetch Threshold                      50
Eviction Enabled                        yes (default)
Number of Write Threads per Gateway     4
IO Flags                                0x40200 (afmRefreshAsync,afmFastCreate)
IO Flags2                               0x40 (afmSyncReadMount)
```

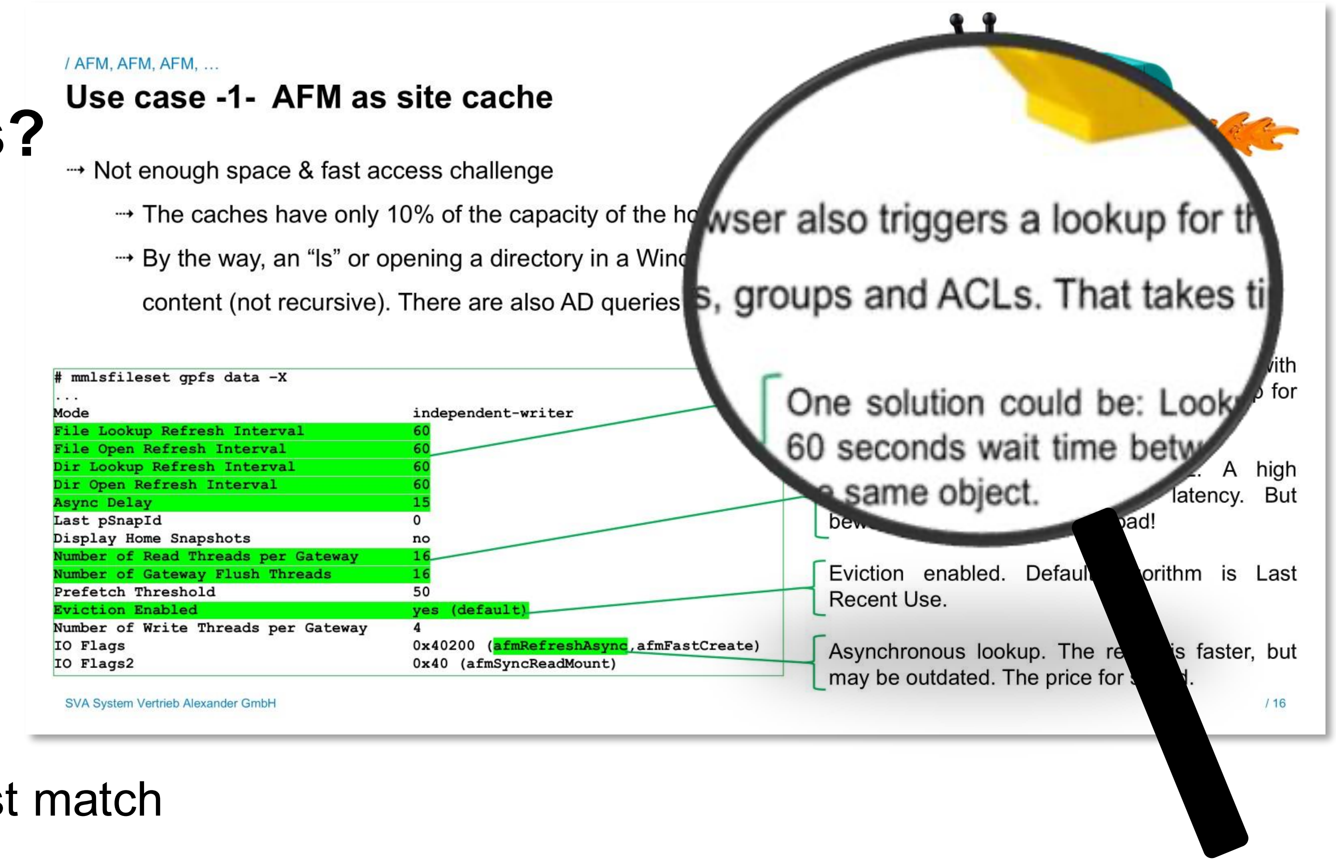
One solution could be: Lookup is enabled, with 60 seconds wait time between a new lookup for the same object.

16 threads to read from HOME. A high parallelism to compensate the latency. But beware, this could result in a high load!

Eviction enabled. Default algorithm is Last Recent Use.

Asynchronous lookup. The result is faster, but may be outdated. The price for speed.

I saw something. AFM + NFS + ACLs?



Use case -1- AFM as site cache

- Not enough space & fast access challenge
  - The caches have only 10% of the capacity of the ho wser also triggers a lookup for th
  - By the way, an “ls” or opening a directory in a Wind
    content (not recursive). There are also AD queries s, groups and ACLs. That takes ti

```
# mmlsfileset gpfs data -X
...
Mode                                    independent-writer
File Lookup Refresh Interval            60
File Open Refresh Interval              60
Dir Lookup Refresh Interval             60
Dir Open Refresh Interval               60
Async Delay                             15
Last pSnapId                            0
Display Home Snapshots                  no
Number of Read Threads per Gateway      16
Number of Gateway Flush Threads         16
Prefetch Threshold                      50
Eviction Enabled                        yes (default)
Number of Write Threads per Gateway     4
IO Flags                                0x40200 (afmRefreshAsync,afmFastCreate)
IO Flags2                               0x40 (afmSyncReadMount)
```

vith
p for
One solution could be: Look
60 seconds wait time betw
e same object.

A high latency. But
bew ad!

Eviction enabled. Defaul orithm is Last Recent Use.

Asynchronous lookup. The re is faster, but may be outdated. The price for s d.

SVA System Vertrieb Alexander GmbH / 16

- Yes! AFM is able to translate and store NFSv4 ACLs into GPFS ACLs.
- Even if the AFM documentation still claims otherwise in many pages
- But be careful, the user and group resolution must match
- The users and groups displayed by nfs4_getfacl must be resolved correctly with getent passwd / group

I saw something. AFM + NFS + ACLs?

```
# root@mygpfs01:/mnt/nfs/jz # nfs4_getfacl data
# file: data
A::OWNER@:rwaDxtTnNcCoy
A::GROUP@:rxtncy
A::EVERYONE@:rxtncy
A:fdi:jzeller:rwaDdxtTnNcy
```

⇢ In order to resolve these ACLs correctly and to be stored by AFM as GPFS ACLs

```
# root@mygpfs01:/mnt/nfs/jz # getent passwd jzeller
jzeller:*:0:0:root:/home/jzeller:/bin/bash
```

⇢ By the way, the gpfs-winbind on CES nodes is not suitable, it resolves SVA\jzeller (more on AFM + CES in a moment)

⇢ This means that you can also migrate SMB shares, e.g. from a Netapp filer, via an NFSv4 share to GPFS

2nd use case - AFM as site cache

- In this use case, there are 3 nodes at each site, but only one is an AFM gateway. Why?
  - Initially we had 3 CES nodes with the additional role AFM Gateway
  - But every lookup is a thread, even if it is waiting for a response
  - In real life, we quickly had a load of >50 on our nodes (4 filesets x 16 threads per fileset)
  - From the point of view of Windows Share users, the file system was no longer responsive
  - There are also situations that trigger an AFM fileset recovery. This recovery starts a policy, which also runs with 24 threads. If a second or third fileset recovers, there are immediately +48 or +72 threads. And the node may run out of memory.
  - By the way, separate AFM gateways are also the recommendation, but I didn't believe it

```
# mmchconfig afmMaxParallelRecoveries=2 -i
```

2nd use case - AFM as site cache

- Another important note about gateway nodes and parallelism
  - If I do not specify a gateway node, AFM sets “all” as gateway node
  - This means that each gateway node mounts the NFS share and the tasks run parallel
  - Initially, I also used this in this project, according to the motto “more parallel is faster”
  - The punishment came quickly: MMFS_FSSTRUCT errors
  - IBM statement was that this is probably due to the latency
  - MMFS_FSSTRUCT has happened also in two other projects. I no longer use “parallel”

```
# mmchfileset FS FSET -p afmGateway=gwnode1
```

  - But I use the nfs option “nconnect” to increase NFS parallelism

```
# mmchconfig afmNFSNconnect=4 -i -N AFMnodes
```

3rd use case - file data migration (extra gold plus)

- Several Isilons, some older and >90% filled, will be replaced by SSS6000
- The data is used in research.
- A lot of data has not been read for a long time, but nobody knows whether it is still needed
- The wish was to use HSM in GPFS and to migrate data based on atime (r.g. atime > 5 years)
- But AFM reads the file in the source (atime = read time), creates in GPFS a new file with new ctime + atime
- It would be so nice to have a “file data migration (extra gold plus)” with unchanged atime

… I called Venkat …

3rd use case - file data migration (extra gold plus)

⇢ Are you interested?

⇢ Part II at 4:20 pm

3rd use case - file data migration (extra gold plus)

- Several Isilons, some older and >90% filled, will be replaced by SSS6000
- The data is used in research.
- A lot of data has not been read for a long time, but nobody knows whether it is still needed
- The wish was to use HSM in GPFS and to migrate data based on atime (r.g. atime > 5 years)
- But AFM reads the file in the source (atime = read time), creates in GPFS a new file with a new ctime + atime
- It would be so nice to have a “file data migration (extra gold plus)”
- … I called Venkat …

3rd use case - data migration (extra gold plus)

⇢ Many thanks to Venkat and the AFM team, best AFM dev team ever!

```
# mmchconfig afmRevalOpt=255 -i -N AFMnodes
# mmachconfig afmMountOpt=noatime -i -N AFMnodes
```

⇢ Attention: this is still an efix and not yet available and documented

⇢ But it works wonderfully ☺

File in SOURCE via NFS after migration:

```
stat /var/mmfs/afm/mnt/dir/file.txt
  File: /var/mmfs/afm/mnt/dir/file.txt  Size: 167255
…
Access: 2022-03-15 16:32:14.000000000 +0100
Modify: 2022-03-15 16:32:14.000000000 +0100
Change: 2023-06-06 13:33:08.293243000 +0200
 Birth: -
```

File in AFM fset after migration:

```
# stat /gpfs/afm/dir/file.txt
  File: /gpfs/afm/dir/file.txt  Size: 167255
…
Access: 2022-03-15 16:32:14.000000000 +0100
Modify: 2022-03-15 16:32:14.000000000 +0100
Change: 2023-06-06 13:33:08.293243000 +0200
 Birth: -
```

3rd use case - data migration (extra gold plus)

- Data migration steps:

```
# mmafmconfig add isilonXY --export-map isilonXY.dns.name/afm-gw-01,isilonXY.dns.name/afm-gw-02

# mmcrfileset <FS> <new fset> -p afmMode=lu,afmTarget=nfs://isilonXY/ifs/data/to/migrate,
afmFastCreate=yes,afmNumReadThreads=4,afmPrefetchThreshold=100,afmEnableAutoEviction=no,
afmGateway=afm-gw-01 --inode-space new --inode-limit 50M: --allow-permission-change chmodAndUpdateAcl

# mmlinkfileset <FS> <fset> -J /gpfs/fti/fset-dir

# mmafmctl <FS> prefetch -j <fset> --directory /gpfs/fti/fset-dir --metadata-only --prefetch-threads 8
```

- `afmMode=lu` means “local update”
  - The AFM fileset reads all data from the NFS share, but changes are not written back
  - We switch the client access to GPFS directly after creating the AFM FSET and prefetching the metadata
    - Metadata prefetch detects e.g. hard links

3rd use case - data migration (extra gold plus)

- Prefetching the remaining data while the clients are working in the new fset:

```
# mmafmctl <FS> prefetch -j <fset> --directory /gpfs/fti/fset-dir --prefetch-threads 2
```

- Not yet prefetched data will be prefetched on access
- Each AFM fileset uses two queues, a priority queue for “on access” prefetches and a normal queue for prefetches
- When prefetch is done:

```
# mmafmctl <FS> checkUncached -j <fset>
…All the data is cached…
```

- If not everything is cached, you get a “dir-list” with not prefetched directories:

```
# mmafmctl <FS> prefetch -j <fset> --dir-list-file "dir-list" --force
```

3rd use case - data migration (extra gold plus)

⇢ Disable the lookups to improve access time:

```
# mmchfileset <FS> <fset> -p afmRefreshOnce=yes
# mmchfileset <FS> <fset> -p afmReaddirOnce=yes
```

⇢ And in the next maintenance window

```
# mmunlinkfileset <FS> <fset> -f
# mmchfileset <FS> <fset> -p afmTarget=disable
# mmlinkfileset <FS> <fset> -J /gpfs/fti/fset-dir
```

⇢ And now we can run mmbackup and HSM over the data

⇢ Attention again: This is my shorten procedure for this data migration case. The almost entire docu:

https://www.ibm.com/docs/en/storage-scale/5.2.2?topic=mdbuafm-data-migration-afm-fileset-by-using-nfs-

4th use case - synch file data with S3 bucket

- Unfortunately, it is difficult to find the right syntax for each object storage
- Example for AWS S3:

Includes access key and secret key

```
# mmafmcoskeys my-gpfs-bucket:eu-north-1@s3.eu-north-1.amazonaws.com set --keyfile aws.key
# mmafmcosconfig mygpfs s3sync --endpoint https://eu-north-1@s3.eu-north-1.amazonaws.com --dir s3sync --
prefix fromGPFS --mode iw  --object-fs --debug --cleanup --bucket my-gpfs-bucket --fast-readdir
```

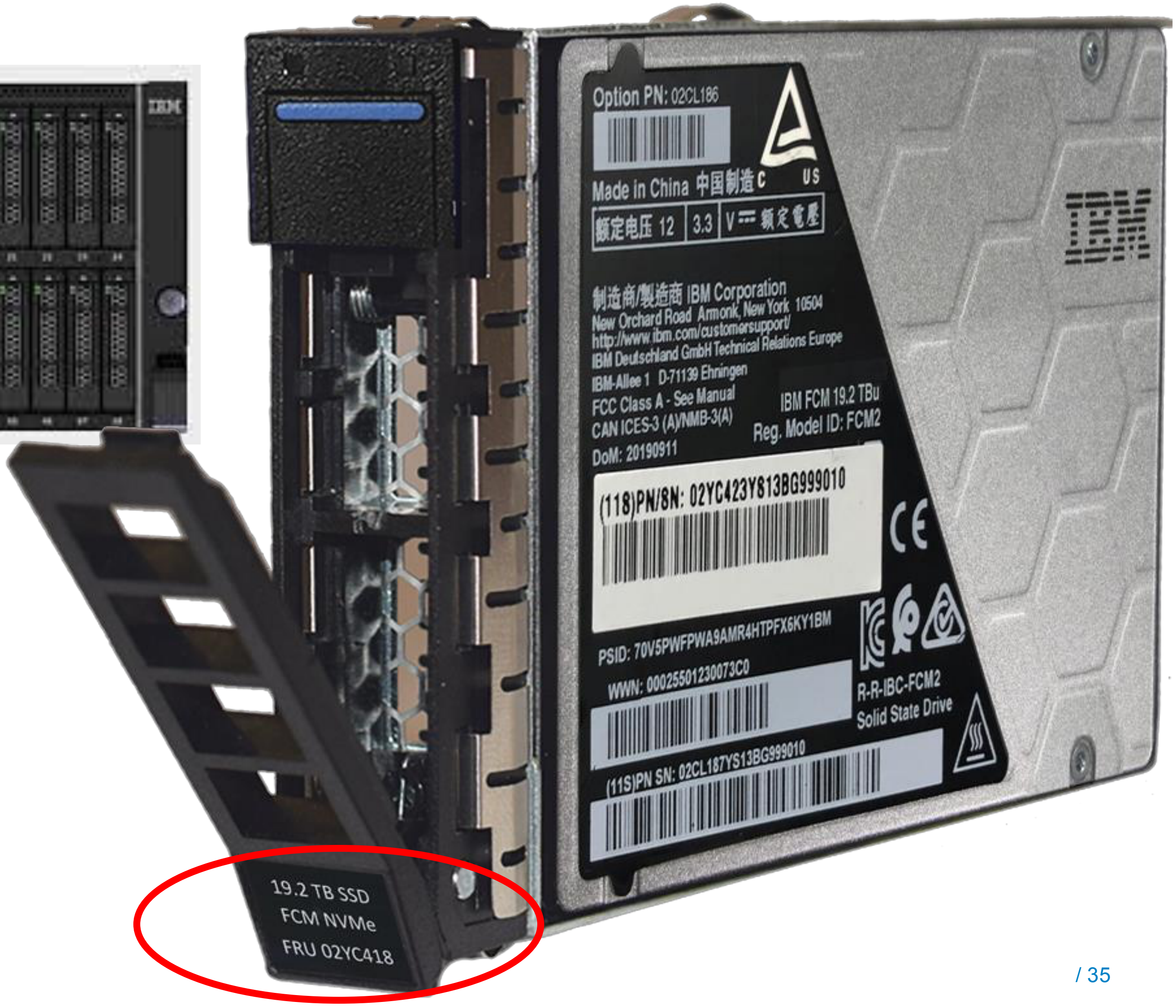
  - Create a GPFS fileset “s3sync” linked to “/mygpfs/s3sync”
  - Synchronizes fset with AWS S3 bucket “my-gpfs-bucket” subfolder “fromGPFS”
- For non-AWS S3 storage you have to fiddle around a bit to find out the correct endpoint syntax

4th use case - synch file data with S3 bucket

- Object is more difficult than just NFS or GPFS
- S3 is not a standard. E.g. NetApp StorageGrid masks some special characters differently than AWS S3 - and does not understand the AFM S3 upload for some files with special special characters
- Again many thanks to the AFM team for the help! NetApp StorageGrid works now for AFM.
- Blob and Google and Co are also different and expect a different syntax for mmafmcos*
- Synching data used by real people to an object storage is difficult because AFM COS does not allow a “mv” of a directory with content. However, for end users is rename/mv of directories essential. For object not.
- Before starting a AFM relation with an object storage:

  https://www.ibm.com/docs/en/storage-scale/5.2.2?topic=storage-afm-cloud-object-limitations

Change of subject …



SSS6k + FCM’s

- FCM = Flash Core Module
  - What does ChatGPT say about this?

  The IBM Flash Core Module (FCM) is a ~~high-~~performance storage technology designed to enhance the capabilities of IBM's storage systems, particularly in the context of flash storage… Data Reduction Technologies: The Flash Core Module incorporates advanced data reduction techniques, such as ~~deduplication and~~ compression, which help to optimize storage efficiency and reduce the overall footprint of data stored.
- In brief: QLC NVMe with compression (“… with up to 3-to-1 inline data compression”)
- FCMs are now also available in SSS6k, as 19.2TB or 38.4TB modules

SSS6k + FCM's

- In principle a good idea
- Inline compression sounds great
- for some tenders, data reduction options must be shown
- Sensational if you have compressible data (and performance is not quite so important)

SSS6k + FCM’s – some insights

- NSDs or vdisks are typically not thin provisioned and cannot change size
- FCMs provide a different usable capacity depending on the compression rate
- How is this currently implemented in an SSS6k?

```
# mmvdisk rg list --rg <RG name> --all
...
declustered   needs                             vdisks       pdisks           capacity          phy capacity
   array      service  type    BER      trim  user log  total spare rt  total raw  free raw    total      free     background task
-----------   -------  ----  -------    ----  ---- ---  ----- ----- --  --------- --------  ---------- --------  ---------------
DA1           no       NVMe  N/A        yes     32   7     24     2  2   1144 TiB  470 TiB    419 TiB   242 TiB   scrub 14d (36%)
...
```

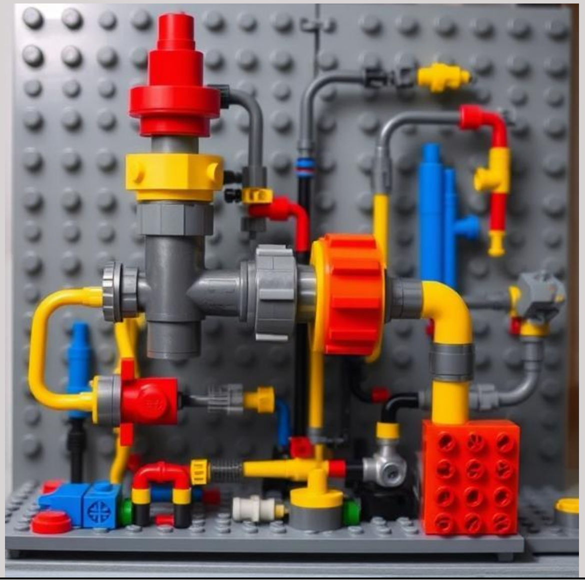
- 419 TiB physical capacity, 1144 TiB available for vdisks → compression rate 1:2,7
- “free” should remain >0

SSS6k + FCM’s – more insights

- In the end, monitoring the compression rate doesn't matter, it just has to be sufficiently “free”
- And if you want to know the compression rate:

```
# /opt/ibm/ess/hal/bin/drivedump 2
# less <dumpfile>
...
=== Compression Stats ===
CLI: ops.comp_stats

                                Total            NODE 0
Percent Full (%):               42.10             42.10
LPage Mapped (%):               17.45             17.45
Comp Ratio (w/ LPH):             1.24              1.24
Comp Ratio (w/o LPH):            1.24              1.24
Comp Ratio Stat (w/o LPH):       1.24              1.24
...
```



SSS6k + FCM’s – more insights

- Back to “free” physical capacity, there is more to know. State escalation steps:
- WARNING at 60% phys. cap. used, write throttle at 10GB/s per NSD server
- CRITICAL at 65% phys. cap. used, write throttle at 1GB/s per NSD server
- Out-Of-Space at 70% phys. cap. used, GPFS writes are rejected
- Out-Of-Space for Background Tasks at 87% phys. cap. used, stop GNR rebalance, rebuild, etc.
- What does this mean for SSS6k FCM users?
  - Try to stay far away from 60% phys. cap. used. A FCM could fail and further reduce the phys. cap. free
  - You must achieve a higher compression rate than 1:2 to get more than the real capacity out of the FCMs
    - 24 x 38TB FCM = 570TiB usable – 50% safety = 285TiB x compression rate 2 = 570TiB
    - It should also be mentioned that FCMs have a higher latency and lower throughput than NVMe’s

Change of subject … back to something cheery

Back to something cheery, AFM!

⇢ Useful monitoring commands when in trouble (on GW node):

```
# mmfsadm dump afm fset <fset name>
# echo afm_s | /usr/lpp/mmfs/bin/mmpmon
# mmhealth node show AFM -v
```

⇢ Cleanup .ptrash

```
# mmafmctl <FS> prefetch -j <fset> --directory /gpfs/afm/fset-dir/.ptrash --prefetch-threads 2 --empty-ptrash
```

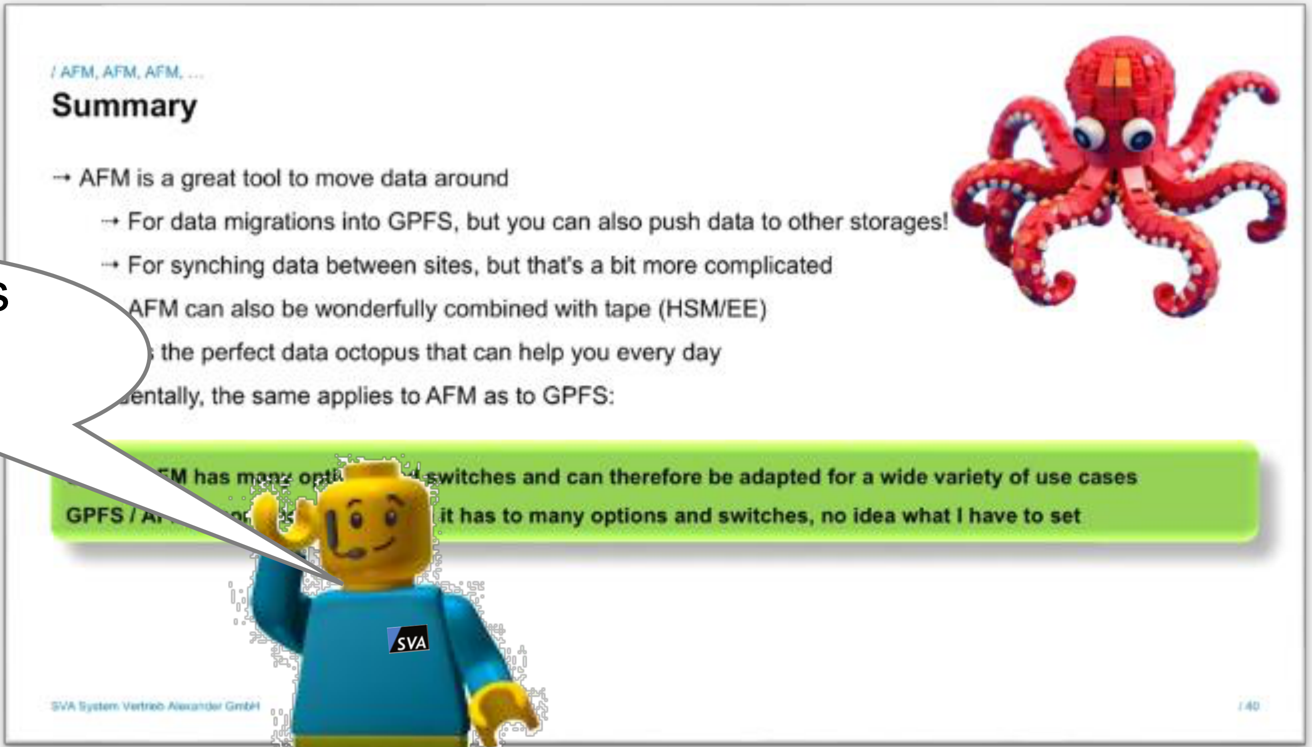
⇢ And of course /var/adm/ras/mmfs.log.latest on the gw node

Summary

- AFM is a great tool to move data around
  - For data migrations into GPFS, but you can also push data to other storages!
  - For synching data between sites, but that's a bit more complicated because of latency
  - AFM can also be wonderfully combined with tape (HSM/EE)
- AFM is the perfect data octopus that can help you every day
- Incidentally, the same applies to AFM as to GPFS:

GPFS / AFM has many options and switches and can therefore be adapted for a wide variety of use cases

GPFS / AFM is complicated because it has to many options and switches, no idea what I have to set



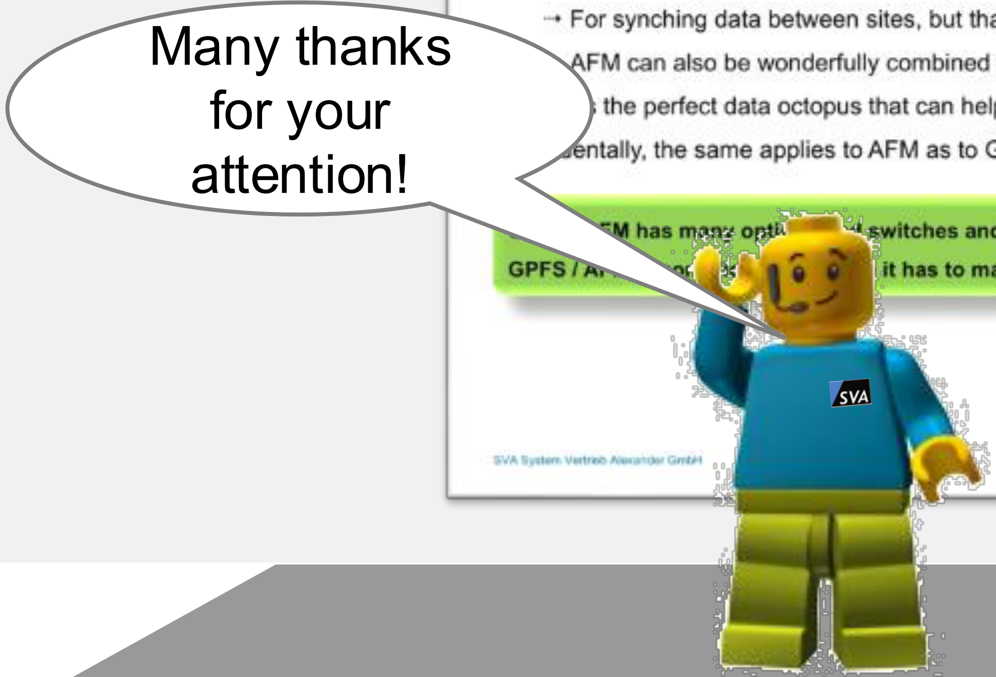
/ AFM, AFM, AFM, ...

Summary

- → AFM is a great tool to move data around
  - → For data migrations into GPFS, but you can also push data to other storages!
  - → For synching data between sites, but that's a bit more complicated
  - AFM can also be wonderfully combined with tape (HSM/EE)
- the perfect data octopus that can help you every day
- entally, the same applies to AFM as to GPFS:

FM has many opti switches and can therefore be adapted for a wide variety of use cases
GPFS / A it has to many options and switches, no idea what I have to set

SVA System Vertrieb Alexander GmbH / 40



Many thanks!

JOCHEN ZELLER
IT Architekt
Technical Leader IBM Storage Scale

Phone.: +49 151 180 256 77
Mail: jochen.zeller@sva.de