GPFS: Installation, Configuration and Administration

Vladimir Sapunenko, Ph.D.
INFN-CNAF

Vladimir.sapunenko@cnaf.infn.it

File system

- A GPFS file system is built from a collection of disks which contain the file system data and metadata.
- A file system can be built from a single disk or contain thousands of disks, storing Petabytes of data.
- A GPFS cluster can contain up to 256 mounted file systems. There is no limit placed upon the number of simultaneously opened files within a single file system.
- As an example, current GPFS customers are using single file systems up to 2PB in size and others containing tens of millions of files

GPFS Features

Main Features

- Disk scaling allowing large, single instantiation global file systems (2 PB tested)
- Node scaling (2300+ nodes) allowing large clusters and high BW (many GB/s)
- Multi-cluster architecture (i.e., grid)
- Journaling (logging) File System - logs information about operations performed on the file system meta-data as atomic transactions that can be replayed
- Data Management API (DMAPI) - Industry-standard interface allows third-party applications (e.g. TSM) to implement hierarchical storage management

Performance and scalability

- Striping across multiple disks attached to multiple nodes
- Efficient client side caching
- Support for large block size (configurable)
- Advanced algorithms for read-ahead and write-behind
- Dynamic optimization of I/O: GPFS recognizes typical access patterns like sequential, reverse sequential and random and optimizes I/O access for these patterns.

Data availability

- Fault tolerance
  - Clustering – node failure
  - Storage system failure – data replication
- File system health monitoring
  - Extensive logging and automated recovery actions in case of failure
  - appropriate recovery action is taken automatically
- Data replication available for
  - Journal logs;
  - Data
  - Metadata
- Connection retries
  - If the LAN connection to a node fails GPFS will automatically try and reestablish the connection before making the node unavailable

Installation

Very simple (2 steps)

1. Install 4 RPM packages:
    1. gpfs.base-3.2.1-1
    2. gpfs.msg.en_US-3.2.1-1
    3. gpfs.docs-3.2.1-1
    4. gpfs.gpl-3.2.1-1
2. Build Linux portability interface (see `/usr/lpp/mmfs/src/README`)
    1. cd /usr/lpp/mmfs/src
    2. make Autoconfig
    3. make World
    4. make InstallImages

Installation (comments)

- Updates are freely available from official GPFS site
- *Passwordless access needed from any to any node within cluster*
  - *Rsh or Ssh must be configured accordingly*
- *Dependencies*
  - *compat-libstdc++*
  - *xorg-x11-devel* (imake needed by Autoconfig)
- No need to repeat portability layer build on all hosts. Once compiled, copy 5 modules to all other nodes (with the same kernel and arch/hardware)

Administration

- Consistent with standard Linux file system administration
  - Simple CLI, most commands can be issued from any node in the cluster
  - No Java and graphic libraries dependency
- Extensions for clustering aspects
  - A single command can perform an action across the entire cluster
- Support for Data Management API (IBM's implementation of X/Open data storage management API)
- Rolling upgrades
  - allow to upgrade individual nodes in the cluster while the file system remains online.
- Quotas management
  - Enable control and monitor file system usage by users and groups across the cluster
- Snapshot funcion
  - Can be used to preserve the file system's contents at a single point in time
- SNMP interface
  - allow monitoring by network management applications

"mm list" commands

GPFS provides a number of commands to list parameter settings, configuration components and other things.

COMMENT:

By default, nearly all of the mm commands require root authority to execute.

However, many sysadm's reset the permissions on mmls commands to allow programmers and others to execute them as they are very useful for the purposes of problem determination and debugging.

Selected "mmls" Commands

- **mmlsfs <device name>**
  - Without specifying any options, it lists all file system attributes

```
[root@gpfs-01-01 ~]# mmlsfs gpfs
flag value            description
---- ---------------- -----------------------------------------------------
 -f  2048             Minimum fragment size in bytes
 -i  512              Inode size in bytes
 -I  8192             Indirect block size in bytes
 -m  1                Default number of metadata replicas
 -M  2                Maximum number of metadata replicas
 -r  1                Default number of data replicas
 -R  2                Maximum number of data replicas
 -j  cluster          Block allocation type
 -D  nfs4             File locking semantics in effect
 -k  all              ACL semantics in effect
 -a  1048576          Estimated average file size
 -n  32               Estimated number of nodes that will mount file system
 -B  65536            Block size
 -Q  none             Quotas enforced
     none             Default quotas enabled
 -F  57600            Maximum number of inodes
 -V  10.00 (3.2.0.0)  File system version
 -u  yes              Support for large LUNs?
 -z  no               Is DMAPI enabled?
 -L  2097152          Logfile size
 -E  no               Exact mtime mount option
 -S  no               Suppress atime mount option
 -K  whenpossible     Strict replica allocation option
 -P  system           Disk storage pools in file system
 -d  disk_hdb_gpfs_01_01;disk_hdb_gpfs_01_02;disk_hdb_gpfs_01_03  Disks in file system
 -A  yes              Automatic mount option
 -o  none             Additional mount options
 -T  /gpfs            Default mount point
```

Selected "mmls" Commands

- **mmlsconfig**
  - Without specifying any options, it lists all current nodeset configuration info

```
[root@gpfs-01-01 ~]# mmlsconfig
Configuration data for cluster gpfs-01-01.cr.cnaf.infn.it:
----------------------------------------------------------
clusterName gpfs-01-01.cr.cnaf.infn.it
clusterId 9483033361199735958
clusterType lc
autoload yes
minReleaseLevel 3.2.0.2
dmapiFileHandleSize 32
pagepool 256M
dmapiWorkerThreads 24

File systems in cluster gpfs-01-01.cr.cnaf.infn.it:
---------------------------------------------------
/dev/gpfs
```

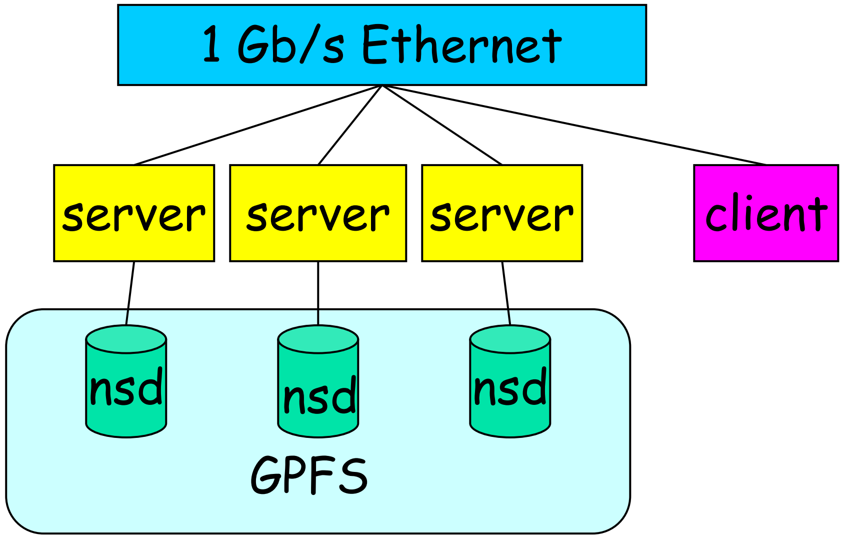
Other Selected "mmls" Commands

- **mmlsattr <file name>**
  - query file attributes
- **mmlscluster**
  - display current configuration information for a GPFS cluster
- **mmlsdisk <device> [-d "disk names list"]**
  - display current configuration and state of the disks in a file system
- **mmlsmgr**
  - display which node is the file system manager for the specified file systems
- **mmlsnsd**
  - display current NSD information in the GPFS cluster
- NOTES
  - See documentation for other parameters and options.

Testbed

Almost all further examples are referred to a simple 4-node cluster used at CNAF for testing purposes:

- 4 dual cpu Xeon@2.2GHz
  - gpfs-01-01 (I/O server)
  - gpfs-01-02 (I/O server)
  - gpfs-01-03 (I/O server)
  - TSM-TEST-1 (client)
- NSD: internal IDE hdd 20GB (hdb)
  - disk_hdb_gpfs_01_01
  - disk_hdb_gpfs_01_02
  - disk_hdb_gpfs_01_03
- Interconnect: 1Gb ethernet



Selected "mm" Commands

GPFS provides a number of commands needed to create the file system.

These commands of necessity require root authority to execute.

- **mmcrcluster** - Creates a GPFS cluster from a set of nodes.

```
>mmcrcluster -n gpfs.nodelist \
            -p gpfs-01-01.cr.cnaf.infn.it \
            -s gpfs-01-02.cr.cnaf.infn.it \
            -r /usr/bin/ssh \
            -R /usr/bin/scp \
            -C test.cr.cnaf.infn.it \
            -U cr.cnaf.infn.it
>
>cat gpfs.nodelist
gpfs-01-01:quorum-manager
gpfs-01-01:quorum-manager
gpfs-01-01:quorum
```

Selected "mm" Commands

- **mmstartup and mmshutdown**
  - startup and shutdown the **mmfsd** daemons
  - if necessary, mount file system after running **mmstartup**
    - properly configured, mmfsd will startup automatically (n.b., no need to run **mmstartup**); if it can not start for some reason, you will see **runmmfs** running and a lot of messages in /var/adm/ras/mmfs.log.latest
  - mmgetstate - displays the state of the GPFS daemon on one or more nodes:

```
[root@gpfs-01-01 ~]# mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      gpfs-01-01       active
       2      gpfs-01-02       active
       3      gpfs-01-03       active
       4      TSM-TEST-1       active
```

Selected "mm" Commands

- **mmcrnsd**

Creates and globally names Network Shared Disks for use by GPFS.
mmfsd daemon must be running to execute mmcrnsd (i.e., do mmstartup first)
> mmcrnsd -F disk.lst

- disk.lst is a "disk descriptor file whose entries are in the format
  - DiskName:ServerList::DiskUsage:FailureGroup:DesiredName:StoragePool
    - **DiskName**: The disk name as it appears in /dev
    - **ServerList**: Is a comma separated list of NSD server nodes.
      - Up to eight NSD servers in this list.
      - preferentially use the first server on the list. If the first server is not available, the NSD will use the next available server on the list
    - **DiskUsage**: dataAndMetadata (default) or dataOnly or metadataOnly
    - **FailureGroup**: GPFS uses this information during data and metadata placement to assure that no two replicas of the same block are written in such a way as to become unavailable due to a single failure. All disks that are attached to the same adapter or NSD server should be placed in the same failure group.
    - **DesiredName**: Specify the name you desire for the NSD to be created. Default format... **gpfs**<integer>**nsd**
- dsk.lst is modified for use as the input file to the mmcrfs command

Selected "mm" Commands

Disk Descriptor Files

```
> cat disk.lst
/dev/hdb:gpfs-01-01.cr.cnaf.infn.it::::disk_hdb_gpfs_01_01
/dev/hdb:gpfs-01-02.cr.cnaf.infn.it::::disk_hdb_gpfs_01_02
/dev/hdb:gpfs-01-03.cr.cnaf.infn.it::::disk_hdb_gpfs_01_03
> mmcrnsd –F disk.lst
…
> cat disk.list
# /dev/hdb:gpfs-01-01.cr.cnaf.infn.it::::disk_hdb_gpfs_01_01
disk_hdb_gpfs_01_01:::dataAndMetadata:4001
# /dev/hdb:gpfs-01-02.cr.cnaf.infn.it::::disk_hdb_gpfs_01_02
disk_hdb_gpfs_01_02:::dataAndMetadata:4002
# /dev/hdb:gpfs-01-03.cr.cnaf.infn.it::::disk_hdb_gpfs_01_03
disk_hdb_gpfs_01_03:::dataAndMetadata:4003
```

- NOTES
  - This is the results from a single node with internal (IDE) disks
  - Using disk descriptor defaults.
  - The integer in nsd disk names is based on a counter. If you delete and re-create the file system, the counter is not generally re-initialized

Selected "mm" Commands

- **mmcrfs <mountpoint> <device name> <options>**
  - Create a GPFS file system
    - -F specifies a file containing a list of disk descriptors (one per line)
      - this is the output file from **mmcrnsd**
      - -A do we mount file system when starting mmfsd (default = yes)
      - -B block size (16K, 64K, 128K, 256K, 512K, 1024K,2M,4M)
      - -E specifies whether or not to report exact mtime values
      - -m default number of copies (1 or 2) of i-nodes and indirect blocks for a file
      - -M default max number of copies of inodes, directories, indirect blocks for a file
      - -n estimated number of nodes that will mount the file system
      - -N max number of files in the file system (default = sizeof(file system)/1M
      - -Q activate quotas when the file system is mounted (default = NO)
      - -r default number of copies of each data block for a file
      - -R default maximum number of copies of data blocks for a file
      - -S suppress the periodic updating of the value of atime
      - -v verify that specified disks do not belong to an existing file system
      - -z enable or disable DMAPI on the file system (default = no)
- **Typical example**
  - mmcrfs /gpfs gpfs -F disk.lst -A yes -B 1024k -v no

"mm change" commands.

GPFS provides a number of commands to change configuration and file system parameters after being initially set.

There are some GPFS parameters which are initially set only by default; the only way to modify their value is using the appropriate mmch command.

N.B., There are restrictions regarding changes that can be made to many of these parameters; be sure to consult the *Concepts, Planning and Installation Guide* for tables outlining what parameters can be changed and under which conditions they can be changed. See the *Administration and Programming Reference manual* for further paramter details.

Selected "mmch" Commands

- **mmchconfig**
  - change GPFS configuration attributes originally set (explicitly or implicitly) by mmconfig
    - relative to **mmconfig**, parameter IDs may be different
- `mmchconfig Attribute=value[,Attribute=value...]  [-i | -I] [-N {Node[,Node...] NodeFile | NodeClass}]`
  - parameters and options
    - **-N** list of node names (default is all nodes in the cluster) can not be used for all options
    - ***autoload*** (same as -a)
    - ***dataStructureDump*** (same as -D)
    - ***maxblocksize*** Changes the maximum file system block size.
    - ***maxMBpS*** (data rate estimate (MB/s) on how much data can be transferred in or out of 1 node) The value is used in calculating the amount of IO that can be done to effectively prefetch data for readers and write-behind data from writers. By lowering this value, you can artificially limit how much IO one node can put on all of the disk servers. This is useful in environments in which a large number of nodes can overrun a few virtual shared disk servers. The default is 150 MB/s which can severally limity performance on HPS ("federation") based systems.
    - ***maxFilesToCache*** (same as -M)
    - ***maxStatCache*** (specifies number of i-nodes to keep in statcache)
    - ***pagepool*** (same as -p)
  - following options apply only to ***dataStructureDump***, ***maxblocksize***, ***pagepool***
    - **-i** changes are immediate and permanent
    - **-I** changes are immediate, but do not persist after **mmfsd** daemon is restarted

Managing disks

mmdf - Queries available file space on a GPFS file system.

```
[root@TSM-TEST-1 ~]# mmdf gpfs
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 61 GB)
disk_hdb_gpfs_01_01      19551168     1001 yes      yes        16899648 ( 86%)          2848 ( 0%)
disk_hdb_gpfs_01_02      19551168     1002 yes      yes        16903424 ( 86%)          2848 ( 0%)
disk_hdb_gpfs_01_03      19551168     1004 yes      yes        16903424 ( 86%)          2848 ( 0%)
                ------------- -------------------- -------------------
(pool total)         39102336                                33803072 ( 86%)          5474 ( 0%)

                =============                         ==================== ===================
(total)              39102336                                33803072 ( 86%)          5474 ( 0%)

Inode Information
-----------------
Number of used inodes:            4049
Number of free inodes:           53551
Number of allocated inodes:      57600
Maximum number of inodes:        57600
```

Deleting a disk

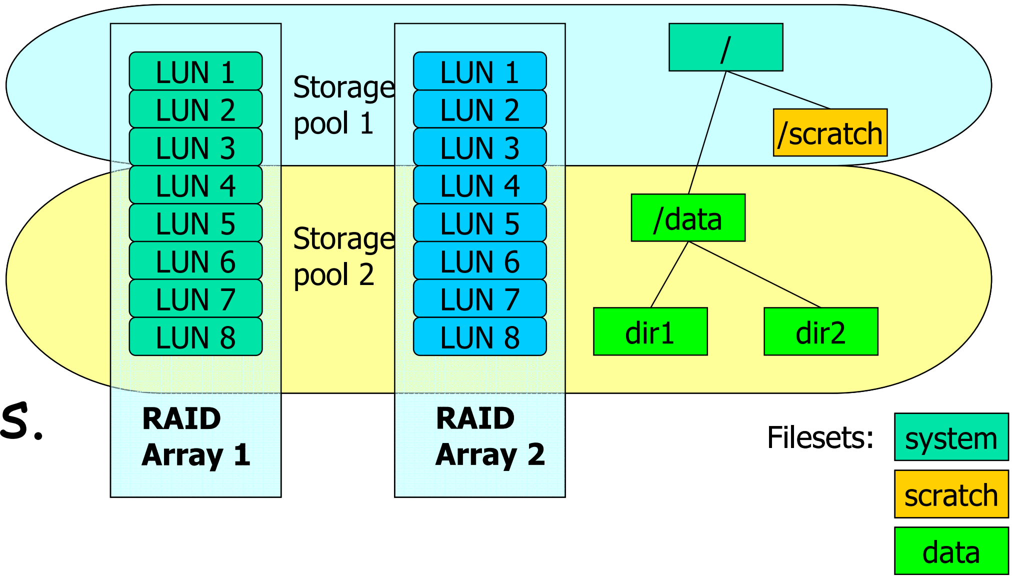
```
[root@gpfs-01-01 ~]# mmdeldisk gpfs "disk_hdb_gpfs_01_03"
Deleting disks ...
Scanning system storage pool
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 100 %  complete on Wed Jun  4 17:22:35 2008
Scan completed successfully.
Checking Allocation Map for storage pool 'system'
tsdeldisk completed.
mmdeldisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root@gpfs-01-01 ~]# mmdf gpfs
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 61 GB)
disk_hdb_gpfs_01_01      19551168     1001 yes      yes        15919616 ( 81%)          3104 ( 0%)
disk_hdb_gpfs_01_02      19551168     1002 yes      yes        15919936 ( 81%)          2994 ( 0%)
                -------------                         -------------------- -------------------
(pool total)         39102336                             31839552 ( 81%)          6098 ( 0%)

                =============                         ==================== ===================
(total)              39102336                             31839552 ( 81%)          6098 ( 0%)

Inode Information
-----------------
Number of used inodes:            4039
Number of free inodes:           53561
Number of allocated inodes:      57600
Maximum number of inodes:        57600
```

Filesets and Storage pools

- New feature of GPFS v3.1
- Storage pools allow the creation of disk groups within a file system (hardware partitioning)
- Filesets is a sub-tree of the file system namespace (Namespace partitioning). For example, it can be used as administrative boundaries to set quotas.



Adding a disk (and a storage pool)

```
[root@gpfs-01-03 ~]# mmadddisk gpfs "disk_hdb_gpfs_01_03:::dataOnly:::data"

The following disks of gpfs will be formatted on node gpfs-01-01.cr.cnaf.infn.it:
    disk_hdb_gpfs_01_03: size 19551168 KB
Extending Allocation Map
Creating Allocation Map for storage pool 'data'
Flushing Allocation Map for storage pool 'data'
Disks up to size 52 GB can be added to storage pool 'data'.
Checking Allocation Map for storage pool 'data'
Completed adding disks to file system gpfs.
mmadddisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root@gpfs-01-03 ~]# mmdf gpfs
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 61 GB)
disk_hdb_gpfs_01_01      19551168     1001 yes      yes        12713728 ( 65%)          4728 ( 0%)
disk_hdb_gpfs_01_02      19551168     1002 yes      yes        12713920 ( 65%)          4562 ( 0%)
                ------------- -------------------- -------------------
(pool total)         39102336                              25427648 ( 65%)          9290 ( 0%)

Disks in storage pool: data (Maximum disk size allowed is 52 GB)
disk_hdb_gpfs_01_03      19551168     4003 no       yes        19549056 (100%)            62 ( 0%)
                -------------                         -------------------- -------------------
(pool total)         19551168                              19549056 (100%)            62 ( 0%)

                =============                         ==================== ===================
(data)               58653504                              44976704 ( 77%)          9352 ( 0%)
(metadata)           39102336                              25427648 ( 65%)          9290 ( 0%)
                =============                         ==================== ===================
(total)              58653504                              44976704 ( 77%)          9352 ( 0%)

Inode Information
-----------------
Number of used inodes:            4040
Number of free inodes:           53560
Number of allocated inodes:      57600
Maximum number of inodes:        57600
```

Initial Placement policy

- Two storage pools:
  - System – data and metadata
  - Data – data only
- Placement policy example
  - use pool “data” until 99% full, then use pool “system”

```
RULE ‘rule1' SET POOL 'data' LIMIT (99)
RULE ‘default' SET POOL 'system'
```

  - Place all files with UID>2048 in pool “data”,and all others in “system”:

```
RULE 'rule1' SET POOL 'data' WHERE USER_ID>2048
RULE 'default' SET POOL 'system‘
```

  - Install placement policy

```
mmchpolicy Device PolicyFilename –I yes
```

User defined polices

- File placement policies
  - Define where the data will be created (appropriate storage pool)
  - Rules are determined by attributes like
    - File name
    - User name
    - Fileset
- File management policies
  - Possibility to move data from one pool to another without changing file location in the directory structure
  - Change replication status
  - Prune file system (deleting files as defined by policy)
  - Determined by attributes like
    - Access time
    - Path name
    - Size of the file

Policy rules examples

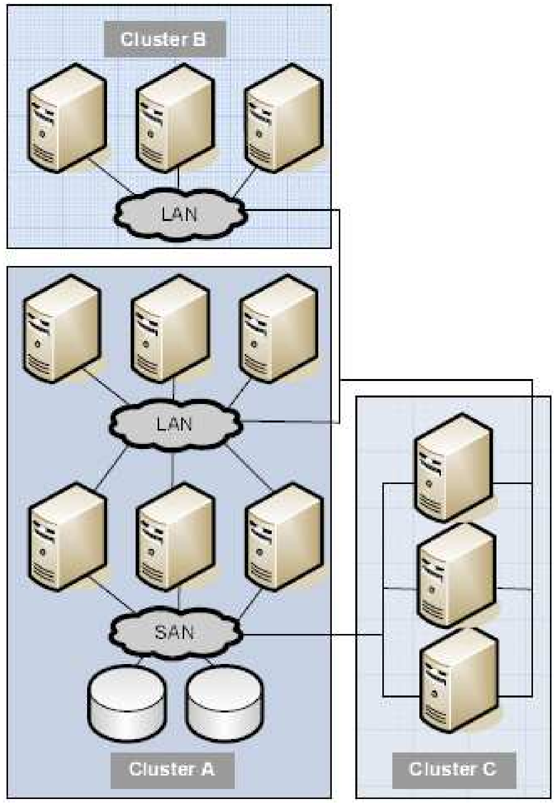
- If the storage pool named pool_1 has an occupancy percentage above 90% now, bring the occupancy percentage of pool_1 down to 70% by migrating the largest files to storage pool pool_2:

```
RULE 'mig1' MIGRATE FROM POOL 'pool_1'
   THRESHOLD(90,70) WEIGHT(KB_ALLOCATED) TO POOL
   'pool_2'
```

- Delete files from the storage pool named pool_1 that have not been accessed in the last 30 days, and are named like temporary files or appear in any directory that is named tmp:

```
 RULE 'del1' DELETE FROM POOL 'pool_1' WHERE
   (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) >
   30) AND (lower(NAME) LIKE '%.tmp' OR PATH_NAME
   LIKE '%/tmp/%')
```

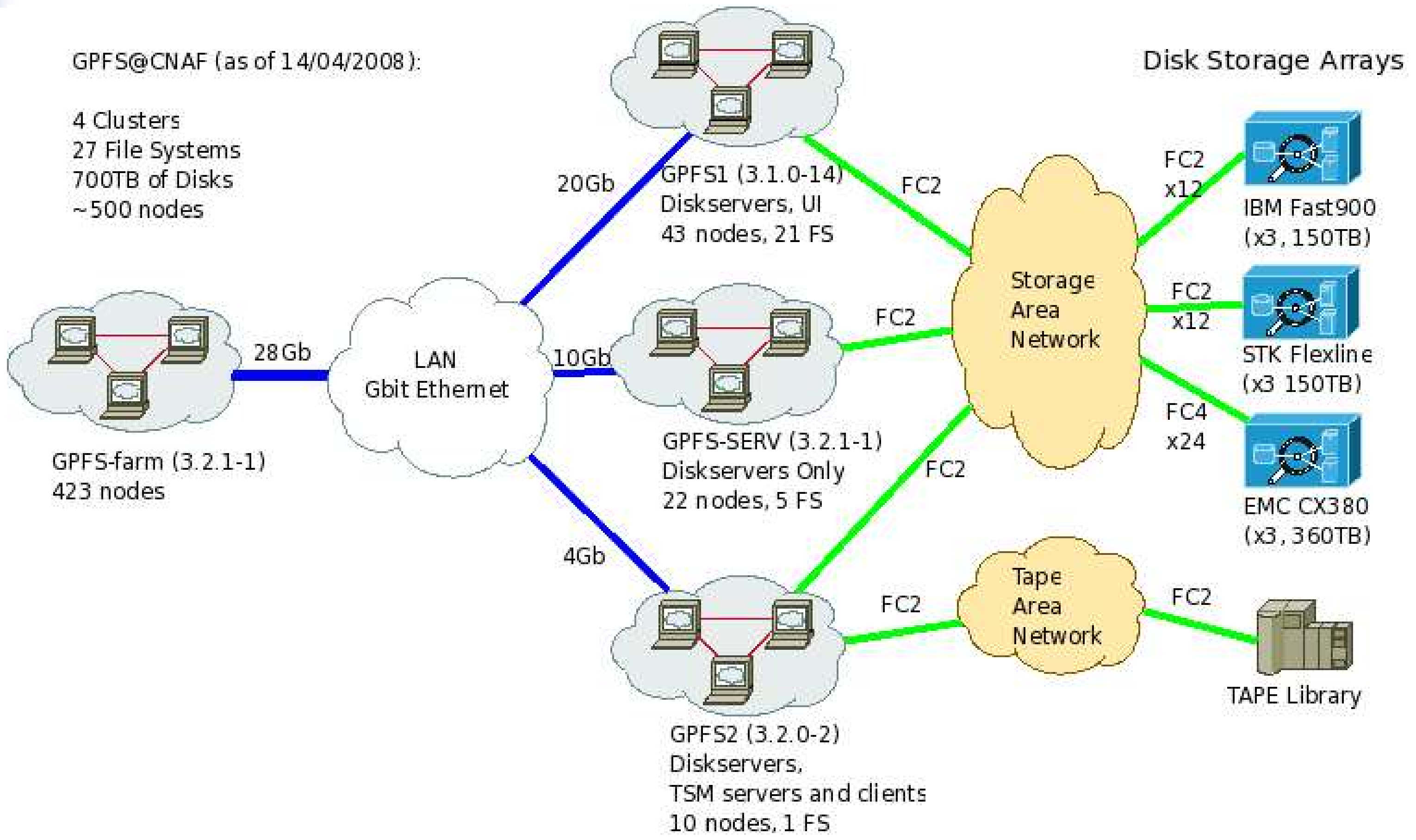
Sharing data between clusters



- GPFS allows to share data across clusters
- Permit access to specific file systems from another GPFS cluster
- higher performance levels than file sharing technologies like NFS or Samba
- requires a trusted kernel at both the owning and sharing clusters
- both LAN and SAN can be used as cluster interconnect

multi-cluster configuration with both LAN and mixed LAN and SAN connections

Cross-cluster file system access



Cross-cluster file system access (requirements)

- OpenSSL must be installed on all nodes in the involved clusters
  - See the GPFS Frequently Asked Questions at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html for current OpenSSL version requirements and for information on the supported cipher suites.
- The procedure to set up remote file system access involves the generation and exchange of authorization keys between the two clusters.
  - administrator of the GPFS cluster that owns the file system needs to authorize the remote clusters that are to access it
  - administrator of the GPFS cluster that seeks access to a remote file system needs to define to GPFS the remote cluster and file system whose access is desired

In this example, cluster1 is the name of the cluster that owns and serves the file system to be mounted, and cluster2 is the name of the cluster that desires to access the file system.

Cross-cluster file system access

1. On **cluster1**, the system administrator generates a public/private key pair. (The key pair is placed in */var/mmfs/ssl)*:
   ```
   mmauth genkey new
   ```
2. On cluster1, the system administrator enables authorization by issuing:
   ```
   mmauth update . -l AUTHONLY
   ```
   1. This should be done when GPFS is stopped on all nodes
3. The system administrator of **cluster1** now gives the file ***/var/mmfs/ssl/id_rsa.pub*** to the system administrator of **cluster2**, who desires to access the **cluster1** file systems. This operation must occur outside of the GPFS command environment.
4. On **cluster2**, the system administrator generates a public/private key pair.
   ```
   mmauth genkey new
   ```
5. On **cluster2**, the system administrator enables authorization by issuing:
   ```
   mmauth update . -l AUTHONLY
   ```
   1. This should be done when GPFS is stopped on all nodes
6. The system administrator of **cluster2** gives file ***/var/mmfs/ssl/id_rsa.pub*** to the system administrator of **cluster1**.
   1. This operation must occur outside of the GPFS command environment.

Cross-cluster file system access (cont.)

7. On cluster1, the system administrator authorizes cluster2 to mount file systems owned by cluster1 utilizing the key file received from the administrator of cluster2:
   ```
   mmauth add cluster2 -k cluster2_id_rsa.pub
   ```
   where:
   cluster2
   Is the real name of cluster2 as given by the **mmlscluster** command in cluster2.
   cluster2_id_rsa.pub
   Is the name of the file obtained from the administrator of cluster2 in Step 6.
8. On cluster1, the system administrator authorizes cluster2 to mount specific file systems owned by cluster1:
   ```
   mmauth grant cluster2 -f /dev/gpfs
   ```
9. On cluster2, the system administrator now must define the cluster name, contact nodes and public key for cluster1:
   ```
   mmremotecluster add cluster1 -n node1,node2,node3 -k \
                                 cluster1_id_rsa.pub
   ```
   where:
   cluster1
   Is the real name of cluster1 as given by the mmlscluster command.
   node1, node2, and node3
   Are nodes in cluster1. The hostname or IP address must refer to the communications adapter that is used by GPFS as given by the `mmlscluster`.
   cluster1_id_rsa.pub
   Is the name of the file obtained from the administrator of cluster1 in Step 3.
   This permits the cluster desiring to mount the file system a means to locate the serving cluster and ultimately mount its file systems.

Cross-cluster file system access (cont.)

10. On cluster2, the system administrator issues one or more mmremotefs commands to identify the file systems in cluster1 that are to be accessed by nodes in cluster2: mmremotefs add
    `/dev/mygpfs -f /dev/gpfs -C cluster1 -T /mygpfs`
    where:
    /dev/mygpfs
        Is the device name under which the file system will be known in cluster2.
    /dev/gpfs
        Is the actual device name for the file system in cluster1.
    cluster1
        Is the real name of cluster1 as given by the mmlscluster command on a node in cluster1.
    /mygpfs
        Is the local mount point in cluster2.
11. Mount the file system on cluster2, with the command:
    `mmmount /dev/mygpfs`

Cross-cluster file system access (summary)

Commands that the administrators of the two clusters need to issue so that the nodes in **cluster2** can mount the remote file system **fs1**, owned by **cluster1**, assigning **rfs1** as the local name with a mount point of **/rfs1**.

Cluster1

```
mmauth genkey new
mmshudown -a
mmauth update . -l AUTHONLY
mmstartup -a
```

Cluster2

```
mmauth genkey new
mmshutdown -a
mmauth update . -l AUTHONLY
mmstartup -a
```

Exchange public keys (file */var/mmfs/ssl/id_rsa.pub***)**

```
mmauth add cluster2 ...
mmauth grant cluster2 -f fs1 ...
```

```
mmremotecluster add cluster1 ...
mmremotefs add rfs1 -f fs1 \
        -C cluster1 -T /rfs1
```

Acknowledgments

Materials used in this presentation, along with presenter's own experience, have been mainly obtained from the following sources:

- **"GPFS Programming, Configuration and Performance Perspectives"** by Raymond L. Paden, Deep Computing, IBM, 2005
- **"An Introduction to GPFS Version 3.2"** by Scott Fadden, IBM Corporation, 2007
- IBM Cluster Information Center Website: http://publib.boulder.ibm.com

.