# GPFS – INTRODUCTION AND SETUP

Jaqui Lynch

lynchj@forsythe.com

Handout at:

http://www.circle4.com/forsythe/gpfs-introandsetup.pdf

Also: http://www.ibmsystemsmag.com/aix/administrator/performance/Breaking-the-Bottleneck/



FORSYTHE

# Agenda:

- GPFS  Terminology
- GPFS Introduction
- Installing a 3 node cluster

# GPFS: General Parallel File System

- Available since 1991 (AIX), on Linux since 2001
- Product available on POWER and xSeries (IA32, IA64, Opteron) on AIX, Linux, Windows & BlueGene.
- Also runs on compatible non-IBM servers and storage.
- Thousands of installs, including many Top 500 supercomputers
- Concurrent shared disk access to a single global namespace.
- Customers use GPFS in many applications
  - High-performance computing
  - Scalable file and Web servers
  - Database and digital libraries
  - Digital media
  - Oracle data
  - Analytics, financial data management, engineering design, …

# What is Parallel I/O?

- A cluster of machines access shared storage with single file system name space with POSIX semantics.
- Caching and striping across all disks with distributed locking
- The robust system design can recover from node failure using metadata logging.
- The solution scales to thousands of nodes with a variety of I/O.

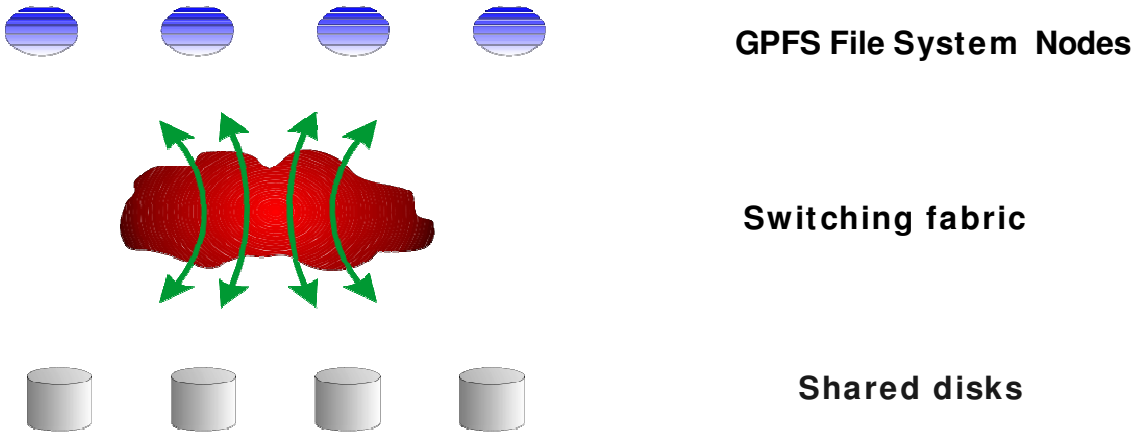



Diagram courtesy of IBM

# GPFS TERMINOLOGY 1/2

***Cluster*** - This consists of a number of nodes and network shared disks (NSDs) for management purposes.

***Storage Pool*** - This groups a file system's storage and allows a user to partition storage based on characteristics such as performance, locality and reliability.

***Node*** - This is an individual OS instance within a cluster.

***Nodeset*** - This is a group of nodes that are all running the same level of GPFS and working on the same file systems. A cluster can contain more than one nodeset, but a node can only belong to one nodeset. A nodeset can contain up to 32 file systems.

***Configuration manager*** - This has overall responsibility for correct operation of all the nodes and the cluster as a whole. It selects the file-system manager node for each file-system, determines succession if a file-system manager node fails and also monitors quorum. By default, quorum is set to 50% +1.

***File-system manager*** - Also referred to as the "stripe group manager," there can be only one at a time. This node maintains the availability information for the disks in the file system. In a large cluster, this may need to be a dedicated node that's separate from the disk servers.

# GPFS TERMINOLOGY 2/2

***Stripe group*** - This is basically a collection of disks a file system gets mounted on.

***Token manager*** - This handles tokens for the file handles and synchronizes concurrent access to files, ensuring consistency among caches. The token-manager server also synchronizes certain pieces of GPFS metadata and some of the internal data structures. The token-manager server usually resides on the file-system manager and may need significant CPU power.

***Metanode*** - A node that handles metadata, also referred to as "directory block updates."

***Application node*** - This mounts a GPFS file system and runs a user application that accesses the file system.

***Network shared disk (NSD)*** - This component is used for global device naming and data access in a cluster. NSDs are created on local disk drives. The NSD component performs automatic discovery at each node to see if any physical disks are attached. If there are no disks, an NSD must be defined with a primary server. Best practices dictate that a secondary server should also be defined. I/O is then performed using the network connection to get to the NSD server that performs the I/O on behalf of the requesting node. NSDs can also be used to provide backup in case a physical connection to disk fails.

# GPFS FUNCTIONS AND FEATURES

## FUNCTIONS

- Performance
- Scaling to thousands of nodes
- Petabytes of storage supported
- Parallel data and metadata from server nodes and disks
- High Availability + Disaster Recovery
- Multi-platform + Interoperability: Linux, AIX and NFS/CIFS support
- Multi-cluster/WAN AFM function
- Online system management
- Cache management, Quotas, Snapshots
- Information Lifecycle Management
- API, Performance tuning
- Fault tolerance & Disaster recovery
- Multi-cluster support

## FEATURES

- Quorum management
- High availability with independent paths
- Striping using blocks (supports sub-blocks)
- Byte/block range locking (rather than file or extent locking)
- GPFS pagepool
- Access pattern optimization
- Distributed management (e.g. metadata + tokens)
- File system journaling with POSIX semantics
- Quotas
- Integrated Life Management: pools, filesets, policies
- Data replication, snapshots, clones

# GPFS SCALING CAPABILITIES - TESTED

- Linux on x86 Nodes: 9300
- AIX Nodes: 1530
- Windows on x86 Nodes: 64
- Linux on x86 & AIX combo: 3906 – 3794 Linux & 112 AIX

Contact gpfs@us.ibm.com if you intend to exceed:

Configurations with Linux nodes exceeding 512 nodes
Configurations with AIX nodes exceeding 128 nodes
Configurations with Windows nodes exceeding 64 nodes
FPO-enabled configurations exceeding 64 nodes

- Filesystem Size: 2PB+ ($2^{99}$ bytes) if >GPFS 2.3
- Number of mounted filesystems in a cluster is 256
- Number of files in a filesystem if created by v2.3 or later 2,147,483,648
- For GPFS 3.5 architectural limit is $2^{64}$ and tested is 9,000,000,000

For additional scaling limits check the FAQ at:
http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=%2Fcom.ibm.cluster.gpfs.doc%2Fgpfs_faqs%2Fgpfsclustersfaq.html

# GPFS SCALING CAPABILITIES - SEEN

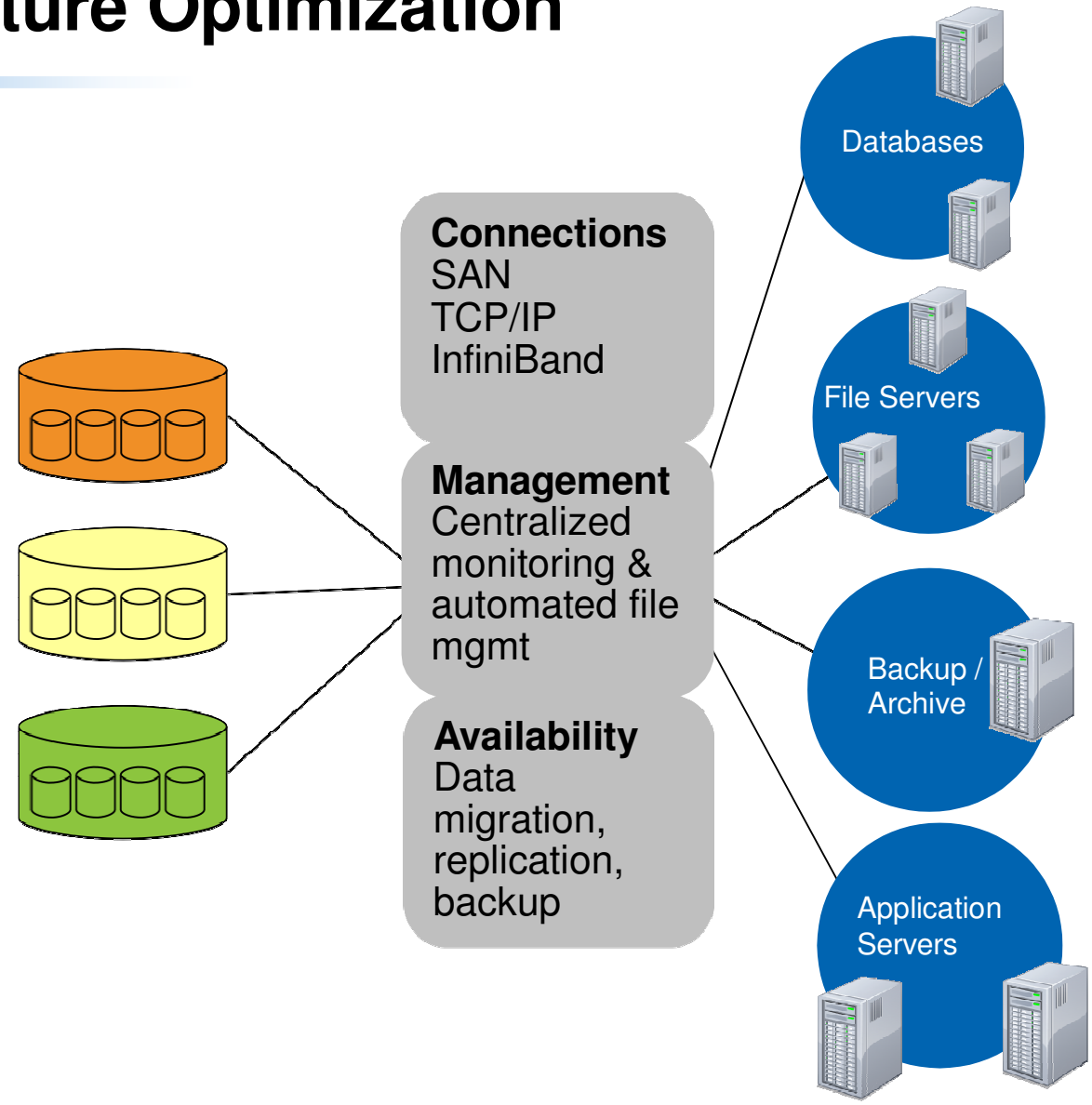
- Nodes: 2000+ (8K)
- LUNS: 2000+
- Filesystem Size: 2PB+ ($2^{99}$ bytes)
- Mounted filesystems: 256
- Lun size: > 2TB (64bit)
- Number of Files in FS: 1 Billion+ - $2^{64}$ files per filesystem
- Maximum file size equals file system size
- Production file systems 4PB
- Tiered storage: solid state drives, SATA/SAS drives, tape

- Disk I/O to AIX @ 134 GB/sec
- Disk I/O to Linux @ 66 GB/sec

# File Data Infrastructure Optimization

IBM GPFS is designed to enable:

- A single global namespace across platforms.
- High performance common storage.
- Eliminating copies of data.
- Improved storage use.
- Simplified file management.

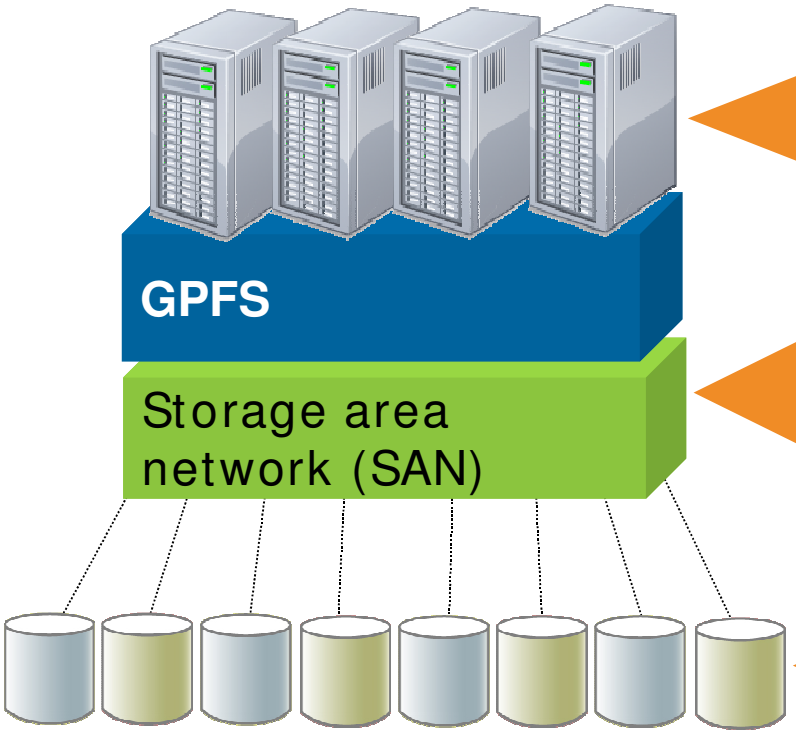



Slide courtesy of IBM



FORSYTHE

# BASIC GPFS CLUSTER WITH SHARED SAN




Each node has “direct” speeds

Can be dedicated SAN, vSCSI, NPIV

Simultaneous LUN access

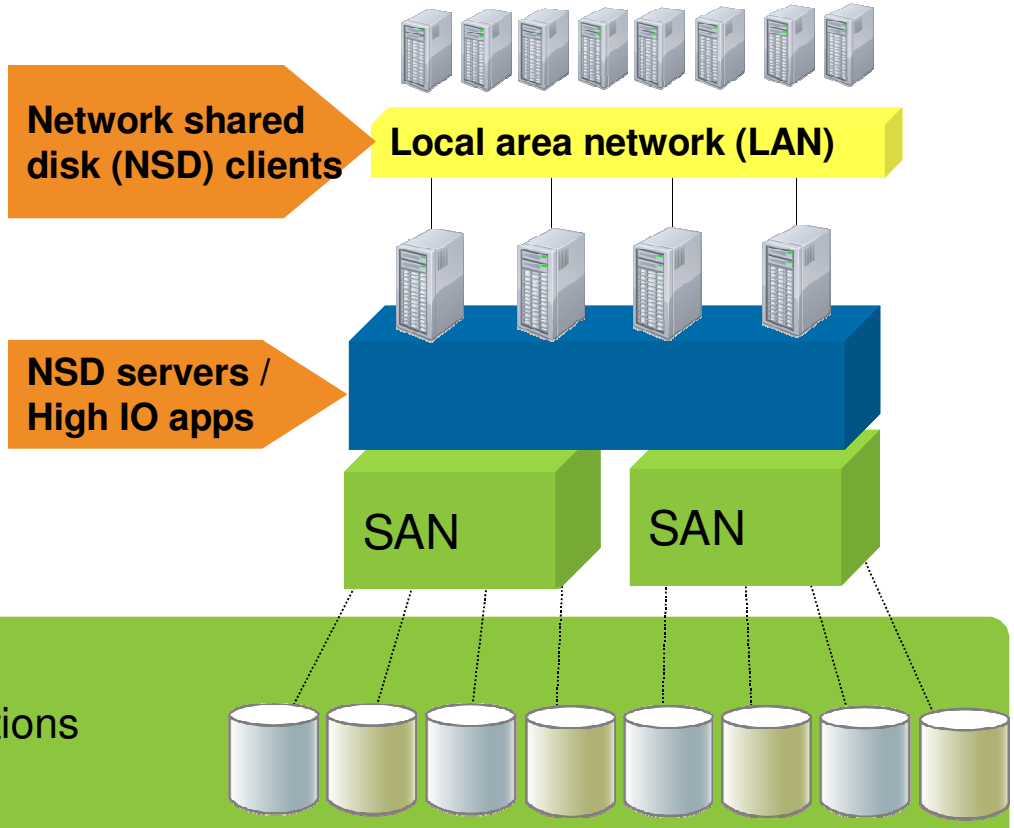
All features are included. All software features: snapshots, replication and multisite connectivity are included in the GPFS license. With no license keys except for client and server to add on, you get all of the features up front.

Slide courtesy of IBM

# NETWORK-BASED BLOCK I/O

Application data access on network attached nodes is exactly the same as a storage area network (SAN) attached node. General Parallel File System (GPFS™) transparently sends the block-level I/O request over a TCP/IP network.

Any node can add direct attach for greater throughput.




**Why?**

Enable virtually seamless multi-site operations

Reduce costs for data administration

Provide flexibility of file system access

Establish highly scalable and reliable data storage

Future protection by supporting mixed technologies

Slide courtesy of IBM

# OPERATING SYSTEMS AND SOFTWARE

**Servers:**

- IBM AIX
  - GPFS cannot run in a WPAR but can be shared with a WPAR as a named FS
- Linux
  - RHEL
  - SLES
  - VMWARE ESX Server
- Windows 2008 Server

**Some examples of Software:**

IBM DB2
Oracle
SAP
SAS
Ab Initio
Informatica
SAF

# Supported Environments

| GPFS Version | 3.3 | 3.4 | 3.5 |
|---|---|---|---|
| **AIX 5.3** | Y | Y | N |
| **AIX 6.1** | Y | Y | Y |
| **AIX 7.1** | Y | Y | Y |
| **Linux on Power** | RHEL4,5,6<br>SLES9,10,11 | RHEL4,5,6<br>SLES10,11 | RHEL5,6<br>SLES10,11 |
| **Linux on x86** | RHEL4,5,6<br>SLES9,10,11 | RHEL4,5,6<br>SLES10,11 | RHEL5,6<br>SLES10,11 |
| **Windows Server 2008 x64 (SP2)** | Y | Y | Y |
| **Windows Server 2008 R2 x64** | Y | Y | Y |

- Check the FAQ for details on OS level, kernel levels, patches, etc
- GPFS V3.2.1 went EOS on September 30, 2011
  GPFS V3.3 goes EOM on April 17, 2012

# LICENSING

- GPFS Server license (performs mgmt functions or exports data)
  - GPFS management roles: quorum node, file system manager, cluster configuration manager, NSD server
  - Exports data through applications such as NFS, CIFS, FTP or HTTP
  - Data access local or via network
    The GPFS cluster *requires*, at minimum, one GPFS Server licensed node (LPAR).
    Two server licensed nodes provide minimum high- availability.
- GPFS client license (no mgmt functions, local consumer)
  - Data access can be local or via network
- Per PVU on x86 for Linux, Windows
- Per core on Power Systems for AIX, Linux
- MultiSystem Offering available (1,000+ nodes), OEM and IP licenses available

# GPFS Architecture

Parallel File System for Cluster Computers Based on Shared Disk (SAN) Model

*Cluster* – a collection of fabric- interconnected nodes (IP, SAN, …)

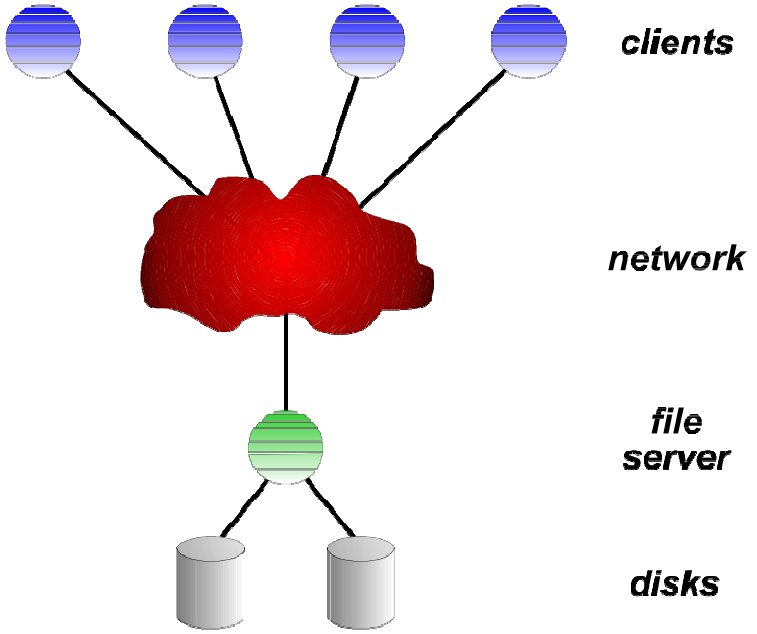
*Switching Fabric – system or storage area network (SAN)*

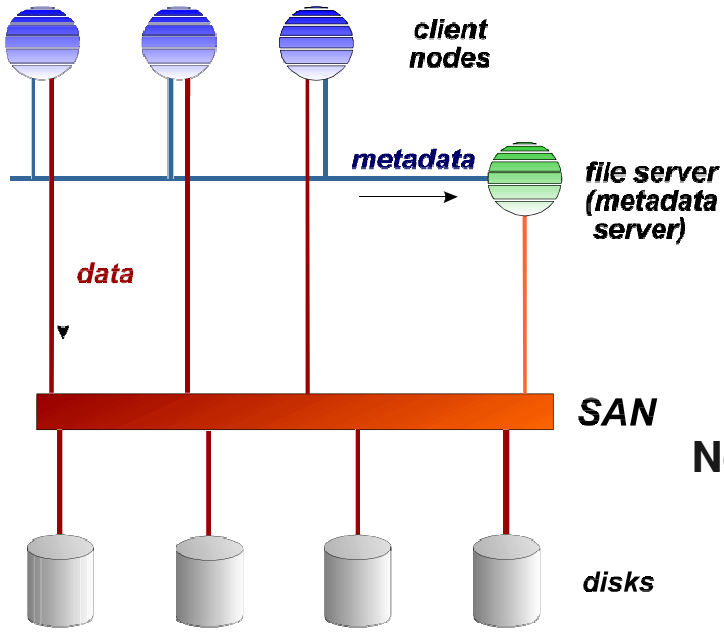
*Shared disk* - all data and metadata on fabric-attached disk. SAN or network block

*Parallel* - data and metadata flows from all of the nodes to all of the disks in parallel under control of distributed lock manager.

# What GPFS is ***NOT***

Not a client-server file system like NFS, DFS, or AFS: no single-server bottleneck, no protocol overhead for data transfer




**Not like some SAN file systems - no distinct metadata server (which is a potential bottleneck)**

Diagram & slide courtesy of IBM

# GPFS striping

- GPFS does block level striping by spinning disks in parallel
  - Stripe across LUNs within a pool
  - File placed in a pool based on creation policy (more later in ILM)
  - Stripe use either scatter or cluster semantics depending on file system policy
- GPFS striping is independent of RAID
  - GPFS blocks reside on Raid LUNs
  - Disk fencing enforced on erratic servers (required)
  - GPFS block size should be integral number of RAID stripe size
- GPFS block sizes for data and metadata
  - Data only LUNs
  - Metadata only LUNs

# GPFS locking and distributed management

- GPFS allows parallel access
  - Supports flock() from multiple nodes on same file. Can use maxFcntlRangePerFile to specify maximum number of fcntl() locks per file
  - Supports byte range locking so entire file is not locked
- Distributed management
  - Clients share data and metadata using POSIX semantics
  - Sending a message for each IO transaction will not scale
  - Sharing and synchronization is management by distributed locking that permits byte ranges
- The GPFS daemon on each node is the management interface
  - File manager coordinates metada token management
  - Tokens are required by nodes for read/write operations
  - Token manager coordinates requests including conflict resolution

# GENERAL TUNING

Check the FAQ at:
http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=%2Fcom.ibm.cluster.gpfs.doc%2Fgpfs_faqs%2Fgpfsclustersfaq.html

Clocks for all cluster nodes must be synchronized

Cache

- pagepool
  - Used to cache user data and filesystem metadata
  - Allows for asynchronous reads and writes
  - Improves prefetch but requires real memory
  - Larger values means avoiding synchronous I/O longer
  - mmchconfig pagepool=4G
- maxFilesToCache
  - Range is 1 to 100,000,000 – default is 4000 on GPFS 3.5
  - Total different files that can be cached at once
  - Memory required to cache inodes and control data structures:
    - maxFilesToCache x 3KB
- maxStatCache
  - Range is 0 to 100,000,000 – default is 1000 on GPFS 3.5
  - Additional memory to improve stat calls
  - Total memory used is
    - maxStatCache x 400

Total memory = pagepool + (maxFilesToCache * 3KB) + (maxStatCache * 400)

Size of the two caches is limited to <=50% of the memory for the node

# GPFS Pagepool

- GPFS pagepool acts as a cache (coherency for atomic operations)
  - Typically 128MB to 16GB
  - For Linux at most 50% of physical memory
- Pinned memory
- Optimum sizes at max performance, larger not useful
  - Formula: enough to hold 2 application operations for each LUN
  - Number of LUNs * sizeof(block) * 2 * max number of tasks per node
- Large pagepools most helpful when
  - Writes overlap computation
  - Heavy re-use of records (temporal locality)
  - Semi-random access with moderate temporal locality
- vmtune does not affect GPFS pagepool Pagepool size can be customized to client
- Recommend setting the variable vm.min_free_kbytes 5-6% of total memory available on a server to prevent exhausting memory

# CLUSTER SETUP

**Defaults**

# mmlsconfig
autoload yes
minReleaseLevel 3.5.0.11
adminMode central

*TUNABLES*
minReleaseLevel 3.5.0.0
pagepool 1G
Maxblocksize 4m

*Oracle General thoughts*
Set GPFS filesystem blocksize:

- 512KB generally
- 256KB if shared filesystem with lots of small files
- 1MB for filesystems >1TB

**Possibly Tweak**

maxMBpS 5000

- Estimate of what a single node can expect for performance
- For 1Gb/sec that is 100MB/sec so we set this to 150 (20-50% higher)

See:
http://www.ibmsystemsmag.com/aix/administrator/performance/Breaking-the-Bottleneck/

# ORACLE AND GPFS

*Oracle General thoughts*

https://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.v3r5.gpfs300.doc/bl1ins_oracle.htm

Set GPFS filesystem blocksize:

- 512KB generally
- 256KB if shared filesystem with lots of small files
- 1MB for filesystems >1TB

Set GPFS worker1Threads to allow parallelism

- mmchconfig prefetchThreads
- mmchconfig worker1Threads
- mmchconfig nsdMaxWorkerThreads
- Maximum total for the above three is 1500 on AIX and 500 for Linux/Windows

See web page above for additional recommendations for Oracle setup especially for Oracle RAC

# BUILDING A CLUSTER

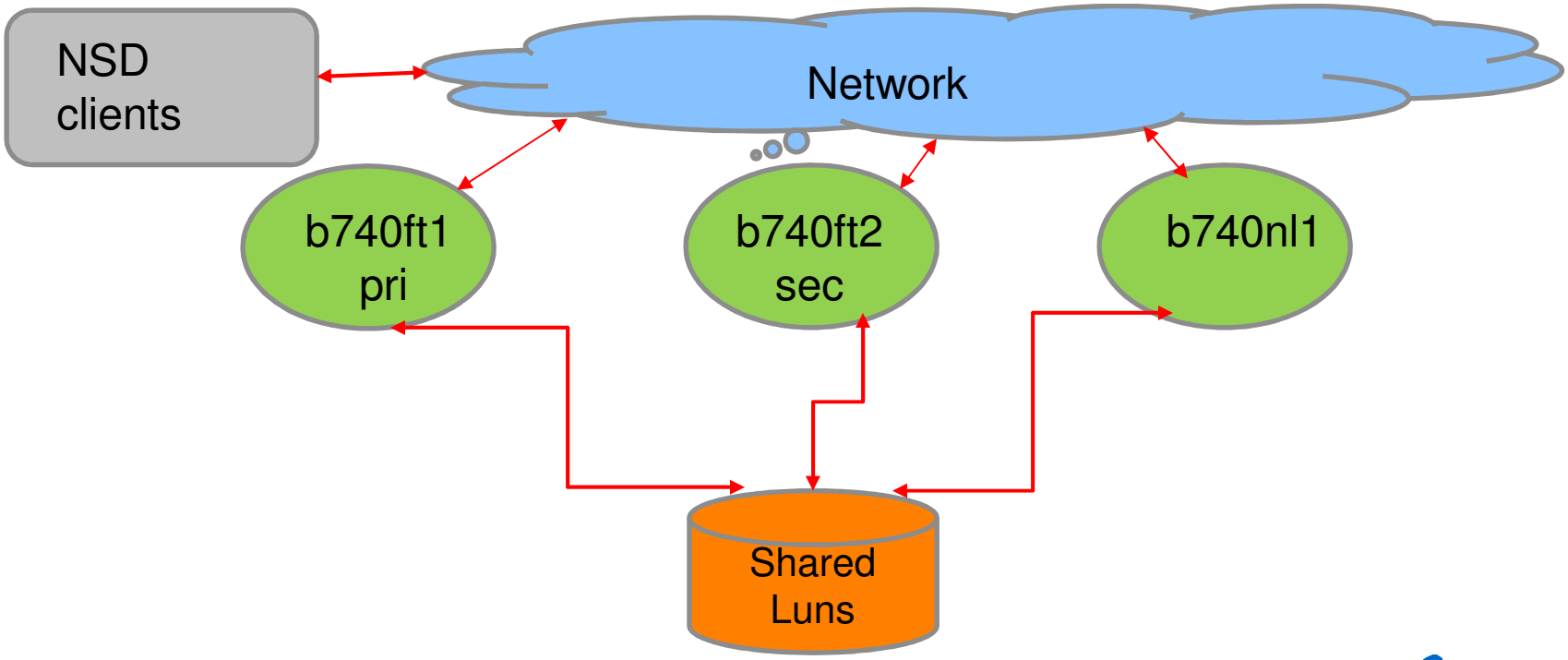
# INSTALLATION

## Options for cluster

1. 2 server nodes with shared luns and a tiebreaker disk
2. 3 server nodes with shared luns (8) and failure groups

## Went with option 2

All other nodes will be GPFS clients using NSDs






FORSYTHE

# FAILURE GROUPS

We want to have disk redundancy from each of 2 v7000s
I only have 1 v7000 so I am going to simulate it by providing 8 disks and pretending I have a set of 4 from each of 2 different v7000s

We have 8 disks zoned to all 3 servers
hdisk1-4 we will treat as v7000a
hdisk5-8 we will treat as v7000b

We will put hdisk1-4 into failure group 2
We will put hdisk5-8 into failure group 3
This provides us with a similar effect to LVM mirroring in AIX, something that is not supported in GPFS

Gives you full redundancy
We tested this across 2 x v7000s and pulled one out by unzoning it and the cluster stays up

# INSTALLATION

Install servers with AIX
Zone the GPFS luns to all 3 servers (I used vSCSI but you can use NPIV)
Set the fibre adapters in each LPAR (and VIO if NPIV)
chdev -l fcs? -a max_xfer_size=0x200000 -a num_cmd_elems=2048 –P
Check with your disk vendors if these are supported
If you are using NPIV make sure the VIO servers are set to at least the same value as the clients

On each LPAR I set the following on the disks
chdev -l hdisk? -a queue_depth=96 -a reserve_policy=no_reserve –P
If using sddpcm set all the disks to use load_balance as the algorithm
    pcmpath set device 1 10 algorithm lb

Additionally put a PVID on each disk
chdev –l hdisk? –a pv=yes
This will help you match up the disks later
Reboot VIO servers if changed and then clients

# INSTALLATION

The new hdisks should now show up on the LPARs

On all 3 LPARs install GPFS

- Install the base for 3.5 and then FP15
- Add /usr/lpp/mmfs/bin to end of PATH in /etc/environment
- Also in /etc/environment add:
  - WCOLL=/usr/local/etc/gpfs-nodes.txt
- In /etc/security/limits set fsize=-1 and nofiles=20000 (or higher)

Setup SSH between all 3 LPARs with no passwords

- ssh from each node to each node including to itself
- This ensures you show up in known_hosts
- On each node:
  - ssh-keygen –t dsa
  - Copy contents of /.ssh/id_dsa.pub and cat to /.ssh/authorized_keys2 on all 3 LPARs
  - This means ensuring it is in your own authorized_keys2
  - /etc/ssh/sshd_config must permit root login – see reference on strengthening ssh
  - You should now be able to ssh between the lpars as root with no password

Logout and back in

# INSTALLATION

On all 3 LPARs create /usr/local/etc/gpfs-nodes.txt with 3 lines:
b740ft1
b740ft2
b740nl1

On b740ft1 (primary node) we will create in /usr/local/etc the following:

gpfsnodes.txt

b740ft1
b740ft2
b740nl1

gpfs-nodesinit.txt

b740ft1:quorum-manager
b740ft2:quorum-manager
b740nl1:quorum-manager

# NSD DEFINITION FILE

Create the NSD stanza to use for disks etc/usr/local/software/gpfs-etc/nsdstanza.txt
We only need this on b740ft1, the primary

```
%nsd:  nsd=nsdfg2disk1 device=/dev/hdisk1 usage=dataAndMetadata failuregroup=2 pool=system
%nsd:  nsd=nsdfg2disk2 device=/dev/hdisk2 usage=dataAndMetadata failuregroup=2 pool=system
%nsd:  nsd=nsdfg2disk3 device=/dev/hdisk3 usage=dataAndMetadata failuregroup=2 pool=system
%nsd:  nsd=nsdfg2disk4 device=/dev/hdisk4 usage=dataAndMetadata failuregroup=2 pool=system
%nsd:  nsd=nsdfg3disk5 device=/dev/hdisk5 usage=dataAndMetadata failuregroup=3 pool=system
%nsd:  nsd=nsdfg3disk6 device=/dev/hdisk6 usage=dataAndMetadata failuregroup=3 pool=system
%nsd:  nsd=nsdfg3disk7 device=/dev/hdisk7 usage=dataAndMetadata failuregroup=3 pool=system
%nsd:  nsd=nsdfg3disk8 device=/dev/hdisk8 usage=dataAndMetadata failuregroup=3 pool=system
```

# GPFS SETUP

On b740ft1, the primary

First create the cluster with b740ft1 as primary and b740ft2 as secondary

```
mmcrcluster -C GPFSCLUST1 -p b740ft1 -s b740ft2 -r /usr/bin/ssh -R /usr/bin/scp -N /usr/local/etc/gpfs-
    nodesinit.txt -A
```

Accept the licenses
mmchlicense server --accept -N b740ft1,b740ft2,b740nl1

# CHECK CLUSTER

```
# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         GPFSCLUST1.b740ft1
  GPFS cluster id:           11167562106602553978
  GPFS UID domain:           GPFSCLUST1.b740ft1
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    b740ft1
  Secondary server:  b740ft2

 Node  Daemon node name  IP address     Admin node name  Designation
-----------------------------------------------------------------------
   1   b740ft1           10.250.134.68  b740ft1          quorum-manager
   2   b740ft2           10.250.134.69  b740ft2          quorum-manager
   3   b740nl1           10.250.134.58  b740nl1           quorum-manager
```

# CHECK CLUSTER

```
mmlsconfig
# mmlsconfig
Configuration data for cluster GPFSCLUST1.b740ft1:
------------------------------------------------
myNodeConfigNumber 1
clusterName GPFSCLUST1.b740ft1
clusterId 11167562106602553978
autoload yes
dmapiFileHandleSize 32
minReleaseLevel 3.5.0.11
adminMode central

File systems in cluster GPFSCLUST1.b740ft1:
---------------------------------------
(none)
```

# CHECK CLUSTER

mmgetstate –av

# mmgetstate -av

```
 Node number  Node name        GPFS state
------------------------------------------
       1      b740ft1          active
       2      b740ft2          active
       3      b740nl1           active
```

# CREATE THE NSDS

mmcrnsd -F /usr/local/software/gpfs-etc/nsdstanza.txt

lspv on b740ft1

```
# lspv
hdisk0          00f6934cde2b117c                    None
hdisk1          00f6934cde2b11c8                    nsdfg2disk1
hdisk2          00f6934cde2b1217                    nsdfg2disk2
hdisk3          00f6934cde2b126a                    nsdfg2disk3
hdisk4          00f6934cde2b12bc                    nsdfg2disk4
hdisk5          00f6934cde2b131f                    nsdfg3disk5
hdisk6          00f6934cde2b1376                    nsdfg3disk6
hdisk7          00f6934cfd8e9d91                    nsdfg3disk7
hdisk8          00f6934cfd8ec029                    nsdfg3disk8
hdisk9          00f6934c94514420                    rootvg          active
```

They will also show on the other 2 servers (b740ft2 and b740nl1)

# CHECK THE NSDS

# mmlsnsd

```
 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 (free disk)   nsdfg2disk1  (directly attached)
 (free disk)   nsdfg2disk2  (directly attached)
 (free disk)   nsdfg2disk3  (directly attached)
 (free disk)   nsdfg2disk4  (directly attached)
 (free disk)   nsdfg3disk5  (directly attached)
 (free disk)   nsdfg3disk6  (directly attached)
 (free disk)   nsdfg3disk7  (directly attached)
 (free disk)   nsdfg3disk8  (directly attached)
```

NOTE the above only support direct attach so we need to add network connectivity for our NSD only clients

# CHANGE THE NSDS

Change NSDs to support Direct and Network connectivity

mmchnsd "nsdfg2disk1:b740ft1,b740ft2,b740nl1"
mmchnsd "nsdfg2disk2:b740ft1,b740ft2,b740nl1"
mmchnsd "nsdfg2disk3:b740ft1,b740ft2,b740nl1"
mmchnsd "nsdfg2disk4:b740ft1,b740ft2,b740nl1"
mmchnsd "nsdfg3disk5:b740ft1,b740ft2,b740nl1"
mmchnsd "nsdfg3disk6:b740ft1,b740ft2,b740nl1"
mmchnsd "nsdfg3disk7:b740ft1,b740ft2,b740nl1"
mmchnsd "nsdfg3disk8:b740ft1,b740ft2,b740nl1“

The default order of access used in disk discovery:
1.Local block device interfaces for SAN, SCSI or IDE disks
2.NSD servers

Each GPFS node does a disk discovery upon daemon startup and will determine at that time if disk access is local, or via the network-based NSD server.

# RECHECK THE NSDS

mmlsnsd
# mmlsnsd

```
 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 (free disk)   nsdfg2disk1  b740ft1,b740ft2,b740nl1
 (free disk)   nsdfg2disk2  b740ft1,b740ft2,b740nl1
 (free disk)   nsdfg2disk3  b740ft1,b740ft2,b740nl1
 (free disk)   nsdfg2disk4  b740ft1,b740ft2,b740nl1
 (free disk)   nsdfg3disk5  b740ft1,b740ft2,b740nl1
 (free disk)   nsdfg3disk6  b740ft1,b740ft2,b740nl1
 (free disk)   nsdfg3disk7  b740ft1,b740ft2,b740nl1
 (free disk)   nsdfg3disk8  b740ft1,b740ft2,b740nl1
```

# CONFIG FILES CREATED ON PRIMARY B740FT1 IN /VAR/MMFS/GEN

```
# pwd
/var/mmfs/gen

# more mmfs.cfg
#
#   WARNING:   This is a machine generated file.  Do not edit!
#   Use the mmchconfig command to change configuration parameters.
#
myNodeConfigNumber 1
clusterName GPFSCLUST1.b740ft1
clusterId 11167562106602553978
autoload yes
dmapiFileHandleSize 32
minReleaseLevel 1340 3.5.0.11

# more mmfsNodeData
%%home%%:20_MEMBER_NODE::1:1:b740ft1:10.250.134.68:b740ft1:manager::::::b740ft1
:b740ft1:1344:3.5.0.15:AIX:Q::::::ser
ver::
```

# CONFIG FILES CREATED ON PRIMARY B740FT1 IN /VAR/MMFS/GEN

```
# more nsdmap
0AFA86445328990B /dev/rhdisk7 hdisk
0AFA864453289909 /dev/rhdisk5 hdisk
0AFA864453289907 /dev/rhdisk3 hdisk
0AFA864453289904 /dev/rhdisk1 hdisk
0AFA864453289906 /dev/rhdisk2 hdisk
0AFA864453289908 /dev/rhdisk4 hdisk
0AFA86445328990A /dev/rhdisk6 hdisk
0AFA86445328990C /dev/rhdisk8 hdisk

# more nsdpvol
hdisk1 nsdfg2disk1
hdisk2 nsdfg2disk2
hdisk3 nsdfg2disk3
hdisk4 nsdfg2disk4
hdisk5 nsdfg3disk5
hdisk6 nsdfg3disk6
hdisk7 nsdfg3disk7
hdisk8 nsdfg3disk8
```

# ADD AN NSD (NETWORK ONLY) CLIENT

On bpicsd install GPFS and setup SSH as above – do not add the license

mmaddnode -N bpicsd:client
mmchlicense client --accept -N bpicsd

mmlscluster
# mmlscluster

```
GPFS cluster information
========================
  GPFS cluster name:         GPFSCLUST1.b740ft1
  GPFS cluster id:           11167562106602553978
  GPFS UID domain:           GPFSCLUST1.b740ft1
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    b740ft1
  Secondary server:  b740ft2

 Node  Daemon node name  IP address     Admin node name  Designation
-----------------------------------------------------------------------
   1   b740ft1           10.250.134.68  b740ft1          quorum-manager
   2   b740ft2           10.250.134.69  b740ft2          quorum-manager
   3   b740nl1           10.250.134.58  b740nl1          quorum-manager
   4   bpicsd            0.250.134.32  bpicsd
```



FORSYTHE

# START AND CHECK NEW CLIENT

```
mmstartup -N bpicsd
# mmstartup -N bpicsd

Tue Mar 18 14:13:11 CDT 2014: mmstartup: Starting GPFS ...
# mmgetstate -av

 Node number  Node name        GPFS state
------------------------------------------
       1      b740ft1          active
       2      b740ft2          active
       3      b740nl1          active
       4      bpicsd           active
# mmgetstate -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
-------------------------------------------------------------------------------
       1      b740ft1           2        3          4        active      quorum node
       2      b740ft2           2        3          4        active      quorum node
       3      b740nl1           2        3          4        active      quorum node
       4      bpicsd            2        3          4        active

 Summary information
---------------------
Number of nodes defined in the cluster:            4
Number of local nodes active in the cluster:       4
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     3
Number of quorum nodes active in the cluster:      3
Quorum = 2, Quorum achieved
```

# OK LETS ADD A FILESYSTEM

Use default blocksize and use replication
This create /gpfs0 across the 4 disks in fg2 and replicates them across the 4 disks in fg3.
-m2 says 2 replicas for metadata, -M2 says max of 2 metadata replicas
-r2 says 2 replicas for data, -R2 says max of 2 data replicas

```
mmcrfs /gpfs0 gpfs0 -F /usr/local/software/gpfs-etc/nsdstanza.txt -m2 -M2 -r 2 -R 2

mmmount all -a

# df -g /gpfs0
Filesystem    GB blocks      Free %Used    Iused %Iused Mounted on
/dev/gpfs0       320.00    319.38    1%     4038     2% /gpfs0
```

# CHECK THE FILESYSTEM IN GPFS

```
# mmdf gpfs0
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 523 GB)
nsdfg2disk1          52428800        2 yes      yes       52347392 (100%)          248 ( 0%)
nsdfg2disk2          52428800        2 yes      yes       52347392 (100%)          472 ( 0%)
nsdfg2disk3          52428800        2 yes      yes       52347392 (100%)          248 ( 0%)
nsdfg2disk4          52428800        2 yes      yes       52346624 (100%)          472 ( 0%)
nsdfg3disk5          52428800        3 yes      yes       52346368 (100%)          472 ( 0%)
nsdfg3disk6          52428800        3 yes      yes       52347392 (100%)          248 ( 0%)
nsdfg3disk7          10485760        3 yes      yes       10404352 ( 99%)          488 ( 0%)
nsdfg3disk8          10485760        3 yes      yes       10404352 ( 99%)          488 ( 0%)
                -------------                       -------------------- -------------------
(pool total)         335544320                          334891264 (100%)         3136 ( 0%)

                =============                       ==================== ===================
(total)              335544320                          334891264 (100%)         3136 ( 0%)

Inode Information
-----------------
Number of used inodes:            4038
Number of free inodes:          327738
Number of allocated inodes:     331776
Maximum number of inodes:       331776
```

# CHANGE THE NSD MOUNT OPTION FOR BPICSD

Change NSDs for the client to support only Network connectivity

By default, if GPFS detects a failure in disk access locally, it will automatically switch to using the network-based NSD server(s). It will periodically check local access and switch back automatically.

The useNSDservers mount option can be set to change this default behavior:

| | |
|---|---|
| asfound: | don't change the access method from local to network |
| asneeded (default): | try local first, then network, switch back |
| always: | only network access |
| never: | only local access |

Change bpicsd (which failed to mount the above) to network mount only
On bpicsd
mmshutdown
mmstartup
mmumount gpfs0
mmmount gpfs0 -o useNSDserver=always

# FILES IN THE FILESYSTEM

Create a 4GB disk file
dd if=/dev/zero of=/gpfs0/4gbfile1 bs=1m count=4096

df -g /gpfs0
# df -g /gpfs0
Filesystem GB blocks Free %Used Iused %Iused Mounted on
/dev/gpfs0 320.00 311.37 3% 4039 2% /gpfs0

# du -sg /gpfs0/*
8.00 /gpfs0/4gbfile1

ls -al /gpfs0
-rw-r--r-- 1 root system 4294967296 Mar 18 14:16 4gbfile1

NOTE - we are using 2 replicas so the file above is 4GB but shows as 8GB due to the replica
The df will show the total space across all 8 disks but the used will show as 8gb instead of 4gb due to the replicas

# mmlsmount all
File system gpfs0 is mounted on 4 nodes.

# ADDING MORE DISKS TO A FILESYSTEM

Add 2 disks by zoning to all, run cfgmgr, set up queue_depth etc
New disks are hdisk10 and hdisk11
Reboot

Set up gpfsrdisks.2 with 2 new disks to be added - ensure they are in the correct failure groups

```
%nsd:  nsd=nsdfg2disk10 device=/dev/hdisk10 usage=dataAndMetadata failuregroup=2 pool=system
%nsd:  nsd=nsdfg3disk11 device=/dev/hdisk11 usage=dataAndMetadata failuregroup=3 pool=system
```

On primary:
mmcrnsd -F /usr/local/software/gpfs-etc/gpfsrdisks.2
Now add the two disks to the nodes
mmchnsd “nsdfg2disk10:b740ft1,b740ft2,b740nl1"
mmchnsd “nsdfg3disk11:b740ft1,b740ft2,b740nl1“

Add the disks to GPFS0 filesystem
mmadddisk gpfs0 -F gpfsrdisks.2 -r

The -r tells it to rebalance the files in the filesystem across all the disks
You can specify -a to make it asynchronous
You can also use -N b740nl1 to tell it to have b740nl1 do the restriping leaving the other 2 nodes to carry on working

Use df and mmdf to check all is well

# REMOVING THE CLUSTER

On the primary – b740ft1

mmumount all -a
mmdelfs gpfs0
mmdelnsd nsdfg2disk1
mmdelnsd nsdfg2disk2
mmdelnsd nsdfg2disk3
mmdelnsd nsdfg2disk4
mmdelnsd nsdfg3disk5
mmdelnsd nsdfg3disk6
mmdelnsd nsdfg3disk7
mmdelnsd nsdfg3disk8

mmshutdown -a
mmdelnode -a
(can also mmdelnode -N bpicsd)  removes just bpicsd

Check /var/mmfs/gen
It should only contain:
BallotNum  mmLockDir  mmfslog  nodeFiles  nsdpvol

# USEFUL COMMANDS

mmlscluster
mmlsconfig
mmgetstate -aLs
Mmlsmgr
mmdiag
mmdsh command

mmdf filesystemname (i.e. vgbackup)
mmlsnsd
mmlsdisk
lspv
mmlspv
mmlsfs filesystem
mslsfileset filesystem -d -i
(for filesystem use filesystem name i.e.
vgbackup)
mmadddisk
mmlspool filesystem
mmlspool filesystem all
mslspolicy filesystem
mmlsrecoverygroup
mmlssnapshot filesystem

lslpp -l | grep gpfs

mmbackup
mmbackupconfig
mmfsmount -L
mmmount filesystem
mmumount filesystem
mmlsmount all
mmdelfs filesystem
mmfsck filesystem
mmchfs filesystem -r 2
(changes filesystem to have 2 replicas)

# MMLSDISK GPFS0

```
# mmlsdisk gpfs0

disk         driver   sector     failure holds    holds                            storage
name         type       size     group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
nsdfg2disk1  nsd         512           2 yes      yes   ready         up           system
nsdfg2disk2  nsd         512           2 yes      yes   ready         up           system
nsdfg2disk3  nsd         512           2 yes      yes   ready         up           system
nsdfg2disk4  nsd         512           2 yes      yes   ready         up           system
nsdfg3disk5  nsd         512           3 yes      yes   ready         up           system
nsdfg3disk6  nsd         512           3 yes      yes   ready         up           system
nsdfg3disk7  nsd         512           3 yes      yes   ready         up           system
nsdfg3disk8  nsd         512           3 yes      yes   ready         up           system
```

# MMLSFS GPFS0

```
# mmlsfs gpfs0
flag                value                    description
------------------- ------------------------ -----------------------------------
 -f                 8192                     Minimum fragment size in bytes
 -i                 512                      Inode size in bytes
 -I                 16384                    Indirect block size in bytes
 -m                 2                        Default number of metadata replicas
 -M                 2                        Maximum number of metadata replicas
 -r                 2                        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
 -n                 32                       Estimated number of nodes that will mount file system
 -B                 262144                   Block size
 -Q                 none                     Quotas enforced
                    none                     Default quotas enabled
 --filesetdf        no                       Fileset df enabled?
 -V                 13.23 (3.5.0.7)          File system version
 --create-time      Tue Mar 18 14:14:42 2014 File system creation time
 -u                 yes                      Support for large LUNs?
 -z                 no                       Is DMAPI enabled?
 -L                 4194304                  Logfile size
 -E                 yes                      Exact mtime mount option
 -S                 no                       Suppress atime mount option
 -K                 whenpossible             Strict replica allocation option
 --fastea           yes                      Fast external attributes enabled?
 --inode-limit      331776                   Maximum number of inodes
 -P                 system                   Disk storage pools in file system
 -d                 nsdfg2disk1;nsdfg2disk2;nsdfg2disk3;nsdfg2disk4;nsdfg3disk5;nsdfg3disk6;nsdfg3disk7;nsdfg3disk8  Disks in file system
 --perfileset-quota no                       Per-fileset quota enforcement
 -A                 yes                      Automatic mount option
 -o                 none                     Additional mount options
 -T                 /gpfs0                   Default mount point
 --mount-priority   0                        Mount priority
```

# MONITORING

mmpmon - performance monitoring - needs an input stanza
Create pmonin.txt:
ver
io_s
fs_io_s
rhist off

mmpmon -i pmonin.txt -r 2 -d 2000
Above runs 2 snaps 2 seconds (2000 microseconds) apart
Add -p flag to get output in single line format

DIRECTORIES:

| | |
|---|---|
| /var/mmfs/gen | Configuration files created and used by GPFS |
| /usr/lpp/mmfs/bin | GPFS commands |
| /usr/local/etc | Our GPFS input files for creation etc |

LOGS

- /var/adm/ras/mmfs.log.latest
- /var/adm/ras/mmfs.log.previous
- AIX Error Log / Linux syslog

# MMPMON

```
# mmpmon -i pmonin.txt -r 2 -d 2000
1.
mmpmon node 10.250.134.68 name b740ft1 version 3.5.1
mmpmon node 10.250.134.68 name b740ft1 io_s OK
timestamp:      1397515894/707607
bytes read:             0
bytes written:  4294967296
opens:                  3
closes:                 3
reads:                  0
writes:              4096
readdir:                4
inode updates:          9
mmpmon node 10.250.134.68 name b740ft1 fs_io_s OK
cluster:        GPFSCLUST1.b740ft1
filesystem:     gpfs0
disks:                  8
timestamp:      1397515894/707738
bytes read:             0
bytes written:  4294967296
opens:                  3
closes:                 3
reads:                  0
writes:              4096
readdir:                4
inode updates:          9
```

```
2 iterations 2000 microseconds apart
2.
mmpmon node 10.250.134.68 name b740ft1 rhist off OK
mmpmon node 10.250.134.68 name b740ft1 version 3.5.1
mmpmon node 10.250.134.68 name b740ft1 io_s OK
timestamp:      1397515896/708125
bytes read:             0
bytes written:  4294967296
opens:                  3
closes:                 3
reads:                  0
writes:              4096
readdir:                4
inode updates:          9
mmpmon node 10.250.134.68 name b740ft1 fs_io_s OK
cluster:        GPFSCLUST1.b740ft1
filesystem:     gpfs0
disks:                  8
timestamp:      1397515896/708253
bytes read:             0
bytes written:  4294967296
opens:                  3
closes:                 3
reads:                  0
writes:              4096
readdir:                4
inode updates:          9
```

# SCRIPT TO DOCUMENT CLUSTER

```
#! /bin/sh
## script dated April 14, 2014, 2013
# Script runs on one of the GPFS cluster nodes
# Assumes directory will be created in /usr/local/perf
#
root="/usr/local/perf"
#
day="`/bin/date +'%d'`"
month="`/bin/date +'%m'`"
year="`/bin/date +'%y'`"
set -- Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
shift $month
lmonth="$1"
machine=`uname -n`
directory="`/bin/date +'%m%d%Y_%H%M'`"
machine_directory=`printf "%s_%s" $machine $directory`
mkdir $root/$machine_directory
cd $root/$machine_directory
logit="$root/$machine_directory/$machine.$day$lmonth$year"

#
mmlsconfig  >$logit.mmlsconfig.txt 2>&1 &
mmlscluster >$logit.mmlscluster.txt 2>&1 &
mmgetstate –Las >$logit.mmgetstate1.txt 2>&1 &
mmgetstate -av >$logit.mmgetstate2.txt 2>&1 &
mmlsmgr >$logit.mmlsmgr.txt 2>&1 &
mmlsnsd >$logit.mmlsnsd.txt 2>&1 &
mmlspv >$logit.mmlspv.txt 2>&1 &
mmlsrecoverygroup >$logit.mmlsrecovery.txt 2>&1 &

#
exit 0
```

# GPFS DOCUMENTATION

GPFS Documentation Home Page - links to Admin, etc guides
http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfsbooks.html

GPFS Commands
http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.v3r5.b740ft100.doc/bl1adm_command.htm

GPFS Backup Questions
https://www.ibm.com/developerworks/community/forums/html/topic?id=77777777-0000-0000-0000-000014482845

Strengthening the ssh configuration for GPFS
http://www.ibm.com/developerworks/aix/library/au-aix-modifying-ssh-configuration/

mmadddisk
https://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.v3r5.b740ft100.doc/bl1adm_mmadddisk.htm

Building a two-node GPFS cluster
https://www.ibm.com/developerworks/aix/library/au-aix-building-two-node-gpfs-cluster/

General
http://www.ibm.com/systems/clusters/software/gpfs/resources.html

# GPFS Documentation

Configure GPFS for reliability
http://www-03.ibm.com/systems/resources/configure-gpfs-for-reliability.pdf
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General%20Parallel%20File%20System%20(GPFS)/page/Configuring%20GPFS%20for%20Reliability
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General+Parallel+File+System+(GPFS)/page/Data+Replication+Scenarios

GPFS Storage Pools
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General%20Parallel%20File%20System%20%28GPFS%29/page/Using%20GPFS%20Storage%20Pools

GPFS Tuning
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General%20Parallel%20File%20System%20(GPFS)/page/Tuning%20Parameters

Integrating GPFS with TSM
https://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.v3r5.b740ft100.doc/bl1adm_backupusingmmbackup.htm
https://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.v3r5.b740ft100.doc/bl1adm_filesystemsbackedupusingtsminterface.htm
http://www-01.ibm.com/support/docview.wss?uid=swg27038312
http://www-05.ibm.com/de/events/gpfs-workshop/pdf/pr-7-TSM-HSM-for-GPFS-DMW-042013.pdf

GPFS Implementation redbook
http://publib-b.boulder.ibm.com/abstracts/sg247844.html?Open

# ARTICLES, MOVIES, ETC

Journal Articles

https://enterprisesystemsmedia.com/author/jaqui-lynch

http://www.ibmsystemsmag.com/authors/Jaqui-Lynch/

ForsytheTalks and Movies Site

http://www.circle4.com/movies

http://www.circle4.com/forsythetalks.html

Direct YouTube Feed

http://www.youtube.com/user/adespota4?feature=watch

# QUESTIONS?

Jaqui Lynch
lynchj@forsythe.com

Handout at:
http://www.circle4.com/forsythe/gpfs-introandsetup.pdf

Associated Article:
http://www.ibmsystemsmag.com/aix/administrator/lpar/three_node_gpfs/

FORSYTHE

# More on Parameters 1/2

The following list highlights common GPFS configuration settings:

***pagepool*** - The pagepool is used for I/O buffers to cache user data and indirect blocks. It's always pinned, and the default is fairly small. It's used to implement read/write requests asynchronously using the read-ahead and write-behind mechanisms. Increasing the pagepool increases the amount of data available in the cache for applications to use. This parameter is critical where applications perform significant amounts of random I/O.

***maxFilesToCache*** - This is the total number of different files that can be cached at any one time. This needs to be set to a large enough value to handle the number of concurrently open files and allow for caching those files.

***maxStatCache*** - This is additional pageable memory that's used to cache file attributes that aren't in the regular file cache. It defaults to 4 * maxFilesToCache.

***preFetchThreads*** - These are the maximum number of threads that can be dedicated to prefetching data for files that are read sequentially.

***Worker1Threads*** - The maximum number of threads that can be used for controlling sequential write-behind.

***Worker2Threads*** - The maximum number of threads that can be used for controlling other operations.

# More on Parameters 2/2

***maxMBps*** (definition from the provided default mmfs.cfg) - maxMBpS is an estimate of how many MBps of data can be transferred in or out of a single node. The value is used in calculating the amount of I/O that can be performed to effectively pre-fetch data for readers and/or or write-behind data from writers. The maximum number of I/Os in progress concurrently will be 2 * min(nDisks, maxMBpS*avgIOtime/ blockSize), where nDisks is the number disks that make a filesystem; avgIOtime is a measured average of the last 16 full block I/O times; and blockSize is the block size for a full block in the file-system (e.g., 256K). By lowering this value, you can artificially limit how much I/O one node can put on all the virtual shared disk (VSD) servers, if there are lots of nodes that can overrun a few VSD servers. Setting this too high will usually not hurt because of other limiting factors such as the size of the pagepool, or the number of prefetchThreads or worker1Threads.

***Blocksize*** - The blocksize determines the largest file system size and should be set to the application buffer size or the stripe size on the raid set. If this is done incorrectly, performance will suffer significantly. Once the blocksize is set, the minimum space required for a file will be 1/32 of the blocksize, so this setting requires an understanding of file sizes as well.