FSs for shared disks: a GFS overview

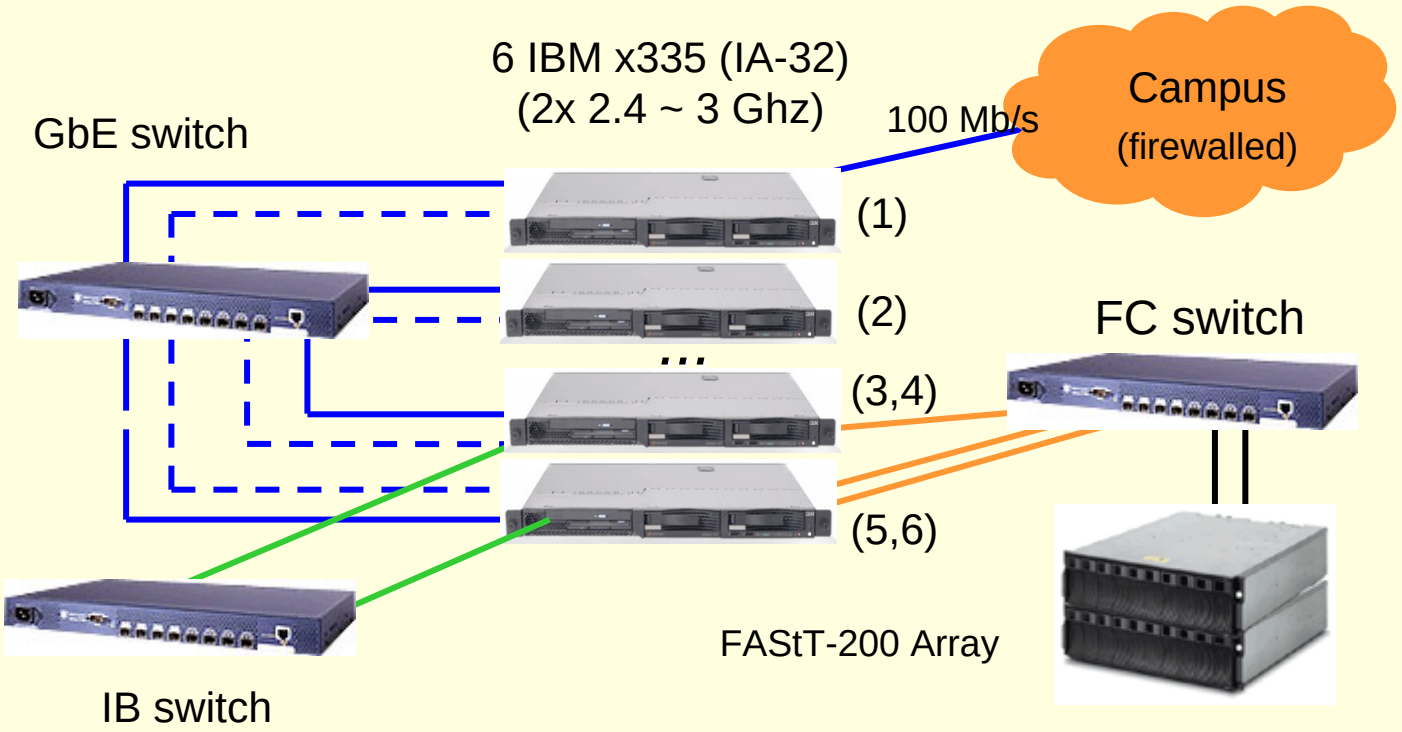
AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

Paulo Lopes

CITI and Department of Informatics, FCT
Universidade Nova de Lisboa

GFS is a SAN cluster file system

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra



We thank IBM for part of this equipment, donated under an Equinox grant

What is a SAN CFS?!

- *A Storage Area Network is:*
  - *An infrastructure to provide access to*
    - **storage** (block devices)
  - *A protocol (FC, SCSI-3)*
    - *Initiators (“hosts”, HBAs)*
    - *Targets (“disks”, LUNs)*
    - *Switches, Cabling,…*

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

What is a SAN CFS?!

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *A Cluster file system is:*
  - *Different things for different people, but…*
  - *Industry has its own view:*

    (1) *A cluster* [a SAN, a bunch of disk arrays…]

    (2) *Targets ("disks") are* **shared** *among hosts*

    (3) *"Single view of a filesystem" in the cluster*

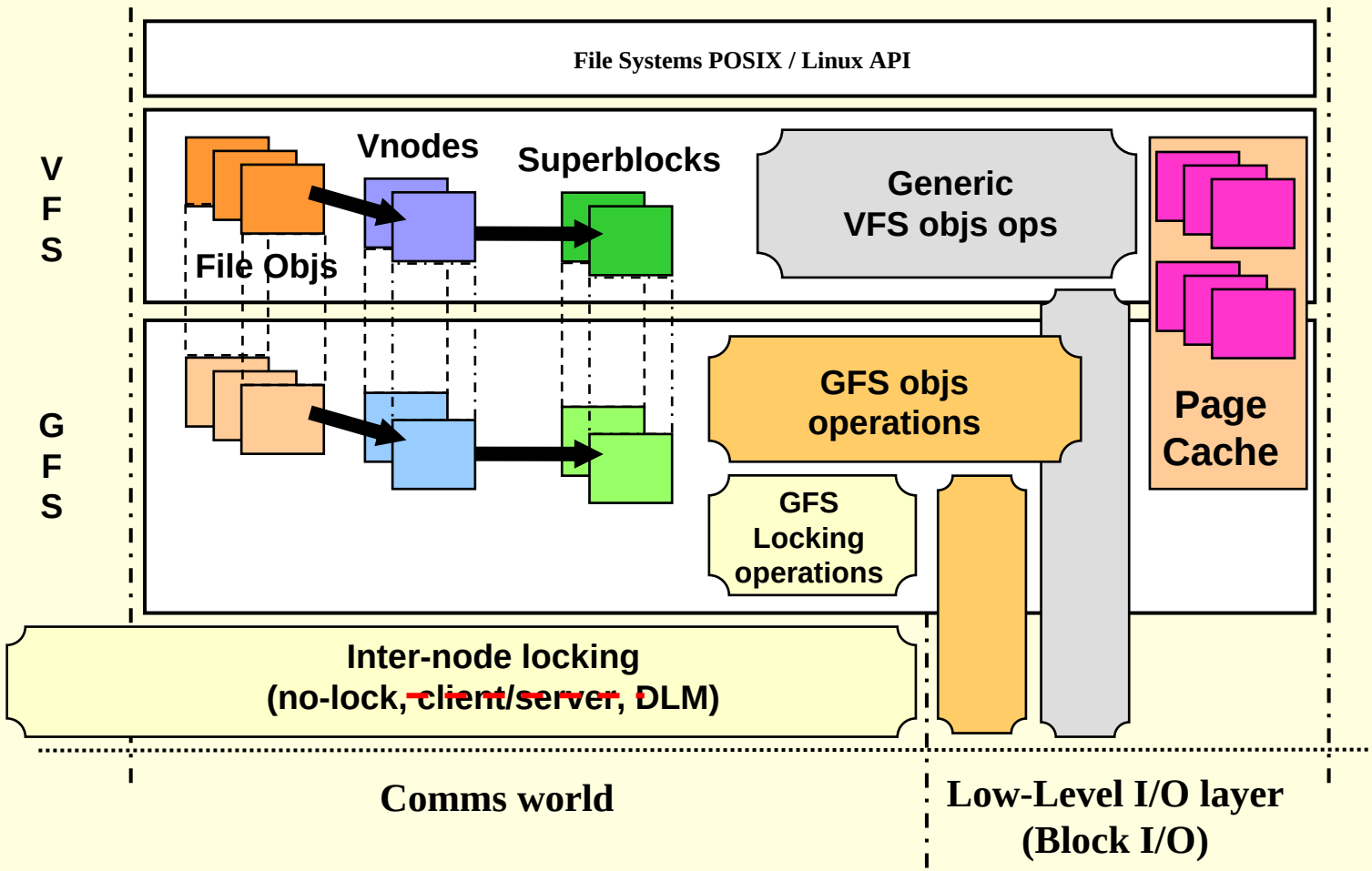
    E.g: *PVFS does not check (2)*
    GFS, GPFS check all
    Lustre… well, we can talk about it

GFS

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

GFS: architecture & VFS integration

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra



GFS: Semantics & VFS integration

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *UNIX single-node equivalent semantics*
  - *A read will return the result of the last write*
- *VFS integration*
  - *Fully integrated within VFS*
    - *All major API functions are implemented (inc. mmap'ed and direct I/O, file locking (mandatory and advisory, fcntl and POSIX)*
  - *Fully integrated with the VFS Page Cache*

GFS: Locking concepts (1)

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *Lock objects*
  - *Managed by a Lock Manager*
  - *Identified by (<Resource type>, < Resource number>)*
    - *Eg: "type == inode" and "number == inode number"*
  - *Two states:*
    - *Locked or unlocked*
  - *Two intentions:*
    - *Shared or Exclusive*

GFS: Locking concepts (2)

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *Glocks*
  - *Protect important objects (superblock, inode, …)*
  - *Have an associated vector of operations (vops)*
    - *Operations are different for distinct types of protected objects, eg: vops for "inode glock" is different from vops for "superblock glock".*
  - *Have a list of holders (see next slide)*
  - *Two stages:*
    - *Glock is held or not* **by a node**
  - *Two states:*
    - *Locked (shared or exclusive) or unlocked*

GFS: Locking concepts (3)

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *Holders*
  - *Record information about an entity (pid) that wants to/has managed to lock the glock*
  - *Record the state (shared, exclusive) desired/succeeded for the lock*
- *Brief description of the locking protocol:*
  - *A holder is created, to ask for a shared lock*
  - *The holder is enqueued, thus locking the glock*
    - *Either we get it immediately (we already had a compatible lock on the glock); or we wait for a call to the LM, and we sooner or later, get the lock*

GFS: Locking concepts (4)

- *A little bit more descriptive:*
  - *The glock is already locked (in this node):*
    - *if we're compatible, we're holders, else we wait*
  - *The glock is unlocked:*
    - *If a lock object for this glock does not exist, ask the LM to be created and granted to us, and we're holders*
    - *If the lock object is still cached in this node:*
      - *If unlocked, or still "cache locked" but in a different state, ask the LM, else we're immediately holders*
      - *If other node has to lose it's (cached) lock so that it can be granted us, that node will probably have to perform some cache flushing…*

GFS: Clusterwide coherency (1)

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *read()*
  - *Ask for a shared lock on "inode glock"*
  - *Perform the read using vfs functions & cache*
  - *Release the lock*
    - *It will stay in the cache, held by this node*
  - *Note:*
    - *If another node asks for an exclusive lock, when we release our lock an invalidation of all data, metadata, and the inode itself will be performed.*

GFS: Clusterwide coherency (2)

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *write()*
  - *Ask for an exclusive lock on "inode glock"*
  - *Perform the write using vfs functions & cache*
  - *Release the lock*
    - *It will stay in the cache, held by this node*
  - *Note:*
    - *If another node asks for a shared or exclusive lock, when we release our lock a flush of all data, metadata, and the inode itself will be performed, and then invalidated.*

GFS: Clusterwide coherency (3)

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *write() is more complicated than that ☺*
  - *Allocation of metadata (indirect blocks, RG bitmaps)*
  - *GFS is journaled…*
  - *supports quotas…*

What is pCFS

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *My filesystem ☺*
- *Wanna prove a CFS*
  - *need not be slow in R/W sharing…*
  - *… can compete with Parallel file systems*
- *How? Prototype built by modifying GFS*
  - *pCFS file = open(… | O_RD… | O_CLST…)*
    - *No O_CLST => standard GFS file*
  - *Uses fcntl/POSIX locks to define regions of sharing*
    - *Within regions, locking is "optimistic"*
  - *Data movement also cache-to-cache (LAN)*
    - *Handle false sharing / Data shipping for performance*

The End...

AFS BXFM CIFS DFS Echo FFS GFS GPFS HPFS InterMezzo
JFS KFS Lustre MFS NFS OCFS PVFS QFS River
Sprite Tiger UFS WAFL VFS XFS YFS Zebra

- *Merci…*

- *Questions?*