Mirror File System

A Multiple Server File System

John Wong

CTO

John.Wong@TwinPeakSoft.com

Twin Peaks Software Inc.

Multiple Server File System

- Conventional File System – EXT3/UFS and NFS
  - Manage files on a single server and its storage devices
- Multiple Server File system
  - Manage files on multiple servers and their storage devices

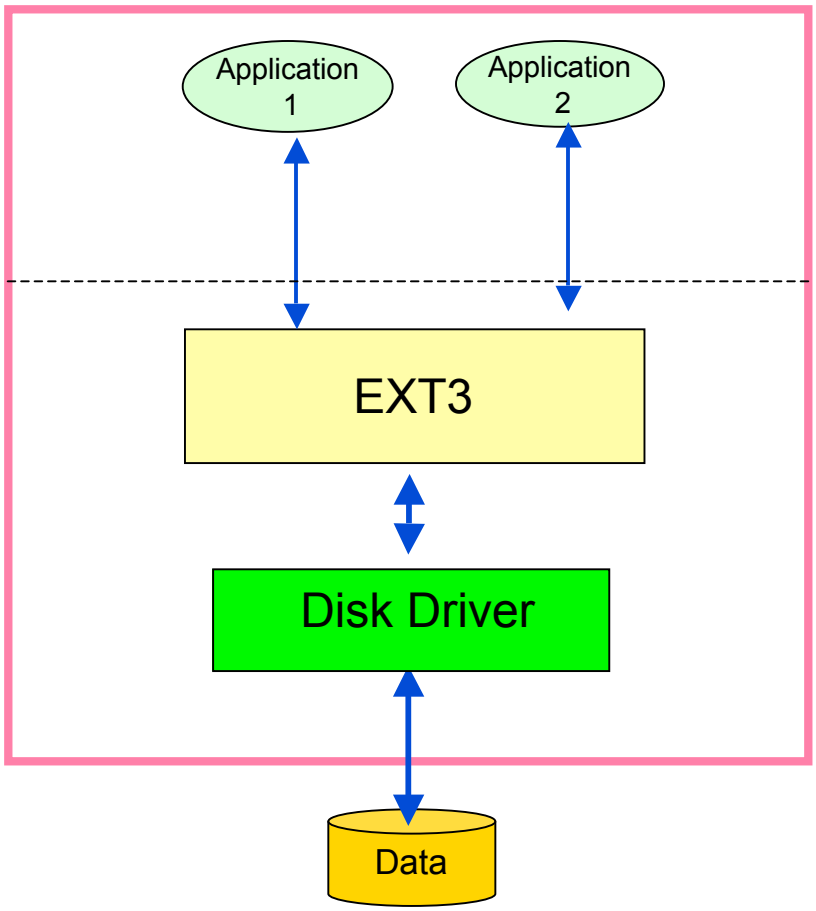
Problems

- Single resource is vulnerable
- Redundancy provides a safety net
  - Disk level => RAID
  - Storage level => Storage Replication
  - TCP/IP level => SNDR
  - File System level => CFS, MFS
  - System level => Clustering system
  - Application => Database

Why MFS?

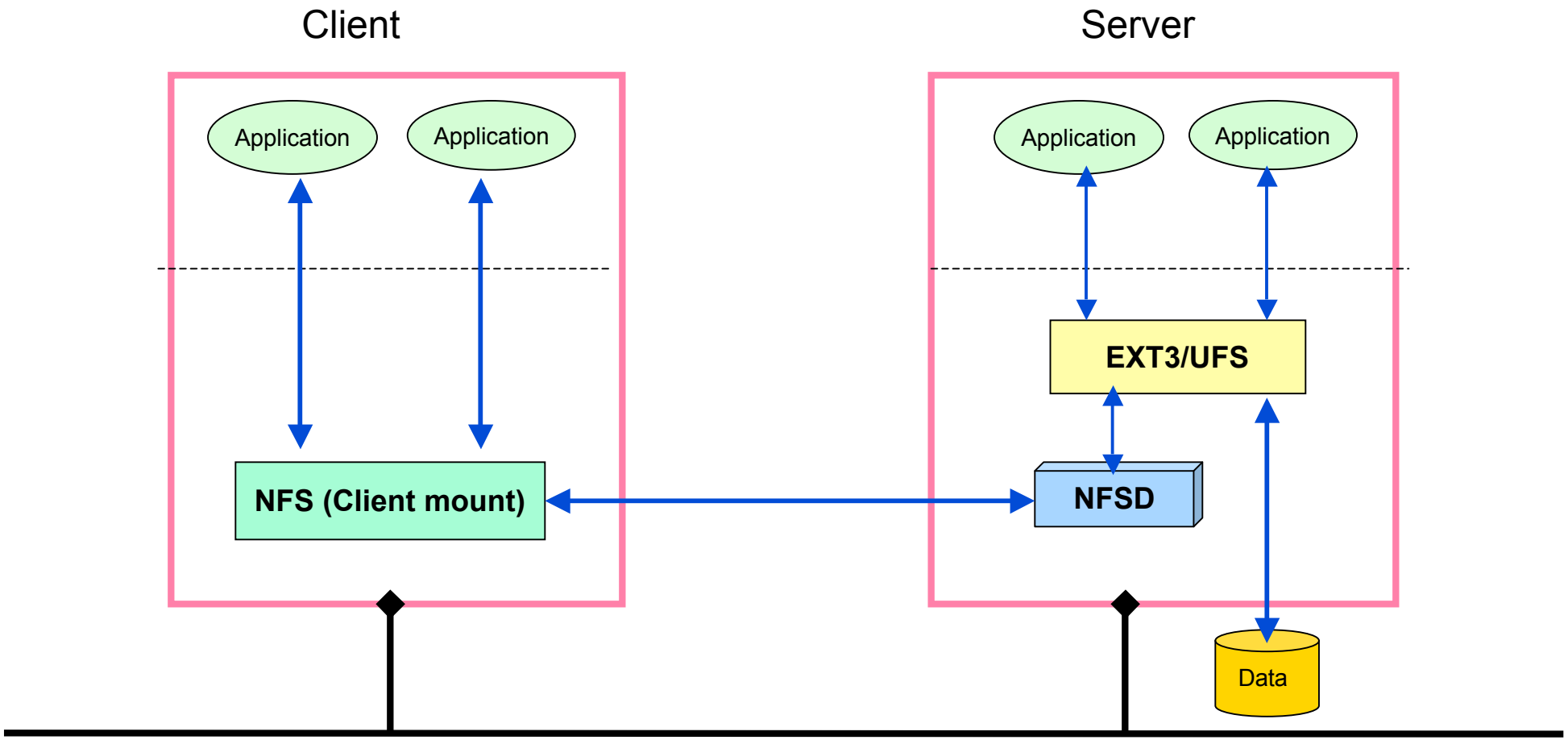
- Many advantages over existing technologies

Local File System



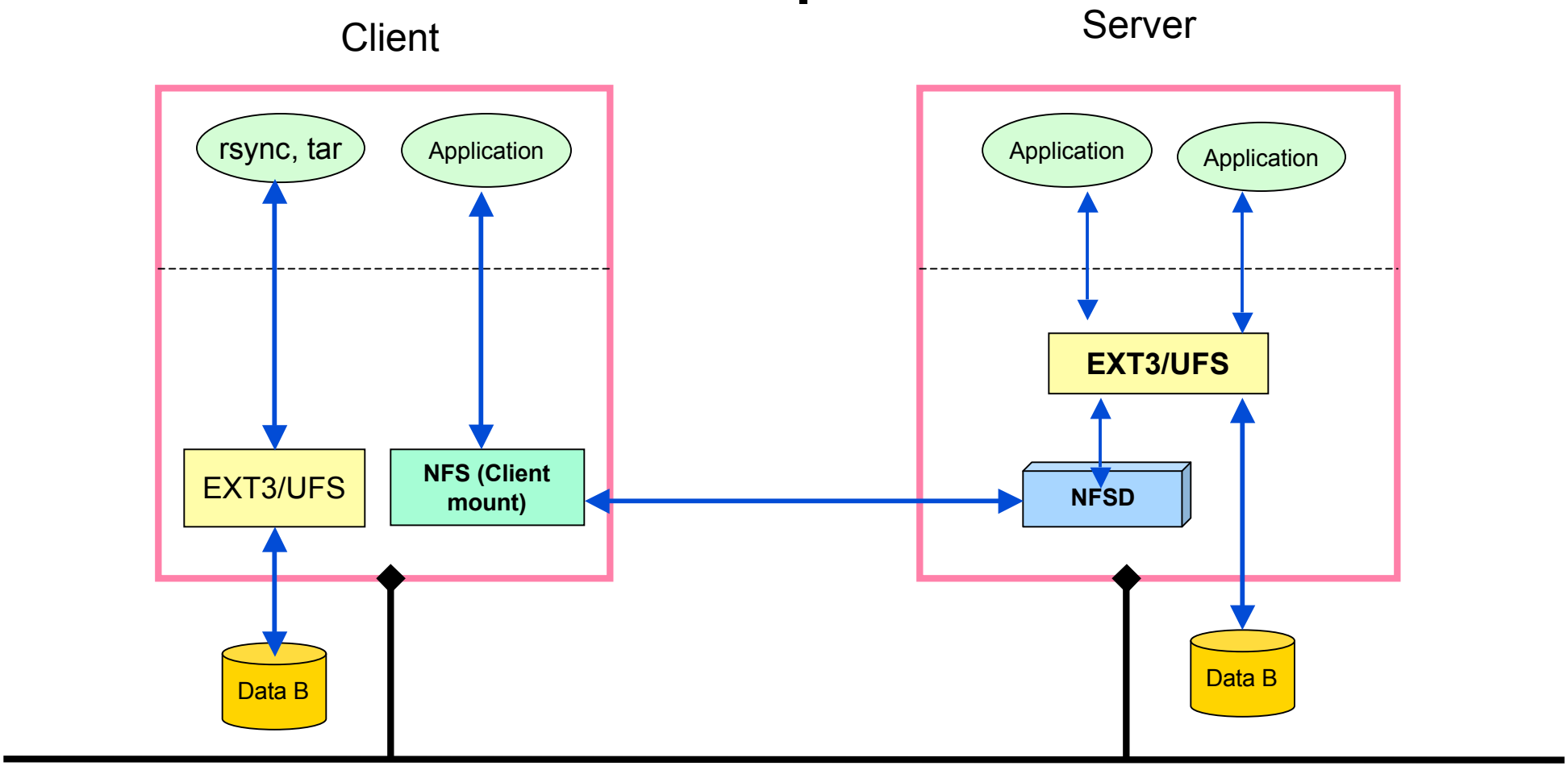
EXT3 manages file on the local server's storage devices

Network File System



NFS manages file on remote server's storage devices

EXT3 | NFS



Applications can only use either one, not both.

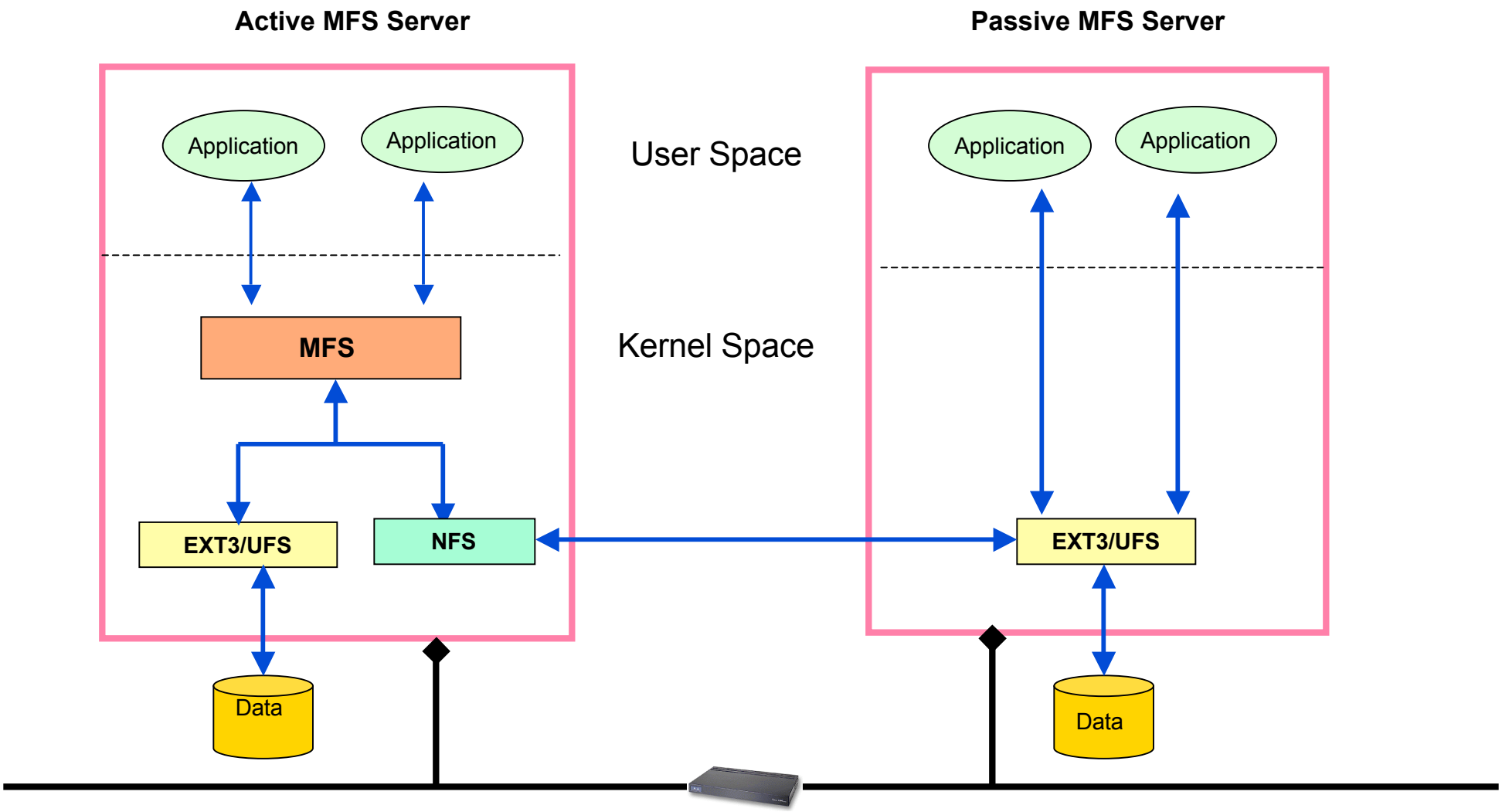
EXT3 + NFS ??

- Combine these two file systems to manage file on both local and remote servers storage devices

  -- at the same time

  -- in real time

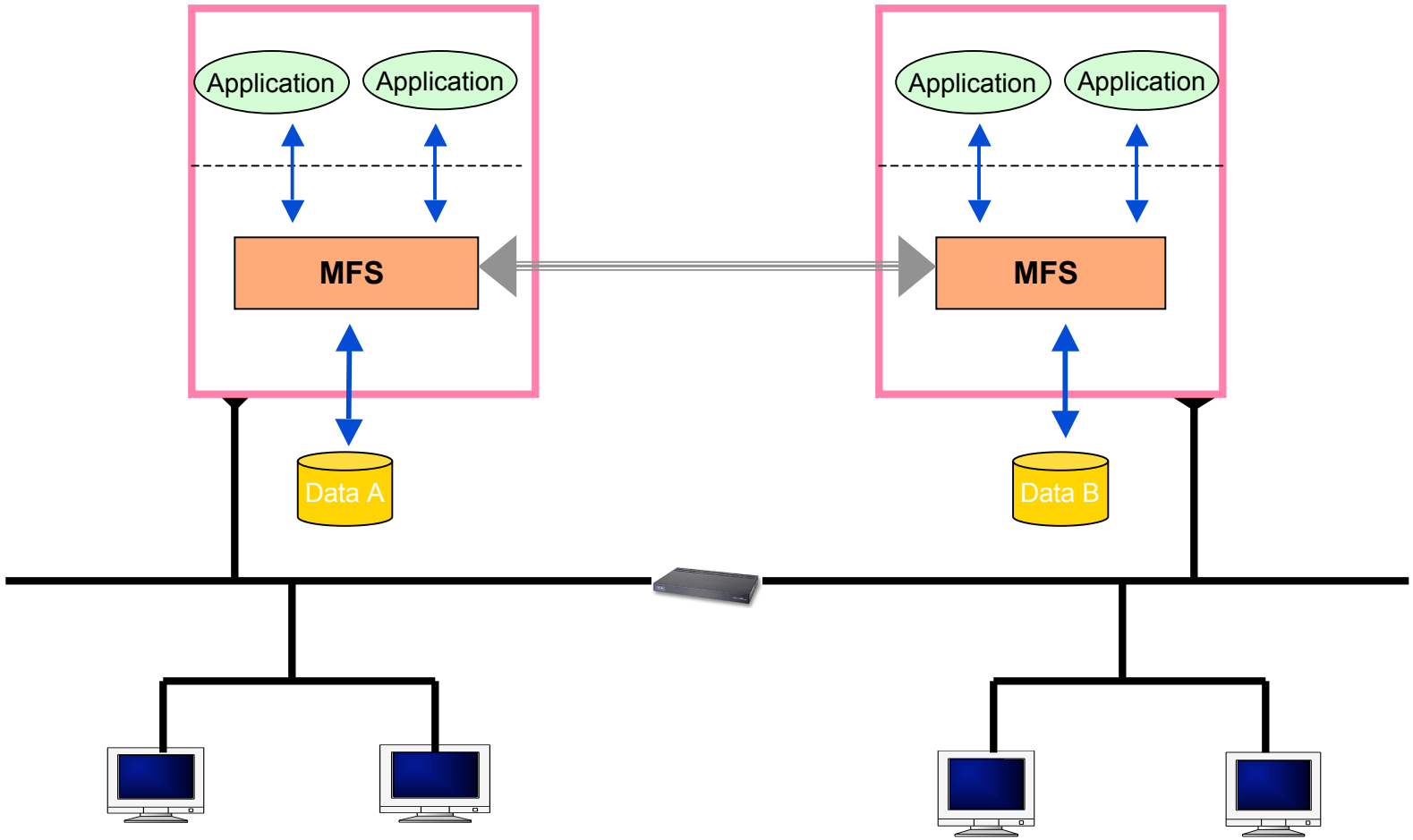
MFS = EXT3 + NFS



Building Block Approach

- MFS is a kernel loadable module
  - loaded on top of EXT3/UFS and NFS
- Standard VFS interface
- Provide Complete Transparency
  - to users and applications
  - to underlining file systems

Q & A



Advantages

- Building block approach

  -- Building upon existing EXT3, NFS, NTFS, CIFS infrastructures

- No metadata is replicated

  -- Superblock, Cylinder group, file allocation map are not replicated.

- Every file write operation is checked by file system

  -- file consistency, integrity

- Live file, not raw data replication

  -- The primary and backup copy both are live files

Advantages

- Interoperability

  -- Two nodes can be different systems

  -- Storage systems can be different

- Small granularity

  -- Directory level, not entire file system

- One to many or many to one replication

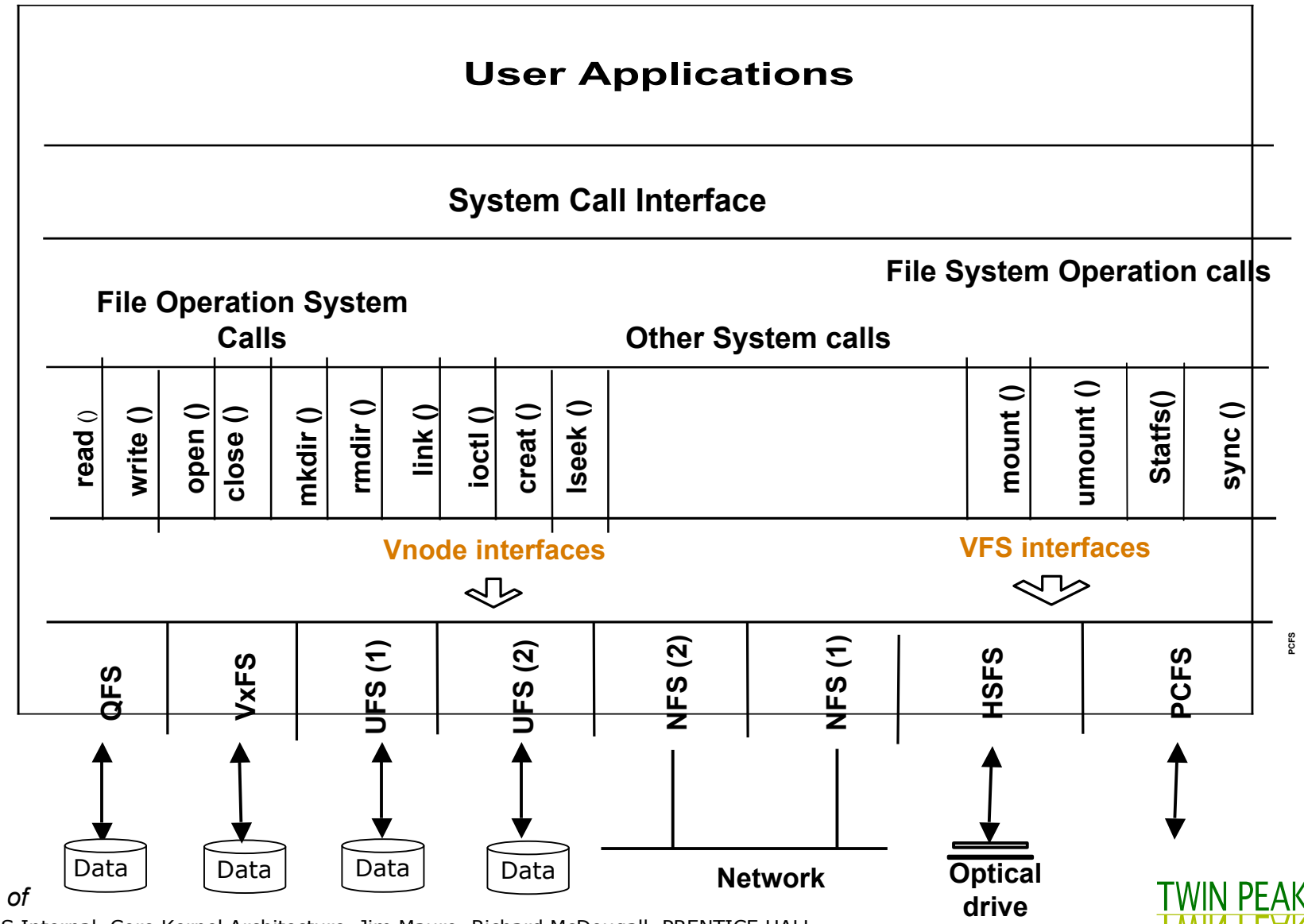
Advantages

- Fast replication
  -- Replication in Kernel file system module
- Immediate failover
  -- No need to fsck and mount operation
- Geographically dispersed clustering
  -- Two nodes can be separated by hundreds of miles
- Easy to deploy and manage
  -- Only one copy of MFS running on primary server is needed for replication

Why MFS?

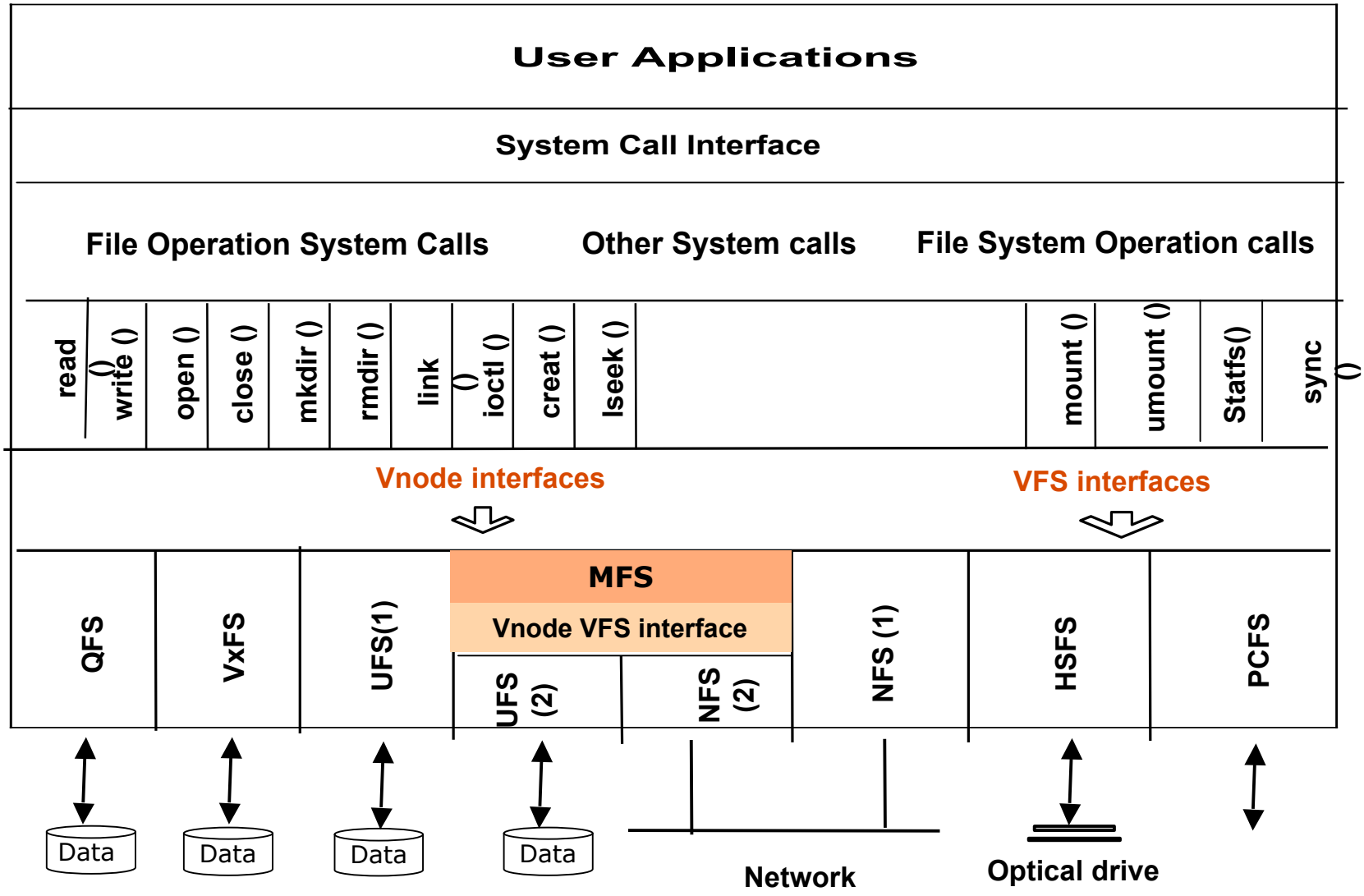
- Better Data Protection
- Better Disaster Recovery
- Better RAS
- Better Scalability
- Better Performance
- Better Resources Utilization

File System Framework



SOLARIS Internal, Core Kernel Architecture, Jim Mauro. Richard McDougall, PRENTICE HALL

MFS Framework



Transparency

- Transparent to users and applications
  - No re-compilation or re-link needed
- Transparent to existing file structures
  - Same pathname access
- Transparent to underlying file systems
  - UFS, NFS

Mount Mechanism

- Conventional Mount
  - One directory, one file system
- MFS Mount
  - One directory, two or more file systems

Mount Mechanism

mount –F mfs host:/ndir1/ndir2 /udir1/udir2

- First mount the NFS on a UFS directory
- Then mount the MFS on top of UFS and NFS
- Existing UFS tree structure /udir1/udir2 becomes a local copy of MFS
- Newly mounted host:/ndir1/ndir2 becomes a remote copy of MFS
- Same mount options as NFS except no ‘-o hard’ option

MFS mfsck Command

/usr/lib/fs/mfs/mfsck mfs_dir

- After MFS mount succeeds, the local copy may not be identical to the remote copy.
- Use mfsck (the MFS fsck) to synchronize them.
- The mfs_dir can be any directory under MFS mount point.
- Multiple mfsck commands can be invoked at the same time.

READ/WRITE Vnode Operation

- All VFS/vnode operations received by MFS
- READ related operation: read, getattr,….
  those operations only need to go to local copy (UFS).
- WRITE related operation: write, setattr,…..
  those operations go to both local (UFS) and remote (NFS) copy simultaneously (using threads)

Mirroring Granularity

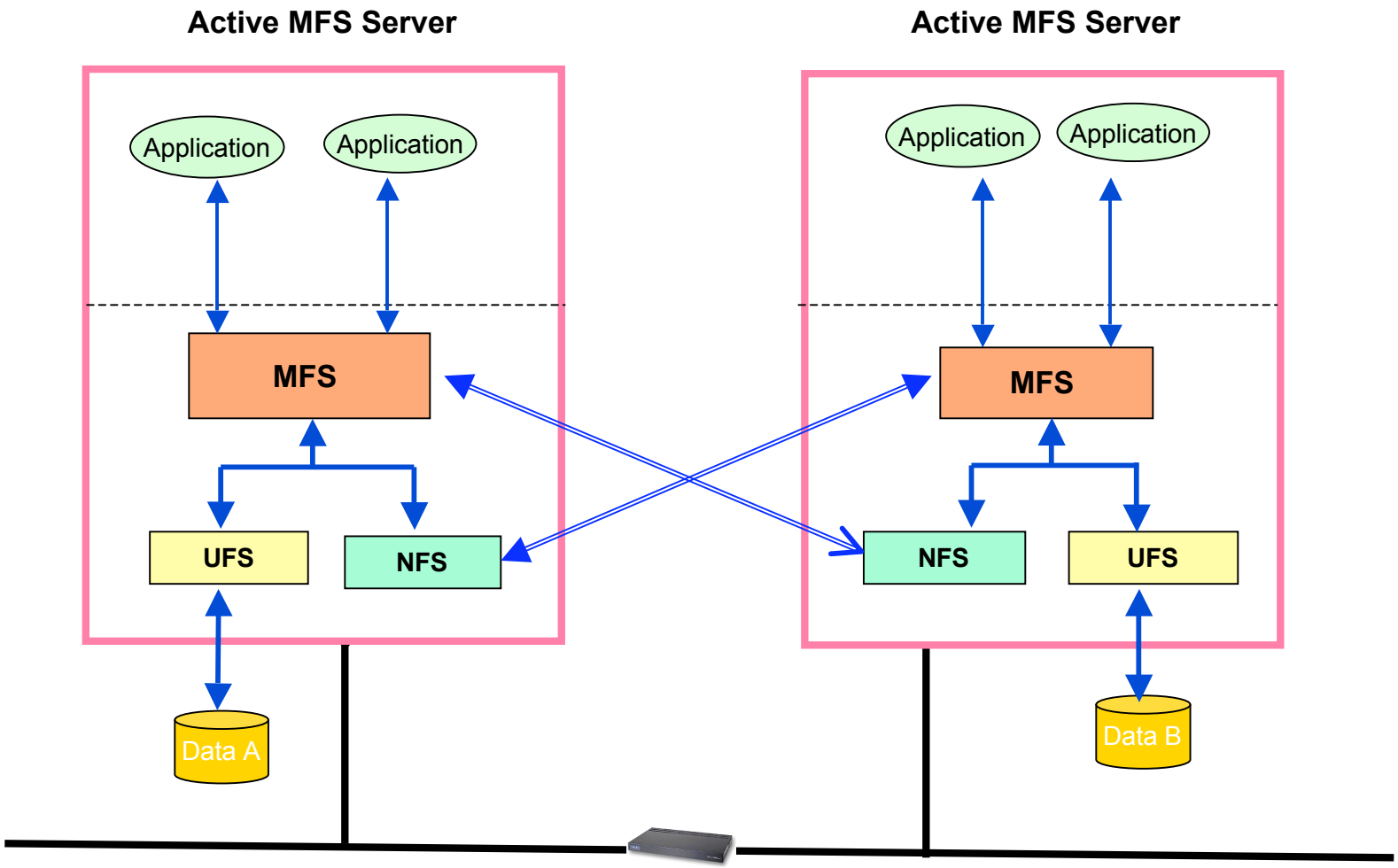
- Directory Level
  - Mirror any UFS directory instead of entire UFS file system
  - Directory A mirrored to Server A
  - Directory B mirrored to Server B
- Block Level Update
  - Only changed block is mirrored

MFS msync Command

/usr/lib/fs/mfs/msync mfs_root_dir

- A daemon that synchronizes MFS pair after a remote MFS partner fails.
- Upon a write failure, MFS:
  - Logs name of file to which the write operation failed
  - Starts a heartbeat thread to verify the remote MFS server is back online
- Once the remote MFS server is back online, msync uses the log to sync missing files to remote server.

Active/Active Configuration



MFS Locking Mechanism

MFS uses UFS, NFS file record lock.

Locking is required for the active-active configuration.

Locking enables write-related vnode operations as atomic operations.

Locking is enabled by default.

Locking is not necessary in active-passive configuration.

Real -Time and Scheduled

- Real-time

  -- Replicate file in real-time

- Scheduled

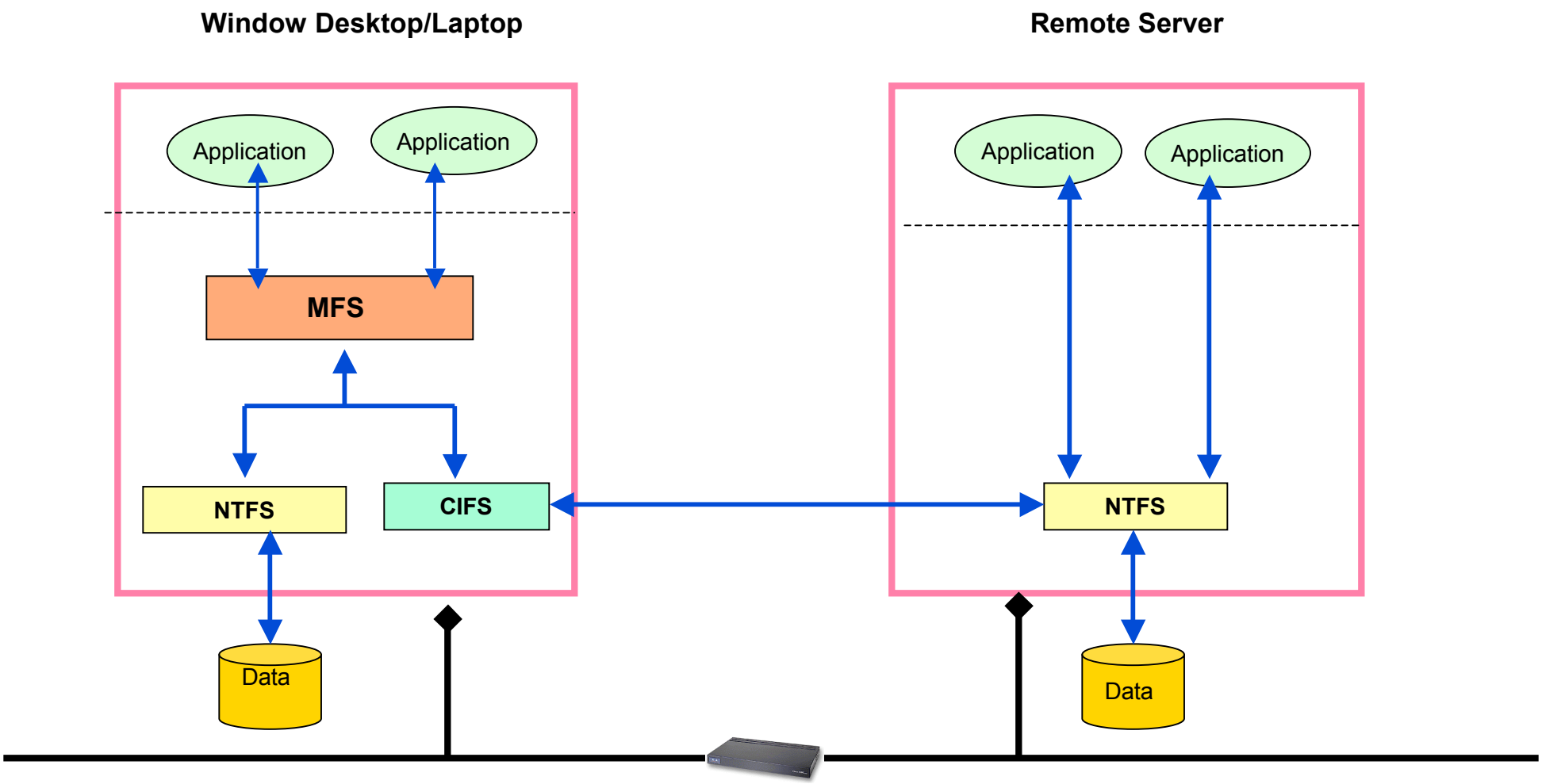
  -- Log file path, offset and size

  -- Replicate only changed portion of a file

Applications

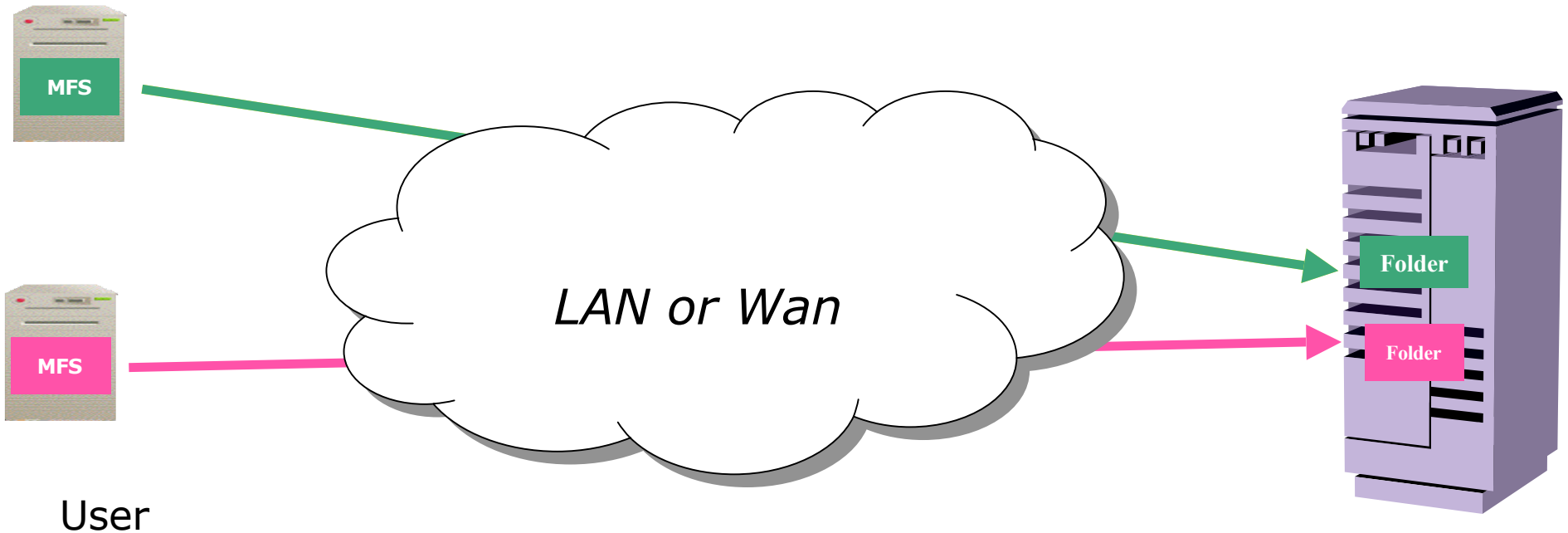
- Online File Backup
- Server File Backup, active → passive
- Server/NAS Clustering, active ←→ Active

MFS = NTFS + CIFS



Online File Backup

Real-time or Scheduled time

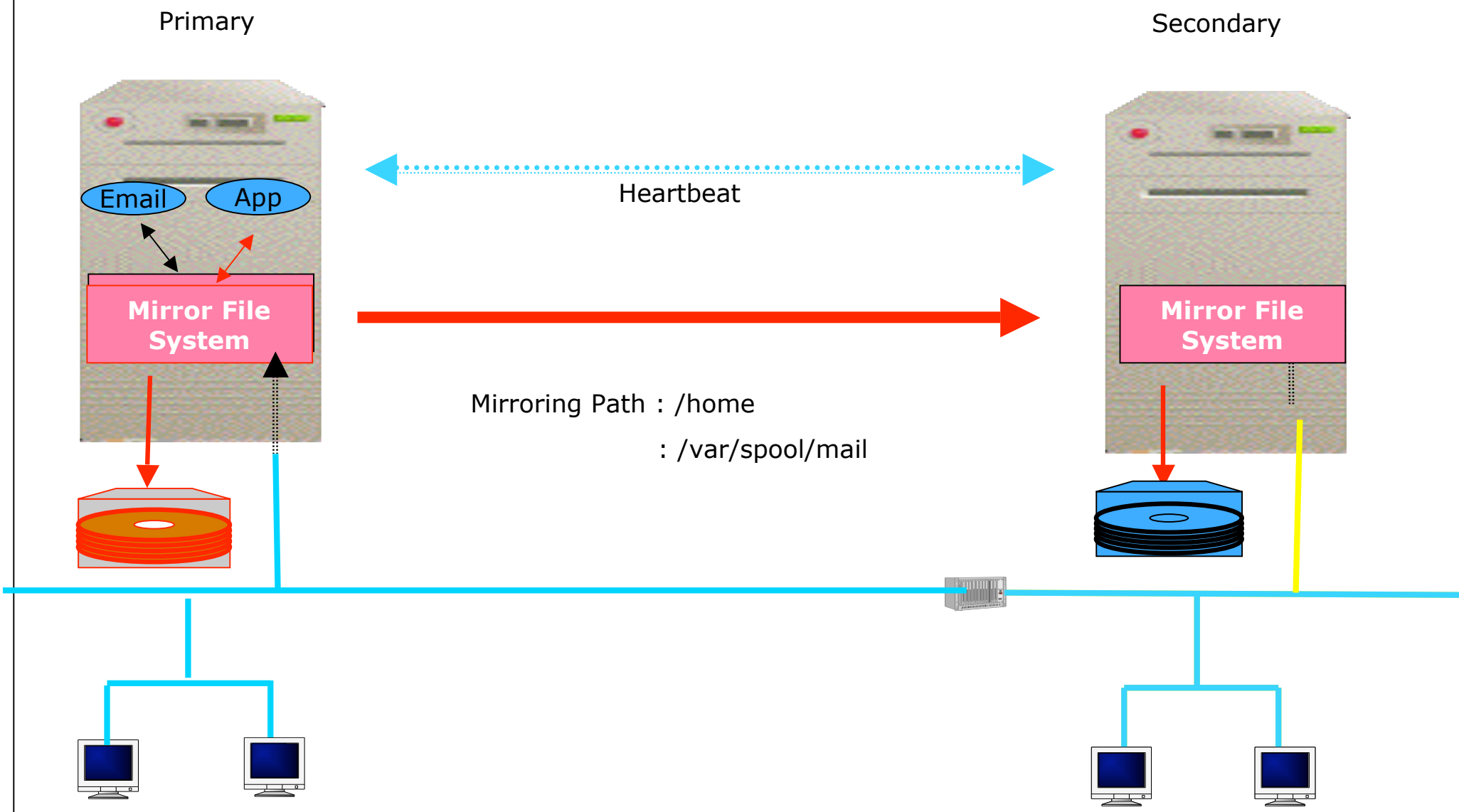


User
Desktop/Laptop

ISP Server

TWIN PEAKS software inc.

Server Replication



Enterprise Clusters