Cluster Shared Volume

Vladimir Petter
Principal Software Design Engineer
Microsoft

Topics

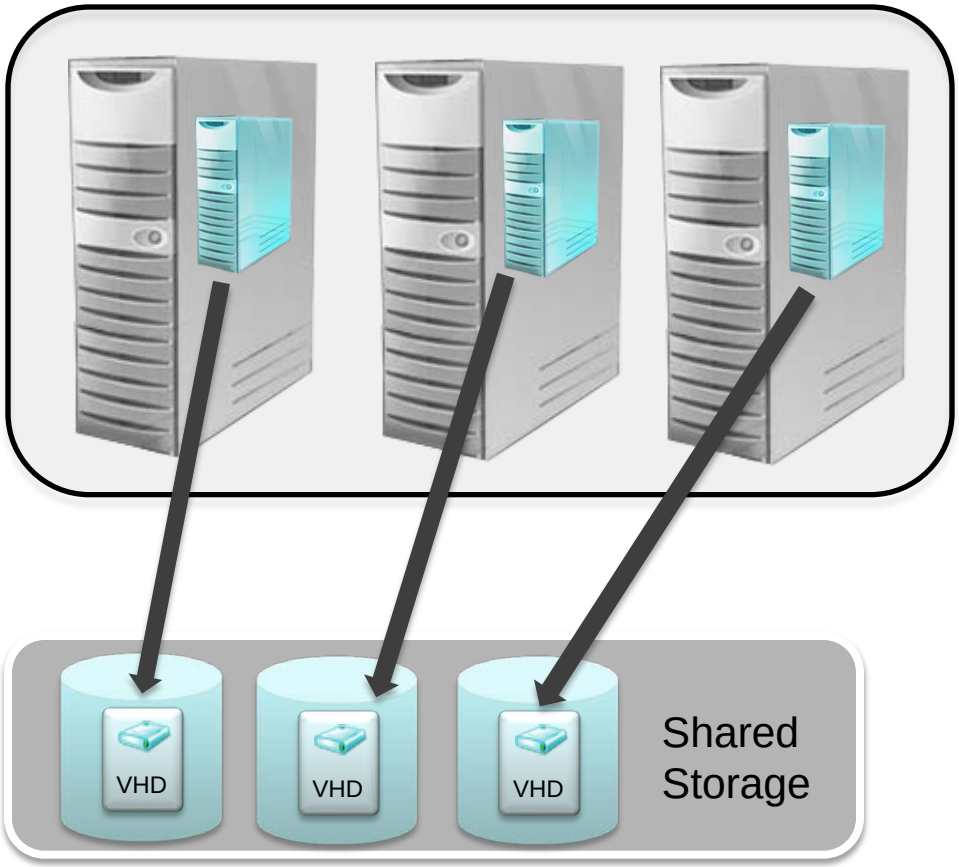
- CSV Requirements and motivation
- CSV Design
- CSV IO operations
- Scale Out File Server
- Developing For CSV

Topics

- CSV Requirements and motivation
- CSV Design
- CSV IO operations
- Scale Out File Server
- Developing For CSV

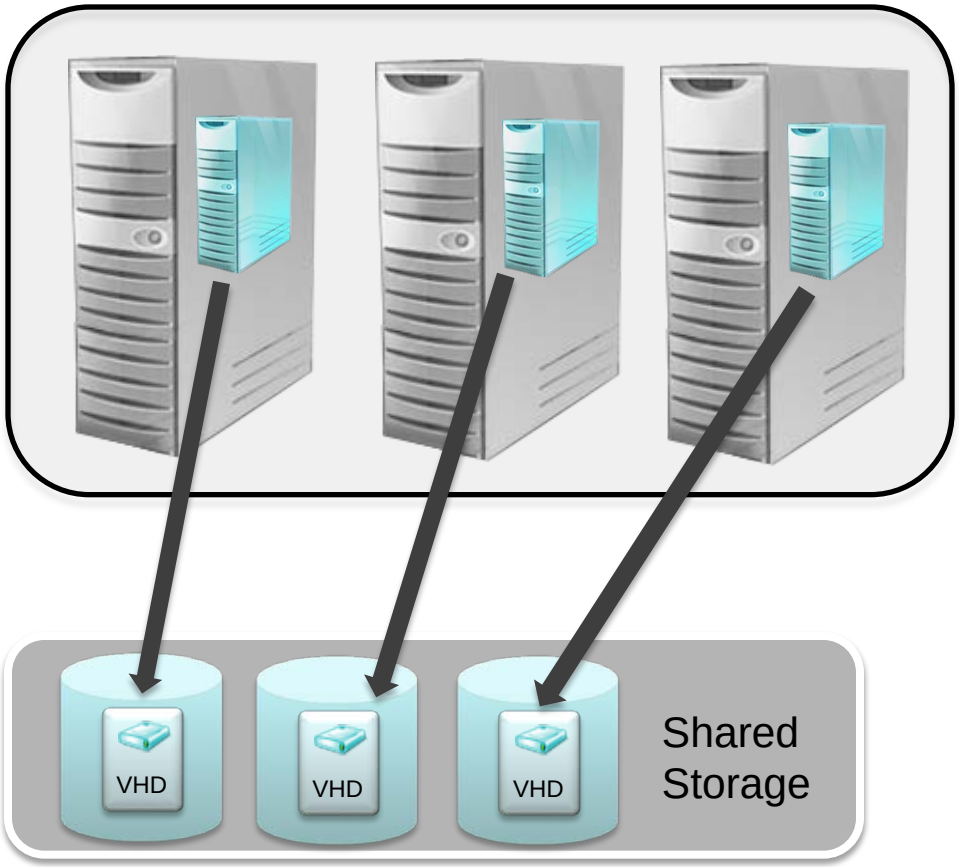
CSV Requirements. Why we did CSV in Windows 2008 R2?

- VM consolidation without scarifying performance while keeping management sane
  - We want to consolidate lots of VMs (1000s) in a cluster
  - We need to be able to move VM between nodes
  - To access VHD file VM required local file system
- We need a LUN per VM (1000 LUNs) or we need a Clustered File System
  - No in box solution that would provide shared LUN access.
  - SMB based NAS at that time was not considered a viable solution.
    - Has changed in Windows 2012. See slide # 18 - 20



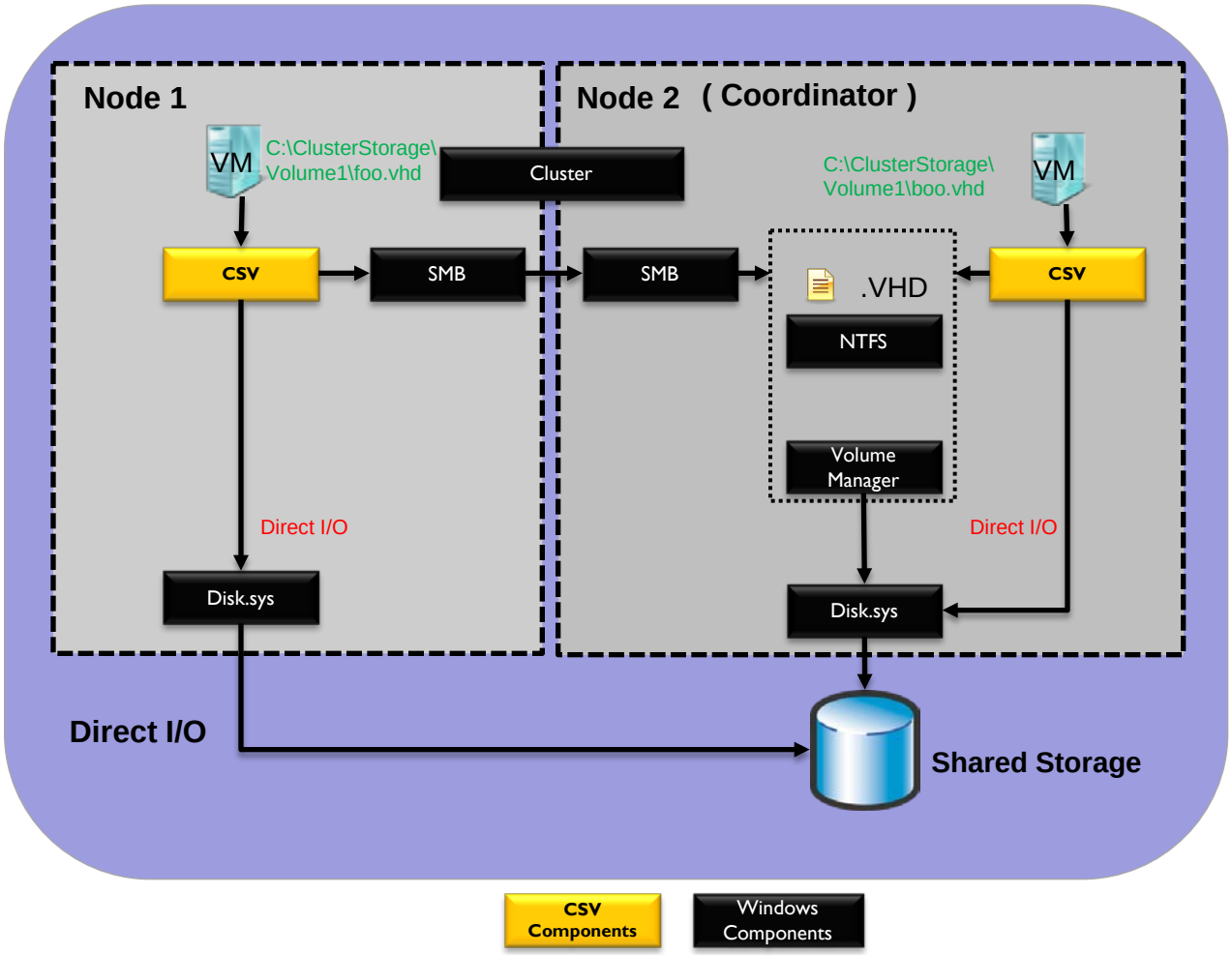
CSV Requirements. Characteristics of the Workloads

- Target workloads VM and SQL
  - Keeps file opened for a long time
  - Few metadata operations. Mostly reads and writes
  - When file is accessed for both read and write it is opened from one node
  - When files accessed from multiple nodes then read is dominant operation
    - Multiple VMs need to access the same Base VHD, and still can run on different cluster nodes.
  - Need to support tens of thousands of opened file
    - VDI scenario with up to 64 node cluster and hundreds VMs per node



CSV Requirements. What we've decided to build

- Design Objectives
  - Multiple workloads can access the same LUN
  - Build up on the current investments
    - NTFS; SMB; Clustering
  - Same path to the file from any node
  - Reads and writes need to go directly to the SAN whenever possible – "**Direct I/O**"
  - All other operations should go to NTFS on the node where NTFS is mounted.
- Performance Objectives
  - Direct IO performance should be as fast as NTFS
  - If we need to Redirect IO over network then it should be as fast as IO over SMB when file is opened for write-through.
- High Availability Objectives
  - Detect and hide storage, network and node failures.

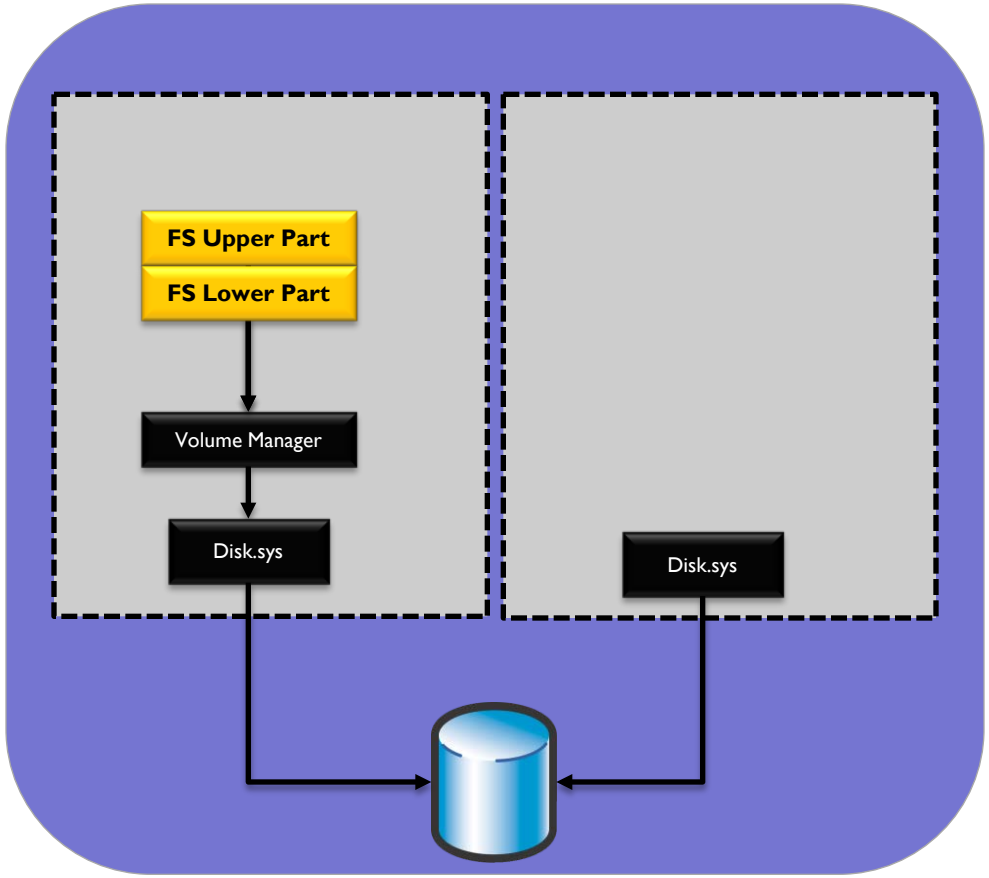


Topics

- CSV Requirements and motivation
- CSV Design
- CSV IO operations
- Scale Out File Server
- Developing For CSV

CSV Design Decisions
Local File System on Windows

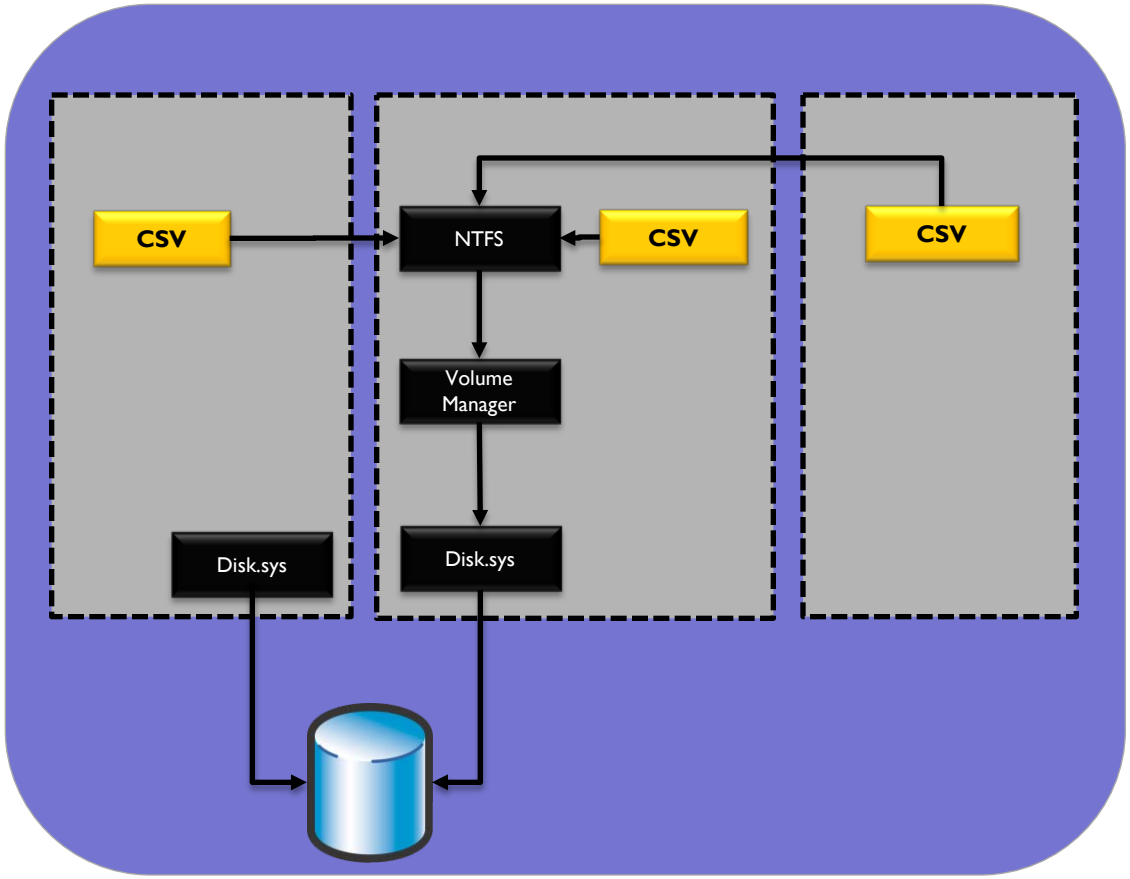
- Logically every local file system in the windows has
  - Upper part that is responsible for implementing File System interface. Applications have dependency on this interface so it is very application compatibility sensitive
  - Lower part that manages how File System lays out data on the disk, and how it writes data to the disk. Applications are not supposed to care how it works. It interfaces with volume for block read/write operations



- CSVFS does not introduce new on disk structure nor does it work directly with on disk layout except of one special case – Direct IO. Consequently CSVFS has only upper part.

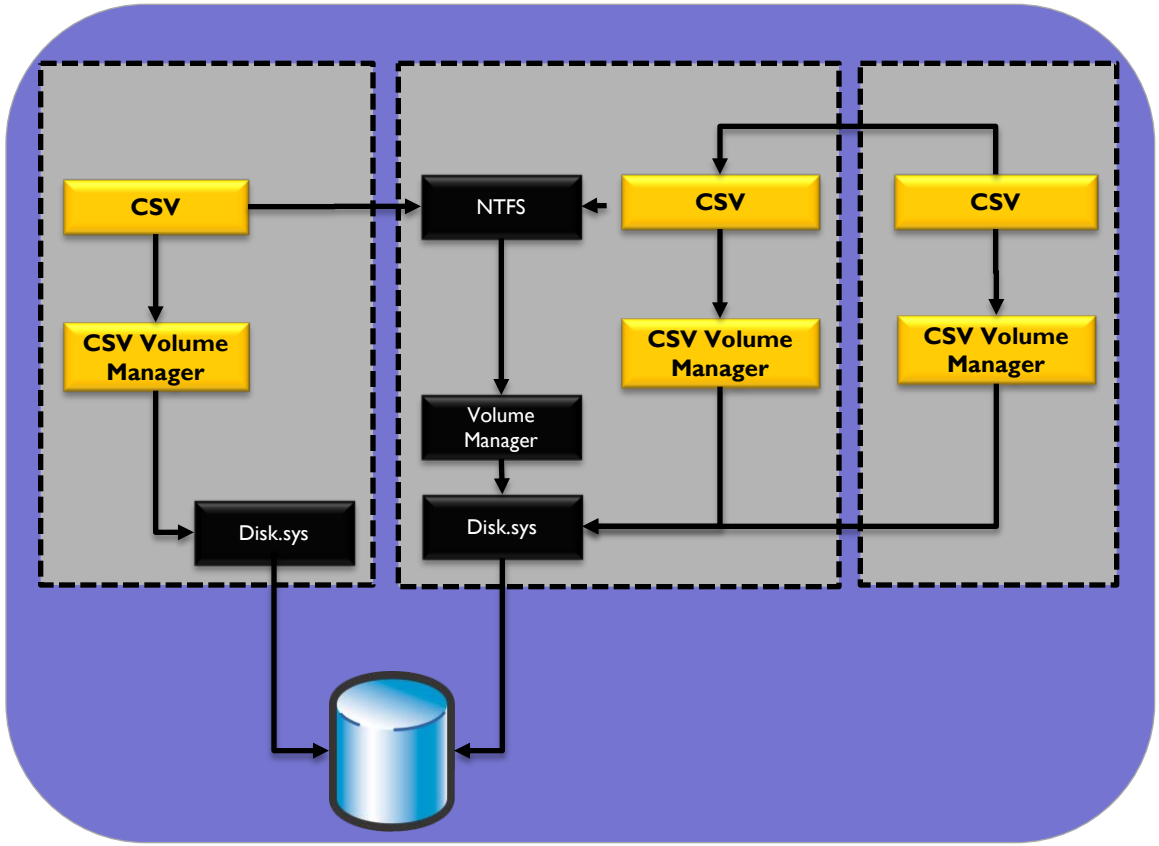
CSV Design Decisions
Remote vs Local File System

- It looks similar to the remote file system, and we almost went the direction of developing it as an extension for the SMB Redirectior
- We chose to implement it as a local file system because
  - It provides better compatibility with existing applications
  - It provides better management story with existing Disk Management tools
  - It provides better backup story with existing backup tools
  - Remote applications can access it
- Local File System needs a volume to mount on so we needed a Volume Manager



CSV Design Decisions
CSV Volume Manager

- CSV volume is created on every cluster node. You can access it using mountpoint like C:\CluserStorage\Volume1 from any cluster node.
- CSV Volume Manager is controlled by Cluster
- CSVFS mounts only on the volumes create by the CSV Volume Manager
- CSV Volume provides CSVFS lifetime that does not have dependency on the disk connectivity. Even on the node where disk is not connected CSVFS is present and accessible.
- CSV Volume is the object that you see in the disk management utilities.
- CSV Volume is what applications find when they enumerate volumes on the machine using Win32 APIs

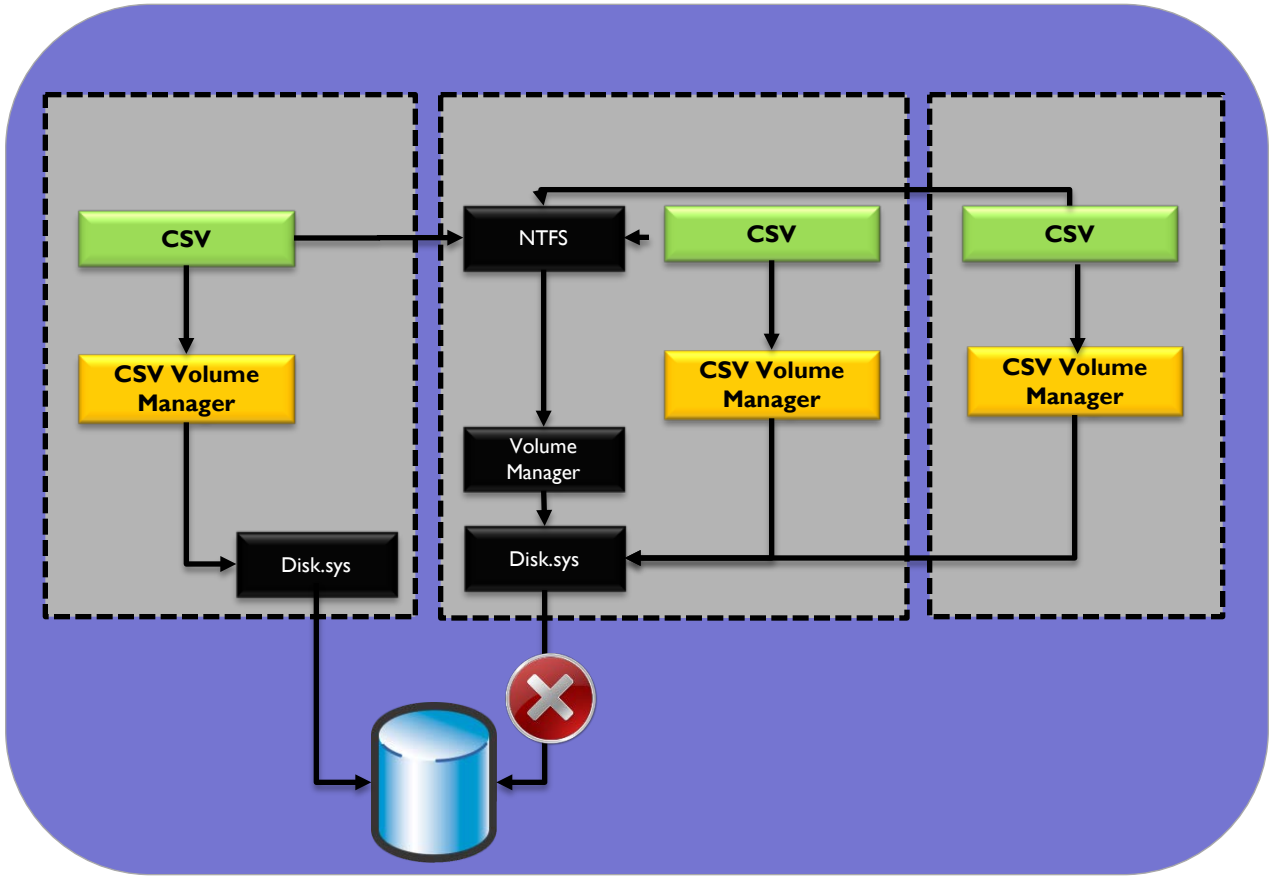


CSV Design Decisions
Recipe how to develop CSVFS

- Take FastFat from DDK
- Make it mountable on CSV Volume
- Remove lower part of FastFat that is managing on disk structure and replace it with routines that either forward IO to NTFS or perform Direct IO.
- Add support for bunch of calls that FastFat did not support to get parity with NTFS and REFS
- Secret ingredients:
  - Read several times File System Internals book by Rajeev Nagar
  - Heavily instrument the code with tracing so you can learn how it really works
- Optional
  - If you care about performance then redesign locking model
- Debug it for at least 3 to 5 years.

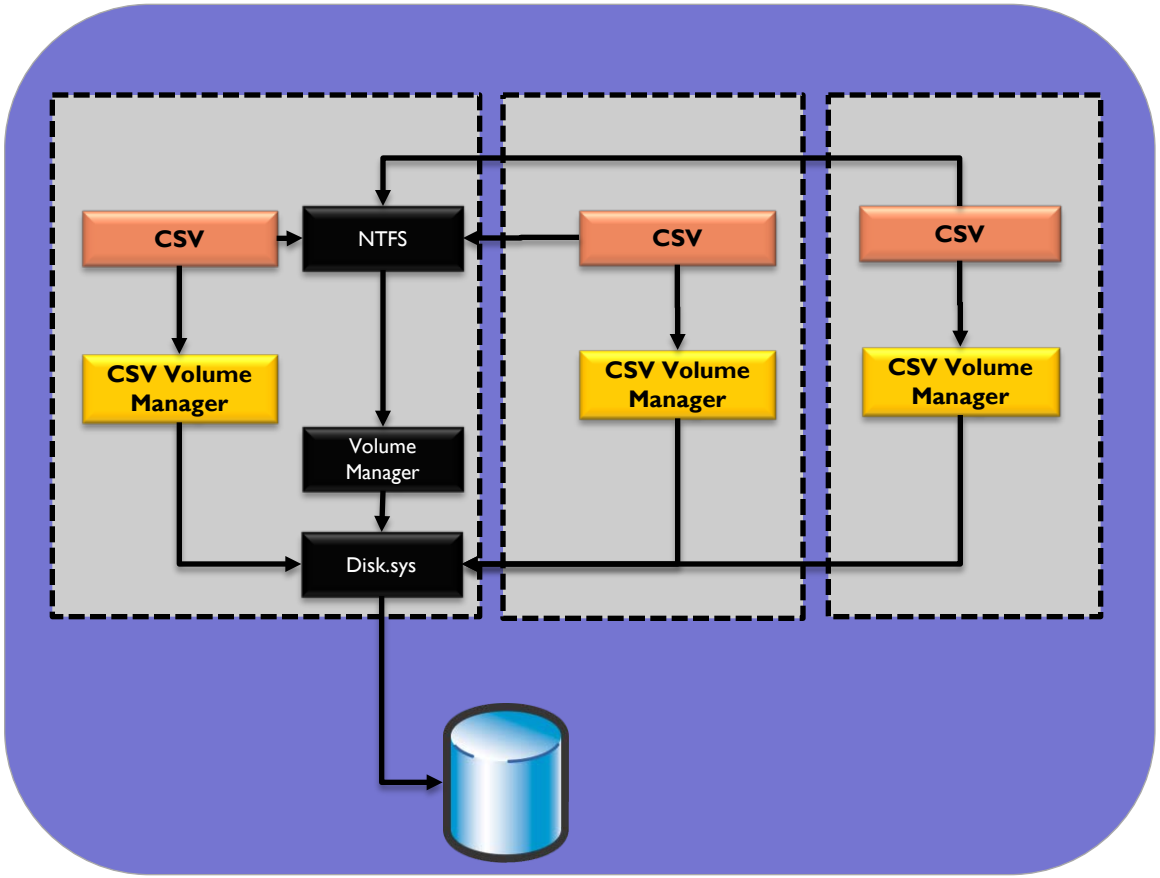
CSV Design Decisions
CSV Fault Tolerance. Failure

- Storage disconnects
- CSVFS on one of the nodes observes IO failure and tells cluster
- On every node cluster tells CSVFS volume to start draining
  - In draining state CSVFS pends new IO and any failing IO
- On every node cluster tells CSVFS volume to pause
  - CSVFS cancels ongoing IO and waits for completion of all IOs
  - Once all IO completed CSVFS closes its internal files opens on NTFS
- Cluster will observe disk failure
- Cluster will tear down volume stack



CSV Design Decisions
CSV Fault Tolerance. Recovery

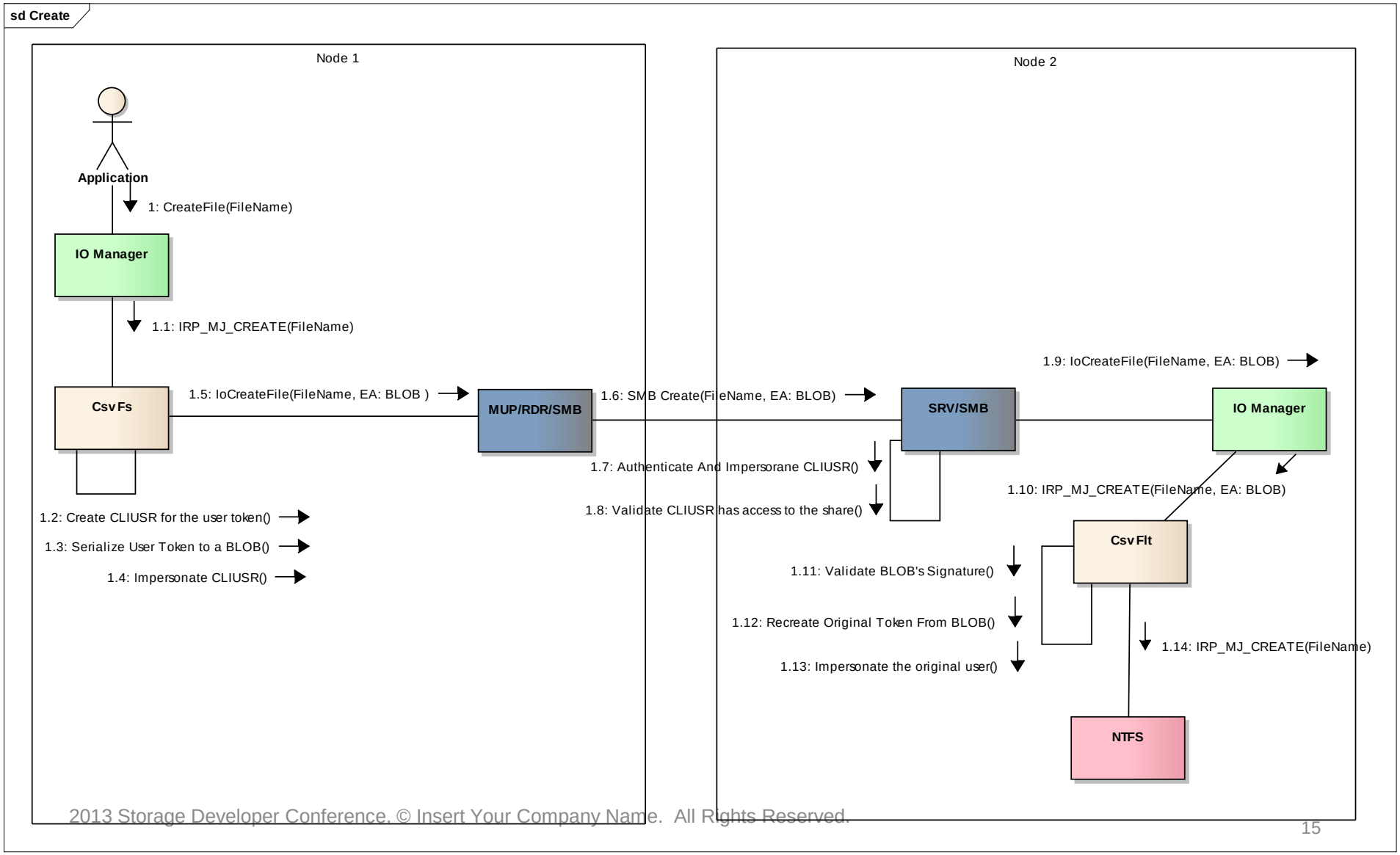
- Application still have files opened on CSVFS and are not aware of the failure
- Cluster finds a node where disk is still connected and mounts NTFS on that node
- On every node cluster tells CSVFS to reopen its internal handles on NTFS
- On every node cluster tells CSVFS to resume IO.
  - CSVFS reissues all paused IO and stop pending any new IOs



Topics

- CSV Requirements and motivation
- CSV Design
- CSV IO operations
- Scale Out File Server
- Developing For CSV

CSV Operation. File Open



CSV Operations
CSV Uses Oplocks for Cache Coherency

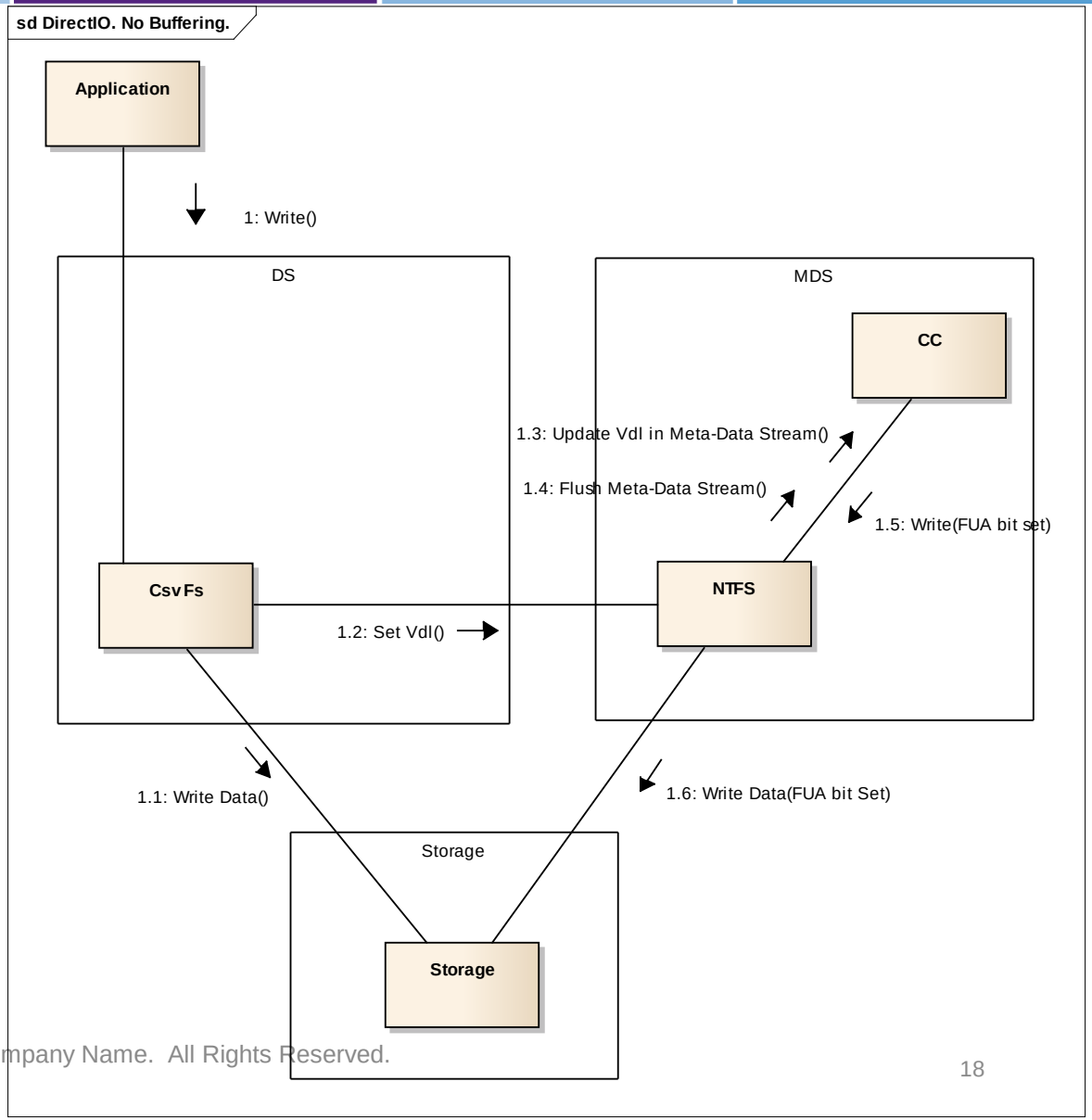
- Oplocks existed in Windows for a long time and are used by SMB clients for cache coherency
- Like SMB Client, CSV uses oplocks for cache coherency
- CSV also uses oplocks to decide when it is safe to perform Direct IO.
  - We can perform Direct IO on read if we have a read containing oplock level (RWH, RW, RH or R)
  - We can perform Direct IO on write if we have a write containing oplock level (RWH or RW)
- A write containing oplock is granted only if file is opened from single node
  - When another node opens file, the open operation will cause revoke of write oplock level. Open will be held until the client that lost oplock acknowledges oplock break
- On files opened from multiple nodes CSV can get read containing oplock
  - Opertions that modify data cause read oplock break

CSV Scenarios.
When Direct IO is Possible?

- We understand on disk file format
- There are no File System filters that might modify file layout
- There are no File System filters that object against Direct IO on the steram
- We were able to make sure NTFS will not change location of the file data on the volume
- No applications that need to make sure IO is observed by NTFS stack
- We have oplocks. Cross node cache coherency.
  - RWH or RH or RW or R for reads
  - RWH or RW for write
- We we able to purge cache on NTFS. Make sure there is no stale cache on NTFS.

CSV Operations Unbuffered Write

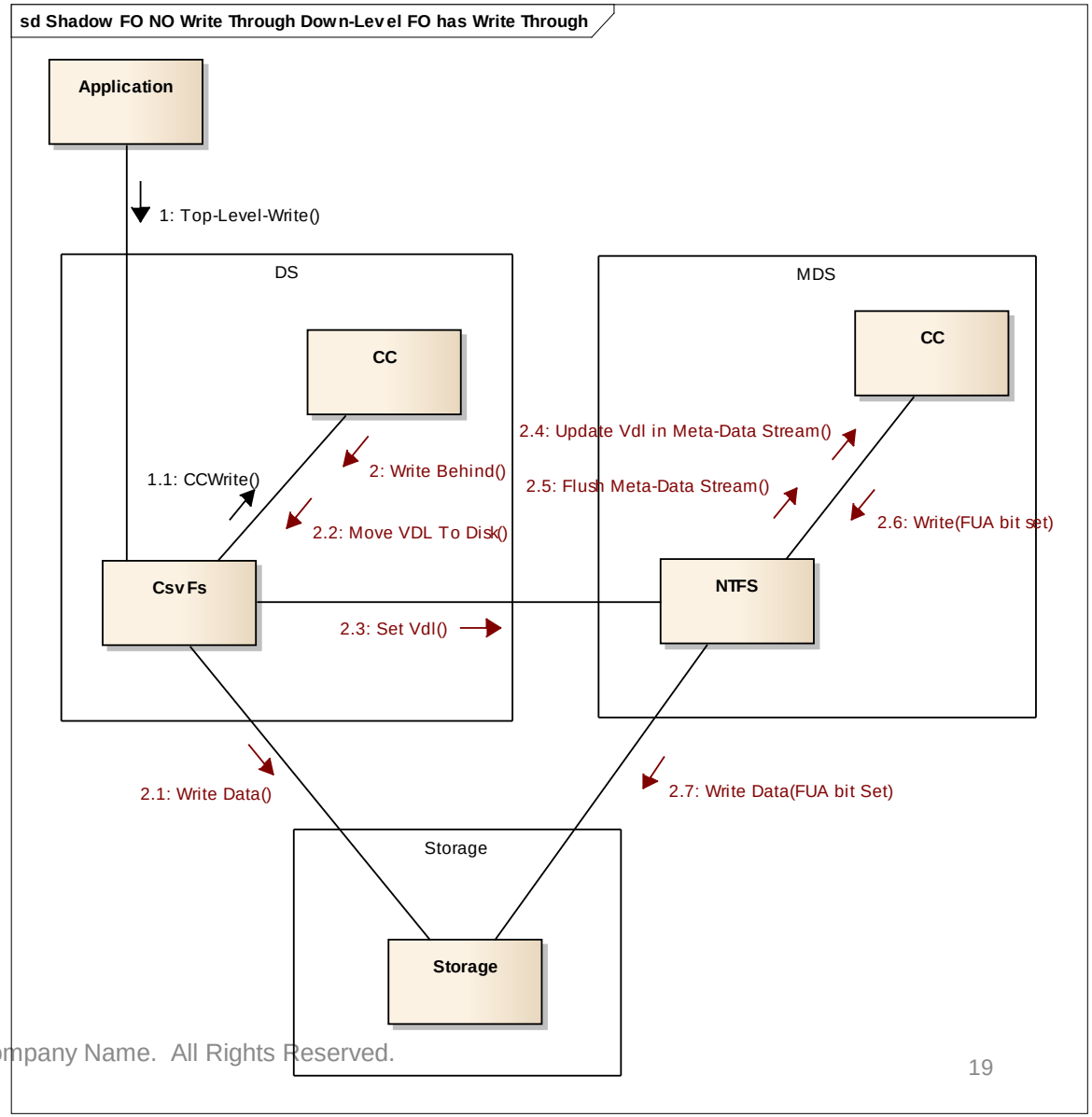
- Data are written to the disk using Direct IO
- If write is extending ValidDataLength then we also update VDL on NTFS



CC – Windows Cache Manager

CSV Operations
Buffered Write

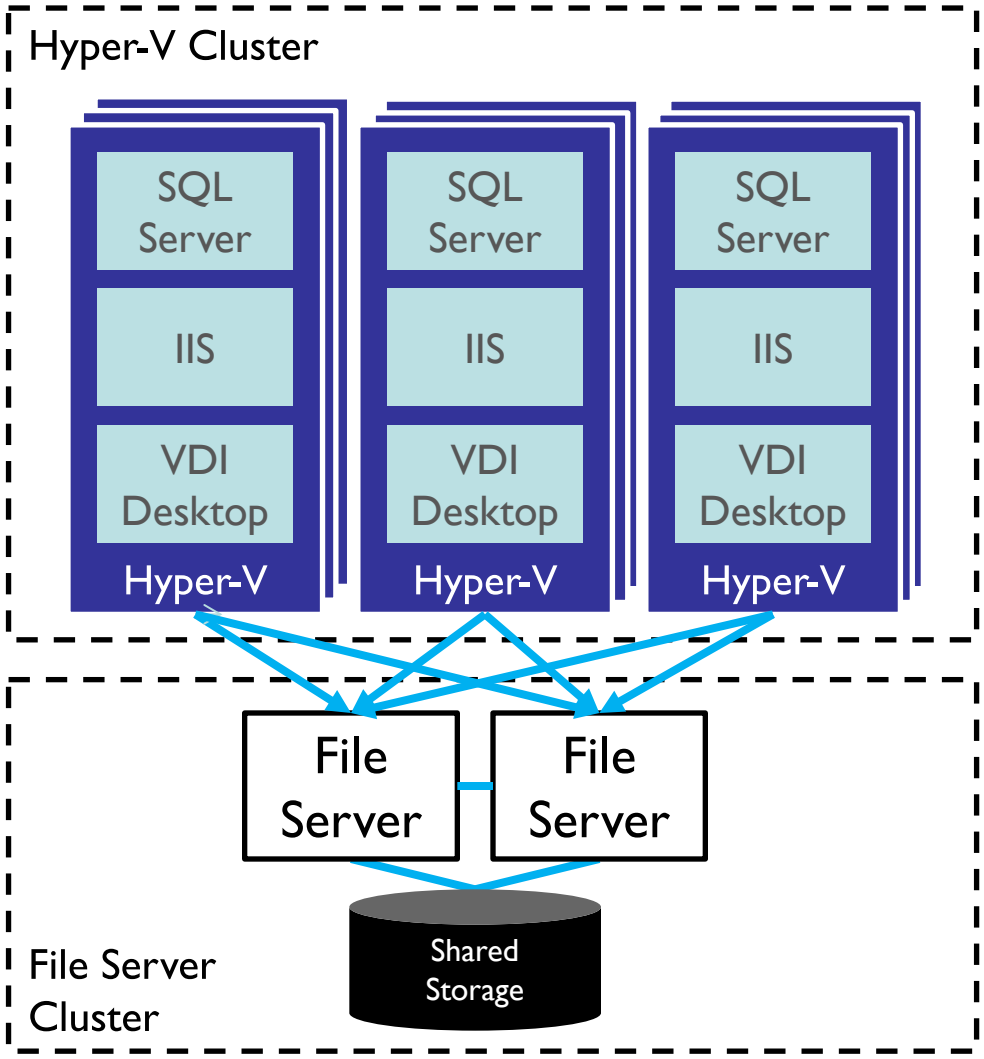
- Data are written to the disk using Direct IO
- If write is extending ValidDataLength then we also update VDL on NTFS



CC – Windows Cache Manager

CSV – Scale Out File Server

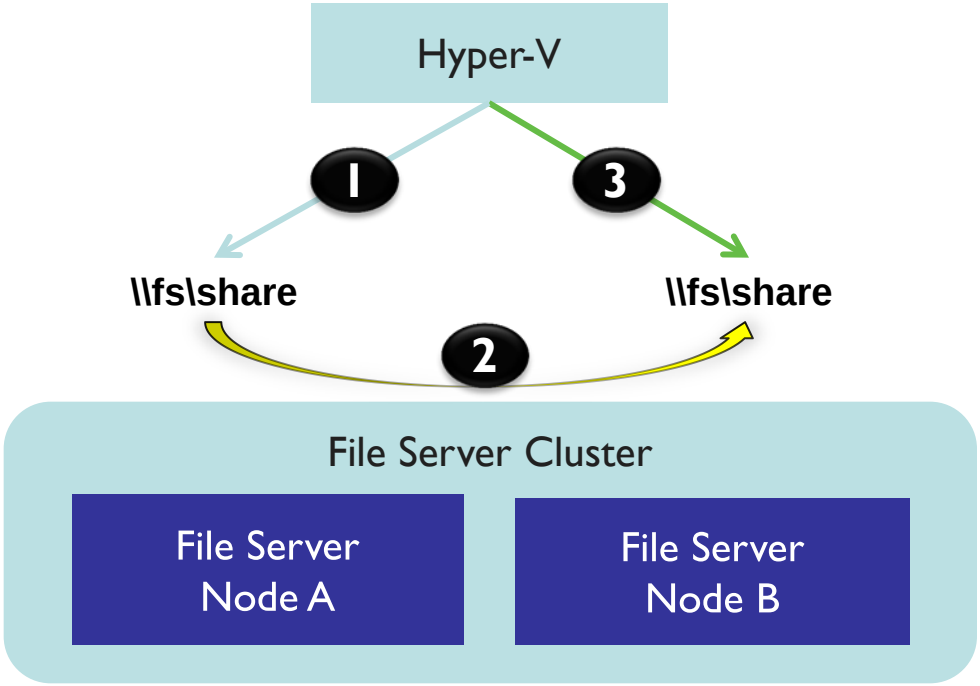
- What is it?
  - Store Hyper-V files in shares over the SMB 3.0 protocol
    (including VM configuration, VHD files, snapshots)
- Highlights
  - Increases flexibility
  - Eases provisioning, management and migration
  - Leverages converged network
  - Reduces capital and operational expenses
- Supporting Features
  - SMB Transparent Failover - Continuous availability
  - SMB Scale-Out – Active/Active file server clusters
  - SMB Direct (SMB over RDMA) - Low latency, low CPU use
  - SMB Multichannel – Network throughput and failover
  - SMB Encryption - Security
  - VSS for SMB File Shares - Backup and restore
  - SMB PowerShell and VMM Support - Manageability



SMB Transparent Failover

- Failover transparent to server application
  - Zero downtime – small IO delay during failover
- Supports planned and unplanned failovers
  - Hardware/Software Maintenance
  - Hardware/Software Failures
  - Load Rebalancing
- Resilient for both file and directory operations
- Requires:
  - File Servers configured as Windows Failover Cluster
  - Windows Server 2012 on both the servers running the application and file server cluster nodes
  - Shares enabled for “continuous availability” (default configuration for clustered file shares)
  - Works for both classic file server clusters (cluster disks) and scale-out file server clusters (CSV)

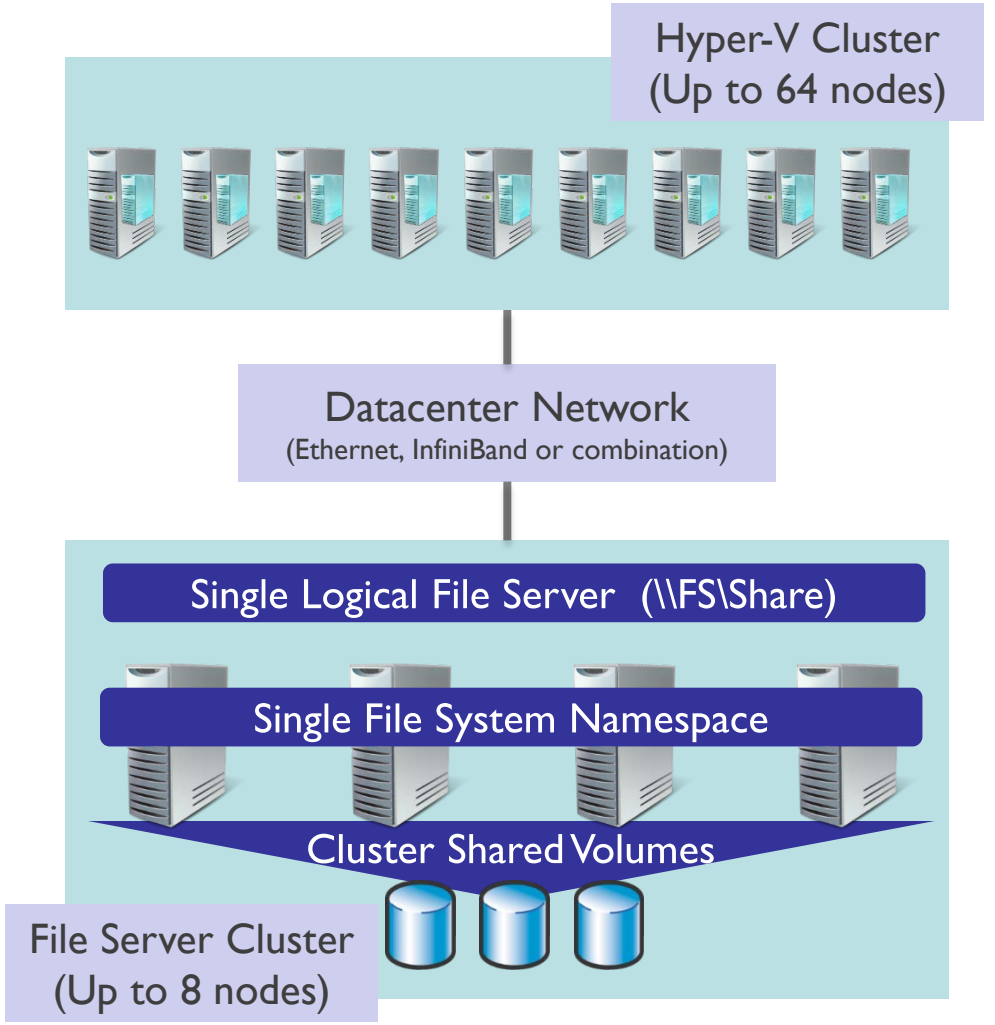
1. **Normal operation**
2. **Failover share - connections and handles lost, temporary stall of IO**
3. **Connections and handles auto-recovered Application IO continues with no errors**



SMB Scale-Out

- Targeted for server app storage
- Example: Hyper-V and SQL Server
- Increase available bandwidth by adding nodes
- Leverages Cluster Shared Volumes (CSV)

- Key capabilities:
- Active/Active file shares
- Fault tolerance with zero downtime
- Fast failure recovery
- CHKDSK with zero downtime
- Support for app consistent snapshots
- Support for RDMA enabled networks
- Optimization for server apps
- Simple management

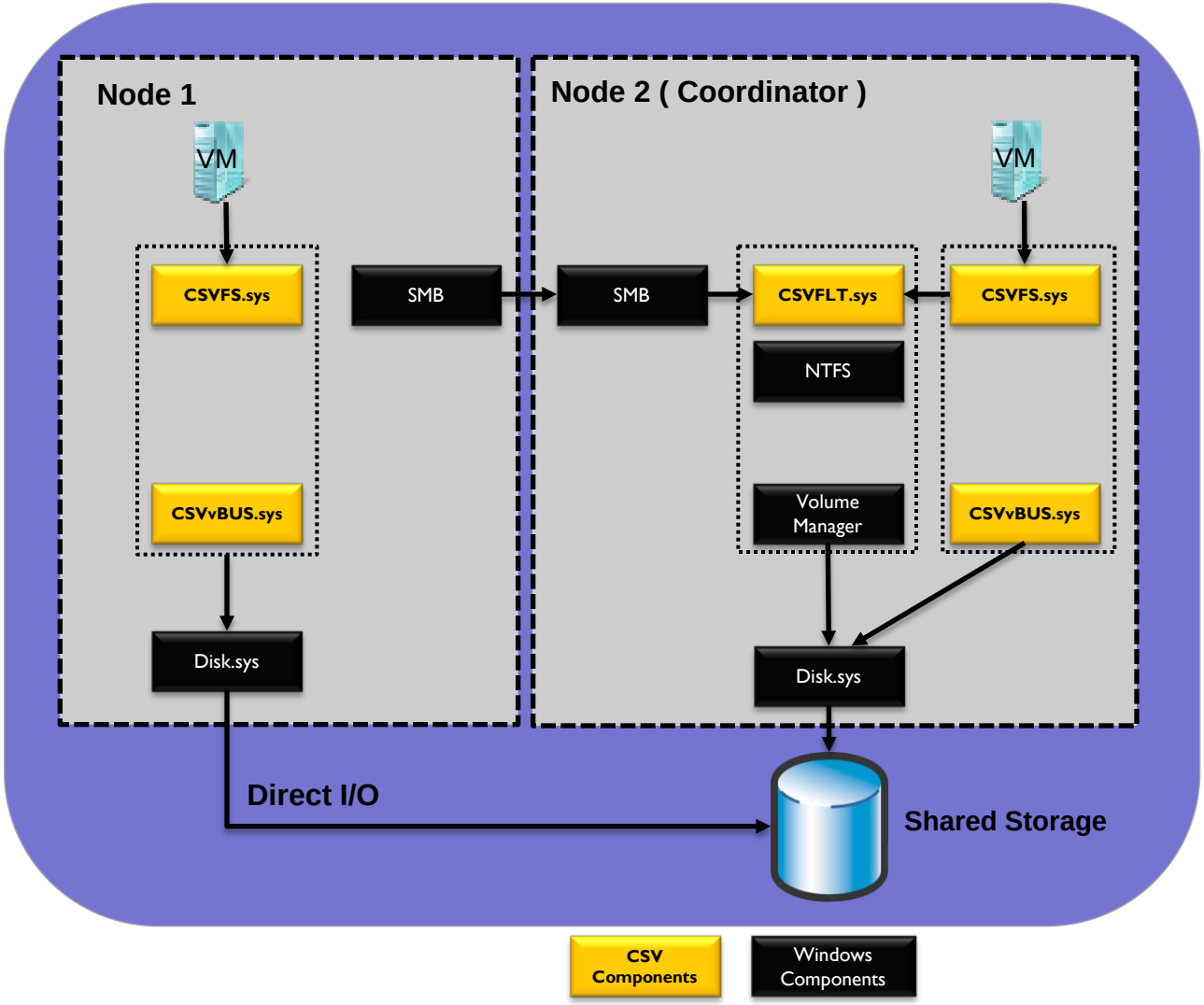


Topics

- CSV Requirements and motivation
- CSV Design
- CSV IO operations
- Scale Out File Server
- Developing For CSV

CSV Component Overview

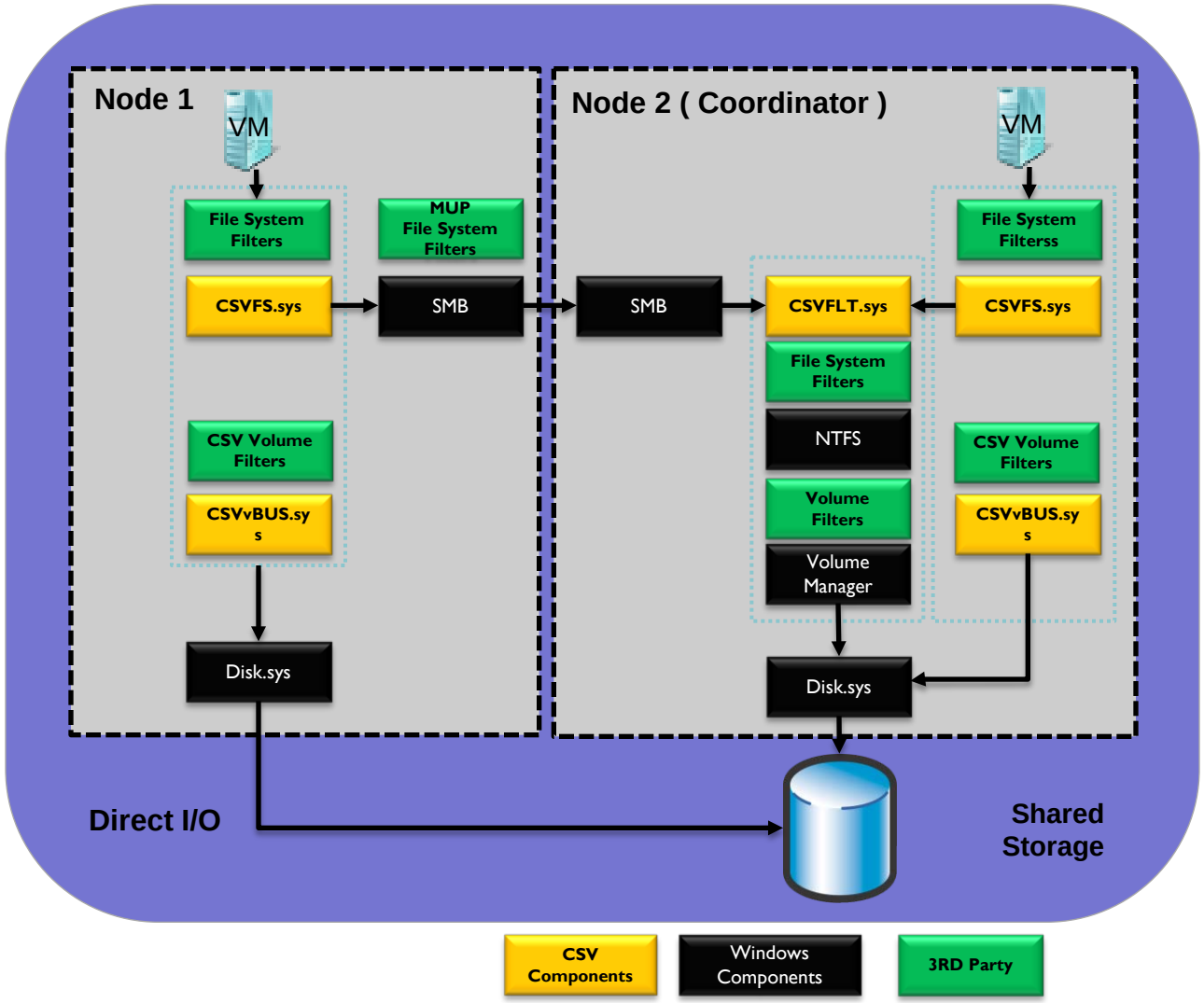
- CSV Filter Driver (CSVFLT.sys)
  - Mounted on Metadata Coordinator Node
  - Blocks access to the NTFS file system
  - Co-ordinates metadata operations over SMB
  - Filter Altitude - 404800
- CSV Proxy File System (CSVFS.sys)
  - Proxy file system on top of an underlying NTFS file system
  - Mounted on every node including Coordinator
  - Performed Direct I/O to the physical disk.
- CSV Volume Manager (CSVvBUS.sys)
  - Responsible for CSV pseudo/virtual volumes
  - Block-level IO redirector



CSV – Filtering Options

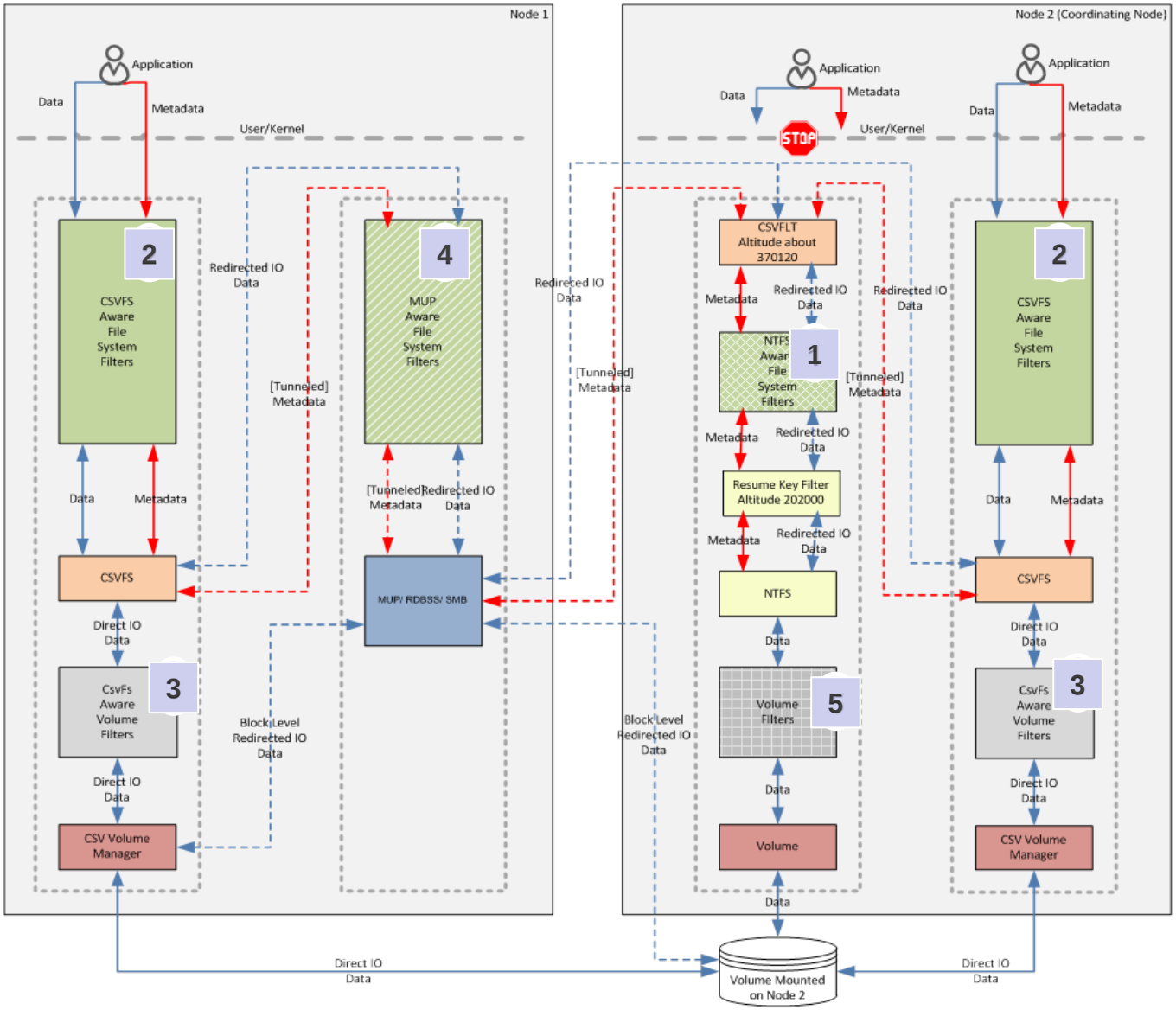
Different options available to Filter Drivers

- File system filters:
  - Attach on top of CSVFS.sys
  - Attach to NTFS
  - Attach to SMB
- Volume filters:
  - Filters attach to CSVvBus.sys
  - Filters attach to Volmgr.sys
- Attaching legacy filters to NTFS stack
  - CSV safeguards with Redirect IO mode
- If attaching to MUP ignore CSV traffic to the coordinator node
  - Extended create parameters



CSV – Filtering Options

Different Options Available to Filter Drivers



1. NTFS Aware File System Filters
2. CSVFS Aware FS Filters
3. CSVFS Aware Volume Filter
4. MUP Aware Volume Filter
5. Traditional Volume Filters

Considerations for Filter Drivers Attached to NTFS

- NTFS hidden and locked for user mode access
  - Mount Manager will not assign volume GUID
    - Use Disk Number and Partition Number instead to identify instance of you filter on a stack
- If anti-virus avoid attaching to the NTFS to prevent double scanning
  - Use IOCTL_DISK_GET_CLUSTER_INFO to detect CSV Metadata Stack
  - Avoid attaching to NTFS stack If output has IsCsv bit set
- Filters should not take a long time to process IO
  - Will be subject of 60 seconds timeout from RDR
  - 2 minutes timeout from CsvFs on volume state transitions.
  - IO cannot be pended for indefinite time by the filter on NTFS.
  - Violation
    - CSV volume in extra pause/resume transitions,
    - Volume invalidation
    - Failure of workloads

Considerations for Filter Drivers attached to NTFS

- Do not modify file sizes while attached to NTFS
  - (Allocation size, File size or Valid Data length while below NTFS).
  - Violation - volume corruption and file invalidation by CsvFs.
- Avoid causing files to get unpinned or to cause allocated blocks to get moved.
  - Violation - volume corruption and file invalidation by CsvFs.
- Avoid building memory sections.
  - Violation - stale cache and stream corruption.

Considerations for Filter Drivers Unsupported CSV Functionalities

- Enabling Compression on directory
- Enabling NTFS File level encryption
  - Bitlocker on CSVFS is supported
- Transactions on CSVFS
- Name grafting reparse points points on CSVFS volumes (mountpoints and symbolic links)
- Defrag files while in CSV Direct IO mode
- Direct IO on a Sparse or Compressed file; Redirected I/O used instead
- DASD IO in File System Redirect-IO mode