Building Windows File Systems: A Look at the OpenAFS Client for Windows

Peter Scott
Kernel Drivers, LLC
Jeffrey Altman
Your File System, Inc

AFS History … on Windows

- The SMB Interface
  - Used the Loopback Adapter for NetBIOS name registration
  - All requests were processed by the AFS Service in user mode
    - Minimal kernel level caching
  - Lack of support for network browsing within Explorer

More History …

- 'Relatively' easy to implement but problems …
  - Performance
    - Generic solution to fit all situations
  - Microsoft components, very difficult to get bugs fixed
    - Several critical bugs found in SMB with no fix in sight
      - Ex: Static thread worker pool count leads to dead locks
  - Documentation is minimal, at best
- Reliant on the Windows redirector for network behavior … good and bad

A Native Approach …

- Custom File System Pros …
  - Complete control of name space parsing
    - Reparse Points
    - Mount Points
    - DFS Links
  - Better optimization of IO pathways
    - Minimize Kernel to User transitions
    - Cache invalidation requirements
    - Better system cache integration
      - Support for write-through behavior when needed

A Native Approach …

- And the down side …
  - Complex implementation
  - Undocumented interfaces
    - srvsvc, wkssvc, IPC$
    - Network Provider support
    - Cache and Memory Manager
  - Must deal with 3rd party products
    - File system filters

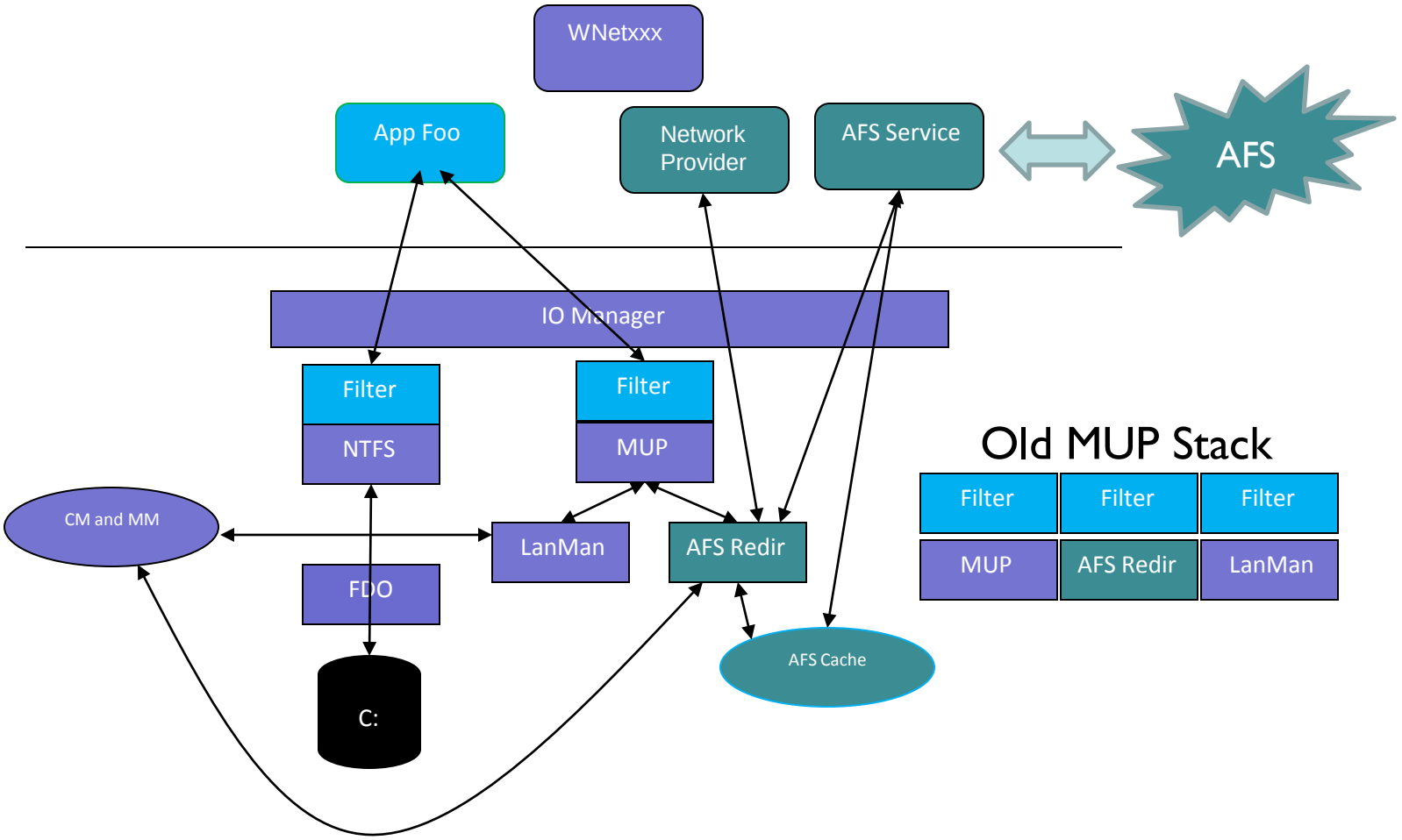
Windows File System Model

- Windows IFS (Installable File System) Interface
  - IRP Based model
    - IO Request Packets
  - 'Fast IO' Interface used for more than just IO
    - Direct-call model, no IRPs … almost
  - Network Provider Interface for Network Redirectors only
    - Support for drive mappings
    - UNC Browsing

Windows File System Model

- A Network File System on Windows
  - MUP (Multiple UNC Provider) Registration
    - Pre-Vista uses different model
  - IOCTL_REDIR_QUERY_PATH(_EX)
    - \\AFS\Your-File-System.com\User\PScott
  - Path Parsing
    - \Device\MUP;AFS\Redirector\;C:\\AFS\Your-File-System.com\User\PScott
      - \;C:\\AFS\Your-File-System.com\User\PScott
    - \Device\MUP\AFS\Your-File-System.com\User\PScott
      - \AFS\Your-File-System.com\User\PScott

Windows Internals



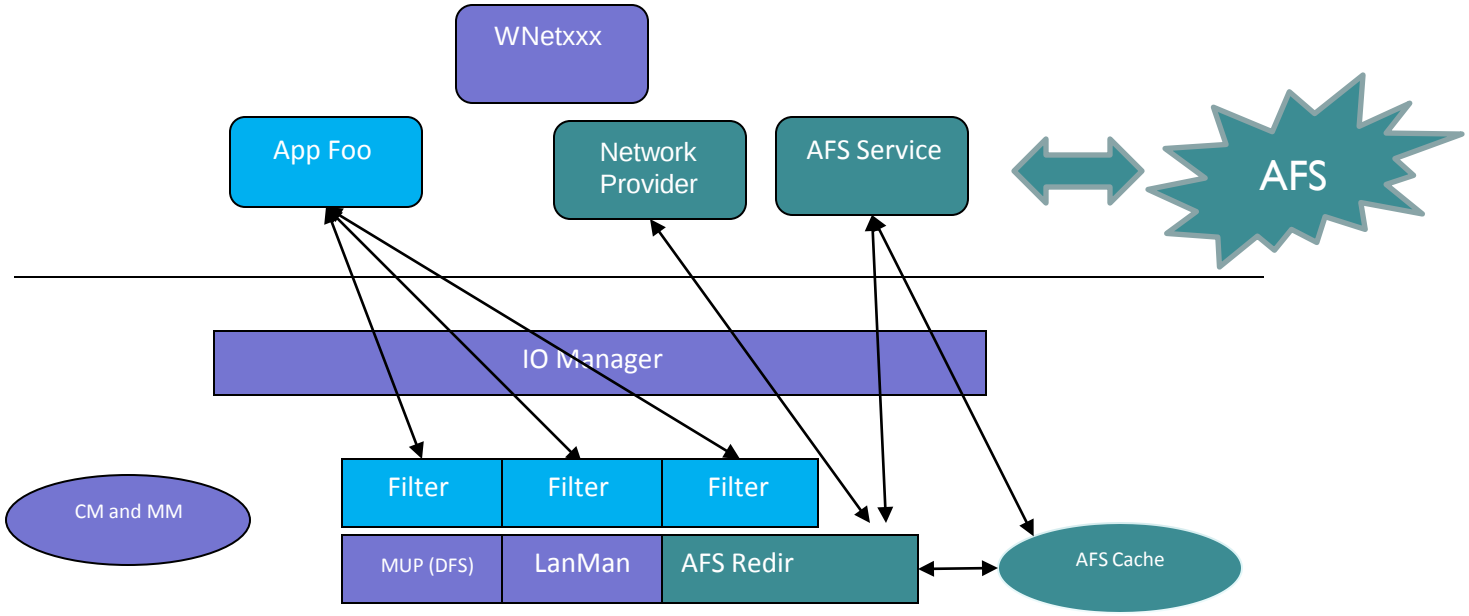
Windows Internals

- Windows Vista and 7 Changes
  - Memory Manger and Cache Manager changes
    - Theoretical limit of 4GB paging IO requests but have not seen anything larger than 256MB
      - Pre-Vista had a maximum of 64KB
  - MUP Changes
  - Tons of new 'features' – Bitlocker, built in AV, Indexer, Single Instance Storage, etc.

Multiple UNC Provider - MUP

- MUP – Handles mappings between the UNC namespace and the file systems which manage them
- MUP changes in Windows Vista and 7
  - Old model (pre-Vista)
    - Register with MUP using a named device object
    - Prefix resolution and IRP_MJ_CREATE requests handled by MUP, all others sent to file system
  - New Model
    - Register with MUP using an unnamed device object and a name of the file system control device

Old MUP Interface

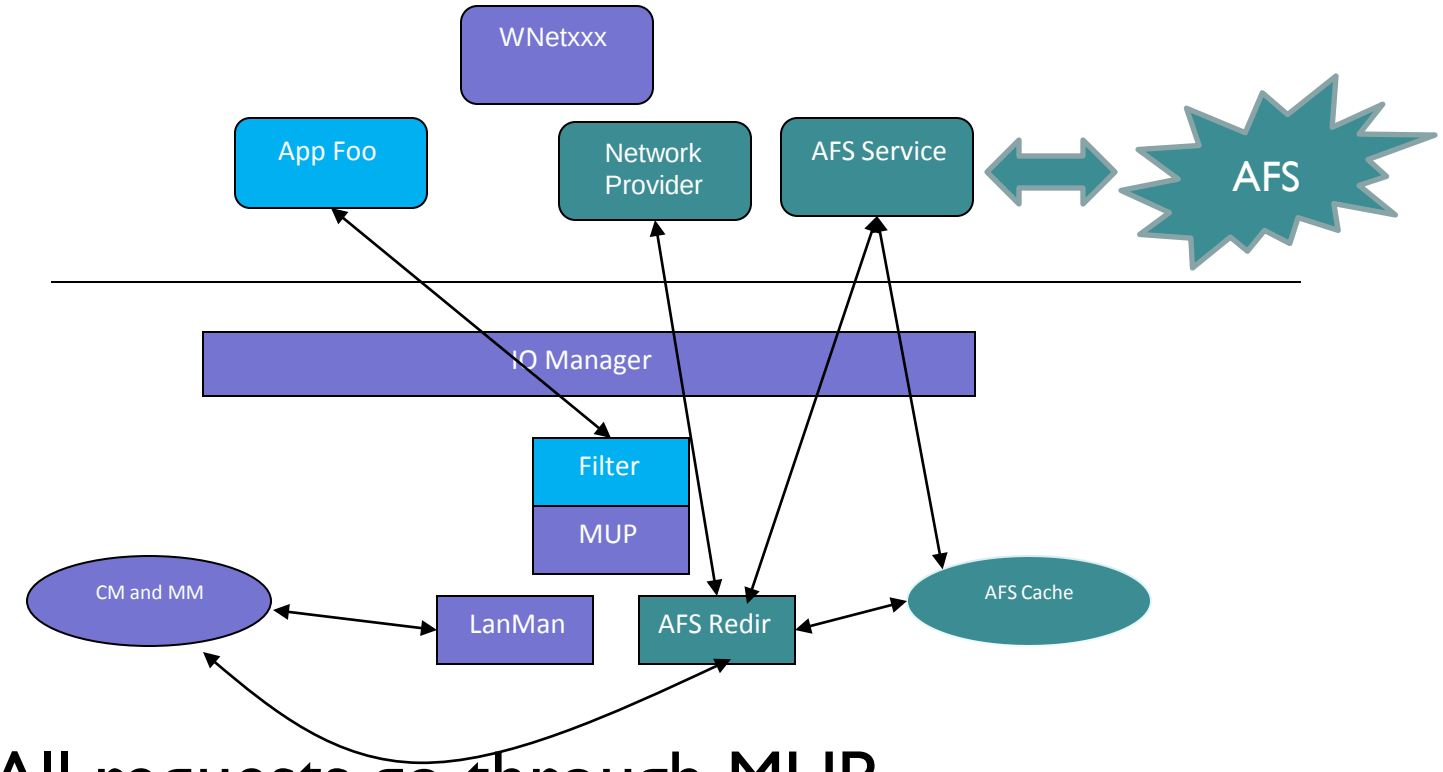


- Only prefix resolution and IRP_MJ_CREATE requests handled through MUP
- All subsequent requests issued to redirector
- Network redirectors would register, separately, as a file system resulting in filter attachment issues

New MUP Interface

- Register with MUP using a device name and an unnamed device object
  - Results in MUP creating a symbolic link from the device name to \Device\MUP
  - Prefix resolution using IOCTL_REDIR_QUERY_PATH_EX
- All requests go through MUP
- Single attachment point for filters

New MUP Interface



- All requests go through MUP
- Single point access

Path Parsing in Windows

- 2 forms can be sent – drive letter or not …
- Drive letter names come into MUP as

  \Device\MUP;AFS\Redirector\;C:\\AFS\Your-File-System.com\User

  Which are mapped by MUP into

  \;C:\\AFS\Your-File-System.com\User
- UNC names come into MUP as

  \Device\MUP\AFS\Your-File-System.com\User

  Which are mapped by MUP into

  \AFS\Your-File-System.com\User

Network Provider Interface

- User mode library with supporting interface in file system
- Used to support WNet API in user mode
- Implements drive letter resolution
- Communicates with file system for state and connection information
- Maintains per user information on mappings

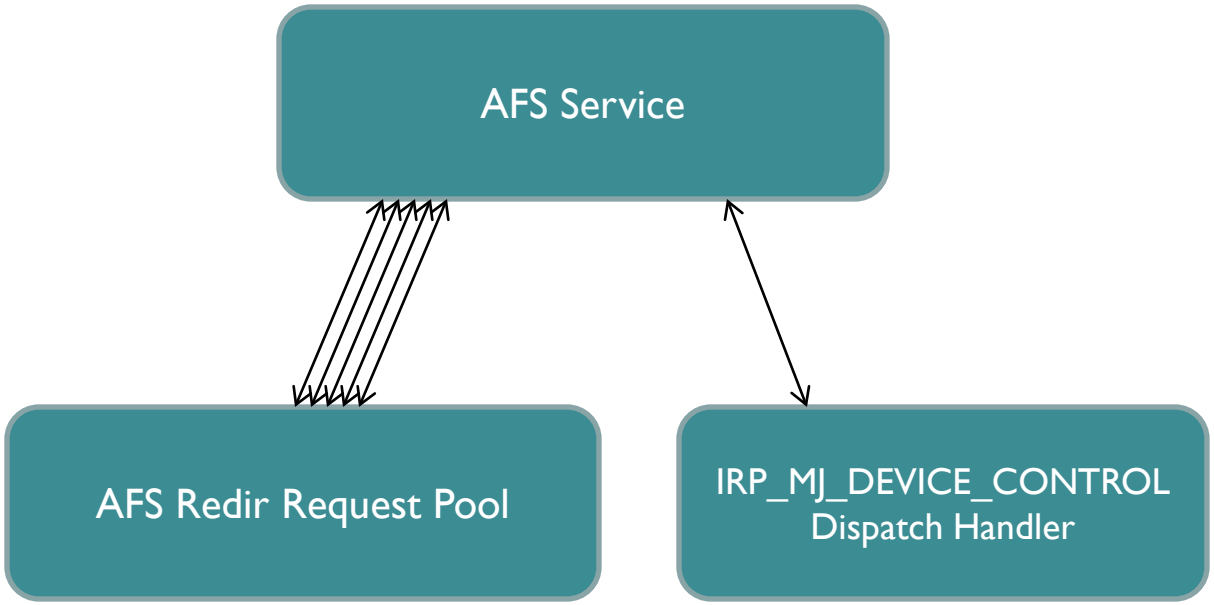
AFS on Windows … Design Goals

- Need to leverage as much functionality within the AFS Service as possible, initially …
  - Keep all server communication in service
    - Data retrieval
    - Callback registration and notification
    - Metadata management
- Complete integration into the Microsoft IFS API
- Stability and performance

AFS Service Interface

- Inverted call model
  - Requests from file system
  - Uses proprietary IOCtl interface
  - Communication through CDO symlink
- IOCtl interface
  - Requests to file system
  - Proprietary IOCtl interface for service initiated requests
- Cancellable interface through CDO handle

AFS Service Interface



- All requests issued through CDO symbolic link - \??\AFSRedirector
- Request pool state controlled through open handle

Merging Worlds

- Name space convergence
  - Symbolic Links – Microsoft and AFS
  - Mount Points
  - DFS Links
  - Component substitutions - @SYS
- File data handling
- PIOCtl Interface – Path IOCtl
- "Special" share name handling - srvsvc, wkssvc …
- Network Provider Interface

Namespace Convergence

- Cells and Shares
  - Share access mapped into cell names
    - \\AFS\Your-File-System.com
  - Dynamic discovery
- Reparse points and symbolic links
  - Must handle all symbolic links internally, they are not understood by Windows
  - Support the generic reparse point interface through FSCTL_xxx_REPARSE_POINT controls
    - Support mappings from C:\AFS to \\AFS\Your-File-System.com

Namespace Convergence

- Mount point processing managed internally
  - AFS Concept of Mount Point
- DFS Links are supported through reparse processing
  - Windows concept of reparse processing
- Must handle path traversal concepts
  - Crossing mount points … in both directions
  - Relative and absolute symbolic link processing
  - Component substitution

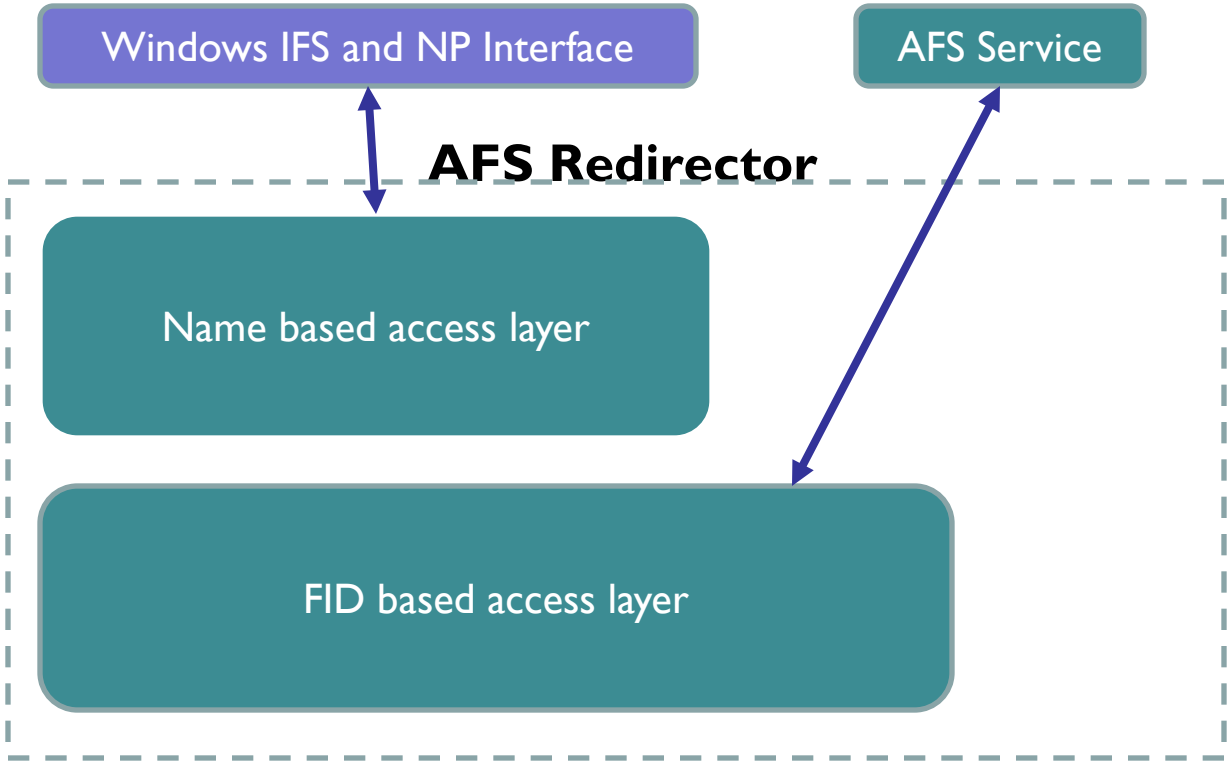
Metadata Handling

- Redirector caching model
  - Cache objects based on File Identifier (FID) on a per volume basis
  - Cache directory entries based on hash of name on a per directory basis
  - Support case insensitive, sensitive and short name lookups
  - Asynchronous pruning of trees when not in use

Metadata Handling

- Path name parsing in Windows
  - Path analyzed component by component, walking a specific branch for achieve the target object
  - Maintains a list of components used to access current target
  - Need to support relative symbolic links within a pathname

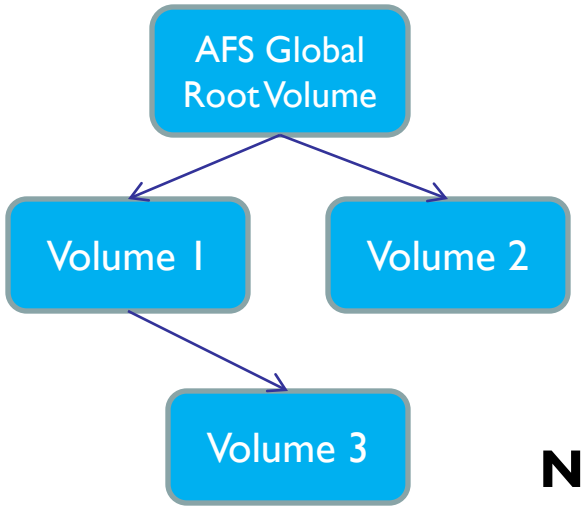
Namespace Implementation



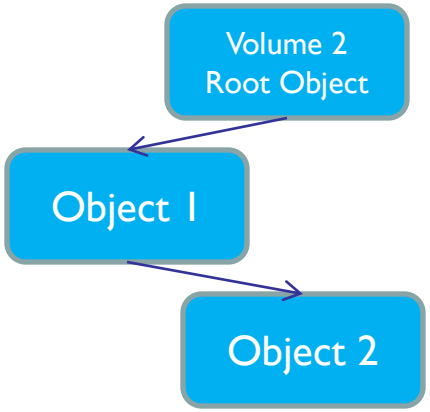
- FID based access is 'almost' lockless – Only volume based lock required
- Name based access is complex due to symlink, mount point, DFS link and other abstractions not recognized by Windows

Namespace Implementation

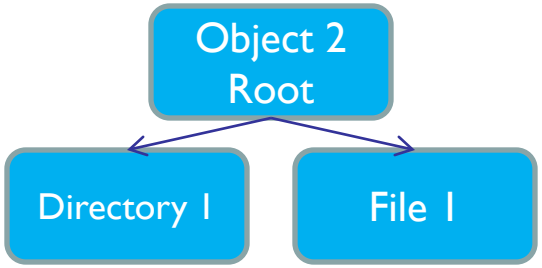
Volume Btree (Cell, Volume) (Vnode, Unique)



Object Btree



Name BTree (Component CRC)



File Data Handling

- Windows caching model
  - Re-entrant model – Need to be careful of locking hierarchy
  - Side band locking interface for memory and cache manager components – Fast IO interface
  - Need to observe IRQ levels while processing requests to underlying AFS Cache

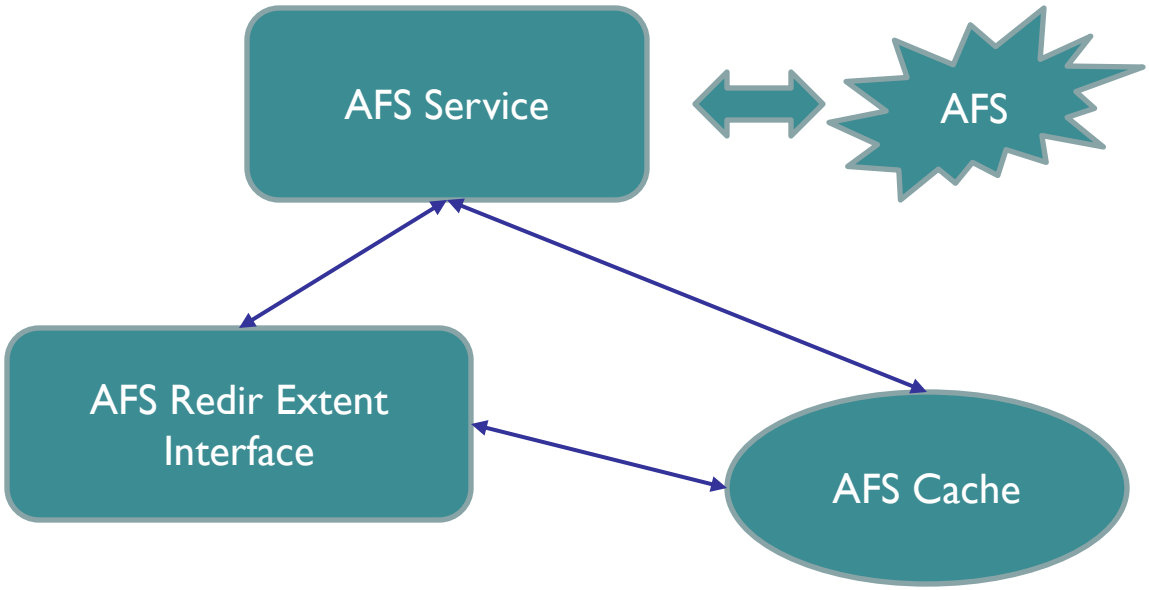
File Data Handling

- File Extent Interface
  - Extents describe the location of file data within the AFS Cache
  - Managed by the AFS Service and provided to the redirector upon request

File Data Handling

- AFS Caching
  - AFS Service populates AFS Cache with requested data and flushes dirty data back to server
  - AFS Redirector talks directly to the underlying AFS Cache through extents retrieved from the AFS Service
  - Interesting edge cases arise when performing large file copies using small AFS Cache sizes
    - Windows 'optimizations' in flushing
  - Leverage Windows Read-Ahead and Write Behind features

File Data Handling



- Allows for better performance by allowing redirector direct access to cache file
- AFS Service still manages cache layout and population

PIOCtl Interface

- The Path IOCtl interface is used to access object metadata trough a path based interface
- The interface has not changed from the AFS perspective
- Implemented within the redirector as 'special' file open requests
- File information and data management handled within the AFS Service

Special Share Names

- \PIPE\IPC$
  - Used for remote processing – currently not supported within the AFS Redirector
- \PIPE\srvsvc
  - Used for server and share information processing through the Net API
    - Currently supported through AFS Service
    - Leverages Microsoft RPC engine for translation
- \PIPE\wkssvc
  - Used for workstation information processing through the Net API

Invalidation Model

- Callback processing and issues in Windows
  - Callbacks can be made as a result of requests issued from the file system. Need to ensure these re-entrant calls do not lead to dead locks
    - 'Almost' lockless model in the callback routine through FID access layer
  - Server initiated callbacks have interesting effects, particularly in the directory change notification interface
    - Callbacks are FID based while notification is name based

Windows Change Notification

- Windows model for directory change notification
  - Objects added, modified or deleted initiate completion of a notification request
- Windows support API is named based … not in AFS
- Implement layer on top of Windows support API to map names to/from FIDs
  - Still edge cases that are not correctly handled, particularly in callback invalidation

Future …

- Alternate Data Streams
- Extended Attributes
- User and process quotas
- Enhanced extent processing interface
- Dynamically loadable functional driver – eliminates reboot for updates to file system