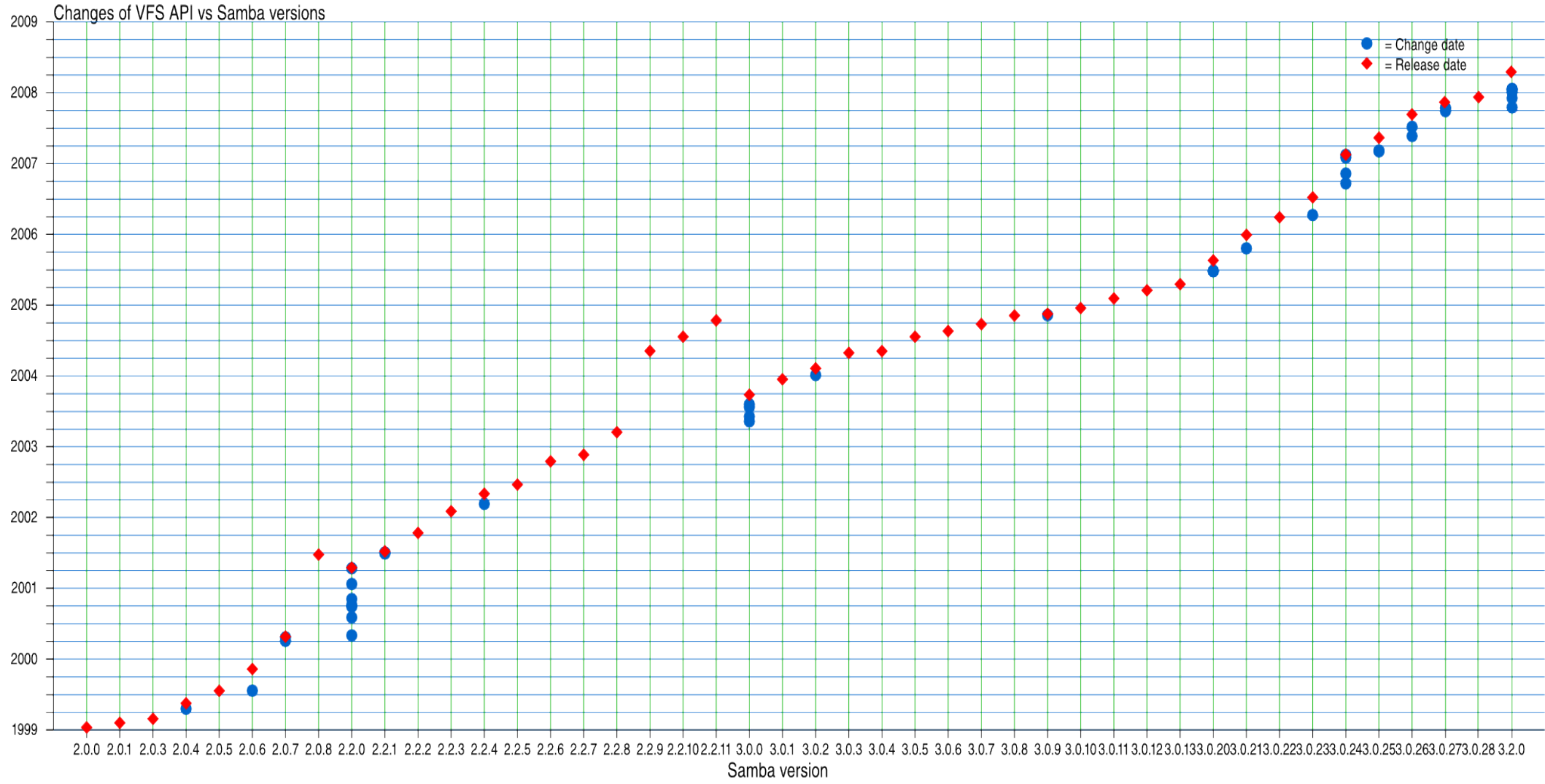
Virtual file system extensions in Samba 3.2

Alexander Bokovoy
ab@samba.org
Samba Team

Happy climbing!



Samba Virtual File System layer

- Started in April 1999 by Tim Potter as an abstraction layer
  - Included 21 operation (and an initialization function)
  - Allowed only one module to be loaded dynamically
- Multiple modules are supported since July 2002
- Number of operations has grown to 103 by 2008
  - Relations between operations are no longer easy to understand and implement
  - Source code analysis is now mandatory for any VFS module developer (if you want your module to work)

Analyzing VFS layer

- Samba 3.2 VFS layer consists of:
  - Definition:
    - include/vfs.h – constants and structures
    - include/vfs_macros.h – macro definitions to simplify accessing VFS structures
    - All structures seen by smbd are accessible from vfs modules
  - Implementation:
    - smbd/vfs.c – main driver and basic helpers
  - In-tree modules:
    - modules/vfs_default.c – default implementation for POSIX-compatible file systems
    - modules/vfs_*.c – various modules to look for examples

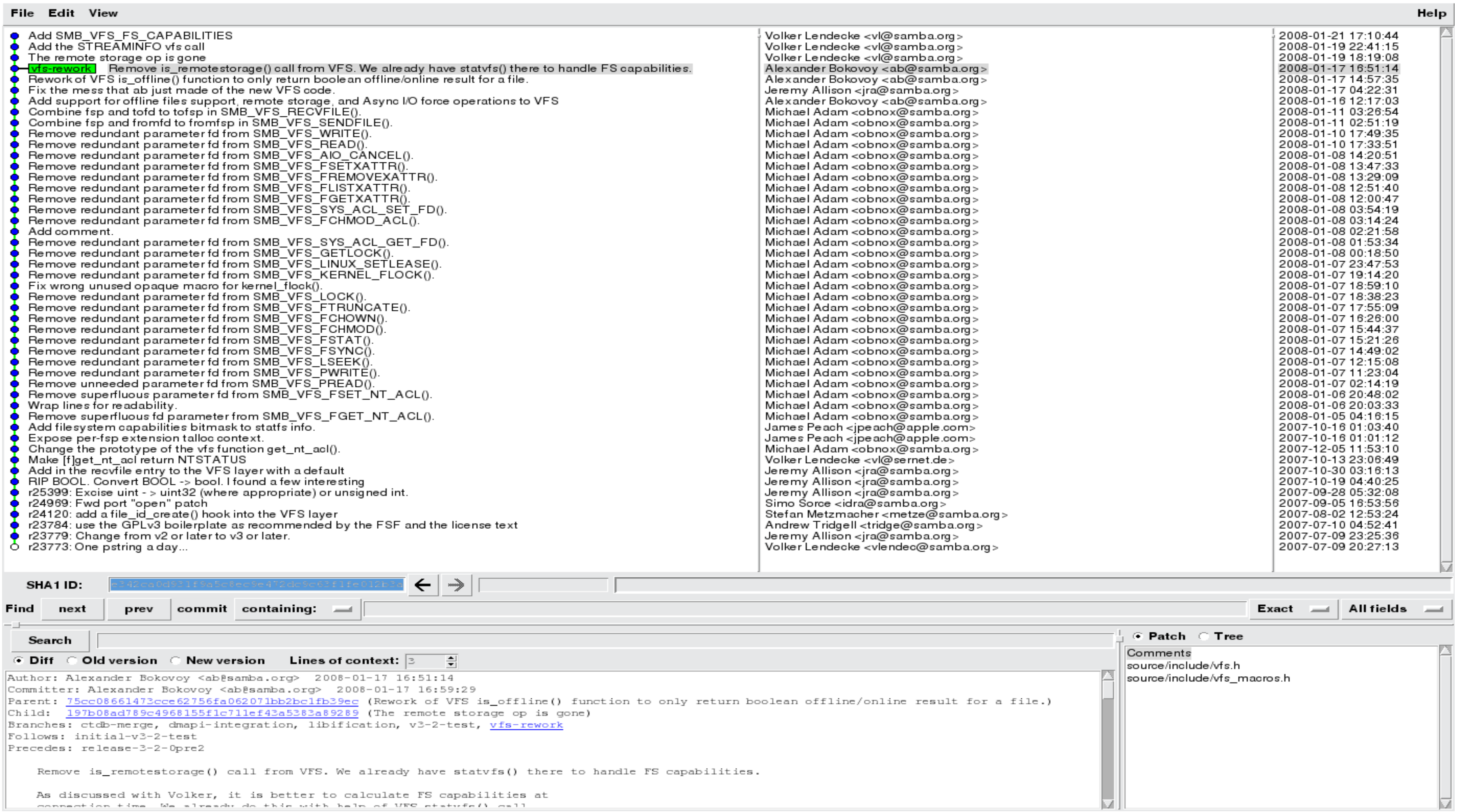
Analyzing VFS layer II

- Samba Team uses git for managing source code
  - Git features for a third-party developer:
    - Git repository contains all the history and branches
    - It is possible to do extensive forensic analysis locally
  - Important git commands for a VFS developer:
    - git log <version or sha-1> source/include/vfs*
      - Gives you all history on VFS layer public API since 1999
    - git diff|log <version1>..<version2> source/include/vfs*
      - Shows difference in VFS public API between versions
    - gitk <version or sha-1> source/include/vfs*
      - Gives you nice visual view of change sets

samba eXperience 2008

Analyzing VFS between 3.0.28 and 3.2

- gitk release-3-0-28..HEAD source/include/vfs*.h



From 3.0.28 towards 3.2

- Major changes in VFS API:
  - License update: GNU Public License v3
  - 8 file-related operations added
    - File system capabilities reporting with fs_capabilities()
    - recvfile() in addition to a sendfile()
    - lchown()
    - file_id_create()
    - streaminfo()
    - aio_force()
    - is_offline()
    - set_offline()

From 3.0.28 towards 3.2 II

- Major changes in VFS API (continued):
  - C structure and type changes
    - BOOL is gone, welcome bool
  - Removal of redundant arguments:
    - File descriptors are part of files_struct (fsp),
    - 'int fd' argument has gone from 26 of file-related functions

New VFS operations: fs_capabilities()

- File system capabilities
  - `uint32_t fs_capabilities(struct vfs_handle_struct *handle);`
  - Called once during connection setup
  - Result is used for filling in the `conn->fs_capabilities`
  - Flags aren't changed after that, only checked
  - If your module extends not replaces the file system behavior, always use SMB_VFS_LAYER_TRANSPARENT for the fs_capabilities() operation

File system capabilities

- FILE_CASE_SENSITIVE_SEARCH
- FILE_CASE_PRESERVED_NAMES
- FILE_UNICODE_ON_DISK
- FILE_PERSISTENT_ACLS
- FILE_FILE_COMPRESSION
- FILE_VOLUME_QUOTAS
- FILE_SUPPORTS_SPARSE_FILES
- FILE_SUPPORTS_REPARSE_POINTS
- FILE_SUPPORTS_REMOTE_STORAGE
- FS_LFN_APIS
- FILE_VOLUME_IS_COMPRESSED
- FILE_SUPPORTS_OBJECT_IDS
- FILE_SUPPORTS_ENCRYPTION
- FILE_NAMED_STREAMS
- FILE_READ_ONLY_VOLUME

Where Filesystem Capabilites are used?

- During open() call semantics is different if named streams are present
  - We can't open file with DELETE access if any of the file's streams is open without DELETE access
  - If file system capabilities show support for named streams, we are able to open specific stream
- During close() call we delete all named streams for the file if file system capabilities support named streams
- File search operations depend heavily on a case-sensitive search capability
- File system attributes reported through TRANS2 query file system info call

Offline Operations

- File could be taken offline by a Hierarchical Management Software
  - File will be on a media with slow access times
  - File access could block until the file migrated back to the disk
  - Windows Explorer shows nice black clock as part of file's icon if FILE_ATTRIBUTE_OFFLINE is set
    - For each file we call is_offline() call to check its status
    - SetFileAttributes() could force us to set file offline, therefore, we have set_offline() call

Offline Operations

- What systems are supported?
  - Offline operation is a feature of Hierarchical storage management software (HSM)
  - HSM usually coordinates its activity with a file system through Data Management API (DMAPI)
  - vfs_tsm module supports generic HSM via DMAPI calls
    - Check for presence of a DMAPI attribute
    - Check for specific value of a specific DMAPI attribute
- What is required for implementation of offline operations?
  - Asynchronous operations
  - File system capabilities

Named streams

- NTFS has notion of a file streams
  - POSIX lacks it, we emulate them
    - vfs_streams_xattr module saves streams as extended attributes
    - vfs_streams_depot module uses a separate directory to store streams
- streaminfo() call is used to get information about file's streams
  - Returns number of streams for given file and their information

Asynchronous operations

- Samba 3 provides support for (sort of) asynchronous operations
  - Requires Async I/O support from the operating system
  - Very fragile as underlying Async I/O is broken on many GNU/Linux distributions
  - Supports a number of outstanding calls:
    - Controlled with 'max mux' smb.conf parameter (default 50)
    - Actual logic for read/write calls is described in the 'aio read size'/'aio write size' smb.conf parameters
  - aio_force() operation is used by offline operations to redirect deferred read/write to an offline file

File system identification

- file_id_create() operation
  - Abstracts out mount point uniqueness
  - Needed to represent uniformly a distributed file system 's mount points on multiple cluster nodes
  - file_id structure is used as a key to share modes and locking databases
    - Are we locking the same file on multiple nodes?
  - vfs_fileid module implements number of algorithms:
    - fsname – hash of mount point (e.g. /gpfs or /gfs)
    - fsid – hash of statfs.fsid value

Questions?

Alexander Bokovoy
ab@samba.org
Samba Team