Scalable FileChangeNotify

SDC 2014
Santa Clara

Volker Lendecke

Samba Team / SerNet

2014-09-17

SerNet

- SLA based support for more than 650 customers
  - firewalls, VPN, certificates, audits
  - based on open standards wherever possible
- Support for many OS: Linux, Cisco IOS, Windows etc.
- Compliant with BSI Grundschutz and ISO 27001 and other international regulations
- SerNet and Samba
  - technological leadership of SerNet worldwide
    - SerNet distributes up-to-date Samba packages
  - samba eXPerience
    - The international Samba conference

What is FileChangeNotify?

- MSDN on "Obtaining Directory Change Notifications":
  - An application can monitor the contents of a directory and its subdirectories by using change notifications.
- Client queries a directory handle for changes
- Filters are sent for just specific events:
  - "I'm only interested in new and deleted files"
  - "Please tell me when a file size changes"
  - ...
- API parameter **bWatchSubtree**:
  - If this parameter is TRUE, the function monitors the directory tree rooted at the specified directory.

[MS-FSA] specification

- 3.1.1.1 Attributes per Volume (i.e. filesystem)
  - ChangeNotifyList: A list of zero or more ChangeNotifyEntries describing outstanding change notify requests for the volume.
- 3.1.4.1 Algorithm for Reporting a Change Notification for a Directory
  - **For each** ChangeNotifyEntry in Volume.ChangeNotifyList:
  - Do something like apply filters, send notifies
- "3.1.4.1," mentioned at least 12 times in [MS-FSA]
- For every metadata operation the spec makes us walk a large array
- What happens if you have 10.000 clients with 10 notifies each?
- How can we maintain the ChangeNotifyList in a cluster?

FileChangeNotify on the wire

- Client opens a directory
- Client sends a `CHANGE_NOTIFY` request
  - FileID references the open directory handle
  - CompletionFilter shows which changes the client wants to see
  - This creates the "ChangeNotifyEntry"
- When changes happen, server replies to the `CHANGE_NOTIFY` request
- Until client sends a fresh `CHANGE_NOTIFY` request, server has to queue changes
- If the queue overflows, server can reply with "Something changed, but I don't know what"
- The ChangeNotifyEntry is only removed when closing the directory

FileChangeNotify in Samba

- Three implementations
  - It seems that Samba often requires a few rounds to get things right
  - Anyone remember the Samba 2.0 oplock implementation? :-)
- Samba 3.0
  - No global ChangeNotifyList equivalent
  - Timeout-based polling of directories per smbd
- Tridge's Samba4 implementation
  - Tridge figured out how much more of the protocol
  - Messaging-based notification
  - Ported to Samba 3.2
- Samba 4.0 notify_index.tdb
  - Starts to make notify possible in a cluster

Samba 3.0

- Contents of the directory are hashed
- Periodically `hash_check_notify` is called
- Recalculates the hash
- Upon changes, Samba returns `STATUS_NOTIFY_ENUM_DIR`
  - Samba did not return exactly what changed
- High load due to polling in every smbd
- Updates can lag
- No recursive notifies

Samba 3.2

- During the NTVFS effort, Tridge figured out the ChangeNotifyList
- PIDL came around, complex data structures could be marshalled
- Tridge implemented the ChangeNotifyList as a hierarchical array of arrays
  - "This function is called a lot, and needs to be very fast. The unusual data structure and traversal is designed to be fast in the average case, even for large numbers of notifies"
- notify.tdb stores the ChangeNotifyList a.k.a. notify_array in one record
- Every smbd has a copy, updated on every change
  - tdb_seqnum was invented for this
  - This does not scale to thousands of smbds and notifies
- Problems in a cluster
  - No real tdb_seqnum
  - One large record bounced back and forth like mad

- The Samba 3.2-3.6 implementation has one tdb record for the complete ChangeNotifyList
- Every change pushes one huge record to every node and smbd
- Goal: Reduce write load on the central notify database
- Every notify event is path-based and needs to look at all the parents' ChangeNotifyEntry records
- Split up the notify_array into records indexed by path
  - notify.tdb now has many path-indexed records
  - Every record holds a number of ChangeNotifyEntry records
  - A change notify event walks the path, looking for recursive entries
- Typically a lot of contention on just a few directories
  - Share root directories are very popular to look at

Samba 4.0 clustered

- Write load on individual tdb records still high
- High n:m messaging load across nodes
  - Notify events inform many smbds, possibly many on the same node
- Split up notify.tdb into a cluster-wide notify_index.tdb and a node-local notify.tdb
  - Both tdbs indexed by path
  - ChangeNotifyEntry records local in notify.tdb
- notify_index.tdb holds just node numbers
  - Every node records itself with the path if any notify.tdb record exists
  - Just one single entry per node in notify_index.tdb
- Notify events are sent to a remote proxy process
  - Proxy multi-casts notify events from its notify.tdb
- notify_index.tdb deletion is deferred
  - Write load on notify_index.tdb is significantly reduced
- Next bottleneck: read access on entry for "/" in notify_index.tdb

FileChangeNotify NextGeneration

- "This function is called a lot ..."
  - This function (notify_trigger) is now O(n) in the number of path components
  - For a 10-level deep file create, tdb_parse_record is called 10 times
  - tdb is fast, but it does cost, in particular with fcntl locks being one systemwide spinlock
- Notify events must be as cheap as possible
  - FileChangeNotify is asynchronous
  - Why not delegate notify_trigger to some other process?
- Until a few months ago, Samba internal messaging was heavy-weight
  - tdb-based with SIGUSR1 as the async notification
  - With unix datagram messaging, sending a message is a single syscall

Notifyd design

- Keep the ChangeNotifyList in one daemon
- Smbd adds and removes ChangeNotifyEntries by messages to notifyd
- Notify events are another type of message
- All recursive filtering is done by notifyd
- notifyd in a cluster distributes the local ChangeNotifyList

But what about delayed messages?

- A delayed ChangeNotifyEntry creation will lose notifies
  - The event (e.g. mkdir) happens before the Entry is created
- Every message carries a timestamp
  - We could save notify events for a while
  - When should we drop them?
- Calculate a hash of the path name
  - Maintain an array of timestamps indexed by that hash
  - When an Entry comes in, check the timestamp
  - If it's later, just reply with overflow
    - All that can happen is false positives

Prereq / Benefits

- One message per metadata modification
  - Fast messaging between smbds
  - Unix domain datagram messages do roughly 150k/sec
  - Cluster inside one host possible for higher demands
- Less load on inotify
  - One notify listener instead of every smbd
- Clusterwide file change notify
  - GPFS does not provide clusterwide inotify
  - inotify works locally, notifyd tells others
- External event sources (Ganesha?)
  - A single unix dgram per event
  - Extremely simple protocol

Questions?

vl@samba.org / vl@sernet.de