SDC 20
BY Developers FOR Developers
Storage Developer Conference
September 22-23, 2020

multichannel / io_uring

Status Update within Samba

Stefan Metzmacher <metze@samba.org>

Samba Team / SerNet

2020-09-22

https://samba.org/~metze/presentations/2020/SDC/

Topics

- What is SMB3 Multichannel
- Multichannel in Samba 4.4 (2016)
- Updates in Samba 4.13 (2020)
- What is io-uring
- vfs_io_uring in Samba 4.12 (2020)
- Future Improvements
- Questions? Feedback!

What is SMB3 Multichannel (Part 1)

- Multiple transport connections are bound to one logical connection
  - This allows using more than one network link
    - Good for performance
    - Good for availability reasons
  - Non TCP transports like RDMA (InfiniBand, RoCE, iWarp)
- All transport connections (channels) share the same CliendGUID
  - This is important for Samba
- An authenticated binding is done at the user session layer
  - SessionID, TreeID and FileID values are valid on all channels
- Available network interfaces are auto-negotiated
  - FSCTL_QUERY_NETWORK_INTERFACE_INFO interface list
  - IP (v4 or v6) addresses are returned together with:
    - Interface Index (which addresses belong to the same hardware)
    - Link speed
    - RSS and RDMA capabilities

What is SMB3 Multichannel (Part 2)

- IO ordering is important for multichannel
  - Requests can get lost between client and server
  - Responses can get lost between server and client
  - The client isn't able to know the difference
  - Replays contain the REPLAY flag in the SMB2 header
  - FILE_NOT_AVAILABLE indicates "please retry" to the client
- State changing operations need replay detection
  - They need to execute only-once
  - SMB2 Create uses a CreateGUID
  - SMB2 Lock uses an array with sequence numbers
    - Windows only supports this on resilient and persistent handles
    - Future Windows versions are supposed to fix that

What is SMB3 Multichannel (Part 3)

- Write/Set operations only need a barrier
  - An epoch number is incremented on each channel failure
  - The current epoch number is part of each request
  - The server remembers the last seen epoch number
  - Non-REPLAY requests with stale epoch fail
  - REPLAY requests fail, when there are pending older epoch numbers
- Read/Get operations can be replayed safely
- Lease/Oplock break notifications should be retried
  - Break notifications wait for transport acks
  - On channel failures they are retried on other channels
  - Windows doesn't retry for oplocks, only leases

Multichannel in Samba 4.4 (Part 1)

- 4.4.0 added the "server multi channel support" option
  - But it is disabled by default (up to now)
  - Not all IO ordering protections are implemented
- FD-passing is used to pass a connection based on the ClientGUID
  - Only one smbd process handles all connections for a ClientGUID
  - At SMB2 Negprot we lookup existing process
  - We pass the socket fd and the full SMB2 Negprot request
- Interface capabilities can be specified
  - interfaces = "eth0;if_index=65,speed=1000000000,capability=RSS"
  - We autodetect the interface index on all platforms
  - On Linux we also autodetect the link speed
  - We support FSCTL_QUERY_NETWORK_INTERFACE_INFO

Multichannel in Samba 4.4 (Part 2)

- We changed the data model to support multiple connections
  - We have a list of struct smbXsrv_connection on struct smbXsrv_client
  - We support Session Binds to make connections valid on a session
  - SessionID, TreeID and FileID tables are hold in struct smbXsrv_client
  - The smbd process only exists when the last connection is disconnected
- 4.4 implemented the following IO ordering protections
  - We implement SMB2 Create replay detection (4.4.0)
  - We implement the channel sequence number verification (4.4.4)
- The following were missing:
  - SMB2 LockSequence replay detection
  - Retries of Lease/Oplock Break Notifications (Bug #11898)
  - Integration with CTDB (Bug #11898)
  - Automated regression tests
    - socket_wrapper does not support fd-passing (Bug #11899)

Updates in Samba 4.13 (Part 1)

- SMB2 LockSequence replay detection
  - Windows only implements this for resilient and persistent handles
  - [MS-SMB2] proposes it also for durable handles and multichannel
  - Samba follows [MS-SMB2] by default
  - "smb2 disable lock sequence checking = yes" can disable it if required
- Integration with CTDB (Bug #11898)
  - A client can only talk to one node at a time
  - Samba hides public addresses and only returns node local addresses
  - We disconnect all connections if one with a public address gets disconnected
  - There might be room for more advanced logic in future
- On Linux we autodetect the RSS capability
  - We use ETHTOOL_GRXRINGS in order to detect it

Updates in Samba 4.13 (Part 2)

- Retries of Lease/Oplock Break Notifications (Bug #11898)
  - smbtorture tests can simulate channel failures
    - It can use iptables for testing real servers
    - SMB2 IOCTL call to simulate failure against Samba
  - We wrote complex tests to find out the Windows behavior
    - The TCP layer retransmits after a timeout (RTO) passed
    - =>Depending on the Version RTO is between 0.2 and 10 seconds
    - After about 5 retransmissions a connection is marked as broken
    - =>The failure is detected after a time between 1.5 and 20 seconds
  - Windows only uses the last channel for Oplocks (without retry)

- Only Linux and FreeBSD have the required kernel interfaces
  - We try to get the RTO via struct tcp_info.tcpi_rto
    - We limit the value between 0.2 and 1 second
  - We need to ask the kernel for the number of unacked bytes
    - Linux (TIOCOUTQ) and FreeBSD (FIONWRITE)
    - We disable multichannel feature if the platform doesn't support this

Updates in Samba 4.13 (Part 3)

- Generic SMB2 Break Notification per struct smbXsrv_client
  - Individual connections are hidden from the Oplock/Lease logic
  - Internally we go async and keep some state around
    - The blob is independend of the connection
    - It's not signed nor encryted
  - We iterate over all available connections
    - Starting with the oldest one (even for Oplocks)
    - "smb2 disable oplock break retry = yes" can disable it if required
    - If we get a failure, we retry on the next channel

- SMB2 Break Notification on per struct smbXsrv_connection
  - After each sendmsg() call we increment our unacked bytes counter
    - We remember the value of the counter for break notifications
  - We get the current RTO and setup a timer firing after 6 * RTO
    - The timer calculates the number of acked bytes
    - If the break notification wasn't acked we teardown the connection
    - Otherwise we report success to the generic layer
  - On any connection teardown, we report a failure to the generic layer

Missing in Samba 4.13 (Part 1)

- Automated regression tests are still not there
  - We already had a regression that made multichannel unusable
  - So we really need automatic testing in autobuild/gitlab-ci
- socket_wrapper needs fd-passing support(Bug #11899)
  - We need to transfer the inet meta data for the passed socket
  - Samba doesn't need concurrent access to a single socket
  - As a start we write the information into a temporary pipe
  - The read end of the pipe fd is passed as last element of the fd array
  - The receiver reads from the pipe fd and builds the in memory meta data
  - The code is almost ready and allows automatic multichannel tests
  - Will hopefully be ready for 4.14

Missing in Samba 4.13 (Part 2)

During the latest development we found a few new problems:

- The connection passing is fire and forget (Bug #14433)
  - There's a race between:
    - Looking an existing process by ClientGUID
    - And passing the connection to that process
  - The sending process doesn't wait for an ack
    - The connection can get silently disconnected

- Pending async operations are canceled (Bug #14449)
  - A disconnect of a connection cancels pending state-changing operations
  - To get the replay semantics right we need to keep the requests running
  - We need to research how SMB2 Create replays work with async opens

- These will hopefully be fixed with 4.14
  - We need feedback from real world installations
  - Then we can change the default to:
    - "server multi channel support = yes"

What is io-uring (Part 1)

- Linux 5.1 introduced a new scalable AIO infrastructure
  - It's designed to avoid syscalls as much as possible
  - kernel and userspace share mmap'ed rings:
    - submission queue (SQ) ring buffer
    - completion queue (CQ) ring buffer
  - See "Ringing in a new asynchronous I/O API" on LWN.NET

- Relevant features for Samba:
  - Between userspace and filesystem (available from 5.1):
    - IORING_OP_READV, IORING_OP_WRITEV and IORING_OP_FSYNC
    - Supports buffered and direct io
  - Between userspace and socket (and also filesystem) (from 5.8)
    - IORING_OP_SENDMSG, IORING_OP_RECVMSG
    - IORING_OP_SPLICE, IORING_OP_TEE
    - Maybe using IORING_SETUP_SQPOLL or IOSQE_ASYNC
  - Path based syscalls with async impersonation (from 5.6)
    - IORING_OP_OPENAT2, IORING_OP_STATX
    - Using IORING_REGISTER_PERSONALITY for impersonation

vfs_io_uring in Samba 4.12 (2020)

- With Samba 4.12 we added "io_uring" vfs module
  - For now it only implements SMB_VFS_PREAD,PWRITE,FSYNC_SEND/RECV
  - It has less overhead than our pthreadpool default implementations
  - I was able to speed up a smbclient 'get largefile /dev/null'
    - Using against smbd on loopback
    - The speed changes from 2.2GBytes/s to 2.7GBytes/s
- The improvement only happens by avoiding context switches
  - But the data copying still happens:
    - From/to a userspace buffer to/from the filesystem/page cache
  - The data path between userspace and socket is completely unchanged
  - For both cases the cpu is mostly busy with memcpy

Future Improvements

- There're a lot of potential for improvements
  - Using sendfile() instead produces much less overhead
    - I got about 9 GBytes/s
  - This indicates that using io-uring based zero-copy would be good
    - IORING_OP_SENDMSG, IORING_OP_RECVMSG
    - IORING_OP_SPLICE, IORING_OP_TEE
  - This would also improve the data path between to/from the socket
  - IORING_OP_TEE would also allow reduced overhead with signing
  - eBPF support in io-uring would also be great for optimizations
- The data paths for multichannel may also be improved
  - IO could be offloaded kernel threads using:
    - IORING_SETUP_SQPOLL or IOSQE_ASYNC

Thanks!

People who helped out:

- Michael Adam
- Günther Deschner
- Sachin Prabhu
- Anoop C S

Questions? Feedback!

- Feedback regarding real world testing would be great!
  - Typically I can only test with VMs on my Laptop
- Stefan Metzmacher, `metze@samba.org`
- https://www.sernet.com
- https://samba.plus

Slides: https://samba.org/~metze/presentations/2020/SDC/