

smb(3)status

Status of SMB(3) in Samba

Michael Adam

SerNet / Samba Team

2014-09-16

Report on work by several people



- ▶ SMB Recap
- ▶ Leases
- ▶ Multi-Channel
- ▶ RDMA/SMB direct
- ▶ Clustering

SMB Protocol in Microsoft Windows

- ▶ 1.0: up to Windows XP / Server 2003
- ▶ 2.0: Windows Vista / Server 2008 [2006/2008]
 - ▶ handle based operations
 - ▶ durable file handles
- ▶ 2.1: Windows 7 / Server 2008R2 [2009]
 - ▶ leases
 - ▶ multi-credit / Large MTU
 - ▶ dynamic reauthentication
 - ▶ resilient file handles
- ▶ 3.0: Windows 8 / Server 2012 [2012]
- ▶ 3.02: Windows 8.1 / Server 2012R2 [2013]
- ▶ 3.1: coming...

SMB Protocol in Microsoft Windows

- ▶ 1.0: up to Windows XP / Server 2003
- ▶ 2.0: Windows Vista / Server 2008 [2006/2008]
 - ▶ handle based operations
 - ▶ durable file handles
- ▶ 2.1: Windows 7 / Server 2008R2 [2009]
 - ▶ leases
 - ▶ multi-credit / Large MTU
 - ▶ dynamic reauthentication
 - ▶ resilient file handles
- ▶ 3.0: Windows 8 / Server 2012 [2012]
- ▶ 3.02: Windows 8.1 / Server 2012R2 [2013]
- ▶ 3.1: coming...

SMB Protocol in Samba

- ▶ Samba < 3.5:
 - ▶ SMB 1
- ▶ Samba 3.5:
 - ▶ experimental incomplete support for SMB 2.0
- ▶ Samba 3.6:
 - ▶ official support for SMB 2.0
 - ▶ missing: durable handles
 - ▶ default server max proto: SMB 1
- ▶ Samba 4.0:
 - ▶ SMB 2.0: complete with durable handles
 - ▶ SMB 2.1: basis, multi-credit, dynamic reauthentication
 - ▶ SMB 3.0: basis, crypto, secure negotiation, durable v2
 - ▶ default server max proto: SMB 3.0

SMB Protocol in Samba

- ▶ Samba < 3.5:
 - ▶ SMB 1
- ▶ Samba 3.5:
 - ▶ experimental incomplete support for SMB 2.0
- ▶ Samba 3.6:
 - ▶ official support for SMB 2.0
 - ▶ missing: durable handles
 - ▶ default server max proto: SMB 1
- ▶ Samba 4.0:
 - ▶ SMB 2.0: complete with durable handles
 - ▶ SMB 2.1: basis, multi-credit, dynamic reauthentication
 - ▶ SMB 3.0: basis, crypto, secure negotiation, durable v2
 - ▶ default server max proto: SMB 3.0

SMB Protocol in Samba

- ▶ Samba < 3.5:
 - ▶ SMB 1
- ▶ Samba 3.5:
 - ▶ experimental incomplete support for SMB 2.0
- ▶ Samba 3.6:
 - ▶ official support for SMB 2.0
 - ▶ missing: durable handles
 - ▶ default server max proto: SMB 1
- ▶ Samba 4.0:
 - ▶ SMB 2.0: complete with durable handles
 - ▶ SMB 2.1: basis, multi-credit, dynamic reauthentication
 - ▶ SMB 3.0: basis, crypto, secure negotiation, durable v2
 - ▶ default server max proto: SMB 3.0

SMB Protocol in Samba

- ▶ Samba < 3.5:
 - ▶ SMB 1
- ▶ Samba 3.5:
 - ▶ experimental incomplete support for SMB 2.0
- ▶ Samba 3.6:
 - ▶ official support for SMB 2.0
 - ▶ missing: durable handles
 - ▶ default server max proto: SMB 1
- ▶ Samba 4.0:
 - ▶ SMB 2.0: complete with durable handles
 - ▶ SMB 2.1: basis, multi-credit, dynamic reauthentication
 - ▶ SMB 3.0: basis, crypto, secure negotiation, durable v2
 - ▶ default server max proto: SMB 3.0

SMB Protocol in Samba

- ▶ Samba < 3.5:
 - ▶ SMB 1
- ▶ Samba 3.5:
 - ▶ experimental incomplete support for SMB 2.0
- ▶ Samba 3.6:
 - ▶ official support for SMB 2.0
 - ▶ missing: durable handles
 - ▶ default server max proto: SMB 1
- ▶ Samba 4.0:
 - ▶ SMB 2.0: complete with durable handles
 - ▶ SMB 2.1: basis, multi-credit, dynamic reauthentication
 - ▶ SMB 3.0: basis, crypto, secure negotiation, durable v2
 - ▶ default server max proto: SMB 3.0

Leases (SMB 2.1)

Leases - Status in Samba

Leases are work in progress, but can be considered almost done. Code already survives most test cases. Still need to fix a few corner cases... ☺
Still hope to get Leases with 4.2?!...

Leases - Status

- ▶ Samba had oplocks (SMB1/SMB2) since a long time.
- ▶ Oplocks per FSA level file handle.
- ▶ No need to keep extra information on SMB2 level.
- ▶ Leases identified by LeaseKey + ClientGUID.
- ▶ Can be shared by multiple opens.
- ▶ ⇒ Changes to `open_files.idl`
- ▶ SMB2 extra: LeaseKey generated by client, based on UNC path.
- ▶ LeaseKey can not be attached to multiple UNC's.
- ▶ ⇒ Need to maintain additional SMB-level Data.

- ▶ Samba had oplocks (SMB1/SMB2) since a long time.
- ▶ Oplocks per FSA level file handle.
- ▶ No need to keep extra information on SMB2 level.
- ▶ Leases identified by LeaseKey + ClientGUID.
- ▶ Can be shared by multiple opens.
- ▶ \Rightarrow Changes to `open_files.idl`
- ▶ SMB2 extra: LeaseKey generated by client, based on UNC path.
- ▶ LeaseKey can not be attached to multiple UNC's.
- ▶ \Rightarrow Need to maintain additional SMB-level Data.

- ▶ Samba had oplocks (SMB1/SMB2) since a long time.
- ▶ Oplocks per FSA level file handle.
- ▶ No need to keep extra information on SMB2 level.
- ▶ Leases identified by LeaseKey + ClientGUID.
- ▶ Can be shared by multiple opens.
- ▶ ⇒ Changes to `open_files.idl`
- ▶ SMB2 extra: LeaseKey generated by client, based on UNC path.
- ▶ LeaseKey can not be attached to multiple UNC's.
- ▶ ⇒ Need to maintain additional SMB-level Data.

- ▶ Samba had oplocks (SMB1/SMB2) since a long time.
- ▶ Oplocks per FSA level file handle.
- ▶ No need to keep extra information on SMB2 level.
- ▶ Leases identified by LeaseKey + ClientGUID.
- ▶ Can be shared by multiple opens.
- ▶ ⇒ Changes to `open_files.idl`
- ▶ SMB2 extra: LeaseKey generated by client, based on UNC path.
- ▶ LeaseKey can not be attached to multiple UNC's.
- ▶ ⇒ Need to maintain additional SMB-level Data.

Leases - Problems to be solved

- ▶ Samba has "magic" shares ("homes" share, variable paths):
 - ▶ Same `//server/share`
 - ▶ different directory/file on disk!
 - ▶ \Rightarrow Client may "think" to access the same file
 - ▶ \Rightarrow Need to break leases and disallow simultaneous leases.

- ▶ Samba has "magic" shares ("homes" share, variable paths):
 - ▶ Same `//server/share`
 - ▶ different directory/file on disk!
 - ▶ \Rightarrow Client may "think" to access the same file
 - ▶ \Rightarrow Need to break leases and disallow simultaneous leases.

Multi-Channel

Multi-Channel - Windows/Protocol

- ▶ find interfaces with interface discovery:
`FSCTL_QUERY_NETWORK_INTERFACE_INFO`
- ▶ bind additional TCP (or RDMA) connection to established SMB3 session (session bind)
- ▶ bind only to a single node
- ▶ Client decides which connections to bind, which channels to use (fastest).
- ▶ replay / retry mechanisms, epoch numbers

Multi-Channel - Windows/Protocol

- ▶ find interfaces with interface discovery:
`FSCTL_QUERY_NETWORK_INTERFACE_INFO`
- ▶ bind additional TCP (or RDMA) connection to established SMB3 session (session bind)
- ▶ bind only to a single node
- ▶ Client decides which connections to bind, which channels to use (fastest).
- ▶ replay / retry mechanisms, epoch numbers

Multi-Channel - Samba - Thoughts

- ▶ Samba/smbd: multi-process
- ▶ currently: process \Leftrightarrow TCP connection
- ▶ idea: transfer new connection to existing smbd
 - ▶ \Rightarrow no need to coordinate between processes on unix file level
- ▶ use fd-passing (sendmsg/recvmmsg) on TCP socket fd
- ▶ idea: don't transfer connection in session bind, but already *in NEGPROT* based on the ClientGUID
 - ▶ less state to coordinate
 - ▶ \Rightarrow essentially single process model per ClientGUID even if multi-channel is not used
 - ▶ rely on good async infrastructure for I/O (pthread-pool, ...)
 - ▶ only affects clients who send a Client GUID (SMB \geq 2.1)
 - ▶ possibly make this tunable-off(?)

Multi-Channel - Samba - Thoughts

- ▶ Samba/smbd: multi-process
- ▶ currently: process \Leftrightarrow TCP connection
- ▶ idea: transfer new connection to existing smbd
 - ▶ \Rightarrow no need to coordinate between processes on unix file level
- ▶ use fd-passing (sendmsg/recvmmsg) on TCP socket fd
- ▶ idea: don't transfer connection in session bind, but already *in NEGPROT* based on the ClientGUID
 - ▶ less state to coordinate
 - ▶ \Rightarrow essentially single process model per ClientGUID even if multi-channel is not used
 - ▶ rely on good async infrastructure for I/O (pthread-pool, ...)
 - ▶ only affects clients who send a Client GUID (SMB \geq 2.1)
 - ▶ possibly make this tunable-off(?)

Multi-Channel - Samba - Thoughts

- ▶ Samba/smbd: multi-process
- ▶ currently: process \Leftrightarrow TCP connection
- ▶ idea: transfer new connection to existing smbd
 - ▶ \Rightarrow no need to coordinate between processes on unix file level
- ▶ use fd-passing (sendmsg/recvmmsg) on TCP socket fd
- ▶ idea: don't transfer connection in session bind, but already *in NEGPROT* based on the ClientGUID
 - ▶ less state to coordinate
 - ▶ \Rightarrow essentially single process model per ClientGUID even if multi-channel is not used
 - ▶ rely on good async infrastructure for I/O (pthread-pool, ...)
 - ▶ only affects clients who send a Client GUID (SMB \geq 2.1)
 - ▶ possibly make this tunable-off(?)

Multi-Channel - Samba - Status

- ▶ preparation: rewrite messaging using unix dgm sockets with sendmsg/recvmmsg [DONE]
- ▶ add fd-passing [ess.DONE]
- ▶ transfer connection in NEGPROT (based on ClientGUID) [ess.DONE]
- ▶ implement session bind [ess.DONE]
- ▶ change smbd behaviour upon client disconnect (don't always exit!) [WIP]
- ▶ implement channel epoch numbers [WIP]
- ▶ implement interface discovery [WIP]

Multi-Channel - Samba - Status

- ▶ preparation: rewrite messaging using unix dgm sockets with sendmsg/recvmmsg [DONE]
- ▶ add fd-passing [ess.DONE]
- ▶ transfer connection in NEGPROT (based on ClientGUID) [ess.DONE]
- ▶ implement session bind [ess.DONE]
- ▶ change smbd behaviour upon client disconnect (don't always exit!) [WIP]
- ▶ implement channel epoch numbers [WIP]
- ▶ implement interface discovery [WIP]

Multi-Channel - Samba - Details

- ▶ Samba 4.0 / durable handles: introduced `smbXsrv_` structures
 - ▶ `smbXsrv_connection` in `smbd` represents client
 - ▶ `smbd_server_connection` (FSA link) ↔ `smbXsrv_connection`
 - ▶ `session_table`
 - ▶ `tcon_table`
 - ▶ `open_table`
- ▶ `master/wip/multi-channel`:
 - ▶ `smbXsrv_client` represents client in `smbd`:
 - ▶ `server_id`
 - ▶ `smbd_server_connection` (FSA link) ↔ `smbXsrv_client`
 - ▶ `client_guid`
 - ▶ `session_table`
 - ▶ `tcon_table`
 - ▶ `open_table`
 - ▶ `connections`

Multi-Channel - Samba - Details

- ▶ Samba 4.0 / durable handles: introduced `smbXsrv_` structures
 - ▶ `smbXsrv_connection` in `smbd` represents client
 - ▶ `smbd_server_connection` (FSA link) ↔ `smbXsrv_connection`
 - ▶ `session_table`
 - ▶ `tcon_table`
 - ▶ `open_table`
- ▶ `master/wip/multi-channel:`
 - ▶ `smbXsrv_client` represents client in `smbd`:
 - ▶ `server_id`
 - ▶ `smbd_server_connection` (FSA link) ↔ `smbXsrv_client`
 - ▶ `client_guid`
 - ▶ `session_table`
 - ▶ `tcon_table`
 - ▶ `open_table`
 - ▶ `connections`

Multi-Channel - Samba - Details

- ▶ Samba 4.0 / durable handles: introduced `smbXsrv_` structures
 - ▶ `smbXsrv_connection` in `smbd` represents client
 - ▶ `smbd_server_connection` (FSA link) ↔ `smbXsrv_connection`
 - ▶ `session_table`
 - ▶ `tcon_table`
 - ▶ `open_table`
- ▶ `master/wip/multi-channel`:
 - ▶ `smbXsrv_client` represents client in `smbd`:
 - ▶ `server_id`
 - ▶ `smbd_server_connection` (FSA link) ↔ `smbXsrv_client`
 - ▶ `client_guid`
 - ▶ `session_table`
 - ▶ `tcon_table`
 - ▶ `open_table`
 - ▶ `connections`

Multi-Channel - Samba - Details

- ▶ 4.0:
 - ▶ smbXsrv_session
 - ▶ smbXsrv_connection
 - ▶ channels (just one)
 - ▶ smbXsrv_channel
 - ▶ server_id
 - ▶ signing_key
- ▶ master/wip/multi-channel:
 - ▶ smbXsrv_session
 - ▶ smbXsrv_client
 - ▶ channels (multiple)
 - ▶ smbXsrv_channel
 - ▶ server_id
 - ▶ signing_key
 - ▶ smbXsrv_connection

Multi-Channel - Samba - Details

▶ 4.0:

- ▶ `smbXsrv_session`
 - ▶ `smbXsrv_connection`
 - ▶ `channels` (just one)
- ▶ `smbXsrv_channel`
 - ▶ `server_id`
 - ▶ `signing_key`

▶ `master/wip/multi-channel:`

- ▶ `smbXsrv_session`
 - ▶ `smbXsrv_client`
 - ▶ `channels` (multiple)
- ▶ `smbXsrv_channel`
 - ▶ `server_id`
 - ▶ `signing_key`
 - ▶ `smbXsrv_connection`

Multi-Channel - Samba - Details

- ▶ 4.0:
 - ▶ smbXsrv_session
 - ▶ smbXsrv_connection
 - ▶ channels (just one)
 - ▶ smbXsrv_channel
 - ▶ server_id
 - ▶ signing_key
- ▶ master/wip/multi-channel:
 - ▶ smbXsrv_session
 - ▶ smbXsrv_client
 - ▶ channels (multiple)
 - ▶ smbXsrv_channel
 - ▶ server_id
 - ▶ signing_key
 - ▶ smbXsrv_connection

Multi-Channel - Notes On Testing

- ▶ Testing with Windows: need interface discovery (WIP)
- ▶ unit testing - smbtorture: multi channel tests exist
- ▶ selftest: `socket_wrapper`
 - ▶ `socket_wrapper` externalized: cwrap, the wrapper project
 - ▶ <http://cwrap.org>
 - ▶ WIP: teach `socket_wrapper` fd-passing

Multi-Channel - Notes On Testing

- ▶ Testing with Windows: need interface discovery (WIP)
- ▶ unit testing - smbtorture: multi channel tests exist
- ▶ selftest: `socket_wrapper`
 - ▶ `socket_wrapper` externalized: cwrap, the wrapper project
 - ▶ <http://cwrap.org>
 - ▶ WIP: teach `socket_wrapper` fd-passing

Multi-Channel - Consequences

- ▶ Opportunity to do durable handles *cross-protocol*! (SMB \geq 2.1)
 - ▶ Keep file open in `smbd` after client has been disconnected.
 - ▶ Reconnecting client's connection is passed to the original `smbd`.
- ▶ Prerequisite for work on SMB Direct (RDMA)

Multi-Channel - Consequences

- ▶ Opportunity to do durable handles *cross-protocol!* (SMB \geq 2.1)
 - ▶ Keep file open in `smbd` after client has been disconnected.
 - ▶ Reconnecting client's connection is passed to the original `smbd`.
- ▶ Prerequisite for work on SMB Direct (RDMA)

Multi-Channel - Consequences

- ▶ Opportunity to do durable handles *cross-protocol!* (SMB \geq 2.1)
 - ▶ Keep file open in `smbd` after client has been disconnected.
 - ▶ Reconnecting client's connection is passed to the original `smbd`.
- ▶ Prerequisite for work on SMB Direct (RDMA)

RDMA / SMB Direct

SMB Direct (RDMA)

- ▶ windows:
 - ▶ requires multi-channel
 - ▶ start with TCP, bind an RDMA channel
 - ▶ reads and writes use RDMA write/read
 - ▶ protocol/metadata via send/receive
- ▶ wireshark dissector: [DONE]
- ▶ samba (TODO):
 - ▶ prereq: multi-channel / fd-passing
 - ▶ buffer / transport abstractions [TODO]
 - ▶ central daemon (or kernel module) to serve as RDMA "proxy"
(libraries: not fork safe and no fd-passing)

SMB Direct (RDMA)

- ▶ windows:
 - ▶ requires multi-channel
 - ▶ start with TCP, bind an RDMA channel
 - ▶ reads and writes use RDMA write/read
 - ▶ protocol/metadata via send/receive
- ▶ wireshark dissector: [DONE]
- ▶ samba (TODO):
 - ▶ prereq: multi-channel / fd-passing
 - ▶ buffer / transport abstractions [TODO]
 - ▶ central daemon (or kernel module) to serve as RDMA "proxy"
(libraries: not fork safe and no fd-passing)

SMB Direct (RDMA)

- ▶ windows:
 - ▶ requires multi-channel
 - ▶ start with TCP, bind an RDMA channel
 - ▶ reads and writes use RDMA write/read
 - ▶ protocol/metadata via send/receive
- ▶ wireshark dissector: [DONE]
- ▶ samba (TODO):
 - ▶ prereq: multi-channel / fd-passing
 - ▶ buffer / transport abstractions [TODO]
 - ▶ central daemon (or kernel module) to serve as RDMA "proxy"
(libraries: not fork safe and no fd-passing)

SMB Direct (RDMA)

- ▶ windows:
 - ▶ requires multi-channel
 - ▶ start with TCP, bind an RDMA channel
 - ▶ reads and writes use RDMA write/read
 - ▶ protocol/metadata via send/receive
- ▶ wireshark dissector: [DONE]
- ▶ samba (TODO):
 - ▶ prereq: multi-channel / fd-passing
 - ▶ buffer / transport abstractions [TODO]
 - ▶ central daemon (or kernel module) to serve as RDMA "proxy"
(libraries: not fork safe and no fd-passing)

SMB Direct (RDMA) - Plan

- ▶ `smbd-d` (rdma proxy daemon)
 - ▶ listens on unix domain socket (`/var/lib/smbd-d/socket`)
 - ▶ listens for RDMA connection (as told by main `smbd`)
- ▶ main `smbd`:
 - ▶ listens for TCP connections
 - ▶ connects to `smbd-d-socket`
 - ▶ request rdma-interfaces, tell `smbd-d` on which to listen
 - ▶ "accepts" new smb-direct connections on `smbd-d-socket`

SMB Direct (RDMA) - Plan

- ▶ `smbd-d` (rdma proxy daemon)
 - ▶ listens on unix domain socket (`/var/lib/smbd-d/socket`)
 - ▶ listens for RDMA connection (as told by main `smbd`)
- ▶ main `smbd`:
 - ▶ listens for TCP connections
 - ▶ connects to `smbd-d-socket`
 - ▶ request rdma-interfaces, tell `smbd-d` on which to listen
 - ▶ "accepts" new smb-direct connections on `smbd-d-socket`

SMB Direct (RDMA) - Plan

- ▶ client
 - ▶ connects via TCP → smbd forks child smbd (c)
 - ▶ connects via RDMA to smbd-d
- ▶ smbd-d
 - ▶ creates socket-pair as rdma-proxy-channel
 - ▶ passes one end of socket-pair to main smbd for accept
 - ▶ sends smb direct packages over proxy-channel
- ▶ main smbd
 - ▶ upon receiving NegProt: pass proxy-socket to c based on ClientGUID
- ▶ c
 - ▶ continues proxy-communication with smbd-d
- ▶ For `rdma_read` and `rdma_write`:
 - ▶ c and smbd-d establish shared memory area

SMB Direct (RDMA) - Plan

- ▶ client
 - ▶ connects via TCP → smbd forks child smbd (c)
 - ▶ connects via RDMA to smbd-d
- ▶ smbd-d
 - ▶ creates socket-pair as rdma-proxy-channel
 - ▶ passes one end of socket-pair to main smbd for accept
 - ▶ sends smb direct packages over proxy-channel
- ▶ main smbd
 - ▶ upon receiving NegProt: pass proxy-socket to c based on ClientGUID
- ▶ c
 - ▶ continues proxy-communication with smbd-d
- ▶ For `rdma_read` and `rdma_write`:
 - ▶ c and smbd-d establish shared memory area

SMB Direct (RDMA) - Plan

- ▶ client
 - ▶ connects via TCP → smbd forks child smbd (c)
 - ▶ connects via RDMA to smbd-d
- ▶ smbd-d
 - ▶ creates socket-pair as rdma-proxy-channel
 - ▶ passes one end of socket-pair to main smbd for accept
 - ▶ sends smb direct packages over proxy-channel
- ▶ main smbd
 - ▶ upon receiving NegProt: pass proxy-socket to c based on ClientGUID
- ▶ c
 - ▶ continues proxy-communication with smbd-d
- ▶ For `rdma_read` and `rdma_write`:
 - ▶ c and smbd-d establish shared memory area

Clustering

Clustering Concepts (Windows)

- ▶ Cluster:
 - ▶ (“traditional”) failover cluster (active-passive)
 - ▶ protocol: `SMB2_SHARE_CAP_CLUSTER`
 - ▶ Windows:
 - ▶ runs off a cluster (failover) volume
 - ▶ offers the Witness service
- ▶ Scale-Out (SOFS):
 - ▶ scale-out cluster (all-active!)
 - ▶ protocol: `SMB2_SHARE_CAP_SCALEOUT`
 - ▶ no client caching
 - ▶ Windows: runs off a cluster shared volume (implies cluster)
- ▶ Continuous Availability (CA):
 - ▶ transparent failover, persistent handles
 - ▶ protocol: `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`
 - ▶ can independently turned on on any cluster share (failover or scale-out)
 - ▶ ⇒ changed client retry behaviour!

Clustering Concepts (Windows)

▶ Cluster:

- ▶ (“traditional”) failover cluster (active-passive)
- ▶ protocol: `SMB2_SHARE_CAP_CLUSTER`
- ▶ Windows:
 - ▶ runs off a cluster (failover) volume
 - ▶ offers the Witness service

▶ Scale-Out (SOFS):

- ▶ scale-out cluster (all-active!)
- ▶ protocol: `SMB2_SHARE_CAP_SCALEOUT`
- ▶ no client caching
- ▶ Windows: runs off a cluster shared volume (implies cluster)

▶ Continuous Availability (CA):

- ▶ transparent failover, persistent handles
- ▶ protocol: `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`
- ▶ can independently turned on on any cluster share (failover or scale-out)
- ▶ ⇒ changed client retry behaviour!

Clustering Concepts (Windows)

▶ Cluster:

- ▶ (“traditional”) failover cluster (active-passive)
- ▶ protocol: `SMB2_SHARE_CAP_CLUSTER`
- ▶ Windows:
 - ▶ runs off a cluster (failover) volume
 - ▶ offers the Witness service

▶ Scale-Out (SOFS):

- ▶ scale-out cluster (all-active!)
- ▶ protocol: `SMB2_SHARE_CAP_SCALEOUT`
- ▶ no client caching
- ▶ Windows: runs off a cluster shared volume (implies cluster)

▶ Continuous Availability (CA):

- ▶ transparent failover, persistent handles
- ▶ protocol: `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`
- ▶ can independently turned on on any cluster share (failover or scale-out)
- ▶ ⇒ changed client retry behaviour!

Clustering Concepts (Windows)

- ▶ Cluster:
 - ▶ (“traditional”) failover cluster (active-passive)
 - ▶ protocol: `SMB2_SHARE_CAP_CLUSTER`
 - ▶ Windows:
 - ▶ runs off a cluster (failover) volume
 - ▶ offers the Witness service
- ▶ Scale-Out (SOFS):
 - ▶ scale-out cluster (all-active!)
 - ▶ protocol: `SMB2_SHARE_CAP_SCALEOUT`
 - ▶ no client caching
 - ▶ Windows: runs off a cluster shared volume (implies cluster)
- ▶ Continuous Availability (CA):
 - ▶ transparent failover, persistent handles
 - ▶ protocol: `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`
 - ▶ can independently turned on on any cluster share (failover or scale-out)
 - ▶ ⇒ changed client retry behaviour!

Clustering – Controlling Flags from Windows

- ▶ a share on a cluster carries
 - ▶ `SMB2_SHARE_CAP_CLUSTER` \Leftrightarrow the shared FS is a cluster volume.
- ▶ a share on a cluster carries
 - ▶ `SMB2_SHARE_CAP_SCALEOUT` \Leftrightarrow the shared FS is a CSV
 - ▶ implies `SMB2_SHARE_CAP_CLUSTER`
- ▶ independently settable on a clustered share:
 - ▶ `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`
 - ▶ implies `SMB2_SHARE_CAP_CLUSTER`

Clustering – Controlling Flags from Windows

- ▶ a share on a cluster carries
 - ▶ `SMB2_SHARE_CAP_CLUSTER` \Leftrightarrow the shared FS is a cluster volume.
- ▶ a share on a cluster carries
 - ▶ `SMB2_SHARE_CAP_SCALEOUT` \Leftrightarrow the shared FS is a CSV
 - ▶ implies `SMB2_SHARE_CAP_CLUSTER`
- ▶ independently settable on a clustered share:
 - ▶ `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`
 - ▶ implies `SMB2_SHARE_CAP_CLUSTER`

Clustering – Controlling Flags from Windows

- ▶ a share on a cluster carries
 - ▶ `SMB2_SHARE_CAP_CLUSTER` \Leftrightarrow the shared FS is a cluster volume.
- ▶ a share on a cluster carries
 - ▶ `SMB2_SHARE_CAP_SCALEOUT` \Leftrightarrow the shared FS is a CSV
 - ▶ implies `SMB2_SHARE_CAP_CLUSTER`
- ▶ independently settable on a clustered share:
 - ▶ `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`
 - ▶ implies `SMB2_SHARE_CAP_CLUSTER`

Clustering – Controlling Flags from Windows

- ▶ a share on a cluster carries
 - ▶ `SMB2_SHARE_CAP_CLUSTER` \Leftrightarrow the shared FS is a cluster volume.
- ▶ a share on a cluster carries
 - ▶ `SMB2_SHARE_CAP_SCALEOUT` \Leftrightarrow the shared FS is a CSV
 - ▶ implies `SMB2_SHARE_CAP_CLUSTER`
- ▶ independently settable on a clustered share:
 - ▶ `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`
 - ▶ implies `SMB2_SHARE_CAP_CLUSTER`

Clustering – Server Behaviour

- ▶ `SMB2_SHARE_CAP_CLUSTER`:
 - ▶ run witness service (RPC)
 - ▶ client can register and get notified about resource changes
- ▶ `SMB2_SHARE_CAP_SCALEOUT`:
 - ▶ do not grant batch oplocks, write leases, handle leases
 - ▶ \Rightarrow no durable handles unless also CA
- ▶ `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`:
 - ▶ offer persistent handles
 - ▶ timeout from durable v2 request

- ▶ **SMB2_SHARE_CAP_CLUSTER:**
 - ▶ run witness service (RPC)
 - ▶ client can register and get notified about resource changes
- ▶ **SMB2_SHARE_CAP_SCALEOUT:**
 - ▶ do not grant batch oplocks, write leases, handle leases
 - ▶ ⇒ no durable handles unless also CA
- ▶ **SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY:**
 - ▶ offer persistent handles
 - ▶ timeout from durable v2 request

- ▶ **SMB2_SHARE_CAP_CLUSTER:**
 - ▶ run witness service (RPC)
 - ▶ client can register and get notified about resource changes
- ▶ **SMB2_SHARE_CAP_SCALEOUT:**
 - ▶ do not grant batch oplocks, write leases, handle leases
 - ▶ \Rightarrow no durable handles unless also CA
- ▶ **SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY:**
 - ▶ offer persistent handles
 - ▶ timeout from durable v2 request

- ▶ `SMB2_SHARE_CAP_CLUSTER`:
 - ▶ run witness service (RPC)
 - ▶ client can register and get notified about resource changes
- ▶ `SMB2_SHARE_CAP_SCALEOUT`:
 - ▶ do not grant batch oplocks, write leases, handle leases
 - ▶ \Rightarrow no durable handles unless also CA
- ▶ `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`:
 - ▶ offer persistent handles
 - ▶ timeout from durable v2 request

Clustering – Client Behaviour (Win8)

- ▶ `SMB2_SHARE_CAP_CLUSTER`:
 - ▶ clients happily work if witness is not available
- ▶ `SMB2_SHARE_CAP_SCALEOUT`:
 - ▶ clients happily connect if `CLUSTER` is not set.
 - ▶ clients DO request oplocks/leases/durable handles
 - ▶ clients are not confused if they get these
- ▶ `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`:
 - ▶ clients happily connect if `CLUSTER` is not set.
 - ▶ clients typically request persistent handle with RWH lease
- ▶ Note:
Win8 sends `SMB2_FLAGS_REPLAY_OPERATION` in writes and reads
(from 2nd in a row)
⇔
The server announces `SMB2_CAP_PERSISTENT_HANDLES`.

Clustering – Client Behaviour (Win8)

▶ SMB2_SHARE_CAP_CLUSTER:

- ▶ clients happily work if witness is not available

▶ SMB2_SHARE_CAP_SCALEOUT:

- ▶ clients happily connect if CLUSTER is not set.
- ▶ clients DO request oplocks/leases/durable handles
- ▶ clients are not confused if they get these

▶ SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY:

- ▶ clients happily connect if CLUSTER is not set.
- ▶ clients typically request persistent handle with RWH lease

▶ Note:

Win8 sends SMB2_FLAGS_REPLAY_OPERATION in writes and reads
(from 2nd in a row)

⇔

The server announces SMB2_CAP_PERSISTENT_HANDLES.

Clustering – Client Behaviour (Win8)

- ▶ **SMB2_SHARE_CAP_CLUSTER:**
 - ▶ clients happily work if witness is not available
- ▶ **SMB2_SHARE_CAP_SCALEOUT:**
 - ▶ clients happily connect if **CLUSTER** is not set.
 - ▶ clients **DO** request oplocks/leases/durable handles
 - ▶ clients are not confused if they get these
- ▶ **SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY:**
 - ▶ clients happily connect if **CLUSTER** is not set.
 - ▶ clients typically request persistent handle with RWH lease
- ▶ **Note:**
Win8 sends **SMB2_FLAGS_REPLAY_OPERATION** in writes and reads
(from 2nd in a row)
⇔
The server announces **SMB2_CAP_PERSISTENT_HANDLES**.

Clustering – Client Behaviour (Win8)

- ▶ **SMB2_SHARE_CAP_CLUSTER:**
 - ▶ clients happily work if witness is not available
- ▶ **SMB2_SHARE_CAP_SCALEOUT:**
 - ▶ clients happily connect if **CLUSTER** is not set.
 - ▶ clients **DO** request oplocks/leases/durable handles
 - ▶ clients are not confused if they get these
- ▶ **SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY:**
 - ▶ clients happily connect if **CLUSTER** is not set.
 - ▶ clients typically request persistent handle with **RWH** lease

▶ Note:

Win8 sends **SMB2_FLAGS_REPLAY_OPERATION** in writes and reads
(from 2nd in a row)



The server announces **SMB2_CAP_PERSISTENT_HANDLES**.

Clustering – Client Behaviour (Win8)

- ▶ `SMB2_SHARE_CAP_CLUSTER`:
 - ▶ clients happily work if witness is not available
- ▶ `SMB2_SHARE_CAP_SCALEOUT`:
 - ▶ clients happily connect if `CLUSTER` is not set.
 - ▶ clients DO request oplocks/leases/durable handles
 - ▶ clients are not confused if they get these
- ▶ `SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY`:
 - ▶ clients happily connect if `CLUSTER` is not set.
 - ▶ clients typically request persistent handle with RWH lease
- ▶ Note:
Win8 sends `SMB2_FLAGS_REPLAY_OPERATION` in writes and reads
(from 2nd in a row)
⇔
The server announces `SMB2_CAP_PERSISTENT_HANDLES`.

Clustering – Client Behaviour (Win8) : Retries

- ▶ Test: Win8 against slightly pimped Samba (2 IPs)
- ▶ \Rightarrow essentially two different retry characteristics: CA \leftrightarrow non-CA
- ▶ non-CA-case
 - ▶ 3 consecutive attempt rounds:
 - ▶ for each of the two IPs:
 - arp IP
 - three tcp syn attempts to IP with 0.5 sec breaks
 - ▶ \Rightarrow some 2.1 seconds for 1 round
 - ▶ between attempts:
 - ▶ dns, ping, arp ... 5.8 seconds
 - ▶ \Rightarrow 18 seconds
- ▶ CA-Case
 - ▶ retries attempt rounds from above for 14 minutes

Clustering – Client Behaviour (Win8) : Retries

- ▶ Test: Win8 against slightly pimped Samba (2 IPs)
- ▶ \Rightarrow essentially two different retry characteristics: CA \leftrightarrow non-CA
- ▶ non-CA-case
 - ▶ 3 consecutive attempt rounds:
 - ▶ for each of the two IPs:
 - arp IP
 - three tcp syn attempts to IP with 0.5 sec breaks
 - ▶ \Rightarrow some 2.1 seconds for 1 round
 - ▶ between attempts:
 - ▶ dns, ping, arp ... 5.8 seconds
 - ▶ \Rightarrow 18 seconds
- ▶ CA-Case
 - ▶ retries attempt rounds from above for 14 minutes

Clustering – Client Behaviour (Win8) : Retries

- ▶ Test: Win8 against slightly pimped Samba (2 IPs)
- ▶ \Rightarrow essentially two different retry characteristics: CA \leftrightarrow non-CA
- ▶ non-CA-case
 - ▶ 3 consecutive attempt rounds:
 - ▶ for each of the two IPs:
 - arp IP
 - three tcp syn attempts to IP with 0.5 sec breaks
 - ▶ \Rightarrow some 2.1 seconds for 1 round
 - ▶ between attempts:
 - ▶ dns, ping, arp ... 5.8 seconds
 - ▶ \Rightarrow 18 seconds
- ▶ CA-Case
 - ▶ retries attempt rounds from above for 14 minutes

Clustering – Client Behaviour (Win8) : Retries

- ▶ Test: Win8 against slightly pimped Samba (2 IPs)
- ▶ \Rightarrow essentially two different retry characteristics: CA \leftrightarrow non-CA
- ▶ non-CA-case
 - ▶ 3 consecutive attempt rounds:
 - ▶ for each of the two IPs:
 - arp IP
 - three tcp syn attempts to IP with 0.5 sec breaks
 - ▶ \Rightarrow some 2.1 seconds for 1 round
 - ▶ between attempts:
 - ▶ dns, ping, arp ... 5.8 seconds
 - ▶ \Rightarrow 18 seconds
- ▶ CA-Case
 - ▶ retries attempt rounds from above for 14 minutes

Clustering with Samba/CTDB

- ▶ all-active SMB-cluster with Samba and CTDB...
...since 2007! ☺
- ▶ transparent for the client
 - ▶ CTDB:
 - ▶ metadata and messaging engine for Samba in a cluster
 - ▶ plus cluster resource manager (IPs, services...)
 - ▶ client only sees one "big" SMB server
 - ▶ we could not change the client!...
 - ▶ works "well enough"
- ▶ challenge:
 - ▶ how to integrate SMB3 clustering with Samba/CTDB
 - ▶ good: rather orthogonal
 - ▶ ctdb-clustering transparent mostly due to management

Clustering with Samba/CTDB

- ▶ all-active SMB-cluster with Samba and CTDB...
...since 2007! ☺
- ▶ transparent for the client
 - ▶ CTDB:
 - ▶ metadata and messaging engine for Samba in a cluster
 - ▶ plus cluster resource manager (IPs, services...)
 - ▶ client only sees one “big” SMB server
 - ▶ we could not change the client!...
 - ▶ works “well enough”
- ▶ challenge:
 - ▶ how to integrate SMB3 clustering with Samba/CTDB
 - ▶ good: rather orthogonal
 - ▶ ctdb-clustering transparent mostly due to management

Clustering with Samba/CTDB

- ▶ all-active SMB-cluster with Samba and CTDB...
...since 2007! ☺
- ▶ transparent for the client
 - ▶ CTDB:
 - ▶ metadata and messaging engine for Samba in a cluster
 - ▶ plus cluster resource manager (IPs, services...)
 - ▶ client only sees one “big” SMB server
 - ▶ we could not change the client!...
 - ▶ works “well enough”
- ▶ challenge:
 - ▶ how to integrate SMB3 clustering with Samba/CTDB
 - ▶ good: rather orthogonal
 - ▶ ctdb-clustering transparent mostly due to management

Clustering with Samba/CTDB

- ▶ all-active SMB-cluster with Samba and CTDB...
...since 2007! ☺
- ▶ transparent for the client
 - ▶ CTDB:
 - ▶ metadata and messaging engine for Samba in a cluster
 - ▶ plus cluster resource manager (IPs, services...)
 - ▶ client only sees one “big” SMB server
 - ▶ we could not change the client!...
 - ▶ works “well enough”
- ▶ challenge:
 - ▶ how to integrate SMB3 clustering with Samba/CTDB
 - ▶ good: rather orthogonal
 - ▶ ctdb-clustering transparent mostly due to management

Clustering with Samba/CTDB

- ▶ all-active SMB-cluster with Samba and CTDB...
...since 2007! ☺
- ▶ transparent for the client
 - ▶ CTDB:
 - ▶ metadata and messaging engine for Samba in a cluster
 - ▶ plus cluster resource manager (IPs, services...)
 - ▶ client only sees one “big” SMB server
 - ▶ we could not change the client!...
 - ▶ works “well enough”
- ▶ challenge:
 - ▶ how to integrate SMB3 clustering with Samba/CTDB
 - ▶ good: rather orthogonal
 - ▶ ctdb-clustering transparent mostly due to management

- ▶ Service Witness Protocol: an RPC service
 - ▶ monitoring of availability of resources (shares, NICs)
 - ▶ server asks client to move to another resource
- ▶ remember:
 - ▶ available on a Windows SMB3 share \Leftrightarrow SMB2_SHARE_CAP_CLUSTER
 - ▶ but clients happily connect w/o witness
- ▶ status in Samba [WIP]:
 - ▶ async RPC: [WIP] good progress (\Rightarrow Metze's talk)
 - ▶ wireshark dissector: [ess.DONE]
 - ▶ client: in rpcclient [DONE]
 - ▶ server: dummy PoC / tracer bullet implementation [DONE]
 - ▶ CTDB: changes / integration needed [TODO]

- ▶ Service Witness Protocol: an RPC service
 - ▶ monitoring of availability of resources (shares, NICs)
 - ▶ server asks client to move to another resource
- ▶ remember:
 - ▶ available on a Windows SMB3 share \Leftrightarrow SMB2_SHARE_CAP_CLUSTER
 - ▶ but clients happily connect w/o witness
- ▶ status in Samba [WIP]:
 - ▶ async RPC: [WIP] good progress (\Rightarrow Metze's talk)
 - ▶ wireshark dissector: [ess.DONE]
 - ▶ client: in rpcclient [DONE]
 - ▶ server: dummy PoC / tracer bullet implementation [DONE]
 - ▶ CTDB: changes / integration needed [TODO]

- ▶ Service Witness Protocol: an RPC service
 - ▶ monitoring of availability of resources (shares, NICs)
 - ▶ server asks client to move to another resource
- ▶ remember:
 - ▶ available on a Windows SMB3 share \Leftrightarrow SMB2_SHARE_CAP_CLUSTER
 - ▶ but clients happily connect w/o witness
- ▶ status in Samba [WIP]:
 - ▶ async RPC: [WIP] good progress (\Rightarrow Metze's talk)
 - ▶ wireshark dissector: [ess.DONE]
 - ▶ client: in `rpcclient` [DONE]
 - ▶ server: dummy PoC / tracer bullet implementation [DONE]
 - ▶ CTDB: changes / integration needed [TODO]

- ▶ Service Witness Protocol: an RPC service
 - ▶ monitoring of availability of resources (shares, NICs)
 - ▶ server asks client to move to another resource
- ▶ remember:
 - ▶ available on a Windows SMB3 share \Leftrightarrow SMB2_SHARE_CAP_CLUSTER
 - ▶ but clients happily connect w/o witness
- ▶ status in Samba [WIP]:
 - ▶ async RPC: [WIP] good progress (\Rightarrow Metze's talk)
 - ▶ wireshark dissector: [ess.DONE]
 - ▶ client: in `rpcclient` [DONE]
 - ▶ server: dummy PoC / tracer bullet implementation [DONE]
 - ▶ CTDB: changes / integration needed [TODO]



Questions?

Michael Adam

ma@sernet.de / obnox@samba.org

→ [SerNet](#) sponsor booth

