

SMB3, Clustering, and Samba...

...The Road To Hyper-V

Michael Adam

Samba Team / SerNet

2014-05-14



**CHILLI
SAMBA
FLAVOUR**

'14



**CHILLI
SAMBA
FLAVOUR**

'14



**CHILLI
SAMBA
FLAVOUR**

'14

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]

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]

SMB 3.0

- ▶ new crypto (signing, transport encryption)
- ▶ persistent file handles
- ▶ multi-channel
- ▶ RDMA transport (SMB direct)
- ▶ storage features
- ▶ clustering
 - ▶ witness
 - ▶ transparent failover (continuous availability)
 - ▶ all-active (scale-out)

SMB 3.0

- ▶ new crypto (signing, transport encryption)
- ▶ persistent file handles
- ▶ multi-channel
- ▶ RDMA transport (SMB direct)
- ▶ storage features
- ▶ clustering
 - ▶ witness
 - ▶ transparent failover (continuous availability)
 - ▶ all-active (scale-out)

SMB3 - Goals

- ▶ fault tolerance / reliability
- ▶ performance / throughput / scaling
- ▶ focus on support for server workloads
(as opposed to workstation workloads)
- ▶ especially support for:
 - ▶ Hyper-V
 - ▶ MS-SQL
- ▶ goals:
 - ▶ replace block storage in data center
 - ▶ block (SCSI) over SMB

SMB3 - Goals

- ▶ fault tolerance / reliability
- ▶ performance / throughput / scaling
- ▶ focus on support for server workloads
(as opposed to workstation workloads)
- ▶ especially support for:
 - ▶ Hyper-V
 - ▶ MS-SQL
- ▶ goals:
 - ▶ replace block storage in data center
 - ▶ block (SCSI) over SMB

Requirements for Hyper-V

- ▶ minimum requirements:
 - ▶ SMB 3.0
 - ▶ is that really all??? - maybe resilient file handles..
- ▶ desired features:
 - ▶ cluster (≥ 2 nodes)
 - ▶ CA / persistent handles
 - ▶ RDMA / SMB direct
 - ▶ multi channel

Requirements for Hyper-V

- ▶ minimum requirements:
 - ▶ SMB 3.0
 - ▶ is that really all??? - maybe resilient file handles..
- ▶ desired features:
 - ▶ cluster (≥ 2 nodes)
 - ▶ CA / persistent handles
 - ▶ RDMA / SMB direct
 - ▶ multi channel

Requirements for Hyper-V

- ▶ minimum requirements:
 - ▶ SMB 3.0
 - ▶ is that really all??? - maybe resilient file handles..
- ▶ desired features:
 - ▶ cluster (≥ 2 nodes)
 - ▶ CA / persistent handles
 - ▶ RDMA / SMB direct
 - ▶ multi channel

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
- ▶ Samba 4.1
 - ▶ SMB 3.0.2: basic

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
- ▶ Samba 4.1
 - ▶ SMB 3.0.2: basic



**CHILLI
SAMBA
FLAVOUR**

'14

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

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
 - ▶ ⇒ no durable handles unless also CA
- ▶ **SMB2_SHARE_CAP_CONTINUOUS_AVAILABILITY:**
 - ▶ offer persistent handles
 - ▶ timeout from durable v2 request

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

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

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)
- ▶ Server-Matrix (on/off):
 - ▶ persistent handle cap
 - ▶ durable handles
 - ▶ cluster share cap
 - ▶ scale out cap
 - ▶ ca share cap
- ▶ The test:
 - ▶ connect to share with explorer
 - ▶ start copying file (2G)
 - ▶ kill smbd
 - ▶ wait for the client to pop up an error dialog
 - ▶ click cancel
 - ▶ stop capture

Clustering – Client Behaviour (Win8) : Retries

- ▶ Test: Win8 against slightly pimped Samba (2 IPs)
- ▶ Server-Matrix (on/off):
 - ▶ persistent handle cap
 - ▶ durable handles
 - ▶ cluster share cap
 - ▶ scale out cap
 - ▶ ca share cap
- ▶ The test:
 - ▶ connect to share with explorer
 - ▶ start copying file (2G)
 - ▶ kill smbd
 - ▶ wait for the client to pop up an error dialog
 - ▶ click cancel
 - ▶ stop capture

Clustering – Client Behaviour (Win8) : Retries

- ▶ Test: Win8 against slightly pimped Samba (2 IPs)
- ▶ Server-Matrix (on/off):
 - ▶ persistent handle cap
 - ▶ durable handles
 - ▶ cluster share cap
 - ▶ scale out cap
 - ▶ ca share cap
- ▶ The test:
 - ▶ connect to share with explorer
 - ▶ start copying file (2G)
 - ▶ kill smbd
 - ▶ wait for the client to pop up an error dialog
 - ▶ click cancel
 - ▶ stop capture

Clustering – Client Behaviour (Win8) : Retries

- ▶ Test: Win8 against slightly pimped Samba (2 IPs)
- ▶ Server-Matrix (on/off):
 - ▶ persistent handle cap
 - ▶ durable handles
 - ▶ cluster share cap
 - ▶ scale out cap
 - ▶ ca share cap
- ▶ The test:
 - ▶ connect to share with explorer
 - ▶ start copying file (2G)
 - ▶ kill smbd
 - ▶ wait for the client to pop up an error dialog
 - ▶ click cancel
 - ▶ stop capture

Clustering – Client Behaviour (Win8) : Retries

- ▶ only two different retry characteristics: CA ↔ 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
 - ▶ ⇒ some 2.1 seconds for 1 round
 - ▶ between attempts:
 - ▶ dns, ping, arp ... 5.8 seconds
 - ▶ ⇒ 18 seconds
- ▶ CA-Case
 - ▶ retries attempt rounds from above for 14 minutes

Clustering – Client Behaviour (Win8) : Retries

- ▶ only two different retry characteristics: CA ↔ 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
 - ▶ ⇒ some 2.1 seconds for 1 round
 - ▶ between attempts:
 - ▶ dns, ping, arp ... 5.8 seconds
 - ▶ ⇒ 18 seconds
- ▶ CA-Case
 - ▶ retries attempt rounds from above for 14 minutes

Clustering – Client Behaviour (Win8) : Retries

- ▶ only two different retry characteristics: CA ↔ 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
 - ▶ ⇒ some 2.1 seconds for 1 round
 - ▶ between attempts:
 - ▶ dns, ping, arp ... 5.8 seconds
 - ▶ ⇒ 18 seconds
- ▶ CA-Case
 - ▶ retries attempt rounds from above for 14 minutes

Clustering – Client Behaviour (Win8) : Retries

- ▶ only two different retry characteristics: CA ↔ 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
 - ▶ ⇒ some 2.1 seconds for 1 round
 - ▶ between attempts:
 - ▶ dns, ping, arp ... 5.8 seconds
 - ▶ ⇒ 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

Witness Service

- ▶ 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 (Metze, Gregor Beck)]:
 - ▶ async RPC: WIP, good progress (\Rightarrow Metze's talk)
 - ▶ wireshark dissector: essentially done
 - ▶ client: in rpcclient - done
 - ▶ server: dummy PoC / tracer bullet implementation done
 - ▶ CTDB: changes / integration needed

Witness Service

- ▶ 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 (Metze, Gregor Beck)]:
 - ▶ async RPC: WIP, good progress (\Rightarrow Metze's talk)
 - ▶ wireshark dissector: essentially done
 - ▶ client: in rpcclient - done
 - ▶ server: dummy PoC / tracer bullet implementation done
 - ▶ CTDB: changes / integration needed

Witness Service

- ▶ 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 (Metze, Gregor Beck)]:
 - ▶ async RPC: WIP, good progress (\Rightarrow Metze's talk)
 - ▶ wireshark dissector: essentially done
 - ▶ client: in `rpcclient` - done
 - ▶ server: dummy PoC / tracer bullet implementation done
 - ▶ CTDB: changes / integration needed

Witness Service

- ▶ 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 (Metze, Gregor Beck)]:
 - ▶ async RPC: WIP, good progress (\Rightarrow Metze's talk)
 - ▶ wireshark dissector: essentially done
 - ▶ client: in `rpcclient` - done
 - ▶ server: dummy PoC / tracer bullet implementation done
 - ▶ CTDB: changes / integration needed

Multi-Channel - Windows/Protocol

- ▶ find interfaces with interface discovery:
`FSCTL_QUERY_NETWORK_INTERFACE_INFO`
- ▶ bind additional TCP (or RDMA) connection (channel) to established SMB3 session (session bind)
- ▶ bind (TCP) connections of same quality
- ▶ bind only to a single node
- ▶ 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 (channel) to established SMB3 session (session bind)
- ▶ bind (TCP) connections of same quality
- ▶ bind only to a single node
- ▶ replay / retry mechanisms, epoch numbers

Multi-Channel - Samba

- ▶ samba/smbd: multi-process
 - ▶ process \Leftrightarrow tcp connection
 - ▶ \Rightarrow transfer new connection to existing smbd
 - ▶ use fd-passing (sendmsg/recvmmsg)
- ▶ preparation: messaging rewrite using unix dgm sockets with sendmsg [DONE,Volker]
- ▶ add fd-passing [WIP]
- ▶ transfer connection already in negprot (ClientGUID) [TODO]
- ▶ implement channel epoch numbers [started]
- ▶ implement interface discovery [TODO]

Multi-Channel - Samba

- ▶ samba/smbd: multi-process
 - ▶ process \Leftrightarrow tcp connection
 - ▶ \Rightarrow transfer new connection to existing smbd
 - ▶ use fd-passing (sendmsg/recvmmsg)
- ▶ preparation: messaging rewrite using unix dgm sockets with sendmsg [DONE,Volker]
- ▶ add fd-passing [WIP]
- ▶ transfer connection already in negprot (ClientGUID) [TODO]
- ▶ implement channel epoch numbers [started]
- ▶ implement interface discovery [TODO]

Multi-Channel - Samba

- ▶ samba/smbd: multi-process
 - ▶ process \Leftrightarrow tcp connection
 - ▶ \Rightarrow transfer new connection to existing smbd
 - ▶ use fd-passing (sendmsg/recvmmsg)
- ▶ preparation: messaging rewrite using unix dgm sockets with sendmsg [DONE,Volker]
- ▶ add fd-passing [WIP]
- ▶ transfer connection already in negprot (ClientGUID) [TODO]
- ▶ implement channel epoch numbers [started]
- ▶ implement interface discovery [TODO]

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 (Metze)]
- ▶ 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 (Metze)]
- ▶ 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 (Metze)]
- ▶ 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 (Metze)]
- ▶ 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` (?) listens for RDMA connection
- ▶ main `smbd` listens for TCP connection
- ▶ main `smbd` listens (for RDMA) via unix socket connect to `smbd-d`
- ▶ client connects via TCP → `smbd` forks child `smbd` (`c1`)
- ▶ client connects via RDMA to `smbd-d`
- ▶ `smbd-d` notifies main `smbd` and transfers connection info
- ▶ `smbd` forks child (`c2`) that inherits connection to `smbd-d`
- ▶ `c2` `smbd` passes [connection to `smbd-d`] to `c1` (via `ClientGUID`) and exits
- ▶ `c1` establishes mmap area with `smbd-d`
- ▶ client does rdma calls to `smbd-d`
 - ▶ metadata and protocol calls are transferred via socket to `tcp-smbd`
 - ▶ rdma read/write directly to `tcp-smbd` via mmap area

SMB Direct (RDMA) - Plan

- ▶ `smbd-d` (?) listens for RDMA connection
- ▶ main `smbd` listens for TCP connection
- ▶ main `smbd` listens (for RDMA) via unix socket connect to `smbd-d`
- ▶ client connects via TCP → `smbd` forks child `smbd` (`c1`)
- ▶ client connects via RDMA to `smbd-d`
- ▶ `smbd-d` notifies main `smbd` and transfers connection info
- ▶ `smbd` forks child (`c2`) that inherits connection to `smbd-d`
- ▶ `c2` `smbd` passes [connection to `smbd-d`] to `c1` (via `ClientGUID`) and exits
- ▶ `c1` establishes mmap area with `smbd-d`
- ▶ client does rdma calls to `smbd-d`
 - ▶ metadata and protocol calls are transferred via socket to `tcp-smbd`
 - ▶ rdma read/write directly to `tcp-smbd` via mmap area

Persistent Handles

- ▶ like durable file handles with strong guarantees
- ▶ framework is already there in samba (by support for durable v2)
 - ▶ \Rightarrow easy to satisfy at the protocol level
- ▶ the difficulty lies in implementing the guarantees
 - ▶ need make metadata persistent
 - ▶ but don't kill performance!
 - ▶ persistent tdb's **would** kill performance
 - ▶ ideas:
 - ▶ need to be sync
 - ▶ record-level transactions (instead of db-level)
 - ▶ only replicate to some nodes, not all

Persistent Handles

- ▶ like durable file handles with strong guarantees
- ▶ framework is already there in samba (by support for durable v2)
 - ▶ \Rightarrow easy to satisfy at the protocol level
- ▶ the difficulty lies in implementing the guarantees
 - ▶ need make metadata persistent
 - ▶ but don't kill performance!
 - ▶ persistent tdb's **would** kill performance
 - ▶ ideas:
 - ▶ need to be sync
 - ▶ record-level transactions (instead of db-level)
 - ▶ only replicate to some nodes, not all

Persistent Handles

- ▶ like durable file handles with strong guarantees
- ▶ framework is already there in samba (by support for durable v2)
 - ▶ \Rightarrow easy to satisfy at the protocol level
- ▶ the difficulty lies in implementing the guarantees
 - ▶ need make metadata persistent
 - ▶ but don't kill performance!
 - ▶ persistent tdb's **would** kill performance
 - ▶ ideas:
 - ▶ need to be sync
 - ▶ record-level transactions (instead of db-level)
 - ▶ only replicate to some nodes, not all

<https://wiki.samba.org/index.php/SMB3>



**CHILLI
SAMBA
FLAVOUR**



Questions?

obnox@samba.org
ma@sernet.de

