What’s new with SMB 3?

Mathew George & David Kruse
Microsoft Corporation
Status update

- SMB3 is 5 years old now.
  - Scalable, continuously available file sharing.
  - Designed for enterprise and cloud-infrastructure workloads.
  - No major changes to the protocol

- Enabling new scenarios
  - Storage spaces direct built on top of SMB3
  - Protocol for container guest to host access.
  - Direct access to persistent memory devices

- SMB1 will be turned off by default on Windows (*)
Status - SMB1 deprecation on Windows

- Windows SMB1 client & server are optional components.
  - Uninstall them individually in the Fall 2017 Update.

- SMB1 Changes coming in the Windows Fall 2017 Update
  - SMB1 server off by default on all Windows SKUs
  - SMB1 client off by default on Windows Server & Enterprise SKUs.
  - SMB1 client on by default on Home and Professional client SKUs
    - Still too many SMB1 only NAS boxes in use!
  - OS upgrades will preserve the state of SMB1 from previous OS.
    - Automatically uninstalled if no usage detected for 15 days.
Synchronous share redirection
Recap: Scaleout shares

- Share is surfaced on all nodes of a cluster
- Client can connect to any node.
- Symmetric
  - all nodes are equivalent and client can issue IO to the share via any node.
- Asymmetric –
  - One node is preferred over others.
  - Client will be lazily redirected to the optimal node by the witness protocol.
  - Still possible to do IO via any node – with the server redirecting the IO to the right node

![Diagram](image-url)
What is Synchronous Redirect mode?

- The server rejects the client’s attempt to connect to the wrong node and redirects it to the optimal node.
  - Client indicates support for synchronous redirection by setting the SMB2_REQ_TREE_CONNECT_FLAG_REDIRECT_TO_OWNER flag.
  - If the client is connected to the wrong node –
    - The server fails the tree connect with STATUS_BAD_NETWORK_NAME
    - Sends back an SMB2_ERROR_CONTEXT_SHARE_REDIRECT error context.
  - The payload in the “share redirect” error context tells the client where to go.
  - The client re-issues the tree connect to the right node and resumes IO.
  - See MS-SMB2 section 2.2.2.2.2 (Share Redirect Error Response)
What are the benefits?

- Redirection is completely implemented in the SMB3 protocol layer.
  - No dependency on the witness protocol.
- Server no longer needs to do back-end redirection of IO.
  - The backend filesystem is considerably simpler.
- Failure modes are simple because the client always connects to the “right” node.
Identity Remoting
Scenario: Infrastructure Shares

- Tenant VM hosting
  - Storage for VM (VHD) resides on file server.
  - VHD is ACL-d using tenant identity
- The VM (host) authenticates to the file server using tenant identity.
  - Access to VHD is granted by the server based on tenant identity.
- Client (VM) does access control to files in the VHD.
Scenario: Infrastructure Shares

What if we remove the VHD container?

- The share is now a container for the tenant.
  - Share ACLs allow access based on tenant identity
  - Client authenticates using tenant identity and gains access to the share.
- Once granted access to the share, the client “remotes” application identity to the server.
  - Server uses the “remoted identity” to enforce access control on files in the share.
Scenario 2: Container Shared volumes

- Share is mounted on container host using container identity.
  - New SMB global mapping functionality establishes a shared SMB session for all users on the box.
    
    ```bash
    new-smbglobalmapping -localpath g: -remotepath \server\share1 -credential $containerCred
    ```

- The global SMB mapping (g:) is shared into the container.
  ```bash
  docker run -v g:\containerdata:c:\appdata mywebserver
  ```

- Identity of the container app can now be remoted to the server on top of the pre-established SMB mapping.
Is there a privilege escalation problem?

- Access to the share is granted via a secure authentication protocol.
- Server allows “identity remoting” only to shares which are explicitly marked.
  - IPC$ share used for RPC must not allow this!
- Server guarantees that share scope cannot be escaped.
Protocol: Tree connect contexts.

- Client can now add a list of “contexts” to tree connect requests.
  - Wire format identical to negotiate contexts.
- The tree connect request is extended by adding an “extension”
  - Backward compatibility is maintained.
### Protocol: Tree connect extension

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StructureSize</td>
<td>0</td>
<td>Total size of the structure</td>
</tr>
<tr>
<td>Flags/Reserved</td>
<td>1</td>
<td>Flags and reserved bits</td>
</tr>
<tr>
<td>SMB2_TREE_CONNECT_FLAG_EXTENSION_PRESENT</td>
<td>2</td>
<td>Extension flag indicating the presence of the extension</td>
</tr>
<tr>
<td>PathOffset</td>
<td>3</td>
<td>Offset of the path offset</td>
</tr>
<tr>
<td>PathLength</td>
<td>4</td>
<td>Length of the path</td>
</tr>
<tr>
<td>TreeConnectContextOffset</td>
<td>5</td>
<td>Offset of the tree connect context offset</td>
</tr>
<tr>
<td>TreeConnectContextCount</td>
<td>6</td>
<td>Number of tree connect contexts</td>
</tr>
<tr>
<td>Reserved</td>
<td>7</td>
<td>Reserved bits</td>
</tr>
<tr>
<td>PathName (variable)</td>
<td>8</td>
<td>Path name variable</td>
</tr>
<tr>
<td>TreeConnectContexts (variable)</td>
<td>9</td>
<td>Array of tree connect contexts</td>
</tr>
</tbody>
</table>
Protocol: Tree connect contexts

- A list of variable length contexts starting at TreeConnectContextOffset.
- ContextType is a unique value identifying the payload.
  - Only one valid type
    
    ```
    #define SMB2_REMOTED_IDENTITY_TREE_CONNECT_CONTEXT_ID 1
    ```
- The context payload follows.
- Contexts must be 8-byte aligned.
Protocol: Remoted identity context

- The tree connect request MUST be signed.
- The context type is set to `SMB2_REMOTED_IDENTITY_TREE_CONNECT_CONTEXT_ID`
- The context data contains a serialized representation of the application identity.
  - The information is very similar to what is present in an ACL.
- Server filesystem enforces access control uses the remoted identity. (instead of the session identity.)
- See MS-SMB2 section 2.2.9 for details.
Remoted Identity : Usage

- Client establishes a single session to the server using “tenant credentials” by setting up a “global mapping”.
- Every user session on the client shares the same global SMB session, but establishes a new tree connect and sends its remoted identity to the server.
- The server grants access to the share based on the authenticated global session.
- The server filesystem uses the remoted identity to enforce access control to the files in the share.
Summary & future directions

- Increasing use of SMB3 for tenant/infrastructure data access.
- SMB3 as a filesystem protocol for container guest-host file sharing & RPC.
- Expect to see more use of PKU2U / NEGOEx
  - Available since 2012 for online ID based authentication
  - Enables clients and servers to do certificate based authentication.
Exploring SMB, RDMA, and NVDIMM
Storage Latencies

- DRAM
- SCM
- SSD
- HDD

Latencies range from 1 μSec to 1,000,000,000 μSec.
1 Traditional i/o
2 DAX load/store by SMB3 Server
3 Push Mode direct from RDMA NIC

Thanks Tom!
Block Mode Access in SMB

- Block Mode Access
  - Client and Server architecture remain the same
  - Decreased latency to storage
  - Still requires file system processing
  - Available today
Push Mode Access in SMB

- "Push Mode"
  - Client requests push-mode registration
  - Server registers memory and returns to client
  - Client reads/writes data directly to memory via RDMA
  - Client releases registration
Challenges of Push Mode

- Commit semantics for write-thru operations require hardware support or explicit SMB3 Flush
- Server needs to balance RDMA resources across all clients
  - Protection Domain is bound to QP at connect, but multichannel isn’t securely bound until Session Setup
  - Without a shared PD, registrations must be per-channel. That could be OK.
- Protocol needs support for recalling registrations
- Signing or Encryption of application data
- Thursday 8:30 – Tom Talpey – “Remote Persistent Memory – With Nothing But Net”
SMB Server - DAX Support

- No changes to clients
- Bypass the file system for data access
- Server controls mapping
  - Synchronize and refresh on file size changes
  - Release mapping on request from file system
  - If unavailable, fall back to normal read/write
- Server ensures flush for write-through access
Daniel McIlvaney

- Interned with us summer 2017
- Modified SMB2 Server code for Dax operation
- All code described from here forward is an unreleased prototype
SMB Server Changes

- On first read/write to file on DAX mode, file section is mapped
- Reads/writes acquire a rundown on map, operations requiring remap drain them
- TCP operations copy data into map. RDMA operations are read or written directly to mapped pages
- On write-through writes, the mapping is flushed before response is sent (clflush)
Test Setup

- HP Servers
  - 2x Xeon E-5 2699 22 Core @ 2.2 GHz
  - 256 GB RAM
  - 120 GB SCM
  - 2x 100 Gbps RDMA NICs
- Disabled hyperthreading, power states, etc.
- DISKSPD as load generator
- NTFS Formatted DAX Volume on single DIMM
Synchronous 4k Reads

IOPS

Baseline: 20000
Prototype: 24000

Difference: +20%

Avg Latency

Baseline: 0.03
Prototype: 0.02

Difference: -19%
Synchronous 4k Reads – Absolute Latency (ms)
Synchronous 4k Reads – Improvement vs. Baseline

![Graph showing improvement percentages across different performance levels.](image)
Synchronous 4k Writes

IOPS

Baseline Prototype

+23%

Avg Latency

Baseline Prototype

-25%
Synchronous 4k Writes – Absolute Latency (ms)
Synchronous 4k Writes – Improvement vs. Baseline

-40.00%
-35.00%
-30.00%
-25.00%
-20.00%
-15.00%
-10.00%
-5.00%
0.00%
5.00%

Minimum 25% 50% 75% 90% 95% 99% 99.9% 99.99% 99.999% 99.9999%
Many Threads - Parallel 4k – 100% Reads

IOPS

Baseline  Prototype

Avg Latency

Baseline  Prototype
Many Threads - Parallel 4k – 100% Writes

**IOPS**

- Baseline: 100,000
- Prototype: 250,000
  - Prototype is 134% higher than Baseline

**Avg Latency**

- Baseline: 1.5 ms
- Prototype: 0.6 ms
  - Prototype is 57% lower than Baseline
Many Threads - Parallel 4k – 50% Reads

IOPS

Baseline | Prototype
---|---
300000 | 350000
+60%

Latency (ms)

Baseline | Prototype
---|---
1.00 | 0.65
-37%
Server-Side CPU Savings

IOPS – 4k Parallel Writes

Server CPU Consumed
Everything looks great. Let’s not look closely or ask questions and go get coffee.
1 Traditional i/o
2 DAX load/store by SMB3 Server
3 Push Mode direct from RDMA NIC

Will be either block based or Cache Manager based depending on Volume Property!
Baseline vs. Block Mode Implementation

Sync Read

Sync Write

Parallel Read

Parallel Write

IOPS

IOPS

IOPS

IOPS

Dax  Block  Prototype

Dax  Block  Prototype

Dax  Block  Prototype

Dax  Block  Prototype
Unexpected…

- Block write uses more than double the server CPU and has an additional context switch in the completion model, but shows consistently higher IOPS and lower latency.
- Do we need push mode to fully utilize NVDIMM?
Potential Reasons?

- Synchronization on mapping uses a different primitive than normal IO to let us drain the map
- Unbuffered/write-through IO uses CLFLUSH to flush memory
  - Disabling both of these brings us on par
  - PMEM may have more efficient system of cache coherency
- With less CPU processing, worker threads are re-entering idle time more often
Summary

- Dax Mode implementation offers compelling benefits in read performance in our implementation, and definite improvement when DAX is used by local services.
- Write performance (vs. Block) requires more work to fully understand.
Questions?

Thanks! Don’t forget to uninstall SMB1.