Troubleshooting clustered Samba
in Enterprise environments

Christof Schmitt
`christof.schmitt@us.ibm.com`
`cs@samba.org`

IBM / Samba Team

May 5, 2021

Outline

Abstract

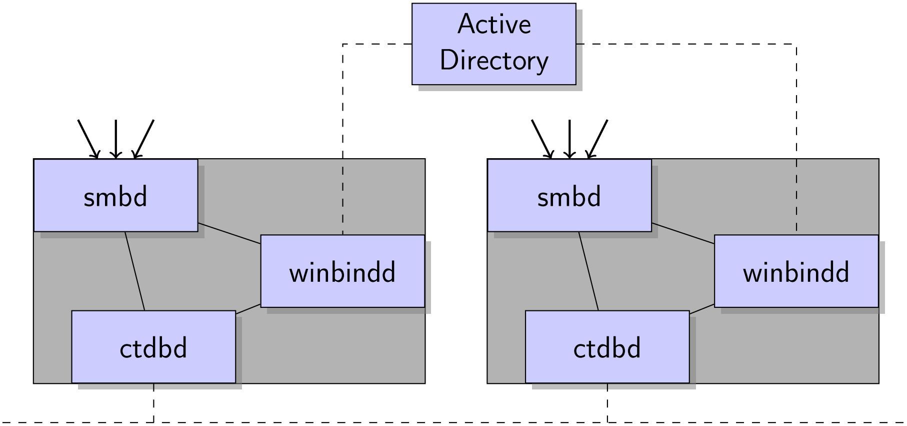
IBM Spectrum Scale is a software defined storage offering of a clustered file system bundled with other services. Samba is included as part of the product for providing a clustered SMB file server and integration into Active Directory environments. This product is commonly used in Enterprise IT environments.
Troubleshooting problems is an essential part of supporting customers. This talk will walk through Samba troubleshooting approaches that have proven useful over the years. It will explain how for this environment Samba is configured to provide logs and indications by default. Methods for collecting additional trace data are demonstrated and how to efficiently analyze these traces. Examples will be used to illustrate debugging problems from the trace data.

Spectrum Scale

- Clustered file system (formerly known as GPFS)
  - Multiple systems (nodes) share direct access to same file system.
- Plus additional services (SMB, NFS, Object Storage, HDFS, . . .)
- Uses clustered Samba for SMB server and Active Directory integration
- In this presentation:
  - focus on clustered Samba
  - demonstrate concepts using Samba tools
    - (actual product implementation is different)

Clustered Samba

- A somewhat high-level view.



Clustered Samba

- Multiple nodes present one SMB file server.
- When using Active Directory, the cluster is one domain member.
- Samba internal data is shared across the nodes through ctdb.
    - SMB metadata for open files
    - SMB share config
    - AD machine account secrets
    - ...
- SMB clients can connect to any node and always present the same credentials
- Round-robin DNS pointing to floating IP addresses on nodes
- Floating IP addresses can get moved between nodes

Clustered Samba implications for troubleshooting

- SMB client can connect to any node
- Can be different node when reconnecting
- Problems can affect one node or multiple nodes
- Network, firewall
  - between cluster nodes
  - between SMB client and cluster
  - between cluster node and Active Directory Domain Controller
- winbindd on different nodes can connect to different DCs
- winbindd depends on ctdb, smbd depends on winbindd and ctdbd
  - Check services: ctdbd, winbindd, then smbd

Troubleshooting methods

- Status check
  - Is everything running as expected?
  - Monitoring framework, manual check or data capture.
- Problems that can be re-created upon request
  - Some operation does not work (e.g. `ACCESS_DENIED`)
  - Enable tracing, recreate, debug with traces.
  - Usually not a performance issue.
- Performance issues
  - Need to find out where time is spent.
  - Cannot afford performance impact from traces.
- Problems at random times
  - Application error at unpredictable time.
  - Need constant tracing that can be quickly stopped.
  - Needs to be affordable
    - Cannot impact performance normal operations.
    - Need to have sufficient disk space.

Logging and data capture in ctdb config

```
/etc/ctdb/ctdb.conf
```

```
[logging]
location = syslog
log level = NOTICE
```

- ctdb does not support log file rotation, so logging to syslog.
- `NOTICE` level sufficient for many cases.

Logging and data capture in Samba config

```
/etc/samba/smb.conf

[global]
clustering=yes
ctdb:registry.tdb=yes
include=registry
include=/etc/samba/smb.conf.%I
```

- Samba config stored in clustered registry database
  - Each update is immediately available on all cluster nodes.
  - Preferred for cluster setups.
- Still need a local file that is read first and "includes" the registry
- Also have include ready for troubleshooting
  - Allow different config for debugging specific SMB clients
  - If client connects from IP address %I, also read config file `/etc/samba/smb.conf.%I`
  - File does not exist by default.
  - `https://wiki.samba.org/index.php/Client_specific_logging`

Logging and data capture in Samba config (continued)

```
net conf list
     ctdb locktime warn threshold = 5000
     time_audit:timeout = 5000
     vfs objects = shadow_copy2 syncops gpfs fileid time_audit
     logging = syslog@0 file
     log level = 1
     smbd profiling level = on
     winbind:request profile threshold = 5  ...
```

- Log error when any operation takes longer than 5 seconds
  - `ctdb locktime` and `time_audit`
  - Good first indicator of serious problems
- Default log levels
  - Errors ("level 0") to syslog, warnings ("level 1") to log file.
  - Has been proven useful, but could be set differently
- Enable profiling for troubleshooting of performance problems.
- Enable winbindd profiling for requests taking longer than 5 seconds

ctdb status check

- `onnode all ctdb status`
  - Is everything reported `OK`?
  - Do all nodes agree on same `Recovery Master`?
  - Anything `UNHEALTHY`?
- `onnode all ctdb scriptstatus`
  - Any problems from event scripts?
- Any problems reported in syslog?
  - Connection problems between nodes?
  - ctdb recovery?

winbindd status check

- `onnode all wbinfo --ping`
  - Is winbindd responding to queries on all nodes?
- `onnode all wbinfo --ping-dc`
  - Can winbindd on each node reach out to a DC?
  - If there is a problem on some nodes, is it tied to a specific DC?
- `onnode all net ads info`
  - Is the time the same on node and DC?
  - Kerberos requires synchronized clocks.

smbd status check

- `onnode all ps -ef`
  - smbd running?
  - Any process in `D uninterruptible sleep`?
    - Cluster file systems introduce complexity in backend, system calls can be waiting for e.g. network problems.
    - If yes, check kernel stack of process: `cat /proc/$(pid)/task/*/stack`
- `smbstatus -p`
  - Check for connected clients.
  - Do not list open files and locks on busy cluster!

smbd log status check

```
[2021/04/22 14:06:01.513570,  0] ../../source3/modules/vfs_...
  WARNING: VFS call "create_file" took unexpectedly long ...
[2021/04/22 14:06:01.507335,  0] ../../source3/modules/vfs_..
  WARNING: VFS call "async pwrite" took unexpectedly long ...
```

- Any errors or warnings?
- `time_audit` module in config
  - Only `VFS call "create_file"` warnings?
    - Likely contention across multiple nodes.
    - See `man vfs_fileid`
  - Other `VFS call` warnings?
    - Need to check file system and storage.
    - Something overloaded?

winbindd log status check

```
[2021/05/03 10:37:11.233653,  0] ../../source3/winbindd/...
  winbind_client_processed: request took 10.003685 seconds
   [struct winbindd_ping_dc_state] ...
  [2021/05/03 10:37:11.233618] [10.003642] -> TEVENT_REQ_DONE
...
```

- Configured threshold (5 seconds from earlier config)
- Error message when request takes longer.
- Detailed breakdown of time spent in different steps.
- Currently only tracks winbindd parent, no child process.

Collecting traces

- Need additional data when default logs are not sufficient.
- What did the SMB client actually send?
  - Network trace
- What did the SMB server do internally?
  - smbd level 10 trace
- What happened with authentication and id mapping?
  - winbindd level 10 trace

Network trace

- What actually arrived from the SMB client?
  - Potential network or firewall problem?
- What is the exact content of the SMB requests and responses?
  - Do SMB client and server adhere to MS-SMB2 spec?
- Capture with tcpdump
  - Cannot limit packet capture size, otherwise Wireshark will not parse.
  - Need to capture trace on all nodes, SMB client could connect to any.
  - Optional, but useful: Force reconnect to get complete trace from connection setup to problem.
- Can also have long running trace for tracing random events.
  - Needs potential large buffer.
  - Luckily storage is cheap and we have a clustered file system.

Network trace example

1. Start network capture

```
onnode all 'tmux new-session -d \
          "tcpdump -i any -w /tmp/$(hostname)-tcpdump.pcap \
                         host 192.168.3.4"'
onnode all 'ps -ef | grep [t]cpdump'
```

2. Optional: Force client reconnect

```
onnode all smbcontrol smbd kill-client-ip 192.168.3.4
```

4. Recreate problem on SMB client

5. Stop network capture

```
onnode all killall tcpdump
```

Network trace for long running trace

- tcpdump allows tracing to rotating buffer
  - `-C` size of one buffer part in million bytes
  - `-W` number of buffer parts

1. Start network capture

```
onnode all 'tmux new-session -d \
           "tcpdump -i any -w /tmp/$(hostname)-tcpdump.pcap \
                           -C 100 -W 100 host 192.168.3.4"'
onnode all 'ps -ef | grep [t]cpdump'
```

2. Wait for problem to hit.

3. Stop network capture when problem hits

```
onnode all killall tcpdump
```

smbd traces

- We want a level 10 / DEBUG trace.
  - Will show all details of smbd internal calls.
- Will be very big, cannot run this constantly
  - Never enable `log level = 10` globally on a cluster!
  - Only want to trace a single client.
- Solution: Deploy client specific config file (see earlier `%I` include)
  - Config is read when client connects from this IP address
  - e.g. `/etc/samba/smb.conf.192.168.3.4`

```
debug hires timestamp = yes
max log size = 200000
log level = 10
log file = /tmp/smb.%h_%I.%d.smbd.log
```

- Enabled level 10 logging to log file in /tmp
  - `%h` hostname
  - `%I` client IP address
  - `%d` smbd PID, in case of multiple connections
- Force client to reconnect

smbd traces example

1. Provide trace config

```
# cat > /etc/samba/smb.conf.192.168.3.4 << EOF
debug pid = yes
> debug hires timestamp = yes
> max log size = 200000
> log level = 10
> log file = /tmp/smb.%h_%I.%d.smbd.log
> EOF
```

2. Deploy on all nodes

```
onnode -P all /etc/samba/smb.conf.192.168.3.4
```

3. Force SMB client to reconnect to use new config

```
onnode all smbcontrol smbd kill-client-ip 192.168.3.4
```

smbd traces example (continued)

4. Recreate problem on SMB client

5. Remove file

```
onnode all rm -f /etc/samba/smb.conf.192.168.3.4
```

6. Force SMB client to reconnect to use default config again

```
onnode all smbcontrol smbd kill-client-ip 192.168.3.4
```

7. Collect all files in one location

```
onnode all scp '/tmp/smb.*' $(hostname):/tmp
```

winbindd traces

- One instance of winbindd on each node.
- Change log level of running winbindd
  - `smbcontrol winbindd debug 10` to enable tracing
  - `smbcontrol winbindd debug 1` to revert back to default.
- Caveat: caching
  - winbindd caches e.g. idmap lookups
  - Trace might not show actual lookup.
  - Might want to clear caches
- Can be left running for random problems
  - Does not introduce not much overhead.

winbindd traces example

Start winbindd tracing

```
onnode all smbcontrol winbindd debug 10
```

Clear caches

```
onnode all net cache flush
onnode all tdbtool /var/lib/samba/winbindd_cache.tdb wipe
```

Recreate problem

Revert back to normal log levels

```
smbcontrol winbindd debug 1
```

Analyzing traces

- Very quick summary.
- Every area would deserve separate discussion.
- Wireshark for network traces
  - `https://www.wireshark.org/`
- Other traces are just plain text files
  - smbd traces
  - winbindd traces

Network traces

- Wireshark for parsing SMB data stream
- Filter on any field, e.g.
  - Request type
  - Status codes
- `tshark`
  - CLI interface
  - Useful for scripting
  - e.g. in each trace file look for TCP reset

smbd traces

```
[2021/04/23 16:37:41.826884, 10, pid=15488, effective(11106654,
  smbd_smb2_request_dispatch: opcode[SMB2_OP_READ] mid = 906
...
[2021/04/23 16:37:41.831691, 10, pid=15488, effective(11106654,
  smbd_smb2_request_done_ex: mid [906] idx[1] status[NT_STA...
```

- smbd still mostly single-threaded
- Look for request dispatch and returned status code.
- Returned status error? Go backwards to find source of error.
- `mid` is field `msg_id` in Wireshark to compare with network trace.

winbindd traces

```
[2021/04/23 16:35:01.039240, 10, pid=18379, effective(0, 0)...
  process_request_send: process_request: Handling async
          request nss_winbind(12850):GETPWNAM
[2021/04/23 16:35:01.039285,  3, pid=18379, effective(0, 0)...
  winbindd_getpwnam_send: [nss_winbind (12850)] getpwnam
          domain\cs
...
[2021/04/23 16:35:01.048190, 10, pid=18379, effective(0, 0)...
  process_request_done: [nss_winbind(12850):GETPWNAM]:
          NT_STATUS_OK
```

- Main log `log.winbindd`
- Parent/child process model
  - Might need to chase down to child logs
    - Multiple child processes for domain communication: `log.wb-DOMAIN`
    - One idmap child process: `log.winbindd-idmap`
- Incoming request, outgoing response status

Troubleshooting performance problems

- Performance counters
  - Number of SMB requests
  - Total time per SMB requests
  - Total number of bytes
  - All counters per node
  - Capture before and after, calculate difference.
- Network trace
  - Wireshark: Statistics > Service Response Time > SMB2
  - Also allows understanding (some) context of slow requests.

Performance counters example

```
$ smbstatus -P
...
syscall_asys_pread_count:                          8
syscall_asys_pread_time:                       71941
syscall_asys_pread_idle:                        3240
syscall_asys_pread_bytes:                   64750080
...
smb2_read_count:                                   8
smb2_read_time:                                72601
smb2_read_idle:                                    0
smb2_read_inbytes:                               904
smb2_read_outbytes:                         64750720
...
```

Summary

- Troubleshooting methods
  - Status check
  - Recreate
  - Random problems
- Useful traces and trace capture in cluster
  - Network trace
  - smbd level 10 trace
  - winbindd trace
  - Performance counters
- Start of trace analysis
- Performance troubleshooting

Legal Statement

Thank you!

- Questions?