multichannel / io_uring

Status Update within Samba

Stefan Metzmacher <metze@samba.org>

Samba Team / SerNet

2021-05-05

https://samba.org/~metze/presentations/2021/SambaXP/

Check for Updates

- Check for an updated version of this presentation here:
- https://samba.org/~metze/presentations/2021/SambaXP/

(draft)

Topics

- What is SMB3 Multichannel?
- Updates in Samba 4.15
- What is io-uring?
- io-uring for Samba
- Performance research, prototyping and ideas
- 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
    - Windows returns ACCESS_DENIED in some cases instead
    - In other cases Windows ignores a replay and deadlocks the client
    - I need to discuss this with Microsoft
    - See: Samba Bug #14449
- 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

Last Status Update SDC 2020

- I gave a similar talk at the storage developer conference:
  - See https://samba.org/~metze/presentations/2020/SDC/
  - It explains the milestones and design up to Samba 4.13

Updates in Samba 4.15

- Automated regression tests are in place:
  - socket_wrapper got basic fd-passing support(Bug #11899)
  - We added a lot more multichannel related regression tests
- The last missing features/bugs are fixed (Bug #14524)
  - The connection passing is fire and forget (Bug #14433)
  - Pending async operations are canceled (Bug #14449)
- 4.15 will hopefully have "server multi channel support = yes"
  - Currently it's still off by default, but may change before 4.15.0rc1
  - We require support for TIOCOUTQ (Linux) or FIONWRITE (FreeBSD)
  - We disable multichannel feature if the platform doesn't support this
    - See: Retries of Lease/Oplock Break Notifications (Bug #11898)
- I have unofficial backport for older branches
  - SerNet's SAMBA+ 4.14 includes the patches
  - "server multi channel support = no" is still the default

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
- This can be nicely integrated with our async tevent model
  - It may delegate work to kernel threads
  - It seems to perform better compared to our userspace threadpool
  - It can also inline non-blocking operations

io-uring for Samba (Part 1)

- 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
  - Improved MSG_WAITALL support (5.12, backport to 5.11, 5.10)
  - 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
  - IORING_OP_UNLINKAT, IORING_OP_RENAMEAT (from 5.10)

io-uring for Samba (Part 2)

IORING_FEAT_NATIVE_WORKERS (from 5.12)

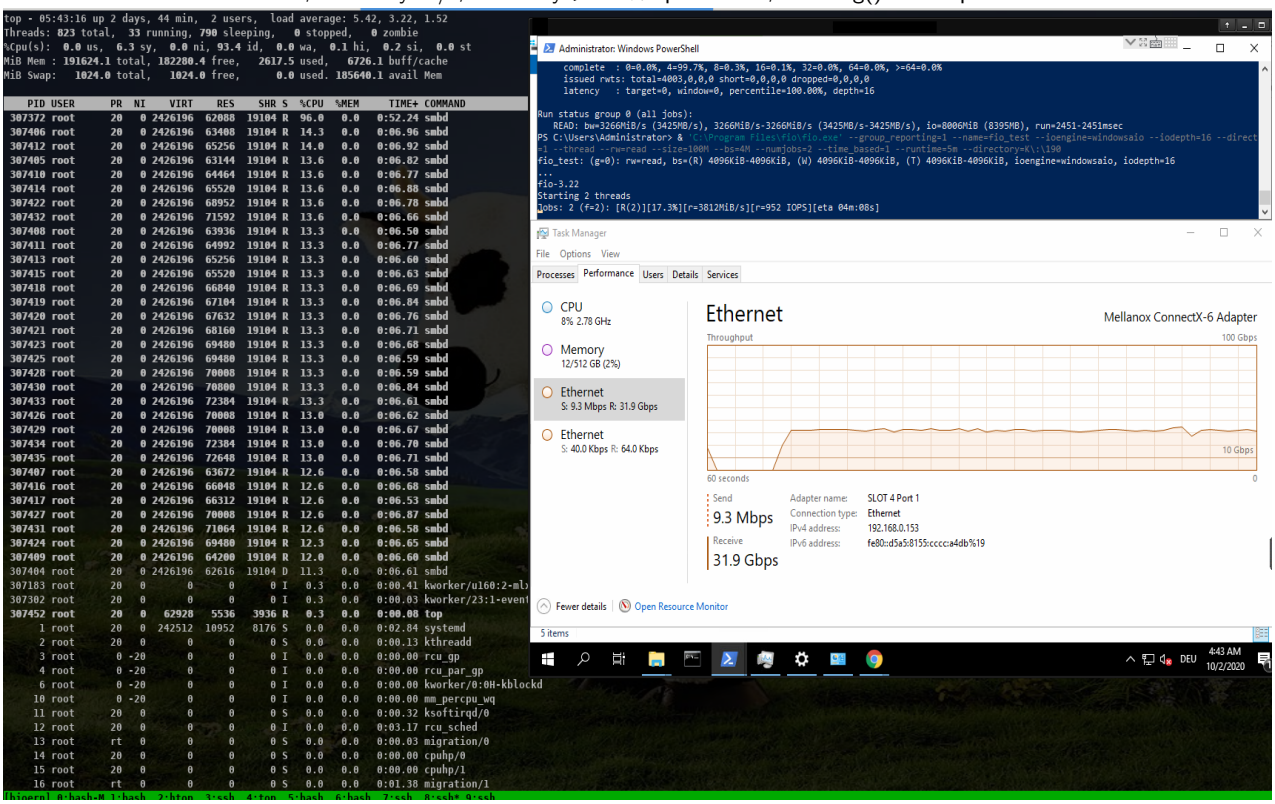
- In the kernel...
  - The io-uring kernel threads are clone()'ed from the userspace thread
  - They just appear to be blocked in a syscall and never return
  - This makes the accounting in the kernel much saner
  - Allows a lot of restrictions to be relaxed in the kernel
  - Most likely to backported to the 5.10 LTS kernel
- For admins and userspace developers...
  - 'top' shows them as part of the userspace process ('H' shows them)
  - They are now visible in containers
  - 'pstree -a -t -p' is very useful to see them
  - gdb may show worrying messages:
    - "warning: Architecture rejected target-supplied description"
    - But it seems they can be ignored and will be fixed soon

Performance research (SMB2 Read)

- Last October I was able to do some performance research
  - DDN was so kind to sponsor about a week of research on real world hardware
  - With 100GBit/s interfaces and two NUMA nodes per server.
- I focussed on the SMB2 Read performance only
  - We had limited time on the given hardware
  - We mainly tested with fio.exe on a Windows client
  - Linux kernel 5.8.12 on the server
- More verbose details can be found here:
  - https://lists.samba.org/archive/samba-technical/2020-October/135856.html

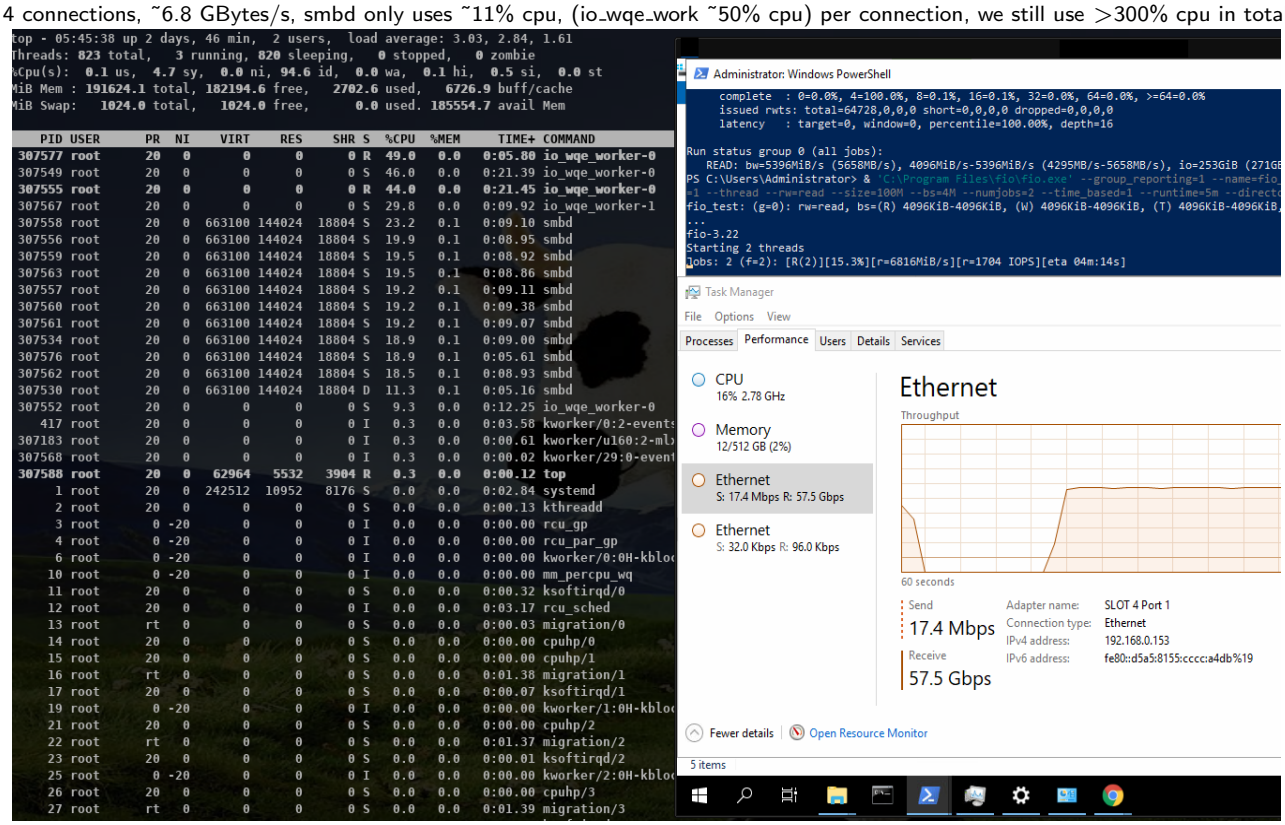
Performance with MultiChannel, sendmsg()

4 connections, ~3.8 GBytes/s, bound by >500% cpu in total, sendmsg() takes up to 0.5 msecs



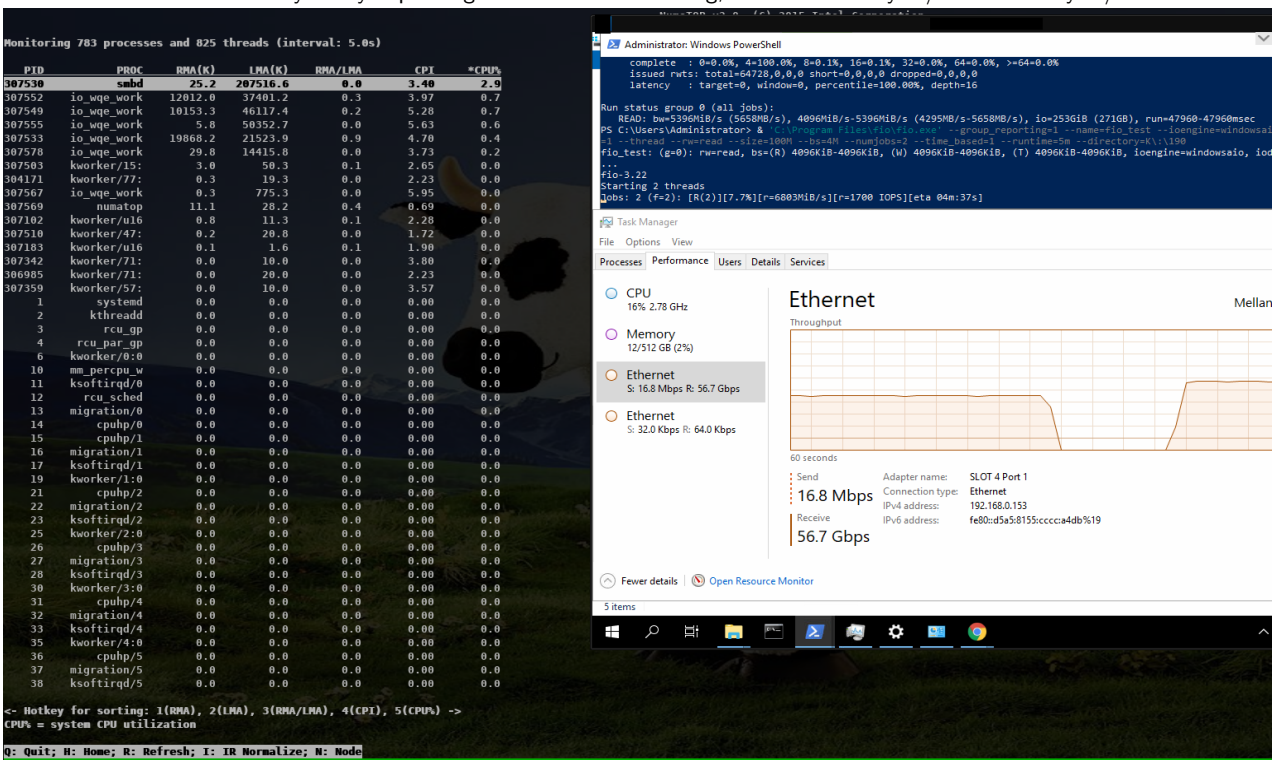
IORING_OP_SENDMSG prototyped (Part1)

4 connections, ~6.8 GBytes/s, smbd only uses ~11% cpu, (io_wqe_work ~50% cpu) per connection, we still use >300% cpu in total



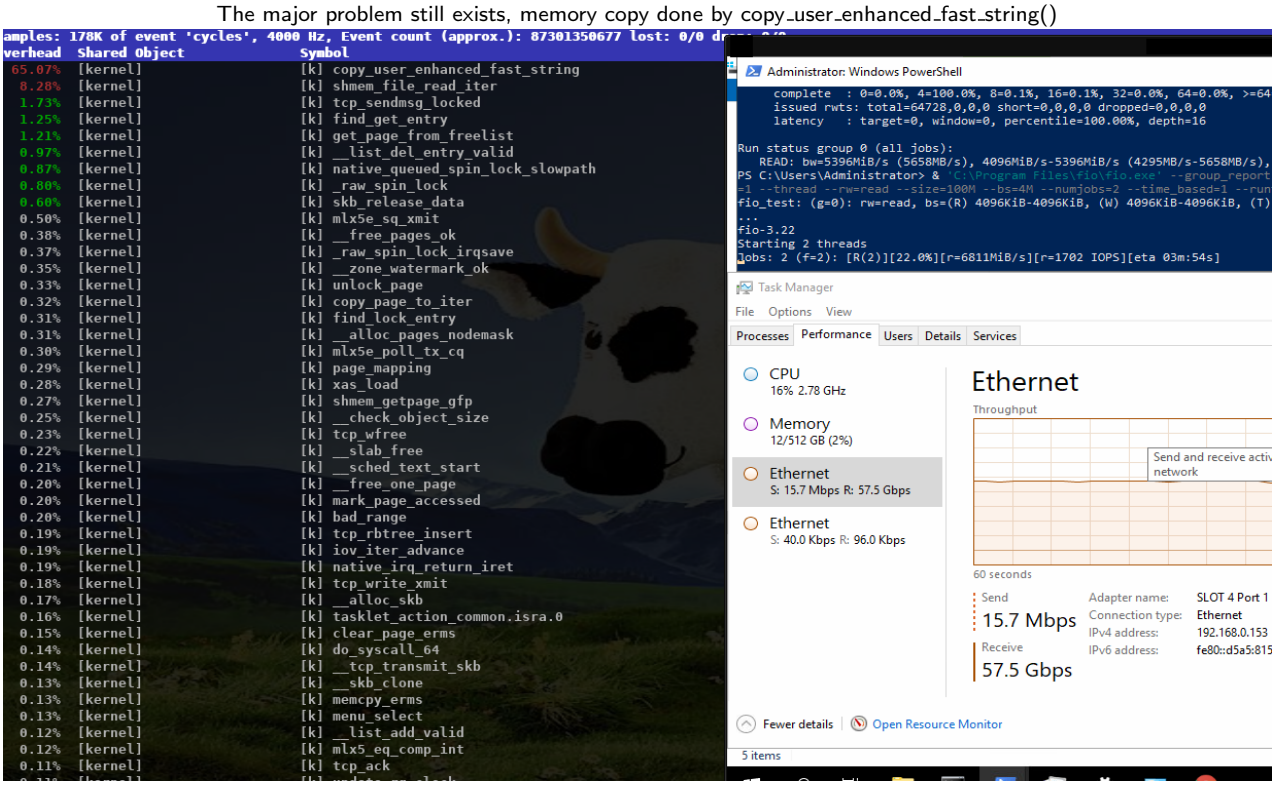
IORING_OP_SENDMSG prototyped (Part2)

The results vary havily depending on the NUMA bouncing, between 5.0 GBytes/s and 7.6 GBytes/s



IORING_OP_SENDMSG prototyped (Part3)

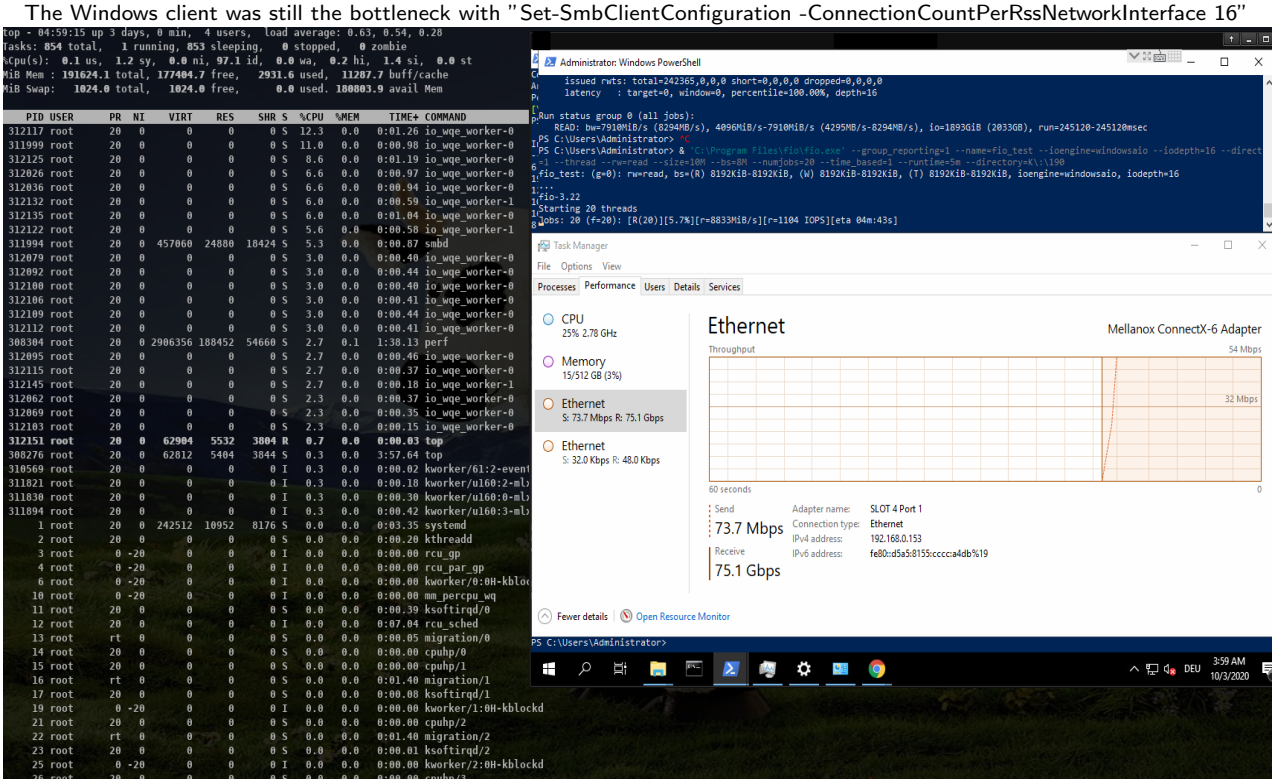
The major problem still exists, memory copy done by copy_user_enhanced_fast_string()



IORING_OP_SENDMSG/SPLICE prototyped (Part1)

16 connections, ~8.9 GBytes/s, smbd ~5% cpu, (io_wqe_work 3%-12% cpu filesystem->pipe->socket), only ~100% cpu in total.

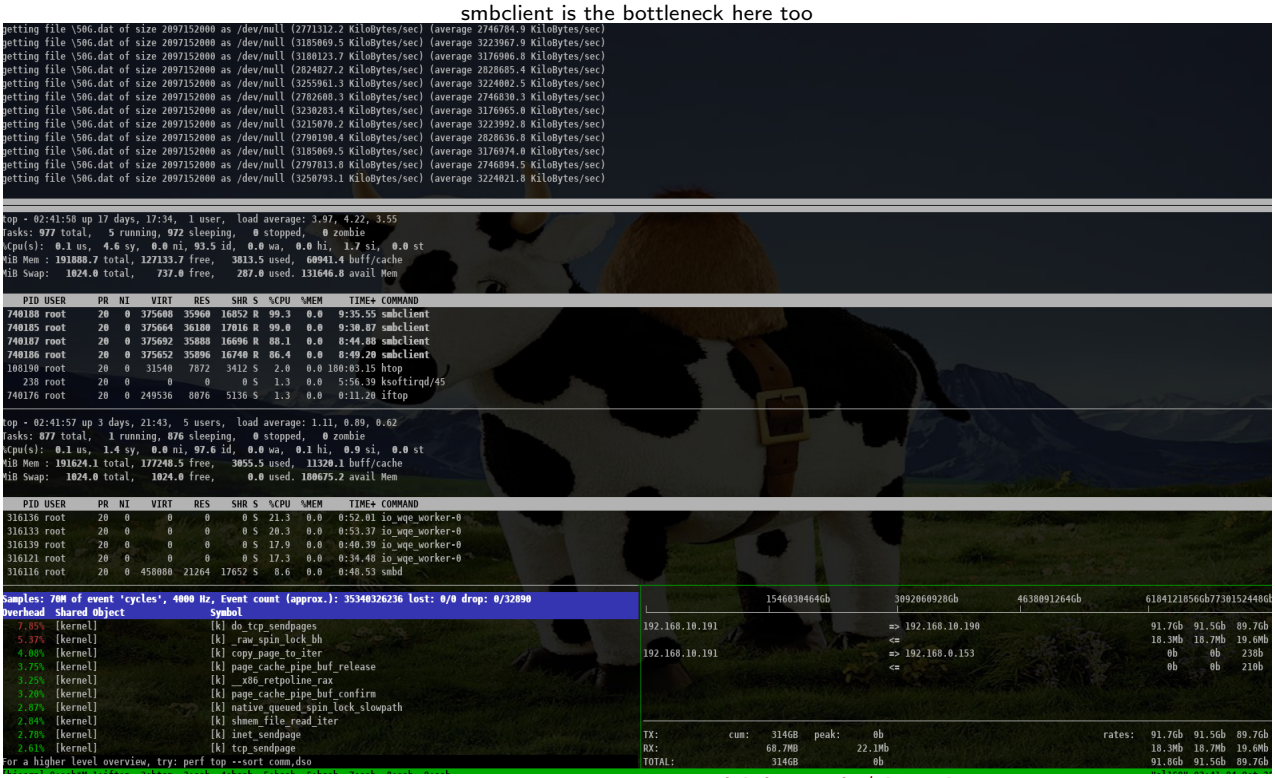
The Windows client was still the bottleneck with "Set-SmbClientConfiguration -ConnectionCountPerRssNetworkInterface 16"



smbclient IORING_OP_SENDMSG/SPLICE (network)

4 connections, ~11 GBytes/s, smbd 8.6% cpu, with 4 io_wqe_work threads (pipe to socket) at ~20% cpu each.

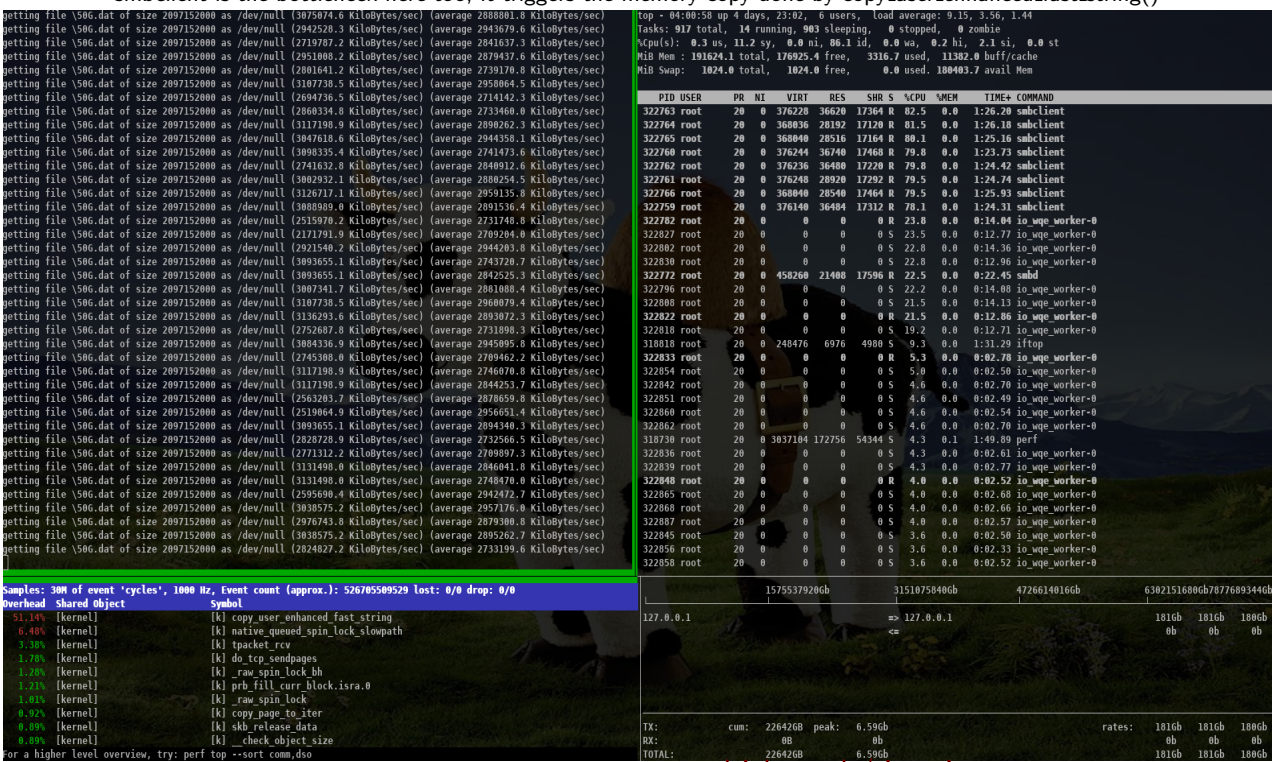
smbclient is the bottleneck here too



smbclient IORING_OP_SENDMSG/SPLICE (loopback)

8 connections, ~22 GBytes/s, smbd 22% cpu, with 4 io_wqe_work threads (pipe to socket) at ~22% cpu each.

smbclient is the bottleneck here too, it triggers the memory copy done by copy_user_enhanced_fast_string()



More loopback testing on brand new hardware

- Recently I re-did the loopback read tests IORING_OP_SENDMSG/SPLICE (from /dev/shm/)
  - 1 connection, ~11 GBytes/s, smbd 7% cpu, with 4 io_wqe_work threads at 7%-50% cpu.
  - 4 connections, 24-30 GBytes/s, smbd 18% cpu, with 16 io_wqe_work threads at 3%-35% cpu.
- I also prototyped SMB2 writes with IORING_OP_RECVMSG/SPLICE (to /dev/null)
  - 1 connection, ~7 GBytes/s, smbd 5% cpu, with 3 io_wqe_work threads at 1%-20% cpu.
  - 4 connections, ~10 GBytes/s, smbd 15% cpu, with 12 io_wqe_work threads at 1%-20% cpu.
- I tested with a Linux Kernel 5.10.25
  - In both cases the bottleneck is clearly on the smbclient side
  - We could apply similar changes to smbclient and add true multichannel support
  - It seems that the filesystem->pipe->socket path is much better optimized

Future Improvements

- recvmsg and splice deliver partial SMB packets to userspace
  - I tested with AF_KCM (Kernel Connection Multiplexor) and an eBPF helper
  - But MSG_WAITALL is the much simpler and faster solution
  - I also prototyped a SPLICE_F_WAITALL
  - eBPF support in io-uring would also be great for optimizations
- It also seems that socket->pipe->filesystem:
  - Does not implement zero copy for all cases
  - Maybe it's possible to optimize this in future
- For SMB3 signing/encryption we may use:
  - IORING_OP_TEE with vmsplice could be used in order to still allow IORING_OP_SPLICE from/to the filesystem
  - vmsplice may also need to be optimized and added to io-uring
  - With eBPF support in io-uring we might be able to offline signing/encryption
- In the end SMB-Direct will also be able to reduce overhead
  - My smbdirect driver is still work in progress...

Questions? Feedback!

- Feedback regarding real world testing would be great!
- Stefan Metzmacher, `metze@samba.org`
- https://www.sernet.com
- https://samba.plus

Slides: https://samba.org/~metze/presentations/2021/SambaXP/