



io_uring

Status Update within Samba

Stefan Metzmacher <metze@samba.org>

Samba Team / SerNet

2023-09-20

https://samba.org/~metze/presentations/2023/SDC/



Check for Updates

- ► Check for an updated version of this presentation here:
- https://samba.org/~metze/presentations/2023/SDC/





Topics

000000000

- What is io-uring?
- io-uring for Samba
- ▶ Performance research, prototyping and ideas
- The road to upstream
- Future Improvements
- Questions? Feedback!





Last Status Updates (SDC 2020/2021 - SambaX 120

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





Last Status Updates (SDC 2020/2021 - SambaX

- I gave a similar talk at the storage developer conference 2020:
 - See https://samba.org/~metze/presentations/2020/SDC/
 - It explains the milestones and design up to Samba 4.13 (in detail)
- I gave a similar talk at the storage developer conference 2021:
 - See https://samba.org/~metze/presentations/2021/SDC/
 - ▶ It explains the milestones and updates up to Samba 4.15 (in detail)
- I gave a similar talk at the SambaXP conference 2023:
 - See https://samba.org/~metze/presentations/2023/SambaXP/
 - It explains the milestones and updates up to Samba 4.19 (in detail)





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





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
 - ► IORING_OP_FSETXATTR, IORING_OP_FGETXATTR (from 5.19)
 - ► IORING_OP_GETDENTS, under discussion, but seems to be tricky
 - IORING_OP_FADVISE (from 5.6)





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
 - ► IORING_OP_FSETXATTR, IORING_OP_FGETXATTR (from 5.19)
 - ► IORING_OP_GETDENTS, under discussion, but seems to be tricky
 - IORING_OP_FADVISE (from 5.6)
- 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)
 - ▶ IORING_OP_MKDIRAT, IORING_OP_SYMLINKAT, IORING_OP_LINKAT (from 5.15)
 - ► IORING_OP_SETXATTR, IORING_OP_GETXATTR (from 5.19)





io-uring for Samba (Part 2)



- ▶ Between userspace and socket (and also filesystem) (from 5.8)
 - ► IORING_OP_SENDMSG, IORING_OP_RECVMSG
 - ▶ Improved MSG_WAITALL support (5.12, backported to 5.11, 5.10)
 - ► Maybe using IOSQE_ASYNC in order to avoid inline memcpy
 - ► IORING_OP_SPLICE, IORING_OP_TEE
 - ► IORING_OP_SENDMSG_ZC, zero copy with an extra completion (from 6.1)
 - IORING_OP_GET_BUF, under discussion to replace IORING_OP_SPLICE





vfs_io_uring in Samba 4.12 (2020)



- ▶ With Samba 4.12 we added "io_uring" vfs module
 - ► For now it only implements SMB_VFS_PREAD,PWRITE,FSYNC_SEND/RECV
 - It has less overhead than our pthreadpool default implementations
 - I was able to speed up a smbclient 'get largefile /dev/null'
 - Using against smbd on loopback
 - ► The speed changes from 2.2GBytes/s to 2.7GBytes/s
- The improvement only happens by avoiding context switches
 - But the data copying still happens:
 - From/to a userspace buffer to/from the filesystem/page cache
 - The data path between userspace and socket is completely unchanged
 - For both cases the cpu is mostly busy with memcpy





vfs_io_uring in Samba 4.12 (2020)



- ▶ With Samba 4.12 we added "io_uring" vfs module
 - ► For now it only implements SMB_VFS_PREAD,PWRITE,FSYNC_SEND/RECV
 - ▶ It has less overhead than our pthreadpool default implementations
 - I was able to speed up a smbclient 'get largefile /dev/null'
 - Using against smbd on loopback
 - ► The speed changes from 2.2GBytes/s to 2.7GBytes/s
- The improvement only happens by avoiding context switches
 - But the data copying still happens:
 - ► From/to a userspace buffer to/from the filesystem/page cache
 - ▶ The data path between userspace and socket is completely unchanged
 - For both cases the cpu is mostly busy with memcpy





Performance research (SMB2 Read)



- In October 2020 I was able to do some performance research
 - ▶ With 100GBit/s interfaces and two NUMA nodes per server.





Performance research (SMB2 Read)



- In October 2020 I was able to do some performance research
 - ▶ With 100GBit/s interfaces and two NUMA nodes per server.
- At that time 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





Performance research (SMB2 Read)



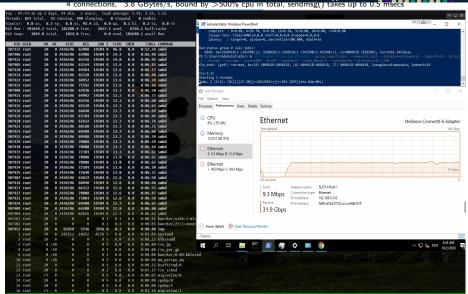
- ▶ In October 2020 I was able to do some performance research
 - ▶ With 100GBit/s interfaces and two NUMA nodes per server.
- ▶ At that time I focussed on the SMB2 Read performance only
 - ▶ We had limited time on the given hardware
 - We mainly tested with fio.exe on a Windows client
 - Linux kernel 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()



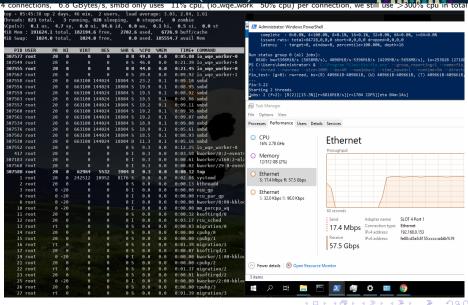




io_uring (10/22) SerNet

IORING_OP_SENDMSG (Part1)







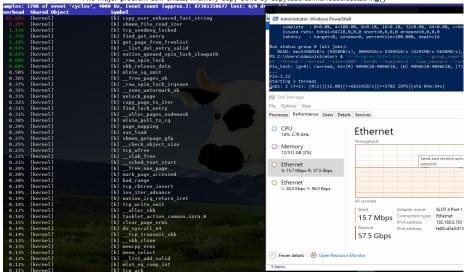




IORING_OP_SENDMSG (Part2)



The major problem still exists, memory copy done by copy_user_enhanced_fast_string





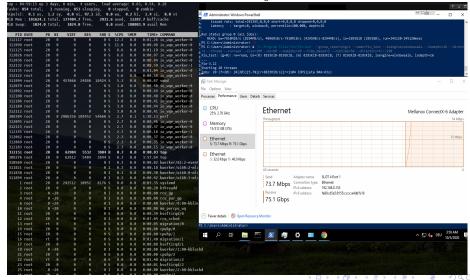




IORING_OP_SENDMSG + IORING_OP_SPLICE (Par

Ib 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"







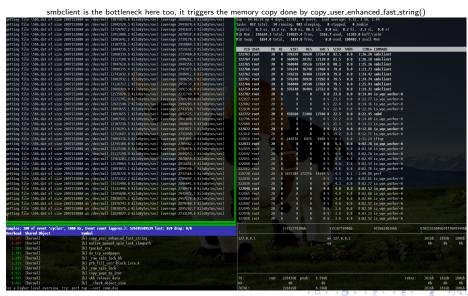
smbclient IORING_OP_SENDMSG/SPLICE (network) 4 connections, 11 GBytes/s, smbd 8.6% cpu, with 4 10.wqe.work threads (pipe to socket) at 20% cpu each.

smbclient is the bottleneck here too etting file \506.dat of size 2097152000 as /dev/mull (2771312.2 KiloBytes/sec) (average 2746784.9 KiloByte etting file \506.dat of size 2097152000 as /dev/mull (3185069.5 KiloBytes/sec) (average 3223967.9 KiloBytes/sec) etting file \506.dat of size 2097152000 as /dev/null (3180123.7 KiloBytes/sec) (average 3176906.8 KiloBytes/sec) etting file \506.dat of size 2097152000 as /dev/null (2824827.2 KiloBytes/sec) (average 2828685.4 KiloBytes/sec) etting file \506.dat of size 2097152000 as /dev/null (3255961.3 KiloBytes/sec) (average 3224002.5 KiloBytes/sec) atting file \506.dat of size 2007152000 as /dev/mull (2782600.3 KiloBytes/sec) (average 2746030.3 KiloBytes/sec) etting file \506.dat of size 2097152000 as /dev/mull (3230203.4 KiloBytes/sec) (average 3176965.0 KiloBytes/sec) etting file \506.dat of size 2097152000 as /dev/mull (3215070.2 KiloBytes/sec) (average 3223992.8 KiloBytes/sec) etting file \506.dat of size 2097152000 as /dev/mull (2790190.4 KiloBytes/sec) (average 2828636.8 KiloBytes/sec) etting file \506.dat of size 2097152000 as /dev/null (2797813.8 KiloBytes/sec) (average 2746894.5 KiloBytes/sec oo - 02:41:58 up 17 days, 17:34, 1 user, load average: 3.97, 4.22, 3.55 asks: 977 total, 5 running, 972 sleeping, 0 stopped, 0 zombie 0.1 us. 4.6 sv. 0.0 ni. 93.5 id. 0.0 ma. 0.0 hi. 1.7 si. 0.0 st 191888.7 total, 127133.7 free, 3813.5 used, 60941.4 buff/cache 287.8 used, 131646.8 avail Mem 188198 root e.e 188:e3.15 htop 1.3 0.0 5:56.39 ksoftirad/45 748176 root 5136 S 1.3 0.0 0:11.20 iftoo 02:41:57 up 3 days, 21:43. 5 users, load average: 1.11, 0.89, 0.62 isks: 877 total, 1 running, 876 sleeping, 0 stopped, 0 zombie 0.1 us. 1.4 sv. 0.0 mi. 97.6 id. 0.0 ma. 0.1 hi. 0.9 si. 0.0 st 191624.1 total, 177248.5 free, 3055.5 used, 11320.1 buff/cache 1824.8 total, 1824.8 free, 0.0 used, 180675.2 avail Mem 8:52 81 in was worker-8 16139 root 8 S 17.9 0.0 8:40.39 io wge worker-8 8 S 17.3 6.6 8:34.48 in was worker-8 8 458888 21264 17652 S 8.6 8.8 8:48.53 subd moles: 78M of event 'cycles', 4888 Hz. Event count (approx.): 35348326236 lost: 8/8 drop: 8/32898 15468384646b 61841218566b7738152448 rhead Shared Object 192,168,18,191 av 192,168,18,198 [k] raw spin lock bh 18.3Mb 18.7Mb 19.6Ml [k] copy page to iter 192,168,18,191 => 192,168,0,153 [k] page cache pipe buf release [k] x86 retpoline rax [k] page cache pipe buf confirm 31468 neak 91.76h 91.56h 89.76 68.7MR 22.1Mb 18.3Mb 18.7Mb 19.6M a higher level overview, try: perf top --sort comm,dso 91.86h 91.56h 89.76



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





More loopback testing on brand new hardware

- Recently I re-did the loopback read tests IORING_OP_SENDMSG/SPLICE (from /dev/shm/)
 - ▶ 1 connection, ~10-13 GBytes/s, smbd 7% cpu, with 4 iou-wrk threads at 7%-50% cpu.
 - ▶ 4 connections, 24-30 GBytes/s, smbd 18% cpu, with 16 iou-wrk threads at 3%-35% cpu.



io_uring (16/22)

More loopback testing on brand new hardware

- Recently I re-did the loopback read tests IORING_OP_SENDMSG/SPLICE (from /dev/shm/)
 - ▶ 1 connection, ~10-13 GBytes/s, smbd 7% cpu, with 4 iou-wrk threads at 7%-50% cpu.
 - ▶ 4 connections, 24-30 GBytes/s, smbd 18% cpu, with 16 iou-wrk threads at 3%-35% cpu.
- I also implemented SMB2 writes with IORING_OP_RECVMSG/SPLICE (tested to /dev/null)
 - ▶ 1 connection, ~7-8 GBytes/s, smbd 5% cpu, with 3 io-wrk threads at 1%-20% cpu.
 - ▶ 4 connections, ~10 GBytes/s, smbd 15% cpu, with 12 io-wrk threads at 1%-20% cpu.



More loopback testing on brand new hardware

- Recently I re-did the loopback read tests
 - IORING_OP_SENDMSG/SPLICE (from /dev/shm/)

 ▶ 1 connection, ~10-13 GBytes/s, smbd 7% cpu,
 - with 4 iou-wrk threads at 7%-50% cpu.
 - ▶ 4 connections, 24-30 GBytes/s, smbd 18% cpu, with 16 iou-wrk threads at 3%-35% cpu.
- I also implemented SMB2 writes with IORING_OP_RECVMSG/SPLICE (tested to /dev/null)
 - ▶ 1 connection, ~7-8 GBytes/s, smbd 5% cpu, with 3 io-wrk threads at 1%-20% cpu.
 - ▶ 4 connections, ~10 GBytes/s, smbd 15% cpu, with 12 io-wrk threads at 1%-20% cpu.
- ▶ I tested with a Linux Kernel 5.13
 - ▶ 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



io_uring (16/22)

SerNet

The road to upstream (TEVENT_FD_ERROR)

000000000

- We need support for TEVENT_FD_ERROR in order to monitor errors
 - When using IORING_OP_SEND,RECVMSG we still want to notice errors
 - This is the main merge request:
 - https://gitlab.com/samba-team/samba/-/merge_requests/2793
 - This merge request converts Samba to use TEVENT_FD_ERROR:
 - https://gitlab.com/samba-team/samba/-/merge_requests/2885
 - ► (It also simplifies other places in the code without io_uring)





The road to upstream (samba_io_uring abstraction)!

API glue to tevent:

- samba_io_uring abstraction factored out of vfs_io_uring:
 - samba_io_uring_ev_hybrid tevent backend (glued on epoll backend)
 - It means every layer getting the tevent_context can use io_uring
 - ▶ No #ifdef's just checking if the required features are available





The road to upstream (samba_io_uring abstraction)

API glue to tevent:

- samba_io_uring abstraction factored out of vfs_io_uring:
 - samba_io_uring_ev_hybrid tevent backend (glued on epoll backend)
 - ▶ It means every layer getting the tevent_context can use io_uring
 - ▶ No #ifdef's just checking if the required features are available





The road to upstream (samba_io_uring abstraction 2)

generic submission/completion api:

- Using it ...
 - convert vfs_io_uring
 - use it in smb2_server.c
 - In future use it in other performance critical places too





4□ > 4周 > 4 = > 4 = > = 90

The road to upstream (samba_io_uring abstraction 2

generic submission/completion api:

- Using it ...
 - convert vfs_io_uring
 - use it in smb2_server.c
 - ▶ In future use it in other performance critical places too.



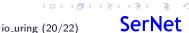


The road to upstream (smb2_server.c)

.....

- Refactoring of smb2_server.c
 - add optional IORING_OP_SENDMSG, IORING_OP_RECVMSG support
- ▶ There are structural problems with splice from a file
 - ▶ I had a discussion with the Linux developers about it:
 - ▶ The page content from the page cache may change unexpectetly
 - https://lists.samba.org/archive/samba-technical/2023-February/thread.html#137945
 - We may not able to use IORING_OP_SENDMSG/SPLICE by default
 - Maybe IORING_OP_RECVMSG/SPLICE is possible
- At least we can have only 1 one copy instead of two:
 - ► IORING_OP_SENDMSG_ZC is able to avoid copying to the socket
 - we get an extra completion once the buffers are not needed anymore
 - This gives good results, between with and without IORING_OP_SENDMSG/SPLICE
 - But I don't have numbers as it doesn't work on loopback
 - Within VM's improvement can be seen





The road to upstream (smb2_server.c)

100000000

- Refactoring of smb2_server.c
 - add optional IORING_OP_SENDMSG, IORING_OP_RECVMSG support
- ▶ There are structural problems with splice from a file
 - ▶ I had a discussion with the Linux developers about it:
 - ▶ The page content from the page cache may change unexpectetly
 - https://lists.samba.org/archive/samba-technical/2023-February/thread.html#137945
 - We may not able to use IORING_OP_SENDMSG/SPLICE by default
 - ► Maybe IORING_OP_RECVMSG/SPLICE is possible
- At least we can have only 1 one copy instead of two:
 - ► IORING_OP_SENDMSG_ZC is able to avoid copying to the socket
 - we get an extra completion once the buffers are not needed anymor
 - This gives good results, between with and without IORING_OP_SENDMSG/SPLICE
 - ▶ But I don't have numbers as it doesn't work on loopback
 - ► Within VM's improvement can be seen





4日 > 4日 > 4日 > 4日 > 日

The road to upstream (smb2_server.c)

00000000

- Refactoring of smb2_server.c
 - add optional IORING_OP_SENDMSG, IORING_OP_RECVMSG support
- ▶ There are structural problems with splice from a file
 - ▶ I had a discussion with the Linux developers about it:
 - ▶ The page content from the page cache may change unexpectetly
 - https://lists.samba.org/archive/samba-technical/2023-February/thread.html#137945
 - We may not able to use IORING_OP_SENDMSG/SPLICE by default
 - Maybe IORING_OP_RECVMSG/SPLICE is possible
- At least we can have only 1 one copy instead of two:
 - ► IORING_OP_SENDMSG_ZC is able to avoid copying to the socket
 - we get an extra completion once the buffers are not needed anymore
 - ► This gives good results, between with and without IORING_OP_SENDMSG/SPLICE
 - But I don't have numbers as it doesn't work on loopback
 - Within VM's improvement can be seen





Future Improvements



- I have a prototype for a native io_uring tevent backend:
 - The idea is to avoid epoll and only block in io_uring_enter()
 - But the semantics of IORING_OP_POLL_ADD.REMOVE are not useable
 - https://lists.samba.org/archive/samba-technical/2022-October/thread.html#137734
 - We may get an IORING_POLL_CANCEL_ON_CLOSE in future
 - And a usable IORING POLL LEVEL





Future Improvements



- ▶ I have a prototype for a native io_uring tevent backend:
 - The idea is to avoid epoll and only block in io_uring_enter()
 - But the semantics of IORING_OP_POLL_ADD,REMOVE are not useable
 - https://lists.samba.org/archive/samba-technical/2022-October/thread.html#137734
 - We may get an IORING_POLL_CANCEL_ON_CLOSE in future
 - And a usable IORING_POLL_LEVEL
- ▶ We can use io_uring deep inside of the smbclient code
 - The low layers can just use samba_io_uring_ev_context_get_ring()
 - And use if available without changing the whole stack





Questions? Feedback!

00000000

- Stefan Metzmacher, metze@samba.org
- ► https://www.sernet.com
- https://samba.plus

→ SerNet/SAMBA+ sponsor booth

 ${\sf Slides:\ https://samba.org/~metze/presentations/2023/SDC/}$



