

# 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 an updated version of this presentation here:
- ▶ <https://samba.org/~metze/presentations/2023/SDC/>

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

- ▶ 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)

- ▶ 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 "[Ring in a new asynchronous I/O API](#)" on LWN.NET
- ▶ This can be nicely integrated with our async event 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 "[Ring in a new asynchronous I/O API](#)" on LWN.NET
- ▶ This can be nicely integrated with our async event 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



- ▶ 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)





- ▶ 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)



- ▶ 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



- ▶ 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



- ▶ 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



- ▶ 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>



- ▶ 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>



- ▶ 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>





# IOURING\_OP\_SENDMSG (Part1)

4 connections, 6.8 GBytes/s, smbdc only uses 11% cpu, (io\_wqework 50% cpu) per connection, we still use >300% cpu in total

```
top - 05:45:38 up 2 days, 46 min, 2 users, load average: 3.03, 2.84, 1.61
threads: 823 total, 3 running, 820 sleeping, 0 stopped, 0 zombie
cpu(s): 0.1 us, 4.7 sy, 0.0 ni, 94.6 id, 0.0 wa, 0.1 hi, 0.0 si, 0.0 st
MEM Mem : 191624.1 total, 182194.6 free, 2702.6 used, 6726.9 buff/cache
MEM Swap: 1024.0 total, 1024.0 free, 0.0 used, 185554.7 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
307577	root	20	0	0	0	0	R	49.0	0.0	0:05.00	io_wqeworker-0
307549	root	20	0	0	0	0	S	46.0	0.0	0:21.39	io_wqeworker-0
307555	root	20	0	0	0	0	R	44.0	0.0	0:21.45	io_wqeworker-0
307567	root	20	0	0	0	0	S	29.8	0.0	0:09.92	io_wqeworker-1
307558	root	20	0	663100	144024	18804	S	23.2	0.1	0:09.10	smbd
307556	root	20	0	663100	144024	18804	S	19.9	0.1	0:08.95	smbd
307559	root	20	0	663100	144024	18804	S	19.5	0.1	0:08.92	smbd
307563	root	20	0	663100	144024	18804	S	19.5	0.1	0:08.86	smbd
307557	root	20	0	663100	144024	18804	S	19.2	0.1	0:09.11	smbd
307560	root	20	0	663100	144024	18804	S	19.2	0.1	0:09.38	smbd
307561	root	20	0	663100	144024	18804	S	19.2	0.1	0:09.07	smbd
307534	root	20	0	663100	144024	18804	S	18.9	0.1	0:09.00	smbd
307576	root	20	0	663100	144024	18804	S	18.9	0.1	0:05.61	smbd
307562	root	20	0	663100	144024	18804	S	18.5	0.1	0:08.93	smbd
307530	root	20	0	663100	144024	18804	D	11.3	0.1	0:05.16	smbd
307552	root	20	0	0	0	0	S	9.3	0.0	0:12.25	io_wqeworker-0
417	root	20	0	0	0	0	I	0.3	0.0	0:03.58	kworker/0:2-event
307183	root	20	0	0	0	0	I	0.3	0.0	0:00.61	kworker/u160:2-ml
307568	root	20	0	0	0	0	I	0.3	0.0	0:00.02	kworker/29:0-event
307588	root	20	0	62964	5532	3904	R	0.3	0.0	0:00.12	top
1	root	20	0	242512	10952	8176	S	0.0	0.0	0:02.84	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.13	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kblc
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00.32	kssoftirqd/0
12	root	20	0	0	0	0	I	0.0	0.0	0:03.17	rcu_sched
13	root	rt	0	0	0	0	S	0.0	0.0	0:00.03	migration/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
16	root	rt	0	0	0	0	S	0.0	0.0	0:01.38	migration/1
17	root	20	0	0	0	0	S	0.0	0.0	0:00.07	kssoftirqd/1
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-kblc
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
22	root	rt	0	0	0	0	S	0.0	0.0	0:01.37	migration/2
23	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kssoftirqd/2
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0H-kblc
26	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/3
27	root	rt	0	0	0	0	S	0.0	0.0	0:01.39	migration/3

```
Administrator: Windows PowerShell

complete : 0=0.0%, 4=100.0%, 8=0.1%, 16=0.1%, 32=0.0%, 64=0.0%, >=64=0.0%
issued rwts: total=64728,0,0,0 short=0,0,0,0 dropped=0,0,0,0
latency : target=0, window=0, percentile=100.00%, depth=16

Run status group 0 (all jobs):
READ: bs=5396KiB/s (5658MB/s), 4096KiB/s-5396KiB/s (4295MB/s-5658MB/s), io=2536iB (271G
PS C:\Users\Administrator> & (Get-ProgramFiles32) |> group-reporting-1 -name=fi
s1 --thread --rwwread --size=100M --bs=4M --numjobs=2 --time_based=1 --runtime=5m --direct
fiio_test: (g=0): rwwread, bs=(R) 4096KiB-4096KiB, (M) 4096KiB-4096KiB, (T) 4096KiB-4096KiB
...
fiio-3.22
Starting 2 threads
Jobs: 2 (f=2): [R:2][15.3W][r=6816MiB/s][r=1704 IOPS][eta 04m:14s]
```

Task Manager

File Options View

Processes Performance Users Details Services

- CPU 16% 2.78 GHz
- Memory 12/512 GB (2%)
- Ethernet S: 17.4 Mbps R: 57.5 Gbps
- Ethernet S: 32.0 Kbps R: 96.0 Kbps

### Ethernet

Throughput

60 seconds

Send: 17.4 Mbps  
Receive: 57.5 Gbps

Adapter name: SLOT 4 Port 1  
Connection type: Ethernet  
IPv4 address: 192.168.0.153  
IPv6 address: fe80:d5a5:8155:ccccca4db%19

Fewer details Open Resource Monitor

5 items

# IOURING\_OP\_SENDMSG (Part2)

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

```
amples: 178K of event 'cycles', 4000 Hz, Event count (approx.): 87301350677 Lost: 0/0 dropped: 0/0
verhead Shared Object Symbol
05.07% [kernel] [k] copy_user_enhanced_fast_string
0.28% [kernel] [k] shmem_file_read_iter
1.79% [kernel] [k] tcp_sendmsg_locked
1.29% [kernel] [k] find_get_entry
1.21% [kernel] [k] get_page_from_freelist
0.97% [kernel] [k] __list_del_entry_valid
0.87% [kernel] [k] native_queued_spin_lock_slowpath
0.80% [kernel] [k] __raw_spin_lock
0.60% [kernel] [k] skb_release_data
0.50% [kernel] [k] mlx5e_sq_xmit
0.38% [kernel] [k] __free_pages_ok
0.37% [kernel] [k] __raw_spin_lock_irqsave
0.35% [kernel] [k] __zone_watermark_ok
0.33% [kernel] [k] unlock_page
0.32% [kernel] [k] copy_page_to_iter
0.31% [kernel] [k] find_lock_entry
0.31% [kernel] [k] __alloc_pages_nodemask
0.30% [kernel] [k] mlx5e_poll_tx_cq
0.29% [kernel] [k] page_mapping
0.28% [kernel] [k] xas_load
0.27% [kernel] [k] shmem_getpage_gfp
0.25% [kernel] [k] __check_object_size
0.23% [kernel] [k] tcp_wfree
0.22% [kernel] [k] __slab_free
0.21% [kernel] [k] __sched_text_start
0.20% [kernel] [k] __free_one_page
0.20% [kernel] [k] mark_page_accessed
0.20% [kernel] [k] bad_range
0.19% [kernel] [k] tcp_rbtrees_insert
0.19% [kernel] [k] iov_iter_advance
0.19% [kernel] [k] native_irq_return_iret
0.18% [kernel] [k] tcp_write_xmit
0.17% [kernel] [k] __alloc_skb
0.16% [kernel] [k] tasklet_action_common.isra.0
0.15% [kernel] [k] clear_page_erms
0.14% [kernel] [k] do_syscall_64
0.14% [kernel] [k] __tcp_transmit_skb
0.13% [kernel] [k] __skb_clone
0.13% [kernel] [k] memcpy_erms
0.13% [kernel] [k] menu_select
0.12% [kernel] [k] __list_add_valid
0.12% [kernel] [k] mlx5_eq_comp_int
0.11% [kernel] [k] tcp_ack
```

The screenshot shows the Windows Task Manager Performance tab. The 'Ethernet' section is highlighted, showing a throughput of 15.7 Mbps (Send) and 57.5 Gbps (Receive). The 'Send' section is expanded, showing a throughput of 15.7 Mbps. The 'Receive' section shows a throughput of 57.5 Gbps. The 'Adapter name' is SLOT 4 Port 1, the 'Connection type' is Ethernet, the 'IPv4 address' is 192.168.0.153, and the 'IPv6 address' is fe80:d5a5:815.

```
Administrator: Windows PowerShell
complete : 0=0.0%, 4=100.0%, 8=0.1%, 16=0.1%, 32=0.0%, 64=0.0%, >=64
issued rmts: total=64728,0,0 short=0,0,0 dropped=0,0,0
latency : target=0, window=0, percentile=100.00%, depth=16

Run status group 0 (all jobs):
READ: bws=5396MiB/s (5658MB/s), 4096MiB/s-5396MiB/s (4295MB/s-5658MB/s),
PS C:\Users\Administrator> & 'C:\Program Files\Fio\Fio.exe' --group_report
n1 --thread --rwrread --size=100M --bs=4M --numjobs=2 --time_based=1 --run
fio_test: (g=0): rw=read, bs=(R) 4096KiB-4096KiB, (W) 4096KiB-4096KiB, (T)
...
fio-3.22
Starting 2 threads
Jobs: 2 (f=2): [R(2)][22.0%][r=6811MiB/s][r=1702 IOPS][eta 03m:54s]

Task Manager
File Options View
Processes Performance Users Details Services

CPU 16% 2.78 GHz
Memory 12/512 GB (2%)
Ethernet Si: 15.7 Mbps R: 57.5 Gbps
Ethernet Si: 40.0 Kbps R: 96.0 Kbps

Ethernet
Throughput
Send and receive activity network
60 seconds
Send 15.7 Mbps
Receive 57.5 Gbps
Adapter name: SLOT 4 Port 1
Connection type: Ethernet
IPv4 address: 192.168.0.153
IPv6 address: fe80:d5a5:815

Fewer details Open Resource Monitor
5 items
```

# IOURING\_OP\_SENDMSG + IOURING\_OP\_SPLICE (Part 1)

10 connections, 8.9 GBytes/s, smbdc 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"

```
top - 04:59:15 up 3 days, 0 min, 4 users, load average: 0.63, 0.54, 0.28
Tasks: 854 total, 1 running, 853 sleeping, 0 stopped, 0 zombie
cpu(s): 0.1 us, 1.2 sy, 0.0 ni, 97.1 id, 0.0 wa, 0.2 hi, 1.4 si, 0.0 st
MiB Mem : 191624.1 total, 177484.7 free, 2931.6 used, 11287.7 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 180893.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	VCPU	%MEM	TIME	COMMAND
312111	root	20	0	0	0	0	S	12.3	0.0	0:01.12	io_wqe_worker-0
311990	root	20	0	0	0	0	S	11.0	0.0	0:00:00	io_wqe_worker-0
312125	root	20	0	0	0	0	S	8.6	0.0	0:01:19	io_wqe_worker-0
312026	root	20	0	0	0	0	S	6.6	0.0	0:00:97	io_wqe_worker-0
312036	root	20	0	0	0	0	S	6.6	0.0	0:00:94	io_wqe_worker-0
312132	root	20	0	0	0	0	S	6.0	0.0	0:00:59	io_wqe_worker-1
312135	root	20	0	0	0	0	S	6.0	0.0	0:01:04	io_wqe_worker-0
312122	root	20	0	0	0	0	S	5.6	0.0	0:00:58	io_wqe_worker-1
311994	root	20	0	457060	24880	18424	S	5.3	0.0	0:00:07	smbd
312079	root	20	0	0	0	0	S	3.0	0.0	0:00:40	io_wqe_worker-0
312092	root	20	0	0	0	0	S	3.0	0.0	0:00:44	io_wqe_worker-0
312100	root	20	0	0	0	0	S	3.0	0.0	0:00:40	io_wqe_worker-0
312106	root	20	0	0	0	0	S	3.0	0.0	0:00:41	io_wqe_worker-0
312109	root	20	0	0	0	0	S	3.0	0.0	0:00:44	io_wqe_worker-0
312112	root	20	0	0	0	0	S	3.0	0.0	0:00:41	io_wqe_worker-0
308304	root	20	0	2986356	108452	54660	S	2.7	0.1	1:38:13	perf
312895	root	20	0	0	0	0	S	2.7	0.0	0:00:46	io_wqe_worker-0
312115	root	20	0	0	0	0	S	2.7	0.0	0:00:37	io_wqe_worker-0
312145	root	20	0	0	0	0	S	2.7	0.0	0:00:18	io_wqe_worker-1
312062	root	20	0	0	0	0	S	2.3	0.0	0:00:37	io_wqe_worker-0
312060	root	20	0	0	0	0	S	2.3	0.0	0:00:35	io_wqe_worker-0
312183	root	20	0	0	0	0	S	2.3	0.0	0:00:15	io_wqe_worker-0
312151	root	20	0	62984	5532	3804	R	0.7	0.0	0:00:03	top
308276	root	20	0	62812	5404	3844	S	0.3	0.0	3:57:04	top
310560	root	20	0	0	0	0	I	0.3	0.0	0:00:02	kworker/61:2-event
311821	root	20	0	0	0	0	I	0.3	0.0	0:00:18	kworker/u168:2-nl
311830	root	20	0	0	0	0	I	0.3	0.0	0:00:38	kworker/u168:0-nl
311894	root	20	0	0	0	0	I	0.3	0.0	0:00:42	kworker/u168:3-nl
1	root	20	0	242512	10952	8176	S	0.0	0.0	0:03:35	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00:28	kthread
3	root	0	-20	0	0	0	I	0.0	0.0	0:00:00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00:00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00:00	kworker/8:0-kblock
18	root	0	-20	0	0	0	I	0.0	0.0	0:00:00	ms_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00:39	ksftirq/0
12	root	20	0	0	0	0	I	0.0	0.0	0:07:04	rcu_sched
13	root	rt	0	0	0	0	S	0.0	0.0	0:00:05	migration/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00:00	cpuhp/0
15	root	20	0	0	0	0	S	0.0	0.0	0:00:00	cpuhp/1
16	root	rt	0	0	0	0	S	0.0	0.0	0:01:40	migration/1
17	root	20	0	0	0	0	S	0.0	0.0	0:00:00	ksftirq/1
19	root	0	-20	0	0	0	I	0.0	0.0	0:00:00	kworker/7:0H-kblock
21	root	20	0	0	0	0	S	0.0	0.0	0:00:00	cpuhp/2
22	root	rt	0	0	0	0	S	0.0	0.0	0:01:40	migration/2
23	root	20	0	0	0	0	S	0.0	0.0	0:00:01	ksftirq/2
25	root	0	-20	0	0	0	I	0.0	0.0	0:00:00	kworker/2:0H-kblock
26	root	15	0	0	0	0	S	0.0	0.0	0:00:00	cpuhp/3

```
Administrator: Windows PowerShell
C:\> issued nsts: total=242165,0,0,0 short=0,0,0,0 dropped=0,0,0,0
latency : target=0, window=0, percentile=100,000, depth=16

C:\> Run status group 0 (all jobs):
PS C:\Users\Administrator> PS C:\Users\Administrator> & C:\Program Files\Fio\Fio.exe --group_reporting=1 --name=fio_test --ioengine=windowsaio --iodepth=16 --direct=6 --latency=target=100 --window=0 --percentile=100 --depth=16 --file-test=[p00] r=redraw,bs=(R) 8192Kx16,(M) 8192Kx16,(W) 8192Kx16,(F) 8192Kx16,ioengine=windowsaio,iodpth=16
1:***
1:fio-3.22
1:Starting 20 threads
1:Jobs: 20 (+*0): [R(20)][5.7%][r=8833M/s][w=1104 IOPS][eta=4m:4s]
```

The screenshot shows the Windows Task Manager Performance tab. The Ethernet section is highlighted, showing a throughput of 73.7 Mbps. The adapter name is SLOT 4 Port 1. The connection type is Ethernet. The IPv4 address is 192.168.0.153. The IPv6 address is fe80::5d81515:cccc4db%19. The adapter is a Mellanox ConnectX-6 Adapter. The throughput graph shows a sharp spike in activity.



Stefan Metzmacher

io\_uring (13/22)



# 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

```
getting file /506.dat of size 2097152000 as /dev/null (2771312.2 KiBytes/sec) (average 2746704.9 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3185069.5 KiBytes/sec) (average 3223967.9 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3180123.7 KiBytes/sec) (average 3176906.8 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (2824427.2 KiBytes/sec) (average 2829685.4 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3225598.1 KiBytes/sec) (average 3224002.5 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (2782680.3 KiBytes/sec) (average 2748030.3 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3230283.4 KiBytes/sec) (average 3176965.8 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3215070.2 KiBytes/sec) (average 3223992.8 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (2790190.4 KiBytes/sec) (average 2828636.8 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3185069.5 KiBytes/sec) (average 3176974.6 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (2797813.0 KiBytes/sec) (average 2748094.5 KiBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3250793.1 KiBytes/sec) (average 3224021.8 KiBytes/sec)
```

```
top - 02:41:58 up 17 days, 17:34, 1 user, load average: 3.07, 4.22, 3.55
Tasks: 977 total, 5 running, 972 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.1 us, 4.6 sy, 0.0 ni, 93.5 id, 0.0 wa, 0.0 hi, 1.7 si, 0.0 st
Mem Mem: 191880.7 total, 127133.7 free, 3813.5 used, 60941.4 buff/cache
Mem Swap: 1824.0 total, 737.0 free, 287.0 used, 131646.0 avail Mem
```

PID	USER	PR	NI	VIKT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
740188	root	20	0	375680	35960	16852	R	99.3	0.0	0:35.55	smbclient
740185	root	20	0	375664	36180	17016	R	99.0	0.0	0:30.87	smbclient
740187	root	20	0	375692	35880	16696	R	88.1	0.0	0:44.88	smbclient
740186	root	20	0	375652	35896	16740	R	66.4	0.0	0:49.28	smbclient
180190	root	20	0	31540	7872	3412	S	2.0	0.0	100:03.15	htop
238	root	20	0	0	0	0	S	1.3	0.0	0:56.39	kssoftirq/45
740176	root	20	0	249536	8076	5136	S	1.3	0.0	0:13.28	lftp

```
top - 02:41:57 up 3 days, 21:43, 5 users, load average: 1.11, 0.89, 0.62
Tasks: 877 total, 1 running, 876 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.1 us, 1.4 sy, 0.0 ni, 97.6 id, 0.0 wa, 0.1 hi, 0.9 si, 0.0 st
Mem Mem: 191824.1 total, 17248.5 free, 3855.5 used, 11320.1 buff/cache
Mem Swap: 1824.0 total, 1824.0 free, 0.0 used, 180675.2 avail Mem
```

PID	USER	PR	NI	VIKT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
310136	root	20	0	0	0	0	S	21.3	0.0	0:52.81	io_wqe_worker-0
310133	root	20	0	0	0	0	S	20.3	0.0	0:53.37	io_wqe_worker-0
310139	root	20	0	0	0	0	S	17.9	0.0	0:48.39	io_wqe_worker-0
310121	root	20	0	0	0	0	S	17.3	0.0	0:34.48	io_wqe_worker-0
310116	root	20	0	458080	21264	17652	S	8.6	0.0	0:46.53	smbd

Sampls: 788 of event 'cycles', 4000 Hz, Event count (approx.): 35348326236 lost: 0/0 drop: 0/32098

Overhead	shared object	Symbol
7.0%	[kernel]	[k] do_tcp_sendpages
5.37%	[kernel]	[k] raw_spin_lock_bh
4.80%	[kernel]	[k] copy_page_to_iter
3.75%	[kernel]	[k] page_cache_pipe_buf_release
3.25%	[kernel]	[k] __x86_retpoline_rax
3.09%	[kernel]	[k] page_cache_pipe_buf_confirm
2.87%	[kernel]	[k] native_mound_spin_lock_slowpath
2.89%	[kernel]	[k] shmem_file_read_iter
2.79%	[kernel]	[k] inet_sendpage
2.61%	[kernel]	[k] tcp_sendpage

For a higher level overview, try: perf top --sort comm,dso

	1546838464cb	3892869928cb	4638801264cb	6184121056db778152448cb
192.168.10.191	=>	192.168.10.150		91.7Gb 91.5Gb 89.7Gb
	<=			18.3Mb 18.7Mb 19.6Mb
192.168.10.151	=>	192.168.0.153		0b 0b 238b
	<=			0b 0b 218b
TX:	cus:	3146B peak:	0b	rates: 91.7Gb 91.5Gb 89.7Gb
RX:		68.7MB	22.1Mb	18.3Mb 18.7Mb 19.6Mb
TOTAL:		3146B	0b	91.8Gb 91.5Gb 89.7Gb

# smbclient IOURING\_OP\_SENDMSG/SPLICE (loopback)

8 connections, 22 GBytes/s, smbdc 22% cpu, with 4 io\_wqework 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()

```
getting file %SMB_DAT of size 2097152000 as /dev/null (3075974.6 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2945250.6 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2719787.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2951088.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2801641.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3107770.5 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2664736.5 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2603374.8 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3117180.9 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3047618.0 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3089355.4 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2741632.0 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3082932.1 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3126717.1 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2893890.4 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2515970.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2173791.8 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2921540.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3093655.1 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3093655.3 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3087341.7 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3107770.5 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3102070.1 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2726897.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3084316.8 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2745380.0 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3117180.9 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3117180.9 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2563829.7 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2519864.9 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2893655.1 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2838780.9 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2773312.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3131480.8 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3131480.8 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2595980.4 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3083875.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2976743.8 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3083875.2 KiloBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2704827.2 KiloBytes/sec)
```

```
top - 04:00:58 up 4 days, 23:02, 0 users, load average: 0.15, 3.56, 1.44
Tasks: 937 total, 14 running, 903 sleeping, 0 stopped, 0 zombie
Cpus(s): 0.3 us, 11.2 sy, 0.0 ni, 0.0 mi, 0.1 wa, 0.0 hi, 2.1 si, 0.0 st
Mem Mem : 191624.1 total, 176925.4 free, 3316.7 used, 11382.0 buff/cache
Mem Swap: 1024.0 total, 1024.0 free, 0.0 used, 100463.7 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	GPU	MEM	TIME+	COMMAND
322763	root	20	0	376220	36680	17364	R	0	0	1:26.28	smbclient
322764	root	20	0	368036	28182	17128	R	0	0	1:26.18	smbclient
322765	root	20	0	368040	28516	17164	R	0	0	1:25.16	smbclient
322766	root	20	0	376245	36740	17468	R	0	0	1:23.73	smbclient
322762	root	20	0	376236	36680	17228	R	0	0	1:24.42	smbclient
322761	root	20	0	376248	28928	17292	R	0	0	1:24.74	smbclient
322766	root	20	0	368040	28540	17464	R	0	0	1:25.93	smbclient
322759	root	20	0	370140	36464	17332	R	0	0	1:24.34	smbclient
322762	root	20	0	0	0	0	R	22.8	0.0	0:18.84	io_wqeworker-0
322827	root	20	0	0	0	0	R	23.5	0.0	0:12.77	io_wqeworker-0
322882	root	20	0	0	0	0	R	22.8	0.0	0:14.36	io_wqeworker-0
322838	root	20	0	0	0	0	R	22.8	0.0	0:12.96	io_wqeworker-0
322772	root	20	0	458260	21488	17596	R	0	0	0:22.45	smbd
322796	root	20	0	0	0	0	R	22.2	0.0	0:14.68	io_wqeworker-0
322880	root	20	0	0	0	0	R	21.5	0.0	0:14.13	io_wqeworker-0
322822	root	20	0	0	0	0	R	21.5	0.0	0:17.06	io_wqeworker-0
322818	root	20	0	0	0	0	R	19.2	0.0	0:17.71	io_wqeworker-0
318818	root	20	0	244876	6976	4988	S	0	0	1:31.20	iftop
322833	root	20	0	0	0	0	R	5.3	0.0	0:02.78	io_wqeworker-0
322854	root	20	0	0	0	0	R	5.0	0.0	0:02.50	io_wqeworker-0
322842	root	20	0	0	0	0	R	4.6	0.0	0:02.78	io_wqeworker-0
322851	root	20	0	0	0	0	R	4.6	0.0	0:02.49	io_wqeworker-0
322860	root	20	0	0	0	0	R	4.6	0.0	0:02.54	io_wqeworker-0
322862	root	20	0	0	0	0	R	4.0	0.0	0:02.70	io_wqeworker-0
317170	root	20	0	303718	172756	54364	S	0	0	1:48.30	perf
322836	root	20	0	0	0	0	R	4.3	0.0	0:02.61	io_wqeworker-0
322839	root	20	0	0	0	0	R	4.3	0.0	0:02.77	io_wqeworker-0
322948	root	20	0	0	0	0	R	4.0	0.0	0:02.52	io_wqeworker-0
322865	root	20	0	0	0	0	R	4.0	0.0	0:02.68	io_wqeworker-0
322868	root	20	0	0	0	0	R	4.0	0.0	0:02.66	io_wqeworker-0
322887	root	20	0	0	0	0	R	4.0	0.0	0:02.57	io_wqeworker-0
322845	root	20	0	0	0	0	R	3.6	0.0	0:02.50	io_wqeworker-0
322858	root	20	0	0	0	0	R	3.6	0.0	0:02.33	io_wqeworker-0
322858	root	20	0	0	0	0	R	3.6	0.0	0:02.52	io_wqeworker-0

Samples: 30M of event 'cycles', 1800 r/s, Event count (approx.): 52678559529 Lost: 0/0 drop: 0/0

Overhead	Shared object	Symbol
51.14%	[kernel]	[k] copy_user_enhanced_fast_string
6.40%	[kernel]	[k] native_queue_spin_lock_slowpath
3.30%	[kernel]	[k] tcpackit_recv
3.09%	[kernel]	[k] do_tcp_sendpages
3.02%	[kernel]	[k] raw_spin_lock_bh
3.02%	[kernel]	[k] prb_fill_curr_block_isra.0
3.01%	[kernel]	[k] raw_spin_lock
0.92%	[kernel]	[k] copy_page_to_iter
0.89%	[kernel]	[k] skb_release_data
0.89%	[kernel]	[k] _check_object_size

	157537920b	315187500b	4726614016b	6382151600b/8777603140b
127.0.0.1		ms 127.0.0.1		1816b 1816b 1806b
		<=		0b 0b 0b
Tx:	cum: 2264240	peak: 6.596b		rates: 1816b 1816b 1806b
Rx:	0b	0b		0b 0b 0b
TOTAL:	2264240	6.596b		1816b 1816b 1806b



Stefan Metzmacher

io\_uring (15/22)



# 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.
- ▶ 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

# 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

# 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



- ▶ 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](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](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 1)

API glue to tevent:

```
void samba_io_uring_ev_register(void);

const struct samba_io_uring_features *samba_io_uring_system_features(void);

struct samba_io_uring *samba_io_uring_ev_context_get_ring(struct tevent_context *ev);

const struct samba_io_uring_features *samba_io_uring_get_features(
    const struct samba_io_uring *ring);

ev = tevent_context_init_byname(mem_ctx, "samba_io_uring_ev");
```

- ▶ 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 1)

API glue to tevent:

```
void samba_io_uring_ev_register(void);

const struct samba_io_uring_features *samba_io_uring_system_features(void);

struct samba_io_uring *samba_io_uring_ev_context_get_ring(struct tevent_context *ev);

const struct samba_io_uring_features *samba_io_uring_get_features(
    const struct samba_io_uring *ring);

ev = tevent_context_init_byname(mem_ctx, "samba_io_uring_ev");
```

- ▶ 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:

```
void samba_io_uring_completion_prepare(struct samba_io_uring_completion *completion,
                                       void (*completion_fn)(struct samba_io_uring_completion *completion,
                                                               void *completion_private,
                                                               const struct io_uring_cqe *cqe),
                                       void *completion_private);

void samba_io_uring_submission_prepare(struct samba_io_uring_submission *submission,
                                       void (*submission_fn)(struct samba_io_uring *ring,
                                                               struct samba_io_uring_submission *submission,
                                                               void *submission_private),
                                       void *submission_private,
                                       struct samba_io_uring_completion *completion);

struct io_uring_sqe *samba_io_uring_submission_sqe(struct samba_io_uring_submission *
                                                    submission);

size_t samba_io_uring_queue_submissions(struct samba_io_uring *ring,
                                         struct samba_io_uring_submission *submission);
```

▶ 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 unexpectedly
  - ▶ <https://lists.samba.org/archive/samba-technical/2023-February/thread.html#137945>
  - ▶ We may not be 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)

- ▶ 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 unexpectedly
  - ▶ <https://lists.samba.org/archive/samba-technical/2023-February/thread.html#137945>
  - ▶ We may not be 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)

- ▶ 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 unexpectedly
  - ▶ <https://lists.samba.org/archive/samba-technical/2023-February/thread.html#137945>
  - ▶ We may not be 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





- ▶ 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



- ▶ 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

- ▶ Stefan Metzmacher, [metze@samba.org](mailto:metze@samba.org)
- ▶ <https://www.sernet.com>
- ▶ <https://samba.plus>

→ SerNet/SAMBA+ sponsor booth

Slides: <https://samba.org/~metze/presentations/2023/SDC/>