

SAMBA

EXPERIENCE

io_uring

Status Update within Samba

Stefan Metzmacher <metze@samba.org>

Samba Team / SerNet

2023-05-10

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

Topics

- ▶ 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 / SDC 2021)

- ▶ 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)

Last Status Updates (SDC 2020 / SDC 2021)

- ▶ 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)

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 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 "[Ring 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)
- ▶ 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 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.
- ▶ 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 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 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()

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

```
top - 05:43:16 up 2 days, 44 min, 7 users, load average: 5.42, 3.22, 1.52
Threads: 823 total, 33 running, 790 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0 us, 6.3 sy, 0.0 ni, 93.4 id, 0.0 wa, 0.1 hi, 0.2 si, 0.0 st
Mem: Mem : 191624.0 total, 182280.4 free, 2617.5 used, 6726.1 buff/cache
Mem Swap: 1024.0 total, 1024.0 free, 0.0 used, 185640.1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
307372	root	20	0	2426196	62088	19104	R	96.0	0.0	0:52.24	sendd
307406	root	20	0	2426196	63408	19104	R	14.3	0.0	0:06.96	sendd
307412	root	20	0	2426196	65256	19104	R	14.0	0.0	0:06.92	sendd
307418	root	20	0	2426196	63146	19104	R	13.6	0.0	0:06.82	sendd
307410	root	20	0	2426196	64664	19104	R	13.6	0.0	0:06.77	sendd
307414	root	20	0	2426196	65520	19104	R	13.6	0.0	0:06.80	sendd
307422	root	20	0	2426196	68952	19104	R	13.6	0.0	0:06.78	sendd
307432	root	20	0	2426196	71592	19104	R	13.6	0.0	0:06.66	sendd
307408	root	20	0	2426196	63936	19104	R	13.3	0.0	0:06.58	sendd
307411	root	20	0	2426196	64992	19104	R	13.3	0.0	0:06.77	sendd
307413	root	20	0	2426196	65256	19104	R	13.3	0.0	0:06.68	sendd
307415	root	20	0	2426196	65520	19104	R	13.3	0.0	0:06.63	sendd
307418	root	20	0	2426196	66048	19104	R	13.3	0.0	0:06.69	sendd
307419	root	20	0	2426196	67104	19104	R	13.3	0.0	0:06.84	sendd
307428	root	20	0	2426196	67632	19104	R	13.3	0.0	0:06.78	sendd
307421	root	20	0	2426196	68160	19104	R	13.3	0.0	0:06.71	sendd
307423	root	20	0	2426196	69408	19104	R	13.3	0.0	0:06.68	sendd
307425	root	20	0	2426196	69408	19104	R	13.3	0.0	0:06.59	sendd
307428	root	20	0	2426196	70800	19104	R	13.3	0.0	0:06.59	sendd
307438	root	20	0	2426196	70800	19104	R	13.3	0.0	0:06.84	sendd
307433	root	20	0	2426196	72384	19104	R	13.3	0.0	0:06.61	sendd
307426	root	20	0	2426196	70800	19104	R	13.0	0.0	0:06.62	sendd
307429	root	20	0	2426196	70800	19104	R	13.0	0.0	0:06.67	sendd
307434	root	20	0	2426196	72384	19104	R	13.0	0.0	0:06.78	sendd
307435	root	20	0	2426196	72648	19104	R	13.0	0.0	0:06.71	sendd
307407	root	20	0	2426196	62672	19104	R	12.6	0.0	0:06.58	sendd
307416	root	20	0	2426196	66048	19104	R	12.6	0.0	0:06.60	sendd
307417	root	20	0	2426196	66312	19104	R	12.6	0.0	0:06.53	sendd
307427	root	20	0	2426196	70800	19104	R	12.6	0.0	0:06.87	sendd
307431	root	20	0	2426196	71064	19104	R	12.6	0.0	0:06.58	sendd
307424	root	20	0	2426196	69408	19104	R	12.3	0.0	0:06.65	sendd
307409	root	20	0	2426196	64200	19104	R	12.0	0.0	0:06.68	sendd
307404	root	20	0	2426196	62616	19104	D	11.3	0.0	0:06.61	sendd
307183	root	20	0	0	0	0	I	0.3	0.0	0:00.41	kworker/u160:2-als
307302	root	20	0	0	0	0	I	0.3	0.0	0:00.03	kworker/23:1-event
307452	root	20	0	62020	5536	3936	R	0.3	0.0	0:00.00	top
1	root	20	0	242512	18952	8176	S	0.0	0.0	0:02.84	system
2	root	20	0	0	0	0	S	0.0	0.0	0:00.13	khthread
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_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/0:00-kblockd
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	ksoftirqd/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

The screenshot shows a Windows PowerShell window running a benchmark script. The script output indicates that the system is bound by CPU, with a total of 4 connections and a throughput of approximately 3.8 GBytes/s. The script also shows the number of threads and the time taken for each job.

The Task Manager window shows the system's performance, including CPU usage (8% at 2.78 GHz), Memory usage (12/312 GB at 2%), and Network usage (9.3 Mbps R, 31.9 Gbps). The network adapter is identified as Mellanox ConnectX-6 Adaptor, and the connection type is Ethernet. The IPv4 address is 192.168.0.153, and the IPv6 address is fe80:d5e5:8155:ccccca4b%19.

IOURING_OP_SENDFMSG (Part1)

4 connections, ~6.8 GBytes/s, smb2 only uses ~11% cpu, (io_wqe_work ~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, 2.04, 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, 0.5 si, 0.0 st
MiB Mem: 191624.1 total, 182194.6 free, 2702.6 used, 6726.9 buff/cache
MiB 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_wqe_worker-0
307549	root	20	0	0	0	0	S	46.0	0.0	0:21.39	io_wqe_worker-0
307555	root	20	0	0	0	0	R	44.0	0.0	0:21.45	io_wqe_worker-0
307567	root	20	0	0	0	0	S	29.8	0.0	0:09.92	io_wqe_worker-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	0:09.36	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_wqe_worker-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-kblor
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-kblor
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-kblor
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 rws: total=64728,0,0,0 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: bw=5396KiB/s (5650MB/s), 4096KiB/s-5396KiB/s (4295MB/s-5650MB/s), io=2536KiB (2710
PS C:\Users\Administrator> & {dir program files\io\io.exe -group_reporting -name=io
*1 -thread -r=write -size=100M --bs=4M --numjobs=3 -time_based=1 --runtime=5m --direct
io_test: (g=0) r=write, bs=(R) 4096KiB-4096KiB, (W) 4096KiB-4096KiB, (T) 4096KiB-4096KiB,
...
io-3.22
Starting 2 threads
Jobs: 2 (f=2): [R(2)][15.3M][r=6816MiB/s][r=1704 IOPS][eta 04m:14s]
```

Task Manager Performance

- 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

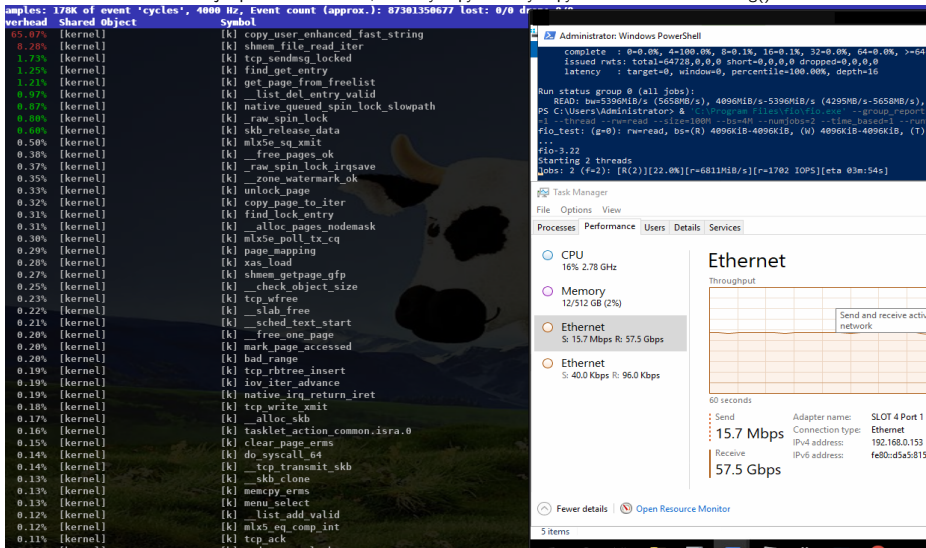
60 seconds

- 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:d5a5:0155:cccc:a4db%19

IORING_OP_SENDMSG (Part2)

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



The screenshot displays the Windows Task Manager Performance tab on the left and a Windows PowerShell terminal window on the right. The Performance tab shows system metrics: CPU at 16% (2.78 GHz), Memory at 12/512 GB (2%), Ethernet at 57.5 Gbps, and another Ethernet adapter at 96.0 Kbps. The PowerShell terminal shows the output of a `runstatus` command, including statistics for read and write operations, and a list of running processes. The background of the Task Manager window is a game scene featuring a character with a large, dark, horned headpiece.

```
amples: 178K of event 'cycles', 4000 Hz, Event count (approx.): 87301350677 lost: 0/0 dropped: 0/0
verhead Shared Object Symbol
65.07% [kernel] [k] copy_user_enhanced_fast_string
 8.28% [kernel] [k] shmem_file_read_iter
 1.73% [kernel] [k] tcp_sendmsg_locked
 1.25% [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
```

```
Administrator: Windows PowerShell

complete : 0=0.0%, 4=100.0%, 8=0.1%, 16=0.1%, 32=0.0%, 64=0.0%, >64
issued ruts: 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: bw=5396MiB/s (5658MB/s), 4096MiB/s-5396MiB/s (4295MB/s-5658MB/s),
PS C:\Users\Administrator> & 'C:\Program Files\Fio\Fio.exe' --group_report
#1 --thread --rurread --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 (r=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
```

IORING_OP_SENDMSG + IORING_OP_SPLICE (Part1)

16 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, 0.0 si, 0.0 st
MiB Mem : 191624.4 total, 177484.7 free, 2931.6 used, 11287.7 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 188883.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
312117	root	20	0	0	0	0	S	17.3	0.0	0:01.26	io_wqe_worker-0
311990	root	20	0	0	0	0	S	11.0	0.0	0:00.40	io_wqe_worker-0
312125	root	20	0	0	0	0	S	6.6	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	457600	24800	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
300304	root	20	0	2986356	108452	54660	S	2.7	0.1	1:38.13	perf
312095	root	20	0	0	0	0	S	2.7	0.0	0:00.40	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	0:00.10	io_wqe_worker-1
312062	root	20	0	0	0	0	S	2.3	0.0	0:00.37	io_wqe_worker-0
312069	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	2.3	0.0	0:00.03	top
300276	root	20	0	62812	5404	3844	S	0.3	0.0	3:57.64	top
310569	root	20	0	0	0	0	I	0.3	0.0	0:00.02	kwoker/61;2-event
311821	root	20	0	0	0	0	I	0.3	0.0	0:00.18	kwoker/u160;2-nl
311830	root	20	0	0	0	0	I	0.3	0.0	0:00.38	kwoker/u160;0-nl
311894	root	20	0	0	0	0	I	0.3	0.0	0:00.42	kwoker/u160;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	0:00.20	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_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kwoker/8;0H-kblck
10	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	kwoker/1;0H-kblck
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	kwoker/2;0H-kblck
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	cpuhp/2

```
Administrator: Windows PowerShell
C:\> issued rwts: total=242105,0,0,0 short=0,0,0 dropped=0,0,0
latency : target=0, window=0, percentile=100,000, depth=16

C:\> Run status group 0 [all jobs]
ID: 0x0: 0x91081B/s (82048B/s), 409081B/s-791081B/s (42958B/s-82948B/s), io=109361B (20336B), run=245120-245120exec
PS C:\Users\Administrator> & "C:\Program Files\Foxit Software\Foxit Reader\Foxit Reader.exe" -group_reporting=1 --name=filo_test --io=io_inetwindowx10 --iodepth=16 -direct
5 1 --io=io_inetwindowx10 --io=io_inetwindowx10 --io=io_inetwindowx10 --io=io_inetwindowx10 --io=io_inetwindowx10 --io=io_inetwindowx10 --io=io_inetwindowx10
FiloTest: (g:0) rwrread, bse(R) 8152XkB-8152XkB, (W) 8152XkB-8152XkB, (T) 8152XkB-8152XkB, io=io_inetwindowx10, io=io_inetwindowx10, io=io_inetwindowx10, io=io_inetwindowx10
1...
1/fo-3.22
Starting 30 threads
Jobs: 30 (f=30) : [R(20)][5.7%][r=8833MIB/s][r=1104 IOPS][eta 04m:45s]
```

Task Manager Performance tab showing Ethernet throughput. The graph shows a peak of 54 Mbps. Statistics: Adapter name: SLOT 4 Port 1, Connection type: Ethernet, IPv4 address: 192.168.0.153, IPv6 address: fe80::5a581155::c44b:719. Send: 73.7 Mbps (73.1 Gbps), Receive: 75.1 Gbps (75.1 Gbps).

smbclient IORING_OP_SENDMSG/SPLICE (network)

4 connections, ~11 GBytes/s, smbld 8.6% cpu, with 4 io_wqework 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 KiloBytes/sec) (average 2746704.9 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3185069.5 KiloBytes/sec) (average 3223967.9 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3180123.7 KiloBytes/sec) (average 3176906.6 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (2824827.2 KiloBytes/sec) (average 2820685.4 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3235901.3 KiloBytes/sec) (average 3224002.5 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (2782680.3 KiloBytes/sec) (average 2746830.3 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3230283.4 KiloBytes/sec) (average 3176965.8 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3215070.2 KiloBytes/sec) (average 3223992.8 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (2790190.4 KiloBytes/sec) (average 2828636.8 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3185069.5 KiloBytes/sec) (average 3176974.6 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (2797813.8 KiloBytes/sec) (average 2746894.5 KiloBytes/sec)
getting file /506.dat of size 2097152000 as /dev/null (3250793.1 KiloBytes/sec) (average 3224021.8 KiloBytes/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: 1024.0 total, 737.0 free, 287.0 used, 131646.0 avail Mem
```

PID	USER	PR	NI	VI	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
740180	root	20	0	0	375680	35968	R	99.3	0.0	0:25.55	smbclient
740185	root	20	0	0	375664	36180	R	99.0	0.0	0:30.87	smbclient
740187	root	20	0	0	375692	35888	R	88.1	0.0	0:44.88	smbclient
740186	root	20	0	0	375652	35896	R	86.4	0.0	0:49.28	smbclient
180190	root	20	0	0	31540	7872	S	2.0	0.0	100:03.15	htop
238	root	20	0	0	0	0	S	1.3	0.0	5:56.39	kssoftirq/45
740176	root	20	0	0	249536	8076	S	1.3	0.0	0:13.20	lftop

```
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, 177240.5 free, 3055.5 used, 11320.1 buff/cache
Mem Swap: 1024.0 total, 1024.0 free, 0.0 used, 100675.2 avail Mem
```

PID	USER	PR	NI	VI	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
310138	root	20	0	0	0	0	S	21.3	0.0	0:52.01	io_wqeworker-0
310133	root	20	0	0	0	0	S	20.3	0.0	0:53.37	io_wqeworker-0
310139	root	20	0	0	0	0	S	17.9	0.0	0:40.39	io_wqeworker-0
310121	root	20	0	0	0	0	S	17.3	0.0	0:34.40	io_wqeworker-0
310116	root	20	0	0	450800	21264	R	17.0	8.6	0:46.53	smbd

Samples: 786 of event 'cycles', 4000 Hz, 17652 count (approx.): 35348326236 lost: 0/0 drop: 0/32090

Overhead	Shared Object	Symbol
7.00%	[kernel]	[k] do_tcp_sendpages
5.97%	[kernel]	[k] raw_spin_lock_bh
4.80%	[kernel]	[k] copy_page_to_iter
3.75%	[kernel]	[k] page_cache_pipe_buf_release
3.35%	[kernel]	[k] ___x86_retpoline_rax
3.08%	[kernel]	[k] page_cache_pipe_buf_confirm
2.97%	[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,dsd

	1546838464eb	3892060928eb	4638091264eb	6184121056eb773015248eb
192.168.10.191		=> 192.168.10.190		91.7Gb 91.5Gb 89.7Gb
		<=		18.3Gb 18.7Gb 19.0Gb
192.168.10.191		=> 192.168.0.153		0b 0b 238b
		<=		0b 0b 218b
TX:	cur: 3146B peak: 0b			rates: 91.7Gb 91.5Gb 89.7Gb
RX:	68.7MB	22.1MB		18.3Gb 18.7Gb 19.0Gb
TOTAL:	3146B	0b		91.8Gb 91.5Gb 89.7Gb

smbclient IORING_OP_SENDMSG/SPLICE (loopback)

8 connections, ~22 GBytes/s, smbdc 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()`

```
getting file %SMB_DAT of size 2097152000 as /dev/null (3075974.6 KiBytes/sec) (average 208800.0 KiBytes/sec) (top - 04:00:50 up 4 days, 23:02, 6 users, load average: 9.15, 3.56, 1.44)
getting file %SMB_DAT of size 2097152000 as /dev/null (2942578.3 KiBytes/sec) (average 2943679.6 KiBytes/sec) Tasks: 937 total, 14 running, 903 sleeping, 0 stopped, 0 zombie
getting file %SMB_DAT of size 2097152000 as /dev/null (2717077.8 KiBytes/sec) (average 2641657.3 KiBytes/sec) (cpu(s): 0.3 us, 11.2 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.2 hi, 2.1 si, 0.0 st
getting file %SMB_DAT of size 2097152000 as /dev/null (2951000.2 KiBytes/sec) (average 2879437.6 KiBytes/sec) (MiB Mem : 191624.1 total, 176925.4 free, 3316.7 used, 11382.0 buff/cache)
getting file %SMB_DAT of size 2097152000 as /dev/null (2081641.2 KiBytes/sec) (average 2739176.8 KiBytes/sec) (MiB Swap : 1024.0 total, 1024.0 free, 0.0 used, 100483.7 avail Mem)
getting file %SMB_DAT of size 2097152000 as /dev/null (3107730.5 KiBytes/sec) (average 2958064.5 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (26049736.0 KiBytes/sec) (average 2774192.3 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2808334.8 KiBytes/sec) (average 2730460.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3117130.9 KiBytes/sec) (average 2892662.1 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3047618.0 KiBytes/sec) (average 2944350.1 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3089335.4 KiBytes/sec) (average 2741473.6 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2741632.0 KiBytes/sec) (average 2649012.6 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3082932.1 KiBytes/sec) (average 2888254.5 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3126717.1 KiBytes/sec) (average 2855135.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3080989.0 KiBytes/sec) (average 2891506.4 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2531870.2 KiBytes/sec) (average 2732406.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2127371.9 KiBytes/sec) (average 2098206.4 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2921540.2 KiBytes/sec) (average 2844283.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3083655.1 KiBytes/sec) (average 2743720.7 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3093655.1 KiBytes/sec) (average 2842525.3 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3087341.7 KiBytes/sec) (average 2881080.4 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3107730.5 KiBytes/sec) (average 2868079.4 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3136293.8 KiBytes/sec) (average 2893072.3 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2726267.0 KiBytes/sec) (average 2710130.3 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3084336.0 KiBytes/sec) (average 2945895.5 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2749580.0 KiBytes/sec) (average 2799462.2 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3117130.9 KiBytes/sec) (average 2746070.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3117130.9 KiBytes/sec) (average 2844252.7 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2563283.7 KiBytes/sec) (average 2828659.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2519664.9 KiBytes/sec) (average 2564661.4 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3083655.1 KiBytes/sec) (average 2894340.3 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2973320.0 KiBytes/sec) (average 2732566.5 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2773312.2 KiBytes/sec) (average 2788079.3 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3131480.8 KiBytes/sec) (average 2846041.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3131480.8 KiBytes/sec) (average 2748740.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2595800.4 KiBytes/sec) (average 2842472.7 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3083655.2 KiBytes/sec) (average 2857370.6 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2876743.8 KiBytes/sec) (average 2878380.8 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (3083655.2 KiBytes/sec) (average 2895262.7 KiBytes/sec)
getting file %SMB_DAT of size 2097152000 as /dev/null (2824827.2 KiBytes/sec) (average 2731919.6 KiBytes/sec)
[...]
```

```

Samples: 30M of event 'cycles', 1000 Hz, Event count (approx.): 52678599529 Lost: 0/0 drop: 0/0
Overhead Shared object
 63.14% [kernel] [k] copy_user_enhanced_fast_string
 6.40% [kernel] [k] native_queued_spin_lock_slowpath
 3.30% [kernel] [k] tcpackit_rcv
 1.70% [kernel] [k] do_tcp_sendpages
 1.20% [kernel] [k] raw_spin_lock_bh
 1.02% [kernel] [k] prb_fill_curr_block_isra.0
 1.01% [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
[...]
```

```

1575379206b 3151075040b 4726614016b 6382151600b87776093460
127.0.0.1 => 127.0.0.1 1816b 1816b 1806b
<=
TX: cum: 226426b peak: 6.596b rates: 1816b 1816b 1806b
RX: 0b 0b 0b
TOTAL: 226426b 6.596b 1816b 1816b 1806b
```

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_SENDDMSG/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

The road to upstream (TEVENT_FD_ERROR)

- ▶ 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 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 via io_uring
 - ▶ use it in smb2_server.c
 - ▶ In future use it in other performance critical places too.

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 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 one copy instead of two:
 - ▶ IORING_OP_SENDMSG_Z 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 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`
- ▶ 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 it available without changing the whole stack

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 it if available without changing the whole stack

Questions? Feedback!

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

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