STORAGE DEVELOPER CONFERENCE

SDC 23 | FREMONT MARRIOTT SILICON VALLEY SEPTEMBER 18-21, 2023

BY Developers FOR Developers

io_uring

Status Update within Samba

Stefan Metzmacher `<metze@samba.org>`

Samba Team / SerNet

2023-09-20

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

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 2020/2021 - SambaXP 2023)

- 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)

Last Status Updates (SDC 2020/2021 - SambaXP 2023)

- 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)

Last Status Updates (SDC 2020/2021 - SambaXP 2023)

- 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)
- 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)

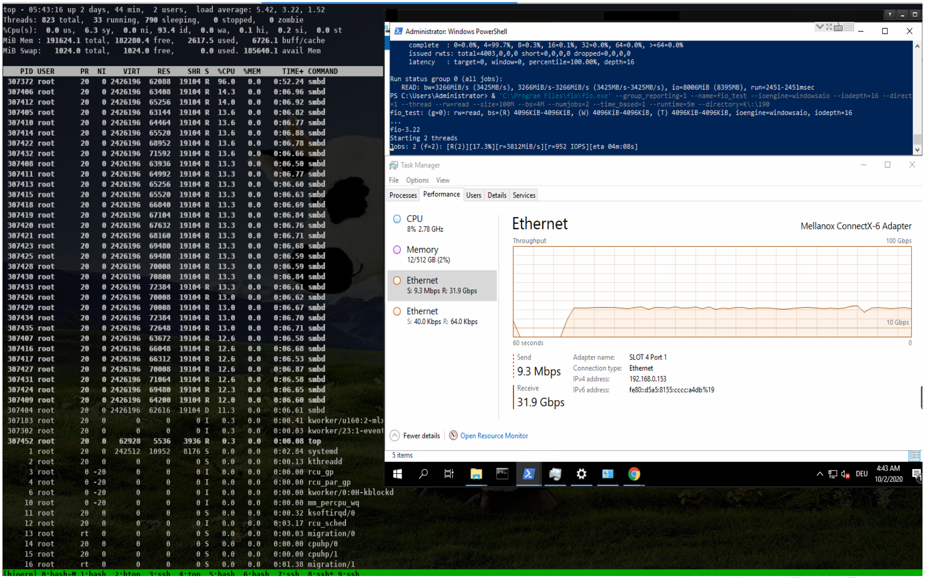
- In October 2020 I was able to do some performance research
  - With 100GBit/s interfaces and two NUMA nodes per server.
- At that time I focussed on the SMB2 Read performance only
  - We had limited time on the given hardware
  - We mainly tested with fio.exe on a Windows client
  - Linux kernel 5.8.12 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 msecs



```
top - 05:43:16 up 2 days, 44 min,  2 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
MiB Mem : 191624.1 total, 182280.4 free,   2617.5 used,   6726.1 buff/cache
MiB 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 smbd
 307406 root      20   0 2426196  63408  19104 R  14.3   0.0   0:06.96 smbd
 307412 root      20   0 2426196  65256  19104 R  14.0   0.0   0:06.92 smbd
 307405 root      20   0 2426196  63144  19104 R  13.6   0.0   0:06.82 smbd
 307410 root      20   0 2426196  64464  19104 R  13.6   0.0   0:06.77 smbd
 307414 root      20   0 2426196  65520  19104 R  13.6   0.0   0:06.88 smbd
 307422 root      20   0 2426196  68952  19104 R  13.6   0.0   0:06.78 smbd
 307432 root      20   0 2426196  71592  19104 R  13.6   0.0   0:06.66 smbd
 307408 root      20   0 2426196  63936  19104 R  13.3   0.0   0:06.50 smbd
 307411 root      20   0 2426196  64992  19104 R  13.3   0.0   0:06.77 smbd
 307413 root      20   0 2426196  65256  19104 R  13.3   0.0   0:06.60 smbd
 307415 root      20   0 2426196  65520  19104 R  13.3   0.0   0:06.63 smbd
 307418 root      20   0 2426196  66840  19104 R  13.3   0.0   0:06.69 smbd
 307419 root      20   0 2426196  67104  19104 R  13.3   0.0   0:06.84 smbd
 307420 root      20   0 2426196  67632  19104 R  13.3   0.0   0:06.76 smbd
 307421 root      20   0 2426196  68160  19104 R  13.3   0.0   0:06.71 smbd
 307423 root      20   0 2426196  69480  19104 R  13.3   0.0   0:06.68 smbd
 307425 root      20   0 2426196  69480  19104 R  13.3   0.0   0:06.59 smbd
 307428 root      20   0 2426196  70008  19104 R  13.3   0.0   0:06.59 smbd
 307430 root      20   0 2426196  70800  19104 R  13.3   0.0   0:06.84 smbd
 307433 root      20   0 2426196  72384  19104 R  13.3   0.0   0:06.61 smbd
 307426 root      20   0 2426196  70008  19104 R  13.0   0.0   0:06.62 smbd
 307429 root      20   0 2426196  70008  19104 R  13.0   0.0   0:06.67 smbd
 307434 root      20   0 2426196  72384  19104 R  13.0   0.0   0:06.70 smbd
 307435 root      20   0 2426196  72648  19104 R  13.0   0.0   0:06.71 smbd
 307407 root      20   0 2426196  63672  19104 R  12.6   0.0   0:06.58 smbd
 307416 root      20   0 2426196  66048  19104 R  12.6   0.0   0:06.68 smbd
 307417 root      20   0 2426196  66312  19104 R  12.6   0.0   0:06.53 smbd
 307427 root      20   0 2426196  70008  19104 R  12.6   0.0   0:06.87 smbd
 307431 root      20   0 2426196  71064  19104 R  12.6   0.0   0:06.58 smbd
 307424 root      20   0 2426196  69480  19104 R  12.3   0.0   0:06.65 smbd
 307409 root      20   0 2426196  64200  19104 R  12.0   0.0   0:06.60 smbd
 307404 root      20   0 2426196  62616  19104 D  11.3   0.0   0:06.61 smbd
 307183 root      20   0       0      0      0 I   0.3   0.0   0:00.41 kworker/u160:2-mlx
 307302 root      20   0       0      0      0 I   0.3   0.0   0:00.03 kworker/23:1-event
 307452 root      20   0   62928   5536   3936 R   0.3   0.0   0:00.08 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-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
[bjoern] 0:bash-M 1:bash  2:htop  3:ssh  4:top  5:bash  6:bash  7:ssh  8:ssh* 9:ssh
```

Administrator: Windows PowerShell

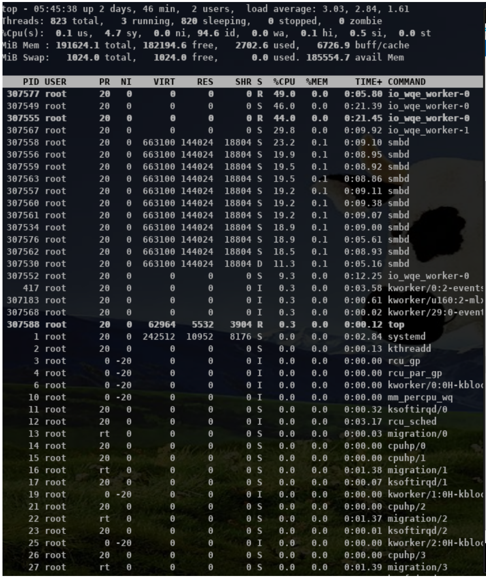
```
     complete  : 0=0.0%, 4=99.7%, 8=0.3%, 16=0.1%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=4003,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=16

Run status group 0 (all jobs):
   READ: bw=3266MiB/s (3425MB/s), 3266MiB/s-3266MiB/s (3425MB/s-3425MB/s), io=8006MiB (8395MB), run=2451-2451msec
PS C:\Users\Administrator> & 'C:\Program Files\fio\fio.exe' --group_reporting=1 --name=fio_test --ioengine=windowsaio --iodepth=16 --direct
=1 --thread --rw=read --size=100M --bs=4M --numjobs=2 --time_based=1 --runtime=5m --directory=K\:\190
fio_test: (g=0): rw=read, bs=(R) 4096KiB-4096KiB, (W) 4096KiB-4096KiB, (T) 4096KiB-4096KiB, ioengine=windowsaio, iodepth=16
...
fio-3.22
Starting 2 threads
Jobs: 2 (f=2): [R(2)][17.3%][r=3812MiB/s][r=952 IOPS][eta 04m:08s]
```

Send
9.3 Mbps
Receive
31.9 Gbps

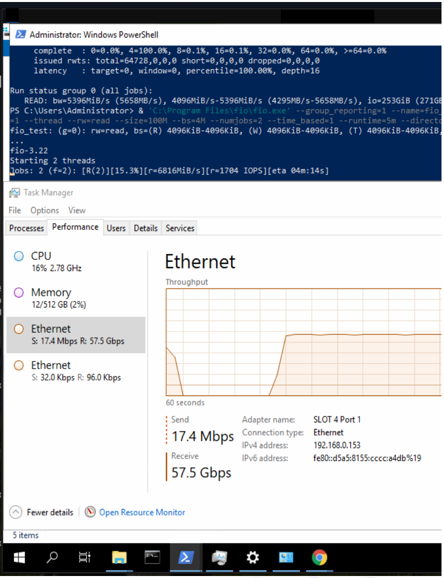
Adapter name: SLOT 4 Port 1
Connection type: Ethernet
IPv4 address: 192.168.0.153
IPv6 address: fe80::d5a5:8155:cccc:a4db%19

IORING_OP_SENDMSG (Part1)

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



```
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,  0.5 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   0:05.80 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   0:09.38 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_wqe_worker-0
    417 root      20   0       0      0      0 I   0.3   0.0   0:03.58 kworker/0:2-events
 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-kbloc
     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
     17 root      20   0       0      0      0 S   0.0   0.0   0:00.07 ksoftirqd/1
     19 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-kbloc
     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 ksoftirqd/2
     25 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/2:0H-kbloc
     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: bw=5396MiB/s (5658MB/s), 4096MiB/s-5396MiB/s (4295MB/s-5658MB/s), io=253GiB (271GB
PS C:\Users\Administrator> & 'C:\Program Files\fio\fio.exe' --group_reporting=1 --name=fio_
=1 --thread --rw=read --size=100M --bs=4M --numjobs=2 --time_based=1 --runtime=5m --directo
fio_test: (g=0): rw=read, bs=(R) 4096KiB-4096KiB, (W) 4096KiB-4096KiB, (T) 4096KiB-4096KiB,
...
fio-3.22
Starting 2 threads
Jobs: 2 (f=2): [R(2)][15.3%][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

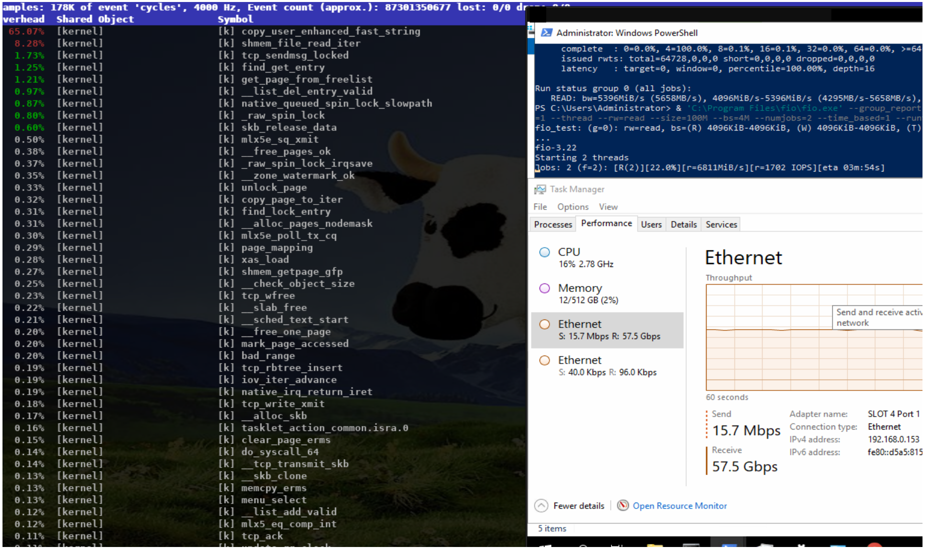
Adapter name:	SLOT 4 Port 1
Connection type:	Ethernet
IPv4 address:	192.168.0.153
IPv6 address:	fe80::d5a5:8155:cccc:a4db%19

Fewer details | Open Resource Monitor

5 items

IORING_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 dr
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_rbtree_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 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: 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 --rw=read --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]
```

CPU
16% 2.78 GHz

Memory
12/512 GB (2%)

Ethernet
S: 15.7 Mbps R: 57.5 Gbps

Ethernet
S: 40.0 Kbps R: 96.0 Kbps

Ethernet

Throughput

Send and receive activ
network

60 seconds

Send
15.7 Mbps

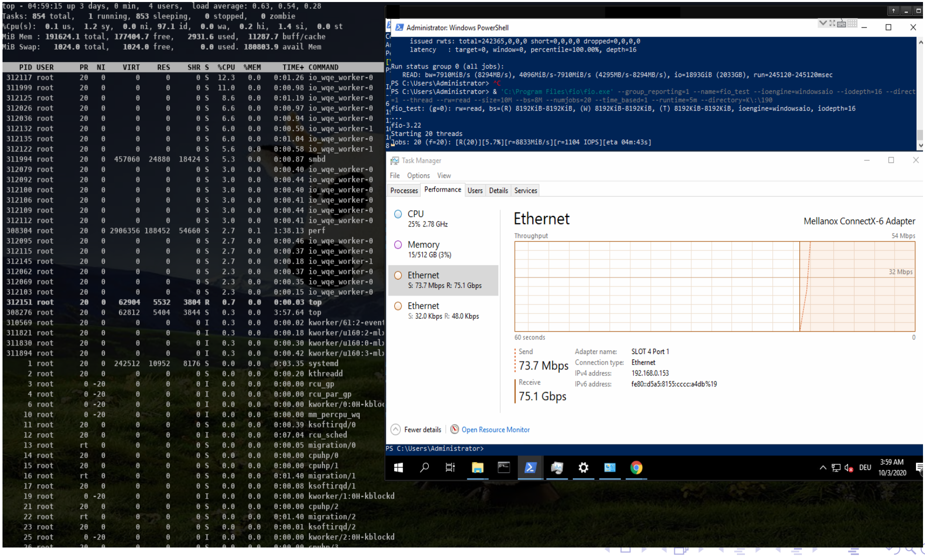
Receive
57.5 Gbps

Adapter name:	SLOT 4 Port 1
Connection type:	Ethernet
IPv4 address:	192.168.0.153
IPv6 address:	fe80::d5a5:815

IORING_OP_SENDMSG + IORING_OP_SPLICE (Part1)

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

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



```
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, 177404.7 free,   2931.6 used,  11287.7 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used. 180803.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 312117 root      20   0       0      0      0 S  12.3   0.0   0:01.26 io_wqe_worker-0
 311999 root      20   0       0      0      0 S  11.0   0.0   0:00.98 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   0:00.87 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 2906356 188452  54660 S   2.7   0.1   1:38.13 perf
 312095 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
 312069 root      20   0       0      0      0 S   2.3   0.0   0:00.35 io_wqe_worker-0
 312103 root      20   0       0      0      0 S   2.3   0.0   0:00.15 io_wqe_worker-0
 312151 root      20   0   62904   5532   3804 R   0.7   0.0   0:00.03 top
 308276 root      20   0   62812   5404   3844 S   0.3   0.0   3:57.64 top
 310569 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/u160:2-mlx
 311830 root      20   0       0      0      0 I   0.3   0.0   0:00.30 kworker/u160:0-mlx
 311894 root      20   0       0      0      0 I   0.3   0.0   0:00.42 kworker/u160:3-mlx
      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 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-kbloc
     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   0:00.39 ksoftirqd/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   0:00.08 ksoftirqd/1
     19 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-kblockd
     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 ksoftirqd/2
     25 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/2:0H-kblockd
     26 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/3
```

Administrator: Windows PowerShell

```
    issued rwts: total=242365,0,0,0 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: bw=7910MiB/s (8294MB/s), 4096MiB/s-7910MiB/s (4295MB/s-8294MB/s), io=1893GiB (2033GB), run=245120-245120msec
PS C:\Users\Administrator> ^C
PS C:\Users\Administrator> & 'C:\Program Files\fio\fio.exe' --group_reporting=1 --name=fio_test --ioengine=windowsaio --iodepth=16 --direct
=1 --thread --rw=read --size=10M --bs=8M --numjobs=20 --time_based=1 --runtime=5m --directory=K\:\190
fio_test: (g=0): rw=read, bs=(R) 8192KiB-8192KiB, (W) 8192KiB-8192KiB, (T) 8192KiB-8192KiB, ioengine=windowsaio, iodepth=16
...
fio-3.22
Starting 20 threads
Jobs: 20 (f=20): [R(20)][5.7%][r=8833MiB/s][r=1104 IOPS][eta 04m:43s]
```

Task Manager

File Options View

Processes | Performance | Users | Details | Services

CPU
25% 2.78 GHz

Memory
15/512 GB (3%)

Ethernet
S: 73.7 Mbps R: 75.1 Gbps

Ethernet
S: 32.0 Kbps R: 48.0 Kbps

Ethernet — Mellanox ConnectX-6 Adapter

Send	Adapter name:	SLOT 4 Port 1
73.7 Mbps	Connection type:	Ethernet
Receive	IPv4 address:	192.168.0.153
75.1 Gbps	IPv6 address:	fe80::d5a5:8155:cccc:a4db%19

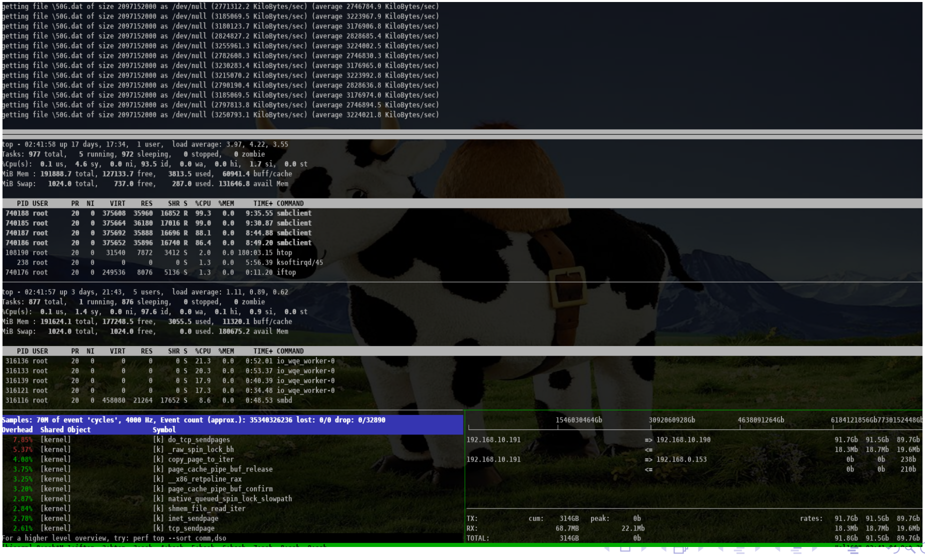
Fewer details | Open Resource Monitor

```
PS C:\Users\Administrator>
```

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 \50G.dat of size 2097152000 as /dev/null (2771312.2 KiloBytes/sec) (average 2746784.9 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3185069.5 KiloBytes/sec) (average 3223967.9 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3180123.7 KiloBytes/sec) (average 3176906.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2824827.2 KiloBytes/sec) (average 2828685.4 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3255961.3 KiloBytes/sec) (average 3224002.5 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2782608.3 KiloBytes/sec) (average 2746830.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3230283.4 KiloBytes/sec) (average 3176965.0 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3215070.2 KiloBytes/sec) (average 3223992.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2790190.4 KiloBytes/sec) (average 2828636.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3185069.5 KiloBytes/sec) (average 3176974.0 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2797813.8 KiloBytes/sec) (average 2746894.5 KiloBytes/sec)
getting file \50G.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: 3.97, 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
MiB Mem : 191888.7 total, 127133.7 free,   3813.5 used,  60941.4 buff/cache
MiB Swap:   1024.0 total,    737.0 free,    287.0 used. 131646.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 740188 root      20   0  375608  35960  16852 R  99.3   0.0   9:35.55 smbclient
 740185 root      20   0  375664  36180  17016 R  99.0   0.0   9:30.87 smbclient
 740187 root      20   0  375692  35888  16696 R  88.1   0.0   8:44.88 smbclient
 740186 root      20   0  375652  35896  16740 R  86.4   0.0   8:49.20 smbclient
 108190 root      20   0   31540   7872   3412 S   2.0   0.0 180:03.15 htop
    238 root      20   0       0      0      0 S   1.3   0.0   5:56.39 ksoftirqd/45
 740176 root      20   0  249536   8076   5136 S   1.3   0.0   0:11.20 iftop
```

```
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
MiB Mem : 191624.1 total, 177248.5 free,   3055.5 used,  11320.1 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used. 180675.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 316136 root      20   0       0      0      0 S  21.3   0.0   0:52.01 io_wqe_worker-0
 316133 root      20   0       0      0      0 S  20.3   0.0   0:53.37 io_wqe_worker-0
 316139 root      20   0       0      0      0 S  17.9   0.0   0:40.39 io_wqe_worker-0
 316121 root      20   0       0      0      0 S  17.3   0.0   0:34.48 io_wqe_worker-0
 316116 root      20   0  458080  21264  17652 S   8.6   0.0   0:48.53 smbd
```

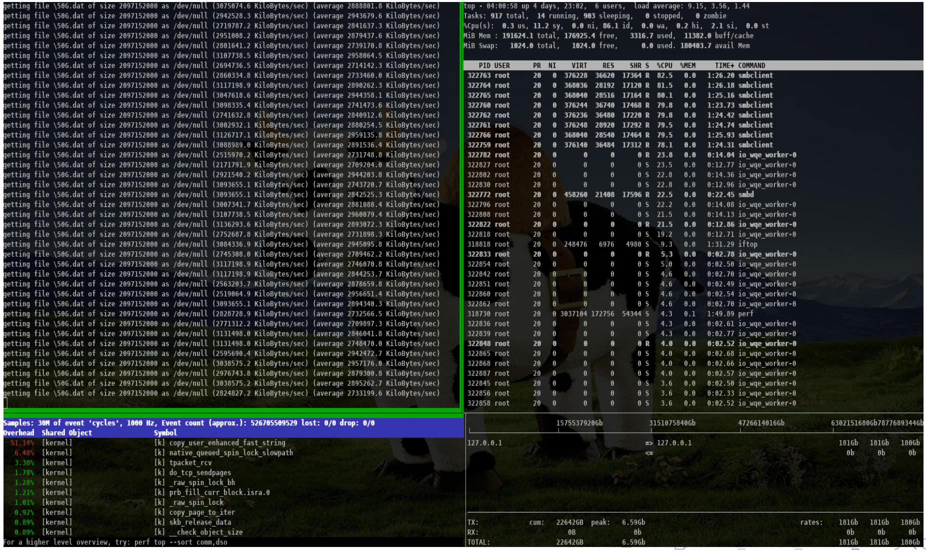
```
Samples: 70M of event 'cycles', 4000 Hz, Event count (approx.): 35340326236 lost: 0/0 drop: 0/32890
Overhead  Shared Object                Symbol
   7.85%  [kernel]                     [k] do_tcp_sendpages
   5.37%  [kernel]                     [k] _raw_spin_lock_bh
   4.08%  [kernel]                     [k] copy_page_to_iter
   3.75%  [kernel]                     [k] page_cache_pipe_buf_release
   3.25%  [kernel]                     [k] __x86_retpoline_rax
   3.20%  [kernel]                     [k] page_cache_pipe_buf_confirm
   2.87%  [kernel]                     [k] native_queued_spin_lock_slowpath
   2.84%  [kernel]                     [k] shmem_file_read_iter
   2.78%  [kernel]                     [k] inet_sendpage
   2.61%  [kernel]                     [k] tcp_sendpage
For a higher level overview, try: perf top --sort comm,dso
```

```
                1546030464Gb          3092060928Gb          4638091264Gb          6184121856Gb7730152448Gb
192.168.10.191                        => 192.168.10.190                         91.7Gb  91.5Gb  89.7Gb
                                      <=                                        18.3Mb  18.7Mb  19.6Mb
192.168.10.191                        => 192.168.0.153                             0b      0b    238b
                                      <=                                           0b      0b    210b

TX:             cum:   314GB   peak:      0b                       rates:   91.7Gb  91.5Gb  89.7Gb
RX:                   68.7MB          22.1Mb                                18.3Mb  18.7Mb  19.6Mb
TOTAL:                 314GB              0b                                91.8Gb  91.5Gb  89.7Gb
```

smbclient IORING_OP_SENDMSG/SPLICE (loopback)

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

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



```
getting file \50G.dat of size 2097152000 as /dev/null (3075074.6 KiloBytes/sec) (average 2888801.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2942528.3 KiloBytes/sec) (average 2943679.6 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2719787.2 KiloBytes/sec) (average 2841637.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2951008.2 KiloBytes/sec) (average 2879437.6 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2801641.2 KiloBytes/sec) (average 2739170.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3107738.5 KiloBytes/sec) (average 2958064.5 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2694736.5 KiloBytes/sec) (average 2714142.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2860334.8 KiloBytes/sec) (average 2733460.0 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3117198.9 KiloBytes/sec) (average 2890262.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3047618.6 KiloBytes/sec) (average 2944358.1 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3098335.4 KiloBytes/sec) (average 2741473.6 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2741632.8 KiloBytes/sec) (average 2840912.6 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3002932.1 KiloBytes/sec) (average 2880254.5 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3126717.1 KiloBytes/sec) (average 2959135.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3088989.0 KiloBytes/sec) (average 2891536.4 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2515970.2 KiloBytes/sec) (average 2731748.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2171791.9 KiloBytes/sec) (average 2709204.0 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2921540.2 KiloBytes/sec) (average 2944203.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3093655.1 KiloBytes/sec) (average 2743720.7 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3093655.1 KiloBytes/sec) (average 2842525.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3007341.7 KiloBytes/sec) (average 2881088.4 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3107738.5 KiloBytes/sec) (average 2960079.4 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3136293.6 KiloBytes/sec) (average 2893072.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2752687.8 KiloBytes/sec) (average 2731898.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3084336.9 KiloBytes/sec) (average 2945095.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2745308.0 KiloBytes/sec) (average 2709462.2 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3117198.9 KiloBytes/sec) (average 2746070.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3117198.9 KiloBytes/sec) (average 2844253.7 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2563203.7 KiloBytes/sec) (average 2878659.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2519064.9 KiloBytes/sec) (average 2956651.4 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3093655.1 KiloBytes/sec) (average 2894340.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2828728.9 KiloBytes/sec) (average 2732566.5 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2771312.2 KiloBytes/sec) (average 2709897.3 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3131498.0 KiloBytes/sec) (average 2846041.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3131498.0 KiloBytes/sec) (average 2748470.0 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2595690.4 KiloBytes/sec) (average 2942472.7 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3038575.2 KiloBytes/sec) (average 2957176.0 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2976743.8 KiloBytes/sec) (average 2879300.8 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (3038575.2 KiloBytes/sec) (average 2895262.7 KiloBytes/sec)
getting file \50G.dat of size 2097152000 as /dev/null (2824827.2 KiloBytes/sec) (average 2733199.6 KiloBytes/sec)
```

```
top - 04:00:58 up 4 days, 23:02,  6 users,  load average: 9.15, 3.56, 1.44
Tasks: 917 total,  14 running, 903 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us, 11.2 sy,  0.0 ni, 86.1 id,  0.0 wa,  0.2 hi,  2.1 si,  0.0 st
MiB Mem : 191624.1 total, 176925.4 free,   3316.7 used,  11382.0 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used. 180403.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 322763 root      20   0  376228  36620  17364 R  82.5   0.0   1:26.20 smbclient
 322764 root      20   0  368036  28192  17120 R  81.5   0.0   1:26.18 smbclient
 322765 root      20   0  368040  28516  17164 R  80.1   0.0   1:25.16 smbclient
 322760 root      20   0  376244  36740  17468 R  79.8   0.0   1:23.73 smbclient
 322762 root      20   0  376236  36480  17220 R  79.8   0.0   1:24.42 smbclient
 322761 root      20   0  376248  28920  17292 R  79.5   0.0   1:24.74 smbclient
 322766 root      20   0  368040  28540  17464 R  79.5   0.0   1:25.93 smbclient
 322759 root      20   0  376140  36484  17312 R  78.1   0.0   1:24.31 smbclient
 322782 root      20   0       0      0      0 R  23.8   0.0   0:14.04 io_wqe_worker-0
 322827 root      20   0       0      0      0 S  23.5   0.0   0:12.77 io_wqe_worker-0
 322802 root      20   0       0      0      0 S  22.8   0.0   0:14.36 io_wqe_worker-0
 322830 root      20   0       0      0      0 S  22.8   0.0   0:12.96 io_wqe_worker-0
 322772 root      20   0  458260  21408  17596 R  22.5   0.0   0:22.45 smbd
 322796 root      20   0       0      0      0 S  22.2   0.0   0:14.08 io_wqe_worker-0
 322808 root      20   0       0      0      0 S  21.5   0.0   0:14.13 io_wqe_worker-0
 322822 root      20   0       0      0      0 R  21.5   0.0   0:12.86 io_wqe_worker-0
 322818 root      20   0       0      0      0 S  19.2   0.0   0:12.71 io_wqe_worker-0
 318818 root      20   0  248476   6976   4980 S   9.3   0.0   1:31.29 iftop
 322833 root      20   0       0      0      0 R   5.3   0.0   0:02.78 io_wqe_worker-0
 322854 root      20   0       0      0      0 S   5.0   0.0   0:02.50 io_wqe_worker-0
 322842 root      20   0       0      0      0 S   4.6   0.0   0:02.70 io_wqe_worker-0
 322851 root      20   0       0      0      0 S   4.6   0.0   0:02.49 io_wqe_worker-0
 322860 root      20   0       0      0      0 S   4.6   0.0   0:02.54 io_wqe_worker-0
 322862 root      20   0       0      0      0 S   4.6   0.0   0:02.70 io_wqe_worker-0
 318730 root      20   0 3037104 172756  54344 S   4.3   0.1   1:49.89 perf
 322836 root      20   0       0      0      0 S   4.3   0.0   0:02.61 io_wqe_worker-0
 322839 root      20   0       0      0      0 S   4.3   0.0   0:02.77 io_wqe_worker-0
 322848 root      20   0       0      0      0 R   4.0   0.0   0:02.52 io_wqe_worker-0
 322865 root      20   0       0      0      0 S   4.0   0.0   0:02.68 io_wqe_worker-0
 322868 root      20   0       0      0      0 S   4.0   0.0   0:02.66 io_wqe_worker-0
 322887 root      20   0       0      0      0 S   4.0   0.0   0:02.57 io_wqe_worker-0
 322845 root      20   0       0      0      0 S   3.6   0.0   0:02.50 io_wqe_worker-0
 322856 root      20   0       0      0      0 S   3.6   0.0   0:02.33 io_wqe_worker-0
 322858 root      20   0       0      0      0 S   3.6   0.0   0:02.52 io_wqe_worker-0
```

```
Samples: 30M of event 'cycles', 1000 Hz, Event count (approx.): 526705509529 lost: 0/0 drop: 0/0
Overhead  Shared Object              Symbol
  51.14%  [kernel]                   [k] copy_user_enhanced_fast_string
   6.48%  [kernel]                   [k] native_queued_spin_lock_slowpath
   3.38%  [kernel]                   [k] tpacket_rcv
   1.78%  [kernel]                   [k] do_tcp_sendpages
   1.28%  [kernel]                   [k] _raw_spin_lock_bh
   1.21%  [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
For a higher level overview, try: perf top --sort comm,dso
```

```
                 1575537920Gb            3151075840Gb            4726614016Gb            6302151680Gb7877689344Gb
└─────────────────────┴───────────────────────┴───────────────────────┴───────────────────────┴──────────
127.0.0.1                               => 127.0.0.1                                     181Gb   181Gb   180Gb
                                        <=                                                0b      0b      0b

TX:              cum:   22642GB   peak:   6.59Gb                              rates:    181Gb   181Gb   180Gb
RX:                         0B              0b                                            0b      0b      0b
TOTAL:                  22642GB           6.59Gb                                        181Gb   181Gb   180Gb
```

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

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 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 (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 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
- With IORING_OP_SENDMSG_ZC only 1 one copy is used:
  - It is able to avoid copying to the socket
  - We get an extra completion once the buffers are not needed anymore
  - Only with real hardware, not on loopback in an upstream kernel
  - A custom kernel loopback gives ~7.5 GBytes/s instead of ~3.5 GBytes/s
  - With a noop vfs module, we get ~18 GBytes/s instead of ~6 GBytes/s

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
- With IORING_OP_SENDMSG_ZC only 1 one copy is used:
  - It is able to avoid copying to the socket
  - We get an extra completion once the buffers are not needed anymore
  - Only with real hardware, not on loopback in an upstream kernel
  - A custom kernel loopback gives ~7.5 GBytes/s instead of ~3.5 GBytes/s
  - With a noop vfs module, we get ~18 GBytes/s instead of ~6 GBytes/s

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
- With IORING_OP_SENDMSG_ZC only 1 one copy is used:
  - It is able to avoid copying to the socket
  - We get an extra completion once the buffers are not needed anymore
  - Only with real hardware, not on loopback in an upstream kernel
  - A custom kernel loopback gives ~7.5 GBytes/s instead of ~3.5 GBytes/s
  - With a noop vfs module, we get ~18 GBytes/s instead of ~6 GBytes/s

Future Improvements

- Patches are slowly getting prepared for master
  - Some preparations are already in or pending merge requests
  - We even have basic automated ci testing in place now
  - But changes need to be checked for performance regressions
- 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

Future Improvements

- Patches are slowly getting prepared for master
  - Some preparations are already in or pending merge requests
  - We even have basic automated ci testing in place now
  - But changes need to be checked for performance regressions
- 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!

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

→ SerNet/SAMBA+ sponsor booth

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