

# SAMBA

## EXPERIENCE

# multichannel / io\_uring

Status Update within Samba

Stefan Metzmacher <metze@samba.org>

Samba Team / SerNet

2021-05-05

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

# Check for Updates

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

(draft)

# Topics

- ▶ What is SMB3 Multichannel?
- ▶ Updates in Samba 4.15
- ▶ What is io-uring?
- ▶ io-uring for Samba
- ▶ Performance research, prototyping and ideas
- ▶ Questions? Feedback!

# What is SMB3 Multichannel? (Part 1)

- ▶ Multiple transport connections are bound to one logical connection
  - ▶ This allows using more than one network link
    - ▶ Good for performance
    - ▶ Good for availability reasons
  - ▶ Non TCP transports like RDMA (InfiniBand, RoCE, iWarp)
- ▶ All transport connections (channels) share the same ClientGUID
  - ▶ This is important for Samba
- ▶ An authenticated binding is done at the user session layer
  - ▶ SessionID, TreeID and FileID values are valid on all channels
- ▶ Available network interfaces are auto-negotiated
  - ▶ FSCTL\_QUERY\_NETWORK\_INTERFACE\_INFO interface list
  - ▶ IP (v4 or v6) addresses are returned together with:
    - ▶ Interface index (which addresses belong to the same hardware)
    - ▶ Link speed
    - ▶ RSS and RDMA capabilities

# What is SMB3 Multichannel? (Part 1)

- ▶ Multiple transport connections are bound to one logical connection
  - ▶ This allows using more than one network link
    - ▶ Good for performance
    - ▶ Good for availability reasons
  - ▶ Non TCP transports like RDMA (InfiniBand, RoCE, iWarp)
- ▶ All transport connections (channels) share the same ClientGUID
  - ▶ This is important for Samba
- ▶ An authenticated binding is done at the user session layer
  - ▶ SessionID, TreeID and FileID values are valid on all channels
- ▶ Available network interfaces are auto-negotiated
  - ▶ FSCTL\_QUERY\_NETWORK\_INTERFACE\_INFO interface list
  - ▶ IP (v4 or v6) addresses are returned together with:
    - ▶ Interface Index (which addresses belong to the same hardware)
    - ▶ Link speed
    - ▶ RSS and RDMA capabilities

# What is SMB3 Multichannel? (Part 1)

- ▶ Multiple transport connections are bound to one logical connection
  - ▶ This allows using more than one network link
    - ▶ Good for performance
    - ▶ Good for availability reasons
  - ▶ Non TCP transports like RDMA (InfiniBand, RoCE, iWarp)
- ▶ All transport connections (channels) share the same ClientGUID
  - ▶ This is important for Samba
- ▶ An authenticated binding is done at the user session layer
  - ▶ SessionID, TreeID and FileID values are valid on all channels
- ▶ Available network interfaces are auto-negotiated
  - ▶ FSCTL\_QUERY\_NETWORK\_INTERFACE\_INFO interface list
  - ▶ IP (v4 or v6) addresses are returned together with:
    - ▶ Interface Index (which addresses belong to the same hardware)
    - ▶ Link speed
    - ▶ RSS and RDMA capabilities

# What is SMB3 Multichannel? (Part 1)

- ▶ Multiple transport connections are bound to one logical connection
  - ▶ This allows using more than one network link
    - ▶ Good for performance
    - ▶ Good for availability reasons
  - ▶ Non TCP transports like RDMA (InfiniBand, RoCE, iWarp)
- ▶ All transport connections (channels) share the same ClientGUID
  - ▶ This is important for Samba
- ▶ An authenticated binding is done at the user session layer
  - ▶ SessionID, TreeID and FileID values are valid on all channels
- ▶ Available network interfaces are auto-negotiated
  - ▶ FSCTL\_QUERY\_NETWORK\_INTERFACE\_INFO interface list
  - ▶ IP (v4 or v6) addresses are returned together with:
    - ▶ Interface Index (which addresses belong to the same hardware)
    - ▶ Link speed
    - ▶ RSS and RDMA capabilities

# What is SMB3 Multichannel? (Part 2)

- ▶ IO ordering is important for multichannel
  - ▶ Requests can get lost between client and server
  - ▶ Responses can get lost between server and client
  - ▶ The client isn't able to know the difference
  - ▶ Replays contain the REPLAY flag in the SMB2 header
  - ▶ FILE\_NOT\_AVAILABLE indicates "please retry" to the client
    - ▶ Windows returns ACCESS\_DENIED in some cases instead
    - ▶ In other cases Windows ignores a replay and deadlocks the client
    - ▶ I need to discuss this with Microsoft
    - ▶ See: Samba Bug #14449
- ▶ State changing operations need replay detection
  - ▶ They need to execute only-once
  - ▶ SMB2 Create uses a CreateGUID
  - ▶ SMB2 Lock uses an array with sequence numbers
    - ▶ Windows only supports this on resilient and persistent handles
    - ▶ Future Windows versions are supposed to fix that



# What is SMB3 Multichannel? (Part 2)

- ▶ IO ordering is important for multichannel
  - ▶ Requests can get lost between client and server
  - ▶ Responses can get lost between server and client
  - ▶ The client isn't able to know the difference
  - ▶ Replays contain the REPLAY flag in the SMB2 header
  - ▶ FILE\_NOT\_AVAILABLE indicates "please retry" to the client
    - ▶ Windows returns ACCESS\_DENIED in some cases instead
    - ▶ In other cases Windows ignores a replay and deadlocks the client
    - ▶ I need to discuss this with Microsoft
    - ▶ See: Samba Bug #14449
- ▶ State changing operations need replay detection
  - ▶ They need to execute only-once
  - ▶ SMB2 Create uses a CreateGUID
  - ▶ SMB2 Lock uses an array with sequence numbers
    - ▶ Windows only supports this on resilient and persistent handles
    - ▶ Future Windows versions are supposed to fix that

# What is SMB3 Multichannel? (Part 3)

- ▶ Write/Set operations only need a barrier
  - ▶ An epoch number is incremented on each channel failure
  - ▶ The current epoch number is part of each request
  - ▶ The server remembers the last seen epoch number
  - ▶ Non-REPLAY requests with stale epoch fail
  - ▶ REPLAY requests fail, when there are pending older epoch numbers
- ▶ Read/Get operations can be replayed safely
- ▶ Lease/Oplock break notifications should be retried
  - ▶ Break notifications wait for transport acks
  - ▶ On channel failures they are retried on other channels
  - ▶ Windows doesn't retry for oplocks, only leases

# What is SMB3 Multichannel? (Part 3)

- ▶ Write/Set operations only need a barrier
  - ▶ An epoch number is incremented on each channel failure
  - ▶ The current epoch number is part of each request
  - ▶ The server remembers the last seen epoch number
  - ▶ Non-REPLAY requests with stale epoch fail
  - ▶ REPLAY requests fail, when there are pending older epoch numbers
- ▶ Read/Get operations can be replayed safely
- ▶ Lease/Oplock break notifications should be retried
  - ▶ Break notifications wait for transport acks
  - ▶ On channel failures they are retried on other channels
  - ▶ Windows doesn't retry for oplocks, only leases

# What is SMB3 Multichannel? (Part 3)

- ▶ Write/Set operations only need a barrier
  - ▶ An epoch number is incremented on each channel failure
  - ▶ The current epoch number is part of each request
  - ▶ The server remembers the last seen epoch number
  - ▶ Non-REPLAY requests with stale epoch fail
  - ▶ REPLAY requests fail, when there are pending older epoch numbers
- ▶ Read/Get operations can be replayed safely
- ▶ Lease/Oplock break notifications should be retried
  - ▶ Break notifications wait for transport acks
  - ▶ On channel failures they are retried on other channels
  - ▶ Windows doesn't retry for oplocks, only leases

# Last Status Update SDC 2020

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

# Updates in Samba 4.15

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

# Updates in Samba 4.15

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

# Updates in Samba 4.15

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



# Updates in Samba 4.15

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

# What is io-uring? (Part 1)

- ▶ Linux 5.1 introduced a new scalable AIO infrastructure
  - ▶ It's designed to avoid syscalls as much as possible
  - ▶ kernel and userspace share mmap'ed rings:
    - ▶ submission queue (SQ) ring buffer
    - ▶ completion queue (CQ) ring buffer
  - ▶ See "[Ring in a new asynchronous I/O API](#)" on LWN.NET
- ▶ This can be nicely integrated with our async event model
  - ▶ It may delegate work to kernel threads
  - ▶ It seems to perform better compared to our userspace threadpool
  - ▶ It can also inline non-blocking operations

# What is io-uring? (Part 1)

- ▶ Linux 5.1 introduced a new scalable AIO infrastructure
  - ▶ It's designed to avoid syscalls as much as possible
  - ▶ kernel and userspace share mmap'ed rings:
    - ▶ submission queue (SQ) ring buffer
    - ▶ completion queue (CQ) ring buffer
  - ▶ See "[Ringing in a new asynchronous I/O API](#)" on LWN.NET
- ▶ This can be nicely integrated with our async tevent model
  - ▶ It may delegate work to kernel threads
  - ▶ It seems to perform better compared to our userspace threadpool
  - ▶ It can also inline non-blocking operations

# io-uring for Samba (Part 1)

- ▶ Between userspace and filesystem (available from 5.1):
  - ▶ IORING\_OP\_READV, IORING\_OP\_WRITEV and IORING\_OP\_FSYNC
  - ▶ Supports buffered and direct io
- ▶ Between userspace and socket (and also filesystem) (from 5.8)
  - ▶ IORING\_OP\_SENDMSG, IORING\_OP\_RECVMSG
  - ▶ Improved MSG\_WAITALL support (5.12, backport to 5.11, 5.10)
  - ▶ IORING\_OP\_SPLICE, IORING\_OP\_TEE
  - ▶ Maybe using IORING\_SETUP\_SQPOLL or IOSQE\_ASYNC
- ▶ Path based syscalls with async impersonation (from 5.6)
  - ▶ IORING\_OP\_OPENAT2, IORING\_OP\_STATX
  - ▶ Using IORING\_REGISTER\_PERSONALITY for impersonation
  - ▶ IORING\_OP\_UNLINKAT, IORING\_OP\_RENAMEAT (from 5.10)

# io-uring for Samba (Part 1)

- ▶ Between userspace and filesystem (available from 5.1):
  - ▶ IORING\_OP\_READV, IORING\_OP\_WRITEV and IORING\_OP\_FSYNC
  - ▶ Supports buffered and direct io
- ▶ Between userspace and socket (and also filesystem) (from 5.8)
  - ▶ IORING\_OP\_SENDMSG, IORING\_OP\_RECVMSG
  - ▶ Improved MSG\_WAITALL support (5.12, backport to 5.11, 5.10)
  - ▶ IORING\_OP\_SPLICE, IORING\_OP\_TEE
  - ▶ Maybe using IORING\_SETUP\_SQPOLL or IOSQE\_ASYNC
- ▶ Path based syscalls with async impersonation (from 5.6)
  - ▶ IORING\_OP\_OPENAT2, IORING\_OP\_STATX
  - ▶ Using IORING\_REGISTER\_PERSONALITY for impersonation
  - ▶ IORING\_OP\_UNLINKAT, IORING\_OP\_RENAMEAT (from 5.10)

# io-uring for Samba (Part 1)

- ▶ Between userspace and filesystem (available from 5.1):
  - ▶ `IORING_OP_READV`, `IORING_OP_WRITEV` and `IORING_OP_FSYNC`
  - ▶ Supports buffered and direct io
- ▶ Between userspace and socket (and also filesystem) (from 5.8)
  - ▶ `IORING_OP_SENDMSG`, `IORING_OP_RECVMSG`
  - ▶ Improved `MSG_WAITALL` support (5.12, backport to 5.11, 5.10)
  - ▶ `IORING_OP_SPLICE`, `IORING_OP_TEE`
  - ▶ Maybe using `IORING_SETUP_SQPOLL` or `IOSQE_ASYNC`
- ▶ Path based syscalls with async impersonation (from 5.6)
  - ▶ `IORING_OP_OPENAT2`, `IORING_OP_STATX`
  - ▶ Using `IORING_REGISTER_PERSONALITY` for impersonation
  - ▶ `IORING_OP_UNLINKAT`, `IORING_OP_RENAMEAT` (from 5.10)

## io-uring for Samba (Part 2)

### IORING\_FEAT\_NATIVE\_WORKERS (from 5.12)

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

## io-uring for Samba (Part 2)

### IORING\_FEAT\_NATIVE\_WORKERS (from 5.12)

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



# Performance research (SMB2 Read)

- ▶ Last October I was able to do some performance research
  - ▶ DDN was so kind to sponsor about a week of research on real world hardware
  - ▶ With 100Gbit/s interfaces and two NUMA nodes per server.
- ▶ I focussed on the SMB2 Read performance only
  - ▶ We had limited time on the given hardware
  - ▶ We mainly tested with `file.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)

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

# Performance research (SMB2 Read)

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

# Performance with MultiChannel, sendmsg()

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

```
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, 185648.1 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
307372	root	20	0	2426196	67808	13104	I	0.0	0.0	0:06.77	send
307406	root	20	0	2426196	67408	13104	R	14.3	0.0	0:06.96	send
307412	root	20	0	2426196	65256	13104	R	14.8	0.0	0:06.92	send
307405	root	20	0	2426196	63144	13104	R	13.6	0.0	0:06.82	send
307410	root	20	0	2426196	64644	13104	R	13.6	0.0	0:06.87	send
307414	root	20	0	2426196	65520	13104	R	13.6	0.0	0:06.88	send
307422	root	20	0	2426196	68952	13104	R	13.6	0.0	0:06.78	send
307432	root	20	0	2426196	71592	13104	R	13.6	0.0	0:06.66	send
307408	root	20	0	2426196	63936	13104	R	13.3	0.0	0:06.58	send
307411	root	20	0	2426196	64992	13104	R	13.3	0.0	0:06.77	send
307413	root	20	0	2426196	65256	13104	R	13.3	0.0	0:06.68	send
307415	root	20	0	2426196	65520	13104	R	13.3	0.0	0:06.69	send
307410	root	20	0	2426196	66048	13104	R	13.3	0.0	0:06.69	send
307419	root	20	0	2426196	67104	13104	R	13.3	0.0	0:06.84	send
307420	root	20	0	2426196	67632	13104	R	13.3	0.0	0:06.76	send
307421	root	20	0	2426196	68160	13104	R	13.3	0.0	0:06.71	send
307423	root	20	0	2426196	69408	13104	R	13.3	0.0	0:06.68	send
307425	root	20	0	2426196	69408	13104	R	13.3	0.0	0:06.59	send
307428	root	20	0	2426196	70808	13104	R	13.3	0.0	0:06.59	send
307430	root	20	0	2426196	70808	13104	R	13.3	0.0	0:06.84	send
307433	root	20	0	2426196	72384	13104	R	13.3	0.0	0:06.61	send
307426	root	20	0	2426196	70808	13104	R	13.0	0.0	0:06.62	send
307429	root	20	0	2426196	70808	13104	R	13.0	0.0	0:06.67	send
307434	root	20	0	2426196	72384	13104	R	13.0	0.0	0:06.70	send
307435	root	20	0	2426196	72648	13104	R	13.0	0.0	0:06.71	send
307407	root	20	0	2426196	63672	13104	R	12.6	0.0	0:06.58	send
307416	root	20	0	2426196	66048	13104	R	12.6	0.0	0:06.68	send
307417	root	20	0	2426196	66312	13104	R	12.6	0.0	0:06.53	send
307427	root	20	0	2426196	70808	13104	R	12.6	0.0	0:06.87	send
307431	root	20	0	2426196	71064	13104	R	12.6	0.0	0:06.58	send
307424	root	20	0	2426196	69408	13104	R	12.3	0.0	0:06.65	send
307409	root	20	0	2426196	64200	13104	R	12.0	0.0	0:06.68	send
307404	root	20	0	2426196	62616	13104	R	11.3	0.0	0:06.63	send
307183	root	0	0	0	0	0	I	0.3	0.0	0:00.41	kworke
307392	root	20	0	0	0	0	I	0.3	0.0	0:00.83	kworke
307452	root	20	0	62928	5536	3936	R	0.3	0.0	0:00.08	top
1	root	0	0	242512	18952	8176	S	0.0	0.0	0:02.84	system
2	root	0	0	0	0	0	S	0.0	0.0	0:00.13	khreadd
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	kworke
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	0	0	0	0	0	S	0.0	0.0	0:00.32	ksoftirqd/0
12	root	0	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

Administrator: Windows PowerShell

```
complete : 0=0.08, 4=09.78, 0=0.38, 16=0.18, 32=0.08, 64=0.08, +=64=0.08
issued puts: 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=3260MiB/s (3425MB/s), 3260MiB/s-3260MiB/s (3425MB/s-3425MB/s), io=8000MiB (8395MB), run=2451-2451msec
PS C:\Users\Administrator> & { .\Program Files\Foxit Software\Foxit Reader\Foxit Reader.exe --group_reporting=1 --name=foo_test --ioengine=windowsaio --iothread=16 --direct
...
foo_test: (g=0) r=rcwd, b=(0) 4095KiB-4095KiB, (w) 4095KiB-4095KiB, (r) 4095KiB-4095KiB, ioengine=windowsaio, iothread=16
...
foo-3.22
Starting 4 threads
Jobs: 2 (+2): [R(2)][I(17.38)][r=3812MiB/s][r=352 IOPS][eta 04m:08s]
```

Task Manager - Performance

- CPU: 8% 2.78 GHz
- Memory: 12/512 GB (2%)
- Ethernet: 9.3 Mbps R: 31.9 Gbps
- Ethernet: 40.0 Kbps R: 64.0 Kbps

5 items

4:41 AM  
10/2/2020

# IORING\_OP\_SENDMSG prototyped (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
Mem Swap: 191624.1 total, 182194.6 free, 2702.6 used, 6726.9 buff/cache
Mem Swap: 1024.0 total, 1024.0 free, 0.0 used. 185554.7 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
307577	root	20	0	0	0	0	R	49.0	0.0	0:05.80	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	10.9	0.1	0:05.61	smbd
307562	root	20	0	663100	144024	18804	S	10.5	0.1	0:00.93	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	0:03.50	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	0:02.04	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-kblol
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	kssoftirq/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	kssoftirq/1
19	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-kblol
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	kssoftirq/2
25	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0H-kblol
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 short=0,0,0 dropped=0,0,0
latency : target=0, window=0, percentile=100.0%, depth=16
```

Run status group 0 (all jobs):  
READ: bw=5396MiB/s (5658MB/s), 4096KiB/s-5396MiB/s (4295MB/s-5658MB/s), io=253GiB (2710  
PS C:\Users\Administrator> & 'C:\Program Files\Fio\Fio.exe' --group\_reporting=1 --name=fio  
-l1 --thread --rwread --size=100M --bs=4M --numJobs=2 --time\_based=1 --runtime=5m --direct  
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)][15.3K][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:a4b%19

Fewer details Open Resource Monitor

5 items



# IORING\_OP\_SENDMSG prototyped (Part3)

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

Examples: 178K of event 'cycles', 4000 Hz, Event count (approx.): 87301350677 Lost: 0/0 dropped: 0/0

Verhead	Shared Object	Symbol
05.07%	[kernel]	[k] copy_user_enhanced_fast_string
08.20%	[kernel]	[k] shmем_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
0.30%	[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] shmем_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 rwts: 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\io\io.exe' --group_report
r1 --thread --rread --size=100M --bs=4M --numjobs=2 --time_based=1 --run
flo_test: (g=0): rw=read, bs=(R) 4096KiB-4096KiB, (W) 4096KiB-4096KiB, (T)
r...
flo=3.22
Starting 2 threads
Jobs: 2 (f=2): [R(2)][22.0%][r=6811MiB/s][r=1702 IOPS][eta 03m:54s]
```

Task Manager

File Options View

Processes Performance Users Details Services

CPU 16% 2.78 GHz

Memory 12/512 GB (2%)

Ethernet Si: 15.7 Mbps R: 57.5 Gbps

Ethernet Si: 40.0 Kbps R: 96.0 Kbps

Ethernet

Throughput

Send and receive activity network

60 seconds

Send 15.7 Mbps

Receive 57.5 Gbps

Adapter name: SLOT 4 Port 1

Connection type: Ethernet

IPv4 address: 192.168.0.153

IPv6 address: fe80::d5a5:8115

Fewer details | Open Resource Monitor

5 items





# smbclient IORING\_OP\_SENDMSG/SPLICE (network)

4 connections, ~11 GBytes/s, smbdc 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 [3175909.5 KiloBytes/sec] (average 3223967.9 KiloBytes/sec)
getting file %506.dat of size 2097152000 as /dev/null [3181213.7 KiloBytes/sec] (average 3179906.9 KiloBytes/sec)
getting file %506.dat of size 2097152000 as /dev/null [2824827.2 KiloBytes/sec] (average 2828605.4 KiloBytes/sec)
getting file %506.dat of size 2097152000 as /dev/null [3255961.3 KiloBytes/sec] (average 3224002.5 KiloBytes/sec)
getting file %506.dat of size 2097152000 as /dev/null [2782680.3 KiloBytes/sec] (average 2746033.3 KiloBytes/sec)
getting file %506.dat of size 2097152000 as /dev/null [3230283.4 KiloBytes/sec] (average 3176965.0 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 2822636.0 KiloBytes/sec)
getting file %506.dat of size 2097152000 as /dev/null [3185909.5 KiloBytes/sec] (average 3170974.0 KiloBytes/sec)
getting file %506.dat of size 2097152000 as /dev/null [2797913.0 KiloBytes/sec] (average 2746894.5 KiloBytes/sec)
getting file %506.dat of size 2097152000 as /dev/null [3250793.1 KiloBytes/sec] (average 3224021.0 KiloBytes/sec)
```

```
top - 02:41:50 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, 4.0 sy, 0.0 ni, 93.5 id, 0.0 wa, 0.0 hi, 1.7 si, 0.0 st
Mem Mem : 191880.7 total, 127133.7 free, 3813.5 used, 60941.4 buff/cache
Mem Swap: 1824.0 total, 737.0 free, 287.0 used, 131646.0 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
740188	root	20	0	375600	35960	16852	R	99.3	0.0	0:35.55	smbclient
740185	root	20	0	375604	36180	17016	R	99.0	0.0	9:30.87	smbclient
740187	root	20	0	375692	35880	16696	R	88.1	0.0	0:44.08	smbclient
740186	root	20	0	375652	35896	16748	R	86.4	0.0	0:49.29	smbclient
100189	root	20	0	31540	7872	3412	S	2.0	0.0	100:03:15	lsop
238	root	20	0	0	0	0	S	1.3	0.0	0:56.18	kssoftirq/45
740176	root	20	0	249536	8076	5130	S	1.3	0.0	0:11.20	lsftp

```
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 : 191624.1 total, 17240.5 free, 3855.5 used, 11320.1 buff/cache
Mem Swap: 1824.0 total, 1824.0 free, 0.0 used, 108675.2 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_wqeworker-0
316133	root	20	0	0	0	0	S	20.3	0.0	0:53.37	io_wqeworker-0
316139	root	20	0	0	0	0	S	17.9	0.0	0:46.39	io_wqeworker-0
316121	root	20	0	0	0	0	S	17.3	0.0	0:34.40	io_wqeworker-0
316116	root	20	0	450800	21264	17652	S	8.6	0.0	0:46.53	smbd

Sampls: 780 of event 'cycles', 4000 Hz, Event count (approx.): 35349326236 last: 0/0 drop: 0/32090

Overhead	Shared object	Symbol	1546038464cb	389286928cb	4638091264cb	6184121056cb778152440cb
7.05%	[kernel]	[k] do_tcp_sendpages	192.168.10.191	==> 192.168.10.190		91.7cb 91.5cb 89.7cb
5.37%	[kernel]	[k] raw_spin_lock_bh		==>		18.3cb 18.7cb 18.6cb
4.00%	[kernel]	[k] copy_page_to_iter	192.168.10.191	==> 192.168.0.153		0b 0b 238b
3.75%	[kernel]	[k] page_cache_pipe_buf_release		==>		0b 0b 210b
3.09%	[kernel]	[k] __mg_retpoline_rx				
3.09%	[kernel]	[k] page_cache_pipe_buf_confirm				
2.07%	[kernel]	[k] native_queued_spin_lock_slowpath				
2.04%	[kernel]	[k] shmem_file_read_iter				
2.03%	[kernel]	[k] inet_sendpage				
2.01%	[kernel]	[k] tcp_sendpage				

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

	1546038464cb	389286928cb	4638091264cb	6184121056cb778152440cb
TX:	cus: 3146B	peak: 0b		rates: 91.7cb 91.5cb 89.7cb
RX:	68.7MB	22.1Mb		18.3cb 18.7cb 18.6cb
TOTAL:	3146B	0b		91.0cb 91.5cb 89.7cb

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

8 connections, ~22 GBytes, 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 %S6.dat of size 2097152000 as /dev/null	13075974.0 KiloBytes/sec	average 208000.8 KiloBytes/sec	top - 04:00:58 up 4 days, 23:02, 0 users, load average: 9.15, 3.56, 1.44 Tasks: 917 total, 14 running, 903 sleeping, 0 stopped, 0 zombie Cpus(s): 0.3 us, 11.2 sy, 0.6 ni, 06.1 id, 0.8 wa, 0.2 hi, 2.1 si, 0.0 st MiB Mem : 101624.1 total, 176025.4 free, 3316.7 used, 11302.0 buff/cache MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 100403.7 avail Mem
getting file %S6.dat of size 2097152000 as /dev/null	20426578.0 KiloBytes/sec	average 2043679.6 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	21778787.2 KiloBytes/sec	average 2041637.3 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	19551800.2 KiloBytes/sec	average 2037437.6 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2401641.2 KiloBytes/sec	average 2037370.9 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13107770.5 KiloBytes/sec	average 2058064.5 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2604716.5 KiloBytes/sec	average 2174142.3 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2806334.0 KiloBytes/sec	average 213460.8 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13117100.0 KiloBytes/sec	average 2090262.3 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	1047610.0 KiloBytes/sec	average 2044350.1 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	10808335.4 KiloBytes/sec	average 2141475.6 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2741832.0 KiloBytes/sec	average 2040912.0 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	10010352.1 KiloBytes/sec	average 2002654.5 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13126717.1 KiloBytes/sec	average 2059135.5 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	1908089.0 KiloBytes/sec	average 2015530.4 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2515970.2 KiloBytes/sec	average 2151740.8 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	1217191.0 KiloBytes/sec	average 2709204.0 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	12025450.2 KiloBytes/sec	average 2044203.8 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	10038655.1 KiloBytes/sec	average 2343270.7 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	10038655.1 KiloBytes/sec	average 2042525.3 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	1007201.7 KiloBytes/sec	average 2011000.4 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13107770.5 KiloBytes/sec	average 2060070.5 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13106293.6 KiloBytes/sec	average 2091072.3 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2752687.0 KiloBytes/sec	average 2131090.3 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	10084305.8 KiloBytes/sec	average 2045095.8 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2745300.0 KiloBytes/sec	average 2709462.2 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13117100.0 KiloBytes/sec	average 2146070.8 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13117100.0 KiloBytes/sec	average 2044253.7 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2503203.2 KiloBytes/sec	average 2010059.0 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2531064.0 KiloBytes/sec	average 2046601.4 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	10038655.1 KiloBytes/sec	average 2094340.3 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2822720.0 KiloBytes/sec	average 2123256.5 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	2771312.2 KiloBytes/sec	average 2709097.3 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	1313490.0 KiloBytes/sec	average 2064001.4 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	1313490.0 KiloBytes/sec	average 2148470.8 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	1259500.4 KiloBytes/sec	average 204252.7 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13030575.2 KiloBytes/sec	average 2957270.6 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	12927643.0 KiloBytes/sec	average 2079300.9 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	13030575.2 KiloBytes/sec	average 2095262.7 KiloBytes/sec	
getting file %S6.dat of size 2097152000 as /dev/null	12824827.2 KiloBytes/sec	average 273109.6 KiloBytes/sec	

  

1575337920b	3151075040b	4726614016b	6302151000b/877769344b
127.0.0.1	= 127.0.0.1	<=	1010b 1010b 1000b 0b 0b 0b
Tx: cum: 226426b peak: 6.59cb rates: 1010b 1010b 1000b Rcv: 0b 0b 0b 0b 0b 0b TOTAL: 226426b 6.59cb 1010b 1010b 1000b			

multichannel / io.uring  
(19/22)

Stefan Metzger



# More loopback testing on brand new hardware

- ▶ Recently I re-did the loopback read tests IORING\_OP\_SENDMSG/SPLICE (from /dev/shm/)
  - ▶ 1 connection, ~11 GBytes/s, smb2 7% cpu, with 4 io\_wqe\_work threads at 7%-50% cpu.
  - ▶ 4 connections, 24-30 GBytes/s, smb2 18% cpu, with 16 io\_wqe\_work threads at 3%-35% cpu.
- ▶ I also prototyped SMB2 writes with IORING\_OP\_RECVMSG/SPLICE (to /dev/null)
  - ▶ 1 connection, ~7 GBytes/s, smb2 5% cpu, with 3 io\_wqe\_work threads at 1%-20% cpu.
  - ▶ 4 connections, ~10 GBytes/s, smb2 15% cpu, with 12 io\_wqe\_work threads at 1%-20% cpu.
- ▶ I tested with a Linux kernel 5.10.25
  - ▶ In both cases the bottleneck is clearly on the smbclient side
  - ▶ We could apply similar changes to smbclient and add true multichannel support
  - ▶ It seems that the filesystem->pipe->socket path is much better optimized

# More loopback testing on brand new hardware

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

## More loopback testing on brand new hardware

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

# Future Improvements

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

# Future Improvements

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

# Future Improvements

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



# Future Improvements

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

# Questions? Feedback!

- ▶ Feedback regarding real world testing would be great!
- ▶ Stefan Metzmacher, [metze@samba.org](mailto:metze@samba.org)
- ▶ <https://www.sernet.com>
- ▶ <https://samba.plus>

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