

# 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!

- ▶ 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

- ▶ Between userspace and filesystem (available from 5.1):
  - ▶ IORING\_OP\_READV, IORING\_OP\_WRITEV and IORING\_OP\_FSYNC
  - ▶ Supports buffered and direct io
  - ▶ IORING\_OP\_FSETXATTR, IORING\_OP\_FGETXATTR (from 5.19)
  - ▶ IORING\_OP\_GETDENTS, under discussion, but seems to be tricky
  - ▶ IORING\_OP\_FADVISE (from 5.6)
- ▶ Path based syscalls with async impersonation (from 5.6)
  - ▶ IORING\_OP\_OPENAT2, IORING\_OP\_STATX
  - ▶ Using IORING\_REGISTER\_PERSONALITY for impersonation
  - ▶ IORING\_OP\_UNLINKAT, IORING\_OP\_RENAMEAT (from 5.10)
  - ▶ IORING\_OP\_MKDIRAT, IORING\_OP\_SYMLINKAT, IORING\_OP\_LINKAT (from 5.15)
  - ▶ IORING\_OP\_SETXATTR, IORING\_OP\_GETXATTR (from 5.19)

- ▶ Between userspace and socket (and also filesystem) (from 5.8)
  - ▶ IORING\_OP\_SENDMSG, IORING\_OP\_RECVMSG
  - ▶ Improved MSG\_WAITALL support (5.12, backported to 5.11, 5.10)
  - ▶ Maybe using IOSQE\_ASYNC in order to avoid inline memcpy
  - ▶ IORING\_OP\_SPLICE, IORING\_OP\_TEE
  - ▶ IORING\_OP\_SENDMSG\_ZC, zero copy with an extra completion (from 6.1)
  - ▶ IORING\_OP\_GET\_BUF, under discussion to replace IORING\_OP\_SPLICE

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

## 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, 3.6 GBytes/s, bound by >500% cpu in total, sendmsg() takes up to 0.5 ms/cs

```
top - 05:45:38 up 2 days, 48 min, 2 users, load average: 3.42, 3.22, 3.12
Mem: 823 total, 33 running, 0 sleeping, 0 stopped, 0 idle
Cpu(s): 0.1 us, 0.3 sy, 0.0 ni, 97.4 id, 0.0 wa, 0.1 bi, 0.2 si, 0.8 st
MiB Mem : 182424 total, 182296 free, 2872.0 used, 6276.0 buff/cache
MiB Swap: 1824.0 total, 1824.0 free, 0.0 used, 1824.0 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	MEM	MEM% <sup>1</sup>	MEM% <sup>2</sup>	TIME+	COMMAND
387322	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
387486	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
387650	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
387814	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
387978	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
388142	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
388306	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
388470	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
388634	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
388798	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
388962	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
389126	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
389290	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
389454	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
389618	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
389782	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
389946	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
390110	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
390274	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
390438	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
390602	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
390766	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
390930	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
391094	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
391258	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
391422	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
391586	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
391750	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
391914	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
392078	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
392242	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
392406	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
392570	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
392734	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
392898	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
393062	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
393226	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
393390	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
393554	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
393718	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
393882	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
394046	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
394210	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
394374	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
394538	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
394702	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
394866	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
395030	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
395194	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
395358	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
395522	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
395686	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
395850	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
396014	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
396178	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
396342	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
396506	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
396670	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
396834	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
396998	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
397162	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
397326	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
397490	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
397654	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
397818	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
397982	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
398146	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
398310	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
398474	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
398638	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
398802	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
398966	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
399130	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
399294	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
399458	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
399622	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
399786	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
399950	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
400114	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
400278	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
400442	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
400606	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
400770	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
400934	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
401098	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
401262	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
401426	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
401590	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
401754	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
401918	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
402082	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
402246	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
402410	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
402574	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
402738	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
402902	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
403066	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
403230	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
403394	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
403558	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
403722	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
403886	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
404050	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
404214	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
404378	root	20	0	2426326	60880	10384	0	36.8	0.0	0.12	24	smbd
404542	root	20	0	242								





## More loopback testing on brand new hardware

- ▶ Recently I re-did the loopback read tests IORING\_OP\_SENDFMSG/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



SAMBA<sup>+</sup>

Stefan Metzmacher

io\_uring (15/21)

SerNet

## 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](https://gitlab.com/samba-team/samba/-/merge_requests/2793)
    - ▶ This merge request converts Samba to use TEVENT\_FD\_ERROR:
      - ▶ [https://gitlab.com/samba-team/samba/-/merge\\_requests/2885](https://gitlab.com/samba-team/samba/-/merge_requests/2885)
      - ▶ (It also simplifies other places in the code without io\_uring)



SAMBA<sup>+</sup>

Stefan Metzmacher

io\_uring (16/21)

SerNet



## 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



SAMBA+

Stefan Metzmacher

io\_uring (17/21)

SerNet

## 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.



SAMBA+

Stefan Metzmacher

io\_uring (18/21)

SerNet

- ▶ Refactoring of smb2\_server.c
  - ▶ add optional IORING\_OP\_SENDMSG, IORING\_OP\_RECVMSG support
- ▶ There are structural problems with splice from a file
  - ▶ I had a discussion with the Linux developers about it:
  - ▶ The page content from the page cache may change unexpectedly
    - ▶ <https://lists.samba.org/archive/samba-technical/2023-February/thread.html#137945>
  - ▶ We may not be able to use IORING\_OP\_SENDMSG/SPLICE by default
  - ▶ Maybe IORING\_OP\_RECVMSG/SPLICE is possible
- ▶ At least we can have only 1 one copy instead of two:
  - ▶ IORING\_OP\_SENDMSG\_ZC is able to avoid copying to the socket
    - ▶ we get an extra completion once the buffers are not needed anymore
  - ▶ This gives good results, between with and without IORING\_OP\_SENDMSG/SPLICE
  - ▶ But I don't have numbers as it doesn't work on loopback
  - ▶ Within VM's improvement can be seen

## Future Improvements

- ▶ I have a prototype for a native io\_uring tevent backend:
  - ▶ The idea is to avoid epoll and only block in io\_uring\_enter()
  - ▶ But the semantics of IORING\_OP\_POLL\_ADD,REMOVE are not useable
    - ▶ <https://lists.samba.org/archive/samba-technical/2022-October/thread.html#137734>
  - ▶ We may get an IORING\_POLL\_CANCEL\_ON\_CLOSE in future
  - ▶ And a usable IORING\_POLL\_LEVEL
- ▶ We can use io\_uring deep inside of the smbclient code
  - ▶ The low layers can just use samba\_io\_uring\_ev\_context\_get\_ring()
  - ▶ And use if available without changing the whole stack

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

→ SerNet/SAMBA+ sponsor booth

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



SAMBA+

Stefan Metzmacher

io\_uring (21/21)

SerNet