credativ

PostgreSQL 18 Asynchronous Disk I/O

How it works under the hood

Josef Machytka <josef.machytka@credativ.de>

2025-10-24 - credativ Tech Talk

credativ GmbH

- Founded 1999 in Jülich, Germany
- Close ties to Open-Source Community
- More than 40 Open-Source experts
- Consulting, development, training, support (3rd-level / 24x7)
- Open-Source infrastructure with Linux, Kubernetes, Proxmox
- Open-Source databases with PostgreSQL
- DevSecOps with Ansible, Puppet, Terraform and others
- Since 2025 independent owner-managed company again

Josef Machytka

credativ

- Professional Service Consultant - PostgreSQL specialist at credativ GmbH
- 33+ years of experience with different databases
- PostgreSQL (13y), BigQuery (7y), Oracle (15y), MySQL (12y), Elasticsearch (5y), MS SQL (5y)
- 10+ years of experience with Data Ingestion pipelines, Data Analysis, Data Lake and Data Warehouse
- 3+ years of practical experience with different LLMs / AI / ML including architecture and principles
- From Czechia, living now 12 years in Berlin

- LinkedIn: linkedin.com/in/josef-machytka
- Medium: medium.com/@josef.machytka
- YouTube: youtube.com/@JosefMachytka

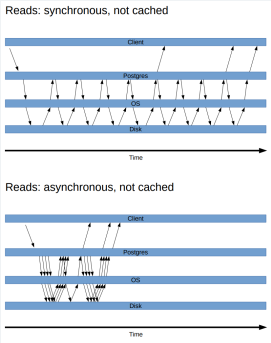
- GitHub: github.com/josmac69/conferences_slides
- ResearchGate: researchgate.net/profile/Josef-Machytka

All My Slides:

Recoded talks:

Streaming I/O in PostgreSQL 17

- PG17 introduced ”streaming I/O” - multiple pages in one request
  - More systematically issues POSIX_FADV_WILLNEED for random access
  - Key posix_fadvise hint to OS for better prefetching
  - Asks kernel to prefetch multiple pages in advance
  - Vectored I/O requests using readv/writev system calls
  - Perform multiple page reads/writes in one syscall
- Improves sequential scan performance by up to 30%
- But still synchronous, one-at-a-time per backend process
- Does not utilize full capabilities of modern storage devices
- Waiting for Postgres 17: Streaming I/O for sequential scans & ANALYZE

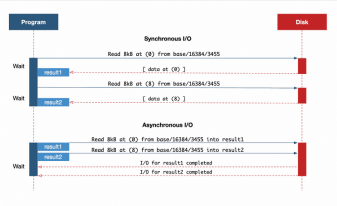


Images from the talk
The path to using AIO in Postgres (PGConf.EU 2023)

Asynchronous I/O

- New AIO subsystem allows multiple concurrent file reads
- Current implementation is only minimal
- Does not support writes - OLTP operations do not benefit
- Multi process model made implementation challenging

- PG18 Async I/O significantly improves performance of:
  - Large sequential scans, GROUP BY, COPY (up to 2-4x faster)
  - Bitmap heap scans
  - Vacuum (some benchmarks up to 3-4x faster, some 0)
  - Read operations on cloud DBs with network-attached storage

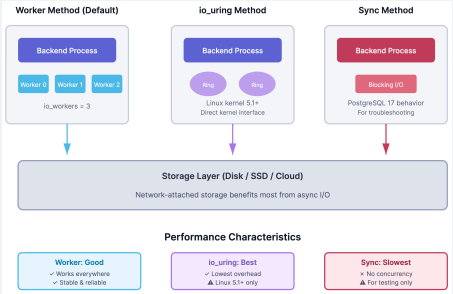


Images from the article
Accelerating Disk Reads with Asynchronous I/O

Async I/O: Configuration Options

- **io_method**:
  - worker - default, pool of I/O worker processes
  - io_uring - Linux-specific async I/O queues
  - sync - "fallback" synchronous I/O, but still uses AIO API

- **io_workers**:
  - 3 - default, too low for larger systems
  - Start with 1/4 of total CPU threads
  - Some benchmarks suggest up to 1/2 of threads

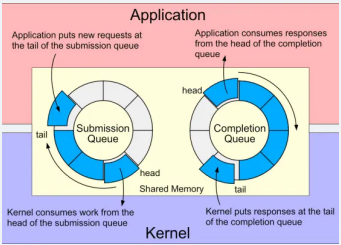


Images from the article
PostgreSQL 18 Asynchronous I/O: A Complete Guide

Linux io_uring method - overview

- Modern Linux I/O interface - needs Linux 5.1+
  - check with `uname -r` command -> shows kernel version
- Requires Postgres build with –with-liburing / -Dliburing
  - check with `pg_config --configure|grep --color=always liburing`
  - select pg_config from pg_config() where pg_config::text ilike '%liburing%';
  - -> both show if liburing is enabled in build
  - `ldd /path/to/postgres | grep uring` -> shows if liburing is linked

- Backends submit I/O requests directly into ring buffer
- Kernel processes requests asynchronously
- Completion queue entries consumed by backends
- Fewer context switches and wakeups
- Better latency on fast NVMe and network-attached SSDs

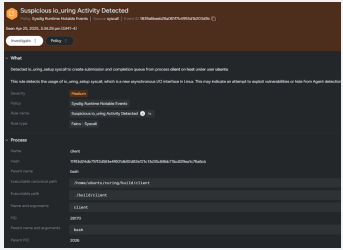


Images from the article
Why you should use io_uring

Linux io_uring method - Security Risks

- io_uring bypasses traditional I/O system read/write calls
- Including network sends/receives
- Rootkits and malware can exploit it to hide I/O activity
- Google reported 60% of kernel exploits in 2022 used io_uring
- Some security tool do not detect io_uring activity
- Sysdig added new detection rules for io_uring abuse
- Therefore some container runtimes disable it for security
- io_uring also complexity led to numerous bugs

- ARMO: io_uring Is Back, This Time as a Rootkit
- Sysdig: Detecting and Mitigating io_uring Abuse for Malware Evasion
- io_uring: Linux Performance Boost or Security Headache?



Images from the Sysdig article

Async io_method = worker: Overview

- Uses dedicated background I/O worker processes
- Backends submit I/O requests to shared memory queue
- I/O workers pick up requests and perform them asynchronously
- Completion queue entries consumed by backends
- More context switches and wakeups than io_uring
- But works on all OSes that support POSIX AIO API
- Recommended for non-Linux systems or older Linux kernels
- Some benchmarks show better performance to io_uring
- Avoids security risks of io_uring method

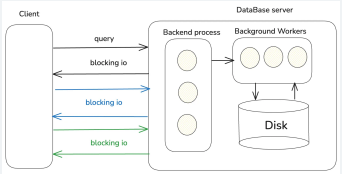


Image from the article
PostgreSQL Async-IO

Async I/O: Monitoring

- Monitored via new pg_aios view:
  - Shows current I/O requests - status, result etc.
  - Status: DEFINED, STAGED, SUBMITTED, COMPLETED_IO, ...
  - Result: UNKNOWN, OK, PARTIAL, WARNING, ERROR
- Enhanced pg_stat_io view - new columns: read_bytes, write_bytes, extend_bytes
- Parameter track_io_timing
  - Enables additional I/O timing statistics
  - Default is off - if on, adds some overhead
- On Linux level using iotop command

```
-- output from iotop command:

   TID PRIO USER      DISK READ   DISK WRITE>    COMMAND
617931 be/4 postgres  32.17 M/s     0.00 B/s postgres: io worker 0
617932 be/4 postgres  72.11 M/s     0.00 B/s postgres: io worker 1
617933 be/4 postgres  41.50 M/s     0.00 B/s postgres: io worker 2
617934 be/4 postgres  58.00 M/s     0.00 B/s postgres: io worker 4
```

Async I/O: Other Configuration Options

credativ

- **io_combine_limit**:
  - 16 - default (i.e. 16 pages = 128kb) - dynamic
  - Without unit - number of pages, with kB/MB - size in kB/MB
  - How many pages can be combined in 1 request
  - Optimal value depends on underlying HW / OS capabilities
  - Hardware and protocols have segment and size limits
  - After some threshold, increasing value has no effect
  - Limited internally in PG by setting io_max_combine_limit

- **io_max_combine_limit**:
  - 16 - default (i.e. 16 pages = 128kb) - requires restart
  - Limits io_combine_limit - maximum pages per request
  - Cluster wide limit for all backends
  - Typical max: Unix 128 (1 MB), Windows 16 (128 kB)
  - Added to manage memory consumption & structure sizes

Async I/O: Other Configuration Options

credativ

- **io_max_concurrency**:
  - maximum number of concurrent I/O operations per process (capped to 64)
  - default -1 = selects number based on shared_buffers and max number of backends
  - Requires restart to change

- **effective_io_concurrency**:
  - number of concurrent I/O operations that can be executed simultaneously
  - new default 16, range 1-1000, value 0 disables async requests
  - Higher values for faster storage - 200 for NVMe SSDs
  - Very high values may increase I/O latency for all queries
  - Can be changed dynamically

How it works under the hood

credativ

- `src/backend/storage/aio/README.md`
- Asynchronous & Direct IO (PostgreSQL Source Code git master)
- `src/backend/storage/aio/method_worker.c`
- Postmaster creates on startup a pool of background worker processes
- Each worker is a separate OS process
- Workers still perform synchronous IO, they just read in parallel
- Blocking syscalls are moved to pool of separater processes
- Workers ignore SIGTERM termination signal
- Get explicit shutdown via SIGUSR2 later in the shutdown sequence
- If worker fails, it marks IO request as failed and exists
- Postmaster starts a new worker to replace it

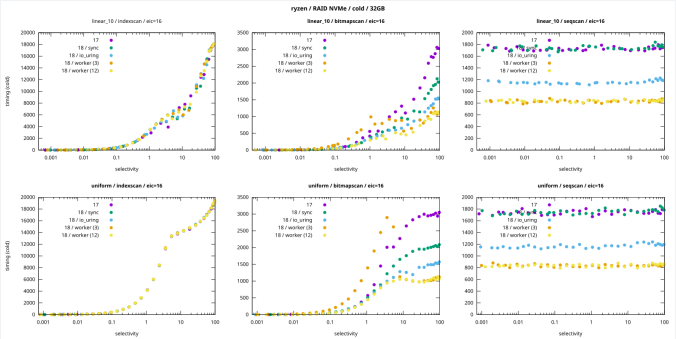
How it works under the hood

credativ

- `src/backend/storage/aio/method_worker.c`
- Each worker loops forever, waiting for requests
- Request for I/O sent to shared memory submission queue
- Worker wakes up, reads requests from shared memory queue
- Each woken IO worker can wake 2 more - note about future improvement to N
- Workers perform synchronous read syscalls for each request
- Put results into shared memory completion queue
- Notify backend process via latch

Async I/O Benchmarks

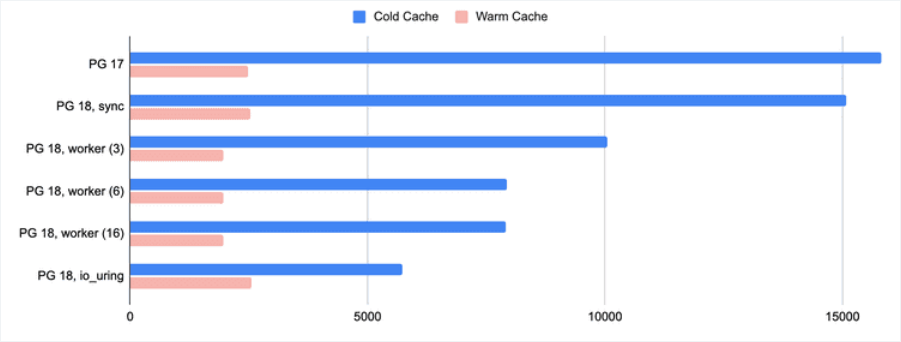
- Tomas Vondra: Tuning AIO in PostgreSQL 18:
- Benchmarks Ryzen 9900X - 12 cores/ 24 threads, 32 GB RAM, 4 NVMe SSD RAID0
- Tests for PG 17, PG 18 with sync, PG 18 with io_uring, PG 18 with AIO worker (3 and 12)
- (uniform = entirely random distribution, linear_10 = sequential with 10% randomness)
- index scan - no effect, index scans do not use AIO yet, only sync I/O
- bitmap scan - 3 workers slower than sync for low selectivity queries, 12 workers 2x faster
- sequential scan - workers 2x faster than sync, io_uring in between



Async I/O Benchmarks

credativ

- Waiting for Postgres 18: Accelerating Disk Reads with Asynchronous I/O:
- AWS c7i.8xlarge instance 32 vCPUs, 64 GB RAM, 100GB io2 EBS volume 20,000 IOPS



Async I/O: Where PG 18 Uses It

credativ

- Major core operations that benefit from AIO:
  - Heap **sequential scans** - plain `SELECT` and `COPY` style scans
  - **ANALYZE** sampling on heap tables
  - **VACUUM** of heap tables and B-tree indexes
  - **Bitmap heap scans** when the executor decides to use them
- These operations usually know many future block numbers in advance
- Can keep multiple reads in flight while doing useful CPU work

- Operations that do *not* yet use AIO:
  - B-tree index scans (including index-only scans)
  - Recovery and replication code paths
  - Write operations (INSERT, UPDATE, DELETE, etc.)
  - Small OLTP lookups -> index + single heap page

Async I/O: Where PG 18 Uses It

credativ

- Autovacuum workers use the same VACUUM and ANALYZE code paths
- Large tables maintained by autovacuum can also profit from async reads
- AIO is intentionally focused on large, predictable scans
  - Avoids fighting the kernel's own readahead heuristics
  - Lets PostgreSQL drive readahead based on query plans
- Future work will add more operations into read streams and AIO
- Definitely index-only scans and eventually writes

Async I/O: Resources

credativ

- Asynchronous & Direct IO (PostgreSQL Source Code git master)
- Tomas Vondra: Tuning AIO in PostgreSQL 18
- Waiting for Postgres 18: Accelerating Disk Reads with Asynchronous I/O
- Boosting Typical Query Patterns - PostgreSQL 18's Performance Enhancements (PGConf.EU 2025)
- Get Excited About Postgres 18
- What went wrong with AIO (PGconf.DEV 2025)
- PostgreSQL 18 Asynchronous I/O (Neon blog)
- PostgreSQL 18: A Comprehensive Guide to New Features for DBAs and Developers

Thank you for your attention!

All my slides