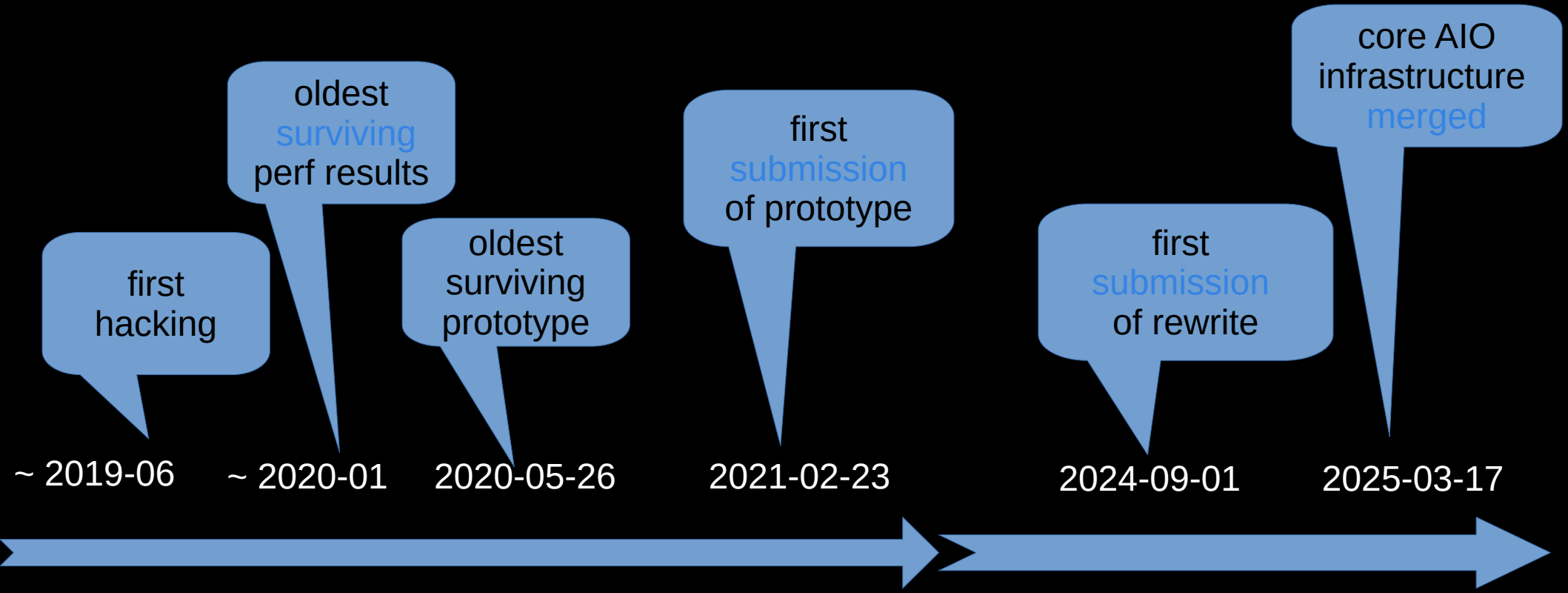
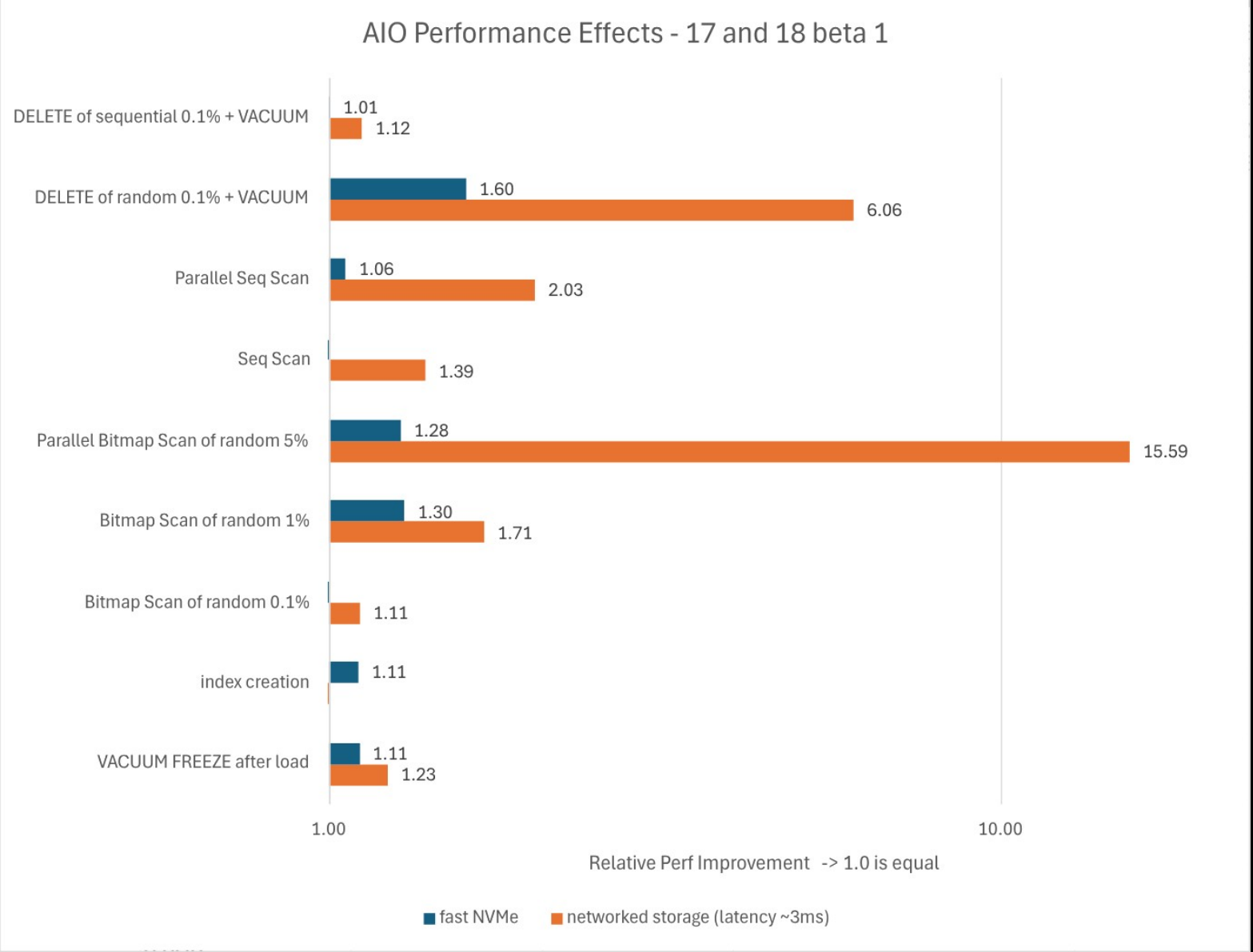
What went wrong with AIO

Andres Freund
PostgreSQL Developer & Committer
Email: andres@anarazel.de
Email: andres.freund@microsoft.com

https://anarazel.de/talks/2025-05-15-pgconf-dev-what-went-wrong-aio/what-went-wrong-aio.pdf

Microsoft


first hacking
~ 2019-06
oldest surviving perf results
~ 2020-01
oldest surviving prototype
2020-05-26
first submission of prototype
2021-02-23
first submission of rewrite
2024-09-01
core AIO infrastructure merged
2025-03-17

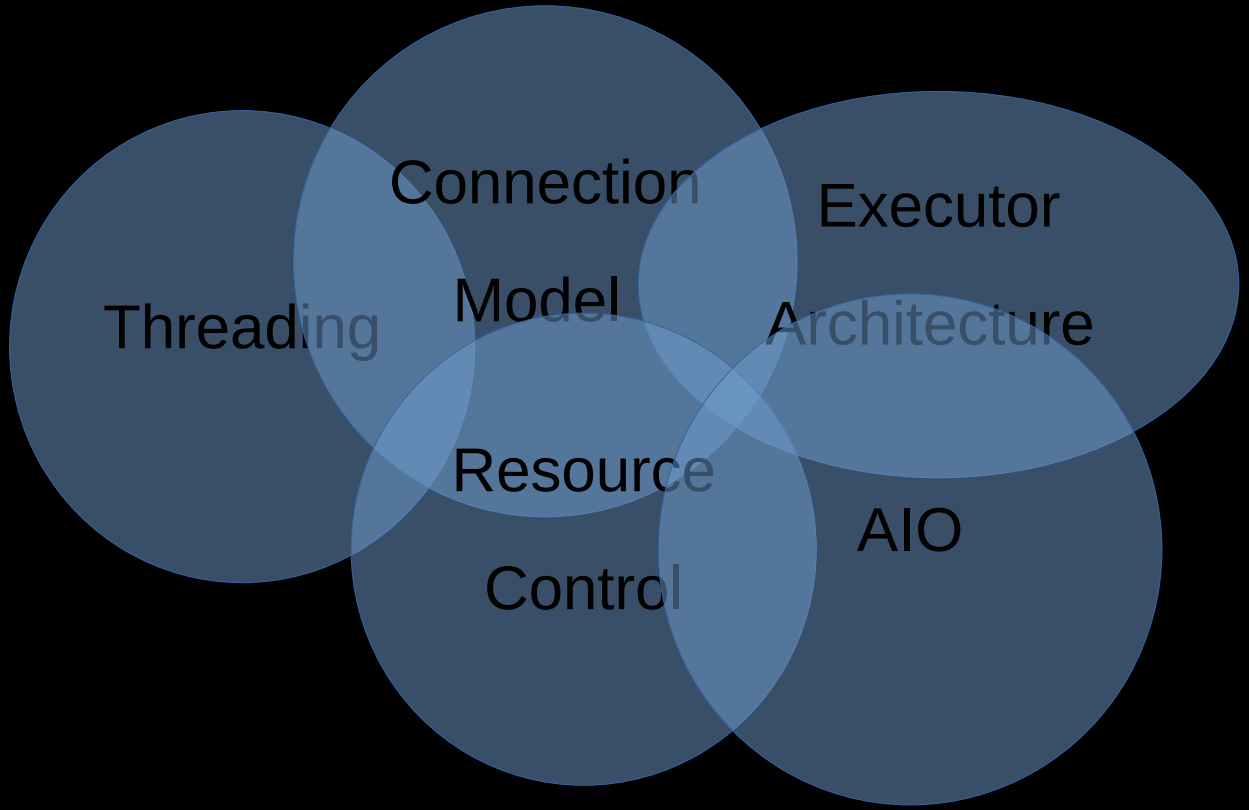


Workload: Bigger than memory

shared_buffers:	10GB
remaining memory:	14GB
table size:	28GB
index size:	8.6GB

PG / OS cache cleared between queries

Microsoft


Threading
Connection
Model
Executor
Architecture
Resource
Control
AIO

Project Level Issues

- AIO interacts with some of the least tested areas of postgres
- Lots of basic assumptions need to be changed
- hard to get architectural review / hard to understand architectural issues without working on AIO
- Postgres not a project doing a lot of in-core iteration

Prototype

- io_uring only first
- worker and then posix_aio (Thomas)
- AIO for reads, checkpointer, bgwriter, backend buffer replacement, sync request queue, WAL writes, …
- lots of unknowns → lots of experimentation / redesign
- Different AIO uses have different design implications

Basic AIO API

/* Acquire an AIO Handle, ioret will get result upon completion. */

PgAioHandle *ioh = pgaio_io_acquire(CurrentResourceOwner, &ioret);

pgaio_io_get_wref(ioh, &iow);

/* update buffer desc state on completion */

pgaio_io_register_callbacks(ioh, PGAIO_HCB_SHARED_BUFFER_READV, 0);

/* start IO on lower level */

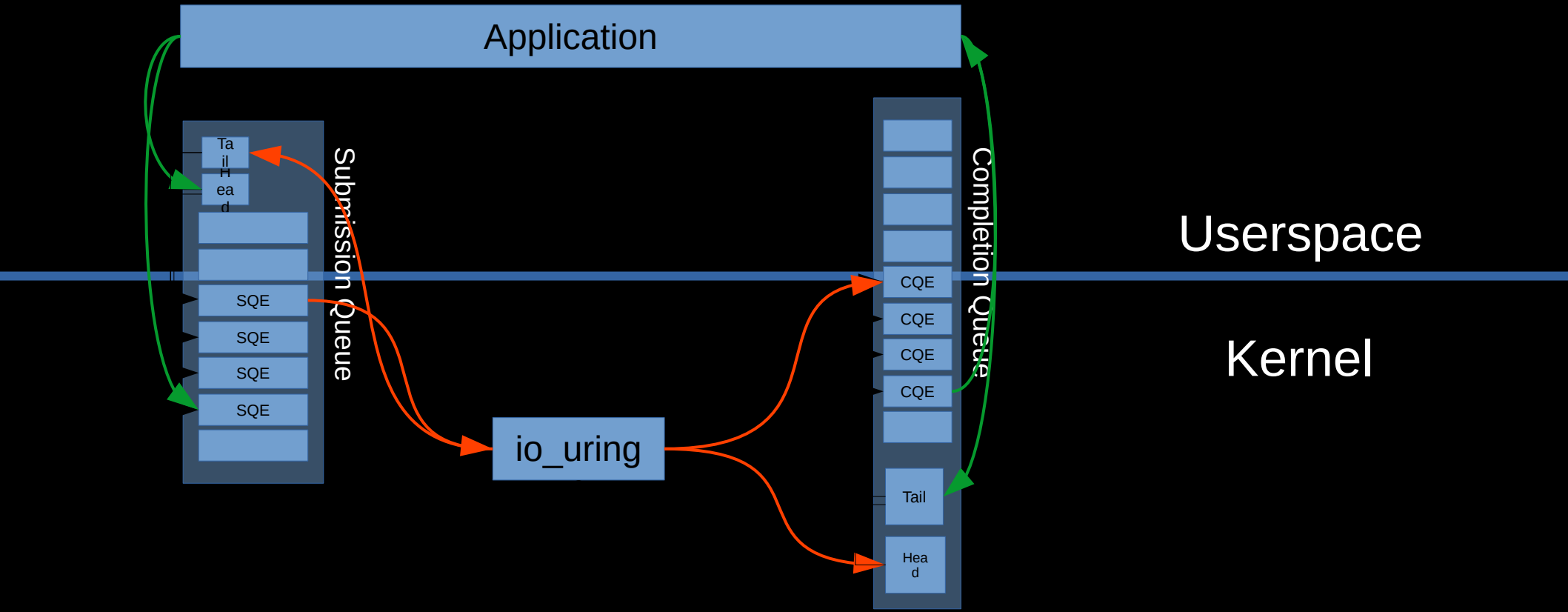
```
smgrstartreadv(ioh, operation->smgr, forknum, blkno,
               BufferGetBlock(buffer), 1);
...
/* now block waiting for IO */
pgaio_wref_wait(&iow);
```

Microsoft

io_uring

- New linux AIO interface, added in 5.1
- Generic, quite a few operations supported
  - open / close / readv / writev / fsync, statx, …
  - send/recv/accept/connect/…, including polling
- One single-reader / single writer ring for IO submission, one SPSC ring for completion
  - allows batched “syscalls”
- Operations that aren’t fully asynchronous are made asynchronous via kernel threads

io_uring basics



Microsoft

Time Sink: Prototype Perf Issues

- Prototype had too high lock overhead
- WAL tuning fscking hard
  - some drives iodepth 1 fastest, others need high
  - some drives small blocks fastest, others largest
  - very latency dependent
  - very workload dependent
- weird perf issues around reads from page cache w/ strategies (SMAP causing slowdowns)

Time Sink: Architecture Mistakes

- limited number of io_uring rings
  - was too concerned with FD limits
- arbitrary # of IOs can be reserved
- IO merging purely inside AIO layer, rather than users
- AIO buffer replacement
- posix_aio support

Lessons: Prototype

- Prototype was absolutely crucial
- Code quality went down
- Don’t optimize all exploratory things as much
- Actually tackle architectural mistakes when they come up

Subprojects

- replace buffer I/O locks with condition variables (Robert, Thomas, 2021)
- bulk relation extension (PG 16 / 2023)
  - includes crucial redesign of buffer management
- aligned allocation (David R, PG 16 / 2023)
- scatter / gather reads & writes (Thomas, PG 17 / 2024)
- read streams & read stream conversions (Thomas et al, PG 17 / 2024)

Time Sink: Arguable Dependencies

- Adding CI
  - Hindsight: definitely the right call
- Adding meson based build system (test runs)
  - before
    - no way to run all tests on windows
    - test parallelism really low
    - hard to find test failures
  - Hindsight: ???

Coincidental Important Developments

- postmaster child & state management
- backend startup refactoring
- procnumber "unification"

A Personal View

- Life sometimes <bleep>
- Committer vs Path Author responsibility
- Maintaining motivation for 6+ years is hard

Parallelize Development

- Architectural work was very hard to parallelize
- Parallel work added dependency bubbles
- Focusing on other AIO users not in critical paths for core AIO infra

AIO in PG 18

- Core AIO infrastructure
- "sync", "worker", "io_uring" io methods
- buffered reads via streaming read interface use AIO
  - seqscans, pg_prewarm (17)
  - bitmap heap scans, vacuum, ... (18)
- no writes in 18!

Microsoft

AIO Future

- bufmgr infrastructure for writes
  - race free exclusive lock vs io-in-progress check
- critical section safe sync request handling
- don't set hint bits on buffers being written out
- redesign bulk strategy interface to be smarter about writes
- make IO methods smarter
- other IO methods

AIO Future

- use AIO for
  - checkpointer, bgwriter writes
  - writes in backends
  - table tuples index scans
  - ...
- Integrate Network IO with AIO

Thanks!

- Thomas (read streams, vectored IO, workers, ...)
- Melanie (read stream users, readahead logic)
- Bilal (CI, meson, read stream users, ...)
- Noah (review)
- Heikki (review)
- Many others
- Microsoft (5 years of work)