Django, SQLite, and Production

PyCon Portugal

Anže Pečar, Oct 18 2024

What is SQLite?

Most widely deployed ~~database~~ software in the world

Every Android device
Every iOS device
Every Mac
Every Windows 10 machine
Every Firefox, Chrome, and Safari web browser

```
import sqlite3
```

Default database for Django

SQLite in production?

Ok, but what about web apps?

"When starting your first real project, however, you may want to use a more scalable database like PostgreSQL, to avoid database-switching headaches down the road."
- Django docs

"SQLite works great as the database engine for most low to medium traffic websites (which is to say, **most websites**).

[...]

Generally speaking, any site that gets fewer than 100K hits/day should work fine with SQLite."

SQLite Docs
https://www.sqlite.org/whentouse.html

It depends

Read only

```
db.sqlite3
```

Readonly Benchmark

Total Requests per Second
● RPS
● Failures/s
Run #1
Run #2
Run #3
2,500
2,000
1,500
1,000
500
0
SQLite
PostgreSQL
MySQL
3:01:05 PM
3:01:09 PM
3:01:13 PM
3:01:17 PM
3:01:21 PM
3:01:45 PM
3:01:49 PM
3:01:53 PM
3:01:57 PM
3:02:01 PM
3:02:21 PM
3:02:25 PM
3:02:29 PM
3:02:34 PM
3:02:38 PM

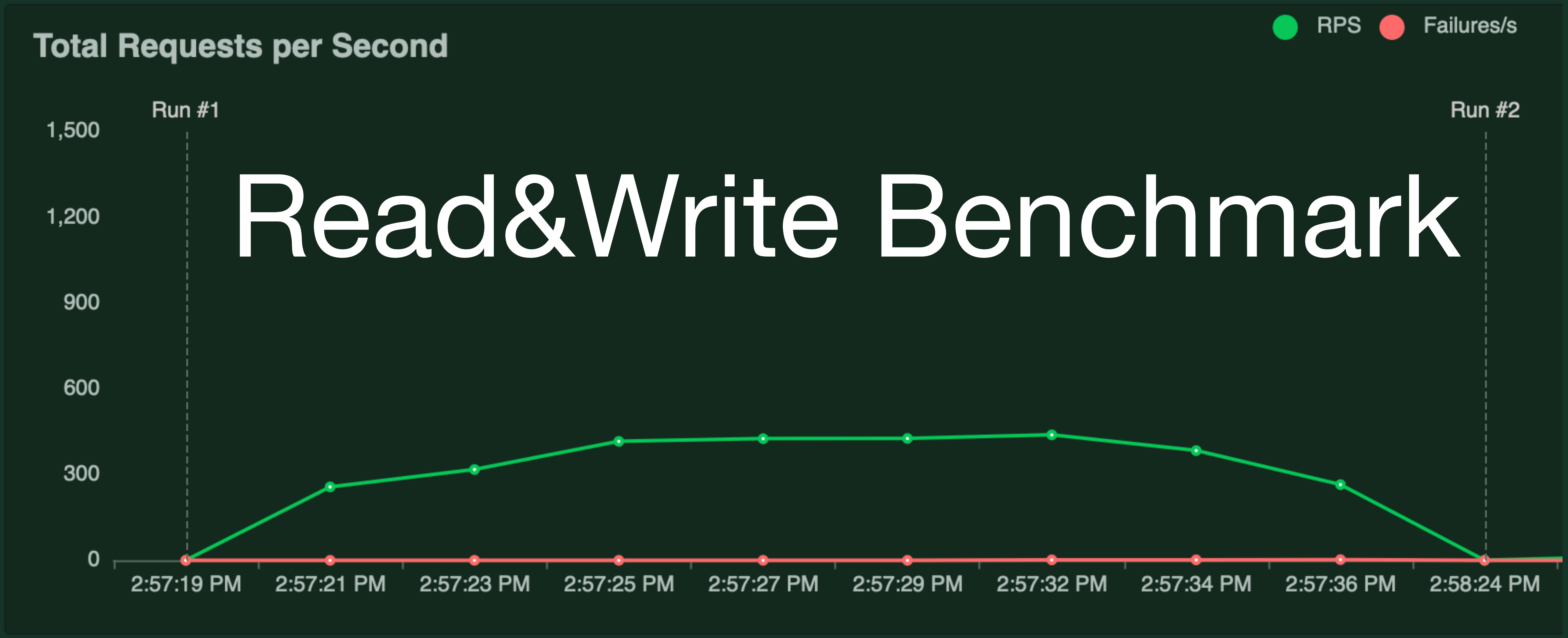
Response Times (ms)
● 50th percentile
● 95th percentile
Run #1
Run #2
Run #3
1,500
1,200
900
600
300
0
3:01:05 PM
3:01:09 PM
3:01:13 PM
3:01:17 PM
3:01:21 PM
3:01:45 PM
3:01:49 PM
3:01:53 PM
3:01:57 PM
3:02:01 PM
3:02:21 PM
3:02:25 PM
3:02:29 PM
3:02:34 PM
3:02:38 PM

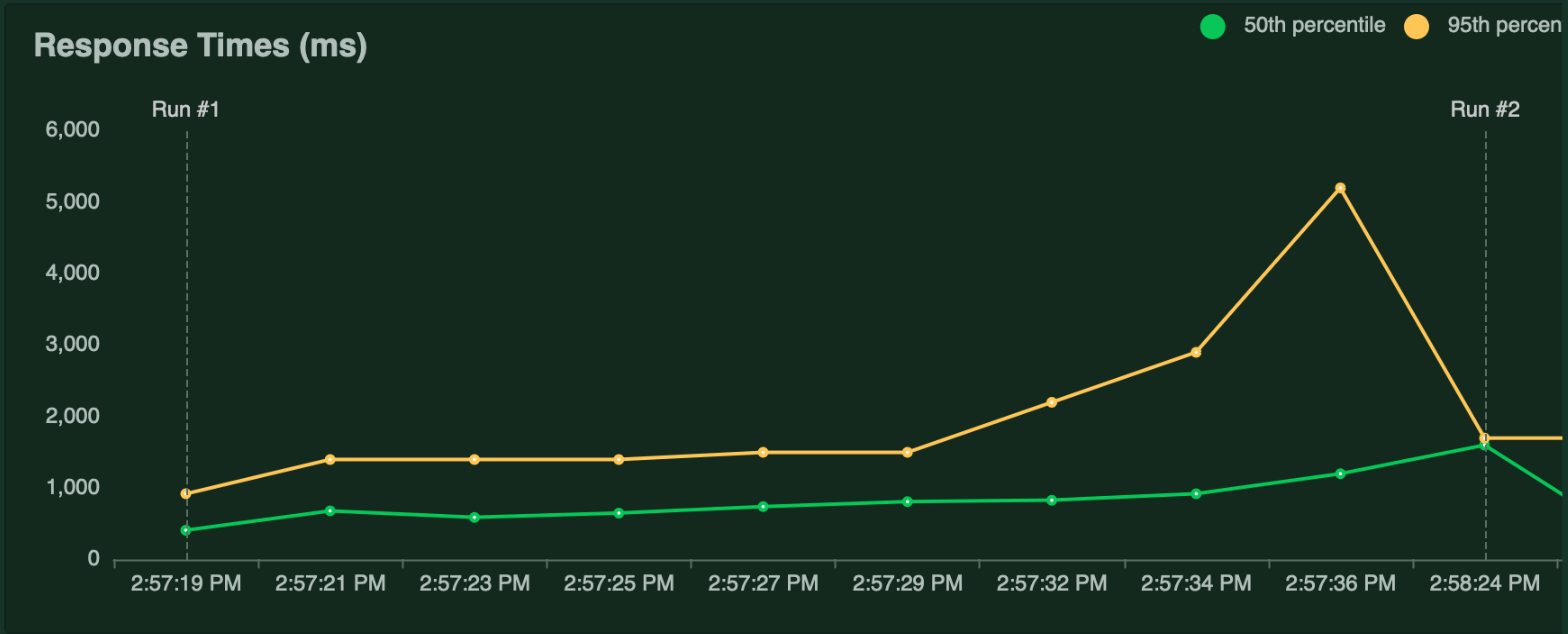
Honeymoon stage

Writes

Writes block

Read&Write Benchmark





Unblock reads

```
PRAGMA journal_mode='WAL'
```

Read & Write Benchmark (WAL)

Total Requests per Second
● RPS ● Failures/s
1,200
1,000
800
600
400
200
0
2:53:40 PM
2:53:42 PM
2:53:44 PM
2:53:46 PM
2:53:48 PM
2:53:50 PM
2:53:52 PM
2:53:54 PM
2:53:56 PM
2:53:58 PM

Response Times (ms)
● 50th percentile ● 95th percentile
1,000
800
600
400
200
0
2:53:40 PM
2:53:42 PM
2:53:44 PM
2:53:46 PM
2:53:48 PM
2:53:50 PM
2:53:52 PM
2:53:54 PM
2:53:56 PM
2:53:58 PM

Database is locked

```
timeout=999999
```

New issue

We notified recently active members in the fedidevs project of this issue

ISSUE

OperationalError /

database is locked

Aug. 28, 2023, 6:38:35 a.m. UTC ID: 5d7b215636954b5488547295551d8d21

project	fedidevs
environment	production
level	error

Suspect Commits

Strip whitespace

c5a1103 — **Anže Pečar**

Exception

```
OperationalError: database is locked
  File "django/db/backends/utils.py", line 89, in _execute
    return self.cursor.execute(sql, params)
  File "django/db/backends/sqlite3/base.py", line 328, in execute
    return super().execute(query, params)

OperationalError: database is locked
(15 additional frame(s) were not displayed)
...
  File "accounts/views.py", line 41, in index
    page_obj = paginator.get_page(page_number)
```

Crisis stage

Transactions

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Possible	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Possible	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Possible	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

Deferred transactions

```
BEGIN;
SELECT * FROM auth_user;
UPDATE auth_user SET last_login_date = NOW();
COMMIT;
```

Deferred transactions

```
BEGIN;
SELECT * FROM auth_user;
UPDATE auth_user SET last_login_date = NOW();
COMMIT;
```

Deferred transactions

```
BEGIN;
SELECT * FROM auth_user;
UPDATE auth_user SET last_login_date = NOW();
COMMIT;
```

Deferred transactions

```
BEGIN;
SELECT * FROM auth_user;
UPDATE auth_user SET last_login_date = NOW();
COMMIT;
```

Deferred transactions

```
BEGIN;
SELECT * FROM auth_user;
UPDATE auth_user SET last_login_date = NOW();
COMMIT;
```

Deferred transactions

```
BEGIN;
SELECT * FROM auth_user;
UPDATE auth_user SET last_login_date = NOW();
COMMIT;
```

```
BEGIN IMMEDIATE;
SELECT * FROM auth_user;
UPDATE auth_user SET last_login_date = NOW();
COMMIT;
```

Write heavy

One concurrent write per <u>database</u>

Writeonly Benchmark

Total Requests per Second
● RPS ● Failures/s
Run #1
Run #2
Run #3
SQLite
PostgreSQL
MySQL
1,800
1,500
1,200
900
600
300
0
2:42:15 PM
2:42:20 PM
2:42:24 PM
2:42:28 PM
2:42:32 PM
2:43:04 PM
2:43:08 PM
2:43:12 PM
2:43:16 PM
2:43:20 PM
2:43:43 PM
2:43:47 PM
2:43:51 PM
2:43:55 PM
2:43:59 PM

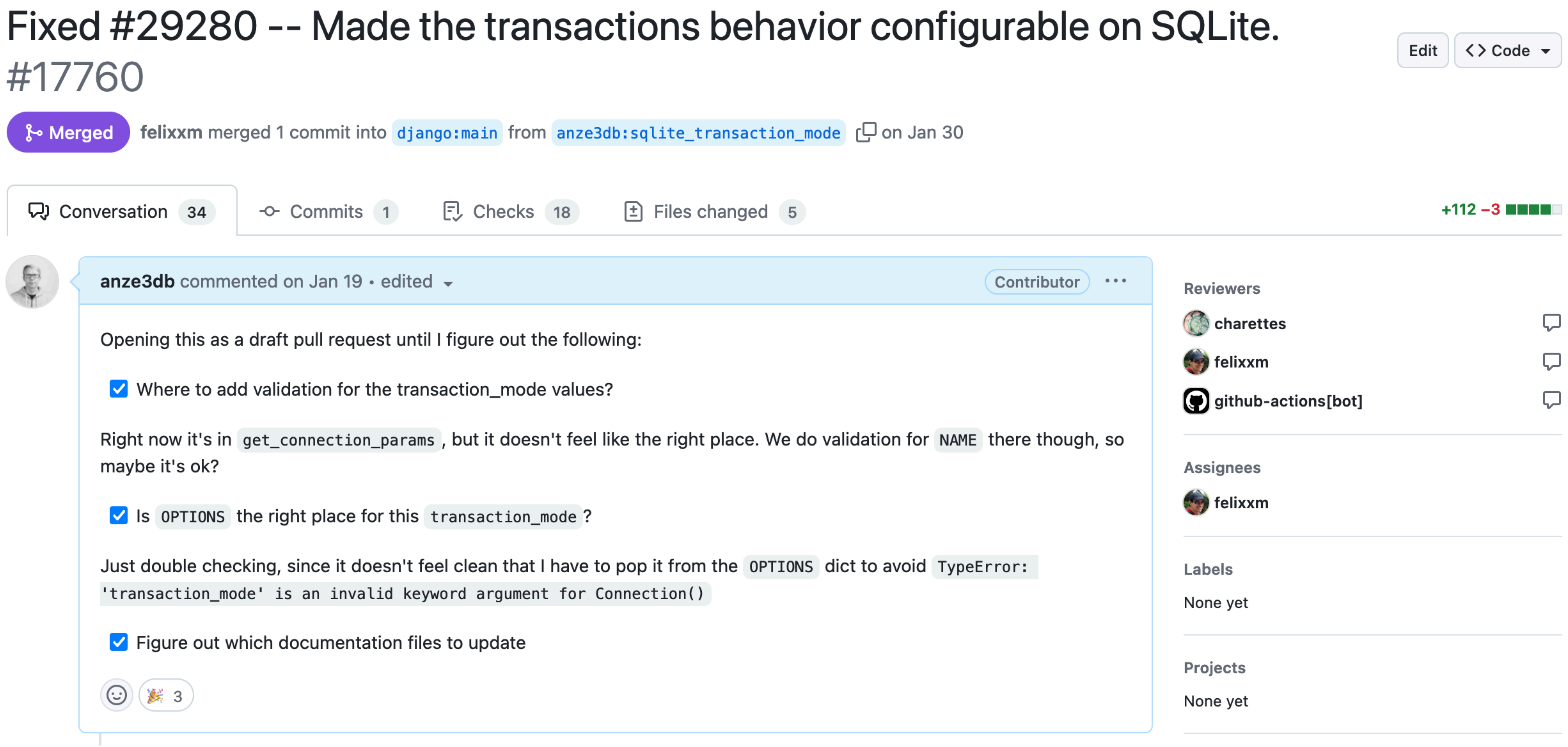
Response Times (ms)
● 50th percentile ● 95th percentile
Run #1
Run #2
Run #3
2,500
2,000
1,500
1,000
500
0
2:42:15 PM
2:42:20 PM
2:42:24 PM
2:42:28 PM
2:42:32 PM
2:43:04 PM
2:43:08 PM
2:43:12 PM
2:43:16 PM
2:43:20 PM
2:43:43 PM
2:43:47 PM
2:43:51 PM
2:43:55 PM
2:43:59 PM

Multiple db.sqlite3 files

Partnership stage

New in Django 5.1

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
        "OPTIONS": {
            "transaction_mode": "IMMEDIATE",
            "init_command": "PRAGMA journal_mode='WAL'", # <-- Enable WAL
        },
    }
}
```



Fixed #29280 -- Made the transactions behavior configurable on SQLite. #17760

Merged felixxm merged 1 commit into `django:main` from `anze3db:sqlite_transaction_mode` on Jan 30

Conversation 34 | Commits 1 | Checks 18 | Files changed 5 | +112 −3

anze3db commented on Jan 19 • edited — Contributor

Opening this as a draft pull request until I figure out the following:

- [x] Where to add validation for the transaction_mode values?

Right now it's in `get_connection_params`, but it doesn't feel like the right place. We do validation for `NAME` there though, so maybe it's ok?

- [x] Is `OPTIONS` the right place for this `transaction_mode`?

Just double checking, since it doesn't feel clean that I have to pop it from the `OPTIONS` dict to avoid `TypeError: 'transaction_mode' is an invalid keyword argument for Connection()`

- [x] Figure out which documentation files to update

🎉 3

Reviewers: charettes, felixxm, github-actions[bot]

Assignees: felixxm

Labels: None yet

Projects: None yet

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
        "OPTIONS": {
            "transaction_mode": "IMMEDIATE",
            "init_command": "PRAGMA journal_mode='WAL'; ...",
        },
    }
}
```

```
PRAGMA journal_mode = WAL;
PRAGMA synchronous = NORMAL;
PRAGMA mmap_size = 134217728; -- 128 megabytes
PRAGMA journal_size_limit = 27103364; -- 64 megabytes
PRAGMA cache_size = 2000;
```

https://fractaledmind.github.io/2024/04/15/sqlite-on-rails-the-how-and-why-of-optimal-performance/

Questions?

@anze3db

Beyond a single server

LiteFS

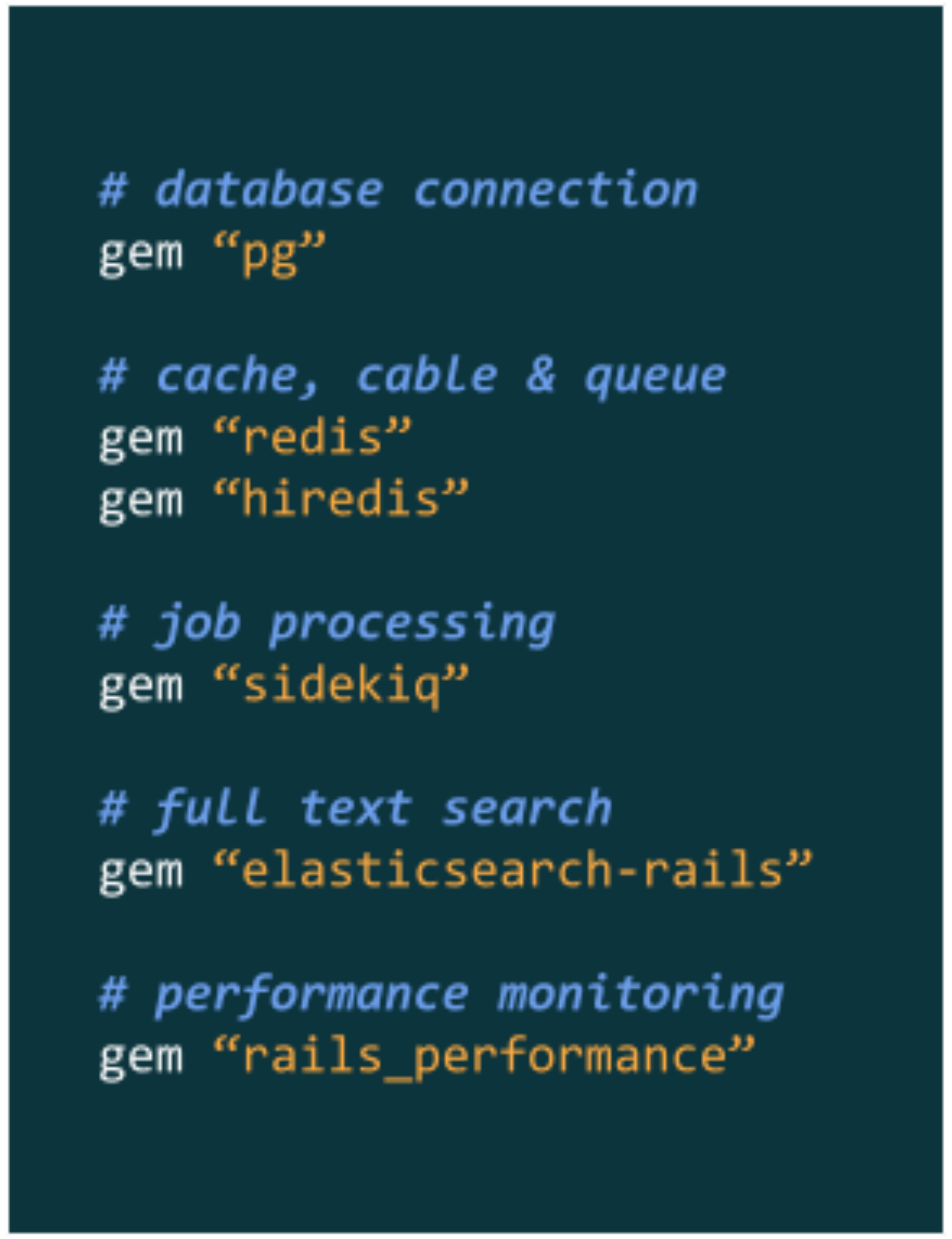
Backups

copy/paste is not safe

```
sqlite3 my.db ".backup 'my.db.bck'"
```

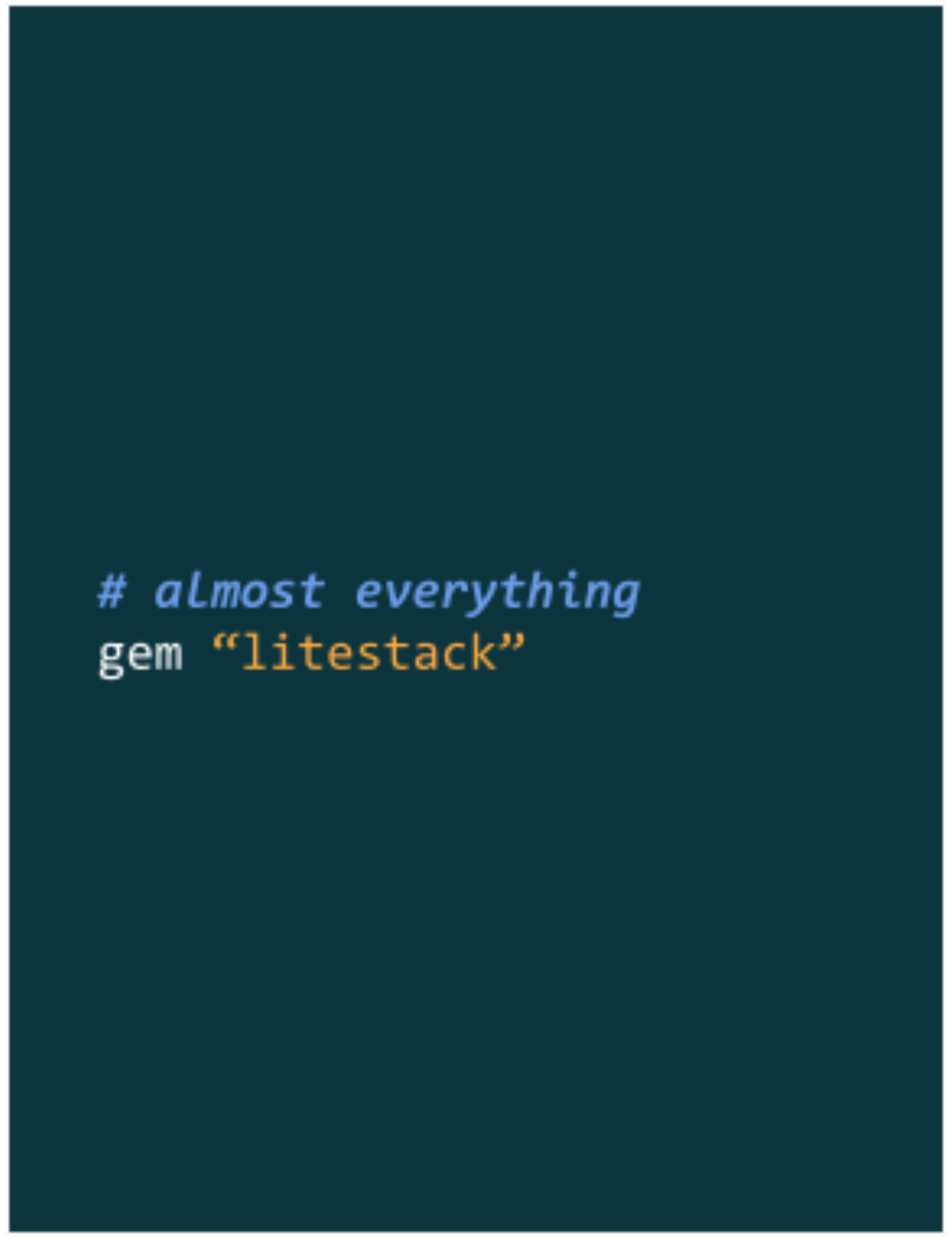
Litestream

Litestack



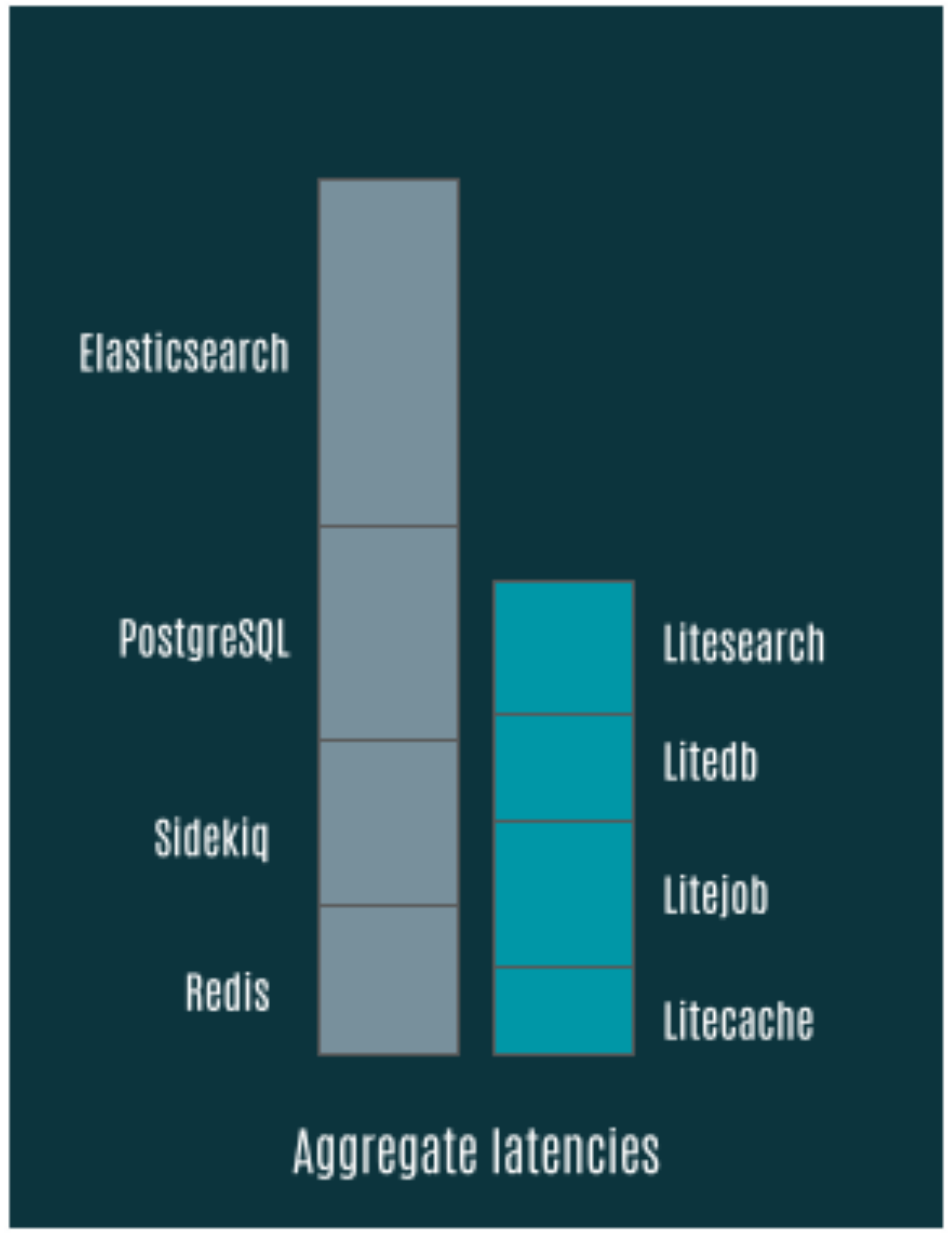
```
# database connection
gem "pg"

# cache, cable & queue
gem "redis"
gem "hiredis"

# job processing
gem "sidekiq"

# full text search
gem "elasticsearch-rails"

# performance monitoring
gem "rails_performance"
```

Turn this ..



```
# almost everything
gem "litestack"
```

.. into this ..



.. and get this!