Optimize Your Database Operations with Asynchronous Processing

Radical Rakhman Wahid

October 8th 2021
Special Region of Yogyakarta, Indonesia

Little bit about me :

- An Indonesian guy 🇮🇩
- Working day to day as a Data Scientist @ widya analytic
- Previously speaks @ GNOME.Asia Summit 2019, PyCon ID 2019, & PyCon APAC 2020.

linkedin.com/in/rakhid16

github.com/rakhid16

Outline

1. Database Operations
2. Sync and Async Processing
3. Why we need to optimize our DB Ops?
4. Asynchronous Python in DB Ops
5. Live demo
6. Conclusions

1. Database Operations

The database operations is set of operation that allowed you to do more than the CRUD (create, read, update, delete) data from the databases.

The operation expressed by a Structured Query Language (SQL).

When your program do a single query it perform the I/O bond operation.

I/O bound refers to a condition in which the time it takes to complete a computation determined principally by **the period spent waiting for I/O operations to be completed**. *(wikipedia)*

Basic Query examples :

```sql
-- Create database
CREATE DATABASE testDB;

-- Create table
CREATE TABLE Persons (
  PersonID int,
  LastName varchar(255),
  FirstName varchar(255),
  Address varchar(255),
  City varchar(255));

-- Insert data
INSERT INTO Customers
VALUES (
  'Cardinal',
  'Tom B. Erichsen',
  'Skagen 21',
  'Stavanger',
  '4006',
  'Norway');

-- Select (specific) data
SELECT Count(*) AS DistinctCountries
FROM (SELECT DISTINCT Country FROM Customers);

-- Update data
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;

-- Delete data
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

w3schools.com/sql

2. Sync and Async Processing

Chess master hosts a chess exhibition in which he plays multiple amateur players. He has two ways of conducting the exhibition: synchronously and asynchronously.
Assumptions:

- 24 opponents
- He makes each chess move in 5 seconds
- Opponents each take 55 seconds to make a move
- Games average 30 pair-moves (60 moves total)

youtube.com/watch?v=iG6fr81xHKA

2. Sync and Async Processing

Synchronous version

He plays one game at a time, never two at the same time, until the game is complete.

Each game takes (55 + 5) * 30 == 1800 seconds, or 30 minutes.

The entire exhibition takes 24 * 30 == 720 minutes, or **12 hours**.

Asynchronous version

He moves from table to table, making one move at each table.

He leaves the table and lets the opponent make their next move during the wait time.

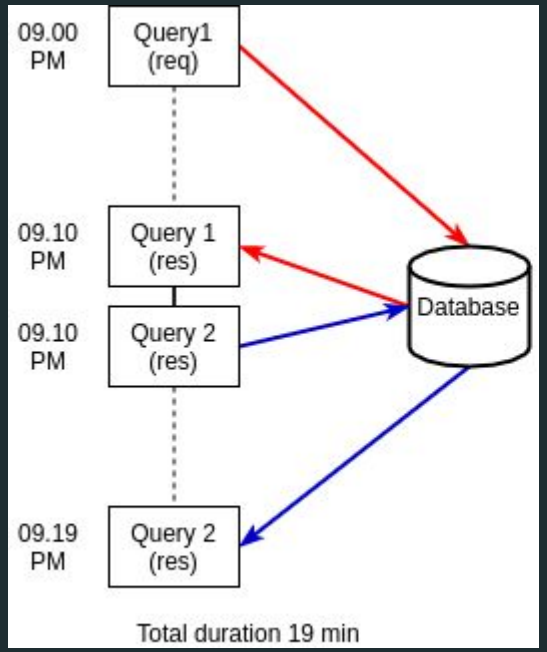
One move on all 24 games takes 24 * 5 == 120 seconds, or 2 minutes.

The entire exhibition is now cut down to 120 * 30 == 3600 seconds, or just **1 hour**.

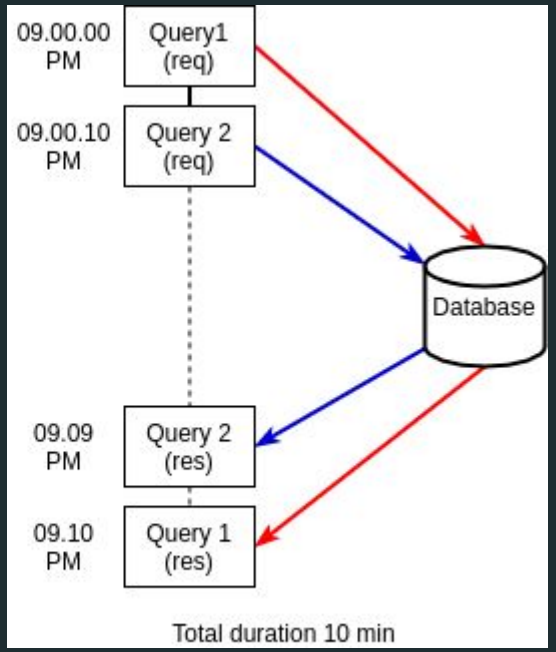
youtube.com/watch?v=iG6fr81xHKA

2. Sync and Async Processing

Synchronous Processing



Asynchronous Processing



3. Why we need to optimize our DB ops with async?

Synchronous DB ops is fine as long the I/O traffics to the DB are low/medium (depend on your use cases).

The basic idea is “How I can do the other things (in my waiting time) while I’m waiting the processes from previous task are done?”.

It can reducing the (waiting) time execution in your program if there are many I/O traffic (in a one time).

Basically it will not make your code faster out of the box, it just use the waiting time to do the other task in a single thread as well.

INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 4, ISSUE 10, OCTOBER 2015 ISSN 2277-8616

Query Optimization Techniques - Tips For Writing Efficient And Faster SQL Queries

Jean HABIMANA

Abstract: SQL statements can be used to retrieve data from any database. If you've worked with databases for any amount of time retrieving information, it's practically given that you've run into slow running queries. Sometimes the reason for the slow response time is due to the load on the system, and other times it is because the query is not written to perform as efficiently as possible which is the much more common reason. For better performance we need to use best, faster and efficient queries. This paper covers how these SQL queries can be optimized for better performance. Query optimization subject is very wide but we will try to cover the most important points. In this paper I am not focusing on, in- depth analysis of database but simple query tuning tips & tricks which can be applied to gain immediate performance gain.

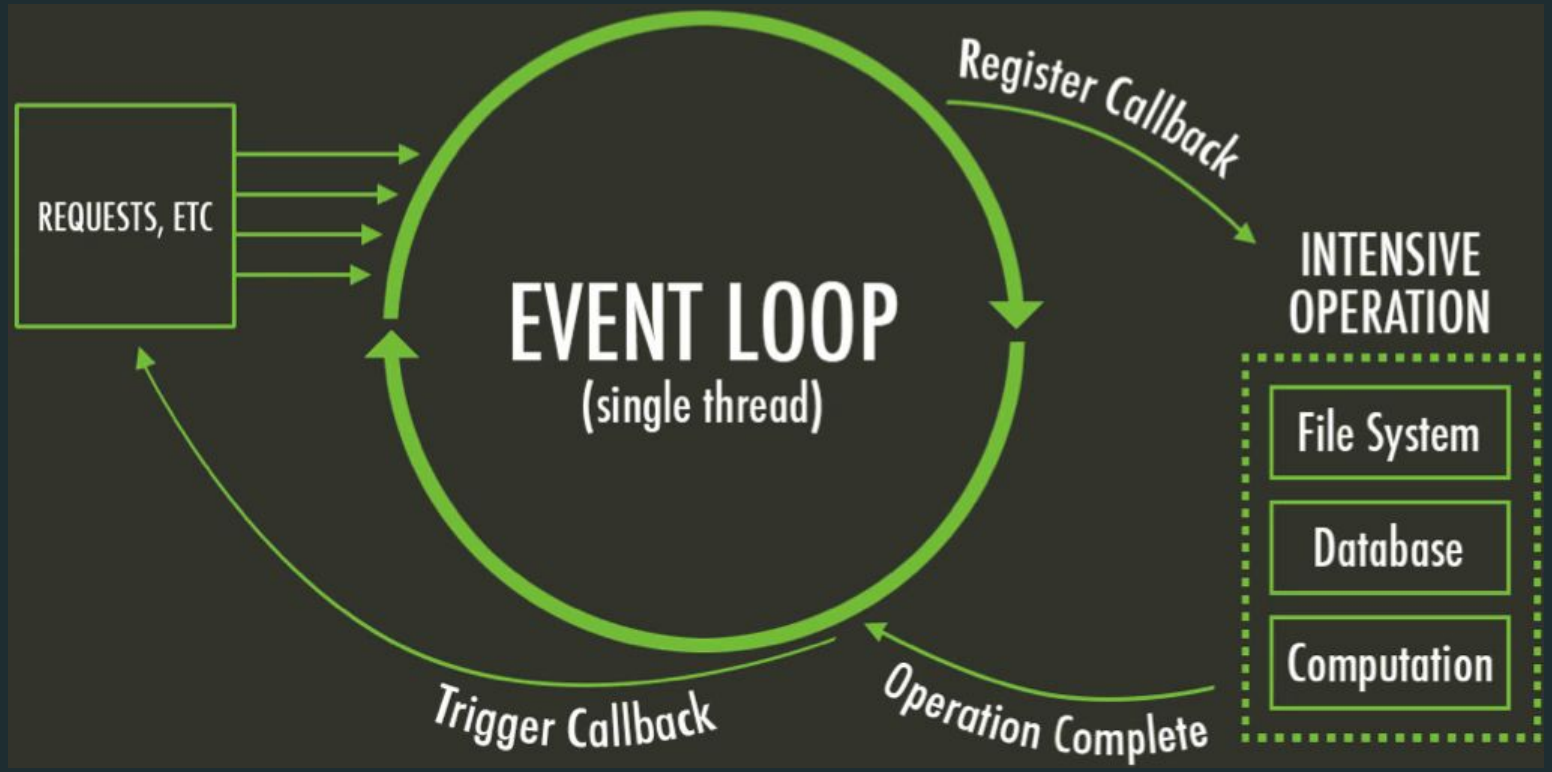
Before applying async db ops you have to make sure that your query is optimized well. Because sometimes the bottleneck is in the query itself :)

4. Asynchronous Python ~~in DB Ops~~

```python
import asyncio

# hi Coroutine
async def hi():
    print("Hello")
    await asyncio.sleep(1)
    print("World")

# main Coroutine
async def main():
    # Run the hi coroutine concurrently
    await asyncio.gather(hi(), hi(), hi())

# Run the main Courutine
# with the event loop
asyncio.run(main())
```

Coroutine, a function that have many entry points for suspending and resuming the execution.

Event loop, is responsible for scheduling (suspend or resume) one or more Coroutine(s) simultaneously.


REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete

4. Asynchronous Python in DB Ops

Libraries to connect to databases.

- asyncpg - Fast PostgreSQL Database Client Library for Python/asyncio.
- asyncpgsa - Asyncpg with sqlalchemy core support.
- aiopg - Library for accessing a PostgreSQL database.
- aiomysql - Library for accessing a MySQL database
- aioodbc - Library for accessing a ODBC databases.
- motor - The async Python driver for MongoDB.
- aioredis - aio-libs Redis client (PEP 3156).
- asyncio-redis - Redis client for Python asyncio (PEP 3156).
- aiocouchdb - CouchDB client built on top of aiohttp (asyncio).
- aioinflux - InfluxDB client built on top of aiohttp.
- aioes - Asyncio compatible driver for elasticsearch.
- peewee-async - ORM implementation based on peewee and aiopg.
- GINO - is a lightweight asynchronous Python ORM based on SQLAlchemy core, with asyncpg dialect.
- Tortoise ORM - native multi-backend ORM with Django-like API and easy relations management.
- Databases - Async database access for SQLAlchemy core, with support for PostgreSQL, MySQL, and SQLite.

github.com/timofurrer/awesome-asyncio#database-drivers

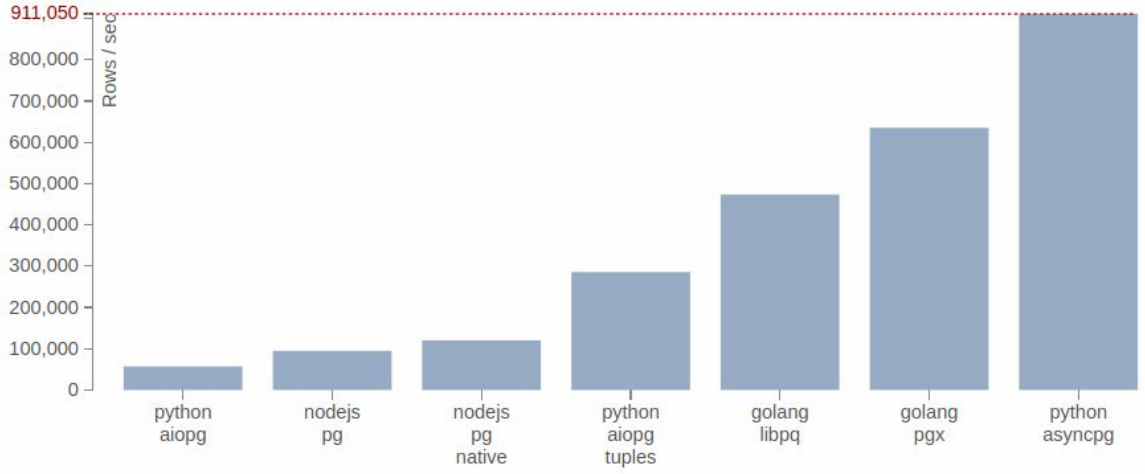
TL;DR

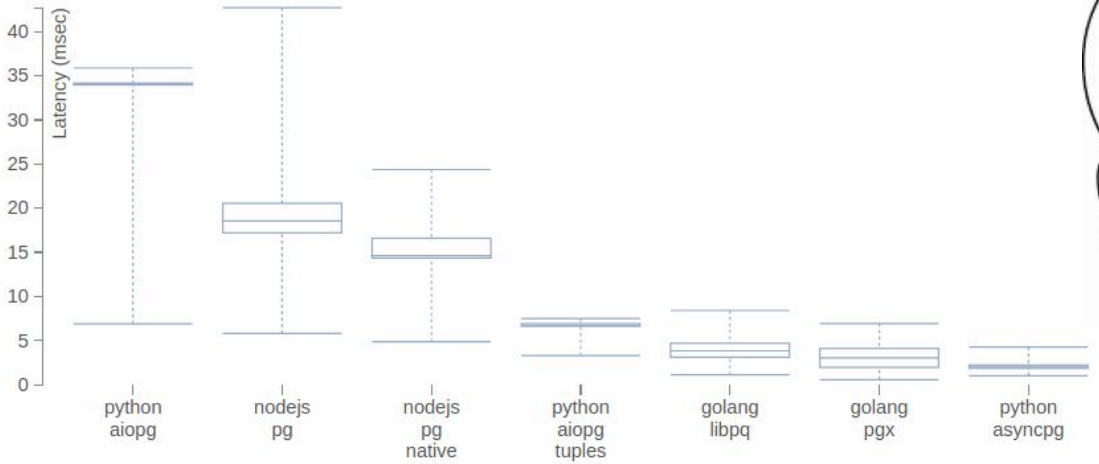
asyncpg is a new fully-featured open-source Python client library for PostgreSQL. It is built specifically for asyncio and Python 3.5 `async` / `await`. asyncpg is the fastest driver among common Python, NodeJS and Go implementations.

asyncpg

Why asyncpg?

https://magic.io/blog/asyncpg-1m-rows-from-postgres-to-python/





https://magic.io/blog/asyncpg-1m-rows-from-postgres-to-python/

5. Live demo

Experimentation :

1. Select data from one table
2. Insert the selected data sync/async to the two tables

Code :
github.com/rakhid16/pycon-zuid-afrika-2021

6. Conclusions

Synchronous DB ops is perfectly fine if there are no high I/O traffic in your program as well. You can optimize db ops with asynchronous processing, if there are any high I/O traffic.

Not all the third party libraries in Python support with Asyncio, but many of them is rapidly growing on development.

(sometimes) Easy to understand != easy to implement. If you want to use it in production you need to truly understand your own data flow, because it will not run synchronously as usual.

Thank you, from Indonesia!

Any question(s)?

Original image design by
Python Conference Indonesia 2019 committee