Philip Quan

cs6650/assignment-02

https://github.com/thephilipquan/cs6650/tree/main/assignment-02

data-model

My database of choice was PostgreSQL as it gave better results than MySQL

My table schema is as follows...

```
CREATE TABLE albums(
      id serial PRIMARY KEY,
      artist varchar(255) NOT NULL,
      title varchar(255) NOT NULL,
      year integer NOT NULL,
      image bytea NOT NULL
);
```

You can view my database schema at my repo at
https://github.com/thephilipquan/cs6650/tree/main/assignment-02.

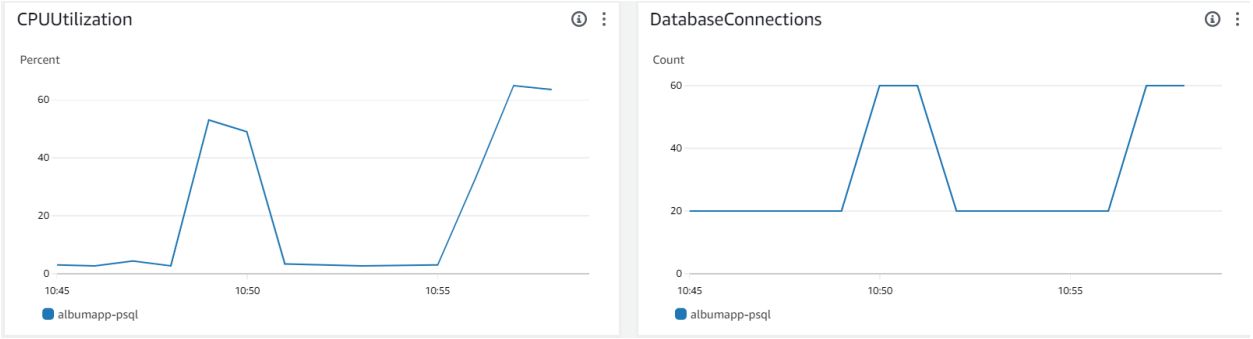
I used the stock image provided by the professor, which is around **3 to 4 kilobytes** in size.

naive-report

The following are the summarized results of my stress tests. You can view the results in more detail (post and get statistics) here.

	no-elb	with-elb
groupsize=10	walltime: 119.405s throughput: 1675 r/s	walltime: 107.266s throughput: 1865 r/s
groupsize=20	walltime: 232.379s throughput: 1721 r/s	walltime: 166.596s throughput: 2401 r/s
groupsize=30	walltime: 350.352s throughput: 1713 r/s	walltime: 236.175s throughput: 2540 r/s

Here's some screenshots of my database through the AWS Management Console and one of the EC2's terminals.




```
postgres=> select count(*) from album_app.albums;
  count
---------
 1613860
(1 row)

postgres=>
```

note

To run my code, my `Main` looks for a file called `run.conf` in the projects `resources` folder, loading it into a `RunConfiguration` that my project uses to reference. If you want to run my client yourself, your `run.conf` should be similar to this...

```
run.groupCount = 10
run.groupThreadCount = 10
run.delayInSeconds = 1
run.hostURL = http://<load-balancer-dns>:80/server_war
run.imagePath = <path/to/image.png>
run.outPrefix = <prefix-for-output-file-generated>
```

optimization

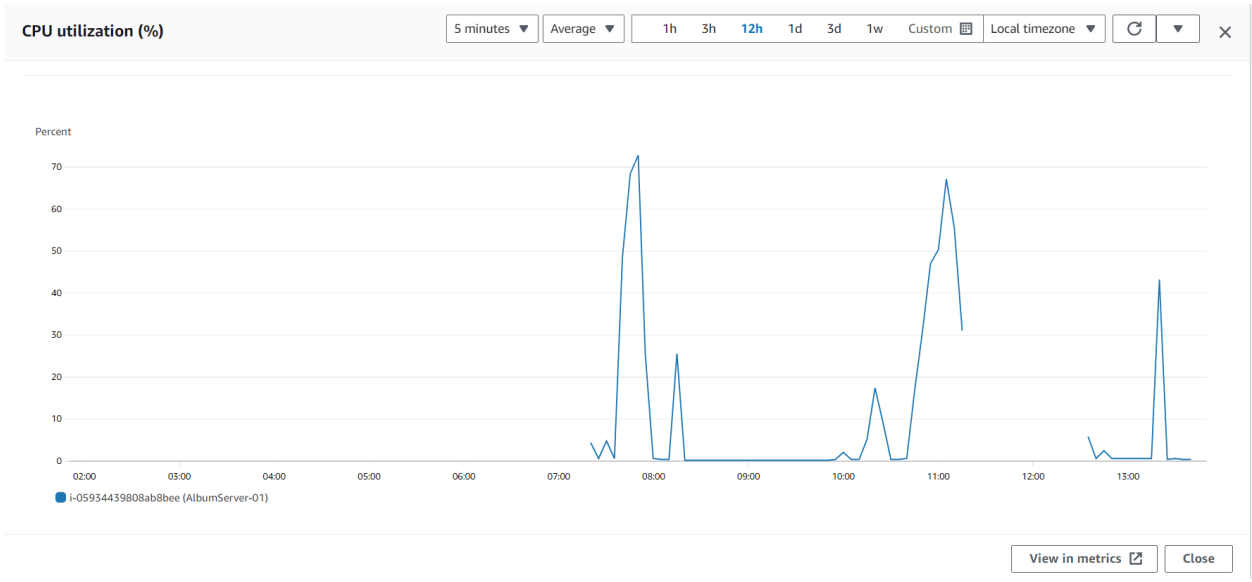
I allocated 30 database connections to each server. I did this in response to knowing that by default RDS `max_connections = 81`. I tried allocating 40 to each server but was running into errors, so I settled with 30.

Because I'm limited by the database's max connections, I decided to try increasing my server count. Consequently, I lower the connection count allocated to each server to `maxActive = 20`.

The following are my results.

	with-elb	optimized-with-elb
groupCount=30	walltime: 236.175s throughput: 2540 r/s	Wall Time: 219.872s Throughput: 2729 r/s

The result is a 7% increase in throughput, as well as 30% decrease in CPU utilization.



But as the throughput barely increased, I'm led to believe the server count is not the bottleneck. I suspect it is the database `max_connections`. But as I can't alter that with our AWS account limitations, this is the best I know of.