Graham Stockton, Jordan Weeks

Project Report

Overview: In this project, we examined the performance of a Postgres database server when 8 million records are inserted. We ran the inserts without any optimizations and timed the process as a baseline, and then tested using the connector's execute_batch() method as an optimization. We also tested increasing the hardware requirements to see their effect on insert time. In each run, a million records were inserted using the %%timeit function, which repeated the insert 8 times and calculated standard deviation and mean time.

Baseline Run: In our baseline run, we inserted 8 million records using the execute() connector method, without any optimization. We used an n1-standard-8 machine with 8 vCPUs and 32gb RAM. The run completed with an average of 4 minutes 54 seconds per loop across 8 loops, with a standard deviation of 13.8 seconds.

```
for row in csv_reader:
    cur.execute(sql, row)
    count += 1
    if count % 5000 == 0:
        conn.commit()
        print(count, "records inserted successfully into Person table")
```

```
for row in csv_reader:
    rows.append(row)
    count += 1
    if len(rows) == 5000:
        p.execute_batch(cur, sql, rows)
        conn.commit()
        print(count, "records inserted successfully into Person table")
        rows = []
```

Figure 1: Comparison of non-batch (top) and batch (bottom) record executes. Note that batch commits are placed in an array before committing as a group.

Batch Runs: For the batch runs, we used the execute_batch() method instead of individually executing every insert. We tested batch sizes of 5000, 50,000, and 500,000 records. For a batch size of 5000, %%timeit recorded an average of 1min 9s +- 483ms per million records. For 50,000, %%timeit recorded 1min 8s +- 375ms per million records, and for 50,000 %%timeit recorded an average of 1min 8s +- 467ms per loop. This was a significant improvement from the non-batch run, but increasing the batch size beyond 5000 did not seem to have a significant effect on the time.

Hardware Upgrade Runs: For the hardware upgrade runs, we upgraded our machine from an n1-standard-8 unit to a c2-standard-30 unit with 120gb RAM and 30 cpus. We ran one test with the hardware upgrade alone, and then upgraded the network bandwidth on the server in an attempt to increate insert rate. The tests used a 500,000 record batch size. The upgraded hardware (with previous bandwidth) had a time of 1min 34s +- 1.34s per million records, which stayed about the same at 1min 1s +- 1.09s when high bandwidth was used. These results seem to suggest that under current conditions, the server is more limited by hardware than by bandwidth.

Potential Future Improvements: To expand this study, it would be worthwhile to take more data points of hardware, batch size, and bandwidth to see where diminishing returns are achieved on upgrades to each. Tests should be designed to hold two of these variables constant and vary the other variable, and should be done at multiple set points for the constant variables. The end goal would be developing models of monetary cost and runtime as a function of these three variables, as well as developing heuristics for where diminishing returns may kick in for each variable.