JSON in PostgreSQL (Part 2: AlloyDB for PostgreSQL)

Some performance numbers when inserting JSON into AlloyDB for PostgreSQL

As outlined in Part 1 (JSON in PostgreSQL (Part 1: Setup and Measurement)) this blog shows the performance numbers I got with AlloyDB for PostgreSQL.

Overview of setup

The table and query is exactly the same as in Part 1. The setup for AlloyDB for PostgreSQL and the driver VM running pgbench is as follows.

AlloyDB for PostgreSQL configuration

I used the largest deployment option available in AlloyDB for PostgreSQL currently and setup an AlloyDB for PostgreSQL cluster as follows:

- Cluster specification (no read pools):

```
Version: PostgreSQL 14 compatible
Type: Highly available
```

- Primary instance specification

```
High availability: Highly available (multi-zone)
Machine type: 64 vCPU, 512 GB
```

The PostgreSQL version in AlloyDB for PostgreSQL is as follows:

```
select version();version
------------------------------------------------------------------
PostgreSQL 14.4 on x86_64-pc-linux-gnu, compiled by Debian clang
```

```
version 12.0.1, 64-bit
(1 row)
```

Driver VM specification

The VM that runs the pgbench execution is specified as follows (the largest I was allowed to create — there are larger once available):

```
Machine type: n2-highcpu-8
CPU platform: Intel Cascade Lake
Architecture: x86/64
```

Execution: inserting with pgbench

Preliminaries

Each of the three insert queries is run for 60 seconds, with 15 clients. The results are as follows (directly copied from the terminal after pgbench completed).

Empty document (size 2 bytes)

```
pgbench -n -c 38 -r -T 60 -h 10.0.0.7 -U jsondev -f writer_2.sql
json_database
Password:
transaction type: writer_2.sql
scaling factor: 1
query mode: simple
number of clients: 38
number of threads: 1
duration: 60 s
number of transactions actually processed: 1907569
latency average = 1.195 ms
tps = 31791.686004 (including connections establishing)
tps = 31800.762826 (excluding connections establishing)
statement latencies in milliseconds:
```

Document of size 1735 bytes

```
pgbench -n -c 39 -r -T 60 -h 10.0.0.7 -U jsondev -f writer_1735.sql
json_database
Password:
transaction type: writer_1735.sql
scaling factor: 1
query mode: simple
number of clients: 39
number of threads: 1
duration: 60 s
number of transactions actually processed: 1704598
latency average = 1.373 ms
tps = 28409.254110 (including connections establishing)
tps = 28417.272609 (excluding connections establishing)
statement latencies in milliseconds:
         1.310  INSERT INTO json_schema.json_document
(document_identifier, time_inserted,
```

Document of size 4503 bytes

```
pgbench -n -c 39 -r -T 60 -h 10.0.0.7 -U jsondev -f writer_4503.sql
json_database
Password:
transaction type: writer_4503.sql
scaling factor: 1
query mode: simple
number of clients: 39
number of threads: 1
duration: 60 s
number of transactions actually processed: 1689114
latency average = 1.385 ms
tps = 28151.091217 (including connections establishing)
tps = 28159.040388 (excluding connections establishing)
statement latencies in milliseconds:
         1.302  INSERT INTO json_schema.json_document
(document_identifier, time_inserted,
```

Execution — Summary

In summary, the larger the document, the less inserts per second can be achieved. That is expected as the binary representation `JSONB` requires parsing and conversation effort that increases with the size of the document.

- TPS for 2 bytes: 31800

- TPS for 1735 bytes: 28417
- TPS for 4503 bytes: 28159

Summary

Obviously a production system like AlloyDB for PostgreSQL has significant better performance compared to my laptop. The system was not tuned and used with its default configuration. In addition, larger driver VMs are available, but not accessible to me. Still, these performance numbers are one data point to get a rough idea on what is possible.