JSON in PostgreSQL (Part 1: Setup and Measurement)

Observations on JSONB insert elasticity

I was wondering about the JSON document insert performance in PostgreSQL and the extent it varies with document size.

Overview

PostgreSQL supports JSON as a data type and I was curious how insert performance changes with the size of a JSON document being inserted. To get an impression to the extent insert performance changes with document size, I am inserting three different sizes of documents and measure performance using pgbench.

Implementation: table and operations

The following shows the table definition of the table that receives the insert statements.

Schema

The schema is a single table, with a primary key (`UUID`), the time it was inserted (`TIMESTAMP`) and the document itself (`JSONB`).

```
CREATE TABLE IF NOT EXISTS json_schema.json_document
(
    document_identifier UUID PRIMARY KEY,
    time_inserted       TIMESTAMP,
    document            JSONB
);
```

JSON vs. JSONB

PostgreSQL has two JSON types: `JSON` and `JSONB` (https://www.postgresql.org/docs/current/datatype-json.html). As outlined in the PostgreSQL documentation, `JSONB` is a binary representation of the input document and makes access more efficient. The documentation recommends this type and so I chose it for my tests.

Query

The query is a simple insert query and has three variations:

- Insert an empty document `{}` (2 bytes)
- Insert a document with 1735 bytes (15 levels deep)
- Insert a document with 4503 bytes (31 levels deep)

This is the query inserting an empty document:

```
INSERT INTO json_schema.json_document
   (document_identifier, time_inserted, document)
VALUES (gen_random_uuid(), current_timestamp, '{}');
```

Machine and PostgreSQL database

The pgbench runs are executed on the following machine:

```
OS Name Microsoft Windows 11 Pro

Version 10.0.22000 Build 22000

Processor Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, 1992 Mhz, 4 Core(s),
8 Logical Processor(s)

Installed Physical Memory (RAM) 32.0 GB

Disk Model Samsung SSD 970 EVO Plus 1TB
```

The database is a standard installation without configuration changes:

```
select version()

PostgreSQL 14.5, compiled by Visual C++ build 1914, 64-bit
```

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 15 -r -T 60 -h 127.0.0.1 -U jsondev -f writer_2.sql
json_database
Password:
pgbench (14.5)
transaction type: writer_2.sql
scaling factor: 1
query mode: simple
number of clients: 15
number of threads: 1
duration: 60 s
number of transactions actually processed: 1208245
latency average = 0.738 ms
initial connection time = 547.437 ms
tps = 20320.507487 (without initial connection time)
statement latencies in milliseconds:
         0.575  INSERT INTO json_schema.json_document
(document_identifier, time_inserted,
```

Document of size 1735 bytes

```
pgbench -n -c 15 -r -T 60 -h 127.0.0.1 -U jsondev -f writer_1735.sql
json_database
Password:
pgbench (14.5)
transaction type: writer_1735.sql
scaling factor: 1
query mode: simple
number of clients: 15
number of threads: 1
duration: 60 s
number of transactions actually processed: 905106
latency average = 0.984 ms
initial connection time = 656.384 ms
tps = 15249.512363 (without initial connection time)
statement latencies in milliseconds:
         0.779  INSERT INTO json_schema.json_document
(document_identifier, time_inserted,
```

Document of size 4503 bytes

```
pgbench -n -c 15 -r -T 60 -h 127.0.0.1 -U jsondev -f writer_4503.sql
json_database
Password:
pgbench (14.5)
transaction type: writer_4503.sql
scaling factor: 1
query mode: simple
number of clients: 15
number of threads: 1
duration: 60 s
number of transactions actually processed: 698619
latency average = 1.277 ms
initial connection time = 543.798 ms
tps = 11748.300928 (without initial connection time)
statement latencies in milliseconds:
         0.955  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: 20320
- TPS for 1735 bytes: 15249
- TPS for 4503 bytes: 11748

Summary

As expected, when inserting `JSONB` documents into a table in PostgreSQL, larger documents take more time to insert. The results shown here are on a laptop, and measurements on a PostgreSQL production system follow in a separate blog.