CS 487/587 Database Implementation
Winter 2021
Database Benchmarking Project - Part 2
Team members: Deep Patel Goutham Podili

In the experiment, we are evaluating PostgreSQL with different memory options and query planner configuration in Postgres server on Google Cloud VM and local computer.

SYSTEM CONFIGURATION:

The memory level configuration of the system is mentioned below.

Machine type	RAM	OS Image	Postgres version
Google VMs	2 GB	Ubuntu-2004	Psql 12.6
Local	16GB	Windows10	Psql 13.1

SYSTEM RESEARCH:

Available postgres memory options and query planner.

`shared_bufferes(integer)`	● Sets the amount of memory the database server uses for shared memory buffers. ● Default is typically 128 MB.
`work_mem(integer)`	● Specifies the amount of memory to be used by internal sort operations and hash tables before writing to temporary disk files. ● Default is 4MB.
`temp_buffers(integer)`	● Sets the maximum number of temporary buffers used by each database session. ● Default is 8MB.
`enable_seqscan(boolean)`	● It is impossible to suppress sequential scans entirely, but turning this variable

	off discourages the planner from using one if there are other methods available. ● Default is on
`enable_hashjoin(boolean)`	● Enables or disables the query planner's use of hash-join plan types. ● Default is on.

PERFORMANCE EXPERIMENTS:

1) **Testing the 10% rule of thumb**
    a) This test explores when it is good to use an unclustered index vs not using an index vs using a clustered index. We will also vary the selectivity of queries like 5%, 25%, and 50%.
    b) We are using relations with size of 10,000 tuples(TENKTUP1) and 100,000 tuples(HUNDREDKTUP).
    c) We are using Wisconsin benchmark queries 2, 4 and 6
        i) INSERT INTO TMP SELECT * FROM TENKTUP1 WHERE unique2 BETWEEN 792 AND 1791 **(no index)**
        ii) INSERT INTO TMP SELECT * FROM TENKTUP1 WHERE unique2 BETWEEN 792 AND 1791 **(clustered index)**
        iii) INSERT INTO TMP SELECT * FROM TENKTUP1 WHERE unique1 BETWEEN 792 AND 1791 **(non-clustered index)**
    d) No parameters are changed in this test.
    e) We are expecting that for queries which has selectivity less than ten percent attributes which have no index will perform better than attributes with index.

2) **Testing work_mem**
    a) This test examines the relationship between the sort operations and work_mem. We will set work_mem to different sizes.
    b) We are using relations with size of 10,000 tuples(TENKTUP1) and 100,000 tuples(HUNDREDKTUP).
    c) The queries we are going to execute for this experiment are
        i) SELECT unique1, strin3 FROM TENKTUP1 ORDER BY string3.
        ii) INSERT INTO TMP SELECT * FROM ONEKTUP, TENKTUP1 WHERE (ONEKTUP.unique1 = TENKTUP1.unique1) AND (TENKTUP1.unique1 = TENKTUP2.unique1) AND (TENKTUP1.unique1 < 1000)
    d) We will change the work_mem parameter in this test.
    e) As the sorting uses work memory we are expecting to see an increase in the performance of the query when increasing the work_mem.

3) **Testing shared_buffer parameter**

a) This test illustrates what size of the shared buffer performs well on given queries. We will be testing with 25 MB, 128 MB, and 256 MB.

b) We consider table size of 10,000 tuples(TENKTUP1), 100,000 tuples(HUNDREDKTUP), and 1,000,000 tuples(ONEMILLIONTUPLES)

c) Queries:

i) Join query
SELECT onemilliontuples.unique1 , hundredktuples.unique1
FROM onemilliontuples, hundredktuples
WHERE onemilliontuples.string3 = hundredktuple.string3;

ii) Aggregate query
SELECT COUNT(DISTINCT unique1)
FROM TENKTUP1

d) We will be testing on 25 MB, 128 MB, and 256 MB. Total RAM size of Postgres server is 2 GB

e) It is recommended to set 25% of RAM for buffer size.
When performing join and aggregate queries with different memory sizes, relatively low memory will provide better results.

4) **Testing temp_buffers parameter**

a) This test demonstrates what size of temp buffers is well suited for sort and hash table joins operations queries. We will be testing 4 MB, 8 MB, and 16 MB.

b) We consider table size of 10,000 tuples(TENKTUP1), 100,000 tuples(HUNDREDKTUP), and 1,000,000 tuples(ONEMILLIONTUPLES)

c) Queries:

i) SELECT TENKTUP1.unique1, HUNDREDKTUP.unique1
FROM TENKTUP, HUNDREDKTUP.
WHERE TENKTUP1.string3 = HUNDREDKTUP..string3;

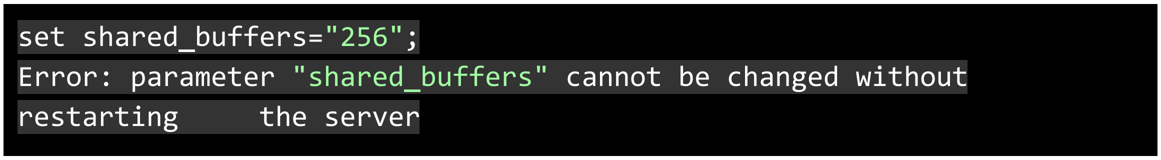
ii) SELECT HUNDREDKTUP.unique1 , ONEMILLIONTUPLES.unique1
FROM HUNDREDKTUP, ONEMILLIONTUPLES
WHERE ONEMILLIONTUPLES.string3 = HUNDREDKTUP.string3;

d) The experiment will be held on 4 MB, 8 MB, and 16 MB temp_buffer size

e) Having a bigger temp_buffer size help for sort and hash table operations.

LESSON LEARNED:

1) Unable to set shared_buffers size on running psql using command



```
set shared_buffers="256";
Error: parameter "shared_buffers" cannot be changed without
restarting     the server
```

Since shared buffers is heart of data base system, it only possible to set up from config file and need to restart the server after changing the size.

REFERENCES

https://www.postgresql.org/docs/9.6/runtime-config-resource.html