Members:

Harie Vashini

Zeba Khan

Database System Implementation

Project Part 2

Benchmark Design

System : PostgreSQL (Option 1. Evaluate a single System)

System Research :

1. **work_mem (integer)**
   This is a parameter that is used to specify the measure of memory to be used by internal sort operations and hash table creations. The default size is 4 MB. Sort operations are used for ORDER BY, DISTINCT and Merge join operations. Hash tables are used in hash joins and hash based aggregates. This is per operation, not per statement or per back-end, so a single complex query can use many times this amount of memory.There can be multiple queries running parallely. These parallel transactions in large work memories can cause performance issues.

2. **enable_hashjoin (boolean)**
   Most commonly and frequently used amongst all join algorithms is the Hash join. It is used for equi joins producing efficient results at low cost. It builds an in-memory hash table on the inner table(smallest amongst the two tables) and scans the outer table for a match.The cost depends on the table fitting in the memory. *enable_hashjoin* parameter enables/disables the query's use of hash join. It is on by default.

3. **enable_indexscan (boolean)**
   Index scan performs a B-tree traversal to find all matching entries and fetch the corresponding data. It enables/disables the query's use of index scan plan types. It is on by default.

4. **enable_seqscan (boolean)**
   It is the most commonly used type of scan for extracting data from a disk. It sequentially scans pages and always returns data. It can't be turned off totally. Disabling this parameter enables a use of other types of scans such as index and bitmap. It is on by default.

5. **enable_hashagg (boolean)**
   Enables/disables the query's use of hash aggregate. It is on by default. Disabling it will avoid the use of hash aggregation and enables the usage of other available options such as group aggregation.

Performance Experiment Design :

Experiment 1

- **What performance issue are you testing?**
  In this experiment we are going to test the performance and execution time of work_mem parameter by changing its default value.

- **What data sets will you include in the test? Will you use a single size data set or will you use multiple data sets of different sizes?**
  We will be using a both TENKTUP1 dataset with 10000 tuples and ONEKTUP dataset with 1000 tuples.

- **What queries will you run? (provide the SQL)**
  SELECT unique1, unique2, stringu1 FROM TENKTUP1 ORDER BY stringu1 DESC;

  SELECT O.unique2 as 1k_unique2 FROM ONEKTUP O ,TENKTUP1 T WHERE O.unique2=T.unique2 ORDER BY T.unique3;

- **If you are doing Option 2, what parameters will you use for this test? How will the parameters be set/varied?**
  work_mem (Integer) parameter
  SET work_mem= 4MB
  SET work_mem= 128MB

- **What results do you expect to get?**
  By default work_mem is set to 4MB, we plan to increase the work_mem parameter value and expect it to result in improved performance i.e. less execution time.

Experiment 2

- **What performance issue are you testing?**
  In this experiment we are going to explore the 10% thumb rule, by testing it with queries having index vs no index and also queries having non-clustered and clustered index.

- **What data sets will you include in the test? Will you use a single size data set or will you use multiple data sets of different sizes?**
  We will be using a single dataset TENKTUP1 with 10000 tuples.

- **What queries will you run? (provide the SQL)**
  With clustered index and no index
  INSERT INTO TMP
  SELECT * FROM TENKTUP1
  WHERE unique2 BETWEEN 792 AND 1791

  With non-clustered index
  INSERT INTO TMP
  SELECT * FROM TENKTUP1
  WHERE unique1 BETWEEN 792 AND 1791

- **If you are doing Option 2, what parameters will you use for this test? How will the parameters be set/varied?**
  We will be using two parameters
  enable_indexscan (boolean)
  SET enable_indexscan =On
  SET enable_indexscan =Off

  enable_seqscan (boolean)
  SET enable_seqscan =On
  SET enable_seqscan =Off

- **What results do you expect to get?**
  Queries with no index will have the maximum execution time followed by non-clustered index and then by clustered index.

Experiment 3

- **What performance issue are you testing?**
  We will be testing the performance of join algorithms.

- **What data sets will you include in the test? Will you use a single size data set or will you use multiple data sets of different sizes?**
  We will be using two datasets, TENKTUP1 and TENKTUP2 with 10000 tuples.

- **What queries will you run? (provide the SQL)**

  ```
  INSERT INTO TMP
  SELECT * FROM TENKTUP1, TENKTUP2
  WHERE (TENKTUP1.unique2 = TENKTUP2.unique2)
  AND (TENKTUP2.unique2 < 1000)
  ```

  This is a joinAselB query, the table TENKTUP2 has only 10% selectivity.

- **If you are doing Option 2, what parameters will you use for this test? How will the parameters be set/varied?**
  enable_hashjoin (boolean) parameter

  ```
  SET enable_hashjoin =On
  SET enable_hashjoin =Off
  ```

- **What results do you expect to get?**
  TENKTUP2 table being the smaller among the two based on the selectivity, will fit in the memory. It chooses hash join for executing the query resulting with best execution time. When the hash join is disabled the query optimizer is forced to use another join say, merge join thus increasing the execution time.

Experiment 4

- **What performance issue are you testing?**
  In this we will be exploring how aggregation performs when enabling and disabling the parameter enable_hashagg (boolean) with a small value of work_mem parameter.

- **What data sets will you include in the test? Will you use a single size data set or will you use multiple data sets of different sizes?**
  We will be using a TENKTUP2 dataset with 10000 tuples.

- **What queries will you run? (provide the SQL)**
  SELECT MIN (unique2) FROM TENKTUP2
  Minimum aggregate function

  SELECT COUNT (unique3) FROM TENKTUP2
  WHERE unique3 < 1000
  GROUP BY onePercent < 10;
  Count Aggregate function with 10 partitions

- **If you are doing Option 2, what parameters will you use for this test? How will the parameters be set/varied?**
  We will be using two parameters
  enable_hashagg (boolean) parameter
  SET enable_hashagg =On
  SET enable_hashagg =Off

  work_mem (Integer) parameter
  SET work_mem= 15MB

- **What results do you expect to get?**
  The queries take less time to execute when they use hash aggregation (when enable_hashagg parameter is set On). When it is set Off, they would take more execution time since it becomes slow.

Lessons Learned :

1. During this part of the project, the focus was more on various Postgres parameters and their optimization techniques. It helped us in understanding how the configuration of parameters can affect the behavior of database systems.
2. Most of the time Postgres would choose the most cost effective query plan however by changing them according to the user we learnt how the performance varied.
3. Initial understanding of queries with different parameters was time consuming. However we were able to learn on how different parameters can be used in queries that would impact on the performance and execution time. We learnt how some queries helped in better performance and some made it terrible.
4. Also through experiments we got to know hash algorithms with aggregates performed better.
5. Configuring the work_mem to different sizes also gave a better understanding of execution times.
6. Most of the time, when we executed a query twice, we were unable to figure out if the system returned the cached result or from the disk. This was one major concern being data retrieved from cached results yielding better performance.