CS487

Project Part 2

Jordan Cantrell

1.

System Choice

In this benchmark, we investigate a single DBMS system, MySql. This system was chosen for investigation due to its ease of use and free pricetag, and to investigate the feasability of running a database on consumer hardware, at home.

2.

System Research

5 (mysql) parameters from Memory or Query Planner options, and describe them.

1. innodb_buffer_pool_size (default value is 128M) This variable controls the size (in bytes) of the buffer pool, which is where table and index data are cached.
2. block_nested_loop (default is on) This flag controls the use of the Block-Nested Loop algorithm to join relations.
3. semijoin (default on) This flag controls whether semijoins are used during a query.
4. condition_fanout_filter (default on) This flag allows the optimizer to use information about conditions on the rows being selected to estimate the number of row combinations during a join, and choose an appropriate execution plan.
5. skip_scan (default on) This flag toggles the use of the Skip Scan method to more more efficiently select queries by scanning through the distinct values of the index, and then doing a subrange scan on the other attribute.

3.

Performance experiment Design

1. Experiment 1: Different relation sizes
   i. This experiment investigates the performance difference of different-sized relations in Mysql.
   ii. This experiment involves different relations of various sizes, from 10k tuples and 100k tuples to 100m tuples.
   iii. This query involves running queries 1 and 2.
   iv. This experiment only varies the sizes of the relations, and does not involve changing the values of any options or parameters.
   v. I expect to see little slow down, until the tables get too large to fit entirely in memory and then see a significant slowdown.
2. Experiment 2: condition_fanout_filter
   i. We examine the affect of disabling the condition fanout filter which helps estimate the selectivity of a where clause and uses that to put smaller (more selective) tables early in execution to prevent the number of row combinations from exploding in magnitude.
   ii. This experiment will be run over the 100k tuple data set.
   iii. We will use a modified version of query 10 from the Wisconsin benchmark which includes a condition comparing a field to a constant value. This is to ensure condition filtering will apply to the query.
   iv. This experiment turns off the “condition_fanout_filter” flag.
   v. I expect turning off condition filter will make some queries with constants in the where clause to run more slowly.
3. Experiment 3: innodb_buffer_pool_size (default value is 128M)
   i. In this experiment, we investigate how the size of the buffer pool affects query run time.
   ii. We will run this experiment over a 100k tuple data set, which follows the general oultline of the data used for the Wisconsin benchmark.
   iii. In this experiment, we use queries 6 and 9.
   iv. The buffer pool size is controlled by the parameter innodb_buffer_pool_size, so we will run the queries for this experiment with a large and small value for that option.
   v. I expect performance to be faster for selections with a larger pool size.
4. Experiment 4: block_nested_loop
   i. We examine the affect of different join algorithms on database performance. Specifically, the effect of disallowing the nested block loop algorithm for joins.
   ii. This experiment will be run over the 100k tuple data set.
   iii. We will use queries 10 and 11 from the Wisconsin benchmark to test if disabling block_nested_loop will affect join queries.

iv. This experiment turns off the “block_nested_loop” flag.

v. I expect turning off nested loops will make the join queries run slowly