CS 487/587 Database Implementation

Theodore Vixay & Daniel Lee

Spring 2019

Database Benchmarking Project - Part II

Project Goals (student perspective):

- **Experience running performance experiments on 1-2 data management systems, including on relational database management system**
- **Increased understanding of database system performance and the impacts of database implementation on performance**
- **Programming experience**
- **Potential for experience with cloud-based systems**

Project Part II: Benchmark Design
Due: Wednesday May 15, midnight (end of the day May 15), D2L

1. **Which option / system (s) you will be working with and why you chose it (them) (20 pts)**
   Option 1
   Azure: SQL Databases:
   A popular database server service with Microsoft.

   Connectivity between your Microsoft Azure SQL Database and the Azure Internet gateway is guaranteed at least 99.99 percent of the time, regardless of your service tier. 0.01% is reserved for patches, upgrades, and failovers.

   PostgreSQL:
   We chose PostgreSQL because we are more familiar with this system and because it's one of the most used systems.

2. **System Research (30 pts)**
   a. **Option 1: Include research on your selected system**
      i. **Describe types of indices, types of join algorithms, buffer pool size/structure, mechanisms for measuring query execution time**

Azure: SQL Database:
Index Types: Azure SQL Database provides several index types, including:

- B-tree
- Hash
- Memory-Optimized Nonclustered
- Clustered
- Nonclustered
- Unique
- Columnstore

- Spatial
- XML
- Full Text

Join Types:

- Nested loop join
- Merge join
- Hash join
    - In-Memory
    - Grace
    - Recursive

Buffer Pool Size:

The buffer pool size is variable by the use of scale elastic pool resources. The

Features: Extra functionality that Azure Provides on top of the SQL database functions

- Automatic Tuning

    Automatically done using artificial intelligence that can optimize the database to run queries faster. Can be turned on or off in the Azure Portal when selecting the target SQL Server.

    - CREATE INDEX:

        Identifies indexes that may improve performance of your workload, creates indexes, and automatically verifies that performance of queries has improved.

    - DROP INDEX:

        Identifies redundant and duplicate indexes daily, except for unique indexes, and indexes that were not used for a long time (>90 days). This is not compatible with applications using partition switching and index hints.

    - FORCE LAST GOOD PLAN:

        This is automatic plan correction which identifies SQL queries using execution plan that is slower than the previous good plan, and queries using the last known good plan instead of the regressed plan. This happens when a sub-optimal plan is used and forces the SQL server to use the last known good plan if regression is detected.

PostgreSQL:

Index Types: PostgreSQL provides several index types, including:

- B-tree
- Hash
- GiST
- SP-GiST and
- GIN

Join Types:

- Nested loop join
- Merge join
- Hash join

Buffer Pool Size:

Each buffer pool slot size is 8KB, which is equal to the size of a page. Thus, each slot can store an entire page. The shared_buffers is default at 128MB, but can be increased to 40% of the system's RAM.

Resource: https://www.postgresql.org/docs/9.1/runtime-config-resource.html

3. **Performance Experiment Design (80 pts)**

Nested Queries

1. Compare how Azure and Psql handle nested queries, is there a threshold when one system performs better?
2. Multiple sizes. We will start with the onektup table then scale up to the tenktup, hundredktup, etc. We will also be scaling our nested queries as well as scaling the table sizes.
3. Example:

```
SELECT *
FROM onektup
WHERE unique1 IN (SELECT oddOnePercent
    FROM onektup
    WHERE oddOnePercent > 100) ;
```

4. When scaling the queries and table sizes up, we expect the times to run slower. We also expect systems that have better join algorithms to detect and rearrange the query operations in a more efficient way.

Join Queries/Algorithms

1. Compare how Azure and Psql handle Join Queries, is there a threshold when one system performs better?
2. Multiple sizes 100,1000, 10000, 100000 etc. We will start with the onektup table then scale up to the 100, onektup, tenktup, hundredktup, etc.
3. Example Query: INNER JOIN, LEFT/RIGHT JOIN, FULL OUTER JOIN

```
SELECT h1.unique1
FROM tenktup1 AS h1
INNER JOIN tenktup2 AS h2 ON
h1.unique1= h2.unique2;
```

4. The scale of the two tables will vary in size with one another. A small table on a large table will run faster than a large table on a small table due to the buffer size that allows a smaller table to stay while comparing to the larger table. Azure will run faster as they have a flexible buffer pool.

Sorting

1. Compare how Azure and Psql handle Queries that require sorting, is there a threshold when one system performs better?
2. Multiple sizes 100,1000, 10000, 100000 etc. We will start with the onektup table then scale up to the 100, onektup, tenktup, hundredktup, etc.
3. Example Query: SELECT DISTINCT, GROUP BY, ORDER BY

```
SELECT DISTINCT unique1
FROM tenktup1 ;

SELECT twenty
FROM tenktup1
WHERE unique1 > 50
GROUP BY twenty
ORDER BY twenty;
```

4. The scale of the table size will vary but as the required buffer size is met is when the query runtime will dramatically increase in time as the number of files that need to be read and written to disk increases as the buffer can no longer hold all values that are required for sorting. The system with a larger buffer size will be able to perform better but only until their buffer size limit is also reached. This will also test Azure's elastic pool's resources.

Force Index Autotuning

1. Compare how Azure and Psql handle Queries that are index use friendly and we will create an index on unique1 for Psql and turn on auto tune for Azure and see how many runs it takes for Azure to comparatively perform. This is because AZURE has autotune which automatically drops and creates indexes based on queries that readily use the same portions of data. We will constantly it the databases with queries that use the unique1 index.

2. Single size of 100000 tuples. We will use queries that will use the index on unique1.

3. Example Query

```
Psql Query Index Create:
CREATE INDEX idx_unique1 ON tenktup1(unique1);

Query Runs:

SELECT *
FROM tenktup
WHERE unique1 = 1;


SELECT *
```

```
FROM tenktup
WHERE unique1 = 2;


SELECT *
FROM tenktup
WHERE unique1 = 3;
```

Continuously increment and record the query times.

```
SELECT *
FROM tenktup
WHERE unique1 = max; // max is the max tuple size
```

4. The scale of the table size will vary but as the required buffer size is met is when the query runtime will dramatically increase in time as the number of files that need to be read and written to disk increases as the buffer can no longer hold all values that are required for sorting. The system with a larger buffer size will be able to perform better but only until their buffer size limit is also reached. This will also test AZURE's elastic pool's resources.

4. Lessons Learned: Include lessons learned or issues encountered (20 pts)

Azure: SQL Database:

Costs increment daily even without a created table albeit slowly between 10-15 cents a day.

Inconsistent firewall recognition for IP address and constant changing of firewall settings to allow IP addresses. This issue is common for remote non-hardwire connected devices.

System uses T-SQL instead of SQL. Insertion of data needs to be done using a query and has not been discovered to be able to use CSV files to import data. A python program was written to generate queries to insert tuples in T-SQL query format. This allows the query to simply copy and pasted to upload the data into the table in the server.

PostgreSQL:

PostgreSQL is highly customizable and the documentation online is wonderful. In order to perform a hot/cold cache performance test, I will need to download a version of psql onto my local machine to be able to flush out the cache and accurately do cold cache performance tests.