Benchmark

Database Administration
Lab Guide 1

2019/2020

Consider a simplified invoice processing system, assuming that each transaction sells one item of a single product. Use the following tables:

Client: Id, Name, Address, Data.

Product: Id, Description, Data.

Invoice: Id, ProductId, ClientId, Data.

This application should provide the following operations:

Sell: Add invoice record.

Account: List names of products sold to some client.

Top10: List currently 10 most sold products

Client and product tables should be pre-populated with `MAX` $= 2^n$ items, for some n. Generate client and product identifiers with `rand.nextInt(MAX)|rand.nextInt(MAX)` when creating new invoices. Use an arbitrary string for Data fields in each table.

Steps

1. Implement the proposed system using an RDBMS, starting with a tool to initialize and populate the database.
2. Design and run queries for each proposed operation.
3. Implement a workload generator, that invokes random operations with a varying number of client threads. Collect response time and throughput statistics.
4. Observe system resource usage with operating system tools while the benchmark is running.
5. Run the benchmark for 1, 2, 4, 8, ... client threads and record throughput and response time results. Plot response time vs. throughput to obtain a scalability curve.
6. Adjust configuration parameters and repeat the benchmark.

Questions

1. After running the benchmark for some time, what are the numerical identifiers of the top 10 products?
2. What is the maximum throughput of your system? What is the expected response time?
3. Can baseline performance be improved?

Learning Outcomes Recall relational database programming with SQL and JDBC. Compute and recognize the significance of key performance statistics. Evaluate and explain performance and scalability using a scalability curve. Use memory configuration parameters to influence system performance. Relate resource usage with performance.

PostgreSQL HowTo

Without Docker (In Linux CentOS/Fedora/Ubuntu/...)

1. Install server binaries using the package manager.
2. Add path to server binaries (Ubuntu only):
   ```
   $ export PATH=/usr/lib/postgresql/11/bin/:$PATH
   ```
3. Initialize data directory:
   ```
   $ initdb -D data
   ```
4. Start server:
   ```
   $ postgres -D data -k.
   ```
5. Create database:
   ```
   $ createdb -h localhost mydb
   ```
6. Command line (optional):
   ```
   $ psql -h localhost mydb
   ```
7. Cleanup after everything stopped:
   ```
   $ rm -rf data
   ```

With Docker

1. Start server:
   ```
   $ docker run --name postgres -p 5432:5432 postgres:11.5
   ```
2. Create user:
   ```
   $ docker exec postgres createuser -U postgres -d $USER
   ```
3. Create database:
   ```
   $ docker exec postgres createdb -U $USER mydb
   ```
4. Command line (optional):
   ```
   $ docker exec -it postgres psql -U $USER mydb
   ```
5. Cleanup after everything stopped:
   ```
   $ docker rm postgres
   ```

From Java

1. Connect to:
   ```
   "jdbc:postgresql://localhost:5432/mydb"
   ```