Midterm Artifact: TAOBench

This document is a midterm artifact for our group, **"Distributed System."** As previously presented, we reviewed the paper "TAO" by Facebook and analyzed two benchmarks: "DCPerf/taobench" and "TAOBench." This artifact shows how we used "TAOBench" to obtain results and provides an overview of the TAOBench structure.

Quick Review

- **Code Repository**: TAOBench on GitHub (Yonghao Lin's GitHub)
- **TAOBench Overview**: OVERVIEW.md provicded by the original author
- **Cloudlab Profile**: Cloudlab TAO Profile
- **Cloudlab Disk Image**: `urn:publicid:IDN+utah.cloudlab.us+image+eecse6894-PG0:TAO`

1. Basic Information

1.1 TAO: Facebook's Distributed Data Store for the Social Graph

- **Paper**: TAO Paper
- **Meta Blog**: Meta TAO Blog

1.2 TAOBench

- **Code Repository**: TAOBench GitHub (Yonghao Lin's GitHub)
- **Paper**: TAOBench Paper
- **Official Website**: TAOBench.org
- **Meta Blog**: TAOBench Blog

1.3 DCPerf/taobench

- **Code Repository**: DCPerf GitHub (Xiaojie Wu's GitHub)
- **Meta Blog**: DCPerf Meta Blog

1.4 Cloudlab Information

- **Profile**: Cloudlab TAO Profile
- **Image Disk**: `urn:publicid:IDN+utah.cloudlab.us+image+eecse6894-PG0:TAO`

---

2. How to Use TAOBench

2.1 Cloudlab Implementation

To implement this artifact on Cloudlab, please follow these steps:

1. Use the "TAO" profile and select the OS image: `urn:publicid:IDN+utah.cloudlab.us+image+eecse6894-PG0:TAO`.

2. Set the hardware to `cloudlab-utah -> c6525-100g`.
3. Follow the Cloudlab instructions to initiate the node.
4. SSH into the node. The artifact files are located in `/local/taobench/`.

2.2 File Structure

To use this artifact, navigate to `/local/taobench/`. Key files include:

- `./src`: Contains all source code for TAOBench.
  - `benchmark.cc`: Main function file for TAOBench:
    - **Command-Line Parsing**: Configures benchmark properties like thread count, database name, and experiment settings.
    - **Operations Tracking**: Tracks completed, failed, and overtime operations.
    - **Workload Functions**:
      - `RunTransactions`: Executes transactions based on experimental settings.
      - `RunBatchInsert`: Performs batch data insertions with multiple threads.
      - `RunTestWorkload`: Runs a simple workload for testing.
    - `./mysqldb`: Contains MySQL database configuration files.
  - `workload_o.json`: Represents a read-heavy workload configuration file.
- `./experiments.txt`: Allows parameter setup for experiments, including `num_threads`, `warmup_len`, and `exp_len`.

2.3 Software Implementation

This image is fully implemented, so the environment is pre-configured. Enter `/local/taobench/` and use the `make` command to ensure any source code modifications are correctly compiled. The MySQL database, named `benchmark`, can be accessed using `sudo mysql`.

Load Data Phase:
To load data, use the following command:

```
sudo ./taobench -load-threads 48 -db mysql -p mysqldb/mysql_db.properties
-c src/workload_o.json -load -n 1500000
```

This command will load data into the benchmark MySQL database. After data loading, verify the row counts with:

SELECT COUNT(*) FROM edges; (Expected: 1,500,000 rows in the edges table) SELECT COUNT(*) FROM objects; (Expected: 3,000,000 rows in the objects table)

Run Experiments Phase:
To run the experiment, use the following command:

```
sudo ./taobench -load-threads 48 -db mysql -p mysqldb/mysql_db.properties
-c src/workload_o.json -run -e experiments.txt
```

This command loads the parameters from experiments.txt, which includes:

```
48: Number of threads
10: Warmup length in seconds
150: Experiment length in seconds
```

After running the experiment, results will be displayed in the terminal and recorded in /local/taobench/results.txt.

3. Result Information

When you check the /local/taobench/results.txt, you can refer to final part of the result which contains like

```
Experiment description: 48 threads, 10 seconds (warmup), 150 seconds
(experiment)
Total runtime (sec): 150.011
Runtime excluding warmup (sec): 140.01
Total completed operations excluding warmup: 22703079
Throughput excluding warmup: 162153
Number of overtime operations: 24298608
Number of failed operations: 0
22703079 operations; [INSERT: Count=17991 Max=12323.52 Min=88.58
Avg=558.00] [READ: Count=22014325 Max=14338.29 Min=25.16 Avg=211.03]
[UPDATE: Count=32115 Max=11956.92 Min=110.62 Avg=660.94] [READTRANSACTION:
Count=635997 Max=169687.88 Min=94.18 Avg=2839.90] [WRITETRANSACTION:
Count=2651 Max=15413.49 Min=255.60 Avg=1597.97] [WRITE: Count=50106
Max=12323.52 Min=88.58 Avg=623.98]
```

After done this execution, you can find slight difference in mysql database compared to original state. Beacause the real workload of TAO is read-heavy, but there is still a few of writes.

```
mysql> SELECT COUNT(*) FROM objects;
+----------+
| COUNT(*) |
+----------+
|  3024351 |
+----------+

mysql> SELECT COUNT(*) FROM edges;
+----------+
| COUNT(*) |
+----------+
|  1500115 |
+----------+
```

> A few clarifications:

> - For throughput, each read/write/read transaction/write transaction counts as a single completed operation.
> - The last line describes operation latencies. The "Count" is the number of completed operations. The "Max", "Min", and "Avg" are latencies in microseconds. The WRITE operation category is an aggregate of inserts/updates/deletes.
> - `Overtime operations` aren't particularly meaningful anymore.