Inventory Management Use Case for rENIAC Row Cache

Contact

rENIAC Support - for technical help or discounts on pricing
support@reniac.com

Thomas Jorgensen, VP of Customer Success
thomas@reniac.com

rENIAC

Table of Contents

rENIAC

1. Overview

rENIAC Row Cache is an FPGA-based Cassandra row cache accelerator. It sits behind a Cassandra database instance, using flash storage that is directly accessible by the FPGA, to store the row cache data. By leveraging extremely fast flash storage, rENIAC Row Cache is able to cache large volumes of data and serve it with predictably low latency.

This document describes the Inventory Management use case for testing the performance benefits provided by rENIAC Row Cache.

2. Use Case Description

This use case is intended to simulate the workload of an inventory management database. The schema contains the following tables:

- Product category
- Product data
- Orders
- Suppliers
- Customers
- Reviews

This is a read-heavy workload, with reads accounting for about 96% of the transactions and writes accounting for 4%. This use case is intended to be tested on AWS, with a Cassandra database cluster. Each database instance is backed by an f1.2xl instance running the rENIAC Row Cache product.

This test creates about 90GB of data on a database node. When queries are accelerated by rENIAC Row Cache, the throughput goes up to almost 7x the throughput vs baseline (i.e. when rENIAC Row Cache is not enabled). The latency also goes down significantly - by about 7x for mean to about 15x for 95th percentile.

Note: These results were obtained in AWS with the following instance types:

- 2 x m5.2xl clients
- 3 x m5.2xl database nodes
- 3 x f1.2xl for running rENIAC Row Cache

3. Running the Test

This section provides details on running the test for this use case. It is assumed that you have a cluster of Apache C* or DSE database and Row Cache nodes already running. For details on setting up rENIAC Row Cache, refer to the user guide.

The tests are executed using the NoSQLBench tool. The `nb` executable can be found in the folder created in the next step.

3.1 Setup

Download the setup files required (it is a tar file) from - https://woir.in/inv_mgmt.tar
To download from unix terminal use following command -

```
wget https://woir.in/inv_mgmt.tar
```

Untar the `inv_mgmt.tar` file to a folder on the client machine(s). Edit the `setup_config` file to specify the following information:

- Absolute path of the `inv_mgmt` folder
- IP address of the client machines (and usernames) from which the test will be executed
- IP address of one the database nodes

3.2 Loading the Data

Start by loading the data in the database cluster using this command, replacing the `<<DB_IP>>` placeholder with the IP address of the database host. This will drop the `inventory_management` keyspace if it exists, create the keyspace and tables, and insert about 90GB of data in each database node.

```
./nb inventory_mgmt.yaml benchmark_setup host=<<DB IP>> port=9042
--progress console:1s
```

3.3 Warming up rENIAC Row Cache

Bring up the f1 nodes and start rENIAC Row Cache on the nodes. See the user guide for details on bringing up the Row Cache nodes. Warm up the row cache using this command:

```
./nb inventory_mgmt.yaml warmup host=<<DB IP>> port=9042 --progress
console:1s
```

At the end of this step, all the data will be loaded into the rENIAC Row Cache instances.

Note: The warmup step is not required in a production environment. Row Cache automatically caches the data for all the tables on which the row cache has been enabled.

3.4 Running the Benchmark Test

Test the performance of rENIAC Row Cache by executing this command on the client(s) simultaneously. This will run the mixed workload test.

```
sh ./mixed_inventory_run.sh
```

At the end of the test, you can view the results by running this command on the client(s):

```
sh ./print_results.sh
```

This will print the results of the test to the console.

Next, turn off the row cache (see the user guide for instructions). Rerun the above two commands to see the performance of the database cluster without the benefit of rENIAC Row Cache. Compare the two sets of results.

4. Conclusion

The above test clearly shows the benefits offered by rENIAC Row Cache - increased throughput and predictably low latencies. rRC acting as a custom implementation of the native Cassandra row cache provides users with capabilities for increased throughput on demand, reduced latency during peak traffic and increased data capacity vs an in-memory cache.

For support, pricing, or any other inquiries please contact support@reniac.com or visit reniac.com for more information.