Data Caching

CloudSuite 2.0 Benchmark Suite

This benchmark uses the Memcached data caching server, simulating the behavior of a Twitter caching server using the twitter dataset. The metric of interest is throughput expressed as the number of requests served per second. The workload assumes strict quality of service guaranties.

Download the data caching benchmark.

Prerequisite Software Packages

1. Memcachd v1.4.15
2. Libevent v2.0.21 library required to run Memcached

Building Memcached

1. Untar the libevent package and configure.

   tar zxvf libevent-2.0.21-stable.tar.gz
   cd libevent-2.0.21-stable
   ./configure make && make install

   This will install the library at the default location (/usr/lib/). To change the destination folder, you should configure the package with a different command:
   ./configure --prefix=/path/to/folder/

2. Untar the Memcachd package and configure.

   tar zxvf memcached-1.4.15.tar.gz
   cd memcached-1.4.15
   ./configure

   In case you did not install libevent at the default location, you should add the following parameter:
   --with-libevent=/path/to/libevent/

   In case you want to install memcached at a specific location, you should add:
   --prefix=/path/to/memcached/
   For example:
   ./configure --prefix=/path/to/memcached/ --with-libevent=/path/to/libevent/

3. Build and install memcached:
   make && make install

Starting the server

The following command will start the server with four threads and 4096MB of dedicated memory, with the minimal object size of 550 bytes:
memcached -t 4 -m 4096 -n 550
The default port the server will be started on is 11211. You can change it by adding the port parameter (e.g., -p 11212).

Preparing the client

Go to memcached/memcached_client/

In case you did not install libevent at the default location, you should update Makefile:
-L path/to/libevent/lib -I path/to/libevent/include

Compile the client code by typing: make

Then prepare the server configuration file, which includes the server address and the port number to connect to in the following format:
server address, port

The client can simultaneously use multiple servers, or one server with several ip addresses (in case the server machine has multiple network cards active), or one server through multiple ports, measuring the aggregate throughput and quality of service. In that case, each line in the configuration file should contain the server address and the port number. For example:
localhost, 11211
127.0.0.1, 11212
n116.epfl.ch, 11213
192.168.10.116, 11213

Scaling the dataset and warming up the server

The following command will create the dataset by scaling up the Twitter dataset, while preserving both the popularity and object size distributions. The original dataset consumes 300MB of server memory, while the recommended scaled dataset requires around 10GB of main memory dedicated to the memcached server (scaling factor of 30).
./loader -a ../twitter_dataset/twitter_dataset_unscaled -o ../twitter_dataset/twitter_dataset_30x -s servers.txt -w 1 -S 30 -D 4096 -j -T 1

(w - number of client threads, S - scaling factor, D - target server memory, T - statistics interval, s - server configuration file, j - an indicator that the server should be warmed up).

If the scaled file is already created, but the server is not warmed up, use the following command to warm up the server:
./loader -a ../twitter_dataset/twitter_dataset_30x -s servers.txt -w 1 -S 1 -D 4096 -j -T 1

Running the benchmark

To determine the maximum throughput while running the workload with eight client threads, 200 TPC/IP connections and a get/set ration of 0.8, please type:
./loader -a ../twitter_dataset/twitter_dataset_30x -s servers.txt -g 0.8 -T 1 -c 200 -w 8

This command will run the benchmark with the maximum throughput, however, the QoS requirements will highly likely be violated. Once the maximum throughput is determined, you should run the benchmark using the following command:

./loader -a ../twitter_dataset/twitter_dataset_30x -s servers.txt -g 0.8 -T 1 -c 200 -w 8 -e -r *rps*

where *rps* is 90% of the maximum number of requests per second achieved using the previous command. You should experiment with different values for *rps* to achieve the maximum throughput without violating the target QoS requirements.

Important remarks

1. It takes several minutes for the server to reach a stable state.
2. The target QoS requires that 95% of the requests are executed within 10ms
3. Memcached has known scalability problems, scaling very poorly beyond four threads. To utilize a machine with more than four cores, you should start several server processes and add the corresponding parameters into the client configuration file.
4. The benchmark is network-intensive and requires a 10Gbit Ethernet card not to be network-bound. Multiple ethernet cards could be used as well, each with a different IP address (two servers in the client configuration file with the same socket, but different IP address). Multisocket machines could also avoid the network problem by running the server and the client on different sockets of the same machine (e.g., pinned using taskset), communicating via localhost.