Distributed Cache Service

Performance Whitepaper

Issue 01

Date 2020-11-30

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

Contents

1 Introduction

This document describes how to test the performance of DCS Redis instances. For example, you can test how fast a specific instance responds to high-concurrency **SET** or **GET** operations.

Test Tools

- redis-benchmark

  The Redis client includes redis-benchmark, a performance testing utility that simulates N clients concurrently sending M query requests.

  Usage of redis-benchmark:

  ```
  redis-benchmark [-h {host}] [-p {port}] [-c {clients}] [-n {requests}> [-k {boolean}]
  ```

- memtier_benchmark

  memtier_benchmark is a command-line tool developed by Redis Labs. It can generate traffic in various modes and supports both Redis and Memcached.

  This tool provides multiple options and reporting features that can be easily used through the CLI.

  For details, visit **https://github.com/RedisLabs/memtier_benchmark**.

Test Procedure

Step 1 Create a DCS Redis instance.

Step 2 Create three ECSs and configure the same AZ, VPC, subnet, and security group for the ECSs and the instance.

Only one ECS is required for testing on a single-node or master/standby instance.

Step 3 Install **redis-benchmark** and **memtier_benchmark** on each ECS.

- Installing redis-benchmark

  ```
  $ wget http://download.redis.io/releases/redis-5.0.8.tar.gz
  $ tar xzf redis-5.0.8.tar.gz
  $ cd redis-5.0.8/src
  $ make
  ```

- Installing memtier_benchmark

  ```
  cd $HOME && \
  apt-get update && \
  apt-get install -y build-essential autoconf automake libpcre3-dev libevent-dev pkg-config zlib1g-dev
  redis-tools && \
  ```



```
wget https://github.com/RedisLabs/memtier_benchmark/archive/1.2.17.zip && \
unzip 1.2.17.zip && \
cd memtier_benchmark-1.2.17 && \
autoreconf -ivf && \
./configure && \
make && \
make install && \
cd $HOME && \
rm -rf 1.2.17.zip memtier_benchmark-1.2.17
```

Step 4 Run the test commands simultaneously on all ECSs.

```
./redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connect_number} -d {datasize} -t {command}
./memtier_benchmark --cluster-mode --ratio=(1:0 and 0:1)-s {IP} -n {nreqs}  -c {connect_number}  -t 4 -d {datasize}
```

Reference values: **-c {connect_number}**: **200**; **-n {nreqs}**: **10,000,000**; **-r {randomkeys}**: **1,000,000**; **-d {datasize}**: **32**

- **-t** indicates the command to run, such as **SET** or **GET**.
- **-c** indicates the number of client connections.
- **-d** indicates the size of a single data record in bytes.
- **-n** indicates the number of test packets.
- **-r** indicates the number of random keys.

Step 5 Repeat **Step 4** with different client connections to obtain the maximum number of operations per second.

Step 6 The sum of operations per second of all the three ECSs indicates the performance of the instance specification.

To test on a Redis Cluster instance, launch two benchmark tools on each ECS.

----End

Test Metric

Queries per second (QPS), which is the number of commands processed per second.

2 Test Data of Master/Standby DCS Redis 3.0 Instances

Test Environment

- Redis instance specifications
  Redis 3.0 | 8 GB | master/standby
  Redis 3.0 | 32 GB | master/standby
- ECS flavors
  General computing-enhanced | c6.xlarge.2 | 4 vCPUs | 8 GB

Test Command

```
./redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connection} -d {datasize} -t {command}
```

Reference values: **-c {connect_number}**: **1000**; **-n {nreqs}**: **10,000,000**; **-r {randomkeys}**: **1,000,000**; **-d {datasize}**: **32**

Test Result

Table 2-1 Test result of running the SET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)
8 GB	x86	1000	107,657.69	20	23	27
		10,000	72,750.55	362	366	371
32 GB	x86	1000	121,088.83	9	12	12
		10,000	79,235.53	203	204	267

Table 2-2 Test result of running the GET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)
8 GB	x86	1000	119,350.25	6	24	27
		10,000	77,574.7	152	358	365
32 GB	x86	1000	124,650.98	16	17	17
		10,000	81,991.41	195	196	199

NOTE

DCS for Redis 3.0 does not support the Arm CPU architecture, so only x86-based instance test results are provided.

3 Test Data of Proxy Cluster DCS Redis 3.0 Instances

Test Environment

- Redis instance specifications
  Redis 3.0 | 64 GB | Proxy Cluster
- ECS flavors
  General computing-enhanced | c6.xlarge.2 | 4 vCPUs | 8 GB

Test Command

```
./redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connection} -d {datasize} -t {command}
```

Reference values: **-c {connect_number}**: **1000**; **-n {nreqs}**: **10,000,000**; **-r {randomkeys}**: **1,000,000**; **-d {datasize}**: **32**

Test Result

Table 3-1 Test result of running the SET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)
64 GB	x86	1000	534,960.92	24	34	209
		10,000	511,362.67	108	171	315

Table 3-2 Test result of running the GET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)
64 GB	x86	1000	584,669.15	23	31	82
		10,000	533,178.04	144	184	370

NOTE

DCS for Redis 3.0 does not support the Arm CPU architecture, so only x86-based instance test results are provided.

4 Test Data of Master/Standby DCS Redis 4.0 or 5.0 Instances

Test Environment

- Redis instance specifications
  Redis 4.0 or 5.0 | 8 GB | master/standby
  Redis 4.0 or 5.0 | 32 GB | master/standby
- ECS flavors
  General computing-enhanced | c6.2xlarge.2 | 8 vCPUs | 16 GB
- ECS image
  Ubuntu 18.04 server 64bit

Test Command

```
./redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connection} -d {datasize} -t {command}
```

Reference values: **-c {connect_number}**: **1000**; **-n {nreqs}**: **10,000,000**; **-r {randomkeys}**: **1,000,000**; **-d {datasize}**: **32**

Test Result

Table 4-1 Test result of running the SET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)	Average Latency (ms)
8 GB	x86	500	16318 8.20	6.4	8.8	210	3.037
		10,000	10223 8.01	180	210	400	98.577

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)	Average Latency (ms)
	Arm	500	106372.52	8.2	8.6	17	4.669
		10,000	75182.96	300	360	470	133.235
32 GB	x86	500	164256.78	5.8	6.4	210	3.033
		10,000	103356.76	160	170	390	96.991
	Arm	500	110064.45	8.4	10	19	4.543
		10,000	84791.04	270	370	480	118.359

Table 4-2 Test result of running the GET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)	Average Latency (ms)
8 GB	x86	500	202,929.67	4.2	4.5	210	2.47
		10,000	122,407.16	300	330	580	81.988
	Arm	500	135,342.56	6.6	7.3	15	3.679
		10,000	92,454.73	340	390	700	108.172
32 GB	x86	500	202,252.89	4.1	4.6	210	2.47
		10,000	123,700.68	140	160	360	81.092
	Arm	500	138,582.99	6.6	7.2	22	3.599

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)	Average Latency (ms)
		10,000	97,174.25	220	350	470	103.06

5 Test Data of Redis Cluster DCS Redis 4.0 or 5.0 Instances

Test Environment

- Redis instance specifications
  Redis 4.0 or 5.0 | 32 GB | Redis Cluster
- ECS flavors
  General computing-enhanced | c6.xlarge.2 | 4 vCPUs | 8 GB

Test Command

```
./memtier_benchmark --cluster-mode --ratio=(1:0 and 0:1)-s {IP} -n {nreqs}  -c {connect_number}  -t 4 -d {datasize}
```

Reference values: **-c {connect_number}**: **1000**; **-n {nreqs}**: **10,000,000**; **-r {randomkeys}**: **1,000,000**; **-d {datasize}**: **32**

Test Result

Table 5-1 Test result of running the SET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)
32 GB	x86	1000	371,780.2	5.6	6.3	44
		10,000	256,073.11	90	220	460
	Arm	1000	317,053.78	17	34	230
		10,000	248,832.33	410	490	750

Table 5-2 Test result of running the GET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)
32 GB	x86	1000	427,000.04	5.0	5.3	78
		10,000	302,159.03	63	220	460
	Arm	1000	421,402.06	13	14	65
		10,000	309,359.18	180	260	500