ApsaraDB for Memcache

Quick Start

Quick Start

Alibaba Cloud ApsaraDB for Memcache only supports the key-value format data and does not support complex data types such as array, map, and list. Therefore, ApsaraDB for Memcache is not suitable for storing complex data types.

ApsaraDB for Memcache supports a maximum of 1 KB and 1 MB respectively for the key size and value size of a single piece of cached data. ApsaraDB for Memcache is not suitable for storing sizable data.

ApsaraDB for Memcache does not support transactions. Therefore, ApsaraDB for Memcache is not suitable for storing data with transaction requirements. Data with transaction requirements should be written into the database directly.

When data accesses are evenly distributed, and there is no obvious hotspot or less popular data, a large number of access requests cannot hit the cache data in ApsaraDB for Memcache. Therefore, ApsaraDB for Memcache is not effective to function as the database cache. You should give full consideration to the data access requirements of the business model when selecting the database cache.

Any client that is compatible with the Memcached protocol can access ApsaraDB for Memcache services. You can choose to use any Memcached client supporting SASL and Memcached Binary Protocol based on your application features.

Protocols

- Memcached Binary Protocol (binary)
- SASL authentication protocol

Commands

ApsaraDB for Memcache supports the following commands for data operations.

Operation code	Operation command	Remarks
0x00	Get	

0x01	Set	
0x02	Add	
0x03	Replace	
0x04	Delete	
0x05	Increment	
0x06	Decrement	
0x07	Quit	
0x08	Flush	Memcache time is accurate to the second
0x09	GetQ	
0x0a	No-op	
0x0b	Version	
0x0c	GetK	
0x0d	GetKQ	
0x0e	Append	
0x0f	Prepend	
0x10	Stat	Not supported
0x11	SetQ	
0x12	AddQ	
0x13	ReplaceQ	
0x14	DeleteQ	
0x15	IncrementQ	
0x16	DecrementQ	
0x17	QuitQ	
0x18	FlushQ	
0x19	AppendQ	
0x1a	PrependQ	
0x1b	Verbosity	Not supported
0x1c	Touch	
0x1d	GAT	
0x1e	GATQ	
0x20	SASL list mechs	
0x21	SASL Auth	

0x22	SASL Step	

ApsaraDB for Memcache supports subscription and pay-as-you-go billing methods. The latter can be upgraded to the former mode, but not the opposite. You can choose your preferred payment method based on your needs.

Notes

The instance is billed immediately after it is activated, whether used or not. So it is recommended that you apply for it as required to avoid unnecessary expenditure.

At least one ECS is required for activating Alibaba Cloud ApsaraDB for Memcache. See Purcahse ECS.

Procedure

Log on to ApsaraDB for Memcache Console, and click the **New Instance** button at the top right corner.

The Subscription purchase mode will be displayed by default. Select **Region**, **Zone**, **Instance Type**, **Service Period**, and set the instance **Login Password**, **Instance Name** and **Quantity**.

Note:

ApsaraDB for Memcache only allows intranet access. We recommend you set the zone of ApsaraDB for Memcache to be the same zone in the same region as the ECS.

If you want pay-as-you-go instances, select “Pay-As-You-Go”. Other items are basically the same as for subscription instances.

It is recommended that you set a password for cloud instance access when purchasing the instance. If no password is set, you need to reset the password in **Instance List**>**Manage**>**Change Password**.

Click **Purchase Now** after you finalize the selection and enter the **Confirm Order** page. Confirm the order information and click the **Pay** button.

On the payment page, select the payment option and click the **Confirm Payment** button.

A prompt will be displayed after successful payment. Go to the console after one to five minutes and you will see the instance you just purchased.

Connect to database through clients

Any client that is compatible with the Memcached protocol can access ApsaraDB for Memcache services. Each Memcached client has its own characteristics. You can select any Memcached client supporting SASL or Memcached Binary Protocol based on application characteristics.

The following Memcache clients can interact smoothly with ApsaraDB for Memcache, and therefore are recommended for use.

Note:

ApsaraDB for Memcache instances can be accessed through remote logins to ECS. See Connect through Internet.

All the below third-party open-source clients are not provided by the Alibaba Cloud official team and it may contain bugs. Developers should ensure the quality of the client on their own. Alibaba Cloud is not held liable for any direct or indirect faults or losses arising from the client.

Download client

Download client

About client

Client versions

Java sample code

Prepare the Java development environment. Log in to the ECS server, and install the Java Development Kit (JDK) and commonly used integrated development environment (IDE) (such as Eclipse) on the server.

Java SDK Download address

Eclipse Download Address 1, Download address 2

The first example code is as follows. Copy the Java code in it into the Eclipse Project.

Note: At this time, the code compilation will fail. A third-party JAR package download address is required to call the Memcache cache service. With this JAR package added, the code compilation will go through.

OcsSample1.java example code (user name and password are required)

```
 import java.io.IOException;
import java.util.concurrent.ExecutionException;
import net.spy.memcached.AddrUtil;
import net.spy.memcached.ConnectionFactoryBuilder;
import net.spy.memcached.ConnectionFactoryBuilder.Protocol;
import net.spy.memcached.MemcachedClient;
import net.spy.memcached.auth.AuthDescriptor;
import net.spy.memcached.auth.PlainCallbackHandler;
import net.spy.memcached.internal.OperationFuture;

public class OcsSample1 {

public static void main(String[] args) {

final String host = "xxxxxxxx.m.yyyyyyyyyy.ocs.aliyuncs.com";//The "intranet address" in the console
final String port =” 11211” ; //Default port 11211, no need to change
final String username = "xxxxxxxxx";//The "access account" in the console. For new OCS, the username can be empty
final String password = "my_password";//The "password" provided in the e-mail
MemcachedClient cache = null;
try {
AuthDescriptor ad = new AuthDescriptor(new String[]{"PLAIN"}, new PlainCallbackHandler(username, password));

cache = new MemcachedClient(
new ConnectionFactoryBuilder().setProtocol(Protocol.BINARY)
.setAuthDescriptor(ad)
.build(),
AddrUtil.getAddresses(host + ":" + port));

System.out.println("OCS Sample Code");

//Store a piece of data with the “ocs” key into the OCS to facilitate subsequent data verification and reading.
String key = "ocs";
String value = "Open Cache Service, from www.Aliyun.com";
int expireTime = 1000; // Expiration time, unit: seconds. Timing started from the data write. After the expireTime elapses, the data will go expired and won't be read;
OperationFuture<Boolean> future = cache.set(key, expireTime, value);
future.get(); // The spymemcached set() method is asynchronous. The future.get() waits for the cache.set() operation to complete. You can also choose not to wait based on your requirements.
```

```
//Store several pieces of data into the OCS and then you can see the statistics information on the OCS
console.
for(int i=0;i<100;i++){
key="key-"+i;
value="value-"+i;

//Execute the set operation and store data into the cache
expireTime = 1000; // Expiration time, unit: seconds
future = cache.set(key, expireTime, value);
future.get(); // Make sure the previous (cache.set()) operation has been completed
}
System.out.println(" Set operation completed.");
//Execute the get operation and read data from the cache. Read data with the "ocs" key.

System.out.println(" Get operation:"+cache.get(key));

} catch (IOException e) {
e.printStackTrace();
} catch (InterruptedException e) {
e.printStackTrace();
} catch (ExecutionException e) {
e.printStackTrace();
}
if (cache != null) {
cache.shutdown();
}

}//eof
}
```

OcsSample2.java example code (user name and password are not required)

```
 import java.io.IOException;
import java.util.concurrent.ExecutionException;

import net.spy.memcached.AddrUtil;
import net.spy.memcached.BinaryConnectionFactory;
import net.spy.memcached.MemcachedClient;
import net.spy.memcached.internal.OperationFuture;

public class OcsSample2 {

public static void main(String[] args) {

final String host = "xxxxxxxx.m.yyyyyyyyyy.ocs.aliyuncs.com"; //The "intranet address" on the console
final String port =" 11211" ; //Default port 11211, no need to change

MemcachedClient cache = null;
try {

cache = new MemcachedClient(new BinaryConnectionFactory(), AddrUtil.getAddresses(host + ":" + port));

System.out.println("OCS Sample Code");

//Store a piece of data with the "ocs" key into the OCS to facilitate subsequent data verification and
```

```
reading.
String key = "ocs";
String value = "Open Cache Service, from www.Aliyun.com";
int expireTime = 1000; // Expiration time, unit: seconds. Timing started from the data write. After the
expireTime elapses, the data will go expired and won't be read;
OperationFuture<Boolean> future = cache.set(key, expireTime, value);
future.get();
//Store several pieces of data into the OCS and then you can see the statistics information on the OCS
console.
for (int i = 0; i < 100; i++) {
key = "key-" + i;
value = "value-" + i;

//Execute the set operation and store data into the cache
expireTime = 1000; // Expiration time, unit: seconds
future = cache.set(key, expireTime, value);
future.get();
}

System.out.println(" Set operation completed.");

//Execute the get operation and read data from the cache. Read data with the "ocs" key.
System.out.println("Get operation:" + cache.get(key));

} catch (IOException e) {
e.printStackTrace();
} catch (InterruptedException e) {
e.printStackTrace();
} catch (ExecutionException e) {
e.printStackTrace();
}
if (cache != null) {
cache.shutdown();
}

}//eof
}
```

Modify several items in OcsSample1.java in Eclipse according to your own instance information.

The ApsaraDB for Memcache instance ID is unique and the Alibaba Cloud intranet address it corresponds to is also unique. All the information is displayed on the ApsaraDB for Memcache console. When establishing connections with your own ApsaraDB for Memcache instance, you need to modify the corresponding information in OcsSample1.java based on the information.

After the information is modified, you can run your own program. Run the main function, and you will see the following result in the console window within Eclipse (ignore the red INFO debugging information that may appear).

```
OCS Sample Code
```

```
Set operation is completed.
Get operation: Open Cache Service, from www.Aliyun.com
```

Download client

Download client

About client

Client versions

System requirements and environment configuration

Note: If you have a PHP Memcache environment, pay attention to the prompts during the tutorial to avoid overwriting the production environment, which may render the business unavailable. It is recommended that you back up the data before upgrading or compiling the environment.

Windows series

If the environment cannot be established successfully using the standard PHP Memcached extensions, you can consider changing to manual packet splicing to access Alibaba Cloud ApsaraDB for Memcache. For connection methods, you can refer to the following link. The example code is very simple and its difference with the PHP Memcached lies in that it only supports mainstream interfaces and you need to supplement some specific interfaces on your own. For installation and usage methods, see here.

CentOS and Alibaba Cloud Linux 6 series

Note: Memcached 2.2.0 extensions must use Libmemcached 1.0.x libraries, and libraries lower than version 1.0 won’ t be compiled successfully. GCC must be of Version 4.2 or above for compiling Libmemcached.

Check whether the gcc-c++ or other components have been installed (use the gcc –v command to see whether the version is 4.2 or above). If not, execute the command yum install gcc+ gcc-c++.

Run the command rpm –qa | grep php to check whether the PHP environment is ready in the system. If no, run the command yum install php-devel,php-common,php-cli to install PHP with source code compiling.

It is recommended that you use PHP 5.3 or above. Part of PHP 5.2 versions may contain the

zend_parse_parameters_none function in the source code, and the function may have errors. If you need to use the function, refer to the PHP official documents. For source code compilation, follow the official PHP compiling and upgrading methods.

Check whether the SASL-related environmental packets have been installed. If no, run the command yum install cyrus-sasl-plain cyrus-sasl cyrus-sasl-devel cyrus-sasl-lib to install the SASL-related environments.

Check whether the Libmemcached source packages have been installed. If no, run the command below to install Libmemcached source package (Recommended version: Libmemcached-1.0.18).

```
 wget https://launchpad.net/libmemcached/1.0/1.0.18/+download/libmemcached-1.0.18.tar.gz
tar zxvf libmemcached-1.0.18.tar.gz
cd libmemcached-1.0.18
./configure --prefix=/usr/local/libmemcached --enable-sasl
make
make install
cd ..
```

Run the command yum install zlib-devel to install the Memcached source packages (Recommended version: Memcached-2.2.0).

Note:

Check whether there is any zlib-devel package to be executed first before installing the Memcached.

Check whether the Memcached client package has been installed first (including the source package). If yes, you need to re-compile it to add the -enable-memcached-sasl extension.

```
wget http://pecl.php.net/get/memcached-2.2.0.tgz
tar zxvf memcached-2.2.0.tgz
cd memcached-2.2.0
Phpize (If there are two sets of PHP environments in the system, you need to call the command on the absolute path /usr/bin/phpize. The path is the PHP environment path for using Alibaba Cloud ApsaraDB for Memcache).
./configure --with-libmemcached-dir=/usr/local/libmemcached --enable-memcached-sasl (Pay attention to this parameter)
make
make install
```

Modify the php.ini file (Locate this file. If there are two sets of PHP environments in the

system, you need to find the PHP environment path for using ApsaraDB for Memcache and modify it accordingly) and add extension=memcached.so memcached.use_sasl = 1.

Use the testing code at the bottom of this page to test whether the environment has been deployed successfully. Modify the corresponding address, port, user name and password in the code.

Centos and Alibaba Cloud Linux 5 series [64-bit]

Check whether the gcc-c++ or other components have been installed. If not, run the command yum install gcc+ gcc-c++.

Run the command rpm –qa | grep php to check whether the PHP environment is ready in the system. If no, run the command yum install php53 php53-devel to install the PHP with source code compiling. If the PHP environment is already prepared, no installation is required. It is recommended that you use PHP 5.3 or later.

Part of PHP 5.2 versions may contain the zend_parse_parameters_none function in the source code, and the function may have errors. If you need to use the function, refer to the PHP official documents.

Run the command yum install cyrus-sasl-plain cyrus-sasl cyrus-sasl-devel cyrus-sasl-lib to install the SASL-related environments.

Check whether the Libmemcached (including source packages) has been installed. If no, run the command below to install Libmemcached (Recommended version: Libmemcached 1.0.2).

```
 wget http://launchpad.net/libmemcached/1.0/1.0.2/+download/libmemcached-1.0.2.tar.gz
tar -zxvf libmemcached-1.0.2.tar.gz
cd libmemcached-1.0.2
./configure --prefix=/usr/local/libmemcached --enable-sasl
make
make install
cd ..
```

Run the command yum install zlib-devel to install the Memcached source package (Recommended version: Memcached 2.0).

Note:

Check whether there is any zlib-devel package to be executed first before installing

the Memcached.

Check whether the Memcached client package has been installed first (including the source package). If yes, you need to re-compile it to add the -enable-memcached-sasl extension.

```
wget http://pecl.php.net/get/memcached-2.0.0.tgz tar -zxvf memcached-2.0.0.tgz
cd memcached-2.0.0 phpize (If there are two sets of PHP environments in the system, you need to call the command on the absolute path /usr/bin/phpize. The path is the PHP environment path for using Alibaba Cloud ApsaraDB for Memcache. Execute phpize in the memcached source code directory).
./configure --with-libmemcached-dir=/usr/local/libmemcached --enable-memcached-sasl (Pay attention to this parameter)
make
make install
```

Modify the php.ini file (Locate the file. The yum is usually installed in /etc/php.ini. If there are two sets of PHP environments in the system, you need to find the PHP environment path for using ApsaraDB for Memcache and modify it accordingly) and add extension=memcached.so memcached.use_sasl = 1.

Run the command php –m |grep ，memcached. If the displayed result includes memcache, it indicates the memcache has been supported in the environment.

Use the testing code at the bottom of this page to test whether the environment has been deployed successfully. Modify the corresponding address, port, user name and password in the code.

Ubuntu Debian series

Change the Ubuntu source.

Solution 1: Run the command vim /etc/apt/source.list and add the following content at the beginning.

```
 deb http://mirrors.aliyun.com/ubuntu/ precise main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ precise-security main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ precise-updates main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ precise-proposed main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ precise-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ precise main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ precise-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ precise-updates main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ precise-proposed main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ precise-backports main restricted universe multiverse
```

```
apt-get update //Update the list
```

Solution 2: Download the update_source package at wget http://oss.aliyuncs.com/aliyunecs/update_source.zip, extract the package and grant it the execution permission chmod 777 file name, and execute the script to perform the automatic source change operation.

Configure GCC, G++ with ape-get.

First, you need to run the command dpkg –s installation package name, such as dpkg –s gcc to check whether gcc-c++ or other components have been installed. If no, run the command apt-get build-dep gcc apt-get install build-essential.

Install PHP 5, PHP5-dev.

First, you need to run the command dpkg –s installation package name, such as dpkg –s php to check whether php or other components have been installed. If no, run the command apt-get install php5 php5-dev (and php5-cli and php5-common will be installed at the same time).

Install and configure SASL support.

First, you need to run the command dpkg –s installation package name, such as dpkg –s libsasl2 to check whether libsasl2 cloog-ppl or other components have been installed. If no, run the following command.

```
apt-get install libsasl2-dev cloog-ppl
cd /usr/local/src
```

Run the following command to install Libmemcache of a specific version.

Note: Check whether these packages have been installed first (including the source package). If yes, no installation is required.

```
wget https://launchpad.net/libmemcached/1.0/1.0.18/+download/libmemcached-1.0.18.tar.gz
tar -zxvf libmemcached-1.0.18.tar.gz
cd libmemcached-1.0.18
./configure --prefix=/usr/local/libmemcached
make
make install
cd ..
```

Run the following command to install Memcached of a specific version.

Note: Check whether the Memcached client package has been installed first (including the source package). If yes, no installation is required, but you need to recompile it to add the --enable-memcached-sasl extension.

```
wget
http://pecl.php.net/get/memcached-2.2.0.tgz tar zxvf memcached-2.2.0.tgz
cd memcached-2.2.0 phpize5
./configure --with-libmemcached-dir=/usr/local/libmemcached --enable-memcached-sasl
make
make install
```

Configure PHP to support Memcached and then run a test.

```
echo "extension=memcached.so" >>/etc/php5/conf.d/pdo.ini echo "memcached.use_sasl = 1"
>>/etc/php5/conf.d/pdo.ini
php -m |grep mem memcached
```

If this component is displayed, it indicates the component has been installed and configuration is completed.

PHP code example

Example 1: Basic connections to ApsaraDB for Memcache and set/get operations

```
<?php
$connect = new Memcached; //Declare a new Memcached connection
$connect->setOption(Memcached::OPT_COMPRESSION, false); //Close the compression function
$connect->setOption(Memcached::OPT_BINARY_PROTOCOL, true); //Use the binary protocol
$connect->setOption(Memcached::OPT_TCP_NODELAY, true); //Important: The php memcached has a bug that when the get value does not exist, there will be a fixed 40 ms of latency. Enabling this parameter can avoid this bug.
$connect->addServer('aaaaaaaaaa.m.yyyyyyyyyy.ocs.aliyuncs.com', 11211); //Add the OCS instance address and port number
$connect->setSaslAuthData('aaaaaaaaaa', 'password'); //Set the OCS account and password for authentication. If the password-free function has been enabled, this step can be skipped. For new OCS, the username can be empty.
$connect->set("hello", "world");
echo 'hello: ',$connect->get("hello");
$connect->quit();
?>
```

Example 2: Cache an array in ApsaraDB for Memcache

```
<?php
$connect= new Memcached; //Declare a new Memcached connection
```

```
$connect->setOption(Memcached::OPT_COMPRESSION, false); //Close the compression function
$connect->setOption(Memcached::OPT_BINARY_PROTOCOL, true);//Use the binary protocol
$connect->setOption(Memcached::OPT_TCP_NODELAY, true); //Important: The php memcached has a bug that when the get value does not exist, there will be a fixed 40 ms of latency. Enabling this parameter can avoid this bug.
$connect->addServer('xxxxxxxx.m.yyyyyyyy.ocs.aliyuncs.com', 11211);//Add the OCS instance address and port number
$connect->setSaslAuthData('xxxxxxxx', 'bbbbbbbb');//Set an OCS account password for authentication. If no password is required, skip this step.
$user = array(
"name" => "ocs",
"age" => 1,
"sex" => "male"
); //Declare an array.
$expire = 60; //Set an expiration time.
test($connect->set('your_name',$user,$expire), true, 'Set cache failed');
if($connect->get('your_name')){
$result =$connect->get('your_name');
}else{
echo "Return code:", $connect->getResultCode();
echo "Retucn Message:", $connect->getResultMessage (); //If an error is prompted, parse the return code
$result=" ";
}
print_r($result);
$connect->quit();

function test($val, $expect, $msg)
{
if($val!= $expect) throw new Exception($msg);
}
?>
```

Example 3: Combined use of ApsaraDB for Memcache and MySQL database

```
<?php
$connect = new Memcached; //Declare a new Memcached connection
$connect->setOption(Memcached::OPT_COMPRESSION, false);//Close the compression feature
$connect->setOption(Memcached::OPT_BINARY_PROTOCOL, true);//Use the binary protocol
$connect->setOption(Memcached::OPT_TCP_NODELAY, true); //Important: The php memcached has a bug that when the get value does not exist, there will be a fixed 40 ms of latency. Enabling this parameter can avoid this bug.
$connect->addServer('xxxxxx.m.yyyyyyyy.ocs.aliyuncs.com', 11211);//Add the instance address and port number
$connect->setSaslAuthData('xxxxxx', 'my_passwd'); //Set an OCS account password for authentication. If no password is required, skip this step.
$user = array(
"name" => "ocs",
"age" => 1,
"sex" => "male"
); //Define an array.

if($connect->get('your_name'))
{
$result =$connect->get('your_name');
print_r($result);
```

```
echo "Found in OCS, get data from OCS"; //If data is obtained, print the data to be fromOCS
exit;
}
else
{
echo "Return code:", $connect->getResultCode();
echo "Retucn Message:", $connect->getResultMessage ();//Throw the return code
$db_host='zzzzzz.mysql.rds.aliyuncs.com'; //Database address
$db_name='my_db'; //database name
$db_username='db_user'; //User name of the database
$db_password='db_passwd'; //Specify a database user password.
$connection=mysql_connect($db_host,$db_username,$db_password);
if (!mysql_select_db($db_name, $connection))
{
echo 'Could not select database'; //If the database connection fails, an error will be thrown out
exit;
}
$sql = "SELECT name,age,sex FROM test1 WHERE name = 'ocs'";
$result = mysql_query($sql, $connection);
while ($row = mysql_fetch_assoc($result))
{
$user = array(
"name" => $row["name"],
"age" => $row["age"],
"sex" => $row["sex"],
);
$expire = 5; //Set a data expiration time in the cache.
test($connect->set('your_name',$user,$expire), true, 'Set cache failed'); //Write to the OCS cache.
}
mysql_free_result($result);
mysql_close($connection);
}
print_r($connect->get('your_name')); //Print the data obtained
echo "Not Found in OCS,get data from MySQL"; //Confirm the data obtained from the database
$connect->quit();

function test($val, $expect, $msg)
{
if($val!= $expect) throw new Exception($msg);
}
?>
```

Download client

Download client

About client

Client versions

Environment configuration

Dependent on Bmemcached (SASL extensions supported). For download link, see here.

Python sample code

```
#!/usr/bin/env python
import bmemcached
client = bmemcached.Client(('ip:port'), 'user', 'passwd')
print client.set('key', 'value11111111111')
print client.get('key')
```

Download client

Download client

About client

Client versions

C#/.NET example code

```
using System.Net;
using Enyim.Caching;
using Enyim.Caching.Configuration;
using Enyim.Caching.Memcached;
namespace OCS.Memcached
{
public sealed class MemCached
{
private static MemcachedClient MemClient;
static readonly object padlock = new object();
//Thread-safe single instance mode
public static MemcachedClient getInstance()
{
if (MemClient == null)
{
lock (padlock)
{
if (MemClient == null)
{
MemClientInit();
}
}
}
return MemClient;
}

static void MemClientInit()
```

```
{
//Initialize cache
MemcachedClientConfiguration memConfig = new MemcachedClientConfiguration();
IPAddress newaddress =
IPAddress.Parse(Dns.GetHostEntry
("your_ocs_host").AddressList[0].ToString()) ; //Replace 'your_ocs_host' with the OCS intranet address
IPEndPoint ipEndPoint = new IPEndPoint(newaddress, 11211);

// Configuration file - IP
memConfig.Servers.Add(ipEndPoint);
// Configuration file - protocol
memConfig.Protocol = MemcachedProtocol.Binary;
// Configuration file - permission
memConfig.Authentication.Type = typeof(PlainTextAuthenticator);
memConfig.Authentication.Parameters["zone"] = "";
memConfig.Authentication.Parameters["userName"] = "username";
memConfig.Authentication.Parameters["password"] = "password";
//Follow the maximum connections of the instance for the following settings
memConfig.SocketPool.MinPoolSize = 5;
memConfig.SocketPool.MaxPoolSize = 200;
MemClient=new MemcachedClient(memConfig);
}
}
}
```

Dependencies

Call code:MemcachedClient MemClient = MemCached.getInstance();

Download client

Download client

About client

Client versions

Environment configuration

Download, compile and install the C++ client.

https://launchpad.net/libmemcached/1.0/1.0.18/+download/libmemcached-1.0.18.tar.gz

Run the following command.

```
 tar -xvf libmemcached-1.0.18.tar.gz
cd libmemcached-1.0.18
```

```
./configure --enable-sasl
make
make install ((Sudo permission may be required)
```

C++ sample code

Download ocs_test.tar.gz.

Run the following command.

```
tar -xvf ocs_test.tar.gz
cd ocs_test
vim ocs_test_sample1.cpp
```

Modify TARGET_HOST to the address of the Memcache you have purchased, modify USERNAME and PASSWORD to your own user name and password.

Run build.sh to generate ocs_test. Run ./ocs_test and you will see a key written to Memcache. Get the key from Memcache, and delete it from Memcache.

The ocs_test_sample1.cpp code is as follows:

```
 #include <iostream>
#include <string>
#include <libmemcached/memcached.h>
using namespace std;

#define TARGET_HOST ""
#define USERNAME ""
#define PASSWORD ""

int main(int argc, char *argv[])
{
memcached_st *memc = NULL;
memcached_return rc;
memcached_server_st *server;
memc = memcached_create(NULL);
server = memcached_server_list_append(NULL, TARGET_HOST, 11211,&rc);

/* SASL */
sasl_client_init(NULL);
rc = memcached_set_sasl_auth_data(memc, USERNAME, PASSWORD);
if(rc != MEMCACHED_SUCCESS) {
cout<<"Set SASL err:"<< endl;
}
rc = memcached_behavior_set(memc,MEMCACHED_BEHAVIOR_BINARY_PROTOCOL,1);
```

```
if(rc != MEMCACHED_SUCCESS) {
cout<<"Binary Set err:"<<endl;
}
/* SASL */

rc = memcached_server_push(memc,server);
if(rc != MEMCACHED_SUCCESS) {
cout <<"Connect Mem err:"<< rc << endl;
}
memcached_server_list_free(server);

string key = "TestKey";
string value = "TestValue";
size_t value_length = value.length();
size_t key_length = key.length();
int expiration = 0;
uint32_t flags = 0;

//Save data
rc = memcached_set(memc,key.c_str(),key.length(),value.c_str(),value.length(),expiration,flags);
if (rc != MEMCACHED_SUCCESS){
cout <<"Save data failed: " << rc << endl;
return -1;
}
cout <<"Save data succeed, key: " << key << " value: " << value << endl;

cout << "Start get key:" << key << endl;

char* result = memcached_get(memc,key.c_str(),key_length,&value_length,&flags,&rc);
cout << "Get value:" << result << endl;

//Delete data
cout << "Start delete key:" << key << endl;
rc = memcached_delete(memc,key.c_str(),key_length,expiration);
if (rc != MEMCACHED_SUCCESS) {
cout << "Delete key failed: " << rc << endl;
}
cout << "Delete key succeed: " << rc << endl;
//free
memcached_free(memc);
return 0;
}
```

Following is a code example of using Memcache with another C++ program. Here we can see that Memcache cache and MySQL databases are used in combination. Compile and install a C++ client by following the same steps in the preceding example.

Create the example database and table in the MySQL database.

```
mysql -h host -uUSER -pPASSSWORD
create database testdb;
create table user_info (user_id int, user_name char(32) not null, password char(32) not null, is_online int,
```

```
primary key(user_id) );
```

Download ocs_test_2.tar.gz , and run the following command.

```
tar -xvf ocs_test_2.tar.gz
cd ocs_test
vim ocs_test_sample2.cpp
```

Note: Modify OCS_TARGET_HOST to the address of the Memcache you have purchased, modify OCS_USERNAME to the Memcache instance name, and modify OCS_PASSWORD to the set passwor. MYSQL_HOST is the MySQL address, MYSQL_USERNAME is the database user name, and MYSQL_PASSWORD is the database password.

Run build.sh to generate ocs_test, and then run /ocs_test.

The ocs_test_sample2.cpp code is as follows:

```
 #include <iostream>
#include <string>
#include <sstream>
#include <libmemcached/memcached.h>
#include <mysql/mysql.h>

using namespace std;

#define OCS_TARGET_HOST "xxxxxxxxxx.m.yyyyyyyyy.ocs.aliyuncs.com"
#define OCS_USERNAME "your_user_name"
#define OCS_PASSWORD "your_password"

#define MYSQL_HOST "zzzzzzzzzz.mysql.rds.aliyuncs.com"
#define MYSQL_USERNAME "db_user"
#define MYSQL_PASSWORD "db_paswd"
#define MYSQL_DBNAME "testdb"

#define TEST_USER_ID "100"

MYSQL *mysql = NULL;
memcached_st *memc = NULL;
memcached_return rc;

int InitMysql()
{
mysql = mysql_init(0);
if (mysql_real_connect(mysql, MYSQL_HOST, MYSQL_USERNAME, MYSQL_PASSWORD,
MYSQL_DBNAME, MYSQL_PORT, NULL, CLIENT_FOUND_ROWS) == NULL )
{
cout << "connect mysql failure!" << endl;
return EXIT_FAILURE;
}
cout << "connect mysql success!" << endl;
return 0;
```

```
}

bool InitMemcached()
{
memcached_server_st *server;
memc = memcached_create(NULL);
server = memcached_server_list_append(NULL, OCS_TARGET_HOST, 11211,&rc);

/* SASL */
sasl_client_init(NULL);
rc = memcached_set_sasl_auth_data(memc, OCS_USERNAME, OCS_PASSWORD);
if (rc != MEMCACHED_SUCCESS)
{
cout<<"Set SASL err:"<< endl;
return false;
}
rc = memcached_behavior_set(memc,MEMCACHED_BEHAVIOR_BINARY_PROTOCOL,1);
if (rc != MEMCACHED_SUCCESS)
{
cout<<"Binary Set err:"<<endl;
return false;
}
/* SASL */
rc = memcached_server_push(memc,server);
if (rc != MEMCACHED_SUCCESS)
{
cout <<"Connect Mem err:"<< rc << endl;
return false;
}
memcached_server_list_free(server);
return true;
}

struct UserInfo
{
int user_id;
char user_name[32];
char password[32];
int is_online;
};

bool SaveToCache(string &key, string &value, int expiration)
{
size_t value_length = value.length();
size_t key_length = key.length();
uint32_t flags = 0;
//Save data
rc = memcached_set( memc,key.c_str(), key.length(), value.c_str(), value.length(), expiration, flags);
if (rc != MEMCACHED_SUCCESS){
cout <<"Save data to cache failed: " << rc << endl;
return false;
}
cout <<"Save data to cache succeed, key: " << key << " value: " << value << endl;
return true;
}
```

```
UserInfo *GetUserInfo(int user_id)
{
UserInfo *user_info = NULL;
//get from cache
string key;
stringstream out;
out << user_id;
key = out.str();
cout << "Start get key:" << key << endl;
size_t value_length;
uint32_t flags;
char* result = memcached_get(memc, key.c_str(), key.size(), &value_length, &flags, &rc);
if (rc != MEMCACHED_SUCCESS)
{
cout << "Get Cache Failed, start get from mysql."<< endl;
int status;
char select_sql[1024];
memset(select_sql, 0x0, sizeof(select_sql));
sprintf(select_sql, "select * from user_info where user_id = %d", user_id);
status = mysql_query(mysql, select_sql);
if (status !=0 )
{
cout << "query from mysql failure!" << endl;
return NULL;
}
cout << "the status is :" << status << endl;
MYSQL_RES *mysql_result = mysql_store_result(mysql);
user_info = new UserInfo;
MYSQL_ROW row;
while (row = mysql_fetch_row(mysql_result))
{
user_info->user_id = atoi(row[0]);
strncpy(user_info->user_name, row[1], strlen(row[1]));
strncpy(user_info->password, row[2], strlen(row[2]));
user_info->is_online = atoi(row[3]);
}
mysql_free_result(mysql_result);
return user_info;
}
cout << "Get from cache succeed" << endl;
user_info = new UserInfo;
memcpy(user_info, result, value_length);
return user_info;
}

bool DeleteCache(string &key, int expiration)
{
rc = memcached_delete(memc, key.c_str(), key.length(), expiration);
if (rc != MEMCACHED_SUCCESS) {
cout << "Delete key failed: " << rc << endl;
return false;
}
cout << "Delete key succeed: " << rc << endl;
return true;
}
```

```
void PrintUserInfo(UserInfo *user_info)
{
cout << "user_id: " << user_info->user_id << " " << " name: " << user_info->user_name << endl;
}

bool SaveMysql(UserInfo *user_info)
{
char insert_sql[1024];
memset(insert_sql, 0x0, sizeof(insert_sql));
sprintf(insert_sql, "insert into user_info(user_id, user_name, password, is_online) values(%d, '%s', '%s',
%d)", user_info->user_id, user_info->user_name, user_info->password, user_info->is_online);
int status = mysql_query(mysql, insert_sql);
if (status != 0)
{
cout << "insert failed" << endl;
return false;
}
cout << "insert user_info" << endl;
//insert mysql
return true;
}

int main(int argc, char *argv[])
{
if (InitMysql() != 0)
{
return -1;
}
if (!InitMemcached())
{
return -1;
}

//generate user_info
UserInfo user_info;
user_info.user_id = atoi(TEST_USER_ID);
strcpy(user_info.user_name, "James");
strcpy(user_info.password, "12345678");
user_info.is_online = 1;

//save to mysql
if (!SaveMysql(&user_info))
{
//return -1;
}
string user_str;
user_str.assign((char*)&user_info, sizeof(UserInfo));
//save to memcached
string key_str = TEST_USER_ID;
SaveToCache(key_str, user_str, 10);

//start get, exist in memcahced
UserInfo *get_user_info = GetUserInfo(user_info.user_id);
PrintUserInfo(get_user_info);

//wait 10 secons
```

```
sleep(2);
//delete memcached or expired
DeleteCache(key_str, 0);

//start get, exist in mysql
delete get_user_info;
get_user_info = GetUserInfo(user_info.user_id);

PrintUserInfo(get_user_info);
delete get_user_info;
//free
memcached_free(memc);
mysql_close(mysql);
return 0;
}
```

About client

EnyimMemcachedCore is a Memcached client for migrating data from EnyimMemcached to .NET Core and supports .NET Core.

> Address of source code hosted on GitHub:
> https://github.com/cnblogs/EnyimMemcachedCore
>
> NuGet package address: https://www.nuget.org/packages/EnyimMemcachedCore

1. Install the NuGet package

```
Install-Package EnyimMemcachedCore
```

2. Configure NuGet

2.1 Configure in appsetting.json

Configuration of NuGet without verification

```
{
"enyimMemcached": {
"Servers": [
{
"Address": "memcached",
"Port": 11211
```

```
}
]
}
}
```

Configuration of NuGet with verification

```
{
"enyimMemcached": {
"Servers": [
{
"Address": "memcached",
"Port": 11211
}
],
"Authentication": {
"Type": "Enyim.Caching.Memcached.PlainTextAuthenticator",
"Parameters": {
"zone": "",
"userName": "username",
"password": "password"
}
}
}
}
```

Configuration code in Startup.cs

```
public class Startup
{
public void ConfigureServices(IServiceCollection services)
{
services.AddEnyimMemcached(options => Configuration.GetSection("enyimMemcached").Bind(options));
}

public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
app.UseEnyimMemcached();
}
}
```

2.2 Direct hard-coding configuration (no configuration files required)

Hard-coding configuration code in Startup.cs

```
public class Startup
{
public void ConfigureServices(IServiceCollection services)
{
```

```
services.AddEnyimMemcached(options =>
{
options.AddServer("memcached", 11211);
//options.AddPlainTextAuthenticator("", "usename", "password");
});
}

public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
app.UseEnyimMemcached();
}
}
```

3. Call

3.1 Use IMemcachedClient interface

```
public class TabNavService
{
private ITabNavRepository _tabNavRepository;
private IMemcachedClient _memcachedClient;

public TabNavService(
ITabNavRepository tabNavRepository,
IMemcachedClient memcachedClient)
{
_tabNavRepository = tabNavRepository;
_memcachedClient = memcachedClient;
}

public async Task<IEnumerable<TabNav>> GetAll()
{
var cacheKey = "aboutus_tabnavs_all";
var result = await _memcachedClient.GetAsync<IEnumerable<TabNav>>(cacheKey);
if (!result.Success)
{
var tabNavs = await _tabNavRepository.GetAll();
await _memcachedClient.AddAsync(cacheKey, tabNavs, 300);
return tabNavs;
}
else
{
return result.Value;
}
}
}
```

3.2 Use IDistributedCache interface(from Microsoft.Extensions.Caching.Abstractions)

```
public class CreativeService
```

```
{
private ICreativeRepository _creativeRepository;
private IDistributedCache _cache;

public CreativeService(
ICreativeRepository creativeRepository,
IDistributedCache cache)
{
_creativeRepository = creativeRepository;
_cache = cache;
}

public async Task<IList<CreativeDTO>> GetCreatives(string unitName)
{
var cacheKey = $"creatives_{unitName}";
IList<CreativeDTO> creatives = null;

var creativesJson = await _cache.GetStringAsync(cacheKey);
if (creativesJson == null)
{
creatives = await _creativeRepository.GetCreatives(unitName)
.ProjectTo<CreativeDTO>().ToListAsync();

var json = string.Empty;
if (creatives != null && creatives.Count() > 0)
{
json = JsonConvert.SerializeObject(creatives);
}
await _cache.SetStringAsync(
cacheKey,
json,
new DistributedCacheEntryOptions().SetSlidingExpiration(TimeSpan.FromMinutes(5)));
}
else
{
creatives = JsonConvert.DeserializeObject<List<CreativeDTO>>(creativesJson);
}

return creatives;
}
}
```

ECS Windows

Currently, ApsaraDB for Memcache is accessible via ECS intranet. If you need to access ApsaraDB for Memcache through the internet, you need to create port mapping via netsh on the ECS Windows server.

Log in to the ECS Windows server, and run the following command in CMD.

```
netsh interface portproxy add v4tov4 listenaddress=ECS server public IP address listenport=11211
connectaddress=ApsaraDB for Memcache connection address connectport=11211
```

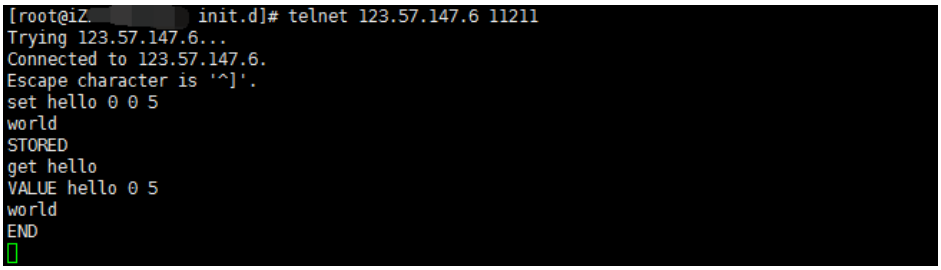
Note:

netsh interface portproxy delete v4tov4 listenaddress=public IP address of ECS listenport=11211 //You can delete unnecessary mappings.

netsh interface portproxy show all //You can view the mappings on the server.

Verify and test the mappings after settings are completed.

Connect to the ECS Windows server locally through the telnet and write data for query and verification. If the IP address of the ECS Windows server is 1.1.1.1, that is telnet 1.1.1.1 11211.



With the preceding steps, your local PC or server can be connected to the ECS Windows 11211 port over the Internet and access ApsaraDB for Memcache.

Note: Because the portproxy is provided by the Microsoft official team, and is not open-source, if you have any questions during configuration and usage, refer to the netsh portproxy usage instructions or consult with Microsoft. Or you can also consider other alternatives, such as configuring proxy mappings through portmap.

ECS Linux

Currently, ApsaraDB for Memcache is accessible via ECS intranet. If you need to access ApsaraDB for Memcache through the Internet, you need to install rinetd on the ECS Linux server for forwarding.

Install rinetd on the ECS Linux.

```
wget http://www.boutell.com/rinetd/http/rinetd.tar.gz&&tar -xvf rinetd.tar.gz&&cd rinetd
sed -i 's/65536/65535/g' rinetd.c (Modify the port range. Otherwise an error will be reported)
mkdir /usr/man&&make&&make install
```

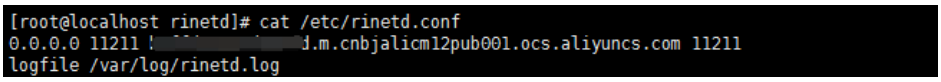
Note: The download address of the rinetd installer may not be available. You can search for the installer on your own for downloading and usage.

Create the configuration file.

```
vi /etc/rinetd.conf
```

Enter the following content.

```
 0.0.0.0 11211 [Memcache connection address] 11211
logfile /var/log/rinetd.log
```

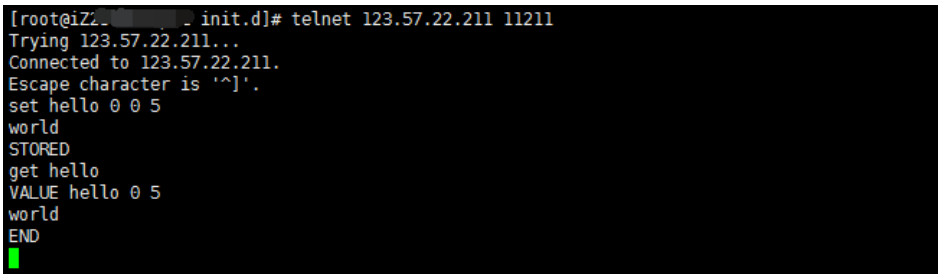


Execute the rinetd command to start rinetd.

Note: You can set it to auto start through echo rinetd >>/etc/rc.local. You can use pkill rinetd to end the process.

Test and verify.

Connect to the ECS Linux server locally through telnet and write data for query and verification. For example, if the IP address of the server with rinetd installed is 1.1.1.1, that is telnet 1.1.1.1 11211.



Through the proceding steps, your local PC or server can be connected to the ECS Linux 11211 port over the Internet and access ApsaraDB for Memcache.

Note: Because the rinetd is open-source software, if you have any questions during the usage, you can refer to its official documents or contact the official side of rinetd for confirmation.