How To Install and Use Memcache on Ubuntu 12.04

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: *http://faq.asphosthelpdesk.com/article.php?id=168*

About Memcache

Memcache is a system that works to speed up virtual private servers by caching server information. The program allows you to allocate a specific amount of the server ram toward caching recently queried data for a certain amount of time. Once the data is requested again, memcache speeds up the process of retrieving it by displaying the cached information instead of generating the result from the database.

Setup

The steps in this tutorial require the user to have root privileges. You can see how to set that up in the Basic Users Tutorial.

Before starting off, it"s a good idea to update apt-get to make sure that all of the packages we download to the server are up to date.

```
sudo apt-get update
```

Additionally, you should have MySQL and PHP installed on the virtual server.

```
sudo apt-get install mysql-server php5-mysql php5 php5-memcache
```

Install Memcache

Installing memcache takes several steps.

To start, install memcached via apt-get.

```
sudo apt-get install memcached
```

The next step is to install php-pear, the repository that stores memcache.

```
sudo apt-get install php-pear
```

If you do not have a compiler on your server, you can download build-essential in order to install memcache:

```
sudo apt-get install build-essential
```

Finally use PECL (PHP Extension Community Library) to install memcache:

```
sudo pecl install memcache
```

Say yes by pressing enter during the installation when you are asked if you would like to "Enable memcache

session handler support? [yes] :â€•

Once you have completed the installation of memcache with PECL on the server, add memcached to memcache.ini:

```
echo "extension=memcache.so" | sudo tee /etc/php5/conf.d/memcache.ini
```

Now you are ready to start using Memcache.

Confirm Memcache and See Stats

After Memcache is downloaded, you can check that it has been installed by searching for it:

```
ps aux | grep memcache
```

Additionally, you can see the memcache stats by typing:

```
 echo "stats settings" | nc localhost 11211
```

How Memcache Works

Memcache works by redirecting code to first attempt to retrieve data from the cache before querying the server"s database. The cache populates by saving recently retrieved server data for a certain amount of time. By caching recently requested information, future queries do not have to go through the longer process of retrieving the information from a database and can, instead, access it through the cache.

The memcache page shows this abbreviated code on its homepage to summarize the memcache process:

```
function get_foo(foo_id)
    foo = memcached_get("foo:" . foo_id)
    return foo if defined foo
    foo = fetch_foo_from_database(foo_id)
    memcached_set("foo:" . foo_id, foo)
    return foo
end
```

A Simple Memcache Example

This section will set up a simple php script to use memcache for retrieving a single value originally found in a mysql table.

The following steps set up a mysql user who can access the appropriate database, create a table to query, and insert the one value that we will test in the new mysql table.

Log into mysql: `mysql -u root -p` and execute the following commands:

```
use test;
grant all on test.* to test@localhost identified by 'testing123';
create table example (id int, name varchar(30));
insert into example values (1, "new_data");
exit;
```

Once you have exited MySQL, create the memcache script file:

```
nano memtest.php
```

We are now going to build up the php script step by step (the entire script will be at the end of the section):

```
 Start off by creating a new persistent connection with memcache, which runs on
 memcache"s default port, 11211.
 <?php
 $meminstance = new Memcache();
 $meminstance->pconnect('localhost', 11211);
```

```
The next step is to connect to the new mysql database with the user that we
created earlier:
mysql_connect("localhost", "test", "testing123") or die(mysql_error());
mysql_select_db("test") or die(mysql_error());
```

```
After that, go ahead and create the query that we will pose to the server, as
well as provide a key to identify that specific action:
$query = "select id from example where name = 'new_data'";
$querykey = "KEY" . md5($query);
```

```
The script first searches the cache for the answer to the query. If the result
does not exist, the script reroutes the question to the original database. Once
the query has been answered by the original database, the script stores the
result in memcache, using the "setâ€• command-- which both saves it and allows
the user to designate the number of seconds that it should remain in the cache
(600 would save it in the cache for 10 minutes).

 When we run the script for the first time, it will inform us that the data was
collected from the mysql database. However, as it does so, it stores the
information in the cache, so that a second run of the script retrieves it from
the cache and lets the user know.

 In 10 minutes the cache is emptied once more and running the script will make
it access the database once again.
```

```
$result = $meminstance->get($querykey);
if (!$result) {
       $result = mysql_fetch_array(mysql_query("select id from example where
name = 'new_data'")) or die('mysql error');
```


PDF Generated by PHPKB Knowledge Base Script

```
       $meminstance->set($querykey, $result, 0, 600);
print "got result from mysql\n";
return 0;
}
print "got result from memcached\n";
return 0;
?>
```

-

Altogether the script looks like this:

```
<?php
$meminstance = new Memcache();
$meminstance->pconnect('localhost', 11211);
mysql_connect("localhost", "test", "testing123") or die(mysql_error());
mysql_select_db("test") or die(mysql_error());
$query = "select id from example where name = 'new_data'";
$querykey = "KEY" . md5($query);
$result = $meminstance->get($querykey);
if (!$result) {
   $result = mysql_fetch_array(mysql_query("select id from example where name =
'new_data'")) or die('mysql error');
   $meminstance->set($querykey, $result, 0, 600);
print "got result from mysql\n";
return 0;
}
print "got result from memcached\n";
return 0;
?>
```

Running the script on the command line produces the following results:

```
# php memtest.php
got result from mysql
# php memtest.php
got result from memcached
# php memtest.php
got result from memcached
```