GETTING STARTED WITH

Memcached

BY JAMES SUGRUE

CONTENTS

INTRODUCTION

This memcached Refcard provides you with basic configuration information for servers and client-side commands to use memcached as a caching solution for your applications.

WHAT IS MEMCACHED?

memcached is defined as "a high performance, distributed memory object caching system, generic in nature, but originally intended for use in speeding up dynamic web applications by alleviating database load" by memcached.org.

memcached works by caching data in RAM so that the number of times an application has to read an external data source is reduced. The in-memory key-value store can be used for data that would be returned from database or API calls. On top of all of this, memcached allows the cache load to be distributed across servers. However, as the data is stored in memory by default, when the server is restarted, memcached will re-initialize with an empty dataset.

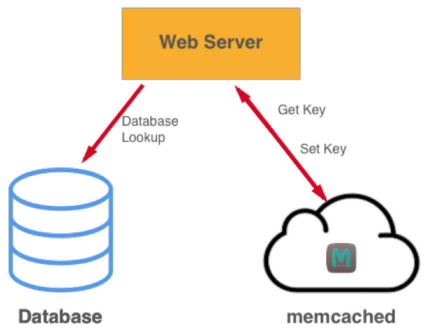


***Diagram 1**: Illustration of how a web server interacts with memcached and a database*

LIMITATIONS

There are four key limitations in memcached to keep in mind:

1. The key used is a string with a maximum length of 250 bytes
2. The value has a 1 megabyte size limit, which can be increased with the –I parameter
3. No persistence
4. it is not highly available

INSTALLING MEMCACHED

To install memcached on your Linux-based system, use either of the following commands, depending on the OS that it is being installed to:

- `apt-get install memcached (Debian / Ubuntu)`
- `yum install memcached (Red Hat / Fedora)`

Once installed you can run memcached using the following command, where –d causes memcached to run as a daemon process:

```
memcached -p <TCP port> -U <UDP port> -u <username> -d
```

***Tip:** Use memcached –h for up-to-date documentation on the memcached version you have installed on your server*

SERVER CONFIGURATION

The configuration file for memcached is available at `/etc/sysconfig/memcached` with defaults set as follows:

```
PORT="11211"
USER="memcached"
MAXCONN="1024"
CACHESIZE="64"
OPTIONS=""
```

The `PORT` value specified in the above configuration file is the same for TCP and UDP ports.

`MAXCONN` represents the maximum number of concurrent connections for memcached. Changing it to a higher value is generally okay. Use the stats command to look for the `listen_disabled_num` attribute. The value for this should be zero. If the number is higher, it indicates that connections have been dropped, and you should then consider increasing the MAXCONN value.

Running multiple instances of memcached on the same server is simply a matter of changing the port that memcached is listening on.

CONNECTING TO YOUR MEMCACHED SERVER

The most straightforward way to connect to memcached and check that it is running okay is to use telnet:

```
telnet <host> <port>
```

Amazon
ElastiCache
μs is the new ms

Fully managed Memcached by AWS

Amazon ElastiCache is a fully-managed service for Redis and Memcached that makes it easy to deploy, operate, and scale an in-memory data store or cache in the cloud with microsecond response times.

- Fully managed Memcached
- Hardened by Amazon
- AWS integration and support

Try it today for free: aws.amazon.com/elasticache

Try it today for Free »

amazon web services

Commands can be run from this telnet session, as outlined in later sections of this Refcard.

Note that your client-side code must be written to take advantage of memcached; it is not a transparent implementation, and as such, caching is not done automatically on the server-side. Libraries are available for the most popular programming languages so you can use memcached in your application, as can be seen in the following table. The commands outlined in the following sections will be exposed as functions in these libraries.

LANGUAGE	LIBRARY	LINK
Ruby	Dalli	github.com/petergoldstein/dalli
Go	gomemcache	github.com/bradfitz/gomemcache
C/C++	libMemcached	libmemcached.org/libMemcached.html
Python	Pymemcache	github.com/pinterest/pymemcache
Java	Spymemcached	github.com/couchbase/spymemcached
Node.js	Memcached	github.com/3rd-Eden/memcached
PHP	php-memcached	github.com/php-memcached-dev/php-memcached

***Table 1**- A selection of memcached client libraries*

CLIENT CONFIGURATION

The typical use case for applications using memcached client libraries is as follows:

1. Web application requests keys using a client library. The library will calculate key hash values (consistent hashing algorithm is the most common example), and thus determine which memcached server to send the request to
2. Client library sends parallel requests to all servers identified in the previous step
3. Responses are sent to the client library from the server

Some client libraries will allow you to apply weight to preferred servers, either by applying a weight value, or by adding that server multiple times to your client configuration. An auto-discovery feature of a memcached client allows you to discover the nodes within the memcached cluster so their IPs do not need to be hardcoded within the application. This is a very useful feature to enable dynamically supported topology changes.

USING MEMCACHED WITH MYSQL

Paired with MySQL, or another database, memcached can provide an efficient way to access data. The typical usage patterns are:

- Use MySQL for persistent data and memcached for transient data only.
- Clients read data from memcached, but if there is a cache miss, the client retrieves the data from the MySQL database and then stores the returned value in the cache, for efficient access in subsequent executions.

COMMON COMMANDS

The following attributes are used across a number of the following commands:

ATTRIBUTE	DESCRIPTION
key	Name of the unique key that will be used to access the data. Limited to 250 bytes.
flag	A 32-bit space stored alongside the main value. Many sub libraries make use of this field, so in most cases it should be avoided, by setting 0 as the flag value.
exptime	Expiration time in seconds of the data stored in cache. Use 0 to never expire. Using more than 30 days will be interpreted as a UNIX timestamp for an exact date to expire.
bytes	The length in bytes that needs to be allocated for this value.
noreply	Optional parameter that ensures no reply is sent from server following command execution.
value	Value to be stored, which needs to be provided in a new line following the set command with options.
unique_cas_token	A unique token number which can be retrieved using the gets command.

***Table 2**- Common command attributes used in storage and retrieval*

STORAGE COMMANDS

The following commands allow you to add and replace values in your memcached store:

SET

Set is the main command that you will use when adding data to your data store, which simply assigns a value to a given key. This command may overwrite existing data. The new items are at the top of the LRU.

```
set key flags exptime bytes [noreply]
value
```

ADD

Stores the value provided only if it does not already exist. New items are at the top of the LRU. If an item exists, and the add command fails, the item is placed to the front of the LRU anyway.

```
add key flags exptime bytes [noreply]
value
```

REPLACE

Stores the value provided only if it already exists.

```
replace key flags exptime bytes [noreply]
value
```

APPEND

Add this value after the last byte in an existing item. If the specified key does not exist, the command will fail.

```
append key flags exptime bytes [noreply]
value
```

PREPEND

Prepend this value before the first byte in an existing item. If the specified key does not exist, the command will fail.

```
prepend key flags exptime bytes [noreply]
value
```

CAS

Stored data but only if no one else has updated the data since you read it last. Abbreviation for "Check And Set" (or "Compare And Swap"). This command is useful to resolving race conditions on updating cache data. If the `unique_cas_token` for the item has changed since you last retrieved it (using gets) it will not be stored.

```
cas key flags exptime bytes unique_cas_token [noreply]
value
```

RETRIEVAL COMMANDS

The following commands allow you to retrieve values from your memcached store.

GET

Command for retrieving data for a given key, or set of keys. If a key does not exist, no value is returned.

```
get key [key2 .. keyn]
```

GETS

Get command to be used with CAS, which returns the unique_cas_token with the item. ICommand for retrieving data for a given key, or set of keys. If a key does not exist, no value is returned.

```
gets key [key2 .. keyn]
```

DELETE

Removes an item from the cache if it exists.

```
delete key [noreply]
```

INCR/DECR

These commands will either increment or decrement the numeric value of an existing key. These commands will only succeed where the key is a numeric value.

```
incr key increment_value
decr key decrement_value
```

FLUSH_ALL

Invalidates all cache items in your memcached server. This command takes an optional time parameter, which allows you to instruct memcached to clear this data after a number of seconds.

```
Flush_all [time]
```

STATISTICS COMMANDS

The following commands allow you to inspect settings on the memcached server. Stats commands give some insights on how performance of your setup could be tuned, as well as monitoring the overall usage of the cache.

STATS

The basic stats command which returns values for server uptime, memcached version, number of connections, number of items, and other traffic-related statistics.

```
stats
```

STATS ITEMS

Returns information about items stored in memcached, broken down by slab id. Returns values for items, age, and cache eviction.

```
stats items
```

STATS SLABS

Returns information about items stored in memcached, broken down by slab id. Values returned in this command are more focused around performance such as memory usage, allocated memory, and number of free chunks.

```
stats items
```

STATS SIZES

Illustrates how items would be distributed if broken into 32 byte buckets instead of your current number of slabs, helping you to determine how efficient your slab sizing is.

```
stats sizes
```

MEMCACHED TOOLS

Although some level of debugging is possible using the commands listed above with the telnet client, there are a number of other tools available to help monitor and troubleshoot your distributed memcached setup. The following is a summary of the most popular tools:

MEMCACHE-TOP

Link: https://code.google.com/archive/p/memcache-top/

Command line tool that displays real-time stats from memcached, displaying the results in a view similar to top. This tool can only monitor a single memcached instance at a time. Results include I/O throughput and evictions per second along with gets and sets per second.

PHPMEMCACHEDADMIN

Link: https://blog.elijaa.org/phpmemcachedadmin-download/

A standalone web-based application that displays memcached stats, including details on server slabs, memory wasted, and item key values. You can execute commands on a memcached server from this interface. Allows the monitoring of multiple memcached instances.

MEMCACHED-MANAGER

Link: https://github.com/memcached-manager/memcached-manager

A single-page Sinatra memcached manager/admin that allows you to read stats, view, edit, and create keys from memcached. Can easily be plugged into a Rails/Rack app and can work with multiple instances of memcached.

REPLICATION

Replication is not supported out-of-the-box in memcached, as it is designed to be a transient cache store. However, replication can be achieved through some third-party solutions.

REPCACHED

Link: https://github.com/ignacykasperowicz/repcached

Repcached allows you to have a redundant memcached installation running on another server. This is done by applying a patch that adds a new –x parameter, which accepts the IP address of the other server. Note that each server will need to have a link to the other.

YRMCDS

Link: http://cybozu.github.io/yrmcds

Developed mainly for session storage, yrmcds (Ymmt's Replicatin MemcacheD for Sessions) is completely compatible with memcached, and as such, can be used as a drop-in replacement. To enable replication, a virtual IP address needs to be specified along with the configuration of clustering software such as keepalived.

TWEMPROXY

Link: https://github.com/twitter/twemproxy

A lightweight proxy for memcached as well as Redis, built by Twitter to reduce the number of open connections to cache servers. Requests to a pool of memcached instances go through this proxy. Using two twemproxy instances, traffic can be directed to both a pool of slaves, as well as a pool of master instances.

PERFORMANCE BEST PRACTICES

Following these best practices will help you get the most from memcached:

1. From `stats`, take note of the `evictions` count, which shows the number of non-expired items that were removed from the cache to make space for new items. If this number is high, it indicates that the memory allocated for items storage is too low.
2. The default number of threads is 4, and for most cases should not be changed. Using a single thread will be too slow, and more than 8 threads can lead to high lock contention.
3. Retrieve values in bulk when possible rather than doing a number of get calls in series.
4. Compressing large values will speed up your application as there is less data being transmitted over the wire, and less memory being used when storing the value in memcached. Most clients support enabling or disabling compression using either an item size threshold or even on a per-item basis.
5. When initializing memcached, you can prepopulate the cache with keys, meaning your application will have less cache misses in its initial execution after a redeploy on the server.

SECURITY

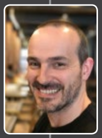
The ports on which your memcached instances are running should not be exposed to the internet as a rule. Since version1.4.3, SASL (Simple Authentication and Security Layer) is included. Note that SASL restricts access to the daemon but does not hide communications.

Tip: *Enable SASL on memcached by using the –S flag on startup. Following this, all commands require that authentication be successful before they are issued on a connection.*

To increase your memcached security beyond SASL, the following should be considered:

- Never run memcached as root. If someone did gain access to memcached and there was a security vulnerability, then they would be able to compromise your machine and network.
- Use a firewall to limit which connections are open to the outside world. If you are running memcached on a single server, then you can bind the instance to localhost using the –l parameter, restricting cache access to just that machine.

ABOUT THE AUTHOR



JAMES SUGRUE is Chief Architect at Over-C, building mobile applications and services for managing compliance, using NFC and Bluetooth sensors for proof of presence. James is an expert in Java and Swift, building everything from desktop applications to high performance servers and mobile applications in Android and iOS.

James has been a Zone Leader at DZone since 2008, and has written many Refcardz, as well as a book on Backbone.js.

DZONE, INC.
150 PRESTON EXECUTIVE DR.
CARY, NC 27513

888.678.0399
919.678.0300

REFCARDZ FEEDBACK WELCOME
refcardz@dzone.com

SPONSORSHIP OPPORTUNITIES
sales@dzone.com