

Memcached magic

Ligaya Turmelle

What is memcached briefly?

- memcached is a high-performance, distributed memory object caching system, generic in nature
- AKA it is a key-based cache daemon that stores data and objects wherever dedicated or spare RAM is available for very quick access
- It is a dumb distributed hash table. It doesn't provide redundancy, failover or authentication. If needed the client has to handle that.

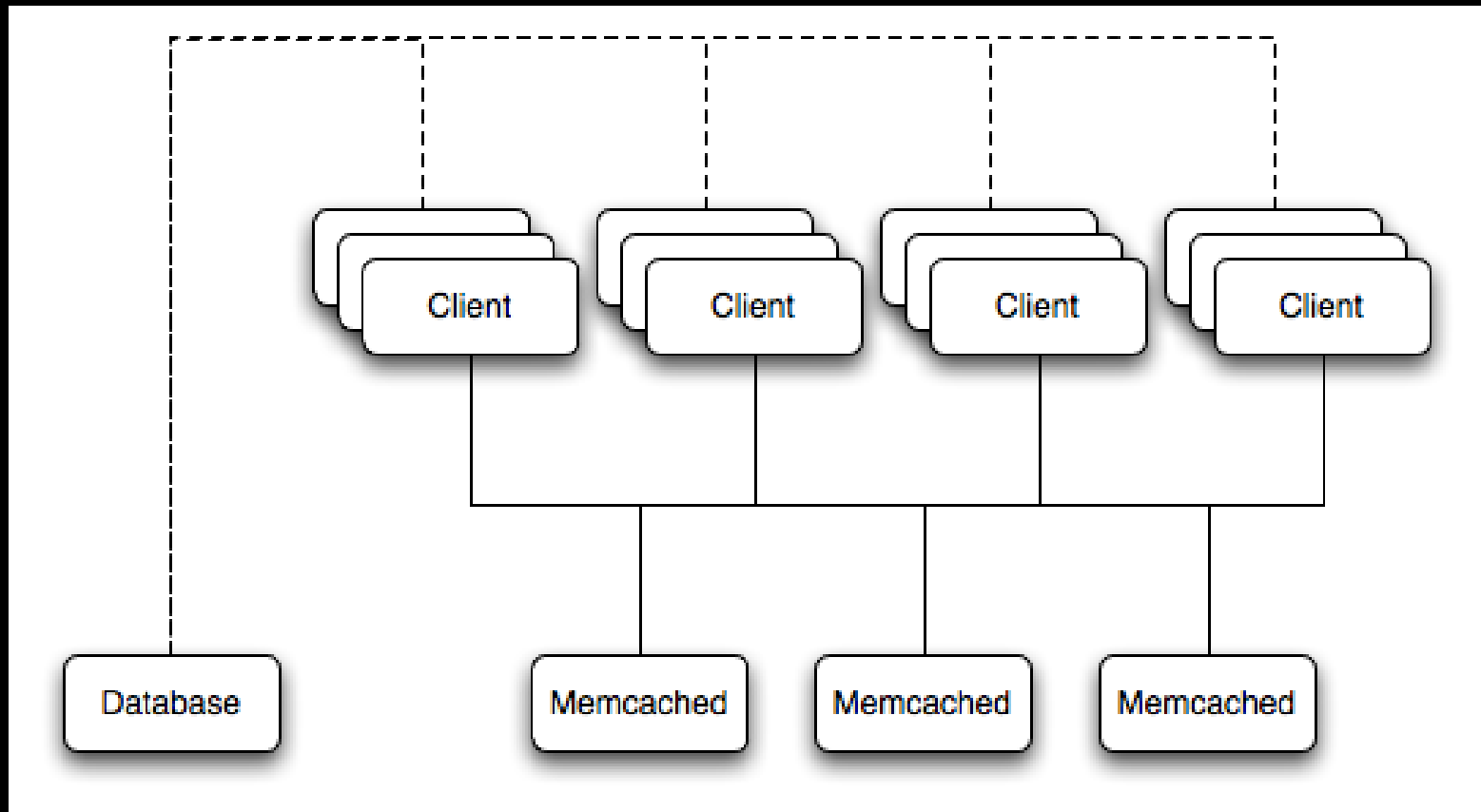
Why was memcached made?

- It was originally developed by Danga Interactive to enhance the speed of LiveJournal.com
- It dropped the database load to almost nothing, yielding faster page load times for users, better resource utilization, and faster access to the databases on a memcache miss
- <http://www.danga.com/memcached/>

Where does memcached reside?

- Memcache is not part of the database but sits outside it on the server(s).
- Over a pool of servers

Pretty picture



When should I use memcached?

- When your database is optimized to the hilt and you still need more out of it.
 - Lots of SELECTs are using resources that could be better used elsewhere in the DB.
 - Locking issues keep coming up
- When table listings in the query cache are torn down so often it becomes useless
- To get maximum “scale out” of minimum hardware

Use cases

- anything what is more expensive to fetch from elsewhere, and has sufficient hitrate, can be placed in memcached
 - How often will object or data be used?
 - How expensive is it to generate the data?
 - What is the expected hitrate?
 - Will the application invalidate the data itself, or will TTL be used?
 - How much development work has to be done to embed it?

Why use memcached?

- To reduce the load on the database by caching data BEFORE it hits the database
- Can be used for more than just holding database results (objects) and improve the entire application response time
- Feel the need for speed
 - Memcache is in RAM - much faster than hitting the disk or the database

Why not use memcached?

- Memcache is held in RAM. This is a finite resource.
- Adding complexity to a system just for complexities sake is a waste. If the system can respond within the requirements without it - leave it alone

What are the limits of memcached?

- Keys can be no more than 250 characters
- Stored data can not exceed 1M (largest typical slab size)
- There are generally no limits to the number of nodes running memcache
- There are generally no limits to the amount of RAM used by memcache over all nodes
 - 32 bit machines do have a limit of 4GB though

How do I easily install memcached?

- You can build and install memcached from the source code directly, or you can use an existing operating system package or installation.
 - on a RedHat, Fedora or CentOS host, use yum:
 - root-shell> yum install memcached
 - on a Debian or Ubuntu host, use apt-get:
 - root-shell> apt-get install memcached
 - on a Gentoo host, use emerge:
 - root-shell> emerge install memcached
 - on OpenSolaris, use the pkg for SUNWmemcached:
 - root-shell> pkg install SUNWmemcached

How do I compile memcached from source?

- Get the source from the website:
<http://www.danga.com/memcached/download>
 - Memcache has a dependancy on libevent so make sure you have that also.
- Decompress, cd into the dir
- `./configure;make;make install;`

How do I start memcached?

- Memcached can be run as a non-root user if it will not be on a restricted port (<1024) - though the user can not have a memory limit restriction
- shell> memcached
- Default configuration - Memory: 64MB, all network interfaces, port:11211, max simultaneous connections: 1024

Memcached options

- You can change the default configuration with various options.
 - -u <user> : run as user if started as root
 - -m <num> : maximum <num> MB memory to use for items
 - If more than available RAM - will use swap
 - Don't forget 4G limit on 32 bit machines
 - -d : Run as a daemon
 - -l <ip_addr> : Listen on <ip_addr>; default to INDRR_ANY
 - -p <num> : port

Memcached options (con't)

- `-s <file>` : unix socket (disables network support)
- `-c <num>` : max simultaneous connections
- `-v` : verbose
- `-vv` : (2 v's) very verbose
- `-P <file>` : PID file (used with `-d`)
- `-t <threads>` : number of threads to use to process incoming requests. Only valid if compiled with thread support enabled. Only useful up to number of CPU's
- There is a man page available with the install with a full listing of options.

How can I connect to memcached?

- Memcached uses a protocol that many languages implement with an API.
- Languages that implement it:
 - Perl, PHP, Python, Ruby, Java, C#, C, Lua, Postgres, MySQL, Chicken Scheme
- And yes - because it is a protocol you can even use telnet
 - `shell> telnet localhost 11211`

Memcached protocol

- 3 types of commands
 - Storage - ask the server to store some data identified by a key
 - set, add, replace, append, prepend and cas
 - Retrieval - ask the server to retrieve data corresponding to a set of keys
 - get, gets

Memcached protocol (con't)

- All others that don't involve unstructured data
 - Deletion: delete
 - Increment/Decrement: incr, decr
 - Statistics: stats,
 - flush_all: always succeeds, invalidate all existing items immediately (by default) or after the expiration specified.
 - version, verbosity, quit

PHP and Memcached

- Make sure you have a working Apache/PHP install
- PHP has a memcached extension available through pecl.
- Installation:
 - `shell> pecl install memcache`
 - Make sure the pear is installed (debian: `apt-get install php-pear`)
 - Make sure that you also have `php5-dev` installed for `phpize`.
 - `shell> apt-get install php5-dev`

PHP Script

- Information about the PHP API at <http://www.php.net/memcache>

```
<?php
// make a memcache object
$memcache = new Memcache;
// connect to memcache
$memcache->connect('localhost', 11211) or die ("Could not connect");
//get the memcache version
$version = $memcache->getVersion();
echo "Server's version: ".$version."<br/>\n";
```

PHP Script (con't)

```
// test data
$tmp_object = new stdClass;
$tmp_object->str_attr = 'test';
$tmp_object->int_attr = 123;
// set the test data in memcache
$memcache->set('key', $tmp_object, false, 10) or die ("Failed to save
    data at the server");
echo "Store data in the cache (data will expire in 10 seconds)<br/>\n";
// get the data
$get_result = $memcache->get('key');
echo "Data from the cache:<br/>\n";
echo '<pre>', var_dump($get_result), '</pre>';
```

PHP Script (con't)

```
// modify the data
```

```
$tmp_object->str_attr = 'boo';
```

```
$memcache->replace('key', $tmp_object, false, 10) or die("Failed to  
save new data to the server<br/>\n");
```

```
Echo "Stored data in the cache changed<br/>\n";
```

```
// get the new data
```

```
$get_result = $memcache->get('key');
```

```
Echo "New data from the cache:<br/>\n";
```

```
Echo '<pre>', var_dump($get_result), "</pre>\n";
```

```
// delete the data
```

```
$memcache->delete('key') or die("Data not deleted<br/>\n");
```

MySQL's memcached UDF's

- Found at http://tangent.org/586/Memcached_Functionior
- Will have to compile and build the functions
 - `./configure; make; make install;`
- May wish to copy the MySQL memcached UDFs into your MySQL plugins directory

MySQL's memcached UDF's (con't)

- you must initialize all the functions that you wish to have accessible within MySQL using CREATE and specifying the return value and library.
(retained after restarts)
 - Ex: `mysql> CREATE FUNCTION memc_get RETURNS STRING SONAME "libmemcached_functions_mysql.so";`
- There is an SQL script in the UDF package to do this if you wish.
 - `shell> mysql < sql/install_functions.sql`

MySQL's memcached UDF's (con't)

- The API is consistent with the other API's
 - Connect: `mysql> SELECT memc_servers_set('192.168.0.1:11211,192.168.0.2:11211');`
 - The list of servers used by the memcached UDFs is not persistent over restarts of the MySQL server.
 - Set: `mysql> SELECT memc_set('myid', 'myvalue');`
 - Retrieve: `mysql> SELECT memc_get('myid');`

Possible ways to secure memcached

- It has no authentication system - so protection is important
- Run as a non-privileged user to minimize potential damage
- Specify the ip address to listen on using -l
 - 127.0.0.1, 192.168.0.1, specific ip address
- Use a non-standard port
- Use a firewall

Memcache project

- Project home:

<http://code.google.com/p/memcached/>

- Bug reports:

<http://code.google.com/p/memcached/issues/>

Memcached project (con't)

- SVN Source code:

- Browse at

<http://code.google.com/p/memcached/source/browse/>

- Use this command to anonymously check out the latest project source code:

```
svn checkout http://memcached.googlecode.com/svn/trunk/  
memcached-read-only
```

- Non-members may check out a read-only working copy anonymously over HTTP.

Questions?
