S Acharya

Using Memcached with Java

August 10, 2009

Why not JBoss Cache?
By default, if you are looking for a caching solution for your Java based enterprise application, the tendency is to go with Java Caches. I have been using JBoss Cache for a couple of years now. It is a very powerful smart cache, which provides clustering, synchronized replication and transaction support. Meaning, given a cluster of JBoss cache, each instance is aware of the others and will be kept in sync. That way, if one of the instance is down, other instances still be serving your data.

Having been plagued with memory problems over and over again, I finally gave up on JBoss Cache and decided to go with a a simple and dumber solution – Memcached.

Memcached is widely popular esp. in the PHP and Rails community. My main reasons for switching from JBoss Cache to Memcached are:

1. JBoss Cache is replicated, so there is the overhead of syncing the nodes. All the nodes try to keep the same state. Memcached is distributed and each node is dumb about the other nodes. Each piece of data lives in only one of the nodes. And the nodes don't know about each other. If one node fails, only some hits are missed. While this may seem like a disadvantage, it is actually a blessing if you are willing to give up the complexity for simplicity and ease of maintenance.

2. JBoss cache comes with a pretty complicated configuration. Memcached doen't require any configuration.

3. JBoss Cache lives in your JVM, and you have to tune the JVM for optimum memory, which isnt always fun as the nature and amount of your data changes . Memcached uses the amount RAM you specify. If the memory becomes full, it will evict older data based on LRU.

In short, the fact that Memcached is so simple and requires almost no maintenance was a big big win for me. However, if your application is such that the sophisticated caches makes sense, you should definitely consider using them.

Memcached:

Memcached server (protocol defined here) is an in memory cache that stores anything from binary to text to primitives associated with a key as a Key-Value pair. Like with any other caches, storing data in memory prevents you from going to the database or fileserver or any backend system everytime a user requests for the data. That saves a lot of load of your backend systems, leading to higher scalability. Since the data is stored in memory, it is generally faster than making an expensive backend call too.

However, Memcached is not a persistent store, and doesn't guarantee something will be in the cache just because you stored it. So you should never rely on the fact that Memcached is storing your data. Memcached should strictly be used for caching purposes only, and not for reliable storage.

The only limitation with Memcached (that you need to be aware of) is that the key in memcached should be less that 255 chars and each value shouldn't exceed 1 MB.

Installation:
1. Install Libevent
Memcached uses the Libevent library for network IO.

```
$ cd libevent-1.4.11-stable
$ autoconf
$ ./configure --prefix=/usr/local
$ make
$ sudo make install
```

2. Install Memcached:
Download the latest version of Memcached from Danga.com who developed Memcached originally for Livejournal.

```
$ cd memcached-1.4.0
$ autoconf
$ ./configure --prefix=/usr/local
$ make
$ sudo make install
```

3. Run memcached:
Start memcached as a daemon with 512MB of memory on port 11211(default). Then you can telnet to the server and port and use any of the available commands.

```
$memcached -d -m 512 127.0.0.1 -p 1121

$ telnet localhost 11211
Trying ::1...
Connected to localhost.
Escape character is '^]'.
get joe
END
set joe 0 3600 10  (Note: TTL 3600 and 10 bytes)
California
STORED
get joe
VALUE joe 0 10
California
END
```

Spy Memcached (Memcached Java Client):
Basic Usage:

There are a few good java clients for Memcached. I briefly looked at the Whalin's Memcached Client and Dustin's SpyMemcached Client, and decided to go with the latter for minor reasons.You can start with the API as shown in the docs:

```
MemcachedClient c=new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
c.set("someKey", 3600, someObject);
Object myObject=c.get("someKey");
c.delete("someKey")
```

The MemcachedClient is a single-threaded client to each of the Memcached server in the pool. The set method sets an object in the cache for a given key. If a value already exists for the key, it overwrites the value. It takes a timeToLive value in seconds, which is the expiration date for the object. Even though there are many requests comings, the client handles only one thread at a time, while the rest wait in the queue. The get method retrieves the object based on the unique queue, and the delete method is used to delete the value.

There are other methods available for storage, retrieval and update but you will get by most of the times just with the three methods get, set and delete.

Security:

By design, memcached Server doesn't have any authentication around it. So its your job to secure the memcached server or the port from outside network. Furthermore just to obscure the key, you can prefix your key with some secret code or use the hash of the key as the key.

For example:

```
String randomCode = "aaaaaaaaaaaaaaaaaaaa";
c.set(randomCode + "someKey", 3600, someObject);
Object myObject=c.get(randomCode + "someKey");
```

Adding/Removing a cache server:

If you need to upscale and want to add a new memcached server, you just need to add the server ip and port to the pool of existing servers, and the memcached client will take it into account. If you want to downscale and get rid of a server, just remove the server from the pool. There will be cache misses for the data living on the server for a while, but cache will soon recover itself as it will starting caching the data onto other available servers. Same thing will happen if you lose connectivity to one of the servers. If you are worried about flooding the database when you lose a memcached server, you should have the data pre-fetched onto another server. However, the memcached server themselves don't know anything about each others. Its all the function of the client.

```
MemcachedClient c =  new MemcachedClient(new BinaryConnectionFactory(),
                        AddrUtil.getAddresses("server1:11211 server2:11211"));
```

Connection Pooling:

The MemcachedClient establishes TCP connection (Facebook has released a modified version of memcached to use UDP to reduce the number of connections) open to the memcached server.So you might want to know how many connections are being used.

```
$ netstat -na | grep 11211
tcp4       0      0  127.0.0.1.11211        127.0.0.1.59321        ESTABLISHED
tcp4       0      0  127.0.0.1.59321        127.0.0.1.11211        ESTABLISHED
```

There is really no way to explicitly close the TCP connections. However, each get or set is atomic in itself. There is no really harm to opening as many TCP connections as you like as Memcached is designed to work well with large number of open connections.
Update: As Matt said in the comments below, each connection is is asynchronous and non-blocking. So while there is one thread per node, requests are multiplexed. So there is very little benefit to doing explicit pooling with multiple copies of the client connections.

MyCache Singleton:

So with all the changes, here's what my wrapper around MemcachedClient looks like:

```
import net.spy.memcached.AddrUtil;
import net.spy.memcached.BinaryConnectionFactory;
import net.spy.memcached.MemcachedClient;

public class MyCache {
    private static final String NAMESPACE= "SACHARYA:5d41402abc4b2a76b9719d91101";
    private static MyCache instance = null;
    private static MemcachedClient[] m = null;

    private MyCache() {
        MemcachedClient c =  new MemcachedClient(
                        new BinaryConnectionFactory(),
             AddrUtil.getAddresses("127.0.0.1:11211"));
                return c;
    }

    public static synchronized MyCache getInstance() {
        System.out.println("Instance: " + instance);
        if(instance == null) {
            System.out.println("Creating a new instance");
            instance = new MyCache();
         }
         return instance;
    }

    public void set(String key, int ttl, final Object o) {
        getCache().set(NAMESPACE + key, ttl, o);
    }

    public Object get(String key) {
        Object o = getCache().get(NAMESPACE + key);
        if(o == null) {
            System.out.println("Cache MISS for KEY: " + key);
        } else {
            System.out.println("Cache HIT for KEY: " + key);
        }
        return o;
    }

    public Object delete(String key) {
```

```
        return getCache().delete(NAMESPACE + key);
    }


    public MemcachedClient getCache() {
        MemcachedClient c= null;
        try {
            int i = (int) (Math.random()* 20);
            c = m[i];
        } catch(Exception e) {

        }
        return c;
    }
}
```

In the above code:
1. I am using the BinaryConnectionFactory (which is a new feature) that implements the new binary wire protocol which provides more efficient way of parsing the text.

2. MyCache is a singleton, and it sets up 21 connections when it is instantiated.

3. My keys are of the format: SACHARYA:5d41402abc4b2a76b9719d91101:key where SACHARYA is my domain. That way I can use the same memcached server to store data for two different applications. The random staring 5d41402abc4b2a76b9719d911017c592 is just for some security through obscurity which we discussed above. Finally the key would be something like userId or username or a sql query or any string that uniquely identifies the data to be stored.

Sample Use:

Generally you can use caching wherever there is bottleneck. I use it at the Data Access Layer layer for saving myself from making a database or a webservice call. If there is a computation-heavy business logic, I cache the output at the business layer. Or you can cache at the presentation layer. Or you can cache at every layer. It all depends on what you are trying to achieve.

```
public List<Product> getAllProducts() {
        List<Product> products = (List<Product>) MyCache.getInstance().get("AllProducts");
        if(products != null) {
              return products;
        }
        products = getAllProductsFromDB()
        if(products) {
              MyCache.getInstance().put("AllProducts", 3600, customer);
        }
        return products;
}

public void updateProduct(String id) {
        updateProductIntoDB(id)
        MyCache.getInstance().delete("AllProducts");
}
public void deleteProduct(String id) {
        deleteProductFromDB(id)
        MyCache.getInstance().delete("AllProducts");
}
```

Warming the Cache:

When the application is first started, there is nothing in the cache. So you might want to pre-warm the cache through a job scheduler, just to avoid large no of backend calls at once. I generally like to put this piece put outside of the application itself. It could be a separate app in itself where you prewarm the cache based on the hit-list of keys.

Measuring Cache Effectiveness:

The stats command provides important information about how your cache is performing. Among other parameters, it provides the total get request and how many were hit and missed.

```
$ telnet localhost 11211
stats
STAT cmd_get 13219
STAT get_hits 12232
STAT get_misses 512
```

This means of total 13219 cache requests, it came back with results for 12232, resulting in 12232/13210=92.5% of cache hit, which isn't that bad.

Now once you have a general idea of your cache hit rate, you can improve it even further by logging which particular requests were missed and optimizing them over time.

You can get the memory stats by using command "stats slabs" or you can invalidate items in cache using "flush all".

Conclusion:

You should never rely on your cache only though. If you somehow lost connectivity to your caching server, the application should perform exactly the same. You should use caching only for scalability and/or speed. Implementing the cache itself is pretty simple. The difficult part is which data to cache, how long to cache, when to invalidate the cache, when to update stale data, and how to prevent the database being flooded once the cache is invalidated. This is something that depends on the nature of your data, how fresh you want it and how you update it. You should keep on measuring the stats and gradually improve the effectiveness over time.

Category: Java, Memcached, performance, Scalability
63 Comments

63 Comments on *Using Memcached with Java*

Respond | Trackback

Chris
August 20, 2009 at 3:49 am

Finished this one. =)

Reply

ashish
August 22, 2009 at 3:03 am

Nice article. Definitely, I will lookup memcache.

Any idea, if it plays well with hibernate api?

Reply

admin
August 22, 2009 at 4:31 am

Sure, you can use Memcached as a second-level cache in Hibernate.

Reply

Federico Grilli
August 22, 2009 at 4:21 pm

Very nice introduction to memcached for Java programmers. Thank you.

Reply

Twitter Trackbacks for Using Memcached with Java - Sudarshan Acharya [sacharya.com] on Topsy.com
August 24, 2009 at 10:50 am

[...] Using Memcached with Java – Sudarshan Acharya sacharya.com/using-memcached-with-java – view page – cached Sudarshan Acharya – Open-sourcing my empty brain! — From the page [...]

Reply

Ivan
August 24, 2009 at 7:12 pm

Simple, practical and informative. Great post!

Reply

Robert
September 4, 2009 at 8:13 am

Thx – very useful.

I deployed the solution within tomcat which appeared to work fine. I did run into one issue though: after a few times deploying my app, tomcat it tells me:

java.net.SocketException: Too many open files

running lsof show that there are a lot open connections to the memcache deamon.

Do you have any idea what the best way is to overcome this issue? (apart from restarting tomcat)

regards,

Robert

Reply

Robert
September 4, 2009 at 9:16 am

ah – never mind:

I fixed it by implementing a ServletContextListener that will calll a shutdown on all of the clients.

cheers,

Robert

Reply

Using Memcached with C++ @ Padraig's Blog
September 19, 2009 at 8:13 pm

[...] In this post, I'll show some simple sample usage of the libmemcached C++ interface based on this article about using memcached with Java. Please note that not all this functionality is in the latest [...]

Reply

Krishna
September 28, 2009 at 6:47 pm

Thanks. Very informative Article.

Reply

Roger Evans
January 13, 2010 at 9:20 am

Good post. Thanks for the example code. But suppose you don't want the data marshalled by java – because it needs to be shared with other apps? It's not clear from the spycachemem javadocs how to do that. I keep getting nulls returned on keys I know I've put in with a python app.

Reply

Vishwanath L
March 9, 2010 at 8:07 am

Helpful post. Thanks for posting.

Reply

Alok
March 12, 2010 at 4:13 pm

How do the caches across the distributed servers stay consistent with each other? It's not clear if there is cache synchronization across servers.

"However, the memcached server themselves don't know anything about each others. Its all the function of the client."

So is the client supposed to take care of synchronizing data across servers? That concerns me.

Reply

admin
March 12, 2010 at 4:40 pm

Alok,
Suppose, if you have 100 different objects and 5 Memcached servers, the data will be distributed between 5 servers, ie. each server will store like 20 different objects. The idea is to store one object on any of the server and give it back to you when you need it. The servers are just independent nodes that dont know anything about each other.

If one of the server goes down, only the 20 objects stored on that server are lost. You will have the 80% of data safe on other servers. What happens to the lost data? You will have to fetch the data from database and store it in the memcached cluster again.

Unless your whole cluster is down, you arent losing all of your data. Memcached is built to fail and hence its so powerful. There is no synchronization or anything like that, although you could write a client that to achieve that. But why would you even use Memcached in that case?

Reply

Srinivas P
April 28, 2010 at 5:10 am

Does it has any mechanism to detect given Client IPs are reachable or not, through API. Currently it seems to be there is no mechanism exist if the MemCached Server is down.

Reply

memcached with java – web technologist
May 5, 2010 at 6:26 pm

[...] found a great post regarding the subject. It's a very good [...]

Reply

Aditya
May 13, 2010 at 8:42 am

MyCache.getInstance().get("AllProducts");
Here, get () is always relying on the getCache () of MyCache class. Here, getCache () is generating a random number and picking out the random instance from the MemCachedClient array. In that case, while storing some data, if getCache () returns '2', for example, we can't assure the same instance would be picked for searching the data when get () is invoked.
Since all the MemCached instances are totally unaware of other client instances, how can we ever fetch the actual data from the client bundle ?
Given this condition, our code would always try to hit the DB to feel the user consistent enough about the application responsiveness.
Am i missing something ?

Reply

admin
May 13, 2010 at 3:46 pm

@Aditya,
The random numbers are just connections. Sort of a poor man's connection pool. Even if you use the 1st or the 2nd connection, you are still talking to the same cache cluster.

Try removing the arrays and just use a single connection if you are confused. I just did that so I have a finite and predictable no of connections.

Makes sense?

Reply

Patrick
May 17, 2010 at 4:52 am

thanks, very informative

Reply

Amar Bhide
May 21, 2010 at 7:42 pm

Nice Article, really very informative.

Reply

Sumved Shami

June 18, 2010 at 10:05 am

Hi Sudarshan

Since, spy memcached client supports asynchronous methods, I have been trying to implement it in my application.

I am just trying to implement it as you have suggested. But, I am getting some problems. My be you can help me out to solve it.

There are 2 ways you can actually initialize your client object
1. MemcachedClient client = new MemcachedClient(AddrUtil.getAddresses(memcacheServersList));

2. MemcachedClient client = new MemcachedClient(new BinaryConnectionFactory(), AddrUtil.getAddresses(memcacheServersList));

If I use 1st one then, I am able to connect to the client. But, in the 2nd way I am getting this exception:

"net.spy.memcached.internal.CheckedOperationTimeoutException: Timed out waiting for operation – failing node: /127.0.0.1:11211"

I am using 1.2.4 version of Memcache server & 2.5 version of spy Memcache client.

Thanks in advance.

Regards,
Sumved Shami

Reply

Praveen
August 5, 2010 at 2:11 am

Nice article!

Reply

joseph
August 15, 2010 at 5:59 pm

Few questions:
1. Does spymemcached support connection pool and auto reconnect? If so, any published example of using spymemcached connection pool?
2. Is default using consistent hashing?

Reply

anand sarwade
September 3, 2010 at 7:24 am

I have been looking for many articles to understand and setup memcached to work with java client. thanks for this article with nice and informative steps.

Reply

dennis
October 18, 2010 at 7:04 am

@joshph

1.Spy doesn't support connection pool,but another client xmemcached has support it,please see

http://code.google.com/p/xmemcached/

2.Default is not consistent hashing,but was standard mod hash.

Reply

SomeotherRandomGuy
October 19, 2010 at 2:34 pm

A quick and a great tutorial…I too was looking up to use Memcached with Java. Very nicely explained. Memcahched itself doesn't have much documentation, so this helped me to undertand thoroughly.

Reply

ehcache.net
December 27, 2010 at 9:08 am

Using Memcached with Java …

Why not JBoss Cache?
By default, if you are looking for a caching solution for your Java based enterprise application, the tendency is to go with Java Caches. I have been using JBoss Cache for a couple of years now. It is a very powerful smart cache, w…

Reply

Akshathkumar Shetty
January 6, 2011 at 2:27 pm

thanks dude.. good article..

Reply

srinivasan
January 27, 2011 at 8:45 am

Is memcached service available for Windows? Please let me know.

Reply

Akshathkumar Shetty
February 12, 2011 at 8:56 am

On Windows.. You could compile the original version or use alternate server implementation like the following QuickCached

http://code.google.com/p/quickcached/

Reply

Memcached and Java « The Yogesh Notes
March 5, 2011 at 2:45 pm

[...] good link on Memcached: 1) using memcached with java by Sacharya 2) A good pdf GA_googleAddAttr("AdOpt", "1"); GA_googleAddAttr("Origin", "other"); [...]

Reply

Mouhamed
March 15, 2011 at 1:25 pm

I tried to use memcached with my J2EE application (glassfish server).
When I have done this instruction:
MyCache.set(key, ttl, response),

I got this error during the execution:
java.lang.IllegalArgumentException: Non-serializable object
at net.spy.memcached.transcoders.BaseSerializingTranscoder.serialize(BaseSerializingTranscoder.java:86)
at net.spy.memcached.transcoders.SerializingTranscoder.encode(SerializingTranscoder.java:135)
at net.spy.memcached.MemcachedClient.asyncStore(MemcachedClient.java:291)
at net.spy.memcached.MemcachedClient.set(MemcachedClient.java:648)
at cache.MyCache.set(MyCache.java:74)

Reply

Sreekanth Kalapur
March 15, 2011 at 1:38 pm

Simple yet pretty useful material on implementing Memcached with Java, though I would like to add something to it.

After memcached is installed, I ran into this error –
"error while loading shared libraries: libevent-2.0.10.so.2: cannot open shared object file: No such file or directory"

It turned out that when libevent got installed, the library was not loaded and I need to do it manually using /sbin/ldconfig

Create a file under /etc/ld.so.conf.d/
vi /etc/ld.so.conf.d/libevent-i386.conf

and add the following to the file
/usr/local/lib/

and run the command
ldconfig

Bingo! It works now…

Reply

Ben
March 31, 2011 at 6:33 pm

Mouhamed,

you have to serialize the object you are storing in memcache.

"public class myClass implements java.io.Seralizable"

Reply

Greg
April 26, 2011 at 1:41 pm

I am also getting a timeout exception when I used the BinaryConnectionFactory(). Same as Sumved above. I am and using the 2.6 memcache client and the memcache server 1.2.6 for Windows. Is this a known issue?

Reply

Dimple
May 31, 2011 at 8:43 am

I need to know how to intall spymemcached.

Reply

Nivu
June 11, 2011 at 4:11 pm

Excellent article!!!! kudos.

Reply

Nalin Bakshi
June 20, 2011 at 5:24 am

Great article!
Helpful and informative.

Reply

Balaji Nandarapu
August 17, 2011 at 12:45 pm

Nice article, It is helpful.

Reply

Error 404 - Not Found - Sudarshan Acharya
August 26, 2011 at 11:05 am

[...] Using Memcached with Java [...]

Reply

http://links.soudev.com.br
August 29, 2011 at 1:30 pm

Using Memcached with Java – Sudarshan Acharya…

Tutorial de uso do Memcache com Java…

Reply

sharma
September 14, 2011 at 1:45 pm

Thanks for the helpful article

Reply

Manish
September 16, 2011 at 3:46 pm

Thanks for this article…it was really helpful.
But, I was wondering whether there is some way of doing multi get/set in java ?

thanks

Reply

anti-neutrino
November 29, 2011 at 6:09 am

Great!
easy to follow

Reply

Nick Weaver
December 17, 2011 at 2:03 pm

Nice article! Thanks alot.
If I may suggest to add the revision number/hash to the memchaced keys prefix to set and get data. Of course this assumes the whole system is under version control. This small addition prevents restarting the memached server due to new data formats stored with the same key. This is especially worthwhile if there are dedicated memcached servers which are shared among many applications.
Nick

Reply

Rupak
December 20, 2011 at 11:46 am

Really helpful article. Thanks

Reply

GoalLin
January 4, 2012 at 3:16 am

Nice article

Reply

Batbond007
February 1, 2012 at 2:02 pm

Hi,
I am able to start the memcached on macOx and also able to store and retrieve the key/values in the cache using Java. My Question to you is how do you store the mysql query data in the cache and then retrieve it. For e.g. let's say I have a query that takes a long time to execute so do you store the result of the query into the memcached with the queryId as the key, and when the app tries to execute the same query again you look into the cache first and if the data is there you retrive the values from there ?

Reply

kirankumar
June 5, 2014 at 12:57 pm

Hi can U please Send me the sample code in java,

Reply

bas
February 8, 2012 at 9:46 am

Really cool – thanks!

Reply

Anoop Halgeri
February 20, 2012 at 11:39 am

Cool.. Its neat.. Wonder if Spring provides some implementation..

Reply

vin
March 29, 2012 at 8:31 pm

Hi,

Will you be able to provide information, on how does memcached handle the replication in clustered environment?

Thanks in Advance,
VIn

Reply

admin
March 29, 2012 at 9:32 pm

@vin,
A typical memcached deployment doesn't provide replication. So if you lose one of the servers, you lose all the key/values on that server.
If you are interested in replication with memcached, take a look at http://repcached.lab.klab.org/. But do you really need that kind of redundancy?

Reply

Ignacio
April 4, 2012 at 12:55 pm

Hi Admin!
I have a very good question for you. I need a something to connecto Java and Memcached. I am using one already but I think It is giving me troubles. Would you help with this, please? THANKS!

Reply

Sean
April 12, 2012 at 2:36 am

there is problem in the getCache() method, you do not prevent two threads from get exactly the same cacheClient and operate at the same point of time

Reply

Matt Ingenthron
May 2, 2012 at 12:06 am

Why were you creating a connection pool? We don't normally recommend doing this in the spymemcached client project. The IO is asynchronous, so while there is only one thread per node, it won't be blocked by actual network IO, thus there's little reason to pool.

Reply

CVega
May 26, 2012 at 3:19 pm

Excellent POST. However I would know more about caching and EJB, anyone knows something about?. I'm looking for a way that allows me integrate caching with the EntityManager in JPA. Somebody can help me?. Does exist a mechanism to add a mem cache layer between Java Persistence Api and Database Server?. I need to reduce queries latency performed by Session Beans directly to EntiyManager Instance. Any suggestions?.

Thanks Everybody!

Reply

Ramalinga Reddy
November 16, 2012 at 6:13 am

Hi This article is good.Here i am using this memcahe to store session information.But i am facing some problem like its storing the data permanently even if i specify the expiration time.Can you please help me to get out of this problem.

Reply

shiva
December 18, 2012 at 12:20 pm

Am using spy memcache……Am getting following error while making an operation – get/set on MemcachedClient which is idle for some period of time….how to check if the MemcachedClient is alive????

My code sample…

public void setMemcachedServersList(List memcachedServersList) {

this.memcachedServersList = memcachedServersList;
try {
m = new MemcachedClient?[getMemcachedClientPoolSize()]; for (int i = 0; i < m.length; i ++) {
MemcachedClient? mc = new MemcachedClient?(new BinaryConnectionFactory?(),AddrUtil?.getAddresses(getMemcachedServersStr())); mi? = mc;
}
} catch (Exception e) {
e.printStackTrace();
}
}

public MemcachedClient? getMemcachedClient() {

MemcachedClient? mc= null; try {
int i = (int) (Math.random() m.length); mc = mi?;
} catch(Exception e) {
return mc;
} return mc;
}

public Object get(String key) {

// Try to get a value, for up to getTimeout seconds, and cancel if it doesn't return
Object value = null; Future f = null; try {
f = getMemcachedClient().asyncGet(key);
value = f.get(getLookUpTimeout(),TimeUnit?.SECONDS);
} catch (Exception e) {
e.printStackTrace();
f.cancel(true); //Handle Exception
} return value;
}

Exception during get operation….

2012-12-17 20:03:27.352 INFO net.spy.memcached.MemcachedConnection?: Reconnecting due to exception on {QA sa=mail-memcache02.lax.qa1.untd.com/10.111.30.46:1918, #Rops=1, #Wops=0, #iq=0, topRop=Cmd: 0 Opaque: 69 Key: WRONG_PWD_COUNT_FOR_AN_IP-11.245.28.134, topWop=null, toWrite=0, interested=1} java.io.IOException: An established connection was aborted by the software in your host machine

at sun.nio.ch.SocketDispatcher?.read0(Native Method) at sun.nio.ch.SocketDispatcher?.read(Unknown Source) at sun.nio.ch.IOUtil.readIntoNativeBuffer(Unknown Source) at sun.nio.ch.IOUtil.read(Unknown Source) at sun.nio.ch.SocketChannelImpl?.read(Unknown Source) at net.spy.memcached.MemcachedConnection?.handleReads(MemcachedConnection?.java:478) at net.spy.memcached.MemcachedConnection?.handleIO(MemcachedConnection?.java:418) at net.spy.memcached.MemcachedConnection?.handleIO(MemcachedConnection?.java:275) at net.spy.memcached.MemcachedClient?.run(MemcachedClient?.java:2030)

Reply

Shaik
February 1, 2013 at 12:54 pm

Hi,

We are facing a problem with memcache. We are using hibernate-memcached-version1.5-SNAPSHOT and spymemcached-provider-version3.0.2.

The following are the configuration
persistence.xml

!– Enable 2nd Level Cache –

property name="hibernate.cache.use_second_level_cache" value="true"

property name="hibernate.cache.use_query_cache" value="true"

!– MemCache Configuration –

property name=”hibernate.cache.region.factory_class” value=”com.googlecode.hibernate.memcached.MemcachedRegionFactory”

property name=”hibernate.cache.use_minimal_puts” value=”true”

property name=”hibernate.cache.use_structured_entries” value=”true”

property name=”hibernate.memcached.servers” value=”${hibernate.memcached.server.host.port}”

property name=”hibernate.memcached.cacheTimeSeconds” value=”1800″

dto

@Cacheable(true)
@org.hibernate.annotations.Cache(usage = CacheConcurrencyStrategy.READ_WRITE)

In GenericDao we are setting query hints cacheable to true for loadAll().

We are using loadAll() method to fetch all the records. Whenever we made a request the loadAll query is executing and also the queries based on id are executing.

when i refer to the log am able to notice that the data is fetching from database and setting in memcache for a request and when we make another request instead of fetching the data from the memcache it is hitting again to the db and again setting to memcache. Am unable to understand that without modifying any data why it is hitting to db?

Please let me know we are missing anything.

Reply

kirankumar
June 5, 2014 at 12:33 pm

hi ,
i have Installed memecahe In My Ubuntu system by following link
https://www.digitalocean.com/community/articles/how-to-install-and-use-memcache-on-ubuntu-12-04

now am folowing ur article to implement the memcahe toi My Project i understood the theory but am unable to implent it,pls suggest me , where how achive mem cahe to my project and my project is in servlets jsp.

thanks in advance

Reply

kirankumar
June 6, 2014 at 8:04 am

Caused by: java.lang.NullPointerException
at com.komitla.controller.MyCache.get(MyCache.java:34)
at com.komitla.model.DataAccessLayer.getDialyLogsList(DataAccessLayer.java:1795)
at com.komitla.businessObjects.TripSheet.dailyLogList(TripSheet.java:91)

Reply

Binh Thanh Nguyen
June 11, 2015 at 10:43 am

Thanks, nice post

Reply

Respond

« Grails, DBCP & Stale Connections

Gant is Awesome »