Best practices for using ConcurrentHashMap

MAY 27, 2013 LOKESH 22 COMMENTS

The ConcurrentHashMap is very similar to the HashMap class, except that ConcurrentHashMap offers internally maintained concurrency. It means you do not need to have synchronized blocks when accessing ConcurrentHashMap in multithreaded application.

```
1  //Initialize ConcurrentHashMap instance
2  ConcurrentHashMap<String, Integer> m = new ConcurrentHashMap<String, Integer>()
3
4  //Print all values stored in ConcurrentHashMap instance
5  for each (Entry<String, Integer> e : m.entrySet())
6  {
```

```
7   system.out.println(e.getKey()+"="+e.getValue());
8   }
```

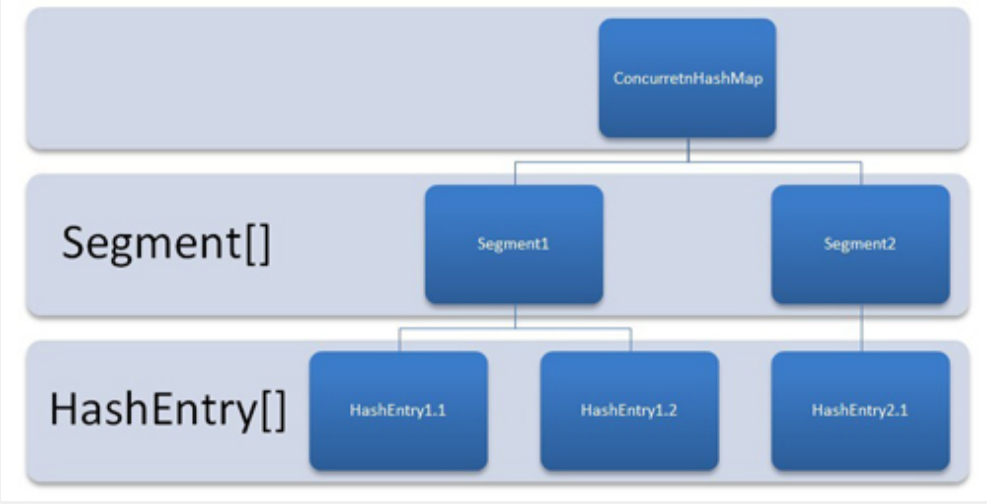
Above code is reasonably valid in multi-threaded environment in your application. The reason, I am saying “reasonably valid” is that, above code yet provides thread safety, still it can decrease the performance of application. And ConcurrentHashMap was introduced to improve the performance while ensuring thread safety, right??

So, what is that we are missing here??

To understand that we need to understand the internal working of ConcurrentHashMap class. And the best way to start is look at the constructor arguments. Fully parametrized constructor of ConcurrentHashMap takes 3 parameters, initialCapacity, loadFactor and concurrencyLevel.

1) initialCapacity
2) loadFactor
3) concurrencyLevel

First two are fairly simple as their name implies but last one is tricky part. This denotes the number of shards. It is used to divide the ConcurrentHashMap internally into this number of partitions and equal number of threads are created to maintain thread safety maintained at shard level.



ConcurrentHashMap

The default value of “concurrencyLevel” is 16. It means 16 **shards** whenever we create an instance of ConcurrentHashMap using default constructor, before even adding first key-value pair. It also means the creation of instances for various inner classes like ConcurrentHashMap$Segment, ConcurrentHashMap$HashEntry[] and ReentrantLock$NonfairSync.

In most cases in normal application, a single shard is able to handle multiple threads with reasonable count of key-value pairs. And performance will be also optimal. Having multiple shards just makes the things complex internally and introduces a lot of un-necessary objects for garbage collection, and all this for no performance improvement.

The extra objects created per concurrent hashmap using default constructor are normally in ratio of 1 to 50 i.e. for 100 such instance of ConcurrentHashMap, there will be 5000 extra objects created.

Based on above, I will suggest to **use the constructor parameters wisely to reduce the number of unnecessary objects and improving the performance**.

A good approach can be having initialization like this:

```
1  ConcurrentHashMap<String, Integer> instance = new ConcurrentHashMap<String, Int
```

An initial capacity of 16 ensures a reasonably good number of elements before resizing happens. Load factor of 0.9 ensures a dense packaging inside ConcurrentHashMap which will optimize memory use. And concurrencyLevel set to 1 will ensure that only one shard is created and maintained.

Please note that if you are working on very high concurrent application with very high frequency of updates in ConcurrentHashMap, you should consider increasing the concurrencyLevel more than 1, but again it should be a well calculated number to get the best results.

Happy Learning !!

Popular HashMap and ConcurrentHashMap interview questions

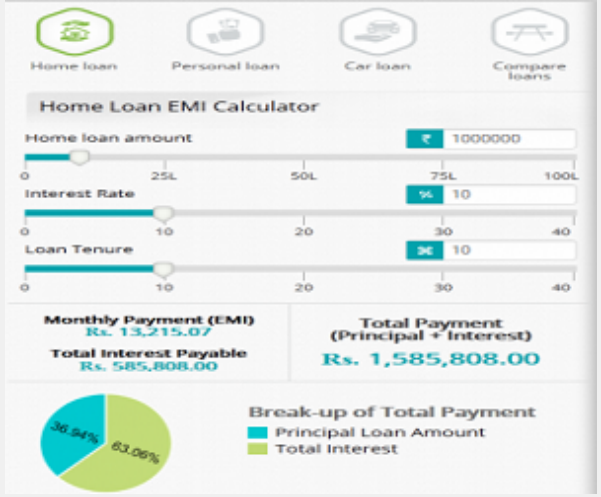
Java executor framework tutorial and best practices

Object initialization best practices : Internal caching in wrapper classes

13 best practices for writing spring configuration files

Best practices to improve JDBC performance

What is Thread Safety?



Related Posts:

1. Popular HashMap and ConcurrentHashMap interview questions
2. Java executor framework tutorial and best practices
3. Object initialization best practices : Internal caching in wrapper classes
4. 13 best practices for writing spring configuration files
5. Best practices to improve JDBC performance
6. What is Thread Safety?

BEST PRACTICES | CONCURRENTHASHMAP | CORE JAVA | INTERVIEW QUESTION | THREAD SAFETY

PREVIOUS POST

[Solved] java.lang.IncompatibleClassChangeError: Implementing class

NEXT POST

Parse CSV files in java

22 THOUGHTS ON “BEST PRACTICES FOR USING CONCURRENTHASHMAP”

Mayank

DECEMBER 4, 2014 AT 4:08 PM

Hi,

Thanks for this amazing blog, concepts are very well explained. When you get a chance could you please answer my question –

1. Say for example I have multiple threads (count = 8) that would add or update my Hashmap. What would be concurrencyLevel you would recommend?
2.

```
ConcurrentHashMap<String, Integer> instance = new ConcurrentHashMap<St
        //OR
Collections.synchronizedMap( HashMap )
```

Won't they have almost same performance since ConcurrentHashMap has only 1 Segment and all Threads will have to wait while updating HashEntry?

REPLY

turkerfatih27

APRIL 24, 2014 AT 10:00 AM

Article contains important details which are rarely explained on this subject. Thanks Lokesh really helped.

REPLY

Binh Thanh Nguyen

APRIL 16, 2014 AT 6:24 AM

Thanks, nice post

REPLY

Vishal

FEBRUARY 6, 2014 AT 7:13 AM

what is meant by dense packaging inside ConcurrentHashMap ?

REPLY

★ Lokesh

FEBRUARY 6, 2014 AT 7:48 AM

Load factor is the ratio between the number of “buckets” in the map and the number of expected elements. A value of 0.75 will suggest that if the buckets are more than 75% full, the Map should be resized. Similarly, 0.90 will suggest that if the buckets are more than 90% full, the Map should be resized.

REPLY

Guru

AUGUST 7, 2014 AT 3:47 AM

This clarified my doubt, Thanks Lokesh,Vishal for asking it.

REPLY

Kumar Sushant

JANUARY 27, 2014 AT 6:33 PM

Nice article. But I would suggest one small correction

“First two are fairly simple as their name implies but last one is tricky part. This denotes the number of shards. ”

The parameter concurrencyLevel isn’t always equal to the number of segments created. It will only be true if it is a power of 2.

REPLY

★ Lokesh

JANUARY 28, 2014 AT 5:26 AM

True. This parameter should be in power of two.

REPLY

H Singh

NOVEMBER 7, 2013 AT 7:37 AM

What is the bucket in HashMap ?

is it the Entry Array ? I am looking for collision condition solution when we have same hash code for 2 objcets.

In case of collision , objects are stored in same bucket and retrieved using equals function to get exact object.

REPLY

★ Lokesh Gupta

NOVEMBER 7, 2013 AT 8:13 AM

Yes. Bucket is single index location in Entry array. If entry array is of size 10, the there are 10 buckets. Regarding conflict, I have written about **internal working of HashMap**. Please refer to linked post.

REPLY

H Singh

NOVEMBER 6, 2013 AT 10:46 PM

Hi,

I have doubt over the concurrencyLevel parameter.

Suppose I have initialized my ConcurrentHashMap with concurrencyLevel of 2.

I have 2 threads trying to update the first shard elements @ same time.

How this situation will be handled?

Do we need to apply extra care and synchronization for such cases ? Or it is handled internally by ConcurrentHashMap itself ?

REPLY

★ Lokesh Gupta

NOVEMBER 7, 2013 AT 12:15 AM

No need to add extra synchronization. ConcurrentHashMap uses locks in put method.

http://grepcode.com/file/repository.grepcode.com/java/root/jdk/openjdk/6-b14/java/util/concurrent/ConcurrentHashMap.java#ConcurrentHashMap.Segment.put%28java.lang.Object%2Cint%2Cjava.lang.Object%2Cboolean%29

REPLY

Souvik

OCTOBER 23, 2013 AT 6:07 PM

Hi,
No doubt it's a wonderfull post. But I have two questions.

1) Is Shards and Segments are same?

2) What is the use of Segment

REPLY

★ Lokesh Gupta

OCTOBER 23, 2013 AT 10:11 PM

1) Yes, both are same.
2) Segments are to create different "groups of key-value pairs" so that different groups can be used used concurrently by different threads.

REPLY

Souvik

OCTOBER 23, 2013 AT 10:21 PM

Is that means that each segments contains same key-value pair? Secondly if I create 20 shards then it means that 20 threads can do concurrent transaction on my map. Correct me if I am wrong.

Is that possible for you to explain that if I enter a key value pair in the concurrent hash map how it's getting updated in multiple segments.

REPLY

★ Lokesh Gupta

OCTOBER 23, 2013 AT 11:23 PM

NO. A key-value pair inserted in ConcurrentHashMap will be stored in one-and-only-one segment. Let say you store 200 key-value pairs in ConcurrentHashMap having 20 segments, and if they are distributed equally the each segment will have 10 key-value pairs. Now these 20 shards are available to be concurrently accessed by 20 different threads.

If you try to lookup for value for any key, then first suitable segment is located and then suitable entry is located inside segment.

REPLY

Souvik

OCTOBER 24, 2013 AT 12:17 AM

As I understand from the above one that each shard will serve one thread. If I create 100 shard then concurrently 100 threads can access the values. Now, suppose I have a Key =”A” and it's stored in the Segment 1. Now if two threads are concurrently wants to access the values

with the Key “A” then will the second thread will wait till the first thread leaves the lock of the Shard?

Do you know the logic to choose the Shard to store the key value?

REPLY

Souvik

OCTOBER 25, 2013 AT 3:13 PM

Suppose a value with the Key “A” inserted in to #5 segment. Now two threads wants to read the value with the key “A”. Then what would happens will the second thread will wait for the first thread to complete it’s task.

Secondly, Could you explain that how Concurrent Hashmap choose segment to store the data or to retrieve.

REPLY

★ Lokesh Gupta

OCTOBER 25, 2013 AT 11:50 PM

To me, its some complex mathematics for locating the segment.

```
int hash = hash(key.hashCode());
int j = (hash >>> segmentShift) & segmentMask;
if ((s = (Segment)UNSAFE.getObject // nonvolatile; recheck
(segments, (j < < SSHIFT) + SBASE)) == null) // in ensureSegment
s = ensureSegment(j);
```

To read value, it used UNSAFE.getObjectVolatile();

Refer:
http://grepcode.com/file/repository.grepcode.com/java/root/jdk/openjdk/6-b14/sun/misc/Unsafe.java#Unsafe.getObjectVolatile%28java.lang.Object%2Clong%29
http://en.wikipedia.org/wiki/Volatile_variable
And
http://howtodoinjava.com/2013/10/19/usage-of-class-sun-misc-unsafe/

REPLY

Priya

OCTOBER 4, 2013 AT 10:50 AM

Nice article..Thanks for that..Suppose I am having a concurrent hashmap with Concurrency level as 2 and i need to read/write values in it by using multiple threads .In the scenario, if one thread has locked up one segment and checking the value and if another thread comes in will it bypass that segment or wait ?
In case of bypass, how come it will check the locked up segment..(because my scenario is to check whole map for a particular count).

REPLY

★ Lokesh Gupta

OCTOBER 4, 2013 AT 10:16 PM

You will have more clear picture after reading this post:
http://howtodoinjava.com/2013/06/14/popular-hashmap-and-concurrenthashmap-interview-questions/

“When getting data, a volatile read is used without any synchronization. If the volatile read results in a miss, then the lock for that segment is obtained and entry is again searched in synchronized block.”

REPLY

aditya kumar

JUNE 19, 2013 AT 11:19 AM

Just awesome …… really happy learning …… thank u very much for this article

REPLY

Note:- In comment box, please put your code inside **[java] ... [/java]** OR **[xml] ... [/xml]** tags otherwise it may not appear as intended.

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Are you Human and NOT robot? Solve this to prove !! *

× 6 = twenty four

Comment

You may use these HTML tags and attributes:

```
<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>
```

POST COMMENT

☐ Notify me of follow-up comments by email.

▣ Notify me of new posts by email.

SUBSCRIBE TO BLOG VIA EMAIL

Join 3,668 other subscribers

Email Address

SUBSCRIBE

REFERENCES

Oracle Java Docs

Spring 3 Reference

Spring 4 References

Jboss RESTEasy JAX-RS

Hibernate Developer Guide

Junit Wiki

Maven FAQs

Dzone Dev Links

JSP Homepage

ANT Homepage

META LINKS

Advertise

Share your knowledge

Privacy policy

Contact Us

About Us

POPULAR POSTS

Spring 3 and hibernate integration tutorial with example

Reading/writing excel files in java : POI tutorial

How hashmap works in java

Singleton design pattern in java

Useful java collection interview questions

Understanding hibernate first level cache with example

How hibernate second level cache works?

Working with hashCode and equals methods in java

How to make a java class immutable

4 ways to schedule tasks in Spring 3 : @Scheduled example