#bytescrolls

Some IdentityHashMap gotchas

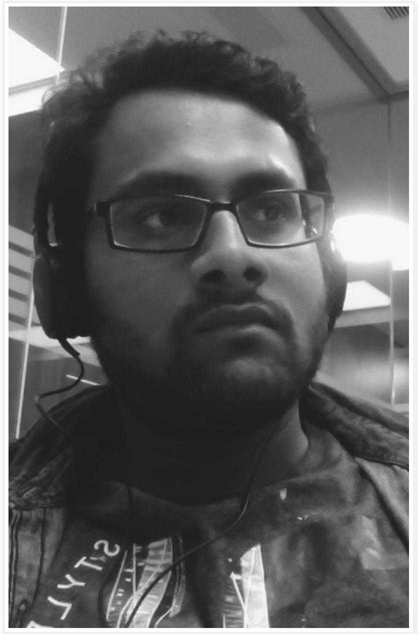
Most of java guys are familiar with IdentityHashMap, which is mainly used when there is a need of maintaining keys based on reference equality. Lot of people, have written about a lot of collections hashing and so on.. But I thought whats so interesting about this map. Even the algorithm behind it is about 45 years old !

Every object is having an "Identity", i.e., the internal "address" is unique for the lifetime of that object. So map will contain unique objects as the keys. Consider HashMap, HashSet, TreeMap, TreeSet, PriorityQueue etc, they are "equality-dependent". One can override the equals() method in the object stored to keep the object as the keys in these collections.But IdentityHashMap is "equality-independent". Even if the objects residing it changes its comparability or equality, the map will stride fine. But it violates the symmetry property of the equals contract. Symmetry means that for any two references a and b, a.equals(b) must return true if and only if b.equals(a) returns true. That sets the contract for IdentityHashMap at odds with the contract for Map, the interface it implements, which specifies that equality should be used for key comparison.

According to the spec in bold says so... "This class is not a general-purpose Map implementation! While this class implements the Map interface, it intentionally violates Map's general contract, which mandates the use of the equals() method when comparing objects. This class is designed for use only in the rare cases wherein reference-equality semantics are required."

IdenityHashMap data structure is based on open addressing, which is a method in which when a collision occurs, alternative locations are tried until an empty cell is found. All the items are stored in an array. This map uses linear probing, wherein the next position is found out sequentially.

@HARISGX



AUTHOR

haris

a programmer

LINKS

ie newlocation = (startingValue + stepSize) % arraySize (here the stepsize be 1)

This is basically storing all keys in a single array. The occupied addresses will form clusters (primary clusters). These clusters are distributed and the keys in them are localized to some are. So the collision will occur when the hash function returns the index of insertion wherein the keys are occupied, consuming time for insertion. Basically this is a form file/memory management.We can say that map behaves like of addressing of arrays. It puts the keys and values alternating in a single array (supposedly good for large data sets) and doesn't need to create or reuse entry objects. Even when the map size be large as it grows the insertions are not affected, while other map implementations(which uses chaining) gets slower. However, lookup is much cheaper than insertion, as we be looking items up much more often than you insert them. Using probing is somewhat faster than following a linked list, when a reference to the value can be placed directly in the array,For other hash-based collections, this is not the case because they store the hash code. This will be efficient when a get operation must check whether it has found the right key by reference (equality is an expensive operation, as it checks for the right hash code). So this takes less memory and its faster.

I read that linear probing was first analyzed by Knuth in a 1963 (at the age of 24 !) memorandum now considered to be the birth of the area of analysis of algorithms. Knuth's analysis, as well as most of the work that has since gone into understanding the properties of linear probing, is based on the assumption that hash function is a truly random function, mapping all keys independently.

The keys are stored in an array Tab. The hash function h will be mapping keys.When a key x is inserted, index = h(x). If Tab[index] is occupied, then we scan sequentially until x is found. If an empty slot found, x is inserted. The next insertion location will be nearest next empty slot from h(x).In IdentityHasMap h(x) is basically System.identityHashCode (as per source some more calculations are done). As this algorithm uses modulo arithmetic it goes around until the size is filled up. So in the implementation MAXIMUM_SIZE is doubled up untill allowed space for object is filled in the permspace.

According to Knuth, it is possible to estimate the average number of probes for a successful search, where l is the load factor as 1/2(1+ 1/(1-l))

The put(k,v) method looks for the position for insertion. As the he index is found out based on calculations involve System.identityHashCode(returning 32-bit integer value), identity hash codes are well distributed, almost like random numbers. Using System.identityHashCode means that for objects that have overridden hashCode and equals, even if two objects are equal() they will be put in a different hash buckets ie it treats them as two distinct objects.

Some popular anomalies:

- Linkedin
- Twitter

SUBSCRIBE

Posts

Comments

Search

TOTAL PAGEVIEWS

TAGS

java hadoop web 2.0 REST programming internet opensocial database distributed systems javascript open source algorithms cloud computing semantic web webservice apache architecture big data codes experiments groovy json ria API adobe air ajax comet data access data integration designpatterns eclipse etl graph hibernate hive indexing information linux lucene map reduce nosql scala security soa spring technical document technology ui web 3.0 widget .net ASP.Net BPM GWT JCR JVM OpenID RCP RPC WCF XML ambient aop avro bigdata bloomfilter

```
IdentityHashMap<long,> orderMap1 = new IdentityHashMap<Long,Order>();
orderMap1.put(1L,null);
orderMap1.put(1L,null);

System.out.println(orderMap1.keySet().size());
```

Prints 1. As the 1L will be autoboxed in Java 1.5. But as the values less than 127 could be pointing to same object there will be a single reference. When use 1000L for the keys, the size will be 2 as it creates distinct objects. Long has a private class for caching these values in consideration to performance as those values are commonly used. This is applicable to Integer,Character etc.

```
private static class LongCache {
private LongCache(){}

static final Long cache[] = new Long[-(-128) + 127 + 1];

static {
for(int i = 0; i < cache.length; i++)
cache[i] = new Long(i - 128);
}
}
```

Take another example

```
IdentityHashMap<String,OrderI> orderMap6 = new IdentityHashMap<String,OrderI>();
OrderI oI2 = new OrderI(1);
OrderI oI3 = new OrderI(2);

orderMap6.put("order1",oI3);
orderMap6.put("order1",oI2);
```

business closures cybernetics e-commerce eii event processing extjs future computing gadgets geo-spatial github google griffon guice hbase ilm information graphics intelligence invisible iphone jBPM jibx jquery networks oAuth objectiveC open solaris orient-db orientdb orm osgi parallel patterns pramati prometheus protocol buffers rdf recommendation ruby silverlight sling social spark standards storage tech ted twitter unicode ux zeppelin

Here the key size will be one. As the string literals are referred from the String heap itself they point to same location.So the second insertion overrides the previous value.

Where can this map be used.Some says , for caches, as they perform very well. Only need to store references though. For cyclic graphs, we must use identity rather than equality to check whether nodes are the same. Calculating equality between two graph node objects requires calculating the equality of their fields, which in turn means computing all their successors and we are back to the original problem. A IdentityHashMap, by contrast, will report a node as being present only if that same node has previously been put into the map. Thus by saving the size and node traversals.

I just wanted to know about this map thats it...

References

http://www.ece.uwaterloo.ca/~ece250/Lectures/Slides/6.07.LinearProbing.ppt
http://www.owasp.org/index.php/Java_gotchas#Immutable_Objects_.2F_Wrapper_Class_Caching
http://www.siam.org/proceedings/soda/2009/SODA09_072_thorupm.pdf
http://www.cs.unm.edu/~moret/graz01.pdf
http://www.it-c.dk/people/pagh/papers/linear-jour.pdf
http://reference.kfupm.edu.sa/content/l/i/linear_probing_and_graphs_122432.pdf

No comments:

Post a Comment

Enter your comment...

Comment as: Amitabh Mandal

Sign out

Publish Preview

Notify me

Links to this post

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

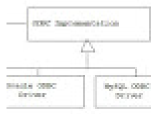
POPULAR POSTS

Using Avro to serialize logs in log4j

I have written about serialization mechanism of Protocol Buffers previously. Similarly, Apache Avro provides a better serialization framew...

Run Datastax Graph in a docker container for windows hosts

#docker #datastax There are some docker images available here . If you don't have docker, download and install docker toolbox from her...

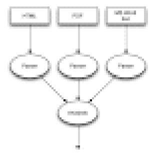


Design Patterns : Bridge

Bridge decouples the abstraction from implementation so that the two can vary independently. Client which uses the classes inheriting the ab...

How Stuff Works: Spring Component Scanning

Spring has an interesting feature of scanning its components defined and load it.So the configuration is tied to application ie the code, us...



[Apache Lucene - Indexing - Part 1]

"Information retrieval (IR) is the science of searching for documents, for information within documents and for metadata about document...

[Experimenting on Space4j , an in memory java database system]

Its always challenging to develop scalable and high performance applications in java.There are different form of data persistence made avail...

[Interesting uses of sun.misc.Unsafe]

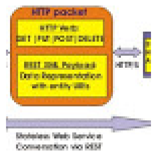
Inspired from the question that found in stackoverflow, I started looking up for the uses. I found some pretty interesting ones... VM &quo...

[Analytics by SQL and Spark using Apache Zeppelin]

#spark #hadoop #analytics #apache #zeppelin #scala I was looking for a cool dashboard based query interface for analytics. I stumble...

[Machine generated data]

At first, the term " machine-generated data " can be confusing. One would think, every data is (or are?) generated from one de...



[What is Computational REST ?]

I have written about REST before .CREST is about computational REST.I think the concept will be a milestone in developing applications on Re...