java - can we do our own memory management?

Ask Question

Is it possible to perform the memory management by yourself. e.g. We allocate a chunk of memory outside the heap space so that it is not subject to GC. And we ourselves take care of allocation/deallocation of objects from this chunk of memory.

Some people have pointed to frameworks like Jmalloc/EHcache. Actually i more want to understand that how they actually do it.

I am fine with some direct approach or even some indirect approach (e.g. first serialize java objects).

java memory-management garbage-collection jvm

edited Jul 11 '16 at 15:30

Peter Lawrey
417k 53 521 878

asked Apr 2 '12 at 14:43

ManojGumber
2,174 16 45

2 Why do you want to do this? Just curious. – Fabian Barney Apr 2 '12 at 14:47

The Unsafe class can possibly help you doing weird stuff... – assylias Apr 2 '12 at 14:47

See also: stackoverflow.com/questions/5574241/… – assylias Apr 2 '12 at 14:48

I have a use-case where I know a lot about the life time of items which I want to store. So I want to try out their Memory management out own my own..outside heap and GCs – ManojGumber Apr 2 '12 at 14:49

See also: stackoverflow.com/questions/9050852/… – Andrew Apr 2 '12 at 18:33

4 Answers

You can not allocate Java objects in a foreign memory location, but you can map memory which is e.g. allocated in a native library into a direct `ByteBuffer` to use it from Java code.

answered Apr 2 '12 at 14:47

Neet
3,277 10 15

I have a library which does this sort of thing. You create excerpts which can be used a rewritable objects or queued events. It keeps track of where objects start and end. The downside is that the library assumes you will cycle all the object once per day or week. i.e. there is no clean up as such. On the plus side its very efficient, can be used in an event driven manner, persisted and can be shared between processes. You can have hundreds of GB of data while using only a few MB of heap.

https://github.com/peter-lawrey/Java-Chronicle

BTW: It supports ByteBuffers or using Unsafe for extra performance.

answered Apr 2 '12 at 15:50

Peter Lawrey
417k 53 521 878

It looks really interesting....I will go through this api – ManojGumber Apr 2 '12 at 16:33

Even if you don't use it, it may give you some ideas. – Peter Lawrey Apr 2 '12 at 19:17

You can use the **off the heap** memory approach Look for example jmalloc

and this is also usefull link Difference between on and off the heap

edited May 23 '17 at 11:58

Community ♦
1 1

answered Apr 2 '12 at 14:55

Julias
2,606 11 46 69

I more want to understand that how frameworks like jmalloc and EHCache for example are doing it. – ManojGumber Apr 2 '12 at 15:01

@MonojGumber: They underlying mechanism ist most likely not written in Java, but in native code. The library then provides a Java interface. – Niklas B. Apr 2 '12 at 15:10

It can be read on the jmalloc project page that they are using serialization. This is a nice idea to store large objects, but not really suitable for performance. – Neet Apr 2 '12 at 15:32

If you mean Java objects, then no, this isn't possible with standard VMs. Although you can always modify the VM if you want to experiment (Jikes RVM for example was made for this very purpose), but bear in mind that the result won't really be Java any more.

As for memory allocation for non-java data structures, that is possible and is being done regularly by native libraries and there is even some Java support for it (mentioned in the other answers), with the general caveat that you can very easily self-destruct with it.

answered Apr 2 '12 at 14:53

biziclop
38.8k 10 59 88

can you please give more details on allocation for non-java data structures. I am ok to convert my Objects to a serialized/byte buffer form. Will some java solution work for that? – ManojGumber Apr 2 '12 at 14:56

@ManojGumber The other two answers contain hints about this: using direct ByteBuffers is one way or you can use a high-level library like jmalloc or BigMemory. But you should really only be using this in production if GC is your main bottleneck. – biziclop Apr 2 '12 at 15:06