An Investigation of the proportion of cold objects

Baoguo Zhou, Gerhard W. Dueck
University of New Brunswick, IBM Canada
Faculty of Computer Science
Email: barry.zhou@unb.ca, gdueck@unb.ca

Definition of cold object

In the Java, objects are stored in the heap. Cold object is an object that is rarely accessed and un-accessed time exceeds a specified time.

Motivation

Cold objects are alive but seldom accessed. When cold objects are moved to cold regions, cold regions do not have to perform the Garbage Collection frequently. Consequently, the pause time of the Garbage Collection will be reduced and the Java application throughput will be increased. In real life Java application, what is the proportion of cold objects? An investigation of the proportion of cold objects has been done.

Methodology

Cold objects are those which remain un-accessed in the heap. Since cold objects are un-accessed, there are not any read/write accesses to cold objects. It is difficult to determine the inactive objects because there are neither cues nor operations on cold objects. However, it is possible to spot the active objects, because there are read/write accesses to the active objects. If these active objects are picked out, the remaining must be the inactive objects.

Solutions

1. The capture of the object activity with Access Barrier

IBM J9 provides an Access Barrier mechanism. Whenever an object is accessed – whatever read/write operation, an Access Barrier can be triggered. There is a good chance to capture the activity of the object at this point.

2. The last accessed time

Whenever an object is accessed and the Access Barrier is triggered, the current access time is recorded with timestamp. If the elapsed time exceeds the preset cold threshold since the last accessed time, this object is supposed to be cold.

3. The storage of the last accessed time

Objects in the heap are always movable. It is difficult to keep track of objects' activities when objects are moved. An efficient way is that timestamp is embedded in the header of object. No matter where the object moves, the timestamp is like a shadow and always accompanies the object.

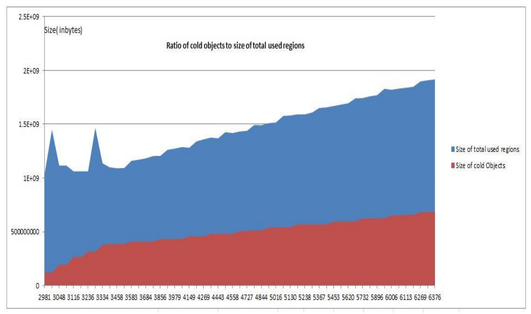
Experimental environment

- ❑ Garbage Collection policy: balanced GC
- ❑ Java heap size: -Xms2048M –Xmx4096M
- ❑ JIT (Just-in-time): turn off
- ❑ Running time: 3 hours
- ❑ Cold threshold: 20 minutes

Experimental results

Seven groups of benchmarks are run and the density of cold objects has been obtained as the following table. SPECjbb2005 collects the size of cold objects of 285MB and the density of cold objects reaches 20.50%.

Benchmark Name	#of total used regions	#of tenured regions	Size of total used regions(MB)	Size of tenured regions(MB)	Size of cold objects(MB)	Density of cold objects
SPECjbb2005	695	440	1390	546	285	20.50%
SPECjvm2008 compiler.compiler	745	80	1490	162	28	1.88%
SPECjvm2008 compiler.sunflow	558	54	1116	108	26	2.33%
SPECjvm2008 derby	1076	935	2152	1769	213	9.90%
SPECjvm2008 sunflow	161	114	322	222	73	22.67%
SPECjvm2008 xml.transform	230	170	460	294	84	18.26%
SPECjvm2008 xml.validation	145	90	290	157	16	5.52%



Conclusion

The results show that the proportion of cold objects can reach more than 20% in some real life applications. That is, 20% objects do not have do garbage collection frequently if cold objects are moved to cold regions. The proportion of cold objects is encouraging.