Are you ready

Saturday, 20 July 2013

ArrayList Using Memory Mapped File

Introduction

In-Memory computing is picking up due to affordable hardware, most of the data is kept in RAM to meet latency and throughput goal, but keeping data in RAM create Garbage Collector overhead especially if you don't pre allocate.
So effectively we need garbage less/free approach to avoid GC hiccups

Garbage free/less data structure

There are couple of option to achieve it

- Object Pool

Object pool pattern is very good solution, i wrote about that in Lock Less Object Pool blog

- Off Heap Objects

JVM has very good support for creating off-heap objects. You can get rid of GC pause if you take this highway and highway has its own risk!

-MemoryMapped File

This is mix of Heap & Off Heap, like best of world.

Memory mapped file will allow to map part of the data in memory and that memory will be managed by OS, so it will create very less memory overhead in JVM process that is mapping file. This can help in managing data in garbage free way and you can have JVM managing large data. Memory Mapped file can be used to develop IPC, i wrote about that in power-of-java-memorymapped-file blog

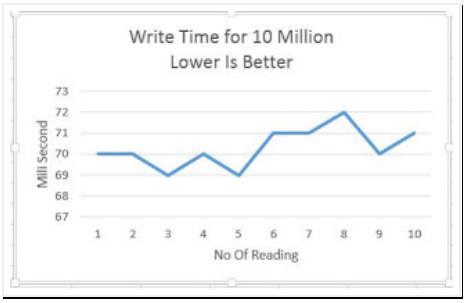
In this blog i will create ArrayList that is backed up by MemoryMapped File, this array list can store millions of object and with almost no GC overhead. It sounds crazy but it is possible.

Lets gets in action

In this test i use Instrument object that has below attribute
- int id
- double price

So each object is of **12** byte.
This new Array List holds **10 Million** Object and i will try to measure writer/read performance

Writer Performance



X Axis - No Of Reading
Y Axis - Time taken to add 10 Million in Ms

Adding 10 Million element is taking around 70 Ms, it is pretty fast.

Writer Throughput

Lets look at another aspect of performance which is throughput

X Axis - No Of Reading

Search

Search

Java Code Geeks

Google+ Followers

About Me

Ashkrit

Pragmatic software developer who loves practice that makes software development fun and likes to develop high performance & low latency system.

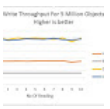
View my complete profile

Blog Archive

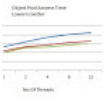
► 2014 (11)
▼ 2013 (13)
► October (1)
► September (3)
▼ July (3)
Which memory is faster Heap or ByteBuffer or Direc...
ArrayList Using Memory Mapped File
How To Write Micro benchmark In Java
► June (2)
► May (1)
► February (1)
► January (2)
► 2012 (6)

Popular Posts

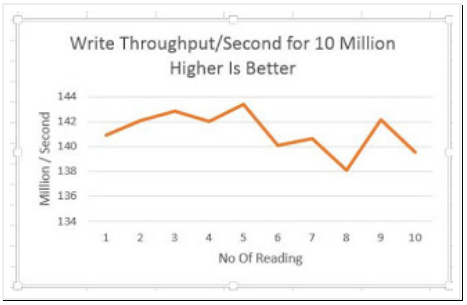
Power of Java MemoryMapped File
Power of Java MemoryMapped File In JDK 1.4 interesting feature of Memory mapped file was added to java, which allow to map any file to O...



Which memory is faster Heap or ByteBuffer or Direct ?
Java is becoming new C/C++ , it is extensively used in developing High Performance System. Good for millions of Java developer like me:-) ...



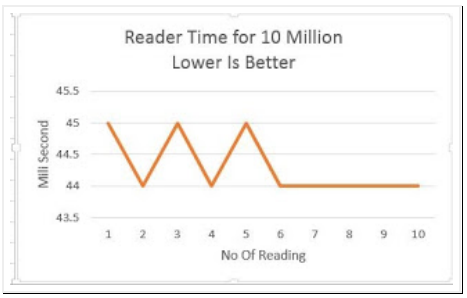
Lock Less Java Object Pool
It is being while i wrote anything, i has been busy with my new job that involve doing some interesting work in performance tuning. One of ...



Y Axis - Throughput /Second , in Millions

Writer throughput is very impressive, i ranges between **138 Million** to **142 Million**

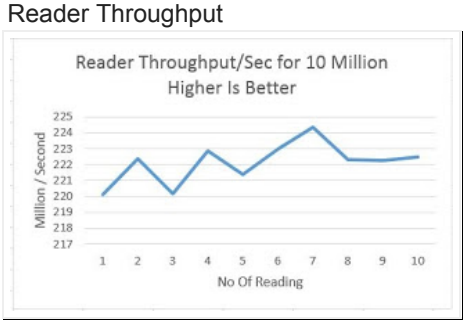
Reader Performance



X Axis - No Of Reading
Y Axis - Time taken to read 10 Million in Ms

It is taking around 44 Ms to read 10 Million entry, very fast. With such type of performance you definitely challenge database.

Reader Throughput



X Axis - No Of Reading
Y Axis - Throughput /Second , in Millions

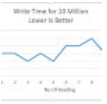
Wow Throughput is great it is **220+** million per second

It looks very promising with **138 Million/Sec** writer throughput & **220 Million/Sec** reader throughput.

Comparison With Array List

Lets compare performance of BigArrayList with ArrayList,

Writer Throughput - BigArrayList Vs ArrayList



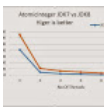
ArrayList Using Memory Mapped File
Introduction In-Memory computing is picking up due to affordable hardware, most of the data is kept in RAM to meet latency and throughput ...

Java Reflection Facts
Java has wonderful feature that allow to inspect any object at run time and extract useful information about it for e.g constructor, metho...

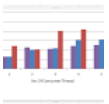
How To Write Micro benchmark In Java
So many article has been written on how to write micro-bench mark in java, this blog is my attempt to explain the topic. What is Micro b...

Executor With ConcurrentLinkedQueue
In this blog i will explore some of waiting strategy for inter thread communication. BlockingQueue is integral part of many concurrency f...

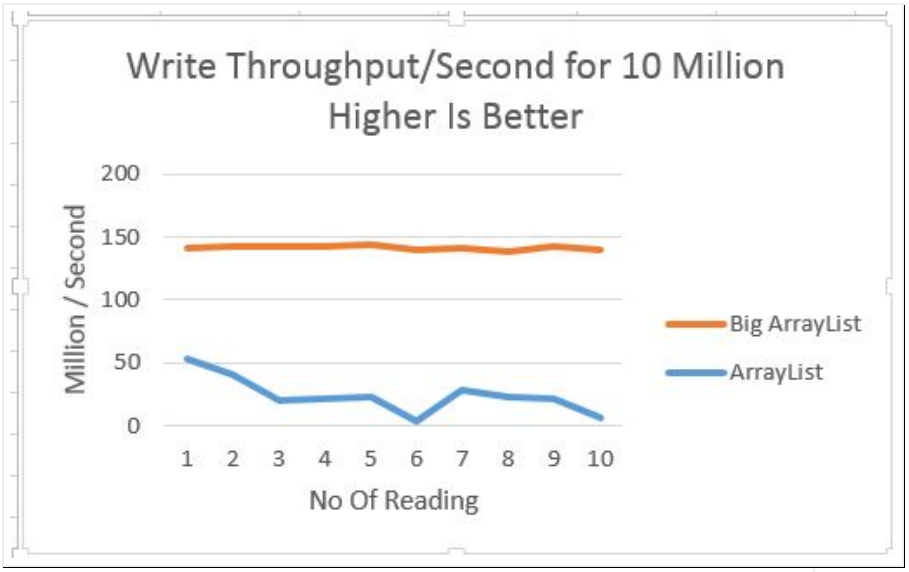
Java Queues - Bad Practices
Writing after long gap, looks like i lost the motivation or ran out of topic :-) Recently while going through code of my current assignme...



AtomicInteger Java 7 vs Java 8
Atomic Integer is interesting class, it is used for building many lock free algorithm. Infact JDK locks are also build using ideas from Ato...



Lock Free bounded queue
Lock free bounded queue JDK 1.5 was first step towards adding serious concurrency support to java, it changed the way you write concurre...

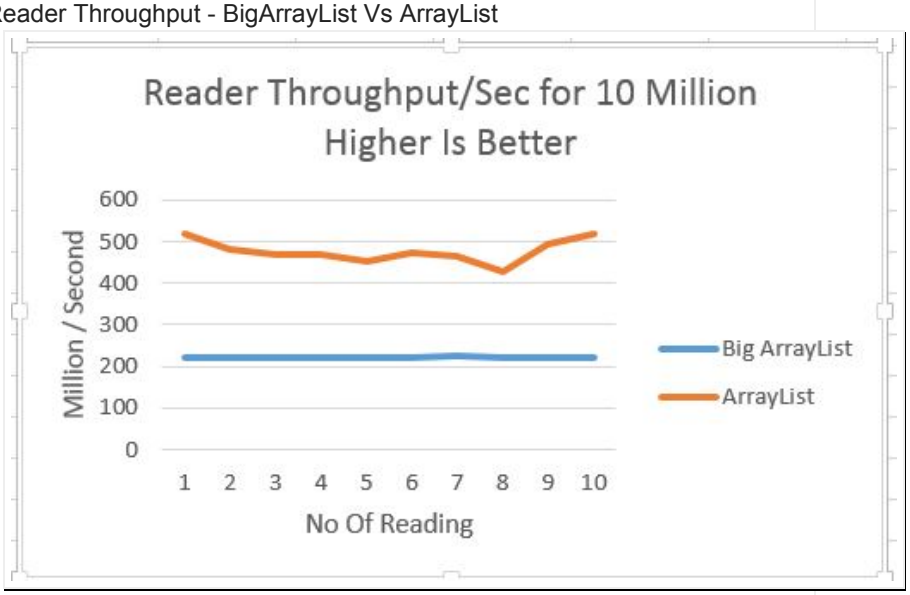


Throughput of BigArrayList is almost constant at around **138 Million/Sec,** ArrayList starts with **50 Million** and drops under **5 million.**

ArrayList has lot of hiccups and it is due to
- Array Allocation
- Array Copy
- Garbage Collection overhead

BigArrayList is winner in this case, it is **7X times** faster than arraylist.

Reader Throughput - BigArrayList Vs ArrayList



ArrayList performs better than BigArrayList, it is around **1X** time faster.

BigArrayList is slower in this case because
- It has to keep mapping file in memory as more data is requested
- There is cost of un-marshaling

Reader Throughput for BigArrayList is 220+ Million/Sec, it is still very fast and only few application want to process message faster than that.
So for most of the use-case this should work.

Reader performance can be improved by using below techniques
- Read message in batch from mapped stream
- Pre-fetch message by using Index, like what CPU does

By doing above changes we can improve performance by few million, but i think for most of the case current performance is pretty good

Conclusion

Memory mapped file is interesting area to do research, it can solve many performance problem. Java is now being used for developing trading application and GC is one question that you have to answer from day one, you need to find a way to keep GC happy and MemoryMapped is one thing that GC will love it.

Code used for this blog is available @ GitHub , i ran test with 2gb memory.
Code does't handle some edge case , but good enough to prove the point that that MemoryMapped file can be winner in many case.

Posted by Ashkrit at 09:30

Labels: BigData, High throughput, In-Memory, Low Latency, MemoryMapped, Performance

7 comments:

Paul 31 July 2013 at 11:16

Interesting....where is the source for BigArrayReaderWriter?

Reply

Replies

Ashkrit 1 August 2013 at 04:26

Thanks.
It is on Github, details at at end of blog.

github url - https://github.com/ashkrit/blog/tree/master/bigarraylist

Reply

Alistair Oldfield 21 September 2013 at 09:42

Very cool, and exactly what I am looking for!

However...
Won't compile. Is missing BigArrayReaderWriter.java in the github dir you reference.
Any chance to upload that please?

Reply

Ashkrit 21 September 2013 at 10:01

Check on github, i have added it
https://github.com/ashkrit/blog/tree/master/bigarraylist

Reply

Harry 26 April 2014 at 03:45

Could you put into a maven repo?

Reply

tell the truth 31 July 2014 at 02:38

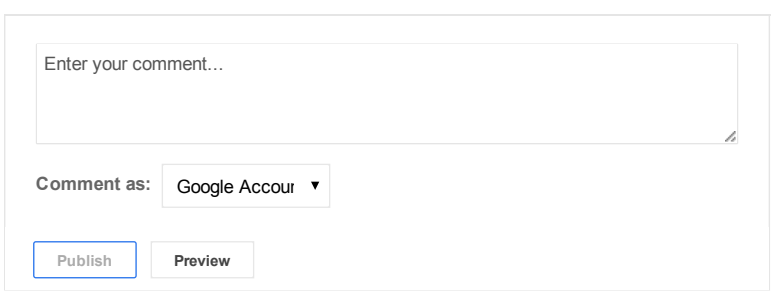
While we add a List, It is fully consists string data.How an we defined the message size,For string it is not writing a whole data.Partially it have be written.Kindly share your thoughts.

Reply

tell the truth 31 July 2014 at 02:41

For a write method,how can we allocate a byte size for string.Please share your idea's.

Reply



Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Share It

Share this on Facebook
Tweet this
View stats

(NEW) Appointment gadget >>