**Using JCACHE to build *standard* Java API bridges to high performance *non-standard* (that’s right, we’re going off-heap!) Cache Implementations.**

***By Ben Cotton***

***INTRODUCTORY PROSE***

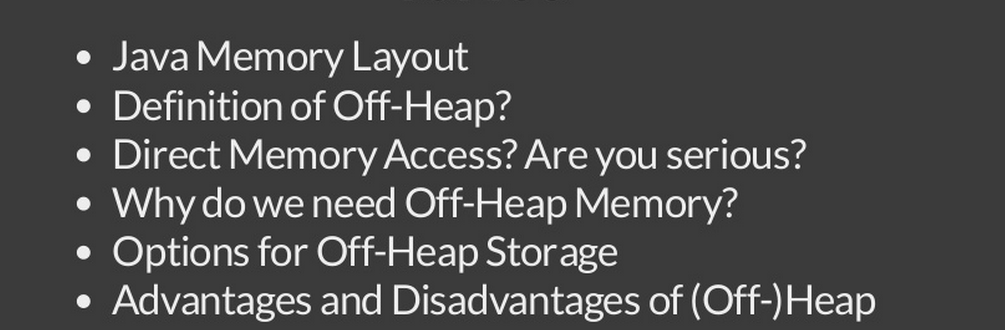
Currently, sitting before the Java Community Process Executive Committee is the Final Draft expert group submission of JSR-107: The Java standard Caching API and SPI. Once approved, Java will (finally!) have an official Caching API standard … JCACHE 1.0. JCACHE is a significant API standard, effectively delivering to the Java Caching community *exactly* what JDBC delivered to the Java RDBMS community.

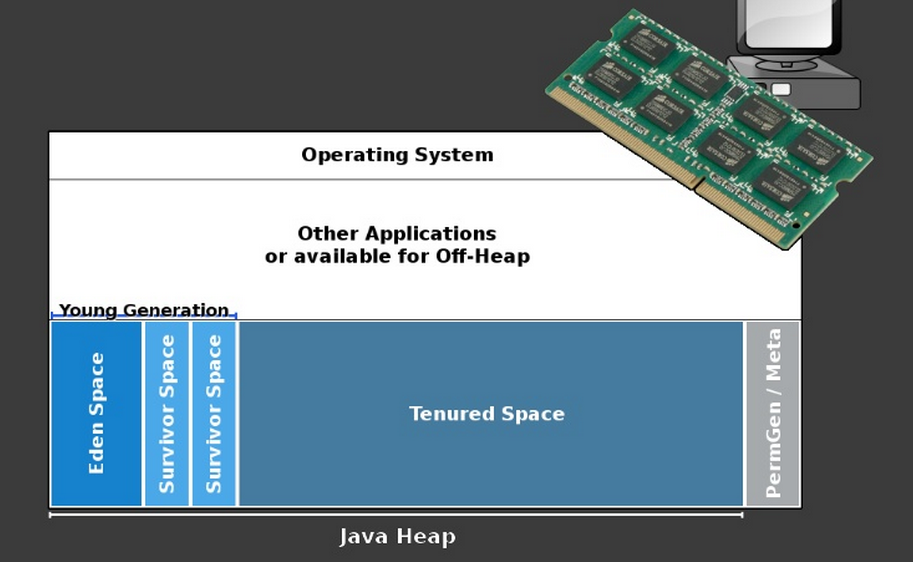
//Ben will blah-blah-blah here about the dramatic “goodness” of how JCACHE will liberate application programmers from CACHING\_VENDOR-specific proprietary API lock-in trauma.

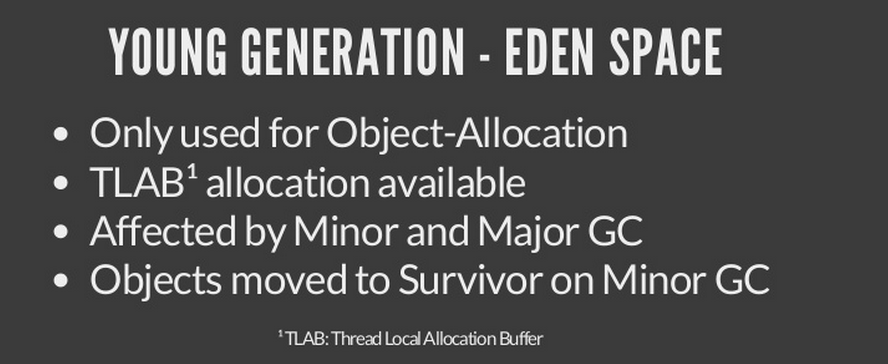
//Ben will blah-blah-blah here of this JCACHE API liberation likely results in this highly ironic consequence = the dramatic “goodness” of a wide number and variety of ultra-high performant (that’s right we’re going off-heap) implementations of javax.cache.Cache<K,V>. In other words, if every one gets to the same place in the same way, it makes sense for Java Caching technology providers to distinguish where they end up. I.e. it makes sense for Java Caching technology providers to go off-heap. It is simply a better place to be.

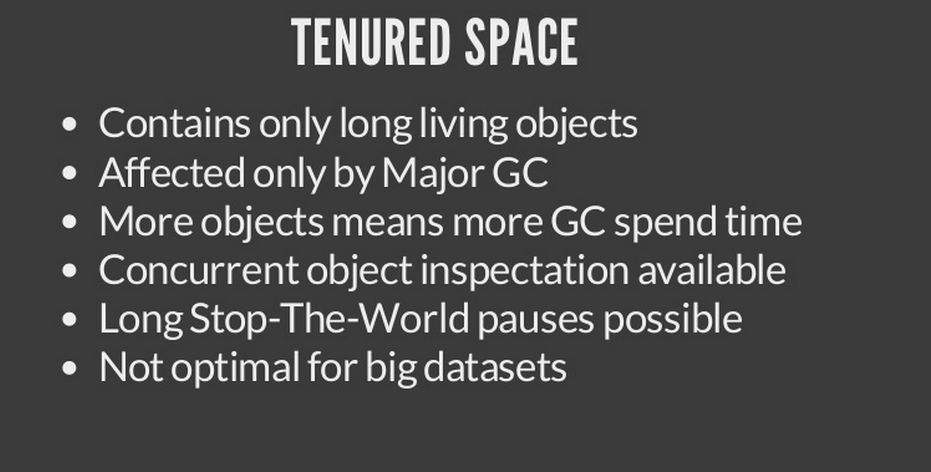
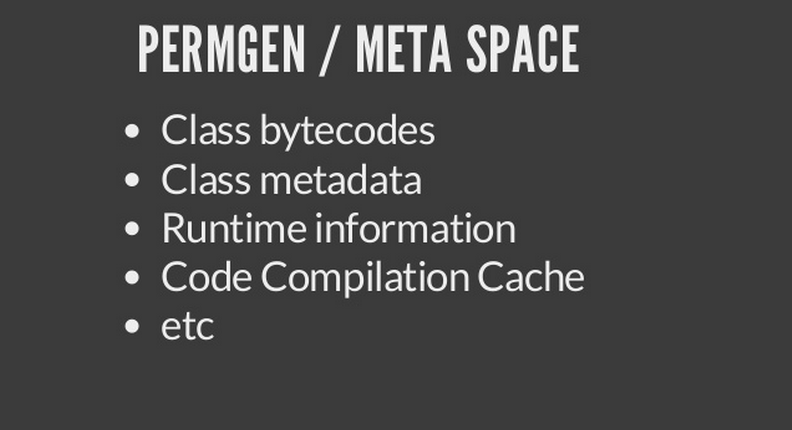
***PART I. The Outline.***

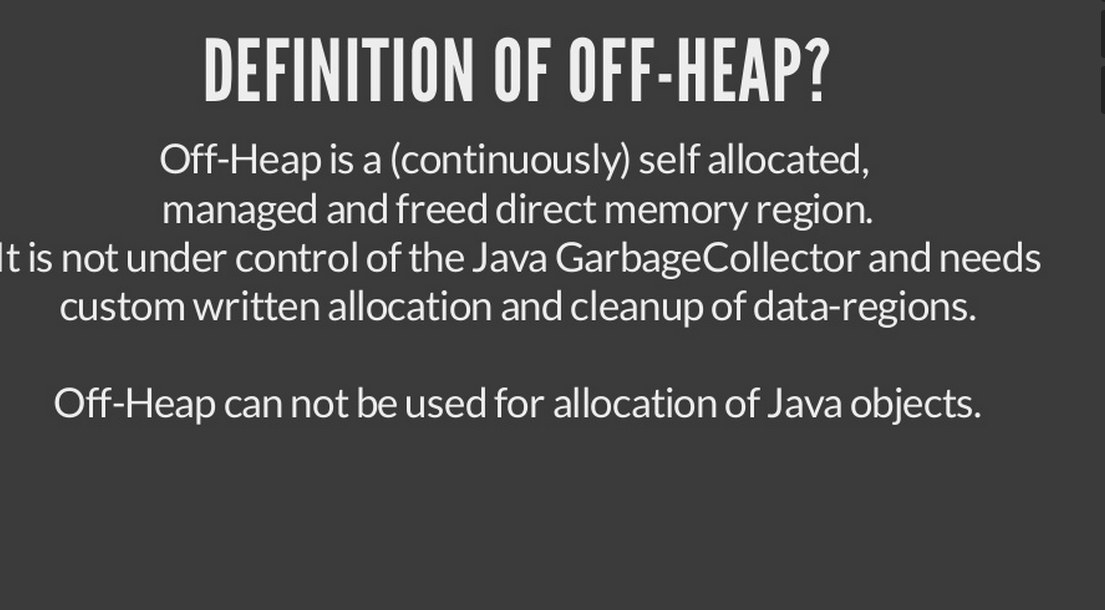
***“You may eat from any tree in the Eden of Paradise. But do not eat the forbidden Fruit from the Tree of Knowledge”.*** God. Genesis 2.17***.***

******

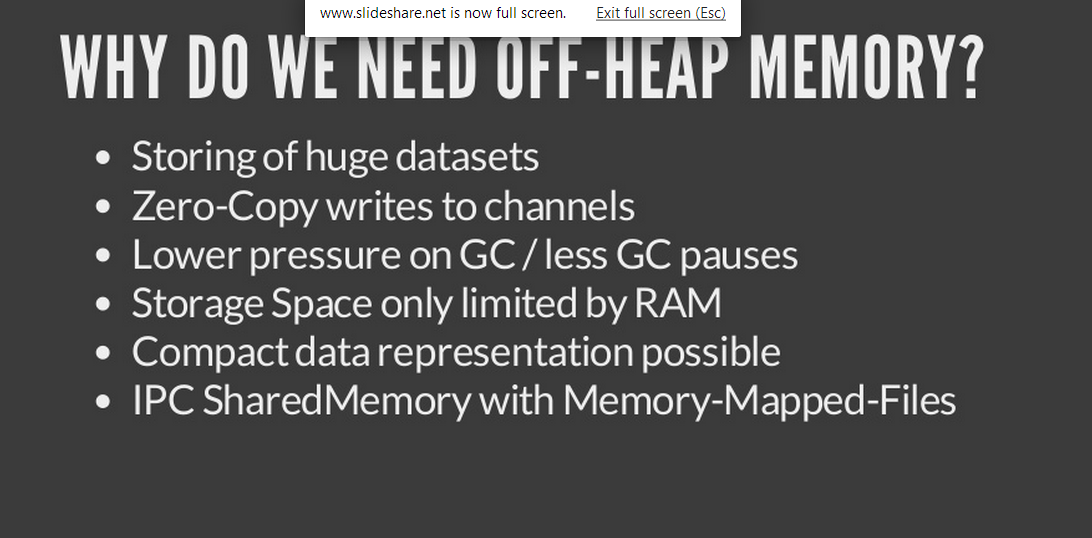
******

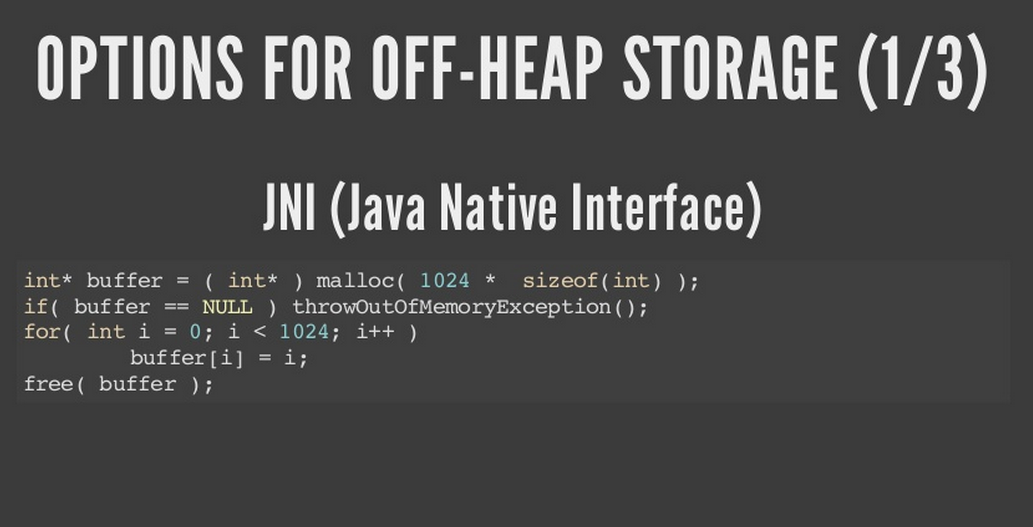


** **

****

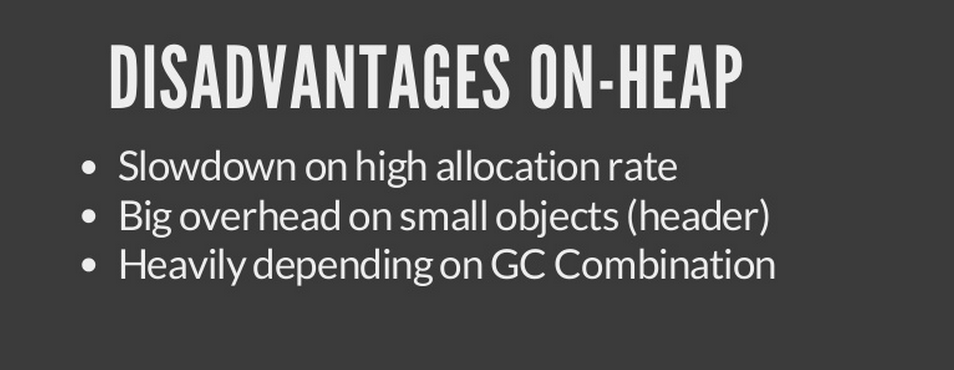
****

****

****

****

****

** **

***PART II. The “Zen”, The “Alphabet”, and the “Shakespeare” of going Off-Heap***

**“*If your data sets are small, and performance is not an issue, then run away now to avoid getting sucked into the dark arts of native memory management”*** Martin Thompson. Founder. *Mechanical Sympathy* community forum.

2.1 Integrate Martin Thompson’s original off-heap Mechanical Sympathy discussion into the article (The “Zen” of going Off-Heap).

<http://mechanical-sympathy.blogspot.de/2012/10/compact-off-heap-structurestuples-in.html>

2.2 Integrate K.D. Gregory’s original off-heap presentation into the article (The “Alphabet” of going Off-Heap)

<http://www.kdgregory.com/misc/java.byteBuffer/kdgregory.com-java.byteBuffer-presentation.pdf>

2.3 Integrate PeterLawrey’s off-heap Masterpiece (openHFT) into the article (The “Shakespeare” of going Off-Heap)

<https://github.com/OpenHFT>

***PART III. CASE\_STUDIES= JCACHE providers willing to build Adapters for off-heap Impls of javax.cache.Cache<K,V>***

3.1 The Infinispan project ([www.infinispan.org](http://www.infinispan.org))

Ben will explicitly show how to adapt the 100% Open-Source Infinispan 6.x project to use Peter Lawrey’s net.openhft.collections.HugeHashMap as an off-heap javax.jcache.Cache Impl

<http://infinispan-developer-list.980875.n3.nabble.com/infinispan-dev-Infinispan-embedded-off-heap-cache-td4026102.html>

3.2 The Hazelcast project ([www.hazelcast.org](http://www.hazelcast.org))

Ben will explicitly show how to adapt the 100% Open-Source Hazelcast 3.x project to use Peter Lawrey’s net.openhft.collections.HugeHashMap as an off-heap javax.jcache.Cache Impl

***PART IV. Recap. Conclusions. Future.***

4.1 Check out the openJDK enhancement proposals for Off-Heap capabilities! Remove the FUD (and, frankly, the technical bigotry) associated with going Off-Heap via sun.misc.Unsafe. Off-Heap PRIDE!

<https://groups.google.com/forum/#!forum/jep-off-heap>