A Profiling Tool for exploiting the use of Packed Objects in JAVA Programs

Umang Pandya, Karl Taylor, Prof Dr. Eric Aubanel and Prof Dr. Kenneth Kent
University of New Brunswick, IBM Canada
Faculty of Computer Science
umang.pandya@unb.ca, Karl_Taylor@ca.ibm.com, aubanel@unb.ca, ken@unb.ca

Packed Objects

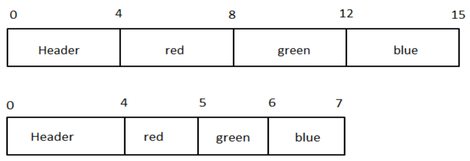
Packed objects[1] can be used to gain greater control over the layout of objects in memory. Applications which uses packed objects have greater flexibility when they work with memory structures that are not in Java code.

Project Goal

- The purpose of developing this profiling tool is to assist software developers in determining how much performance gain can be achieved if they switch to packed objects from standard Java objects
- This switch can result in reduced memory consumption by the application
- If the application is accessing a lot of native data using JNI method calls, then use of the packed objects will eliminate marshaling/unmarshaling of native data into Java objects and thus eliminating redundant data copying

After analyzing the Packed Object Data Model, our Profiling Tool will be searching for following four cases in a Java program.

Primitive data type fields: In the packed object data model, fields of primitive data type, such as byte, char, boolean and short, occupy the minimum amount of space as necessary. This is more efficient especially when compared to an implementation where all primitive data types are stored in either 32-bits (for byte, short, int, float, char and boolean data types) or 64-bits (for long and double data types).



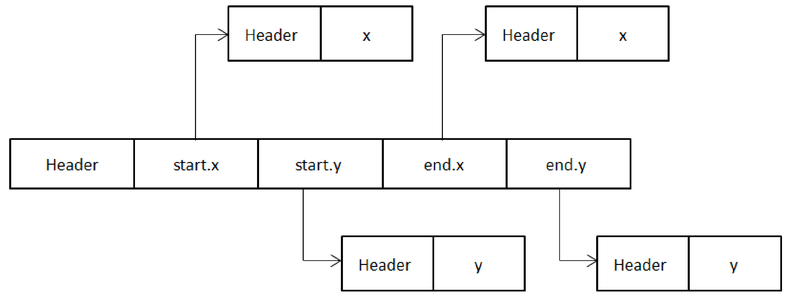
Layout of an object with three byte fields

Packed objects layout of the same object

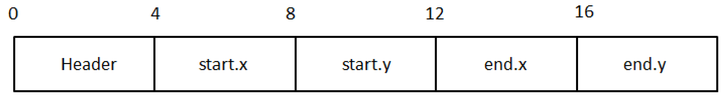
The packed objects data model can be useful, especially when there are multiple fields of small primitive data types.

Reference type objects: If a packed object contains one or more instance fields of packed type then these fields are embedded within the object. Which is in contrast to the generic Java data model where if an object contains a filed of either a primitive or non-primitive data type, then the filed contains a reference to the actual data value or null.

[1]**Packed objects is an experimental feature in IBM J9 Virtual Machine**



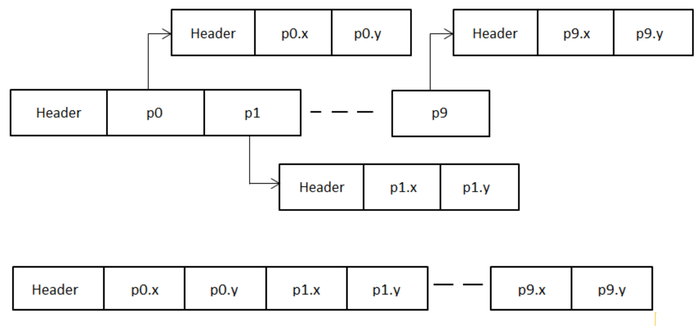
Layout of a reference type object in Standard Java



Layout of a reference type object in Packed objects

The advantage of this property of packed objects is that nested packed fields of a packed object reside next to each other in the memory, which improves cache locality and results in faster data access and also it reduces the overhead associated with arrays.

Arrays: The elements of an array of packed objects are embedded within the array, which is in contrast to a standard Java array of objects, where elements are references which point to the actual values. This results in less overhead and improved object locality.



Data: 80 bytes
Overhead: 88 bytes

Data: 80 bytes
Overhead: 8 bytes

This approach results in significant reduction in memory consumption especially if the application uses large arrays or a large number of smaller arrays.

Off-heap packed objects: We can create a packed object in native memory outside the Java heap and these objects are known as **off-heap packed objects**. These objects are very small and they mainly consists of a pointer to actual data which lies in the native memory.