Using Field Access Frequency to Optimize Layout of Objects in the JVM

Taees Eimouri 1, Kenneth B. Kent 2, Aleksandar Micic 3, Karl Taylor 4
1,2University of New Brunswick, Faculty of Computer Science, Fredericton, Canada
3,4IBM, Ottawa, Canada
{teimouri,ken}@unb.ca,{aleksandar_micic,taylor}@ca.ibm.ca

ABSTRACT
Increasing spatial locality of data can alleviate the gap between memory latency and processor speed. Structure layout optimization is one way to improve spatial locality, and consequently improve runtime performance, by rearranging fields inside objects. This research examines modifying IBM’s JVM with the ability to reorder fields inside Java objects from access frequency information (hotness) in the presence of storage optimization.

INTRODUCTION
Structure layout optimization is performed based on:
• Hotness: the total number of accesses to a particular field.
• Affinity: fields accessed close to each other in time.

HOTNESS ANALYSIS
Step 1: gathering information about particular classes and their non-static fields in a profiling run.

HEURISTICS
IBM’s JVM has its own layout scheme for objects that is well optimized from a memory footprint point of view.

RESULTS
Suppose an object has the following fields in the order defined by the programmer: F1 (4 bytes), F2 (8), F3 (4), F4 (8), F5 (8), F6 (4), F7 (8), F8 (8).

Figure 5: An example of object layout in the different approaches

CONCLUSION
There is almost a 9% increase in the number of cache misses in the worst case and more than a 12% improvement in the best case in different benchmarks. Also, in the worst case, the speed is slowed by around 10%, and in the best case it is improved by more than 20%. In the future, field reordering could be done using affinity as well as hotness. In this approach, fields could be reorganized according to the number of times they are accessed together within a short time interval and as a result, the chance that related fields are placed near each other in the cache would be increased.