Object Layout Optimization in JVM

Taees Eimouri, Kenneth Kent, Aleksander Micic
University of New Brunswick, IBM Canada
Faculty of Computer Science
teimouri@unb.ca, ken@unb.ca, Aleksandar_Micic@ca.ibm.com

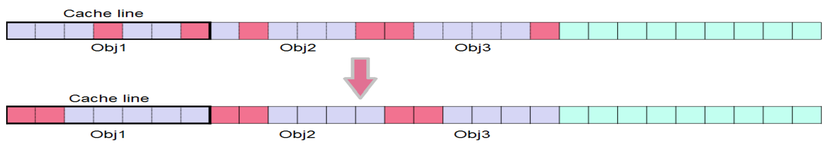
ABSTRACT

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 reorganizing fields inside objects. This research examines modifying IBM's JVM with the ability to reorder fields inside Java objects from access frequency information (hotness), affinity and false-sharing in the presence of storage optimization.

INTRODUCTION

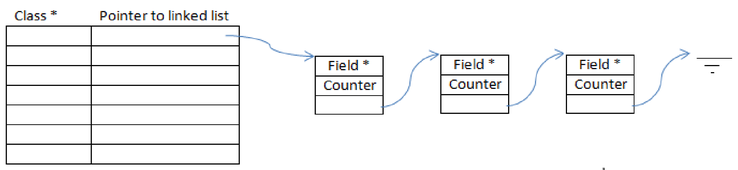
Objects on the heap access each other's fields.

- Some fields are accessed more often → Hot
- Some fields are accessed close to each other → Affine



HOTNESS ANALYSIS

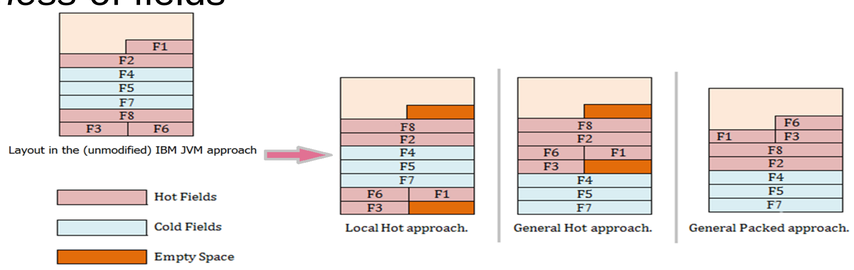
Gather information about particular classes and their non-static fields in a profiling run.



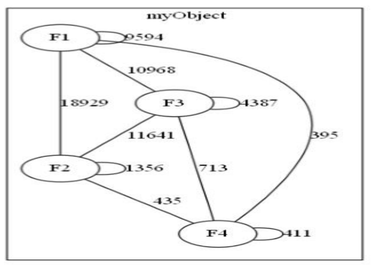
HEURISTICS

In IBM's JVM, fields are organized inside objects based on their size to optimize the memory footprint. We implemented object layout optimization based on:

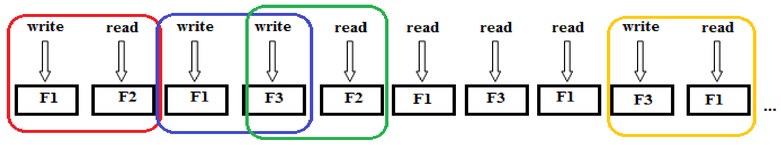
1- *Hotness* of fields



2- *Affinity* among fields



3- *False-sharing occurrence possibility* (in multi-threaded applications running on multi-core systems)

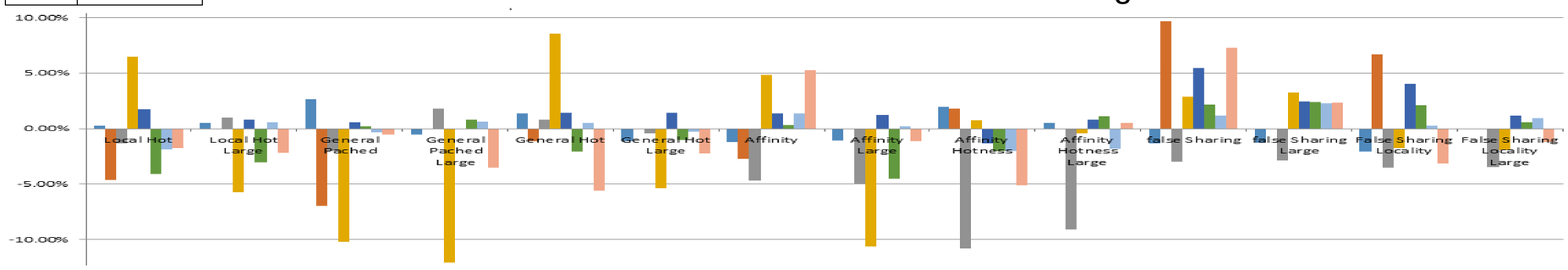


CONCLUSION

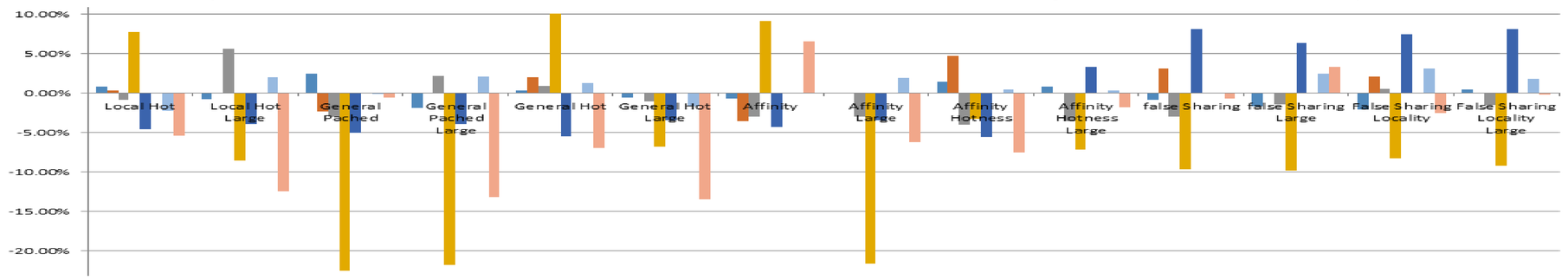
Our results show:

- Improvements in some benchmarks (benchmarks show different behaviors in different approaches).
- Locality has a greater effect on optimizing object layout than false-sharing.

RESULTS



A comparison of the number of cache misses in different benchmarks (negative numbers show improvement).



A comparison of the execution time in different benchmarks (negative numbers show improvement).