
Some Java compilers use an optimization called object inlining. In the usual form of this optimization, if the compiler is certain that an object A is pointed to only once, from another object B, then A can be allocated as part of B’s heap record, rather than in its own, separate heap record. This saves the cost of dereferencing the pointer to A, and the heap allocation and garbage collection overheads associated with the second record.

a. Based on the ideas in the assigned paper by Chilimbi, et al., argue why object inlining is not necessarily a good idea. Sketch some Java code that might be made slower if object inlining were performed.

b. [For discussion.] How should a compiler decide which kinds of object layout rearrangement to use?


a. Is it possible to remove all measurement bias by deliberately varying experimental setups?

b. Give an example from the paper where causal analysis would be made easier if manufacturers provided more features or information about their processors.

c. [For discussion.] Suggest some other possible sources of measurement bias besides memory layout.

d. [For discussion.] A later study (Curtsinger and Berger, “Stabilizer: Statistically Sound Performance Evaluation,” ASPLO2013) suggests that, after controlling the effects of memory layout, the difference between -O2 and -O3 level optimizations in LLVM is indistinguishable from random noise. In light of this, should LLVM simply abandon attempts to optimize beyond the -O2 level?