

Safe C using Fat Pointers

Miguel Silva
15-745 Spring 2009

Goal

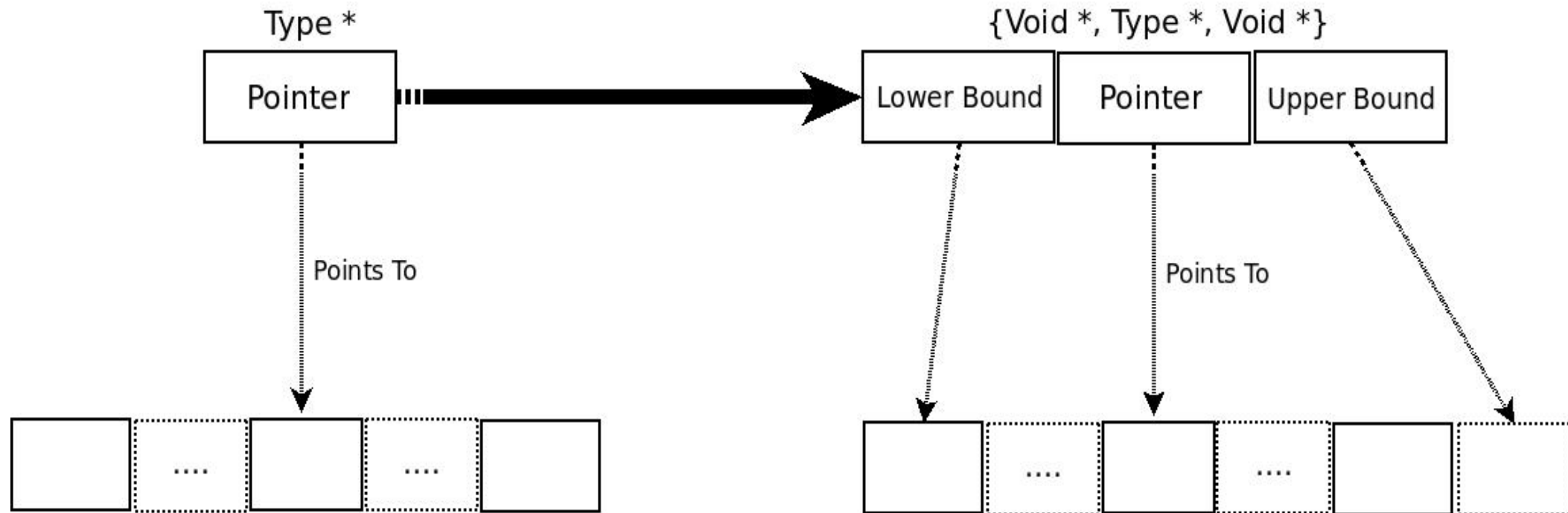
- Avoid memory errors in unsafe languages such as C
 - Replace some pointers with fat pointers that can be checked dynamically
 - Reduce overhead

Type of errors detected

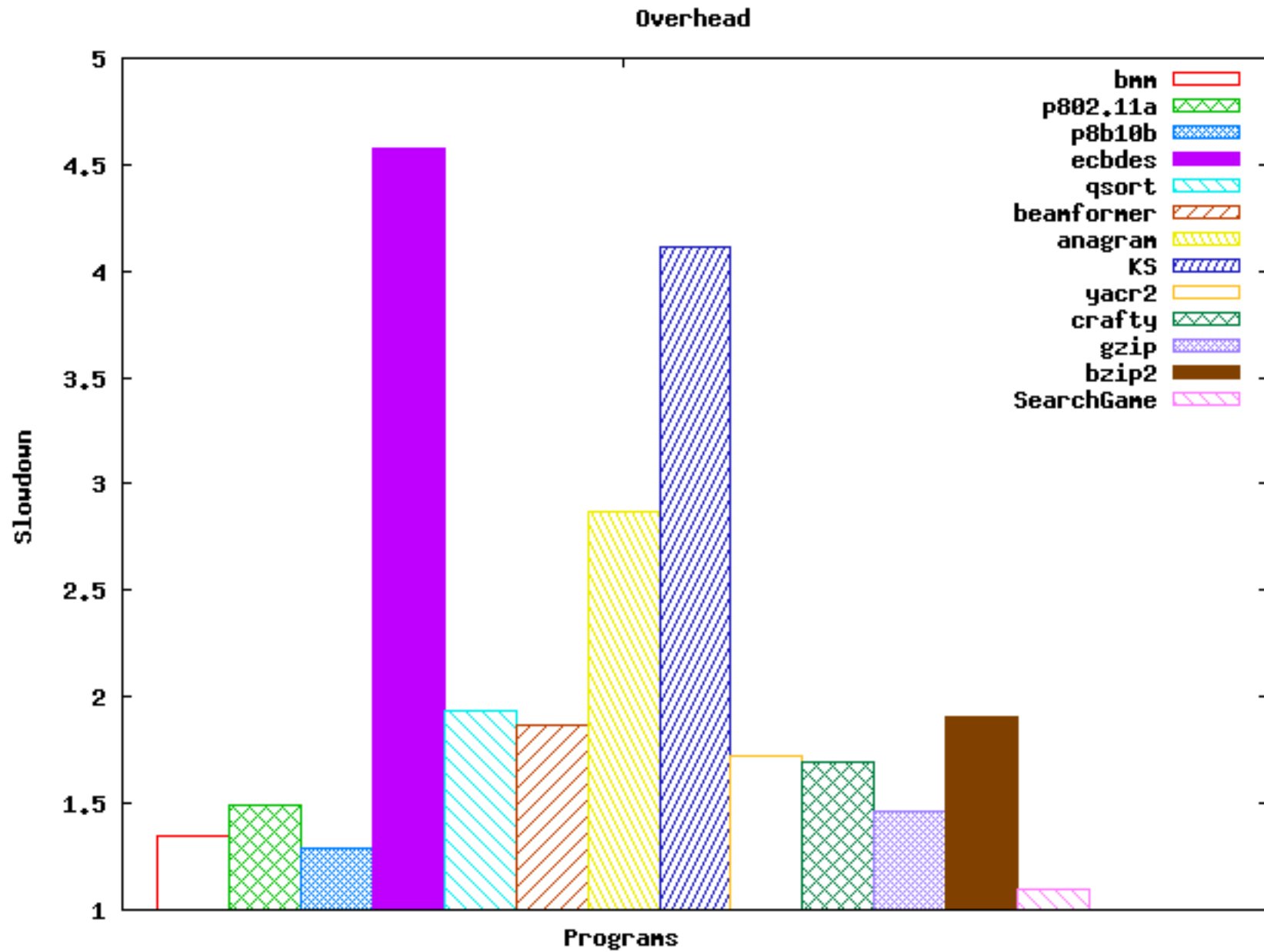
- Array Out of Bounds
- Integer Dereference
- Null Dereference

Implementation

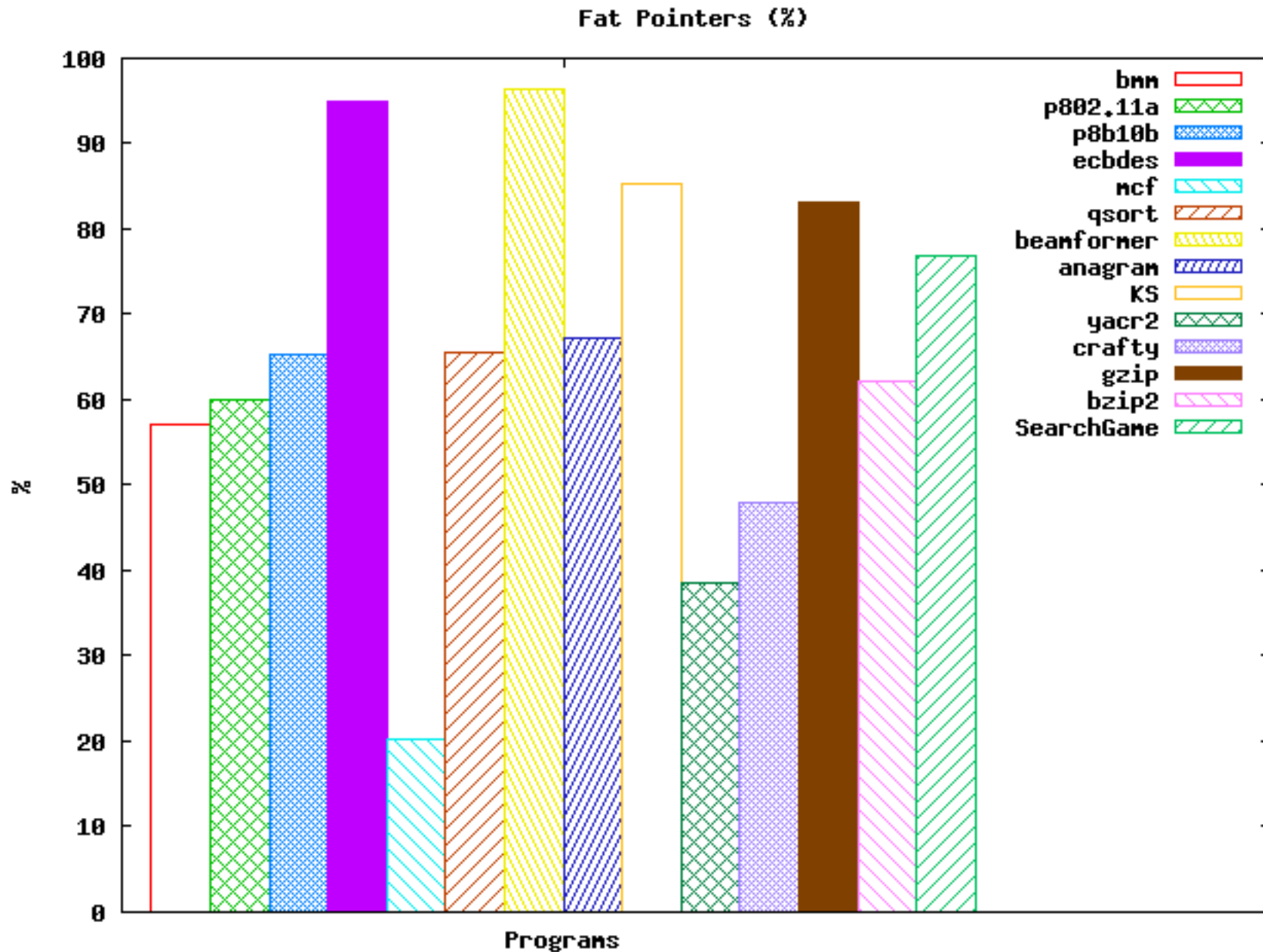
- Every pointer that is subject to pointer arithmetic becomes a fat pointer
- Type inference algorithm



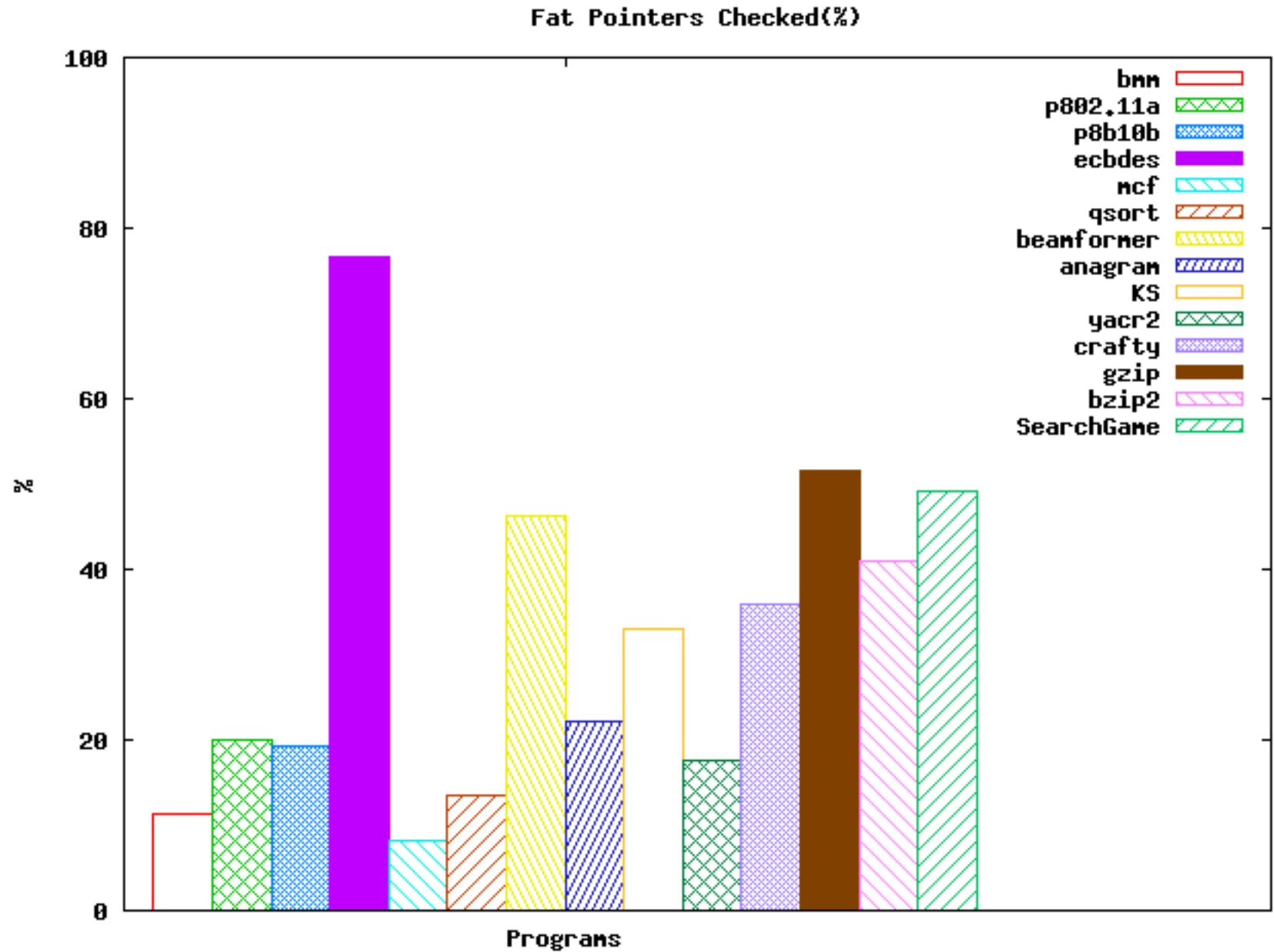
Overhead



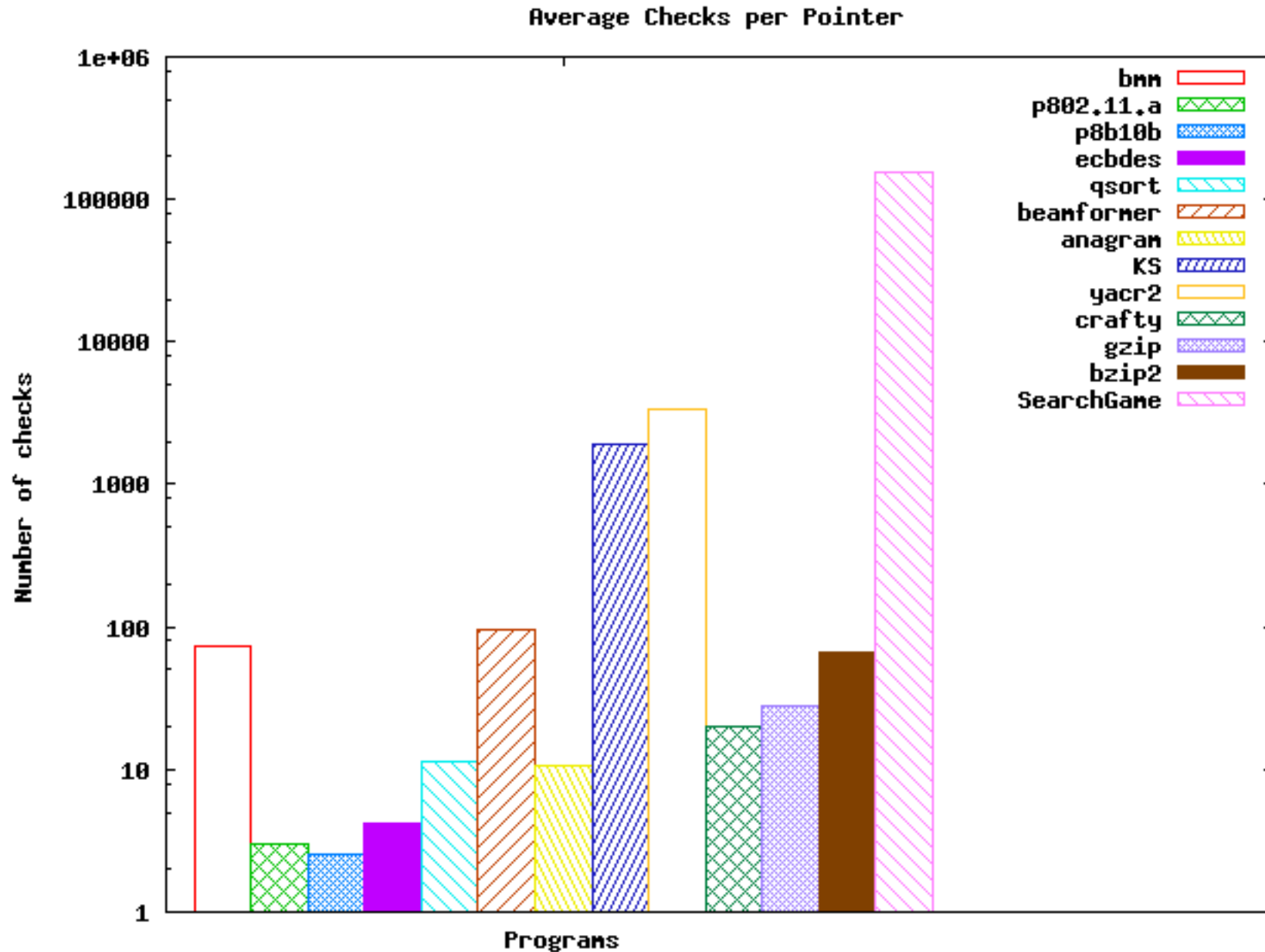
Percentage of Fat Pointers



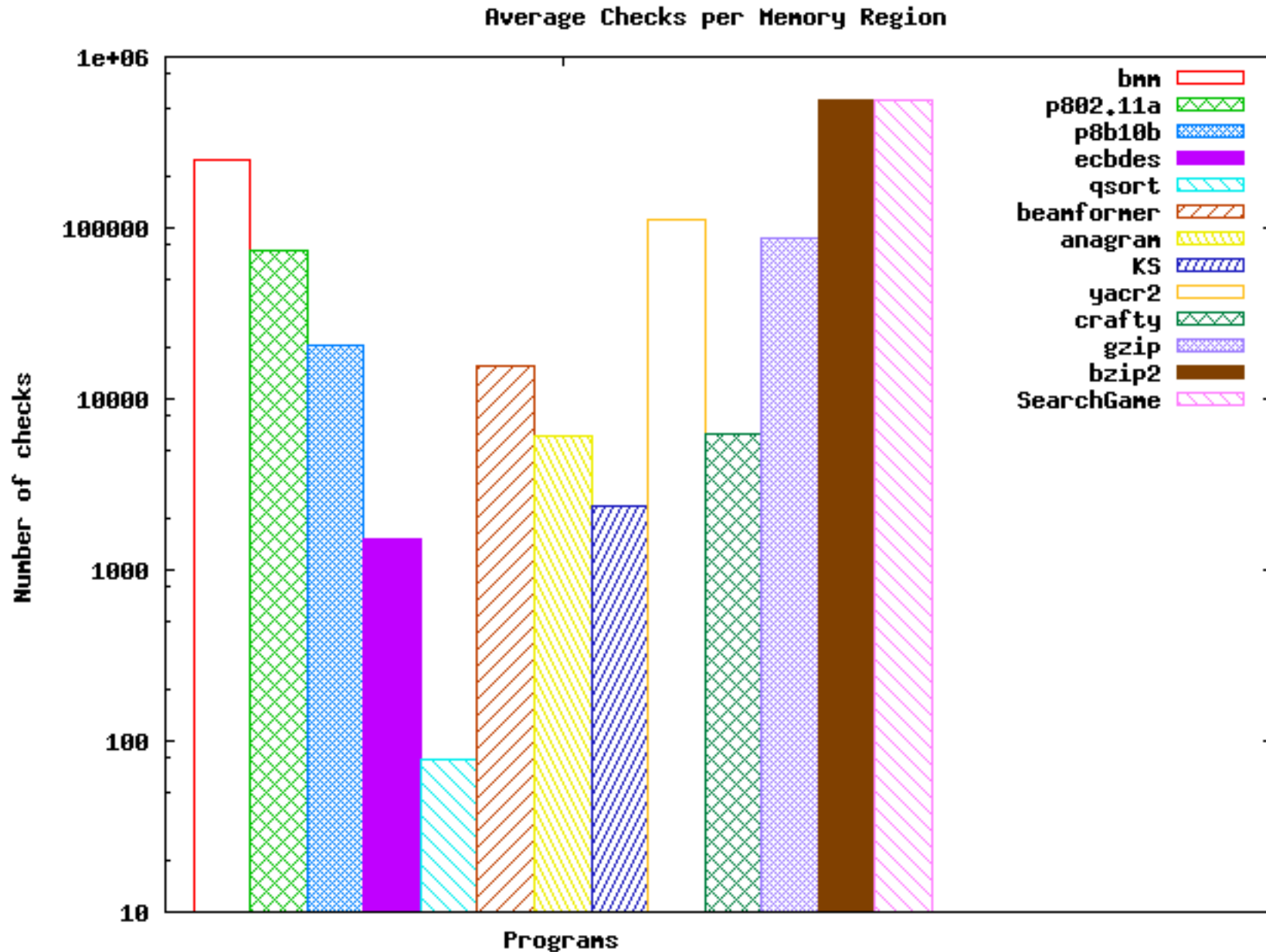
Percentage of checked Fat Pointers



Repeated Checks



Repeated Checks in the same region



Solutions

- Partial Redundancy Elimination
- Loop Bounds Check
- More Efficient Checks