EECS 583 – Homework 2

Fall 2011
Assigned: Mon, October 3, 2011
Due: Fri, October 21, 2011 (11:59:59 pm)

Speculative LICM

The goal of this homework is to extend the LLVM loop invariant code motion (LICM) optimization to identify more opportunities for optimization using *memory dependence speculation*. LICM of loads/stores is limited by the quality of the memory dependence analysis to disambiguate potential invariant memory operations from other memory operations inside the loop. Often, memory disambiguation returns an answer of 'maybe' because it can neither prove nor disprove that two memory operations alias. To overcome this limitation, your job is to extend LICM by breaking unlikely memory dependences using statistics about how often memory dependences occur during a sample run of the program (memory profile). This will enable you to hoist not only more loads out of loops, but other instructions that solely depend on hoisted loads can also be moved out of the loop.

A pass that performs loop-aware memory profiling (LAMP) provided in HW1 collects statistics on how often the addresses of two memory operations overlap for every pair of load/store instructions. You will use these statistics to identify profitable opportunities to perform speculative LICM. However, it is not that easy. The memory profile data can obviously be wrong and the situation where speculation is incorrect must be handled. Thus, you need to check for mis-speculation and correct execution when it happens. With LICM, this can be done by simply re-doing a subset of the code that was hoisted. Think carefully about this.

3 Parts

There are really 3 parts to this homework: (1) Extending LLVM LICM to use the memory profile data to override static memory dependence analysis (see llvm-source-dir/lib/Transforms/Scalar/LICM.cpp); (2) Modifying LICM to insert the proper checks into the loop to identify mis-speculations and perform the necessary fixup; and (3) An intelligent heuristic of when to apply speculative LICM when profitable (i.e., there is a net gain in performance by performing the transformation and handling mis-speculations. Part (2) is the hardest part of this assignment and you will be provided some code to assist you. Note that part (2) can also be done in a variety of ways with different performance overheads, so you will need to carefully consider your strategy here and obviously your heuristic depends on your checking strategy.

Submission

You should submit a single .tgz (gzipped tar) file into the directory /y/submit on andrew.eecs.umich.edu via scp. Please name your tar file uniquename_hw2.tgz (don't ovewrite your hw1 by mistake!). Your tar file should contain:

1. Source code (complete project dir and no binaries) for your LLVM pass
2. Bitcode files and binaries (after performing speculative LICM) for each benchmark (both performance and correctness) compiled on one of the hurricane machines. Name the files according to the benchmark name.
3. README that summarizes what you did and what works/does not.

Contest

The person with the fastest average execution time across the benchmarks will be crowned EECS 583 F11 Optimization Champ and be awarded a fabulous prize.