Incremental Path Profiling

Kevin Bierhoff and Laura Hiatt

April 5, 2006

1 Team Information

This project will be done by Kevin Bierhoff (kevin.bierhoff@cs.cmu.edu), and Laura Hiatt (lahiatt@cs.cmu.edu).

2 Website

Our project's website is at http://www.cs.cmu.edu/~kbierhof/15745/.

3 Project Description

3.1 Goal

The goal of our project is to look at what we will call "incremental path profiling". Path profiling is an important part of compiler optimization; however, its computational overhead is quite high. Our project's aim is to lessen the computational overhead of path profiling while still providing useful information about the hot paths of the program being compiled. We will do this by incrementally increasing the number of paths we profile by slowly adding instrumentation along high-use paths. By doing this, we hope to show that partial instrumentation of paths incurs less overhead than full path profiling while still providing enough information about which paths are hot to be useful in optimizations.

3.2 Background

Path profiles can be used by compilers for better optimization of programs. In particular, the "hot path", the most frequently taken path through a function, can receive special treatment that results in faster execution (Ammons & Larus, 1998). While somewhat impractical for normal compilers, these optimizations are commonly performed by modern Just in Time (JIT) compilers. Unfortunately, path profiling is expensive. For example, the original "efficient" algorithm proposed by Ball & Larus (1996) is reported by the authors

to have 30% overhead. For a JIT compiler, 30% are a very high cost because they can be directly perceived by the end users of the software. One remedy is to work with edge profiles. Edges profiles are cheaper to acquire, but they also often fail to identify the hot path correctly (Ball & Larus, 1996, p. 10). Both edge profiling and path profiling result in "complete counts": After execution, the compiler knows exactly how often each edge (path) was taken. This project investigates a technique for picking out the hot path with less overhead than full path profiling.

3.3 Approach

Our project has several steps. The first is to implement incremental path profiling. Our implementation will begin by only profiling a single branch. After it is clear which branch is hotter than the other, the algorithm will instrument the next branch along that hotter path. This process recurses until it knows which path is the hot path.

Once this implementation is complete, we will need to run several experiments in order to compare this approach with the more standard profiling algorithm. We will time how long it takes each method to identify the hottest path using a variety of test cases.

We will use time as our metric. We hope to show that incremental path profiling is a more efficient way of identifying the hot path; barring this, we hope to at least encounter some situations in which incremental path profiling is a better choice for instrumentation than traditional path profiling. An additional metric we will consider is the amortized cost of our experimental approach, in order to ensure that the total process of incremental path profiling does not exceed the traditional approach.

3.4 Plan

We shouldn't need any additional resources other than the ones we've been using in class. Our two milestones are completing the implementation, and running the experiments. If our progress is not as good as expected, or some aspect of the implementation fails, we will attempt to simplify the incremental algorithm by instrumenting more paths at a time, or randomizing which paths it instruments. If our experimentation fails to show that instrumental path profiling is faster, we will attempt to think of non-traditional situations in which it is, in fact, a better choice.

References

Ammons and Larus. Improving data-flow analysis with path profiles. PLDI'98.
Ball and Larus. Efficient Path Profiling. MICRO'96.