Programming for Performance: Assignment 1

Patrick Lam

Due: January 27, 2011

The goals of this assignment are to get you to 1) work with profiling tools on real-world code; and to 2) think about algorithmic improvements. You may work on this assignment in groups of 2, or you may hand in an individual assignment.

I'll suggest some applications for you to work with, but I encourage you to go and find your own applications to profile and improve. I haven't looked at these applications in any detail, so you shouldn't think they come with any guarantees. If you do choose your own application, it should be at least moderately complicated (say 20k lines of code) and have some userbase.

- Rascal (C), an RDF graph library: `http://librdf.org/rasqal`.
- rspamd (C), a spam filtering library: `https://bitbucket.org/vstakhov/rspamd/wiki/Home` (plugin-based; may be tricky)
- relax (C), molecular dynamics software: `http://www.nmr-relax.com/`
- 0 AD (C++), a real-time strategy game: `http://trac.wildfiregames.com/wiki/GettingStartedProgrammers`
- SweetHome3D (Java), an interior design application: `http://www.sweethome3d.com`

Profiling (60 points)

The first part of this assignment is to choose an application, get it running, develop a workload, and collect data about it.

Your workload should be something that takes a measurable amount of time to run; it seems like it should take at least 15 seconds to run on your system.

I'm not going to specify a profiling tool to use, but you can use any of the tools that I've described in class (or others). Tell me about what you discover using your profiling tool, and how you might be able to improve the code based on the profiling results. In particular, you should submit a written document (5 pages sounds about right to me), describing the following:

- characteristics of the software: language and platform, size of the codebase, how actively-maintained is the codebase, any design information you can find (does it follow a design pattern?), userbase;
- a sample, reproducible workload which tests some aspect of the performance (you might want to put this on the Internet);
- baseline performance data on this workload;
- the profiler you are going to use; and
- profiling results for the workload, and discussion of these results.

I encourage you to include results from CPU-level monitoring tools e.g. CodeAnalyst or Shark.

Your workload might be a synthetic one which calls into a particular part of your application (like a unit test), especially if you are profiling a graphical application.

Code Improvement (40 points)

Now that you've collected data on the baseline performance of the program and on its hotspots, your next task is to make the program faster. I expect you to implement a change to the program which makes it faster. You may change the program's specifications as long as your change is reasonable; for instance, you can compute something less expensive than what the original program computed, if you can say why the change is unlikely to be a problem. An example of an inappropriate (and un-useful) change would be pulling the type checker out of a compiler.

Provide details of the system performance before and after your change, including enough information about the workload that we can reproduce your observations. I suggest that you hand in something like:

- diffs for your changes;
- a discussion of why these changes improve performance; and
- evidence that your changes do improve performance.