Using Instruments and Shark to analyse and improve stability and performance in software development

Project specification
2010-02-12

Supervisor:
Mads Dam
<mfd@kth.se>

Participant:
David Rönnqvist
870827-0239
<ronnq@kth.se>

Summary

There is a lot more code in software today and the amount alone makes it hard to keep track of everything. This makes it difficult to keep the code free from flaws. There are tools available to help developers and my intention is to use such tools to investigate how they can help me in the software development process.

Background

As hardware gets better and cheaper the demand for more functionality in the software increases. Software projects in turn becomes bigger and harder to overview.

When software gets bigger and even more complex it gets even harder for the developers to maintain the code and keep it free from bugs and other errors. Most errors in software is because of human error in one way or another.

There are some great tools available today that wasn't available a couple of years ago to help developers test their code for errors and bottlenecks in performance but they haven't had much attention. For the Mac OS X platform[1] there is, among others, Shark and Instruments; both bundled with the rest of the developer tools.

Shark is a very low profile statistical profiler. Statistical profilers collect runtime information from a system or application at regular intervals. The collected data is never exact, only a statistical approximation. This may sound like a big flaw but there is a cost for achieving exact data. Tools gathering exact data can have serious effect on the system or application performance. So in practice statistical profilers often provide a more accurate result than other approaches because of their very low overhead.

Instruments is a performance analyzer and visualizer based upon DTrace which is tightly integrated with both the operating system and the rest of the developer tools for Mac and iPhone. DTrace uses probes that triggers specific actions when conditions are met. A typical probe might "fire" when a file is opened or some line of code is executed and then analyze the run-time situation at that point.

Problem formulation

How can modern analysis tools help developers track down bugs and write better code?

Project plan

I intend to read and write about how the different techniques work; statistical profiling and instrumentation. I will discuss their strengths and weaknesses and describing why these are.

I aim to use these modern code analysis tools to investigate how they can help the developer and improve the software being developed. I will document how I use them, the time it takes and the results I gain.

I'm planning to write an application in two phases using these tools rigorously during the entire development process, both phases. In the first phase I plan to develop the program while continuously analyzing it for leaks and other kind of errors and critical performance issues. In the

[1] both computer and iPhone OS are Mac OS X platforms

second phase I plan to take the completed application and thoroughly track down bottlenecks and tune and reanalyze the application until it performs well.

Throughout both phases I intend to document closely the way I use these tools and what I gain in terms of runtime performance and bugs found. This will allow me to track down the cost in time and effort and compare this with an estimate of the time it could have taken me to make these or such improvements myself.

Schedule

During the entire project I intend to read as much about these tools as I can, at least one paper about DTrace, one about statistical profiling and one about general software performance.

I plan to use the first few days to limit the development project to something that I alone could write in little less than two weeks.

I then plan to spend about one entire week to analyze the performance and fine tune the code, documenting the work and the performance increases it gives.

References

- Shark User Guide (http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/SharkUserGuide/SharkUserGuide.pdf)
- Instruments User Guide (http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/InstrumentsUserGuide.pdf)
- Dtrace User Guide (http://dlc.sun.com/pdf/819-5488/819-5488.pdf)
- Low Overhead Memory Leak Detection Using Adaptive Statistical Profiling (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.6156&rep=rep1&type=pdf)
- Using DTrace to Profile and Debug A C++ Program (http://developers.sun.com/solaris/articles/dtrace_cc.html)
- Dynamic Instrumentation of Production Systems (http://www.sun.com/bigadmin/content/dtrace/dtrace_usenix.pdf)
- Need for speed -- Eliminating performance bottlenecks (http://www.ibm.com/developerworks/rational/library/05/1004_gupta/)
- Hidden in plain sight (http://queue.acm.org/detail.cfm?id=1117401)