Instrumenting V8 to Measure the Efficacy of Dynamic Optimizations on Production Code

Michael Maass and Ilari Shafer
{mmaass, ishafer}@cs.cmu.edu

April 17, 2012

Project Page: `http://www.cs.cmu.edu/~ishafer/compilers/`

1 Major Changes

We have no significant changes to our project plan. The primary broad revision to our plan is that we do not expect to focus on experimenting with another platform (e.g. node.js) given time constraints and the unforeseen obstacles we have faced in adding instrumentation and running tests (see Section 4).

2 What We Have Accomplished So Far

The revised schedule in section 5 summarizes our accomplishments to date fairly succinctly. We successfully acquired and compiled both Chromium and V8, and we have learned key elements of both projects. Having obtained sufficient insight into the inner workings of the compiler and its broader context, we instrumented the build of Chromium we use for testing. We have written a series of tests (using a tool called Sikuli) to perform typical operations on real world web applications. In combination with these pieces, we have constructed a framework to toggle optimization passes in V8, and are currently using it to mostly automatically run our test cases. We are currently collecting benchmarking data in the form of internal counters and the time required to run V8 operations such as compiling or running JavaScript. A brief preview of our results so far is available at `http://www.cs.cmu.edu/~ishafer/compilers/#report`

3 Meeting our Milestone

In our project proposal, we wrote:

> For the project milestone on Thursday, April 19, we plan to have completed the framework for evaluating optimizations, and have experimented with one or two real-world applications.

We have met (and slightly exceeded) this milestone. Our basic framework for evaluating optimizations is complete, and we have experimented with the four real-world applications (and benchmark) described in Section 2.

4 Surprises

One significant surprise thus far has been the difficulty of instrumenting the V8 compiler within a browser. Though V8 contains a profiler that can be used by developers from an in-browser utility, and the V8 command-line shell has support for reporting many internal counters, these are not exposed by Chromium. We have adapted the internal-counter reporting framework from the V8 shell into the Chromium source

and added an additional counter for the number of times on-stack replacement with deoptimized code occurs (we will describe the significance of this counter in the final report). Furthermore, we cannot collect counters and profiling information simultaneously.

Another unanticipated obstacle has been the painfulness of testing with real-world applications. After our initial discussion we de-emphasized the benchmarking aspect of the project to instead focus on selective optimization. Unfortunately, simply getting to the point of running consistent real-world scenarios has required writing UI-based tests, which have been exceptionally temperamental. We have automated the tests, but real web applications have significant response-time variation and occasionally our UI testing framework's computer vision engine fails to find click targets. This means seemingly simple behaviors like "click a submit button once an image has been uploaded" have required care and supervision of tests.

5 Revised Schedule

Week Beginning	Plan for the Week
March 26	*Done:* Turn off all runtime optimizations, and build framework for selectively enabling single optimizations. Build test scripts to perform typical operations on real web applications.
April 2	*Done:* Identify metrics for evaluation. Learn where and how to instrument V8, produce a simple modification (such as emitting information from within an optimization). Test one optimization.
April 9	*Done:* Identify set of real-world applications and apply selective optimization to a few of them. Begin instrumenting metrics for evaluation (e.g. number of instructions executed) to determine benefit of optimizations.
April 16	Continue running set of real-world applications with instrumentation. Finish instrumenting evaluation metrics. If significant extra time, begin experimenting on another platform
April 23	Transform raw data into condensed results, establish results. If time, finish experimenting with other platform.
April 30	Complete project writeup and poster

6 Resources Needed

We will need no additional resources to complete the project. In addition to our additional list of required resources, we have obtained a few VMs for testing purposes. We will explain how these were used in the final report.