Native Client: Running Native Code in the Browser and Comparing its Performance to JavaScript

Dennis Jarosch
me@dennisjarosch.de

May 13, 2011
He won't be talking about a new Microsoft technology, will he?
Agenda

• About Me
• Why Native Code in the Browser?
• The Native Client Project
• Web Application Benchmarks
• Benchmark Results and Conclusion
• Live Demo!
About Me

- Doctorate student at Berlin's Humboldt University
- Former director of an online gaming company
- Led the software development team for 5 years
- Open Source advocate and Linux user
Introduction
Introduction

- The World Wide Web is in a transition from interactive web sites to web applications
- Bandwidth and connectivity are no longer a problem (in many parts of the world)
- Modern web applications offer the functionality and complexity of native applications
- Always accessible, regardless of the device
Evolution of the World Wide Web

World Wide Web

The WorldWideWeb (W3) is a wide-area hypertext information retrieval initiative aiming to provide universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including:
- executive summary of the project
- Mailing list
- Policy
- November's W3 news
- Frequently Asked Questions

What's out there?
- Pointers to the world's online information, subjects, W3 servers, etc.

Help
- on the browser you are using

Software Products
- A list of W3 project components and their current state. (e.g. Line Mode, X11 Viola, NextStep, Servers, Tools, Mad Robot, Library)

Technical
- Details of protocols, formats, program internals etc

Bibliography
- Paper documentation on W3 and references.

People
- A list of some people involved in the project.

History
- A summary of the history of the project.

How can I help?
- If you would like to support the web.

Getting code
- Getting the code by anonymous FTP, etc.

1992

2011
Web Applications

- Web applications are operating system independent – must not be installed
- Require a web browser and an active Internet connection
- Run on mobile devices, with cloud computing performing the heavy lifting
- Cloud computing: seemingly infinite computing resources on a pay per use basis
As more and more people *live* on the Web, the web browser becomes a dominant platform for application development.
Browser as an Application Platform

- Seamless Internet access
- High level programming languages, e.g. JavaScript
- HTML and CSS for user interface and presentation
- HTML5 enables richer web applications, without the need for proprietary plugins
The web browser has become an appealing platform, but lacks computational power.
JavaScript

JavaScript has become the programming language of the Web.

Browser competition has led to fast JavaScript engines. But are they good enough to take web applications to the next level?

Image source: blog.michaelrice.com
Why Native Code in the Browser?

Native code could enable performance critical web applications, such as 3D Games, CAD tools, and audio and video editors.

Image sources: www.hotdoor.com and www.pitivi.org
The Native Client Project
Native Client - NaCl

- Open source project by Google
- Goal: to run compiled native code in web applications
- Maintains browser neutrality and OS portability
- Compiled NaCl binaries can run unmodified on different operating systems
- Mozilla and Opera have already declined
Native Client - NaCl

Designed explicitly for

Security & Speed

Image source: www.openclipart.com
Native Client – What Else?

- Supports threads, assembler code, and SSE
- x86 (32- and 64-bit) and ARM support
- Applications can be written in C and C++
- Developers can re-use existing code – low porting overhead
- NaCl SDK is based on gcc tool chain, many libraries have already been ported
Native Client Architecture

- Browser plugin using the Pepper 2 API
- Uses sandboxing to separate trusted and untrusted components
- Components communicate via Inter-Module Communications (IMC)
- CISC software fault isolation, using x86 segments
- Static analysis is used to detect defects in untrusted code
Native Client Architecture

The NaCl module runs in a sandbox – in a separate process from the renderer.

Sandboxes are considered trusted, while code running in sandboxes is untrusted.
How Does it Work?

User navigates to website

JavaScript loads and Invokes the NaCl module

No pop-ups!
Three benchmarks to compare the performance of C, NaCl, and JavaScript, in respect to number crunching, 3D graphics, and data-intensive tasks.
Pi-Benchmark

Computes the number pi using the Leibnitz formula, for a given number of iterations.
Gears-Benchmark

A port of the infamous glxgears application from C to NaCl and JavaScript (WebGL).
Spectral-Benchmark

Performs a spectral analysis of an audio file.
Benchmarking Process

- Benchmarks were carried out on 64-bit Ubuntu Linux 10.10 on a MacBook Pro (5, 5)
- Google Chrome 9 was used for JavaScript and NaCl benchmarks
- NaCl SDK 0.1.507.0 with NPAPI
- Multiple runs were benchmarked for each round and arithmetically averaged
Pi Benchmark Results

The graph shows the performance of different programming languages in calculating Pi, with iterations on the x-axis and time in milliseconds on the y-axis. The languages compared are C, NaCl, and JavaScript.

- **C** represents the fastest performance, with the shortest time required for each iteration range.
- **NaCl** is slower than C but faster than JavaScript, as indicated by the higher time values.
- **JavaScript** is the slowest among the three languages, requiring the most time for each iteration range.

The data points for each language form clusters, with C showing the tightest clustering, indicating consistent performance, while JavaScript shows the widest spread, indicating variability in performance across iterations.
Gears Benchmark Results

![Graph showing benchmark results for C, NaCl, and JavaScript.]

- **Size of the drawing area in pixels**
- **Frames per second**

- **C**
- **NaCl**
- **JavaScript**
Spectral Benchmark Results

The chart shows the performance of different programming languages and memory sizes on a spectral benchmark. The x-axis represents the time in milliseconds, and the y-axis represents different programming languages and memory sizes.

- **C**: The chart indicates that C performs exceptionally well, with times ranging from 0 to 10,000 milliseconds, depending on the memory size.
- **JavaScript**: The performance varies, with times ranging from 30,000 to 40,000 milliseconds for smaller memory sizes, and up to 90,000 milliseconds for larger memory sizes.
- **NaCl**: Similar to JavaScript, NaCl's performance also varies significantly, with times ranging from 8,000 to 90,000 milliseconds for different memory sizes.

The chart highlights the significant performance advantage of C over JavaScript and NaCl, particularly with larger memory sizes.
Results Summarized

- Both C and NaCl are roughly twice as fast as JavaScript in the pi benchmark
- C and NaCl outperform JavaScript almost by factors 10 to 15 in the gears benchmark
- NaCl disappoints in the spectral benchmark – JavaScript is approx. 8 times faster
Conclusion

Native Client shows a strong performance in number crunching and 3D graphics – is not yet suitable for data intensive tasks.

Image source: www.openclipart.com
Three Things to Remember

• Compiled native code could provide more computational power to web applications

• The Native Client project makes it possible to run native code in the web browser

• NaCl shows strong performance for number crunching and 3D graphics – lacks in data intensive tasks
Live Demo!
Thank You!

Questions?
Contact

Dennis Jarosch

me@dennisjarosch.de
http://dennisjarosch.de

Follow me on Twitter: @dennisjarosch