Chris Laffra

Staff Software Engineer at Google NYC

laffra@gmail.com 919-200-0160 http://chrislaffra.com 431 W 37St, Apt 7G, NY, NY 10018

Summary

My interests are in the design and implementation of programming languages, compilers, IDEs, toolkits, run-times, APIs, and life cycle development tools. Examples are tools to monitor and analyze performance, support agile development workflows, or to visualize internal execution. I have worked on various complex software projects, such as Eclipse, and I can understand them quickly, improve their performance and/or effectiveness, and make them more accessible to others by writing books or training materials. Check out chrislaffra.com for links to my work on algorithm visualization.

Experience

Staff Software Engineer at Google

March 2014 - Present (2 years 9 months)

Projects I worked on include:

* Cloud development tools
* Android apps for location people, offices, and meeting rooms
* Tools for discovering other people and projects
* Agile scrumboard tools
* Affinity graphs of Googlers to improve internal search queries, office allocation, and promotion
* The future of Google Video Conferencing

Technology Enablement at Bank of America

May 2010 - February 2014 (3 years 10 months)

Member of the Quartz Core team that is building a Python-based strategic platform for Bank of America/ Merrill Lynch with real-time risk assessment capabilities across asset classes. The platform offers numerous in-house solutions for storing trading data in a graph/object database, delivering centrally versioned applications, developing both web and rich user interface solutions, supporting scalability though cloud-based hosted execution, and implementing an agile development process.

In addition to working on various components in the project, I wrote a large majority of the available documentation and training materials. I also played a big role in supporting an ever growing number of developers on the project, from the handful during our initial startup mode, to many thousands that make up the community today. I played a big role in adoption of the platform through various internal support media, in addition to playing an active role in both problem resolution and support.

To make our training efforts as flexible and effective as possible, I developed a special training environment, the Quartz Academy, which has been used to teach thousands of hands-on and interactive classes on various topics from introductory sessions to technical components to software development life cycle approaches. As part of that effort, I also developed a multi-day introductory/advanced Python tutorial with numerous assignments. I have designed classes, taught them, and built a global training team to teach our technology in various global locations.

EGL/RBD/Rich UI Architect at IBM Rational

November 2006 - May 2010 (3 years 7 months)

Architect and chief contributor of EGL Rich UI; a technology that adds support for web2.0, Ajax, Dojo, and rich user interfaces to IBM's business development language called EGL. I also developed various iPhone web apps, such as conference schedulers by scraping static HTML pages and turn them into scheduling apps.

Added support for Rich Internet Applications, Web2.0, and Ajax to IBM's EGL Language.

This role involved development of a set of browser widgets, runtimes, debug support, supporting a large number of browsers, and JavaScript generation from the EGL language. I managed the R&D, analysis, design, implementation, and marketing of numerous EGL Rich UI solutions. Hands-on Product Architect of Rational Business Developer (RBD). During this time, I published 3 patents on the technology.

I presented at various conferences, including SD West, where I won the Developer Bowl, beating the Google/ Intel/CodeGear teams.

Performance Engineering Team Lead at IBM Rational

July 2005 - November 2006 (1 year 5 months)

Analysis of large commercial IBM products based on Eclipse, such as Rational Application Developer (RAD). Goal was to improve performance behavior caused by time spent in network I/O, reading jars from the hard drive, and virtual memory access. Such behavior can manifest itself at the OS level, around Java components, or with even higher level Eclipse plugins. A full understanding of the entire stack is needed before solutions to improved behavior can be implemented.

I collaborated with a wide variety of internal partners and customers to analyze, understand, and improve startup time and memory consumption of large installed WebSphere and Eclipse applications. I developed a strategy to load WebSphere jar files in a JVM native format, to allow multiple instances to share the memory behind the jar files and reduce overhead of class loading. This self-driven work led to 2.5X improvement of scalability of WebSphere instances per server. I then applied similar techniques to desktop applications, such as RAD, thereby improving startup times by 2X and memory consumption by 35%.

I published on Eclipse performance topics at various conferences.

Senior Software Engineer at IBM Ottawa Labs

January 2003 - September 2005 (2 years 9 months)

Senior contributor to IBM WebSphere Studio Device Developer. Performance analyst to make Eclipse run best on J9 (IBM's Java virtual machine).

I co-authored the "Official Eclipse 3.0 FAQs" book in the Eclipse series on writing plugins for the Eclipse/OSGi platform. My main personal reason for writing the book was to force myself to thoroughly learn the platform. The book includes a chapter on how to write your own IDE. There are so many steps involved, that I decided to write a tool to generate IDEs from a language grammar. That eventually led to the Eclipse IMP project.

During my time in Ottawa, I analyzed the performance of Eclipse applications and developed innovative tools to analyze and visualize the execution inside the OSGi component system that makes up Eclipse. I developed solutions to optimize Eclipse startup and memory consumption though persisting class file formats in a VM-internal format to be loaded at runtime using memory mapping. I negotiated adoption of these techniques with both the J9 team, Eclipse core runtime team, and consumers of the platform.

Technical Lab Director at OTI Amsterdam

August 1999 - December 2002 (3 years 5 months)

Technical Lab Director for OTI lab in Amsterdam, the Netherlands. I founded the lab, growing and leading a small software development team as part of the larger IBM Java virtual machine and Eclipse teams.

The main product we developed in Amsterdam was IBM WebSphere Studio Device Developer (Eclipse-based IDE for embedded platforms such as PalmOS, J2ME, Nokia, PocketPC). WSDD was the first commercial product based on Eclipse, and many of our features have sculpted the shape of Eclipse.

During this time, I led my local team and collaborated with the J9 team in Ottawa, Canada, and Eclipse teams in Ottawa and Zurich, Switzerland. We also played a major role as a member of PECOS project, which was a European Community IST funded research together with ABB in Germany, University of Bern in Switzerland, and FZI in Germany.

A major focus of our lab was performance of resource-constrained embedded platforms, running a Java virtual machine, its libraries, and applications in total space of 1MB (that is not a typo, I really mean megabyte).

During this time I represented IBM on JSR075, the Java PDA profile, as part of the Java standardization process.

Research Staff Member at IBM T.J. Watson Research Center

May 1997 - August 1999 (2 years 4 months)

Research staff member. Performed R&D on Java analysis tools (such as JikesBT and Jax). I developed tooling to represent, edit, and emit the Java bytecode class file format to support full static analysis on Java applications. End result was reduced Jar size of up to 98% but also performance improvements due to peephole optimization, constant folding, class hierarchy collapsing, and method inlining.

I published various articles on the research that was performed and received 3 patent applications.

IT Analyst at Morgan Stanley

May 1994 - July 1997 (3 years 3 months)

Analyse emerging technologies and integrate into Morgan Stanley's IT portfolio. Developed a large user interface toolkit in C++ on UNIX and Windows for financial applications (MSTK).

I also evaluated the then-emerging technique called Java in 1995 time frame, and published the fourth book on Java as a result (Advanced Java). I converted major parts of MSTK to Java by developing an automated transformation tool (C2J) to help facilitate the transformation from one language to another.

The last half year of my time at Morgan Stanley was spent with the private banking division setting up a portfolio management system to run over the Internet (a very challenging proposition in those times).

Postdoc at IBM T.J. Watson Research Center

May 1992 - May 1994 (2 years 1 month)

Developed an IDE for the Oberon language, including a parser and model, where the goal was to use it as an education environment at colleges.

I also worked on my personal hobby, program visualization. I developed a tool called Hotwire, to visualize the inner workings of Smalltalk and C++ programs; the two major programming languages of the time. This work led to a publication at the Usenix C++ conference.

Researcher at SERC

1990 - 1992 (2 years)

PhD Research on object-oriented design, UI development, and visual programming techniques.

Publications

Livecode Python Teaching at Bank of America

Pycon 2013 March 17, 2013

Authors: Chris Laffra

Describes a custom teaching environment developed by a lazy programmer who did not want to write new training materials in PowerPoint and instead reused existing Sphinx docs by running them live in a cool training tool where you can try out Python in a live shell.

PyAlgoViz: Python Algorithm Visualization

Google AppEngine September 7, 2013

Authors: Chris Laffra

PyAlgoViz: Python Algorithm Visualization.

My coding weekend project. The main ingredients: Google AppEngine, Python, the Google ndb database API, sys.settrace, JSON, CodeMirror, jQuery, D3.js, HTML, Chrome debugger, and links to Wikipedia. The result lets you analyze, visualize, edit, and share classic Computer Science algorithms.

The hosted web application shows both the algorithm and visualizations for various sorting algorithms, balanced and non-balanced search trees, calculating Pi, searching, and common string operations. Through the visualizations, I hope to be able to help people more easily understand how various algorithms work internally.

This has been a great project to rekindle my previous experiences with JavaScript toolkits, writing full stack applications, and thinking hard about Computer Science algorithms. Aside from doing the full-stack application development itself, I wrote and uploaded a few dozen algorithms to show-case the environment. This was great fun.

AppEngine is a great environment to host something like this on. Everything just works as you expect it to. Initial hosting is free as well. Here's to hoping this site will not become too popular :-)