Résumé of John Stracke

Contact Info

- jstracke@gmail.com
- www.johnstracke.org

Current Status

I am employed as a Principal Software Engineer at Ocient. I have the following constraints:

- I cannot relocate out of the Greater Boston area.
- I will not take a contract position.
- I will not work under security clearance.
- I will not work for Meta, Amazon, or Alphabet.

Technical Summary

Primary Languages
: C/C++ (up through C++23), Java, Python, Common Lisp

Secondary Languages
: Scala, Haskell, Standard ML, Erlang, Perl, many others

Operating Systems
: Linux, Android, various commercial Unices

Disciplines
: Software design, algorithm design, performance analysis, network programming, protocol design, object-oriented design, design patterns, compiler design, Web-based applications, Unix/Linux systems programming, unit testing

Technologies
: TCP/IP, sockets, ANTLR, Bison/Yacc, Flex/Lex, Apache, XML, HTML, CSS, SQL, pthreads, gcc, gdb, Android, Unix, Linux, JUnit, PostgreSQL, CPPUnit, gtest, Git, Mercurial, Perforce, STL.

Education

- B.A. in Mathematics, Northwestern University. Graduated with departmental honors.
- M.S. in computer science, University of Massachusetts, Lowell. Coursework included programming language design, compiler construction, algorithms, and operating systems. My master's thesis was PackOS, a microkernel in which all IPC was IPv6, for which I wrote an IPv6 stack from scratch.

Employment history

Ocient, October 2022-present: Principal Software Engineer, Query Processing
: Ocient produces a large-scale OLAP database. I have worked on various new features and performance improvements. Recently, I have implemented random forests, sped up decision trees by about a factor of 60 (one run went from 2 hours to 2 minutes), and greatly extended the test suite for decision trees. Work is done in C++23. I mentor junior engineers, do design and code reviews, interview candidates, and run a reading group dedicated to improving less experienced engineers' understanding of modern C++.

Paradigm4, January-October 2022: Principal Software Developer

Paradigm4 produces an array-based database, targeted at life science applications. I worked on transaction improvements, RPM packaging, and a client for our REST API. Work was done in C++17.

Motional, November 2020 to January 2022: Principal Software Engineer

Motional develops software for self-driving cars. I started in the platform infrastructure team, working on middleware to let the various components communicate. Later, I helped design a new runtime framework, managing the various tasks that make up the software stack, and then moved to the team that was created to build that framework. Work was done in C++20, with some Python 3.

Smartsheet, April-November 2020: Senior SDE II

Designed and developed parts of a zero-management NoSQL database designed to fit into the existing Smartsheet ecosystem. As a senior member of the team, I also participated in design reviews, guided less senior engineers, and helped management pick technical direction. Work was done in Java and SQL. The product, a multitenant SaaS service, is implemented with microservices, based on OpenAPI and Micronaut, and runs on AWS Lambda. I applied compiler technologies to optimize execution of requests.

Google, April 2011 to April 2020: Senior Software Engineer

My last team at Google was working on a project called Longform, for writing and editing text with a stylus. The library was usable in various contexts; the first released product based on it is an enhancement to the ChromeOS virtual keyboard, to make ChromeOS tablets easier to use without a keyboard. Work was done in C++20 and compiled to Webassembly via emscripten.

Before that, I worked on QPX, which is the backend for Google Flights, as well as for many airline and third-party travel sites. QPX searches for flights and fares; it's a vast optimization problem. Some highlights of my work:

- Route restrictions for the new C++ scheduler microservice. Route restrictions are business logic above and beyond the publicly filed rules; I applied NFA theory, extended with arbitrary predicates, to make them perform efficiently.
- Diagnostics: a system for explaining why QPX didn't return the results a customer was expecting.
- Database security: locking down QPX's low-level queries for looking up fares and rules, so that they could be exposed to customers without letting them see each other's private data.
- In my time at Google, I received eight peer bonuses (suggested by peers, approved by management) and four spot bonuses (larger bonuses, from management), for achievements ranging from mentoring and collaboration to outstanding technical work.

Work was done in C++11/14 and Common Lisp.

ITA Software, 2008 to April 2011 (acquired by Google)

For a description of my work at ITA, see QPX, under Google, above.

Akamai, 2007: Senior Software Engineer

I worked on making one of Akamai's DNS servers multithreaded. Work was done in C++. I also ran an experiment with writing a DNS server in Erlang.

Endeca, 2004 to 2006: Principal Software Engineer

Endeca (since acquired by Oracle) sold an enterprise information access platform to help people search and analyze their information.

- I worked in the Engines group, maintaining and developing the search engine.
- Enhanced the data store to keep certain large data structures offline, to improve scalability.
- Worked on the design and implementation of XQuery support, including an optimization strategy to enable queries to be answered via the engine's data index, instead of by iterating.
- Work was done in C++, with extensive unit testing using CPPUnit.

Centive, 2001 to 2004: Principal Software Engineer

Centive sold a platform for calculating complex commission plans for companies with millions of transactions per month.

- Architected and implemented a compiler to translate the compensation plans (written in a SQL dialect) into Oracle PL/SQL and Microsoft T-SQL.
- Focused on improving the performance, both of the compiler itself and of the generated SQL.
- The result was around 100,000 times faster than Centive's previous version.
- Work was done in Java, on Windows.

eCal Corp., 1999 to 2001: Chief Scientist

eCal made Web-based calendaring software: a packaged server for the enterprise space, a hosted service for the consumer space.

- I was the chief architect; for example, I designed and implemented a Web application framework for the enterprise server. Pages were compiled to Perl on the fly and cached; this technique allowed us to exceed the performance budget of 140ms per page.
- I was also responsible for researching new technologies to keep ahead of the competition.
- I represented eCal in the IETF.
- I researched, and prototyped, instant messaging, when eCal wanted to diversify.

Netscape, 1996 to 1999: Principal Software Engineer

I came to Netscape as part of the acquisition of InSoft (see below).

- Integrated InSoft's conferencing architecture, which I had developed, with the H.323 standard.
- Developed an object-oriented widget library in JavaScript, for use with Netcaster.
- Represented Netscape in the IETF's WebDAV working group.

InSoft, 1993 to 1996: Senior Software Engineer

InSoft made enterprise videoconferencing and streaming media systems.

- Architected and implemented the peer-to-peer OpenDVE networking technology used in InSoft's Communique! videoconferencing program.
- Created the prototype of Netscape Media Server (a streaming media server).
- Work was done in C on seven different Unix platforms.

Analysis & Technology, Inc., 1992 to 1993: Software Engineer II

A&T was a naval contractor.

- Developed a Windows 3.1 application, in C++, for editing descriptions of underwater sound sources, for use in sonar training simulators.
- Held a Secret clearance.

National Science Center Foundation, 1991 to 1992: Software Engineer

The NSCF sold computer-aided instruction software for algebra.

- Created an X Windows front-end for the student database, on Unix, in a proprietary language.

US Patents

- 5892761: Method and apparatus for routing data in collaborative computing system
- 6047330: Virtual router discovery system

- 6167451: Multiple push protocol unifying system
- 8886644: User control of search filter bubble
- 8914229: Systems and methods for transferring navigation data
- 9448073: System and method for assessing road quality using data collected from a mobile device
- 9824685: Handsfree device with continuous keyword recognition

All Languages

I've used a *lot* of programming languages over the years. In chronological order, as best as I can remember, they are:

1. BASIC
2. 6809 machine language
3. Logo
4. Pascal
5. 6502 machine language
6. Prolog
7. Shell script
8. 68000 machine language
9. 8086 machine language
10. Forth
11. HyperCard/HyperTalk
12. Object Pascal
13. Learning Logic Language
14. Emacs Lisp
15. C
16. Common Lisp
17. Fortran
18. C++
19. Java
20. JavaScript
21. PHP
22. Perl
23. PostScript
24. SQL, PL/SQL
25. Python
26. Smalltalk
27. Standard ML
28. OCaml
29. Scheme
30. Haskell
31. Erlang
32. Ruby
33. Go
34. Scala
35. Rust
36. Lua

Of course, I'm not listing things like HTML, only Turing-complete languages. SQL makes the list only because I used PL/SQL, too.