Bennet Huber

bennet.huber@gmail.com
(215) 490-4297

1219 NW Richmond Beach Rd
Shoreline, WA 98177

Job Objective

To work as a lead engineer helping my team solve challenging and exciting problems.

Education

The Pennsylvania State University - May 2010
Bachelors of Science in Computer Science with a minor in Mathematics
GPA 3.55, Dean's list 7/8 semesters

Experience

Senior SDE	Amazon Inc	April 2021 - Present
SDE II	Seattle, WA	April 2016 - April 2021

Datapath Platform

Datapath is a serverless execution platform and programming language designed for high throughput, low latency execution of business logic organized in an SOA architecture. It powers a significant portion of the real-time backend business logic for amazon.com traffic, and is used by hundreds of internal teams and thousands of developers.

Datapath Responders (April 2016 - August 2016)

The Responders team at the time owned the language, execution logic, and real-time runtime fleets for executing customer logic. I was officially on Responders only a short time before being reorged, which I mostly spent learning the system and codebase. I was in oncall rotations from June 2016 - January 2017, and participated in several full website outage incidents. My main project was building a tool to allow customers to find which top level REST bindings could transitively invoke their business logic. The core codebase was later adapted to solve several other similar analysis problems.

Datapath LTCX (August 2016 - February 2019)

In summer 2016 the Datapath platform was approved to double their headcount; a team reorganization was necessary. Due to my personal interests, I was the first (and initially only) member of the newly formed Language, Tools, and Customer Experience (LTCX) team, responsible mainly for the developer experience of platform users. My main accomplishments included:

- Setting team priorities from the outset. As a former user of the product, I had insight into what problems our platform faced, and as the only member of the team I was left to my own devices on what to prioritize. I focused on easy improvements with large customer impact, such as making our web based tools more readable and self-documenting, fixing minor yet annoying bugs, etc. Several of my more ambitious ideas were eventually made into onboarding projects for new team members as they were hired, such as rewriting the language build CLI, and adding a concept of a service contract to the language.
- With a small amount of help from a senior engineer on another team, I successfully trained and onboarded six new team members in the course of the first six months, including our new manager. Of the six developers on that team, four have since been promoted to Senior SDEs at Amazon (including myself).
- As part of a data migration project, I added the capability for the entire platform to resolve customer business logic from non-Prod data sources. Before this, many features could only be tested by publishing user code to production, making some changes simply impossible to test safely. This required auditing, modifying, and testing access to the data store across three critical services, dozens of packages, and hundreds of thousands of lines of Java code.
- In early 2017, builds for the shared packages used throughout Datapath had grown to be four hours long, with two hours accounted for in a single top-level package my team owned. This was a major impediment to developer progress throughout the organization, as every Code Review and every deployment must do a full build of all consuming packages. The long build times were mostly due to inefficient unit tests. Since I was actively working on the slowest package, I profiled the unit tests and discovered a class of tests which could be refactored to be much faster - my initial prototype on a single test suite reduced it from 20 minutes to 20 seconds. I then assigned the junior engineer assisting me on my project the task of applying the same technique to all other relevant tests across the

package, and worked with sister teams to apply it to tests in packages they owned. I also found and fixed a performance in the database mock system we were using, and did several other more targeted optimizations. These combined to reduce our package's test runtime from 2 hours to 4.5 minutes, and the full build time for the shared packages from 4 hours to under 30 minutes.

- Developed a novel testing framework for testing equals()/hashcode() consistency across a complex type hierarchy. Changes to our core data model had introduced many bugs of this type over several years, resulting in numerous failed customer deployments and several outages for amazon.com. My test framework automatically identified all existing bugs, and ensured bugs introduced by new additions/modifications would be caught in the future.
- The Datapath Deployer is responsible for ingesting customer code changes and sending updates to all relevant server fleets to run the new code versions, with no downtime. If the deployer doesn't work, customers can't update the business logic for their services that power amazon.com's backend. The deployer had slowly evolved from a simple shell script to a complex, convoluted, and tightly coupled Java codebase with lots of implicit and shared mutable state. As a result, changes to the internal business logic would often break when deployed to production, despite good test coverage of individual components.
  I successfully refactored the internal deployer code, breaking the monolithic parts into decoupled pieces and removing much of the shared mutable state. This took several weeks of effort. Future changes were made with much fewer problems manifesting outside of unit/integration tests.
- Amazon does not have an official position of technical lead, but I filled that role over the lifetime of LTCX

Datapath Artifacts Team (February 2019 - April 2021)
In February 2019 Datapath teams were again reorganized, and I ended up on the newly formed Artifacts team, which owns the global language data store and deployment system that updates both it and the runtime fleets when customers deploy new versions of their code. As one of the senior developers on Artifacts, I spend a lot of time training and assisting newer team members in addition to my own work. Some of my accomplishments while on the Artifacts team included:

- In early 2019 we had significant new work planned for the deployment system. My manager encouraged me to develop a long term vision for what the deployment architecture should eventually look like, as by this time I had the most experience with it. Over the course of six weeks I developed a long term architectural vision document for a system that could successfully deliver our long term feature goals. This vision has guided much of my team's major project goals ever since. From a high level, it takes existing business logic implemented in two monolithic systems and splits them out into a microservice architecture. One of the microservices has already been fully developed (a telemetry database and UI built in native AWS), two others have begun development this year.
- Our customer code artifacts are stored in an internal, proprietary NoSQL document store. After many years of data written to this store, it needed what can best be described as a reindex operation run on it. It was getting so slow it was regularly timing out queries, leading to failed customer deployments. Unfortunately, the team's internal tooling to perform the "reindex" was insufficient for the amount of data we had stored. After partnering with the team owning the database to root cause our timeout issues, I helped them develop a custom reindexing script specific to our needs and a migration plan to safely apply it. After it had successfully run, the rate of transiently failing customer deployments significantly reduced.
- I successfully drove a campaign to eliminate failed customer deployments due to transient system errors as much as possible. These were caused by several subtle concurrency issues present in our distributed deployment system, some of which were several years old. I also lead customer communication on the issue.
- I identified several inefficiencies in the Datapath language build and test runners, leading to a 3x - 10x build time improvement for all our customers. One of the solutions I developed was later leveraged to provide similar speedups to a separate customer test tool.
- Traditional Service Oriented Architectures (SOAs) are normally constructed of sets of loosely coupled microservices that call each other in a dependency graph. These services are deployed separately to their own independent fleets. As a service owner, a common testing methodology is to deploy new updates to a single host (or small set of hosts) in the service fleet and monitor it for problems for some amount of time, before fully deploying to the rest of the fleet. This technique is known as "onebox testing", and is a last line of defense to catch any problems before they're fully deployed to production.

At the end of 2019, onebox test support was a top feature request from our biggest customers. Unfortunately, Datapath is not a traditional SOA - rather than deploying each “service” to separate fleets, each is deployed to every host across a single fleet. This makes even defining what “onebox testing” means difficult in Datapath. I was first tasked with requirements gathering and initial design for the Customer Onebox project, and eventually led the implementation as well. I identified several possible behaviors onebox testing could have for service owners in our architecture. I then presented these to senior engineers from our primary customers, to gather feedback on which attributes of testing were most important to them. After a few design iterations, I developed one which required only minor changes to our existing architecture and could be delivered in under a year. Over the course of the next 10 months, I led 1-3 junior engineers (depending on availability) in fully designing and implementing the necessary changes across the various systems involved, and collaborated with our fleet management team to deploy the new onebox testing hardware. The first beta customers were onboarded at the end of 2020. The project is still in beta testing.

- Extensively mentored 5 new hires
- Promoted to Senior Software Development Engineer (SDE III) 4/2021

SDE I	Amazon Inc Seattle, WA	August 2014 - April 2016

Featured Merchant Algorithm (FMA) Team

FMA owns the logic that picks which merchant offer is tied to Amazon’s "Add to Cart" button. It is a low latency, high throughput system - its logic is invoked millions of times per second worldwide with latency on the order of 10s of milliseconds. It is built on the Datapath platform (see other Amazon experience). While on FMA, some of my accomplishments included:

- Developed a shadow traffic automated testing mechanism to reduce developer time to run integration tests from several hours to a few minutes.
- Designed and implemented a new version of FMA’s "public" API vended to other Amazon teams. It was used for ~3 years before being deprecated and replaced by a new version to incorporate new product features (original is still in use in production).
- Improved stability of ETL system for gathering data analytics - went from regularly failing without notification (resulting in data loss) to full alarming and advanced retry logic
- Added latency-neutral incorporation of competitive pricing to FMA algorithm and metrics gathering through downstream services under significant time pressure (6 weeks).
- Promoted to SDE II November 2015

General Amazon Stats

- Mentored ~14 junior engineers/interns, 4 of whom are now senior engineers at Amazon
- Conducted ~150 interviews
- Amazon has an internal stackoverflow.com clone called Sage. For Sage points, I’m ranked in the top 150 in the
- company, or top 0.3% of active users. Most of my participation has been answering customer questions about Datapath.
- Code: ~1900 commits, ~2000 wiki edits, ~380 Code Reviews (CRs) submitted, ~1000 CRs read
- ~1000 tickets resolved
- Participated in oncall rotations for two tier-1 services (FMA, Datapath Responders)

Software Developer	Azavea Inc Philadelphia, PA	December 2010 - April 2014

Azavea is a medium sized consulting firm specializing in web/mobile applications related to GIS problems. I worked on teams of two to six people and projects lasting from several months to many years. I was a full-stack developer working on things such as UI design and implementation, application logic, spatial predictive modelling, distributed architecture design, application deployment, and database schema design. One of the primary projects I worked on is called HunchLab, a product designed for crime analysts that implements many cutting-edge crime prediction algorithms and statistical models.

Intern Cisco Systems Summers 2007 and 2008
San Jose, CA

Internal web development and server administration.

Relevant Skills

Proficient in Java development, designing and maintaining highly available distributed systems, microservices/SOA architectures, and mentoring junior engineers. I've worked on a large variety of platforms and technologies, including C#/.NET, Python, AWS, Javascript, IOS/Android, and too many others to list.

Activities

Recipient of the Lockheed Martin Engineering Scholars Award
Child Rearing
Past instructor for Girl Develop It Philly
Too many local hackathons to enumerate
Bicycled across America, Summer 2009

References

Available upon request