Rick Hutcheson

rick@oddlyaccurate.com
740.221.7449
oddlyaccurate.com

SUMMARY

Software engineer with 8 years of experience building large-scale applications and developer frameworks across multiple languages and software stacks, looking to transition into **low-level systems software**. In service of that goal, I'm interested in roles at any level.

WORK EXPERIENCE

Principal/Senior Staff/Senior Engineer — Udemy, Inc.

Feb 2015 to Nov 2022

Role: Worked as an engineer and tech lead on a variety of teams across the company. On our Platform team, I built distributed backend systems and frameworks for the rest of engineering. On our application and data teams, I built a variety of software systems, including a real-time user targeting engine and a customer data system. I worked across the stack, from web-facing systems to data-processing applications.

RELEVANT PROJECTS	*SKILLS*
Task-orchestration system Designed and led development of a set of coordinating daemons that managed Udemy's background-job cluster, responsible for running hundreds of millions of tasks per day. The system balanced task-queue throughput against resource usage and system load across dozens of daily releases. Reduced average waiting time for an individual task by 3-4x and increased per-machine utilization by more than 50%, compared to the previous static cluster configuration.	*AMQP, Distributed systems, Python, RabbitMQ, Sockets*
User-targeting platform Architected and led development of a realtime user-targeting system that allowed marketing to assign customers to experiments, discounts, or messaging flows based on dynamically-updated customer attributes and behaviors. Assignment durations were calculated on a per-user basis, enabling "behavior-based" flows (e.g. time-limited discounts). Extended the system to offline use cases with APIs for access in multiple programming languages. While this system was initially scoped for use only by Udemy's marketing department, its utility and flexible design led to its adoption by multiple other departments.	*Kafka, MySQL, Python, Protobuf, Redis, Scala, Spark*
Developer Frameworks Developed a series of internal developer frameworks to improve performance and ease development, including:	
⇒ An HTTP-caching framework, allowing developers to cache any page to a client browser or a reverse proxy, with automatic invalidation based on a hash of the user's state. By eliminating redundant requests, the framework allowed the webserver cluster to handle the same traffic with 10-20% fewer servers.	*Django, HTTP, Javascript, nginx, Varnish*
⇒ Declarative frameworks for caching database queries or routing them to read-replicas based on the set of joins used. Reduced total number of database queries by 30%, and queries against the main database by 50%.	*Django, MySQL, ProxySQL, memcached, nginx*
⇒ A framework to convert any long-running "batch-processing" task into a number of smaller, more performant parallel tasks, whose results could be combined on completion using a "map-reduce" pattern. The parallelization strategy could be customized based on the nature of the job and its side-effects.	*Celery, Python, Redis*

LOW-LEVEL PROJECTS

Linux Kernel Modules

	SKILLS
Wii Nunchuk Kernel Driver Implemented a kernel joystick driver for the Nintendo Wii Nunchuk; communicates with the hardware via polling and I^2C commands.	*C, DeviceTree, I^2C, Kernel Locking, SMBus*
Serial Communication Driver Implemented a driver for interrupt-driven, two way communication between two systems connected via a serial port. It served as a testbed for learning the Linux DMA interfaces.	*Direct-Memory Access, Programmed I/O, Serial communication, UART*

Virtual Bus & In-Memory Character Device Drivers[1] Wrote a kernel module that creates a virtual bus, in the style of the platform-device subsystem. When other modules register their own (virtual) devices, the virtual bus can create / destroy instances of these devices via sysfs. Implemented a virtual "in-memory storage" character device for this bus, based on the scull driver in *Linux Device Drivers*. *Linux Device Model, sysfs*

System Software & C Programming

hoc: Interpreted Programming Language Implemented an interpreted programming language `hoc`[2], based on the version described by Kernighan & Pike. Wrote an accompanying set of blog posts[3] to describe its architecture and implementation. *BATS, C, Make, Yacc/Bison*

UNIX System Tools Implemented C versions of classic UNIX utilities described in *Software Tools in Pascal*, including a version of the original `ed` editor. *C, Make*

OTHER SKILLS

I have commercial experience with Java, Kotlin, PHP, C#/.NET, git, and subversion. I am comfortable working in Linux, macOS, or other UNIX based environments. I have developed software using agile practices, with both long-term and short-term sprints.

EDUCATION

B.S. Computer Science & Engineering *(cum laude)* The Ohio State University *2009 to 2014*

GPA: 3.66/4.0

Dean's List (GPA $\geq$ 3.5) for 2012-2014

[1] `https://git.sr.ht/~rickhutcheson/virtual-devices`

[2] `https://git.sr.ht/~rickhutcheson/hoc`

[3] `https://oddlyaccurate.com/projects/hoc`