BigDataX 2020: Operating Systems and Architecture Projects

Kyle C. Hale, Illinois Institute of Technology

Operating Systems and Architecture (Hale)

The Laboratory for High-Performance, Experimental Systems and Architecture (HExSA Lab) is generally focused on building, evaluating, and modeling novel and experimental computer systems. The below projects will generally involve the exploration of *specialization* both at the OS level and at the level of hardware to support high-performance and emerging computing applications.

Enhancing High-Level Languages with OS Support *Hybrid runtimes* (HRTs) are mashups of an extremely light-weight OS kernel (called an Aerokernel) and a runtime system [1]. The idea here is to eschew overheads imposed at the system call layer by general-purpose kernels, and to prevent the kernel from imposing abstractions on the runtime system. Here, the runtime operates at the *same privilege* as a typical kernel, and therefore has access to privileged hardware and ultimate control of the abstractions built on top of the hardware. Projects involving HRTs will primarily involve bringing other languages and parallel runtimes to the HRT model, such as Charm++, SWARM, or the Julia language runtime.

Pushing Serverless Computing to the Extreme The *serverless* model allows users to write code without worrying about the underlying platform on which their code executes. Often this code runs in reaction to external events (e.g. a sensor reading or a database update). Today's serverless platforms run this code on pre-provisioned containers, usually running in a virtual machine on the providers' infrastructure. However, the mechanisms that support these code invocations are often costly, rendering short-running (e.g. microsecond-scale) serverless functions (or lambdas if you like) untenable. This project involves developing novel virtualization and programming language technology to support these short-running serverless functions.

Interweaving Software and Hardware Layers (with Northwestern University) We imagine that future specialized system software might look radically different from what we have today, where high-level tools customize a software stack by combining components of what we currently designate as different *layers* in novel ways. This is an exploratory project that involves using compiler and OS techniques to *interweave* different components of the hardware/software stack. Hybrid runtimes are one example of interweaving, where the OS and runtime are *interwoven*. Other such *interweavings* are possible, e.g. application/OS, compiler/scheduler. The latter example we call *blending*, where the compiler might inject OS-level code into the application. There are several sub-projects within the interweaving project which span the software/hardware stack, but some salient ones include exploring new address translation techniques, incorporating system software components into FPGAs, blend calls in the compiler back-end, exploring the dynamics of process address spaces, programmable cache coherence engines, and unwinding of interwoven code for debugging purposes. Projects in this area would likely occur with significant interaction with Hale's collaborators at Northwestern University[1].

Modeling the Performance of Multi-kernel Systems Multi-kernels are a new systems paradigm where two operating systems run side-by-side on a single system. A program's execution is *split* between these two OSes to optimize for various phases of behavior. However, such systems often involve developing an entirely new operating system, which is a daunting task. Before pursuing this, developers should be able to reason about the performance of their program in this model. This project would involve developing and empirically validating analytical performance models which developers can use to this effect. Students working on this project will ideally have a solid background in math and statistics, and ideally will have some background in modeling probabilistic processes (e.g. markov models, queueing theory, arrival processes).

[1]`https://interweaving.org`

Designing AI-assisted Smart Hearing Aids (with Northwestern University) Modern hearing aids are typically built using low-power digital signal processors (DSPs). The basic idea is to separate audio coming into the aid into different frequency bands, filter out some of the bands, and boost others. While the advent of digital systems improved the situation over analog circuits, users of hearing aids still suffer from the "cocktail party problem," where they have difficulty distinguishing audio sources (speakers) in a noisy (or multi-speaker) environment. *Source separation* is a technique which can be applied to audio streams to tackle this problem [2, 3]. Different sources are extracted from the signal, often using machine learning algorithms. The most successful of these techniques involves deep neural nets (DNNs). This project involves training such a source separation DNN, and building a domain-specific architecture (a custom chip) the takes in an audio stream, uses the DNN to perform inference and separation, then filters out (or dampens) unwanted components of the audio stream, and boosts others [4]. This project involves a collaboration with the Interactive Audio Lab at Northwestern University that focuses on machine learning for audio and speech. The HExSA lab component of this project would mostly be centered on developing a hardware designs (first using FPGAs) to implement existing DNN and source separation algorithms. Students working on this project will ideally have some experience in computer architecture and familiarity with HDLs like Verilog or VHDL.

Hybrid Runtimes for Compiled Dataflows: Database/OS Co-Design (with IIT's DBGroup) This project would involve designing a new database-oriented virtual machine intended to execute atop special purpose kernels. The idea here is that database queries are compiled down to a virtual ISA (vISA), which is interpreted (or dynamically compiled) by a virtual machine (VM). Current database engines compile queries either directly to existing low-level languages, architecture-specific machine code, or byte-code for a high-level language. The former two leave little room for flexibility, and the latter reduces performance significantly. We would build a virtual machine which is *specialized* for query execution. The vISA it exposed would be very simple, and its back-end implementation could take direct advantage of running inside of a specialized kernel environment. Students working on this project should have some basic knowledge of database organization and systems background.

Processes as Dynamical Systems Program execution (i.e. a running process) can be seen as a dynamical system, where the state (both visible and invisible) of the machine during execution comprises the state space, and the program's specification determines evolution in time. While such framing is not traditional in computer systems, we believe it has the potential to reveal properties of programs which allow them to be represented in compact ways amenable to searching, classification, characterization, and prediction. The goal of this project is to apply principles of physics to the study of computer systems and discover new ways to represent running programs compactly based on incomplete run-time information. Students will explore the application of such ideas to areas like malware classification, process scheduling, and resource prediction. Students working on this project will ideally have some background in mathematical physics and computer systems.

While these projects are ambitious ones, we expect that with the assistance of Dr. Hale and his students, who have an established track record in systems research, a diligent undergraduate student with systems background could make considerable headway over the 10-week period.

Mentor

Dr. **Kyle C. Hale** (the mentor) is an Assistant Professor of Computer Science at Illinois Institute of Technology. He received his Ph.D. in 2016 from Northwestern University. His research generally is in experimental computer systems and lies at the intersection of operating systems, high-performance parallel computing, and computer architecture. He developed the *hybrid runtime* model for parallel runtimes and has worked on major open-source research codebases, including the Nautilus Aerokernel (for which he is lead developer), the Philix OS toolkit for Intel Xeon Phis, and the Palacios Virtual Machine Monitor.

References

[1] Kyle C. Hale and Peter A. Dinda. A case for transforming parallel runtimes into operating system kernels. In *Proceedings of the 24th ACM Symposium on High-performance Parallel and Distributed Computing (HPDC 2015)*, June 2015.

[2] Prem Seetharaman, Gordon Wichern, Jonathan Le Roux, and Bryan Pardo. Bootstrapping single-channel source separation via unsupervised spatial clustering on stereo mixtures, 2018.

[3] Masahiro Sunohara, Chiho Haruta, and Nobutaka Ono. Low-latency real-time blind source separation with binaural directional hearing aids. In *Proceedings of the* 1^{st} *International Workshop on Challenges in Hearing Assistive Technology*, CHAT-2017, August 2017.

[4] Oliver Townend, Jens Brehm Nielsen, and Jesper Ramsgaard. Real-life applications of machine learning in hearing aids. Accessed: 2018-11-19.