BigDataX 2017: Operating Systems and Architecture Projects

Kyle C. Hale, Illinois Institute of Technology

Operating Systems and Architecture (Hale)

High-performance parallel applications and the runtimes that support them often require very fine-grained control over the execution environment of the system. When raw performance is the goal, even small overheads that arise from the operating system or other system software (over which the runtime/app has no control) can have considerable effects on performance. For example, general-purpose OS kernels must be all things to all apps, and so must necessarily trade off performance for flexibility.

These projects will involve the exploration of *specialization* both at the OS level and at the level of computer architecture to support high-performance computing environments. This work will extend existing work on *hybrid runtimes* [2, 4, 3] and other specialized OS concepts.

Hybrid Runtimes *Hybrid runtimes* (HRTs) are mashups of an extremely light-weight OS kernel (called an Aerokernel) and a runtime system [2]. 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 target bringing other parallel runtimes to the HRT model, such as UPC, Charm++, or the Julia language runtime.

Specialized OS-based Microservices Currently, specialized OSes [1, 8] are deployed in a very inflexible manner. For example, with HRTs, the application and/or runtime is compiled together *with* the Aerokernel to create an HRT. Other techniques like Unikernels [7, 6, 5] also rely on similar compile-time techniques. This project will involve making these specialized OS and runtime (OS/R) environments more easily deployable, for example using existing tools like Docker.

Debugging Hybrid Runtimes HRTs currently involve great difficulty when debugging. This is partly due to the kernel mode nature of the HRT's operation. While there are existing tools like kGDB that allow programmers to remotely debug a kernel on a separate system, they do not account for the involvement of the runtime system, and they do not present a generic interface that can be adopted by several runtimes. The goal of this project would be to develop such a debugging tool for HRTs.

Compiler-generated OS Code Current parallel programs and runtimes are compiled assuming that they will be running at Linux user-level. When this assumption is broken (e.g. when they are running as a kernel), there is a lost opportunity for performance and functionality. This project would involve adapting a code generator that *assumes* low-level, privileged access and thus leverages kernel functionality and the available, privileged devices.

Programmable NoC Architectures Networks-on-Chip (NoCs) are the substrate upon which chips in a multicore system communicate. Current NoC designs are *static* in that their properties cannot be changed by software at runtime. This project would involve working in simulation to show how programmable NoCs (where the NoC's design parameters could be changed at runtime) could benefit the system software (primarily the OS).

Programmable Cache Coherence Engines FPGAs (Field Programmable Gate Arrays) are chips that can be *reprogrammed* to implement different logic functions. In the limit, these FPGAs can instantiate different CPU designs. They are currently enjoying a surge in popularity due to the gradual improvement in their programming interface. A multicore chip instantiated on an FPGA has much more flexibility in terms of experimentation than one that is "set in stone" on an application specific integrated circuit (ASIC). One thing that we can change at this level is a multicore chip's cache coherence protocol, which is used to transparently synchronize state between CPU cores. Current protocols are not "programmable" in the sense that they cannot be modified by software. However, some applications with well-known memory access patterns might benefit from augmented cache coherence protocols. This project would involve studying the access patterns of applications to identify cases in which the existing protocols are suboptimal.

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.

1 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] Engler, D. R., Kaashoek, M. F., and O'Toole, Jr., J. Exokernel: An operating system architecture for application-level resource management. In *Proceedings of the* 15th *ACM Symposium on Operating Systems Principles (SOSP 1995)* (Dec. 1995), pp. 251–266.

[2] Hale, K. C., and Dinda, P. A. 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).

[3] Hale, K. C., and Dinda, P. A. Automatic hybridization of runtime systems. In *Proceedings of the* 25th *ACM International Symposium on High-Performance Parallel and Distributed Computing* (June 2016), pp. 137–140.

[4] Hale, K. C., and Dinda, P. A. Enabling hybrid parallel runtimes through kernel and virtualization support. In *Proceedings of the* 12th *ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2016)* (Apr. 2016), pp. 161–175.

[5] Kantee, A. *The Design and Implementation of the Anykernel and Rump Kernels.* PhD thesis, Helsinki, Finland, 2012.

[6] Lankes, S., Pickartz, S., and Breitbart, J. HermitCore: A unikernel for extreme scale computing. In *Proceedings of the* 6th *International Workshop on Runtime and Operating Systems for Supercomputers (ROSS 2016)* (June 2016).

[7] Madhavapeddy, A., Mortier, R., Rotsos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., and Crowcroft, J. Unikernels: Library operating systems for the cloud. In *Proceedings of the* 18th *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2013)* (Mar. 2013), pp. 461–472.

[8] Peter, S., and Anderson, T. Arrakis: A case for the end of the empire. In *Proceedsings of the* 14th *Workshop on Hot Topics in Operating Systems (HotOS 2013)* (May 2013), pp. 26:1–26:7.