I want to pursue a Ph.D in Computer Science, and my future career goal is to become a professor. During the past two years I have worked closely with Prof. Orran Krieger (Boston University), Dr. Raja Santhanam (Boston University), Prof. Rodrigo Fonseca (Brown), and Prof. Peter Desnoyers (Northeastern) at the Massachusetts Open Cloud and have had the chance to lead or co-lead several research projects. My experience working on a novel network architecture and building an advanced performance diagnosis framework has inspired me to focus on networking and systems research for my Ph.D. For example, I am interested in exploring different problems in advanced networking such as congestion control, routing, and network availability for the cloud. In fact, networks were originally designed to be resilient, but today’s networks often take high performance into serious consideration due to the many critical time-sensitive applications people want to run on the cloud. This begs the question about how should we architect or design our cloud network infrastructure. The experience of working on advanced performance diagnosis tools and methods made me realize how important, yet difficult debugging can be to today’s microservices-based distributed applications. I am interested in building frameworks and tools to help people understand complex distributed systems and diagnose problems.

I intend to perform systems research and work on problems that are both intellectually challenging and practical. Also, I strongly believe that researchers should conduct their research in the public domain and prototype their research in real systems. This is why I strive to test my hypotheses in production systems and contribute my work back to the open-source community. I now briefly describe two of the key projects I have worked on and my contributions to them: Advanced Networking: Many of today’s applications (e.g., web services or cloud systems) running in datacenters impose diverse requirements on the network. For example, a distributed data-parallel framework like MapReduce can take advantage of different network scheduling policies in different phases to greatly improve the performance [2]. However, today’s datacenter network is always developed in a “one-size-fits-for-all” fashion [1]. This design results in two intertwined issues. First, protocols implemented for different applications are not designed to co-exist with one another, which limits the options available to tenants. Second, even if there were multiple networking options, there is no uniform way for tenants (and their applications) to select different options. To help, I worked with a senior Ph.D student to co-lead FLEXNET [5]. FLEXNET aims to explore the capability of providing differentiated network resources in cloud datacenters to various tenants and applications so that applications will be able to select networking options that best suit their needs.

We architected and prototyped FLEXNET, which decouples datacenter networks into pods, inter-pod networks, and Edge-of-Pod switches instead of the traditional unified design. Pods can be extended to different scales and are usually less over-provisioned. Inter-pod networks are network resource offerings that could provide specific protocols to custom-tailor the needs for cross-pod traffic. Within FLEXNET, pods and inter-pod networks are connected via EoP switches. This enables FLEXNET to schedule network resources in a decentralized manner. I implemented the EoP switch using Open Virtual Switch in merchant silicon and led two key experiments to evaluate our design. I also worked with the other student and implemented a prototype SDN controller in Ryu (2900 lines of code). We showed that FLEXNET is sufficiently flexible to better support various kinds of applications running on the cloud for different tenants while it also enables niche protocols comprised of individual network technologies. I surveyed conference publications from NSDI and SIGCOMM for the last 3-5 years and derived the key insights about the two intertwined issues. To acquire the background needed for this project, I also took a Ph.D level advanced networking course at Brown. This required me putting in the additional extra effort of going back and forth between Providence and Boston twice a week to catch every lecture.

Performance Debugging: In the past year, I have become interested in an emerging distributed systems domain: performance diagnosis for distributed systems. The growing complexity of applications running within datacenters has made performance debugging crucial, yet extremely difficult. To be more specific, it is hard to know a priori where, within what stack level (e.g., application level, network stack level or kernel level), and at what granularity to add instrumentation a priori to help diagnose future problems. It is also difficult to predict what information needs to be collected at each instrumentation point to help diagnose far-off problems. I have been working on PYTHON, which is a cross-layer just-in-time instrumentation framework that aims to automate such daunting processes by dynamically deciding where to expose additional instrumentation (log points or perf. counters) and what additional instrumentation is needed to identify unexpected behaviors.

A key enabler for PYTHON is the recent research and industry effort in end-to-end tracing [5]. End-to-end tracing captures the workflow of individual requests within and among components in the format of directed acyclic graphs (DAGs), wherein nodes are records of enabled instrumentation points accessed, and edges are causal dependencies among them. The skeletons include all the causally-related activities to allow developers to understand and inspect the application. Our proposed framework comprises of two major components: an advanced tracing instrumentation library that could enable and disable instrumentation points at different layers at runtime, and a complete pipeline that can reconstruct, analyze trace data, and identify problematic areas in near real-time. We are starting with the default tracing infrastructure for OPENSTACK (OSPROFILER) and CEPH. Currently I am upgrading OSPROFILER to use DAGs to represent and programmatically abstract the causal relationships in the application. I am also working on
features to allow developers to enable and disable instrumentation points at runtime. In the meantime, I co-lead the group to build the pipeline which tackles research questions like how can we consume the trace data and root-cause the unexpected behaviors in near real-time. Both OpenStack and Ceph are running in-production systems in MOC, and our findings will subsequently be contributed back to the community.

While working on FlexNet and Pythia, a research question came to my mind: in order to help developers diagnose performance problems for distributed applications, should we design frameworks that cross the boundary between the infrastructure (e.g. network, hypervisor, kernel) and application level? There is research where developers can write queries to help diagnose problems at different layers: Pivot Tracing [3] allows developers to write queries that are translated into the instrumentation needed to answer performance problems in Hadoop. PathQuery [4] uses the same methodology for debugging networking problems. There has been work that applies similar methodology for network monitoring [7]. I believe a framework that use this methodology cross-layers can make debugging for distributed applications much easier. I intend to research the answer to this question with my peers in the future.

References


