Research Statement

Johann-Alexander Hauswald

My research interests lie in system design with an emphasis on understanding the performance of emerging applications. Specifically, I research techniques, tools, and runtime systems to optimize the placement of the computation and the implications on the hardware and software design.

What once defined the Edge as wimpy cloud reliant devices is now made up of mobile supercomputers, wearables, and smart speakers generating massive amounts of data that need computing housed in Warehouse Scale Computers (WSC). Intelligent web services represent an emerging class of these services that leverage machine learning (ML) algorithms on the critical path of a query. These have seen an explosion of usage in recent years because of the very high accuracies achieved and the resulting superior end-user experience. However, the amount of computation required for these applications is still significantly larger than traditional web services and the demand for experiences that have ML on the critical path of a query continues to increase.

My dissertation work focuses on intelligent web services and their impact on existing WSC infrastructures. Specifically, building end-to-end workloads, studying the performance bottlenecks on real systems, and designing system architectures optimized for these workloads. To that end, I developed an open-source virtual assistant Sirius [2] exposing the gap between existing computing infrastructures and what's needed for an application with ML on the critical path of a query. In follow-up work, I show GPUs are a cost-effective cloud accelerator when optimizing for the algorithmic common case, namely deep neural networks [1]. Finally, I investigate how computation can be partitioned between the edge and cloud [5, 6] (latter work with a colleague), researching the tradeoffs that come when operating in a constrained environment.

Toward the end of my dissertation, I pursued commercializing my work building virtual assistants at Clinc, an AI company I founded with my research advisors and another graduate student. I was part of the core engineering team which designed novel Natural Language Processing (NLP) techniques to enable free-form dialogue with a virtual assistant, specifically designing the microservice and infrastructure architecture of the end-to-end system. After the successful completion of the core system, I worked closely with our customers in bringing our technology to market, adapting it to specific requirements and uses, and understanding the unique challenges of deploying software on-prem at large enterprises. During my time at Clinc, I co-authored 2 papers investigating the importance of high-quality training data in building robust text classification and slot-value pairing NLP models for virtual assistants [7, 8]. I also coauthored Clinc's accepted NSF SBIR Phase I and II Grants and I am a co-inventor of 9 granted patent applications in the field of virtual assistant design and NLP technologies.

At the time of publishing my work, our community was still focused on traditional web services and lacked the necessary tooling to begin their investigations. My findings, paired with the open-source artifacts made available, spurred exciting research into these applications in academia and industry. Indeed, the extremely strong interest from industry led to the founding of Clinc. My work was honored with the prestigious Micro Top Picks award [3], selected as an invited paper in the ACM Transactions on Computer Systems [4], built an active open-source community, has been used in numerous undergraduate research projects, and has been cited in many academic and industry papers.

Dissertation Research - System Design for Intelligent Web Services

The proliferation of AI applications that use highly accurate ML algorithms to make inferences on end-user speech, natural language and images has enabled a new suite of intelligent web service driven experiences. However, the amount of compute needed for these web services is orders of magnitude larger than traditional cloud applications and building increasingly larger WSCs to meet the demand is not feasible. Systems architects are faced with the challenge of coming up with novel system architectures to accommodate this increase in demand. My dissertation investigates the question of how to redesign WSCs for a future reliant on intelligent web services.

In my dissertation research, I design Sirius [2], an end-to-end open-source virtual assistant decomposed into a benchmark suite (Sirius Suite) used to study the computational bottlenecks of an application with ML on the critical path of a query. The open-source workload and the benchmark suite are major technical contributions to the community because until now the knowledge of how to build a virtual assistant, an application that has significantly grown in usage in recent years, was limited to a small number of enterprises. One key insight of this work shows that the amount of compute needed to add to existing datacenter infrastructures if we were to replace traditional web-search queries with queries from a virtual assistant would increase the size of current infrastructures by up to 168 times their existing size. I perform a real system analysis to understand the computational bottlenecks of Sirius Suite on a CPU server and show that, even if the software were to be perfectly optimized for the CPU, the performance improvement would not be significant enough. Consequently, I investigate mapping the 7 computational bottlenecks of the benchmark suite to 4 commercial off the shelf accelerators to quantify the potential of accelerator based WSC architectures. The performance improvements of each accelerator enables the study of the performance vs. cost (power consumption and purchase price) of each accelerator option and proposes system architectures to improve the Total Cost of Ownership (TCO). Sirius concludes that GPU- and FPGA-based WSC architectures mitigate unsustainable WSC scaling.

AI applications are ubiquitous and widely deployed in large part because of the high accuracies achieved by deep neural networks (DNNs). However, this increase in accuracy comes at a very high computational cost: 1) DNNs require large amounts of labelled data to train the models and 2) the networks have to be "deep" (i.e., with many parameters) to represent the learned data making inference computationally intensive. In my work, which focuses on the inference phase of the DNN, I exploit the algorithmic homogeneity across applications (i.e., DNNs can be used to solve a range of tasks) and widespread industry adoption to design a single, highly optimized DNN-as-a-service engine using GPUs as the accelerator of choice. I investigate the key bottlenecks inhibiting peak utilization of a multi GPU-based server in a real system analysis. To study this, I built and open-sourced 2 key artifacts: DjiNN, the highly optimized web service for DNN applications, and Tonic Suite: a suite of 7 DNN based applications representative of emerging workloads spanning computer vision, speech recognition, and natural language processing applications [1]. I identify several performance bottlenecks (low utilization and bandwidth to the GPU) in the service and design strategies to mitigate them, achieving high throughput and multi-GPU scalability without diminishing query latency. One key insight of this work is that DNNs do not benefit the same from acceleration because of the different neural network architectures across applications (i.e., computational characteristics). I show batching computation to the accelerator, storing the models in GPU memory, and running multiple DNN services improves the utilization and increases the overall throughput of the system. With the design of a single, highly-optimized DNN service, my work concludes that GPU-based server designs reduce the overall TCO of a WSC.

I am also interested in unlocking the potential of efficiently partitioning compute in the edge-cloud computing ecosystem as edge devices and cellular networks become more powerful. In addition to research focusing on WSCs, I have published [5] and collaborated [6] to investigate how to partition ML compute between the edge and cloud. The key question we pose in our recent work is whether completely offloading computation to the cloud is optimal (minimal latency and power consumption) for the edge device. We show that DNNs can be either fully executed on the edge device, the cloud or there exists a partition point within a neural network that minimizes the latency and power consumption of the mobile device. We show this also improves datacenter throughput. We design a runtime system, Neurosurgeon, that makes decisions on how to execute the neural network using its architecture and a prediction model for each layer type that statically estimates the amount of computation needed.

Future Research Directions

Below I outline some areas of future work where I am excited to continue my research in system design.

Defragmenting the Edge-Cloud Ecosystem

The number of edge devices that rely on cloud connectivity for most of their processing has continued to increase in recent years. Cloud providers are forced to design increasingly specialized accelerators or build massive cloud infrastructures to keep pace. It is still an open question how to take advantage of computing at the edge to mitigate the need to continue scaling WSC infrastructures. One of the challenges in this regard recently described by Facebook is "there is no standard mobile SoC to optimize for" and this leads to large variability in end-user experience [1]. This creates a *fragmented compute ecosystem*. I believe we are leaving future compute on the table and will have to continue building larger cloud infrastructures if we do not research novel ways to reduce this variability and design systems for efficient edge computation. Below I lay out specific questions in this research direction:

- Device Taxonomy: how should the landscape of devices be taxonomized with the goal of understanding if and where we can take advantage of spare cycles?
- Secure: what techniques must be developed to harness this ecosystem in a privacy conscious approach allowing secure, anonymized compute offload and incentive between independent devices?
- Cost-effective: what type of applications and what fraction must be offloaded to the edge for it to be financially beneficial for cloud providers?

Device Taxonomy - The naive approach would be to taxonomize by device type, however it's not clear this is the best approach because it precludes compute sharing between heterogeneous devices. We can study the compute available as a whole taxonomizing by compute potential, idle cycles, distance to next best compute, and connection reliability to inform scheduling policies between different devices. Part of this taxonomy is also performance benchmarking that I anticipate can be accomplished using a combination of static and dynamic information. Recent work provides a starting point for the dynamic measurements required for ML applications (high offload potential from the cloud to edge) to run predictable on-device benchmarks [2]. This will provide an understanding of the device performance given for a particular SoC there can be large variations (e.g., device type and aging, connectivity strength, impact of background applications, unknown hardware defects, etc).

Secure - the proliferation of microservice architectures using lightweight, virtualization techniques like Docker and Kubernetes make it possible to separate services into smaller units of compute that can be mapped to heterogeneous computing hosts. These techniques are already widely used in cloud infrastructures because they are secure and allow flexible mapping of compute needs to the underlying hardware resources. I anticipate a similar approach can be used to allocate compute in the edge-cloud ecosystem. Additionally there exist secure, anonymized device-to-device communication protocols that use Bluetooth Low Energy (BLE) to exchange information between mobile devices. This approach can be extended to develop a federated edge computing protocol.

Cost-effective - in most cases, the cloud is on the critical path of edge processing. Communication with the cloud must become a second class design factor to reduce the need to continue building massive cloud infrastructures. I do foresee the partitioning of compute to span the edge-cloud spectrum because, as some of my prior research has shown, the environment changes frequently meaning the partitioning decision needs to be constantly reevaluated. While edge computing does not come for free, there may be spare cycles that can be leveraged instead of communicating with the cloud. I envision creating a supply-and-demand economy where devices are incentivized to sign up in exchange for offsetting cloud computing costs using secure, decentralized, and anonymous payment methods. The open question is at what point does edge computing become financially more attractive than the cloud given the cloud is already very aggressively optimized?

[1] See C-J. Wu et al. *Machine Learning at Facebook: Understanding Inference at the Edge*, HPCA'19.
[2] See V. J. Reddi et al. *Mlperf Inference Benchmark*, ISCA'20.

Cross-Layer Support for the ML Lifecycle

Sirius and DjiNN take for granted an important insight in deploying ML applications: a single ML web service has significant software management overhead that is greater than traditional web services. There are a host of challenges from a systems perspective that emerge when taking a model to production that I have experienced first-hand in industry. Once the model is live, there is a continuous feedback loop of curating more training data, validating the model, and pushing it to production; a serial and latent process. Indeed, industry has recently built a number of tools (Google's AirFlow, Netflix's MetaFlow and Databricks' MLFlow to name a few) to manage the ML model lifecycle commonly referred to as "MLOps". It's an open question what the proper approach should be moving forward as models grow and our reliance on accurate, unbiased, and low-latency models continues to grow. Given this reliance, I anticipate this to become an active area of systems research.

First, the MLOps workflow of train-test-deploy is a new workflow where the steps are represented as a pipeline or a Directed Acyclic Graph (DAG). Similar to traditional CI/CD pipelines, the responsibility is on the programmer to design the steps. I foresee a number of inefficiencies emerge when giving the programmer the responsibility of designing the MLOps pipeline. I plan to investigate the following improvements: automated validation of the DAG, efficient scheduling tasks to large clusters, and job collocation of DAG nodes to the underlying hardware resources to improve utilization.

Second, once models are live, the typical process is to use a feedback loop to continue training the model with live data to improve accuracy. The burden is on the developer to ascertain whether the new model is "better" than the previous. Indeed, Amazon recently published a tool to aid users of its AI platform with understanding how to define "better" given accuracy isn't always the best metric [3]. By treating the model as a blackbox, developers build regression suites to negate model drift. Given limited information about how the model behaves, developers end up with very large regression suites. This is a considerable bottleneck in deploying the model I witnessed firsthand during my time in industry. I would like to investigate systems to validate ML models from a blackbox and whitebox approach that are scalable and accurate which improve on the status quo of having to grow the data the longer the model is deployed.

Third, as models are improved through a combination of more training data, changing learning parameters, and tuning the model architecture, the computational footprint of the model typically grows (some work lately in Neural Architecture Search (NAS) improves this). This makes it infeasible to run on older devices while still meeting a strict SLA. MLOps tools combined with recent advanced compiler frameworks like TVM [4] enable an interesting analysis of treating models as cross-compiled binaries for target computing platforms. I am interested in studying the tradeoffs between accuracy, model bias, and efficiency to design automated tools to reduce the burden on the programmer in having to pick a certain optimization target as part of the MLOps pipeline.

I expect tackling the challenges of the ML lifecycle will create many opportunities to collaborate with experts in programming languages, databases, algorithms, and distributed systems.

References

[1] J. Hauswald, Y. Kang, M. A. Laurenzano, Q. Chen, C. Li, T. Mudge, R. G. Dreslinski, J. Mars, and L. Tang. Djinn and tonic: Dnn as a service and its implications for future warehouse scale computers. ISCA'15.

[2] J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, and J. Mars. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. ASPLOS'15.

[3] See Rauschmayr et al. *Amazon SageMaker Debugger: A System for Real-Time Insights Into Machine Learning Model Training*, MLSys'21.

[4] See Chen et al. *TVM: An Automated End-to-End Optimizing Compiler for Deep Learning*, OSDI'18.

[3] J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, and J. Mars. Sirius implications for future warehouse-scale computers. IEEE Micro Top Picks'16.

[4] J. Hauswald, M. A. Laurenzano, Y. Zhang, H. Yang, Y. Kang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, and J. Mars. Designing future warehouse-scale computers for sirius, an end-to-end voice and vision personal assistant. TOCS'15.

[5] J. Hauswald, T. Manville, Q. Zheng, R. Dreslinski, C. Chakrabarti, and T. Mudge. A hybrid approach to offloading mobile image classification. ICASSP'14.

[6] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. ASPLOS'17.

[7] Y. Kang, Y. Zhang, J. K. Kummerfeld, P. Hill, J. Hauswald, M. A. Laurenzano, et al. Data collection for dialogue system: A clinc perspective. NAACL'18.

[8] S. Larson, A. Mahendran, A. Lee, J. K. Kummerfeld, P. Hill, M. A. Laurenzano, J. Hauswald, et al. Outlier detection for improved data quality and diversity in dialog systems. NAACL'19.