Jongse Park — Research Statement

My primary area of interest is computer architecture. However, my PhD research work has spanned multiple areas including machine learning, distributed system, and software engineering, as the cutting-edge research requires multi-disciplinary efforts. My other strong interests include operating systems/networking, security, and cyber-physical systems. I am looking forward to collaborating in all of these areas. More specifically, I am interested in cross-stack innovations that holistically design system architecture for emerging applications, such as machine learning. These innovations offer solutions that redefine the traditional abstractions of computing for a variety of platforms–from datacenters to Internet of Things (IoT).

Dissertation Research

Over the last decades, general-purpose computing stack and its abstractions have provided both *performance* and *productivity*, which have been the main drivers for the revolutionary advances in IT industry. However, the computational demand of emerging applications grows rapidly and the rate of data generation exceeds the level where the capabilities of current computing systems can match. The challenges have coincided with the Dark Silicon era in which conventional technologies offer insufficient performance and energy efficiency. Thus, it is timely to move beyond conventional techniques and explore radical approaches that can overcome the limitations of general-purpose systems and deliver large gains in performance and efficiency. One such approach is *specialization*, where the hardware and systems are developed for a domain of applications. However, the specialization creates a tension between the performance and productivity, since (1) programmers need to delve into the details of specialized hardware, and (2) perform low-level programming. Hence, the objectives are (1) delivering large gains in performance and efficiency (2) while retaining automation and productivity through high-level abstractions. Achieving both of these conflicting objectives is a crucial challenge to place the specialization techniques in a position of practical utility, which is the main focus of my PhD research. My works offer algorithm-driven computing stacks, which span from algorithms and languages to micro-architectural designs. I have primarily focused on two paradigms of specialization: *acceleration* and *approximation*. My efforts in acceleration leverage algorithmic insights to redefine the hardware-software abstractions and enable programmers to automatically utilize hardware accelerators (e.g., FPGAs) in scale-out setting for emerging workloads, such as data analytics and machine learning. For approximation, I have devised programming language and crowdsourcing software engineering solutions that improve the productivity and utility of approximation technologies, and bridges the gap between unconventional research innovations and practical real-world applications.

Resolving Performance and Productivity with Full Stack Designs for Acceleration of Machine Learning

A growing number of commercial and enterprise systems increasingly rely on compute-intensive Machine Learning (ML) algorithms. Hardware accelerators offer several orders of magnitude higher performance than general-purpose processors and provide a promising path forward to accommodate the needs of ML algorithms. Even software companies have begun to incorporate various forms of accelerators in their data centers. Microsoft's Project Brainwave integrated FPGAs in datacenter scale for real-time AI calculations and Google developed the TPU as a specialized matrix multiplication engine for machine learning. However, not only do the benefits come with the cost of lower programmability, but also the acceleration requires long development cycles and extensive expertise in hardware design. Moreover, conventionally, accelerators are integrated with the existing computing stack by profiling hot regions of code and offloading the computation to the accelerators. This approach is suboptimal since the stack is designed and optimized merely for CPUs, the sole processing platform up until very recently. To tackle these challenges, we developed cross-stack and algorithm-hardware co-designed solutions that rebuild the computing stack for acceleration of machine learning. These solutions break the conventional abstractions of computing stack by reworking the entire layers of computing stack, which include programming language, compiler, system software, accelerator architecture, and circuit generator.

Full stack solution for scale-out acceleration of learning. Recently, community has started exploring mostly single-node acceleration techniques to meet the massive compute demand of ML. In a concurrent yet disjoint effort, others have also explored the use of distributed general-purpose systems (e.g., Spark and Hadoop) as a mean to scale the learning frameworks. However, there is a gap between these accelerators and scale-out systems due to the lack of solutions that enable distributed acceleration of learning at scale. To bridge these two paradigms, we developed CoSMIC [1], a full computing stack that constitutes language, compiler, system software, template architecture, and circuit generators, which enable programmable acceleration of learning at scale. CoSMIC enables programmers to exploit scale-out acceleration using FPGAs and Programmable ASICs (P-ASICs) from a high-level and mathematical Domain-Specific Language (DSL). Nonetheless, CoSMIC does not require programmers to delve into the onerous task of system software development or hardware design. CoSMIC achieves three conflicting objectives of efficiency, automation, and programmability, by integrating a novel multi-threaded template accelerator architecture and a cohesive stack that generates the hardware and software code from its high-level DSL. CoSMIC can accelerate a wide range of learning algorithms that are most commonly trained using parallel variants of gradient descent. The key is to distribute partial gradient calculations of the learning algorithms across the accelerator-augmented nodes of the scale-out system. Additionally, CoSMIC leverages the parallelizability of the algorithms to offer multi-threaded acceleration within each node. Multi-threading allows CoSMIC to efficiently exploit the numerous resources that are becoming available on modern FPGAs/P-ASICs by striking a balance between multi-threaded parallelism and single-threaded performance. CoSMIC takes advantage of algorithmic properties of machine learning to offer a specialized system software that optimizes task allocation, role-assignment, thread management, and internode communication. While accelerators gain traction, their integration in the system stack is not well understood. CoSMIC takes an initial step toward such an integration for an important class of applications, while providing generality and a high-level programming interface.

An algorithmic approach to accelerate machine learning. As a preliminary effort for the CoSMIC project, in collaboration with my fellow graduate students, Divya Mahajan and Hardik Sharma, we developed a single-node FPGA accelerator generation framework for data analytics, dubbed Tabla, which enables FPGA acceleration from high-level specifications of algorithms. We open-sourced the code and it is available at http://act-lab.org/artifacts/tabla. Tabla leverages the insight that many learning algorithms can be solved using stochastic gradient descent that minimizes an objective function. The solver is fixed while the objective function changes with the learning algorithm. Therefore, Tabla uses stochastic optimization as the abstraction between hardware and software. Consequently, programmers specify the learning algorithm by merely expressing the gradient of the objective function in our domain specific language. Tabla then automatically generates the synthesizable implementation of the accelerator for FPGA realization using a set of template designs. Real hardware measurements show orders of magnitude higher performance and power efficiency while the programmer only writes 60 lines of code. This work received the Distinguished Paper Award in HPCA 2016. As a follow-on work, we developed DnnWeaver [2], a framework that automatically generates a synthesizable accelerator for a given (DNN, FPGA) pair from a high-level specification in Caffe. To achieve large benefits while preserving automation, we devised hand-optimized design templates that the DnnWeaver framework uses to generate the accelerators. First, DnnWeaver translates a given high-level DNN specification to its novel ISA that represents a macro dataflow graph of the DNN. The DnnWeaver compiler is equipped with our optimization algorithm that tiles, schedules, and batches DNN operations to maximize data reuse and best utilize target FPGA's memory and other resources. The final result is a custom synthesizable accelerator that best matches the needs of the DNN while providing high performance and efficiency gains for the target FPGA.

Improving Productivity in Approximate Computing

Approximate computing is another form of specialization, which brings forth an unconventional yet innovative computing paradigm that trades accuracy of computation for otherwise hard-to-achieve performance and efficiency. This new computing paradigm is built upon the property that emerging applications (e.g., sensor processing, translation, vision, and data analytics) are increasingly tolerant to imprecision. Leveraging this property, approximation techniques are able to provide orders of magnitude higher performance and efficiency gains, while maintaining the acceptable level of functionalities. However, these techniques are only pragmatic when (1) they are easy to use for the programmers, and (2) they produce acceptable output quality from the perspective of application users. To this end, my research efforts for approximation focus on improving productivity and utility of approximation technologies by developing programming language and crowdsourcing-based software engineering solutions.

Practical approximate programming. While approximate computing is currently a hot area, the programmability of approximation techniques is still one of the pivotal challenges to enable their practical and prevalent use. State-of-the-art approximate programming models require extensive manual annotations on program data and operations to guarantee safe execution of approximate programs. The need for extensive manual annotations hinders the pragmatic use of approximation techniques. We developed a small set of language extensions, dubbed FlexJava, that significantly reduces the annotation effort, paving the way for practical approximate programming [3]. These extensions enable programmers to annotate approximation-tolerant method outputs. The FlexJava compiler, which is equipped with an approximation safety analysis, automatically infers the operations and data that affect these outputs and selectively marks them as approximable while providing safety guarantees. The automation and the language-compiler co-design relieve programmers from manually and explicitly annotating data declarations or operations as safe to approximate. FlexJava is designed to support safety, modularity, generality, and scalability in software development. Compared to other models, FlexJava largely reduces the number of annotations and programmers spend significantly less time annotating programs based on our user study.

Crowdsourcing quality target determination in approximate computing. Approximation is useful only if its impact on application output quality is acceptable to the users. However, there is a lack of systematic solutions and studies that explore users' perspective on the effects of approximation. We sought to provide one such solution for the developers to probe and discover the boundary of quality loss that most users will deem acceptable. We proposed AxGames, a crowdsourced solution that enables developers to readily infer a statistical common ground from the general public through three entertaining games [4]. The users engage in these games by betting on their opinion about the quality loss of the final output while the AxGames framework collects statistics about their perceptions. The framework then statistically analyzes the results to determine the acceptable levels of quality for a pair of (application, approximation technique). The three games are designed such that they effectively capture quality requirements with various tradeoffs and contexts. We recruited 700 participants/users through Amazon's Mechanical Turk to play the games that collect statistics about their perception on different levels of quality. Subsequently, the AxGames framework uses the Clopper-Pearson exact method statistically project the quality level that satisfies a given percentage of users. The developers can use these statistical projections to tune the level of approximation based on the user experience.

In addition to the aforementioned works, I have worked on many collaborative projects that aim to develop hardware-software co-designed approximation techniques. These techniques seek opportunities for approximation at various components of computing stack, which span from neural accelerators [5, 6] and memory subsystems [7] to GPUs [8] and hardware description languages [9].

Future Research Directions

Recent innovations in ML are set to revolutionize medicine, robotics, commerce, transportation, and many other aspects of our lives. Such transformative effects are predicated on providing (1) high-performance compute capabilities that enable learning of compute-intensive ML models, and (2) constantly advancing ML algorithms that can adopt to ever-changing application needs. Computer system and architecture community has taken the charge of fulfilling the first precondition. The advances in the realm of computer system and architecture have not only unleashed the capabilities of unsung ML algorithms, which used to be computationally infeasible for many practical problems, but also offered opportunities for further advances in the algorithms. However, current systems mostly rely on completely offloading intelligence to the cloud. This approach is not scalable in the era of Intelligent IoT, and raises privacy and security concerns. To this end, my future research will focus on enabling intelligence and learning on the edge. As the first step, I will devise an algorithm-defined specialized computing stack for accelerating ML and AI on edge devices. Then, I will develop hardware-assisted privacy and security solutions for intelligent edge devices. To even further reduce the reliance on cloud, I will develop in-network acceleration techniques, which enable the reduction of edge-cloud communication overhead as well as offloading partial computation of learning to network fabric.

Pushing intelligence to the edge. IT industry has reached to a point where the capabilities of ML are enough to be integrated with real-world applications. ML based services (e.g., Amazon Alexa) are expanding their capabilities and available on edge devices such as mobile phone. However, currently, the machine intelligence at the edge devices is still only active when they are connected to centralized, high performance cloud platforms. This system architecture is built upon the producer-consumer model that the high-performance, power-hungry cloud servers learn and execute the ML models, while the low-performance, energy-limited edge devices merely communicate the model inputs and outputs with the cloud. This separation is suboptimal since the data exchange between these compute platforms not only incurs significant intercommunication cost, but also raises privacy and security concerns when sensitive personal information is exchanged. To this end, I plan to explore solutions that push intelligence to the edge, where the edge devices have the learnt ML models in the local storage and perform the model inference. The goal is to obtain sufficient energy-efficiency and performance to enable inference at the edge while offering programmability to support diverse ML algorithms. Therefore, I plan to research reconfigurable accelerators for the energy-limited edge systems and to develop programming abstractions for the accelerators, building upon my most recent work [16].

Online learning at the edge. My next step after realizing inference at the edge is to enable learning and adaptation, which is still left on the cloud platforms. The challenge is that it is infeasible for edge devices to completely take over the entire learning tasks from remote servers due to the growing scale and complexity of modern learning models. As such, the objective will be to design learning system architectures for heterogeneous compute platforms–from high-performance cloud servers to power-constrained edge devices–which the learning tasks are split and distributed over. With my experience in building specialized computing stacks, I will develop a complete stack that provides high-level abstractions to unite the heterogeneous compute platforms as a single learning system. In developing the stack, I will leverage the online active learning algorithms, such as federated and few/one/zero shot learning, which are designed for learning in the decentralized and small data setup.

Private and secure learning. Although the effort of pushing intelligence towards the edge reduces the inter-platform communication, it cannot completely resolve the privacy and security concerns due to the unavoidable exchange of sensitive data between clouds and edge systems. Large internet companies such as Apple, Amazon, and Google run their own clouds, which collect enormous amount of sensitive personal data from edge devices to provide the ML based services and improve the ML model accuracy. While conventional cryptographic algorithms and more recently introduced blockchain technologies may meet the privacy and security demand, these solutions require enormous amount of compute power, which make them infeasible to be hosted on general-purpose systems in the energy-budgeted environment. To tackle this challenge, I will first develop the hardware-friendly security algorithms that can enable the private and secure learning at the end systems without imposing significant hardware complexity. I will then devise programmable accelerators and their programming abstractions so that the acceleration systems not only offer efficiency to meet the performance and energy constraints of the edge systems, but also provides expressibility for a wide range of security algorithms.

In-network computation. Utilizing heterogeneous yet interconnected compute platforms for learning puts the inter-platform networking as a part of learning process. In this setup, the edge systems should exchange not only the input/output data, but also the intermediate model parameters with the remote systems. Moreover, the increasing number of compute devices and their complex network connectivity amplify the burden on the inter-platform networking fabric. I will explore the in-network acceleration techniques that not only enable transparent data transformation (e.g., compression), but also offloads parts of the learning computation to the compute-enabled network components. Then, I will extend the research to integrate in-network acceleration with large-scale datacenter network fabrics for scale-out learning frameworks, such as Spark and Tensorflow.

Publications

[1] Jongse Park, Hardik Sharma, Divya Mahajan, Joon Kyung Kim, Preston Olds, and Hadi Esmaeilzadeh. Scale-out acceleration for machine learning. In *MICRO*, 2017.

[2] Hardik Sharma, Jongse Park, Divya Mahajan, Emmanuel Amaro, , Joon Kyung Kim, Chenkai Shao, Asit Misra, and Hadi Esmaeilzadeh. From High-Level Deep Neural Models to FPGAs. In *MICRO*, 2016.

[3] Jongse Park, Hadi Esmaeilzadeh, Xin Zhang, Mayur Naik, and William Harris. FlexJava: Language Support for Safe and Modular Approximate Programming. In *FSE*, 2015.

[4] Jongse Park, Emmanuel Amaro, Divya Mahajan, Bradley Thwaites, and Hadi Esmaeilzadeh. AxGames: Towards Crowdsourcing Quality Target Determination in Approximate Computing. In *ASPLOS*, 2016.

[5] Renée St Amant, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Hadi Esmaeilzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger. General-Purpose Code Acceleration with Limited-Precision Analog Computation. In *ISCA*, 2014.

[6] Divya Mahajan, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, and Hadi Esmaeilzadeh. Towards Statistical Guarantees in Controlling Quality Tradeoffs for Approximate Acceleration. In *ISCA*, 2016.

[7] Bradley Thwaites, Gennady Pekhimenko, Amir Yazdanbakhsh, Jongse Park, Girish Mururu, Hadi Esmaeilzadeh, Onur Mutlu, and Todd Mowry. Rollback-Free Value Prediction with Approximate Loads. In *PACT*, 2014.

[8] Amir Yazdanbakhsh, Jongse Park, Hardik Sharma, Pejman Lotfi-Kamran, and Hadi Esmaeilzadeh. Neural Acceleration for GPU Throughput Processors. In *MICRO*, 2015.

[9] Amir Yazdanbakhsh, Divya Mahajan, Bradley Thwaites, Jongse Park, Anandhavel Nagendrakumar, Sindhuja Sethuraman, Kartik Ramkrishnan, Nishanthi Ravindran, Rudra Jariwala, Abbas Rahimi, Hadi Esmaeilzadeh, and Kia Bazargan. Axilog: Language Support for Approximate Hardware Design. In *DATE*, 2015.

[10] Divya Mahajan, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, and Hadi Esmaeilzadeh. Prediction-Based Quality Control for Approximate Accelerators. In *WACAS*, 2015.

[11] Hardik Sharma, Jongse Park, Emmanuel Amaro, Bradley Thwaites, Praneetha Kotha, Anmol Gupta, Joon Kyung Kim, Asit Misra, and Hadi Esmaeilzadeh. DnnWeaver: From High-Level Deep Network Models to FPGA Acceleration. In *CogArch*, 2016.

[12] Divya Mahajan, Kartik Ramkrishnan, Rudra Jariwala, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Anandhavel Nagendrakumar, Abbas Rahimi, Hadi Esmaeilzadeh, and Kia Bazargan. Axilog: Abstractions for Approximate Hardware Design and Reuse. *IEEE MICRO Special issue on Alternative Computing Designs and Technologies*, 2015.

[13] Amir Yazdanbakhsh, Bradley Thwaites, Jongse Park, and Hadi Esmaeilzadeh. Methodical Approximate Hardware Design and Reuse. In *WACAS*, 2014.

[14] Amir Yazdanbakhsh, Renee St. Amant, Bradley Thwaites, Jongse Park, Hadi Esmaeilzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger. Toward General-Purpose Code Acceleration with Analog Computation. In *WACAS*, 2014.

[15] Jongse Park, Kangqi Ni, Xin Zhang, Hadi Esmaeilzadeh, and Mayur Naik. Expectation-Oriented Framework for Automating Approximate Programming. In *WACAS*, 2014.

[16] Hardik Sharma, Jongse Park, Naveen Suda, Liangzhen Lai, Benson Chau, Vicas Chandra, and Hadi Esmaeilzadeh. Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks. In *ISCA*, 2018.