Research Statement

Chao Chen, *Amazon Science – chao.chen@gatech.edu*

My research interests span the field of compilers and computing systems with an emphasis on system reliability and performance. To pursue performance requirements of emerging workloads, such as machine learning and big data applications, modern computing systems are designed with increasing complexity by integrating heterogeneous computing units (e.g., CPU, GPU and FPGA), various types of memory modules (including DRAM and Non-Volatile RAM) and additional components in a single system. However, it also raises several challenges for applications to efficiently fulfill the hardware performance. For example, the future extreme-scale systems are anticipated to be more vulnerable to transient faults, and the hybrid computing/memory resources make it difficult for applications to maximize performance gains due to modern operating systems treating applications as black-boxes, therefore lacking knowledge of precise dynamic characteristics/requirements of applications. To efficiently address these problems, **there is an increasing need to explore applications' properties in system designs, and the system software should be knowledgeable about the applications and their requirements at runtime to dynamically make better decisions**. Thus, my main research goal is to build efficient systems by exploring applications' both static and dynamic characteristics, e.g., code properties, computation patterns and resource requirements, via program analysis and compiler techniques. Here, I would like to highlight my thesis work on resiliency, and my research agenda for future works in the mentioned areas.

Compiler-Assisted Resilience Framework for Reliable Computing

The ability to boost system performance is mainly driven by the manufacturing trends toward smaller transistor size, higher circuit density, and near-threshold voltage operations. However, these technologies would make electronic circuits more vulnerable to particle-strike induced transient errors that may cause crashes (Soft Failures) or silent data corruption (SDC). As a result of system failures, scientific applications running on exascale high performance computing facilities are anticipated to be interrupted roughly once every 15 minutes. Unfortunately, traditional hardware approaches are not a viable solution to the reliability problem, because they put additional pressure on the nations tight power budget. Given that the Exascale resilience must be achieved in an energy-efficient manner within the power budget, I have endeavored to design and implement compiler-assisted resilience techniques, which were published in one of the top conferences and journals selected by csrankings, e.g., HPDC, SC (Best Student Paper Finalist), and TPDS.

Low-cost Application-level Detector for Reducing SDCs [HPDC'18]. SDC is a big threaten to modern scientific discoveries, since they could lead to wrong computing results therefore incorrect scientific insights. Complete and comprehensive SDC detection always requires the duplication of computing, but the overheads required ($\sim$2$\times$ resources) limits their adoption in scientific computing. Following the state-of-the-art anomaly-based methods, I created LADR, which is to detect SDCs by watching for data anomalies in state variables of scientific applications and minimize the overhead by exploiting correlations among state variables and data points mainly via data-flow analysis. LADR detects SDCs by monitoring the value changes of a tiny set of data points. It utilizes the data-flow of the application in order to determine the data points to be monitored, therefore to limit its monitoring overhead but without losing fault coverage as compared to the state-of-the-art

techniques. Its negligible overheads coupled with attractive detection precision makes it a viable approach for assuring the correct output from large-scale high performance simulations.

Compiler-Assisted Recovery from Soft failures [SC'19, Best Student Paper Finalist]. As compared to SDC, less research effort has gone into handling soft failures, mainly because the standard Checkpoint/Restart (C/R) methods can provide adequate recovery. But C/R techniques are very costly in terms of lost opportunities (batch job slots), lost computation (everything since the last checkpoint) and I/O overheads (repeatedly writing checkpoint files). These costs are particularly significant for massively parallel jobs. In addition, it is also very challenge to scale them for exascale systems. The crux of the problem is also due to the lack of knowledge that can help designing light-weight recovery methods for soft failures. By carefully examining the manifestation of soft failures, I find that the majority of them are due to invalid memory accesses, which means faults corrupted instructions that were involved in address computations. Motivated by this observation, my thesis proposed a new framework to repair the corrupted address for the failing process on-the-fly through replaying related computations from applications. It is guided by compiler techniques, and leverages applications' knowledge to determine what computations to replay by building recovery kernels for each memory access instruction at compile-time. In particular, the proposed method has *no run-time overhead* unless an actual transient fault occurs and spends only a few microseconds in recovery. This work was selected as the best student finalist for IEEE/ACM Supercomputing Conference (SC) 2019, one of the premier conference in the field of Supercomputing.

Leveraging Code Optimizations for Resilience Purpose [TPDS'22]. Despite the promising results I achieved in our SC19 work, it is still very challenging to recover from failures in such things as induction variable updates, which cause a significant portion of failures in many scientific workloads. To address this challenge, I looked into the code optimization techniques available in modern compilers, and find that some of these techniques, such as strength-reduction, can introduce independent but equivalent computations (patterns) by turning array accesses into strength-reduced pointers which are updated independently in lockstep. If one of these values is corrupted in an update, a correct value for the corrupted pointer can be inferred from the value of another. To this end, I created two additional compiler passes to expose such compiler-induced "accidental" redundancy for resilience purpose, and leverage them to build smarter kernels for recovery from a broader range of soft failures, with no impact on code speed. The beauty of this work is that it introduces almost zero performance overhead, but has achieved significant fault coverage due to its novel idea of leveraging side-effects of modern code optimization techniques.

Future Plan

In my future work, I wish to provide users with systems and tools to manage and leverage next-generation computing platforms. Ideally, such systems would harness modern commodity hardware designs (such as multi-core processors, accelerators, and Non-Volatile RAMs) and explore the properties of applications via compiler techniques to maximize system efficiency and performance. In particular, I plan to expand my research in the following areas:

• **Resiliency.** Failures and their effects on system performance and correctness are a major roadblock to the extreme-scale systems. Despite techniques introduced in my thesis, there are still many open challenges coming from various sources. For example, how can we handle faults manifested in

accelerators (which have become de facto computing units in modern HPC systems)? Different programming abstractions, such as objects, also pose challenges because they impact the relationship between memory access and address computations. In the near term, I plan to continue my work on resiliency to support new hardware and new language features.

• **Active Storage.** The performance and scalability of the I/O subsystem are also of significant concern to future HPC systems. Following the idea of moving computation nearer to data, active storage is an interesting topic for mitigating this issue by moving computation into storage nodes, even disk controllers. While active storage has a great potential in many areas, such as Big Data, there is still a significant lack of tools and systems that give access to that potential. I see several open challenges in this space, such as how to identify and decouple computations that can benefit from its design from applications, and how to flexibly offload computations to the data site (e.g., storage nodes, or disk controllers). My current work with compiler techniques has led to insights that I believe are applicable in this space, and I hope to extend the work in this area, with the ultimate goal of developing tools and run-times to automatically and flexibly offload computations along the I/O path to maximize system performance.

• **Emerging Hardware and Workloads.** Finally, I am also interested in exploring the properties of new hardware, such as GPU and NVRAM, to improve the system performance. In my current research pipeline, I am looking at how to efficiently share (schedule) a GPU among different workloads, and how to explore the capacity of NVRAM and hide its latency for applications. I believe the solutions to these problems requires a good understanding of fine-grain properties of workloads and that the compiler can play an important role in helping to identify those requirements. A paper (arXiv:2107.08538) summarizing our early results about GPU scheduling was recently accepted by **PPoPP'22**. In the future, I will extend the related technique and my research to the area of **Deep Learning Compiler and Systems** to improve the performance and efficiency for deep learning workloads, as inspired by my current work at Amazon Science.

To support my future research in these areas, NSF programs (e.g., PPoSS, RINGS), DoE (via the cooperation with National Labs e.g., Los Alamos and Argonne) and industries (e.g, Amazon Science) will be my main funding targets. I started to build relationships with national labs and industry via my internships, work experience and dissertation committee. These experiences will help me to establish potential collaborations with their scientists and employees.