Statement of Research Interests

Yuxuan Zhang, University of Pennsylvania

As we have entered the Post-Moore Era where transistor size is reaching its limit, single-core performance can no longer be improved much. Although peripherals' performance and capacity have been dramatically increased and the concept of heterogeneous designs which allow the processor to offload part of its work to accelerators for speeding up program's execution have been proposed, the general purpose processor itself can still be a bottleneck for application's performance.

To address the inefficiency caused by processors, especially cache bottlenecks that remain despite decades of hardware innovations, I build systems that leverage both hardware and software techniques to improve application's performance. In these systems, hardware *monitors* performance and provides feedback to software, then software *optimizes* based on the feedback reported from hardware while applications are running. My past research includes two projects: (1) Online COde Layout Optimizations (Ocolos) [6] [7] and (2) Robust Profile-Guided Runtime Prefetch Generation (RPG^2). The former has shown it is feasible to apply profile-guided code layout optimization at runtime with low overhead. The latter shows that runtime optimization can also be applied to data cache prefetching if the code layout is carefully tuned to support on-stack replacement and continuous optimization. In this statement, I summarize my prior works, and detail my future research plans.

1 Prior Research - Profile-Guided Code Layout Optimization at Runtime

Online COde Layout Optimizations (Ocolos): Prior research [1] has pointed out that the capacity of the instruction-cache of server processors cannot handle the growth of today's data center applications and their growing instruction working set. Several profile-guided compiler-based approaches [2] [3] [5] have been proposed to address the gap between the instruction cache capacity and the code size. However, they all require halting the program to re-launch the optimized application. This is unacceptable for many services which handle thousands of queries or tasks per second, and these queries or tasks may change rapidly depending on how the inputs change. A newly optimized code layout may become stale after a relatively short period of time. Then it becomes necessary to change the code of the running process again once the system detects that the current code layout is not optimal for the incoming queries or tasks anymore.

To address this problem, I implemented Ocolos [6] [7], the first *online* code layout optimization system for unmodified applications written in unmanaged languages like C and C++. Ocolos allows profile-guided optimization to be performed on a running process, instead of being performed offline and requiring the application to be re-launched. By running online, profile data is always relevant to the current execution and always maps perfectly to the running code. Ocolos demonstrates how to achieve robust online code replacement in complex multithreaded applications like MySQL and MongoDB, without requiring any application changes. Our experiments show that OCOLOS can accelerate big-code applications such as MySQL by up to 1.41×.

Robust Profile-Guided Runtime Prefetch Generation (RPG^2): Data cache prefetching is a well-established problem. Since current major server processors employ simple hardware prefetchers, which fail to capture complex indirect access patterns, a number of static data prefetch compilers [8] [9] have been proposed. However, as data needs to be prefetched depends on not only program structure, but also program input and the microarchitecture, it is hard for static compiler-based prefetch insertion approaches to provide a solution that can benefit to an application across all inputs. Moreover, depending on the input and the microarchitecture, prefetch instructions can sometimes be harmful to performance. It is hard for developers or compilers to know up-front if prefetch will help or hurt.

To make up for the shortcomings caused by static compiler-based prefetch insertion approaches, I designed and implemented the RPG^2 system for *online* prefetch injection and tuning. RPG^2 profiles a running C/C++ program, injects prefetch instructions and then tunes those prefetches to maximize performance. RPG^2

can provide a comparable speedup to the best profile-guided prefetching insertion compilers, but can also respond when prefetching ends up being harmful and roll back to the original code – something that static compilers cannot. In other words, RPG^2 makes prefetching a safer optimization by preserving its upsides, and protecting programs from its downsides. According to our experimental results, RPG^2 can provide speedup up to 2.15× in the case where prefetch can benefit to the performance. In static compiler-based prefetch insertion approach's worst case scenario where there is a 70% slowdown, the slowdown caused by RPG^2 is only 7%.

2 Future Research Directions

PGO-based Instruction Cache Prefetching at Runtime: Ocolos performs online code layout optimization by reordering basic blocks and functions, because if frequently accessed basic blocks and functions are reordered to be adjacent, capacity and conflict misses of instruction cache caused by big-code applications will be significant reduced. However, reordering basic blocks and functions cannot help with reducing compulsory misses. It is also impossible to depend on the existing hardware prefetchers to figure out the instruction to be prefetched from complicated control flow, since processors have only simple next-line prefetchers in hardware. Although previous work [2] [3] [4] tried to address the problem of reducing instruction cache's compulsory misses, the solutions are offline only and simulation only because of a lack of ISA support for code prefetch instructions. The good news is that new instructions being rolled out soon by Intel with its Redwood Cove microarchitecture will allow hardware instruction prefetching from software. Hence, even complicated instruction memory access can be prefetched on real machines if a software prefetching mechanism that leverages this hardware instruction prefetching is carefully designed.

With the arrival of Intel's new hardware instruction prefetching, now is the time to consider a real implementation of online instruction cache prefetching. It would be interesting to see how well the static instruction cache prefetching approaches cooperate with Ocolos to perform online instruction prefetching. For example, if we leverage I-SPY's strategy [3] to drive *online* analysis of instruction cache miss behavior, and replace the code region with the newly constructed executable that has conditional prefetching instructions injected at runtime, we might be able to see a lower instruction-cache miss ratio. So, one of my future research directions will be leveraging these existing offline instruction-cache layout optimization solutions to build a corresponding online version.

References

[1] S. Kanev, J. P. Darago, K. Hazelwood, P. Ranganathan, T. Moseley, G. Wei, D. Brooks, "Profiling a warehouse-scale computer," *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture* (*ISCA*), Portland, OR, USA, 2015, pp. 158-169.

[2] G. A. Nayana, P. Nagendra, D. I. August, H. Cho, S. Kanev, C. Kozyrakis, T. Krishnamurthy, H. Litz, T. Moseley, P. Ranganathan, "AsmDB: Understanding and Mitigating Front-End Stalls in Warehouse-Scale Computers," *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture* (*ISCA*), Phoenix, AZ, USA, 2019, pp. 462-473.

[3] T. A. Khan, A. Sriraman, J. Devietti, G. Pokam, H. Litz and B. Kasikci, "I-SPY: Context-Driven Conditional Instruction Prefetching with Coalescing," *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture* (*MICRO*), Athens, Greece, 2020, pp. 146-159.

[4] T. A. Khan , D. Zhang, A. Sriraman, J. Devietti, G. Pokam, H. Litz, B.Kasikci, "Ripple: Profile-Guided Instruction Cache Replacement for Data Center Applications," *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture* (*ISCA*), Valencia, Spain, 2021, pp. 734-747.

[5] M. Panchenko, R. Auler, B. Nell and G. Ottoni, "BOLT: A Practical Binary Optimizer for Data Centers and Beyond," *2019 IEEE/ACM International Symposium on Code Generation and Optimization* (*CGO*), Washington, DC, USA, 2019 pp. 2-14.

[6] Y. Zhang, T. A. Khan, G. Pokam, B. Kasikci, H. Litz and J. Devietti, "OCOLOS: Online COde Layout OptimizationS," *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 530-545.

[7] Y. Zhang, T. A. Khan, G. Pokam, B. Kasikci, H. Litz and J. Devietti, "Online Code Layout Optimizations via OCOLOS," in *IEEE Micro, vol. 43, no. 4, pp. 71-79, "Top Picks From the 2022 Computer Architecture Conferences"*, July-Aug. 2023.

[8] S. Jamilan, T. A. Khan, G. Ayers, B. Kasikci, and H. Litz. 2022, "APT-GET: profile-guided timely software prefetching," *Proceedings of the Seventeenth European Conference on Computer Systems (EuroSys '22)*, New York, NY, USA, 747–764.

[9] T. A. Khan, I. Neal, G. Pokam, B. Mozafari, and B. Kasikci. "Dmon: Efficient detection and correction of data locality problems using selective profiling", *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21), pages 163–181*. USENIX Association, July 2021.