Review of: "Division of Labor: A More Effective Approach to Prefetching"

Bryce A Hickle

"Division of Labor: A More Effective Approach to Prefetching" conducted by Sushant Kondguli and Michael Huang is an article published through the joint ACM and IEEE international symposium in 2018. The problem addressed in this research is one such that what is the optimal way to minimize latency (the amount of time spent between the input of a system and its output) via prefetching (or gathering information from memory preemptively) (Kondguli and Huang, 83). Kondguli and Huang demonstrate that prefetching methods, consisting of smaller prefetchers designed for more specific and smaller scoped patterns, lead to both a performance and accuracy increase in comparison to traditional prefetching techniques in use which rely on large-scoped/generalized prefetchers (prefetchers that are much less "trained" in a sense to the patterns necessary to know when pre-retrieval from memory is possible) (Kondguli and Huang, 83).

This article attempts to verify its research by comparing their approach to that of prefetching techniques used currently (Kondguli and Huang, 94). To do so, this modular, composite, prefetcher (made of three sub-component prefetchers) was tested on a variety of inputs ranging from CPU benchmark tests, data structures gathered from internet resources, embedded applications, and scientific computing and detected repetition of an instruction inside the innermost loops of the given input (Kondguli and Huang, 89-90, 94). Data collected was presented as accuracy (whether the prefetching activated on useful addresses) over scope (the amount of addresses attempted at being prefetched) (Kondguli and Huang, 91-92). The team also compared the speed of the new prefetcher compared to typically designed prefetchers on the testing suite mentioned earlier and found that the new prefetcher was, on average, six percent faster than the competitors in execution time (measured in nanoseconds) (Kondguli and Huang, 90). Also, the prefetcher was the number one best performer in 11 out 21 benchmarks in the test suite (Kondguli and Huang, 90). Accuracy was the most dramatic with the new prefetcher

achieving 49 percent effective accuracy on a worst-case input compared to a range of 7 to 23 percent effective accuracy on a worst-case input of traditional prefetchers; the new prefetcher also achieved 82 percent accuracy on average input compared to 69 percent on average input in traditional prefetchers (Kondguli and Huang, 91). Throughout testing, the composite prefetcher showed improvements compared to monolithic, traditional prefetchers (Kondguli and Huang, 91).

The research team concludes that the modular/composite prefetcher was both more efficient and more accurate in comparison to the traditional prefetchers which had a larger, more general scope (Kondguli and Huang, 94). They infer that fewer addresses need to be saved in regard to special patterns in the composite technique which caused an efficiency boost (Kondguli and Huang, 85). Whereas the accuracy boost was much more expected since the sub-components are better trained for finding specific patterns by design. Finally, the research team states a call to action for further research into composite prefetchers consisting of more sub-components than the three used in this experiment (Kondguli and Huang, 94).

This article is relevant to Computer Organization on multiple levels. First, the prime problem addressed in the article involves minimizing latency in architecture (Kondguli and Huang, 83). Latency is a fundamental property analyzed in circuits to predict time constraints. We've calculated latency in simple circuits recently and have seen firsthand how output is restricted by the slowest path in a system (the maximal latency in a necessary circuit path). This article was focused on preemptive memory access in order to minimize latency during memory retrieval, a well-known limiting factor in practical computation.

The concepts used in this article also relate to speculative computing which was addressed briefly at the assembler level. Jump instructions preemptively executing the next line

in MIPS, for instance. Preloading information from memory is also an example of speculative execution of data for the sake of increasing performance.

Thirdly, compiling, assembling, and running programs discussed in Computer Organization are also related conceptually to this research. The tests included finding patterns to recognize when to gather memory prematurely. Primarily, these prefetchers were looking to recognize loops in the form of single repetitive instruction execution for testing (Kondguli and Huang, 86). Improving the means to do so would improve the runtime of programs by reducing, say, the amount of addresses needed to be stored through the use of a composite prefetcher used in the experiment (Kondguli and Huang, 85).

The team's work is convincing. The size and variety of test input via the test suite mentioned previously in this review is impressive (running everything from CPU benchmark tests to executing a variety of algorithms relevant to multiple domains). The data (showcasing accuracy over scope and execution time in nanoseconds) used very standard methods of analyzing in my opinion. For example, comparison of averages were used in the analysis of all categories (Kondguli and Huang, 92). The modulated prefetcher showed a double percentage increase in accuracy in a worse and average case scenario, with just as competitive speed and scope, which only furthers my agreement in the conclusion of this research (Kondguli and Huang, 92).

References

S. Kondguli and M. Huang, "Division of Labor: A More Effective Approach to Prefetching," *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, Los Angeles, CA, 2018, pp. 83-95. doi: 10.1109/ISCA.2018.00018