1 Summary

Designing and Evaluation of Compiler Algorithm for Prefetching (1992): This is Mowry's 1992 Ph.D. thesis proposing a selective software prefetching algorithm. Their algorithm uses locality analysis, loop splitting, and software pipelining in order to reduce instruction prefetching overhead and stresses on the memory subsystem that were previous issues in software-controlled prefetching. Comparing their results on benchmarks done with no prefetching and indiscriminate prefetching, selective prefetching with loop splitting improved performance. Their conclusions lead to their assertion that, with the cost of complex hardware prefetches, microarchitectures should instead be supporting memory hierarchy optimizations like lockup-free caches and prefetching instructions.

2 Strengths

- Figures 3 and 4 were good representations of their performance results of their algorithm against control benchmarks, showing not only how their algorithm improved in overall execution time, but also a breakdown of how instructions and prefetching overhead compared across the benchmarks.

3 Weaknesses

- The setion 2 description of the prefetching algorithm was very verbose, and only referencing the same figure for the entire section made it easier to get lost in the details of the algorithm.
- The scope of their compiler algorithm was limited to affine array accesses within scientific applications, but a more broad application was not included, nor a discussion about feasibility/infeasibility of their algorithm generalized for more kinds of access patterns.

4 Rating: 3

5 Comments

It is interesting to read early papers on how specific kinds of memory optimizations were approached; in this case, instruction prefetching. Looking at their citations and on ACM, other papers on software-controlled prefetching were also published in the early 90s around the same time as this one, as well. The authors' assertions were correct, in that architectures since the time of their publishing in 1992 do accommodate for more software-controlled optimizations, either given by certain programming language features or new compiler optimizations being released. For a time, though, around the early 2000s, the focus shifted back on the architecture and optimizations on memory organization to improve performance; but,

referencing the 2017 Turing Lecture (Hennessey and Patterson), with another shift towards collaborative approaches to designing and optimizing processors (with architects, compiler writers, OS designers, etc.), there may be more opportunities for these optimization software algorithms in the microarchitecture. That being said, given this paper's age and the weaknesses listed above, another more recent paper or study covering prefetching could replace this one in the readings. This paper has had 633 forward citations, but are in the context of background information for these more recent papers, often as a one-sentence reference about the existence of software approaches to memory optimizations. The high level idea of software-controlled algorithms is still relevant but this particular study and their algorithm are less so.