McGovern, Jeff
CPE 315, Section 03
February 24, 2016
Lab 5

Lab 5: Cache Optimization

Contents

List of Tables

1 Differences

1.1 Runtime

The runtime of the optimized `matmul` shows a 5.5× speedup on the Raspberry Pi 1 and a 2.2× speedup on the Intel Lab Machine. It's possible that the lack of dramatic speedup on the Intel Lab Machine machine is due to the relatively small size of the program compared to, maybe, the process scheduler or to more programs running on the machine at the same time. Compared to the Raspberry Pi 1, which is a headless system, there are a multitude of aspect to the desktop environment that may demand the processor.

The runtime of the optimized `structs` program shows a 3.2× speedup on the Raspberry Pi 1 and a 1.1× speedup on the Intel Lab Machine. Again, it seems that the relatively small size of the program — markedly smaller than `matmul` — may be the culprit in why the Intel Lab Machine saw a smaller amount of improvement over the Raspberry Pi 1.

1.2 Cache Misses

The number of cache misses of the optimized `matmul` amount to 103% of the original on the Raspberry Pi 1 and 90% of the original on the Intel Lab Machine. The fact that the Raspberry Pi 1 had more cache misses is intriguing, but the fact that amount of times the optimized version accessed D cache was 19% of the original version[1], might explain why the increase is okay and still allowed for an improved runtime.

The number of cache misses of the optimized `structs` amount to 78% of the original on the Raspberry Pi 1 and 20% of the original on the `structs`. These numbers are surprisingly close to one another. The fact that these two are so similarly low may suggest that both architectures handle such a comparatively small computation similarly.

2 Conclusion

On both architectures, the optimized versions perform noticeably better than the original versions from a runtime standpoint. A notable exception is the increase in cache misses for the `matmul` optimized version on the Raspberry Pi 1. Such a discrepancy, while still having an improved runtime, may be explained by the decrease in cache accesses. For the `structs` optimizations, it appears that from a cache miss standpoint, both architectures had nearly identical improvements which may be attributed to the simplicity of the computation. Overall, it appears that cache access on modern processors is an important consideration when developing high performance programs.

[1] On the Raspberry Pi 1, the original version accessed the DC cache 79,701,846 times, while the optimized version only accessed it 14,938,111 times.

Appendix A

Table 1: Runtimes and Cache Misses for ARM and Intel naive and optimized solutions to multiplying two matrices and two vectors

	ARM no opt	**ARM opt**	**Intel no opt**	**Intel opt**
Runtime (`matmul`)	2,574,138,840 ns	455,362,448 ns	3,742,091,490 ns	1,682,116,854 ns
Cache Misses (`matmul`)	3,392,429	3,487,873	22,809	20,605
Runtime (`structs`)	104,925,000 ns	32,314,000 ns	99,169,799 ns	91,352,099 ns
Cache Misses (`structs`)	356,689	278,760	3,809,677	747,105