Shibo Xing

April 20, 2020

Dr. Xulong Tang

CS2210

Compiler Optimizations for Improving Data Locality Review

This paper aims enhance the data locality of the compilation process and consequently results in a smaller number of cache misses. It identified this optimization measure as a purely implicit one, meaning that it will be the burden of the compiler solely. Programmers can thus write their programs in a more machine independent style while achieving outstanding data locality. Overall, data locality utilizes the cache and fully take advantage of the fact that processor has superior speed to the main memory to enhance the run-time of a compiler.

The authors first mention their method of estimating the cost of the nest loop, which usually occupies 20% of the code segment but takes up 80% of the run-time. Since the fewer the accesses to main memory is, the better the data locality, the measures they use for estimating the loop cache references. Their algorithm first aggregates group re-uses, which includes memory accesses to identical and nearby locations and then calculate the cache line accesses in these groups. Large cache accesses indicate a high cost of the loop.

Then they move on to determining the loop permutation with the fewest cache accesses. As a matter of fact the auxiliary measure proposed by this paper focuses on exploiting the memory access order via different loop transformations. Among all the loop transformations, loop reversal gives poor result. Loop fusion enhances data locality by placing the same cache line accesses together and also gives better data locality by fusing inner loops and enabling loop permutations. Loop distribution also enables loop permutation on unpermutable loops nest. If the optimal memory order computed by the algorithm cannnot be placed as the innermost loop, permutation in memory order will be enabled by fusing all inner loops.

The unique part of the approach proposed by the paper is that this is the first data locality improvement method that involves compound loop transformations. It allows some imprecision along the process and has efficiency as an advantage. I think this paper’s accomplishment is indeed essential to today’s compiler performance as it utilizes machine-independent measures. It is an old paper and the cost models are used on Fortran programs. In today’s world with more machine platforms, data locality can certainly be studied and applied to other types of compilation and languages as well. Similarly to the scientific programming mentioned by the paper, Python nowadays is a dominating language used for deep learning and data science. And the measures of improving data locality might be used for Python JIT interpreter as well.