In More Depth: The Difficulty with Kernel Benchmarks

Even well intentioned efforts are sometimes ensnared in the process of choosing good benchmarks. For example, the SPEC processor benchmark suite was chosen to use primarily real applications. Unfortunately, the first release of the SPEC suite in 1989 included a benchmark called matrix300, which consists solely of a series of matrix multiplications. In fact, 99% of the execution time is in a single line of this benchmark. The fact that so much time is spent in one line doing the same computation many times has led several companies to purchase or develop special compiler technology to improve the running time of this benchmark.

Figure 4.8.6 shows the SPEC89 results for the standard and enhanced version of a compiler. The enhanced compiler has essentially no effect on the running time of 8 of the 10 benchmarks, but it improves performance on matrix300 by a factor of more than 9. On matrix300, the program runs 729.8 times faster using the enhanced compiler than the reference time obtained from a VAX-11/780—but the more typical performance of the computer is much slower. The other programs run from just over 30 times faster to just over 140 times faster. A user expecting a program to run 700 times faster than it does on a VAX-11/780 would likely be very disappointed! In the 1992 release of the SPEC benchmark suite, matrix300 was dropped.

4.49 [5] <§§4.2, 4.3, 4.4> Suppose that, on average, computer A is 10 times faster than computer B before optimization. A special optimizer improves the performance of one of the ten benchmarks, which originally required 1/5 of the execution time of the suite on A, by an extra factor of 50. How much faster will the entire benchmark suite run on computer A than on computer B, assuming the performance of computer B is left unchanged?

4.50 [15] <§§4.2, 4.3, 4.4> One of the remarkable aspects of the matrix300 incident is that the optimization (a technique called blocking) actually reduces the number of instructions executed from $O(n^3)$ to $O(n^2)$, where n is the dimension of the matrix! Hence, the optimizer can produce a relative speedup over the unoptimized version that scales with the size of the data! If we want the optimized version to be 10% faster than the unoptimized version, how large should the data size of the matrixXXX be? Assume that matrix300 (which has a data size of 300) is responsible for 20% of the execution time for both the optimized and unoptimized suite.

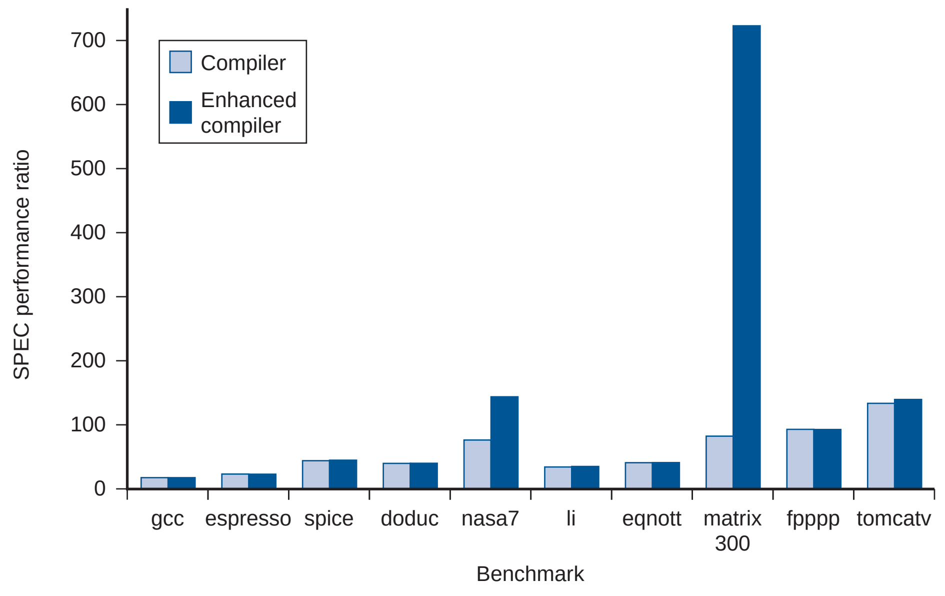

FIGURE 4.8.6 SPEC89 performance ratios for the IBM Powerstation 550 using two different compilers. The higher numbers on matrix300 (and nasa7) result from applying an optimization technique to these two kernel-oriented benchmarks. For the enhanced compiler, special flags are passed to the compiler for both nasa7 and matrix300, which are not used for the other benchmarks. In both programs, the compiler transforms the program by blocking the matrix operations that are in the inner loops. These blocking transformations substantially lower the number of memory accesses required and transform the inner loops from having high cache miss rates to having almost negligible cache miss rates. Interestingly, the original motivation for including matrix300 was to exercise the computer's memory system; however, this optimization basically reorganizes the program to minimize memory usage. This data appeared in two SPEC reports during the fall and winter of 1991. The susceptibility of this benchmark to compiler optimization, and the relatively uninteresting behavior of the benchmark after optimization, led to the elimination of matrix300 from the 1992 release of the SPEC benchmarks.