In More Depth: Synthetic Benchmarks

Synthetic benchmarks are artificial programs that are constructed to try to match the characteristics of a large set of programs. The goal is to create a single benchmark program where the execution frequency of statements in the benchmark matches the statement frequency in a large set of benchmarks. Whetstone and Dhrystone are the most popular synthetic benchmarks. Whetstone was based on measurements of Algol programs in a scientific and engineering environment. It was later converted to Fortran and became popular. Dhrystone, which was inspired by Whetstone, was created as a benchmark for systems programming environments and was based on a set of published frequency measurements. Dhrystone was originally written in Ada and later converted to C, after which it became popular.

One major drawback of synthetic benchmarks is that no user would ever run a synthetic benchmark as an application because these programs don't compute anything a user would find remotely interesting. Furthermore, because synthetic benchmarks are not real programs, they usually do not reflect program behavior, other than the behavior considered when they were created. Finally, compiler and hardware optimizations can inflate performance of these benchmarks, far beyond what the same optimizations would achieve on real programs. Of course, because these benchmarks are not natural programs, they may not reward optimizations of behavior that occur in real programs. Here are some examples of how Dhrystone may distort the importance of various optimizations:

Optimizing compilers can easily discard 25% of the Dhrystone code; examples include loops that are executed only once, making the loop overhead instructions unnecessary. To address these problems, the authors of the benchmark "require" both optimized and unoptimized code to be reported. In addition, they "forbid" the practice of inline procedure expansion optimization because Dhrystone's simple procedure structure allows elimination of all procedure calls at almost no increase in code size.

One C compiler appears to include optimizations targeted just for Dhrystone. If the proper option flag is set at compile time, the compiler turns the portion of the C version of this benchmark that copies a variable-length string of bytes (terminated by an end-of-string symbol) into a loop that transfers a fixed number of words. The compiler also assumes that the source and destination of the string is word-aligned in memory. Although an estimated 99.70% to 99.98% of typical string copies could *not* use this optimization, this single change can make a 20% to 30% improvement in Dhrystone's overall performance.

4.27 [3 hours] <§4.3> Pick two computers, A and B, and run the Dhrystone benchmark and some substantial C program, such as the C compiler, calling this program P. Try running the two programs using no optimization and maximum optimization. Then calculate the following performance ratios:

a. Unoptimized Dhrystone on computer A versus unoptimized Dhrystone on computer B

b. Unoptimized P on A versus unoptimized P on B

c. Optimized Dhrystone on A versus optimized Dhrystone on B

d. Optimized P on A versus optimized P on B

e. Unoptimized Dhrystone versus optimized Dhrystone on computer A

f. Unoptimized P versus optimized P on A

g. Unoptimized Dhrystone versus optimized Dhrystone on B

h. Unoptimized P versus optimized P on B

We want to explore whether Dhrystone accurately predicts the performance of other C programs. If Dhrystone does predict performance, then the following equations should be true about the ratios:

$$a = b \text{ and } c = d$$

If Dhrystone accurately predicts the value of compiler optimizations for real programs, then

$$e = f \text{ and } g = h$$

Determine which of the above relationships hold. For the situations where the relationships are not close, try to find the explanation. Do features of the computers, the compiler optimizations, or the differences between P and Dhrystone explain the answer?

4.28 [3 hours] <§4.3> Perform the same experiment as in Exercise , replacing Dhrystone with Whetstone and choosing a floating-point program written in Fortran to replace P.