THE EDITOR'S VIEW

Stupid Compiler Tricks

SPEC Results Inflated by Too Many Compiler Flags

There is an ongoing debate in the SPEC community about the extensive use of compiler flags to tune the performance of microprocessors on the various SPEC programs. On the one hand, the purpose of having a source-code benchmark is to demonstrate the combined performance of a compiler/processor pair; both the hardware and the software make a valid and valuable contribution to overall performance. On the other hand, well, things seem to have gotten out of hand.

For example, the compiler flags used by one vendor for a single SPECfp92 program are: -mips2 -non_shared -jmpopt -Nn16000 -Olimit 2000 -O3 -sopt,-s=3,-r=3,-o=5,-ur=8,-ur2=3000,-limit=10000,-arclimit=10000,-lo=CKLNS,-chs=16 -w -G 4800. That's 18 different parameters being set, many of which have a wide range of possible values. As in all recent SPEC submissions, this vendor used a different set of flags for each of the 20 programs in the SPEC suite.

The problem with this type of tuning is that the results are not indicative of performance on real applications. Most users have neither the time nor the expertise to fiddle with all these compiler options to achieve the best performance; they just set -O (general optimizations) to the highest level they can stand and let it fly. The current benchmarking system disguises the fact that some compilers and processors require intensive hand-tuning to achieve peak performance while others are more flexible; most users would probably prefer the latter.

Even users willing to play with parameters like -ur (number of times to unroll scalar inner loops) might have problems deciding which subroutines to copy into parent code, a process called inlining. While most compilers decide which routines to inline by trading off the overhead of calling the subroutine against the code expansion caused by duplicating the instructions, occasionally the compiler's decision is not optimal. Some vendors use a compiler flag to name specific routines that the compiler should inline, whether it wants to or not; users must make a long, tedious search to discover which routines respond to such special treatment.

Another set of flags can improve performance in specific instances but in general are "unsafe." These flags allow the compiler to use more aggressive optimizations by assuming that pointers are never aliased, or data objects are always aligned, or literals are always read-only. Flags in this category include -mP20PT_disamb_types=true, -qassert=typeptr, -aligned_data_env, and -qro. These flags are considered unsafe because, if they are used incorrectly, programs may crash or even generate incorrect results.

A more insidious issue is that some companies are essentially reconfiguring the SPEC programs in ways that confuse the intent of the original benchmark. Some new processors are actually faster on double-precision floating-point math than single-precision. For those SPEC programs that use single-precision math, these vendors convert all data structures to the larger values.

The problem with this trick is that, for applications with large data arrays, doubling the size of these memory structures can penalize the user by forcing them to purchase extra main memory to maintain high performance. Users whose applications have large amounts of single-precision data know which SPEC programs share these characteristics and use these programs as a guideline; converting data to double-precision disguises the actual performance characteristics of the processor being benchmarked.

Unfortunately, such abuses have been common over the past year and have become accepted. It is too late to redefine SPEC92 to make these practices illegal. We recommend that the next-generation benchmark, dubbed SPEC94, rule out compiler flags that modify data types, which distort the intention of the original benchmarks, and unsafe flags, which can create runtime problems when used with some programs.

Furthermore, the new benchmark should require that all programs in the suite use the same compiler options. This will inform users how to configure the compiler to obtain maximum performance on a variety of programs. For users that take the time and have the expertise to play with compiler flags (primarily high-end scientific users and a few ISVs), a different set of benchmarks could be created for which anything goes.

Until then, SPEC92 continues to provide the best measurement of processor performance across a wide range of platforms and applications. The current results, however, are somewhat inflated, particularly for some individual SPEC components. SGI's John Mashey, one of SPEC's founders, likens excessive compiler tuning to measuring the performance of a typical automobile using a professional driver on a closed track burning rocket fuel. Hopefully, the SPEC committee will correct these excesses, so future measurements will reflect the performance achievable by you or me driving across town. ♦