Performance Nonmonotonicity Bugs

Nate Kushman and Volker Strumpen
Laboratory for Computer Science
Massachusetts Institute of Technology
{nkushman,strumpen}@supertech.lcs.mit.edu

August 14, 1998

We argue that future microprocessors should be designed with *performance monotinicity* as a basic design principle. Otherwise, compilers will not be able to cope with the increasing complexity of the hardware design, and nonmonotonicities will be exposed to software developers and users. We consider such anomalies to be unreasonable performance bugs.

We have identified several types of architectural performance bugs, the worst of which are *nondeterminisms* and *nonmonotonicities*. Whereas many non-monotonicities that have been identifed in the past are caused by the operating systems, these non-monotonicities are caused by the processor architectures. The most pronounced example of these anomalies was identified on the Sun UltraSPARC processor. We have found similar performance bugs on most state-of-the-art processor architectures, however.

When executing the program shown on the right on a Sun UltraSPARC, an execution time of 1.79 seconds can be observed. If we delete the call to `foo()` *before* the `for` loop, however, execution time varies nondeterministically between 4.77 and 7.15 seconds for repeated executions of the same executable. Thus, ***removing*** a tiny portion of work leads to an ***increase*** in execution time by a ***nondeterministic*** factor varying between 3 and 4.

```
void foo(void) { /* empty */ }

main()
{
  int i;

  foo();

  for (i=0; i<100000000; i++)
    foo();
}
```

We have developed a code restructurer for assembly code that inserts `nops` to avoid performance nonmonotonicity bugs in instruction schedules. With this tool, we increased the performance of the SPECint benchmarks by $2.2\,\%$ on average, and up to $9\,\%$ for individual benchmarks on the UltraSPARC.

Performance nonmonotonicities cause invalid results when experimenting on real architectures, potentially misleading the conclusions drawn from experimental research. Most of today's state-of-the-art systems have processors that exhibit such unreasonable performance behavior and compilers that do not conceal these nonmonotonicities from the user. If these nonmonotonicities and nondeterminisms remain undetected, they can easily cause a variety of problems not only for research and development, but for any performance critical application including real-time systems.

We argue that performance nonmonotonicities should not be exhibited to the programmer or user. Compilers or code restructuring tools should conceal such anomalies. However, as a result of the complexity of hardware designs, elaborate contracts are formulated which compiler writers must adhere to when scheduling instructions in order to generate high-performing codes. Unfortunately, these contracts are very complex due to a multitude of exceptional cases. Because of this difficultly, compilers are not adapted to fulfil the contracts. Hence, the processor architecture itself should be designed for performance monotonicity.