Andrew Zurn

Reading 9

Summarized Article: <http://web.mit.edu/newsoffice/2013/system-flags-useful-code-compilers-might-discard-1016.html?tmpl=component&print=1>

Other Article: <http://cacm.acm.org/magazines/2010/4/81482-sql-databases-v-nosql-databases/fulltext>

In his article, Larry Hardesty of the MIT News Office, takes a look at a compiler researched and developed by MIT that solves a very real problem that many in the larger software developing industry might experience. The problem that this compiler, dubbed “Stack,” is as he writes, “compilers [that] can be overaggressive, [which] dispense not only with functional code but also with code that actually performs vital security checks.” In other words, he gives an examples, pointing out that “when hundreds of developers work on an application with millions of lines of code that have been continually revised for decades, one of them may well end up inserting a seemingly innocuous condition that ensures that a function thousands of lines away, written by someone else, never gets executed.”

Compilers, over time, have gotten a lot more aggressive in optimizing code, and sometimes that optimization comes from cutting corners or using techniques that could lead to “undefined behavior”, that could lead to code that a developer thinks will execute and solve a problem, but will never actually run on the hardware. The way that Stack solves this problem, as Hardesty claims, is that it “compiles a program twice: once just looking to excise dead code and a second time to excise dead code and undefined behavior. Then it identifies all the code that was cut the second time but not the first and warns the programmer that it could pose a problem.” In sum, it will show the programmer what code in their program will be cut out by the more aggressive compilers.

Although I'm not surprised that some of these compilers have taken shortcuts to optimize code compilation times and also code execution times, I'd like to know exactly how often this happens in projects of varying sizes. I'm interested in seeing larger companies use this and how much of their actual source code is executed on there machines. This is definitely a fascinating problem, one that I'm sure will make some headlines and see integration if it does solve the problems that Hardesty proposes in this article.