Loop Optimizations - Introduction*

Charles Severance
Kevin Dowd

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

In nearly all high performance applications, loops are where the majority of the execution time is spent. In here[1] we examined ways in which application developers introduced clutter into loops, possibly slowing those loops down. In this chapter we focus on techniques used to improve the performance of these "clutter-free" loops. Sometimes the compiler is clever enough to generate the faster versions of the loops, and other times we have to do some rewriting of the loops ourselves to help the compiler.

It's important to remember that one compiler's performance enhancing modifications are another compiler's clutter. When you make modifications in the name of performance you must make sure you're helping by testing the performance with and without the modifications. Also, when you move to another architecture you need to make sure that any modifications aren't hindering performance. For this reason, you should choose your performance-related modifications wisely. You should also keep the original (simple) version of the code for testing on new architectures. Also if the benefit of the modification is small, you should probably keep the code in its most simple and clear form.

We look at a number of different loop optimization techniques, including:

- Loop unrolling
- Nested loop optimization
- Loop interchange
- Memory reference optimization
- Blocking
- Out-of-core solutions

Someday, it may be possible for a compiler to perform all these loop optimizations automatically. Typically loop unrolling is performed as part of the normal compiler optimizations. Other optimizations may have to be triggered using explicit compile-time options. As you contemplate making manual changes, look carefully at which of these optimizations can be done by the compiler. Also run some tests to determine if the compiler optimizations are as good as hand optimizations.

*Version 1.3: Aug 25, 2010 10:49 am -0500
†http://creativecommons.org/licenses/by/3.0/
[1]"Eliminating Clutter - Introduction" <http://cnx.org/content/m33720/latest/>