By Clay P. Breshears

OpenMP and Intel® Threading Tools

How Parallelism Can Be Used Effectively in a Serial Application

Note: *This white paper examines the use of the Intel Threading Tools—Intel Thread Checker and Intel Thread Profiler—to analyze OpenMP pragmas. The complete text, which contains code analysis, can be downloaded in its entirety from http://www.devx.com/go-parallel/Article/32724*

OpenMP programs are threaded programs and can suffer from the same errors and performance problems as explicitly threaded applications. For the purposes of this white paper, we assume you are familiar with OpenMP and so we delve into the ways you can use the Intel Threading Tools—Intel Thread Checker and Intel Thread Profiler—to analyze OpenMP programs.

Explicit threading methods, such as Windows threads or POSIX threads, use library calls to create, manage and synchronize threads. Use of explicit threads requires an almost complete restructuring of affected code. On the other hand, OpenMP is a set of pragmas, API functions and environment variables that enable you to incorporate threads into your applications at a relatively high level. Pragmas are used to denote regions in the code that can be run concurrently. An OpenMP-compliant compiler transforms the code and inserts the proper function calls to execute these regions in parallel. In most cases, the serial logic of the original code can be preserved and recovered by ignoring the OpenMP pragmas at compilation time.

Intel Thread Checker as Programming Assistant

Intel Thread Checker can determine if a section of code is a candidate for parallelization. With long code sections or those that have a deep call stack, it is extremely tedious and time-consuming to determine what dependencies exist within potential parallel loops. Dependencies such as an induction variable (a variable that is incremented at each iteration of a loop) or recurrence relation (accessing information computed on a previous loop iteration) inhibit correct parallelization without some algorithmic modifications to eliminate the dependence. Intel Thread Checker points out storage conflicts and, with examination of the code, you can confirm that the use of the noted variable does constitute a loop dependence. More often than not, some threads will need private copies of certain variables in order to avoid data races. In other cases, the logic of the program will be better served if access to some of these variables is synchronized.

Performance Tuning With Intel Thread Profiler

After a correctly threaded code has been created with the help of Intel Thread Checker, the performance of that code should be gauged. The serial and threaded execution times can be compared easily. If the threaded time is half that of the serial time when run with two threads on a dual-core system, you have done a great job implementing parallelism. If the threaded execution time is closer to (or even greater than) the serial time, certain questions arise. Do large segments of code still execute in serial? Does the required synchronization adversely affect the execution performance? Is the amount of work per thread properly balanced?

Thread Profiler for OpenMP can be used to answer these questions and guide the programmer to points within the code that could be improved and lead to better parallel performance. Because of the structured nature of OpenMP, Thread Profiler is able to make assumptions about the execution model for the application and point out very specific performance problems.

Summary

The Intel Threading Tools are often touted as being able to find threading errors in code quickly and point out performance bottlenecks that may not be apparent. By using these tools earlier in the development cycle, you can automate some of the more tedious tasks required to find where parallelism can be implemented effectively within a serial application.

For more details and information on using the Intel Threading Tools, see "Getting Started with the Intel Thread Checker" and "Getting Started with the Intel Thread Profiler," which are included in the threading tools documentation.