Lab 2: Analyzing Parallelism

Development Product Division

Legal Disclaimer & Optimization Notice

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Table of Contents

Lab 2: Finding Parallelism Issues

Time Required	Thirty minutes
Objective	In this lab session, you will use Intel® VTune™ Amplifier XE to determine the amount of parallelism in an application. After successfully completing this lab's activities, you will be able to: • Collect parallelism performance data for an application • Determine the amount of parallelism in an application

Activity 1 – Build the Application

Time Required	Ten minutes
Objective	• Build the application in preparation for finding its hotspot

1. Cd day1/lab5, make release

Activity 2 – Collect Parallelism Data

Time Required	Ten minutes
Objective	• Run the application while collecting parallelism data
Codes Description	• Tachyon is a 2-D raytracer/rendering program that displays an image

1. get **tachyon_analyze_locks executable.**
2. Select Menu > Tools > VTune Amplifier XE 2016 > New Analysis… or click on the "New Analysis" button in the VTune Amplifier XE tool bar.
3. Select "Algorithm Analysis->Concurrency" in the analysis type pane
4. Click "Start" – The tachyon application will run. Note that as the application runs it draws and image of several different silver balls on the screen. Notice the execution time displayed in the applications title bar immediately after the image is completely displayed. You will need this execution time in Lab 3
5. After the application completes the Intel® VTune™ Amplifier XE will spend some time analyzing the data. When it is finished analyzing, the summary pane appears. Note the analysis explanation pane comes up. Read it and then clear the pane.

   At this point the application has run to completion and the Intel® VTune™ Analyzer is ready to display the analyzed results.

Review Questions

Question 1: What is the result screen that appears after clearing the analysis explanation pane?

Question 2: What useful data is in this first screen?

Activity 3 – Analyze the app's Parallelism

Time Required	Twenty minutes
Objective	• Analyze the amount of parallelism in the application both as an overall average and also as the application runs.
Codes Description	• Tachyon is a 2-D raytracer/rendering program that displays an image

1. Click the "Bottom-up" tab.

   Notice the list of hotspot functions and the CPU utilization while executing of the functions.

   Notice the timeline view at the bottom of the screen. It shows the multiple threads that executed as it ran. There are a number of interesting things to notice.

   There is a very large amount of thread transition time as indicated by the large amount of yellow color in the top thread graph. It looks like the threads are spending a lot of time transitioning locks between them.
2. Zoom-in on a yellow portion of the graph by left-clicking and dragging over a 1 second portion of it. A dialog box appears - select "Zoom in on selection". Now you can see that there is no time in which both worker threads executed at the same time. Something is causing these 2 threads to "take turns" executing and to not execute at the same time!
3. Click on the "Undo Previous Zoom Selection" icon to get back to the original timeline view. Notice also that if you move the mouse pointer slowly over the Thread Concurrency graph at the bottom of the screen, the concurrency numbers are always around one. We are not getting any useful parallelism in this app.

   Notice also that there seems to be very little CPU usage or thread execution near the end of the program. This is the phase of the program in which it finished but kept the application windows visible so the user has time to see the overall execution time.

Review Questions

Question 1: Is this really a parallel application? Did it have more than 1 thread executing simultaneously at any time?