# Lab 1: Finding Hotspots

---

## 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 1: Finding Hotspots** ..... **i**

Development Product Division ..... i

**Disclaimer** ..... Error! Bookmark not defined.

**Lab 1: Finding Hotspots** ..... **1**

Activity 1 – Build the Application ..... 2

Activity 2 – Collect Performance Data ..... 3

Activity 3 – Find the Hotspot ..... 4

# Lab 1: Finding Hotspots

| | |
|---|---|
| **Time Required** | Forty Five minutes |
| **Objective** | In this lab session, you will use Intel® VTune™ Amplifier XE to find a performance hotspot in an application.<br><br>After successfully completing this lab's activities, you will be able to:<br><br>• Collect performance data for an application<br><br>• Determine an application's performance bottleneck<br><br>• Drill down to the source code of a hotspot |

# Activity 1 – Build the Application

| | |
|---|---|
| **Time Required** | Ten minutes |
| **Objective** | • Build the application in preparation for finding a hotspot |

1. Cd /home/intel-workshop/day1/lab5/tachyon_sample
2. **Make release**
3. **Check that tachyon_analyze_locks and tachyon_find_hotspots are built**

# Activity 2 – Collect Performance Data

| | |
|---|---|
| **Time Required** | Fifteen minutes |
| **Objective** | • Run the application while collecting performance data |

1. Run putty, start vncserver; open vncviewer;
2. set Amplifier environment . /opt/intel/vtune_amplifier/amplxe-vars.sh
3. Run vtune Amplifier: amplxe-gui
4. Select Menu > Tools > VTune Amplifier XE 2016 > New Analysis… or click on the "New Analysis" button  in the VTune Amplifier XE tool bar.
5. Select "Algorithm Analysis->Basic Hotspots" in the analysis type pane
6. 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 application's title bar immediately after the image is completely displayed.

   Note: do not switch between windows while the image rendering is performed.
7. After the application completes the Intel® VTune™ Amplifier XE will spend some time analyzing the data. When it is finished analyzing, the Hotspots 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™ Amplifier XE 2016 displays the analyzed results.

## Review Questions

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

Question 2: Which function used the most CPU time?

# Activity 3 – Find the Hotspot

| | |
|---|---|
| **Time Required** | Twenty minutes |
| **Objective** | • Find the source code for a performance hotspot<br>• Identify the calling sequence into the performance hotspot that generated the most CPU time |

1. Make sure the "Bottom-up" tab is highlighted. The functions in the tachyon program will be listed in order of execution time. Notice the Timeline View at the bottom of the screen. This shows the various states of the threads in the program and the number of CPUs used as the program ran. 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 window visible so the user has time to see the overall execution time. Notice also that there is only 1 thread shown with any significant execution time.
2. Go back to looking at the function list at the top of the result screen. The functions grid_intersect and sphere_intersect will be at the top of the list, but there is another function that seems to have used a surprisingly large amount of CPU time given what it does: initialize_2D_buffer.
3. Click on the  sign box to the left of the function name initialize_2D_buffer. Notice the different calling sequences into that function, and to see the relative amounts of execution time generated by those calling sequences. In this case there appears to be only 1.
4. Double click on the function name initialize_2D_buffer. The source code for that function is displayed at the hottest point in that function along with assembly code to the right. Each source and assembly statement is annotated with execution time on the right.

   Vertical panes to the right of the source and assembly vertical scroll bars show relative position and density of execution time throughout the view. Scroll up and down to see that.
5. Note the comments surrounding the nested for loops in lines 78-88 that contain the statement that consumed the most CPU time, line numbers 83-86.

   To show the use of the hotspot collector the source code has 2 different ways of explicitly initializing the code. One referencing sequential memory locations (the "faster" method) and one using a slower, non-sequential method).
6. Comment out a slower version in src/linux/find_hotspots/find_hotspots.cpp
7. Rerun VTune Amplifier's Hotspots profiling mode. After it runs and the tool shows the results, notice that "initialize_2D_buffer" is much further down the function list and took less time.

## Review Questions

1. How much faster was the modified application?
2. At what point in the program does the CPU usage drop off?
3. Which function took the most time in the optimized version?