CS 420, Parallel Programming, Fall 2019
MP2

Szaday, Justin Shah, Ishan Pankaj

Due Date: October 2nd, 2019, 11:59PM CDT

1 Overview

Effective vectorization is necessary to achieve good performance on modern machines. The purpose of this assignment is two-fold, for you to gain experience with vector intrinsics and compare hand-vectorized code to auto-vectorized code. To this end, you will be asked to hand-vectorize your implementation of tiled matrix-matrix multiplication from MP1 and analyze its performance with VTune.

2 Tasks

2.1 Getting Started

Pull the skeleton for MP2 from the `_release` repository:
https://github-dev.cs.illinois.edu/cs420-fa19/_release

2.2 Part A, Vectorization with Intrinsics

2.2.1 Your Task

Using the AVX2 vector intrinsics discussed in class, vectorize your implementation of tiled matrix-matrix multiplication from MP1. For this assignment:

- You must not use scalar loads, stores, or floating-point operations.
- You may use any non-scalar intrinsics, e.g. packed instructions and broadcasts.
- You may use unaligned or aligned loads or stores.
- You may assume all tile sizes are divisible by the vector length. (You do not need to enforce this.)

2.2.2 Hints

You may find the following intrinsics helpful for this assignment:

- `_mm256_broadcast_ss`
- `_mm256_load_ps`

- `_mm256_store_ps`
- `_mm256_add_ps`
- `_mm256_mul_ps`

If your code fails with a segmentation fault and you are using aligned loads and/or stores, you may need to switch to unaligned memory accesses (e.g. `_mm256_storeu_ps`) or use an aligned version of `malloc` (e.g. `_mm_malloc`). Do not forget to include `immintrin.h` in your code. The Intel guide to vector intrinsics can be found at: https://software.intel.com/sites/landingpage/IntrinsicsGuide/

2.3 Part B, Profiling with VTune

2.3.1 Hotspots Analysis with VTune

In MP1, we used VTune's `memory-access` analysis to gain insights about our program's memory behaviors. This time, we will be using its `hotspots` analysis to gain insights about the *hotspots* in our program, or the sections of code that make up the bulk of our program's execution time. To use the `hotspots` analysis, run the following:

```
module load intel/.19.0
amplxe-cl -collect hotspots <RunCommand>
```

NOTE: Please ensure that your program is compiled with `-g` to be able to correctly view the results in VTune's GUI.

2.3.2 Loading Results in VTune's GUI

Due to the limitations of VTune's command-line interface, we will be using its graphical user interface (GUI) to view the hotspots data collected from our program. Therefore, you will need to connect to the campus cluster with an X11-enabled SSH client (e.g. MobaXTerm on Windows). To start, launch an interactive `qsub` session with X11-forwarding enabled (do not attempt to use VTune's GUI from the login nodes, as it would be lag-heavy):

```
qsub -q cs -X -I
```

After you enter the command, you will have to wait for Torque to start the job. Once the job starts, you will be presented with an interactive shell prompt on the launch node. Next, to launch VTune's GUI, run the following commands:

```
module load intel/.19.0
amplxe-gui --path-to-open <.amplxe file>
```

Replace `<.amplxe file>` with a hotspots results file collected from your matmul program (found in a VTune results directory), e.g. `r001hs/r001hs.amplxe`. Once the GUI has loaded, click the *Search Source/Binaries* button under the *Collection Log* tab and add the directory containing your `matrix` executable and source files to both the *Binaries/Symbols* and *Sources* panes. Next, click the *Re-resolve* button in the bottom-left-hand corner of the *Collection Log* tab (the icon is a circular arrow with a partially dashed tail); this will finalize the results and present you with the *Summary* tab. NOTE: If you cannot reserve an interactive session on the campus-cluster, you might consider installing VTune on your personal machine, copying the results files from the campus-cluster to view them locally.

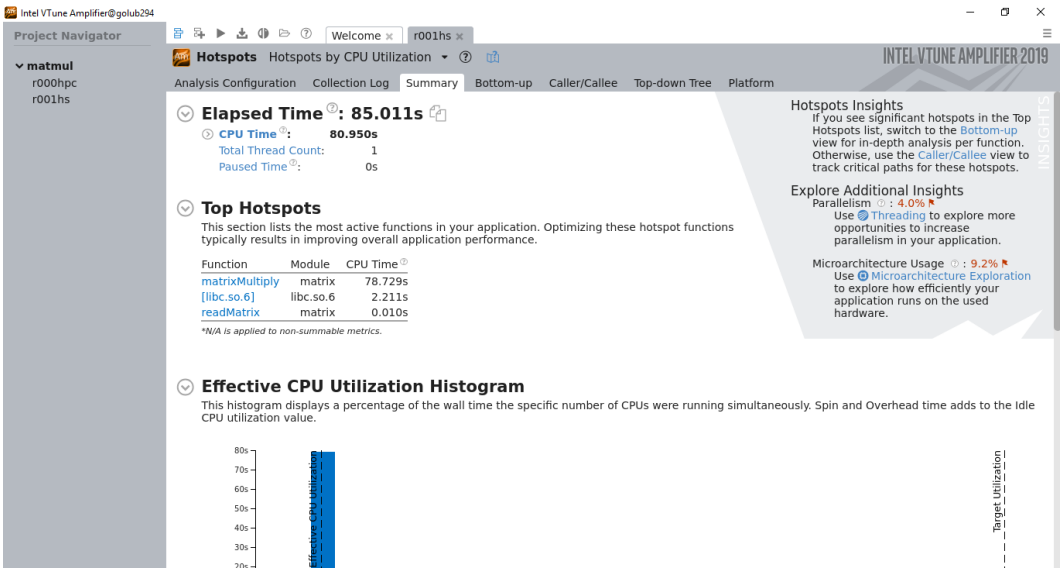

Figure 1: Example Hotspots Summary Tab

2.3.3 Viewing Results in VTune's GUI

An example of the hotspots summary tab is shown in Figure 1. To view the details for a hotspot, click a hotspot under the *Top Hotspots* list, then double-click the highlighted entry under the *Bottom-up* tab. This will open the source file under a new tab, and you can click the *Assembly* button to display the program's assembly as well, as shown in Figure 2. This view also shows the percentage of time spent running each line of code in the hotspot.

2.3.4 Your Task

Run VTune's hotspots analysis on your tiled matrix-matrix multiplication program with (aka the version from MP1) and without vector intrinsics, for matrix sizes $2000x2000$ with tile size 100. Answer the following questions in your report:

? For the version of matmul without vector intrinsics, did the compiler generate packed vector instructions? If so, pick one of the packed vector instructions it generated and briefly explain what it does. Hint: Consult the assembly tab.

? Which of the vector intrinsics in your hand-vectorized program dominated your program's execution time? Briefly explain why and include a screenshot of your hand-vectorized program's top hotspot like Figure 2 in your report.

? How did the performance of your hand-vectorized version compare to the original version? Include the MFLOPs rates for both versions and briefly comment about whether or not you think the compiler's auto-vectorization was adequate and why.

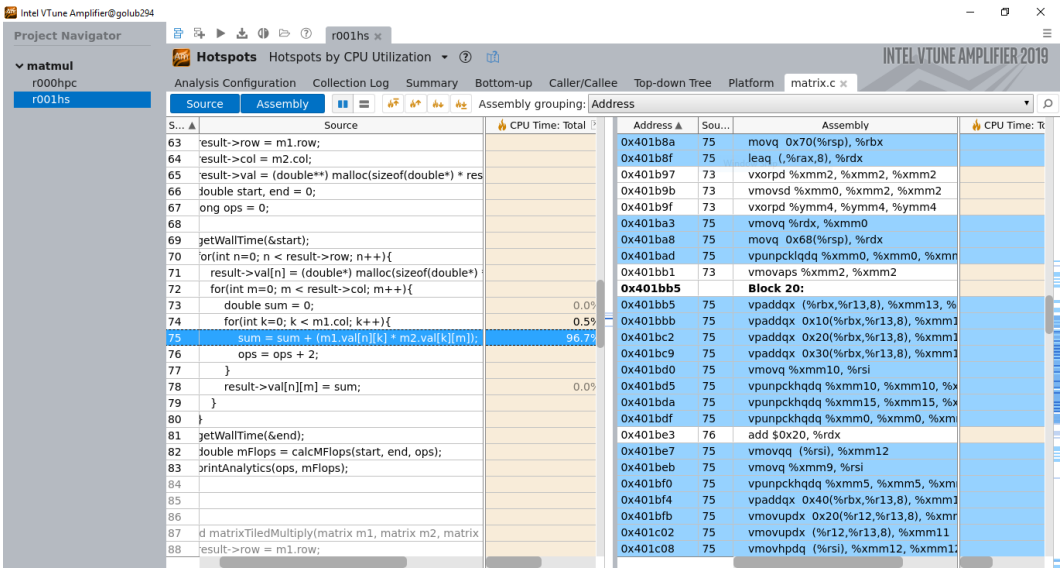

Figure 2: Viewing Source/Assembly with VTune

3 Part C, LU Decomposition (4 Credits Only)

3.1 Your Task

To gain more experience with vector intrinsics, students taking this course for 4-credits are to vectorize the implementation of *LU Decomposition* included in `matrix.c`; an algorithm that accepts a square matrix and generates an upper and a lower triangular matrix whose product is the provided matrix. Like before, you are free to use any packed vector intrinsics for this part of the assignment, but you may not use any scalar floating point operations. This portion of the assignment will be graded based on your use of intrinsics and implementation's correctness – you need not worry about its performance.

4 Submission Guidelines

4.1 Expectations For This Assignment

The graded portions of this assignment are your hand-vectorized matrix-matrix program and answers to the short-answer questions.

4.1.1 Points Breakdown

- $3 * 15pts.$ — Each Short Answer Question
- $5pts.$ — VTune Screenshot
- $50pts.$ — Hand-Vectorized Tiled Matmul Program
- $50pts.$ — Hand-Vectorized LU Decomposition (4 cr. only)

4.2 Pushing Your Submission

Follow the git commands to commit and push your changes to the remote repo. Ensure that your files are visible on the GitHub web interface, your work will not be graded otherwise. Only the most recent commit before the deadline will be graded.