Viva la Convolution!

Dr. Richard M. Veras

October 5, 2023

Overview: The purpose of this lab is to dive into the interactions of the memory hierarchy, data layout, loop transformations, and performance. You will be provided correct, unoptimized, baseline implementations of these three operations, along with the appropriate testing and timing harnesses.

Your primary objective will be to iterate and create multiple variants of these operations using the concepts learned in class. From those iterations you will select a specified number variants for each operation and provided a writeup that sumarizes the results and the relationship to the material covered.

You will create multiple implementations of the operation. The number of which is not as important as having implementations that hit the requirements. However, if you are aiming for the performance bonus, more is better.

What I want to see at least one variant that demonstrates the following loop transformations. (In the report these will be represented by a plot, and diagrams describing how the flow of execution has changed with the transformation):

1. Loop Unswitching (Hint: there is a modulo operator that I want you to eliminate. This will create a handful of if-conditions that can be unswitched).
2. Index Set Splitting (Hint: This will be another trick to get rid of the modulo)
3. Loop Fission
4. Loop Interchange

In the report create a plot that compares these four transformations. Examples of these transformations can be found in the Fall 2022 slides for lecture 14 and 15, as well as the loop homework (to be released).

I want you to do the following loop transformations that have a degree of tuning (In the report each of these will have a plot):

1. Loop Unrolling - I want you to vary the unrolling factor and create a plot that compares performance of using different unrolling factors.
2. Loop blocking - I want you to vary the block size and create a plot that compares the performance of different block sizes.

For both of these you are sweeping through some factor. This will require you to run the timer for each parameter selected. You can modify the timer and use code generation to automate this.

You will demonstrate combinations of these loop transformations and plot their performance:

1. The use of two of the previous loop transformation.

2. The use of two different loop transformations.

For demonstrating parallelism you will do the following (You will have one plot comparing all four, along with 2-3 separate plots for sweeping through numbers of a parallel elements):

1. Demonstrate the use of Instruction Level Parallelism.
2. Parallelize the code using SIMD operations (on Schooner you might have access to AVX512).
3. Parallelize the code using OpenMP for shared memory parallelism (Create a separate plot that compares the performance of using a different number of threads).
4. Parallelize the code using MPI Send and Receives for distributed memory parallelism (Create a separate plot that compares the performance of using a different number of nodes/ranks).
5. Parallelize the code using MPI Collective Communications – such as broadcast, reduce, gather, scatter, etc. – for distributed memory parallelism (Create a separate plot that compares the performance of using a different number of nodes/ranks. This will require modifying your sbatch file).

Note that you will likely use loop transformations to expose opportunities for parallelism. Additionally, see Pacheco's book, along with online resources, for using MPI for distributed memory parallelism and OpenMP for shared memory parallelism.

Last, I want you to combine multiple forms of parallelism (This will be one plot and will include your best implementations, you may also fuse these results with the individual parallelism performance).

1. Demonstrate the use of two forms of parallism in the same code.
2. Demonstrate the use of three forms of parallelism in the same code.
3. Demonstrate the use of four forms of parallelism (ILP,SIMD,Shared Memory, Distributed Memory) in the same code.

Note that a variant may demonstrate more than one topic, and a topic can be represented by multiple variants. The top five teams with the fastest implementation will receive bonus points. Specifically, this measurement wil be run on the same machine (gpel), with the metric being the greatest area under the curve for from several elements to a problem size large enough to no longer fit in the last level of cache.

The assumption is that no significant prior knowledge of low-level hardware details is necessary. The expectation is that you acquire this knowledge through reference manuals and more importantly through practice. The exercise of iterating over the code allows you to test your hunches and refine your ideas. Do not copy outside code, because that is plagairism.

Deliverables: Your team will be responsible for pushing several key deliverables to a private GitLab (not github) repository. The name of your repository will be of the form *pdn_SEM_YR_lab00_TEAM*, where SEM is the semester (fa or sp), YR is the year and TEAM is your team name. You will grant maintainer access to the instructor and TA.

1. `sow_timeline.txt`: What is the work to be done, how the work will be divided and a tentative timeline that you and your team will follow? This is a contract between you and your teammates, so please complete this before anything else.

2. `minutes_and_logs.pdf`: Every time your team meets (either synchronously or asynchronously) keep a written record of the discussion. Feel free to include git commit activity/logs. Note, this is not a recording or the number of minutes spent on a task.

3. Code:

   `test_varXXX.c` These are your team's implementations of the target operations. You will have a variety of these to demonstrate the concepts listed in the overview and those covered in class.

4. `writeup.pdf`: This is a 2-4 page analysis of your work, and I strongly suggest you use overleaf with the IEEE two column article class. The writeup should be a well written document that includes performances plots for the variants in addition to the following sections:

   Overview What is the problem you are trying to solve? What are the pain points in this problem? How do you plan to solve this? And how does this relate back to the material covered in class.

   Refinements For each variant of answer the following questions: What did you change between each variant? What should we expect to see in the results and why? Include performance plots, what do the results show? What interesting features should we note in the plot? What would you do to improve these result? Diagrams, figures, and pictures can help drive your point.

5. `selfassessment.txt`: In this file perform a postmortem. What was the work that was done? How was it divided? How would you distribute the points amongst your teammates (50/50, 60/40, 33/33/33?). How did your original statement of work line up with what was done? How did it evolve? How much time (and/or effort) did this take? How was the time (and/or effort) distributed across the task and team members? What would you do differently next time? This can be answered as a team or individually. With the point distribution, there is a small consensus bonus if the team agrees on the point distribution. If the team does agreee on a distribution then the average of the distributions will be used. Lastly, include the total number of person-hours spent on this lab.

If there any any issues with this document or the provided code, please let me know so I can update it.