CS 4473/5473: PDN Programming

Dr. Richard M. Veras

Transposing Tensors for Fun and Profit

Overview: Tensor (matrix) Transposition is a significant bottleneck in many critical tasks in areas of machine learning, signal processing, and data analytics. Large commitments of resources and person-hours are routinely thrown at this problem for even the smallest of speedups. 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 summarizes the results and the relationship to the material covered in the other course activities.

The required tasks will be detailed in a task list, but the main objectives are as follows:

A. A dispatcher based on the strides of the source and destination matrix. You will dispatch to the cross product of row-major, column-major, and general stride ordering of the two matrices. This will lead to nine paths! Isolating these paths will allow you to specialize your optimizations later on. Hint: The trick is there is plenty of symmetry to exploit here and these nine paths reduce to only a few outcomes. Additionally, for some paths there is not much you can really do. If done right you will be able to reuse a lot of your tuned code effectively for each case. You will likely have refinements in your logic!
B. A variant that computes transposition as matrix transpositions by blocks. You can think of this as being recursive, or as blocking (or tiling) the loops. In this you will have one level of tiling (a matrix of matrices).
C. An analysis of different block sizes for B. This can be refined if you do this tuning on refined versions of B. You are trying to find out good block sizes.
D. Use of two levels of tiling. I.e. transposition of a matrix of blocked matrices. If no blocking requires two loops, one level of tiling (B) will take 4 loops, and two levels of tiling will take 6 loops.
E. An analysis of different block sizes for D. This can be refined if you do this tuning on refined versions of D. You are trying to find out good block sizes. In the past I used to have teams write code that automates this analysis by generating and running the code under test.
F. Refine the dispatch based to leverage problem size. You can further refine this as you create more variants.
G. Minimizing memory traffic through loop restructuring. How do you traverse your data in a way that makes the most effect use of the cache? PG2 provides the context for this. For this you will need to look up the CPU information of the gpel machines and collect the details (size and set associativity) for each level of cache. CPU World is invaluable here (https://www.cpu-world.com/).

H. Use of a Single Instruction Multiple Data (SIMD) based kernel. You will cover this in PG3 and it will give you a 4-8x improvement in performance. You can further refine by creating more kernels for different small problem sizes.
I. Use of shared memory parallelism (OpenMP). In your report you will have a scalability plot that compares the performance of 1,2,4,8, etc. threads. Use enough threads such that performance tanks, then stop.

The top five teams with the fastest implementation will receive bonus points. Specifically, this measurement will 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. You must provide a plot and code of the variant you will use for the bonus. If there is a TA implementation and your implementation outperforms it, there will be an additional be additional points.

The assumption of this lab 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. It is likely that you might want to modify the infrastructure code. This is acceptable, provided you are not attempting to game the assignment.

Do not copy outside code, because that is plagiarism. Additionally, do not use generative AI models (including copilot) for the code or writeup. This will be flagged and reported. See the syllabus.

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's name. You will grant maintainer access to both the instructor and TA).

1. sow timeline.pdf: 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. You will then update this document by appending updated versions of it.
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 of the number of minutes spent on a task.
3. **Code**: The majority, but not all, of your changes will be in the creation of new variants (test varXXX.c). In addition to sharing maintainer access to your repository, you will submit a tarball of your code.
4. writeup.pdf: This is a 2-4 page analysis of your work. If more space if needed (for example to explain additional refinements) you can follow this main matter with an appendix section. This will be written in LaTeX (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 your degree.

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 not agree on a distribution, the average will be used.
6. **Tasks and rubric:** This will be a spreadsheet that tracks all of the components you're your team completed

Additionally, weekly you will submit a checkpoint. It is sufficient to include the most up-to-date copy of items 1, 2 and 6, along with any questions that you have.

Grading: Individual scores will be calculated as a sum of a team pot that is scaled per individual by attendance and a personal pot that is total points scaled by your percentage.

If there are any issues with this document or the provided code, please let me know so I can update it. There will be frequent updates to this lab and it is your responsibility to keep track of updates on canvas.