CS348K: Visual Computing Systems
Class/Reading Response Template

Reminder: please make sure the PDF you submit to Gradescope DOES NOT have your name on it. We will concatenate all responses and give everyone in the class a PDF of all responses.

Part 1: Top N takeaways from discussions in the last class. Note: this part of the response is unrelated to the current reading, but should pertain to the discussion of the prior reading in class (or just discussion in the class in general, if there was no reading):

- What was the most surprising/interesting thing you learned?
    - I think global tone mapping is really impressive, and HDR can be use to solve the problem of too dark or too bright in global tone mapping and display a wider range of brightness levels. From last class I understand what HDR can do but I'm unaware of the origin of the problem.
- Is there anything you feel passionate about (agreed with, disagreed with?) that you want to react to?
    - Merging images. This is mentioned in HDR paper, but is not as what mentioned in class. In HDR, during the merging process, the tone-mapping algorithm may adjust the brightness and contrast of the images in order to produce an image that is more realistic and detailed, so it does not involve blurring the image.
- Did class cause you to do any additional reading on your own? If so, what did you learn?
    - I read more about TVM because of Halide. These papers are great but kind of difficult for me. I think a main difference is that Halide's scheduling system is more flexible. Halide allows the programmer to specify complex loop fusion and tiling strategies, while TVM doesn't.
- Major takeaways in general?
    - Summary of an image pipeline. Before this class I don't know how those algorithms mentioned in previous lectures could contribute for an image pipeline.

Part 2: Answers/reactions to instructor's specific prompts for this reading. (Please see course website for prompts).

In reading this paper, I want you to specifically focus on describing the philosophy of Halide. Specifically, if we ignore the "autotuner" described in Section 5 of the paper (you can skip this section if you wish), what is the role of the Halide programmer, and what is the role of the Halide system/compiler?
Hint 1: Which component is responsible for major optimization decisions?
Hint 2: Can a change to a schedule change the output of a Halide program?

The philosophy of Halide is to to make it easier for developers to write high-performance image processing code. The key idea of Halide is to separate the algorithm of image processing from its execution on a specific architecture, so that the algorithm can be optimized for different hardware. Also, Halide privides lots of techniques for optimization.

The role of the Halide programmer is to define the algorithm in Halide.

The role of the Halide system/compiler is to transform the Halide code into an efficient implementation for the specific architecture, using techniques such as loop unrolling, tiling, vectorization, and scheduling.

Let's consider what type of programm Halide provides the most value for. Again, ignoring the autoscheduler, what class of programmer do you think is the target of Halide? Novices? Experts? CS149 students? Why?

I think Halide provides the most value for CS149 students because they have learned techniques for blocking and scheduling to enhance locality and improve performance. However, they know very little about what the compiler does to help. For experts, they already possess comprehensive knowledge, so using Halide is a good option but not a necessity. As for novices, they may have difficulties writing reasonable code and schedulers, so Halide may not be of much assistance.

In your own words, in two sentences or less, attempt to summarize what you think is the most important idea in the design of Halide?

Halide provides a domain-specific language for image processing that separates the algorithm from execution, and it explores various optimization strategies while maintaining correctness and reproducibility across platforms.

Advanced question: In my opinion, there is one major place where the core design philosophy of Halide is violated. It is described in Section 4.3 in the paper, but is more clearly described in Section 8.3 of the Ph.D. thesis. (see sliding window optimizations and storage folding). Why do you think am I claiming this compiler optimization is a significant departure from the core principles of Halide? (there are also valid arguments against my opinion.)
Hint: what aspects of the program's execution is not explicitly described in the schedule in these situations?

It violates the design philosophy of separating algorithm and execution because it is related to the memory management strategy. The sliding window optimization and storage folding introduce execution details into the algorithm by specifying how data is accessed and stored in memory.

Part 3: [Optional] Questions I'd like to have specifically addressed via in class. (We also encourage you to just post these questions on Ed immediately so anyone can answer!)