Naio CUDA Developer Interview Questions

Question: CUDA Image Processing Performance Optimization

You are tasked with processing a grayscale image of size `1024 x 1024` pixels using CUDA. Each pixel is represented as a `float`, and the operation you need to perform is an element-wise transformation of the image where each pixel's value is squared.

Part 1: 1D Kernel Implementation
Implement a CUDA kernel that treats the image as a 1D array (of size `1024 x 1024` = `1048576` elements) and processes each pixel in parallel. Write the kernel and the corresponding kernel launch configuration.

Part 2: 2D Kernel Implementation
Now, rewrite the kernel to process the image using a 2D grid and block configuration, with each thread handling a specific `(x, y)` pixel. Write the 2D kernel and the corresponding kernel launch configuration.

Part 3: Performance Consideration
Compare the two approaches in terms of:

- Hardware optimizations
- Scalability to larger images (e.g., `3848 x 2160`)

Which approach would you recommend for efficient image processing on larger datasets, and why?

Bonus: Optimization (Optional)
What optimizations could be to further improve performance?

Numpy to CUDA translation question:

Part 1: CUDA Implementation

Translate the following Python NumPy code into a C++ program that uses CUDA. Your solution should:

1. Implement **matrix multiplication** using a CUDA kernel.
2. Implement an element-wise **sigmoid** transformation using another CUDA kernel.

The Python code is as follows:

```python
import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def matrix_operations(A, B):
    # Perform matrix multiplication
    C = np.dot(A, B)

    # Apply element-wise transformation: sigmoid function
    C_transformed = sigmoid(C)

    return C_transformed

A = [[4, 2], [7, 5]]
B = [[8, 6], [9, 1]]
```

Bonus: Testing (Optional)

Write a test to verify the correctness of your CUDA implementation. You can use any testing framework (e.g., Catch2) or simple assertions to compare the CUDA results with the expected results from the original Python code.