SAXPY Fine-Grained CUDA Benchmarking - Full Conversation Transcript

Full Technical Transcript

[Conversation Transcript: SAXPY CUDA Benchmarking with Timing Breakdown and Visualization]

1. User asked how to run CUDA code in Google Colab.
-> I explained enabling GPU in Colab, using CuPy for easy GPU usage, and compiling `.cu` files with nvcc.

2. User requested CUDA SAXPY code that measures execution time using cudaEvent.
-> I provided a SAXPY kernel and host code measuring execution time.

3. User asked to loop N from 2^16 to 2^30 and log each kernel timing.
-> I updated the code to sweep over N, measure time, and print results.

4. User noted suspicious results (constant time, max error = 2).
-> I diagnosed that the kernel wasn't actually running due to PTX error.

5. User ran `nvidia-smi` and showed they had an RTX 2080 and CUDA 12.4.
-> I identified correct compute capability: `sm_75` and gave the proper `nvcc` command.

6. User recompiled with `-arch=sm_75` and confirmed results looked better.
-> Execution time increased with N, max error = 0.000000.

7. User asked to log results to CSV and visualize them.
-> I added CSV logging and gave a Python script for bar graph.

8. User posted the execution time plot.
-> I confirmed it was accurate and readable, suggested dual y-axis or log scale if needed.

9. User asked to add effective bandwidth (GB/s) to the plot.
-> I added bandwidth computation and dual-y axis plot for timing and bandwidth.

10. User shared the updated dual-y plot.
-> I confirmed it looked great and offered ideas for further improvements.

11. User asked to measure both kernel-only and total time (allocation + kernel).
-> I added cudaEvent for kernel and std::chrono for total timing in the code.

12. User asked for a dual timing comparison plot.
-> I provided a grouped bar plot comparing kernel and total time, and noted kernel bars might be hard to see.

13. User shared the dual bar plot and noted the issue.
-> I explained the scale problem and gave log-y version + stacked bar alternative.

SAXPY Fine-Grained CUDA Benchmarking - Full Conversation Transcript

14. User asked to break down timings for each step.
-> I provided updated CUDA code measuring: malloc, cudaMalloc, H2D copy, kernel, D2H copy, and free separately.

15. User requested a Python plot for the fine-grained breakdown.
-> I provided a stacked bar chart with log-y axis to visualize all timing components.

This conversation included:
- CUDA kernel timing (cudaEvent)
- Total execution time profiling
- Memory transfer measurement
- CSV logging and automated bar chart generation
- Effective bandwidth calculation
- Stacked timing breakdown visualization

Each stage of the SAXPY algorithm is now fully timed, logged, and visualized.

[End of Transcript]