graph_cuda

August 11, 2025

```
[8]: # !pip install numpy matplotlib pandas tqdm
```

```
[9]: import json

     import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

```
[10]: with open("metrics_cuda.json", encoding="utf-8") as f:
          data = json.load(f)
```

```
[11]: df = pd.DataFrame(data)
      # df["size"] = df["count"] * df["block_size"]
      df.sort_values("duration", ascending=False)
```

```
[11]:           size  duration  value  block_size  grid_size         runtime device
      0    100000000  3.676688    0.0           1          1  CUDA Reduction   TODO
      1    100000000  2.998379    0.0           2          1  CUDA Reduction   TODO
      11   100000000  2.968538    0.0           1          2  CUDA Reduction   TODO
      2    100000000  2.075491    0.0           4          1  CUDA Reduction   TODO
      12   100000000  2.071986    0.0           2          2  CUDA Reduction   TODO
      ..          …         …     …           …          …               …      …
      120  100000000  0.274478    0.0        1024       1024  CUDA Reduction   TODO
      119  100000000  0.274084    0.0         512       1024  CUDA Reduction   TODO
      92   100000000  0.272847    0.0          16        256  CUDA Reduction   TODO
      42   100000000  0.272365    0.0         512          8  CUDA Reduction   TODO
      84   100000000  0.271958    0.0         128        128  CUDA Reduction   TODO

      [121 rows x 7 columns]
```

```
[14]: runtimes = df["runtime"].unique()

      for target_runtime in runtimes:
          # target_runtime = 'OpenCL Reduction'
          df_filtered = df[df['runtime'] == target_runtime]

          block_sizes = sorted(df_filtered['block_size'].unique())
          counts = sorted(df_filtered['grid_size'].unique())
```

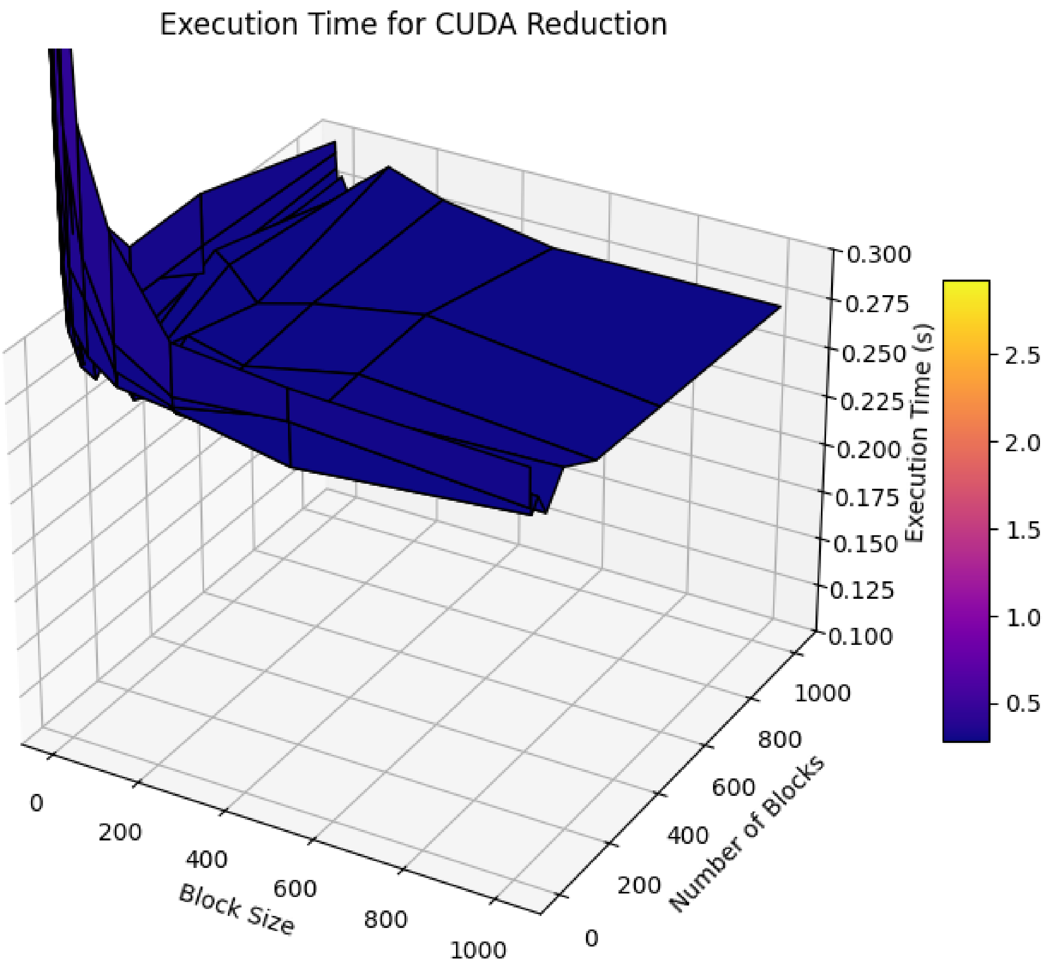
```
X, Y = np.meshgrid(block_sizes, counts)
Z = np.zeros_like(X, dtype=float)

for i, count in enumerate(counts):
    for j, block in enumerate(block_sizes):
        match = df_filtered[
            (df_filtered['block_size'] == block) &
            (df_filtered['grid_size'] == count)
        ]
        if not match.empty:
            Z[i, j] = match['duration'].values[0]
        else:
            Z[i, j] = np.nan

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, Y, Z, cmap='plasma', edgecolor='k')

ax.set_xlabel('Block Size')
ax.set_ylabel('Number of Blocks')
ax.set_zbound(0.1, 0.3)
ax.set_zlabel('Execution Time (s)')
ax.set_title(f'Execution Time for {target_runtime}')
fig.colorbar(surf, shrink=0.5, aspect=10)

plt.show()
```


Execution Time for CUDA Reduction
Block Size
0
200
400
600
800
1000
Number of Blocks
0
200
400
600
800
1000
Execution Time (s)
0.100
0.125
0.150
0.175
0.200
0.225
0.250
0.275
0.300
0.5
1.0
1.5
2.0
2.5