graph_pc

August 9, 2025

[1]:

```
# !pip install numpy matplotlib pandas tqdm
```

[2]:

```
import json

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

[3]:

```
with open("metrics_pc.json", encoding="utf-8") as f:
    data = json.load(f)
```

[4]:

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

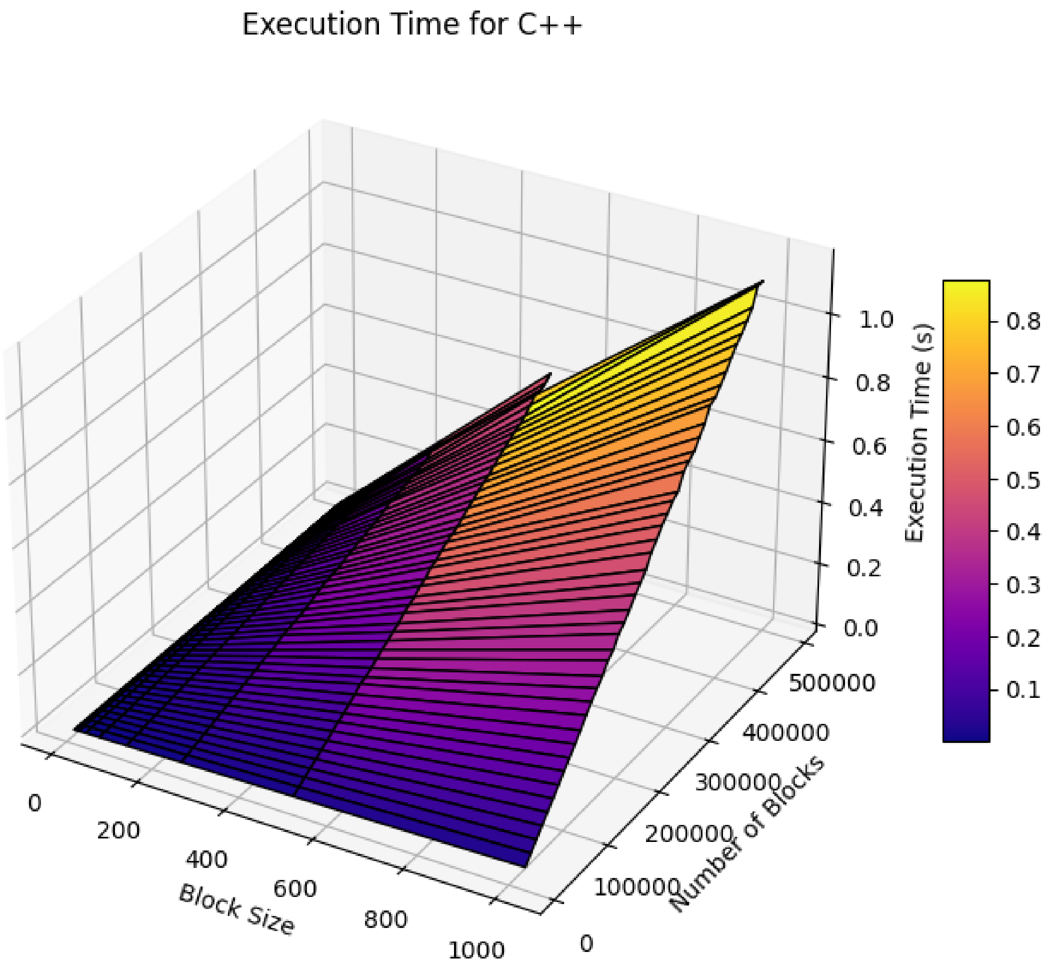
[4]:

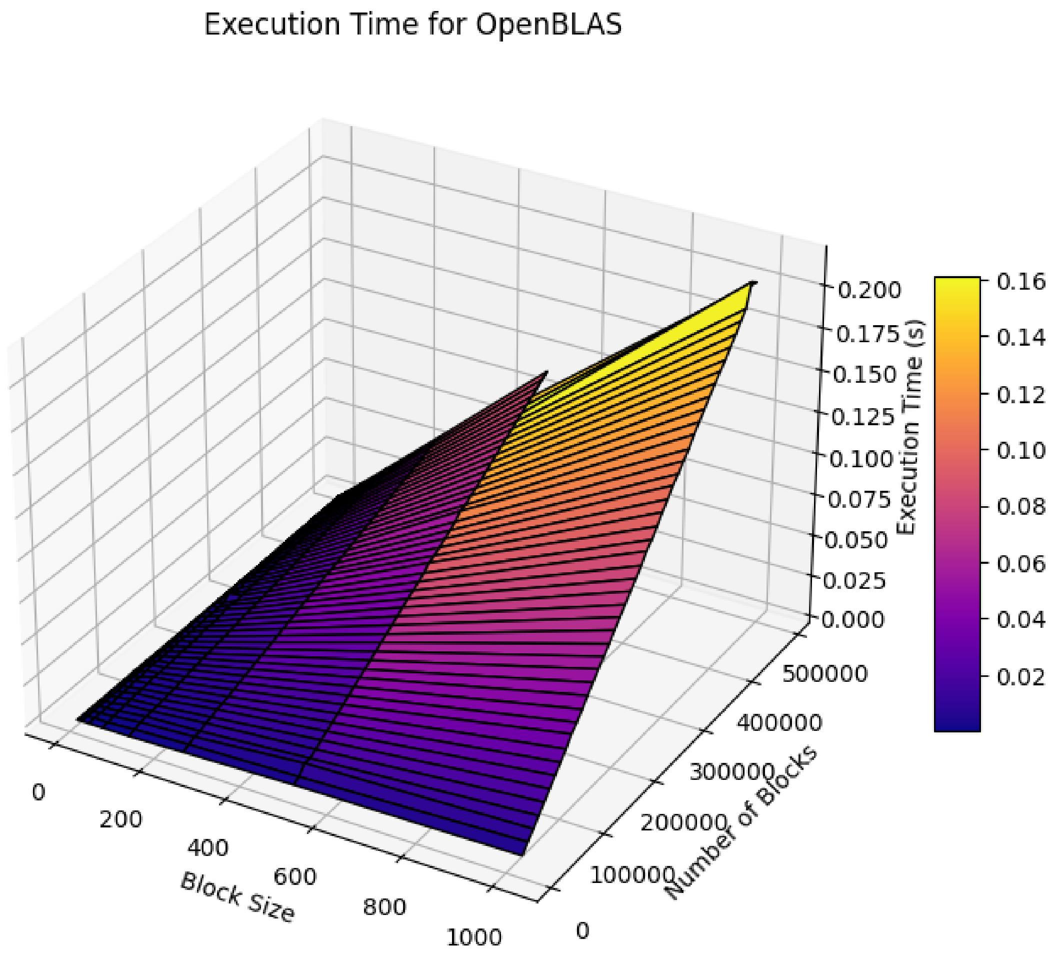
```
      duration     value  block_size   count   runtime  \
1616  1.174223  0.250000        1024  450000       C++
1580  1.173617  0.250000        1024  440000       C++
1544  1.120909  0.250000        1024  430000       C++
1508  1.093320  0.250000        1024  420000       C++
1472  1.067841  0.250000        1024  410000       C++
…          …         …         …       …        …
9     0.000019  0.000532          16   10000  OpenBLAS
109   0.000019  0.000532           4   40000  OpenBLAS
41    0.000017  0.000532           8   20000  OpenBLAS
37    0.000011  0.000266           4   20000  OpenBLAS
1     0.000007  0.000133           4   10000  OpenBLAS

                       device       size
1616  NVIDIA GeForce RTX 2060  460800000
1580  NVIDIA GeForce RTX 2060  450560000
1544  NVIDIA GeForce RTX 2060  440320000
1508  NVIDIA GeForce RTX 2060  430080000
1472  NVIDIA GeForce RTX 2060  419840000
…                         …         …
9     NVIDIA GeForce RTX 2060     160000
109   NVIDIA GeForce RTX 2060     160000
41    NVIDIA GeForce RTX 2060     160000
```

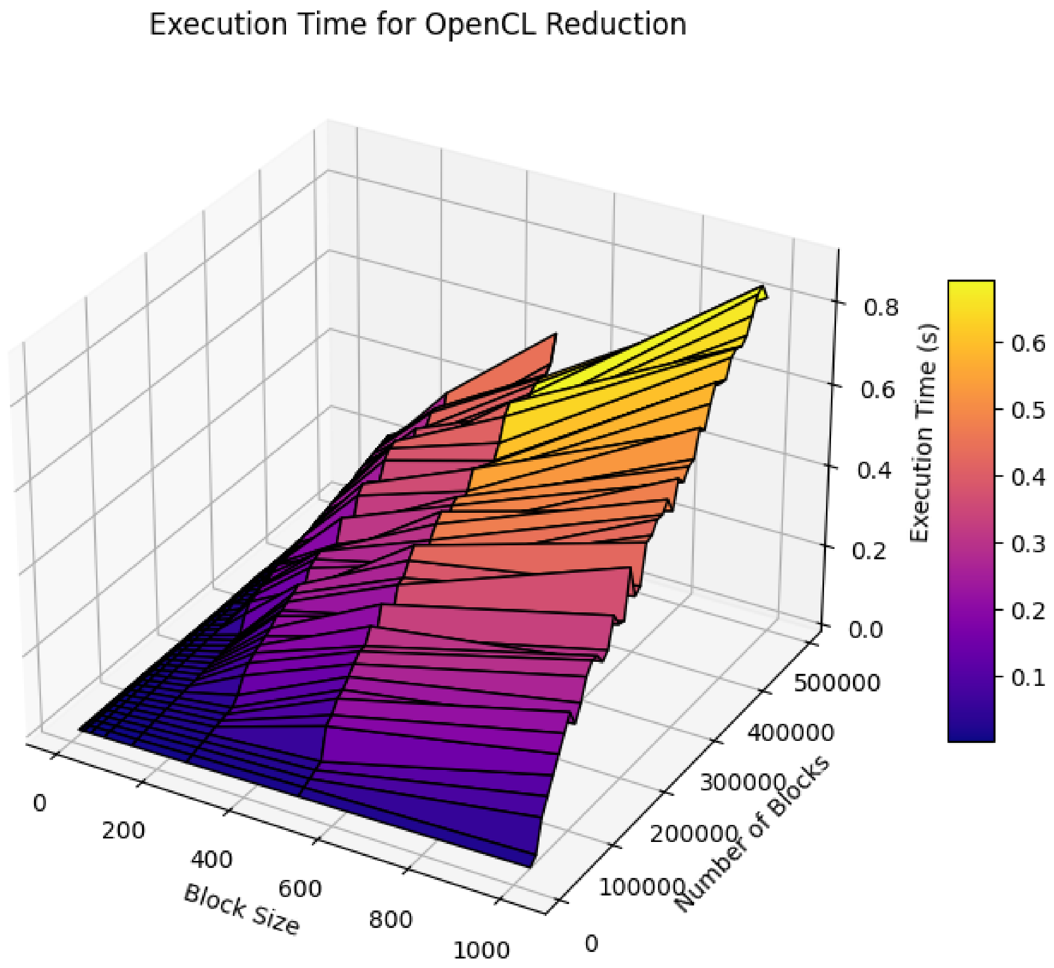
```
37    NVIDIA GeForce RTX 2060      80000
1     NVIDIA GeForce RTX 2060      40000

[1748 rows x 7 columns]
```

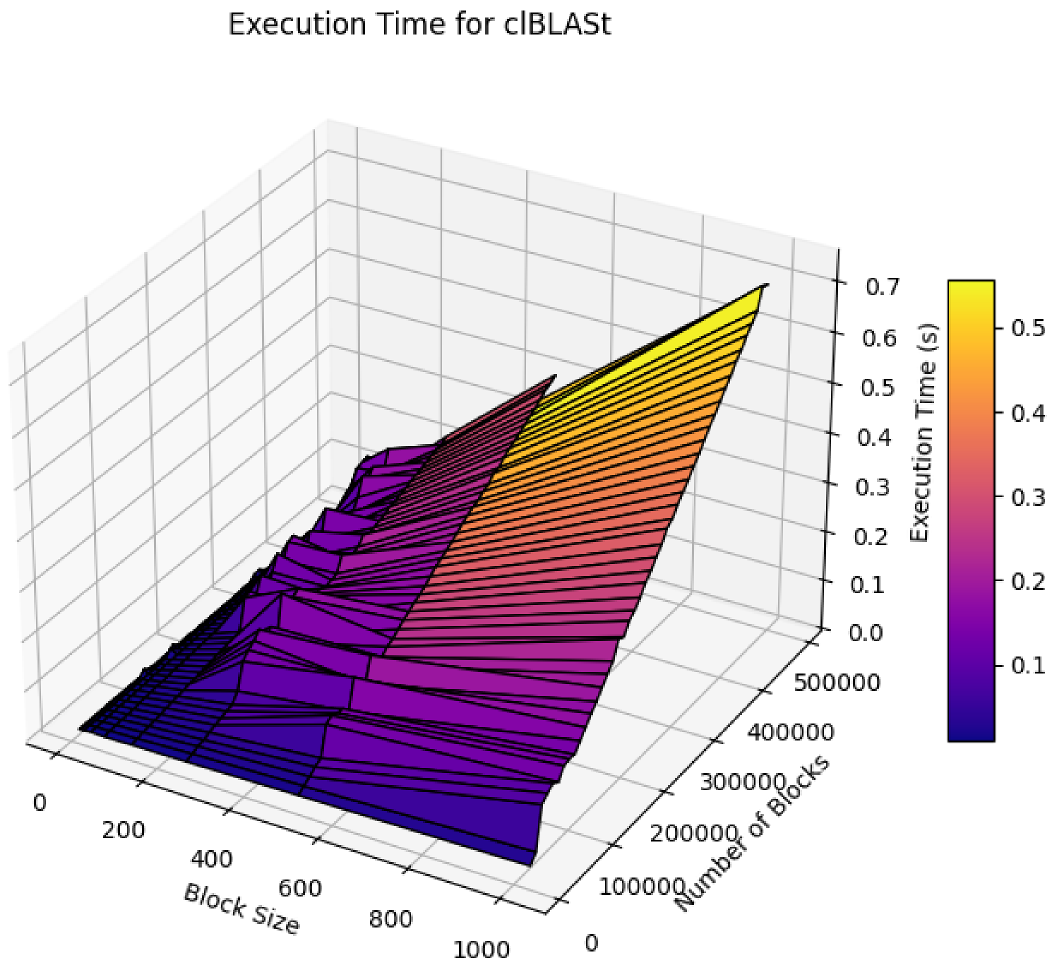
[5]:

```python
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['count'].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['count'] == 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, 5.0)
    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 C++
Block Size
0
200
400
600
800
1000
Number of Blocks
0
100000
200000
300000
400000
500000
Execution Time (s)
0.0
0.2
0.4
0.6
0.8
1.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


Execution Time for OpenBLAS
Block Size
0
200
400
600
800
1000
Number of Blocks
0
100000
200000
300000
400000
500000
Execution Time (s)
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16


Execution Time for OpenCL Reduction
Block Size
0
200
400
600
800
1000
Number of Blocks
0
100000
200000
300000
400000
500000
Execution Time (s)
0.0
0.2
0.4
0.6
0.8
0.1
0.2
0.3
0.4
0.5
0.6


Execution Time for clBLASt
Block Size
0
200
400
600
800
1000
Number of Blocks
0
100000
200000
300000
400000
500000
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Execution Time (s)
0.1
0.2
0.3
0.4
0.5