plots

November 21, 2022

[ ]:

```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pandas as pd
import subprocess as sp
import itertools as it
```

[ ]:

```
def genData(procs, sizes, data_file, cycles=20, clean=False):
    sp.run(
        "/usr/bin/mpiCC -O2 -o 15-matrix-vector.o ./10-matrix-vector.cpp",
        shell=True,
        stdout=sp.DEVNULL,
        stderr=sp.DEVNULL
    )
    if clean:
        sp.run(
            f'echo "num_rows num_cols num_procs avg_time total_time" >␣
 ↪{data_file}',
            shell=True,
            stdout=sp.DEVNULL,
            stderr=sp.DEVNULL
        )

    for size, proc in it.product(sizes, procs):
        for _ in range(cycles):
            sp.run(
                f"/usr/bin/mpirun -n {proc} ./15-matrix-vector.o 0 {size}␣
 ↪{size}",
                shell=True,
                stdout=sp.DEVNULL,
                stderr=sp.DEVNULL
            )
        print(f"Finished {size}x{size} with {proc} processes")

def plotModel(
    data: pd.DataFrame,
    xdata: str,
    ydata: str,
    plot: bool,
    loglog: bool
):
    x = data[xdata].values.reshape(-1, 1)
    y = data[ydata].values.reshape(-1, 1)
    xname = xdata
    yname = ydata
    if loglog:
        x = np.log10(x)
        y = np.log10(y)
        xname = "log(" + xdata + ")"
        yname = "log(" + ydata + ")"

    linear_model = LinearRegression()
    linear_model.fit(x, y)
    model = linear_model.predict(x)

    if plot:
        fig, ax = plt.subplots(figsize=(12,8))
        ax.set_title("Model: y = {:.6f}x + {:.6f}".format(linear_model.
 ↪coef_[0][0], linear_model.intercept_[0]), size=24)
        ax.set_xlabel(xname, size=20)
        ax.tick_params(axis='x', labelsize=16)
        ax.set_ylabel(yname, size=20)
        ax.tick_params(axis='y', labelsize=16)
        ax.scatter(x, y)
        ax.plot(x, model, color='red')
        plt.show()

    return linear_model
```

[ ]:

```
# Variables
# data_file = "/home/jared/Desktop/mv-timings.txt"
data_file = "./mv-timings.txt"
procs = np.array([3, 4, 5, 6, 7, 8])
sizes = np.array([10, 50, 100, 250, 500, 1000, 10000])

# Generate data if needed
# genData(procs=procs, sizes=sizes, data_file=data_file, clean=True)

# Load the data
data = pd.read_csv(data_file, delimiter=" ")
```

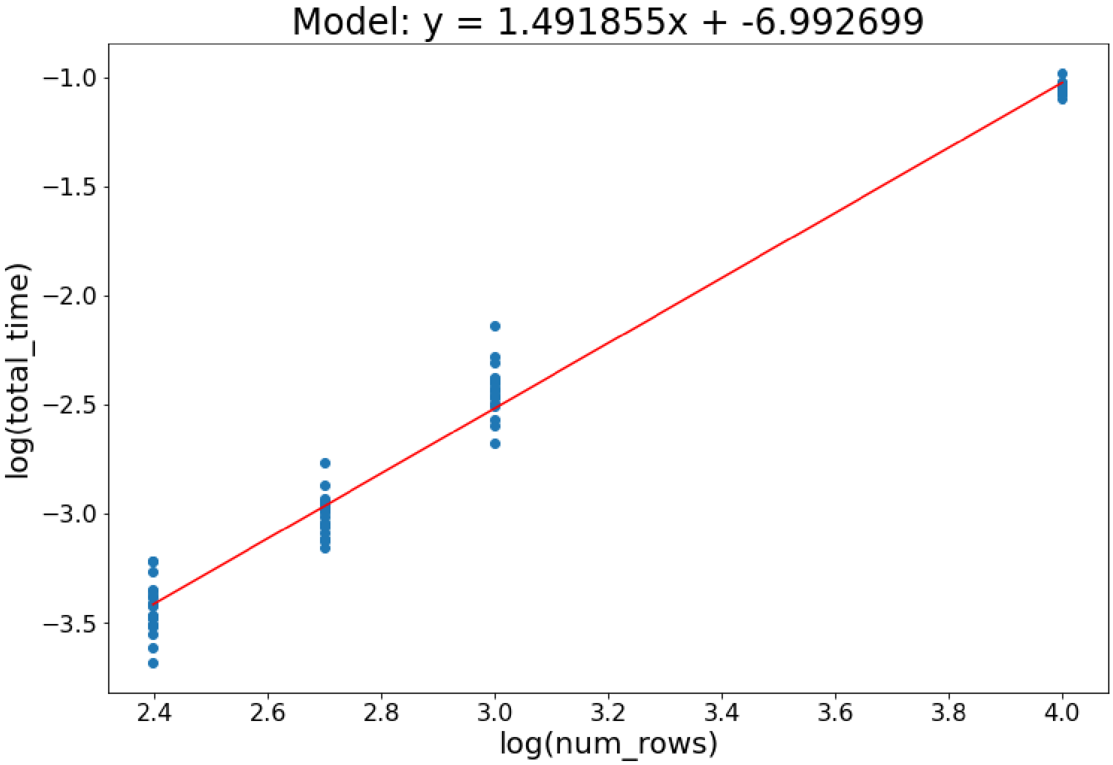
[ ]:

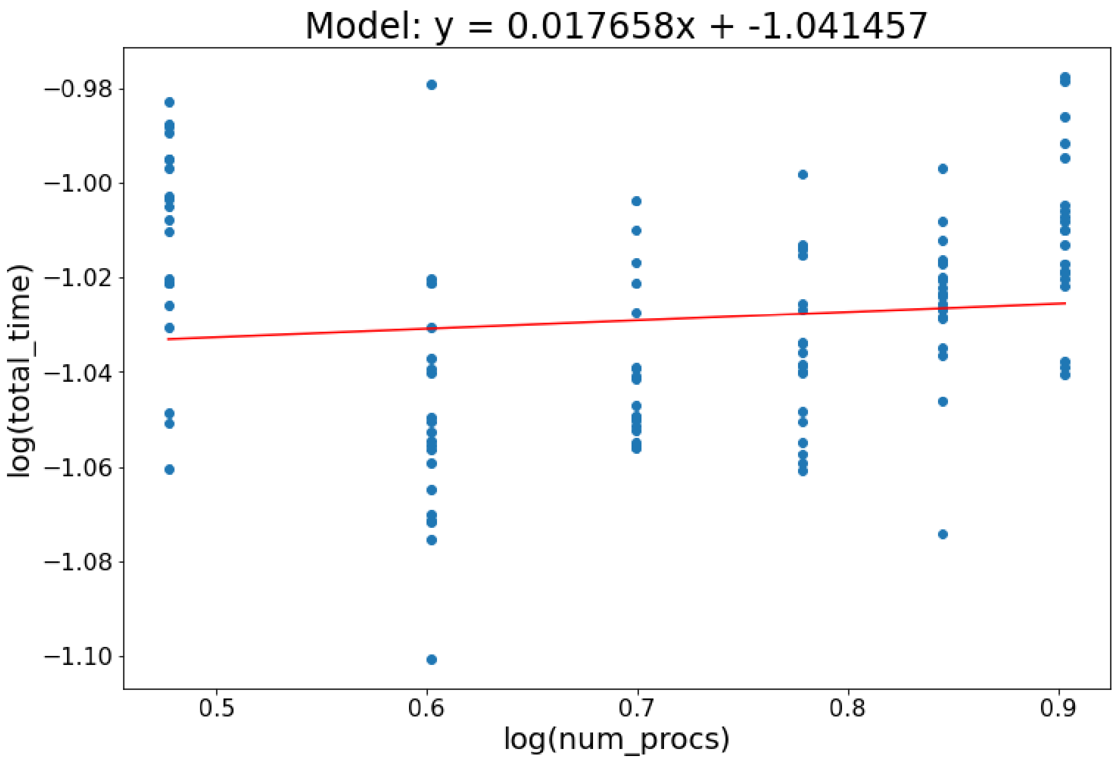
```
num_procs = 4
num_rows = 10000
min_procs = 2
```

```
min_rows = 100

data_reduced_row_scaling = data[data["num_procs"] == num_procs]
data_reduced_row_scaling =␣
 ↪data_reduced_row_scaling[data_reduced_row_scaling["num_rows"] > min_rows]

data_reduced_proc_scaling = data[data["num_rows"] == num_rows]
data_reduced_proc_scaling =␣
 ↪data_reduced_proc_scaling[data_reduced_proc_scaling["num_procs"] > min_procs]
```

[ ]:

```
reduced_row_scaling_model = plotModel(data_reduced_row_scaling, "num_rows",␣
 ↪"total_time", plot=True, loglog=True)
reduced_proc_scaling_model = plotModel(data_reduced_proc_scaling, "num_procs",␣
 ↪"total_time", plot=True, loglog=True)
```





[ ]:

```
num_rows = 10000

serial_times = {
    10: 0.0000047090,
    50: 0.0000092870,
    100: 0.0000360770,
    250: 0.0001662810,
    500: 0.0008027130,
    1000: 0.0025987250,
    10000: 0.4020480850,
}

prediction = np.array([
    10**val[0] for val in
    reduced_proc_scaling_model.predict(procs.reshape(-1, 1))
])

speedup = serial_times[num_rows] / prediction
print("speedup = \n", np.array_str(speedup))

a = speedup / procs
print("a = \n", np.array_str(a))
```

```python
efficiency = speedup / procs
print("efficiency = \n", np.array_str(efficiency))
```

```
speedup =
 [3.91524996 3.75924844 3.60946274 3.46564519 3.32755799 3.19497282]
a =
 [1.30508332 0.93981211 0.72189255 0.57760753 0.47536543 0.3993716 ]
efficiency =
 [1.30508332 0.93981211 0.72189255 0.57760753 0.47536543 0.3993716 ]
```