JMeter Analysis

September 7, 2022

[86]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import MinMaxScaler
```

[69]:
```
def load_dataset(results):
    df = pd.read_csv(results,␣
 ↪usecols=['timeStamp','elapsed','success','bytes','Latency',␣
 ↪'IdleTime','Connect'])
    df['totalElapsed'] = df.elapsed.cumsum()
    df['throughput'] = ((df.index+1)/(df.totalElapsed/(df.index+1))*60000)
    return df
```

[70]:
```
def load_summary(summary):
    return pd.read_csv(summary)
```

[71]:
```
addCand100 = load_dataset('AddCand/100tAddCandBase.csv')
addSummary100 = load_summary('AddCand/100tAddCandBaseSummary.csv')

addCand300 = load_dataset('AddCand/300tAddCandBase.csv')
addSummary300 = load_summary('AddCand/300tAddCandBaseSummary.csv')

addCand500 = load_dataset('AddCand/500tAddCandBase.csv')
addSummary500 = load_summary('AddCand/500tAddCandBaseSummary.csv')
```

[72]:
```
voteCand100 = load_dataset('VoteCand/100tVoteCandBase.csv')
voteSummary100 = load_summary('VoteCand/100tVoteCandBaseSummary.csv')

voteCand300 = load_dataset('VoteCand/300tVoteCandBase.csv')
voteSummary300 = load_summary('VoteCand/300tVoteCandBaseSummary.csv')

voteCand500 = load_dataset('VoteCand/500tVoteCandBase.csv')
voteSummary500 = load_summary('VoteCand/500tVoteCandBaseSummary.csv')
```

[73]:
```
getCand100b = load_dataset('GetCand/100tGetCandBase.csv')
getSummary100b = load_summary('GetCand/100tGetCandBaseSummary.csv')

getCand300b = load_dataset('GetCand/300tGetCandBase.csv')
```

```
getSummary300b = load_summary('GetCand/300tGetCandBaseSummary.csv')

getCand500b = load_dataset('GetCand/500tGetCandBase.csv')
getSummary500b = load_summary('GetCand/500tGetCandBaseSummary.csv')
```

[74]:
```
getCand100 = load_dataset('GetCand/100tGetCand250c.csv')
getSummary100 = load_summary('GetCand/100tGetCand250cSummary.csv')

getCand300 = load_dataset('GetCand/300tGetCand250c.csv')
getSummary300 = load_summary('GetCand/300tGetCand250cSummary.csv')

getCand500 = load_dataset('GetCand/500tGetCand250c.csv')
getSummary500 = load_summary('GetCand/500tGetCand250cSummary.csv')
```

[75]:
```
addCand100
```

[75]:
```
        timeStamp  elapsed  success  bytes  Latency  IdleTime  Connect  \
0   1662506845771     3557     True    261     3557         0     2036
1   1662506845819     3561     True    261     3561         0     2036
2   1662506845867     3555     True    261     3555         0     2036
3   1662506845915     3558     True    261     3558         0     2037
4   1662506845963     3555     True    261     3555         0     2034
..            …        …      …     …      …        …      …
95  1662506850356     4044     True    261     4044         0     2023
96  1662506850406     4024     True    261     4024         0     2004
97  1662506849905     4549     True    261     4549         0     2021
98  1662506850456     4052     True    261     4052         0     2032
99  1662506850506     4039     True    261     4039         0     2014

    totalElapsed   throughput
0           3557    16.868147
1           7118    33.717336
2          10673    50.594959
3          14231    67.458366
4          17786    84.335995
..           …          …
95        367841  1503.258201
96        371865  1518.131580
97        376414  1530.867609
98        380466  1545.630884
99        384505  1560.447849

[100 rows x 9 columns]
```

[77]:
```
getCand100b
```

[77]:
```
        timeStamp  elapsed  success  bytes  Latency  IdleTime  Connect  \
0   1662508610908     3566     True   1903     3566         0     2032
1   1662508610955     3550     True   1903     3550         0     2031
2   1662508611002     3587     True   1903     3587         0     2062
3   1662508611049     3568     True   1903     3568         0     2047
4   1662508611456     3567     True   1903     3567         0     2041
..            …        …       …      …       …        …        …
95  1662508615447     4032     True   1903     4032         0     2010
96  1662508615496     4046     True   1903     4046         0     2023
97  1662508615546     4043     True   1903     4043         0     2020
98  1662508615597     4058     True   1903     4058         0     2031
99  1662508615646     4041     True   1903     4041         0     2014

    totalElapsed   throughput
0           3566    16.825575
1           7116    33.726813
2          10703    50.453144
3          14271    67.269287
4          17838    84.090145
..           …          …
95        369327  1497.209790
96        373373  1512.000064
97        377416  1526.803315
98        381474  1541.546737
99        385515  1556.359675

[100 rows x 9 columns]
```

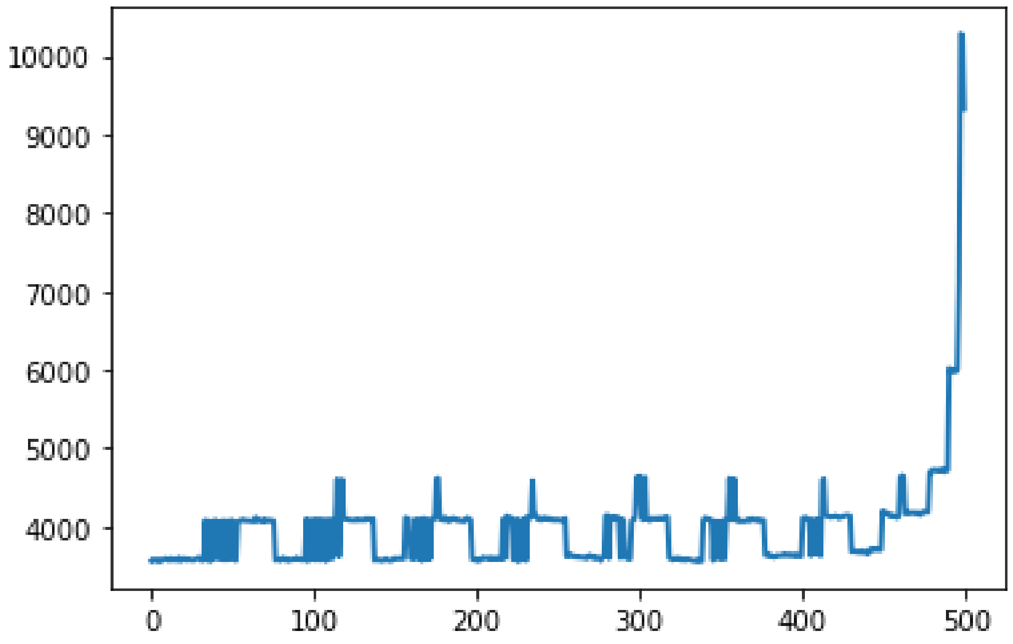
[78]:
```
getCand100
```

[78]:
```
        timeStamp  elapsed  success  bytes  Latency  IdleTime  Connect  \
0   1662510108928     3553     True   8654     3553         0     2029
1   1662510108974     3551     True   8654     3551         0     2029
2   1662510109021     3565     True   8654     3565         0     2044
3   1662510108574     4046     True   8654     4046         0     2022
4   1662510109068     3567     True   8654     3567         0     2043
..            …        …       …      …       …        …        …
95  1662510113514     4041     True   8654     4041         0     2016
96  1662510113463     4092     True   8654     4092         0     2019
97  1662510113413     4143     True   8654     4143         0     2037
98  1662510113064     4589     True   8654     4589         0     2026
99  1662510113014     4641     True   8654     4641         0     2029

    totalElapsed   throughput
0           3553    16.887138
1           7104    33.783784
2          10669    50.613928
```

```
3          14715    65.239551
4          18282    82.047916
..           …          …
95        384248  1439.070600
96        388340  1453.726116
97        392483  1468.190979
98        397072  1480.990853
99        401713  1493.603642

[100 rows x 9 columns]
```

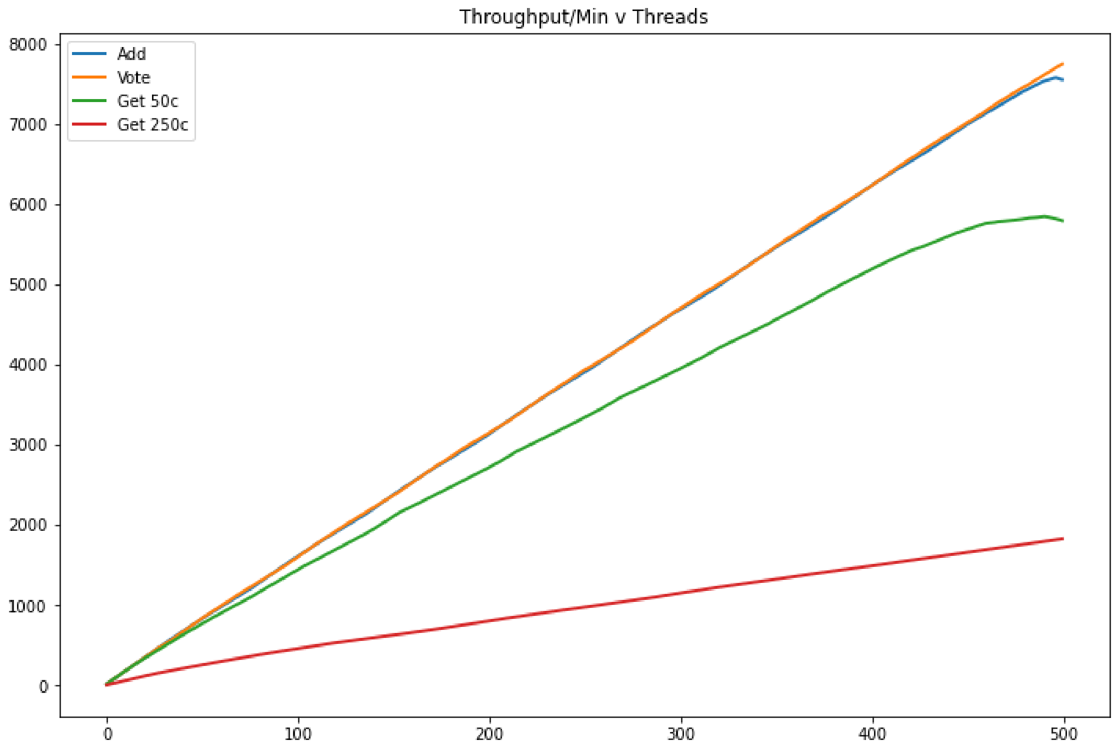
[18]:
```
fig, ax = plt.subplots()
ax.plot(range(0,500), addCand500.elapsed, linewidth=2.0)
```

[18]:
```
[<matplotlib.lines.Line2D at 0x2690a0641c0>]
```



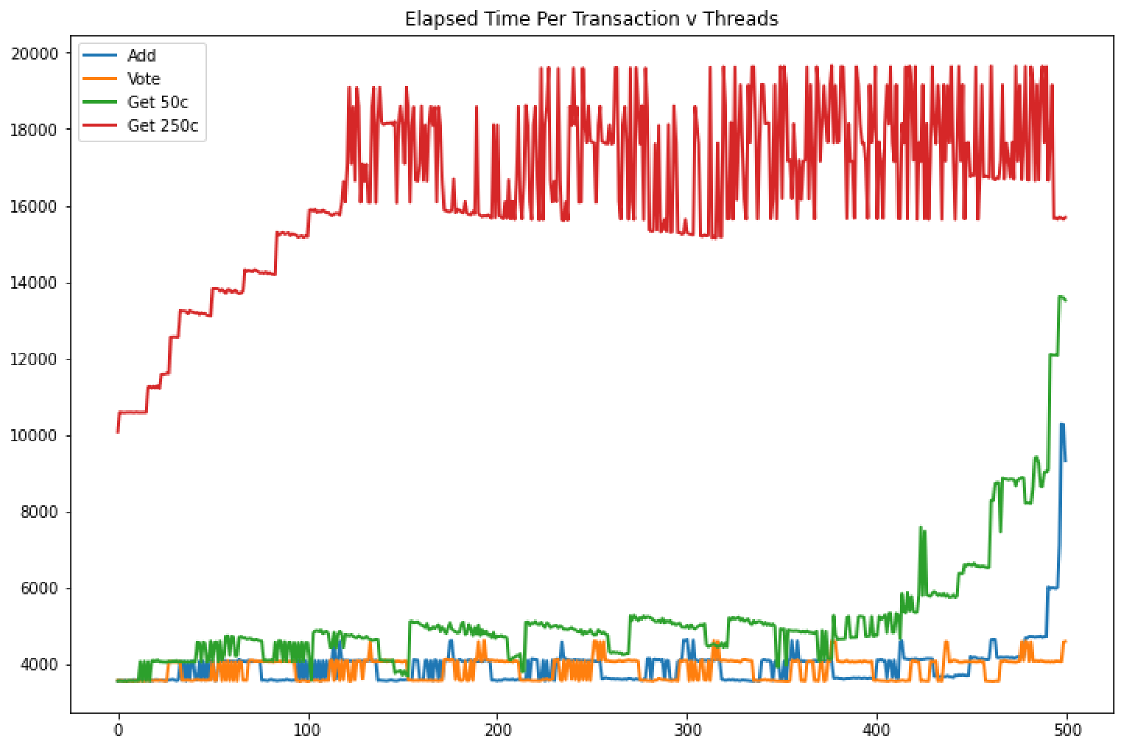
[25]:
```
fig, ax = plt.subplots()
fig.set_figheight(8)
fig.set_figwidth(12)
ax.plot(range(0,500), addCand500.throughput, linewidth=2.0, label="Add")
ax.plot(range(0,500), voteCand500.throughput, linewidth=2.0, label="Vote")
ax.plot(range(0,500), getCand500b.throughput, linewidth=2.0, label="Get 50c")
ax.plot(range(0,500), getCand500.throughput, linewidth=2.0, label="Get 250c")
ax.title.set_text('Throughput/Min v Threads')
ax.legend(loc="upper left")
```

```
[25]: <matplotlib.legend.Legend at 0x2690a535e20>
```



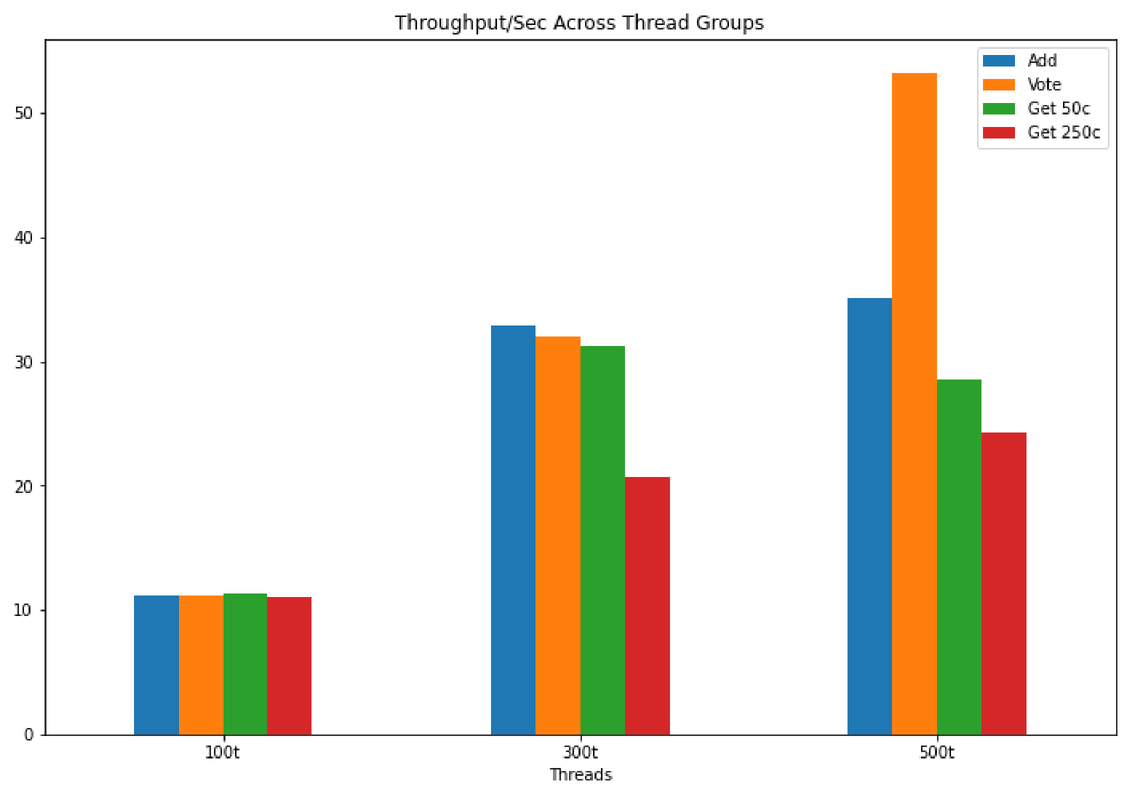
```
[26]: fig, ax = plt.subplots()
      fig.set_figheight(8)
      fig.set_figwidth(12)
      ax.plot(range(0,500), addCand500.elapsed, linewidth=2.0, label="Add")
      ax.plot(range(0,500), voteCand500.elapsed, linewidth=2.0, label="Vote")
      ax.plot(range(0,500), getCand500b.elapsed, linewidth=2.0, label="Get 50c")
      ax.plot(range(0,500), getCand500.elapsed, linewidth=2.0, label="Get 250c")
      ax.title.set_text('Elapsed Time Per Transaction v Threads')
      ax.legend(loc="upper left")
```

```
[26]: <matplotlib.legend.Legend at 0x2690ac4dac0>
```

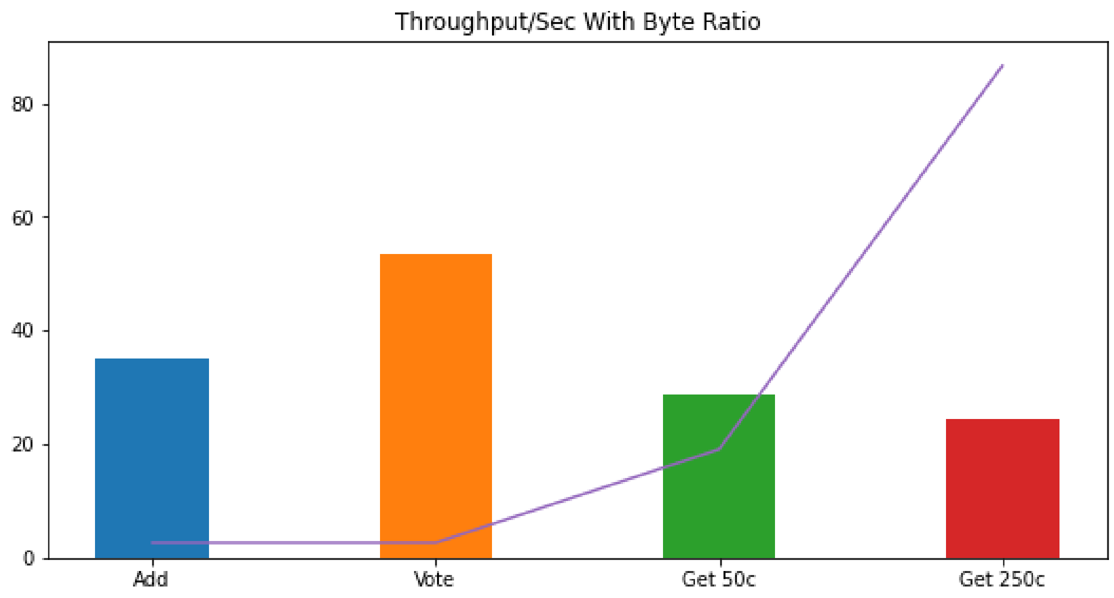


```
[68]: r1 = ['100t', addSummary100.Throughput[1], voteSummary100.Throughput[1],␣
       ↪getSummary100b.Throughput[1], getSummary100.Throughput[1]]
      r2 = ['300t', addSummary300.Throughput[1], voteSummary300.Throughput[1],␣
       ↪getSummary300b.Throughput[1], getSummary300.Throughput[1]]
      r3 = ['500t', addSummary500.Throughput[1], voteSummary500.Throughput[1],␣
       ↪getSummary500b.Throughput[1], getSummary500.Throughput[1]]

      df = pd.DataFrame([r1, r2, r3], columns=['Threads', 'Add', 'Vote', 'Get 50c',␣
       ↪'Get 250c'])

      df.plot(x='Threads',
              kind='bar',
              stacked=False,
              title='Throughput/Sec Across Thread Groups',
              figsize=(12, 8),
              rot=0)
```

```
[68]: <AxesSubplot:title={'center':'Throughput/Sec Across Thread Groups'},
      xlabel='Threads'>
```



[102]:
```
Bytes = [addCand100.bytes[1], voteCand100.bytes[1], getCand100b.bytes[1],␣
 ↪getCand100.bytes[1]]
Throughput = [addSummary500.Throughput[1], voteSummary500.Throughput[1],␣
 ↪getSummary500b.Throughput[1], getSummary500.Throughput[1]]
labels = ['Add', 'Vote', 'Get 50c', 'Get 250c']
colors= ['tab:blue', 'tab:orange', 'tab:green', 'tab:red']
ScaledBytes=[x/100 for x in Bytes]
fig = plt.figure(figsize = (10, 5))
plt.bar(labels, Throughput, width=0.4, color = colors)
plt.plot(labels, ScaledBytes, color = 'tab:purple')
plt.title("Throughput/Sec With Byte Ratio")
plt.show()
```


Throughput/Sec With Byte Ratio
80
60
40
20
0
Add
Vote
Get 50c
Get 250c

```
[ ]:
```