Untitled

April 23, 2018

1 Name: Mishuk Dutta

2 ID: 811361849

3 Project 4 Data-Analysis: Conncetion time, buffer size and Transfer Speed

4 Q1. Introduction

```
The problem we're about to encounter is sending files over connections effieicently. Depending

The purpose of this analysis is to find the sweet spot for FTS. The sample in question is a fi
size of 6.19 MB (6488666 bytes)

By sweetpot, we mean the optimum number of connections and buffer size to get the fastest poss
this might look unnecessary given that the file is just 6mb in size, but in real conditions, a
```

```
In [457]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt

In [458]: data = pd.read_csv('Data.csv')

In [459]: data['total'] = data.apply(lambda(x): x['time']+x['compile'],axis =1 )

In [460]: threads = [0]
          for i in range  (1,17):
              threads_n = data[data['Threads'] == i].copy()
              threads.append(threads_n)

          thr = data[data['Threads'] == 2]
```

5 Q2. Designing The Experiment

A script has been run over my custom FTS program that gathers data from 1024 to 204800 buffer size for 1 to 16 connections.

The Data was primarily stored into a .txt file in nike, then downloaded and run through python Notebook for analyzation. In total, there's 3200 (1 for the Headers) lines of data with 5 datapoints in each line. Each thread gets its own line in the plot area for comparision. The plot will be designed as time vs Buffer Size plot. The average, min and max times along with the next 5 results will be logged and shown. The format of the Data is below with details

```
In [461]: data

Out[461]:       Threads    size      time   compile      total
          0           1    1024  9.910089  0.131973  10.042062
          1           1    2048  4.033282  0.119510   4.152792
          2           1    3072  2.468233  0.122235   2.590467
          3           1    4096  2.006029  0.134829   2.140857
          4           1    5120  1.645603  0.126016   1.771619
          5           1    6144  1.330139  0.125340   1.455479
          6           1    7168  1.267401  0.139792   1.407192
          7           1    8192  1.035530  0.123222   1.158752
          8           1    9216  0.964448  0.124987   1.089435
          9           1   10240  0.943138  0.121612   1.064750
          10          1   11264  0.912140  0.120193   1.032333
          11          1   12288  0.708308  0.140112   0.848420
          12          1   13312  0.747296  0.130062   0.877358
          13          1   14336  0.703058  0.125369   0.828427
          14          1   15360  0.666068  0.122467   0.788535
          15          1   16384  0.603914  0.133811   0.737726
          16          1   17408  0.611772  0.179911   0.791682
          17          1   18432  0.533632  0.144865   0.678497
          18          1   19456  0.519680  0.125323   0.645003
          19          1   20480  0.452819  0.127863   0.580681
          20          1   21504  0.518547  0.120216   0.638762
          21          1   22528  0.437613  0.126067   0.563681
          22          1   23552  0.437348  0.122940   0.560288
          23          1   24576  0.437767  0.121165   0.558932
          24          1   25600  0.431974  0.120636   0.552609
          25          1   26624  0.440804  0.120254   0.561058
          26          1   27648  0.407711  0.124374   0.532086
          27          1   28672  0.407273  0.121388   0.528662
          28          1   29696  0.368238  0.121659   0.489897
          29          1   30720  0.351866  0.120085   0.471952
          ...       ...     ...       ...       ...        ...
          3169       16  174080  0.154329  0.135967   0.290295
          3170       16  175104  0.164087  0.137608   0.301694
          3171       16  176128  0.163418  0.161894   0.325311
          3172       16  177152  0.170971  0.146684   0.317655
          3173       16  178176  0.162620  0.149084   0.311704
          3174       16  179200  0.161887  0.132616   0.294503
          3175       16  180224  0.241901  0.141928   0.383829
          3176       16  181248  0.159925  0.130770   0.290694
```

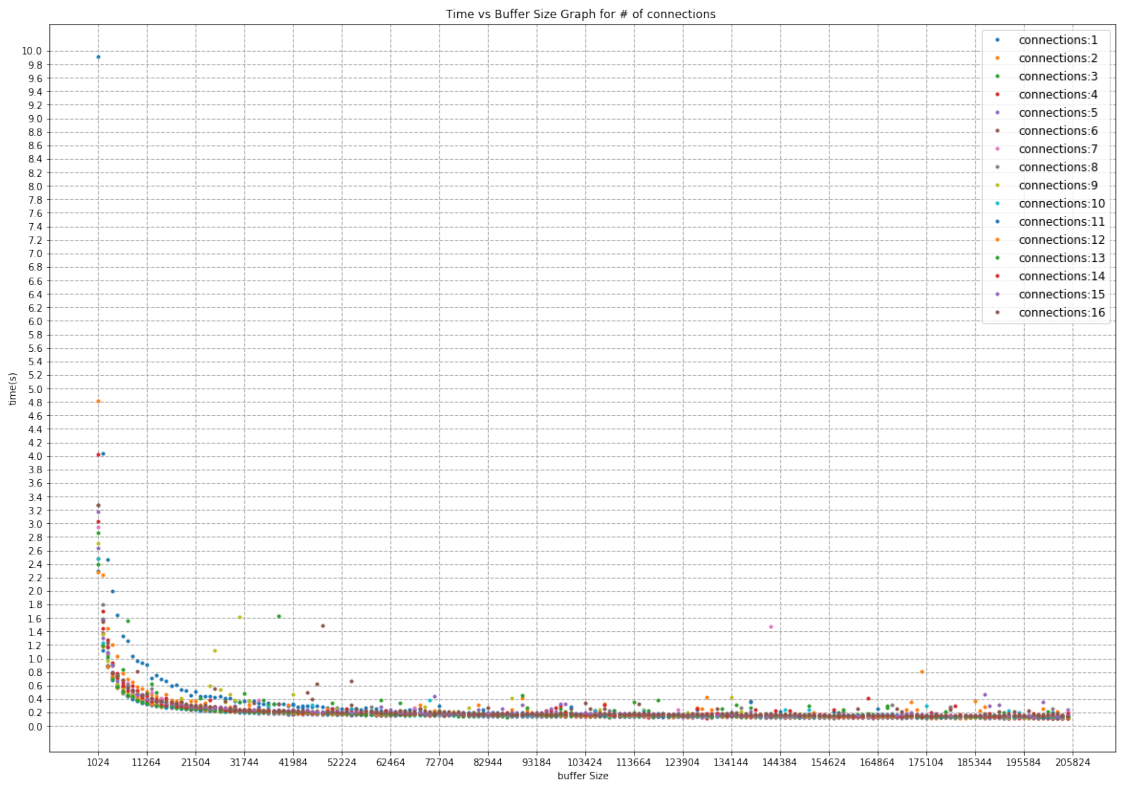
```
3177        16  182272  0.159552  0.150359   0.309911
3178        16  183296  0.160464  0.130405   0.290869
3179        16  184320  0.158351  0.130784   0.289135
3180        16  185344  0.161769  0.148167   0.309936
3181        16  186368  0.164675  0.358519   0.523195
3182        16  187392  0.171359  0.139245   0.310604
3183        16  188416  0.158648  0.141852   0.300500
3184        16  189440  0.157556  0.156660   0.314216
3185        16  190464  0.165063  0.135294   0.300357
3186        16  191488  0.160541  0.137590   0.298131
3187        16  192512  0.181392  0.143959   0.325351
3188        16  193536  0.151943  0.135953   0.287897
3189        16  194560  0.154366  0.155444   0.309810
3190        16  195584  0.160126  0.166809   0.326935
3191        16  196608  0.226377  0.141902   0.368280
3192        16  197632  0.166205  0.140405   0.306610
3193        16  198656  0.164528  0.138194   0.302722
3194        16  199680  0.165979  0.135149   0.301128
3195        16  200704  0.178252  0.156781   0.335033
3196        16  201728  0.171697  0.139633   0.311330
3197        16  202752  0.183250  0.144723   0.327973
3198        16  203776  0.155423  0.125666   0.281089

[3199 rows x 5 columns]
```

6 Plotting the Graphs

```
In [462]: n = 1

          plt.rcParams['figure.figsize'] = 20, 14
          for i in range (1,17):
              plt.plot('size','time','.',data=threads[i],label = 'connections:'+str(i),alpha =
          plt.yticks((np.arange(0, 10.2, 0.2 )))
          plt.rc('legend',fontsize='Large') # using a named size
          plt.ylabel('time(s)')
          plt.xlabel('buffer Size')
          plt.xticks(np.arange(1024, 204800+10240, 10240 ))
          plt.title('Time vs Buffer Size Graph for # of connections ')
          plt.legend()
          plt.grid(linestyle='--', linewidth=1)
          plt.show()
```



7 Mininum Transfer Times for each Thread and the Combinations

```
In [463]: min =[0]
          absolute_min = [0]

          for i in range (1,17):
              print 'No. of Connections',i
              th = threads[i].sort_values('time',ascending = True).head(5).reset_index().copy(
              absolute_min.append(th.at[0,'time'])
              min.append(th)
              print(th)
```

```
No. of Connections 1
   index  Threads    size      time   compile     total
0    196        1  201728  0.118481  0.123985  0.242465
1    194        1  199680  0.123515  0.120388  0.243903
2    193        1  198656  0.124267  0.119042  0.243309
3    198        1  203776  0.124572  0.132711  0.257282
4    189        1  194560  0.125415  0.126639  0.252055
No. of Connections 2
   index  Threads    size      time   compile     total
0    387        2  192512  0.115090  0.122167  0.237257
```

```
1    398        2  203776  0.116110  0.111383  0.227493
2    371        2  176128  0.117413  0.109769  0.227182
3    378        2  183296  0.118057  0.119136  0.237194
4    390        2  195584  0.118498  0.118568  0.237066
No. of Connections 3
   index  Threads    size      time   compile     total
0    586        3  191488  0.114867  0.119211  0.234078
1    399        3  204800  0.115275  0.116353  0.231627
2    596        3  201728  0.116799  0.122714  0.239513
3    570        3  175104  0.117828  0.123176  0.241004
4    593        3  198656  0.117864  0.117440  0.235304
No. of Connections 4
   index  Threads    size      time   compile     total
0    788        4  193536  0.112652  0.108218  0.220870
1    789        4  194560  0.113687  0.110157  0.223845
2    798        4  203776  0.113755  0.109129  0.222883
3    796        4  201728  0.114638  0.111063  0.225701
4    787        4  192512  0.115955  0.116813  0.232768
No. of Connections 5
   index  Threads    size      time   compile     total
0    983        5  188416  0.118741  0.110255  0.228996
1    959        5  163840  0.119536  0.114799  0.234336
2    965        5  169984  0.122130  0.122385  0.244515
3    987        5  192512  0.122493  0.332179  0.454671
4    984        5  189440  0.123466  0.117128  0.240595
No. of Connections 6
   index  Threads    size      time   compile     total
0   1196        6  201728  0.119525  0.117394  0.236920
1   1198        6  203776  0.122431  0.122182  0.244613
2   1194        6  199680  0.124984  0.111440  0.236423
3   1197        6  202752  0.129041  0.130031  0.259072
4   1182        6  187392  0.129711  0.119447  0.249158
No. of Connections 7
   index  Threads    size      time   compile     total
0   1385        7  190464  0.118025  0.124467  0.242491
1   1398        7  203776  0.120039  0.117868  0.237906
2   1387        7  192512  0.124211  0.114459  0.238669
3   1383        7  188416  0.124233  0.114296  0.238528
4   1384        7  189440  0.125274  0.113333  0.238608
No. of Connections 8
   index  Threads    size      time   compile     total
0   1579        8  184320  0.123924  0.116867  0.240791
1   1598        8  203776  0.126286  0.117801  0.244086
2   1587        8  192512  0.126881  0.136662  0.263544
3   1577        8  182272  0.129128  0.115861  0.244990
4   1560        8  164864  0.130977  0.117194  0.248171
No. of Connections 9
   index  Threads    size      time   compile     total
```

```
0   1788        9  193536  0.127306  0.117974  0.245281
1   1776        9  181248  0.129421  0.115470  0.244891
2   1781        9  186368  0.129944  0.123214  0.253158
3   1787        9  192512  0.130065  0.112839  0.242904
4   1784        9  189440  0.132024  0.114903  0.246927
No. of Connections 10
   index  Threads    size      time   compile     total
0   1981       10  186368  0.130541  0.121433  0.251973
1   1976       10  181248  0.131269  0.122015  0.253283
2   1986       10  191488  0.133131  0.125427  0.258558
3   1988       10  193536  0.134621  0.121047  0.255668
4   1983       10  188416  0.135529  0.129380  0.264909
No. of Connections 11
   index  Threads    size      time   compile     total
0   2194       11  199680  0.134871  0.123602  0.258473
1   2195       11  200704  0.135345  0.136283  0.271629
2   2173       11  178176  0.136137  0.130952  0.267089
3   2197       11  202752  0.136294  0.127487  0.263781
4   2166       11  171008  0.136823  0.131025  0.267848
No. of Connections 12
   index  Threads    size      time   compile     total
0   2379       12  184320  0.130548  0.120307  0.250855
1   2382       12  187392  0.137622  0.126896  0.264519
2   2199       12  204800  0.140193  0.123631  0.263823
3   2381       12  186368  0.140527  0.132878  0.273405
4   2380       12  185344  0.140704  0.125928  0.266632
No. of Connections 13
   index  Threads    size      time   compile     total
0   2571       13  176128  0.141788  0.127318  0.269105
1   2581       13  186368  0.142949  0.131320  0.274269
2   2568       13  173056  0.143138  0.132703  0.275841
3   2594       13  199680  0.143312  0.126901  0.270212
4   2580       13  185344  0.143554  0.139102  0.282656
No. of Connections 14
   index  Threads    size      time   compile     total
0   2783       14  188416  0.142898  0.126811  0.269710
1   2785       14  190464  0.145022  0.143395  0.288417
2   2782       14  187392  0.145432  0.135160  0.280592
3   2784       14  189440  0.146238  0.126963  0.273201
4   2771       14  176128  0.147902  0.130226  0.278128
No. of Connections 15
   index  Threads    size      time   compile     total
0   2977       15  182272  0.147493  0.137376  0.284869
1   2997       15  202752  0.149488  0.136093  0.285581
2   2992       15  197632  0.149933  0.135416  0.285348
3   2989       15  194560  0.150663  0.145437  0.296100
4   2980       15  185344  0.152536  0.154354  0.306890
No. of Connections 16
```

```
   index  Threads    size      time   compile     total
0   2999       16  204800  0.148591  0.126267  0.274858
1   3188       16  193536  0.151943  0.135953  0.287897
2   3169       16  174080  0.154329  0.135967  0.290295
3   3189       16  194560  0.154366  0.155444  0.309810
4   3198       16  203776  0.155423  0.125666  0.281089
```

8 Maximum Transfer Times for each Thread and the Combinations

```
In [464]: max =[0]
          absolute_max= [0]
          for i in range (1,17):
              print '# of Connections',i
              th = threads[i].sort_values('time',ascending = False).head(5).reset_index().copy
              max.append(th)
              absolute_max.append(th.at[0,'time'])
              print(th)
```

```
# of Connections 1
   index  Threads  size      time   compile      total
0      0        1  1024  9.910089  0.131973  10.042062
1      1        1  2048  4.033282  0.119510   4.152792
2      2        1  3072  2.468233  0.122235   2.590467
3      3        1  4096  2.006029  0.134829   2.140857
4      4        1  5120  1.645603  0.126016   1.771619
# of Connections 2
   index  Threads  size      time   compile     total
0    200        2  1024  4.815254  0.121990  4.937245
1    201        2  2048  2.236547  0.121900  2.358447
2    202        2  3072  1.453443  0.125003  1.578446
3    203        2  4096  1.199789  0.127006  1.326795
4    204        2  5120  1.033803  0.125697  1.159501
# of Connections 3
   index  Threads   size      time   compile     total
0    400        3   1024  2.858205  0.130939  2.989145
1    437        3  38912  1.632128  0.149337  1.781465
2    401        3   2048  1.594107  0.221414  1.815521
3    406        3   7168  1.566637  0.131480  1.698117
4    402        3   3072  1.063828  0.140776  1.204605
# of Connections 4
   index  Threads  size      time   compile     total
0    600        4  1024  4.024405  0.141784  4.166189
1    601        4  2048  1.704144  0.131747  1.835890
2    602        4  3072  1.270328  0.168770  1.439098
3    603        4  4096  0.940474  0.148276  1.088750
4    604        4  5120  0.782933  0.133174  0.916106
```

```
# of Connections 5
   index  Threads  size      time   compile     total
0    800        5  1024  2.638034  0.140546  2.778580
1    801        5  2048  1.312172  0.131303  1.443475
2    802        5  3072  0.896364  0.132243  1.028607
3    803        5  4096  0.732023  0.132641  0.864664
4    804        5  5120  0.607239  0.132943  0.740183
# of Connections 6
   index  Threads  size      time   compile     total
0   1000        6  1024  2.486748  0.142638  2.629385
1   1001        6  2048  1.371364  0.130725  1.502089
2   1002        6  3072  0.888541  0.136964  1.025505
3   1003        6  4096  0.729023  0.253213  0.982236
4   1004        6  5120  0.587167  0.134364  0.721531
# of Connections 7
   index  Threads    size      time   compile     total
0   1200        7    1024  2.944579  0.131028  3.075607
1   1201        7    2048  1.580151  0.146661  1.726812
2   1338        7  142336  1.479002  0.127070  1.606072
3   1202        7    3072  1.083758  0.134312  1.218071
4   1203        7    4096  0.790825  0.134336  0.925162
# of Connections 8
   index  Threads  size      time   compile     total
0   1400        8  1024  3.270864  0.142309  3.413174
1   1401        8  2048  1.802576  0.146329  1.948905
2   1402        8  3072  1.158351  0.134469  1.292820
3   1403        8  4096  0.910260  0.131819  1.042079
4   1404        8  5120  0.746873  0.131896  0.878768
# of Connections 9
   index  Threads   size      time   compile     total
0   1600        9   1024  2.713881  0.132760  2.846641
1   1629        9  30720  1.615979  0.143303  1.759282
2   1601        9   2048  1.361164  0.176892  1.538056
3   1624        9  25600  1.126961  0.204141  1.331102
4   1602        9   3072  0.971171  0.142538  1.113710
# of Connections 10
   index  Threads  size      time   compile     total
0   1800       10  1024  2.477823  0.136066  2.613889
1   1801       10  2048  1.232102  0.135899  1.368001
2   1802       10  3072  0.876146  0.139812  1.015958
3   1803       10  4096  0.708241  0.136082  0.844323
4   1804       10  5120  0.583788  0.133631  0.717420
# of Connections 11
   index  Threads  size      time   compile     total
0   2000       11  1024  2.292008  0.152659  2.444667
1   2001       11  2048  1.122429  0.139282  1.261712
2   2002       11  3072  0.876754  0.143412  1.020166
3   2003       11  4096  0.677206  0.269717  0.946923
```

```
4   2004       11  5120  0.570481  0.144089  0.714570
# of Connections 12
   index  Threads  size      time   compile     total
0   2200       12  1024  2.276215  0.141580  2.417795
1   2201       12  2048  1.176049  0.140458  1.316507
2   2202       12  3072  0.875093  0.139724  1.014817
3   2203       12  4096  0.729481  0.137898  0.867379
4   2204       12  5120  0.573877  0.147579  0.721456
# of Connections 13
   index  Threads   size      time   compile     total
0   2400       13   1024  2.402086  0.161569  2.563656
1   2401       13   2048  1.196947  0.155121  1.352068
2   2402       13   3072  1.025164  0.140655  1.165819
3   2403       13   4096  0.692387  0.140230  0.832617
4   2411       13  12288  0.631149  0.139406  0.770556
# of Connections 14
   index  Threads  size      time   compile     total
0   2600       14  1024  3.037651  0.148731  3.186382
1   2601       14  2048  1.443250  0.153321  1.596570
2   2602       14  3072  1.176334  0.158212  1.334546
3   2603       14  4096  0.782447  0.147590  0.930037
4   2604       14  5120  0.715869  0.160155  0.876024
# of Connections 15
   index  Threads  size      time   compile     total
0   2800       15  1024  3.167803  0.157900  3.325703
1   2801       15  2048  1.581138  0.146292  1.727430
2   2802       15  3072  1.097496  0.151206  1.248701
3   2803       15  4096  0.892376  0.149211  1.041587
4   2805       15  6144  0.685117  0.143581  0.828698
# of Connections 16
   index  Threads   size      time   compile     total
0   3000       16   1024  3.274145  0.151793  3.425939
1   3001       16   2048  1.548934  0.143196  1.692130
2   3046       16  48128  1.488671  0.156787  1.645458
3   3002       16   3072  1.238741  0.145108  1.383849
4   3008       16   9216  0.813624  0.144249  0.957873
```

9 Average times for each Thread

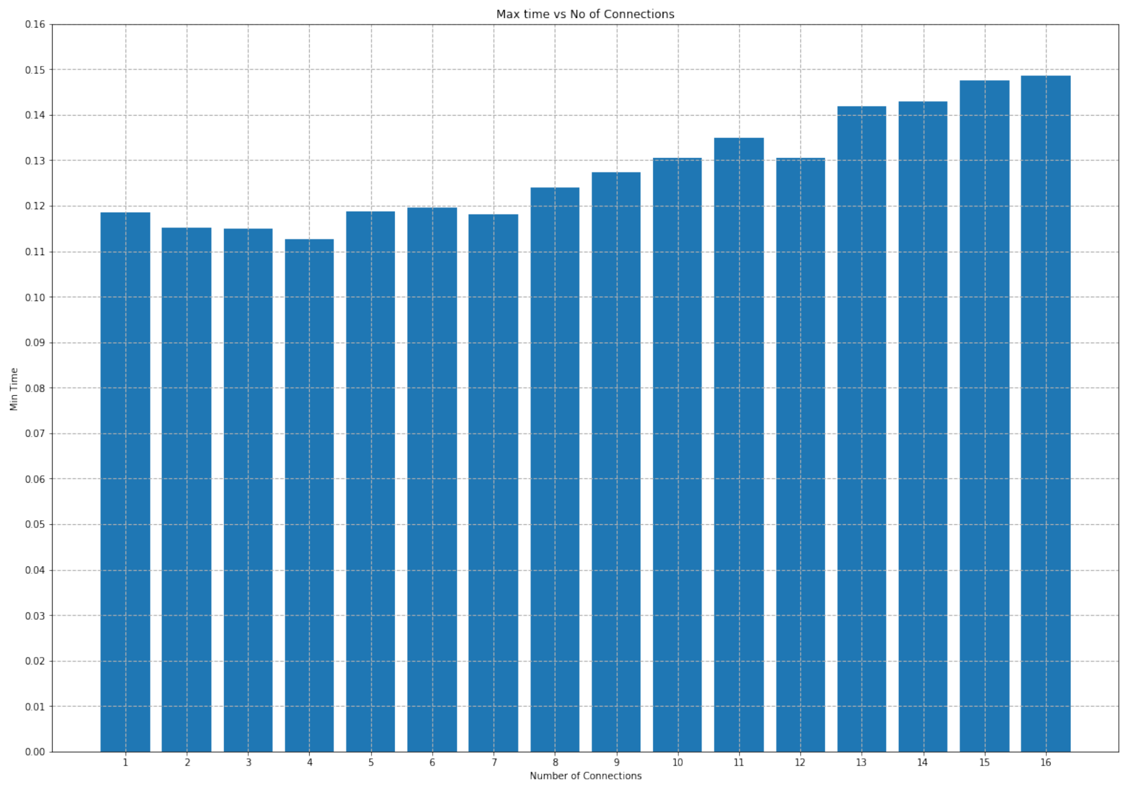
```
In [465]: average = [0]
          print 'No. of Connections'
          for i in range (1,17):
              th = threads[i].time.mean()
              average.append(th)
              print i,': ',th
```

```
No. of Connections
```

```
1 :  0.340249192161
2 :  0.24970511557
3 :  0.229303561455
4 :  0.22616542701
5 :  0.19763129829
6 :  0.207722271685
7 :  0.21813781623
8 :  0.228183550855
9 :  0.234856599585
10 :  0.208607228995
11 :  0.209124852415
12 :  0.214886378025
13 :  0.21941656352
14 :  0.232190015335
15 :  0.24467004777
16 :  0.26230425597
```

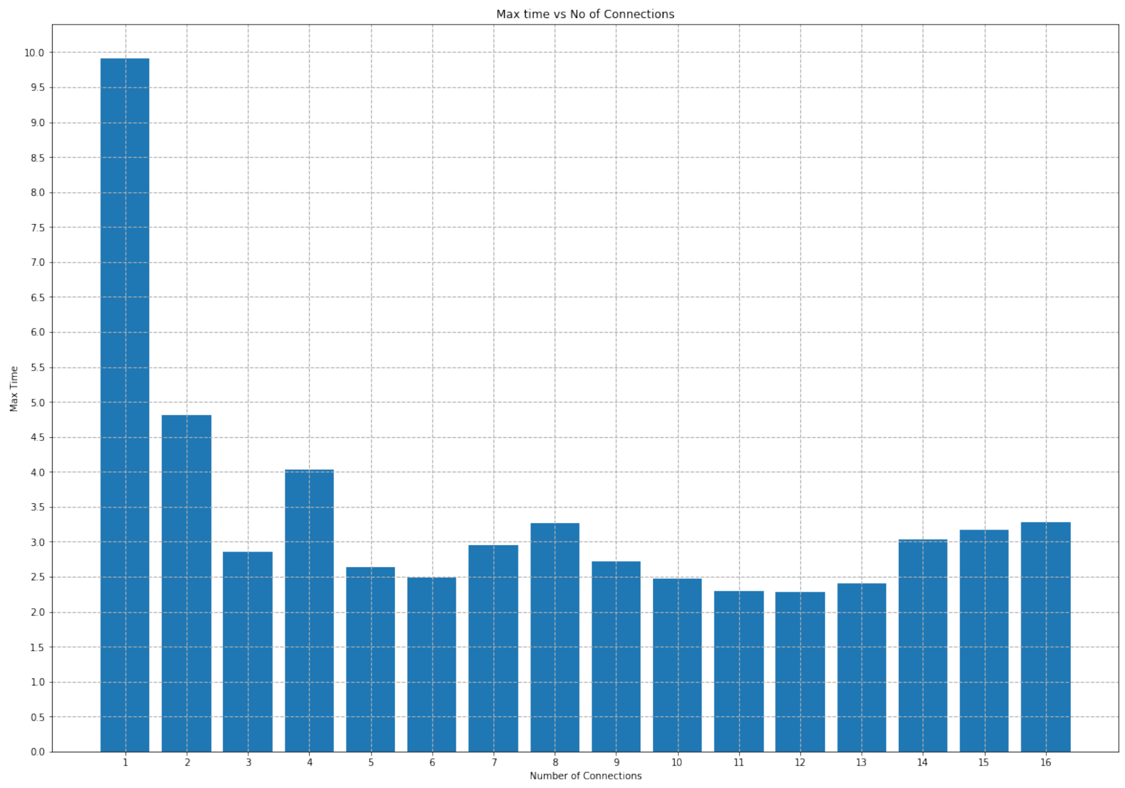
10 Bar Graph Showing Min Times

In [466]:
```
connections =[0]
for i in range (1,17):
    connections.append(i)
connections = tuple(connections)
#print connections
plt.bar(connections[1:],absolute_min[1:],align='center', alpha=1)
plt.yticks((np.arange(0, np.amax(absolute_min)+0.02 , 0.01 )))
plt.rc('legend',fontsize='Large') # using a named size
plt.ylabel('Min Time')
plt.xlabel('Number of Connections')
plt.xticks(np.arange(1, 17, 1 ))
plt.title('Max time vs No of Connections')
plt.legend()
plt.grid(linestyle='--', linewidth=1)
plt.show()
```



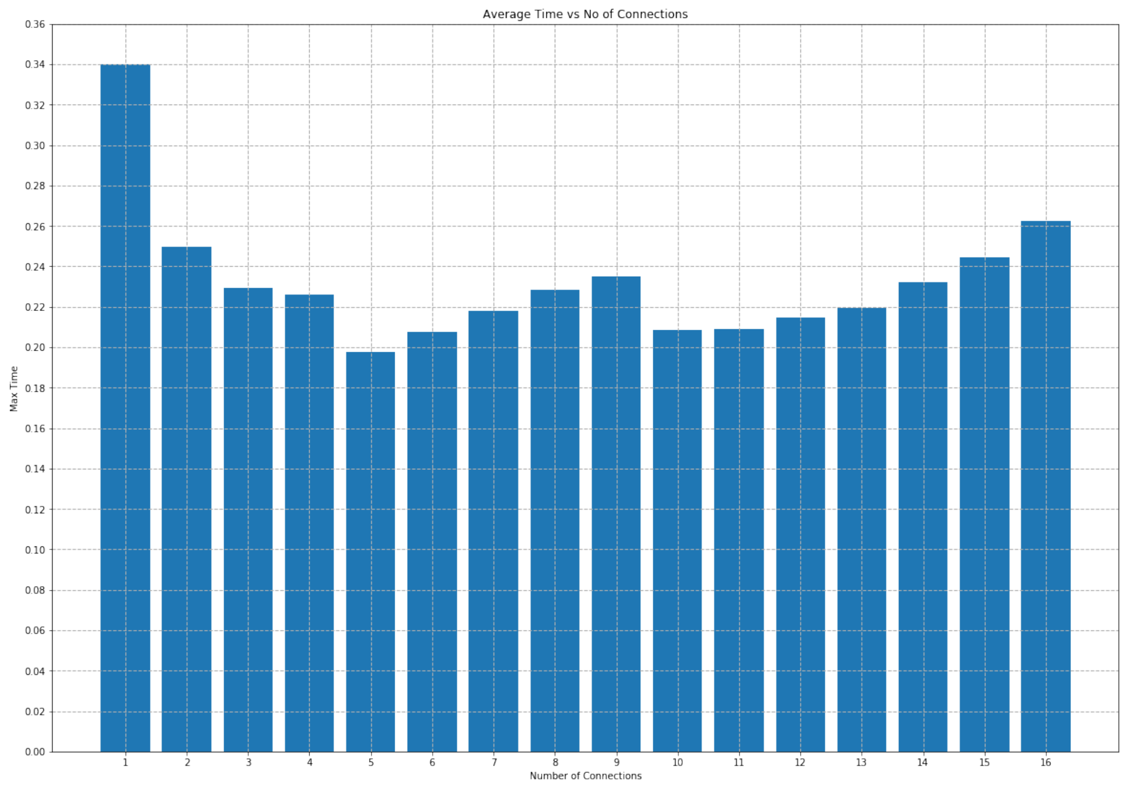
11 Bar Graph Showing Max Times

```
In [467]: plt.bar(connections[1:],absolute_max[1:],align='center', alpha=1)
          plt.yticks((np.arange(0, np.amax(absolute_max)+0.5 , 0.5 )))
          plt.rc('legend',fontsize='Large') # using a named size
          plt.ylabel('Max Time')
          plt.xlabel('Number of Connections')
          plt.xticks(np.arange(1, 17, 1 ))
          plt.title('Max time vs No of Connections')
          plt.legend()
          plt.grid(linestyle='--', linewidth=1)
          plt.show()
```



12 Bar Graph Showing Average Times

```
In [468]: plt.bar(connections[1:],average[1:],align='center', alpha=1)
          plt.yticks((np.arange(0, np.amax(average)+0.02 , 0.02 )))
          plt.rc('legend',fontsize='Large') # using a named size
          plt.ylabel('Max Time')
          plt.xlabel('Number of Connections')
          plt.xticks(np.arange(1, 17, 1 ))
          plt.title('Average Time vs No of Connections')
          plt.legend()
          plt.grid(linestyle='--', linewidth=1)
          plt.show()
```



13 Fastest 15 Times

```
In [469]: data_sorted = data.sort_values('time',ascending = True).reset_index().copy()

In [470]: data_sorted.head(15)

Out[470]:     index  Threads    size      time   compile     total
          0     788        4  193536  0.112652  0.108218  0.220870
          1     789        4  194560  0.113687  0.110157  0.223845
          2     798        4  203776  0.113755  0.109129  0.222883
          3     796        4  201728  0.114638  0.111063  0.225701
          4     586        3  191488  0.114867  0.119211  0.234078
          5     387        2  192512  0.115090  0.122167  0.237257
          6     399        3  204800  0.115275  0.116353  0.231627
          7     787        4  192512  0.115955  0.116813  0.232768
          8     398        2  203776  0.116110  0.111383  0.227493
          9     596        3  201728  0.116799  0.122714  0.239513
          10    792        4  197632  0.116830  0.107298  0.224128
          11    371        2  176128  0.117413  0.109769  0.227182
          12    570        3  175104  0.117828  0.123176  0.241004
          13    593        3  198656  0.117864  0.117440  0.235304
          14   1385        7  190464  0.118025  0.124467  0.242491
```

14 Slowest 15 Times

```
In [471]: data_sorted.tail(15).sort_values('time',ascending = False)
```

```
Out[471]:       index  Threads  size      time   compile      total
          3198      0        1  1024  9.910089  0.131973  10.042062
          3197    200        2  1024  4.815254  0.121990   4.937245
          3196      1        1  2048  4.033282  0.119510   4.152792
          3195    600        4  1024  4.024405  0.141784   4.166189
          3194   3000       16  1024  3.274145  0.151793   3.425939
          3193   1400        8  1024  3.270864  0.142309   3.413174
          3192   2800       15  1024  3.167803  0.157900   3.325703
          3191   2600       14  1024  3.037651  0.148731   3.186382
          3190   1200        7  1024  2.944579  0.131028   3.075607
          3189    400        3  1024  2.858205  0.130939   2.989145
          3188   1600        9  1024  2.713881  0.132760   2.846641
          3187    800        5  1024  2.638034  0.140546   2.778580
          3186   1000        6  1024  2.486748  0.142638   2.629385
          3185   1800       10  1024  2.477823  0.136066   2.613889
          3184      2        1  3072  2.468233  0.122235   2.590467
```

15 Q3 Result and Discussion

From the above, it's clear that more threads/bufferSize doesn't necessarily mean the best possible time. It's also evedent that small sizes are EXTREMELY inefficient for File Transfer Systems. Buffer sizes of less than exponentially increase the transfer time. Times between 11264 21504 buffer sizes, although are smoothing out, aren't the best either. 41k bytes is where the Time differences smoothens out. As such, Conclusively it's best to use bufferSizes above that. For max efficiencty, the best buffer size should be 62k+ as all the data points seem to merge somewhat linearly.

Connections: Connections <3 or >11 effects trasfer time adverely. For the given file, 3 to 9 connections are linearly merged at 50k+ buffer sizes. The best Averages for thread results were 5 and 10, 11. 4 and 34 followed close by. So the observation remains true, and the other connections higher transfer times

Thus, for the most efficiency and flexibility, the buffer sizes should be kept above 64k and conncections should range from 3 to 11

16 Q4. Conclusion

Higer connections doesn't mean the best possible transfer times. Our Best Times were at 4 connections with high buffer sizes but the overall average best transfer time was at 5 connections. The buffer size conclusion remains the same