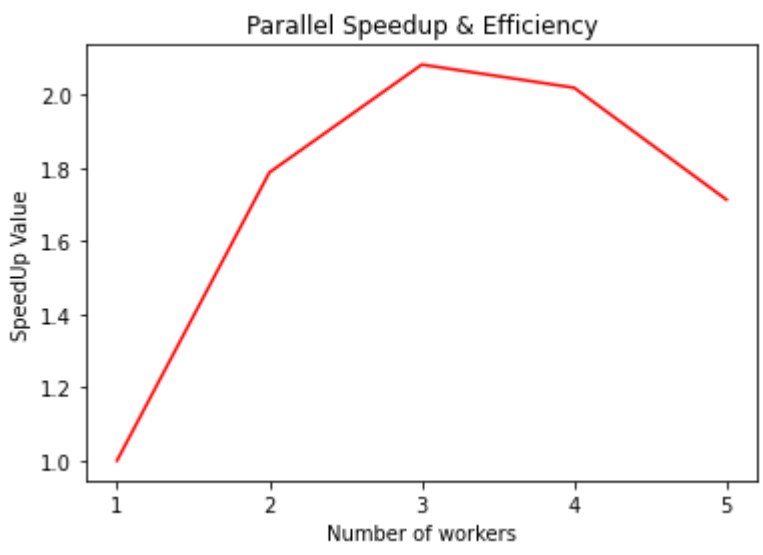
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

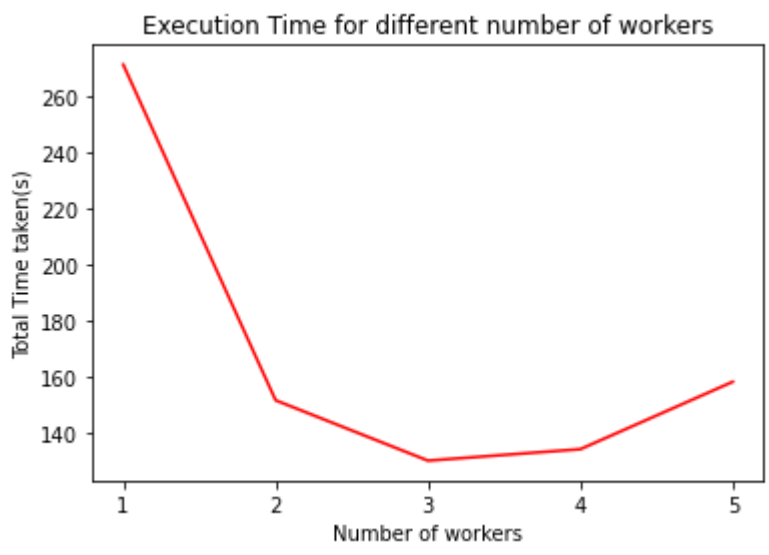
Exercise 1 Implementing Parallel Stochastic Gradient Descent

In [5]:
```
#Initializing the number of workers
workers = ["1", "2", "3", "4", "5"]

#Intializing the array with all the time
Tp = [271.3823,151.8623,130.3759,134.4552,158.4506 ]
Ts = Tp[0] #Time with 1 worker(serial)
Sp = [Ts / i for i in Tp] #calculating speedup
```

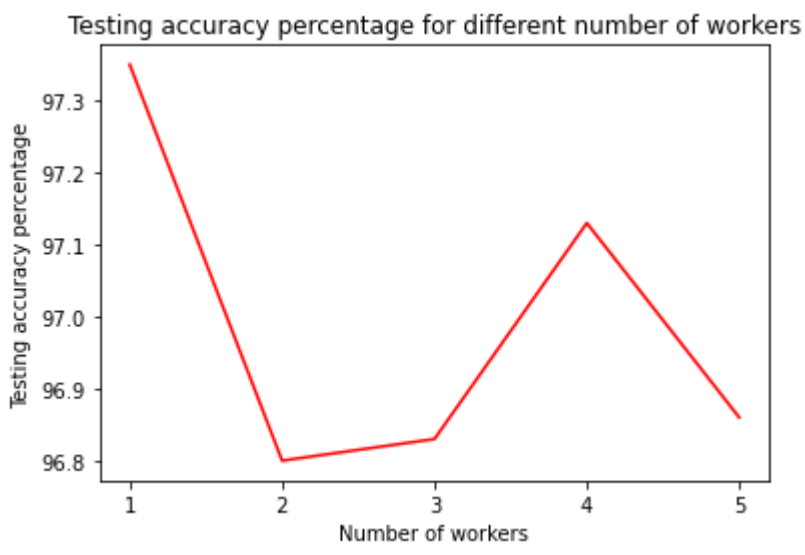
In [6]:
```
#plot of speedup
plt.plot(workers, Sp, color='red')
plt.title('Parallel Speedup & Efficiency')
plt.xlabel('Number of workers')
plt.ylabel('SpeedUp Value')

plt.show()
```



In [7]:
```
#plotting graph for checking the execution time
#vs number of workers

plt.title('Execution Time for different number of workers')
plt.xlabel('Number of workers')
plt.ylabel('Total Time taken(s)')
plt.plot(workers, Tp, color='red')

plt.show()
```



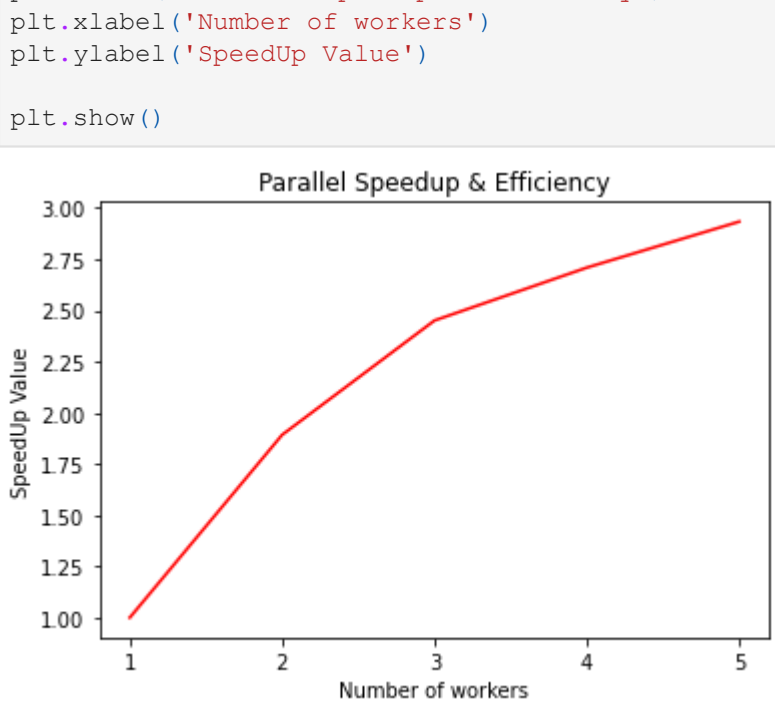
In [12]:
```
test_acc = [97.3500,96.8000,96.8300,97.1300,96.8600]
#Plotting graph for testing accuracy for different workers

plt.title('Testing accuracy percentage for different number of workers')
plt.xlabel('Number of workers')
plt.ylabel('Testing accuracy percentage')
plt.plot(workers, test_acc, color='red')

plt.show()
```



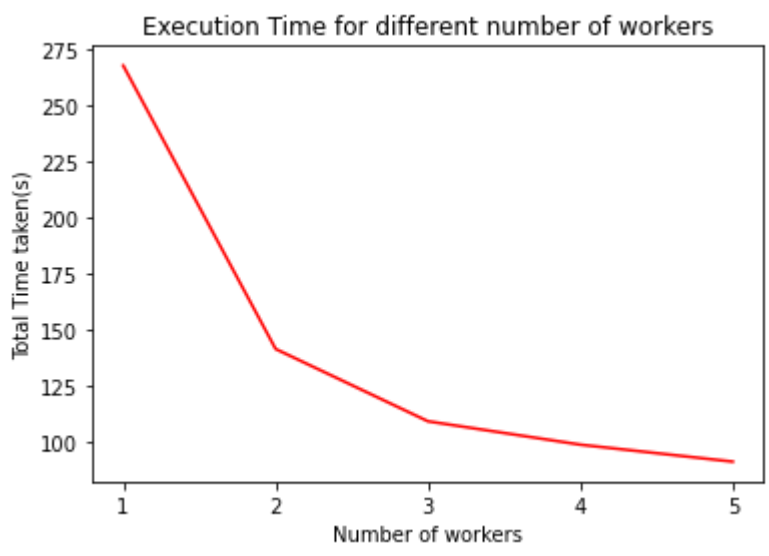
Exercise 2 PyTorch distributed execution

In [8]:
```
#Number of workers is same as previous


#Intializing the array with all the time
Tp = [268.0132, 141.6145,109.3527,98.9295,91.3572 ]
Ts = Tp[0] #Time with 1 worker(serial)
Sp = [Ts / i for i in Tp] #calculating speedup
```

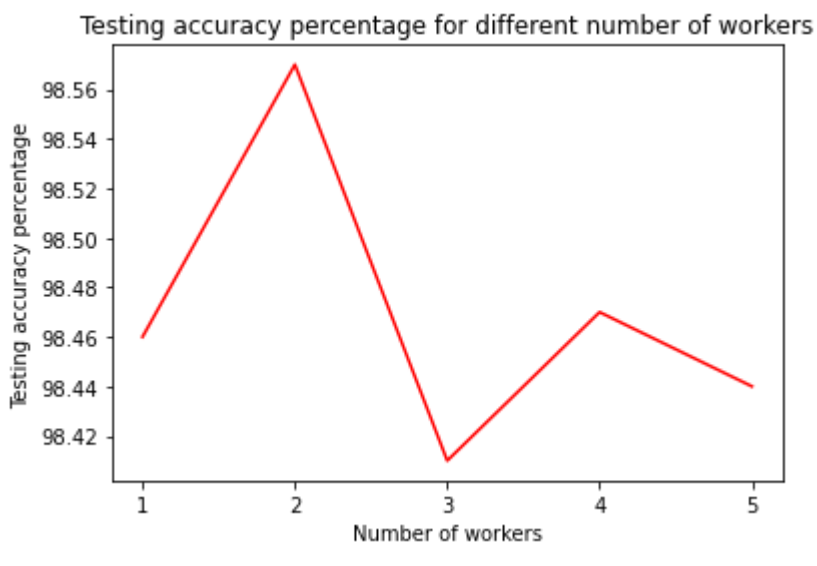
In [9]:
```
#plot of speedup
plt.plot(workers, Sp, color='red')
plt.title('Parallel Speedup & Efficiency')
plt.xlabel('Number of workers')
plt.ylabel('SpeedUp Value')

plt.show()
```



In [10]:
```
#plotting graph for checking the execution time
#vs number of workers

plt.title('Execution Time for different number of workers')
plt.xlabel('Number of workers')
plt.ylabel('Total Time taken(s)')
plt.plot(workers, Tp, color='red')

plt.show()
```



In [11]:
```
test_acc = [98.4600, 98.5700, 98.4100, 98.4700, 98.4400]
#Plotting graph for testing accuracy for different processes

plt.title('Testing accuracy percentage for different number of processes')
plt.xlabel('Number of processes')
plt.ylabel('Testing accuracy percentage')
plt.plot(workers, test_acc, color='red')

plt.show()
```



In [ ]: