2016-05-20-155457

May 20, 2016

In [ ]:
```
XIAO WANG
1333447
WANG19
```

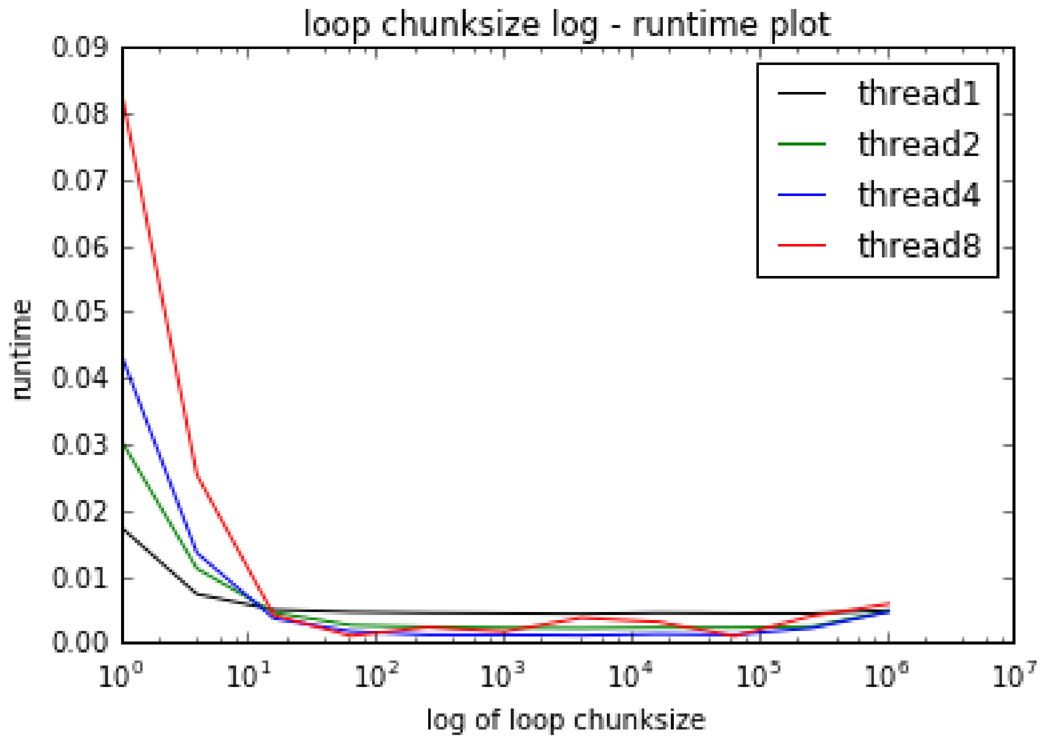
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from numpy import array, linspace, pi, sin, exp

from homework3 import (
    simps_parallel_chunked,
    time_simps_parallel_chunked,
)
runtime1 = []
runtime2 = []
runtime4 = []
runtime8 = []
N = 2**20
x = np.linspace(-1,3,N)
y = sin(exp(x))
chunk_sizes = [2**0, 2**2, 2**4, 2**6, 2**8, 2**10,2**12,2**14,2**16,2**18,2**20]


for size in chunk_sizes:
    runtime1.append(time_simps_parallel_chunked(y,x,num_threads = 1,chunk_size = size))
    runtime2.append(time_simps_parallel_chunked(y,x,num_threads = 2,chunk_size = size))
    runtime4.append(time_simps_parallel_chunked(y,x,num_threads = 4,chunk_size = size))
    runtime8.append(time_simps_parallel_chunked(y,x,num_threads = 8,chunk_size = size))



thread_1, = plt.semilogx(chunk_sizes, runtime1, label = 'thread_1', color = 'black')
thread_2, = plt.semilogx(chunk_sizes, runtime2, label = 'thread_2', color = 'green')
thread_4, = plt.semilogx(chunk_sizes, runtime4, label = 'thread_4', color = 'blue')
thread_8, = plt.semilogx(chunk_sizes, runtime8, label = 'thread_8', color = 'red')
plt.xlabel('log of loop chunksize')
plt.ylabel('runtime')
plt.title('loop chunksize log - runtime plot')
plt.legend([thread_1, thread_2, thread_4, thread_8], ['thread1', 'thread2', 'thread4', 'thread8
plt.show()
```



What can be attributed to the nthreads = 8 behavior when the chunk_size is small? Because the chunck size is small and thread will take on less work every time. But the threads have to work more time each thread because they have same workload. And for each thread, it has to use more time to find the next work. Thus, it requires more time.

What happens when the chunk_size is equal to the problem size (i.e. when chunk_size = 2**20 = N) and why does this affect the timings in the way that the plot suggests? since the chunk_size= 2ˆ20 = N that means there will be only one thread working entirely. So the run time will be very close to the run time of thread where thread = 1.And the runn time will be lot more than runtime of mutiple thread working at the same time. From the draw above, we can conclude threads 2,4,8 need more time to run.

Why is chunk_size relevant to this implementation of Simpson's rule as opposed to the simple example we did in class during Lecture 12? (5 May, 2016?) In my view, this implementation of simpson's rule has different memory access because it iterates with a step of i+2,If we want to take the advantage of cache locality (i.e. contiguous memory access pattern), then we need to consider the chunk_size in order to assure that the iteration in each chunk is contiguous.

What range of chunk values appear to be optimal for this particular value of N in parallel Simpson? Can you conjecture an ideal chunk value as a function of N . If so, why? If not, why not? Note that this "optimal" chunk size may not be the same for non-Simpson's rule problems.

When the chunk_size is between 2ˆ10and 2ˆ21 appear to be optimal. An ideal chunk value equals problem size N divided by number of threads.