1. Context:

- We have to compute the value of pi using integration of 4/(1+x^2) in the interval [0, 1]. For computing this, we can run a for loop which iterates over values of x, starting from 0 to 1. After every iteration, the value of x is incremented by a small amount. This leads to a large running time. Since the values x takes is independent of each other, the portion can be parallelized to achieve speedup.

- Complexity of the serial algorithm is O(n), where n is the number of divisions, the interval [0, 1] is divided into.

- Possible speedup(theoretical) :
  Speedup S = 1/ (P/n + s)
  n – number of cores
  P – percentage of code that can be parallelized
  s – percentage of serial code(which is not parallelized)
  For our code, P ~ 1 and s ~0
  n=4, So theoretical speedup = 4

- Optimization strategy

We have four cores, so the code can be parallelized into 4 segments. The interval of x, i.e. [0,1] if divided into n steps, then n/4 iterations can be performed by each segment. The program is divided into four threads, therefore for n operations, there would be n/4 concurrent operations. Theoretically, this will increase the speedup the process 4 times than the serial code.

- Problems faced in parallelization :

The time consumed in calculation of pi varies every time. Since it is not consistent, problems are faced while plotting different graphs.

2. Hardware details:

CPU : Intel® Core™ i5-4200U CPU @ 1.60GHz × 4
Compiler : gcc
Precision : Double

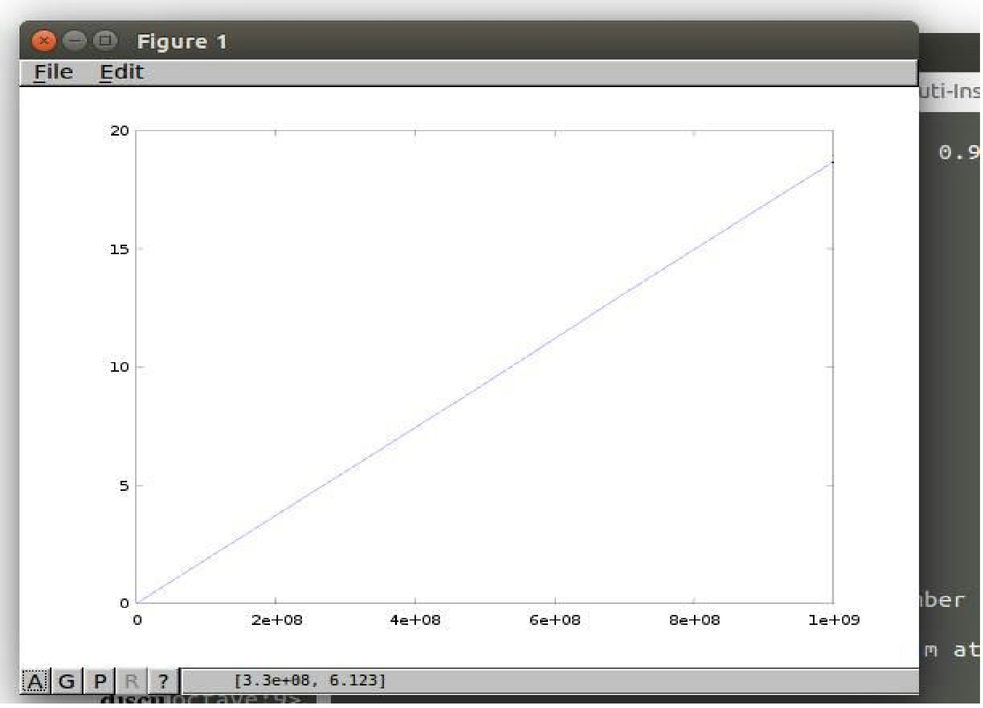
3. Output: The value of pi approximated by the machine. The time taken for computation of the value.

Value of pi estimated from serial code= 2.141593
Value of pi estimated from parallel code= 2.141593

4. Problem Size Vs Time

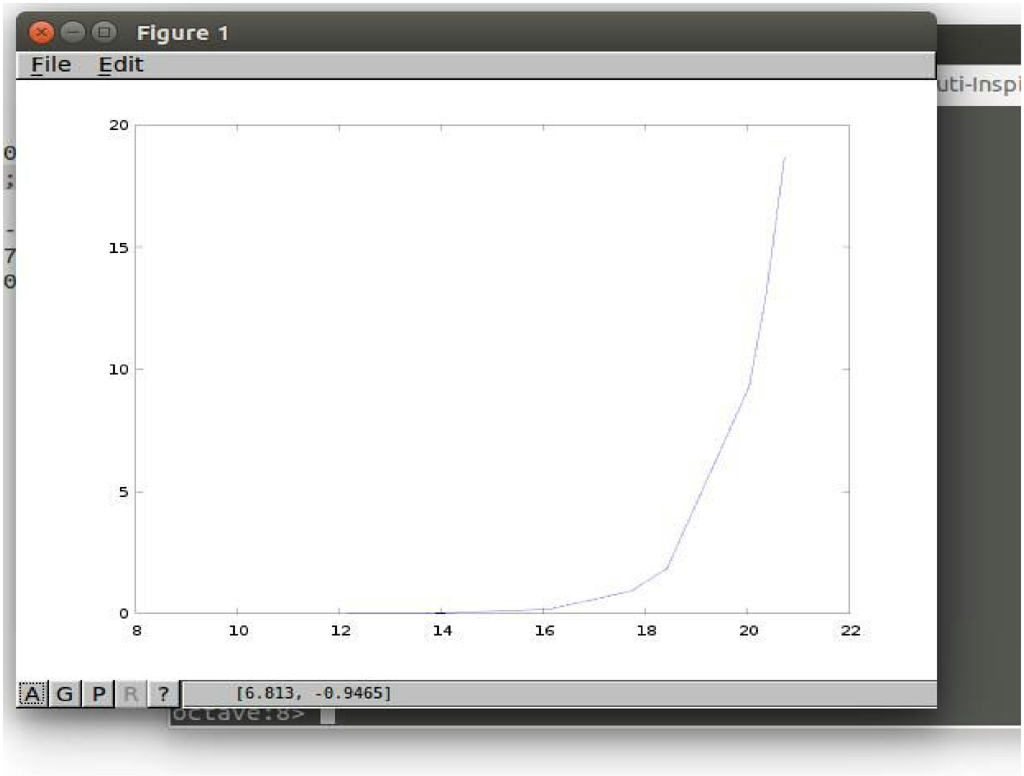
SERIAL CODE :

Problem size



Time(seconds)

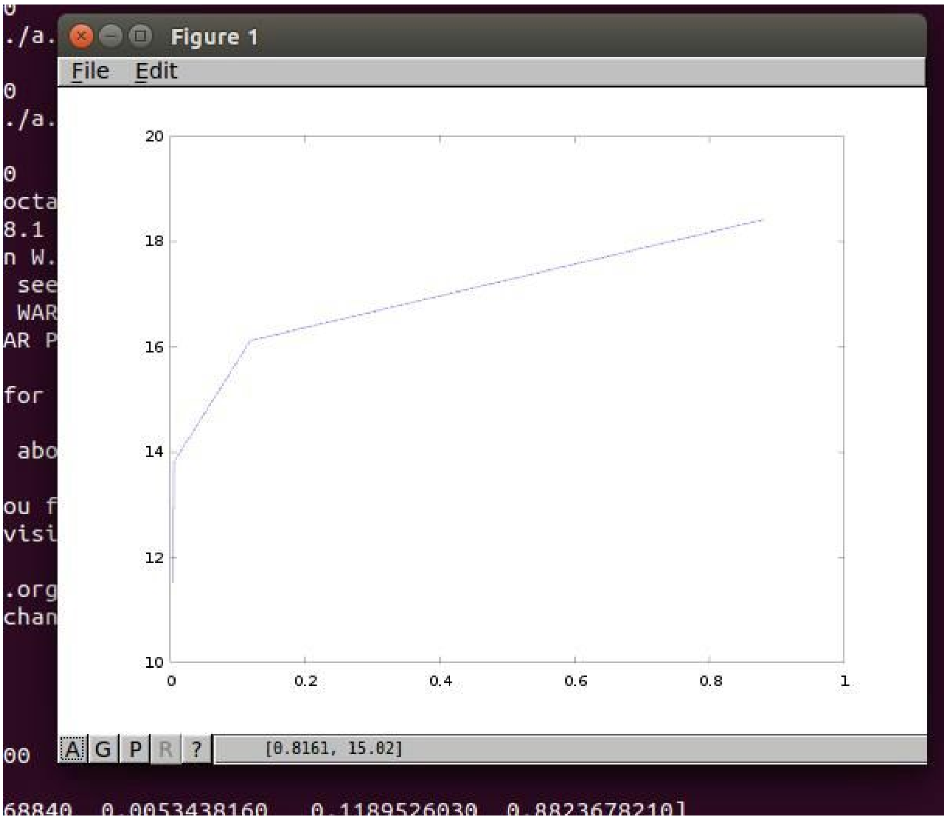
log(Problem Size)



Time(seconds)

PARALLEL CODE:

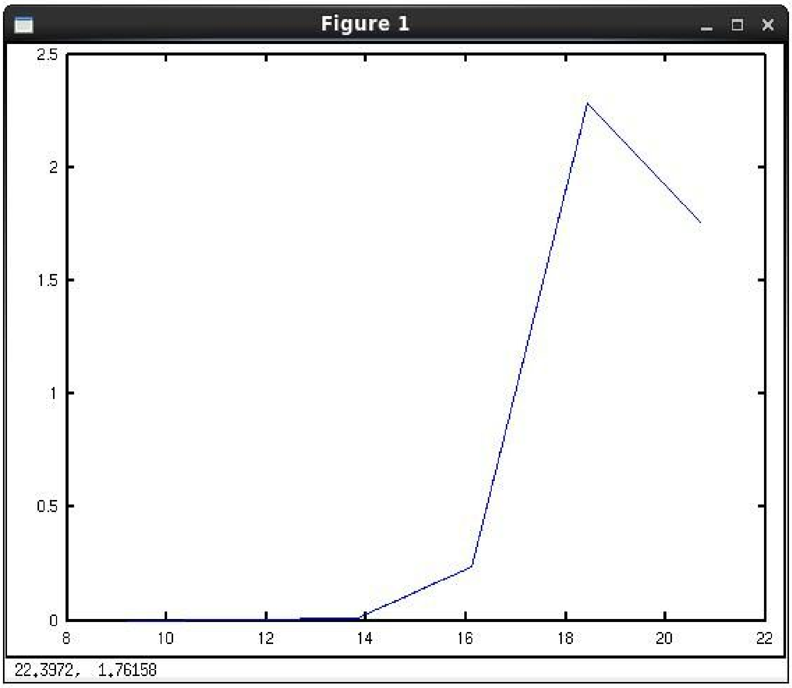
Problem
Size



```
0
./a.
0
./a.
0
octa
8.1
n W.
 see
 WAR
AR P
for
 abo
ou f
visi
.org
chan
00
68840   0.0053438160   0.1189526030   0.8823678210]
```

Time

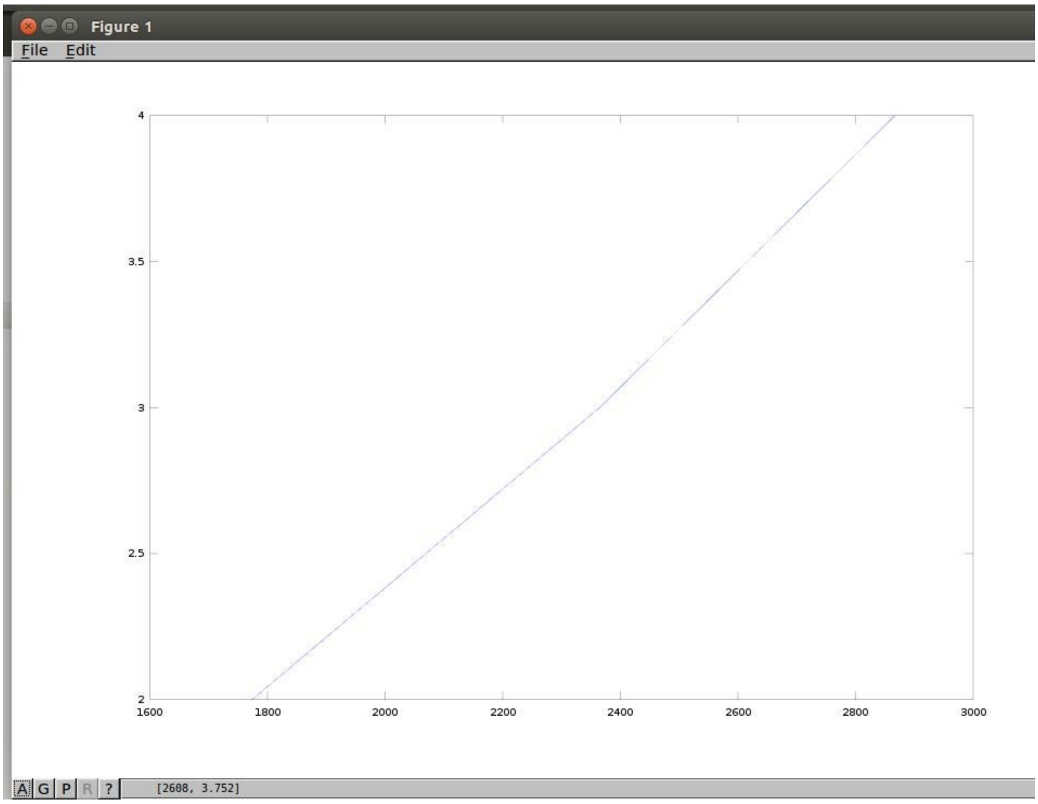
Speedup Vs Problem Size   Maximum speedup=2.30
Speedup on y axis



log(ProblemSize)

Number of cores V/s Speedup

Number
of cores



Speedup

ProblemSize=100000