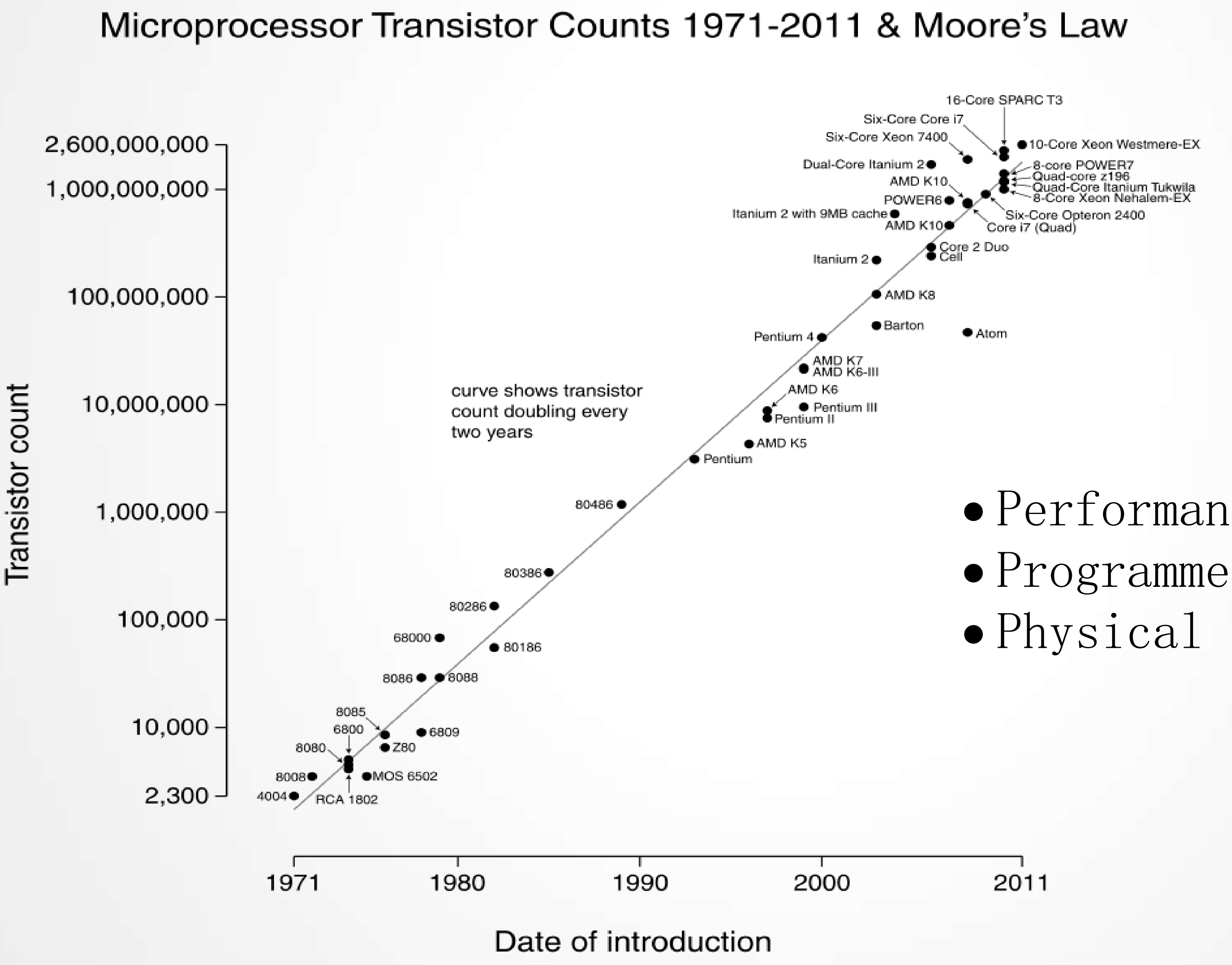
Extending GCC

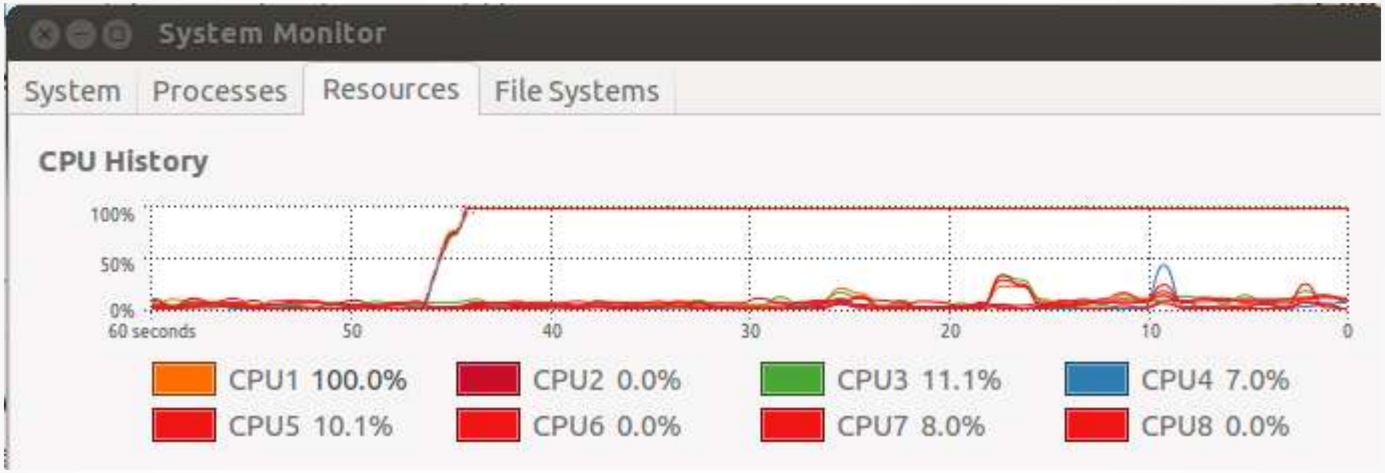
- Yajnesh T

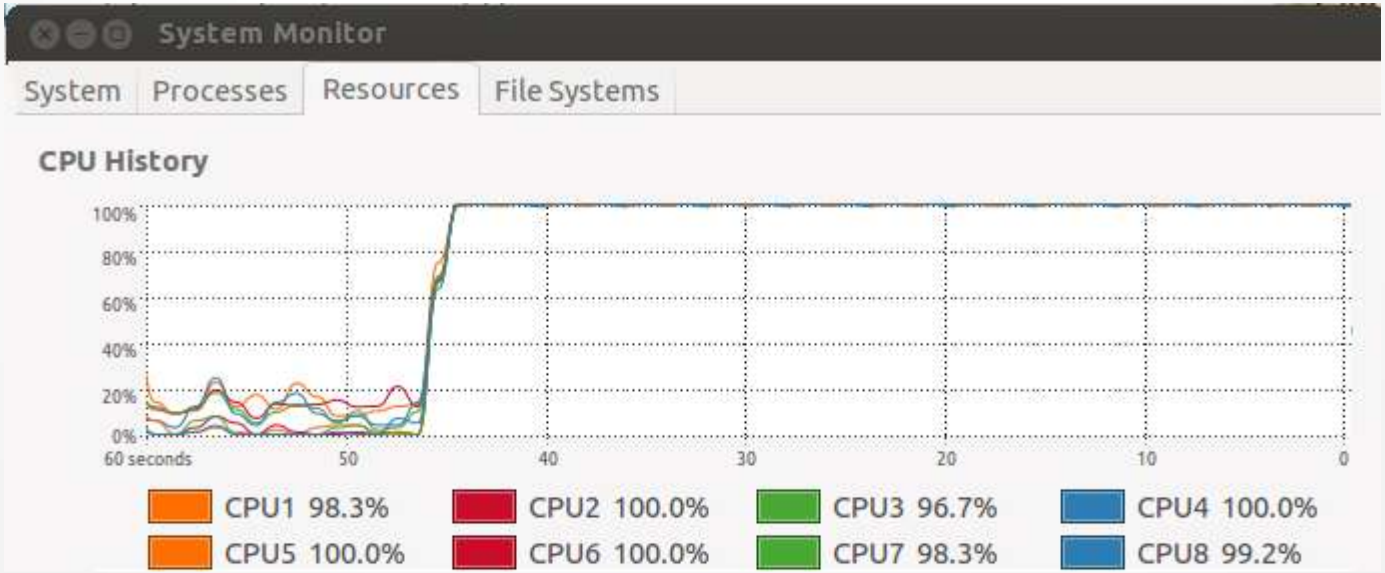
Moore's Law


Microprocessor Transistor Counts 1971-2011 & Moore's Law

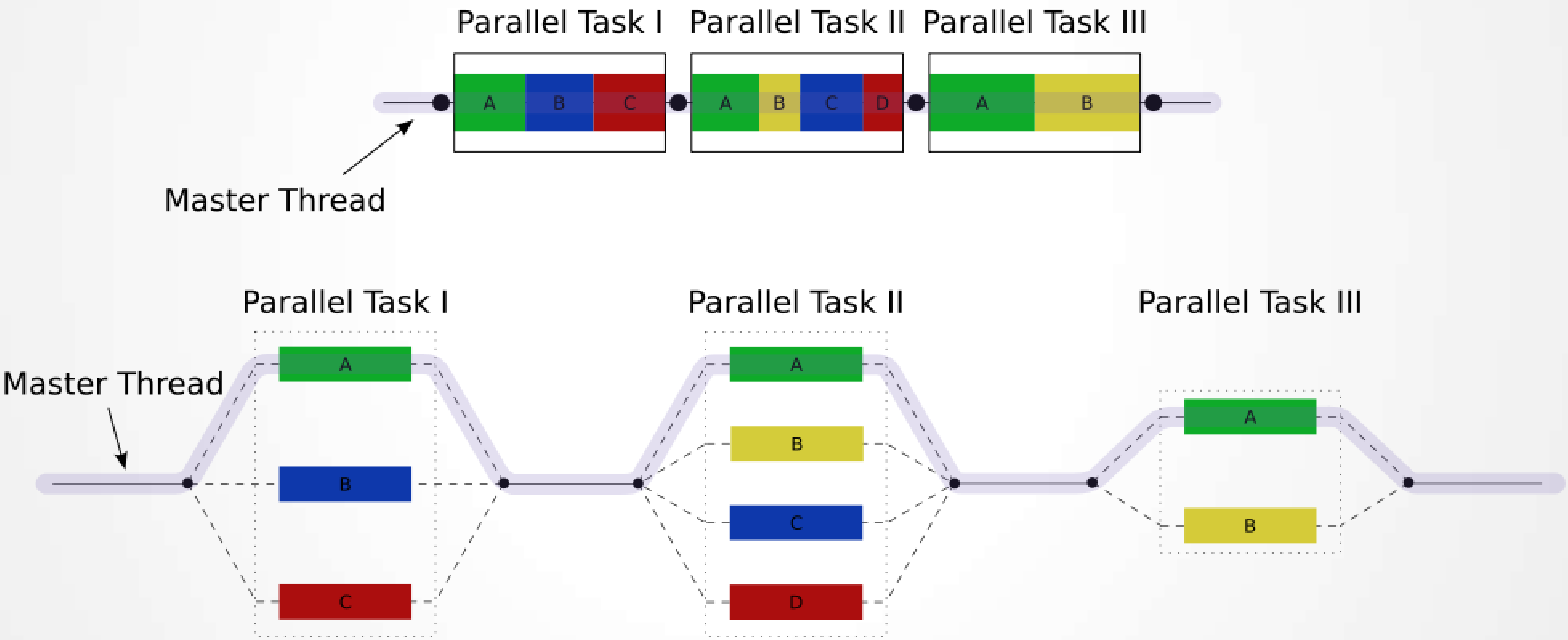
- Performance
- Programmers Perspective
- Physical Limit

What do we achieve in Parallelization (general)





What do we achieve in Parallelization



OpenMP

- OpenMP is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran.

Problems faced while parallelizing

- Tracing the program becomes difficult.
- Debugging will be complicated.
- **Once the code is parallelized , along with the speed, the complexity of the code increases.**
- This results in increased overall human time & effort spent on a problem.

Auto Parallelization

- Let the compiler do the job.

OR

- Guided compilation

Modify gcc

- Analyze the internals of gcc to add our own pass.
- Build local gcc
- $gcc input.c -fparallelize

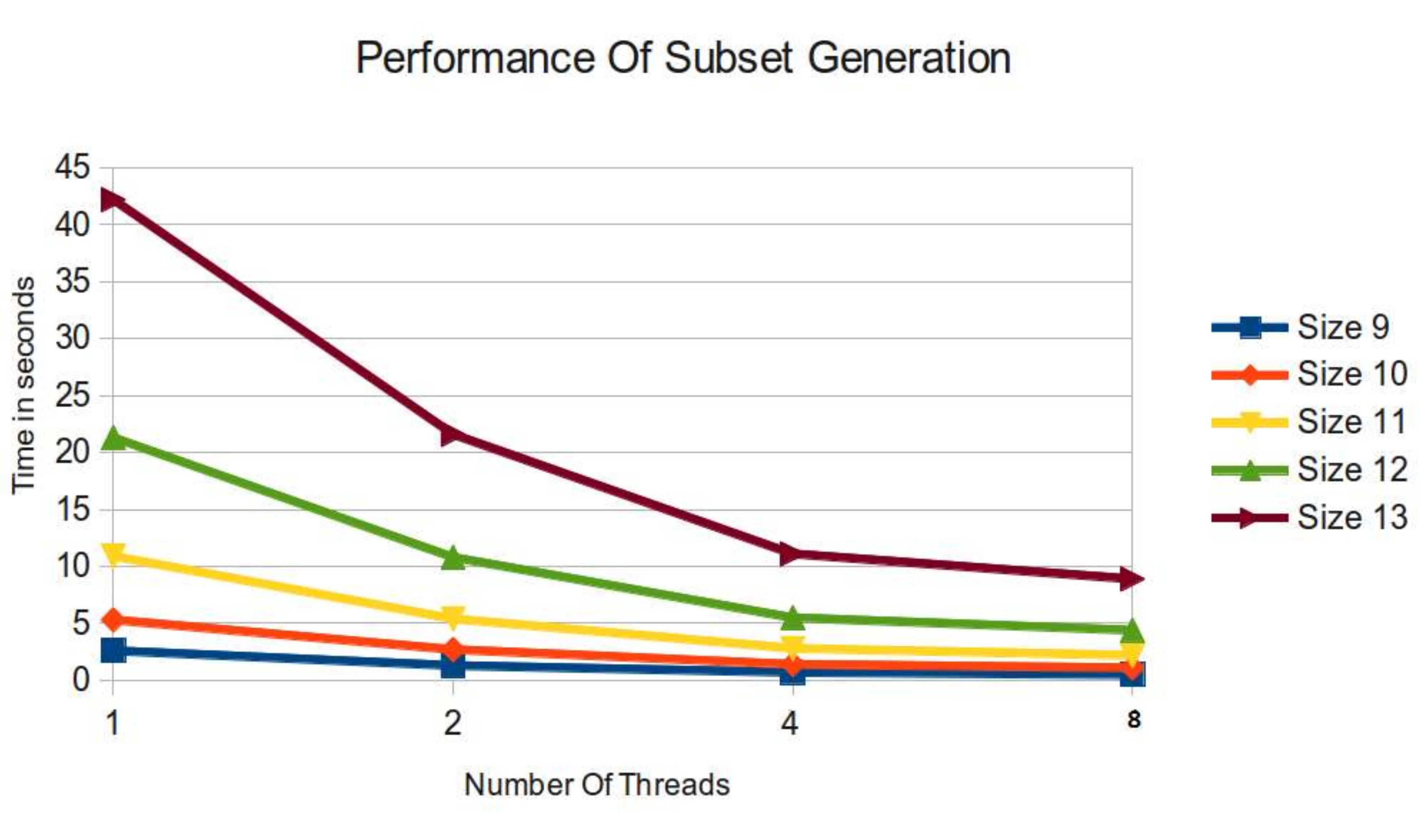
Command line options

```
$gcc –fparallelize
```

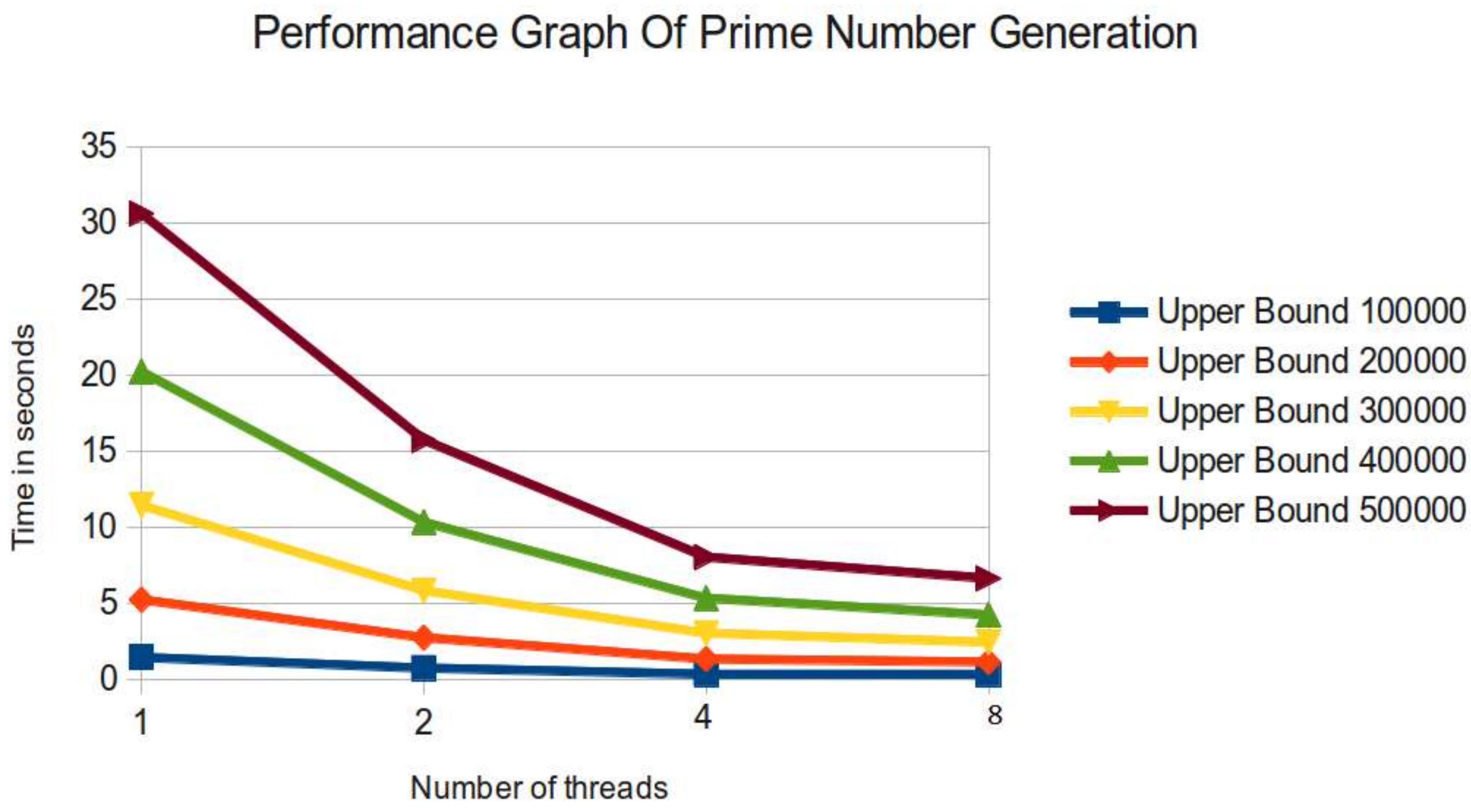
```
$gcc –fparallelize –fnested
```

```
$gcc -fparallelize -finteractive
```

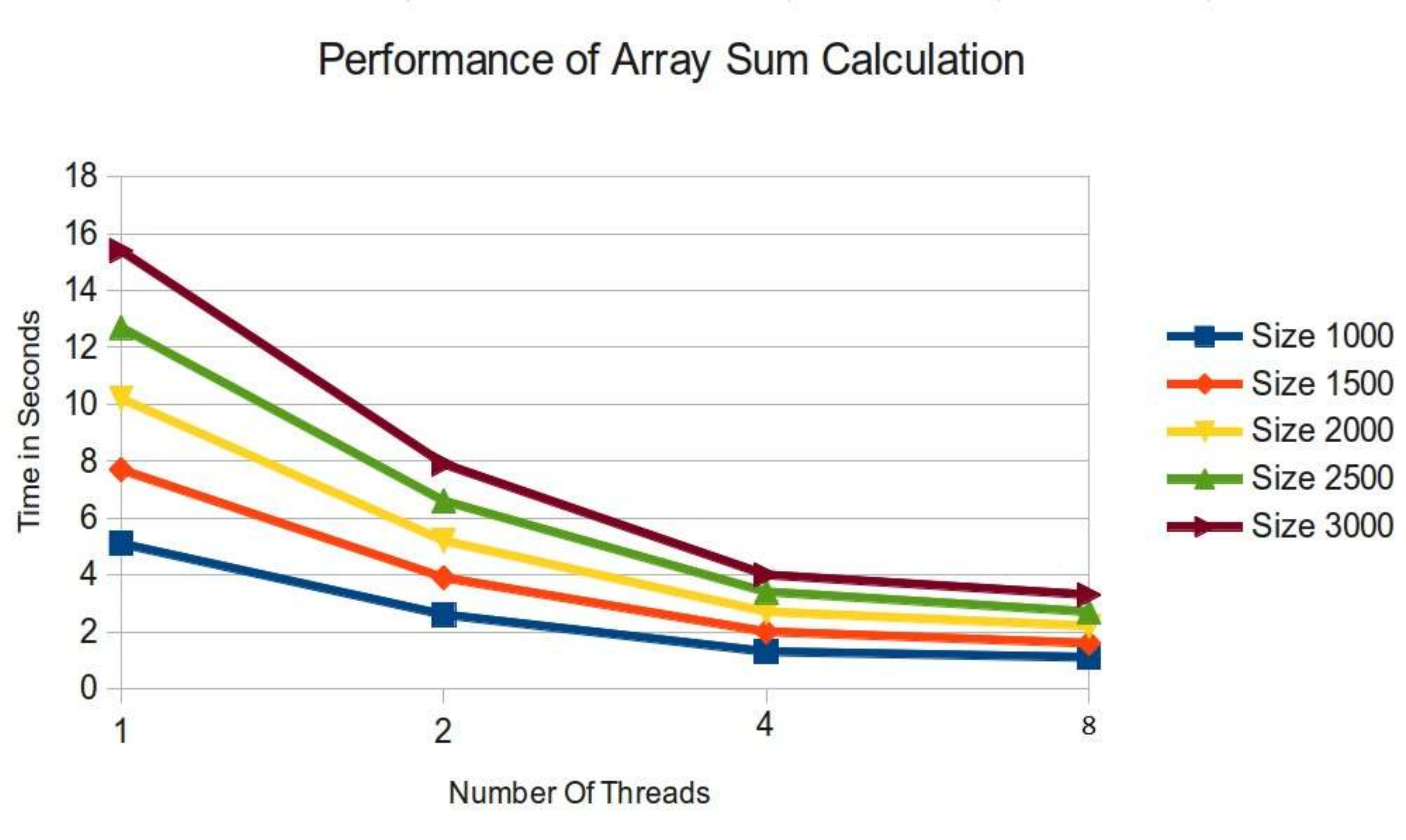
Results:


Performance Of Subset Generation
Time in seconds
45
40
35
30
25
20
15
10
5
0
1
2
4
8
Number Of Threads
Size 9
Size 10
Size 11
Size 12
Size 13

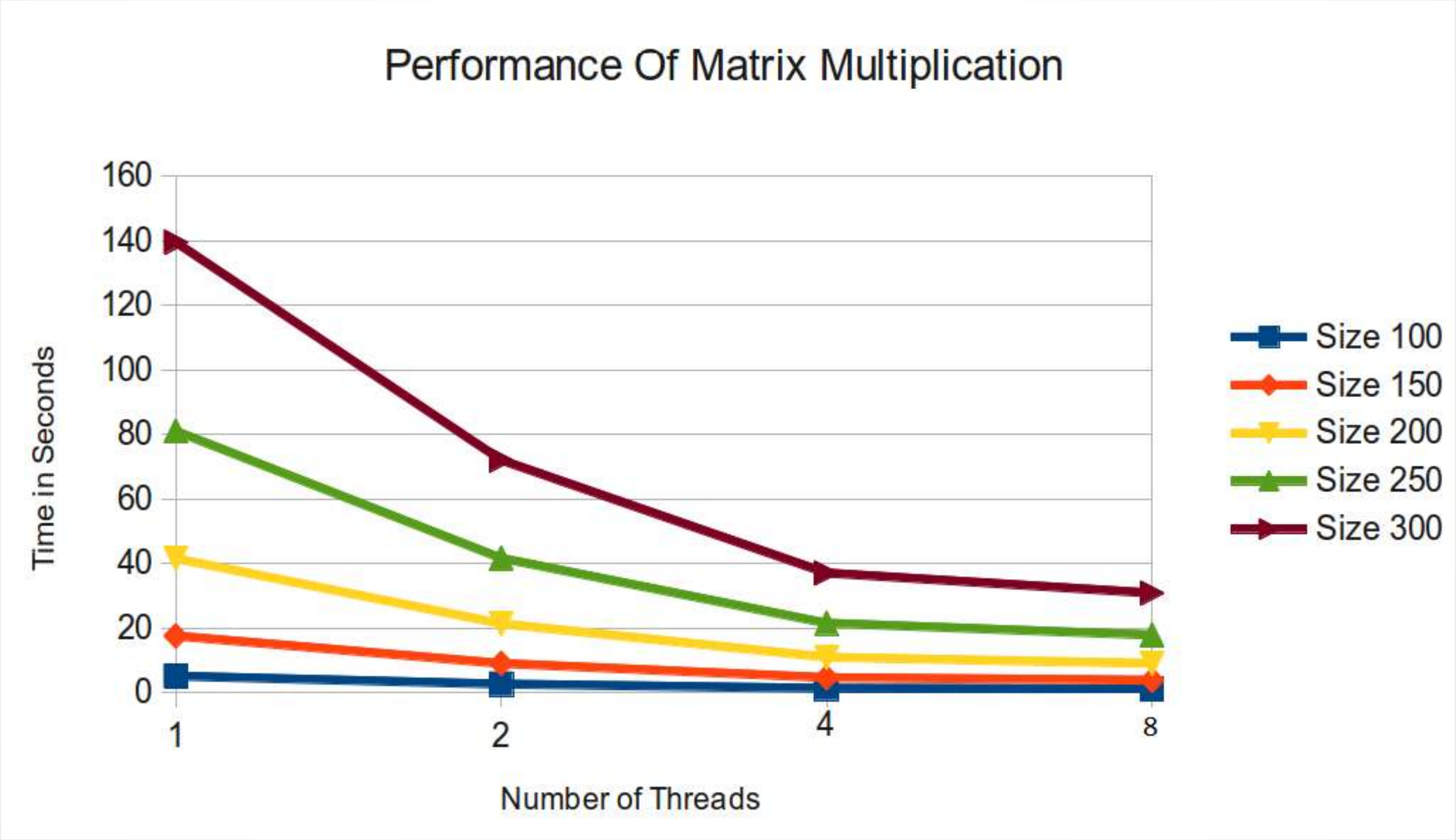
Results:


Performance Graph Of Prime Number Generation
35
30
25
20
15
10
5
0
Time in seconds
1
2
4
8
Number of threads
Upper Bound 100000
Upper Bound 200000
Upper Bound 300000
Upper Bound 400000
Upper Bound 500000

Results:

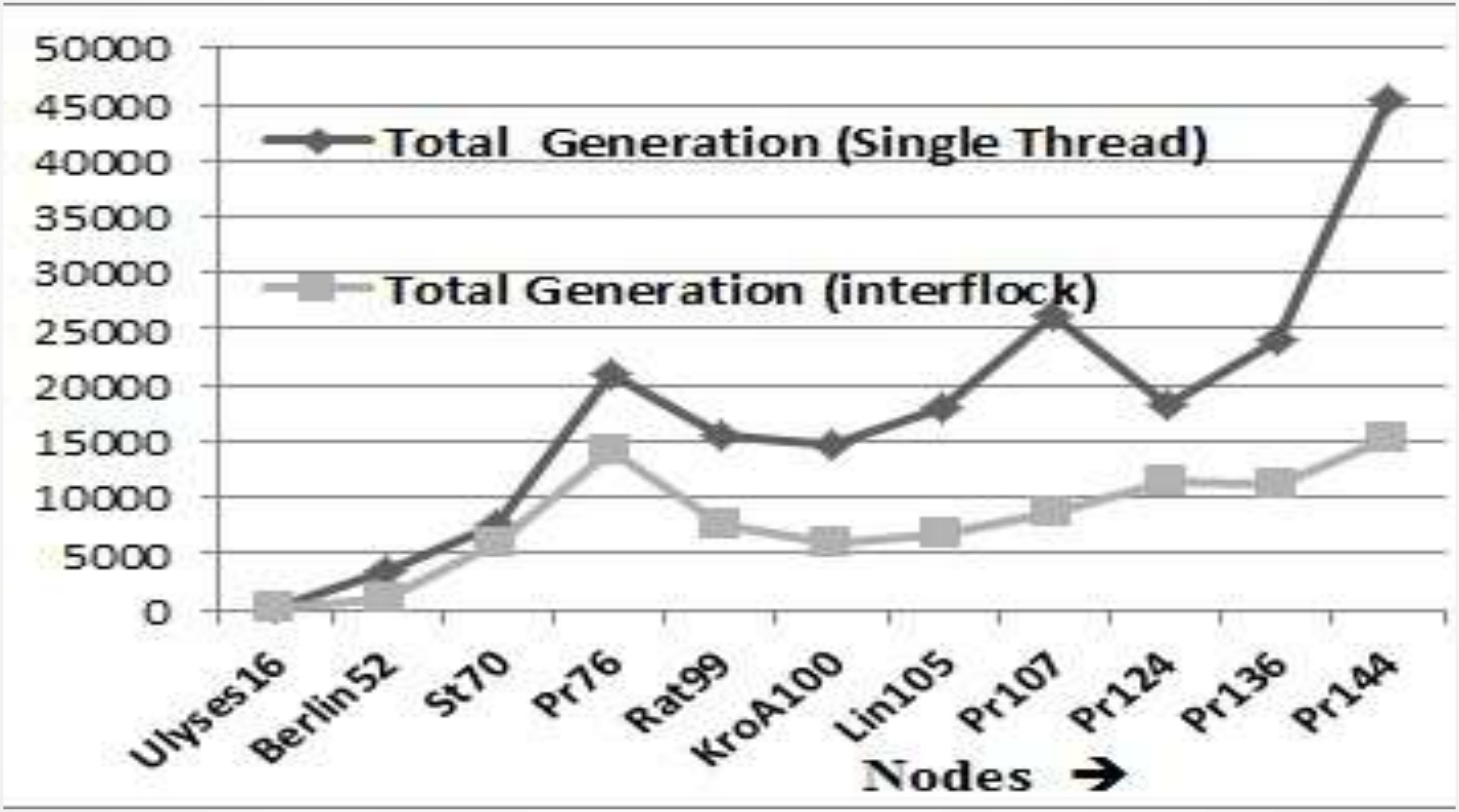

Performance of Array Sum Calculation
18
16
14
12
10
8
6
4
2
0
Time in Seconds
1
2
4
8
Number Of Threads
Size 1000
Size 1500
Size 2000
Size 2500
Size 3000

Results:


Performance Of Matrix Multiplication
160
140
120
100
80
60
40
20
0
Time in Seconds
1
2
4
8
Number of Threads
Size 100
Size 150
Size 200
Size 250
Size 300

Results:

Interflock-Crossover


Total Generation (Single Thread)
Total Generation (interflock)
Ulyses16
Berlin52
St70
Pr76
Rat99
KroA100
Lin105
Pr107
Pr124
Pr136
Pr144
Nodes →