Project Title: Fourier Analysis using MPI

Name: Nirmit Patel

Email: patenirm@oregonstate.edu

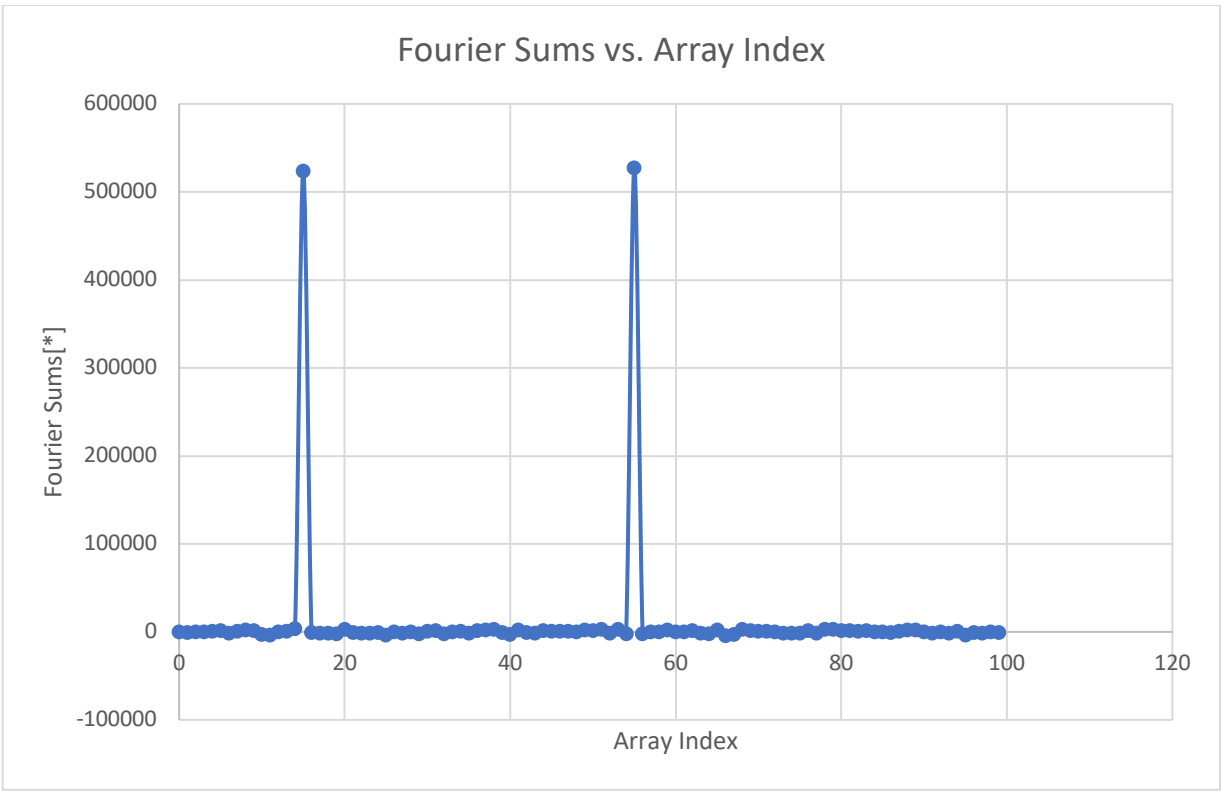
I executed the program on the Rabbit server. The program generated the following data during execution, and the graph below illustrates the results.

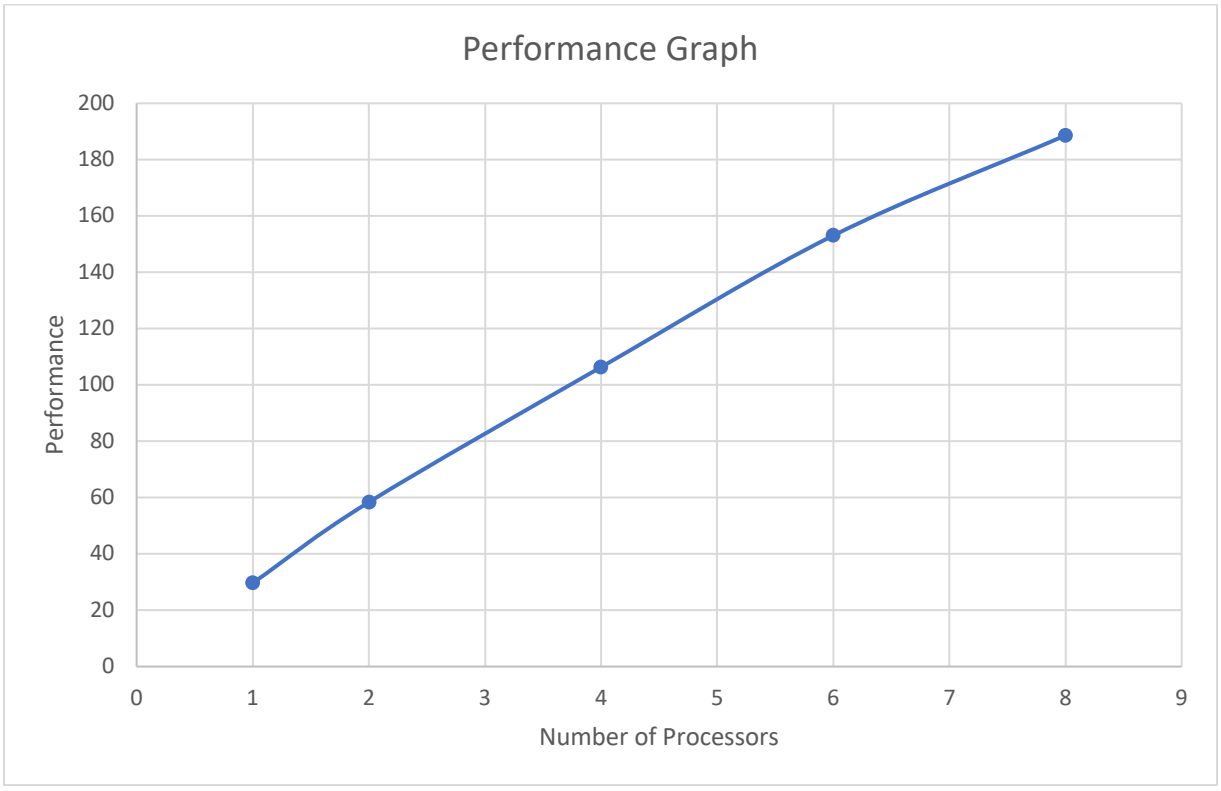
Array Index	Fourier Sums
0	0
1	-536.19
2	137.82
3	176.88
4	675.07
5	1216.93
6	-1498.37
7	894.27
8	2540
9	1484.96
10	-2507.93
11	-3577.06
12	-29.75
13	773.55
14	3748.67
15	523753.66
16	-1005.35
17	-1441.7
18	-1138.32
19	-1874.7
20	3003.55
21	-569.27
22	-1065.43
23	-1165.94
24	-481.43
25	-3455.39
26	247.49
27	-1486.51
28	-312.52
29	-1999.94
30	519.21
31	1289.24
32	-1872.96
33	-195.01

Array Index	Fourier Sums
34	1004.31
35	-1200.4
36	1175.9
37	2242.02
38	2691.74
39	-589.91
40	-2534.23
41	1998.31
42	-676.91
43	-1061.33
44	1145
45	724.76
46	564.97
47	898.93
48	-97.79
49	2489.34
50	1822.98
51	2717.08
52	-1122.09
53	2826.57
54	-1803.31
55	527281.62
56	-1964.45
57	-161.83
58	-226.77
59	2186.18
60	338.97
61	371.85
62	1668.95
63	-1454.62
64	-1793.9
65	2365.24
66	-4431.36
67	-2778.95

Array Index	Fourier Sums
68	3087.57
69	1462.38
70	544.74
71	1130.79
72	205.52
73	-1488.54
74	-1270.45
75	-1449.15
76	1800.34
77	-1456.65
78	2955.83
79	2825.42
80	1506.03
81	1169.04
82	541.46
83	1309.91
84	287.87
85	-303.36
86	-1009.35
87	798.53
88	2057.42
89	1929.61
90	-271.97
91	-1378.15
92	-263.39
93	-1065.8
94	1053.08
95	-3344.41
96	-744.22
97	-1571.62
98	-167.89
99	-1031.42

```
Node   0 entered DoOneLocalFourier( )
  1 processors,    1048576 elements,     29.67 mega-multiplies
computed per second
Node   0 entered DoOneLocalFourier( )
Node   1 entered DoOneLocalFourier( )
  2 processors,    1048576 elements,     58.38 mega-multiplies
computed per second
Node   1 entered DoOneLocalFourier( )
Node   2 entered DoOneLocalFourier( )
Node   0 entered DoOneLocalFourier( )
Node   3 entered DoOneLocalFourier( )
  4 processors,    1048576 elements,    106.34 mega-multiplies
computed per second
Node   1 entered DoOneLocalFourier( )
Node   2 entered DoOneLocalFourier( )
Node   3 entered DoOneLocalFourier( )
Node   4 entered DoOneLocalFourier( )
Node   0 entered DoOneLocalFourier( )
Node   5 entered DoOneLocalFourier( )
  6 processors,    1048576 elements,    153.07 mega-multiplies
computed per second
Node   1 entered DoOneLocalFourier( )
Node   2 entered DoOneLocalFourier( )
Node   3 entered DoOneLocalFourier( )
Node   4 entered DoOneLocalFourier( )
Node   5 entered DoOneLocalFourier( )
Node   6 entered DoOneLocalFourier( )
Node   0 entered DoOneLocalFourier( )
Node   7 entered DoOneLocalFourier( )
  8 processors,    1048576 elements,    188.61 mega-multiplies
computed per second
```


Fourier Sums vs. Array Index
600000
500000
400000
300000
200000
100000
0
-100000
Fourier Sums[*]
0
20
40
60
80
100
120
Array Index


Performance Graph
200
180
160
140
120
100
80
60
40
20
0
Performance
0
1
2
3
4
5
6
7
8
9
Number of Processors

The secret sine-waves' periods are at array index 15 and 55.

In the performance graph, we observe an increasing trend in performance as the number of processors increases. As the number of processors doubles from 1 to 2, there is a significant improvement in performance. However, the performance gains become less pronounced as the number of processors continues to increase, indicating diminishing returns.

The performance improvements with an increasing number of processors can be attributed to parallelization and the ability to distribute the workload among multiple processors. With more processors, the computational tasks can be divided and executed concurrently, leading to faster completion times. The initial significant performance improvement from 1 to 2 processors is likely due to the utilization of additional resources and reduced contention. However, as the number of processors further increases, factors such as communication overhead and resource sharing may start to limit the scalability and result in diminishing performance gains.