Assignment#9 report: MPI Scaling Analysis

Introduction

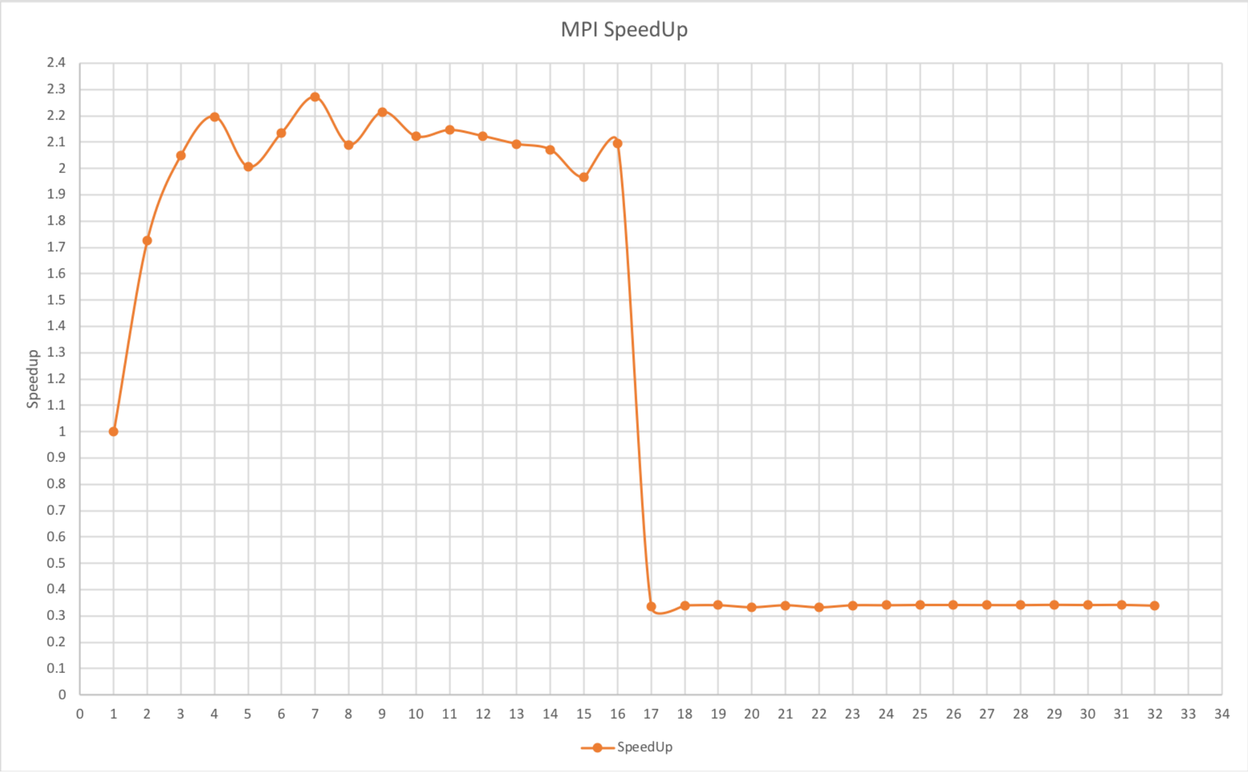
In this project we are using the same code that we used in the previous assignment but parallelized it using MPI instead to see how It differs from OpenMP in both implementation and performance. In MPI parallel implementation, we parallelize the whole code instead of just parallelizing the for loops. So, each processor is solving the same problem in its chunk. These parallel tasks communicate the required information at some specific points (Ghost/Guard cells) that could potentially become bottle neck and overheads.

Scaling Analysis

The presented table bellow shows the scaling analysis data over 2 nodes (32 CPUs), And also speed up plot is shown subsequently.

Number of cores	Total time(s)	Speed-up
1	106.2574	1
2	61.5134	1.727386228
3	51.8408	2.049686733
4	48.3686	2.19682604
5	52.9883	2.005299283
6	49.7482	2.135904415
7	46.7969	2.270607668
8	50.8434	2.08989564
9	47.9897	2.214170958
10	50.0945	2.121139047
11	49.5319	2.145231659
12	50.0784	2.121820985
13	50.7845	2.092319507
14	51.3217	2.070418556
15	54.0613	1.965498425
16	50.7074	2.095500854
17	315.9931	0.336264937
18	312.5819	0.33993459
19	312.137	0.340419111
20	319.2651	0.332818714
21	312.2952	0.340246664
22	319.5508	0.332521152
23	312.2573	0.340287961
24	312.189	0.340362409
25	311.2693	0.341368069
26	311.3864	0.341368069
27	311.5891	0.341017706
28	311.6977	0.34089889

29	310.7968	0.341887046
30	311.4271	0.341195098
31	311.0391	0.341620716
32	313.7195	0.33870193



Discussion

As you can see in the chart above, the code starts to scale quite acceptable in early stages until 4 cores (Not perfect, but acceptable). After, the speedup value starts to oscillate around the same value. One of the most important reasons of that could be the effect of over communications. As the number of cores increases to more than 5, ghost cells increases and as a results the communications between cores increases that makes overheads. At this point, overheads dominate and no more significant speed-up would be achievable, and even it drops as the number of cores increases to 16. When we use more than 16 cores, the performance drops drastically and experiences a lot of overhead. Owing to the fact that each node of the teach cluster contains 16 cores, as we go from 16 cores to 17 cores, there is clearly communication involved, and communicating outside of the node (even using a fast network connection) will never be as fast as communicating within a node. Therefore, the off-node communication overheads here dominate and we see some drop in the performance when going to two nodes; as of now, we can hardly see any speed up.

Surely, this will ultimately depend on the problem and how much work there is to do per processors, we may be able to see better performance by increase each core's work same as previous assignment.