Cpt S 411 Assignment Cover Sheet

(To be turned in along with each homework and program project submission)

Assignment # 2

For individual assignments:

Student name (Last, First): Cuevas, Jessica

For team projects:

List of all students (Last, First): Cuevas, Jessica

List of collaborative personnel (excluding team participants):
N/A

I[1] certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received or given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment. I also certify that I have not referred to online solutions that may be available on the web or sought the help of other students outside the class, in preparing my solution. I attest that the solution is my own and if evidence is found to the contrary, I understand that I will be subject to the academic dishonesty policy as outlined in the course syllabus.

Please print your names.
Jessica Cuevas

Assignment Project Participant(s):
Jessica Cuevas

Today's Date:
10/27/20

[1] If you worked as a team, then the word "I" includes yourself and your team members.

Programming Project 3 Report

Table and Chart Results:

Table 1. Run-time of My All Reduce(microsecs)

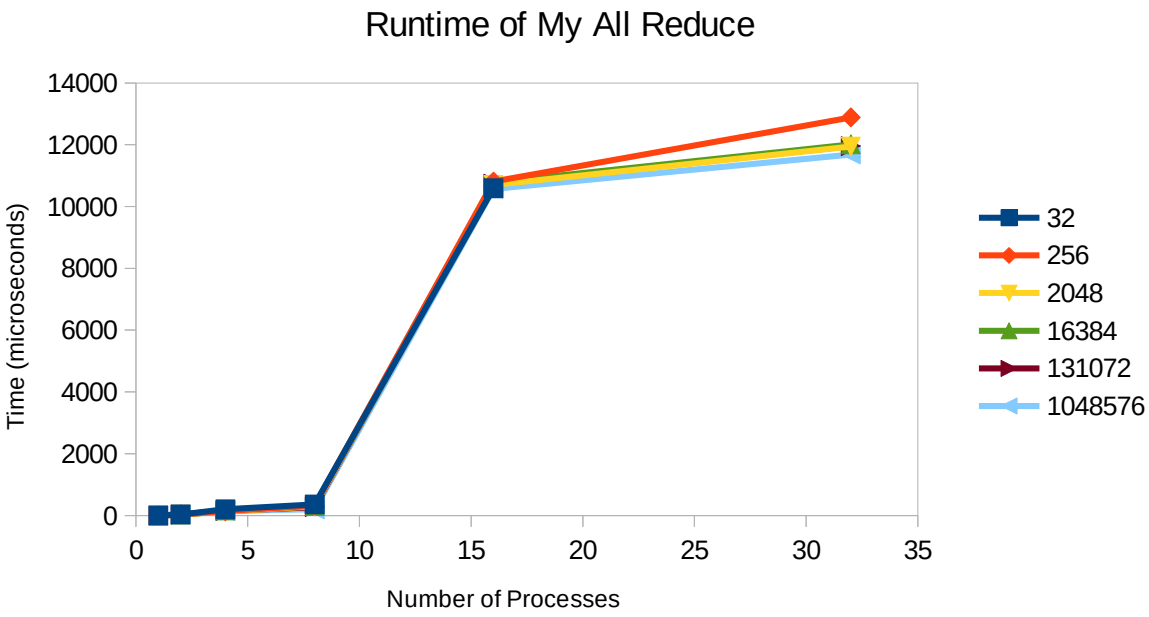
Size (n)	p=1	p=2	p=4	p=8	p=16	p=32
32	0	33	202	359	10594	NA
256	0	34	137	317	10808	12883
2048	0	30	126	302	10697	11941
16384	0	30	165	308	10781	12013
131072	1	28	146	276	10736	11956
1048576	1	38	140	203	10570	11683

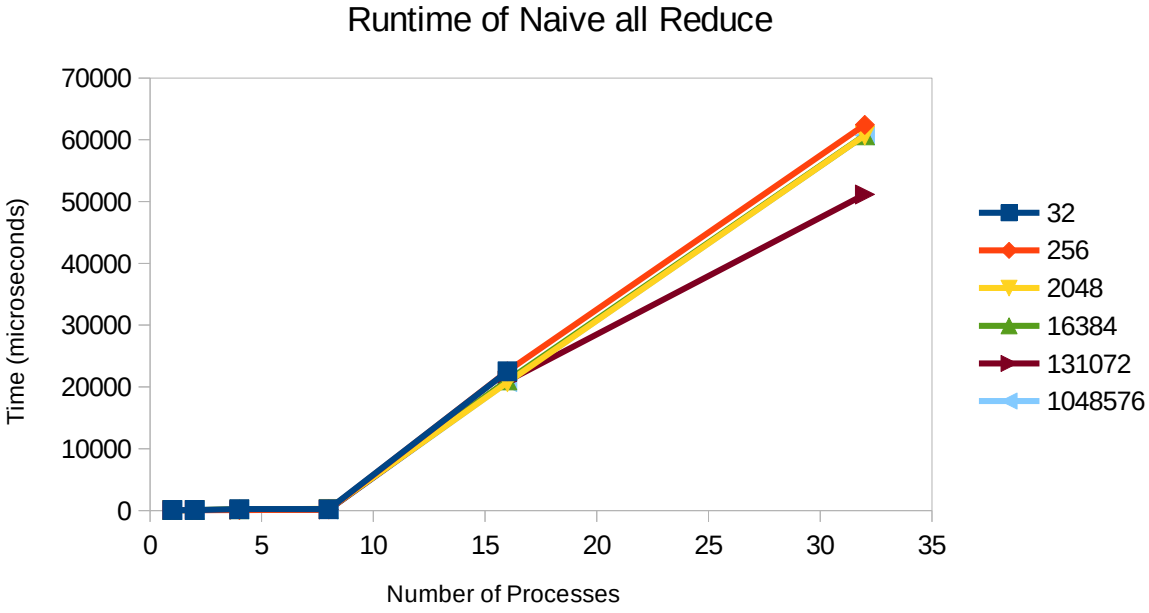
Table 2. Run-time of Naive All Reduce(microsecs)

Size (n)	p=1	p=2	p=4	p=8	p=16	p=32
32	64	64	227	230	22513	NA
256	64	64	84	169	22539	62422
2048	64	64	143	260	20746	60689
16384	64	64	184	237	21043	60737
131072	64	64	104	168	20957	51149
1048576	64	64	156	145	20781	60692

Table 3. Run-time of MPI All Reduce(microsecs)

Size (n)	p=1	p=2	p=4	p=8	p=16	p=32
32	3	34	171	260	10759	NA
256	4	37	169	281	11557	13006
2048	4	38	148	278	10762	11935
16384	3	38	167	222	10700	13086
131072	4	34	142	232	10864	12233
1048576	7	69	128	230	10692	13820


Runtime of My All Reduce
Time (microseconds)
14000
12000
10000
8000
6000
4000
2000
0
0
5
10
15
20
25
30
35
Number of Processes
32
256
2048
16384
131072
1048576


Runtime of Naive all Reduce
Time (microseconds)
70000
60000
50000
40000
30000
20000
10000
0
0
5
10
15
20
25
30
35
Number of Processes
32
256
2048
16384
131072
1048576

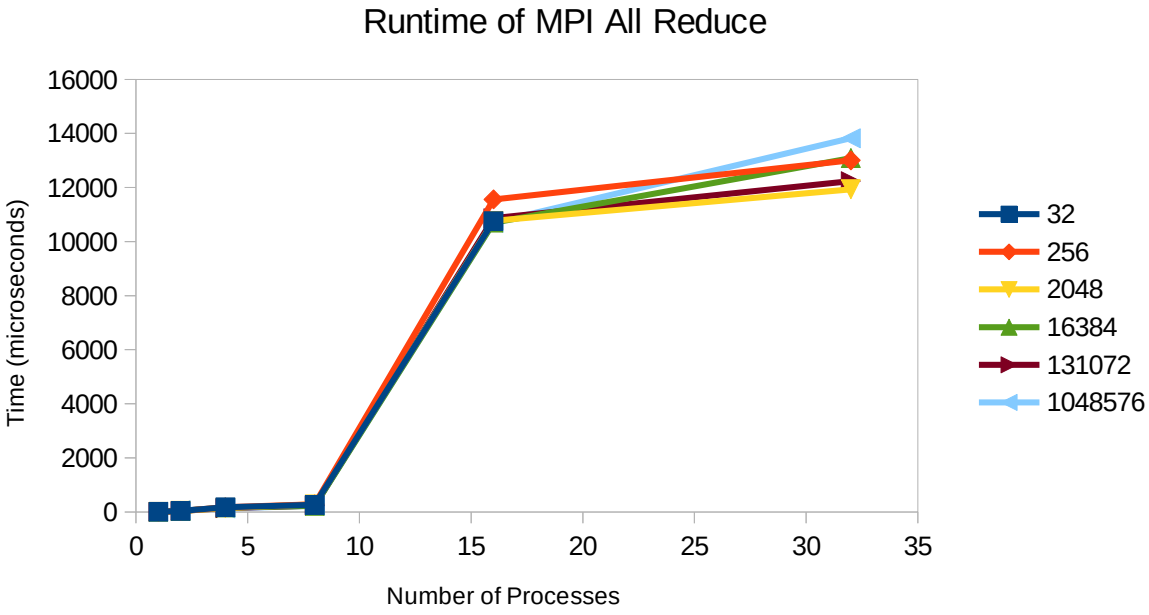

Runtime of MPI All Reduce
Time (microseconds)
16000
14000
12000
10000
8000
6000
4000
2000
0
0
5
10
15
20
25
30
35
Number of Processes
32
256
2048
16384
131072
1048576

Table 4. *All* Reduce Speedup (T(n,1) / T(n,p))

Size (n)	p=1	p=2	p=4	p=8	p=16	p=32
32	0.00000	0.00000	0.00000	0.00000	0.00000	NA
256	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2048	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
16384	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
131072	1.00000	0.03571	0.00685	0.00362	0.00009	0.00008
1048576	1.00000	0.02632	0.00714	0.00493	0.00009	0.00009

Table 5. Naive All Reduce Speedup (T(n,1) / T(n,p))

Size (n)	p=1	p=2	p=4	p=8	p=16	p=32
32	1	1	0.281938326	0.2777777778	0.0028428019	NA
256	1	1	0.7619047619	0.3786982249	0.0028394722	0.001025273
2048	1	1	0.4475524476	0.2461538462	0.003084932	0.0010545568
16384	1	1	0.347826087	0.2700421941	0.0030413914	0.0010537234
131072	1	1	0.6153846154	0.380952381	0.0030538722	0.0012512464
1048576	1	1	0.4102564103	0.4413793103	0.0030797363	0.0010545047

Table 6. MPI All Reduce Speedup (T(n,1) / T(n,p))

Size (n)	p=1	p=2	p=4	p=8	p=16	p=32
32	1	0.0901162791	0.018096906	0.0119322556	0.0002881335	NA
256	1	0.095890411	0.0207715134	0.0124688279	0.0003028415	0.0002691086
2048	1	0.1093333333	0.0277401894	0.0147694524	0.0003809701	0.0003435361
16384	1	0.0873015873	0.0197841727	0.0148916968	0.0003084141	0.000252176
131072	1	0.1065088757	0.0253699789	0.0155105558	0.0003313666	0.0002942956
1048576	1	0.0937950938	0.050623053	0.0282977797	0.0006079084	0.0004703226

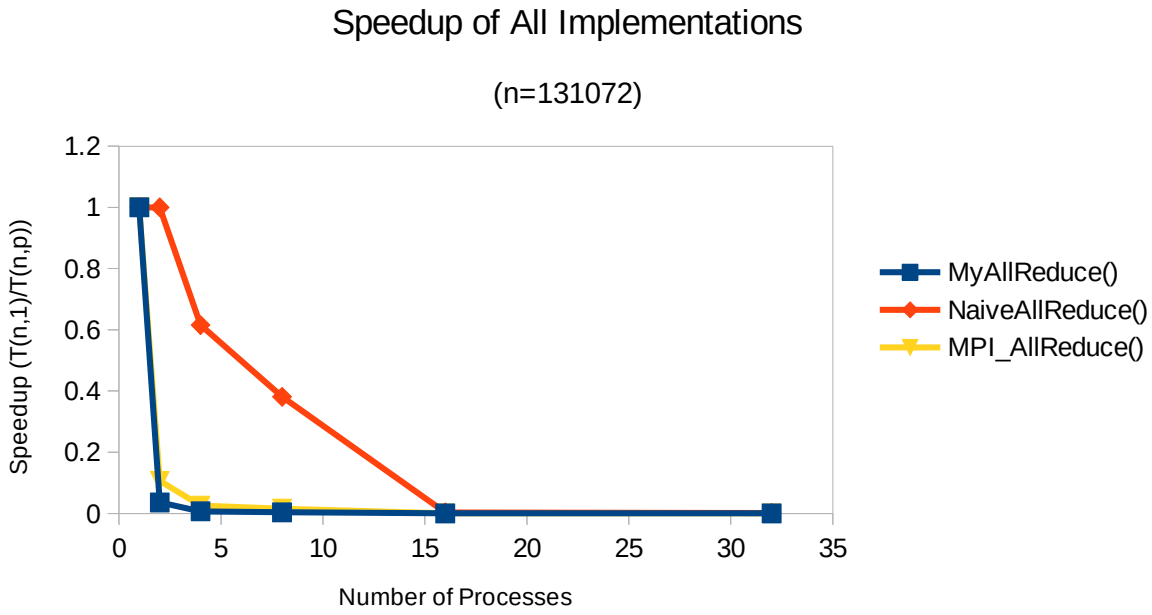

Speedup of All Implementations
(n=131072)
Speedup (T(n,1)/T(n,p))
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
30
35
Number of Processes
MyAllReduce()
NaiveAllReduce()
MPI_AllReduce()

Table 7. My All Reduce Efficiency (S/p)

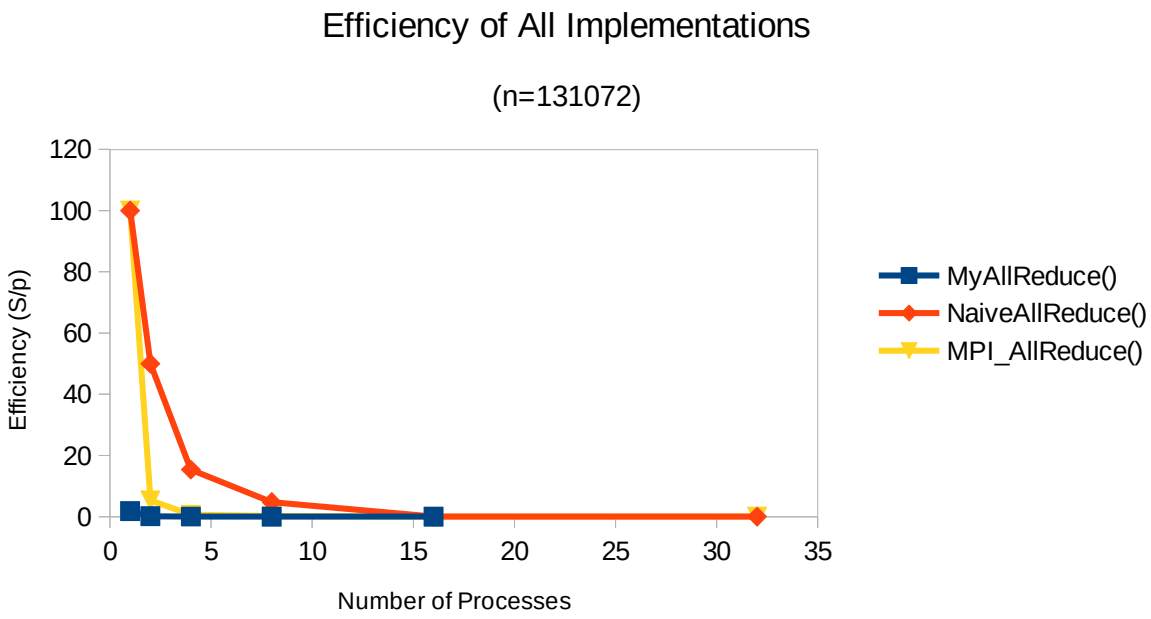
Size(n)	p=1	p=2	p=4	p=8	p=16	P=32
32	0	0	0	0	0	NA
256	0	0	0	0	0	0
2048	0	0	0	0	0	0
16384	0	0	0	0	0	0
131072	100	1.7857142857	0.1712328767	0.0452898551	0.0005821535	0.000261375
1048576	100	1.3157894737	0.1785714286	0.0615763547	0.0005912961	0.0002674827

Table 8. Naive All Reduce Efficiency (S/p)

Size(n)	p=1	p=2	p=4	p=8	p=16	p=32
32	100	50	7.0484581498	3.4722222222	0.0177675121	NA
256	100	50	19.047619048	4.7337278107	0.0177467013	0.0032039781
2048	100	50	11.188811189	3.0769230769	0.0192808252	0.0032954901
16384	100	50	8.6956521739	3.3755274262	0.0190086965	0.0032928857
131072	100	50	15.384615385	4.7619047619	0.0190867013	0.0039101449
1048576	100	50	10.256410256	5.5172413793	0.0192483519	0.0032953272

Table 9. MPI All Reduce Efficiency (S/p)

Size(n)	p=1	p=2	p=4	p=8	p=16	p=32
32	100	4.5058139535	0.4524226503	0.1491531948	0.0018008347	NA
256	100	4.7945205479	0.5192878338	0.1558603491	0.0018927595	0.0008409645
2048	100	5.4666666667	0.6935047361	0.1846181556	0.002381063	0.0010735502
16384	100	4.3650793651	0.4946043165	0.1861462094	0.0019275881	0.0007880499
131072	100	5.325443787	0.6342494715	0.1938819474	0.0020710413	0.0009196737
1048576	100	4.6897546898	1.265576324	0.3537222464	0.0037994276	0.0014697583


Efficiency of All Implementations
(n=131072)
Efficiency (S/p)
120
100
80
60
40
20
0
0
5
10
15
20
25
30
35
Number of Processes
MyAllReduce()
NaiveAllReduce()
MPI_AllReduce()

Interpretation of Results:

I. Run-times of All Implementations:

What I primary noticed and expected in the run-times was that my implementation of all reduce was the fastest among all implementations regardless of the size of the array and the number of processes used. However I was surprised that the MPI build in implementation was slightly slower than the hyper-cubic permutation. I expected it to be the fastest or at pair with my build in implementation. I expected the Naive implementation to be the slowest and after testing it clearly ended as is.

II. Speedup:

For the speedup of each implementation, no matter the number of processes all my values were very small, but once the size of the array increases we can observe a strong scaling. The speedup was no ideal from any of the implementations.

III. Efficiency:

The efficiency curve that we observe on each implementation were expected. The most efficient implementation (myAllReduce) stayed clearly flat and neutral, while the other two implementations started high and decreased the increase of processes.