CS/ECE 566 Lab 1

Vitaly Parnas

Fall 2015

October 9, 2015

Contents

1 Overview

The application measures the time to sum a randomly generated n-number array in three different parallel topologies: *Ring*, *Hypercube*, and a 2-D *mesh* with wraparound, and two parallel formulations: *scatter and reduce* and *scatter and gather*. The executable, *lab1*, is parameterized to execute, in one invocation, all formulations for one or variable number of input sizes, and any one parallel topology. Additionally, each summation attempt for any combination of parameters is conducted a total of three times to account for fluctuation in computational or network resources. The following is the command line syntax:

```
$ ./lab1
usage: ./lab1 [-n <base input size>] [-x <num input sizes] [-t <topology> (R, M, H)]
[-m <max array value>]
    -n <base input size> (default 100) - initial input size to experiment.
    -x <num input sizes> (default 1) - starting from the base size, experiment with this
many different input sizes incrementing in powers of two.
    -t <topology> - R/ring, M/mesh, H/hypercube (default).
    -m <max array value> (default 1000) - Maximum value for each random array element.
```

For example, the following, invoked on one node for demonstration purposes, executes all formulations in a hypercube topology for three input sizes: 1024, 2048, and 4096, and prints out the resulting metrics.

```
$ ./lab1 -n 1024 -x 3 -t H

1 total nodes
Node 0: topology = 0-dim hypercube.
1 nodes present in topology

...... Debug information ommitted ------

------------------Runtime summary (usecs) --------------------------------
   N |    Formulation |       Mean |    Minimum |        Max |  Std. Dev.
---------------------------------------------------------------------------
1024 |          serial |     3.0994 |     3.0994 |     3.0994 |     0.0000
1024 |  scatter-reduce |     5.6426 |     2.8610 |    10.9673 |     3.7664
1024 |  scatter-gather |     4.0531 |     3.0994 |     5.0068 |     0.7787
2048 |          serial |     5.7220 |     5.0068 |     6.1989 |     0.5150
2048 |  scatter-reduce |     8.3447 |     5.9605 |    12.8746 |     3.2046
2048 |  scatter-gather |     5.9605 |     5.9605 |     5.9605 |     0.0000
4096 |          serial |    11.4441 |    11.2057 |    11.9209 |     0.3372
4096 |  scatter-reduce |    13.1130 |    12.1593 |    15.0204 |     1.3487
4096 |  scatter-gather |    12.0004 |    11.9209 |    12.1593 |     0.1124
```

The arrays are created randomly, in the range of $[0, max]$, where max is indicated via the m parameter, or set to 1000 by default.

2 Formulations

The general MPI calls involved in the application and the setup of the formulations below are the following:

MPI_Init
MPI_Comm_rank
MPI_Comm_size
MPI_Cart_create
MPI_Wtime
MPI_Comm_free
MPI_Finalize

Table 1: General MPI Calls

2.1 Scatter-Reduce

The *Scatter-Reduce* formulation scatters all array elements $A_1, A_2, ..., A_n$ from the root node across all p nodes, $n > p$, such that each node $N_i \in \{N_1, N_2, ..., N_p\}$ receives n/p elements. Each n/p element block of numbers is then serially added by each node, and the local sums are globally summed and transmitted across the topology back to the root via the *Reduce* operation. Naturally, in all the experiments, n was chosen to be a multiple of p.

Table 2 lists the MPI calls involved in this formulation beyond the general calls in Table 1.

MPI_Scatter
MPI_Reduce

Table 2: Scatter-Reduce formulation MPI Calls

2.2 Scatter-Gather

The *Scatter-Gather* formulation, similarly to *Scatter-Reduce*, distributes all array elements across all nodes. However, once each node locally computes the n/p element sum, the local sums are transmitted across the topology back to the root via the *Gather* operation, and the root computes the global sum from all the received local sums.

Table 3 lists the MPI calls involved in this formulation beyond the general calls in Table 1.

MPI_Scatter
MPI_Gather

Table 3: Scatter-Gather formulation MPI Calls

2.3 Serial addition

Along with the two parallel formulations above, serial addition is always performed at the root node for all input sizes. This helps to not only compare the parallel and serial performance, but also verify correctness by comparing the parallel and serial results, visible in the debug output. If, for example, the parallel summation is not properly carried out, in the case where one parallel node prematurely terminates, or the numbers are not correctly distributed across nodes, the serial and parallel sums will differ.

3 Parameter Ranges

Table 4 lists the parameter combinations that were used in realizing the experiments. For each, all parallel formulations were carried out along with the serial addition. The total number of nodes was chosen and incremented in quantities that facilitate each given topology, although the application was designed such that any nodes in excess of the topology size invariant, would be simply left out of the communication. For the hypercube, the total number of nodes was always a power of two. For the ring, the number of nodes is initially a multiple of 4, but then becomes a multiple of 7, mainly to distribute the input among all 7 processors made available on the Extreme cluster. For the mesh, the number of total nodes was always a perfect square. Similarly, the input sizes, n, were incremented starting from the base size in powers of two, and chosen such that for each topology, $\forall p, k.\ pk | min(n)$, so that the inputs are divisible among all nodes present.

It must be noted that the (4 x 4) Mesh geometry was limited to the input size of 115200, smaller than the input sizes for other geometries due to an anomaly experienced in the timings, further discussed in the analysis.

Topology	Processors	Cores	Total nodes	Range of n
Hypercube	4	1	4	1024-65536
	4	2	8	4096-262144
	4	4	16	4096-262144
	4	8	32	4096-2097152
Mesh	4	1	4	3600-115200
	3	3	9	3600-460800
	4	4	16	3600-115200
	5	5	25	3600-921600
	6	6	36	3600-1843200
Ring	4	1	4	4200-134400
	4	2	8	4200-268800
	7	2	14	4200-268800
	7	4	28	4200-268800

Table 4: Parameter ranges

4 Results

For each topology and total number of nodes, the operation was performed with an increasing input size for both parallel formulations, as well as the serial computation at the root node. Each operation was carried out three times, with the mean, minimum, and maximum timings recorded in microseconds, in addition to the standard deviation. Below each table containing these timings follows a plot comparing the parallel/serial formulations with each other as the input size increases.

4.1 Hypercube

4.1.1 Hypercube 4 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
1024	serial	2.8610	2.8610	2.8610	0.0000
1024	scatter-reduce	10.6494	5.0068	20.0272	6.6767
1024	scatter-gather	5.7220	4.0531	8.1062	1.7302
2048	serial	6.0399	5.9605	6.1989	0.1124
2048	scatter-reduce	5.3247	5.0068	5.9605	0.4496
2048	scatter-gather	5.9605	5.9605	5.9605	0.0000
4096	serial	11.9209	11.9209	11.9209	0.0000
4096	scatter-reduce	22.6498	12.8746	41.0080	12.9903
4096	scatter-gather	11.9209	11.9209	11.9209	0.0000
8192	serial	22.6498	21.9345	23.1266	0.5150
8192	scatter-reduce	22.3319	17.8814	30.9944	6.1261
8192	scatter-gather	18.3582	17.8814	19.0735	0.5150
16384	serial	45.3790	45.0611	46.0148	0.4496
16384	scatter-reduce	51.3395	33.1402	69.8566	14.9911
16384	scatter-gather	35.8423	32.1865	42.2001	4.5125
32768	serial	97.0364	93.9369	102.9968	4.2158
32768	scatter-reduce	70.6514	56.9820	96.7979	18.4948
32768	scatter-gather	58.0152	56.9820	60.0815	1.4611
65536	serial	195.3443	187.8738	202.1790	5.8573
65536	scatter-reduce	127.9513	99.8974	180.9597	37.5039
65536	scatter-gather	112.2952	102.0432	122.7856	8.4697

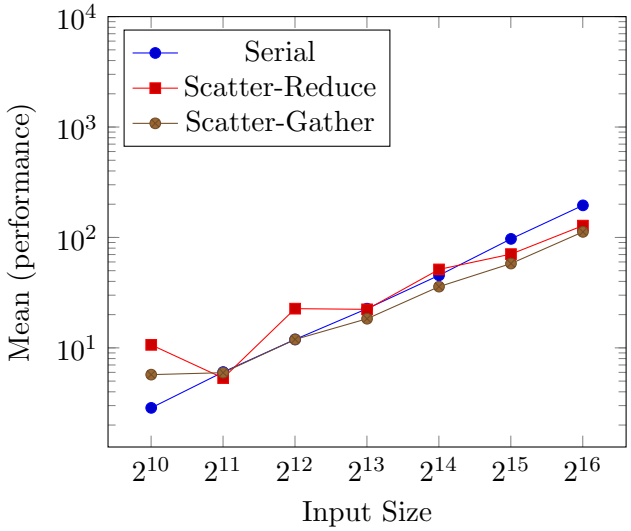

Figure 1: **Hypercube 4 node cluster.** Input size vs performance, in microseconds.

4.1.2 Hypercube 8 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
4096	serial	13.9872	10.9673	19.0735	3.6175
4096	scatter-reduce	19.7093	10.9673	36.0012	11.5304
4096	scatter-gather	14.3051	10.9673	20.9808	4.7204
8192	serial	22.8882	22.8882	22.8882	0.0000
8192	scatter-reduce	43.3127	25.9876	68.9030	18.4692
8192	scatter-gather	34.3323	24.0803	53.8826	13.8296
16384	serial	45.0611	45.0611	45.0611	0.0000
16384	scatter-reduce	46.8095	37.1933	63.1809	11.6351
16384	scatter-gather	36.6370	36.0012	36.9549	0.4496
32768	serial	94.3343	90.8375	101.0895	4.7776
32768	scatter-reduce	76.7708	62.2272	105.1426	20.0640
32768	scatter-gather	61.0352	60.0815	61.9888	0.7787
65536	serial	184.6155	181.9134	190.0196	3.8213
65536	scatter-reduce	144.6406	116.1098	191.9270	33.6738
65536	scatter-gather	113.0899	110.1494	117.0635	2.9157
131072	serial	365.3367	362.8731	370.0256	3.3170
131072	scatter-reduce	234.3655	184.0591	331.1634	68.4642
131072	scatter-gather	185.9665	180.0060	190.9733	4.5279
262144	serial	730.4351	725.9846	733.1371	3.1710
262144	scatter-reduce	597.6359	501.8711	782.0129	130.4069
262144	scatter-gather	503.6195	498.0564	512.8384	6.5651

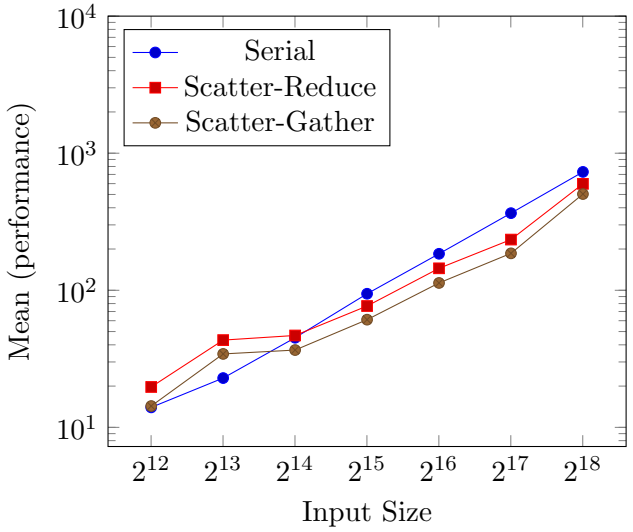

Figure 2: **Hypercube 8 node cluster.** Input size vs performance, in microseconds.

4.1.3 Hypercube 16 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
4096	serial	13.3514	13.1130	13.8283	0.3372
4096	scatter-reduce	9968.9960	9948.0152	9994.0300	19.0028
4096	scatter-gather	7684.6282	3064.8708	9996.8910	3266.6623
8192	serial	25.0340	25.0340	25.0340	0.0000
8192	scatter-reduce	18.0403	17.1661	18.8351	0.6837
8192	scatter-gather	29.0871	17.1661	35.0475	8.4294
16384	serial	50.3063	49.8295	51.0216	0.5150
16384	scatter-reduce	75.3403	46.9685	118.0172	30.7187
16384	scatter-gather	46.6506	46.0148	47.9221	0.8991
32768	serial	103.0763	100.1358	108.9573	4.1585
32768	scatter-reduce	79.3139	65.0883	105.8578	18.7855
32768	scatter-gather	77.2476	65.8035	83.9233	8.1296
65536	serial	204.0068	200.9869	209.0931	3.6175
65536	scatter-reduce	133.6734	106.0963	181.9134	34.2274
65536	scatter-gather	109.9904	105.1426	118.0172	5.7165
131072	serial	403.0069	401.0201	406.9805	2.8098
131072	scatter-reduce	256.3794	190.9733	344.0380	64.4390
131072	scatter-gather	192.3243	190.0196	195.9801	2.6142
262144	serial	808.0006	806.0932	809.9079	1.5573
262144	scatter-reduce	412.7026	311.1362	608.9211	138.7761
262144	scatter-gather	320.9909	304.9374	349.9985	20.5505

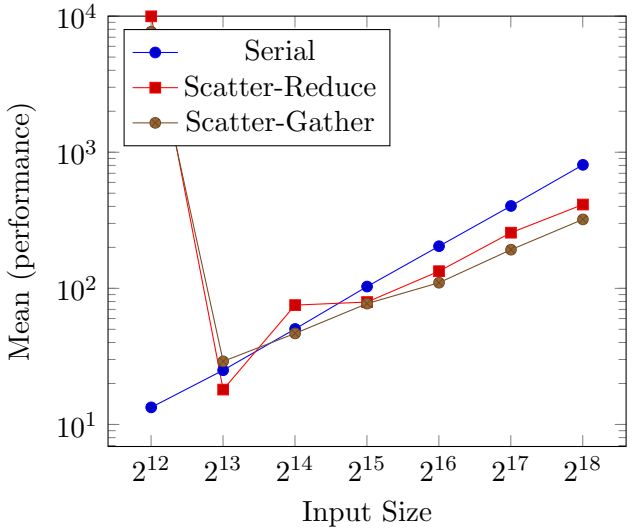

Figure 3: **Hypercube 16 node cluster.** Input size vs performance, in microseconds.

4.1.4 Hypercube 32 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
4096	serial	12.7157	11.9209	13.1130	0.5620
4096	scatter-reduce	93.7780	32.1865	127.0771	43.5997
4096	scatter-gather	38.6238	25.9876	55.0747	12.1773
8192	serial	25.0340	25.0340	25.0340	0.0000
8192	scatter-reduce	29.6434	26.9413	33.8554	3.0179
8192	scatter-gather	50.3063	30.0407	68.9030	15.9093
16384	serial	50.3858	50.0679	51.0216	0.4496
16384	scatter-reduce	36.6370	36.0012	36.9549	0.4496
16384	scatter-gather	72.7177	35.0475	147.1043	52.6007
32768	serial	100.4537	100.1358	101.0895	0.4496
32768	scatter-reduce	241.6770	161.8862	291.1091	56.9554
32768	scatter-gather	133.0376	123.9777	144.9585	8.8011
65536	serial	205.1195	200.0332	213.1462	5.7430
65536	scatter-reduce	240.3259	211.9541	262.9757	21.2188
65536	scatter-gather	210.3647	199.0795	220.0603	8.6388
131072	serial	405.3116	401.0201	412.9410	5.4088
131072	scatter-reduce	356.9921	283.9565	463.9626	77.3056
131072	scatter-gather	309.0700	284.1949	339.9849	23.1731
262144	serial	811.6563	810.8616	813.0074	0.9603
262144	scatter-reduce	639.6770	483.0360	918.8652	197.9059
262144	scatter-gather	520.8651	474.9298	563.8599	36.3651
524288	serial	1613.3785	1610.9943	1615.0475	1.7302
524288	scatter-reduce	988.9603	803.9474	1343.9655	251.1015
524288	scatter-gather	860.0553	827.0741	885.0098	24.3236
1048576	serial	3218.3329	3216.9819	3220.0813	1.2962
1048576	scatter-reduce	1796.3250	1439.0945	2480.9837	484.2797
1048576	scatter-gather	1451.9691	1427.8889	1477.0031	20.0621
2097152	serial	6428.2417	6426.8112	6430.8643	1.8570
2097152	scatter-reduce	4330.3967	4085.0639	4819.1547	345.6050
2097152	scatter-gather	4125.1977	4073.8583	4220.9625	67.7747

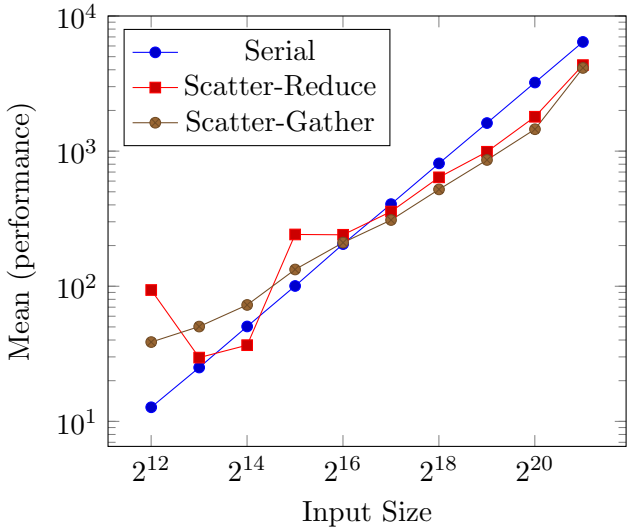

Figure 4: **Hypercube 32 node cluster.** Input size vs performance, in microseconds.

4.2 Mesh

4.2.1 Mesh 4 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
3600	serial	10.0136	10.0136	10.0136	0.0000
3600	scatter-reduce	16.3714	10.0136	29.0871	8.9913
3600	scatter-gather	8.7420	7.1526	10.9673	1.6209
7200	serial	19.9477	19.7887	20.0272	0.1124
7200	scatter-reduce	25.9876	18.1198	32.9018	6.0723
7200	scatter-gather	16.9277	16.9277	16.9277	0.0000
14400	serial	40.0543	40.0543	40.0543	0.0000
14400	scatter-reduce	37.0344	26.9413	56.0284	13.4396
14400	scatter-gather	27.5771	26.9413	28.8486	0.8991
28800	serial	80.1086	79.1550	81.0623	0.7787
28800	scatter-reduce	69.9361	51.9753	86.7844	14.2325
28800	scatter-gather	51.6574	51.0216	52.9289	0.8991
57600	serial	160.3762	159.9789	161.1710	0.5620
57600	scatter-reduce	114.5999	93.9369	154.9721	28.5501
57600	scatter-gather	92.3475	87.9765	100.1358	5.5209
115200	serial	332.7529	320.1962	339.9849	8.9130
115200	scatter-reduce	209.9673	165.9393	287.0560	54.6935
115200	scatter-gather	171.3435	164.9857	174.9992	4.5125

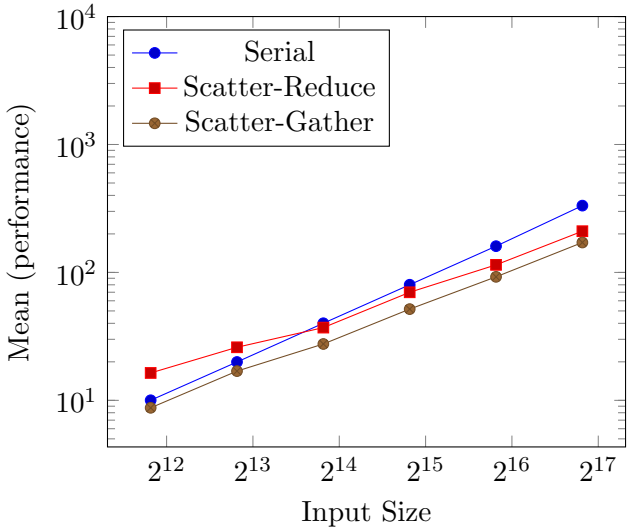

Figure 5: **Mesh 4 node cluster.** Input size vs performance, in microseconds.

4.2.2 Mesh 9 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
3600	serial	12.9541	10.0136	17.8814	3.5058
3600	scatter-reduce	30.2792	11.9209	52.9289	17.0142
3600	scatter-gather	13.9872	10.0136	20.9808	4.9605
7200	serial	20.6629	20.0272	20.9808	0.4496
7200	scatter-reduce	14.3846	13.1130	15.0204	0.8991
7200	scatter-gather	13.3514	12.8746	14.0667	0.5150
14400	serial	41.0080	41.0080	41.0080	0.0000
14400	scatter-reduce	46.6506	35.0475	66.9956	14.4335
14400	scatter-gather	44.3459	34.0939	63.8962	13.8296
28800	serial	91.3143	82.0160	96.0827	6.5756
28800	scatter-reduce	83.3670	55.0747	126.1234	30.7559
28800	scatter-gather	54.9952	54.8363	55.0747	0.1124
57600	serial	168.4030	164.9857	174.0456	4.0195
57600	scatter-reduce	125.9645	93.9369	184.0591	41.1511
57600	scatter-gather	94.3343	93.9369	95.1290	0.5620
115200	serial	343.0049	329.9713	350.9521	9.2898
115200	scatter-reduce	211.7157	155.9258	316.1430	73.8990
115200	scatter-gather	157.9920	155.9258	161.1710	2.2813
230400	serial	674.9630	658.9890	699.9969	17.9253
230400	scatter-reduce	374.3172	288.0096	520.9446	104.2226
230400	scatter-gather	284.9897	283.0029	287.0560	1.6556
460800	serial	1340.6277	1323.9384	1359.9396	14.8140
460800	scatter-reduce	1018.2858	882.8640	1286.9835	190.0000
460800	scatter-gather	912.2690	885.0098	934.8392	20.6112

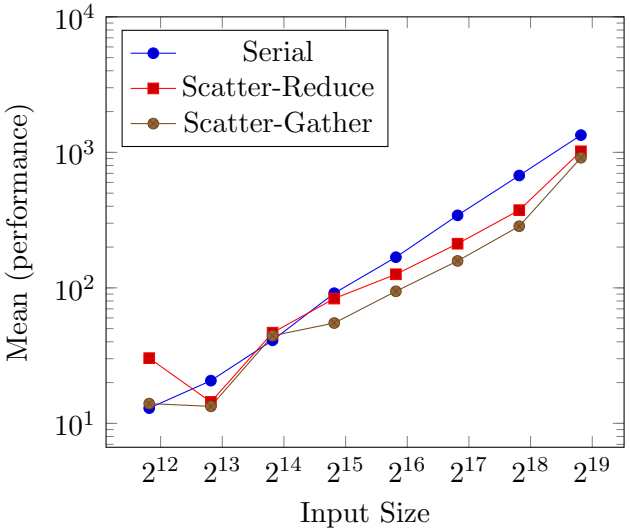

Figure 6: **Mesh 9 node cluster.** Input size vs performance, in microseconds.

4.2.3 Mesh 16 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
3600	serial	10.9673	10.9673	10.9673	0.0000
3600	scatter-reduce	25.3518	11.9209	51.0216	18.1578
3600	scatter-gather	14.3846	11.2057	19.7887	3.8411
7200	serial	22.0140	21.9345	22.1729	0.1124
7200	scatter-reduce	27.2592	16.9277	46.9685	13.9420
7200	scatter-gather	16.6098	15.9740	16.9277	0.4496
14400	serial	43.9485	43.8690	44.1074	0.1124
14400	scatter-reduce	21.3782	20.0272	22.1729	0.9603
14400	scatter-gather	21.9345	20.9808	22.8882	0.7787
28800	serial	92.9832	87.9765	101.8047	6.2567
28800	scatter-reduce	87.6586	61.9888	132.0839	31.5400
28800	scatter-gather	61.6709	60.0815	64.8499	2.2478
57600	serial	180.8008	177.1450	188.1123	5.1700
57600	scatter-reduce	137.0112	99.8974	198.1258	43.5449
57600	scatter-gather	98.3079	97.0364	98.9437	0.8991
115200	serial	356.3563	351.9058	363.1115	4.8563
115200	scatter-reduce	229.2792	174.0456	338.7928	77.4388
115200	scatter-gather	199.9537	173.8071	215.0536	18.5623

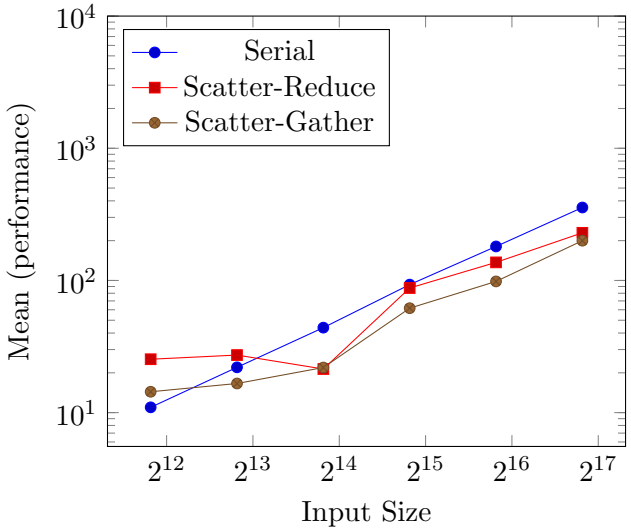

Figure 7: **Mesh 16 node cluster.** Input size vs performance, in microseconds.

4.2.4 Mesh 25 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
3600	serial	10.9673	10.9673	10.9673	0.0000
3600	scatter-reduce	68.0288	25.9876	150.9190	58.6142
3600	scatter-gather	30.1202	25.0340	39.1006	6.3687
7200	serial	22.0140	21.9345	22.1729	0.1124
7200	scatter-reduce	67.9493	29.8023	144.0048	53.7795
7200	scatter-gather	27.6566	25.0340	30.9944	2.4854
14400	serial	43.9485	43.8690	44.1074	0.1124
14400	scatter-reduce	58.6510	36.9549	102.0432	30.6829
14400	scatter-gather	38.9417	33.8554	47.9221	6.3687
28800	serial	88.3738	87.9765	88.9301	0.4052
28800	scatter-reduce	146.7069	107.0499	212.9078	47.1162
28800	scatter-gather	127.6334	97.0364	174.9992	33.9651
57600	serial	176.4297	175.9529	177.1450	0.5150
57600	scatter-reduce	197.9669	159.0252	272.9893	53.0617
57600	scatter-gather	185.0128	179.0524	190.0196	4.5279
115200	serial	356.2768	351.9058	362.8731	4.7458
115200	scatter-reduce	285.0691	247.9553	356.1974	50.3112
115200	scatter-gather	268.3004	247.0016	288.9633	17.1367
230400	serial	715.0173	713.8252	716.2094	0.9733
230400	scatter-reduce	550.6674	422.9546	735.0445	133.5618
230400	scatter-gather	455.6179	438.9286	470.8767	13.0819
460800	serial	1416.6832	1410.9612	1420.0211	4.0648
460800	scatter-reduce	938.4950	749.8264	1288.8908	248.0113
460800	scatter-gather	734.9650	716.9247	761.9858	19.4613
921600	serial	2828.9954	2827.8828	2830.9822	1.4083
921600	scatter-reduce	2312.0244	2053.9761	2810.9550	352.8668
921600	scatter-gather	2124.5480	2072.8111	2205.8487	58.1966

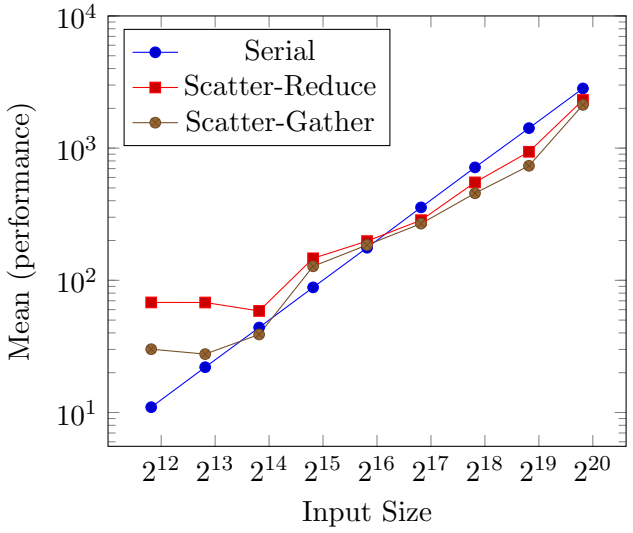

Figure 8: **Mesh 25 node cluster.** Input size vs performance, in microseconds.

4.2.5 Mesh 36 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
14400	serial	82.0955	44.1074	158.0715	53.7232
14400	scatter-reduce	95.6853	38.8622	204.0863	76.6810
14400	scatter-gather	67.3135	41.0080	100.8511	24.9603
28800	serial	125.9645	87.9765	201.9405	53.7232
28800	scatter-reduce	52.3726	48.1606	56.0284	3.2360
28800	scatter-gather	50.3858	46.0148	55.0747	3.7055
57600	serial	213.3052	175.9529	287.0560	52.1512
57600	scatter-reduce	363.0320	282.0492	432.0145	61.8083
57600	scatter-gather	229.9945	222.2061	243.9022	9.8578
115200	serial	424.7030	352.8595	461.1015	50.8025
115200	scatter-reduce	449.0217	329.0176	562.9063	95.5826
115200	scatter-gather	414.2920	329.9713	488.0428	64.9638
230400	serial	814.6763	813.0074	818.0141	2.3602
230400	scatter-reduce	695.0696	524.9977	939.1308	176.9790
230400	scatter-gather	634.3524	573.8735	699.9969	51.6191
460800	serial	1516.5806	1515.8653	1517.0574	0.5150
460800	scatter-reduce	1200.9939	983.9535	1555.9196	253.0423
460800	scatter-gather	960.6679	921.9646	1029.0146	48.4697
921600	serial	2927.0649	2926.1112	2928.0186	0.7787
921600	scatter-reduce	1923.3227	1636.9820	2427.1011	357.3341
921600	scatter-gather	1601.0602	1579.9999	1640.0814	27.6211
1843200	serial	5748.5898	5747.7951	5748.9872	0.5620
1843200	scatter-reduce	4575.2525	4305.8395	5089.9982	364.1129
1843200	scatter-gather	4350.7417	4331.1119	4388.0939	26.4235

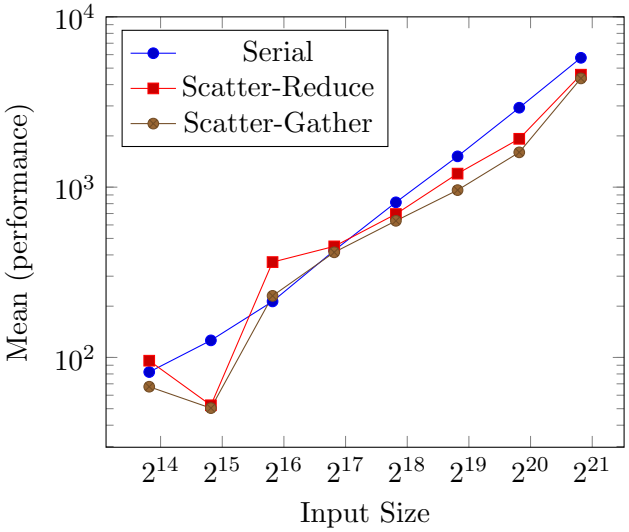

Figure 9: **Mesh 36 node cluster.** Input size vs performance, in microseconds.

4.3 Ring

4.3.1 Ring 4 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
4200	serial	12.0799	11.9209	12.1593	0.1124
4200	scatter-reduce	24.0008	11.9209	45.0611	14.9455
4200	scatter-gather	13.4309	12.1593	15.0204	1.1894
8400	serial	23.5240	22.8882	23.8419	0.4496
8400	scatter-reduce	23.2855	18.8351	30.9944	5.4727
8400	scatter-gather	18.6761	18.1198	19.0735	0.4052
16800	serial	46.6506	46.0148	46.9685	0.4496
16800	scatter-reduce	39.7364	32.1865	51.9753	8.7326
16800	scatter-gather	31.3918	30.9944	31.9481	0.4052
33600	serial	97.4337	94.1753	103.9505	4.6081
33600	scatter-reduce	81.3007	58.8894	113.9641	23.6239
33600	scatter-gather	57.9357	57.9357	57.9357	0.0000
67200	serial	189.3044	185.9665	195.0264	4.0648
67200	scatter-reduce	134.3091	103.9505	194.0727	42.2610
67200	scatter-gather	110.3083	102.9968	124.9313	10.3400
134400	serial	381.6287	374.0788	391.9601	7.5604
134400	scatter-reduce	353.0184	298.0232	432.9681	57.8479
134400	scatter-gather	313.3615	298.0232	330.9250	13.5239

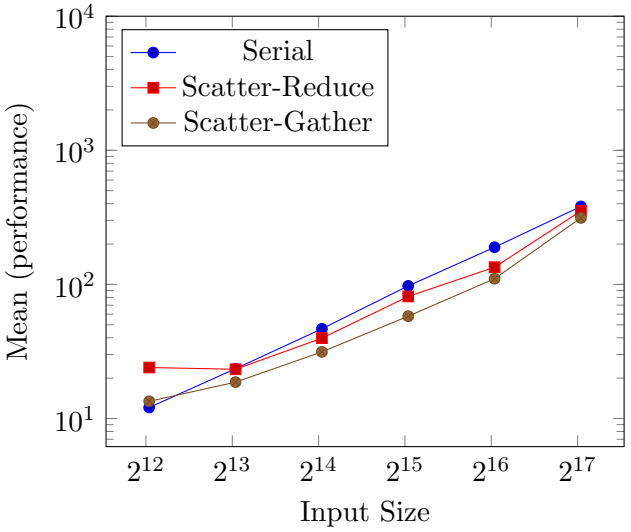

Figure 10: **Ring 4 node cluster.** Input size vs performance, in microseconds.

4.3.2 Ring 8 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
4200	serial	14.7025	11.9209	19.0735	3.1288
4200	scatter-reduce	22.3319	10.9673	42.9153	14.5810
4200	scatter-gather	13.6693	11.9209	16.9277	2.3061
8400	serial	27.9744	23.8419	36.0012	5.6766
8400	scatter-reduce	39.1006	27.1797	57.9357	13.4743
8400	scatter-gather	36.0012	25.0340	56.9820	14.8408
16800	serial	46.7300	46.0148	47.2069	0.5150
16800	scatter-reduce	53.4058	40.0543	76.0555	16.1011
16800	scatter-gather	37.9086	36.9549	38.8622	0.7787
33600	serial	96.3211	92.9832	102.9968	4.7204
33600	scatter-reduce	75.9761	60.7967	102.9968	19.1551
33600	scatter-gather	60.9557	59.8431	61.9888	0.8778
67200	serial	189.6222	185.9665	195.9801	4.5125
67200	scatter-reduce	143.6869	113.0104	199.0795	39.2440
67200	scatter-gather	115.3151	113.0104	118.9709	2.6142
134400	serial	382.3439	371.9330	401.0201	13.2350
134400	scatter-reduce	258.1278	198.1258	375.0324	82.6737
134400	scatter-gather	188.3507	187.1586	189.0659	0.8485
268800	serial	761.0321	754.1180	768.8999	6.0723
268800	scatter-reduce	654.6179	546.9322	864.0289	148.0959
268800	scatter-gather	549.7138	545.9785	552.1774	2.6857

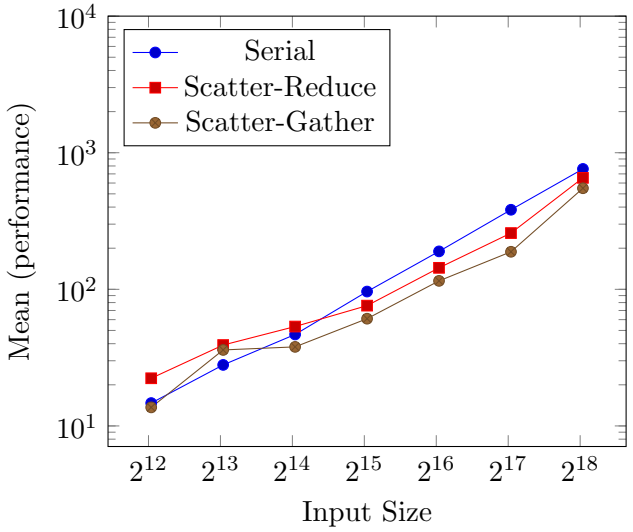

Figure 11: **Ring 8 node cluster.** Input size vs performance, in microseconds.

4.3.3 Ring 14 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
4200	serial	12.9541	12.8746	13.1130	0.1124
4200	scatter-reduce	28.7692	11.2057	61.0352	22.8454
4200	scatter-gather	24.0803	11.9209	37.1933	10.3394
8400	serial	25.3518	25.0340	25.9876	0.4496
8400	scatter-reduce	18.3582	15.9740	20.9808	2.0510
8400	scatter-gather	15.2588	14.7820	15.9740	0.5150
16800	serial	51.3395	51.0216	51.9753	0.4496
16800	scatter-reduce	78.3602	53.1673	108.0036	22.6069
16800	scatter-gather	42.9948	41.9617	44.1074	0.8778
33600	serial	106.3347	102.9968	113.0104	4.7204
33600	scatter-reduce	79.3139	66.9956	102.9968	16.7509
33600	scatter-gather	66.4393	65.0883	68.1877	1.2962
67200	serial	208.3778	205.0400	215.0536	4.7204
67200	scatter-reduce	150.3627	112.0567	205.9937	40.2588
67200	scatter-gather	111.6594	108.0036	118.9709	5.1700
134400	serial	413.7357	411.0336	419.1399	3.8213
134400	scatter-reduce	252.3263	202.1790	342.8459	64.1313
134400	scatter-gather	208.3778	200.0332	220.0603	8.5099
268800	serial	830.3324	828.9814	832.0808	1.2962
268800	scatter-reduce	469.2872	328.0640	718.8320	176.9653
268800	scatter-gather	328.2229	322.8188	337.8391	6.8171

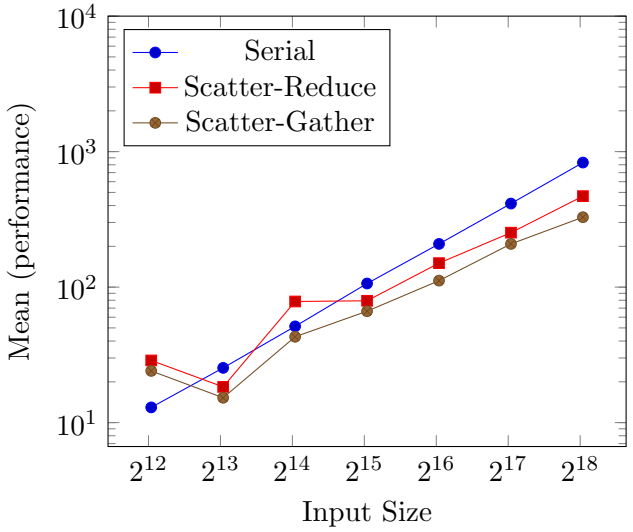

Figure 12: **Ring 14 node cluster.** Input size vs performance, in microseconds.

4.3.4 Ring 28 node cluster

N	Formulation	Mean	Minimum	Max	StdDev
4200	serial	12.6362	11.9209	13.1130	0.5150
4200	scatter-reduce	142.6538	123.9777	174.0456	22.3303
4200	scatter-gather	36.9549	21.9345	66.0419	20.5713
8400	serial	26.0671	25.9876	26.2260	0.1124
8400	scatter-reduce	25.9876	25.9876	25.9876	0.0000
8400	scatter-gather	26.0671	24.0803	28.1334	1.6556
16800	serial	51.5779	50.7832	51.9753	0.5620
16800	scatter-reduce	30.3586	30.0407	30.9944	0.4496
16800	scatter-gather	69.3798	33.1402	118.9709	36.2898
33600	serial	103.3147	102.9968	103.9505	0.4496
33600	scatter-reduce	211.6362	140.9054	289.9170	61.0675
33600	scatter-gather	119.2888	111.8183	133.9912	10.3967
67200	serial	209.6494	205.0400	217.9146	5.8573
67200	scatter-reduce	212.0336	177.8603	280.1418	48.1599
67200	scatter-gather	190.6554	176.9066	203.8479	11.0058
134400	serial	415.0073	411.0336	422.9546	5.6196
134400	scatter-reduce	340.2233	268.9362	471.8304	93.1679
134400	scatter-gather	283.7181	272.0356	300.1690	11.9701
268800	serial	831.3656	830.8887	832.0808	0.5150
268800	scatter-reduce	580.6287	452.9953	819.9215	169.3312
268800	scatter-gather	513.7126	469.9230	580.0724	47.7164

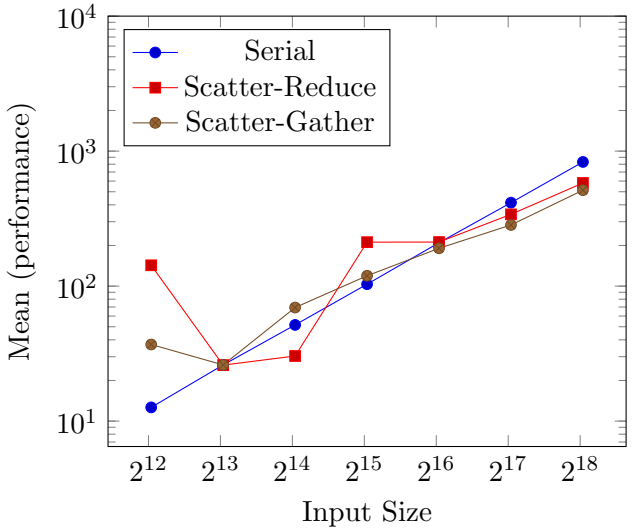

Figure 13: **Ring 28 node cluster.** Input size vs performance, in microseconds.

4.4 Performance with growing cluster and input size

Table 5 lists the average serial runtime experienced with increasing input sizes used across all topologies. Tables 6 - 11 compare the performance for increasing node and input sizes for the parallel formulations, grouped by topology.

N	SerialRuntime
4096	12.7157
8192	25.0340
16384	50.3858
32768	100.4537
65536	205.1195
131072	405.3116
262144	811.6563
14400	43.9485
28800	92.9832
57600	180.8008
115200	356.3563
4200	12.6362
8400	26.0671
16800	51.5779
33600	103.3147
67200	209.6494
134400	415.0073
268800	831.3656

Table 5: **Serial Runtime.** n vs performance, in μsecs.

Nodes	N=4096	N=8192	N=16384	N=32768	N=65536	N=131072	N=262144
4	22.6498	22.3319	51.3395	70.6514	127.9513	0	0
8	19.7093	43.3127	46.8095	76.7708	144.6406	234.3655	597.6359
16	9968.9960	18.0403	75.3403	79.3139	133.6734	256.3794	412.7026
32	93.7780	29.6434	36.6370	241.6770	240.3259	356.9921	639.6770

Table 6: **Hypercube Scatter/Reduce.** pk and n vs performance, in μsecs.

Nodes	N=4096	N=8192	N=16384	N=32768	N=65536	N=131072	N=262144
4	11.9209	18.3582	35.8423	58.0152	112.2952	0	0
8	14.3051	34.3323	36.6370	61.0352	113.0899	185.9665	503.6195
16	7684.6282	29.0871	46.6506	77.2476	109.9904	192.3243	320.9909
32	38.6238	50.3063	72.7177	133.0376	210.3647	309.0700	520.8651

Table 7: **Hypercube Scatter/Gather.** pk and n vs performance, in μsecs.

Nodes	N=14400	N=28800	N=57600	N=115200
4	37.0344	69.9361	114.5999	209.9673
9	46.6506	83.3670	125.9645	211.7157
16	21.3782	87.6586	137.0112	229.2792
25	58.6510	146.7069	197.9669	285.0691
36	95.6853	52.3726	363.0320	449.0217

Table 8: **Mesh Scatter/Reduce.** pk and n vs performance, in μsecs.

Nodes	N=14400	N=28800	N=57600	N=115200
4	8.7420	16.9277	27.5771	51.6574
9	44.3459	54.9952	94.3343	157.9920
16	21.9345	61.6709	98.3079	199.9537
25	38.9417	127.6334	185.0128	268.3004
36	67.3135	50.3858	229.9945	414.2920

Table 9: **Mesh Scatter/Gather.** pk and n vs performance, in μsecs.

Nodes	N=4200	N=8400	N=16800	N=33600	N=67200	N=134400	N=268800
4	24.0008	23.2855	39.7364	81.3007	134.3091	353.0184	0
8	22.3319	39.1006	53.4058	75.9761	143.6869	258.1278	654.6179
14	28.7692	18.3582	78.3602	79.3139	150.3627	252.3263	469.2872
28	142.6538	25.9876	30.3586	211.6362	212.0336	340.2233	580.6287

Table 10: **Ring Scatter/Reduce.** pk and n vs performance, in μsecs.

Nodes	N=4200	N=8400	N=16800	N=33600	N=67200	N=134400	N=268800
4	13.4309	18.6761	31.3918	57.9357	110.3083	313.3615	0
8	13.6693	36.0012	37.9086	60.9557	115.3151	188.3507	549.7138
14	24.0803	15.2588	42.9948	66.4393	111.6594	208.3778	328.2229
28	36.9549	26.0671	69.3798	119.2888	190.6554	283.7181	513.7126

Table 11: **Ring Scatter/Gather.** pk and n vs performance, in μsecs.

5 Analysis

On average, as seen in figures 1 - 13, for every cluster size and topology combination, the time to perform the addition grew linearly with respect to a growing input size. The observation was nearly precise with the serial addition performed on one node, but less so for the parallel formulations, especially under the smaller input sizes due to the dominating communication cost.

Figure 1 shows that in a Hypercube 4-node geometry, the serial addition outperformed the parallel

additions until the input size (n) of at least 2048 was established. In the Hypercube 8-node geometry, figure 2, the parallel advantage is only noticeable after $n = 16348$. Figure 3 shows that for $n = 4$, the parallel formulations experienced unusual delays, but stabilized in the subsequent input size, and did not outperform the serial addition until $n = 32768$. In the 32-node geometry, the threshold was not reached until $n = 131072$.

The Mesh topology demonstrated a similar observation as Hypercube, although with increased variation. In the 4-node cluster, figure 5, the Scatter-Gather formulation summed the numbers quicker than the serial version for all input sizes, with the advantage only increasing for the larger n values. For Scatter-Reduce, the same only applies after $n = 14400$. For the 9-node cluster, the threshold point for both parallel formulations only stabilized after $n = 28800$. The 16-node cluster actually overperformed with the input size of $n = 14400$, but continued to be favorable for greater input sizes. However, this geometry displayed anomalies when the maximum input size was increased beyond $n = 115200$.* The 25-node geometry, figure 8, did not surpass the threshold until $n = 115200$, and the 36-node geometry not until $n = 230400$.

For the Ring topology, figure 10 demonstrates the 4-node geometry threshold of $n = 8400$. In the 8-node geometry, this occurs at $n = 33600$ (figure 10). The 14-node geometry, after some fluctuations, performed with increased improvement over the serial variant after $n = 33600$. Lastly, for the 28-node cluster, the improvement was not noted until $n = 134400$.

Almost all topologies and node cluster sizes yielded a similar pattern in that the parallel runtime does not outperform the serial runtime until a certain threshold in the minimal input size, as typical for parallel programs. The Scatter-Gather formulation almost always outperformed Scatter-Reduce, with the exception of a few cases for lower input sizes in the Hypercube 16 and 32-node geometries, as well as the Ring 28 node geometry. Scatter-Gather also yielded far more consistent results, while Scatter-Reduce frequently experienced high deviation between readings, as observed in the standard deviation column of any setup.

Despite any advantages of parallel runtime over serial, even the more effective Scatter-Gather rarely performed more than a little over than twice the speed of the serial addition, and never 3X the serial speed, irrespective of the number of nodes involved. See figures 3, 4, 6, 7, 11, and 12 for the more successful cases. Because the parallel summation was performed in an optimal sense with respect to serial, communication time is the dominating overhead. To see why the parallel summation is optimal, it is sufficient to compare the asymptotic computation time of the parallel and serial variants:

$$
\begin{aligned}
T_s &= \Theta(n) \\
T_p &= \Theta(\frac{n}{p} + \log p) \\
\implies E &= \frac{T_s}{pT_p} = \frac{\Theta(n)}{p(\Theta(\frac{n}{p} + \log p))} \\
&= \frac{\Theta(n)}{\Theta(n)} \sim 1, \text{provided } n = \Omega(p \log p)
\end{aligned}
$$

The parallel runtime, T_p, follows because upon performing the MPI Scatter of n/p elements, each

node performs a local sum serially in time $\Theta(\frac{n}{p})$, and the summation of the resulting p local sums can be performed in parallel in $\log p$ steps. Additionally, the overall input size was always greater than $p \log p$, so the efficiency requirement was met. Note that p, in this context, refers to all nodes used, and not only the processors used in the Extreme cluster.

Looking at table 5, and comparing it to the parallel performance for increasing input sizes in tables 6 - 11, confirms the dominating communication time. For example, taking the Hypercube 16 and 32 node clusters, and the Scatter-Gather formulation, the performance for the input sizes of $n = 131072$, and $n = 262144$ is 309.0700 and 520.8651 μsecs, respectively, while the serial performance was 405.3116 and 811.6563 μsecs. The respective parallel speedups are 1.31 and 1.56, despite the number of nodes involved in the computation. Similarly, for the Mesh with the Scatter-Gather formulation, with 16 nodes and $n = 14400$, the parallel speedup $= S = \frac{43.9485}{21.9345} = 2.00$. Again, the communication time is a dominating factor.

Based on tables 6 - 11, looking at Scatter-Gather statistics alone, which performed better, the Hypercube near-ideal performance was experienced with 4 nodes, after which, the parallel performance decreased. The same was true for the Mesh, with the exception of $n = 28800$, where the 36-node cluster proved slightly advantageous. The Ring was the only topology for which an optimal cluster size varied based on input size, making it a more facilitating choice for varying the number of nodes involved, although it did not, in many cases, perform better than the other two topologies (a comparison that was approximated since the data sizes across the three topologies were multiples of somewhat different quantities.)

* Some anomalies were experienced with 16 node topologies. For example, the (4 X 4) Mesh combination resulted in unusually high timings for all inputs sizes and parallel formulations, even after multiple attempts, until the maximum input size was lowered to 115200, despite that smaller and larger node geometries did not experience this limitation. Granted, the application was designed such that the root node allocates memory for the maximum input size it will experiment with, which could, in theory, adversely impact the communication, but this does not explain why larger geometries performed within expectations. Additionally, the parallel formulations in the hypercube demonstrated unusually high timings for the smallest input size of 4096, but then stabilized for larger input sizes.

6 Lessons

- Without having knowledge of the underlying physical network topology, it is difficult to predict the impact of a logical topology and distribution scheme on the performance of a parallel computation.
- The communication and inter-process communication overheads for certain input sizes can completely dominate the parallel computation time, making any parallel optimization not a viable focus (Amdahl's law).
- The unpredictable nature of network and cluster resources make the task of semi-reliably measuring performance difficult, and often necessitate many trials and variations.