Boyao Zhu & Jiaxin Yang

CMSE 822 Parallel Computing

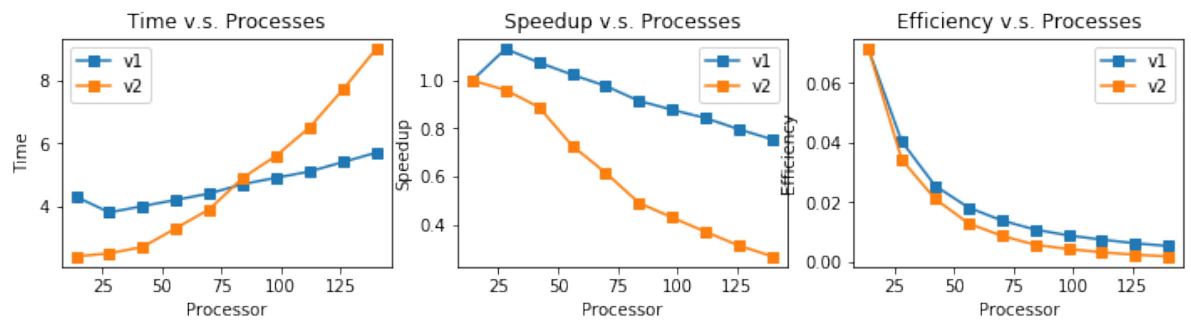
Homework 6

Part 1

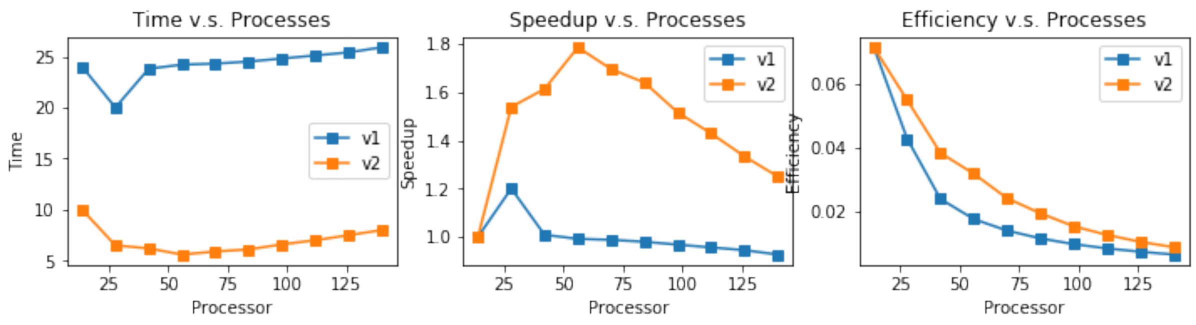
(a)

We test our programs on HPCC, starting from 14 cores (single socket) and using up to 140 cores (5 nodes). The program sizes N are 100 millions, 500 millions and 2 billions ($1 \times 10^8, 2 \times 10^8, 2 \times 10^9$), respectively. The total execution times, efficiencies and speedups are shown below.

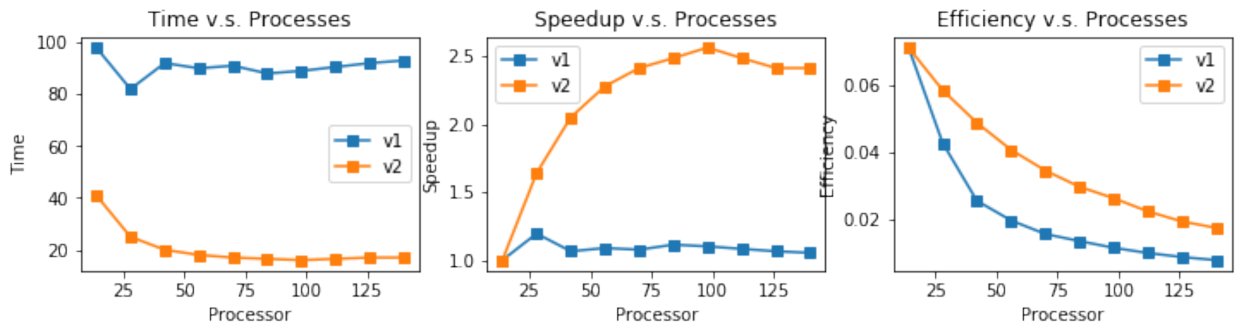
$$N = 10^8$$



$$N = 5 \times 10^8$$



$$N = 2 \times 10^9$$



Both two versions don’t scale well and perform worse with increasing process (p). We believe it is due to the competition between increasing communications and decreasing local calculations. Moreover, we observe that v2 outperforms v1 as N, the number of random number, grows. This can be understood that in v1, root process bears all preparation, and this increases as N increases. But, in v2, loads are evenly distributed among all processes.

(b)

$$N = 2 \times 10^9$$

	Generate time		Bin time		Distribute time		Local sort time		Gather time	
#processes	V1	V2	V1	V2	V1	V2	V1	V2	V1	V2
14	42.07	2.95	13.11	0.95	6.53	0.95	30.20	27.68	3.51	7.22
28	42.20	1.50	13.68	0.67	7.05	0.94	14.68	13.54	3.52	7.43
42	41.54	0.98	29.63	0.81	7.44	1.24	9.25	8.88	3.44	6.83
56	42.44	0.79	30.29	0.60	7.55	1.78	6.55	6.62	3.30	6.79
70	42.85	0.60	31.27	0.54	7.59	2.44	5.59	5.33	3.21	6.48
84	41.83	0.47	31.22	0.44	8.21	3.10	4.38	4.21	3.18	6.46
98	41.40	0.44	31.66	0.32	8.63	3.82	3.84	3.22	3.16	6.52
112	42.12	0.39	31.72	0.30	9.11	4.28	3.23	2.99	3.14	6.44
126	41.88	0.33	30.22	0.27	9.15	4.99	2.87	2.48	3.16	6.38
140	42.01	0.29	31.31	0.24	10.01	5.58	2.67	2.16	3.15	6.40

In this part, we measure the execution time of different phases, which consists of array generation, bin determination, array distribution, local sort and result gather. To make results numerically readable, We set $N = 2 \times 10^9$, which is characteristic enough for our analysis.
The most significant differences between v1 and v2 locate in the first three phases. Indeed, v1 spends much more time than v2 on these parts. And these phases contribute significantly to total execution time, especially with large process number. In v1 root process bears a heavy load, which seriously hurts the performance. However, v2 processes share almost the same load, which increases its efficiency. In addition, both v1 and v2 show similar performance in the phase of local sort, because each process works independently. V1 outperforms v2 in gathering

because less communication occurs in v1’s gathering, and v2 has a better load balance and faster than v1. Other results are attached below.

$$N = 1 \times 10^8$$

	Generate time		Bin time		Distribute time		Local sort time		Gather time	
#processes	V1	V2	V1	V2	V1	V2	V1	V2	V1	V2
14	2.67	0.16	0.79	0.05	0.58	0.26	1.69	1.39	0.17	0.32
28	2.98	0.08	0.87	0.02	0.80	0.46	0.83	0.67	0.17	0.38
42	2.72	0.06	1.11	0.03	1.00	0.58	0.48	0.48	0.25	0.41
56	3.17	0.05	1.35	0.02	1.24	0.77	0.42	0.37	0.35	0.66
70	3.19	0.05	1.29	0.02	1.51	0.97	0.35	0.26	0.20	0.51
84	3.34	0.04	1.27	0.02	1.67	1.18	0.29	0.26	0.21	0.56
98	3.01	0.03	1.30	0.01	1.93	1.46	0.26	0.23	0.30	0.64
112	3.55	0.03	1.36	0.01	2.16	1.18	0.23	0.20	0.25	0.68
126	3.59	0.02	1.54	0.01	2.54	2.29	0.20	0.17	0.27	0.75
140	3.79	0.02	1.43	0.01	2.63	2.35	0.19	0.17	0.24	0.78

$$N = 5 \times 10^8$$

	Generate time		Bin time		Distribute time		Local sort time		Gather time	
#processes	V1	V2	V1	V2	V1	V2	V1	V2	V1	V2
14	13.19	0.78	3.85	0.24	1.90	0.40	9.01	7.17	0.75	1.51
28	14.25	0.42	4.10	0.14	2.31	0.51	4.81	4.13	1.12	1.56
42	14.55	0.31	6.37	0.13	2.55	0.66	3.32	3.03	0.93	1.45
56	14.70	0.21	6.16	0.10	2.81	0.82	2.41	1.88	0.95	1.51
70	20.04	0.17	5.85	0.08	2.80	0.99	1.60	1.50	0.84	1.52
84	13.95	0.15	6.18	0.07	3.09	1.25	1.37	1.30	0.91	1.63
98	14.21	0.13	5.84	0.07	3.38	1.45	1.19	1.66	0.87	1.66
112	14.99	0.12	6.34	0.06	3.67	1.84	1.06	1.08	0.89	1.73
126	15.37	0.11	6.47	0.05	4.32	2.38	1.43	1.00	0.88	5.37
140	16.45	0.10	6.39	0.05	4.32	2.42	0.94	0.94	0.77	1.92

P.S.

In this Homework, we had 2 versions of bucket_sort_v2.c, which was coded separately. One compared bucket_sort_v1.c with the first version of bucket_sort_v2.c, which contribute to part I. While the other used second version of bucket_sort_v2.c to compare with bucket_sort_v3, which contribute to part II. So the data of bucket_sort_v2 may looks different in 2 parts.