ID1217 CONCURRENT PROGRAMMING

Performance Report 2

Jinyu Yu

February 9, 2020

(Using atlantis.sys.ict.kth.se)

Problem 1:

Matrix size = 10000 program h1a

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	0.393503	0.392314	0.393199	0.393285	0.393016	0.393199	1
2	0.204026	0.203802	0.203889	0.204647	0.201108	0.203889	1.93
4	0.102962	0.102721	0.103179	0.102902	0.10291	0.10291	3.82
8	0.0540044	0.0528705	0.0533634	0.0528417	0.0527766	0.0528705	7.44

Matrix size = 5000 program h1a

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	0.0990519	0.098935	0.0987687	0.0989724	0.0989581	0.0989581	1
2	0.0516222	0.0519441	0.0519379	0.0519755	0.0517313	0.0519379	1.91
4	0.0264538	0.0264843	0.0263657	0.0264854	0.0262986	0.026425	3.74
8	0.0141887	0.0140463	0.0138734	0.0142533	0.0140824	0.0140824	7.03

Matrix size = 1000 program h1a

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	0.0041569	0.004147	0.0041169	0.0041532	0.0041519	0.0041519	1
2	0.0023865	0.0023764	0.0023741	0.0023862	0.0023762	0.0023764	1.75
4	0.0013636	0.0013439	0.0013532	0.0013688	0.0013472	0.0013532	3.07
8	0.0009429	0.0009369	0.0009613	0.0009335	0.0009504	0.0009429	4.40

Matrix size = 10000 program h1b

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	0.329712	0.32491	0.32481	0.325119	0.325091	0.325091	1
2	0.171445	0.171735	0.171177	0.171382	0.171638	0.171445	1.90
4	0.0867074	0.0861547	0.0870198	0.0869919	0.086719	0.086719	3.75
8	0.0496062	0.0496846	0.0495366	0.0495874	0.0494685	0.0495874	6.56

Matrix size = 5000 program h1b

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	0.0819658	0.0819569	0.0820435	0.0819456	0.0818399	0.0819569	-
2	0.0442178	0.0438477	0.0440414	0.0442773	0.0438437	0.0440414	1.86
4	0.0227883	0.0226204	0.0225991	0.0224705	0.0225504	0.0225991	3.63
8	0.012888	0.0129376	0.0130806	0.0130479	0.013148	0.0130479	6.28

Matrix size = 1000 program h1b

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	0.0035744	0.0035382	0.0034617	0.0035772	0.0035525	0.0035525	-
2	0.0022027	0.0021369	0.0021853	0.0021457	0.0021766	0.0021766	1.63
4	0.0012789	0.0012891	0.0012855	0.0013013	0.001263	0.0012873	2.76
8	0.000946	0.0009347	0.0009335	0.0009187	0.0009304	0.0009335	3.81

Speed up table

size	1a,2proc	1a,4proc	1a,8proc	1b,2proc	1b,4proc	1b,8proc
10000	1.93	3.82	7.44	1.90	3.75	6.56
5000	1.91	3.74	7.03	1.86	3.63	6.28
1000	1.75	3.07	4.40	1.63	2.76	3.81

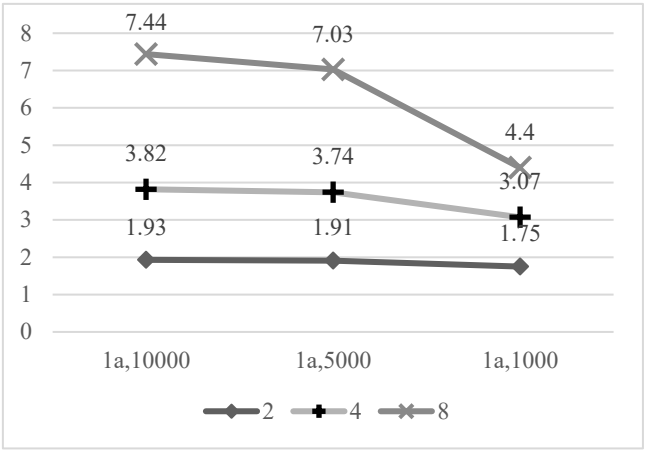

Figure 1 Speed up of program 1a

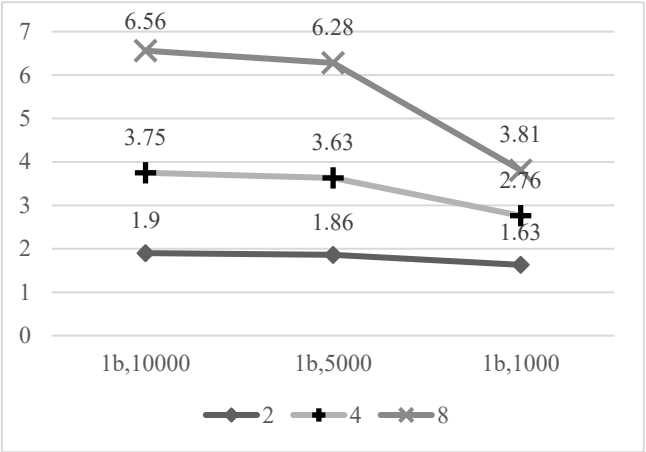

Figure 2 Speed up of program 1b

From the table, we can find that the speed up of program 1b is less than 1a. This is because 1b doesn't use array to save intermediate values and use an OpenMP critical region so that only one thread can run at a same time. This increases the sequential fraction and lowers the parallel fraction, so the speedup is relatively lower.

On the other hand, we can find that the larger the size, the more speed up rate. This is because small matrix only needs little time to complete actual calculation, but the fork and join and task allocation time stays unchanged, since these are sequential, the speedup will be lower.

Problem 3:

Word list length 25143:

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	15.2751	15.2749	15.4052	15.3254	15.2909	15.2909	1
2	7.65987	7.67297	7.66538	7.66391	7.66143	7.66391	1.995
4	3.83247	3.84055	3.84046	3.82781	3.83523	3.83523	3.987
8	1.91967	1.92036	1.92305	1.92244	1.91966	1.92036	7.963

Word list length 5000:

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	3.05673	3.06657	3.0504	3.05703	3.03838	3.05673	1
2	1.52957	1.52961	1.52755	1.52803	1.52897	1.52897	1.999
4	0.766139	0.766829	0.766335	0.77036	0.767279	0.766829	3.986
8	0.38374	0.384635	0.383462	0.383122	0.384321	0.38374	7.966

Word list length 1000:

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	0.604246	0.604774	0.603485	0.607505	0.608486	0.604774	1
2	0.304399	0.303207	0.304082	0.303778	0.302999	0.303778	1.991
4	0.15252	0.152946	0.152688	0.154333	0.152928	0.152928	3.955
8	0.0777836	0.0779505	0.0772062	0.0778691	0.0775101	0.0777836	7.775

Speed up table:

size	2	4	8
25143	1.995	3.987	7.963
5000	1.999	3.986	7.966
1000	1.991	3.955	7.775

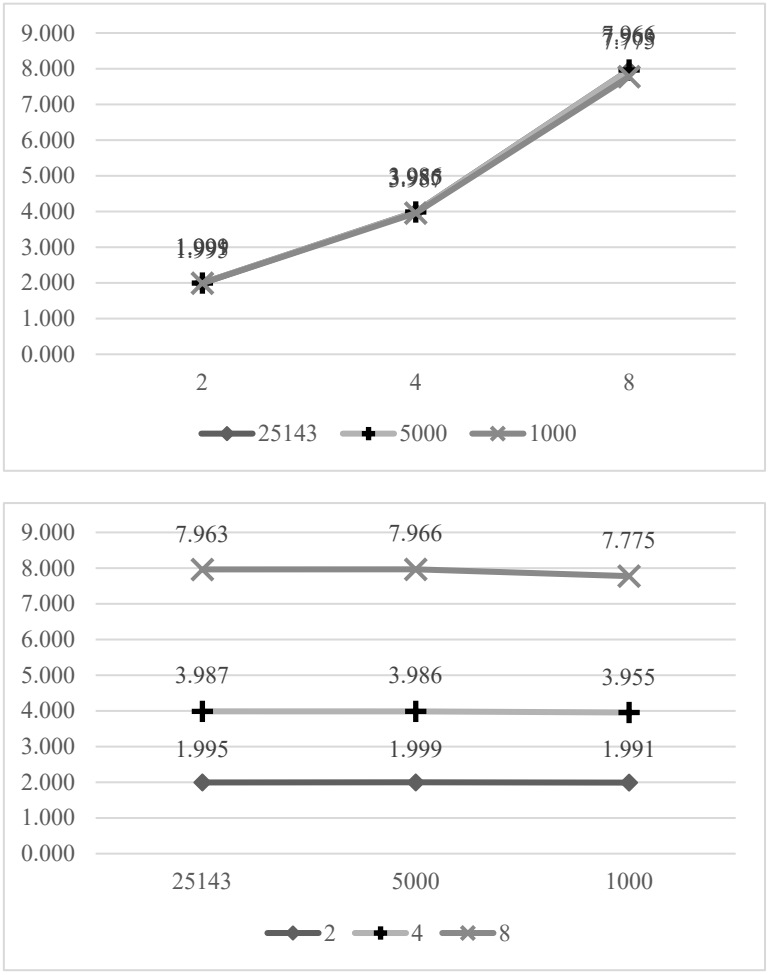

Figure 3 Speed up of program 3

The speed up of program 3 is high, because the sequential code only takes a small part in the code. At most there's one OpenMP critical statement in the inner loop which will run 25143 times. The statement is used to increase the count number, also use little time. If the word is not a palindromic word, there will be no sequential code.

Problem 4:

Speed up table:

nproc	Run1	Run2	Run3	Run4	Run5	Median	speedup
1	0.0005735	0.0005701	0.0005661	0.0005815	0.000571	0.000571	1
2	0.0003748	0.0003718	0.000374	0.0003682	0.0003827	0.000374	1.527
4	0.000283	0.0002811	0.0002761	0.000291	0.0002757	0.0002811	2.031

The speed up of program 4 is low, and it has a same reason with problem 1. The execute speed is so high that the program only need less then 1ms to complete. And the fork and join phase still takes time, which add to the sequential part of the code.