Big Data Homework 2
Student ID: 2019280513
Student name: Sami Emre Erdogan 夏承思

Running with and without Paralel Programming

Threads	Serial Reference Page Rank	Threads Page Rank	openmp used(Y/N)
16	11.011506 s	1.449266 s	YES
1	10.711349 s	11.553451 s	YES
1	10.904681 s	11.034485 s	NO

- We can see by optimizing the code by using paralel programming (see 16 threads), the speed was around 7.5 times faster.
- We can see that trying to run the code with 1 thread in paralel programming can even sometimes make it slower compared to a non-pararlel program.

Comparing The Amount of Threads with Paralel Programming

Threads	Serial Reference Page Rank	Threads Page Rank
32	11.172128 s	2.236902 s
24	11.125584 s	1.586065 s
16	11.011506 s	1.449266 s
8	11.040543 s	2.092157 s
4	11.493757 s	3.910837 s
2	10.836027 s	7.069899 s

- We can see by increasing the number of threads, the speed of the calculation increases and the number of seconds needed to compute the results decreases until 16 threads.
- As I increased the threads more than 16. The performance didn't improve. It either stayed around the same or perfomed slightly slower.
- Using 16 threads gave me the optimal solution for my paralel programming task.

In my paralel programming I have used :
#pragma omp parallel for schedule(guided,16)
With for loops as it has given me the optimal speed. In my next table I will play with the chunk sizes using schedule type and will use just **16 threads,** as I showed above **16 threads** has given me the optimal speed.

X = chunk size

Comparing Using Different Chunk Sizes with Schedule Guided

for schedule(guided,X)	**Serial Reference Page Rank**	**Threads Page Rank**
16	11.011506 s	1.449266 s
8	11.388863 s	1.631617 s
4	13.808653 s	1.661494 s
2	11.094040 s	1.611905 s

- You can see now as why I have chosen **#pragma omp parallel for schedule(guided,16)**
- I have tested the same values a couple of time in a row. 16 showed speeds between:1.40-1.58. As to others it was usually between 1.50-1.68. There were times that 16 performed worse than the others but overall it performes better.

In my next and final table I would like to show as to why I have chosen **#pragma omp parallel for schedule(guided)** among different schedule types such as dynamic and static. To keep it simple I will just run it with **16 threads** and use **16 as my chunk size**.

X = different schedule types.

Comparing Using Different Schedule Types with The Same Chunk Size

for schedule(x,16)	**Serial Reference Page Rank**	**Threads Page Rank**
guided	11.011506 s	1.449266 s
dynamic	10.904425 s	1.645771 s
static	11.222729 s	2.111160 s

- As you can see from the table above. The guided schedule type has performed better than the dynamic and the static type.
- I have also ran it a couple of times to see the accuracy average among different types. I have came to a conclusion that although rarely guided performes worse than other types. It has given me overall better performance when we compare it with others.

Finally I would also like to mention that I have also used: **#pragma omp atomic** to only one if statement operation to optimize the speed by ensuring its serialisation.