Line Segment Visibility from Origin

-Uthish Balaji

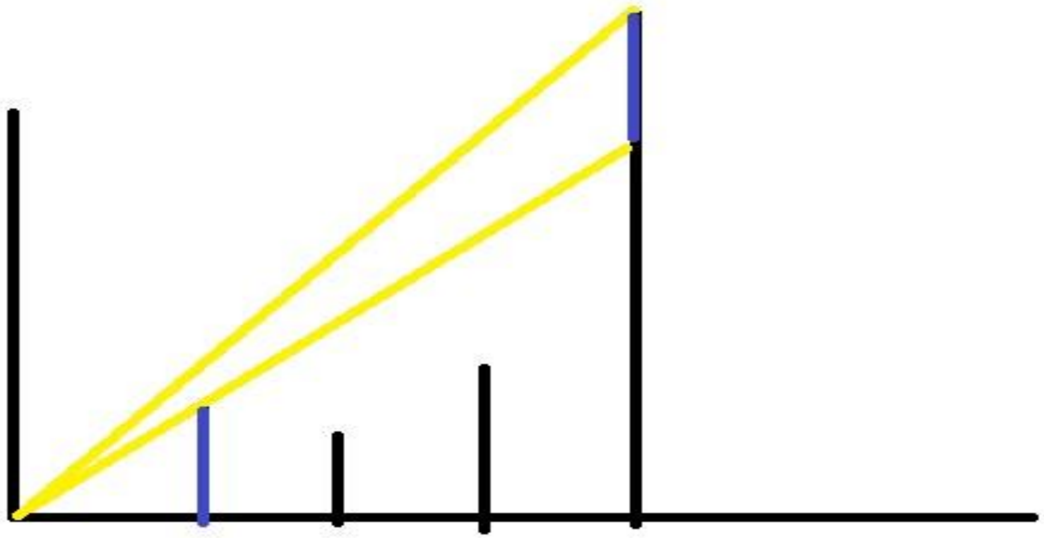
CSE-633 Parallel Algorithms

Wednesday 23rd April 2014

Problem Definition

- Given a set of line segments which are perpendicular to the x axis , determine what piece of each line segment would be visible from origin .
- Input :- A set of y coordinate values
- Output :- A set of results which contain the piece of each line segement that is visible

Illustration



Assumptions

- One end of the line segment is at the x-axis.
- The lines are placed unit distance apart in x-axis .
- The data is arranged in ascending order with respect to the x axis values.
- One cannot see through the lines

Applications

- When creating a virtual reality , one would want the users to not be able to look through walls/buildings.
- Determining the length of a runway when building an airport.

What I've done

- Developed a serial algorithm
- Developed a parallel algorithm
- Compared them.

Serial Algorithm

- Read the data.
- Currentslope=y[0]/x[0]
- For i=1 to n
- If y[i]>Currentslope*x[i]
- out[i]=y[i]-Currentslope*x[i]
- Currentslope=y[i]/x[i]
- Else
- out[i]=0;
- Print out[i].

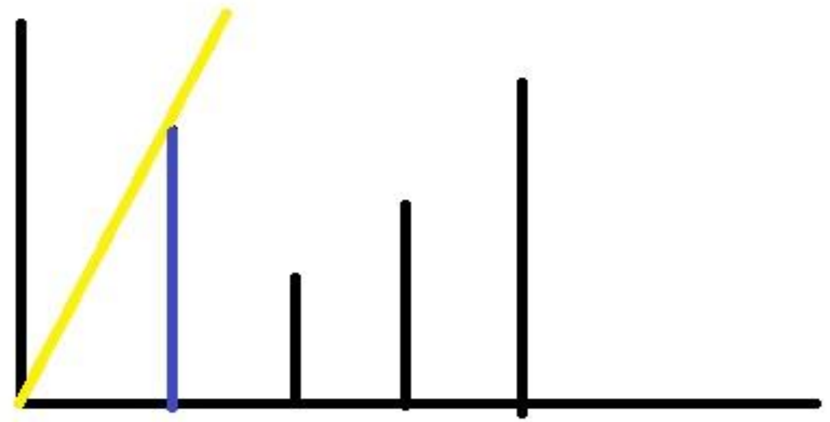
Parallel Solution

- Parallel prefix on the maximum slope .
- Perform the serial algorithm to get the results within the particular processor
- Pass on the maximum slope seen within the processor to the next processor
- If the value obtained is greater than maximum local slope then

  pass on the value obtained to the next processor
- Else pass on maximum local slope to next processor

  recalculate the local values with the value obtained .

Implementation – Input Data problem



Implementation – generation of input data

- Created dataset using random function.
- prevslope=0
- If rand()%4==3

  arr[i]=prevslope*i+.001

  prevslope=arr[i]/I;
- Else arr[i]=0

Implementation – distribution of data

- Processor with rank=0 creates the dataset
- Rank0 processor then distributes the entire dataset to all other processors
- Since all processors know n(size of data) and p(number of processors) they'll calculate the lower limit and upper limit of the entire data they are supposed to read
- Lower limit = ((n/p)*rank)+1
- Upper limit = (n/p)*(rank+1)

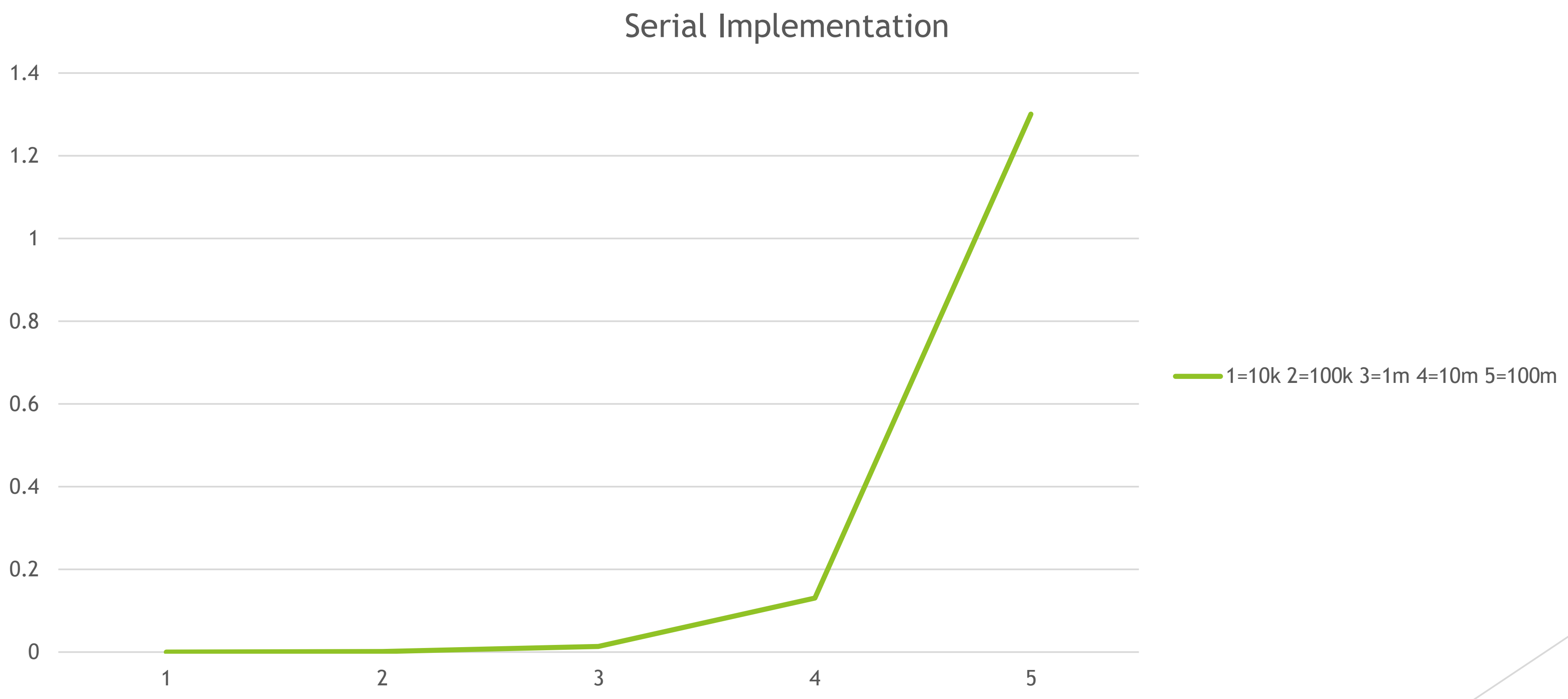
Running Time Serial Implementation

10,000	0.000131
100,000	0.001348
1,000,000	0.013202
10,000,000	0.129934
100,000,000	1.300618

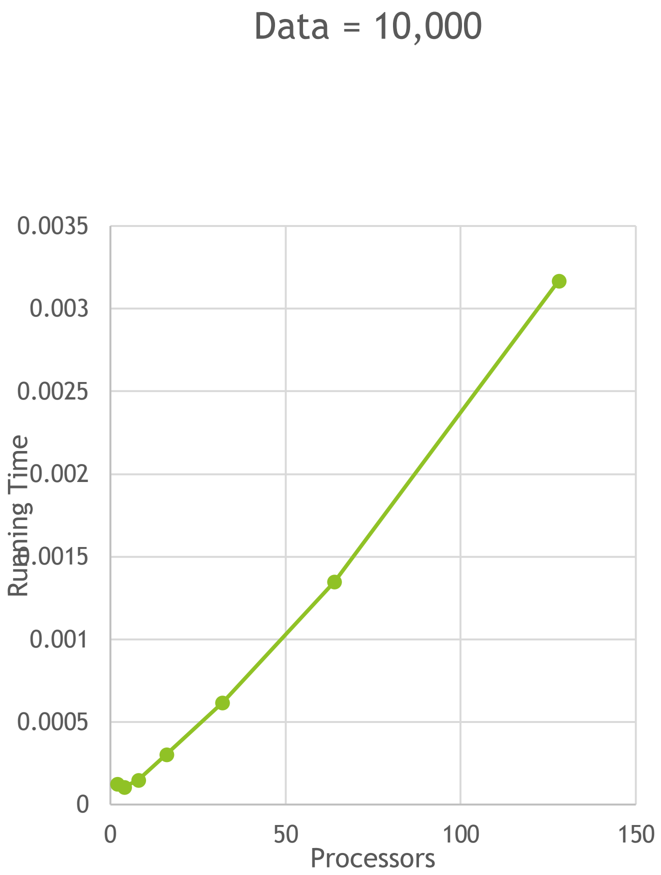
Running Time – Parallel Implementation

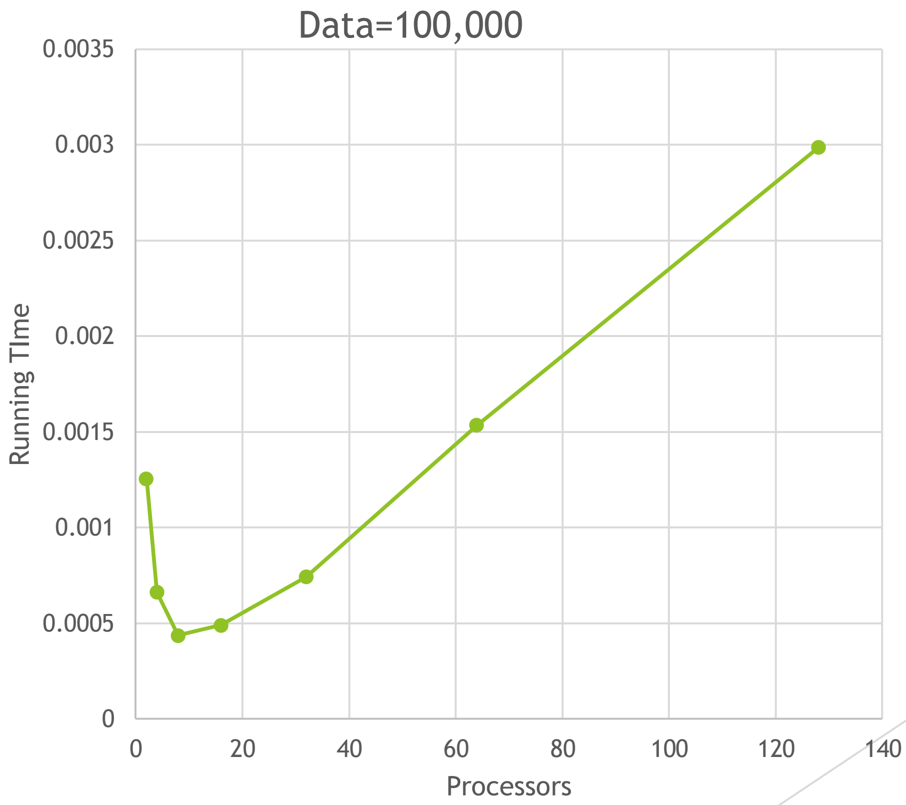
	10,000	100,000	1,000,000	10,000,000	100,000,000
2	0.000125	0.001257	0.012603	0.09296	0.300813
4	0.000105	0.000663	0.006354	0.031587	0.275691
8	0.000149	0.000436	0.003248	0.016084	0.135961
16	0.000303	0.00049	0.002037	0.008422	0.074135
32	0.000617	0.000742	0.00176	0.005049	0.040772
64	0.001349	0.001536	0.001824	0.003948	0.021979
128	0.003168	0.002987	0.002502	0.002737	0.01576

Results – Serial Implementation



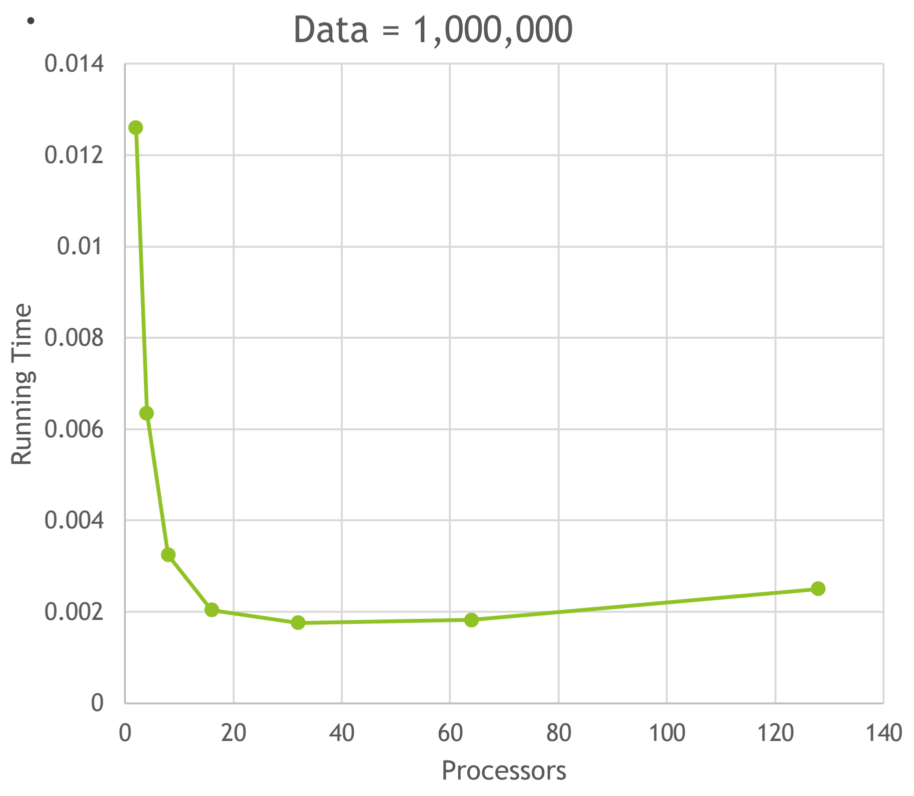
Result – parallel implementation

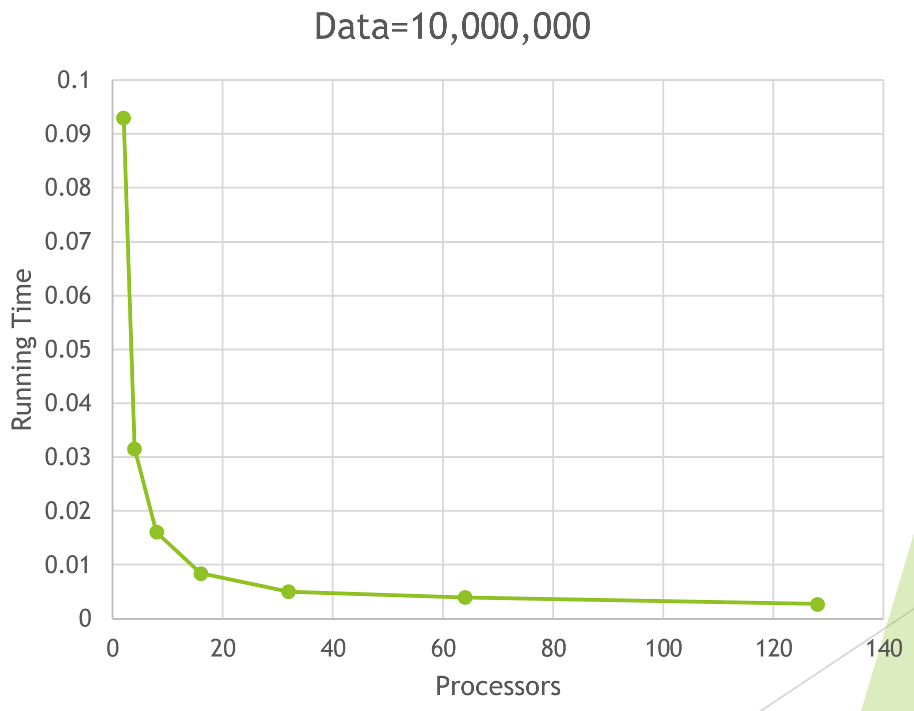




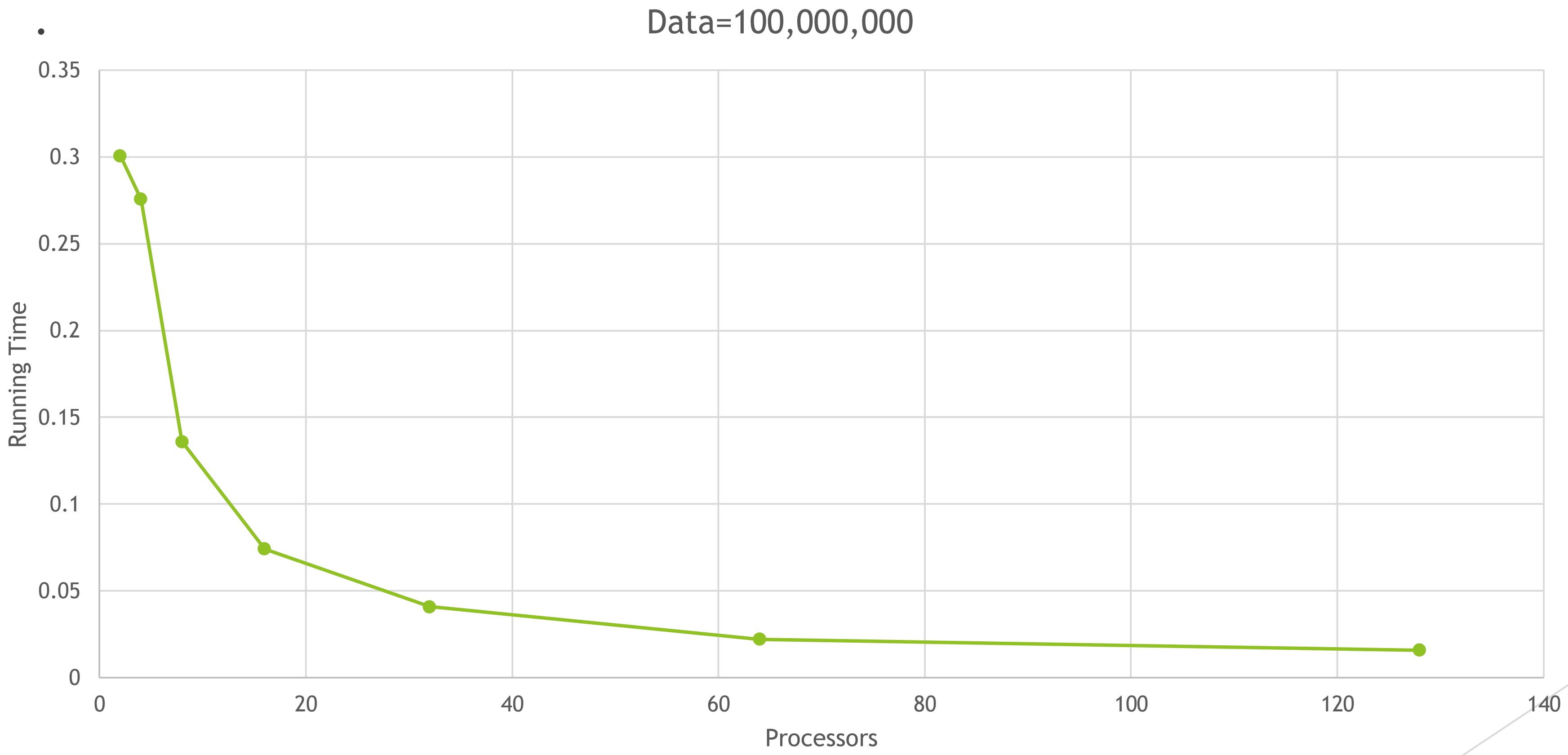
Results – parallel implementation

•





Results – parallel implementation

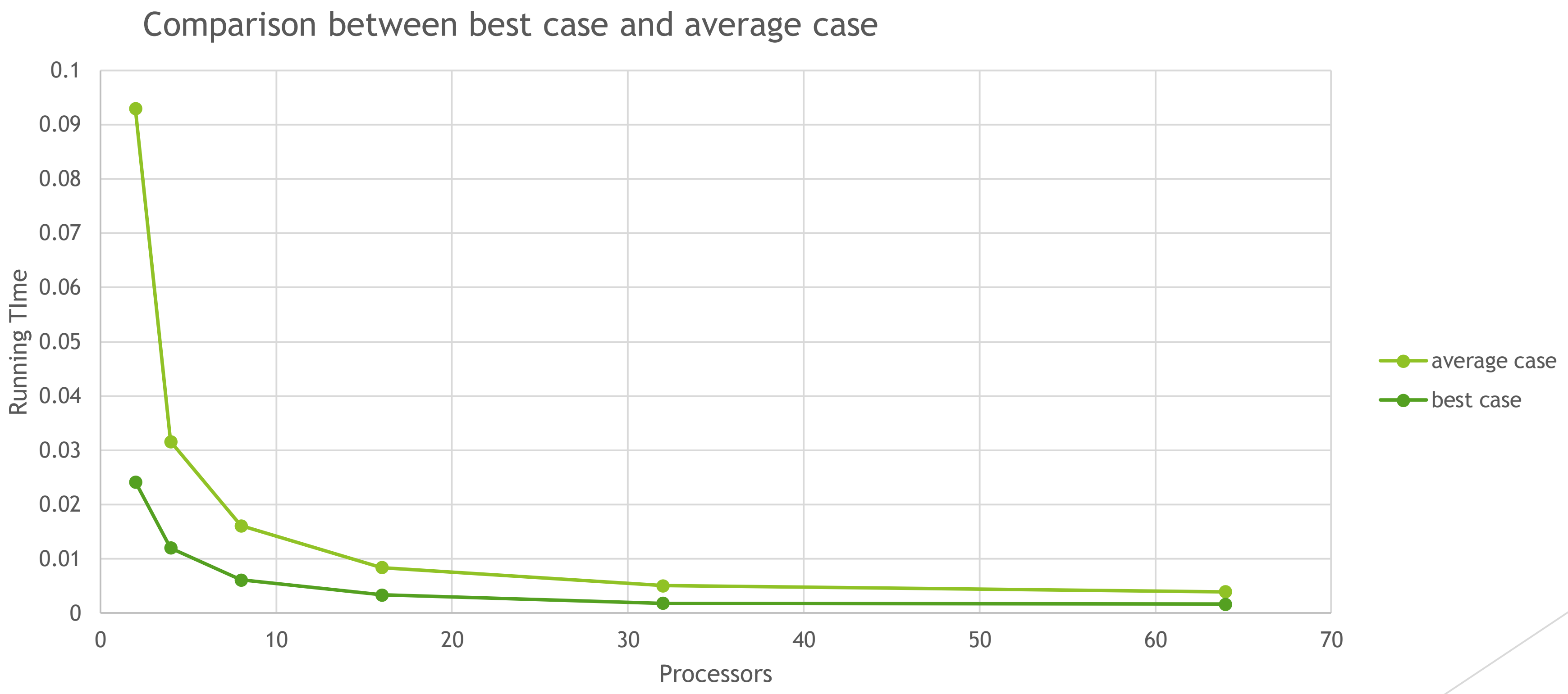
- 

Best Case vs Average/Worst Case

- Best case scenario : In the best scenario , the very first line would be visible and nothing beyond it will be . This is best case because all the local prefixes will be skipped because the local maximum slope will be less than the received slope .
- Average case : In the average case , there may or may not be a local maximum slope which is greater than the received slope .
- Worst case scenario : In the worst case scenario the local maximum is always greater than the prefix value that was received , so thus all local prefix calculations have to be done .

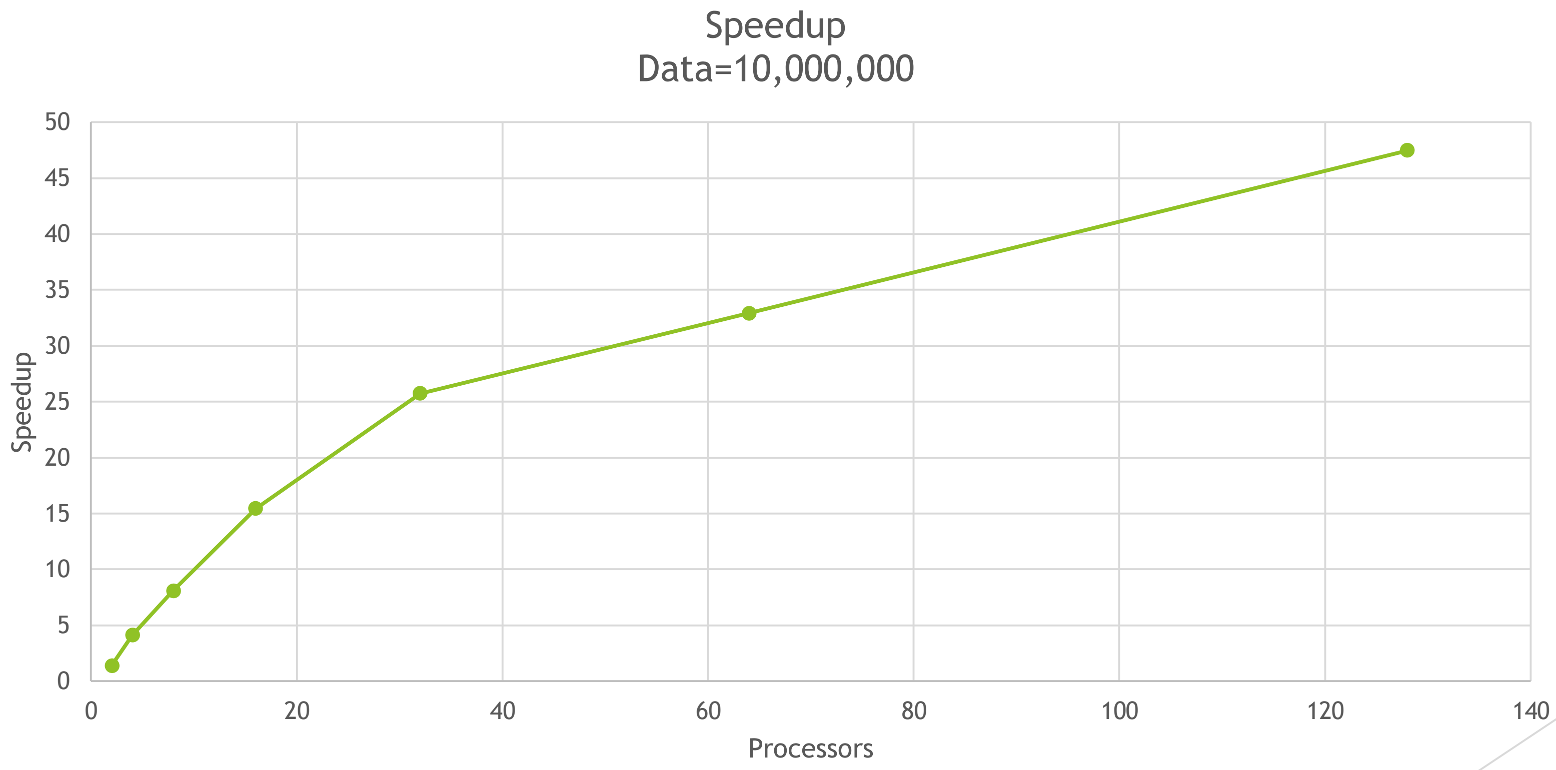
Best vs Average/worst case scenario

•



Speedup(10m data)

•



Questions ?