CSE 702: SEMINAR ON PROGRAMMING MASSIVELY PARALLEL SYSTEMS

Implementing Parallel Prefix Minima using Message Passing Interface

University at Buffalo
School of Engineering and Applied Sciences

PREPARED BY:

VIJAYA HARSHAVARDHAN PALLA

(UB PERSON NUMBER: 50246589)

UNDER THE GUIDANCE OF:

PROF. DR. RUSS MILLER

SUNY DISTINGUISHED PROFESSOR, DEPARTMENT

OF COMPUTER SCIENCE AND ENGINEERING,

UNIVERSITY AT BUFFALO (SUNY)

Agenda

- What is parallel prefix?
- Parallel Prefix Minima
- Sequential Algorithm
- Parallel Algorithm
- Snapshot of the Parallel Algorithm implemented
- Processors vs Time for fixed data size in each processor
- Processors vs Time for small total data size
- Processors vs Time for a large total data size
- What I learnt!
- References

What is parallel prefix?

The parallel execution of an operation that is defined by a recurrence involving an associative operator.

For e.g:

If there is an Array ‘A’ with n elements in it then the parallel prefix at Ai is defined as:

$Ai = A1 \oplus A2 \oplus \ldots\ldots. \; Ai-1 \oplus Ai$

Here, $\oplus$ is any operation like Sum, Minima, Maxima, Product, etc.

Parallel Prefix Minima

- If we replace $\oplus$ with a minima operation in the previous slide, it gives us Parallel Prefix Minima.
- This will let us know at any given element, what is the minimum of the array till the given element.

Sequential Algorithm

Given an array A of n elements

- Go through each element in a sequential order
- Find parallel prefix at each element by finding the minima between the element and the parallel prefix of the element before it
- Runtime: **Θ (n)** n: No of elements in an array

Parallel Algorithm

- It can be seen in the previous slide that the runtime of the algorithm is equivalent to the number of elements in the array
- We can get it better by implementing the algorithm in parallel which gives the base to the following algorithm

For better understanding I am going to consider there are n processors and each processor gets one element.

P1, P2, P3, P4,……. Pn-1,Pn

P1 stores n1, P2 stores n2 and so on..

Parallel Algorithm contd..

- Following algorithm is used to implement parallel prefix minima in log(n) steps where n is no. of processors participating.

```
for j=0 to log(n)-1 do
 for i=2^j to n-1 in parallel do
  A[i]=min(A[i],A[i-2^j])
```

- Here A[i] is the element present in the processor ‘i’
- Second for loop runs in parallel and takes **θ(1)** time
- Hence total time taken is **θ(logn)**

Parallel Algorithm contd..

- It is not idealistic to have one processor for each element as it is going to be costly.
- So, we consider p processors and each processor get n/p elements.

Parallel prefix is calculated in three main steps:

1. **Pre-processing**

Every processor simultaneously and in parallel finds out parallel prefix for n/p elements.

2. Processing

In this step, parallel prefix of all the last elements in each processor (Total p elements and p processors) is calculated using the algorithm mentioned in the earlier slide in **Θ(Log(p))** time

Parallel Algorithm contd..

3. Post processing

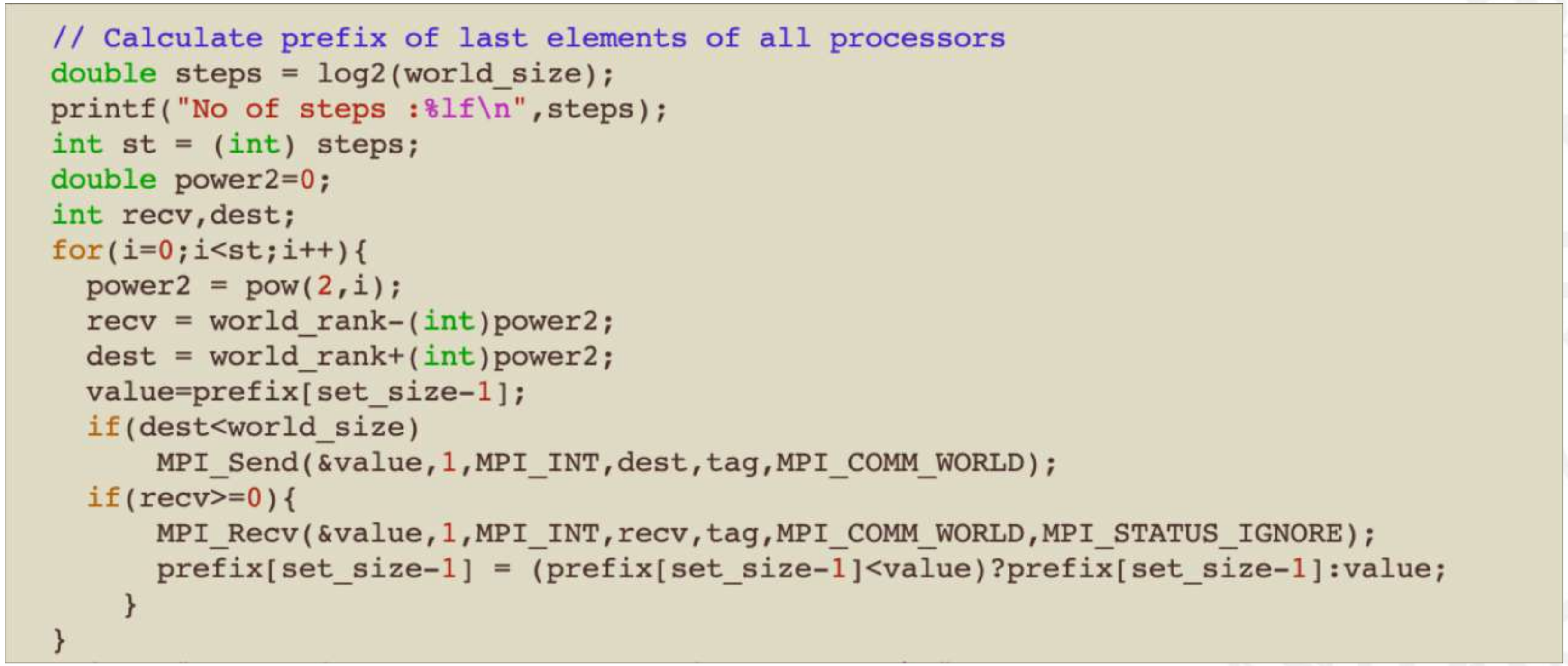
Each processors’ last element has the correct prefix value whereas other elements in any given processor are missing one value to find their ultimate prefix value. This one value is the prefix value of the last element of the processor before it.

So, every processor sends it last elements prefix value to its next processor simultaneously

Each processor runs parallel prefix with the received value across all elements.

Now, we have reached the end where parallel prefix at each element has been found out.

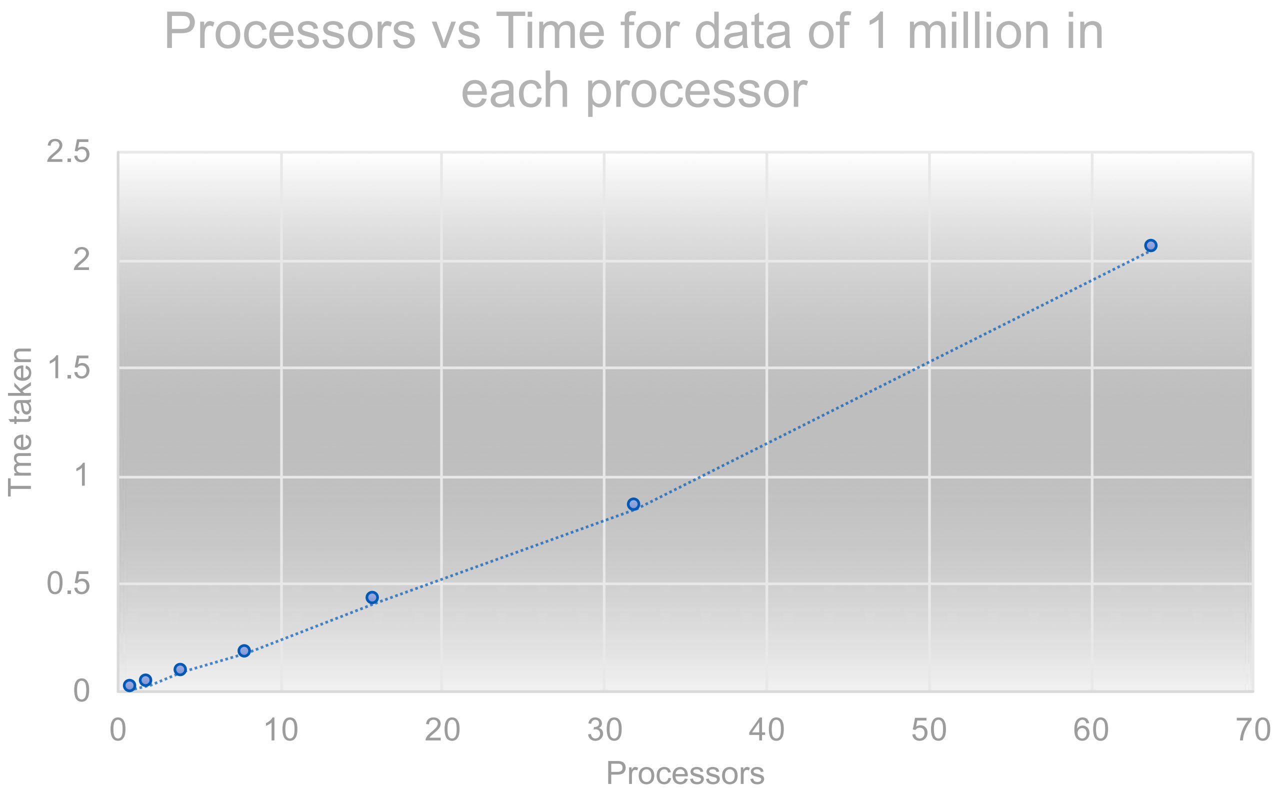
Snapshot of the Parallel Algorithm implemented



```
// Calculate prefix of last elements of all processors
double steps = log2(world_size);
printf("No of steps :%lf\n",steps);
int st = (int) steps;
double power2=0;
int recv,dest;
for(i=0;i<st;i++){
  power2 = pow(2,i);
  recv = world_rank-(int)power2;
  dest = world_rank+(int)power2;
  value=prefix[set_size-1];
  if(dest<world_size)
      MPI_Send(&value,1,MPI_INT,dest,tag,MPI_COMM_WORLD);
  if(recv>=0){
      MPI_Recv(&value,1,MPI_INT,recv,tag,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
      prefix[set_size-1] = (prefix[set_size-1]<value)?prefix[set_size-1]:value;
    }
}
```

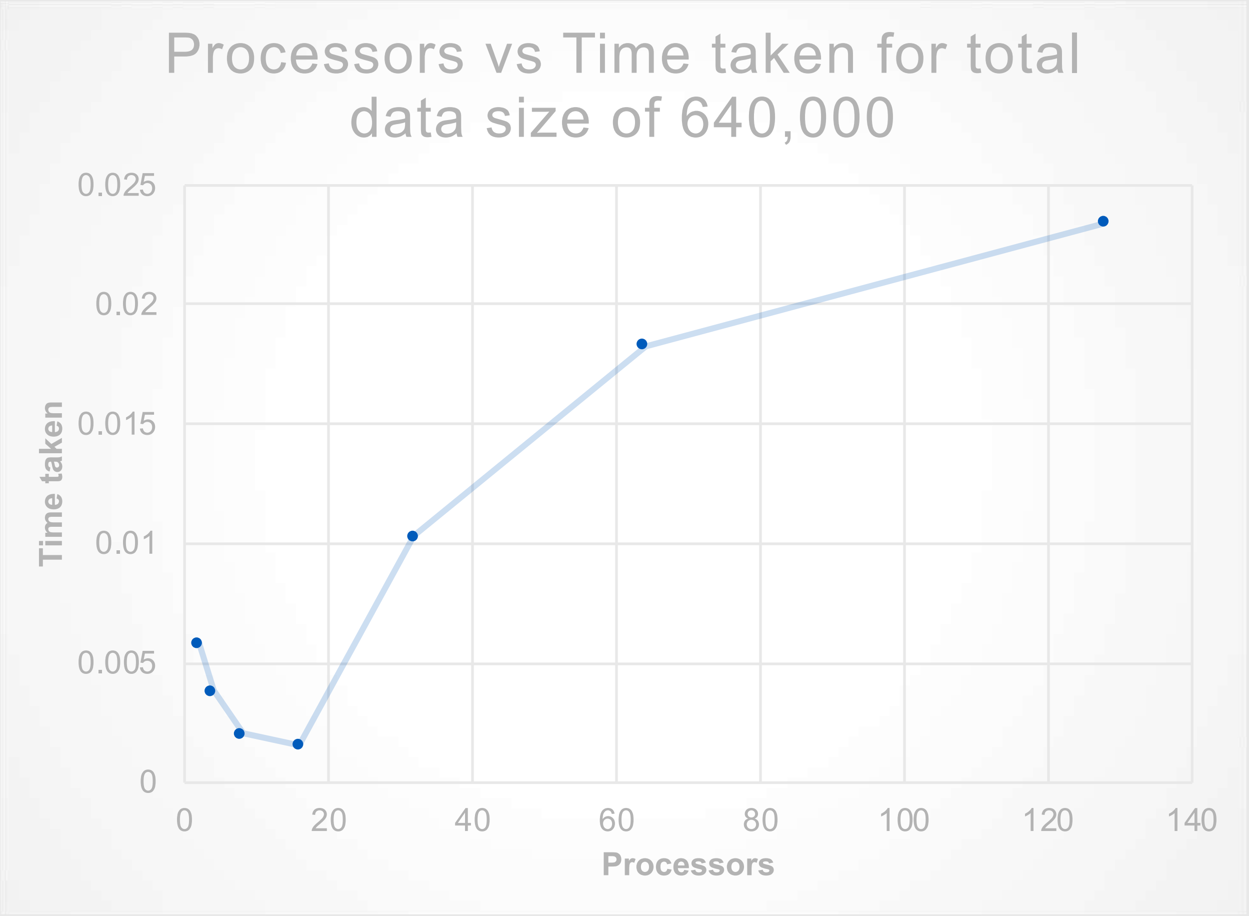
Processors vs Time for fixed data size in each processor

Processors	Time (In Seconds)
1	0.0115
2	0.032784
4	0.088401
8	0.177595
16	0.419077
32	0.851107
64	2.05456



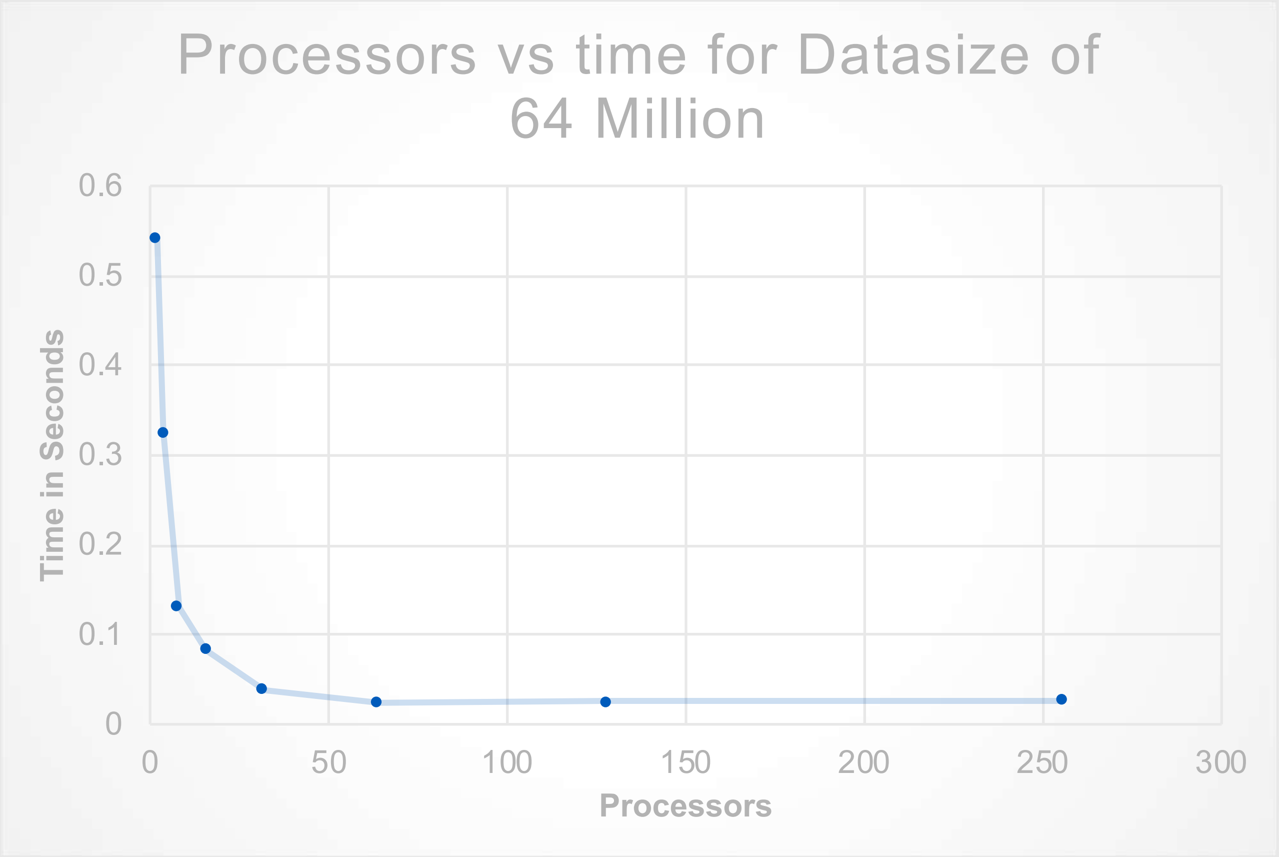
Processors vs Time for small total data size (640000)

Processors	Time (In Seconds)
2	0.005823
4	0.003833
8	0.002064
16	0.001563
32	0.010316
64	0.0183
128	0.023428



Processors vs Time for large total data size (64 Million)

Processors	Time (In Seconds)
1	0.48615
2	0.542148
4	0.32567
8	0.130272
16	0.08345
32	0.03801
64	0.023948
128	0.025
256	0.027382



Speedup

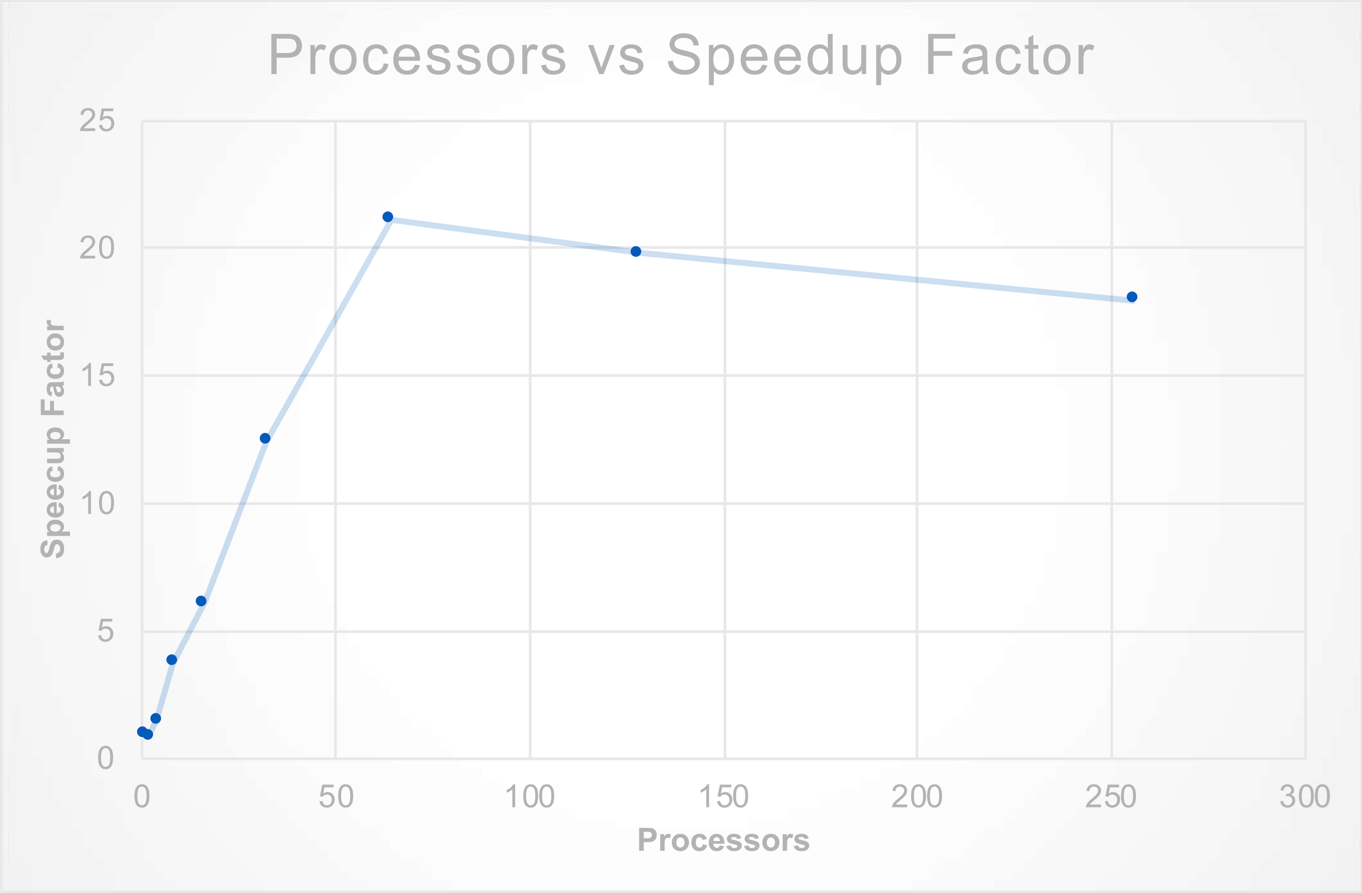
Wikipedia Definition:

In computer architecture, **speedup** is a number that measures the relative performance of two systems processing the same problem. More technically, it is the improvement in speed of execution of a task executed on two similar architectures with different resources. The notion of speedup was established by Amdahl's law, which was particularly focused on parallel processing. However, speedup can be used more generally to show the effect on performance after any resource enhancement.

So, the formula used for calculating speed up is Time taken by one processor divided by time taken by two or more processors.

Speed Up Analysis

Processors	Speedup Factor
1	1
2	0.9
4	1.5
8	3.74
16	6.1
32	12.46
64	21.13
128	19.8
256	18



What I have learnt!

- How to use Message passing interface model to communicate across processors to implement algorithm in parallel
- Depending on the data one needs to choose if increasing total processors is a wise thing or not
- Increasing processors is not always directly proportional to reduction in the time taken to complete the task

References

http://www2.hawaii.edu/~nodari/teaching/f17/notes/notes06.pdf

http://www.cs.princeton.edu/courses/archive/fall13/cos326/lec/23-parallel-scan.pdf

https://buffalo.app.box.com/s/vb6lkxg72jgekuyo5xbps7xesfj076ok

https://en.wikipedia.org/wiki/Speedup

THANK YOU

University at Buffalo
School of Engineering and Applied Sciences