STUDY AND PERFORMANCE ANALYSIS OF THE WYLLIE'S LIST RANKING ALGORITHM USING VARIOUS PARALLEL PROGRAMMING MODELS

B. Muni Lavanya

JNTUA college of Engineering,
Pulivendula, A.P. India
b.muni.lavanya19@gmail.com

Abstract

The Wyllie's list ranking algorithm takes a linked list data structure as an input and it pass the linked list successor elements to the succ1 array to find the Rank. The algorithm depends upon the Pointer jumping operation and its concepts. The motivation for this work is to parallelize the Wyllie's list ranking algorithm using three different parallel programming platforms and compare its performance on all the three platforms.

***Keywords*: Wyllies algorithm,Pointer jumping,Performance analysis,parallelization.**

1. INTRODUCTION

The Wyllie's list ranking operation takes a linked list data structure as an input and passes the linked list successor elements to the succ1 array and thus it calculate the last element in the list .Again, it calculates the rank of the each node to the end of the list. This is the aim of the algorithm; the algorithm depends on the Pointer jumping operation. The motivation for this work is to parallelize the Wyllie's operation using three different parallel programming platforms and compare the performance on all platforms. It is done because in the serial algorithm, the program simply traverses each link in the link list. So to improve the performance of the algorithm, parallelization needs to be done. We have used three programming models: Open MP, MPI and Concurrent Java to realize the parallel version of our custom algorithm..

2. PARALLEL WYLLIE'S ALGORITHM

Algorithm for pointer jumping operation in Wyllie's algorithm

```
Input: Linked list elements
Output: Rank of each node to the end node
Method: pointerjump
Begin
1.      Read list successor element in succ array
2.      For every instruction of i
Do
3.      If succ1 equal to i
then
4.      Last1[i] ←succ1[i] - n;
5.      rank[i] = 0;
   else
6.      last1[i] ← succ1[i];
7.      rank[i] = 1;
od
8.      boolean f= 1;
9.      while f
do
```

```
10.      f = 0;
11.     for every instruction of i
do
12.     if last1[i] greater than equal to 0
then
13.     f = 1;
14.     rank[i] ← rank[i]+ rank[last1[i]];
15.     last1[i] ← last1[last1[i]];
od od
16.     for every instruction of i
17.     last1[i] ← last1[i]+ n;
od
End
```

In the above algorithm we are passing successor elements to succ1 array, if succ1 is equal to i,make rank as zero otherwise one and also make last1 equal to succ-n .In second for loop, if last1 is greater than and equal to zero, find the rank and last1.After that adjust last1 by adding number of nodes.

3. FLOW CHART

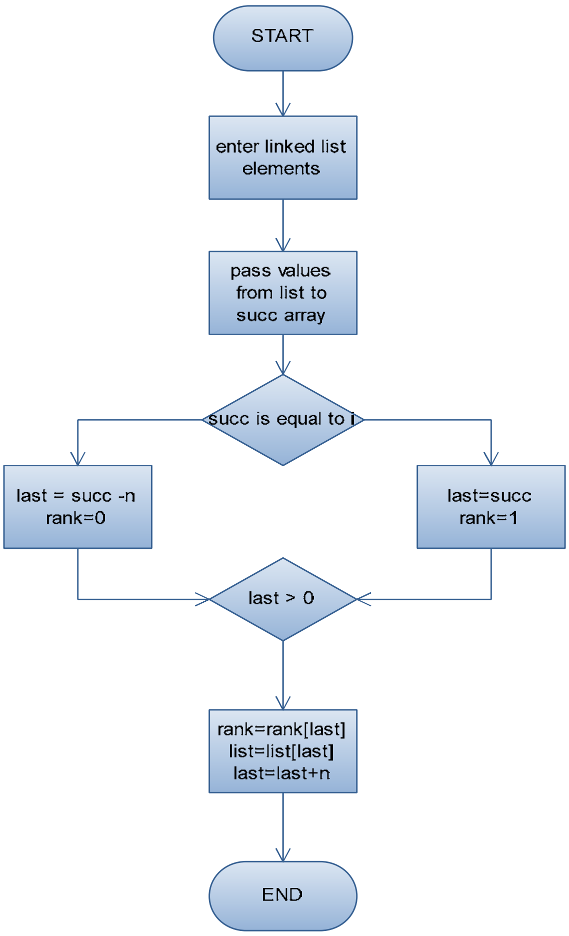

Fig.1:Flow chart

It specifies the flow of control of our algorithm.The algorithm is parallelized in pointer jumping function after initializing the last1 and rank variables; finally we have to find rank and last1. Rank calculation is the final output of the algorithm.

4. METHODOLOGY

OpenMP: #pragma omp parallel for , #pragma omp critical

MPI:MPI_Send(),MPI_Recv(),MPI_Init(),MPI_Comm_size(),MPI_Comm_rank().

Concurrent java:newFixedThreadPool(),execute(),Runnable(),currentTimeMillis();

Major headings should be typeset in boldface with the first letter of important words capitalized.

5. RESULT

5.1. *OpenMP:*

nodes	serial	parallel
5	11296.4	10294.6
8	11502.8	10831
10	9626.8	7537.2
12	13690.2	12915.4

Table1:OMP

The above table gives the serial and parallel execution times of OPENMP.

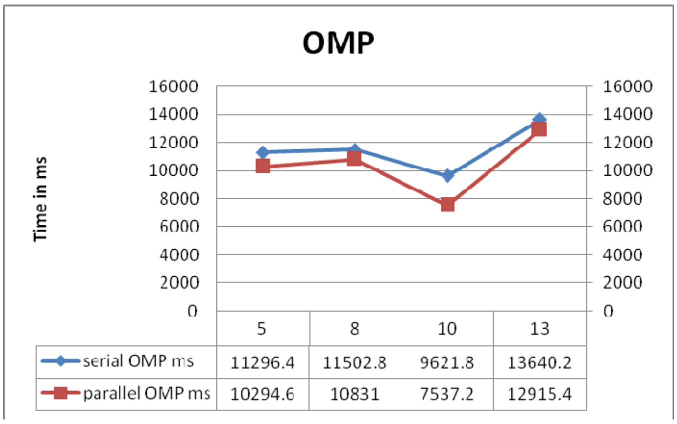

FIG.2: OMP

We compared the parallel and sequential execution timing of Wyllie's list ranking algorithm using OPENMP. Parallel OMP gives better execution time as compared to sequential; it can be analyzed easily in the above graph.

5.2. *MPI:*

nodes	serial	parallel
5	2514.76	2508.57
8	3572.25	3428.66
10	4026.35	4026.35
12	5984.48	3998.22
20	6539.22	4009.49

Table 2:MPI

The above table gives serial and parallel execution time of MPI for different nodes values of the linked list.

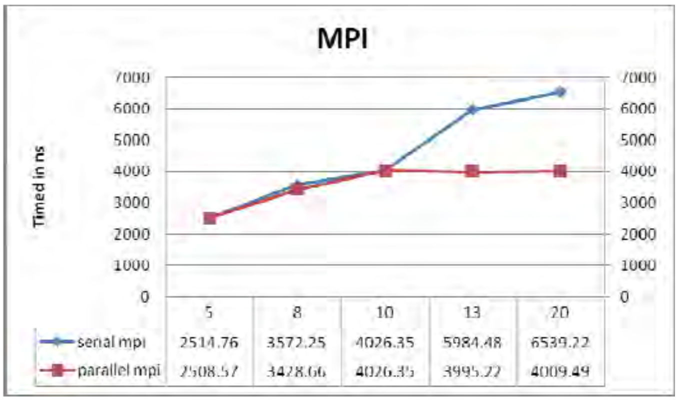

Fig.3:MPI

We compared the parallel and sequential execution timings of Wyllie's list ranking algorithm using MPI. It can be identified that up to a certain limit, both behaves as same, but for larger nodes values, parallel computation time is far better than the sequential one.

5.3. ***Concurrent JAVA:***

nodes	serial	parallel
5	17.4	4
8	42.26	15
10	18.53	6
12	27.18	10
20	33.54	16

Table 3:Concurrent Java

The above table gives serial and parallel execution timings of Concurrent Java

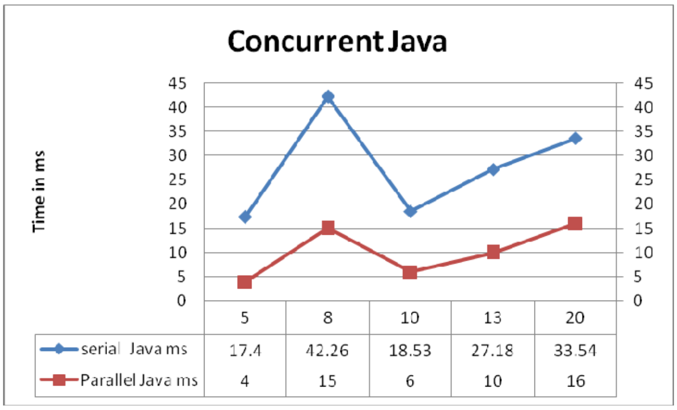

Fig.4:CONCURRENT JAVA

We compared the parallel and sequential execution timings of Wyllie's list ranking algorithm using Concurrent java. Parallel implementation of the algorithm gives best performance as compared to sequential one, i.e. parallel takes less time as compared to sequential.

We compared all three algorithms in which concurrent java gives better performance compare to OpenMp and Mpi

6. CONCLUSION

In this work we have studied and analyzed the performance increase by parallelizing the Wyllie's list ranking algorithm. We have run the parallel version of the three programming models where speedup is attained as opposed to the sequential version. Concurrent java has given the best speedup and is consequently the best programming platform for running this particular algorithm.

7. References

[1] R.cole and U.vishikin.Fast optimal parallel prefix sum and List ranking, Tel Aviv university,December 1986.
[2] R.cole and U.vishikin.Appropriate scheduling,exact scheduling and parallel algorithms .In 27th symposium in foundation of computer science,1986.
[3] Y.Han Designing fast and efficient parallel algorithms.P.hd thesis,Duke university,1987.
[4] M.Suhail Rehman,Kishore Kottapalli and P.J.Narayanan.Fast and schedulable list ranking.Proceedings on 23rd international conference on super computing.Newyork,NY,USA,2009
[5] Joseph An Introduction to parallel algorithms.addition Wesley Longman Publishing co.Redwoodcity,CA,USA,1992.
[6] R.J.Anderson and G.L.Miller .Deterministic parallel list ranking ,pages 81-90,1988.
[7] R.J.Anderson and G.L.Miller.A simple randamized parallel algorithm for list ranking 1990.
[8] David A.Bader,Virat Agarwal and Kamesh Madhuri.on the design and analysis of irregular algorithms.A case study of list ranking.In 27th international parallel and distributed processing symposium,proceeding 26-30 march 2007,Long Beach,California,USA.