A Parallel Random Access Machine (PRAM) Model for English Language Recognizer (PRAM-ELR)

Madiha Khurram Pasha
Department of Computer Science (UBIT)
University of Karachi

Maryam Feroze
Department of Computer Science (UBIT)
University of Karachi

Khurram Ahmad Pasha
Department of Computer Science
Beaconhouse School System

ABSTRACT

As, technology grows day by day ,computers become ever faster with its importance and having maximum computing power, computational cost also increases, development at its high end then we state that parallel computing technology (parallelism) is the main building block of this era of development and, for this purpose, therefore, all aspects have been thought about to meet the need of time.

Uni-processing abstracted into parallel processing. Many visions of parallel computing included Parallel Random Access Machine (PRAM), Graphical Processing Unit (GPU) , Compute Unified Device Architecture extension in C-language (CUDA –C) , Local Area Multicomputer- Message Passing Interface (LAM-MPI) ,cluster and grid computer etc have been used in different fields.

Many of the sequential algorithms can be implemented parallel to increase efficiency, computing power, computational cost and speed. Here, we proposed a Parallel Random Access Model (PRAM) of the algorithm of ELR – NPDA [1] to make it more efficient, fast and powerful so that, this new parallel version of ELR–NPDA [1] called PRAM-ELR can be parse large data set speedier.

General Terms

Parallel Computing, Uni-Processing, Cluster, Grid, Parallel Random Access Machine (PRAM), Graphical Processing

Unit (GPU), Compute Unified Device Architecture extension in C-language (CUDA –C), Local Area Multicomputer-Message Passing Interface (LAM-MPI).

Keywords

English Language Recognizer-Nondeterministic Pushdown Automata (ELR-NPDA), Parallel Random Access Machine-English Language Recognizer (PRAM-ELR).

1. INTRODUCTION

Parallel computing is a group of calculations or "computations" where these calculations run simultaneously. Parallel computing is a paradigm where several processors execute several applications in a parallel manner. The basic standard architectures that based on parallel computing or parallel processing are Single Instruction Multiple Data (SIMD) and Multiple Instructions Multiple Data (MIMD). In SIMD 'single instruction' means single control unit (CU) or single processor and 'multiple data' means multiple arithmetic and logic unit (ALU) and memory or we can say that every ALU have its own memory and all ALU perform same operations because source CU is same for all ALU. SIMD having two types array processing and vector processing. (See figure 1 and 2 for SIMD architecture.)

On the other hand, In MIMD 'multiple instructions' by way of multiple processors having its own CU and ALU but memory is shared or distributed among the processors and 'multiple data' means it can execute different tasks of different processors . It can also be classified into two major categories, such as, multiprocessor and multicomputer (see figure 3).

MIMD multiprocessor is also called Symmetric Multi-Processor (SMP) because in which all processors symmetric and identical, but they can do same and different tasks at a time. Also the main memory is shared among all the processors (see fig 4). The second category of MIMD is multicomputer also called distributed memory architecture. It has multiple computers which may be single instruction single data (SISD), single instruction multiple data (SIMD), SMP and etc as shown in figure 5.

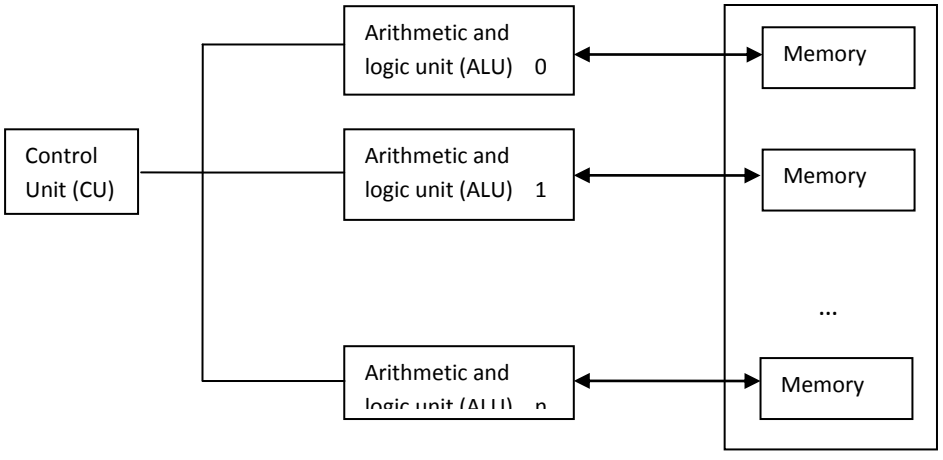


Fig 1: SIMD Vector Architecture

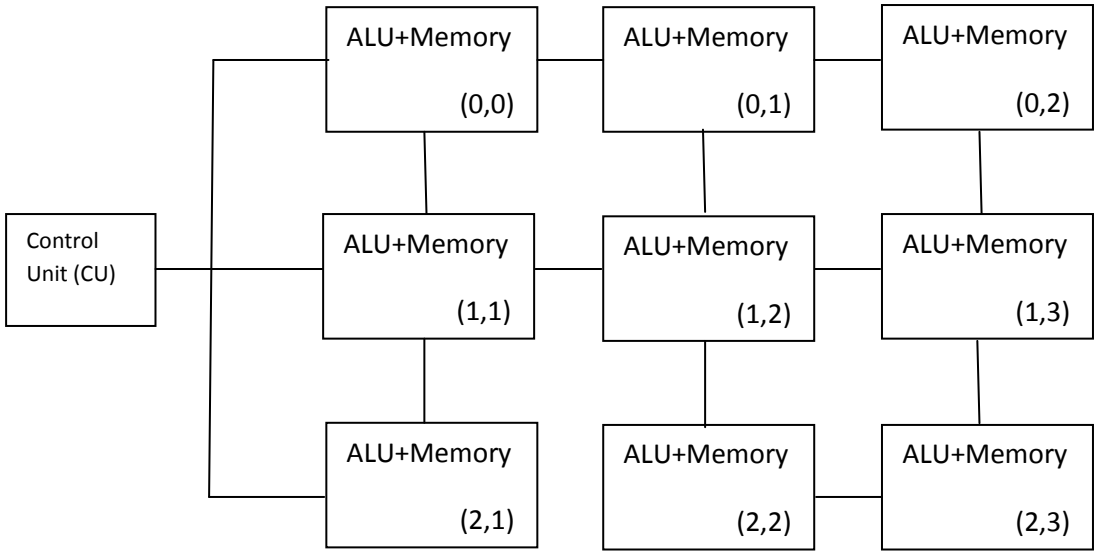


Fig 2: SIMD Array Architecture

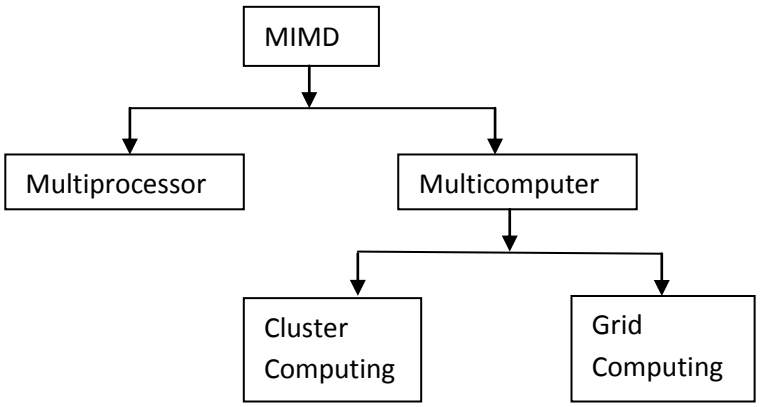


Fig 3: Classification of MIMD

The MIMD multicomputers further classifies into cluster and gird computing. Cluster computing is suitable for dedicated environment in which all computers must be homogeneous having same operating systems. It is fast but not appropriate for large network. Grid computing may be groups of clusters. It may be homogeneous or heterogonous. It is also fast but apposite for very large networks. Operating systems may be different to each others. (See table 1)

Table 1: Difference between cluster computing and grid computing

CLUSTER	GRID
• Groups of homogenous computers	• Groups of homogenous and hetrogenous computers
• Processing speed fast	• Processing speed very fast
• Not suitable for large network.	• Suitable for large networks
• Network size: 1000 Pcs	• Network size: 10^6, 10^{10}
• Same operating systems	• Different operating systems
• Short term result required	• Long term result required

PRAM is the extended form of RAM .The RAM (random access model) used to analyze sequential algorithm and complexity analysis can be done by using asymptotic notation. While PRAM is another model for analyzing parallel algorithm and specially designed for parallel machines which consists of unlimited set of processors , unlimited collection of shared memory , each processor having its own set of registers and all processors processes PRAM instruction simultaneously. The diagrammatical view of PRAM model can be seen in figure 6. The complexity of PRAM can be calculated as:

Run Time = T(n)

No of processors = P(n)
Cost= C(n) = T(n) * P(n)

The PRAM model can used sequential algorithm in a parallel manner. We apply this PRAM over the algorithm of ELR-NPDA[1].

There are four memory access methods and sub classes of PRAM model (See figure 7). The memory access methods are Exclusive Read (ER), Exclusive Write (EW), Concurrent Read (CR), and Concurrent Write (CW). In ER not more than one processor can read the same memory location at a time. In CR all processors can read the same memory location at a time. While in EW not more than one processors can write the same memory location at a time. In CW all processors can write the same memory location at a time. The sub classes are Exclusive Read and Exclusive Write (EREW), Exclusive Read and Concurrent Write (ERCW), Concurrent Read and Exclusive Write (CREW), and Concurrent Read and Concurrent Write (CRCW). The proposed PRAM model is based on CRCW.

2. LITERATURE REVIEW

The author of [1] proposed an algorithm that can parse valid English language sentences by using the technique of non deterministic pushdown automata (NPDA). There is an inadequacy that it cannot parse large data set very rapid. We proposed an effective way to overcome this limitation by proposing the PRAM model for ELR-NPDA. The anticipated model can easily be map in Graphical Processing Unit (GPU) and can be implemented in CUDA-C and Open-CL.

The PRAM can be applied on many areas to increase the efficiency of RAM algorithms. A Flexible Job Shop Scheduling (FJSP) model which based on genetic algorithm for parallel machine [2] has been anticipated. This model having two major modules: machine selection module (MSM) and operation scheduling module (OSM). The MSM used to select one of the parallel machines and OSM used to arrange the sequences of all assigned operations to machine. A guideline for genetic algorithms based on parallel machine scheduling and Flexible Job Shop Scheduling (FJSP) [3] have been projected. Resource scheduling is one of the major problem in cloud computing. For this problem a parallel genetic algorithm (PGA) [4] have been proposed which is faster than sequential genetic algorithm and can efficiently allocate resources in cloud computing.

In [5], author planned the extended form of Strassen's fast matrix multiplication algorithm, which is more proficient then classical Strassen's algorithm. The traveling salesman problem (TSP) is most broadly studied topic. The author of [6] proposed a parallel immune algorithm for TSP which based on GPU. This one is more efficient and effective than sequential TSP algorithm. The first parallel algorithm to compute perfect reuse distance for characterizing data cache locality proposed in [7].

The authors of [8] develop a PRAM having four symmetric and similar parallel ports by using VHDL hardware description language. In this parallel system two important instances, first one is used as shared instruction memory and the second one is used as shared data memory. A very detailed survey of PRAM has been conducted in [9]. In [10] the researchers show that how the PRAM model mapped onto the Intel SCC many-core processor.

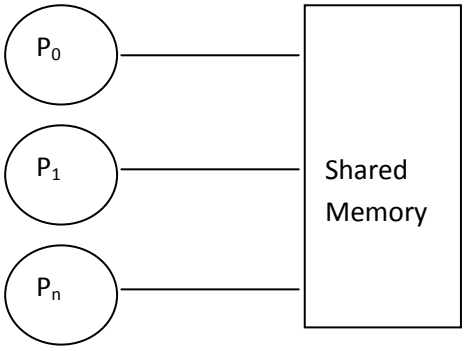


Fig 4: MIMD Shared Architecture

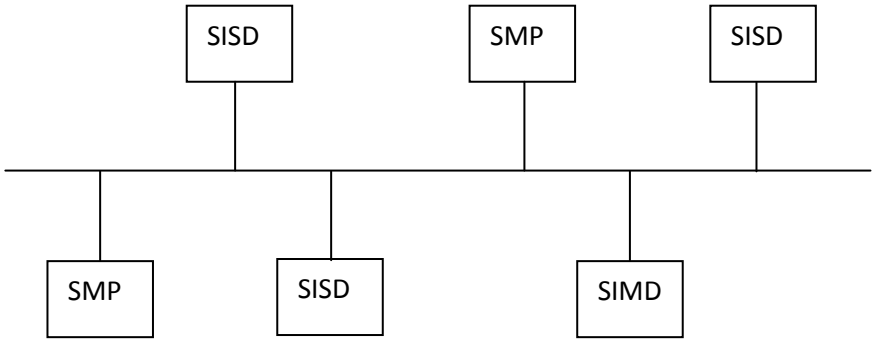


Fig 5: MIMD Distributed Architecture

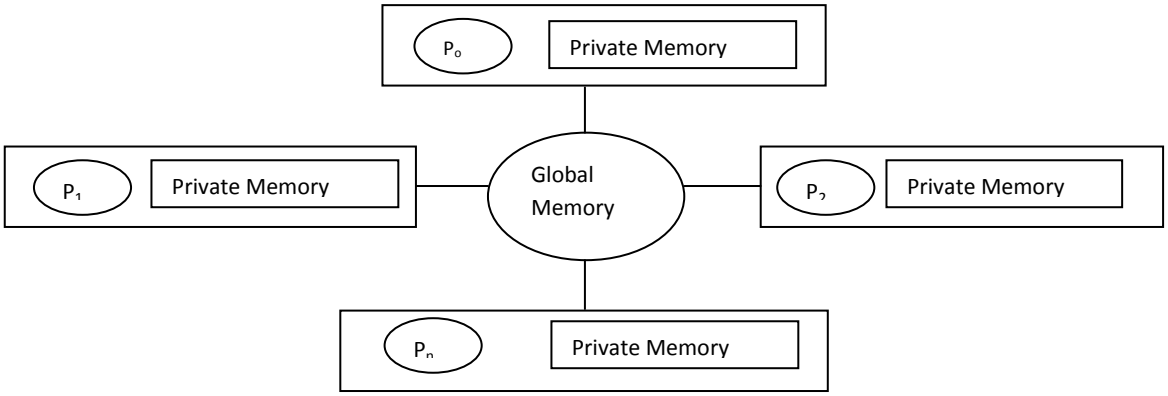


Fig 6: The PRAM Model for Parallel Processing

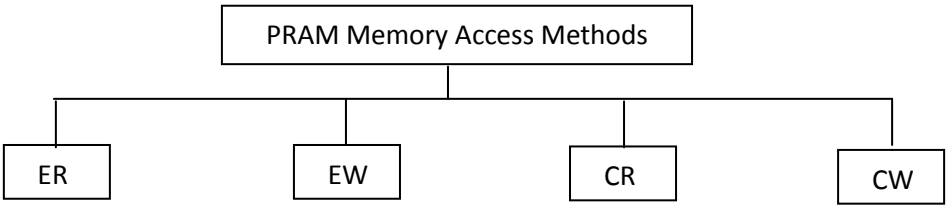
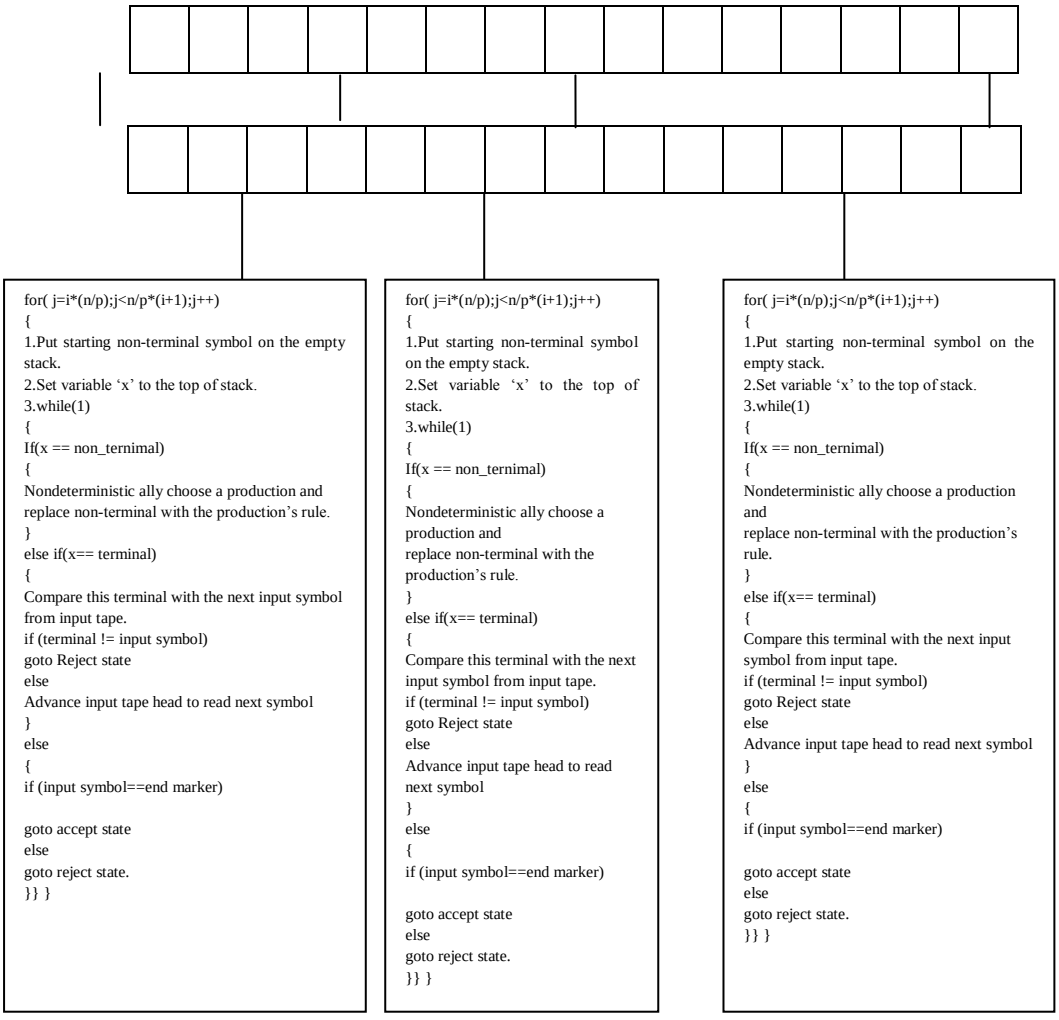


Fig 7: PRAM Memory Access Method



```
for( j=i*(n/p);j<n/p*(i+1);j++)
{
1.Put starting non-terminal symbol on the empty stack.
2.Set variable 'x' to the top of stack.
3.while(1)
{
If(x == non_ternimal)
{
Nondeterministic ally choose a production and replace non-terminal with the production's rule.
}
else if(x== terminal)
{
Compare this terminal with the next input symbol from input tape.
if (terminal != input symbol)
goto Reject state
else
Advance input tape head to read next symbol
}
else
{
if (input symbol==end marker)

goto accept state
else
goto reject state.
}} }
```

```
for( j=i*(n/p);j<n/p*(i+1);j++)
{
1.Put starting non-terminal symbol on the empty stack.
2.Set variable 'x' to the top of stack.
3.while(1)
{
If(x == non_ternimal)
{
Nondeterministic ally choose a production and
replace non-terminal with the production's rule.
}
else if(x== terminal)
{
Compare this terminal with the next input symbol from input tape.
if (terminal != input symbol)
goto Reject state
else
Advance input tape head to read next symbol
}
else
{
if (input symbol==end marker)

goto accept state
else
goto reject state.
}} }
```

```
for( j=i*(n/p);j<n/p*(i+1);j++)
{
1.Put starting non-terminal symbol on the empty stack.
2.Set variable 'x' to the top of stack.
3.while(1)
{
If(x == non_ternimal)
{
Nondeterministic ally choose a production and
replace non-terminal with the production's rule.
}
else if(x== terminal)
{
Compare this terminal with the next input symbol from input tape.
if (terminal != input symbol)
goto Reject state
else
Advance input tape head to read next symbol
}
else
{
if (input symbol==end marker)

goto accept state
else
goto reject state.
}} }
```

Fig 8: The PRAM –ELR Model

3. METHODLOGY

Figure 8 represents the overall view and steps to convert ELR-NPDA into PRAM-ELR. In which we perform following steps:

1. A string array of infinite size containing input sentences of English language.
2. Divide array into 'n' sizes at after every full stop (.).
3. Assign each divided part to different processor P_i.
4. And, then parallel run the algorithm of ELR-NPDA [1] into each processor.

3.1 Sequential Algorithm of ELR-NPDA[1]:

1. Put starting non-terminal symbol on the empty stack.

2. Set variable 'x' to the top of stack.

3. while(1)

```
 {
 if(x == non_ternimal) //x = top of
                          stack
 {
 Nondeterministic ally choose a
 production and replace non-terminal with
 the production's rule.
 }
 else if(x== terminal)
 {
 Compare this terminal with the next input
 symbol  from input tape.
 if (terminal != input symbol)
      goto Reject state
 else
      Advance input tape head to read next
      symbol
 }
 else      //codition if top of stack points to
            start  of stack
{
 if (input symbol==end marker) //input
                                  ends
     goto accept state
 else
     goto reject state.
} }
```

3.2 Algorithm of PRAM-ELR

1. Take English language sentences into an array of string of size 'n'.

2. Assume PRAM has 'i' processors (P0, P1, P2, ,,, , Pi)

3. Divide array into 'n' parts after every full stop (.).

4. For all Pi, do in parallel

```
 for( j=i*(n/p);j<n/p*(i+1);j++)
{
 1. Put starting non-terminal symbol on the
 empty stack.
 2. Set variable 'x' to the top of stack.
 3. while(1)
  {
  if(x == non_ternimal) // x = top of stack
  {
   Nondeterministic ally choose a
   production and replace non-terminal
   with the production's rule.
  }
  else if(x== terminal)
  {
  Compare this terminal with the next
  input symbol from input tape.
  if (terminal != input symbol)
       goto Reject state
  else
       Advance input tape head to read
   next symbol
 }
     else  //codition if top of stack points to
            start of stack
     {
 if (input symbol==end marker)
                        //input ends
     goto accept state
else
       goto reject state.
     }
     }          //end of while loop
     }          //end of for loop
```

4. RESULT

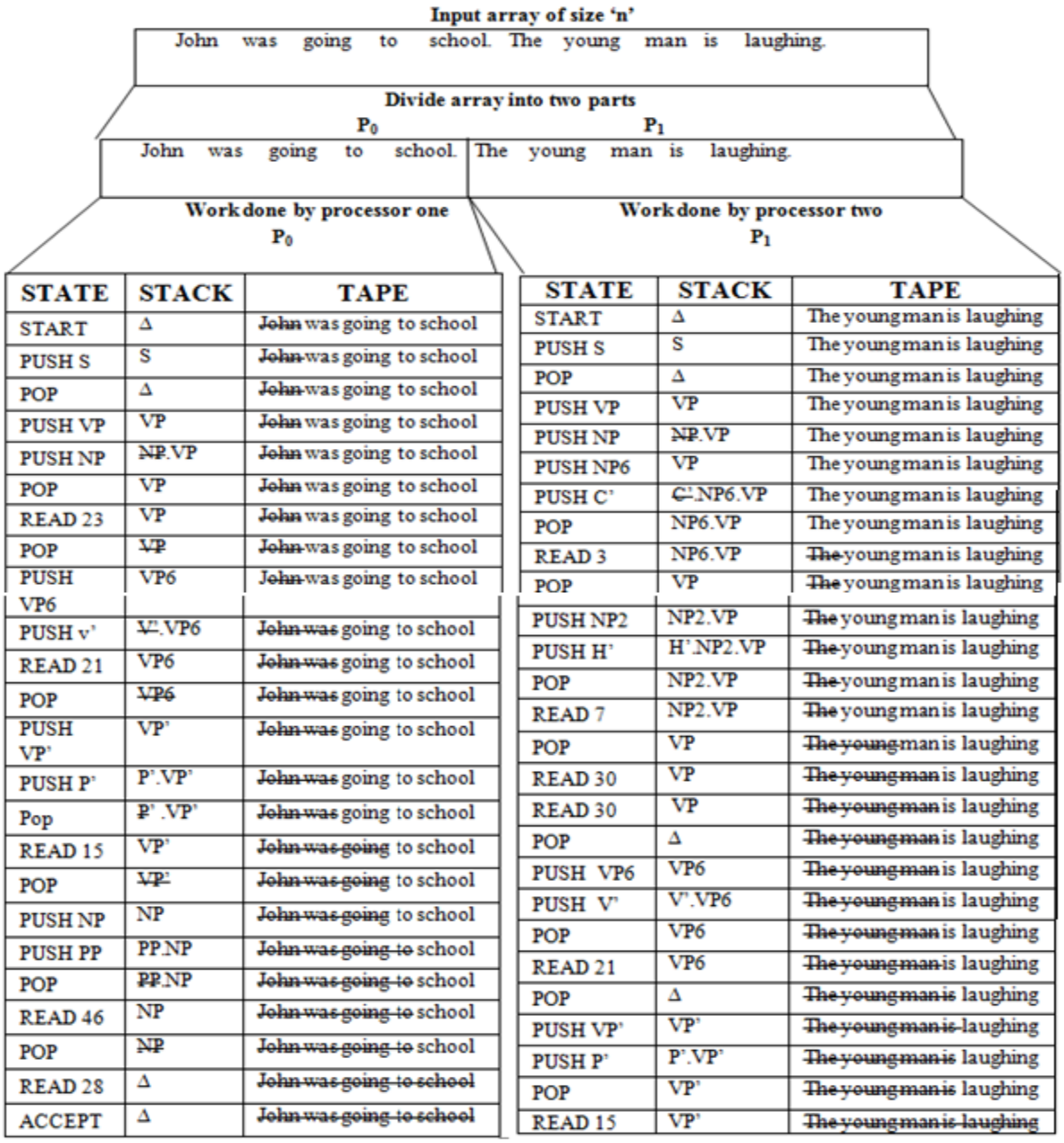
The above cite algorithm of PRAM-ELR can parse all legitimate sentences of English language but in very capable and effectual manner in a very high speed. The PRAM-ELR is the extended form of ELR-NPDA [1]. The ELR-NPDA is based on RAM model and a sequential algorithm having less competence to recognize large sentences. The extended version called PRAM-ELR having burly capability to over this problem in a very proficient approach. It can be view in figure 9. In this figure as we noticed that input array having

two sentences separated by full stop. So, PRAM-ELR used two processors in this situation or we can say that the array divided into two parts, one part assign to processor 1 having id 'P0' and second part of array will assign to the processor 2 having id 'P1'. Then finally the algorithm of ELR-NPDA works parallel and pares the sentences.

5. CONCLUSION AND FUTURE WORK

In this paper we conclude that every parallel random access machine (PRAM) is more effectual and proficient than random access machine (RAM) because PRAM increases efficiency, computing power, computational cost and speedier. PRAM has multiple processors that is why we can easily work on large data set in a very competent manner. The conclusion is shown in table 2.

In future it could be map with Graphical Processing Unit (GPU) and we will implement PRAM-ELR through CUDA-C or Open-CL. And further we will propose more PRAM model for other problems. Also many sequential algorithm will be mapped into PRAM.



STATE	STACK	TAPE
START	Δ	~~John~~ was going to school
PUSH S	S	~~John~~ was going to school
POP	Δ	~~John~~ was going to school
PUSH VP	VP	~~John~~ was going to school
PUSH NP	~~NP~~.VP	~~John~~ was going to school
POP	VP	~~John~~ was going to school
READ 23	VP	~~John~~ was going to school
POP	~~VP~~	~~John~~ was going to school
PUSH VP6	VP6	~~John~~ was going to school
PUSH v'	~~V'~~.VP6	~~John was~~ going to school
READ 21	VP6	~~John was~~ going to school
POP	~~VP6~~	~~John was~~ going to school
PUSH VP'	VP'	~~John was~~ going to school
PUSH P'	P'.VP'	~~John was~~ going to school
Pop	~~P'~~ .VP'	~~John was~~ going to school
READ 15	VP'	~~John was going~~ to school
POP	~~VP'~~	~~John was going~~ to school
PUSH NP	NP	~~John was going~~ to school
PUSH PP	PP.NP	~~John was going to~~ school
POP	~~PP~~.NP	~~John was going to~~ school
READ 46	NP	~~John was going to~~ school
POP	~~NP~~	~~John was going to~~ school
READ 28	Δ	~~John was going to school~~
ACCEPT	Δ	~~John was going to school~~

STATE	STACK	TAPE
START	Δ	The young man is laughing
PUSH S	S	The young man is laughing
POP	Δ	The young man is laughing
PUSH VP	VP	The young man is laughing
PUSH NP	~~NP~~.VP	The young man is laughing
PUSH NP6	VP	The young man is laughing
PUSH C'	~~C'~~.NP6.VP	The young man is laughing
POP	NP6.VP	The young man is laughing
READ 3	NP6.VP	~~The~~ young man is laughing
POP	VP	~~The~~ young man is laughing
PUSH NP2	NP2.VP	~~The~~ young man is laughing
PUSH H'	H'.NP2.VP	~~The~~ young man is laughing
POP	NP2.VP	~~The~~ young man is laughing
READ 7	NP2.VP	~~The~~ young man is laughing
POP	VP	~~The young~~ man is laughing
READ 30	VP	~~The young man~~ is laughing
READ 30	VP	~~The young man~~ is laughing
POP	Δ	~~The young man~~ is laughing
PUSH VP6	VP6	~~The young man~~ is laughing
PUSH V'	V'.VP6	~~The young man~~ is laughing
POP	VP6	~~The young man~~ is laughing
READ 21	VP6	~~The young man~~ is laughing
POP	Δ	~~The young man is~~ laughing
PUSH VP'	VP'	~~The young man is~~ laughing
PUSH P'	P'.VP'	~~The young man is~~ laughing
POP	VP'	~~The young man is~~ laughing
READ 15	VP'	~~The young man is laughing~~

Fig 9: Parsing In PRAM-ELR Model

6. REFERENCES

[1] Madiha Khurram Pasha, Dr. M. Sadiq Ali Khan, To Design a English Language Recognizer by Using Nondeterministic Pushdown Automata (ELR-NPDA), IJCA, Volume 105 – No. 1, November 2014, pp.36-43Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] James C. Chen, Cheng-Chun, Chia-Wen Chen, Kou-Huang Chen, Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm, Expert Systems with Applications, 2012, pp. 10016-10021

[3] Bilgesu AK, Erdem KOC, A guide for genetic algorithm based on parallel machine scheduling and flexible job-shop scheduling, SciVerse ScienceDirect, Procedia - Social and Behavioral Sciences 62 (2012) 817 – 823.

[4] Zhongni Zheng ,Rui Wang, Hai Zhong, Xuejie Zhang, An Approach for Cloud Resource Scheduling Based on Parallel Genetic Algorithm, IEEE, 2012, pp. 444-447

[5] Grey Ballard, James Demmel, Olga Holtz, Benjamin Lipshitz, Oded Schwartz, Communication-Optimal Parallel Algorithm for Strassen's Matrix Multiplication, ACM,June 25–27, 2012, pp. 193-204

[6] Jun Zhao , Quanli Liu , Wei Wang , Zhuoqun Wei, Peng Shi, A parallel immune algorithm for traveling salesman problem and its application on cold rolling scheduling, Information Sciences, 2011, pp. 1212-1223

[7] Qingpeng Niu, James Dinan, Qingda Lu, P. Sadayappan, PARDA: A Fast Parallel Reuse Distance Analysis Algorithm, IEEE 26th International Parallel and Distributed Processing Symposium, 2012, pp. 1284-129