ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X

International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST)
Vol. 2, Issue 9, September 2016

A Study and Simulation of Membrane Hierarchy with Dynamic Parallelism using CUDA

Amutha A L[1], Dorathi Jayaseeli J D[2], Dr. Malathi D[3]

Assistant Professor, CSE Department, SRM University, Chennai, India [1]

Assistant Professor, CSE Department, SRM University, Chennai, India [2]

Professor, CSE Department, SRM University, Chennai, India [3]

Abstract— **Membrane computing is on FastTrack in the field of parallelism. It is a bio-inspired computing that implements the working of the parallel model of membranes. This parallelism is not stable on CPU. Hence the computation is simulated using GPU computing and developing tool CUDA. NVIDIA CUDA has Dynamic parallelism with nesting the kernels and blocks that can be executed on parallel manner leveraging the concept on membrane computing. As concern with the hierarchy of the membrane, Dynamic parallelism plays important role. With help of SIMD architecture of CUDA simulation of parallelism of membrane computing in true sense can be implied.**

***Index Terms*— Bio-inspired computing, CUDA, Dynamic Parallelism and Membrane Computing.**

I. INTRODUCTION

A. Bio-Inspired Computing

Nature has always inspired human life, may it be social or artificial. There are many computationally hard problems that have been faced for the entire implementation of the technology. It inspires to analyze the working pattern of the biological techniques in the surrounding and estimating the processing model. The artificial electronic devices are manmade, and they have limitation over the time optimization and the components that build them. Computation at the molecular level has higher complexity level. They are considered to be virtual components for information process device. There had been challenges designed based on the specific enzymes on DNA. L. Adleman [6] had experimented in the laboratory for the solution of NP-complete problem with the DNA molecule manipulation. Various problems possibly can be solved with the proficient design of the algorithm taking in concern to the high cost of resolution and time/space complexity. Hence, it is necessary to bring out new model capable of the reduction in both the above parameters. Natural computing tries to follow the acts and the operations, simulation defined for nature- inspired computing haves unique interpretations when describing the emerged models. In the year of 1943, McCulloch and Pitts [30] presented an initial model artificial neural networks, motivated by the neural connection of the brain, also J. Holland proposed genetic algorithm inspired by the evolution of a living organisms from ages. Nature has

given inspiration for DNA computation, Ant-Colony Optimization, Termite nest, etc.

B. Membrane Computing

In late 90's George Paun [38] brought into insight a new computational approach from nature named as Membrane Computing. It unfolded the concept of parallelism and non-deterministic approach from a living cell; cell is organized and well-disciplined basic unit of a living organism. The highlighted feature of internal cell structure is that the Bio-system has several types of membrane. The outer membrane is separated from inner child membrane regardless of the functioning of the membrane in nature as in Fig. 1.1 these are divided into the compartments that do not restrict the communication among the membranes. The processes occurring within the cell are complex, and it is impossible to implement as a whole.

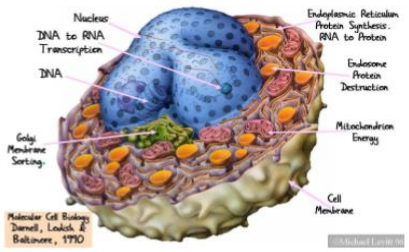

Fig.1.1 Membrane Structure and Hierarchy

The target is to simulate the task performed by the entity of living organism like replicating, energy production, metabolism and syntheses of the proteins. Computational device for membrane computing is defined as P system on the name of founder George Paun. There are several types of P system like Cell like P system, Tissue-like P System, Recognizer P System, P System with Active Membrane. In cell like P system membranes have structure inspired from the cell using graphical node structure. As shown in Fig. 1.2, membrane structure consists of membranes that arrange themselves in the hierarchical structure within the parent membrane. Each membrane has environmental region inside the membrane. The membrane that does not consist of inner membrane is called as an elementary membrane. A membrane structure is as that of the rooted tree. Membrane consists chemical substance within the compartment that corresponds

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X

International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST)
Vol. 2, Issue 9, September 2016

to them object of the membrane which can by represented as the set or the string of the objects termed as a multiset. Objects can evolve based on the reaction taking place. These reactions are abstracted as the rules. The aim behind the natural computing is to point the computationally hard problems to solve them in specific time. The rules for the reaction of the chemicals within the membranes are defined. Rules that are appropriate are applied non-deterministically and with maximum parallelization. Objects are evolved along with the steps of rules. Here every region or the compartment rules are applied at a time bringing up the maximal parallelization. The rules are applied until there is no object left for the match for the rule. This brings the terminating stage. The rule can be applied many times in the same step as we want with the presence of the number of copies. All computations start from an initial configuration i.e. the skin membrane and proceed as stated above only halting computations give a result, which is capsuled by the objects present in the output region in the halting

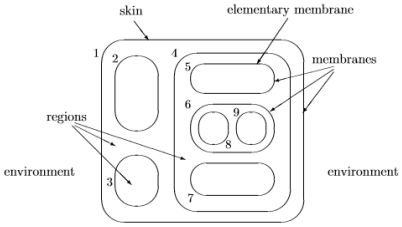

Fig. 1.2 Structure of membrane

Configuration based on the condition. Mitosis is a process of cell division which produces two daughter cells from a single parent cell. Daughter cells are twins to one another and identical to the original parent cell.

C. Applications of P System

Although the Membrane Computing is a biological inspiration, it contains a good theoretical model for distributed computing, where different calculations are operated in a hierarchical structure. For example, the hierarchy utilized to establish in networking connection, such as the Internet can be represented as that of membrane structure. Membrane Computing, according to the true motivation, were not intend to provide a comprehensive and perfect accurate model of the living cell without aiming to faithfully model biological facts in such a way to provide a modeling framework for the use of biologists, insteadto explore the computational nature with various quality of biological membranes. Indeed, most variants of membrane systems have been proved to be computationally complete, that is equivalent in power to Turing machines, and computationally efficient, such that it has the ability to solve computationally hard problems in polynomial time by managing time with space. After the firm development of theoretical model, this domain had been started to develop practically with the application on the framework of Biology system and Population Dynamics. A standard procedure followed here is the following: a) Construction of Membrane Computing model for a given phenomenon/process/population/system, b) Developing software application for simulation of the model, and c) then test are carried out to experimentally validate and test the model by using experimental data. Once the model is considered as validated, further process can be carried out virtually regarding the data models. Membrane systems have been recently used to model biological phenomena with computational systems biology framework representing the models of signal transduction [35], oscillatory systems [19], metapopulations [37], quorum sensing [39], metabolic systems [27, 12, 97], gene regulation control [40] and real ecosystems [10, 9, 15]. It is observed that the macroscopic, deterministic and unbroken continues approach followed by ordinary diff erential equations (ODEs) is questioned in cellular systems with a low number of standalone and heterogeneous structure Membrane Computing is a multiset processing unit using rewriting similar rules – is a complementary technique to systems for diff erential equations. Along with applications in biology, Membrane Computing was also considered helpful in other areas, such as computer graphics (models based on compartmentalized Lindenmayer systems proved to be more powerful and efficient instead of those using classic L-systems), cryptography, modeling in a unique uniform way parallel architecture, in linguistics, economics, etc.

This paper is organized as follows: Chapter 2 deals with the historical account of membrane computing, Chapter 3 explains about GPU and HPC, Chapter 4 explains parallel P system simulation with active membrane, Chapter 5 gives the result and analysis and finally we gave conclusion in Chapter 6.

II. A HISTORICAL ACCOUNT OF MEMBRANE COMPUTING

A. Overview

The use of P system provides three important quantities for scientific use as Educational purpose, to assist the design related to P system models and serving as more elaborates software tool. To simulate P system one shall have the model to be use a structure of the membrane and initial multiset and set of rules. Simulation core must implement the algorithm that is aimed to obtain one or more computations of the defined p system. It is important to design simulator with better efficiency. Simulation core shall be done step by step following the computing defined by the P systems, after that it is necessary to extract relevant information on the simulated computation and print it as an output.

B. Related Simulators

A review of the simulators developed in the first ten years of Membrane Computing is described in [16]. The change of those simulators is an expression of the evolution of the research itself. In this sense, it is usually to speak about two generations of simulators according to its finality. It is worth to mention that both generations overlap on time. One of the first simulators is Romero-Campero's simulator, which implements the multi-compartmental Gillespie algorithm [33] in SciLab and C. This simulator has been successfully used in addressing several real-world problems as the simulation of a signaling pathway associated with the Epidermal Growth Factor [34]; simulation of FAS-induced apoptosis [13]; modeling gene expression control [40]; or the first

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X

International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST)
Vol. 2, Issue 9, September 2016

computational model of Quorum Sensing [39, 44] in Vibrio Fischeri. Other relevant simulators in the second generation are: Cyto-Sim [42], a software that can simulate micro and macroscopic biological processes using arbitrary kinetic laws; Meta P-lab [11], a virtual laboratory which aims at assisting modelers both to understand the internal mechanisms of biological systems and to forecast, in silico, their response to external stimuli, environmental condition alterations or structural changes; and Infobiotics Workbench [7], an integrated software suite incorporating model specification, simulation, parameter optimization and model checking for Systems and Synthetic Biology.

C. PLingua Simulator

To simulate the feature of P system and their model's a simulator has developed named as PLingua. Here some matching libraries are used for parsing the input format. It is user-friendly and user need not to learn input format each time the uses different simulator The P-Lingua project also provides free software tools under GNU GPL license [1] for compilation, simulation and debug tasks. These tools has integration with Java library called as pLinguaCorer that will parse and handle P-Lingua input files and check possible programming errors in regarding Syntactic as well as Semantic. P-Lingua file can be export to another format in to gain the interoperability of the file with various software environments. Fig. 2.1 illustrates this approach to define the simulator input by using the P-Lingua framework. Such inputs are free of programming errors since the parser inside pLinguaCore has already checked them. As mentioned above, the pLinguaCore library includes several built-in simulators for the supported models. The current version of pLinguaCore is 3.0 and can download from the P-Lingua website [5]. Each version of PLingua and pLinguaCore adds new supported models and implements new simulation algorithms, the syntax definition of the language and more details is specified on the related papers and the website.

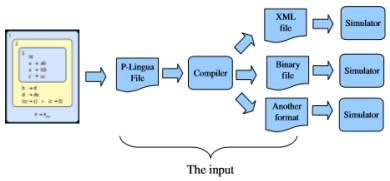

Fig. 2.1 PLingua simulator approach

There are the various versions of Plingua and its pLinguaCorelibrary for the models of the P-System. The versions description is as follows:

1. *P-Lingua 1.0* [17, 8]: this is initial software and is able to define as well as simulate tractable instances of P-systems with active membranes. The simulator only procreate one possible computation and hence the simulated P system must be flowing together to obtain a useful answer.

2. *P-Lingua 2.0* [21, 20, 5]: several cell-like P system models are incorporated, together with one or more built-in simulators for each: • Transition P systems. •Symport/antiport P systems. • Active membranes. • Active membranes with creation rules. • Stochastic P systems. • Probabilistic P systems.

3. *P-Lingua 2.1* [29, 5]: tissue-like P systems with division rules are also supported, including its built-in simulator and some fixed bugs.

4. *P-Lingua 3.0* [28, 5]: Population Dynamics P systems (PDP Systems) are also supported, and several built-in simulators for this model are added. It includes sequential implementations of the simulation algorithms presented in this thesis (DNDP and DCBA). Furthermore, some general bugs are fixed and the support of stochastic P systems is discontinued, encouraging the use of Infobiotics Workbench [7] in this respect. The PLingua is standard format to define P system to remove the error syntactically as the parser check in pLinguaCore. PLingua is an inspiration for the simulator to be built for the membrane computing. This simulator has implemented the sequential versions, using the functionality within the PLinguaCore software library. To evaluate parallelism in true sense same Simulators can be implemented via GPU with comparison of small occurrence of P system.

D. P System Representation

In [24], it is said that: "the next generation of simulators may be oriented to solve (at least partially) the problems of storage of information and massive parallelism by using parallel language programming or by using multiprocessor computers". There were some simulators developed but most of them was based and designed for sequential approach and sequential simulators have less performance efficiency. As the double parallelism gets serialized. Hence sequential simulation time increases because of the implementation of parallelism. Ecosystem model of P system needs high throughput simulation tool. Massively parallel nature of computation point out parallel technology where simulations can be made faster.

Formally,a transition Psystem (ofdegreem) is aconstruct of theform $\Pi = (O, C, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m, i_o)$, where: O is the (finite and non-empty) alphabet of objects,

1. C ⊂O is the set of catalysts,
2. μ is a membrane structure, consisting of m membranes, labelled with 1, 2,...,m; one says that the membrane structure, and hence the system, is of degree m,
3. $w_1, w_2, \ldots, w_m$ are strings over O representing the multi-sets of objects present in the regions 1,2, . . . , m of the membrane structure,
4. $R_1, R_2, \ldots, R_m$ are finite sets of evolution rules associated with the regions 1,2, . . . , m of the membrane structure,
5. I_o is either one of the labels1,2,...,m, and then the respective region is the output region of the system, oritis0, and then the result of a computation is collected in the environment of the system.

R is a finite set of developmental rules, of the following forms:

[h a → v] e , forh ∈H,e ∈{+,−, 0}, a ∈O, v∈O∗

(Object evolution rules, associated with membranes and depending on the label and the charge of the membranes, but

not directly involving the membranes, in the sense that the membranes are neither taking part in the application of these rules nor are they modified by them);

a[h]e1 → [h b] e2 , for h ∈H, e1 , e2∈{+, −, 0}, a, b ∈O

(In communication rules; an object is introduced in the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);

1. [h a] e1 → [h] e2 b, for h ∈H, e1 ,e2 ∈{+,−,0}, a, b ∈O

(Out communication rules; an object is sent out of the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);

2. [h a] e → b, forh ∈H,e ∈{+,−, 0}, a, b ∈O

(Dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);

3. [h a] e1 → [h b] e2 [hc]e3 , for h ∈H,e1 , e2, e3 ∈{+,−, 0}, a, b, c∈O(Division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations ;the object specified in the rule is replaced in the two new membranes by possibly new objects; there main objects are duplicated and may evolve in the same step)

E. Parallel P System Simulation

High Performance Computing can be one of the solutions for the Membrane Computing Parallel simulation. There are many approach followed for HPC simulation techniques. Some of them are as below:

Simulator Based on Cluster

Clusters are inter-connected computers through a local network. They work in coordination to execute parallel programs. As connection increases system bus acts as bottleneck. Several experiments are made to work on such platform. Also cluster had been first platform to implement parallelism for simulator. Ciobanu and Guo [14] simulate a restricted set of transition P systems using a Linux based cluster, using C++ and MPI library. Here each membrane is assigned to the each system, while each of them executes sequential code. Another approach was introduced by Syropoulos et al. [46]. This simulator works under Java, and makes use of RMI (Remote Method Invocation) to distribute the workload. Finally, a novel alternative has been initiated by DiezDolinski et al. [18]. They provide a highly scalable solution to the natural exponential growth of space made by P systems. For this purpose they use MapReduce algorithms over distributed environments. It has been categorized here as for clusters, but it has been conceived to work over grids and Internet. Moreover, a new branch of P-Lingua, called Distributed P-Lingua, was developed. Finally, the simulator is implemented in Java, using the freely Hadoop library. Although no performance analysis was provided, the first results suggest a promising research line.

Simulator Based on Microcontroller

A microcontroller is an integrated circuit containing a processor, memory and input/output units. Although they are limited by their set of instructions and number precision, they are very chip technology that can also be interconnected. Guti´errez et al. [23, 22] utilize this technology to provide an alternative platform. Their aim is to better balance flexibility and performance at a low cost. The work is based on one of the most used microcontrollers, that is, the PIC (Peripheral Interface Controller). The implementation is based on the previously proposed algorithms/architectures for improving the communication among units dealing with membranes [43, 8]. They categorize their solution as a "partially parallel evolution with partially parallel communication" [22]. Although no performance analysis is provided, their design will be useful for future work.

Cloud Based Simulation

Cloud computing [45] provides a service where performing calculus in virtual network. They implements a system similar to a cluster, but for a cheap rental price. In this regard, another novel approach to provide a high number of resources for simulating P systems creating exponential workspace, was initiated by Nabil et al. [31]. Authors use the SAT problem as a case study, and run an instance with 11 variables (requiring 211 membranes). Although no performance analysis is provided, the design serves as the base for future work.

Simulator based on GPU Computing

The GPU based simulation use the graphics card for processing. Accelerating GPU code for HPC and it is cheap compared to relative technologies. Not all the programs need to be accelerated by GPU. There is flexibility for the performance based on the CPU and GPU computation, One of the major support for such computation is NVIDIA CUDA with NVIDIA's GPU[26, 5] and OpenCL supported by AMD. Present simulation of the project is subjected to the GPU computing.

PMCGPU

This consist of several simulators that have been abstracted as project work for p system models.

1. PCUDA- First simulation of P system with CUDA having sequential and parallel version supporting P system with active membrane.
2. PCUDASAT- It is a part of PCUDA for Solving the SAT problem via P system.
3. TSPCUDASAT- it is a branch of the above PCUDASAT that deals with tissue P system with division rule solving SAT.
4. ABCD-GPU- It is research project to Population Dynamics P system. It is based on CUDA and OpenMP.

Simulation of Spiking Neural P Syatems

Here algorithm is applied using matrix representation of the model. Each computation is manipulated by multiprocessor of GPU.

Simulation of evolution-communication P systems with energy and without antiport rules.

Simulation is performed with linear algebra operation and execution performed on GPU.

Simulation of enzymatic numerical P systems

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X

International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST)
Vol. 2, Issue 9, September 2016

This Simulation is designed for robot controller and results were significant for the artificial intelligence.

III. PARALLEL P SYSTEM SIMULATION WITH ACTIVE MEMBRANES

For the validation of P system it is mandatory for the extraction of the data via simulation on the electronic devices. This will add benefit for researchers to compute and analyze as well as extract result from the model. Membrane computing until now have ben simulated sequentially with CPU but to utilize the maximum parallelism GPU can be utilized to fulfill the definition of parallelism by P system. Hence requirement of highly parallel computation technology is needed to accomplish the P system formal definition. New era of GPU generation is meant for heterogeneous parallelism with several threads in execution at a time.

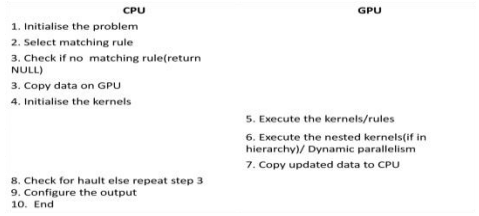

Fig. 3.1 Parallel simulation process work flow

Fig. 3.1 shows two stage of processing and work flow of the algorithm. It has two stages as: Selection and Execution. The selection stage consists in the search for rules to be executed in each membrane of a given configuration. The selected rules are executed at the execution stage, what finalizes the simulation of a computation step.

A. Algorithm for the Simulation

As the simulation is to be done in two parts, CPU and GPU memory shall be mapped accordingly and execution should be done. Fig. 4.2 shows the basic algorithm behind the work flow of the simulator. 1) Initialization is done with the definition of environment in the terms Lingua parser with the formal definition of P system. 2) That file is read and matching rules are found from the define rule set. These rules are stored as a list in applicable rule set. After the selection of rules to be executed are sorted in maximum parallel rule set. 3) All relevant data is to be copied from CPU to GPU using CUDA API library function. 5) Kernels are executed. 6) Dynamic parallelism is implied. 7) Resultant data is copied on GPU.

In order to avoid non-determinism somehow, the simulator assumes only confluent P systems. Thus, instead of working with the entire tree of possible computations,

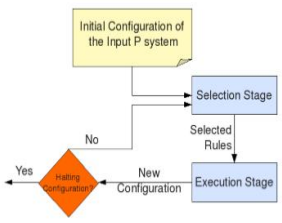

Fig. 3.2 Algorithm for the simulation of membrane computing

the simulator selects and simulates only one computation path, since all paths are guaranteed to give the same answer. Priorities among rules in the selection stage are:

1. Dissolution rules: they decrease the number of membranes (highest priority);
2. Evolution rules: they do not need any communication among membranes (avoids synchronization);
3. Send-out rules: they do need communication between the given membrane and its parent (adding one object to its parent);
4. Send-in rules: they do need communication between the given membrane and its parent (reserving one object from its parent and adding the object to itself);
5. Division rules: they increase the number of membranes (lowest priority).

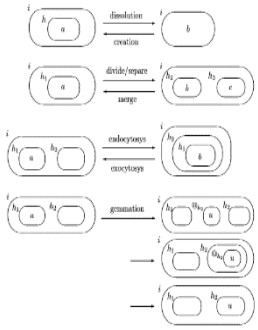

Fig. 3.3 Diagrammatic Representation of rules

Fig. 3.3 shows the diagrammatic representation of rules. Finally, note that this two-staged algorithm allows keeping coherence in the simulation. If we perform selection and execution of rules, one by one, it would be difficult to ensure the semantic constraints of the system. Moreover, the selected and executed rules in a step of the simulator may not correspond to the rules applied in a computing step of the theoretical model. An alternative solution might be to take two copies of the configuration, one to be updated with the right-hand sides of the rules, and another to select rules (subtracting the left-hand side of rules). As this involves a bigger use of memory, our simulator uses the two stages, and a temporary data structure to store information about the selection of rules.

B. PLingua Input File

The input of the simulator (the P system with active membranes to simulate) is given by a binary file. It is a file whose information is encoded in Bytes and bits (not understandable by humans like plain text), which is suitable for compressing data. This binary file contains all the information of the P system (alphabet, labels,

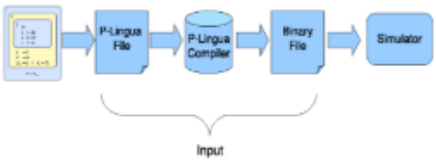

Fig. 3.4 PLingua input for simulator

rules, etc.) which is the input of our simulator. The format is depicted in Section C. pLinguaCore 2.0 [21] is able to

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X

International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST)
Vol. 2, Issue 9, September 2016

translate a P system written in P-Lingua language into a binary file. We therefore use a pipeline of applications to simulate P systems, as shown in Fig. 3.4. First, we define our P system into P-Lingua. PLinguaCore translate it to a binary file, which is used as the input of the simulator. The output is a plain text generated with a format similar to the provided in pLinguaCore.

Some sample of PLingua input file is shown in Fig. 3.5 and 3.6:

Fig. 3.5 Sample 1 for PLingua file

Fig. 3.6 Sample 2 for PLingua file

C. Binary Format for Input

Fig. 3.7 shows binary format of the input file for our simulator. It specifies the number of bytes used for each required element, and comments with explanations are given by '#'. Note that it is thought for compressing the information of the P system. The format is structured in 3 main parts: 1) A header storing general information about the file (version, etc.). 2) The P system general information: alphabet, membrane structure(plus initial charges)and initial multisets. 3) The rules of the P system.

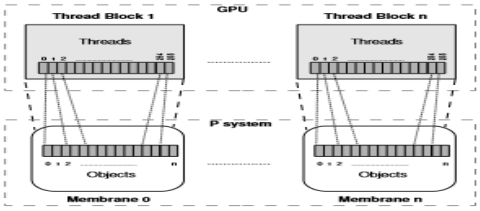

Fig. 3.7 Binary format for input file

D. Simulation on CUDA

In the CUDA programming model, our main effort is dividing the required work into processing pieces, which have to be processed by TB thread blocks of T threads each. Using a thread block size of T=256, it is empirically determined to obtain the overall best performance on the Tesla C1060 [41]. each thread block can be considered independent to the other, and it is at this level at which internal communication (among threads) is cheap using explicit barriers to synchronize, and external communication (among blocks) becomes expensive, since global synchronization can only be achieved by the barrier implicit between successive kernel calls. The need of global synchronization in our designs requires successive kernel calls even to the same kernel. Fig. 3.8 shows the overall design of the simulator that we have implemented on the GPU. We distribute the thread blocks and threads as follows. Each membrane of the simulated P system is attributed to each thread block. In this way, we identify the parallelism between membranes by

Fig. 3.8 Design of Simulator

using the parallelism between thread blocks. Precautions must be taken with this design decision. Membranes can communicate accordingly to the hierarchical tree structure, while thread blocks are all independent. Communication through send-out and dissolution rules (down-up direction) is controlled by globally synchronizing the selection and execution stages. This is implemented by using different kernels. However, send-in rules (up-down direction in the tree) are more complicated to control. In this case, different membranes can compete for single objects. The sequential simulator controls this issue by looping the tree from the top to the bottom. However, the parallel simulator has to run all the membranes in parallel. Therefore, for the sake of simplicity, the parallel simulator can handle only two levels of membrane hierarchy: the skin (controlled by the host) and the rest of elementary membranes (controlled by the thread blocks in device). This is the tree structure we can find in the literature for the majority of solutions based on P systems with active membranes (note that division rules enlarge the tree widthwise) [36]. Furthermore, each individual thread is assigned to each object within a membrane (corresponding to its thread block). It is responsible for identifying the rules that can be executed using the corresponding object. That is, rules that have that object in their left-hand sides. Since all blocks must have the same number of threads, and each membrane can contain a different multiset of objects in every time step, we identify as common for all membranes the whole alphabet. Note that threads can work with many objects that do not really exist in the membrane, as all the alphabet of objects is usually not present within a membrane at a given instant. In fact, the simulator assigns multiple objects to the same thread for not restricting the number of objects in the alphabet. However, the number of objects in the alphabet must be divisible by a number smaller than 512 (the maximum number of threads per thread block), in order to equally distribute the objects among the threads. The simulator contains five kernels to implement the selection and execution stages. The first kernel implements the selection stage and also the execution stage for evolution rules. The other four kernels implement the other execution rules (dissolution, division, send-out and send-in rules). All the kernels follow this basic design. The selection kernel starts with the selection stage. After the selection stage, we also execute in this kernel the evolution rules. These rules are executed inside this kernel for three main reasons: the evolution rules do not imply communication (and therefore, synchronization) among membranes; they are executed in a maximal way, and this

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X

International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST)
Vol. 2, Issue 9, September 2016

decision allows us to use less global memory because it is not necessary to store the selected evolution rules for the execution stage. The rest of the rules to be applied are executed in four different kernels, one kernel per each kind of rule (dissolution, division, send-out, send-in)

E. Dynamic Parallelism

Dynamic parallelism is nested parallelism executed by CUDA. Parent kernel will be fired from the CPU or the host that will launch child kernel via parent kernel on GPU/device itself. Thus the depth is travelled by GPU kernel itself. This quality of CUDA programming eliminates the recursion and irregular structure of the loop. This feature provides solution to the hierarchical algorithms where data partitions will be decided by the main kernel launched on GPU via CPU. Hence the problem structure with higher complexity can be solved with less effort. Fig. 3.5 shows the nested structure of the parent and child kernel. All the child kernels will follow parallelism and will run at a time.

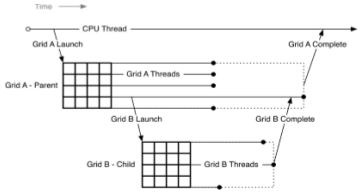

Fig. 3.9 Nested parallelism of Dynamic parallelism

Child Grid will be inherited from the parent Grid and shared memory will be configured. Two synchronizations will be followed: Implicit –after the parent grid's execution. And second is Explicit – for all the stream data i.e. for all threads. CUDA 5.0+ will support for the compilation of such hierarchical parallelism.

IV. RESULTS AND ANALYSIS

PLingua file to binary file conversion is as follows via PLinguaCore.

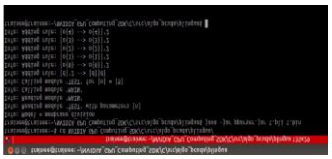

Fig. 4.1 Binary input files 1

As shown in Fig. 4.1, basically text file with .pli extension is converted to the binary file via pparser.

Fig. 4.2 Make executable file

After the .bin file is generated, make shall be executed for the generation of executable file to develop and run the simulator. Commands shall be as shown in the Fig. 4.2. Fig. 4.3 shows the help commands.

Fig. 4.3 shows the help commands.

Fig. 4.4 Running Sequentially:

Command for Fig. 4.4 is as follows:

trainee@trainee:~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release$./pcuda-i/home/trainee/NVIDIA_GPU_Computing_SDK/C/src/pcuda/plingua/sat_5_qu een_v2.bin -s -l 0

Fig. 4.5 sample output 1

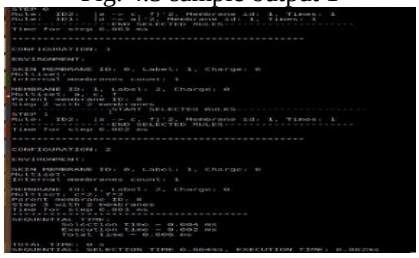

Fig. 4.5.1 Sample output 1 continuation.

Above Fig. 4.3 and Fig. 4.4 shows the help and supported commands to run the .bin file and extract the data. Command are as follows:

trainee@trainee:~/NVIDIA_GPU_Computing_SDK/C/src/bin/linux/release$./algo_pcuda–i/home/trainee/NVIDIA_GPU_Computing_SDK/C/src/algo_pcuda/plingua/bin\ files/a4.bin it 2 –v3 –o obj –b fac –m memb – f –p3

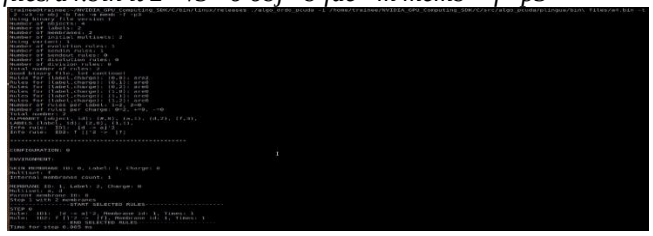

Fig. 4.6 sample output 2

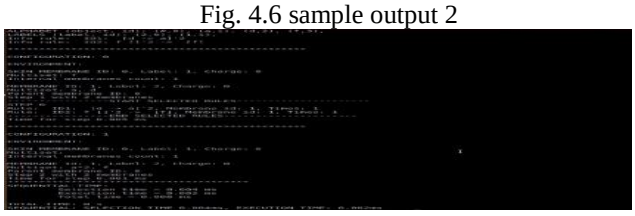

Fig. 4.6.1 sample output 2 continue

Fig. 4.5 and Fig 4.5.1 shows sample output along with the rules as the extraction. Fig. 4.6 and Fig. 4.6.1 shows sample output along with the rules as the extraction Command followed here is as follows:

trainee@trainee:~/NVIDIA_GPU_Computing_SDK/C/src/bin/linux/release$./algo_pcuda–i /home/trainee/NVIDIA_GPU_Computing_SDK/C/src/algo_pcuda/plingua/bin\ files/a3.bin it 2 –v3 –o obj –b fac –m memb–f –p3

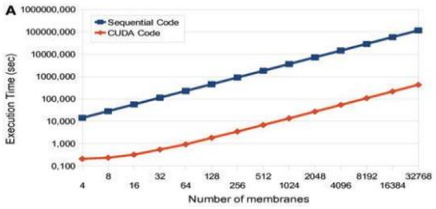

Fig. 4.7 Membrane generation analysis

Above Fig. 4.7 shows the membrane generation comparison via sequential code and parallel code using CUDA

V. CONCLUSION

This simulator is primary version extension to PCUDA simulator with additional feature of dynamic parallelism. Dynamic parallelism is added to implement the hierarchy of the membrane structure, so that two level hierarchies can be extended. This was an experimental approach to overcome the instability of PLingua simulator in terms of parallelism and membrane level hierarchy of PCUDA with the dynamic parallelism.

For the future work, simulation of 3 level hierarchies can be done. Also reduction in the time and space complexity by assigning the objects to the threads, specific model can be implemented based on the requirements. Multiplicity can be increased with the double configuration on the multiset. PParser can be modeled such a way to accept the multiplicity.

REFERENCES

[1] GNU GPL license. http://www.gnu.org/licenses/gpl. html.

[2] NVIDIA CUDA C Programming Guide 4.2. http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf.

[3] P-Lingua release 1.0 website. http://www. gcn.us.es /plingua.

[4] The GPGPU organization. http://www.gpgpu.org.

[5] The P-Lingua web page. http://www.p-lingua.org

[6] L. M. Adleman. Molecular computation of solutions to combinatorial problems. Science, 266(11):1021–1024, 1994

[7] J. Blakes, J. Twycross, F. J. Romero-Campero, and N. Krasnogor. The infobiotics workbench: an integrated in silicomodelling platform for systems and synthetic biology. Bioinformatics, 27(23):3323–3324, 2011.

[8] G. Bravo, L. Fern´andez, F. Arroyo, and M. A. Pen˜a. Hierarchical masterslave architecture for membrane systems implementation. In Thirteenth International Symposium on Artificial Life and Robotics 2008, AROB 13th, 2008.

[9] M. Cardona, M. A. Colomer, A. Margalida, I. P´erez-Hurtado, M. J. P´erez-Jim´enez, and D. Sanuy. A P system based model of an ecosystem of some scavenger birds. In G. Pa˘un, M. P´erez-Jim´enez, A. Riscos-Nu´n˜ez, G. Rozenberg, and A. Salomaa, editors, Membrane Computing, volume 5957 of Lecture Notes in Computer Science, pages 182–195. Springer Berlin Heidelberg, 2010

[10] M. Cardona, M. A. Colomer, M. J. P´erez-Jim´enez, D. Sanuy, and A. Margalida. Modeling ecosystems using P systems: the bearded vulture, a case study. In D. Corne, P. Frisco, G. Pa˘un, G. Rozenberg, and A. Salomaa, editors, Membrane Computing, volume 5391 of Lecture Notes in Computer Science, pages 137–156. Springer Berlin Heidelberg, 2009.

[11] A. Castellini and V. Manca. Metaplab: A computational framework for metabolic p systems. In Membrane Computing, volume 5391 of Lecture Notes in Computer Science, pages 157–168. Springer Berlin Heidelberg, 2009.

[12] A. Castellini, V. Manca, and Y. Suzuki. Metabolic P system flux regulation by artificial neural networks. In G. Pa˘un, M. P´erez-Jim´enez, A. Riscos-Nu´n˜ez, G. Rozenberg, and A. Salomaa, editors, Membrane Computing, volume 5957 of Lecture Notes in Computer Science, pages 196–209. Springer Berlin Heidelberg, 2010.

[13] S. Cheruku, A. Pa˘un, F. J. Romero-Campero, M. J. P´erez-Jim´enez, and O. H. Ibarra. Simulating FAS-induced apoptosis by using P systems. Progress in Natural Science, 17:424–431, 2007.

[14] G. Ciobanu and G. Wenyuan. A P system running on a cluster of computers. In Lecture Notes in Computer Science, WMC 2003, pages 123–150. Springer-Verlag, 2004.

[15] M. Colomer, A. Margalida, D. Sanuy, and M. J. P´erez-Jim´enez. A bioinspired computing model as a new tool for modeling ecosystems: The avian scavengers as a case study. Ecological Modelling, 222(1):33–47, 2011.

[16] D. D´ıaz-Pernil, C. Graciani-D´ıaz, M. A. Guti´errez-Naranjo, I. P´erezHurtado, and M. J. P´erez-Jim´enez. Software for P systems, chapter 17, pages 437–454. Oxford University Press, Oxford (U.K.), 2010.

[17] D. D´ıaz-Pernil, I. P´erez-Hurtado, M. J. P´erez-Jim´enez, and A. RiscosNu´n˜ez. A P-Lingua programming environment for Membrane Computing. In D. Corne, P. Frisco, G. Pa˘un, G. Rozenberg, and A. Salomaa, editors, Membrane Computing, volume 5391 of Lecture Notes in Computer Science, pages 187–203. Springer Berlin Heidelberg, 2009.

[18] L. DiezDolinski, R. Nu´n˜ezHerva´s, M. Cruz Echeand´ıa, and A. Ortega. Distributed simulation of P systems by means of Map-Reduce: first steps with Hadoop and P-Lingua. In J. Cabestany, I. Rojas, and G. Joya, editors, Advances in Computational Intelligence, volume 6691 of Lecture Notes in Computer Science, pages 457–464. Springer Berlin Heidelberg, 2011.

[19] F. Fontana, L. Bianco, and V. Manca. P systems and the modeling of biochemical oscillations. In R. Freund, G. Pa˘un, G. Rozenberg, and A. Salomaa, editors, Membrane Computing, volume 3850 of Lecture Notes in Computer Science, pages 199–208. Springer Berlin Heidelberg, 2006

[20] M. Garc´ıa-Quismondo, R. Guti´errez-Escudero, M. A. Mart´ınez-delAmor, E. Orejuela-Pinedo, and I. P´erez-Hurtado. P-Lingua 2.0: a software framework for cell-like P systems. International Journal of Computers, Communications and Control, 4(3):234–243, 2009.

[21] M. Garc´ıa-Quismondo, R. Guti´errez-Escudero, I. P´erez-Hurtado, M. J. P´erez-Jim´enez, and A. Riscos-Nu´n˜ez. An overview of P-Lingua 2.0. In G. Pa˘un, M. J. P´erez-Jim´enez, A. Riscos-Nu´n˜ez, G. Rozenberg, and A. Salomaa, editors, Membrane Computing, volume 5957 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010.

[22] A. Guti´errez and S. Alonso. P systems: from theory to implementation, chapter 17, pages 205–226. Concept Press Ltd, Hong Kong, 2010.

[23] A. Guti´errez, L. Fern´andez, F. Arroyo, and S. Alonso. Hardware and software architecture for implementing membrane systems: A case of study to transition P systems. In M. Garzon and H. Yan, editors, DNA Computing,

volume 4848 of Lecture Notes in Computer Science, pages 211–220. Springer Berlin Heidelberg, 2008

[24] M. A. Guti´errez-Naranjo, M. J. P´erez-Jim´enez, and A. Riscos-Nu´n˜ez. Available membrane computing software. In G. Ciobanu, G. Pa˘un, and M. J. P´erez-Jim´enez, editors, Applications of Membrane Computing, Natural Computing Series, pages 411–436. Springer Berlin Heidelberg, 2006.

[25] B. W. Kernighan and D. Ritchie. The C programming language. Prentice Hall, 2nd edition, 1988.

[26] D. B. Kirk and W. W. Hwu. Programming massively parallel processors: a hands-on approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2010.

[27] V. Manca. Fundamentals of Metabolic P Systems, chapter 19, pages 475–498. Oxford University Press, Oxford (U.K.), 2010.

[28] M. Mart´ınez-del-Amor, I. P´erez-Hurtado, M. Garc´ıa-Quismondo, L.F. Mac´ıas-Ramos, L. Valencia-Cabrera, ´A. Romero-Jim´enez, C. GracianiD´ıaz, A. Riscos-Nu´n˜ez, M.A. Colomer, and M.J. P´erez-Jim´enez. DCBA: Simulating population dynamics P systems with proportional object distribution. In Proceedings of the 13th International Conference on Membrane Computing (CMC13), pages 291–310, Budapest, Hungary, August 2012.

[29] M. A. Mart´ınez-del-Amor, I. P´erez-Hurtado, M. J. P´erez-Jim´enez, and A. Riscos-Nu´n˜ez. A p-lingua based simulator for tissue p systems. The Journal of Logic and Algebraic Programming, 79(6):374–382, 2010.

[30] M. Minsky and S. Papert. Perceptions. MIT Press, 1970.

[31] E. Nabil, H. Hameed, and A. Badr. A cloud based P systems algorithm. International Journal of Computer Applications, 54(13):26–31, 2012.

[32] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. GPU Computing. Proceedings of the IEEE, 96(5):879–899, 2008.

[33] G. P˘aun and F. J. Romero-Campero. Membrane Computing as a modeling framework. Cellular systems case studies. In M. Bernardo, P. Degano, and G. Zavattaro, editors, Formal Methods for Computational Systems Biology, volume 5016 of Lecture Notes in Computer Science, pages 168– 214. Springer Berlin Heidelberg, 2008.

[34] M. P´erez-Jim´enez and F. Romero-Campero. A study of the robustness of the EGFR signalling cascade using continuous membrane systems. In Membrane Computing, volume 3561 of Lecture Notes in Computer Science, pages 268–278. Springer Berlin Heidelberg, 2005.

[35] M. P´erez-Jim´enez and F. Romero-Campero. P systems, a new computational modelling tool for systems biology. In C. Priami and G. Plotkin, editors, Transactions on Computational Systems Biology VI, volume 4220 of Lecture Notes in Computer Science, pages 176–197. Springer Berlin Heidelberg, 2006.

[36] M. J. P´erez-Jim´enez, ´A. Romero-Jim´enez, and F. Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. Natural Computing, 2(3):265–285, 2003.

[37] D. Pescini, D. Besozzi, G. Mauri, and C. Zandron. Dynamical probabilistic p systems. International Journal of Foundations of Computer Science, 17(1):183–204, 2006.

[38] G. Pa˘un. Computing with membranes. Journal of Computer and System Sciences, 61:108–143; Turku Center for CS–TUCS Report No 208 (1998), 2000.

[39] F. J. Romero-Campero and M. J. P´erez-Jim´enez. A model of the quorum sensing system in vibrio fischeri using P systems. Artificial Life, 14(1):95– 109, 2008.

[40] F. J. Romero-Campero and M. J. P´erez-Jim´enez. Modelling gene expression control using p systems: The lac operon, a case study. Biosystems, 91(3):438–457, 2008.

[41] N. Satish, M. Harris, and M. Garland. Designing efficient sorting algorithms for manycore GPUs. In IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, pages 1– 10. IEEE Computer Society, May 2009.

[42] S. Sedwards and T. Mazza. Cyto-sim: a formal language model and stochastic simulator of membrane-enclosed biochemical processes. Bioinformatics Applications Note, 23(20):2800–2802, 2007.

[43] J. A. Tejedor, L. Ferna´ndez, F. Arroyo, and G. Bravo. An architecture for attacking the communication bottleneck in P systems. Artificial Life and Robotics, 12(1-2):236–240, 2008.

[44] G. Terrazas, N. Krasnogor, M. Gheorghe, F. Bernardini, S. Diggle, and M. Ca´mara. An environment aware P system model of quorum sensing. In S. Cooper, B. Lo¨we, and L. Torenvliet, editors, New Computational Paradigms, volume 3526 of Lecture Notes in Computer Science, pages 479–485. Springer Berlin Heidelberg, 2005.

[45] W. Voorsluys, J. Broberg, and R. Buyya. Introduction to Cloud Computing, pages 1–41. Wiley press, 2011.

[46] A. Syropoulos, L. Mamatas, P. C. Allilomes, and K. T. Sotiriades. A distributed simulation of transition P systems. In Workshop on Membrane Computing, pages 357–368, 2003.