The Thirteenth International Symposium on Artificial Life and Robotics 2008(AROB 13th '08),
B-Con Plaza, Beppu, Oita, Japan, January 31-February 2, 2008

Hierarchical Master-Slave Architecture for Membrane Systems Implementation

Ginés Bravo, Luis Fernández, Fernando Arroyo, Miguel A. Peña

Natural Computing Group - Universidad Politécnica de Madrid – Madrid
(Tel : +34-91-336-7901; Fax : +34-91-336-7893)
({gines,setillo,farroyo,mapc}@eui.upm.es)

Abstract: Membrane systems are computational equivalent to Turing machines. However, their distributed and massively parallel nature obtains polynomial solutions opposite to traditional non-polynomial ones. At this point, it is very important to develop dedicated hardware and software implementations exploiting those two membrane systems features.

Nowadays, it is possible to find out at least three different viable architectures that implements P Systems in a distributed cluster of processors: P2P, Hierarchical P2P and Master-Slave. These proposed architectures have reached a certain compromise between the massively parallelism character of the system and evolution step times. In particular, they have focused in the second phase of an evolution step and have obtained good results in the throughput in the external communication among processors and parallelization levels of the system. We propose a variation of the Master-Slave as target of our study in order to improve it because the times for the communication phase get linearly slower as the system grows in number of membranes.

In this paper, we suggest an alternative software architecture where several membranes are located in each processor, proxys are used to communicate with membranes located in different processors and, above all, interconnection among processors are made in a hierarchical master-slave shape. Also, a communication policy is stated to prevent collision and network congestion. All this yields an improvement in the system parallelism implementation than in our target study because it gets an increment of the parallelism of the external communication among processors. As a consequence of it, the number of processors grows in such a way that is notorious the increment of the parallelism in the application of the evolution rules and the internal communications. Also, as a result, time for every evolution step is improved and this new hierarchical master-slave architecture is close to the massively parallel nature of the P System model.

Keywords: Architecture, Hierarchy, Bottleneck, Communication, Master-Slave, PSystems.

I. INTRODUCTION

Possibilities offered by Natural Computation and, specifically P-Systems, for solving NP-problems, have made researchers concentrate their work towards HW and SW implementations of this new computational model. Transition P systems were introduced by Păun in 1998 [1]. They were inspired by "basic features of biological membranes". One membrane defines a region where there are a series of chemical components (multisets) that are able to go through chemical reactions (evolution rules) to produce other elements. Inside the region delimited by a membrane can be placed other membranes defining a complex hierarchical structure that can be represented as a tree. Generated products by chemical reactions can remain in the same region or can go to another region crossing a membrane. As a result of a reaction, one membrane can be dissolved (its chemical elements are transferred to the container membrane) or can be inhibited (the membrane becomes impermeable and not let objects to pass through).

Membrane systems are dynamics because chemical reactions produce elements that go through membranes to travel to other regions and produce new reactions. This dynamic behaviour is possible to be sequenced in a series of evolution steps between one system configuration to another. These system configurations are determined by the membrane structure and multisets present inside membranes. In the formal Transition P systems model can be distinguished two phases in each evolution step: rules application and communication. In application rules phase, rules of a membrane are applied in parallel to the membrane multiset inside of it. Once application rules phase is finished, then it begins communication phase, where those generated multisets travel through membranes towards their destination in case they are another region. These systems carry out computations through transitions between two consecutive configurations, what turn them into a computational model with the same capabilities as Turing machines.

Power of this model lies in the fact that the evolution process is massively parallel in application rules phases as well as in communication phase. The challenge for researchers is to achieve hardware and/or software implementations of P systems respecting the massively parallelism in both phases. The goal of this work is to design a new hierarchical communication architecture that increases the simultaneity

(parallelization) levels between application and communication phases that present other proposed architectures.

This paper is structured in the following way: in first place, related works are enumerated reviewing the proposed architectures; next a hierarchical architecture communication model is introduced stating detailed analysis of the model; afterward, a comparative analysis with other architecture is presented and, finally, the conclusions obtained are presented.

II. RELATED WORKS

Syropoulos [2] and Ciobanu [3] present distributed P systems implementations. They use respectively, the Java Remote Method Invocation (RMI) and the Message Passing Interface (MPI) over a PC cluster's Ethernet network. However, these authors don't make a detailed analysis about importance of time spent in communication phase respect total time of P system evolution, although Ciobanu [3] declares that "the response time of the program has been acceptable. There are however executions that could take a rather long time due to unexpected network congestion".

In reply to the problem in the communication phase, there are three viable different works that suggest distributed architecture of processors each:

- "Distributed architecture with both application and communication phases partially parallel", [4], (in the rest of the paper known as "P2P")
- "Hierarchical Architecture with parallel communication", [5] (in the rest of the paper known as "hierarchical P2P")
- "Master-Slave Distributed Architecture", [6]

Among them, we choose Master-Slave model [6] as the target of our study because **parallelize application and communications phases** (neither P2P nor hierarchical P2P does) and also, because, unlike P2P and hierarchical P2P, Master-Slave has **no restriction in the topology** of the P System when it comes to distribute membranes over processors, in other words, is independent from the topology of the P system.

1. Master-Slave Basements

Our target architecture has the next basements:

a. Membranes distribution. At each processor, K membranes are allocated that will evolve, at worst, sequentially. Where,

$$K = \frac{M}{P} \quad , K \geq 1 \qquad (1)$$

and M is the total number of membranes of the P system and P is the number of processors of the distributed architecture. The physical interconnection of processors is made through a shared communication line. In this scenario, there are two sorts of communications:

- Internal communications that are the ones that occur between membranes allocated at the same processor, and whose communication times is negligible because they are carried out using shared memory techniques.
- External communications that are those that occur between different processors because the membranes that needs to communicate are in different processors.

The benefit obtained is that the number of the external communications decreases.

b. Proxy for processor. Membranes that are in different processors do not communicate directly. They do by the means of proxys hosted at their respective processor. Proxys are used to communicate among processors. A proxy assumes communications among membranes of one processor towards the proxy of another one. In the same way, when information from other proxys is received, it is redistributed to the membranes of the processor.

The benefit of using proxys in the communication among membranes instead of direct communication occurs because the communication protocols penalize the transmission of small packets due to protocol overhead. So, communicate N messages of L length is slower than one message of $(S * L)$ length.

c. Order in Communications. There is a communication policy: each of the slave processors will receive a communication order number so that communications in the two directions are orderly. The master distributes multisets to different slaves. The slaves begin to apply evolution rules as soon as they receive them. When they are finished, they have to communicate their results to the master. However, to prevent collisions with other slaves, they wait their turn to do so.

The benefit is the absence of packet collisions and congestion of the common communication medium.

d. Flat communications architecture: slave processors do not communicate among themselves, but only with the master; for its part, the master processor communicates only with slave processors. The physical interconnection between master and slave is through a common communication line.

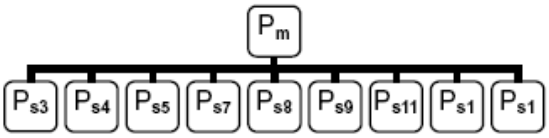

Fig. 1. Layout of Master-Slave Architecture

2. Master-Slave Analysis

The formal analysis of this distributed architecture leads to the following conclusions:

- In this model, given that each slave processor has K membranes, the total time of the rule application phase is $K \cdot T_{apl}$, minimum time for an evolution step (T_{mtn}) is determined by the formula:

$$T_{\min} = 2\sqrt{2\,M\,T_{apl}\,T_{com}} - 2T_{com} \qquad (2)$$

  where, T_{apl} is the maximum time used by the slowest membrane in applying its rules, and T_{com} is the maximum time used by the slowest membrane for communication

- The number of processors (P_{opt}) that leads to the minimum time is:

$$P_{opt} = \sqrt{\frac{T_{apl}\,M}{2T_{com}}} \qquad (3)$$

- For its part, the optimal number of membranes (K_{opt}) with regards to the optimal number of slave processors (P_{opt}) which gives a minimal time (T_{mtn}) is:

$$K_{opt} = \sqrt{\frac{M\,T_{com}}{T_{apl}}} \qquad (4)$$

III. HIERARCHICAL MASTER-SLAVE ARCHITECTURE

Previous model parallelizes in several points:

- parallelizes the application of rules in some processors (and even over P_{opt} processors)
- parallelizes the internal communications among membranes in the same processor, and
- foremost, parallelizes both phases: the application rule phase among some processors with the external communication phase in others

On the other hand, in that architecture model, **external communications**, necessaries for the communication among membranes allocated at different processors, **are sequential**. Even more, in a scenario with a lot of slaves, slaves will spend so much time waiting for its turn to transmit (upward) results to master because media is common (even with Ethernet switches, the fact is that last arrival point, i.e., the master, is unique). In short, the phase of results communication in the flat Master-Slave model is sequential, so times of this phase are linear.

The new architecture suggested here consists in distribute the processors in a hierarchical way, specifically, in a balanced tree of L levels in depth and A processors in amplitude. The introduction of intermediate nodes/levels permit that communications can group, that is, parallelize. For instance, Figure 2 shows a balanced tree of $L = 3$ and $A = 3$. As we can see in the figure, two consecutive levels create subtrees: for example, there are 3 subtrees between levels three and two and, finally, one an only subtree between sublevel two and one.

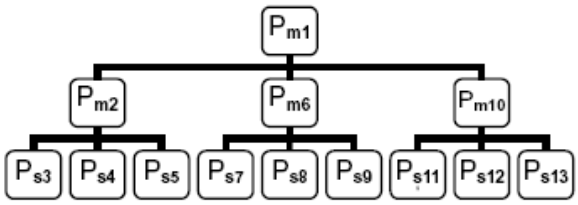

Fig. 2. Layout of Hierarchical Master-Slave Architecture

From a logical point of view, each subtree requires a particular physical network to reach the parallelism of its external communications. This way, the processors of intermediate subtrees need 2 communication interfaces, one for the network of the subtree which is root, and another one for network of the subtree which is a leaf. On the other hand, only one interface is required for the processors in the extreme levels 1 and L because they are part of just one subtree. Nevertheless, from a physical point of view, the number of logical networks can be reduced to one using Ethernet switches because they permit the separation of collision domains.

In this hierarchical model, due to the grouping of communications, in the phase of results communication, communication among subtrees of the same levels occurs simultaneous (in parallel), specifically, among master processors of level two and slaves processors of level three.

Of course, in this model there are two kinds of processors, masters and slaves:

- Master processors are all nodes that interconnect two different levels, in other words, are those that are not leaf nodes. The only purpose of master processors is to send/forward multisets: downwards to slaves to give them new job, and upwards to the root master from the slaves or intermediate masters.

- Slave processors are the leaf nodes of the tree and

their role is to apply rules and communicate the results to their immediate master processor in the tree.

1. Basements of Hierarchical Master-Slave Architecture

Considering the hierarchical distribution of processors, the pillars of this model are:

a. Membranes distribution as in Master-Slave.

b. Proxy for processor as in Master-Slave.

c. Balanced tree topology of processors. Benefit obtained from this interconnection topology among processors is that external communications among processors can be group and thus communications can occurs simultaneously, that is, parallel.

d. Communication policy. Inside each subtree, communication is carried out in the same ordered way that Master-Slave, that is, each slave has been assigned a position order and only transmits to its master when it is its turn.

Downward delivery of multisets is done by the root master processor in a sequential way to the intermediate master processors that are in their immediate lower level. These intermediate processors in turn do the same with the next lower level. The delivery finishes when multisets reach the last level, i.e., slave processors. It is important to note that there is parallelism in the communication at this stage because intermediate masters begin to transmit as soon as they get their multiset from its master parent regardless of brother processors that are at their left (at the same or lower levels). Of course, at the same time some intermediate master processors are communicating there is also some slave processors applying rules, i.e., there is parallelism between two phases of an evolution step. We can see the parallelization in the interval (t_x-t_y) in the Figure 3.

Upward return of results is done by slave processors in a sequential and ordered way to the intermediate master processor that is in the immediate higher level. The upward propagation of the results is done in the same way that downwards delivery. At this stage there is also parallelism in the communication phase. We can see the parallelization in the interval (t_y-t_z) in the Figure 3.

The benefit obtained with this communication policy is that it avoids collisions and network congestion, but additionally, and most of all, permits parallelize external communications.

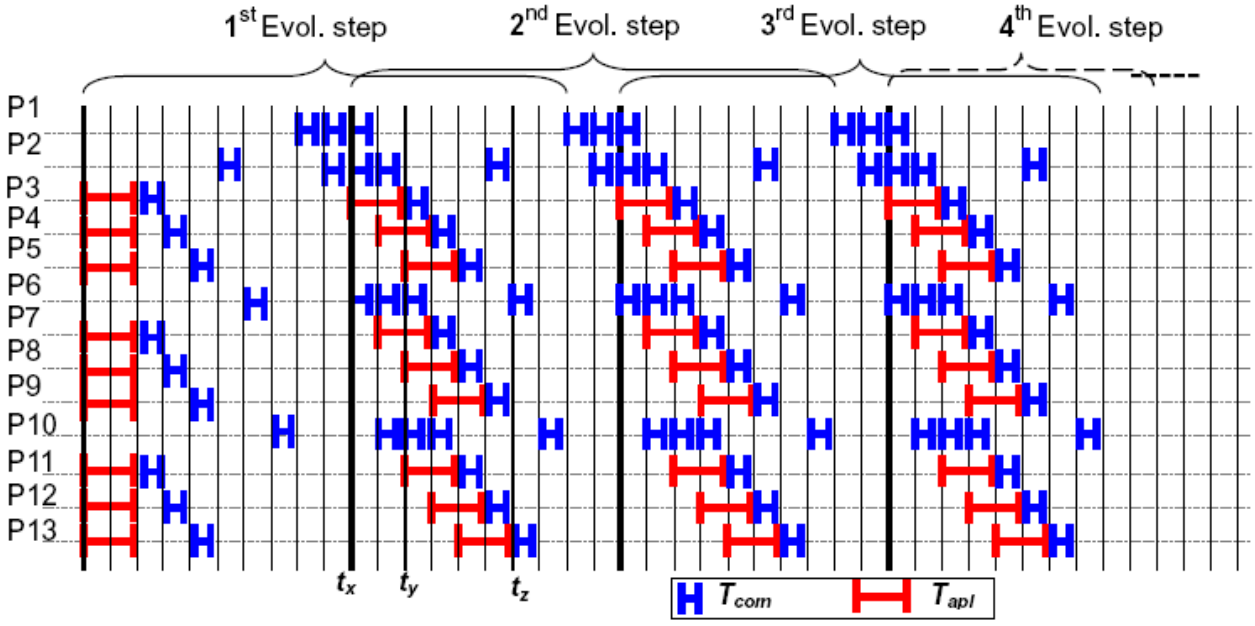

Fig. 3. Chronogram of Hierarchical Master-Slave Architecture

2. Hierarchical Master-Slave Analysis

Given that K membranes have been located in each processor, at the worst, the application of the rules in each one of these membranes will be made sequentially in each processor. Therefore, the required time to carry out the application of the rules of M membranes will be:

$$K\,T_{apl} \tag{5}$$

Number of slaves (P_s), masters (P_m) and total processors (P) is:

$$P_s = A^{(L-1)} \tag{6}$$

$$Pm = \frac{A^L - 1}{A-1} - A^{(L-1)} \tag{7}$$

$$P = \frac{A^L - 1}{A-1} \tag{8}$$

From (1) and (5) the required time to carry out the application of the rules of M membranes will be:

$$\frac{T_{apl}\,M}{A^{(L-1)}} \tag{9}$$

On the other hand, the required time to carry out the communication among processors in this architecture is:

$$T_{com}(2L + A + A^{(L-2)} - 4) \tag{10}$$

Therefore, from (9) and (10) the required time to perform a complete evolution step will be:

$$T = \frac{T_{apl}M}{A^{(L-1)}} + T_{com}(2L + A + A^{(L-2)} - 4) \tag{11}$$

Once the required time to perform an evolution step is known, we can determine the number of levels (L_{opt}) and the amplitude (A_{opt}) of the architecture in order to minimize this time. This function gets its minimum for a value of A (A_{opt}) that satisfies the expression below:

$$T_{apl}Mg(A) + T_{com}A^{g(A)} + T_{com}(g(A)-1)A^{(2g(A)-2)} = 0 \tag{12}$$

Being:

$$g(A) = \frac{\ln\left(\dfrac{-2T_{com} + \sqrt{4T_{com}^{\,2} + \dfrac{4T_{apl}T_{com}M\ln^2 A}{A}}}{\dfrac{2T_{com}\ln A}{A}}\right)}{\ln A} \tag{13}$$

And so,

$$L_{opt} = 1 + g(A) \tag{14}$$

3. Comparative Analysis

In this section, we present a comparative analysis of proposed flat Master-Slave architecture vs. the Hierarchical Master-Slave architecture suggested here.

Figure 4 shows the number of processors of both architectures to reach their respective optimum times for an evolution step. As it can be seen, hierarchical architecture has a bigger number of processors than the flat architecture. Also, the growing slope becomes steeper as the number of membranes of the P system is growing. This way, hierarchical architecture reaches a better parallelism degree in proportion to a bigger number of processors in the architecture. This fact increases the parallel application of evolution rules and the internal communication among membranes allocated at the same processor.

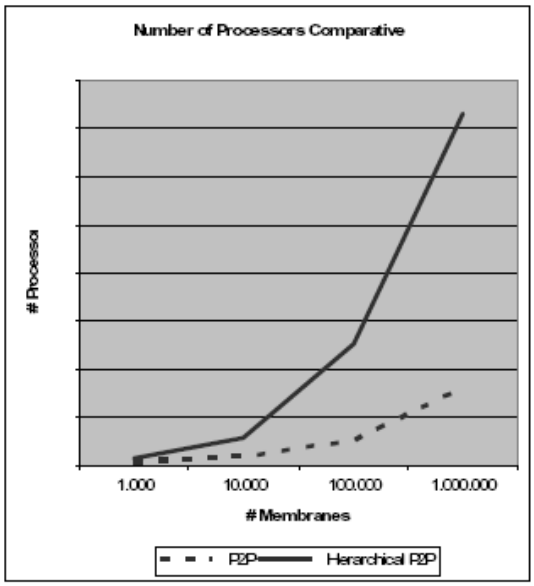

Fig. 4. Number of processors to reach optimum times per evolution step among membranes in both architectures.

Consequently, the bigger parallelization degree of hierarchical architecture and external communications parallelization among subtrees of same level obtains smaller times per evolution step. Figure 5 shows resulting times for both architectures as the number of membranes of the P system grow up.

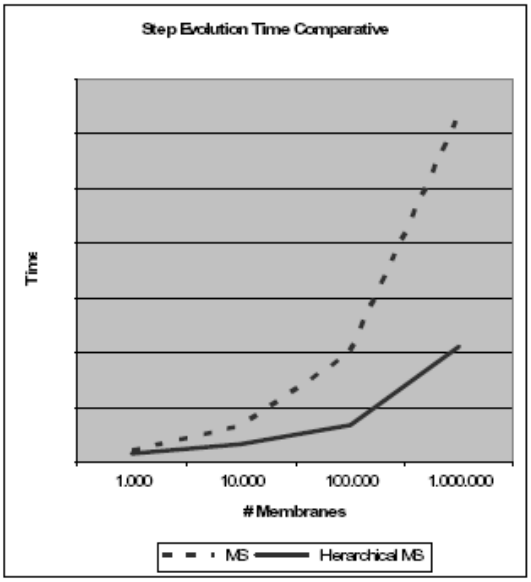

Fig. 5. Optimum times per evolution step in both architectures.

IV. CONCLUSION

In this paper a hierarchical architecture of communications to implement P system has been introduced. It is based on having a series of processors that redistributes multisets (called masters) to a series of another kind of processors (called slaves) that apply rules. Moreover, it places several membranes in each processor, uses proxies for communication among processors, and establishes a communication policy for the nodes.

This solution, just like flat Master-Slave architecture, avoids communication collisions and, moreover and unlike flat Master-Slave, reduces the number and length of the external communications, permits the parallelization of external communications and increases drastically the parallelization of application rules and external communications phases. All this, allows us to obtain a better step evolution time than flat Master-Slave architecture, as well as this architecture is closer to the massively parallelism character inherent to the membranes computer model and, last but not least, for the first time we have a hierarchical architecture that is suitable for any P System, regardless of its topology.

REFERENCES

[1] Gh. Pãun, *Computing with membranes, Journal of Computer and System Sciences*, 61, 1 (2000), 108-143.

[2] A. Syropoulos, E.G. Mamatas, P.C. Allilomes, K.T. Sotiriades, *A distributed simulation of P systems*, A. Alhazov, C. Martin-Vide and Gh. Pãun (Editors): Preproceedings of the Workshop on Membrane Computing; Tarragona, July 17-22 2003, 455-460.

[3] G. Ciobanu, M. Pérez-Jiménez, Gh. Pãun. *Applications of Membrane Computing*. Natural Computing Series, Springer Verlag, (October, 2006).

[4]A. Tejedor, L. Fernandez, F. Arroyo, G. Bravo, *An architecture for attacking the bottleneck communication in P systems.* M. Sugisaka, H. Tanaka (eds.), Proceedings of the 12th Int. Symposium on Artificial Life and Robotics, Jan 25-27, 2007, Beppu, Oita, Japan, 500-505.

[5] G. Bravo, L. Fernández, F. Arroyo, J. Tejedor, *Master-Slave Distributed Architecture for Membrane Systems Implementation*. 8th WSEAS Int. Conf. on Evolutionary Computing (EC'07) June, 2007, Vancouver (Canada).

[6] G. Bravo, L. Fernández, F. Arroyo, J.A. Frutos, *A hierarchical architecture with parallel communication for implementing P-Systems*. ITA 2007 X-th Joint International Scientific Events on Informatics, June 2007, Varna, Bulgaria.