Parallel Metropolis-Montecarlo simulation for Potts model using an adaptable network topology based on dynamic graph partitioning

Carlos Castañeda-Marroquín, Carmen B. Navarrete, Alfonso Ortega, Manuel Alfonseca, Eloy Anguiano
Dpto. Ingeniería Informática. Escuela Politécnica Superior
Universidad Autónoma de Madrid. 28049 Madrid, Spain
{carlos.castanneda, carmen.navarrete, alfonso.ortega, manuel.alfonseca, eloy.anguiano}@uam.es

Carmen B. Navarrete, Eloy Anguiano
Centro de Referencia Linux (CRL, UAM–IBM)
Escuela Politécnica Superior Universidad Autónoma de Madrid.
28049 Madrid, Spain

Abstract

In the last years, the computers have increased their capacity of calculus and networks - for the interconnection of these machines - have been improved until obtaining the actual high rates of data transferring. The programs that nowadays try to take advantage of these new technologies, cannot be written using the traditional techniques of programming, since most of the algorithms were designed for being executed in only one processor, in a nonconcurrent form, instead of being executed concurrently in a set of processors, working and communicating through a network. This work aims to present the ongoing development of a new method to simulate the Ferromagnetic Potts model, taking into account these new technologies.

1. Introduction and motivation

The use of clustering computing to solve computational problems has been the focus of high-performance computing community for more than two decades. The advances made in microprocessors and computer networks have caused the appearance of clusters or networks of workstations (NOWs) to be an alternative to the more and more expensive supercomputers. However, the demand of computing power continues growing as long as most of the available machines are underused. Due to this continuous growth of the capabilities of networks, there have been posed multiple optimisation problems associated with algorithms for the design of networks and topologies. For general optimisation of hosts in farms of computers, a methodology of dynamic resources allocation is needed [1, 2, 3]. Nowadays, this is one of the main reasons of the existence of a special interest in searching new algorithms, which will enable the replacement of traditional methods. The efficiency and possibility of scaling in parallel to the architectures of processors make the use of the traditional methods inapplicable in many cases [4].

The main problem when designing a parallel or distributed algorithm resides in the communication and synchronisation of processes for its concurrent execution in different processors. This is a very hard non-deterministic optimisation problem that not allways have the best solution. In many cases, only an approximation can be calculated.

From the point of view of the heterogeneity of processors, a good parallel application for Heterogeneous Network of Computers (HNoC) must distribute computations unevenly, taking into account, at least, the speed of the processors, the heterogeneous in terms of the network topology and the resources needed by each processor. The efficiency of the parallel application also depends on the accuracy of stimation of the processors speed of the HNoCs, which is difficult, because the processors may have different speeds for different applications due to the differences in the set of instructions, the number of instruction execution units, the number of registers, the structure of the memory hierarchy and so on [5].

From the point of view of the communications, one of the main problems desining the implementation of networks of communication is choosing a network topologies that could verify certain characteristics of trustworthiness, assuming this as a measurement, that evaluates the probability of success in the communication between pairs of processors. This is a non-trivial factor in the quality of services offered to all the computers. The evaluation of the ex-

978-0-7695-3472-5/08 $25.00 © 2008 IEEE
DOI 10.1109/ISPDC.2008.51

act parameters that determine the trustworthiness of a communication network is also a NP-hard problem [6, 7]. For this reason, an optimisation of the topology of the net is needed. It must be assured that the temporary delay due to the communication and synchronisation of the processes, is less than the delay of processing the data by each processor. Also, it is important to consider that when the problem is largely partitioned, the spent time to transmit data between computers and synchronise them, exceeds the time of computation of that CPU.

The common communication network is usually heterogeneous. The speed and bandwidth of the network, between different pairs of processors, may differ significantly. This makes the problem of optimal distribution of computations and communications across the HNoC much more difficult than across a dedicated cluster of workstations interconnected by an homogeneous high-performance network. Other issue is that the common communication network can use multiple network protocols for communication between different pairs of processors. A good parallel application should be able to use multiple network protocols between different pairs of processors [8].

The main idea of this ongoing work, focuses on the simulation of the Potts Ferromagnetic model, by the allocation of resources and tasks and the networks topology configuration, by transferring the workload onto other computers of the farm, to find a dynamic matching between the tasks and the global resources of the NOWs, that optimises this simulation completion time. This will be done assuming a non-static and decentralised approach.

The Potts model [9] was described by Renfrey B. Potts in 1952 and it is a generalisation of the Ising [10] model. This model is used to study the spins and behaviour of ferromagnets materials. It is based on a Montecarlo simulation that minimise the energy functions related to a N-dimensional computational grid. The Potts model has a great disadvantage related to the computational efficiency. This is because of the model is implemented using a Monte Carlo simulation, with a standard Metropolis algorithm. The election of this algorithm was choosen due to the hard computations that are needed to obtain the N possibles configurations for approach to the system behaviour.

In the following sections, we will introduce the definitions and backgrounds needed to understand the ongoing project. Then we will introduce the mathematical model that this system follows in order to study the performance and be able to explain the results that we have obtained. The third section of this paper explains the Potts model simulation implemented on this paper. At the end of the paper we plan the proposal and comment the results obtained.

2. Definitions and background

The NOWs can be seen as a weighted directed graph with costs in the vertices (resources costs) and in the edges (communications between pair of nodes costs). The vertices of the graph represent the nodes of the NOWs; the edges, the physical communication between pair of nodes of the farm. Discretely, one node is an instance of one processor, that is, every processor is one node. When applying the concept of the graph to a distributed and heterogeneous environment (HNoC), the costs associated with these weights depend on various system characteristics, such as the processing speed and the communication latency between network nodes. Each node of the NOW, or of the HNoC, can act as master or slave process: masters are entrust to send the tasks to be done by the slaves nodes and to manage the synchronism between the processes. Slaves play the role of executing the tasks sent by the master process. In our case of study, all nodes are equally master or slaves.

In order to build a computing environment for farms of computers, it is also necessary to have an algorithm that requires the ability to predict the performance and the resource consumption of different cluster configurations. This algorithm is called the Resource Management System (RMS). The problem is that, it is difficult to predict the computing time by a node before it receives some arbitrary load. Also, this will be even more difficult, if we consider the variation of the topology of communication of the farm. The basic tasks of the RMS is to accept requests for resources made by the applications and allocate them from the poll of resources. This is a slightly approximation of a computer middleware.

As all nodes are equally master or slaves, we need one special node, the supermaster, in where the RMS plays its role. The RMS uses a predictive estimation based on a mathematical function (heuristic function, Υ), in order to map the tasks of the parallel application to the pool of resources. This heuristic function will be the one needed to define the graph partition and assigning the workload to each node. These techniques for predicting the performance of the dynamic system are nowadays based on queueing techniques and/or on historical techniques [11]. Making a comparison between layered queues and the historical model, it is well known, that the layered queue requires more CPU time to make the mean response time prediction, whereas the historical predictions model are almost instantaneous. However, queuing techniques are easier to implement with a minimum level of performance than the historical model. This is because designing a historical model involves specifying and validating how predictions will be made, whereas the queueing model can be solved automatically. Both techniques can be combined to take advantages of them [12].

The layered queueing performance model defines an application's queueing network. The solution strategy, in this case, involves dividing the problem into tasks depending on the resources and corresponding them to the tiers of servers in the system model, generating an initial topology solution and then iterating with the historical method, solving and redimensioning the queues in each step of the algorithm, until the solution converge to an optimal distribution of resources, tasks and communications delays.

For the queue model it is necessary to define a queue structure for each node of the HNoC, which will be shared by all the incoming requests (figure 1). The nodes can be both clients (request information) and servers (process the information). The queue can be managed by a FCFS (First-Come, First-Servered), LCFS (Last-Come, First-Servered) or SIRO (Service In Random Order) policy

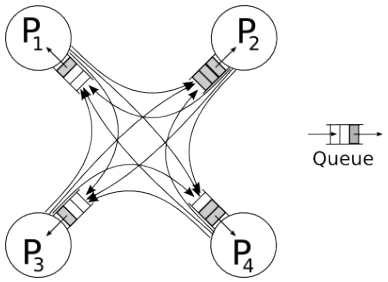

Figure 1. Example of an HNOC with 4 nodes. Each server has its own queue for receiving the incoming requests from the rests of nodes of the farm.

The different queue models are defined by 6 factors, in shortband notation, called Kendall's Notation [13], as A/B/C/X/Y/Z, where:

- A: the arrival process distribution. Distribution of probability assumed between arrivals.
- B: the service time distribution. Distribution of probability that gives the distribution of time of the service of a customer.
- C: the number of servers.
- X: maximum number of customers allowed in the system, including those being active. When this number is at the maximum value, arrivals are turned away.
- Y: the population size (number of processes).
- Z: the discipline of the queue. Priority order of jobs in the queue.

Thus, solving the problem of the optimisation of the global resources of the HNoC, can be approached by solving the problem of finding the minimum path of the associated graph [14], taking into account the communication delays and the execution time per task of each processor.

3. Mathematical model

The HNoC system can be modelled as a weighted directed graph G_s, denoted by $G_s(P, L, \tau, \delta)$, referred as the $SystemGraph$; P denotes a finite set of processors that represents the nodes or vertices of the graph G_s; L is a finite set of links that represents the communication links between pair of processors: the edges of the graph G_s; Each vertex $p_i \in P$ is characterised by a set of system parameters (memory, frequency, operating system...), based on its available resources of the HNoC. Due to this, each processor has a processing weight $\tau(p_i)$ that denotes the processing cost per unit of computation. Each link between two processors p_i and p_j, denoted by $l_{i,j} \in L$, has a link weight $\delta_{i,j}$ that means the communication latency between those two nodes per transfer unit. If two nodes $(p_i, p_j) \in P$ are not connected to each other, then $l_{i,j} = \infty$. We assume that all nodes of the graph are connected to at least one node of the HNoC (connected graph) but we not enforce constraints on the network topology, as this is not completely defined and can vary between two steps of the simulation of the problem, that is being executed in the HNoC. It is necessary to define a neighbourhood function which will return the set of nodes that are linked with any node of the HNoC,

$$\forall p_i \in P : neig(p_i) = \{p_k\} \mid l_{i,k} \neq \infty$$

So, we must define another function $path(p_i, p_j)$, defined as

$$\forall p_i, p_j \in P : path(p_i, p_j) = \{p_k\}^*$$

were p_k^* is a sorted set of nodes were each node either $p_k \in neig(p_i)$ or $p_k \in neig(p_j)$ or $p_k^n \in neig(p_k^{n-1})$. It is necessary to define a latencies matrix C_L containing the network latencies between any two processors, $\forall p_i, \forall p_j \in P, C_{L_{i,j}} = lat(p_i, p_j)$, which will depend also on the physical and data link network layer. For two adjacent nodes p_i, $p_j, \in P, C_{L_{i,j}} = lat(p_i, p_j) = \delta_{i,j}$ but if p_i and $p_j \in P$ are not adjacent in the HNoC, the latency will be defined as the sum of the links weights on the shortest path between them,

$$C_{L_{i,j}} = lat(p_i, p_j) = \sum (\delta_{p_k} \mid p_k \in min(path(p_i, p_j)))$$

(figure 2). The matrix C_L could be symmetric or not, since all communications could be different, also in a duplex communication, because of the directed property of the graph.

The graph of minimum paths for a specific node of the graph G_s, denoted by ${G_m}^a(P, L, \tau, \delta)$, will be defined as

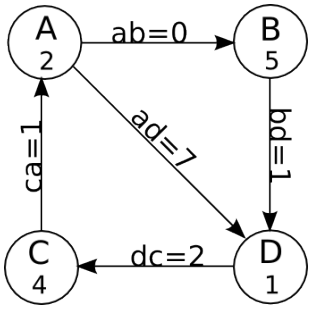

Figure 2. Example of a System Graph G_S. The characteristics of the graph are: nodes $P = \{A, B, C, D\}$, links $L = \{ca, ab, ad, bd, dc\}$, processing weights $\tau = \{2, 5, 4, 1\}$ communication latencies $\delta = \{1, 0, 7, 1, 2\}$ and $neig(A) = \{B, D\}, path(C, D) = \{C, A, B, D\}U\{C, A, D\}$, but $path_{min}(C, D) = \{C, A, B, D\}$

The C_L matrix for this example is:

$$C_L = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 0 & 3 & 7 \\ 4 & 0 & 3 & 1 \\ 1 & 1 & 0 & 2 \\ 3 & 3 & 2 & 0 \end{pmatrix} \end{array}$$

the graph that contain the minimal paths from the node $a \in P$ to any node of the graph (figure 3).

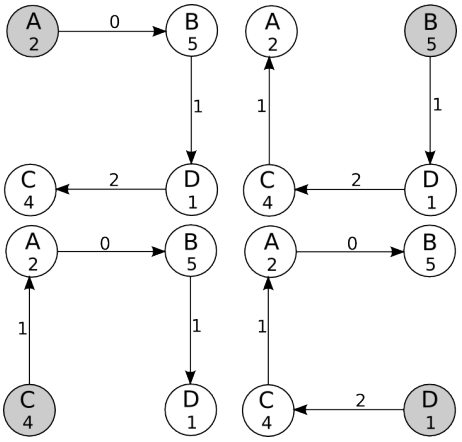

Figure 3. Obtained graphs ${G_m}^a$ for each node of the graph of the figure 2.

The application can also be modeled as a weighted directed graph G_a, denoted by $G_a(T, D, \omega, \lambda)$, referred as the $ApplicationGraph$; T denotes a set of vertices of the graph that represents the tasks to be done; D represents a finite set of edges of the graph where $\{(t_i, t_j) \mid t_i, t_j \in T\}$; Each vertex has a computation weight $\omega(t_i), \forall t_i \in T$ that represents the amount of computations required by the task t_i to accomplish one step of the algorithm. Each edge has a value $\lambda_{i,j}$ that represent de amount of data to be sent from v_i to v_j.

Thus, the execution time Γ of a task $t_i \in T$ on a processor $p_j \in P$, assuming the worst case in which there is no-overlapping between computation and communication, is defined as:

$$\Gamma(t_i, p_j) = \omega(t_i) \times \tau(p_j) + \sum_{t_l \in neig(p_k)} \sum_{\substack{p_k \in P \\ k \neq j}} \lambda(t_i, t_l) \times \delta(p_j, p_k)$$

where $\omega(t_i) \times \tau(p_j)$ represents the amount of computation required by the task t_i per processing cost per unit of computation.

Given a system graph $G_S(P, L, \tau, \delta)$ and an application graph $G_a(T, D, \omega, \lambda)$, the objective is to map characteristics $\Gamma : (T, D) \mapsto (P, L)$ for minimising the function Γ, based on the application requirements and the system constraints such as the topology of the system graph.

From the point of view of the queue prediction technique, we can assume that the HNoC can be modelled by a $M/G/1/\infty/\infty/SIRO$ model, where the arrival process distribution is a Markovian process [15], the service time is a general distribution referring to independent arrivals to the

system and there is only one server for that queue [16]. We must allow any server to pop an item from the queue in an arbitrary order, according to a certain priority value. This characteristic forces the $SIRO$ parameter of the queue. We will maintain only a queue per each server, but, from the point of view of the whole system, the queues will be modelled as a $M/G/c/\infty/\infty/SIRO$, where c is the number of nodes of the HNoC. The length of the queue (maximum number of jobs in the queue) L, will depend on the arrival rate λ, and on the service rate μ and will be calculated by the expression:

$$L = \frac{\lambda^2 E[S^2]}{2(1-\rho)} + \rho$$

where $\rho = \frac{\lambda}{\mu}$ and $E[S^2]$ is the second moment of the expected value of the service rate random distribution S.

The historical modelling technique involves sampling performance metrics (response times, resources availability, communication delays...) and will associate these measurements with variables representing the workload being processed and the machines architecture. The historical function, in most cases, cannot be solved mathematically and it is necessary to solve it, by using an optimisation method as a simplex or $tableau$ algorithm [17].

4. Potts Model

The Potts model consists of spins that are placed on a lattice; the lattice is usually taken to be a two-dimensional rectangular Euclidean lattice, but is often generalised to other dimensions or other lattices. The Hamiltonian function is defined as follows:

$$H = J \sum_{(i,j)neighbors} 1 - \delta(\sigma(i'), \sigma(j')) \qquad (1)$$

where J is a positive constant, δ is the Kronecker delta ($\delta(x,y) = 0$ if $x \neq y$ and 1 if $x = y$) and $(i,j), (i',j')$ denotes the first neighbourhood area.

The probability function is defined as:

$$P = e^{-\frac{\Delta H}{K_B T}} \qquad (2)$$

where K_B is the Boltzman constant, T is a certain temperature value. The new state of the spin is accepted when (see expression 1) $\Delta H > 0$ with probability P or 1 if $\Delta H \leqslant 0$

Evolution of the model proceeds using the Metropolis Montecarlo [18] simulation as follows:

1. Select randomly a lattice spin
2. Choose randomly a neighboring spin of the current site
3. If the neighboring spin is equal to the spin of the current site, go to step 1
4. Change the current site spin for the spin of its neighboring site, according to the probability *P* as in expression 2
5. Return to step 1

The lattice is represented as a torus and it is continually updated: for each lattice point, a different spin state is proposed, and the new overall energy calculated. It depends on the neighbour's interactions and the overall temperature. If the new energy is smaller than the old one, the new state is accepted. If not, there is still a certain chance that will be accepted, leading to random spin flips representing the overall temperature.

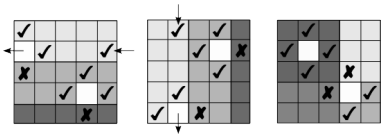

Figure 4. Possible domain decomposition and explanation of the situation in which one processor must send a signal to another one, to update the current grid position.

The critical part of the algorithm, from the point of view of the parallelisation, resides in the step 2. Not allways the domain and data decomposition assure us, that the needed variables for the calculus are at the processor that is calculating a certain expression (see figure 4).

The Potts model using the conventional Monte Carlo method has two drawbacks: The critical slowing down and the generation of random numbers. These drawbacks are the two computer time-consuming factors. Our work describes an alternative parallel Metropolis - Monte Carlo solution, using a C and an MPI library, which is more efficient than the traditional method. There are several publications about parallel implementations of this model such us the one described in [19]. Our goal is to parallelise the Potts model to be executed it in a parallel environment, created by a heterogeneous cluster, using the MPI library [20]. The aim is to increase the speed-up and the efficiency to reduce these drawbacks across of a correct domain and data decomposition, assuring us that, on the one hand the needed values for the "spin-flip" attempt calculus are accessible by the processor that is calculating the possible spin-flip attempt and not have problem with a wait queue, and, on the other hand always after of a Monte Carlos Step (MCS) the domain decomposition is correct.

5. Proposal

Until this moment, we have considered only a static network topology, that depends on the domain decomposition, defined before the execution of the algorithm or application in the HNoC; in those problems were it is not trivial to make a definition of a domain decomposition, it would be usefull to have a dynamic topology of communications. This dynamic topology will vary around the distribution of the data and the different latencies of the network and this will depend on the different connections between the nodes.

The model proposed for the reconfiguration of nodes of an HNoC will be based on a sufficiently ample language of communications. Using this language any node will be able to know in real time, which information contains any other node of the system (data middleware), without having to communicate first with the master node of that system. This language will also allow the nodes to modify their roles of master/slave depending on what value or data structure is needed and on what node has asked for it (figure 5).

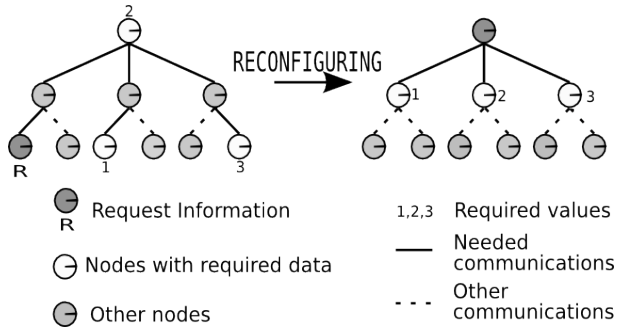

Figure 5. Example of the reconfiguration of an HNOC applied to a distributed database. At least 3 nodes have changed its master/slave role, to adapt the HNoC network topology to the data distribution. This has been made in order to minimise the global delay for the execution of the query in the system.

The HNoC will be represented as a directed dynamic graph (as in figure 2) in which the different connections will have weights of edges (cost of moving directly from one vertex to another one) equals to the different delays from network and those due to the overload of the processors of the nodes. The vertex of the graph will represent each processors available at the HNoC. The information of each node of the graph will contain the effective load of the processor, considering this as the availability to execute other processes. Also the node will contain as well, a statistic value proportional to the execution time of the processes in previous steps of the algorithm. Therefore, the edges of the graph will represent the connections between the available processors, according to a certain instant of the algorithm. These connections are statistically weighted according to the network delays and the time of transmission of the data through the net that depends, as well, on the network protocol, and the physical layer used for this communication. These two values will be important for the calculation of the optimised values of the historical function.

In addition to the nodes that are included in the HNoC, a super master node will exist. This node will be the one which will administer the minimum graphs, that represents the path used by each node to communicate with the other ones (as in figure 3). The super master node will maintain the resulting graph of all the existing communications between any two nodes of the HNoC and also the execution times of each processor. With this information, and according to a statistical and heuristic function Υ, this node will generate an optimal graph, probably different, for each node of the HNoC. These graphs will be calculated based on the communications that each node needs to make with any other one, its workbalance, the value $\tau(p_i)$, some historical values, ..., and they will be calculated according to some algorithm of minimum path for graphs (figure 6). Knowing that all the weights of the vertices and of the edges of the graph are allways positive values, we could use algorithms based on Dijkstra [21] or Bellman-Ford [22].

In order to be able to reconfigure the HNoC, the language of communications will include control dataframes and data dataframes. The data dataframes will contain the data needed for the execution of the algorithm in the HNoC, whereas the control dataframes will be those with information about the execution times and commands, and will be sent for knowing which nodes have what information and what hierarchy exists between the different nodes of the HNoC. This will be possible by developing a monitoring layer that will advise the supermaster node on each event that occurs in any node of the HNoC.

The control dataframes will also allow to any node, to join or exit the farm of computers dynamically, sending a specific command to the supermaster node, to be included on, or to be deleted from the grouping. When a computer join the group, the supermaster will calculate again the graph of minimum paths for each node of the farm, in order to put in context the new node with the rest of computers. This calculus will be done with the new parameters of the system. When a node leaves the grouping, it will advise the supermaster node and the unregistration process will cause again a reorganisation of the nodes of the HNoC.

The main advantage of this system is to avoid the design of an HNOC by the administrator of the cluster. This person, without this system, must design the architecture and the HNOC topology in function of the accesses that are

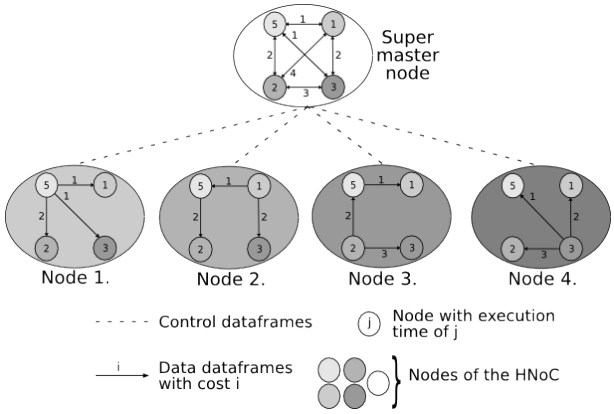

Figure 6. HNoC formed by 5 processors, 4 nodes and 1 supermaster node. The supernode contains a complete-graph with all the possibles communication between each two nodes and the statistical value for the delays and effective load of each processor. The other nodes are dedicated to realize the calculus assigned by the supermaster node. Each of these nodes can play the roll of a master, a slave or even a mixed solution. In this example we have also to consider that all nodes contains the same information data. The supermaster node will calculate the minimum graphs for each processor and will communicate it to them. With these graphs, the other nodes know to which one must it send the information data. Each processor can resend the information to another node as a proxy.

needed depending on the problem to study. Also, this model will be usefull to eliminate the need of distributing the resources among the different nodes, in function of the pool of availables resources.

6. Results

Because of the development status of the application, currently there are not performance and efficiency results of the use of this reconfiguration technique.

Four simulations have been performed: one simulation for the non-parallel applications and three more to study the best domain decomposition, in order to optimise the communications between pairs of processes, to increase the speed up of the execution of the problem. In these simulations we have decided to use a domain decomposition based on horizontal or vertical stripes or based on a square decomposition, also taking into account the processor speed of each node of the cluster. These initial domain decompositions are needed to get higher performance values as possible, in order to study the behaviour of the problem, depending on the initial input values and the initial distribution of nodes and data. The goal of our algorithm is the way we solve the possible situation in which one node needs some information in order to update the current grid position that is in another node of the cluster due to the domain and data decomposition.

The results obtained with this simulation are physically equivalent to the ones obtained with the standard Metropolis Montecarlo algorithm (non-parallelised simulation). This assure us that the initial partition has nothing to do with the Potts model and the Ferromagnetic properties. The ongoing work will consist on applying these concepts of reconfiguration, to simulate the cell growth using a Cellular Potts Model (CPM [23]) and using a dynamic domain decomposition based on the cell dimensions.

References

[1] R. Canal, J.M.l Parcerisa, and A. Gonzalez. Dynamic cluster assignment mechanisms. In *HPCA*, pages 133–, 2000.

[2] R. Bhargava and L. John. Improving dynamic cluster assignment for clustered trace cache processors. Technical report, 2003.

[3] K. Amiri, D. Petrou, G. Ganger, and G. Gibson. Dynamic function placement in active storage clusters. Technical report, 1999.

[4] A. Lastovetsky. Scientific programming for heterogeneus systems - bridging the gap between algorithms

and applications. In *PARELEC'06 IEEE Proceedings*, pages 3–8, 2006.

[5] A. Lastovetsky and R Reddy. HeteroMPI: Towards a message-passing library for heterogeneous networks of computers. *Journal of Parallel and Distributed Computing*, 2005.

[6] M.O. Ball. Computing network reliability. 1979.

[7] J.S. Provan and M.O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, (12):777 – 788, 1983.

[8] J.Dongarra and A. Lastovetsky. *An overview of heterogeneous high performance and grid computing*. American Scientific Publishers, 2006.

[9] Renfrey B. Potts. Some generalized order-disorder transformations. volume 48, pages 106–109. Proceedings of the Cambridge Philosophical Society, 1952.

[10] E.Ising. Beitrag zur theorie des ferromagnetismus. *J Physics*, 31, 1925.

[11] David A. Bacigalupo, Stephen A. Jarvis, Ligang He, Daniel P. Spooner, and Graham R. Nudd. Comparing layered queuing and historical performance models of a distributed enterprise application. In *IASTED International Conference on Parallel and Distributed Computing and Networks*, pages 608–613, 2005.

[12] David A. Bacigalupo, Stephen A. Jarvis, Ligang He, D. Spooner, D. Pelych, and Graham R. Nudd. A comparative evaluation of two techniques for predicting the performance of dynamic enterprise systems. In *PARCO*, pages 163–170, 2005.

[13] L. Kleinrock. *Queuing Systems: Theory*. Wiley, 1975.

[14] Bassel R. Arafeh, Khaled Day, and Abderezak Touzene. A paradigm for allocating parallel application tasks to heterogeneous computing resources on the grid. In *PARCO*, pages 41–48, 2005.

[15] B. Song, C. Ernemann, and R. Yahyapour. Parallel computer workload modeling with markov chains. In *Proceedings of the 10th Job Scheduling Strategies for Parallel Processing (JSSPP)*, volume 3277, pages 47–62. Lecture Notes in Computer Science, Springer-Verlag, 2004.

[16] D. Gross and C.M. Harris. *Fundamentals on Queuing Theory*. Wiley, 1998.

[17] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, January 1965.

[18] M. Rosenbluth A. Teller E. Teller N. Metropolis, A. Rosenbluth. Equation of state calculations by fast computing machines. *J. Chem. Phys*, 21(13):1087–1092, 1953.

[19] Eunice E. Santos and Gayathri Muthukrishnan. Efficient simulation based on sweep selection for 2-d and 3-d ising spin models on hierarchical clusters. *ipdps*, 14:229b, 2004.

[20] Message Passing Interface Forum. MPI: A message-passing interface standard. Technical Report UT-CS-94-230, 1994.

[21] Edsger. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[22] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87 – 90, 1958.

[23] J.A. Glazier F. Graner. Simulation of biological cell sorting using a 2-dimensional extended potts model. *Phys. Rev Lett*, 69(13):2013–2016, 1992.