Evaluating the cost of the dynamic reconfiguration of a multicomputer network

J. M. García
Dpto. de Informática
Universidad de Castilla-La Mancha
Campus Universitario s/n
02071 Albacete (Spain)

J. Duato
Facultad de Informática
Universidad Politécnica
Camino de Vera s/n
46071 Valencia (Spain)

Abstract

The dynamic reconfiguration of the interconnection network is an advanced feature of some multicomputers. It allows to reduce the communication overhead, improving the performance. In a previous paper [6], we presented a reconfiguration algorithm based on a cost function. It reconfigured the network in such a way that the original topology was preserved. However, that algorithm did not take into account the cost of the reconfiguration itself. In this paper, we consider that cost. The resulting reconfiguration algorithm has been evaluated by simulating the execution of a test problem (triangularization of a sparse matrix). The results show the benefits obtained when the interconnection network is reconfigured dynamically.

Keywords: Multicomputer network, multicomputer performance, dynamic network reconfi-guration, computer architecture.

1. INTRODUCTION.

Multicomputers [3] are among the most interesting architectures, meeting the high performance computing requirements in application areas like computational fluid dynamics, image processing and circuit simulation. Multicomputers rely on an interconnection network between nodes (processors) to support the message-passing mechanism.

The interconnection network plays a central role in determining the overall performance of a multicomputer [2,11]. If the network cannot provide adequate performance, nodes will frequently be forced to wait for data to arrive. Further, as technological improvements increase the computing power of nodes, additional pressure is placed on the communication network to provide a comparable improvement in performance to avoid becoming a bottleneck. In addition to performance considerations, the interconnection network should be tolerant to failures.

A interesting solution to alleviate this problem is to make the interconnection topology reconfigurable, i.e., the network topology can change almost arbitrarily at runtime [7,10]. A network with this feature has two properties: on the one hand, it can easily match the network topology to the communication requirements of a given program, thus exploiting the *communication locality*, even when those requeriments vary over time. On the other hand, faulty nodes or links can be easily bypassed and spare nodes can be switched on. Nowadays, there are several multicomputers with this advanced feature [1,4].

In a previous paper [6], we presented a reconfiguration algorithm based on a cost function to handle the dynamic reconfiguration of the interconnection network in a multicomputer. The goal of the dynamic reconfiguration is to increase the multicomputer performance by means of minimizing the traffic of messages through the network. Therefore, the cost function evaluate the network traffic for each node, taking into account the total message traffic between a node and the other ones, weighed with the nominal distance between nodes.

This algorithm reconfigure the network in such a way that the original topology is preserved. The first results showed an important improvement for application programs with a time varying communication pattern. However, that algorithm did not take into account the cost of the reconfiguration itself.

In this paper we consider this cost, evaluating the time needed to reconfigure the network. We also show the performance improvement obtained in a complex mathematical problem, when that time is taken into account.

The rest of paper is organized as follows. Section 2 summarizes our previous work detailing the reconfiguration protocol we have implemented. In section 3 the time needed to reconfigure the network is computed. The new algorithm which includes the reconfiguration cost has been evaluated, showing in section 4 the results obtained by simulation. Finally, the main conclusions are presented in section 5.

2. MACHINE MODEL AND DYNAMIC RECONFIGURATION PROTOCOL.

This work was originated while trying to increase the communication performance of a transputer-based machine, the PARSYS SN 1000. Then, we have mainly focussed on store-and-forward networks. Also, we assume that a static scheduling is used to map processes to processors, thus preventing process migration. To handle the dynamic reconfiguration, we have set up a control in a centralized way, such as it is available in the SN 1000.

All the work has been carried out on the FDP environment [8], which permits the simulation of the behaviour of a multicomputer with a particular topology. FDP also offers a simple and efficient programming environment with a friendly user interface.

The basic idea for the dynamic reconfiguration is the following: When the traffic between a pair of nodes is intense, the reconfiguration algorithm will try to put the source node close to the destination node by exchanging the positions of either the source node and a neighbour of it or the destination node and one of its neighbours. It must be noticed that the implementation of this algorithm is distributed, each node taking into account the information recorded locally.

The dynamic reconfiguration is a suitable technique to deal with those problems whose communication pattern varies over time, especially when it is not possible to predict the variation a priori. The reconfiguration algorithm we have developed decides when it is convenient to carry out a change in the topology. The algorithm has been described in [6].

Our machine model is based on the Parsys SN1000. In this model, a master node (the system controller) is responsible for controlling the reconfiguration.

The network reconfiguration protocol works in the following way:

1).- When a node decides that it is necessary to reconfigurate the network, it sends a signal to the system controller through the control bus.

2).- Then, the system controller informs all the nodes that it is going to make a reconfiguration and therefore they should stop sending messages to each other. To minimize the reconfiguration time and relieve the cost that it implies, messages in transit are only allowed to reach the next intermediate node. After stopping messages in transit, each node sent an acknowlegement to the system controller.

3).- The node which made the request sends the reconfiguration data to the system controller to carry out the reconfiguration.

4).- The system controller modifies the interconnection network topology, adapting it to the new circumstances.

5).- Once the new configuration has been established, the system controller broadcasts this configuration to all the nodes and permits node communication again.

This protocol is easily implementable using the control bus available in the SN1000 architecture, which does not add message traffic to the network.

The centralized control could be a bottleneck for the whole system, since all the nodes must send reconfiguration requests to the system controller. To avoid it, the reconfiguration algorithm will only reconfigurate the network when it estimates that a large amount of data have to be transferred.

3. THE RECONFIGURATION COST.

As the machine is not ideal, whenever a change is made in the topology, some time is wasted, which reduces the improvement that would be obtained in an ideal machine. In this paper, we focus on the cost involved when a change in the network topology is performed.

By analysing the network reconfiguration protocol, the following times can be obtained:

a) T_a : Time spent in each node to evaluate the cost function and to decide whether a reconfiguration is needed. It measures part of the step 1 of the previous protocol.

b) T_b : Time spent to transfer the reconfiguration information from the node which requests the change to the system controller. It includes the time required to request the change as well as broadcasting the message to stop the communication between nodes, receiving the acknowledgement and sending information about the new topology to the system controller. Therefore, this time includes part of the step 1 and steps 2 and 3 of the previous protocol.

c) T_{rec} : Time needed to reconfigurate the interconnection network topology. It corresponds to step 4 of the previous protocol.

d) T_c : Time needed to broadcast the new configuration to all the nodes, allowing again communications between processors. It corresponds to the last step of the previous protocol.

Then, the cost for making a change in the network topology is given by the following expression:

$$T_{change} = T_a + T_b + T_{rec} + T_c \qquad (1)$$

Let us compute an upper bound for this expression. T_a is the time required to compute the cost function. We estimate a value of 100 μsec for it.

The communications between the system controller and the nodes are performed through the control bus, requiring less than 100 μsec. With regard to the time needed to stop message traffic, we are going to compute the time required to transfer a message between adjacent nodes. In [12] the following formula is used:

$$T_{comm} = \alpha + \beta * L \qquad (2)$$

where L is the length of the message (in bytes) and α and β are machine dependent constants. Usually, long messages are split into fixed size packets. For the transputer [9], we have obtained for T_{comm} a value ranging from 500 µsec to 1 ms, giving for T_b an estimated value ranging from 600 µsec to 1.1 ms.

The other constant factor, T_c, also measures communications through the control bus. Again, we estimate a value equal to 100 µsec.

Finally, we have to evaluate the time needed to perform a reconfiguration in the interconnection network. In [5] the following formula to compute the reconfiguration time is proposed:

$$T_{rec} = \gamma + \delta * N \qquad (3)$$

where γ is the start-up time for the reconfiguration, δ is the required time per modified link and N is the number of modified links in each reconfiguration. Since changes are local, the number of links which are modified in each change is always the same. When two nodes exchange their positions, the number of modified links is equal to 6. Using the values proposed in [5] for γ (100 µsec) and δ (100 µsec/link), the time needed to perform a network reconfiguration is equal to:

$$T_{change} = 800 + 100 + 100 * 6 = 1,500 \mu sec = 1,5 ms \qquad (4)$$

for short messages. For long messages, T_{change} = 2 ms.

4. EVALUATION OF THE ALGORITHM.

To evaluate the performance of the reconfiguration algorithm in a complex case, we have chosen the triangularization of a sparse matrix based on Givens rotations. This algorithm is very suitable for parallel machines because of its inherent parallelism. This algorithm has been chosen because the communication pattern will vary over time, making it very suitable for dynamic reconfiguration. A detailed explanation of the algorithm as it has been simulated under FDP can be found in [7].

The results we show measure the total message traffic in the network. In the case of dynamic reconfiguration, we can take into account the reconfiguration cost. The reconfiguration cost is approximately equal to the time required to transfer three messages between adjacent nodes in the worst case (short messages). Therefore, it is important to limit the number of changes.

In figure 1 we show the total traffic of messages through the network for a 600x300 sparse matrix with an average of two non-zero elements per row. For comparison purposes, we show the results obtained with three static topologies (ring, 2-D mesh and hypercube) and the results obtained when network reconfiguration is allowed, both with (real reconfiguration) and without (ideal reconfiguration) the reconfiguration cost.

In the case of reconfiguration, the basic topology is a hypercube. The values obtained for the network traffic vary as a function of different parameters from the reconfiguration algorithm [6].

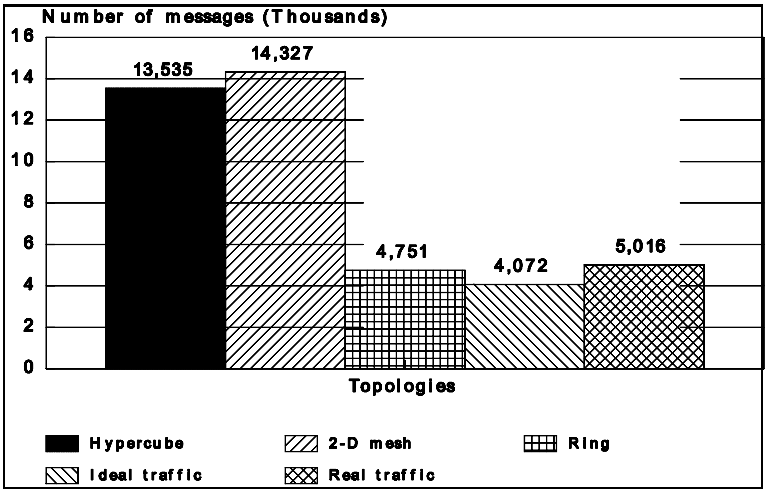

Fig. 1. Message traffic in the network for a 600x300 matrix

As can be seen in the figure, the reduction in the network traffic is drastic with respect to the static hypercube. However, the reduction is smaller when compared with the ring topology, because this is the optimal topology when the matrix is full. It must be noticed that small matrices are quickly filled.

These results have been confirmed with many tests with matrices of different sizes [7]. The results obtained are very good, showing that the larger the matrix, the better results are achieved.

These results confirm the suitability of the dynamic reconfiguration as a means of achieving a reduction in message traffic, therefore improving the performance of multicomputers.

5. CONCLUSIONS.

In this paper we have presented the dynamic reconfiguration of the interconnection network as a valid alternative to solve the main problem associated to multicomputers: the communication bottleneck. By means of using a dynamic topology, the principle of locality in communications is exploited, leading to a substantial improvement in network latency [2].

The main contribution of this paper is the evaluation of the time needed for reconfiguring the topology according to the proposed protocol. Once this time has been computed we have shown how the use of the network reconfiguration leads to a reduction in message traffic.

REFERENCES

[1] Adamo, J. and Bonello, C. "Tenor++: A dynamic configurer for Supernode machines". *Lecture Notes in Computer Science* No. 457, pp. 640-651, Springer-Verlag, 1990.

[2] Agarwal, A. "Limits on interconnection network performance". *IEEE Trans. on Parallel and Distributed Systems*, Vol. 2, No. 4, pp. 392-412, October 1991.

[3] Athas, W. C. and Seitz, C. L. "Multicomputers: Message-passing concurrent computers". *IEEE Computer*, Vol. 21, No. 8, pp. 9-24, August 1988.

[4] Bauch, A.; Braam, R. and Maehle, E. "DAMP: A dynamic reconfigurate multiprocessor system with a distributed switching network". *Proc. 2nd*

European Distributed Memory Computing Conference, Munich, April 1991.

[5] Desprez, F. and Tourancheau, B. "Evaluation des performances de la machine T.node". La lettre du Transputer, LIB Besacon, France, No. 7, 1990.

[6] García, J.M. and Duato, J. "An algorithm for dynamic reconfiguration of a multicomputer network". *Proc. Third IEEE Symposium on Parallel and Distributed Processing*, Dallas, December 1-5, 1991.

[7] García Carrasco, José M. "*Desarrollo de Herramientas para una Programación Eficiente de las Redes de Transputers: Estudio de la Reconfiguración Dinámica de la Red de Interconexión*". PhD thesis. Universidad Politécnica de Valencia. December, 1991.

[8] García, J.M. and Duato, J. "An Advanced Environment for Programming Transputer Networks with Dynamic Reconfiguration". *Int. Conf. on Parallel Computing and Transputer Applications*, Barcelona, September 1992.

[9] Inmos Corporation. *"The Transputer Databook"*. Inmos Ltd., England, 1989.

[10] Nicol, D.A. "Reconfigurate transputer processor architectures", in T.J. Fountain and M.J. Shute (Ed), *Multiprocessor Computer Architectures*, North-Holland, 1990.

[11] Reed, D.A. and Fujimoto, R.M. "*Multicomputer Networks: Message-based parallel processing"*. The Mit Press. London, England, 1987.

[12] Zhang, X. "System effects of interprocessor communication latency in multicomputers". *IEEE Micro,* pp. 12-15 and 52-55, April 1991.