Edinet: An Execution Driven Interconnection Network Simulator for DSM Systems *

J. Flich, P. López, M. P. Malumbres and J. Duato

Depto. Informática de Sistemas y Computadores, Universidad Politécnica de Valencia
Camino de Vera s/n, 46071 - Valencia, SPAIN
jflich,plopez,mperez,jduato@gap.upv.es

Abstract. Evaluation studies on interconnection networks for distributed memory multiprocessors usually assume synthetic or trace-driven workloads. However, when the final design choices must be done a more precise evaluation study should be performed. In this paper, we describe a new execution-driven simulation tool to evaluate interconnection networks for distributed memory multiprocessors using real application workloads. As an example, we have developed a NCC-NUMA memory model and obtained some simulation results from the SPLASH-2 suite, using different network routing algorithms.

1 Introduction

Interconnection network evaluation studies usually assume that the workload generated by actual applications can be modeled by synthetic workloads. Constant message generation rate (equal for all nodes), uniform distribution of message destinations (with or without locality) and fixed traffic patterns are frequently used. However, when the designer must do the final choices, a more precise evaluation study should be performed.

Some studies have relied on trace-driven simulations [9, 8]. However parallel traces generated for one architecture may never happen on another. For instance, some messages should not be generated until some other has been arrived at its destination, and this depends on the interconnection network performance, which is not being simulated when the trace file is being created.

The definitive tool to get accurate evaluation results are the execution-driven simulators [4, 5]. In such simulators, an application runs on the host processor and special call-outs are inserted into the original code to instrument the required events. These events are scheduled as requests to the simulator. In our case, the events are the message send/receive primitives, and the simulator is the interconnection network simulator. In addition, a parallel application simulator is also required in order to execute the parallel code on a single processor.

In this paper, we present a new simulation tool, called EDINET, that allows executing a shared-memory application on a simulated DSM system. Our goal will be accurately analyze the interconnection network performance using real application workloads.

* This work was supported by the Spanish CICYT under Grant TIC97–0897–C04–01.

2 Edinet

The EDINET simulator is composed of two simulators. The first one is Limes [5], an execution driven simulator that allows parallel program execution and models the memory subsystem. The second one is the interconnection network simulator (NetSim) that we have already used in several evaluation studies [3, 6]. The memory simulator part of Limes (MemSim) simulates the memory subsystem sending requests to NetSim in order to simulate the advance of messages.

MemSim drives the network simulator using the following commands: *InitNetSim* (starts the simulation process), *InsertRequest* (injects a new message into the network), *Simulate* (simulates from the last simulated time to the actual simulation time), *EndNetSim* (ends the simulation and collects statistics). On the other hand, NetSim issues the following commands to MemSim: *MessageArrived* (a message has just arrived at its destination node), *TimeSimulated* (the requested time has just been simulated).

The memory simulator controls the applications threads. When it needs to send a request into the network, the involved thread is stalled until all the messages due to the request has been completed.

3 Performance evaluation

In this section we present some simulation results for a DSM system with a NCC-NUMA memory model [7]. Data distribution and process allocation are crucial in this model. In fact, some SPLASH-2 [1] applications recall that good distribution has great impact on performance. In addition, each process should be assigned to proper processor in order to improve data locality. The best data distribution and process allocation strategies have been used in the evaluation.

The interconnection network is a 64-node wormhole switched bidirectional *k-ary 2-cube*. Three routing algorithms have been used: deterministic, partially adaptive and fully adaptive. These routing algorithms were proposed in [3, 2]. The PRAM (Perfect RAM) model is also evaluated for comparison purposes. In this model, each request lasts only one cycle. Thus, we will use PRAM results as an approximation of an ideal memory subsystem and interconnection network.

Different complexity problems for OCEAN and FFT applications (SPLASH-2) are simulated. Each process is assigned to one single processor.

Performance evaluation is mainly done by means of application execution time. Taking into account that initialization phase can not be paralellized, we will not include it in the evaluation. This approach is also used in [1].

3.1 Simulation results

Figure 1 shows the execution time for the OCEAN application. The partially adaptive routing algorithm improves performance by almost a 40 % with respect to the deterministic one. The fully adaptive routing algorithm outperforms both the deterministic and partially adaptive one, reducing execution time by 70 %

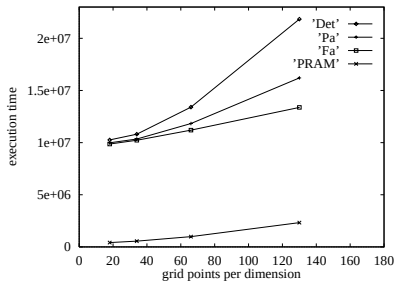

Fig. 1. OCEAN execution time.

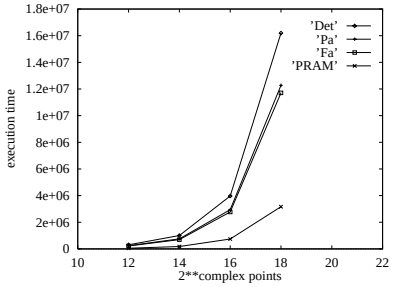

Fig. 2. FFT execution time.

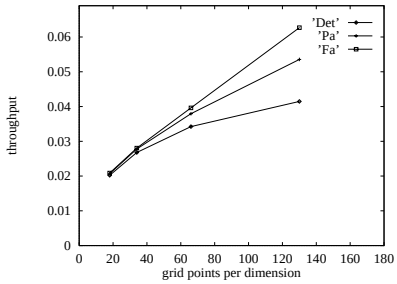

Fig. 3. Throughput for OCEAN.

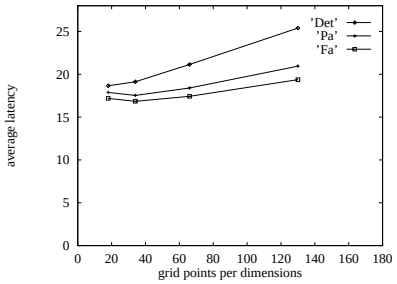

Fig. 4. Average latency for OCEAN.

and 20 % respectively. However, the execution time achieved by the fully adaptive routing algorithm is still far from the PRAM model (by a factor of 7, approx.). Remember that the memory model considered is highly inefficient, as it does not cache remote requests.

Figure 2 shows the execution time for the FFT application. Partially adaptive reduces execution time by a 30 % over deterministic routing. Although fully adaptive routing slightly improves partially adaptive routing, the improvement is lower than in OCEAN. This behavior is due to the fact that the FFT application does not generate so many traffic as OCEAN does, so that the greater routing flexibility offered by fully adaptive routing is useless.

As figure 3 shows, OCEAN application generates very low network traffic rate. The maximum network throughput is reached with the fully adaptive routing algorithm at the highest complexity (0.06 flits/node/cycle). As known, the network bisection bandwidth limit is more than one order of magnitude higher. It is important to note that only a single process is assigned to each network node and a sequential consistency memory model has been assumed. Thus, global message injection is limited to 64 (the number of nodes) messages at a time. Figure 4 shows differences between the latencies achieved by each of the routing algorithms for OCEAN application. When using adaptive routing, the improvement on the application execution time is given by the combination of lower contention and higher network bandwidth. The analysis of throughput and latency for FFT (not shown) leads to similar results.

Performance improvement of adaptive routing even with the low traffic generated by these applications and analyzed memory model, lead us to expect important improvements of the execution time from other applications that make a more intensive use of the network. Also, more efficient memory models (CC-NUMA, COMA [7]) may demand more network traffic.

4 Conclusions

In this paper we have proposed an execution-driven network simulator that allows the evaluation of DSM's interconnection network using real application workloads. This tool can also be used to analyze the impact of memory subsystem on performance. As a consequence, both subsystems can be jointly evaluated in order to achieve the best overall performance. As an example, we have implemented the NCC-NUMA memory model and some applications have been executed using different network routing algorithms. The results show that adaptive routing improves performance over deterministic routing by a 30 %.

As future work we plan to evaluate interconnection network performance using more realistic memory models like CC-NUMA, COMA, Simple COMA and other memory consistency models. We are also interested in evaluating irregular interconnection networks for Networks of Workstations.

References

1. Steven Cameron Woo et al., "The SPLASH-2 Programs: Characterization and Methodological Considerations," *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pp 24-36, June 1995.
2. W.J. Dally and C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, May 1987.
3. J. Duato and P. López, "Performance evaluation of adaptive routing algorithms for k-ary n-cubes," *Parallel Computer Routing and Communication*, K. Bolding and L. Snyder (ed.), Springer-Verlag, pp. 45–59, 1994.
4. Stephen Goldschmidt, "Simulation of Multiprocessors: Accuracy and Performance," *Ph.D. Thesis, Stanford University*, June 1993.
5. Davor Magdic, "Limes: A Multiprocessor Simulation Environment," *TCCA Newsletter*, pp 68-71, March 1997.
6. M.P. Malumbres, J. Duato and J. Torrellas, "An Efficient Implementation of Tree-Based Multicast Routing for Distributed Shared-Memory Multiprocessors," in *Proc. of the eighth IEEE Symp. on Parallel and Distributed Processing*, pp. 186-189, October 1996.
7. J. Protic, I. Tartalja, V. Milutinovic, "Distributed Shared Memory: Concepts and Systems", *IEEE Parallel and Distributed Technology*, Vol.5, No 1, Summer 1996.
8. F. Silla, M. P. Malumbres, J. Duato, D. Dai and D. K. Panda, "Impact of Adaptivity on the Behavior of Networks of Workstations under Bursty Traffic", submitted to *International Conference on Parallel Processing 1998*.
9. D. Thiebaut, J.L. Wolf, H. S. Stone, "Synthetic Traces for Trace-Driven Simulation of Cache Memories", *IEEE Transactions on Computers*, vol. 41, april 1992.