CONSERVATIVE PARALLEL SIMULATION OF A MESSAGE-PASSING NETWORK: A PERFORMANCE STUDY

José Miguel and Agustín Arruabarrena
Dep. Arquitectura y Tecnología de Computadores
Universidad del País Vasco
Apdo. 649 20080 San Sebastián, Spain
acpmialj@si.ehu.es

Ramón Beivide
Departamento de Electrónica
Universidad de Cantabria
Av. de los Castros s/n 39005 Santander, Spain
beivide@ccucvx.unican.es

Keywords: parallel simulation, performance evaluation, parallel computers

ABSTRACT

In this paper we evaluate the behaviour of a conservative parallel simulator implemented on a network of Transputers, when used to simulate an interconnection network proposed as the communication mechanism of a multicomputer. After a set of experiments, we conclude that some parameters of the model (*size, load, message length*) have a great impact on the achieved performance. Additionally, some parameters of the simulatior itself may allow the user to run the simulation faster, namely *grain size* of the processes and ability to exploit the *lookahead* of the model.

INTRODUCTION

Our research group is mainly interested in the analysis and evaluation of interconnection networks for multicomputer systems, in order to be able to propose adequate cost/performance alternatives. The tools we use for the study of this kind of networks are analytical models and, especially, computer simulation. With this last tool we have to face the fact that, when the network to evaluate grows to thousands of nodes, the memory and execution time demands of the simulator grow accordingly, making the study extremely slow.

One of the possible ways to accelarate our simulations is to parallelize the simulators, taking advantage of the currently available parallel systems. Our research group operates a Transputer-based multicomputer, so we decided to port some simulations to this machine. Before doing so, an evaluation of the main parallel simulation alternatives was done (a summary of it can be found in (Miguel *et al.* 1994)). From this study we concluded that, for the kind of models we are interested in, and the computer system we use, the conservative approach to PDES (Parallel Discrete Event Simulation) is very promising. Throughout this paper, we assume that the reader is familiar with the main parallel simulation algorithms.

THE SIMULATOR AND THE MODEL

In our first analysis of parallel simulation techniques we used very simplistic models for our benchmarks: toroidal networks of FCFS servers, traversed by a fixed number of *jobs*. This model was simulated using four different simulators: sequential event-driven, sequential time-driven, parallel conservative event-driven (with deadlock avoidance via null messages) and parallel optimistic event-driven (Chandy and Misra 1979; Jefferson 1985; Fujimoto 1990). For this kind of models we found that speedups up to 10 could be obtained running the conservative simulator in a network of 16 Transputers. The performance of our optimistic simulator was not as good, maybe due to an inefficient implementation of the algorithm. More detailed information of this experiments can be found in (Miguel *et al.* 1993; Miguel *et al.* 1994).

After evaluating the behaviour of the simulators the next step was to work with a realistic model. We already had a fairly complete set of measurements of the characteristics of some interconnection networks, obtained with a sequential simulator (Arruabarrena 93). A small set of these well-known models were selected and tested with the conservative PDES, to measure its actual performance with a useful workload.

The details of the selected model can be found in (Arruabarrena 93). It is basically a torus network of message routing elements (*routers*). Each router is connected to four neighbours and to a processing element, via bi-directional channels. Messages are of a fixed length. The flow-control mechanism is *cut-through* and, for this reason, message queues are needed, which are associated to each output channel. Figure 1 represents one of the studied routers.

For this set of experiments, several improvements were made to the first version of the parallel simulator, mainly in the implementation of the event queues and in the way of mapping the physical elements (routers) of the model onto the logical processes (LPs) that constitute the simulator. A fairly complete description of the implementation of the simulator is available in (Miguel *et al.* 1995).

In this work we concentrate on the evaluation of the behaviour of the conservative parallel simulator, and on the parameters that affect its performance, measured as the achieved speedup. Some of the parameters whose influence has been measured relate to the model, and some others realate to the way the simulator is organized.

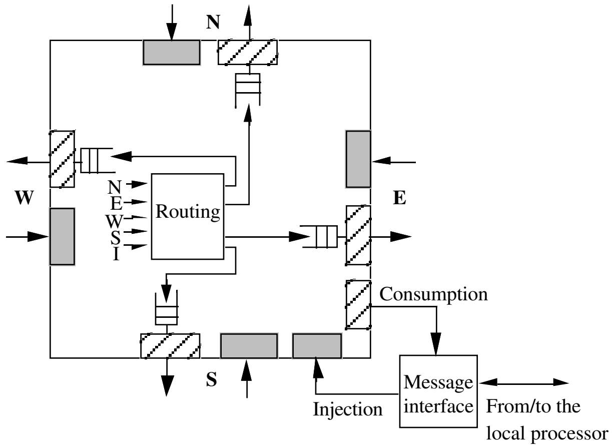

Figure 1. Model of message router.

DESCRIPTION OF THE EXPERIMENTS

We made a set of experiments varying the following parameters of the model:

- Message length. Values 4 and 32 were considered. It is measured in *flits* (flow-control digits). It is supposed that a flit advances from one router to another in one cycle (of simulated time).

- Load of the network: amount of information generated by the processing nodes, measured from 0 (none) to 100 (theoretical maximum). The maximum is calculated taking into account the bisection bandwidth (in flits) of the network, thus a given load level can be reached with many short messages or with few long messages.

- Size of the network. Most of the experiments were done with a reduced size network of 16x16 routers, but some results for 32x32 routers are also available.

Other parameters as the size of the queues, the communication patterns or the routing strategy were fixed, although they can be easily changed. The transit queues can hold 10 messages, while the injection queue has enough room for 4 messages. A node can produce messages for any other node; the selection of the destination node is made randomly. The (time) separation between messages produced in a given node is exponentially distributed, with a mean that depends on the load and the message lenght. The routing strategy is oblivious in order of dimension (first X, then Y), following a technique described in (Arruabarrena 93; Izu 94) to avoid deadlocks.

We ran the sequential and parallel simulators for 4000 cycles, using different number of processing elements (Transputers) and different router-to-LP mapping alternatives (grain size, see Figure 2). The potential benefits of exploiting the *lookahead* (Fujimoto 90) of the model were also tested. Now we describe when and how can the lookahead properties of this model be exploited.

When a LP sends null messages to its neighbours, the best thing to do is to maximize the value of the timestamp of the messages, allowing the largest possible clock advance at the neighbours. Unfortunately, sometimes it is impossible to do that. If a LP simulates only one router, then the behaviour of the simulation is quite *predictable*, and some *lookahead* can be extracted. Let us see the case that, at time t, a router has sent the header flit of a message through an output port; the simulator is able to *know* that no new header will be sent through that port at least until $t+L$, where L is the length of the message. The (time) difference between the current value of the clock and $t+L$ is the lookahead.

However, when a LP simulates an aggregate of routers, the computation of the lookahead is too expensive for the achieved result: most of the times the obtained value is 1, a minimum that can be assumed without any computation. For this reason, we only show the effect of exploiting the lookahead for the case of minimum grain size.

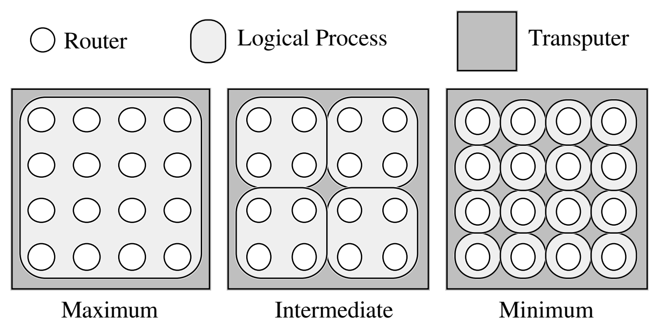

Figure 2. Grain size alternatives for 4x4 routers on a Transputer.

We also implemented a sequential, event-driven simulator, in order to have a reference to compare the achieved performance. Then, after measuring the execution time of a sequential and a parallel simulation of the same model, we could compute a speedup value. The sequential and parallel versions are as similar as possible but, in order to make fair comparisons, we included some optimizations in the sequential version that takea advantage of the use of a single memory space.

In the following subsections we present the results of a series of seven experiments performed with the simulators. An analysis of these results is done in the next section.

Experiment 1

We study a network of 16x16 nodes, managing 4-flit messages, for different values of load. The simulator ran over 4 Transputers. Several grain size alternatives have been implemented (see table below). The impact that the use of the lookahead of the model has on the performace of the simulator is also considered. Figure 3 shows the achieved speedups.

	LPs per Transp.	Routers per LP	lookahead
4Max	1	64	no
4ih	4	16	no
4il	16	4	no
4min	64	1	no
4minL	64	1	yes

As can be seen, the best results are obtained with intermediate grain sizes (**4ih**, **4il**). It is also clear that the performance significantly improves increasing the load. The attept of exploiting the lookahead of this model is dissapointing: the experiment **4minL** performs better than **4min** only for very high loads. At any rate, using intermediate grain sizes is always better, with or without lookahead.

The maximum achieved speedup, obtained in experiment **4il** at load 90, is 2.39.

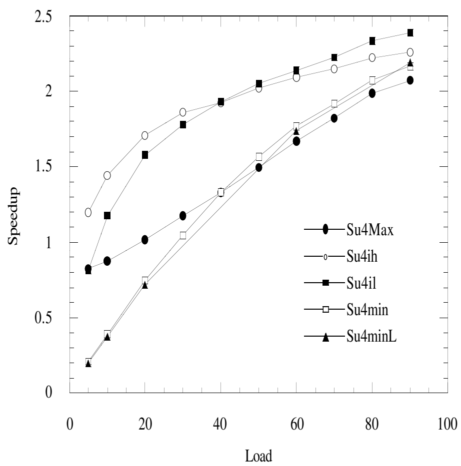

Figure 3. Results of Experiment 1.

Experiment 2

This experiment is similar to the previous one, but using 16 Transputers instead of 4. The alternatives of grain size decrease, and we don't try to exploit the lookahead of the model.

	LPs per Transp.	Routers per LP
16Max	1	16
16i	4	4
16min	16	1

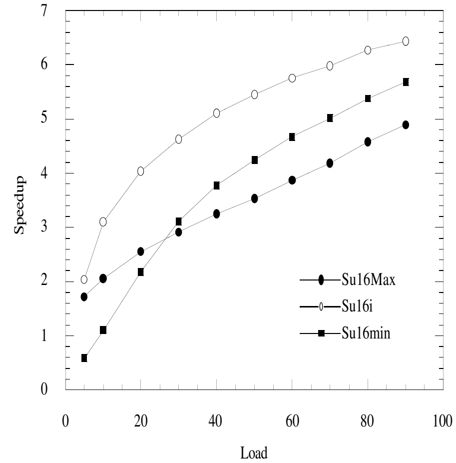

Figure 4. Results of Experiment 2.

Again, the best results are for intermediate grain sizes and highly loaded models. The maximum achieved speedup is 6.44.

Experiment 3

This experiment is like Experiment 1, but with messages of 32 flits. The most salient point of this set of experiments is that, in most cases, speedups are really poor: only **4ih** runs faster than the sequential simulator. At any rate, the execution times are always quite short. In this case the effect of exploiting the lookhaead is clearly noticeable, but again using intermediate grain size performs better. The maximum achieved speedup is 1.87.

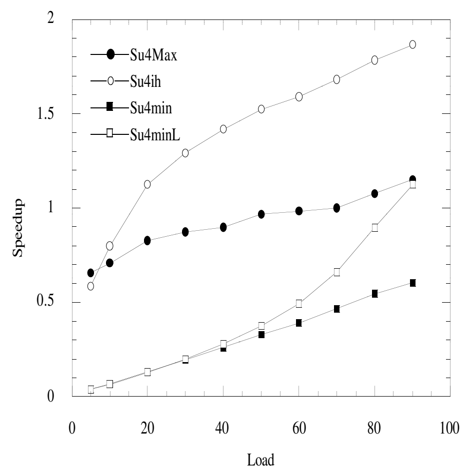

Figure 5. Results of Experiment 3.

Experiment 4

This experiment is simitar to experiment 2, but using 16 Transputers for the simulations. The comments about experiment 3 are valid for this one too, although the speedups are noticeably better. The maximum speedup is 4.17.

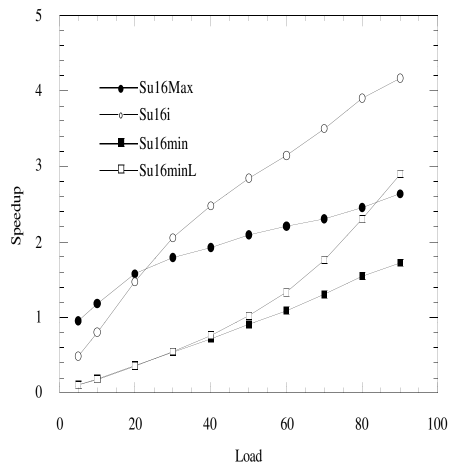

Figure 6. Results of Experiment 4.

Experiment 5

In this and the next experiments, we simulated bigger models: a network of 32x32 routers. Message lenght was 4 flits. We ran the simulator over 16 Transputers, considering only two cases of intermediate grain size (see below).

	LPs per Transp.	Routers per LP
ih	4	16
il	16	4

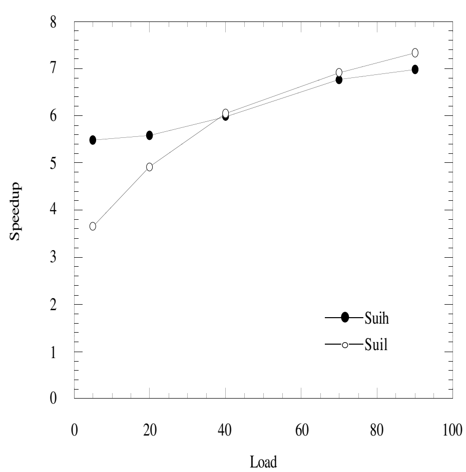

Figure 7. Results of Experiment 5.

This experiment can be considered an extension of experiment 2, using a bigger model. The speed gain is clearly noticeable, even better than the achieved for the 16x16 network. The maximum speedup is 7.33.

Experiment 6

This experiment is like the previous one, but for 32-flit messages. It can be seen as an extended version of experiment 4, as most of the parameters are the same, but the size of the simulated network. The peak speedup is 7.65.

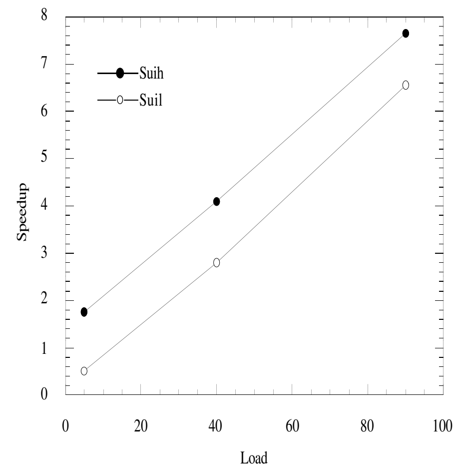

Figure 8. Results of Experiment 6.

Experiment 7

This experiment was designed to test how well the performance improves when adding more processors. We simulated a network of 24x24 routers, with load 50 and 4-flit messages, for 4000 cycles. We tested the simulator using 1 (sequential), 4, 9, 16 and 25 Transputers. In the last case, the simulated network was slightly incremented (25x25 routers). Intermediate values of grain size were used in all but the last case, where the maximum grain size was used. The results are plotted in Figure 9.

ANALYSIS OF THE RESULTS

Just looking at the figures, it is clear that the following parameters of the model affect the performance of the simulation: load, message length and network size. The best situation (for the simulator) arises with big and highly loaded systems that interchange short messages. Interestingly enough, this is the worst possible scenario for the sequential simulator. When the model has the opposite characteristics, the performance (speedup) is not brilliant, but the actual execution time is not extremely long.

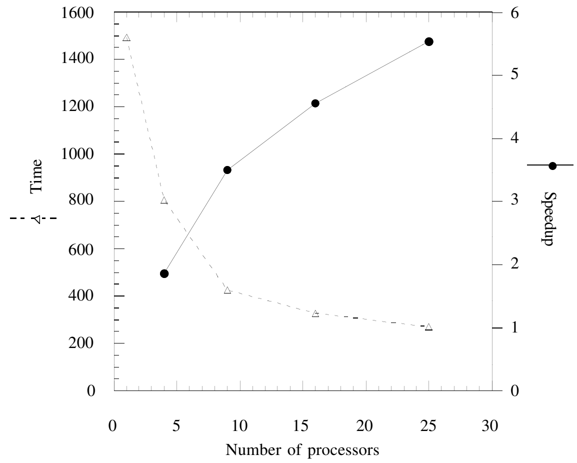

Figure 9. Results of Experiment 7.

The reason for this behaviour can be found in the way LPs synchronize in a conservative simulator. Any of the mentioned parameters affects the number of "useful" messages (that is, non-null messages) managed by the simulator. An increment in the number of these messages means that the LPs have more opportunities to synchronize, while doing useful computation. Null messages are needed less often, as LPs don't block frequently. We can say that there is a high degree of "natural" synchronization. When there are only a few useful messages to process, LPs block often, and null messages are needed to maintain the LPs' clocks updated. Then LPs spend most of their (real) time blocked or processing null messages, that is, they are not able to make progress in the simulation.

Previous experiments reported in (Miguel *et al.* 1993, Miguel *et al.* 1994), led us to think that the performance of a conservative parallel simulator running on a network of Transputers, where communication overhead is very big (Izu 1994), would be better when bigger were the grain size of the simulator. This way, the computation/communication ratio would be more balanced. If, looking at the figures, we compare maximum vs. minimum grain size, we can see that coarse grain simulation is more effective than fine grain simulation for low and medium loaded systems, because a small number of LPs synchronizing with null messages results in a lower overhead. If now we compare with the intermediate grain size alternatives, it is clear that this is *the best* alternative for intermediate and high loads. Only for very low loaded systems the maximum grain size gives sometimes better performance. This behaviour can be explained analyzing the time used for syncronization in the simulator:

- With maximum grain size, a LP can evolve quite autonomously, due to the big number of interactions between the routers assigned to that particular LP. Nevertheless, sometimes the LP has to co-ordinate with the others, and that leads the LP to send null messages and then block. This is extremely inefficient because, as the LP is the only *user* of the Transputer, if it blocks, the freed CPU power is wasted.
- With minimum grain size, there is no problem if a LP blocks: plenty of others are awaiting to get the CPU. Here the problem is that, the bigger the number of LPs, the bigger the number of null messages needed to keep the system synchronized and, what makes things even worse, as every fine-grain LP has very little work to do, it blocks very often.
- With intermediate values of grain size we try to find a balance: there are few LPs, each one with enough work to do, so null messages are not needed very often. In addition to that, as several LPs share a Transputer, the probability of having a stopped CPU decreases.

As a conclusion, we can confirm that, in general, coarse grain simulations are faster than fine-grain simulations, but it must be taken into account that is very efficient to share a processor among several LPs, to avoid stopped processors and make the maximum usage of the available CPU power. This is especially true for the Transputer, where the communication operations are blocking and the context swhitches are very fast.

From the description of how lookahead is computed, it should be clear that the biggest values are obtained when (1) messages are long and (2) the LP has recently interacted with its neighbours. The first point appears clearly comparing experiments 1 and 3: while for 4-flit messages **4min** and **4minL** give nearly the same results, when the lenght is 32 flits **4minL** is clearly better than **4min**. The second point can be seen in experiments 3 and 4: in highly loaded systems, the LPs interact often and, if they have to block, the computation of the timestamp of the null messages can take advantage of the knowledge of which ports have recently sent header flits. An any rate, as lookahead only can be effectively exploited for minimum grain size, and this is not the best situation for the simulator, for this model we don't see any advantage of using it.

From the complete set of experiments, and especially from experiment 7, it can be seen that the performance of the parallel simulator scales quite well with the number of processors, although not linearly. There are, though, some factor that can help to understand why the speedup gain is not perfectly linear:

- Our way of distributing the workload among processors and processes, using squares, does not allow to find the *optimum* grain size value. For example, the 25x25 network, when simulated over 5x5 processors, only allows for maximum or minimum grain size.

- The processors used in our experiments were not all identical. In most cases, T805 running at 30 MHz were used but, as we only have 14 processors of this kind, in the experiments involving 16 or 25 processors some 20 MHz devices were needed. Those processors are a bottleneck, imposing their rithm to the others, and for this reason the resulting speedups are not as good as they should be.

- As the problem does not increase with the number of resources, the processors are not fully utilized. When the number of processors (and of LPs) increases, the probability of having a blocked LP increases too, so some processing power is wasted and more null messages are needed. To confirm this assertion, we have represented the ratio of useful messages to total messages managed by the simulator in Experiment 7. It can be seen how this ratio reduces when the number of processors is increased (Figure 10).

It should be noted that the chosen parameters for experiment 7 do not constitute the best possible scenario for the parallel simulator, neither the worst for the sequential one. With a higher load the results are much more favourable for the parallel simulator.

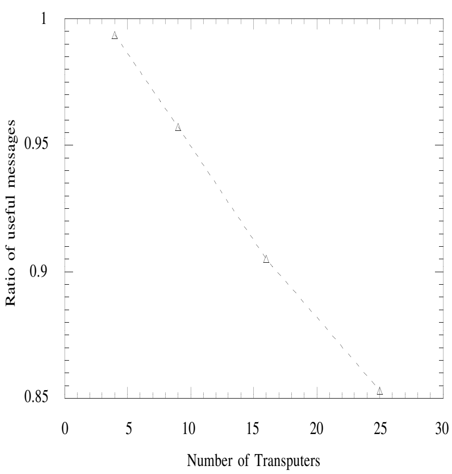

Figure 10. Ratio of useful messages for the different number of Transputers used in Experiment 7.

CONCLUSIONS

From the complete study, the following conclusions were drawn out:

- The performance of the conservative PDES is better when bigger is the message interchange between the LPs. This happens when the simulated network increases, when the load increases and when the message lenght decreases. The reason is that, under these circumstances, the LPs are able to synchronize while doing useful computation, and null messages are needed less often.

- The mapping of routers onto LPs has a major influence on the achieved speedups. The best results are obtained when the number of LPs assigned to a Transputer is small, but bigger than one, in such a way that, when a LP is blocked, another one is ready to take the CPU.

- The lookahead can not be effectively used for this model, because it can be exploited only when a few routers are mapped onto one LP, and this is incompatible with the previous recommendation.

Our experience with conservative PDES is quite encouraging, because it allows us to accelerate our simulations and to study bigger problems. Achieved speedups are not always brilliant, but they are best precisely when the sequential simulator behaves worst.

REFERENCES

Arruabarrena A. 1993. *Análisis y Evaluación de Sistemas de Interconexión para Procesadores Masivamente Paralelos*. PhD Dissertation, Dept. of Computer Architecture and Technology, University of the Basque Country.

Chandy, K.M. and Misra, J., 1979. “Distributed Simulation: a Case Study in Design and Verification of Distributed Programs” *IEEE Transactions on Software Engineering SE-5*, no. 5 (Sept), 440—452.

Fujimoto, R.M., 1990. “Parallel Discrete Event Simulation”. *Comm. of the ACM 33*, no. 10 (Oct.) 31—53.

Izu, C. 1994. *Análisis, Evaluación y Aportaciones al Diseño de Arquitecturas para Encaminadores de Mensajes*. PhD Dissertation, Dept. of Computer Architecture and Technology, University of the Basque Country.

Jefferson, D.R., 1985. “Virtual Time”. *ACM Trans. on Programming Languages and Systems 7*, no. 3 (July).

Miguel, J.; Arruabarrena A.; and Beivide, R. 1993. “Conservative Parallel Discrete Event Simulation in a Transputer-Based Multicomputer”. In *Transputer Applications and Systems ‘93*. R. Grebe et al. (Eds.) IOS Press, 636—650.

Miguel, J.; Arruabarrena A.; and Beivide, R. 1994. “Conservative and Optimistic Distributed Simulation in Massively Parallel Computers: a Comparative Study.” In *Proceedings of the Conference on Massively Parallel Computing Systems MPCS’94* (Ischia, Italy, Sept.).

Miguel, J.; Arruabarrena, A.; and Beivide, R. 1995. “Parallel Simulation of Message Routing Networks”. In *Proceedings 3rd. Euromicro Workshop on Parallel and Distributed Processing PDP’95* (San Remo, Italy, Jan.).