## Comparison of Parallel Simulation Techniques Parsytec Transputer Supercluster / SLIM

### Part I

### Introduction

Following the recent announcement in EUROSIM SNE 10 of the proposed series on comparisons of parallel simulation methods, some work has been undertaken at Glasgow University to port a continuous system simulation tool (SLIM) to the SIMD/MIMD environment provided by a Parsytec Supercluster. This is a T800 transputer based parallel system with Parix operating system. This note provides a brief account of the way in which the SLIM simulation language has been implemented on the Parsytec system using a master-slave approach. Results are presented for the Monte Carlo study that forms the first test example for the comparison series. Part II will report on other solutions on the Parsytec Supercluster.

### SLIM

The Simulation Language for Introductory Modeling (SLIM) is a continuous system simulation tool developed within the Department of Electronics and Electrical Engineering at the University of Glasgow to provide students with an introduction to the main concepts of simulation languages before they attempt to use industry-standard software such as ACSL. It was developed for a PC/DOS based environment but has been ported successfully to other platforms such as VAX/VMS and SUN/SunOS.

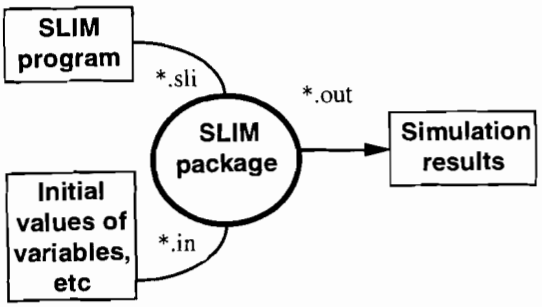


Figure 1: Context diagram of SLIM package

The facilities provided within SLIM are a subset of the CSSL'67 recommendations and the syntax of this language therefore appears to the user to be a reduced version of that found in many other widely used simulation tools. Figure 1 shows the operations carried out when a SLIM program is run. The process is essentially interpretative. Two files are normally provided by the user. One of this is the simulation application program written in the SLIM language, while the second is a file that may involve values of parameters and initial values of state variables. The SLIM code for the Monte Carlo study is very simple and is not included in this note due to space limitations. It can be made available to anyone interested on request to the authors.

### The Master - Slave approach

Since we did not wish to make significant changes in the structure of SLIM it was decided to adopt an approach involving a master program running on one processor that would distribute tasks among a number of slaves. In this example the slave programs are complete SLIM programs, with only very minor changes compared with the original version of the software. This master-slave approach is well suited for the solution of simulation problems involving experiments on a number of similar models that differ in terms of parameters or initial conditions.

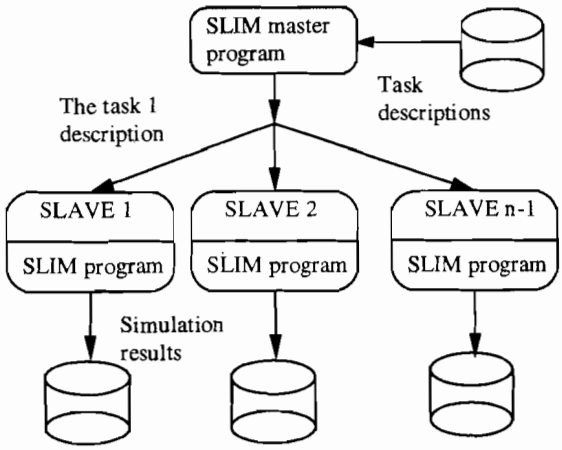


Figure 2: Master-slave model of SLIM package

The preparation of the SLIM master program and the changes within the SLIM slave program took about three man weeks to complete for someone without previous experience of the Parsytec system. Preparation of the SLIM application program for Monte Carlo Study took approximately five minutes.

### Task 1) - The Simulation process

Our simulations took place on the array of transputers (T800/25Mhz/4MB RAM) connected to SUN IPX workstation as a front end computer. We executed our test problem on various numbers of transputers. The results are depicted in Tab. 1.

Due to limitations of the operating system and/or this specific FORTRAN compiler, we could execute the simulation on a maximum of 17 processors. As can be seen from Table 1 the speedup of the task execution

is almost linearly dependent on the number of participating processors and for 17 processors it is nearly 14.5 times. In a configuration up to 8 transputers the difference between theoretical linear speedup relative to the result for two processors and actual results is almost zero, but in configurations 16 and 17 the relative error is 7-9%. Our assumption is that when in configurations with 10 or more transputers, the front-end system forms a bottleneck, because of the large amount of calculated data transferred from a transputer array to the main memory of the front-end processor.

| Proc # | Expected Time[sec] | Elapsed Time[sec] | Diff % | Speed-up factor f |
|---|---|---|---|---|
| 1 | --- | 1708.59 | --- | --- |
| 2 | --- | 1719.86 | --- | --- |
| 3 | 859.93 | 861.93 | 0.23% | 1.99 |
| 4 | 574.43 | 577.27 | 0.49% | 2.98 |
| 5 | 429.97 | 435.21 | 1.22% | 3.95 |
| 6 | 343.97 | 349.57 | 1.63% | 4.91 |
| 7 | 287.22 | 293.90 | 2.33% | 5.85 |
| 8 | 245.94 | 254.58 | 3.51% | 6.74 |
| 9 | 214.98 | 222.54 | 3.52% | 7.71 |
| 16 | 115.23 | 123.63 | 7.29% | 13.87 |
| 17 | 108.35 | 117.54 | 8.48% | 14.57 |

Table 1: Elapsed time on different configurations (speed-up factor added by the editors)

**How to improve results**

The SLIM package and the system parameters could be tuned up in several ways to achieve even better results. We could change the format of an output file from the ASCII to binary, since this would reduce the amount of data transferred from the transputer array to the front-end computer and thus raise the level of saturation of the front-end. In similar way we could change the size of disk cache, to get a better response from a front-end system, but in that way we can gain only a small profit that depends upon physical computer configuration and the model executed.

A much better, but also more demanding task, would be to change the syntax of the SLIM language and consequently the structure of the SLIM package itself, so it can support partially parallel execution of sequential SLIM programs. We can achieve parallelism in two ways: internally and externally. Internally, hidden from users' eyes we could change the way in which the program is executed. Externally we could change syntax of the language, so it would be able to support several transputers in parallel.

*Matjaz Ostroversnik, Prof. Dr. David Murray-Smith, Department of Electronics and Electrical Engineering, University of Glasgow, Glasgow G128QQ, Scotland, U.K.*