DS-LED: A DISTRIBUTED APPLICATION FOR THE SIMPLEX METHOD

J. Herrero*, A. Paccanaro*†

*Laboratorio de Electrónica Digital
Universidad Católica "Nuestra Señora de la de Asunción"
Asunción, Paraguay
Tel: +595-21-334650
Fax: +595-21-310587
e-mail: *apaccana@galactica.it*

†Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Milano, Italy

ABSTRACT

A parallel version of the simplex method has been developed and then implemented in a distributed system. This paper presents the distributed application which includes a dynamical load balancing algorithm and is fault tolerant. Finally the results of some tests are presented which show the great advantages in the response time of the parallel algorithm with respect to the linear one.

1. A PRELIMINARY STUDY OF THE PROBLEM

DS-LED implements a variation of the linear programming algorithm (simplex Method) in a distributed environment. The idea of this project was to take advantage from parallel processing in order to optimize the resolution time for this problem.
The reader is supposed to be well acquainted with the main concepts and terminology of the simplex method and distributed systems. For further readings see Prawda[1982], Shivaratri et al.[1992], Bowen et al.[1992], Shieh et al.[1990].

A study of the response time of the linear algorithm for the simplex method was carried out. Experiments were executed on a MicroVAX 3100 machine with Ultrix operating system. Fig.1 illustrates some results. It can be seen that the operation of search of the column vector which has to enter the base - from now on this operation will be called entry - is a very expensive operation when the number of columns is large.

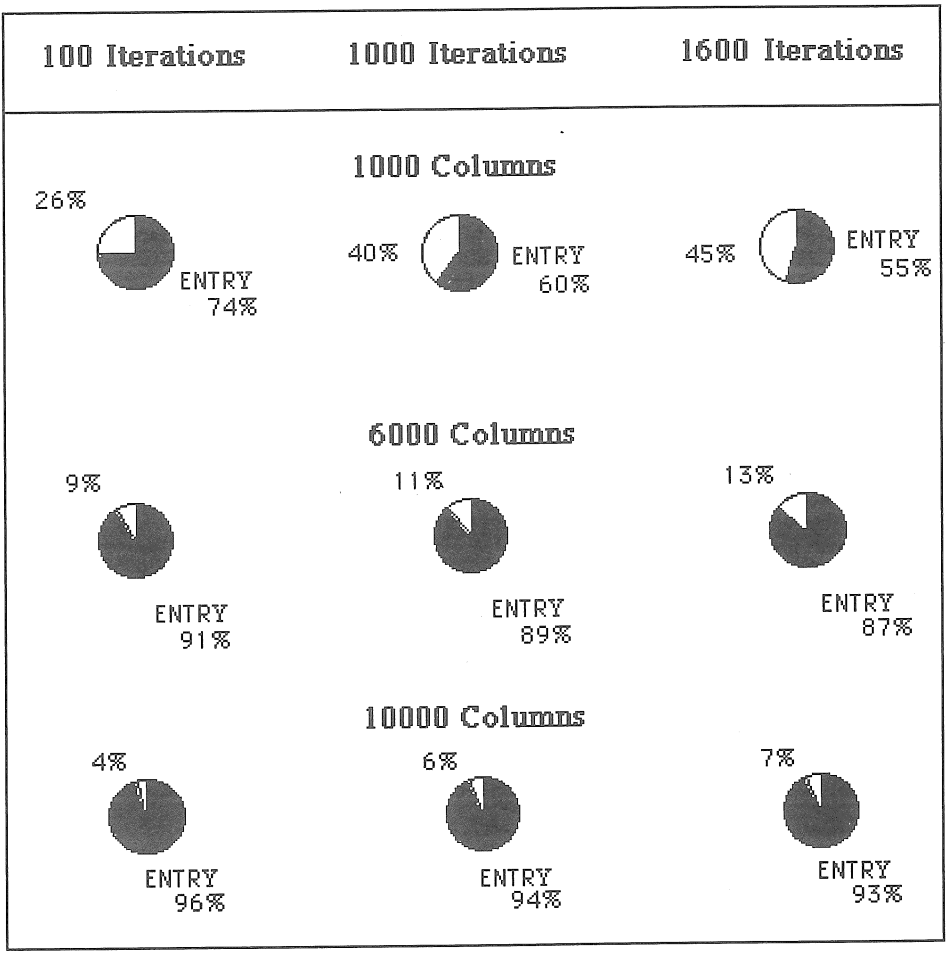

Fig. 1

It was noticed that the entry operation could be parallelized since the operations which have to be carried out in order to find such a vector can be executed in an independent fashion over each column. Therefore it made sense to try which performances could have been obtained with a

parallel version of the algorithm, where different machines could have carried out the entry operation simultaneously and independently.
For this purpose a distributed application called DS-LED has been designed and implemented, and is presented in this paper.

2. SYSTEM IMPLEMENTATION

The project was carried out at the LED -Laboratorio de Electronica Digital - of the Catholic University of Asuncion, Paraguay.
The system was developed using a MicroVAX 3100 with Ultrix operating system with 16 Mb RAM, and 3 PC with SCO Unix operating system, of which 2 were 80486s at 66 MHz and 33 MHz with 8 Mb and 4 Mb RAM respectively and one was a 80386 at 40 MHz with 8Mb RAM; the programming language used was C; communication between processes was realized through sockets interface. The various machine communicated through an Ethernet Net with the TCP-IP communication protocol.

The parallel system implemented is constituted of 2 kind of processes, called client and server respectively.

The number of servers can be greater than one and the various servers cooperate in order to realize the entry operation. To do this the matrix of the elements which are not in base will be partitioned in various disjunct sets, and each one of them will be given to a distinct server.

On the other hand there will exist only one client which will have to carry out the non-parallelizable part of the algorithm.
Therefore the client will have to keep a complete copy of all data.

All the processes will be run on independent machines in order to take full advantage of the parallel processing of the data.
Main features of the application are a load balancing and fault tolerance algorithm, which are presented in the following sections.

a) Load Distribution

In order to obtain better response time the amount of columns which has to be distributed to each server must take into consideration both the speed of the machine on which the server is running and the number of processes which are already present on it.

Moreover since the amount of processes being executed by the various servers may vary during time, a dynamical load balancing should be carried out in order to obtain better performances.
DS-LED implements such load balancing in the following way.
At the beginning of the execution the client waits from each server for a message containing information about the speed of the machine and the number of processes executing on it. Then it can distribute the columns among the various machines according to these measures. It is necessary to point out that the information relative to the speed of each processor is a static coefficient, empirically assigned "a priori" to each processor. Nevertheless it has proven to be extremely important in order to obtain a better response time from the parallel algorithm. Fig. 2 shows the difference in response time obtained introducing such coefficients.

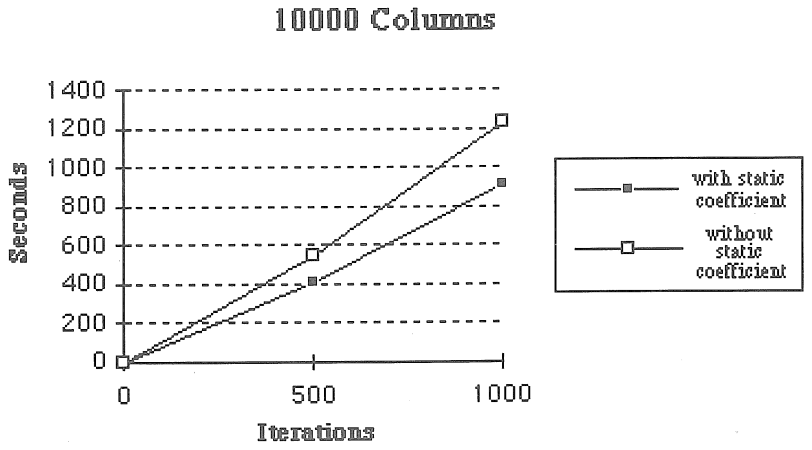

Fig. 2

During execution the client at each iteration of the algorithm measures the response time of each server and accordingly to it, periodically redistributes some columns among the various processor, if necessary.
Fig. 3 shows the performances of the parallel algorithm with dynamical load balancing as compared to the same algorithm without it. The size of the problem was of 10000 columns and there were no variations on the load of each server. The dynamical load balancing algorithm transferred columns among servers between the 700th and 800th iteration.

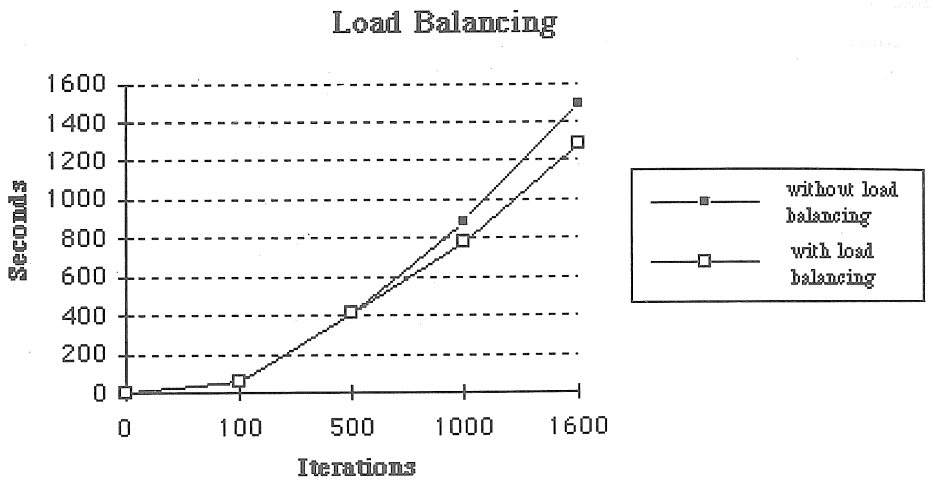

Fig. 3

b) Fault Tolerance

The fault tolerance system implemented in DS-LED deals with the case in which a machine is completely lost. This implies the loss of all data and processes which are being executed on it. Moreover the fault of a machine can cause problems to the whole system if other machines are waiting for some message from it. In order for the system to recover from such a fault it is necessary to (Knight [1987]):

i) detect the fault
ii) detect the damages due to the fault
iii) select and execute an appropriate answer

The actual version of DS-LED can only deal with a complete loss of one or more servers; it can not recover the loss of the client.
Let us see how each of the step mentioned above are implemented.

i) fault detection

In order to detect that a fault occurred to a machine belonging to the system 2 processes were implemented: the Client Fault Tolerant process (CFT) and the Server Fault Tolerance process (SFT).
The SFT process periodically sends a message to the CFT process, which immediately sends back an acknowledgement message. This message is used by the SFT in order to be sure that the

machine where the client is being executed is still alive. On the other hand the CFT process keeps a table containing the addresses of the machines where the servers are executed which is updated each time that a message is received from a SFT process. Periodically the CFT process checks this table in order to see if a server process died. Once a fault has been detected the CFT and SFT processes communicate the problem to the client and server processes respectively.

ii) damage detection

In order to determine the damages, the client process keeps a table of how the columns were distributed over the various servers. In this way he can know which columns were lost due to the fault of one or more servers. With this information it is possible to decide which action to take in order to recover the fault.

On the other hand the server does not need to keep a similar table since, as mentioned before, this version of DS-LED can not recover the loss of a client.

iii) selection and execution of an appropriate answer

Case 1 - fault of the client

If the client dies, the server process will be waiting indefinitely for a message from it. In order to avoid this problem when the SFT detects the fault of the client it sends to the server process a message, it can select which action to take. In this version of the system the only possible answer will be to abort.

Case 2 - fault of the server

If a fault occurs to one or more servers, the client has 2 options: to abort or to redistribute the columns which had been given to the dead server processes.

The abort option is executed when there are no more server processes available. If some server is still alive the client process will redistribute all the columns belonging to the dead processes to the servers which remain alive.

3. SYSTEM EVALUATION

The system was tested in order to measure the advantage in response time obtained with the parallel algorithm with respect to the linear one.

The performance of the parallel algorithm was analyzed while varying the number of columns, since the performance of the linear algorithm depended on the number of columns in the problem.

In the following (fig. 4, 5 and 6) some diagrams are shown representing the time responses obtained for 1000, 4000 and 10000 columns. Response time were taken after 100, 1000, and 1600 iterations.

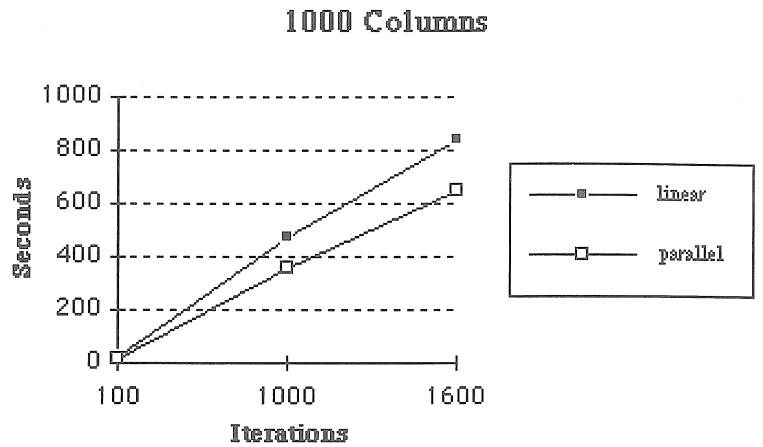

Fig. 4

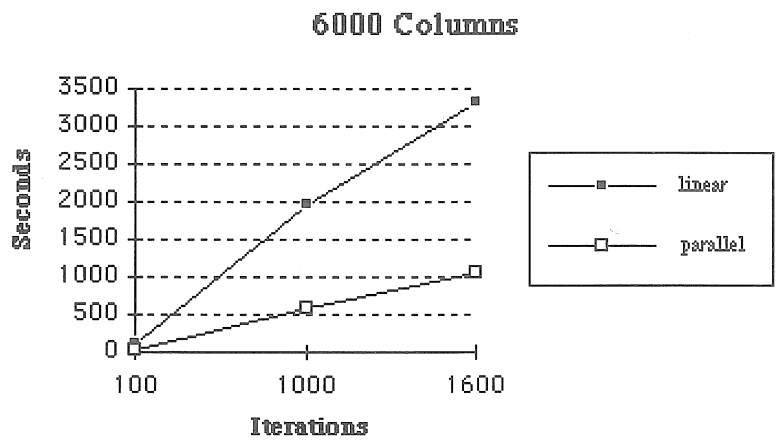

Fig. 5

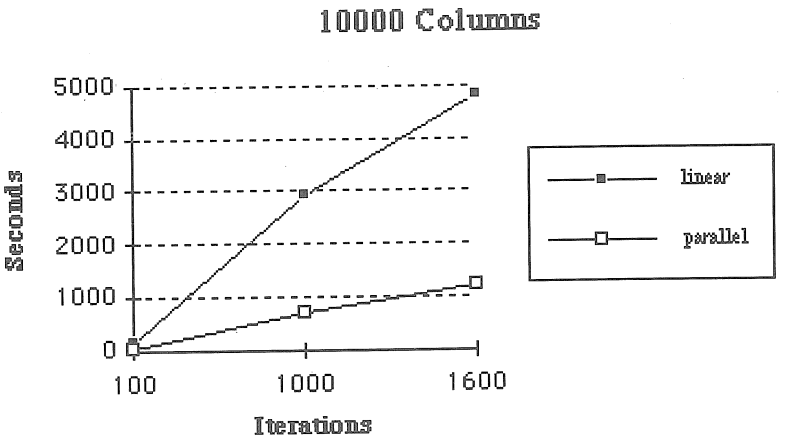

Fig. 6

The advantage obtained using a parallel system are greater for larger sizes of the problem. This can also be seen analyzing the numerical values of the tests presented above which are shown in the following table:

	500 Columns	1000 Columns	2000 Columns	4000 Columns	6000 Columns	8000 Columns	10000 Columns
Linear	635	839	1193	2398	3323	3822	4818
Parallel	637	650	616	901	1056	1158	1203

Table 1

An interesting measure of the advantage in response time obtained with the parallel algorithm with respect to the linear one is given by an the acceleration index. This index is calculated through the following formula:

$$\textbf{acceleration} = \frac{Tl}{Tp}$$

where t_l and t_p are the time consumed by the linear and by the parallel algorithm respectively Fig. 7 shows a graph of the acceleration obtained for the tests presented above. The numerical values were taken after 1600 iterations, while varying the size of the problem.

It is important to notice that the acceleration is greater when the size of the problem increases and it reaches the value of 4 for 10000 columns.

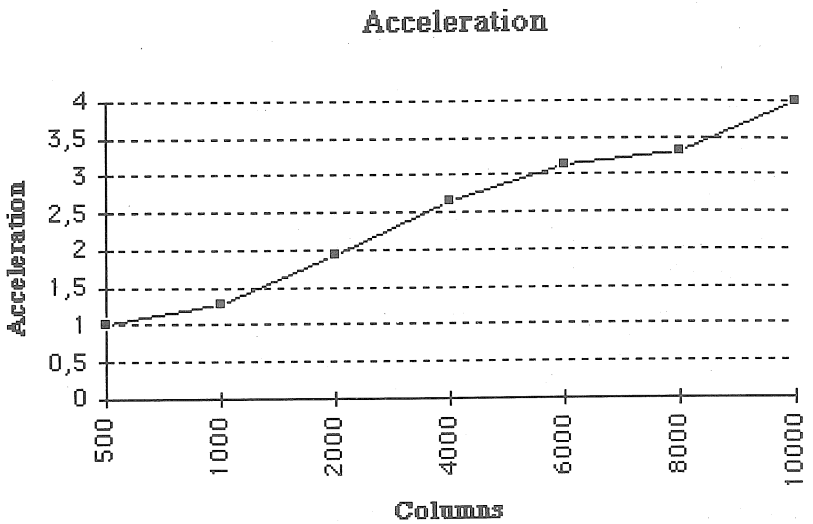

Fig. 7

The increase in the acceleration which is obtained when increasing the size of the problem is due to the fact that the serial section of the program remains constant while the parallelizable section increases when the number of columns increases.

Finally, Fig. 8 presents a comparison of the time used by the 2 algorithms to iterate 1600 times on problems of different size. It is important to point out that the increase in the execution time is much slower for the parallel algorithm.

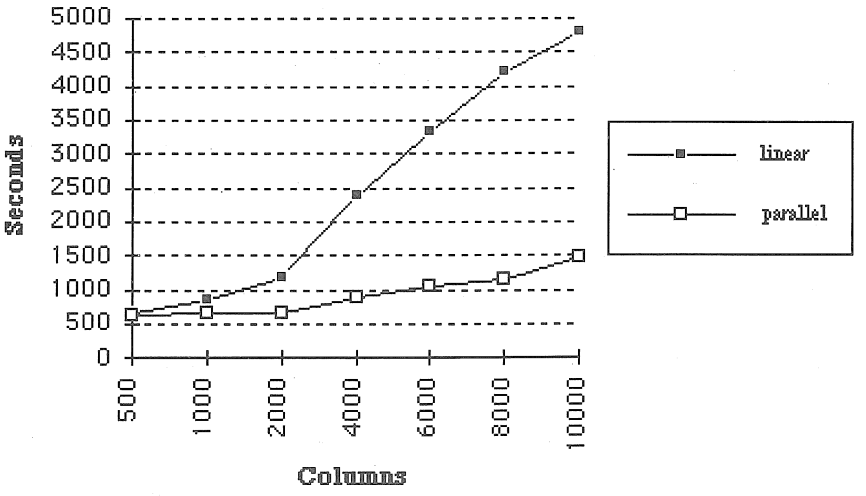

Fig. 8

4. CONCLUSIONS AND FURTHER DEVELOPMENTS

We presented the implementation on a distributed system of a parallel version of the simplex method. The application includes a dynamical load balancing algorithm and is fault tolerant with respect to the loss of one or more servers. The results presented show a great advantage in the response time of the parallel algorithm with respect to the linear one. We are now thinking of including a fault tolerant algorithm which would be able to deal with the loss of the client.

REFERENCES

Niranjan G. Shivaratri, Phillip Krueger and Mukesh Singhal [1992], "Load Distributing for Locally Distributed Systems", COMPUTER, pp. 33-44, DECEMBER 1992.

N.S. Bowen, C.N.Nikolau y A.Ghaffour [1992], "On the Assigment Problem of Arbitrary Process Systems to Heterogeneus Disributed computer Systems", IEEE TRANS. ON COMPUTERS, VOL. 41, pp. 257-273, MARCH 1992.

J.C. Knight and J.I.A. Urguhart [1987], "On the Implementation and Use of ADA on foult-tolerant Distributed Systems", IEEE TRANS. ON SOFT. ENG, VOL SE-13, pp 553-563, MAY 1987.

Y.B. Shieh, D. Ghosal, P.R. Chintamaneni, and S.K. Tripathi [1990], "Modeling of Hierarchical Distributed Systems with Fault-Tolerance", IEEE TRANS. ON SOFT. ENG, VOL 16, pp. 444-457, APRIL 1990.

Juan Prawda [1982], "Métodos y Modelos de Investigación de Operaciones", Limusa, 1982.