AN EFFICIENT ADAPTIVE PREDICTIVE LOAD BALANCING METHOD FOR DISTRIBUTED SYSTEMS

ESQUIVEL, S., C. PEREYRA C, GALLARD R.
Proyecto UNSL-338403[1]
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - Local 106
5700 - San Luis - Argentina
E-mail:{esquivel, cpereyra, rgallard}@.unsl.edu.ar
legui@inter2.unsl.edu.ar
Teléfono: 54++ 652 20823. Fax : 54++ 652 30224

Abstract

When allocating processors to processes in a distributed system, load balancing is a main concern of designers. By its implementation, system performance can be enhanced by equally distributing the dynamically changing workload and consequently user expectation are improved through an additional reduction on mean response time. In this way, through process migration, a rational and equitable use of the system computational power is achieved, preventing degradation of system performance due to unbalanced work of processors.

This article presents an Adaptative Predictive Load Balancing Strategy (APLBS), a variation of Predictive Load Balancing Strategy (PLBS) reported elsewhere [1]. As PLBS, APLBS is a sender initiated, prediction-based strategy for load balancing. The predictive approach is based on estimates given by a *weighted exponential average* [12] of the load condition of each node in the system. The new approach tries to minimise traffic en the network selecting the most suitable subset of candidates to request migration and the novel aspect is that the size of this subset is adaptative with respect to the system workload**.** APLBS was contrasted against Random (R), PLBS and Flexible Load Sharing (FLS) [7] strategies on diverse scenarios where the load can be characterised as static or dynamic. A comparative analysis of mean response time, acceptance hit ratio and number of migration failures under each strategy is reported.

Keywords: Distributed systems, load balancing strategies, mean response time, acceptance hit ratio, migration failures.

[1] The Research Group is supported by the Universidad Nacional de San Luis and the ANPCYT (National Agency to Promote Science and Technology).

AN EFFICIENT ADAPTIVE PREDICTIVE LOAD BALANCING METHOD FOR DISTRIBUTED SYSTEMS

1. INTRODUCTION

A typical example of a Distributed System is formed by a set of processors interconnected by a Local Area Network (LAN). The model studied here consist of a set of homogeneous an independent nodes having their own processing resources and information storage. In such a system, users from different sites create autonomous processes, which sporadically need synchronisation to share critical resources.
A node workload embodies a set of local and, possibly, external demands on all or some of its resources. This global demand varies during the execution of processes and could lead the system to an unbalanced state; some of the nodes can be overloaded while others are underloaded and even idle.
In general, the load balancing strategies are clasified as [2]:

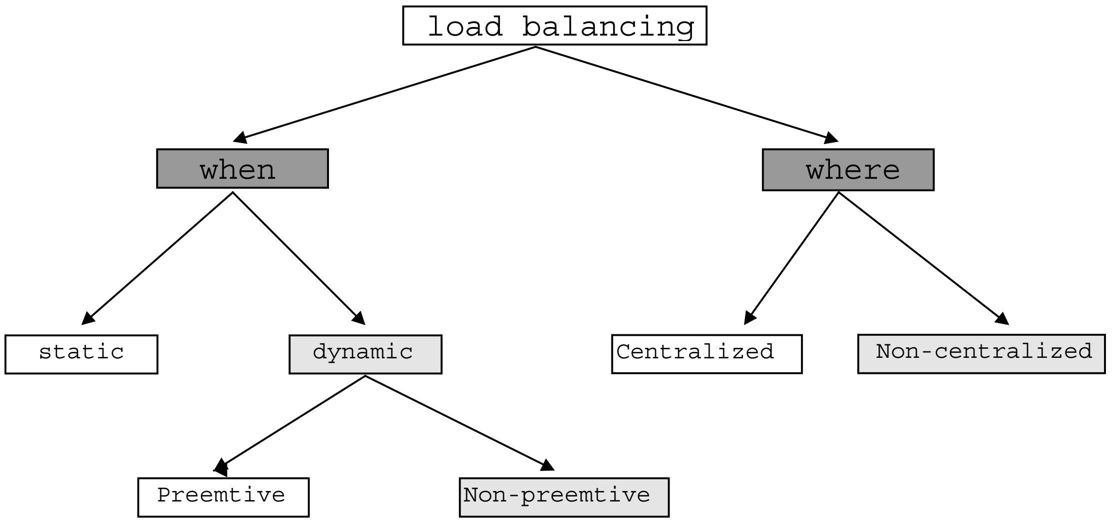


Fig.1: Load Balancing Strategies

The classification of the proposed Adaptative Predictive Load Balancing Strategy (APLBS), is characterized by the shaded squares in figure 1. APLBS is a *dynamic* load balancing strategy because decides while executing to which processor assign a process. It is *non preemptive* because once the processor is assigned the process must finish execution on that processor without any further migration. And finally, it is *decentralised* because decision making is distributed between network nodes.
Many research groups have been investigating load sharing and balancing [3],[5],[6], [7], [8], [11] and [13].
Also some approaches based on evolutionary computation achieved improved results [4], [9] for dynamic load balancing. In most cases the improvements were the consequence, of a lower number of migration requests from an overloaded node.
This paper presents APLBS, an strategy which by *guessing* the working load at each node attempts to predict the subset, of adaptive size, containing the candidates more prone to accept the migration request.

2. ADAPTIVE PREDICTIVE LOAD BALANCING DESIGN

Internal components and their interaction are depicted below

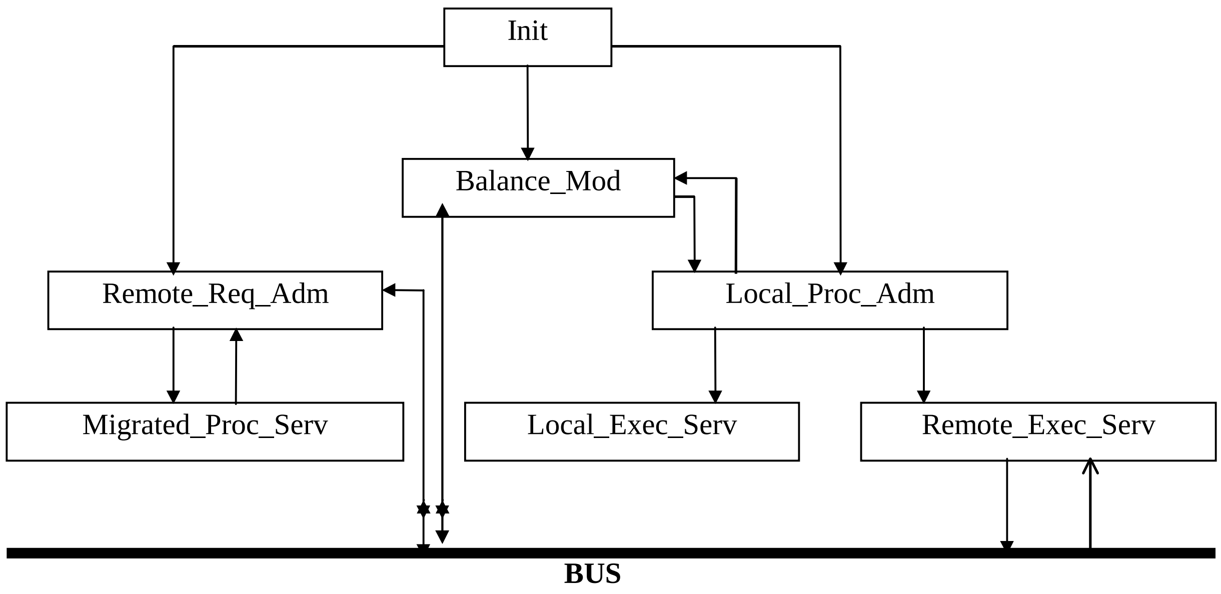


Fig.2: Internal Structure of Load Balancing Strategy

Short description of modules functions follows:

INIT, executes only once at node bootstrapping, is in charge of activating the three central system modules which in their turn manage local or external requests and load balancing
.
BALANCE_MOD, this module implements the load balancing strategy. In the simulated system it contains diverse load balancing algorithms for performance studies.

REMOTE_REQ_ADM, has two main tasks:

- Replies migration requests from other nodes, giving information about local loading state (number of waiting processes, or ready queue length). Also, when an immigrated process finishes execution in the local node, informs about this event to the (original) sending node.

- Activates a child server process when a remote process from an overloaded node arrives and the local node is idle or in a low loading condition.

MIGRATED_PROC-SERV, executes locally an immigrated process and, on completion, signals the event to ***REMOTE_REQ_ADM***.

LOCAL_EXEC_SERV, is in charge of local execution for a process.

LOCAL_PROC_ADM, it is responsible, at local process creation time, to verify the local node balancing state by comparing the current queue length with a prefixed threshold to determine overloading.
Depending on comparison results some of the following actions will be undertaken:

- If $L \leq 0$ then the task will be locally executed and a child process, ***LOCAL_EXEC_SERV*** will be activated.

- Otherwise, invokes ***BALANCE_MOD***, who indicates if the new process can be migrated and to which node. If a receiving node can be found then a ***REM_EXEC_SERV*** process is created. On the contrary, local execution will be accepted and behaves as above explained.

REM_EXEC_SERV, is responsible for migration of the process (indicated by ***LOCAL_PROC_ADM***) to the receiving node (pointed by ***BALANCE_MOD***). Finally blocks itself waiting for reply related to the remote execution completion.

3. ADAPTIVE PREDICTIVE LOAD BALANCING STRATEGY DESCRIPTION

APLBS attempts to reduce the number of requests from an overloaded node and is an improvement on PLBS proposed in [1]. PLBS proceeded in the following way:

Two thresholds O and U (related to a metric) were used to define the loading state L of a node:

$$\begin{array}{ll} L > O & \Rightarrow \text{ overloaded,} \\ U \leq L \leq O & \Rightarrow \text{ medium load} \\ L < U & \Rightarrow \text{ underloaded} \end{array}$$

Each node maintains its loading state metric, which is determined at fixed time intervals. In our case this metric is related to the number of processes in the ready queue. At each node there are two daemon processes: *spreader* and *refresh*. The scheme for the ***BALANCE_MOD*** is shown below:

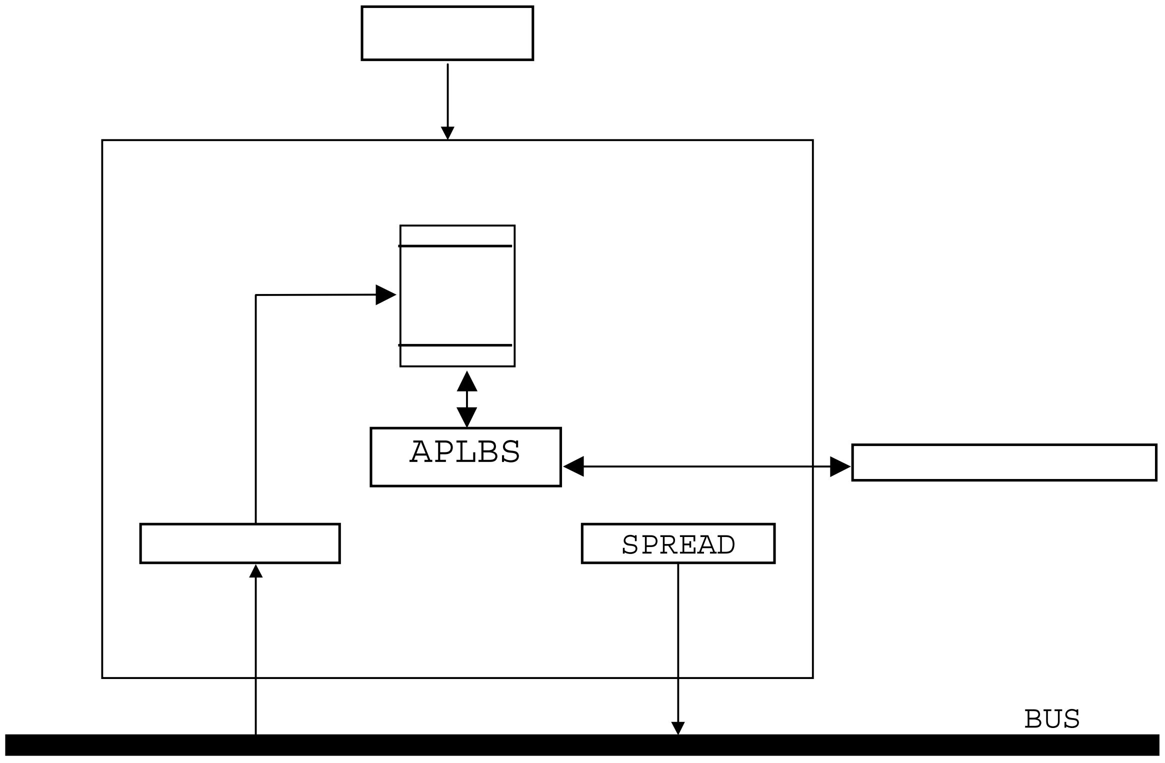


Fig.3: Scheme for PLBS and APLBS implementations

The *spreader* process disseminates local metric information to a random subset of 40% of the system nodes when either metric values changed or a time limit is exceeded without changes.

The *refresh* process maintains, in a global information table at each node location, the metric values of all nodes in the system. Values in that table are weighted exponential average values of collected metric (load) values.
The philosophy behind APLBS is to predict the current working load of a given processor when a migration request is necessary.

In PLBS the migration requests were sent to those nodes that were expected to be more inclined to accept due to their probable low loads. In this strategy request were sent to the *best subset* of the nodes in the system. This information was retrieved from the global load state table of the overloaded node. Best subset sizes were of 50%, 20% and 10% of the total for 10, 25 and 50 nodes in the system, respectively.

APLBS differs with PLBS essentially in that the subset size is dynamically updated. It retains the property of requesting migrations to the nodes more inclined to accept, but the subset sizes varies according to the feedback each node receives from the system. Best subset size begin with 10% of nodes in the system. After requesting migration of a process to candidate nodes, the requester have the answer of each of the targets and this is its feedback from the system.
Let us call *BSS* the best subset size, *NA* the number of accepted requests and *NR* the number of rejected requests
The following simple rule, based on the number of requests accepted and rejected, is followed by APLBS:

```
If $NR > NA$ then $BSS = BSS+1$
        else if $NA > NR$ then $BSS = BSS-1$
```

In this way , for a given node, *BSS* varies dynamically while the system is running. The mechanism to control this parameter is adaptive because some feedback information is used to determine the direction of the change in the parameter.

For establishing a reliable current metric value, Weighted Exponential Average [12] permits to predict a value on the basis of values appeared during certain elapsed time.
In our model, due to the dynamic behaviour of ths system, it is appropriate to give a larger weight to the recent history. For an arbitrary processor, the predicted working load is given by:

$$L_n + 1 = \alpha S_n + (1 - \alpha) L_n , \quad 0 \leq \alpha \leq 1 \qquad (1)$$

where:
$L_n + 1$: predicted processor load for the next migration request.
S_n : effective processor load at the n^{th} sample interval.

The parameter α allows controlling the relative weight to be given to immediate or old history. If α is equal to zero the recent history is considered irrelevant (present conditions are transient), otherwise if α is equal one then recent history is important and past history obsolete.
In this way the past and recent history is maintained and weighted for each system component and when a migration is needed from an overloaded node, requests are addressed to those candidates more inclined to accept the request.
The corresponding side effect is lower number of request, high acceptance hit ratio and therefore an enhanced performance of the distributed system is achieved. In order to decrease the communication traffic, typically generated by load balancing schemes, exchange of information relative to load level in a node is controlled by a daemon process, local to each node, and then broadcast to a random select subset of nodes each time.

4. Experiment descriptions

The distributed system implementing the strategy was simulated using a computer systems modelling tool, PARASOL [10], which is oriented to modern distributed or parallel computer systems.

The system parameters were defined as follow: Systems with 10, 25 and 50 nodes were simulated. Each node executed concurrent process under a *round-robin* policy maintaining a *ready* queue. The network topology was *Ethernet.* All processes were CPU intensive of 64 KB. The service time was fixed for all processes in 50 time units. The network transfer rate was of 10 Mbits.

Process arrivals follow a Poisson distribution of mean λ. A simulation was completed when 50,000 processes where executed in the network nodes.

Experiments were carried out on following scenarios, using 1/λ as mean interarrival time with λ = 0.002, 0.004, ... , 0.016, 0.018, 0.019.

- **Scenario 1:** 60% of the nodes are receiving processes with equal arrival rate λ while in the remaining nodes does not occur any arrival. This schema allows simulation of a clearly unbalanced situation.
- **Scenario 2:** 40% of the nodes are receiving processes with low arrival rate (λ) and the other 60% with more high arrival rate (2 λ).
- **Scenario3 :** Each node has its own arrival rate λ , which varies randomly through time.

Scenario 1 attempted to reflect a real situation, which frequently occurs, where the workload is not evenly distributed. Scenarios 2 and 3, are similar in the sense that arrivals occur in every node, but scenario 3 differs reflecting time depending arrival rates as often occurs in a computer network.

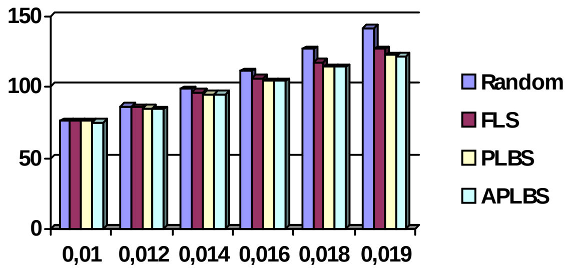
5. Experiment results

For the discussions of results we choose three representative instances of the diverse scenarios. But in general the same trend is observed in any scenario with any number of nodes in the network.. In every case the mean response time, number of requests issued, acceptance hit ratio and number of migration failures are reported.

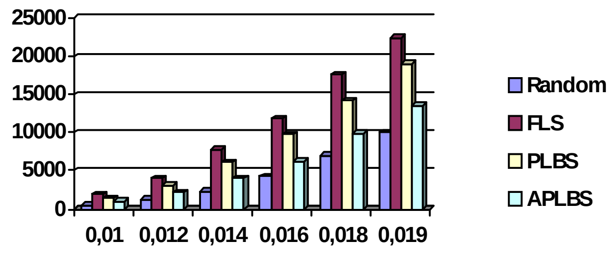
Scenario 1, 25 Nodes

λ	Random	FLS	PLBS	APLBS
0,01	75,9679	75,7693	75,3716	75,0413
0,012	86,1944	85,2521	84,7125	84,1877
0,014	98,3181	96,0296	94,9120	94,6534
0,016	111,2317	105,8215	103,8312	104,1134
0,018	126,7020	117,1297	113,5067	113,8647
0,019	140,9986	126,1461	122,1383	121,7103

MEAN RESPONSE TIME

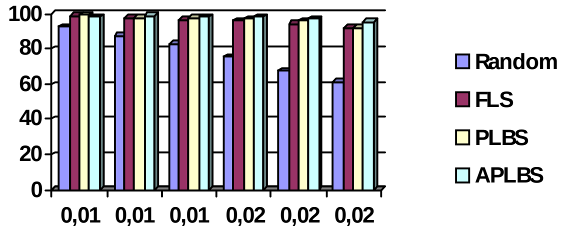


λ	Random	FLS	PLBS	APLBS
0,01	454	1946	1380	920
0,012	1171	3959	3132	2156
0,014	2355	7855	6027	4025
0,016	4253	11793	9693	6282
0,018	6947	17588	14196	9846
0,019	9968	22510	19089	13491



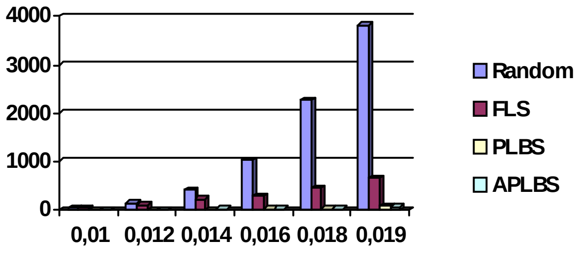
NUMBER OF REQUESTS

λ	Random	FLS	PLBS	APLBS
0,01	92,95	98,97	99,64	98,70
0,012	87,62	97,95	97,92	98,93
0,014	82,89	96,68	98,11	98,76
0,016	75,97	96,10	97,50	98,74
0,018	67,47	94,30	95,94	97,14
0,019	61,58	92,40	92,40	95,54



ACCEPTANCE HIT RATIO

λ	Random	FLS	PLBS	APLBS
0,01	32	36	0	0
0,012	145	95	0	0
0,014	403	212	0	2
0,016	1022	285	4	2
0,018	2260	436	10	12
0,019	3830	651	68	46



MIGRATION FAILURES

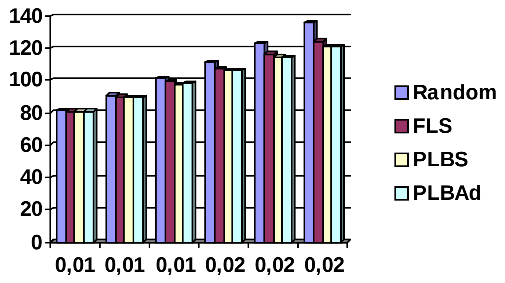
In scenario 1 with 25 nodes, when contrasted against PLBS, results about the performance variables for APLBS are the following:

The *mean response time* is almost always less than or equal to those of PLBS, and consequently the minimum when compared with other strategies.

The *number of requests* is reduced about 33% to 36% and the *acceptance hit ratio* is augmented between 1 and 3% along the varying arrival rate λ. *Migration failures* remains null, augmented and diminished for low, intermediate to moderate and high values of λ, respectively.

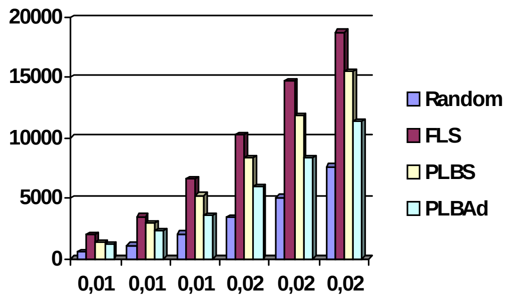
Scenario 2, 50 Nodes

λ	Random	FLS	PLBS	APLBS
0,01	81,4379	80,7430	80,6736	80,7703
0,012	90,9181	89,6548	89,2745	89,3846
0,014	101,0775	98,9344	97,2472	97,9350
0,016	111,0505	106,9376	106,0185	105,9856
0,018	122,9868	116,5085	114,6033	113,7497
0,019	135,6011	124,2325	121,0705	121,1705



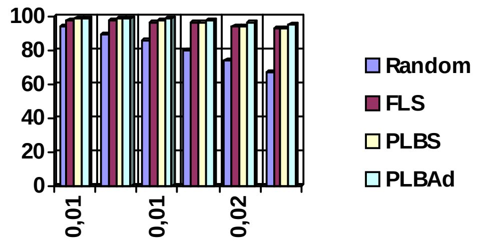
MEAN RESPONSE TIME

λ	Random	FLS	PLBS	APLBS
0,01	492	1928	1311	1182
0,012	1078	3466	2880	2326
0,014	2059	6561	5223	3603
0,016	3352	10223	8289	5914
0,018	5052	14724	11799	8282
0,019	7627	18760	15459	11313



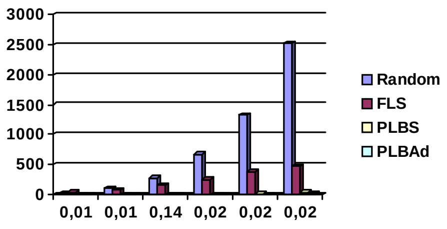
NUMBER OF REQUESTS

λ	Random	FLS	PLBS	APLBS
0,01	94,51	98,55	99,39	99,41
0,012	90,17	97,84	99,10	99,14
0,014	86,55	96,89	98,14	98,97
0,016	79,86	96,59	97,25	97,95
0,018	73,91	95,21	94,98	96,97
0,019	66,87	93,60	93,60	96,00



ACCEPTANCE HIT RATIO

1/λ	Random	FLS	PLBS	APLBS
0,01	27	38	0	0
0,012	106	78	0	0
0,014	277	154	0	0
0,016	675	249	0	4
0,018	1318	385	10	4
0,019	2527	469	36	26



MIGRATION FAILURES

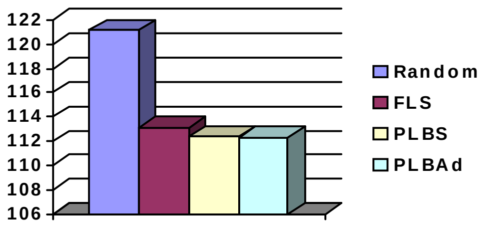
Again in scenario 2 with 50 nodes, when contrasted against PLBS, results about the performance variables for APLBS are the following:
The *mean response time* is almost always less than or equal to those of PLBS, and consequently the minimum when compared with other strategies.

The *number of requests* is reduced about 10% to 32% and the *acceptance hit ratio* is augmented between 2% and 3% in the higest values of λ. The number of *migration failures* shows a behaviour similar to that observed in scenario 2 and 50 nodes.

Scenario 3, 10 Nodes

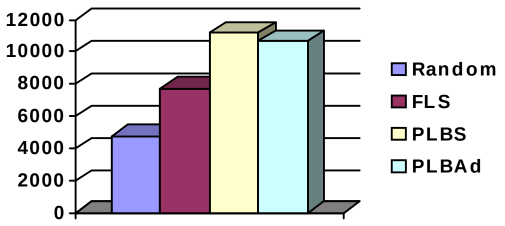
MEAN RESPONSE TIME

Random	FLS	PLBS	APLBS
121,1294	113,1027	112,3549	112,3047



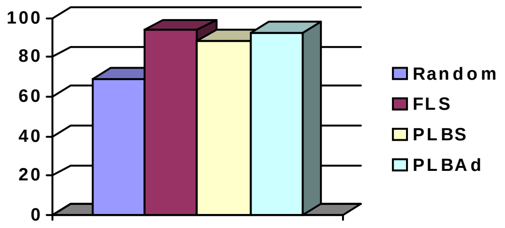
NUMBER OF REQUESTS

Random	FLS	PLBS	APLBS
4793	7747	11226	10647



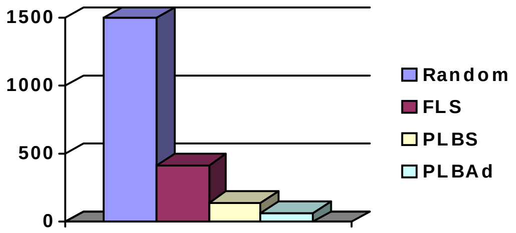
ACCEPTANCE HIT RATIO

Random	FLS	PLBS	APLBS
68,75	93,77	88,55	92,46



NUMBER OF MIGRATION FAILURES

Random	FLS	PLBS	APLBS
1498	418	146	68



Again in scenario 3 with 10 nodes, when contrasted against PLBS, results about the performance variables for APLBS are the following:

The *mean response time* is equal to that of PLBS, and consequently the minimum when compared with other strategies.
The *number of requests* is reduced about 6%, the *acceptance hit ratio* is augmented between 4%, and the number of migration failures is reduced in about 50%.

6. CONCLUSIONS

This paper introduced an improvement of the new load balancing strategy PLBS presented in [1]. The main characteristics of APLBS are simplicity and efficiency. Analogous to the previous one APLBS is based on prediction of the current working load of prospective migration request acceptors. But differently, here the size of the best candidates subset is updated dynamically by means of an adaptive technique.
Comparative analysis of results show interesting improvements when mean response time, acceptance hit ratio and percentage of migration failures are considered as relevant performance variables. all of this is achieved with a sensitive reduction of the number of requests.

7. ACKNOWLEDGEMENTS

We acknowledge the cooperation of the project group for providing new ideas and constructive criticisms. Also to the Universidad Nacional de San Luis the CONICET, and the ANPCYT from which we receive continuous support.

8. BIBLIOGRAPHY

[1] Esquivel S., Pereira C. and Gallard R. - "Predictive Load Balancing for a Workstation Distributed System, Proceedings de la International Conference on Applied Informatics, Garmish, Germany, February 1998

[2] Kremien O. and Kramer J. - "Methodical Analisys of Adaptative Load Sharing Algorithms", IEEE Transaction on Paralell and Distributed Systems, V. 3, Nº 6, págs. 747-760, November 1992.

[3] Chu W., Holloway L., Lan M., Efe K. - "Task Allocation in Distributed Data Processing" - Distributed Computing: Concepts and Implementations, pp 109-119, Addison Wesley - 1984.

[4] Esquivel S., Leguizamón G., Gallard R. - "A Hybrid Strategy for Load Balancing in Distributed Systems Environments", Proceedings of the Fourth IEEE International Conference on Evolutionary Computation (ICEC'97), pp. 127 -132, Indianapolis, USA, April 1997.

[5] Ferrari D., - "Study of Load Indices for Load Balancing Schemes", University of California, Berkeley, 1985.

[6] Jun C., Li X., Zhong-xiu S. - "A Model for Intelligent Task Scheduling ina Large Distributed System", ACM Press, Operating Systems Review, Vol.24, Nº 4, October 1990.

[7] Kremin O., Kramer J. and Magee J. - "Scalable, Adaptive Load Sharing for Distributed Systems - IEEE Parallel and Distributed Technology, pp. 62-70August 1993.

[8] Mullender S. - "Distributed Systems" - Addison Wesley, 2da. edition, 1995.

[9] Munetomo M., Takai Y., y Sato Y. - " A Genetic Approach to Dynamic Load Balancing in a Distributed Computing System", Proceeding of the First IEEE Conference on Evolutionary Computation, June 1994, Vol. 1, pp.419-421.

[10] Neilson J., - " Parasol User's Manual ", School Of Computer Science, Carleton University, Canada.

[11] Panjak M. - " Automated Learning of Load Balancing Strategies for a Distributed Computer Systems" PhD. Thesis, University of Illinois at Urbana, Chapaign, 1993.

[12] Stallings William - "Operating Systems" - MacMillan publishing Company, New York, 1992.

[13] Stone H., Bokhari S. - “Control of Distributed Processes” - Distributed Compu-ting: Concepts and Implementations, pp. 109-119, Addison Wesley - 1984.