A Competitive Symmetrical Transfer Policy for Load Sharing*

Konstantinos Antonis, John Garofalakis, Paul Spirakis

[1] Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece
Phone: (+30)61-273496, Fax: (+30)61-222086
{antonis , garofala , spirakis}@cti.gr
[2] University of Patras, Department of Computer Engineering and Informatics
26500 Rion, Patras, Greece.

Abstract. Load Sharing is a policy used to improve the performance of distributed systems by transferring workload from heavily loaded nodes to lightly loaded ones in the system. In this paper, we propose a dynamic and symmetrical technique for a two-server system, called *Difference-Initiated (DI)*, in which transferring decisions are based on the difference between the populations of the two servers. In order to measure the performance of this policy, we apply the SSP analytical approximation technique proposed in [3]. Finally, we compare the theoretically derived results of the DI technique with two of the most commonly used dynamic techniques: the *Sender-Initiated (SI)*, and the *Receiver-Initiated (RI)* which were simulated.

1 Introduction

Transferring of jobs between two nodes lies in the heart of any distributed load sharing algorithm. Especially in the cases of *dynamic* or *adaptive* load sharing techniques (in which scheduling decisions are based on the current state of the system), the transferring of jobs from a heavily loaded node to a lightly loaded node, requires in most cases the mutual knowledge of the current state of the two nodes (server and receiver). Such dynamic load sharing techniques are the nearest neighbor approach, the sender-initiated, the receiver-initiated, etc.

In this work, we propose a *symmetrical transfer policy* for load sharing between two nodes, that uses a threshold on the difference of the populations of the respective queues, in order to make decisions for remote processing. We evaluate the performance of this policy, by applying the State Space Partition (SSP) supplementary queueing approximation technique [3], and also simulations. In the simulation model that we developed, we incorporated two of the most widely used dynamic load sharing policies, the *sender-initiated (SI)* and *receiver-initiated (RI)* [1], in order to compare their performance with our policy, which we call *difference-initiated (DI)*.

* This work was partially supported by ESPRIT LTR Project no. 20244 - ALCOM-IT basic research program of the E.U. and the Operational Program for Research and Development 2 No. 3.3 453 sponsored by the General Secreteriat for Research and Technology.

2 The Difference-Initiated Technique

2.1 The Model

We consider two serving queues as the processing nodes, which are subject to the following simple load sharing mechanism: when the difference of their queue lengths is greater than or equal to a threshold c, then jobs are transferred from the heavier loaded queue to the lightly loaded queue at a constant rate β (i.e. the transfer period is exponential of mean length $1/\beta$). The above algorithm behaves symmetrically, either as server-intitiated or receiver-initiated, depending on whether the difference between the two queue lengths exceeds or is under the corresponding threshold c. A server probes for the queue length of the other server, every time a job arrives to or departs from its own queue.

2.2 The State Space Partition (SSP) Technique

It is easy to observe that the behaviour of our system as time progresses, balances between 3 elementary queueing systems, QS_1, QS_2, and QS_3, when $|n_1 - n_2| < c$, $n_1 - n_2 \geq c$, and $n_2 - n_1 \geq c$, respectively. This technique computes the steady-state probability $P(n_1, n_2)$ for finding n_1 and n_2 customers at $queue_1$ and $queue_2$ respectively. The approximation technique conjectures that:

$$P(n_1, n_2) = A\, P_1(n_1, n_2) + B\, P_2(n_1, n_2) + C\, P_3(n_1, n_2) \tag{1}$$

where: $P_1(n_1, n_2)$, $P_2(n_1, n_2)$, $P_3(n_1, n_2)$ denote the steady-state probabilities for finding n_1 and n_2 customers at $queue_1$ and $queue_2$, for the three elementary, easily solvable systems above, with no constraints for $(n_1 - n_2)$, and

- $A = Prob\{\epsilon_1\}$, where $\epsilon_1 = \{(n_1, n_2) : |n_1 - n_2| < c\}$,
- $B = Prob\{\epsilon_2\}$, where $\epsilon_2 = \{(n_1, n_2) : n_1 - n_2 \geq c\}$,
- $C = Prob\{\epsilon_3\}$, where $\epsilon_3 = \{(n_1, n_2) : n_2 - n_1 \geq c\}$.

In applying equation 1 we are confronted with the problem of estimating A, B, and C. So, we use the following iteration technique:

Iteration Algorithm

- Solve QS_1, QS_2, QS_3 by any existing technique for Product Form Queueing Networks (Mean Value Analysis, LBANC, etc.) and get $P_i(n_1, n_2)$, $i = 1, 2, 3$.
- Assume initial values for A, B, C, namely $A^{(0)}$, $B^{(0)}$, $C^{(0)}$. Here we select: $A^{(0)} = P_1(\epsilon_1)$, $B^{(0)} = P_1(\epsilon_2)$, $C^{(0)} = P_1(\epsilon_3)$.
- $i = 1$
- $(\star)$ Compute the i^{th} iteration value of $P(n_1, n_2)$ by equation 1 using $A^{(i-1)}$, $B^{(i-1)}$, $C^{(i-1)}$. Call it $P^{(i)}(n_1, n_2)$
- Find new values $A^{(i)}, B^{(i)}, C^{(i)}$ by using the following equation:

$$A^{(i)} = A^{(i-1)} P_1(\epsilon_1) + B^{(i-1)} P_2(\epsilon_1) + C^{(i-1)} P_3(\epsilon_1) \tag{2}$$

  Similarly, for $B^{(i)}$ and $C^{(i)}$, summing over ϵ_2 and ϵ_3, respectively.
- $i = i + 1$
- If a convergence criterion for A, B, C is not satisfied goto $(\star)$.
- Use $P_{(i+1)}(n_1, n_2)$ as an approximation to $P(n_1, n_2)$.

3 Simulation Results and Comparisons

Three different transfer policies for load sharing between two FIFO servers have been analysed or simulated and compared: the difference-initiated (DI), the sender-initiated (SI), and the receiver-initiated (RI) technique. The analytical model of DI was described in section 2, while the models for the other two simulated techniques can be found in [1]. We mention here, that only one job can be in transmission any time for each direction. We assume Poisson arrival times, and exponential service and transfer times for all the above policies. The parameters used are the following: T_s = the threshold adopted for SI, T_r = the threshold adopted for RI, c = the threshold adopted for DI, λ_1, λ_2 = the arrival rates for the two servers, μ_1, μ_2 = the service rates for the two servers, β = the transfer rate.

In order to have comparative systems, corresponding to the three policies, we choose to have $T_s = c$ and $T_r = 0$ in all cases. We compare the three above transfer policies concerning the average response time for a job completion (the total time spent in the two server system by a single job). We have taken analytical (SSP method) or simulation results for the above performance measures for DI including: homogeneous and heterogeneous arrival times and variable β, for heavier and lighter workloads. The same results for SI and RI were taken by our simulation model.

3.1 Comparative Results for Homogeneous Arrival Streams

First, we consider the lightly loaded conditions. Our results have shown that RI has the worst average response time. On the ohter hand, DI behaves better, especially for large β (which means a low transfer cost, see fig. 1). As the arrival rate grows, the differences between the performance measures of the three techniques are decreased, but DI continues to behave a little better. The results for SI and RI are likely to be almost identical under heavily loaded conditions. We think that under more heavier conditions, RI should present better results compared to SI, confirming the results of [2]. DI performs better than the other two because of its symmetrical behaviour.

3.2 Comparative Results for Heterogeneous Arrival Streams

Considering the extremal case of fig. 2, we can see that DI continues to perform better than the other two techniques. DI balances the load between the two servers very well, and its results are very close to the results of the SI policy. On the other hand, RI presents very poor results, because of its nature, since it can not move a big number of jobs from the heavier to the lighter server. The explaination is that the other two policies can move jobs even when the lightly loaded server is not idle. According to the definition of RI and the threshold used in this case, a server has first to be idle, in order to receive a job from another server.

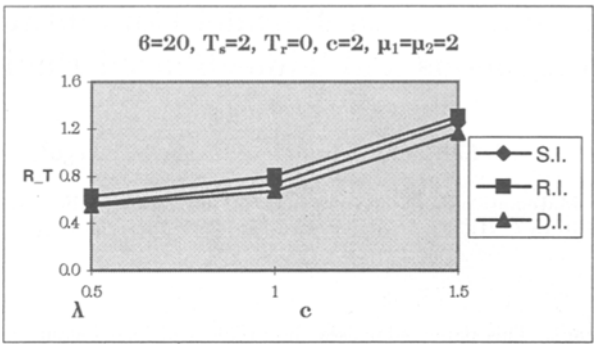

Fig. 1. Comparative results for the av. res. times (homogeneous arrival times, β=20).

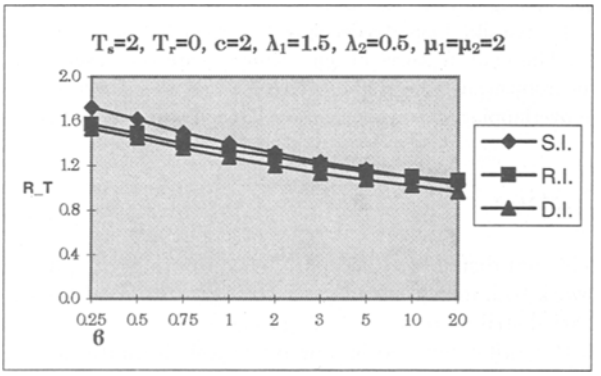

Fig. 2. Comparative results for the av. res. time (heterogeneous arrival times).

References

1. S. P. Dandamudi, "The Effect of Scheduling Discipline on Sender-Initiated and Receiver-Initiated Adaptive Load Sharing in Homogeneous Distributed Systems", *Technical Report, School of Computer Science, Carleton University*, TR-95-25 1995.
2. D. L. Eager, E. D. Lazowska, and J. Zahorjan, "A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing", *Performance Evaluation*, Vol. 6, March 1986, pp. 53-68.
3. J. Garofalakis, and P. Spirakis, "State Space Partition Modeling . A New Approximation Technique", *Computer Technology Institute, Technical Report*, TR. 88.09.56, Patras, 1988.