Using the Right Amount of Monitoring in Adaptive Load Sharing

David Breitgand* Rami Cohen † Amir Nahir * Danny Raz †

Consider a service that is being provided by a set of servers over the network. The goal of the service provider is to provide the best service (say, to minimize the service time) given the amount of available resources (*e.g.*, the number of servers). To achieve this goal the provider may introduce a load sharing capability into the system. In order to cope with dynamic environments, where the actual availability of each server changes over time, the load sharing mechanism needs to adapt to the current global state of the system. This implies that updated load information needs to be collected from the servers. Handling such load information requests requires small but nonzero resources (*e.g.*, CPU) from each server. This may reduce the actual service rate of the server, and thus, it is not easy to predict the actual amount of improvement expected from preferring a specific configuration of a dynamic load sharing scheme. For this reason it is important to identify just the right amount of resources that should be allocated for the monitoring of the servers' load in order to maximize the overall system performance. Moreover, since the optimal amount of monitoring depends on external parameters (such as the arrival rate of the service requests stream), the system should self-adjust the amount of monitoring according to the current conditions.

A very efficient load sharing model was studied in [3] by Mitzenmacher. This model, termed the *supermarket model*, uses a very simple randomization strategy: when a new task arrives, $d < n$ servers are selected uniformly at random, and the task is assigned to the server with the shortest queue among these d chosen servers. For $d = 1$ this process simply assigns jobs to servers uniformly at random, regardless of their load. However, for $d = 2$, the job is assigned to the least loaded server (the one with the shortest queue) among the two randomly chosen servers. It is shown in [3] that this simple process results in an exponential improvement of the expected overall time in the system compared to the (load independent) random assignment scheme. Further increasing d improves the expected service time linearly. The results of [3] suggest that even a very small amount of management information coupled with random job assignment may lead to a very efficient adaptive load balancing strategy, and as we use more information we keep on improving the scheme. However, this study assumes that the information about the local load of the servers is obtained and processed at no cost. As explained above, in many scenarios this assumption is not realistic.

In this work we extend the aforementioned supermarket model by incorporating the management costs into it. In particular, we assume that when a server is polled about its load, it has to allocate resources in order to answer this query. We consider a system that consists of n identical servers. Each server processes its incoming service requests according to the FIFO discipline. Service requests arrive to the system in a Poisson stream of rate $\lambda \cdot n$, $0 < \lambda < 1$, service time is exponentially distributed with mean 1. In the centralized ESM model, all clients' requests arrive at a centralized load balancing device. This device then selects $d < n$ servers uniformly at random (with replacement) and sends d inquiries about the length of the server queue to each of the selected servers. These monitoring requests have a precedence over the actual service requests, *i.e.*, upon receiving a monitoring request, the server preempts the currently running job (if such exists) and answers the load request immediately. We assume that processing the monitoring request takes a fraction $0 < C < 1$ of the mean service time of the actual service. This factor is the load monitoring efficiency ratio that reflects the fraction of the resources (i.e., CPU) needed in order to answer a load request. When the load balancing device (dispatcher) obtains all d answers (we assume that there are no message losses), it selects the server with the minimal queue length (ties are broken arbitrarily) and forwards the job to this server.

The theoretical analysis of this model (see [1]) results in the following equation that determines the total time in the system as a function of the load λ, the number of monitored server d, and the load monitoring efficiency ratio C.

$$T_d(\lambda) = \frac{\rho}{\lambda} \sum_{i=1}^{\infty} \rho^{\frac{d^i - d}{d-1}}, \ where \ \ \rho = \frac{\lambda}{1 - \lambda \cdot d \cdot C} \qquad (1)$$

The main outcome of this analysis is that for each system load and monitoring efficiency ratio C, there exists an optimal number d^* of servers that should be monitored in order

*IBM Haifa Research Lab, Israel. E-mail: davidbr@il.ibm.com, nahir@il.ibm.com

†Computer Science Department, Technion Haifa, Israel. E-mail: ramic@cs.technion.ac.il, danny@cs.technion.ac.il

Fourth International Conference on Autonomic Computing (ICAC'07)
0-7695-2779-5/07 $20.00 © 2007 IEEE

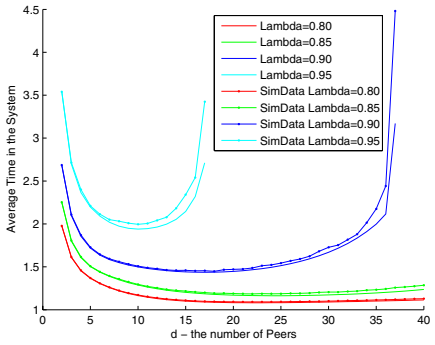

Figure 1. Model Vs. simulation

to minimize the overall expected time of jobs in the system. As shown in [3], in the original supermarket model, $T_d(\lambda)$ is a monotonically decreasing function for any $0 < \lambda < 1$. However, Equation 1 suggests that as d increases, the denominator decreases, resulting in higher waiting times, and when $\lambda \cdot d \cdot C$ approaches 1, the waiting time in the system goes to infinity. Thus, we expect $T_d(\lambda)$ to decrease when d increases, but at some point as d keeps on increasing, the value of $T_d(\lambda)$ should increase as the servers put more and more resources into monitoring the load and this affects the service time.

In order to verify that the Extended Supermarket Model (ESM) indeed models correctly the behavior of server systems as described in the previous sections, we conducted an extensive set of simulation runs. Figure 1 depicted the simulated results and the model prediction for relatively high load values, with 500 servers for $C = 0.003$. We observed typical service time of 300-400 milliseconds when monitoring a Web based application called PlantsByWebSphere on top of IBM WebSphere 6.0 application server, under normal load conditions. The load query time of 1 millisecond was observed in our preliminary experiments conducted on a realistic testbed.

One can see that the expected behavior indeed realized. The expected time in the system decreases when d increases, and at some point it starts increasing. When the system load is low (*e.g.*, $\lambda = 0.80$) the value of $T_d(\lambda)$ drops from almost 2 for $d = 2$ to about 1.1 for $d = 20$, and then it increase as d increases. This is due to the fact that when d increase from 2 to 10 the load sharing quality increases, while the effect of the cost devoted to monitoring is still small, however at this point, the return from increasing the number of peers becomes negligible while the effect of the monitoring cost keeps on increasing linearly. When the load is higher (e.g., 90%, 95%) one can see that there is a point in which $T_d(\lambda)$ starts increasing very fast. This happens when the system approaches the instability point. In this area the results becomes much more noisier, and the standard deviation of the simulation results increase.

Another important aspect that should be addressed is the dynamic behavior of the environment. In practical systems, many parameters such as the service requests rate (i.e., the system load), traffic load, and the availability of servers varies over time. Thus, the optimal number of servers that should be queried changes over time. An adaptive load sharing system is a system that self adapts its working point according to the current environment parameters. Such a system should be able to accurately approximate the value of the relevant parameters, and to dynamically adjust the number of the monitored servers to the optimal value. The most relevant parameter in our case is the overall system load. Note that the load results from both the mean time between arrivals and the service time. However, the actual service time is known only to the servers and not to the dispatcher. Hence estimating the current system load is not trivial.

In today's complex systems, management costs are inherent. Even though they can be reduced by configuring a system in a different way, or by using a different algorithm, they cannot be avoided completely. This is the case with servers' load monitoring for the purposes of efficient and adaptive load sharing. The main contributions of this work are as follows:

1. We develop a novel analytical model for the total utility of adaptive load sharing; this model allows rigorous quantitative reasoning about the tradeoff between the monitoring costs and the acquired benefits.
2. We evaluate the accuracy of the model through an extensive simulation study.
3. Based on the presented model, we develop an autonomous load sharing system that self adjusts the amount of monitoring to achieve best available utility.

This paper is just the first step in the formal study of this monitoring dilemma. One possible next step is to extend this model to a distributed setting in which load sharing is done by the servers themselves and not a separated centralized unit. Some steps in this direction are reported in [1]. Other very promising directions for further research deals with non-Poisson arrival rate and non-homogeneous servers.

References

[1] D. Breitgand, A. Nahir, and D. Raz. To know or not to know: on the needed amount of management information. Technical Report IBM-TJ-0242, IBM Research, Watson, 2006.

[2] M. Mitzenmacher. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems*, 11(1):6 – 20, January 2000.

[3] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094 – 1104, October 2001.

0-7695-2779-5/07 $20.00