Performance Comparison of Assignment Policies on Cluster-based E-Commerce Servers

Victoria Ungureanu*
Department of MSIS
Rutgers University,
180 University Ave.
Newark, NJ 07102
USA

Benjamin Melamed
Department of MSIS,
Rutgers University,
94 Rockafeller Rd.
Piscataway, NJ 08854
USA

Michael Katehakis
Department of MSIS,
Rutgers University,
180 University Ave.
Newark, NJ 07102
USA

Abstract: Cluster-based server architectures combine good performance and low cost, and are commonly used for applications that generate heavy loads. Essentially, a cluster-based server consists of a front-end dispatcher and several back-end servers. The dispatcher receives incoming requests, and then assigns them to back-end servers, which are responsible for request processing. The many benefits of cluster-based servers make them a good choice for e-commerce applications as well. However, applying this type of architecture to e-commerce applications is hindered by the fact that e-commerce clusters have the additional task of verifying that requests comply with contract terms. The problem is further complicated by the fact that contract terms may be expressed as functions of dynamic, mutable states.

We have proposed a policy, called TDA (Type Dependent Assignment), which is designed to balance load among back-end servers and to that request validation is done in a an efficient manner. In this paper we are interested in evaluating the performance of TDA. To gauge its efficacy, we present an empirical study, using real-life trace data measured at Internet cluster-based servers. Experimental results of this study show that TDA outperforms the Least Connected policy by a factor of four.

Keywords: stateful contract, assignment policy, enforcement

*Work supported in part by DIMACS under contract STC-91-19999
1 Introduction

Motivated by the need to cut costs and increase competitiveness, more and more enterprises are migrating to on-line transactions with trading partners [3]. Among the problems inherent in this migration, none is more serious than the difficulty of controlling the activities of the disparate agents involved in e-commerce.

Trading relations between enterprises are based on mutually agreed contracts. Generally, contracts enumerate agents authorized to participate in transactions, and spell out such things as the rights and obligations of each partner and the terms and conditions of trades. Moreover, at any given time an enterprise may be bound simultaneously by several contracts. For example, Ford has approximately thirty thousand suppliers, each operating under a different contract, and General Motors has about forty thousand [4].

As an example of a contract, consider that agents in a client enterprise (say Ford) may purchase merchandise (tires, in this example) from a supplier enterprise (say Firestone), under the following terms:

- **Firestone honors purchase orders (POs) issued by Ford, for up to a cumulative value, called blanket, of 10,000 tires each month at a prescribed price of $25/tire.**
- **Only agents duly certified as purchase officers by ford_CA (a designated certification authority of the client enterprise) may issue POs; only agents duly certified as sale representatives by firestone_CA (a designated certifying authority of the supplier enterprise) are authorized to respond to POs.**

Contract terms may take into account dynamic information, referred to as state. The state, which evolves over the lifetime of the contract, might consist of various contract phases, the state of transactions regulated by the contract, or past actions of various agents. A contract whose terms are expressed in terms of state is called stateful; a contract that consists only of immutable terms is called stateless. For example, the contract presented above is stateful, and its state is the value of the blanket. The validity of a PO depends on the current value of the blanket: a PO is valid provided the number of tires requested does not exceed the current value of the blanket. Furthermore, the blanket value is decreased whenever a valid PO is issued.

The satisfactory execution of e-commerce applications depends on the performance of the corresponding servers. To date, applications that generate heavy loads, commonly use cluster-based server architectures [2, 6, 9] that combine good performance and low cost. Essentially, a cluster-based server consists of a front-end dispatcher and several back-end servers, as depicted in Figure 1. The dispatcher receives incoming requests, and then assigns them to back-end servers, which are responsible for request processing. This type of architecture achieves transparency and a certain degree of scalability and fault-tolerance by decoupling request receiving from processing.

The many benefits of cluster-based servers make them a good choice for e-commerce applications. However, applying this type of architecture to e-commerce is hindered by the fact that an e-commerce cluster needs to verify that requests comply with the governing contracts.

In order to implement cluster-based architectures in an e-commerce setting, we have proposed in [13] a policy designed both to make request validation efficient and to distribute load in a balanced manner among back-end servers. The proposed policy, called TDA (Type Dependent Assignment), takes account of contract types. Under TDA the contract rules (immutable part) are replicated on all back-end servers. In contrast, the state of any stateful contract C, is preassigned to only one designated back-end server, called the base of C.
The broad outline of TDA can be described as follows. For every incoming request, the dispatcher determines the type of contract governing it (stateless or stateful). If the contract is stateless, the dispatcher assigns the request to the least loaded server (determined according to a prescribed load metric). However, if the contract is stateful, the dispatcher assigns the request to the base of C, whenever the base is not overloaded. In contrast, if the base is heavily loaded, the dispatcher reassigns the base to the least loaded server. We point out that only in this case, the state is sent to the new base and communication overhead is incurred. To support TDA implementation, we have proposed in [12] a contract formulation, which explicitly states contract type, and which distinguishes between mutable and immutable parts of a contract.

In this paper we are interested to evaluate TDA performance. We gauge the efficacy of TDA in a simulation study, using real-life trace data. As performance metric we use waiting time, computed as the time interval starting at request arrival in the dispatcher and ending at the point when processing commences. The study compares TDA with LC (Least Connected) policy, which calls for the dispatcher to assign a job to the server with the least number of open connections. We have chosen LC as base of comparison because it yields very good performance when compared with other traditional policies [11]. The experimental results show that for e-commerce applications, TDA outperforms traditional LC by a factor of four.

The rest of the paper is organized as follows. Section 2 describes in detail the TDA policy, and Section 3 presents a performance analysis for TDA, based on a simulation driven by empirical data. Section 4 discusses related work, and Section 5 concludes the paper.

2 The TDA Policy

The TDA policy makes the following assumptions and stipulations. First, TDA stipulates that the initial choice of bases by the dispatcher is made in Round-Robin manner. Second, TDA assumes that at any given time, the dispatcher knows the number of requests pending completion at each back-end server. These values are estimated by the number of active server connections and are used as a measure of server loads. This is a practical assumption, because the dispatcher is responsible for handling connections and passing incoming data from clients to back-end servers. Consequently, the dispatcher must keep track of open and closed connections at each server (cf., for e.g., [9, 11]). Finally, TDA assumes that the dispatcher has a parameter, AC, defined as the maximum number of requests (expressed as the number of active connections) that a server may process concurrently (to avoid excessive delays).

Subject to the assumptions and stipulations above, TDA operates on a request R governed by a contract C as follows:

1. When R arrives at the dispatcher, the dispatcher determines the
type of C.

2. If C is stateless, the dispatcher assigns R to the least loaded the server, where the load of a server is given by its number of active connections.

3. If C is stateful, the dispatcher determines the base S of C. If the number of active connections of S is smaller than AC, then the request is assigned S.

4. However, if the number of active connections of S exceeds AC, the dispatcher determines the least loaded back-end server, say S'. If S' is less loaded than S, then the dispatcher marks S' as the base of C and assigns the request to it. The dispatcher further notifies S of the change. After S finishes processing all requests in its queue that are governed by C, it sends the state of C to S'.

We point out that base reassignment is motivated by the fact that the initial assignment is made in Round-Robin manner, which is only a crude means for load balancing. Even though all back-end servers act as bases for the same number of contracts, some of them may get an unusually high number of “active” contracts, which govern a large fraction of the requests received in a certain time interval, and consequently their performance may degrade.

Changing bases incurs communication overhead, because the new assignment needs to be communicated to the old and new bases, and the state needs to be transferred to the new base. To balance the benefit of base change with its cost, the dispatcher does not change the base every time there is a server which is less loaded than the base. Rather, a base reassignment is carried out only when: (a) the number of active connections of the current base exceeds the prescribed threshold, AC, and (b) there is a server which is substantially less loaded than the current base. In the simulation experiments, described in the next section, the base is reassigned only when the least loaded server has fewer than AC/2 active connections.

3 Experimental Performance Study

We conducted an experimental study to compare TDA with the LC (Least Connected) policy, which calls for the dispatcher to assign a job to the server with the least number of open connections. The policies were compared with respect to the performance metric of waiting time, computed as the time interval starting at request arrival in the dispatcher and ending at the point when processing commences (for a stateful contract, this point is reached after the contract state becomes available).

The simulation model consisted of a cluster of four back-end servers, and 100 contracts, of which 80% were stateful. Since no traces from e-commerce sites are available, the study used one of the data traces of request arrivals at a Web cluster, available on the Internet Traffic Archive. Figure 2 depicts this trace, which contains approximately 170,000 requests over 200 seconds. The contract governing each of the requests was randomly chosen from the aforementioned set of 100 contracts.

The upshot of the experimental results is that TDA outperforms LC by a factor of four. More specifically, the waiting time averages computed over the simulated time interval trace are 47.1 ms for TDA and 215.8 ms for LC.

We now proceed to detail the simulation study and the associated results. Standard LC has been extended in the simulation to ensure state consistency. Specifically, a request, R, governed by a stateful contract, C, is handled in the following manner. The dispatcher sends to

the back-end server both R and the base of C. When the back-end server receives R, it requests the state from the base. After validating R, the back-end server sends to the base the updated state. The service time of a request is estimated as the sum of the time to verify compliance with the governing contract and the time to establish and close a connection. Furthermore, the simulation makes the following assumptions:

1. Communication times between the dispatcher and back-end servers are negligible.
2. The overhead incurred by the dispatcher to select (job, server) pairs is negligible;
3. The back-end servers are multi-threaded, where each thread performs all steps associated with a request before accepting a new one. When a thread blocks (because, for example, the state is used by another thread/server), another thread is chosen to run.

The parameters used by the simulation have the following values: (1) the time to evaluate a contract was measured empirically as 3.3 ms; (2) the time to establish and close a connection was measured empirically (see [11]) as 0.9 ms; (3) the time to retrieve the state from another server was estimated as 0.9 ms; (4) the threshold, AC was set at 20.

Figure 3 depicts the average waiting times in successive 1 second intervals. The figure shows that TDA improves over LC in almost all time intervals. A few exceptions occur only when the traffic is low, namely, when the number of incoming requests is less than 800 requests/second.
4 Related Work

The literature contains a considerable body of work on job scheduling (see [1, 2, 7, 5, 10] and references therein). Many of the policies devised took into consideration the size of a job when scheduling it for execution. A policy that does not consider job sizes is LARD (Locality-Aware Request Distribution) policy [8]. The goal of LARD is to assign HTTP requests so that access to disk — an expensive operation — is kept low. To this end, the dispatcher maintains for each document a set of back-end servers that have the document cached with high probability, and assigns a request for a document to the least loaded server from this set. It has been shown in [8] that assigning a job to a back-end server that can serve it from cache yields significant performance improvement when compared with Round-Robin.

5 Conclusion

In this paper we have shown experimentally that a previously proposed policy called TDA, which takes into consideration request validation, yields good performance. More specifically, TDA outperforms LC by a factor of four (both policies consider that the load of back-end servers is estimated by the number of active connections).

The results are obtained when incoming requests are governed by 100 contracts, out of which 80 are stateful. We are interested in studying how the number of contracts and the percentage of requests governed by stateful contracts affect the performance of the policy. The performance results of TDA under various
numbers of stateful and stateless contracts will be presented in a forthcoming paper.

References


