1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

A MULTI-AGENT APPROACH TO MANAGING ADAPTIVE SERVICE SYSTEMS – PRELIMINARY EXPLORATIONS*

Jyi-Shane Liu

Department of Computer Science
National Chengchi University, Taipei, Taiwan
E-mail: jsliu@cs.nccu.edu.tw

ABSTRACT

Service systems are abundant in our day to day lives. The establishment of a service system involves resource investment to provide the intended quality of service. When service resources are expensive and/or demand vary, it is desirable to have an adaptive service system that dynamically allocates resources to meet changing demands and evenly distributes loading to each service channel. In this paper, we present an approach to designing a service system such that task assignment is balanced in loading and resource utilization is adaptive to demands. The approach employs a multi-agent system with auction-based control and coordinated resource utilization adjustment. We chose world-wide-web servers as an application domain for the approach. We implemented the design in AWSA (Adaptive Web Server Agents) and analyzed its qualitative characteristics. The system is expected to provide efficient disk space utilization and balanced service quality with some overhead in data output rate.

1. INTRODUCTION

Service systems represent a generic modeling of many facilities operated for specific purposes. In general, a service system can be viewed as having finite resources, and by the usage of which, providing certain types of services to demands of work. For example, a bank has a limited number of service windows (and booths), each of which is a service channel (or server) for certain classes of services, such as regular transactions, opening new accounts, certificate of deposits, payments and fees collection, etc. Customers arrive at a bank and are directed to an appropriate service channel that can meet their demands. Each service channel has a finite capacity (rate of work). Customers who arrive at a busy service channel must join in a queue and wait for their turns for services.

Service systems are usually designed to operate and sustain for a long time. To establish a service system involves investment of certain amount of resources to provide the intended quality of service. One of the primary performance metrics is the average *system* time, which is the time a customer spent in the system. System time is composed of *waiting* time, the time a customer spent in queue for an available server, and *service* time, the time for a server to complete the requested service. System time can be reduced to service time only when infinite service channels are available or the service time is always smaller than the interarrival time. This is either infeasible or seldom, if not impossible. On the other hand, service time can be reduced with higher capacity service channels, which may be economically expensive. Most service systems must strike a strategic balance between the amount of resources invested and the quality of service provided.

When service resources are expensive and/or demands vary, it is desirable to have a service system that dynamically allocates resources to meet demands and evenly distributes loading to each service channel. An *adaptive service system* is a system that changes its service configuration in accordance with current demand situation so as to maximize its performance. Intuitively, an adaptive service system will provide better quality of service and better utilization of resources than a fixed service system. However, system reconfiguration usually involves *switch time* or *setup time* – the time it takes to change the service contents of a service channel. During this period of time, the service channel may not be available to provide any services. More importantly, substantial management cost may be required to constantly monitor the demand situation, perform load balancing, and arrange the restructuring of service contents. Sometimes these costs may outweigh its gain.

With the advances in computer technology, it is important to address how some of the computer control techniques can be used to help reduce system management cost. We propose a multi-agent system that takes charge of allocating tasks and managing service configuration. We assign an agent to each service channel. The agents monitor the service demand situation, exchange information with each other, determine the service location of a task, and initiate changes of service contents as situation warrants. In other words, these agents cooperate with each other to allocate tasks with even loading and coordinate with each other to organize an appropriate service configuration.

Our goal in this paper is to design such an intelligent multi-agent system so that an adaptive service system can

* This research was supported by a grant from the National Science Counsel under project number NSC 87-2213-E-004-006.

achieve better service performance with minimal management cost. In the next section, we justify the selection of world-wide-web servers as our case study in managing adaptive service systems and present the design of a multi-agent system called AWSA (Adaptive Web Server Agents). We also examines the qualitative characteristics of the system. Section 3 describes the design of experiments to obtain quantitative performance measures. Finally, we conclude with a discussion on the assumptions of the system, a review on related work, and a summary of the paper.

2. ADAPTIVE WEB SERVERS

We choose the Internet web servers as an example application domain in order to verify our multi-agent approach to managing adaptive service systems. Web servers represent an important class of information services on the Internet and are constantly servicing a tremendous amount of information requests from users all over the world. The efficiency of a web server is important not only to its users but also to the organization that runs it. Users are investing their time and money to submit a request and wait for its delivery. The longer a user waits the less satisfactory he/she is. Users' unhappy experiences not only have a negative effect on the image of the organization but also reduce the chances of their future visits.

Recently, the issue of web sites' quality of service has received much attention and has been raised in several articles on the Internet [1][2][3][4]. Researchers have begun to devise ways to improve web services. For example, predictive prefetching was proposed to reduce response time [5]. There are also commercial software packages for better web management. For example, PacketShaper manages and controls web network bandwidth [6]. Resonate Dispatch employs resource-based scheduling to control web traffic [7].

Many popular web sites employ multiple machines to handle large volume of requests. Most of these multiple-machine web servers are totally homogeneous service systems where every machine has the same data contents. Requests of services are distributed to each machine in round robin regardless of the data requested. This simple solution works fine if data storage space in a machine is always large enough to accommodate all the data any web site wishes to provide and data storage space is relatively inexpensive such that additional resources do not raise economic consideration. Rapid advances in disk technology seem to further encourage this view. However, there are situations where such luxury in data storage resources is not available. Particularly, in business world, every resource must be utilized to maximize its return in profit. If a piece of data is not requested much, there is no need to duplicate it in every machine. The saved space can be used to store other data and offer more information services to more users.

An adaptive web server system will manage its resources effectively by duplicating copies as necessary to maintain its service quality and removing excessive copies that are not needed. For example, commercial ISPs may provide virtual web sites to their users by renting machine disk space. Better management of disk space will facilitate more profits. In addition, users of virtual web sites may have very different purposes in presenting their data on the Internet. Some are individual expression and expect only little traffic. Some provide data that are more related to the general public and expect large volume of visits. Users can be charged according to the intended level of service quality and the actual space usage, instead of the inaccurate nominal rental disk space with no control over service quality. Proxy servers are another example where adaptive service systems are useful. Some referenced web pages are of common interest. Some are just personal flavor of a few. Better management of disk space will result in better performance.

2.1 System Design

We design a multi-agent system, called AWSA (Adaptive Web Serving Agents), to provide better management of web sites. Each web server is controlled independently by an agent. Each agent has an equal role in the system and may accept information requests from clients and service some of the requests as necessary. In other words, AWSA exercises distributed control to avoid control and communication bottleneck. Agents are regulated by an auction based coordination mechanism that integrates both task and resource allocation. As a result, AWSA achieves dynamic load balancing and dynamic reconfiguration in accordance with demand situation. In the following, we describe the system architecture in the sequence of operation process.

2.1.1 System Initialization/Maintenance

AWSA is designed to operate with minimal requirement of human intervention. For the most parts, system manager is involved only when data (web pages) are to be added or updated or permanently removed. A control interface is provided for system manager to issue commands. The control interface is an independent module and is executed only when necessary. Whenever new web pages are to be added to the system, the control interface broadcasts a message and asks agents to report the quantity of their free disk space. For each web page, the control module selects two agents to carry the data based on even distribution of storage loading among agents. In this way, the system contains two copies of each web page to provide fault tolerance. When certain web pages are to be removed from the system, the control module simply asks all agents to remove the specified web pages. Agents who do not carry these web pages just ignore the request. Web pages are updated by removing old copy and adding of new copy.

2.1.2 Accepting Requests from Clients

To avoid system bottleneck, each agent should be able to accept requests from clients. However, a unique domain name can only map to one IP address. A dispatch mechanism is required to direct the mapping to one of the agents. In addition, an even distribution of mapping is preferred. For this purpose, a simple round robin domain name server can be used.

2.1.3 Processing Requests from Clients

When an agent receives a request from a client, it checks if it owns the requested web page. If not, the agent proceeds to redirect the task of servicing the request by auction. The agent announces the task to all agents. When an agent receives a task announcement, it submits a bid to complete the task if it has a copy of the requested web page. Otherwise, it ignores the task announcement. A bid contains a predicted completion time of the task, which is estimated by the size of the requested web page, the current loading, and the data output rate of the bidding agent. The agent who announces a task waits for a specified period of time to collect bids from other agents. The agent then selects a best bid and redirects the task to the agent who submitted the winning bid.

When an agent receives a request of web page that it can service, it will first compare its current loading with loading of other agents. If the agent's loading is higher than the average loading of other agents, the agent will announce the task and select the best available agent, including itself as a candidate, to service the request. Otherwise, the agent will take the task itself without going through an auction. This is to reduce unnecessary communication traffic since the goal is to balance system loading not optimizing each individual request.

2.1.4 Description of Loading

Loading of an agent refers to the amount of tasks waiting to be finished by the agent with respect to its capacity. It depends on the number of task at hand, the data size of each task, and the data output rate. We use the estimated time of finishing the pending tasks to represent an agent's loading. Loading of each agent is broadcast to every other agent from time to time. An agent records and updates loading descriptions of other agents. These loading information are used to decide an agent's loading situation with respect to the other agents in the system.

2.1.5 Duplicating Data from Other Agents

When an agent receives a request that it cannot service and needs to redirect by auction, it may decide whether to acquire the requested data for such requests in the future. The agent considers the frequency of the requested item as the primary factor for the decision. If the requested item is expected to be requested again very often in the near future, it may be a good idea for the agent to obtain a copy of the requested data so that it can service such requests as well and help reduce their waiting time.

For each requested item, an agent always counts the number of times it has been requested or serviced (redirected by other agents) and calculates its frequency over a time interval. If the frequency of the requested item that it cannot service (cf. foreign data) is higher than the average request frequency of the agent's stored data (cf. local data), then the foreign data can be judged to be of high demand and may be worth duplicating. In the case that the agent still has free disk space (when the agent is a new member to the system or when certain data are removed by system manager), the agent will duplicate as much foreign data as possible starting from those of higher demand. In all cases, an agent duplicates foreign data by issuing HTTP requests at an appropriate time to the auction winners of task announcements for those requested data.

2.1.6 Removing Data to Obtain Free Space

When an agent decides to duplicate foreign data and needs to obtain free disk space for this purpose, local data of low request frequency are candidates for removal. However, if the system is design to be fault tolerant and needs to maintain two copies of an item in the system, then the agent needs to broadcast to ask for the existence of the selected item. The agent will remove the item after it confirms that two other agents have the item. Otherwise, the selection of removal is moved to the next candidate item in the list. In the case that the list has been moved up to an item of request frequency higher than that of the foreign data to be duplicated, the agent will give up the intended duplication.

2.1.7 Adding New Agents

A new agent joins in the system by broadcasting its loading description. When an agent receives a message from another agent that it has not known, it will record the identity of the new agent and return its own identity. The new agent may represent an additional machine to enhance system performance. With its free disk space, the new agent will rapidly duplicate as much foreign data from other agents and take up a portion of request services. In other words, a new agent can be merged into the system easily and is capable of utilizing its own resources for subsequent operation without much intervention from system manager

2.2 Performance Analysis

Most web sites that involve multiple servers (and machines) adopt simple round robin system in which a new request is blindly assigned to a server in a fixed order. Task distribution is balanced in terms of number not loading. A

server that happens to be assigned a number of heavy tasks in sequence may begin to accumulate quite a lot of pending tasks. In contrast to requests that are promptly serviced by other less busy servers, these pending requests may be significantly delayed. In other words, service quality is more varied unless requests are strictly stochastic and task loading is evenly distributed among servers over a period of time. In addition, a round robin web server system is a totally homogeneous service system. Either every server must have a disk space large enough to store all the data that need to be provided or the amount of data that can be provided is limited by the available disk space of a server. Resource investment with respect to range of services is less efficient. We expect the adaptive web server system can remove these drawbacks.

2.2.1 Efficient Disk Space Utilization

AWSA is a partially heterogeneous service system that allows a flexible configuration of web servers. Each server contains only a partial set of data services. A given set of data services can be achieved with less disk space requirement on each web server. In fact, there is no constraint on the storage configuration of a server as long as the collection of all servers covers the entire set of data services. This allows an adaptive system configuration in which the storage space allocated to a data item is dynamically adjusted according to its frequency of being requested. In particular, "hot" data will be duplicated by all servers and occupy more storage space, while the number of copies of "cold" data are reduced. Resources are put in tasks that are needed the most.

2.2.2 Balanced Service Quality

AWSA performs auction based task allocation. Each task is directed to an agent that can complete the task with the earliest finish time. The service quality of the same requested item from different clients is expected to be of lower variance than in round robin system. The agents also actively adjust their service portfolios by removing "cold data" and duplicating "hot data". This is to relieve congestion of tasks in high demand with minimal affects in servicing other tasks. By dynamically allocating more resources to the type of tasks in high demand, the system is expected to offer a better balance of service quality among different types of tasks.

2.2.3 Distributed Control

AWSA is designed to be an open agent system. Each agent is an independent control module and plays an equal role. The system is governed collectively by agents' interactions. Failure of an agent does not shut down the system operation. Agents can be instructed to maintain a certain number of copies for every data item in the system. Therefore, the system is robust and fault-tolerant in data services. In addition, the system is extendable. A new agent can be easily added and smoothly merged into the system. This allows incremental upgrading of a web site.

3. EXPERIMENT DESIGN

We have implemented AWSA in Java programming language. As an initial examination, we constructed a system with five agents and tested for its qualitative characteristics. In particular, we verified that the system actually performed auction based task allocation and dynamically adjusted its resource allocation. We also confirmed the extensibility of the system by temporarily adding an agent to the system and observing the merging process. Our next step is to investigate the quantitative aspects of AWSA by conducting a set of simulated experiments. Our goal is to provide a set of quantitative performance measures so that the capability of AWSA and the trade-off of using AWSA can be better understood.

We set up a set of data item as the set of web pages to be requested. The data size ranges from 25 K Bytes to 500 K Bytes with an increment of 25 K Bytes, e.g., {25K, 50K, 75K, ..., 475K, 500K}. A request pattern is a particular sequence of requested data item sampled from the data set over a period of time. For each experiment, we prepare three request rates that are considered to be high, medium, and low. With these environment setups, we observe AWSA's performance. In particular, we would like to be able to examine the following issues.

3.1 The Processing and Communication Overhead

A careful cost/gain analysis is an important first step before a decision can be made to adopt a new management approach. One cost involves slightly lower maximum data output rate because the system performs auction to assign tasks rather than simple round robin. A portion of CPU time and network communication time is consumed for system management and is not available for service. When the request rate is under system capacity, auction may mostly utilize system resources that would be idle anyway. However, these additional management activities may reduce system data output rate when the request rate is close to system capacity. In any case, we need to have a quantitative measure of AWSA's lower data output rate as compared to a round robin system under different request rates.

3.2 System Loading Distribution

A round robin system evenly distributes tasks in terms of number. Loading balance relies on two assumptions – one on the system and one on the environment. First, each server must be run on machines with similar capabilities. Second, clients' requests are stochastic such that each

server receives about the same total amount of loading over some period of time. On the other hand, AWSA is expected to perform real load balancing at each instance no matter what the request patterns are. In order to verify this capability, we need to measure loading of each server over time under different request patterns. We set up two request patterns as a contrast. One is random request or no pattern at all. The other is the "oscillatory" request such that, by round robin, heavy tasks are all assigned to one of the servers. In other words, there are substantial loading differences among servers that would not be smoothed out over time. The contrast between AWSA and a round robin system under these two request patterns will provide more solid evidence of AWSA's load balance capability.

3.3 Service Quality

One of the management goals in a service system may be a uniform service quality to clients who request the same type of service in about the same period of time. A client's system time is related to the loading of the server who is assigned to service the client. Therefore, in the previous experiment, we can also observe system time of each type of requests and calculate their averages and variances over time. A system with lower variance in system time indicates its ability to provide more uniform service quality.

3.4 Dynamic Resource Allocation

AWSA is designed to adjust its resource allocation according to demands. A demand pattern is a particular distribution of requests for the data set. If a subset of the data is requested more frequently than other data, AWSA is expected to allocate more resources to handle these requests so that they would not be congested. To verify this, we can measure the system time for each data item before and after resource adjustment under the same demand pattern. The performance comparison would allow us to provide a quantitative measurement of the gain.

In order to understand the effects better, we need to test AWSA under different demand patterns. However, the numbers of possible demand patterns are indefinite. We make several simplifications in constructing a demand pattern so that the amount of testing is reasonable. First, we divide the request frequency into two classes only – high and low. We vary the ratio of the high/low request frequency, e.g., 4:1, 16:1, 64:1, 128:1, to provide a sampling of the intensity of the high demand. Second, we consider only three percentages of data items that are of high request frequency, e.g., 5%, 10%, and 25%. Third, we select two sizes of the data set to be requested more frequently, e.g., small size, large size. For example, if the "hot data" is of small size and occupies 10% of the data set, data items of sizes 25 K Bytes and 50 K Bytes are requested more frequently.

With these experiments, we hope to construct a more complete picture of the cost and gain of AWSA and its performances under different environment settings. We have constructed a system with five agents and a round robin system with five servers to be our experimental platforms. We have also finished setting up the different conditions of the environment. We are currently in the process of conducting these experiments.

4. DISCUSSION

AWSA is a partially heterogeneous service system with load balancing and adaptive resource utilization. Such a system relies on sophisticated management activities and therefore, involves certain amounts of cost. Besides from cost/gain trade-off, there are a number of assumptions on the environment such that AWSA's characteristics are indeed desirable. In this section, we will discuss these assumptions.

As we have mentioned earlier, changing the service contents of a service channel usually involves switch time during which the server is not available for service. In other words, the system cannot adjust its configuration too frequently, otherwise, its cost may not be justified against its gain. We make the assumption that most web sites have "rush hours" and "quite spells", as observed by most service systems. Therefore, AWSA should reconfigure itself only when the system is in its "quite spells" periods, such as early morning (from 3 to 5 A.M.) and possibly lunch/dinner hours.

Another assumption has to do with the continuity of demand pattern so that it is useful to adjust system configuration and better service the upcoming demands. If we observe that some sets of data are requested more frequently, we assume that they will be demanded as much for some time after system reconfiguration. We conjecture that the popularity of a data item over time may be similar to a tide of wave. It goes through a rising period after its appearance, and gradually subsides after it reaches its peak. Nevertheless, this assumption will need to be verified on real web sites.

5. RELATED WORK

Recently, auction-based mechanisms have become very popular for distributed control in many practical application domains. In general, auction-based mechanisms achieve coordination among decision-making entities by performing auctions to distribute things, where things can be tasks, resources, or any other concepts. Task allocation is mostly carried out by contract net protocol (CNP) [8], which is generally viewed as the original form of auction-based mechanisms. CNP is considered to provide natural load balancing and reliable distributed control [9] and has been successfully applied to many problems [10][11].

Later, market-based mechanisms are proposed to solve more complex dynamic resource allocation problems [12]. Resources are viewed as goods and are distributed to needed entities on a public market (e.g., auctioning). In general, these resources are common pool resources that are not owned by any individual entity and are to be shared and competed by all. Market-based mechanisms have also seen increasing interest and many successful applications [13][14].

In our problem, the adaptive web server system requires dynamic task allocation for load balancing as well as dynamic resource reconfiguration for better service capability to meet demands. We adopt contract net protocol as AWSA's coordination mechanism because of its perfect fit with the problem. Only agents who are capable of performing the announced task are allowed to submit a bid and only the agent who will perform the task with the best terms is assigned the task. In this way, tasks are dynamically allocated to an appropriate agent and loading is dynamically balanced.

The resources in AWSA are not common pool resources. Each agent controls its own resource (disk space) and does not share it with others. Agents dynamically allocate their own resources to acquire necessary capabilities for servicing needed tasks. However, their decisions are coordinated and are intended to meet current demand situations as reflected by each agent's request frequency records.

6. CONCLUSION

In this paper, we present an approach to designing a service system such that task assignment is balanced in loading and resource utilization is adaptive to demands. The approach employs a multi-agent system with auction-based control and coordinated resource utilization adjustment. We chose world-wide-web servers as an application domain for the approach. We implemented the design in AWSA (Adaptive Web Server Agents) and analyzed its qualitative characteristics. The system is expected to provide efficient disk space utilization and balanced service quality with some overhead in data output rate. We also prepared a set of experiments to examine AWSA's performances quantitatively for better understanding of the system. Finally, the approach relies on the assumptions that there are "quite spells" periods for system reconfiguration and that demands rise and fall over some period of time.

7. REFERENCES

[1] Ben Adida, "It All Starts at the Server", *IEEE Internet Computing Online*, 1(1), http://www.computer.org/internet/9701/weaving9701.htm, 1997.

[2] Brain L. Wong, "Sizing Up Your Web Server", *SunWorld*, http://www.sun.com/sunworldonline/swol-10-1997/swol-10-sizeserver.html , 1997.

[3] Dow Patten, "10 Steps You Need to Take to Prepare Your Web Server for a Siege", *NetscapeWorld*, http://www.netscapeworld.com/netscapeworld/nw-03-1997/nw-03-siege.html, 1997.

[4] Rick Cook, "10 Tactics to Make Your Web Pages Load Faster", *NetscapeWorld*, http://www.netscapeworld.com/netscapeworld/nw-02-1997/nw-02-bestpract.html, 1997.

[5] Venkata N. Padmanabhan & Jeffrey C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency", *ACM SIGCOMM Computer Communication Review*, July 1996.

[6] Packeteer, PacketShaper Solutions, http://www.packeteer.com/solutions_wm.htm.

[7] Resonate, Resonate Dispatch, http://www.resonateinc.com/

[8] Reid G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", *IEEE Transactions on Computers*, C-29(12):1104-1113, December 1980.

[9] H. S. Nwana, L. Lee, and N. R. Jennings, "Coordination in Software Agent Systems", *BT Technology Journal*, Vol. 14, No. 4, pp. 79-87, October 1996.

[10] H. Van Dyke Parunak, "Manufacturing Experience with the Contract Net", In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, pp. 285-310, Pitman, 1987.

[11] Tuomas Sandholm & Victor Lesser, "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework", *Proceedings of the First International Conference on Multi-Agent Systems*, pp. 328-335, 1995.

[12] Michael P. Wellman, "A Market-Oriented Programming Environment and Its Application to Distributed Multicommodity Flow Problems", *Journal of Artificial Intelligence Research*, Vol. 1, pp. 1-22, 1993.

[13] S. H. Clearwater (ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*, World Scientific, 1996.

[14] Hirofumi Yamaki, Michael P. Wellman, and Toru Ishida, "A Market-Based Approach to Allocating QoS for Mutlimedia Applications", In *Proceedings of the Second International Conference on Multi-Agent Systems*, 1996.