LDMA and WLDMA: Load Balancing Strategies for Distributed LAN and WAN Environments

M.Aramudhan, and V.Rhymend Uthaiaraj
Dept.of Information Technology,
Madras Institute of Technology, Anna University,
Chennai-600 044, Tamilnadu, India

Abstract

This paper introduces a new load balancing algorithms, called LDMA (Load balancing using Decentralized decision-making Mobile Agents) and WLDMA (WAN Load balancing using Decentralized decision making Mobile Agents) which distributes load among local and geographically distributed web servers connected in a mesh topology, by a communications network. The performance of LDMA is compared with MALD (Mobile Agent Load Balancing) an existing mobile agent based load balancing and WLDMA is compared with without load balancing. LDMA and WLDMA architectures and all necessary attributes such as load deviation, system throughput and response time incurred as a result of the work are dealt with. In the proposed approach, a decentralized decision making algorithm is used for distributing requests among the web servers. A simulator is designed in C++ and implemented to model the LDMA and WLDMA techniques in the distributed web server environment.LDMA is new algorithm, which could be very useful in LAN environment. WLDMA could be very useful in WAN environment.

Key words:
Mobile Agents, LDMA, WLDMA, load balancing, distributed system.

Introduction

A distributed computer system is a collection of self-sufficient computers located at diverse or identical sites and associated by a communication network. The performance of a distributed system is enhanced to an adequate level by distributing the workload among the servers. Normally, load balancing at the server side assists to balance the load in distributed computer system. Winston [1] proved that the most excellent mechanism for achieving optimal response time is to distribute the workload equally among the servers. Traditional load balancing approaches on distributed web servers are implemented based on message passing paradigm. At present, mobile agent, technology is used to implement load balancing on distributed web servers [2]. A mobile agent is defined as a software component that can move freely from one host to another on a network, transport its state and code from home host to other host, and execute various operations on the site [2]. The mobile agent based approaches have the merit of high flexibility, low network traffic and high asynchrony.

Distributed web servers deploy in different geographical scopes. They can be organized into cluster of web servers linked through Local Area Network (LAN), to provide high processing power and reliability. The servers are heterogeneous in terms of hardware configuration, operating systems, and processing power. Generally, load balancing on Wide Area Network (WAN) is more time consuming since it involves the interaction between remote servers for gathering load information, negotiating on load reallocation, and transporting the workload [2]. LDMA strategy introduces the concept of "decentralized decision making" among the web servers in the cluster for processing the requests. LDMA is based on all-to-all communication without collection workload information and request transfer policies among the clustered web servers. Moving a large amount of request requires high bandwidth and it is difficult if a node's load changes quickly in relation to the time needed to move requests [6]. In WLDMA, each server processes client requests independently and periodically interacts with others to share the workload.

LDMA framework

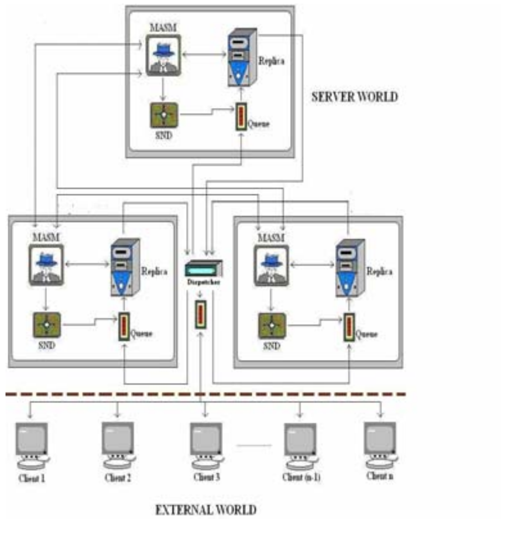


Fig. 1. Block diagram of the LDMA architecture

Manuscript received September 5, 2006.
Manuscript revised September 25, 2006.

The architecture of LDMA consists of a set of clients and a network of servers. The overall architecture of the LDMA framework is as shown in Fig.1.The LDMA framework defines two worlds, namely: client world and server world. The client world is an aggregation of all the clients in the physical world, and the server world is an aggregation of the clustered web servers, which are called replicas. The client world communicates with the server world through the dispatcher. The queue at the dispatcher has a finite buffer and a tail drop discard policy. But, unlike the other approaches, in which the dispatcher re-routes the client requests to corresponding servers (centralized decision making), the work of the LDMA dispatcher is to broadcast the client request to all the replicas. The dispatcher rewrites the IP address into broadcast IP address for each request is the overhead in the LDMA. The decision-making for load balancing among replicas takes place only by the interaction of mobile agents between the replicas. The replicas are inter-connected by mesh topology. Each replica has the following two types of agents:

- MASM – Mobile Agent Servicing Management.
- SND – Search aNd Destroy agent.

The work of the MASM is to communicate between the other to make them decide which replica may process a request. The work of the SND agent is to search for and delete (remove) a particular request from the replicas queue. The LDMA framework uses the concept of “ranked web-servers”, i.e., each replica is statically assigned a rank based on which priority given for processing a request. The mobile agents provide a novel technology for implementing load balancing mechanism on distributed servers. A replica can dispatch a mobile agent to the other replicas with Request ID and rank of the source server.

LDMA Load balancing Scheme

At the start, upon the arrival of a client request, the dispatcher broadcasts it to all the replicas, after assigning a RID (Request ID) to it. The replicas accept the request, but the request processing does not start immediately. Instead, the request is placed in its “*waiting state*”. Each replica sends a mobile agent, with the replica’s rank and the RID just accepted, which we call “*RID under siege*”. The mobile agents travel to the other replicas and check the state of the same request in the *destination replica*. Then, they return back to the *source replica* with any one of the following messages:

1. ***Accepted***: This case occurs when the rank of the source replica is less than the rank of the destination replica, and the RID under siege is in waiting state in the destination replica. On receiving this message, the source replica just ignores the accepted request, and chooses the next request.

2. ***Deleted***: This case occurs,

i. when the rank of the source replica is greater than the rank of the destination replica and the RID under siege is in waiting state in the destination replica, or

ii. When the RID in waiting state of the destination replica is less than the RID of the source replica.

The mobile agent triggers the SND module at the destination replica, which removes the RID under siege from the destination replica. On receiving this message, the source replica starts processing the request.

2. ***Not Found***: A mobile agent returns to source replica with this message, when RID under siege is not found either in its waiting state or even at the queue of the destination replica. This case occurs when the RID under siege has already been removed from the destination replica’s queue, by a mobile agent from the other replica. On receiving this message, the source replica ignores the accepted request, and chooses the next request.
To summarize, a replica, on accepting a request, sends mobile agents to other replicas and waits for the response. It ignores the request, even if at least one of the responses is an “*Accepted*” message. It starts processing the request otherwise. Moreover, in case of a packet loss of a mobile agent, a replica waits for a maximum of twice the RTT (Round Trip Time) of the mobile agent. In case of no response message, the replica starts processing the request.

The dispatcher assigns RIDs using mod N arithmetic, i.e.

RID = i mod N, where i = 0, 1, 2, 3… and N is the total number of requests in a cycle.

The RIDs are in ascending order. Hence, the work of SND agent at a replica is simple and it searches for the RID under siege from the top (using the *sequential search* algorithm), till i^{th} request in the queue, where i is “just greater than” the RID under siege. After the i^{th} request, the RID under siege cannot be found elsewhere in the queue (since RIDs are constantly increasing) except in the next cycle of RIDs.

The LDMA transition diagram is shown in Fig.2. After the completion of a request, next request moves to the wait state and activates the mobile agent. The mobile agent carries delete or not found message and then source replica will execute the request. If it carries accept message, then the SND agent will start its operation in source replica and promote next request to the wait state.

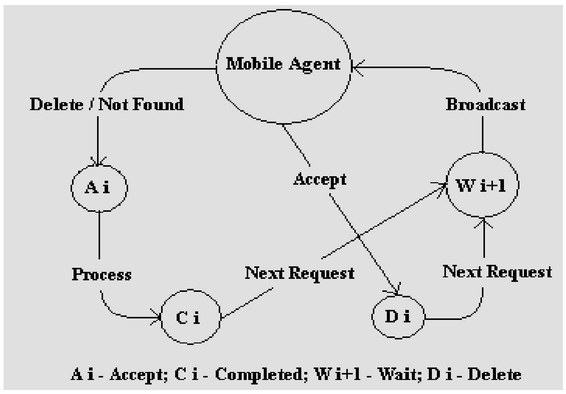


Fig.2: LDMA Transition diagram

From this section, input the body of your manuscript according to the constitution that you had. For detailed information for authors, please refer to [1].

LDMA Mathematical Model

Model Description: The system consists of n replicas, which represent host computers connected in mesh topology by a communication network. Each replica contains one or more resources (CPU, I/O devices...) contended by the requests processed at the replica. Replicas may be heterogeneous, that is, they may have different configurations, resources and speed characteristics. But, the processing capability of the requests at any server in the cluster is same.

Let the request arrival rate at the *i*th replica be ω_i. This is a time-invariant poisson process. The total request arrival rate at the dispatcher is denoted by ω. Let the request processing rate at replica *i* be γ_i. The mobile agents are constantly traveling from one replica to another. Let their rate of flow from *i*th replica to *j*th replica be a_{ij}. The response time of a job in the system depends on the node delay at the processing node in addition to a possible mobile communication delay incurred due to request search and delete in other web servers.

The mean replica (node) delay of a request at *i*th replica is a function of the load at replica *i*, $F_i(\gamma_i)$. Now,

$$F_i(\gamma_i) = \frac{1}{\mu_i - \gamma_i}$$

Where, $1/\mu_i$ is the mean service time. The mean node delay function $F_i(\gamma_i)$ depends on the load γ_i

The mean communication delay between the *i*th and *j*th replica is, $M_{ij}(A)$, where, $A = [a_{ij}]$. A is called the *Mobile Agent Matrix*. The $F_i(\gamma_i)$ and $M_{ij}(A)$ are non-decreasing functions. This delay has two components: first, a delay due to sending a mobile agent from its source server to other web servers and sending the response back to the source server. Second, delays caused by search and delete the request in the servers.

Thus, the mean response time of a request can be represented as $R(\gamma, A)$ where $\gamma = [\gamma_i]$. The mean response time is the sum of the mean replica delay and mean communication delay, that is,

$$R(\gamma, A) = \sum_{i=1}^{i=n} \frac{\gamma_i}{\omega} F_i(\gamma_i) + \sum_{i=1}^{i=n}\sum_{j=1}^{j=n} \frac{a_{ij}}{\omega} M_{ij}(A) + \nabla_{ij} \quad \text{---- (1)}$$

∇_{ij} is the delay for searching and deleting the requests in the other servers. The other performance measures, such as weighted sum of mean response times relative to request processing at different web servers, may be considered in a similar fashion.

The total traffic through the network is T. It is defined by

$$T = \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} a_{ij} \quad \text{---------- (2)}$$

For n>2, where n is the number of servers in the cluster connected by communication media in mesh fashion. The server k is sending mobile agent to replica j and i ($a_{kj}>0$, a_{ki}) and yet not receiving a mobile agent from replica i ($a_{ik} >0$). This case arises if the communication delay incurred as a result of sending a mobile agent directly from replica i to replica j is greater than the sum of delays from node i to node k and from node k to node j. In order to ensure that this case does not arise, assume that the communication network satisfies the triangle inequality property that is characterized by [5]

$$M_{ij}(T) \le M_{ik}(T) + M_{jk}(T), \quad i, j, k=1, 2, 3\ldots n$$

Moreover, the communication delay from replica i to replica j, $M_{ij}(A)$, is independent of the source-destination pair (i, j). Thus, substituting equation.2 into equation.1 yields

$$R(\gamma, a) = \sum_{i=1}^{i=n} \frac{\gamma_i}{\omega} F_i(\gamma_i) + \frac{T}{\omega} M(T) + \nabla_T \quad \text{--- (3)}$$

where network traffic T is expressed in terms of variable γ_i as

$$T = \frac{1}{2} \sum_{i=1}^{i=n} |\omega_i - \gamma_i|$$

Now, for optimal solution in general load distribution strategies, with no queue overflow at node *i* is

$$\sum_{i=1}^{i=n} \gamma_i = \omega_i \quad \text{------------- (4)}$$

But, in LDMA, at any node *i*,

$$\gamma_i = \omega/n$$

Therefore,

$$\sum \gamma_i = \sum_{i=1}^{i=n} \frac{\omega_i}{n} = \frac{n\,\omega}{n} = \omega$$

Thus, the total job transfer rate is equal to the total job arrival rate in LDMA where no queue is in overflowing condition.

From equation.4 if ω_i remains constant, as n increases, γ_i or job processing rate at a replica can decrease. In other words, as n increases, replicas can be less configured to serve fewer requests per second. And, as n increases, ω_i also increases proportionally; γ_i needs to remain constant to a certain upper bound of ω_i, for optimal performance. Hence, the incoming request packets are dropped at the replicas' queues only in the worst case.

3. THE WLDMA FRAMEWORK

The overall architecture of the WLDMA framework is as shown in Fig 3. The architecture considers the web servers to be widely separated from each other in a WAN environment. There is a physical or virtual connection between the servers so that, any web server can communicate with any other web servers using Mobile Agents. The clients usually send requests to the web server that is located geographically.

closer. Still, the client requests may be re-distributed among the web servers according to the WLDMA algorithm to ensure better response time to clients. Though the architecture shown above has only three web servers in a WAN, the WLDMA architecture is perfectly scalable to employ n web servers in WAN. The load on the overloaded server is transferred to an under-loaded server to enhance the system throughput. Thus, the system resources can get full utilization. In general, the servers can be heterogeneous in terms of hardware configuration, operating systems, and processing power. To simplify our discussion, all web servers are considered equivalent in their capabilities. The capacity of a server may change at runtime due to the variation of workload. In WLDMA scheme, job redistribution decision is taken by the individual server depends on the status of the jobs in the queue.

The client requests arrive at the web servers according to a Poisson distribution. The web servers process the client requests and respond to the clients. But, in order to share the loads among the servers almost equally. Mobile Agents are sent from/to the web servers to distribute the workload. Different functions in this scheme can be defined and encapsulated in Mobile Agents. The Mobile Agents carry the functions to other servers and execute it on the server. A Mobile Agent can be proprietary to a server where it is created and perform dedicated operations for the owner.

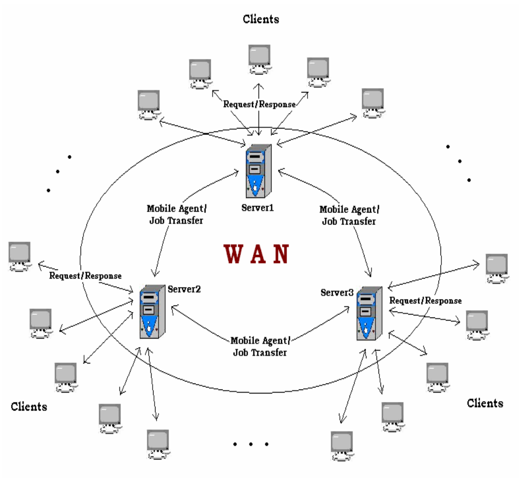


Fig. 3: The WLDMA Framework

The Mobile Agents can interact with each other by direct data exchange. They can interact using the stigmergy technique in which Mobile Agents can collect the information from the traces left in the environment by one another [3]. A Mobile Agent can gather the information placed on a server by other Mobile Agents who have previously visited here. The stigmergy is an indirect method for the interaction between Mobile Agents, which can reduce network traffic and achieve quick decision making[2]. The WLDMA frame work specifies four types of Mobile Agents:

-Load Status Agent (LSA).

-Load Gathering Agent (LGA)

-Job Reallocation Agent (JRA)

-Threshold Agent (TA)

LSA constantly monitors the queue size of the web server. Each server has its own LSA. It is a stationary agent that motionlessly sits at the server and responsible for monitoring the workload on local server. Each server deploys and sends a LGA (consisting of queue size of the source server) to all the other web servers every 150ms theoretically.

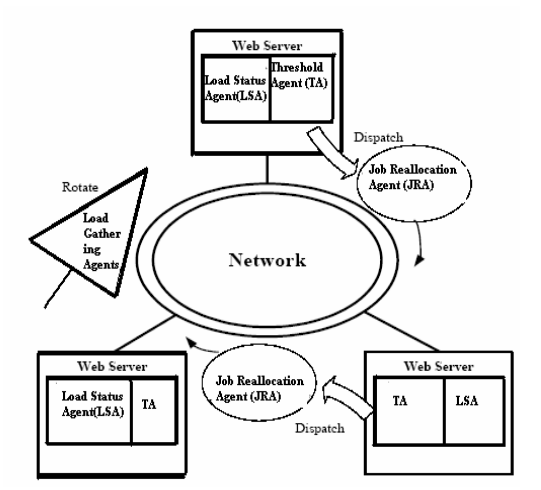


Fig 4: WLDMA functional architecture

LGA travels around the servers, collects the load information from the servers and propagates it to visiting servers. TA is a decision making agent that motionlessly sits at the server and collects all the LGA from other servers. TA calculates the adaptive threshold after receiving the LGA from the servers connected in the WAN cluster. It compares its queue size with other web servers queue size in the network and it also selects the web server with minimum queue size at that time with the updated information. JRA is activated by the TA and transfers jobs to the selected server which satisfies an equation 1. JRA aids in job re-direction. A server can dispatch JRA to the other system if it is required. The mobile agent approach can minimize the network traffic and enhance the flexibility of a load balancing mechanism. The functional architecture of WLDMA is as shown in Fig-4. The heavily loaded server attempts to transfer the job to lightly loaded server in sender-initiated policy. This policy is incorporated in WLDMA strategy.

Let *A*, *B*, *C* … be the web servers located at different sites in a WAN. Let $queueLength_A$, $queueLength_B$, $queueLength_{C\ldots}$ be the queue sizes of the web servers *A*, *B*, *C* … respectively, at a given point of time. The Mobile Agents are sent from *A*, *B*, *C* … to each other. The web server which sends a Mobile Agent to other web servers is considered as the source and the web server where those Mobile Agents are received and manipulated is considered as the destination. The Mobile Agents carry the queue size of the source web server. This value is compared with its value by the destination server, and job re-direction is performed based on the following algorithm:

Step 1: If $queueLength_{Destination} > (queueLength_{Source1}\ \&\ queueLength_{Source2}\ \&\ \ldots\ \&\ queueLength_{Sourcen})$ then,

Step 2: Compute q_x such that $q_x = \min(queueLength_{Source1}, queueLength_{Source2} \ldots queueLength_{Sourcen})$

Step 3: Compute *n* such that $n = (queueLength_{Destination} - q_x)/2$

Step 4: Transfer the last *n* jobs from $queueLength_{Destination}$ to q_x *server, if it holds equation 1.*

A job j on server x is reallocated to a remote server y only when:

$$\sum_{i=1}^{j} p_{ix} > j\_rt_x + j\_rp_{xy} + \sum_{i=n}^{i=1} p_{iy} + p_{jy} \qquad \text{----------}\ 1$$

Where,

p_{ix}=Processing time of ith request at web server x.
j_rt_x=Transmission time of jth request by web server x.
j_rp_{xy}=Propagation time of jth request from web server x to web server y.
p_{iy}=Processing time of ith request at web server y.
p_{jy}=Processing time of jth request at web server y.

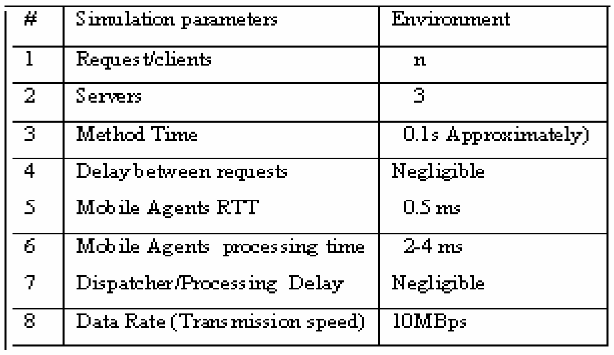
The purpose of computing *n* is to redistribute the jobs in order. In the existing WAN load balancing schemes, the job reallocation is done only when the workload on a server exceeds local threshold value [5]. In WLDMA, the job reallocation is based on the adaptive threshold where the node knows which node has the minimum load and decides to send a process to this node, unless its load after transferring the process is larger than the threshold in the distributed web server system.

LDMA SIMULATION MODEL

A software simulator was designed in C++ and implemented to model the LDMA load balancing technique in the distributed LAN web server environment. The workload of a replica is determined by the number of requests processed at each replica. LDMA is applied to minimize the workload difference between the replicas. Table-1 shows the simulation parameters and their default values used in the environment.

Simulation parameters governing the generation of client's events are summarized below:

- *Request/clients* is the number of request from the clients
- *Request processing time* is the average server processing time for completion of a single request.
- *Delay between requests* is the average time delay between requests made to each server.
- *Dispatcher delay* is the time taken for assigning the ID for requests and broadcast to cluster servers.
- *Mobile Agents processing Time* is the time taken by the agents for finalizing the type of message to be passed to the other server.



#	Simulation parameters	Environment
1	Request/clients	n
2	Servers	3
3	Method Time	0.1s Approximately)
4	Delay between requests	Negligible
5	Mobile Agents RTT	0.5 ms
6	Mobile Agents processing time	2-4 ms
7	Dispatcher/Processing Delay	Negligible
8	Data Rate (Transmission speed)	10MBps

Table-1: Simulation Parameters and their default values used in environment.

A network of servers is described by using two parameters

- *Servers* are the number of servers in a network that can process requests from clients.
- *Mobile Agents delay* is the time taken by the mobile agents to pass from one server to another and return with message.

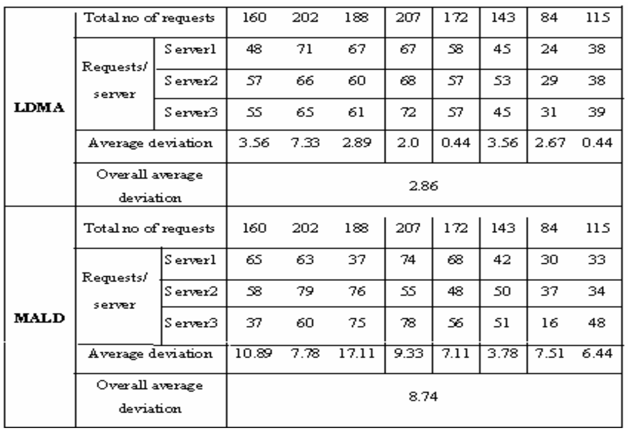
The performance of LDMA load balancing scheme is evaluated and compared with the MALD, an existing mobile agent based load balancing. In MALD, web servers dispatch mobile agents to retrieve load information of the servers and redirect the request to another server depending on the workload where as in LDMA, web servers dispatch the mobile agents to receive the message for deleting or processing the request. In this experiment, the performance of a load balancing scheme is assessed using the following criteria

Load distribution: The load on the server is denoted by the number of requests processed in the server. The average load distribution deviation over all servers is calculated to show the effect of load balancing.

System throughput: the overall throughput of the web server cluster is measured by the number of requests processed per second.

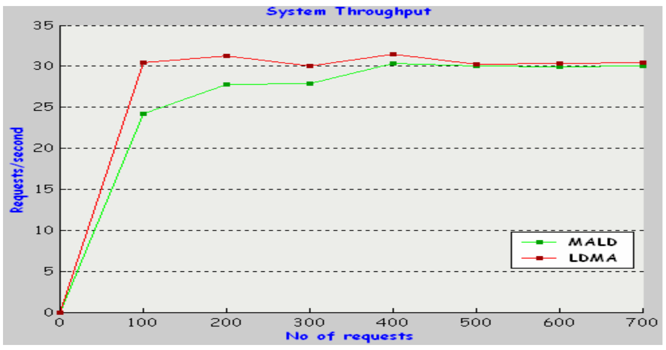
Network traffic: the overall communication overhead in the cluster is measured by the total number of data (bytes) transferred in the communication

In the experiment, every server processes the client requests independently. The load distribution generated by the LDMA and MALD scheme on three servers at different moments is as shown in Table -2.It also includes the average deviation of workload on the three servers. The system throughput of LDMA and MALD is as shown in graph-1. The system throughput of LDMA is closer to the MALD in all the cases. LDMA system's throughput remains constant approximately around 30 requests/seconds, while MALD system's throughput is lesser for minimum number of requests and improves gradually for higher number of requests. The overall deviation of the LDMA scheme is three times lower than the MALD scheme that shows the better performance of the LDMA scheme.

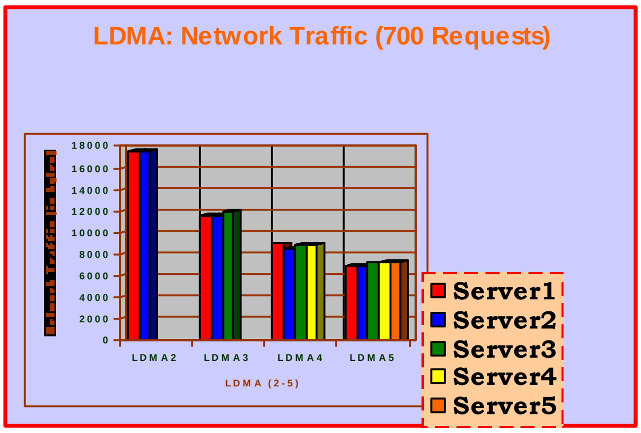


LDMA	Total no of requests		160	202	188	207	172	143	84	115
	Requests/ server	Server1	48	71	67	67	58	45	24	38
		Server2	57	66	60	68	57	53	29	38
		Server3	55	65	61	72	57	45	31	39
	Average deviation		3.56	7.33	2.89	2.0	0.44	3.56	2.67	0.44
	Overall average deviation		2.86							
MALD	Total no of requests		160	202	188	207	172	143	84	115
	Requests/ server	Server1	65	63	37	74	68	42	30	33
		Server2	58	79	76	55	48	50	37	34
		Server3	37	60	75	78	56	51	16	48
	Average deviation		10.89	7.78	17.11	9.33	7.11	3.78	7.51	6.44
	Overall average deviation		8.74							

Table-2: Simulation results of LDMA on three servers



Graph -1: LDMA and MALD system throughput



Graph -2: Network Traffic

LDMA	Total no of requests		160	202	188	207	172	143	84	115
	Requests/ Server	Server1	44	52	43	53	49	38	18	29
		Server2	41	42	47	55	39	33	23	28
		Server3	35	51	50	52	40	39	23	32
		Server4	40	57	48	47	41	33	20	26
	Average deviation		2.5	4.25	2.00	2.38	3.75	2.25	2.00	1.81
	Overall deviation		2.75							
	System Throughput		41	38	38	37	38	40	38	38
	Average system Throughput		38.5							
	Response Time (Sec)		2.18	2.69	2.29	2.78	2.23	2.02	1.26	1.63

Table -3: Simulation results of LDMA on four servers

LDMA	Total no of requests		160	202	188	207	172	143	84	115
	Requests/ Server	Server1	29	48	35	44	37	26	17	24
		Server2	37	41	46	42	33	32	19	23
		Server3	33	38	40	39	37	31	17	24
		Server4	34	39	34	44	33	29	15	20
		Server5	27	36	33	38	32	25	16	24
	Average deviation		3.20	2.80	4.40	1.84	1.80	2.60	1.80	1.20
	Overall deviation		2.43							
	System Throughput		47	49	50	48	47	51	46	47
	Average system Throughput		48.13							
	Response Time (Sec)		1.69	1.89	1.77	2.09	1.72	1.49	1.01	1.28

Table -4: Simulation results of LDMA on five servers

The simulation results of LDMA having four and five servers in the cluster are as shown in the table 3, 4 and 5. The performance of LDMA is improved when the number of servers in the cluster is increased is shown in table 3, 4 and 5.The average system throughput is considerably increased with respect to the number of servers in the cluster.

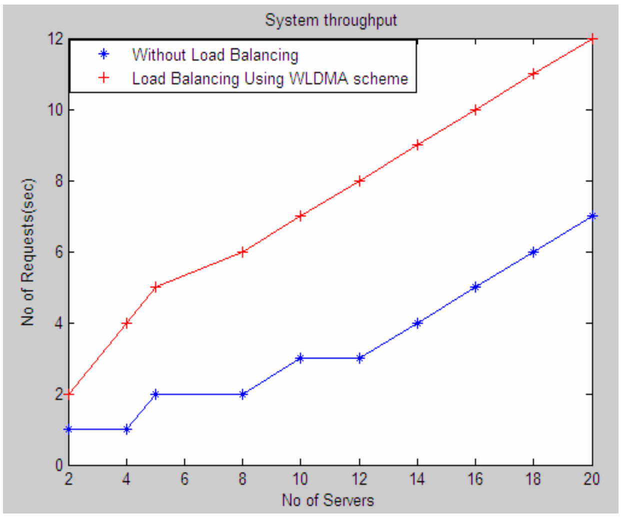
Graph 2 shows the network traffic of LDMA. The high network traffic largely restrains the performance of a web server system [7]. The overhead of the packet in LDMA is 50 bytes. The dispatcher broadcasts the incoming requests to all servers. The communication overhead for totally N servers is as high as O (N^2) [2]. The network traffic is measured by the total number of bytes transferred in the communication. Assume that the distributed web server environment consists of three servers in the cluster connected in mesh topology by a communication media. In the beginning, three servers are involved in decision making (message exchange) for processing the request takes 300 bytes communication overhead. The communication overhead of the subsequent requests depends on the previous request processing time and also, the requests deleted in the other server's queues using LDMA scheme. Hence, the communications overhead of the successive requests are reduced to 200, 100 bytes respectively.

Performance Evaluation

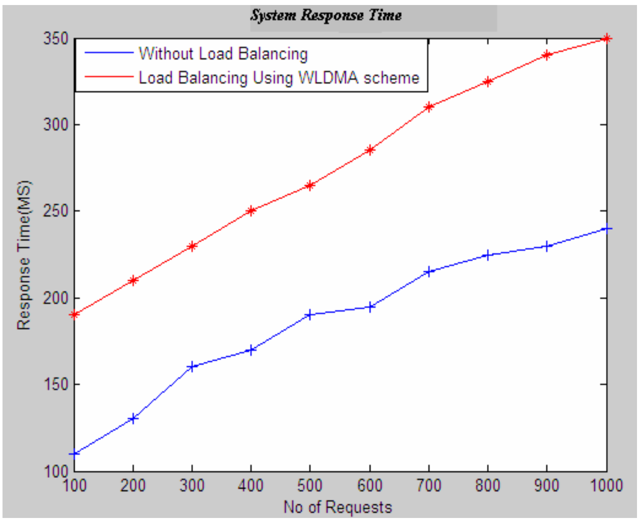
To study the performance of the WLDMA scheme on wide-area network, new simulation software was developed in C++. The performance of the WLDMA is analyzed for system throughput, response time and load deviation. In this experiment, client requests are generated and sent to the servers. The server receives the client requests independently. If a server is overloaded, the requests are redirected to minimum loaded server. The performances of WLDMA system throughput and response time are shown in graph 3 and 4 respectively. The simulation parameters and their default values are summarized in Table 5. The experiment reveals that the performance of Load distribution on three servers using WLDMA is two times better than the scheme without load balancing.

Simulation Parameter	Value
Number of web servers	3
Task processing time	10 ms
Propagation delay	50 ms
Transmission delay	20 ms
Mobile Agents Round trip delay Time	150 ms

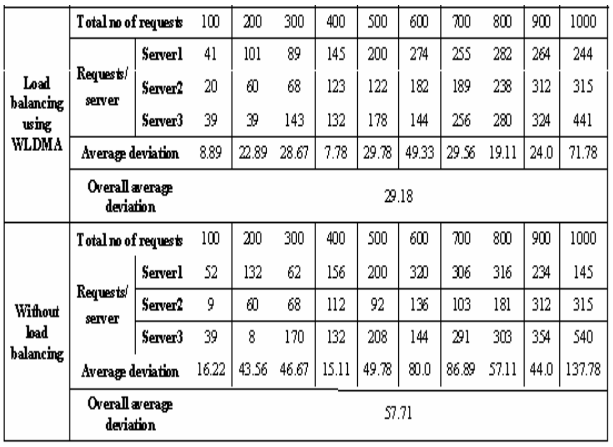
Table 5: Simulation parameters used



Graph 3: System throughputs of the LDMA scheme and the case without load balancing



Graph 4: System response time of the WLDMA scheme and the case without load balancing



Load balancing using WLDMA	Total no of requests		100	200	300	400	500	600	700	800	900	1000
	Requests/ server	Server1	41	101	89	145	200	274	255	282	264	244
		Server2	20	60	68	123	122	182	189	238	312	315
		Server3	39	39	143	132	178	144	256	280	324	441
	Average deviation		8.89	22.89	28.67	7.78	29.78	49.33	29.56	19.11	24.0	71.78
	Overall average deviation		29.18									
Without load balancing	Total no of requests		100	200	300	400	500	600	700	800	900	1000
	Requests/ server	Server1	52	132	62	156	200	320	306	316	234	145
		Server2	9	60	68	112	92	136	103	181	312	315
		Server3	39	8	170	132	208	144	291	303	354	540
	Average deviation		16.22	43.56	46.67	15.11	49.78	80.0	86.89	57.11	44.0	137.78
	Overall average deviation		57.71									

Table- 6: Load distribution on three servers

The simulation parameters governing the events are summarized below

- Web Servers are the number of servers in a network that can process requests from the clients. These servers are widely separated across the WAN environment.
- Task processing time is the time taken for executing the request alone.
- Propagation delay is the time required for a request to travel from one point to another.
- Transmission delay is the time taken from the start of request reception to the end of request reception.
- Mobile Agent round trip delay time is the time taken by the mobile agent to travel between the servers on the network.

The result shows that the WLDMA scheme can obviously improve the system throughput when increasing the number of servers. On contrary, there is not obvious improvement of the throughput in the case without load balancing. In the latter case, the processing capacities of the servers are wasted. Table 6 compares the load distribution generated by the WLDMA scheme and running without load balancing on three servers at different moment. Table 5.2 also includes the average deviation of load on the three servers. It shows that the WLDMA scheme has lower load deviation than the running without load balancing scheme in most of the cases. That means WLDMA can distribute client requests more evenly onto the web servers. The overall average deviation in Table 6 is the mean of average deviations at all moments. The overall average deviation of the WLDMA scheme is lower than the running without load balancing strategy that verifies the better performance of the WLDMA scheme in supporting load balancing.

6. CONCLUSION

This paper examines two new algorithms for improving the performance of a distributed system through load balancing a workload in the distributed client-server architecture. LDMA framework possesses several advantages. First, decision making in cluster is decentralized and response time improves as the number of replicas increase. Second, use of mobile agents imposes the merits of high flexibility, low network traffic and high asynchrony. Third, the result shows that no replica remains idle at any time while the other replicas are processing more than one request each. In LDMA, the requests start their execution processing in arrival order irrespective of their processing time. This method has still some drawbacks. First, the usual approach of job transfer from an overloaded web server to an under loaded web server still persists, which is tedious. But, since the WLDMA algorithm redirects jobs only when the mentioned equation holds good, all servers remain busy for the same amount of time. Thus, WLDMA framework proves to be effective in distributing the workload among web servers in a WAN environment

- Transmission delay is the time taken from the start of request reception to the end of request reception.
- Mobile Agent round trip delay time is the time taken by the mobile agent to travel between the servers on the network.

The result shows that the WLDMA scheme can obviously improve the system throughput when increasing the number of servers. On contrary, there is not obvious improvement of the throughput in the case without load balancing. In the latter case, the processing capacities of the servers are wasted. Table 6 compares the load

distribution generated by the WLDMA scheme and running without load balancing on three servers at different moment. Table 5.2 also includes the average deviation of load on the three servers. It shows that the WLDMA scheme has lower load deviation than the running without load balancing scheme in most of the cases. That means WLDMA can distribute client requests more evenly onto the web servers. The overall average deviation in Table 6 is the mean of average deviations at all moments. The overall average deviation of the WLDMA scheme is lower than the running without load balancing strategy that verifies the better performance of the WLDMA scheme in supporting load balancing.

Conclusion

This paper examines two new algorithms for improving the performance of a distributed system through load balancing a workload in the distributed client-server architecture. LDMA framework possesses several advantages. First, decision making in cluster is decentralized and response time improves as the number of replicas increase. Second, use of mobile agents imposes the merits of high flexibility, low network traffic and high asynchrony. Third, the result shows that no replica remains idle at any time while the other replicas are processing more than one request each. In LDMA, the requests start their execution processing in arrival order irrespective of their processing time. This method has still some drawbacks. First, the usual approach of job transfer from an overloaded web server to an under loaded web server still persists, which is tedious. But, since the WLDMA algorithm redirects jobs only when the mentioned equation holds good, all servers remain busy for the same amount of time. Thus, WLDMA framework proves to be effective in distributing the workload among web servers in a WAN environment.

References:

[1].W. Winston (1977), “Optimality of the Shortest Line Discipline “, in Journal of Applied Probability pp.17-28.

[2]. Jiannong cao, Yudong Sun, Xianbin Wang and Sajal K. Das (2003), “Scalable Load Balancing on Distributed Web Servers Using Mobile Agents”, in Journal of Parallel and Distributed Computing, Vol.63, Issue 10. pp. 996–1005.

[3].Marco Conti, Enrico Gregori and Fabio Panzieri,(1999) “Load Distribution among Replicated Web Servers: A QoS-based Approach.

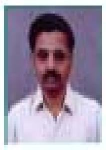
[4] Asser N. Tantawi and Don Towsley,(1985)"Optimal Static Load Balancing in Distributed Computer Systems" Journal of the Association for computing machinery, Vol.32, No.2 , pp. 445-465, April .

[5] Majeed M. Hayat, Sagar Dhakal, Chaouki T. Abdallah, J. Douglas Birdwell, and John Chiasson (2003)., " Dynamic time delay models for load balancing part II: A stochastic analysis of the effect of delay uncertainty. CNRS-NSF Workshop: Advances in Control of Time-Delay Systems, January 2003.

[6] Kameda H., El-Zoghdy S. F., and Li J., (2001)" A performance comparison of dynamic Vs. static load balancing policies in a mainframe-personal computer network mode," published in ISE Technical Report, ISE-TR-01-183, September.

[7] Milan E. Soklic (2002), "Simulation of Load Balancing Algorithms: A comparative Study," SIGCSE Bulletin, Vol.34, No.4, pp. 138-141, Dec.

M. Aramudhan received the BE. and M.E. degrees in Computer science Engineering from Regional Engineering college in 1997 and 2001, respectively. Now, he is research scholar in Anna University. His area of interests are distributed system, Network security and performance evaluation of networks.



V.Rhymend Uthaiaraj received his PhD degree from Anna University. He is working as professor & Head , Information Technology department, Anna University. His area of interests are Network security,Optimization and Pervasive computing.