International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 12 4223 - 4229

Application of Linear Programming in Scheduling

Ajay Tiwari
School of Computer Science
Devi Ahilya University (DAVV)
Indore, India
e-mail: tiwariajay8@yahoo.com

***Abstract*—** Distributed computing virtually combines the scattered interconnected computing resources and satisfies the demand of compute-bound and data-hungry applications. The paper highlights various Distributed Computing Environments (DCE), scheduling techniques and need of incorporation of Dynamic Load Balancing (DLB) in scheduling. The paper also opens a new area of research by introducing Linear Programming as a scheduling technique in DCE.

Keywords-*Linear programming, distributed computing; cluster; scheduling; load balancing.*

*****

I. INTRODUCTION

From 1994 to 1998, CPU clock speeds rose by about 300% and it was expected by the processor manufacturers that in near future processors clock speed will reach increase up to of 10.0 Ghz with processing capability of one trillion operations per second. But with the increase of clock speed, processors consumed more power and generated more heat. This extra heat was barrier to speed acceleration of CPU. Therefore from 2007 to 2011, maximum CPU clock speed rose from 2.93 GHz to 3.9 GHz i.e. an increase of just 33%. Later on, the improvement in the processor's performance was observed with multi-core processors. In multi-core architecture, a processor possesses more than one CPU, each having restrained clock speed which provides hardware parallelism and is named as Chip Multi Processing (CMP). Prior to CMP, HPC community used Symmetric Multi Processors (SMP). The main drawback of SMP is that, cores of SMP communicate through motherboard whereas lying on the same die, cores of CMP communicate faster. Use of multi-core nodes made clusters more popular and has attracted HPC community due to better cost-to-performance ratio.

The growth in network speed, storage size, processing power and Internet usage has changed the way of managing information. Geographically distributed resources are interconnected and can be accessed by the user as a single service point. These resources may be supercomputer, storage devices, workstations and nodes interconnected across LAN, MAN and WAN. Resource-intensive or repetitive tasks can be outsourced and the task is executed optimally with respect to cost, execution time etc. The third parties offer computing infrastructure as a utility service where the users are charged only for resources actually used [7]. As a result, the user's requests on servers are growing exponentially in last few decades. These servers host business or scientific applications for which customers expect the guaranteed 24x7 service and refuse to accept any delay or failure [1]. In the age of cut-throat competition, loosing even a single customer is not acceptable for a commercial website.

The rest of the paper organizes as follows. Section II describes various DCEs, need of incorporation of DLB is discussed in section III. Section IV characterized scheduling techniques which may be very useful for business and scientific applications. Linear programming is introduced as scheduling technique in section V.

II. DISTRIBUTED COMPUTING ENVIRONMENT

Technical and enterprise customers had been solving their computational problems through uni-processor computers for years. With the time the processing needs of these customers increased from simple applications to complex ones. Examples of such applications include oil and gas discovery, automotive design, climate predictions, life sciences and biotechnological simulations, aerospace engineering, financial forecasting, data mining and data ware housing etc. Multiprocessor and super computers were evolved to improve the performance of uni-processor systems to meet the demand due to increased processing needs of the customers. However, due to excessive cost, the solution was not feasible for medium and small organizations. At the same time, there has been a rapid growth in networking technology so that cheap and sophisticated computing resources viz. processing power, storage size etc. became available. A cost effective solution was to combine the resources of two or more computers. The approach gave rise to distributed computing and opened a new area of research where efforts were made to utilize interconnected computing resources [2] [11].

Distributed computing consists of a collection of interconnected autonomous resources that appears to its users as a single coherent system. Resources of distributed computing are connected by high speed LAN, MAN or WAN. The distributed computing system is transparent i.e. client is not aware of process is being executed. In such systems the chances of failure are very less even if a resource goes down. The main objective of distributed computing is to serve users by providing resources in a transparent, cost effective, secure, reliable and scalable way. The resources shared by the system may be computational power, storage devices, communication capacity, license S/W, OS, critical services available on servers etc. In the journey of fast and efficient computing, a variety of distributed computing systems were evolved over a period of time. Distributed computing is used by business organizations and scientific institutions to address these needs.

In Figure 1.1, it is shown that distributed computing system comprises of application (client) layer, middle layer and resource layer. The application layer is used to receive the information from the user, send/receive the request/response to the middle layer and display the response. The middle layer is transparent to the user and is responsible for scheduling, load balancing, gathering statistics and monitors various activities. Statistics is used for billing and other administrative tasks and

provides interface between application layer and resource layer. Resource layer comprises of resources placed at centrally dispersed locations.

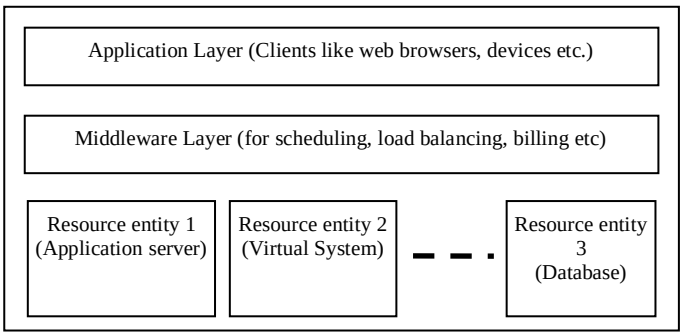

Figure 1.1 Layered Architecture of Distributed Computing System

Network of Workstations (NOW), clusters, server farms, grids and clouds are some of the examples of distributed computing which are briefly described as follows:

A. *Network of Workstations (NOW)*

NOW utilizes resources of loosely coupled nodes of the system while primary users keep on using their computers. When nodes of the NOW are dedicated, the distributed system is known as Beowulf type cluster. As NOW uses unused resources of the existing nodes, it provides a cost effective solution to small scale scientific institutions. It has been observed that between 1:00 AM to 8:00 AM only 5% keyboard and mouse activities are performed and during these night hours 80% time of the nodes are idle. Several authors proposed a solution to build high performance computing environment by using the power of several unused individual workstations, powerful PCs in night hours and the system is named as Super Computer at Night [9] [10].

B. *Cluster*

Cluster is an example of distributed computing system used for jobs with high computing needs and is a cost effective and fail safe alternative to the super computers. As shown in Figure 1.2, cluster is a collection of interconnected nodes working together usually on a homogeneous platform. Cluster technology allows building of a super computer by using simple computers which are connected via high speed network and off-the-shelf technology. The cluster computing is financially and technically attractive due to low price /performance ratio. Clusters are more or less homogeneous in terms of hardware and S/W as compared to grids and cloud. Clusters comprise of nodes controlled by master node connected through high speed Local Area Network (LAN). Master node distributes jobs to the nodes and provides interface to the client. It is responsible for overall management of the cluster. Multiple nodes of the cluster create a fail-safe environment as the system continues to work even after failure of some of its nodes.

C. *Server farm*

Server farm (also known as server cluster) is a collection of replicated servers connected via high speed LAN and housed in one location. All these servers communicate to the outside world via a virtual server. Virtual server is a single point entry. Therefore, it also works as a scheduler also. Server farms are used to manage enormous amount of computerization of tasks and services through web sites.

D. *Grid Computing*

Unlike cluster computing, Grid computing is the collection of resources where each resource falls under the different administrative control [6]. The resources from different organizations/institutions are brought together to form a grid. The resources may be super computers, clusters, database servers, high performance devices etc. The resources of the grid belong to different administrative control hence there is a high degree of heterogeneity in terms of hardware, operating system, network, security policies etc.

E. *Cloud Computing*

Cloud computing is well known distributed computing technology where user is required to install an application program which interacts with cloud. Cloud is a web based service which hosts all kind of programs, as simple as e-mail or as complex as heavy scientific or share market applications. The users of cloud computing need not install the updated software or hardware; they simply login to the cloud of their choice and get the service. Cloud's front end is the application program which resides on the user's computer and accesses the cloud computing system, whereas the cloud's back end comprises of various types of servers like http, https, ftp, e-mail, storage servers, gaming servers etc. which actually farm cloud. From one point of view, cloud computing is not new because it uses similar concepts, approaches, and best practices that have already been established but from another perspective, it's a new technology which works on on-demand, self-service and pay-by-use nature [6].

III. SCHEDULING USING DYNAMIC LOAD BALANCING

As discussed in the early sections, Distributed Computing Environment (DCE) is an interconnected collection of number of computing resources. These resources serve the processing demand of large number of clients. If the client requests are not evenly distributed among the resources, some of the resources would be heavily loaded whereas other would be under loaded and the overall performance of the system is poor. For the effective distribution of client's requests, a middle ware is needed between client and servers. This middleware receives clients request and forward them to appropriate server according to some predefined logic and is known as *Load balancer or Scheduler*. Load balancing can be effectively used to balance the workload on DCE [12]. A load balancer can distribute connections among two or more servers, proportionately rationalizing the work every server has to do [3] [13]

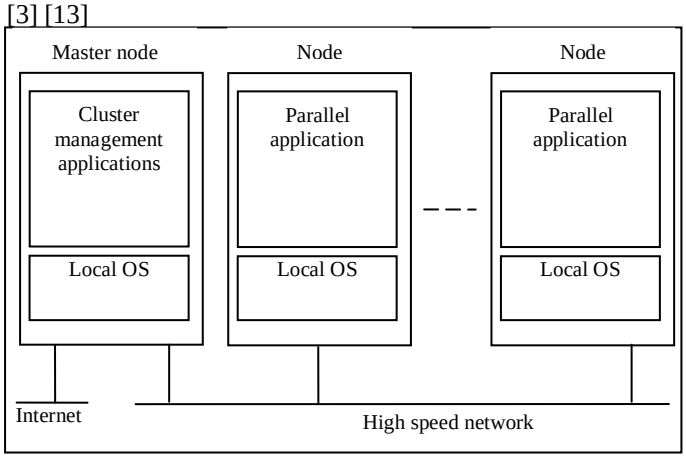

Figure 1.2 General Cluster Architecture

Load balancing can be categorized as: *Static, dynamic and adaptive*. The difference between *static and dynamic* load balancing lies on the use of run time state information of the resources of DCE. While static load balancing does not use run time information, dynamic load balancing use current state information of the resources to direct the request to the appropriate resource. *Adaptive* load balancing is the subset of dynamic load balancing that uses the run time information to update the parameters of scheduling algorithm or chooses different algorithm for different set of run time systems information. Each load balancer collects the information of the connected nodes by using health monitoring activity periodically. Health monitoring helps in finding the availability and load of nodes as well as for applications running on these servers. The working of a general load balancer can be described as follows:

Step 1. The client sends request to the virtual server which is usually a load balancer.
Step 2. The load balancer forwards the request to the most appropriate server after replacing the destination IP and/or port (if required).
Step 3. The server processes the request and sends the response to the load balancer which is its default route.
Step 4. Now the load balancer changes the source IP of the response packet from processing servers IP to its own IP and forwards packet back to the client.
Step 5. The client accepts the packet and the process continues.

Step 2 and step 4 are performed using Network Address Translation (NAT) mechanism. In step 2, load balancer replaces the destination IP of the packet sent by the client with the destination IP of the selected server to process the request. The source address of the response packet is the virtual servers address. Packet will not be accepted by the client where request was not made. Therefore, load balancer replaces the source address with its own address by using NAT.

Load balancing is a critical activity to manage the load on high performance Web server cluster. Requested load must be distributed among the nodes of the Web cluster to reduce the response time and provide the best possible service to the Web customers. Load balancer not only attempts to equalize the load but also provides other important features like high availability, server security, scalability etc.

i) **High Availability:** If a commercial application is not available for even a very short duration, organization incurs heavy losses in terms of revenue, loyalty etc. Load balancing improves uptime of critical application servers, database servers and Web servers and hence assures the availability of the servers. For mission critical applications, load balancers support standby failover mechanism i.e. if one server fails another server takes over. The switching is stateful and for connection-oriented applications, users need not restart the session.
ii) **Security:** As all client requests are handled by load balancer, there is no threat for back end servers. Clients don't have various details about back-end servers viz. IP, content etc. If enabled, load balances may work as firewall also.
iii) **Scaling:** There are two approaches for scaling. In the first approach, existing servers are upgraded and in the second approach new servers are added to the server farm. Network administrators choose one of these or combination of both approaches to make it cost and service effective. The key variables that affect client response time are application capability, server capability, network bandwidth and job size. Before improving the scalability, network administrator should check the performance bottleneck caused by server application capability, network bandwidth or job size. Network bandwidth and server capability should be synchronized i.e. increasing the server performance is not useful if the bottleneck is caused by inadequate bandwidth and vice versa. Moreover while calculating average response time or average turnaround time, job size should be considered.

IV. Classification of Scheduling Techniques

The most critical task in distributed computing is selection of resources for client requests. Cardellini et al. have proposed a classification of existing scheduling techniques based on the entity that dispatches the client requests among the distributed systems [14]:

- Client-based/Sender-based approach
- DNS-based approach
- Receiver based approach: A receiver based approach may be server-based approach or scheduler-based approach

All these approaches satisfy the desirable features of distributed system like system transparency, scalability (both local and global), availability and applicability to existing web server standards.

A. Client-Based/Sender-Based Approach

In a distributed computing system, selection of resource can be achieved at client side using web clients (web browsers) or by client side proxy servers. In web client's approach, the client decides the server where the request will be processed. Following procedure may be used:

Step 1. The request is generated at the client
Step 2. A resource is selected by the client and request is sent to the selected resource
Step 3. Response is sent by the resource to the client

Figure 1.3 shows the client side scheduling where user submits the request to the client (Step 1), client identifies the resource (Step 2) and the request is submitted to the identified resource after establishing the connection and finally (Step 3) the response is received from the resource.

Mosedale et al. have discussed client side scheduling technique for distributing the load across the multiple Netscape servers for Netscape sites through its own Netscape browser. When user requests for the Netscape home page through Netscape browser, the browser generates a random number i between 1 and number of Netscape's server n, and routes the request to the i^{th} server at wwwi.netscape.com. In this manner, the load is distributed randomly in server cluster and no node in the system is heavily loaded [5]. This approach cannot be generalized because it is not possible for all the websites to have their own browsers for distributing the requests. But this solution is useful for the organizations having Intranets.

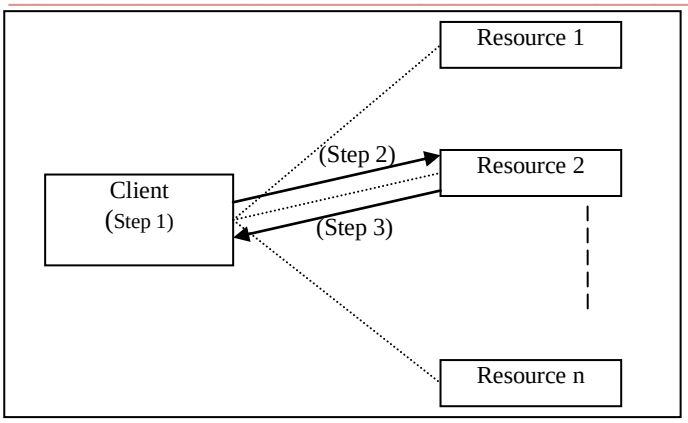

Figure 1.3 Web Client Based Scheduling

Yoshikawa et al. have proposed a similar concept known as Smart client. To collect the current state of the nodes, smart client runs a light weight process Java applet on the nodes of the cluster. For collecting the information of the nodes, smart client needs the address of the nodes of the cluster. Before sending the request, the client collects information regarding the nodes in the cluster and sends the request to the suitable node [4]. A major drawback of this approach is the increase in the network traffic due to message exchanges and difficulty in providing the address of all the nodes of the clusters.

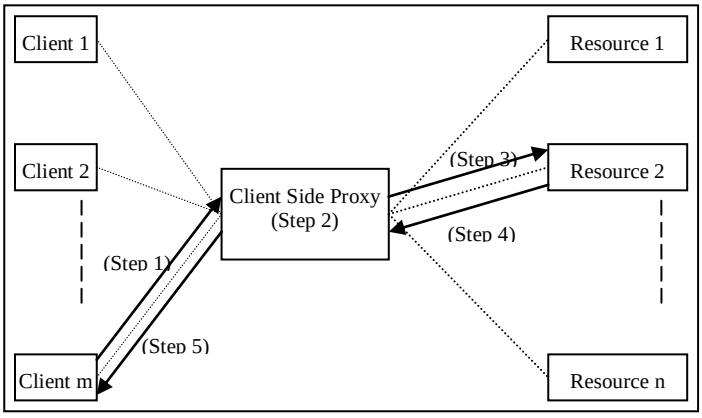

Figure 1.4 Client Side Proxy Based Scheduling

As depicted in Figure 1.4, in client side proxy approach, client's proxy decides the node where the request will be processed. Following steps are involved in the procedure:

Step 1. The request is generated at the client and sent to client's proxy
Step 2. If the required object is in proxies cache, it responds to the request as in Step 5
Step 3. Otherwise resource is identified by the proxy and request is sent to the selected resource
Step 4. Response is received by the proxy and proxy cached the response if TTL is not zero
Step 5. Proxy responds to the client

The main limitation of client based approach is to install the setup for selecting the resource from number of replicated resources. This reduces the load at server side. However it is not suitable for situation where individual client acquires the value of attributes of resources needed for work distribution. The approach may be feasible for LAN and WAN having a sufficient number of clients and client side proxy but it is not appreciated for individual client.

B. DNS-Based Approach

In July1996, Cisco launched first IP based server load balancer. Prior to this load balancing was achieved via DNS based round robin where DNS server returns IP address of one of the server from list of replicated servers to a DNS query. Whenever higher server capacity is required, a mirrored server is added to server farm and its IP address is added for the rotation to the DNS. In first implementation of cluster side scheduling solution in distributed computing, the responsibility of distributing the request in cluster is given to the authoritative DNS. When a request for IP having multiple replicated servers is received by a DNS, DNS selects a server from available servers corresponding to a URL and returns its IP. The selection process of the server uses variety of scheduling policies. DNS performs the following process for translation:

Step 1. Client sends the request for IP address translation to intermediate name server
Step 2. If the required IP is in intermediate name server, name server sends it to the client and goes to Step 5
Step 3. Otherwise IP address and TTL value of resource is selected out of number of replicated resources by the authoritative name server
Step 4. The intermediate name server caches the IP address and responds to the client
Step 5. Client sends the request to the received IP address i.e. actual server
Step 6. Server sends the respond to the client

In the above steps, DNS based scheduling is performed at Step 3, where authoritative name server resolves IP address of server. For scheduling, it uses server stateless, server state full, constant TTL, variable TTL or mix of these algorithms.

The DNS load balancing method introduces several problems. It distributes load in rotation without considering server's load due to which some servers of the server farm receive heavy load and other servers are lightly loaded decreasing overall performance of server farm. To improve the performance, an additional server is needed. Moreover, DNS server doesn't have any knowledge of status of server, applications running on it or number of TCP connections. If a server is unavailable due to failure or any other reason, DNS will keep on sending the requests to the unavailable server and user receives error message. Even if server's IP address entry is removed from DNS by the caching mechanism of DNS entry, users get the correct server after a number of attempts. In some situations, DNS scheduling is not effective. For example due to the TTL value set by the authoritative name server instead of authoritative name server, intermediate DNS performs the IP translation. As scheduling is performed by authoritative DNS not intermediate DNS, no requests could be scheduled. To avoid this, the value of TTL is set to zero by authoritative DNS which results network overload. Despite of these problems, DNS scheduling is used widely as it is cheaper and easy to implement [14].

C. Server Based Approach

Whereas DNS based approach works at URL level scheduling, server based approach accomplished at HTTP or IP level. In server based approach, after receiving the client's

request, either server processes the query or redirects the request to some other lightly loaded server. Thus, each server in the server farm acts as a scheduler as well as a Server. For scheduling, servers maintain the scheduling information of other servers. The redirection of requests can be performed either using http redirection by the server or using packet redirection by the server.

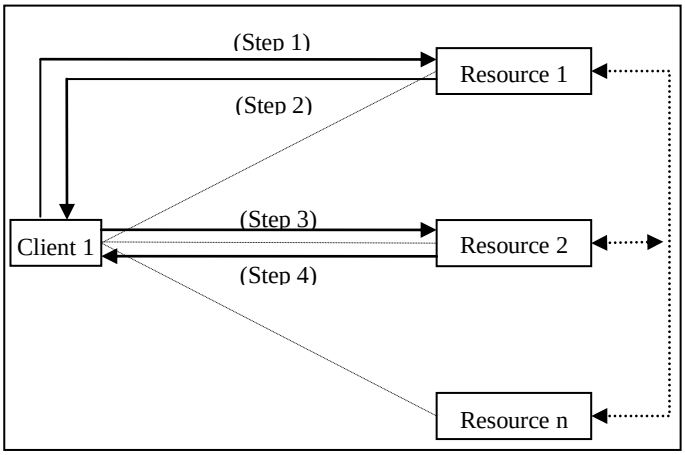

Figure 1.5 http Redirection by the Server

As shown in Figure 1.5, in http redirection by the server, the following steps are performed:

Step 1. The request is generated at the client and after resolving IP address, request is forwarded to the server.
Step 2. If the server finds some lightly loaded nodes in the server farm, the http redirection is done.
Step 3. Client sends the request to redirected server.
Step 4. Client receives the response from redirected server.

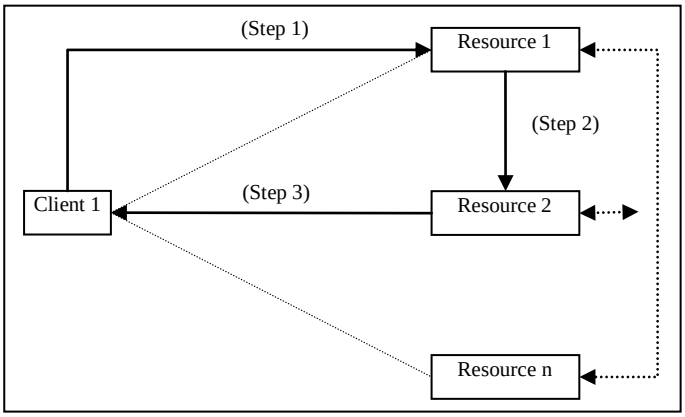

Figure 1.6 Packet Redirection by the Server

Similarly, as depicted in Figure 1.6, in IP redirection by the server, the following steps are involved:

Step 1. Request is generated at the client and after resolving IP address, request is sent to the server.
Step 2. If the server finds some lightly loaded node X in the server farm, the IP redirection is performed i.e. the request is forwarded to the X transparently.
Step 3. Client receives the response from redirected server.

D. Scheduler Based Approach

Scheduler based approach accomplishes the load balancing at IP level. In this approach, a scheduler is placed between clients and server farm. Clients send their requests to the scheduler and the scheduler forwards these requests to the most appropriate server in the server farm. Using some scheduling algorithm, the IP address assigned to scheduler is known as virtual IP as the communication takes place between a client and a server behind the scheduler. The scheduler is a single point entry to the server farm which has its own merits and demerits. Scheduler based approach follows two methods for responding to the clients.

i) Scheduler Return
ii) Direct Server Return

In scheduler return, scheduler receives the request from the client; gets it processed by some of the server from the server farm and responds to the client (double packet rewriting method). In direct server return method, scheduler receives the request from the client, forwards it to the selected server and instead of responding to the scheduler, server responds to the client directly (single packet rewriting method) to reduce the workload of scheduler.

As shown in Figure 1.7, Scheduler Return based approach follows the following steps:

Step 1. The request is generated at the client and sent to scheduler
Step 2. Scheduler selects a server based on the given criteria and forwards the request to the server
Step 3. Server processes the request and sends response to the scheduler
Step 4. Scheduler forwards the response to the client

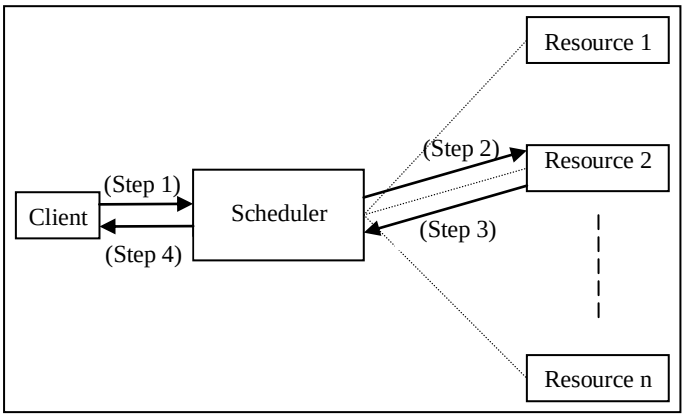

Figure 1.7 Scheduler Return Approach

As depicted in Figure 1.8, direct server return method performs the following steps to avoid the bottleneck at scheduler:

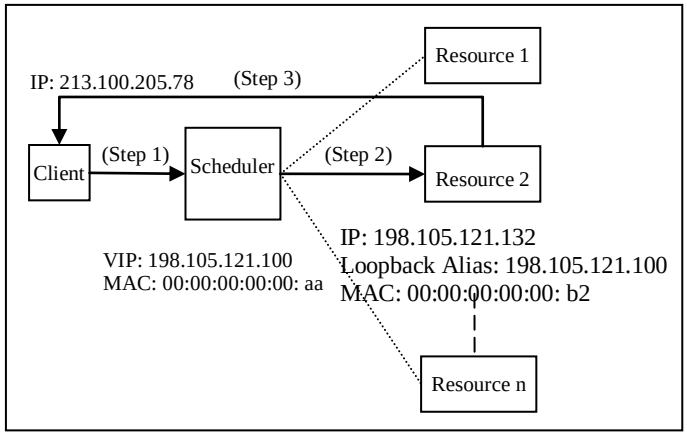

Figure 1.8 Direct Server Return

Step 1. The request is generated at the client and sent to scheduler

Step 2. Scheduler selects the server based on the given criteria and forwards the request

Step 3. Server processes the request and sends response to the client directly

Step 3 faces a problem where client will discard the packets received from server as it is expecting the response from the scheduler and not from the server. The problem is solved by using MAC Address Translation. Às shown in Figure 1.8, in Step 1, the packets are sent from client's IP 213.10.205.78 to scheduler's VIP 198.105.121.100. In the server farm, all the servers are assigned one addition alias IP of scheduler. In Step 2, only MAC address is rewritten from 00:00:00:00:00:aa to 00:00:00:00:00:b2 and packets are delivered to real server having MAC address 00:00:00:00:00:b2 with source IP address 213.10.205.78 and destination IP address 198.105.121.100. In Step 3 the server receives the packets as its IP and MAC address matches and responds to the source IP (clients) by keeping its own alias address as source address. The client accepts the response as the response is from IP 198.105.121.100, where it had sent the request.

This method has a number of advantages over others methods viz. the control is at server side and is easy to maintain; and less network overload to maintain server states at scheduler.

V. SHEDULING USING LINEAR PROGRAMMING

There are situations when the decision makers need to allocate resources to various activities with the objective of improving the response time or processing time. No difficulty is experienced if the resources are in access. Normally, the situation is adverse when decision makers have to perform various activities within limited resources. Therefore the only way to improve the performance is optimal allocation of resources to the activities. Linear Programming (LP) model is an optimization model of resource allocation for choosing the best alternative from a set of given alternatives [2] [8].

The mathematical representation of the LP model is given as:

Maximize (Minimize)

$$Z = \sum_{i=1}^{n} c_i x_i$$

Subject to

$$\sum_{j=1}^{n} a_{ij} x_j \begin{matrix} \leq \\ = \\ \geq \end{matrix} b_i \qquad for\ i = 1,2,\ldots,m$$

and non-negativity condition

$$x_j \geq 0 \qquad j = 1,2, \ldots., n$$

Where c_i, a_{ij} and b_k are *constants* and x_i*'s* are *decision variables*. c_i*'s* are known as *profit or loss coefficients*, a_{ij}*'s* are known as *technological constants* and b_k*'s* are known as *resource values*.

The model optimizes the objective function under given constraints. In other words, the model utilizes the resources optimally therefore the profit given by the objective function will be maximum.

A. Motivation behind Use of Linear Programming

In distributed system, the nodes are allotted by the scheduler to client requests with some predefined criteria. The objective of the scheduler is to minimize the response time or maximizes the throughput of the overall system. The study correlates this scheduling problem with LP model and identifies the objectives and constraints:

- The objective of the scheduler is to maximize overall throughput of the system and the objective can be represented as a linear function of decision variables. The decision variables are allotted processing time of requests on a particular node (p_{ij}).
- Constraints may be
  - i) Resource utilization should be Maximum.
  - ii) No process can be allotted more than one unit of time.
  - iii) Non negativity condition holds as processing time cannot be negative.

Hence, LP model can be used by scheduler for work distribution in server cluster system. As optimal resource utilization is one of the objectives of the LP model, the assumption is that the overall performance of system will be improved.

B. Linear Programming Formulation

For linear programming formulation of the problem, let:

- n is the number of request category and m is the number of nodes.
- w_i is the service rate ratio (weight) associated with the category-class i.
- p_{ij} is the rate at which category-class i request is serviced by node j
- t_{ij} is the time duration for which category-class i request is serviced by node j.

If processing rate and service time are represented by P and T, then the matrix representation of P and T is given as:

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & a_{nm} \end{bmatrix}$$

$$T = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{n1} & \cdots & t_{nm} \end{bmatrix}$$

The linear programming objective of the model is to maximize the overall throughput of the sytem, say Z, while ensuring fairness to each category-class and can be given as:

$$Z = \sum_{i,j} (p_{ij} * t_{ij}) \ \ where\ i = 1,2 \ldots n\ and\ j = 1,2 \ldots m \quad (1)$$

Subject to the constraints:

Resource utilization: If resource utilization of the system is maximum, the performance of the system is optimum. In this case, maximum resource utilization means, servers of the cluster are never idle i.e. algebraic sum of service times of same server should be 100%.

$$\sum_j \ t_{ij} = 1 \qquad \forall\, i \qquad (2)$$

and

$$0 \le t_{ij} \le 1 \qquad \forall\, i,j \qquad (3)$$

Fairness: As resources of the system under consideration are of heterogeneous nature, fairness insures that all requests get equal opportunity on resources. For the given model, the fairness can be achieved if the ratio of sum of product of processing speed and service time for a category-class with associated service rate ratio is constant and similar for all category-classes i.e.:

$$\frac{\sum_{k=0}^{n} p_{ik} t_{ik}}{w_i} = \frac{\sum_{k=0}^{n} p_{jk} t_{jk}}{w_j} = K \qquad \forall\, i,j \ and\ i \neq j \qquad (4)$$

By solving the objective function given in eq. (1) one can find the optimal allotted time [tij] to the processes under the constraints stated in eq. (2), eq. (3) and eq. (4) which results overall performance improvement of the system.

VI. Conclusion

In distributed computing system, performance issues have become more critical due to proliferation of heterogeneous resources and variety of communication medium. The paper proposes a linear programming based load balancing policy for DCE. There is no constraint about processing power of the servers (servers may be of different processing power), request handling etc by the proposed model. Scheduler in consideration uses deterministic linear programming model to identify the resources of the WSC for optimal allocation of client requests which may be very useful for all kind of applications.

Acknowledgment

I would like to give my heartfelt thanks to my Ph D supervisor Dr. Priyesh Kanungo for taking on multitude of roles that provided guidance and direction for this research. I would like to express deep sense of gratitude for his truly insightful thoughts that envisioned the topic of the paper.

References

[1] A Karimi, Z Faraneh , A Jantan and A R Ramli, "A New Fuzzy App.roach for Dynamic Load Balancing Algorithm," International Journal of Computer Science and Information Security (IJCSIS), Vol. 6, No 1, 2009, pp. 1-5.

[2] A Tiwari, "Effective Mechanisms for Dynamic Load Balancing under Heterogeneous Environment in Distributed Computing," *Ph D Thesis, School of Computer Science & IT, Devi Ahilya University, Indore, MP, India*, 2014.

[3] B Jessica and G Anthony, "Creating a Knowledge Sharing Community: If You Built It, Will They Come," ACM Communication, Vol. 46, No. 2, Feb 2003, pp. 23-25.

[4] C Yoshikawa, B Chun, P Eastham, A Vahdat, T Anderson and D Culler, "Using Smart Clients to build scalable services," Proc. of Usenix, Jan 1997.

[5] D Mosedale, W Foss, R McCool, "Lesson learned administering Netscape's Internet site," IEEE Internet Computing, Vol. 1, No. 2, Mar-Apr. 1997, pp. 28-35.

[6] H K Mehta, "Performance Enhancement of Scheduling Algorithms in Clusters and Grids using Improved Dynamic Load Balancing Techniques," Ph D Thesis, School of Computer Science & IT, Devi Ahilya University, Indore, MP, India, Nov 2010.

[7] M L Chiang, C H Wu, Y J Liao and Y F Chen, "New Content Aware Request Distribution Policies in Web-Cluster providing Multiple Services," In Proceedings of SAC-2009: ACM Symposium on App.lied Computing, Honolulu, Hawaii, USA , March 2009, pp. 79-83.

[8] N D Vohra, "Quantitative Techniques in Management," Tata McGraw-Hill, New Delhi, 2007.

[9] R Arpaci, A Vahdat, T Anderson and D Ptterson "Combining Parallel and Sequential Workstations on a Network of Workstations," Technical Report CSD-94-838, University of California, Berkeley, 1994.

[10] R Sharma, "Dynamic Load Balancing in Network of Workstations of Different Computing Powers," Ph D Thesis, Department of Computer Engineering, IET, Devi Ahilya University, Indore, MP, India, Jul 2013.

[11] P Kanungo, "Contributions in Dynamic Load Balancing Methodologies in Distributed Computing Environment," Ph D Thesis, Department of Computer Engineering, IET, Devi Ahilya University, Indore, MP, India, May 2007.

[12] S Hofmeyr, C Iancu and F Blagojevic, "Load balancing on speed," ACM Symposium on Principles and Practice of Parallel Programming, Bangalore, India, Jan 2010, pp. 147-158.

[13] W Scacchi, "Understanding the Requirements for Developing Open Source Software," IEEE Proceedings-Software, Vol. 149, No. 1, 2001, pp. 24-39.

[14] V Cardellini, M Colajanni and P S Yu, "Dynamic Load Balancing on Web-Server Systems," IEEE Internet Computing, Vol. 3, No 3, 1999, pp. 28-39.