ISSN No. 0976-5697

Volume 3, No. 3, May-June 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Easy-Chair Mechanism Based Algorithmic Generalized Approach for load Balancing in Database Servers

*P.Mohan Kumar, DR.J.Vaideeswaran
School of Information Technology
VIT University, Vellore. -14 T.N India.
pmohankumar@vit.ac.in
Jvaideeswaran@vit.ac.in

Abstract: Load balancing is the swinging challenging task in computing application, whenever any approach either parallel or distributed becomes popular. There were lot of analysis, approaches, algorithms and implementation mechanisms and architectural variants for the past few decades, but their exist till a thrust among the researchers. In this marathon as a participant in our paper we propose a generalized approach for load balancing issue in database server, an algorithm is designed based on easy-chair, a simple game mechanism which seems to be optimistic with respect to few existing technique when tested with a case study.

Key words: load balancing, algorithm, database server.

I. INTRODUCTION

Load balancing in its simplest form is ensuring that every processor does the same amount of work. As per Amdahl's law it was concerned not only with the proper assignment and reducing serialize ability but also limiting on potential speedup. As speed up is less than or equal to sequential work by max work on any processor .i.e. not only different processor do same amount work but they should also be working at same time We concentrate with this concern but exclusively only on scheduling the incoming request to the processors (database server) and avoiding delay, i.e. mainly on how the load is dispatched and balanced. The paper is organized as discussing some basic issues in load balancing; communication abstracts a programming model [1] which gives details of user level communication primitives of the system, the proposed model and the actual algorithm to be implemented and the case study sample. With these aspects we present our work as generalized approach for optimizing the database server by balancing the work load and reducing the time spent at synchronous events, and by reducing the extra work done to determine and manage a good assignment.

II. MOTIVATION

The real goal of load balancing is to minimize the time processes spend waiting at synchronization points, including an implicit one at end of the program. The process of balancing the workload and reducing the synchronization wait time include four aspects a) Enough concurrency in decomposition must be found b).deciding statistically or dynamically the concurrency is managed c) determining the granularity at with the concurrency to be exploited and finally serialization and synchronization cost must be reduced as specified in [1]. Before discussing these issues we see about the communication abstract a program model which gives the details of user level communication primitives of the system which deals with software that maps the communication abstractions to the actual hardware primitives. The communication architecture defines the set of communication operations available to the user software. The detail discussion about the layers is specified in [1] but we consider only the vision of communication process.

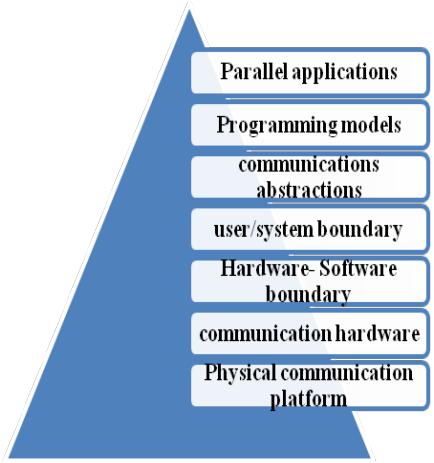

Figure 1: Abstraction layers in parallel architectures.

As the base work in load balancing is to identify enough concurrency can be perform by either data parallelism or functional parallelism .Somehow our process is not at in depth data or function level it concentrates at initial stage we discuss more on managing concurrency and reducing the wait time based on reducing inherent and reducing extra work. Firstly managing concurrency, concurrency can be managed in two ways one by static a algorithm mapping of process to task and a predetermined assignment which incur management overhead at runtime and next the dynamic one adapt to load imbalances at runtime.ie actual workload is determined during runtime. To overcome this semi static technique is deployed i.e. periodic recomputations of load balanced assignment .A distributed task queue is used to manage this technique [2].Thirdly the inherent communications is reduced by assigning task to processor based on the impact communication to computation ratio (domain decomposition) i.e. not only by the absolute amount of communication but also by the computation quantity. This can be accomplished statically or semi statically

depending upon the nature of computation. Based on these aspects the speed up limit is calculated as speedup is less than or equal to sequential work by maximum of work with synchronous wait time and communication cost and extra work. As it's basic to identify the aspects for load balancing we discussed a little and we traverse to the proposed work as follows.

III. PROPOSED WORK

The base work of our paper is to design an algorithm based on easy chair mechanism and test the case to minimizing the delay and optimize the communication time between user and server (request and response between process and processor) as load balancing in the server side. Firstly we say about easy chair mechanism then about the algorithmic design and implementation. Easy-chair is a small game played in schools were 'n' number of chairs will be kept in round form and 'n+1 or 2' pupils are allowed to run around the chair for ample of time, by counting using stop clock or playing music and suddenly ask the persons to occupy the chair by stopping the clock or music and check the status who occupied seats. Possibly n pupils occupied and remaining will be removed .Similarly n-1 chairs are kept and allowed to run around with the existing persons, this process will be repeated till two persons exist with one chair and finally one person occupy the chair he is winner and other is removed. This concept is mapped in our approach and analyzing for load balancing among processors. In the below diagram the representation is present.

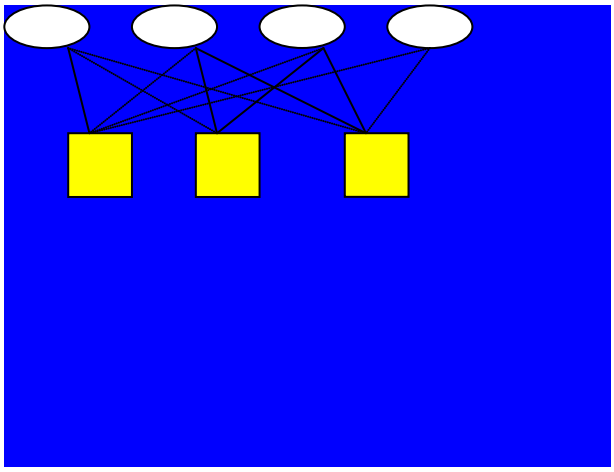

Figure: 2 Easy chair mechanism.
…… possibilities
_____ occupied status

In the above diagram square boxes are assumed to be processor and elliptical are the request process. Here the entire incoming request is allowed to participate in the competition and no specific assignment of particular process to particular processor. When ever the processor becomes free the participating process may occupy the processor, if all the processor were filled then he remaining process will put in the temp buffer queue and allowed make request continuously it won't go out so that no separate count is made The process will assigned to the processor whenever the process becomes free .This will be understood by the below algorithm and the case study better. The architecture model similar to [2] specified below may be considered.

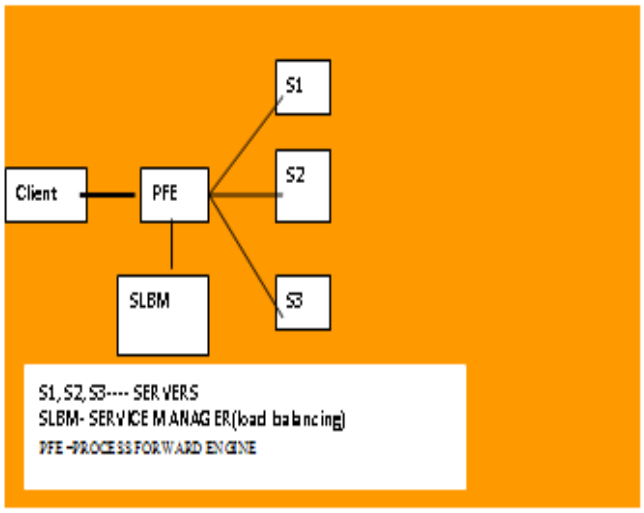

Figure: 3 proposed system

The above diagram shows the work flow of our algorithm. User request server through clients the entire request will be stored in the PFE (process forward engine) which is responsible for forwarding the process (user request) to server (processor) based on the service manager response status about the processor. These all managed by user routine specified in program model as well which controls and manages the load during assignment of process to the sever. Basically the PFE is a temporary buffer which process on circular queue principle to store the hit and miss process before and after assignment of process to processor's .SLBM the load balance performs its operation on monitor routine a user written .Which records status about the processor (server) which is free or busy and the weight assigned, the processor which forwarded through PFE got hit or Miss and queue status as well as the in coming process request details. The working procedure is as follows user requesting process all forwarded through client and stored in PFE queue. The PFE before passing the process to processor the status is verified by service manager which is responsible for load balancing. Based on this the processor in queue will be assigned to availability.

All miss process will be stored at end of queue in general .in our concept since circular queue is employed so the missed process will occupy the position which ever is empty i.e. the process which get immediate miss will placed in the position where process comes out of queue in this way the delay is minimized. A random mechanism is organized instead of round robin. This process of assignment will be repeated in a cyclic manner .The service load balancing manger in parallel monitors the processor as well as the queue whether it's getting empty with ample of time. If queue becomes empty the 1st set of incoming request is completed and next set is allowed for processing. This is explained with below algorithm. The load balancing is achieved for N+1 process with N processor is the base analysis.

IV. AIALGORITHM _EASY-CHAIR MECHANISM.

Data structure: Circular queue
Input set of process
Output processed request.

Step 1. Initialize the queue // queue size must be greater than the number of processor available.

Let queue size be N+1 or N+2.

Let number of processor be N.

Queue = Null;

Queue = number of incoming processes (user request)//in our case assume process as queries since its database server.

Step 2. Initialize service manager.

//Get the status of processors (servers).

If all the processors are busy then

Status=wait;

Else

Assign the process in queue sequentially to all the available processors.

Q [i--] =p[j]// i be the process and j be the processor

Step.3 Monitor the status.

If process get hit then

P[j]=p[j+1];

If miss then store the process in the queue to the available empty position exist.

//This performed by circular queue principle.

Step 4.Repeat the above process until all the process get hit and completed.

Q [N+1] = P [N]//i.e. the entire incoming request gets assigned.

Step 5.Repeat the above process until the initial assigned time elapses.

Step 6.End.

V. RELATED WORK

A. Scheduling the Process to Server:

The objective of scheduling is to provide service differentiation (or QoS-quality of service guarantee) for different user classes in terms of the allocation of CPU and disk I/O capacities and balance the Load among various nodes in the cluster to ensure maximum utilization and minimum execution time. Basic components involved in scheduling are each service subscriber maintains a queue to request classification determines the queue for each input request, for request scheduling determines which queue to serve next to meet the QoS requirement for each subscriber and for resource usage accounting capture detailed resource usage associated with each subscriber's service requests as shown in [3].

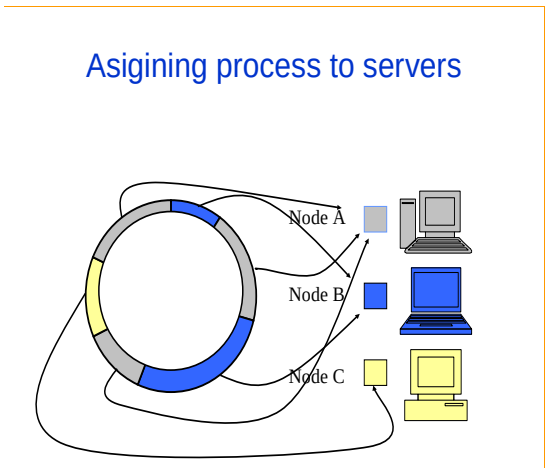

Figure 5: process assignment to server sample representation.

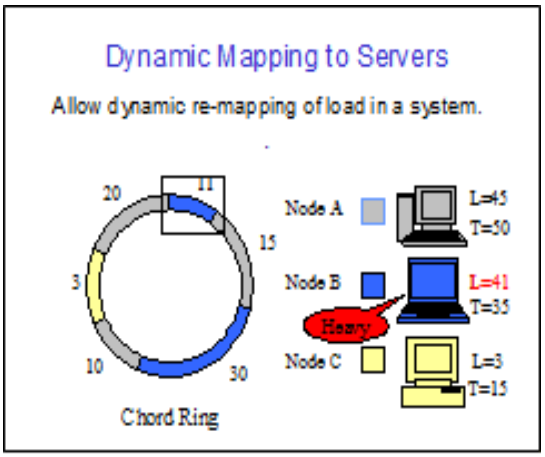

Figure 6: Dynamic mapping of process to server

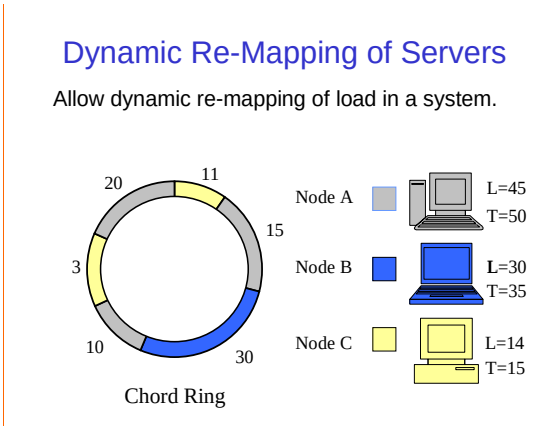

Figure 7: Dynamic remapping of process to server.

Random allocation[4],[5] distributes the incoming requests uniformly with equal probability of reaching any server for dispatching thus ,we prefer the random allocation due to the following Advantages i).flexibility in being able to move load from any node to any other node, not just to neighbor and ii).Easily supported by the underlying direct hash table because movement of virtual server appears as join and leave to the direct hash table and load balancing scheme is achieved by performing two methods one taking periodic action -Try to bring the nodes below target to bring the system in a good state and secondly Take emergency action -If the arrival of an item causes a node to go over capacity, then seek help immediately as specified detail in [4]. This is present in fig: 7 and fig: 8.Thus an analysis is made to load balance the database server wit respect to user request.

B. Case Study:

We tested this approach using the case study the FFCS i.e. a University based academic system called *Fully Flexible Credit System* [6] where students are allowed to choose their subject, faculty and time of class based on their credit system. Here during the registration period n number students participate for registration. All the users request are maintained in queue and their access will processed based on the availability. These managed by the database listener which takes responsibility for all user process. User request are allowed to access the server continuously if they get hit then it will be processed if not they will be placed in the empty position available in the queue. And the server status will be recorded with ample time intervals any how no

request will be denied it makes a in process wait time to get hit, it won't go for extra time such as Input-output like. Thus the load is managed to process all the incoming process. By the proposed algorithm this can be achieved. The existing system simply denies if the processor is busy and database listener queue is full, our approach overcomes this. Here the concept of content blind approach is used. The incoming requests uniformly distributed with equal probability of reaching any server. One overhead arises in collecting state information and analyzing them cause expensive anyhow it's managed with Input-output cost. Since our method process in dynamic and random allocation we use the server state aware concept which has the following benefits least loaded, server index based identification, fewest active connection first, fast response and weighted round robin approach i.e. variation of static Round-Robin, associates each server with a dynamically evaluated weight that is proportional to the server load .

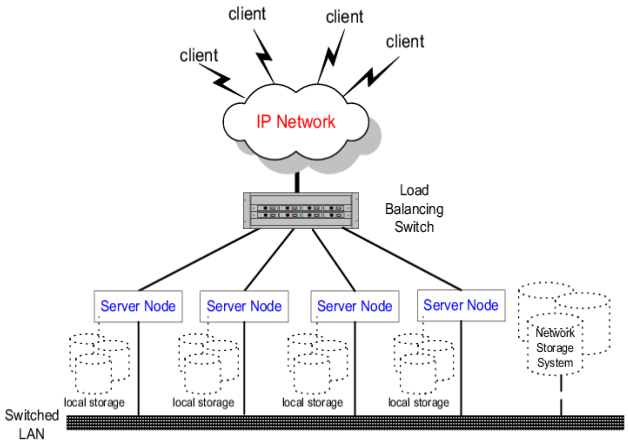

Figure 8: system model.

VI. CONCLUSION

Thus in this paper an easy chair mechanism algorithm is designed and implemented to test load balancing among database server at user request response level and a case study is analyzed based on the algorithm. It seem to be optimistic when compared to the existing methods since it allows n+1 process to n processor as well as it manage with minimum delay without denying the incoming process. Further the details about the dynamic scheduling is presented and we can conclude this paper by the extension work as content level management since we talk about database server in spite of request level it can be for query level.

VII. ACKNOWLEDGEMENT

I thank my guide for giving me valuable suggestions on behalf my research work whenever I approach him and I would like to thank our university for motivating research activities by providing lab without time limit.

VIII. REFERENCES

[1]. David E.Culler, Anoop Gupta "Parallel computer Architecture" Elsevier publications.1st Edition August 1998 unit-1.pp-128-138.

[2]. Curt kersey, Cisco Systems and Aaron Silverton Microsoft system- "A seminar series on Rserpool and server load balancing."pp-15 -19.

[3]. Sonesh Surana Berkley education, a seminar series on "Load Balancing in structured P2P systems" June 2003.pp-5-10.

[4]. Q.Zhang, A Riskw, A.W. Sun "Scheduling in web server clusters" IBM Technical report 2007. Pp-14 -17

[5]. V.Cardellini, E Casilicchio, O.H. Biro The state of Art in locally distributed system", ACM Computing surveys Vol: 34 No: 2, PP 1-49 June 2002.

[6]. Academics Section in Vellore Institute of Technology Vellore www.vit.ac.in.

Short Bio Data for the Authors

Mr. P. Mohan Kumar is working as Assistant Professor.,(senior) in School of Information Technology and Engineering(SITE), VIT University, Vellore. His area of Research includes Advanced Database Management Systems and Neural networks. He is having more than ten years teaching and academic activities. He is currently working in optimization of parallel query processing in distributed databases as a research.

Dr. J. Vaideeswaran. is working as Senior Professor in Architecture and Embedded systems division, in School of Computing Sciences and Engineering, VIT University, Vellore. His research includes High Performance Computing and Computer Architecture. Having more the 2 decades teaching experience.