International Journal on Recent and Innovation Trends in Computing and Communication
Volume: 3 Issue: 7

ISSN: 2321-8169
4742- 4746

Dynamic Load Balancing and Self-load Migration with Delay Queue in DVE

Mr. Kundan B.Pagar
Department Of Computer Engineering
G. H. Raisoni College of Engineering and Management Wagholi, Pune, India
kundanpagar@gmail.com

Prof. Sachin Patil
Department Of Computer Engineering
G. H. Raisoni College of Engineering and Management Wagholi, Pune, India
sachin.3400@gmail.com

Abstract**—**Distributed Virtual environments are gaining a lot of attention recently, due to the ever improving popularity of on the internet and social networking sites. As the variety of contingency users of a distributed virtual environment increases the critical issue is coming up, the issue describes as improving amount of work between several web servers how can be balanced to maintain real-time efficiency. The variety of load balancing methods has been suggested recently but they either try to produce high quality load balancing outcomes and become too slow or highlight on efficiency and the load balancing outcomes become less effective. In this perform, the new approach is suggested to address this issue based on the Front load balancer. The heat diffusion methods is used to develop a load balancing system after that the front load balancer will create improvements in the Dynamic load balancing of the several web servers with Delay queue. The numbers of tests are performed to evaluate the efficiency of the suggested technique. The trial outcomes show that the suggested technique works effectively in reducing server over-loading while at the same time being efficient.

Keywords- *Dynamic Load Balancing, Heat Diffusion Algorithms, Communication Latency, Distributed Virtual Environments*

*****

I. INTRODUCTION

In a distributed virtual environment lots of customers with access to the Internet are discovering a place of interest without having to go there. Or get involved in some activities with some other customers from different geometric places. Programs for distributed virtual environment techniques include multi-player free internet games, Social media, army and commercial remote training. One popular strategy is to deal with the problem of multi-server assistance is to divided the load to several web servers by dividing the virtual environment into several areas, with each area being provided by a single server. This dividing procedure is handled as fixed procedure. The advantages of this static load balancing strategy are that it is simple and the speed of the dividing procedure does not impact the performance of the distributed virtual environment But, when the customers of a distributed virtual environment program move around inside the exclusive environment, some areas may have too many customers due to some application goals while others may have too few customers. As a result, the web servers providing the populated area will become bombarded and the customers being provided may suffer from significant delay, while other server may be under used. To deal with the above restriction, a different strategy is to perform the load balancing procedure dynamically during playback However, an important need of distributed virtual environment is entertaining reaction. This means that the computational cost of the load balancing procedure should be as light as possible in order not to impact the interaction of the distributed virtual environment program. In this work we examine the potency of applying the balancing plan for load balancing and to reduce the system latency in allocated exclusive surroundings. The Heat diffusion strategy will be used to develop load balancing of distributed virtual environment .

II. RELATED WORK

Keep your text and Instead of splitting the users, a different strategy is to split the categories into fixed areas, with each area provided by one server however, when a huge number of user move into the same area. The server providing the area can still be bombarded. An enhanced strategy to this problem is to dynamically subdivide the game scene into areas. With each area provided by one server. Currently, there are two main direction of research, global load balancing and local load balancing.

In [5], a global load balancing technique is suggested. It models the lots of the allocated exclusive surroundings as a chart, with each node comprising a user and each age comprising the interaction cost between two nearby nodes. The load balancing problem becomes finding the best way to partition the chart into areas to be managed by different web servers. Although this approach may produced the best partition, it is NP complete, including handling all the nodes in the allocated exclusive surroundings. As such, it is very costly and difficult to meet the real-time need of allocated exclusive surroundings. [9] Suggests a global load balancing technique that uses a tracking server to keep check of the load produced by each of the customers and hence the quantity of lots of each server. It reassigns the customers to different web servers in order to redistribute the load s among the web servers. However, this tracking server may possibly become a center of failing. In addition, As the load balancing process is conducted slightly by this tracking server, it may actually take longer for an bombarded server to start redistributing the extreme plenty.

In [7], a load balancing technique is suggested. It separates the DVE into areas, each provided by a server. When a server is bombarded, it only connections its next door neighbor server to determine suitable server for load redistribution.

Results from the paper show that this technique is very effective. However, since the bombarded server amount not have the load position of all web servers, the load balancing process may only be regarded as a temporary one and the server may quickly become bombarded again. In [10,11], a technique is suggested to improve [7] to include additional web servers for load redistribution. An bombarded server will first look at its next door neighbor web servers to see if it may transfer all its extra load to its others who live nearby. If not, it will consider the next door neighbor web servers of the available nearby web servers and so on.

Limited factor analysis research encounters identical issues to ours in that they need to dynamically subdivide a capable for handling and analysis. However as their involved is more on the quality of the community, they generally make use of global information to enhanced the dividing [12]. Hence their techniques are generally to slowly for our objective.

III. PROPOSED SYSTEM

Problem Definition: Dynamic balancing of http requests for configurable number of servers. Make application servers capable of migrating load and avoid loss of requests by introducing Request Delay Queue.

Proposed Solution: All incoming http request will be handled by front load balancer and will be distributed to back web servers for further processing .Front controller will also consider different capabilities of every web server. Making web servers capable of themselves of migrating load to his less loaded neighbour. This will allow servers to be utilized to their high capacity. Request delay queue will be introduced when all the servers are fully loaded. It will buffer the incoming requests to process further. Scheduler will schedule the buffered requests for further process.

Request Delay Queue: Time may come, when capabilities of servers are less or total number of available servers are less to fulfil the current load on system. At such situations, when all servers will be over loaded, coming requests will be discarded because of lack of resources. Request Delay Queue(RDQ) is
Introduced for such overloaded situations. When all the servers are overloaded, coming requests will be buffered in RDQ. Capacity of this buffer can be modified by attaching more resources to it.

Queue Scheduler: This scheduler has access to the RDQ, scheduler has contract of data exchange with load accountant. As load is subsequently average, scheduler pops requests from RDQ and processes it. This avoids loss of requests on very high load situations.

A. The Load Balancing Algorithm

This load balancing criteria is based on the centralized approach. So, to come up with the problem, server chart is constructed for the distributed exclusive atmosphere program as G= (S: E), where S represents the set of web servers in the DVE program and E represents the set of sides with each edge connecting two web servers handling the two nearby categories.

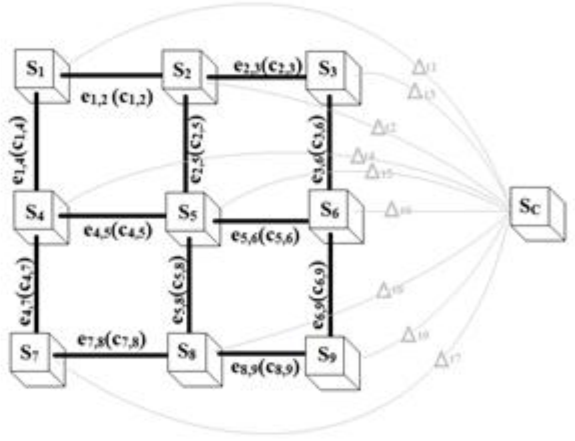


Fig. 1. A server graph with 9 nodes representing 9 partitions. Edge represents the adjacency among the partitions.

B. The Heat Diffusion Algorithm

The main idea of the heat diffusion algorithm is that given server graph, each node sends some load to its next nodes in each time. The amount to transfer to an next node is proportional to the change between the load of the present node and that of the next node. This process may take a number of iterations in order to nearly balance the load of each node. This load migration process is similar to heat diffusion process, which is governed by the heat equation:

$$\frac{\partial u}{\partial t} = \alpha(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2})$$

Where, u (x: y: t) is a temperature function that describes the variation of temperature across special location (x : y) at time t, and a is the thermal conductivity. The heat equation is used to determine the change in the temperature function over time as heat spreads throughout space.

C. The Global Heat Diffusion Algorithm

It has two key stages : global migration planning and local load transfer. In the global migration planning, the middle server, which has the global knowledge of all the servers, determines the amount of load for each node to transfer to each of its next nodes through the equivalent edge. This stage typically involves a number of iteration of prospective flow computation. In the local load transfer, each server carries out the load transfer locally. We are trying to develop the following mathematical equation. The load balancing problem can be define as how to find out the balancing flow along each edge. For convenience, we assume that all servers have the same processing performance. The balancing flow can then be formulated as follows: Where l is the is the average load over all nodes represents the summation of the balancing flows between Si and each of its adjacent nodes Sj.

Global Migration Planning:

$$\bar{l} = l_i - \sum_j \lambda_{i\to j}$$

At this level, a main server will compute the amount of load to be moved through each edge in the server chart based on the load information of all the servers. After finishing the global scheduling level, the main server will inform each server in the server chart the accumulated potential flow for each of its adjacent edges.

Local Load Transfer:
It is to transfer load between adjacent servers based on the amount indicated by the balancing flows. This process is carried out in a distributed way by each local server that manages a partition.

A. DYNAMIC LOAD BALANCING SCHEME

Now , we will see the balancing plan strategy for the powerful load balancing which we are applying in the distributed exclusive environment, the balancing plan strategy will improve the efficiency of load balancing and reduced the network latency in the distributed exclusive environment The balancing scheme divides the balancing process in three interdependent sequential phases:

1) Monitoring Phase: A group manager request information from its set of local management agents and sub-group manager and accesses third party tool. Each local management agent, as well as each subgroup manager, forwards information regarding the communication actions of simulation federates. The third party tool is accessed to collect information concerning the load of the shared resources that a group manager is responsible for managing. At the end of the process, a group manager emits migration calls, which are forwarded to the respective local management agent.

2) Reallocation Phase: After an ordered list of communicative federates is selected, a repartitioning is performed to search for the most appropriate destination resources for such a federates. This federates are reallocated by evaluating them according to the redistribution procedure. This repartitioning of federates is performed with the objective of precisely determining migration moves to destination resources that can benefits simulation performance by decreasing the communication latencies.

3) Migration Phase: In a migration procedure a manager releases a migration manager remotely, suspends a federates execution, regenerates its state, and coordinate the required data transactions. The migration manager also triggers a migration proxy to support the federates migration when the destination resource of a migration call is inaccessible by the manager, so a peer - to - peer data exchange cannot be realized. The migration proxy acts as an advanced migration aspect that has the roll of forwarding the data to a migrating federate.

B. SYSTEM ARCHITECTURE

The system architecture comprises three basic concepts, Which are as follows : 1)Central Server, 2) Local Server, 3) Local Management Agents. These three will implemented by using heat diffusion algorithm and dynamic load balancing scheme. The central server will monitor the all the phases of system. The local servers of system will the servers who directly interact with the users and they are the servers whose load balancing will be done using balancing scheme and ultimately it will result in the latency reduction. The local management agents are called as LMAs. It will incorporate all the data structures of improved system. All the phases will execute in this agents. A repartitioning is performed to search for the most suitable destination resources for such federates. These federates are reallocated by analysing them in accordance to the redistribution method described in Algorithm1

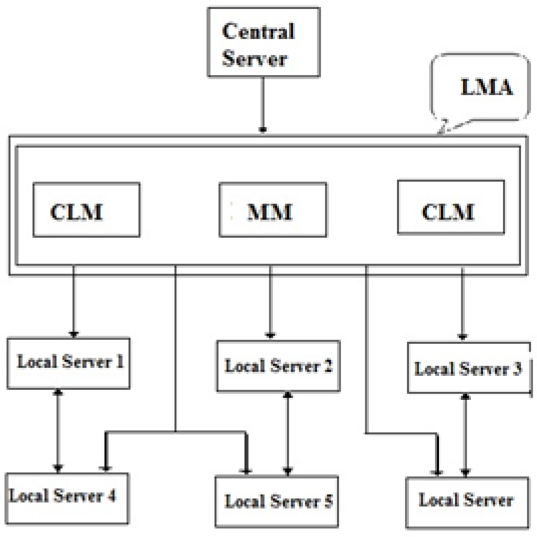


Fig. 2. System architecture for improved system

Figure 2: System architecture for improved system In the algorithm 2 the resource candidate is not able to receive a federates for migration, the next ring in the structure is selected, consequently identifying the resource with the least load. The same comparisons are consecutively applied to determine a migration move for the communicative federates.

IV. ALGORITHM AND MATHEMATICAL MODEL

Mathematical model of system is as follow:

$l = li - \sum j\ \lambda i \rightarrow j$

Where,
l = is the average load over all nodes,
$\sum j\ \lambda i \rightarrow j$ = represents the summation of the balancing flows between Si and each of its adjacent nodes Sj .
$Si = \{ S1, S2, S3, \ldots . Si\}$ are local servers.
$Sj = \{ S1, S2, S3, \ldots . Sj \}$ are local servers

Information Measures:
To consider the communication delay,
Where,
$li\ '\ t$ = The load of server Simeasured by Si itself locally at time t
li = The load status of Siavailable at the central server at time t
Δti = Time to reach load status to servers.
$li\ '\ t = li(t - \Delta ti)$

The above formula calculates the load on local server at time 't' by measuring the difference between load at time 't' and Δti

The load variation of Si between these two time moments,

$$\Delta li\ t - \Delta ti\ ,\ t + \Delta ti = li\ t + \Delta ti) - li(t - \Delta ti)$$

The formula calculates the load variation at local server by measuring the load status received at two time moments
Due to the communication delay, the balancing flow received by Si ,

$$\lambda' i \rightarrow j\ (\ t) = \lambda i \rightarrow j\ (t - \Delta ti)$$

The formula calculates the balancing flow at time 't' by measuring the balancing flow at two different time moments.

A. Algorithm 1:

Communication Redistribution Algorithm Require:
current loads, spec loads,federates loads, path distances

```
order(federates_loads)
cmean <= calc_mean(federates_loads)
cstd <= calc_STD(locservrs_loads, cmean)
comm_federates <=
find_comm(locservrs_loads,mean, std)
for all locservers IN comm_locservrs do
while !resource_found and path distances(next)
do
ring <= get_closest_ring(path_distances,RTI)
resource <= least_load_resource(ring)
If !overloaded(resource.load)then
migration_move.add(locservrs, resource)
resource_found <= TRUE
end if
end while
end for return migration_moves.
```

The procedure of looking for a resource candidate in the range rings continues while the destination resource is not found or the communication latency of a ring is not higher than the latency of the resource where the federate is running. The stop condition of the algorithm determines that any communication progress cannot be reached since the scenario of the federates cannot be improved in a given load configuration of the distributed system.

V. IMPLEMENTATION:

The hardware requirements are a simulator in java to model a multi-server distributed virtual system for evaluations, All the algorithms are also written in java. We will conduct a number of experiments to verify the effectiveness of the proposed methods. All the experiments will be conducted on a PC with an Intel Core i3v2.80GHZ and 2GB RAM. The software requirements are a simulator in java to model a multi server distributed virtual system for evaluations. All the algorithms are also written in java. So, JDK environment is required, Also we will need Eclipse java framework for the developments of java server programs.

VI. PROPOSED RESULTS:

We will conduct two set of experiments to compute the performance of our system. These two experiments are as follows 1)Ratio of overloaded servers, 2) Ratio of migrated users. We will show here the proposed results of the two experiments which are depicted in graphs. These two sets of experiments will prove as important performance measurements for our system. As it willshow the previous results and can show what our system is improving in terms of the performance.
Experiment 1:- This experiment studies the effects of the proposed balancing scheme on the performance of the load balancing algorithm based on the ratio of the overloaded severs, ROS.

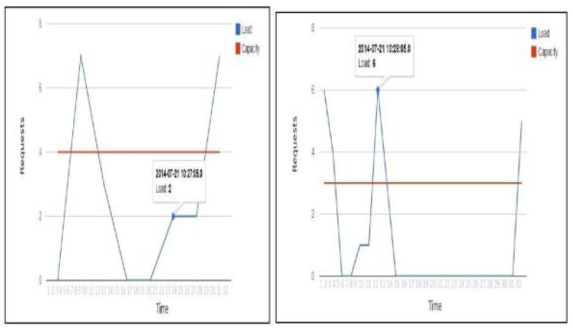
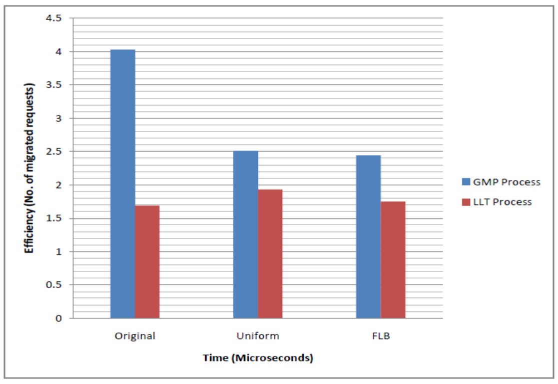


Fig: Server load analysis of local server 1&2

Experiment 2:- This experiment studies the effects of the proposed balancing schemes on the performance of the load balancing algorithm based on the ratio of migrated users, RMU.



VII. CONCLUSION:

In this Paper we have proposed a new dynamic load balancing with latency reduction approach for distributed virtual environments based on request delay queue, which has been studied in distributed virtual simulations and proved to be a very effective and efficient tool for dynamic load balancing . We have investigated heat diffusion based load balancing algorithms and improved balancing scheme of distributed virtual simulations.Ultimately, we are developing an improved balancing scheme of distributed virtual environment model based on heat diffusion algorithms and the improved balancing scheme of distributed virtual simulations. As a future work, we are currently conducting a more detailed investigation into the relevant problems.

REFERENCES:

[1] Yunhua Deng and Rynson W.H. Lau, "On Delay Adjustment for Dynamic Load Balancing in Distributed Virtual Environments" IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 18, NO. 4, APRIL 2012

[2] Robson Eduardo De Grande, Azzedine Boukerche, and Hussam Mohamed Soliman Ramadan "Decreasing communication latency through Dynamic measurements,Analysis and Partitioning For Distributed virtual simulations" IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 60, NO. 1, JANUARY 2011

[3] S.Dhakal,B.Paskaleva, M. Hayat, E. Schamiloglu, and C.Abdallah. "Dynamical discrete-time load balancing in distributed systems in the presence of time delays". In Proc. IEEE Conference on Decision and Control,volume 5, pages 5128–5134, 2003.

[4] Robson Eduardo De Grande, Azzedine Boukerche, andHus sam Mohamed Soliman Ramadan "Measuring And Analysing Migration Delay For The Computational Load Balancing of Distributed Virtual Simulations".IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT,61(12); December 2012.

[5] J. Lui and M. Chan. An efficient partitioning algorithm for distributed virtual environment systems. IEEE Trans. on Parallel and Distributed Systems, 13(3):193–211, 2002.

[6] Z. Lan, V. Taylor, and G. Bryan. Dynamic load balancing of SAMR applications on distributed systems. In Proc. ACM/IEEE Conference on Supercomputing, pages 24–24, 2001.

[7] B. Ng, A. Si, R. Lau, and F. Li. "A multi-server architecture for distributed virtual walkthrough". In Proc. ACM VRST, pages 163–170, 2002.

[8] J. Douglas Birdwell, J. Chiasson, Z. Tang, C. Abdallah, M. Hayat, and T. Wang. "Dynamic time delay models for load balancing. Part I: Deterministic models". In Proc. CNRS-NSF Workshop: Advances in Control of Time-Delay System, 2003.

[9] P. Morillo et al,"Improving The Performance Of The Distributed Virtual Environment Systems" IEEE Trans. on Parallel And Distributed systems,16(7):637-649,2005.

[10] K. Lee and D. Lee. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution. In Proc. ACM VRST, pages 160–168, 2003.

[11] D. Lee,M.lim, S. Han,K.Lee,"ATLAS: A Scalable Network Framework for Distributed Virtual Environments,"Presence. 16(2):125-156,April 2007.

[12] R. Diekmann, R. Preis, F. Schlimbach, and C. Walshaw. Shape-optimized mesh partitioning and load balancing for parallel adaptive FEM. Parallel Computing, 26:1555–1581, 2000.

[13] M. Hayat, S. Dhakal, C. Abdallah, J. Douglas Birdwell, and J. Chiasson. Dynamic time delay models for load balancing. Part II: A stochastic analysis of the effect of delay uncertainty. In Proc. CNRS-NSF Workshop:Advances in Control of Time-Delay System, 2003.

[14] R. Lau. Hybrid load balancing for online games. In Proc. ACM Multimedia, pages 1231–1234, 2010.