ISSN: 0975-5662, August, 2014 www.ijrct.org

Scheduling Allowance Adaptability in Load Balancing technique for Distributed Systems

G.Rajina[#1], P.Nagaraju[#2]

[#1]M.Tech, Computer Science Engineering, TallaPadmavathi Engineering College, Warangal, Telangana, India
[#2]Sr. Asst. Professor, Department of CSE, TallaPadmavathi Engineering College, Warangal, Telangana, India
[#1]gaderajina@gmail.com
[#2]nagaraju.pampati@gmail.com

Abstract: ***Load balancing is used to increase capacity (concurrent users) and reliability of applications. They improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions, as well as by performing application-specific tasks. The fact that processors in a distributed system are autonomous and communicate only through message-passing mechanisms, the best load balancing algorithm cannot escape overhead costs, both in terms of computation costs and communication delays, and uneven periods of load distribution. Adaptive scheduling can be expressed as finding the right balance between two conflicting issues. The first issue is the minimisation of the overhead cost by using the estimate costs established for the system environment (encouraging only the cost-effective actions). The second issue is the reduction of the duration and magnitude of these undesirable states. This article ensure reliability and availability by monitoring the tolerance difference between of two nodes and sending requests to servers and applications that can respond in a timely manner by using adaptability factors in load balancing.***

Index Terms: Load balancing control theory, request balancing, Adaptive scheduling.

1. Introduction

Content Distribution Networks (CDN) being developed to efficiently deliver content to end-users in the internet. A number of CDNs already exist as part of different infrastructures. Still, with constantly evolving networks, applications and user requirements the developments in this area are very much on-going work. Over the years CDNs have become more and more sophisticated. First generation CDNs have mainly focused on Web documents using caching and replication to provide a better. The second generation of CDNs has been mainly concerned with continuous media issues such as audio and video streaming, video on demand (VoD) schemes, scalable video support, etc. It is expected that the next generation of CDNs will focus on additional functionalities, e.g. to facilitate the creation, modification, active placement and management of content within the content infrastructure. To gain accurate insights into the performance of load balancing in distributed systems, more realistic system characteristics need to be taken into account in terms of both load balancing overheads and system model attributes.

2. Areas to navigate Load balancing

These are some of the issues related to the adaptability of an algorithm that need to be investigated:

1) Definition of a stable and balanced system

2) To which algorithm dimension(s) and/or component(s) is adaptability to be added?

3) Trade-offs in the design of an adaptive scheme:

-complexity of algorithm and range of adaptability

-responsiveness and accuracy of adaptability

-extent of variability in distributed systems and performance gain

4) How to quantify adaptability?

e.g. improvement in response time, quality of host selection.

To give more flexibility for the scheduling

algorithm to adapt to the changing environment, it must also be allowed to deviate from its main strategy by varying within a range for each scheduling parameter and policy option. The magnitude and duration of these deviations can be specified as the tolerance of the algorithm. The adaptability mechanism is used by the algorithm to tune its strategy (Le. taken corrective actions) within the algorithm tolerance according to variations in the environment and maintain an acceptable level of performance. Three types of tolerance can be identified:

a) Minimum tolerance:

This corresponds to the ideal case where no periods of unbalanced states occur. A balanced load on all the machines at all times is maintained. Unfortunately this would be achieved with excessive costs and may even result in performance degradation as in the situation where processes are transferred from nodes with few processes to an idle node. The costs of the transfers can far outweigh the gain in load balancing.

b) Heuristic values:

These values are obtained though experimentation and can achieve adequate results. For example instead of using the strictest load difference of one between two nodes in order to perform a transfer, it is more sensible to use the higher difference of three, which gives more gain to outweigh the load distribution overhead cost. However, these results are not acceptable when we are dealing with a widely changing environment.

c) Adaptive values:

Here not all the parameters or policies are fixed. The sensitive features are varied to allow, based on the system state, a dynamic adjustment of the tolerance of the algorithm to be carried out to optimize the performance.

3. Adaptability Parameters

Instead of striving for the minimum tolerance, we examine how the algorithm components can be adjusted and the scheduling strategy tuned to maintain performance and stability. The adaptability of a load balancing algorithm can be explored along two paths *parametric tuning* with three types of parameters involved: scalar, timing, and threshold, and *policy switching* with four types of policies involved: condition of the node initiating the load redistribution process, type of process transfer, load redistribution objective, and basic algorithm component options.

1) Parametric Tuning

The parameters of a load balancing algorithm that can be tuned can be classified into three types: threshold, timing, and scalar. This adjustment can occur within any of the components of the load balancing scheme: load measure, information policy, transfer policy, and negotiation policy.

Threshold

Threshold parameters include local load threshold, difference between local load and global average load value, limit on the number of successive process transfers, size of the subset of nodes that exchange information or negotiate process transfer with a given node.

Timing

These parameters determine how often the load redistribution actions will be performed, for example slowing down the scheduler activity for periodic policies or the tuning of the amount of idle time.

Scalar

The scheduling parameters of an algorithm can be assigned a weight to emphasize their static or dynamic importance in a decision function or to modify the weight of a decision based on static or dynamic local conditions.

2) Policy scheduling

To cope with a changing environment, the scheduling algorithm involves many policies. These policies can be classified further according to their nature and the

ISSN: 0975-5662, August, 2014 www.ijrct.org

options available. They are invoked dynamically for example the initiation of transfer can be performed by either the overloaded or the under loaded node.

a) Node initiating the load redistribution process

The algorithms based on this approach are called *senderinitiated* and are commonly used for dynamic load balancing. They do not require preemptive scheduling. In this case the node is searching for an overloaded node for the purpose of relieving it of part of its load. These algorithms are called *receiver-initiated* most cases assume the availability of preemptive migration of processes, because there is a small probability that at the time the idle node interrogates the overloaded node, a new external job arrives. A third alternative is to have either the sender or the receiver node initiate the load distribution process. These types of algorithms are called *symmetrically-initiated* and have more potential for adaptability.

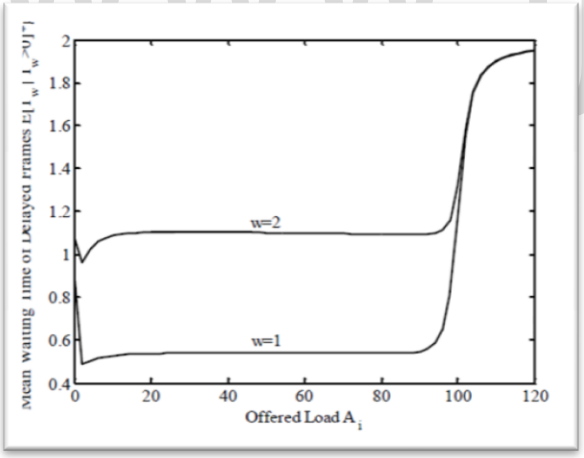


Fig1 Mean delay time for delayed frames versus offered load

b) Type of process transfer

The transfer of a process for remote execution can be done before it begins execution on the local node or even while it is running on the local node. In this case it is interrupted and sent, along with its image including the changes which occurred due to execution to another node for remote execution. In both types of transfer, the results of execution are sent back to the originating node if no shared file system is used. They deal only with newly arriving processes. Before a new process arrives no load anomalies can be corrected.

d) Load redistribution objective

Based on the system conditions and the performance objectives sought, different degrees of load distribution can be implemented. When all the nodes in the pool are idle or lightly loaded, or all heavily loaded; there is no performance gain in trying to distribute the load. If the load distribution goal is to maximize the rate at which work is performed by the system by making sure no node is idle while processes are waiting for service at other nodes then load sharing is the solution. This assumes that keeping all the nodes busy results in a better mean job response time. Load balancing extends the load sharing objective by aiming at allocating a near equal number of jobs to each node in the system. In addition to mean response time, the mean and standard deviation of the wait are to be minimized. Load balancing reduces both wait time and wait ratio. This implies a fairness of scheduling, but may degrade performance in some cases as in a system with heterogeneous node capacities. By allowing the load balancing algorithm to select the degree of distribution to aim for, adaptability to wide ranging system conditions can be achieved.

4. Representative Load Balancing Algorithms

1) Random Algorithm

When a node load level crosses the threshold T_{si} (load.i > T_{si}), it sends the newly arrived job to a randomly selected node. Only the local information is used.

2) Sender Algorithm

This algorithm is based on the Sender policy and THRHLD policy . When a node becomes overloaded it sequentially polls a set of (Lp) rand?m nodes looking for one whose load is below the *threshold*. If so an ACCEPT

message is sent back, otherwise it replies with a REJECT message.

3) Receiver Algorithm

This algorithm is based on the Receiver policy If the completion of a job brings the load of a node below the *threshold* (load.i < *Tri),* this node polls a random set of nodes up to a *probe limit* looking for an overloaded node (load.j > *Tra),* in which case a non-preemptive "migration" of a job from the ready queue of the overloaded node is done. A special case is the idle node state (load.i = 0).

4) Shortest Algorithm

This algorithm is based on the DISTED algorithm in. It allocates a new job that brings load.i above *Tsi* , to the node with the shortest queue (node.j = min *(L ι, L2 , ... , Ln»*. It maintains a load vector at each node. This vector is periodically updated using a broadcast mechanism. To reduce the number of information exchanges, the nodes broadcast their state only when the load changes.

5. Load Balancer Implementations

Most implementations of load balancing in distributed systems have been done in an ad hoc and have been added on the top of already existing operating systems. This involves a special syntax for command submission and a modification of the operating system to provide for remote execution mechanisms. Provided below is an example case for the multiple parallel hysteresis model applied on a data canter having 100 servers and 200 buffer places. Results are shown for hysteresis width *w* =1 and 2 for all hysteresis. Figure 1 shows the average waiting time of buffered frames versus offered load. The average delay is almost constant over a wide range of values, which allows increasing the offered load without affecting the average delay.

Applying the proposed load balancing algorithm on several data canters in the cloud will lead to routing some users' requests to a data canter that is not the nearest if the nearest one is highly loaded. This will increase the transfer delay because data will travel a longer distance. However, these delays are much smaller than the delays that the user could have experienced if requests were still routed to the highly-loaded data centres.

The study given in this chapter has shown that the Diffuse algorithm leads to the smallest job mean response time of the load balancing algorithms studied for a range of system attributes and workload models. Two other issues are explored in this section:

• Scalability

• Confidence Levels for the Results

Scalability

An important feature of any load balancing algorithm is that performance improvements are maintained as the number of processors in the system increases. In this thesis we have looked at three broad types of algorithms based on their information policy:

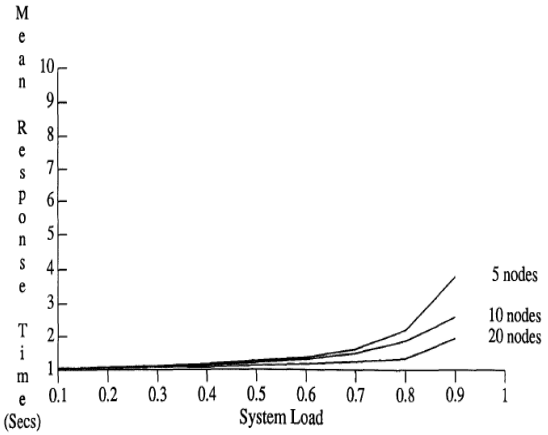
i) no system information

ii) system wide information

iii)information about subset of nodes

Confidence Levels for the Results

The need for statistical output analysis is based on the observation that the output data from a simulation exhibits random variability when random numbers generators are used to produce the values of the input variables. Consequently two



ISSN: 0975-5662, August, 2014 www.ijrct.org

Fig 2 Effect of network size on Adaptive load balancing algorithm

6. Conclusions

Load balancing can become a reality only when its performance and cost effectiveness is proven on actual distributed systems which allows users' requests to be routed to the nearest data centre while guaranteeing limited delays even if the data centre is loaded up to its near capacity limit. Also shifting of any extra load from overloaded servers to under loaded ones could be done without affecting performance or users' service level agreements in terms of delay.

Since these modules are independent because they need to interact with each other to carry out their tasks, load balancing in this context involves different objectives and requirements. It is worthwhile to investigate the applicability of the load balancing concepts considered in this work to distributed systems.

7.References

1. Wang, Y., Wen, X., Sun, Y., Zhao, Z., Yang, "The Content Delivery Network System Based on Cloud Storage," Network Computing and Information Security (NCIS), 2011 International Conference on , vol.1, no., pp.98-102, 14-15 May 2012
2. Lin, C., Leu, M., Chang, C., Yuan, S.: "The Study and Methods for Cloud Based CDN," Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on , vol., no., pp.469-475, 10-12 Oct. 2011.
3. ANSA87. ANSA, *ANSA Reference Manual Release 00.03*, ANSA Project, Cambridge(1987).
4. S. Manfredi, F. Oliviero, and S. P. Romano, "Distributed management for load balancing in content delivery networks," in *Proc. IEEE GLOBECOM Workshop*, Miami, FL, Dec. 2010, pp. 579–583.
5. T. Plagemann, V. Goebel, A. Mauthe, L.Mathy, T. Turletti, G. Urvoy-Keller, "*From content distribution to content networks - issues and challenges*", Computer Communications, Elsevier, 29 (5), 2006.
6. Network Systems Group, "Projects," Princeton University, Princeton, NJ, 2008 [Online]. Available: http://nsg.cs.princeton.edu/projects
7. A. Barbir, B. Cain, and R. Nair, "Known content network (CN) request- routing mechanisms," IETF, RFC 3568 Internet Draft, Jul. 2003 [Online]. Available: http://tools.ietf.org/html/rfc3568.
8. B. Beck, "AAMP: A Multiprocessor Approach for Operating System and Application Migration," *ACM Operating Systems Review* **24** (2) pp. 41-55 (April 2000).