Load Balancing in a Network

Rahul Godha [*], Sneh Prateek [**]

[*] Cisco Systems, Bangalore, India
[**] Akamai Technologies, Bangalore, India

Abstract- This paper introduces a new mechanism for load balancing in a network. Load balancing is a computer networking method for distributing workload across multiple computing resources such as computers, a computer cluster network links ,central processing units or disk drives. Load Balancing is usually provided by dedicated software or hardware, such as a multilayer switch or Domain Name System Sever Process. There are various algorithms to perform load balancing. In this paper we will discuss how to perform load balancing using heaps and discuss the advantage and disadvantages of using this method to perform load balancing.

Index Terms- Load Balancing; Heaps; Network .

I. INTRODUCTION

Load imbalance is an insidious factor that can reduce the performance of a parallel application significantly. For some applications, load is easy to predict and does not vary dynamically. However, for a significant class of applications, load representations by pieces of computations vary over time, and may be harder to predict. This is becoming increasing prevalent with emergence of new-sophisticated applications.

This new method of load balancing using heaps will automatically and dynamically distribute traffic across multiple servers. It will enable us to achieve greater levels of fault tolerance and improved performance seamlessly providing the required amount of load balancing capacity needed to distribute network traffic.

Rest of the paper is organized as follows: Section 2 gives an overview, background and real life applications of load balancing using heaps. Advantages and disadvantages are discussed in Section 3. Section 4 concludes the paper with references at the end.

II. BACKGROUND

Load characteristics in dynamic applications can change it is important to incur less overhead due to load balancing. Otherwise benefit of load balancing is lost in the overhead. Therefore, we evaluate quality of load balance, cost of the load balancing strategy and the total application time.

III. RELATED WORK

Load balancing has been studied extensively in the literature. For applications with regular load, static load bal- ancing can be performed where load balance is achieved by carefully mapping the data onto processors. Numerous al- gorithms have been developed for statically partitioning a computational mesh. These model the com- putation as a graph and use graph partitioning algorithms to divide the graph among processors. Graph and hyper- graph partitioning techniques have been used to map tasks on to processors to balance load while considering the local- ity. They are generally used as a pre-processing step and tend to be expensive. Our algorithm is employed where the application work has already been partitioned and used to balance the computation load imbalance that arises as the application progresses. Our algorithm also takes into con- sideration the existing mapping and moves tasks only if a processor is overloaded.

For irregular applications, work stealing is employed in task scheduling and is part of runtime systems such as Cilk Work stealing is traditionally used for task parallelism of the kind seen in combinatorial search or divide- and-conquer applications, where tasks are being generated continuously. A recent work by Dinan et al. scales work stealing to 8192 processors using the PGAS programming model and RDMA. In work that followed, a hierarchical technique described as retentive work stealing was employed to scale work-stealing to over 150K cores by exploiting the principle of persistence to iteratively refine the load balance of task-based applications. CHAOS provides an inspector-executor approach to load balancing for irregular applications. Here the data and the associated computation balance is evaluated at runtime before the start of the first iteration to rebalance. The proposed strategy is more focused towards iterative computational science applications, where computational tasks tend to be persistent.

Dynamic load balancing algorithms for iterative applications can be broadly classified as centralized, distributed and hierarchical. Centralized strategies tend to yield good load balance but exhibit poor scalability. Alternatively, several distributed algorithms have been proposed in which processors autonomously make load balancing decisions based on localized workload information. Popular nearest neighbor algorithms are dimension-exchange and the diffusion methods. Dimension-exchange method is performed in an iterative fashion and is described in terms of a hypercube architecture. A processor performs load balancing with its neighbor in each dimension of the hypercube. Diffusion based load balancing algorithms were first proposed by Cybenko and independently by Boillat This algorithm suffers from slow convergence to the balanced state. Hu and Blake proposed a non-local method to deter- mine the flow, which is minimal in the l2-norm but requires global communication. The token distribution problem was studied by Peleg and Upfal where the load is considered to be a token. Several diffusive load balancing policies, like direct neighborhood, average neighborhood, have been proposed. In a sender-initiated model is compared with receiver-initiated in an

asynchronous setting. It also compares Gradient Method ,Hierarchical Method and DEM (Dimension exchange). The diffusion based load balancers are incremental and scale well with number of processors. But, they can be invoked only to improve load balance rather than obtaining global balance. If global balance is required, multiple iterations might be required to con- verge. To overcome the disadvantages of centralized and distributed, hierarchical strategies have been proposed. It is another type of scheme, which provides good performance and scaling.

In our proposed algorithm, global information is spread using a variant of gossip protocol. Probabilistic gossip- based protocols have been used as robust and scalable methods for information dissemination. Demers et al. use a gossip-based protocol to resolve inconsistencies among the Clearinghouse database servers. Birman et al. employ gossip-based scheme for bi-modal multicast, which they show to be reliable and scalable. Apart from these, gossip- based protocols have been adapted to implement failure detection, garbage collection, aggregate computation etc.

IV. LOAD BALANCING USING HEAPS

At every sub-network level maintain a max. heap, which stores the score(threshold-load) on every server. We will also a buddy heap in sync with this primary heap for redundancy.

4.1 IMPLEMENTATION

We will store the information (threshold – load on server) for each server at sub-network level. We can organize this information using a max heap. So the next request will always be routed to server at the top of the max heap. After this the heap will change in real time as the (threshold – load on server) will change for the server to which the request went to, hence the heap will have to re-ordered accordingly every time a request has been routed.

To make this system more dynamic and efficient we need to know real time what will be the load initiated by each request. To do this we can use machine learning and build a system, which will record the load caused by each new request. In this way this system will keep on evolving.

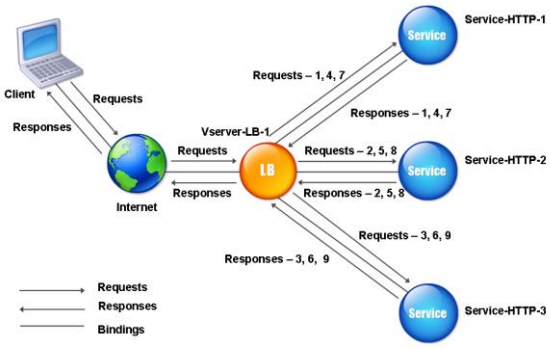

Fig1: Traditional Load Balancing

This is what a traditional load balancer does it distributes requests among the various servers. Now with our idea of load balancing the load balancer will store and maintain a max heap and redirect the request to the server whose score (threshold-load) is maximum.

The ones with scores <0 are overloaded hence no requests will be redirected to these servers.

V. ADVANTAGES

1) The algorithm since it uses heap the search time will be order of O(n).
2) The algorithm helps achieve higher levels of fault tolerance. It automatically detects unhealthy servers and only routes traffic to healthy instances.
3) It is secure. You can create an internal (non-internet facing) load balancer to route traffic using private IP addresses within your virtual network.
4) It is robust as we have a buddy heap in sync, so we can immediately failover.

VI. DISADVANTAGES

1) To make the reordering of the heap faster, we need to know load on each of the servers beforehand. Periodic polling of the servers to determine their load can help in this regard.
2) Keeping a buddy in sync can be taxing. Again periodic syncing might help.

VII. CONCLUSION

We have presented Load Balancing using heaps, a novel algorithm for distributed load balancing, It includes a max. heap which stores the threshold-load information on the nodes of the heap. The request is redirected to the server at the top of the heap.

We have listed down the advantages of using heaps against traditional load balancing algorithms. We also listed down the advantages and disadvantages of using this approach.

REFERENCES

[1] Harshita Menon and Laxmikant Kale: A Distributed Dynamic Load Balancer for Iterative Applications
[2] KJ(Ken) Salchow Jr. : Load Balancing 101: Nuts and Bolts.

AUTHORS

First Author – Rahul Godha, Cisco Systems, Bangalore, India
Second Author – Sneh Prateek, Akamai Technologies, Bangalore, India