International journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.1., Issue.3., 2013

RESEARCH ARTICLE

ISSN: 2321-7758

# GLOBAL LOAD DISTRIBUTION USING SKIP GRAPH, BATON AND CHORD

**J.K.JEEVITHA, B.KARTHIKA***

Information Technology,PSNA College of Engineering & Technology, Dindigul, India

Article Received: 25/11/2013 | Article Revised on:03/12/2013 | Article Accepted on:07/12/2013

J.K.JEEVITHA

B.KARTHIKA

**ABSTRACT**

Over the past few years, Distributed systems have rapidly grown in popularity and have become a dominant means for sharing resources. In these systems, load balancing is a key challenge because nodes are often heterogeneous. While several load-balancing schemes have been proposed in the literature, these solutions are typically ad hoc, heuristic based, and localized. In this paper, the present a general framework, HiGLOB, for global load balancing in Distributed systems. Each node in HiGLOB has two key components: 1) a histogram manager maintains a histogram that reflects a global view of the distribution of the load in the system, and 2) a load-balancing manager that redistributes the load whenever the node becomes overloaded or underloaded. The exploit the routing metadata to partition the Distributed network into non overlapping regions corresponding to the histogram buckets. The propose mechanisms to keep the cost of constructing and maintaining the histograms low. Finally, we demonstrate the effectiveness of HiGLOB by instantiating it over three existing structured Distributed systems: Skip Graph, BATON and Chord.

**Key words**— framework, load balancing, histogram, DHT, Distributed network, Skip Graph, BATON, Chord.

## INTRODUCTION

Distributed systems have emerged as an appealing solution for sharing and locating resources over the Internet. Several Distributed systems have been successfully deployed for a wide range of applications. In fact, a recent study showed that Distributed systems dominate up to 70 percent of Internet traffic. Thus, it is critical to design a Distributed system that is scalable and efficient. To build an efficient Distributed system, researchers have turned to structured architectures, which offer a bound on search performance as well as completeness of answers. However, one key challenge that has not been adequately addressed in the literature is that of load balancing. In a large-scale Distributed system, nodes often have different resource capabilities (storage, CPU, and bandwidth)[4]. Hence, it is desirable that each node has a load proportional to its resource capability.

The basic approach to load balancing is to find a pair of nodes—one that is heavily loaded and the other lightly loaded—and redistribute the load across these two nodes. However, it is far from trivial to (globally) balance the load in a Distributed system. There are two main issues in Distributed load balancing: 1) how to determine if a node is overloaded or underloaded, and 2) if so, how to find a suitable partner node with which to redistribute the load. The main problem with this method is that it can only guarantee the global load balance of the system with some probability. On the other hand suggests a use of a separate DHT such as Skip Graph to maintain the nodes' load distribution. Nevertheless, this solution still has a problem: it incurs a substantial cost for maintaining complete information about the load at every node in the system.

In this paper, we propose a new framework, called Histogram-based Global Load Balancing (HiGLOB) to facilitate global load balancing in structured Distributed systems. Each node P in HiGLOB has two key components. We exploit the routing metadata to partition the Distributed network into nonoverlapping regions corresponding to the histogram buckets. We propose mechanisms to keep the cost of constructing and maintaining the histograms low. We further show that our scheme can control and bound the amount of load imbalance across the system. Finally, we demonstrate how HiGLOB can balance the load in three existing structured Distributed systems—Skip Graph, BATON, and Chord [5][7][8]. To summarize, this paper makes the following contributions:

- It proposes a general framework that uses histograms maintain a global view of the load distribution on structured Distributed systems. These histograms enable efficient load-balancing algorithms that can effectively control the amount of load imbalance across the system to globally balance the load.
- We suggest two techniques that effectively reduce the cost of constructing and maintaining the histograms.
- We show how to apply the general framework to three well-known structured Distributed systems: Skip Graph , BATON, and Chord.

We present experimental results that characterize the effectiveness and efficiency of the proposed techniques

**RELATED WORK**

Load balancing across multiple nodes has been widely studied in the context of distributed systems. Techniques that are based on static and/or dynamic methods have been developed . In static methods, load balancing is triggered when either a new node joins the system or an existing node leaves the system. When a new node joins the system, it attempts to find a heavily loaded node and take over some of the load from that node. On the other hand, when a node leaves the system, it searches for a lightly loaded node to pass its current load to that node. In a different approach, dynamic methods operate when nodes that have already joined the system become overloaded or under loaded.

Load balancing can also be handled using a preventive mechanism that avoids load imbalance. In each peer node keeps exactly the same logN virtual nodes. As a result, with high probability, the system is load balanced. However, the cost of keeping additional virtual nodes at a peer is expensive. On the other hand, proposes that for each piece of inserted data, multiple hash functions are used to find multiple nodes, and the most lightly loaded node is selected to store the data. When a data item is deleted or searched, multiple hash functions are also used to find the node storing the data. Nevertheless, this simple approach is not efficient since it incurs an expensive cost for data insertion, data deletion, and data search.

**THE HIGLOB FRAMEWORK**

In this section, we present the HiGLOB framework. We focus on the histogram and load-balancing managers.

**A. The Histogram Manager:** The objective of the histogram manager is to maintain statistics about the load distribution across the entire Distributed network. These statistics allow a node to know its own load status (in comparison with other nodes in the system) and to identify its counterpart if global load balancing is triggered.

***B. Histogram Structure***: The histogram contains several buckets, each of which keeps statistical information about the load of a group of nodes that is connected to P through a neighbor node. The statistical information includes the current workload of CPU, storage, and bandwidth of nodes in the group. As a result, from the histogram information, the system can balance the load of nodes according to any one of a variety of node capabilities—storage, CPU, or bandwidth. Since histogram buckets have to be disjoint, different groups belonging to different buckets Fig. 1 illustrates a histogram structure. A has six neighbor nodes, and hence, its histogram contains six buckets corresponding to six nonoverlapping groups connected by six neighbor nodes. To build this histogram structure, we need to be able to partition the Distributed network into several disjoint groups. However, this is difficult since there may be multiple paths between two nodes. In other words, a node may be connected to the histogram owner node through different neighbor nodes, and hence, it can belong to different groups (violating the nonoverlapping constraint).

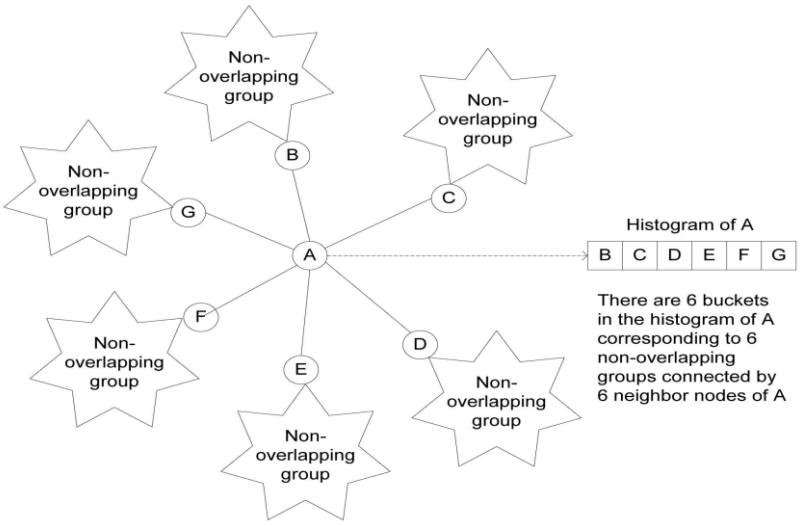


Fig. 1. Histogram structure

***C. Histogram Construction and Maintenance:*** The load distribution histogram of a node is first constructed when the node joins the system. Later, histogram values can be updated when the load distribution of the system changes. To construct a histogram, a node needs information from all neighbor nodes connected to it. To update a histogram or a bucket value, a node only needs updated information from the neighbor node connected to the corresponding group of that bucket.

***D. Improvement Techniques***: While histograms are useful, the cost of constructing and maintaining them may be expensive especially in dynamic systems. As a result, we introduce two techniques that reduce the maintenance cost.

- **Reduce the cost of constructing histogram.**
- R**educe the cost of maintaining histogram**

An example is shown in Fig. 2 in which a major change at node A may affect updating histograms at its neighbor nodes B, C, D, E, F, and G first. After that, the update process may continue at C1 and C2.Clearly, if _ is set to a low value, it is still costly for updating histograms as the updating process may be triggered often. On the other hand, if _ is set with a high value, the load imbalance between nodes maybe high because there maybe a big difference between the real load value of a onoverlapping group and its stored value in the histogram.

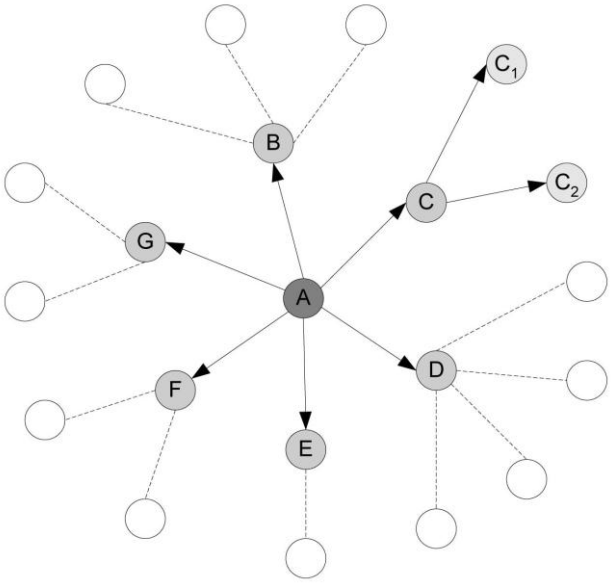


Fig. 2. Histogram update

***E. The Load-Balancing Manager***: In our framework, load balancing is done both statically when a new node joins the system or an existing node leaves the system and dynamically when an existing node in the system becomes overloaded or underloaded. In static load balancing, a new node needs to find a heavily loaded node to join as an adjacent node while an existing node wishing to leave the system needs to find a lightly loaded node to shed its workload[5][6][8]. On the other hand, dynamic load balancing is realized by either local load balancing or network load balancing.

***F. The General Framework***: The first component contains overridden methods, which extend the original methods for piggybacked histogram information. The other two components are the Histogram Manager, which is in charge of managing histogram information, and the Load Balancing Manager, which has a responsibility for load balancing among nodes in the system. These two components contain abstract methods to be implemented depending on specific systems.

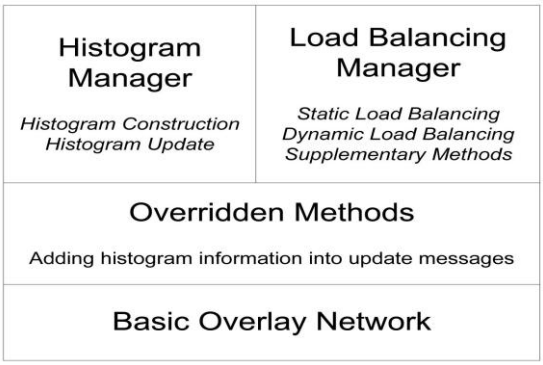


Fig. 3. The general framework

***BATON:*** Balanced Tree Overlay Network (BATON) is a structure based on a binary balanced tree in which each peer in the network maintains a node of the tree. A node has connections to other nodes by four different kinds of links: a parent link pointing to the parent node; child links pointing to child nodes; adjacent links pointing to adjacent nodes, which maintain adjacent ranges of values; and neighbor links pointing to selected neighbor nodes, which are nodes at the same level, having distances equal to a power of two from the node. In

BATON, data stored at nodes is ordered increasingly from the left to the right of the tree. An example of BATON is shown in Fig. 4.

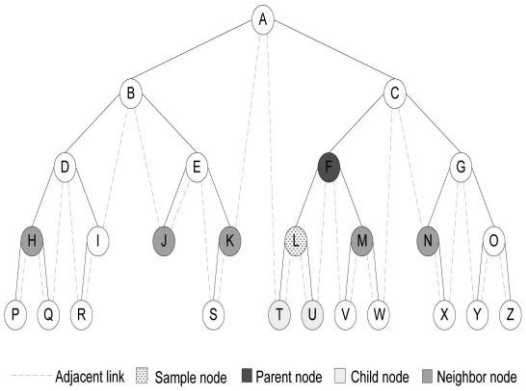


Fig. 4. BATON structure

In BATON, when a node x processes a query, if the searched key does not fall into the range of values managed by x, x forwards the query to the farthest neighbor node in the routing table, which is nearer to the searched key. If there is no such neighbor node, x forwards the query to either a child (if it exists) or an adjacent node of x in the search direction. In particular, if x is a leaf node without a full routing table on the search direction, x always forwards the query to its parent node for processing

***A. Histogram Structure***: A difficulty arises in defining nonoverlapping groups for a histogram in BATON. The search region, which is bounded by a query when it is sent from a node to its child, always, includes the adjacent node of that node on the same search direction (the adjacent node is either a descendant node of the child node or the child node itself). As a result, there is no way to define nonoverlapping groups connected by both child link and adjacent link of a node at the same side. First, if a node does not have a child, the nonoverlapping group, which contains nodes falling between the node and the first neighbor node on the direction of the missing child, is connected by the adjacent node on that direction. For example, node J in Fig. 8 has a histogram with nonoverlapping groups connected by nodes B, E, H, I, K, L, and N (B and E are adjacent nodes of J). The values are respectively 0.53, 0.56, 0.75, 0.61, 0.66, 0.61, and 0.65. Second,
if a node does not have full routing table, it always asks its parent to find a lightly or heavily loaded node as in the search algorithm, and hence, it does not need to keep exact histogram values for this node.

***B. Histogram Construction and Maintenance***: The way histogram is constructed and maintained in BATON is similar to that of Skip Graph. When a new node joins the system, all of its neighbor nodes have to calculate and send the summary of load and capacity of themselves and all nodes following their position on the same side of the new node from their histogram values. However, in BATON, the histogram values are not recalculated right away if the new node does not have full routing tables. These values are kept until the node has full routing tables. At that time, histogram values are recalculated. The histogram values of nonoverlapping groups, which contain nodes falling between the node and the first neighbor node, are calculated with additional information from histogram values of the parent node. When the load of a node is changed, it sends updated information, which is calculated in the same way as in the process of histogram construction, to all of its neighbor nodes.

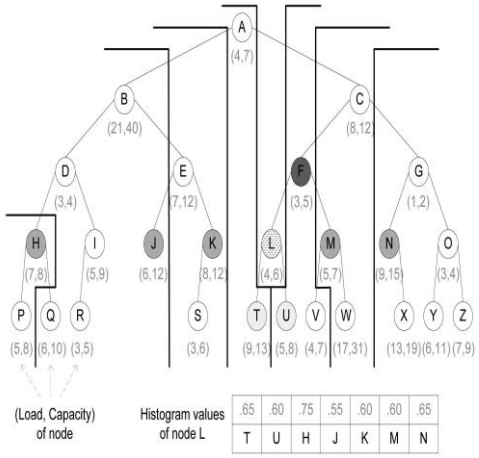


Fig. 5. Histogram structure in BATON

**CONCLUSION**

In this paper, we proposed a framework, HiGLOB, to enable global load balance for Distributed systems. Each node in HiGLOB maintains the load information of nodes in the systems using histograms. This enables the system to have a global view of the load distribution and hence facilitates global load balancing.Even though the proposal is a general framework; it is possible to deploy different kinds of Distributed systems on it. We demonstrated this by building three well-known structured Distributed systems: Skip Graph, BATON, and Chord on our proposal. Our performance evaluation shows that our HiGLOB enabled systems are superior over other methods.

**REFERENCES**

[1]. K.Aberer, "P-Grid: A Self-Organizing Access Structure for P2P Information Systems", proc. Int"l conf. Cooperative Information Systems(CoopIS),2001.

[2]. P.Ganesan, M.Bawa, and H.Garcia-Molina,"Online Balancing of Range-partitioned Data with Application to Peer-to-Peer Systems," Proc. very Large Databases conf.(VLDB '04), pp.444-455,2004.

[3]. M. Adler, E. Halperin, R.M. Karp, and V.V. Vazirani, "A Stochastic Process on the Hypercube with Application to peer-to-peer Networks,"proc. 35th ACM Symp. Theory of computing (STOC '03),pp. 575-584,2003.

[4]. S. Saroiu, P.K. Gummadi, and S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," Proc. MultimediaComputing and Networking Conf. (MMCN), 2002.

[5]. "Load Balancing in Structured P2P Systems," Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS), 2003.

[6]. D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," Proc. ACM Symp. Parallelism in Algorithms and Architectures (SPAA), 2004.

[7]. D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," Proc. Int'l Workshop Peerto Peer Systems (IPTPS), 2004.

[8]. B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, andStoica, "Load Balancing in Dynamic Structured P2P Systems,"Proc. INFOCOM, 2004.

[9]. K. Kenthapadi and G.S. Manku, "Decentralized Algorithms Using Both Local and Random Probes for P2P Load Balancing," Proc.ACM Symp. Parallelism in Algorithms and Architectures (SPAA),2005.

[10]. G. Giakkoupis and V. Hadzilacos, "A Scheme for Load Balancing in Heterogenous Distributed Hash Tables," Proc. ACM Symp.Principles of Distributed Computing Conf. (PODC), 2005.

[11]. J. Ledlie and M. Seltzer, "Distributed, Secure Load Balancing with Skew, Heterogeneity, and Churn," Proc. INFOCOM, 2005.

[12]. S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and Stoica, "Load Balancing in Dynamic Structured P2P Systems," Performance Evaluation, vol. 63, no. 6, pp. 217-240, 2006.