ISSN No. 0976-5697

Volume 3, No. 5, Sept-Oct 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Novel Architecture of Heterogeneity based Dynamic Load Balancing for DHT Based P2P System

M.Buvana*
Associate Professor Dept. of CSE,
PSNA College of Engg. and Tech., Dindigul, Tamilnadu.
Dindigul, Tamilnadu
bhuvana_beula@yahoo.co.in

K.Muthumayil
Associate Professor, Dept. of IT.
PSNA College of Engg. and Tech., Dindigul, Tamilnadu.
Dindigul, Tamilnadu
muthumayil@yahoo.com

Abstract: Distributed Dynamic load balancing (DDLB) is an important system function destined to distribute workload among available processors to improve throughput and / or execution times of parallel computer . Instead of balancing the load in cluster by process migration or by moving an entire process to a less loaded computer, we make an attempt to balance load by splitting processes into separate jobs and then balance them to nodes. Many solutions have been proposed to tackle the load balancing issue in DHT-based P2P systems. However, all these solutions either ignore the heterogeneity nature of the system, or reassign loads among nodes without considering heterogeneity relationships, or both. In this paper, we present an efficient, Heterogeneity-aware load balancing scheme by using the concept of virtual servers. Proximity information is used to guide virtual server reassignments such that virtual servers are reassigned and transferred between physically close heavily loaded nodes and lightly loaded nodes, thereby minimizing the load movement cost and allowing load balancing to perform efficiently

Keywords: Dynamic load balancing, Heterogeneity-aware, load balancing, peer-to-peer, virtual server

I. INTRODUCTION

Load balancing is an efficient strategy to improve throughput or speed up execution of the set of jobs while maintaining high processor utilization. Basically Load balancing is the allocation of the workload among a set of co-operating nodes. The demand for high performance computing continues to increase everyday. Load balancing [1],[2] strategies fall broadly into either one of two classes static or dynamic. A multi-computer system with static load balancing[3],[4],[5] distributes tasks across nodes before execution using a priori known task information and the load distribution remains unchanged at run time. A multi-computer system with Dynamic Load balancing (DLB)[6],[7],[8] uses no priori task information and satisfies changing requirements by making task distribution decisions during run-time. Two classes of solutions have been proposed to tackle the load balancing issue in DHT-based P2P systems. Solutions in the first class use the concept of virtual servers [9],[10]. Each physical node instantiates one or more virtual servers with random IDs those act as peers in the DHT. In the case of a homogeneous system, maintaining $\Theta(\log n)$ virtual servers per physical node reduces the load imbalance to a constant factor. To handle heterogeneity, each node picks a number of virtual servers proportional to its capacity. Unfortunately, virtual servers incur a significant cost: a node with k virtual servers must maintain k sets of overlay links. Typically $k = \Theta(\log n)$, which leads to an asymptotic increase in overhead.

The second class of solutions uses just a single ID per node [11],[12]. However, all such solutions must reassign IDs to maintain the load balance as nodes arrive and depart the system [13]. This can result in a high overhead because it involves transferring objects and updating overlay links.

However, existing load balancing approaches have some limitations in our opinion. They either ignore the heterogeneity of node capabilities, or transfer loads between nodes without considering heterogeneity relationships, or both. In this paper, we present heterogeneity - aware load balancing scheme by using the concept of virtual servers previously proposed in [14]. The goals of our scheme are not only to ensure fair load distribution over nodes proportional to their capacities[15], but also to minimize the load-balancing cost (e.g., bandwidth consumption due to load movement) by transferring virtual servers (or loads) between heavily loaded nodes and lightly loaded nodes in a heterogeneity-aware fashion[16],[17]. There are two main advantages of Heterogeneity-aware load balancing scheme. First, from the system perspective, a load balancing scheme bearing network heterogeneity in mind can reduce the bandwidth consumption (e.g., bisection backbone bandwidth) [18],[19],[20] dedicated to load movement. Second, it can avoid transferring loads across high-latency wide area links, thereby enabling fast convergence on load balance and quick response to load imbalance.

We operate under the uniform load assumption that the load of each node is proportional to the size of the ID space it owns. This is reasonable when all objects generate similar load (e.g., have the same size), the object IDs are randomly chosen (e.g., are computed as a hash of the object's content), and the number of objects is large compared to the number of nodes (e.g., $(n \log n)$). Alternately, we can unconditionally balance the expected load over uniform-random choices of object IDs.

Our main contributions are the following.

a. Relying on a self-organized, fully distributed k-ary tree structure constructed on top of a DHT, load balance is achieved by aligning those two skews in

© 2010, IJARCS All Rights Reserved

load distribution and node capacity inherent in P2P systems—that has higher capacity nodes carry more loads.

b. Heterogeneity information is used to guide virtual server reassignments such that virtual servers are reassigned and transferred between physically close heavily loaded nodes and lightly loaded nodes, thereby minimizing the load movement cost and allowing load balancing to perform efficiently.

II. PROPOSED WORK

A. Virtual Servers:

The concept of virtual servers was first proposed to improve load balance. Like a physical peer node, a virtual server is responsible for a contiguous portion of the DHT's identifier space. A physical peer node can host multiple virtual servers and, therefore, can own multiple noncontiguous portions of the DHT's identifier space. Each virtual server participates in the DHT as a single entity. For example, each virtual server has its own routing table and stores data items whose IDs fall into its responsible region of the DHT's identifier space.

From the perspective of load balancing, a virtual server represents certain amount of load (e.g., the load generated by serving the requests of the data items those IDs fall into its responsible region). When a node becomes overloaded, it may move part of its loads to some lightly loaded nodes to become light in which the basic unit of load movement is virtual server. Hence, load balance can be achieved by moving virtual servers from heavy nodes to light nodes. Note that the movement of a virtual server can be simulated as a leave operation followed by a join operation, both of which are supported by all DHTs. Therefore, using the concept of virtual servers could make our load balancing scheme simple and easily applied to all DHTs.

B. System Overview:

The load balancing scheme we present in this paper is not restricted to a particular type of resource (e.g., storage, bandwidth, or CPU). However, we make two assumptions in our work. First, we assume that there is only one bottleneck resource in the system, leaving multi resource balancing to our future work. Second, we assume that the load on a virtual server is stable over the timescale it takes for the load balancing algorithm to perform.

Basically, our load balancing scheme consists of four phases:

a. ***Load balancing information (LBI) aggregation*****:** Aggregates load and capacity information in the whole system.

b. ***Node classification:*** Classify nodes into overloaded (heavy) nodes, under loaded (light) nodes, or neutral nodes according to their loads and capacities.

c. ***Virtual server assignment (VSA):*** Determine virtual server assignment from heavy nodes to light nodes in order to have heavy nodes become light. The VSA process is a critical phase because it is in this phase that the heterogeneity information is used to make our load balancing scheme heterogeneity -aware.

d. ***Virtual server transferring (VST):*** Transfer assigned virtual servers from heavy nodes to light nodes. We allow VSA and VST to partly overlap for fast load balancing.

Each node may depend solely on its own load and capacity to determine whether it is overloaded or under loaded, without requiring the system-wide load balancing information. Consider a node i with the load Li and the capacity Ci. Node i's utilization Ui is the fraction of its capacity that is used: Ui = Li / Ci. If Ui > 1, node i is identified as a heavy node. Otherwise, it is a light node or neutral node (Ui = 1).

C. *Mechanism for load transfer between different nodes*:

For load transfer among different nodes, each node maintains its own list of participating nodes to which it wants to communicate for load sharing. Each node maintains its own job queue along with some predefined threshold values to initiate load transfer. Let t be the time when tasks were last executed and a(tj) be the arrival time of task tj and e(tj) be time when it starts executing. Then the jobs in the queue are those being executed and ready to be executed are given by { tj / a(tj) ≤ t t, e(tj) ≤ t } and { tj / a(tj) ≥ t , e(tj) ≥ t }.

D. Load balancing information (LBI) aggregation:

Load Balancing Information (LBI) Aggregation Based on the k-ary tree structure. First we create the k-ary tree and calculate the total machine load and Max load using the following procedure that is shown in figure1 and then aggregate load and capacity information in the whole system. Each KT node periodically, at an interval T, requests LBI from its children, while each KT leaf node simply asks its hosting virtual server to report LBI. Note that a DHT node hosts multiple virtual servers. In order to avoid reporting redundant LBI of a DHT node, a DHT node (say i) randomly chooses one of its virtual servers to report LBI, in the form of < TLi, Ci, Li, max >(where TLi, Ci, Li,max stand for the total load of virtual servers, the capacity, and the maximum load of virtual servers on the node i, respectively).

```
Procedure GetMachineload ()
                        int TL =0
   For t = 0 To nCounters – 1
          tValue = GetPerformanceCounterValue(
 CounterObjects(t), CounterNames(t), CounterInstances(t))
          TL += tValue * CounterWeights(t)
   Next
                    Return (tMachineLoad)
   End
Procedure GetMaxLoad()
        t, tMaxLoad =0;
    For t = 0 To nCounters - 1
      Int tValue  = CounterMax(t)
       tMaxLoad += tValue * CounterWeights(t)
    Next
```

© 2010, IJARCS All Rights Reserved

Return (tMaxLoad)
End

E. Node classification:

After the LBI aggregation, the KT root node disseminates <TL,C,Lmin> along the k-ary tree in a top-down fashion to each KT leaf node, which in turn distributes the<TL,C,Lmin> to its own hosting virtual server. Note that one of the goals of our load balancing scheme is to ensure fair load distribution over DHT nodes by assigning the load to a DHT node in proportion to its capacity. Let Ti denote the target load of a DHT node i proportional to its capacity. We have Ti = (TL/ C + ε)Ci (ε is a parameter for a trade off between the amount of load moved and the quality of balance achieved. Ideally, ε is 0). Therefore, a DHT node i can be defined as:

a. A heavy node if TLi > Ti.
b. A light node if (Ti - TLi) >= Lmin.
c. A neutral node if 0 <= (Ti - TLi) < Lmin.

F. LC-Virtual Server Selection (LC-VSS) and Transferring:

To assign IDs to virtual servers, called Low Cost Virtual Server Selection (LC-VSS). The Virtual Server Assignment process proceeds along the k-ary tree in a bottom-up sweep, it recursively assigns virtual servers among DHT nodes scattered in an increasingly larger contiguous portion of the DHT's identifier space3 until the whole DHT's identifier space (for which the k-ary tree root node is responsible). In other words, the VSA process is identifier space-based in that the virtual server assignments are performed earlier among those DHT nodes which are closer to each other in the DHT's identifier space. Similar to the LBI aggregation process, the VSA process is also resilient to system failures due to the robustness of the k-ary tree it depends on. After the k-ary tree recovers from DHT node's failures, the VSA process can continue along the tree in a bottom- up fashion. It is worth pointing out that the VSA process discussed above is heterogeneity-ignorant because the logical closeness in the DHT's identifier space does not necessarily reflect physical closeness of DHT nodes. We name it heterogeneity -ignorant VSA.

G. ID space balance for heterogeneous DHTs.:

In a DHT-related work, Reference [15] developed two schemes which divide an ID space fairly among a set of nodes of heterogeneous capacities, providing efficient ID lookup and node join and leave algorithms. However, they assume a centralized system with no overlay network. Their SHARE strategy is very similar to our VSS: in both, each node selects an interval of the ID space of size Θ (log n), and ownership of a segment is "shared" among the nodes whose intervals overlap that segment. However, they employ this technique to handle nodes of very low capacity. In contrast, we cluster a node's IDs in order to share overlay links. Moreover, the way in which the ID space sharing is performed in Reference [15] is more complicated than in our scheme; notably, nodes need Θ (log2 n) IDs, rather than Θ (log n).

H. Load balance by object reassignment.:

The above strategies balance load by changing the assignment of IDs to nodes. Another approach is redirection: store a pointer from an object's ID to the arbitrary node currently storing it. This can balance the load of storing and serving data, but not load due to routing or maintenance of the overlay — and if objects are small, routing dominates the bandwidth and latency of storing and finding an object. Reference [3] demonstrated heterogeneous capacities and obtains a constant-factor load balance. Each node periodically contacts another, and they exchange objects if one node's load is significantly more than the other's. But their bound on movement cost depends on the ratio of the maximum and minimum node capacities.

III. INTERACTION BETWEEN THE FRAMEWORK OF LOAD BALANCING

Figure 1 shows how the load balancing framework components interact with each other at run-time.

a. Assign load to the Distributed P2P system however, the client transparently invokes the request on the load manager itself.
b. The load balancer dispatches the system enumerated information such as Bandwidth, CPU usage, load as a request to its distributed P2P system.
c. The distributed P2P system queries the load analyzer and it analysis the Nodal information.
d. Node is classified by the Load Balancer like heavy loaded, light loaded, neutral based upon the capacity and total load.
e. Then assign the load to the virtual server then allocate the heavy loaded system resources are shared with the light loaded system
f. Calculate the CPU usage and it sends to the another system through the interface.

IV. HETEROGENEITY-AWARE LOAD BALANCING

The basic idea behind the heterogeneity aware load balancing is to make virtual server assignments (i.e., the VSA process) heterogeneity-aware by using heterogeneity information

A. Generating Heterogeneity Information:

Landmark clustering has been widely used to generate heterogeneity information. It is based on an intuition that nodes physically close to each other are likely to have similar distances to a few selected nodes. In a DHT overlay network, the landmark nodes can be chosen from either the overlay itself or from the Internet. For a DHT node D, it measures the distances to a set of m landmark nodes (e.g., m =15) and obtains a landmark vector < d1; d2; . . . ; dm>. Node D is then mapped into a point in a m-dimensional Cartesian space by having the landmark vector as its coordinates.

© 2010, IJARCS All Rights Reserved

B. Heterogeneity-Aware VSA Using heterogeneity Information:

After generating heterogeneity information, a big challenge we now face is how to effectively use it to guide virtual server assignments such that they are assigned between physically close heavy nodes and light nodes.

Therefore, we divide the m-dimensional landmark space into 2mn grids of equal size (where n controls the number of grids used to divide the landmark space) and fill a Hilbert curve within the landmark space to number each grid. We then number each DHT heavy/light node with the grid number of the grid in which its landmark vector falls. We call this grid number the Hilbert number, which will serve as a DHT key. Due to the heterogeneity preserving property of the Hilbert curve, closeness in the Hilbert number reflects physical heterogeneity.

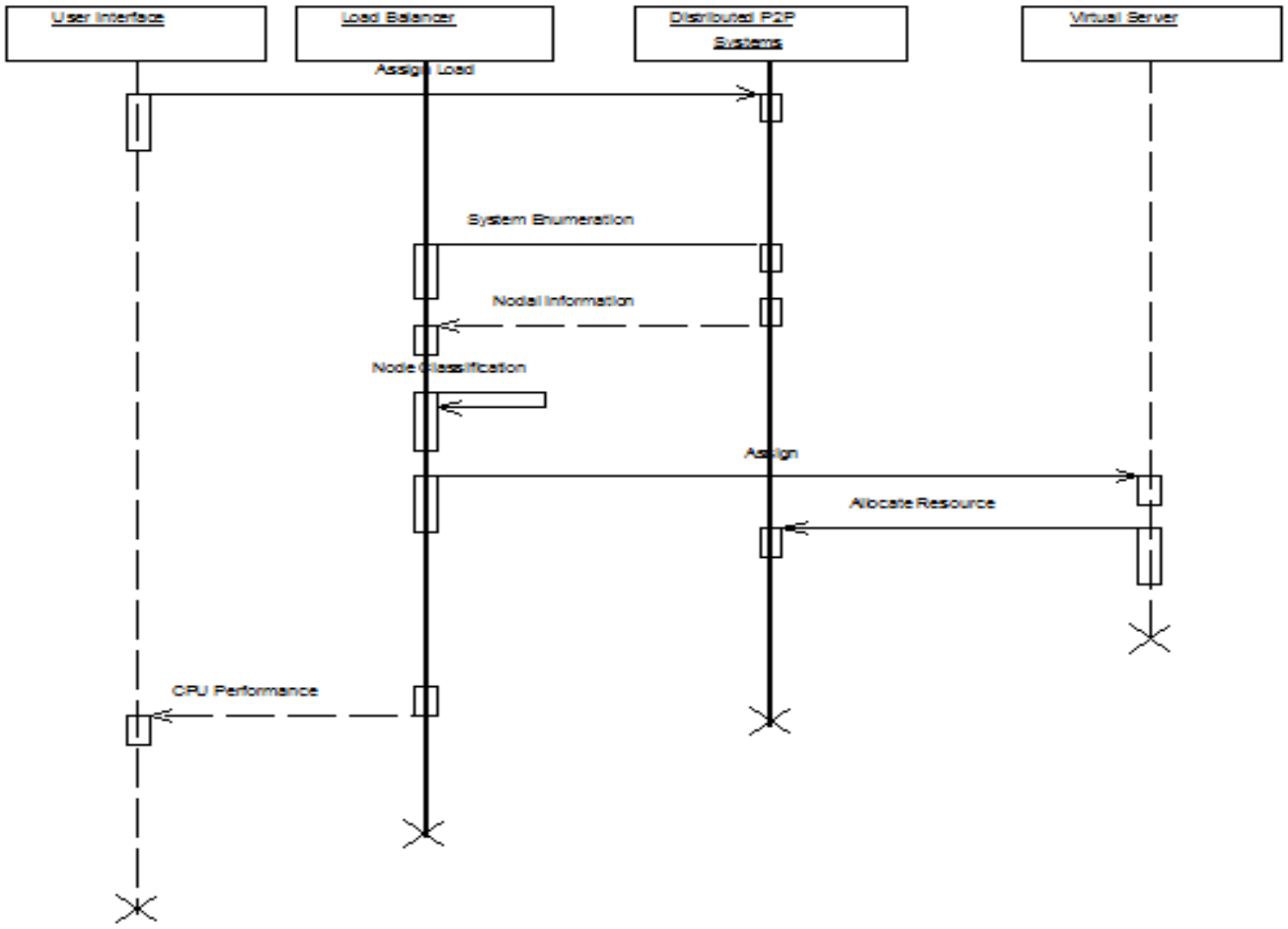

Figure 1 illustrates how the load balancing components interact with each other at run-time.

V. CONCLUSION

In this paper, we present an efficient, heterogeneity-aware Dynamic load balancing scheme to tackle the issue of load balancing in DHT-based P2P systems. This framework is a flexible foundation to implement different load balancing schemes for distributed applications. The first goal of our load balancing scheme is to align those two skews in load distribution and node capacity inherent in P2P systems to ensure fair load distribution among nodes—that is, have nodes carry loads proportional to their capacities. The second goal is to use the heterogeneity information to guide load reassignment and transferring, thereby minimizing the cost of load balancing and making load balancing fast and efficient.

VI. REFERENCES

[1] Karger.D and M. Ruhl. “New Algorithms for Load Balancing in Peerto- Peer Systems”. Technical Report MIT-LCS-TR-911, MIT LCS, July 2003.

[2] Karger.D.R and M. Ruhl, “Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems,” Proc. Third Int’l Workshop Peer-to-Peer Systems (IPTPS), Feb. 2004

[3] Karger.D and M. Ruhl. “Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems”. In Proc. SPAA, 2004.

[4] Zhu .Y and Y. Hu, “Towards Efficient Load Balancing in Structured P2P Systems,” Proc. 18th Int’l Parallel and Distributed Processing Symp. (IPDPS), Apr. 2004.

[5] Rao.A, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, “Load Balancing in Structured P2P Systems,” Proc. Second Int’lWorkshop Peer-to-Peer Systems (IPTPS), pp. 68-79, Feb. 2003.

© 2010, IJARCS All Rights Reserved

[6] Godfrey.B, K. Lakshminarayanan, S. Surana, R. Karp, and I.Stoica."Load balancing in dynamic structured P2P systems". In Proc. IEEE INFOCOM, Hong Kong, 2004.

[7] Surana.S , B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica. "Load balancing in dynamic structured P2P systems. In Performance Evaluation".Special Issue on Performance Modeling and Evaluation of Peer-to-Peer Computing Systems, 2005.

[8] H. Shen and C. Xu. "Locality-aware randomized load balancing algorithms for structured p2p networks". In Proc. of ICPP, pages 529–536, 2005.

[9] Karger.D, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web". In ACM Symposium on Theory of Computing, pages 654–663, May 1997.

[10] Haiying Shen and Cheng-Zhong Xu "Hash-based Proximity Clustering for Load Balancing in Heterogeneous DHT Networks".PODC 2006.

[11] Manku.G . "Balanced binary trees for ID management and load balance in distributed hash tables. In Proc. PODC, 2004.

[12] Naor.D and U. Wieder. "Novel architectures for P2P applications: thecontinuous-discrete approach". In Proc. SPAA, June 2003.

[13] Stoica.I , R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. ACM SIGCOMM, pp. 149-160, Aug. 2001.

[14] Dabek. F, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica,"Wide-Area Cooperative Storage with CFS," Proc. 18th ACM Symp. Operating Systems Principles (SOSP), pp. 202-215, Oct. 2001.

[15] Brinkmann.A, K. Salzwedel, and C. Scheideler. "Compact and adaptive placement strategies for non-uniform capacities". In Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA), Winnipeg, Canada, 2002.

[16] Zhao.B.Y, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-Tolerance Wide-Area Location and Routing,"Technical Report UCB/CSD-01-1141, Computer Science Division, Univ. of California, Berkeley, Apr. 2001.

[17] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. "Tapestry: An Infrastructure for Fault-tolerant wide-area location and routing".IEEE Journal on Selected Areas in Communications,12(1):41–53, 2004.

[18] Dipanjan Chakraborty, Anupam Joshi, and Yelena Yesha. "Integrating service discovery with routing and session management for ad-hoc networks". Journal of Ad hoc Networks,UMBC ebiquity publications,Vol2 No.4,2006.

[19] Kaouther Abrougui, Azzedine Boukerche and Hussam Ramadan. "Efficient load balancing and QoS-based location aware service discovery protocol for vehicular ad hoc networks". EURASIP Journal on Wireless Communications and Networking 2012,

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "A scalable content-addressable network". In Proc. of ACM SIGCOMM, pages 329–350, 2001.

© 2010, IJARCS All Rights Reserved