ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

A Symmetric Load Balancing Algorithm with Performance Guarantees for Distributed Hash Tables

Dr K S VijayaSimha, M.Tech,PhD
Department Of Computer Science & Engineering
Krishna Murthy Institute of Technology & Engineering,
Ghatkesar, Hyderabad

Yarramsetty Naga Savitri
Department Of Computer Science & Engineering
Krishna Murthy Institute of Technology & Engineering,
Ghatkesar, Hyderabad

Abstract*—Peers participating in a distributed hash table (DHT) may host numbers of virtual servers and are enabled to balance their loads in the reallocation of virtual servers. Most decentralized load balance algorithms that are designed for DHTs based on virtual servers do require the participating peers to be asymmetric, where some serve as the rendezvous nodes to pair virtual servers and participating peers, thereby introducing another load imbalance problem. The state-of-art studies introduce significant algorithmic overheads and guarantee no rigorous performance metrics. Here, a novel symmetric load balancing algorithm for DHTs is presented by having the participating peers approximate the system state with histograms and cooperatively implement a global index. Each peer independently does reallocate in our proposal its locally hosted virtual servers by publishing and inquiring the global index based on their histograms. Unlike competitive algorithms, our proposal exhibits analytical performance guarantees in terms of the load balance factor and the algorithmic convergence rate, and introduces no load imbalance problem due to the algorithmic workload. Through computer simulations, we show that our proposal clearly outperforms existing distributed algorithms in terms of load balance factor with a comparable movement cost.

Key Terms*—Distributed hash tables, load balance, virtual server.

1 INTRODUCTION

Dibuted hash tables (DHTs) are key building blocks in the implementation and design of successful distributed applications. Examples of DHTs are Chord [1] and Pastry [2]. Applications/infrastructures built based on DHTs include storage clouds [3], file-sharing network [4], and distributed file systems [5], among many others. In a typical DHT, the participating peers (or nodes) cooperatively manage a global hash table. Essentially, DHTs provide the GET(x) operation to retrieve a published object whose key is x as well as the PUT(y; v) operation to store the value v of an object with hash key y.

As peers that are participating in a DHT are often heterogeneous, the work in [1] introduces the notion of virtual servers to cope with peer heterogeneity. Let N be the set of participating peers in the DHT and V be the set of virtual servers hosted by the peers in N. (Typically, $\|N\| < \|V\|$; $\|\cdot\|$ denotes the cardinality of a set.) let Su be the hash subspace managed by virtual server $u \in V$. Then, $\mathcal{S}_u \cap \mathcal{S}_v = \emptyset$ for any $u \neq v \in V$ and $\bigcup_{k \in V} \mathcal{S}_k = \mathcal{S}$. Thus, peer heterogeneity can be exploited because the participating peers can host different numbers of virtual servers [1].

In particular, load balancing algorithms designed for DHTs based on virtual servers need to take the following into consideration:

- Load balance and movement cost. By load balance, we mean that each peer manages the load proportional to its capacity. Previous studies (e.g., [6]) suggest migrating virtual servers among the participating peers in order to balance peer load. However, this is at the expense of introducing movement cost due to the migration of virtual

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

servers. How to balance peer load while reducing movement cost as much as possible thus is a critical issue.

- System dynamics. Load balancing algorithms need to bear the system dynamics in mind because nodes may dynamically join and leave DHTs. In addition, the load of a virtual server may change from time to time, aggravating the load imbalance problem in the DHTs.

- Algorithmic robustness and workload. Load balancing algorithms need to be robust without introducing the performance bottleneck and the single point of failure. In addition, as load balancing algorithms incur algorithmic workloads, such workloads shall not induce another load imbalance problem. On the other hand, a well-designed load balancing algorithm will not generate considerable overheads.
- Performance guarantee. Load balancing algorithms shall work well with performance guarantee, given any system instance. Specifically, DHT networks may operate in dynamic and large-scale environments, thus presenting a large number of problem instances for performance investigation.

2 RELATED WORK

Assume an object set O and a peer set N. Conventional DHTs assume that the loads of objects in O are identical. In this way, the load of a peer can be estimated as the number of objects hosted by the peer. If we use a uniform hash function, then the size of the key subspace that is managed by a peer is proportional to the number of objects hosted by the peer. In contrast to evenly partitioning the number of objects and thus the key space to each participating peer, Chord suggests the notion of virtual servers to balance the loads of the peers [1] further. Different virtual servers in a system manage disjoint key subspaces, with a virtual server serving as an elementary entity for balancing loads among peers.

Note that if the key subspace S is managed by a vertual server v, then the objects, which may have "unequal" loads and whose keys are within S, contribute their loads to v. As a result, the load of a virtual server may not be proportional to the size of the key space allocated to the virtual server. Whereas the technique based on virtual servers is orthogonal to that of uniformly partitioning the key space, the idea of virtual servers enables a DHT to reallocate the virtual servers by exploiting the heterogeneities of the objects and peers, such that the resultant load of a peer is proportional to its capacity.

Shen and Xu organize a DHT as a two-tier network, where the higher level of the network consists of local clusters. A node in a local cluster is selected as a rendezvous. Objects with excess loads are moved to the local cluster through the reallocation performed by the rendezvous. For objects that cannot be moved to the local cluster, they are transferred to foreign clusters; this is accomplished by the reallocation performed by the corresponding rendezvous in the remote and local clusters. Although Shen and Xu's solution aims to migrate objects for load balancing, their solution can be adopted by DHTs with virtual servers by having each virtual server serve as the elementary entity for load migration. Here, our proposal does not assume any specific geometry of DHTs and is thus applicable to any DHT.

In the recent proposal by Hsiao et al., each peer i performs its load balancing algorithm based on the histograms it doews estimate for representing the system state.here , in this paper, we present novel histograms for nodes to estimate and represent the system state, and 2) that the participating peers in our proposal maintain a global index in a distributed manner. These techniques enable the light peers to query and allocate virtual servers of interest precisely, intending to introduce no contention of reallocation of virtual servers and thus redundant traffic. In addition, with the histograms and the global index, our proposal provides rigorous performance guarantees.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

3 OUR LOAD BALANCE ALGORITHM

3.1 Notation and Problem Definition

Let N be the set of participating peers in a DHT and V be the set of virtual servers deployed over N. Each peer $i \in N$ has a maximum capacity of $C_i^{\max}$ and hosts a set of virtual servers $V_i \subset V$. Here, $V_i \cap V_j = \emptyset$, for any $i \neq j \in N$. Each virtual server $v \in V_i$ has a load denoted by Lv (where Lv $\geq$ 0). Our objective in this study is to develop a load balancing algorithm A to reallocate and balance the loads among the participating peers, such that any peer **i** manages the total load of virtual servers proportional to its $C_i^{\max}$. That is, A computes a subset $V_i^A \subset V$ for each peer **i**, such that the following equation is minimized:

$$\left| \sum_{v \in V_i^A} L_v - \frac{\sum_{v \in V} L_v}{\sum_{k \in N} C_k^{\max}} \times C_i^{\max} \right|, \qquad (1)$$

Definition 1. The load per unit capacity, μ, which is a peer that hosts in a load-balanced DHT, is defined as

$$\mu \triangleq \frac{\sum_{v \in V} L_v}{\sum_{k \in N} C_k^{\max}}. \qquad (2)$$

Without loss of generality, we assume that $\sum_{v \in V} L_v < \sum_{k \in N} C_k^{\max}$. Thus, $\mu < 1$.

3.2 Overview

Here, In this paper, we have proposed load balancing algorithm based on the Chord DHT. We assume in the following discussion that each virtual server in Chord has a unique ID selected uniformly at random from the key space [0,1].

In our proposal, each peer $i \in N$ estimates the ideal load (i.e., Ii) it will host in a load-balanced state and then computes its remaining capacity Ci. If i is heavy, in order to balance i's load, i registers a subset of its local virtual servers with a pending pool P (see Fig. 1). Thus, if migrating such virtual servers, i's resultant load will be no more than Ii. By the "registration" of a virtual server v hosted by peer i, we mean that i hosts v and simply provides the tuple information $<L_v, i>$ to $\mathcal{P}$.

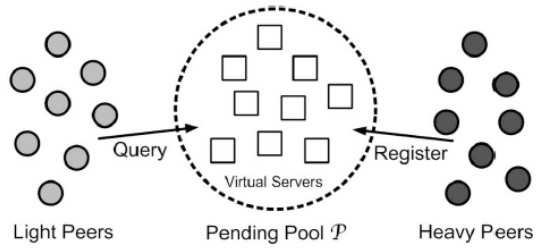

Fig. 1. The concept.

In contrast, if i is a light peer, i queries P and then allocates some virtual servers from corresponding peers. I requests the virtual servers in P as many times as possible without exceeding its Ii.

One key design aspect of our proposal is that for each virtual server v in $\mathcal{P}$, we estimate the cumulative distribution function (CDF), denoted by $\Pr(\mathbf{X} \preceq L_v)$, for the random variable X that represents the total load of virtual servers in P, with each having a load value no more than Lv. Additionally, we let $\Pr(\mathbf{Y} \not\preceq C_i)$ (or $\Pr(\mathbf{Y} \preceq C_i)$) represent the CDF for the random variable Y that denotes the total remaining capacity of light peers, with each having a remaining capacity smaller than (or no more than) Ci.

Each light peer i computes $\Pr(\mathbf{X})$ and $\Pr(\mathbf{Y})$. i then requests the virtual servers in P satisfying

$$\{v \in \mathcal{P} : \Pr(\mathbf{Y} \not\preceq C_i) \leq \Pr(\mathbf{X} \preceq L_v) \leq \Pr(\mathbf{Y} \preceq C_i)\}$$

Fig. 2 illustrates an example showing our idea, where there are five virtual servers **{v1,v2, v3, v4, v5}** with loads of **Lv1, Lv2 , Lv3 , Lv4 , Lv5** in P, and three light peers **{i1, i2, i3}** with remaining capacities of Ci1 , Ci2 , and Ci3 in the system As shown in Fig. 2, v1 manages the ratio of the load to the total load of virtual servers in P equal to (50/100) Such ratios for v2, v3, v4, and v5 are **(20/100), (10/100), (10/100) and (10/100**) respectively

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

$L_{v_1}^{\text{index}} = 0.5$, $L_{v_2}^{\text{index}} = 0.7$, $L_{v_3}^{\text{index}} = 0.8$, $L_{v_4}^{\text{index}} = 0.9$, and $L_{v_5}^{\text{index}} = 1.0$. Moreover, the ratio of the remaining capacity of i1 to the aggregate remaining capacity of the system is (50/100)

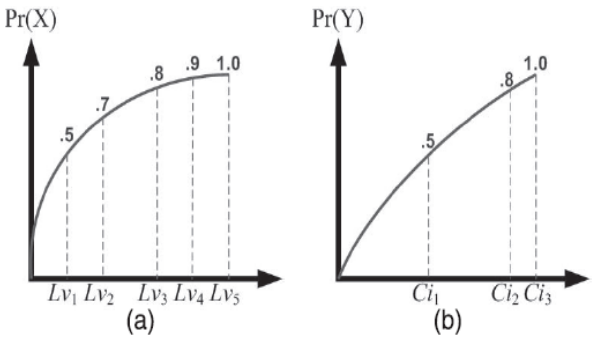

Fig. 2. A toy example, where (a) $\Pr(\mathbf{X})$ and (b) $\Pr(\mathbf{Y})$.

3.3 Histogram-Based Probability Distribution

Given a probability distribution PrðZÞ (where the random variable Z represents both X and Y), we rely on histograms as defined in the following to approximate the distribution.

Definition 8. *The histograms representing* $\Pr(\mathbf{Z})$ *are denoted by* $(\cup_{1\le i\le k}\{s_i\}, \Delta R)$. *Here,* $s_i < s_j$ *for any* i *and* j $(1 \le i \le j \le k)$, *and* $[s_i, s_{i+1}]$ $(i = 1, 2, \ldots, k-1)$ *indicates the interval in the domain of* $\mathbf{Z}$ *such that*

$$\Pr(s_i \le \mathbf{Z} \le s_{i+1}) = \Delta R \tag{9}$$

and

$$\sum_{i=1}^{k-1} \Pr(s_i \le \mathbf{Z} \le s_{i+1}) = 1, \tag{10}$$

where ΔR *is a given parameter and* $0 < \Delta R < 1$.

Fig. 3 depicts the concept of using histograms to approximate a probability distribution Pr(Z) given ΔR. We illustrate an example in Fig. 4, where there are 15 elements in the population for a CDF, Pr(Z). Among the 15 elements, 10 are sampled randomly to approximate Pr(Z). here, ΔR is 0.2, and there are five histograms, each $[s_i, s_{i+1}]$ $(1 \le i \le 4)$ containing two samples Note that each $[s_i, s_{i+1}]$ consists of three elements in the population. Consequently, the probability of samples appearing in each $[s_i, s_{i+1}]$ is $\Delta R = \frac{2}{10}$. This ideally approximates the probability of the elements within $[s_i, s_{i+1}]$ (i.e., $\frac{3}{15}$) in the population.

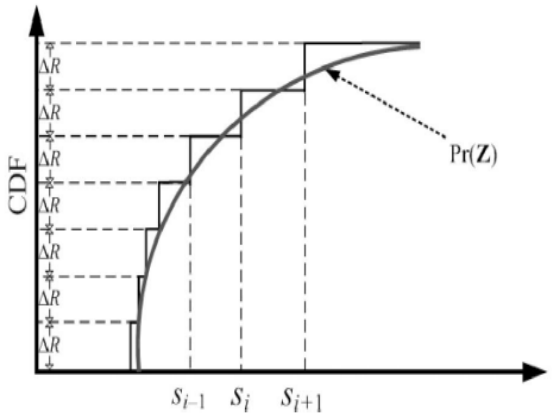

Fig. 3. The histogram-based approximation.

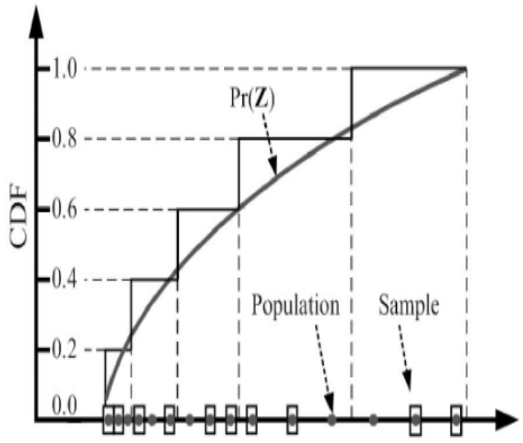

Fig. 4. An example with 15 elements (denoted by "•") in the population and 10 ones (denoted by "□") are randomly sampled (here, $\Delta R = 0.2$).

4 THEORETICAL PERFORMANCE ANALYSIS

4.1 Load Balance Factor

In our proposal, as each heavy peer selects its virtual servers with small sizes to migrate, the resultant movement cost is small. Thus, analyzing the load balance factor for each peer suffices. The load balance factor of peer I (denoted by LBFi) is defined as follows:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

$$LBF_i \triangleq \frac{\sum_{v \in V_i^{\mathcal{A}}} L_v}{C_i^{\max}},$$

where A represents our load balancing algorithm. Consequently, due to (1), we have

$$\left| \sum_{v \in V_i^{\mathcal{A}}} L_v - \frac{\sum_{v \in V} L_v}{\sum_{k \in N} C_k^{\max}} \times C_i^{\max} \right| = \left| C_i^{\max} \left(\frac{\sum_{v \in V_i^{\mathcal{A}}} L_v}{C_i^{\max}} - \mu \right) \right| = |C_i^{\max}(LBF_i - \mu)|.$$

As $C_i^{\max}$ is given, minimizing $|LBF_i - \mu|$ as much as possible suffices for our load balancing algorithm.

4.2 Analytical Results

Our analysis mainly relies on the Dvoretzky-Kiefer-Wolfowitz inequality [28] as stated in the following:

Theorem 1. Let Z1;Z2; . . . ;Zk be independent and identical distribution random variables with a CDF F(.). Let Fk(.) denote the empirical CDF, which approximates F(.), defined as

$$F_k(z) \triangleq \frac{\sum_{i=1}^{k} 1_{\{\mathbf{z}_i \le z\}}}{k}.$$

Then, for any $\varepsilon > 0$,

$$\Pr\Big(\sup_z |F_k(z) - F(z)| > \varepsilon \Big) \le \frac{2}{e^{2k\varepsilon^2}}.$$

Fig. 5 demonstrates the effectiveness of approximating a probability distribution based on sampling, where we only depict the probability distribution estimated by some peer (say, peer i) for the capacities of peers in the system (i.e., Pr(Y)). Notably, in the experiment, 1) Pr(Y) is a Pareto distribution, 2) **i** samples k = 50;100;200 peers to estimate Pr(Y), and 3) up to 10,000 peers are in the system. (Details of the experimental setting will be discussed in Section 5.) As we can see in Fig. 5, the approximation for a probability distribution based on sampling is very effective in case a modest number of samples (e.g., 200 samples among 10,000 peers) is performed.

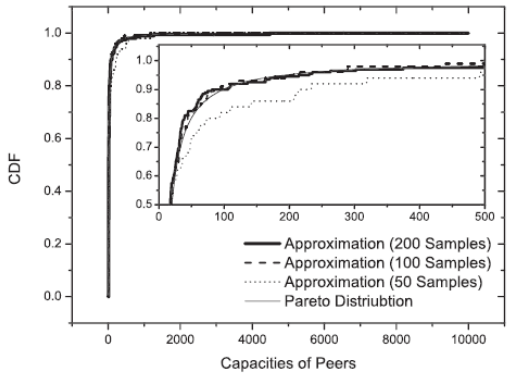

Fig. 5. The effectiveness of approximating a probability distribution based on sampling.

4.2.1 Load Balance Factor for Imprecisely Estimated Pr(X) and Pr(Y)

We are now ready to show the load balance factor of each participating peer through our imprecise estimation of Pr(X) and Pr(Y). We denote the probability distribution, our proposal estimates for the remaining capacities of the participating peers, as ~Pr(Y)

Lemma 3. *For any peer* $i \in N$, *let*

$$C_{\preceq i}^{\text{index}} = \Pr(\mathbf{Y} \preceq C_i), C_{\not\preceq i}^{\text{index}} = \Pr(\mathbf{Y} \not\preceq C_i),$$
$$\widehat{C_{\preceq i}^{\text{index}}} = \widetilde{\Pr}(\mathbf{Y} \preceq C_i),$$

and $\widehat{C_{\not\preceq i}^{\text{index}}} = \widetilde{\Pr}(\mathbf{Y} \not\preceq C_i)$. *For any* $0 < \Delta < 1$ *and* $\varepsilon > 0$, *if i samples* $k \ge \frac{\ln(\frac{2}{\Delta})}{\varepsilon \mathcal{C}_{\mathcal{A}}}$ *light peers uniformly at random from the system (where* $\mathcal{C}_{\mathcal{A}}$ *is the expectation of the remaining capacity of a participating peer), then with a probability of no less than* $1 - \Delta$,

$$|\widehat{C_{\preceq i}^{\text{index}}} - \widehat{C_{\not\preceq i}^{\text{index}}}| < (1 + 2\varepsilon)|C_{\preceq i}^{\text{index}} - C_{\not\preceq i}^{\text{index}}|.$$

Proof. As mentioned, through the Chernoff bound [31], if any peer i samples $k \ge \frac{\ln(\frac{2}{\Delta})}{\varepsilon \mathcal{C}_{\mathcal{A}}}$ peers with each peer j having Cj > 0, then

$$\Pr(|\mathbf{E}[\widehat{C_j}] - \mathcal{C}_{\mathcal{A}}| > \varepsilon \mathcal{C}_{\mathcal{A}}) < \Delta,$$

where $\mathcal{C}_{\mathcal{A}} = \mathbf{E}[C_j]$ for any peer j with $C_j > 0$, and

$$\mathbf{E}[\widehat{C_j}] = \frac{\sum_{1 \le l \le \|S\|} C_l}{k}$$

is the value our proposal estimates for CA based on the set S of the k samples. But we have

$$|\widehat{C_{\preceq i}^{\text{index}}} - \widehat{C_{\not\preceq i}^{\text{index}}}| = \frac{C_i}{\|N\| \mathbf{E}[\widehat{C_j}]}.$$

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Due to above last two equations, and $\frac{1}{1-\varepsilon} < (1+2\varepsilon)$, with a probability no less than $1-\Delta$,

$$\left|\widehat{C}_{\preceq i}^{\text{index}} - \widehat{C}_{\not\preceq i}^{\text{index}}\right| < \frac{C_i}{\|N\|(1-\varepsilon)\mathcal{C}_{\mathcal{A}}} < (1+2\varepsilon)\frac{C_i}{\|N\|\mathcal{C}_{\mathcal{A}}} = (1+2\varepsilon)\left|C_{\preceq i}^{\text{index}} - C_{\not\preceq i}^{\text{index}}\right|.$$

Fig. 6 shows the simulation results. Here, we measure the load deviation (i.e $|C_i^{\max}(LBF_i - \mu)|$ for each participating peer i by varying the number of samples performed by each participating peer. n and m are 10,000 and 100,000, respectively, and the number of samples investigated in the experiment is 100, 200, 400, 800, 1,600, 3,200, and 6,400. In Fig. 6, we also provide the theoretical upper bound based on (21) in Theorem 3 for each participating peer. More specifically, each peer i has the expected load deviation of less than $(1+2\varepsilon)\delta\mu \times C_i^{\max}$. Notably, the load deviation presented in Fig. 6 is averaged over all participating peers. As seen in Fig. 6, the measured load deviation is clearly bounded from above by the theoretical result

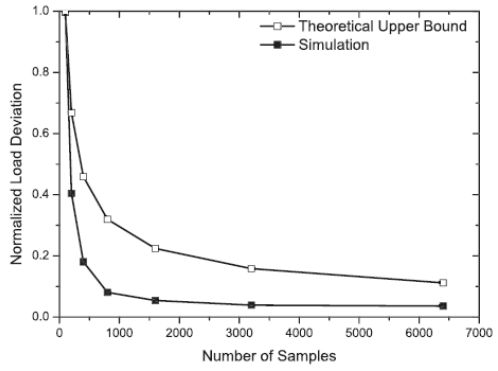

Fig. 6. Theoretical upper bound versus the simulation result (here, $t_1 = t_2 = 0.99$).

5 SIMULATIONS

5.1 Simulation Results

Fig. 7 presents the simulation results for the load balance factor, where W/O represents the distribution of the load balance factors of participating peers before the load balancing algorithms are performed. As shown in Fig. 7, the studied load balancing algorithms improve the load balance factor. Precisely, there are around 98, 70, 60, 70, and 90 percent of nodes with load balance factors within (0:7, 0:9) in the centralized directory approach (Dir), the tree-based solution (Tree), the two-tier-based solution (2-tier), Hsiao's algorithm (Hsiao), and our proposal (Ours), respectively.

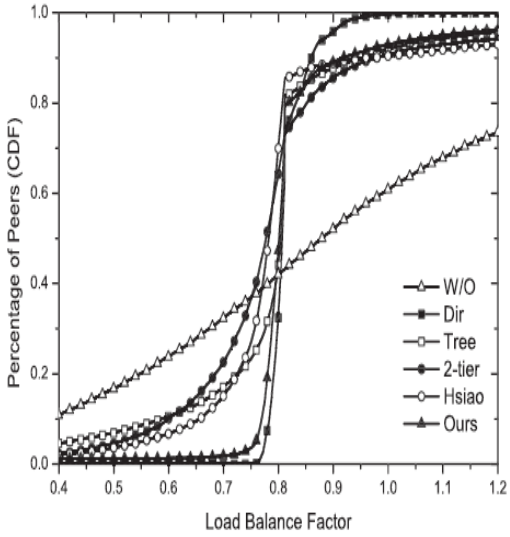

Fig. 7. The load balance factor.

Fig. 8 shows the movement costs for Dir, Tree, 2-tier, Hsiao, and our proposal. Dir introduces a significant movement cost, as the centralized directory maintains the global knowledge and thus performs as many matches as possible. Compared with Dir, our proposal generates less movement cost. Notably, although our proposal has movement cost comparable with that of Tree, 2-tir, and Hsiao, it obviously outperforms Tree, 2-tir, and Hsiao in terms of the load balance factor. Unlike Tree, 2-tir, and Hsiao, our load balancing algorithm, which is based on the estimation of the probability distributions of the capacities of participating peers and the loads of virtual servers, is very effective, such that the virtual servers with heavy (light) loads can be effectively reallocated to capable (incapable) peers.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

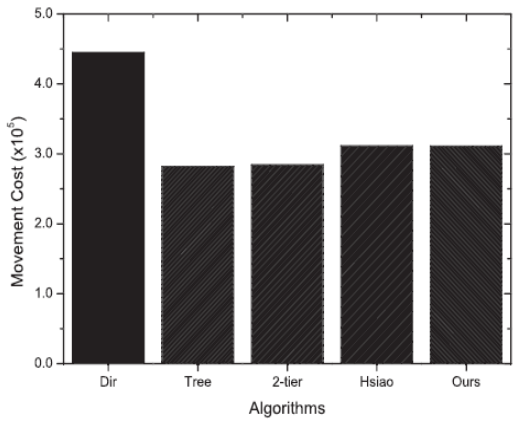

Fig. 8. The movement cost.

We present in Fig. 9 the protocol message overhead, where the details of the protocol behaviors in Dir, Tree, 2-tier, Hsiao, and our proposal are simulated. Hsiao takes more message overheads because in Hsiao, a virtual server may receive multiple reallocation requests issued from different light peers. In contrast, in Dir, Tree, and 2-tier, a virtual server selected for migration receives only one reallocation request from a light peer. The reason is that these approaches heavily rely on rendezvous nodes to match virtual servers and light peers.

Fig. 10 further details the number of messages sent by each participating peer. We plot the CDF function regarding the number of messages sent by each peer in Fig. 10. Fig. 10 illustrates that while each peer in Hsiao and our proposal introduces a nearly identical number of protocol messages to manipulate its load balancing algorithm, ours considerably outperforms Hsiao as each peer in our proposal takes ≈50% of messages required in Hsiao. In contrast to ours, in Dir, Tree, and 2-tier, the participating peers perceives imbalance algorithmic workloads in balancing their loads. In particular, several peers (i.e., the rendezvous) in Dir, Tree, and 2-tier generate considerable messages for matching virtual servers and light peers, introducing another load imbalance issue due to the load balancing algorithms. Compared with Dir, Tree, and 2-tier, ours performs very well in terms of loads due to the load balancing algorithm, thus simplifying the system provisioning and maintenance. This validates our analytical results in Lemma 2 and Corollary 1.

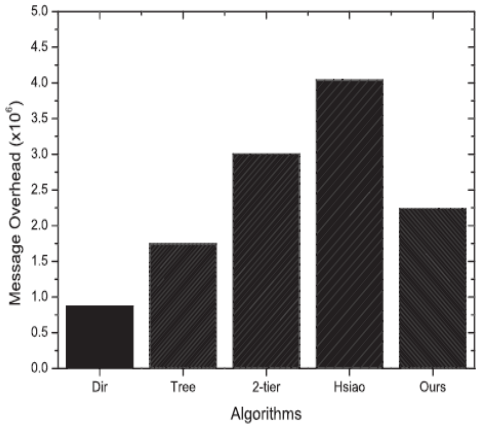

Fig. 9. The protocol message overhead.

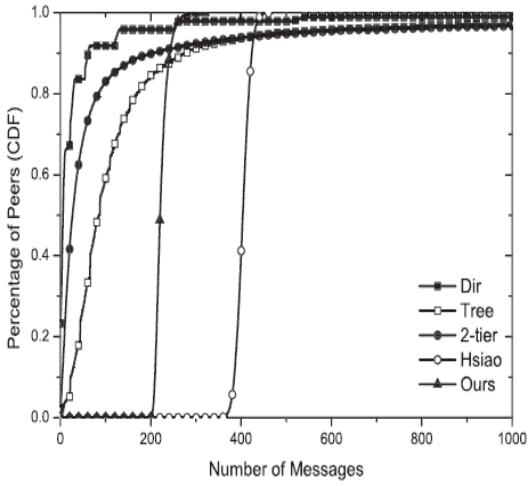

Fig. 10. The distribution of protocol messages.

5.1.1 Convergence

In this section, we demonstrate the load balance factor against algorithmic rounds for the 1-, 5-, 10-, 90-, 95- and 99-percentile peers. Fig. 11 depicts the simulation results, where the algorithmic rounds investigated for our proposal are 1, 2, 3, and 10. Here, the 0 round denotes the initial situation that no peer performs our proposal and that the y-axis is in a logarithmic scale. The experimental results clearly demonstrate that the light and heavy peers in our proposal improve their load balance factors through the algorithm rounds toward the ideal value.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

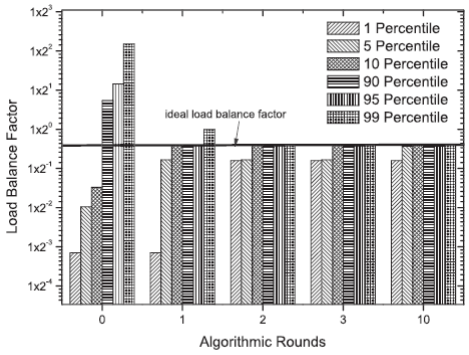

Fig. 11. The load balance factor versus the algorithm rounds.

5.1.2 Effect of Numbers of Samples

A few samples of the capacities of peers and the loads of virtual servers are sufficient to estimate the system state effectively, i.e., the probability distribution for the capacities of peers (i.e., Pr(Y)) and the probability distribution for the loads of virtual servers (i.e., Pr(X)). Note that in our implementation, each peer issues a random walker to sample the capacities of peers and the loads of virtual servers simultaneously. That is, when a random walker visits a peer, the walker not only collects the capacity value of the visited peer but also gathers the load values of the virtual servers the visited peer selects to migrate.

Fig. 12 shows the simulation results for the effect of different numbers of random walk steps in our proposal. Apparently, 200 walk steps performed by each peer are sufficient to estimate the system state, such that the resultant load balance factors of the participating peers are comparable with those based on more walk steps (e.g., 400 and 800).

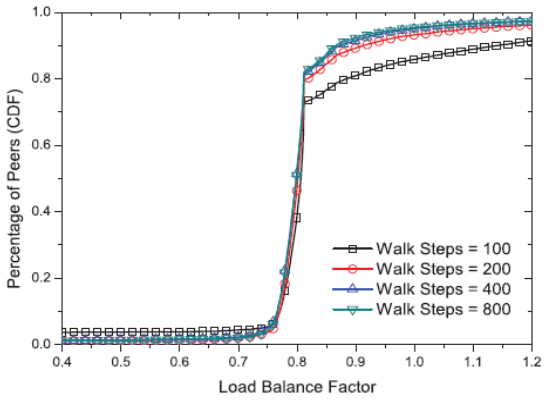

Fig. 12. The effect of varying walk steps, k.

5.1.3 Effect of ρ_i

To cope with the approximation error each light peer i seeks virtual servers with load indices within $[C_{\tilde{i}}^{\text{index}} - \rho_i, C_{\tilde{i}}^{\text{index}} + \rho_i]$. We investigate the effect of various values of _i on the resultant load balance factor of i. Fig. 13 presents the simulation results. In Fig. 13, although our proposal can effectively estimate Pr(X) and Pr(Y), it cannot perform very well without dealing with the approximation error (i.e., $\rho_i = 0$ in Fig. 13). In contrast, if $\rho_i > 0$, then a virtual server in the pending pool is likely to be acquired by multiple light peers.

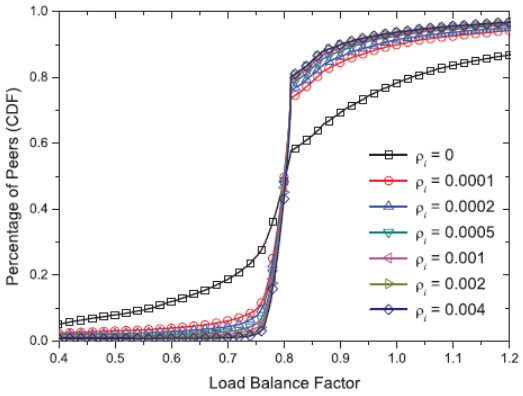

Fig. 13. The effect of varying ρ_i.

5.1.4 Effect of System Dynamics

In this section, We study the impact of system dynamics on our proposal. In the experiment, we first stabilize the system for 20 minutes, where no peer joins and leaves the system. Then, peers start to join and leave the system, and the expected lifetime of each participating peer is 100 minutes. We investigate the system operating for 3,600 minutes. Given the probability distribution of the capacities of peers, the probability distribution of the loads of virtual servers is varied during the 3,600-minute simulation period, such that the ideal load balance factors in the periods of [0, 1,200], [1,200], [2,400], and [2,400], [3,600] are 0.81, 0.16, and 0.49, respectively. Note that in the experiment, each message takes 0.1 minute in our proposal to traverse an overlay link.

As a result, at most 200 distinct peers are visited by a

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

random walker every 20 minutes. Each participating peer performs our load balancing algorithm every 20 minutes.

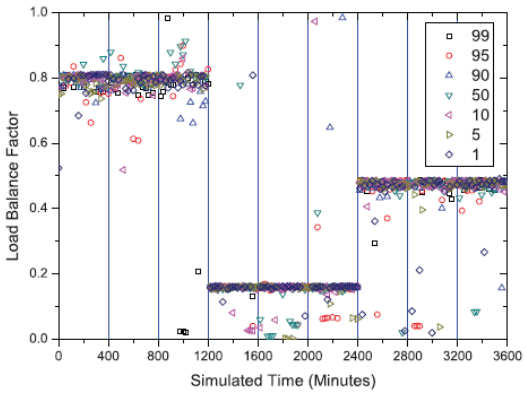

Fig. 14. The effect of system dynamics.

Fig. 14 illustrates the simulation results, where we measure the load balance factors for the 1-, 5-, 10-, 50-, 90-, 95-, and 99-percentile peers. The simulation results indicate that our proposal performs very well as the participating peers strive to manage their load balance factors close to the ideal value even if the ideal load balance factor varies over time.

6 CONCLUSION

In this project we achieve a load balancing algorithm for the reallocation of virtual servers in DHTs. Our load balancing algorithm operates in a fully decentralized manner by having each participating peer estimate the probability distribution of loads of virtual servers selected for migration and the probability distribution of the remaining capacities of under-loaded peers.

REFERENCES

[1] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-PeerLookup Protocol for Internet Applications," IEEE/ACM Trans.Networking, vol. 11, no. 1, pp. 17-21, Feb. 2003.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed ObjectLocation and Routing for Large-Scale Peer-to-Peer Systems," Proc.IFIP/ACM Int'l Conf. Distributed Systems Platforms, pp. 161-172,Nov. 2001.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A.Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W.Vogels, "Dynamo: Amazon's Highly Available Key-Value Store,"Proc. 21st ACM Symp. Operating Systems Principles (SOSP '07),pp. 205-220, Oct. 2007.

[4] BitTorrent, http://www.bittorrent.org/index.html, 2012.

[5] J. Stribling, E. Sit, M.F. Kaashoek, J. Li, and R. Morris, "Don't GiveUp on Distributed File Systems," Proc. Sixth Int'l Workshop Peer-to-Peer Systems (IPTPS '07), Feb. 2007.

[6] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica,"Load Balancing in Structured P2P Systems," Proc. Second Int'lWorkshop Peer-to-Peer Systems (IPTPS '02), pp. 68-79, Feb. 2003.

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

[8] C. Chen and K.-C. Tsai, "The Server Reassignment Problem forLoad Balancing in Structured P2P Systems," IEEE Trans. ParallelDistributed Systems, vol. 12, no. 2, pp. 234-246, Feb. 2008.