Stable Load Balancing with Overlapping ID-space Management in Range-based Structured Overlay Networks

Kimihiro Mizutani　Takeru Inoue　Toru Mano　Osamu Akashi
Satoshi Matsuura　Kazutoshi Fujikawa

Structured overlay networks that support range queries cannot hash data IDs for load balancing, in order to preserve the total order on the IDs. Since data and queries are not equally distributed on the ID-space without hashing in range-based overlay networks, uneven loads are imposed on the overlay nodes. Existing load balancing techniques for range-based overlay networks distribute the loads by using data reallocation or node migration, which makes the networks very unstable due to heavy data reallocation or frequent churn.

This paper proposes a novel scheme that distributes, fairly, the loads without node migration and with little data reallocation, by sharing some ID-space regions between neighboring nodes. Our "overlapping" ID-space management scheme derives the optimal overlap based on kernel density estimations; the query loads based on the statistical theory are used to calculate the best overlap regions. This calculation is executed in a distributed manner with no central coordinator. We conduct thorough computer simulations, and show that our scheme alleviates the worst node load by 20–90 % against existing techniques without node migration and with least data reallocation.

1 Introduction

A structured overlay network with N nodes provides scalable search with just $O(\log N)$ messages [14] [17] [19]. Each node in the network maintains a small portion of "range" of the ID-space, which is structurally coordinated by pointers between the nodes for efficient message routing. Conventional structured overlay networks provide only exact match searches, so only the data with the specified ID is returned. Users, however, are demanding support for range queries, which retrieve all the data in the given ID range [1] [4] [9] [13]. Range queries are an efficient way to retrieve data, since they eliminate the need to issue individual queries for all IDs in the range. Unfortunately, overlay networks that support range queries cannot hash the data IDs since doing so would disturb the total order, a property essential for range queries. Note that hashed IDs are indispensable to distribute query loads over nodes in conventional overlay networks. As a result, the loads tend to become heavily skewed for supporting the range queries (Fig. 1(a)).

There are two major solutions to load balancing with range queries. One is range reallocation; like NIX [12] (Fig. 1(b:top)), each node expands or shrinks its own ID-space range in response to its load, but this range reallocation can cause global reallocation involving very distant nodes and the transfer of massive amounts of data among nodes. The other is node migration; as shown by Mercury [5] (Fig. 1(b:bottom)), unloaded nodes are moved into heavily loaded ID-space regions, and

---

範囲探索可能な構造化オーバレイにて Overlap ID-Space を用いた安定的な負荷分散手法

水谷后宏, 井上武, 間野暢, 明石修, NTT 未来ねっと研究所, NTT Network Innovation Labs.

松浦知史, 東京工業大学, Tokyo Institute of Technology.

藤川和利, 奈良先端科学技術大学院大学, Nara Institute of Science And Technology.

コンピュータソフトウェア, Vol.32, No.3 (2015), pp.101–110.
[研究論文] 2014 年 7 月 18 日受付.

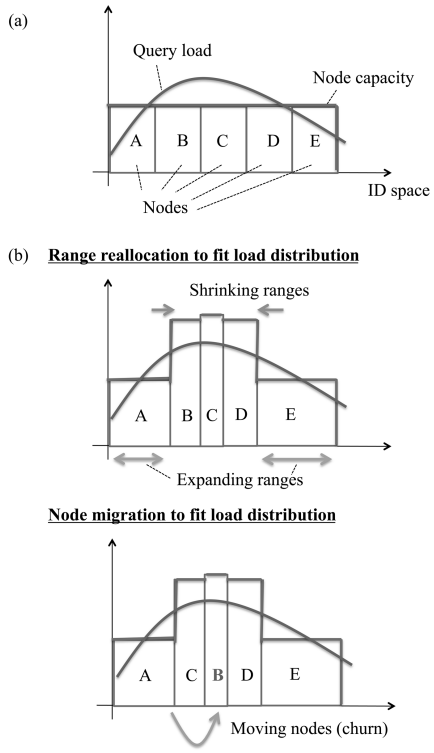

Fig. 1: Unbalanced loads in overlay networks with range queries; each box with alphabet tag represents a node, each node maintains an ID-space range of box width with the capacity being the box area. (a) Since data IDs are not hashed for range queries, queries can be unevenly distributed over the ID-space. (b) Existing techniques realize load balancing by reshaping boxes to fit the load function, and/or by moving some boxes from highly-loaded regions to lightly loaded regions. These techniques involve massive data transfers at several box boundaries or frequent node migration.

share the region with overloaded nodes to suit the load function. Such node migration raises churn frequently, which makes overlay networks very unstable. Very few existing works have attempted load balancing without relying on data or node migration [16], but these approaches often fail to balance the loads since they handle the loads indirectly.

This paper proposes a novel ID-space management scheme with overlapping ID-space regions. As shown in Fig. 2, nodes share some ID-space regions with neighboring nodes, and this "overlap" allows the nodes to reshape their capacity function locally. The benefits of our overlapping ID-space management scheme comes at the cost of duplicated data storage, but it is freed from the globally cascading range reallocation or any node migration. We use kernel density estimation in deriving the optimal overlap ranges [15], and the node capacity function is changed to well fit the load function. Kernel density estimation is a well-studied statistical method that yields a function that well fits a set of given data points, and so our scheme has the mathematical rationale unlike existing techniques. We design this optimization process to run in a distributed manner. Our scheme is evaluated by intensive computer experiments, the results of which reveal that our scheme provides fairer load balancing than existing techniques with no significant overheads.

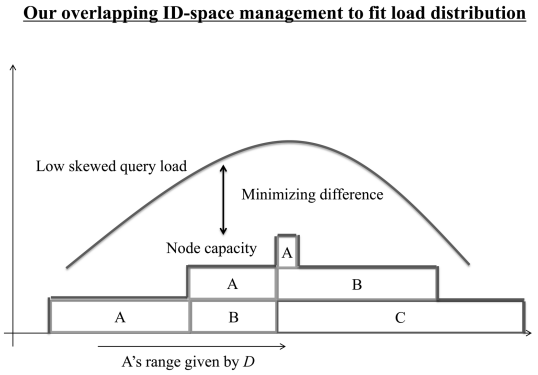

Fig. 2: Basic idea of our scheme. Overlapping regions, in which the boxes are accumulated, provide larger capacity for query processing. Our scheme can find a capacity distribution function (the ridge of boxes) that approximately fits the query distribution by minimizing the difference with D.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes our scheme and Section 4 reports the extensive experiments conducted and their results. Section 5 concludes this paper.

2 Related work

This section discusses related work on load balancing for range-based structured overlay networks.

Neighbor item exchange (NIX) [12] is a range reallocation scheme. A node changes its range and transfers the out-of-range data to its neighbors, when the amount of data stored on it exceeds a

predefined limit. This transfer process continues until the nodes satisfy the limit, and so reallocation can cascade on a massive scale. Reference [2] proposed a range reallocation scheme that uses skip graphs [3]. This scheme rearranges ID-space ranges, called buckets, based on the amount of data in a bucket. The clustering approach introduced in [18] exchanges data within a cluster, locally if possible, but inter-cluster reallocation may be needed if the load distribution is heavily unbalanced. Massive data reallocation is a common problem with these schemes.

Mercury [5] is a node migration scheme designed for efficient load balancing, which moves unloaded nodes to heavily loaded regions in order to balance the load among more nodes. The load distribution is estimated using $O(\log^2 N)$ load query messages; a node sends a load query to $O(\log N)$ randomly selected nodes, which also send the query at random. Another load estimation method was proposed in [11]. This method realizes 99 % accurate estimations with $O(\log^3 N)$ messages. Reference [20] employs a tree-like overlay network to collect load information, but we note that the tree root can be a single point of failure. These schemes rely on frequent node migration to balance the load, and heavy churn is triggered as shown in the experiments.

Reference [12] proposed a hybrid scheme that fuses range reallocation and node migration. This scheme, named NIXMIG, deals with slightly unbalanced load distributions by range reallocation, while heavily unbalanced loads are corrected by node migration.

There are few studies that try to avoid heavy data exchange and frequent churn. Saturn [16] employs data replication to balance the load without data or node migration, like our scheme. Saturn takes, as node load, the product of the amount of data per node and the number of get queries per data. Saturn only controls the number of queries per data and ignores the amount of data per node. As a result, Saturn can fail to balance the node load. In contrast, our scheme directly handles the node load, as shown in Fig. 2.

We define two methods for determining the D, in the following subsections.

3 Load balancing with overlapping ID-space management

This section describes our novel ID-space management scheme that assigns ID-space ranges to overlay node allowing some overlaps with the neighbors. Routing mechanisms for the ID-spaces are also discussed. We assume one dimensional ID-space in this paper, but we believe our scheme can easily be extended to cover multi-dimensional spaces. Our scheme is agnostic as to the routing protocol, and so we do not assume any particular routing protocol unless otherwise noted.

3.1 Overlapping ID-space management

In our scheme, ID-space range R_i, assigned to node i, is given by,

$$R_i = [s_i, \max(suc_i, s_i + D)),$$

where s_i is node ID and suc_i is the ID of its successor (e.g., the clockwise neighbor in Chord). Here, D is a key variable in our scheme; larger D such that $s_i + D > suc_i$ allows the range to overlap with that of the successor, unlike conventional structured overlay systems.

Figure 2 shows basic idea of our scheme. The overlapping ID-space regions provide more query processing capacity, as shown in the figure. This capacity function defined over the ID-space is called *capacity distribution* in this paper. Queries not covered by the capacity distribution cannot be processed. Here, we define *load balancing* as the problem of maximizing query distribution coverage by rearranging the distributed capacity. The query distribution is statistically estimated by the similar method as Mercury. With the estimated query distribution, each node updates D independently and tries to balance skewed loads without data and node migration[†1]. Our scheme is executable at all times; e.g. when a load factor [6] calculated by the query distribution exceeds certain threshold.

3.2 Average-based estimation method

Our first method is simple in that it takes advantage of the average width of the ID-space ranges; existing load balancing techniques often rely on the

†1 Each node calculates D independently, but these values are theoretically matched. Therefore, we denote these values by the single symbol.

average value. We extend ranges that are shorter than the average, and so mitigate the skewed load distribution. Each node independently estimates D as follows,

$$D = \frac{\sum_{i \in P} suc_i - s_i}{|P|},$$

where P is the set of nodes from which ranges $suc_i - s_i$ are sampled. Nodes of P are randomly selected following [5], and only $O(\log^2 N)$ messages are required to estimate the average.

3.3 Distribution-based estimation method

Our second method utilizes the query distribution, not its simple average; the method normalizes the query and capacity distribution functions, and then determines D so as to minimize their difference. It results the heavy load regions are covered from many nodes and the skew loads are flatted in overall overlay network. This method employs kernel density estimation [15], which is used to minimize the difference between a set of data samples and a function. The normalized query distribution function, $f(x)$, is estimated as,

$$f(x) = \frac{L_x}{\sum_{j \in Q} L_j},$$

where x is a point in the ID-space, L_x is the query load at x, and Q is the set of IDs at which query loads are sampled; we can also use the sampling method of [5]. The normalized capacity distribution is given by,

$$\hat{f}(x, D) = \frac{1}{ND} \sum_{j=1}^{N} K\Big(\frac{x - j}{D}\Big),$$

where the number of nodes, N, is estimated by the sampling technique as well, and $K(\frac{x-j}{D})$ is a kernel function that is equal to one if node j maintains ID x as follows,

$$K(x) = \begin{cases} 1 & \text{if } |x| < 1/2, \\ 0 & \text{otherwise.} \end{cases}$$

We minimize the square error of these distribution functions as follows,

$$\arg\min_{D} \sum_{j \in Q} |\hat{f}(j, D) - f(j)|^2. \quad (1)$$

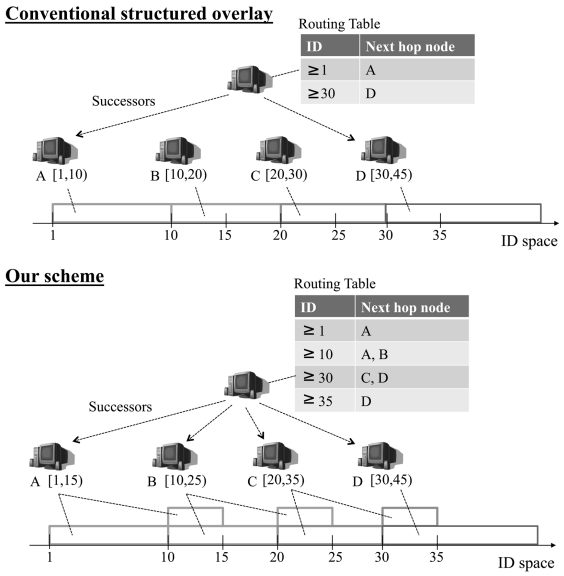

Fig. 3: Routing tables in conventional overlay networks like Chord [19] and our scheme with $D = 15$. In our scheme, nodes that maintain a common region are stored together in a routing table, in order to query the region; e.g., nodes B and C have to be added to the table, since their ranges overlap those of nodes A and D.

To simplify the computation, we solve this optimization problem approximately by assuming that binary search is a convex function. It can be completed in a logarithmic time against the ID-space size. The minimum error is not bounded theoretically, and node loads are evaluated in the experiments.

3.4 Message routing

In our scheme, data is stored on all nodes that maintain the data's ID, and is retrieved from one of these nodes at random. All nodes maintaining common IDs must populate the same routing table, as shown in Figure 3; such common nodes can be found easily in a structured overlay network. This duplication of data and routing entries imposes storage overhead on nodes. The duplicated routing entries might detour a message, and increase hop counts to a destination. We evaluated these overheads in the experiments. We do not discuss data synchronization issues among nodes maintaining overlapping ranges, since they have been stud-

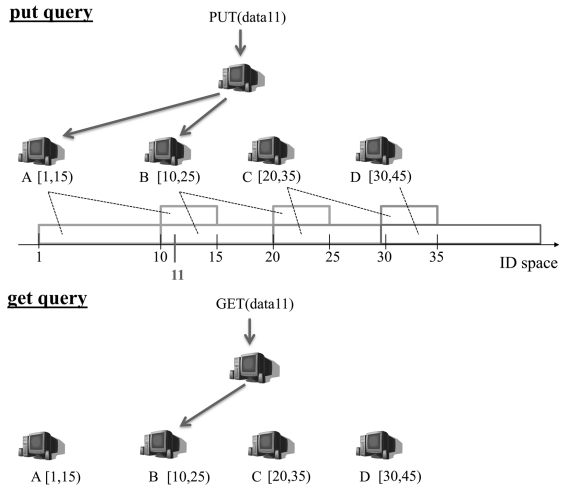

Fig. 4: Message routing in `put` and `get` queries. The `put` query is routed to all nodes maintaining the data ID, while the `get` query is routed to one of them at random. We assume the top node has the routing table shown in Figure 3 (bottom).

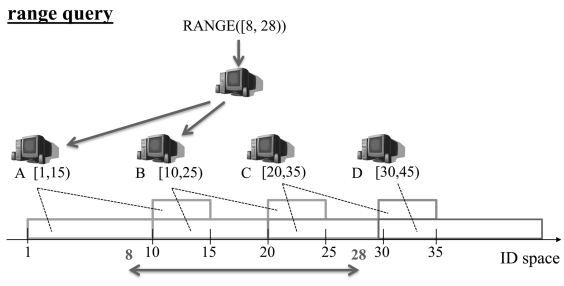

Fig. 5: Message routing in a `range` query. The query is forwarded to the appropriate nodes to cover the whole range to be queried. Overlapping regions like $[10, 15)$ can be covered by any node maintaining the region; e.g., queries sent from the top node are (1) $[8, 15)$ to A and $[15, 28)$ to B, or (2) $[8, 10)$ to A and $[10, 28)$ to B; then, B sends query $[25, 28)$ to C. We assume the top node has the routing table shown in Figure 3 (bottom).

ied intensively [8].

Figures 4 and 5 show routing procedures for `put`, `get`, and `range` queries. They do not include any procedure specific to a particular routing protocol, so our scheme can be integrated with any routing protocol; the only requirement is that the protocol maintains routing entries for nodes of common regions.

Table 1: Mandelbrot-Zipf distribution of P2P traffic

AS #	Parameters q and α
AS1	5, 0.47
AS2	25, 0.78
AS3	55, 0.6
AS4	25, 0.55

4 Experiments

We evaluated our scheme against existing schemes including NIX [12], Mercury($\alpha = 0.8$) [5], NIXMIG [12], and Saturn [16] through computer simulations. Similar to Mercury, we set node, data, and query distributions following Zipf distributions. Though our scheme is designed not to be application specific, we consider a kind of geo-location application in the experiments; the ID-space represents locations in Japan sorted in descending order of population and the size is 2^{31}. It roughly follows a Zipf distribution[†2]. We first generate 1,000 nodes with their IDs, assuming that these nodes are carried by people. The nodes then construct an overlay network with coordination by Chord protocol [19] unless otherwise noted. We place 10^5 data on the overlay network and publish 10^7 `get` queries for the data. These data and queries are randomly generated following the Mandelbrot-Zipf distribution [7] [10], as measured in real P2P traffic for certain ASes(Autonomous Systems) [10] (Table 1). Finally, a load balancing scheme runs to distribute the query load, and another set of 10^7 queries were published to measure the load at each node.

4.1 Estimation errors

We first evaluate estimation errors of our distribution-based estimation method. Figure 6 shows relative errors of D yielded by the method. We repeated the sampling process for the estimation for one to five rounds. As shown in the figure, 90 % of nodes estimated D with less than 7 % error in the first round, while 99 % of them had errors under 2 % in the fifth round. We present no evaluation result for the average-based estimation

†2 http://www.stat.go.jp/data/jinsui/2011np/

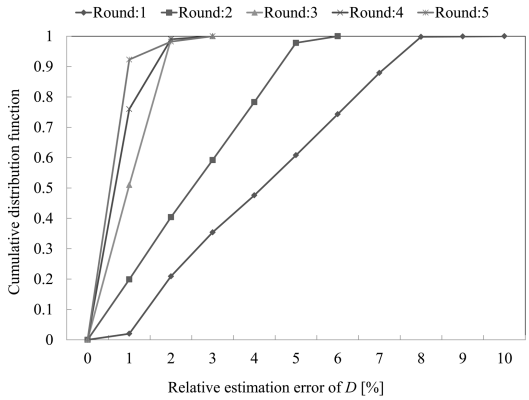

Fig. 6: Cumulative distribution functions of relative error of D as estimated by the distribution-based estimation method. Errors shrink with sampling round number and D calculated by each nodes went to approximately 1.5×10^6. Estimation error for 99 % of nodes was less than 2 % after just five rounds.

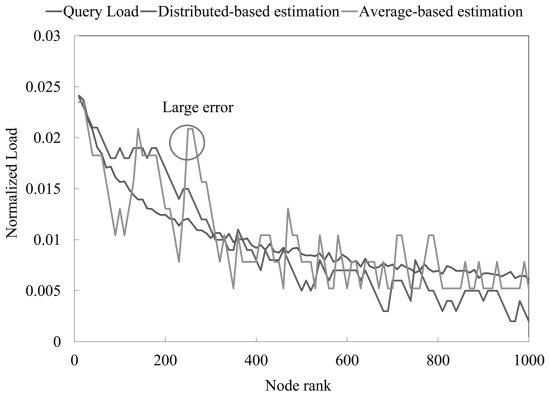

Fig. 7: Load distribution $f(x)$, and capacity distributions $\hat{f}(x)$'s with the average-based and distribution-based estimation methods. In this figure, load distribution is plotted as a function of original load values measured at node points, while capacity distributions are given as functions of node capacities at the corresponding node points. The capacity distributions roughly follow the load distribution thanks to our estimation methods.

method, but it shows similar behavior. We ignore these small estimation errors in the following experiments.

Figure 7 shows load distribution $f(x)$ and capacity distributions $\hat{f}(x)$'s created by our average-based and distribution-based estimation methods. We present only the result of AS1's load distribution, other ASes exhibited identical behavior. The capacity distributions based on the D well fit the load distribution. The distribution-based method is slightly better than the average-based method; it has no large error, and the square error (1) of the distribution-based method was 13 % smaller than that of the average-based method. This is because the distribution-based method considers the whole distribution, while the average-based method focuses on just a single mode.

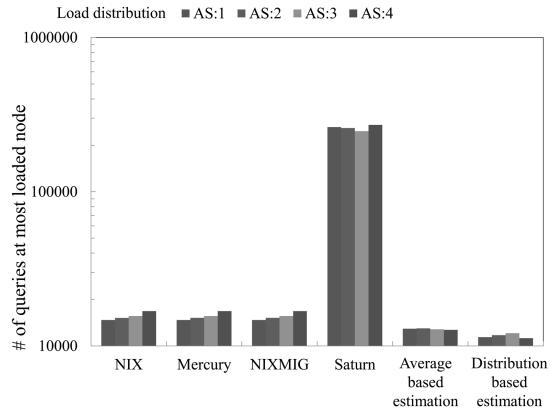

Fig. 8: Most loaded nodes yielded our schemes and the existing schemes. Our schemes with the two estimation methods yield smaller loads for the other schemes.

4. 2 Load balancing performance

In order to evaluate the load balancing performance, we compare the Most heavily loaded nodes yielded by our scheme and the existing schemes. Figure 8 shows the results for the four ASes. Our schemes reduced the maximum load by 90% for Saturn and by 20% for the others. Of our schemes, the distribution-based estimation method slightly outperforms the average-based estimation method. This great reduction comes from the estimation accuracy shown in Figure 7. Since Saturn yielded very poor performance, we drop Saturn from the following discussion.

4. 3 Data and node migration

We evaluated data and node migration in the load balancing process. Figure 9 shows the num-

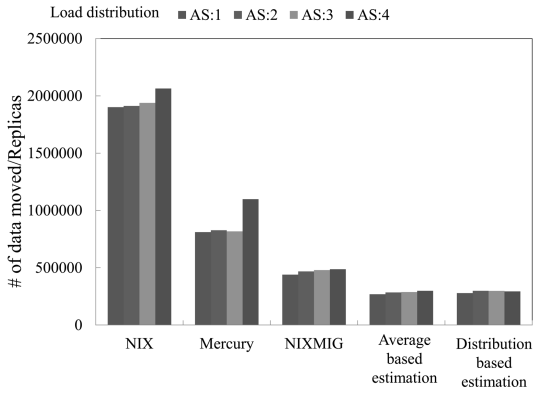

Fig. 9: The number of data moved/replicated to other nodes during load balancing. Our scheme with the two estimation methods migrated/replicated much less data than the other schemes.

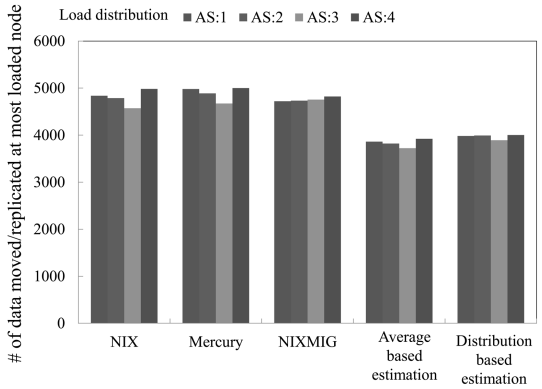

Fig. 10: The number of data moved/replicated at the most loaded node. Our scheme with the two estimation methods migrated/replicated less data than the other schemes.

ber of data moved/replicated to other nodes. NIX moved nearly 20 % of all data for load balancing due to the cascade of data reallocation; the other schemes needed much less data migration. Mercury and NIXMIG migrated slightly more data than ours, which was caused by node migration. Figure 10 shows the maximum number of data migrated/replicated to other nodes per node. This paper mainly focuses on the load balancing over `get` queries, and our scheme has to replicate data

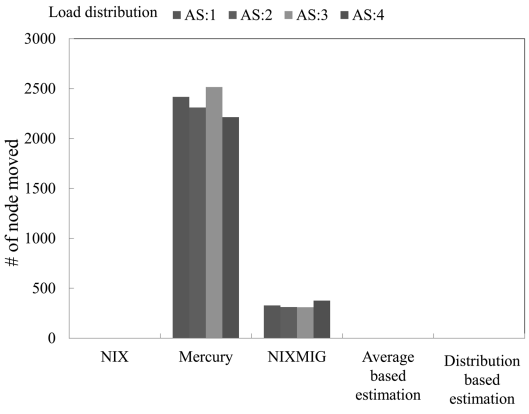

Fig. 11: Number of nodes moved for load balancing. Our proposal with the two estimation methods needed no node migration.

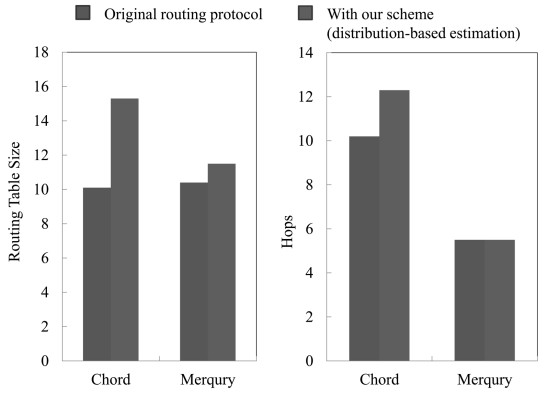

Fig. 12: (left) Average number of entries in a routing table, and (right) average number of hops routed to a destination node per `get` query. Three routing protocols, Chord and Mercury were executed with and without our scheme with distribution-based estimation method. No significant overhead was found in our scheme. Load distribution of AS1 was used for the plots.

for `put` queries. However, it was still least loaded in terms of the most loaded node.

Figure 11 shows the number of node moved in the load balancing process. Our scheme and NIX issues no join nor leave messages for node relocation, while Mercury and NIXMIG moved several nodes. The results confirm that our scheme realized fair load balancing without churn and with less data reallocation.

4.4 Overheads of our scheme

We evaluate the overheads of our scheme, such as duplicated routing entries and extra routing hops. We used routing protocols including Chord [19] and Mercury [5], in order to examine the dependency of the overheads to routing protocols. Figure 12 (left) shows average number of entries per routing table for each routing protocol with and without our scheme. As shown in the figure, our scheme yielded 10 % to 30 % duplication in routing entries; Chord had the largest duplication, since it is based on the ID distance while others are based on the hop counts. Figure 12 (right) shows average number of hops to route a `get` query to a destination. Only Chord incurred overhead here because it is based on ID distance. As the results show that overhead of our scheme is acceptable, and did not significantly depend on routing protocol. Finally, we discuss storage cost. Our scheme replicated each data item to twenty nodes on average among all ASes; this roughly matches Saturn, and is much larger than the other schemes. However, it is worth noting that despite the data replication, our scheme moved (copied) much less data than the others (Fig. 9). If storage cost is relatively cheaper than processing and network costs (this is the most likely scenario in current computer systems), the proposal offers a better trade-off for fair load balancing.

5 Conclusion

This paper introduced a new load balancing scheme for range-based structured overlay networks. Our scheme allows the ranges of overlay nodes to overlap, and reshapes the node capacity distribution to fit the load distribution. Kernel density estimation is used to determine the optimal value of range width D, for curve fitting. Our scheme was evaluated against existing work including NIX, NIXMIG, Mercury, and Saturn. Simulations showed the value of D can be estimated with only 2% error in a distributed manner. The load of most heavily loaded node was eased by more than 20-90 % without churn and with minimal data reallocation. The routing table overhead was quite small and not significant.

In this paper, we formulated load balancing as curve fitting, and implemented it as a distributed algorithm. This idea is quite general and not limited to structured overlay networks. We believe that our work sheds new light on the traditional load balancing problem through its new viewpoint of using statistical techniques.

In future work, we will evaluate its estimation stability under dynamic loads. We will investigate another estimation method with more variables to handle more complicated load distributions.

References

[1] Aguilera, M. K., Golab, W. and Shah, M. A.: A practical scalable distributed B-tree, *Proc. VLDB Endow.*, Vol. 1, No. 1(2008), pp. 598–609.

[2] Aspnes, J., Kirsch, J. and Krishnamurthy, A.: Load balancing and locality in range-queriable data structures, in *Proc. the twenty-third annual ACM symposium on Principles of distributed computing*, PODC '04, New York, NY, USA, ACM, 2004, pp. 115–124.

[3] Aspnes, J. and Shah, G.: Skip graphs, *ACM Transactions on Algorithms*, Vol. 3, No. 4(2007), pp. 1–25.

[4] Beltran, A., Sage, P. and Milligan, P.: Skip Tree Graph: a Distributed and Balanced Search Tree for Peer-to-Peer Networks, in *Proc. IEEE International Conference on Communications*, 2007, pp. 1881–1886.

[5] Bharambe, A. R., Agrawal, M. and Seshan, S.: Mercury: supporting scalable multi-attribute range queries, *ACM SIGCOMM Computer Communication Review*, Vol. 34, Issue 4 (2004), pp. 353–366.

[6] Drougas, Y., Repantis, T. and Kalogeraki, V.: Load balancing techniques for distributed stream processing applications in overlay environments, in *Proc. IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, 2006, pp. 1–8.

[7] Gabaix, X.: Zipf's Law for Cities: An Explanation, *The Quarterly Journal of Economics*, Vol. 114, No. 3(1999), pp. 739–767.

[8] Gilbert, S. and Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, *SIGACT News*, Vol. 33, No. 2(2002), pp. 51–59.

[9] Harvey, N. J., Jones, M. B., Saroiu, S., Theimer, M. and Wolman, A.: SkipNet: A Scalable Overlay Network with Practical Locality Properties, in *Proc. USENIX Symposium on Internet Technologies and Systems*, Vol. 274, 2003.

[10] Hefeeda, M. and Saleh, O.: Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems, *IEEE/ACM Transactions on Networking*, Vol. 16, No. 6(2008), pp. 1447–1460.

[11] Hsiao, H.-C., Liao, H., Chen, S.-T. and Huang, K.-C.: Load Balance with Imperfect Information

in Structured Peer-to-Peer Systems, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 4(2011), pp. 634–649.
[12] Konstantinou, I., Tsoumakos, D. and Koziris, N.: Fast and Cost-Effective Online Load-Balancing in Distributed Range-Queriable Systems, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, No. 8(2011), pp. 1350–1364.
[13] Matsuura, S., Fujikawa, K. and Sunahara, H.: Mill: A geographical location oriented overlay network managing data of ubiquitous sensors, *IEICE transactions on communications*, Vol. 90, No. 10(2007), pp. 2720–2728.
[14] Maymounkov, P. and Mazieres, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, *Peer-to-Peer Systems*, Druschel, P., Kaashoek, F. and Rowstron, A.(eds.), Lecture Notes in Computer Science, Vol. 2429, Springer Berlin Heidelberg, 2002, pp. 53–65.
[15] Parzen, E.: On estimation of a probability density function and mode, *The annals of mathematical statistics*, Vol. 33, No. 3(1962), pp. 1065–1076.
[16] Pitoura, T., Ntarmos, N. and Triantafillou, P.: Saturn: Range Queries, Load Balancing and Fault Tolerance in DHT Data Systems, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 24, No. 7(2012), pp. 1313–1327.
[17] Rowstron, A. and Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems, *Middleware 2001*, Guerraoui, R.(ed.), Lecture Notes in Computer Science, Vol. 2218, Springer Berlin Heidelberg, 2001, pp. 329–350.
[18] Shen, H. and Xu, C.-Z.: Locality-Aware and Churn-Resilient Load-Balancing Algorithms in Structured Peer-to-Peer Networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 6(2007), pp. 849–862.
[19] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. and Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications, in *Proc. ACM SIGCOMM Annual Conference*, SIGCOMM '01, New York, NY, USA, ACM, 2001, pp. 149–160.
[20] Zhu, Y. and Hu, Y.: Efficient, proximity-aware load balancing for DHT-based P2P systems, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 4(2005), pp. 349–361.

Kimihiro Mizutani

is a researcher in NTT Network Innovation Labs. He received the M.S. degree in information system from Nara Institute Science and Technology in 2010. His research interests include future Internet architectures. He received the best student paper from International Conference on Communication Systems and Application(ICCSA) in 2010. He also received research awards from IPSJ and IEICE in 2010 and 2013, respectively. He is a member of IEEE Communication Society and IEICE.

Takeru Inoue

is a senior researcher in NTT Laboratories. He was an ERATO researcher at the Japan science and technology agency from 2011 through 2013. His research interests range widely and cover the design and control of network systems. He received the best paper award from Asia-Pacific conference on communications in 2005, and also research awards of the IEICE Information Network Group in 2002, 2005, and 2012. He received the B.E., M.E., and Ph.D. degrees from Kyoto University, Kyoto, Japan, in 1998, 2000, and 2006, respectively. He is a member of IEEE and IEICE.

Toru Mano

is a researcher in NTT Network Innovation Labs. He received the B.E. and M.E degrees in Information Science and Technology from the University of Tokyo in 2009 and 2011, respectively. His research interests include network architectures and network optimization.

Osamu Akashi

is a senior research scientist in NTT Network Innovation Laboratories. He received the M.S. degree in information science and the Ph.D. degree in mathematical and computing sciences from Tokyo Institute of Technology, in 1989 and 2001, respectively. His research interests include distributed systems, multi-agent systems, and network architectures.

Satoshi Matsuura

He received the M.E. and Ph.D. degrees in information sciences from Nara Institute Science and Technology in 2005 and 2008, respectively. Currently, he is an associate professor in Tokyo Institute of Technology. His research interests include overlay networks and sensor networks.

Kazutoshi Fujikawa

has been an Associate Professor in the Graduate School of Information Science, Nara Institute of Science and Technology since 2002. He was a visiting researcher of the Multimedia Communications Laboratory at Boston University from March 1996 through January 1997. He has been engaged in the project of bioIT related R&D called "BioGrid Project," which consists of several institutes in the Kansai area of Japan. His research interests cover distributed systems and multimedia systems. Now he is very interested in Grid systems for scientific simulations. He received the M.E. and Ph.D. degrees in information computer sciences from Osaka University in 1990 and 1993, respectively. He is a member of IEICE, IEEE and ACM.