Comparison of Load Balancing Algorithms for Structured Peer-to-Peer Systems

Simon Rieche, Leo Petrak, Klaus Wehrle
{Simon.Rieche, Leo.Petrak, Klaus.Wehrle}@uni-tuebingen.de

Protocol-Engineering and Distributed Systems
Wilhelm-Schickard-Institute for Computer Science
University of Tübingen, 72076 Tübingen

Abstract: Among other things, Peer-to-Peer (P2P) systems are very useful for managing large amounts of widely distributed data. Distributed Hash Tables (DHT) offer a highly scalable and self-organizing approach for efficient and persistent distribution and retrieval of data. However the scalability and performance of DHTs is strongly based on an equal distribution of data across participating nodes. Because this concept is based on hash functions, one assumes that the content is distributed nearly evenly across all DHT-nodes. Nonetheless, most DHTs show difficulties in load balancing as we will point out in this paper. To ensure the major advantages of DHTs – namely scalability, flexibility and resilience – we discuss three approaches of load balancing and compare them corresponding to simulation results.

1 Introduction

Distributed Hash Tables are more and more used in widely distributed applications [BKK+03, SW04]. Their efficient, scalable and self-organizing algorithms for data retrieval and management offer crucial advantages compared to unstructured approaches. However the underlying assumption is a roughly equal data distribution among the cooperating peers of a DHT. If there is a significant difference in the load of nodes in terms of data managed by each peer, the effort for distributed self-organization of such systems may increase dramatically. Therefore, appropriate mechanisms for load balancing are highly recommended to keep the complexity of DHT management in the range of $O(logN)$ or even less.

2 Algorithms to Balance Load in DHTs

To solve the problem of load balancing, several techniques have been developed to ensure an equal data distribution across DHT nodes: *Virtual Servers* [RLS+03], *Power of Two Choices* [BCM03] and the *Heat Dispersion Algorithm* [Ri03]. We present these algorithms briefly and compare their efficiency based on simulation results.

2.1 The Concept of Virtual Servers

The virtual server approach [RLS+03] is based on the idea of managing multiple partitions of a DHT's address space in one node. Thereby, one physical node may act as several independent logical nodes. Each virtual server will be considered by the underlying DHT as an independent node. Within a Chord system, one virtual server is responsible for an

interval of the address space. Thus, the corresponding physical node may be responsible for several different, independent, and incoherent intervals.

The basic advantage of this approach is the displacement simplicity of virtual servers among arbitrary nodes. This operation is similar to the standard join or leave procedure of a DHT. Thereby, content will be distributed as ranges of hash values across plenty of nodes. Every participating node manages virtual servers and has the knowledge of all its neighbors. E.g., in case of using Chord this relates to all fingers within the routing table.

2.2 Power of Two Choices

The algorithm *Power of Two Choices* [BCM03] relies on the concept of multiple hash functions. These functions are used to map data into the address space of a DHT. For the process of inserting and retrieving, the results of all hash functions are calculated. In case of inserting a new document, all respective hash values are computed and the corresponding nodes are retrieved. Finally, the document is stored on the retrieved node with the lowest load in terms of stored data.

2.3 Load Balancing Similar to Heat Dispersion

[Ri03] introduces a new load balancing algorithm for Distributed Hash Tables. Content is moved among peers similar to the process of heat expansion [Ro03]. Usually, a material warmer than its environment emits heat to its surrounding. This process continues until a balanced distribution of the heat is reached in the entire system. To deploy a similar algorithm for balancing load among peers in a DHT, [Ri03] proposes a very simple approach which needs only three rules to balance the load among the peers.

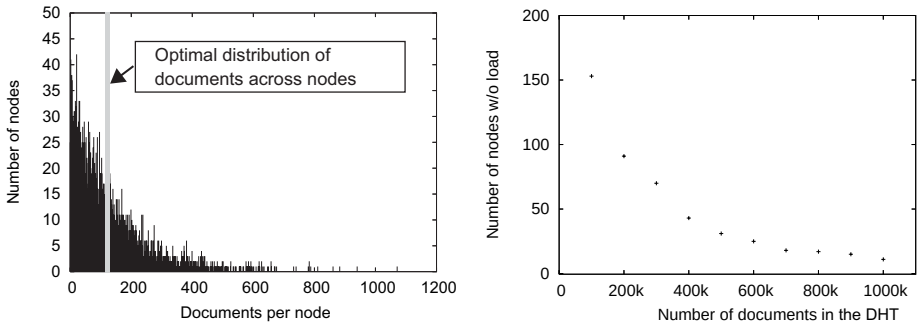
But nodes in a Distributed Hash Table can not simply move arbitrary documents to other nodes, e.g. their neighbors, because this would result in an inconsistent and inefficient search. This reduces the performance and advantages of a DHT. Therefore the algorithm moves only complete intervals or contiguous parts of them between the nodes in the DHT. Although the Chord system has been modified the efficient Chord routing algorithms remain unchanged.

In the following we give a brief summary of the algorithm on the basis of the DHT system Chord [SMK+01]. For further details, please refer to [Ri03].

First of all, any fixed positive number f is chosen, which indicates how many nodes have to act within one interval at least. On the one hand this number f helps to balance load, but also to make Chord more fault tolerant.

The first node takes a random position in the Chord ring and a new node is assigned to any existing node in the system. This „receiving" node announces each joining node to all other nodes responsible for the same interval. Following, part of documents located within this interval are copied to this new node. Then the original methods to insert a node in Chord are performed.

Now the nodes, located within the same interval, can balance the load between nodes of other intervals according to three various methods. (1) if $2f$ different nodes are assigned to the same interval, and each node stores significantly more documents than average, this interval is divided. (2) intervals with more than f but less than $2f$ nodes can release some nodes to other intervals. (3) interval borders may be shifted between neighbors.



(a) Frequency distribution of DHT nodes storing a certain number of documents

(b) Number of nodes without storing any document

Figure 1: Distribution of data among a Chord DHT without load balancing mechanisms.

3 Comparison of Load Balancing Approaches

3.1 Simulation Scenarios

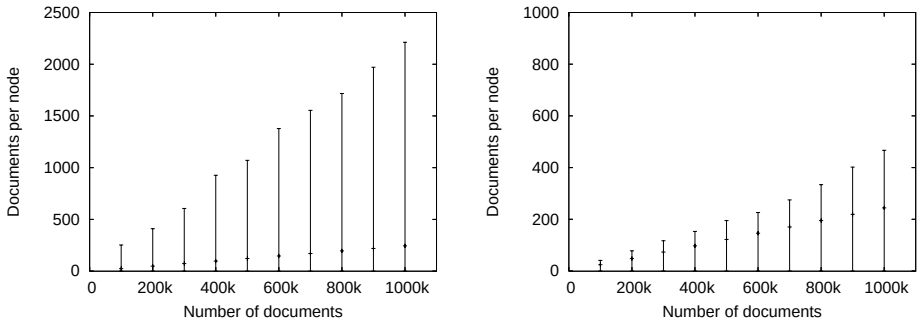
To analyze load balancing in a Distributed Hash Table, a complete Chord ring simulator was developed to investigate and to compare the load balancing algorithms mentioned previously. The focus was put on the distribution of documents among the nodes.

In each scenario a Chord DHT with $4{,}096$ nodes was simulated and multiple simulations runs per scenario have been made to confirm the results. The size of $4{,}096$ nodes was chosen in order to compare our measurements to related work. The number of nodes is equal to the number of virtual servers and equal to half of the nodes used by „Power of Two Choices". Although the number of nodes were halved, the simulation shows that the results are comparable with the simulations presented in [BCM03]. The total number of documents to be stored ranged from $100{,}000$ to $1{,}000{,}000$. Thereby the keys for the data and nodes were generated randomly. For this purpose, the Chord ring's address space had a size of $m = 22$ bits. Consequently, $2^{22} = 4{,}194{,}304$ documents and/or nodes could be stored and managed in the ring. In our simulation, the load of a node is defined by the number of documents stored by itself.

3.2 Simulation Results

Figure 1 proofs that the assumption of getting an equal distribution of data among peers by using a hash function is not defensible. E.g., Fig. 1(a) shows how many nodes store a certain number of documents. It is clearly observable that there is no equal distribution of documents among the nodes. For a better comparison the grey line indicates the optimal number, which is approximately 122 documents per node. Additionally, Fig. 1(b) plots the number of nodes without any document.

Fig. 2(a) shows the distribution of documents in Chord without load balancing. Between 10^5 and 10^6 documents were distributed across $4{,}096$ nodes. The upper value indicates the maximum number of documents per node, the lower value the minimum number. The optimal number of documents per node is indicated by the marker in the middle.



(a) Chord without load balancing

(b) Chord with „Power of Two Choices”

(c) Chord with virtual server

(d) Chord with heat dispersion algorithm

Figure 2: Simulation Results comparing Different Approaches for Load Balancing in Chord.

Even for a large number of documents in the DHT, there are some nodes without managing any documents and consequently without any load. Some nodes have a load up to ten times above the optimum. Fig. 2(b) shows that „Power of Two Choices” works way more efficient than the original Chord without load balancing. However there are still obvious differences in the load of the nodes. Some are still without any document.

Applying the concept of Virtual Servers (cf. Fig. 2(c)) results in a more efficient load balancing. Nevertheless, this is coupled with a much higher work load for each node, because it has to manage many virtual servers. Additionally, the data of all virtual servers of one physical node has to be stored in the memory of the managing node.

Fig. 2(d) shows that the best load balancing results are achieved by using the heat dispersion algorithm. Each node manages a certain amount of data and load fluctuations are pretty small. Documents are only moved from neighbor to neighbor. Using virtual severs however results in copying the data of a whole virtual server. As a result the copied load is always balanced. In addition more node management is necessary.

4 Conclusion

In this contribution, we argued that the scalability and efficiency of structured Peer-to-Peer-systems crucially depends on a balanced distribution of data between DHT peers. As

shown in our analysis and confirmed by other research [SMK+01, BCM03, RLS+03], the approach to ensure equally distributed value spaces simply by using hash functions is not feasible. To achieve a continuous balance of data and moreover to ensure the scalability and efficiency in structured Peer-to-Peer systems, load balancing mechanisms have to be applied. In this paper we discussed three approaches for load balancing in Distributed Hash Tables, and compared their efficiency.

The algorithm „Power of Two Choices" uses two or more hash functions to map data in a DHT. This results in additional messages sent among the nodes, when nodes are inserting or retrieving data. By the additional administration of pointers, again each node has to manage additional data, and its load continues to rise. With large quantities of documents this causes heavy network traffic and high management cost in the nodes.

The virtual server approach is based on the idea of managing multiple partitions of a DHT's address space on one node. The main point of criticism is the large effort that a node has to provide when using this algorithm. In order to keep all its routing information – stored in the finger tables of the virtual servers – consistent, too many messages need to be sent. Using this approach almost 50% of the entire content will be replaced [RLS+03]. In large Peer-to-Peer systems the replacement of such a large amount of data results in an immense network traffic.

On the other hand, load balancing according to heat dispersion is based on the idea, that a node emits content to its surrounding nodes. The results are very promising as they show a much better balancing of data among peers. Furthermore, this algorithm causes less network traffic and ensures a better load distribution than the two other approaches.

Acknowledgment

The first author wants to thank the supervisors of his diploma thesis, namely Prof. Alexander Reinefeld (ZIB and HU Berlin) and Prof. Jochen H. Schiller (FU Berlin) as well as Florian Schintke and Thorsten Schütt (ZIB) for their help and advise.

References

[BCM03] Byers, J., Considine, J., und Mitzenmacher, M.: Simple Load Balancing for DHTs. In: *Proc. of 2^{nd} Intern. Workshop on P2P Systems*. Berkeley. February 2003.

[BKK+03] Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., und Stoica, I.: Looking Up Data in P2P Systems. *Communications of the ACM*. 46(2):43–48. 2003.

[Ri03] Rieche, S. Load Balancing in Peer-to-Peer Systems. Diploma thesis, Freie Universität Berlin, Germany. November 2003. (in german).

[RLS+03] Rao, A., Lakshminarayanan, K., Surana, S., a. o.: Load Balancing in Structured P2P Systems. In: *Proc. of 2^{nd} Intern. Workshop on P2P Systems*. Berkeley. February 2003.

[Ro03] Robertazzi, T. G.: Ten Reasons to Use Divisible Load Theory. *IEEE Computer Society: Computer magazine*. 36(5):63–68. May 2003.

[SMK+01] Stoica, I., Morris, R., Karger, D., a. o.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: *Proc. of ACM Sigcomm, San Diego, CA*. 2001.

[SW04] Steinmetz, R. und Wehrle, K.: Peer-to-peer-networking & -computing. *Informatik-Spektrum*. 27(1):51 – 54. Februar 2004. Springer, Heidelberg (in german).