2012 International Conference on Information and Computer Networks (ICICN 2012)
IPCSIT vol. 27 (2012) © (2012) IACSIT Press, Singapore

Efficient Data Load Balancing for Constant Degree P2P Systems

YANG XiaoXiao [1+], WANG XiaoHai [2], Xiao HanHua[3]

[1] Postgraduate Department, Navy Submarine Academy, Qingdao, China

[2, 3] Campaign Command Department, Navy Submarine Academy, Qingdao, China

Abstract. Constant degree DHTs are turning into the P2P domain's new rising star and promising hotspot in recent years. Load balancing is the importance guarantee that P2P system will perform better by against several specific nodes storing too much and becoming system's bottleneck. However, all known methods either can't fit dynamically for different load distributions, or bring too much overhead, or can't fulfill the requirement for balanced topology of constant degree P2P. To address this problem, we presents an efficient load balancing algorithm Routing Information Statistic for constant degree P2P systems. Without adding any data structures, RIS judges whether a new join data object should be redirected and which node is the most suitable redirect storage using the statistic of routing information. Experimental results show, with low redirect overhead, RIS keeps the constant degree P2P system load-balanced.

Keywords: constant degree topology; P2P; RIS; load balance

1. Introduction

In structural P2P systems, DHT maps each data object to a definite node according to its key, and divides the whole key space into subspaces in which all objects (or their indexes) should be stored and maintained by one node. However, the consistent hashing can't assure all subspaces same in size and all nodes storage equally number of objects relatively [1]. What's more, in order to support range queries and many other applications, P2P systems usually replace the consistent hashing with other mapping functions that could maintain a better "data locality" property. However, these pre-defined functions will bind the key space to nodes "innately", and then bring fundamental contradiction with load balancing [2, 3]. Therefore, a ideal balancing algorithm should be able to use as little as possible redirect overhead to assure all nodes' data load lower than a threshold or be same as far as possible under different data distribution.

A constant degree P2P system is a kind of structured P2P system with O(log$N$) routing efficiency and a constant number of neighbors. Because of the invariant node degree, a constant degree P2P system enjoys superiority not only in high routing efficiency, but also in less updating, controlling messages, which decreases network overhead and improves system performance [4]. However, most research on constant degree P2P are focus on DHT's construction and maintenance, while leaving the topologies' characteristics and load balancing been not well studied.

Combining constant degree topology's characteristics and bit-shift routing schema, we present an efficient load balancing algorithm called Routing Information Statistic (RIS) for constant degree P2P systems. Without adding any data structure, RIS judges whether a new join data object should be redirected and which node is the most suitable redirect storage using the statistic of routing information; aiming at redirecting hot spot data objects evenly, RIS provides an method to construct new routing path. Experimental results show, with low redirect overhead, RIS keeps the constant degree P2P system load-balanced under different load distributions.

[+] Corresponding author.Tel.: +86-532-51858140 ; fax: +86-532-51858140
E-mail address: xiaoxiao19811030@yahoo.com.cn

2. Basic Definition

Kautz digraph: Given an positive integer d and $n \geq 1$, Kautz[5, 6] digraph K(d, n) is a directed graph with diameter n and all nodes' in-degree and out-degree equal d, each node's ID is a d-based Kautz string $x=x_1x_2\ldots x_n$, where $x_i \in \{0,1,...,d\}$ ($1 \leq i \leq n$), $x_i \neq x_{i+1}$ ($1 \leq i \leq n-1$, and links to nodes $x_2\ldots x_n y$, where $y \in \{0,1, ...,d\}$, $y \neq x_n$. Fig.1 shows Kautz digraph K(2,3).

Kautz order: For any two different nodes $x=x_1x_2\ldots x_n$ and $y=y_1y_2\ldots y_n$, in K(d, n), we can define the total order < as following:

$$x < y \Leftrightarrow \exists j \in [1,n]\big(\forall i \in [1,j)\big)\big((x_i = y_i) \wedge (x_j < y_j)\big)$$

We further note node x' sequence number based on 0 as *seq*(x), e.g., in K(2,3), *seq*(010)=0, *seq*(012)=1.

Load ratio: The ratio μ between a node's actual load and its capability.

We use light load ratio l and heavy load ratio h, to describe nodes' load qualitatively, then all nodes can divided into 3 classes: *light* loaded ($\mu \leq l$), *middle* loaded ($l \leq \mu \leq h$), and *heavy* loaded ($\mu \geq h$). We also define load ratio of whole system *system.μ* as the ratio between sum of all nodes' load and the their capabilities.

Surplus space: For node q, surplus space *q.space* is the load value needed to reach heavy loaded. Unlike a light loaded node which can accept new data object freely, a middle or heavy loaded node may redirect o to other nodes if possible.

Wild card set: For a Kautz String x containing wild card "*", such as 20*0, where "*" represents a bit that can be any legal character, wild card set WCS(x) is a set contains all nodes whose ID match x.

In K(2,4), WCS(20*0)={2010, 2020}. |WCS(x)| denotes the order of this set. Then in K(d,n) node x's out-degree neighbors are WCS($x_2\ldots x_n$*), and |WCS($x_2\ldots x_n$*)|=d.

Routing path set: For a source node x and destination node (or Kautz String) y, a set RT(xy) contains all nodes in the long path routing path including, that is

$$\mathrm{RT}(xy) = \{x,\ x_2 ... x_n y_1\ , x_3 ... x_n y_1 y_2\ , ...,\ x_n y_1 y_2 ... y_{n-1},\ y\}$$

In K(2,3), RT (212101) = {212, 121, 210, 101}. |RT(xy)| denotes the order of this set. When y's length is longer than the actual destination ID, its first character can be underlined to indicate the destination. For example, in K(2,3) , RT (212$\underline{1}$01) = RT (212$\underline{1}$012).

Routing surplus space (RTSpace): The load value needed for all nodes in routing path set to reach heavy loaded. That is RTSpace(sp) = $\sum$(q.space), $q \in$ RT(sp).

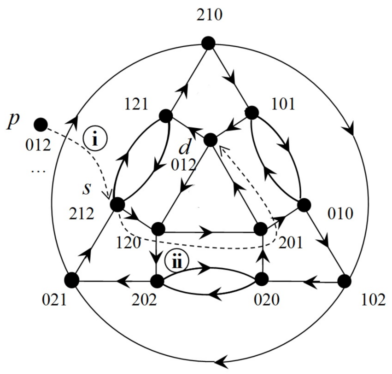


Fig. 1: Typical data joining process in K(2, 3)

3. RIS algorithm

We introduce RIS on FissionE, which can extended to other Constant degree P2P systems[6] easily.

Considering the new data object o's sub join process *FissionERouting*[8], as the *i, ii* step shown in Fig.1:

i. *Contacts a node s in the system, and obtains a hash Kautz string p as its ID;*

ii. *According to the long path routing algorithm, o initiates a message routing that from s to p. passing through |s| nodes, the INSERT message stops at a unique node which id is p's prefix.*

iii. *Joining node d directly.*

Therefore, the INSERT message must pass all nodes in RT(*sp*) in turn. Because we can regard RT(*sp*) as nodes distributed in the completed node set evenly for its dispersion property (Theorem 1), the statistical information about these nodes can guide data's joining to achieve load balance effectively: their load information can make destination node determine accurately whether *o* need to redirect and provide more redirect candidates than multi-hashing technique.

When INSERT message accesses nodes' load information, it also records node's address that may be used to redirect storage. Comparing these information, *o* will join *d* or other node in RT(*sp*). The principle is to decrease redirection number and assure all nodes' data load at a lower level. There are four specific cases, node *d* will save the redirect pointer except the first case.

i. *d is light loaded: Join the target node d directly, i.e., FissionEJoin(o, d);*
ii. *d is middle or heavy loaded, and there are light loaded nodes in RT(sp): Join the minimum load rate node in RT(sp).*
iii. *d is middle or heavy loaded, and there are no light load nodes in RT(sp), but not all of them are heavy loaded: join the node with the maximum surplus space in RT(sp).*
iv. *all nodes in RT(sp) are heavy loaded: join the minimum load rate node in RT(sp).*

RIS performs depend on $system.\mu$: when $system.\mu$ is low, RIS accepts data objects deterministically to reduce the cost of redirection while ensuring that all nodes light loaded; as $system.\mu$ increases, RIS will redirect load to other light loaded nodes in the Routing Path Set, maintaining destination node to be light loaded; as $system.\mu$ increases until all nodes' in routing path set are not light loaded. RIS will select node with the largest surplus space, thus the joining of data will affect nodes load ratio a little; when $system.\mu$ continues to rise and all nodes in the routing path set are heavy loaded, the algorithm's goal is to reduce the difference in nodes' load rate and assure no bottleneck exist. Table I shows the algorithm: the gate node initialize the INSERT message first, while new routing path set will be constructed if necessary. When passing each node, INSERT updates information with which destination node decides the storage position of the data object.

Once the destination node *d* is heavy loaded or overload, there will be O(log*N*) (size of RT(*sp*)) nodes can be potential redirect choice, therefore load balancing can be achieved under different data distribution with low overhead. Exception is the hot spot data distribution, if lots of data with key *p* join the system through node *s*, because the routing path set is always RT(*sp*), all nodes in this set will suffer a high load. Thus, an algorithm *GenerateRT* that can construct different route path sets is badly needed:

The exact value of RTSpace can only calculated at real time, *GenerateRT* takes $s.capability\cdot|s|(h-s.\mu)\cdot\delta$ as its estimated value, where $|s|$ is *s*'s ID length, δ is a factor less than 1, which stands for the value will be slightly smaller than the union of surplus spaces of $|s|$ nodes with load rate $s.\mu$.

Table 1 Data Joining Algorithm

```
Initialization:
        // initialization of information in INSERT   message
MinRatio = ∞, MaxSpace = -∞,
existLightLoadNode = false, allHeavyLoad = true
--------------------------------------------------------------
if(data >s·(h-s.μ)·δ)     GenerateRT(s, p)
for each node q in RT(sp)
    if(q.μ≤h)   allHeavyLoad = false
    if(q.μ≤l)   existLightLoadNode = true
    if(q.μ≤ MinRatio)    MinRatio=q.μ, MinRatioPeer = q
    if(q.space> MaxSpace) && (!existLightLoadNode)
        MaxSpace = q.space; MaxSpaceNode = q
//When reach node d
if(d.μ≤l)   FissionEJoin(o, d)
else if (existLightLoadNode)  FissionEJoin(o,MinRatioNode)
else if (!allHeavyLoad) && (!existLightLoadNode)
        FissionEJoin(o, MaxSpaceNode)
else   FissionEJoin(o, MinRatioNode)
```

Table 2 Generatert Algorithm

```
Procedure GenerateRT(gateNode s, KautzString p)
    ω = 0.8;  RT'= newNodeSet = null; s' = s
  //there are ω·|RT(sp)| new nodes in RT' at least
--------------------------------------------------------------
while(|newNode|<ω·|RT(sp)|)
    for each character x∈{0, 1, ..., d}
    s' = s'·x
    //generate new string by inserting new character into s and p
    if (s'·p is a Kautz string)
        RT' = RT(s·x·p)
        for each node q in RT
            if(!q∈RT(sp)) newNodeSet.add(q)
        if(|newNodeSet| ≥ ω·|RT(sp)|)   return RT'
```

When the number of data objects joining from *s* beyonds RTSpace(*sp*), it can be determined that load rate of all nodes in RTSpace(*sp*) will be close to or already be heavy loaded, then the algorithm will construct another routing path set RT', table II shows the algorithm.

GenerateRT can construct a new routing path set RT' which contains at least $\omega\cdot|\mathrm{RT}(sp)|$ new nodes, thus redirecting loads to RT' can effectively avoid nodes' loading rate getting too high in RT(*sp*). For example in K(2, 4), RT(01201210) = {0120, 1201, 2012, 0121, 1210}, by adding character '2', we have new routing path RT'(012021210) = {0120, 1202, 2021, 0212, 2121, 1210}.

4. Analysis

4.1. Characteristics of the routing path set

Theorem 1 In constant degree P2P systems, nodes in routing path set distribute dispersive.

Lemma 1 Suppose $x=x_1x_2\ldots x_n$ and $y=y_1y_2\ldots y_m$ are Kautz string with same degree, then *seq*(*x*), *seq*(*y*) and Kautz *seq*(*yx*) following the relationship below:

$$seq(yx)=\begin{cases} seq(y)\cdot d^n+seq(x) & (x_1<y_m)\\ seq(y)\cdot d^n+seq(x)-d^{n-1} & (x_1>y_m)\end{cases}$$

Proof: in K(*d*, *m*), note Less (*y*) $=\{u{:}u<y,\ u\in \mathrm{K}(d,m)\}$, then $|\mathrm{Less}(y)| = seq(y)$ obviously. By the definition of order, for any $u\in\mathrm{Less}(y)$ and $v\in\mathrm{K}(d, n)$, we have $uv<yx$. Thus the $seq(y)\cdot d^n$ Kautz strings in K(*d*, *m*+*n*) with prefix in Less(*y*) must have a smaller sequence number than *yx*.

If $x_1<y_m$, then for any $v\in\mathrm{K}(d, n)$ and $v<x$, i.e., $v\in\mathrm{Less}(x)$, we have $yv<yx$, thus *yx*'s sequence number is *seq*(*x*)th in WCS(y^{*m}); if $x_1>y_m$, because there are d^{n-1} Kautz Strings with prefix y^m that can't connect with *y*, thus *yx*'s sequence number is seq(*x*)-d^{n-1} in WCS(y^{*m})

seq(*yx*) is the sum of former two values, so lemma holds.

For any node *x* and its successor $x_2\ldots x_ny_1$ in routing path set, their sequence numbers are x_1d^{n-1} + $seq(x_2\ldots x_n)$ (or x_1d^{n-1} + sep$(x_2\ldots x_n)$ – d^{n-2}, and $seq(x_2\ldots x_n)d + y_1$ (or $seq(x_2\ldots x_n)d + y_1$ - 1), there is no regularity or continuity between them, thus theorem 1 holds. □

4.2. Data balance analysis

Dispersion of routing path node set determines data objects can be delivered to different position in whole topology, which benefits balancing. In contrast, other structured P2P systems does not have this character for their Un-bit-shift routing schema. For example in Chord, distances between node and its neighbours are 20, 21…, which aren't evenly distributed in ID space certainly. Furthermore, Chord chooses the neighbour with smaller sequence number difference from destination at each routing step, thus the farther away from destination, the more scattered and fewer nodes distribute in Chord routing path set, and it is always hard to construct routing node sets containing lots of different nodes.

Balancing overhead usually includes a variety of additional data structures' maintenance, and exchanging load and update information regularly. Because RIS finishes statistics during the inherent routing process, no additional data structures and exchange information are needed. Its joining overhead is about 1/*n* of the *n*-choice [7] technique for only one route process is needed.

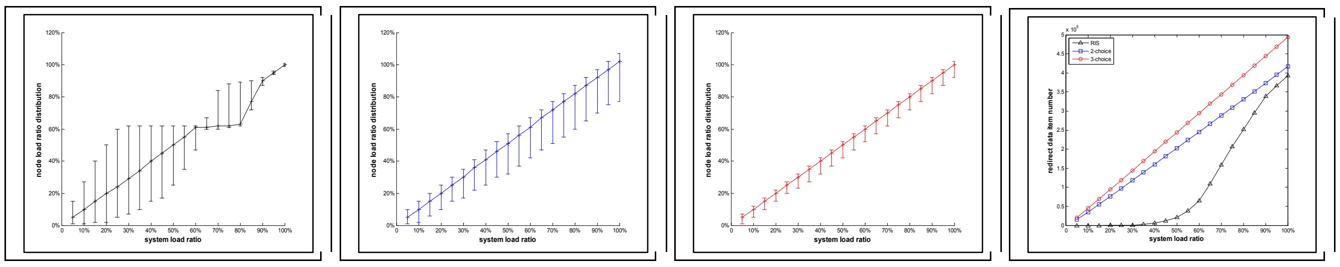
Redirection may result in extra overhead in data maintenance and query, while RIS only redirection data until the destination node is middle loaded, so small load ratio difference is allowed when $system.\mu \le l$, which decrease the number of redirect data objects. What's more, the number of nodes in routing path set is far more than the *n*-choice technique, thus a better balance effect can be assured which is shown in section VI's experiments.

5. Experiments

We implement experiments based on FissionE simulator: data objects belongs to interval [0, 1], and compare the balance effect and the number of redirect data objects under random distribution with multi-hashing technique including 2-choice and 3-choice ($l = 60\%$, $h = 90\%$).

Fig.2 a~c shows nodes' load ratio distribution (y-axis) against system's load ratio (x-axis) when using RIS, 2-choice, and 3-choice algorithm. Curves represent the median number of nodes' load ratio under different system load ratio, and each error bar's top and bottom represent 99% and 1% load rate percentile. It can be seen, nodes' load distribution are quiet depend on the algorithms' mechanism: two muti-hash algorithms don't change redirect strategy when system load rate changes, nodes' load rate are almost increase linearly. For RIS algorithm, because no light loaded nodes need redirect, there are always light loaded nodes with rate l when system load rate is between 40% and 60%. As $system.\mu$ increases, some nodes load rate reach h, but the RIS can guarantee the majority of the load rate is still close to l and no nodes exceeds h, thus the curve have a gentle section when system load rate is between 60% and 80%. After the system is overloaded, all nodes' load rate increases synchronously, but the load rate difference is less than 3-choice, proving that RIS reaches better balance than 3-choice with only 1/3 joining overhead of latter.

Fig. 2.d compares the number of redirect data objects. We observe that, the number of redirect data objects increases linearly under the muti-hash algorithms when system load rate increasing, and 3-choice redirects more objects than 2-choice because its better balance effect is based on more redirect cost. However, RIS redirects a few at the beginning, and increases linearly only after system's load rate exceeds l while still less than former two muti-hashing algorithms. Thus, RIS ensure all light loaded nodes redirect a small amount of objects when system. $\mu \leqslant$l, and its redirect cost is less when system. $\mu \geqslant$h.



(a) RIS (b) 2 choice (c) 3-choice (d) redirect overhead

Fig. 2: Basic performance.

6. Conclusion

Combining constant degree topology's bit-shift routing schema, we presents an efficient load balancing algorithm RIS. Without adding any data structures, RIS achieves balance based on the statistic of routing information and redirection technology. Experimental results show, with low redirect overhead, RIS keeps the constant degree P2P system load-balanced.

7. References

[1] X. Wang and D. Loguinov, "Load-Balancing Performance of Consistent Hashing: Asymptotic Analysis of Random Node Join," IEEE/ACM Transactions on Networking, vol. 15, no. 4, August 2007.

[2] Abhishek Gupta, Divyakant Agrawal, and Amr El Abbadi. Approximate Range Selection Queries in Node-to-Node Systems[C]. CIDR, Asilomar, California, USA, January 2003.

[3] Artur Andrzejak and Zhichen Xu. Scalable, efficient range queries for grid information services[C]. In Proceedings of the 2nd IEEE P2P, 2002,33-40.

[4] S. Ratnasamy, S. Shenker, and I. Stoica. Routing Algorithms for DHTs: Some Open Questions. Proc. of First International Workshop on Node-to-Node Systems (IPTPS'02), Cambridge, USA, 2002

[5] D. Li, X. Lu, and J. Wu. FissionE: A scalable constant degree and low congestion dht scheme based on kautz graphs. In Proc. IEEE INFOCOM, pages 1677–1688, Miami,Florida,USA, March 2005.

[6] D. Guo, J. Wu, H. Chen, X. Luo. Moore: An Extendable Node-to-Node Network Based on Incomplete Kautz Digraph with Constant Degree. Proc. of IEEE INFOCOM 2007.

[7] Byers J, Considine J, Mitzenmacher M. Simple load balancing for distributed hash tables. In: Kaashoek MF, Stoica I, eds. LNCS. Berlin: Springer-Verlag, 2003. 80-87.

[8] Li DS. Research on peer-to-peer resource location in large-scale distributed systems [Ph.D. Thesis]. Changsha:

National University of Defense Technology, 2005 (in Chinese with English abstract).

[9] Bryce Wilcox-O'Hearn. Experiences deploying a large-scale emergent network. In Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, March 2002.