doi:10.3969/j.issn.1000-436x.2013.12.007

DHT 网络中一种基于虚拟服务器拆分的负载平衡算法

杨磊，李仁发，柳石，陈志兴，李肯立
（湖南大学 计算机与通信学院，湖南 长沙 410082）

摘 要：分析比较了目前 DHT 网络中基于虚拟服务器（virtual server）的负载平衡算法，根据节点间间距的分布概率，建立了基于虚拟服务器的 DHT 网络负载分布数学模型，详细分析和计算了 DHT 网络中单虚拟服务器问题（SVSP）发生的概率，提出了一种基于虚拟服务器拆分的负载平衡算法（VSSLBA），仿真实验验证了理论推导的正确性与算法方案的有效性。

关键词：DHT 网络；虚拟服务器；负载平衡；SVSP；虚拟服务器拆分算法
中图分类号：TP393.4 **文献标识码**：A **文章编号**：1000-436X(2013)12-0060-11

VS-split load balancing algorithm in DHT-based P2P systems

YANG Lei, LI Ren-fa, LIU Shi, CHEN Zhi-xing, LI Ken-li
(College of Computer and Communication, Hunan University, Changsha 410082, China)

Abstract: The method using virtual servers for balancing the load in DHT-Based P2P systems was studied. The load distribution in DHT-based overlay network using virtual servers were modeled, the occurrence probability of single virtual server problem (SVSP) was analyzed and computed in details, and a novel VS-split load balancing algorithm (VSSLBA) was proposed to deal with the SVSP by splitting virtual server into small ones. Simulations verify the correctness of probability analysis and the performance of VSSLBA.

Key words: DHT-based network; virtual server; load balance; single virtual server problem; VS-split load balancing algorithm

1 引言

在 DHT(distributed hash table)网络中，基于虚拟服务器(VS, virtual server)的负载平衡算法因具有灵活的负载转移能力和容易实现等优点而得到广泛应用。基于 VS 负载平衡算法的基本思想是在具有不同能力的物理节点上运行多个 VS 实例（通常为 5 个）,并根据节点的负载状态利用 VS 的迁移以实现负载平衡。现有研究主要集中在算法的应用方式、收敛性能以及节点负载信息的有效收集上。在负载平衡过程中，由于节点能力以及单个 VS 上承担的负载存在差异，部分节点上的 VS 不断地被迁移出去。当这些节点上的 VS 只剩下一个且其承担的负载超过了节点的能力阈值时，将不能通过迁移 VS 实现负载减少，本文把这种现象称之为单虚拟服务器问题（SVSP, single virtual server problem）。目前，有关 SVSP 的研究较少，但处于该状态的节点不但不能实现负载平衡，而且还极大地影响存储节点服务的有效应用能力。

2 相关研究

覆盖网络环境下的负载平衡研究最早出现在基于 DHT 算法的 Chord[1]和 CFS[2]存储系统中，它们均提出通过在物理节点上建立多个 VS 并通过 VS 的迁移来实现动态的负载平衡。随后的 DHT 算法和存储系统，如 Pastry[3]、PAST[4]、tapestry[5]也提

收稿日期：2012-11-01；修回日期：2013-10-30
基金项目：国家自然科学基金资助项目（61133005）；湖南省科技计划基金资助项目(2011FJ3122)；湖南大学 “ 青年教师成长计划 ” 资助项目
Foundation Items: The National Natural Science Foundation of China(61133005);The Science and Technology Program Project of Hunan Province(2001FJ3122); Program for the Growth of Young Teachers of Hunan Univercity

出了相应的负载平衡算法，但这些算法都局限于特定的应用环境。

文献[6,7]沿用了虚拟节点的负载转移机制，详细讨论了 VS 迁移的具体方式。文献[7]在文献[6]的基础上，利用 random walk 的信息收集算法，通过维护一张其他节点的负载信息表并周期性地对表中节点的负载进行调整以实现整个系统动态的负载均衡。文献[8]假设负载在整个 DHT 网络 ID 空间均匀分布前提下，提出一种根据物理节点能力为其虚拟服务器分配连续 Overlay network ID 的方法以减少负载移动开销和路由表空间。文献[9]在 DHT 网络上构建了一个 K-ary 树，利用 K-ary 树收集节点负载信息，并在重新分配 VS 和负载迁移时考虑节点相对 landmark 的 d 维距离信息，可以显著减少负载平衡过程的开销。在较新的研究中，文献[10] 在常数度 P2P 系统中提出一种基于 REST(reverse spanning tree)的负载平衡算法，利用 REST 算法可以估算新加入物理节点能够容纳的 VS 个数，并帮助 VS 获得与其能力相适应的 DHT 网络 ID 空间范围以实现系统负载均衡。与大部分需要依赖特殊的 DHT 网络结构或者辅助网络结构收集节点和负载信息进行负载平衡的方法不同，文献[11]采用随机采样的方法借助系统部分节点信息估算节点能力和负载的概率分布并实现 VS 的迁移，可以显著减少算法实现的消息开销并提高系统顽健性能。本文工作主要基于文献[6]和文献[7]展开，因此文中也将其所提出的方法称为经典负载平衡算法。

3　SVSP 问题

3.1　SVSP 问题定义

在 DHT-Base P2P 网络中，基于 VS 的负载平衡算法可以通过在物理节点中初始化多个 VS 实例并通过在节点间进行 VS 的转移来实现负载平衡。但是，虚拟服务器机制并未改变 DHT 网络中采用一致性散列方法的随机性特点，这导致 VS 所管理的 ID 空间依然存在着差异，而这种差异最终会使 DHT 网络上的负载分布不均衡，即网络中依然可能存在着管理较大负载的 VS。

同时，基于 VS 的负载平衡算法并未改变物理节点的异构特性。现有算法的基本思想是希望能在具有不同能力的物理节点上建立不同数目的 VS。但实现这个目标需要以建立所有节点的能力视图为基础，DHT 网络节点的自治性使得单个节点不可能了解整个网络的能力信息。因此，现有算法在节点初始化时均赋以其相同数目的 VS，再通过负载平衡算法实现 VS 在不同节点之间迁移来体现节点异构性，这种方法不仅会带来较大的节点搅动和通信开销，而且难以保证负载重的 VS 一定能迁移到能力强的物理节点上。

综上所述，以下情况将难以避免：能力较弱的物理节点上出现一个或多个负载较重的 VS，往往成为负载平衡过程中 VS 被迁移的对象。当该过程不断发生，其拥有的 VS 个数会持续的减少直到剩下一个 VS 并仍然使节点处于过载状态时，该物理节点将面临两难。一方面，由于节点过载，根据负载平衡算法需要把该 VS 迁移出去；另一方面，由于节点上只剩下一个 VS，如果把该 VS 转移出去，那么节点将不再具有参与 DHT 网络的能力。本文把物理节点在这种状态下所面临的问题定义为单虚拟服务器问题（SVSP, single virtual server problem）。现有算法没有考虑 SVSP 问题，而一旦节点出现 SVSP，将使该节点长期处于过载状态，同时其他空闲节点的资源又得不到有效利用，严重影响了 DHT 网络的性能和效率。

3.2　SVSP 数学建模

本文利用 DHT 网络的统计规律对 SVSP 进行数学建模。在 DHT 网络中，数据对象数量远大于节点数量，一个数据对象落在一个节点上的概率与其 ID 管理空间成正比，节点的 ID 管理空间越大，节点就拥有更多的数据对象。所以，节点负载的概率分布可以等价于 ID 管理空间的概率分布，而后者可以通过节点间距的分布推导而来。

为了计算 SVSP 发生的概率，本文采用 Chord 网络的环形拓扑[12,13]作为基础网络模型，因为其他 DHT 网络拓扑结构如 Pastry、Kademlia[14]等都可以抽象成 Chord 的环形拓扑结构，具有一定的代表性。在 Chord 网络中，某个节点的 ID 管理空间可以定义为其前驱节点到该节点自身 ID 之间的范围，文献[15]已经证明，其分布与节点的间距分布等价，近似满足负指数分布，其概率密度函数（PDF）与累计分布(CDP)可以分别表示为

$$f(l)=n\mathrm{e}^{-nl},F(l)=1-\mathrm{e}^{-nl} \tag{1}$$

其中，Chord 网络中节点 ID 随机选择，n 为在线节点数，l 为节点 ID 管理的空间长度。

定理 1 DHT 网络中具有 k 个 VS 的物理节点所管理的 ID 空间长度的概率密度为 $f_k(l)=\frac{1}{(k-1)!l}\cdot(knl)^k \mathrm{e}^{-knl}$。

证明 由于 Chord 中每个物理节点上运行多个 VS，则每个物理节点所占据的 ID 管理空间可以分析如下：用 l_i 表示某个物理节点上第 i 个 VS 所管理的 ID 空间范围，因此，容纳 k 个 VS 的物理节点的 ID 管理空间可以用集合 $\{l_1,l_2,\cdots,l_k\}$ 表示，其大小为所有 VS 所管理的 ID 空间范围大小之和[16~19]，即 $l=l_1+l_2+\cdots+l_k$。由于各个 VS 的 ID 独立选取，所以物理节点的 ID 管理空间的概率密度（PDF）是所有 VS ID 管理空间的 PDF 卷积 $f(l)=f(l_1)f(l_2)\cdots f(l_k)$，利用归纳法可以求得这个概率密度如式（2）所示。

$$f_k(l)=\frac{1}{(k-1)!l}(knl)^k \mathrm{e}^{-knl} \tag{2}$$

证毕。

根据定理 1，DHT 网络中节点负载与节点的 ID 管理空间是成正比的，因此负载平衡问题可以演化为物理节点总的 ID 管理空间大于均值的概率，如式（3）所示。

$$p(f_k>\mathrm{E}(l))=p_m\left(\frac{(nk)^m}{(m-1)!}l^{m-1}\mathrm{e}^{-(nkl)}>\mathrm{E}(l)\right) \tag{3}$$

定理 2 DHT 网络中物理节点出现 SVSP 的概率模型可以转化为马尔科夫链过程。

证明 为了建立 SVSP 的概率模型，首先要研究 VS 转移的统计学意义。对于一个拥有 k 个 VS 的物理节点，其 VS 的转移由系统的负载阈值和该节点的负载利用率（节点上的负载与节点能力的比值）决定。当节点的负载利用率大于系统负载阈值时，说明此物理节点已经处于过载状态，负载平衡算法被调用以实现 VS 的迁移。就单个物理节点而言，其 VS 的迁移与上一次负载平衡过程中 VS 的增加或者减少无关。因此，物理节点上 VS 迁入迁出的过程可以转化为马儿科夫链过程，如图 1 所示。

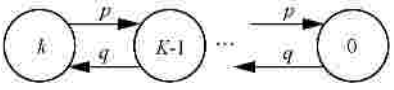

图 1 虚拟服务器迁移过程

图 1 说明了 DHT 网络中一个物理节点出现 SVSP 的过程，即物理节点上 VS 的个数从 k 个减少到 1 个，并且剩下的这个 VS 的负载仍然使该物理节点处于过载状态的过程。因此，计算 SVSP 的概率转化为计算该马尔科夫链状态转移的概率。

证毕。

为便于进行数学计算，本文给出如下假设。

假设 1 VS 的迁移单位是单个的。在实际过程中，由单次负载平衡导致迁移的 VS 虽然有可能有多个，但可以把多个 VS 的迁移过程看做是在上述状态中的多次迁移，因此，并不会对数学推导造成影响。

假设 2 物理节点上 VS 的迁移过程是单向的。在迁移过程中，如果考虑物理节点上 VS 数目的反复，即马尔科夫链在几个相邻状态之间来回转换，那么计算过程将变得异常复杂。因此，在以下的计算和推导过程中，本文把物理节点上 SVSP 的出现过程看成是一个从 k 个 VS 一直转移到 0 个 VS 的过程。这种假设也是与实际过程相符合的，因为对于大部分能力弱的物理节点，在出现过载之后再接收其他节点转移负载的概率是非常小的。

因此，根据式（3），单个物理节点上出现 SVSP 的概率 P 可以表示如下（T 为物理节点应该承担的负载）

$$P=p(f_k>T)p(f_{k-1}>T)\cdots p(f_1>T) \tag{4}$$

其中，k 为物理节点上初始放置的 VS 个数。而对于具有 n 个物理节点的样本空间来说，发生 SVSP 的概率应该属于二项式分布。因此该样本空间中有 j 个节点发生 SVSP 的概率可以表示为

$$P(j)=\binom{n}{j}P^j(1-P)^{n-j} \tag{5}$$

由于概率值小于 1，因此，通过式（4）和式（5），可以得出一个简单的结论：物理节点上的 VS 个数越多，出现 SVSP 的概率越低。接下来，本文将利用以上数学模型对 SVSP 进行详细的概率分析。

3.3 SVSP 的概率分析

在实际 DHT 网络中，受存储容量、计算能力、带宽等因素影响，节点能力并非均匀分布。因此，本文以 x 表示某个物理节点的能力，$T(x)$ 表示某个物理节点的负载阈值。

定理 3 节点能力非均匀分布中的 DHT 网络中具有 K 个虚拟服务器的物理节点上出现 SVSP 的概率为 $\left(\frac{1}{(k-1)!}(nkT(x))^{k-1}+\cdots+1\right)\mathrm{e}^{-nkT(x)}$。

证明　根据式（4）可得到如下表达式

$$P = p(f_k > T(x))\,p(f_{k-1} > T(x)) \cdots p(f_1 > T(x)) \quad (6)$$

代入上面 f 的公式，可以有

$$p(f_m > T(x)) = p_m\left(\frac{(nk)^m}{(m-1)!} \cdot l^{m-1} \cdot \mathrm{e}^{-(nkl)} > T(x)\right) \quad (7)$$

综合式（6）和式（7），推导可得式（8）

$$p(f_1 > T(x)) \to \int_{T(x)}^{\infty} n\mathrm{e}^{-nkl}\mathrm{d}l \to \mathrm{e}^{-nkT(x)}$$

$$p(f_2 > T(x)) \to \int_{T(x)}^{\infty} \frac{n^2}{1} l\mathrm{e}^{-nkl}\mathrm{d}l \to (1+nkT(x))\mathrm{e}^{-nkT(x)}$$

$$\vdots$$

$$p(f_k > T(x)) \to \int_{T(x)}^{\infty} \frac{n^k}{(k-1)!} l^{k-1}\mathrm{e}^{-nkl}\mathrm{d}l \to$$

$$\left(\frac{1}{(k-1)!}(nkT(x))^{k-1} + \cdots + 1\right)\mathrm{e}^{-nkT(x)} \quad (8)$$

证毕。

根据定理 3，物理节点出现 SVSP 的概率与节点的能力有关。为了完善概率的推导结果，设 $q(x)$ 为节点的能力分布，$nq(x)$ 表示能力为 x 的节点个数。对于总负载为 1 的网络，某个物理节点应该承担的负载与该类节点共同承担的负载和该类节点个数有关，可以对 $T(x_k)$ 做如下计算

$$T(x_k) = \frac{nq_k(x)x_k}{nq_1(x_1)x_1 + \cdots + nq_n(x_n)x_n} = \frac{x_k}{n\mathrm{E}(x)} \quad (9)$$

其中，$\mathrm{E}(x)$ 被用来表示系统所有节点的能力分布均值，代入式(8)可以得到

$$p(f_1 > T(x)) \to \mathrm{e}^{-k\frac{x}{\mathrm{E}(x)}}$$

$$p(f_2 > T(x)) \to \left(1 + k\frac{x}{\mathrm{E}(x)}\right)\mathrm{e}^{-k\frac{x}{\mathrm{E}(x)}}$$

$$\vdots$$

$$p(f_k > T(x)) \to \left(\frac{1}{(k-1)!}\left(k\frac{x}{\mathrm{E}(x)}\right)^{k-1} + \cdots + 1\right)\mathrm{e}^{-k\frac{x}{\mathrm{E}(x)}} \quad (10)$$

通过以上推导，在 DHT 网络的 ID 管理空间已经决定的前提下，单个物理节点出现 SVSP 的概率与该节点的能力 x、节点能力分布的均值 $\mathrm{E}(x)$以及节点上初始化 VS 个数 k 有关。从式（10）可以看出，VS 的个数 k 是节点 ID 管理空间差异的具体表现，k 越大，网络中的 VS 个数越多，相应的 ID 管理空间的分布也越均匀。$x/\mathrm{E}(x)$ 则代表节点能力的异构特性。由于 p 属于负指数分布，从式（10）可以看出，某个节点的能力低于节点能力分布均值 $\mathrm{E}(x)$ 越大，发生 SVSP 的概率越高；相反，节点发生 SVSP 的概率则是可以忽略的。根据推导，为便于计算，设 $x/\mathrm{E}(x)$ 为定值，定义为 1/5~4/5，可以计算得到表 1 结果。

从表 1 数据可以看出，在不同程度的节点能力偏离均值情况下（1/5~4/5），配置低于 4 个 VS，单个节点都将会发生严重的 SVSP，尤其是当节点配置 2 个 VS，节点能力均值比为 1/5 时，发生 SVSP 的概率高达 62%。而当配置 6 个 VS，节点能力均值比达到 1/5 时，单个节点发生 SVSP 的概率也高达 17%。因此，物理节点的能力分布对于单个节点发生 SVSP 的概率会带来较大的影响。

表 1　节点能力非均匀分布下单个节点出现 SVSP 概率

k 的值	$x/\mathrm{E}(x)$ 的值	p 的值
2	1/5	0.629 1
2	2/5	0.363 5
2	3/5	0.199 6
2	4/5	0.106 0
3	1/5	0.470 8
3	2/5	0.175 5
3	3/5	0.055 9
3	4/5	0.016 0
5	1/5	0.243 3
5	2/5	0.030 1
5	3/5	0.002 2
5	4/5	0.000 1
6	1/5	0.168 0
6	2/5	0.010 8
6	3/5	0.000 3
6	4/5	$5.107\,9\times10^{-6}$

利用表 1 的数据，可以对整个 DHT 网络的 SVSP 概率进行计算。假设 DHT 网络中节点的能力分布为 $\{x_1, x_2, \cdots, x_n\}$，$p(x_n, j=0)$ 表示某一类物理节点不发生 SVSP 的概率。

定理 4　节点能力非均匀分布的 DHT 网络中发生 SVSP 的概率为

$$\sum_{i=1}^{n}\left(p(x_i, j=1)\prod_{k=1,k\neq i}^{n} p(x_k, j=0)\right)$$

证明 由概率乘法定律，可以得到整个 DHT 网络中不发生 SVSP 的概率为各能力类型节点不发生 SVSP 概率的乘积，表示如式（11）所示。

$$P(j=0)=p(x_n,j)\,p(x_{n-1},j)\cdots p(x_1,j) \tag{11}$$

其中，某一特定能力类型节点发生 SVSP 的概率依然可以采用二项分布进行计算为式（12）所示。

$$p(x_n,j)=\begin{pmatrix} nq(x_n) \\ j \end{pmatrix} P^j(1-P)^{nq(x_n)-j} \tag{12}$$

对于整个 DHT 网络发生 SVSP（$P(j=1)$）情况的讨论，则不适用概率论的乘法原则。这是因为在 $P(j=1)$ 的情况下，如果某一能力类型物理节点内已经有一个节点处于 SVSP 状态，则要求其他类的节点不再发生 SVSP。因此，整个网络处于 $P(j=1)$ 的概率可以表示为某一类型节点发生 SVSP 而其他类型节点不发生 SVSP 的概率总和

$$P(j=1)=\sum_{i=1}^{n}\left(p(x_i,j=1)\prod_{k=1,k!=i}^{n} p(x_k,j=0)\right) \tag{13}$$

证毕。

在实际的 DHT 网络环境中，节点的能力分布一般类似于 Pareto 分布[20,21]。为便于计算，本文以层次分布的方法来模拟各类型的节点能力分布。假设网络规模为 1 000 个物理节点，可通过 $x/\mathrm{E}(x)$ 值来划分具有不同能力类型的节点，每种类型的节点数目相同。如：$x/\mathrm{E}(x)$ 值为 1/5~4/5，表示 $x/\mathrm{E}(x)$ 值是 1/5、2/5、…、4/5，每个值的节点数为 125 个。依照式（12）和式（13）计算可得到表 2 的值。

比较表 1 和表 2，可以发现物理节点的能力分布越均匀，发生 SVSP 的概率越低。而单个节点发生 SVSP 的概率直接影响整个网络发生 SVSP 的概率。因此，可以得出以下结论：DHT-Base P2P 网络中，在随机选择虚拟服务器 ID 管理空间的情况下，节点能力分布越均匀，发生 SVSP 的概率越小。

因此，在以服务器为集群节点的 DHT 网络中，节点的能力相差不大，SVSP 不易发生。但对于在 Internet 上以不同接入方式接入的普通节点来说，其能力相差较大，SVSP 就成为了一个不可回避的问题。

表 2 节点能力非均匀分布下 DHT 网络发生 SVSP 的概率

k 的值	能力分布范围（$x/\mathrm{E}(x)$）	$P(0)$的值
2	1/5~4/5	$2.900\,5\times10^{-97}$
2	3/5~4/5	$4.583\,0\times10^{-37}$
2	4/5	$4.664\,0\times10^{-25}$
3	1/5~4/5	$9.504\,9\times10^{-50}$
3	3/5~4/5	$1.007\,6\times10^{-8}$
3	4/5	$3.144\,5\times10^{-4}$
5	1/5~4/5	$1.206\,0\times10^{-17}$
5	3/5~4/5	0.562 4
5	4/5	0.951 2
6	1/5~4/5	$2.566\,6\times10^{-11}$
6	3/5~4/5	0.926 6
6	4/5	0.997 5

4 基于虚拟服务器拆分的负载平衡算法(VSSLBA)

4.1 算法思想与分析

通过上述数学分析，SVSP 的存在主要是由于物理节点的异构性和虚拟服务器管理的 ID 空间不平衡所造成的，现有研究成果尚未对此问题提出合适的解决方案。考虑 DHT 网络中节点自治性特点，物理节点的异构性是无法改变的。面对 SVSP，一个关键的问题是物理节点上没有足够的 VS 转移负载，增加 VS 的数目固然能解决 SVSP 问题，却带来较大的系统消耗。在本文中，提出一种拆分 VS 的算法（VSSLBA）来解决 SVSP 问题。

VS 拆分的基本思想是在节点初始化阶段仅仅维护一个 VS，随着系统运行，节点将根据负载需要对原 VS 进行动态拆分，拆分后原 VS 管理的地址空间将由新生成的 VS 分散管理。由于在 DHT 算法中，负载随地址空间而移动。所以，原 VS 管理的负载将由新生成的多个 VS 管理。在 SVSP 状态下，可以通过拆分 VS 的方法把只剩下的一个 VS 拆分为多个，如此即有足够的 VS 来转移负载。这种动态调整 VS 个数的方法既可以有效解决 SVSP 问题，又可以最大限度地节约系统资源。下面，详细分析 VS 拆分的理论依据。

定理 5 虚拟服务器的拆分可以转化为对该虚拟服务器原 ID 管理空间的拆分。

证明 根据 DHT 网络的数据管理机制，数据存储在与其标识 ID 相近的节点上，即某个 VS 所管

理的负载分布可以近似表示为其所管理的 ID 空间范围。因此，对 VS 进行拆分可以通过对其所管理的 ID 空间范围进行拆分来实现。当某个 VS 需要进行拆分时，它可以根据负载分布从属于自己管理的 ID 空间范围中取出一个合适的 ID 以分配给新的 VS，而拆分生成的新 VS 将管理原 VS 的部分 ID 空间范围。本文以 Chord 为例对这个过程进行说明。

在图 2 中，假设虚拟节点 v9 在 DHT 网络上的前驱为 v5，因此，v9 所管理的 ID 空间为（5,9]。同时假设，网络中存在着 ID 为 6、7 和 9 的 3 个负载，简称：l6、l7 和 l9。根据 DHT 算法，l6、l7 和 l9 归属于 v9 管理。如果现在要对 v9 进行拆分，可以根据 v9 的负载分布和负载阈值，在（5,9]这一段 ID 空间中选择一个合适的 ID 来生成新的虚拟服务器。本文假设需要将 l6 和 l7 拆分出去。从图中可以看出，可以管理到负载 l6 和 l7，但又不影响 l9 的地址空间在[7，8]上。因此，可以在[7，8]之间任意选择一个 ID 来作为新的虚拟服务器的 ID，从而达到拆分出 l6 和 l7 的效果。如果选择 8 作为新虚拟服务器的 ID，则可以得到 2 个新的虚拟服务器及其所管理的负载如图 3 所示。

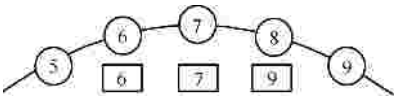

图 2　v9 的负载分布

图 3　v9 拆分后的结果

证毕。

VS 拆分可能会带来潜在的 ID 冲突问题。ID 冲突即拆分后 VS 的 ID 与原 DHT 网络中的 VS ID 相同，这在 DHT 网络中是不允许发生的，本文将对这种情况加以讨论和分析。

定理 6　不同物理节点上同时经过拆分的虚拟服务器不会产生相同的新虚拟服务器 ID。

证明　由于每个 VS 所管理的 ID 空间是独立且互不相同的，所以不同的虚拟服务器拆分后生成的 VS ID 不可能存在冲突。

证毕。

导致 ID 冲突的另外一种情况是新节点加入 DHT 网络时。由于节点新加入 DHT 覆盖网络时的节点 ID 标识一般会根据节点的唯一物理属性独立生成，而 VS 的拆分将直接根据负载分布来产生新 VS 的 ID 标识，这样就会出现节点 ID 冲突的情况。这种冲突的概率与 DHT 覆盖网络总的 ID 空间和包含的物理节点数目有关。由于 2 个相同 ID 的 VS 产生时，它们都会向它们的后继节点获取网络信息才可以加入 DHT 网络。因此，可以在节点加入 DHT 网络时使用 ID 冲突检测机制，一旦接收到一个已经存在的节点 ID 加入请求，后继节点可以通过冲突拒绝消息来使 DHT 网络中的每个 VS 的 ID 保持唯一性。

4.2　VSSLBA

本文以 Split_VS(threshold T)来描述上述的拆分过程。其中 T 是负载平衡的阈值，它由每次负载平衡计算出来，并且通过目录节点传递给需要拆分的虚拟服务器。

算法 1　Split_VS(threshold T)

1) /*L 记录 VS 上的总负载*/

2) /* (ID_p, ID_s] vs 的 ID 管理空间范围*/

3) /* b 保存检索过程中的负载值*/

4) /* split_ID 是通过负载检索得到的新 VS 的 ID 值*/

5) vs’Split_VS(threshold T)

6) BEGIN

7) $L=\sum l_v$ /*计算 VS 上的总负载，l_v 代表单个负载*/

8) $split\_burden = L * T$ /*计算该 VS 需要拆分的负载*/

9) FOR 从 i =ID-p 到 i =ID-s 检索每一个负载的值

10) BEGIN

11) $b = b + l_i$

12) IF $b >= split\_burden$

13) THEN BEGIN

14) $split\_ID$ = random(l_i ’s ID, l_{i+1} ’s ID)

15) END

16) END

17) Initial_new_vs($split\_ID$)

18) END

物理节点收到需要进行 VS 拆分的消息和系统相应的负载阈值 T 后将执行上述拆分算法。目录节点把系统负载阈值 T 传递给需要拆分的 VS，VS 调用 Split_VS(threshold T)。该 VS 首先将统计自己目前所保存的总负载大小，然后通过与负载阈值 T 相乘得到相应的需要拆分出来的负载 split_burden

（第 7~8 行）。根据该 split_burden，节点开始按序检索该 VS 管理的负载（第 9 行）。在检索过程中，split_burden 被用来与累加的负载值不断进行比较，直到该过程找到一个适合的负载值使得 $b >= split\_burden$ 。由该负载值即可得到 $split\_ID = \text{random}$（l_i's ID，l_{i+1}'s ID）（第 10~15 行）。算法最后通过调用 Initial_new_vs($split\_ID$) 完成新生成 VS 的加入。

拆分后的新 VS 可以按照 DHT 网络原有的算法加入网络。但是，考虑拆分后新加入的 VS 和原 VS 在网络中一定是前驱和后继的关系，它们之间关于 DHT 网络中的信息基本相同，同时，新旧 VS 都处于同一个物理节点内。所以，本文从资源节约的角度出发，拆分后新 VS 的生成和加入直接通过该物理节点的虚拟服务器管理进程就可完成。其伪代码表述形式如算法 2 所示。

算法 2 Create_new_vs($split\_ID$)

1) /* new_vs 新创建的 VS 进程*/
2) /* old_vs 被拆分的 VS 进程*/
3) Create_new_vs($split\_ID$)
4) BEGIN
5) new_vs=fork(old_vs 进程)
6) new_vs.setID($split\_ID$)
7) adjust_information(new_vs，old_vs)
8) END

算法 2 中，fork（）函数对原 VS 进程进行复制（第 5 行），新产生的 VS 进程将获得原进程的数据空间等相应资源副本。新 VS 进程产生后，通过 setID($split\_ID$)（第 6 行）改变新进程的节点 ID 以使其可以正式加入到 DHT 网络。adjust_information (new_vs，old_vs)（第 7 行）分别调整新旧 VS 中如指取表等相关的 DHT 网络信息。当然，本文提出 Create_new_vs（）方法只为在创建新虚拟服务器时提供一种便捷方式，在某些 DHT 算法中如果此过程过于复杂，依然可以采用初始节点加入的方法完成。

虚拟服务器拆分算法将在物理节点被检测到处于 SVSP 状态时调用。目录节点完成对物理节点状态的检测，其实现伪码如算法 3 所示。

算法 3 New_Balance_process()

1) /* c_n 代表参与 DHT 网络的物理节点能力*/
2) /*L_v 表示物理节点上单个 VS 的负载*/
3) /*T 表示某次负载平衡过程中计算得到的系统负载阈值*/
4) New_Balance_process (void)
5) BEGIN
6)获取物理节点相关信息 $\leftarrow (c_n, \{L_{v1} \cdots L_{vm}\})$
7) $T_n = \sum L_n / c_n$ ， $T = \sum_{\text{all}} L_n / \sum c_n$ /*计算单个物理节点的负载阈值和系统的总负载阈值*/
8) FOR 从 1 到 n 检测每一个参与负载平衡的节点
9) IF $T_n > T$ AND 物理节点 n 只有一个 VS
10) THEN BEGIN
11) 调用物理节点 n 上的拆分算法 vs'Split_VS(threshold T)
12) 接收拆分后的物理节点信息 $\leftarrow (c_n, \{L_{v1} \cdots L_{vm}\})$
13) 更新被拆分后的物理节点信息
14) END
15) END
16) 继续执行其他负载平衡步骤
17) END

算法 3 由目录节点调用。目录节点首先获取参与网络的物理节点的负载信息 $(c_n, \{L_{v1} \cdots L_{vm}\})$ 并计算得到各个物理节点的负载阈值 $T_n = \sum L_n / c_n$ ，由此得到系统总的负载阈值 $T = \sum_{\text{all}} L_n / \sum c_n$（第 5~7 行）。目录节点随后根据阈值 T 对其管理的物理节点负载进行检查（第 8 行）。当节点处于 SVSP 状态时，目录节点远程调用该物理节点的 VS 拆分算法，并接收 VS 拆分后的相应信息。由于拆分过程只是改变 VS 个数，并不会改变物理节点的负载信息，因此，目录节点只需要更新其相关的节点信息即可（第 9~13 行），之后目录节点可以按照原负载平衡算法继续运行[7]（第 16 行）。与经典的虚拟服务器负载平衡算法相比，算法 3 会带来一定的网络消息消耗。但由于不需要传递过大的数据结构，额外开销可以容忍。

5 仿真实验

5.1 实验环境

仿真实验在 oversim[22]平台上进行，本文以 Chord 网络作为实验原型，分别实现了经典的基于虚拟服务器的动态负载平衡算法[7] 和 VSSLBA。实际节点数设定为 1 000。虚拟服务器的负载设定与其 ID 管理空间成比例。节点能力的配置采用 2 种

节点能力分布（层次能力分布和 Pareto 能力分布），分别应用到不同的实验环境中。模拟环境中 DHT 网络已实现完全负载平衡的假设如下：由某次负载平衡过程中需要进行负载平衡的节点数为 0 开始计算，经过 50 次负载平衡都没节点需要迁移虚拟服务器则说明网络已处于完全负载平衡；若没达到 50 次即有节点需要进行负载平衡，则计算重新开始。为表示系统利用率和负载平衡容忍度 a 之间的关系，本文采用基本参数 Load/Target[6],它的定义为

$$\frac{\bar{L}}{\bar{T}}=\frac{\sum_{\text{all}} load}{\sum_{\text{all}} load + a\sum_{\text{all}} capacity}$$

其中，a 被设置为 0.3。

5.2　SVSP 概率分析仿真

本文第 3 节对 SVSP 进行了概率推导和数学计算，本节将对其结果进行验证。由于 DHT 网络中物理节点能力分布均匀时发生 SVSP 的概率基本可以忽略不计，所以将不对该情况进行实验验证。根据前文讨论，当物理节点能力非均匀分布时，单个物理节点发生 SVSP 的概率与整个网络发生 SVSP 的概率存在着直接的联系。因此，本节将只对 DHT 网络中物理节点能力分布不均时单个物理节点发生 SVSP 的概率进行验证。节点的能力分布使用层次分布：1 000 个节点中，节点的能力分别设置为 100、200，直到 900，其中除了节点能力为 500 的节点数是 200 个外，其他能力的节点皆为 100 个。实际节点能力值的设定是为了构造出 $x/\mathrm{E}(x)$ 值对应的 1/5~4/5 时的情况。实验结果均为 25 次实验后统计得出，如图 4~图 7 所示。

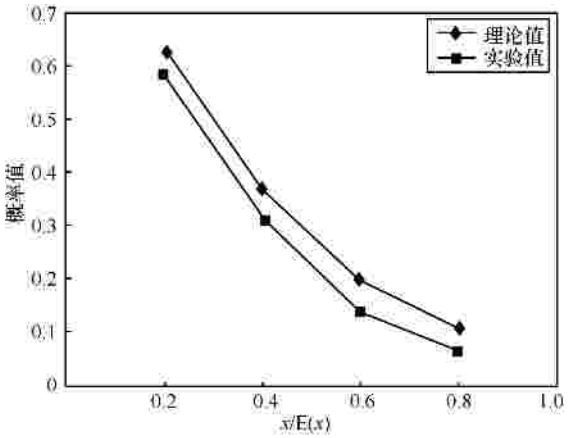

图 4　初始化 2 个 VS 发生 SVSP 的概率

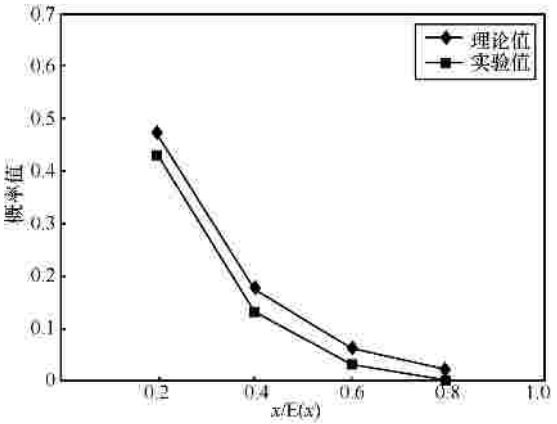

图 5　初始化 3 个 VS 发生 SVSP 的概率

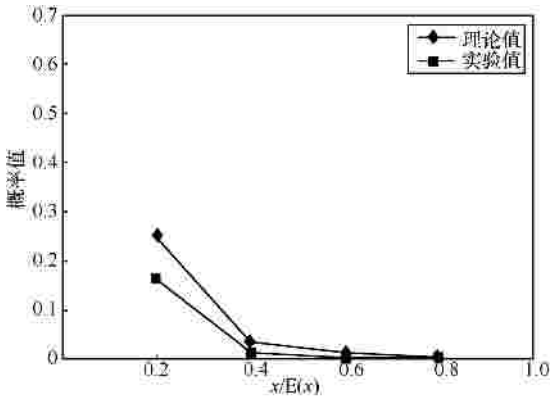

图 6　初始化 5 个 VS 发生 SVSP 的概率

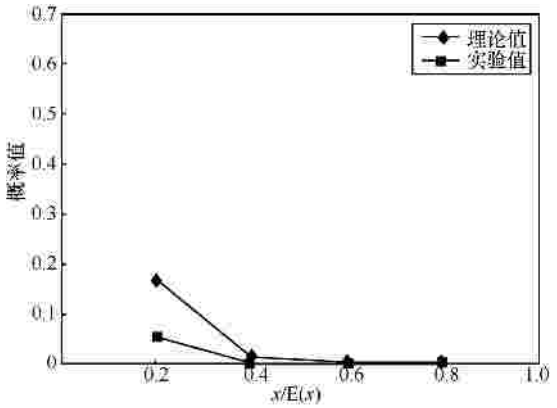

图 7　初始化 6 个 VS 发生 SVSP 的概率

从图 4~图 7 可以看出，实验中测得的发生 SVSP 概率值曲线基本与理论值相似，证明本文对 DHT 网络中发生 SVSP 问题的概率推导是正确的。实验结果同时也反映出发生 SVSP 的实际概率值均较理论值稍低。通过对实验过程仔细分析，产生这种结果的原因主要是由于 DHT 网络在实际执行负载平衡过程中，VS 将优先从能力小的物理节点转移到能力较大的物理节点上，这将导致留在能力较小物理节点上的 VS 负载相对较轻，而这种状况是在概

率推导过程中没有考虑到的。当然，虽然实验值与理论值具有一定偏差，但是计算结果与实验结果在概率趋势上具有一致性。

5.3 虚拟服务器拆分算法的有效性检验

虚拟服务器拆分算法的基本思想是在物理节点出现 SVSP 问题时对单个虚拟服务器进行拆分来解决该问题。为了检验拆分算法的有效性，本节测试了 1000 个物理节点在不同初始化 VS 个数情况下 DHT 网络最终达到负载平衡后的 VS 总个数。实验结果取 5 次实验的平均值，如图 8 所示。

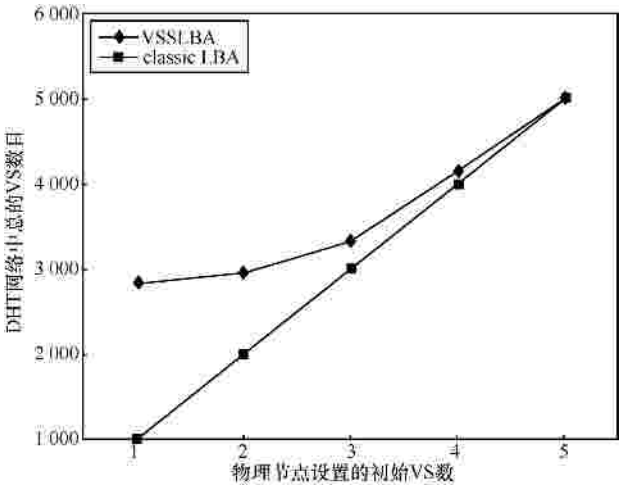

图 8 物理节点上不同数目初始 VS 拆分后的 VS 总个数

图 8 比较了执行 2 种算法后 DHT 网络中初始和最终的 VS 个数，其中，x 轴表示单个物理节点放置的初始 VS 个数，y 轴表示网络中经过负载平衡后的 VS 总个数。经典算法中 VS 的个数在网络中没有节点离开或加入的情况下不会发生变化。从图 8 中可以看出，当物理节点设置的初始 VS 个数较少时，VSSLBA 算法最终所产生的 VS 个数与经典算法中 VS 的个数相差较大。而当物理节点设置的初始 VS 达到 5 个时，VSSLBA 算法最终所产生的 VS 个数与经典算法已经非常接近。这说明在物理节点设置较少的初始 VS 个数时容易出现 SVSP 问题，使得 VSSLBA 算法被多次调用，从而产生了较多的 VS。从图中同时可以看出，当物理节点初始设置 1 个或 2 个 VS 时，VSSLBA 算法最终所产生的 VS 个数在 3 000 左右，远远低于经典算法通常所设置的 5~6 个 VS，这说明 VSSLBA 算法可以利用更少的虚拟服务器个数来实现系统平衡，具有较好的负载平衡效率。

由第 4 节可知，VSSLBA 算法在解决节点 SVSP 问题的同时也会带来一定的消息开销。图 9 给出了初始化 2~5 个 VS 下经典算法和 VSSLBA 算法所消耗的消息数比较结果。

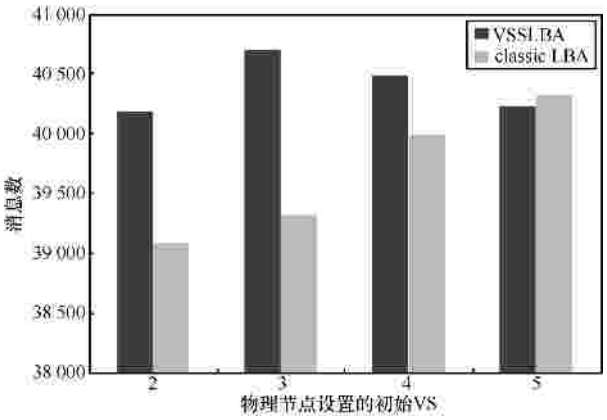

图 9 算法消耗的消息数对比

如图 9 所示，x 轴表示单个物理节点初始化时放置的 VS 个数，y 轴表 DHT 网络中进行负载平衡所产生的总消息数。从图中可以看出，在设置不同初始 VS 个数情况下，两种算法消耗消息的总数差别不大。这主要是由于实验中物理节点个数不会变化，而网络消息数的产生则主要与物理节点个数有关。随着初始虚拟服务器个数的递增，经典平衡算法消息数的递增较为明显，而 VSSLBA 算法则没有明显的变化趋势，这主要是由于 VSSLBA 算法被调用的随机性引起的。从图 9 中还可以看出，VSSLBA 算法的引入增加一定的消息开销，但增加的消息开销都控制在 10%以内，这种开销完全可以接受。尤其是当初始放置 5 个 VS 时，由于发生 SVSP 的概率较小，2 种算法消耗的消息数大致相同。

5.4 VSSLBA 对负载平衡性能的影响

实现负载平衡是对 VSSLBA 算法的基本要求。负载平衡算法应该使系统中能力越强的节点承担的负载越大。为了验证 VSSLBA 算法的负载平衡性能，本文在模拟环境中将 Load/Target 值设定为 0.80，从而使系统处于较高的利用率。同时，节点能力分布采用更符合实际环境的 Pareto 分布，实验结果如图 10 和图 11 所示。

图 10 给出了模拟实验中初始阶段物理节点能力和其相应负载的分布情况。如图 10 所示，系统中的高能力节点相对较少且节点利用率较低，低能力节点较多且节点利用率较高，尤其是能力分布在 1 000 到 4 000 的部分节点所承担的负载已经基本处于利用率的上限。

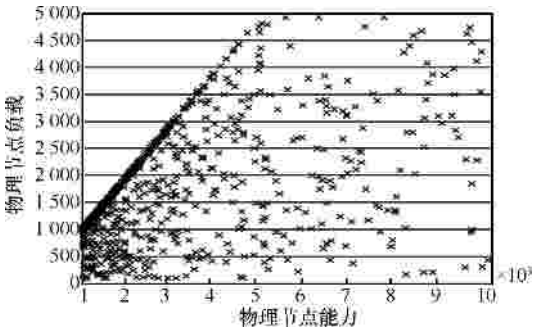

图 10　初始阶段物理节点能力和负载分布

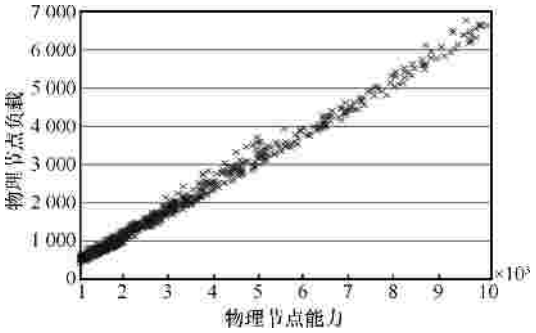

图 11　VSSLBA 算法的负载平衡效果

图 11 给出了运行 VSSLBA 算法后的物理节点能力和其相应负载的分布情况。从图 11 中可以看出，运行 VSSLBA 算法后各类型物理节点的利用率均按照其能力分布控制在相应的合理水平，这与经典负载平衡算法效果基本相同，表明 VSSLBA 算法的加入在避免 SVSP 的同时可以有效地进行负载平衡。

VSSLBA 算法的另一个特点是可以动态调整网络中 VS 的个数。如前文所述，在物理节点上部署较多 VS 可以有效避免 SVSP 问题，但 VS 越多所带来的系统开销越大。经典负载平衡算法只能部署固定数目的 VS，无法有效的应对负载分布和虚拟节点数目的平衡问题。图 12 和图 13 给出了 VSSLBA 算法在此方面的性能评价。

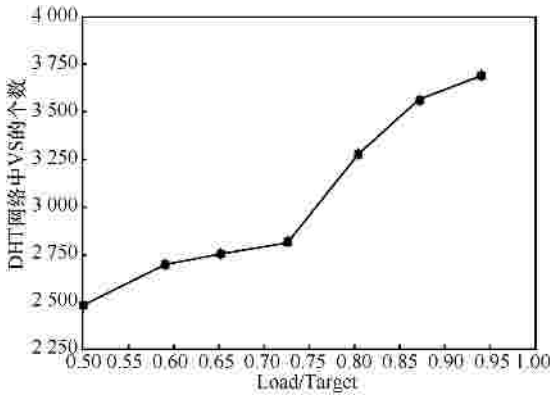

图 12　不同 Load/Target 值下的虚拟服务器个数

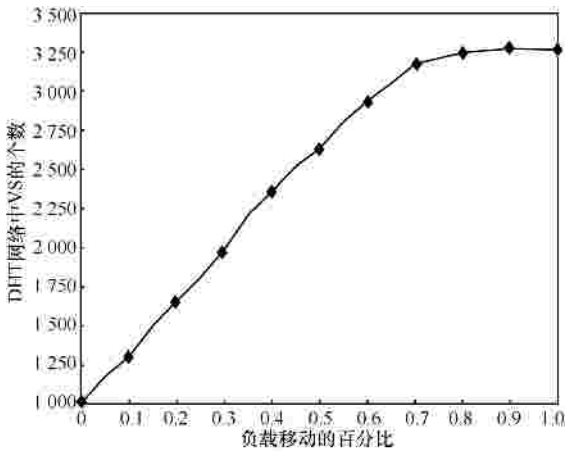

图 13　不同比例负载迁移对虚拟服务器个数的影响

图 12 给出了系统处于不同 Load/Target 值下运行 VSSLBA 算法达到负载平衡后 VS 的最终个数。如图 12 所示，系统中 VS 的个数随着系统利用率的增加而动态增加。当系统利用率接近 1 时，VSSLBA 算法可以将 VS 的个数控制在 3 750 以下，相较于经典算法中每个物理节点部署 5 个 VS 的作法，可以有效减少维护 VS 的开销。

图 13 给出了 Load/Target 值固定为 0.8 时不同移动负载比例下（系统达到负载平衡后所移动的负载定为 1）系统中 VS 个数的变化情况。从图 13 中可以看出，在负载移动比例从 0 到 0.6 范围之间变化时，系统中 VS 个数也呈线性显著增加。这说明在系统的初始阶段 VSSLBA 算法主要靠拆分 VS 来应对 SVSP 和负载平衡问题。当负载移动比例超过 0.6 以后， VS 个数增长明显变缓直至没有增加。这说明 VS 的个数已能满足需要，VSSLBA 算法利用系统已有的 VS 即可实现负载平衡。

图 12 和图 13 的实验结果表明，VSSLBA 算法可以根据需要动态的调整网络中 VS 的个数以实现负载平衡，从而能够有效地节约系统资源。

6　结束语

覆盖网络中经典的负载平衡算法不能解决 SVSP 问题，本文在对 SVSP 建立数学分析的基础上，提出了一种虚拟服务器拆分算法，仿真实验结果表明本文的解决方案在有效解决 SVSP 问题的同时能够实现良好的负载平衡性能。

参考文献：

[1] STOICA I, MORRIS R, KARGER D, *et al*. Chord: a scalable peer-to-peer lookup service for internet applications[A]. Proc of the

ACM SIGCOMM '01 Conference on Applications, Technologies, Architectures, and Protocols for COMPUTER Communications[C]. San Diego, 2001.149-160.

[2] DABEK F, KAASHOEK M F, KARGER D, *et al*. Wide-area cooperative storage with CFS[A]. Proc of the 18th ACM Symp on Operating System Principles(SOSP)[C]. Banff, 2001. 202-215.

[3] ROWSTRON A, DRUSCHEL P. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems[A]. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)[C]. Heidelberg, 2001. 329-350.

[4] DRUSCHEL P, ROWSTRON A. PAST: a large-scale, persistent peer-to-peer storage utility[A]. Proc of the Eighth Workshop o Hot Topics in Operating Systems[C]. Schoss Elmau, 2001. 75-80.

[5] ZHAO B Y, KUBIATOWICZ J D, JOSEPH A D. Tapestry: an Infrastructure for Faulttolerance Wide-Area Location and Routing[R]. U C Berkeley Technical Report UCB//CSD-01-1141, Berkeley, 2001. 329-350.

[6] ANANTH R, LAKSHMINARAYANAN K, SURANA S, *et al*. Load balancing in structured P2Psystems[A]. Proc IPTPS[C]. 2003. 68-79.

[7] GODFREY B, LAKSHMINARAYANAN K, SURANA S, *et al*. Load balancing in dynamic structured P2P systems[A]. Proceedings of IEEE INFOCOM[C]. 2003. 2253-2262.

[8] GODFREY P, STOICA I. Heterogeneity and load balance in distributed hash tables[A]. Proceedings of IEEE INFOCOM[C]. 2005. 596-606.

[9] ZHU Y W, HU Y M. Efficient, proximity-aware load balancing for DHT-based P2P systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2005,16(4):349-361.

[10] WANG X H, PENG Y X, LI D S. An efficient load balancin method for constant degree P2P systems[A]. Computer Design and Applications (ICCDA), 2010 International Conference on[C]. Qinhuangdao, China, 2010. 25-27.

[11] HSIAO H C, LIAO H. Load balance with imperfect information in structured peer-to-peer Systems[J]. IEEE Trans Parallel Disturb System, 2011, 22(4):634-649.

[12] GUPTA A, AGRAWAL D, A ABBADI E. Approximate range selection queries in peer-to-peer systems[A]. Proceedings of the First Biennial Conference on Innovative Data Systems Research[C]. Asilomar, 2003.

[13] CAI M, FRANK M, CHEN J, *et al*. MAAN：a multi-attribute addressable network for grid information services[J]. Journal of Grid Computing, 2004,2(1):3-14.

[14] MAYMOUNKOV P, MAZIERES D. Kademlia: a peer-to-peer information system based on the XOR metric[A]. Proceedings f the 1st International Workshop on Peer-to-Peer Systems(IPTPS'02)[C]. 2002. 53-65.

[15] KRISHNAMURTHY S, LANSARY S E, ERIK A, *et al*. An analytical study of a structured overlay in the presence of dynamicmembership[J]. IEEE/ACM Transactions on Networking(TON), 2008, 16(4): 814- 825.

[16] **聂晓文, 卢显良, 周旭等**. DHT **算法基本统计特性及应用**[J]. **四川大学学报(工程科学版)**, 2009，41(5):170-175.
NIE X W, LU X L, ZHOU X, *et al*. Elementary statistical properties in DHT and their application[J]. Journal of Sichuan University(Engineering Science Edition), 2009, 41(5):170-175.

[17] **聂晓文, 卢显良, 孟江涛等**. **一类** DHT **算法中负载的概率分布**[J]. **计算机应用研究**,2009,26(10):3763-3766.
NIE X W, LU X L, MENG J T, *et al*. Probabilistic distribution of load in one class of DHT[J]. Application Research of Computers, 2009, 26(10):3763-3766.

[18] **肖波, 聂晓文, 侯孟书**. DHT **网络规模估计算法的定量分析与设计**[J]. **电子科技大学学报**,2011,40(2):261-266.
XIAO B, NIE X W, HOU M S. Quantitative analysis and design of an estimation algorithm on DHT network size[J]. Journal of University of Electronic Science and Technology of China, 2011,40(2): 261-266.

[19] **聂晓文**. DHT **覆盖网若干基础性问题研究**[D]. **西安:电子科技大学**, 2009.37-45.
NIE X W. Research on Several Fundamental Problems of DHTOverlay[D]. Xi'an: University of Electronic Science and Technology of China, 2009.37-45.

[20] BUSTAMANTE F E, QIAO Y. Friendships that last：peer lifespan and its role in P2P protocols, in Web content caching and distribution[A]. Proceedings of the 8th International Workshop[C]. 2004. 233-246.

[21] STUTZBACH D, REJAIE R. Understanding chum in peer-to-peer networks[A]. Proceedings of the 6th ACM sJGCoMM Conference On Internet Measurement Rio de Janeriro[C]. Brazil, 2006. 89-202.

[22] Group of Prof Zitterbart Oversim[EB/OL]. http://www.oversim.org, 2012.

作者简介：

杨磊（1976-），男，湖南临湘人，博士，湖南大学副教授，主要研究方向为分布式系统与网络。

李仁发（1957-），男，湖南郴州人，湖南大学教授、博士生导师，主要研究方向为嵌入式系统与网络。

柳石（1988-），男，湖南岳阳人，湖南大学硕士生，主要研究方向为对等网络。

陈志兴（1984-），男，广西玉林人，湖南大学硕士生，主要研究方向为对等网络。

李肯立（1971-），男，湖南娄底人，湖南大学教授、博士生导师，主要研究方向为并行处理、科学可视化。