分布式后缀树在P2P搜索中的应用

孙名松[1],周三山[1],刘 杰[2]
(1.哈尔滨理工大学 网络信息中心,黑龙江 哈尔滨 150080;
2.哈尔滨理工大学 荣成学院 计算机系,山东 荣成 264300)

摘 要:由于当前的基于DHT的P2P系统在语言搜索方面都有很大的限制,因此建立一种恰当的既具有语言能力又有伸缩性的语言覆盖P2P网络是一种挑战。文中提出一种介于DHT和支持关键字序列查找的语言覆盖之间的中间层DST覆盖网,通过DHT获取并返回给DST覆盖网相应的处理和索引数组,由DST实现关键字序列查找。分析表明它的时间复杂度与关键字序列的长度成线性关系,实验证明在P2P网络上使用基于DST的搜索获得一个确切的文本的查找具有快速性、负载平衡和可用性。

关键词:DHT;知识网络;P2P;语言覆盖;后缀树;负载平衡
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2011)03-0031-05

Distributed Suffix Tree Overlay for Peer-to-Peer Search

SUN Ming-song[1], ZHOU San-shan[1], LIU Jie[2]
(1. Network Information Center, Harbin University of Science and Technology, Harbin 150080, China;
2. Computer Department of Rongcheng College of Harbin University, Rongcheng 264300, China)

Abstract: Current DHT-based P2P networks are limited in their ability to support a semantic search, establishing an appropriate semantic overlay on peer-to-peer (P2P) networks to obtain both semantic ability and scalability is a challenge. This paper proposes the Distributed Suffix Tree (DST) overlay as the intermediate layer between the DHT overlay and the semantic overlay to support the search of a keyword sequence. Through DHT overlay get and return the corresponding net and indexed array to DHT overlay, then lookup the key words sequence by DHT. Analysis show its time cost is sublinear with the length of the keyword sequence, and experiments prove that the DST-based search is fast, load-balanced, and useful in realizing an accurate content search on P2P networks

Key words: DHT; knowledge grid; peer-to-peer; semantic overlay; suffix tree; load balance

1 简 介

1.1 目 的

基于DHT(Distributed Hash Table)的结构化P2P系统有很好的扩展性,该网络内任何一个数据项的查找能够在$O(\log n)$的时间复杂度内完成。早期的基于DHT的系统像CAN, Pastry, Chord, Tapestry都能实现精确标识符匹配的查找。这些P2P系统大多都在语义支持智能应用的能力方面有局限性。在P2P网络上创建一个语义覆盖方法是达到既具有好的扩展性又支持能够提供在大规模网络上的智能应用的语义能力的一个途径。文中提出了一个方法,以达到在基于DHT的系统上创建一个分布式后缀树覆盖网以实现有效的关键字序列查找。关键字序列查找被应用于很多领域,主要根据一篇论文的标题、摘要等条件搜索一篇论文[1]。

1.2 相关工作

目前有三种典型的索引方法解决分布式关键字查找:根据文档分区,根据关键字分区,混合索引。在第一种方法中,每个节点有一批文件[2],在查找中,请求被广播到每个节点并且在本地完成。查找结果将会聚合并返回给用户。第二种方法要求每个节点为它所持有的每个关键字保存一个倒置的文件列表[3-6],用多个关键字响应请求,把这个请求发送到持有这些关键字的节点。由这些节点返回的倒置的列表将会被加入并且返回给用户。第三种方法综合了前两种方法并且提供了一种混合索引结构[7],它得到了搜索效率和通过牺牲存储空间来降低带宽消耗。这些方法都是关于鼓励的关键字的查找。Brushwood系统给P2P网络提供了一个用来分发一个树的结构的总框架[8,9]。它通过前序遍历树节点来保留本地数据,并给每个对等端的每个分区分配连续的片段。DST使用一个不同方法在P2P网络上分发树的结构,并支持有效关键字序列

收稿日期:2010-07-11;**修回日期**:2010-10-27
作者简介:孙名松(1963-),男,教授,研究方向为网络应用、网络安全。

的查找。

2 系统模型

基于DST的系统的架构如图1所示[6,9-11]。

应用程序、DST覆盖网和DHT覆盖网互相影响如下:使用DHT覆盖网的应用程序在P2P网络上储存每一个资源并得到它们的DHT key(表示为DHT-key)。应用程序也负责提供每一个]源的一个概括描述,表示为text,并且发送一个用来处理和索引的数组(text,DHT-key)给DST覆盖网。DST覆盖网提供一个查找字(keyword_sequence),这个查找字被当作应用程序的一个接口,此接口将产生DHT-key资源,这些资源的文本包含了key-sequence。

DST方法支持一个快速并且可靠的关键字序列查找服务,有以下特征:

(1)分散化。它不需要任何超级节点,并没有任何一个节点比其余节点更重要。这就使得DST特别适合于P2P应用程序。

(2)负载平衡。它从来不制造超级节点。它通过使用自然负载平衡的DHT在每个对等端上分配虚拟后缀树。

(3)快速查找。经过优化后,时间复杂度变为$O(T+m)$,并且它有一个上限$O(T+\log n)$,其中T是在DHT下的查找开销,m是关键字序列的长度,n是索引文本的最大长度。

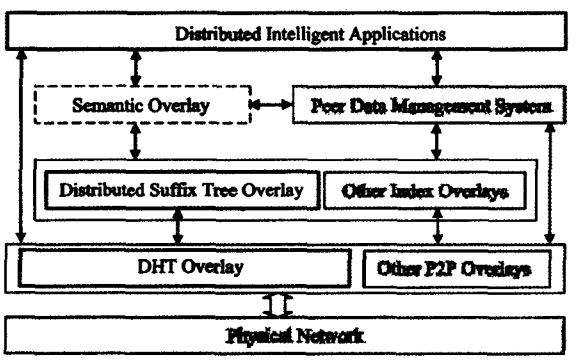

图1 基于DST的系统的架构

3 DST方法

DST方法由四部分组成:

(1)构造虚拟DST的方法。

(2)满足给定的关键字序列的搜索机制结果。

(3)后续操作,例如数组的插入和删除。

(4)用来处理失败节点的恢复机制。

3.1 后缀树

一棵后缀树包含给定文本的所有后缀[10]。在一个长为N的文本中查找一个长为M的字符串,后缀树仅仅只需要M次比较——完成一次查找的最少比较数。

它含有以下特征:

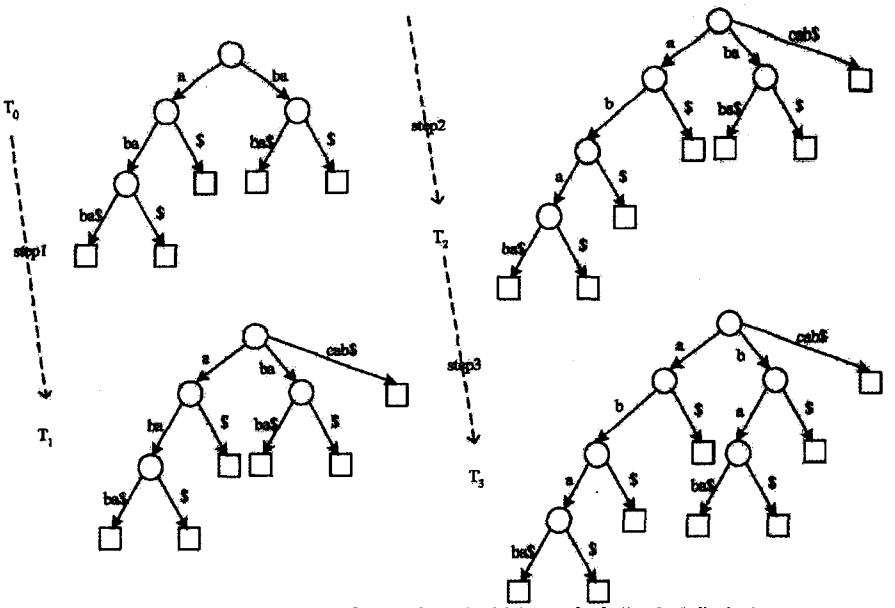

图2 向后缀树插入文本"cab $"的处理

(1)一条弧(边)可以表示文本的任何非空子串。

(2)每个非终端节点,除了根节点,必须至少有两个孩子弧(边)。

(3)兄弟弧(边)表示的子串必须开始于不同的字符。从一棵空树T_0开始,构造过程中产生一个后缀树路径,这个路径对应给定文本的从最长到最短的后缀。图2展示了构造文本"ababa $"的后缀树,这里使用$来表示结尾。

3.2 索 引

DST方法通过在DHT覆盖网使用应用程序交付的文本构造一整棵虚拟后缀树来实现有效关键字序列查找[7]。它需要DHT覆盖网提供的key-to-peer接口。它用词汇代替了字母来扩展后缀树以支持关键字序列查找。

当根边都被分配到DHT覆盖网上之后,其他层的边的Keys可以通过迭代获得,就如他们的Keys对等于他们父边的ChildKey字段。每一个边都被存储在负载这些Key的对等端中。这样,在后缀树中的所有边都被分配到DHT覆盖网上(见图3)。

3.3 查找过程

当开始一个查找,它沿着这条路径,从虚拟后缀树

的根节点开始到下一个满足查找条件的节点,然后再下一个直到到达目标节点[12]。

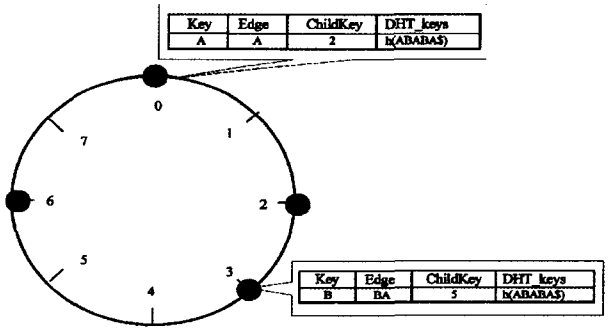

图 3　使用 DHT 覆盖网的 Chord 系统的 DST 中的根边

对于一个给定的关键字序列,$(A_1A_2A_3\cdots A_m)$,查找过程开始于提取 A_1 的初始值,并把它作为 Key K_1。根据上面的特性,这里只有一个条目符合(K_1, A_1)。查找过程通过以下两个方法找出这个条目:

(1)调用 key-to-peer(K_1)找到存有这个条目的对等端 p。

(2)从对等端 p 中找出这个条目,查找那个条目的 Key 值是 K_1,初始值是 A_1。

假设这个条目的 Edge 字段是 $B_1B_2B_3\cdots B_n$;如果 $A_1 = B_1$, $A_2 = B_2$, $\cdots A_n = B_n$,那么(ChildKey, An+1)被用 Key,找出下一个条目的初始值对。继续这些步骤直到到达 A_m,并且返回最后一条结束条目的 DHT-keys 字段。如果某一部失败或者在此过程中某些条件不满足,整个的查找将失败,而且以在 DST 覆盖网中没找到匹配的结束。

如果 DHT 调用 key-to-peer(key)被看作是查找过程的一个步骤,DST 方法可以在 m 步内完成一个关键字序列查找,每一步将会截取至少一个字,m 表示关键字序列的长度。

为了避免频繁调用 DHT 的 key-to-peer 服务,每一个对等端可以使用一个表来存储(cache)ChildKeys 对应的对等端的 IP 地址。由于 DHT 覆盖网的动态化,每一个对等端应周期性的使用 HeartBeat 消息去更新缓存表,以保证 IP 地址和 ChildKeys 对是最新的。因此,当查找过程从父边跳到子边时,key-to-peer(ChildKeys)很少被唤醒。查找的时间复杂度可以被降到 $O(T + m)$,T 表示 key-to-peer 函数调用的时间复杂度,m 表示关键字序列的长度。

4　模拟结果

我们模拟了一个由 1000 个对等端构成的网络,并将 Chord 用作 DHT 覆盖网。从 DBLP XML 数据库中收集数据,那里包含 500000 个页面的 XML 的元数据。首先将选择的页面分布到 DHT 覆盖网上,然后,将那些页面的标题用做文本描述,来构造 DST 覆盖网。

4.1　负载平衡

第一个实验室是,怎样通过 DST 创建条目的项数,这些项数伴随于插入到 DST 的文本项数。DST 和 DHT 覆盖网之间负载平衡的比较是基于这个实验中条目项数的。文本随机的从那 500000 个元数据中挑选,并且总的文本项数从 10^4 - 10^5,从 10^4 递增。对于每一个值,实验都重复 10 遍。图 4 展示了条目总项数和文本总项数之间的联系。我们能够观察到 DST 的空间消耗几乎和插入的资源的数目成线性关系。这是 P2P 网络空间的一个合理使用。

DST 方法的负载平衡的特性,是由通过比较我们模拟中的潜在的 Chord 覆盖网所得出。因此,实验将测试从第一次实验中得出的 DST 的条目和在 Chord 中随机生成的 keys 如何分布在 Chord 上[13]。

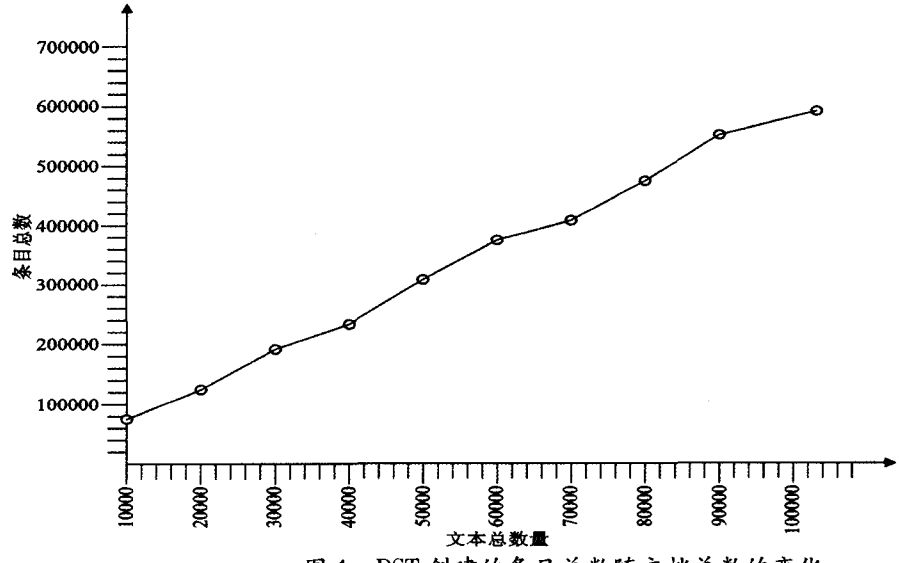

图 4　DST 创建的条目总数随文档总数的变化

图 5 标出了第一个的百分位数、第 50 个的百分位数以及 DST 中每个词条数目的平均数和 Chord 环中每个节点的关键字序列相应的值。图中显示了每个节点中词条的数量跟总的词条数量相比呈现出线性的巨大的变化。第 50 个百分位数是平均数的 5.2 倍,其对应的最大值是平均数的 7.8 倍。尽管有着巨大的差异性,DST 方法显示出与分布式哈希表对等网相似的模

型。

到此,证明了DST方法在不考虑词条数量的单纯方式下具有负载均衡的特性。下面的试验将评估不同级别的词条的分布。可以发现相同等级的边的分布与Chord中关键字的分布相似[9]。

4.2 搜索路径长度

搜索路径的长度是任何P2P系统执行的一个关键因素。在DST环境下,搜索路径长度主要考虑在搜索过程中横越的节点数和它在搜索执行中所用的评估准则。

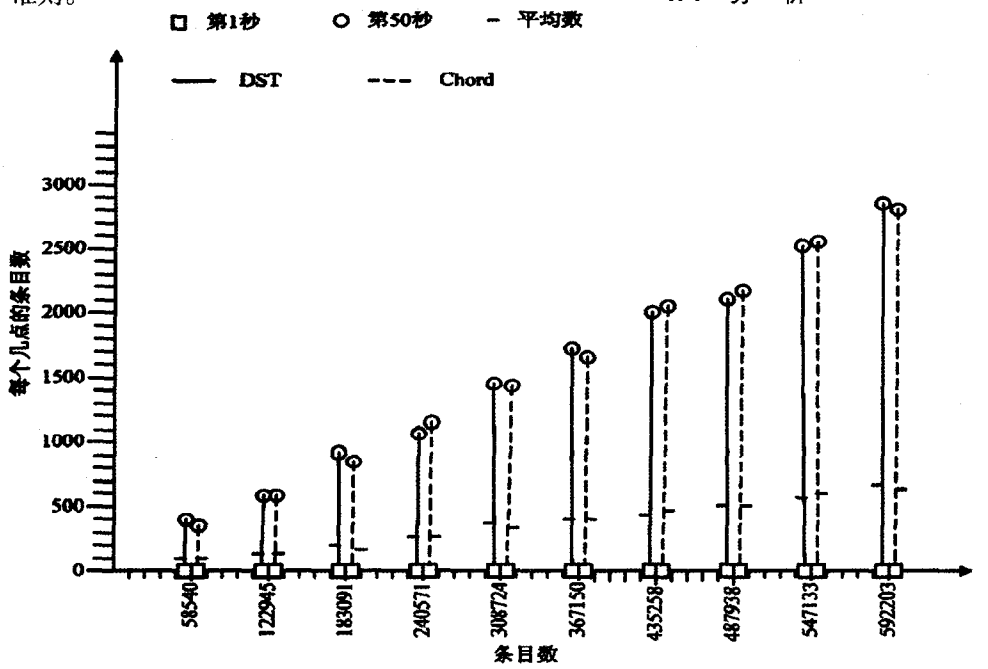

图5 第一个的百分位数、第50个的百分位数以及DST中每个词条数目的平均数和Chord环中每个节点的关键字序列相应的值

索路径长度的有两个因素决定:关键字序列的长度和DHT覆盖网上关键字到节点的函数。关键字到节点的函数所耗费的时间由DHT覆盖网决定(例如,Chord中关键字到节点的函数所耗费的时间是$O(\log n)$)。因此,只评估搜索路径长度是如何随关键字序列长度的不同而变化。搜索请求的关键字序列是从模拟中存储的文本里随机提取的,它们的长度从1到10不等,对于每个长度值,都重复做100次的评估来记录搜索路径的长度。

4.3 结 论

DST方式在关键字序列搜索上具有高的执行效率,尤其是关键字序列较长时。用一个试验来比较DST方式和倒转列表方式。搜索请求的关键字序列从文本中随机提取,每一个提取的关键字序列都同时被发送到DST方式和倒转列表方式。没有高速缓存的DST方式和倒转列表方式的搜索路径长度平均值,当关键字长度大于2时,倒转列表方式呈现出线性的特点,而DST方式趋向于一个恒值。此外,DST方式自动的保持了关键字的顺序关系,而倒转列表方式必须进行类似与交叉和位置计算的深度操作来确保关键字在结果文档中继续存在。尽管这些操作可以在本地处理,但是当结果数据非常庞大的时候就很费时。然而,在DST方式中数组的插入和删除操作比在倒转列表方式中产生更多的信息量。如这个试验所表明的,DST方式引起的数组插入和删除操作产生的数据量是倒转列表方式的2倍。

4.4 分 析

以上仿真实验表明当关键字序列长度比确定的值大时搜索路径长度趋向于一个恒定的值,根本的原因是搜索路径长度有一个较高的上限(即整个虚拟后缀树的深度)。DST的深度可以用一个概率模型来分析:一个文本可以看做一系列独立的随机的引起它在词汇中值变化的因子。

首先定义文本中重复出现的子链为那些在文本中出现超过2次链的子链,例如:对于文本ABABA,因为ABA在ABABA中出现2次,因此子链ABA是一个子链。任何从根节点到非终端节点都作为文本的一个重复子链,每个边放在最后一个字里,我们得出下面的定理:

定理1:对于一个文本,它相应的后缀树的深度小于或等于它的重复子链的最大深度。

现在,通过分析后缀树对应文本中重复子链的深度得到后缀树深度的期望值变得可行了,我们希望知道重复子链深度所展示的是哪种分布概率。找到精确的概率非常困难并且在这个研究中是不必要的。下面的定理给出了上限。

定理2:P^l为文本包含的重复子链的长度,$P^l \geq l$,假定文本的长度为L,词汇表的大小为M,则有$P_l \leq \frac{(L-l)^2}{M^l}$。

定理2表明概率P^l随着l的增长而降低。如果$M=1000$,$L=100000$,则$l \geq 20$的概率接近0。这意味着

一个长度为 10000 的文本对应词汇量为 1000 的词汇表中,后缀树的深度小于 20 的概率很大。

同样,从这些概率论分析来看,当 $l \geq 0$ 时,随机变化值大于 l 的概率之和是随机变化的平均值,因此,重复子串的平均长度是

$$\sum_{l=1}^{L-1} \frac{(L-l)^2}{M^l} < \sum_{l=1}^{L-1} \frac{L^2}{M^l}$$

它也是一个上限值。当 $l < \log_M^{L^2}$ 时,有 $\frac{L^2}{M^l} > 1$,但是概率值永远不会大于 1,因此,重复子串的平均长度可以精确地表达为:

$$\sum_{l=1}^{\log_M^{L^2}} 1 + \sum_{l=\log_M^{L^2}}^{L-1} \frac{L^2}{M^l} < \sum_{l=1}^{\log_M^{L^2}} 1 + \sum_{l=\log_M^{L^2}}^{\infty} \frac{L^2}{M^l} = \log_M^{L^2} + \frac{M}{M-1}$$

正如 $\frac{M}{M-1}$ 一样,长度的平均值呈现对数性的小,因此,搜索请求能够在确定的距离内得到回答。

5　结束论

DST 方法反映了词汇之间的语义顺序,因此它支持拥有大规模分布式文档的 P2P 网络的高效搜索,DST 覆盖网具有可扩展性、快速性和负载均衡性。它进行关键字序列搜索的时间花费与被搜索的串的长度呈准线性关系。由于它只通过关键字-节点函数与 DHT 覆盖网进行交互,因此它不依赖任何特殊的 DHT 覆盖网。DST 方式适用于任何提供绘图函数的下层重叠网。这些工作是我们努力建立支持知识网络应用的分布式语义覆盖网的一部分。

参考文献:

[1] 何　丽,韩文秀. 一种基于后缀树的 Web 访问模式挖掘算法[J]. 计算机应用, 2004, 24(11):68-70.

[2] Steinmets R, Wehrle K. P2P 系统及其应用[M]. 北京:机械工业出版社,2008:80-85.

[3] Wen Yuanfeng. A Novel Distributed Index Approach for Service Discovery in MANETs[C]//IEEE International Conference on Parallel and Distributed Systems. [s. l.]:[s. n.], 2008:415-421.

[4] Jiang Yan. A Suffix Tree Based Handwritten Chinese Address Recognition System[J]. IEEE, 2007,3(1): 292 - 296.

[5] Kale A. A New Suffix Tree Similarity Measure and Labeling for Web Search Results Clustering [J]. Engineering and Technology, 2009, 24(19):856-860.

[6] 包小源,宋再生,唐世渭,等. SuffIndex——一种基于后缀树的 XML 索引结构[J]. 计算机研究与发展,2004, 41(10):1793-1799.

[7] 吴国庆. 对等网络技术研究[J]. 计算机技术与发展, 2006, 16(7):100-103.

[8] 张建宇,马　皓,廖唯浆,等. P2P 系统测量关键技术研究[C]//全国网络与信息安全技术研讨会论文集(下册),全国网络与信息安全技术研讨会. 中国山东青岛,2007:619-625.

[9] 冯冰洁,杨天奇. 后缀树聚类算法在元搜索引擎中的应用[J]. 微计算机信息, 2010, 27(11):1-3.

[10] 葛　健,王国仁,于　戈. 后缀树的并行构造算法[J]. 计算机科学,2004, 31(5):96-99.

[11] 曾本胜,廉玉忠,李世取. 后缀树的平均高度[J]. 运筹与管理,1996, 8(4):9-13.

[12] Harren M, Hellerstein J M, Huebsch R, et al. Complex Queries in DHT-Based Peer-to-Peer Networks[C]//Proc. First Int'l Workshop Peer-to-Peer Systems, Lecture Notes in Computer Science, US. (IPTPS '02). [s. l.]:[s. n.], 2002:242-250.

[13] 孙名松,刘　杰,李胜利. 基于语义划分的 P2P 搜索技术研究[J]. 计算机技术与发展,2010, 20(8):67-71.

(上接第 30 页)

法。实践证明用该方法设计 Tab 面板是可行的,该方法简单实用,为网站的页面节约了空间,减轻了服务器的负担。

参考文献:

[1] 三扬科技飞思科技产品研发中心. 大道 PHP 开发与实战[M]. 北京:电子工业出版社,2009.

[2] Jensen S H, Moller A. Type Analysis for JavaScript[J]. STATIC ANALYSIS,2009,5673:238-255.

[3] 温　谦. CSS 设计彻底研究[M]. 北京:人民邮电出版社, 2008.

[4] Wilton P, McPeak J. JavaScript 入门经典[M]. 第 3 版. 北京:清华大学出版社,2009.

[5] 胡孟杰. JavaScript 动态网页开发案例指导[M]. 北京:电子工业出版社,2009.

[6] 李　烨. 别具光芒 DIV+CSS 网页布局与美化[M]. 北京:人民邮电出版社,2007.

[7] 郑俊生,姜　敏. 一种基于 Div+CSS+JavaScript 的网页布局特效研究[J]. 电脑知识与技术,2008(17):1556-1558.

[8] 周利荣 基于 JavaScript 的动态网页制作[J]. 电脑学习, 2005(3):11-13.

[9] 赵清华. 基于 DIV+CSS 的网页布局技术应用研究[J]. 现代计算机,2010(5):140-142.

[10] 胡　波. JavaScript 的动态网页制作探讨[J]. 软件导刊, 2006(9):71-73.

[11] 熊　迪. 结合 CSS 和 JavaScript 技术实现动态选项卡菜单[J]. 湖北广播电视大学学报,2009,29(2):155-156.

[12] 周广田. 用 JavaScript 创建交互式 web 页面[J]. 信息技术,2001(2):35-36.