一种基于 Peer-to-Peer 技术的 Web 缓存共享系统研究

凌 波[1)] 王晓宇[2)] 周傲英[2)] Ng Wee-Siong[3)]

[1)](中国浦东干部学院 上海 200233)

[2)](复旦大学计算机科学与工程系 上海 200433)

[3)](新加坡国立大学计算机科学系 新加坡 117576)

摘 要 提出了一种基于 peer-to-peer 技术的分布式 Web 缓存共享系统:BuddyWeb.该系统的核心理念是让企业网络中的所有 PC 能够相互共享浏览器中的本地缓存,从而形成一个高效的、大规模的分布式缓存共享系统,并使系统具备易管理、易实现、低成本等优点;接着详细阐述了 BuddyWeb 的工作原理和算法策略;然后,针对 BuddyWeb 系统的特性提出了仿真实验模型和评估方法.实验结果证明了 BuddyWeb 在命中率、网络通信流量负荷、系统响应延迟等诸方面均能取得令人满意的效果.

关键词 peer-to-peer Web 缓存;动态自配置;自适应跳步

中图法分类号 TP303

A Collaborative Web Caching System Based on Peer-to-Peer Architecture

LING Bo[1)] WANG Xiao-Yu[2)] ZHOU Ao-Ying[2)] Ng Wee-Siong[3)]

[1)](*China Executive Leadership Academy Pudong*, *Shanghai* 200233)

[2)](*Department of Computer Science & Engineering*, *Fudan University*, *Shanghai* 200433)

[3)](*Department of Computer Science*, *National University of Singapore*, *Singapore* 117576)

Abstract In this paper, a collaborative P2P-based Web caching system, named BuddyWeb, has been proposed, whose underlying ideology is that all the PCs in an Intranet are able to share their local caching to constitute a large-scale, effective collaborative Web caching system. BuddyWeb distinguishes itself from others with the advantages of scalability, effectiveness and low cost. Furthermore, the working mechanism and the algorithm of the system have also been detailed. In addition, a simulation model has been devised to evaluate such a system. And the evaluation results show BuddyWeb achieves a satisfied performance in hits-ratio, traffic-load and latency of response.

Keywords peer-to-peer Web caching; dynamic self-reconfiguration; self-adaptive hopping

1 引 言

Peer-to-Peer (简称 P2P)计算模型正越来越广泛地应用于数据挖掘、资源交换、数据管理、文件共享等领域.P2P 技术得以风行的最初动机是人们希望创建属于自己的在线通信通道,以实时地访问和交换信息,典型的 P2P 系统有 Napster 和 Gnutella 等.本文提出了一种新型的 P2P 应用:Web 缓存共享.不同于以往基于 proxy 端的 Web 缓存技术,本文着眼于

收稿日期:2003-04-24;修改稿收到日期:2004-10-08.**凌 波**,男,1974 年生,博士,主要研究兴趣包括对等计算、基于对等计算的数据管理和信息检索、信息经济和领导科学等.E-mail: bling@celap.org.cn.**王晓宇**,男,1975 年生,博士,主要研究兴趣包括 Web 数据管理、对等计算和嵌入式系统等.**周傲英**,男,1965 年生,博士,教授,博士生导师,主要研究兴趣包括 Web 数据管理、数据挖掘、数据流管理与分析、对等计算、金融数据管理与分析等.**Ng Wee-Siong**,男,1973 年生,马来西亚人,2004 年在新加坡国立大学获得计算机科学专业理学博士学位,主要研究兴趣包括数据库性能、对等计算和基于 Internet 的应用等.

如何在一个企业局域网环境中利用所有节点(PC)(浏览器中)的本地缓存.

企业环境 P2P 模型与互联网资源共享模型的主要区别为：(1)连结节点(peer)间的带宽不同，企业局域网的带宽比广域网带宽高得多；(2)企业局域网的安全性好，因为局域网中的所有节点都受防火墙保护；(3)企业环境中整个网络都是可控制和可管理的. 为了更好地阐述本文的研究背景和动机，让我们考虑复旦大学校园网这样一个企业级局域网. 在这类网络中，有成千台 PC 相互连接在一起，每台 PC 都有一个 Web 浏览器. 这里，防火墙把校园网与“外面的世界”隔离开，任何进入和外出的请求都要通过一个中心 proxy；另外，中心 proxy 还承担数据流量“计费器”的角色，每个通过它出入的字节都要收费.

在当前的计算模式下，用户不能相互共享节点浏览器中的缓存内容，即使正查找的信息已缓存在校园网(LAN)内的其它节点，也不能在 LAN 内获得. 所有查询不得不通过中心 proxy 发送到远程服务器，这样既导致了较长的响应时间，又增加了额外的费用. 相反，如果能利用 P2P 技术使 LAN 内所有 PC 共享本地缓存，则不但能缩短响应时间，而且能节省费用.

带着这样的动机，本文设计了基于 P2P 技术的 Web 缓存共享系统：BuddyWeb. 在 BuddyWeb 中，所有参与节点的本地缓存都可共享，在触发外部(到 Intranet 之外的)访问之前，首先搜索 LAN 内节点的缓存. 总之，BuddyWeb 具有以下独特性：

(1)网络的拓扑结构可以依据节点的兴趣而自调整. 运行一段时间后，网络中将形成不同兴趣的虚拟社区. 例如，数据库研究社区，生物信息研究社区等.

(2)采用了基于节点缓存内容相似度的路由策略. 基于相似度的判断，查询将从一个节点路由到和该节点具有较高相似度的相邻节点.

(3)实现了基于兴趣相似度的自适应跳步策略(self-adaptive hopping strategy)，每个查询消息的 TTL 值(Time-To-Live，生命周期)会自动调整，从而实现最大化搜索结果和最小化带宽消耗的优化目标.

本文第 2 节介绍 BuddyWeb 的体系结构，首先回顾实现系统的 P2P 系统平台 BestPeer，然后描述 BuddyWeb 节点的体系结构；第 3 节介绍 BuddyWeb 的特性和算法，包括基于兴趣相似度的动态自配置策略以及基于兴趣相似度的路由策略与自适应跳步机制；第 4 节提出仿真实验模型并展示实验结果；第 5 节讨论相关的工作；第 6 节总结全文.

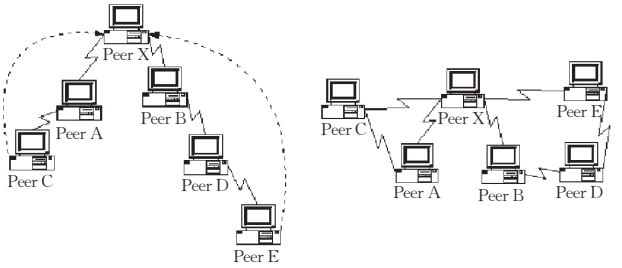
2　BuddyWeb 系统体系结构

本节首先介绍实现 BuddyWeb 系统的 P2P 平台，以更好地理解系统的工作机制，然后详细描述 BuddyWeb 节点的体系结构和工作流程.

2.1　BestPeer 平台

BuddyWeb 是在 BestPeer[1] 平台上实现的. BestPeer 是集成了移动 Agent 技术的通用 P2P 平台，可以在它上面有效地开发各种 P2P 应用. 平台系统由两类实体组成：大量的对等节点和少量的位置独立全局名查找服务器(LIGLO). 每个对等节点都运行一个基于 Java 的 BestPeer 软件，并能互相通信和共享资源. LIGLO 服务器是有固定 IP 地址并运行着 LIGLO 软件的特殊节点，主要有两个功能：(1)为每个对等节点分配全局唯一的标识符(BPID)，即使一个节点以不同 IP 地址登录，也能被唯一地识别出来；(2)维护自己所管辖的对等节点的当前状态信息，包括在线与否及当前 IP 地址等元数据.

在传统的 P2P 系统(如 Gnutella)中，邻居节点(直接相连的节点)一般是人工静态指定或随机决定的，而 BestPeer 的节点可以动态地重新配置自己的邻居. 该机制基于以下简单假设：对于任意一个给定的节点，曾经对它有益的节点，在以后的查询中很可能仍然有益. 因此，BestPeer 的节点会把曾经对自己最有益的节点维护成直接邻居；由于自身资源的限制，每个节点的邻居数量受限. 图 1 展示了节点重新配置邻居的过程. 在图 1(a)中，Peer X 发出的请求直接发送给 Peer B 和 Peer A. 但是，只有 Peer C 和 Peer E 才有 Peer X 当前请求的对象. 那么 Peer X 从 Peer C 和 Peer E 获得结果；并且，Peer X 发现虽然 Peer C 和 Peer E 对它有益，却不是直接邻居，因此，Peer X 重新配置自己的邻居节点，将 Peer C 和 Peer E 节点加入自己的邻居列表中，结果网络拓扑



(a) 配置前　　(b) 配置后

图 1　BestPeer 节点自配置过程

结构如图 1(b).

自配置策略的实质就是将最有益的节点保持在近邻位置上,以便为自己提供更好的服务. BestPeer 支持两种缺省动态自配置策略[1]:(1)MaxCount,以提供有效查询结果数最大化为基准;(2)MinHop,以获得相同查询结果所历经的跳数(hop)最少为基准.

2.2 BuddyWeb 节点体系结构

BuddyWeb 节点的体系结构如图 2 所示. Web 浏览器充当前端用户界面,采用 Microsoft Internet Explorer (IE). 在 BuddyWeb 支持的浏览器中由一个本地 proxy 来操纵本地 Web 缓存;用一个 HTTP 后台线程支持 HTTP 请求. 底层数据通信由 BestPeer 平台管理. BuddyWeb 节点正是借助于 BestPeer 平台与其它节点共享缓存内容. 对用户而言,这种设置是透明的,用户觉察不到 BuddyWeb 浏览器和普通浏览器的差别.

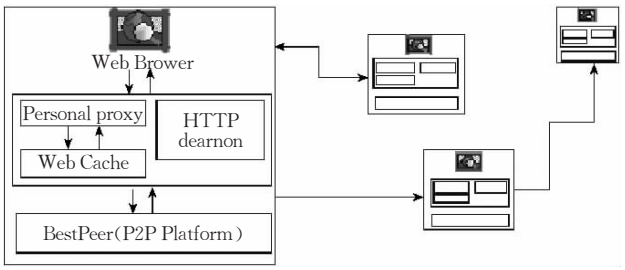


图 2 BuddyWeb 节点体系结构

当用户从浏览器提交一个 URL 查询,本地 proxy 将其改写成 BestPeer 平台可以接受的输入形式并传送给底层 BestPeer 平台. 接着,BestPeer 产生一个移动 Agent 并将其派到 BuddyWeb 网络中搜寻匹配结果.

一旦查找到匹配结果,BestPeer 将结果的位置等元数据返回给发出查询的本地 proxy;本地 proxy 立刻使 HTTP 后台线程直接向存有匹配文档的节点发出 HTTP 连接请求;一旦收到 HTTP 连接请求,被请求节点上的 HTTP 后台线程立即处理该请求,并传送被请求的结果.

3 BuddyWeb 的特性与算法

本节详细介绍 BuddyWeb 的特性与算法,包括基于兴趣相似度的自配置策略、基于兴趣相似度路由策略和自适应跳步,并展示了 LIGLO 和对等节点的工作算法.

3.1 基于兴趣相似度的动态自配置策略

由于 BestPeer 的缺省自配置策略是基于查询行为的,不能有效利用共享内容的语义信息. 在 BuddyWeb 中,我们提出了一种基于兴趣(或内容语义)相似度的动态配置策略. 该策略充分利用了 BestPeer 中的 LIGLO 服务器. 即每个注册的节点不仅要将当前 IP 地址传送给管辖它的 LIGLO 服务器,同时还要定期向 LIGLO 发送关于自己浏览兴趣的信息.

节点浏览兴趣是通过它浏览过的网页信息来提取的. 虽然具体方法很多,但抽取的浏览兴趣信息必须同时满足代表性和简练性的要求. BuddyWeb 通过节点浏览过网页的部分元数据(meta-data,例如〈TITLE〉〈/TITLE〉)来定义节点的浏览兴趣. 每个节点的浏览兴趣会在管辖它的 LIGLO 服务器用一组词列表的形式来表示. 本文提出的动态自配置策略就是基于这些词列表进行的. 我们可以把所有词列表中的词看作一个词袋(wordbag),并用这个词袋构造一个向量空间. 基于向量空间,每个词列表可以依据某种权重方案映射为该向量空间中的向量,最简单的权重方案是布尔权重. 这样,每个节点的浏览兴趣就由向量表示. 由于 BuddyWeb 有多个 LIGLO 服务器,每个 LIGLO 服务器中仅保存了在它登记的节点浏览兴趣词列表. 因此,所有 LIGLO 必须相互协商,从而决定由谁接收其它 LIGLO 所保存的兴趣词列表信息. 解决该问题的另一种方法是通过哈希影射的方法,这样可以避免所有的向量计算在一个服务器上进行. 但由于在企业网环境中,BuddyWeb 的 LIGLO 服务器数量有限,而且 LIGLO 服务器间传输的词列表只是一些“轻量级”文件,因此采用协商方法.

基于上述定义的向量空间,节点可以采用余弦相似度函数来计算它们之间的兴趣相似度. 为了使浏览兴趣最相近的节点直接连接在一起,每个节点都仅将跟自己具有最高相似度的数个节点维护为邻居. 当网络中很多节点时,计算节点浏览兴趣向量的两两相似度非常耗时. 解决这个问题的简单方法是将相似度的计算分布到每个节点. 这样,负责计算的 LIGLO 服务器只要计算向量空间及每个节点在该空间中的向量. 计算完成后,这些向量会传发网络中的所有 LIGLO 服务器. 基于兴趣相似度的动态自配置的具体过程如下:

(1)经过一个指定的时间周期(如一天)后,LIGLO在特定时间(如午夜)相互协商并推举一台 LIGLO 计算向量空间和所有登记节点的向量.

(2)当一个节点登陆时,首先跟它登记的 LIGLO 通信,上传其当前 IP 地址,并接收当前网络中所有

节点(包括该登陆节点本身)的浏览兴趣向量.该节点与其它节点间的两两相似度就在登陆节点本地计算出来,并按由高到低排序.

(3)k 个具有最高相似度的节点将保持为直接连接邻居,k 是一个指定的系统参数.

采用基于兴趣的自配置策略,BuddyWeb 系统运行一段时间后,会形成特定的(节点)虚拟社区.

3.2　基于兴趣相似度的路由与自适应跳步

目前,大多数 P2P 系统(如 Gnutella)的查询采用广播路由策略,即查询节点把查询广播给所有邻居,邻居收到查询后在本地检索的同时,继续把查询转发给它们的所有邻居,直到查询转发次数达到限制值(TTL).这种广播路由策略导致了很高的网络通信流量负担.

在 BuddyWeb 中,每个节点都保存自配置过程中计算得到的节点间兴趣相似度值.由于该值实际上是节点缓存内容的相似度,因而,当节点转发查询时,只需转发至和该节点具有很高相似度值的邻居,而不是转发给所有邻居.注意,查询提交节点将查询传播给所有邻居.

利用节点间的兴趣相似性度,BuddyWeb 还实现了一种自适应查询跳步策略.以往的 P2P 系统(包括 BestPeer)都预先设定查询跳数(TTL 值).如果 TTL 值设得太小,那么查询处理被限制在很小范围,查全率低;反之,如果 TTL 设得过大,网络的通信流量负担非常沉重.因此,不可能事先为不同的查询设定一个相同的 TTL 值,并同时达到上面两种考虑的折衷.

BuddyWeb 充分利用了 BestPeer 提供的 Agent 机制,所有查询被封装在 Agent 中,Agent 在路由途中记录一些"历史"信息.为了更直观地阐述自适应跳步策略,我们用距离(跟相似度本质相同)来描述节点间的关系.在网络中,两节点间最长的距离被看作 BuddyWeb 概念空间的直径,即所有节点的兴趣所覆盖范围.自适应跳步策略的具体工作过程如下:

(1)节点发起一个携带一个参数 s(而不是 TTL 定值)的查询 Agent.这里参数 s 是一个由 P2P 网络预先设定值,称为概念空间参数.网络直径用 D 表示,其值在动态自配置的过程中计算出.

(2)当查询 Agent 被转发到邻居节点,Agent 记录下当前节点与其邻近节点间的"距离".并且,该距离跟以前路由路径上的距离值累加.

(3)如果累加值超过 $s \cdot D$ 的值,那么查询路由终止(查询 Agent 不再被转发).否则,节点继续将查询 Agent 向其直接相连的节点转发.

采用该策略,每个查询的 TTL 会依据它所搜索的概念空间范围而自适应调整.概念空间参数 s 反映了系统希望节点在概念空间中搜索的范围.通过设定参数 s 可以使 BuddyWeb 在通信量和查全率两方面动态实现折衷.

BuddyWeb 系统中的 LIGLO 服务器和对等节点的工作算法分别如算法 1 与算法 2 所示.

算法 1.　LIGLO 工作算法.

```
For LIGLO server:
void ProcessPeerLogin()      //等待 Peer 登陆
{
  while(true)
  {
    if(AnyPeerRequestToLogin())
                              //是否有 Peer 请求登陆
    {
      AcceptPeerRequest(); //接受登陆请求
      GetPeerIP();         //获得登陆节点 IP
      SendCurrentRegisteredPeersInterest();
      //将当前所有登陆节点的兴趣向量发给请求节点
    }
    else
    {
     Sleep(WAIT_INTERVAL);
    }
  }
}
```

算法 2.　对等节点的工作算法.

```
For each peer:
bool Login ()
{
  bool bRet=ConnectToLIGLOServer();
            //请求连接到 LIGLO 服务器
  if(bRet)
  {
    SendCurrentIPAddress();
              //将当前 IP 地址发送给 LIGLO 服务器
    GetRegisteredPeerInterests();
              //获得当前登记的所有节点的兴趣向量
    ComputeSimlarities(); //计算同其它节点的相似度
    RankSimilarities();     //对兴趣相似度排序
    ConnetToKNeighbors(); //同相似度最高的 k 个
                          节点建立连接
  }
  return bRet;
}
//发起查询
void Query(StringList strKeys,float s)
          //StringList 为关键字组,s 为概念空间参数
```

```
{
  int D=GetCurrentNetworkDiameter();
        //获得或者计算当前网络的近似直径
  int TTL=D×s;
  SendCurrentQueryToNeibs(strKeys,TTL,0);
        //将当前查询发送到邻居节点开始查询,0
          表示当前查询经过的距离为 0
  int iTotalWaitTime=0;
  while(iTotalWaitTime<MAX_WAIT_TIME)
  {
    GotQueryResult(); //取得当前返回的查询结果
    DisplayResults();  //将查询结果显示给用户
    Sleep(WAIT_INTERVAL);
    iTotalWaitTime+=WAIT_INTERVAL;
  }
}
//收到查询请求的节点处理查询
void ProcessQuery(StringList strKeys, int TTL, int
curSteps)
{
  ResultInfo result=SearchInThisPeer(strKeys);
                    //在当前节点上查找查询结果
  ReturnResultsToInitialPeer(result);
                    //将查询结果返回给查询节点
  if (curSteps<TTL)
  SendCurrentQueryToNeibs(strKeys,TTL,curSteps+1);
        //将当前查询发送到邻居节点开始查询
}
```

4 系统评估

4.1 仿真评估模型

Web 缓存系统的性能很大程度上取决于实际环境的工作负荷.因此,为了比较客观地评估 BuddyWeb 系统算法的有效性,本文采用仿真模拟的方法,并建立一个可控的仿真环境.并且,可假设 BuddyWeb 节点所请求的都是静态可缓存对象.虽然这和实际环境存在差异,但由于不可缓存的对象对于任何缓存方法的影响都是同样的(HTTP 请求都被直接发至原始站点),所以并不影响仿真的有效性.实际上,仿真实验的目的是验证 BuddyWeb 在一般情况下的行为特征,而不是具体模拟某种现实的 Web 缓存实例.

4.1.1 仿真模型初始数据集的构造

假设在某段时间内 BuddyWeb 所有节点所请求的 Internet 对象集合为 $Q=\{q_1,q_2,\cdots,q_n\}$,被定义为仿真试验对象全集.设对象集合 Q 一共包含 u 个不同的主题,每个主题的对象集合用 QT 表示.为简化仿真过程,可假设不同主题的对象集合间不存在交集,即 $QT_1\cup QT_2\cup\cdots\cup QT_u=Q$;$QT_i\cap QT_j=\varnothing$,其中 $i,j\in\{1,2,\cdots,u\}$.

因为仿真过程并不需要真实存在可缓存对象集合 Q,因而可用一组映射关系来模拟从对象的元数据中抽取关键词的过程.设从每个主题集中的对象的元数据中抽取的关键词集为 T,那么,对象主题集 QT 与其相应的关键词集 T 之间的抽取关系可以通过一个映射来表达,$F(QT_i)\rightarrow T_i$,其中 $i\in\{1,2,\cdots,u\}$.通过映射 F 可以为每个主题对象集 QT 与其相应的关键词集 T 之间建立一个一一对应的关系.为简化仿真过程,可设每个关键词集 T 中含有同样数目的关键词 m 个,且 $T_i\cap T_j=\varnothing$,其中 $i,j\in\{1,2,\cdots,u\}$.

另外,对于 Internet 对象实验全集 Q 中的对象,我们将为其中每一个对象的大小分配一个值.其分布符合在[1K,1M]上的正态分布.

4.1.2 BuddyWeb 仿真网络节点的构造

创建一个 BuddyWeb 的仿真网络,设网络一共有 w 个节点,每个节点由一个 BuddyWeb 本地 proxy 操纵该节点的缓存、发送 HTTP 请求及响应其它节点的 HTTP 请求.每个节点的本地缓存能保存 k 兆的可缓存静态对象.在仿真实验之前,首先为每个节点初始化本地缓存内容,具体过程如下:从 u 个主题对象集合$\{QT_1,QT_2,\cdots,QT_u\}$中随机地挑选 $1\sim p$ 个主题集合(p 为仿真实验参数,称为最大主题浏览数).然后从挑中的主体集合中随机选取其中的静态可缓存对象,直至达到该节点本地缓存容量,即 kMB.

BuddyWeb 仿真网络的每个节点在选择了 $x(1\leq x\leq p)$个主题对象集合后,就依据映射 F 得到 x 个相应的关键词集合.按照该节点从这 x 个不同的主题对象集合中随机选取的对象数目的比例,可从相应的 x 个关键词集合中随机地挑选出 q 个不同的关键词构成该节点的浏览兴趣词列表.如果该节点的本地缓存中存有 y 个对象,则 $q=\Delta\cdot y$,Δ 是仿真实验参数,该参数定义为节点浏览兴趣词列表生成系数.

4.1.3 BuddyWeb 网络行为的模拟

为了比较全面模拟 BuddyWeb 网络的实际运作过程,仿真模型需要确定以下两个因素:(1)每个节点发出的 HTTP 所请求的对象;(2)每个节点发出 HTTP 请求的时间间隔.

设节点 i 为 BuddyWeb 中的任意一个节点,它本地缓存中的 Internet 对象所涉及的主题对象集合的并集为 NQT_i.节点的浏览过程(发出 HTTP 请求的过程)直接受到浏览者浏览兴趣的影响.浏览者既会保持一定的浏览兴趣连续性(继续浏览和其缓

存中对象主题相同的主题)，也会浏览一些以前没有浏览过的主题，后一种现象称为浏览者的浏览兴趣漂移.因此，仿真模型引入了浏览兴趣飘移系数 μ 来刻画这种现象.$\mu=1$ 表示节点浏览的 Internet 对象和它本地缓存中的对象完全没有重合的主题，$\mu=0$ 表示节点浏览的 Internet 对象和它本地缓存中对象的主题完全重合.节点 i 所产生的 HTTP 请求对象有 $100\mu\%$ 从集合 NQT_i 中随机地产生，有 $1-100\mu\%$ 从 Q－NQT$_i$ 中随机产生.

节点产生 HTTP 请求的时间间隔直接影响整个 BuddyWeb 网络中通信流量的负荷.如果节点发出请求的时间间隔很短，那么网络的信息流量负荷就会很重，否则，流量负荷就很轻.为此，仿真模型引入请求间隔参数 t，节点产生 HTTP 请求的时间间隔将从[0，t]范围内随机地产生一个值来确定.由于实验对象全集是所有网络中的节点所发出的 HTTP 请求对象，因此实验结束时，实验对象全集 Q 中所有对象都被仿真网络的节点请求过.

最后，因为每个节点的缓存容量都是有限的，所以仿真实验还要为网络节点所缓存的对象确定替换策略，本研究采用 LRU 缓存替换策略.

表 1 展示了仿真模型参数的预设定值.

表 1　仿真模型参数缺省值

仿真模型参数	代表符号	设定值
BuddyWeb 网络中的节点数目	w	2000
实验对象全集 Q 中的对象数目	n	1000000
实验对象中包含的主题数目	u	20
每个节点本地缓存中包括的最大主题数目	p	8
节点浏览兴趣词列表生成系数	Δ	3
每个主题关键词集合中含有的关键词数目	m	500

表 1 中未被列出的参数为评估实验参变量，用于研究 BuddyWeb 网络的行为特征.参变量的具体设定将在评估实验过程中详细说明.

4.2　仿真实验评估

本节评估 BuddyWeb 系统的基于相似度的动态自配置策略以及路由与自适应动态跳步方法的有效性.实验评估主要基于以下度量指标：系统性能(外部带宽或命中率，系统响应延迟或跳步数)和网络通信流量负荷.实验比较了加入动态自配置策略的 BuddyWeb 系统和没有自配置的策略的静态 BuddyWeb 系统，评估了自适应动态跳步算法对系统性能和网络通信负荷的影响.实验结果表明了基于兴趣相似度的动态自配置策略的优越性；同时也证明了自适应动态跳步算法在没有牺牲系统性能的前提下，有效地降低了网络的通信流量负荷.

4.2.1　命中率与外部带宽

命中率定义为网络中所有节点发出的请求被 BuddyWeb 网络中缓存所响应的百分比.本小节首先研究基于相似度的动态自配置策略对 BuddyWeb 命中率的影响以及节点缓存对提高整个系统命中率的贡献；接着研究浏览者行为(不同的浏览兴趣漂移系数)对整个系统的命中率的影响；最后比较自适应跳步算法与路对系统命中率的影响.

未加入动态自配置策略的 BuddyWeb 系统等同于一个传统的静态 P2P 系统，称为静态 BuddyWeb 系统(简称 SBW)；采用了动态自配置策略的 BuddyWeb 系统称为动态 BuddyWeb 系统(简称 DBW).为了研究自适应跳步算法与路由策略的影响，动态 BuddyWeb 系统分为加入自适应跳步算法与路由策略的动态 BuddyWeb 系统(简称 ADBW)和未加入的动态 BuddyWeb 系统(DBW).DBW 采用传统 P2P 系统的广播策略，每个消息的 $TTL=7$.

图 3 横坐标为每个节点的本地缓存的容量，纵坐标为系统的命中率.实验结果表明采用动态配置策略的 BuddyWeb 系统(DBW)的命中率明显高于静态 BuddyWeb 系统(SBW).并且，随着本地缓存容量的增加，DBW 越来越优于 SBW.这充分说明了动态自配置策略对有效利用本地缓存的作用.

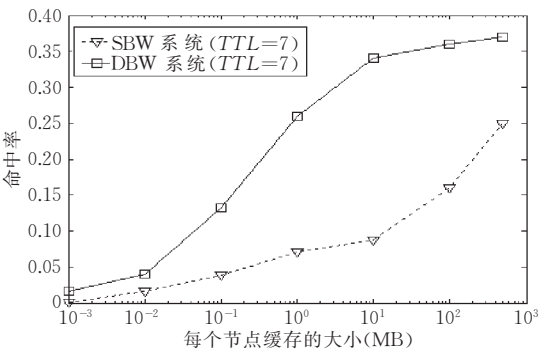


图 3　加入动态自配置策略前后系统的命中率

另外，由图 3 可知，在较为合理的缓存容量上(例如 100MB)，动态自配置的 BuddyWeb 系统能获得满意的命中率，这说明有基于兴趣相似的动态自配置策略的 BuddyWeb 系统能有效利用网络中每个节点的本地缓存.所以随着每个本地节点本地缓存的增加，命中率得到了极大的提高.注意到水平坐标的刻度是对数级的.这也说明即使每个节点的本地缓存容量很小，大量节点的协作能够取得较高较好的命中率.

图 4 展示了在不同浏览兴趣漂移系数下，BuddyWeb 的命中率(节点缓存容量为 100MB).

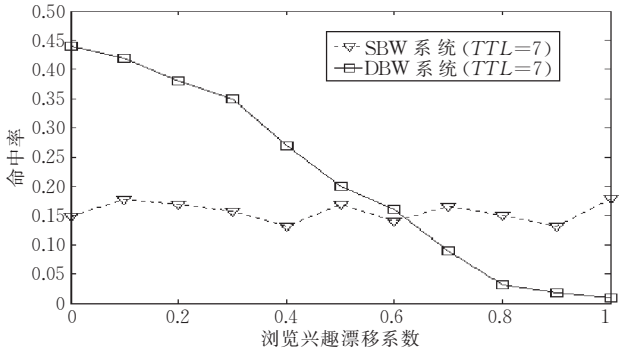


图 4 不同浏览兴趣漂移系数下的命中率

由图 4 可见,无动态自配置策略的 BuddyWeb 系统几乎不受浏览漂移系数变化的影响,而动态自配置系统则随着浏览漂移系数增加,系统的命中率呈下降趋势,并且浏览漂移系数超过 0.6 时,动态自配置策略下的 BuddyWeb 系统的命中率开始低于不采用自配置策略的系统.

接下来评价自适应跳步算法与路由策略.加入自适应跳步算法与路由算法后,BuddyWeb 可以依据预设定的概念空间参数 s 自适应地决定跳步的数目.实验中,$s=0.7$.

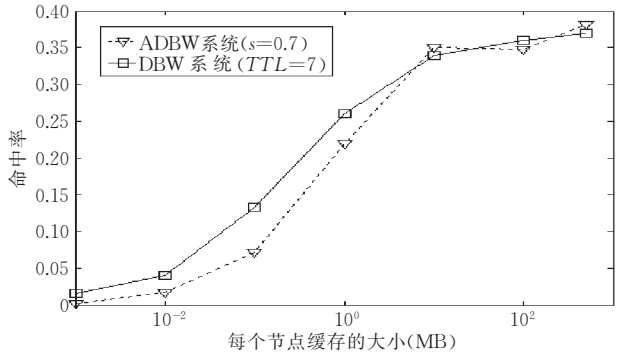


图 5 加入自适应跳步算法与路由策略前后 DBW 系统的命中率

由图 5 可见在较小的节点本地缓存容量下,ADBW 系统的命中率较低于 DBW 系统.当节点本地缓存达到或超过 10MB 时,ADBW 系统的命中率开始接近甚至优于 DBW 系统.该结果说明了自适应跳步算法与路由策略在本地缓存达到一定的容量时,不会降低系统命中率,但有效地下降了整个网络的通信流量负荷(见 4.2.2 节).

现在研究概念空间参数 s 对于系统命中率的影响,实验结果如图 6 所示.

由图 6 可知,随着概念空间参数 s 的增加,系统的命中率得到了显著的提高,但 s 超过 0.7 时,它对系统命中率的提高作用开始减缓.但随着 s 的增大,网络的通信流量负荷也会相应地增加.4.2.2 节将

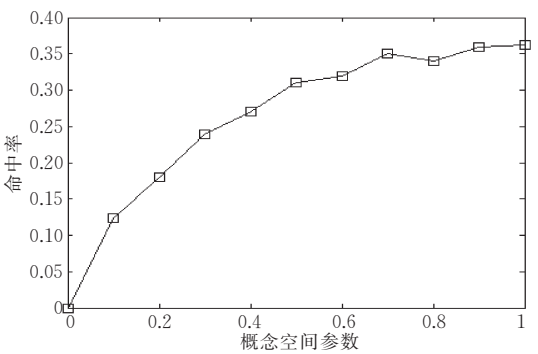


图 6 概念空间参数 s 在 ADBW 系统中对于命中率的影响

会进一步研究概念空间参数 s 对网络流量负荷的影响.并且,由上述实验结果可知,BuddyWeb 系统的缓存命中率已经和集中式的 proxy 缓存系统相当.

4.2.2 网络通信流量负荷

网络通信流量负荷定义为单位时间内 BuddyWeb 网络中传输字节数量的总和.表 2 列出了估算通信流量参数.

表 2 通信流量相关信息字节数的估计值

请求头(Request headers)信息字节数	350 字节(估计值)
响应头(Response headers)信息字节数	150 字节(估计值)
传输的对象字节数	在实验对象全集中相应对象的大小

接下来研究自适应跳步算法与路由策略对减轻 BuddyWeb 网络通信流量负荷的作用以及参数 s 对网络通信流量负荷的影响.

图 7 给出了在不同的请求间隔参数 t 下,ADBW 与 DBW 系统在单位时间内的通信流量.实验随机地选取了 10 个单位采样时间段,每个采样时间段设为 1min.实验中节点本地缓存容量设为 100MB,DBW 的 $TTL=7$,ADBW 系统的概念空间参数 s 设为 0.7.

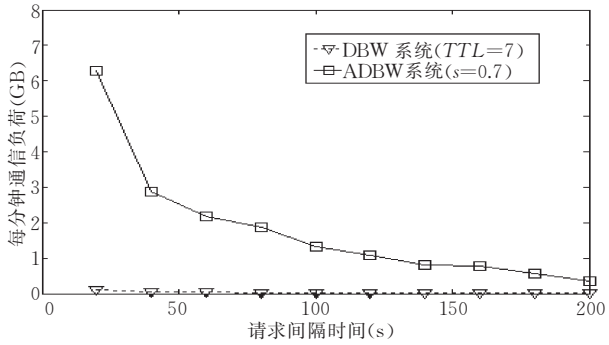


图 7 不同请求间隔参数下,BuddyWeb 网络通信流量负荷

由图 7 可知,采用自适应跳步算法和路由策略的 BuddyWeb 系统的网络通信流量负荷得到了极大的改善,而传统的 P2P 系统的广播式路由机制导

致网络通信流量负荷呈指数式增长，并且当请求间隔参数较小时，网络的流量负荷高得惊人. 可见，自适应跳步算法和路由策略在不牺牲缓存系统命中率的前提下，极大地降低了网络通信流量负荷.

图 8 研究了概念空间参数对 ADBW 系统的网络通信流量负荷的影响.

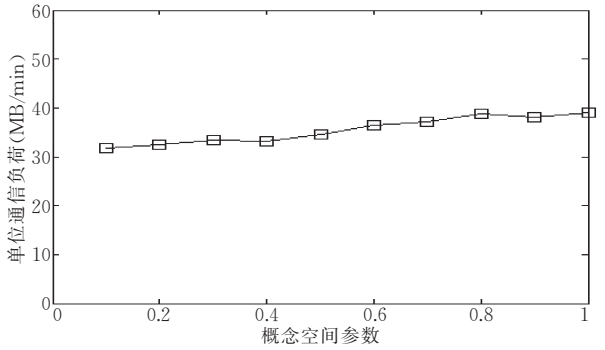


图 8　不同概念空间参数下 ADBW 网络的通信流量负荷

由图 8 可见，随着概念空间参数的加大，网络的通信流量负荷会有轻微的增加. 这表明搜索半径的增加所导致的流量负荷增加是非常有限的.

4.2.3　响应延迟与跳步数

设局域网内部的通信时间为毫秒级，与外部的通信则为秒级，节点内的处理时间也为毫秒级(不包括传输时间). 4.2.1 节的实验结果已经表明 BuddyWeb 的缓存命中率已经和集中式的 proxy 缓存系统相当. 集中式缓存系统在局域网内的跳数为 2，即请求和响应. 未采用自适应跳步算法与路由策略的系统的 TTL 为固定值，设为 7.

图 9 表明在采用了自适应跳步策算法和路由策略后，BuddyWeb 系统内所有查询的跳步数在 1～6 之间，大部分查询的跳步数在 2～3 之间，最大的跳步数也没有超过 6. 用户发出的请求在局域网内的缓存被命中时，响应的延迟是很短的，因而系统的响应延迟也很短. 而且，对于大的 Internet 对象，较高的局域网网络带宽导致更小的响应延迟.

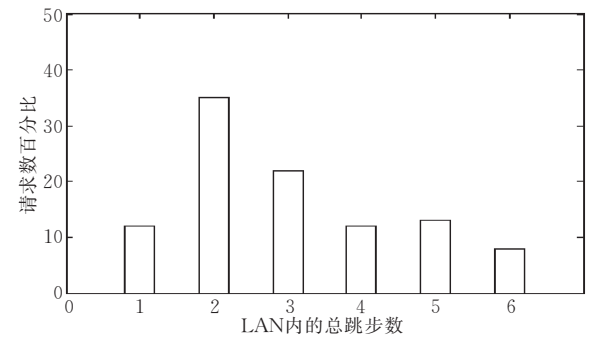


图 9　BuddyWeb 局域网内查询的跳步数

5　相关工作

BuddyWeb 系统是协作式 Web 缓存系统与 P2P 技术的结合. 在协作式 Web 缓存系统中，所有存储在缓存中的 Internet 对象为整个系统所共享[2,3]. 而 P2P 技术提供了点对点的路由机制、分布式的自组织结构以及抗单点失败的能力[4～6]等优点.

协作式的缓存方案在小的组织中对提高命中率较为有效. 该方案存在多种形式：(1)层次 Web 缓存，通常较高的层次涵盖较大的地理区域；(2)基于 Hash 的方案，通常是客户端将请求通过 Hash 发给系统中某个专用缓存；(3)基于目录的方案，通常维护一张能够重定向客户端请求的表；(4)基于组播的方案[7]. 但是，这些方案存在相当弱的集成度，导致很多问题发生在各自分离的 Web 缓存之间. 所以这些分布式的缓存方案主要应用于集中式的 Web 缓存系统中. 一组专用的 Web 代理服务器被组织成一个层次结构，有前端、后台以及管理器. 通过这种方案，协同式缓存用来实现单点失败、大规模及消除热点瓶颈等目标[8].

BuddyWeb 完全不同于上述方案，在系统中，除在客户端的浏览器透明地“嵌入”了 BuddyWeb 软件，不需要任何额外的体系结构. 就对象定位而言，BuddyWeb 采用的是 P2P 的路由方案. 它可看作是基于内容语义相似度分析的缓存方案，且完全是自组织的和分布式的. 这些特性既使 BuddyWeb 系统获得抗单点失败、大规模性等优点，又有无需任何额外附加投资的优势.

MangoSoft 的 CacheLink 产品① 和 BuddyWeb 系统一样，也支持共享浏览器的缓存对象. 关于 CacheLink 软件如何工作的细节以及其一般的性能指标目前还无法得到，但是，CacheLink 软件宣称其最大所能支持的机器数目是 250 台，相比而言，BuddyWeb 系统能够支持超过千台的节点计算机.

就分布式文件系统而言，实现方案种类繁多，而近年来对 P2P 方案的研究十分活跃，典型代表系统有 PAST[9,10]，PeerDB[11] 等，它们从不同的角度、不同的目标以及不同的折衷实现了分布式文件系统、分布式关系数据共享等应用. 从理念层面上，BuddyWeb 系统探究了一种传统文本分析技术和 Web 缓存跟 P2P 方案相结合的思路以及在相关环境中应用的可能性.

① Romine P.. LAN-Based Web caching for accelerated Web access. Mangosoft Technical White Paper, http://www.mangosoft.com/products/cachelink.

6 结 论

本文设计并评估了基于 P2P 的分布式 Web 缓存系统：BuddyWeb. BuddyWeb 在系统建设成本、规模可扩充性、抗节点失败能力以及易于管理等方面都有传统集中式系统不可比拟的优越性. 文章详细论述了 BuddyWeb 的工作原理、特性和算法，并针对系统的特性，提出了仿真实验模型. 并且，实验结果展示出该系统在命中率、通信流量负荷和响应延迟等方面都有令人满意的性能.

参 考 文 献

1 Ng WS, Ooi BC, Tan KL. BestPeer: A self-configurable peer-to-peer system. In: Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, 2002, 272

2 Bowman C. M., Danzig P. B., Hardy D. R., Manber U., Schwartz M. F.. The harvest information discovery and access system. Computer Networks and ISDN Systems, 1995, 28(1～2): 119～125

3 Wolman A., Voelker G. M., Sharma N., Cardwell N., Karlin A. R., Levy H. M.. On the scale and performance of cooperative Web proxy caching. Operating Systems Review, 1999, 34(5): 16～31

4 Ratnasamy S., Francis P., Handley M., Karp R., Shenker S.. A scalable content-addressable network. In: Proceedings of ACM SIGCOMM, San Diego, CA, 2001, 161～172

5 Rowstron A., Druschel P.. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. Lecture Notes in Computer Science 2218, 2001, 329～350

6 Stoica I., Morris R., Karger D., Kaashoek M. F., Balakrishnan H.. Chord: A scalable peer-to-peer lookup service for Internet applications. In: Proceedings of the ACM SIGCOMM, San Diego, 2001, 149～160

7 Wang J.. A survey of Web caching schemes for the Internet. ACM Computer Communication Review, 1999, 29(5): 36～46

8 Kurcewicz M., Sylwestrzak W., Wierzbicki A.. A distributed WWW cache. Computer Networks and ISDN Systems, 2001, 30(22～23): 2261～2267

9 Druschel P., Rowstron A.. PAST: A large-scale persistent peer-to-peer storage utility. In: Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII). Schoss Elmau, Germany, 2001, 65～70

10 Rowstron A., Druschel P.. Storage management and caching in PAST, A large-scale persistent peer-to-peer storage utility. In: Proceedings of the 18th ACM SOSP'01, Banff, Canada, 2001, 188～201

11 Ng WS, Ooi BC, Tan KL, Zhou A.. Peerdb: A P2P-Based system for distributed data sharing. In: Proceedings of the 19th International Conference on Data Engineering (ICDE). Bangalore, India, 2003, 633～644

12 Kalnis P., Ooi B., Papadias D., Tan K.. An adaptive peer-to-peer network for distributed caching of loap results. In: Proceedings of ACM Conference on Management of Data (ACM SIGMOD), Madison, Wisconsin, USA, 2002, 25～36.

LING Bo, born in 1974, Ph. D.. His main research interests include data management and information retrieval in peer-to-peer computing, information economics, and leadership science etc.

WANG Xiao-Yu, born in 1975, Ph. D.. His main research interests include Web data management, P2P computing, and embedded system etc.

ZHOU Ao-Ying, born in 1965, Ph. D., professor. His main research interests include Web data management, data mining, data stream management and analysis, P2P computing, and financial data management and analysis etc.

Ng Wee-Siong, born in 1973, Ph. D.. He is currently a post-doctor at Singapore-MIT Alliance, National University of Singapore. His main research interests include database performance, P2P computing and applications based on Internet etc.

Background

This paper is supported by the cooperative research project between Fudan University and National University of Singapore: "Data Management in Peer-to-Peer Computing". This project aims to address the challenges related to the data management in P2P environment, so that the potential merits of P2P are exploited to effectively manage and process data in such an environment, and to propose new P2P-based applications. Specifically, its research topics include relational data sharing and processing, data mining, information retrieval, web cache, and so on.

Although advances have been made in this field since 2000, other projects have following characters and limitations: (1) They are operating system or network oriented but not data-centered; (2) Most of them just supported semantics-free file sharing; (3) No comprehensive systems and their sub-subsystems have been proposed. On the contrary, this research manages to achieve semantic and effective data management in P2P environment and devise new applications. Indeed, this paper has addressed the challenges related to web cache in P2P computing; Furthermore, it has also proposed a P2P-based web cache system (i. e., BuddyWeb) and detailed its algorithms.