P2P 信息检索系统的查询结果排序与合并策略

凌 波[1)] 周水庚[2)] 周傲英[2)]
[1)](中国浦东干部学院信息技术部 上海 201204)
[2)](复旦大学计算机科学与工程系 上海 200433)

摘 要 基于 P2P 信息检索系统的特性,提出了一种完全分布式的查询结果排序与合并策略.首先分析当前 P2P 信息检索系统查询结果排序和合并问题的根源;接着提出一种完全分布式的查询结果排序与合并策略,包括元数据管理策略、查询结果的排序与合并的实现;然后用详细的实验证明了该策略的有效性.

关键词 P2P;信息检索;查询结果排序与合并
中图法分类号 TP18

A Strategy of Query Result Ranking and Merging for P2P Information Retrieval Systems

LING Bo[1)] ZHOU Shui-Geng[2)] ZHOU Ao-Ying[2)]
[1)](*Department of Information Technology, China Executive Leadership Academy, Shanghai* 201204)
[2)](*Department of Computer Science and Engineering, Fudan University, Shanghai* 200433)

Abstract This paper proposes a fully distributed strategy to rank and merge the results retrieved from different peers in P2P information retrieval systems. First, the challenges of query answering in P2P IR systems are investigated and the issue of retrieved results ranking and merging is identified. Second, a fully distributed strategy is developed and its implementing issues are addressed. Finally, an extensive experimental study is conducted and the results verify the effectiveness of this proposed strategy.

Keywords P2P; information retrieval; query result ranking and merging

1 引 言

自 2000 年起,对等计算(peer-to-peer,简称 P2P)倍受计算机研究界的关注.在 P2P 系统中,每个对等节点(peer,简称节点,如用 Internet 连接的 PC)都拥有对等的功能与责任,既可充当服务器向其它节点提供数据与服务,又可作为客户机享用其它节点提供的数据与服务、节点间的交互直接对等;此外,任何一个节点可随时加入或离开该系统,形成一个真正的动态网络环境.已经取得的研究成果表明,这类系统具有许多潜在优良特性,如系统可扩展性好、资源(种类与数量)丰富、性能高等,可应用于许多领域:如 CPU 周期共享①、信息传输②、协同工作组件③和数据共享[1-5]等.其中,数据共享已经成为当前 P2P 研究的热点.但已有的 P2P 数据共享系

收稿日期:2005-01-19;修改稿收到日期:2006-09-24.本课题得到国家自然科学基金(60373019,60573183 和 90612007)资助.**凌 波**,男,1974 年生,博士,副教授,目前主要研究方向为 P2P 环境下的数据管理、信息系统及区域竞争力等.E-mail: lingbo@fudan.edu.cn.**周水庚**,男,1966 年生,博士,教授,博士生导师,研究兴趣包括信息检索与文本挖掘、空间数据库与地理信息系统以及对等计算等.**周傲英**,男,1965 年生,博士,教授,博士生导师,研究兴趣包括 Web 数据管理、数据挖掘、流数据分析与处理以及对等计算等.

① SETI@home Home Page. http://setiathome.ssl.berkely.edu
② ICQ Home Page. http://www.icq.com
③ Groove Home Page. http://www.groove.net

统大都仅限于缺乏语义的粗粒度(文件水平)共享,用户通过文件名进行查找.这种不能基于语义来共享数据的机制限制了 P2P 潜能的发挥.

信息检索(IR)技术已取得很大突破,能对多种数据(文本、图像等)进行语义丰富的检索.但传统的信息检索系统存在固有的局限性,如系统缺乏可扩展性及能力(包括处理能力和存储能力)有限等.这些局限性在当今信息爆炸的时代显得更为突出,限制了 IR 的广泛应用.

为应对上述两类技术面临的挑战,研究界提出了 P2P 信息检索的理念[4,6-7],把 P2P 与信息检索相集成,充分发掘各自的优点并相互克服对方的不足.由于 P2P 信息检索技术还处于研发的初始阶段,面临不少挑战.其中,如何排序和合并从多个节点检索出来的答案就是亟待解决的难题之一.本文基于 P2P 的特性,深入分析该问题的根源,并提出了一种完全分布的检索结果排序与合并策略,取得了如下成果:

(1)通过深入分析,明确了 P2P 信息检索系统在查询结果排序与合并上所面临的挑战的根源;

(2)提出了一种完全分布的查询结果排序与合并策略,包括元数据管理策略、排序与合并的实现;

(3)进行了详细的实验分析,用实验结果证明了该策略的有效性.

本文的其余部分组织如下:第 2 节讨论相关工作;第 3 节对研究问题的根源进行理论分析;第 4 节提出分布排序与合并策略;第 5 节进行实验分析;第 6 节总结全文.

2 相关工作

P2P 信息检索系统是一种新型的分布式信息检索系统.传统的分布式信息检索系统通常根据某种划分标准(如文档语义)将整个系统的文档集划分为多个子集,并分配给不同的文件库服务器维护.虽然整个系统物理上有多个文档库,分别由不同节点(服务器)管理,但整个系统在逻辑上仍然只有一个统一的文档库[8].可见,P2P 信息检索系统的核心理念和实现机制明显区别于传统的分布式信息检索系统:(1)节点数目不同.P2P 信息检索系统可有成千上万个节点,且每个节点都是独立自治的信息检索系统.每个节点都有一个独立的文档库,系统在逻辑上有成千上万个独立的文档库.(2)文档和统计信息管理方式不同.在传统的分布式信息检索系统中,服务器之间相互备份统计信息,每个服务器节点都有全局统计信息;而在 P2P 环境中,每个节点主要管理本地文档及其统计信息,节点决策时只能借助于局部信息或全局信息近似值.(3)检索处理模式不同.传统信息检索系统有一个中心节点,维护所有文档服务器各自存储文件的特征描述文件,对于一个查询,基于该描述文件可以很好地定位查询答案所在的文档服务器,检索结果排序也由这个中心节点集中式地完成;而在 P2P 环境中存在中心节点的假设是不现实的.因此,在传统信息检索领域所取得的成果不能直接用于解决 P2P 环境面临的问题.

最近,P2P 信息检索已经开始受到关注.在文献[4]中,我们介绍了在 BestPeer[9] 平台上开发的 P2P 信息检索系统 PeerIS,并论述了体系结构、节点内部的模块及工作流程等关键技术,但没有深入研究检索结果排序和合并问题.

PlanetP[7] 也是一个采用向量空间模型[10] 表达文本数据的 P2P 信息检索系统.该系统采用基于 Bloom filter[11] 的资源定位和查询结果排序机制.每个节点都采用 Bloom filter 归纳本地文件的索引,并用 gossiping 算法[12] 将其在网络中散布.每个节点根据自己收集到的 Bloom filter 来查询共享文件:首先借助已收集到的 Bloom filter 来判断哪些节点可能维护查询答案;然后跟最可能提供答案的数个节点建立联系并下载答案;最后,查询提交节点排列这些被检索出的答案.可见,PlanetP 的排序机制存在以下局限性:(1)该机制是基于启发式规则的,因而不具备确定性.查询在哪些节点执行取决于提交节点所收集到的 Bloom filter.由于一个节点不可能收集系统所有节点的 Bloom filter,因而查询答案集具有随机性,即对于同一个查询和同一个在线节点集,查全率和查准率不确定.(2)查询排序计算的是集中式的,由查询提交节点完成,因而其计算负荷很重,进而影响系统的响应时间和可扩展性等性能.(3)尤其是它没有考虑各节点本地统计信息相异而导致的问题,这正是当前 P2P 信息检索系统排序问题的根源所在.本文正是要解决这个问题.

Psearch[6] 是在 CAN[16] 上开发的结构化 P2P 信息检索系统,它提出了一种具有确定性的排序策略.通过 CAN 提供的映射机制,包含特定关键词的索引项被发布到特定的节点.查找文档时,查询被转发给那些包含关键词的节点,后者把相关的索引项返回给查询节点,查询节点计算查询与文档的相似性并排序结果.本文提出的策略明显区别于 Psearch

的机制，在查询排序计算方式上，Psearch 的排序计算由查询提交节点独自完成；而本文提出的机制则把排序计算分散于所有被查询的节点，因而能更好地共享计算资源和平衡工作负荷.

3　检索结果排序与合并的问题根源分析

不失一般性，可假设 P2P 信息检索系统的每个节点都采用向量空间模型[14-15]表达数据，因为该模型是信息检索中最通用和最成熟的技术. 接下来深入分析导致 P2P 信息检索系统检索结果排序问题的根源.

基于上述假设，P2P 信息检索系统的每个文件及用户提交的查询都用向量表达，向量的维度就是系统中关键词的个数. 向量的每个权重就是对应关键词表达文档或查询的语义重要程度[16]. 由此，对于任意一个文档 d_j，其向量定义如下：

$$\boldsymbol{d}_j = (w_{1,j}, w_{2,j}, \cdots, w_{t,j}) \tag{1}$$

并且，对于用户提交的任意一个查询 q 的向量为

$$\boldsymbol{q} = (w_{1,q}, w_{2,q}, \cdots, w_{t,q}) \tag{2}$$

那么，查询 q 和文档 d_j 的语义相关程度可用它们向量之间夹角的余弦值来衡量，即

$$SR(d_j, q) = \frac{\boldsymbol{d} \cdot \boldsymbol{q}}{|\boldsymbol{d}_j| \times |\boldsymbol{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \tag{3}$$

上式中，文档和查询的关键词的权重(w_i)由 tf/idf 规则决定. 其中，tf(term frequency)是词频，代表一个关键词在同一文档中出现的次数；idf(Inverse Document Frequency)为反比文档频数，表示同一关键词在不同的文档中出现次数的倒数. 根据该规则，如果一个关键词在同一文档中出现的频率越高，那么就越能表征该文档的语义，其重要程度越高；如果同一个词在不同的文档中出现的次数越多，那么它就越不能区分文档，其重要程度越低.

现在讨论如何计算一个索引项(关键词)的词频. 如果用 $freq_{i,j}$ 表示关键词 k_i 在文档 d_j 中出现的原始次数，即原始词频；标准化词频用 $f_{i,j}$ 表示，那么可由以下公式计算：

$$f_{i,j} = \frac{freq_{i,j}}{\max_l freq_{i,j}} \tag{4}$$

上式中，$\max_l freq_{i,j}$ 为本地最大原始词频. 如果 k_i 不在文档 d_j 中出现，则 $f_{i,j}=0$. 由式(4)可计算出索引项的词频. 可见，任一索引项的词频仅决定于其所在的文档.

现在分析反比文档频数的情况. 传统信息检索系统(包括分布式信息检索系统)中，可把整个系统看成一个逻辑节点(该推理在 P2P 中不成立). 如果用 N 表示该逻辑节点的文档总数，n_i 表示含有索引词 k_i 的文件总数，idf_i 表示 k_i 的反比文档频数；那么 k_i 的反比文档频数可由以下公式计算而得：

$$idf_i = \log \frac{N}{n_i} \tag{5}$$

至此，关键词 k_i 的权重可由下列公式计算：

$$w_{i,j} = f_{i,j} \times idf_i = \frac{freq_{i,j}}{\max_l freq_{i,j}} \times \log \frac{N}{n_i} \tag{6}$$

现在分析 P2P 环境中关键词权重计算的情况. P2P 环境中，每个节点都是独立自治的信息检索系统，让我们在这种背景下分析式(6). 由式(4)可知，文档 d_j 的索引项 k_i 的词频 $f_{i,j}$ 仅由文件本身决定，而跟节点无关，即同一文档 d_j 的索引项 k_i 的词频 $f_{i,j}$ 在不同节点的取值仍然相同. 所以，元数据 $f_{i,j}$ 并未给 P2P 信息检索系统的检索结果排序产生问题.

idf_i 的情况却不同. 假设 x 和 y 为 P2P 信息检索系统中的两个不同节点. 由于节点是自治的，各自本地文档库中文档总数不同，包含关键词 k_i 的文档个数也不同，即

$$N^x \neq N^y, \quad n_i^x \neq n_i^y \tag{7}$$

现在分析索引项 k_i 在这两个节点中的反比文档频数. 在节点 x 中，k_i 的反比文档频数为

$$idf_i^x = \log \frac{N^x}{n_i^x} \tag{8}$$

在节点 y 中，索引项 k_i 的反比文档频数为

$$idf_i^y = \log \frac{N^y}{n_i^y} \tag{9}$$

由式(7)可知：$N^x \neq N^y$，$n_i^x \neq n_i^y$，那么在大多数情况下：

$$idf_i^x = \log \frac{N^x}{n_i^x} \neq idf_i^y = \log \frac{N^y}{n_i^y} \tag{10}$$

由元数据 N 与 n 的特性可知，式(9)在大多数情况下是成立的. 因此，对于同一文档 d_j 的同一关键词 k_i，它在不同的节点的权重很可能不等. 可见，由于 P2P 系统节点的自治性，同一索引项在不同节点中的反比文档频数不同.

类似地，对于同一个查询 q 中的关键词 k_i，它在不同节点解析出的反比文档频数也可能不同. 即对

于两个不同的节点 x 和 y，式(10)在大多数情况下成立：

$$w_{i,q}^{x}=\left(0.5+\frac{0.5freq_{i,q}}{\max_{l}^{q}(freq_{i,q})}\right)\times\log\frac{N^{x}}{n_{i}^{x}}\neq$$

$$w_{i,q}^{y}=\left(0.5+\frac{0.5freq_{i,q}}{\max_{l}^{q}(freq_{i,q})}\right)\times\log\frac{N^{y}}{n_{i}^{y}} \quad (11)$$

由式(10)和(11)可得：由于 P2P 系统的节点的自治性，单个节点没有全局信息且节点间的元数据不一致，使得节点仅按本地元数据计算得到的排序结果不可比. 因此，传统信息检索系统的排序机制不能直接移植到 P2P 信息检索系统.

4 检索结果排序与合并策略

本节首先提出一种分布式的元数据管理策略，以支持 P2P 环境下检索结果的排序；然后讨论与该策略相关的问题并提出解决方案；最后论述排序与合并的实现过程.

4.1 分布式元数据管理策略

如上所述，当前 P2P 信息检索系统的结果排序与合并所面临的挑战根源在于计算环境的变化. 解决该问题有三类可供选择的策略：(i)系统提供一个或数个服务器管理所有节点的元数据. 检索时，查询请求首先被发送到服务器并由服务器进行检索结果排序计算，或者查询首先散发给可能提供答案的节点，再由这些节点从服务器下载元数据进行检索结果排序计算. 采用前一种做法，所有的排序计算都由服务器承当，其必然不堪负重，系统存在可扩展性瓶颈. 采用后一种方法，每当有节点处理检索时，都必须从服务器下载元数据，由于系统有大量的节点，且每时每刻都可能有检索处理. 因而，服务器须不停响应下载请求，网络传输负荷非常大. 并且，这两种方法都存在单点出错的缺点. 显然，采用集中式机制的策略不适应于 P2P 环境. (ii)所有节点都维护整个网络的相关元数据. 该策略必然给所有节点带来极大的存储开销，可扩展性面临致命的挑战；另外，如何散布这些元数据以及维护它们的一致性又是一个问题. (iii)设计一种完全分布式的策略，让每个节点只为本地文件的索引项获得相关元数据的系统全局近似值. 该方法有三个优点：(1)每个节点只分摊一小部分存储负荷，消除了(ii)的局限性；(2)索引项的元数据的全局近似值维护在本地，检索结果排序计算可在本地即时完成，可分摊计算负荷，提高查询响应速度；(3)由于检索结果排序只需元数据的全局近似值[①]，元数据的更新频率不必很高，因而保证元数据的全局一致性和“新鲜性”不会引发高的传输负荷. 显然，该类策略适合解决 P2P 信息检索的问题.

接下来论述该策略的实现. 假设 P2P 信息检索系统的每个节点 p 都有唯一的标识符，用 PID 表示；k_i 是该节点本地的任一索引项；n_i^p 是节点 p 中含有关键词 k_i 的文档数；N^p 是该节点的本地文档总数. 进而，定义一个映射方程 $\Re$(如分布式散列表 DHT)，把索引项 k_i 映射到系统中对应的节点 X，X 被定义成关键词 k_i 的目标节点，用 $Target$ 表示. 于是，节点 p 可把本地任一索引项 k_i 的相关的元数据以元组〈$PID(p)$，n_i^p，N^p，$TimeStamp$〉的形式映射到(k_i的)目标节点上. 由于系统中所有节点都遵循同一 P2P 协议，即每个节点都嵌有相同的映射方程 $\Re$，使得 $Target$ 能够“收集”齐系统中所有与 k_i 相关的元组〈$PID(p)$，n_i^p，N^p，$TimeStamp$〉. 然后，$Target$ 周期性地“归纳”这些统计信息，并把“归纳”好的统计信息返回给那些上传元组的节点，为方便后面的表述，用 $Uploaders$ 代表上传节点. 实际上，这些“归纳”好的聚集信息就是关键词 k_i 的全局统计信息的近似值. 基于这些全局近似值，任何节点一旦接到查询，就可在本地计算该查询与本地文档的相似度 $SR(d_j,q)$，并即时排序本地检索结果. 注意，$Target$ 与 $Uploaders$ 只是节点在承担不同角色时被赋予不同的称呼而已. 索引项的元数据的上传、归纳和返回过程如算法 1 所述.

算法 1. 索引项 k_i 的 n_i 和 N 的全局近似值获取算法.

```
输入：ℜ, PID, n_i^p, N^p
输出：n_i, N
For Peer P (Initial Peer)：
 While (true)
 {
  if (bExit) //user want to exit
     exit (0)；
  if (IsInitiated()|FileModify()|FileAdd()|FileDelete())//节点刚初始化或其文件库更新
  {
     GetStatistics (n_i^p N^p)；//获取本地节点 p 的 n_i^p N^p
     PID=ℜ(k_i)； //根据映射机制确定目标节点
     SendToTarget(PID,〈PID(P),n_i^p,N^p〉)；
                          //把统计信息发送到目标节点
     GetFromTarget(PID,n_i,N)；
                          //从目标节点获取全局值 n_i, N
  }
  if(RunTime()>1 day)
                          //持续运行时间超过更新周期
```

① Waterhouse S.. Jxta search：Distributed search for distributed networks. http://search.jxta.org/JXTAsearch.pdf

```
    {
     GetFromTarget(PID,n_i,N);
                          //从目标节点获取全局值 n_i, N
     SetRunTimeToZero(); //运行时间归零
    }
    sleep(InternalTime); //等待
  }
  For Peer X (Target Peer):
  While (true)
  {
   if (bExit) //节点可能离线
     exit (0);
   if(!QueueEmpty()) //数据队列非空
   {
      ReceiveData(PID,n_i^p,N^p);
                          //从远程节点接受数据
    n_i=∑_p n_i^p+∑_p (n_i^p t−n_i^p t−1);
                          //计算 n_i,t 为当前更新周期,
    N=∑_P N^p+∑_p (N_t^p−N_t−1^p);
                          //计算 N,t−1 为上一更新周期
   SendBackData(PID,n_i,N); //返回更新后的数据
   }
   sleep(InternalTime); //等待
  }
```

4.2　节点动态性带来的挑战与对策

P2P 系统的显著特征之一就是节点的动态性，即节点可随时加入或离开系统. 节点的动态性必然影响节点元数据的“上载”、“归纳”和“返回”过程，进而影响系统元数据的一致性. 回顾算法 1，对于索引项 k_i，其所在的文档的维护节点（*Uploaders*）必须把相关元数据根据映射方程 $\Re$“上载”到对应的 *Target*. 由于节点的动态性，当 *Uploaders* 把这些元数据“上载”给 *Target* 时，如果 *Target* 恰好离线或出错，则 *Target* 就不能接收到相关的信息；同样，当 *Target* 把归纳好的信息返回给 *Uploaders* 节点时，如果 *Uploaders* 刚好离线，那么 *Uploaders* 也接收不到最新的统计信息. 因而，节点的动态性给保持系统统计信息的一致性带来了挑战. 解决该问题可采用代理机制. 对于系统中的任一节点 p，可邀请一个或数个节点形成一个虚拟节点并互为代理，当自己离线时代替自己接收相关信息，并在自己在线时交还自己. 这些节点的行为必须满足下列条件：

$$\bigcup Online_{\text{proxy}} \geq Offline_p$$

且

$$\bigcup Online_{\text{proxy}} \bigcup Online_p = 7\times 24 \qquad (12)$$

上式中，$\bigcup Online_{\text{proxy}} \geq Offline_p$ 表示代理节点在线的时间大于等于节点的离线时间；而 $\bigcup Online_{\text{proxy}} \bigcup Online_p = 7\times 24$ 表示节点及其代理中，每周 7 天、每天 24 小时的每时每刻都至少有一个节点在线，即节点 p 的代理节点的在线时间必须覆盖节点 p 的离线时间.

4.3　索引更新时机与策略

对于 P2P 信息检索系统而言，数据更新时机与更新策略的选择至关重要. 由于系统的索引项的数量非常大，数据更新时必然增加节点的计算负荷；同时，节点间的数据传输必然消耗带宽. 如果在查询处理时执行数据更新操作，必然影响系统的响应时间. 为了消除该影响，系统可选择大部分节点比较空闲的时机（如午夜）更新数据. 实验证明该机制是有效的.

当系统节点分布范围较小时，用户行为相对一致，数据更新时机（如午夜）的选择较为容易. 当系统的节点分布于不同的时区时，用户的行为往往不一致，因而难以确定统一的节点空闲时机. 此时，可以采用 4.2 节提出的代理机制来选择合理的数据更新时机，以缓解或消除数据更新对系统响应时间的影响.

数据更新策略是 P2P 信息检索面临的另一个重要的问题. 有效的更新策略必须能保证查询的正确程度（包括查全率、查准率和正确率，见第 5 节），又要能有效地利用系统资源（计算资源与带宽资源消耗少），涉及到更新周期、更新方式等多个方面. 信息检索的成果（如增量式更新[17-18]方式）很有借鉴价值. 数据更新本身就是一个重要的研究问题，将专门论述.

4.4　检索结果分布式排序与合并实现过程

本文提出的分布排序策略在检索处理过程中即时实现. 当一个查询通过某个节点提交给系统，这个查询首先在本地进行解析；然后，查询处理以并行的方式在 P2P 系统中进行：如果高层元数据表明本地可能有答案，就对本地文档库进行查询，同时根据系统的资源定位机制及查询路由协议把该查询转发给其它可能提供合格答案的节点. 其它节点一旦接到查询，便根据高层元数据判断本地是否可能提供合格答案，如果是，就根据式（3）计算查询与相关文档的语义相似度 $SR(d_j,q)$，否则仅转发查询消息. 由于相关的统计信息的全局近似值已维护在本地，所以被查询的节点的检索结果排序就可在本地即时完成（而不必进行元数据异地传输）. 最后，这些节点就把本地满足语义要求的所有答案（即 $SR(d_j,q)\geq threshold$ 的文档 d_j）或者排在最前面的 k 个答案（即 $SR(d_j,q)$ 最大的 k 个文档）的相关信息以 $\langle Title,SR,Summary,PID,Hops\rangle$ 形式返回给查询提交节点. 这里，*Title* 是被检索出的文档名，*SR* 是

查询与该文档的语义相似性，*Summary* 是被检索出的文章摘要，*PID* 是答案提供节点的标识符，而 *Hops* 则是答案提供节点到查询提交节点的逻辑距离. 查询提交节点一旦接收到答案提供节点返回的这些元组，就根据 *SR* 的大小重新排列和合并这些元组列表并展现给用户选择；如果两元组的 *SR* 的大小相同，则基于 *Hops* 进行二次排序. 如果所有这些答案仍然不能满足用户的需求，则查询提交节点再根据答案提供节点返回的信息从相关节点提取那些具有较大 *SR* 的元组，然后进行相同的排序合并操作. 一旦用户选择了满意的答案，就可从答案提供节点直接下载. 整个排序过程可用算法 2 表示.

算法 2. 查询结果排序与合并算法.

```
For Peer P (querying Peer): //对于查询发起者
void InitiateQuery(query)
{
 results+=QueryInLocalPeer(query);
                                  //在当前节点查询
 SendQueryToNeibs(query);
                  //将查询发送到邻居节点进行查询
 results+=WaitForResults(timeslot);
            //等待查询结果，等待时间为 timeSlot.
 RankingResults(results); //进行结果排序
 DisplayResultsToUser(results);
                                  //将结果返回给用户
}
for Peer X (queried Peer): //对于被查询节点
void RecieveQuery(query)
{
 if(isThereAnyRelatedResults(query))
                                  //本地有无相关答案
{
   results=ComputeSimilarity(query);
                                  //根据式(3)计算相似度
 }
 ReSendQueryToNeibs(query);
                                  //将查询转发给其它邻居节点
 for (resultA in results)
                  //对于 results 中的任一结果 result A
 {
  if(SR(resultA,query)>threshold) //如相似度大于
  阈值
  {
   SendResultToQueryInitiator (resultA); //返回结
   果给查询节点
  }
 }
}
```

5 实验分析

本节用实验证明本文提出的策略的有效性. 首先描述实验环境；然后阐述实验方法论，包括数据分布、对比策略以及实验指标体系；最后依次基于这些指标比较不同的策略.

5.1 实验设置

实验由同一局域网内的 12 台 PC 机构成，配置：CPU 为 Intel Pentium 1.8GHz 微处理器、128MB RAM、Windows 2000 操作系统. 每台 PC 运行 5～6 个 PeerIS[4] 线程，每个线程代表一个 P2P 信息检索节点，由其所在的 PC 的 IP 地址和端口标识，于是构造出由 64 个节点组成的 P2P 信息检索系统. 接着，为每个节点嵌入一个分布式 Hash 方程来实现元数据的“上载”、“归纳”和“返回”等操作，使节点获得统计信息的全局近似值并保持其一致性. 另外，再产生一个文档集，每个文档的大小为 10～100KB，由 100 个关键词索引；并为每个节点分配 1000～2000 个文档. 再产生一个包含 10 个关键词的查询转发给相关节点进行 $SR(d_j,q)$ 计算，并根据检索结果评价相关的指标.

5.2 评价方法

定义三个指标来评价本文提出的策略：查全率（*Recall*）、查准率（*Precision*）和正确率（*Correctness*）. 三个指标的内涵将在实验分析过程中详细论述.

P2P 查询的答案取决于查询访问过的节点. 在动态的 P2P 系统中，同一查询在不同的时间可能访问不同的节点. 为了保证实验的可比性，实验环境必须可控，不同的比较方案必须基于同一个节点集评价. 我们设计了 3 种比较方案：(1) Grank，系统的所有文档及与排序相关的元数据都由一个节点维护，这样获得的查全率、查准率和正确率是系统在理想状态下才能够获得的. 在实验中，Grank 所能获得的查全率、查准率和正确率均设为 100%，用作评价基准；(2) Drank 表征本文提出的分布式排序策略；(3) Trank 表示节点仅基于本地文档的元数据进行排序，用于反映把传统信息检索系统的排序策略直接用于 P2P 信息检索系统时出现的情形.

实验采用两种不同的数据分布来评价排序策略：均匀分布和 80/20 分布. 采用均匀分布数据分布时，系统的整个文档集中的文档随机且均匀地分布给每个节点，因而每个节点本地文档的 n_i^p/N^p 与系统的全局值 n_i/N 很相近，但这种数据分布在现实中是最不可能出现的. 80/20 分布能够很好地反映基于 Internet 的现实应用中信息消费的情形[9].

此外，实验机器均为专用；为消除实验误差，实验结果取 5 次实验的平均值. 同时，当评价查全率与查准率时，由于阈值在 0.68～0.77 时，实验结果曲

线涵盖了所有的变化趋势，并且此时本文提出的策略相对于传统策略的优势最小，最能说明问题，故 5.3 节与 5.4 节的实验结果图中仅展示该阈值区间的结果.

5.3　查全率

查全率(*Recall*)是检索处理完成时检索出的相关文档数占系统总的相关文档数的比例. 由于节点的动态性，P2P 系统的查全率应该以查询提交至查询完成这段时间内所有在线节点为计算基础，即查询所获得的合格答案的数量与在查询执行过程中所有在线节点所能提供的合格答案之比，计算公式如下：

$$Recall = \frac{\sum_{\text{Retrieved}} Answer}{\sum_{\text{CurrentAvailable}} Answer} \tag{13}$$

上述公式中，$\sum_{\text{Retrieved}} Answer$ 是用户在规定时间内获得的符合需求答案的总数，而 $\sum_{\text{CurrentAvailable}} Answer$ 则为整个系统范围内查询执行过程中所有在线节点所能提供的满足需求答案的总数目. 显然，系统的查全率越高就越能满足用户的需求，最理想的情况是用户检索出所有合格答案. 在实验中，查询执行 5 次，查全率的平均结果如图 1 和图 2 所示.

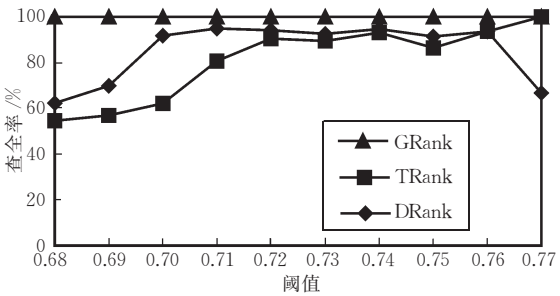

图 1　均匀数据分布的查全率之比较

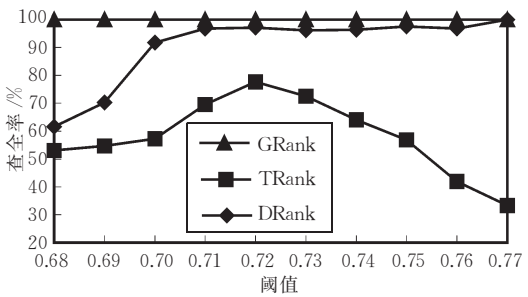

图 2　80/20 数据分布下的查全率之比较

图 1 所展示的均匀数据分布情形最有利于 Trank，即仅采用节点本地文档的统计数据进行检索结果排序计算的情形. 由上图可见，DRank 和 TRank 均很接近于 GRank，这是因为采用均匀数据分布，在 TRank 场景中，每个节点本地文档的统计信息 n_i^p/N^p 从整体而言与系统全局的统计信息 n_i/N 比较接近；而采用本文提出的分布式元数据管理策略，即 DRank，每个节点通过分布式算法能够获得本地所有索引项元数据的全局近似值. 注意，即使在最有利 TRank 的情形下(采用均匀数据分布)，DRank 仍然可以获得更接近理想状态的查全率，这说明本文提出的分布式策略在元数据管理方面确实有效.

图 2 展示了 80/20 数据分布的情形，它最能反映现实应用中信息共享的实际情况. 在这种数据分布情况下，DRank 远远优于 TRank. 并且，随着查询与文档的相似度的阈值(*threshold*)的增加，DRank 开始迅速增加后几乎保持不变，并非常接近 GRank. 相反，TRank 的查全率随着查询与文档的相似性的阈值的增加而减少. 这是因为本文提出的分布式元数据管理策略能保证节点获取其索引项的元数据的全局近似值，因而可以保证很高的查全率. 相反，TRank 策略仅根据本地文档的局部统计信息进行排序计算，不可避免作出错误的决策；并且，随着查询与文档的相似性的阈值的增加，合格答案被不合格答案取代的机会增加，因而越来越多合格答案被遗漏，导致查全率不断下降.

从上面两个图可见，无论是采用均匀数据分布还是采用 80/20 数据分布，DRank 的查全率都比 TRank 的查全率高，即本文提出的排序策略的查全率在两种数据分布情况下都高于仅用节点局部统计数据进行检索结果排序的查全率.

5.4　查准率

P2P 信息检索系统与传统信息检索系统的查准率的计算机理相同，均可用下列公式：

$$Precision = \frac{\sum_{\text{Qualified}} Answer}{\sum_{\text{Retrieved}} Answer}.$$

上式中，$\sum_{\text{Retrieved}} Answer$ 是在查询过程中从所有被查询节点检索出的所有答案的总数目，$\sum_{\text{Qualifed}} Answer$ 则是检索出的结果中所有合格答案总数目，即满足不等式 $SR(d_j,q)\geq threshold$ 的答案的总数.

在检验查全率的过程中，也记录了不同阈值时的 $\sum_{\text{Retrieved}} Answer$，$\sum_{\text{Qualifed}} Answer$，并根据式(14)计算出对应的查准率，五次实验的平均结果如图 3 与图 4 所示.

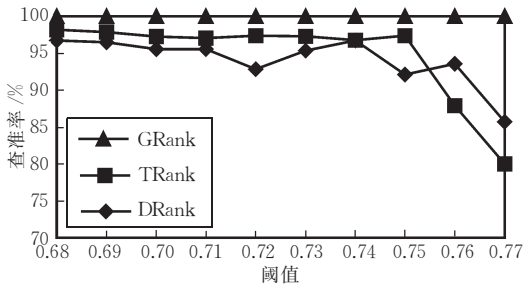

图 3 均匀数据分布的查准率之比较

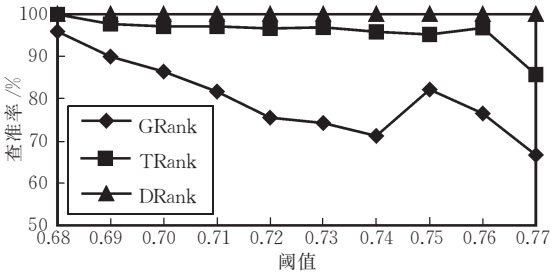

图 4 80/20 数据分布下的查准率之比

图 3 展示了采用均匀数据分布时,各种排序策略的查准率的平均值.

图 4 展示了采用 80/20 分布时不同排序策略的查准率的平均值.从图 3 和图 4 可以看到出现了分别类似于图 1 和图 2 的情形.实际上,导致这种结果的原因是一致的.

5.5 正确率

由于当前 P2P 数据共享系统的节点在返回检索结果时大多采用 top-k 协议.即如果被查询节点能够提供可能的合格答案,则返回本地最满足条件的 M 个答案给查询提交节点,查询提交节点重新排序和合并这些结果后,从中选出最能满足语义要求的 K 个答案给用户选择.

为了验证本文提出的分布式检索结果排序策略在 top-k 协议中的有效性,我们定义了一个新的评价指标:正确率(*Correctness*).接下来论述正确率的内涵与计算方法.假设在 P2P 信息检系统中的一次查询处理中,任意一个查询处理节点返回 M 个最符合语义要求的答案给查询提交节点,该集合定义为 $\{a_1,a_2,\cdots,a_M\}$,那么所有查询处理节点提供的答案所构成的集合为 $\bigcup_{\text{Queried\_peer}}\{a_1,a_2,\cdots,a_M\}$.并且,假设所有的共享文档仅由一个节点维护(即 GRank),那么对于同一查询,该节点提供与 $\bigcup_{\text{Queried\_peer}}\{a_1,a_2,\cdots,a_M\}$ 相同数目的最好答案构成的集合为 $\{GlobalAnswers\}$,而其中最满足语义的 K 个(top k)答案构成的子集为 $\{TopK_{\text{GlobalAnswers}}\}$,那么 P2P 信息检索系统的正确率可由下列公式计算:

$$Correctness=\frac{\bigcup_{\text{Quieried\_peer}}\{a_1,a_2,\cdots,a_M\}\cap\{TopK_{\text{GlobalAnswers}}\}}{\{TopK_{\text{GlobalAnswers}}\}} \quad (15)$$

我们分别在均匀数据分布和 80/20 数据分布的情形下,用前面定义的查询来评价这三种排序策略的正确率.显然,基于上述定义,GRank 策略的正确率为 100%.在数据均匀分布时,查询执行 5 次所得的平均正确率如图 5 所示.

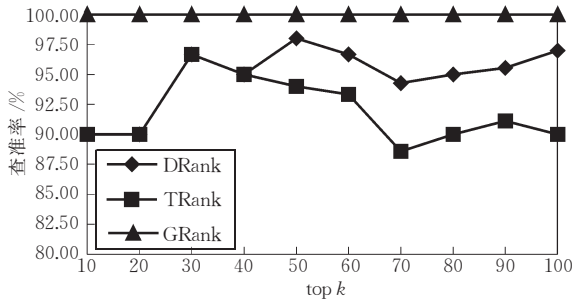

图 5 均匀数据分布的正确率之比较

从上图中可见,TRank 和 DRank 的正确率都非常接近于 GRank.这也是因为采用均匀数据分布时,TRank 中单个节点的 n_i^p/N^p 接近于 n_i/N,而本文提出的排序策略(DRank)保证了查询处理节点获得(相关全局信息)n_i/N 的近似值,因而两者的正确率很高并且接近.即使在这种情形下,DRank 的正确率仍然比 TRank 的正确率高.

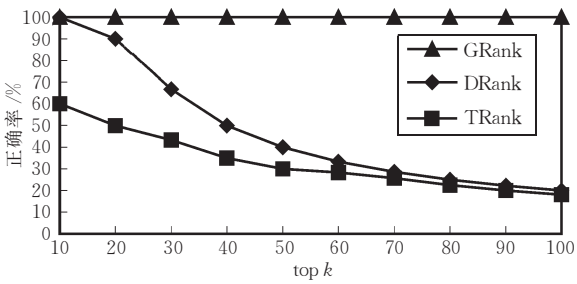

图 6 80/20 数据分布下的正确率之比较

图 6 展示了 80/20 数据分布时不同排序策略的正确率.该图表明,DRank 的正确率远远高于 TRank,尤其当 K 值较小时,两者的差异更加明显.这是因为本文提出的策略保证了节点获得非常接近于系统的 n_i^p/N^p,而 TRank 中单个节点的 n_i^p/N^p 远远偏离于系统的 n_i/N.此外,可以观察到,随着 K 的增加,DRank 与 TRank 的正确率都变差.这是因为,随着 K 的增加,M 也增加,所有查询节点返回越来越多的答案,虽然这些答案在查询节点本地排列

较前，但从系统全局来看却很可能不是较前的. 因而，随着 K 的增加，两者的正确率都下降. 由于在 P2P 应用中，用户仅关注最前几个答案. 所以，本文提出的分布式排序策略是有效的，而传统信息检索系统的排序策略难于满足需求.

总之，无论是在均匀数据分布还是在 80/20 数据分布情形下，本文所提出的分布式排序策略的查全率、查准率和正确率都比仅根据节点在本地文档的局部统计信息计算排序优先权对应的指标更好.

6 结　论

本文深入分析了导致 P2P 信息检索系统在查询结果排序与合并上产生问题的根源，提出了一种完全分布式的检索结果排序与合并策略、采用该策略的相关问题及其解决方案，并简要描述了这种检索结果排序和合并的实现过程. 此外，用实验验证了该策略的有效性和实用性. 实验结果表明，本文提出的策略远远优于仅根据节点在本地文档的局部统计信息进行检索结果排序的传统做法，并非常接近理想情况下所能取得的结果.

参　考　文　献

[1] Druschel P, Rowstron A. PAST: A large-scale persistent peer-to-peer storage utility//Elphinstone K ed. Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII). Schoss Elmau, Germany: IEEE Press, 2001: 65-70

[2] Kalnis P, Ooi B, Papadias D, Tan K. An adaptive peer-to-peer network for distributed caching of olap results//Ramakrishnan R ed. Proceedings of ACM Conference on Management of Data (ACM SIGMOD). Madison, Wisconsin, USA: ACM Press, 2002: 25-36

[3] Wee Siong Ng, Beng Chin Ooi, Kian-Lee Tan, Aoying Zhou. PeerDB: A P2P based system for distributed data sharing//Proceedings of ICDE, 2003: 633-644

[4] Ling Bo, Lu Zhi-Guo, Ng Wee Siong, Qian Wei-Ning, Zhou Ao-Ying. PeerIS: A Peer-to-Peer based Information retrieval System. Journal of Software, 2004, 15(9): 1375-1384(in Chinese)
(凌　波，陆治国，黄维雄，钱卫宁，周傲英. PeerIS：基于 Peer-to-Peer 的信息检索系统. 软件学报，2004，15(9)：1375-1384)

[5] Ling Bo, Wang Xiao-Yu, Zhou Ao-Ying, Ng Wee Siong. A collaborative Web caching system based on Peer-to-Peer architecture. Chinese Journal of Computers, 2005, 28(2): 170-178(in Chinese)
(凌　波，王晓宇，周傲英，黄维雄. 一种基于 Peer-to-Peer 技术的 Web 缓存系统研究. 计算机学报，2005，28(2)：170-178)

[6] Tang Chun-Qiang, Xu Zhi-Chen et al. pSearch: Information retrieval in structured overlays. Computer Communication Review, 2003, 33(1): 89-94

[7] Cuenca-Acuna FM, Nguyen TD. Text-Based content search and retrieval in ad hoc P2P communities. Department of Computer Science, Rutgers University: Technical Report DCS-TR-483, 2002

[8] Richardo Baeza-Yates, Berthier Ribeiro-Neto. Modern Information Retrieval. New York: ACM Press, Addison-Wesley, 1999

[9] Ng W S, Ooi B C, Tan K L. BestPeer: A self-configurable Peer-to-Peer system//Chrysanthis P K ed. Proceedings of the 18th International Conference on Data Engineering. San Jose, CA, USA: IEEE Computer Society Press, 2002: 272

[10] Anick P G. Adapting a full-text information retrieval system to computer the troubleshooting domain//Proceedings of the ACM SIGIR'94, 1994: 349-358

[11] Bloom B H. Space/time trade-offs in hash coding. Journal of the ACM, 1970

[12] Demers A, Greene D, Hauser C, Irish W, Larson J, Shenker S, Sturgis H, Swinehart D, Terry D. Epidemic algorithms for replicated database maintenance//Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing, 1987: 1-12

[13] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network//Govindan R ed. Proceedings of the ACM SIGCOMM. San Diego, CA: ACM Press, 2001: 161-172

[14] Salton G, Lesk M E. Computer evaluation of indexing and text processing. Journal of the ACM, 1968, 15(1): 8-36

[15] Raghavan V V, Wong S K M. A critical analysis of vector space model for information retrieval. Journal of the American Society for Information Science, 1986, 375(3): 279-287

[16] Yu C T, Salton G. Precision weighting — An effective automatic indexing method. Journal of the ACM, 1976, 23(1): 76-88

[17] Brown E W, Callan J P, Bruce Croft W. Fast incremental indexing for full-text information retrieval//Proceedings of the 20th International Conference on Very Large Databases, 1994: 192-202

[18] Anthony Tomasic, Hector Garcia-Molina, Kurt Shoens. Incremental update of inverted list for text document retrieval//Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, 1994: 289-300

LING Bo, born on 1974, Ph. D., associate professor. His main research interests include data management and information retrieval in peer-to-peer computing, information systems, and regional economics etc.

ZHOU Shui-Geng, born on 1966, Ph. D., professor, Ph. D. supervisor. His main research interests include information retrieval and text mining, GIS and special database, and P2P computing etc.

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

Background

This paper is supported by projects funded by National Natural Science Foundation of China: "Query Processing in P2P Environment" (grant No. 60373019) and "Searching in P2P Networks" (grant No. 90612007) etc. These projects manage 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 propose new P2P-ased applications, including relational data sharing and processing, data mining, and information retrieval. Specifically, the key techniques of information retrieval, such as result ranking and merging, are also the important topics of the projects.

Although great advances have been achieved in peer-to-peer computing since 2000, most of the other projects have following characters and limitations: (1) They are operating system or network oriented but not data-centered; (2) Most of them just support semantics-free file sharing of file granularity; (3) No comprehensive data operation and algorithms or strategies have been proposed. On the contrary, these projects manage to achieve semantic and effective data management in P2P environment and propose P2P-based information retrieval as well. Specifically, this paper targets one of the key challenges of information retrieval in P2P environment, e. g., how to rank and merge results retrieved from different peers, and successfully proposed a fully distributed strategy to rank and merge the results retrieved from different peers in P2P information retrieval systems. First, the challenges of query answering in P2P IR systems have been investigated and the issue of retrieved results ranking and merging has been identified. Second, a fully distributed strategy has been developed and its implementing issues have been addressed. Finally, an extensive experimental study has been conducted and its results have verified the effectiveness of this proposed strategy.