

# PATCOM: 基于分割树的无结构 P2P 系统 一致性维护方法

李振宇<sup>1),2)</sup> 谢高岗<sup>1)</sup> 李忠诚<sup>1)</sup>

<sup>1)</sup>(中国科学院计算技术研究所 北京 100080)

<sup>2)</sup>(中国科学院研究生院 北京 100039)

**摘 要** 无结构 P2P 技术逐渐被应用在新型的协同计算系统中, 这些新型业务支持数据的动态更新, 不仅要求副本数据的强一致性, 而且要求更新数据的快速传播. 高效的一致性维护方法是保证新业务顺利开展的基础. 在比较分析现有的 P2P 系统一致性维护方法的基础上, 针对无结构 P2P 系统, 提出了一种基于分割树的一致性维护方法——PATCOM. PATCOM 使用 Chord 协议作为组管理协议, 通过不断分割由副本节点组成的 Chord 环, 动态地建立更新消息传播树(Update Message Propagation Tree, UMPT). 论文进一步从理论上分析了 UMPT 的平均高度、PATCOM 的性能、容错能力以及算法开销, 并和基于 Gossip 的一致性维护方法进行了比较. 理论分析和仿真实验结果表明: PATCOM 不仅能够快速地维护 P2P 系统的强一致性, 而且产生的冗余更新消息少.

**关键词** 无结构 P2P 系统; 一致性维护; 分割树; Chord; 性能分析

**中图法分类号** TP393

## PATCOM: Partition Tree-Based Consistency Maintenance for Unstructured P2P Systems

LI Zhen-Yu<sup>1),2)</sup> XIE Gao-Gang<sup>1)</sup> LI Zhong-Cheng<sup>1)</sup>

<sup>1)</sup>(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

<sup>2)</sup>(Graduate University of Chinese Academy of Sciences, Beijing 100039)

**Abstract** Unstructured P2P technique is gradually applied in newly-developed cooperative computing systems. These applications support the dynamical updates of data, and require not only strong consistency but also fast propagation of update messages. An efficient consistency maintenance method is the basis for the developing of newly-developed applications. Based on intensive analysis and comparisons for existing methods, the authors propose a partition tree-based consistency maintenance scheme for unstructured P2P systems, PATCOM. PATCOM uses Chord as the group management protocol and propagates update messages along with the Update Message Propagation Tree(UMPT), which is built dynamically on top of the Chord ring composed of replica nodes. The authors theoretically analyze the average height of UMPT, the performance of PATCOM, the failure tolerance and the overhead of the proposed scheme. Then, the authors compare PATCOM with the Gossip-based consistency maintenance method. Finally, they verify the theoretical results and the performance of PATCOM by simulation experiments. The performance analysis and simulation results show that PATCOM not only maintains a strict consistency, but also brings fewer redundant update messages.

**Keywords** P2P systems; consistency maintenance; partition tree; Chord; performance analysis

收稿日期: 2006-04-25; 最终修改稿收到日期: 2007-04-28. 本课题得到国家自然科学基金(60403031, 90604015)和国家“八六三”高技术研究发展计划项目基金(2005AA121560)资助. 李振宇, 男, 1980 年生, 博士研究生, 主要研究方向为 P2P 系统、网络测量和管理. E-mail: zyl@ict.ac.cn. 谢高岗, 男, 1974 年生, 博士, 副研究员, 主要研究方向为对等计算、网络测试、监控和管理. 李忠诚, 男, 1962 年生, 博士, 研究员, 博士生导师, 主要研究领域为计算机网络、测试、可信计算.

# 1 引言

无结构 P2P 技术逐渐被应用在新型的协同计算系统中. 在这些新型业务中, 同一数据会被多处复制, 而且数据可以被合法地修改. 如何维护这些复制副本数据的一致性称为无结构 P2P 系统的一致性维护问题. 在基于无结构 P2P 技术的文件共享系统中(如 Gnutella<sup>①</sup>、Freenet<sup>[1]</sup>等), 数据(如 mp3、video 等)通常被认为是静态的, 不会产生频繁的更新. 但是, 随着 P2P 技术的发展, 无结构 P2P 技术逐渐被应用在新型的协同计算系统中, 其典型应用有可信管理<sup>[2]</sup>、多人游戏、目录服务、联机拍卖、远程协作等. 在这些新型应用中, 数据被多处复制和共享(一般每个节点都有一份副本), 而且允许用户修改, 所以更新操作频繁. 这些新业务的一个突出特征在于: 不仅要求副本数据的强一致性, 而且要求更新数据被快速地传播到所有副本节点. 因此, 设计高效的算法来维护 P2P 系统的一致性, 是保障 P2P 新业务顺利开展的关键. 一方面, 如果没有有效的一致性维护算法, P2P 业务只能局限于提供静态数据的共享; 另一方面, 新业务的开展需要一致性维护算法为数据频繁更新提供支持和保障.

P2P 系统中, 保存关键字  $k$  对应的数据( $data_k$ )的节点称为关键字  $k$  的副本节点. 这些副本节点构成一个组( $group_k$ ), 副本节点是组的成员. P2P 系统的一致性维护指当某个数据  $data_k$  被合法修改后, 保证组  $group_k$  的每个组成员尽快与修改后的数据保持一致. 组管理协议是 P2P 系统一致性维护的关键, 需要具有以下特征: (1) 支持组成员的动态加入和离开; (2) 具有较高的容错能力; (3) 具有较高的扩展性. 而对于基于组管理协议的一致性维护方法应该满足如下条件: (1) 保证所有成员的一致. 虽然概率性的一致对于某些应用已经足够, 但像联机拍卖、远程协作等协同计算应用必须保证全部成员的一致, 即强一致性; (2) 快速, 即保证所有成员能够尽快地收到更新; (3) 开销尽量小, 即冗余的更新消息尽量少. P2P 系统的动态、分布式特性使得 P2P 系统的一致性维护不同于 Web 代理缓存中的系统一致性维护.

集中式方法<sup>[3-5]</sup>是解决 P2P 系统一致性维护的最直观方法: 由一个和几个副本节点保存所有组成员信息, 更新消息由这些节点向所有副本节点发送. 集中式方法虽然能够快速地维护所有成员的一致, 但这种组管理协议扩展性差, 容易造成单点失效问

题. 基于流言(Gossip)的组管理协议<sup>[6]</sup>具有较高的容错能力和扩展性, 但基于该协议的一致性维护方法只能保证副本的概率性一致, 而且会产生大量的冗余. 另一种能够快速维护 P2P 系统一致性的方法是把组成员组成一个树状结构<sup>[7]</sup>, 更新消息自顶向下传播. 这种结构冗余更新消息少, 更新速度快, 但树状结构容错能力差. 如何提高树状结构的容错能力, 减少树的维护开销是这种方法的关键所在.

为了提高 P2P 系统一致性维护的收敛速度、保证所有成员的完全一致性, 针对无结构 P2P 系统, 本文提出了一种基于分割树的副本一致性维护方法——PATCOM (A PArtition-Tree-based COnsistency Maintenance for unstructured P2P systems). PATCOM 使用 Chord<sup>[4]</sup>作为组管理协议,  $group_k$  的组成员在逻辑上构成一个 Chord 环. 通过不断分割 Chord 环所代表的 ID 空间, 动态地建立更新消息传播树(Update Message Propagation Tree, UMPT)来维护副本的一致性.

基于分布式 Hash 表(Distributed Hash Table, DHT)的 Chord 协议支持节点的动态加入和离开, 具有较强的扩展性和容错能力. 在一个由  $M$  个节点组成的 Chord 网络中, 所有查询操作可在  $\log M$  跳内完成. 这是 Chord 协议最突出的特征之一. 作为组管理协议, Chord 为 UMPT 的快速建立提供了支持, 极大地提高了 UMPT 错误恢复容错能力, 从而为高效、快速地传播更新数据到所有副本节点提供了最基本的保障.

理论分析和仿真实验结果显示, 对于关键字  $k$  的  $N$  个副本节点, 一次更新操作平均需要  $O(\log^2 N + \log N)$  跳, 每个节点平均产生  $O(1)$  更新消息, 且每个节点平均维护  $O(\log N)$  其它副本节点的信息; 而在更新过程中, 一个节点的失效平均产生  $O(\log N)$  冗余更新消息. 所以, PATCOM 具有良好的扩展性和容错能力, 不仅能够快速地维护所有副本的一致, 而且产生的冗余更新消息少.

本文第 2 节介绍相关工作; 第 3 节详细介绍 PATCOM, 并在理论上分析其性能; 第 4 节使用仿真实验验证理论分析结果和 PATCOM 的性能; 最后, 第 5 节总结全文并介绍进一步工作计划.

## 2 相关工作

副本与缓存的一致性问题近年来被广泛研究.

① Gnutella. <http://gnutella.wego.com/>

文献[8-9]研究了 Web 代理缓存中的一致性维护,其中代理是稳定的,而 P2P 系统中保存副本的节点可以动态加入、离开,甚至失效,所以这些方法不能直接应用在 P2P 系统中. Gnutella 为了提高查询请求命中率以及响应时间,查询所得结果沿查询请求路径返回,并在经过的节点上保存结果. Gnutella 中,副本一致维护和资源定位算法一样使用基于泛洪(flooding)的方法,这种方法缺乏扩展性,且冗余更新消息多. Chord<sup>[4]</sup>为了提高系统容错能力,在存储数据 *data* 节点的 *c* 个连续后继节点上保存数据 *data* 的副本. CAN<sup>[3]</sup>为了解决负载不均问题,对于访问率较高的数据,在邻居节点上复制数据. Chord、CAN 以及基于 Chord 的 CFS<sup>[5]</sup>中,副本的个数以及存储位置由保存原始数据的节点控制,这种集中式的管理方法不易扩展,而且会带来单点失效问题.

文献[6]提出了一种“推”(push)与“拉”(pull)相结合的无结构 P2P 系统一致性维护算法. 该方法使用 Gossip 协议<sup>[10]</sup>作为组管理协议. 更新消息的传播类似于谣言传播:某个组成员收到更新消息后,以一定的概率把该消息传播给组内其他若干个成员,这就是“推”. 当某个节点加入组后,主动地从组内其他成员获取最新的数据,这就是“拉”. 该方法能够适应高度动态的环境,而且比其他一致性维护方法有较高的性能. 为了减少冗余更新消息以及学习更多的节点信息,更新消息包含了该消息已经经过的节点的地址信息(IP 地址),将不再给这些节点发送更新消息. 但这并不能完全消除冗余更新消息,因为更新消息是沿着多条路径传播出去的,沿这条路径传播的消息中并不包含其他路径中经过的节点,所以冗余更新消息还是比较多. 另外,该方法旨在高度动态的环境下保证系统的概率性一致,并不能保证系统的完全一致. 为了提高系统一致性的概率,必须增加更新消息传播的轮数,但这将增加冗余的消息数. 文献[11]提出了基于泛洪的一致性维护算法,这种算法并没有真正的组成员管理协议,副本节点的关系是随意的. 这种方法的冗余消息更多,而且同样不能保证系统的一致性.

SCOPE<sup>[7]</sup>通过为每个关键字建立“副本分割树”(Replica-Partition-Tree, RPT)解决结构化 P2P 系统中副本一致性问题. 分析表明,在由  $N$  个节点的系统中,更新操作在  $O(\log^2 N)$  跳内完成,而且对于一个关键字,节点平均保存  $O(\log N)$  分割向量. SCOPE 利用了结构化 P2P 系统中固有的 DHT,能够保证所有副本的一致. 但是,在 RPT 中,某些节点

会出现在多个分割层,与这些节点连接的节点也就很多(即出度大),所以这些节点极有可能过载(overloaded). 另外,SCOPE 中需要静态地维护多个 RPT,其维护开销是不可忽略的.

本文提出的算法 PATCOM 是基于 UMPT 传播更新消息的. 与 SCOPE<sup>[7]</sup>不同, PATCOM 中的 UMPT 是动态、按需建立的,在没有更新消息传输时,不需要维护 UMPT. 另外,在 UMPT 中每个节点只出现一次,节点过载的几率大大减少. 虽然 PATCOM 同样可以看作是“推”与“拉”结合的更新算法,但是 PATCOM 中的“推”是在树型结构 UMPT 上传播更新消息,更新所需时间短,冗余更新消息少.

### 3 PATCOM

在介绍 PATCOM 之前,做如下的假设:

- (1) 每个副本节点知道组内部分节点的信息,但没有一个副本节点知道组内所有副本节点的信息;
- (2) 副本节点是合作的,即副本节点会按照一致性维护方法参与一致性维护;
- (3) 副本数据的强一致性是至关重要的.

PATCOM 基于 Chord<sup>[4]</sup>协议管理组成员,关键字  $k$  的所有在线副本节点逻辑上组成 Chord 环. P2P 系统中的节点在获取数据 *data<sub>k</sub>* 的副本后,如需保持与该副本一致时,则根据 Chord 协议加入组 *group<sub>k</sub>*,并从指针表所指的节点中获取最新的副本信息和内容. 若 *group<sub>k</sub>* 中的某个成员不再需要保持与副本一致时(节点删除副本或者离开 P2P 系统),则根据 Chord 协议离开组 *group<sub>k</sub>*. 当组 *group<sub>k</sub>* 中的某个成员需要发起更新操作时,以该成员为根,通过不断分割 Chord 环的 ID 空间动态地建立 UMPT,更新消息沿着 UMPT 自顶向下传播. 为了增加 PATCOM 的容错能力,在更新过程中维护 UMPT,通过高效的错误恢复机制处理更新过程中节点的失效. 更新操作完成后,删除 UMPT. 算法包括 UMPT 建立、更新操作、副本节点加入与离开以及节点失效处理过程. 以下分别进行详细描述.

#### 3.1 UMPT 建立

关键字  $k$  的在线副本节点逻辑上组成一个 Chord 环 *ring<sub>k</sub>*,关键字  $k$  的 UMPT 建立在环 *ring<sub>k</sub>* 之上. 当环上的某个节点需要发起更新操作时,以该节点为根,通过逐步分割环 *ring<sub>k</sub>* 代表的 ID 空间建立 UMPT. 建立  $d$ -叉 UMPT 的过程如下:发起操作的节点拥有整个 ID 空间,把整个 ID 空间分割为大

小相等的  $d$  个区域. 选取每个区域沿顺时针方向的第一个节点为这个区域的代表节点 (Representative Node, RPN), 这些代表节点组成根节点的孩子节点集合. 接着, 每个代表节点把自己所代表的区域继续分隔. 以此类推, 直到每个区域只有一个节点. 区域分割的伪代码如算法 1 所示, 其中  $X.foo()$  表示函数  $foo()$  由节点  $X$  调用并在节点  $X$  上执行, 而  $Get(id)$  函数是用来获取  $id$  对应的后继节点 (successor node), 该函数由 Chord 协议提供.

#### 算法 1. ID 区域分割算法.

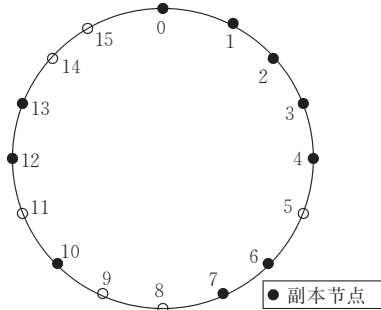
```

 $X.region\_partition(region\_x)$ 
1. if ( $X.id+1 > region\_x.end$ )
2.   return;
3.  $region \leftarrow (X.id+1, region\_x.end)$ ;
4. Split  $region$  into  $d$  partitions with equal size
5. for  $i=1$  to  $d$  {
6.    $region[i] \leftarrow$  the  $i$ -th partition;
7.    $RPN_{region[i]} = X.get\_rpn(region[i])$ ;
8.   if ( $RPN_{region[i]} \neq NULL$ ) {
9.      $X.children = X.children \cup RPN_{region[i]}$ ;
10.     $RPN_{region[i]}.region\_partition(region[i])$ ;
11.  }
12. }
```

$X.get\_rpn(region)$

```

1.  $id \leftarrow$  first ID of this  $region$ ;
```



(a) 副本节点组成的覆盖网络,  $m=4$

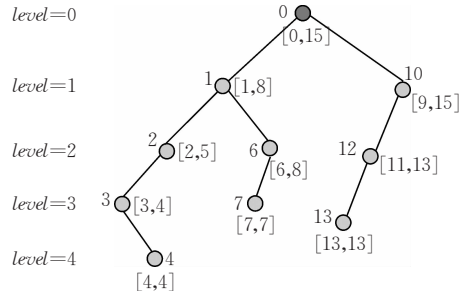
```

2.  $node \leftarrow X.Get(id)$ ;
3. if ( $node.id \notin region$ )
4.   return NULL;
5. return  $node$ .
```

为了使每个节点在 UMPT 仅出现一次, 在分割区域的过程中, 把区域的代表节点排除在进一步分割之外 (函数  $region\_partition$  的第 3 行). 分割所得区域的代表节点组成当前区域代表节点的孩子节点集合 (函数  $region\_partition$  的第 9 行), 接着由这些代表节点继续分割区域 (函数  $region\_partition$  的第 10 行). 当分割所得区域内一个节点也没有 (函数  $get\_rpn$  的第 3~4 行) 或者只有一个节点时 (函数  $region\_partition$  的第 1 行), 该区域不再被分割.

在 Chord 覆盖网络上需要多跳才能查询到  $id$  对应的后继节点, 由此带来的时间开销是不能忽略的. 为了缩短建立 UMPT 所需时间, 代表节点在分割区域时开启  $d$  个线程, 其中线程  $i$  负责获取第  $i$  个区域 (region) 的代表节点.

通过上述过程, 建立了一个  $d$ -叉的更新消息传播树 UMPT, 且该 UMPT 的高度为  $O(\log N)$ . 图 1 显示了副本节点组成的 Chord 环以及在该 Chord 环上建立的 2-叉 UMPT.



(b) 2-叉 UMPT 树 ( $d=2$ )

图 1 副本节点组成的 Chord 环以及在该 Chord 环上建立的 2-叉 UMPT

**定理 1.** 在一个由  $N$  个副本节点组成的 Chord 环上建立的  $d$ -叉 UMPT 的平均高度为  $O(\log_d N)$ , 且每个节点只出现一次.

**证明.** 假设 ID 空间大小为  $2^m$ , 即 ID 长度  $m$  位. 每一次分割所得 ID 空间是上一级区域的  $1/d$ , 所以经过  $\log_d N$  次划分后所得区域大小为  $2^m / (d)^{\log_d N}$ , 即  $2^m / N$ . 由于 DHT 随机地把节点分布在 ID 空间上, 在大小为  $2^m / N$  区域内平均只有一个节点, 即不需要再次划分该区域, 所以  $d$ -叉 UMPT 的高度为  $O(\log_d N)$ .

在建立 UMPT 时, 每次划分所得区域都不包括

已经出现过的节点 ID, 所以每个节点只出现一次.

证毕.

事实上, 在每次分割区域时, 我们忽略没有节点的区域 (函数  $region\_partition$  的第 3 行), 所以定理 1 所得 UMPT 的高度是一个上限值. 定理 1 说明了每个副本节点在 UMPT 只出现一次, 这大大减小了节点过载的几率.

PATCOM 中, 一次更新所需时间包括建立 UMPT 的时间和传播更新消息的时间. 下面分析建立 UMPT 的时间.

**引理 1.** 由  $N$  个副本节点组成的 Chord 环,

如果把整个 ID 空间分割为大小相等的  $r$  个 ID 区域,则在分割所得区域内查找属于该区域的关键字的后继节点平均需要  $O(\log \frac{N}{r})$  跳。

证明. 关键字  $k$  的后继节点指在 ID 空间上节点 ID 第一个大于等于  $k$  的节点<sup>[4]</sup>. 假设 ID 空间大小为  $2^m$ , 即 ID 长度  $m$  位; 假设节点  $node$  和关键字  $k$  都在节点区域  $region$  内, 且  $k > node$ , 而  $region$  的大小是整个 ID 空间的  $1/r$ , 现在分析  $node$  查找  $k$  的后继节点时平均所需跳数. 在 Chord 中, 每一跳都可以使处理查询请求的节点到关键字  $k$  的距离减少一半<sup>[4]</sup>, 那么经过  $O(\log \frac{N}{r})$  跳后, 处理查询请求的节点到  $k$  的距离减少为  $(2^m/r)/2^{\log(N/r)} = 2^m/N$ . 由于 DHT 随机地把节点分布在整个 ID 空间上, 在大小为  $2^m/N$  区域内平均只有一个节点, 即找到了关键字  $k$  的后继节点. 证毕.

**定理 2.** 在一个由  $N$  个副本节点组成的 Chord 环上建立  $d$ -叉 UMPT, 平均需要  $O(\log^2 N/(2\log d))$  跳。

证明. 假设 UMPT 的高度为  $h$ , 则  $h = O(\log_d N)$ . 由引理 1 知, 在 UMPT 的第  $i$  ( $0 \leq i \leq h$ ) 层查找一个节点平均需要  $\log \frac{N}{d^i}$  跳. 由于区域的划分以及代表节点的获取是同时进行的, 所以建立 UMPT 平均需要的跳数为

$$\begin{aligned} \#hop(\text{build}) &= \log N + \log \frac{N}{d} + \cdots + \log \frac{N}{d^{h-1}} \\ &\approx h \log N - \log d \frac{h^2}{2} \\ &= \log^2 N / (2 \log d) \end{aligned} \quad (1)$$

定理得证.

证毕.

### 3.2 更新操作

更新消息在已建立的 UMPT 上传播: 根节点 (即发起更新操作的节点) 把更新消息传递给孩子节点. 孩子节点在收到更新消息后, 检查消息的合法性等, 然后对相关内容做适当的更新, 并把更新消息转发给自己的子节点. 以此类推, 直到 UMPT 的叶子节点. 更新消息一般有两种: 只发送所要更新的数据的最新版本号, 而内容则由副本节点自行获取; 或者是直接发送更新内容. 由于每个节点在 UMPT 上只出现一次, 故每个节点只会收到一个更新消息. 由此我们有定理 3.

**定理 3.** PATCOM 中, 在建立 UMPT 后, 平均在  $O(\log_d N)$  时间内完成  $N$  个副本节点的更新,

且没有冗余更新消息.

从定理 2 可以看出建立 UMPT 平均需要  $O(\log^2 N/(2\log d))$  跳, 而从定理 3 得知在 UMPT 传播更新消息平均需要  $O(\log N/\log d)$  跳, 所以整个更新过程平均需要的跳数如式 (2) 所示.

$$\begin{aligned} \#hop(\text{update}) &= O(\log^2 N/(2\log d)) + O(\log N/\log d) \\ &\approx O\left(\frac{\log^2 N + \log N}{\log d}\right) \end{aligned} \quad (2)$$

这说明 PATCOM 能够快速地向所有副本节点传播更新消息. 另外, 虽然可以通过增加 UMPT 中节点的出度 ( $d$ ) 减少更新过程所需时间, 但是从后面的开销分析看出, 增加  $d$  会增加算法的开销, 所以  $d$  的选择上需要折中考虑.

需要指出的是, UMPT 的建立和更新消息的传播可以同时进行, 即节点在确定其孩子节点后立即传播更新消息. 此时, 整个更新过程所需时间与 UMPT 的建立时间近似相等, 即整个更新过程可在  $\log^2 N$  跳内完成.

更新消息转发之后, 删除 UMPT, 减少维护 UMPT 的开销. 3.4 节将详细描述和分析删除 UMPT 的过程.

### 3.3 副本节点加入和离开

节点在获取数据  $data_k$  的副本后, 如果试图保持该副本的一致性时, 则加入组  $group_k$ . 加入步骤遵循 Chord<sup>[4]</sup> 协议: 初始化指针表 (finger table), 并更新其他受影响组成员的指针表. 文献 [4] 指出节点加入产生的消息数为  $O(\log^2 N)$ , 而初始化指针表以及更新其他节点指针表在  $O(\log^2 N)$  时间内完成. 在完成加入操作后, 加入节点与其指针表中的节点通信, 获取最新的副本信息. Chord 中, 节点指针表中所指的节点分布在大小为  $2^{m-1}$  的 ID 区域内 (整个 ID 空间为  $2^m$ ), 所以这些节点保存的副本都不是最新的概率非常小. 从上面的分析可以看出, 副本节点加入时主动地从组内其它节点获取最新的副本, 而不影响已经建立好的 UMPT (如果有的话).

节点删除副本或者离开 P2P 系统时, 将离开组  $group_k$ . 离开步骤遵循 Chord 协议: 更新其它节点的指针表, 并清空自己的指针表. 如果将要离开的副本节点  $r$  是 UMPT (如果有的话) 中的节点, PATCOM 选取  $r$  在 Chord 环中的后继节点  $s$  代替节点  $r$ , 即  $r$  把相关的信息传递给  $s$ , 并通知父节点和孩子节点这种改变. 这里选择后继节点作为代替节点是因为: (1) 假设在更新过程中没有节点加入到  $r$  和  $s$  之间, 则如果重新建立 UMPT,  $r$  的位置将会被

$s$  取代; (2) 指针表的第一项是该节点的后继节点, 定位快, 查找时间短. 从式(2)看出, 更新过程所需时间短, 更新过程中离开组的节点很少, 所以这种操作也是非常少的.

### 3.4 节点失效时的处理

在 PATCOM 中, 如果没有更新操作, 并不建立和维护 UMPT, 此时节点失效并不影响更新操作, 失效节点处理借助于 Chord 协议完成. 但是, 在更新过程中, UMPT 中的节点可能失效, 此时该节点的所有子孙节点可能收不到更新消息, 称这种现象为 UMPT 节点失效.

PATCOM 使用了一种简单、高效的方法处理 UMPT 节点失效, 以保证更新消息传播到所有副本节点. 在传播更新消息过程中, 每个副本节点保存其孩子节点的相关信息(包括地址信息以及对应的 ID 区域), 直到收到孩子节点  $c$  确认消息后, 副本节点  $r$  才删除对应于  $c$  的相关信息. 副本节点向父节点发送确认消息的条件是收到了其所有孩子节点的确认消息. 这是一个递归过程. 由于叶子节点不需要再传播更新消息, 所以叶子节点在收到更新消息后立即向父节点发送确认消息. 需要指出的是, 删除孩子节点的信息相当于销毁了以该孩子节点为根节点的子树, 所以随着自下向上的删除, UMPT 将逐渐被销毁.

每个副本节点维护一个计时器, 如果在计时器超时仍然没有收到孩子节点  $c$  的确认消息, 则重新在节点  $c$  代表的区域中选取代表节点, 再次向新选取的代表节点发送更新消息, 并重新启动计时器. 副本节点  $r$  的计时器的超时时间与  $r$  所代表的区域大小成正比, 即

$$timeout_r = T_0 \times \frac{region_r}{2^m} \quad (3)$$

其中  $m$  为 ID 长度,  $T_0$  是某个设计值(design parameter). 从式(3)可以看出父节点的超时时间是其孩子节点的  $d$  倍, 这主要是为了使得失效节点的处理尽量不涉及 UMPT 中的上层节点, 限制 UMPT 节点失效造成影响范围, 减少节点失效带来的冗余更新消息.

**定理 4.** 使用 PATCOM 算法对关键字  $k$  的  $N$  个副本节点更新时, 假设节点失效概率为  $p$ , 且相互独立, 则平均最多需要  $O(N(1 + p \times (\log_d N - 1)))$  更新消息.

证明. 根据副本节点失效的时间点可以把 UMPT 节点失效分为

(1) 在收到更新消息之前失效. 这时不会产生

多余的更新消息, 只是增加了建立 UMPT 时所需的 Chord 查询消息数;

(2) 在收到更新消息后、转发消息之前失效. 这时仍然不会产生多余的更新消息, 只是增加了更新所需时间;

(3) 在转发消息之后、收到所有孩子节点的响应消息之前失效. 这将会产生多余的更新消息.

在第(3)种情况下, 假设  $d$ -叉 UMPT 的高度为  $h$ , 则  $h = O(\log_d N)$ . 假设 UMPT 中的某个节点失效, 该节点属于第  $i$  ( $0 < i < h$ , 根节点和叶子节点的失效并不会产生冗余更新消息)层的概率  $p_i = \frac{d^i}{N}$ , 而当该节点在第  $i$  层时, 产生的冗余消息数为该节点的子孙节点个数, 即  $\#Msg\_rdt_i = \sum_{j=1}^{h-i} d^j = \frac{d[d^{h-i} - 1]}{d - 1}$ , 所以一个节点的失效平均产生的冗余更新消息数可以表示为

$$\begin{aligned} & \#Msg_{rdt} \\ &= \sum_{i=1}^{h-1} p_i \times \#Msg\_rdt_i \\ &= \sum_{i=1}^{h-1} \frac{d^i}{N} \times \frac{d[d^{h-i} - 1]}{d - 1} \\ &= \frac{d}{d - 1} \times \frac{1}{N} \times \sum_{i=1}^{h-1} [d^h - d^i] \\ &< \frac{d}{d - 1} \times \frac{1}{N} \times \sum_{i=1}^{h-1} d^h \\ &= \frac{d}{d - 1} \times (\log_d N - 1) \\ &\approx O(\log_d N - 1) \end{aligned} \quad (4)$$

若节点失效概率为  $p$ , 且相互独立, 那么总共需要的更新消息数为

$$\begin{aligned} & \#Msg(\text{update}) \\ &= N + N \times p \times \#Msg_{rdt} \\ &\approx O(N(1 + p \times (\log_d N - 1))) \end{aligned} \quad (5)$$

定理得证.

证毕.

从式(4)看出, 更新过程中, 一个节点的失效平均仅产生  $O(\log N)$  冗余更新消息, 所以 PATCOM 中的失效处理机制具有较好的扩展性.

### 3.5 算法开销分析

更新所用的时间以及更新消息的数目是衡量一致性维护算法的重要指标. 以下将从副本节点保存的状态信息、建立 UMPT 产生的消息数、副本节点加入和离开产生的消息这几方面分析 PATCOM 的性能, 并和基于 Gossip 协议的一致性维护算法作比较.

(1) 副本节点保存的状态信息

在 Chord 中,节点的指针表有  $O(m)$  行,而和节点个数无关,其中  $m$  为 ID 长度.当副本节点较少时,指针表中将会有多个行指向同一个节点,所以指针表可以进一步减小.

**引理 2.** 由  $N$  个节点组成 Chord 覆盖网络中,指针表中不同节点的数目平均为  $O(\log N)$ .

证明. 假设 ID 长度为  $m$ ,则 ID 空间大小为  $2^m$ .由于 DHT 随机地把节点分布在整个 ID 空间上,所以在大小为  $2^m/N$  区域内平均有一个节点.令  $q = \lfloor m - \log N \rfloor$ ,则指针表前  $q$  行以大概率(with high probability)保存同一个节点的信息.所以指针表中不同节点的数目平均为  $O(1 + m - (m - \log N)) \approx O(\log N)$ . 证毕.

PATCOM 中关键字  $k$  的副本节点只保存  $O(\log N)$  不同节点的信息.另外,根据 Chord 协议,为了保证在有节点失效情况下的查询性能,每个副本节点还保存连续的  $r$  个后继节点,当某个后继节点失效时,从后继节点列表中选择下一个节点代替失效节点.若单个节点失效的概率为  $p$ ,且相互独立,则后继节点列表中的  $r$  个节点全部失效的概率为  $p^r$ ,增强了系统的容错能力.故有如下定理.

**定理 5.** PATCOM 中,每个副本节点平均保存  $O(\log N + r)$  其他副本节点的信息.

传统的 Gossip 协议假设消息的下一跳节点是从系统中随机选择的,这要求节点保存系统中所有其他节点的信息,所以扩展性差.文献[10]提出了一种扩展性较好的 Gossip 协议——SCAMP. SCAMP 中,每个节点只需保存  $O((c+1)\log N)$  其他节点的信息就可以达到和传统 Gossip 协议相近的性能.所以 PATCOM 所使用的成员管理协议 Chord 和基于 Gossip 的组成员管理协议对存储空间要求基本相同.

## (2) Chord 环维护

UMPT 的维护已经在 3.4 节和 3.5 节进行了详细的分析,这里分析无更新操作时 Chord 环的维护.

### ① 副本节点加入和离开

Chord 协议中,节点加入和离开平均产生  $O(\log^2 N)$  消息, $N$  为覆盖网络中节点数目.而在 SCAMP 中,节点加入消息平均需要传播  $O((c+1)\log N)$  轮,而每轮传播都是向  $O((c+1)\log N)$  节点发送消息,故加入时产生的消息数平均为  $O((c+1)^2 \log^2 N)$ ;但节点的离开只影响  $O((c+1)\log N)$  节点,所以节点离开时平均产生  $O((c+1)\log N)$  消息.

从副本节点离开和加入系统产生的消息数看,

SCAMP 和 PATCOM 中的成员管理协议 Chord 都有较好的扩展性.但在节点离开时,SCAMP 比 Chord 产生更少的消息.考虑到  $O(\log^2 N)$  本身比较小,而且 PATCOM 能够保证在不产生冗余更新消息的情况下保证所有节点都能够收到更新消息,有理由认为使用 Chord 协议作为 PATCOM 中的成员管理协议是合理的.

### ② 副本节点失效

当没有更新操作时,借助 Chord 协议的容错机制处理副本节点失效.在 Chord 协议中,每个节点保存连续的  $r$  个后继节点.当节点的后继节点失效时,从后继节点列表中选择下一个节点代替失效节点.节点周期性地检查和更新指针表(finger table),每个周期更新指针表中的一项,这将会产生  $O(\log N)$  个查询消息.而在 SCAMP 中,为了保证覆盖网络的连通性,节点周期性地与其邻居节点通信.由于 SCAMP 中,每个节点平均有  $O(\log N)$  个邻居节点,所以会产生  $O(\log N)$  个消息.

从容错机制的开销看,可扩展的 Gossip 协议 SCAMP 和 PATCOM 中的成员管理协议 Chord 的性能近似.

### (3) 建立 UMPT 产生的消息数

PATCOM 中的另外一部分额外的开销是建立 UMPT 产生的 Chord 查询消息.对于产生的消息数有如下定理.

**定理 6.** PATCOM 中,假设关键字  $k$  的副本节点个数为  $N$ ,则建立  $d$ -叉 UMPT 时,平均产生  $O(N \times \log d)$  Chord 查询消息.

证明. 假设整个 ID 区域大小为  $2^m$  ( $m$  为 ID 长度),根据引理 1,在大小为  $2^m/r$  的区域内的定位操作平均使用  $O\left(\log \frac{N}{r}\right)$  查询消息.假设  $d$ -叉 UMPT 的高度为  $h$ ,则  $h = O(\log_d N)$ . UMPT 中,第  $i$  ( $0 \leq i \leq h$ ) 层节点代表的区域平均大小为  $2^m/d^i$ ,而每个节点平均有  $d$  个孩子节点.所以在建立 UMPT 时,第  $i$  ( $0 \leq i \leq h$ ) 层的一个节点平均产生  $\left(d \times \log \frac{N}{d^i}\right)$  查询消息.因此,建立 UMPT 共使用的消息数如式(6)所示(注:建立 UMPT 时,叶子节点不会产生 Chord 查询消息).

$$\#Msg(\text{build})$$

$$\begin{aligned} &= d \times \log N + d^2 \times \log \frac{N}{d} + \cdots + d^h \times \log \frac{N}{d^{h-1}} \\ &= \log N \times \sum_{i=1}^h d^i - \log d \times \sum_{i=1}^{h-1} i d^{i+1} \\ &= \log N \times \frac{d(d^h - 1)}{d - 1} - \end{aligned}$$



$$\begin{aligned}
& \log d \times \left( \frac{(h-1)d^{h+1}}{d-1} - \frac{d^2(d^{h-1}-1)}{(d-1)^2} \right) \\
& \approx N \times \frac{d}{d-1} \times \log d \\
& \approx O(N \times \log d)
\end{aligned} \quad (6)$$

定理得证。

证毕。

从定理 6 可以看出,建立 UMPT 时,每个节点平均产生  $O(\log d)$  消息,和副本节点个数无关,所以  $d$  越小,建立 UMPT 的开销就越小。但是从式(2)看出,减小  $d$ ,将会增加整个更新过程(包括建立 UMPT 和传播更新消息的过程)所需的时间,所以在 UMPT 出度的选择上要有一定的权衡,从仿真实验结果看, $d$  取 8 是合理的选择。

建立 UMPT 产生的 Chord 查询消息是 PATCOM 比基于 Gossip 协议的一致性维护算法多出的开销。注意到 Chord 查询消息(仅包括所要查询的  $id$ 、发起查询节点的地址标识)的大小与更新消息相比比较小,而且 PATCOM 不会产生冗余更新消息,所以我们认为这种开销是可以接受的,第 4 节的仿真实验也验证了这点。

## 4 仿真实验和性能评估

为了验证 PATCOM 的性能并和基于 Gossip 协议的一致性维护算法进行比较,使用 C++ 编写了 Chord 协议<sup>[4]</sup>和 SCAMP 协议<sup>[10]</sup>的模拟程序,实现了算法 1 所示的区域分割伪代码,模拟了节点加入和离开、更新消息传播以及节点失效处理过程。UMPT 的缺省出度为 8,即  $d=8$ 。为了比较结果的公平性,基于 Gossip 协议更新方法中,节点扇出(fanout)和 UMPT 的出度相同,缺省为 8。另外,在基于 Gossip 协议的更新方法中,当超过 95% 的节点得到更新后,停止继续传播更新消息,这主要是为了减少冗余更新消息。

需要说明的是,后面图中的每个数据点都是 10 次实验所得结果的平均值,“push/pull”指文献[6]提出的基于 Gossip 的“推”、“拉”结合的一致性维护方法。

### (1) UMPT 的高度

图 2 显示了 UMPT 的高度和副本节点个数以及节点出度的关系,其中横坐标是对数坐标。从图中可以看出 UMPT 的高度和副本节点个数成对数关系,而和 UMPT 中节点的出度成对数反比关系。这验证了定理 1,并间接地验证了定理 2 和定理 3。

### (2) 节点保存的副本节点数目

节点保存的副本节点数目反映了 PATCOM 对存储空间的需求程度。如 3.6 节所述,为了减少指针表大小, PATCOM 中副本节点只保存不同节点的信息。图 3 显示了精简后指针表大小的分布图,其中“#replicas”是指副本节点的个数。图中没有显示大于 22 的点,因为在实验结果中指针表最大为 22。图 3 中纵坐标是对数坐标。从图中可以看出,每个节点平均保存  $O(\log N)$  其他节点的信息,具有较好的扩展能力。这里,我们没有考虑更新期间为了维护 UMPT 所需的存储空间。这一方面是因为更新时间很短,维护 UMPT 所需时间较短;另一方面,为了维护 UMPT 每个节点保存的信息数目为  $(d+1)$  ( $d$  个孩子节点和 1 个父节点),并不随副本节点数目的增加而增加。

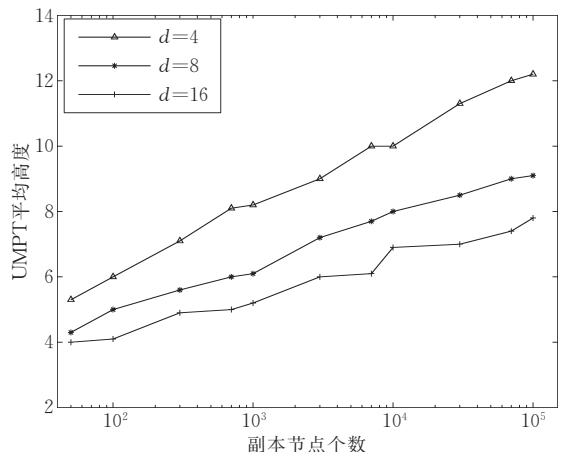


图 2 UMPT 高度

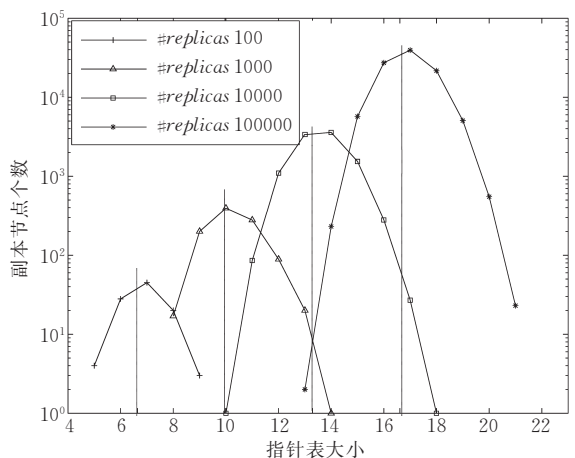


图 3 节点保存的副本节点数目

(3) 建立 UMPT 所需的 Chord 查询消息的数目  
与基于 Gossip 的一致性维护算法相比, PATCOM 除了使用组管理协议 Chord 外,还需要动态地建立 UMPT。在建立 UMPT 时,会产生查询消息。图 4 显示了建立 UMPT 所需的 Chord 查询消息



的数目与副本节点个数以及 UMPT 中节点出度之间的关系,其中横坐标为对数坐标.从图 4 可以看出,随着副本节点数目的增加,平均每个节点产生的查询消息数目基本保持不变,而平均每个节点产生的查询消息数目和 UMPT 中节点出度大致上成对数关系.这验证了定理 6.

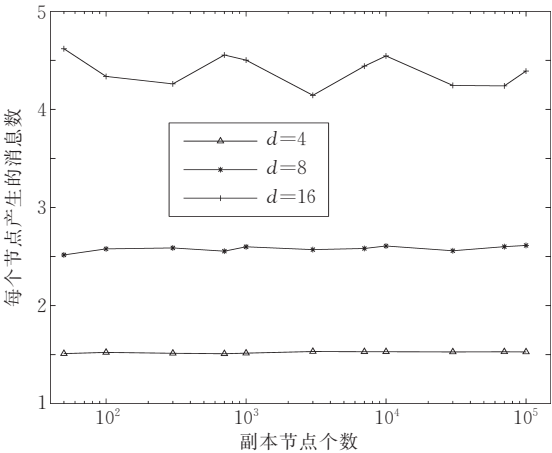


图 4 建立 UMPT 所需的消息数

(4) 更新操作所需消息数目

PATCOM 能保证所有副本节点的一致,但我们并不希望产生过多的消息.在正常情况下,建立 UMPT 平均每个节点产生  $O(\log d)$  查询消息数(定理 6),而每个节点只会收到  $O(1)$  更新消息(定理 3),所以一次更新平均每个节点产生的消息数为  $(\log d + 1)(\log d \text{ 查询消息和 } 1 \text{ 个更新消息})$ .图 5 验证了这个结论,并和基于 Gossip 的“推”、“拉”结合的一致性算法进行了比较,其中横坐标为对数坐标.图中纵坐标表示平均每个节点产生的消息数目,对于 PATCOM 包括更新消息和建立 UMPT 所需的查询消息,而对于“推”、“拉”结合的一致性算法只包括更新消息.可以看出,两种算法中,平均每个副本

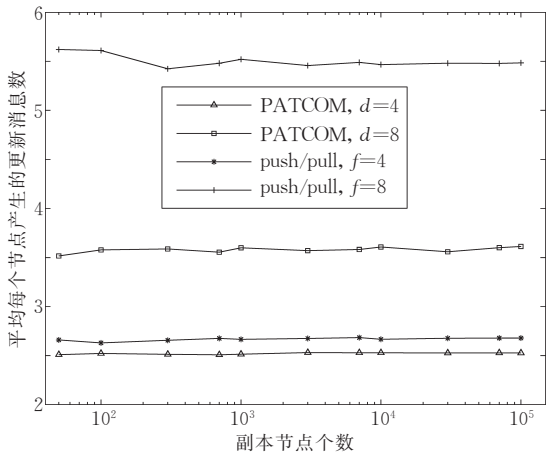


图 5 更新操作所需的消息数

节点产生的消息并不随副本节点数目的变化而变化,但“推”、“拉”结合的一致性算法对出度的变化比较敏感:出度为 4 时,每个节点平均产生 2.6 个更新消息;而出度为 8 时,每个节点平均产生 5.5 个更新消息,呈线性增长趋势.所以在这种算法中需使用较低的出度,但减小出度会增加更新操作所需时间.而 PATCOM 中,消息个数和出度成对数关系.从图中还可以看出,无论哪种情况,“推”、“拉”结合的算法产生的消息数目都大于 PATCOM 的消息数目.另外需要指出的是,“推”、“拉”结合的算法只能保证副本节点的概率性一致.

(5) 更新所需消息的大小

前面比较了 PATCOM 和“推”、“拉”结合算法产生的消息数目,这部分分析消息大小.更新消息一般有两种:只发送所要更新的数据的最新版本号,而内容由副本节点自行获取;或者是直接发送更新内容.我们分别分析对于不同类型的更新消息平均每个节点产生的消息大小.设定 Chord 查询消息大小为 27 字节;20 字节标识所要查询的  $id$ 、1 个字节标识是查询消息、6 个字节标识发起查询节点的地址信息(IP 地址和端口号).对于第一种更新消息(只发送所要更新数据的最新版本号),消息大小为 40 字节;32 字节的版本号向量<sup>[6]</sup>和 8 字节的更新内容描述(包括验证码);对于第二种更新消息(发送副本内容),我们设定消息大小为 64KB.图 6 和图 7 分别显示了更新消息大小为 40 字节和 64KB 时平均每个节点产生的消息的变化趋势,其中横坐标和纵坐标都是对数坐标.比较图 6 和图 5 我们可以看出“推”、“拉”结合算法消息大小和消息数不成正比关系.这是因为在“推”、“拉”结合算法中,为了减少冗余更新消息以及学习更多的节点信息,更新消息包

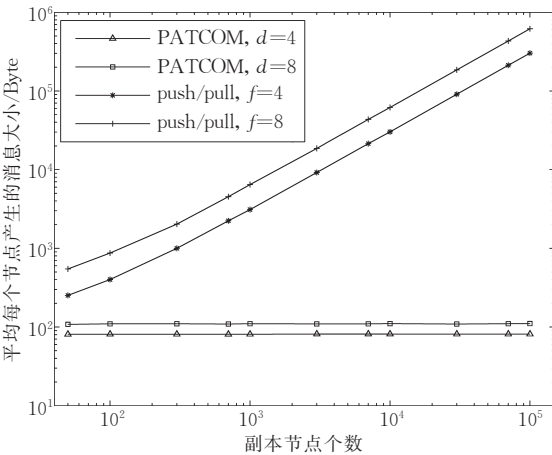


图 6 更新所需消息大小(更新消息大小: 40B, 查询消息大小: 27B)

含了该消息经过的副本节点的地址信息(IP 地址). 所以消息大小会随着副本节点增加而增加. 图 7 显示了类似的情形,但此时更新消息较大(64KB),所以在副本节点数目较少(小于  $10^4$ )时节点地址信息的增加带来的影响相对较小,更新消息增长相对缓慢. PATCOM 中,更新消息并不需要包含经过的节点的地址信息,所以消息大小和消息数目成正比关系. 图 6 和图 7 都说明了与基于 Gossip 的“推”、“拉”结合的一致性维护方法相比, PATCOM 的开销很小.

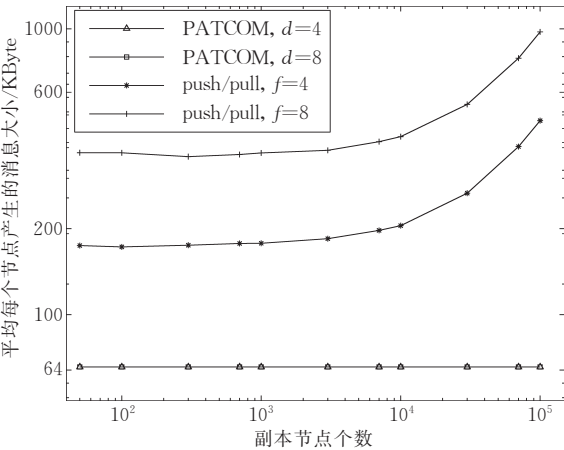


图 7 更新所需消息大小(更新消息大小:64KB, 查询消息大小:27B)

(6) 有节点失效情况下的更新消息数目

这部分验证在有节点失效情况下 PATCOM 的性能. 图 8 和图 9 中,“# replicas”是指副本的个数,而纵坐标为平均每个节点产生的更新消息数,不包括重建 UMPT 所需消息,这主要是因为从图 6 和图 7 看出建立 UMPT 所需消息对算法开销影响相对较小. 另外,每个节点产生的更新消息是总的更新消息数目和存活的节点数目的比值. 图 8 说明了

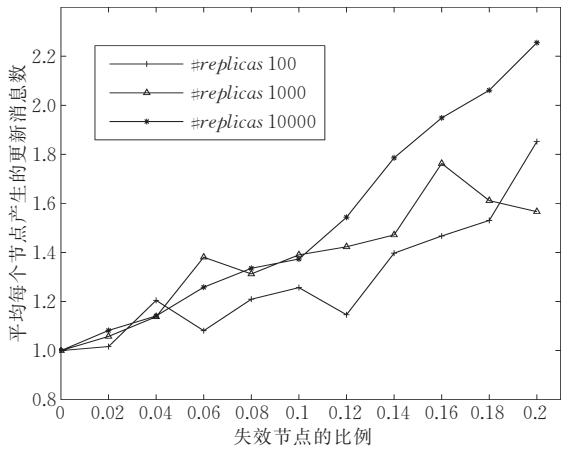


图 8 失效情况下更新的消息数目(PATCOM, d=8)

PATCOM 中更新消息数目随副本节点个数大致上呈对数关系,而和失效率呈线性关系,这验证了定理 4. 图 9 比较了 PATCOM 和“推”、“拉”结合算法. 可以看出基于 Gossip 的算法有较高的容错能力,更新消息数随失效率变化很小. 对于 PATCOM,更新消息数和节点出度大致呈对数反比关系,这也验证了定理 4. 但即使是失效率高达 0.2 时, PATCOM 仍然比“推”、“拉”结合算法有较低的开销:节点出度为 4 时,对于 PATCOM,平均每个节点产生 2.1 个更新消息,而对于“推”、“拉”结合算法,平均每个节点产生 2.8 个更新消息;节点出度为 8 时,对于 PATCOM,平均每个节点产生 1.6 个更新消息,而对于“推”、“拉”结合算法,平均每个节点产生 5.9 个更新消息.

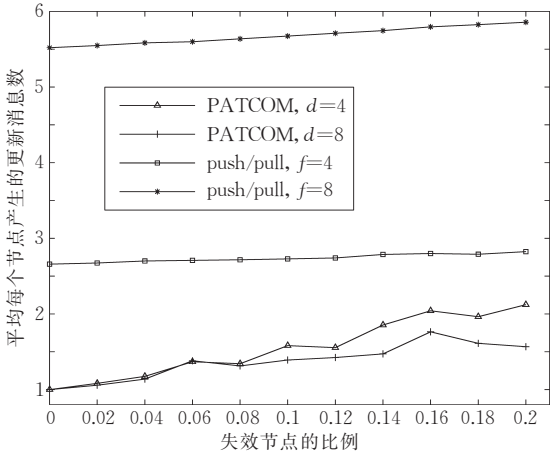


图 9 失效情况下更新的消息数目比较(#replicas 1000)

从上面的实验结果分析看出,在实际的应用中, UMPT 的出度  $d$  设为 8 是比较合理的选择.

5 结 论

为了提高副本维护一致性速度与减少冗余更新消息,针对无结构 P2P 系统,本文提出了一种高效的一致性维护方法——PATCOM. 该方法的目的在于:在尽量不产生冗余更新消息的条件下,快速地保证所有副本的一致. PATCOM 使用 Chord 协议作为组管理协议,通过在 Chord 环上动态地建立更新消息传播树完成更新消息的传播. 对于关键字  $k$  的  $N$  个副本节点,一次更新操作平均需要  $O(\log^2 N + \log N)$  跳,每个节点平均产生  $O(1)$  更新消息,且每个节点平均维护  $O(\log N)$  其他副本节点的信息;更新过程中,一个节点的失效平均产生  $O(\log N)$  冗余更新消息. 所以, PATCOM 具有良好的扩展性和容错能力. 与基于 Gossip 的一致性维护方法相比,

PATCOM 不仅能快速地维护所有副本的一致,而且产生的冗余更新消息少.

下一步的工作将考虑位置感知的一致性维护方法. 位置感知的一致性维护方法旨在使更新消息在物理网络中距离近的节点之间传输,这不仅可以进一步加快副本的一致性维护,而且能够节省一致性维护方法对骨干网的带宽消耗. 另外,文献[12]中提出的分布式负载均衡方法为异构环境下的一致性维护提供了依据. 这些工作将会使 PATCOM 更加实用.

参 考 文 献

[1] Clarke I, Sandberg O, Wiley B, Hong T W. Freenet: A distributed anonymous information storage and retrieval system//Federrath H ed. Proceedings of the Workshop on Design Issues in Anonymity and Unobservability. Berlin: Springer-Verlag, 2000: 46-66

[2] Aberer K, Despotovic Z. Managing trust in a Peer-2-Peer information system//Proceedings of the 10th International Conference on Information and Knowledge Management. New York, USA, 2001: 310-317

[3] Ratnasamy S, Francis P, Handley M, Karp R. A scalable content-addressable network//Proceedings of the SIGCOMM 2001. San Diego, CA, USA, 2001: 161-172

[4] Stoica I, Morris R, Karger D, Kaashoek M, Balakrishnan H. Chord: A scalable Peer-to-Peer lookup service for internet applications//Proceedings of the SIGCOMM 2001. San Deigo, CA, USA, 2001: 149-160

[5] Dabek F, Kaashoek M F, Karger D, Morris R, Stoica I. Wide-area cooperative storage with CFS//Proceedings of the 18th ACM Symposium Operating Systems Principles (SOSP). Banff, Canada, 2001: 202-215

[6] Datta A, Hauswirth M, Aberer K. Updates in highly unreliable, replicated Peer-to-Peer systems//Proceedings of the 23rd International Conference on Distributed Computing Systems. Washington, 2003: 76-85

[7] Chen X, Ren S S, Wang H N, Zhang X D. SCOPE: Scalable consistency maintenance in structured P2P systems//Proceedings of the IEEE Infocom 2005. Washington, 2005: 1502-1513

[8] Duvvuri V, Shenoy P, Tewari R. Adaptive leases: A strong consistency mechanism for the World Wide Web//Proceedings of the IEEE INFOCOM 2000. Tel Aviv, Israel, 2000: 834-843

[9] Yin J, Alvisi L, Dahlin M, Lin C. Hierarchical cache consistency in a WAN//Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems (USITS). Colorado, USA, 1999: 13-24

[10] Ganesh A J, Kermarrec A-M, Massouli L. Peer-to-Peer membership management for gossip-based protocols. IEEE Transactions on Computers, 2003, 52(2): 139-149

[11] Lan J, Liu X, Shenoy P, Ramamritham K. Consistency maintenance in Peer-to-Peer file sharing networks//Proceedings of the 3rd IEEE Workshop on Internet Applications. Washington, USA, 2002: 90-94

[12] Li Z, Xie G. A distributed load balancing algorithm for structured P2P systems//Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC' 2006). Pula, Italy, 2006: 417-422



**LI Zhen-Yu**, born in 1980, Ph. D. candidate. His interests include Peer-to-Peer system, network measurement and management.

**XIE Gao-Gang**, born in 1974, Ph. D. , associate professor. His main research interests include Peer-to-Peer computing, network measurement and management.

**LI Zhong-Cheng**, born in 1962, Ph. D. , professor, Ph.D. supervisor. His main research interests include computer networks, testing and dependable computing.

Background

This work is supported by the National Natural Science Foundation of China under grant No.60403031 and No.90604015, by the National High Technology Research

and Development Program (863 Program) of China under grant No. 2005AA121560.