ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.1, January 2009, pp.80−95
doi: 10.3724/SP.J.1001.2009.03382

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

Peer-to-Peer 存储系统中一种高效的数据维护方案*

杨 智+, 朱 君, 代亚非

(北京大学 计算机科学技术系,北京 100871)

Efficient Data Maintenance Scheme for Peer-to-Peer Storage Systems

YANG Zhi+, ZHU Jun, DAI Ya-Fei

(Department of Computer Science and Technology, Peking University, Beijing 100871, China)

+ Corresponding author: E-mail: yangzhi@net.pku.edu.cn

Yang Z, Zhu J, Dai YF. Efficient data maintenance scheme for peer-to-peer storage systems. *Journal of Software*, 2009,20(1):80−95. http://www.jos.org.cn/1000-9825/3382.htm

Abstract: This paper presents a new scheme based on the dynamical characteristics of P2P systems. First, the differences in node availability are taken into consideration, and a new data placement algorithm using less redundancy to guarantee target data availability is proposed. Second, permanent failure detector is used to distinguish between permanent and transient failures, which can reduce data recovery cost by decreasing the number of unnecessary repairs. Results from a trace-driven simulation suggest that this scheme can reduce about 80% maintenance bandwidth, compared with traditional methods.

Key words: peer-to-peer storage; data maintenance; permanent failure; transient failure; failure detection; bandwidth optimization

摘 要: 提出一套完整的数据维护方案.该方案建立在 P2P 环境动态性特点的基础上.一方面,该方案考虑了节点动态性差异,它基于不同的动态性作相应的数据冗余,能够用更少的冗余开销来保证数据的目标可用性;另一方面,该方案给出如何利用判别器来区分永久失效和暂时失效,以减少由于不必要的数据修复而带来的额外修复开销.通过在真实 P2P 系统 Maze 上的实验结果表明,该方案比目前主流的方案能够节省大约 80%的数据维护带宽.

关键词: 对等网络存储;数据维护;永久失效;暂时失效;失效检测;带宽优化

中图法分类号: TP311 **文献标识码**: A

对等网络(peer-to-peer,简称 P2P)技术在即时通信、文件共享及流媒体传输等应用领域均显示出了极大的优势,成为构建新型大规模互联网应用的主要结构和技术之一.P2P 存储系统相比传统的存储系统有如下优点:不依赖中央控制,系统自然具有高扩展性,且不存在单点性能瓶颈问题;各个节点功能对等,使得整个系统在缺失任意节点后仍能正常工作,也即具有高容错性.使得 P2P 存储系统一直被认为是一种极富前景的应用,是学术界研究的热点.到目前为止,已有大量 P2P 存储系统的知名设计,如文献[1−5]等,然而,P2P 存储应用仍然面临着巨大的技术挑战,并没有获得相应的商业成功.

* Supported by the National Natural Science Foundation of China under Grant No.60873051 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z154 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2004CB318204 (国家重点基础研究发展计划(973))

Received 2008-01-29; Accepted 2008-04-30

文献[6,7]指出,限制 P2P 存储系统发展的主要原因是其运行环境中节点的动态性远高于任何经典的存储环境,尤其出现了大量节点的暂时性失效,而经典存储系统一般仅考虑永久失效的情况.这也使得其数据可用性研究面临新的问题和挑战,需要研究更新的适用于 P2P 系统这种高动态环境的方案.这里的高动态是指系统中节点的平均可用性很低(导致大量的暂时失效),并且节点的平均寿命很短(导致大量的永久失效).目前,对数据高可用性的分析模型和解决方案都还处于起步阶段,本文的研究也表明,已有的方案给系统带来了极高的数据维护开销,不能很好地适用于 P2P 存储的高动态运行环境.事实上,目前还没有大规模真正实用的 P2P 存储系统,这很大程度上是由于受到高数据维护开销的制约[6].

本文提出一种能够极大降低系统开销的数据可用性维护方案,更好地适用于高动态的 P2P 存储系统.本文基于 P2P 系统节点的动态性规律,研究存储节点的选择方案,以降低系统屏蔽暂时失效所需的数据冗余度,同时;本文将研究如何使用永久失效判别器来检测系统中节点的永久失效,以降低由于系统无法区分暂时失效和永久失效所带来的额外开销.本文的主要贡献如下:

(1) 本文考虑了节点的动态性差异,提出一种基于节点动态性的数据分发算法,它把数据放在动态性相近的节点上,并精确计算出相应的冗余度.相对于传统的随机分发算法,能够在 P2P 这种高动态环境下用更少的冗余开销达到数据的目标可用性.

(2) 如何在时间阈值判别器中设置时间阈值来区分永久和暂时失效是当前 P2P 存储系统研究中的一个难题,本文基于漏判抵消误判的思想,给出了永久失效判别器中参数设置的统一原则,使得时间阈值判别器的性能接近于最优的理想判别器.

(3) 为了提高判别器精度,本文提出了一种新的永久失效判别器:基于节点离开概率的判别器.它在不同的离线长度下给出相应的永久失效概率,而不是只用一个阈值来确定性地判别永久失效,因此,本文提出的判别器比时间阈值判别器更为精确.

本文利用真实的 P2P 系统(Maze)运行日志来构造系统的运行环境,实验结果表明,本文提出的算法相对于主流算法可以降低接近 80%的维护带宽.

本文第 1 节介绍当前 P2P 存储系统中的维护数据可用性的关键技术及存在的问题.第 2 节给出系统的开销模型,指出降低系统开销的关键是降低两个参数:目标冗余度和额外冗余度.第 3 节和第 4 节分别给出降低目标冗余度和额外冗余度的方法.第 5 节给出实验分析.第 6 节总结全文.

1 P2P 存储系统中数据的可用性维护

本节首先介绍了 P2P 存储系统中维护数据可用性的相关研究,给出了本文的研究背景和主要研究的问题.其次,根据现有的 P2P 存储系统实例,本节构造一个抽象的 P2P 存储系统来说明本文研究所依据的系统模型.

1.1 维护数据可用性的研究

对于P2P系统中数据可用性的研究必须以系统中节点的动态性规律为基础,认识P2P系统中节点的动态性非常重要.文献[8–10]测量了目前流行的 P2P 系统,描述了其主要的动态特征.其中,节点的高失效率是制约 P2P 存储系统发展的关键问题.由于 P2P 系统中每个节点可以自由地加入或离开系统,使得系统中出现节点失效的情况明显高于其他类型的存储系统.在 P2P 系统中,任意节点都可能出现暂时失效(例如,网络的暂时拥塞、用户的暂时离线)和永久失效(例如,磁盘失效、用户的永久退出)[7−10].暂时失效只会造成数据在一段时间内不可访问,并不会造成数据丢失.而永久失效会导致数据丢失,造成数据的永久不可访问.因此,两种类型的失效都会威胁数据的可用性.这就需要 P2P 存储系统采用冗余和修复算法屏蔽节点失效对数据可用性的影响.目前,P2P 存储系统中数据的可用性研究主要集中在这两个方面.

1.1.1 数据冗余

数据冗余的主要目的是屏蔽系统中节点的暂时失效,能够让系统在没有永久失效时保证数据的目标可用性[1,6,11].它可以看作是对数据对象的一个配置:定义了数据的冗余方式、放置策略和冗余度.

冗余方式主要包含两类:副本和纠删码(erasure code).参数是(m,n)的纠删码将原始文件分成 m 个碎片,然后

编码成 $n(n>m)$个碎片,并保证其中任意 m 个碎片都可以重构文件[12].最早人们用量化的方法分析了纠删码和副本方式冗余对系统可靠性的影响,但考虑的是较为稳定的环境[13].之后,研究者重新考察了纠删码对 P2P 存储系统可用性的影响,更关注高动态性的系统,指出对于节点可用性比较低的系统,用副本的方式较好,但并没有考虑修复带宽开销[14].关注在 P2P 系统中两种冗余方式下的修复开销,指出副本方式在高动态系统中更为有效.更为复杂的方式是采用副本和纠删码结合的方式,文献[15]的研究表明,这种方式能够更好地优化修复带宽.

数据放置策略决定一个数据对象的副本存放在哪些节点上.数据放置策略可以分为两类:随机放置策略和选择放置策略.随机放置策略实现简单,是现在主流的数据放置策略[1,6,11,12,16].选择放置策略选择那些满足某些约束条件的节点来存放数据(例如,选择错误相关性比较低[11],或者高可用性的节点来存放数据[17−20]).但选择放置策略通常会使数据的可用性计算复杂,或者使系统负载不平衡(例如,选择高可用性的节点放置数据趋向于耗尽高在线概率节点的资源,会导致用户通过减少自身节点可用性来避免承担过多负载,从而进一步增加了系统动态性,故不实用).

冗余度定义为冗余后的数据量与原始数据量的比值.对于副本方式,冗余度就等于数据副本的个数.对于参数是(m,n)的纠删码,冗余度为 n/m.目标冗余度是指能够达到数据目标可用性的最小冗余度[1,2,6,11].目标冗余度由数据的目标可用性、冗余方式以及数据放置策略决定.文献[1,2,6,11]给出在随机放置策略下,对于给定的目标可用性,不同冗余方式的冗余度计算方法.冗余度体现了数据冗余策略的效率,高效的冗余策略可以用较低的冗余度保证数据的目标可用性.

目前,主流的数据冗余策略是随机分发数据,然后利用节点在线概率的平均值计算出数据的目标冗余度[1,6,11,12,16].这种方法具有实现简单和冗余度计算模型简单的优点,但是,这种方法却忽略了节点间动态性的差异.本文提出一种基于节点动态性的数据分发算法,它采用与主流方法相同的可用性计算模型,但根据不同的节点动态性产生不同的冗余度,更好地适用于节点动态性差异较大的 P2P 系统.同时,与其他启发式数据分发算法[17−20]不同,本文提出的算法仍能够保证系统的负载平衡,并保证高可用的节点会获得较好的服务质量,激励节点提高自身的在线概率,使整个系统能够进一步减少维护开销.

1.1.2 数据修复

根据现有测量的结果,实际 P2P 系统中的节点永久失效也是非常频繁的[6−10].如果不修复永久失效节点上的副本,则数据冗余度和数据可用性必将逐渐降低,并最终导致最终数据丢失.数据修复针对的是系统中节点的永久失效,系统通过不断修复永久失效节点上的数据来保证目标冗余度.

在数据修复的过程中,最为关键的问题是对节点永久失效的判断[7,8,11,21].在 P2P 系统中,当发现一个节点下线时,系统无法知道其是否会再次上线,即系统无法及时、准确地知道节点是暂时失效还是永久失效.一方面,若系统把暂时失效误判为永久失效并触发修复,则将浪费系统带宽资源.因为节点的暂时失效不会降低数据的冗余度,系统不需要复修就能保证数据的目标可用性.另一方面,若系统把永久失效漏判为暂时失效,则系统不能及时修复丢失的副本,数据的冗余度将低于目标冗余度,造成数据的可用性降低.若在更长的时间内不修复永久丢失的副本,则可能造成同时多个副本永久失效,将威胁存储数据的可靠性.

早期的分布式存储系统[5]采用立即修复策略,即一旦发现节点失效(暂时或永久)马上进行修复.目前,大多数系统[1,4,11,22,23]采用预先加入额外冗余数据的方式来延迟修复,以减少由于暂时失效触发的修复次数.例如,为了保证数据的可用性水平需要 R_1 倍的冗余,系统就会为数据做 R_1+R_2 倍的冗余,只有当系统观察到在线的数据冗余度比 R_1 低的时候才开始修复数据[21].建议额外的冗余数据可以在系统运行时动态加入,并用实验表明动态加入的方式比 TotalRecall[1]采用的预选加入的方式更优.然而,由于没有区分暂时失效和永久失效,延迟修复仍然会浪费大量的系统带宽.因为系统为了保证观察到在线的数据冗余度不低于 R_L,所以需要维护 R_1+R_2 的冗余度,而事实上,系统只需要 R_1 的冗余度屏蔽暂时失效.

研究[6,21]表明,采用时间阈值判别器(即当节点离线时间超过阈值 τ 时就判定其永久失效,反之判定其为暂时失效)来区分永久失效和暂时失效能够进一步降低系统的开销.然而,时间阈值 τ 如果增大,则导致对永久离开的漏判率增高,误判率降低;反之,τ 减小则漏判率降低,误判率增高.因此,阈值的选择要在漏判率和误判率之间加

以折衷.根据节点的性质合理地选取判断所用的时间阈值是一个非常困难的事情,目前仍没有研究明确给出阈值的选取方案,这也是目前大多数系统没有采用时间阈值判别器的主要原因.本文将回答如何自动、合理地设置时间阈值τ的难题.另外,还提出一种新的永久错误判别器:基于概率的判别器.基于真实 P2P 系统运行日志的模拟实验表明,这两种判别器的性能都接近最优的理想判断器(没有误判和漏判的判断器).

1.2 系统的基本模型

本节给出一个抽象的可维护数据可用性的 P2P 存储系统,是现有大多数 P2P 存储系统的基本系统模型.本文下面提出的系统优化算法都将基于该抽象系统模型,而不是针对某个具体的系统实例,以保证优化策略可应用于任何符合抽象模型的系统实例.为了保证数据的可用性,P2P 存储系统可以抽象地看作由以下元素构成:(1) 算法构件:主要是数据的冗余和修复算法,本文在上一节对此进行了详细介绍.(2) 服务构件:为实现算法构件提供服务,包括数据目录服务和监控服务.数据目录服务负责维护所有对象的每个副本与系统中节点的映射关系;节点监控服务用来监控节点的动态行为,它维护了系统中每个节点当前的在线状态.服务的提供可以是集中式的(由 1 个节点统一提供),也可以是分布式的(由许多节点共同提供,每个节点负责部分对象的目录或部分节点的监控).本文优化的目标是系统的算法构件.

例如,图 1 给出 P2P 存储系统实例 Total Recall[1]的系统结构图.系统中的算法构件可以分为两部分:a) 冗余算法,包括 Erasure code、副本冗余方式和数据的随机分发算法.b) 修复算法,包括积极修复和懒惰修复算法.服务构件也分为两部分:a) 数据目录服务,Total Recall 中,每个对象都由一个节点来负责,称为 Master.Master 知道它所负责的对象的每个副本存放在哪个节点上,维护了副本与系统中节点的映射关系.当用户(对应图中的 client)读取数据时,首先联系 Master,获取副本的位置,然后再联系存储副本的节点(对应图中的 Data storage hosts)下载.b) 节点监控服务,系统中存储每个存储副本的节点向其对应的存储索引的 Master 节点定期发送心跳信息,汇报自己的在线状态.因此,Master 也维护了节点的在线状态,它同时是目录服务和监控服务的提供者.当用户读取数据时,Master 可以用这两种信息告诉用户一个当前可用的副本的位置.

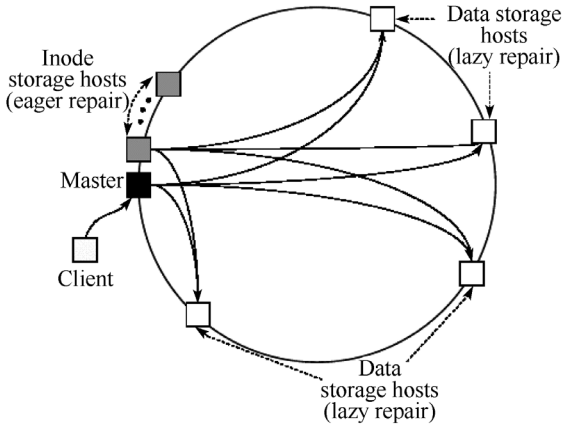


Fig.1 Architecture of Total Recall

图 1 Total Recall 系统结构

系统的运行过程如下:数据对象最初存入系统时,冗余算法被调用.冗余算法根据数据的目标可用性并通过监控服务统计出的节点动态性为对象产生一个配置,包括冗余方式、冗余度和放置策略.当数据存入系统后,系统通过目录服务建立副本和节点的对应关系.对于每个数据对象,系统定期地通过目录服务和监控服务查看存放其副本的节点失效情况,以评估对象的冗余度.如果冗余度低于能够保证目标可用性的最低冗余度,则系统触发修复并调用修复算法产生新的副本.

2 系统的开销模型

冗余和修复会耗费系统的资源,从而产生冗余数据或修复数据,这种开销称为系统维护开销,目前,维护开销主要以所消耗的系统带宽来衡量[1,6,11].系统的维护开销必须尽可能地少,否则会占用过多的带宽而使系统无法向上层应用提供正常的读写服务.本节首先给出系统的开销模型,用以说明本文优化的指标.

为了分析系统平均情况的维护开销,本文对系统作以下假设:(1) 假设节点间是相互独立的[9];(2) 节点加入和永久离开系统的平均速率是恒定的[6];(3) 系统中节点的数量是稳定的.P2P 系统的测量研究[8,9]表明,这些假设是符合系统真实情况的.

设系统中有 N 个节点,$\alpha(0<\alpha\leq 1)$为节点的平均可用性,节点的到达率和离开率分别为β和λ,并且系统平均节点个数是恒定的($\beta=\lambda$),则节点的平均寿命为 $T=N/\lambda$.设系统总共要存储 D 字节的原始数据量,我们来考虑系统需

要的维护带宽.设 $B(t)$表示系统在时间段$[t,t_0]$的平均维护带宽,t_0 为起始时刻.系统按维护时间 t 分为 3 个阶段:

(1) 数据的初始写入阶段($t_0<t\le t_1$):t_1 是完成时刻,该阶段系统调用冗余算法,估计出目标度 k_L,然后根据选定的冗余方式对原始数据作冗余,最后存入总共 k_LD 字节的数据.

(2) 产生额外冗余阶段($t_1<t\le t_2$):由于误判和漏判,系统需要增加 k_R 的额外冗余来屏蔽暂时失效,使得系统观察到的永久失效率与系统真实的永久失效率相符.例如,TotalRecall[1]预先加入一定量的额外副本,Carbonite[21]通过重用暂时失效的副本来动态加入额外副本,达到屏蔽暂时失效的目的.t_2 是系统完成添加额外冗余的时刻.对于一个理想判断器,由于没有误判和漏判,则系统只需发现一个永久失效就修复一个新副本,因此能够准确地维护 k_L 的目标冗余度,即 k_R=0.现实中的系统都是非理想系统,k_R 通常情况下大于 0(例如,在 Carbonite 系统中,$k_R\approx(2/\alpha-1)k_L$).

(3) 修复永久失效节点阶段($t>t_2$):t_2 时刻以后,系统不断地修复维护 k_R+k_L 的平均冗余度,因此,单个节点上平均存有$(k_R+k_L)D/N$的数据量.由于失效率为系统节点离开率λ,因此,单位时间需要修复$\lambda(k_R+k_L)D/N$的数据量,利用 $T=N/\lambda$,得到系统的平均维护带宽为

$$B(t)=\frac{(k_L+k_R)D+(t-t_0)(k_L+k_R)D/T}{\alpha(t-t_0)} \tag{1}$$

其中,$t>t_2$,$(k_L+k_R)D$ 是系统冗余带来的累积带宽,$(t-t_0)(k_L+k_R)D/T$ 是系统修复带来的累积带宽.$\alpha(t-t_0)$是节点带宽可用的平均时间.当 $t\to\infty$时, $B(t)\to\dfrac{(k_L+k_R)D}{\alpha T}$.

式(1)说明了系统的动态性和系统算法对维护带宽的影响:1) 节点的平均可用性α和平均寿命 T 刻画了系统动态性.现有的测量研究[8−10]表明,典型的 P2P 系统都是高动态的,表现在节点平均可用性较低(例如,Maze 中的节点只有 0.27 的可用性)和寿命较短(例如,Maze[24]中的节点只有平均 60 天的寿命,远低于用户机器[25]或磁盘的寿命[11])两个方面,导致在 P2P 环境中的维护开销 $B(t)$高于传统系统的维护开销,说明了优化 P2P 存储系统的维护带宽的重要性,这也是本文的动机.2) 目标冗余度 k_L 和额外冗余度 k_R 体现了系统算法对维护带宽的影响.因此,k_R 和 k_L 分别是冗余算法和修复算法的优化指标.本文的目标就是提出更适合 P2P 环境的可用性维护算法,来降低 k_L(本文的第 3 节)和 k_R(本文的第 4 节),优化维护带宽.

3 数据冗余优化

数据冗余优化的目标是降低目标冗余度 k_L,即在没有永久失效时,用更少的冗余度来屏蔽暂时失效,保证数据的目标可用性.目前,主流的数据冗余算法是在整个系统里随机选择节点分发数据,然后利用节点在线概率的平均值计算出数据的目标冗余度[1,6,11,12,16].但这种方法并没有考虑系统中不同节点的可用性差异,本节将提出一种基于节点动态性的概率分发算法,证明它能在 P2P 这种高动态环境中优化系统维护带宽.与其他优化算法[17−19]不同,本节提出的优化算法仍能保证系统中节点的负载平衡(即分发后每个节点上存储相等的数据量),而不是通过不断将负载移向高可用节点来减少冗余度.

3.1 节点动态性差异对k_L的影响

设 h 为节点的可用性,$f(h)$为节点可用性的概率密度,其均值为α.在 P2P 系统中,α比较低,现有的 P2P 测量研究[8−10]表明:在不同的 P2P 系统中,节点的平均可用性α都不超过 0.3.由于相关研究[12,14]表明高动态系统中适合采用副本方式,因此本文只考虑冗余方式是副本方式的情况.设目标可用性为 $1-\varepsilon$,首先,计算主流方法所需的平均目标冗余度 k_L.

$$\varepsilon=P(\text{object } o \text{ is unavailability})=(1-\alpha)^{k_L} .$$

得出

$$k_L=\frac{\log\varepsilon}{\log(1-\alpha)} \tag{2}$$

由于α比较小($\alpha\le0.3$),从图 2 可以看出, $-\log(1-\alpha)$ 近似等于α,式(2)可以重写为

alpha 直接变成积分式了?

$$k_L=\frac{\log\varepsilon}{-\alpha}=\frac{\log 1/\varepsilon}{\int_0^1 hf(h)\mathrm{d}h} \tag{3}$$

下面考虑另一种放置算法 P,理想情况下,算法 P 能够使系统完全满足两个约束:(1) 同一对象的副本存放在可用性相同的节点上;(2) 每个节点仍存放相等的数据量(冗余后的数据).根据式(2),副本都存放在可用性为 h 的节点上的数据所需冗余度为 $k(h)=\frac{\log\varepsilon}{\log(1-h)}$.设冗余后的数据量为 S,则算法 P 下存储的原始数据量为

$$D=\int_0^1\frac{Nf(h)\frac{S}{N}}{k(h)}\mathrm{d}h=S\int_0^1\frac{f(h)}{k(h)}\mathrm{d}h \tag{4}$$

其中,$Nf(h)$是可用性为 h 的节点数,S/N 是每个节点上的数据量.由此得到算法 P 下的平均目标冗余度为

$$\tilde{k}_L=\frac{S}{D}=\frac{1}{\int_0^1 f(h)/k(h)\mathrm{d}h}=\frac{\log 1/\varepsilon}{\int_0^1-\log(1-h)f(h)\mathrm{d}h} \tag{5}$$

由式(3)和式(5)得到算法 P 相对主流算法的对目标冗余度的改进为

$$I=1-\frac{\tilde{k}_L}{k_L}=1-\frac{\int_0^1 hf(h)\mathrm{d}h}{\int_0^1-\log(1-h)f(h)\mathrm{d}h} \tag{6}$$

从图 2 可以看出,只有当 h 较小(例如,小于 0.3)时,$-\log(1-h)$ 才近似等于 h,而当 h 较大时,$-\log(1-h)$ 将比 h 大很多,因此,$I<1$.式(6)说明:(1) 由于 P2P 系统平均可用性较低,随机分发算法中使用平均可用性高估了存储在高可用节点上的数据所需的冗余度,而算法 P 能够准确评估每个可用性值下的冗余度,使得系统平均冗余度比主流方法要小.(2) 改进量 I 取决于系统的可用性分布情况 $f(h)$(即节点的可用性差异).例如,图 3 给出了 Maze 系统中节点可用性的分布情况,约 30%节点的可用性超过 0.4.结果接近于 Overnet 网络的测量结果[9].根据式(2)和式(5)分别计算两种算法的目标冗余度可以得出:Maze 系统中算法 P 对目标冗余度的改进 $I\approx0.3$,即在和 Maze 动态性相近的 P2P 系统里,算法 P 相对主流算法在理论上能够节省 30%左右的带宽.

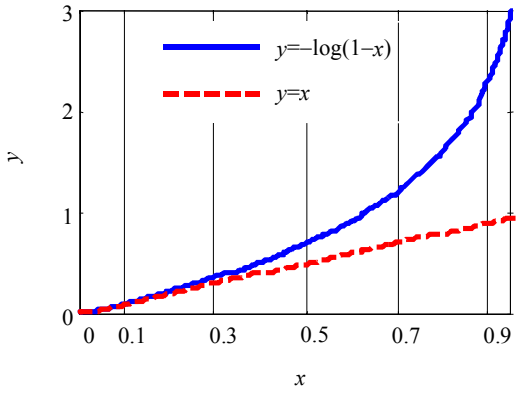


Fig.2 Comparison between $y=-\log(1-x)$ and $y=x$

图 2 函数 $y=-\log(1-x)$和函数 $y=x$ 比较

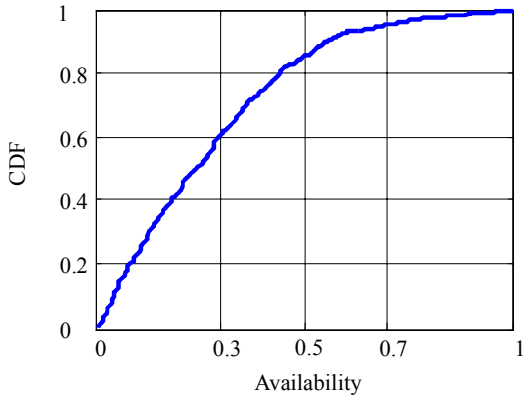


Fig.3 Cumulative distribution of peer availability in the Maze system

图 3 Maze 系统中节点可用性的累积分布图

3.2 基于节点动态性的概率放置算法

本节给出算法 P 在系统中的具体实现.节点的会话时间(time to failure,简称 TTF)和离线时间(time to recovery,简称 TTR)分别是指节点的一次连续在线时间长度和连续离线的时间长度,其均值分别称为 MTTF (mean time to failure)和 MTTR(mean time to recovery).MTTF 和 MTTR 可以通过对节点的不断监控得到,节点的可用性 $h=MTTF/(MTTF+MTTR)$. 算法 P 用以下方式满足第 3.1 节中说明的两个约束条件:(1) 节点聚类,把 MTTF 和 MTTR 相近(即可用性相近)的节点聚为一类,数据对象的每个副本放在同类节点上,以满足第 1 个约束条件.(2) 概率选组,为了满足第 2 个约束条件,每个组以不同的概率来选取放置数据.设 h_i 是第 i 组的可

用性,n_i 是第 i 组节点的个数,k_i 是存放在第 i 组数据所需要的目标冗余度,则选取第 i 组的概率为 $p_i=\frac{n_i/k_i}{\sum_i n_i/k_i}$.

下面给出算法的详细描述:

算法 1. 初始向系统存入数据.

输入:系统中当前的节点 S,以及每个节点的 MTTF 和 MTTR.

(1) 在 S'中随机选择 n 个节点作为分类的训练样本.

(2) 定义任意两个节点 x_i,x_j 之间的距离

$$D_{ij}=\|x_i-x_j\|^2=(MTTF_i-MTTF_j)^2+(MTTR_i-MTTR_j)^2$$

(3) 选择分组个数 k,利用 K-means 算法把训练样本分为 k 个类

(4) **for** S 中的每个非样本节点

(5) 按照到样本中心的距离归并到一个类中.

(6) **for** 每个数据对象 o,系统

(7) 以 p_i 的概率分配类 i

(8) 将对象 o 冗余到 k_i 的冗余度

(9) 在类 i 中随机选择节点放置副本.

这个“冗余”是个什么操作，按照你的说法不应该就是下面的“放置副本”？

放置完之后如何保证所有节点负载均衡，也就是都保存同样的数据量？

至此,本文已给出一种新的数据冗余算法,算法中考虑了 P2P 系统中节点的动态性差异,准确地评估了不同节点可用性下所需的冗余度,降低了系统冗余带来的维护开销.

4 数据修复优化

数据修复优化的目标是降低额外冗余度 k_R,即在存在节点永久失效的情况下,降低由于系统误判和漏判而带来的额外开销.由于存在 P2P 存储系统中时间阈值的难题,目前大多数系统并不是用时间阈值来区分暂时和永久失效,而是用它来检测节点失效(即观察节点是否在线),因此,系统中时间阈值一般很小(例如,τ=1hr)[1,6,11]. 系统通过增加添加额外冗余的方式来屏蔽暂时失效,但也同时增加了维护额外冗余的开销.文献[21]通过计算 Chernoff bound 得出:为了保证系统观察到至少 k_L 的在线的冗余度(不触发修复的条件),系统需要总共维护大约 $2k_L/\alpha$的冗余度,其中α 是节点的平均可用性,即在不区分失效的情况下,系统的额外冗余度为

$$k_R\approx\left(\frac{2}{\alpha}-1\right)k_L \tag{7}$$

由式(7)可以看出,$k_R\propto 1/\alpha$,因此,添加额外冗余的方式对α较高的环境(例如在 PlanetLab 中,α>0.7)较为有效,而对α较低的 P2P 环境,将导致额外冗余度 k_R 将远高于目标冗余度 k_L,会使系统维护占用掉所有可用的系统带宽,因此,这种环境下,区分暂时失效和永久失效是必须的.本节首先介绍两种判别器.时间阈值判别器和概率判别器,并给出基于判别器的数据修复算法;然后讨论判别器产生的误判和漏判对 k_R 的影响,给出判别器参数选取的统一原则;最后给出如何在实际系统中获取判别器的参数.

4.1 基于判别器的数据修复算法

失效判别器可以看作是一个函数: $f(node,t)\rightarrow\{A,U,D\}$.任意时刻 t,判别器把每个节点 $node$ 映射到以下 3 个状态:可用(available),暂时失效(unavailable),永久失效(dead);其中,状态集$\{A,U\}$可以合并成一个状态 L(Live). 本文将给出两类永久判别器:时间阈值判别器和概率判别器.本文用 $Timeout(\tau)$ 表示时间阈值判别器,其中τ是时间阈值. $Timeout(\tau)$ 用阈值τ区分暂时和永久失效,属于确定型永久判别器.设 $d(t)$表示节点 $node$ 在 t 时刻的离线时间,则 $Timeout(\tau)$ 的映射关系 $f(node,t)\rightarrow\{A,U,D\}$ 为

$$f(node,t)=\begin{cases}A, & \text{if } d(t)=0\\ U, & \text{if } 0<d(t)<\tau\\ D, & \text{if } d(t)\geq\tau\end{cases} \tag{8}$$

概率判别器是本文提出的一类新的永久判别器,以下用 $Probability(f(\xi))$ 来表示. $f(\xi)$ 是一个以节点离线时间ξ为自变量的概率函数.当发现一个节点已经离线ξ,判别器以 $f(\xi)$的概率判断节点永久离开.设 Z 是 $Probability(f(\xi))$ 判断节点所处的状态,其映射关系 $f(node,t)\rightarrow\{A,U,D\}$ 为

$$f(node,t)=\begin{cases}Z=A, & \text{if } d(t)=0\\ P(Z=U)=1-f(d(t)), & \text{if } d(t)>0\\ P(Z=D)=f(d(t)), & \text{if } d(t)>0\end{cases} \tag{9}$$

阈值τ和概率函数 $f(\xi)$分别是两类判别器的待设参数,本文将在下面的两个小节中讨论如何设置参数的问题,本节先给出基于永久判别器的数据修复算法,其中判别器根据每组自身的动态性设置不同的参数:

算法 2. 基于永久判别器数据的修复.

输入:类 i 的 MTTF 和 MTTR,节点平均寿命.

(1) 依据节点动态性设置判别器参数.

(2) 判断类 i 内每个对象 o 当前仍存活的副本数 k':

for 存于 i 类的每个对象 o

$k'\leftarrow 0$;

for 存储对象 o 副本的每个节点 i

if $f(i,t)\rightarrow L$ 这个L是个神马玩意儿？

$k'\leftarrow k'+1$;

(3) 触发修复:

for 存于 i 类的每个对象 o

if $k'<k_L$

修复 k_L-k'个副本;

(4) 隔一时间周期 l,重新执行第(2)步;

下面我们讨论如何设置判别器的参数,第 4.2 节给出判别器参数选取的统一原则,第 4.3 节给出如何依据节点动态性求解判别器参数.

4.2 误判和漏判对k_R的影响

理想的判别器没有误判(把暂时失效判为永久失效)和漏判(把永久失效判为暂时失效).但是,真实系统中的判别器都是非理想的:判别器会由于不准确造成误判,或者由于判断延迟造成漏判;并且误判率和漏判率是此消彼涨的关系,因此,判别器参数选取要在漏判率和误判率之间加以折衷.

首先分析判别器的误判和漏判对单个副本存活情况的影响.设二维状态向量 $V=\langle S_1,S_2\rangle$,其中 $S_i\in\{D,L\}$,S_1 表示存放副本节点的真实状态,S_2 表示判别器所判断的节点的状态.系统对副本 i 的维护可以看作一个过程:任意时刻 t,副本 i 在系统中处于以下几个状态:(1) $V(t)=\langle L,L\rangle$:判别器判断准确,不影响对象冗余度.(2) $V(t)=\langle L,D\rangle$:判别器发生误判,由于原来副本并未丢失,系统此时触发修复,将增加对象的额外冗余度.(3) $V(t)=\langle D,L\rangle$:判别器发生漏判,由于不能及时发现丢失的副本,对象的冗余度会有所降低.(4) $V(t)=\langle D,D\rangle$:判别器判断正确,此时系统选取节点更新存储副本 i 的永久失效的节点.

当选取的节点与被更新的节点具有相同的动态行为(即 TTR,TTF 和 Lifetime 都服从相同分布)时,副本 i 的维护过程可以近似成为一个再生过程,节点更新的时刻是更新点,副本 i 在节点更新前处于$\{\langle L,L\rangle,\langle L,D\rangle,\langle D,L\rangle\}$3个状态之一,如图 4 所示.设 X 表示节点的寿命,其分布为 $F(X)$;Y 表示判别器检测到节点永久失效所需要的延迟,其分布为 $F(Y)$.设$\{U_n\}$是相邻两次更新的时间间隔长度,则$\{U_n\}$服从均值为 $E[X]+E[Y]$的分布 F.根据是文献[26].

判别器对每个副本误判的平均概率 p_{FP} 为一个再生循环内系统处于$\langle L,D\rangle$状态的平均时间比例:

$$p_{FP}=\lim_{t\rightarrow\infty}P\{V(t)=\langle L,D\rangle\}=\frac{E[\text{amount of time in state }\langle L,D\rangle\text{ in a cycle}]}{E[\text{length of cycle}]} \tag{10}$$

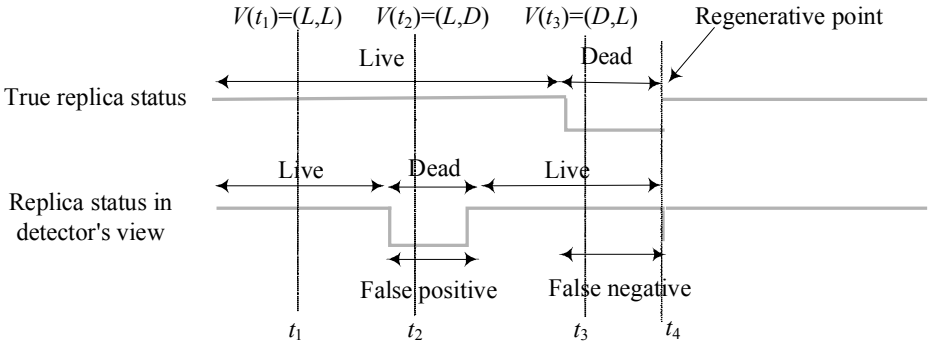


Fig.4 Maintenance process for a single replica based on detector

图 4 系统基于判别器对单个副本的维护过程

漏判的平均概率 p_{NP} 为一个再生循环内系统处于$\langle D,L\rangle$的平均时间比例:

$$p_{NP}=\lim_{t\to\infty}P\{V(t)=\langle D,L\rangle\}=\frac{E[\text{amount of time in state }\langle D,L\rangle\text{ in a cycle}]}{E[\text{length of cycle}]} \tag{11}$$

下面分析平均情况下误判和漏判对整个对象冗余度的影响.误判增加对象冗余度,漏判减少对象冗余度,当判断正确时,系统的冗余度不受影响.假设系统采用副本的冗余方式,一个数据对象 o 当前有 r 个没有永久失效的副本,初始时 $r=k_L$.当系统观察到的冗余度低于目标冗余度时,系统触发修复并将冗余度修复到目标冗余度.由于每个副本下一时刻发生误判的概率为 p_{NP},因此,对象将增加的平均额外副本个数 e 为

$$e(k_L,p_{FP},r)=\sum_{i=0}^{k_L-1}(k_L-i)C_r^i(1-p_{FP})^i p_{FP}^{r-i} \tag{12}$$

其中,$C_r^i(1-p_{FP})^i p_{FP}^{r-i}$ 是系统误判 r 中 $r-i$ 个副本的概率.特殊地,当 $r=k_L$,$e(k_L,p_{FP},r)=rp_{FP}$.另一方面,每个副本下一时刻发生漏判的概率为 p_{NP},因此,对象将丢失的平均副本个数 l 为

$$l(p_{NP},r)=\sum_{i=0}^{r}iC_r^i p_{NP}^i(1-p_{NP})^{r-i}=rp_{NP} \tag{13}$$

由式(12)和式(13)可以看出,增加的额外副本个数 $e(k_L,p_{FP},r)$随着未永久失效的副本个数 r 的增加而减少,而丢失的副本个数 $l(p_{NP},r)$随着 r 的增加而增加,对象的冗余度 r 将达到一个平衡值 r_0,r_0 满足 $e(k_L,p_{FP},r_0)=l(p_{NP},r_0)$:当 $r>r_0$,$e(k_L,p_{FP},r)<l(p_{NP},r)$时,$r$ 开始减少,趋向于 r_0,反之,当 $r<r_0$,$e(k_L,p_{FP},r)>l(p_{NP},r)$时,$r$ 开始增加,趋向于 r_0.特殊地,当 $r=r_0$,$e(k_L,p_{FP},r)=l(p_{NP},r)$时,对象的增加的冗余度抵消掉丢失的冗余度,对象的冗余度保持在 r_0.

最后分析误判和漏判对额外冗余度 k_R 的影响.初始时 $r=k_R$,考虑以下 3 种情况:

(1) 若 $p_{FP}>p_{NP}$,则 $e(k_L,p_{FP},k_L)=k_Lp_{FP}>k_Lp_{NP}=l(p_{NP},k_L)$,因此对象的冗余度 r 开始增加,直到 $r=k_L+\varepsilon'$ $(\varepsilon'>0)$ 达到平衡值 r_0,由于误判大于漏判导致对象有 $k_R=\varepsilon'$ 的额外冗余度.特殊地,对于目前不区分暂时失效和永久失效的系统,实际是 p_{FP} 最大而 $p_{NP}=0$ 的特例.此时,平衡值 $r_0=k_L+\varepsilon'$ $(\varepsilon'>0)$ 满足 $e(k_L,p_{FP},r_0)=0$,所以ε'就是屏蔽所有误判的副本数.

(2) 若 $p_{FP}=p_{NP}$, $e(k_L,p_{FP},k_L)=l(p_{NP},k_L)$,对象的冗余度 r 维持在平衡值 $r_0=k_L$,额外冗余度 $k_R=0$.此时,尽管系统由于误判和漏判无法准确得到对象的冗余度 r,系统仍能保证以下重要性质:当对象的冗余度 r 低于目标冗余度 k_L 时,对象的冗余度 r 就开始增加并趋向于 k_L,反之,当对象的冗余度 r 高于目标冗余度 k_L 时,对象的冗余度 r 开始减少并趋向于 k_L.

(3) 若 $p_{FP}<p_{NP}$,则 $e(k_L,p_{FP},k_L)<l(p_{NP},k_L)$,因此,对象的冗余度 r 开始减少,直到 $r=k_L-\varepsilon'$ $(\varepsilon'>0)$ 达到平衡值 r_0.此时,系统不能满足维护的冗余度 r 不低于目标冗余度的约束.

根据上面的分析,得出下面重要结论:永久判别器参数的选择应该使判别器产生的平均误判概率 p_{FP} 等于平均漏判概率 p_{NP},即

$$p_{FP}=p_{NP} \tag{14}$$

此时,判别器的误判和漏判达到平衡,系统在满足 $r_0\geq k_L$ 的约束条件下使额外冗余度 k_R 最小.

4.3 两类永久判别器中参数的设置

根据上节分析,判别器的设置应该满足式(12).同时,由于不同类别的动态性不同(不同类有不同的 MTTF 和 MTTR),因此,每类内判别器的参数也应该不同.

为了选取判别器参数,需要分析单个节点的动态行为.测量研究[8]表明,P2P 系统中单个节点的在线时间和离线时间服从指数分布.由于每类内的节点 MTTF 和 MTTR 相近,节点的在线和离线时间近似服从均值相同的指数分布.与其他 P2P 存储研究[1,11,14,21]相同,本文假设节点的寿命服从指数分布.由于指数分布的无记忆性,节点的更新可以看作是再生过程,p_{FP} 和 p_{NP} 利用式(10)和式(11)计算.设λ_i 和μ_i 分别表示第 i 类节点的在线均值 MTTF 和离线均值 MTTR,T 表示节点的平均寿命.图 5 给出节点动态行为的 Markov 模型.模型中系统有 3 个状态$\{A,U,D\}$,q_{ij} 是节点从状态 i 到状态 j 的转移率,设 k 表示节点在进入吸收态 D 之前访问暂时失效态 U 的次数(即节点下线次数),p_{13} 是节点从状态 A 转移到状态 D 的概率,$p_{13}=q_{13}/(q_{13}+q_{12})$,则 k~Geometric(p_{13}).解下面方程:

$$\begin{cases}(q_{12}+q_{13})=1/\lambda_i\\ q_{21}=1/\mu_i\\ q_{13}/q_{12}=E[k]\\ \lambda_i+E[k](\lambda_i+\mu_i)=T\end{cases},$$

得到各状态间的转移率为

$$\begin{cases}q_{12}=1/\lambda_i-1/(\alpha_i T)\\ q_{13}=1/(\alpha_i T)\\ q_{21}=1/\mu_i\end{cases},\quad \text{其中},\alpha_i=\lambda_i/(\lambda_i+\mu_i) \tag{15}$$

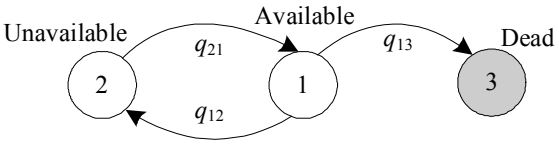


Fig.5 Markov mode of node dynamical behavior

图 5 节点动态行为的 Markov 模型

对于 *Timeout*(τ),若节点未永久失效但离线时间 $\xi\geq\tau$,则判别器将发生误判;若节点永久失效但离线时间 $\xi<\tau$,则发生漏判,根据式(10)、式(11)和式(14),参数τ可以通过下面方程解出:

$$E[k]\int_{\tau}^{\infty}(\xi-\tau)\frac{1}{\lambda_i}\mathrm{e}^{-\xi/\lambda_i}\mathrm{d}\xi=\tau \tag{16}$$

其中,$E[k]=q_{13}/q_{12}$,式子左边是一个再生循环内判别器处于误判状态的平均时间,右边是处于漏判状态的平均时间.系统无法得到τ 的解析解,但可以通过二分查找的方法得出近似解.

对于 *Probability*($f(\xi)$),假设节点离线时间是ξ,若节点未永久失效,则判别器发生误判的概率为 $f(\xi)$;若节点永久失效,则发生漏判的概率为 1–$f(\xi)$,则根据式(10)、式(11)和式(14),参数$f(\xi)$可以通过下面方程解出:

$$E[k]\int_{0}^{\infty}f(\xi)\mathrm{e}^{-\xi/\lambda_i}\mathrm{d}\xi=\int_{0}^{\infty}1-f(\xi)\mathrm{d}\xi \tag{17}$$

式子左边是一个再生循环内处于误判状态的平均时间,右边是处于漏判状态的平均时间,解得$f(\xi)$的解析解:

$$f(\xi)=\frac{1}{1+E[k]\mathrm{e}^{-\xi/\lambda_i}} \tag{18}$$

因此,当系统通过监控获得节点的 MTTF,MTTR 和寿命后,就能分别依据式(16)和式(18)设置两类判别器的参数.

下面分析 *Probability*($f(\xi)$) 相对于 *Timeout*(τ) 的不同.设 $p(D\mid\xi)$ 表示节点在已经离线ξ时处于永久失效状态 D 的概率,根据贝叶斯定理得出,

$$p(D\mid\xi)=\frac{p_{13}}{p_{13}+(1-p_{13})\mathrm{e}^{-\xi/\lambda_i}} \tag{19}$$

把 $p_{13}=q_{13}/(q_{13}+q_{12})$和 $E[k]=q_{13}/q_{12}$ 代入上式,得出:

$$p(D|\xi)=\frac{1}{1+E[k]e^{-\xi/\lambda_i}} \tag{20}$$

式(18)和式(20)表明,当观察到一个节点已经离线ξ,满足式(12)的概率判别器 $Probability(f(\xi))$ 判断节点永久失效的概率 $f(\xi)$等于节点真实永久失效的概率 $p(D|\xi)$.这使得相对于同样满足式(14)的时间阈值判别器 $Timeout(\tau)$ 而言,概率判别器 $Probability(f(\xi))$ 还具有以下性质:

(1) 对于单个节点的任意离线长度,判别器发生误判和漏判概率相同.假设节点当前离线时间为ξ,则判别器发生误判的概率为 Pr{节点未永久离开,判别器判断离开},表达式为

$$(1-P(D|\xi))\times f(\xi) \tag{21}$$

而发生漏判的概率为 Pr{节点永久离开,判别器判断未离开},表达式为

$$p(D|\xi)\times(1-f(\xi)) \tag{22}$$

由于 $f(\xi)=p(D|\xi)$,因此,对于任意离线长度ξ,误判和漏判概率相同,都为 $f(\xi)\times(1-f(\xi))$.

(2) 对于单个节点,由于误判和漏判的表达式都为 $f(\xi)\times(1-f(\xi))$,因此误判和漏判的概率都不超过 0.25.

5 实验分析

为了分析上述优化策略的效率,我们采用实际 P2P 系统的节点上下线日志,模拟在 P2P 系统中维护数据可用性的过程.本节通过对累积带宽使用量、系统级可用性和目标冗余度的达到率等指标的测量,表明本文提出的优化策略在维护目标冗余度的同时,能够极大地减少系统的维护开销.

5.1 数据集和模拟器

本文模拟实验的数据来自于 Maze 系统[24].Maze 是中国教育网 CERNET 上规模最大的 P2P 系统之一,平均任意时刻有 2 万用户同时在线.Maze 系统中用户的带宽是受约束的,同时,节点的可用性也不高,平均只有 0.27.本节采用 2005 年 3 月 1 日~2005 年 5 月 1 日的系统运行日志来构造实验环境.

系统的运行需要知道每个节点的 MTTF 和 MTTR 以及节点的平均寿命.为了测出节点的 MTTF 和 MTTR,节点最初的 x 个 TTF 和 TTR 将当作样本值,系统只有当节点离线 x 次以后才使用它,本节实验中取 x=4.这种做法可以避免使用系统中极为动态的节点,测量研究[8]表明,P2P 系统中很多节点上线了 1~2 次就永久退出系统,如果使用这样的节点会极大地浪费系统的带宽.同时文献[8]还表明,Maze 中很少有节点的暂时离线时间超过 15 天,因此,在本节的实验中,用连续 15 天离线作为节点永久离开的标准.通过测量发现,系统中节点在 3 月份平均离开率为λ=0.017N/天,其中 N 为平均每天存活的节点个数,因此,节点平均寿命 $T=N/\lambda\approx$60 天.

模拟器由 Maze 中节点的上下线日志驱动:如果一个节点在日志中在线,那么在模拟系统中也在线;若节点在日志中失效,在模拟系统也失效,反之亦然.模拟器中使用的参数如下:(1) 系统在使用第 3.2 节冗余算法时随机选择 1 000 个节点作为样本节点进行分类,即在第 3.2 节算法中取 n=1000,其中节点总数$|S'|$=10000;(2) 系统在使用第 4.1 节的修复算法时假设节点的寿命服从指数分布,并依据式(16)和式(18)计算判别器参数.系统用 3 月份测得的节点的 MTTF,MTTR 和平均寿命作为永久判别器的输入参数,然后系统观察在 4 月份维护数据的情况.(3) 系统每隔 l=1 小时判断一次剩余的副本数,不足目标冗余度则进行修复.

系统运行过程如下:我们在 4 月 1 日(每小时存入 500 个数据对象)向系统存入 12 000 个数据对象,系统分别按照目前主流的方法和本文提出的方法来维护数据在 4 月份数据的可用性,系统在运行过程中记录带宽使用量和对象在每一时刻存活的冗余度等指标.

5.2 实验结果

5.2.1 对额外冗余度 k_R 的优化结果

由于时间阈值配置的难题,目前主流的方法是不区分失效,采用添加额外副本的方式来屏蔽暂时失效,而这种方法只适合动态性较低的系统环境.本文基于误判抵消漏判的思想,提出了合理配置永久判别器的原则和方法.本节的实验通过对比基于额外副本方法和基于判别器方法下系统的维护带宽,表明本文提出的算法对额外冗余度 k_R 和系统带宽的减少量.由于有研究[21]表明,在不区分失效的前提下,动态添加额外副本的方式优于静态

添加的方法,因此,实验中用文献[21]提出的 Carbonite 算法代表基于额外副本方法的最优效果.同时,实验中为了避免对 k_L 的影响,仍采用主流的冗余算法,没有对节点按动态性分类.实验中取目标可用性为 0.9,根据式(2)得到目标冗余度为 k_L=8.

图 6 给出了对额外冗余度 k_R 的减少量.从图 6 可以看出,在 Carbonite、基于 *Timeout*(τ) 和基于 *Probability*($f(\xi)$) 这 3 种方法下平均每个对象的未永久失效的副本个数(k_L+k_R)随时间的变化趋势.图 6 表明,每个对象未永久失效副本个数的变化与本文第 2 节提出的模型相符,分为 3 个阶段:

(1) 初始写入阶段(图 6 中最初 0~24 小时的直线段):由于采用相同的冗余算法,因此在这一阶段,3 种方法下平均每个对象的未永久失效副本数相同,都为 8 个副本.

(2) 产生额外冗余阶段(图 6 中最初 24~360 小时单调递增的曲线段):该阶段 3 种方法下对象的副本个数均有增加,但增加程度不同.Carbonite 由于不区分暂时失效和永久失效,需要动态产生大量的额外副本来屏蔽暂时失效.如图 6 所示,在 k_L=8 的情况下,Carbonite 将最终平均为每个对象维护 30 个未永久失效副本(22 个额外副本),远远高于实际所需要的冗余度 k_L.而基于 *Timeout*(τ) 和基于 *Probability*($f(\xi)$) 的方法只产生很少的额外副本,在 k_L=8 的情况下,两种方法将最终平均为每个对象维护大约 10 个未永久失效副本(2 个额外副本).基于判别器方法产生额外副本的原因主要是由于系统中所有节点的 TTR 的分布 F 并不严格服从指数分布.研究[8]表明,F 具有长尾性质,它比具有相同均值的指数有更大比例的长 TTR,因此我们在用式(16)、式(18)配置判别器时,并没有使误判抵消掉漏判,而是使误判的概率稍高于漏判的概率,导致系统产生额外的冗余.这个问题可以通过节点的动态性聚类来进一步解决.由于文献[8]表明,具有相同 MTTR 的节点的 TTR 分布服从指数,因此动态性聚类可以避免节点 TTR 的长尾性质导致的额外冗余.

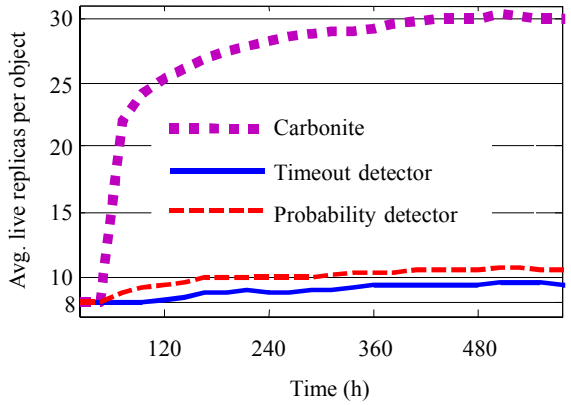


Fig.6 Average No. of live replicas per object (k_L=8)

图 6 平均每个对象的副本个数的变化趋势(k_L=8)

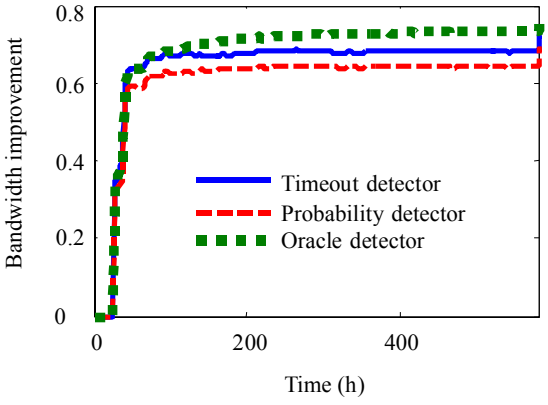


Fig.7 Bandwidth improvement with different detector (k_L=8)

图 7 使用不同判别器时带宽减少量的变化趋势(k_L=8)

(3) 修复永久失效节点阶段(图 6 中大于 360 小时的直线段):由于该阶段对象的副本个数达到每种方法下平衡值 r_0,系统修复率和失效率平均一致,对象在每种方法下平均副本个数不变.

图 7 给出了基于两类判别器算法相对于 Carbonite 算法的带宽减少量.对于给定时刻 t,设 Carbonite 在[t_0,t]时间内的平均带宽为 $B(t)$,使用判别器(*Timeout*(τ) , *Probability*($f(\xi)$))的算法在[t_0,t]内的平均带宽为 B_D(t),则改进量 $I_D(t)$为

$$I_D(t)=\frac{B(t)-B_D(t)}{B(t)}=1-\frac{B_D(t)}{B(t)} \tag{23}$$

为了标定两类判别器的效果,图中还给出基于理想判别器(oracle detector)算法相对于 Carbonite 的减少量.理想判别器没有漏判和误判,系统只针对永久失效来触发修复,因此,基于理想判别器算法对应于系统对带宽减少量的上界.与图 6 相对应,$I(t)$的变化随不同算法下 k_R 的变化分为 3 个阶段.$I_D(t)$随着 k_R 的增加而增加,并在对象的副本数达到每种方法下的平衡值时达到稳定值.从图中可以看出以下几点:

(1) 基于判别器的方法能够极大地减少系统的维护带宽.图 7 表明,当 $I(t)$达到稳定值时,基于判别器的方法

能够相对于 Carbonite 减少大约 70%的带宽.这是由于基于判别器的方法能够极大地降低额外冗余度 k_R.设 $I(t)$ 达到稳定时,Carbonite 产生的额外冗余度为 k_R,使用判别器的算法产生的额外冗余度为 k_R^D.取 $t\to\infty$,根据式(1)、式(23)得出, $I_D(t)\to 1-(k_L+k_R^D)/(k_L+k_R)$.由图 6 可知,在 *Timeout*($\tau$) 和 *Probability*($f(\xi)$) 两类判别器中, $k_L+k_R^D\approx 10, k_L+k_R\approx 30$,代入得 $I_D(t)\to 0.68$;理想判别器中 $k_R^D=0$,减少量最大,代入得 $I_D(t)\to 0.73$;结果都与图 7 相符.

(2) *Timeout*(τ) 和 *Probability*($f(\xi)$) 的性能非常接近最优的理想判别器,它们只比理想判别器低不足 0.1 的减少量.通过上面对图 6 的分析可知,当节点按动态性聚类后,两类判别器的效果能够进一步接近理想判别器.

图 6 中给出的是平均每个对象的副本数,由于判别器存在误判和漏判,对象的副本数会存在方差.图 8 给出在系统维护数据一个月后,对象最终的副本个数的分布图.从图 8 可以看出,在 Carbonite 算法下,每个对象的副本个数分布在 25~50 这一区间内,远远高于对象的 k_L=8 个副本的目标.而在 *Timeout*(τ) 和 *Probability*($f(\xi)$) 方法下,分别有大约 99%和 99.9%的对象的副本个数分布在 8~12 的区间内,等于或略高于目标.因此,它们使系统用很小的额外开销维护了每个对象的目标冗余度.同时,图 8 也表明,在任一时刻,由于一些漏判不能及时被误判抵消掉,因此,极少部分对象的副本个数可能会低于目标.例如,在 *Timeout*(τ) 和 *Probability*($f(\xi)$) 方法下分别有大约 1%和 0.1%的对象的副本个数未达到 8 个副本的目标.因此,我们研究系统的另一项指标:目标达到率.

目标达到率 p 是指系统能够保证每个对象当前的副本个数达到目标的概率.在任一时刻 t,设 r 表示对象当前副本个数,目标为 k_L,则 $p=\Pr\{r\geq k_L\}$.设系统有 N_o 个对象,则达到目标的对象个数 n_o 可以看作是伯努利分布 $B(N_o,p)$,因此,p 可以通过 $p=n_o/N_o$ 计算得出.对于系统来说,一方面,r 低于 k_L 主要是由于判别器的误判,误判发生的概率越小,p 越高;另一方面,由于误判和漏判是此消彼长的关系,减少漏判必然增大误判,导致系统的额外冗余增加.例如,Carbonite 算法下,系统漏判率为 0,此时 p=1;但图 6 已表明 Carbonite 极大地增加了系统的开销,事实上,此时系统的误判率最高,额外开销也最大,因此系统必须在二者之间加以折衷.

本文的目标就是给出合理的折衷方法.图 6 和图 7 表明 *Timeout*(τ) 和 *Probability*($f(\xi)$) 能够将系统的开销降低到接近最优值.图 9 给出了 *Timeout*(τ) 和 *Probability*($f(\xi)$) 方法下系统的目标达到率随时间变化的趋势.从图 9 中可以看出,两类判别器都能达到很高的目标达到率,例如, *Timeout*(τ) 方法下,当系统稳定时,p 能够维持在 0.98 左右, *Probability*($f(\xi)$) 方法下,p 始终保持在 0.999 左右.因此,两类判别器在保证了高目标达到率的基础上使开销降低到接近最优值,很好地解决了开销和目标达到率之间的折衷问题.

图 9 还表明了 *Timeout*(τ) 和 *Probability*($f(\xi)$) 方法的区别. *Timeout*(τ) 方法下,p 在 120 小时附近达到最低值 0.94,120 小时以后逐渐增加,最终维持在 0.98 左右.这是由于判别器识别一个误判需要 τ 的延迟,使一些漏判未能被及时抵消,导致 p 最初下降.而随着时间的增加,被识别的误判开始抵消漏判,p 开始增加.而 *Probability*($f(\xi)$) 方法下,p 保持在 0.999 左右,始终优于 *Timeout*(τ) 方法.这是由于 *Probability*($f(\xi)$) 方法的特殊性质:任何离线长度 ξ 下,误判的概率都等于漏判的概率,并且都不超过 0.25.因此, *Probability*($f(\xi)$) 方法下漏判能够被及时抵消,目标达到率高于 *Timeout*(τ) 方法.

5.2.2 对目标冗余度 k_L 的优化结果

节点分类对系统的性能优化具有重要作用:首先,本文第 3.1 节的分析表明,把节点按照动态性分类可以降低目标冗余度 k_L,从而进一步降低系统的维护带宽.其次,由于同类中的节点具有相似的上下线行为,即近似服从相同均值的指数分布,因此,在同一类节点内分发数据并设置判别器参数可以提高判别器精度.设类的个数为 k,k 越大,系统对节点的分类越精确,每个类内节点的动态性也越相近;但是每个类内的节点个数也越少.系统需要在二者之间加以折衷,本节实验中取 k=3 和 k=7.如图 10~图 13 所示.

5.2.3 对维护带宽整体优化结果

本节给出使用本文的优化算法后(包括使用分类和判别器),相对于目前主流算法(以 Carbonite 为代表)对系统维护带宽总的减少量 $I(t)$.根据前两节的分析,$I(t)$可以表达为

$$I(t)=1-\frac{B_D^C(t)}{B(t)}=1-\frac{B_D^C(t)}{B_D(t)}\bullet\frac{B_D(t)}{B(t)}=1-(1-I_C(t))(1-I_D(t)) \tag{25}$$

当 $t\to\infty$时,根据式(1),得出 $I(t)\to 1-(k_L^C+k_R^D)/(k_L+k_R)$,即当系统的每个对象副本数达到平衡值后,优化算法对系统的减少量取决于对目标冗余度 k_L 和额外冗余度 k_R 的减少量.图 14 和图 15 给出同时使用分类法和判别器后对系统整体带宽的减少量.从图中可以看出,减少量随时间变化分为 3 个非递减阶段,当系统达到稳态后,本文提出的算法相对于主流算法可以降低接近 80%的维护带宽.

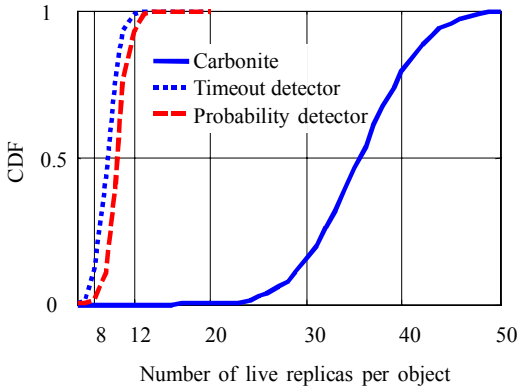


Fig.8 CDF of the number of live replicas per object (k_L=8)

图 8 对象的副本数的累积分布图(k_L=8)

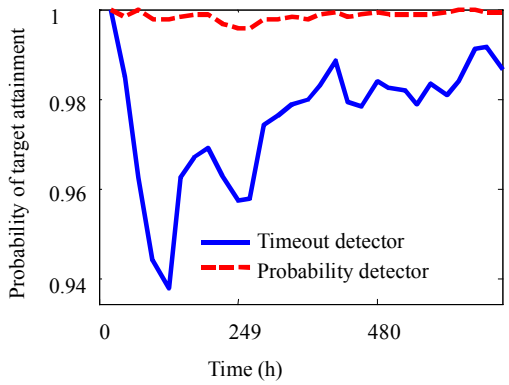


Fig.9 Probability of target attainment (k_L=8)

图 9 目标冗余度的达到率的变化趋势(k_L=8)

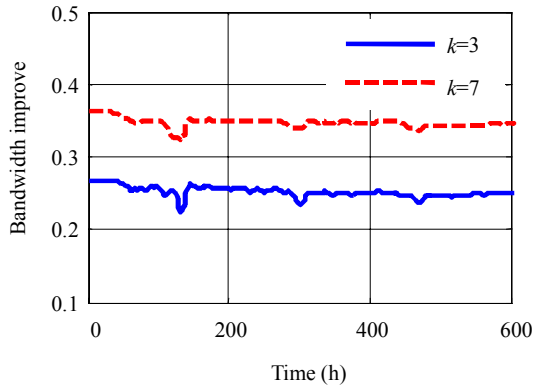


Fig.10 Bandwidth improvement after being clustered given *Timeout*(τ) is used

图 10 *Timeout*(τ) 下分类后带宽减少量的变化趋势

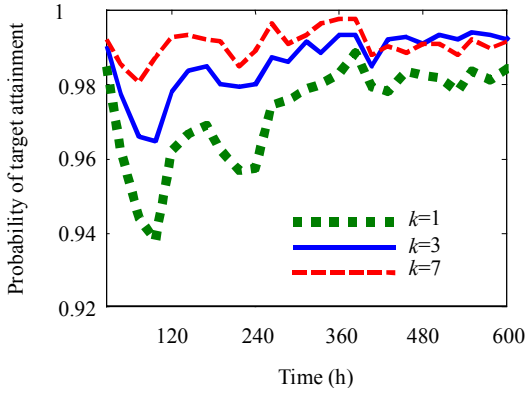


Fig.11 Probability of target attainment after being clustered given *Timeout*(τ) is used

图 11 *Timeout*(τ) 下分类后目标冗余度达到率的变化趋势

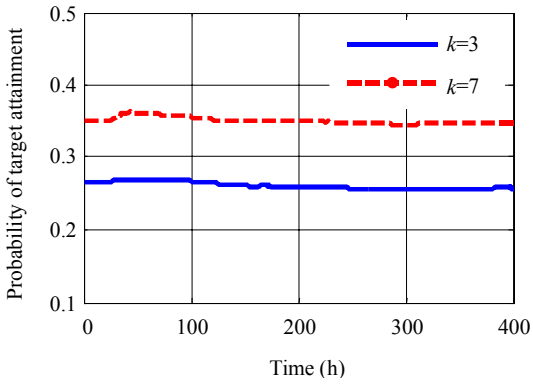


Fig.12 Bandwidth improvement after being clustered given *Probability*($f(\xi)$) is used

图 12 *Probability*($f(\xi)$) 下分类后带宽减少量的变化

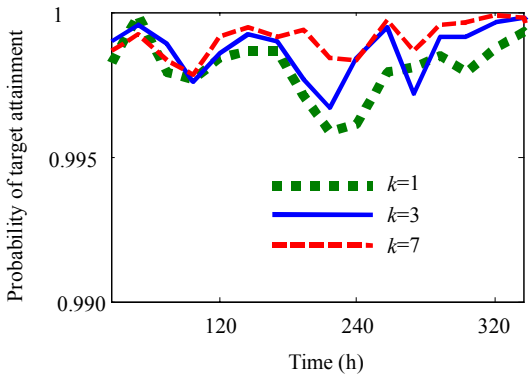


Fig.13 Probability of target attainment after being clustered given *Probability*($f(\xi)$) is used

图 13 *Probability*($f(\xi)$) 下分类后目标冗余度达到率的变化

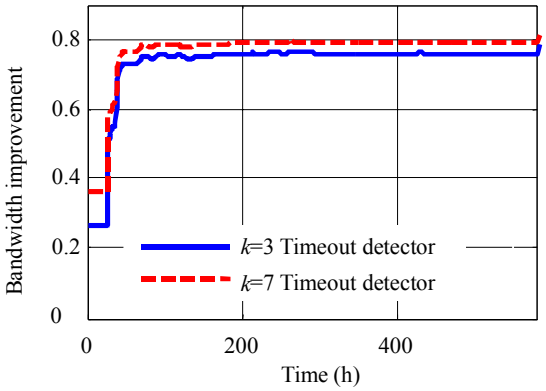


Fig.14 Overall bandwidth improvement comparing with Carbonite after using clustering and *Timeout*(τ)

图 14 使用聚类和 *Timeout*(τ) 后,相对 Carbonites 所减少的带宽总量的变化趋势

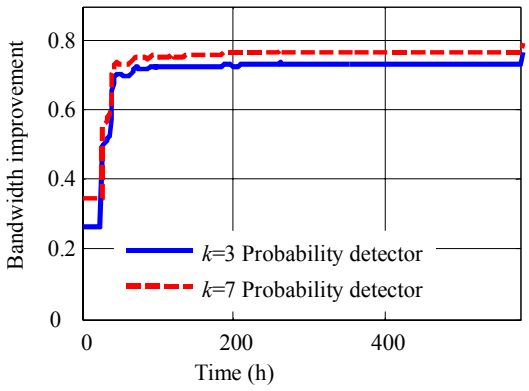


Fig.15 Overall bandwidth improvement comparing with Carbonite after using clustering and *Probability*($f(\xi)$)

图 15 使用聚类和 *Probability*($f(\xi)$) 后,相对 Carbonites 所减少的带宽总量的变化趋势

6 总 结

本文的目标是提出一种在 P2P 存储系统中高效维护数据的方案.本文首先分析了现有方法的不足,然后给出了 P2P 存储系统中数据维护的开销模型.基于模型,本文指出对维护开销的降低关键是降低两个参数:目标冗余度 k_L 和额外冗余度 k_R.随后,本文提出一种基于节点动态性的数据分发算法来降低目标冗余度 k_L,在 Maze 中的实验结果表明,该算法比目前已有的算法能减少 30%左右的带宽.然后,本文给出了基于永久失效判别器的数据修复算法来降低额外冗余度 k_R,实验结果表明,该算法比目前算法能够减少 70%左右的带宽.另外,本文还给出一种新的基于概率的失效判别器,它比已有的基于时间阈值的判别器更为准确.最后的实验结果表明,本文的方案比目前已有的方案总共可以减少 80%左右的带宽.

本方案除了能够减少系统维护带宽以外,还可以用于提供差异型的存储服务.由于节点依据动态性聚类后,节点只能把数据存放在和它动态性相近的节点上,这就使得系统可以为不同动态性节点提供有差异的服务.例如,假设系统中节点贡献的存储空间均为 S 并且数据的目标可用性相同,设任意一个节点 i 的可用性为 h_i,目标冗余度为 $k_L(h_i)$,对于两类节点 i,j,若 $h_i<h_j$,根据式(1)可得 $k_L(h_i)>k_L(h_j)$,因此,两类中每个节点能够存储的原始数据量满足以下关系:$D_i=S/k_L(h_i)<S/k_L(h_j)=D_j$,即低可用性的节点能够存储的数据量比高可用的节点的数据量要少.这样,系统一方面可以依据节点对系统的贡献(包括为系统提供的存储空间大小以及存储空间可用的时间)来提供不同的服务(节点能够存储的原始数据量);另一方面,可以激励节点提高自身的在线时间,使整个系统能够进一步减少维护开销.

References:

[1] Bhagwan R, Tati K, Cheng Y, Savage S, Voelker G. Total Recall: System support for automated availability management. In: Proc. of the 1st ACM/Usenix Symp. on Networked Systems Design and Implementation (NSDI). 2004. http://www.usenix.org/events/nsdi04/

[2] Stribling J. OverCite: A cooperative digital research library. In: Proc. of the 4th Int'l Workshop on Peer-to-Peer Systems. 2005. http://www.springerlink.com/content/l05n13078786nt15/

[3] Adya A, Wattenhofer R, Bolosky W, Castro M, Cermak G, Chaiken R, Douceur J, Howell J, Lorch J, Theimer M. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. In: ACM SIGOPS Operating Systems Review. 2002. http://www.usenix.org/events/osdi02/

[4] Kubiatowicz J, Wells C, Zhao B, Bindel D, Chen Y, Czerwinski S, Eaton P, Geels D, Gummadi R, Rhea S. OceanStore: An architecture for global-scale persistent storage. In: Proc. of the 9th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. 2000. 190−201. http://foothill.lcs.mit.edu/asplos2k/

[5] Dabek F, Kaashoek M, Karger D, Morris R, Stoica I. Wide-Area cooperative storage with CFS. In: Proc. of the 18th ACM Symp. on Operating Systems Principles (SOSP 2001). 2001. http://portal.acm.org/citation.cfm?id=502059.502054

[6] Blake C, Rodrigues R. High availability, scalable storage, dynamic peer networks: Pick two. In: Proc. of the 9th Workshop on Hot Topics in Operating Systems. 2003. http://www.usenix.org/events/hotos03/

[7] Tati K, Voelker G. On object maintenance in peer-to-peer systems. In: Proc. of the 5th Int'l Workshop on Peer-to-Peer Systems. 2006. http://iptps06.cs.ucsb.edu/

[8] Tian J, Dai Y. Understanding the dynamic of peer-to-peer systems. In: Proc. of the 6th Int'l Workshop on Peer-to-Peer Systems. 2007. http://www.iptps.org/papers-2007/TianDai.pdf

[9] Bhagwan R, Savage S, Voelker G. Understanding availability. In: Proc. of the 2nd Int'l Workshop on Peer-to-Peer Systems (IPTPS 2003). 2003. http://www.springerlink.com/index/EHCFGW36N3J1YPR6.pdf

[10] Saroiu S, Gummadi P, Gribble S. A measurement study of peer-to-peer file sharing systems. In: Proc. of the Multimedia Computing and Networking 2002 (MMCN 2002). 2002. http://www.cs.princeton.edu/courses/archive/fall02/cs597C/P2P/PerfScalability/

[11] Weatherspoon H, Chun B, So C, Kubiatowicz J. Long-Term data maintenance in wide-area storage systems: A quantitative approach. Computer, 2005. http://www.eecs.berkeley.edu/~bgchun/csd-05-1404.pdf

[12] Weatherspoon H, Kubiatowicz J. Erasure coding vs. replication: A quantitative comparison. Proc. of the IPTPS, 2002,2.

[13] Lin W, Chiu D, Lee Y. Erasure code replication revisited. In: Proc. of the 4th Int'l Conf. on Peer-to-Peer Computing. 2004. 90−97.

[14] Utard G, Vernois A. Data durability in peer to peer storage systems. In: Proc. of the IEEE Int'l Symp. on Cluster Computing and the Grid. 2004. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1336553

[15] Rodrigues R, Liskov B. High availability in DHTs: Erasure coding vs. replication. In: Proc. of the 4th Int'l Workshop on Peer-to-Peer Systems. 2005. http://www.informatik.uni-trier.de/~ley/db/conf/iptps/iptps2005.html

[16] Bhagwan R, Savage S, Voelker G. Replication strategies for highly available peer-to-peer storage systems. In: Proc. of the FuDiCo: Future Directions in Distributed Computing. 2002. http://sysnet.ucsd.edu/projects/recall/papers/rep-p2p.pdf

[17] Douceur J, Wattenhofer R. Competitive hill-climbing strategies for replica placement in a distributed file system. In: Proc. of the 15th DISC. 2001. 48−62. http://www.informatik.uni-trier.de/~ley/db/conf/icdcs/

[18] Ramanathan M. Increasing object availability in peer-to-peer systems. In: Proc. of the Parallel and Distributed Process Symp. 2004. http://doi.ieeecomputersociety.org/10.1109/IPDPS.2004.1303097

[19] Schwarz TJE, Xin Q, Miller EL. Availability in global peer-to-peer storage systems. In: Proc. of the 6th Workshop on Distributed Data and Structures (WDAS). 2004. http://lsirwww.epfl.ch/wdas2004/

[20] Lin WK, Ye C, Chiu DM. Decentralized replication algorithms for improving file availability in P2P networks. In: Proc. of the IWQos 2007. 2007. http://home.ie.cuhk.edu.hk/~dmchiu/iwqos07_cye.pdf

[21] Chun B, Dabek F, Haeberlen A, Sit E, Weatherspoon H, Kaashoek M, Kubiatowicz J, Morris R. Efficient replica maintenance for distributed storage systems. In: Proc. of the 3rd Symp. on Networked Systems Design and Implementation. 2006. http://www.cise.ufl.edu/~lzhong/Papers/EfficientReplicaMaintenance4DistributedStorageSysstems.pdf

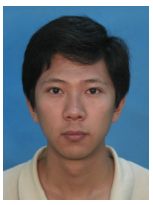
[22] Cates J. Robust and efficient data management for a distributed hash table [MS. Thesis]. MIT, 2003.

[23] Duminuco A, Biersack E, En-Najjary T. Proactive replication in distributed storage systems using machine availability estimation. In: Proc. of the Int'l Conf. on Emerging Networking Experiments and Technologies (CoNEXT). 2007.

[24] The Maze Web site. 2002. http://maze.pku.edu.cn

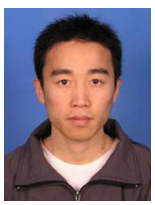
[25] Bolosky W, Douceur J, Ely D, Theimer M. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In: Proc. of the 2000 ACM SIGMETRICS Int'l Conf. on Measurement and Modeling of Computer Systems. 2000. 34−43. http://portal.acm.org/citation.cfm?doid=339331.339345

[26] Kao EPC. An Introduction to Stochastic Processes. Wadsworth Publishing Company, 1997.



杨智(1982－),男,内蒙古包头人,博士生,主要研究领域为分布存储.

代亚非(1958－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络与分布式系统,语义 Web,移动计算.



朱君(1983－),男,博士生,主要研究领域为网络与分布式系统.