面向海量数据的数据一致性研究*)

周 婧 王意洁 阮 炜 李思昆

(国防科学技术大学计算机学院并行与分布处理国家重点实验室 长沙 410073)

摘 要 复制是实现海量数据管理的关键技术之一,多副本之间的数据一致性维护是提高分布式系统的容错能力与性能的重要保证。强一致性确保并发的修改操作不会发生冲突,但是限制了系统的可用性、连通性以及副本数量;弱一致性确保副本的最终一致,提高了系统的容错能力。本文从已有的一致性维护方法出发,结合海量数据的特点,对一致性维护过程中所涉及的更新发布、更新传播方式、更新传播内容以及更新冲突解决等几个方面进行了分析,提出了相应的解决方法。

关键词 海量数据,复制,数据一致性

Research on Massive Data Oriented Data Consistency

ZHOU Jing WANG Yi-Jie RUAN Wei LI Si-Kun

(National Key Laboratory for Parallel and Distributed Processing, School of Computer Science, National University of Defense Technology, Changsha 410073)

Abstract Replication is the key technology of massive data management. Data consistency of replicas is one of the keys for achieving fault tolerance and performance enhancement in distributed systems. Strong consistency ensures that concurrent updates will not conflict but limits system availability, throughput, and the practical degree of replication, while weak consistency only guarantees eventual agreement but provides strong fault tolerance. The current techniques of research in data consistency are studied. Then, according to the characteristic of massive data, the problems such as update issuer, update propagation manner, update propagation content and conflict resolution that should be studied in data consistency are brought forward and the feasible methods are proposed respectively.

Keywords Massive data, Replication, Data consistency

1 引言

广域的分布计算环境中,数据密集且高性能的计算应用要求TB或PB级信息的有效管理和传输,这样的应用有虚拟现实、高能物理、天文学、生物工程等领域的实验分析和模拟,分布在世界各地的成百上千的研究人员共享海量数据集。研究人员总是期望只要本地系统可用,就可以访问到这些数据,并且所接收的服务好像在本地提供的一样。传统的解决方法是将需要处理的远程数据传送到本地,但是由于网络延迟且数据量大,数据传输的时间延迟不能满足数据访问的要求。高效地访问数据是对构建分布应用的严峻挑战。

复制是实现海量数据管理的关键技术之一[1~3],可以提高系统可用性、减少访问延迟和提高访问效率。由于系统中有多个副本存在,因此即使某些结点不能正常工作,仍然可以获取访问的数据,从而提高了系统的可用性。另外,用户可以避免远程网络访问而从附近的结点获取数据,或者从负载小的结点获取数据,从而提高了访问效率。复制需考虑副本数目、副本放置、复制粒度以及数据更新和一致性等问题。

在分布式环境中,数据是动态变化的,用户在每个副本结点都可能更新访问到的数据,因此必须对同一数据对象的所有副本进行一致性维护,向外界提供统一的视图。许多分布式系统都把一致性维护作为研究重点[4~6],但是一致性势必以牺牲计算开销、存储空间或通讯开销等作为代价。

分布式系统中结点数目众多,各个结点上的数据信息资源异构且数目巨大,导致系统组织、管理复杂,同时每个数据对象可能达到GB甚至TB,这些特点对海量数据的一致性维护提出了新的挑战。另外,虽然在复制时需要控制副本数量,从而平衡系统可用性和系统维护开销,但是随着分布规模的扩大,系统中副本的数量相对而言是不断增加的,副本数量众多在一定程度上给分布数据管理带来了相当的复杂性。我们从一致性维护所涉及的主要内容出发,结合海量数据的特点,对海量数据的数据一致性维护进行讨论和设计。

2 一致性分类

一致性分为强一致性[7,8]和弱一致性[9,10]两种。强一致性是指对数据的任意修改,都将同时作用到该数据的所有副本上,所有的副本在任何时刻都保持一致;弱一致性(又称为最终一致性)不执行同步修改,修改消息首先传送给一个副本,然后异步地传送给其他副本,最终每个副本都会接收到修改消息,从而达到一致状态。

*)基金项目:国家"九七三"重点基础研究发展规划基金项目(2002CB312105)、高等学校全国优秀博士学位论文作者专项基金项目(200141)、国家自然科学基金项目(69903011)。周 婧 博士生,主要研究方向为网络计算、虚拟现实等;王意洁 博士,教授,主要研究方向为网络计算、数据库技术、移动计算等;阮 炜 硕士研究生,主要研究方向为网络计算;李思昆 教授,博士生导师,主要研究方向为虚拟现实、VLSI和CAD等。

强一致性确保并发的修改操作不会发生冲突，但是限制了系统的可用性、连通性以及副本数量。该方法最主要的缺点是对硬件的要求非常高，大量结点同步几乎是不可能的；另外，对分布系统的稳定性和连通性要求也比较高，一旦某个副本不可用，则可能导致整个系统的瘫痪。与强一致性相比，弱一致性放松对数据一致性的要求，提高了系统包容通讯失效和结点失效的能力。

弱一致性是对复制算法的最低要求，如果满足不了最终一致，副本内容可能总是保持在"被破坏"的状态，从而导致放弃该副本甚至整个系统不可用；其次，弱一致性提供的最终一致性服务总是尽最大努力在副本之间快速地传播更新，实际上对许多应用来讲这已经足够了。我们在下面的叙述中涉及到的一致性均为弱一致性。

3 海量数据的一致性维护

更新一致性维护的流程大致是：用户更新所访问的数据对象，并提交到系统中；系统根据设计的一致性维护方法在多个副本间进行更新传播；副本按照不同的顺序接收更新，然后根据一定的规则应用更新，最终达到一致状态，如图 1 所示。由于海量数据的数据特点，其一致性维护在该流程中遇到许多新的问题，下面一一进行阐述。

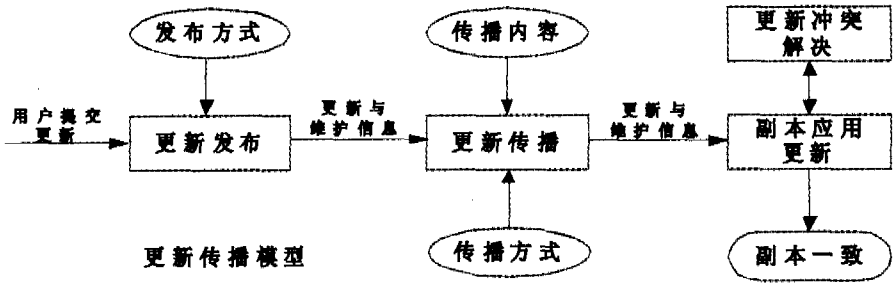

图 1· 数据一致性维护流程示意图

3.1 更新传播模型

当一个系统中每个对象存在很多个副本时，会导致传播延迟增加、副本负载不平衡以及更新频率提高、冲突增加、确认复杂，系统的性能在很大程度上受更新传输模型的影响。更新传播模型包括更新发布和更新传播方式两个方面的内容，这两个方面是相互关联的。

(1)更新发布

系统中存在的多个副本对于用户来讲是透明的。从用户的角度来讲，用户更新所访问的数据对象，并提交到系统中。发布用户提交的更新，有单主本和多主本两种策略[11,12]。单主本方法中每个数据对象指定一个副本作为主本，所有的更新最初由主本接收，然后再传播给其他副本。该类系统主要的优点是简单，缺点是在主本会出现单点失效。多主本方法中每个数据对象的任何副本在任何时间都可以发布更新，系统在后台进行更新处理。该类系统可用性高；缺点是算法复杂，存在丢失更新问题。

更新传播延迟增加、负载不平衡在单主本方法中表现得较为明显。单主本系统中主本一方面负责对各个副本发送来的更新进行确认，另一方面要将已经确认的更新分发给其他副本，主本所在结点很容易成为瓶颈。另外，每个结点都要以主本为基准进行更新推进，如果某次主本发送给远程副本的更新消息没有被接收到，远程副本结点仍然向主本发送消息，而此时其附近的某个副本可能已经接收到最新的更新。该问题的解决方法是将副本按照一定的拓扑结构进行组织，更新沿着拓扑结构中的边进行传播。例如将所有副本按照一定的规则组织成树形结构[13]，主本为根结点，更新沿着树形结构从根结点向叶结点波动。采用该方法后，传播延迟由 $O(N)$ 降低为 $O(\log N)$（N 为副本的数量），主本的负载减少为常量。

(2)更新传播方式

更新在副本间传播的方式有：direct mail 方式，一种不可靠的组播，尽可能地将更新传播到多个副本；rumor mongery 方式，将最近的更新从一个副本传播到另一个副本；anti-entropy 会话方式，两个副本阶段性地交换自身已知的更新，直到达成一致。在这三种传播方式中，只有 anti-entropy 方式可以确保更新被传递到所有副本[9]。在选择 anti-entropy 会话对象时有随机选择策略、确定性策略和基于拓扑的策略等[14]。

在副本之间进行 anti-entropy 会话，可以解决单主本发布更新遇到的问题。每个副本维护一个时间戳向量，记载其他副本的最新更新进度，每次选择进度最慢的副本相互交换更新。实验证明，该方法在传播时间上要优于前面提到的基于树形结构的解决方法。多主本系统中可以对拓扑结构的思想进行改进：将所有副本连接为树形结构，由于树中有多个结点可以发布更新，因此可以通过"捷径"提高更新传播的速度，同时采用组播协议有效分发更新。但是组播不能保证每个副本都一定接收到更新消息，因此需要其他可靠算法进行支持。

3.2 更新传播内容

对于数据对象本身很大的分布式存储系统，一致性维护设计时首先确定更新传输的内容。在传播内容上有传播更新的语义描述（称为更新日志[4]）或更新后的对象内容[15]两种选择。

(1)日志传输

传输更新日志可以更灵活地处理更新冲突、减少计算和网络开销，但是需要对更新日志进行维护。更新日志的产生主要依赖于更新发生的频率以及更新本身的大小，而不是对象的大小，因此大型对象更新时更适宜传输更新日志。同时我们还必须注意到，如果对象本身很大，且更新力度较强，将产生大量的更新日志。更新日志中每项的数据量很大，一方面在维护更新日志时占用了较多的本地存储空间，另一方面更新传输时增加了网络负载，加大了更新传播延迟，降低了系统可用性。

对于前一个问题的解决办法是提高系统对更新的确认效率，从而减小本地记载的更新日志中历史纪录的长度；通过设

计一些日志维护策略，在对一致性维护不造成恶意影响的前提下，进行日志内容的有效取舍。对于后一个问题，我们可以借鉴数据传输上的方法，如流水传输、多对一传输等方法，提高传输的速度。由于副本一致性维护中更新传播本身就需要很多的控制，因此需要将更新传播机制与提高传输效率二者有效地结合起来。

(2)内容传输

传播更新对象比较简单，但会网络开销大。内容传输的代价随着对象大小的增加线性增长。对于大型对象而言，传输更新内容的代价尤为昂贵。但是，与日志传输相比，内容传输控制起来较为简单，因此研究人员对内容传输进行了改进：积极增量技术[16]，在更新发送方的内容与远程副本内容不匹配时，不是传输全部内容，只是传输两个对象内容不同的部分，这种方法节约了网络带宽，但是增加了计算开销；另一种方法是把对象分割成一组子对象，这些子对象组成层次结构，然后在每层采用 Thomas 写规则[17]，该方法可以扩展为允许结点仅仅复制层次结构内的子树，从而进一步地减少复制开销。

3.3 更新冲突解决

更新一致性协议中必须解决哪些更新操作被确认以及按何顺序确认的问题，目前对更新主要有全排序和部分排序两种。

(1)全排序

全排序就是所有副本按照相同的顺序应用更新，典型的实现方法有 Golding 等人[14]提出的通过时间戳向量获取其他副本的状态、Bayou 系统[18]采用的宿主副本方法以及 Deno 系统[19]提出采用 quorum consensus 协议的方法等。时间戳向量方法的优点是分散执行，不会放弃任何更新，但是某个副本失效可能阻止所有其他副本前进(活锁问题[12])。随着副本数量的增加，这种情况会变得更糟。基于宿主方法中，只要宿主副本可以工作，它就可以对更新及时排序，因此可以解决活锁问题；并且宿主副本可以不理会其他副本而采用任何方式解决更新冲突。但是宿主方法有两个缺点：第一，在更新数据对象时，宿主副本可能成为系统的瓶颈；第二，在分布式环境中，一旦所有的结点都无法与宿主副本连接，则任何更新操作都不能被确认，任何应用程序不能执行，宿主副本的失效导致整个系统处于瘫痪状态。最后一种方法，如果将所有的选票分配给一个副本，则类似于宿主方法；如果给每个副本分配相同的选票权值，则类似传统的投票模式，因此其执行空间比较大。该方法的缺点是在某个时刻只允许一个更新胜出，所有其他更新因为没有赢得绝大多数选票而被丢弃，因此该方法不适合通讯延迟大以及更新频率高的应用。

由于每种更新冲突解决方法都有各自的优点和存在一定的弊端，因此在系统设计时总是难以取舍。对于大规模的分布式海量数据管理系统，可以结合当前网格技术的发展，采用一种局域集中、广域对等相结合的方法解决更新冲突，使其在可扩展性、负载均衡和性能等方面均有获益。另外，由于分布式海量数据管理系统中资源是异构的，不同类别资源对一致性维护的要求不同，应该区别对待。例如，海量数据的元数据信息一般存储在数据库中，每个数据项内容比较单纯，但数目繁多，并且对底层数据的修改常常涉及到元数据信息的更新。因此，元数据信息的更新粒度比较小，但是更新频率高。由于元数据是关于数据的数据，如果元数据的一致性维护不够精确，会导致访问底层数据出现错误(如定位到错误的文件)。元数据的一致性维护通常采用传播更新日志的方法，基于既不丢失更新、又要快速地确认更新的出发点，采用宿主副本方法进行更新确认。

(2)部分排序

部分排序可以解决全排序方法中所忽略的更新间的语义关系(如更新的可交换性)而导致的不必要的更新应用延迟，以及全排序不能保存不同副本发出的更新之间的依赖关系等问题。解决方法有：给每个更新附加上前提更新的名字，更新必须等到所有前提更新执行之后[20]，但是一次更新的大小随着它依赖的前提更新的增加可能会无限增加。

目前大多数的系统对更新都是进行全排序，但是适当地引入部分排序对于较高的更新频率不失为一种解决方法。对于多主本系统，可以取定时间阈值 Δt，该阈值是一次更新可以被另一副本结点接收到的最短时间，因此在内不同副本结点上的并发更新存在一定的可交换性，对其可以进行部分排序，通过该方法减少不必要的更新延迟。

上述海量数据的数据一致性维护内容可以用图 2 进行形象描述，图中给出了数据一致性维护的技术层次及其关键问题，用折线进行连接的内容说明其间关系密切，未标注的技术点之间也存在一定的联系。总之，设计实用有效的数据一致性维护方法，需要从多方面进行综合考虑。

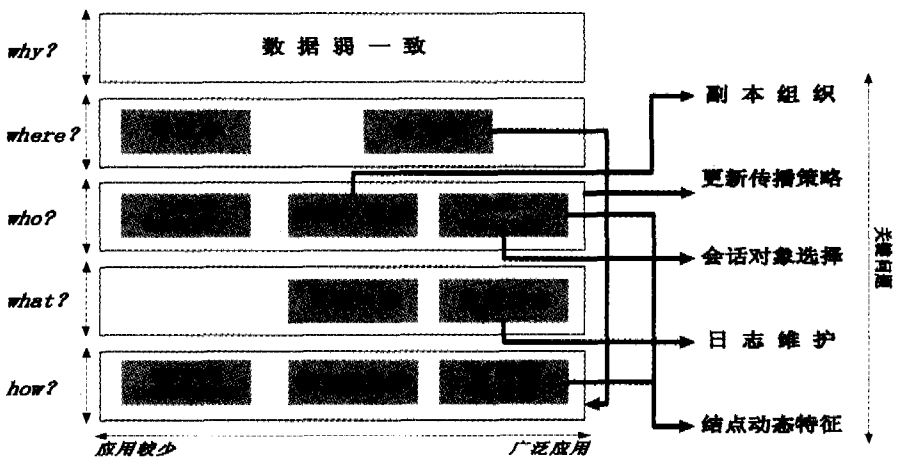

图 2 数据的弱一致性维护方法示意图

3.4 并发更新面临的挑战

现在的许多系统都设计为允许任何副本在任何时刻可以

发布更新，同一对象的多个相互冲突或不冲突的更新在系统中流动。并发的更新会带来很多问题，尤其对于海量数据而言，这些问题会表现得更为突出。虽然存在多种更新解决方法，但是对于有些冲突是很难解决的。

例如，系统中某个 word 文档分为 A、B 两个段落，副本结点 S_i 在时间 t_0 对其中的段落 A 进行了更新（记为 $U_{i,0}$），结点 S_j 在时间 t_1（$t_0<t_1$）对段落 B 进行了更新（记为 $U_{j,1}$），且两个更新均没有被确认。根据更新确认的基本规则，认定更新 $U_{j,1}$ 必须等到更新 $U_{i,0}$ 执行完毕之后才可以应用（如图 3a 所示）。而实际上，这是两次“伪冲突”更新，完全可以并发执行。由于副本数量巨大，更新发生频率高，因为“伪冲突”而被延迟的更新不仅数量上多，而且阻止了更新的推进，降低了副本的可用性。另外，随着每个数据对象的增大，该类并发更新的发生频率也会随之提高。这种情况的解决办法是增加对同一对象的更新检测，但会产生大量的计算开销；另外，可以将相对大的数据对象进行分段（如图 3b 所示），将每段数据作为一个独立的数据对象对待，但是仍然不能从根本上解决这个问题。

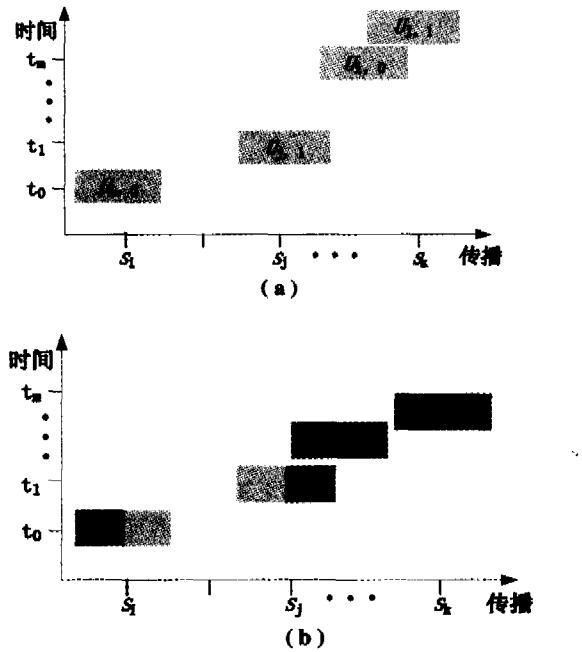

图 3　并发更新问题及解决方法示意图

让我们来看另一种情况。同样，假设系统中某个 word 文档分为 A、B 两个段落，副本结点 S_i 在时间 t_2 对其中的段落 A 进行了更新（记为 $U_{i,2}$），结点 S_k 在时间 t_3（$t_2<t_3$）也对段落 A 进行了更新（记为 $U_{k,3}$），且更新 $U_{k,3}$ 以原始段落 A 的内容为前提条件。按照更新确认规则，更新 $U_{i,2}$ 先于 $U_{k,3}$ 执行。由于更新传播延迟，结点 S_k 在发布更新 $U_{k,3}$ 时还没有看到更新 $U_{i,2}$，这样就导致在更新确认阶段更新的前提条件已经不能满足（$U_{i,2}$ 将其前提条件已经进行了修改）或前提条件的范围被扩大（$U_{i,2}$ 将某些内容更新为可以满足 $U_{k,3}$ 的前提条件）。这两种情况都将违背用户的本意，破坏了数据，只能拒绝其中某次更新。但由于无法有效地区分该类冲突与其他冲突，因此在系统设计时一味地拒绝并发的更新，会降低系统的可用性。

在多主本系统中（假设所有的更新之间都不会出现以上两种情况），如果每个副本等概率地发布更新，则系统中流动的更新数量多、更新冲突高，任何一种更新传播和冲突解决策略都不能取得令人满意的结果。这时必须寻求将众多的更新（副本）组织起来的有效方法，提高更新传播和确认的速度。

近年来，数据网格（data grid）[2,21,22] 和 P2P（peer to peer）[3,23] 的研究引起了广泛的关注。数据网格和 P2P 系统存在自治性、异构性、动态性、规模巨大等相似的特点，为海量数据的分布共享提供了好的平台。这两类系统先前的研究重点集中在大规模、具有 ad-hoc 结构的分布系统的信息分布（副本放置）和信息发现（副本定位）上，而对信息的访问通常局限为只读，或出现更新冲突时，通过人工干预进行解决，从某种程度上来讲不提供分布应用的一致性保障。但是随着应用需求的发展，这些系统开始将更多的注意力放在大规模分布系统中的信息共享所带来的问题，其中就包括信息一致性维护方面的问题。我们的研究必须充分考虑到网格和 P2P 分布存储系统的特点，使得数据一致性维护在可扩展性、性能、可靠性、自组织性等各个方面都具有良好的特征。

结论　复制可以提高分布式存储系统的性能、可靠性和可用性，但同时带来了必须保证副本一致性的问题。本文在介绍数据一致性基本理论的基础上，针对海量数据的特点，对数据一致性维护过程中的关键问题进行讨论并提出了相应的解决办法。下一步的工作将进一步深入各种解决办法，提出详尽的解决方案并进行模拟。

参 考 文 献

1　Gray J, Helland P, O'Neil P, et al. The dangers of replication and a solution. ACM SIGMOD Record, 1996, 25(2): 173～182

2　Chervenak A L, Foster I, Kesselman C, et al. Data management and transfer in high performance computational grid environments. Parallel Computing Journal, 2002, 28(5): 749～771

3　Kubiatowicz J, Bindel D, Chen Y, et al. OceanStore: an architecture for global-scale persistent storage. ACM SIGARCH Computer Architecture News, 2000, 28(5): 190～201

4　Petersen K, Spreitzer M J, Terry D B, et al. Flexible update propagation for weakly consistent replication. In: 16th ACM Symposium on Operating Systems Principles. New York: ACM Press, October 1997. 288～301

5　Yu Haifeng, Vahdat A. The costs and limits of availability for replicated services. In: 18th ACM Symposium on Operating Systems Principles. New York: ACM Press, October 2001. 29～42

6　Dahlin M, Gao L, Nayate A, et al. PRACTI replication for large-scale systems: [Technical Report TR-04-28]. The University of Texas at Austin, 2004

7　Bernstein P A, Hadzilacos V, Goodman N. Concurrency control and recovery in database systems. Boston, MA, USA: Addison-Wesley Longman Publishing Co, Inc, 1987

8　Davidson S B, Garcia-Molina H, Skeen D. Consistency in partitioned networks. ACM Computing Surveys, 1985, 17(3): 341～370

9　Schroeder M D, Birrell A D, Needham R M. Experience with Grapevine: the growth of a distributed system. ACM Transactions on Computer Systems, 1984, 2(1): 3～23

10　Frolund S, Kalogeraki V, Pedone F, et al. Scalable state replication with weak consistency. In: Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03). Oakland: IEEE Computer Press, June 2003. 96～105

（下转第 161 页）

中选择了500,1000,1400,1800,2200,2600个样本组成训练集,600个样本组成测试集,比较单个朴素贝叶斯分类器,BEPOL算法,以及AdaBoost算法的学习曲线,除朴素贝叶斯分类器外,每个算法运行10次,并以这10次运行结果的平均值作为最终的结果,如图1所示。

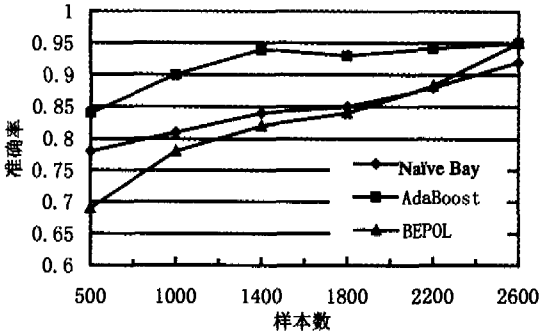

图1 基于数据集Chess的学习曲线

从图1可以看出,在线学习算法BEPOL具有与AdaBoost算法相近的分类性能,尽管算法BEPOL的初始性能不如AdaBoost,甚至比单一贝叶斯分类器更差,但随着训练样本集的增加,算法BEPOL的性能稳定增长,并最终取得与批量集合学习算法相同的分类能力。

表2是算法BEPOL与朴素贝叶斯分类器、AdaBoost算法基于样本集Breast Cancer、Mushroom、Waveform-40、Credit-g的分类性能比较结果。对朴素贝叶斯分类器和AdaBoost算法,我们采用5倍交叉验证方法,将交叉验证的平均值作为算法的有效结果。并行算法因为受样本顺序性的影响,我们采用5轮样本随机序列作为在线样本集,并以其平均值作为有效结果。

表2 BEPOL、朴素贝叶斯分类器、AdaBoost算法在UCI样本集上的分类正确率

数据集	Naïve Bayesian	AdaBoost	BEPOL
Breast Cancer	0.752	0.938	0.896
Credit-g	0.795	0.821	0.814
Waveform-40	0.842	0.826	0.837
Mushroom	0.927	0.969	0.975

通过比较我们看到,在Breast Cancer中,AdaBoost算法明显好于BEPOL,但在Waveform-40和Mushroom上,BEPOL表现则优于AdaBoost算法,在数据集Credit-g上,则性能几乎相差不大,而且我们注意到,随着样本集容量的增长,BEPOL算法的优势体现得更加明显。

结论 Boosting算法可以用于分类器集合的学习,并提高总体的分类性能。但对于连续型样本集,它需要缓存大量的数据,这会显著增加算法的时间和空间开销。对此我们提出了一种Boosting方式的在线集合学习算法,它利用与AdaBoost算法相同的样本加权采样思想,通过减少对同一样本的采样次数,满足在线学习中对样本的要求。进一步的实验证明,在数据量充足的情况下算法BEPOL具有与AdaBoost算法相近的分类能力,并可同时用于在线学习和批量学习过程。

参考文献

1 Shi X, Manduchi R. A study on bayes feature fusion for image classification. In: Workshop on Statistical Analysis in Computer Vision, (Madison, WI), 2003

2 Bauer E, Kohavi R. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Machine Learning, 1999, 36 (1-2): 105~139

3 Kivinen J, Warmuth M K. Additive versus exponentiated gradient updates for linear prediction. 2000

4 Fern A, Givan R. Online ensemble learning: An empirical study. In: Proc. of the Seventeenth International Conference on Machine Learning, 2000. 279~286

5 Zhang T. On Sequential greedy approximation for certain convex optimization problems: [Technical report]. IBM T. J. Waston Research Center, 2002

6 Freund Y, Schapire R E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Computer and System Sciences, 1997, 55(1):119~139

7 Chelba C, Acero A. Conditional maximum likelihood estimation of naive Bayes probability models using rational function growth transform: [Technical Report MSR-TR-2004-33]. Microsoft, 2004

8 Blake C, Keogh C J M. UCI repository of machine learning databases. 1999

(上接第140页)

11 Demers A, Greene D, Hauser C, et al. Epidemic algorithms for replicated database maintenance. In: 6th ACM Symposium on Principles of distributed computing. New York: ACM Press, 1987. 1~12

12 Saito Y. Consistency management in optimistic replication algorithms. http://www.hpl.hp.com/personal/Yasushi_Saito/replica.pdf, June 2001

13 Adly N. Management of replicated data in large scale systems: [PhD dissertation]. UK: Corpus Christi College, University of Cambridge, August 1995

14 Golding R A. Modeling replica divergence in a weak-consistency protocol for global-scale distributed data bases: [Technical Report UCSC-CRL-93-09]. University of California at Santa Cruz, February 1993

15 Saito Y, Levy H, Bershad B H. Manageability, availability and performance in Porcupine: a highly scalable, cluster-based mail service. ACM Transactions on Computer Systems, 2000, 18(3): 298~332

16 Saito Y, Levy H M. Optimistic replication for Internet data services. In: 14th International Conference on Distributed Computing. London: Springer-Verlag, October 2000. 297~314

17 Thomas R H. A majority consensus approach to concurrency control for multiple copy databases. ACM Transactions on Database Systems, 1979, 4(2): 180~209

18 Terry D B, Theimer M M, Petersen K, et al. Managing update conflicts in Bayou, a weakly connected replicated storage system. In: 15th ACM Symposium on Operating Systems Principles. New York: ACM Press, December 1995. 172~183

19 Keleher P J. Decentralized replicated-object protocols. In: 18th ACM symposium on Principles of distributed computing. New York: ACM Press, May 1999. 143~151

20 Birman K P, Joseph T A. Reliable communication in the presence of failures. ACM Transactions on Computer Systems, 1987, 5 (1): 47~76

21 Qin Xiao, Jiang Hong. Data Grid: Supporting Data-Intensive applications in Wide-Area Networks: [Technical Report TR-03-05-01]. University of Nebraska-Lincoln, May 2003

22 Lamehamedi H, Szymanski B, shentu Z, et al. Data replication strategies in Grid environments. In: Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02). Oakland: IEEE Computer Press, 2002. 378~383

23 Cai Min, Chervenak A, Frank M. A peer-to-peer replica location service based on a distributed hash table. Proceedings of the ACM/IEEE SC2004 Conference. Pittsburgh, Pennsylvania, 2004