采用动态投票机制的数据网格副本一致性维护

吴长泽[1] 田 东[1,2] 吴中福[3]
(重庆大学通信工程学院 重庆 400030)[1] (贵州省电子计算机软件开发中心 贵阳 500050)[2]
(重庆大学计算机学院 重庆 400030)[3]

摘 要 针对网格动态性引起副本数据一致性维护困难的问题,根据网格的动态特性建立了网格系统模型和副本一致性维护模型;提出了适用于低在线率情况的动态投票机制,并在此基础上给出了一种副本数据一致性维护算法,从全局有序性和读一致性等方面证明了该算法的正确性。最后通过模拟实验对副本数取值及可扩展性等方面进行分析,探讨了网格动态性对数据一致性维护的影响。

关键词 数据网格,数据副本,数据一致性,动态投票机制
中图法分类号 TP393.01 **文献标识码** A

Strategy of Keeping Replica Consistency with Dynamic Voting Mechanism in Data Grid

WU Chang-ze[1] TIAN Dong[1,2] WU Zhong-fu[3]
(College of Communication Engineering, Chongqing University, Chongqing 400030, China) [1]
(Guizhou Electronic Computer Software Development Center, Guiyang 550005, China) [2]
(College of Computer Science, Chongqing University, Chongqing 400044, China) [3]

Abstract Aiming at the problem that it hard to keeping replica data consistency due to dynamic character in data gird, a keeping replica data consistency strategy in grid environments was presented. According to the dynamic character of Grids, models for Grid systems and replica data consistency respectively were established. Combining dynamic voting mechanism, the presented data consistency algorithm satisfies low online probability. Its correctness was proofed with global ordered and read consistency. At last, simulation shows the influence of dynamic on keeping replica consistency from replica number and scalability in data grid.

Keywords Data grid, Data replica, Data consistency, Dynamic voting mechanis

为了获得更高的可用性、可靠性和安全性,提高系统效率,通常会在数据网格中引入多个数据副本[1],如何保证这些副本数据的一致性是数据网格研究中的一个重要问题。

副本一致性维护研究针对分布式存储系统的较多[2,3],而针对网格特性的研究工作目前还较少。网格系统与传统的分布式存储系统相比较,具有更大的规模和高度的动态性,因此,在副本数据一致性维护过程中对可扩展性、数据副本可用性及通信开销方面都有更高的要求。这使得分布式存储系统中已有的一致性维护方法不能直接应用到网格环境中。

为了获得良好的可扩展性,采用投票机制较为符合网格副本维护的需求。它通过减少一致性维护过程中读写操作节点参与数,来改善系统性能和增强可用性,降低系统内通信开销从而达到提高系统可扩展性的目的。然而,现有的投票机制在投票系统总规模和投票表决约束条件两方面表现为静态[4-6]。在高度动态的网格环境下,副本节点在线率可能很低,很难达到现有的半数投票约束值。因此,需要采用动态投票机制来满足网格环境的需求。

基于上述分析,本文提出一种动态投票机制,动态确定表决约束值;在此基础上给出了网格副本数据一致性维护算法,重点解决了网格动态性造成的副本数据不一致问题。

1 副本一致性维护模型

Dirk Dullmann[1]指出在网格环境中100%的数据一致是不可能达到的,可以放宽严格一致性的要求,允许某些部分数据在一段时间内不达到同步一致。基于这种思想,结合网格系统的特点,建立如下网格副本异步更新一致性模型。

1.1 副本系统模型

定义1 若存在数据文件 R,被复制到5个节点上,分别为 $R1,R2\cdots R5$,则数据文件 R 称为逻辑副本名,$R1,R2\cdots R5$ 称为物理副本名。

网格系统由有限个节点构成,每个节点存储一个或多个副本。同一逻辑副本名的副本节点构成一个投票系统 *Quorum*,假设网格系统节点全集 $S=\{Q_1,Q_2,\cdots,Q_n\}$,则 $\forall Q_i,Q_i\in Q,Q_i\cap Q_j\neq\emptyset$。$Q_i$ 内每个物理副本关联一个 *Key* 值,这样同

到稿日期:2008-10-10 本文受贵州省科技厅2007年度工业科技攻关计划项目(黔科合GZ字(2007)3005),贵州省科学技术基金项目(黔科合J[2007]2232号)资助。
吴长泽(1980—),男,博士后,主要研究方向为网格计算、可信计算,E-mail:wczhero@126.com;田 东(1975—),男,博士后,主要研究方向为网格计算、容错与诊断;吴中福(1941—),男,教授,博导,主要研究方向为远程教育、网格计算、可信计算等。

一逻辑副本可看作一个有序的集合 $Q=\{Key1, Key2, \cdots, KeyN\}$。

1.2 副本更新一致性模型

在上述网格系统模型下，构造图 1 所示的副本更新一致性模型。在参考文献[7]的基础上将新增和改进的一致性组件描述如下。

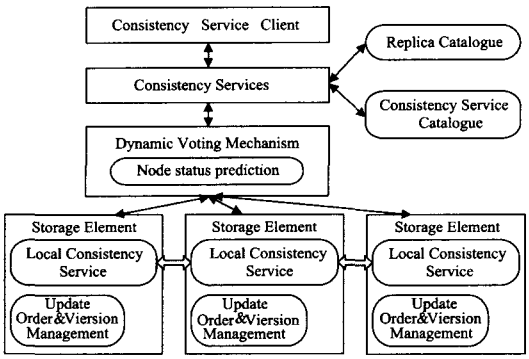

图 1 副本更新一致性模型

动态投票机制：通过互斥协议处理冲突，通过选举协议为读写操作投票，确定是否提交该操作。选举过程中通过马尔科夫预测模型动态预测节点状态来确定 *Quorum* 规模及表决票数约束值。

版本控制管理：每个物理副本节点均管理 *Prep-Update* 和 *Commit-Update* 两个有序集合，*Prep-Update* 存储写操作请求，并返回写入顺序 *Order*；*Commit-Update* 存储经过互斥协议判断的可提交的写请求，并返回本地副本版本 *Version* 值。

2 动态投票机制

动态投票机制采用互斥协议和选举协议进行副本一致性维护。互斥协议主要解决写冲突和读冲突；选举协议主要在含有无效副本节点的系统中通过投票表决确定读写操作是否有效。在网格环境中，满足读一致性的前提下，动态确定表决约束值。

2.1 互斥协议

• 写操作互斥

同一逻辑副本名的投票系统 Quorum 内，对各节点提交的写请求，比较优先权，*Key* 值小的节点优先级高。如果写请求之间存在同步关系，则优先级低的写请求等待优先级高的写请求执行后才执行；若无同步关系，则 Order 值高的写请求被排斥。

• 读写操作互斥

同一逻辑副本名的投票系统 Quorum 内，如果读请求的目标数据正执行写操作或对该目标数据的写请求在 Commit-Update 中，则读请求被排斥。

2.2 动态投票选举协议

客户端提出读写请求后，通过马尔科夫预测模型动态确定表决约束条件。读请求“先投票，再发送”，即在发送前预测节点状态进行投票，并只向预测结果达到约束条件的节点发送请求；写请求“先发送，再投票”即在向投票系统 Quorum 内所有节点发送请求后，通过预测返回消息的节点下一时段的节点状态，再判断是否满足写投票约束条件。

定义 2 按照在线状态和版本是否过期有效，副本状态可划分为“在线有效 *OnC*”、“在线无效 *OnW*”、“离线有效 *OffC*”和“离线无效 *OffW*”。

• 投票权值 *Weight* 的确定

设所测试物理副本的平均在线概率为 p，副本节点 i 离线后到再次上线期间发生了成功更新的概率为 $P_{u=1}$，未发生成功更新的概率为 $P_{u=0}$。

步骤一 通过读写操作日志的统计结果，计算出 p，$P_{u=0}$，$P_{u=1}$，则状态转移矩阵：

$$P=\begin{bmatrix} p & pP_{u=1} & (1-p)P_{u=0} & 0 \\ pP_{u=1} & pP_{u=0} & 0 & 1-p \\ pP_{u=0} & pP_{u=1} & (1-p)p_{u=0} & (1-p)P_{u=1} \\ 0 & p & 0 & 1-p \end{bmatrix}$$

步骤二 选取被预测节点上次访问时的状态作为原始序列，并在每次访问后，动态更新节点状态。

$p_{status}(0)=\{p_{onc}(0), p_{onw}(0), p_{offc}(0), p_{offw}(0)\}$

步骤三 计算预测结果，确定权值 *Weight*。

$p_{status}=p_{status}(0)P=\{p_{onc}, p_{onw}, p_{offc}, p_{offw}\}$

取 p_{onc} 为该节点可读权值。

对返回消息节点，预测下一时段该节点在线状态概率

$p_{status}'=p_{status}(0)P^2=\{p_{onc}', p_{onw}', p_{offc}', p_{offw}'\}$

取 p_{onc}' 为该节点可写权值。

• 表决约束条件的确定

投票的目的是为了确保副本读一致，即在全局节点不同步一致的情况下，读操作所读取的数据有效一致。这在至少一个副本节点在线有效的情况下即可满足。按此标准确定约束条件如下：

读投票约束：$RC=1-p_{onci}p_{oncj}\cdots p_{oncn}\geqslant Trust=90\%$

写投票约束：$WC=1-p'_{onci}p'_{oncj}\cdots p'_{oncn}\geqslant Trust=90\%$

其中，p_{onci} 表示节点 i 在线有效的概率，p'_{onci} 表示节点 i 在下一时段在线有效的概率，*Trust* 表示置信度由用户自定义，这里取 90%。

3 副本读写一致性维护算法

3.1 副本读写一致性维护方法

• 写操作方法

1)副本节点 i 提出更新请求后，将更新目标及相应值(*UpdateTarget*, *Value*)按 *Key* 顺序发送到同一投票系统 *Quorum* 中的其它所有副本节点；

2)通过互斥协议判断是否拒绝该请求。如果没有写冲突，通过动态投票选举协议判断是否返回消息节点，是否达到表决约束条件；

3)如果满足约束条件，则节点 i 取所有节点返回 *Order* 的最大值 max(*Order*)作为全局更新顺序再次进行全局确认，确认成功则按该 max(*Order*)值作为更新版本号 *Version* 对所有返回消息节点进行提交。

• 读操作方法

1)副本节点 i 提出读请求后，将读请求(*Read*, *Target*)按 *Key* 顺序发送到同一投票系统 *Quorum* 中的其它所有副本节点；

2)通过互斥协议判断是否拒绝该请求。如果没有写冲突，通过动态投票选举协议判断是否返回消息节点，是否达到

表决约束条件；

3)如果满足约束条件，则节点 i 取所有返回消息节点中最大版本号所对应的目标值。

3.2 算法描述

算法 1 副本一致性维护读写方法

```
Task 1 Write and Read Request
1:procedure read() //Read Request
2:  replies←voting([Read,Target])
3:  val←the value with highest ver from replies
4:  if state in all replies is true and ver in all replies is equation then
      return val
5:  else return recover()
6:procedure write(val) //Write Request
7:  replies←voting([Write,Target,val])
8:  if any state in a reply is false then return NOK
9:  Order←max(order in all replies)
10:  replies← message([Write,Order,val])
11:  if any state in a reply is false then return NOK
12:    ver←Order
13:  replies←message([Write,val,ver ])
14:  if the state in all replies is true then return OK
15:  else return NOK
16:procedure recover() // update to stale replica
17:  replies←message([Read,Target ])
18:  if any state in a reply is flase then return NIL
19:  val←the value with highest ver from replies
20:  replies←message([Write,val,max(ver)])
21:  if the state in all replies is true then return val else return NIL
22:procedure message(msg) //Send message to all
23:  Send msg to all,
24:  await receive(rep) from processes matches msg
25:  return set of received replies
26:procedure prediction(node) //Node status predict
27:    pstatus(0)={ponc(0),ponw(0),poffc(0),poffw(0)}
28:    if Read return
         pstatus=pstatus(0)P={ponc,ponw,poffc,poffw}
       else return pstatus'=pstatus(0)P²={ponc',ponw,'poffc,'poffw'}
29:procedure voting(msg)//voting for satify Constrains
30:  if Read then prediction(node Keyi) ∈nodes
       until satisfy1−ponci poncj⋯poncn≥Trust=90%
     message to nodes return set of received replies
31:  if Write then message to all until received replies satisfy 1−
     ponci'poncj'⋯poncn'≥Trust=90%
Task 2 Message handlers
32:  when receive [Read,Target] from coordinator
       //Read conflict handler
33  state(←(no write operation for Target)
34:  reply [Read,state,ver] to coordinator
35:  when receive [Write,Target,val] from coordinator
       //Judge Write conflict and obtain Order
36:  state← (write for Target∉preUpdate)
37:  if state then Order←order+1;
38:  reply [Order,state ] to coordinator
39:when receive [Write,Order,val] from coordinator
     // Validate Order is the max order
40:  state(←(Order≥order)
41:  if state then order++
42:  reply [Write,state ] to coordinator
43:when receive [Write,val,ver ] from coordinator
     //Write operate and get new version
44:  state←(Version≤ver)
45:  if state then Val←val and Version←ver
46:  reply [Write,val,state ]
```

3.3 正确性证明

要满足网格环境下异步更新来维护副本一致性，应该具有全局有序性[8]和读一致性才能保证算法的正确性。

定理 1 本文算法保持读写操作的全局有序性。形式化描述为：$\exists S$ is *Sequence set*，$oper_1(v)$，$oper_2(v')\in S$；if $Vers\ v\leqslant Vers\ v'$，$oper_1(v)\rightarrow_s oper_2(v')$，$oper(v)=read(v)$ or $write(v)$。$\rightarrow_s$ 表示前者在后者之前操作。

证明：根据一致性算法若 $write(v)\rightarrow_s read(v)$，则需要证明 $write(v)$和 $read(v)$之间没有写操作。

假设在 $write(v)$和 $read(v)$之间有一个写操作 $write(v')$，$write(v)\rightarrow_s write(v')(1)$，$write(v')\rightarrow_s read(v)(2)$，则对于式(1)，按照自适应冲突协调机制 $Vers\ v<Vers\ v'$；而对于式(2)，按照一致性算法读操作应读取全局最新副本 $Vers v'\leqslant Vers\ v$，导致了矛盾。因此在 $write(v)$和 $read(v)$之间没有写操作。所以一致性算法保持了读写请求的全局有序化，得证。

定理 2 本文算法满足读一致性，即在节点动态加入退出的情况下能够读取到有效数据。

证明：设逻辑副本 LRN 的初始值为 L_0，数据网格系统中所有对 LRN 的操作在 Key_1 节点上按照抵达时间排序为 Req_1，Req_2，…，Req_K。我们对请求 Req_K 采用归纳法证明。

①当 $K=1$ 时，无论 LRN 是否存在多个副本，Req_1 所读到的数据都为 L_0，因此命题成立；

②假设 $K=N$ 时，命题成立。即在数据网格系统不采用复制技术时，Req_K 所读到的数据为 L_{N-1}，LRN 在数据网格系统中存在多个副本时 Req_K 所读到的副本为 L'_{N-1}，则 $L_{N-1}=L'_{N-1}$；

③当 $K=N+1$ 时，数据网格系统不采用复制技术时，Req_K 所读到的数据为 L_N；LRN 在数据网格系统中存在多个副本时，Req_K 所读到的副本为 L'_N。

若 Req_N 为读操作时：Req_N 并没有改变数据的值，因此 $L_N=L_{N-1}$，$L'_N=L'_{N-1}$，故 $L_N=L'_N$。

若 Req_N 为写操作时：当数据网格系统不采用复制技术时，操作 Req_K 所读到的数据为 Req_N 在 L_{N-1} 上的操作结果，即 $L_N=Req_N(N-1)$；当 LRN 存在多个副本时，由于算法要求写操作提交前进行冲突协调，保证了在执行操作 Req_{N+1} 时，操作 Req_N 已经对在线节点执行完毕。由于一致性算法只能更新在线副本，设此时逻辑文件 LRN 的在线副本为 L'_{NM}，则 $L'_{NM}=Req_N(L'_{N-1})=Req_N(L_{N-1})=L_N$，根据算法 Req_K 所读到的副本在线副本的最新状态 L'_{NM}，则 $L'_N=L_N$，故命题成立。

4 模拟实验

模拟实验在数据网格模拟环境 OptorSim 上进行。读写请求按照随机方式分配到不同的物理副本节点上。模拟环境共模拟了 60 个网格节点，单个节点的存储能力为 100G，每个

(下转第 189 页)

Technology & Applications. ICITA2002. March 2002:595-600
[5] Wu Di, Yang Zongkai, Cheng Wenqing. A Mobile Agent Assisted Learning Resource Service Framework Based on SOAP. Web-based Learning: Technology and Pedagogy, July 31-August 3, 2005:125-134
[6] Labrou Y, Finin T. A Proposal for a New KQML Specification. February 1997:3
[7] 冯志勇,洪卫林. 基于 SOAP 协议的 KQML 语言通信实现[J]. 计算机工程,2003,29(6):97-98
[8] 曹军海,张和明,熊光愣. 多 Agent 仿真中 Agent 行为的形式化描述方法[J]. 系统仿真学报,2004,16(11):2399
[9] 吴砥. 学习资源管理与服务关键技术研究[D]. 武汉:华中科技大学,2006:100
[10] 王红. 移动 agent 关键技术研究[D]. 北京:中科院计算所,2002:15
[11] Venners B. The Architecture of Aglets. http://www.java_world.com/javaworld/jw-04-1997/jw-04-hood.html, April 1997

(上接第 174 页)

数据文件的大小为 500M,数据网格系统中共有 1000 个均匀分布在不同网格节点上的数据源文件。

基于上述模拟环境,分析副本数 R 的选取,以及不同在线概率对一致性维护性能的影响,并比较分析了读写请求量、请求执行时间等因素对本文提出的基于自适应冲突协调机制的一致性算法和 Andrea Domenici[4] 提出的基于分布锁的数据网格副本一致性方案的性能影响。

实验 1:副本数 R 对读一致性的影响

本文算法在至少有一个有效节点在线的情况下满足读一致性,而在网格环境下满足该条件的概率为 $P_{Read} \geqslant 1-(1-P_{OnlineCorrect})^R$。

因此,通过改变同一逻辑副本的物理副本数 R 分析在不同在线概率 p_{Online} 下对读一致性的影响。可以为 R 选取一个合适的值,使得在低在线概率情况下,可以满足读一致性。

从图 2 可以看出,随着 R 的增,大对在线概率的要求越低。当 $R=60$ 时,在线概率为 20%即可使读一致性概率达到 90%。考虑到副本越多开销越大,因此,可选取 R 为 60。

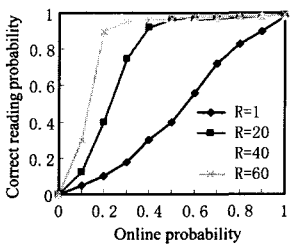

图 2 副本数 R 对读一致性的影响

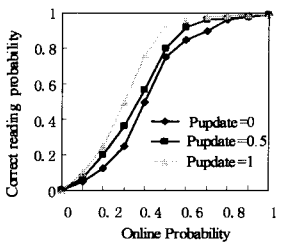

图 3 P_{update} 对读一致性的影响分析

实验 2:P_{update} 对读一致性的影响

当 $R=10$ 时,副本节点在离线与在线状态之间转换的过程中发生更新的概率 P_{update} 也是对读一致性影响的重要因素。

从图 3 可以看出,副本节点在离线与在线状态之间转换过程中发生更新比不发生更新的读一致性概率提高了 20%。因此,写概率越高,网格系统的读一致性概率就越高。

实验 3:分布锁算法的读写请求量与请求执行时间关系的比较

在 $R=60$,低在线概率 $Ponline=0.2$ 的条件下,本文算法与 Andrea Domenici[4] 提出的基于分布锁的副本一致性算法进行比较分析。

从图 4 可以看出,本文算法的执行请求的总时间明显低于基于分布锁的数据网格副本一致性算法。因此,本文算法对于读写请求量具有更好的可扩展性,性能较优。

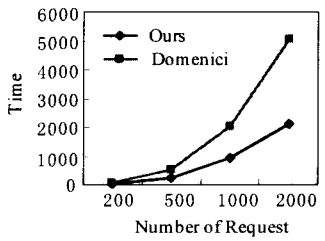

图 4 本文算法与 Domenici 分布锁算法的比较

结束语 针对网格系统高度动态异构的特点,基于动态投票机制,提出了副本一致性维护算法,通过对全局有序性和读一致性的分析,证明了算法的正确性。经仿真实验表明:在实验确定的低在线概率满足读一致性副本数 R 下,与 Andrea Domenici 所提出的基于分布锁机制的副本一致性算法相比,具有更好的可扩展性。

下步研究工作将研究与容错相合的副本一致性维护算法,考虑失效副本的一致性恢复。

参 考 文 献

[1] Düllmann D, Hoschek W, Jaen-Martinez J, et al. Models for replica synchronization and consistency in a data grid[C]// Tenth IEEE Symposium on High Performance and Distributed Computing (HPDC-10). San Francisco, CA, August 2001
[2] Proti'c J, Tomasevi'c M, Milutinovi'c V. Distributed Shared Memory: Concepts and Systems[C]//IEEE Computer Society Press: Los Alamitos, CA, 1997
[3] Levy E, Korth H F, Silberschatz A. An optimistic commit protocol for distributed transaction management// James C, Roger K, eds. Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1991:88-97
[4] Jim_enez-Peris R, Pati_no-Mart_enez M, Alonso G, et al. Are quorums an alternative for data replication? [J]. ACM Trans. Database System, 2003, 28:257-294
[5] Adzilacos V H, Toueg S. Fault-tolerant broadcasts and related problems // S. Mullender, ed. Distributed Systems, second edition. New York: ACM Press, 1993:97-145
[6] Schiper A, Sandoz A. Uniform reliable multicast in a virtually synchronous environment // Proceedings of the 13th International Conference on Distributed Computing Systems (ICDCS-13). Pittsburgh, Pennsylvania, USA, IEEE Computer Society Press, May 1993:561-568
[7] Domenici A, Donno F, Pucciani G. Replica consistency in a Data Grid[J]. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2004, 534(1/2):24-28
[8] Herlihy M, Wing J. Linearizability: A correctness condition for concurrent objects[J]. ACM Trans Programming Languages and Systems, 1990, 12(3):463-492