并行文件系统中 disk cache 一致性协议的正确性证明*

武北虹 邢汉承 黄大海

(东南大学计算机科学与工程系 南京 210018)

摘要 本文叙述了两类并行文件系统中 disk cache 一致性的维护方法,并基于 release 一致性模型,给出了其正确性证明.

关键词 并行文件系统,disk cache,一致性协议,顺序一致性,释放一致性.

超高速的并行计算机是现代科学技术的至高点,在高科技和基础研究中都发挥着极重要的作用.并行机操作系统是并行机研究的一个重要组成部分,这方面已出现了以 MACH[1],AMOEBA[2]为代表的一些并行操作系统,但它们的文件系统仍然是传统的单一文件系统,使得 I/O 瓶颈严重影响了整个系统的并行性能.目前,该矛盾已成为并行计算机系统中的一个主要问题,这就使得并行文件系统的引入成为不可避免.并行文件系统的主要思想是在硬件支持下,将磁盘管理分散到多个 I/O 处理结点上,并将它们作为一个逻辑上的大磁盘来处理,逻辑文件存储块交叉地分布在不同结点所控制的磁盘上,以实现多个结点对同一文件的并行访问,在很大程度上解决 I/O 瓶颈.

与单一文件系统一样,引入并行文件系统后,使用被称为"disk cache"的内存缓冲区技术同样会非常显著地提高系统性能.[3]在以往的并行文件系统研制中,由于主要是面向科学计算,它们的文件处理模式绝大多数是对一个大文件的顺序读写[3],故其 disk cache 技术均采用单副本[3],即每个 I/O 结点上的 disk cache 中仅有存放本结点所带磁盘的盘块内容的缓冲区.但是科学计算毕竟是计算机应用中的一小部分,从并行机的通用性来考虑,其文件处理模式将显示出很大的随机性.例如在应用中占很大比例的数据库,其规模正日益庞大,作为底层支持的并行文件系统将会在速度上为其带来很大益处.但由于各结点上的事务处理所针对的文件块访问是完全由日常需要而定的随机访问,那么在 disk cache 单副本中,当多个结点同时对同一 I/O 结点所控制的盘块进行访问时,此 I/O 结点处又将发生瓶颈.为此引入 disk cache 多副本技术,即各 I/O 结点的 disk cache 中可以存放其它结点所带磁盘的盘块内容,使这种瓶颈得以很大解决.

---

* 作者武北虹,女,1970年生,博士,讲师,主要研究领域为并行操作系统.邢汉承,1938年生,教授,主要研究领域为系统软件,人工智能,图象处理,模式识别.黄大海,1953年生,副教授,主要研究领域为并行处理,分布式系统.

本文通讯联系人:武北虹,厦门 361005,厦门大学计算机科学系

本文1994-12-06收到修改稿

当然,引入 disk cache 的多副本后,其一致性维护问题要远比单机复杂得多.因为同一盘块的映象可能存在于不同处理机的 disk cache 中,这样不仅要维护缓冲区与盘块间的一致性,还要维护这些映象副本间的一致性,本文将提出 2 类一致性维护方法,并对其正确性加以证明.

1 一致性维护协议

为研究方便,将并行计算机系统中用于并行文件系统的部分抽象成 n 个 I/O 结点通过互连网络进行通讯的消息传递型结构.每个 I/O 结点上有一个专门用于监视总线命令并进行处理的监视进程.一个逻辑上的文件以盘块为单位交叉地分布在不同结点所带磁盘上,每一结点上均有一套 mapping 技术,可知一个 I/O 请求的文件块对应哪一盘块,称此盘块所在结点为该块的本地结点,将有 I/O 请求的结点称为其请求结点,disk cache 中有其映象的结点称为该块的副本结点,各块的副本信息仅存在于本地结点上.

关于并行文件系统中 disk cache 一致性问题,国内外很少有文献谈到.基于硬件 cache 中一些思想的启发[4~6],并通过对 disk cache 具体特点的分析,我们提出"主从式"及"对称式"2 类一致性维护协议,限于篇幅,本文仅对此简述如下,详细内容见另文.在"读命中"、"读失效"、"写命中"及"写失效"4 种事件中,监视进程对"读命中"的处理较简单,只要将命中的数据直接返回,而对"写失效"的处理是由对"读失效"和"写命中"的处理联合完成的,故下面仅对"读失效"和"写命中"进行叙述,淘汰算法仍采用精确 LRU 算法.[7]假设各协议中发送的请求为双向通迅,即要求对方收到该请求后回送一个回答信息.

1.1 "主从式"维护方法

这类方法的特点是以本地结点上的副本为"主",其余各副本的存在均依赖于本地结点上的副本.当请求结点失效时其所需数据均由本地结点提供,若本地结点上的副本被淘汰,则其它结点上的副本不能继续存在,根据当一结点在执行写操作时,对其它副本的不同处理方法,此类方法又可分为 WIOC(write-invalid-other-copies)及 WTAC(write-through-all-copies)2 种,前者使其它副本全部失效而后者将其同时写入所有副本中.

(1)WIOC 协议

· 位于本地结点及非本地结点的 disk cache 块的状态转换图分别见图 1(a),(b).

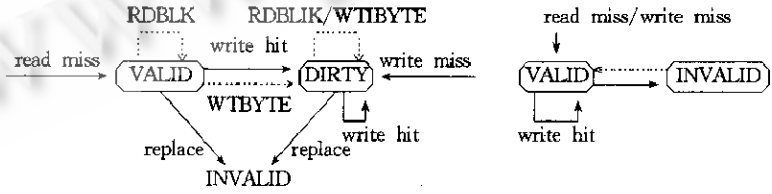


图1

· 读失效:若请求结点不是本地结点,请求结点首先向本地结点发 RDBLK 请求,本地结点先检查该块是否存在于其 disk cache 中,若不存在,则从磁盘中读入 disk cache,并将状态置为 VALID,然后向请求结点回送该块内容;请求结点接到后,放入其 disk cache,将状态置为 VALID 后,读取所需字节.若请求结点即为本地结点,则直接从磁盘中读入 disk

cache,将状态置为 VALID 后,读取所需字节.

· 写命中:若请求结点不是本地结点,请求结点首先向本地结点发 WTBYTE 请求,然后进行写操作,本地结点接到 WTBYTE 请求后,先检查是否还有其它副本,若有,则向这些副本结点发INVALID命令,然后进行相应写操作并将状态置为 DIRTY. 副本结点接到 INVALID命令后,将相应 disk cache 块的状态变为 INVALID.

(2)WTAC 协议

· 对"读失效"的处理与 WIOC 相同.

· 对"写命中"的处理与 WIOC 的不同仅在于向本副本结点发的不是 INVALID 命令,而是 WTBYTE,副本结点接到后进行相应写操作.

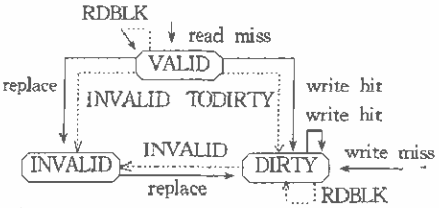
· 位于本地结点的 disk cache 块的状态转换图与 WIOC 相同;位于非本地结点的 disk cache 块状态转换图与 WIOC 的差别仅在于在 VALID 的状态时可以接收 WTBYTE 命令,且不改变状态.

1.2 "对称式"维护方法

与前类方法相比,这类方法的特点体现了各副本间的相对对称性关系. 当请求结点失效时,其所需数据可由任一有此块副本的结点提供,当本地结点上的副本被淘汰时,其它结点上的副本仍能独立存在. 同样,这类方法也有 WIOC 及 WTAC 2 种.

(1)WIOC 协议

· 位于本地结点及非本地结点上的 disk cache 块的状态转换图相同,如图 2 所示.



注:TODIRTY为淘汰算法选中一状态为DIRTY的 disk cache块时,通知某结点将其相应 disk cache块状态变为DIRTY的命令.

图2

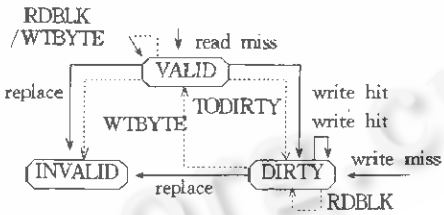


图3

· 读失效:若请求结点不是本地结点,请求结点首先向本地结点发 RDBLK 请求,本地结点先检查其 disk cache,若有,则将其内容回送请求结点,若没有,则继续检查是否有其它副本,若有,则向任一副本结点发 RDBLK 请求,否则从磁盘中读入 disk cache,将状态置为 VALID 并回送请求结点. 副本结点收到 RDBLK 后,直接将相应 disk cache 块的内容回送请求结点. 若请求结点即为本地结点,则首先检查是否有其它副本,若有,则向任一副本结点发RDBLK 请求,副本结点将其内容回送请求结点;否则从磁盘中读入 disk cache 并将状态置为 VALID.

· 写命中:若请求结点不是本地结点,请求结点首先向本地结点发 INVALID 命令,然后进行写操作,并将相应 disk cache 块的状态变为 DIRTY. 本地结点接到 INVALID 后,将其disk cache 中相应块状态变为 INVALID(如果有),然后向所有其它副本(如果有)结点发 INVALID 命令. 若请求结点即为本地结点,则先向所有其它副本(如果有)结点发INVALID 命令,然后进行相应写操作,并将其状态变为 DIRTY. 副本结点接到 INVALID 命令后,将

相应 disk cache 块变为 INVALID 状态.

(2)WTAC 协议

· 对"读失效"的处理与 WIOC 相同.

· 对"写命中"的处理与 WIOC 的不同仅在于所有 INVALID 命令均为 WTBYTE 命令,相应结点接到后,将要写的内容写入相应 disk cache 块中,并将其状态置为 VALID.

· 位于本地结点及非本地结点上的 disk cache 块的状态转换图相同,如图 3 所示,从此也可看出与"主从式"相比所具有的对称性.

2 一致性模型选取

在多机系统特别是共享内存的研究中,曾提出过许多一致性模型,如顺序(Sequential)一致性模型、处理机(Processor)一致性模型、弱(Weak)一致性模型、释放(Release)一致性模型等.[8]这些模型对一致性协议的约束条件按上述顺序逐步减弱,但其效率逐步增高,而且仍然为用户提供了合理的程序设计模型. 此处选取 release 一致性模型,其有关定义如下:

定义 2.1. 设 P_i 为处理机 i,对于 P_i 上的读操作来说,如果在某一时刻 P_k 上对同一地址的写操作不影响该读操作的返回值,则称此读操作在该时刻关于 P_k 已完成. 对于 P_i 上的写操作来说,如果在某一时刻 P_k 上对同一地址的读操作返回该写操作或其后的写操作定义的值,则称此写操作在该时刻关于 P_k 已完成. 若某一操作(读或写)关于所有的处理机已完成,则称此操作已完成.

定义 2.2. 如果读操作已完成,并且定义该读操作返回值的写操作也已完成,则称此读操作全局已完成.

定义 2.3. 在一并行系统中,若任一执行的结果与在某一顺序执行中运行所有处理机的操作的结果相同,并且此顺序执行中每个单处理机的操作以其程序指定的次序出现,则称此系统是顺序一致的.

在 release 一致性模型中,要为用户提供基于硬件支持的同步操作(acquire 和 release 操作),由用户来保证访问临界段时的互斥性,acquire 操作指的是获得访问临界段许可权的操作(如 *lock*,*P* 操作),而 release 操作指的是释放该许可权的操作(如 *unlock*,*V* 操作).

定义 2.4. 满足下列 3 个条件的系统称为是 release 一致的:

· 同步操作(acquire 及 release)是顺序一致的.

· 在一个普通读写操作关于任一其它处理机已完成之前,所有在其之前的 acquire 操作必须已完成.

· 在一个 release 操作关于任一其它处理机已完成之前,所有在其之前的普通读写操作必须已完成.

Gharachorloo 等已经证明[8]在用户程序中正确使用同步互斥控制的条件下,release 一致性与顺序一致性是等价的.

3 disk cache 一致性模型

为证明前述 disk cache 一致性协议的正确性,建立下列 disk cache 一致性模型,而且将

证明该模型是 release 一致的.

定义 3.1. 与一文件的逻辑块 LB 相对应的物理盘块以及 disk cache 块统称为该逻辑块的映象. LB 及映象 M 在 t 时刻的内容分别记为 LB/t 及 M/t. 将由逻辑块及其所有映象组成的多映象系统 S 作为我们的研究对象.

定义 3.2. 设 OP 为一算法 A 定义的操作,将 OP 按此定义运行完毕的时刻称为 OP 的结束时刻. 对一逻辑块 LB 的某一操作 OP 的结束时刻若先于 t,则称在 t 时刻对 LB 作用了操作 OP,此时 LB 的内容记为 $OP(LB)/t$.

定义 3.3. 设从 S 的初始状态的开始时刻 t_0 到 t 时刻对一逻辑块 LB 共作用了 n 次写操作 $W_1,\cdots,W_n$,如果此时其某一映象 M 的内容与其经过 n 次写操作后的结果相同,即: $M/t=W_n(\cdots(W_2(W_1(LB)/t)/t)/t$(此处相等指内容上相同,下同)则称 M 处于 t 时刻的一致性状态,记为 $C(M)/t$.

定义 3.4. 若在任一时刻 t 按照算法 A 对一逻辑块 LB 作用的读操作的结果与 LB 的处于 t 时刻的一致性状态的映象内容相同,即: $\forall t\exists M_i(C(M_i)/t \wedge Read(LB)/t \overset{A}{=} M_i/t)$ 则称 A 是可维护 S 的一致性的.

该一致性定义是满足定义 2.4 的 3 个条件的. 由 acquire 及 release 在操作系统中的实现及定义 2.3 可知,条件 1 显然成立;在某一结点进行普通读写操作时,说明该结点已进入临界段,故在其之前的该结点上的 acquire 已完成,而对于在其之前的其它结点上的 acquire,由基于硬件提供的同步机制而实现的 acquire 与普通读写操作之间的互斥可知,必须当 acquire 完成后释放该机制,普通读写操作才能开始,故条件 2 成立;在某一结点上的 release 完成前,只有该结点进入临界段,故此时的读写操作均在本结点上,因而由定义 3.2 知当开始 release 时,已作用了这些读写操作. 由定义 3.3、3.4 及定义 2.1 知这些读写操作已完成,故条件 3 成立. 因此该模型为 release 一致性模型.

实际上条件 3 中的读操作不仅已完成,而且是全局已完成的(定义 2.2),故该模型的约束条件比 release 一致性稍强.

4　disk cache 一致性协议正确性证明

公理 1. 在 S 的初始状态的开始时刻 t_0,任一逻辑块 LB 在 S 中有且仅有一个映象,即 LB 的磁盘映象 Md,且 $LB/t_0=Md/t_0$.

公理 2. 在 S 中,仅对逻辑块 LB 作用的写操作可以改变 LB 及 disk cache 映象的内容.

公理 3. 在 S 中,仅当完成"replace"事件(指某逻辑块被淘汰算法选中的事件)且需要回写磁盘时,才可以改变一逻辑块的磁盘映象的内容.

引理 1. 在"主从式"及"对称式"方法中,若 Md 为一逻辑块 LB 的磁盘映象,并且从 S 的初始状态的开始时刻 t_0 到 t 时刻对 LB 作用了 0 次写操作,那么 $C(Md)/t$.

该引理的证明较为简单,由公理 3、公理 1、公理 2 分别可知: $Md/t=Md/t_0=LB/t_0=LB/t$,由定义 3.3 即得: $C(Md)/t$. 其详细证明略.

引理 2. 在"主从式"方法中,若一逻辑块 LB 于 t 时刻在其本地结点上存在 disk cache 映象 Mo,那么 $C(Mo)/t$;否则对于 LB 的磁盘映象 Md 有: $C(Md)/t$.

证明：设在$[t_0,t]$内共对LB作用了n次写操作($n\geqslant 0$)，t_0为S处于初始状态的开始时刻，对n进行归纳证明如下：

①当$n=0$时，若LB于t时刻在其本地结点上存在disk cache映象Mo，则由“主从式”方法知，Mo必是由于在某一时刻$t_1(t_0<t_1\leqslant t)$完成了“read miss”事件而开始存在于disk cache中，直到t时刻未对Mo完成过“replace”事件，并且$Mo/t_1=Md/t_1$ (a)

又由引理1知：$C(Md)/t$，故$W_n(\cdots(W_2(W_1(LB)/t)/t)/t \overset{\text{定义3.3}}{=} Md/t \overset{\text{公理3}}{=} Md/t_1 \overset{\text{(a)}}{=} Mo/t_1 \overset{\text{公理2}}{=} Mo/t$，由定义3.3知$C(Mo)/t$. 若$t$时刻$Mo$不存在，由引理1直接可得$C(Md)/t$，即此命题成立.

②假设当$n=k$时此命题成立，设$t_1(t_0<t_1<t)$为第$k+1$次写操作的开始时刻，那么在t_1时刻对LB作用了前k次写操作，但尚未作用第$k+1$次，则在t_1时刻若Mo存在，则$C(Mo)/t_1$，否则$C(Md)/t_1$. 当$n=k+1$时，设$t_2(t_1<t_2\leqslant t)$为第$k+1$次写操作的结束时刻，由“主从式”方法知，不论此写操作的请求结点是哪一个，都将同时写入Mo中.

(i)若t_1时刻Mo存在，则由归纳假设：$C(Mo)/t_1$，即

$$Mo/t_1 \overset{\text{定义3.3}}{=} W_k(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1 \tag{b}$$

而$Mo/t_2 \overset{\text{“主从式”}}{\underset{\text{公理2}}{=}} W_{k+1}(Mo/t_1)/t_2 \overset{\text{(b)}}{=} W_{k+1}(W_k\cdots W_2(W_1(LB)/t_1)/t_1)t_1)/t_2 \overset{\text{定义3.2}}{=} W_{k+1}(\cdots(W_2(W_1(LB)/t_2)/t_2)/t_2$.

(ii)若t_1时刻Mo不存在，则由归纳假设：$C(Md)/t_1$，即

$$Md/t_1 \overset{\text{定义3.3}}{=} W_k(\cdots W_2(W_1(LB)/t_1)t_1/t_1 \tag{c}$$

经过第$k+1$次写操作后Mo存在，且$Mo/t_2 \overset{\text{“主从式”}}{\underset{\text{公理2}}{=}} W_{k+1}(Md/t_1)/t_2 \overset{\text{(c)}}{=} W_{k+1}(W_k(\cdots W_2(W_1(LB)/t_1)/t_1/t_1/t_2 \overset{\text{定义3.2}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_2)/t_2)/t_2$.

由(i)(ii)知：$$Mo/t_2=W_{k+1}(\cdots W_2(W_1(LB)/t_2)/t_2)/t_2 \tag{d}$$

根据假设知在$[t_2,t]$内对LB无写操作作用，对此分下列2种情形讨论：

(Ⅰ)若在$[t_2,t]$内对Mo无“replace”事件完成，则Mo一直存在到t时刻，且$Mo/t \overset{\text{公理2}}{=} Mo/t_2 \overset{\text{(d)}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_2)/t_2)/t_2 \overset{\text{定义3.2}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t)/t)/t$，由定义3.3即得$C(Mo)/t$.

(Ⅱ)若在$[t_2,t]$内对Mo有“replace”事件完成，并设$t_3(t_2<t_3\leqslant t)$为完成第1次“replace”事件的时刻，则由“主从式”方法知，在t_3时刻要将Mo的内容回写到磁盘，即

$$Md/t_3=Mo/t_2 \tag{e}$$

且Mo不再存在. 而在$[t_3,t]$内由于无写操作发生，当再次有“replace”完成时，LB的disk cache映象不可能为DIRTY状态，故不再需要回写磁盘，所以

$$\forall\, t_i(t_i\in[t_3,t]\rightarrow Md/t_i=Md/t) \tag{f}$$

若在t时刻Mo存在，由“主从式”方法知，Mo必是由于在某一时刻$t_4(t_3<t_4\leqslant t)$完成了“read miss”事件而开始存在，直到t时刻再未完成过“replace”事件，且$Mo/t_4=Md/t_4$ (g)

故$Mo/t \overset{\text{公理2}}{=} Mo/t_4 \overset{\text{(g)}}{=} Md/t_4 \overset{\text{(f)}}{=} Md/t \overset{\text{(f)}}{=} Md/t_3 \overset{\text{(e)}}{=} Mo/t_2 \overset{\text{(d)}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_2)/t_2)/t_2$

$\overset{\text{定义3.2}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t)/t)/t$,由定义3.3即得:$C(Mo)/t$.

若在t时刻Mo不存在,直接由(f)得:$Md/t \overset{(f)}{=} Md/t_3 \overset{(e)}{=} Mo/t_2 \overset{(d)}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_2)t_2)/t_2 \overset{\text{定义3.2}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t)/t)/t$,由定义3.3即得:$C(Md)/t$.

综合(Ⅰ)(Ⅱ)知,当$n=k+1$时,此命题亦成立.由①②,根据数学归纳法,此理得证.

引理3. 在"主从式"方法中,若一逻辑块LB于t时刻在其某一非本地结点Ns的disk cache中存在映象Ms,则$C(Ms)/t$.

证明:设在$[t_0,t]$内共对LB作用了n次写操作($n\geqslant 0$),t_0含义同上,如果LB于t时刻在Ns上存在disk cache映象Ms,由"主从式"方法知,Ms必是由于某一时刻$t_1(t_0<t_1\leqslant t)$完成了"read miss"或"write miss"事件而开始存在,直到t时刻再未完成过"replace"事件且未接到过INVALID命令,并且t_1时刻Mo存在,$Ms/t_1=Mo/t_1$　(a)

设在$[t_0,t_1]$内共对LB作用了$W_1,\cdots,W_k$,k次写操作,在$[t_1,t]$内共对LB作用了$W_{k+1},\cdots,W_n$,$n-k$次写操作.由引理2知,$C(Mo)/t_1$,即

$$Mo/t_1 \overset{\text{定义3.3}}{=} W_k(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1 \qquad \text{(b)}$$

由上述题设知,在WIOC中,$W_{k+1}\cdots W_n$这$n-k$次写操作的请求结点均为Ns,即对LB的这$n-k$次写操作要同时对Ms进行;而在WTAC中,每次写操作作用时,其内容也同时写入当时存在的所有disk cache映象中,由于Ms在$[t_1,t]$内一直存在,故$W_{k+1}\cdots W_n$也同时对Ms进行.所以不论"主从式"方法中哪一种,都有:$Ms/t=W_n(\cdots W_{k+1}(Ms/t_1)/t)/t \overset{(a)}{=} W_n(\cdots W_{k+1}(Mo/t_1)/t)/t \overset{(b)}{=} W_n(\cdots W_{k+1}(W_k(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1)/t)/t \overset{\text{定义3.2}}{=} W_n(\cdots W_2(W_1(LB)/t)/t)/t$,由定义3.3即可得$C(Ms)/t$.□

定理1. "主从式"方法是可维护S的一致性的.

证明:由"主从式"方法可知,在任一时刻t于请求结点Nr上作用的LB的读操作有下列4种情况,其中Mo,Mr,Md分别为LB在本地结点,请求结点上的disk cache映象(若有)以及磁盘映象.

(1). Nr为LB的本地结点且读命中,则$Read(LB)/t \overset{\text{"主从式"}}{=} Mo/t$

(2). Nr为LB的本地结点且读失效,则$Read(LB)/t \overset{\text{"主从式"}}{=} Md/t$

(3). Nr为LB的非本地结点且读命中,则$Read(LB)/t \overset{\text{"主从式"}}{=} Mr/t$

(4). Nr为LB的非本地结点且读失效,则若Mo存在$Read(LB)/t \overset{\text{"主从式"}}{=} Mo/t$,否则$Read(LB)/t \overset{\text{"主从式"}}{=} Md/t$

由引理2、引理3知上述各情况中$C(Mo)/t,C(Md)/t,C(Mr)/t$,故$\forall t\exists Mi(C(Mi)/t \wedge Read(LB)/t \overset{\text{"主从式"}}{=} Mi/t)$,由定义3.4此理得证.

定义3.5. 由"对称式"维护方法可知,当一结点发生LB的"read miss"事件时,其disk cache上便产生一映象Mi,其内容或是从此时系统内LB的另一disk cache映象Mj复制(如果有),或是从LB的磁盘映象Md复制(若没有disk cache映象),称Mj(或Md)为Mi的前继映象,记为$pred(Mi)$.

定义3.6. 设LB的一映象序列$M_1,M_2\cdots M_n$中,$M_2\cdots M_n$分别为产生于$t_2\cdots t_n$时刻的

disk cache 映象，M_1 为存在于 t_1 时刻的 disk cache 映象($t_1<t_2\cdots<t_n$)或 LB 在 t_2 时刻的磁盘映象，并且 $Mi=pred(M_{i+1})(i=1,2,\cdots,n-1)$，称 $M_1,M_2,\cdots,M_n$ 为 LB 的映象 M_n 在$[t_1,t_n]$上的映象产生轨迹.

引理 4. 在"对称式"方法中，设 t_0 为 S 处于初始状态的开始时刻，$M_1,M_2,\cdots,M_n$ 为 LB 的映象 M_n 在$[t_0,t_n]$上的映象产生轨迹，则 M_1 必为 LB 的磁盘映象.

证明：用反证法. 假设 M_1 不是 LB 的磁盘映象，则由定义 3.6 知 M_1 一定是 LB 的 disk cache 映象，且存在于 t_0 时刻，而由公理 1 知 LB 在 t_0 时仅有唯一映象，即 LB 的磁盘映象，这是矛盾的. □

引理 5. 在"对称式"方法中，若 $M_1,M_2,\cdots,M_n$ 为 LB 的映象 M_n 在$[t_1,t_n]$上的映象产生轨迹，且在$[t_1,t_n]$上对 LB 作用了 0 次写操作，其中 $t_n<t$，M_n 从产生一直存在到 t 时刻，则若 M_1 为 LB 的 disk cache 映象，有 $M_n/t=M_1/t_1$；若 M_1 为 LB 的磁盘映象，有 $M_n/t=M_1/t_2$(t_2 为 M_2 的产生时刻).

证明：设 $t_3,\cdots,t_{n-1}$分别为 $M_3,\cdots,M_{n-1}$的产生时刻，由定义 3.5、3.6 知 t_n 为 M_n 的产生时刻，$M_2,\cdots,M_n$ 均为 LB 的 disk cache 映象，且 $Mi(i=2,\cdots,n-1)$至少存在到 t_{i+1}时刻，所以：

$$M_n/t_n \overset{\text{定义3.6 3.5}}{=} M_{n-1}/t_n \overset{\text{公理2}}{=} M_{n-1}/t_{n-1} \overset{\text{定义3.6 3.5}}{=} M_{n-2}/t_{n-1} \overset{\text{公理2}}{=} M_{n-2}/t_{n-2}=\cdots \overset{\text{定义3.6 3.5}}{=} M_2/t_3 \overset{\text{公理2}}{=} M_2/t_2 \quad \text{(a)}$$

若 M_1 为 LB 的 disk cache 映象，则由定义 3.5、3.6 知，M_1 存在于 t_1 时刻，且至少存在到 t_2 时刻，故

$$M_n/t \overset{\text{公理2}}{=} M_n/t_n \overset{\text{(a)}}{=} M_2/t_2 \overset{\text{定义3.6 3.5}}{=} M_1/t_2 \overset{\text{公理2}}{=} M_1/t_1$$

若 M_1 为 LB 的磁盘映象，则由定义 3.5、3.6 知，M_2 在 t_2 时刻从 M_1 复制而产生，故

$$M_n/t \overset{\text{公理2}}{=} M_n/t_n \overset{\text{(a)}}{=} M_2/t_2 \overset{\text{定义3.6 3.5}}{=} M_1/t_2$$ □

引理 6. 在"对称式"方法中，若一逻辑块 LB 于 t 时刻在某一结点 Ni 的 disk cache 中存在映象 Mi，则 $C(Mi)/t$；若 LB 于 t 时刻无任何 disk cache 映象存在，则 $C(Md)/t$ (Md 为 LB 的磁盘映象).

证明：设在$[t_0,t]$内共对 LB 作用了 n 次写操作($n\geqslant 0$)，t_0 为 S 处于初始状态的开始时刻，对 n 进行归纳证明如下：

①当 $n=0$ 时，若 LB 于 t 时刻在 Ni 上存在 disk cache 映象 Mi，则由"对称式"方法知，Mi 必是由于在某一时刻 $t_n(t_0<t_n\leqslant t)$完成了"read miss"事件而开始存在直到 t 时刻未对 Mi 完成过"replace"事件，故 Mi 产生于 t_n 时刻. 考察 Mi 在$[t_0,t_n]$上的映象产生轨迹 M_{i1}，$M_{i2},\cdots,M_{in}$(即 Mi)，得：

$$Mi/t \overset{\text{引理5}}{=} M_{i1}/t_2 \overset{\text{引理4}}{=} Md/t_2 \quad \text{(a)}$$

其中 t_2 为 M_{i2}的产生时刻. 又由引理 1：$C(Md)/t$，故 $W_n(\cdots W_2(W_1(LB)/t)/t)/t \overset{\text{定义3.3}}{=} Md/t \overset{\text{公理3}}{=} Md/t_2 \overset{\text{(a)}}{=} Mi/t$，由定义 3.3 知，$C(Mi)/t$；若 t 时刻 LB 无任何 disk cache 映象存在，由引理 1 直接得 $C(Md)/t$. 即此命题成立.

②假设当 $n=k$ 时此命题成立，设 $T_1(t_0<T_1<t)$为第 $k+1$ 次写操作的开始时刻，那么

在T_1时刻对LB作用了前k次写操作,但尚未完成第$k+1$次,则在T_1时刻,对LB的任一disk cache映象Mx有:$C(Mx)/T_1$,且所有这些映象在T_1时内容相同(定义3.3),若无disk cache映象,则$C(Md)/T_1$. 当$n=k+1$时,设$t_1(T_1<t_1\leqslant t)$为第$k+1$次写操作的结束时刻,并设此写操作的请求结点为Np,Np上LB的disk cache映象为Mp.

(i)若在T_1时刻Mp存在,则根据归纳假设有

$$C(Mp)/T_1. \tag{b}$$

故 $Mp/t_1 \overset{\text{“对称式”}}{\underset{\text{公理2}}{=}} W_{k+1}(Mp/T_1)/t_1 \overset{\text{(b),定义3.3}}{=} W_{k+1}(W_k(\cdots W_2(W_1(LB)/T_1)T_1)/T_1)/t_1 \overset{\text{定义3.2}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1$.

(ii)若在T_1时刻Mp不存在,但有其它disk cache映象存在,则首先从这映象中复制,使Mp存在,再进行写操作,此时与(i)相同.

(iii)若在T_1时刻不存在任何disk cache映象,则根据归纳假设

$$C(Md)/T_1 \tag{c}$$

经过第$k+1$次写操作后,Mp存在,且 $Mp/t_1 \overset{\text{“对称式”}}{\underset{\text{公理2}}{=}} W_{k+1}(Md/T_1)/t_1 \overset{\text{(c),定义3.3}}{=} W_{k+1}(W_k(\cdots W_2(W_1(LB)/T_1)T_1)T_1)/t_1 \overset{\text{定义3.2}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1$

由(i)~(iii)知:$Mp/t_1=W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1$. 对于WIOC方法,当进行第$k+1$次写操作时,$LB$的其它disk cache副本全部被INVALID(不再存在);而在WTAC方法中,此写操作同时写入到其它所有副本中,故在“对称式”方法中,对t_1时刻存在的任一disk cache映象Mj,均有

$$Mj/t_1=Mp/t_1=W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1 \tag{d}$$

根据假设知在$[t_1,t]$内对LB无写操作发生,若LB于t时刻在结点Ni上的disk cache中存在映象Mi,见下列情形(Ⅰ)(Ⅱ)中的讨论;若LB于t时刻无任何disk cache映象存在,则见情形(Ⅲ)的讨论:

(Ⅰ)若Mi从t_1时刻一直存在到t时刻,则

$$Mi/t \overset{\text{公理2}}{=} Mi/t_1 \overset{\text{(d)}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1 \overset{\text{定义3.2}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t)/t)/t.$$

(Ⅱ)若Mi在t时刻存在,但不是从t_1时刻一直存在到t时刻,则Mi必是由于某一时刻$t_n(t_1<t_n\leqslant t)$在Ni上完成了“read miss”事件而产生,并直到t时刻未完成过replace事件,考察Mi在$[t_1,t_n]$上的映象产生轨迹$M_{i1},M_{i2},\cdots,M_{in}$(即$Mi$),设$t_2$为$M_{i2}$的产生时刻. 若$M_{i1}$为disk cache映象,由定义3.6知$M_{i1}$存在于$t_1$时刻,故 $Mi/t \overset{\text{引理5}}{=} M_{i1}/t_1 \overset{\text{(d)}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1 \overset{\text{定义3.2}}{=} W_{k+1}(W_2(W_1(LB)/t)/t)/t$;若$M_{i1}$为$LB$的磁盘映象$Md$,由定义3.6、定义3.5知,在$[t_1,t_n]$内$S$中曾出现过无$LB$的disk cache映象的时刻,假设第1次出现这种现象的开始时刻为$t_a(t_1<t_a<t_2)$,那么在t_a时S中LB当前最后一个disk cache映象M_L(其state=DIRTY)完成了“replace”事件,且

$$Md/t_a=M_L/t_a \tag{e}$$

设t_{a1}为M_L的产生时刻($t_{a1}<t_a$),若$t_{a1}>t_1$,设M_{L1}为M_L在$[t_1,t_{a1}]$上的映象产生轨迹中的第1个映象,则M_{L1}必为存在于t_1时刻的某一disk cache映象. 故 $M_L/t_a \overset{\text{引理5}}{=} M_{L1}/t_1 \overset{\text{(d)}}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1$;若$t_{a1}\leqslant t_1$,则$M_L$于$t_1$时刻存在,故 $M_L/t_a \overset{\text{公理2}}{=} M_L/t_1 \overset{\text{(d)}}{=} W_{k+1}(\cdots W_2$

$(W_1(LB)/t_1)/t_1)/t_1$. 所以无论 t_{a1} 与 t_1 的关系如何,均有

$$M_L/t_a=W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1 \quad (f)$$

故 $Mi/t \overset{引理5}{=} Md/t_2 \overset{公理3}{=} Md/t_a \overset{(e)}{=} M_L/t_a \overset{(f)}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1 \overset{定义3.2}{=} W_{k+1}(\cdots W_2(W_1(LB)/t)/t)/t$.

(Ⅲ)若在 t 时刻不存在任何 LB 的 disk cache 映象,与(Ⅱ)的分析类似可知在 $t_a(t_1<t_a\leqslant t)$ 时刻作用$[t_1,t]$上第 1 次写盘操作. 此时

$$Md/t \overset{公理3}{=} Md/t_a \overset{(e)}{=} M_L/t_a \overset{(f)}{=} W_{k+1}(\cdots W_2(W_1(LB)/t_1)/t_1)/t_1 \overset{定义3.2}{=} W_{k+1}(\cdots W_2(W_1(LB)/t)/t)/t.$$

由(Ⅰ)(Ⅱ)(Ⅲ)及定义 3.3 知,当 $n=k+1$ 时,若 LB 于 t 时刻在 Ni 上存在 disk cache 映象 Mi,则 $C(Mi)/t$;若在 t 时刻无任何 disk cache 映象存在,则 $C(Md)/t$. 由①②,根据数学归纳法此理得证.

定理 2. "对称式"方法可维护 S 一致性的.

证明:由"对称式"方法可知,在任一时刻 t 于请求结点上作用的 LB 的读操作有下列 2 种情况,其中 Mr,Mc,Md 分别为 LB 在请求结点、某一副本结点上的 disk cache 映象(若有)以及磁盘映象.

(1)读命中,有 $Read(LB)/t \overset{"对称式"}{=} Mr/t$;

(2)读失效,若有其它副本存在,则 $Read(LB)/t \overset{"对称式"}{=} Mc/t$;若无任何副本存在,则 $Read(LB)/t \overset{"对称式"}{=} Md/t$.

由引理 6 知上述各情况中 $C(Mr)/t,C(Mc)/t,C(Md)/t$,故 $\forall t\exists Mi(C(Mi)/t \wedge Read(LB)/t \overset{"对称式"}{=} Mi/t)$. 由定义 3.4,此理得证.

5 结束语

在 release 一致性模型中,"主从式"及"对称式"disk cache 一致性协议是可维护系统一致性的. 各 disk cache 多副本一致性协议的维护开销是以增加若干内存操作和通讯为代价的,但其带来的益处却是减少了对磁盘块的读写次数,而内存操作和并行计算机中的通迅时间与盘块的操作时间相比要少几个数量级. 今后准备在各协议的性能分析及实际测试上做进一步研究.

参考文献

1 Accetta *et al*. Mach: a new kernel foundation for UNIX development. In: Proc. USENIX Conf., 1986. 93~112.

2 Mullender. Amoeaba: a distributed operating system for 1990s. IEEE Computer, 1990, **23(5)**: 44~53.

3 Kotz D *et al*. Caching and writeback policies in parallel file systems. Journal of Parallel and Distributed Computing, 1993, **17(1&2)**: 140~145.

4 Archibald J *et al*. Cache coherence protocols: evaluation using a multiprocessor simulation model. ACM Trans. on Computer Systems, 1986, **4(4)**: 273~298.

5 Eggers S *et al*. Evaluating the performance of four snooping cache coherency protocols. In: Proc. 16th Int'l Symp. on Computer Architecture, 1989. 2~15.

6 Lenoski D *et al*. The directory-based cache coherence protocol for the DASH multiprocessor. In: Proc. 17th Int'l

Symp. on Computer Architecture, 1990. 148～159.

7 尤晋元. UNIX 操作系统教程. 西安:西北电讯工程学院出版社,1985.

8 Garachorloo K *et al*. Memory consistency and event ordering in scalable shared_memory multiprocessors. In: Proc. 17th Int'l Symp. on Computer Architecture, 1990. 15～26.

CORRECTNESS PROVEMENTS OF DISK CACHE COHERENCE PROTOCOLS IN PARALLEL FILE SYSTEM

WU Beihong　XING Hancheng　HUANG Dahai

(*Department of Computer Science and Engineering　Southeast University　Nanjing*　210018)

Abstract　Two kinds of disk cache coherence protocols in parallel file system are proposed in this paper, and their correctness provements are given on the base of release consistency model.

Key words　Parallel file system, disk cache, coherence protocol, sequential consistency, release consistency.