# 优化记录法在 PVM 中的改进和应用*

黄 宁 金茂忠

(北京航空航天大学软件工程研究所 北京 100083)

**摘要** 并行程序的不确定性使其调试时不能象串行程序调试那样通过反复执行程序来查找错误,因此,记录并重演是并行调试的关键技术,而这一技术的最大开销就是事件的记录. 本文结合PVM(parallel virtual machine)的特点,改进了 Netzer 等人提出的优化记录法,使事件记录在出现非传递的消息竞争时也是最优的,并保证同一进程发送消息时,不会象原算法一样可能进行记录,同时,还在 PVM 并行调试环境中实现了该方法.

**关键词** PVM(parallel virtual machine),记录,偏序,重演,消息竞争,传递竞争,非传递竞争.

**中图法分类号** TP311

并行程序的不确定性使事件的记录与重演成为并行调试的关键技术. 在消息传递模型的环境中,由于再次执行时系统环境的不同,消息到达的先后次序和接收顺序可能与原执行不一致,从而导致程序的执行结果不同,致使程序调试中有可能无法再现前一次执行的错误,无法用串行调试常采用的反复执行的手段来进行调试. 因此,如何保证程序的再次执行重现原执行的情况是并行调试中首先要解决的问题. 这一方法通常称为记录并重演方法,方法的实现包括记录和重演两步. 在调试运行时间长或通讯量大的并行程序时,常常会产生相当大的记录文件,致使系统崩溃. 因此,记录方法是这一技术是否实用的关键.

要保证事件的顺序,必须知道事件发生的时间,但由于分布式环境没有一个统一的全局时钟,因此判断消息的先后次序需要专门的机制. 早期的实现采用观察者的方法,以观察者观察到的先后次序作为消息的先后次序,不能真实地反映消息的关系. 直到 1978 年,Lamport 提出了事件间"Happened Before"的偏序关系及逻辑时钟的概念[1],之后的方法大都采用了这一偏序关系描述并记录事件间的关系. 其中最有名的方法有 3 种,即 Lamport Clocks,Interval Clocks 和 Vector Time,其中 Vector Time 因其准确和易实现的特性得到了较为广泛的应用.

为保证消息的内容也与原执行一致,初期的记录方法不但记录下消息的顺序,还记录了消息的内容,这种方法无疑加大了记录文件. 对并行程序的进一步了解,使人们发现,记录所有消息的内容不太现实,也没有必要,LeBlanc 和 Mellor-Crummey 于是提出了 Instant

* 本文研究得到航空基金资助. 作者黄宁,女,1968 年生,在读博士生,主要研究领域为并行处理,软件工程. 金茂忠,1941 年生,教授,博士导师,计算机系主任,主要研究领域为软件工具,软件测试,并行处理.

本文通讯联系人:黄宁,北京 100083,北京航空航天大学计算机系

本文 1996-08-14 收到修改稿

Replay方法[2],认为只要记录下所有消息事件的顺序,并保证重新执行时的事件顺序与记录顺序一致,则程序便会产生正确的结果和中间的消息内容,而不用再对消息内容作记录.之后,人们又发现,并行程序之所以引入了事件记录是因为并行程序的不确定性,而不确定性的引发仅仅是由于消息的竞争,这几个竞争消息通常又仅占1%的比例,针对此特点,Netzer等人提出了"Optimal Tracing"的方法,动态判断当前接收的消息是否与以前的消息竞争,如果有竞争才记录该消息,这一方法减小了事件记录的文件大小,且在仅出现传递竞争的程序中是最优的.

## 1　优化记录法简介

Netzer提出的优化方法基于消息竞争的定义,当2个消息竞争时,记录其中的一个,反之则不进行记录.重演时只要保证竞争消息安全到达原执行时接收该消息的地方,另一个消息自然无处可去,只能正确到达.

### 1.1　基本概念及定义

在描述Netzer的算法和改进算法之前,我们先介绍几个概念.

$a \xrightarrow{M} b$ 表示有消息从事件 $a$ 到事件 $b$,$a \xrightarrow{XO} b$ 表示事件 $a$,$b$ 属于同一个进程,且事件 $a$ 在事件 $b$ 之前.$P=\langle E, \xrightarrow{HB} \rangle$ 表示并行程序的一次执行,其中 $E$ 表示程序执行中的所有事件集合,$\xrightarrow{HB}$ 表示事件间Happened Before的关系.

**定义**1. Happened Before关系是并行系统中事件间的一种偏序关系,当事件 $a$,$b$ 有Happened Before关系时,记作 $a \xrightarrow{HB} b$.它满足以下条件:

(1) 如果 $a \xrightarrow{XO} b$,那么 $a \xrightarrow{HB} b$.

(2) 如果 $a \xrightarrow{M} b$,那么 $a \xrightarrow{HB} b$.

(3) 如果 $a \xrightarrow{HB} b$ 并且 $b \xrightarrow{HB} c$,那么 $a \xrightarrow{HB} c$.

对并行程序的某次执行 $P$,我们可考虑这样的一个时刻(称之为边界),它横穿执行 $P$ 的所有进程,把 $P$ 分为两段,对此边界而言,存在这样一个执行集合Fsame,这些执行在边界前与 $P$ 的事件和消息传递顺序一致,而在边界后消息传递的顺序与 $P$ 不同.

**定义**2. Fsame是程序运行的所有可能运行,$P'=\langle E', \xrightarrow{HB'} \rangle$ 是Fsame中满足以下条件的一次程序执行:

(1) $P'$ 可在程序的实际运行中产生.

(2) 在某一边界之后,$P'$ 的事件仍与 $P$ 相同,定义如下:

对每个进程 $p$,$E'_p$ 集合包括:

(a) $E_p$ 的前半部分(即边界前的事件)

(b) 其余部分(即定义为边界后的事件)

(3) 在边界前,$P'$ 的消息传递与 $P$ 相同,即对所有 $a,b \in E'$,且 $a,b$ 在边界前,有 $a \xrightarrow{M} b$

$\Leftrightarrow a \xrightarrow{M'} b$.

**定义** 3. $a \xrightarrow{M} b$ 与 $c \xrightarrow{M} d$ 竞争，当且仅当程序的一次执行 $P'=\langle E', \xrightarrow{HB'}\rangle$，存在于 Fsame 中，使 $a,b,c \in E'$，且 $c \xrightarrow{M'} b (a \neq c)$.

### 1.2 方法描述

Netzer 认为仅有当两个消息竞争时才有记录的必要，判断算法很简单，即往前回溯找到能够接收当前已接收的 Msg 的前一个接收 PrevRecv，判断是否有 PrevRecv $\xrightarrow{HB}$ Send，如果没有，则表明 PrevRecv 和 Recv 竞争，记录下 Send $\xrightarrow{HB}$ Recv 事件. 反之，则不记录.

如图 1(a)所示的执行，由于满足 PrevRecv $\xrightarrow{HB}$ Send 的条件，2 个消息可能会在不同的执行中颠倒接收顺序，故 Netzer 的算法将记录下 Send $\xrightarrow{M}$ Recv，反之，在图 1(b)中，由于满足 PrevRecv $\xrightarrow{HB}$ Send 的条件，第 2 个消息一定要等第 1 个消息已接收后才能发送，因而也就不会有竞争存在，不用进行记录.

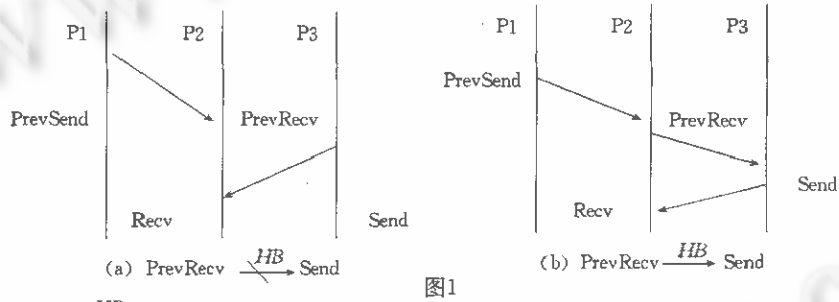


图1

(以下用 a $\xrightarrow{HB}$ b 表示 a，b 两事件间不存在 Happened Before 的关系)

## 2 优化记录法的改进

针对 Netzer 所提的优化算法的不足和 PVM 环境的特点，本文提出了一种改进算法，使事件记录在出现非传递的消息竞争时也是最优的，并保证同一进程发送消息时，不会象原算法一样可能进行记录.

### 2.1 PVM 中消息接收函数的分析

PVM 中消息接收函数要求指明发送消息的进程号，以及消息标志，例如最常用的是函数 pvm_recv(int tid,int msgtag)，表示从进程号为 tid 的进程那里接收消息标志为 msgtag 的消息.

当 tid=−1 时，pvm_recv 可从任意进程接收消息，msgtag=−1 表示可接任意标志的消息，因此 pvm_recv(−1,−1)可和任一次接收竞争，而当 tid 和 msgtag 均大于零时就仅会和 pvm_recv(−1,−1)有竞争. tid 号相同的接收不会产生竞争，因为 PVM 的机制保证了当进程 A 向进程 B 发送消息 Msg1 和 Msg2 时，进程 B 收到的消息顺序一定是先 Msg1 后 Msg2，如果利用 Netzer 的算法，此时 PrevRecv $\xrightarrow{HB}$ Send，则该记录这两次发送之一，但实

际上,在PVM中,不需要对此类情形作记录,而此类情况满足的条件是 PrevSend $\xrightarrow{XO}$ Send.

### 2.2 优化方法的不足

Netzer提出的优化记录法在出现传递竞争时是最优的,如图2所示,此时不但第1个消息与第2个消息竞争,第2个消息与第3个消息竞争,而且第3个消息还与第1个消息竞争.要保证重演的正确,必须同时记录第2和第3个消息.图3所示的是非传递竞争,此时虽然第1与第2、第2与第3个消息仍然有竞争,但第3与第1个消息却不竞争,如果用Netzer的算法,仍要记录第2与第3个消息,但实际上,只要记录了第2个消息就能保证程序的重演的正确.因此,Netzer的优化记录法在出现非传递竞争时不是最优的.

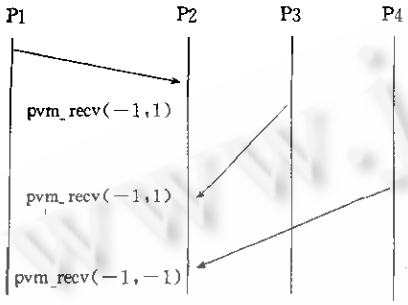


图2 传递竞争

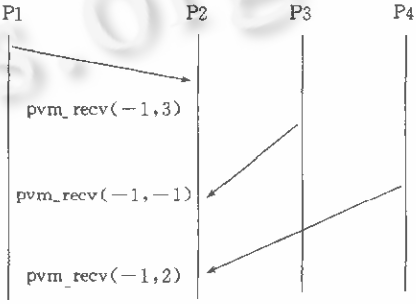


图3 非传递竞争

### 2.3 算法的改进

针对PVM的特点和非传递竞争的判定,本文提出了改进方案,以下是改进后的算法.

接收消息Msg;

(1) Send=发送该Msg的事件;

(2) PrevRecv=上一个能够接该Msg的接收事件(同一进程中);

(3) While (PrevRecv不空)

(4)　{PrevSend=向PrevRecv发送消息的事件;

(5)　if(PrevRecv $\xrightarrow{HB}$ Send) $\wedge$ (PrevSend $\xrightarrow{XO}$ Send) $\wedge$ (PreSend $\xrightarrow{M}$ PrevRecv 没记录过)

(6)　then { 记录 Send $\xrightarrow{M}$ Recv;break;}

(7)　PrevRecv=再往前一个能接收Msg的事件;}

### 2.4 证　明

首先,当算法中的条件(PrevRecv $\xrightarrow{HB}$ Send)和(PrevSend $\xrightarrow{XO}$ Send)得以满足,并且Recv的msgtag=−1时,演化为Netzer算法中传递竞争的条件,即图2所示的例子,后2个事件都应记录下来.

而当如图3所示的例子,出现非传递竞争时,该算法的第5行保证不会再记录pvm_recv(−1,2).因为pvm_recv(−1,−1)的记录已能保证消息的正确传递,第3行While语句的循环保证当前的Recv事件和再以前的事件竞争时记录下当前的事件,如图4所示,此

时改进算法能够保证第4个事件即pvm_recv(−1,1)仍能正确地记录下来,因为它与第1个事件即pvm_recv(−1.1)竞争.

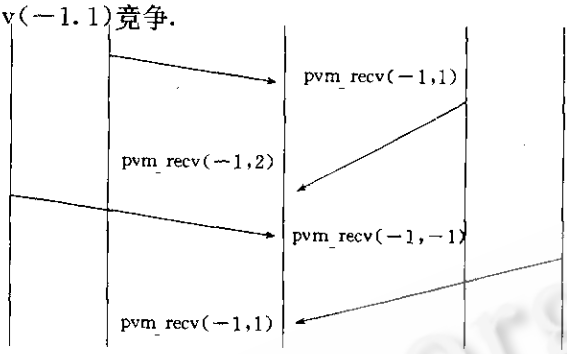


图4

其次,第2.1节介绍了PVM能够保证同一进程发送的消息的先后次序,故算法中采用了(PrevRecv $\xrightarrow{HB}$ Send) ∧ (PrevSend $\xrightarrow{XO}$ Send)的判断条件,用以保证当同一进程发送消息时不作记录,如图5所示.

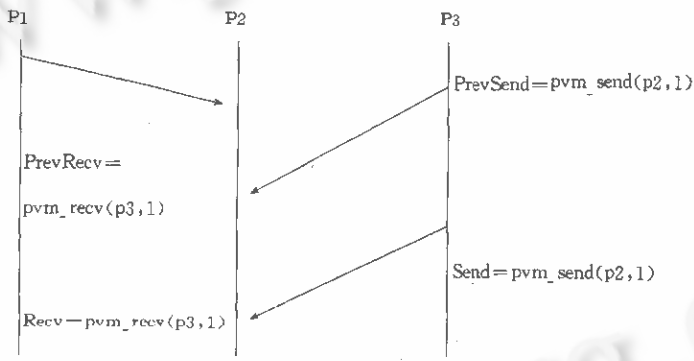


图5

根据原算法,由于此时(PrevRecv $\xrightarrow{HB}$ Send),应记录下Send $\xrightarrow{HB}$ Recv,但在PVM中则没有竞争,一定是第1个发送的消息第1个接收.

**引理1.** PVM中,当Send $\xrightarrow{M}$ Recv和PSend $\xrightarrow{M}$ PRecv满足条件:(PRecv $\xrightarrow{HB}$ Send) ∧ (PSend $\xrightarrow{XO}$ Send) ∧ (PSend $\xrightarrow{M}$ PRecv没记录过) ∧ (PRecv $\xrightarrow{XO}$ Recv) ∧ (SEND(Send) ∩ RECEIVE(PRecv) ≠ ∅)时,出现竞争.

证明:用反证法可证(略).Netzer为此已给出证明.由于PVM中保证PSend $\xrightarrow{XO}$ Send时不会有竞争,故所加内容也即可得证.

**引理2.** PVM中,当Send $\xrightarrow{M}$ Recv和PSend $\xrightarrow{M}$ PRecv竞争时,一定有(PRecv $\xrightarrow{HB}$ Send) ∧ (PSend $\xrightarrow{XO}$ Send).

证明:同引理1(略).

当出现传递竞争时,算法演化为Netzer的优化算法,需要记录下两两竞争的消息之一,

而当出现非传递竞争时,如果 PSend $\xrightarrow{M}$ PRecv 已记录,而且 Send $\xrightarrow{M}$ Recv 不再与其它消息竞争时,就不再记录 Send $\xrightarrow{M}$ Recv.

**竞争重演定理**:如果消息传递环境中并行程序的第 X 次执行时 PSend $\xrightarrow{M}$ PRecv 和 Send $\xrightarrow{M}$ Recv 竞争,那么如果在该并行程序的第 Y 次执行时保证 Send $\xrightarrow{M}$ Recv 的正确接收,即可保证这两次执行的相等.(证明略)

**引理** 3. PVM 应用程序中,改进算法必能保证重演的正确.

证明:用反证法.假设采用改进算法完成并行程序 P 记录阶段的执行,并在重演阶段保证了所记录的消息顺序后,该并行程序仍然有一个消息发送和接收事件 Send $\xrightarrow{M}$ Recv 使 P 出现了竞争.

设 Race_E 为所有能接收 Recv 所接收的 Msg 的消息发送和接收事件的集合,则由假设和引理 2 可推出∃(PSend $\xrightarrow{M}$ PRecv) ∈ Race_E 使(PRecv $\xrightarrow{HB}$ Send) ∧ (PSend $\xrightarrow{XO}$ Send)的条件成立.

进一步我们可证明 Race_E 中的事件必然均是记录过的,否则与假设矛盾.而据竞争重演定理,此时 P 在重演阶段的执行必是确定的,不会出现竞争,从而导致与假设的矛盾,故引理 3 得证.

**优化定理**. 如 PPSend $\xrightarrow{M}$ PPRecv, PSend $\xrightarrow{M}$ PRecv 和 Send $\xrightarrow{M}$ Recv 有非传递竞争,则算法仅记录 PSend $\xrightarrow{M}$ PRecv(其中 PPSend $\xrightarrow{M}$ PPRecv 为同一进程中再前一个可接收 Msg 的事件).

证明: Netzer 的证明已表明如果这 3 个事件是传递竞争,则其算法记录 PSend $\xrightarrow{M}$ PRecv 和 Send $\xrightarrow{M}$ Recv. 现假设非传递竞争的条件得以满足,但改进算法仍记录了 2 个事件,则算法在接收 Send $\xrightarrow{M}$ Recv 时必满足(PRecv $\xrightarrow{HB}$ Send) ∧ (PSend $\xrightarrow{XO}$ Send) ∧ (PSend $\xrightarrow{M}$ PRecv 没记录过)方能记录,从而推出 PSend $\xrightarrow{M}$ PRecv 没记录过,这与假设"已记录了两个事件"矛盾,故当出现传递竞争时改进算法仅记录一个.

以上的引理 3 证明了改进算法的正确性,而优化定理证明了改进算法在传递竞争和非传递竞争中的最优.

## 3 重　演

改进算法中有关 Happened Before 的关系判断均采用了 Vector Time 的机制,重演时仍维护这样的一个时间向量,由于不是每个消息都记录,故不能保证每一消息的发送顺序.但因为不竞争的消息在重复执行中必然产生相同的结果,故而仅保证竞争消息的正确发送与接收即可保证重演的正确.

重演时首先要对记录文件进行整理,使每一个被记录的消息在发送时都发送自己的向

量时钟，而PVM接收函数则更改为先判断是否为被记录的消息，如是，则接收应该接收的消息(向量时钟与记录文件的相符)，否则挂起，如不是，则不接收带向量进钟一起发送的消息.

## 4 结束语

由于重演已成为解决并行调试中不确定性的重要方法，因此良好的事件记录算法在并行程序的调试中有其重要意义. 改进算法针对PVM的特点和原算法的缺点作了改进，使非传递性竞争出现时算法也是最优化地记录事件，并且减少了由于 PrevSend $\xrightarrow{XO}$ Send 的条件得以满足而在原算法中引发的记录. 在PVM的并行调试器中，作者用此算法实现了并行程序的重演，由于该算法在PVM的实现中有更好、更优化的效果，而这种优化相应地减少了对记录文件的读写次数，从而减少了对被调试程序的入侵影响. 在调试运行时间长或通讯量大的程序时，这一算法的优点更加突出.

## 参考文献


1 Leslie Lamport. Time, clocks and the ordering of events in a distributed system. Communications of the ACM, 1978, **21**(7): 558～565.

2 LeBlanc T J, Mellor-Crummey J M. Debugging parallel programs with instant replay. IEEE, Transactions on Computers, 1987, **36**(4): 471～482.

3 Netzer R H B, Miller B P. Optimal tracing and replay for debugging message-passing parallel programs. Proceedings of Supprcomputing'92, Minneapolis, MN, Nov, 1992. 502～511.


# THE IMPROVEMENT AND APPLICATION OF OPTIMAL TRACING IN PVM


HUANG Ning JIN Maozhong

*(Institute of Software Engineering Beijing University of Aeronautic and Astronautics Beijing* 100083)



**Abstract** Parallel programs can not be debugged as sequential ones repeatly because of their nondetermination, so tracing and replay become the key technique of parallel debugging, and the most overhead of this technique is tracing the events. This paper improves the optimal tracing method provided by Rober Netzer etc. by combining the characteristics of PVM(parallel virtual machine), makes the events tracing optimal even when non_transitive message race appears. The improved method has been implemented in PVM_based parallel clebugger.



**Key words** PVM, tracing, partial order, replay, message race, transitive race, non_transitive race.

**Class number** TP311