文章编号 1672-6634(2018)04-0104-07 **DOI** 10.19728/j.issn1672-6634.2018.04.018

基于占优关系的MPI并行程序死锁检测①

田 甜 郭 庆 张 辰

(山东建筑大学 计算机科学与技术学院,山东 济南 250101)

摘 要 随着高性能技术的发展,MPI并行程序得到了越来越广泛的应用,其可靠性也得到了软件工程界越来越多的关注.一个MPI并行程序有多个并行执行的进程,每个进程包含一系列串行执行语句.死锁是并行程序的一类典型错误,本文提出一种基于占优关系的死锁检测方法.首先,根据通信语句之间的占优关系,得到每个进程内,通信边之间的占优关系;然后,通过判定通信边占优关系是否相互违背,检测程序中的死锁.基于上述思想,开发了基于占优关系的死锁检测原型工具,用于检测多个典型的MPI并行程序.结果表明,本文方法能够检测出程序中的死锁缺陷.

关键词 并行程序;占优关系;死锁;通信边

中图分类号 TP311 **文献标识码** A

0 引言

高性能计算在地震监测、石油勘探、生物医药,以及电力传输网络等领域发挥着重要作用.随着待处理问题的复杂度不断增加,以及对求解结果的精度要求日益提高,问题求解所需要的数据量和计算量不断增加,高性能计算的需求迅速增长[1].与此同时,高性能计算技术引起广泛关注.作为高性能计算系统的重要组成部分,分布式并行程序的可靠性,越来越得到人们的重视.

这里,分布式并行程序是指基于分布式存储架构的多进程并行程序.这些进程并行执行,通过消息传递的方式相互协作,共同完成求解任务.一种主流的并行程序开发方式是,基于消息传递接口(Message Passing Interface,MPI)[2]并行编程标准,扩展传统的串行程序,称这样的程序为MPI并行程序.MPI并行编程标准提供了丰富的并行程序接口,用于进程之间的消息传递,从而实现多进程的相互协作.然而,MPI并没有为消息传递函数的正确使用提供充分的保障[3],导致在并行程序中,经常出现不匹配的发送接收数据类型或长度,以及死锁等故障.

本文针对并行程序中典型的死锁故障,提出基于占优关系的死锁检测方法.称程序中调用通信函数实现消息传递的语句为通信语句;进一步,我们用通信边表示进程之间的消息传递.一条通信边连接来自不同进程的两个通信语句,通信边的起点和终点分别代表了消息的发送端和接收端.根据每个进程内部通信语句的占优关系,推导出通信边之间的占优关系;检查每个进程的通信边占优关系,如果存在与它相互违背的的占优关系,说明程序中可能存在死锁.

本文后续部分做如下安排.第1部分介绍与本文相关的工作;第2部分阐述本文所提方法,包括通信语句占优关系和通信边占优关系的判定,所提方法流程等;第3部分采用3个实例说明本文方法的应用过程和正确性;第4部分是本文方法的实现,并展示3个并行程序的检测结果;第5部分总结全文,提出进一步的研究方向.

① 收稿日期:2018-08-20
基金项目:国家自然科学基金项目(61503220)资助
通讯作者:田甜,女,汉,博士,副教授,研究方向:并行软件测试,E-mail:tiantian@sdjzu.edu.cn.

1　相关工作

MPISE 基于符号执行方法,调度各个进程,当没有合理进程可以调度,但是仍然有进程没结束时,说明程序中存在死锁[4]. 符号执行的基本思想是:使用符号值而不是具体值作为输入值,基于这些符号值跟踪数值操作的结果,属于静态方法. EMPDD 也是一种静态死锁检测技术,包括进程匹配、操作匹配和死锁检测三个算法,基于可能的匹配进程,以及可能匹配的发送和接收操作,检测死锁[5]. 除了静态检测方法以外,学者们还提出了运行时检测方法. Luecke 等开发了检测工具 MPI-Check,实现了一种基于延时的方法,用于检查 Fortran 90 和 Fortran 77 开发的消息传递程序的错误,例如,数组越界、不匹配的参数类型、消息长度为负数、非法 MPI 调度,和死锁等[6]. 这种基于延时的方法可能会导致误报. MPICH Extension 是一个扩展的 MPI 库,用于检测 MPI 集合操作的使用错误,但是忽略了死锁的检测[7]. ISP 是一种基于模型的检测方法,使用一个中央调度器,多次运行被测程序,检查所有可能的进程交互. 尽管这种方法实现了进程交互的完全覆盖,但是,随着交互数量呈指数级增长,这种方法的实际应用将受到很大限制[8]. DAMPI 取消了 ISP 的集中调度,使用分布式检测方法克服了 ISP 的缺陷[9]. Umpire[10] 使用 AND ⊕ OR 模型[11],实现了基于图形的死锁检测. MUST 对 Umpire 没有考虑的死锁情况进行了补充,并针对工具的可扩展性和效率问题进行了改进[12,13].

这些方法大都是从通信操作本身出发,来研究死锁问题. Souza 等考虑多种通信方式,研究了消息传递并行程序的覆盖测试,并基于控制和通信流,以及数据和消息传递流提出了多种覆盖准则[12,13];我们针对路径覆盖问题,研究了基于优化的测试数据生成方法[14]. 虽然并行程序的覆盖测试没有直接解决死锁问题,但是,执行覆盖测试过程中得到的信息,可以辅助用于死锁检测[13]. 学者们研究了语句之间的关系,用于解决串行程序测试中存在的问题. Gong 等基于目标语句等价性研究了可测试性转化,解决了标记变量问题[15];Yao 等提出了语句之间的占优关系,给出了占优关系的判定原则;利用占优关系将覆盖目标进行转化,减少语句覆盖测试数据生成的代价[16]. 考虑一个并行程序的执行逻辑. 它包含多个进程,进程之间并行执行,每个进程内部是串行执行的代码序列,通过调用 MPI 函数实现进程之间的通信和同步. MPI 函数的不当使用导致了程序死锁. 受此启发,本文将占优关系的概念应用到并行程序的死锁检测中. 分析进程内部通信语句之间的占优关系,进而得到通信边之间的占优关系;通过检查这些通信边的占优关系是否可行,判定程序中的死锁. 如果不可行,说明程序存在死锁.

2　基于占优关系的死锁检测

本节提出基于占优关系的并行程序死锁检测方法. 首先阐述通信边和占优两个关键概念,然后给出基于占优关系的并行程序死锁检测方法.

通信边:若 s 是一条消息发送语句,r 是与 s 匹配的消息接收语句,那么,s 与 r 之间存在一条由 s 指向 r 的通信边. 一般地,称 s 为起点,r 为终点;相应地,s 所在的进程为发送进程,r 所在的进程为接收进程.

占优:考虑一个进程内部的 2 条语句:$s1$ 和 $s2$. 如果语句 $s1$ 执行,语句 $s2$ 必然执行,称 $s1$ 占优 $s2$,记作 $s1 \prec s2$. 类似地,如果通信边 ℓ 执行,$\wp$ 必然执行,称 ℓ 占优 $\wp$,记作 $\ell \prec \wp$.

占优关系具有传递性,也就是说,对于 3 条语句:$s1$, $s2$ 和 $s3$, 如果 $s1 \prec s2$ 并且 $s2 \prec s3$, 那么 $s1 \prec s3$.

类似地,考虑 3 条通信边:∂,$\wp$,ℓ 如果 $\partial \prec \wp$ 并且 $\wp \prec \ell$, 那么 $\partial \prec \ell$.

考虑传统串行程序的一个语句. 如果它能够被程序入口到出口的任意路径所经过,换句话说,该语句不嵌套于任何选择结构或循环结构内,称这样的语句为主干语句. 同样地,对于 MPI 并行程序,从进程入口到出口的任意路径必须经过的语句,称为主干语句. 每次并行程序运行,进程内的主干语句都被执行. 因此,所有主干语句,都被位于它前面的主干语句占优.

考虑一个并行程序,它的通信语句属于主干通信语句. 首先,考察每个进程内通信语句的占优关系:一个主干通信语句被位于它前面的通信语句占优. 然后,通过通信语句之间的占优关系得到通信边之间的占优关系.

通信边占优关系的判定原则为:

假设 $\wp$ 和 ℓ 是两条分别以 $s1$ 和 $s2$ 为起点或终点的通信边,如果 $\wp \prec \ell$.

上述占优关系判定原则的正确性说明:根据占优定义,$s1 \prec s2$,说明如果 $s1$ 执行,$s2$ 一定被执行.因此,以 $s1$ 为起点或终点的通信边 ℓ 被覆盖,以 $s2$ 为起点或终点的通信边 $\wp$,必然也被覆盖.

最后,检查上述占优关系的可行性.如果有两条通信边,它们存在不同的占优关系,那么,程序可能存在死锁.所提方法的基本流程如图 1 所示.

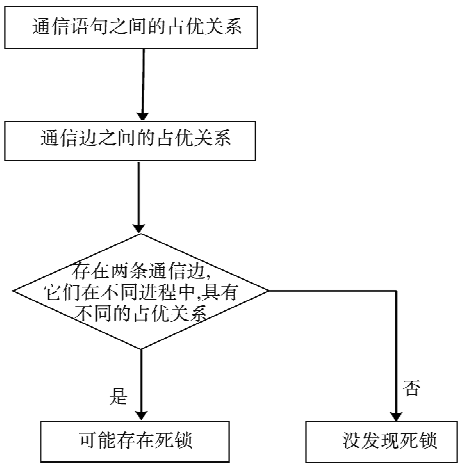

图 1 基于占优关系的死锁检测流程图

3 实例研究

本节通过 3 个实例阐述本文方法的应用过程和有效性.在给出实例之前,介绍两个典型通信函数:MPI_Send 和 MPI_Recv[2],他们用以实现消息的发送和接收.

函数原型为:

intMPI_Send (void * buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

int MPI_Recv (void * buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI Status * status).

MPI_Send(MPI_Recv)的第 1 个参数指发送(接收)缓冲区;第 2 个为消息长度,第 3 个参数为数据类型;第 4 和第 5 个参数分别为目的(源)进程和消息标签;第 6 个参数为通信域;MPI_Recv 的第 7 个参数为状态变量,记录收到消息的发送进程和消息标签等信息.通过这些参数实现发送和接收语句的匹配.

3.1 实例 1

考虑图 2 的被测程序(Program Under Test),PUT1,其中,MPI_Init 函数的作用是初始化并行环境;相应地,MPI_Finalize 用于退出并行环境;MPI_Comm_rank 获取进程号,存放在变量 myid 中.因此,条件语句"if(myid==0){}",表示如果进程号为 0,执行相应的语句.由图 2 可以看出,这个程序有两个进程.进程 0 首先执行语句 1,接收从进程 1 发送来的消息,然后,执行语句 2 向进程 1 发送一个消息;进程 1 通过语句 3 和语句 4 分别接收来自进程 0 的消息,以及向进程 0 发送消息.这里,通信语句 1,2,3,4 都是主干通信语句,语句 1 和 2 是进程 0 的主干通信语句,语句 3 和 4 是进程 1 的主干通信语句.图 3 为 PUT1 的通信示意图,其中,→代表不同进程中两个通信语句之间的通信边,…>代表了同一进程内两个通信语句之间的占优关系.图 3 中有两条通信边:E1 和 E2.

```
#include <mpi.h>
int main(int argc, char **argv)
{
    int x, y;
    int myid;
    MPI_Comm comm=MPI_COMM_WORLD;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank (comm, &myid);
    if(myid==0)
    {
1       MPI_Recv(&y,1,MPI_INT,1,0,comm,NULL);
        x=3;
2       MPI_Send(&x,1,MPI_INT,1,0,comm);
    }
    if(myid==1)
    {
3       MPI_Recv(&y,1,MPI_INT,0,0,comm,NULL);
        x=5;
4       MPI_Send(&x,1,MPI_INT,0,0,comm);
    }
    printf( "process-%d Finished\n" ,myid);
    MPI_Finalize();
    return 0;
}
```

图 2 被测程序 PUT1

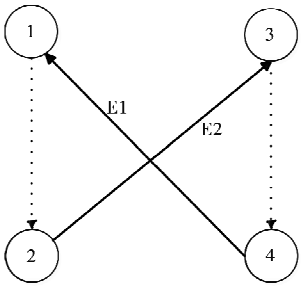

图 3 被测程序 PUT1 的通信图

通过图 2 和图 3 能够发现,E1 是以语句 1 为终点,以语句 4 为起点的通信边,接收进程为 0,发送进程为 1;E2 以语句 2 为起点,语句 3 为终点,它的发送和接收进程为分别为进程 0 和 1.

一方面,在进程 0 中,语句 1 位于语句 2 之前,因此,语句 1 占优语句 2,根据上述通信边占优关系判定原则,可以得到进程 1 内的通信边占优关系:$E1 \prec E2$;另一方面,在进程 1 中,语句 3 占优语句 4,并且 E2 是以语句 3 为终点的通信边,E1 是以语句 4 为起点的通信边,根据上述通信边占优关系判定原则,可以得到 $E2 \prec E1$.

明显地,进程 0 的 $E1 \prec E2$ 和进程 1 的 $E1 \prec E2$ 不能同时被满足,示例程序 1 存在死锁.

```
#include<mpi.h>
int main(int argc, char **argv)
{
    int x,y;
    int myid;
    MPI-Comm Comm=MPI_COMM_WORLD;
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(comm,&myid);
  if(myid==0)
  {
      x=2;
1     MPI_Send(&x,1,MPI_INT,1,0,comm);
  }
  if(myid==1)
  {
2     MPI_Recv(&x,1,MPI_INT,MPI_ANY_SOURCE,0,comm,NULL);
3     MPI_Recv(&y,1,MPI_INT,2,0,comm,NULL);
  }
  if(myid==2)
  {
      x=3;
4     MPI_Send(&x,1,MPI_INT,1,0,comm);
  }
  printf("process-%d Finished\n",myid);
  MPI_Finalize();
  return 0;
}
```

图 4　被测程序 PUT2

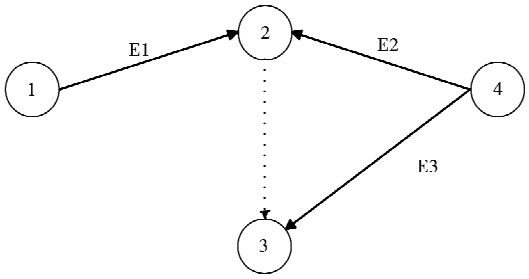

图 5　被测程序 PUT2 的通信图

3.2　实例 2

图 4 是被测程序 PUT2,通信图如 5 所示,其中,语句 2 的第 4 个参数"MPI_ANY_SOURCE"表示进程 1 在语句 2 可以接收任意进程发送来的消息.并行程序图 3 的语义为:进程 0 执行发送语句 1,向进程 1 发送消息;进程 1 首先执行接收语句 2,接收任意进程发送来的消息,然后执行接收语句 3,接收进程 2 发送来的消息;进程 2 执行发送语句 4,向进程 1 发送消息.图 5 为相应的通信示意图.

通过图 4 和 5 能够看出,示例程序 2 有 3 条通信边:E1,E2 和 E3. E1 的发送和接收节点分别 1 和 2. E2 和 E3 的发送节点都是 4,接收节点分别为 2 和 3,发送和接收进程是 1 和 2.

考察 E2 和 E3. 对于进程 1 来讲, E2 和 E3 分别是以语句 2 和语句 3 为终点的通信边,并且语句 2 占优语句 3,从而, $E2 \prec E3$; 考虑进程 2,E2 和 E3 都是以语句 4 为起点,然而,语句 4 是点到点通信语句,每次运行,E2 和 E3 只能有一条被覆盖,违反占优关系 $E2 \prec E3$, 能够判定该程序存在死锁.

3.2　实例 3

```
#include<mpi. h>
int main(int argc, char **argv)
{
    int x,y;
    int myid;
    MPI_Comm comm=MPI_COMM_WORLD;
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(comm,&myid);
  if(myid==0)
  {
        x=0;
1       MPI_Recv(&y,1,MPI_INT,1,0,comm,NULL);
2       MPI_Send(&x,1,MPI_INT,2,0,comm);
  }
  if(myid==1)
  {
        x=1;
3       MPI_Recv(&y,1,MPI_INT,2,0,comm,NULL);
4       MPI_Send(&x,1,MPI_INT,0,0,comm);
  }
  if(myid==2)
  {
        x=2;
5       MPI_Recv(&y,1,MPI_INT,0,0,comm,NULL);
6       MPI_Send(&x,1,MPI_INT,1,0,comm);
  }
  printf("process-%d Finished\n",myid);
  MPI_Finalize();
  return 0;
}
```

图 6　被测程序 PUT3

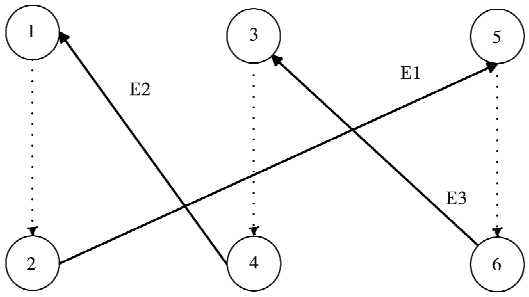

图 7　被测程序 PUT3 的通信图

被测程序 PUT3 如图 6 所示. 进程 0 通过语句 1 接收进程 1 发送来的消息,并通过语句 2 向进程 2 发送消息;进程 1 执行语句 3 和 4,接收来自进程 2 的消息,并发送消息到进程 0;进程 2 通过语句 5 和 6,接收进程 0 发送来的消息,并发送到消息到进程 1.

正如图 7 所示,该程序包含 3 条通信边,E1,E2 和 E3. E1 的发送和接收节点分别为进程 0 的语句 2 和进程 2 的语句 5;E2 以进程 1 的语句 4 为起点,进程 0 的语句 1 为终点;E3 分别以语句 6 和语句 3 为起点和终点,发送和接收进程分别为 2 和 1.

E1 和 E2 是两条分别以语句 2 和语句 1 为起点和终点的通信边. 对于进程 0 来说,语句 1 占优语句 2,因此, $E2 \prec E1$; E2 和 E3 分别以语句 4 和语句 3 为起点和终点,同时,进程 1 的语句 3 占优语句 4,因此, $E3 \prec E2$; 类似地,对进程 2 来说,E1 和 E3 分别以语句 5 和语句 6 为终点和起点,同时,语句 5 占优语句 6,因此 $E1 \prec E3$.

根据占优关系的传递性,因为 $E2 \prec E1$ 并且 $E1 \prec E3$,所以 $E2 \prec E3$,违反了进程 1 中的占优关系 $E3 \prec E2$,同样地,由 $E3 \prec E2$ 并且 $E2 \prec E1$,可以得到 $E3 \prec E1$;由 $E1 \prec E3$ 并且 $E3 \prec E2$,得 $E1 \prec E2$. 然而, $E3 \prec E1$ 和 $E1 \prec E2$ 分别违反了进程 2 和进程 0 中的占优关系. 由此,判定示例程序 3 中存在死锁.

4 所提方法的实现

4.1 原型工具框架

基于所提方法初步实现了基于占优关系的并行程序死锁检测,如图 8 所示. 原型工具以被测程序为输入,使用所提方法分析程序,并报告程序中是否有死锁. 原型工具包含 3 个模块:Comm,Comm-Dom,以及 Detection. Comm 扫描被测程序源代码,标注每个进程中的通信语句,根据通信语句之间的匹配关系,产生通信边;基于通信语句的占优关系,Comm-Dom 分析每个进程内通信边之间的占优关系;Detection 通过检查通信边之间的占优关系检测死锁. 具体地讲,在 Detection 中,对于两条通信边 $\wp$ 和 ℓ,如果在一个进程中存在占优关系: $\wp \prec \ell$,而在另一进程,或者通过占优关系的传递性得到的占优关系违反 $\ell \prec \wp$,程序中可能存在死锁.

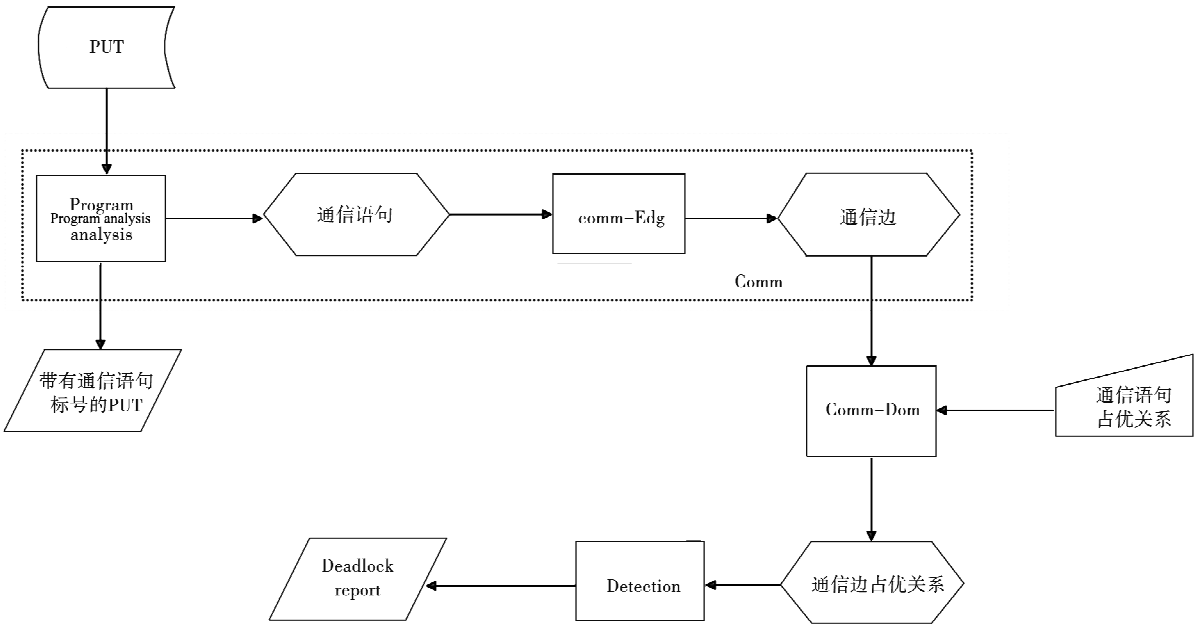

图 8 原型工具框架

4.2 实验结果

本节列出了 3 个被测程序 PUT1 和 PUT2, PUT3 的检测结果,如表 1-3 所示. 由表 1-3 可以看出,一个通信边由如下 5 个元素表示:通信边序号、发送进程、发送语句、接收进程,以及接收语句组成. 例如,表 1 所示 E1 是一条通信边,它的发送端

表 1 PUT1 的检测结果

通信边	发送进程	发送语句	接收进程	接收语句
E1	0	2	1	3
E2	1	4	0	1

检测结果:死锁. 相关的通信边为:E2 和 E1.

是进程 0 的语句 2,接收端是进程 1 的语句 3.

表 1-3 表明,基于本文方法实现的原型系统能够检测到死锁并报告相关的通信边.

表 2　PUT2 的检测结果

通信边	发送进程	发送语句	接收进程	接收语句
E1	0	1	1	2
E2	2	4	1	2
E3	2	4	1	3

检测结果:死锁. 相关的通信边为:E2 和 E3.

表 3　PUT3 的检测结果

通信边	发送进程	发送语句	接收进程	接收语句
E1	0	2	2	5
E2	1	4	0	1
E3	2	6	1	3

检测结果:死锁. 相关的通信边为:E2 和 E1,E3 和 E2,E1 和 E3.

4.3　讨论

由于通信模式,在一些情况下,虽然存在相互违背的占优关系,但是,并不会引起死锁.一种比较典型的情况如图 9 和 10 所示.

进程 0 通过语句 1 和 2 向进程 1 发送 2 个消息,进程 1 执行语句 3 和 4 接收消息.接收语句 4 与发送语句 1 匹配;接收语句 3 与发送语句 2 匹配.结合图 10 能够看出,程序中有两条通信边:E1 和 E2.对进程 0 来讲,由于语句 1 占优语句 2,因此 $E1 \prec E2$;然而,在进程 1 中, $E2 \prec E1$.根据占优关系理论,实例程序中,可能存在死锁.

进一步分析可知,当消息被安全保存后,MPI_Send 发送函数就可以返回,进程 0 可以继续向下执行语句 2.当进程 1 执行到语句 3 时,一直阻塞等待,直到收到语句 2 发送来的消息,继续向下执行语句 4,接收语句 1 发送来的消息.这说明,虽然存在相互违背的占优关系: $E1 \prec E2$ 和 $E2 \prec E1$,但是在实际执行过程中,并不会发生死锁.本文实现的原型工具对上述典型情况进行了分析,从而避免了误报.

因此,在判定占优关系的基础上,需要对违反占优关系的通信边进一步分析,以确定占优关系的相互违背是否真的引起死锁,减少误报.

```
#include <mpi. h>
int main(int argc, char **argv)
{
    int x,y;
    int myid;
    MPI_Comm comm=MPI_COMM_WORLD;
    MPI_Init(& argc, & argv);
    MPI_Comm_rank(comm, & myid);
    if(myid==0)
    {
        x=3;y=5;
1       MPI_Send(& y,1,MPI_INT,1,1,comm);
2       MPI_Send(& x,1,MPI_INT,1,2,comm);
    }
    if(myid==1)
    {
3       MPI_Recv(& y,1,MPI_INT,0,2,comm,NULL);
4       MPI_Recv(& x,1,MPI_INT,0,1,comm,NULL);
      printf( "x=%d,y=%d\n" ,x,y);
    }
    printf( "process-%d Finished\n" ,myid);
    MPI_Finalize( );
    return 0;
}
```

图 9　示例程序 3

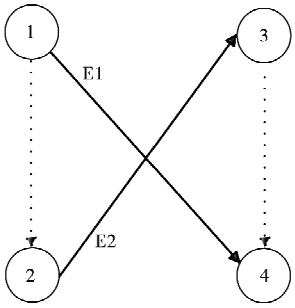

图 10　示例程序 3 的通信图

5　结论

本文利用通信语句之间的关系检测死锁,提出了基于占优关系的死锁检测方法.根据通信语句之间的占优关系分析通信边之间的占优关系.通过检查通信边占优关系是否可以满足,判定程序中的死锁.通过 3 个实例给出本文方法的应用过程,验证了本文方法的有效性,并实现了一个基于占优关系的死锁检测原型工具.本文实现了应用语句占优理论检测死锁的探索.被本文考虑了一种普遍存在的情况,即并行程序的通信语句全部位于主干语句.对于其他情况需要考虑分支相关性,得到通信语句的占优关系,根据本文思想考察通信边的占优关系实现死锁检测.此外,非阻塞通信下的死锁检测、原型工具的完善、以及所提方法与其他方法的对比,是下一步工作的重点.

参　考　文　献

[1] 张军华,臧胜涛,单联瑜,等. 高性能计算的发展现状及趋势[J]. 石油地球物理勘探, 2010, 45(6): 918-925.

[2] Gropp W D, Gropp W, Lusk E, et al. Using MPI: Portable Parallel Programming with the Message-passing Interface[M]. MIT press, 1999.

[3] Hilbrich T, Protze J, Schulz M, et al. MPI runtime error detection with MUST: advances in deadlock detection[J]. Scientific Programming, 2013, 21: 109-121.

[4] Fu X, Chen Z, Zhang Y, et al. MPISE: Symbolic execution of MPI programs[C]. //2015 IEEE 16th International Symposium on High Assurance Systems Engineering (HASE), 2015.

[5] AlDhubhani R, Eassa F, Saeed F. Exascale message passing interface based program deadlock detection[J]. International Journal of Electrical and Computer Engineering, 2016, 6(2): 887.

[6] Luecke G, Chen H, Coyle J, et al. MPI-check: a tool for checking Fortran 90 MPI programs[J]. Concurrency and Computation: Practice and Experience, 2003, 15(2): 93-100.

[7] Falzone C, Chan A, Lusk E, et al. Collective error detection for MPI collective operations[C]. //European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting, 2005.

[8] Vakkalanka S S, Sharma S, Gopalakrishnan G, et al. ISP: A tool for model checking MPI programs[C]. //The 13th ACM sigplan Symposium on Principles and Practice of Parallel Programming, 2008.

[9] Vo A. Scalable formal dynamic verification of MPI programs through distributed causality tracking[D]. PhD dissertation, University of Utah, School of Computing, March, 2011.

[10] Vetter J S, De Supinski B R. Dynamic software testing of MPI applications with umpire[C]. //The 2000 ACM/IEEE Conference on Supercomputing, 2000.

[11] Hilbrich T, De Supinski B R, Schulz M, et al. A graph based approach for MPI deadlock detection[C]. //The 23rd International Conference on Supercomputing, 2009.

[12] Souza S R S, Vergilio S R, Souza P S L. Structural testing criteria for message-passing parallel programs[J]. Concurrency and Computation: Practice and Experience, 2008, 20: 1893-1916.

[13] Souza P S L, Souza S R S, Zaluska E. Structural testing for message-passing concurrent programs an extended test model[J]. Concurrency and Computation: Practice and Experience, 2013, 25(18): 149-158.

[14] Tian T, Gong D. Evolutionary generation approach of test data for multiple paths coverage of message-passing parallel programs[J]. Chinese Journal of Electronics, 2014, 23(2): 291-296.

[15] Gong D, Yao X. Testability transformation based on equivalence of target statements[J]. Neural Computing and Applications, 2012, 21(8): 1871-1882.

[16] Yao X, Gong D, Luo Y, et al. Test data reduction based on dominance relations of target statements[C]. //IEEE Congress on Evolutionary Computation, 2012.

Deadlock Detection of Parallel Programs based on Dominance Relation

TIAN Tian GUO Qing ZHANG Chen

(School of Computer Science and Technology, Shandong Jianzhu University, Jinan 250101, China)

Abstract MPI parallel programs have been applied broadly and their reliability has attracted more and more attention from the software engineering community. A MPI parallel program contains several processes, each of which has a series of statements executed sequentially. The deadlock is a typical error that occurs in parallel programs. This study proposes a method of detecting deadlocks based on dominance relation. Firstly, the dominance relations of communication edges are firstly conducted from those of communication statements. Secondly, the deadlock is detected by checking the dominance relations of communication edges. The proposed method is preliminarily implemented and used for detecting several MPI parallel programs. The experimental results verify the effectiveness of the proposed method.

Key words parallel programs; dominance relation; deadlock; communication edge

(上接第 82 页)

Phase Estimation with Thermal State and Fock State via Parity Detection

WANG Shuai WU Shi-chen SUI Yong-xing

(School of Mathematics and Physics, Jiangsu University of Technology, Changzhou 213001, China)

Abstract In quantum precision based optical interferometer, the phase sensitivity crucially depends on the nature of the quantum state. We analytically prove when a Mach-Zehnder interferometer powered by a Fock state into one input port, a thermal state, a squeezed vacuum state or a squeezed thermal state with same average photon number at the other input, the same performances in the phase estimation via parity detection can be obtained for small phase shift. We analytically prove that the parity detection saturates the quantum Cramér-Rao bound. In addition, our results show that besides a coherent state, a thermal state is another useful high-intensity classical state for the phase estimation.

Key words quantum precision; Mach-Zehnder interferometer; parity detection; thermal state; Fock state