并行计算通信库测试方法研究及实践*

熊玉庆[1], 张云泉[2]

[1](中国科学院 计算技术研究所,北京 100080)

[2](中国科学院 软件研究所,北京 100080)

E mail: yqxiong@publica.bj.cninfo.net; zyq@mail.rdcps.ac.cn

http://www.rdcps.ac.cn

摘要: 并行计算通信库的测试在并行计算系统中起着重要的作用.对通信库的测试一般都是通过设计一些测试程序对库的各个或几个部分分别进行单独隔离测试.但是有许多库中的错误用这种隔离测试方法测不出来,只有当库的多个部分以某种复杂的、有机的方式组合运行时才会暴露出来.而这种复杂的、有机的组合方式,从设计库的测试角度看很难形成.提出两种新的测试方法,根据通信库结构的分层特性,利用可移植的上层库的测试程序来测试下层库.上层库的测试程序也可看做是下层库的应用程序,但与一般的下层库应用程序不同,它几乎覆盖了下层库的各个部分,且有机地将它们组合起来,运行时形成某种复杂的形态,而仅用下层库的测试程序往往达不到这种形态.这样,逃过下层库测试程序的错误就可能暴露出来.

关键词: 并行计算;通信库;测试方法;MPI;BLACS

中图法分类号: TP338 **文献标识码**: A

并行计算通信库的测试对并行计算系统至关重要,因为一个正确的通信库是并行计算系统的基础.对通信库的测试,一般是对库的各个或几个部分分别进行隔离测试[1,2].虽然也有所谓的系统测试,但往往只是考虑到了有限的方面.然而,库中的许多错误是在多个成分以某种有机的方式组合起来相互作用时才暴露出来,而人们很难从测试角度有意识地构造出这种情形.

本文根据通信库的结构特性,提出两种新的测试方法.一般地,通信库结构都是分层的.例如,用于线性代数计算的通信库 BLACS[3](basic linear algebra communication subprograms)是建立在 MPI(message passing interface)[4]和 PVM[5]等通信库上的,而 MPI 的一个实现 MPICH[6]又是建立在 P4[7]之上的.这两种新的测试方法就是利用某平台上正确的上层通信库的可移植测试程序来测试另一平台上的下层通信库.上层通信库测试程序也可看做是下层通信库的应用程序,但是,由于上层库的实现一般覆盖了下层库的各个部分,上层库的测试程序又覆盖了上层库的各个部分,因此与一般的下层库应用程序不同,上层库的测试程序也覆盖了下层库的几乎所有部分,且将它们有机地组合起来,在运行时,它们形成一种对下层库来说极为复杂的情形.这种复杂的情形从下层通信库测试角度来看很难人为地构造出来.在这种情形下,许多下层库测试程序不能测试的错误就可能暴露出来.

本文应用已成功地在 Hitachi SR2201 和曙光 2000 上运行的、建立在 MPI 和 PVM 上的 BLACS 的测试程序来测试其他并行计算系统(在本文中将它们统称为 X 系统)上已通过 MPI 测试程序测试的 MPI 系统为例,证明本文提出的测试方法的可行性.

1 通信库测试新方法原理

测试方法 1. 设 LC 是一个在平台 Ω 上的要测试的通信库,UC 是建立在 LC 上的可移植通信库(如图 1 所

* 收稿日期: 1999-07-26; 修改日期: 1999-09-28

基金项目: 国家 863 高科技项目基金资助项目(863-306-ZD01-03-02)

作者简介: 熊玉庆(1964—),男,江西吉安人,博士,主要研究领域为并行处理,人工智能;张云泉(1973—),男,山东聊城人,博士,主要研究领域为大型并行数值软件,并行程序设计和性能评价.

示).TEST_UC 是 UC 的一个测试程序,在另一平台 Θ 上 TEST_UC 成功地通过对 UC 的测试.因而在平台 Θ 上确保了 TEST_UC 的正确性.为了对 Ω 上的 LC 库进行测试,将 TEST_UC 在 Ω 上运行.由于 UC 是可移植的,因而如果 Ω 上的 LC 是正确的话,TEST_UC 应该与在 Θ 上一样成功通过;若不能,则应该可以判定是 Ω 上的 LC 或其下层库有错误.

Fig. 1 Software structure of communication library UC (testing method 1)
图1 通信库UC的结构(测试方法1)

测试方法 1 只能确定 LC 或其下层库有错误.如果需要准确地确定错误位置,则可应用下面的测试方法 2.

测试方法 2. 设 LCa 和 LCb 是两个在平台 Ω 上的要测试的通信库,它们建立在另一较低层通信库 LLC 上.UC 是建立在 LCa 和 LCb 上的可移植通信库(如图 2 所示).TEST_UC 是 UC 的一个测试程序,在另一平台 Θ 上,TEST_UC 成功地通过对 UC 的测试.因而在平台 Θ 上确保了 TEST_UC 的正确性.为了对 Ω 上的 LCa 和 LCb 库进行测试,将 TEST_UC 在 Ω 上运行.由于 UC 是可移植的,因而如果 Ω 上的 LCa 和 LCb 是正确的话,TEST_UC 应该与在 Θ 上一样成功通过;若不能,则:(1) 若对建立在 LCa 上的 UC,TEST_UC 不能通过,而对建立在 LCb 上的 UC,TEST_UC 能通过,则应该可以判定是 Ω 上的 LCa 有错误;(2) 若对建立在 LCb 上的 UC,TEST_UC 不能通过,而对建立在 LCa 上的 UC,TEST_UC 能通过,则应该可以判定是 Ω 上的 LCb 有错误;(3) 若都不能通过,则 Ω 上 LLC 有错误的可能性比 LCa 和 LCb 都错误的可能性要大.

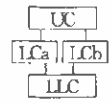

Fig. 2 Software structure of communication library UC (testing method 2)
图2 通信库UC的结构(测试方法2)

在上述两种测试方法中,UC 的可移植性很关键,否则,当 TEST_UC 在 Ω 上不能通过时,就不能判定是 Ω 上的 LC(对于测试方法 1)或 LCa 或 LCb 或 LLC(对于测试方法 2)有错误.因为这时候的错误可能是由于 UC 不可移植带来的.

测试方法 1 比测试方法 2 简单、快捷,但测试方法 2 可以更准确地定位错误.当用测试方法 1 确定 LC 或其下层库有错误,但很难确定错误的准确位置时,可用测试方法 2.若满足测试方法 2 中(3)的条件,则错误出在 LC 下层库的可能性很大.这时,可将注意力集中在 LC 的下层库中进行排错.

2 BLACS,BLACS 测试程序及 MPICH 测试程序

BLACS 是面向线性代数计算的、可移植的通信库,是建立在 MPI 和 PVM 等较低层的通信库上的.在并行计算系统 X 上,BLACS 的结构如图 3 所示.

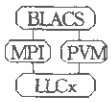

Fig. 3 Software structure of communication library BLACS on X system
图3 通信库BLACS在X系统上的结构

BLACS 系统基本上由两部分组成:通信子程序和支撑子程序.通信子程序由点到点通信、广播通信和组合操作组成.支撑子程序主要由系统初始化、退出系统及逻辑进程网格信息获取组成.建立在 MPI 和 PVM 上的 BLAC 的这些部分几乎涉及到 MPI 和 PVM 的各个部分.

BLACS 测试程序对 BLACS 的上述各个部分分别进行测试,重点在通信子程序,尤其是广播通信和组合操作测试较多,对各种矩阵的形状和大小、各种数据类型、各种逻辑进程网格形状以及群通信时数据在进程之间的流向等进行多种组合测试.

MPICH 是 MPI 的一个著名的实现[1].MPICH 测试程序也是对其各个部分分别进行测试.这些测试是由很多小的测试程序来执行的.它们分成以下几类:

(1) 测试各种点到点通信子程序;

(2) 测试各种群通信子程序;

(3) 测试通信上下文(context)操作;

(4) 测试环境子程序;

(5) 测试拓扑子程序;

(6) 一些其他测试程序.它们进行一些所谓的系统测试,但考虑的情形很简单.

由此可以看出,MPICH 的测试程序并没有对 MPICH 的各个部分进行复杂的混合测试.事实上,很多复杂的情形从测试程序的设计角度也很难构造出来.

BLACS 测试程序虽然是为测试 BLACS 而设计的,但考虑的情况是很有限.然而,它对于下层的 MPI 来说,却是一个极复杂的应用程序.由于 BLACS 几乎涉及到 MPI 的各个部分,而 BLACS 测试程序又完全覆盖了 BLACS 的所有部分,因而与一般的 MPI 应用程序不同,它几乎覆盖了 MPI 的各个部分,且将它们有机地组合在一起,运行时,呈现出一个极其复杂的状态,这种状态是 MPICH 测试程序达不到的.这样,逃过了 MPICH 测试程序的错误可能这时就暴露出来了.基于这一情况并利用第 1 节的方法,我们用已经在曙光 2000 和 Hitachi SR2201 上顺利运行了的 BLACS 测试程序来测试新的并行系统上的 MPI 实现.

3 BLACS 测试程序对 X 系统上的 MPI 的测试

X 系统上的 MPI(其实现是 MPICH)已经通过 MPI(即 MPICH)测试程序的测试.但运用上述测试方法及 BLACS 测试程序,我们发现这些系统的 MPI 存在以下问题:

(1) MPI 同步及进程挂起问题.由于 BLACS 测试程序提供了输入参数的组合测试(数据精度、实现算法(逻辑拓扑)、进程网格、消息长度),使得用户很容易通过修改输入文件参数改变测试的组合数目及测试模式,从而改变了原有的 MPI 软件包中同一种通信模式多次重复测试的做法,使得整个测试具有动态的效果.由于有些系统在实现 MPI 的同步时为了提高同步速度,采用硬件实现所有进程的同步,但未考虑到只有部分进程参与同步时的情况,这会导致部分进程同步时的挂起现象.这种情况在原有的 MPI 软件包单一重复的测试情况下,是不可能查出来的.而运用测试方法 1 和 BLACS 测试程序,通过不断改变参与同步的进程数目,使该问题暴露出来.

(2) 消息处理器与 MPI 通信 buffer 之间的 Cache 一致性问题.MPI 软件包在进行正确性测试时,只选取小规模问题进行测试,这就使得系统的某些关键参数不能受到充分的测试.而这些关键参数正是消息系统设计时的分支点(重新判断分包及 buffer 重用等操作),极易发生意想不到的问题.由于 BLACS 测试程序可以很容易地产生大量不同类型和长度的消息传递测试,所以我们运用测试方法 1 和 BLACS 测试程序,在消息量大、buffer 出现重用时,发现了有的系统消息处理器与 MPI 通信 buffer 之间的 Cache 一致性问题.

(3) 更底层库中出现的问题.利用方法 1 找到 MPI 的上述 Cache 一致性问题后,我们没有在 MPI 上发现问题所在,于是再利用测试方法 2.由于 BLACS 是建立在 MPI 和 PVM 等底层通信库上的,而且 PVM 和 MPI 版的 BLACS 测试程序都在 Hitachi SR2201 和曙光 2000 上成功通过.因而可以利用测试方法 2.在前面(2)中提到的 Cache 一致性问题,由于在 MPI 层上不能确定问题的所在,为了更准确地定位错误的位置,我们运用测试方法 2,将 PVM 版的 BLACS 测试程序进行同样的测试,结果也发现类似的问题.由于当出现测试方法 2 中的情形(3)时,就可将错误更精确地定位在 MPI 和 PVM 的下层库,因此,我们可将问题定位在 MPI 和 PVM 的共同下层库 LLCx 上.事实证明这样的判断是正确的.

在上面(2)的测试中,我们只能将问题定在 MPI 及其以下的库上.由于在 MPI 层很难找到问题所在,于是采用测试方法 2,将问题更准确地定位.从上面的(2)和(3)可以得到,如何将测试方法 1 和测试方法 2 结合起来进行查错.

(4) MPI_Isend()的问题.运用测试方法 1 和 BLACS 测试程序,我们发现 MPI_Isend()对各种不同长度的长消息连续、大量地发送处理上的问题.如果消息长度变化不是足够大,这个问题就很难发现.

(5) 消息流 Send/Recv 不同组合(个数,顺序,长度)产生的问题.由于 BLACS 中的群通信采用大量的不同算法,这些算法能测试到 Send/Recv 的大量不同组合,从中发现一些不易发现的问题.应用方法 1,当 BLACS 测试程序测试分裂环(split ring)(如图 4 所示)广播算法时出现问题.经检查,问题出在 MPI 中.排除该问题后,BLACS 测试程序测试该广播算法时顺利通过.MPI 中的这种问题只用一般的 Send/Recv 组合是很难发现的.

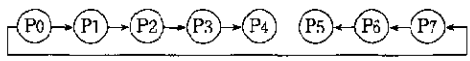

Fig. 4 Software structure of split ring topology
图 4 分裂环拓扑结构

4 总 结

通过用BLACS测试程序对并行系统上的已经通过MPI测试程序测试的MPI进行测试,结果表明,本文对通常测试方法缺陷的分析是正确的,所提出的新的测试方法是可行的,具有实用价值.目前,我们已经应用上述两种测试方法,将BLACS测试软件作为测试新的并行计算系统上的MPI和PVM等通信库的重要辅助工具.

致谢 本文的工作是在中国科学院软件研究所并行软件研究开发中心进行的,并在该中心孙家昶研究员的帮助和指导下完成.在此,向他们表示诚挚的谢意.

References:

[1] Bridges, P., Doss, N., Gropp, W., *et al*. Installation Guide to Mpich, a Portable Implementation of MPI, 1995. http://www.mcs.anl.gov/mpi/mpich/index.html.

[2] Whaley, R.C. Installing and Testing the BLACS, 1995. http://www.netlib.org/blacs/Blacs.html.

[3] Dongarra, J., Whaley, R.C. A user's guide to the BLACS v1.0. Technical Report, UT CS-95-281, LAPACK Working Note #94, University of Tennessee, 1995.

[4] Forum, M.P.I. MPI: a message passing interface standard. International Journal of Supercomputer Applications and High Performance Computing, 1994,8(3/4).

[5] Geist, A., Beguelin, A., Dongarra, J., *et al*. PVM: Parallel Virtual Machine——ε User's Guide and Tutorial for Network Parallel Computing. Cambrige, MA: MIT Press, 1994.

[6] Gropp, W., Lusk, E., Doss, N., *et al*. A high-performance, portable implementation of the MPI message passing interface standard. Parallel Computing, 1994,22(6):789~828.

[7] Butler, R., Lusk, E. Monitors, messages and clusters: the P4 parallel programming system. Parallel Computing, 1994,20(4):547~564.

Study and Practice of Testing Approaches of Communication Libraries for Parallel Computing

XIONG Yu-qing[1], ZHANG Yun-quan[2]

[1](*Institute of Computing Technology, The Chinese Academy of Sciences, Beijing* 100080, *China*);
[2](*Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)
E-mail: yqxiong@publica.bj.cninfo.net; zyq@mail.rdcps.ac.cn
http://www.rdcps.ac.cn

Received July 26, 1999; accepted September 28, 1999

Abstract: Testing of communication libraries for parallel computing plays an important role in parallel computing systems. In general, testing of communication libraries is done by some testers designed to test every or several parts of the libraries separately. However, many errors of libraries can not be tested by the separate methods, and they will appear when many parts of libraries are running by combination of them in term of a kind of complicated and organic way. But it is rather difficult that the complicated and organic combinations result from the design of library testers themselves. This paper proposes two new testing approaches, which are based on the feature of layered library architectures and test lower libraries by portable testers of upper libraries. The testers of upper libraries can also be regarded as application programs of lower libraries, but they are different from general application programs of lower libraries. They almost cover every part of lower libraries, combine them organically, and form a complicated situation in run time, which can not be easily obtained only by testers of lower libraries. In this case, the errors may be exposed which can escape from the testers of lower libraries.

Key words: parallel computing; communication library, testing approach; MPI (message passing interface); BLACS (basic linear algebra communication subprograms)