多机虚拟执行技术的应用研究

薛瑞尼 翟季冬 陈文光 郑纬民
清华大学

关键词：虚拟执行 并行计算 行为分析 性能预测

引言

由于记录/重演（record/replay）并不是在机器上完全放开程序的执行，而是在一定的控制下执行，特别是在重演阶段，是在运行时的控制下，按照记录阶段的过程进行虚拟的执行，所以，本文将记录/重演的技术手段称为虚拟执行方式。在记录运行过程中，虚拟执行工具记录的是程序的输入数据（如消息）。如果需要对某个进程进行调试，则将其运行在重演状态。程序的执行顺序和记录阶段完全相同，且运行完全受程序员控制，可以方便地跟踪程序状态。

现有的面向并行的程序，例如消息传递接口（message passing interface，MPI）程序的虚拟执行工具，可以分为两类：第一类是**数据重演**（data replay）。在记录过程中，每个进程都将所有收到的消息完整地记录下来，在重演过程中，程序员就可以独立地重演任何一个进程而无需启动其它进程；第二类是**顺序重演**（order replay）。在记录过程中，只记录可以引起程序不确定性的函数的输出（即不确定事件的确定性顺序），如MPI程序中带有MPI_ANY_SOURCE参数的MPI_Recv。重演过程中的发送进程负责重新生成消息内容，并发送至接收进程。不确定性操作的执行序列和记录阶段的完全相同。由于顺序重演只记录不确定性事件的时序信息，产生的日志规模远小于数据重演。

数据重演的好处是进程之间相互独立，缺点是需要记录的日志量可能非常庞大。而顺序重演正好相反，所记录的非确定性时序的数据量比较少，但是需要运行多个进程来配合。文献[1]提出基于这两种方式的优点，可将二者结合在一起，通过将程序进行分组，在分组之间进行数据重演，而在分组内部则进行顺序重演。这种虚拟执行的方式能够在降低数据记录量的同时降低分组之间的程序依赖性。

记录/重演可以用于性能预测。性能预测是通过分析、测试和模拟等手段预测一个程序在某个目标平台上的执行时间。这也是计算机领域中一个重要的研究课题。在并行程序性能预测方面，一般有基于分析的方法、基于模拟的方法以及黑盒技术。本文所阐述的通过虚拟执行方式进行并行程序性能预测是一种结合了分析与模拟方式的方法。此方法的思路是先对并行程序进行分析，利用虚拟执行方式在模拟器上解析出其通信模式，这是因为通信模式对于并行程序的性能预测具有非常大的影响[3]；随后，通过使用一个目标节点虚拟执行程序记录下的行为，完成对节点的性能模拟，并结合前述的通信模式，就能够完成对于并行程序的性能分析[4]。

MPIWiz：针对并行MPI程序的分组再现重演的虚拟执行方法

并并行程序的虚拟执行方法MPIWiz针对的是当前主流的并行编程模式MPI。MPI是分布式计算环境中实现粗粒度并行编程的事实标准。然而，MPI程序内在的不确定性和计算规模使得调试程序非常困难。

MPIWiz方法的基本思路是在进行记录的时候做分组记录，重演阶段按照组来虚拟执行。这种

设计基于这样的观察事实：并行程序中的进程通常聚集为不同的小组，其通信主要是组内进程间的通信。一些研究发现[5~8]消息传递并行程序有很好的通信局部性和相似性：（1）根据通信频率和通信量，所有进程可以划分为多个小组。各进程与组内进程的通信明显超过组外进程，因此组内通信占主要地位；（2）不同组别之间的通信模式非常类似，因此在某个小组中出现的问题在其它组中也可能出现。

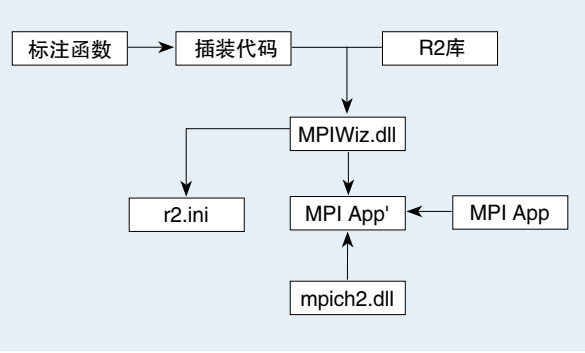

图1 MPIWiz的构建和使用流程

基于这些事实可以得到一种新的重播方法：以组为粒度进行记录和重播。组间通信使用数据重播，组内通信使用顺序重播。组内通信在重播时会再次产生，而组间通信则直接从日志中获取。这种方案可以从两方面减小日志规模：一是只有一个或者几个重播组中的进程参与记录，而不是所有进程都记录；二是占据主要通信量的组内通信可以在重播时生成，无需记录。因此，日志中只包含组间通信的消息内容、组内通信的顺序信息以及有不确定性特征的系统调用的信息。

MPIWiz原型系统支持对应用程序函数和系统调用的动态插装，用户程序无需代码修改和重新链接就可以处理MPI程序和系统调用中的不确定性函数。Windows平台的MPIWiz构建于R2[9]平台之上。图1描述了MPIWiz的构建和使用流程。首先由程序员提供被插装MPI函数的标注信息（标注函数），然后利用MPIWiz的代码生成工具基于标注函数生成对应的插装代码，再将插装代码与R2库链接生成MPIWiz动态链接库（MPIWiz.dll）。将R2的启动动态链接库注入用户编译好的MPI程序（MPI App）得到可插装的应用程序（MPI App'）。MPI程序启动时首先将R2模块初始化，同时读取配置文件（r2.ini）以确定当前程序运行在记录状态还是重播状态。我们在Linux平台的MPIWiz基本开发完毕，近期将以开源软件的形式发布。MPIWiz虚拟执行方式将MPI程序的进程划分成多个重演组，每个重演组可以独立记录和重演。具体介绍如下：

日志记录 只有重播组中的进程参与日志记录。

1. 所有具有不确定性特征的组内通信采用顺序重播的方式：即记录消息顺序，忽略消息内容。MPI程序中的消息具有“不可超越性（non-overtaking）”，如果一个进程给另一个进程发送两条或者更多消息，则目的进程接受消息的顺序必须与源进程发送消息的顺序完全一致。因此，如果接收操作是确定性的（没有任何参数为任意值），则这些操作之间的顺序就无需记录。相反，如果消息具有不确定性，则必须记录其顺序。

2. 不确定性的系统调用采用数据重播的方式。这是因为具有不确定性特征的系统调用数量多、语义复杂，此处的简化可以降低系统的复杂度。

3. 所有来自于组外的消息都采用数据重播的方式：即记录消息的顺序和内容。因为只有同一重播组内的进程才能被重播，所以来自组外进程的消息无法重新生成，必须采用数据重播的方式记录。向组外进程发送的消息不影响重播，可以直接忽略，无需记录。

重播虚拟执行 同一个重播组中的进程必须一起重播，而不属于任何重播组的进程不能重播。所有组内通信都要重新生成。如果操作具有不确定性则从日志中读取真实参数替换通配符，以保证消息的顺序跟记录时的顺序相同。所有来自组外的消息直接从日志中获取，所有发往组外的消息直接忽略。

以NPB（NAS Parallel Benchmarks，美国宇航局高级并行计算基准程序）为例，实验结果显示，使用虚拟执行方法MPIWiz可以正确重演含有不同类型不确定性函数的MPI程序。对于通信局部性较好

的程序，分组再现重播产生的日志量平均只有数据重播的38%。与程序的正常执行时间相比，日志记录运行时间开销约为其27%，虚拟执行后期的重演过程可以在53%的时间内完成。

FACT：基于部分虚拟执行方式的快速通信模式获取技术

通信性能是影响基于消息传递的并行程序性能的关键因素。并行程序的通信模式可以通过通信量、通信的空间和时间属性三个方面衡量。准确地理解并行程序的通信模式，不但可以优化并行程序的性能，还可以帮助设计通信子系统模块，改进基于重演技术的并行程序调试工具的开销。

为了达到上述目的，基于虚拟执行技术，我们提出一个可以快速获取通信踪迹（Traces）的方法——FACT（FAst communication trace collection）。FACT通过静态分析的方法得到原始程序的一个程序切片（program slice），然后再执行这个程序切片获得程序的通信模式。这种执行方式实际上并不真正执行原来的程序，而只是虚拟执行一次，以获得相关的参数。FACT系统的设计是基于并行程序的一个特性：并行程序中大部分的计算代码和通信内容都与程序的通信量和通信空间无关，因此，删除这些计算代码和通信操作并不影响程序实际的通信模式。

相比现有的方法，FACT系统具有如下特点：

资源需求小 由于FACT系统通过静态分析技术、程序切割，删除了程序中不相关的计算语句和内存分配语句，因此，获得的程序切片在执行过程中需要计算资源和内存资源很少。

踪迹收集时间短 由于不需要执行全部的计算代码和通信操作，因此，整个收集通信踪迹的时间可以大大减少。

图2是FACT系统的设计框架图。FACT系统主要包括两个模块：编译模块（compilation framework）和运行时模块（runtime environment）。运行时模块是支持分片后的程序进行虚拟执行的引擎。FACT有如下使用方式：

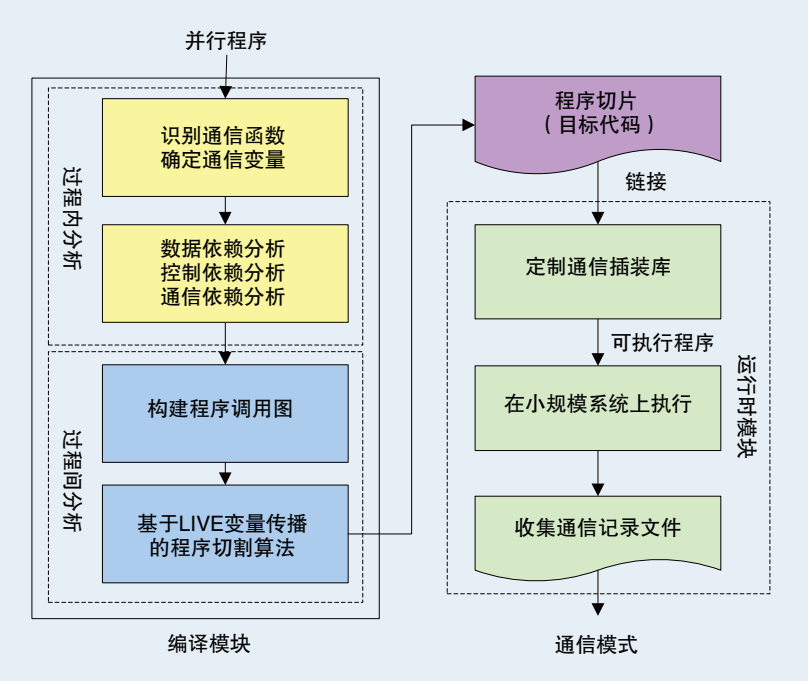

图2 FACT系统的设计框架图

用以程序切片的编译模块 编译模块分成进程内分析和进程间分析两个阶段。在进程内分析中，FACT系统解析程序源码，识别MPI通信语句并确定需要保留的通信参数，同时收集程序的依赖关系：数据依赖、控制依赖和通信依赖；在进程间分析，需要构建程序的调用流程图，采用我们提出的基于LIVE变量的切割算法删除不相关的计算语句和内存分配语句，并标记相应的通信操作语句。

用以支持虚拟执行的运行时模块 编译模块得到的程序切片不能直接在并行系统上运行，需要一个运行时模块。运行时模块提供了一个定制的通信插装库，程序切片后链接插装库，然

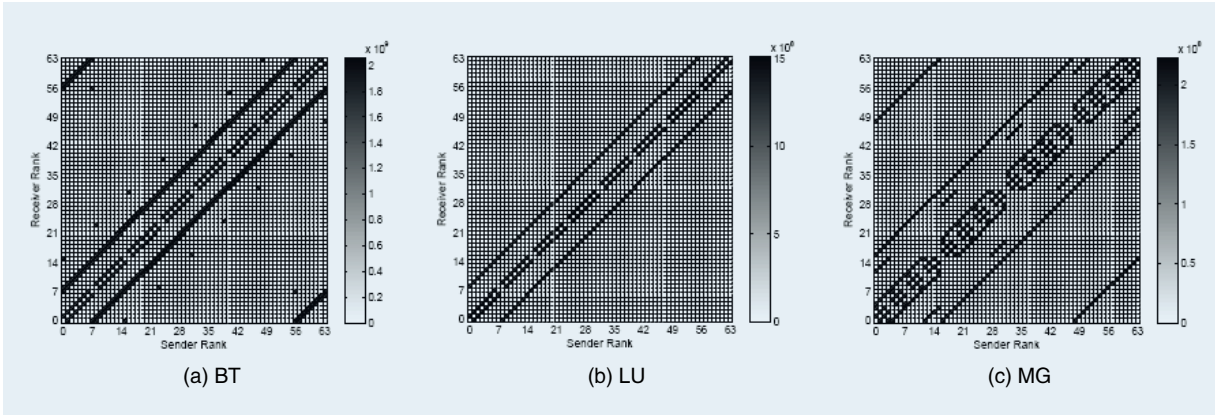

图3 使用FACT系统获取的程序的通信模式

后在小规模系统上执行。运行时模块根据编译时标记的信息，决定在执行过程中是否真正在网络上传输消息内容。运行时模块还负责记录通信模式相关的信息，包括消息类型、消息大下、消息源目的地址等，并保存在本地的日志文件中。

我们用7个NPB程序和ASCI Sweep3D程序在曙光5000A系统上验证了FACT系统。图3列出了部分程序的通信模式，包括BT（Block Tri-Diagonal，NPB核心测试程序之一，求解三对角块方程组。）、LU（Lower-Upper Triangular，NPB核心测试程序之一，对称超松弛法求解块稀疏方程组。）分解和MG（3D MultiGrid，三维多重网格，NPB核心测试程序之一，采用三维多重网格算法求解三维Poisson方程。）。可以看出大部分的通信都发生在相邻进程上，程序的通信模式表现出很好的局部性。

FACT能有效减少程序的计算和内存需求。对大部分程序而言，程序的资源需求减少了1～2个数量级。例如，在64进程下，原始的Sweep3D消耗26.61GB内存，在512进程下，消耗213.83GB 内存，而FACT系统分别仅需要0.13GB内存和1.25GB内存。我们在一个拥有32个核心的小规模系统上，成功地收集了直到512个进程的D规模NPB程序和Sweep3D（150×150×150）的通信traces。由于内存的限制，传统的基于插装的方法无法在这么小的系统上收集程序的通信。

PHANTOM：使用虚拟执行方式的并行程序性能预测技术

大规模并行计算机包括成千上万个计算节点，在研制过程中，一般需要花费数年的时间和很高的成本。对于这些计算机的设计人员，在设计阶段预测某个应用在未来设计系统上的性能对于提前发现系统瓶颈并改进系统设计具有重要意义。

精确地预测某个应用在未来系统上的性能不但可以辅助设计未来计算机，还可以帮助应用人员和开发者在未来系统可用前，提前发现应用程序的瓶颈，从而优化并改进程序性能。

然而，精确地预测并行程序在未来系统上的性能是一件非常复杂的事情。因为，并行程序的执行时间由程序中的顺序计算时间、通信时间以及它们之间的卷积三个因素决定。当前已有的技术不能精确地获得并行程序中顺序计算时间，尤其是对大规模的并行程序，从而导致并行程序的性能预测精度不高。

我们设计的PHANTOM技术能够使用虚拟执行方式在单个节点上对并行程序进行性能预测。PHANTOM的系统总体结构如图4所示。

PHANTOM系统（图3）主要包括计算模块、通信模块和网络模拟器模块。计算模块负责分析原始程序中不同进程之间的计算相似性，选择有代表

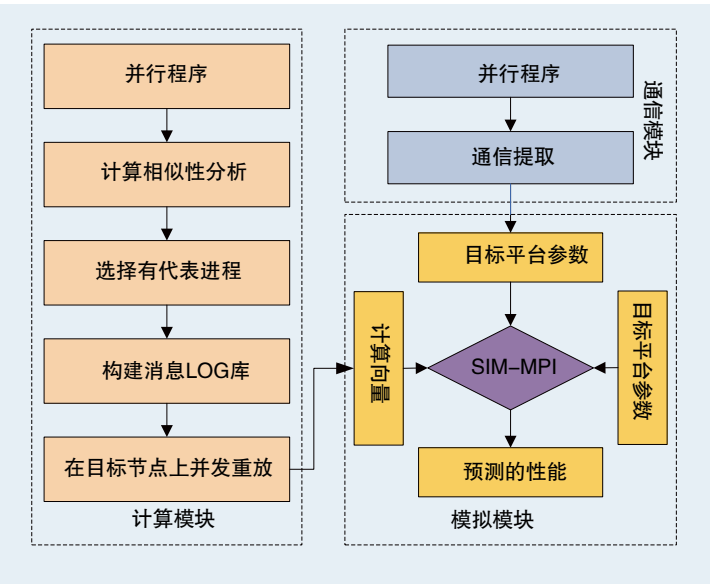

图4 PHANTOM系统的框架

性的进程，同时在已有虚拟并行平台上收集程序的通信日志，以通信日志数据库的形式存放。然后，当预测程序在目标平台上的性能时，把有代表性的进程在目标平台的一个节点重演执行，得到程序的顺序计算时间。最后，将上述几个步骤获得的参数值通过虚拟执行的方式，获得并行程序在目标平台的性能预测结果。

PHANTOM系统的特点：（1）是一个自动的虚拟执行引擎与性能预测框架，不需要用户了解并行程序本身复杂的算法和程序的设计结构；（2）采用代表性重播虚拟执行技术获取并行程序的顺序计算时间，比已有的方法更加精确，同时该技术仅仅需要目标平台单个节点；（3）模拟器所需的通信记录文件是通过本文提出的程序切片技术获取的，可以解决在一个小规模系统上获取大规模并行程序通信记录的难题；（4）事件驱动模拟器SIM-MPI是运行MPI程序的模拟器，对并行程序的网络通信模型进行了虚拟化。SIM-MPI基于一个改良的Log-GPO通信模型实现，该模型能够比较精确地对MPI程序中各种通信开销，尤其是程序中计算和通信的重叠行为对程序性能的影响建模。

通过大量的实验数据分析，PHANTOM能够以非常高的精度预测程序在集群中的运行性能。在曙光5000A平台上通过比较实际测试时间，PHANTOM预测时间以及基于回归模型的预测时间发现，PHANTOM预测时间点上的误差线显示了90%的置信区间，使用PHANTOM系统预测的时间与程序的实际执行时间非常接近。对所有的程序，PHANTOM系统预测的平均误差小于8%。对EP（Embarrassingly Parallel，NPB核心测试程序之一，并行度高，通信少，测试数学函数的浮点性能）程序，由于该程序几乎没有通信操作，所以它的预测精度几乎反映了基于代表性重放技术获取程序顺序计算时间的精度。PHANTOM 对EP 程序的平均预测误差仅为0.34%。

通过对多个平台的分析，PHANTOM在曙光5000A、深腾胖节点服务器和深腾刀片节点服务器上的平均预测误差分别为：2.67%，1.30% 和2.34%。与基于回归的方法相比，PHANTOM 预测系统具有更高的预测精度和更稳定的预测结果。

《CCF科学技术奖评选条例》获得通过

2011年9月8~19日，CCF常务理事对《中国计算机学会科学技术奖评选条例》进行了通信投票表决。CCF秘书处共收到有效投票24张，其中同意票21张，弃权及反对票3张。有效投票数超过常务理事会人数（32人）三分之二法定人数。本次投票有效，上述条例获得通过。（阳）

结语

本文介绍了清华大学有关虚拟执行技术在分析并行程序行为方面的最新进展。文献[2]叙述了在多机环境下，如何通过将进程分组降低记录的开销，提高虚拟执行的效率。该方法结合了数据重演与顺序重演的优点。文献[3 ~ 4]叙述了在多机环境下如何通过虚拟执行的方式来预测大规模程序在集群环境下的性能。■

薛瑞尼

CCF学生会员，清华大学计算机系博士。主要研究方向为分布式系统、并行计算、移动云计算。

xueruini@gmail.com

翟季冬

CCF会员、2010 CCF优秀博士学位论文奖获奖者。清华大学博士。主要研究方向为高性能计算机性能和可靠性评测技术、大规模并行程序性能分析、优化和预测技术。zhaijidong@gmail.com

陈文光

CCF高级会员、YOCSEF（2011~2012年度）主席，本刊编委。清华大学计算机系教授。主要研究方向为并行计算的编程模型、并行化编译和操作系统。

cwg@tsinghua.edu.cn

郑纬民

CCF副理事长、会士。清华大学计算机系教授。主要研究方向为并行/分布处理、网络存储器、编译系统。

zwm-dcs@tsinghua.edu.cn

参考文献

[1] Xue R, Liu X, Wu M, et al. MPIWiz: Subgroup Reproducible Replay of MPI Applications. Proceedings of 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP' 09), Raleigh, NC, USA, 2009. 251~260

[2] LeBlanc T J, Mellor-Crummey J M. Debugging Parallel Programs with Instant Replay. IEEE Trans. Computers, 1987, 36(4):471~482

[3] Jidong Zhai, Tianwei Sheng, Jiangzhou He, Wenguang Chen, Weimin Zheng. FACT: Fast Communication Trace Collection for Parallel Applications through Program Slicing. Conference on High Performance Computing Networking, Storage and Analysis (SC' 09), 1~12, Portland, Oregon, Nov 14-20, 2009

[4] Jidong Zhai, Wenguang Chen, Weimin Zheng. PHANTOM: Predicting Performance of Parallel Applications on Large-scale Parallel Machines Using a Single Node. 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP' 10), Bangalore, India, Jan 9-14, 2010, 305~314

[5] Faraj A, Yuan X. Communication Characteristics in the NAS Parallel Benchmarks. Proceedings of International Conference on Parallel and Distributed Computing Systems (PDCS' 02), 2002. 724~729

[6] Vetter J S, Mueller F. Communication Characteristics of Large-scale Scientific Applications for Contemporary Cluster Architectures. J. Parallel Distrib. Comput., 2003, 63(9):853~865

[7] Kim J, Lilja D J. Characterization of Communication Patterns in Message-Passing Parallel Scientific Application Programs. Proceedings of Network-Based Parallel Computing: Communication, Architecture, and Applications (CANPC' 98), volume 1362 of Lecture Notes in Computer Science, Las Vegas, Nevada, USA: Springer, 1998. 202~216

[8] NAS Parallel Benchmarks: ProActive implementations. http://proactive.inria.fr/index.php?page=nas_benchmarks, 2007

[9] Guo Z, Wang X, Tang J, et al. R2: An Application-Level Kernel for Record and Replay. Proceedings of Symposium on Operating System Design and Implementation (OSDI’08), 2008