文章编号：2096-1618(2018)06-0617-07

基于 MPI 的多核并行模式的性能测试与分析

王亚茹， 王 鹏， 王德志
(西南民族大学计算机科学与技术学院，四川 成都 610041)

摘要：为了对蒙特卡洛并行程序进行有效的表达，提出一种并行计算程序流程图的表示方法，并行算法在实际应用中对效率制约的瓶颈就是进程间的通信过程。在“多计算、少通信”与“少计算、多通信”两种不同并行模式下，通过控制集群节点数以及节点核数来测试基于 MPI 实现的蒙特卡洛算法的加速比与效率，分析并行程序在多核并行模式下的计算性能。结果表明采取“计算换通信”的策略能有效地实现对算法的并行加速，提高了系统的可扩展性，为今后提升并行编程效率提供了参考，弥补了传统的串行程序流程图无法表述多进程之间信息沟通及协作的不足。

关 键 词：并行算法；蒙特卡洛；加速比；效率；计算换通信；并行计算程序流程图
中图分类号：TP302.7 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2018.06.004

0 引言

高性能计算已被广泛应用于天气预报、人工智能等众多领域，俨然成为科技创新的重要手段。并行计算是实现高性能计算的技术基础。消息传递接口(message passing interface，MPI)是一种基于消息传递的并行编程标准[1]，为集群节点之间的通信提供一个重要框架[2]。传统的 MPI 要通过网络联系，所以 MPI 自身具有多核并行能力。由于节点间的多核结构导致核与核之间存在通信过程，而通常网络情况下的数据传输速度与 CPU 计算速度相比要慢得多，消息通信量的增加会大大降低系统运行程序的速率，集群规模越大影响越大。若要充分发挥高性能计算集群的高速硬件优势，则需采用适宜的调优算法来实现众核之间的并行[3]，因此在 MPI 的编程原则中就有“计算换通信”的说法[1]。通过蒙特卡洛法对 π 值的计算来研究在多核并行状态下 MPI 的编程性能，对 MPI 多核并行模式在不同通信方式下加速比与效率进行分析，了解在多核并行模式下该如何提高系统性能[4-5]。在实验中通过控制集群间的节点数以及核数来进行进程之间“多计算、少通信”以及“少计算、多通信”这两种并行模式的多核并行系统的性能测试。此外，传统的串行流程图已无法满足并行程序的需求，故通过定义并行程序流程图的基本符号以详细介绍蒙特卡洛求 π 算法程序的具体运行步骤。

收稿日期：2018-06-05
基金项目：国家自然科学基金资助项目（60702075）；四川省青年科学基金资助项目（09ZQ026-068）

1 关键技术及性能测试标准

1.1 MPI

MPI 是一种基于消息传递编程的模型，并成为并行编程模型中的代表和标准。消息传递机制使服务器之间能够有机地结合在一起形成一个庞大的计算系统，使服务器之间有效地进行数据交换从而实现计算目标[1]。通过适当的配置，MPI 可用于多核之间的并行，在并行规模上实现平滑的线性性能提升，可伸缩性极强。

1.2 加速比与并行效率

并行计算中，衡量代码并行化后的性能提升基准就是加速比 S_p。对于并行计算来说，多核并行模式的运行时间与核数成反比则是并行化成功的理想状态，即核数越多，运行时间越短[6]。但在并行程序运行过程中，处理器不可能将全部时间都用来执行程序，在计算时也有可能消耗部分时间来进行消息通信或产生其他系统噪声使理想状态产生偏差，而并行效率 E_p 就用来衡量这种偏差[7]。采用加速比和效率来对系统性能进行评估。

S_p 定义为相同任务分别在单处理器上运行消耗的时间 T_1 和多个并行处理器 p 上运行消耗的时间 T_p 的比值[6]；效率 E_p 定义为 S_p 与 p 的比值：

$$S_p = \frac{T_1}{T_p} \tag{1}$$

$$E_p = \frac{S_p}{p} \tag{2}$$

理想状态下，p 个处理器的并行系统的加速比为

p,效率为 1。通常加速比越大,表明程序并行化后越具有良好的可扩展性,也具有线性的性能提升能力;而效率越大表明一个处理器的计算能力被有效利用的比率越大。由文献[7]可知,运行效率一般而言如果低于 0.5 就说明并行优化失败,此时的并行效果还不如串行程序;而并行效率在0.75以上则表明系统并行化效果显著。

1.3 并行计算流程图

并行算法流程的描述一直以来都没有一个通用的方法,给理解和分析并行程序带来了极大的困难。由于并行算法在运行过程中涉及多个进程且多个进程之间存在复杂关联,而传统的串行计算程序流程图无法表述并行环境下复杂信息之间的沟通和多进程之间的协作,所以长期以来,在并行领域都没有找到一个能有效地表述存在于进程间的动态通信过程的并行程序表述方法。尝试提出一种描述简单的并行程序的基本符号并将其用于蒙特卡洛求 π 算法当中,具体的基本符号表述如表 1 所示。

表 1 并行计算流程图基本符号

流程	示例	主进程任务	全进程任务	子进程任务
开始	–	–	开始	–
结束	–	–	结束	–
过程	m++	0 m++	m++	N-1 m++
判定	$i<1$?	0 $i<1$?	$i<1$?	N-1 $i<1$?
发送通信	发送 m 值	0 m N-1	–	N-1 m 0
接收通信	接收 m 值	0 m N-1	–	N-1 m 0

主要采用 6 个标志分别对于流程中的开始、结束、过程、判定以及消息通信进行定义。假设系统共有 N 个进程,0 为主进程,则 N-1 表示所有子进程。当只有一个进程执行流程时,则用单框表示;当程序并发执行时则用复框表示且左侧输入执行该任务的进程号。对于开始和结束标志与串行流程图基本符号相同;对于过程流程,以自增运算 m++为例,框内输入流程 m++,当只有主进程单独执行时,由于只有一个进程执行流程,则用单框表示,且左侧输入 0 以表示执行本流程的仅有主进程。同理,若由所有子进程并发执行时,则用复框表示,且左侧输入 N-1 以表示该流程由所有子进程执行。对于判定流程同上。而对于消息传递流程,以传递 m 值为例,消息通信框中输入 m,发送通信流程符号上方表示消息的发送者,下方表示接收者;反之,在进程的接收通信中,上方表示消息的发送者,下方表示消息的接收者。

1.4 通信机制

1.4.1 内核通信机制

操作系统为系统内部各进程间的消息传递与数据交换提供了共享内存的通信方式,将需要共享的内存页面映射到各个相互通信的进程的虚拟地址空间里,使其如同访问自己的私有物理内存页面一样[8]。所有共享内存的各个进程由于同处于同一个计算机系统,其通信速度显然最为快捷,但并行程序存在临界区,当需要使用互斥机制来执行共享内存程序的临界区时,使用互斥机制就会强制执行并行程序中临界区的代码,各进程之间会存在抢占总线资源的竞争。这也是串行程序中无须消耗的调用互斥量的代价。

1.4.2 节点通信机制

由于节点间是物理上独立的存在,所以节点的变量地址空间是独立的。若节点之间没有能够进行消息传递的机制,则节点间是无法进行消息通信的。MPI 可以提供相当便捷的节点间数据交换和数据控制的能力,可以说这种消息传递机制就是 MPI 编程的核心功能。但 MPI 的消息传递机制要靠网络进行传输,因此在并行计算集群中要采用高速通信技术来实现节点间的数据通信[1]。这种方式比通过访问局部内存存储器中的数据要慢得多,而串行程序没有这些额外的开销[9]。此外,随着进程数的增多,开销也会增大,因此增加节点数意味着有更多的数据需要跨网络进行传输。

在节点内进行并行计算时,采用内核通信机制,通过共享内存的方式进行数据的传输。多核并行理论上运行效率优于串行计算,但此时串行程序无须执行临界区代码,而并行程序各进程要抢占总线资源。在“多计算、少通信”的并行模式下,通信量较低,理论上会优于串行计算,但在“少计算、多通信”的并行模式下,通信量急剧增加,各进程抢占资源激烈,此时运行效率理论上低于“多计算、少通信”的并行模式。

当跨节点进行并行计算时,节点间通过网络进行传输。并行程序运行速率理论上优于串行程序,但节点之间会有网络通信的开销。在“多计算、少通信”的并行模式下,通信量较低,对网络的依赖性低,此时并行计算理论上会优于串行计算,但在“少计算、多通信”的并行模式下,通信量极大,各进程在网络上对时间的消耗增大,理论上运行效率低于“多计算、少通信”的并行模式。

2 π 值的蒙特卡洛计算方法

概率算法可以较容易地设计两种极端情况下的并行模式,一种是具有线性加速比的良好算法,另一种是

可扩展性较差的并行算法。而利用蒙特卡洛求 π 算法可以灵活地控制进程间的计算量和通信量,理论上可以设计出具有线性加速比的并行模式[10]。实验设计引入蒙特卡洛求 π 算法进行系统性能的测试与分析。以下介绍在串行和并行两种不同方式下蒙特卡洛求 π 的方法[2,10]。

2.1　串行方法

根据蒙特卡洛法原理,以坐标原点为圆心,以直径为 1 作一个单位圆,该圆外切一正方形,由系统在正方形内产生 *count* 个随机点,并判断该随机点是否落在圆内,将落在圆内的点数记为 m。根据概率学理论,则单位圆的面积与正方形的面积之比就是落在圆内的点数与正方形内的点数与之比,即:

$$\frac{m}{count}=\frac{\pi\times 0.5^2}{1},\ \pi=\frac{4\times m}{count}$$

2.2　并行方法

蒙特卡洛方法求解 π 值可以简易控制并行模式中计算与通信的频率,故有以下两种并行模式[11-12]:

第一种并行模式(多计算、少通信):各个进程产生 *count* 个随机点并统计落在圆内的点数 m,统计完成后与主进程通信发送 m 值,主进程汇总各个进程的 m 值后统一求 π,子进程流程图如图 1 所示。

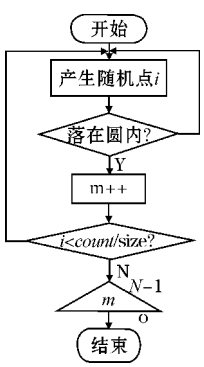


图 1　多计算少通信流程图

第二种并行模式(少计算、多通信):各个进程产生 *count* 个随机点并判断该点是否落在圆内,若在圆内则与主进程进行通信发送该点,主进程最终汇总所有圆内点数求 π,子进程流程图如图 2 所示。

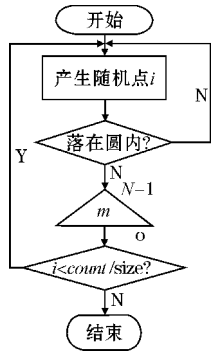


图 2　少计算多通信流程图

第一种并行模式中,各个进程各自统计出在圆内的点数 m 后向主进程发送 m 值,在整个计算过程中子进程计算 m 次,与主进程通信 1 次;而第二种并行模式中,当判断出点落在圆内时便与主进程通信 1 次,由主进程汇总所有进程发送的点数,则子进程需计算 0 次,通信 m 次。显然第二种并行模式在整个集群下的计算过程中会给集群造成较大的通信压力,理论上会使系统运行时间加大,加速比下降,并行效率下降,系统性能降低。

3　实验结果及分析

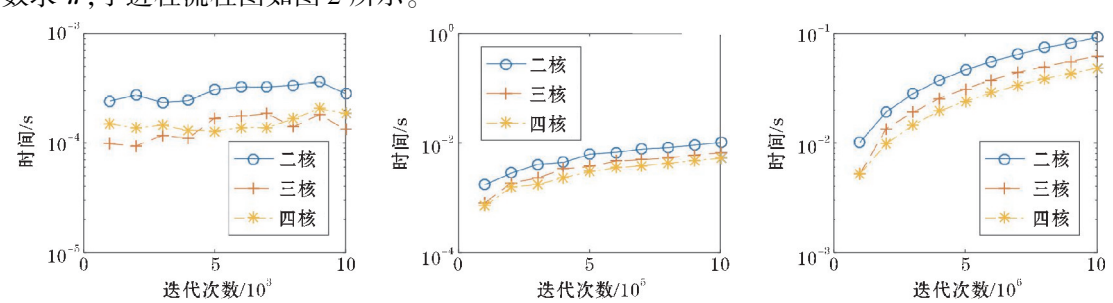
3.1　并行实验环境

实验所采用的并行计算系统搭建的软硬件规格为:实现 MPI 的免费开源软件 MPICH 3、5 台配置完全相同的 PC 机、Centos6.8 的操作系统、CPU 型号为 Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz、主频为 800 MHz。

实验随着迭代次数的增加,采用不同节点数与不同核数,分别测试在两种并行模式下程序运行所需的时间变化,以下数据由实验取 20 次结果求平均所得。

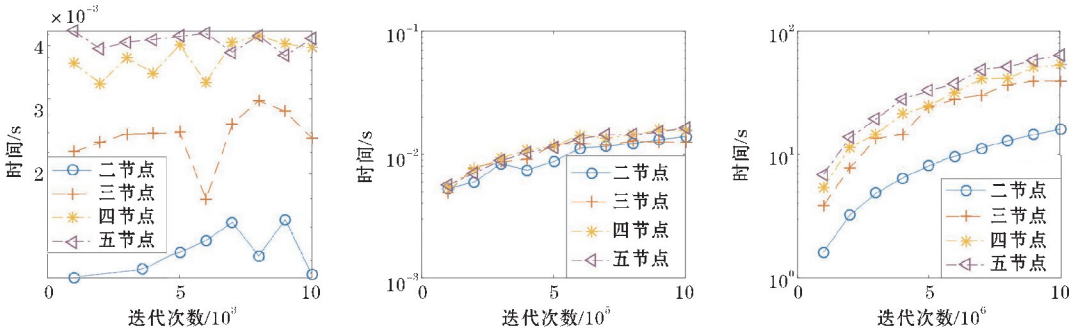
3.2　迭代次数选择

以节点内以及跨节点的"多计算,少通信"并行模式为例:当迭代次数选择10^3量级、10^5量级、10^6量级下的结果分别如图 3 ~4 所示。



(a) 10^3量级　(b) 10^5量级　(c) 10^6量级

图 3　节点内多计算少通信结果变化



(a) 10^3量级 (b) 10^5量级 (c) 10^6量级

图 4 跨节点多计算少通信结果变化

由图 3、4 可知,随着迭代次数量级的增加,运行时间逐渐增加。由于计算机无法完全屏蔽系统运行的噪声,起初迭代次数采取10^3、10^5量级时,程序运行时间出现无规则状态,测试实验计算量不足,受系统其他进程运行噪声影响较大,实验结果规律不明显;当迭代次数较大时,实验测试时间较长,其他进程等的噪声干扰可忽略不计,测试结果逐渐成线性状态。测试时加大迭代次数才能保证测试结果有效,故选择10^6量级作为测试标准以减少对测试程序时间的干扰。今后检测也应以10^6量级为最低标准。

3.3 单节点多核计算

在同一节点内,采用“多计算、少通信”并行模式,随着迭代次数的增加分别测试在同一节点内使用单核和多核情况下的运行结果如图 5 所示。

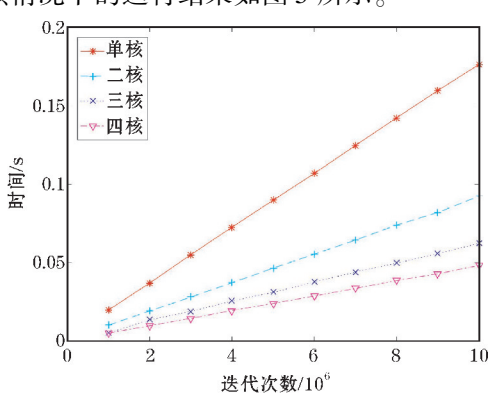


图 5 单节点多核多计算少通信比较

由图 5 可知,运行时间:$T_{p1}>T_{p2}>T_{p3}>T_{p4}$,在该并行模式下,对于只有单节点运行程序的系统,多核并行有利于系统性能的提高,且在同一节点内,核数越多,运行时间越短。

3.4 单节点多核通信

在同一节点内,采用“少计算、多通信”的并行模式,随着迭代次数的增加分别测试在同一节点内使用单核和多核的情况下的运行结果如图 6 所示。

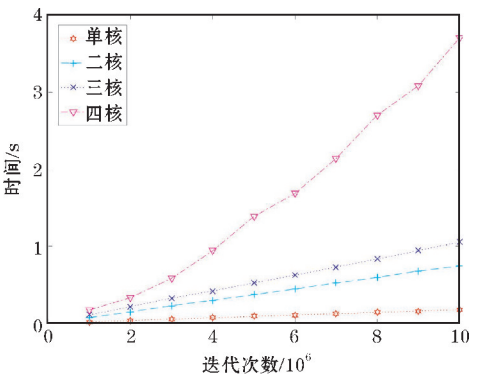


图 6 单节点多核少计算多通信比较

由图 6 可知,运行时间:$T_{p1}>T_{p2}>T_{p3}>T_{p4}$。则对于多通信的并行模式而言,当只有单节点运行程序时,随着核数增加通信次数在增加,共享内存的进程数增加,以此对总线资源的竞争越激烈,核数越多,运行速率越慢,此时的多核并行效果甚至不如串行计算。

3.5 多节点单核计算

在多个节点内,每个节点仅采用一核,通过“多计算,少通信”的并行模式,随着迭代次数的增加分别测试使用单节点及多节点并行时系统运行情况如图 7 所示。

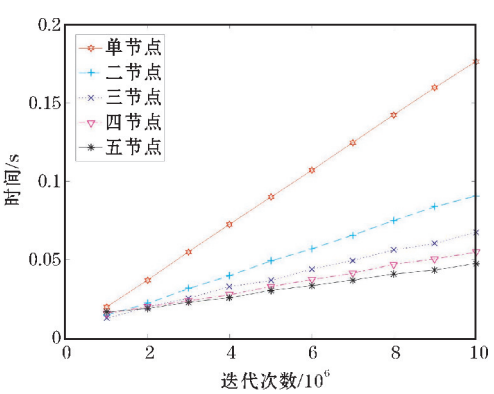


图 7 多节点单核多计算少通信比较

以上结果表明,运行时间:$T_{p1}>T_{p2}>T_{p3}>T_{p4}>T_{p5}$。跨节点运行时,随着节点数的增加,即使通过网络的跨

节点通信对时间的消耗代价巨大,而在该并行模式下的通信量远远小于计算量,此时多节点并行仍会加速系统运行。

3.6　多节点单核通信

在多个节点内,每个节点仅采用一核,通过少计算,多通信并行模式,随着迭代次数的增加分别测试在使用单节点及多节点并行时程序的运行结果如图 8 所示。

以上结果表明,运行时间:$T_{p1}>T_{p2}>T_{p3}>T_{p4}>T_{p5}$。跨节点运行时,由于“少计算、多通信”的并行模式大大加剧了通过网络通信的时间损耗,程序运行时间随着节点数的增加而急剧增加,此时多节点并行不利于程序的运行,反而串行计算下程序的运行时间更短。

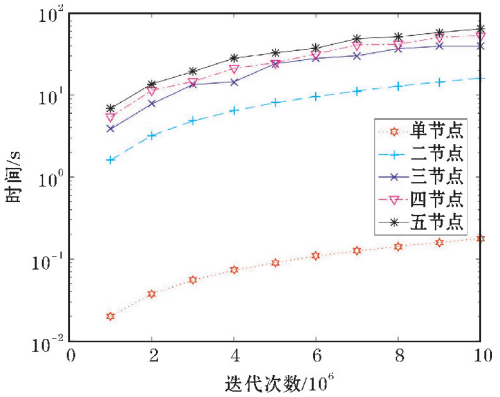
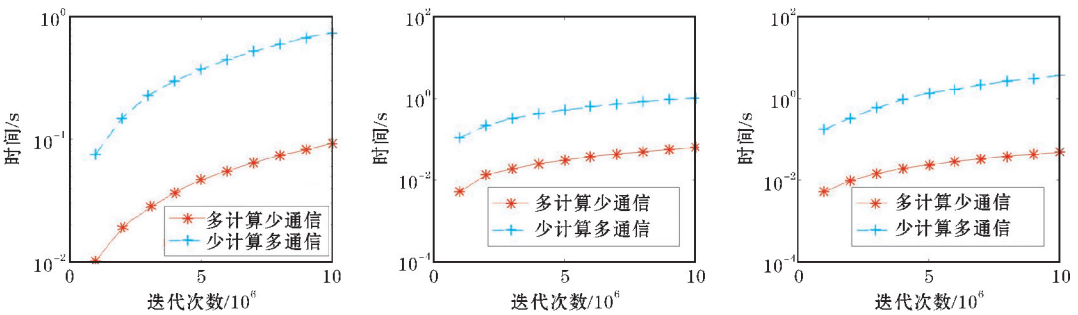


图 8　多节点单核少计算多通信比较

3.7　两种并行模式下的比较

在以上两种并行模式下,分别采用同一节点内的二核、三核、四核做运行时间的数值比较,结果图 9 所示。



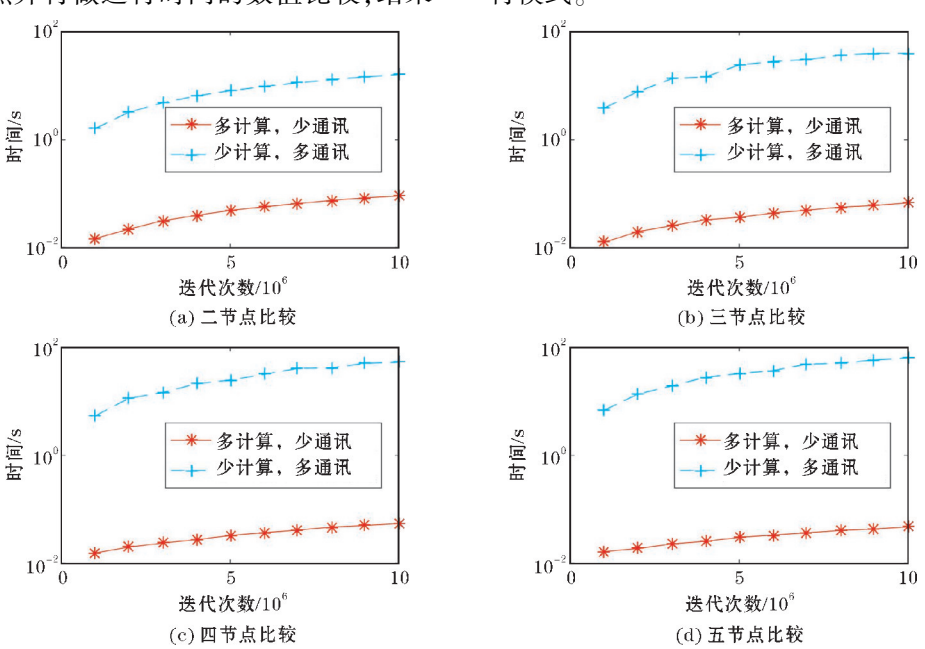
(a) 二核比较　(b) 三核比较　(c) 四核比较

图 9　单节点多核下两种并行模式的比较

由图 9 可见无论在单一节点内采用多少核实现并行计算,即使迭代次数很低,多计算,少通信的模式的时间消耗都远低于少计算,多通信的并行模式。

在两种并行模式下,集群分别采用二节点、三节点、四节点、五节点并行做运行时间的数值比较,结果如图 10 所示。

类似于单节点多核的运行效果,迭代次数与节点数的多少并不影响两种并行模式的比较结果。多计算少通信模式对于时间的消耗远小于少计算多通信的并行模式。



(a) 二节点比较　(b) 三节点比较

(c) 四节点比较　(d) 五节点比较

图 10　多节点单核两种并行模式比较

算法并行化后的加速比与效率用来衡量并行系统或程序的可并行性。不同并行模式下的多核并行方式的加速比与效率如表 2 所示。

表 2 两种并行模式下的性能比较

计算方式	多计算少通信模式		少计算多通信模式	
	S_p	E_p	S_p	E_p
单节点单核	1	1	1	1
单节点二核	1.907	0.954	0.240	0.120
单节点三核	2.829	0.943	0.169	0.056
单节点四核	3.655	0.914	0.048	0.012
二节点单核	1.945	0.973	0.111	0.056
三节点单核	2.612	0.871	0.005	0.002
四节点单核	3.220	0.805	0.003	0.0075
五节点单核	3.707	0.741	0.002	0.0004

由表 2 可知,在第一种并行模式中,加速比随着核数或节点数的增加而增加,此时多核与多节点并行有利于系统速率的提升,而此时的运行效率虽然随着核数或节点数的增加而降低,但运行效率都在0.75附近或以上,此时的并行效果是非常成功的,系统可扩展性也非常好;若并行效率在0.5以下,表明每个处理器的计算能力被有效利用的比率还不足 50%,造成了多核硬件资源的极大浪费。而在第二种并行模式中,加速比不仅逐渐减小,而效率在0.1附近及以下,甚至低至0.04%,此时系统并行不利于程序运行性能的提高,运行效果甚至远不如串行程序。

3.8 实验结论

根据以上实验,可以得到以下结论:

(1)起初迭代次数较小时,程序运行时间较短,此时的实验结果受系统其他进程产生的噪音等影响较大,故今后应选择较大量级(不低于10^6量级)的迭代次数以进行实验。

(2)无论集群采用多少核数和节点数来实现并行计算,在“少计算、多通信”的并行模式下,此时通信量大大增加意味着节点内对于总线资源抢占的时间消耗巨大,跨节点对于网络通信消耗的时间巨大,此时“少计算、多通信”的并行模式更有利于系统性能的提高。

(3)当采用“多计算、少通信”并行模式时,由于对程序运行影响最大的通信过程的时间代价降为最低,此时集群的多核或多节点并行配置有利于系统性能的提高,且进程数越多,系统运行速率越快。

(4)当采用“少计算、多通信”的并行模式时,由于系统的通信量大大增加,节点内每增加一核或集群每增加一个节点会使系统运行速率降低,系统加速比下降,并行效率下降,此时多核并行不利于系统性能的提高,串行计算效果反而更佳。

(5)系统通信量较大时,由于网络带宽有限,此时各进程对于抢占总线资源的时间消耗大大低于各进程跨网络进行数据交换的时间消耗。

(6)开发人员在编程过程中应当尽量采用适宜的调优算法来减少通信量,提高各个进程的运算量而减少进程之间的通信以使整个程序运行时间缩小,提高整个系统的运算速率。

(7)在并行编程中若无法避免频繁通信时,应注意尽量减少节点数或核数以减少算法的运行时间且优先使用核内并行模式。

4 结束语

在 MPI 编程中,进程间的通信对于程序运行时间的消耗会大大降低系统的运行效率,尤其是当节点个数较多时,通信会成为影响系统性能的重要因素。通过使用蒙特卡洛计算 π 值的算法对系统的性能进行了测试,结果表明系统可以通过增加节点或内核的运算量,减少其通信次数的途径,即用“计算换通信”的方式使其尽量降低通信量,或在通信量较低的情况下增加核数或节点数以提高系统性能。若系统无法避免高额通信量时,应尽量减少进程数或尽量采用核内并行模式来高速实现高性能计算的并行处理,对算法并行化,提高算法并行性能具有一定价值。提出的表述并行算法流程的新方法,可以有效地对并行算法的运行过程进行可视化描述,使并行程序的执行过程更加清楚明朗。

参考文献:

[1] 王鹏,黄焱,安俊秀. 云计算与大数据技术[M]. 北京:人民邮电出版社,2014:66-67.

[2] 王鹏. 云计算的关键技术与应用实例[M]. 北京:人民邮电出版社,2010:29-31.

[3] Shubhangi Rastogi, Hira Zaheer. Significance of Parallel Computation over Serial Computation Using OpenMP, MPI, and CUDA[J]. Quality, IT and Business Operations, 2017:359-367.

[4] 罗红兵,张晓霞. MPI 集合通信性能可扩展性研究与分析[J]. 计算机科学与探索,2017,11(2):252-261.

[5] 米东,施小清. 基于 MPI 的 TOUGHREACT 并行

性能分析[J]. 水文地质工程地质,2016,43(1):34-40.
[6] 谢超,麦联叨,都志辉. 关于并行计算系统中加速比的研究与分析[J]. 计算机工程与应用,2002,26(3):66-68.
[7] 刘文志. 并行算法设计与性能优化[M]. 北京:机械工业出版社,2015:110-130.
[8] 高珂,陈荔城,范东睿. 多核系统共享内存资源非配和管理研究[J]. 计算机学报,2015,38(5):1020-1034.
[9] 周伟. 并行计算在海洋水龄谱模拟中的应用研究[D]. 邯郸:河北工程大学,2013:18-19.
[10] 康辉,王家琦. 基于 PI 演算的并行编程语言[J]. 吉林大学学报(工学版),2016,46(1):235-241.
[11] Yang C T, Huang C L, Lin C F. Hybrid CUDA, OpenMP, and MPI parallel programming on multicore GPU clusters[J]. Computer Physics Communications, 2011, 182(1):266-269.
[12] 王蕾,崔慧敏. 任务并行编程模型研究与进展[J]. 软件学报,2013,24(1):77-90.

Performance Test and Analysis of Multi-core Parallel Mode based on MPI

WANG Ya-ru, WANG Peng, WANG De-zhi
(School of Computer Science and Technology, Southwest Minzu University, Chengdu 610225, China)

Abstract: In order to express the Monte Carlo parallel program effectively, an expressing method of the parallel computing program flowchart is proposed, It's the communication between processes that restricts the efficiency of parallel algorithms in practical applications. There are two communicating parallel modes, which is "more-computing, less-communicating" and "less-computing and more-communicating". In order to analyze the computation performance of parallel programs in multi-core and the two communicating parallel modes, the Monte Carlo algorithm based on MPI is used to test the speedup and efficiency by controlling the number of cluster nodes and the cores within node. The results show that the strategy of "computing for communicating" can effectively speedup the paralleling acceleration of the algorithm, improve the scalability of the system and provide reference for improving the efficiency of parallel programming in the future. which makes up for the lack of the traditional serial program flow chart that can't express the information communication and cooperation between multiple processes.

Keywords: parallel algorithm; Monte Carlo; speed up; efficiency; computing for communicating; parallel computing program flow chart