Computer Science and Application 计算机科学与应用, 2019, 9(10), 1859-1866
Published Online October 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.910208

Hans汉斯

Implementation and Performance Analysis of MPI + Open MPI Hybrid Programming

Peiqin Fan, Lin Zhang, Shuai Tang, Jingyi Liu

Navy Submarine Academy, Qingdao Shandong
Email: similaroly05@163.com

Received: Sep. 19th, 2019; accepted: Oct. 4th, 2019; published: Oct. 11th, 2019

Abstract

In order to give full play to the computing power of SMP cluster, combined with the characteristics of SMP cluster parallel system architecture, the design and implementation of multilevel hybrid parallel program based on SMP cluster are studied by using MPI + OpenMP hybrid parallel programming mode. The performance test case of hybrid parallel programming is developed by using MPI to realize the rough degree partition of tasks between nodes and OpenMP to realize the fine degree partition of tasks in nodes. The test result shows that MPI + OpenMP hybrid programming mode can effectively make use of the multi-level parallel mechanism of SMP cluster, and give full play to the advantages of the two programming models and effectively improve the computing efficiency of SMP cluster.

Keywords

Cluster, Hybrid Programming, Parallel Computation, Performance Analysis

MPI + OpenMPI混合编程的实现与性能分析

范培勤，张　林，唐　帅，刘敬一

海军潜艇学院，山东　青岛
Email: similaroly05@163.com

收稿日期：2019年9月19日；录用日期：2019年10月4日；发布日期：2019年10月11日

摘　要

为充分发挥SMP集群的计算能力，结合SMP集群并行系统体系架构的特点，采用MPI + OpenMP混合并行编程模式，研究了基于SMP集群的多级混合并行程序的设计和实现方法。通过节点间使用MPI消息传递接口实现任务的粗粒度划分，节点内使用OpenMP实现细粒度任务并行划分的方法，开发了混合并行

文章引用: 范培勤, 张林, 唐帅, 刘敬一. MPI + OpenMPI 混合编程的实现与性能分析[J]. 计算机科学与应用, 2019, 9(10): 1859-1866. DOI: 10.12677/csa.2019.910208

编程性能测试用例，并对其进行了测试和分析。测试结果表明，MPI + OpenMP混合编程模式，能够有效利用SMP集群节点间和节点内多级并行机制，充分发挥消息传递编程模型和共享内存编程模型各自的优势，可有效提升SMP集群的计算效率。

关键词

集群，混合编程，并行计算，性能分析

1. 引言

SMP 集群作为高性能计算系统成为目前应用最为广泛的高性能并行计算平台。并行程序设计模型通常可以分为共享内存和消息传递两类，MPI 和 OpenMP 分别是这两类并行程序设计模型的代表，也是目前应用最为广泛的并行程序开发环境。MPI 基于大粒度的进程级并行，通过消息传递实现进程间的数据交互和协同控制，具有扩展性强、便于移植等优点，但实现难度较大；OpenMP 基于线程级细粒度并行，通过内存共享，使用显示指导语句，实现应用问题的并行求解，具有实现难度小的优点，但可扩展性差[1]。笔者在实际的科研和教学活动中发现，MPI 并行编程环境在并行求解计算量小、通信量大的应用问题时常常表现出较差的可扩展性，这是由于随着进程规模的增加，并行计算部分花费的时间虽然逐渐减小，但进程间通信所花费的时间却在不断地增加，并逐渐在整个计算时间中占据主导地位。针对这个问题，本文结合 SMP 集群的体系结构特点，开展了基于 MPI + OpenMP 混合并行编程方法的研究，充分发挥 MPI 和 OpenMP 并行编程环境的优点，提高并行程序的效率，并对比分析了 MPI 与 MPI + OpenMP 混合程序的性能优劣，总结了造成性能差异的主要原因。

2. 并行编程模式

2.1. OpenMP (Open Multi-Processing)

OpenMP 支持多线程并发编程和 C、C++、FORTRAN 等多种编程语言，是一种共享存储方式的并行编程标准，可在 Linux、UNIX 等操作系统中运行。OpenMP 采用标准的 Fork/Join 式并行执行模型。OpenMP 通过在代码中加入指导语句来控制代码的并行处理，实际并行过程由底层支持库来实现。OpenMP 基本的结构包括：parallel、工作共享、master 和同步、任务等结构，其中 parallel 结构是最基本的指导语句，所有的 OpenMP 程序都会用到[1] [2]。

2.2. MPI (Message Passing Interface)

MPI 是由工业、科研和政府部门等联合建立的消息传递并行编程标准[1]。MPI 凭借其标准化、可移植、可扩展等优点，成为目前最通用的并行编程标准。MPI1.0 版于 1994 年发布，1998 年 MPI2.0 版本发布，在 MPI2.0 中，共有 287 个调用接口。

2.3. MPI + OpenMP

MPI + OpenMP 混合编程模型通过在节点内使用共享存储模型，节点外采用消息传递模型实现具体

问题的多级并行求解。该模型将 OpenMP 和 MPI 模型有机融合到一起，通过充分发挥两种模型的各自的优点，从而获得更高的计算性能和可扩展性[3] [4] [5] [6]。具体来讲：节点外采用消息传递，解决了节点间 OpenMP 模型计算结果无法交互的问题；节点内采用共享存储，减少进程的数量，降低了进程通信所花费的时间。经测试，该模型对大规模循环问题的并行处理具有较高的并行效率和可扩展性。

3. MPI + OpenMP 混合并行编程的实现

3.1. 算法设计

本文选用圆周率 π 数值积分并行求解算法来分析 MPI + OpenMP 混合编程的具体实现。π 数值求解过程物理意义清晰，其计算规模灵活可调，虽然其并行实现过程虽相对简单，仍不失为用于并行程序计算性能测试分析的良好测例。

求解主要围绕着循环计算各部分梯形面积展开，其算法设计也主要针对循环过程的二级并行划分开展。具体如下：

(1) 节点间任务划分

采用均匀划分或交叉划分方法，将计算区间 S 划分为 np 个小区间 $s_i, S = \bigcup s_i, i = 1, 2, \cdots, np$，若采用交叉划分，每个区间所含的循环次数尽量为节点内核心数的整数倍。

(2) 节点内任务划分

在每个计算节点内调用 OpenMP for 并行制导语句，将所分配的循环分配给不同的线程，期间需要调整每个线程分配的循环次数，尽可能做到负载均衡，求解流程如图 1 所示。

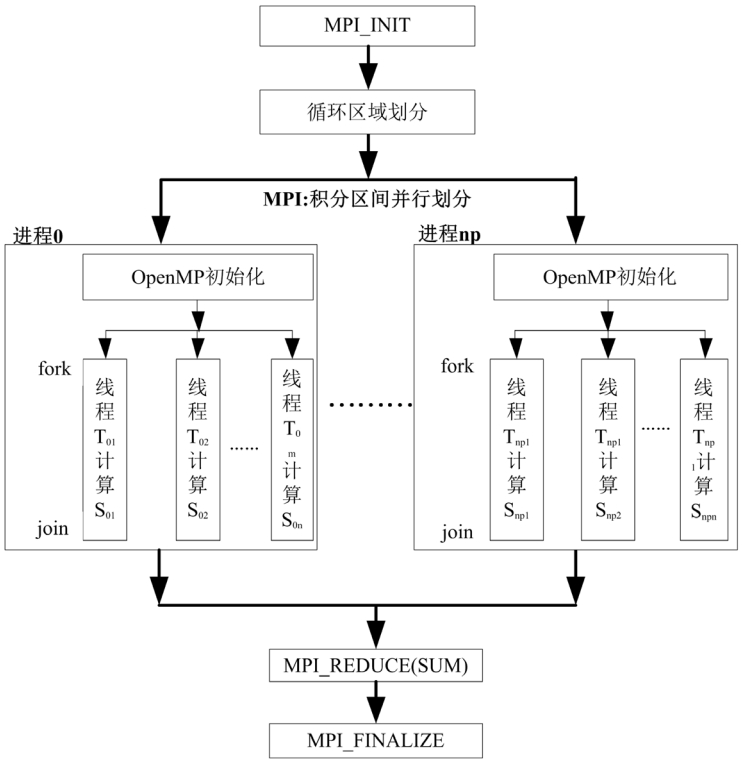

Figure 1. MPI + OPenMP hybrid parallel computation flow of π numerical calculation

图 1. π 值 MPI + OpenMP 混合并行求解流程图

3.2. 算法实现

算法的具体实现过程分为以下几步：

(1) 设定线程数

根据计算节点 CPU 配置，指定每个节点内部的线程数，本文中为 24 个。

(2) 计算区间划分(第一级并行 MPI)

采用均匀划分方法，将循环分配给不同的进程(第一级并行处理)。如下所示：

```
n=N/np;//N 为总循环次数，np 为进程数
for (i=myid*n;i<myid*(n+1),i=++)
                                    {
                                  #pragma omp parallel private(tid)
{
第二级并行;
}
                                      }
```

(3)计算任务划分(第二级并行 OpenMP)

调用 OpenMP for 并行制导语句，将分配的计算区间内的计算任务分配给不同的线程(第二级并行处理)。如下所示：

```
#pragma omp parallel private(tid)
                                {
                                    tid=omp_get_thread_num();
                                    #pragma omp for
                                    for(j=0;j<n;j++)
                                      {
                                          s_j = f(j);
S_i = S_i + s_j;
}
                                      }
```

(4) 数据收集和输出

每个节点分配的计算任务完成后，调用 MPI_REDUCE()，将计算结果收集到进程 0，并将计算结果输出。

4. 性能测试和分析

4.1. 测试环境

本文利用 SMP 集群对程序进行了测试，该集群共 314 个计算节点，节点内 2 颗 Intel Xeon E5-2680 v3 12 核 CPU，集群理论峰值计算能力 310TFlops，Linback 实测计算能力 210TFlops。

4.2. 测试结果

测试分为节点内测试、节点间测试两部分，节点内测试主要开展节点内 MPI、OpenMP 并行程序性能测试，节点间测试主要用于开展跨节点 MPI、MPI + OpenMP 并行程序性能测试。

4.2.1. 节点内测试

表 1 给出了使用一个计算节点，分别用 3、6、12、24 个线程/进程运行 OpenMP、MPI 并行计算程序时间统计情况(每种情况运行 10 次，取计算时间最小值，积分次数 N = 100,000,000,000)。图 2 为表 1 对应的加速比。

Table 1. Parallel computing time of MPI and OpenMP parallel program
表 1. 节点内 OpenMP、MPI 并行程序计算时间

实现方式 \ 核心数	3	6	12	24
OpenMP	94.22	47.12	23.57	11.90
MPI	94.71	47.37	24.78	13.14

备注：单核心只运行一个进程或线程

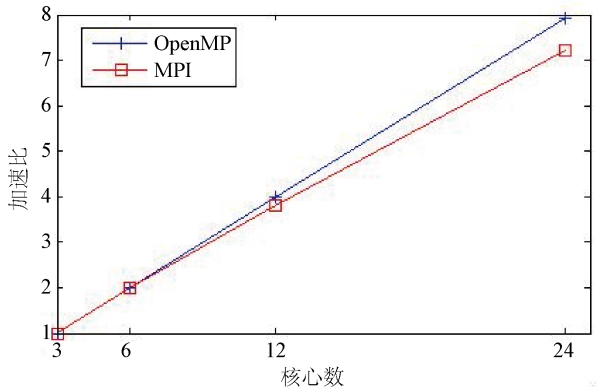

Figure 2. Table 1 acceleration diagram
图 2. 表 1 对应的加速比图

从表 1 和图 2 可以看出，单节点内 OpenMP 与 MPI 并行程序性能基本一致，都具有比较高的并行加速比，OpenMP 并行程序计算性能略高。

4.2.2. 节点间测试

表 2 给出了分别用 1、2、4、8、16、32 个节点运行 π 值 MPI、MPI + OpenMP 并行程序计算运行时间统计情况(积分次数 N = 100,000,000,000)。其中：MPI 程序每个节点分配 24 个进程，MPI + OpenMP 每个节点分配 1 个进程，24 个线程。对于 MPI + OpenMP 程序，核心数 = 节点数 * 单节点核心数；MPI 程序，核心数 = 进程数。图 3 为表 2 对应的加速比。

Table 2. Parallel computing time of MPI + OpenMP and MPI parallel program
表 2. 节点间 MPI + OpenMP、MPI 并行程序计算时间

实现方式 \ 核心数	24	48	96	192	384	768
MPI + OpenMP	11.96	6.25	3.46	2.55	1.49	1.73
MPI	13.14	7.39	4.84	4.18	4.08	7.41

备注：单核心只运行一个进程或线程

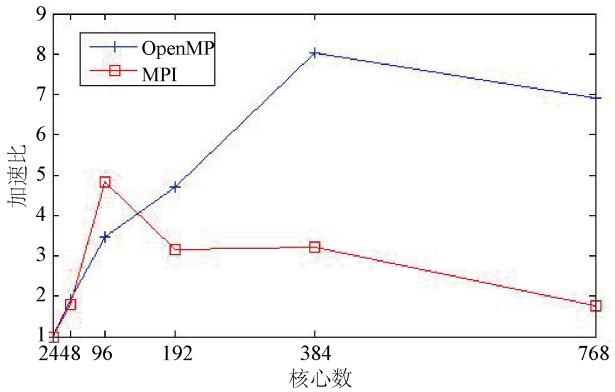

Figure 3. Table 2 acceleration diagram
图 3. 表 2 对应的加速比图

从表 2、图 4 可以看出，MPI + OpenMP 和 MPI 程序加速比均呈现先增加后减小的趋势，但 MPI + OpenMP 程序的加速比随计算规模的增加基本保持稳定，且高于 MPI 程序近 4 倍。因此，与 MPI 并行程序相比，MPI + OpenMP 并行程序具有更快的计算速度和更好的可扩展性。

为了进一步分析两种并行程序计算性能差异的原因，对 MPI + OpenMP 和 MPI 并行程序执行过程中时间的分布按照：并行计算时间、通信时间两部分进行统计，表 3、表 4 分别给出了 MPI 和 MPI + OpenMP 并行程序执行过程中不同部分消耗时间的统计表。图 4、图 5 分别为表 3、表 4 对应的时间分布表。

Table 3. Parallel computing distribute time of MPI parallel program
表 3. MPI 并行程序计算时间分配表

时间分布 \ 进程数	24	48	96	192	384	768
并行计算时间	11.95	6.10	3.14	2.17	1.43	0.64
进程通信时间	1.09	1.29	1.70	2.01	2.65	6.77
合计	13.14	7.39	4.84	4.18	4.08	7.41

备注：单节点启动 24 个进程

从表 3 可以看出，随着计算规模的增加，MPI 程序并行计算部分花费时间由 11.95 s 减少为 0.64 s，降低近 19 倍，占总计算时间的百分比由 91.0%减小为 8.6%；进程间通信消耗的时间由 1.09 s 增加至 6.77 s，提高 6.2 倍，占总计算时间的 9.0%提高至 91.4%。进程间通信开销远高于并行计算所花费的时间，极大的制约了并行效率的提高。

Table 4. Parallel computing distribute time of MPI + OpenMP parallel program
表 4. MPI + OpenMP 并行程序计算时间分配

时间分布 \ 进程数	1	2	4	8	16	32
并行计算时间	11.92	6.10	3.32	2.39	1.28	1.27
进程通信时间	0.04	0.15	0.14	0.16	0.21	0.46
合计	11.96	6.25	3.46	2.55	1.49	1.73

备注：单节点启动 1 个进程，24 个线程

从表 4 可以看出，在同样的计算规模下，MPI + OpenMP 程序并行部分计算花费时间由 11.92 s 减少为 1.27 秒，占总计算时间的百分比由 99.7%减少为 73.4%；进程通信部分消耗的时间由 0.04 s 增加为 0.46 s，占总计算时间的百分比由 0.3%提高至 26.6%。采用 MPI + OpenMP 混合并行程序方法，有效降低了进程间通信的时间开销，大大提高了程序的并行执行效率和可扩展性。

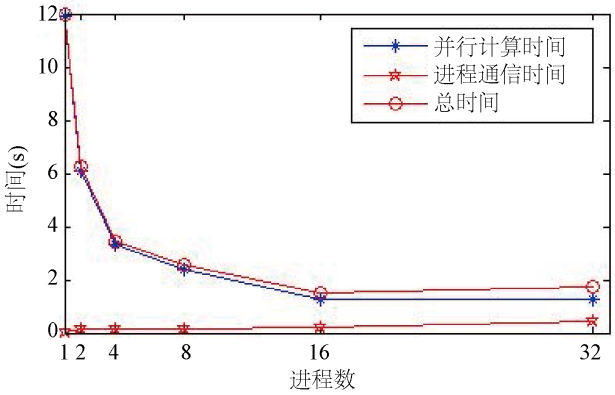

Figure 4. Time distribution of **Table 3**
图 4. 表 3 对应的时间分布图

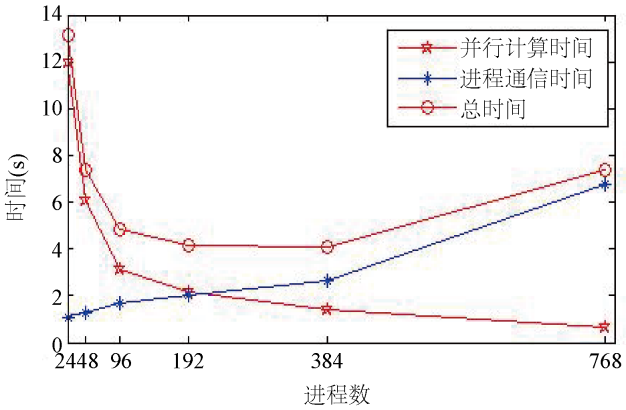

Figure 5. Time distribution of **Table 4**
图 5. 表 4 对应的时间分布图

从表 3、表 4 和图 4、图 5 可以见，随着进程数的增加，MPI + OpenMP 两种并行编程方式下，应用问题并行部分的计算时间均逐渐减小，但 MPI + OpenMP 并行程序的通信消耗时间只占 MPI 并行程序的 6.8%。由此可见，MPI + OpenMP 混合并行程序在很好地继承了两种并行编程环境的优点的同时，可以有效克服两种并行编程环境的缺点，可以有效提高并行程序的效率。

5. 结束语

MPI + OpenMP 混合编程模型通过节点间消息传递，节点内数据共享的方式，将分布式存储系统和共享式存储系统的优点互相结合，既能解决 MPI 并行程序性能随进程数增加而降低的问题，同时又克服了 OpenMP 并行程序扩展性差的问题，可有效地发挥 SMP 集群的计算性能。并行程序的性能受并行机体系结构、编程环境、问题自身结构、并行程序的优化、编程人员的习惯等多重因素制约，很难给出统一的编程和性能评价标准，需结合具体的应用问题，合理地采取编程和优化方式，系统地进行设计。

基金项目

本文基金资助项目为国防科技创新特区项目(18-H863-05-ZT-001-012-06)；水中军用目标特性国防科技重点实验室基金(614240704040317)。

参考文献

[1] 迟学斌, 王彦棡, 王钰, 等. 并行计算与实现技术[M]. 北京: 科学出版社, 2015.

[2] 罗秋明, 明仲, 刘刚, 等. OpenMP 编译原理及实现技术[M]. 北京: 清华大学出版社, 2012.

[3] 朱昶胜, 邓新, 冯力, 李浩, 等. MPI + OpenMP 环境下的二元合金三维相场模型的并行方法[J]. 兰州理工大学学报, 2017, 43(4): 16-22.

[4] 杨书博, 乔先孝, 车小花. 基于 MPI + OpenMP 的三维弹性波方程混合并行有限差分算法[J]. 应用声学, 2018, 31(1): 75-83.

[5] 唐兵, Laurent BOBELIN, 贺海武. 基于 MPI 和 OpenMP 混合编程的非负矩阵分解并行算法[J]. 计算机科学, 2017, 44(3): 51-55.

[6] 刘超, 祝永志. 多核 SMP 集群混合并行编程技术的研究[J]. 微型机与应用, 2017, 36(4): 18-21.