基于聚类分析的进程拓扑映射优化

王 涛[1)] 卿 鹏[1)] 魏 迪[1)] 漆锋滨[2)]

[1)](江南计算技术研究所 江苏 无锡 214083)

[2)](国家并行计算机工程技术研究中心 北京 100080)

摘 要 高性能计算机系统规模的持续增大使通信墙问题越来越突出.逻辑进程与物理拓扑的映射优化方法能够提高应用的通信效率,已经成为高性能计算的研究热点之一.传统的进程映射优化模型由于映射粒度过细,导致映射效率低,且易破坏通信密集的进程簇的整体性.为此,文中提出了一种聚合的二次分配问题(Aggregated Quadratic Assignment Problem,AQAP)模型,并以 AQAP 模型为指导,提出了一种新颖的基于聚类分析的进程映射优化方法.该方法首先使用谱聚类算法对进程通信模式进行聚类分析,然后采用自适应聚合进程映射策略实现进程簇到物理拓扑的映射,最后使用聚合 Pair-Exchange 算法对进程簇映射进行进一步优化.文中提出的优化方法首次将谱聚类分析应用于进程映射问题,可以有效减少远距离通信,增强通信的局部性.NPB 基准程序及两道实际应用的实验结果表明,文中提出的进程映射优化方法可以使程序获得明显的性能提升,优于现有的基于 Pair-Exchange 以及基于图划分的进程映射方法.

关键词 通信模式;物理拓扑;进程映射;谱聚类;MPI

中图法分类号 TP319 **DOI 号** 10.3724/SP.J.1016.2015.01044

Optimization of Process-to-Core Mapping Based on Clustering Analysis

WANG Tao[1)] QING Peng[1)] WEI Di[1)] QI Feng-Bin[2)]

[1)](*Jiangnan Institute of Computing Technology*, *Wuxi*, *Jiangsu* 214083)

[2)](*National Research Center of Parallel Computer Engineering & Technology*, *Beijing* 100080)

Abstract With the increase of the scale of high-performance computers, the communication wall problem is becoming increasingly severe. Optimizing for process-to-core mapping can help improving applications' communication efficiency. Due to the over fine mapping grain, the traditional process mapping optimization model leads to low mapping efficiency and tends to split the process clusters within which communication is dense. To settle this problem, we propose an aggregated quadratic assignment problem (AQAP) model. Guided by AQAP model, we propose a novel process mapping optimization method based on clustering analysis. In this method, we first use spectral clustering algorithm to analyze process communication pattern, followed by mapping the process clusters to physical topology using self-adaption aggregated process mapping strategy, and finally optimizing the mapping result further by using aggregated Pair-Exchange algorithm. To our knowledge, this is the first instance where the spectral clustering algorithm has been applied to solve the process placement problem. Our method can effectively reduce long-distance communications as well as enhance the communication locality. We evaluated the performance of our method with the NPB benchmarks and two practical applications. Experimental

收稿日期:2013-05-18;最终修改稿收到日期:2014-10-24.本课题得到国家"八六三"高技术研究发展计划项目基金(2012AA010903)资助.**王 涛**,男,1989 年生,硕士研究生,中国计算机学会(CCF)会员,主要研究方向为并行编译、高性能计算.E-mail:wangtao.waves@gmail.com.**卿 鹏**,男,1979 年生,硕士,主要研究方向为运行时系统.**魏 迪**,男,1984 年生,硕士,主要研究方向为运行时系统.**漆锋滨**,男,1966 年生,研究员,博士生导师,主要研究领域为编译技术、高性能计算.

results show that the optimized process placement generated by our method can achieve significant performance improvement, and outperform existing Pair-Exchange-based and Graph Partition-based methods.

Keywords　communication pattern; physical topology; process-to-core mapping; spectral clustering; Message Passing Interface (MPI)

1　引　言

随着高性能计算机系统并行规模尤其是片内并行度的持续增大，通信墙问题变得越来越突出.片内处理器核心数量的增多使得单芯片计算能力有了跨越式发展，但增长缓慢的网络端口数量和日渐增大的网络直径使得通信能力成为限制程序性能的瓶颈.已有研究[1-3]表明，逻辑进程与物理拓扑的映射对于程序性能有着重要影响.合理的进程-物理拓扑映射可以使更多的进程间通信控制在通信效率更高的局部范围内(如节点内)，提高通信效率并减少拥塞.这对于具有高片内并行度和长网络直径的高性能计算机系统尤为重要.

进程拓扑映射问题属于组合优化范畴，已经被证明是一个 NP 难问题[4-5]，因此无法在合理时间内给出实际规模并行程序的最优进程拓扑映射.传统的进程映射优化模型以进程作为映射粒度，由于映射粒度过细，导致搜索最优映射效率偏低，且可能破坏通信密集的进程簇的整体性.针对上述问题，本文提出一种聚合的二次分配问题(AQAP)模型，该模型使用聚合的进程簇作为最优映射的搜索粒度，不仅能够提高最优解搜索效率，而且可以避免进程簇整体性遭到破坏.

以传统进程映射优化模型为指导，研究人员提出了多种启发式算法用以求解进程拓扑映射问题的近似最优解，包括基于对交换(Pair-Exchange[6])以及基于图划分的一系列启发式算法[7-11].然而上述大多数算法存在易收敛于较差的局部最优解或搜索效率低等问题.为此，本文以 AQAP 模型为指导提出一种新颖的基于聚类分析的进程拓扑映射优化方法.与传统方法不同，本文方法以进程簇为搜索粒度求解优化映射，因此进程簇内部的通信密集程度对于程序通信的局部性至关重要，进而对程序性能产生直接影响.我们选用谱聚类算法对进程进行分簇，是因为谱聚类算法具有以下特点：

(1)对进程进行谱聚类分析可以得到若干簇内通信量最大化、簇间通信量最小化的进程簇，符合进程簇内部通信密集化的根本需求.

(2)谱聚类是建立在谱理论基础上的新兴聚类算法，通过使用特征向量完成对原始数据的降维分析处理.因此谱聚类算法具有收敛于全局最优解且计算复杂度低的突出优点[12].以上特点可以避免本文优化方法陷入较差的局部最优解，并且可以高效处理大规模数据.

(3)通信模式矩阵可以满足谱聚类对数据的要求.而其他聚类算法(如 K-means 算法)一般要求数据必须是多维欧氏空间中的向量，而这是通信模式无法提供的.

在进程分簇的基础上，本文设计了朴素映射、First-Fit 和 Most Reservation 三种进程簇映射策略.进程簇映射模块根据聚类分析结果的均衡性和进程簇数量的差异，自适应地选择上述三种策略进行进程簇映射.最后，本文通过在进程簇映射结果的基础上复用已有的启发式算法，进一步优化映射结果.

本文的主要贡献总结如下：

(1)针对现有的进程映射模型粒度过细的问题，提出了聚合的二次分配问题 AQAP 模型.

(2)首次在进程映射优化问题中引入谱聚类分析，提出了通信模式正规化谱聚类算法(CP-NSC 算法)，利用谱聚类算法效率高、处理数据能力强、收敛于全局最优解、聚类结果均衡等特点解决进程分簇问题.

(3)针对进程簇映射问题，提出了自适应聚合进程映射策略.策略根据进程聚类结果的均衡性和进程簇数量，自适应地在朴素映射、First-Fit 以及 Most Reservation 三种进程簇映射算法中做出选择.另外，本文还提出了适用于 AQAP 模型的聚合 Pair-Exchange 算法(APE 算法)，对进程簇映射作进一步优化.

本文第 2 节是相关工作介绍；第 3 节介绍进程拓扑映射问题的模型和方法，并提出改进的聚合模型；第 4 节介绍基于聚类分析的进程映射优化的设

计与实现;第 5 节给出基准测试程序和实际应用的实验结果,对本文提出的方法进行验证;最后得出结论,并对本文的内容进行总结和展望.

2 相关工作

众多学者对进程拓扑映射优化问题做了多方位的研究.共同的指导思想是通过将进程重新映射到底层物理拓扑,使得总通信开销最小.由于 MPI① 已经成为消息传递编程模型事实上的工业标准,在 MPI 内部实现进程拓扑优化成为一种常用的思路.文献[13-14]研究了在 MPI 中通过修改 MPI 拓扑相关函数实现虚拟拓扑到物理拓扑的映射.但是此类方法需要调用 MPI 接口实现映射优化功能,这对于源码复杂的程序是不易用的.

Chen 等人在文献[8]中首次提出了一种全自动化的进程映射方案 MPIPP,该方案不再依靠用户知识给出通信模式和物理拓扑,而是通过 MPI 通信模式采集工具以及物理拓扑信息探测工具自动化获取.MPIPP 还提出了一种比以往方法更有效的映射优化算法.然而 MPIPP 只考虑了点对点通信.Zhang 等人[11]的工作对此进行了完善,他们提出了一种针对集合通信的进程映射优化方法,称为 OPP.OPP 方法通过将集合通信转换为等效的点对点通信实现对集合通信信息的自动采集,然后使用已有的进程映射优化算法实现进程到物理拓扑的映射.

Bhatele 等人在文献[1,3]中提出了一个基于多种启发式算法的并行程序自动映射框架.该框架通过程序剖面技术获取通信模式,然后分析通信模式并将其归类,最后根据通信模式的种类动态选择不同的启发式算法寻找最优进程拓扑映射.该研究工作总结并集成了众多传统的启发式映射算法,主要针对 IBM Blue Gene 等具有环网(torus)结构的计算机系统.

基于开源的图划分软件实现进程重新映射是近年来的研究热点,常用的开源图划分软件包有 METIS[15]、Scotch[16] 和 Jostle[17] 等.文献[7]中首先使用与文献[8]相似的方法追踪程序进程间通信量,不同的是后者在后续处理中使用了 Scotch 图划分软件获得通信图到物理拓扑图的优化映射.其后,Mercier 和 Jeannot 在文献[9]中提出了 TreeMatch 算法,用于解决 NUMA 体系结构下的 MPI 进程近似最优映射.该算法采用了与本文类似的思想,首先将进程按照通信亲和性分组,然后以进程组为单位进行图匹配,与本文不同,TreeMatch 使用的进程分组算法 GroupProcesses 采取贪心策略寻找独立进程集,并且算法的复杂度较高.文献[18]提出了 GroupProcesses 算法的快速版本,但是需要借助用户指定切换阈值,增加了用户参与程度,并且沿用了分组的贪心策略.

Subramoni 等人在文献[10]中提出了一种针对 IB 网络的拓扑感知进程映射服务.使用 Neighbor-Joining 算法在用户级发掘 IB 网络路由和交换信息,并使用 Jostle 等图划分工具完成进程图到网络拓扑图的映射优化.与本文不同,该工作是通过修改 MPI 开源实现 MVAPICH2② 实现的,并且更侧重于底层网络拓扑的获取和抽象.

von Alfthan 等人[19]的工作则更侧重于解决不规则物理拓扑上的进程映射优化问题.该工作以进程作为映射粒度,针对高性能计算系统的可用资源不连续的实际情景,使用标准模拟退火算法寻找最优映射,并借助并行化加快搜索速度.

Leung 等人[20]的工作同样针对非连续分配任务的计算机系统(如 Cray X 系列).该工作提出了一组任务映射算法解决具有筛网(stencil)通信模式的并行程序在 Cray XE 系统上的任务映射问题.聂鹏程等人在文献[21]中提出了一种自适应任务映射算法,该算法首先通过分析任务运行时的平均停驻时间得出任务的计算需求然后根据需求以及各 CPU 核的负载情况将任务映射到合适的 CPU 核上运行.

Brandfass 等人在文献[6]中形式化地总结了进程重新映射的数学模型,评估了已有启发式算法的复杂度,并对基于 Pair-Exchange 的启发式算法进行了优化,降低了其计算量.与文献[19]类似,该研究工作仍然以进程作为映射粒度.

本文提出了一种聚合的二次分配问题模型 AQAP,并以 AQAP 模型为指导,提出了一种基于聚类分析的进程簇拓扑映射优化方法,着重解决传统模型和方法的低效、易破坏进程簇整体性及易陷入较差局部最近解等问题.

3 模型和方法

在消息传递模型中,两个进程间一次独立的通

① MPI Documents. http://www.mpi-forum.org
② MVAPICH2. http://mvapich.cse.ohio-state.edu/

信行为的时间开销取决于两部分：消息的长度和通信双方所在的物理位置. 因此一个合理的通信开销模型必须将通信模式和物理拓扑考虑在内.

3.1　QAP 模型

已有的研究[2,6,19]为进程映射优化问题建立了如下模型. 对于一个拥有 n 个进程、运行在 n 个 CPU 核心上的应用程序，定义 $\boldsymbol{A}\in\boldsymbol{R}^{n\times n}$ 为通信模式矩阵，其中元素 a_{ij} 表示进程 i 发送至进程 j 的通信量；定义 $\boldsymbol{B}\in\boldsymbol{R}^{n\times n}$ 为拓扑距离矩阵，其中元素 b_{ij} 表示 CPU 核心 i 和 CPU 核心 j 之间的距离. 至此寻找 MPI 进程到物理拓扑的最优映射可以形式化为如下二次分配问题(Quadratic Assignment Problem，QAP).

寻找一种进程-CPU 核心间的一对一映射 π，使得式(1)总体通信开销最小化.

$$Z(\boldsymbol{A},\boldsymbol{B},\pi)=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}a_{ij}\cdot b_{\pi(i)\pi(j)} \tag{1}$$

每一种映射 π 即是 QAP 问题的一个解，一般用序列$\{0,1,2,\cdots,n-1\}$的一种排列来表示，其含义是将进程 i 映射到 CPU 核心 $\pi(i)$ 上.

在过去的数十年中，尽管吸引了大量的研究，QAP 问题仍然是最难解决的优化问题之一，甚至当 n 很小时(不大于 50)就无法在合理的时间内计算出准确的最优解. 事实上，Sahni 和 Gonzalez[4] 已经证明 QAP 问题是 NP 难问题，因此提高启发式算法的效率和准确性对于进程映射优化问题至关重要.

3.2　AQAP 模型

在过去长时期内，以 QAP 模型为指导的多种启发式算法取得了良好的进程映射效果. 然而随着程序和系统规模的快速增长，QAP 模型粒度过细的缺陷越来越明显，导致最优解搜索效率低、易破坏进程簇整体性和易陷入较差局部最优解等问题.

针对上述问题，我们提出了如下聚合 QAP 模型(Aggregated QAP，AQAP).

寻找一种进程簇-CPU 核心组之间的映射 Π(注意到由 Π 细化即可得到映射 π)，使得式(2)总体通信开销最小化.

$$Z(\boldsymbol{A},\boldsymbol{B},\Pi)=\sum_{i=0}^{k-1}\sum_{j=0}^{k-1}Cost_{ij} \tag{2}$$

其中 k 表示进程簇的数量，$Cost_{ij}$ 表示进程簇 i 中各进程与进程簇 j 中各进程之间通信的总开销. 一般地，$Cost_{ij}$ 可由式(3)计算得到.

$$Cost_{ij}=\sum_{s=0}^{n_i-1}\sum_{t=0}^{n_j-1}a_{is,jt}\cdot b_{\pi(is),\pi(jt)}+\sum_{t=0}^{n_j-1}\sum_{s=0}^{n_i-1}a_{jt,is}\cdot b_{\pi(jt),\pi(is)} \tag{3}$$

其中 n_i 和 n_j 分别表示进程簇 i 和进程簇 j 中的进程数量，等式右边两项分别表示进程簇 i 中各进程向进程簇 j 中各进程发送消息的开销以及反方向发送消息的开销.

与传统 QAP 模型相比，AQAP 模型有如下特点：

(1)求解 AQAP 仍然是 NP 难问题，但是搜索最优解的效率大大提高. 在求解过程中待映射的对象数量由进程数 n 变为进程簇数 k(通常 $k\leq n/2$)，搜索空间显著缩小.

(2)求解过程不会破坏进程簇的整体性. 在初始映射时保证绝大多数进程簇映射到邻近连续核心上，在进行映射的重新调整时，以进程簇为单位进行重新映射，因此重新映射后进程簇还是映射到邻近连续核心上，其整体性得到保留.

(3)以 AQAP 为指导的进程重新映射的质量十分依赖于进程聚类的准确性. 由于特点(2)，具有良好通信局部性的进程簇得以映射到拓扑局部性的核心上，但若聚类结果较差(进程簇内部通信不够密集)，这种耦合度差的进程簇也将作为整体出现在最终的进程映射结果中.

3.3　通信模式矩阵和拓扑距离矩阵模型

通信模式矩阵须要反映出并行程序进程间的通信需求，通信需求高的进程映射到邻近的物理位置上可以获得较高的通信效率. 已有研究中使用了多种度量标准对通信模式建模，比如通信量、通信频率以及通信量和通信频率的组合等.

拓扑距离矩阵须反映的物理意义是 CPU 核心间传输单位消息所耗的时间，即 LogP 模型[22] 中的 gap，因此通信量与拓扑距离的乘积反映的物理意义便是通信耗时. 而通信量和通信频率的组合量与拓扑距离的乘积反而失去了原有的物理意义，因此我们选用通信量作为通信模式的度量标准.

拓扑距离矩阵本质上须要反映出 CPU 核心间通信带宽的差别. 根据目前高性能计算机典型的层次化拓扑结构，我们定义节点内、交换机内和交换机间 CPU 核心单位消息传输耗时分别为 $t_{\text{intra-node}}$、$t_{\text{intra-switch}}$ 和 $t_{\text{inter-switch}}$. 因此构建拓扑距离矩阵 $\boldsymbol{B}$ 的模型如下：

$$b_{ij}=\begin{cases}t_{\text{intra-node}}, & \text{核心 } i,j \text{ 属于同一节点}\\ t_{\text{intra-switch}}, & \text{核心 } i,j \text{ 属于同一交换机下不同节点.}\\ t_{\text{inter-switch}}, & \text{核心 } i,j \text{ 属于不同交换机}\end{cases}$$

$t_{\text{intra-node}}$、$t_{\text{intra-switch}}$ 和 $t_{\text{inter-switch}}$ 的取值需要实测或网络参数的指导.

4 基于聚类分析的进程映射优化

以 AQAP 模型为指导，设计基于聚类分析的进程映射优化框架如图 1 所示.

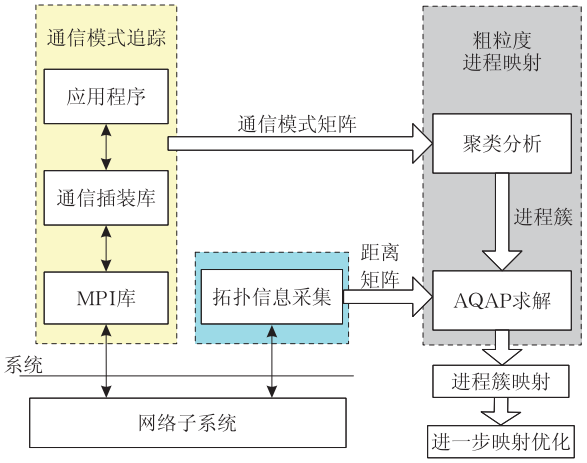


图 1 基于聚类分析的进程映射优化框架

根据应用需求，本文进程映射优化方法分为单层映射和层次化指定映射两类. 单层映射将进程直接映射到节点内，适用于扁平式拓扑结构的计算环境. 层次化指定映射将进程映射到指定的聚合节点上，适用于大规模层次化拓扑结构的计算系统. 4.1～4.4 小节和 4.5 小节将分别阐述基于聚类分析的单层映射以及层次化指定映射的设计与实现.

4.1 轻量级通信模式追踪

多种功能复杂的工具可以对 MPI 程序进行追踪(tracing)和分析，比如 MPICH2 自带的分析工具 MPE. 然而使用这些工具进行通信模式采集存在如下问题：(1) 因为追踪的事件繁多，追踪文件的体积往往过于巨大；(2) 采集到的信息不能满足需要，如 MPE 对通信量的采集没有考虑集合通信.

基于上述原因我们开发了专用的轻量级通信插装库，通过重新定义 MPI 点对点通信和集合通信接口，覆盖 MPI 原有的弱引用接口，完成通信模式所需数据的采集. 考虑到数据的通用性，我们只采集了 MPI 进程间的用户数据传递信息，而由 MPI 内部协议实现引起的额外通信我们没有考虑在内. 我们的轻量级通信插装库还针对性地采集了通信次数等信息. 图 2 是采集到的通信模式示例，矩阵表示一个 NPB 程序(cg.W.8)进程间发送的消息数据量.

4.2 拓扑信息仿真采集

拓扑距离矩阵描述了集群的互连层次，为了根据 3.3 节模型构建拓扑距离矩阵 $\boldsymbol{B}$，需要判断给定

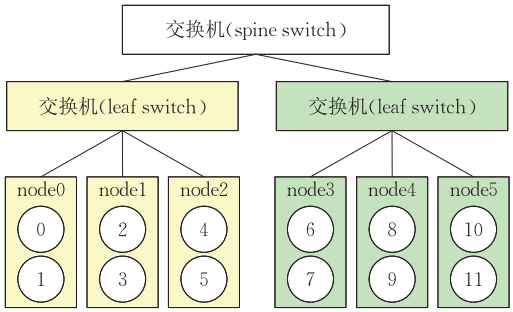
$$\boldsymbol{A}=\begin{pmatrix}
5824 & 5830.9 & 5830.9 & 0 & 0 & 0 & 0 & 0\\
5830.9 & 5824 & 0 & 5830.9 & 0 & 0 & 0 & 0\\
5830.9 & 0 & 0 & 5830.9 & 5824 & 0 & 0 & 0\\
0.008 & 5830.9 & 5830.9 & 0 & 0 & 5824 & 0 & 0\\
0.008 & 0 & 5824 & 0 & 0 & 5830.9 & 5830.9 & 0\\
0.008 & 0 & 0 & 5824 & 5830.9 & 0 & 0 & 5830.9\\
0.008 & 0 & 0 & 0 & 5830.9 & 0 & 5824 & 5830.9\\
0.008 & 0 & 0 & 0 & 0 & 5830.9 & 5830.9 & 5824
\end{pmatrix}$$

图 2 NPB cg.W.8 程序通信模式矩阵(a_{ij} 代表进程 i 发送至进程 j 的通信量，单位：KB)

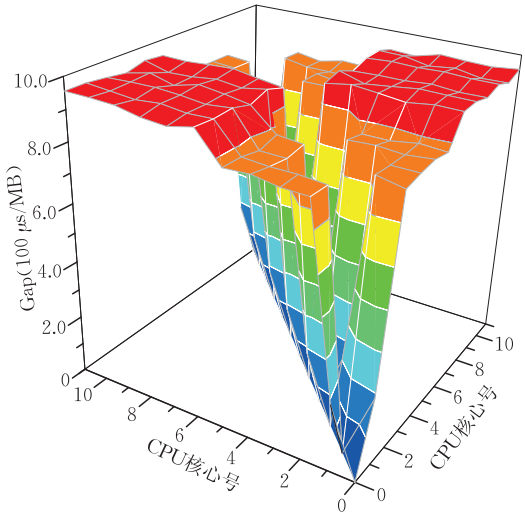
的核心之间的关系和距离(即 gap，取值为通信带宽的倒数). 这些参数可以由系统配置文件或通过实验测试获取.

本文采用 single-circle-match 算法[8]测量拓扑距离，该算法由 n 次 ping-pong 测试组成(n 为 CPU 核心数)，在每轮迭代中，$n/2$ 对核心同时交换 ping-pong 消息. 该方法模拟了进程间同时进行数据传输的真实程序. 另外，为了避免并行执行中可能出现的异常通信拥塞对结果准确性的影响，gap 值最高的 10% 的数据会被忽略掉. 图 3 展示了一套示例测试集群及其 gap 拓扑图，根据图 3(b)的测试结果，我们设置该集群的 $t_{\text{intra-node}}:t_{\text{intra-switch}}:t_{\text{inter-switch}}=1:3.7:4.1$.

相比于使用经验值(如文献[6])或配置文件参



(a) 示例测试集群



(b) 示例测试集群的gap拓扑图

图 3 示例测试集群及其 gap 拓扑图

数构建距离矩阵，上述方法可以准确反映程序实际运行时的有效带宽，而配置值无法反映运行时的拥塞等状态.除此之外，上述方法中所有进程同时参与，具有高度的可扩展性，适用于大规模系统.

4.3　基于 CP-NSC 算法的进程聚类分析

本小节介绍进程聚类分析的设计，为保证进程映射的整体效果和效率，对于聚类分析算法有如下要求：(1)得到的进程簇内部通信尽可能密集，进程簇间通信尽可能稀疏；(2)分簇结果尽可能均衡；(3)聚类过程高效，具有处理大规模进程数据的能力.

综合分析上述要求和各类聚类算法的特点，我们选用正规化谱聚类算法[23](Normalized Spectral Clustering，NSC)对进程进行聚类分析.然而原生的谱聚类算法处理的对象是具有多维属性的点数据，无法直接处理通信拓扑矩阵，例如文献[24]提出的正规化谱聚类的并行算法.

算法 1 给出了我们设计的处理进程聚类的 NSC 算法版本，称为通信模式 NSC 算法(Communication Pattern NSC，CP-NSC)，这也是改进的谱聚类算法首次应用于进程映射优化问题.CP-NSC 算法的核心思想是对问题的不断转化：第 1～2 步将根据通信模式进行聚类的问题转化为根据相似性进行聚类的问题；第 3～5 步利用 Laplacian 矩阵性质和特征向量将相似性矩阵转化为正规化特征向量矩阵，实现了对问题的降维转化；最后将原问题转化成低维度聚类问题并利用 K-means 算法进行聚类，其中聚类簇数 k 对于进程映射的效果有重要影响，其最优值与通信模式及节点数等因素相关.一般地，k 值应不小于节点数量，且应随着通信模式不规则程度的加剧而增大.根据相似性矩阵的定义，我们在 CP-NSC 算法中引入两个常量 $SELF\_SIMILARITY$ 和 $MAX\_SIMILARITY$，分别用以表示自身相似度(一般设置为 1)和非自身最大相似度.

算法 1.　CP-NSC 算法.

输入：进程通信模式矩阵 $\boldsymbol{A}$，进程数 n，聚类簇数 k

输出：k 个进程簇 $cluster(i)$，$i=0,1,\cdots,k-1$

1. 根据矩阵 $\boldsymbol{A}$ 计算通信量矩阵 $\boldsymbol{A}'$：$\boldsymbol{A}'=\boldsymbol{A}+\boldsymbol{A}^{\mathrm{T}}$
2. 根据矩阵 $\boldsymbol{A}'$ 计算相似性矩阵 $\boldsymbol{S}$

```
set maxVolm = max{a'_ij | a'_ij ∈ A'}
FOREACH s_ij ∈ S DO
  IF i==j THEN
    s_ij = SELF_SIMILARITY
  ELSE
    s_ij = a'_ij / maxVolm * MAX_SIMILARITY
  END
END
```

3. 计算 Laplacian 矩阵 $\boldsymbol{L}$

set $d_{ii}=\sum_{j=0}^{n-1}s_{ij}$

$\boldsymbol{L}=\boldsymbol{I}-\boldsymbol{D}^{-1/2}\boldsymbol{S}\boldsymbol{D}^{-1/2}$

4. 计算矩阵 $\boldsymbol{L}$ 的前 k 个特征向量，并使用特征向量构造矩阵 $\boldsymbol{V}$

$\boldsymbol{V}=[v_1,\cdots,v_k]\in\boldsymbol{R}^{n\times k}$

5. 计算矩阵 $\boldsymbol{V}$ 的正规化矩阵 $\boldsymbol{U}$

FOR $i=0$ TO $n-1$，$j=0$ TO $k-1$ DO

$$u_{ij}=v_{ij}\Big/\sqrt{\sum_{r=0}^{k-1}v_{ir}^2}$$

END

6. 使用 K-means 算法对矩阵 $\boldsymbol{U}$ 的 n 行进行聚类，得到 k 簇，并还原到原问题进程分簇的解.

原生 NSC 算法的性能瓶颈在于由多维属性点数据构建相似性矩阵，文献[24-25]详细分析了 NSC 算法的时空复杂度，并指出实际聚类的时间复杂度相对于构建相似性矩阵“几乎是可忽略的”.而在 CP-NSC 算法中，构建相似性矩阵的时间复杂度降低为 $O(n^2)$，低于 K-means 算法的时间复杂度 $(O(nk^2)\times t$，k 为分簇数量，一般为$\sqrt{n}$量级，t 为迭代次数)，相比于原生 NSC 算法则时间开销几乎可忽略.

4.4　自适应聚合进程映射策略及优化

基于 CP-NSC 算法的进程聚类分析结果是 k 个大小不等的进程簇，本小节目标是将上述 k 个进程簇映射到 N 个节点的 n 个 CPU 核心上，并使得进程间通信尽可能多地集中在节点内.

4.4.1　MPI 默认进程映射

$MPI_Init()$会创建默认的逻辑进程-物理核心映射.MPI 的实现一般会提供以下 3 种映射方式：

(1) Block 方式.进程依次映射到按序排列的所有节点的所有核心上.

(2) Round-robin 方式.进程被循环映射到节点列表中的下一个节点.

(3) Custom 方式.映射由配置文件给出.

图 4 展示了 3 种进程分簇情景下不同进程映射策略的映射结果，其中(a1)、(a2)分别表示 Round-Robin 方式和 Block 方式的映射结果.从映射结果可以看出默认的进程映射方式没有考虑进程簇的内部密集的通信需求.

4.4.2　自适应进程簇映射策略

虽然谱聚类算法倾向于产生元素个数相近的进程簇，但是通信模式的特点各异，使得聚类算法并不能保证聚类结果总是均衡的，并且不同应用情景下进

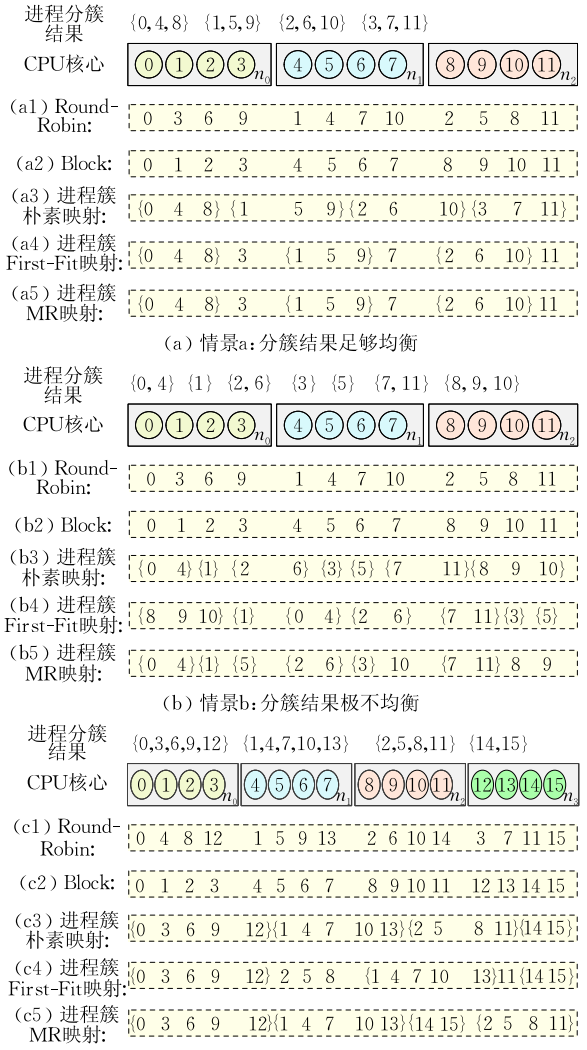


图 4 不同进程映射策略映射结果示例

程簇的数量差异巨大.为了适用多样的应用情景,我们设计了朴素映射、First-Fit 以及 Most Reservation 三种进程簇映射策略,在聚合进程映射时我们的方法会根据进程聚类结果均衡性和进程簇数量的差异自适应地在以下 3 种策略中做出选择.

(1) 朴素映射策略(Plain Scheme).我们将进程簇中进程号最小的进程称为进程簇首进程.在每次迭代中,该策略按照首进程由小到大的顺序依次选择下一个进程簇,然后将该进程簇中的进程顺序映射到 CPU 核心上.使用该策略的映射结果如图 4(a3)、(b3)和(c3)所示.

(2) First-Fit 策略(FF Scheme).如算法 2 所示.首先对进程簇按照所含进程数量由多至少进行稳定排序,排序的目的是在后续的映射中优先考虑体积较大的进程簇.然后按照顺序依次将进程簇进行映射.在映射的过程中维护一张空闲表 $idleTable[N]$,记录每个节点中空闲核心的数量.在进程簇映射时通过查询 $idleTable$ 获得首个可容纳该进程簇的位置进行映射.若 $idleTable$ 中已经没有足够大的空闲块容纳该进程簇(该情况较少发生),将该进程簇拆分并映射到 $idleTable$ 中仍旧空闲的位置.注意 $idleTable[i]$初始值是 MAX,表示空闲的节点总是可以容纳任一进程簇.使用该策略的映射结果如图 4(a4)、(b4)和(c4)所示.

算法 2. 进程簇 First-Fit 算法.

输入:进程数量 n,节点数量 N,k 个进程簇 $cluster(i)$,$i=0,1,\cdots,k-1$

输出:映射 π:进程→CPU 核心

1. 对进程簇按照进程数量由大到小进行稳定排序 $sort(cluster,k)$
2. 初始化空闲表
   FOR $i=0$ TO $N-1$ DO
     $idleTable[i]=$MAX
   END
3. 进程簇 First-Fit 映射
   FOR $i=0$ TO $k-1$ DO
     IF 找到首个可容纳 $cluster(i)$的位置 pos THEN
       将 $cluster(i)$映射到位置 pos
     ELSE //已无可容纳的空闲块
       将 $cluster(i)$拆分并按顺序映射到 $idleTable$ 中仍空闲的位置
     END
     更新 $idleTable$
   END

(3) Most Reservation 策略(MR Scheme).如算法 3 所示.顾名思义,该策略倾向于保留尽量多的完整节点,与 First-Fit 策略类似,MR 策略同样需要在映射的过程中维护一张空闲表 $idleTable[N]$,而与 First-Fit 策略不同,MR 策略不再需要对进程簇按照体积进行排序.具体过程如下.首先初始化 $idleTable$,使得空闲节点可以容纳任意体积的进程簇.然后依次为每一个进程簇 $cluster(i)$寻找首个可以容纳该进程簇的位置 pos.如果可以找到符合条件的 pos 则将 $cluster(i)$拟映射到 pos 并进行前移条件检查,即如果满足以下条件:

条件 1. pos 之前还有空闲的邻居核心.

条件 2. 若将 $cluster(i)$前移至占用 pos 前的所有空闲邻居可以增加空闲节点的数量.

则将 $cluster(i)$映射到 pos 前首个空闲的位置.如此可以保留尽量多的完整空闲节点.使用该策略的映射结果如图 4(a5)、(b5)和(c5)所示.

算法 3. 进程簇 Most Reservation 算法.

输入:进程数量 n,节点数量 N,k 个进程簇 $cluster(i)$,$i=0,1,\cdots,k-1$

输出：映射 π：进程→CPU 核心

1. 初始化空闲表
   FOR $i=0$ TO $N-1$ DO
     $idleTable[i]=$MAX
   END
2. 进程簇 Most Reservation 映射
   FOR $i=0$ TO $k-1$ DO
     IF 找到首个可容纳 $cluster(i)$的位置 pos THEN
       IF ($preIdle(pos)$ && $moreReserve(pos)$) THEN
         将 $cluster(i)$映射到 pos 前首个空闲的位置
       ELSE
         将 $cluster(i)$映射到位置 pos
       END
     ELSE　　//已无可容纳的空闲块
       将 $cluster(i)$拆分并按顺序映射到 $idleTable$ 中仍空闲的位置
     END
     更新 $idleTable$
   END

朴素映射策略过程最简单，开销最小，适用于进程簇体积均衡的情景（图 4 情景 a），该情景下进程簇所含元素数量几无差异，不存在琐碎的进程簇（本文中称为噪音），在这种情景下，First-Fit 和 Most Reservation 策略会将某些进程簇分拆映射到节点的碎片上，而朴素映射策略则可以避免这种情况. 而当进程簇的体积差异巨大且噪音较多时，我们采用 First-Fit 映射策略，因为 First-Fit 策略会首先保证大块体积的进程簇的优先映射，而其他策略则可能破坏大块进程簇的完整性，如图 4 情景 b 所示. 然而由于需要对进程簇进行排序，First-Fit 策略的开销最大，当分簇均衡性介于上述两种极端情景且进程簇数量庞大时，Most Reservation 映射策略能够避免由排序引起的高开销，并且取得更优的映射效果，如图 4 情景 c 所示. 在此情境下，使用 MR 策略使得更少的进程簇分跨在多个节点上.

根据上述分析我们设计自适应选择规则如下：

$$scheme=\begin{cases}Plain, & 若\ stdev(clusters)<T_l\\ FF, & 若\ stdev(clusters)>T_h \&\& k<T_k,\\ MR, & 其他\end{cases}$$

其中 $stdev(clusters)$为进程簇体积的均方差，k 为进程簇数量，T_l，T_h和 T_k分别为均方差的低、高阈值和进程簇数量阈值. 规则表示当进程簇体积足够均衡时采用朴素映射策略，当不均衡程度达到一定阈值且进程簇数量较少时采用 First-Fit 策略，当均衡程度介于两者之间时采用 Most Reservation 策略.

4.4.3　基于 APE 算法的进程簇映射优化

在实现了 4.4.2 小节进程簇自适应映射后，我们可以设计相应的适用于 AQAP 模型的启发式算法实现对已有算法的复用，进一步优化映射结果.

文献[6]提出的 Pair-Exchange 改进算法是一种耗时可控的启发式算法，简称 PE 算法. 其算法描述如下：在每一轮迭代中，随机或按序选择进程对(i,j)，试探性交换两个进程所在的物理位置得到新映射 π'，如果 π'使得式(1)的总开销减小则执行实际交换，否则维持原映射 π.

利用 AQAP 为指导，我们基于 PE 算法设计聚合 PE 算法(Aggregated PE，APE)，如算法 4 所示. 算法首先将体积较小进程簇标记为噪音，然后对大小相同的噪音使用 PE 算法.

算法 4.　APE 算法.

输入：迭代次数 l，通信模式矩阵 $\boldsymbol{A}$，拓扑距离矩阵 $\boldsymbol{B}$，初始映射 π，k 个进程簇 $cluster(i)$，$i=0,1,\cdots,k-1$

输出：优化映射 π'

1. 在 $cluster$ 中分离噪音 $noise$
   FOR $i=0$ TO $k-1$ DO
     IF $|cluster(i)|<NOISE\_THRESHOLD$ THEN
       $cluster(i)\in noise$
     END IF
   END FOR
2. 对噪音进程簇使用 PE 算法
   FOR $i=0$ TO $l-1$ DO
     choose $cluster(i),cluster(j)\in noise$
     IF $|cluster(i)|==|cluster(j)|$ THEN
       CALL PE(π, $cluster(i)$, $cluster(j)$)
     END IF
   END FOR

图 5 展示了 APE 算法的一个示例，图中 c 代表大块进程簇(chunk)，n 代表噪音(noise). 图 5(1)表示进程簇初始映射结果，双向箭头所连接的噪音是可尝试交换的. 图 5(2)给出了一次噪音交换后的结果. 只对噪音使用 PE 算法是因为交换噪音即可获得与交换非噪音相同的组合结果(图 5(3)所示)，而噪音交换开销较小.

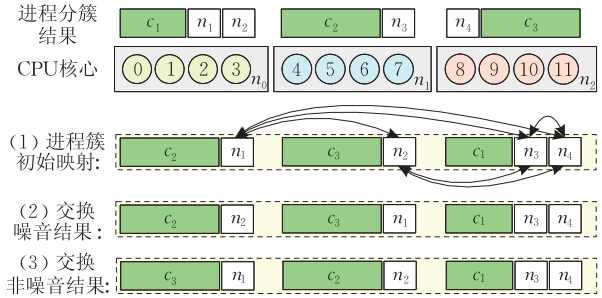


图 5　交换噪音与非噪音结果

4.5 层次化指定映射

前述单层映射可以将进程直接映射到各个节点上,而在拥有多层次拓扑的大规模计算环境中(如胖树结构超级计算机),出于性能/开销的考虑将进程映射到大的聚合节点上往往是更实用的方法.为满足上述需求,我们设计层次化指定映射,可将进程指定映射到某一层交换机上.

典型的应用情景如图 6 所示,图中系统是常见的层次化拓扑结构集群,其聚合节点内部的拓扑距离差别细微,因此只需要将进程簇映射到聚合节点上即可获得近似最优映射.具体设计如下,层次化指定映射可以分解为 h 次单层映射,h 为聚合节点到根交换机的层次.以图 6 中系统为例,h 为 2,在第 1 次映射迭代中,将 Level2 交换机视为虚拟节点进行单层进程簇映射,然后按同样的方式进行第 2 次映射迭代.在第 2 次迭代中,Level2 交换机中的进程簇是待映射进程簇,将 Level1 交换机视为虚拟节点进行单层映射,依此类推,直至进程映射到指定的聚合节点上.

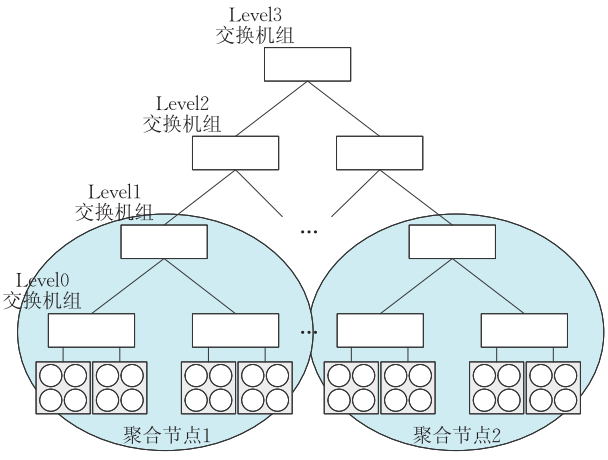


图 6 典型的具有层次化拓扑的集群系统

在层次化指定映射中,将开销较大的单层映射分解为若干次低开销的粗粒度单层映射,在如图 6 系统的大规模计算环境下具有性能/开销优势.

5 实验结果

5.1 实验环境及方法

5.1.1 实验环境

实验集群包含 12 个计算节点,节点间使用 InfiniBand 网络互连.每个节点拥有 62 GB 内存,2 个 8 核心 Intel® Xeon® E5-2670 CPU,主频为 2.60 GHz.每个 CPU 拥有 20 MB L3 cache.操作系统为 Red Hat Enterprise Linux Server 6.3,MPI 版本 MVAPICH2 v1.9a(icc 13.0.1).在实验中未启用处理器的超线程特性.

5.1.2 测试程序

本文使用 NAS 基准测试程序集以及两个实际应用程序进行实验测试.

(1) NPB[26].NAS 并行基准测试程序中的 4 道检测通信性能的典型程序:

① CG.共轭梯度方程测试程序,使用共轭梯度法计算稀疏对称有限矩阵的最小特征值.

② LU.上下对角线测试程序,采用对称的超松弛法求解块稀疏方程组.通信多使用"方块化"数据.

③ SP.标量五角测试程序,倾向于检测计算和通信之间的平衡,与上述程序不同,SP 要求进程数量为平方数.

④ BT.块状三角测试程序,同样要求进程数量为平方数,通信强度较 SP 低.

(2) 3dwing.全称 3D-acoustic wave-modeling,用来模拟三维中声波传播规律.三维声波模型仿真广泛运用于航空发动机降噪、乐器制造等领域.

(3) openform.计算流体力学(CFD)程序,通过计算模拟分析流体流动性质.

5.1.3 实验方法

(1) 实验对比设置

实验中我们比较了 4 类进程映射策略:

① 两种 MPI 默认进程映射方法.block 及 round-robin,以前者结果作为基线.

② Pair-Exchange 映射.根据程序规模,设置迭代次数为 5×10^5,记为 pe-500 k.

③ 图划分映射.由现有的图划分算法生成,我们选用最流行的图划分软件 METIS[15].

④ 聚类映射.本文提出的优化方法,记为 clustering mapping.根据测试程序特点本文设置分簇数 k 为节点数的两倍(经验值),k 的最优取值与通信模式及物理拓扑相关,其量化关系及 k 的自调谐取值是未来研究的方向.

(2) 重新映射绑定方法

根据运行环境的不同,实验中使用两种进程映射绑定方法:① 使用 MPICH2 的进程管理器 hydra 提供的绑定机制;② 使用资源管理程序 slurm① 提供的进程-节点绑定机制.

① Simple Linux Utility for Resource Management. https://computing.llnl.gov/linux/slurm/

5.2　实验结果及分析

5.2.1　进程映射优化效果对比

我们使用 5.1.3 节设置的对比方法分别对 NPB 基准测试程序和 3dwing 及 openform 两道实际应用进行了测试.

(1) NPB 程序测试结果如图 7 所示.

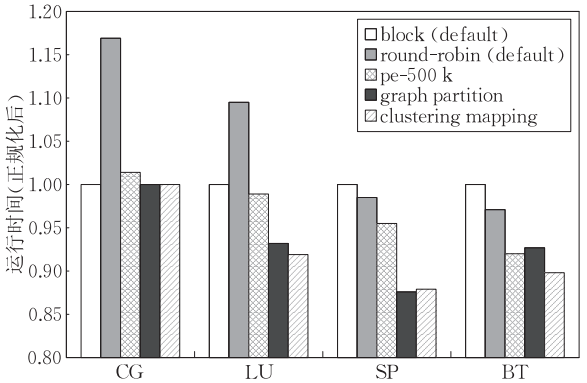


图 7　不同进程映射方法 NPB 测试结果对比(程序规模：cg.A.128，lu.A.128，sp.A.144，bt.A.144)

图 7 结果表明，对于选取的除 CG 之外的 NPB 程序，相对于 MPI 默认的进程映射方法，聚类映射方法可以获得 8.1%～12.1%的性能提升，并且在整体上优于 PE 和图划分映射. 这是由于上述 PE 和图划分算法均以某种初始映射(或初始划分)为基础寻找最优解，其结果受初始解的影响，易陷入较差的局部最优解[15]. 对于 CG 程序，我们在实验中发现使用聚类映射和图划分映射的结果与 block 方式相同，因此结果表现为无加速. 我们通过分析通信模式发现，原因为 CG 程序通信模式具有明显的簇状结构(图 8)，因此默认 block 映射已是最优解.

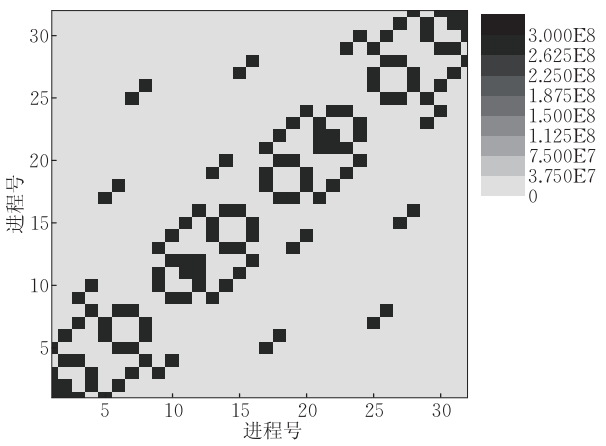


图 8　CG 程序通信模式矩阵图(cg.C.32)

上述分析也从侧面说明聚类映射方法在发现进程簇方面的有效性.

(2) 图 9 所示是实际应用的测试结果. 结果表明在更大规模的应用中使用聚类映射算法也取得了 4.3%～12.3%的性能提升. 相比于图划分算法也有最高达 3.7%的性能提升. 再次验证了聚类映射对较差局部最优解的规避能力.

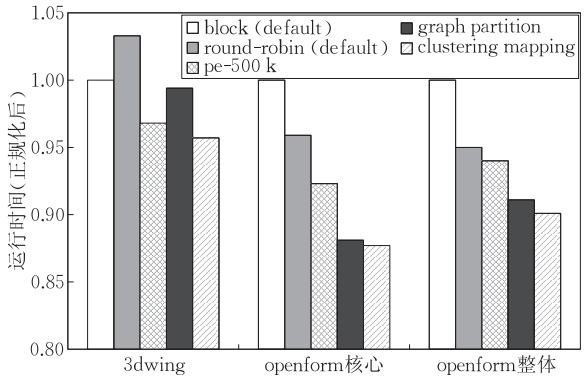


图 9　不同进程映射方法实际应用测试结果对比(程序规模：3dwing：144 进程(12×12)，openform：192 进程(16×12))

(3) 使用 SP 基准测试程序(CLASS＝A)对聚类映射方法效果的扩展性进行测试，结果如图 10 所示. 图中数据表明，随着进程数量的增多，使用聚类映射方法的 SP 程序获得的性能提升呈上升趋势. 这是因为在进程规模较小时，核心间的远距离通信和端口的拥塞较少. 因此进程拓扑映射对小规模程序的性能影响甚微，并会随着进程规模的增大影响越来越明显.

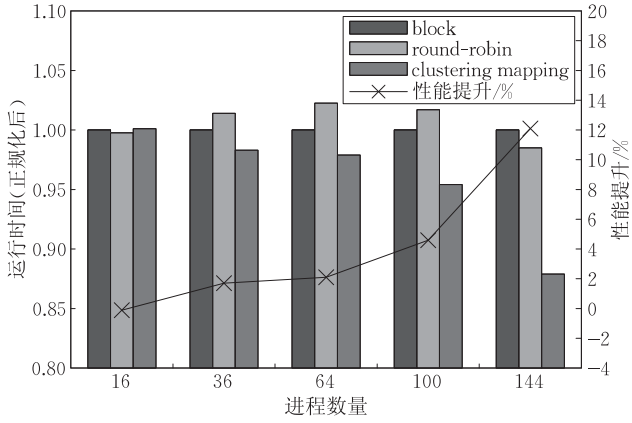


图 10　聚类映射方法效果扩展性测试(SP)

5.2.2　进程映射优化开销

本小节测试我们提出的聚类进程映射方法的开销. 使用 SP、3DWING 和 OPENFORM 程序的通信模式作为输入，测试环境为配备 Intel® Core™ i3 处理器的 Desktop PC，软件环境为 Linux 2.6 和 GCC 4.4.7.

测试结果如表 1 所示.

表 1 基于聚类分析的进程映射算法执行时间

测试程序	规模	映射执行时间/s
SP	144	0.224
3DWING	144	0.227
OPENFORM	192	0.241

从表 1 中可以看出,对于小于 192 进程的程序,使用聚类进程映射的开销几乎是可以忽略的.由 $O(nk^2)$ 的复杂度可粗略估算对 2000 进程的程序进行聚类进程映射的开销约 250 s.达到了可处理大规模数据的预期目标.

6 结论及未来工作

在本文中,我们提出了一种聚合的进程拓扑映射模型 AQAP,并基于 AQAP 模型提出了一种新颖的基于聚类分析的进程映射优化方法.与已有研究不同,该优化方法使用进程簇作为映射单位搜索 AQAP 问题的近似最优解,其映射效果十分依赖于进程聚类结果的准确性,为此我们提出了一种基于谱聚类分析的进程分簇算法,即 CP-NSC 算法.可以将通信模式聚类问题转化为低维聚类问题,在保证效率的同时避免了陷入较差的局部最优解.然后,提出了一种用于进程簇映射的自适应聚合进程映射策略,根据进程聚类结果的均衡性自适应地选择最优映射算法.并提出了一种基于 Pair-Exchange 的聚合 PE 算法,通过复用启发式算法进一步优化映射结果.此外,为了满足层次化大规模系统上指定聚合节点上的进程映射,我们提出了层次化指定映射方法,通过分解为多次单层映射实现进程-聚合节点映射.

我们使用多道基准测试程序和实际应用进行了广泛的实验.实验结果表明本文提出的进程优化方法可以使程序获得明显的性能提升,优于现有的 Pair-Exchange 方法和图划分方法.并且进程映射的开销微小,具备处理大规模数据的能力.

由于聚类分析的性能对于本文方法的有效性至关重要,因此未来工作主要从以下 3 个方面进行:(1)量化研究谱聚类分析参数(如分簇数量 k)对聚类结果和进程映射性能的影响,提高进程聚类分析的准确性;(2)研究进程簇自调谐聚类算法对本文方法的影响,即不再规定分簇数量 k 而由聚类算法主动寻找最优分簇数量;(3)进一步研究具有复杂网络拓扑结构的超级计算机系统上的聚类进程映射优化问题.

参 考 文 献

[1] Bhatele A. Automating Topology Aware Mapping for Supercomputers[Ph. D. dissertation]. Department of Computer Science, University of Illinois, Illinois, USA, 2010

[2] Hoefler T, Snir M. Generic topology mapping strategies for large-scale parallel architectures//Proceedings of the 25th International Conference on Supercomputing. Tucson, USA, 2011: 75-84

[3] Bhatele A, Kalé L V. An evaluative study on the effect of contention on message latencies in large supercomputers//Proceedings of the 23th IEEE International Parallel & Distributed Processing Symposium (IPDPS). Rome, Italy, 2009: 1-8

[4] Sahni S, Gonzalez T. P-complete approximation problems. Journal of the Association of Computing Machinery, 1976, 23(3): 555-565

[5] Ercal F, Ramanujam J, Sadayappan P. Task allocation onto a hypercube by recursive mincut bipartitioning. Journal of Parallel and Distributed Computing, 1990, 10(1): 35-44

[6] Brandfass B, Alrutz T, Gerhold T. Rank reordering for MPI communication optimization. Computers & Fluids, 2013, 80 (Complete): 372-380

[7] Mercier G, Clet-Ortega J. Towards an efficient process placement policy for MPI applications in multicore environments//Proceedings of the 16th European PVM/MPI Users' Group Meeting. Espoo, Finland, 2009: 104-115

[8] Chen H, Chen W, Huang J, et al. MPIPP: An automatic profile-guided parallel process placement toolset for SMP clusters and multiclusters//Proceedings of the 20th Annual International Conference on Supercomputing. Queensland, Australia, 2006: 353-360

[9] Jeannot E, Mercier G. Near-optimal placement of MPI processes on hierarchical NUMA architectures//Proceedings of the 16th International Euro-Par Conference on Parallel Processing. Ischia, Italy, 2010: 199-210

[10] Subramoni H, Potluri S, Kandalla K, et al. Design of a scalable InfiniBand topology service to enable network-topology-aware placement of processes//Proceedings of the International Conference for High Performance Computing. Salt Lake City, USA, 2012: 1-12

[11] Zhang J, Zhai J, Chen W, et al. Process mapping for MPI collective communications//Proceedings of the 15th International Euro-Par Conference on Parallel Processing. Delft, Netherlands, 2009: 81-92

[12] von Luxburg U. A tutorial on spectral clustering. Statistics and Computing, 2007, 17(4): 395-416

[13] Träff J L. SMP-aware message passing programming//Proceedings of the 17th IEEE Parallel & Distributed Processing Symposium (IPDPS). Washington, USA, 2003: 56-65

[14] Rashti M J, Green J, Balaji P, et al. Multi-core and network aware MPI topology functions//Proceedings of the 18th EuroMPI Conference. Santorini, Greece, 2011: 50-60

[15] LaSalle D, Karypis G. Multi-threaded graph partitioning//Proceedings of the 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS). Boston, USA, 2013: 225-236

[16] Pellegrini F. Scotch and libScotch 6.0 User's Guide. Bacchus team, INRIA Bordeaux Sud-Ouest Technical Report: CNRS 5800, 2012

[17] Walshaw C, Cross M. JOSTLE: Parallel multilevel graph-partitioning software—An overview. Mesh Partitioning Techniques and Domain Decomposition Techniques, United Kingdom: Civil-Comp Ltd., 2007

[18] Jeannot E, Mercier G, Tessier F. Process placement in multicore clusters: Algorithmic issues and practical techniques. Parallel and Distributed Systems, 2014, 25(4): 993-1002

[19] von Alfthan S, Honkonen I, Palmroth M. Topology aware process mapping//Proceedings of the 11th International Conference on Applied Parallel and Scientific Computing. Helsinki, Finland, 2012: 297-308

[20] Leung V J, Bunde D P, Ebbers J, et al. Task mapping stencil computations for non-contiguous allocations//Proceedings of the 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Orlando, USA, 2014: 377-378

[21] Nie Peng-Cheng, Duan Zhen-Hua, et al. Adaptive scheduling on performance asymmetric multicore processors. Chinese Journal of Computers, 2013, 36(4): 773-781(in Chinese)
(聂鹏程，段振华等. 性能非对称多核处理器上的自适应调度. 计算机学报，2013, 36(4): 773-781)

[22] Martin R P, Vahdat A M, Culler D E, et al. Effects of communication latency, overhead, and bandwidth in a cluster architecture//Proceedings of the 24th Annual International Symposium on Computer Architecture. Denver, USA, 1997: 85-97

[23] Ng A Y, Jordan M I, Weiss Y. On spectral clustering: Analysis and an algorithm//Proceedings of the 14th Neural Information Processing Systems (NIPS). Vancouver, Canada, 2001: 849-856

[24] Chen W Y, Song Y, Bai H, et al. Parallel spectral clustering in distributed systems. Pattern Analysis and Machine Intelligence, 2011, 33(3): 568-586

[25] Liu R, Zhang H. Segmentation of 3D meshes through spectral clustering//Proceedings of the 12th Pacific Graphics. Seoul, Korea, 2004: 298-305

[26] Bailey D H, Barszcz E, Barton J T, et al. The NAS parallel benchmarks 2.0. NASA Ames Research Center, Technical Report: NAS-95-020, 1995

WANG Tao, born in 1989, M. S. candidate. His research interests include parallel compiling and high performance computing.

QING Peng, born in 1979, M. S. His research interest is run-time system.

WEI Di, born in 1984, M. S. His research interest is run-time system.

QI Feng-Bin, born in 1966, researcher, Ph. D. supervisor. His research interests include compilation technology and high performance computing.

Background

The increase both in network diameter and on-chip parallelism degree has made the communication wall problem increasingly severe. Optimizing for process-to-core mapping helps to improving applications communication efficiency. Unfortunately it is a NP-hard problem to find the optimal mapping. The traditional process mapping optimization model called Quadratic Assignment Problem (QAP) has over fine grain, resulting in inefficiency and tendency to split the process clusters within which the communication is dense. Many heuristics had been proposed to settle the mentioned QAP, including Pair-Exchange-based and graph-partition-based methods. However, most heuristics guided by the QAP model which are consequently inefficient tend to split the process clusters and drop into poor locally optimal solutions.

Our work in this paper focuses on solving the problems above. For the traditional mapping model's flaw, we propose a coarse-grain mapping model called Aggregated Quadratic Assignment Problem (AQAP) model. Guided by AQAP model, we propose a novel process mapping optimization method based on clustering analysis. In this optimization method, we propose a series of algorithms and strategies which solve the clustering-based process mapping problem well. Those algorithms and strategies include Communication Pattern Normalized Spectral Clustering (CP-NSC) algorithm, self-adaption aggregated process mapping strategy, Aggregated Pair-Exchange (APE) algorithm and hierarchical specified mapping strategy.

To the best of our knowledge, this is the first instance where the spectral clustering algorithm has been applied to solve the process-to-core mapping problem. We evaluated the performance of our method with the NPB benchmarks and two practical applications. Experimental results show that the optimized process placement generated by our method can achieve significant performance improvement. Moreover, our method has very low overhead as well as good scalability.

This work is supported by the National High Technology Research and Development Program (863 Program) of China under Grant No. 2012AA010903.