

**学士学位论文开题报告**

**论文题目**：基于拓扑映射的并行计算通信性能优化

**专 业**：计算机科学与技术

**本 科 生**：汪圣灵

**学 号**：15061014

**指导教师**：肖利民

**北京航空航天大学计算机学院**

2018年10月16日

目 录

[1题目背景与意义 4](#_Toc527679030)

[1.1课题来源 4](#_Toc527679031)

[1.2选题的背景与意义 4](#_Toc527679032)

[2研究现状 5](#_Toc527679033)

[2.1集成的任务布局算法现状 5](#_Toc527679034)

[2.1.1 MPI Process Placement toolset（MPIPP）[8] 5](#_Toc527679035)

[2.1.2 METIS[12] 6](#_Toc527679036)

[2.1.3 Chaco[9] 6](#_Toc527679037)

[2.2并行计算通信优化算法研究现状 6](#_Toc527679038)

[2.2.1 TreeMatch[10] 6](#_Toc527679039)

[2.2.2 SCOTCH[11] 6](#_Toc527679040)

[2.2.3 Topo LB[13] 6](#_Toc527679041)

[2.2.4 Fast Approximate Quadratic（FAQ）[14] 6](#_Toc527679042)

[2.3尚且存在的问题： 7](#_Toc527679043)

[3研究内容与解决方案 7](#_Toc527679044)

[3.1研究目标 7](#_Toc527679045)

[3.1.1调查研究现有高效的拓扑映射方法 7](#_Toc527679046)

[3.1.2实现几种拓扑映射算法 7](#_Toc527679047)

[3.1.3设计并实现原型系统 8](#_Toc527679048)

[3.2研究内容 8](#_Toc527679049)

[3.2.1调查研究现有的高效的拓扑映射方法 8](#_Toc527679050)

[3.2.2基于拓扑映射的并行计算性能优化算法实现 8](#_Toc527679051)

[3.2.3设计基于并行计算性能优化算法的网络拓扑图生成系统 8](#_Toc527679052)

[3.3解决方案 8](#_Toc527679053)

[3.3.1. 调查研究现有的高效的拓扑映射方法 9](#_Toc527679054)

[3.3.2. 基于拓扑映射的并行计算性能优化算法实现与验证 9](#_Toc527679055)

[3.3.3. 设计基于并行计算性能优化算法的网络拓扑图生成系统 9](#_Toc527679056)

[4关键技术及难点 10](#_Toc527679057)

[4.1关键技术 10](#_Toc527679058)

[4.2技术难点 10](#_Toc527679059)

[5进度安排 10](#_Toc527679060)

[6参考文献 10](#_Toc527679061)

# 1题目背景与意义

随着科技的不断发展，高性能计算集群（High Performance Computing Cluster）得到了广泛的应用。无论是工业领域还是科研领域，如CAE（汽车、航空航天、船舶）和EDA（芯片设计单位），以及计算物理/化学/材料/气象等环境下，用计算方法模拟和分析现实应用场景等技术都取得了巨大的进展，而这对计算能力产生了巨大的需求，传统的单个主机由于其处理器计算能力以及磁盘存储能力的局限性，已经不能够适应如此大的任务量和数据量，而高性能计算系统则拥有多处理器共同完成一个任务、多用户、多作业等特点，是一种支持计算与IO密集型应用的多服务器计算机集群系统。纵观CPU体系架构的发展趋势，单个CPU的性能由于功耗和散热问题已经达到了一个瓶颈，于是，对高性能的需求促进了高性能计算的发展，从“天河二号”到“神威·太湖之光”，从云计算到GPGPU，都在使用并行计算处理巨大规模的计算问题。因此，对并行计算的性能提升成为了当前研究的主要课题。

而针对不同的运算环境，如单台服务器，并行计算通常使用OpenMP[1]或者Pthread[2]技术来形成多线程程序以充分利用多核多线程的并行处理能力，OpenMP技术根植于编译器，通过将原来串行的程序通过加入适当的编译器指令，编程并行执行从而提高程序的运行速度。而Pthread则是一套基于POSIX对线程的标准而设计的关于线程的API，提供多线程开发的库，实现多线程编程来提高程序运行速度。但另一方面，当计算的规模超过了单个节点的处理能力时，则需要多台机器同时进行计算，使用的技术则为MPI技术，即通信接口技术（Message-Passing Interface）[3]。此时所牵扯到的问题便是多台电脑并行计算时的数据通信问题，在理想状态下，并行计算机的性能应该是所有节点计算机的性能之和，但是，这种理想状态是假设任意两台计算机之间的通信距离为0，即所有计算机在进行并行计算时通信开销都理想化为无开销。其中，影响应用程序计算性能发挥的主要瓶颈之一是消息传递通信[4]。但是，并行计算机是由一组处理单元组成的，这组处理单元通过相互之间的通信与协作，以更快的速度共同完成一项大规模的计算任务。因此，并行计算机的两个最主要的组成部分是计算节点和节点间的通信与协作机制[5]。因此，要提高并行计算能力，减少通信开销起着极其重要的作用[6][7]。

而另一方面，随着网络技术的发展，分布式计算为高性能计算开辟了新的领域，人民可以使用并行计算机以便在更长和更精确的时间尺度上解决更大的问题。然而，随着应用程序并发性和输入数据大小的预期增加，未来几年要解决的最重要挑战之一是局部性，即如何改进应用程序内的数据访问和传输[8]，一个分布式系统由多台同构或异构的计算机、高速网络、接口、操作系统、通信协议和程序设计环境组成，而分布式计算则需要将一个计算问题划分成多个任务，即任务布局，并分配给各个计算节点进行并行计算。为了提高并行计算的速率，则必须将一个计算问题尽可能的划分为均匀分布在各台计算机上的单个任务进行处理，充分利用分布式系统中所有的计算资源。但如此也增加了各个任务之间的通信量，因此，如何平衡多任务平均划分和通信量两个互相矛盾的问题以达到最佳性能，便是任务到处理机的映射问题[8]。目前高性能计算机中普遍使用的任务布局算法包括EFG法以及其拓扑优化、MPIPP、METIS等，这些算法各自都在某一个方面解决了部分问题，但却存在着集成性高、可移植性弱、源码不开源等问题，因此，本毕设的最终研究目的便是通过设计一个实现算法的系统来解决用户友好性问题。

# 2研究现状

## 2.1基于拓扑映射的任务布局算法研究现状

通信延迟是影响并行应用程序性能的一大重要因素，而基于拓扑映射的任务布局算法旨在将并行程序的子任务以最小化网络负载的方式分配给处理器，从而减少通信所造成的时间开销。到目前为止，学术界已经提出了很多处理此问题的布局方案和映射算法。

### 2.1.1 TreeMatch[11]

当一个高性能计算机的体系结构中的节点具有多核或者多核处理器时，开发对应的并行计算程序就需要考虑到底层的硬件特性，如存储器的层次机构。而TreeMatch便是通过将通信的模式和底层的硬件架构相匹配来优化应用程序进程之间的数据传输，从而降低通信开销的任务布局算法。

### 2.1.2 MPI Process Placement toolset（MPIPP）[9]

在SMP集群和多集群体系结构下，将并行过程映射到处理器或核心上的任务布局过程要么需要密集的用户干预，要么无法较好的处理多集群的情况。针对此类问题，MPIPP提供一种配置文件引导方法，可以自动查找优化的映射，从而最大限度的降低消息传递的通信开销，对于多集群的现有图分区算法效果显著。MPIPP主要包含获取MPI应用程序通信配置文件的组件，以及获取目标集群网络拓扑的组件和映射优化算法。

### 2.1.3 Topo LB[14]

在实际的拓扑感知映射过程中，应用程序集的多样化、通信模式的可选择性、算法的领域特定性等问题导致拓扑映射算法并没有得到广泛的应用。Topo LB则是一个完整的自动映射系统，能够提取程序和硬件的关键模式提供给算法，适用于各种通信模式和机器拓扑结构。

### 2.1.4 Fast Approximate Quadratic（FAQ）[15]

任务布局中拓扑图的匹配问题是二次分配问题的一个个例，而在此类图匹配问题（GMP）中需要解决的是准确性和效率的平衡问题，较慢的算法可以实现更好的性能时间。而快速近似二次分配算法（FAQ）实现了准确性和效率之间的最佳可行权衡。

## 2.2基于拓扑映射的任务布局算法工具

### 2.2.1 METIS[13]

图分区在任务布局优化过程中有着广泛的应用，是指将图的定点划分为大致相等的部分，使得连接不同部分的顶点的边的数量最小。METIS是图形划分程序，能够完成图分区算法功能。且具有速度快、划分质量高等优点。

### 2.2.2 Chaco[10]

在图分区的应用上，Chaco是Python上的交互式图表2D绘图库，其中集成了图分区算法，允许用户递归应用几种方法来在加权图中查找小边缘分隔符，包括inertial、spectral、Kernighan–Lin、multilevel methods以及其他更简单的划分策略。

### 2.2.3 SCOTCH[12]

将并行程序的子进程分配给高性能计算机以最小化整体运行时间的优化问题称为映射，而当此类优化在程序执行之前就已经完成并且在程序运行过程中不再修改，则为静态映射。SCOTCH便是一种基于源过程图和目标体系结构图的递归二分法静态映射软件包，该软件包主要实现了将任何加权源图映射到任何加权目标图形上的功能，以及图形双分区启发式算法等。

## 2.3尚且存在的问题：

由上面的算法可以看出，部分算法仅仅只是被提出，但却没有落实被实现成可用的工具，实际的开发人员或者试验人员如物理学家、生物学家，在使用过程中，无法将这些算法具体实现出来，也就无法在实验中使用和验证。还有一部分算法虽然实现出来了，但是却没有形成一个系统，可拓展性和可移植性缺失，在实际运用过程当中，使用人员需要花费大量的精力来移植算法或者重写算法，仍然无法满足实际需求。另一部分算法则是以程序的形式呈现，没有开放算法源代码，仅局限于部分特定领域使用，在其他领域或者环境下则无法使用其程序来进行实验操作。

另一方面，作为一项实际应用于并行计算通信性能优化的任务布局技术，需要能够适用于大部分的程序集和高性能计算机集群，且能适用于各种通信模式，还要有自动选择特定算法或者最优算法的功能。前篇提到的各类算法和系统，都难以从全局上满足这些功能。

# 3研究内容与解决方案

## 3.1研究目标

### 3.1.1调查研究现有高效的拓扑映射方法

本项目将通过调查和研究相关文献资料和算法程序，进行理论分析和验证分析，理解算法的功能原理，并总结出各个算法的特性，包括算法对应适用的程序集、硬件集、通信模式等关键特性。最后通过理论验证算法的优化效率，通过对比找出现有的高效率任务布局策略或者算法。

### 3.1.2实现几种拓扑映射算法

本项目将针对调查研究的结果，对现有的高效率的拓扑映射算法进行编程设计，设计出算法的流程图、数据结构以及外部运行环境，实现出算法的功能特性，并通过编程方式实现算法程序，并设计出相应的测试程序和测试数据进行有效性验证。

### 3.1.3设计并实现原型系统

本毕设针对以上已实现的拓扑映射算法程序，设计系统的流程图、输入输出的数据结构以及系统运行环境，其中流程图包括算法选择的自动化流程以及算法流程。并最终通过编程方式实现一个系统，能够通过算法，以并行程序集、通信模式、计算节点拓扑图为计算数据，生成一个并行程序集到计算节点图的最优任务布局映射方案。

## 3.2研究内容

### 3.2.1调查研究现有的高效的拓扑映射方法

并行计算的基本原则包含：与体系结构相结合，具有可拓展性，粗粒度，减少通信，优化性能[5]。而减少通信的一个方面在于减少计算机之间的通信距离，本毕设在并行计算性能优化方面的主要研究内容是通过研究和调查找到一些已经验证过高效性和可用性的算法，并分析和总结出算法的特性，包括适用的程序集、硬件集、通信模式等关键特性。

### 3.2.2基于拓扑映射的并行计算性能优化算法实现

本毕设在此阶段的项目内容主要是研究如何通过编程方式，将调查研究后的具有高效性的拓扑映射算法进行设计和实现。并进行有效性验证。其中对算法进行设计时，需要设计算法的流程图、数据结构以及运行环境，在实现过程中，需要搭建相应的编程环境和测试环境，在有效性验证过程中，需要设计测试数据和测试程序，并搭建相应的测试环境。

### 3.2.3设计基于拓扑映射算法的并行计算通信性能优化系统

本毕设在此阶段的主要研究内容在于设计一种能够通过算法，以并行程序集、通信模式、计算节点拓扑图为计算数据，生成一个并行程序集到计算节点图的最优任务布局映射方案的系统。系统需要至少实现一种算法，并能够自动根据配置信息调整布局方案。

## 3.3解决方案

本项目研究方案包括以下三点：

### 3.3.1. 调查研究现有的高效的拓扑映射方法

阅读相关领域论文，包括高性能计算、并行计算、拓扑映射、通信性能等，并进行理论分析和验证分析，对比理论结果和验证结果，得出现有的具有高效性的拓扑映射算法。

### 3.3.2. 基于拓扑映射的并行计算性能优化算法实现与验证

算法实现：

本项目针对调查研究的内容和不同的应用场景选择不同的算法，根据论文描述实现算法程序，并以文件的形式产生输入规则和输出规则，对特定的算法，输入指定的计算节点数据文件，产生网络拓扑结构图。

算法验证：

(1). 有效性验证：

按照生成的拓扑结构图布置超级计算机的计算节点，并运行并行计算程序，检测程序的运行有效性。

(2).对比试验：

分别按照本项目算法和实验室已有算法生成的拓扑结构图布置超级计算机的计算节点，并多次运行同一个并行计算程序，运算时间取平均值，比较每个算法生成的拓扑图的运行时间。

### 3.3.3. 设计基于拓扑映射算法的并行计算通信性能优化系统

系统采用python来实现，根据选择的算法，将读取的数据文件分析并按照算法计算出最优的拓扑结构图，并输出为文件。



图 1 系统结构图

(1). 系统有效性检测

输入指定的数据文件，选择每种算法一次，检测程序是否正常运行。检测输出文件是否符合输出格式和是否正确

(2). 系统界面优化

将系统的交互式界面进行优化，体现较好的用户友好性

# 4关键技术及难点

## 4.1关键技术

### 4.1.1拓扑映射算法

并行机器上并行程序的有效执行要求将程序的通信过程分配给机器的处理器，以便最小化其整体运行时间。当假定进程在所有程序的持续时间内同时共存时，这种优化问题称为映射。它相当于平衡机器处理器之间进程的计算权重，同时通过在附近的处理器上保持密集的通信进程来减少并行性引起的通信开销。如果映射在执行程序之前计算并且从不在运行时修改，则称为静态。在一般情况下，静态映射是NP完全的[17]。

### 4.1.2Python程序设计与实现

Python是当前环境下主流的编程语言之一，具有高可读性、高一致性、高质量等特性，能够显著提升程序开发效率。Python开发的程序具有可移植性，Python还提供了大量的标准库，其中包含众多应用级编程库，大大降低了开发难度。在进行算法实现过程中，Python的标准库能够极大的减少系统开发时间，能够使研究的主要重心放在算法实现上而不是代码编写上。

## 4.2技术难点

拓扑映射算法有很多种，无论是传统的还是最新的，不同的算法模型适用于不同的场景，也会有不同的输入数据和参数选择，在进行理论分析和验证分析时，需要综合参考各项参数，而如何寻找出具有高效性的拓扑映射算法将会是一个难点。

拓扑映射算法部分仍然停留在理论层次，没有具体的代码实现，而本毕设需要通过分析和验证算法，并使用编程方法将算法设计和实现出来，将会使一个难点。

# 5进度安排

|  |  |
| --- | --- |
| 2018年10月20日—2018年10月31日 | 开题报告撰写 |
| 2018年11月01日—2018年11月30日 | 算法调查和研究 |
| 2018年12月01日—2018年12月31日 | 算法程序实现 |
| 2019年01月01日—2019年02月31日 | 算法系统实现 |
| 2019年03月01日—2019年04月01日 | 论文撰写 |

# 6参考文献

|  |  |
| --- | --- |
|  | Chandra R, Dagum L, Kohr D, et al. Parallel programming in OpenMP[M]. Morgan Kaufmann Publishers, 2001. |
|  | Buttlar D, Farrell J, Nichols B. PThreads Programming[J]. Oreilly Media, 1996. |
|  | Li Q, Huo Z, Sun N. Optimizing MPI Alltoall Communication of Large Messages in Multicore Clusters[C]// International Conference on Parallel and Distributed Computing, Applications and Technologies. IEEE, 2011:257-262. |
|  | 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. |
|  | 迟学斌, 赵毅. 高性能计算技术及其应用[J]. 中国科学院院刊, 2007, 22(4):306-313. |
|  | 刘青昆, 王佳, 韩颖,等. 一种并行计算通信优化策略[J]. 辽宁工程技术大学学报(自然科学版), 2011, 30(2):268-271. |
|  | Guo Q, Paker Y. Concurrent communication and granularity assessment for a transputer-based multiprocessor system[M]. CRL Publishing Ltd. 1990. |
|  | Jeannot E, Mercier G, Tessier F. Process Placement in Multicore Clusters:Algorithmic Issues and Practical Techniques[J]. IEEE Transactions on Parallel & Distributed Systems, 2014, 25(4):993-1002. |
|  | 胡金初. 并行计算中的任务分配算法[C]// 全国理论计算机科学学术年会. 2005. |
|  | Chen H, Chen W, Huang J, et al. MPIPP:an automatic profile-guided parallel process placement toolset for SMP clusters and multiclusters[C]// International Conference on Supercomputing. ACM, 2006:353-360. |
|  | B. Hendrickson and R. Leland, “The Chaco User’s Guide: Version 2.0,” Technical Report SAND94–2692, Sandia Nat’l Laboratory, 1994 |
|  | Jeannot E, Mercier G. Near-Optimal Placement of MPI Processes on Hierarchical NUMA Architectures[M]// Euro-Par 2010 - Parallel Processing. Springer Berlin Heidelberg, 2010:199-210. |
|  | Pellegrini F, Roman J. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs[J]. 1996, 1067(1067):493-498. |
|  | G. Karypis and V. Kumar. MiETiS - Unstructured Graph Partitioning and SparseMatrix Ordering System - Version 2.0. University of Minnesota, jun 1995. |
|  | Agarwal T, Sharma A, Laxmikant A, et al. Topology-aware task mapping for reducing communication contention on large parallel machines[C]// International Conference on Parallel and Distributed Processing. IEEE Computer Society, 2006:145-145. |
|  | Vogelstein J T, Conroy J M, Lyzinski V, et al. Fast approximate quadratic programming for graph matching.[J]. Plos One, 2015, 10(4):e0121002. |
|  | Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness[M]. W.H. Freeman and company, 1979. |