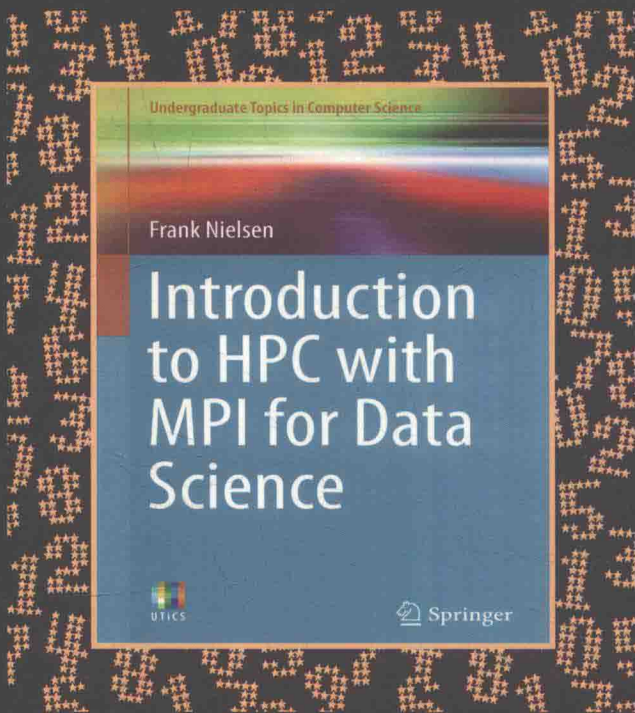


基于MPI的大数据 高性能计算导论

[法] 弗兰克·尼尔森 (Frank Nielsen) 著

张伟哲 等译



INTRODUCTION TO HPC WITH
MPI FOR DATA SCIENCE

数据科学与工程丛书

INTRODUCTION TO HPC WITH
MPI FOR DATA SCIENCE

基于MPI的大数据 高性能计算导论

[法] 弗兰克·尼尔森 (Frank Nielsen)

张伟哲 郝萌 鲁刚钊 王德胜 孙博文 译



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

基于 MPI 的大数据高性能计算导论 / (法) 弗兰克·尼尔森 (Frank Nielsen) 著; 张伟哲等译.
—北京: 机械工业出版社, 2018.6

(数据科学与工程丛书)

书名原文: Introduction to HPC with MPI for Data Science

ISBN 978-7-111-60214-9

I. 基… II. ①弗… ②张… III. 数据处理 IV. TP274

中国版本图书馆 CIP 数据核字 (2018) 第 133210 号

本书版权登记号: 图字 01-2017-4121

Translation from the English language edition:

Introduction to HPC with MPI for Data Science

by Frank Nielsen.

Copyright © Springer International Publishing Switzerland 2016.

This work is published by Springer Nature.

The registered company is Springer International Publishing AG.

All rights reserved.

本书中文简体字版由 Springer 授权机械工业出版社独家出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

本书使用 MPI 标准介绍了数据科学中的高性能计算, 帮助读者了解分布式存储模型中的并行编程的知识。全书分为两部分, 第一部分 (第 1~6 章) 基于消息传递接口介绍高性能计算, 内容包括: 阻塞与非阻塞的点对点通信、死锁、全局通信函数 (广播、散播等)、协同计算 (归约) 的基本概念; 互连网络的拓扑结构 (环、环面和超立方体) 以及相应的全局通信程序; 基于分布式内存的并行排序及其实现, 涵盖相关并行线性代数知识; MapReduce 模型。第二部分 (第 7~11 章) 介绍计算机集群中的高性能数据分析, 内容包括: 数据聚类技术 (平面划分聚类、层次聚类); 基于 k-NN 的有监督分类; 核心集以及相关降维技术; 图算法 (最稠密子图、图同构检测)。每章章末附有各种难度的练习和参考文献, 可供读者进行自测和深入学习。

本书适合作为“高性能计算”相关课程的本科生教材。

出版发行: 机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码: 100037)

责任编辑: 唐晓琳

责任校对: 殷虹

印刷: 北京瑞德印刷有限公司

版次: 2018 年 7 月第 1 版第 1 次印刷

开本: 185mm × 260mm 1/16

印张: 14.5

书号: ISBN 978-7-111-60214-9

定价: 59.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

译者序

数据科学(Data Science, DS)的目的是研究数据本身,研究数据的各种类型、状态、属性及变化形式和变化规律。随着当前数据量的指数级增长,数据科学的战略意义已经不在于掌握“大数据”(big data)信息本身,而在于对这些含有意义的数据进行快速、准确、高效地分析处理。

高性能计算(High Performance Computing, HPC)是辅助“大数据”快速分析与处理的利器之一。通过高性能网络互连的很多处理器或者某一集群中的大量计算机,对“大数据”进行并行聚类、分类以及降维等操作,从而能够更加快速地获取数据中蕴含的规律和有价值的信息。

本书的主要目的是将数据科学与高性能计算两大研究分支交叉融合起来,让高性能计算更好地为数据科学服务,让数据科学对高性能计算发展提出新的需求。因此,本书分为两个部分:第一部分,介绍了高性能计算的基本概念及基于标准消息传递接口(Message Passing Interface, MPI)的并行编程方法;第二部分,给出了高性能计算在数据科学中的应用,尤其重点阐述了如何利用MPI来并行化数据科学中的算法。

本书由哈尔滨工业大学计算机科学与技术学院张伟哲教授对全书统稿和校对,其中第1章、第4章、第5章、第11章和附录由哈尔滨工业大学鲁刚钊翻译,第2章由哈尔滨工业大学王德胜翻译,第3章由哈尔滨工业大学孙博文翻译,第6~10章由哈尔滨工业大学郝萌翻译。同时,我们的工作还得到哈尔滨工业大学梁智颖、任雪健的帮助,在此一并表示感谢。

还要特别感谢机械工业出版社的编辑唐晓琳、王春华在本书的编辑、出版方面所付出的辛勤劳动。

我们希望本书的出版对相关科技人员和读者有所帮助,同时也期望广大专家和读者提出宝贵意见。

前 言

欢迎来到高性能计算的世界！欢迎来到高性能数据科学的世界！

在本书中，我们将介绍面向数据科学(Data Science, DS)的高性能计算(High Performance Computing, HPC)。因此，本书主要分为两个部分：第一部分(前6章)涵盖HPC的基本原理；第二部分(后5章)介绍了数据科学的基本知识，并展示了如何编写面向基本串行算法的分布式程序，以应对大规模数据集。当前，许多大规模数据集都是公开的，这些数据集中蕴含了丰富的信息，但是这些信息需要通过精心设计才能被提取出来。

我们主要区分两种并行算法的设计方法：在单个共享内存多核机器上使用多线程并行化算法；在分布式内存集群系统上并行化算法。

一方面，当在共享内存架构(如智能手机、平板电脑，以及智能手表和其他物联网设备)上设计并行化算法时，所有的硬件计算单元(核)位于同一芯片上，我们可以使用多线程来轻松地对视频解码、渲染等任务进行并行化。这种并行是细粒度的(fine-grained)，但它受到芯片上物理核数的限制(2015年高端智能手机通常只有8个核)。另一方面，集群系统(即分布式内存架构)可以根据待处理的数据集规模来实时扩展资源。集群的构建具有很大的灵活性，例如可以选择异构的计算机节点，然后确定最适合这些节点的互连拓扑结构。这种并行是粗粒度的(coarse-grained)，因为在集群中发生节点间通信之前，每个节点可以独立地进行大量的本地计算。

本书侧重于在分布式内存系统上利用标准消息传递接口(Message Passing Interface, MPI)来设计并行算法。MPI是管理集群节点之间通信和全局协同计算的实用标准。目前存在多种MPI标准的供应商实现，它们可以与C、C++、Fortran、Python等多种编程语言绑定。我们选择面向对象的语言C++来实现数据科学中的算法，并使用和C语言绑定的OpenMPI应用程序编程接口(Application Programming Interface, API)来编写并行程序。

本书中两部分内容的简要介绍如下。

第一部分：基于消息传递接口的高性能计算

第 1 章首先简单介绍了 HPC 世界，然后讲解了 Amdahl 定律和 Gustafson 定律，这两个定律刻画了并程序的理论最优加速比和扩展加速比。

第 2 章讲解了 MPI 的主要概念和编程接口：阻塞/非阻塞通信的概念、死锁和多种全局通信函数(例如 broadcast、scatter、gather、all-to-all、reduce、parallel prefix 等)。

第 3 章着重介绍了互连网络拓扑的作用。我们首先区分物理拓扑和虚拟拓扑(或称为逻辑拓扑)，并在设计并行算法的时候考虑不同网络拓扑对性能的影响。特别讲解了环形(包括优化的流水线广播)和超立方体形网络拓扑上的通信过程，后者依赖于节点的特定编号，称为格雷码。

第 4 章讲解了基于分布式内存的主要的并行排序算法。首先对著名的快速排序算法(Quicksort)进行了简单的并行化，然后介绍实际中广泛使用的 Hyper-Quicksort 和 PSRS(Parallel Sorting by Regular Sampling)算法。

第 5 章研究了一些矩阵相乘和向量相乘的算法，并简要介绍了在环和环面(torus)的拓扑结构中计算矩阵乘积的各种技术。

第 6 章介绍了一个比较热门的并行编程范式，称为 MapReduce(通常与开源系统 Hadoop 一起使用)。MapReduce 可以通过两个主要的用户定义的函数(map 和 reduce)来构建程序，然后部署到大量的网络互连的计算机上来完成计算任务。然而，MapReduce 也是一个完整的框架，包括一个主从架构。该主从架构能够处理各种硬件故障，或者当一些机器执行得太慢时，将这些机器上的并行计算任务(作业)重新发送到其他的机器上执行。该章还讲解了如何利用专门的名为 MR-MPI 的软件库在 MPI(MPI 没有容错能力)中实现这些类型的 MapReduce 算法。

第二部分：面向数据科学的高性能计算

这部分简要介绍了数据科学，并进一步讲解了如何使用 MPI 并行化数据科学中的算法。

首先介绍了两个最基本的数据聚类技术，分别是平面划分聚类(第 7 章)和层次树聚类(第 8 章)。聚类是探索性数据科学中一个非常重要的概念，用于发现数据集中的分类、同质数据中的分组。

第 9 章介绍了基于 k 最近邻规则(k-nearest neighbor)的有监督分类，并和 k 均值(k-means)聚类算法进行关联。

第 10 章介绍了另一个计算科学中的新范式，允许人们在大型数据集(潜在的高维度)上解决优化问题。这种新范式就是寻找核心集(core-set)，这些核心集就是原数据集的子集，而且和原数据集相比具有良好的近似性。这种技术最近

变得非常流行，能够将大数据(big data)缩小到小数据(tiny data)! 由于数据通常具有高维度特征，所以还简要介绍了一种有效的线性降维技术，其中讲解了 Johnson-Lindenstrauss 定理，并给出一个简单的方法计算低失真嵌入，从而将数据从高维转化为低维，并确保在规定的近似因子内数据点之间的距离保持不变。有趣的是，嵌入的维度与原始外在维度无关，而是依赖于数据集大小的对数和近似因子。

第 11 章涵盖了一些图(graph)算法。图在社交网络分析和其他应用领域中是比较常见的。因此首先介绍一个顺序启发式方法和一个并行启发式方法来查找图的稠密子图，该子图近似于“最稠密”子图。然后介绍了在计算机集群上利用分支限界法来进行图同构检测。图同构检测是一个备受关注的问题，因为它的理论复杂度还没有得到解决(尽管对于图的某些特定子类存在一些多项式算法)。

每章最后会对该章的一些要点进行总结。请读者浏览这些总结，以便进行第一遍快速阅读。在一些章节结束时会给出 40 多道练习题，这些练习标有各种难度，并允许读者对练习所涵盖内容的理解程度进行自测。以星号开头的部分可以先跳过，稍后再进行阅读。

本书的主要目的是帮助读者设计并行算法，然后利用 C++ 和 C 语言绑定的 MPI 编写程序实现相应的并行算法。第二个目的是让读者对高性能计算和数据科学有更深刻的了解，并希望更好地促进两者之间的交叉。

本书是关于高性能计算和数据科学的入门教材，面向具有基本算法知识和编程能力的读者。因此，本书不包含(也没有提及)高性能计算和数据科学领域的高级概念。例如，任务调度问题和嵌套循环的自动并行化虽然在高性能计算中很重要，但是本书并没有涉及。类似地，本书也省略了数据科学领域中的回归技术和核心机器学习方法。

教辅资源

本书的额外资源(包括超过 35 个用 MPI/C++/R/Scilab/Gnuplot/Processing 编写的程序、幻灯片、相关链接和其他精彩内容)可以通过网址 <https://www.lix.polytechnique.fr/nielsen/HPC4DS/> 获取。

程序的源代码可以在上述网址以下列方式获取：

WWW source code: `example.cpp`

祝阅读愉快!

Frank Nielsen

2015 年 12 月

致 谢

非常感谢以下这些有才华的同事，他们给了我非常宝贵的反馈意见（姓名按随机顺序排列）并帮助我完善了本书：Claudia d' Ambrosio, Ulysse Beaugnon, Annaël Bonneton, Jean-Baptiste Bordes, Patrice Calégari, Henri Casanova, Antoine Delignat-Lavaud, Amélie Héliou, Alice Héliou, Léo Liberti, Frédéric Magoulès, Gautier Marti, Sameh Mohamed, François Morain, Richard Nock, Pierre-Louis Poirion, Stéphane Redon, Thomas Sibut-Pinote, Benjamin Smith, Antoine Soulé, Bogdan Tomchuk, Sonia Toubaline 和 Frédéric Vivien。除了以上这些同事，我还与其他很多同事进行了讨论，当你们读到这句话时，希望你们能够知道，从这些宝贵的交谈中，我受益匪浅。我还要感谢所有巴黎综合理工大学 INF442 课程的学生，感谢他们富有成效的意见和反馈，并且感谢巴黎综合理工大学计算机科学学院（DIX）的支持。

目 录

译者序
前言
致谢

第一部分 基于消息传递接口 的高性能计算

第 1 章 走进高性能计算	2
1.1 什么是高性能计算	2
1.2 为什么我们需要 HPC	3
1.3 大数据：四个特性（数据量、 多样性、生成速度、价值）	4
1.4 并行编程范式：MPI 和 MapReduce	4
1.5 粒度：细粒度并行与粗粒度 并行	5
1.6 超级计算架构：内存和 网络	5
1.7 加速比	8
1.7.1 扩展性和等效率 分析	9
1.7.2 Amdahl 定律：描述 数据规模固定时渐近 加速比的变化趋势	9
1.7.3 Gustafson 定律：可 扩展的加速比，随	

着资源的增加不断 扩大数据量	11
1.7.4 在串行计算机上模拟 并行机	12
1.7.5 大数据和并行输入/ 输出	13
1.8 关于分布式系统的八个常见 误区	13
1.9 注释和参考	15
1.10 总结	15
1.11 练习	16
参考文献	17

第 2 章 MPI 简介：消息传递 接口	18
2.1 基于 MPI 的并行程序 设计：基于消息通信	18
2.2 并行编程模型、线程和 进程	19
2.3 进程之间的全局通信	20
2.3.1 四个基本的 MPI 原语：广播、收集、 归约和全交换	20
2.3.2 阻塞与非阻塞和同步 与异步通信	22

2.3.3	阻塞通信产生的 死锁	24	2.9.1	MPI 中的矩阵 向量积	40
2.3.4	并发性：局部计算 可以与通信重叠 执行	27	2.9.2	MPI 归约操作示例： 计算数组的阶乘和 最小值	41
2.3.5	单向与双向通信	27	2.9.3	Monte-Carlo 随机积 分算法估算 π	42
2.3.6	MPI 中的全局计算： 归约和并行前缀 (扫描)	27	2.9.4	Monte-Carlo 随机 积分算法估算分子 体积	44
2.3.7	采用通信器定义 通信组	29	2.10	注释和参考	48
2.4	同步屏障：进程的交汇点	30	2.11	总结	49
2.4.1	MPI 中的一个同步 示例：测量运行 时间	30	2.12	练习	49
2.4.2	整体同步并行计算 模型	31		参考文献	50
2.5	开始使用 MPI：使用 OpenMPI	31	第 3 章	互联网络的拓扑结构	51
2.5.1	用 MPI C++ 编写 “Hello World” 程序	32	3.1	两个重要概念：静态与 动态网络，以及逻辑与 物理网络	51
2.5.2	用 C 绑定进行 MPI 编程	33	3.2	互联网络：图建模	51
2.5.3	通过 C++ Boost 使用 MPI	34	3.3	一些描述拓扑结构的 属性	52
2.6	通过 OpenMP 使用 MPI	34	3.3.1	度和直径	53
2.7	MPI 中的主要原语	36	3.3.2	连通性和对分	53
2.7.1	广播、散播、收集、 归约和全归约的 MPI 语法	36	3.3.3	一个好的网络拓扑 结构的标准	53
2.7.2	其余混杂的 MPI 原语	38	3.4	常见的拓扑结构：简单的 静态网络	54
2.8	环形拓扑上利用 MPI 进行的 通信	38	3.4.1	完全图：团	54
2.9	MPI 程序示例及其加速比 分析	39	3.4.2	星形图	55
			3.4.3	环和带弦环	55
			3.4.4	网（网格）与环面簇 (环面的集合)	55
			3.4.5	三维立方体与循环 连接立方体	56
			3.4.6	树与胖树	57

3.5	超立方体拓扑结构以及使用格雷码进行节点标识	57	4.7	基于网络的排序： ShearSort	89
3.5.1	超立方体的递归构造	57	4.8	使用比较网络排序： 奇偶排序	89
3.5.2	使用格雷码对超立方体节点编号	58	4.9	使用比较网络合并有序列表	92
3.5.3	使用 C++ 生成格雷码	59	4.10	双调归并排序	93
3.5.4	格雷码和二进制的相互转换	61	4.11	注释和参考	95
3.5.5	图的笛卡儿乘积	61	4.12	总结	95
3.6	一些拓扑结构上的通信算法	63	4.13	练习	95
3.6.1	有向环上的通信原语	64		参考文献	96
3.6.2	超立方体上的广播： 树状通信	68	第 5 章 并行线性代数		97
3.7	将（逻辑）拓扑结构嵌入到其他（物理）拓扑结构中	72	5.1	分布式线性代数	97
3.8	复杂规则拓扑结构	73	5.1.1	数据科学中的线性代数	97
3.9	芯片上的互连网络	74	5.1.2	经典线性代数	99
3.10	注释和参考	76	5.1.3	矩阵-向量乘法： $y = Ax$	101
3.11	总结	77	5.1.4	并行数据模式	101
	参考文献	77	5.2	有向环拓扑上的矩阵-向量乘积	102
第 4 章 并行排序		78	5.3	网格上的矩阵乘法：外积算法	108
4.1	串行排序快速回顾	78	5.4	二维环面拓扑上的矩阵乘积	108
4.1.1	主要的串行排序算法	78	5.4.1	Cannon 算法	110
4.1.2	排序的复杂性： 下界	80	5.4.2	Fox 算法：广播-相乘-循环移位矩阵乘积	111
4.2	通过合并列表实现并行排序	80	5.4.3	Snyder 算法：在对象线上进行本地乘积累加	115
4.3	利用秩实现并行排序	81	5.4.4	Cannon、Fox 和 Snyder 算法的比较	116
4.4	并行快速排序	82	5.5	注释和参考	116
4.5	超快速排序	86	5.6	总结	116
4.6	正则采样并行排序	87			

5.7 练习	117	7.2.2 k 均值优化问题的复 杂性和可计算性 ...	137
参考文献	117	7.3 Lloyd 批量 k 均值局部启 发式方法	138
第 6 章 MapReduce 范式	118	7.4 基于全局启发式的 k 均值 初始化方法	140
6.1 快速处理大数据的挑战	118	7.4.1 基于随机种子的初始 化方法	140
6.2 MapReduce 的基本原理 ...	119	7.4.2 全局 k 均值：最佳 贪心初始化	141
6.2.1 map 和 reduce 过程	119	7.4.3 k-means ++：一种 简单的概率保证的 初始化方法	141
6.2.2 历史视角：函数式 编程语言中的 map 和 reduce	120	7.5 k 均值向量量化中的应用	142
6.3 数据类型和 MapReduce 机制	121	7.5.1 向量量化	142
6.4 MapReduce 在 C++ 中的 完整示例	122	7.5.2 Lloyd 的局部最小值和 稳定 Voronoi 划分 ...	143
6.5 启动 MapReduce 作业和 MapReduce 架构概述	123	7.6 k 均值的物理解释：惯性 分解	143
6.6 基于 MR-MPI 库在 MPI 中使用 MapReduce	125	7.7 k 均值中 k 的选择：模型 选择	144
6.7 注释和参考	127	7.7.1 基于肘部法则的 模型选择	144
6.8 总结	127	7.7.2 模型选择：用 k 解释 方差减少	145
参考文献	128	7.8 集群上的并行 k 均值聚类 ...	145
第二部分 面向数据科学的 高性能计算		7.9 评估聚类划分	147
第 7 章 基于 k 均值的划分聚类 ...	130	7.9.1 兰德指数	148
7.1 探索性数据分析与聚类	130	7.9.2 归一化互信息	148
7.1.1 硬聚类：划分 数据集	131	7.10 注释和参考	148
7.1.2 成本函数和模型 聚类	131	7.11 总结	150
7.2 k 均值目标函数	132	7.12 练习	151
7.2.1 重写 k 均值成本函数 以对聚类效果进行双 重解释：聚类簇内数 据或分离簇间数据 ...	136	参考文献	153
		第 8 章 层次聚类	155
		8.1 凝聚式与分裂式层次聚类 及其树状图表示	155

8.2 定义一个好的连接距离的几种策略	158	9.5.3 k-NN 规则的误差概率	178
8.2.1 一个用于凝聚式层次聚类的通用算法 ..	158	9.6 在计算机集群上实现最近邻查询	178
8.2.2 为元素选择合适的基	159	9.7 注释和参考	178
8.3 Ward 合并准则和质心	161	9.8 总结	180
8.4 从树状图中获取平面划分 ..	162	9.9 练习	181
8.5 超度量距离和进化树	163	参考文献	182
8.6 注释和参考	164	第 10 章 基于核心集的高维快速近似优化和快速降维 ..	183
8.7 总结	165	10.1 大规模数据集的近似优化	183
8.8 练习	166	10.1.1 高维度的必要性示例	183
参考文献	167	10.1.2 高维度上的一些距离现象	184
第 9 章 有监督学习: k-NN 规则分类的理论和实践	169	10.1.3 核心集: 从大数据集到小数据集	184
9.1 有监督学习	169	10.2 核心集的定义	184
9.2 最近邻分类: NN 规则	169	10.3 最小闭包球的核心集	185
9.2.1 最近邻查询中欧几里得距离计算的优化方法	170	10.4 一个用来近似最小闭包球的简单迭代启发式方法 ..	186
9.2.2 最近邻(NN)规则和 Voronoi 图	171	10.4.1 收敛性证明	187
9.2.3 利用 k-NN 规则通过表决来增强 NN 规则	172	10.4.2 小闭包球和用于 SVM 的边缘线性分离器	189
9.3 分类器性能评估	173	10.5 k 均值的核心集	189
9.3.1 误判错误率	173	10.6 基于随机投影矩阵的快速降维	189
9.3.2 混淆矩阵与真/假及阳性/阴性	173	10.6.1 维数灾难	189
9.4 准确率、召回率和 F 值	174	10.6.2 高维度任务的两个示例	190
9.5 统计机器学习和贝叶斯最小误差界	174	10.6.3 线性降维	191
9.5.1 非参数概率密度估计	175	10.6.4 Johnson-Lindenstrauss 定理	191
9.5.2 误差概率和贝叶斯误差	176	10.6.5 随机投影矩阵	191

10.7	注释和参考	192	11.2	判断(子)图同构	200
10.8	总结	193	11.2.1	枚举算法的一般原则	201
10.9	练习	193	11.2.2	Ullman 算法: 检测子图同构性	202
	参考文献	193	11.2.3	枚举算法并行化	203
第 11 章	图并行算法	194	11.3	注释和参考	204
11.1	在大图中寻找(最)稠密子图	194	11.4	总结	204
11.1.1	问题描述	194	11.5	练习	205
11.1.2	最稠密子图的复杂度和一个简单的贪心启发式算法	195		参考文献	205
11.1.3	最稠密子图的并行启发式算法	198	附录 A	笔试	206
			附录 B	SLURM: 集群上的资源管理器和任务调度器	216

第一部分

基于消息传递接口的高性能计算

- 第 1 章 走进高性能计算
- 第 2 章 MPI 简介：消息传递接口
- 第 3 章 互连网络的拓扑结构
- 第 4 章 并行排序
- 第 5 章 并行线性代数
- 第 6 章 MapReduce 范式

第 1 章

走进高性能计算

1.1 什么是高性能计算

高性能计算(High Performance Computing, HPC)是一个综合的领域,包括各种并行编程范式、与各范式相关的编程语言和应用编程接口(Application Programming Interface, API)、定制的软件工具、专门的国际会议(ACM/IEEE Super-Computing, SC)等。笼统地讲, HPC 是一个在科学和技术上研究超级计算机(Super Computer, SC)的领域。

世界排名前 500 的超级计算机榜单(简称 Top500[⊖])会定期更新并发布在互联网上。在 2014 年,由中国国家超级计算广州中心研制的“天河二号”超级计算机(英文翻译为 MilkyWay-2)获得了该榜单的第一名。该超级计算机包含了惊人的 312 万个核,每个核是一个处理单元(PU),整体性能达到了 54.9PFlops, PFlops 表示每秒 10^{15} 次浮点运算。这个排名第一的超级计算机需要 17.8MW 的电力才能正常工作!粗略地算, 1MW 的电力大约花费 100 美元/时,这意味着该超级计算机每年的电费大约是 1 百万美元。

表 1-1 总结了(超级)计算机的处理能力和内存大小对应的规模数量级。如今,国内外的各超级计算团队正在研制 Exaflop 性能的超级计算机(10^{18} Flops, 1024PFlops),期望在 2017~2020 年间实现这一目标,紧接着,在 2030 年让超级计算机的性能迎来 zetaFlops(10^{21})时代,然后是 yottaFlops(10^{24}),并一直保持性能不断提升。

表 1-1 超级计算机的计算性能和内存大小所对应的规模数量级:超级计算机是根据 Flops (每秒的浮点运算次数)和内存的字节(8 位为 1 字节)容量进行排序的

单位	规模	计算性能	内存大小(字节)
K (kilo)	10^3	KFlops	KB
M (mega)	10^6	MFlops	MB

⊖ <http://www.top500.org/>。

(续)

单位	规模	计算性能	内存大小(字节)
G (giga)	10^9	GFlops	GB
T (tera)	10^{12}	TFlops	TB
P (peta)	10^{15}	PFlops	PB
E (exa)	10^{18}	EFlops(2017~2020年)	EB
Z (zeta)	10^{21}	ZFlops	ZB
Y (yotta)	10^{24}	YFlops	YB
...
googol	10^{100}	googolFlops	googol 字节
...

注: 当选用 $1024=2^{10}$ (2 的次幂) 来代替 $1000=10^3$ 作为两个连续的数量级之间的倍数时, 我们可以得到以下的国际单位(SI): Ki(2^{10})、Mi(2^{20})、Gi(2^{30})、Ti(2^{40})、Pi(2^{50})、Eo(2^{60})、Zo(2^{70})和 Yi(2^{80})

这个传统的超级计算机评价标准只依据算术运算的峰值计算速度, 而完全忽略了为了达到这样的计算速度所消耗的能源。另外一个绿色的榜单叫作绿色 HPC[⊖] (green HPC), 每年评比两次, 该榜单根据超级计算机的 MFlops/W 对各超级计算机进行评比。在 2014 年 11 月, 来自 GSI 亥姆霍兹中心(德国, 达姆施塔特)的 L-CSC 超级计算机取得了 5.27 GFlops/W 的性能。比照来看, L-CSC 超级计算机在 Top500 排行榜中列 168 位, 共 10 976 个核, 并且其峰值计算速度达到了 593.6 TFlops。尽管处理能力是一个非常重要的评价标准(同时也是激励研究人员不断开发 HPC 解决方案的重要因素), 但同时我们也需要考虑其他因素, 例如整体的内存大小、互网络的带宽等。最后, 我们需要指出每 GFlops 的花费正在呈指数下降, 在 2015 年 1 月, 每 GFlops 估计花费 0.08 美元。

1.2 为什么我们需要 HPC

对此, 浮现在人们脑海中的第一个答案是 HPC 有助于更快、更加准确地(对于模拟类的应用, 例如天气预报、汽车碰撞测试中的计算力学, 或者其他各种复杂现象的建模)运行程序。HPC 还能解决更大规模的问题: 在更细粒度的网格或者更大的数据集(大数据趋势)上进行模拟。但是很少有人知道 HPC 同样有助于节约能源: 在相同的浮点计算性能下, 我们更倾向于使用低功耗的处理器, 因为高功耗的处理器需要消耗更多的能源才能正常工作。最后一点, HPC 非常适合某些在本质上具备天然并行性的算法。实际上, 在图像或视频处理领域用到的算法经常需要进行滤波计算, 这种运算是针对每个像素或体素(医学图像中)独立进行的, 因此滤波计算可以并行执行。在后一种情况下, 图像显卡(图像处理单元, GPU)是一个众核(现今已经有几千个核)硬件显卡。举例来说, 一个高端的 NVIDIA 显卡拥有 1536 个核, 计算速度能够达到 2.3 TFlops。

⊖ <http://www.green500.org/>。