南京大学

本 科 毕 业 论 文

院　　系　地球科学与工程学院

专　　业　地质学

题　　目　MPI/Hadoop 在 K-means 算法上的性能比较分析

年　　级　2011 级　学 号 111150034

学生姓名　蒋鑫

指导老师　周会群　职 称　教授

论文提交日期　2015 年 6 月 19 日

南 京 大 学

本科生毕业论文（设计、作品）指导情况记录

开题简况

题目：MPI/Hadoop在K-means聚类分析算法上的性能比较分析

1、选题质量（简述选题与专业培养目标、专业要求关系、题目难度、工作量、创新性、理论性、实用性）

基于学生对未来职业的选择和个人发展的考量批准此题。大数据分析方法过去一直用于地球科学领域（数字地球），如今更是成为全社会的热门话题。对此领域的重要算法-聚类分析在云同平台上进行性能分析评价，可以成为大数据分析平台发展的重要参考。中上难度，需要系统构建、并行程度设计等多方面的综合知识。

2、开题意见：

同意开题。重点放在海量数据、高精度分析的性能比较上。

指导教师签名：周会平

2015年 1 月 17 日

中期检查

指导教师检查论文的进展情况：（指导和培养学生查阅文献资料、综合运用知识、研究方案设计、研究方法和手段运用和外文应用等能力简况）

已完成文南大调研工作。必要时启补充外文文献的阅读。已完成计算集群的集成（主控服务器1加计算服务器4），并已完成基础软件的安装和调试。正在两种平台上，聚类算法的并行程序设计。注意算法的边设计边调优。

指导教师签名：周会平

2015年 4 月 5 日

南 京 大 学

本科生毕业论文（设计、作品）指导教师评阅意见

指导教师评语： 大数据分析的理论基础是概率论中的中心极限定理，方法则归属于多元统计分析。过去以及现在大量应用于科学研究领域，其目的是探明随机现象，即不确定性现象的统计规律。近年来，大数据分析的概念被过度地高出包装，相当多的分析平台偏离了正确的发展方向。蒋鑫同学的论文以高性能计算平台为基础，利用过去几十年科学计算领域积累下来的经验、技术和软件资产，与目前非常流行的大数据分析平台 Hadoop 作了比较研究。通过搭建两种不同分析平台，对机器学习中最常用的 K-means 算法，利用相同的数据，进行了细致的性质评估。以显见的事实说明大数据分析的未来发展方向就是高性能计算。论文立意正确、分析合理、实验设计周到，结论有重要意义。是一篇优秀的大学毕业论文。

指导教师签名： 周志成

2015 年 6 月 12 日

南 京 大 学

本科生毕业论文（设计、作品）评阅教师评阅意见

评阅教师评语：

蒋鑫同学的毕业论文“MPI/Hadoop 在 K-means 算法上的性能比较分析”是对地球科学中的海量计算应用研究与计算机科学中流行的并行计算、分布式计算的一次结合。选题十分新颖，是地球科学的前言领域，应用性较强。该生在毕业论文完成期间，态度端正，仔细认真，能够独立设计出合理实验并进行了相关项目的测试，表现出了较强的专业素养和创新能力。从最终的论文中可以看出，该生查阅了国内外的大量相关文献，具备了一定的文献综述和资料整理能力。同时论文内容比较完整，思路清晰，观点突出，逻辑性强，在论文最后对所得到的结果进行了比较详尽的分析解释，并得出了令人信服的结论。表明该生具备了一定的独立工作能力，达到了南京大学地科院本科培养的目标。

评阅教师签名： 刘国东

2015 年 6 月 1 日

南 京 大 学

本科生毕业论文（设计、作品）答辩记录、成绩评定

答辩记录：

1. 进行测试的软件是哪一款？

使用 Hadoop-2.6 和 openmpi-1.8.4 进行试验，实验数据使用随机算法进行生成。运行环境是 CentOS6.5。

2.测试结果在地质学模拟上的应用前景和潜力。

从测试结果来开，对于地学中常见的复杂迭代性计算，使用 MPI 相对于 Hadoop 有着比较大的性能优势，所以对于今后的地质学模拟中的海量计算可以更多的考虑基于 MPI 的高性能计算模型，例如论文中所提的油气储藏量预测，高精度地震资料处理等。

答辩记录人签名： 黄晓路

答辩小组评语：

该生的研究方向比较新颖，希望对地学研究和计算机科学中的并行计算进行结合，通过答辩可以看出该生进行了比较深入的研究，并取得了一定的实验成果。对于答辩老师提出的问题，能够比较合理的解释说明，思路比较清晰，有理有据，达到了我院对本科毕业论文答辩的要求。

答辩小组成员： [illegible] 董少春 [illegible]

成绩 87

组长签名： [illegible]

答辩时间： 2015年 6月 8日

南京大学本科生毕业论文（设计、作品）中文摘要

题目： MPI/Hadoop 在 K-means 算法上的性能比较分析

院系 地球科学与工程学院 专业 地质学

2011 级本科生 姓名： 蒋鑫

指导教师（姓名、职称）： 周会群 教授

摘要：

过去几十年，由于计算机软硬件的进步，已有相当数量的科研领域开始利用计算机进行辅助处理、计算和模拟。然而随着现代科学研究的不断深入，处理数据海量化、计算过程复杂化、结果要求精细化，导致了很多情形下单机计算模型已经不能满足客观需要，并行计算成为解决这一问题的关键。

基于消息传递的 MPI 编程模型和基于 HDFS、Mapreduce 的 Hadoop 编程模型是当前比较热门的两种并行编程框架。本文首先介绍了并行计算领域的一些基本概念，讨论了并行计算在实际科研中的意义以及常见并行计算机系统；然后概要的介绍了 MPI、Hadoop 两种并行模型的特点和基本使用方式；接着对地学研究中比较常用的 K-means 聚类算法进行简述；最后通过设计实验，将 MPI 和 Hadoop 同时作用于 K-means 算法，得到一系列不同维度的实验结果，对不同数据量，不同计算精度，不同集群规模所得到的两者性能数据进行了较详细的分析讨论，并指出了性能差异的可能原因，进而对两者的最适应用场景进行了描述，希望可以对今后的科研计算起到建议性作用。

关键词：并行计算； MPI； Hadoop； Mapreduce； K-means

南京大学本科生毕业论文（设计、作品）英文摘要

THESIS：Comparative Analysis of MPI / Hadoop performance on the K-means algorithm

DEPARTMENT：School of Earth Sciences and Engineering

SPECIALIZATION: Geology

UNDERGRADUATE: Jiang Xin, 2011

MENTOR: Zhou Huiqun, Professor

ABSTRACT：

Over the past decades, due to the great progress in hardware and software, there has been a considerable amount of research fields that began to use computer-assisted processing, calculation and simulation. However, with the deepening of modern scientific research, data processed gets bigger, calculation process gets more complicated, and the results requires more accurate. In many cases the stand-alone compute model cannot meet the requirement. So parallel computing has become the key to solving this problem.

Nowadays, MPI based on message passing and Hadoop based on HDFS/Mapreduce are two popular parallel programming frameworks. This paper first introduces the basic concepts of parallel computing, discusses the significance of parallel compute in actual research and the varieties of ordinary parallel computers; in the next chapters describes the characteristics and basic usage of MPI and Hadoop parallel model; then this thesis gives an introduction of K-means clustering algorithm, a commonly used algorithm in geoscience and other fields; and finally we designs experiments to make the MPI and Hadoop process K-means algorithm. A series of experimental results obtained in different amount data, different calculation accuracy and different cluster size. I give a detailed analysis and discussion on these experiment results and point out the possible reasons for the differences in performance. I hope this thesis can give some scientific computing research an effective guide.

KEY WORDS: Parallel Computing; MPI; Hadoop; Mapreduce; K-means

目录

1. 绪论 1

1.1 研究背景 1

1.2 国内外关于该课题的研究现状 2

1.3 本文的结构 4

2. 并行计算基础理论 5

2.1 并行计算 5

2.2 并行计算的意义 5

2.3 并行计算机系统 6

3. MPI 与 Hadoop 简介 7

3.1 MPI 简介 7

3.2 Hadoop 简介 8

4. K-means 算法及实验设计 11

4.1 K-means 聚类算法概述 11

4.2 K-means 算法基本步骤 11

4.3 K-means 算法复杂度分析 11

4.4 基于 Hadoop 的并行 K-means 算法设计 12

4.4.1 Map 函数实现 12

4.4.2 Reduce 函数实现 12

4.5 基于 MPI 的并行 K-means 算法实现 13

4.5.1 主进程的主要任务 13

4.5.2 从进程的主要任务 13

4.6 实验环境说明 13

5. 实验结果 15

5.1 使用 Hadoop 运行 K-means 算法性能分析 15

5.2 使用 MPI 运行 K-means 算法性能分析 17

5.3 Hadoop 与 MPI 在 K-means 算法中的性能比较 24

6. 结论 28

参考文献 29
致谢. 31

1. 绪论

1.1 研究背景

随着科学技术的迅猛发展，越来越多的科研领域采用计算机作为辅助存储和运算工具，然而随着各项研究的深入，复杂化、精细化、智能化已经成为当今科学计算的主流，传统的单机存储、计算模型已经难以满足科研人员处理海量数据、海量计算的工作需要，特别是在天体物理学、分子生物学、地震学、材料化学、地理学等领域，该问题尤其突出。

以地质学中的矿产预测为例，由于矿产资源预测理论的持续进步，相关模型的不断完善，以及与其它地学信息的不断融合，以矿产资源预测理论为核心，以空间数据库为根基的矿产资源预测系统不断涌现[1]。由于该系统涉及到地学大数据融合分析、跨学科数据集成、矿产资源定位预测、地质空间建模等一系列复杂过程，所以涉及到的计算量十分庞大，传统的单机计算系统已经难以维持，必须使用现代的高性能计算或者分布式云计算系统进行支撑，以实现对地学大数据的快速处理、准确分析。因此基于现代云计算模型的矿产资源预测系统已经成为前沿研究方向。另外在据曲寿利[2]等人有关地震资料勘探的研究中，也表达了相似的观点。由于勘探程度的持续提高，勘探对象也日趋复杂，矿产勘探的当前形势可以用“深，隐，低，稠，小”五个字进行概括，也就是说勘探对象埋深不断加大；勘探目标隐蔽性逐渐增加；低渗透性、特低渗透性油气藏所占的比例上升；稠油、超稠油的油藏比例持续增大；勘探对象的单元储量规模却越来越小。所以为了适应目前这种趋势，地震资料勘探的精度要求大幅提高，而精度的提高必然带来计算量的陡然上升，这之中高性能计算，云计算在其中起到了关键性的作用。

传统高性能计算采用基于并行处理器为基础的超级计算机对复杂的现实世界进行计算模拟。随着相关科技的发展，现代高性能计算已经有了新的定义，开始融入集群计算、分布式计算、网格计算、普适计算等一系列新的技术。正是因为有这种以高性能计算为基础的地学计算，地理地质环境中以时空演变为特征的全球性或者大区域性现象的模拟（如地理信息系统、全球气候变化、数值天气预报、图像解译和遥感定量解译等）才能真正实现；原有的一些受单机条件限制而难以

进行的模型才能得以完善、改进和运行[3]。

目前高性能计算模型层出不穷，各种新技术令人眼花缭乱，但其中最著名的两个流派就是 MPI 并行计算模型和 Hadoop 分布式计算模型。

MPI(Message Passing Interface)是一种使用非常广泛的基于消息传递的并行编程模型。除了广泛应用于高校科研场所之外，MPI 还用在许多商用计算机系统之中。互联网的存在意味着每两个计算机处理器之间都直接或间接的相连，也就是说每个处理器都可以和其它任何一个处理器进行通讯。而 MPI 并行编程模式就是依赖这种特性，它可以使集群中的任何两台计算节点在计算过程中进行消息传递，最大限度的利用每一台机器的处理器计算能力。

Hadoop 是一种新兴的分布式基础架构。用户可以在不了解分布式底层细节的情况下开发分布式程序。充分利用集群的威力进行高速运算和海量存储。Hadoop 主要由 HDFS 和 Mapreduce 两大核心组成。HDFS 处于底层，Mapreduce 则利用 HDFS 高容错，高可靠，易扩展等特点，在其上构造了一个分布式并行计算引擎。Mapreduce 将分布式并行编程抽象为两个简单的基本操作（map 操作和 reduce 操作），开发人员仅需要实现对应的接口即可，很好的屏蔽了其底层的容错、数据流、程序的并行执行等细节。这种设计无疑大大降低了开发分布式并行程序的难度。

1.2 国内外关于该课题的研究现状

目前 MPI 和 Hadoop 这两种并行计算模型都发展的比较成熟，其中 MPI 已经在地质学相关的高性能计算中得到了广泛的应用。中国地质大学的学者利用 MPI 开展了大地电磁三维正反演并行算法的研究[4][5]，上海超算中心的部分学者则利用 MPI 进行了黄河下游二维水沙数学模型的构建[6]，并且已经取得了初步的研究成果。由于 Hadoop 计算模型出现较晚，当前在地学领域的应用还不常见，但已有学者开始尝试，比如 Chaowei Yang 等人希望利用 Hadoop 的 Mapreduce 计算框架进一步论证数字地球的可行性[7]。

虽然 MPI 和 Mapreduce 并行编程模型都属于高性能计算的范畴，但是由于两种模型差异较大，各有优缺点。到底哪种并行编程模型更加适用于地学的普遍应用场景，目前还没有一个定论。

MPI 是一种基于消息传递机制的并行编程规范，它在程序设计上非常简单而且符合普通科研人员的使用习惯。但是 MPI 有一个比较大的缺点，即底层缺乏一个优秀的分布式文件系统对其进行很好的容错支撑。数据处理节点和数据存储节点往往是相异的，这种数据的存储和处理分离的情况要求在每次计算开始时从集群的数据存储节点读取数据分配给各个计算节点，然后再进行接下来的数据处理。因此对于计算密集型的应用 MPI 能表现出相当良好的性能，但对于处理数据密集型应用，由于大量的数据需要在节点之间进行传递交换，网络 IO 时间将成为影响系统性能的瓶颈，用“计算换通信”也是 MPI 并行程序设计中的基本原则。当然这种情况正在改观，已经出现一些分布式文件系统开始在底层支持 MPI 模型，但是其中的大部分与 HDFS 相比在容错性、可扩展性等方面差距明显[8]。

在 Hadoop 中利用 HDFS 可以实现分布式存储，这就意味着可以使各个节点直接读取存储在本地的数据进行处理计算，避免在网络 IO 上损失过多的性能，实现了“计算向存储迁移”。在数据密集型的应用中会有比较好的表现，已有研究表明 Hadoop 在 PB 级数据存储和处理中，仍然表现优良[9]。同时 Hadoop 可以很好的应对节点失效或者网络的通信中断情况。如果出现了异常情况，HDFS 的备份和容错机制可以保证应用的持续、稳定进行，而不用重新开始所有的计算处理，因而可靠性得到了保证。但是由于 Hadoop 对于并行模型的过于封装，导致人们很难实现应用特定的适配性调整。特别是在一些科学计算领域，很难深入到底层进行灵活控制。另外 Hadoop 的核心使用 Java 语言编写，虽然 Java 也是一种运行性能较高的语言，但是与 C 或者 Fortran 相比仍有较大差距，而后者通常是 MPI 的宿主语言。

为了比较两者哪个更适用于地学研究中的高计算量应用，本文通过实验比较了两者在地学研究中常见的 K-means 算法中的性能。并对实验结果进行了较为详细的说明和解释。

K-means 算法是聚类分析中非常重要的一种算法，而聚类分析又在地学研究中普遍使用。例如王龙、王晓青等人使用 K-means 进行了水库地震烈度衰减的研究[10]。牛瑞卿等人“利用数据挖掘的滑坡监测数据处理流程”[11]中也多次使用了 K-means 算法。北京邮电大学的段炬霞提出可以使用 K-means 算法对地质勘探地震数据进行聚类，并尝试改进该算法以提高其有效性和适用性[12]。由此可见 K-means 算法在地学研究中有着广泛的应用场景，所以本文通过对该算法在 MPI

和 Hadoop 两种并行编程模型中的实际性能进行比较分析，希冀为未来的地学高计算量应用提供参考性建议。

1.3 本文的结构

本文通过对 MPI 并行计算模型和 Hadoop 分布式计算模型在 K-means 聚类算法方面的性能进行比较分析，探讨两者的性能差异，为今后的科学研究特别是地学研究中的高计算量应用提供一定的参考。

本文各章节的安排如下：

第 1 章（绪论）概括性的描述了本选题的研究背景和国内外关于该课题的研究现状，并给出了本文的结构和安排。

第 2 章（并行计算基础理论）主要介绍了并行计算的一些基础概念，并行计算的主要意义以及常见并行计算机系统。

第 3 章（MPI 和 Hadoop 简介）分别描述了 MPI 和 Hadoop 两种并行编程框架的基本内容和主要特点。

第 4 章（K-means 算法及实验设计）详细的介绍了 K-means 算法的内容，基本算法步骤，然后分别给出了 Hadoop 和 MPI 的 K-means 算法设计，最后对实验环境进行了简单说明。

第 5 章（实验结果）分析了 MPI 和 Hadoop 两种并行编程模型在 K-means 算法方面的性能差异，并做了比较详细的解释。

第 6 章（结论）对实验结果和性能分析进行概括性总结。

2. 并行计算基础理论

2.1 并行计算

并行计算是指同时使用多种计算资源对多任务、多指令或者多数据项进行处理，解决计算问题的过程[13]，并行计算是提高计算机处理能力的一种有效手段。其核心思想就是要把一个任务分解成多个子任务，利用多处理器的实时协作，共同求解该问题，从而达到快速求解大规模应用问题的目的。目前并行计算及其思想被认为是科研工作者和工程师用来解决各种复杂领域的问题的标准方法，如地质学中的古气候还原，天文学中银河系的演变过程，化学中分子动力模拟等一系列需要海量计算的问题[14]。

2.2 并行计算的意义

当前科研和生产之中大规模使用并行计算的原因就是传统的单机环境已经难以满足问题求解的要求。包括计算时间、计算精度、实时响应等一系列的客观需求。下面分别阐述并行计算在这些方面的优势：

首先来讲，使用并行计算可以提高问题求解的速度，进而缩短计算时间。在计算负载恒定不变的情况下，通过并行计算将处理任务分散在超级计算机的各个 CPU 上或者集群中的不同主机上，显然可以降低每个处理单元的荷载，提高执行速度，降低计算时间。在很多情况下，这种执行速度的提高会带来质的改变，使一些原本无法进行的实验得以进行。

其次，使用并行计算可以提高计算精度。在许多工程计算中，为了获得足够高的计算精度，就需要稠密的计算网格，而网格的稠密程度直接影响到计算量的大小。在很多时候这种复杂的大规模计算问题必须依靠并行计算才可以实际运行。以油藏模拟为例，截止 2010 年，胜利油田就有油井 20071 口，为了获得比较精确的模拟精度，假定没每口油井取 8×8×50 个点[15] ，在这种情况下总的观察点数目达到千万级别，对于这种计算量，单机很难处理，必须依靠并行计算系统。

另外，使用并行计算可以提高时效要求。对于实时响应要求比较高的科学问题，一旦得到最终结果的时间超过了阈值，求解问题就失去了意义。例如在天气

预报中，要得到可用的预测分析，往往需要取几十万个网格点，总数据量在 TB 级别[16]，且要求在 2 小时内完成全部计算，才能获得可以利用的未来 24 小时的天气预报。

2.3 并行计算机系统

上世纪 60 年代，Flynn 根据数据流和指令流的特点将计算机系统分为以下四类：

单指令流单数据流(SISD, Single Instruction Stream Single Data Stream)

单指令流多数据流(SIMD, Single Instruction Stream Multiple Data Stream)

多指令流单数据流(MISD, Multiple Instruction Stream Single Data Stream)

多指令流多数据流(MIMD, Multiple Instruction Stream Multiple Data Stream)

这里所指的指令流是指计算机所执行的指令序列。同样，数据流指的是计算机所使用的数据序列。我们这里所说的并行计算机一般只包括 SIMD 和 MIMD，特别是近十年来，MIMD 类型的并行机逐渐占据主导位置。当代 MIMD 主要包括以下 4 类：

- 对称多处理机(SMP, Symmetric Multiprocessor)
- 大规模并行处理机 (MPP, Massively Parallel Processor)
- 分布式共享存储多处理机(DSM,Distributed Shared Memory)
- 工作站机群 (COW, Cluster of Workstation)

由于并行计算机系统是个很庞杂且不断发生变化的概念，因此本文不再详述，更多介绍，可参考郭庆平等人的著作[17]。

3. MPI 与 Hadoop 简介

3.1 MPI 简介

MPI(Message Passing Interface)是由科研机构和工程部门共同设计的一套基于消息传递并行编程模型的标准规范，使用标准的 MPI 规范编写的程序，可以不加修改的运行在异构的计算机集群上。目前比较知名的 MPI 实现是 MPICH 以及 OPENMPI，它们都已经支持最新的 MPI 3 标准，并都经过了严格的功能和性能测试。一般来说 MPI 的实现都会提供 C，C++和 Fortran 的语言绑定，因为这三种语言编写出来的程序运行速度较快，而且对于科研人员也比较熟悉。但是现在已有组织和个人开始尝试 MPI 的 GoLang，RUST 等新兴并发语言的绑定。

MPI 的核心是消息传递模型。如图 3-1 所示，一组计算节点通过网络相连，每个节点拥有独立的处理器和内存，每个处理器既能访问本地内存中的数据，又能通过互联网络获得其它计算节点发送过来的数据。在运行程序时，用户需要指定并发进程的数量，每个进程拥有唯一的标识 ID。每个进程之间在程序开展之后可以执行差异化的操作。或者仅针对其局部变量进行操作，或者与其它进程进行通信，这两个过程可以交替进行。

进程之间互相传递数据（消息）有两个目的，一是相互通信，使数据可以从一个节点传送到另一个节点。二是进行同步确认，通过节点之间的消息传递，一个进程可以获得其它所有进程的当前状态，因此在某些情况下，即使一个空消息也是有意义的。

需要注意的是，由于消息传递模型访问本地数据相对于访问远程节点数据速度要快的多，所以我们在实际开发中还是要使本地化操作最大化，通信操作最小化。

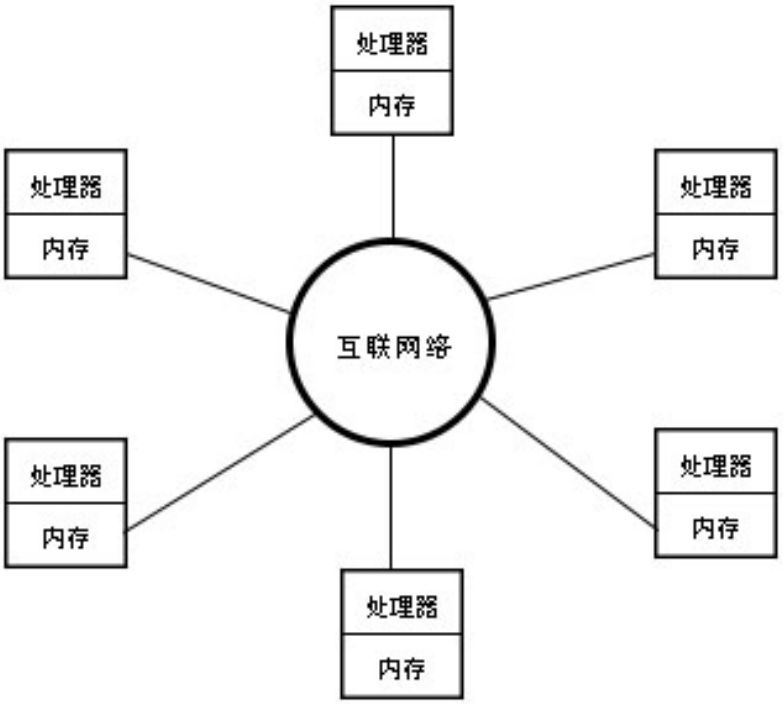

图 3-1 消息传递模型示意图

MPI 的并行编程主要依靠一组消息传递库函数进行。这组函数可以在不同的运算机器上进行数据交换，提供了并行任务间的数据发送，接收和同步接口。MPI 有两种基本的并行程序设计模式：主从模式和对等模式。这两种模式可对应于 Web 开发中的 C/S 模式和 C/C 模式。

主从模式：该模式中一个主进程(master)对整个并行程序进行监控和协调，各个从进程(slave)在主进程的控制下对数据进行并行的处理和运算。这是当前的并行程序使用比较多的一种模式。也是本实验所采用的方式。

对等模式：并行程序中的各个进程的地位完全等价，不同的只是它们处理的数据对象有所不同。

3.2 Hadoop 简介

Hadoop 是一个由 Apache 软件基金会开发的一个开源分布式系统基础架构。Hadoop 允许用户在不了解分布式系统底层细节情况下，开发分布式应用程序，充分利用分布式集群进行高速运算和存储。如图 3-2 所示，Hadoop 主要由两部分组

成（图中黄色部分），底层是一个分布式文件系统 HDFS（Hadoop Distributed File System），上层是 Mapreduce 并行编程框架。另外由于 Hadoop 在海量数据存储运算上拥有着无可比拟的优势，现在围绕 Hadoop 已经有了一个完整的开发，运维生态系统（图中非黄色部分），但这不是本文讨论的重点，本文主要探讨 Hadoop 中核心组件 HDFS 和 Mapreduce 的特点。

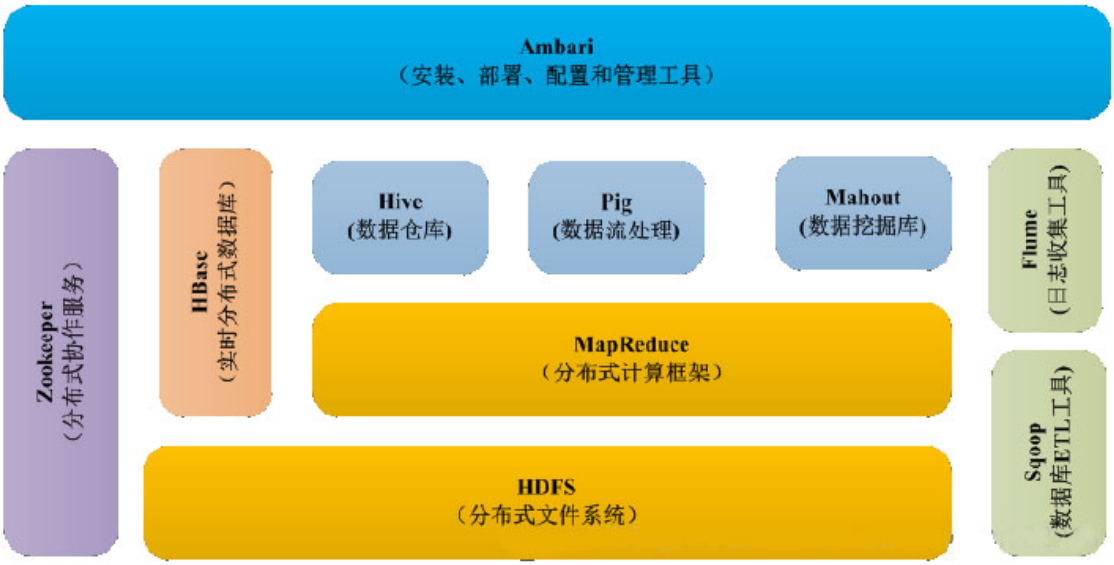

图 3-2 Hadoop 体系结构

HDFS 有高容错的特点，被设计用来部署在低廉的硬件上，而且它提供高吞吐量来访问应用程序的数据，适合那些有着超大数据集的应用程序。HDFS 放宽了 POSIX 的要求，可以以流的形式访问文件系统中的数据。Hadoop 是一个能够让用户轻松架构和使用的分布式计算平台。用户可以轻松地在 Hadoop 上开发和运行处理海量数据的应用程序。概括来讲主要有以下几个优点：

高可靠性。Hadoop 存储和处理数据的能力值得人们信赖。

高扩展性。Hadoop 是在可用的计算机集簇间分配数据并完成计算任务的，这些集簇可以方便地扩展到数以千计的节点中。

高效性。Hadoop 能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此处理速度非常快。

高容错性。Hadoop 能够自动保存数据的多个副本，并且能够自动将失败的任务重新分配。

低成本。与一体机、商用数据仓库以及 QlikView、Yonghong Z-Suite 等数据集

市相比，hadoop 是开源的，因此项目的软件成本会大大降低。

如图 3-3 所示，在使用 Mapreduce 进行科学计算时，我们首先需要将输入数据进行分片，然后发送给不同的节点进行 map 操作，Map 函数接受分片数据并将其转换为一个 Key/Value 列表，分片中的每个元素对应一个 Key/Value。接着这个 Key/Value 列表会被发送到 reduce 节点进行规约操作，Reduce 函数根据它们的键缩小键/值对列表。

当然在整个过程中，还包括一些额外操作，比如在 Map 操作之后按照 Key 值进行排序，还可以设置本地 Reduce 操作（Combine）来加快规约速度。

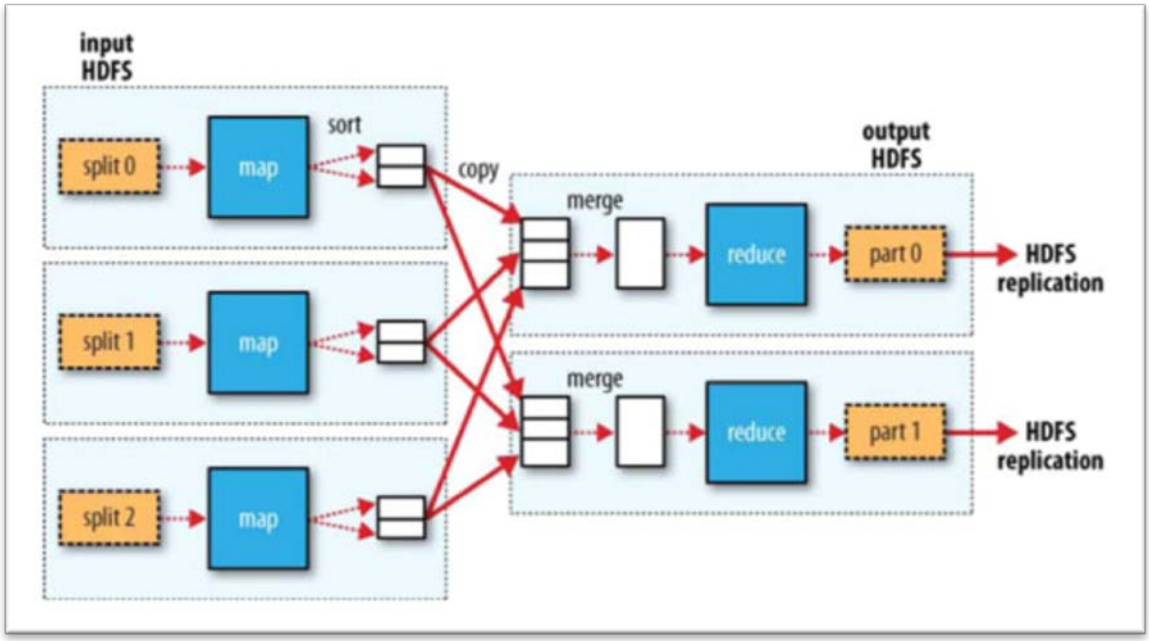

图 3-3 Mapreduce 原理示意图

4. K-means 算法及实验设计

4.1 K-means 聚类算法概述

K-means 算法是数据挖掘过程中一种重要的聚类算法。K-means 聚类要求将 n 个数据对象分配到 k 个分区中，并且要求分区满足以下条件：在同一个分区中的数据对象相似度要达到最大，不同分区中的数据对象的相似度要达到最小。该算法的形式化描述如下：

对于给定的一组数据对象集合$(x_1, x_2, x_3 \ldots, x_n)$，每个数据对象都是一个 d 维的实数向量，K-means 聚类算法就是要把这 n 个数据对象分配到 k 个分区之中 $S = \{S_1, S_2, S_3, \ldots S_k\}$，使得下式的值最小：

$$\sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

其中μ_i是S_i的分区中心，与数据对象相同，也是一个 d 维的实数向量。

4.2 K-means 算法基本步骤

K-means 算法的具体实现步骤如下：

a) 从 n 个数据对象集合中随机选择 k 个数据对象作为分区中心；

b) 依次对集合中的每个数据对象进行分区操作，即分别求该对象与 k 个分区中心的欧式距离（或者其它需要的度量），得到一个最小距离，然后将数据对象划分到该分区中。

c) 按照每个分区所包含的数据对象，重新计算该分区的分区中心；

d) 计算标准测度函数，当满足一定条件，如函数收敛时或者已经达到所需精度，则算法终止；如果条件不满足则回到步骤 b)。

4.3 K-means 算法复杂度分析

由于我们并不能保证 K-means 算法一定会收敛于全局的最优解，通常该算法会终止于某个局部最优解，该最优解取决于初始分区中心的选择。在实际使用过

程中，通常用不同的分区中心进行多次 K-means 运算，以得到比较优良的结果。

K-means 聚类算法的复杂度是 O（nkt），其中 n 是数据对象的数目。k 是分区数量，t 是迭代的次数。一般情况下，k 和 t 是远小于 n 的。这也就给我们在实际应用中很大的伸缩性。

4.4 基于 Hadoop 的并行 K-means 算法设计

使用 Hadoop 进行并行计算，用户一般只需要进行 Map 和 Reduce 函数的设计。通常包括该函数的输入输出键值类型以及两者的具体逻辑实现。之前已经提过，K-means 算法的主要工作是根据不同数据对象到分区中心的距离大小，把不同的数据对象划到各个分区之中。这些操作是彼此相互独立的，可以实现并行计算，在每次迭代过程中，使用相同的 Map 和 Reduce 操作即可完成。在设计这两个函数之前，我们首先需要在 n 个数据对象中随机选择 k 个数据对象作为初始的分区中心。然后将 n 个数据对象和 k 个分区中心所在的文件保存到 Hadoop 的 HDFS 之上（为了方便叙述，文件分别命名为 cluster.txt 和 centers.txt）

4.4.1 Map 函数实现

首先选取需要的输入键值对，我们按照惯例做法，用 cluster.txt 文件中的每条数据相对于 cluster.txt 文件的起始位置的偏移量作为输入 Key，用表示当前数据对象的 d 维实数矢量作为 Value 值。然后同样读取 centers.txt 文件中数据，得到 k 个 d 维的初始分区中心坐标数据。接着依次算出每个数据对象与各个分区中心的欧式距离，找出其中与该数据对象距离最短的那个分区中心，最终的输出键值对表示为（Key’, Value’），其中 Key’代表与该观察点最近的分区中心，Value’代表该数据对象的坐标。Mapreduce 将这些（Key’, Value’）发送给之后的 Reduce 函数进行处理。

4.4.2 Reduce 函数实现

Reduce 函数接收到 Map 函数发送过来的（Key’, Value’），对所有 Key’相同的 Value’进行求均值操作，也就是将所有 Key’值相同的 Value’进行聚合，然后利用聚

合后的数据求出一个新的分区中心，然后用这个分区中心更新该聚类初始的中心。反映在实际操作中就是更新 centers.txt 文件。如果已经达到了所要求的精度则停止运算，否则进行下一轮迭代。

4.5 基于 MPI 的并行 K-means 算法实现

在基于 MPI 的并行 K-means 算法中，我们采用主从模式进行设计，在我们的程序之中有若干个进程，他们可以通过进程号进行识别，我们假设 0 号进程是我们的主进程，其他进程为从进程。与基于 Hadoop 的并行 K-means 算法实现类似，我们同样拥有两个数据文件，cluster.txt 文件中放置的是所有 n 个数据对象的坐标数据，centers.txt 文件中放置的是随机选取的 k 个初始分区中心的坐标数据。

4.5.1 主进程的主要任务

主进程主要负责初始化 MPI 进程环境，检查文件的合法性，检测集群的状态。然后读取 cluster.txt 文件，获取一些初步的信息，比如 n 的值，各个观察点的维度信息等。然后将数据对象进行划分，尽量保证每个从进程可以分得近似的数据量。然后将划分结果发送给各个从节点，让每个从进程进行计算，当计算完成，主进程收集各个从进程的运算结果得到新的聚类中心。

4.5.2 从进程的主要任务

从进程获得主进程发送过来的数据划分结果，然后读取文件进行计算。首先检查每个属于自己的数据对象，通过计算每个数据对象到分区中心的欧式距离，找到距离他们最近的分区中心。然后进行一次 Reduce 操作，将与某个分区中心最近的所有观察点进行聚合，算出这些这些数据对象的中心点，如果该中心点与出原始分区中心的欧式距离在要求的精度之内则停止运算，否则进行下一次迭代。

4.6 实验环境说明

本次实验的实验环境由 5 台普通商用服务器组建，5 台机器的主机的 IP 地址以及主机名称分别为：

- 10.1.100.106 cluster-ib.ipoib
- 10.1.100.254 compute-ib-0-0.ipoib
- 10.1.100.253 compute-ib-0-1.ipoib
- 10.1.100.252 compute-ib-0-2.ipoib
- 10.1.100.251 compute-ib-0-3.ipoib

其中 cluster-ib.ipoib 的参数为：

- CPU: 4×32 Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
- RAM: 128G
- System: Linux 2.6.32-431.11.2.el6.x86_64
- OS: CentOS release 6.5 (Final)
- NetWork: infinite band

另外 4 台服务的参数为：

- CPU: 2×8 Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.13GHz
- RAM: 8G
- System: Linux 2.6.32-431.11.2.el6.x86_64
- OS: CentOS release 6.5 (Final)
- NetWork: infinite band

在 Hadoop 集群实验中，Hadoop 版本为 Hadoop-2.6.0-rdma。共设有 1 个 NameNode，1 个 SecondNameNode，4 个 DataNode，1 个 ResourceManager（JobTracker），4 个 NodeManager（TaskTracker），其中 cluster-ib.ipoib 作为 NameNode，SecondNameNode，ResourceManager 载体，其余 4 台机器作为 DataNode 和 NodeManager 载体。实验时，Map 和 Reduce 数量均根据集群配置和输入数据规模进行自动选择。集群之间通过 SSH 进行关联。

在 MPI 集群实验中，我们使用 openmpi-1.8.4 版本，五个节点均作为等价计算节点，主节点和从节点在程序运行时自主选择。底层集群之间通过 SSH 进行关联。

所有数据对象均使用随机算法产生，并不代表实际物理意义。

5. 实验结果

5.1 使用 Hadoop 运行 K-means 算法性能分析

在此次试验中，我们考虑不同运算精度、不同输入数据量对于运算时间的影响。运算精度是指经过某次迭代后，新旧聚类中心的欧式距离的偏差。例如：精度为 1 代表新旧聚类中心的欧式距离偏差如果小于 1，即停止迭代，程序终止，否则继续进行下一次迭代。之所以没有直接采用迭代次数作为程序终止条件，是因为 Hadoop 和 MPI 两种编程模型中，迭代相同次数，并不能保证最终输出结果精度完全相同，而在实际科研领域，我们一般只考虑结果，而忽略具体的计算过程。

在 Hadoop 试验中，我们设置精度分别为 1/0.1/0.01。输入数据规模分别为 10 万/20 万/40 万/80 万/160 万，最终在满足精度要求的情况下，将输入数据聚集到 5 个集合中。得到的计算时间与计算精度（数据规模）的关系可见表 5-1：

表 5-1 Hadoop 计算时间与计算精度（数据规模）关系表

精度 \ 数据量	100000	200000	400000	800000	1600000
1	1398.4	1888.9	1909.3	2377.2	3351.9
0.1	8117.4	10951.9	15527.8	28090.3	49565.6
0.01	20495.6	25281.0	42860.0	74050.4	141924.8

为了更清楚的分析性能，我们从不同角度进行绘图，首先考虑 Hadoop 计算时间与计算精度的关系。从图 5-1 可以看到随着计算精度的提高，也就是说随着运算迭代次数增多，运算时间不断增加，这种关系对于不同的数据规模具有普适性。在图中我们还能看出，这种增长的趋势是逐渐加剧的。也就说精度每提高 10 倍，时间的增加量会越来越大，体现在 K-means 算法中，就是随着精度要求的提高，继续提高相同的精度，需要迭代的次数会越来越多。

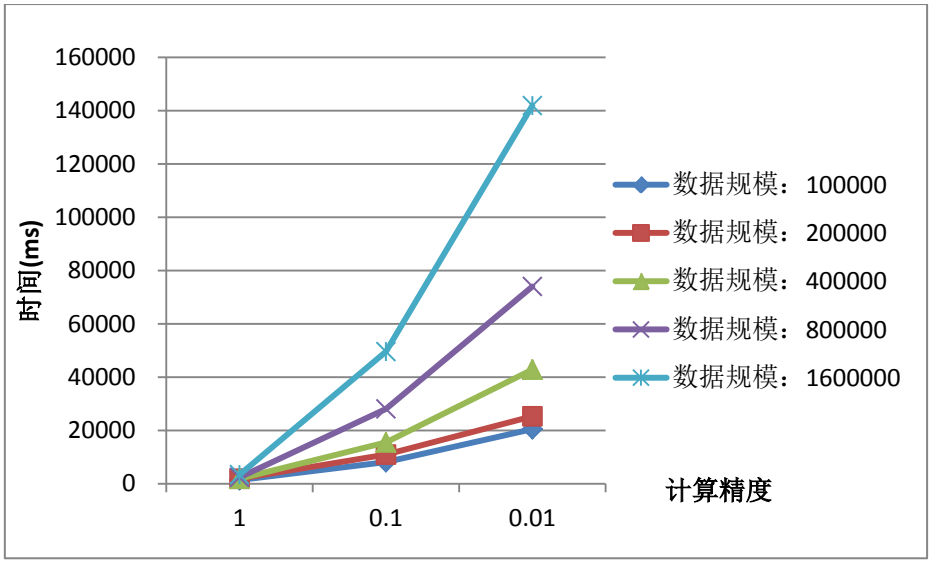

图 5-1 Hadoop 计算时间与计算精度关系图

接下来，我们考虑 Hadoop 计算时间与数据规模的关系，如图 5-2 所示，随着数据规模的增加，K-means 算法在 Hadoop 集群中的运算时间持续增加，结合图 5-3 的对数图观察，很容易发现这种增长趋于线性。也就是说 Hadoop 并行计算模型对于输入数据的增长表现友好。

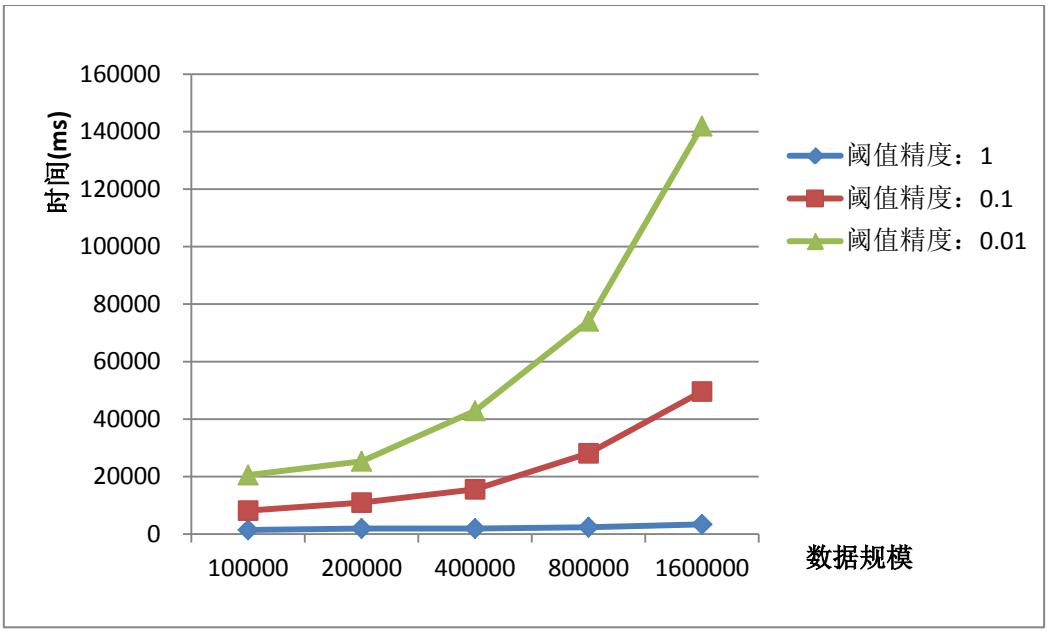

图 5-2 Hadoop 计算时间与数据规模关系图

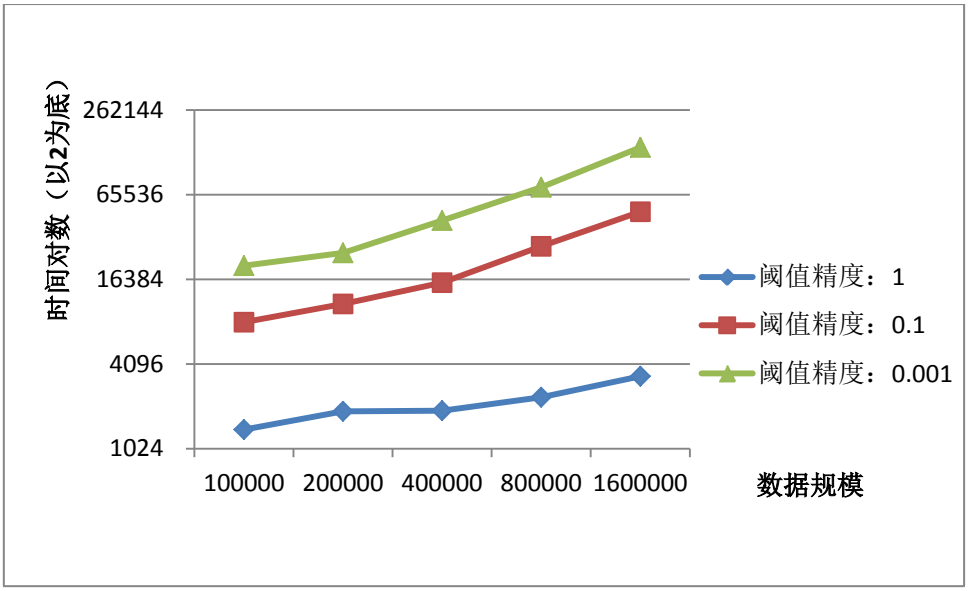

图 5-3 Hadoop 计算时间的对数与数据规模关系图

5.2 使用 MPI 运行 K-means 算法性能分析

为了更全面地分析 MPI 计算模型的性能特性，我们分别统计了在 K-means 算法中的 IO 时间，计算时间，总时间及其与集群规模，数据规模，计算精度的关系。

其中集群规模通过进程数来体现，为了叙述方便，下文将不再区分进程数和集群规模这两个术语。对于数据规模，同 Hadoop 中一样，我们设置了 10 万/20 万/40 万/80 万/160 万 5 组。对于计算精度我们设置了 1/0.1/0.01/0.001/0.0001 共 5 组。

我们首先考虑 MPI 程序中的 IO 时间与集群规模，计算精度的关系。其中具体数据可以参考下表（表 5-2）。

表 5-2 MPI I/O 时间与集群规模（计算精度）关系表

精度 / 进程数	1	0.1	0.01	0.001	0.0001	0.00001
1	516.0	513.3	556.6	522.5	520.7	521.6
2	521.6	519.6	508.0	540.0	522.9	511.0
4	520.4	522.3	539.3	523.0	531.9	530.1
8	517.8	508.4	522.6	529.3	507.0	520.8
16	514.1	517.5	518.1	546.2	527.2	523.6

我们将该表进行绘图，在图 5-4 中我们可以发现 MPI 中 IO 时间与集群规模和阈值精度之间并没有必然关系，当然这与具体算法有关。体现在本实验中，虽然进程数增加，但是每个进程所分到的数据量会减少，导致总的 IO 时间并没有太大变化。另一方便，阈值精度提高，也就是说需要迭代的次数增多，仍没有导致 IO 时间的上升，主要是因为 IO 时间主要是初始化时的文件 IO，在计算过程中传递的信息主要是一些观察点的序号，以及数量很少的中心点坐标，这些导致的网络 IO 与文件 IO 时间相比，所占比例很小。所以即使迭代次数增加，网络传递的次数增加，但是由于基数很小，仍然不会对整体 IO 造成很大影响。

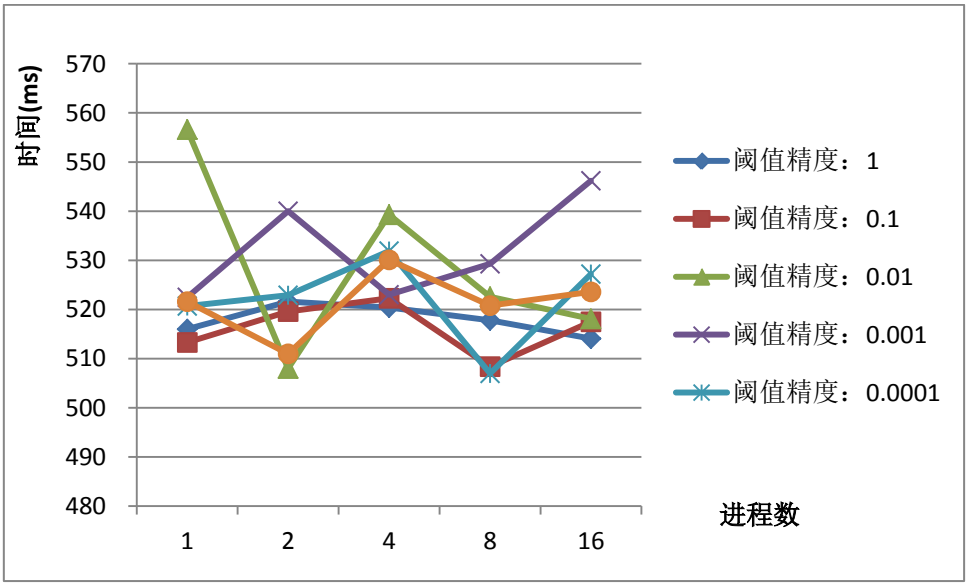

图 5-4 MPI I/O 时间与集群规模关系图

接下来，我们考虑真正的计算时间与集群规模（计算精度）的关系。具体数据可以参考表 5-3：

表 5-3 MPI 计算时间与集群规模（计算精度）关系表

精度 / 进程数	1	0.1	0.01	0.001	0.0001	0.00001
1	9.7	40.4	249.9	474.6	724.0	1132.4
2	4.8	21.7	125.2	237.0	305.1	648.2
4	3.0	10.4	62.8	119.0	181.9	329.7
8	1.3	5.3	32.1	60.1	93.9	157.1
16	0.7	2.8	16.2	30.6	46.6	72.0

针对该图，我们分别绘制了下面 3 幅图，图 5-5 展示了实际计算时间与计算精度的关系。图 5-6 展示了实际计算时间与集群规模的关系。图 5-7 是将图 5-6 中的实际计算时间转换成对数形式。

从图 5-5 可以很明显的看出，MPI 计算时间与计算精度正相关，且变化趋势与 Hadoop 类似。具体原因前面已经叙述。

从图 5-6 和图 5-7 可以得到，MPI 计算时间与集群规模呈现比较严格的线性负相关。说明 MPI 有着优良的可扩展性，可以通过扩展集群规模处理更大规模的计算量。

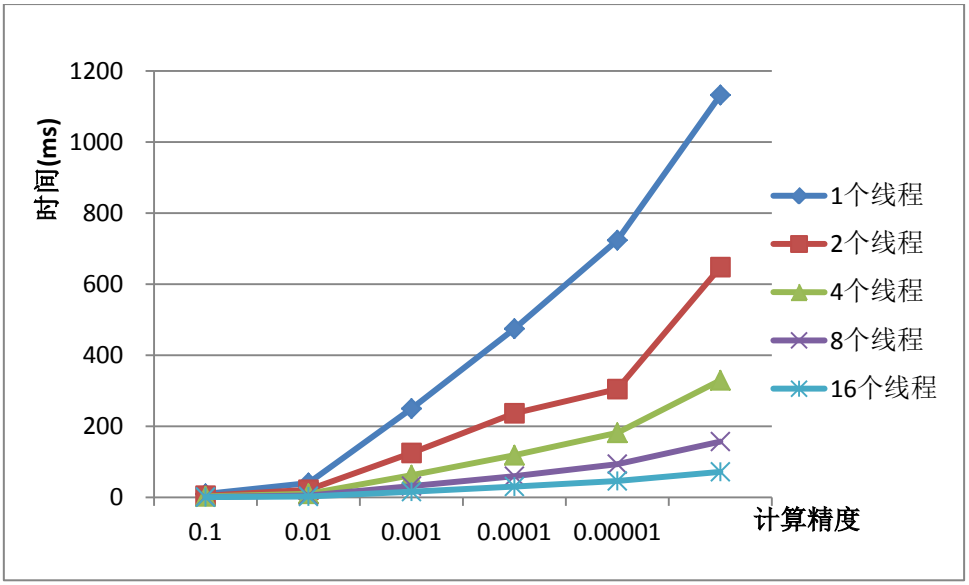

图 5-5 MPI 计算时间与计算精度关系图

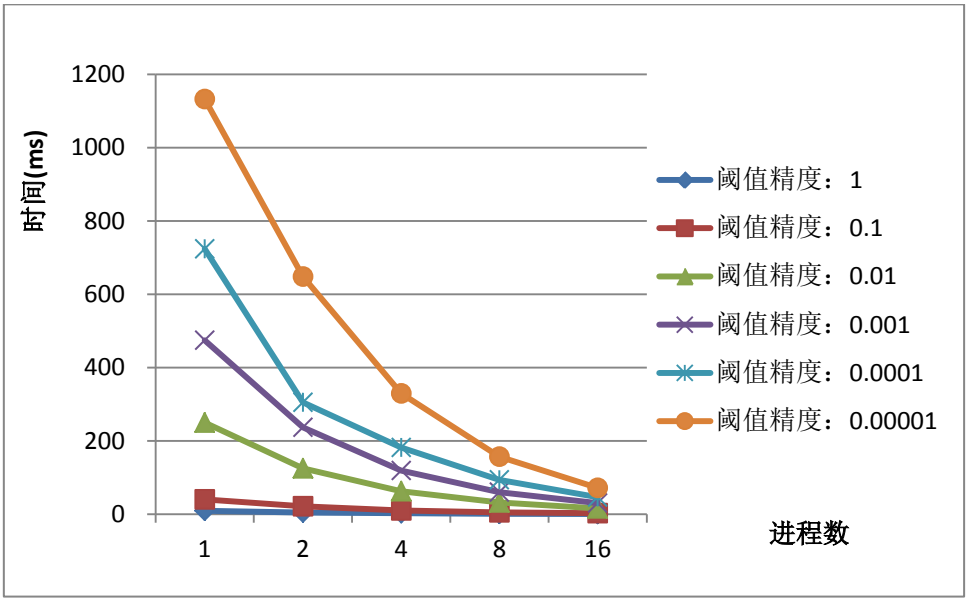

图 5-6 MPI 计算时间与集群规模关系图

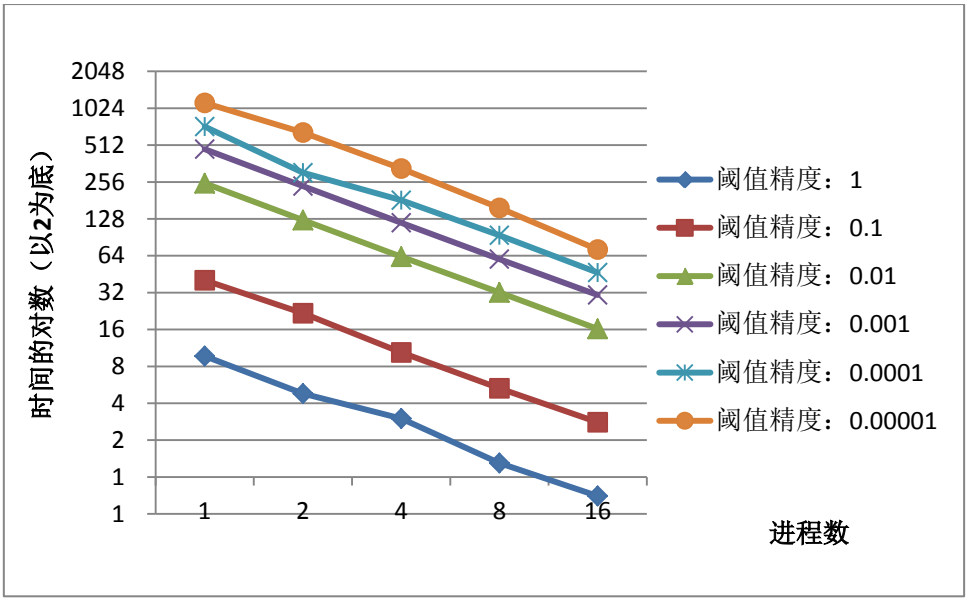

图 5-7 MPI 计算时间的对数与集群规模关系图

接着我们考虑总时间（IO 时间+计算时间）与集群规模的关系，具体实验数据见表 5-4：

表 5-4 MPI 总时间与集群规模关系图

精度 / 进程数	1	0.1	0.01	0.001	0.0001	0.00001
1	525.7	553.7	806.5	997.1	1244.7	1654.0
2	526.4	541.3	633.2	777.0	828.0	1159.2
4	523.4	532.7	602.1	642.0	713.8	859.8
8	519.1	513.7	554.7	589.4	600.9	677.9
16	514.8	520.3	534.3	576.8	573.8	595.6

我们用总时间的对数作为纵轴，以集群规模作为横轴进行作图，得到图 5-8，我们看到随着集群规模的增加，总时间逐渐减少，但不是呈现线性下降，当精度越高，也就是计算时间占总时间的比重越大，线性就越明显。原因可以从之前的讨论中获得，IO 时间与集群规模没有相关性，纯粹计算时间与集群规模线性负相关，所以总时间肯定是负相关，但不是线性的。从这里我们可以看出，MPI 对于高精度运算可以通过扩充集群规模提升性能，而且对于计算精度要求越高的应用，可扩展性越好。

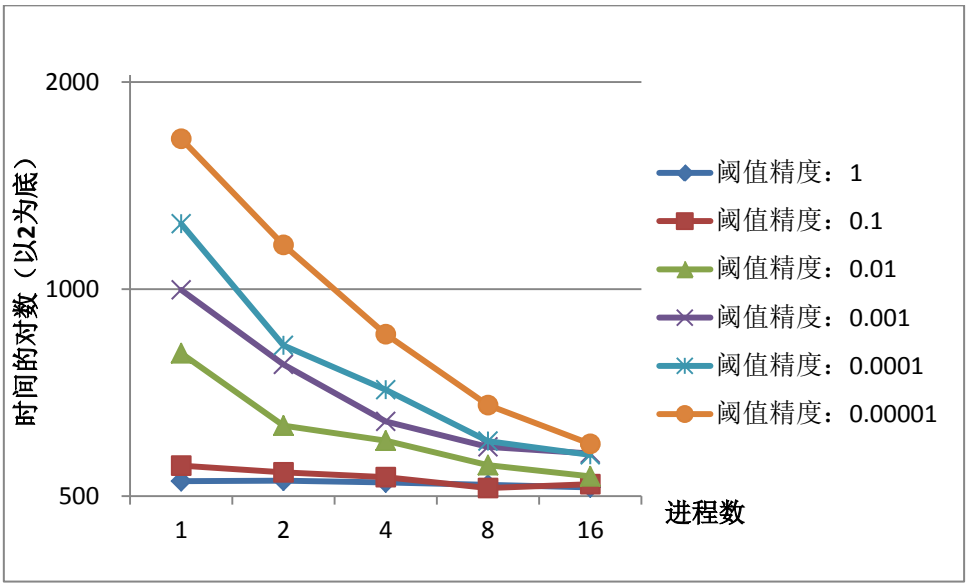

图 5-8 MPI 总时间的对数与集群规模关系图

在分别考虑了与计算精度，集群规模之后，我们继续统计 MPI 程序的 IO 时间，

计算时间与输入数据规模的关系，具体数据见表 5-5：

表 5-5 MPI 运行时间与数据量的关系

数据量 / 进程数	100000		200000		400000		800000		1600000	
	I/O	计算	I/O	计算	I/O	计算	I/O	计算	I/O	计算
1	128.8	10.3	256.8	20.3	513.1	40.7	1483.2	80.4	2051.3	161.8
2	128.9	5.2	257.4	10.2	548.3	20.3	1082.3	40.4	2214	81.3
4	127.7	2.6	264.7	5.2	508.8	10.3	1008.4	20.4	2066.5	41
8	128.5	1.4	257.1	2.7	519.8	5.3	1016.5	10.4	2091.2	20.8
16	129.2	0.8	255.2	1.5	745.7	2.8	1090.9	5.4	2837	10.6

我们分别用 IO 时间以及计算时间的对数与输入数据规模进行作图（图 5-9，图 5-10），可以发现无论是 IO 时间还是计算时间都与数据规模呈现线性正相关，这就导致了总时间与 I/O 时间是线性正相关的，这一特点和 Hadoop 的表现类似。

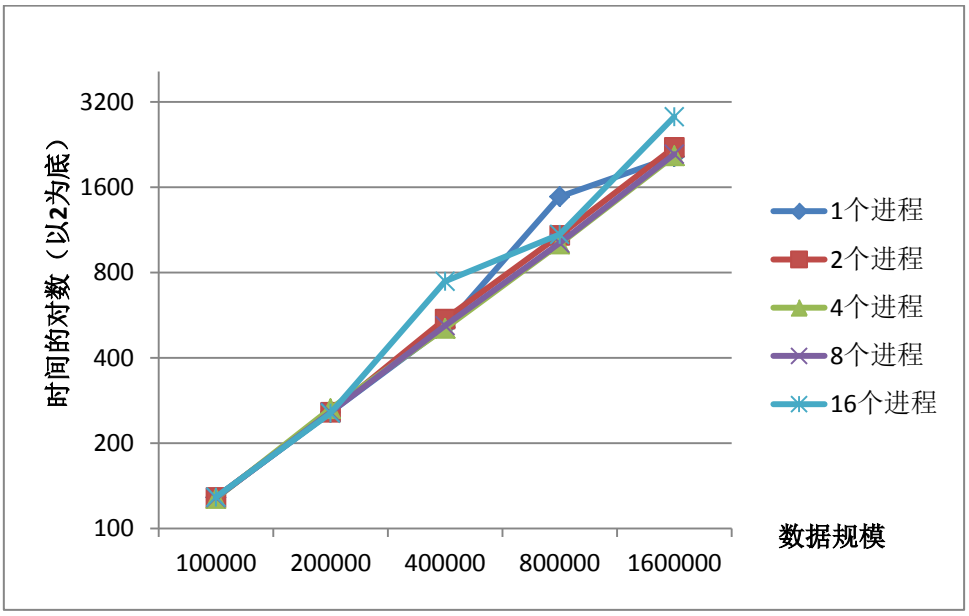

图 5-9 MPI I/O 时间的对数与数据规模关系图

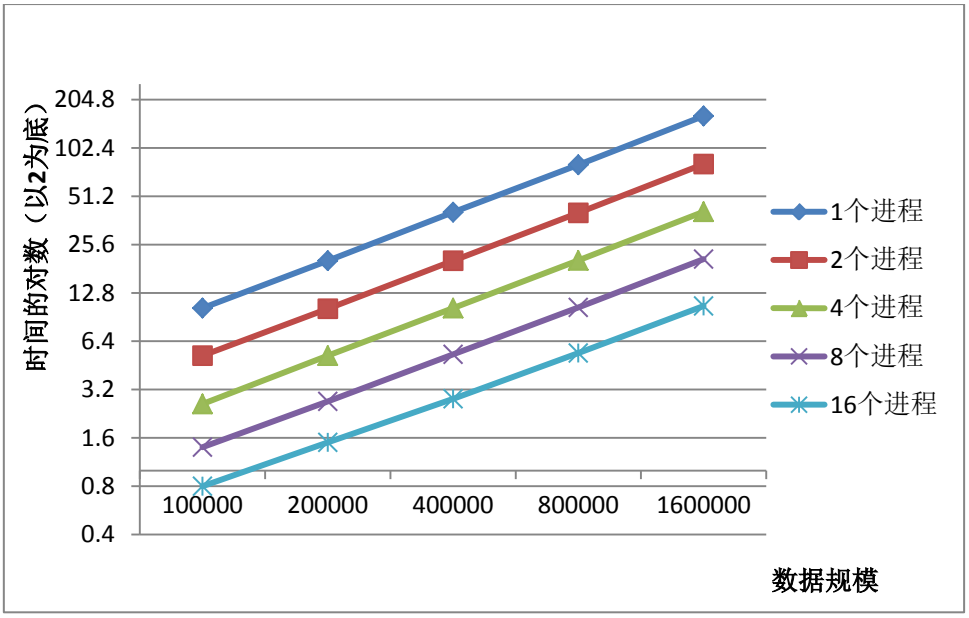

图 5-10 MPI 计算时间的对数与数据规模关系图

5.3 Hadoop 与 MPI 在 K-means 算法中的性能比较

下面我们对 Hadoop 与 MPI 进行对比，具体数据参见表 5-6，该表数据均是在相同集群规模，相同计算精度(0.01)下得到。从 5-11 的柱状图可以发现对于给定范围内的输入数据量，MPI 所使用的时间要比 Hadoop 少的多，两者相差几十倍。这在一定情况下说明对于 K-means 这类需要较多迭代次数的算法，MPI 更占优势，这是因为 Hadoop 每次运算之后的结果都需要存储在 HDFS 里面，下次迭代再使用的话，还需要读出来进行一次计算，磁盘 IO 开销比较大，而且由于 HDFS 的高容错特性，还会耗费一些时间进行数据备份。而 MPI 使用消息传递机制，内存中少量的中间量经过高速网络直接传递，无需重复读写磁盘，也无需进行额外的数据备份。

表 5-6 Hadoop 与 MPI 计算时间与数据规模关系表

数据规模	100000	200000	400000	800000	1600000
Hadoop	8117.4	10951.9	15527.8	28090.3	49565.6
MPI	130.0	256.7	728.5	1096.3	2847.6
Hadoop/MPI	62.4	42.7	21.3	25.6	17.4

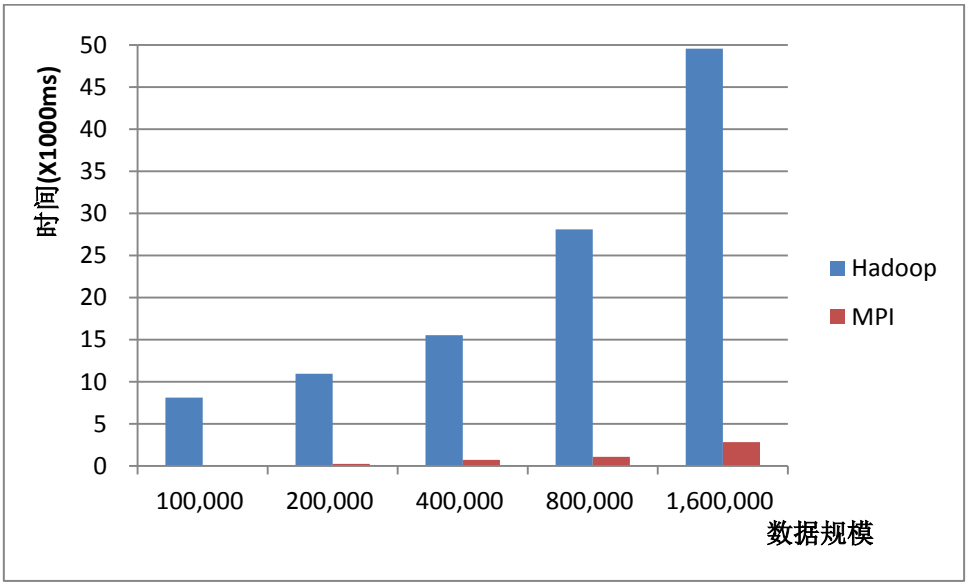

图 5-11 Hadoop 与 MPI 计算时间与数据规模关系图

我们以数据规模作为横坐标，两者的比率作为纵坐标进行作图（图 5-12），结果近似为负相关，也就是说随着数据量的增加，MPI 相对于 Hadoop 耗时增幅更大。这是因为随着数据量增加，使用 MPI 的各个节点之间需要传送的数据量就更大，耗费更多的网络 I/O 时间。而在 Hadoop 中，由于数据存储与计算更多在同一个节点中，耗费的网络 I/O 时间较少。但是考虑到在我们的实验中 MPI 底层没有使用分布式文件系统，在数据传输效率方面与 Hadoop 相比低了数倍，设想如果采用类似于 HDFS 的分布式文件系统，MPI 传输效率可能会更高。所以即使 Hadoop 对数据规模增长比较友好，但是要在 TB 以下数量级超越 MPI，可能性依然较小。

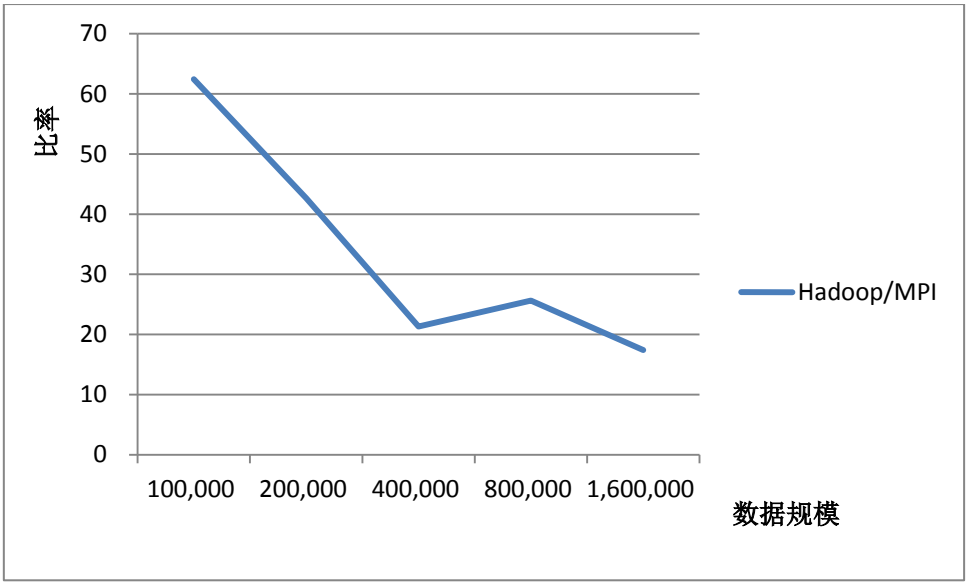

图 5-12 Hadoop 与 MPI 计算时间比率图

最后我们考虑 Hadoop 与 MPI 计算时间与计算精度之间的关系。具体数据如表 5-7（相同规模集群，400000 规模数据量）。我们以计算精度作为横坐标，两者的比率作为纵坐标进行做图，结果呈现比较严格的正相关。就是说 MPI 对于计算量、迭代量增长更加友好。这是因为 Hadoop 在每次迭代之后都需要进行文件的读写，随着计算精度的提高，这些文件读写时间会大量增加；而 MPI 只需要在程序运行初始阶段进行一次文件读写，之后只需要从内存中读数据即可，效率要高的多。

表 5-7 Hadoop 与 MPI 计算时间与计算精度关系表规模

计算精度	1	0.1	0.01	0.001
Hadoop	1909.3	15527.8	42860.0	115722.0
MPI	514.8	520.3	534.3	576.8
Hadoop/MPI	3.708819	29.843936	80.217106	200.627601

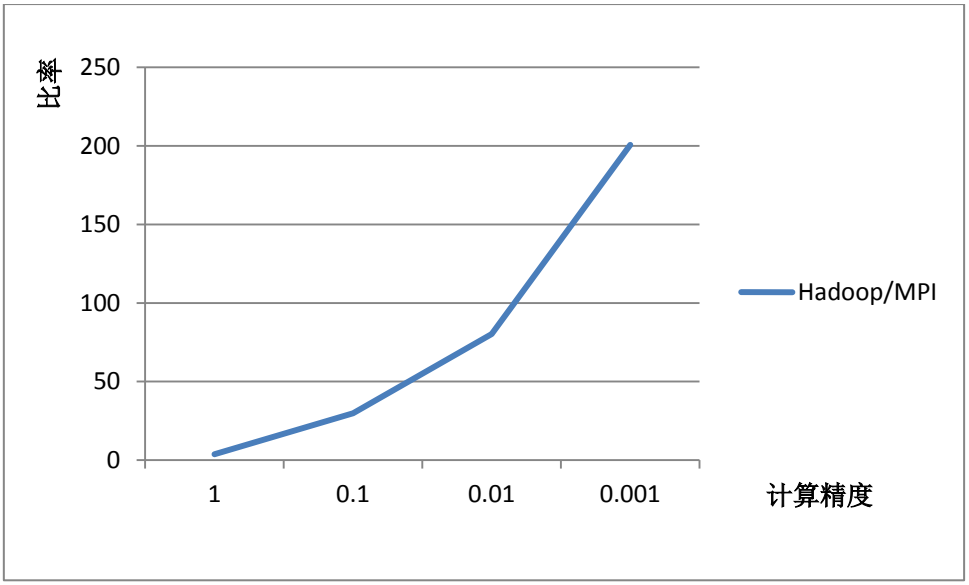

图 5-13 Hadoop 与 MPI 计算时间比率图

6. 结论

我们通过一系列的实验数据比较了 Hadoop 和 MPI 并行程序在 K-means 算法中的性能特点。我们可以得到下面的结论：

从整体来看，对于类似 K-means 这类需要数次迭代的算法，MPI 相对于 Hadoop 有着较大优势，两者有着数十倍的性能差异。

对于精度提高，MPI 和 Hadoop 所需时间的增加都近似指数增加，但是相对来说 MPI 的表现更加优异。也就说随着计算精度的提高，MPI 相对 Hadoop 的优势会越来越大。

对于数据规模的提高，MPI 和 Hadoop 所需时间的增加都近似线性增加，Hadoop 的表现更加优良，也就说随着数据规模的提高，MPI 相对 Hadoop 的优势会越来越小，但是在 PB 级规模下很难被 Hadoop 超越。

所以我们可以得到结论：MPI 更加适合高迭代类算法，而且在此类应用中，MPI 拥有很好的可扩展性，可以通过增加集群规模，线性降低运行时间。

参考文献

[1] 张嘉桐. 基于云计算的地学 G~4I 系统结构设计. 吉林大学, 2013.

[2] 曲寿利. 地震勘探技术的发展促进油气勘探新发现. 石油地球物理勘探, 2005, 40(3): 366-370.

[3] 薛勇, 万伟, 艾建文. 高性能地学计算进展. 世界科技研究与发展, 2008, 30(3): 314-319.

[4] 李焱. 基于 MPI 的大地电磁三维正反演并行算法研究. 中国地质大学, 2011.

[5] 张治宏. 基于 MPI 的并行计算研究. 中国地质大学 (北京), 2006.

[6] 余欣, 杨明, 王敏, 等. 基于 MPI 的黄河下游二维水沙数学模型并行计算研究. 人民黄河, 2005, 27(3): 49-50.

[7] Yang C, Xu Y, Nebert D. Redefining the possibility of digital Earth and geosciences with spatial cloud computing. International Journal of Digital Earth, 2013, 6(4): 297-312.

[8] Hamster: Hadoop And Mpi on the same cluster. https://issues.apache.org/jira/browse/MAPREDUCE-2911

[9] 余飞. 电信运营商大数据应用典型案例分析. 信息通信技术, 2014, 6: 014.

[10] 王龙, 王晓青, 郑友华, 等. 水库地震烈度衰减的研究. 地震地质, 2009, 31(4): 758-767.

[11] 牛瑞卿, 韩舸. 利用数据挖掘的滑坡监测数据处理流程. 武汉大学学报: 信息科学版, 2012, 37(7): 869-872.

[12] 段炬霞. K-means 算法的改进及其在地质勘探地震数据分析中的应用. 北京邮电大学, 2012.

[13] Valiant L G. A bridging model for parallel computation. Communications of the ACM, 1990, 33(8): 103-111.

[14] 伍少坤 基于影元胞技术的城市元胞自动机——以珠江三角洲与东莞市为例. 中山大学，2007.

[15] 陈国良等 并行计算机体系结构 北京：高等教育出版社，2002.9.

[16] 祥港. 借助中尺度数值天气预报模式实现在自搭建的网格环境下移植 MPI 的研究. 天津科技大学, 2010.
[17] 郭庆平, 肖金生. 多处理机并行系统的现状与发展. 武汉交通科技大学学报, 1998, (1):13-17.

致谢

伴随着本科毕业论文的完成，我的大学生活也将告一段落了。在这里我真心向给予我帮助和指导的老师，同学和我的父母表示衷心的感谢。

首先真的很感谢指导老师周会群老师。在整个论文撰写过程中，周老师都给予了我很大的帮助。特别是在实验开始阶段，由于在集群配置过程中出现了这样或那样的问题，周老师在百忙之中一直耐心指导，并且对我很信任，也教会了我正确的科研态度。在这个过程中我学习到了很多额外的知识。

由衷的感谢南京大学地球科学与工程学院的老师们。在大学四年之中，地科院老师们严谨治学的态度，钻研求实的精神都深深地影响了我。相信在以后的工作生活中都将受益匪浅。在生活中，老师特别是辅导员也无时无刻不在替我们着想，真的很感谢你们。

另外我的同窗好友让我的课余生活变得丰富多彩。感谢 2011 级的全体同学，当然还有那些曾经帮助我的学长学姐们。有了你们的陪伴，大学四年的生活必将是我今后人生中最美好的记忆。

感谢这段时间，我敬爱的父母在生活中的关怀，是你们一直支持着我前进。

最后感谢答辩组的各位老师在百忙之中的指点。