****



**本 科 毕 业 论 文**

院 系

专 业

题 目

年 级 学 号

学生姓名

指导老师 职 称

论文提交日期

**南京大学本科生毕业论文（设计、作品）中文摘要**

题目：

院系 专业 级本科生姓名：

指导教师（姓名、职称）：

摘要：

关键词：并行计算， MPI， Hadoop， Mapreduce， 地质学应用，矩阵计算

**南京大学本科生毕业论文（设计、作品）英文摘要**

THESIS：

DEPARTMENT：

SPECIALIZATION:

UNDERGRADUATE:

MENTOR:

ABSTRACT：

KEY WORDS:

[1. 绪论 5](#_Toc419641867)

[1.1 研究背景和意义 5](#_Toc419641868)

[1.2 国内外关于该课题的研究现状及趋势 5](#_Toc419641869)

[1.3 本文的工作内容和安排。 6](#_Toc419641870)

[2. MPI并行编程模型 7](#_Toc419641871)

[2.1 并行计算的相关理论 7](#_Toc419641872)

[2.1.1并行计算的相关概念 7](#_Toc419641873)

[2.1.2并行计算机结构模型 7](#_Toc419641874)

[2.1.3并行编程模型 9](#_Toc419641875)

[2.2 MPI简介 10](#_Toc419641876)

[3. Hadoop分布式计算框架 11](#_Toc419641877)

[3. MPI和Hadoop分布式计算模型相关比较 12](#_Toc419641878)

[4. 总结 12](#_Toc419641879)

[5. 致谢 13](#_Toc419641880)

[参考文献： 13](#_Toc419641881)

# 绪论

## 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主流。在这个过程中大数据量、高计算量开始成为科研工作者新的挑战，传统的单机存储、计算模型已经难以满足人们的需要，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、矿产资源定位定量预测、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。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。另外据曲寿利等人有关地震资料勘探的研究中[2]，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、超稠油油藏的比例持续增大，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，分布式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（包括地理的时间、空间世界）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。正是因为有以这种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（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）的模拟才能真正实现；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并行计算模型和以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还用于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CPU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存储。Hadoop主要由两大核心组成，HDFS(Hadoop Distributed File System)和Mapreduce。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。

## 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研究[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，比如在Chaowei Yang等人的文章中提出希望利用Hadoop的Mapreduce进一步探讨数字地球可行性[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的文件系统对其进行很好的支撑。数据处理节点和数据存储节点往往是相异的，这种数据存储和处理分离的情况就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](https://issues.apache.org/jira/browse/MAPREDUCE-2911)]。

在Hadoop中Mapreduce有了HDFS的支持，利用HDFS的特性，Mapreduce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，避免了在网络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，同时Hadoop可以很好的应对节点失效或者网络的通信终端情况。在某些情况下如果出现了异常情况，HDFS的备份机制和容错机制可以保证应用的持续进行，而不用重新开始所有的计算，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。特别是在部分科学计算领域，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，其在运算性能上要低于MPI的C和Fortran绑定。而且使用Hadoop与MPI相比，在网络带宽提高的情况下，性能增幅较低。

为了比较两者哪个更适用于地学研究中的普通大数据量计算应用，我通过实验比较了两者在常见的大矩阵相乘算法和K-means算法中的性能。

大矩阵相乘是许多运算的基础，包括求解线性方程组，加密算法，压缩算法等，在地学的相关应用也比较常见，特别是在大地动力学，地球物理学等二级学科当中。比如陈秋杰等人的有关地球重力场分析[9]以及王佳佳等人的三峡库区滑坡灾害预测预报系统研究[10]都广泛使用了矩阵相乘。同样k-means算法是聚类分析中非常重要的一种方法，而聚类分析又在地学研究中普遍使用。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[11]。牛瑞卿等人“利用数据挖掘的滑坡监测数据处理流程”[12]中也多次使用了k-means算法。由此可见这两种算法都在地学研究中有着广泛的应用场景，所以我在本文中通过对这两种算法在MPI和Hadoop两种并行编程模型中的实际性能进行比较分析，以希冀能为未来的地学高计算量应用提供参考性建议。

## 本文的结构

本文同过对MPI并行计算模型和Hadoop分布式计算模型在大矩阵相乘、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：

第一章（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。最后给出了本文的结构和安排。

第二章（并行计算基础理论）主要介绍了并行计算的一些基础概念，常见的并行计算机结构模型和并行编程模型。

第三章（MPI基础理论）和第四章（Hadoop基础理论）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。

第五章分析了MPI和Hadoop两种并行编程模型在大矩阵相乘方面的性能差异。

第六章分析了MPI和Hadoop两种并行编程模型在k-means算法方面的性能差异。

第七章为结论部分，对之前的实验结果和性能分析进行概括性总结。

# 并行计算基础理论

## 并行计算的相关概念

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]。

## 并行计算机结构模型

一般大型并行机系统结构一般可分为5类：

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

5种多指令流多数据流并行计算机 (MIMD, Multiple-Instruction Multiple-Data)的结构模型如图2-1所示。



图2-1 5种并行机结构模型

* B(Bridge)：存储总线和I/O总线间的接口
* DIR(Cache Directory)：高速缓存目录
* IOB(I/O Bus)：I/O总线
* LD(Local Disk)：本地磁盘
* MB(Memory Bus)：存储器总线
* NIC(Network Interface Circuitry)：网络接口电路
* P/C(Microprocessor and Cache)：微处理器和高速缓存
* SM(Shared Memory)：共享存储器

### 并行向量处理机（PVP）

PVP的结构示如图2-1(a)所示。这样的系统中包含了若干个专门定制的高性能向量处理器（Vector Processor），一般来说每个VP都具有1GFLOPS以上的高速处理能力。在该系统中使具有高带宽、低迟延的交叉开关网络将各个向量处理器连向共享存储模块，共享存储器以每秒兆字节的速度向处理器提供所需处理的数据。一般来说并行向量处理机不使用高速缓存，而是使用数量较多的的向量寄存器和指令缓存器。

### 对称多处理机（SMP）

SMP的结构如图2-1(b)所示。对称多处理机系统中微处理器经由高速总线或者类似PVP中的交叉开关与共享存储器相连，给系统与PVP相比，最大的不同是该系统中的各个处理器是对称的，可以等价的方位共享处理器或者I/O设备。由于这种对称性，才赋予了SMP比PVP更高的并行度，但是由于使用了与PVP类似的共享存储，导致系统所负载的处理器一般低于64，于此同时总线和交互开关也是影响其并行度的关键因素。目前SMP重要用于数据仓库（data warehouse），在线事务处理（OLTP）等应用中。

### 大规模并行处理机（MPP）

MPP的结构如图2-1(c)所示。大规模并行处理机一般具有一下特性：

* 处理节点采用类似SMP的微型处理器；
* 系统中的存储器是分离的；
* 采用专门设计的高通信带宽；
* 能扩充至最多几千个并行处理器；
* 它是一种异步的MIMD机器，程序系由多个进程组成，每个都有其私有地址空间，进程间采用传递消息相互作用。

大规模并行处理机目前的主要应用领域包括高性能科学计算，复杂物理信号的处理以及高仿真的工程模拟。

### 分布共享存储多处理机(DSM)

DSM的结构如图2-1(d)。拥有高速缓存目录是DSM与MPP最大的不同，由于DIR的存在，DSM可以支持各个分布高速缓存的一致性。相对于最终的用户来说，SDM提供了一个单地址的编程空间，也就是说用户无需指定到底使用哪个存储器里的内容，简化了编程过程。

### 工作站机群（COW）

COW结构示于图1.15(e)。工作站机群一般都是成本较低并行处理机，COW的每个节点通过普通的低成本网络，例如以太网，ATM开关等进行互联，每个节点都是一个相对独立的工作节点，可以使一台PC也可以是SMP。与MPP有所不同，COW的每个节点都没有本地磁盘。现今，MPP和COW之间的界线越来越模糊。例如，IBM SP2 它虽视为MPP，但它却有一个机群结构。机群相对于MPP有性能/价格比高的优势，所以在发展可扩放并行计算机方面呼声很高。

## 并行编程模型

数据并行编程模型和消息传递编程模型是目前最重要的两种并行编程模型，也是本论文重点讨论的两种模型。数据并行编程模型（比如Hadoop）通过高度抽象，编程过程十分简单，有人把它比喻成常规编程中的Java语言。消息传递编程模型（比如MPI）抽象程度较低，但是可以操作的东西也更多，因此拥有更高的应用领域，有人把它比喻成常规编程中的C语言。当然还有一些其它并行编程模型，比如隐式模型和共享变量，隐式并行通过编译器和运行时环境的支持来挖掘程序的潜在并行性。共享变量一般适合在SMP系统和多核系统中，代码分别驻留在若干个并行处理器中，但是可以通过读写共享存储器中的变量进行通信[15]。

### 数据并行编程模型

　　数据并行编程模型利用将要处理的数据集合的无关性，将相同的操作并行作用于不同的数据，该模型为使用者提供了一个全局的地址空间，一般这种形式的语言本身就提供并行执行的语义，因此对于使用者来说，只需要简单地指明将要执行怎样的并行操作以及并行操作的作用对象，就可以实现数据并行编程。因此数据并行的表达是非常简单和简洁的，它不需要编程者关心并行机是如何对该操作进行并行执行的。

### 消息传递并行编程模型

　　消息传递即各个并行执行的部分之间通过传递消息来交换信息、协调步伐、控制执行。消息传递一般是面向分布式内存的，但是它也可适用于共享内存的并行机。

　　消息传递为编程者提供了更灵活的控制手段和表达并行的方法，一些用数据并行方法很难表达的并行算法，都可以用消息传递模型来实现，灵活性和控制手段的多样化，是消息传递并行程序能提供高的执行效率的重要原因。

　　消息传递模型一方面为编程者提供了灵活性，另一方面，它也将各个并行执行部分之间复杂的信息交换和协调、控制的任务交给了编程者，这在一定程度上增加了编程者的负担，这也是消息传递编程模型编程级别低的主要原因。虽然如此，消息传递的基本通信模式是简单和清楚的，学习和掌握这些部分并不困难，因此目前大量的并行程序设计仍然是消息传递并行编程模式。

消息传递与数据并行的对比

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 数据并行与消息传递并行编程模型   |  |  |  | | --- | --- | --- | | 对比内容 | 数据并行 | 消息传递 | | 编程级别 | 高 | 低 | | 适用的并行机类型 | SIMD/SPMD | SIMD/MIMD/SPMD/MPMD | | 执行效率 | 依赖于编译器 | 高 | | 地址空间 | 单一 | 多个 | | 存储类型 | 共享内存 | 分布式或共享内存 | | 通信的实现 | 编译器负责 | 程序员负责 | | 问题类 | 数据并行类问题 | 数据并行、任务并行 | | 目前状况 | 缺乏高效的编译器支持 | 使用广泛 | |

讲解:

　　不同的并行编程模型也就是不同的通过并行程序解决问题的模式，数据并行的主要特征是以数据为中心，通过对数据的划分和并行处理来解决问题，消息传递当然也可以实现上述功能，但是消息传递在问题的表述上更具体，更低级，可以解决的问题相对于数据并行模型来说也更广泛。在一定程度上，可以把数据并行看作是消息传递的特殊形式。

# 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标准且可扩展的消息传递系统。它可以运行在一组异构的并行计算机上。MPI标准制定的一套语法和语义核心库可被用户使用编写可移至的消息传递程序。目前MPI已经有了C,C++,Fortran,Java和Go绑定，有几个经过严格测试并且保证高效的MPI具体实现，其中一些是开源的。

MPI已经成为目前应用最广泛和最稳定的并行程序设计规范,几乎所有的主流操作系统（Windows，Linux，Unix，MacOS等）都支持最新的MPI2.0标准。MPI的并行编程主要靠一组消息传递库函数惊醒。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发送，接受和同步接口。

MPI有两种基本的并行程序设计模式：总从模式和对等模式。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：

该种模式中一个主进程对整个并行程序进行总体控制，从进程在主进程的控制下对数据进行处理和运算。这是当前的一些程序使用比较多的一种并行程序设计模式。也是本文对应实验所采用的方式。

对等模式：

程序的各个进程的地位完全等同，只不过是处理的数据和对象有所不同。

MPI的通讯模式

分为四种：标准通信模式、缓存通信模式、同步通信模式、就绪通信模式

标准通信模式：

是否缓存数据由MPI决定

如果缓存数据，发送操作的进行与接收操作是否执行无关。数据缓存完毕时发送操作就会正确返回。

如果不缓存数据，只有当相应的接收调用执行，并且发送数据完全到达接收缓冲区后，发送操作才算完成。这时，发送缓冲区可以被覆盖。对于非阻塞通信，发送操作虽然没有完成，但发送调用已经正确返回。

缓存通信模式：程序员直接对通信缓冲区进行控制。

    用户必须保证发送前必须有缓冲区可用，否则失败返回。阻塞发送后，缓冲区可覆盖，但非阻塞发送正确返回后，缓冲区不能立刻被覆盖。这种通信模式下发送调用是否正确返回只和有无缓冲区有关。另外，请注意区别MPI的发送/接收缓冲区与系统缓冲区。

同步通信模式

    这种通信的开始不依赖于相应的接收操作是否起动，但是同步发送必须等相应的接收进程开始后才可正确返回。这个返回意味着发送缓冲区中的数据被系统缓冲区缓存并且开始发送，就是说，返回时发送缓冲区可以被覆盖了。

就绪通信模式

    只有当接收操作起动时，发送操作才能开始，否则发送操作会出错。对于非阻塞发送操作的正确返回，不等于发送完成，但对于阻塞发送的正确返回，发送缓冲区可被覆盖。

    这是一个特别的通信模式，所以，一般在使用时，会先起动接收操作，然后再向发送进程发消息，发送进程收到这个消息后，才会开始进行发送操作。

# Hadoop简介

Hadoop是一个由Apache基金会所开发的分布式系统基础架构。

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存储。

[1] Hadoop实现了一个分布式文件系统（Hadoop Distributed File System），简称HDFS。HDFS有高容错性的特点，并且设计用来部署在低廉的（low-cost）硬件上；而且它提供高吞吐量（high throughput）来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。HDFS放宽了（relax）POSIX的要求，可以以流的形式访问（streaming access）文件系统中的数据。

Hadoop的框架最核心的设计就是：HDFS和MapReduce。HDFS为海量的数据提供了存储，则MapReduce为海量的数据提供了计算。[2]

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。它主要有以下几个优点：

高可靠性。Hadoop按位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，项目的软件成本因此会大大降低。

Hadoop带有用Java语言编写的框架，因此运行在 Linux 生产平台上是非常理想的。Hadoop 上的应用程序也可以使用其他语言编写，比如 C++。

# MPI和Hadoop分布式计算模型相关比较

实验情况搭建

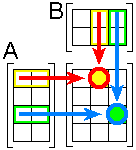
# 矩阵与矩阵相乘

矩阵相乘最重要的方法是一般矩阵乘积。它只有在第一个矩阵的列数（column）和第二个矩阵的行数（row）相同时才有定义。一般单指矩阵乘积时，指的便是一般矩阵乘积。若*A*为m\times n矩阵，*B*为n\times p矩阵，则他们的**乘积***AB*(有时记做*A* · *B*）会是一个m\times p矩阵。其乘积矩阵的元素如下面式子得出：

 (AB)_{ij} = \sum_{r=1}^n a_{ir}b_{rj} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}. 

以上是用矩阵单元的代数系统来说明这类乘法的抽象性质。本节以下各种运算法都是这个公式的不同角度理解，运算结果相等：

**由定义直接计算**

[](http://zh.wikipedia.org/wiki/File:Matrix_multiplication_diagram.PNG)

左边的图表示出要如何计算*AB*的（1,2）和（3,3）元素，当*A*是个4×2矩阵和*B*是个2×3矩阵时。分别来自两个矩阵的元素都依箭头方向而两两配对，把每一对中的两个元素相乘，再把这些乘积加总起来，最后得到的值即为箭头相交位置的值。

(AB)_{1,2} = \sum_{r=1}^2 a_{1,r}b_{r,2} = a_{1,1}b_{1,2}+a_{1,2}b_{2,2}


(AB)_{3,3} = \sum_{r=1}^2 a_{3,r}b_{r,3} = a_{3,1}b_{1,3}+a_{3,2}b_{2,3}


# K-means聚类算法性能比较分析

## K-means聚类算法概述

k-means算法是数据挖掘过程中的一种重要聚类算法。k-means聚类要求将n个观察点分配到k个集群中，并且要求改集群满足一下条件：

所在同一个集群中的点相似度要达到最大，在不同集群中的点的相似度要最小。

该算法的形式化描述如下：

对于给定的一组观察点集合，每个观察点都是一个d维的实数向量，k-means聚类算法就是要把这n个观察点分配到k个簇之中，使得下式的值最小：

其中是的中心点。

## K-means 算法基本步骤

1. 从 n个数据对象任意选择 k 个观察点作为初始聚类中心；
2. 根据每个聚类对象聚类中心，计算每个对象与这些聚类中心的欧式距离；并根据最小距离对相应观察点进行划分；
3. 重新计算每个（有变化）聚类的聚类中心；
4. 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②。

## 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聚类中心点的选择。在实际使用过程中，通常用不同的初始聚类中心进行多次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是所有观察点的数目。k是我们需要聚集的簇数，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。

## 基于Hadoop的并行K-means算法设计

使用Hadoop进行并行算法的设计，用户一般只需要进行Map和Reduce函数的设计。通常包括这个函数的输入输出键值类型已经两者的具体

# 结论

在Hadoop 出现之前，高性能计算和网格计算一直是处理大数据问题主要的使用方法和工具，它们主要采用消息传递接口（Message Passing Interface，MPI）提供的API 来处理大数据。高性能计算的思想是将计算作业分散到集群机器上，集群计算节点访问存储区域网络SAN 构成的共享文件系统获取数据，这种设计比较适合计算密集型作业。当需要访问像PB 级别的数据的时候，由于存储设备网络带宽的限制，很多集群计算节点只能空闲等待数据。而Hadoop却不存在这种问题，由于Hadoop 使用专门为分布式计算设计的文件系统HDFS，计算的时候只需要将计算代码推送到存储节点上，即可在存储节点上完成数据本地化计算，Hadoop 中的集群存储节点也是计算节点。在分布式编程方面，MPI 是属于比较底层的开发库，它赋予了程序员极大的控制能力，但是却要程序员自己控制程序的执行流程，容错功能，甚至底层的套接字通信、数据分析算法等底层细节都需要自己编程实现。这种要求无疑对开发分布式程序的程序员提出了较高的要求。相反，Hadoop 的MapReduce 却是一个高度抽象的并行编程模型，它将分布式并行编程抽象为两个原语操作，即map 操作和reduce 操作，开发人员只需要简单地实现相应的接口即可，完全不用考虑底层数据流、容错、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。

# 参考文献：

[1]张嘉桐. 基于云计算的地学 G~ 4I 系统结构设计[D]. 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[J]. 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[J]. 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[D]. 中国地质大学, 2011.

[5] 张治宏. 基于 MPI 的并行计算研究 [D][D]. 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[J]. 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[J]. International Journal of Digital Earth, 2013, 6(4): 297-312.

[8] https://issues.apache.org/jira/browse/MAPREDUCE-2911

[9] 陈秋杰, 沈云中, 张兴福. MKL 和 OpenMP 多核并行算法解算高阶地球重 я 场的效率 я 析[J]. 大地测量与地球动力学, 2012, 32(5): 10Л.

[10] 王佳佳, 殷坤龙. 基于 WEBGIS 和四库一体技术的三峡库区滑坡灾害预测预报系统研究[J]. 岩石力学与工程学报, 2014, 33(5): 1004-1013.

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

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

[13] Valiant L G. A bridging model for parallel computation[J]. 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 2007

[15] 张武生，薛巍，李建江，郑伟民 MPI并行程序设计实例教程 清华大学出版社 2009.2

# 5. 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。在这里我真心想给与我帮助和指导的老师，同学和我的父母表示衷心的感谢。

首先真的很感谢我指导老师周会群周老师。在真个论文撰写过程中，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，周老师都在百忙之中给与了我指导。并对我给与了很大的信任，也教会了我很多科研态度和精神。在这个过程中我学习到了很多额外的知识。

另外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，地科院老师们的严谨治学的态度，钻研求实的科研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。另外在生活中，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们的课余大学生活变得丰富多彩。感谢2011级的全体同学，当然还有那些曾经帮助我们的学长学姐们。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。