获取教材和讲义 PPT 等各种课程资料请访问 http://dblab.xmu.edu.cn/node/422

=课程教材由林子雨老师根据网络资料编著=

厦门大学计算机科学系教师 林子雨 编著

http://www.cs.xmu.edu.cn/linziyu

2013 年 9 月

前言

本教程由厦门大学计算机科学系教师林子雨编著，可以作为计算机专业研究生课程《大数据技术基础》的辅助教材。

本教程的主要内容包括：大数据概述、大数据处理模型、大数据关键技术、大数据时代面临的新挑战、NoSQL 数据库、云数据库、Google Spanner、Hadoop、HDFS、HBase、MapReduce、Zookeeper、流计算、图计算和 Google Dremel 等。

本教程是林子雨通过大量阅读、收集、整理各种资料后精心制作的学习材料，与广大数据库爱好者共享。教程中的内容大部分来自网络资料和书籍，一部分是自己撰写。对于自写内容，林子雨老师拥有著作权。

本教程 PDF 文档及其全套教学 PPT 可以通过网络免费下载和使用（下载地址：http://dblab.xmu.edu.cn/node/422）。教程中可能存在一些问题，欢迎读者提出宝贵意见和建议！

本教程已经应用于厦门大学计算机科学系研究生课程《大数据技术基础》，欢迎访问 2013 班级网站 http://dblab.xmu.edu.cn/node/423。

林子雨的E-mail是：ziyulin@xmu.edu.cn。

林子雨的个人主页是：http://www.cs.xmu.edu.cn/linziyu。

林子雨于厦门大学海韵园

2013 年 9 月

第 5 章 HDFS

厦门大学计算机科学系教师 林子雨 编著

个人主页：http://www.cs.xmu.edu.cn/linziyu

课程网址：http://dblab.xmu.edu.cn/node/422

2013 年 9 月

第 5 章 HDFS

Hadoop 分布式文件系统 HDFS（Hadoop Distributed File System）是一个能够兼容普通硬件环境的分布式文件系统，和现有的分布式文件系统不同的地方是，Hadoop 更注重容错性和兼容廉价的硬件设备，这样做是为了用很小的预算甚至直接利用现有机器就实现大流量和大数据量的读取。Hadoop 使用了 POSIX 的设计来实现对文件系统文件流的读取。HDFS 原来是 Apache Nutch 搜索引擎（从 Lucene 发展而来）开发的一个部分，后来独立出来作为一个 Apache 子项目。

本章介绍 HDFS 的相关知识，内容要点如下：

- HDFS 的假设与目标
- HDFS 的相关概念
- HDFS 体系结构
- HDFS 命名空间
- HDFS 存储原理
- 通讯协议
- 数据错误与异常
- 从 HDFS 看分布式文件系统的设计需求

5.1HDFS 的假设与目标

HDFS 是基于流数据模式访问和处理超大文件的需求而开发的，它可以运行于廉价的商用服务器上。HDFS 在设计时的假设和目标包括以下几个方面：

- **硬件出错**：Hadoop 假设硬件出错是一种正常的情况，而不是异常，为的就是在硬件出错的情况下尽量保证数据完整性，HDFS 设计的目标是在成百上千台服务器中存储数据，并且可以快速检测出硬件错误和快速进行数据的自动恢复。
- **流数据读写**：不同于普通的文件系统，Hadoop 是为了程序批量处理数据而设计的，而不是与用户的交互或者随机读写，所以 POSIX 对程序增加了许多硬性限制，程

序必须使用流读取来提高数据吞吐率。

- **大数据集**：HDFS 上面一个典型的文件一般是用 GB 或者 TB 计算的，而且一个数百台机器组成的集群里面可以支持过千万这样的文件。
- **简单的文件模型**：HDFS 上面的文件模型十分简单，就是一次写入多次读取的模型，文件一旦创建，写入并关闭了，之后就再也不会被改变了，只能被读取，这种模型刚好符合搜索引擎的需求，以后可能会实现追加写入数据这样的功能。
- **强大的跨平台兼容性**：由于是基于 Java 的实现，无论是硬件平台或者是软件平台要求都不高，只要是 JDK 支持的平台都可以兼容。

正是由于以上的种种考虑，我们会发现，现在的 HDFS 在处理一些特定问题时，不但没有优势，而且有一定的局限性，主要表现在以下几个方面：

- **不适合低延迟数据访问**：如果要处理一些用户要求时间比较短的低延迟应用请求，则 HDFS 不适合。HDFS 是为了处理大型数据集分析任务，主要是为了达到较高的数据吞吐量而设计的，这就可能以高延迟作为代价。目前的一些补充的方案，比如使用 HBase，通过上层数据管理项目来尽可能地弥补这个不足。
- **无法高效存储大量小文件**：在 Hadoop 中需要使用 NameNode（目录节点）来管理文件系统的元数据，以响应客户端请求返回文件位置等，因此，文件数量大小的限制要由 NameNode 来决定。例如，每个文件、索引目录及块大约占 100 字节，如果有 100 万个文件，每个文件占一个块，那么，至少要消耗 200MB 内存，这似乎还可以接受。但是，如果有更多文件，那么，NameNode 的工作压力更大，检索处理元数据的时间就不可接受了。
- **不支持多用户写入及任意修改文件**：在 HDFS 的一个文件中只有一个写入者，而且写操作只能在文件末尾完成，即只能执行追加操作。目前 HDFS 还不支持多个用户对同一文件的写操作，以及在文件任意位置进行修改。

5.2HDFS 的相关概念

5.2.1 块（Block）

我们知道，在操作系统中都有一个文件块的概念，文件以块的形式存储在磁盘中，此处块的大小代表系统读/写可操作的最小文件大小。也就是说，文件系统每次只能操作磁盘块

大小的整数倍数据。通常来说，一个文件系统块为几千字节，而磁盘块大小为 512 字节。文件的操作都由系统完成，这些对用户来说都是透明的。

这里，我们所要介绍的 HDFS 的块是抽象的概念，它比上面操作系统中所说的块要大得多。在配置 Hadoop 系统时会看到，它的默认块为 64MB。和单机上的文件系统相同，HDFS 分布式文件系统中的文件也被分成块进行存储，它是文件存储处理的逻辑单元。

HDFS 作为一个分布式文件系统，是设计用来处理大文件的，使用抽象的块可以带来很多好处。一个好处就是，可以存储任意大的文件，而又不会受到网络中任一单个节点磁盘大小的限制。可以想象一下，单个节点存储 100TB 的数据是不可能的，但是，由于逻辑块的设计，HDFS 可以将这个超大的文件分成众多块，分别存储在集群的各台机器上。另外一个好处是使用抽象块作为操作的单元，可以简化存储子系统。这里之所以提到简化，是因为这是所有系统的追求，而对故障出现频繁和种类繁多的分布式系统来说，简化就显得尤为重要。在 HDFS 中块的大小是固定的，这样就简化了存储系统的管理，特别是元数据信息可以和文件块内容分开存储。不仅如此，块更有利于分布式文件系统中复制容错的实现。在 HDFS 中为了处理节点故障，默认将文件块副本数设定为 3 份，分别存储在集群的不同节点上。当一个块损坏时，系统会通过 NameNode 获取元数据信息，在另外的机器上读取一个副本并进行存储，这个过程对用户来说都是透明的。当然，这里的文件块副本冗余量，可以通过文件进行配置，比如在有些应用中，可能会为操作频率较高的文件块设置较高的副本数量以及提高集群的吞吐量。

5.2.2 NameNode 和 DataNode

HDFS 体系结构中有两类节点，一类是 NameNode，另一类是 DataNode。这两类节点分别承担 Master 和 Worker 的任务。

- **目录节点（NameNode）**是集群里面的主节点，负责文件名的维护管理，也是客户端访问文件的入口。文件名的维护包括文件和目录的创建、删除、重命名等。同时也管理数据块和数据节点的映射关系，客户端需要访问目录节点才能知道一个文件的所有数据块都保存在哪些数据节点上。
- **数据节点（DataNode）**一般就是集群里面的一台机器，负责数据的存储和读取。在写入时，由目录节点分配数据块的保存位置，然后客户端直接写到对应的数据节点。在读取时，当客户端从目录节点获得数据块的映射关系后，就会直接到对应的

数据节点读取数据。数据节点也要根据目录节点的命令创建、删除数据块和冗余复制。

5.3HDFS 体系结构

Hadoop 文件系统采用主从架构对文件系统进行管理，一个 HDFS 集群由唯一一个目录节点（NameNode）和数个数据节点（DataNodes）组成（如图 5-1 所示）。目录节点是一个中心服务器，负责管理文件系统的名字空间及客户端对文件的访问。集群中的数据节点一般是一个节点运行一个数据节点进程，管理负责它所在节点上的存储。

HDFS 对外表现为一个普通的文件系统，用户可以用文件名去存储和访问文件，而实际上文件是被分成不同的数据块，这些数据块就是存储在数据节点上面。数据节点负责处理文件系统客户端的读/写请求，在目录节点的统一调度下进行数据块的创建、删除和复制。

一个典型的 Hadoop 文件系统集群部署，是由一台性能较好的机器运行目录节点，而集群里面的其它机器每台上面运行一个数据节点。当然一个机器可以运行任意多个数据节点，甚至目录节点和数据节点一起运行，不过这种模式在正式的应用部署中很少使用。

唯一的目录节点的设计大大简化了整个体系结构，目录节点负责 Hadoop 文件系统里面所有元数据的仲裁和存储。这样的设计使数据不会脱离目录节点的控制。

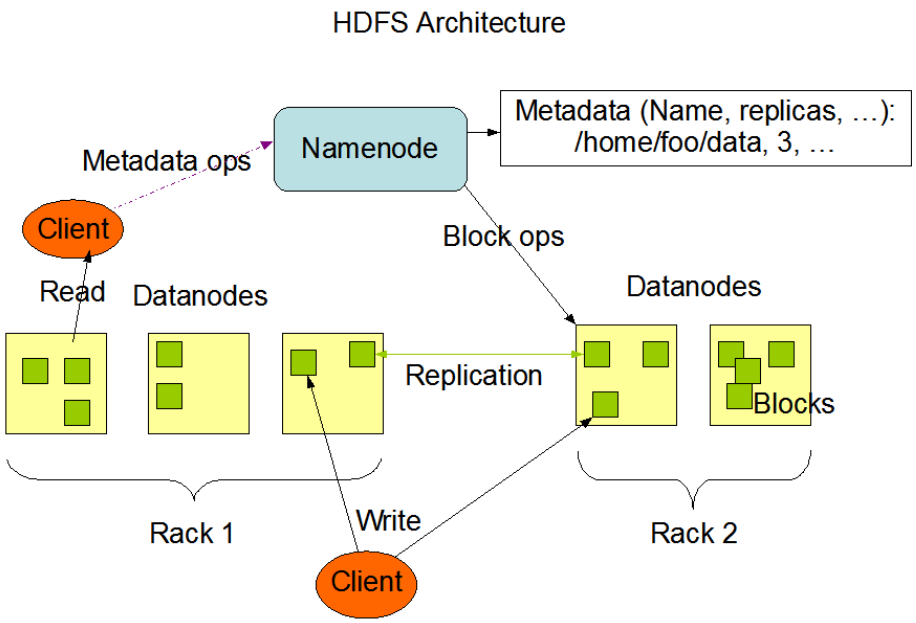

图 5-1 HDFS 的体系结构

5.4HDFS 命名空间

Hadoop 文件系统使用的是传统的分级文件体系，客户端程序可以创建目录并且在目录里面保存文件，类似于现在一般的文件系统。Hadoop 允许用户创建、删除文件，在目录间转移文件，重命名文件等，但是，还没有实现磁盘配额和文件访问权限等功能，也不支持文件的硬连接和软连接（快捷方式），这些功能在短期内不会实现。

目录节点负责存储和管理整个文件系统的命名空间，任何对文件系统命名空间或属性的修改都将被目录节点记录下来。应用程序可以指定某一个文件需要在 Hadoop 文件系统中冗余多少份，这个在 Hadoop 中称为冗余因子，保存在目录节点里面。

5.5HDFS 存储原理

5.5.1 冗余数据保存

Hadoop 文件系统是为了大文件的可靠保存而设计的，一个文件被划分成一连串的数据块，除了文件的最后一块以外其它所有的数据块都是固定大小的，为了数据容错性，每一个数据块都会被冗余存储起来，即存储多个副本，而每个文件的块大小和冗余参数都是可以设置的，程序可以设置文件的数据块要被复制多少份，而且这个冗余参数除了可以在创建的时候指定，还可以在之后改变。在 Hadoop 文件系统里面文件只会被写入一次，并且任何时间只会有一个程序在写入这个文件。

目录节点是根据数据块的冗余状况来作出处理决策的，数据节点会定期发送一个存在信号（Heartbeat）和数据块列表给目录节点，存在信号使目录节点认为该数据节点还是有效的，而数据块列表包括了该数据节点上面的所有数据块编号。

5.5.2 数据存取策略

复制策略是 Hadoop 文件系统最核心的部分，对读写性能影响很大，Hadoop 和其它分布式文件系统的最大区别就是可以调整冗余数据的位置，这个特性需要很多时间去优化和调整。

- **数据存放**

目前 Hadoop 采用以机柜为基础的数据存放策略，这样做的目的是提高数据可靠性和充

分利用网络带宽。当前具体实现了的策略只是这个方向的尝试，Hadoop 短期的研究目标之一就是在实际产品环境中观察系统读写的行为，测试性能和研究更深入的规则。

一个大的 Hadoop 集群经常横跨多个机柜，而不同机柜之间的数据通讯需要经过交换机或者路由，所以，同一个机柜中不同机器之间的通讯带宽，要比不同机柜之间机器的通讯带宽大。

Hadoop 提供了一个 API 来决定数据机所属的机柜 ID，当文件系统启动的时候，数据机就把自己所属的机柜 ID 发给目录机，然后目录机管理这些分组。

Hadoop 默认是每个数据机都是在不同的机柜上面，这种方法没有做任何性能优化，但是也有不少优点：

- 数据可靠性是最高的，因为这样可以防止机柜出错的时候数据丢失；
- 在读取数据的时候充分利用不同机柜之间的带宽；
- 而且这个策略可以很容易地完成负载平衡和错误处理。

缺点就是写入数据的时候并不能完全利用同一机柜里面机器的带宽。

在默认的配置下，Hadoop 的冗余复制因子是 3，意思就是每一块文件数据一共有 3 个地方存放，Hadoop 目前的存放策略是其中两份放在同一个 rack id 的不同机器上面，另外一个放在不同 rack id 的机器上面，简单来说就是 1/3 的冗余数据在一个机柜里面，2/3 的冗余数据在另外一个机柜里面，这样既可以防止机柜异常时候的数据恢复，又可以提高读写性能。

- **数据读取**

数据读取策略是：根据前面所说的数据存放策略，数据读取的时候，客户端也有 api 确定自己的机柜 id，读取的时候，如果有块数据和客户端的机柜 id 一样，就优先选择该数据节点，客户端直接和数据节点建立连接，读取数据。如果没有，就随机选取一个数据节点。

- **数据复制**

数据复制主要是在数据写入和数据恢复的时候发生，数据复制是使用流水线复制的策略。当客户端要在 Hadoop 上面写一个文件，首先它先把这个文件写在本地，然后对文件进行分块，默认 64MB 一块，每块数据都对 Hadoop 目录服务器发起写入请求，目录服务器选择一个数据机列表，返回给客户端，然后客户端就把数据写入第一台数据机，并且把列表传给数据机，当数据机接收到 4KB 数据的时候，写入本地，并且发起连接到下一台数据机，把这个 4KB 数据传过去，形成一条流水线。当最后文件写完的时候，数据复制也同时完成，这个就是流水线处理的优势。

5.6通讯协议

Hadoop 的通讯协议基本是在 TCP/IP 的基础上开发的，客户端使用 ClientProtocol 和目录服务器通讯，数据机使用 DatanodeProtocol 和目录服务器通讯，而目录服务器一般只是应答客户端和数据机的请求，不会主动发起通讯。

5.7数据错误和异常

Hadoop 文件系统的主要目标就是在硬件出错的时候保证数据的完整性，它把磁盘错误作为肯定会出现的情况来对待，而不是异常。一般数据存储中出现的错误有几种，分别是目录服务器错误、数据机错误和网络传输异常。

- **数据机出错**

每个数据机会定时发送一个心跳信息给目录服务器，表明自己仍然存活，网络异常可能会导致一部分数据机无法和目录服务器通讯，这时候目录服务器收不到心跳信息，就认为这个数据机已经死机，从有效 I/O 列表中清除，而该数据机上面的所有数据块也会标记为不可读。这个时候某些数据块的冗余份数有可能就低于它的冗余因子了，目录服务器会定期检查每一个数据块，看看它是否需要进行数据冗余复制。

- **出现数据异常**

由于网络传输和磁盘出错的原因，从数据机读取的数据有可能出现异常，客户端会采用 md5 和 sha1 对数据块进行校验。客户端在创建文件的时候，会对每一个文件块进行信息摘录，并把这些信息写入到同一个路径的隐藏文件里面。当客户端读取文件的时候，会先读取该信息文件，然后对每个读取的数据块进行校验，如果校验出错，客户端就会请求到另外一个数据机读取该文件块，并且报告给目录服务器这个文件块有错误，目录服务器就会定期检查，并且重新复制这个块。

- **目录服务器出错**

FsImage 和 Editlog 是目录服务器上面两个最核心的数据结构，如果其中一个文件出错的话，会造成目录服务器不起作用。由于这两个文件非常重要，所以，目录服务器上面可以设置多个备份文件和辅助服务器，当这两个文件有改变的时候，目录服务器就会发起同步操作，虽然这样增加了系统的负担，但是，对于目前这个架构而言，为了实现数据的可靠性，这个同步操作是非常必要的。

5.8从 HDFS 看分布式文件系统的设计需求

分布式文件系统的设计目标大概包括：透明性、并发控制、可伸缩性、容错以及安全需求等。从这几个角度去观察 HDFS 的设计和实现，可以更清楚地看出 HDFS 的应用场景和设计理念。

- **透明性**

按照开放分布式处理的标准，一般包括 8 种透明性：访问的透明性、位置的透明性、并发透明性、复制透明性、故障透明性、移动透明性、性能透明性和伸缩透明性。对于分布式文件系统，最重要的是希望能达到 5 个透明性要求：

（1）访问的透明性

用户能通过相同的操作来访问本地文件和远程文件资源，HDFS 可以做到这一点。如果把 HDFS 设置成本地文件系统，而非分布式文件系统，那么，读写分布式 HDFS 的程序可以不用修改地读写本地文件，要做修改的只是配置文件。可见，HDFS 提供的访问透明性是不完全的，毕竟它构建于 Java 之上，不能像 NFS 或者 AFS 那样去修改 unix 内核，同时将本地文件和远程文件以一致的方式处理。

（2）位置的透明性

使用单一的文件命名空间，在不改变路径名的前提下，文件或者文件集合可以被重定位。HDFS 集群只有一个目录节点来负责文件系统命名空间的管理，文件的块可以重新分布复制，块可以增加或者减少副本，副本可以跨机架存储，而这一切对客户端都是透明的。

（3）移动的透明性

这一点与位置的透明性类似，HDFS 中的文件经常由于节点的失效、增加或者复制因子的改变或者重新均衡等原因而进行着复制或者移动，而客户端和客户端程序并不需要改变什么，目录节点的日志文件记录着这些变更。

（4）性能的透明性和伸缩的透明性

HDFS 的目标就是构建在大规模廉价机器上的分布式文件系统集群，可伸缩性毋庸置疑，至于性能可以参考它首页上的一些测试基准（benchmark）。

- **并发控制**

客户端对于文件的读写不应该影响其他客户端对同一个文件的读写。要想实现近似原生文件系统的单个文件拷贝语义，分布式文件系统需要做出复杂的交互，例如采用时间戳，或者类似回调承诺（类似服务器到客户端的 RPC 回调，在文件更新的时候；回调有两种状态：

有效或者取消；客户端通过检查回调承诺的状态，来判断服务器上的文件是否被更新过）。HDFS 并没有这样做，它的机制非常简单，任何时间都只会有一个程序在写入某个文件，这个文件经创建并写入之后不再改变，它的模型是“一次写入多次读取”。这与它的应用场合是一致的，HDFS 的文件大小通常是 MB 至 TB 级的，这些数据不会经常修改，最经常的是被顺序读取并处理，随机读很少，因此，HDFS 非常适合 MapReduce 框架或者网页爬虫应用。HDFS 文件的大小也决定了它的客户端不能像某些分布式文件系统那样缓存常用到的几百个文件。

- **文件复制功能**

一个文件可以表示为其内容在不同位置的多个拷贝，这样做带来了两个好处：（1）访问同一个文件时，可以从多个服务器中获取，从而改善服务的伸缩性；（2）提高了容错能力，某个副本损坏了，仍然可以从其他服务器节点获取该文件。HDFS 文件的块，为了容错都将被备份，并且可以配置复制因子，默认是 3。副本的存放策略也是很有讲究的，一个放在本地机架的节点，另一个放在同一机架的另一节点，还有一个放在其他机架上。这样可以最大限度地防止因故障导致的副本丢失。不仅如此，HDFS 读文件的时候也将优先选择从同一机架乃至同一数据中心的节点上读取块。

- **硬件和操作系统的异构性**

由于构建在 Java 平台上，HDFS 的跨平台能力毋庸置疑，得益于 Java 平台已经封装好的文件 IO 系统，HDFS 可以在不同的操作系统和计算机上实现同样的客户端和服务端程序。

- **容错能力**

在分布式文件系统中，尽量保证文件服务在客户端或者服务端出现问题的时候能正常使用是非常重要的。HDFS 的容错能力大概可以分为两个方面：文件系统的容错性以及 Hadoop 本身的容错能力。文件系统的容错性主要借助于以下几个手段：

（1）在目录节点和数据节点之间维持心跳检测。当由于网络故障之类的原因，导致数据节点（DataNode）发出的心跳包没有被目录节点（NameNode）正常收到的时候，目录节点就不会将任何新的 IO 操作派发给那个数据节点，该数据节点上的数据被认为是无效的，因此，目录节点会检测是否有文件块的副本数目小于设置值，如果小于就自动开始复制新的副本，并分发到其他数据节点上。

（2）检测文件块的完整性。HDFS 会记录每个新创建的文件的所有块的校验和。当以后检索这些文件的时候，从某个节点获取块，会首先确认校验和是否一致，如果不一致，会从其他数据节点上获取该块的副本。

（3）集群的负载均衡。由于节点的失效或者增加，可能导致数据分布的不均匀，当某

个数据节点的空闲空间大于一个临界值的时候，HDFS 会自动从其他数据节点迁移数据过来。

（4）维护多个 FsImage 和 Editlog 的拷贝。目录节点上的 fsimage 和 edits 日志文件是 HDFS 的核心数据结构，如果这些文件损坏了，HDFS 将失效。因而，目录节点可以配置成支持维护多个 FsImage 和 Editlog 的拷贝。任何对 FsImage 或者 Editlog 的修改，都将同步到它们的副本上。它总是选取最近的一致的 FsImage 和 Editlog 来使用。 目录节点在 HDFS 是单点存在的，如果目录节点所在的机器错误，手工的干预是必须的。

（5)文件的删除。一个文件被删除时，并不是马上从目录节点移除命名空间，而是放在 / trash 目录下，随时可恢复，直到超过设置时间才被正式移除。

再说 Hadoop 本身的容错性，Hadoop 支持升级和回滚，当升级 Hadoop 软件时出现 bug 或者不兼容现象，可以通过回滚恢复到老的 Hadoop 版本。

- **安全性问题**

HDFS 的安全性是比较弱的，只有简单的与 unix 文件系统类似的文件许可控制，未来版本会实现类似 NFS 的 Kerberos 验证系统。

本章小结

本章介绍 Hadoop 分布式文件系统 HDFS，介绍了块、目录节点、数据节点等相关概念，并介绍了 HDFS 体系结构、命名空间、存储原理、通讯协议、数据错误和异常、尚未实现的功能总结，最后讲述了从 HDFS 看分布式文件系统的设计需求。

总体而言，HDFS 作为通用的分布式文件系统并不适合，它在并发控制、缓存一致性以及小文件读写的效率上是比较弱的。但是它有自己明确的设计目标，那就是支持大的数据文件（TB 级），并且这些文件以顺序读为主，以文件读的高吞吐量为目标，并且与 MapReduce 框架紧密结合。

参考文献

[1] HDFS 详解. http://www.cnblogs.com/chinacloud/archive/2010/12/03/1895369.html

附录 1:任课教师介绍

林子雨(1978－),男,博士,厦门大学计算机科学系助理教授,主要研究领域为数据库,数据仓库,数据挖掘.

主讲课程：《大数据技术基础》

办公地点：厦门大学海韵园科研 2 号楼

E-mail: ziyulin@xmu.edu.cn

个人网页：http://www.cs.xmu.edu.cn/linziyu