# 面向用户体验的智能应用使用模式分析与优化研究

# 摘要

随着Android,ios,Windows Phone等智能操作系统的崛起，移动互联网时代正处于快速成长期，在北京召开的移动互联发展大会上发布的由人民网研究院主编的《中国移动互联网发展报告（2014）》蓝皮书显示，截至2014年1月，我国移动互联网用户总数达8.38亿户，在移动电话用户中的渗透率达67.8%；手机网民规模达5亿，占总网民数的八成多，手机保持第一大上网终端地位。我国移动互联网发展进入全民时代。通过智能终端设备接入互联网获取信息和服务成为当今信息时代的一种新的主流交互方式。紧随着移动互联网发展的大数据时代也接踵而来,“移动互联网”和“大数据”成为当前互联网领域内最火热的两大话题。因此各大互联网公司对用户行为分析的认识程度直接关系到自身能否在新的浪潮得到新的发展,所以对移动互联网的用户行为相关领域的学术研究有着深远的价值,无论是对移动互联网相关企业的市场发展的推动还是移动互联网的格局都有着重要的意义。

在这种背景下，如何在大量数据里帮助用户选择合适的app，以提升用户的使用体验并且能为用户节约使用成本变得重要起来。基于这一点本文主要的工作如下：

一、通过Ambari搭建Hadoop平台，Apache Ambari是对Hadoop进行监控、管理和生命周期管理的开源项目。它也是一个为Hortonworks数据平台选择管理组建的项目。Ambari向Hadoop MapReduce、HDFS、 HBase、Pig, Hive、HCatalog以及Zookeeper提供服务。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。这就为下面的数据分析提供了技术保证。

二、对现有的数据进行规模上的统计分析，例如流量情况，人数情况等，对本文要研究的数据有一个大体的认识

三、通过对用户的app喜好类型及使用习惯进行分析，以便为用户进行相关的推荐，然后在对app进行聚类分析，研究不同类别下相应的app被使用的情况。最终在满足用户喜好和使用习惯的情况的情况下，为用户推荐相应的app以便提升用户的使用体验并能节省使用成本。

**关键词： Hadoop 用户体验 使用模式 推荐 移动互联网 app**

# The research of the usage patterns and optimization of the smartphone application for user experience

# Abstract

With the development of the smart operating systems such as Android, IOS, Windows phone, the mobile internet is in a fast-growing period. According to the Blue Book, which is called 《The development report of Chinese Mobile Internet（2015）》, edited by People's Daily Online, published in Mobile Internet Development Conference held in Beijing, until January 2015, Chinese Mobile Internet users has reached 830 million. More importantly, the penetration rate of mobile phone users has reached 67.8%, mobile phone users has reached 500 million. Smart phones has become the biggest Internet Appliance. Mobile Internet in Chinese has come into the all people age. It has become the main communicate method that people acquire information and service through mobile internet with Intelligent terminal. Follow the development of mobile Internet, big data time arrive in quick succession. ‘Mobile Internet’ and ‘Big Data’ has become the hottest topics in the internet area. The internet company’s understanding of user behavior [have](http://cn.bing.com/dict/clientsearch?mkt=zh-CN&setLang=zh&form=BDVEHC&ClientVer=BDDTV3.5.0.4311&q=%E7%9B%B4%E6%8E%A5%E5%85%B3%E7%B3%BB%E5%88%B0) [a](http://cn.bing.com/dict/clientsearch?mkt=zh-CN&setLang=zh&form=BDVEHC&ClientVer=BDDTV3.5.0.4311&q=%E7%9B%B4%E6%8E%A5%E5%85%B3%E7%B3%BB%E5%88%B0) [direct](http://cn.bing.com/dict/clientsearch?mkt=zh-CN&setLang=zh&form=BDVEHC&ClientVer=BDDTV3.5.0.4311&q=%E7%9B%B4%E6%8E%A5%E5%85%B3%E7%B3%BB%E5%88%B0) [bearing](http://cn.bing.com/dict/clientsearch?mkt=zh-CN&setLang=zh&form=BDVEHC&ClientVer=BDDTV3.5.0.4311&q=%E7%9B%B4%E6%8E%A5%E5%85%B3%E7%B3%BB%E5%88%B0) [on](http://cn.bing.com/dict/clientsearch?mkt=zh-CN&setLang=zh&form=BDVEHC&ClientVer=BDDTV3.5.0.4311&q=%E7%9B%B4%E6%8E%A5%E5%85%B3%E7%B3%BB%E5%88%B0) the development in the new wave. As a result, the study of the research of the mobile internet user behavior has a profound value, which will also has a great benefit on the development of the company and the structure of the mobile internet.

Under this circumstances, it becomes more and more important that helping people find the fittest application and enhance their use experience in massive data. The main work of this paper consist of the next three parts:

First, we build the Hadoop platform through Ambari. The Apache Ambari project is aimed at making Hadoop management simpler by developing software for provisioning, managing, and monitoring Apache Hadoop clusters. Ambari provides an intuitive, easy-to-use Hadoop management web UI backed by its RESTful APIs. The key service of Hadoop is HDFS and MapReduce, The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing.

Second, we analyze the data to have a basic understanding, such as the flow situation and user information.

Third, for one thing, in order to recommend application for users, we analyze the user’s hobby in using app and their use experience. For another, we study the applications to record how they been used and group them into different type. Finally, under the condition that we satisfy user’s hobby and use experience, we would recommend the user more suitable applications which may consume less traffic.

**Key words: Hadoop user experience use pattern mobile internet recommend application**

# 绪论

## 1.1 研究背景

2014年，我国移动互联网用户数量进入中低速稳步增长时期，但建成了全球最大的4G网络，机站数量超过80万。联想、华为和小米等国产移动终端出货量在2014第四季度共同占据了全球约17%的市场份额。中国移动互联网的用户、终端、网络基础设施规模持续稳定增长，移动应用开发者数量超过300万人，同比增长约16%，移动应用生态链初步形成，移动应用的高渗透率与高集中度并存，即时通信、移动支付、电子商务、视频、广告、阅读、医疗等各细分市场都获得长足进步，展现出勃勃生机。移动互联网发展如此迅速，同广大的第三方应用的发展紧密相关，在这种情况下，各种各样的应用程序进入移动互联网用户的视线。以苹果的 App Store 和 Google 的 Google Play 为例，Play Store的Android应用总量达到143万款，而App Store的iOS应用总量为121万款，并且在2015年6月份之前的12个月里，Google Play应用下载量攀升至500亿次，App Store下载量也有250亿次。另外一方面，APP应用的同质化已经成为了这个市场的普遍现象，大量的APP应用软件充斥着人们的生活。在如今生活节奏越来越快的今天，人们想要在APP应用的同质化大海里寻找一款创意十足的手机APP软件十分困难，加上新的APP应用必定会替代旧的APP应用，用户对APP应用的喜新厌旧是APP应用行业里必须面对的一个事实。很多人都会问：在如今同质化严重的海量APP应用中，该如何挑选高质量的应用软件呢?登上APP应用商店中，上千百万的APP应用让用户眼花缭乱，各种各样的APP，该如何选择正确，符合自己的使用习惯并且能话费更少的流量，确实也是一个难题。一般来讲，现在的情况下用户在选择app的时候会从APP应用的类别、用户评论、APP软件的UI界面、开发商等等来选择APP应用。类别可以帮助用户提高搜索的精准度。用户评论帮助用户判断APP应用的质量。UI界面则影响着APP应用的第一形象。开发商可以看出APP应用的整体质量，但是这个过程都是在用户亲自去查看的情况下进行的，也就是说这还得需要用户去判断。并且以视频类应用来讲，如果用户想查找一款能够替代当前使用的视频类应用，并且在移动网络下能够花费较少的流量，通过刚才的方法怕是做不到的。

在新的市场环境下,运营商必须在原有优势的基础上,做出顺应移动互联网发展的调整。在移动互联网络中,用户的需求趋于娱乐化、媒体化、实用化。在机遇与挑战的面前,运营商必须第一时间了解潜在市场的用户需求,就能适应市场,为用户提供更完善更个性化的服务,就能在竞争中占有一席之地。如何获取用户的需求信息,最根本的途径是针对用户的上网数据进行分析,即实现用户行为分析。用户行为分析对精准营销的巨大作用是不言而喻的。在移动互联网潜在客户类型识别过程中,移动互联网用户行为分析可以有效、快捷地获取用户潜在需求,能够持续不断地吸引客户。对于成熟稳定的客户群体可以通过用户行为分析不断增强和改善用户服务感受,提升用户体验。

在这个移动互联网迅速发展的时代，大数据时代也悄然而至，如何处理海量的数据成为人们开始讨论的话题，就像刚才提到的，在大量的app使用记录里如何提取出对用户有用的信息，如何分析出用户使用app的相关行为特征，传统的数据处理方法可能无法满足我们的要求，这里就需要我们借助大数据处理的工具-Hadoop，来完成数据分析的要求。Hadoop是一个开发和运行处理大规模数据的软件平台,是Apache的一个用java语言实现开源软件框架，实现在大量计算机组成的集群中对海量数据进行分布式计算.Hadoop框架中对的核心设计就是：HDFS和MapReduce. HDFS提供了海量数据的存储, MapReduce提供了对数据的计算.除此之外，hadoop还具有其他的很多优势：

高可扩展性：Hadoop是一个高度可扩展的存储平台，因为他可以存储和分发横跨数百个并行操作的廉价的服务器数据集群。不同于传统的关系型数据库系统不能扩展到处理大量的数据，Hadoop是能给企业提供涉及成百上千TB的数据节点上运行的应用程序。

成本效益：Hadoop的架构被设计为一个向外扩展的架构，可以经济的存储所有公司的数据供以后使用， Hadoop提供数百TB的存储和计算能力。

灵活性更好：Hadoop能够使企业轻松访问到新的数据源，并可以分析不同类型的数据，从这些数据中产生价值

快：Hadoop拥有独特的存储方式，用于数据处理的工具通常在与数据相同的服务器上，从而导致能够更快的处理器数据。

容错能力：使用Hadoop的一个关键优势就是他的容错能力。当数据被发送到一个单独的借点，该数据也被复制到集群的其它节点上，这意味着在故障情况下，存在另一个副本可供使用。

通过使用Hadoop对数据进行分析处理，为用户进行应用推荐，使用户能够得到更合适的应用以及更好的用户体验的同时，能够得到更少的流量消耗，这也迎合了未来的发展趋势。

## 1.2 国内外研究现状

Google的Google Play应用商店，也会在一定程度上对用户进行应用推荐。具体方案包括：第一、根据你的好友下载并且对软件评价的情况，为你推荐类似的应用，这里的好友是你的Google社交软件Google+里的好友，他们来自世界各地，有着不同的生活习惯，甚至和我们的生活习惯千差万别，所以这种推荐并没有太大的意义；第二、应用商店需要你对你下载的应用程序进行评分，你进行评分之后，系统会为你推荐类似功能的软件，这主要是根据个人喜好来推荐，并没有考虑到软件对流量的消耗以及用户对流量的承受能力。在我国，虽然这些年流量正在变得越来越便宜，但是应用程序对流量的消耗也变得越来越多，所以移动流量问题始终是困扰移动互联网用户的一大问题。国内的应用市场，就360手机助手来说，推荐策略包括几个方面：第一，根据你下载的应用程序，告诉你下载该应用的其他用户还下载了什么，这是根据相似下载来推荐相似应用的一种策略；还有推荐的方式是下载排行等等。

360手机助手也没有考虑的软件的流量问题，当然他们可能也没有相关的信息。

文章1研究了基于安全性和隐私方面的应用推荐，主要从app需要的权限问题入手，分析app需要的权限，例如，位置权限、读取通讯录权限、使用摄像头权限等等，从使用这些权限可能会带来的安全问题，隐私问题来讨论，并且结合这些app的特征以及流行度，为用户推荐不同安全级别的的应用软件，最终为用户的隐私和个数据安全性提供了保障。文章2主要是将社交网络和个性化推荐结合在了一起，提出了一套基于社交网络的应用推荐系统，通过分析朋友圈、社交圈子中具有相似特征，相同喜好的用户，从而为他们找到可能感兴趣的应用。

基于hadoop的分布式存储功能，以及高效的数据处理能力，本文使用hadoop数据分析平台对软件的流量监控信息进行分析，对用户的使用模式进行讨论，并对用户行为喜好软件建造模型，量化喜好和流量，根据软件的一些特征参数匹配出最大化满足用户喜好模型以及最小化流量使用的应用程序。

## 1.3 本文主要工作

本文主要提出了一套基于用户使用应用程序的行为偏好，为用户推荐使用流量最少的应用程序的一套理论。

## 1.4 文章的结构

# 数据分析平台说明

## 2.1 Hadoop 简介

### 2.2.1 Hadoop发展史

Hadoop起源于Apache Nutch，一个开源的网络搜索引擎。Nutch项目始于2002年，一个可以运行的网页爬取工具和搜索引擎系统很快浮出水面。但是后来，开发者认为这一架构可拓展度不够，不能解决数十亿网页的搜索问题。2003年发表的一篇论文为此提供了帮助，文中描述的是谷歌的产品架构，称为谷歌分布式文件系统(Google Distributed File System),简称GFS。GFS或者类似的其他架构，可以解决他们在网页爬取和索引的过程中产生的超大文件的存储需求。特别关键的是，GFS能够节省系统管理(如管理存储节点)所花的大量时间。在2004年的时候，他们开始着手实现一个开源的实现，及Nutch的分布式文件系统NDFS。

MapReduce编程思想室友Google工程师Jeffrey Dean与2004年提出来的。于此同时，Google也发表了GFS、BigTable等底层系统以应用MapReduce模型。2007年Google公司发布了Google’s MapReduce Programming Model-Revisted论文，在该论文中进一步详细介绍了Google MapReduce模型以及Sazwall并行处理海量数据分析语言。Google公司以MapReduce为基石，逐渐成为互联网发展的领头羊，然后Google公司并没有透露MapReduce的具体实现细节。

Hadoop之父Doug Cutting开发出的Hadoop是MapReduce的开源实现，是的MapReduce技术能够快速、容易的来到我们身边。2006年Hadoop项目 从Nutch项目中脱离出来，并正式成为Apache组织中一个专注于DFS和MapReduce的开源项目，目前Hadoop已经成为Apache的顶级项目。如今Hadoop不仅致力于应对网络流量的科学研究，而且还涉及搜索引擎、广告优化、机器学习等领域，并成为IT产业里优秀的大数据平台。

### 2.2.2 什么是Hadoop

提升单台计算机的的速度已经慢慢走到了“天花板“，CPU的速度已经不可能再大幅度提升。人们一直希望通过增加计算机的数量提升运算和数据处理速度，例如希望同时在300台计算机上处理数据，让处理这批数据的速度变成0.1小时。Hadoop是一个分布式计算框架，它能在由大量廉价的硬件设备组成的集群上运行应用程序，并且为应用程序提供了一组既稳定又可靠的接口。Hadoop计算框架的目的是构建一个具有高可靠性和良好拓展的分布式操作系统。随着云计算的逐渐流行，这一项目被越来越多的个人和企业应用。

Hadoop主要用来处理大量数据，它实现了MapReduce一样的编程模式和框架，能在由大量计算机组成的集群中运行海量数据并进行分布式运算。它处理的海量数据能够达到PB级别，并可以让应用程序在上千个节点中进行分布式处理，处理的方式是可靠的、高效的、可伸缩的。Hadoop是可靠的如果计算元素或者存储数据失败，他可以启动或维护多个工作数据副本，确保失败的节点重新对数据进行分布式处理。Hadoop是高效的，他的工作方式是并行的，采用这种方式可以加快处理数据的速度。Hadoop是可伸缩的，可以处理不同级别的数据，达到能够处理PB级别的数据。除此之外，Hadoop依赖于社区服务器，所以他的成本很低。

Hadoop自带Java语言编写的框架，在Linux平台上运行是非常理想的。Hadoop平台上的应用程序也可以用其他语言编写，如C++，Python，Ruby等等。

Hadoop族群包括很多项目，如下表所示：

|  |  |
| --- | --- |
| 项目名称 | 项目说明 |
| HDFS | 分布式文件系统，是GFS的开源实现 |
| MapReduce | 分布式并行编程模型和程序执行框架，Google公司MapReduce的开源实现 |
| Common | 整个Hadoop项目的核心，包括一组分布式文件系统和通用I/O的组件与接口（序列化、Java RPC和持久化数据结构） |
| Avro | 一种支持高效、跨语言的RPC以及永久存储数据的序列化实现 |
| Pig | 一种数据流语言和运行环境，用以检索非常大的数据集，运行在MapReduce和HDFS集群上 |
| Hive | 一个分布式、按列存储的数据仓库。Hive管理HDFS之中的数据，并提供基于SQL的查询语言（运行时翻译成MapReduce作业）用以查询数据 |
| Hbase | 一个分布式、按列存储的数据库。使用HDFS作为底层数据，同时支持MapReduce的批量式计算和点查询 |
| Mahout | 一个在Hadoop上运行的机器学习类库 |
| Zookeeper | 一个分布式、可用性高的协调服务。 |
| Cassandra | 一套开源分布式NoSQL数据库系统，最初由Facebook开发，用于存储收件箱等简单格式数据。 |

本文中用到的Hadoop组件主要是Hadoop的核心组件HDFS和MapReduce，以及Hive，其他的组件我们暂且不讨论。

## 2.2 HDFS的体系结构

Hadoop分布式文件系统(HDFS)被设计成适合运行在通用硬件(commodity hardware)上的分布式文件系统。它和现有的分布式文件系统有很多共同点。但同时，它和其他的分布式文件系统的区别也是很明显的。HDFS是一个高度容错性的系统，适合部署在廉价的机器上。HDFS能提供高吞吐量的数据访问，非常适合大规模数据集上的应用。HDFS放宽了一部分POSIX约束，来实现流式读取文件系统数据的目的。HDFS在最开始是作为Apache Nutch搜索引擎项目的基础架构而开发的。HDFS是Apache Hadoop Core项目的一部分。

HDFS有着高容错性（fault-tolerant）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。

1. HDFS有以下几个主要特点：

处理超大文件：存储的一个超大文件可以达到数GB级、数TB级、数PB级。

集群规模动态扩展：节点动态加入到集群，可以数百数千个

流式数据读写：HDFS的设计思想“一次写入，多次读取”，一个数据集一旦由数据源生成，就会被复制分发到不同的存储节点中，然后响应各种各样的数据分析任务请求。

运行于廉价的商用机器集群上：HDFS设计时充分考虑了可靠性、安全性及高可用性，因此Hadoop对硬件要求比较低，可以运行于廉价的商用机器集群，无需昂贵的高可用性机器

2.HDFS的局限性：

不适合低延迟数据访问： HDFS是为了处理大型数据集，主要是为了达到高的数据吞吐量而设计，这就可能以高延迟作为代价。10毫秒以下的访问可以无视hdfs，不过hbase可以弥补这个缺

无法高效存储大量小文件： namenode节点在内存中存储整个文件系统的元数据，因此文件的数量就会受到限制，每个文件的元数据大约150字节

不支持多用户写入及任意修改文件 ：不支持多用户对同一文件进行操作，而且写操作只能在文件末尾完成，即追加操作。

3. HDFS结构模型如下所示：



一个HDFS集群是由一个NameNode和若干个DataNode组成的。NameNode主节点是主服务器，管理文件系统的命名空间和客户端对文件的访问操作；DataNode是集群中的一般节点，负责节点数据的存储。客户端通过NameNode和DataNode节点交互访问文件系统，联系NameNode获得文件的元数。而文件的I/O操作则是直接和DataNode进行交互。HDFS的文件通常是按照64MB被切分成不同的数据块（Block）的，每个数据块尽可能的分散存储在不同的DataNode中，若干个数据块存放在一组DataNode上。

假设客户端要访问一个文件，首先，客户端熊NameNode中获得组成该文件的数据块位置列表，即知道数据块存储在哪些DataNode上；然后，客户端直接从DataNode上读取文件数据，在这一过程中NameNode不参与文件传输。

4．HDFS的数据复制与存放

HDFS可以在多个机器上可靠并分布式的存储超大文件，每个文件按照64MB大小的数据块被切分成多个数据块，所以从这一点来讲，如果数据大小正好是64的倍数，那么所有的数据块都是同等大小的，但是一般来讲，最后一个数据块达不到一个数据块的大小。

首先，HDFS数据的复制。为了提高数据的容错性，HDFS允许用户指定每个文件的副本数目。其中数据的复制由NameNode来管理，每个DataNode会通过心跳信号周期性的向NameNode发送本节点的相关数据信息。如果NameNode没有收到这个信号的话，则说明这个DataNode可能出现了问题。数据复制结构图如下所示：



数据进行复制之后，副本的存放是HDFS可靠性的性能的关键所在。HDFS和其他的分布式文件系统不同的一点就是HDFS优化的副本存放策略。一般来讲，默认情况下，数据块的副本系数是3，也就是说，每一个数据块都有3个副本。其中一个副本直接存放在本地机架的节点上，另外一个副本存放在本地机架的另外一个节点上，第三个副本放在不同机架的节点上。这样的好处是，不同的机架之间不需要进行复杂的数据传输。这样一来，写操作的效率就大大的提高了。并且和节点相比，机架有着更少的错误，这在另外一方面保证了数据的可靠性和可用性。在存在三个副本的情况下，当进行数据处理时，为了降低整体的带宽消耗和读取时延，HDFS会尽量优先读取距离程序最近的数据副本。这里还需要注意的一点是，当用户删除某个HDFS上的文件时，这个文件并没有立即从HDFS里面消失，而是被移到了HDFS系统上的一个trash文件夹下。这样设置的目的是防止用户误操作。一旦发生了错误删除的情况我们还可以在这个文件夹下将这个文件恢复。但是这个文件夹下的文件是有时间保存限制的，默认情况下是在360分钟，也就是6个小时下进行完全删除。到时候就无法恢复错误删除的文件了。当然这个时间限制是可以重新设置的。如果用户减少了数据的副本系数，这是NameNode会选择多余的副本进行删除。NameNode会通过心跳告知DataNode删除相应的数据副本块。

## 2.3 MapReduce的体系结构

MapReduce是一种编程模式，这种模式采用的是分布式的计算方法。MapReduce致力于解决大规模数据集的问题。它可以用在非常广泛的应用程序中，既然要解决大规模数据集的问题，就要考虑从一部分数据开始，利用局部分析的方法，将大规模数据集的问题分解成小部分数据的问题。也就是我们熟悉的分而治之的数据处理方法。就像上一结HDFS里我们讲到的一样，在用户进行数据处理之前，数据集已经分布在各个节点上了。我们在处理数据时，每个节点会优先处理存储在你本地的数据来进行map处理，map处理过后，再讲数据进行合并（combine），同时进行一定的排序（sort），最后分发到reduce节点上。

MapReduce包括两个核心的操作，map和reduce。简单来说，map就是一个映射的过程，从一组数据到另外一组数据。这个过程通过map函数来实现。简单举个例子进行乘以2的映射就变成了。Reduce过程则进行对映射后的数据进行规约处理，这一过程同样通过reduce函数来实现。如果对映后的数据进行求和计算则最终的运算结果就是20。概括来讲，map的任务就是将大任务分解成多个小任务，而reduce的工作就是将每个小任务处理的结果汇总起来。当然这个过程还有很多复杂的问题，如分布式存储、工作调度、负载均衡、容错处理和相关的网络通信等。这些事情都是MapReduce框架来解决的。不需要我们用户关心这些问题。MapReduce的处理流程图如下所示：



从图中可以看出，数据输入之后，被分割成不同的区块，然后通过map任务，分配给不同的节点来同时处理数据的不同部分，这就达到了分布式的效果。map处理完成之后，通过reduce将结果汇总。MapReduce的具体流程如下所述：

1. 任务调度与执行

每一个MapReduce任务都是有一个JobTracker和多个TaskTracker两种节点共同控制完成，其中TaskTracker主要是任务节点，而JobTracker是主节点。JobTracker管理TaskTracker，管理内容主要包括，将mappers和reducers分配给空闲的TaskTracker，然后TaskTracker负责这些任务并开始执行，这个过程是并行进行的。TaskTracker必须运行在本文之前讨论过的DataNode节点上，所以我们称DataNode既是存储数据的节点，也是对数据进行计算的节点。JobTracker和TaskTracker之间的通信是通过心跳信号来维护，如果超过一个时间间隔内，JobTracker没有收到TaskTracker的信号，JobTracker会认为这个节点已经宕机。JobTracker会将这个节点之前负责的任务再分配给其他空闲的TaskTracker节点。如果JobTracker发生故障，整个任务将会失败。

1. 列表处理（list processing）

从概念上来讲，MapReduce是将输入的元素列表转变成另外一组输出元素列表的过程。这个过程会持续两次，分别位于我们之前说的map和reduce两个过程之中

1. Mapping数据列表（lists）

MapReduce的第一步我们称为mapping，这一过程中，有一部分数据作为mapper函数的输入，这个输入过程会一直进行，直到数据全部输入完毕。数据输入后，mapper函数会将结果单独写到一个数据数据元素里，这一过程如下图所示：

其实，一个mapper函数就是对一些有独立元素组成的概念上的列表（例如，一个测试成绩的列表）中的每一个元素进行指定的操作。这个过程中，每个元素都是独立被操作的，原始的数据并没有被修改，因为这一过程创建了一个新的列表来保存结果。这也就意味着，这一操作是可以高度并行进行的。从而使MapReduce在很多并行计算的领域非常有用。因为DataNode既是数据节点又是计算节点，也就是说计算节点和数据节点在同一台机器上，这有利于MapReduce在执行任务的时候，尽量的减少数据在网络中的传输，大大的降低了执行任务对带宽的需求，在一定程度上避免了网络的“瓶颈”。其次，MapReduce后输入的split块应该设置为不大于HDFS数据块的大小，这样就能保证输入的数据不会跨越两台计算机存储，更加利于本地存储。

Mapping处理过程的输出数据会按照key值分成R份，其中R代表reducer的个数，这个划分是通过hash函数来实现的，Hadoop的默认方法是hash（key）mod R，也就是将输出数据的按照其key值均匀的将数据分配到R个reducer中，这样做的意义在于，某一范围内的key一定由某个reducer来处理，这一过程也称之为“shuffle”。这一过程如下所示



1. 读取中间结果

“Shuffle”过程之后，数据达到mapper数据缓冲区的80%（可以由用户设定）后，mapper的输出结果会被写到本地磁盘上，于此同时，这个mapper的TaskTracker告知JobTracker本任务运行的状态以及输出结果相关的位置信息等等。JobTracker再去告知相应的reducer的TaskTracker去获取mapper输出的结果。需要注意的是，由于mapper的输出结果是按照key值分配的，所以reducer获取数据的时候，只会获取属于自己这个reducer的一部分数据，获取到数据后，再执行reduce函数进行相关处理。

1. Reducing 数据列表（lists）

Reduce的过程其实就是一个聚集数据的过程。Reduce函数接收一个包括数据的迭代器，对这个迭代器里的数据进行一个聚合操作，最后输出一个值，这一过程如下所示：通过列表迭代器对输入数据进行reducing操作来输入聚合结果。Reducing一般的作用是生成“总结”数据，也就是把原始的大规模数据，通过一系列的数据处理变成更小的总结数据。需要注意的是，有几个reducer就会产生几个结果，当然reducer的个数可以由用户来设定，如果用户不设定，系统会根据输入数据的情况，自行判断需要的reducer的个数。

1. 分布和可靠性

MapReduce通过把大规模的数据集分发给网络上的每一个节点来实现这一过程的可靠性，与此同时，每个节点会周期性的把相关的已经完成的工作和本节点的其他一些状态更新报告回来。如果一个节点并没有进行相关的报告，并且时间间隔超过了一个预设的时间。那么这个节点会被主节点标记为已经死亡。主节点会把分配给这个节点的相关数据分发给其他的节点。

## 2.4 Hive的体系结构

Hive是Facebook向Apache基金会贡献的一个Hadoop子项目，国内的淘宝也是Hive的主要使用者，并且为Hive的发展做出了贡献。Hive其实就相当于一个数据仓库，它可以将结构化的一些文件，一些数据等映射成类似数据库中的一张数据表。这样一来用户在处理数据的时候就可以将数据当作数据库中的数据那样来处理。而Hive比传统的数据库先进的地方在于，当在Hive中用传统的类似于SQL的语句处理数据时，Hive会自动将命令执行为一个MapReduce任务，这就大大提高了数据的处理能力。这就使得那些会写Hadoop的MapReduce程序的用户，可以通过Hive的方式来使用MapReduce。而MapReduce开发人员可以通过写mapper函数和reduce函数并且以插件的形式来支持Hive做更复杂的数据处理。Hive的SQL语句和传统的关系型数据库SQL语句略有不同，他们的区别如下表所示：

|  |  |  |
| --- | --- | --- |
| 特性 | SQL | HiveSQL |
| 更新 | Update，insert，delete | Insert overwrite table（填充整个表或分期） |
| 事务 | 支持 | 不支持 |
| 索引 | 支持 | 不支持 |
| 延迟 | 少于1秒 | 分钟级别 |
| 数据类型 | 整数、浮点数、定点数、文本、二进制串和时间 | 整数、浮点数、布尔型、字符串、数组、映射、结构 |
| 函数 | 数百个内置函数 | 几十个内置函数 |
| 多表插入 | 不支持 | 支持 |
| Create table select | Sql-92中不支持，有些数据库支持 | 支持 |
| 选择 | SQL-92 | From字句中只能有一个表或视图。支持偏序的sort by。可限制返回行的数量limit。不支持having |
| 连接 | SQL-92 支持或变向支持 | 内连接，外连接，半连接，映射连接和带提示的SQL-92语法 |
| 子查询 | 在任何字句中支持 | 只能在from字句中 |
| 视图· | 可更新 | 只读 |
| 扩展点 | 用户定义函数  存储过程 | 用户定义函数  MapReduce脚本 |

从表中可以看出，HiveSQL除了不支持事务和索引外，其他的SQL特性都能够直接或者间接的支持或者部分支持。这说明HiveSQL能够完成我们对数据处理的大部分要求。

Hive的体系结构如下图所示：



Hive的数据管理可以从三个方面来介绍，他们分别是元数据存储，普通数据存储和查询计划。Hive的元数据是存储在关系型数据库中的，例如MySQL，derby等等。本文用到的集群中Hive的元数据存储是选择的MySQL。元数据主要包括了表的名字，表的列等一些信息，还有表的分区，相关的属性和表中数据存储所在的目录这些信息。Hive的具体数据是存储在HDFS中的，所以Hive的大部分对数据的操作都会转化成MapReduce任务。而最基本的select \* from tablename这样的带\*的查询就不会生成MapReduce任务。Hive里没有特定的数据储存的格式，并且也没有数据索引信息，所以用户可以随意的在Hive里组织表相关信息。但是创建表的时候我们需要指定表的行分隔符以及表的列分隔符以保证Hive能够解析数据。

## 2.5 Hadoop集群管理工具-Ambari介绍

Apache Ambari现在是一个Apache的顶级项目，早在2011年8月，Hortonworks引进Ambari作为Apache Incubator项目，制定了Hadoop集群极致简单管理的愿景。在两年多的开发社区显著成长，从一个小团队，成长为Hortonworks各种组织的贡献者。Ambari用户群一直在稳步增长，许多机构依靠Ambari在其大型数据中心大规模部署和管理Hadoop集群。它提供一个直观的操作工具和一个健壮的Hadoop API,可以隐藏复杂的Hadoop操作，使集群操作大大简化，首个版本发布于2012年6月。

Apache Ambari是对Hadoop进行监控、管理和生命周期管理的开源项目。它也是一个为Hortonworks数据平台选择管理组建的项目。Apache Ambari是一种基于Web的工具。 Apache Ambari 支持HDFS、MapReduce、Hive、Pig、Hbase、Zookeeper、Sqoop和Hcatalog等的集中管理。也是5个顶级Hadoop管理工具之一。

Ambari主要取得了以下成绩：

* 通过一步一步的安装向导简化了集群供应。
* 预先配置好关键的运维指标（metrics），可以直接查看Hadoop Core（HDFS和MapReduce）及相关项目（如Hbase、Hive和Hcatalog）是否健康。
* 支持作业与任务执行的可视化与分析，能够更好地查看依赖和性能。
* 通过一个完整的RESTful API把监控信息暴露出来，集成了现有的运维工具。
* 用户界面非常直观，用户可以轻松有效地查看信息并控制集群。

Ambari使用Ganglia收集度量指标，用Nagios支持系统报警，当需要引起管理员的关注时（比如，节点停机或磁盘剩余空间不足等问题），系统将向其发送邮件。

此外，Ambari能够安装安全的（基于Kerberos）Hadoop集群，以此实现了对Hadoop 安全的支持，提供了基于角色的用户认证、授权和审计功能，并为用户管理集成了LDAP和Active Directory。

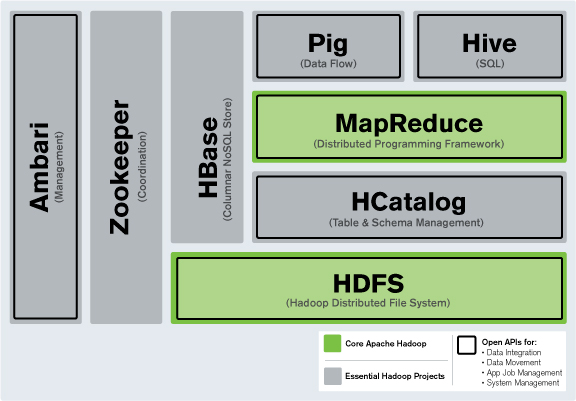


图 1

本文处理数据用到的的Hadoop集群是通过Ambari来进行搭建和管理的。搭建好的Ambari管理界面如下所示，管理界面包含和很多集群方面的信息。

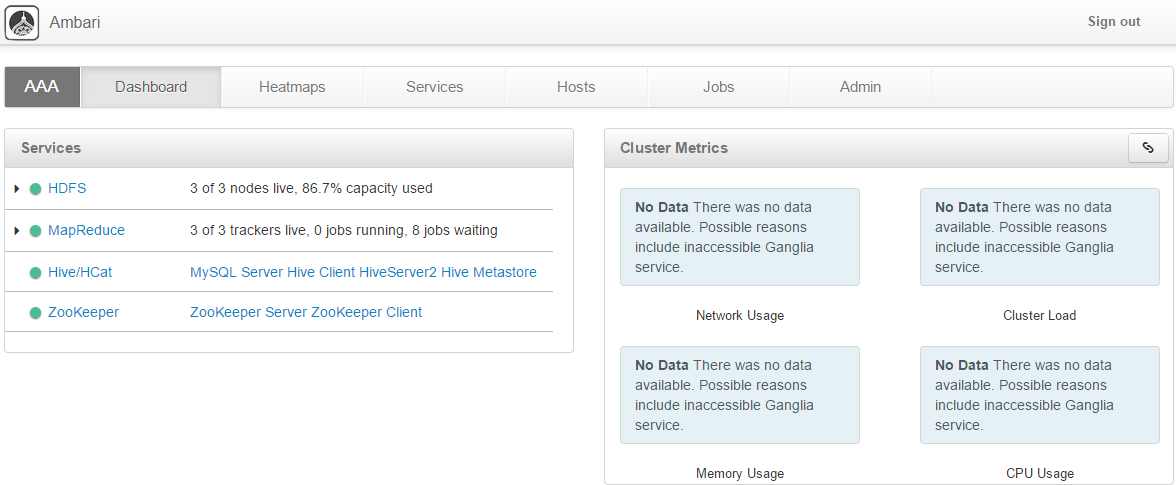
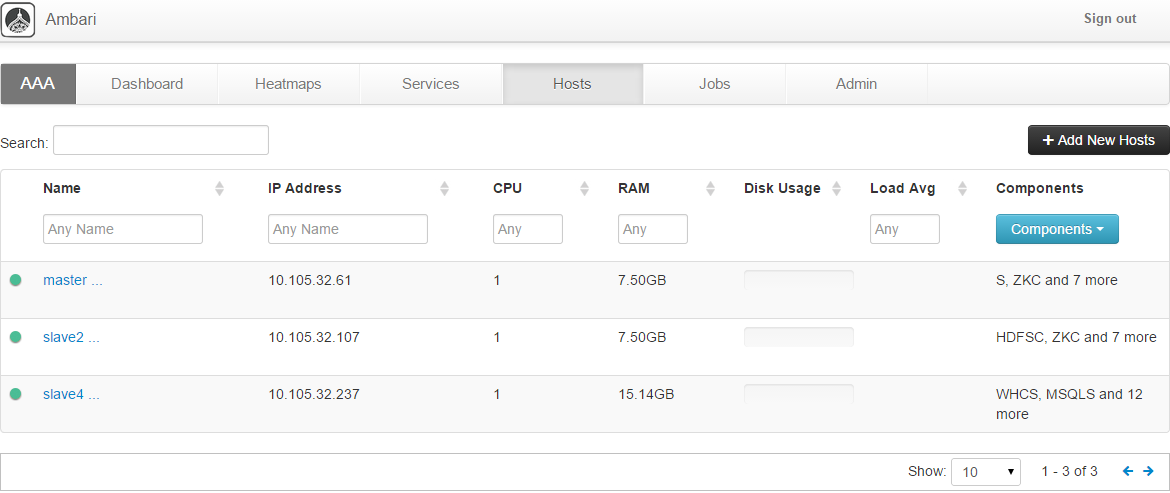


图 2

我们的集群只使用了3台机器，其中一台是是主节点，另外两台是从节点，如图所示。

从图中可以看出，本集群的名字叫做AAA，已经运行的服务包括HDFS，MapReduce，Hive，Zookeeper。其中HDFS服务包括3个node，3个node都处于启动状态，并且HDFS磁盘使用率已经达到了86.7%。其次，MapReduce也有三个节点，且都处于运行状态，进行该截图时，集群并没有进行mapreduce任务，所有界面显示0 jobs running，0 jobs waiting。Hive也处于启动状态，hive启动后相应启动了一个MySQL server进行相关数据的存储。除了介绍过的这三个主要的组件外。本Ambari系统，还启动了一个zookeeper服务。ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，它包含一个简单的原语集，分布式应用程序可以基于它实现同步服务，配置维护和命名服务等。Zookeeper是hadoop的一个子项目，其发展历程无需赘述。在分布式应用中，由于工程师不能很好地使用锁机制，以及基于消息的协调机制不适合在某些应用中使用，因此需要有一种可靠的、可扩展的、分布式的、可配置的协调机制来统一系统的状态。Zookeeper的目的就在于此。本文并没有使用到zookeeper，故在此不做多余说明。点击上图中的hosts，可以查看集群的主机信息，点开后界面如下所示：



从这个截图可以看出，集群共有三台主机，主机名分别为master，slave2，slave4。各主机IP地址位于同一个网段内。

从Ambari的集群管理界面可以获取到大量的集群的相关信息，例如哪些节点的datanode没有启动，哪些节点的tasktracker没有启动，以及每个节点的空间使用情况，当运行一个MapReduce任务的时候，能够查看任务运行的详细信息。从图2 可以看出集群中的3个节点的hdfs服务和MapReduce服务都运行正常。一旦集群出现异常信息，便可在此界面查看相应的异常信息。其次job运行的详细过程也可以在这个管理界面中看到。这不仅极大的方便了集群的管理，还为我们查看任务运行的状态提供了方便的接口。

## 2.6 本章小结

本章主要介绍了Hadoop基本的信息，详细说明了Hadoop核心组件HDFS的基本数据存储情况和一些特点。另外也介绍了另一核心组件MapReduce的基本框架和运行原理，还结合HDFS做了数据相关的分析。并且，本章进一步介绍了本文数据分析平台的基本情况，这也是本文数据处理的基础。除此之外本文对Hadoop的插件Hive也做了详细的说明。以便在后来使用的过程中更深的了解。有了这些技术，本文的数据分析才能够得以进行

# 基本数据分析

## 3.1 本文使用的数据集说明

### 3.1.1 数据来源

本文使用的数据集来源于一款测速软件产生的用户相关的信息。软件的测试分为两种，一种是用户端主动发起的，需要消耗流量的主动测试，另一种是不需要消耗额外的流量，只对程序消耗的流量进行监控，可以分别对用户消耗流量的APP进行监控，可以详细记录用户详细的流量使用行为，和使用APP的具体网速。被动流量监控主要针对的是APP使用网络流量进行的监控。当网络发生变化时，这部分数据进行全部更新；当小区发生切换时被动流量数据库全部更新；当网络和小区都没有发生变化时，只针对流量变化的APP数据库进行更新。如下图所示



Figure 1 被动流量监控模块流程图

程序启动后，监控程序就已经在后台运行了，这一过程每天都会产生大量的数据。当产生了相关的数据或者数据更新后，保存到手机端的数据库之后，程序会定时每隔20s触发一次上传，上传到公网服务器的数据库中，上传的数据库主要有几个表，首先是被动监控各个app后台使用流量的数据表，我们称之为app\_traffic\_db，这也是本文数据分析的主要数据来源，还有一个数据表，是用户主动测试当前网速时，向数据库上传的数据表，我们称之为testinfo\_db，接下来也会使用一点。还有其他一些软件会收集但是本文不予分析的表。当数据传到公网服务器上的数据库之后，由于公网数据库的容量有限，只有1G的空间，上传的数据很容易就能把数据库填满，导致后来的数据会上传失败。其次，考虑到数据的安全性等相关问题。我们将实验室备份到了本地的数据库里。这一过程采用增量备份的方式。也就是，每天都会获取新增的数据，将新增的数据备份的本地数据库，同时将已经成功备份到本地数据库的数据，在公网服务器的数据库里将其删除。这样公网数据库也不会出现数据长满的情况。备份到本地数据库之后，由于数据量较大，MySQL已经处理不了我们的一些需求。进一步我们将本地数据库里的数据导入了Hadoop里的HDFS里进行数据存储。在此基础上，对数据进行分析。整个数据存储的过程，如下图所示：



### 3.2.2 数据格式

上一节我们讨论过，本文主要分析用到的数据表是app\_traffic\_db和testinfo\_db，其中app\_traffic\_db是对手机上安装的其他软件的几个监控信息，具体内容如下表所示：

|  |  |  |
| --- | --- | --- |
| 表项 | 数据类型 | 解释 |
| id | int(10) | 自增长的关键字 |
| package\_name | text | app包名（例如：com.tencent.qq） |
| app\_name | text | app的名称（例如：QQ） |
| uid | text | app的标识id（每个app唯一） |
| network\_type | text | Wi-Fi/mobile |
| mobile\_type | text | 具体移动网络类型（  network\_type为Wi-Fi的情况下，本列值为Wi-Fi，network\_type为mobile的情况下，本列可能的值为，GPRS，edge，hspa，lte等等） |
| cell\_id | text | 移动蜂窝网小区识别码 |
| wifi\_bssid | text | 一种特殊的Ad-hoc LAN的应用的id |
| start\_time | text | 本条记录程序监控起始时间 |
| end\_time | text | 本条记录程序监控结束时间 |
| upload\_traffic | text | 当次监控软件上传流量 |
| download\_traffic | text | 当次监控软件下载流量 |
| date | text | 监控日期 |
| time\_index | text | 详细监控日期时间 |
| imei | varchar(60) | 手机imei（也是用户的唯一标识） |
| sdk\_version | varchar(20) | 本测速软件的SDK版本 |
| gps\_lat | varchar(20) | 定位纬度 |
| gps\_lon | varchar(20) | 定位经度 |
| location\_type | varchar(60) | 定位类型 |

这个数据表，相对来说数据量比较大，用户只要安装了这款软件，软件就会在后台时刻进行着监控其他软件的相关信息，并且定期上传监控的信息。和这个数据表不同的是，另外一张数据表testinfo\_db，数据量就比较小，因为只有当用户安装了这款软件并且使用软件进行了一次测速之后，软件才会上传本次测速相关的一些信息，比如测得的网速，网络类型等等，具体上传信息参见下表：

|  |  |  |
| --- | --- | --- |
| 表项 | 数据类型 | 解释 |
| id | int(10) | 自增长的关键字 |
| ping | varchar(10) | 时延(单位：ms) |
| ave\_downloadSpeed | varchar(10) | 平均下载速度(单位：KB/s) |
| max\_downloadSpeed | varchar(10) | 最大下载速度(单位：KB/s) |
| ave\_uploadSpeed | varchar(10) | 平均上传速度(单位：KB/s) |
| max\_uploadSpeed | varchar(10) | 最大上传速度(单位：KB/s) |
| rssi | varchar(10) | 信号强度（单位：DBM） |
| gps\_lat | varchar(20) | 测试点的经度 |
| gps\_lon | varchar(20) | 测试点的纬度 |
| location\_type | varchar(60) | 定位类型 |
| imei | varchar(60) | 手机的imei |
| server\_url | text | 测试服务器地址 |
| ant\_version | varchar(60) | app版本号 |
| detail | text | 详情 |
| time\_client\_test | varchar(20) | 客户端测试时间 |
| time\_server\_insert | varchar(20) | 服务器端插入时间 |
| networkType | varchar(10) | Wi-Fi/4G/3G/2G |
| operator\_name | varchar(20) | CMCC/CUCC/CTCC |
| wifi\_bss\_id | varchar(20) | Wi-Fi bssid |
| cell\_id | varchar(60) | plmn+lac+cid/plmn+sid+bid+nid |
| province | varchar(20) | 测试点的省份 |
| city | varchar(20) | 测试点的城市 |
| street | text | 街道地址 |
| location\_detail | text | 定位详细信息 |
| upload\_traffic | varchar(20) | 上行测试流量 |
| download\_traffic | Varchar(20) | 下行测试流量 |

## 3.2 相关的数据分析

前面已经讨论过相关的数据信息和数据的相关流程信息，在前面讨论的这些内容的基础上，我们开始对数据进行初步的分析。这些分析的主要步骤是，编写MapReduce程序或者直接使用Hive对数据进行处理，然后，将处理之后的结果从HDFS上面拷贝到本地服务器的本地磁盘上。为了对处理结果进行展示，能够让其他人也看到我们的处理结果，本文处理的数据结果又被上传到了公网服务器上（此服务器和之前用户上报监控数据的服务器是同一个服务器），在服务器上通过一些程序将数据展现出来，展示地址为，<http://buptant.cn/autochart/>。这一流程如下图所示：



这一部分的数据分析，有利于更好的了解数据集的情况，对本文后面的分析有很大的帮助。

### 3.2.1 数据总体情况分析

截止到本文完成之前，数据集中统计的总的WiFi流量已经达到15TB的级别，移动网络流量也达到了将近2TB的量。除此之外，被动监控数据表app\_traffic\_db中的监控记录已经达到了将近**12000000**多条。下面，本文将对数据集中的一些情况做简要的分析。

首先本文分析了各网络制式下的用户数，也就是说，使用不同网络制式的用户各有多少，这里的网络制式，是本文前面数据表里提到过的具体的网络信息，统计方式，就是以网络制式为分组然后计算每个分组下不同的IMEI个数，每个IMEI对应着每一个单独的用户。下面是统计结果：

图表中1xRTT属于中国电信网络的2.5G网络，对应中国联通，中国移动的2.5G网络是EDGE。Ehrpd网络是电信3G向4G演进过程中的一个过渡。还有一点需要说明的是，同一用户的网络可能会跳转，比如一个中国移动4G网络的用户，在网络不好的时候可能会变为TD-SCDMA的3G网络，甚至会变为EDGE网络，这都是有可能的。从这个图表里可以看到，电信用户在三大运营商里还处于少数。

下图统计了每天活跃的用户数

从图中可以看出，用户数大致趋于变少。在做相关的数据分析时，本文会对这一特点进行评估，以保证相关推荐的合理性和正确性。

下图统计了，每天监控到的其他软件的总的流量信息

从这幅图可以看出，用户在使用网络上网时，大部分用户还是选择使用WiFi网络，这都是因为WiFi网络对手机来说是免费的，并且WiFi的速度要比普通的移动网络速度要快。尽管现在的各大运营商都在声称调整相关的资费，并且提升网速，但是实际上还是换汤不换药，本质上根本没有什么改变。毕竟用户每个月的流量是有限的，并且如果使用流量看视频，用户每个月的包月流量几乎在一天之内就会消耗完毕。移动网络流量始终还是一个限制用户上网的关键因素，所以在这种情况下，为用户推荐流量少的应用是很必要的，毕竟WiFi网络，还没有移动网络那么普及。

下图统计了，每天上传的监控的记录数量的情况，这个记录和每天活跃的用户目基本上成正比，虽然用户不多，但是用户产生的记录数还是有一定的数量的，这也很好的为我们的分析提供了数据支撑

以上几个图表，从数据集的大小方面对数据做了一个统计意义上的分析，通过这些分析，我们可以对数据的规模情况有一个基本的了解，接下来的几个小节，出发点会更深一步。

### 3.2.2 用户方面的情况分析

本文的主题在于为用户推荐相关的应用情况，所以针对这个数据集，对数据集中一些基本的用户情况进行分析还是很有必要的。从下面的一些分析中我们可以对用户在流量使用方面有一个基本的了解，除此之外，我们还可以在用户的流量使用模式上有一个基本的认识。下面是分析的具体情况。

首先我们分析了，月均流量这个概念，对用户平均一个月消耗的流量做了一个基本的统计，结果如下所示：

下图是WiFi情况下的：

从这两幅图里可以看出，移动网络情况下，大多数用户月均流量消耗集中在10M-100M，100M-1G，以及1G-10GB之间，并且100M-1G之间的用户数量最多，与之对应的WiFi情况下，用户平均每月在使用WiFi网络时花费的流量大约集中在100MB-1GB，1GB-10GB，以及10GB-100GB之间，由此可知，用户在WiFi上花费的流量比在移动网络上花费的流量整整高了一个数量级。所以，用户在移动网络上使用流量还是有很大的限制的，如果我们能为用户进行相关的推荐，比如说在移动网络每月100MB-1GB流量的情况下，使用原来类别下其他APP能够体验到，使用原来APP每月1G-10G流量的体验。这样一来，就能够让用户花费较少的流量体验相同的使用感受，这也就达到了本文的目的。

### 3.2.3 APP方面的情况分析

由于在后面我们需要对app相关情况进行分析，在这里先大体对app的总体情况了解一下，下面两幅图是本文统计的，分别在移动网络和WiFi情况下，不同app平均每天消耗的流量分布图情况。通过这两幅图我们可以对不同的app每天的在不同的网络下的流量消耗情况有一个基本的了解。

从这两幅图可以看出，移动网络情况下，每天消耗流量1M的app种类占到了大约85%。这也可以理解，一方面，由于某些app的使用频率不高，每天可能就打开一次，对于大部分应用程序来说都不需要传递很多数据，所以消耗的流量也不是很多，这一部分应用程序，对用户的使用习惯影响可能也不大，我们不需要密切关注。而消耗流量大于1M，小于10M，以及日均消耗流量在10M到100M之间的app，这一部分app的种类大约占得比例在百分之十几，从用户的使用情况来看，这一部分app是我们需要密切关注的app。与之对应的wifi情况下app的分布情况，与移动网络下稍有不同，日均销量流量在1K-1M之间的app占得比例大约在75%，较移动网络有所下滑，在日均消耗流量在1M到10M以及10M到100M之间的app比例攀升到了接近25%。这和移动网络的情况区别还是很大的。

由上面的分析，我们了解到用户在移动网络下和WiFi网络下使用app的一个变化，那就是在WiFi网络下，用户会毫无顾虑的去使用各种各样的app，而从来不需要去考虑流量的问题，这也是很正常的，毕竟在wifi网络下，流量是不需要额外花钱的。有的用户可能是用家里的宽带搭的无线路由器，这样的话，上网是不计流量费的。相对来说很划算。而在移动网络下，流量是用有时候就会显得捉襟见肘了，毕竟用户在使用移动网络时，每个月的流量都是固定的，超过了就得再花钱买，并且很多用户每个月的流量包不一定够用，这就导致了用户在使用移动网络上网的时候，使用app时格外的小心。花费流量大的app是不敢使用的。由于这种情况的存在，如何让用户在移动网络下能够放心的使用各种app，而不必太过于担心流量问题，正是本文需要考虑的问题。本文接下来的内容将围绕着这个问题继续展开讨论。

### 3.2.4 本章小结

本章内容主要是基于现有的数据，对数据的基本情况做一下介绍，然后从用户的角度，以及app的角度，分析了WiFi网络和移动网络情况下，用户和app方面的一下区别，WiFi网络下，用户不用考虑流量的问题，而能够放心的使用各种app，移动网络下，用户流量使用情况就会有所收敛。基于这种情况，也正论证了本文提出的问题，如何能让用户在使用移动网络的时候，能花费更少的流量而又不丧失原来的app使用体验。最好能让用户在移动网络下体验到WiFi网络下的使用体验，这是本文努力研究的目标。

# 第四章 推荐模型的建立

在已知用户所使用的各类app的情况下，其中包括用户的app使用时间，使用频度等等，研究用户的使用模式及偏好。关于偏好模型的建立，首先需要对app进行分类，本文分类的原则很简单：第一，数据集中有软件名称这一项，第二，参考各大应用市场对软件的分类情况。将相应的软件属于的类别打上标签，以便在以后使用。本文对app分的类主要包括视频类，软件下载类，音乐类以及浏览器类这四大类。对于社交类软件，用户本身对社交类的软件依赖较大，并且使用频繁，用户的朋友关系大多维系在社交类软件上，用户很难改变社交类软件的使用习惯，所以本文暂且不对社交类软件相关app进行推荐。而其他几类软件都有很多类似的软件可供用户选择，具有很高的推荐价值。

## 4.1 用户使用偏好模型的建立

使用偏好旨在分析用户是否对某一类软件的使用喜好超出了大众的平均使用情况，并根据超出的情况来选择性的为用户进行推荐。本文的使用偏好模型和用户对某一类软件使用的时间，使用的频度有关。我们把使用偏好用来表示，则，其中是指用户在对某一类软件使用时间方面的评估，得出用户在某一类软件的时间方面的偏好程度。是指在某一类软件上使用频率方面的评估，得出用户在某一类软件的频度方面的偏好程度。这两者之间并不存在干扰，使用时间长并不影响使用频度，反之亦然。偏好模型的建立是本文分析的一个基础。

### 4.1.1 使用时间的分析

衡量一个用户对一款软件的使用喜好程度，最直接的一个体现就是，用户在一款软件上花费的时间，时间能够说明一切问题。用户在一款软件上花费的时间越长，则能说明用户比较喜欢使用这款软件。但是这个时间的长度该怎么去划分是一个问题，也就是说，用户一天内在一款软件或者说一类软件上使用多长时间算是喜好这款软件，使用多长时间又算是对这款软件使用的还算可以，这个度还需要去权衡。在实际情况中，只有比较才能得出一个用户是不是比另一个用户更喜欢使用某一款或者某一类软件。在得到每一个用户在某一款软件或者某一类软件平均每天花费的时间后，就可以进一步处理这个问题。首先，统计出平均每一个用户在某一类软件上平均每天的使用时间。以视频类软件为例，先统计每一个用户每天在在视频类软件上使用的时间，得到一个时间集，这个时间集就是一个用户每天花费在某一类软件上的时间，然后统计每天在视频类软件上使用的平均时间，也就是对这个时间集内的每个时间求平均，得到，其中为用户使用的天数，为用户编号，如表所示

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 天数  用户  时间 | 第1天 | 第2天 | 第3天 | ……… | 第n天 | 用户平均时间 |
| 用户1 |  |  |  | ……… |  |  |
| 用户2 |  |  |  | ……… |  |  |
| 用户3 |  |  |  | ……… |  |  |
|  |  |  |  | ……… |  |  |
| 用户m |  |  |  | ……… |  |  |
| 平均时间 |  |  |  |  |  |  |

这样就得到了每个用户在视频类软件平均每天的使用时间，有了这个时间集，就可以比较每个用户在这一类软件上使用时间的异同，就能够进一步区分这个不同。把所有用户在视频类软件每天的平均使用时间做一个平均，就得到了平均每个用户在视频类软件上平均每天花费的时间，其中为用户的人数。有了这个最终的平均值，就可以将不同的用户对这一类软件的使用程度进行区分了。用用户每天在视频类软件平均每天的使用时间去减去平均每个用户在视频类软件上平均每天的的使用时间，就得了用户平均每天在视频类软件使用的时间和平均值之间的一个差值，这里记为，其中，这里有了这个时间差值，只能将不同用户在这一类软软件上的使用时间区分出来，只能说明一个用户比另一个用户在这一类软件上花费的时间更多或者更少。上面得到的差值，可能很大，也就是说用户在这一类软件上每天花费的时间远远超过了平均值；也可能为0，这时意味着用户在这类软件上花费的时间处于一个平均水平；更有可能为负值，也就说这类用户在这类软件上花费的时间很少，这类用户也不是我们将要关心的对象，可以忽略掉。这个差值无法定量的表现问题，如何将这个差值，这个不同通过概率或者什么方式表现出来是一个问题。经过大量的调查研究，发现逻辑回归是在这种情况下比较常用的一种方法，逻辑回归用于估计某种事物的可能性。比如某用户购买某商品的可能性，某病人患有某种疾病的可能性，以及某广告被用户点击的可能性等。着这里逻辑回归表示用户使用某一类软件的可能性。

下面引入逻辑回归

为上面计算出来的差值，其图形如下所示



通过逻辑回归，将用户平均每天的使用时间和平均每个用户平均每天的时间映射到这个区间内，从而判断出用户使用这一类软件的可能性。从图中可以看出差值越大，则通过逻辑回归计算出来的值就会越大，也就是用户使用这类软件的可能性越大，反之亦然，而当差值为0的时候，也就是之前文章讨论过的那样，这时说明用户在这一类软件的使用时间处于平均值，通过逻辑回归计算出来的值是0.5，这也是符合常理的，差值为0，相比差值比较大的用户和差值比较小的用户，0.5的可能性，还是正确的。从上面的分析我们得出用户在时间使用方面的偏好程度为



根据这个公式，就能够将用户在某一类软件上的使用时间，映射到一个概率值上上，使用时间比平均值越大，计算出来的值越大，反之相同。这个计算出来的值能够反映用户在某一类软件上使用的时间，进而能够部分反映用户在这类软件上的使用偏好程度。

### 4.1.2 使用频度的分析

除了使用时间能够说明用户对某一类软件的使用偏好外，使用频度也能在一定程度上说明这个问题。但是使用频度的分析，相对来说比较简单，由于很难确切的定义一天使用多少次某一类软件就说明使用该类软件很频繁。也很难说一天内使用某一类软件次数比较多，而使用另外一类软件次数少，我们就设定用户对于使用次数多的软件更喜欢用，这种假定也不是很合理。比如用户一天内只使用了一次视频软件看了2个小时的电视剧，在此期间用QQ进行了几次聊天，这种情况无法判断用户更喜欢用哪一款软件。根据本文使用的数据集情况，由于对于一些用户的软件使用情况的搜集信息表明，从开始收集信息的日期，到最后用户卸载这款可以收集用户信息的软件，在这一段时间里，从收集到的信息可以看出，用户每天会使用多款软件，但是有些软件，用户并不是每天都在使用。也就是说，用户使用某些软件的时候有可能每天都在用，也有可能用过一次就不在使用了。基于这种情况本文将使用频度定义在每一天，也就是说我们统计的时间天数里，统计用户使用某一类软件的天数，然后计算使用某一类软件的天数和统计的总的天数的比值，用这个比值来表明使用一类软件的频率。我们用来表示统计到的用户记录的总的天数，来表示用户使用某一类软件的天数，这样的话频度方面的偏好程度



其中表示用户使用的天数，表示观察到的所有天数。根据这个比值，就可以确定用户在某一类软件的使用频度情况。

根据用户使用某一类软件的可能性和用户使用的频度，我们确定用户对某一类软件最终的使用偏好，。这个值是本文分析的一个基本，文章后面的推荐将基于这个值进行相关的推荐。

### 4.1.3 用户使用时间段的分析

## 4.2 APP画像的分析

在对用户相关软件使用偏好做了相关的分析之后，了解了用户在使用app时的一些行为特征，接下来对app的一些特征做一些分析。对app的分析主要分为以下几个方面：首先分析的是最重要的一点，那就是app的平均流量消耗，这也是本文分析的一个根本问题，本文需要找出消耗流量最少的app并且符合用户行为特征，并对用户进行相关app推荐。在找出消耗流量最少的app之后，还需要了解app其他的信息，以完善app画像的分析。其次，需要分析app的使用时间，对一个用户来说，在什么时候用什么样的app是很不同的，以视频类app来讨论，有的用户可能在中午喜欢观看视频，有的用户可能在晚上喜欢观看视频，这时就需要将各个时间段中不同视频类app的使用情况加以分析，然后对在不同时间段内观看视频的用户推荐相同时间段内同类别其他消耗流量少的app。