

---

## Table of Contents

大数据实践	1.1
Demo	1.2

---

## Hadoop安装

使用虚拟机安装	2.1
CentOS 7上安装	2.2
Untitled	2.3

---

## 计算广告

简介	3.1
生态模式	3.2
点击率预估	3.3

---

## HDFS

简介	4.1
实验一：分布式存储	4.2
实验二：数据冗余	4.3
HDFS设计目标	4.4
实验三：设计分布式系统	4.5
HDFS架构设计	4.6
HDFS Shell	4.7
实验四：HDFS Shell入门	4.8
实验五：数据拷贝和分布式拷贝	4.9

---

## MapReduce

简介	5.1
实验一：一道真实的大数据面试题	5.2
实验二：让问题稍微复杂些	5.3
Hadoop 架构设计	5.4
Hadoop Streaming	5.5
实验三：统计特征分布	5.6

---

---

实验四：数据倾斜	5.7
Key, Value, Partition和Sort	5.8
实验五：二次排序	5.9
Map Reduce的瓶颈	5.10

---

## Hive

简介	6.1
HQL	6.2
实验一：HQL入门	6.3
实验二：使用HQL进行统计	6.4
HQL转换成MapReduce	6.5
实验三：HQL性能瓶颈	6.6
DAG	6.7

---

## 机器学习

简介	7.1
特征工程	7.2
实验一：构造特征	7.3
实验二：特征选择	7.4
模型训练与预测	7.5
实验三：Logistic Regression	7.6
模型评估	7.7
实验四：评估指标的选择	7.8
可视化	7.9
实验五：模型可视化	7.10

---

## Spark

简介	8.1
RDD & DataFrame	8.2
实验一：使用Spark构造模型特征	8.3
MLlib	8.4
Spark SQL	8.5

---

## 数据可视化



# 大数据实践

## 介绍

在2003, 2004和2006这三年中, Google连续发表了三篇大数据领域重量级的论文: GFS (Google File System), MapReduce和BigTable。基于这三篇论文, Doug Cutting在一个大型全网搜索引擎 ([Nutch] (<http://nutch.apache.org/>)) 项目中实现了DFS和MapReduce。之后Doug Cutting加入了Yahoo, 该项目也被独立出来命名为[Hadoop] (<https://hadoop.apache.org/>), 成为了Apache的顶级开源项目, 大数据技术由此开始。



### 为什么是Google引领了大数据技术?

Google作为世界上最大的搜索引擎, 有着对大规模数据存储和计算的需求, 在互联网规模成指数级发展的21世纪, 这个需求越来越强烈。另一方面, Google将大数据技术应用在了计算广告领域, 广告效果大幅提升, 从而给公司带来大量营收, 这部分营收又反过来促进Google不断投入到大数据技术的研究当中。在这样的正向激励下, Yahoo、Facebook、Microsoft, Amazon等科技巨头也纷纷加入了大数据领域, 促使这项技术得到快速的发展。

### 什么是计算广告?

计算广告的目的是在特定语境下找到用户和广告之间的“最佳匹配”, 即根据用户搜索的关键词和浏览的网页等信息, 推荐给用户对应的广告, 并根据用户的点击率持续评估和优化广告效果。可以看到, 相比传统广告近乎随机的投放, 计算广告可以充分利用大数据技术做到更加精准。因此可以说, 在广告领域的成功让大数据技术得到了充足的资源支持。



### 本书你可以学到什么?

我们以计算广告中最重要的任务: 点击率预估 (CTR Prediction) 作为背景, 阐述大数据技术是如何有效解决这个任务, 包括HDFS, MapReduce, Hive, Spark和数据可视化。另外, 本书按照大数据技术发展的时间轴设计了众多实验内容, 帮助读者循序渐进的了解各项技术的起源, 并在实践中理解各项技术的特点, 以及这些技术是如何解决点击率预估任务中不同的子任务。最终, 本书希望读者可以在完成点击率预估整体任务的同时, 对大数据技术有一个全面的了解。



## 本书你学不到什么？

本书假定读者已经具备一定的计算机编程基础，全书使用Python作为实验语言，但本书并不是一本Python教程。如果想更好的了解Python编程语言，请读者学习或查阅《Python基础教程》。另外，本书受篇幅所限，并不能对所有大数据技术进行全面讲解。但读者仍可以通过本书的链接到大数据技术对应的官网上进行学习。最后，本书目的在于帮助读者快速上手应用大数据技术解决实际问题，本身并不是一本大数据技术的API手册。鉴于大数据技术发展日新月异，编者强烈建议读者通过技术官网去查阅最新的API文档。



GFS: <https://static.googleusercontent.com/media/research.google.com/zh-CN//archive/gfs-sosp2003.pdf>

MapReduce: <https://static.googleusercontent.com/media/research.google.com/zh-CN//archive/mapreduce-osdi04.pdf>

BigTable: <https://static.googleusercontent.com/media/research.google.com/zh-CN//archive/bigtable-osdi06.pdf>

## Demo

### Getting Super Powers

Becoming a super hero is a fairly straight forward process:

```
$ give me super-powers
```

Super-powers are granted randomly so please submit an issue if you're not happy with yours.

Once you're strong enough, save the world:

```
// Ain't no code for that yet, sorry
echo 'You got to trust me on this, I saved the world'
```

## 使用虚拟机安装

## CentOS 7上安装

# Untitled

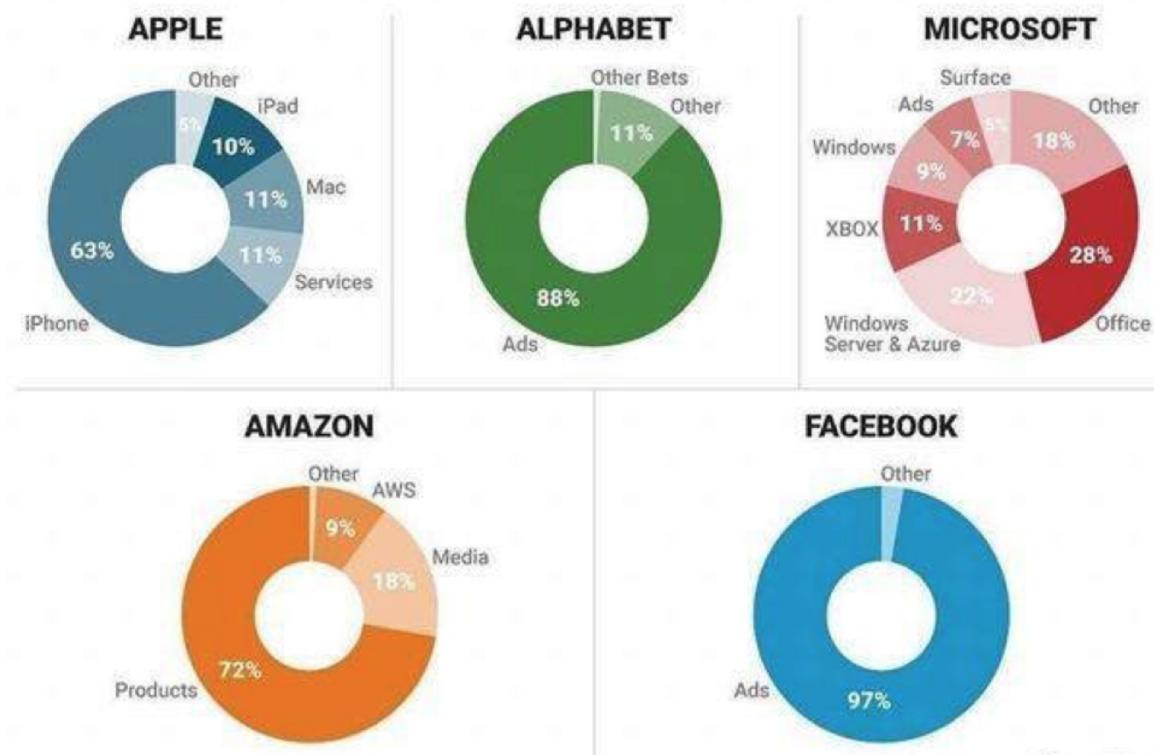
## 简介

在开始了解各项大数据技术之前，我们先简单介绍下我们要使用大数据解决的问题背景。可以毫不夸张的说，是广告支撑起了互联网模式的生态系统，使得免费模式可以成为成功的商业模式。从Business Insider发布的2017年科技公司营收报告来看，在全球最大的五家技术公司中，可以戏说有两家是广告公司，其中Google广告收入占公司总营收的比例为88%，Facebook这一比例甚至达到了97%。可以想象，没有广告收入的支撑，Google和Facebook不可能有持续的资源去支持大数据领域的发展。那么，从另一方面来看，在一个正向激励的系统中，大数据技术或许也应该对这些科技公司获取更多的收入有着帮助。就让我们带着这个疑问来看看背后的逻辑，究竟大数据技术是如何帮助科技公司提高广告收入的。



TECH CHART OF THE DAY

### REVENUE STREAMS OF THE BIG 5 TECH COMPANIES



为什么广告需要计算？

谈到广告大家肯定不会陌生，几乎我们生活中的每一个角落都会有广告出现。但是传统广告行业有一个自己的“哥德巴赫猜想”：

Half the money I spend on advertising is wasted; the trouble is I don't know which half.

- John Wanamaker

翻译过来就是：有一半的广告花费是无效的，但问题是我不知道是哪一半。计算广告则给这个问题提供了一个有效的解决办法，即通过计算去衡量应该展示什么广告，又通过计算去评价广告的效果。

### 计算广告怎么计算？

在一个场景下，对于给定的用户和合适的广告之间找到一个最佳的匹配

对于搜索广告

搜索词，搜索词历史用户ID，用户浏览历史，用户注册数据，其他数据可以展现的广告，不同公司的不同产品广告

以上可以转换成有约束条件的优化问题Maximize  $f(c, u, a)$

一旦有了数学问题，剩下的问题就是计算了最优解虽然不可能，但是可以逐渐找到效果更好，效率更高的办法

其中一个解决办法就是点击率预估

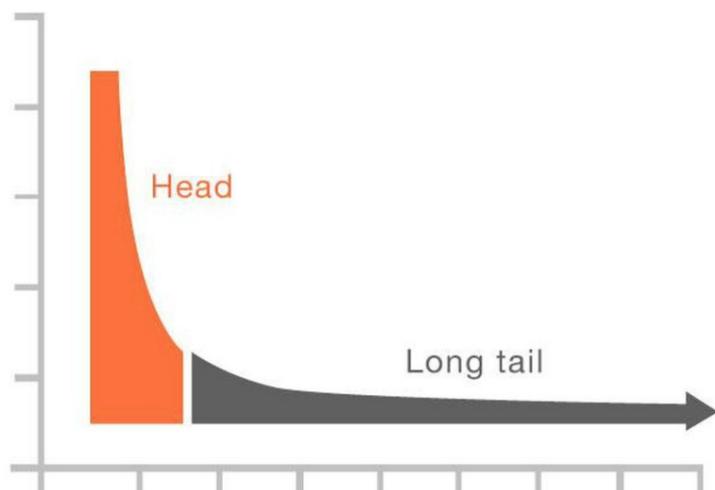
### 计算广告与大数据的关系？

计算广告和大数据的关系需要从互联网一个著名概念“长尾理论”说起。该理论的提出人Chris Anderson认为只要存储和流通的渠道足够大，需求不旺或销量不佳的产品共同占据的市场份额可以和那些数量不多的热卖品所占据的市场份额相匹敌甚至更大。



## THE NEW MARKETPLACE

blockbusters  
& megabrands  
that are distributed through  
traditional retailers such  
as Wal-Mart, Macy's,  
Neiman-Marcus and others



An infinite array of smaller, niche brands  
distributed through the Internet.



## 生态模式

计费方法

CPM CPC CPA

CPM 按照展现付费，即千次展现多少钱

CPC 按照点击付费，即每次点击多少钱

CPA/CPT 按照行为/成交付费，即广告带来的购买行为后付费

竞价机制

英式拍卖 (English Auction)

英式拍卖又叫公开增价拍卖OAB (Open Ascending Bid)，卖家提供物品，在物品拍卖过程中，买家按照竞价阶梯由低至高喊价，出价最高者成为竞买的赢家。为了保证竞价收敛，一般会为竞价设定一个终止时间。这种模式非常容易理解，平时电影电视中经常看到。

荷兰式拍卖 (Sealed-bid Auction)

荷兰式拍卖亦称公开减价拍卖ODB (Open Descending Bid)，其过程与英式拍卖过程相反：竞价由高到低依次递减直到第一个买家应价时成交的一种拍卖方法。

第一价格密封拍卖FPSB (The first-price sealed auction)

第一价格密封拍卖，买方将自己的出价写在一个信封里，众多买方进行投标，同一时间揭晓信封价格，出价最高者竞价成功。

第二价格密封拍卖SPSB (The Second Price Sealed Auction)

又称为维克里拍卖 (Vickrey Auction)，其拍卖过程和第一价格密封拍卖过程一样，由出价最高的买家获得物品，但他只需要支付所有投标人中的第二高价。

第二高价优势

没必要去思考第二高价究竟是多少，按照收益出价即可

避免风险，每个广告所支付的钱不会高于成本

收益产生于我的广告收益大于别人的广告收益的部分

生态上趋势是最优（最赚钱）的广告得到展示

## 点击率预估

## 简介

HDFS的思想可以说起源于Google的GFS。2000年左右，当Google的爬虫不断从互联网上抓取回新的网页后，Google发现即使使用当时存储容量最大的机器也无法存储全部需要索引的网页内容了，更不用说2000年之后的互联网的规模还在成指数型增长。因此Google设计了自己的分布式文件系统，即Google File System - GFS。该系统包括几百甚至几千台普通的廉价设备组装的存储机器，同时被相当数量的客户端访问，提供数据的读写服务。



## 实验一：分布式存储

为了帮助大家理解分布式系统如何工作，请大家完成（查看）下面的python程序

1. 创建一个Data目录，并按照0 ~ 9 的顺序创建10个目录
2. 该程序需要把1W行数据存储在这10个目录里，但是每个目录下最多只能存储2000行数据。
3. 完成步骤2之后，程序需要接受一个0~9999的数字，返回之前该数字所对应的那行数据。

实验代码：

```
for line in
```

## 实验二：数据冗余

在实验1的基础上，我多增加一个步骤，实验变为：

1. 创建一个Data目录，并按照0 ~ 9 的顺序创建10个目录
2. 该程序需要把1W行数据存储在这10个目录里，但是每个目录下最多只能存储2000行数据。
3. 我选择1~9任意一个数字，然后删掉你的目录
4. 程序需要接受一个0~9999的数字，返回之前该数字所对应的那行数据。

### 实验代码\

```
#/
```

# HDFS设计目标

## 硬件错误

硬件错误是常态而不是异常。HDFS可能由成百上千的服务器所构成，每个服务器上存储着文件系统的部分数据。我们面对的现实是构成系统的组件数目是巨大的，而且任一组件都有可能失效，这意味着总是有一部分HDFS的组件是不工作的。因此错误检测和快速、自动的恢复是HDFS最核心的架构目标。

## 流式数据访问

运行在HDFS上的应用和普通的应用不同，需要流式访问它们的数据集。HDFS的设计中更多的考虑到了数据批处理，而不是用户交互处理。比之数据访问的低延迟问题，更关键的在于数据访问的高吞吐量。POSIX标准设置的很多硬性约束对HDFS应用系统不是必需的。为了提高数据的吞吐量，在一些关键方面对POSIX的语义做了一些修改。

## 大规模数据集

运行在HDFS上的应用具有很大的数据集。HDFS上的一个典型文件大小一般都在G字节至T字节。因此，HDFS被调节以支持大文件存储。它应该能提供整体上高的数据传输带宽，能在在一个集群里扩展到数百个节点。一个单一的HDFS实例应该能支撑数以千万计的文件。

## 简单的一致性模型

HDFS应用需要一个“一次写入多次读取”的文件访问模型。一个文件经过创建、写入和关闭之后就不需要改变。这一假设简化了数据一致性问题，并且使高吞吐量的数据访问成为可能。Map/Reduce应用或者网络爬虫应用都非常适合这个模型。目前还有计划在将来扩充这个模型，使之支持文件的附加写操作。

## “移动计算比移动数据更划算”

一个应用请求的计算，离它操作的数据越近就越高效，在数据达到海量级别的时候更是如此。因为这样就能降低网络阻塞的影响，提高系统数据的吞吐量。将计算移动到数据附近，比之将数据移动到应用所在显然更好。HDFS为应用提供了将它们自己移动到数据附近的接口。

## 异构软硬件平台间的可移植性

HDFS在设计的时候就考虑到平台的可移植性。这种特性方便了HDFS作为大规模数据应用平台的推广。

## 实验三：设计分布式系统

如果说让你根据HDFS的设计目标去设计一个自己的分布式文件系统，那么你应该如何设计？请大家在纸上完成自己的设想，简单提示如下：

1. 程序怎么能知道文件存储在什么地方？
2. 如果有一个磁盘故障，数据如何保证不丢失？

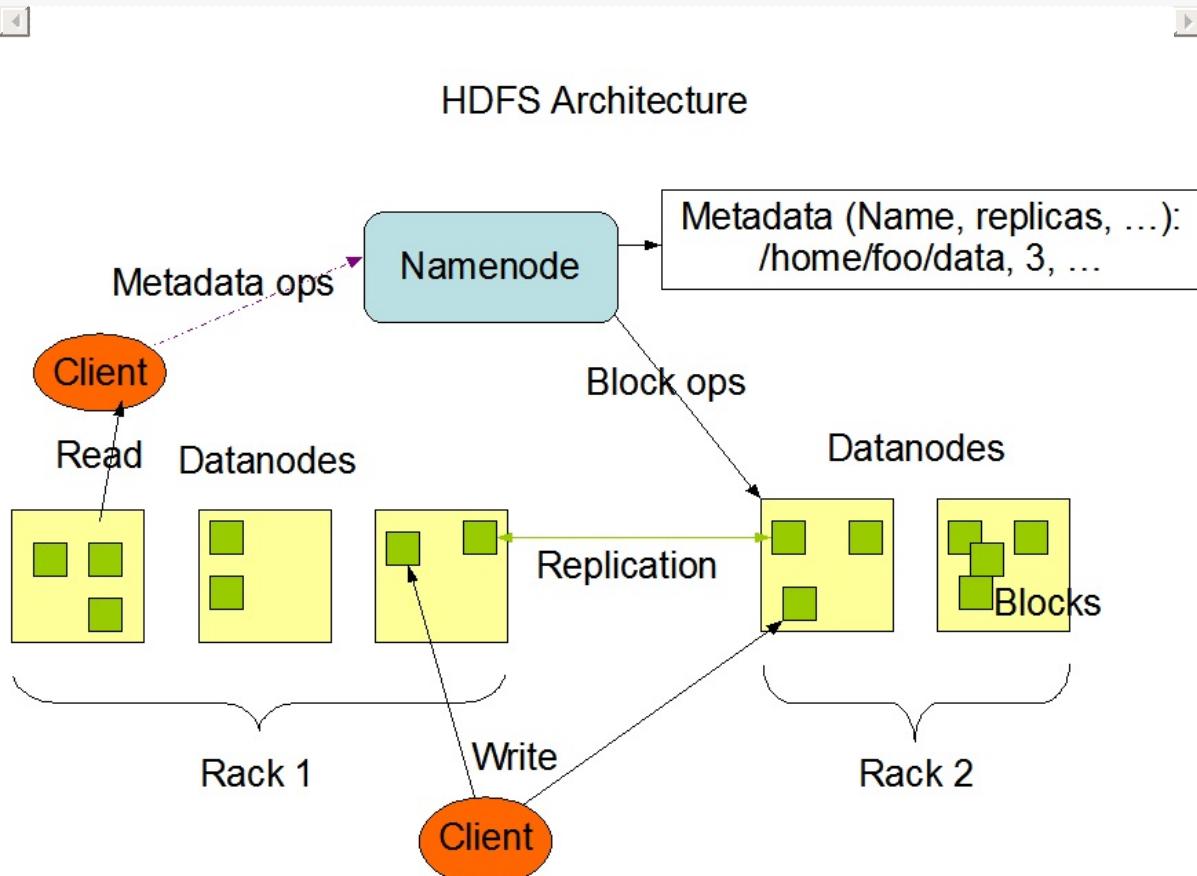
# HDFS架构设计

## Namenode 和 Datanode

HDFS采用master/slave架构。一个HDFS集群是由一个Namenode和一定数目的Datanodes组成。Namenode是一个中心服务器，负责管理文件系统的名字空间(namespace)以及客户端对文件的访问。集群中的Datanode一般是一个节点一个，负责管理它所在节点上的存储。HDFS暴露了文件系统的名字空间，用户能够以文件的形式在上面存储数据。从内部看，一个文件其实被分成一个或多个数据块，这些块存储在一组Datanode上。Namenode执行文件系统的名字空间操作，比如打开、关闭、重命名文件或目录。它也负责确定数据块到具体Datanode节点的映射。Datanode负责处理文件系统客户端的读写请求。在Namenode的统一调度下进行数据块的创建、删除和复制。

Namenode和Datanode被设计成可以在普通的商用机器上运行。这些机器一般运行着GNU/Linux操作系统(O/S)。HDFS采用Java语言开发，因此任何支持Java的机器都可以部署Namenode或Datanode。由于采用了可移植性极强的Java语言，使得HDFS可以部署到多种类型的机器上。一个典型的部署场景是一台机器上只运行一个Name node实例，而集群中的其它机器分别运行一个Datanode实例。这种架构并不排斥在一台机器上运行多个Datanode，只不过这样的情况比较少见。

集群中单一Namenode的结构大大简化了系统的架构。Namenode是所有HDFS元数据的仲裁者和管理者，这样，用户数据永远不会流过Namenode。



## 文件系统的名字空间 (namespace)

HDFS支持传统的层次型文件组织结构。用户或者应用程序可以创建目录，然后将文件保存在这些目录里。文件系统名字空间的层次结构和大多数现有的文件系统类似：用户可以创建、删除、移动或重命名文件。当前，HDFS不支持用户磁盘配额和访问权限控制，也不支持硬链接和软链接。但是HDFS架构并不妨碍实现这些特性。

Namenode负责维护文件系统的空间，任何对文件系统空间或属性的修改都将被Namenode记录下来。应用程序可以设置HDFS保存的文件的副本数目。文件副本的数目称为文件的副本系数，这个信息也是由Namenode保存的。

## 数据复制

HDFS被设计成能够在一个大集群中跨机器可靠地存储超大文件。它将每个文件存储成一系列的数据块，除了最后一个，所有的数据块都是同样大小的。为了容错，文件的所有数据块都会有副本。每个文件的数据块大小和副本系数都是可配置的。应用程序可以指定某个文件的副本数目。副本系数可以在文件创建的时候指定，也可以在之后改变。HDFS中的文件都是一次性写入的，并且严格要求在任何时候只能有一个写入者。

Namenode全权管理数据块的复制，它周期性地从集群中的每个Datanode接收心跳信号和块状态报告(Blockreport)。接收到心跳信号意味着该Datanode节点工作正常。块状态报告包含了一个该Datanode上所有数据块的列表。

参考链接：[http://hadoop.apache.org/docs/r1.0.4/cn/hdfs\\_design.html](http://hadoop.apache.org/docs/r1.0.4/cn/hdfs_design.html)

## HDFS Shell

HDFS Shell提供了HDFS最基础的操作功能，包括查看目录，创建和删除文件，移动和复制文件等等。

# 实验四：HDFS Shell入门

## 创建目录：mkdir

```
Usage: hdfs dfs -mkdir [-p] <paths>
Takes path uri's as argument and creates directories.
Options:
The -p option behavior is much like Unix mkdir -p, creating parent directories along the path.
Example:
hdfs dfs -mkdir /user/hadoop/dir1 /user/hadoop/dir2
hdfs dfs -mkdir hdfs://nn1.example.com/user/hadoop/dir hdfs://nn2.example.com/user/hadoop/dir
Exit Code:
Returns 0 on success and -1 on error.
```

## 上传实验数据到该目录：copyFromLocal

```
Usage: hdfs dfs -copyFromLocal <localsrc> URI
Similar to put command, except that the source is restricted to a local file reference.
Options:
The -f option will overwrite the destination if it already exists.
```

## 查看文件并统计行数

hadoop fs, hadoop dfs, hdfs dfs的区别?

在Hadoop的设计中，hadoop fs

FS relates to a generic file system which can point to any file systems like local, HDFS etc. But dfs is very specific to HDFS.

/opt/cloudera/parcels/CDH-6.1.0-1.cdh6.1.0.p0.770702/lib/hadoop/bin

/opt/cloudera/parcels/CDH-6.1.0-1.cdh6.1.0.p0.770702/lib/hadoop-hdfs/bin/hdfs

手册链接：<https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html>



## 实验五：数据拷贝和分布式拷贝

### cp

```
Usage: hdfs dfs -cp [-f] URI [URI ...] <dest>
Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.
Options:
The -f option will overwrite the destination if it already exists.
Example:
hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2
hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2 /user/hadoop/dir
Exit Code:
Returns 0 on success and -1 on error.
```

### distcp

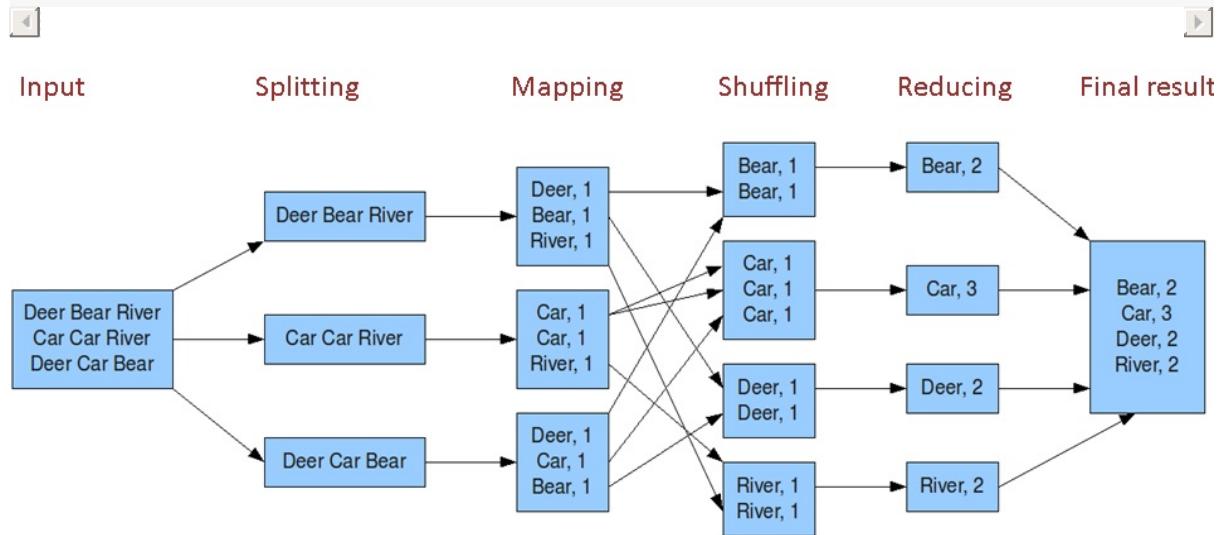
```
https://hadoop.apache.org/docs/current/hadoop-distcp/DistCp.html
```

# 简介

Hadoop Map/Reduce是一个使用简易的软件框架，基于它写出来的应用程序能够运行在由上千个商用机器组成的大型集群上，并以一种可靠容错的方式并行处理上T级别的数据集。

一个Map/Reduce \_作业（job）\_ 通常会把输入的数据集切分为若干独立的数据块，由 \_map任务（task）\_ 以完全并行的方式处理它们。框架会对map的输出先进行排序，然后把结果输入给 \_reduce任务\_。通常作业的输入和输出都会被存储在文件系统中。整个框架负责任务的调度和监控，以及重新执行已经失败的任务。

通常，Map/Reduce框架和分布式文件系统是运行在一组相同的节点上的，也就是说，计算节点和存储节点通常在一起。这种配置允许框架在那些已经存好数据的节点上高效地调度任务，这可以使整个集群的网络带宽被非常高效地利用。



# 实验一：一道真实的大数据面试题

## 实验描述

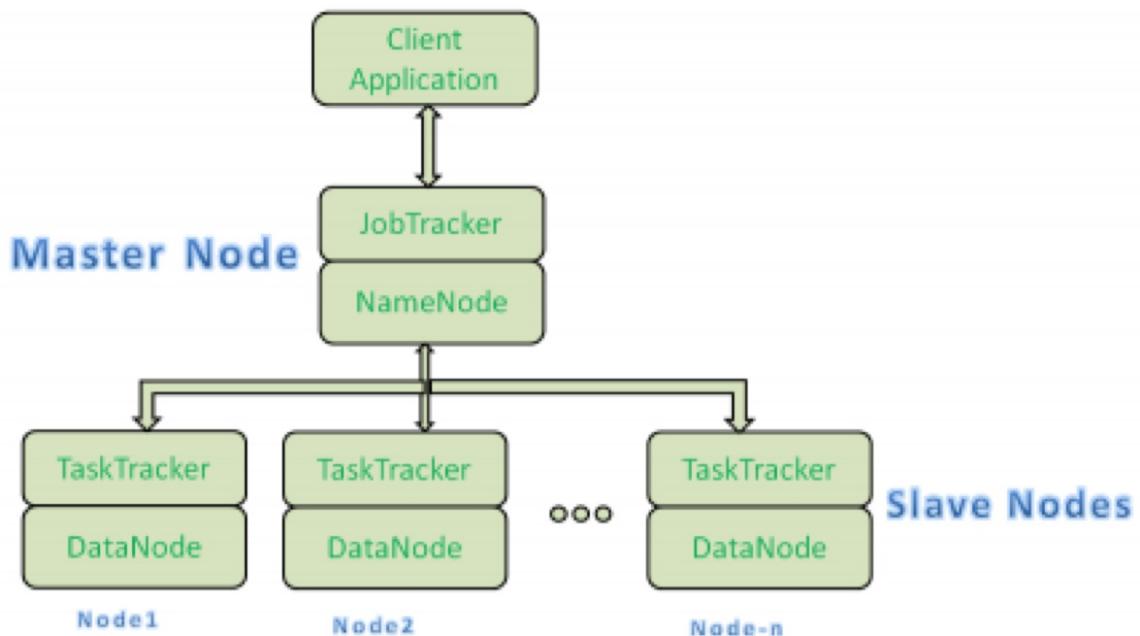
你有1G的文本数据，需要写一个python程序去统计这1G数据中每个单词出现的次数，但是这个程序每个进程最多只允许占用256M内存，请问你如何去统计。

## 实验二：让问题稍微复杂些

### 更苛刻点的要求

假设在实验1中，你通过使用Python多进程去平行计算。但是在计算过程中，如果程序可能被手动Kill掉，那么这个程序如何能自动修复。

# Hadoop 架构设计



## Hadoop 1.x Components Architecture

### JobTracker & TaskTracker

Job Tracker is used to assign MapReduce Tasks to Task Trackers in the Cluster of Nodes. Sometimes, it reassigned same tasks to other Task Trackers as previous Task Trackers are failed or shutdown scenarios.

Job Tracker maintains all the Task Trackers status like Up/running, Failed, Recovered etc.

Task Tracker executes the Tasks which are assigned by Job Tracker and sends the status of those tasks to Job Tracker.

# Hadoop Streaming

## 介绍

Hadoop streaming是Hadoop的一个工具， 它帮助用户创建和运行一类特殊的map/reduce作业， 这些特殊的map/reduce作业是由一些可执行文件或脚本文件充当mapper或者reducer。例如：

```
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar \
-input myInputDirs \
-output myOutputDir \
-mapper /bin/cat \
-reducer /bin/wc
```

## 工作原理

在上面的例子中， mapper和reducer都是可执行文件， 它们从标准输入读入数据（一行一行读）， 并把计算结果发给标准输出。Streaming工具会创建一个Map/Reduce作业， 并把它发送给合适的集群， 同时监视这个作业的整个执行过程。

如果一个可执行文件被用于mapper，则在mapper初始化时， 每一个mapper任务会把这个可执行文件作为一个单独的进程启动。mapper任务运行时， 它把输入切分成行并把每一行提供给可执行文件进程的标准输入。同时， mapper收集可执行文件进程标准输出的内容，并把收到的每一行内容转化成key/value对， 作为mapper的输出。默认情况下， 一行中第一个tab之前的部分作为**key**， 之后的（不包括tab）作为**value**。如果没有tab， 整行作为key值， value值为null。不过， 这可以定制，在下文中将会讨论如何自定义key和value的切分方式。

如果一个可执行文件被用于reducer， 每个reducer任务会把这个可执行文件作为一个单独的进程启动。Reducer任务运行时， 它把输入切分成行并把每一行提供给可执行文件进程的标准输入。同时， reducer收集可执行文件进程标准输出的内容，并把每一行内容转化成key/value对， 作为reducer的输出。默认情况下， 一行中第一个tab之前的部分作为key， 之后的（不包括tab）作为value。在下文中将会讨论如何自定义key和value的切分方式。

这是Map/Reduce框架和streaming mapper/reducer之间的基本通信协议。

用户也可以使用java类作为mapper或者reducer。上面的例子与这里的代码等价：

```
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar \
-input myInputDirs \
-output myOutputDir \
-mapper org.apache.hadoop.mapred.lib.IdentityMapper \
-reducer /bin/wc
```

用户可以设定`stream.non.zero.exit.is.failure true` 或`false` 来表明streaming task的返回值非零时是 Failure 还是 Success。默认情况， streaming task返回非零时表示失败。



# 实验三：统计特征分布

## 实验描述

## 实验代码

### mapper.py

```
#!/usr/bin/env python3

import sys
# id,click,hour,C1,banner_pos,site_id,site_domain,site_category,app_id,app_domain,app_cate
gory,device_id,device_ip,device_model,device_type,device_conn_type,C14,C15,C16,C17,C18,C19
,C20,C21
for line in sys.stdin:
    key = line.strip().split(',')[7]
    value = 1
    print( "%s\t%d" % (key, value) )
```

### reducer.py

```
#!/usr/bin/env python3

import sys

last_key = None
running_total = 0

for input_line in sys.stdin:
    input_line = input_line.strip()
    this_key, value = input_line.split("\t", 1)
    value = int(value)

    if last_key == this_key:
        running_total += value
    else:
        if last_key:
            print( "%s\t%d" % (last_key, running_total) )
        running_total = value
        last_key = this_key

    if last_key == this_key:
        print( "%s\t%d" % (last_key, running_total) )
```

## 单步调试

```
head -n1000 train.txt | ./mapper.py | sort | ./reducer.py
```

## 作业提交

```
hadoop \
jar /opt/cloudera/parcels/CDH-6.1.0-1.cdh6.1.0.p0.770702/lib/hadoop-mapreduce/hadoop-s
treaming.jar \
-mapper "python $PWD/mapper.py" \
-reducer "python $PWD/reducer.py" \
-input XXXXX \
-output XXXX
```

## 实验四：数据倾斜

### 实验描述

统计数据的点击率

点击与否只有1, 0

Hash到两个Part, 严重不平衡

## Key, Value, Partition和Sort

Hadoop Streaming的一些参数设置

## 实验五：二次排序

### 实验描述

统计每个site category下面site domain的分布

## Map Reduce的瓶颈

多轮Shuffle的系统IO

## 简介

## **HQL**

# 实验一：HQL入门

## 创建Hive外部表（External Table）

注：相比Hive内部表，Hive外部表本身不拥有数据，在表被drop掉后数据不会从HDFS上删除

### 实验代码

```
CREATE EXTERNAL TABLE IF NOT EXISTS ctr_data (
    id STRING,
    click INT,
    hour STRING,
    C1 STRING,
    banner_pos STRING,
    site_id STRING,
    site_domain STRING,
    site_category STRING,
    app_id STRING,
    app_domain STRING,
    app_category STRING,
    device_id STRING,
    device_ip STRING,
    device_model STRING,
    device_type STRING,
    device_conn_type STRING,
    C14 STRING,
    C15 STRING,
    C16 STRING,
    C17 STRING,
    C18 STRING,
    C19 STRING,
    C20 STRING,
    C21 STRING,
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/exp/hive/data';
```

### 查看数据

## 实验二：使用HQL进行统计

统计各个 site category 出现的次数

## HQL转换成MapReduce

## 实验三：HQL性能瓶颈

## DAG

## 简介

## 特征工程

## 实验一：构造特征

## 实验二：特征选择

## 模型训练与预测

## 实验三：Logistic Regression

## 模型评估

## 实验四：评估指标的选择

# 可视化

## 实验五：模型可视化

## 简介

## RDD & DataFrame

## 实验一：使用Spark构造模型特征

## MLlib

## Spark SQL

## 简介