[第一章 分布式计算框架与资源调度 6](#_Toc515544216)

[1.1 分布式计算框架 6](#_Toc515544217)

[1.1.1 编程模型 6](#_Toc515544218)

[1.1.2 特殊的组件partitioner与combiner 11](#_Toc515544219)

[1.1.3 用mr进行数据的排序，然后求出topN 14](#_Toc515544220)

[1.1.4 MapReduce 求两个人的共同好友算法 17](#_Toc515544221)

[1.2 分布式资源调度框架 21](#_Toc515544222)

[1.2.1 yarn的概念 21](#_Toc515544223)

[1.2.2 yarn的架构 22](#_Toc515544224)

[1.2.3 yarn的工作流程 22](#_Toc515544225)

[1.2.4 yarn的调度器 Scheduler 24](#_Toc515544226)

[1.3 分布式文件存储系统 26](#_Toc515544227)

[1.3.1 架构 26](#_Toc515544228)

[1.3.2 原理 27](#_Toc515544229)

[1.3.3 API 30](#_Toc515544230)

[1.4 项目 45](#_Toc515544231)

[1.4.1 点击流日志模型 45](#_Toc515544232)

[1.4.2 推荐系统项目 54](#_Toc515544233)

[1.5 相关面试题 57](#_Toc515544234)

[第2章 数据分析及其步骤 59](#_Toc515544235)

[2.1 数据分析定义 59](#_Toc515544236)

[2.2 数据分析的作用 59](#_Toc515544237)

[2.2.1 现状分析 59](#_Toc515544238)

[2.2.2 原因分析 60](#_Toc515544239)

[2.2.3 预测分析 60](#_Toc515544240)

[2.3 数据分析基本步骤（重点） 60](#_Toc515544241)

[2.3.1 明确分析目的和思路 61](#_Toc515544242)

[2.3.2 数据收集 65](#_Toc515544243)

[2.3.3 数据处理 66](#_Toc515544244)

[2.3.4 数据分析 66](#_Toc515544245)

[2.3.5 数据展现（数据可视化） 67](#_Toc515544246)

[2.3.6 报告撰写 67](#_Toc515544247)

[2.4 行业前景 68](#_Toc515544248)

[2.4.1 蓬勃发展的趋势 68](#_Toc515544249)

[2.4.2 数据分析师的职业要求 69](#_Toc515544250)

[2.5 大数据时代 70](#_Toc515544251)

[2.5.1 大数据的含义 70](#_Toc515544252)

[2.5.2 产生背景 70](#_Toc515544253)

[2.5.3 影响 71](#_Toc515544254)

[2.5.4 特征 73](#_Toc515544255)

[2.5.5 思维变革 74](#_Toc515544256)

[2.5.6 第三个思维变革： 74](#_Toc515544257)

[不是所有的事情都必须知道现象背后的原因，而是要让数据自己“发声”，即不是因果关系，而是相关关系。 74](#_Toc515544258)

[2.5.7 科技发展带来的挑战 74](#_Toc515544259)

[2.5.8 分布式系统 75](#_Toc515544260)

[2.6 面试题: 76](#_Toc515544261)

[2.6.1 分布式和集群有啥区别呢？ 76](#_Toc515544262)

[2.6.2 集群 负载均衡 分布式 有什么区别？ 77](#_Toc515544263)

[2.6.3 Web流量日志数据自定义采集 78](#_Toc515544264)

[2.7 Hive 87](#_Toc515544265)

[2.7.1 Hive的基本简介 87](#_Toc515544266)

[2.7.2 Hive数仓开发的基本流程。 88](#_Toc515544267)

[2.7.3 Hive sql 知识点。 90](#_Toc515544268)

[2.8 Hive面试题 94](#_Toc515544269)

[2.8.1 Hive常见 94](#_Toc515544270)

[2.8.2 海量数据处理方法 152](#_Toc515544271)

[2.9 Hadoop HA 165](#_Toc515544272)

[2.9.1 知识点 165](#_Toc515544273)

[2.9.2 上课用图 174](#_Toc515544274)

[2.9.3 3、 面试题 175](#_Toc515544275)

[2.10 Hadoop的联邦机制 176](#_Toc515544276)

[2.10.1 为什么会出现联邦？ 176](#_Toc515544277)

[2.10.2 联邦的实现 176](#_Toc515544278)

[2.10.3 主要优点： 177](#_Toc515544279)

[2.10.4 配置： 178](#_Toc515544280)

[2.10.5 操作 179](#_Toc515544281)

[2.11 项目 180](#_Toc515544282)

[2.11.1 点击流日志模型 180](#_Toc515544283)

[2.11.2 相关面试题 189](#_Toc515544284)

[2.11.3 推荐系统项目 189](#_Toc515544285)

# 数据分析及其步骤

## 数据分析定义

数据分析离不开数据，计量和记录一起促成了数据的诞生。

数据分析是指用适当的统计分析方法对收集来的数据进行分析，将它们加以汇总和理解并消化，以求最大化地开发数据的功能，发挥数据的作用。

数据分析的目的是把隐藏在一大批看似杂乱无章的数据背后的信息集中和提炼出来，总结出所研究对象的内在规律。

数据分析可划分为： 描述性数据分析、 探索性数据分析 、 验证性数据分析。

描述性数据分析属于初级数据分析，另两个属于高级数据分析。其中探索性分析.侧重于在数据之中发现新的特征，而验证性数据分析则侧重于验证已有假设的真伪证明。我们日常学习和工作中所涉及的数据分析主要是描述性数据分析。

## 数据分析的作用

在商业领域中，数据分析的目的是把隐藏在数据背后的信息集中和提炼出来，总结出所研究对象的内在规律，帮助管理者进行有效的判断和决策。

数据分析在企业日常经营分析中主要有三大作用：

### 现状分析

简单来说就是告诉你当前的状况。具体体现在:

第一，告诉你企业现阶段的整体运营情况，通过各个指标的完成情况来衡量企业的运营状态，以说明企业整天运营是好了还是坏了，好的程度如何，坏的程度又到哪里。

第二，告诉你企业各项业务的构成，让你了解企业各项业务的发展以及变动情况，对企业运营状况有更深入的了解。

### 原因分析

简单来说就是告诉你某一现状为什么发生。

经过现状分析，我们对企业的运营情况有了基本了解，但不知道运营情况具体好在哪里，差在哪里，是什么原因引起的。这时就需要开展原因分析，以进一步确定收入下降的具体原因，对运营策略做出调整与优化。

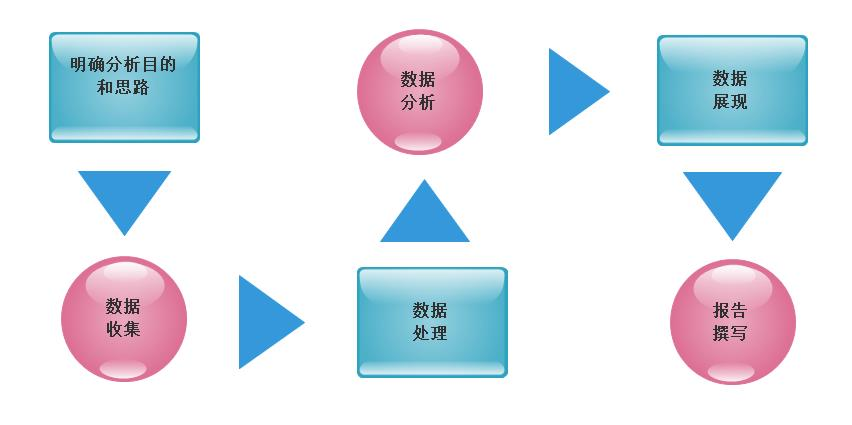
### 预测分析

简单来说就是告诉你将来会发生什么。

在了解企业运营现状后，有时还需要对企业未来发展趋势做出预测，为制订企业运营目标及策略提供有效的参考与决策依据，以保证企业的可持续健康发展。预测分析一般通过专题分析来完成，通常在制订企业季度、年度等计划时进行，其开展的频率没有现状分析及原因分析高。

## 数据分析基本步骤（重点）

典型的数据分析包含以下几个步骤：



### 明确分析目的和思路

确保数据分析过程有效进行的先决条件，为数据的收集、处理及分析提供清晰的指引方向。

目的是整个分析流程的起点。目的不明确则会导致方向性的错误。

思考:为什么要开展数据分析，通过这次数据分析要解决什么问题？

当明确目的后，就要校理分析思路，并搭建分析框架，把分析目的分解成若干个不同的分析要点，即如何具体开展数据分析。需要从哪几个角度进行分析，采用哪些分析指标。

只有明确了分析目的，分析框架才能跟着确定下来，最后还要确保分析框架的体系化，使分析更具有说服力。

体系化也就是逻辑化，简单来说就是先分析什么，后分析什么，使得各个分析点之间具有逻辑联系。避免不知从哪方面入手以及分析的内容和指标被质疑是否合理、完整。所以体系化就是为了让你的分析框架具有说服力。要想使分析框架体系化，就需要一些营销、管理等理论为指导，结合着实际的业务情况进行构建，这样才能保证分析维度的完整性，分析结果的有效性以及正确性。

比如以用户行为 理论为指导，搭建的互联网网站分析指标框架如下：



把跟数据分析相关的营销、管理等理论统称为数据分析方法论。比如用户行为理论、PEST 分析法、5W2H 分析法等等

**扩展**

**1）数据分析方法论与数据分析区别**

数据分析方法论主要用来指导数据分析师进行一次完整的数据分析，它更多的是指数据分析思路。数据分析方法论主要从宏观角度指导如何进行数据分析，它就像是一个数据分析的前期规划，指导着后期数据分析工作的开展。

而数据分析法则是指具体的分析方法。数据分析法主要从微观角度指导如何进行数据分析。

**2）常用的数据分析方法论**

**PEST 分析法**

PEST分析法用于对宏观环境的分析。宏观环境又称一般环境，是指影响一切行业和企业的各种宏观力量。对宏观环境因素作分析时。由于不同行业和企业有其自身特点和经营需要，分析的具体内容会有差异，但一般都应对 政治(Political )、经济（Economic）、 技术（Technological）和 社会（Social）这四大类影响企业的主要外部环境因素进行分析，这种方法简称为 PEST 分析法。

**5W2H 分析法**

5w2H 分析法是以五个 w 开头的英语单词和两个 H 开头的英语单词进行提问，从回答中发现解决问题的线索，即 何因 （Why） ） 、何事(What) 、何人(Who) 、何时(When) 、何地(Where ) 、如何做(How) 、何价(How much)，这就构成了 5W2H分析法的总框架。

**逻辑树分析法**

逻辑树又称问题树、演绎树或分解树等。它是将问题的所有了问题分层罗列，从最高层开始，并逐步向下扩展。

把一个已知问题当成树干，然后开始考虑这个问题和哪些相关问题有关。每想到一点，就给这个问题所在的树干加一个“树枝”，并标明这个“树枝”代表什么问题。

逻辑树的使用必须遵循以下三个原则。

要素化：把相同的问题总结归纳成要素。

框架化：将各个要素组织成框架。遵守不重不漏的原则。

关联化：框架内的各要素保持必要的相互关系，简单而不独立。

**4P 营销理论**

营销组合实际上有几十个要素，这些要素可以概括为 4 类: 产品(Product) 、价格(Price) 、 渠道(Place ) 、促销(Promotion )。

**用户行为理论**

网站分析的发展已经较为成熟，有一套成熟的分析指标。比如 IP、PV、页面停留时间、跳出率、回访者、新访问者、回访次数、回访相隔天数、流失率、关键字搜索、转化率、登录率，等等。需要我们梳理指标之间的逻辑关系。

用户使用行为是指用户为获取、使用物品或服务所采取的各种行动，用户对产品首先需要有一个认知、熟悉的过程，然后试用，再决定是否继续消费使用，最后成为忠诚用户。

可利用用户使用行为理论，梳理网站分析的各关键指标之间的逻辑关系，构建符合公司实际业务的网站分析指标体系。

### 数据收集

数据收集是按照确定的数据分析框架，收集相关数据的过程，它为数据分析提供了素材和依据。

包括第一手数据与第二手数据，第一手数据主要指可直接获取的数据，第二手数据主要指经过加工整理后得到的数据。

一般数据来源主要有以下几种方式：

数据库：每个公司都有自己的业务数据库，存放从公司成立以来产生的相关业务数据。这个业务数据库就是一个庞大的数据资源，需要有效地利用起来。

公开出版物：可以用于收集数据的公开出版物包括《中国统计年鉴》《中国社会统计年鉴》《中国人口统计年鉴》《世界经济年鉴》《世界发展报告》等统计年鉴或报告。

互联网：随着互联网的发展，网络上发布的数据越来越多，特别是搜索引擎可以帮助我们快速找到所需要的数据，例如国家及地方统计局网站、行业组织网站、政府机构网站、传播媒体网站、大型综合门户网站等上面都可能有我们需要的数据。

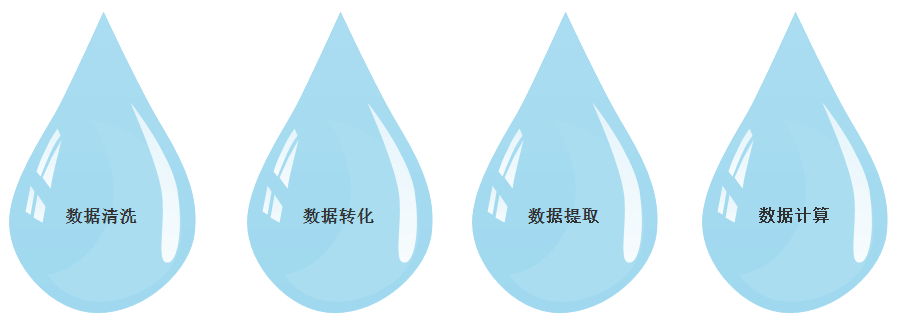
市场调查：市场调查就是指运用科学的方法，有目的、有系统地收集、记录、整理有关市场营销的信息和资料，分析市场情况，了解市场现状及其发展趋势，为市场预测和营销决策提供客观、正确的数据资料。

市场调查可以弥补其他数据收集方式的不足（比如：需要了解用户的想法与需求），但进行市场调查所需的费用较高，而且会存在一定的误差，故仅作参考之用。

### 数据处理

数据处理是指对收集到的数据进行加工整理，形成适合数据分析的样式，它是数据分析前必不可少的阶段。

基本目的:从大量的、杂乱无章、难以理解的数据中，抽取并推导出对解决问题有价值、有意义的数据。



数据处理主要包括数据清洗、数据转化、数据提取、数据计算等处理方法。

一般拿到手的数据都需要进行一定的处理才能用于后续的数据分析工作，即使再“干净”’的原始数据也需要先进行一定的处理才能使用。数据处理是数据分析的基础。通过数据处理，将收集到的原始数据转换为可以分析的形式，并且保证数据的一致性和有效性。

### 数据分析

数据分析是指用适当的分析方法及工具，对处理过的数据进行分析，提取有价值的信息，形成有效结论的过程。

由于数据分析多是通过软件来完成的，这就要求数据分析师不仅要掌握各种数据分析方法，还要熟悉数据分析软件的操作。

数据挖掘其实是一种高级的数据分析方法，就是从大量的数据中挖掘出有用的信息，它是根据用户的特定要求，从浩如烟海的数据中找出所需的信息，以满足用户的特定需求。数据挖掘技术是人们长期对数据库技术进行研究和开发的结果。一般来说，数据挖掘侧重解决四类数据分析问题***:分类、聚类、关联和预测，重点在寻找模式和规律***。数据分析与数据挖掘的本质是一样的，都是从数据里面发现关于业务的知识。

### 数据展现（数据可视化）

一般情况下，数据是通过表格和图形的方式来呈现的，我们常说用图表说话就是这个意思。常用的数据图表包括饼图、柱形图、条形图、折线图、散点图、雷达图等，当然可以对这些图表进一步整理加工，使之变为我们所需要的图形，例如金字塔图、矩阵图、漏斗图等。大多数情况下，人们更愿意接受图形这种数据展现方式，因为它能更加有效、直观地传递出分析所要表达的观点。

一般情况下能用图说明问题的就不用表格，能用表格说明问题的就不要用文字。

### 报告撰写

数据分析报告其实是对整个数据分析过程的一个总结与呈现。通过报告，把数据分析的起因、过程、结果及建议完整地呈现出来，供决策者参考。

一份好的数据分析报告，首先需要有一个好的分析框架，并且图文并茂，层次明晰，能够让阅读者一目了然。

另外，数据分析报告需要有明确的结论，没有明确结论的分析称不上分析，同时也失去了报告的意义，因为我们最初就是为寻找或者求证一个结论才进行分析的，所以千万不要舍本求末。

最后，好的分析报告一定要有建议或解决方案。作为决策者，需要的不仅仅是找出问题，更重要的是建议或解决方案，以便他们做决策时作参考。所以，数据分析师不仅需要掌握数据分析方法，而且还要了解和熟悉业务，这样才能根据发现的业务问题，提出具有可行性的建议或解决方案。

## 行业前景

### 蓬勃发展的趋势

**市场巨大**

许多企业都在讨论大数据分析，也有实际的需求并愿意为此付钱，但是比较零碎尚不系统化。

目前对数据需求最强烈的行业依次是：金融机构，以广告投放及电商为代表的互联网企业等；

**尚没出现平台级公司的模式**

（这或许往往是大市场或者大机会出现之前的混沌期）

**企业技术外包的氛围在国内尚没完全形成**

对于一些有能力的技术公司，如果数据需求强烈的话，考虑到自身能力的健全以及数据安全性，往往不会外包或者采用外部模块，而倾向于自建这块业务；

**整个行业很大而且需求旺盛**

未来 BAT 及京东、58 和滴滴打车等企业，凭借其自身产生的海量数据,必然是数据领域的大玩家。即使没有留给创业公司出现平台级巨型企业的机会，也将留出各种各样的细分市场机会让大家可以获得自己的领地。

### 数据分析师的职业要求

懂业务：从事数据分析工作的前提就是需要懂业务，即熟悉行业知识、公司业务及流程，最好有自己独特见解，若脱离行业认知和公司业务背景，分析的结果只会是脱了线的风筝，没有太大的实用价值。

懂管理：一方面是搭建数据分析框架的要求。另一方面的作用是针对数据分析结论提出有指导意义的分析建议，如果没有管理理论的支撑，就难以确保分析建议的有效性。

懂分析：是指掌握数据分析的基本原理与一些有效的数据分析方法，并能灵活运用到实践工作中，以便有效地开展数据分析。

懂工具：是指掌握数据分析相关的常用工具。

数据分析工具就是实现数据分析方法理论的工具，面对越来越庞大的数据，依靠计算器进行分析是不现实的，必须利用强大的数据分析工具完成数据分析工作。

懂设计：是指运用图表有效表达数据分析师的分析观点，使分析结果一目了然。

**面试题**

1. 数据清洗后最后会保留哪些数据?
2. 数据采集过来怎么进行处理的？
3. 数据清洗的规则.
4. 需求分析是什么, 需求文档是什么.
5. 关于集群搭建的大概思路

6.1.4分布式、集群

分布式（distributed）是指在多台不同的服务器中部署不同的服务模块，通过远程调用协同工作，对外提供服务。

集群（cluster）是指在多台不同的服务器中部署相同应用或服务模块，构成一个集群，通过负载均衡设备对外提供服务。

## 面试题:

### 分布式和集群有啥区别呢？

集群和分布式都是由多个节点组成，但是集群之间的通信协调基本不需要，而分布式各个节点的通信协调必不可少。

集群主要是为了应对请求压力的分担，从而有了LB，负载均衡集群；为了应对可用性，从而有了HA，高可用性集群；为了更强的性能，从而有了HP，高性能集群；为了高并发大规模性能，从而有分布式系统集群。

### 集群 负载均衡 分布式 有什么区别？

服务器集群：  
 服务器集群就是指将很多服务器集中起来一起进行同一种服务，在客户端看来就像是只有一个服务器。集群可以利用多个计算机进行并行计算从而获得很高的计算速度，也可以用多个计算机做备份，从而使得任何一个机器坏了整个系统还是能正常运行。  
服务器负载均衡：  
 负载均衡 （Load Balancing） 建立在现有网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。  
[分布式服务器](https://www.baidu.com/s?wd=%E5%88%86%E5%B8%83%E5%BC%8F%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)：  
 所谓分布式资源共享服务器就是指数据和程序可以不位于一个服务器上，而是分散到多个服务器，以网络上分散分布的[地理信息](https://www.baidu.com/s?wd=%E5%9C%B0%E7%90%86%E4%BF%A1%E6%81%AF&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)数据及受其影响的数据库操作为研究对象的一种理论计算模型服务器形式。分布式有利于任务在整个计算机系统上进行分配与优化，克服了传统集中式系统会导致中心主机资源紧张与响应瓶颈的缺陷，解决了网络GIS 中存在的数据异构、数据共享、运算复杂等问题，是[地理信息](https://www.baidu.com/s?wd=%E5%9C%B0%E7%90%86%E4%BF%A1%E6%81%AF&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)系统技术的一大进步。  
这个三种架构都是常见的服务器架构，集群的主要是IT公司在做，可以保障重要数据安全；负载均衡主要是为了分担[访问量](https://www.baidu.com/s?wd=%E8%AE%BF%E9%97%AE%E9%87%8F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)，避免临时的网络堵塞，主要用于[电子商务类型](https://www.baidu.com/s?wd=%E7%94%B5%E5%AD%90%E5%95%86%E5%8A%A1%E7%B1%BB%E5%9E%8B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)的网站；[分布式服务器](https://www.baidu.com/s?wd=%E5%88%86%E5%B8%83%E5%BC%8F%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)主要是解决跨区域，多个单个节点达到高速访问的目前，一般是类似CDN的用途的话，会采用[分布式服务器](https://www.baidu.com/s?wd=%E5%88%86%E5%B8%83%E5%BC%8F%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)。

### Web流量日志数据自定义采集

#### 1 网站流量日志数据分析系统

**数据分析**：用适当的统计分析方法对收集来的数据进行分析，将这些数据加以汇总和理解消化，追求最大化开发数据的功能，发挥数据的作用。

本小节主要介绍点击流日志采集。

**网站流量日志数据分析系统**主要**目的**：帮助网站管理员、运营员、推广员等实时获取网站流量信息，从而提高网站流量，提升网站用户体验，让更多的访客转沉淀下来称为会员或客户，以更少的投入获取最大化的收入

网站流量日志数分析系统图如图1所示。

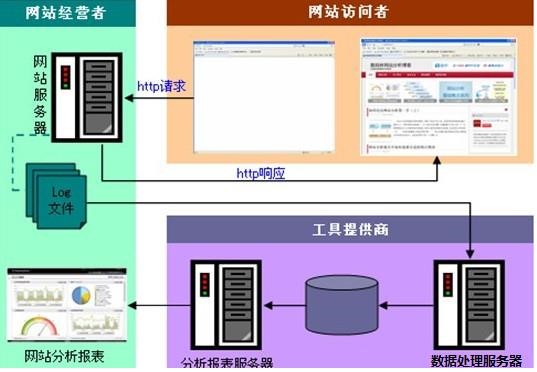
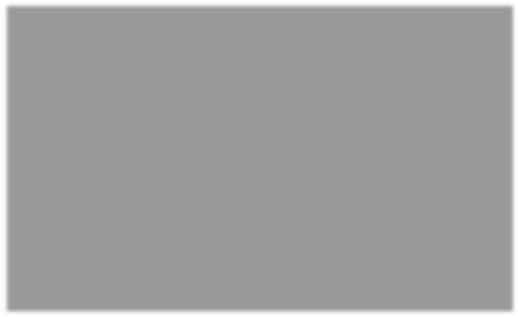


图1 网站流量日志数分析系统

**Web访问日志**：用户访问网站时所有访问、浏览、点击行为数据。比如点击了哪一个链接，打开了哪一个页面，采用了哪个搜索项、总体会话时间等。而所有这些信息都可通过网站日志保存下来。通过分析这些数据，可以获知许多对网站运营至关重要的信息。采集的数据越全面，分析就能越精准。

**日志生成渠道**主要包括以下两种：

一是：web 服务器软件（httpd、nginx、tomcat）自带的日志记录功能，如 Nginx的 access.log 日志；

二是：自定义采集用户行为数据，通过在页面嵌入自定义的 javascript 代码来获取用户的访问行为（比如鼠标悬停的位置，点击的页面组件等），然后通过 ajax请求到后台记录日志，这种方式所能采集的信息会更加全面。

**Tips:**

tomcat服务器的资源存放在webapp目录下；

nginx服务器资源存放在 html目录下；

https服务器的资源存放在 /var/www/html/目录下。

在实际操作中，有以下几个方面的数据**可以自定义的采集**：

* 系统特征：比如所采用的操作系统、浏览器、域名和访问速度等。
* 访问特征：包括停留时间、点击的 URL、所点击的“页面标签<a>”及标签的属性等。
* 来源特征：包括来访 URL，来访 IP 等。
* 产品特征：包括所访问的产品编号、产品类别、产品颜色、产品价格、产品利润、产品数量和特价等级等。

#### 2网站流量日志数据自定义采集

用户的行为会触发浏览器对被统计页面的一个 http 请求，比如打开某网页。当网页被打开，页面中的埋点 javascript 代码会被执行。

**埋点**：在网页中预先加入小段 javascript 代码，这个代码片段一般会动态创建一个 script 标签，并将 src 属性指向一个单独的js文件，此时这个单独的js文件（图中绿色节点）会被浏览器请求到并执行，这个js往往就是真正的数据收集脚本。网站流量日志数据自定义采集分析图如图2所示。

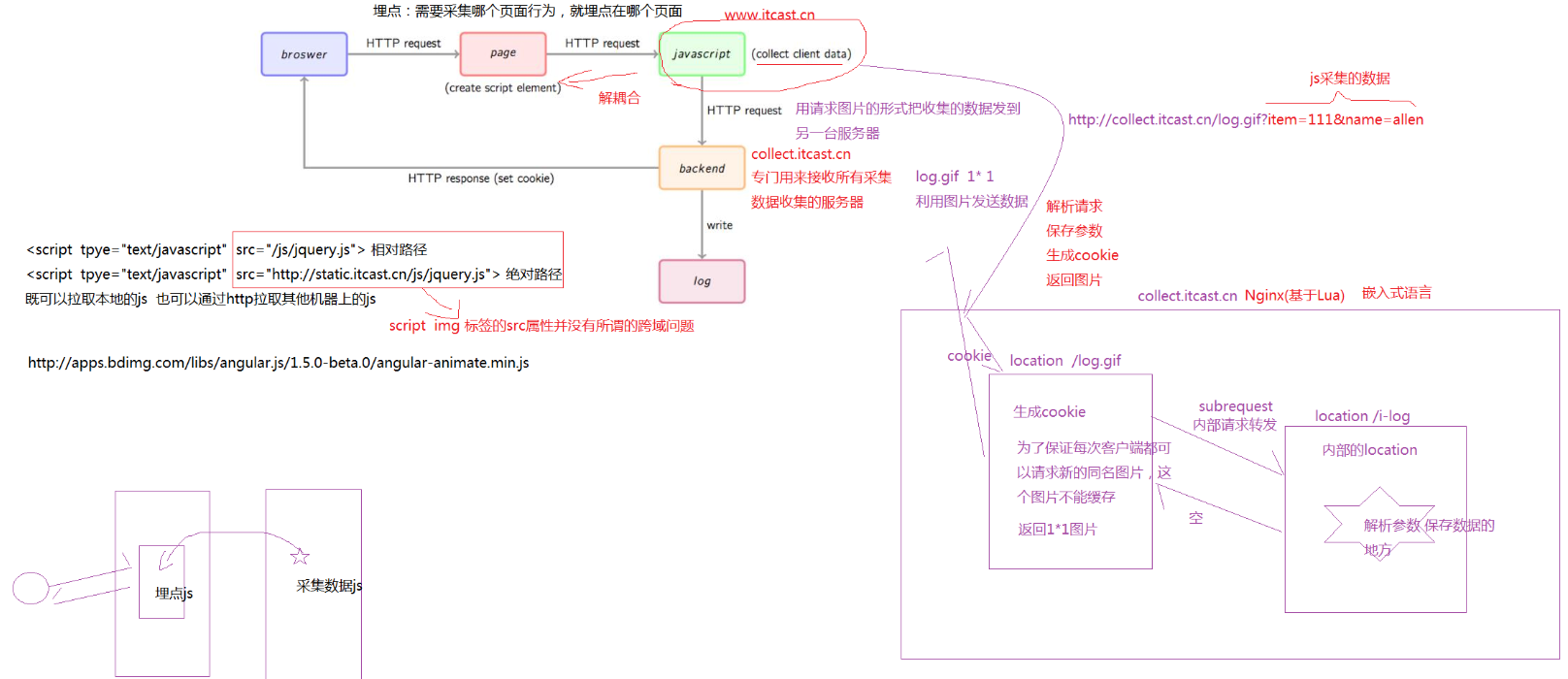


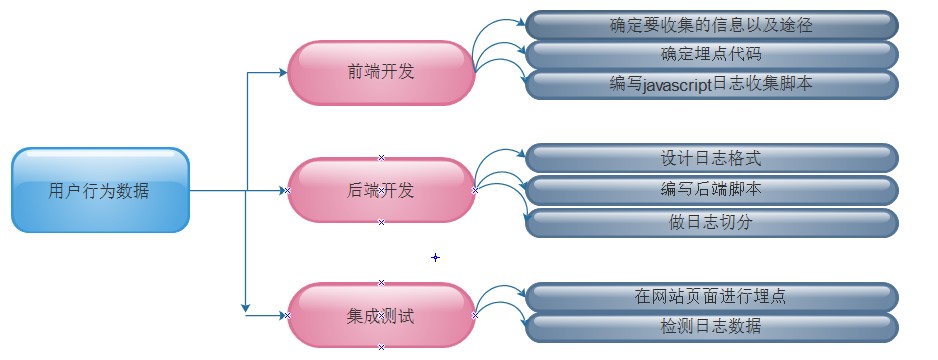
图2 网站流量日志数据自定义采集分析图

数据收集完成后js会请求一个后端的数据收集脚本（图中的 backend），

这个脚本一般是一个伪装成图片的动态脚本程序，js 会将收集到的数据通过http 参数的方式传递给后端脚本，后端脚本解析参数并按固定格式记录到访问日志，同时可能会在 http 响应中给客户端种植一些用于追踪的 cookie。

##### 设计实现

根据原理分析并结合 Google Analytics，想搭建一个自定义日志数据采集系统，要做以下几件事：



1. 确定收集信息

例如访问时间、ip、域名、URL、页面标题、浏览客户端、客户端语言、方可标识、状态码、发送内容量等信息

1. 确定埋点代码

在需要进行数据采集的关键点植入统计代码

js自调用匿名函数

格式： (function(){})();

第一对括号向脚本返回未命名的函数；后一对空括号立即执行返回的未命名函数，括号内为匿名函数的参数。

自调用匿名函数的好处是，避免重名，自调用匿名函数只会在运行时执行一次，一般用于初始化。

1. 前端收集脚本

数据收集脚本被请求后会被执行，一般需要做以下几件事：

1. 通过浏览器内置javascript对象收集信息，如页面title（通过document.title）、referrer（上一跳 url，通过 document.referrer）、用户显示器分辨率（通过windows.screen）、 cookie 信息（通过 document.cookie）等等一些信息。
2. 收集配置信息
3. 将上面两步收集的数据按预定义格式解析并拼接（get 请求参数）
4. 请求一个后端脚本，将信息放在 http request 参数中携带给后端脚本
5. 后端脚本

log.gif 是后端脚本，是一个伪装成 gif 图片的脚本。后端脚本一般需要完

成以下几件事情：

1. 解析 http 请求参数得到信息。
2. 从 Web 服务器中获取一些客户端无法获取的信息，如访客 ip 等。
3. 将信息按格式写入 log。
4. 生成一副 1×1 的空 gif 图片作为响应内容并将响应头的 Content-type设为 image/gif。
5. 在响应头中通过 Set-cookie 设置一些需要的 cookie 信息。 之所以要设置 cookie 是因为如果要跟踪唯一访客，通常做法是如果在请求时发现客户端没有指定的跟踪 cookie，则根据规则生成一个全局唯一的 cookie 并种植给用户，否则 Set-cookie 中放置获取到的跟踪 cookie 以保持同一用户 cookie不变。这种做法虽然不是完美的（例如用户清掉 cookie 或更换浏览器会被认为是两个用户），但是目前被广泛使用的手段。

我们使用 nginx 的 access\_log 做日志收集，不过有个问题就是 nginx 配置本身的逻辑表达能力有限，所以选用 OpenResty 做这个事情。

OpenResty是一个基于 Nginx 扩展出的高性能应用开发平台，内部集成了诸多有用的模块，其中的核心是通过 ngx\_lua 模块集成了 Lua，从而在 nginx 配置文件中可以通过 Lua 来表述业务。

Lua 是一种轻量小巧的脚本语言，用标准 C 语言编写并以源代码形式开放，其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。需要在 nginx 的配置文件中定义日志格式。

1. 日志格式

日志格式主要考虑日志分隔符，一般会有以下几种选择：

固定数量的字符、制表符分隔符、空格分隔符、其他一个或多个字符、特定的开始和结束文本。

1. 日志切分

日志收集系统访问日志时间一长文件变得很大，而且日志放在一个文件不便于管理。通常要按时间段将日志切分，例如每天或每小时切分一个日志。

**Tips**：

1. 如果埋点js涉及频繁更新，则需要让埋点js代码与页面解耦合，让更新采集数据的js不影响页面。
2. 为了高效地采集数据，同时不影响web服务器的运行压力，则需要把后端收集数据的服务单独部署。
3. ajax 是不能跨域请求的。一种通用的方法是 js 脚本创建一个 Image 对象，将 Image对象的 src 属性指向后端脚本并携带参数，此时即实现了跨域请求后端。这也是后端脚本为什么通常伪装成 gif 文件的原因。

**跨域问题的其他解决方案：**

<https://blog.csdn.net/qq_31617637/article/details/72955239>

##### 环境部署

本次搭建两台服务器，一台用于正常的web服务器，一台用于收集数据，需要在该台服务器部署基于lua的nginx。

1. **服务器中安装依赖**

yum -y install gcc perl pcre-devel openssl openssl-devel

1. **上传LuaJIT-2.0.4.tar.gz并安装LuaJIT**

tar -zxvf LuaJIT-2.0.4.tar.gz -C /usr/local/src/

cd /usr/local/src/LuaJIT-2.0.4/

make && make install PREFIX=/usr/local/luajit

1. **设置LuaJIT环境变量**

vi /etc/profile 添加如下内容：

export LUAJIT\_LIB=/usr/local/luajit/lib

export LUAJIT\_INC=/usr/local/luajit/include/luajit-2.0

source /etc/profile

1. **创建modules保存nginx的模块**

mkdir -p /usr/local/nginx/modules

1. **上传依赖的模块**

* set-misc-nginx-module-0.29.tar.gz
* lua-nginx-module-0.10.0.tar.gz
* ngx\_devel\_kit-0.2.19.tar.gz
* echo-nginx-module-0.58.tar.gz

1. **将依赖的模块直接解压到/usr/local/nginx/modules目录**

tar -zxvf lua-nginx-module-0.10.0.tar.gz -C /usr/local/nginx/modules/

tar -zxvf set-misc-nginx-module-0.29.tar.gz -C /usr/local/nginx/modules/

tar -zxvf ngx\_devel\_kit-0.2.19.tar.gz -C /usr/local/nginx/modules/

tar -zxvf echo-nginx-module-0.58.tar.gz -C /usr/local/nginx/modules/

1. **安装openresty**

下载对应的安装包：openresty-1.9.7.3.tar.gz

解压：tar -zxvf openresty-1.9.7.3.tar.gz -C /usr/local/src/

编译安装openresty:

cd /usr/local/src/openresty-1.9.7.3/

执行命令：

./configure --prefix=/usr/local/openresty --with-luajit && make && make install

1. **安装nginx**

下载对应的安装包：nginx-1.8.1.tar.gz

解压：tar -zxvf nginx-1.8.1.tar.gz -C /usr/local/src/

1. **编译nginx并支持其他模块**

进入到nginx的安装目录 cd /usr/local/src/nginx-1.8.1/

执行如下命令：

./configure --prefix=/usr/local/nginx \

--with-ld-opt="-Wl,-rpath,/usr/local/luajit/lib" \

--add-module=/usr/local/nginx/modules/ngx\_devel\_kit-0.2.19 \

--add-module=/usr/local/nginx/modules/lua-nginx-module-0.10.0 \

--add-module=/usr/local/nginx/modules/set-misc-nginx-module-0.29 \

--add-module=/usr/local/nginx/modules/echo-nginx-module-0.58

make -j2 && make install

##### 自定义采集数据实现

方案一：基本功能实现

1. 创建页面index.html，添加埋点代码，放入nginx默认目录nginx/html下。
2. 在默认目录nginx/html下添加一个数据采集脚本ma.js。
3. 修改nginx的配置文件，添加自定义相关业务逻辑。
4. 启动nginx

sbin/nginx -c conf/nginx.conf

1. 通过游览器访问nginx
2. 观察自定义日志采集文件是否有对应的内容输出

tail -f logs/user\_defined.log

此时还可以观察nginx默认的输出日志文件

tail -f logs/access.log

停止nginx:

sbin/nginx –s stop

**问题：解决nginx在记录post数据时，中文字符转成16进制的问题**

**解决方法:**

/usr/local/src/nginx-1.8.1/src/http/modules/ ngx\_http\_log\_module.c

修改源码如下图所示



然后重新编译，安装nginx

./configure --prefix=/usr/local/nginx \

--with-ld-opt="-Wl,-rpath,/usr/local/luajit/lib" \

--add-module=/usr/local/nginx/modules/ngx\_devel\_kit-0.2.19 \

--add-module=/usr/local/nginx/modules/lua-nginx-module-0.10.0 \

--add-module=/usr/local/nginx/modules/set-misc-nginx-module-0.29 \

--add-module=/usr/local/nginx/modules/echo-nginx-module-0.58

make -j2 && make install

## Hive

### Hive的基本简介

#### 1）Hive是什么?

Hive 是基于 Hadoop 的一个数据仓库工具，可以**将结构化的数据**文件**映射**为  
一张数据库表，并提供类 **SQL 查询**功能。**Hive 利用 HDFS 存储数据，利用 MapReduce 查询分析数据。**  
本质是将 **SQL 转换为 MapReduce** 程序,比直接用 MapReduce 开发效率更高。

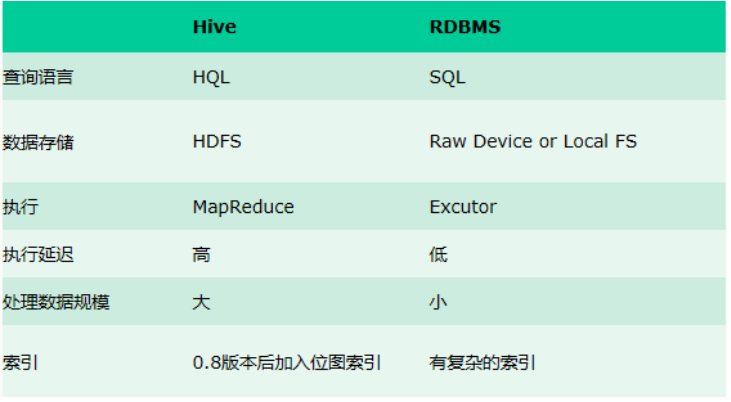
#### 2）Hive的源数据存储?

通常是存储在关系数据库如 mysql/derby 中。 Hive 将元数据存储在数据库中。 Hive 中的**元数据包括表的名字，表的列和分区及其属性，表的属性（是否为外部表等），表的数据所在目录**等。

#### 3）Hive与传统DB的区别?

传统数据库：**OLTP-->面向事务(Transaction) 操作型处理**  就是关系型数据库：mysql oracle sqlserver db2 主要是**支持业务，面向业务**。

Hive：**OLAP-->面向分析（Analytical）分析型处理**  就是数据仓库 面对的是**历史数据**（历史数据中的一部分就来自于数据库） **开展分析**



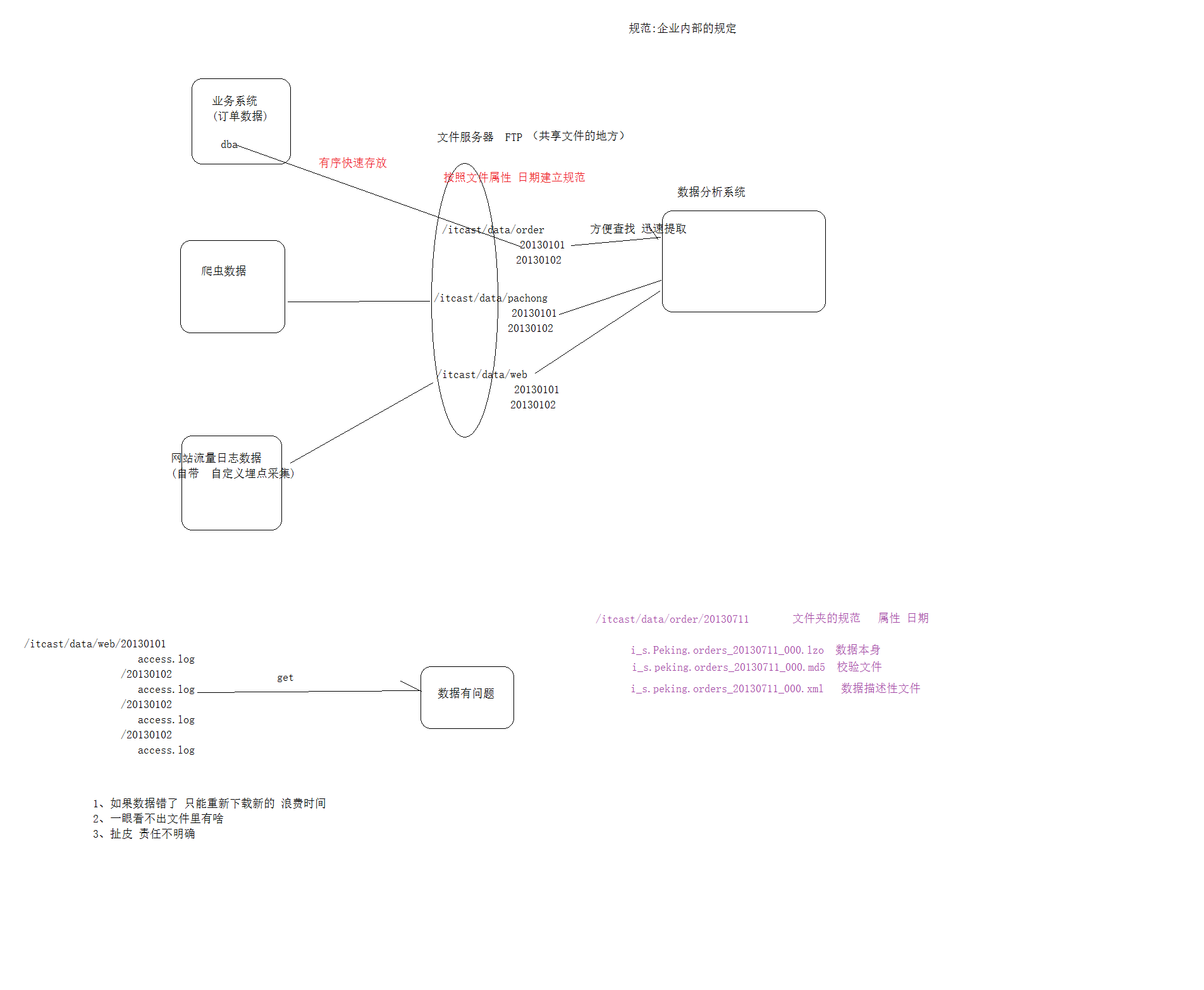
### Hive数仓开发的基本流程。

#### 1）从业务系统获取数据

Sqoop 导入数据库的数据：spoop可以在 Hive与传统的数据库间进行数据的传递，可以将一个关系型数据库中的数据导进到Hadoop的HDFS中，也可以将HDFS的数据导进到关系型数据库中。

Flume 采集文本数据：Flume可以将各类型的文件进行采集，存放入hdfs中。

Ftp 文件服务器：从文件服务器上下载分析所需的源数据（增量数据、全量数据）。如下图所示：



#### 2）数据存储

数据仓库分层 ods层、dw层、da层：

源数据层 ODS :直接引用外围的数据 没有统一格式化的 不会直接应用使用 不利于分析

数据仓库层 DW :来自于ODS 要经过ETL的过程 格式统一 数据规整 干净清洁

数据应用层 DA ：要去用DW层数据 真正的数据使用者。

数据集市：

[数据集市](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E9%9B%86%E5%B8%82)也叫数据市场，数据集市就是满足特定的部门或者用户的需求，按照多维的方式进行存储，包括定义维度、需要计算的指标、维度的层次等，生成面向决策分析需求的数据立方体。

#### 3）写sql开发需求

#### 4）配置调度系统

#### 5）导出数据&展示

### Hive sql 知识点。

#### 1）DML&DDL

DML：**数据操纵语言** ,负责对数据库对象运行数据访问工作的指令集DELETE  FROM  table\_name;

DDL：**数据定义语言**，用语定义和管理数据库中的对象。

TRUNCATE TABLE table\_name；

#### 2）外表和内表

内部表：

create table student(Sno int,Sname string,Sex string,Sage int,Sdept string) row format delimited fields terminated by ',';

未被external修饰、表数据由Hive自身管理、表数据存储的位置是默认的hive.metastore.warehouse.dir 、删表会直接删除元数据及存储数据、对表的修改会将修改直接同步给元数据的是内部表

外部表：

create external table student\_ext(Sno int,Sname string,Sex string,Sage int,Sdept string) row format delimited fields terminated by ',' location '/stu';

被external修饰的、表数据由HDFS管理、表数据的存储位置由自己指定、删表仅仅会删除元数据，HDFS上的文件并不会被删除的是外部表。

#### 3）分区、分桶

分区（重点）

为了避免select查询时候的全表扫描问题 hive提出了分区表（partitioned by）概念 给文件归类 打上标识（标识不能为表中已有的字段）

静态分区:

单分区建表 create table par\_tab (name string,nation string) partitioned by (sex string) row format delimited fields terminated by ',';

加载 load data local inpath '/home/hadoop/files/par\_tab.txt' into table par\_tab partition (sex='man');

在新建分区表的时候，系统会在hive数据仓库默认路径/user/hive/warehouse/下创建一个目录（表名），再创建目录的子目录sex=man（分区名），**最后分区名下存放实际的数据文件。**

多分区建表create table par\_tab\_muilt (name string, nation string) partitioned by (sex string,dt string) row format delimited fields terminated by ',' ;

加载：load data local inpath '/home/hadoop/files/par\_tab.txt' into table par\_tab\_muilt partition (sex='man',dt='2018-05-26');

新建表的时候定义的分区顺序，决定了文件目录顺序（谁是父目录谁是子目录），因为有了这个层级关系，**当我们查询所有man的时候，man以下的所有日期下的数据都会被查出来**。如果只查询日期分区，但父目录sex=man和sex=woman都有该日期的数据，那么Hive会对输入路径进行修剪，**从而只扫描日期分区，性别分区不作过滤（即查询结果包含了所有性别）。**

动态分区

静态分区在插入的时候必须首先要知道有什么分区类型，而且每个分区写一个load data，太烦人。使用动态分区可解决以上问题，其可以根据查询得到的数据动态分配到分区里。其实动态分区与静态分区区别就是不指定分区目录，由系统自己选择。

开启动态分区功能set hive.exec.dynamic.partition=true;

加载数据 insert overwrite table par\_dnm partition(sex='man',dt) select name, nation, dt from par\_tab;

注：

1.动态分区不允许主分区采用动态列而副分区采用静态列，这样将导致所有的主分区都要创建副分区静态列所定义的分区。

2.动态分区可以允许所有的分区列都是动态分区列，但是要首先设置一个参数hive.exec.dynamic.partition.mode 。

set hive.exec.dynamic.partition.mode;

hive.exec.dynamic.partition.mode=strict

它的**默认值是strick，即不允许分区列全部是动态的**，这是为了防止用户有可能原意是只在子分区内进行动态建分区，但是由于疏忽忘记为主分区列指定值了，这将导致一个dml语句在短时间内创建大量的新的分区（对应大量新的文件夹），对系统性能带来影响。修改后即可。

set hive.exec.dynamic.partition.mode=nostrick;

总结：

1、分区表是为了减少查询时候的全表扫描而出现。

2、分区表的现象就是在表的文件夹下多了一个文件夹 而文件夹的名字就是分区 字段=分区值

3、分区字段的值在查询的时候 会显示出来 但是并不代表结构化数据中有这个字段 分区字段是一个虚拟字段 只是用来标识文件

方便用户查询的时候 根据这个标识进行过滤 从而减少了全局扫描

4、分区表的数据通过load data的方式加载 加载的时候要指定分区的值（这个分区的值就是这批数据文件夹名字的值）

5、关于分区字段一定不会是表中存在的字段 如果是直接编译报错 体会什么叫给数据打标识

分桶

1．为了提高join查询时的效率 减少了笛卡尔积的数量，分桶表出现。

clustered by （字段） into num\_buckets buckets（如：clustered by （sex） into 2 buckets）

2．分桶的功能默认不开启 需要自己手动开启

set hive.enforce.bucketing = true;

3．分成几桶 也需要自己指定

set mapreduce.job.reduces=N;

4.分通表导入数据的方式：insert+select

insert数据来自于后select查询的结果

insert overwrite table stu\_buck

select \* from student cluster by(Sno);

#### 4）UDF

1. 什么是UDF

当 Hive 提供的内置函数无法满足你的业务处理需要时，此时就可以考虑使用用户自定义函数（UDF： user-defined function）。

2.UDF开发步骤示例

新建 JAVA maven 项目  
添加 hive-exec-1.2.1.jar 和 hadoop-common-2.7.4.jar 依赖（见参考资料）  
1、 写一个 java 类，继承 UDF，并重载 evaluate 方法

|  |
| --- |
| package cn.itcast.bigdata.udf import org.apache.hadoop.hive.ql.exec.UDF; import org.apache.hadoop.io.Text;  public class Lower extends UDF{  public Text evaluate(Text s){  if(s==null){return null;}  return new Text(s.toString().toLowerCase());  } } |

2、打成 jar 包上传到服务器  
3、将 jar 包添加到 hive 的 classpath  
hive>add JAR /home/hadoop/udf.jar;  
4、 创建临时函数与开发好的 java class 关联

|  |
| --- |
| create temporary function tolowercase as 'cn.itcast.bigdata.udf.ToProvince'; |

5、 即可在 hql 中使用自定义的函数 tolowercase ip  
Select tolowercase(name),age from t\_test;

## Hive面试题

### Hive常见

1. hive 的使用，内外部表的区别，分区作用， UDF 和 Hive 优化

(1)hive 使用：仓库、工具

(2)hive 内部表：加载数据到 hive 所在的 hdfs 目录，删除时，元数据和数据文件都删除

外部表：不加载数据到 hive 所在的 hdfs 目录，删除时，只删除表结构。

(3)分区作用：防止数据倾斜

(4)UDF 函数：用户自定义的函数 (主要解决格式，计算问题 )，需要继承 UDF 类

java 代码实现

class TestUDFHive extends UDF {

public String evalute(String str){

try{

return "hello"+str

}catch(Exception e){

return str+"error"

}

}

}

(5)Hive 优化：看做 mapreduce 处理

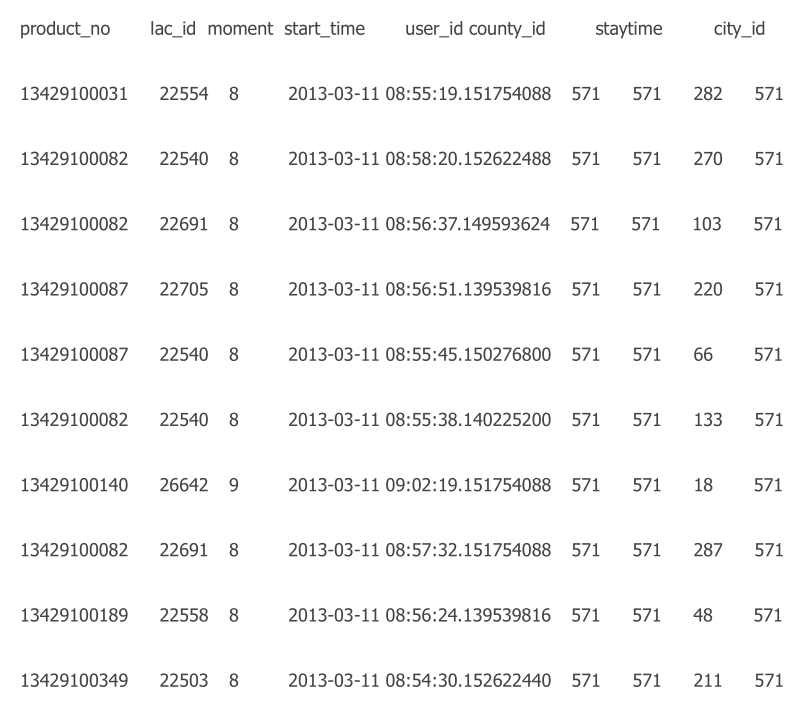
排序优化： sort by 效率高于 order by

分区：使用静态分区 (statu\_date="20160516",location="beijin") ，每个分区对应 hdfs 上

的一个目录，减少 job 和 task 数量：使用表链接操作，解决 groupby 数据倾斜问题：设置 hive.groupby.skewindata=true ，那么 hive 会自动负载均衡，小文件合并成大文件：表连接操作，使用 UDF 或 UDAF 函数：

http://www.cnblogs.com/ggjucheng/archive/2013/02/01/2888819.html

1. 使用 Hive 或者自定义 MR 实现如下逻辑



字段解释：

product\_no：用户手机号；

lac\_id：用户所在基站；

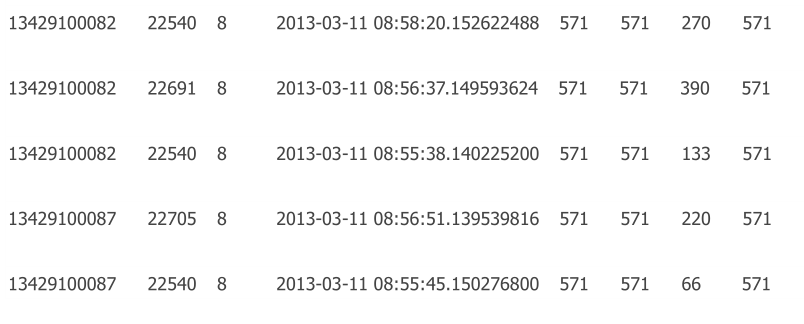
start\_time：用户在此基站的开始时间；

staytime：用户在此基站的逗留时间。

需求描述：

根据 lac\_id 和 start\_time 知道用户当时的位置，根据 staytime 知道用户各个基站的逗留时长。根据轨迹合并连续基站的 staytime。最终得到每一个用户按时间排序在每一个基站驻留时长。

期望输出举例：



1. 你们的数据是用什么导入到数据库的？导入到什么数据库？
2. 你们业务数据量多大？有多少行数据？(面试了三家，都问这个问题)
3. 你们写 hive 的 hql 语句，大概有多少条？
4. hive 跟 hbase 的区别是？

**共同点：**

1.hbase 与 hive 都是架构在 hadoop 之上的。都是用 hadoop 作为底层存储

**区别：**

2.Hive 是建立在 Hadoop 之上为了减少 MapReduce jobs 编写工作的批处理系统，HBase

是为了支持弥补 Hadoop 对实时操作的缺陷的项目 。

3.想象你在操作 RMDB 数据库，如果是全表扫描，就用 Hive+Hadoop,如果是索引访问，

就用 HBase+Hadoop 。

4.Hive query 就是 MapReduce jobs 可以从 5 分钟到数小时不止，HBase 是非常高效的，

肯定比 Hive 高效的多。

5.Hive 本身不存储和计算数据，它完全依赖于 HDFS 和 MapReduce，Hive 中的表纯逻辑。

6.hive 借用 hadoop 的 MapReduce 来完成一些 hive 中的命令的执行

7.hbase 是物理表，不是逻辑表，提供一个超大的内存 hash 表，搜索引擎通过它来存

储索引，方便查询操作。

8.hbase 是列存储。

9.hdfs 作为底层存储，hdfs 是存放文件的系统，而 Hbase 负责组织文件。

10.hive 需要用到 hdfs 存储文件，需要用到 MapReduce 计算框架。

1. 你们处理数据是直接读数据库的数据还是读文本数据？
2. 你们提交的 job 任务大概有多少个？这些 job 执行完大概用多少时间？
3. reduce 后输出的数据量有多大？
4. 你自己写过 udf 函数么？写了哪些？作用是什么？
5. 你在项目中主要的工作任务是？
6. hive 底层与数据库交互原理
7. 使用 Hive 进行手机流量统计

问题导读

1. hive 实现统计的查询语句是什么？
2. 生产环境中为什么建议使用外部表？
3. hadoop mapreduce 创建类 DataWritable 的作用是什么？
4. 为什么创建 类 类 DataWritable ？
5. 如何实现统计手机流量？

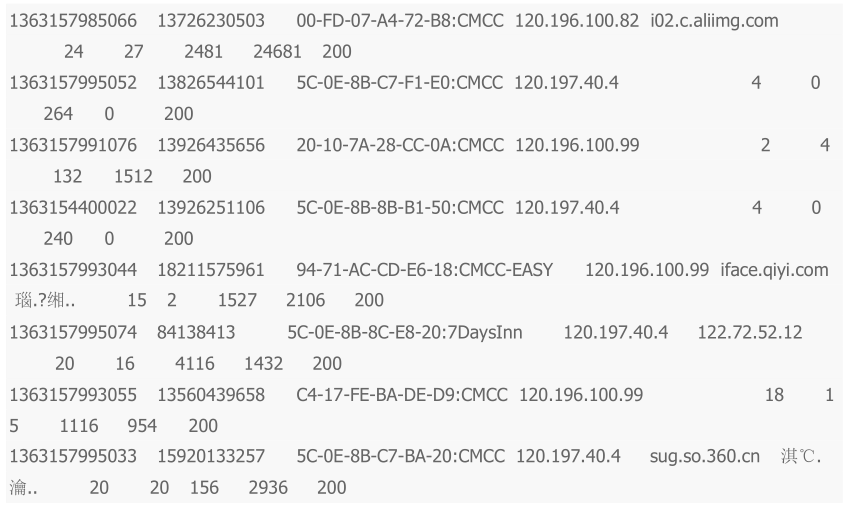
6. 对比 hive 与 与 mapreduce 统计手机流量的区别？

很多公司在使用 hive 对数据进行处理。hive 是 hadoop 家族成员，是一种解析 like sql 语句的框架。它封装了常用 MapReduce 任务，让你像执行sql 一样操作存储在 HDFS 的表。

hive 的表分为两种，内表和外表。Hive 创建内部表时，会将数据移动到数据仓库指向的路径；若创建外部表，仅记录数据所在的路径，不对数据的位置做任何改变。在删除表的时候，内部表的元数据和数据会被一起删除， 而外部表只删除元数据，不删除数据。这样外部表相对来说更加安全些，数据组织也更加灵活，方便共享源数据。

Hive 的内外表，还有一个 Partition 的分区的知识点，用于避免全表扫描，快速检索。后期的文章会提到。

原始数据



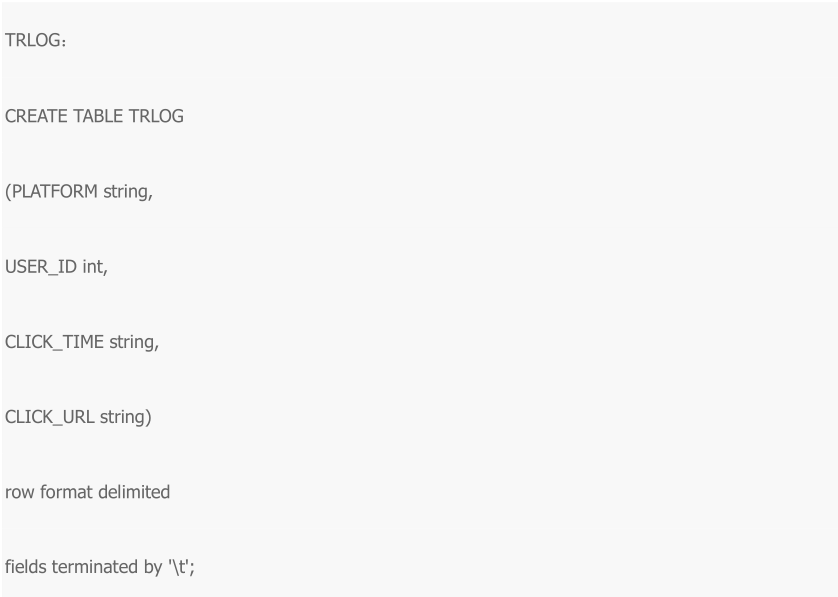
操作步骤



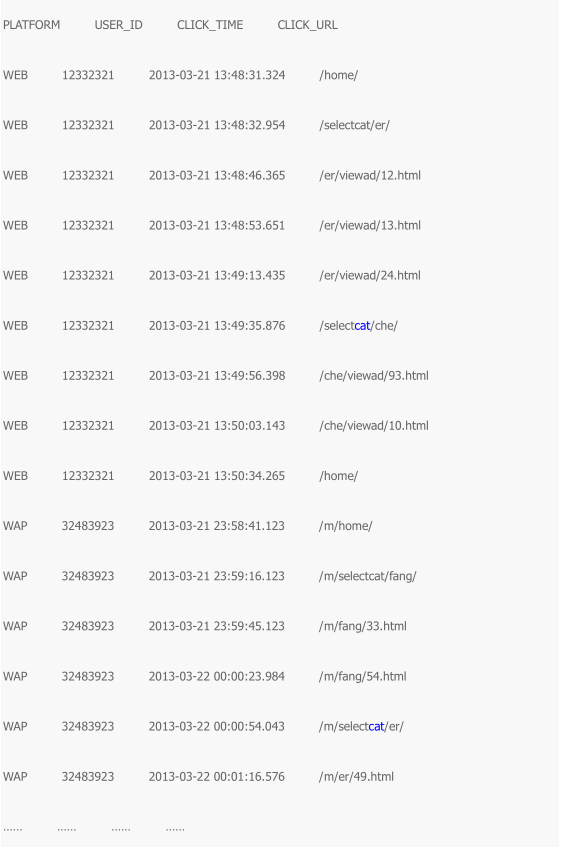
执行结果如下



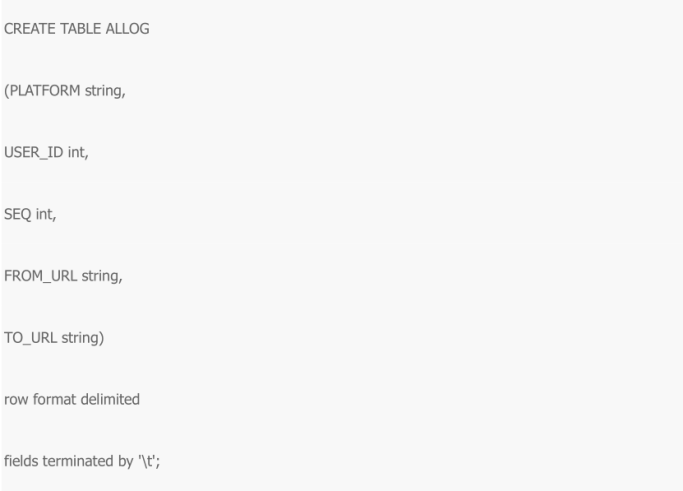
1. 表大概有 2T 左右，对表数据转换



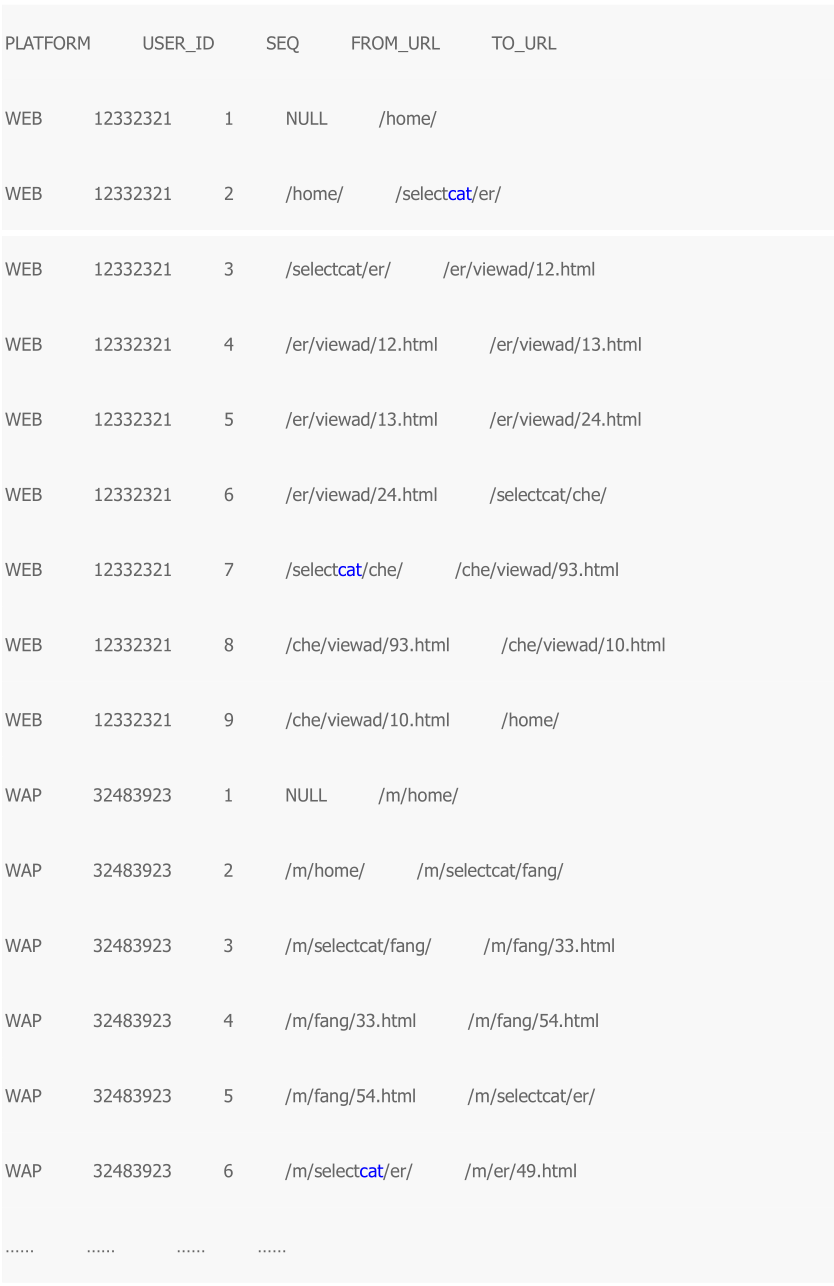
数据



需要把上述数据处理为如下结构的表 ALLOG：



整理后的数据结构：

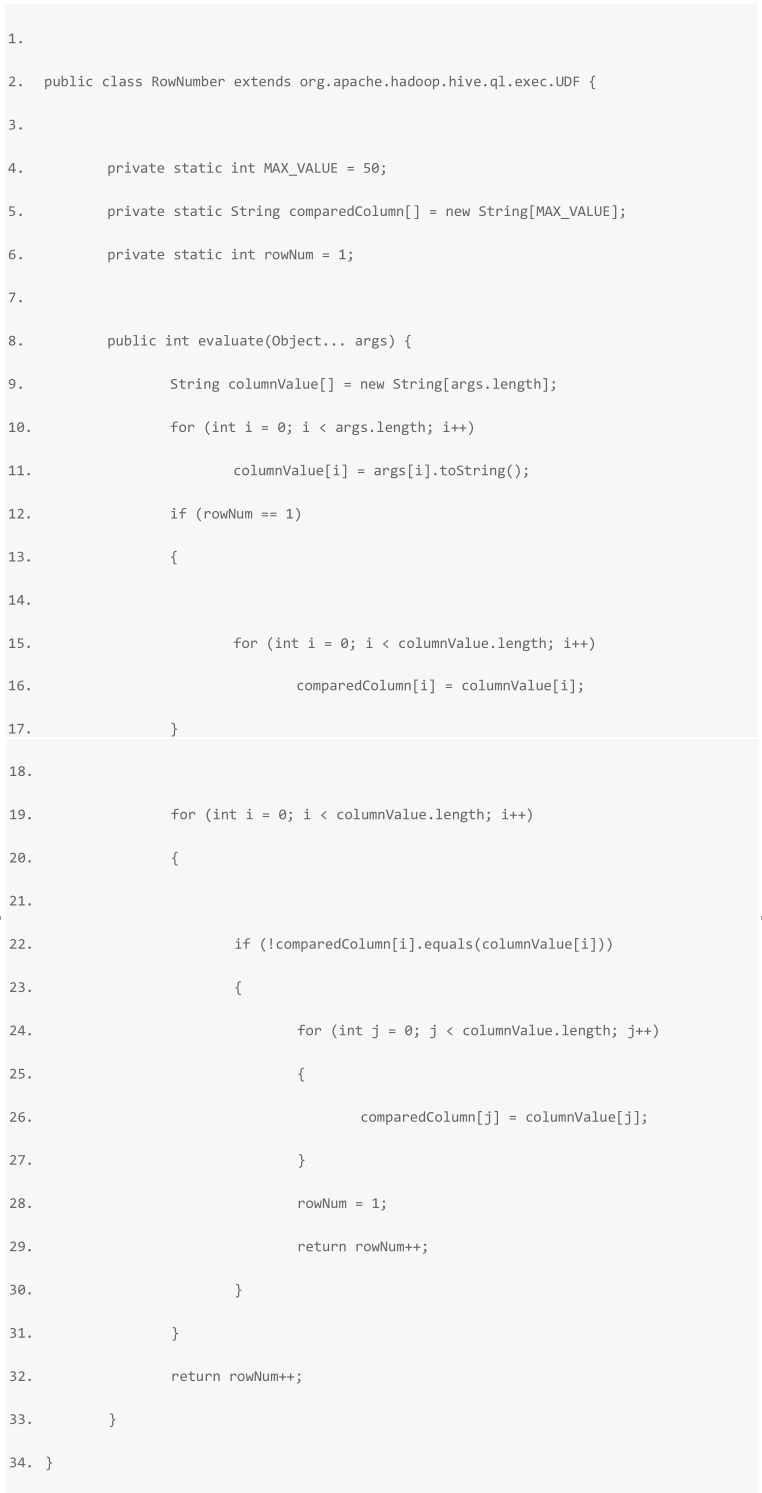


PLATFORM 和 USER\_ID 还是代表平台和用户 ID；SEQ 字段代表用户按时间排序后的访问顺序，FROM\_URL和 TO\_URL 分别代表用户从哪一页跳转到哪一页。对于某个平台上某个用户的第一条访问记录，其FROM\_URL 是 NULL（空值）。

面试官说需要用两种办法做出来：

1、实现一个能加速上述处理过程的 Hive Generic UDF，并给出使用此 UDF 实现 ETL 过程的 Hive SQL

给你个 JAVA 写的 RowNumber 方法



把这个 JAVA 打包，编译成 JAR 包，比如 RowNumber.jar。然后放到 HIVE 的机器上在 HIVE SHELL 里执行下面两条语句：

1. add jar /root/RowNumber.jar;

2. #把 RowNumber.jar 加载到 HIVE 的 CLASSPATH 中

3. create temporary function row\_number as 'RowNumber';

4. #在 HIVE 里创建一个新函数，叫 row\_number ，引用的 CLASS 就是 JAVA 代码里的 RowNumber

提示成功后，执行下面这条 HIVE SQL

2. #INSERT OVERWRITE TABLE ALLOG 如果要写入 ALLOG 表，可以把注释去掉

3. SELECT t1.platform,t1.user\_id,row\_number(t1.user\_id)seq,t2.click\_url

FROM\_URL,t1.click\_url TO\_URL FROM

4. (select \*,row\_number(user\_id)seq from trlog)t1

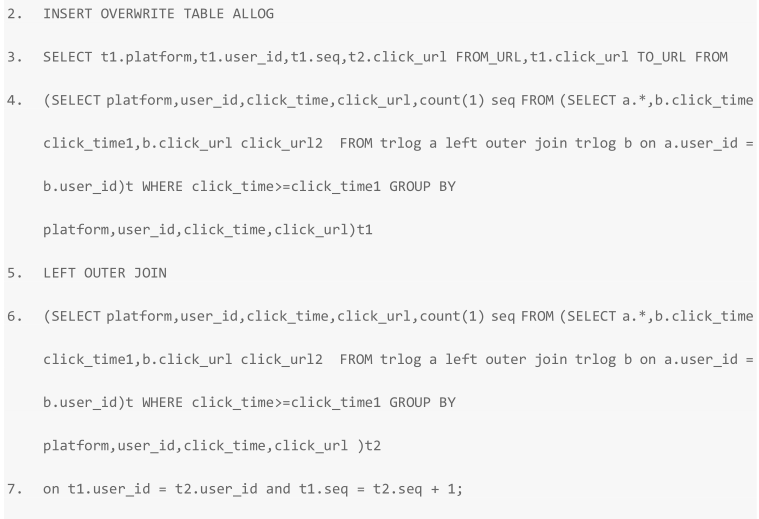
5. LEFT OUTER JOIN

6. (select \*,row\_number(user\_id)seq from trlog)t2

7. on t1.user\_id = t2.user\_id and t1.seq = t2.seq + 1;

第一题中的 RN 貌似是 HIVE 转译 SQL 的 BUG，你可以把外层的 ROW\_NUMBER 去掉，用 T1 的 SEQ，就能发现问题了。

2、实现基于纯 Hive SQL 的 ETL 过程，从 TRLOG 表生成 ALLOG 表；（结果是一套 SQL）



1. hive 有哪些保存元数据的方式，个有什么特点。

1、 内存数据库 derby，安装小，但是数据存在内存，不稳定

2、 mysql 数据库，数据存储模式可以自己设置，持久化好，查看方便

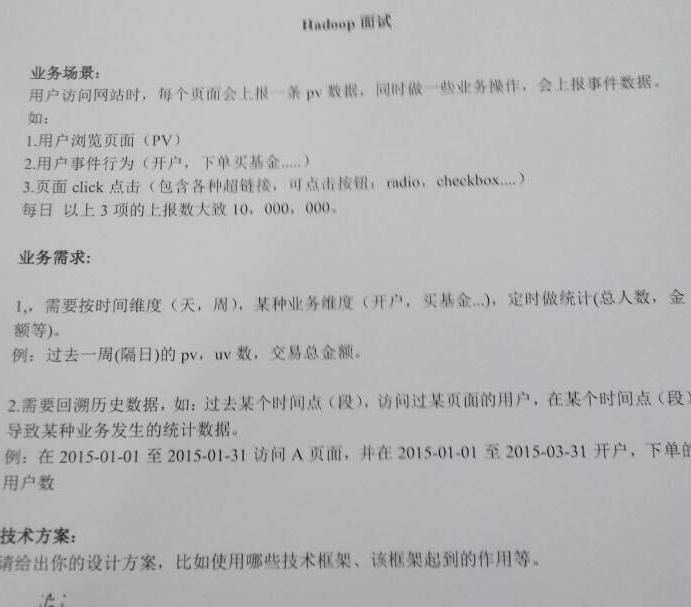
1. 生产环境中为什么建议使用外部表？

1、因为外部表不会加载数据到 hive，减少数据传输、数据还能共享。

2、hive 不会修改数据，所以无需担心数据的损坏

3、删除表时，只删除表结构、不删除数据。

1. 如何用 hive 分析业务数据，将数据导入到 hive 中



1、 用 hive 分析业务数据即可

2、 将数据导入到 hive 中

sql 的设计思路：多表关联

1、 找到所有在 2015-01-01 到 2015-01-31 时间内访问 A 页面的用户

2、 在这些用户中删选在 2015-01-01 到 2015-03-31 下单的用户

3、 统计总数

1. 你们数据库怎么导入 hive 的, 有没有出现问题

在导入hive的时候，如果数据库中有blob或者text字段会报错，解决方案在sqoop笔记中

1. Hive 你们用的是外部表还是内部表, , 有没有写过 UDF,hive 的版本外部表和内部表的区别
2. 一个Hadoop环境，整合了hbase和hive，是否有必要给HDFS和hbase都分别配置压缩策略？请给出对压缩策略的建议

hdfs 在存储的时候不会将数据进行压缩，如果想进行压缩，我们可以在向 hdfs 上传数据的时候进行压缩。

1、 采用压缩流

//压缩文件

public static void compress(String codecClassName) throws

Exception{

Class<?> codecClass = Class.forName(codecClassName);

Configuration conf = new Configuration();

FileSystem fs = FileSystem.get(conf);

CompressionCodec codec =

(CompressionCodec)ReflectionUtils.newInstance(codecClass, conf);

//指定压缩文件路径

FSDataOutputStream outputStream = fs.create(new

Path("/user/hadoop/text.gz"));

//指定要被压缩的文件路径

FSDataInputStream in = fs.open(new

Path("/user/hadoop/aa.txt"));

//创建压缩输出流

CompressionOutputStream out =

codec.createOutputStream(outputStream);

IOUtils.copyBytes(in, out, conf);

IOUtils.closeStream(in);

IOUtils.closeStream(out);

}

2、 采用序列化文件

public void testSeqWrite() throws Exception {

Configuration conf = new Configuration();// 创建配置信息

conf.set("fs.default.name", "hdfs://master:9000");//hdfs 默认路径

conf.set("hadoop.job.ugi", "hadoop,hadoop");// 用户和组信息

String uriin = "hdfs://master:9000/ceshi2/";// 文件路径

FileSystem fs = FileSystem.get(URI.create(uriin),conf);// 创建 filesystem

Path path = newPath("hdfs://master:9000/ceshi3/test.seq");// 文件名

IntWritable k = new IntWritable();// key，相当于 intText v = new Text(); value，相当于 String

SequenceFile.Writer w = SequenceFile.createWriter(fs,conf, path,k.getClass(), v.getClass());// 创建 writer

for (int i = 1; i < 100; i++) {// 循环添加

k.set(i);

v.set("abcd");

w.append(k, v);

}

w.close();

IOUtils.closeStream(w);// 关闭的时候 flush

fs.close();

}

hbase 为列存数据库，本身存在压缩机制，所以无需设计。

1. 简述Hive中的虚拟列作用是什么?使用他的注意事项

Hive 提供了三个虚拟列：

INPUT\_\_FILE\_\_NAME

BLOCK\_\_OFFSET\_\_INSIDE\_\_FILE

ROW\_\_OFFSET\_\_INSIDE\_\_BLOCK

但 ROW\_\_OFFSET\_\_INSIDE\_\_BLOCK 默认是不可用的，需要设置 hive.exec.rowoffset 为 true才可以。可以用来排查有问题的输入数据。

INPUT\_\_FILE\_\_NAME, mapper 任务的输出文件名。

BLOCK\_\_OFFSET\_\_INSIDE\_\_FILE, 当前全局文件的偏移量。对于块压缩文件，就是当前块的文件偏移量，即当前块的第一个字节在文件中的偏移量。

hive> SELECT INPUT\_\_FILE\_\_NAME, BLOCK\_\_OFFSET\_\_INSIDE\_\_FILE, line

> FROM hive\_text WHERE line LIKE '%hive%' LIMIT 2;

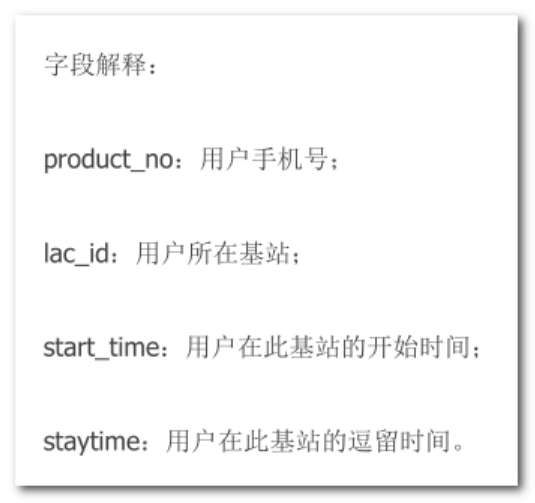
har://file/user/hive/warehouse/hive\_text/folder=docs/

data.har/user/hive/warehouse/hive\_text/folder=docs/README.txt 2243

har://file/user/hive/warehouse/hive\_text/folder=docs/

data.har/user/hive/warehouse/hive\_text/folder=docs/README.txt 3646

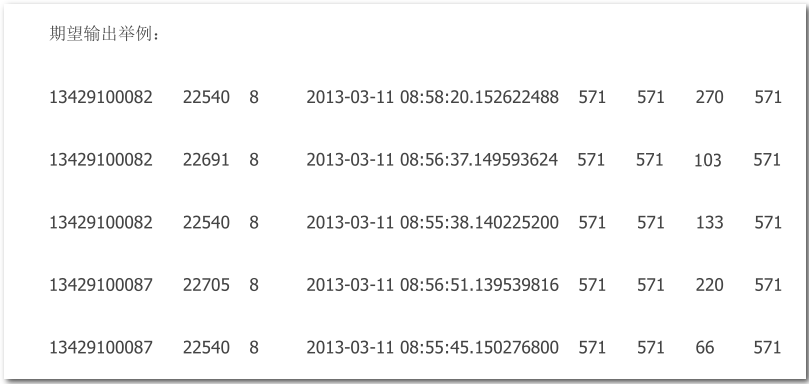
基站逗留时间

需求



期望



思路：

将数据导入 hive 表中，查询时，用电话号码和时间排序即可！

1. 你们数据库怎么导入 hive 的,有没有出现问题

使用 sqoop 导入，我们公司的数据库中设计了 text 字段，导致导入的时候出现了缓存

不够的情况（见云笔记），开始解决起来感觉很棘手，后来查看了 sqoop 的文档，加上

了 limit 属性，解决了

1. Redis,传统数据库,hbase,hive 每个之间的区别(问的非常细)

Redis 是缓存，围绕着内存和缓存说

Hbase 是列式数据库，存在 hdfs 上，围绕着数据量来说

Hive 是数据仓库，是用来分析数据的，不是增删改查数据的。

1. Hive 你们用的是外部表还是内部表,有没有写过 UDF,hive 的版本外部表，udf，udaf 等，hive 版本为 1.0
2. hive partition 分区分区表，动态分区
3. insert into 和 override write 区别？

insert into：将某一张表中的数据写到另一张表中

override write：覆盖之前的内容。

1. 假如一个分区的数据主部错误怎么通过 hivesql 删除 hdfs

alter table ptable drop partition (daytime='20140911',city='bj');

元数据，数据文件都删除，但目录 daytime= 20140911 还在

1. 请简述一下 Hadoop/MapReduce， Spark ， Strom ， Hive 的特点及适用场景？

Hadoop ：是一种分布式系统基础架构当处理海量数据的程序，开始要求高可靠、高扩展、

高效、低容错、低成本的场景

MapReduce： MapReduce 是一种编程模型，用于大规模数据集（大于 1TB）的并行运算。

MapReduce 的典型应用场景中，目前日志分析用的比较多，还有做搜素的索引，

机器学习算法包 mahout 也是之一，当然它能做的东西还有很多，比如数据掘、

信息提取。

Spark：拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是 Job 中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。数据过于繁杂，并且需要让计算通过迭代，并在内存中，极大地提高效率的场景

Strom：一个分布式实时计算系统，Storm 是一个任务并行连续计算引擎。 Storm 本身并不

典型在 Hadoop 集群上运行，它使用 Apache ZooKeeper 的和自己的主/从工作进程，协调拓扑，主机和工作者状态，保证信息的语义。无论如何， Storm 必定还是可以从 HDFS 文件消费或者从文件写入到 HDFS。

Hive：基于 Hadoop 的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，

并提供简单的 sql 查询功能，可以将 sql 语句转换为 MapReduce 任务进行运行。

应用场景：十分适合数据仓库的统计分析。

Hbase:应用场景： 数据量太大，以至于传统 RDBMS 无法胜任、联机业务功能开发、离线数据分析（数据仓库），

1. Hive 的条件判断有几种？

hive 的条件判断（if、coalesce、case）

1. 请适用 hive 写出查询某网站日志中方位多页面 a 和页面 b 的用户数量的语句：

Select count(user) from urla a , urlb b where a.url = b.url ;

1. Orcal、hive、mysql 分页

Orcal 分页

第一种：显示从 5 到 10 的员工

1、table01 按照工资降序排列出所有员工

select \* from emp order by sal desc

2、table02 在员工表 table01 中设定 rownum，rownum<=5

select rownum no, e.\*

from (select \* from emp order by sal desc) e

where rownum <= 5

3、在 table02 中限定条件 rownum>3

select \*

from (select rownum no, e.\*

from (select \* from emp order by sal desc) e

where rownum <= 10) where no>5

第二种：显示从 5 到 10 的员工

select \*

from

(select rownum no, e.\* from (select \* from emp order by sal desc) e)

where

no >= 5 and no <= 10

hive 中的分页

第一种：

一、借助唯一标识字段

如果分页的表有唯一标识的字段，可以通过这个字段来实现分页：

• 获取第一页数据：

注：同时需要记录这 10 条中最大的 id 为 preId，作为下一页的条件。

select \* from table order by id asc limit 10;

• 获取第二页数据：

注：同时保存数据中最大的 id 替换 preId。

select \* from table where id >preId order by id asc limit 10;

二、使用 row number() 函数

如果分页的表没有唯一标识的字段，可以通过 row number()函数来实现分页。

• 首先使用 row number()函数来给这个表做个递增的唯一标识：

create table newtable as select row number(1) as id ,\* from table;

• 通过 row number 函数给表加了唯一标识之后，就可以利用 第一个方法 来进行分页。

MySQL 中分页实现

我们来贴例子吧！

mysql> select pname from product;

+--------+

| pname |

+--------+

| 产品 1 |

| 产品 2 |

| 产品三 |

+--------+

3 rows in set (0.00 sec)

这个地方是说，从 product 中选出所有的 pname 来，一共有三条记录。

MySQL 中的分页非常简单，我们可以使用 limit

比如：

mysql> select pname from product limit 0,2;

+-------+

| pname |

+-------+

| 产品 1 |

| 产品 2 |

+-------+

2 rows in set (0.00 sec)

Limit 用法如下：

第一个参数是指要开始的地方，第二个参数是指每页显示多少条数据；注意：第一页用 0

表示。

l Mysql 分页：

select \* from tableName where 条件 limit 当前页码\*页面容量

-1 , 页面容量

1. Hive 中的排序方法有哪些

order by

order by 会对输入做全局排序，因此只有一个 reducer（多个 reducer 无法保全局有序）

只有一个 reducer，会导致当输入规模较大时，需要较长的计算时间。

Sql 代码

1. set hive.mapred.mode=nonstrict; (default value / 默认值)

2. set hive.mapred.mode=strict;

order by 和数据库中的 Order by 功能一致，按照某一项 & 几项 排序输出。与数据库中 order by 的区别在于在 hive.mapred.mode = strict 模式下 必须指定limit 否则执行会报错。

Sql 代码

1. hive> select \* from test order by id;

Java 代码

1. FAILED: Error in semantic analysis: 1:28 In strict mode, if ORDER BY is specified, LIMIT must also be specified. Error encountered near token 'id'

原因： 在 order by 状态下所有数据会到一台服务器进行 reduce 操作也即只有一个reduce，如果在数据量大的情况下会出现无法输出结果的情况，如果进行 limit n ，那只有 n \* map number 条记录而已。只有一个 reduce 也可以处理过来。

sort by

sort by 不是全局排序，其在数据进入 reducer 前完成排序.因此，如果用 sort by 进行排序，并且设置 mapred.reduce.tasks>1， 则 sort by 只保证每个 reducer 的输出有序，不保证全局有序。

sort by 不受 hive.mapred.mode 是否为 strict ,nostrict 的影响

sort by 的数据只能保证在同一 reduce 中的数据可以按指定字段排序。使用 sort by 你可以指定执行的 reduce 个数 （set mapred.reduce.tasks=<number>）,对输出的数据再执行归并排序，即可以得到全部结果。注意：可以用 limit 子句大大减少数据量。使用 limit n 后，传输到 reduce 端（单机）的数据记录数就减少到 n\* （map 个数）。否则由于数据过大可能出不了结果。

distribute by

按照指定的字段对数据进行划分到不同的输出 reduce / 文件中。

insert overwrite local directory '/home/hadoop/out' select \* from test order by

name distribute by length(name);

此方法会根据 name 的长度划分到不同的 reduce 中，最终输出到不同的文件中。length 是内建函数，也可以指定其他的函数或这使用自定义函数。

Cluster By

cluster by 除了具有 distribute by 的功能外还兼具 sort by 的功能。但是排序只能是倒序排序，不能指定排序规则为 asc 或者 desc

1. hive 和 hbase 方法：项目中采集的哪些指标，
2. 你们提交的 job 任务大概有多少个？这些 job 执行完大概用多少时间？(面试了三家，都问这个问题)

每天六百个，2，3 个小时。

1. 你在项目中主要的工作任务是？

利用 hive 分析数据

1. 一个网络商城

**1.**1天大概产生多少 G 的日志？大概有多少条日志记录（在不清洗的情况下）？一个网络商城 1 天大概产生多少 G 的日志？ 4tb

**2.**大概有多少条日志记录（在不清洗的情况下）？ 7-8 百万条

**3.**日访问量大概有多少个？ 百万

**4.**注册数大概多少？ 不清楚 几十万吧

**5.**我们的日志是不是除了 apache 的访问日志是不是还有其他的日志？

**6.**假设我们有其他的日志是不是可以对这个日志有其他的业务分析？这些业务分析有什么

1. 你认为用 java,streaming,pipe 方式开发 map/reduce ,各有哪些优点就用过 java 和 hiveQL。

Java 写 mapreduce 可以实现复杂的逻辑，如果需求简单，则显得繁琐。HiveQL 基本都是针对 hive 中的表数据进行编写，但对复杂的逻辑很难进行实现。写起来简单。

1. hive 有哪些方式保存元数据，各有哪些优点

**三种**：内存数据库 derby，挺小，不常用 。本地 mysql。。常用远程端 mysql。。不常用上网上找了下专业名称：single user mode..multi user mode...remote user mode

1. hive 内部表和外部表的区别

Hive 创建内部表时，会将数据移动到数据仓库指向的路径；若创建外部表，仅记录数据所在的路径， 不对数据的位置做任何改变。在删除表的时候，内部表的元数据和数据会被一起删除， 而外部表只删除元数据，不删除数据。这样外部表相对来说更加安全些，数据组织也更加灵活，方便共享源数据。

1. hive 底层与数据库交互原理

hive 有一套自己的 sql 解析引擎，称为 metastore，存储在 mysql 或者 derby 数据库中，可以将 sql 语句转化为 mapreducejob 任务执行。

1. sqoop 在导入数据到 mysql 中，如何不重复导入数据，如果存在数据问题，sqoop 如何处理？
2. pig , latin , hive 语法有什么不同
3. hive 如何调优

底层是 MapReduce，所以又可以说是 MapReduce 优化。

1. 小文件都合并成大文件

2. Reducer 数量在代码中介于节点数\*reduceTask 的最大数量的 0.95 倍到 1.75 倍

3. 写一个 UDF 函数，在建表的时候制定好分区

4. 配置文件中，打开在 map 端的合并

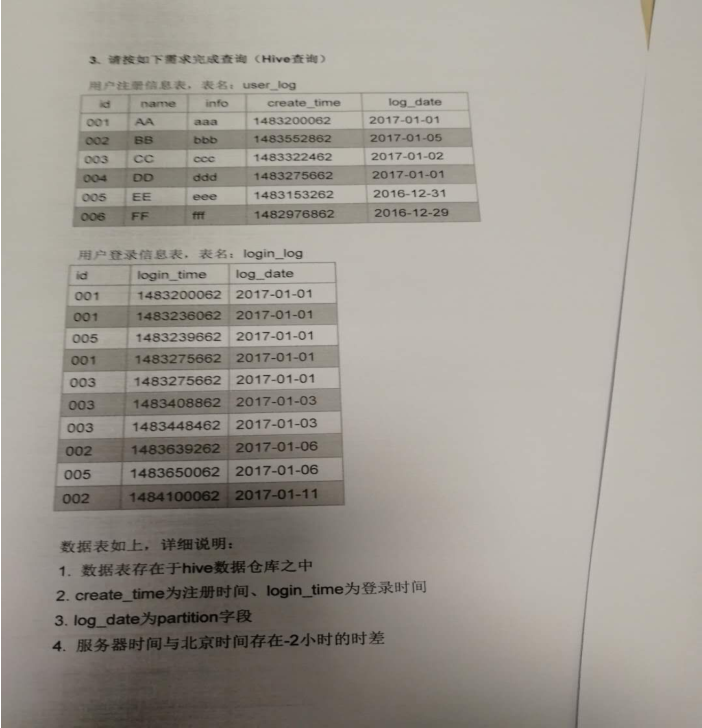
5.在库表设计的时候，尽量考虑rowkey和columnfamily的特性

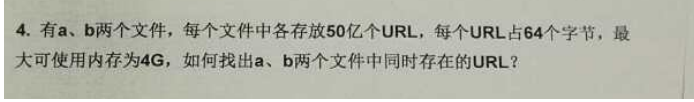
6.进行hbase集群的调优：见hbase调优

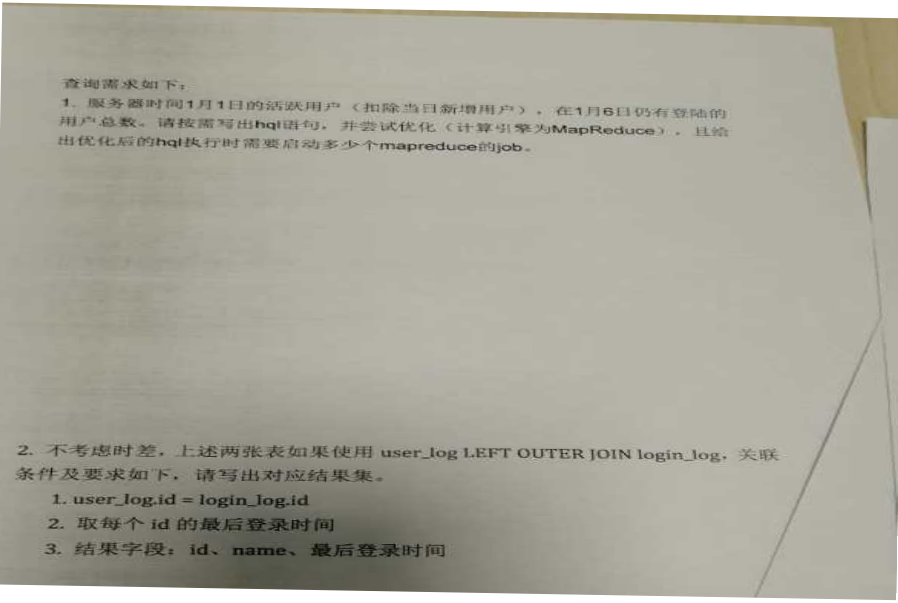
1. hive 如何控制权限
2. hive 能像关系型数据库那样建多个库吗？
3. 假设公司要建一个数据中心，你会如何处理？
4. Hive 中的 metastore 用来做什么的？

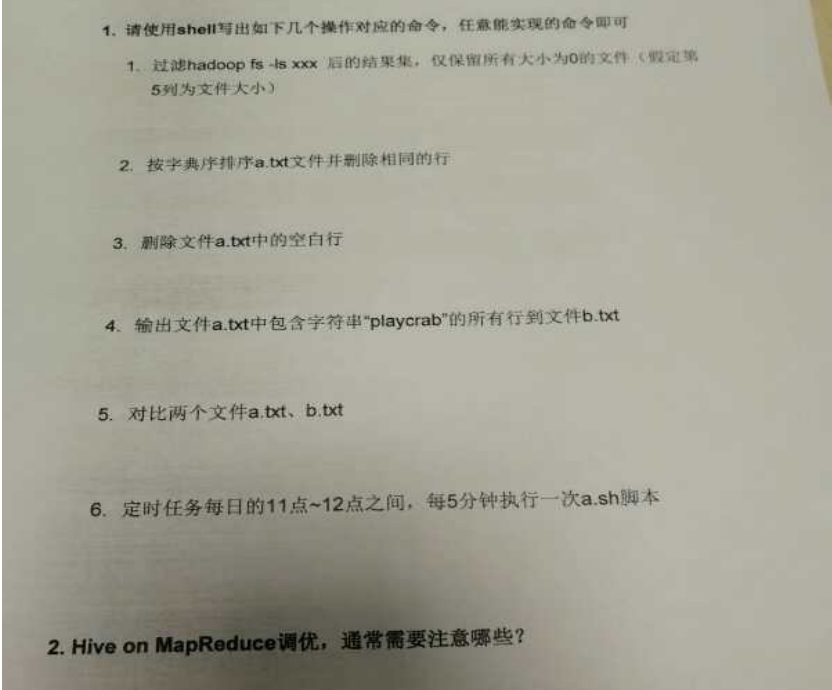
metastore 是一套映射工具，将 sql 语句转换成对应的 job 任务去进行执行。

1. hive 中的压缩格式 RCFile、TextFile、SequenceFile 各有什么区别？以上 3 种格式一样大的文件哪个占用空间大小
2. 你们的 hive 处理数据能达到的指标是多少？
3. Hive查询案例









1. 给定 a、b 两个文件，各存放 50 亿个 url，每个 url 各占 64 字节，内存限制是 4G，让你找出 a、b 文件共同的 url？

方案 1：可以估计每个文件安的大小为 50G×64=320G，远远大于内存限制的 4G。所以不

可能将其完全加载到内存中处理。考虑采取分而治之的方法。 遍历文件 a，对每个 url 求取 ，然后根据所取得的值将 url 分别存储到1000 个小文件（记为 ）中。这样每个小文件的大约为 300M。s 遍历文件 b，采取和 a 相同的方式将 url 分别存储到 1000 个小文件（记为 ）。这样处理后，所有可能相同的 url 都在对应的小文件（ ）中，不对应的小文件不可能有相同的 url。然后我们只要求出 1000 对小文件中相同的 url 即可。s 求每对小文件中相同的 url 时，可以把其中一个小文件的 url 存储到 hash\_set 中。然后遍历另一个小文件的每个 url，看其是否在刚才构建的 hash\_set 中，如果是，那么就是共同的 url，存到文件里面就可以了。

方案 2：如果允许有一定的错误率，可以使用 Bloom filter，4G 内存大概可以表示 340 亿bit。将其中一个文件中的 url 使用 Bloom filter 映射为这 340 亿 bit，然后挨个读取另外一个文件的 url，检查是否与 Bloom filter，如果是，那么该 url 应该是共同的 url（注意会有一定的错误率）。

1. 有 10 个文件，每个文件 1G ，每个文件的每一行存放的都是用户的 query ，每个文件的 query 都可能重复。要求你按照 query 的频度排序。

方案 1：

顺序读取 10 个文件，按照 hash(query)%10 的结果将 query 写入到另外 10 个文件（记为）中。这样新生成的文件每个的大小大约也 1G（假设 hash 函数是随机的）。 找一台内存在 2G 左右的机器，依次对用 hash\_map(query, query\_count)来统计每个 query 出现的次数。利用快速/堆/归并排序按照出现次数进行排序。将排序好的 query 和对应的 query\_cout 输出到文件中。这样得到了 10 个排好序的文件（记为）。 对这 10 个文件进行归并排序（内排序与外排序相结合）。

方案 2：

一般 query 的总量是有限的，只是重复的次数比较多而已，可能对于所有的 query，一

次性就可以加入到内存了。这样，我们就可以采用 trie 树/hash\_map 等直接来统计每

个 query 出现的次数，然后按出现次数做快速/堆/归并排序就可以了。

方案 3：

与方案 1 类似，但在做完 hash，分成多个文件后，可以交给多个文件来处理，采用分

布式的架构来处理（比如 MapReduce），最后再进行合并。

1. 有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超

过 16 字节，内存限制大小是 1M。返回频数最高的 100 个词。方案 1：顺序读文件中，对于每个词 x，取 ，然后按照该值存到 5000 个小文件（记为 ）中。这样每个文件大概是 200k 左右。如果其中的有的文件超过了 1M 大小，还可以按照类似的方法继续往下分，知道分解得到的小文件的大小都不超过1M。对每个小文件，统计每个文件中出现的词以及相应的频率（可以采用 trie 树/hash\_map等），并取出出现频率最大的 100 个词（可以用含 100 个结点的最小堆），并把 100 词及相应的频率存入文件，这样又得到了 5000 个文件。下一步就是把这 5000 个文件进行归并（类似与归并排序）的过程了。

1. 海量日志数据，提取出某日访问百度次数最多的那个 IP。

方案 1：首先是这一天，并且是访问百度的日志中的 IP 取出来，逐个写入到一个大文件中。

注意到 IP 是 32 位的，最多有 个 IP。同样可以采用映射的方法，比如模 1000，把整个

大文件映射为 1000 个小文件，再找出每个小文中出现频率最大的 IP（可以采用 hash\_map

进行频率统计，然后再找出频率最大的几个）及相应的频率。然后再在这 1000 个最大的 IP

中，找出那个频率最大的 IP，即为所求。

1. 在 2.5 亿个整数中找出不重复的整数，注 ，内存不足以容纳这 2.5 亿个整数。

方案 1：采用 2-Bitmap（每个数分配 2bit，00 表示不存在，01 表示出现一次，10 表

示多次，11 无意义）进行，共需内存 2^32 \* 2 bit=1 GB 内存，还可以接受。然

后扫描这 2.5 亿个整数，查看 Bitmap 中相对应位，如果是 00 变 01，01 变 10，

10 保持不变。所描完事后，查看 bitmap，把对应位是 01 的整数输出即可。

方案 2：也可采用与第 1 题类似的方法，进行划分小文件的方法。然后在小文件中找出

不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

1. 海量数据分布在 100 台电脑中，想个办法高校统计出这批数据的 TOP10。

方案 1：

s 在每台电脑上求出 TOP10，可以采用包含 10 个元素的堆完成（TOP10 小，用最大堆，

TOP10 大，用最小堆）。比如求 TOP10 大，我们首先取前 10 个元素调整成最小堆，如果

发现，然后扫描后面的数据，并与堆顶元素比较，如果比堆顶元素大，那么用该元素替换堆

顶，然后再调整为最小堆。最后堆中的元素就是 TOP10 大。

s 求出每台电脑上的 TOP10 后，然后把这 100 台电脑上的 TOP10 组合起来，共 1000 个

数据，再利用上面类似的方法求出 TOP10 就可以了。

1. 怎么在海量数据中找出重复次数最多的一个？

方案 1：先做 hash，然后求模映射为小文件，求出每个小文件中重复次数最多的一个，并

记录重复次数。然后找出上一步求出的数据中重复次数最多的一个就是所求（具体参考前面

的题）。

1. 上千万或上亿数据（有重复），统计其中出现次数最多的钱 N 个数据。

方案 1：上千万或上亿的数据，现在的机器的内存应该能存下。所以考虑采用 hash\_map/

搜索二叉树/红黑树等来进行统计次数。然后就是取出前 N 个出现次数最多的数据了，可以

用第 6 题提到的堆机制完成。

1. 1000 万字符串，其中有些是重复的，需要把重复的全部去掉，保留没有重复的字符串。

请怎么设计和实现？

方案 1：这题用 trie 树比较合适，hash\_map 也应该能行。

1. 一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前 10 个词，

请给出思想，给出时间复杂度分析。

方案 1：这题是考虑时间效率。用 trie 树统计每个词出现的次数，时间复杂度是 O(n\*le)（le

表示单词的平准长度）。然后是找出出现最频繁的前 10 个词，可以用堆来实现，前面的题

中已经讲到了，时间复杂度是 O(n\*lg10)。所以总的时间复杂度，是 O(n\*le)与 O(n\*lg10)

中较大的哪一个。

1. 一个文本文件，找出前 10 个经常出现的词，但这次文件比较长，说是上亿行或十亿行，

总之无法一次读入内存，问最优解。

方案 1：首先根据用 hash 并求模，将文件分解为多个小文件，对于单个文件利用上题的方

法求出每个文件件中 10 个最常出现的词。然后再进行归并处理，找出最终的 10 个最常出

现的词。

1. 100w 个数中找出最大的 100 个数。

方案 1：在前面的题中，我们已经提到了，用一个含 100 个元素的最小堆完成。复杂度为

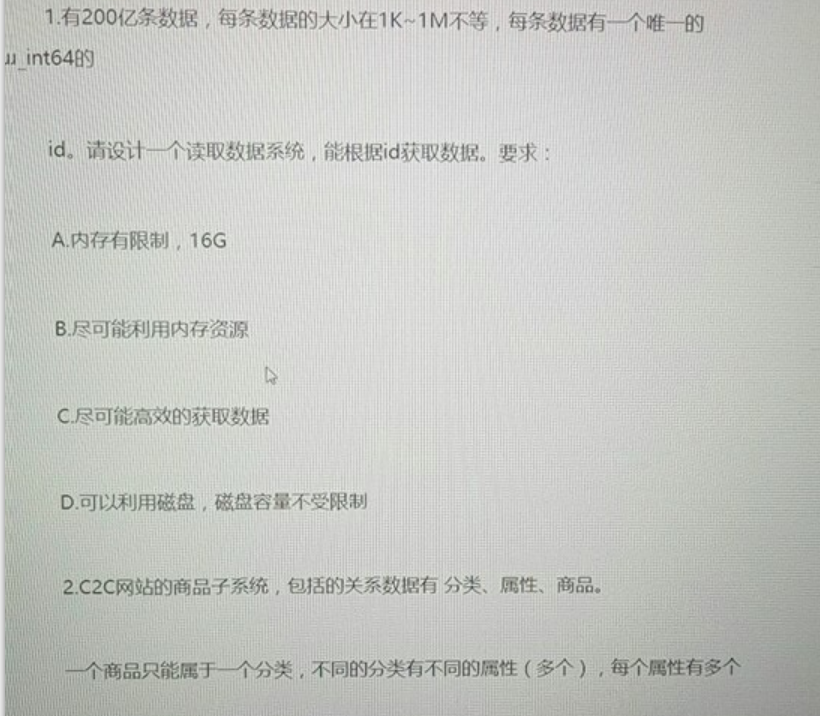
O(100w\*lg100)。

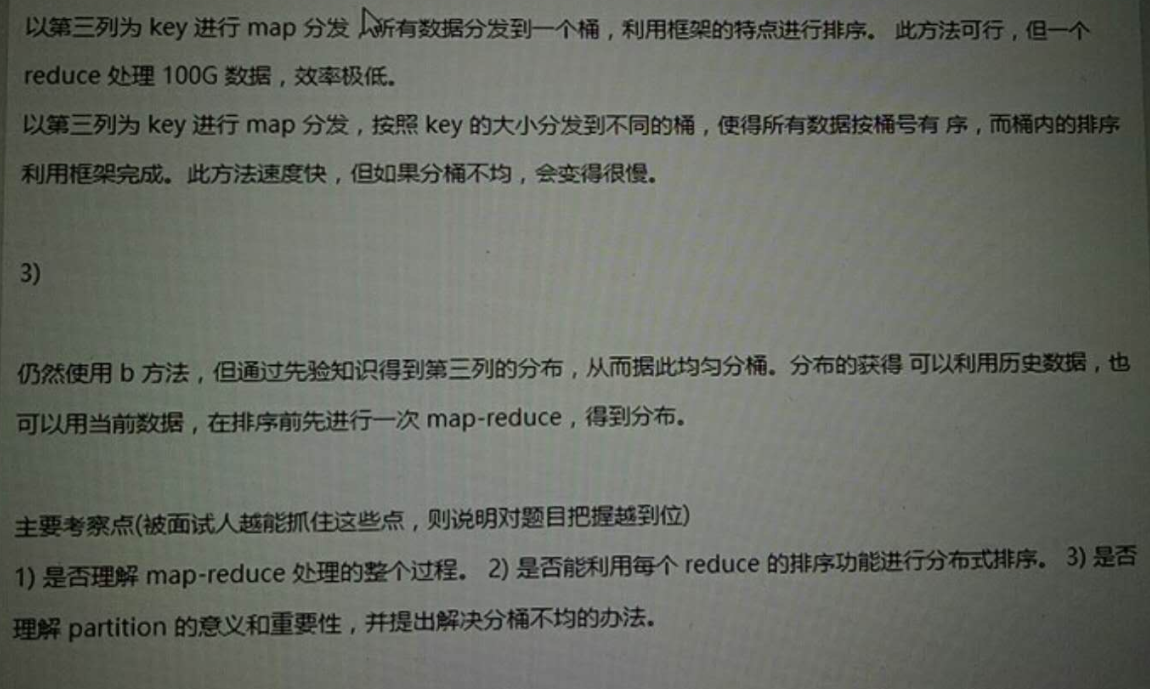
方案 2：采用快速排序的思想，每次分割之后只考虑比轴大的一部分，知道比轴大的一部分

在比 100 多的时候，采用传统排序算法排序，取前 100 个。复杂度为 O(100w\*100)。方案

3：采用局部淘汰法。选取前 100 个元素，并排序，记为序列 L。然后一次扫描剩余的元素x，与排好序的 100 个元素中最小的元素比，如果比这个最小的要大，那么把这个最小的元素删除，并把 x 利用插入排序的思想，插入到序列 L 中。依次循环，知道扫描了所有的元素。复杂度为 O(100w\*100)。

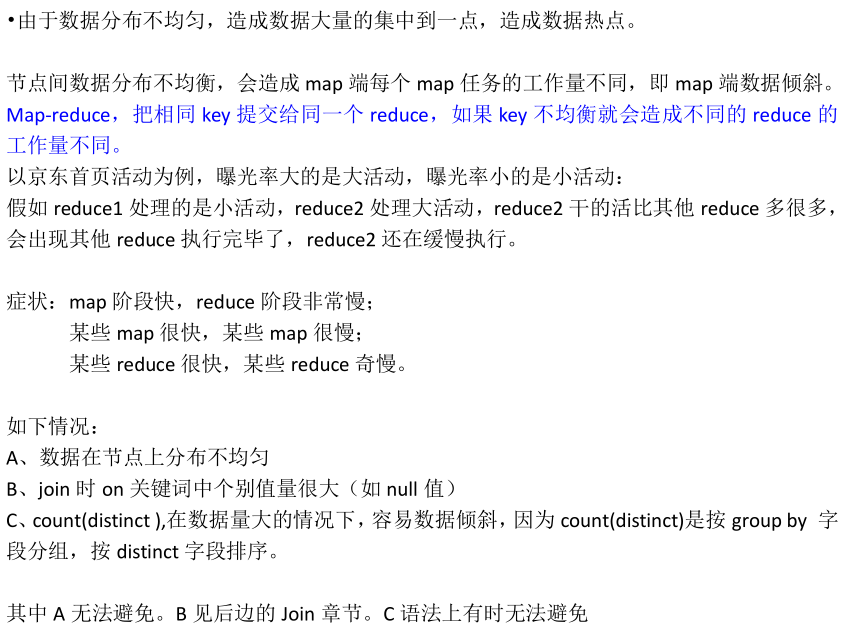
1. ABC三个hive表，每个表中只有一列int类型且列名相同，求三个表中互不重复的数，比如表明：T\_a T\_b T\_c每个列都叫做id（int类型）
2. 系统设计



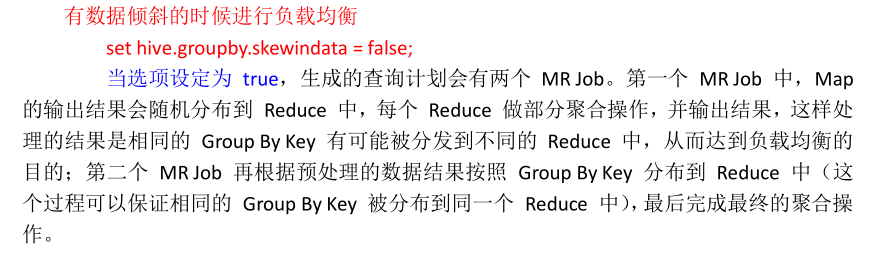




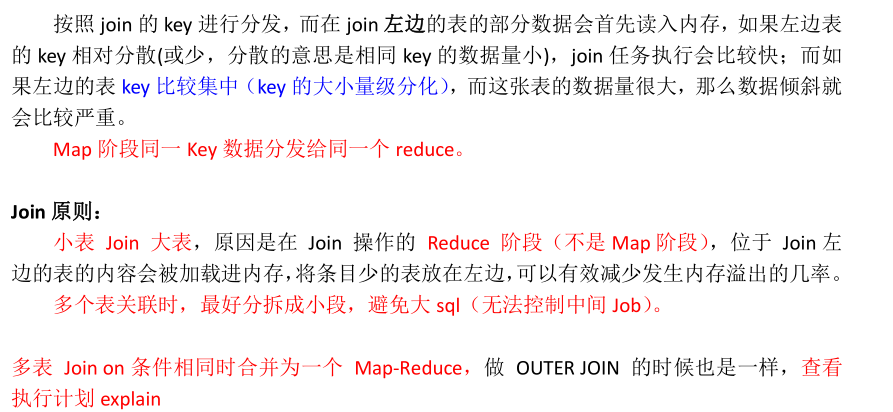
1. 什么是数据倾斜？hadoop框架的特性决定最怕数据倾斜



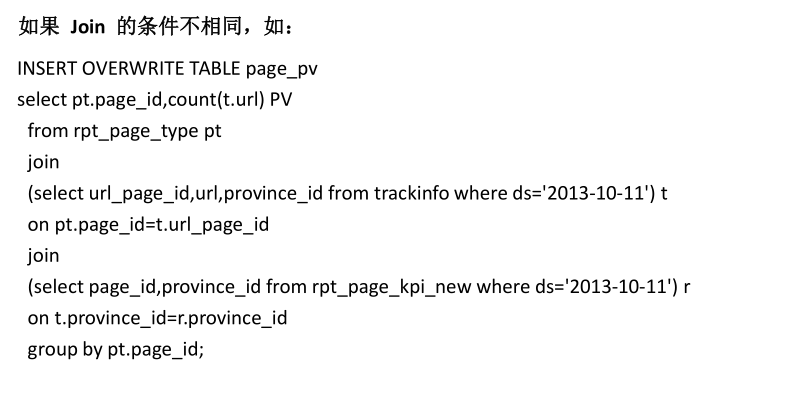
1. 如何解决数据倾斜？实际上没有办法避免，这里的解决只是个别情况起效

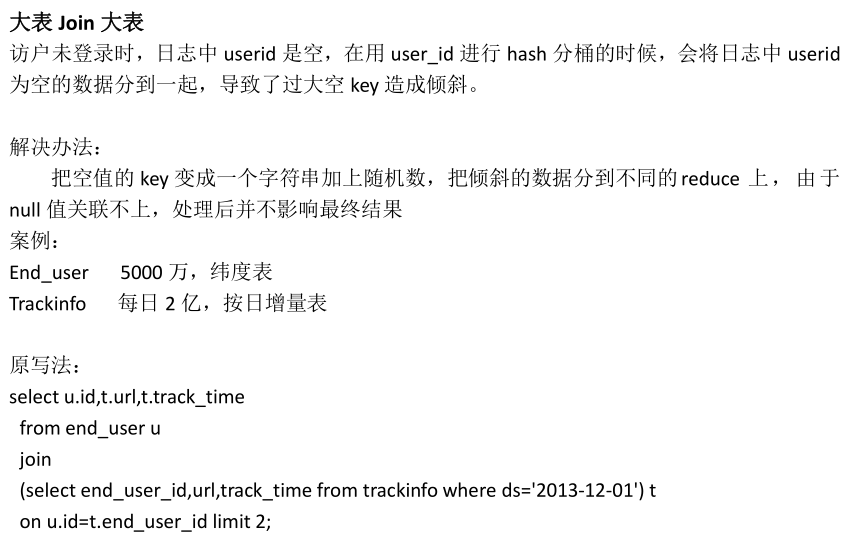


1. Hive中Join MapJoin Group by的特点

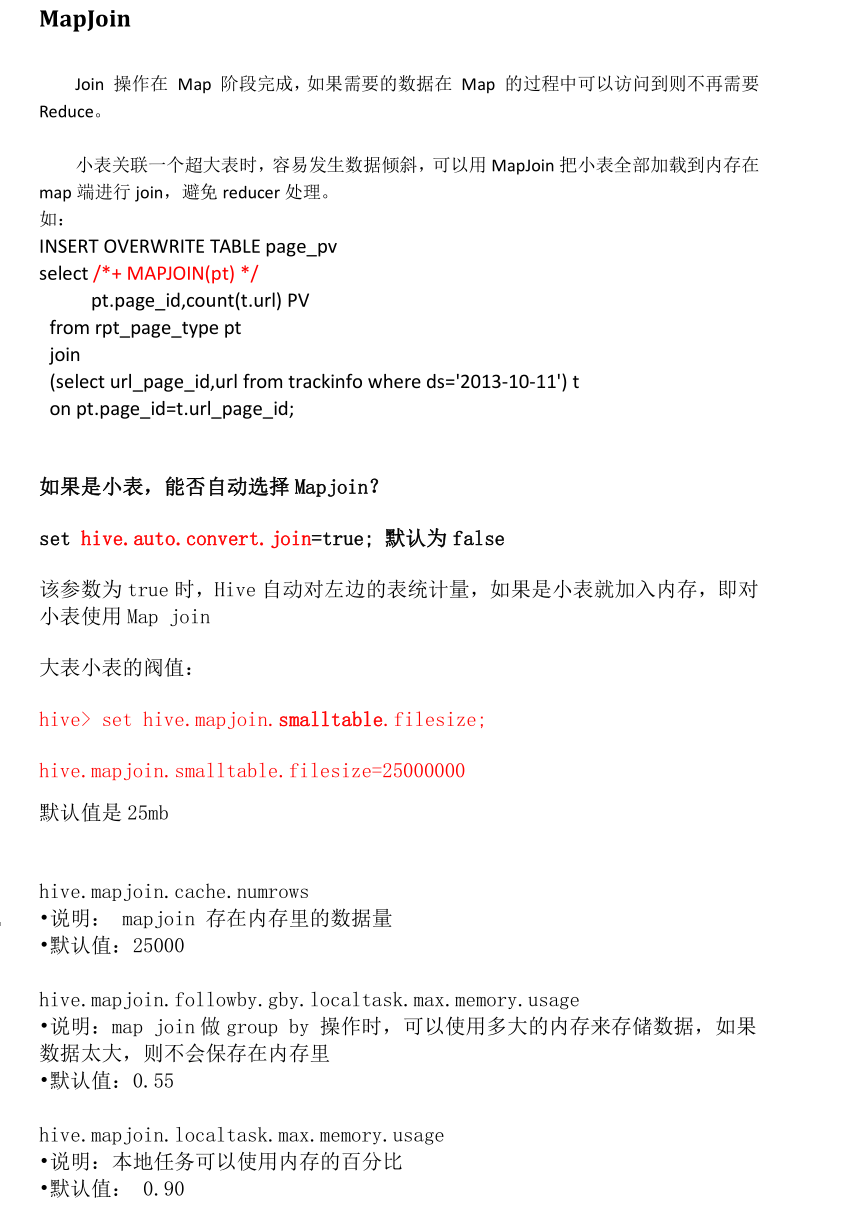


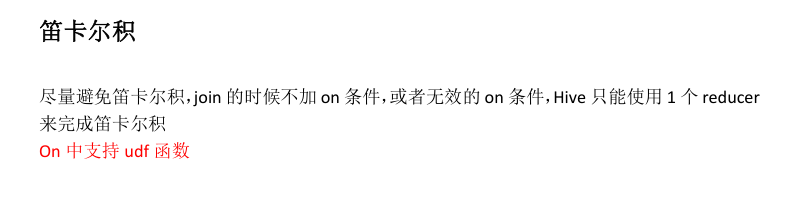


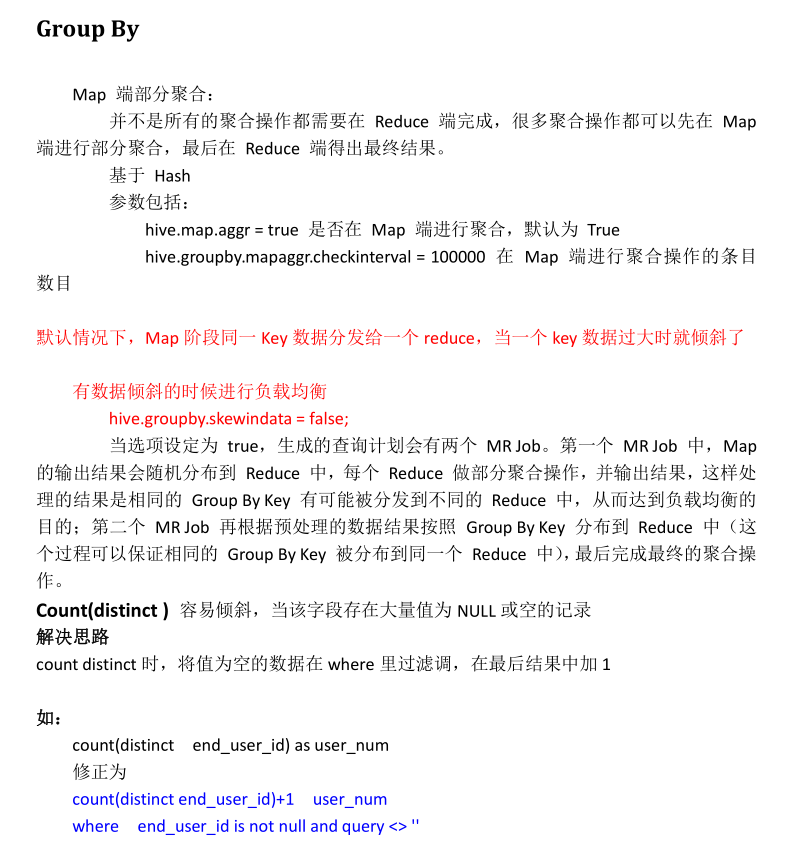












1. Hive 你们用的是外部表还是内部表,有没有写过UDF,hive 的版本
2. Hive语句实现WordCount

1.建表

2.分组（group by）统计wordcount

select word,count(1) from table1 group by word;

1. 共同朋友



思路：例如A，他的朋友是B\C\D\E\F\，那么BC的共同朋友就是A。所以将BC作为key，将A作为value，在map端输出即可！其他的朋友循环处理。

|  |
| --- |
| import java.io.IOException;  2. import java.util.Set;  3. import java.util.StringTokenizer;  4. import java.util.TreeSet;  5.  6. import org.apache.hadoop.conf.Configuration;  7. import org.apache.hadoop.fs.Path;  8. import org.apache.hadoop.io.Text;  9. import org.apache.hadoop.mapreduce.Job;  10. import org.apache.hadoop.mapreduce.Mapper;  11. import org.apache.hadoop.mapreduce.Reducer;  12. import org.apache.hadoop.mapreduce.Mapper.Context;  13. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  14. import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  15. import org.apache.hadoop.util.GenericOptionsParser;  16.  17. public class FindFriend {  18.  19. public static class ChangeMapper extends Mapper<Object, Text, Text,  Text>{  20. @Override  21. public void map(Object key, Text value, Context context) throws  IOException, InterruptedException {  22. StringTokenizer itr = new StringTokenizer(value.toString());  23. Text owner = new Text();  24. Set<String> set = new TreeSet<String>();  25. owner.set(itr.nextToken());  26. while (itr.hasMoreTokens()) {  27. set.add(itr.nextToken());  28.}  29. String[] friends = new String[set.size()];  30. friends = set.toArray(friends);  31.  32. for(int i=0;i<friends.length;i++){  33. for(int j=i+1;j<friends.length;j++){  34. String outputkey = friends[i]+friends[j];  35. context.write(new Text(outputkey),owner);  36. }  37. }  38. }  39. }  40.  41. public static class FindReducer extends Reducer<Text,Text,Text,Text>  {  42. public void reduce(Text key, Iterable<Text> values,  43. Context context) throws IOException,  InterruptedException {  44. String commonfriends ="";  45. for (Text val : values) {  46. if(commonfriends == ""){  47. commonfriends = val.toString();  48. }else{  49. commonfriends =  commonfriends+":"+val.toString();  50. }  51. }  52. ontext.write(key, new  Text(commonfriends));  53. }  54. }  55.  56.  57. public static void main(String[] args) throws IOException,  58. InterruptedException, ClassNotFoundException {  59.  60. Configuration conf = new Configuration();  61. String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();  62. if (otherArgs.length < 2) {  63. System.err.println("args error");  64. System.exit(2);  65. }  66. Job job = new Job(conf, "word count");  67. job.setJarByClass(FindFriend.class);  68. job.setMapperClass(ChangeMapper.class);  69. job.setCombinerClass(FindReducer.class);  70. job.setReducerClass(FindReducer.class);  71. job.setOutputKeyClass(Text.class);  72. job.setOutputValueClass(Text.class);  73. for (int i = 0; i < otherArgs.length - 1; ++i) {  74. FileInputFormat.addInputPath(job, new Path(otherArgs[i]));  75. }  76. FileOutputFormat.setOutputPath(job,  77. new Path(otherArgs[otherArgs.length - 1]));  78. System.exit(job.waitForCompletion(true) ? 0 : 1);  79.  80. }  81.  82. } |

结果：

|  |
| --- |
| 1. AB E:C:D  2. AC E:B  3. AD B:E  4. AE C:B:D  5. BC A:E  6. BD A:E  7. BE C:D:A  8. BF A  9. CD E:A:B  10. CE A:B  11. CF A  12. DE B:A  13. DF A  14. EF A |

1. 简要描述数据库中的null，说出null在hive底层如何存储，并解释select a.\* from t1 a left outer join t2 b on a.id=b.id where b.id is null；语句的含义。

null在hive底层默认是用"\N"来存储的，所以在sqoop到mysql之前需要将为null的数据加工成 其他字符，否则sqoop提示错误。

1. 写出hive中spilt、coalesce及collect\_list函数的用法（可举例）

split将字符串转化为数组

    coalesce(T v1,T v2,...) 返回参数中的第一个非空值；如果所有值都为null,那么返回null

    collect\_list列出该字段所有的值,不去重 select collect\_list(id) from table;

1. 写出text.txt文件放入hive中test表‘2017-12-12’分区的语句，test的分区字段是l\_date。

 load data local inpath '/a.txt' overwrite into table test partition(xx='xx')

1. hive中有一个表test，结构是create table arrays（x array<string>）,且有：

hive>select \* from test;

OK

[“a”,”b”]

[“c”,”d”, ”e”]]

（1）请写出select explode（x）as xx from test；语句执行后的结果。

（2）请写出select ‘xx’，sp from test lateral view explode（split（concat\_ws（‘，’，‘1’，‘2’），‘，’））a as sp；语句执行后的结果。

1. 解释一下什么是数据倾斜，并说明在hive中如何避免数据倾斜。

参数调节：

    hive.map.aggr=true

    hive.groupby.skewindata=true

有数据倾斜的时候进行负载均衡，当选项设定为true,生成的查询计划会有两个MR Job。第一个MR Job中，Map的输出结果集合会随机分布到Reduce中，每个Reduce做部分聚合操作，并输出结果，这样处理的结果是相同Group By Key有可能被分发到不同的Reduce中，从而达到负载均衡的目的；第二个MR Job在根据预处理的数据结果按照 Group By Key 分布到Reduce中(这个过程可以保证相同的 Group By Key 被分布到同一个Reduce中)，最后完成最终的聚合操作。

2>SQL语句调节：

   1)选用join key 分布最均匀的表作为驱动表。做好列裁剪和filter操作，以达到两表join的时候， 数据量相对变小的效果。

   2)大小表Join： 使用map join让小的维度表（1000条以下的记录条数）先进内存。在Map端 完成Reduce。

   3)大表Join大表：把空值的Key变成一个字符串加上一个随机数，把倾斜的数据分到不同的reduce 上，由于null值关联不上，处理后并不影响最终的结果。

   4)count distinct大量相同特殊值：count distinct时，将值为空的情况单独处理，如果是计算count distinct，可以不用处理，直接过滤，在做后结果中加1。如果还有其他计算，需要进行group by， 可以先将值为空的记录单独处理，再和其他计算结果进行union.

1. 学生表（students），有字段学号（id），姓名（name），性别（sex）等字段。课程表（classes），有字段课程标识（id），课程名称（class\_name）。学生选课分数表（sc），有字段学生表的学号（student\_id），课程标识（class\_id），分数（scores）。

（1）查询平均成绩大于60分的，且性别为“男”的同学的学号和平均成绩。

（2）x代表分数，x>=80表示优秀，x>=60&&x<80为及格，x<60为不及格等三个等级。请查询所有学生的姓名，课程名称及分数等级。

1. Hive是什么，以及适用场景，Hive与mysql关系，Hive与MapReduce关系，hive与Hbase的区别是？

基于hadoop的数据仓库工具，以hdfs方式存储，hive是面向分析的。Mysql关系型的、Hbase非关系都是面向存储的。Hive运行的就是mapreduce

1. Hive中创建表有哪几种方式，其区别是什么？

内部表：删除时，删除的时表结构，数据不没

外部表：删除数据也就没了

1. Hive中的UDF函数有哪几种，你写了哪些？

数学函数，字符串函数

字段大小写转换操作

1. 一个100G的文件，内存只有4G，对其进行全排序，如何用普通java程序编写处理。先把文件中的work。Hash（）% 100 分成100个小文件，work。Hash（）相同的会 进一个文件。再去求一个文件中每个word的次数。拿出每个文件最多的，全局比。
2. 用hive实现 select a.key,a.value from a where a.key not in (select b.key from b)Key不在b表中一样的
3. sort by ,order by ,cluster by ,distribute by 代表什么意思

order by：reduce阶段全局排序，所以自有一个reduce

sort by：map端输出会做排序

distribute by：按照指定的字段对数据进行划分，输入到不同的reduce

cluster by：具有distribute by 的功能外还兼具sort by 的功能

1. Hive文件压缩格式有哪些，压缩效率如何

开启压缩

set hive.exec.compress.output=true;

set mapred.output.compress=true;

set mapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec;

set io.compression.codec=org.apache.hadoop.io.compress.GzipCodec;

set mapred.output.compression.type=BLOCK;

TextFile (压缩方式Gzip,Bzip2压缩后不支持split)

SequenceFile

RCFile(存储方式：数据按行分块，每块按列存储。结合了行存储和列存储的优点)

ORCFile

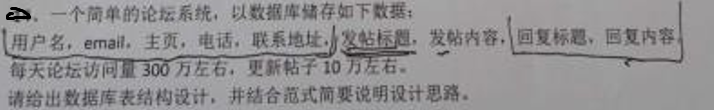
1. **Hive的分组方式**

row\_number() 是没有重复值的排序(即使两天记录相等也是不重复的),可以利用它来实现分页dense\_rank() 是连续排序,两个第二名仍然跟着第三名rank()       是跳跃排序的,两个第二名下来就是第四名

1. 数据库索引的创建、hive中join的问题
2. 为什么spark on hive 来使用
3. Hive 的 UDF

参考链接: http://blog.csdn.net/bitcarmanlee/article/details/51249260

1. Hiveserver2
2. Spark 的版本 hive 的版本 hadoop 版本
3. 数据库表结构设计



1. Hive存储元数据的方式及特点

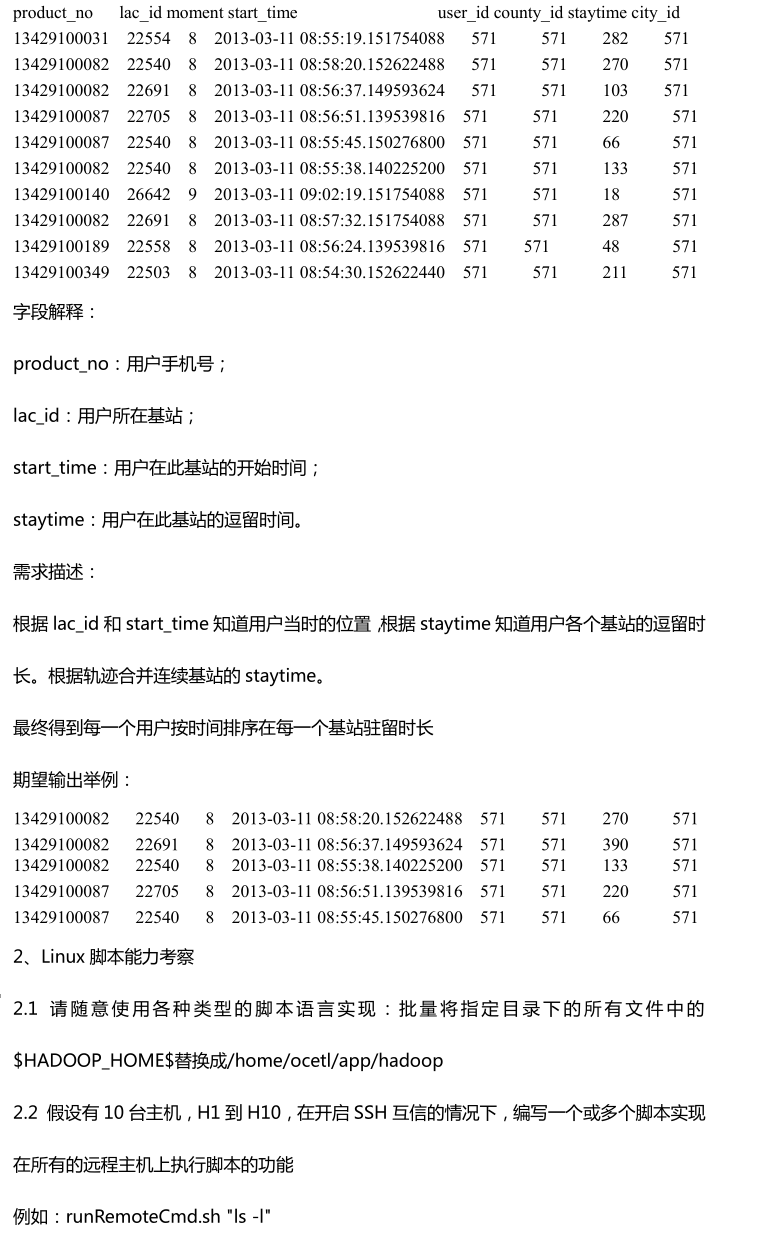
三种：内存数据库 derby，挺小，不常用 。本地 mysql。。常用，远程端 mysql。。不常用

上网上找了下专业名称：single user mode..multi user mode...remote user mode

1. 

Java 写 mapreduce 可以实现复杂的逻辑，如果需求简单，则显得繁琐。HiveQL 基本都是针对 hive 中的表数据进行编写，但对复杂的逻辑很难进行实现。写起来简单。

1. Hive 内部表和外部表的区别？
2. hive 底层与数据库交互原理
3. 使用 Hive 或者自定义 MR 实现如下逻辑



1. hive 如何权限控制？
2. hive 能像关系数据库那样，建多个库吗？
3. hive 实现统计的查询语句是什么？
4. 生产环境中为什么建议使用外部表？
5. hadoop mapreduce 创建类 DataWritable 的作用是什么？
6. 为什么创建类 DataWritable？
7. 如何实现统计手机流量？
8. 对比 hive 与 mapreduce 统计手机流量的区别？
9. 如今有 10 个文件夹,每个文件夹都有 1000000 个 url.如今让你找出top1000000url

不思考歪斜，功能，运用 2 个 job，第一个 job 直接用 filesystem 读取 10 个文件夹作为

map 输入，url 做 key，reduce 计算个 url 的 sum，下一个 job map 顶用 url 作 key，运用-sum 作二次排序，reduce 中取 top10000000第二种方法，建 hive 表 A，挂分区 channel，每个文件夹是一个分区.select x.url,x.c from(select url,count(1) as c from A where channel ='' group byurl)x order by x.c desc limie 1000000;

1. hive 的物理模型跟传统数据库的不同
2. hive 底层与数据库交互原理

Hive和Hbase有各自不同的特征：hive是高延迟、结构化和面向分析的，hbase是低延迟、非结构化和面向编程的。Hive数据仓库在hadoop上是高延迟的。Hive集成Hbase就是为了使用hbase的一些特性。

        Hive集成HBase可以有效利用HBase数据库的存储特性，如行更新和列索引等。在集成的过程中注意维持HBase jar包的一致性。Hive集成HBase需要在Hive表和HBase表之间建立映射关系，也就是Hive表的列(columns)和列类型(column types)与HBase表的列族(column families)及列限定词(column qualifiers)建立关联。每一个在Hive表中的域都存在于HBase中，而在Hive表中不需要包含所有HBase中的列。HBase中的RowKey对应到Hive中为选择一个域使用:key来对应，列族(cf:)映射到Hive中的其它所有域，列为(cf:cq)。

1. 有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 字节，内存限制大小是 1M。返回频数最高的 100 个词。

方案：顺序读文件中，对于每个词 x，取 hash(x)%5000，然后按照该值存到 5000 个小文

件（记为 x0,x1,…x4999）中。这样每个文件大概是 200k 左右。如果其中的有的文件超过了 1M 大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过 1M。对每个小文件，统计每个文件中出现的词以及相应的频率（可以采用 trie 树/hash\_map 等），并取出出现频率最大的 100 个词（可以用含 100 个结 点的最小堆），并把 100 个词及相应的频率存入文件，这样又得到了 5000 个文件。下一步就是把这 5000 个文件进行归并（类似与归并排序）的过程了。

1. hive 中的压缩格式 RCFile、TextFile、SequenceFile 各有什么区别？以上 3 种格式一样大的文件哪个占用空间大小
2. 你们的 hive 处理数据能达到的指标是多少？
3. Hive分桶

在实际生产环境中，每一个表都可能存储很多的数据，如果直接进行join，通过笛卡尔积会造成大量的shuffle溢出，会延迟job完成时间甚至内存不足的想象，那么就可以对两个表先进行分桶，然后在join。这样就可以大大减少笛卡尔积

1. Hive内部表和外部表的区别：

1、导入数据的时候：

外部表是复制，内部表是剪切

2、删除表的时候：

外部表只删除元数据，数据仍然保留

内部表会删除元数据和数据

1. Hive中的UDF

先开发一个java类，继承UDF，并重载evaluate方法

1. 常用的hive分析函数

1、row\_number:行号递增

2、rank：行号递增，如果有相等的，相当的部分行号一样。然后下面直接递增N

3、dense\_rank：行号递增，相等的行号仍然递增

1. join中出现数据倾斜优化

hive.optimize.skewjoin=true;

如果join过程中出现了数据倾斜，应该设置为true

set hive.skewjoin.key=100000;

 这个是join的键对应的记录条数超过这个值则会进行优化，优化措施：正常是只有一个job的，优化后会有两个job。当数据量达到100000以上的时候，hive会在启动一个job，然后将原有的数据的key加上的一个随机数，将数据打乱。这样数据就会分不到不同的节点上计算。在原打乱的基础上在做一次计算，然后启动一个job，恢复原来的key，在做一次计算。

1. Hive中的mapjoin

就是将小表加载到缓存中，这样所有的节点都可以访问到这个小表

开启方式：

set hive.auto.convert.join=true;

hive.mapjoin.smalltable.filesize默认值是25Mb

这个是自动判断的，当文件小于25M的时候，自动启动

或者这样使用：手动

select /\*+ mapjoin(A)\*/ f.a,f.b from A t join B f  on ( f.a=t.a and f.ftime=20110802)

其中A表是小表，B表是大表，这样让join发生在map端

mapjoin的使用场景：

1、关联操作中有一张小表

2、不等值的链接操作

mapjoin最好是手动操作

1. hive bucket join

1、两个表以相同的方式划分桶

2、两个表的桶的个数是倍数关系

create table order(cid int , price float) clustered by (cid) into 32 buckets;

create table customer(id int , first string) clustered by(id) into 32 buckets;

select price from order t join customer s on t.cid=s.id;

1. 关于where和join的优化

优化前:

select o\* from order o join customer c on o.cid=c.id where o.time='2017-01-01'

优化后：

select o.\* from

(select cid from order where time='2016-01-01')o

join customer c on o.cid=c.id;

优化前是先join后再通过where进行过滤，这样并没有减轻reduce的压力。

优化后是先在map端执行where，过滤数据，然后在join。这样就会降低计算量

1. Hive group by优化

1、hive.groupby.skewindata=true;

如果group by过程出现倾斜，应该设置为true

2、set hive.groupby.mapagger.checkinterval=100000;

group对应的键对应的记录条数超过这个值则会进行优化

这个仍然是启动两个job

1. hive的表优化

1、分区

‘表’相当于一个大目录，分区就是在这个大目录下面的小目录

分区包括静态分区和动态分区

静态分区：就是在建表的时候指定分区的

如果数据量量多，一个分区需要一个insert，就会很麻烦，所以可以使用动态分区

动态分区：

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

默认值：strict

描述：strict是避免全分区字段是动态的，必须有至少一个分区字段是指定有值的

1. hive job的优化 并行化执行

hive执行过程中的job是按照默认的顺序来执行的，如果没有太大的依赖关系，最好并行执行，减少执行的时间，每个查询被hive转化成多个阶段，有些阶段关联性不大，则可以并行执行，减少执行时间。

set hive.exec.parallel=true;

set hive.exec.parallel.thread.numbe=16（默认8）;

1. hive job的优化本地化执行

set hive.exec.mode.local.auto=true;

当一个job满足如下条件的时候才能真正使用本地模式；

1、job的输入数据大小必须小于参数：

 hive.exec.mode.local.auto.inputbytes.max（默认128M）

2、job的map数必须小于参数：

hive.exec.mode.local.auto.tasks.max（默认4）

3、job的reduce数量必须是0或者1

1. hive job的优化job合并输入小文件

在集群中面临这样的问题，集群中很多的小文件，这样会启动很多的（FileInputFormat会将输入文件分割成split。split的个数决定了map的个数），而且这些小文件的执行时间特别短，造成集群的资源没有良好的利用

解决：

set hive.input.format=oar.apache.hadoop.hive.ql.io.CombineHiveInputFormat

这样做后，就会把多个split分片合并成一个。合并的文件数由mapred.max.split.size限制的大小决定

1. hive job的优化 job合并输出小文件

set hive.merge.smallfiles.avgsize=256000000;当输出文件平均大小小于该值，启动新job合并文件set hive.merge.size.per.task=64000000;设定合并之后的文件大小

1. hive job的优化JVM重利用

JVM重用可以使得JVM实例在同一个JOB中重新使用N次

set mapred.job.reuse.jvm.num.tasks=10;

JVM重利用对hive的性能有很大的影响，特别是对小文件的场景或者task特别多的场景，可以有效减少执行时间。

当然，这个值不能设置过大，因为有一些job是有reduce任务的。如果reduce任务没有完成，map任务占用的资源不能都饿到释放，这样其他作业就可能处于等待

1. hive job的优化压缩数据

中间压缩就是处理hive查询的多个job之间的数据

中间压缩，减少网络传输的数据量，中间压缩，不需要压缩效果特别好，选择一个节省CPU耗时的压缩方式

set hive.exec.compress.intermediate=true;

set hive.intermedia.compression.codec=org.apache.hadoop.io.compress.SnappyCode;

set hive.intermedia.compression.type=BLOCK;（按照块进行压缩）

hive查询最终输出的压缩

这个阶段可以选择一个压缩效果比较明显的 , 这样可以降低集群存储的数据量，占用较小的空间

set hive.exec.compress.ouput=true;

set mapred.output.compresssion.codec=org.apache.hadoop.io.compress.GzipCodec;

set hive.intermedia.compression.type=BLOCK;

1. Hive中的order by，sort by ， distribute by ， cluster by的区别

Order by

会将所有的数据在一个reducer上执行，得到的结果是整体有序的。但是由于不能并发执行，所以效率比较低。

sort by

排序操作在多个reducer上执行，多以只能保证局部有序。无法保证整体有序

而且，使用sort by 有多个reduce的情况下，可能会造成reduce处理的数据范围可以重叠

distribute by

可以保证每个reduce处理数据范围不重叠。但是不负责排序

cluster by

Cluster by = distribute by + sort by

1. Hive 数据块取样

1、按照数据块取样-----根据百分比取样

SELECT \* FROM tableA TABLESAMPLE (50 PERCENT);

2、按照数据块取样-----指定取样大小

SELECT \* FROM tableA TABLESAMPLE (30M);

3、按照数据块取样-----指定函数取样

这里指定的行数，每个Map中都取样n ROWS

SELECT \* FROM tableA TABLESAMPLE (200 ROWS)

1. Hive 分桶取样

语法：

table\_sample: TABLESAMPLE (BUCKET x OUT OF y [ON colname])

其中x是要抽样的桶编号，桶编号从1开始，colname表示抽样的列，y表示桶的数量。

1、假如当前的表没有分过桶

SELECT count(1)

FROM tableA TABLESAMPLE (BUCKET 1 OUT OF 10 ON name);

将tableA 中的数据随机分成了10个桶，抽取第一个桶

2、当然，如果试先已经分过桶了，我可以这样做，效率会更高一点

仍然是试先分好了10个桶

正常的执行方式：

SELECT count(1)

FROM tableA TABLESAMPLE (BUCKET 1 OUT OF 10 ON name);

但是如果这样执行：

SELECT count(1)

FROM tableA TABLESAMPLE (BUCKET 1 OUT OF 20 ON name);

就会在第一个桶里面只抽取一半的数据

分桶和未分桶的抽样区别：

已经分桶的表抽样，查询只会扫描相应桶中的数据;

未分桶表的抽样，查询时候需要扫描整表数据，先分桶，再抽样

1. insert into 和 override write区别

insert into：将某一张表中的数据写到另一张表中

override write：覆盖之前的内容

1. order by与sort by的区别

a.hive中的ORDER BY语句和关系数据库中的sql语法相似。他会对查询结果做全局排序，这意味着所有的数据会传送到一个Reduce任务上，这样会导致在大数量的情况下，花费大量时间。

b.SORT BY不是全局排序，其在数据进入reducer前完成排序，因此在有多个reduce任务情况下，SORT BY只能保证每个reduce的输出有序，而不能保证全局有序 \*hive中通过set mapred.reduce.tasks=3;来设定reduce的数量\*

c.一点小知识：DISTRIBUTE BY可以按指定字段将数据划分到不同的reduce中当DISTRIBUTE BY的字段和SORT BY的字段相同时，可以用CLUSTER BY来代替 DISTRIBUTE BY with SORT BY

1. hive分区 如何将数据定义到哪一个分区中

Create table logs(ts bigint,line string) Partitioned by (dt string,country string);

Load data local inpath ‘/home/hadoop/par/file01.txt’ into table logs partition (dt=’ 2012-06-02’,country=’cn’);

1. HIVE 优化？

1.输出小文件合并

hive.merge.smallfiles.avgsize 设为5000000

增加map数量，可提高hive运行速度

set mapred.reduce.tasks=10;

2.map join

大小表join时通过使用hint的方式指定join时使用mapjoin。

/\*+ mapjoin(小表)\*/

3.hive索引

4.优先过滤数据，减少每个阶段的数据量，对分区表加以分区，同时只选择需要使用的字段

5.根据不同的使用目的优化使用方法

6.尽量原子化操作，尽量避免一个sql包含复杂逻辑

7.join操作小表放在join的左边

8.如果union all的部分个数大于2，或者union部分数据量大，应拆分成多个insert into语句。

1.参数优化，小于6M自动合并

2.加功能，改成分区表，做join写成任务流

3.mapjoin

4.加索引

5.先where 再join

6.加小型的sql

1. hive有哪些方式保存元数据，各有哪些特点

hive的数据模型包括：database、table、partition和bucket。

1.Database：相当于关系数据库里的命名空间（namespace），它的作用是将用户和数据库的应用隔离到不同的数据库或模式中

2.表（table）：hive的表逻辑上由存储的数据和描述表格中的数据形式的相关元数据组成。

Hive里的表友两种类型一种叫托管表，这种表的数据文件存储在hive的数据仓库里，一种叫外部表，这种表的数据文件可以存放在hive数据仓库外部的分布式文件系统上，也可以放到hive数据仓库里（注意：hive的数据仓库也就是hdfs上的一个目录，这个目录是hive数据文件存储的默认路径，它可以在hive的配置文件里进行配置，最终也会存放到元数据库里）。

3.分区（partition）：hive里分区的概念是根据“分区列”的值对表的数据进行粗略划分的机制，在hive存储上就体现在表的主目录（hive的表实际显示就是一个文件夹）下的一个子目录，这个文件夹的名字就是我们定义的分区列的名字，没有实际操作经验的人可能会认为分区列是表的某个字段，其实不是这样，分区列不是表里的某个字段，而是独立的列，我们根据这个列存储表的里的数据文件。

4.桶（bucket）：上面的table和partition都是目录级别的拆分数据，bucket则是对数据源数据文件本身来拆分数据。

1. hive与hbase的区别

hive：

1.数据保存在hdfs上，以hdfs格式保存数据，映射为hive中的表结构

2.支持sql语言，调用MR

3.不能做实时操作

4.相对数据量较小

Hbase：

1.Hbase自己的存储机构

2.不支持sql语言，不调用MR

3.可以支持实时操作

4.相对数据量大，对于反复使用的数据比较适用

1. hive SQL语句中 select from where group by having order by 的执行顺序？

from--where--group by--having--select--order by,

from:需要从哪个数据表检索数据

where:过滤表中数据的条件

group by:如何将上面过滤出的数据分组

having:对上面已经分组的数据进行过滤的条件

select:查看结果集中的哪个列，或列的计算结果

order by :按照什么样的顺序来查看返回的数据

1. 有文件dim\_city.txt 如何加载dim\_city表中

load data local inpath"./dim\_city.txt" insert into table dim\_city;

1. hive如何将下表table\_1中的数据

col1 col2 col3

a b 1,2,3

c d 4,5,6

变为：

col1 col2 col 3

a b 1

a b 2

a b 3

c d 4

c d 5

c d 6

create table table\_1(col1 string,col2 string,col3 string)

select col1,col2,name from table\_1 LATERAL VIEW explode(split(col3,',')) col3 as name;

1. HIVE数据库与ORACLE数据库有什么区别,目前HIVE数据库不支持哪些函数?

a. 查询语言。由于 SQL 被广泛的应用在数据仓库中，因此，专门针对 Hive 的特性设计了类 SQL 的查询语言 HQL。熟悉 SQL 开发的开发者可以很方便的使用 Hive 进行开发。

b. 数据存储位置。Hive 是建立在 Hadoop 之上的，所有 Hive 的数据都是存储在 HDFS 中的。而数据库则可以将数据保存在块设备或者本地文件系统中。

c. 数据格式。Hive 中没有定义专门的数据格式，数据格式可以由用户指定，用户定义数据格式需要指定三个属性：列分隔符（通常为空格、”\t”、”\x001″）、行分隔符 （”\n”）以及读取文件数据的方法（Hive 中默认有三个文件格式 TextFile，SequenceFile 以及 RCFile）。由于在加载数据的过程中，不需要从用户数据格式到 Hive 定义的数据格式的转换，因此，Hive 在加载的过程中不会对数据本身进行任何修改，而只是将数据内容复制或者移动到相应的 HDFS 目录中。而在数据库中，不同的数据库有不同的存储引擎，定义了自己的数据格式。所有数据都会按照一定的组织存储，因此，数据库加载数据的过程会比较耗时。

d. 数据更新。由于 Hive 是针对数据仓库应用设计的，而数据仓库的内容是读多写少的。因此，Hive 中不支持对数据的改写和添加，所有的数据都是在加载的时候中确定好的。而数据库中的数据通常是需要经常进行修改的，因此可以使用 INSERT INTO … VALUES 添加数据，使用 UPDATE … SET 修改数据。

e. 索引。之前已经说过，Hive 在加载数据的过程中不会对数据进行任何处理，甚至不会对数据进行扫描，因此也没有对数据中的某些 Key 建立索引。Hive 要访问数据中满足条件的特定值时，需要暴力扫描整个数据，因此访问延迟较高。由于 MapReduce 的引入， Hive 可以并行访问数据，因此即使没有索引，对于大数据量的访问，Hive 仍然可以体现出优势。数据库中，通常会针对一个或者几个列建立索引，因此对于少量的特定条件的数据的访问，数据库可以有很高的效率，较低的延迟。由于数据 的访问延迟较高，决定了 Hive 不适合在线数据查询。

f. 执行。Hive 中大多数查询的执行是通过 Hadoop 提供的 MapReduce 来实现的（类似 select \* from tbl 的查询不需要 MapReduce）。而数据库通常有自己的执行引擎。

g. 执行延迟。之前提到，Hive 在查询数据的时候，由于没有索引，需要扫描整个表，因此延迟较高。另外一个导致 Hive 执行延迟高的因素是 MapReduce 框架。由于 MapReduce 本身具有较高的延迟，因此在利用 MapReduce 执行 Hive 查询时，也会有较高的延迟。相对的，数据库的执行延迟较低。当然，这个低是有条件的，即数据规模较小，当数据规模大到超过数据库的处理能力的时 候，Hive 的并行计算显然能体现出优势。

h. 可扩展性。由于 Hive 是建立在 Hadoop 之上的，因此 Hive 的可扩展性是和 Hadoop 的可扩展性是一致的（世界上最大的 Hadoop 集群在 Yahoo!，2009年的规模在 4000 台节点左右）。而数据库由于 ACID 语义的严格限制，扩展行非常有限。目前最先进的并行数据库 Oracle 在理论上的扩展能力也只有 100 台左右。

i. 数据规模。由于 Hive 建立在集群上并可以利用 MapReduce 进行并行计算，因此可以支持很大规模的数据；对应的，数据库可以支持的数据规模较小。

Hive不支持的函数：decode、rownum、to\_char、replace、||、nvl、months\_between、add\_months、rollup、cube、rank() over、dense\_rank() over、row\_number() over

1. 使用如下示例数据及数据说明情况,分别实现(1)该数据在HIVE库中建表,(2)数据导入到所建表中,(3)使用所建数据表,使用HQL统计2014-12-31账期手机用户上网总流量.数据说明 文件为test.txt字段分割符为|
2. 编写HIVE自定义函数实现ORACLE数据库中的addmonths函数功能,然后封装到HIVE函数库中

addmonths(data a,int b)函数功能简单说明,求传入日期a经过B月后的日期是多少?

1. Hive的条件判断有几种？

hive 的条件判断（if、coalesce、case）

1. 请适用hive写出查询某网站日志中方位多页面a和页面b的用户数量的语句：

Select count(user) from urla a , urlb b where a.url = b.url ;

### 海量数据处理方法

1. 海量日志数据，提取出某日访问百度次数最多的那个 IP

此题，在我之前的一篇文章算法里头有所提到，当时给出的方案是：

IP 的数目还是有限的，最多 2^32个，

所以可以考虑使用 hash 将 ip 直接存入内存，然后进行统计。

再详细介绍下此方案：首先是这一天，并且是访问百度的日志中的 IP 取出来，逐个写入到一个大文件中。

注意到 IP 是 32 位的，最多有个 2^32 个 IP。

同样可以采用映射的方法，比如模 1000，把整个大文件映射为 1000 个小文件，再找出每个小文中出现频率最大的 IP（可以采用 hash\_map 进行频率统计，

然后再找出频率最大的几个）及相应的频率。然后再在这 1000 个最大的 IP 中，找出那个频率最大的 IP，即为所求。

1. 搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为1-255 字节。

假设目前有一千万个记录（这些查询串的重复度比较高，虽然总数是 1 千万，但如果除去重复后，不超过 3 百万个。

一个查询串的重复度越高，说明查询它的用户越多，也就是越热门。），请你统计最热门的10 个查询串，要求使用的内存不能超过 1G。

典型的 Top K 算法，还是在这篇文章里头有所阐述。

文中，给出的最终算法是：第一步、先对这批海量数据预处理，在 O（N）的时间内用 Hash 表完成排序；然后，第二步、借助堆这个数据结构，找出 TopK，时间复杂度为 N‗logK。 即，借助堆结构，我们可以在 log 量级的时间内查找和调整/移动。因此，维护一个 K(该题目中是 10)大小的小根堆，然后遍历 300 万的 Query，分别和根元素进行对比所以，我们最终的时间复杂度是：O（N） + N'\*O（logK），（N 为 1000 万，N‘为 300 万）。ok，更多，详情，请参考原文。或者：采用 trie 树，关键字域存该查询串出现的次数，没有出现为 0。最后用 10 个元素的最小推来对出现频率进行排序。

1. 有一个 1G 过 大小的一个文件，里面每一行是一个词，词的大小不超过 16 字节，内存限制大小是1M 。返回频数最高的 100 个词。

方案：顺序读文件中，对于每个词 x，取 hash(x)%5000，然后按照该值存到 5000 个小文件（记为x0,x1,...x4999）中。这样每个文件大概是 200k 左右。如果其中的有的文件超过了 1M 大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过 1M。 对每个小文件，统计每个文件中出现的词以及相应的频率（可以采用 trie 树/hash\_map等），并取出出现频率最大的 100 个词（可以用含 100 个结点的最小堆），并把 100 个词及相应的频率存入文件，这样又得到了 5000 个文件。下一步就是把这 5000 个文件进行归并（类似与归并排序）的过程了。

1. 有 10 个文件 件，每个文件 1G ，每个文件的每一行存放的都是的 query ，每个文件的 query都可能重复。要求你按照 query 的频度排序。

还是典型的 TOP K 算法，解决方案如下：

方案 1： 顺序读取 10 个文件，按照 hash(query)%10 的结果将 query 写入到另外 10 个文件（记为）中。这样新生成的文件每个的大小大约也 1G（假设 hash 函数是随机的）。 找一台内存在 2G 左右的机器，依次对用 hash\_map(query, query\_count)来统计每个 query 出现的次数。利用快速/堆/归并排序按照出现次数进行排序。将排序好的 query 和对应的 query\_cout 输出到文件中。这样得到了 10 个排好序的文件（记为）。对这 10 个文件进行归并排序（内排序与外排序相结合）。

方案 2： 一般 query 的总量是有限的，只是重复的次数比较多而已，可能对于所有的 query，一次性

就可以加入到内存了。这样，我们就可以采用 trie 树/hash\_map 等直接来统计每个 query 出现的次数，然

后按出现次数做快速/堆/归并排序就可以了。

方案 3： 与方案 1 类似，但在做完 hash，分成多个文件后，可以交给多个文件来处理，采用分布式

的架构来处理（比如 MapReduce），最后再进行合并。

1. 给定 a.b 两个文件，各存放 50 亿个 url ，每个 url 各占 64 字节，内存限制是 4G ，让你找出 a .b 文件共同的 url ？

方案 1：可以估计每个文件安的大小为 5G×64=320G，远远大于内存限制的 4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。遍历文件 a，对每个 url 求取 hash(url)%1000，然后根据所取得的值将 url 分别存储到 1000 个小文件（记为 a0,a1,...,a999）中。这样每个小文件的大约为 300M。遍历文件 b，采取和 a 相同的方式将 url 分别存储到 1000 小文件（记为 b0,b1,...,b999）。这样处理后，所有可能相同的 url 都在对应的小文件（a0vsb0,a1vsb1,...,a999vsb999）中，不对应的小文件不可能有相同的 url。然后我们只要求出 1000 对小文件中相同的 url 即可。求每对小文件中相同的 url 时，可以把其中一个小文件的 url 存储到 hash\_set 中。然后遍历另一个小文件的每个 url，看其是否在刚才构建的 hash\_set 中，如果是，那么就是共同的 url，存到文件里面就可以了。

方案 2：如果允许有一定的错误率，可以使用 Bloom filter，4G 内存大概可以表示 340 亿 bit。将其中一个文件中的url使用Bloom filter映射为这340亿bit，然后挨个读取另外一个文件的url，检查是否与Bloomfilter，如果是，那么该 url 应该是共同的 url（注意会有一定的错误率）。Bloom filter 日后会在本 BLOG 内详细阐述。

1. 在 2.5 亿个整数中找出不重复的整数，注，内存不足以容纳这 2.5 亿个整数

方案 1：采用 2-Bitmap（每个数分配 2bit，00 表示不存在，01 表示出现一次，10 表示多次，11 无意义）进行，共需内存内存，还可以接受。然后扫描这 2.5 亿个整数，查看 Bitmap 中相对应位，如果是 00变 01，01 变 10，10 保持不变。所描完事后，查看 bitmap，把对应位是 01 的整数输出即可。

方案 2：也可采用与第 1 题类似的方法，进行划分小文件的方法。然后在小文件中找出不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

1. 一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前 10 个词，请给出思想，给出时间复杂度分析。

方案 1：这题是考虑时间效率。用 trie 树统计每个词出现的次数，时间复杂度是 O(n\*le)（le 表示单词的平准长度）。然后是找出出现最频繁的前 10 个词，可以用堆来实现，前面的题中已经讲到了，时间复杂度是 O(n\*lg10)。所以总的时间复杂度，是 O(n\*le)与 O(n\*lg10)中较大的哪一个。附、100w 个数中找出最大的 100 个数。在前面的题中，我们已经提到了，用一个含100个元素的最小堆完成。复杂度为O(100w\*lg100)。

方案 2：采用快速排序的思想，每次分割之后只考虑比轴大的一部分，知道比轴大的一部分在比 100多的时候，采用传统排序算法排序，取前 100 个。复杂度为 O(100w\*100)。

方案 3：采用局部淘汰法。选取前 100 个元素，并排序，记为序列 L。然后一次扫描剩余的元素 x，与排好序的 100 个元素中最小的元素比，如果比这个最小的要大，那么把这个最小的元素删除，并把 x 利用插入排序的思想，插入到序列 L 中。依次循环，知道扫描了所有的元素。复杂度为 O(100w\*100)。

1. 有一千万条短信，有重复，以文本文件的形式保存，一行一条，有重复。 请用 5 分钟时间，找出重复出现最多的前 10 条。

分析：常规方法是先排序，在遍历一次，找出重复最多的前 10 条。但是排序的算法复杂度最低为 nlgn。

可以设计一个 hash\_table, hash\_map<string, int>，依次读取一千万条短信，加载到 hash\_table 表中，并且统计重复的次数，与此同时维护一张最多 10 条的短信表。这样遍历一次就能找出最多的前 10 条，算法复杂度为 O(n)。

实现如下：

1. #include<iostream>

2. #include<map>

3. #include<iterator>

4. #include<stdio.h>

5. using namespace std;

6.

7. #define HASH \_\_gnu\_cxx

8. #include<ext/hash\_map>

9. #define uint32\_t unsigned int

10. #define uint64\_t unsigned long int

11. struct StrHash

12. {

13. uint64\_t operator()(const std::string& str) const

14. {

15. uint32\_t b = 378551;

16. uint32\_t a = 63689;

17. uint64\_t hash = 0;

18.

19. for(size\_t i = 0; i < str.size(); i++)

20. {

21. hash = hash \* a + str[i];

22. a = a \* b;

23. }

24.

25. return hash;

26. }

27. uint64\_t operator()(const std::string& str, uint32\_t field) const

28. {

29. uint32\_t b = 378551;

30. uint32\_t a = 63689;

31. uint64\_t hash = 0;

32. for(size\_t i = 0; i < str.size(); i++)

33. {

34. hash = hash \* a + str[i];

35. a = a \* b;

36. }

37. hash = (hash<<8)+field;

38. return hash;

39. }

40. };

41. struct NameNum{

42. string name;

43. int num;

44. NameNum():num(0),name(""){}

45. };

46. int main()

47. {

48. HASH::hash\_map< string, int, StrHash > names;

49. HASH::hash\_map< string, int, StrHash >::iterator it;

50. NameNum namenum[10];

51. string l = "";

52. while(getline(cin, l))

53. {

54. it = names.find(l);

55. if(it != names.end())

56. {

57. names[l] ++;

58. }

59. else

60. {

61. names[l] = 1;

62. names[l] = 1;

63. }

64. }

65. int i = 0;

66. int max = 1;

67. int min = 1;

68. int minpos = 0;

69. for(it = names.begin(); it != names.end(); ++ it)

70. {

71. if(i < 10)

72. {

73. namenum[i].name = it->first;

74. namenum[i].num = it->second;

75. if(it->second > max)

76. max = it->second;

77. else if(it->second < min)

78. {

79. min = it->second;

80. minpos = i;

81. }

82. }

83. else

84. {

85. if(it->second > min)

86. {

87. namenum[minpos].name = it->first;

88. namenum[minpos].num = it->second;

89. int k = 1;

90. min = namenum[0].num;

91. minpos = 0;

92. while(k < 10)

93. {

94. if(namenum[k].num < min)

www.aboutyun.com

95. {

96. min = namenum[k].num;

97. minpos = k;

98. }

99. k ++;

100. }

101. }

102. }

103. i++;

104.

105. }

106. i = 0;

107. cout << "maxlength (string,num): " << endl;

108. while( i < 10)

109. {

110. cout << "(" << namenum[i].name.c\_str() << "," << namenum[i].num << ")" << endl;

111. i++;

112. }

113. return 0;

114. }

使用 g++ 编译如下：

g++ main.cpp -o main

短信文本文件为：msg.txt

运行：./main < msg.txt

输出结果为：

maxlength (string,num):

(点点母婴坊,4)

(农机配件维修,5)

(红胜超市,6)

(龙溪大酒店,8)

(张记饺子馆,3)

(友谊旅店,3)

(明珠通讯,3)

(金源旅馆,3)

(洞庭山天然泉水,2)

(清源超市,3)

1. 有20亿条数据，每条数据的大小在1K-1M不等，每条数据有一个唯一的u\_int64的id，请设计一个读取数据系统，能根据id获取数据，要求：内存有限制，16G,尽可能利用内存资源，尽可能高效的获取数据，可利用磁盘，磁盘容量不受限制（写出主要设计思路）
2. 总结

**首先处理大数据的面试题，有些基本概念要清楚：**

（1）1Gb = 109bytes（1Gb = 10亿字节）：1Gb = 1024Mb，1Mb = 1024Kb，1Kb = 1024bytes；

（2）基本流程是，分解大问题，解决小问题，从局部最优中选择全局最优；（当然，如果直接放内存里就能解决的话，那就直接想办法求解，不需要分解了。）

（3）分解过程常用方法：hash(x)%m。其中x为字符串/url/ip，m为小问题的数目，比如把一个大文件分解为1000份，m=1000；

（4）解决问题辅助数据结构：hash\_map，Trie树，bit map，二叉排序树（AVL，SBT，红黑树）；

（5）top K问题：最大K个用最小堆，最小K个用最大堆。（至于为什么？自己在纸上写个小栗子，试一下就知道了。）

（6）处理大数据常用排序：快速排序/堆排序/归并排序/桶排序

一、Bloom filter

适用范围：可以用来实现数据字典，进行数据的判重，或者集合求交集

**基本原理及要点**：

对于原理来说很简单，位数组+k 个独立 hash 函数。将 hash 函数对应的值的位数组置 1，查找时如果发现所有 hash 函数对应位都是 1 说明存在，很明显这个过程并不保证查找的结果是 100%正确的。同时也不支持删除一个已经插入的关键字，因为该关键字对应的位会牵动到其他的关键字。所以一个简单的改进就是 counting Bloom filter，用一个 counter 数组代替位数组，就可以支持删除了。

还有一个比较重要的问题，如何根据输入元素个数 n，确定位数组 m 的大小及 hash 函数个数。当 hash函数个数 k=(ln2)\*(m/n)时错误率最小。在错误率不大于 E 的情况下，m 至少要等于 n\*lg(1/E)才能表示任意 n 个元素的集合。但 m 还应该更大些，因为还要保证 bit 数组里至少一半为 0，则 m 应该>=nlg(1/E)\*lge大概就是 nlg(1/E)1.44 倍(lg 表示以 2 为底的对数)。

举个例子我们假设错误率为 0.01，则此时 m 应大概是 n 的 13 倍。这样 k 大概是 8 个。

注意这里 m 与 n 的单位不同，m 是 bit 为单位，而 n 则是以元素个数为单位(准确的说是不同元素的个数)。通常单个元素的长度都是有很多 bit 的。所以使用 bloom filter 内存上通常都是节省的。

扩展：

Bloom filter 将集合中的元素映射到位数组中，用 k（k 为哈希函数个数）个映射位是否全 1 表示元素在不在这个集合中。Counting bloom filter（CBF）将位数组中的每一位扩展为一个 counter，从而支持了元素的删除操作。Spectral Bloom Filter（SBF）将其与集合元素的出现次数关联。SBF 采用 counter 中的最小值来近似表示元素的出现频率。

问题实例：

给你 A,B 两个文件，各存放 50 亿条 URL，每条 URL 占用 64 字节，内存限制是 4G，让你找出 A,B文件共同的 URL。如果是三个乃至 n 个文件呢？

根据这个问题我们来计算下内存的占用，4G=2^32 大概是 40 亿\*8 大概是 340 亿，n=50 亿，如果按出错率 0.01 算需要的大概是 650 亿个 bit。现在可用的是 340 亿，相差并不多，这样可能会使出错率上升些。另外如果这些 urlip 是一一对应的，就可以转换成 ip，则大大简单了。

二、Hashing

适用范围：快速查找，删除的基本数据结构，通常需要总数据量可以放入内存

基本原理及要点：

hash 函数选择，针对字符串，整数，排列，具体相应的 hash 方法。碰撞处理，一种是 open hashing，也称为拉链法；另一种就是 closed hashing，也称开地址法，openedaddressing。

扩展：

d-left hashing 中的 d 是多个的意思，我们先简化这个问题，看一看 2-left hashing。2-left hashing 指的是将一个哈希表分成长度相等的两半，分别叫做 T1 和 T2，给 T1 和 T2 分别配备一个哈希函数，h1 和h2。在存储一个新的 key 时，同时用两个哈希函数进行计算，得出两个地址 h1[key]和 h2[key]。这时需要检查 T1 中的 h1[key]位置和 T2 中的 h2[key]位置，哪一个位置已经存储的（有碰撞的）key 比较多，然后将新 key 存储在负载少的位置。如果两边一样多，比如两个位置都为空或者都存储了一个 key，就把新 key存储在左边的 T1 子表中，2-left 也由此而来。在查找一个 key 时，必须进行两次 hash，同时查找两个位置。

问题实例：

1).海量日志数据，提取出某日访问百度次数最多的那个 IP。

IP 的数目还是有限的，最多 2^32 个，所以可以考虑使用 hash 将 ip 直接存入内存，然后进行统计。

三、bit-map

适用范围：可进行数据的快速查找，判重，删除，一般来说数据范围是 int 的 10 倍以下

基本原理及要点：使用 bit 数组来表示某些元素是否存在，比如 8 位电话号码

扩展：

bloom filter 可以看做是对 bit-map 的扩展

问题实例：

1)已知某个文件内包含一些电话号码，每个号码为 8 位数字，统计不同号码的个数。

8 位最多 99 999 999，大概需要 99m 个 bit，大概 10 几 m 字节的内存即可。

2)2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这 2.5 亿个整数。

将 bit-map 扩展一下，用 2bit 表示一个数即可，0 表示未出现，1 表示出现一次，2 表示出现 2 次及以上。或者我们不用 2bit 来进行表示，我们用两个 bit-map 即可模拟实现这个 2bit-map。

四、堆

适用范围：海量数据前 n 大，并且 n 比较小，堆可以放入内存

基本原理及要点：最大堆求前 n 小，最小堆求前 n 大。方法，比如求前 n 小，我们比较当前元素与最大堆里的最大元素，如果它小于最大元素，则应该替换那个最大元素。这样最后得到的 n 个元素就是最小的 n 个。适合大数据量，求前 n 小，n 的大小比较小的情况，这样可以扫描一遍即可得到所有的前 n 元素，效率很高。

扩展：

双堆，一个最大堆与一个最小堆结合，可以用来维护中位数。

问题实例：

1)100w 个数中找最大的前 100 个数。

用一个 100 个元素大小的最小堆即可。

五、双层桶划分---- 其实本质上就是【分而治之】的思想，重在分的 技巧上！

适用范围：第 k 大，中位数，不重复或重复的数字基本原理及要点：因为元素范围很大，不能利用直接寻址表，所以通过多次划分，逐步确定范围，然后最后在一个可以接受的范围内进行。可以通过多次缩小，双层只是一个例子。

扩展：

问题实例：

1).2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这 2.5 亿个整数。

有点像鸽巢原理，整数个数为 2^32,也就是，我们可以将这 2^32 个数，划分为 2^8 个区域(比如用单个文件代表一个区域)，然后将数据分离到不同的区域，然后不同的区域在利用 bitmap 就可以直接解决了。也就是说只要有足够的磁盘空间，就可以很方便的解决。

2).5 亿个 int 找它们的中位数。

这个例子比上面那个更明显。首先我们将 int 划分为 2^16 个区域，然后读取数据统计落到各个区域里的数的个数，之后我们根据统计结果就可以判断中位数落到那个区域，同时知道这个区域中的第几大数刚好是中位数。然后第二次扫描我们只统计落在这个区域中的那些数就可以了。实际上，如果不是 int 是 int64，我们可以经过 3 次这样的划分即可降低到可以接受的程度。即可以先将 int64 分成 2^24 个区域，然后确定区域的第几大数，在将该区域分成 2^20 个子区域，然后确定是子区域的第几大数，然后子区域里的数的个数只有 2^20，就可以直接利用 direct addr table 进行统计了。

六、数据库索引

适用范围：大数据量的增删改查

基本原理及要点：利用数据的设计实现方法，对海量数据的增删改查进行处理。

七、倒排索引(Inverted index)

适用范围：搜索引擎，关键字查询

基本原理及要点：为何叫倒排索引？一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。以英文为例，下面是要被索引的文本： T0 = "it is what it is" T1 = "what is it" T2 = "it is a banana"

我们就能得到下面的反向文件索引：

"a": {2} "banana": {2} "is": {0, 1, 2} "it": {0, 1, 2} "what": {0, 1}

检索的条件"what","is"和"it"将对应集合的交集。

正向索引开发出来用来存储每个文档的单词的列表。正向索引的查询往往满足每个文档有序频繁的全文查询和每个单词在校验文档中的验证这样的查询。在正向索引中，文档占据了中心的位置，每个文档指向了一个它所包含的索引项的序列。也就是说文档指向了它包含的那些单词，而反向索引则是单词指向了包含它的文档，很容易看到这个反向的关系。

扩展：

问题实例：文档检索系统，查询那些文件包含了某单词，比如常见的学术论文的关键字搜索。

八、外排序

适用范围：大数据的排序，去重

基本原理及要点：外排序的归并方法，置换选择败者树原理，最优归并树

扩展：

问题实例：

1).有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 个字节，内存限制大小是1M。返回频数最高的 100 个词。这个数据具有很明显的特点，词的大小为 16 个字节，但是内存只有 1m 做 hash 有些不够，所以可以用来排序。内存可以当输入缓冲区使用。

九、trie 树

适用范围：数据量大，重复多，但是数据种类小可以放入内存

基本原理及要点：实现方式，节点孩子的表示方式

扩展：压缩实现。

问题实例：

1).有 10 个文件，每个文件 1G，每个文件的每一行都存放的是用户的 query，每个文件的 query 都可能重复。要你按照 query 的频度排序。

2).1000 万字符串，其中有些是相同的(重复),需要把重复的全部去掉，保留没有重复的字符串。请问怎么设计和实现？

3).寻找热门查询：查询串的重复度比较高，虽然总数是 1 千万，但如果除去重复后，不超过 3 百万个，每个不超过 255 字节。

十、分布式处理 mapreduce

适用范围：数据量大，但是数据种类小可以放入内存

基本原理及要点：将数据交给不同的机器去处理，数据划分，结果归约。

扩展：

问题实例：

1).The canonical example application of MapReduce is a process to count the appearances ofeach

different word in a set of documents:

2).海量数据分布在 100 台电脑中，想个办法高效统计出这批数据的 TOP10。

3).一共有 N 个机器，每个机器上有 N 个数。每个机器最多存 O(N)个数并对它们操作。如何找到 N^2个数的中数(median)？

经典问题分析

上千万 or 亿数据（有重复），统计其中出现次数最多的前 N 个数据,分两种情况：可一次读入内存，不可一次读入。

可用思路：trie 树+堆，数据库索引，划分子集分别统计，hash，分布式计算，近似统计，外排序

所谓的是否能一次读入内存，实际上应该指去除重复后的数据量。如果去重后数据可以放入内存，我们可以为数据建立字典，比如通过 map，hashmap，trie，然后直接进行统计即可。当然在更新每条数据的出现次数的时候，我们可以利用一个堆来维护出现次数最多的前 N 个数据，当然这样导致维护次数增加，不如完全统计后在求前 N 大效率高。

如果数据无法放入内存。一方面我们可以考虑上面的字典方法能否被改进以适应这种情形，可以做的改变就是将字典存放到硬盘上，而不是内存，这可以参考数据库的存储方法。当然还有更好的方法，就是可以采用分布式计算，基本上就是 map-reduce 过程，首先可以根据数据值或者把数据 hash(md5)后的值，将数据按照范围划分到不同的机子，最好可以让数据划分后可以一次读入内存，这样不同的机子负责处理各种的数值范围，实际上就是 map。得到结果后，各个机子只需拿出各自的出现次数最多的前 N 个数据，然后汇总，选出所有的数据中出现次数最多的前 N 个数据，这实际上就是 reduce 过程。实际上可能想直接将数据均分到不同的机子上进行处理，这样是无法得到正确的解的。因为一个数据可能被均分到不同的机子上，而另一个则可能完全聚集到一个机子上，同时还可能存在具有相同数目的数据。比如我们要找出现次数最多的前 100 个，我们将 1000 万的数据分布到 10 台机器上，找到每台出现次数最多的前 100 个，归并之后这样不能保证找到真正的第 100 个，因为比如出现次数最多的第 100 个可能有 1 万个，但是它被分到了 10 台机子，这样在每台上只有 1 千个，假设这些机子排名在 1000 个之前的那些都是单独分布在一台机子上的，比如有 1001 个，这样本来具有 1 万个的这个就会被淘汰，即使我们让每台机子选出出现次数最多的 1000 个再归并，仍然会出错，因为可能存在大量个数为 1001 个的发生聚集。因此不能将数据随便均分到不同机子上，而是要根据 hash 后的值将它们映射到不同的机子上处理，让不同的机器处理一个数值范围。

而外排序的方法会消耗大量的 IO，效率不会很高。而上面的分布式方法，也可以用于单机版本，也就是将总的数据根据值的范围，划分成多个不同的子文件，然后逐个处理。处理完毕之后再对这些单词的及其出现频率进行一个归并。实际上就可以利用一个外排序的归并过程。

另外还可以考虑近似计算，也就是我们可以通过结合自然语言属性，只将那些真正实际中出现最多的那些词作为一个字典，使得这个规模可以放入内存。

## Hadoop HA

### 知识点

详情参考课件:大数据离线阶段--Day09.pdf

#### 1) Namenode HA

HA(High Available), 高可用，是保证业务连续性的有效解决方案,一般有两个或两个以上的节点，分为 活动节点 （ Active ）及 备用节点 （ Standby） ）。用于实现业务的不中断或短暂中断

NN 是 HDFS 集群的单点故障点.在 HA 具体实现方法不同情况下，HA 框架的流程是一致的, 不一致的就是如何存储、管理、同步 edits 编辑日志文件。

QJM/Qurom Journal Manager,基本原理就是用 2N+1 台 JournalNode 存储 EditLog，每次写数据操作有>=N+1 返回成功时即认为该次写成功，数据不会丢失了

在 HA 模式下，datanode 需要确保同一时间有且只有一个 NN 能命令 DN。

FailoverController 主要包括三个组件: :

HealthMonitor: 监控 NameNode 是否处于 unavailable 或 unhealthy 状态。当前通过

RPC 调用 NN 相应的方法完成。

ActiveStandbyElector: 监控 NN 在 ZK 中的状态。

ZKFailoverController: 订阅 HealthMonitor 和 ActiveStandbyElector 的事件，并管理 NN 的状态,另外 zkfc 还负责解决 fencing（也就是脑裂问题）。

ZKFailoverController 主要职责：

 健康监测：周期性的向它监控的 NN 发送健康探测命令，从而来确定某个 NameNode

是否处于健康状态，如果机器宕机，心跳失败，那么 zkfc 就会标记它处于一个不

健康的状态

 会话管理：如果 NN 是健康的，zkfc 就会在 zookeeper 中保持一个打开的会话，如

果 NameNode 同时还是 Active 状态的，那么 zkfc 还会在 Zookeeper 中占有一个类

型为短暂类型的 znode，当这个 NN 挂掉时，这个 znode 将会被删除，然后备用的

NN 将会得到这把锁，升级为主 NN，同时标记状态为 Active

 当宕机的 NN 新启动时，它会再次注册 zookeper，发现已经有 znode 锁了，便会自

动变为 Standby 状态，如此往复循环，保证高可靠，需要注意，目前仅仅支持最

多配置 2 个 NN

 master 选举：通过在 zookeeper 中维持一个短暂类型的 znode，来实现抢占式的

锁机制，从而判断那个 NameNode 为 Active 状态

#### 2) yarn HA

Hadoop 2.4.0版本开始，Yarn 实现了 ResourceManager HA

由于资源使用情况和 NodeManager 信息都可以通过 NodeManager 的心跳机制重新构建出来，因此只需要对 ApplicationMaster 相关的信息进行持久化存储即可。

在一个典型的 HA 集群中，两台独立的机器被配置成 ResourceManger。在任意时间，有且只允许一个活动的 ResourceManger,另外一个备用。切换分为两种方式：

手动切换：在自动恢复不可用时，管理员可用手动切换状态，或是从 Active 到 Standby,或是从 Standby 到 Active。

自动切换：基于 Zookeeper，但是区别于 HDFS 的 HA，2 个节点间无需配置额外的 ZFKC守护进程来同步数据。

#### 3) HA Hadoop搭建

参考资料:Hadoop>Hadoop ha下文档hadoop2.X-HA集群搭建.txt

1.修改Linux主机名

2.修改IP

3.修改主机名和IP的映射关系 /etc/hosts

4.关闭防火墙

5.ssh免登陆

6.安装JDK，配置环境变量等

7.注意集群时间要同步

集群部署节点角色的规划（7节点）

------------------

server01 namenode zkfc

server02 namenode zkfc

server03 resourcemanager

server04 resourcemanager

server05 datanode nodemanager zookeeper journal node

server06 datanode nodemanager zookeeper journal node

server07 datanode nodemanager zookeeper journal node

------------------

集群部署节点角色的规划（3节点）

------------------

server01 namenode resourcemanager zkfc nodemanager datanode zookeeper journal node

server02 namenode resourcemanager zkfc nodemanager datanode zookeeper journal node

server03 datanode nodemanager zookeeper journal node

------------------

---------------------------------------------------------------------------------------------------------------

安装步骤：

1.安装配置zooekeeper集群

1.1解压

tar -zxvf zookeeper-3.4.5.tar.gz -C /home/hadoop/app/

1.2修改配置

cd /home/hadoop/app/zookeeper-3.4.5/conf/

cp zoo\_sample.cfg zoo.cfg

vim zoo.cfg

修改：dataDir=/home/hadoop/app/zookeeper-3.4.5/tmp

在最后添加：

server.1=hadoop05:2888:3888

server.2=hadoop06:2888:3888

server.3=hadoop07:2888:3888

保存退出

然后创建一个tmp文件夹

mkdir /home/hadoop/app/zookeeper-3.4.5/tmp

echo 1 > /home/hadoop/app/zookeeper-3.4.5/tmp/myid

1.3将配置好的zookeeper拷贝到其他节点(首先分别在hadoop06、hadoop07根目录下创建一个hadoop目录：mkdir /hadoop)

scp -r /home/hadoop/app/zookeeper-3.4.5/ hadoop06:/home/hadoop/app/

scp -r /home/hadoop/app/zookeeper-3.4.5/ hadoop07:/home/hadoop/app/

注意：修改hadoop06、hadoop07对应/hadoop/zookeeper-3.4.5/tmp/myid内容

hadoop06：

echo 2 > /home/hadoop/app/zookeeper-3.4.5/tmp/myid

hadoop07：

echo 3 > /home/hadoop/app/zookeeper-3.4.5/tmp/myid

2.安装配置hadoop集群

2.1解压

tar -zxvf hadoop-2.6.4.tar.gz -C /home/hadoop/app/

2.2配置HDFS（hadoop2.0所有的配置文件都在$HADOOP\_HOME/etc/hadoop目录下）

#将hadoop添加到环境变量中

vim /etc/profile

export JAVA\_HOME=/usr/java/jdk1.7.0\_55

export HADOOP\_HOME=/hadoop/hadoop-2.6.4

export PATH=$PATH:$JAVA\_HOME/cluster1n:$HADOOP\_HOME/cluster1n

#hadoop2.0的配置文件全部在$HADOOP\_HOME/etc/hadoop下

cd /home/hadoop/app/hadoop-2.6.4/etc/hadoop

2.2.1修改hadoop-env.sh

export JAVA\_HOME=/home/hadoop/app/jdk1.7.0\_55

###############################################################################

2.2.2修改core-site.xml

<configuration>

<!-- 集群名称在这里指定！该值来自于hdfs-site.xml中的配置 -->

<property>

<name>fs.defaultFS</name>

<value>hdfs://cluster1</value>

</property>

<!-- 这里的路径默认是NameNode、DataNode、JournalNode等存放数据的公共目录 -->

<property>

<name>hadoop.tmp.dir</name>

<value>/root/apps/hadoop/tmp</value>

</property>

<!-- ZooKeeper集群的地址和端口。注意，数量一定是奇数，且不少于三个节点-->

<property>

<name>ha.zookeeper.quorum</name>

<value>hadoop05:2181,hadoop06:2181,hadoop07:2181</value>

</property>

</configuration>

###############################################################################

2.2.3修改hdfs-site.xml

<configuration>

<!--指定hdfs的nameservice为cluster1，需要和core-site.xml中的保持一致 -->

<property>

<name>dfs.nameservices</name>

<value>cluster1</value>

</property>

<!-- cluster1下面有两个NameNode，分别是nn1，nn2 -->

<property>

<name>dfs.ha.namenodes.cluster1</name>

<value>nn1,nn2</value>

</property>

<!-- nn1的RPC通信地址 -->

<property>

<name>dfs.namenode.rpc-address.cluster1.nn1</name>

<value>mini1:9000</value>

</property>

<!-- nn1的http通信地址 -->

<property>

<name>dfs.namenode.http-address.cluster1.nn1</name>

<value>hadoop00:50070</value>

</property>

<!-- nn2的RPC通信地址 -->

<property>

<name>dfs.namenode.rpc-address.cluster1.nn2</name>

<value>hadoop01:9000</value>

</property>

<!-- nn2的http通信地址 -->

<property>

<name>dfs.namenode.http-address.cluster1.nn2</name>

<value>hadoop01:50070</value>

</property>

<!-- 指定NameNode的edits元数据在JournalNode上的存放位置 -->

<property>

<name>dfs.namenode.shared.edits.dir</name>

<value>qjournal://hadoop05:8485;hadoop06:8485;hadoop07:8485/cluster1</value>

</property>

<!-- 指定JournalNode在本地磁盘存放数据的位置 -->

<property>

<name>dfs.journalnode.edits.dir</name>

<value>/home/hadoop/app/hdpdata/journaldata</value>

</property>

<!-- 开启NameNode失败自动切换 -->

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

<!-- 指定该集群出故障时，哪个实现类负责执行故障切换 -->

<property>

<name>dfs.client.failover.proxy.provider.cluster1</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

</property>

<!-- 配置隔离机制方法，多个机制用换行分割，即每个机制暂用一行-->

<property>

<name>dfs.ha.fencing.methods</name>

<value>

sshfence

</value>

</property>

<!-- 使用sshfence隔离机制时需要ssh免登陆 -->

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/home/hadoop/.ssh/id\_rsa</value>

</property>

<!-- 配置sshfence隔离机制超时时间 -->

<property>

<name>dfs.ha.fencing.ssh.connect-timeout</name>

<value>30000</value>

</property>

</configuration>

###############################################################################

2.2.4修改mapred-site.xml

<configuration>

<!-- 指定mr框架为yarn方式 -->

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

###############################################################################

2.2.5修改yarn-site.xml

<configuration>

<!-- 开启RM高可用 -->

<property>

<name>yarn.resourcemanager.ha.enabled</name>

<value>true</value>

</property>

<!-- 指定RM的cluster id -->

<property>

<name>yarn.resourcemanager.cluster-id</name>

<value>yrc</value>

</property>

<!-- 指定RM的名字 -->

<property>

<name>yarn.resourcemanager.ha.rm-ids</name>

<value>rm1,rm2</value>

</property>

<!-- 分别指定RM的地址 -->

<property>

<name>yarn.resourcemanager.hostname.rm1</name>

<value>node-1</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm2</name>

<value>node-2</value>

</property>

<!-- 指定zk集群地址 -->

<property>

<name>yarn.resourcemanager.zk-address</name>

<value>node-1:2181,node-2:2181,node-3:2181</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

</configuration>

2.2.6修改slaves(slaves是指定子节点的位置，因为要在hadoop01上启动HDFS、在hadoop03启动yarn，所以hadoop01上的slaves文件指定的是datanode的位置，hadoop03上的slaves文件指定的是nodemanager的位置)

hadoop05

hadoop06

hadoop07

2.2.7配置免密码登陆

#首先要配置hadoop00到hadoop01、hadoop02、hadoop03、hadoop04、hadoop05、hadoop06、hadoop07的免密码登陆

#在hadoop01上生产一对钥匙

ssh-keygen -t rsa

#将公钥拷贝到其他节点，\*\*\*\*包括自己\*\*\*\*

ssh-coyp-id hadoop00

ssh-coyp-id hadoop01

ssh-coyp-id hadoop02

ssh-coyp-id hadoop03

ssh-coyp-id hadoop04

ssh-coyp-id hadoop05

ssh-coyp-id hadoop06

ssh-coyp-id hadoop07

#注意：两个namenode之间要配置ssh免密码登陆 ssh远程补刀时候需要

###注意：严格按照下面的步骤!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

2.5启动zookeeper集群（分别在hadoop05、hadoop06、tcast07上启动zk）

bin/zkServer.sh start

#查看状态：一个leader，两个follower

bin/zkServer.sh status

2.6手动启动journalnode（分别在在hadoop05、hadoop06、hadoop07上执行）

hadoop-daemon.sh start journalnode

#运行jps命令检验，hadoop05、hadoop06、hadoop07上多了JournalNode进程

2.7格式化namenode

#在hadoop00上执行命令:

hdfs namenode -format

#格式化后会在根据core-site.xml中的hadoop.tmp.dir配置的目录下生成个hdfs初始化文件，

把hadoop.tmp.dir配置的目录下所有文件拷贝到另一台namenode节点所在的机器

scp -r tmp/ hadoop02:/home/hadoop/app/hadoop-2.6.4/

##也可以这样，建议hdfs namenode -bootstrapStandby

2.8格式化ZKFC(在active上执行即可)

hdfs zkfc -formatZK

2.9启动HDFS(在hadoop00上执行)

start-dfs.sh

2.10启动YARN

start-yarn.sh

还需要手动在standby上手动启动备份的 resourcemanager

yarn-daemon.sh start resourcemanager

到此，hadoop-2.6.4配置完毕，可以统计浏览器访问:

http://hadoop00:50070

NameNode 'hadoop01:9000' (active)

http://hadoop01:50070

NameNode 'hadoop02:9000' (standby)

验证HDFS HA

首先向hdfs上传一个文件

hadoop fs -put /etc/profile /profile

hadoop fs -ls /

然后再kill掉active的NameNode

kill -9 <pid of NN>

通过浏览器访问：http://192.168.1.202:50070

NameNode 'hadoop02:9000' (active)

这个时候hadoop02上的NameNode变成了active

在执行命令：

hadoop fs -ls /

-rw-r--r-- 3 root supergroup 1926 2014-02-06 15:36 /profile

刚才上传的文件依然存在！！！

手动启动那个挂掉的NameNode

hadoop-daemon.sh start namenode

通过浏览器访问：http://192.168.1.201:50070

NameNode 'hadoop01:9000' (standby)

验证YARN：

运行一下hadoop提供的demo中的WordCount程序：

hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.4.1.jar wordcount /profile /out

OK，大功告成！！！

测试集群工作状态的一些指令 ：

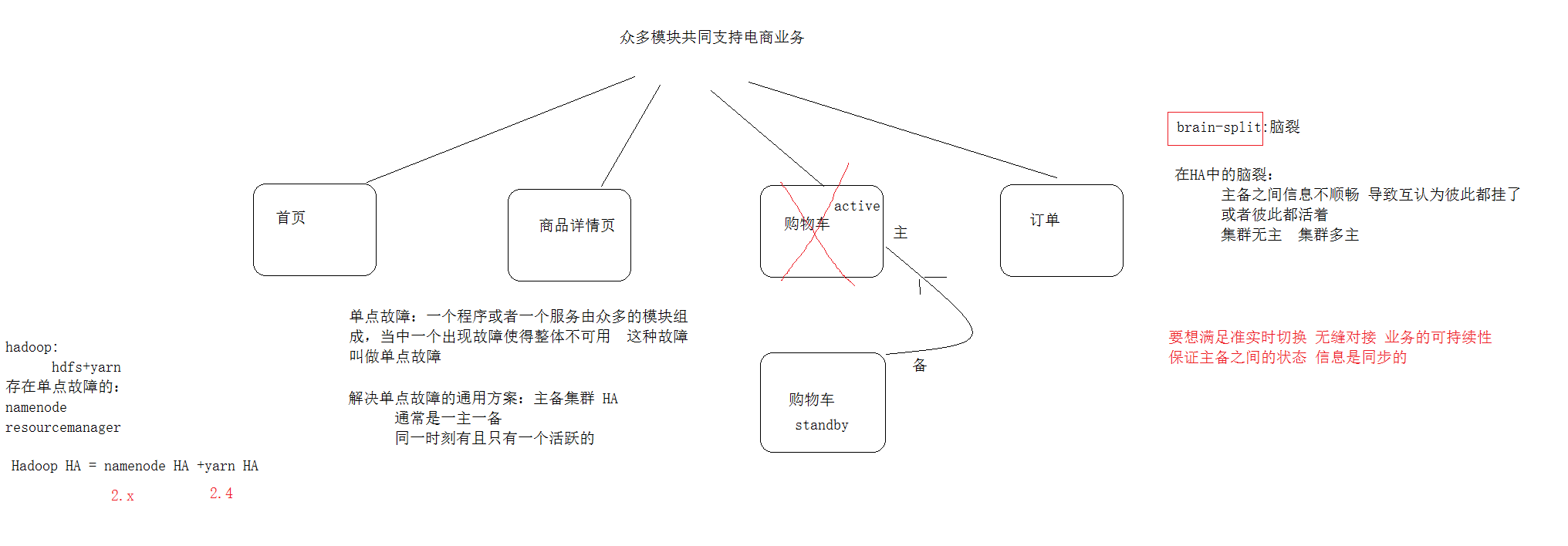
hdfs dfsadmin -report 查看hdfs的各节点状态信息

cluster1n/hdfs haadmin -getServiceState nn1 获取一个namenode节点的HA状态

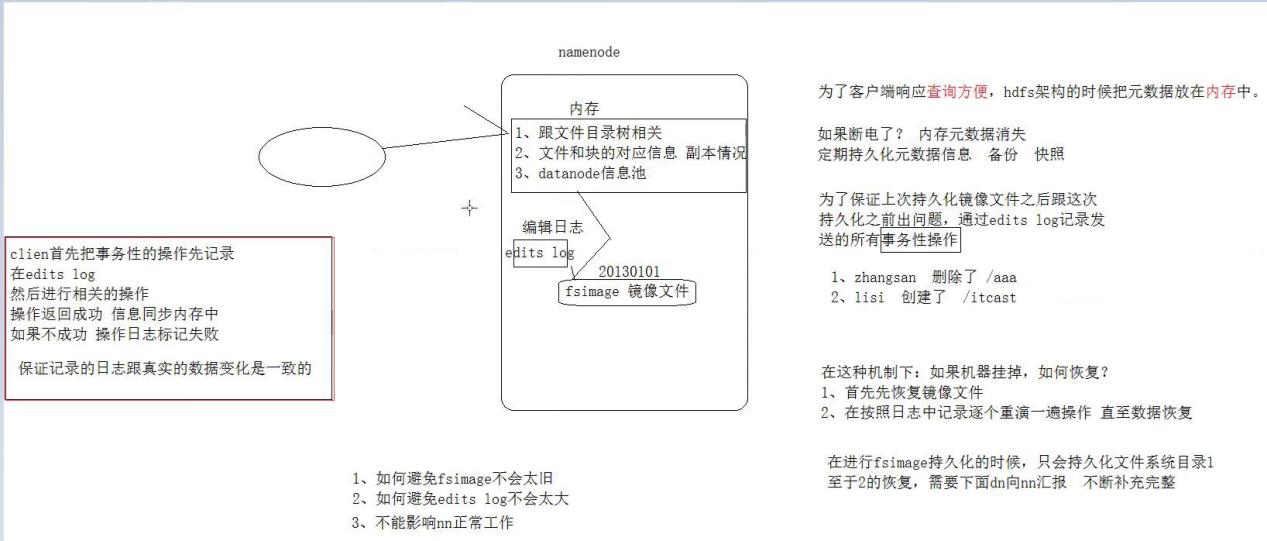
scluster1n/hadoop-daemon.sh start namenode 单独启动一个namenode进程

./hadoop-daemon.sh start zkfc 单独启动一个zkfc进程

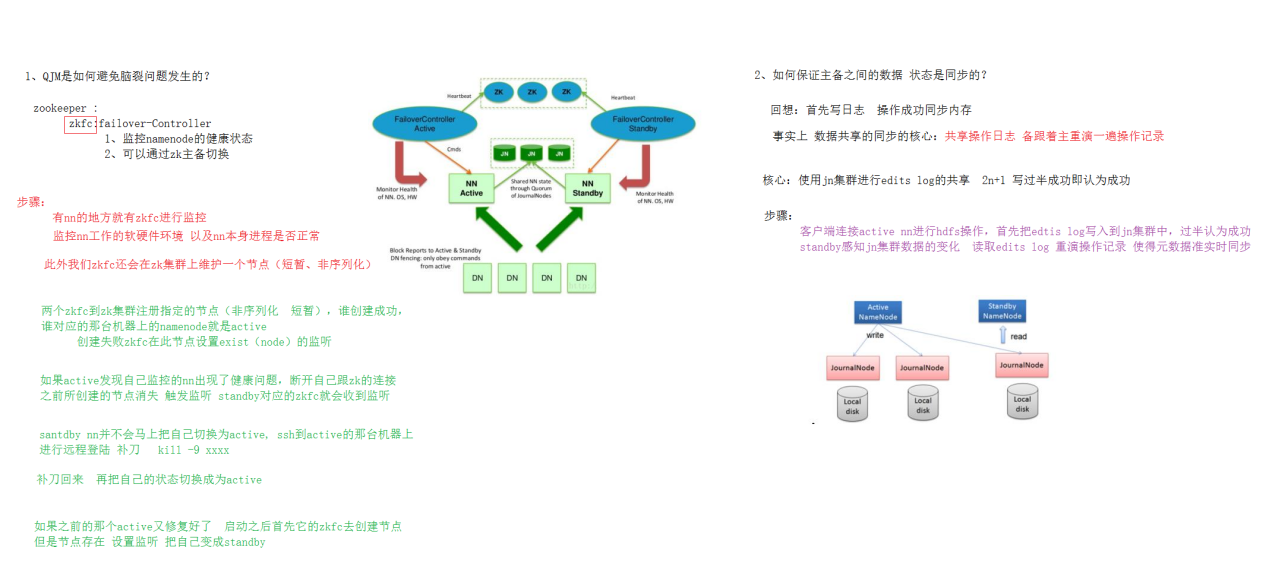
#### 1) 单点故障与”脑裂”



2) client的事务性操作对HA提供了支持



#### 3) QJM



### 3、 面试题

#### 1) hadoop 的 namenode 宕机,怎么解决？

先分析宕机后的损失，宕机后直接导致client无法访问，内存中的元数据丢失，但是硬盘中的元数据应该还存在，如果只是节点挂了，重启即可，如果是机器挂了，重启机器后看节点是否能重启，不能重启就要找到原因修复了。但是最终的解决方案应该是在设计集群的初期就考虑到这个问题，做namenode的HA。

#### 2) Hadoop 节点分布



## Hadoop的联邦机制

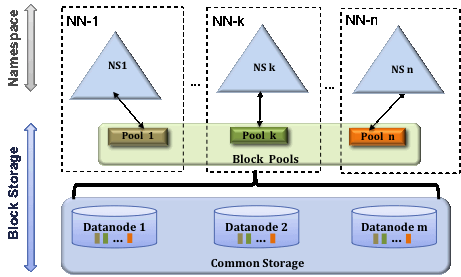
### 为什么会出现联邦？

Hadoop的NN所使用的资源受所在服务的物理限制，不能满足实际生产需求。

### 联邦的实现

采用多台NN组成联邦。NN是独立的，NN之间不需要相互调用。NN是联合的，同属于一个联邦，所管理的DN作为block的公共存储。

如下图：



图中概念：

* block pool的概念，每一个namespace都有一个pool，datanodes会存储集群中所有的pool，block pool之间的管理是独立的，一个namespace生成一个block id时不需要跟其它namespace协调，一个namenode的失败也不会影响到datanode对其它namenodes的服务。
* 一个namespace和它的block pool作为一个管理单元，删除后，对应于datanodes中的pool也会被删除。集群升级时，这个管理单元也独立升级。
* 这里引入clusterID来标示集群所有节点。当一个namenode format之后，这个id生成，集群中其它namenode的format也用这个id。

### 主要优点：

* 命名空间可伸缩性——联合添加命名空间水平扩展。DN也随着NN的加入而得到拓展。
* 性能——文件系统吞吐量不是受单个Namenode限制。添加更多的Namenode集群扩展文件系统读/写吞吐量。
* 隔离——隔离不同类型的程序，一定程度上控制资源的分配

### 配置：

联邦的配置是向后兼容的，允许在不改变任何配置的情况下让当前运行的单节点环境转换成联邦环境。新的配置方案确保了在集群环境中的所有节点的配置文件都是相同的。

这里引入了NameServiceID概念，作为namenodes们的后缀。

第一步：配置属性dfs.nameservices，用于datanodes们识别namenodes。

第二步：为每个namenode加入这个后缀。

conf/hdfs-site.xml

<configuration>  
 <property>  
 <name>dfs.nameservices</name>  
 <value>ns1,ns2</value>  
 </property>  
 <property>  
 <name>dfs.namenode.rpc-address.ns1</name>  
 <value>nn-host1:rpc-port</value>  
 </property>  
 <property>  
 <name>dfs.namenode.http-address.ns1</name>  
 <value>nn-host1:http-port</value>  
 </property>  
 <property>  
 <name>dfs.namenode.secondary.http-address.ns1</name>  
 <value>snn-host1:http-port</value>  
 </property>  
 <property>  
 <name>dfs.namenode.rpc-address.ns2</name>  
 <value>nn-host2:rpc-port</value>  
 </property>  
 <property>  
 <name>dfs.namenode.http-address.ns2</name>  
 <value>nn-host2:http-port</value>  
 </property>  
 <property>  
 <name>dfs.namenode.secondary.http-address.ns2</name>  
 <value>snn-host2:http-port</value>  
 </property>  
  
 .... Other common configuration ...  
</configuration>

### 操作

# 创建联邦，不指定ID会自动生成  
$HADOOP\_HOME/bin/hdfs namenode -format [-clusterId <cluster\_id>]  
# 升级Hadoop为集群  
$HADOOP\_HOME/bin/hdfs start namenode --config $HADOOP\_CONF\_DIR -upgrade -clusterId <cluster\_ID>  
# 扩展已有联邦  
$HADOOP\_HOME/bin/hdfs dfsadmin -refreshNamenodes <datanode\_host\_name>:<datanode\_rpc\_port>  
# 退出联邦  
$HADOOP\_HOME/sbin/distribute-exclude.sh <exclude\_file>  
$HADOOP\_HOME/sbin/refresh-namenodes.sh

什么是 CDH 下载地址 : <http://archive.cloudera.com/cdh5/cdh/5/>

CDH (Cloudera's Distribution, including Apache Hadoop)，是 Hadoop

众多分支中的一种，由 Cloudera 维护，基于稳定版本的 Apache Hadoop 构建，

并集成了很多补丁， 可直接用于生产环境。

CDH 的优点：

版本划分清晰

版本更新速度快

支持 Kerberos 安全认证

文档清晰

支持多种安装方式（Cloudera Manager、YUM、RPM、Tarball）

什么是 CM

Cloudera Manager 是为了便于在集群中进行 Hadoop 等大数据处理相关

的服务安装和监控管理的组件，对集群中主机、Hadoop、Hive、Spark 等服务

的安装配置管理做了极大简化。

Cloudera Manager 有四大功能：

（1）管理：对集群进行管理，如添加、删除节点等操作。

（2）监控：监控集群的健康情况，对设置的各种指标和系统运行情况进行全面监控。

（3）诊断：对集群出现的问题进行诊断，对出现的问题给出建议解决方案。

（4）集成：对 hadoop 的多组件进行整合

## 项目

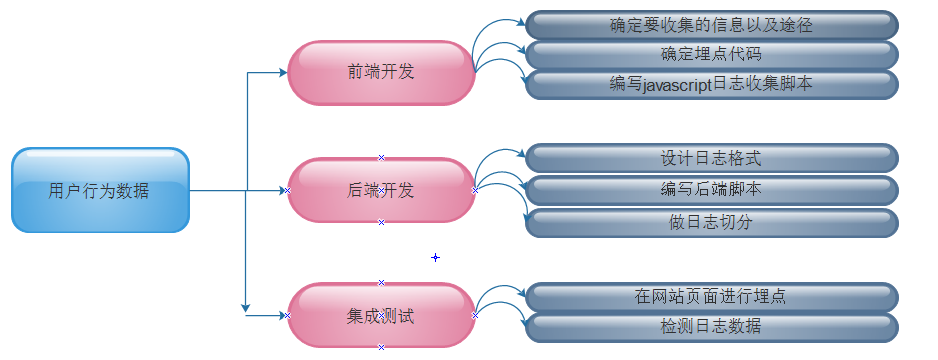
### 点击流日志模型

#### 1分析什么

点击流日志模型主要是分析用户在网站上持续访问的轨迹，查看网站的访问量，分析关键路径得到漏斗模型，以此来优化网站，提高用户的网站停留时间。

#### 2确定数据源

在本次案例中，使用埋点的方式收集数据，埋点是指在网页中插入小段javascript代码，这个代码片段会动态创建一个script标签，并将src属性指向一个单独的js文件，此时这个单独的js文件会被浏览器请求到并执行，这个js往往就是真正的数据收集脚本。

数据收集完成后，js会请求一个后端的数据收集脚本，这个脚本一般是一个伪装成图片的动态脚本程序，js会将收集的数据通过http参数的方式传递给后端脚本，后端脚本解析参数并按固定格式记录到访问日志，同时可能会在http响应中给客户端种植一些用于追踪的cookie。设计

1）确定收集信

1）确定埋点代码

<script type=”text/javascript”>

val \_map=\_map || [];

\_map.push([‘\_setAccount’],’ UA-XXXXX-X’);

//自调用匿名函数只会在运行时执行一次，一般用于初始化

(function() {

//引入一个外部的js文件ma

var ma = document.createElement('script');

ma.type ='text/javascript';

//异步调用外部js文件

ma.async = true;

//根据协议将src指向对应的ma.js

ma.src = ('https:' == document.location.protocol ?

'https://ssl' : 'http://www') + '.google-analytics.com/ma.js';

//将这个元素插入到dom树上

var s = document.getElementsByTagName('script')[0];

s.parentNode.insertBefore( ma, s);

})();

</script>

2）前端收集脚本

在第二步配置中的ma.js被请求后会被执行，一般要做如下几件事：

1. 通过浏览器内置js对象收集信息
2. 解析\_map数组，收集配置信息，这里面可能会包括用户自定义的事件跟踪，业务数据等
3. 将上面两步收集的数据按预定义格式解析并拼接
4. 请求一个后端脚本，将信息放在http request参数中携带给后端脚本

示例代码

(function(){

var params={};

//document对象数据，实现了A步骤

if(document){

params.domain=document.domain||’’;

params.url=document.URL||’’;

params.title=document.title||’’;

params.referrer=document.referrer||’’;

}

//window对象数据

if(window&&window.screen){

params.sh=window.screen.height||0;

params.sw=window.screen.width||0;

params.cd=window.screen.colorDepth||0;

}

//nevigator对象数据

if(nevigator){

params.lang=navigator.language||’’;

}

//接卸map数组。收集配置信息

if(\_maq){

for(var i in\_maq){

switch(\_map[i][0]){

case ‘\_setAccount’:

params.account=\_maq[i][1];

break;

default:

break;

}

}

}

//拼接参数串

var args=’’;

for(var i in params){

if(agrs!=’’)args+=’&’;

args += i + '=' + encodeURIComponent(params[i]);

}

//通过image对象请求后端脚本

var img=new Image(1,1);

//请求后端脚本常用的是ajax，但是ajax是不能跨域请求的，通用的方

//法是js脚本创建一个image对象，将image对象的src属性指向后端脚//本并携带参数,就实现了跨域请求。

img.src=’http://xx.xxx.xxx/log.gif?’+args;

})();

后端脚本

log.gif是后端脚本，是一个伪装成gif图片的脚本，后端脚本一般需要完成以下几件事情

1. 解析http请求参数得到信息
2. 从web服务器获取一些客户端无法获取的信息
3. 将信息按格式写入log
4. 生成一副1\*1的gif图片作为响应内容并将响应头的content-type设置为image/gif
5. 在响应头中通过set-cookie设置一些需要的cookie信息

我们使用nginx的access\_log做日志收集

首先，需要在 nginx 的配置文件中定义日志格式：

log\_format tick

"$msec||$remote\_addr||$status||$body\_bytes\_sent||$u\_domain||$u\_url|

|$u\_title||$u\_referrer||$u\_sh||$u\_sw||$u\_cd||$u\_lang||$http\_user\_ag

ent||$u\_account";

注意这里以 u\_开头的是我们待会会自己定义的变量，其它的是 nginx 内置变

量。然后是核心的两个 location：

location / log.gif {

#伪装成 gif 文件

default\_type image/gif;

#本身关闭 access\_log，通过 subrequest 记录 log

access\_log off;

access\_by\_lua "

-- 用户跟踪 cookie 名为\_\_utrace

local uid = ngx.var.cookie\_\_\_utrace

if not uid then

-- 如果没有则生成一个跟踪 cookie，算法为

md5(时间戳+IP+客户端信息)

uid = ngx.md5(ngx.now() ..

ngx.var.remote\_addr .. ngx.var.http\_user\_agent)

end

ngx.header['Set-Cookie'] = {'\_\_utrace=' .. uid ..

'; path=/'}

if ngx.var.arg\_domain then

-- 通过 subrequest 子请求 到/i-log 记录日志，

将参数和用户跟踪 cookie 带过去

ngx.location.capture('/i-log?' ..

ngx.var.args .. '&utrace=' .. uid)

end

";

#此请求资源本地不缓存

add\_header Expires "Fri, 01 Jan 1980 00:00:00 GMT";

add\_header Pragma "no-cache";

add\_header Cache-Control "no-cache, max-age=0, must-

revalidate";

#返回一个 1×1 的空 gif 图片

empty\_gif;

}

location /i-log {

#内部 location，不允许外部直接访问

internal;

#设置变量，注意需要 unescape，来自 ngx\_set\_misc 模块

set\_unescape\_uri $u\_domain $arg\_domain;

set\_unescape\_uri $u\_url $arg\_url;

set\_unescape\_uri $u\_title $arg\_title;

set\_unescape\_uri $u\_referrer $arg\_referrer;

set\_unescape\_uri $u\_sh $arg\_sh;

set\_unescape\_uri $u\_sw $arg\_sw;

set\_unescape\_uri $u\_cd $arg\_cd;

set\_unescape\_uri $u\_lang $arg\_lang;

set\_unescape\_uri $u\_account $arg\_account;

#打开日志

log\_subrequest on;

#记录日志到 ma.log 格式为 tick

access\_log /path/to/logs/directory/ma.log tick;

#输出空字符串

echo '';

}

日志切分

日志收集系统访问日志时间一长文件变得很大，而且日志放在一个文件不便

于管理。通常要按时间段将日志切分，例如每天或每小时切分一个日志。通过

crontab 定时调用一个 shell 脚本实现，如下：

\_prefix="/path/to/nginx"

time=`date +%Y%m%d%H`

mv ${\_prefix}/logs/ma.log ${\_prefix}/logs/ma/ma-${time}.log

kill -USR1 `cat ${\_prefix}/logs/nginx.pid `

这个脚本将 ma.log 移动到指定文件夹并重命名为 ma-{yyyymmddhh}.log，

然后向 nginx 发送 USR1 信号令其重新打开日志文件。

USR1 通常被用来告知应用程序重载配置文件, 向服务器发送一个 USR1 信号

将导致以下步骤的发生：停止接受新的连接，等待当前连接停止，重新载入配置

文件，重新打开日志文件，重启服务器，从而实现相对平滑的不关机的更改。

cat ${\_prefix}/logs/nginx.pid 取 nginx 的进程号

然后再/etc/crontab 里加入一行：

59 \* \* \* \* root /path/to/directory/rotatelog.sh

在每个小时的 59 分启动这个脚本进行日志轮转操作。

#### 3确定采集方案

在本次案例中，使用Flume日志采集系统来采集数据到HDFS系统。

#### 在服务器上部署agent节点，修改配置文件

配置文件如下

a1.sources=r1

a1.sinks=k1

a1.channels=c1

//监视指定的一些文件，将近实时的tail这些文件，获取这些文件的新追加的行

a1.sources.r1.type= TAILDIR

a1.sources.r1.channels=c1

//配置检查点文件的路径，检查点文件会以json格式保存已经tail文件的位置，解决了断点不能续传的缺陷

a1.sources.r1.positionFile=/export/log/flume/taildir\_position.json

//可以指定要tail的组

a1.sources.r1.filegroups=f1 f2

//指定每个文件的全路径

a1.sources.r1.filegroups.f1=/export/log/test1/example.log

a1.sources.r1.filegroups.f2=/export/log/test2/.\*log.\*

//配置channels

a1.channels.c1.type=memory

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

//配置下沉点，将文件写到hdfs中去

a1.sinks.k1.type=hdfs

a1.sinks.k1.channel=c1

a1.sinks.k1.hdfs.path=/flume/events/%y-%m-%d/

a1.sinks.k1.hdfs.filePrefix=events-

a1.sinks.k1.hdfs.round=true

a1.sinks.k1.hdfs.roundValue=10

a1.sinks.k1.hdfs.roundUnit=minute

#### 4确定三层架构

ETL工作的实质就是从各个数据源提取数据，对数据进行转换，并最终加载填充数据到数据仓库纬度建模后的表中。

##### 1）创建ODS层数据表

1. 创建原始数据表

drop table if exists ods\_weblog\_origin;

create table ods\_weblog\_origin(

valid string,

remote\_addr string,

remote\_user string,

time\_local string,

request string,

status string,

body\_bytes\_sent string,

http\_referer string,

http\_user\_agent string)

partitioned by (datestr string)

row format delimited

fields terminated by '\001';

1. 点击流模型pageviews表

drop table if exits ods\_weblog\_origin;

create table ods\_weblog\_origin(

valid string,

remote\_addr string,

remote\_user string,

time\_local string,

request string,

visit\_step string,

page\_staylong string,

http\_referer string,

http\_user\_agent string,

body\_bytes\_sent string,

status string)

partitioned by(datestr string)

row format delimited fields terminated by ‘\001’;

1. 点击流visit模型

drop table if exits ods\_click\_stream\_visit;

create table ods\_click\_stream\_click(

session string,

remote\_addr string,

inTime string,

outTime string,

inPage string,

outPage string,

referal string,

pageVisits int)

partitioned by (datastr string)

row format delimited fields terminated by ‘\001’;

##### 2）导入ODS层数据

load data inpath ‘/weblog/preprocessed/’ overwrite into table ods\_weblog\_origin partition(datastr=’20180526’);--数据导入

show partitions ods\_weblog\_origin;--查看分区

##### 3）生成ODS层明细宽表

#### 5模块开发---统计分析

##### 流量分析

1. 多维度统计PV总量

按时间维度

1. 计算每小时的pv数

select count(\*) as pvs ,month,day,hour from ods\_weblog\_detail group by month,day,hour;

1. 计算该处理批次一天中的各个小时pvs

drop table dw\_pvs\_everyhour\_oneday;

create table dw\_pvs\_everyhour\_oneday(month string,day string,hour string ,pvs bigint) partitioned by(datestr);

insert into table dw\_pvs\_everyhour\_oneday partition(datestr=”20180526”) select a.month as month,a.day as a.day,a.hour as hour,a.count(\*) as pvs from ods\_weblog\_detail where a.datastr=’20180526’ group by a.month,a.day,a.hour;

1. 计算每天的pvs

drop table dw\_pvs\_everyday;

create table dw\_pvs\_everyday(pvs bigint,month string,day string);

insert into table dw\_pvs\_everyday select count(\*) as pvs,a.month as month,a.day as day from ods\_weblog\_detail a group by a.month,a.day;

1. 人均浏览量

需求：统计今日所有来访者平均请求的页面数

计算方式：总页面请求数/去重总人数

remote\_addr表示不同的用户，可以先统计出不同的remote\_addr的pv量，然后累加所有pv作为总的页面请求数，再count所有remote\_addr作为总的去重总人数。

--总页面请求数/去重总人数

drop table dw\_avgpv\_user\_everyday;

create table dw\_avgpv\_user\_everyday(

day string,

avgpv string

);

Insert into table dw\_avgpv\_user\_everyday

select ‘20180526’,sum(b.pvs)/count(b.remote\_addr)from (select remote\_addr, count(1) as pvs from ods\_weblog\_detail where datestr=’20180526’ group by remote\_addr)b;

1. 统计pv总量最大的来源

统计每个小时各来访host的产生pv数最多的前N个（topN）

drop table dw\_pvs\_refhost\_topn\_everyhour;

create table dw\_pvs\_refhost\_topn\_everyhour(

hour string,toporder string,ref\_host string,ref\_host\_cnts string)

partitioned by (datestr string);

insert into dw\_pvs\_refhost\_topn\_everyhour

select ref\_host,ref\_host\_cnts,concat(month,day,hour),row\_number() over(partition by concat(month,day.hour) order by ref\_host\_cnts desc)as od from dw\_pvs\_refererhost\_everyhour where od<=3;

##### 关键路径转化率分析（漏斗模型）

A、需求分析

在一条指定的业务流程中，各个步骤的完成人数及相对上一个步骤的百分比

B、模型设计

定义好业务流程中的页面标识，下例中的步骤为：

C、模型设计

查询每一个步骤的总访问人数

Create table dw\_oute\_numbs as

select ‘step1’ as step ,count(distinct remote\_addr) as numbs from ods\_clik\_pageviews where datestr=’20180526’ and request like ‘/item%’

union

select ‘step2’ as step,count(distinct remote\_addr) as numbs from ods\_clik\_pageviews where datestr=’20180526’ and request like ‘/category%’

union

select ‘step3’ as step,count(distinct remote\_addr) as numbs from ods\_clik\_pageviews where datestr=’20180526’ and request like ‘/order%’

union

select ‘step4’ as step,count(distinct remote\_addr) as numbs from ods\_clik\_pageviews where datestr=’20180526’ and request like ‘/index%’

查询每一步相对于路径起点人数的比例

select tmp.rnstep,tmp.rnnumbs/tmp.rrnumbs as ratio from (select rn.step as rnstep,rn.rumbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs from dw\_oute\_numbs rn inner join dw\_out\_numbs rr)tmp where tmp.rrstep=’step1’;

查询每一步骤相对于上一步骤的漏出率

Select tmp.rnstep,tmp.rnnumbs/tmp.rrnumbs from (select rn.step as rnstep,rn,rumbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs from dw\_oute\_numbs rn inner join dw\_out\_numbs rr )where cast(substr(tmp.rnstep,5,1),int)=cast(substr(tmp.rrstep,5,1),int)-1;

#### 6 模块开发—结果导出

采用apache sqoop将数据导出

#### 7模块开发—数据可视化

采用Echarts对数据进行可视化展示。

### 相关面试题

1. 电商总体运营指标
2. 网站流量指标

### 推荐系统项目

推荐系统项目：本质上是商品售卖系统，和电商网站的目的一样，用来售卖商品。

会收集用户的所有行为信息（网站浏览信息、订单信息、关注信息、收藏商品、评论系统、外部信息（微博信息、联盟网站）），通过分析分析用户的历史行为给用户的兴趣建模，从而主动给用户推荐能够满足他们兴趣和需求的信息。

**数据源：**

基础数据的来源有很多维度，包括用户的访问、浏览、下单、收藏、用户的历史订单数据，评价信息等很多数据。

基础数据主要包括：

1. 要推荐物品或内容的元数据。例如关键字、属性描述等
2. 系统用户的基本信息，例如性别，年龄等
3. 用户对物品或者信息的偏好，包括用户对物品的评分，用户查看物品的记录，用户购买记录等

可以将这些用户的偏好信息分为两类：

**显示的用户反馈**：这类是用户在网站上自然浏览或者使用网站以外，显式的提供反馈信息，例如用户对物品的评分，或者对物品的评论。

**隐式的用户信息：**用户在使用网站是产生的数据，隐式的反应了用户对物品的喜好，例如用户查看了某物品的信息等等用户反馈。

**数据的采集方案：**

数据收集模块**：**

点击流模块：flume+kafka+storm+redis处理用户当前浏览的信息，将信息计算出来，保存到redis中（比如用户对一个品类的偏好）

订单支付（AMQ）：开发一个消费者程序（storm、JavaAPP），用来计算用户的偏好

外部数据：通过爬虫技术爬取用户的社交网站数据。

合作数据：比如京东和腾讯合作，可以获取一些用户在腾讯上的信息。

1. 将用户产生的数据（浏览商品、关注商品、收藏商品、加入购物车、下订单、评论等）保存到消息队列（kafka）中，将不同类型的数据保存到不同的topic中，以作分类。
2. 另一种方式就是将用户产生的数据保存到分库分表的数据库中

**数据的存储与计算：**

1. **离线推荐**

**离线推荐计算数据的周期可以是一个月或者15天**

1. 由FTP服务对推送的数据进行校验，将数据库中校验通过的数据定时的上传到HDFS文件系统中。
2. 基于HDFS建立hive数据仓库，将HDFS中的数据映射成一张张的表（用户表、收藏表、购物车表、订单表等）保存到hive数据仓库中。
3. 将hive数据仓库中的数据通过HSQL计算和导入到算法中（协同过滤算法）进行计算，得到一些用户的偏好数据。
4. 将计算出来的用户偏好数据定时导入到hbase/redis中。
5. 用户通过点击浏览商品，在推荐引擎（Javaweb）中获取用户id，根据用户的id从redis或hbase中拿去已经计算好的推荐结果（离线计算）推荐给用户。在此过程中，还要对结果数据进行过滤，比如商品上下线状态的判定、推荐结果种类丰富性的判定、推荐结果数量的补足等。最后将计算过滤之后的商品展示给用户。
6. **实时推荐**
7. 通过storm，实时的消费消息队列kafka的topic中的数据，实时的计算一些用户的偏好，同样将计算好的偏好数据实时的导出到hbase或redis中，等待用户点击浏览商品，实时的将计算好的结果展示给用户。

**大数据处理平台：**

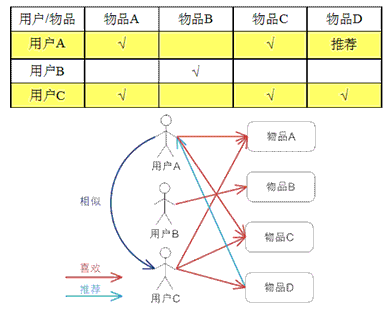
离线计算和实时计算

**数据的计算：**

协同过滤算法：也就是计算相似度，关于相似度的计算，就是计算像个向量的距离，距离越近相似度越大。

1. 基于用户的协同过滤算法

原理：基于用户对物品的偏好找到相邻邻居用户，然后将邻居用户喜欢的推荐给当前用户。



假如：用户A喜欢物品A、物品C

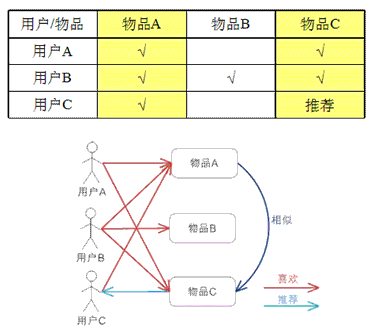
用户B喜欢物品B

用户C喜欢物品A、物品C、物品D

通过计算发现用户A和用户C相似，所以将用户C喜欢的物品D推荐给用户A。

1. 基于物品的协同过滤算法

原理：与基于用户的类似，只是在计算邻居时采用物品本身，而不是从用户角度。基于用户对物品的偏好找到相似的物品，然后根据用户的历史偏好，推荐相似的物品给他。



如图：根据用户历史数据的偏好计算出喜欢物品A的用户大部分都喜欢物品C，计算出物品A和物品C比较相似，而用户 C 喜欢物品 A，那么可以推断出用户 C 可能也喜欢物品 C，所以将物品C推荐给用户C。

**数据展现：**

Web网站

面试题：

1. 推荐系统的本质是什么
2. 推荐系统的指标是什么
3. 推荐系统的y和x
4. 推荐系统的样本构造和数据拼接
5. 推荐系统的场景思考
6. 推荐系统相关组件

## 面试题

##### 1）公司技术选型可能利用storm 进行实时计算,讲解一下storm

描述下storm的设计模式，是基于work、excutor、task的方式运行代码，由spout、bolt组成等等

##### 2）storm 如果碰上了复杂逻辑,需要算很长的时间,你怎么去优化,怎么保证实时性

拆分复杂的业务到多个bolt中，这样可以利用bolt的tree将速度提升

##### 3）Spark Streaming 和 Storm 有何区别？

一个实时毫秒一个准实时亚秒，不过 storm 的吞吐率比较低。