|  |
| --- |
| 全国云计算应用创新大赛 |
| 项目计划方案 |
| 求职者智能分析系统 |

|  |
| --- |
| 参赛团队：EFISH  项目开发者：刘港  liu920848992@163.com  2017-11-25 |

目录

[内容概述 1](#_Toc499489096)

[功能描述 1](#_Toc499489097)

[技术路线 1](#_Toc499489098)

[总体设计框架 2](#_Toc499489099)

[设计实现 4](#_Toc499489100)

[网络数据爬虫 4](#_Toc499489101)

[网络数据存储 5](#_Toc499489102)

[Spark-Hadoop集群平台 7](#_Toc499489103)

[网络数据ETL 11](#_Toc499489104)

[数据分析和可视化 12](#_Toc499489105)

[数据挖掘和机器学习 16](#_Toc499489106)

[应用开发部署 17](#_Toc499489107)

[智能检索和推荐 17](#_Toc499489108)

[Web开发部署 18](#_Toc499489109)

# 内容概述

### 功能描述

本系统首先通过网络爬虫爬取51job等招聘网站上大量的职位招聘信息。提取出其中的关键数据，通过对这些信息的挖掘分析，可以更加精准、清晰地指导求职者所在行业的待遇水平，自身可能的待遇，以及对公司、行业的选择。

### 技术路线

网络爬虫：Python开源框架Scrapy

网络数据存储：MySQL5.7 （Windows10）、HBase、HDFS

云计算平台：spark-Hadoop生态平台

Web开发：Python框架Django

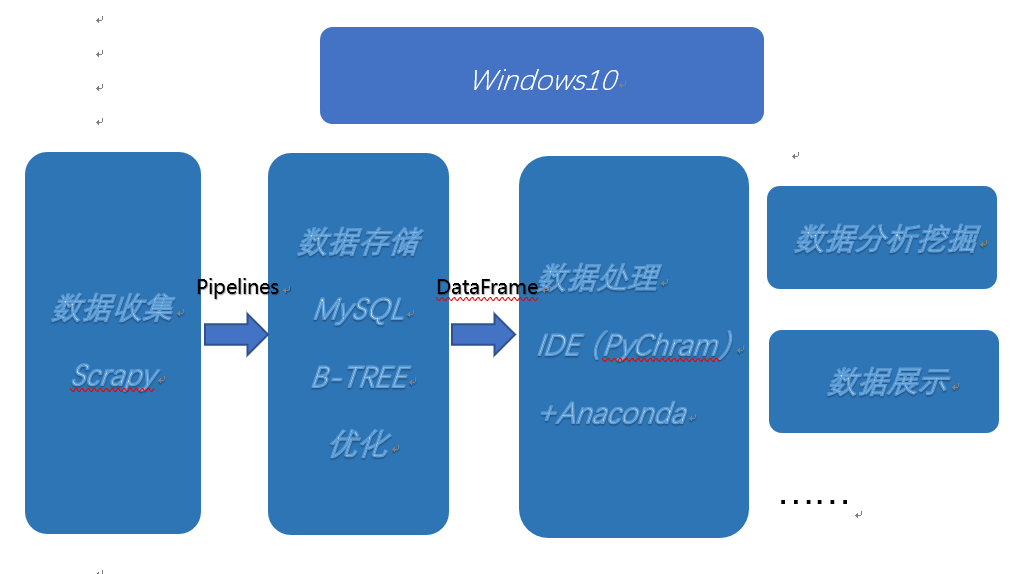
开发语言：Python、SQL

开发环境：Windows+Linux

### 总体设计框架

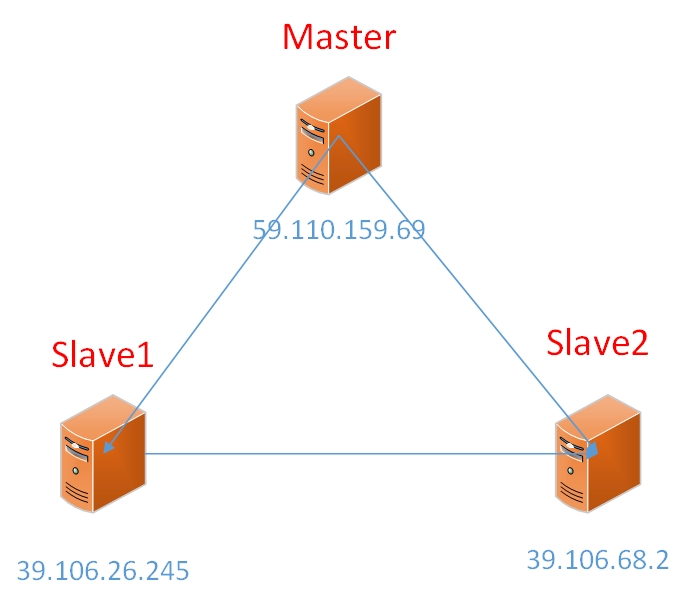
本系统设计开发采用Windows+Linux开发模式。即在Windows上进行系统的编码实现，在Linux上进行测试开发部署。

Windows环境设计如下：



在Windows环境下主要是对数据进行初步地认识、理解，并对相关算法进行初步的设想和一些测试。在此主要采用MySQL关系型数据库和python语言进行认识和测试。具体应用环境采用linux中的应用环境。

应用环境部署（centos7.2 x 3）



本实验项目的应用环境采用3台服务器构成。

Master(主节点服务器):

1. HDFS分布式文件系统的NameNode节点

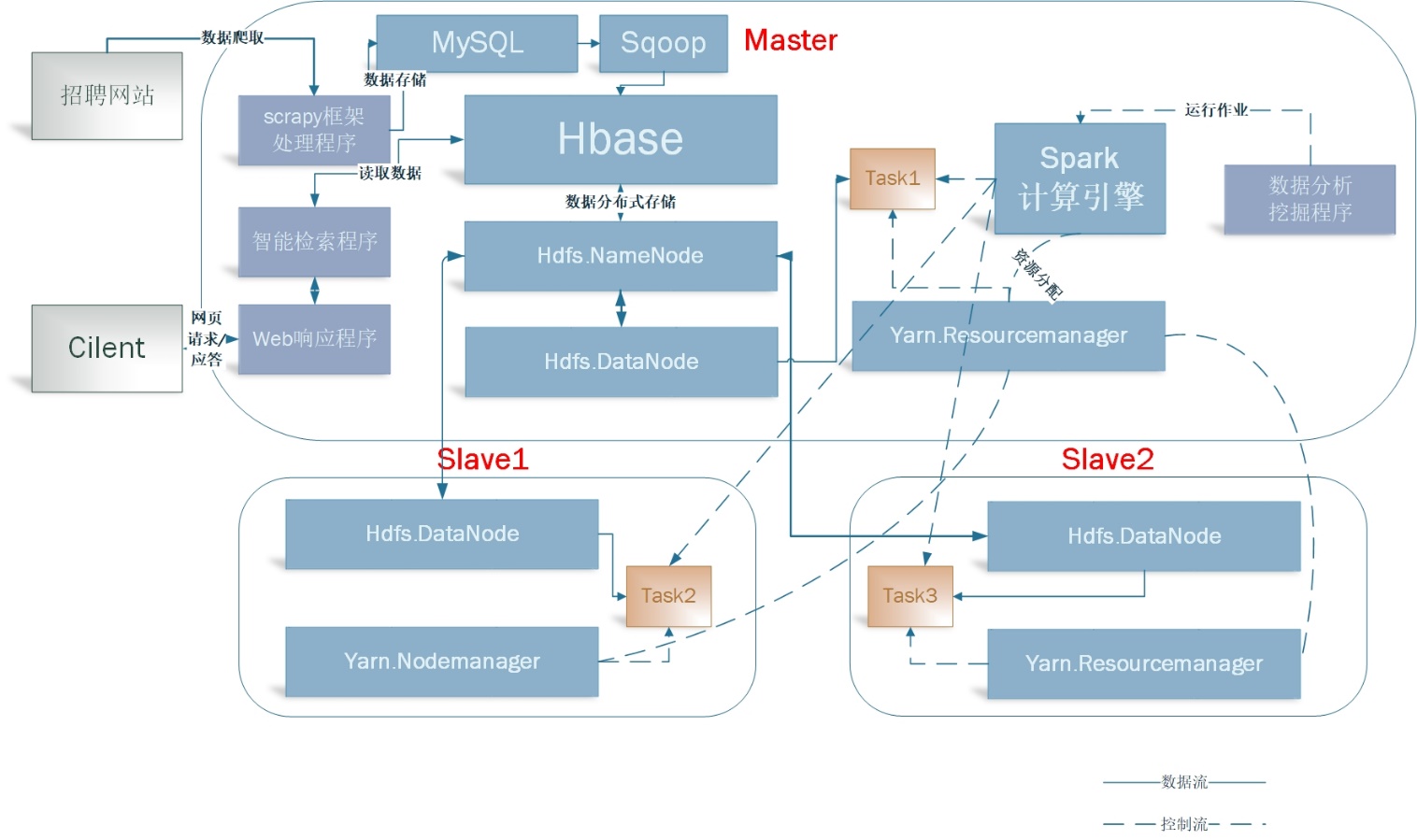
SecondaryNamenode节点、DataNode节点（实验条件限制，最大化利用资源）

1. Yarn资源管理的Resourcemanager节点
2. Spark计算引擎的Master节点，同时也作为Worker节点
3. Web应用部署服务器

Slave1 & Slave2（节点服务器）:

1. HDFS分布式文件系统的DataNode节点
2. Yarn资源管理的Nodemanager节点
3. Spark计算引擎的Worker节点

项目总体实现流程图如下：



实验项目总体过程分为以下四个方面：

1.数据抓取收集：

首先从招聘网站使用Python开源爬虫框架Scrapy爬取数据，并整理存入MySQL数据库中进行存储。

2.网络数据ETL：

将MySQL中的数据通过Sqoop工具抽取转换到HBase数据库中，HBase数据库基于HDFS分布式文件系统，对数据进行分布式存储。

3.数据分析挖掘：

对爬取的数据进行初步认识，结构化所爬取的数据，对收集的数据进行初步分析，如求取其分布函数，平均值，最大（小）值等，还对数据进行分析挖掘有用的信息，如企业特征，薪酬分布，职位技能需求等。

分析时采用Spark计算引擎。作业提交至Spark，由Yarn进行资源分配，并创建一系列的Task任务，进行计算向数据转移的分布式计算，以提高程序的运算速度，并把结果存储在HBase数据库中。

4.Web端的应用：

客户机访问网页并发送http响应请求，由Python开源框架Django做出相应的响应，如客户机发送智能检索请求，用Django调用智能检索处理程序，并将其结果进行响应返回。

# 设计实现

### 网络数据爬虫

1. 目标

对招聘网站公布的招聘信息进行爬取

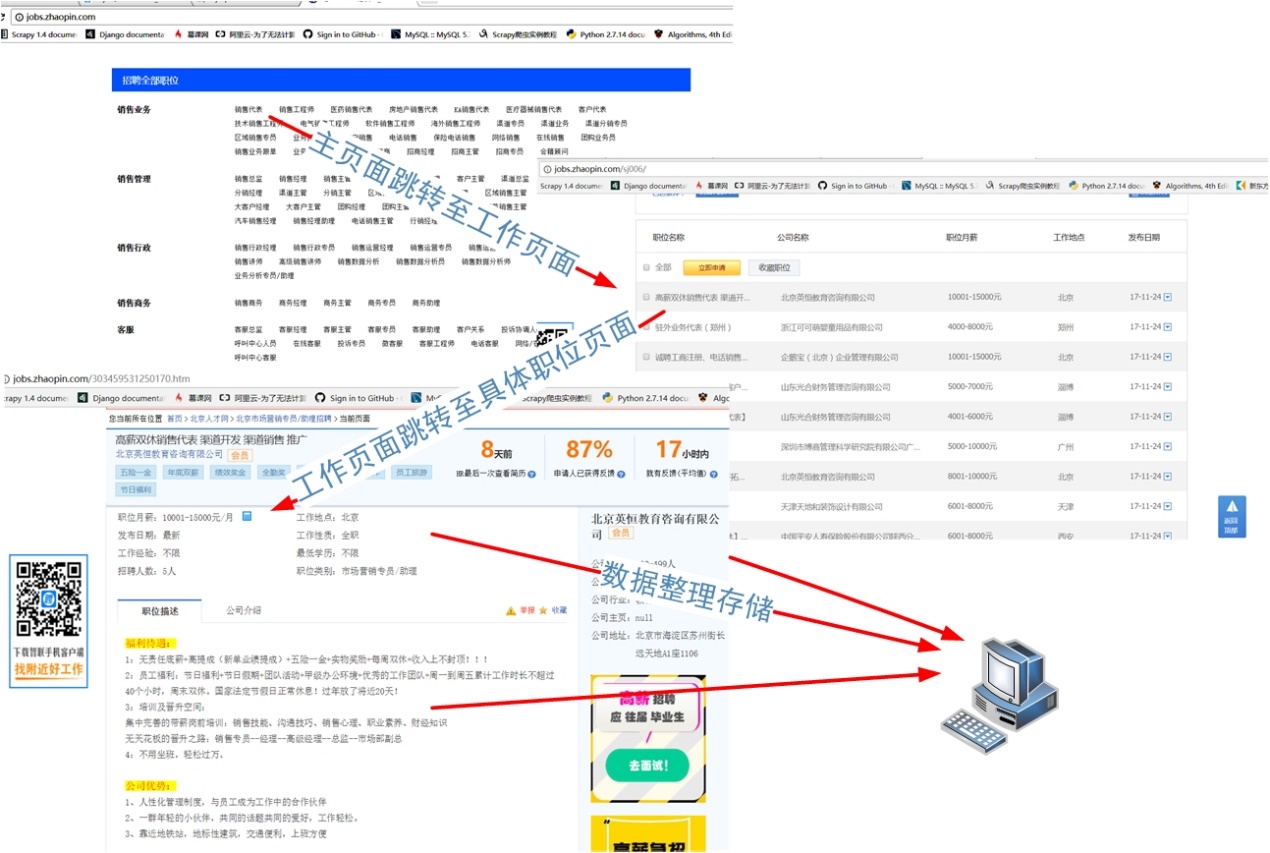
1. 工具

Python开源框架Scrapy、MySQL数据库

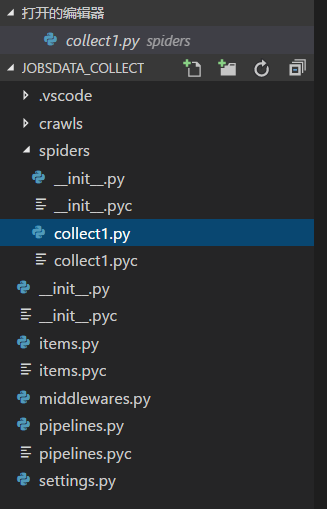
1. 爬取网站

智联招聘职位信息网（<http://jobs.zhaopin.com/>）

1. 爬取流程概述：



1. 爬虫框架Scrapy：

jobsdata\_collect

--crawls

----co1 #保存scrapy爬取状态

--spider

----collect1.py #爬取主程序

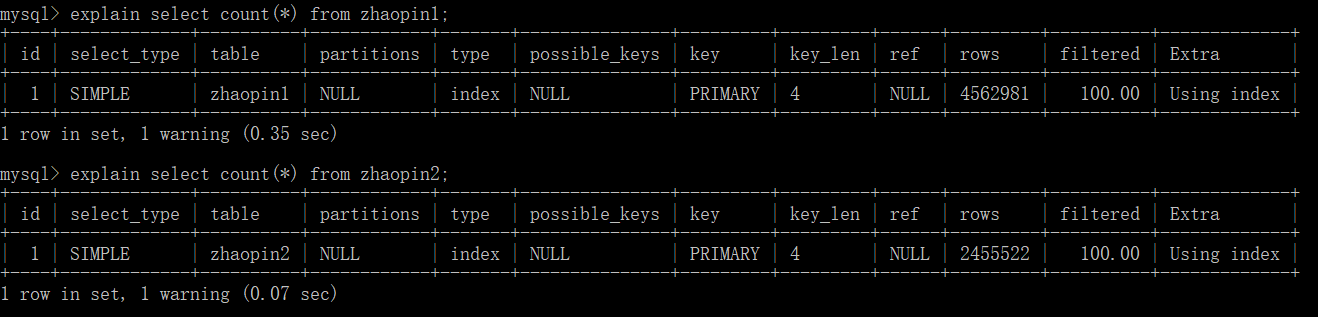
--items.py #爬取数据类

--middlewares.py #爬取中间件

--pipelines.py #管道程序，用于连接数据库等

--settings.py #爬虫主设置

1. 数据集数量：

爬取了近4500000+2500000条数据存储于MySQL数据库中。

相关代码请访问<https://github.com/efishliu/-/tree/master/zhaopin>

### 网络数据存储

网络数据的存储分为两个方面：本地MySQL存储和分布式HBase存储

MySQL存储：

MySQL数据存储类型



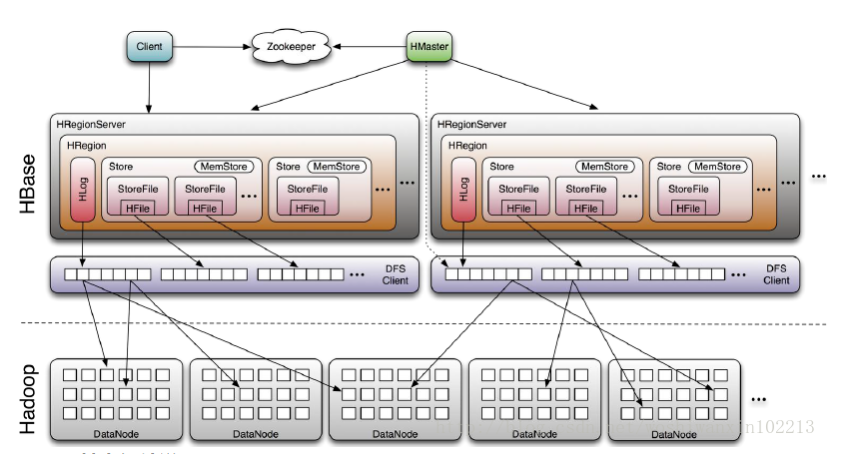
大数据的存储使得数据的查询更新等操作变得很缓慢，于是进行了索引优化和分表优化。

1.分表优化：把数据分为多个表，并对这些表创建一个索引表，每个表的大小为500000行，对于5000000行的数据，即可拆分为10个表。使得查询等操作更快。

2.索引优化：对每一个表进行b-tree索引优化，其操作速度大幅度加快。对于500000行数据，即使采用二叉树构建，速度也可加快100倍以上。（实际建立的树取决于字段的类型大小，字段类型占的空间越小，树的高度越低，查询次数越少。）

HBase存储：

HBase基于HDFS分布式存储系统，基本原理流程图如下：



HBase是一个构建在HDFS上的分布式列存储系统，为典型的key/value系统，主要用于海量结构化数据存储。从逻辑上讲，HBase将数据按照表、行和列进行存储。  
与Hadoop一样，HBase目标主要依靠横向扩展，通过不断增加廉价的商用服务器，来增加计算和存储能力。  
HBase表的特点  
大：一个表可以有数十亿行，上百万列；  
无模式：每行都有一个可排序的主键和任意多的列，列可以根据需要动态的增加，同一张表中不同的行可以有截然不同的列；  
面向列：面向列（族）的存储和权限控制，列（族）独立检索；  
稀疏：空（null）列并不占用存储空间，表可以设计的非常稀疏；  
数据多版本：每个单元中的数据可以有多个版本，默认情况下版本号自动分配，是单元格插入时的时间戳；  
数据类型单一：HBase中的数据都是字符串，没有类型

### Spark-Hadoop集群平台

目标：搭建spark-hadoop平台，并对分布式计算工作原理进行初步的认识理解。

相关软件及版本：

openssh-server-7.4

Hadoop2.8.2

Spark-2.1.2-bin-hadoop2.7

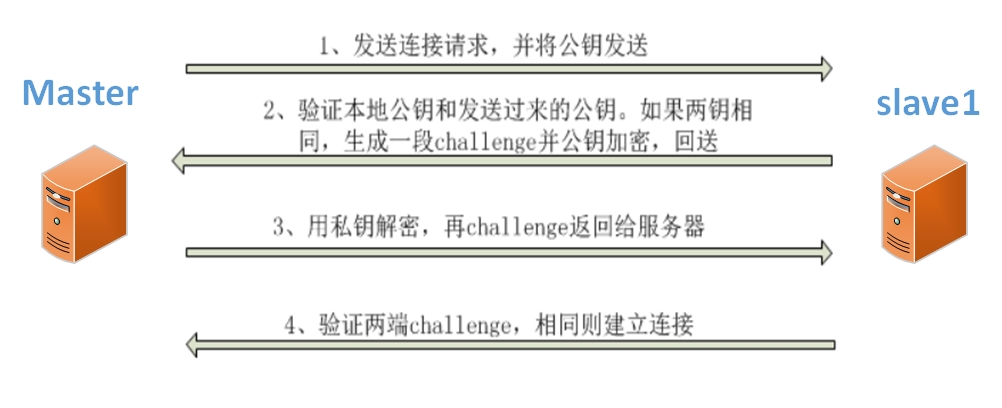
hbase-1.1.12

集群部署实现（简略版）

相关源码及过程请访问

https://github.com/efishliu/-

1. ssh免密登录



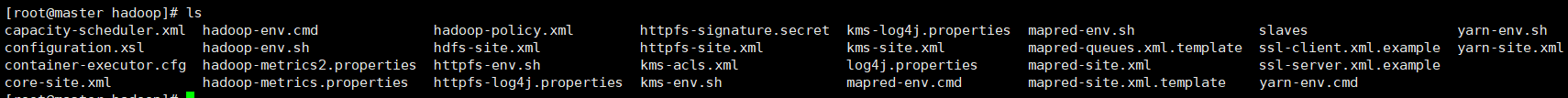
同理配置其他节点，是各节点之间能相互免密连接通信

1. Hadoop安装

解压安装包



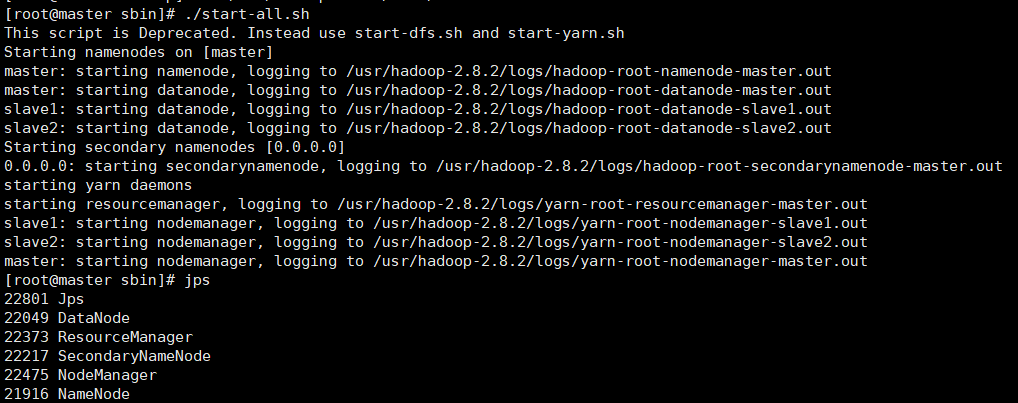
切换至./etc/Hadoop/ 修改配置文件



格式化namenode



启动hadoop并查看

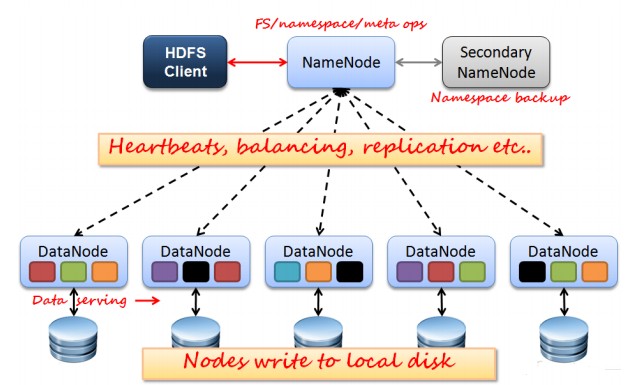


Hadoop集群中，着重介绍hdfs分布式文件系统和yarn介绍

即namenode-datanode resourcemanager-nodemanager两部分

Mapreduce部分的数据处理层部分由spark替代

Hdfs介绍



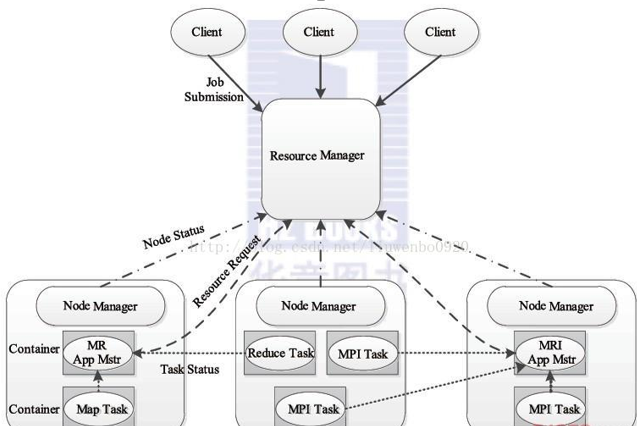
如上图所示，HDFS也是按照Master和Slave的结构。分NameNode、SecondaryNameNode、DataNode这几个角色。

NameNode：为Master节点。管理数据块映射，处理客户端的读写请求，配置副本策略，管理HDFS的名称空间；

SecondaryNameNode：是一个备用namenode，分担namenode的工作量，是NameNode的冷备份，合并fsimage和fsedits然后再发给namenode。

DataNode：Slave节点。负责存储client发来的数据块block；执行数据块的读写操作。

Yarn介绍



1.ResourceManager（RM）

RM是一个全局的资源管理器，负责整个系统的资源管理和分配。它主要由两个组件构成：调度器（Scheduler）和应用程序管理器（Applications Manager，ASM）。

（1）调度器

调度器根据容量、队列等限制条件（如每个队列分配一定的资源，最多执行一定数量的作业等），将系统中的资源分配给各个正在运行的应用程序。

需要注意的是，该调度器是一个“纯调度器”，它不再从事任何与具体应用程序相关的工作，比如不负责监控或者跟踪应用的执行状态等，也不负责重新启动因应用执行失败或者硬件故障而产生的失败任务，这些均交由应用程序相关的ApplicationMaster完成。调度器仅根据各个应用程序的资源需求进行资源分配，而资源分配单位用一个抽象概念“资源容器”（Resource Container，简称Container）表示，Container是一个动态资源分配单位，它将内存、CPU、磁盘、网络等资源封装在一起，从而限定每个任务使用的资源量。此外，该调度器是一个可插拔的组件，用户可根据自己的需要设计新的调度器，YARN提供了多种直接可用的调度器，比如Fair Scheduler和Capacity Scheduler等。

（2） 应用程序管理器

应用程序管理器负责管理整个系统中所有应用程序，包括应用程序提交、与调度器协商资源以启动ApplicationMaster、监控ApplicationMaster运行状态并在失败时重新启动它等。

2. ApplicationMaster（AM）

用户提交的每个应用程序均包含1个AM，主要功能包括：

与RM调度器协商以获取资源（用Container表示）；

将得到的任务进一步分配给内部的任务；

与NM通信以启动/停止任务；

监控所有任务运行状态，并在任务运行失败时重新为任务申请资源以重启任务。

当前YARN自带了两个AM实现，一个是用于演示AM编写方法的实例程序distributedshell，它可以申请一定数目的Container以并行运行一个Shell命令或者Shell脚本；另一个是运行MapReduce应用程序的AM—MRAppMaster，我们将在第8章对其进行介绍。此外，一些其他的计算框架对应的AM正在开发中，比如Open MPI、Spark等。

3. NodeManager（NM）

NM是每个节点上的资源和任务管理器，一方面，它会定时地向RM汇报本节点上的资源使用情况和各个Container的运行状态；另一方面，它接收并处理来自AM的Container启动/停止等各种请求。

4. Container

Container是YARN中的资源抽象，它封装了某个节点上的多维度资源，如内存、CPU、磁盘、网络等，当AM向RM申请资源时，RM为AM返回的资源便是用Container表示的。YARN会为每个任务分配一个Container，且该任务只能使用该Container中描述的资源。

需要注意的是，Container不同于MRv1中的slot，它是一个动态资源划分单位，是根据应用程序的需求动态生成的。截至本书完成时，YARN仅支持CPU和内存两种资源，且使用了轻量级资源隔离机制Cgroups进行资源隔离。

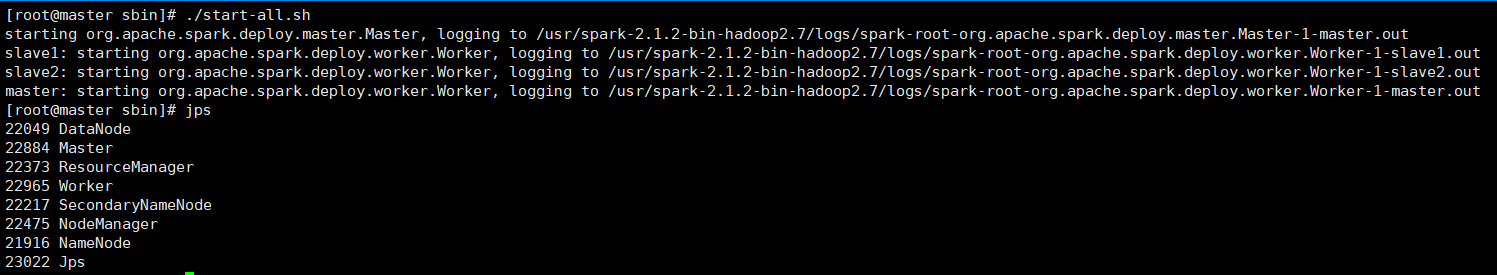
1. Spark安装和启动

解压spark安装包



配置配置文件，把Hadoop配置文件导入spark-env.sh

启动spark并查看

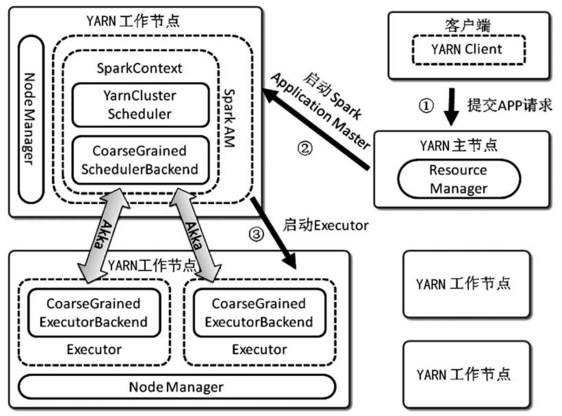


对于数据处理层，Spark相对于MapReduce的优势

Spark 在内存中处理数据，而 Hadoop MapReduce 是通过 map 和 reduce 操作在磁盘中处理数据。因此从这个角度上讲 Spark 的性能应该是超过 Hadoop MapReduce 的。

Spark 有着灵活方便的Java，Scala和 Python 的API，同时对已经熟悉 SQL 的技术员工来说， Spark 还适用 Spark SQL（也就是之前被人熟知的 Shark）。多亏了 Spark 提供的简单易用的构造模块，我们可以很容易的编写自定义函数。它甚至还囊括了可以即时反馈的交互式命令模式。

Spark有多种运行模式，在这里主要介绍下YARN cluster模式的内部实现原理。如下图是YARN cluster模式的原理框图，相对于其他模式，该模式比较特殊的是它需要由外部程序辅助启动APP。用户的应用程序通过辅助的YARN Client类启动。YARN cluster模式和YARN client模式的区别在于：YARN client模式的AM是运行在提交任务的节点，而YARN cluster模式的AM是由YARN在集群中选取一个节点运行，不一定是在提交任务的节点运行。例如spark-shell如果需要使用YARN模式运行，只能为yarn-client模式，启动命令可以使用spark-shell --master yarn-client。



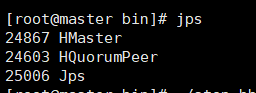
1. HBase的安装

解压HBase安装包



配置相关配置文件

启动并查看HBase（在启动前需启动hdfs）



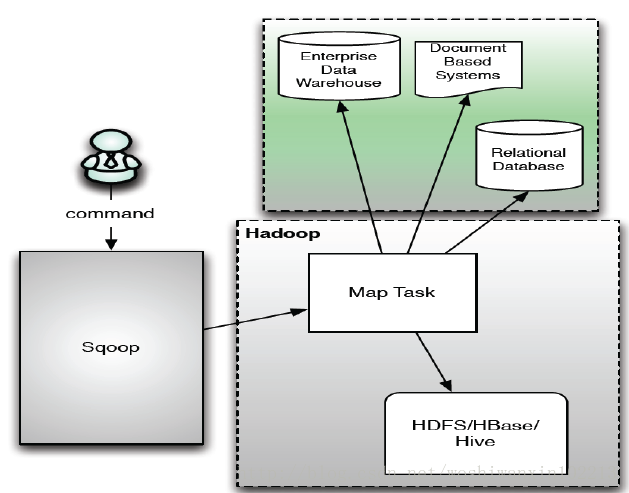
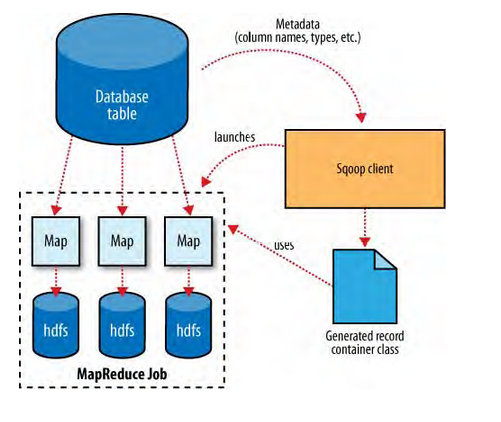
### 网络数据ETL

目标：

解决把爬取下来的数据从MySQL数据库中，把数据抽取、转换、加载到Hadoop平台，即把MySQL的数据通过etl工具加载到Hadoop平台中的HBase数据库。

工具：sqoop1.4.6

流程概述：



从传统数据库如MySQL获取元数据信息(schema、table、field、field type)，把导入功能转换为只有Map的MapReduce作业，在MapReduce中有很多map，每个map读一片数据，进而并行的完成数据的拷贝。

主要步骤：

1、 数据清洗：

·数据补缺：对空数据、缺失数据进行数据补缺操作，无法处理的做标记。

·数据替换：对无效数据进行数据的替换。

·格式规范化：将源数据抽取的数据格式转换成为便于进入仓库处理的目标数据格式。

·主外键约束：通过建立主外键约束，对非法数据进行数据替换或导出到错误文件重新处理。

2、 数据转换

·数据合并：多用表关联实现，大小表关联用lookup，大大表相交用join（每个字段家索引，保证关联查询的效率）

·数据拆分：按一定规则进行数据拆分

·行列互换、排序/修改序号、去除重复记录

·数据验证：lookup、sum、count

实现方式：

·在ETL引擎中进行（SQL无法实现的）

·在数据库中进行（SQL可以实现的）

3、 数据加载

方式：

·时间戳方式：在业务表中统一添加字段作为时间戳，当OLAP系统更新修改业务数据时，同时修改时间戳字段值。

·日志表方式：在OLAP系统中添加日志表，业务数据发生变化时，更新维护日志表内容。

· 全表对比方式：抽取所有源数据，在更新目标表之前先根据主键和字段进行数据比对，有更新的进行update或insert。

·全表删除插入方式：删除目标表数据，将源数据全部插入。

4、异常处理

在ETL的过程中，必不可少的要面临数据异常的问题，处理办法：

1、将错误信息单独输出，继续执行ETL，错误数据修改后再单独加载。中断ETL，修改后重新执行ETL。原则：最大限度接收数据。

2、对于网络中断等外部原因造成的异常，设定尝试次数或尝试时间，超数或超时后，由外部人员手工干预。

3、 例如源数据结构改变、接口改变等异常状况，应进行同步后，在装载数据。

### 数据分析和可视化

目标：

对所爬取的数据进行初步的认识，包括分析各行业，各地区的企业信息，职位的各种信息等，利用可视化技术进行直观认识，为数据挖掘和智能推荐作基础。

相关信息前500行如下：



相关分析分为几个主题进行分析，主要分析设想如下：

主题一：职位的分析：

1. 职位分类：对已有数据进行职位分类。

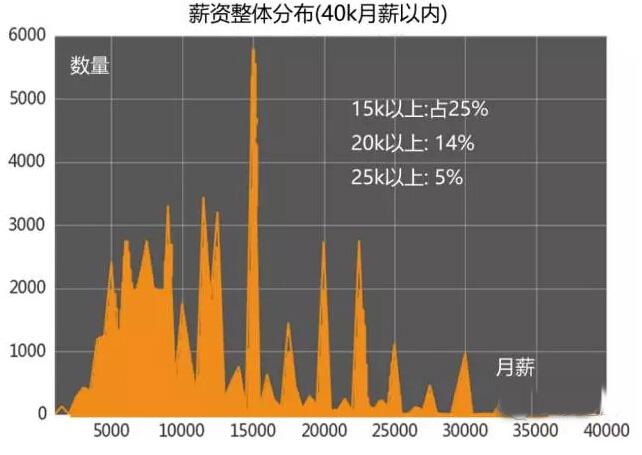


对于跨多个行业的职位，采用多行业保存的冗余处理。如



此职位则既归入证券行业，又归于期货、外汇经纪人行业。

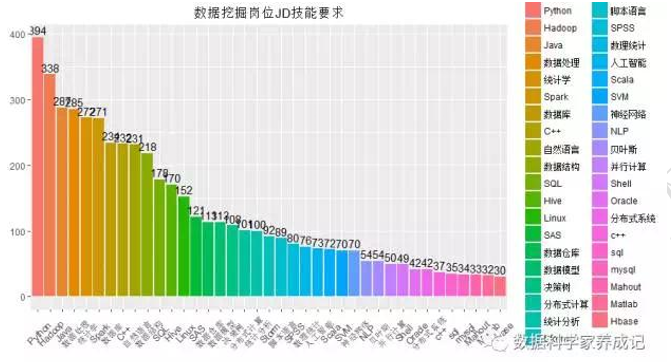
1. 职位薪酬分析：对于相同行业的职位进行横向分析。主要分析其薪酬的分布函数，极大（小）值，平均值等。



1. 职位地域分析：对于该行业的职位，分析各城市的需求状况。



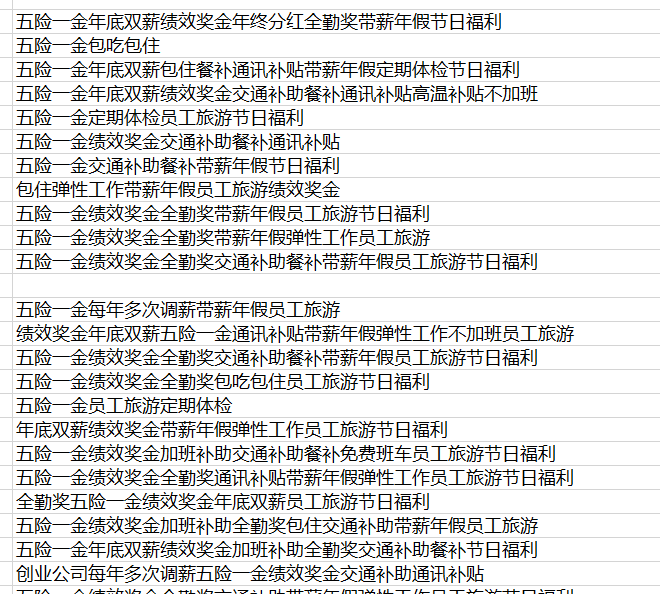
1. 职位技能需求分析：通过职位的要求描述进行解析、分词、提取关键字等方式，得到每个大数据职位的技能需求。



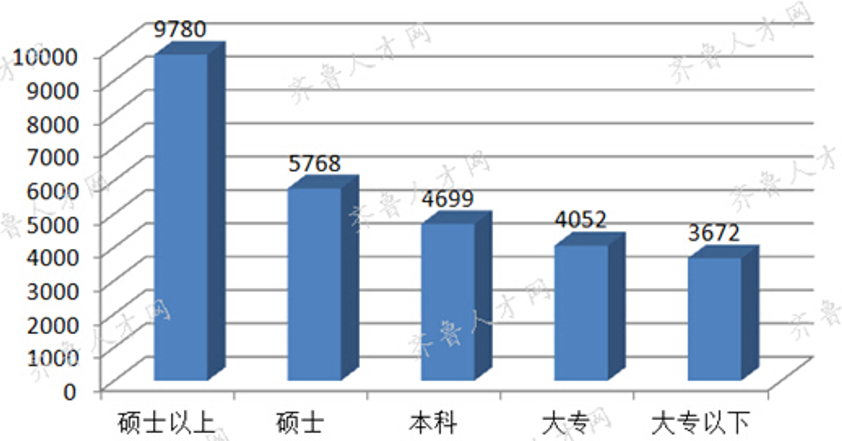
1. 职位工作经验要求分析：对于收集数据的工作要求，对于各行业工作经验的分布进行系统的分析



1. 职位福利分析：对于各个职位的工作福利进行提取关键字等方式，得到每个职位的工作福利信息。



1. 职位其他要求分析：如对学历、年龄及特殊要求的分析



使用语言及相关包：python。Pandas(数据结构化)、matplotlib(可视化)

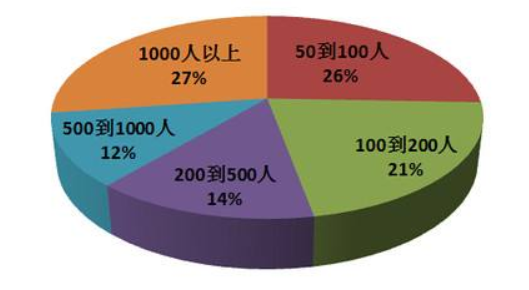
相关算法：文本数据结构化，提取关键字，分类算法，概率密度分布等。

主题二：企业的分析

1. 对各职位的企业性质进行分析，企业性质包括民营、国企、合资、上市等



1. 对企业的规模、行业等进行分类



以上的数据分析等只是一个初步的设想，详情请访问（暂未更新）

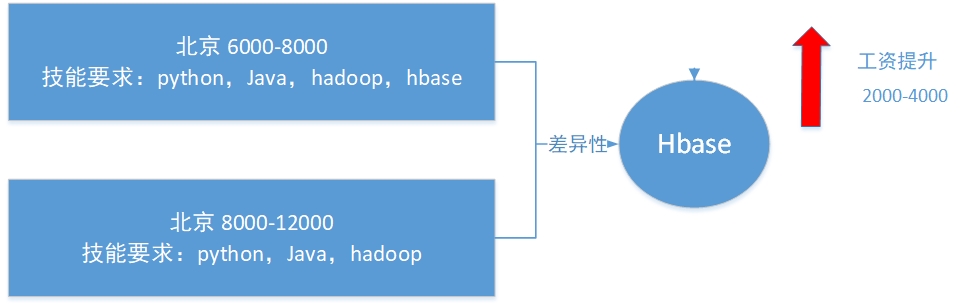
<https://github.com/efishliu/->

### 数据挖掘和机器学习

目标：通过之前的数据分析的铺垫，对数据进行进一步的挖掘，提取出更加有价值的信息，为求职者提供一些中肯的建议。

主题一：职业前景规划

对于职位薪酬进行分段处理，对于不同的薪酬段进行纵向对比，挖掘出其中的共同点和差异性。并为职位的提升突破找出关键点，相关实例如下：



参考汽车之家的模式，也可以进行职位间的相互对比，使求职者找到更合适的工作。



主题二：生活质量建议（工资购买力、生活幸福指数（WHI））

对于不同地区，不同城市，相同薪酬的购买力不尽相同，比如在上海这种一线城市和一些三线城市等，相同的薪酬其购买力，生活幸福指数等完全不同。

这个主题的目的在于对这个问题进行建模处理，计算出各城市各薪酬的购买力等数据，为求职者的职位选择提供相关的建议和帮助。

相关算法：聚类算法、贝叶斯过滤算法、关键词提取、条件随机场、深度学习等

# 应用开发部署

### 智能检索和推荐

目标：根据求职者提供的个人信息和对数据库中整理出的数据的检索，建立相应的推荐模型，为求职者提供更加合理的职位推荐。

实现步骤如下：

1. 发现群组

通过对求职者提供的信息，对职位，工作城市要求等进行“数据聚类”。这是一种用于

寻找紧密关系的方法，对于数据量很大的应用中，我们可以采用相应的聚类算法，自动寻找出具有相似特征的群组，为之后的决策作数据基础。

1. 决策树建模

通过决策树进行数据建模。在此过程中决策时不只是一种预测方法，还用它来为决策

过程进行建模。我们用它来预测数据库中的某一职位是否满足求职者的要求。

1. 优化算法（检索）

优化算法主要是为检索优化，对于大数据的数据检索，往往相应时间很长，而这里的

优化算法，则是为了满足推荐系统的低延时而设计的。

1. 提供推荐

通过求职者所提供的信息，使用协作型过滤算法、基于内容的推荐算法等，并根据决策

树的建模计算结果，对相应的职位进行检索和推荐，并把分析结果展示给用户。

相关算法：协作型算法，决策树建模、基于内容的推荐算法、优化算法等

### Web开发部署

目标：

实现与用户进行交互的推荐系统页面，具有良好的网页交互体验。

工具与框架：

网页创建：Python开源网页框架Django

相关源码

<https://github.com/efishliu/->/liugangweb

预效果图如下：

