|  |
| --- |
| 全国云计算应用创新大赛 |
| **系统技术路线** |
| 命题三-求职者智能分析系统 |

|  |
| --- |
| 参赛团队：EFISH  项目开发者：刘港  liu920848992@163.com  2018-3-10 |

# Part1、爬虫

在选取爬虫工具之初，试过很多十分简单的爬虫工具，但使用起来觉得很不灵活，于是便选取了python的一个开源爬虫框架—scrapy.

简单来看一下在运行scrapy startproject tutorial 后scrapy的结构框架

tutorial**/**

scrapy**.**cfg *# deploy configuration file*

tutorial**/** *# project's Python module, you'll import your code from here*

\_\_init\_\_**.**py

items**.**py *# project items definition file*

middlewares**.**py *# project middlewares file*

pipelines**.**py *# project pipelines file*

settings**.**py *# project settings file*

spiders**/** *# a directory where you'll later put your spiders*

\_\_init\_\_**.**py

它主要由items.py、middlewares.py、pipelines.py、spiders/等组成。

下面来介绍一下本系统爬虫的具体技术

Items.py 创建爬取信息的类

class Posts(scrapy.Item):

#企业信息

company\_name=scrapy.Field()

company\_form=scrapy.Field()#公司性质，eg国企

company\_industry=scrapy.Field()#公司涉及行业

company\_scale=scrapy.Field()#公司规模

company\_introduce=scrapy.Field()

company\_web=scrapy.Field()

company\_address=scrapy.Field()

#职位信息

#job\_trade=scrapy.Field()#职位行业（大类）1

workcity=scrapy.Field()

job\_name=scrapy.Field()

job\_welfare=scrapy.Field()

job\_inwhichcompany=scrapy.Field()

job\_category=scrapy.Field()#职位行业细分

min\_salary=scrapy.Field()

max\_salary=scrapy.Field()

workplace=scrapy.Field()

job\_releasetime=scrapy.Field()

education\_background=scrapy.Field()

zhaopin\_numbers=scrapy.Field()

min\_workexperience=scrapy.Field()

job\_form=scrapy.Field()#工作性质，eg全职

job\_require=scrapy.Field()

data\_addtime=scrapy.Field()

data\_sourceweb=scrapy.Field()

salary=scrapy.Field()

Pipelines.py 管道连接MySQL数据库

from twisted.enterprise import adbapi

import MySQLdb

import MySQLdb.cursors

from scrapy.crawler import Settings as settings

from jobsdata\_collect.items import Posts

import string

class JobsdataCollectPipeline(object):

def \_\_init\_\_(self):

dbargs=dict(

host='127.0.0.1',

db='zhaopin',

user='root',

passwd='liugang666',

charset='utf8',

cursorclass=MySQLdb.cursors.DictCursor,

use\_unicode=True,

)

self.dbpool=adbapi.ConnectionPool('MySQLdb',\*\*dbargs)

def process\_item(self, item, spider):

res = self.dbpool.runInteraction(self.insert\_into\_table,item)

return item

将爬取到的信息插入MySQL数据库

def insert\_into\_table(self,conn,item):

conn.execute('INSERT INTO collect2(workcity,job\_name,job\_inwhichcompany,min\_salary,max\_salary,job\_category,workplace,zhaopin\_numbers,\

job\_welfare,education\_background,min\_workexperience,job\_form,job\_releasetime,company\_name,company\_form,\

company\_industry,company\_scale,company\_web,company\_address,data\_addtime,data\_sourceweb,job\_require,company\_introduce,salary)\

VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)'\

,(item['workcity'],item['job\_name'],item['job\_inwhichcompany'],item['min\_salary'],item['max\_salary'],item['job\_category'],item['workplace'],\

item['zhaopin\_numbers'],item['job\_welfare'],item['education\_background'],item['min\_workexperience'],\

item['job\_form'],item['job\_releasetime'],item['company\_name'],item['company\_form'],item['company\_industry'],item['company\_scale'],\

item['company\_web'],item['company\_address'],item['data\_addtime'],item['data\_sourceweb'],item['job\_require'],item['company\_introduce'],item['salary']))

spider/collect1.py 爬虫主程序

# -\*- coding: utf-8 -\*-

import scrapy

from scrapy.loader import ItemLoader

from jobsdata\_collect.items import Posts

import datetime

import string

import re

class Collect1Spider(scrapy.Spider):

name = 'collect1'

allowed\_domains = ['jobs.zhaopin.com']

start\_urls = ['http://jobs.zhaopin.com/']

解析逻辑

def parse(self, response):

#

for inlist in response.css('div.listcon a::attr(href)').extract():

yield response.follow(inlist,self.parse\_href)

def parse\_href(self,response):

for nextlist in response.css('div.searchlist\_page span.search\_page\_next a::attr(href)').extract():

yield response.follow(nextlist,self.parse\_href)

#

for wolist in response.css('div.main-left.main\_current\_items div.details\_container span.post a::attr(href)').extract():

yield response.follow(wolist,self.parse\_message)

#parse 具体信息

def parse\_message(self,response):

item=Posts()

item['workcity']=''.join(response.css('div.terminalpage-left ul.terminal-ul.clearfix li')[1].css('strong a::text').extract\_first())

item['company\_name']=''.join(response.css('div.company-box p.company-name-t a::text').extract\_first().strip())

item['company\_form']=''.join(response.css('ul.terminal-ul.clearfix.terminal-company.mt20 li strong::text').extract()[1].strip())

item['company\_industry']= ''.join(response.css('ul.terminal-ul.clearfix.terminal-company.mt20 li strong a::text').extract()[0].strip())

item['company\_scale']= ''.join(response.css('ul.terminal-ul.clearfix.terminal-company.mt20 li strong::text').extract()[0].strip())

item['company\_web']=''.join(response.css('div.company-box p.company-name-t a::attr(href)').extract\_first())

item['company\_address']=''.join(response.css('ul.terminal-ul.clearfix.terminal-company.mt20 li strong::text').extract()[-1].strip())

item['job\_name']=''.join(response.css('div.fixed-inner-box h1::text').extract())

item['job\_inwhichcompany']=''.join(response.css('div.fixed-inner-box h2 a::text').extract())

item['job\_welfare']=''.join(response.css('div.fixed-inner-box div.welfare-tab-box span::text').extract())

try:

s=''.join(response.css('div.terminalpage-left ul.terminal-ul.clearfix li strong::text').extract()[0])

item['min\_salary']=re.findall('(\d+)',s)[0]

item['max\_salary']=re.findall('(\d+)',s)[1]

item['salary']=str((float(item['min\_salary'])+float(item['max\_salary']))/2)

except:

pass

item['job\_releasetime']=''.join(response.css('span#span4freshdate::text').extract())

if item['job\_releasetime']=='0002-01-01 00:00:00':

item['job\_releasetime']=datetime.datetime.now().strftime('%Y-%m-%d')

item['job\_form']=''.join(response.css('div.terminalpage-left ul.terminal-ul.clearfix li strong::text').extract()[-4]) #工作性质,全职

item['min\_workexperience']=''.join(response.css('div.terminalpage-left ul.terminal-ul.clearfix li strong::text').extract()[-3])

item['education\_background']=''.join(response.css('div.terminalpage-left ul.terminal-ul.clearfix li strong::text').extract()[-2])

item['zhaopin\_numbers']=''.join(response.css('div.terminalpage-left ul.terminal-ul.clearfix li strong::text').extract()[-1])

item['job\_category']=''.join(response.css('div.terminalpage-left ul.terminal-ul.clearfix li strong a::text').extract()[-1]) #职位类别

item['workplace']=''.join(response.css('div.tab-cont-box div.tab-inner-cont h2::text').extract()[0].strip())

item['data\_addtime']=datetime.datetime.now().strftime('%Y-%m-%d')

introduce=response.css('div.terminalpage-main.clearfix div.tab-inner-cont')

a1=''.join(introduce[1].css('p::text').extract()).strip(),''.join(introduce[1].css('span::text').extract()).strip(),''.join(introduce[1].css('div::text').extract()).strip()

a2=''.join(introduce[0].css('p::text').extract()).strip(),''.join(introduce[0].css('span::text').extract()).strip(),''.join(introduce[0].css('div::text').extract()).strip()

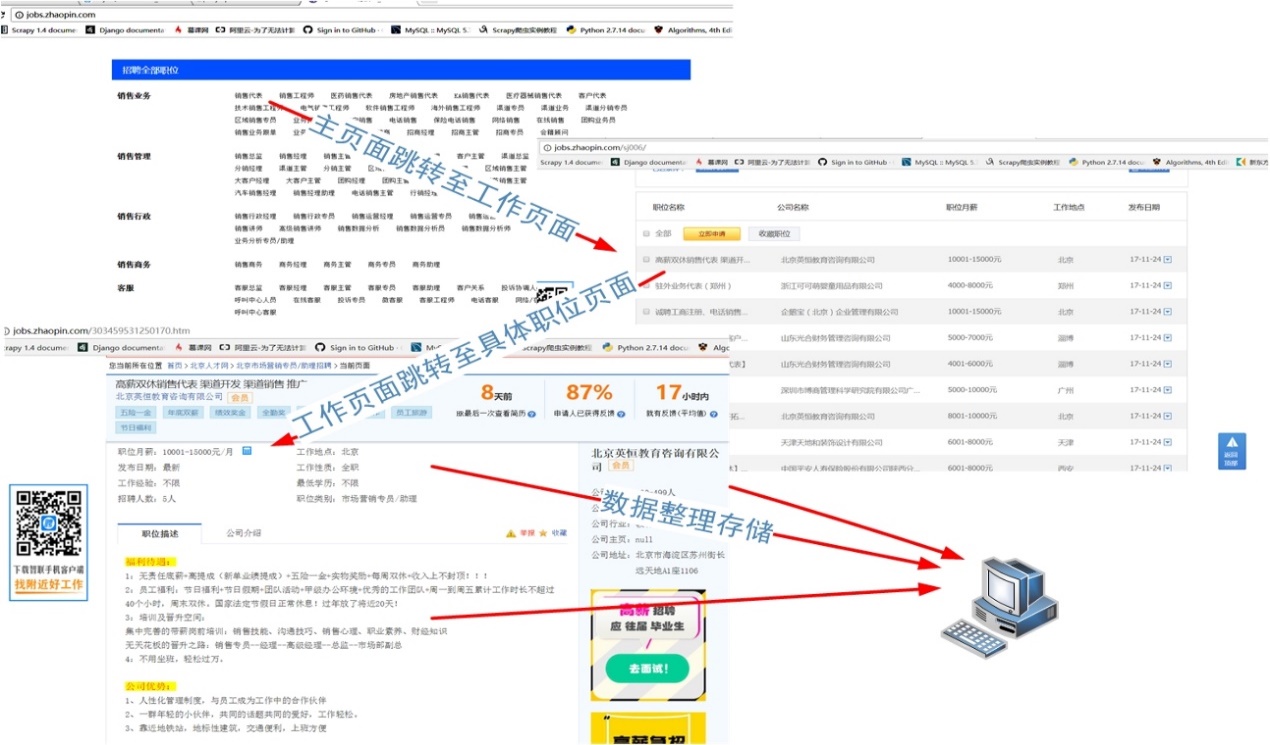
item['company\_introduce']=str(a1)

item['job\_require']= str(a2)

item['data\_sourceweb']=''.join(response.url)

yield item

爬虫解析逻辑



Scrapy 框架其实主要就由items.py创建类，由pipelines.py管道连接，由spider爬虫主程序进行爬取。

# Part2、mysql数据库存储

1. MySQL数据存储结构

CREATE TABLE collect1(

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

workcity VARCHAR(50) NOT NULL,

job\_name VARCHAR(100) NOT NULL,

job\_inwhichcompany VARCHAR(100) NOT NULL,

min\_salary VARCHAR(20) NOT NULL,

max\_salary VARCHAR(20) NOT NULL,

job\_category VARCHAR(20) NOT NULL,

workplace VARCHAR(100) NOT NULL,

zhaopin\_numbers VARCHAR(20) NOT NULL,

job\_welfare VARCHAR(100) NOT NULL,

education\_background VARCHAR(20) NOT NULL,

min\_workexperience VARCHAR(20) NOT NULL,

job\_form VARCHAR(20) NOT NULL,

job\_releasetime VARCHAR(20) NOT NULL,

job\_require TEXT NOT NULL,

company\_name VARCHAR(100) NOT NULL,

company\_form VARCHAR(20) NOT NULL,

company\_industry VARCHAR(100) NOT NULL,

company\_scale VARCHAR(30) NOT NULL,

company\_web TEXT NOT NULL,

company\_introduce TEXT NOT NULL,

company\_address VARCHAR(100) NOT NULL,

data\_addtime VARCHAR(20) NOT NULL,

data\_sourceweb TEXT NOT NULL,

PRIMARY KEY(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8

;

其中包括

**企业信息**：企业名称、企业形式、企业所属行业、企业规模、企业介绍、企业页面地址、数据添加时间、数据来源网站

**职位信息**：职位名称、职位工作城市、职位所属企业名称、工作性质 、最低月薪 、最高月薪、工作地点、职位发布时间、学历、招聘人数、最低工作经验、职位类型、职位要求、数据添加时间、数据来源网站

1. 数据库优化

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。）

# Part3、大数据平台搭建

相关软件及版本：

openssh-server-7.4

Hadoop2.8.2

Spark-2.1.2-bin-hadoop2.7

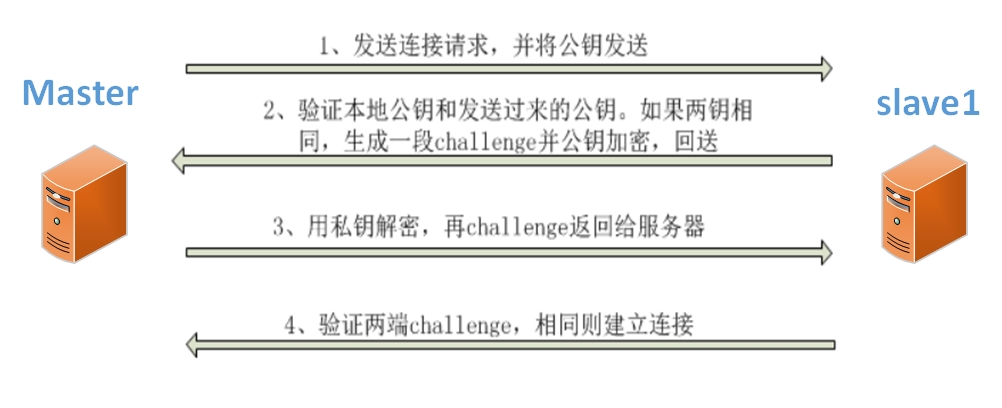
hbase-1.1.12

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



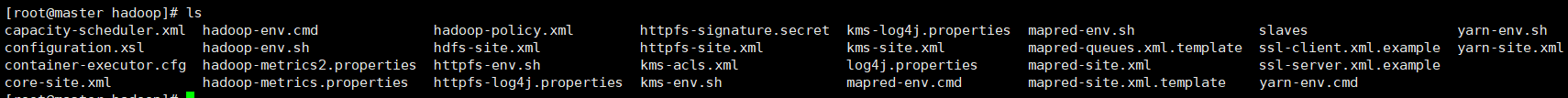
同理配置其他节点，是各节点之间能相互免密连接通信

1. Hadoop安装

解压安装包



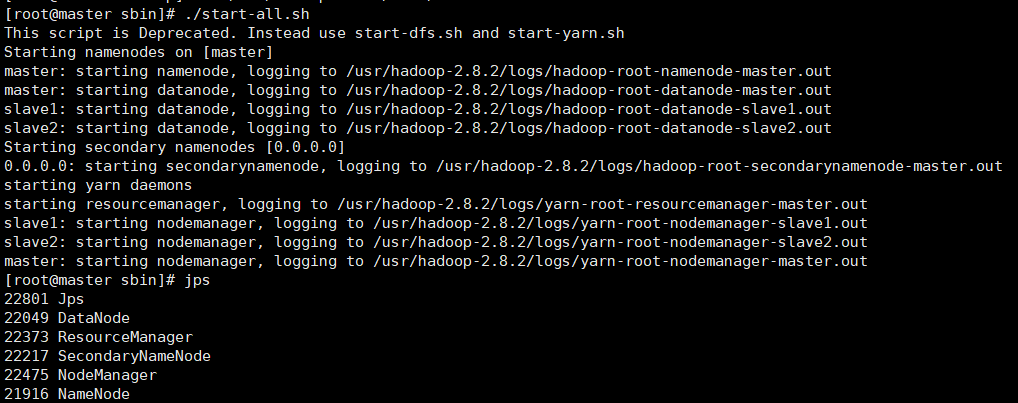
切换至./etc/Hadoop/ 修改配置文件



格式化namenode



启动hadoop并查看

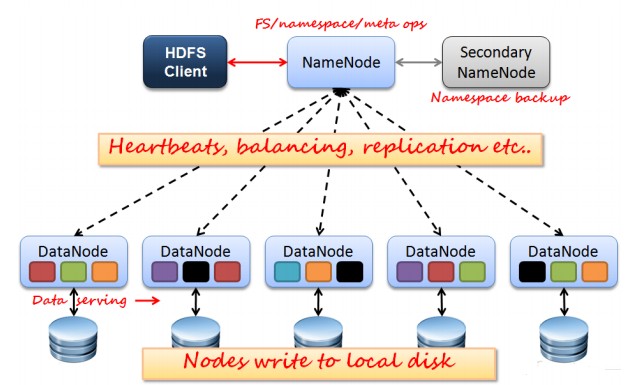


Hadoop集群中，着重介绍hdfs分布式文件系统和yarn介绍

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



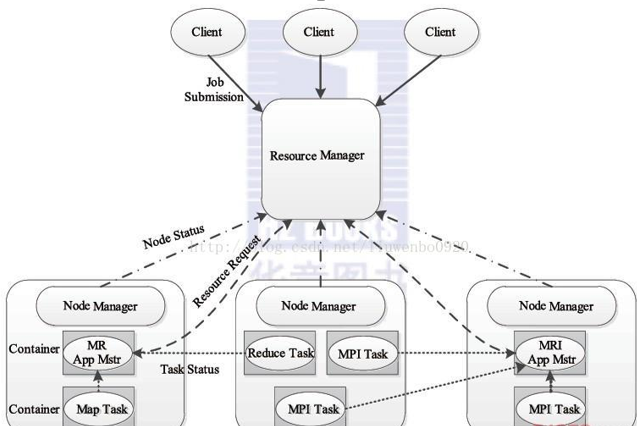
如上图所示，HDFS也是按照Master和Slave的结构。分NameNode、SecondaryNameNode、DataNode这几个角色。

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。

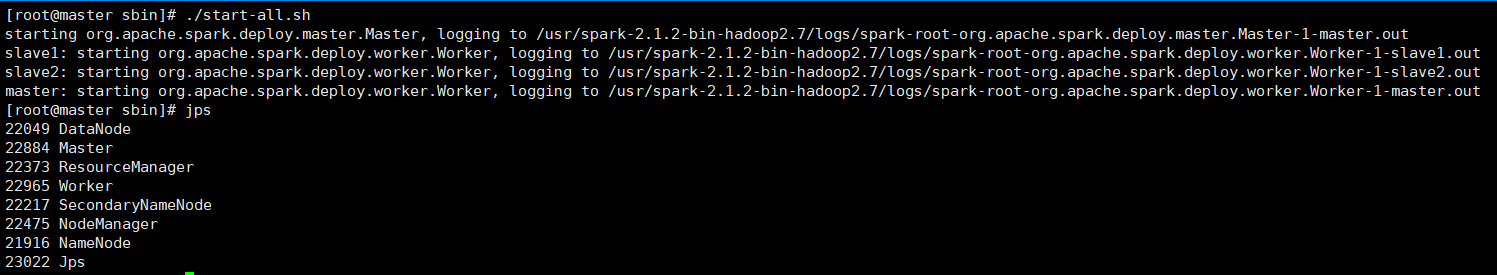
1. Spark安装和启动

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

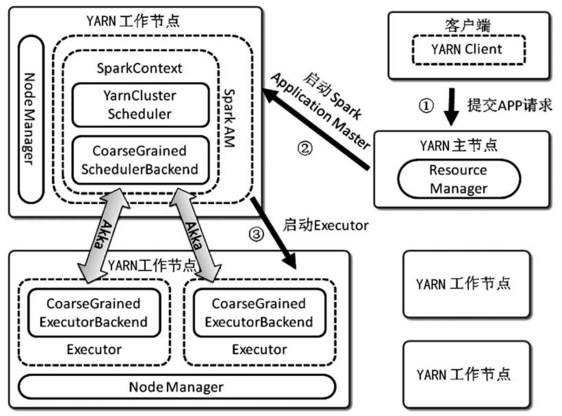


对于数据处理层，Spark相对于MapReduce的优势

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。



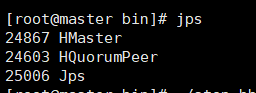
1. HBase的安装

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）



# Part4、数据分析

(尴尬的是很多代码文件找不到了)

就以统计行业需求为例，统计各行业的需求人数：

由于在MySQL中，用groupby进行分组，所以用SQL语句来解决

#encoding=utf8

import sys

reload(sys)

sys.setdefaultencoding( "utf-8" )

import pymysql

import numpy as np

import pandas as pd

def conmysql():

conn=pymysql.connect(

host='localhost',

port=3306,

user='root',

passwd='root',

database='zhaopin',

charset='utf8',

)

return conn

if \_\_name\_\_ == '\_\_main\_\_':

qeurysql='select distinct company\_industry,count(\*) from collect2 group by company\_industry order by count(\*) desc'

conn=conmysql()

cursor=conn.cursor()

data=pd.read\_sql(qeurysql,conn)

print data

其实也可以用python 的groupby方法

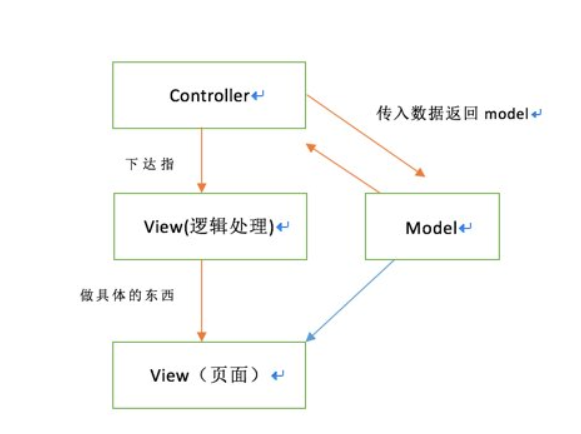
如: grouped=data['workcity'].groupby(data['company\_industry'])

Workcity 按company\_industry来分组

# Part5、应用开发部署

选取Apache+Django框架

Django框架是python的一个开源框架，简单分为以下几个主要部分



首先我们看一下url.py

分为了主url.py和应用url.py

主url.py

from django.conf.urls import url,include

from django.contrib import admin

admin.autodiscover()

urlpatterns=[url(r'^blog/',include('blog.urls')),

url(r'^admin/',include(admin.site.urls)),

url(r'/',include('blog.urls')),

url(r'^',include('blog.urls')),

]

Blog.url.py

# -\*- coding: utf-8 -\*-

from django.conf.urls import url,include

import blog.views

urlpatterns=[ url(r'^$', blog.views.index),

     url(r'^sensor/$',blog.views.sensor),

     url(r'^cloud\_index/$',blog.views.cloud),

url(r'^selfinformation/$',blog.views.selfinformation),

url(r'^jobresult/$',blog.views.jobresult),

url(r'^data\_analysis/$',blog.views.datasis),

url(r'^jobscompare/$',blog.views.jobscompare),

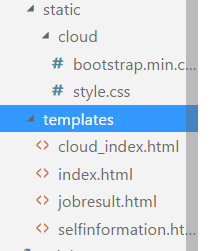
]

当来地址时，先在主url找，再去app.url找

Setting连接MySQL



在app中，创建template放置html文件，static放置js或css文件



处理视图层 view.py

# -\*- coding: utf-8 -\*-

from \_\_future\_\_ import unicode\_literals

from django.shortcuts import render

from blog.jobdeal import qeurydata

# Create your views here.

def selfinformation(requset):

return render(requset,'selfinformation.html')

def jobresult(requset):

industry=requset.POST.get('industry')

workcity=requset.POST.get('workcity')

education\_background=requset.POST.get('education\_background')

job\_form=requset.POST.get('job\_form')

company\_form=requset.POST.get('company\_form')

data=qeurydata(industry,workcity,education\_background,job\_form,company\_form)

context={'job\_name':data['job\_name'],'company\_name':data['company\_name'],'min\_salary':data['min\_salary'],'max\_salary':data['max\_salary'],'workcity':data['workcity'],'data\_sourceweb':data['data\_sourceweb']}

return render(requset,'jobresult.html',context)

结合上下文返回response

最后连接apache服务器，用wsgi.py

import os

from os.path import join,dirname,abspath

PROJECT\_DIR = dirname(dirname(abspath(\_\_file\_\_)))

import sys

sys.path.insert(0,PROJECT\_DIR)

from django.core.wsgi import get\_wsgi\_application

os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "liugangweb.settings")

application = get\_wsgi\_application()

最后再写前端文件，如cloud\_index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1">

<title>求职者智能分析系统</title>

{% load staticfiles %}

<link rel="stylesheet" href="{% static '/cloud/bootstrap.min.css' %}" >

<link rel="stylesheet" href="{% static '/cloud/style.css' %}">

</head>

<body>

<section class="features-area section-padding" id="features">

<div class="container">

<div class="row">

<section class="services-area section-padding" id="hangyequshi" >

<div class="container">

<div class="col-sm-12">

<div class="section-heading text-center wow fadeInUp">

<h3 style="color: rgb(0, 112, 192);">职位智能分析系统</h3>

<br >

<h5>全国云计算应用创新大赛</h5>

<hr />

</div>

</div>

</div>

</section>

<div class="col-md-2 col-sm-4 col-xs-12 " align='center'>

<div class="single-feature wow fadeInUp" data-wow-delay="0.1s">

<i class="pe-7s-diamond"></i>

<h4 ><a href="/blog/selfinformation/" target="\_blank" >智能评估推荐</a></h4>

<p>求职者输入个人信息后，通过基于内容推荐等深度学习算法，并由大数据与云计算技术作基础支持，为求职者智能推荐职位并进行薪酬评估</p>

</div>

</div>

<div class="col-md-2 col-sm-4 col-xs-12" align='center'>

<div class="single-feature wow fadeInUp" data-wow-delay="0.2s">

<i class="pe-7s-gym"></i>

<h4 ><a href="/blog/data\_analysis/" target="\_blank">职位需求分析</a></h4>

<p>通过大数据与云计算对信息进行分析处理，了解各行业需求、各行业薪酬分布、职位地域分布、各职位技能需求等</p>

</div>

</div>

<div class="col-md-2 col-sm-4 col-xs-12" align='center'>

<div class="single-feature wow fadeInUp" data-wow-delay="0.3s">

<i class="pe-7s-headphones"></i>

<h4 ><a href="/blog/jobscompare/" target="\_blank">职业对比选择</a></h4>

<p>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</p>

</div>

</div>

<div class="col-md-2 col-sm-4 col-xs-12" align='center'>

<div class="single-feature wow fadeInUp" data-wow-delay="0.4s">

<i class="pe-7s-settings"></i>

<h4 ><a href="http://article.zhaopin.com/" target="\_blank">求职指导</a></h4>

<p>分享一些求职好文章，为求职者求职提供更好的帮助</p>

</div>

</div>

</div>

</div>

</div>

</section>

</body>

</html>

效果图如下：

