|  |  |
| --- | --- |
| 成绩 |  |



**课程设计报告**



**题 目 Nginx日志分析及可视化展示**

**课 程 名 称 工程项目实践**

**院 部 名 称**  软件工程学院

**专 业** 软件工程（本）

**班 级** 15软件工程（Z）

**学 生 姓 名**  吴跟强

**学 号**  1512001066

**课程设计地点**  **科技楼2号楼**

**指 导 教 师**  陶玉婷

金陵科技学院教务处制

**课程设计报告书写要求**

课程设计报告原则上要求学生手写，要求书写工整。若因课程特点需打印的，要遵照以下字体、字号、间距等的具体要求。纸张一律采用A4的纸张。

**课程设计报告书写说明**

课程设计报告应包含以下七部分内容：1、摘要  2、目录  3、前言/引言  4.正文 5. 结论   6. 参考文献    7. 附录，每部分的书写要求参见具体条目要求。

**填写注意事项**

（1）准确说明，层次清晰。

（2）尽量采用专用术语来说明事物。

（3）外文、符号、公式要准确，应使用统一规定的名词和符号。

（4）应独立完成课程设计报告的书写，严禁抄袭、复印，一经发现，以零分论处。

**课程设计报告批改说明**

课程设计报告的批改要及时、认真、仔细，一律用红色笔批改。课程设计报告的批改成绩采用五级计分制或百分制，具体评分标准由各院部自行制定。

**课程设计报告装订要求**

报告批改完毕后，任课老师应将课程设计报告以自然班为单位、按学号升序排列，并附上一份该课程设计的教学大纲。

实验项目名称： **Nginx日志分析及可视化展示**

实验学时： 4周

同组学生姓名： 实验地点： **科技楼2号楼**

实验日期： 2018.07.09~08.05 实验成绩：

批改教师： 批改时间：

1. **实验目的**
2. 认识大数据，认识大数据技术在新时代对企业的重要性。
3. 掌握Linux的基础知识和应用。
4. 掌握hadoop集群环境的搭建。
5. 熟悉HDFS分布式文件系统的原理，掌握HDFS的API的开发。
6. 了解MapReduce原理，掌握MapReduce的开发技术。
7. 掌握flume传输数据的方法和任务启动命令。
8. 掌握spark处理数据的API。
9. 掌握Nginx反向代理的配置。
10. 掌握highCharts插件的使用。
11. **实验要求**
12. 独立完成下面项目：

采集1-3台机器的nginx的access.log/var/log/nginx/access.log）实时保存在HDFS中,使用spark对当天的日志进行汇总分析,在web界面中以图表的形式展示出来，需要体现如下2个表：

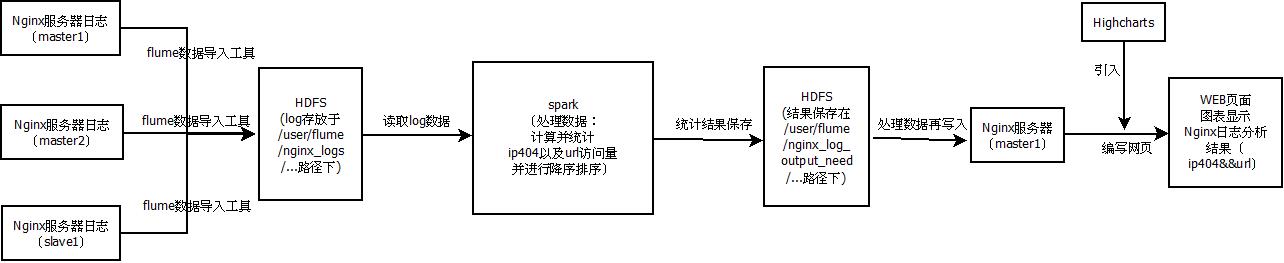
（1）哪个URL访问数量最大,按访问量从多到少排序展示出来

（2）哪些IP访问造成404错误最多，按从多到少排序展示出来

提高练习：

使用spark对所有的日志进行汇总分析，在web界面中展示出来

1. 自学Spark，完成对其的算法操作
2. 自学数据爬取技术
3. 自学Highcharts，完成图表显示工作
4. 自学如何优化算法
5. **实验环境、设备**
6. 系统为Linux系统。
7. 在VMware中搭建三台机器的集群：master1、master2、slave1。
8. 在master1和master2上配置flume数据导入工具、nginx服务器、spark。
9. **实验代码**
10. 项目架构图



图一 项目架构图

1. 使用技术



图二 技术罗列

1. 技术点说明



图三 技术点说明

1. 项目源代码
2. /etc/flume/conf/flume.conf

作用：修改exec数据源，配置flume代理

代码如下：

##配置Agent

a1.sources = r1

a1.sinks = k1

a1.channels = c1

# # 配置Source

a1.sources.r1.type = exec

a1.sources.r1.channels = c1

a1.sources.r1.deserializer.outputCharset = UTF-8

# # 配置需要监控的日志输出目录

a1.sources.r1.command = tail -F /var/log/nginx/access.log

# # 配置Sink

a1.sinks.k1.type = hdfs

a1.sinks.k1.channel = c1

a1.sinks.k1.hdfs.useLocalTimeStamp = true

a1.sinks.k1.hdfs.path = hdfs://master:9000/user/flume/nginx\_logs/%Y%m%d

#nginx日志存放在hdfs路径地址

a1.sinks.k1.hdfs.filePrefix = nginx\_log\_%Y%m%d

a1.sinks.k1.hdfs.fileSuffix = .log

a1.sinks.k1.hdfs.minBlockReplicas = 1

a1.sinks.k1.hdfs.fileType = DataStream

a1.sinks.k1.hdfs.writeFormat = Text

a1.sinks.k1.hdfs.rollSize = 1000000

a1.sinks.k1.hdfs.rollCount = 10000

a1.sinks.k1.hdfs.idleTimeout = 0

a1.sinks.k1.hdfs.rollInterval = 600

a1.sinks.k1.hdfs.round = true

a1.sinks.k1.hdfs.roundValue = 5

a1.sinks.k1.hdfs.roundUnit = minute

# # 配置Channel

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

# # 将三者连接

a1.sources.r1.channel = c1

1. flume\_start.sh

作用：启动flume代理，收集nginx服务器的日志到hdfs中，数据采集

代码如下：

[root@master ~]# cd /usr/share/nginx/

[root@master nginx]# cat flume\_start.sh

#!/bin/bash

#start flume 将日志记录传入hdfs

flume-ng agent --conf /etc/flume/conf/ --conf-file /etc/flume/conf/flume.conf --name a1

[root@master nginx]#

（3）test\_log\_records\_add.sh

作用：启动flume代理时运行，访问nginx服务器，增加日志记录量，

代码如下：

step=1 #间隔的秒数，不能大于60

host\_list=("localhost" "192.168.75.213" "master" "127.0.0.1")

while [ 1 ]

do

num=$(((RANDOM%10)+1))

seq=$(((RANDOM%4)))

url="http://"${host\_list[$seq]}"/"$num".html";

echo " `date +%Y-%m-%d\ %H:%M:%S` get $url"

#curl http://192.168.75.213/1.html #调用链接

curl -s $url > /dev/null

sleep $step

done

（4）nginx\_log.py

作用：处理log数据的脚本，将spark处理的数据上传hdfs，为脚本计划运行做准备

代码如下：

[root@master nginx]# cat nginx\_log.py

#!/usr/bin/python

#coding=utf-8

from pyspark import SparkContext

import os

os.environ['PYTHONPATH']='python2'

import sys

reload(sys)

sys.setdefaultencoding('utf-8')

sc=SparkContext("yarn")

#将url访问量降序排序，存入hdfs对应路径中:

rdd\_url = sc.textFile("hdfs://master:9000/user/flume/nginx\_logs/20180803/\*").map(lambda s:s.split(" ")).map(lambda w:[str(w[6])]).flatMap(lambda w:w).map(lambda w:[w,1]).reduceByKey(lambda x,y:x+y).sortBy(lambda x:-x[1]).map(lambda x:list(x)).coalesce(1).saveAsTextFile("/user/flume/nginx\_log\_output\_need/url/20180803")

#ip\_404错误降序排序，存入hdfs对应路径中:

rdd\_ip404 = sc.textFile("hdfs://master:9000/user/flume/nginx\_logs/20180803/\*").filter(lambda line:"404" in line).map(lambda s:s.split(" ")).map(lambda w:[str(w[0])]).flatMap(lambda w:w).map(lambda w:[w,1]).reduceByKey(lambda x,y:x+y).sortBy(lambda x:-x[1]).map(lambda x:list(x)).coalesce(1).saveAsTextFile("/user/flume/nginx\_log\_output\_need/ip404/20180803")

1. spark\_run\_nginx\_log.sh

作用：启动数据分析，执行即可实现日志的处理并在local生成file，供web端调用

代码如下：

[root@master nginx]# cat spark\_run\_nginx\_log.sh

#!/bin/bash

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.151-1.b12.el7\_4.x86\_64

echo "`date` job start..."

echo ' --- rm hdfs output\_nginx --- '

echo ' --- rm url information ---'

hdfs dfs -rm -R /user/flume/nginx\_log\_output\_need/url/20180803

echo $?

echo ' --- rm url information ok---'

echo ' --- rm ip404 information ---'

hdfs dfs -rm -R /user/flume/nginx\_log\_output\_need/ip404/20180803

echo $?

echo ' --- rm ip404 information ok---'

echo ' --- rm hdfs output\_nginx Done --- '

echo ' --- spark-submit Done --- '

echo ' --- copy file from hdfs to local file : url.txt ---'

rm -rf /usr/share/nginx/html/nginx\_log\_need/url.txt

/usr/bin/hdfs dfs -getmerge /user/flume/nginx\_log\_output\_need/url/20180803 /usr/share/nginx/html/nginx\_log\_need/url.txt

echo $?

echo ' --- copy file from hdfs to local file : url.txt Done ---'

echo ' --- copy file from hdfs to local file : ip404.txt ---'

rm -rf /usr/share/nginx/html/nginx\_log\_need/ip404.txt

/usr/bin/hdfs dfs -getmerge /user/flume/nginx\_log\_output\_need/ip404/20180803 /usr/share/nginx/html/nginx\_log\_need/ip404.txt

echo $?

echo ' --- copy file from hdfs to local file : ip404.txt Done ---'

echo " All Done !!!"

echo " `date` job end "

exit 0

（6）crontab -e

作用：计划任务，设置脚本自动运行，每10分钟运行一次

代码如下：

\*/10 \* \* \* \* /usr/share/nginx/spark\_run\_nginx\_log.sh 2>&1 >> /usr/share/nginx/crontab\_spark\_run\_nginx.log

（7）ip404\_Access.html，ajax算法部分

作用：ajax获取spark处理后的数据，拼接成需要的json格式

代码如下：

$.ajax({

type: "get",

async: false,

url: "/nginx\_log\_need/ip404.txt",

success: function(data) {

//从本地文件中读取数据，以换行符分割

var data\_source=data.split("\n");//数据源

var data\_handle="";//数据处理

var sum=0;//求和变量，为算比例做准备

//控制台打印调试

console.log("数据源："+data\_source)//控制台查看数据源

for(i=0;i<data\_source.length-2;i++){//数据源中数据处理后放入data\_handle

data\_handle=data\_handle+data\_source[i].substr(0,data\_source[i].length)+",";} data\_handle=data\_handle+data\_source[i].substr(0,data\_source[i].length);

console.log("数据源处理结果:"+data\_handle);//控制台打印处理好的数据

rst=eval('['+data\_handle+']');//eval处理表达式

for(i=0;i<rst.length;i++){

sum=sum+rst[i][1];//value值求和

}

for(i=0;i<rst.length;i++){

rst[i][1]=(parseFloat(rst[i][1])/parseFloat(sum));

//可用toFixed(2)控制小数精度

}}});

（8）url\_Access.html，ajax算法部分

作用：ajax获取spark处理后的数据，拼接成需要的json格式

代码如下：

$.ajax({//获取url

type: "get",

async: false,

url: "/nginx\_log\_need/url.txt",

success: function(data) {

//数据源通过]符号进行分割,命名为data\_source

var data\_source = data.split(']');

console.log("数据源："+data\_source);

var url\_value = "";//url值

for(i = 0; i < data\_source.length - 1; i++) {

if(i == 0) {

url\_value = url\_value + (data\_source[i].split(',')[0]).substr(1) + ",";

} else {

url\_value = url\_value + (data\_source[i].split(',')[0]).substr(2);

if(i < data\_source.length - 2) {

url\_value = url\_value + ",";

}

}

}

console.log("url："+url\_value);

rst = eval('[' + url\_value + ']');

console.log(rst);

} });

$.ajax({//获取url访问量

type: "get",

async: false,

url: "/nginx\_log\_need/url.txt",

success: function(data) {

//数据源

var data\_source = data.split(']');

var data\_handle = "";//数据处理

console.log(data\_source);

for(i = 0; i < data\_source.length - 1; i++) {

data\_handle = data\_handle.concat(data\_source[i].split(',')[1]);

if(i < data\_source.length - 2) {

data\_handle = data\_handle + ",";

}

}

console.log("处理："+data\_handle);

rst = eval('[' + data\_handle + ']');

}

});

（9）frame\_log\_analysis.html（当天）

作用：将ip404和url生成的图表，显示在一个页面中，直观

代码如下：

[root@master html]# cat frame\_log\_analysis.html

<frameset id="frame" name="frame" border="0" framespacing="0" rows="\*" frameborder="no" cols="\*,\*">

---左边显示url\_Access

<frame name="left\_html" src="analysis\_log/url\_Access.html" frameborder="0">

---右边显示ip404\_Access

<frame name="right\_html" src="analysis\_log/ip404\_Access.html" frameborder="0" >

</frameset>

1. frame\_all\_log\_analysis.html（所有天）

代码如下：

[root@master html]# cat frame\_all\_log\_analysis.html

<frameset id="frame" name="frame" border="0" framespacing="0" rows="\*" frameborder="no" cols="\*,\*">

---左边显示url\_all

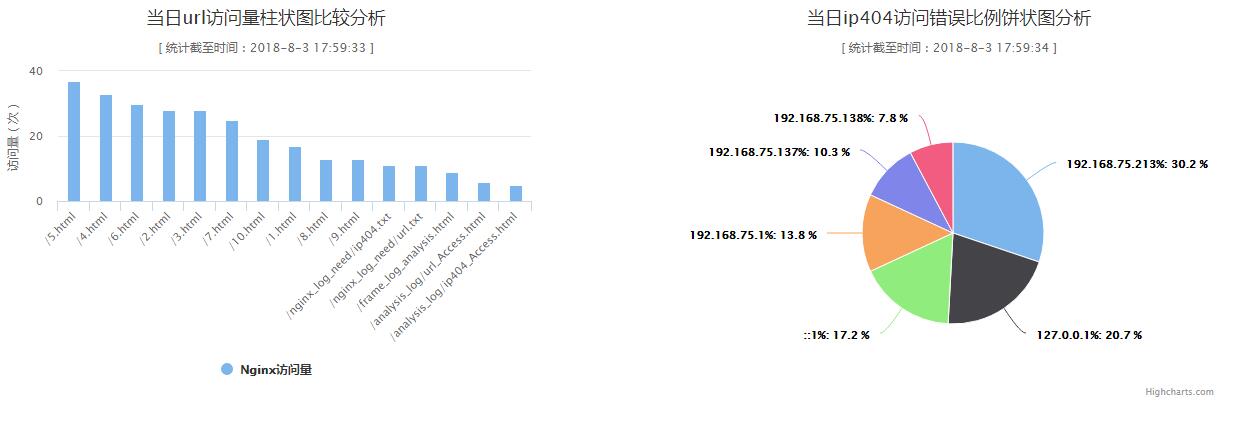
<frame name="left\_html" src="analysis\_log/url\_all.html" frameborder="0">

---右边显示ip404\_all

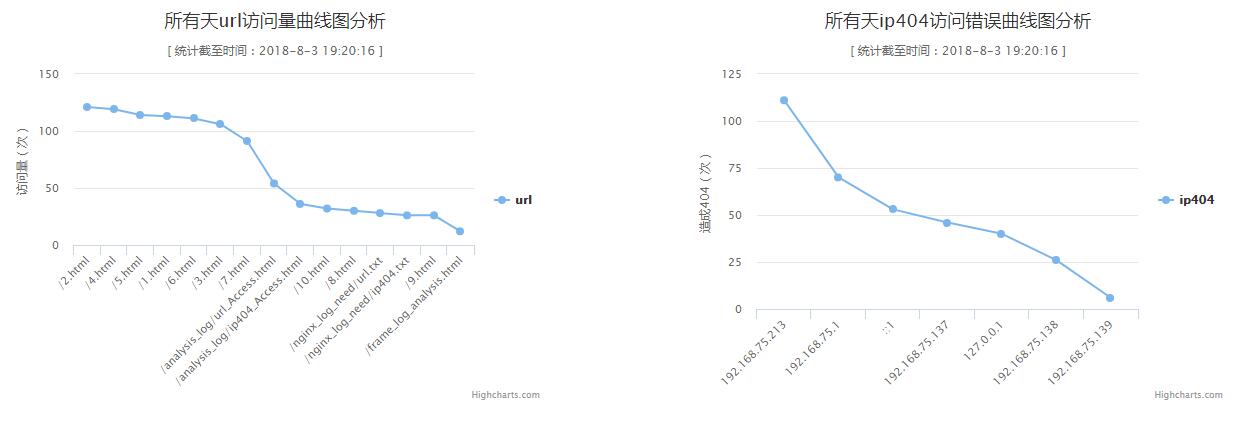
<frame name="right\_html" src="analysis\_log/ip404\_all.html" frameborder="0" >

</frameset>

1. 界面截图



图四 url和404访问图表分析（当天）



图五 url和404访问图表分析（所有天）

1. **实验心得**

这次大数据实训让我收获了很多，从最开始老师带着我们学习linux，到中期的大数据HDFS、yarn、mapreduce、spark以及python等知识的学习和使用，以及最后的实训项目：处理nginx日志并在web页面上图表显示处理过的日志信息，过程中离不开老师的帮助与指导，在这里我成长了很多，很高兴，在即将离开大学校园的我，可以进一步学习大数据的相关知识，真的很开心！

这次实训项目是处理nginx日志信息，提取出需要的ip404和url访问量信息，并图表显示出来，首先我对项目任务进行了分解，开始的时候，针对本项目，搭建了集群并配置了Hdfs文件系统以及yarn资源管理系统等系统，确保环境变量都正常，继而安装了flume数据导入工具，通过flume，实时获取nginx服务器的数据，并将数据传入分布式文件系统HDFS，通过pyspark将数据获取并处理有用的数据并保存至HDFS相应文件中，使用的时候将数据传回本地目录中使用，最后就是编写ajax代码，实现web页面和本地文件的连接，使得数据以图表的形式动态显示在web页面，到这里项目几乎完成了，但是手工操作还是费时间，于是优化项目，采取了计划任务运行，通过crontab定时任务，使spark定时运行脚本化，后台自动执行，大大提高了应用效率，也让自己感到知识需要拓展的地方真的很多，还需要进一步学习，才能更好的做事！在这里再一次感恩指导老师的帮助，谢谢啦！