In [1]:

```
##创建环境

import findspark
findspark.init()
#################################################
from pyspark.sql import SparkSession
from pyspark.sql.types import *

spark = SparkSession.builder \
        .master("local[*]") \
        .appName("NoteFrame") \
        .getOrCreate()

sc = spark.sparkContext
```

数据说明

http.log： 用户访问网站所产生的日志。日志格式为：时间戳、IP地址、访问网址、访问数据、浏览器信息等，样例如下：

In [2]:

```
!head -3 data/http.log
```

```
20090121000132095572000|125.213.100.123|show.51.com|/shoplist.php?phpfile=shoplist2.php&style=1&sex=
137|Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Mozilla/4.0(Compatible Mozilla/4.0(Compa
tible-EmbeddedWB 14.59 http://bsalsa.com/ (http://bsalsa.com/) EmbeddedWB- 14.59  from: http://bsals
a.com/ (http://bsalsa.com/) )|http://show.51.com/main.php|
20090121000132124542000|117.101.215.133|www.jiayuan.com|/19245971|Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.1; SV1; TencentTraveler 4.0)|http://photo.jiayuan.com/index.php?uidhash=d1c3b69e9b8355a
5204474c749fb76ef|__tkist=0; myloc=50%7C5008; myage=2009; PROFILE=14469674%3A%E8%8B%A6%E6%B6%A9%E5%9
2%96%E5%95%A1%3Am%3Aphotos2.love21cn.com%2F45%2F1b%2F388111afac8195cc5d91ea286cdd%3A1%3A%3Ahttp%3A%2
F%2Fimages.love21cn.com%2Fw4%2Fglobal%2Fi%2Fhykj_m.jpg; last_login_time=1232454068; SESSION_HASH=817
6b100a84c9a095315f916d7fcbcf10021e3af; RAW_HASH=008a1bc48ff9ebafa3d5b4815edd04e9e7978050; COMMON_HAS
H=45388111afac8195cc5d91ea286cdd1b; pop_1232093956=1232468896968; pop_time=1232466715734; pop_123224
5908=1232469069390; pop_1219903726=1232477601937; LOVESESSID=98b54794575bf547ea4b55e07efa2e9e; main_
search:14469674=%7C%7C%7C00; registeruid=14469674; REG_URL_COOKIE=http%3A%2F%2Fphoto.jiayuan.com%2Fs
howphoto.php%3Fuid_hash%3D0319bc5e33ba35755c30a9d88aaf46dc%26total%3D6%26p%3D5; click_count=0%2C3363
619
20090121000132406516000|117.101.222.68|gg.xiaonei.com|/view.jsp?p=389|Mozilla/4.0 (compatible; MSIE
7.0; Windows NT 5.1; CIBA)|http://home.xiaonei.com/Home.do?id=229670724|_r01_=1; __utma=204579609.31
669176.1231940225.1232462740.1232467011.145; __utmz=204579609.1231940225.1.1.utmccn=(direct)
```

ip.dat： ip段数据，记录着一些ip段范围对应的位置，总量大概在11万条，数据量也算很小的，样例如下

In [3]:

```
!head -3 data/ip.dat
```

```
1.0.1.0|1.0.3.255|16777472|16778239|亚洲|中国|福建|福州||电信|350100|China|CN|119.306239|26.075302
1.0.8.0|1.0.15.255|16779264|16781311|亚洲|中国|广东|广州||电信|440100|China|CN|113.280637|23.125178
1.0.32.0|1.0.63.255|16785408|16793599|亚洲|中国|广东|广州||电信|440100|China|CN|113.280637|23.125178
```

任务描述

将 http.log 文件中的 ip 转换为地址。如将 122.228.96.111 转为温州，并统计各城市的总访问量

定义一些辅助函数

In [4]:

```
#ip转化为int类型
def ip2int(s):
    new_str = s.split(".")
    ipNum = 0
    for i in range(len(new_str)):
        ipNum = int(new_str[i]) | ipNum << 8
    return ipNum
```

In [5]:

```
print(ip2int('192.168.220.21'))
```

```
3232291861
```

In [6]:

```
#二分法匹配ip规则
def binary(_list, ip):
    left = 0   # 列表的起始索引
    right = len(_list) - 1   # 列表的结束索引
    mid = int((left + right)/2)  # 采用此方法，通过四舍五入刚好可以定位到列表的中间位置
    while left <= right:
        mid = int((right + left)/2)
        if _list[mid][0] <= ip  and _list[mid][1] >= ip:
            return mid
        elif _list[mid][0] > ip:
            right = mid -1
        else:
            left = mid+1
    return -1
```

ip转换并统计各城市的总访问量

In [7]:

```
#提取日志中的ip地址
rddlog = sc.textFile("data/http.log").map(lambda x:x.split("|")[1])
rddlog.collect()
```

```
 '123.197.64.247',
 '222.55.57.176',
 '123.197.66.93',
 '115.120.12.157',
 '115.120.7.240',
 '117.101.219.241',
 '123.197.49.171',
 '117.101.213.104',
 '115.120.10.205',
 '117.101.218.147',
 '115.120.17.80',
 '117.101.220.175',
 '123.197.66.12',
 '125.213.100.236',
 '123.197.66.208',
 '115.120.19.122',
 '115.120.2.192',
 '117.101.204.182',
 '117.75.230.192',
 '123.197.43.186'
```

In [8]:

```
#将ip范围与对应城市整合
rddip = sc.textFile("data/ip.dat").map(
    lambda x:(x.split("|")[0],
    x.split("|")[1],
    x.split("|")[7]))

rddip.collect()
```

```
 ('1.15.0.0', '1.15.119.255', '北京'),
 ('1.15.120.0', '1.15.127.255', '天津'),
 ('1.15.128.0', '1.15.159.255', '北京'),
 ('1.15.160.0', '1.15.167.255', '天津'),
 ('1.15.168.0', '1.15.207.255', '北京'),
 ('1.15.208.0', '1.15.223.255', '大连'),
 ('1.15.224.0', '1.15.255.255', '天津'),
 ('1.24.0.0', '1.24.7.255', '呼和浩特'),
 ('1.24.8.0', '1.24.15.255', '锡林郭勒盟'),
 ('1.24.16.0', '1.24.31.255', '包头'),
 ('1.24.32.0', '1.24.39.255', '乌兰察布'),
 ('1.24.40.0', '1.24.63.255', '锡林郭勒盟'),
 ('1.24.64.0', '1.24.79.255', '阿拉善盟'),
 ('1.24.80.0', '1.24.95.255', '乌兰察布'),
 ('1.24.96.0', '1.24.127.255', '巴彦淖尔'),
 ('1.24.128.0', '1.24.135.255', '包头'),
 ('1.24.136.0', '1.24.147.255', '鄂尔多斯'),
 ('1.24.148.0', '1.24.159.255', '乌海'),
 ('1.24.160.0', '1.24.183.255', '包头'),
 ('1.24.184.0', '1.24.187.255', '乌海'),
```

In [9]:

```
#定义广播变量
brIP = sc.broadcast(rddip.map(lambda x: (ip2int(x[0]), ip2int(x[1]), x[2])).collect())
```

In [10]:

```
brIP
```

Out[10]:

```
<pyspark.broadcast.Broadcast at 0xfffcb407be10>
```

In [11]:

```
def getcity(x):
    index = binary(brIP.value, ip2int(x))
    if index != -1:
        return brIP.value[index][2]
    else:
        return 'NULL'
```

In [13]:

```
#关联后的结果rdd
from operator import add
rddlog.map(lambda x: getcity(x)).map(lambda x:(x,1)).reduceByKey(add).map(lambda x: {"城市":x[0],"访问量":x[1]}).
```

In [20]:

```
!head -n 5 data/work/new_output_ips/part-00000
```

```
{'城市': '重庆', '访问量': 868}
{'城市': '北京', '访问量': 1535}
{'城市': '西安', '访问量': 1824}
{'城市': '石家庄', '访问量': 383}
{'城市': '昆明', '访问量': 126}
```

In [ ]:

```

```