In [1]:

```
# The following program is based on Apache Spark using Python based on the mooc ht
tps://courses.edx.org/courses/BerkeleyX/CS100.1x/1T2015/info


# program created by : dhongadeaakash@gmail.com

#this script performs analysis on log files in the CLF format

# example of a log file is 127.0.0.1 - - [01/Aug/1995:00:00:01 -0400] "GET /image
s/launch-logo.gif HTTP/1.0" 200 1839
#class datetime.datetime(year, month, day[, hour[, minute[, second[, microsecond[,
tzinfo]]]]])
import re
import datetime
import os.path
import matplotlib.pyplot as plt
from pyspark.sql import Row
from operator import add
month_map = {'Jan': 1, 'Feb': 2, 'Mar':3, 'Apr':4, 'May':5, 'Jun':6, 'Jul':7,'Au
g':8,  'Sep': 9, 'Oct':10, 'Nov': 11, 'Dec': 12}
# a regular expression to extract fields from the logline
APACHE_ACCESS_LOG_PATTERN = '^(\S+) (\S+) (\S+) \[([\w:/]+\s[+\-]\d{4})\] "(\S+)
(\S+)\s*(\S*)\s*" (\d{3}) (\S+)'

#returns a datetime object
def apache_parse_time(s):
    return datetime.datetime(int(s[7:11]),
                             int(month_map[s[3:6]]),
                             int(s[0:2]),
                             int(s[12:14]),
                             int(s[15:17]),
                             int(s[18:20]))

#a function to parse a log line of the CLF format
def parseApacheLogLine(logline):
    match=re.search(APACHE_ACCESS_LOG_PATTERN,logline)
    if match is None:
        return (logline,1)
    size_field=match.group(9)
    if size_field=='-':
        size=long(0)
    else:
        size=long(match.group(9))
    # the following fields are extracted from the log file and stored in respectiv
e Columns
    return (Row(
            host                   =match.group(1),
            client_identification  =match.group(2),
            user_id                =match.group(3),
            date_time              =apache_parse_time(match.group(4)),
            method                 =match.group(5),
            endpoint               =match.group(6),
            protocol               =match.group(7),
            response_code          =int(match.group(8)),
            content_size           =size
        ),1)
def parseLogs():
```

```
    parse_Logs=(sc.textFile(logFile).map(parseApacheLogLine).cache())
    access_Logs=parse_Logs.filter(lambda s:s[1]==1).map(lambda s:s[0]).cache()
    failed_Logs=parse_Logs.filter(lambda s:s[1]==0).map(lambda s:s[0]).cache()
    print "%d Total lines parsed %d Successfull lines parsed %d failed lines parse
d" %(parse_Logs.count(),access_Logs.count(),failed_Logs.count())
    return parse_Logs,access_Logs,failed_Logs
# set file name and path
baseDir = os.path.join('data')
inputPath = os.path.join('cs100', 'lab2', 'apache.access.log.PROJECT')
logFile=os.path.join(baseDir,inputPath)
#parse_logs
parse_Logs,access_Logs,failed_Logs=parseLogs();
```

```
1043177 Total lines parsed 1043177 Successfull lines parsed 0 failed
lines parsed
```

In [2]:

```
# 1st Analysis  : Content Size Analysis
# show Minimum ,Maximum and average sizes returned by the web server
content_sizes=access_Logs.map(lambda log:log.content_size).cache()
print 'Content Size Avgerage : %i \n Minimum : %i \n Maximum: %s' % (
    content_sizes.reduce(lambda a, b : a + b) / content_sizes.count(),
    content_sizes.min(),
    content_sizes.max())
```

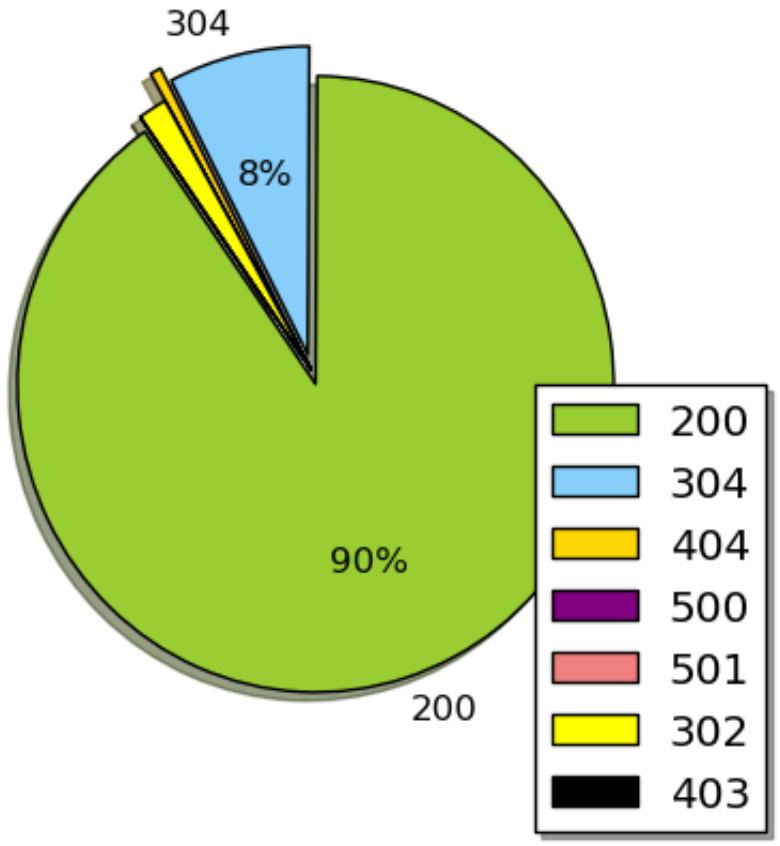
```
Content Size Avgerage : 17531
 Minimum : 0
 Maximum: 3421948
```

In [9]:

```
# 2nd Analysis Response Code Analysis
# Display the result in the form of a  chart
import matplotlib.pyplot as plt
responseRDD=access_Logs.map(lambda log:(log.response_code,1)).reduceByKey(add)
print "Found %d response Codes\n Response Codes :\n %s" %(responseRDD.count(),resp
onseRDD.take(15))
codes=responseRDD.map(lambda (x,y):x).collect()
count = access_Logs.count()
fractions=responseRDD.map(lambda (x, y): (float(y) / count)).collect()
def pie_pct_format(value):
    return '' if value < 7 else '%.0f%%' % value

fig = plt.figure(figsize=(4.5, 4.5), facecolor='white', edgecolor='white')
colors = ['yellowgreen', 'lightskyblue', 'gold', 'purple', 'lightcoral', 'yellow',
'black']
explode = (0.05, 0.05, 0.1, 0, 0, 0, 0)
patches, texts, autotexts = plt.pie(fractions, labels=codes, colors=colors,
                                    explode=explode, autopct=pie_pct_format,
                                    shadow=True,  startangle=125)
for text, autotext in zip(texts, autotexts):
    if autotext.get_text() == '':
        text.set_text('')  # If the slice is small to fit, don't show a text label
a=plt.legend(codes, loc=(0.80, -0.1), shadow=True)
```

```
Found 7 response Codes
 Response Codes :
 [(200, 940847), (304, 79824), (404, 6185), (500, 2), (501, 17),
(302, 16244), (403, 58)]
```



In [11]:

```
#3rd Analysis Frequency Host
HostCount=access_Logs.map(lambda log:(log.host,1)).reduceByKey(add)
top15Host=HostCount.takeOrdered(15,lambda (x,y):-y)
print top15Host
```

```
[(u'edams.ksc.nasa.gov', 4034), (u'piweba5y.prodigy.com', 3237), (u'p
iweba4y.prodigy.com', 3043), (u'piweba3y.prodigy.com', 2830), (u'www-
d1.proxy.aol.com', 2715), (u'www-b3.proxy.aol.com', 2518), (u'news.t
i.com', 2507), (u'www-b2.proxy.aol.com', 2481), (u'163.206.89.4', 247
8), (u'www-c2.proxy.aol.com', 2438), (u'www-c3.proxy.aol.com', 2400),
(u'www-d2.proxy.aol.com', 2371), (u'www-d4.proxy.aol.com', 2356),
(u'www-b5.proxy.aol.com', 2354), (u'www-b4.proxy.aol.com', 2297)]
```

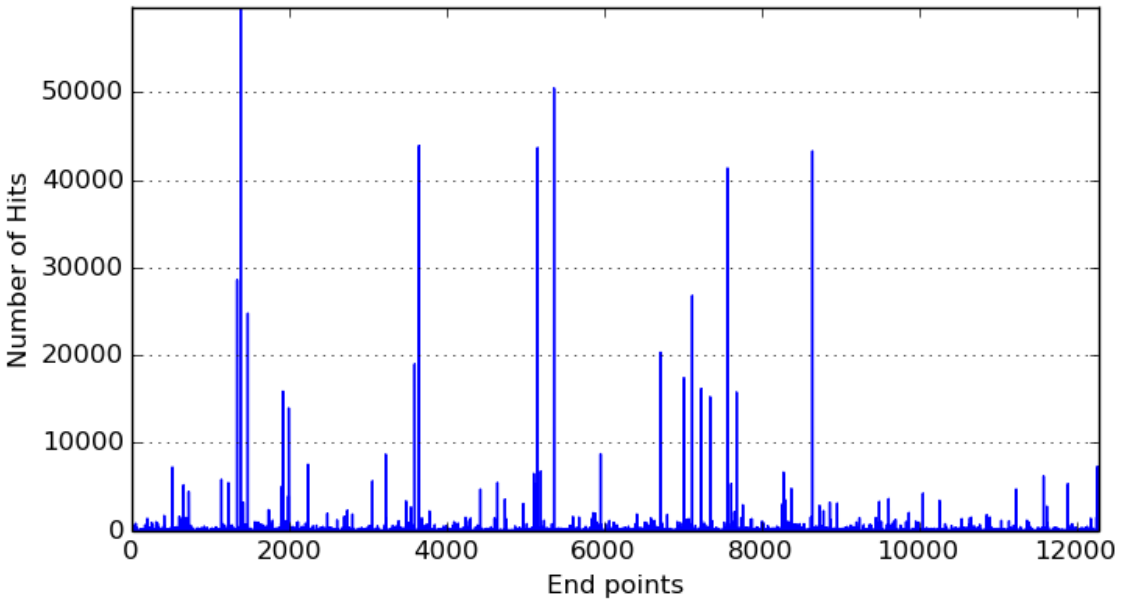
In [17]:

```
#4th is Number of Hits on EndPoints with top 10 endpoints
endpoints=access_Logs.map(lambda log:(log.endpoint,1)).reduceByKey(add)
endpointcounts=endpoints.map(lambda (x,y):y).collect()
#top 10 endpoints are
top10endpoints=endpoints.takeOrdered(10,lambda (x,y):-y)
print"\n".join(map(lambda (w,c): '{0} :  {1} '.format(w,c),top10endpoints))
fig = plt.figure(figsize=(8,4.2), facecolor='white', edgecolor='white')
plt.axis([0,len(endpointcounts),0,max(endpointcounts)])
plt.grid(b=True,which='major',axis='y')
plt.xlabel('End points')
plt.ylabel('Number of Hits')
plt.plot(endpointcounts)
```

```
/images/NASA-logosmall.gif :  59737
/images/KSC-logosmall.gif :  50452
/images/MOSAIC-logosmall.gif :  43890
/images/USA-logosmall.gif :  43664
/images/WORLD-logosmall.gif :  43277
/images/ksclogo-medium.gif :  41336
/ksc.html :  28582
/history/apollo/images/apollo-logo1.gif :  26778
/images/launch-logo.gif :  24755
/ :  20292
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0xb0834a0c>]
```



In [21]:

```
#5th Analysis  Error Endpoints.
not200endpointsSum=access_Logs.filter(lambda log:log.response_code!=200).map(lambda log:(log.endpoint,1)).reduceByKey(add)
#to print all values
#  print"\n".join(map(lambda (w,c): '{0} : {1}'.format(w,c),not200endpointsSum.collect())
#to print top 10 values
print"\n".join(map(lambda (w,c): '{0} : {1}'.format(w,c),not200endpointsSum.takeOrdered(10,lambda (x,y):-y)))
```

```
/images/NASA-logosmall.gif : 8761
/images/KSC-logosmall.gif : 7236
/images/MOSAIC-logosmall.gif : 5197
/images/USA-logosmall.gif : 5157
/images/WORLD-logosmall.gif : 5020
/images/ksclogo-medium.gif : 4728
/history/apollo/images/apollo-logo1.gif : 2907
/images/launch-logo.gif : 2811
/ : 2199
/images/ksclogosmall.gif : 1622
```

In [22]:

```
#6th Analysis - Number of Unique Hosts
count=access_Logs.map(lambda log:(log.host,1)).reduceByKey(add).count()
print "The number of Unique hosts are  %d" %(count)
```

```
The number of Unique hosts are  54507
```

In [53]:

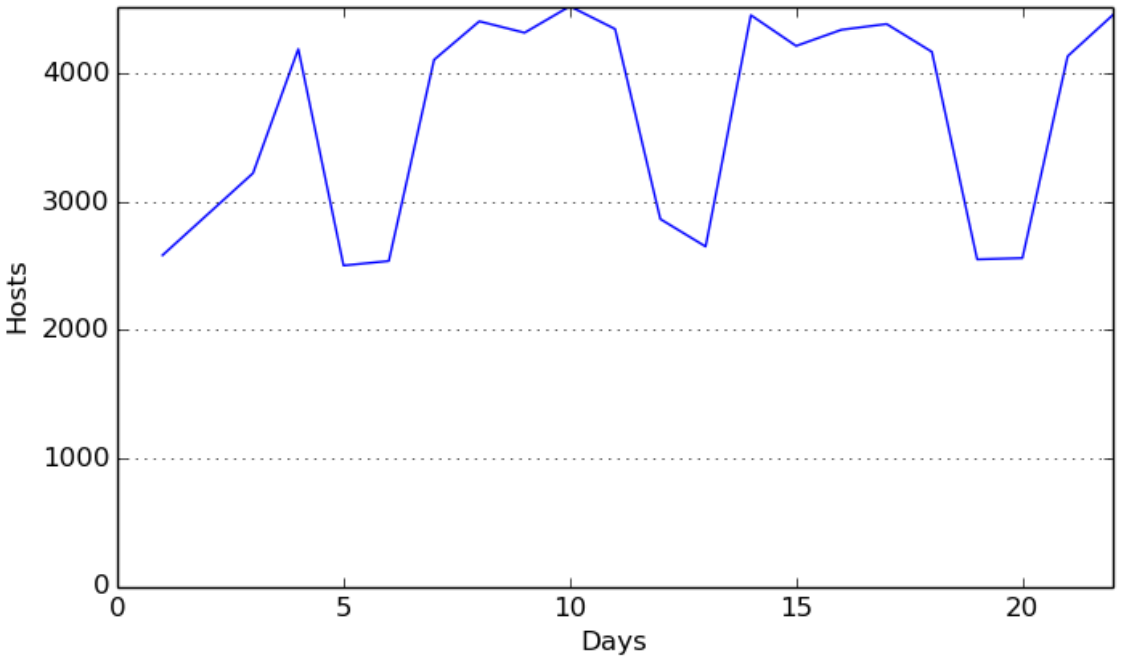
```
#7th Number of Unique Hosts date wise are
NumberOfUniqueHosts=access_Logs.map(lambda log:(datetime.date(log.date_time.year,l
og.date_time.month,log.date_time.day),log.host)).groupByKey().map(lambda s: (s[0],
len(list(set(s[1]))))).sortByKey().cache()
print "\n".join(map(lambda (x,y):' {0}/{1}/{2} : {3}'.format(x.day,x.month,x.year,
y),NumberOfUniqueHosts.take(15)))

daysWithHosts = NumberOfUniqueHosts.map(lambda (x,y):x.day).collect()
hosts =NumberOfUniqueHosts.map(lambda (x,y):y).collect()
fig=plt.figure(figsize=(8,4.5) , facecolor='white' , edgecolor='white')
plt.axis([0,max(daysWithHosts),0,max(hosts)])
plt.grid(b=True,which='major',axis='y')
plt.xlabel('Days')
plt.ylabel('Hosts')
plt.plot(daysWithHosts,hosts)
```

```
1/8/1995 : 2582
3/8/1995 : 3222
4/8/1995 : 4190
5/8/1995 : 2502
6/8/1995 : 2537
7/8/1995 : 4106
8/8/1995 : 4406
9/8/1995 : 4317
10/8/1995 : 4523
11/8/1995 : 4346
12/8/1995 : 2864
13/8/1995 : 2650
14/8/1995 : 4454
15/8/1995 : 4214
16/8/1995 : 4340
```

Out[53]:

```
[<matplotlib.lines.Line2D at 0xb08c366c>]
```



In [58]:

```
# 8th Analysis - Total  Number of 404 Response Codes along with top 20 404 Respons
e Code EndPoints
badRecords=access_Logs.filter(lambda log:log.response_code==404).cache()
print "The Total Number of 404 Response Code Records are %d" %(badRecords.count())

badRecordsEndpoints=badRecords.map(lambda log:(log.endpoint,1)).reduceByKey(add)
print "The top 20 404 Response Code Endpoints are\n"
print "\n".join(map(lambda (w,c):'{0}'.format(w),badRecordsEndpoints.takeOrdered(2
0,lambda (x,y):-y)))
```

```
The Total Number of 404 Response Code Records are 6185
The top 20 404 Response Code Endpoints are

/pub/winvn/readme.txt
/pub/winvn/release.txt
/shuttle/missions/STS-69/mission-STS-69.html
/images/nasa-logo.gif
/elv/DELTA/uncons.htm
/shuttle/missions/sts-68/ksc-upclose.gif
/history/apollo/sa-1/sa-1-patch-small.gif
/images/crawlerway-logo.gif
/://spacelink.msfc.nasa.gov
/history/apollo/pad-abort-test-1/pad-abort-test-1-patch-small.gif
/history/apollo/a-001/a-001-patch-small.gif
/images/Nasa-logo.gif
/shuttle/resources/orbiters/atlantis.gif
/history/apollo/images/little-joe.jpg
/images/lf-logo.gif
/shuttle/resources/orbiters/discovery.gif
/shuttle/resources/orbiters/challenger.gif
/robots.txt
/elv/new01.gif>
/history/apollo/pad-abort-test-2/pad-abort-test-2-patch-small.gif
```

In [60]:

```
#9th Analysis - Listing 404 Response Codes Per Day

ResponseCodesPerDay=badRecords.map(lambda log:(datetime.date(log.date_time.year,lo
g.date_time.month,log.date_time.day),1)).reduceByKey(add).sortByKey().cache()
print "\n".join(map(lambda (x,y):' {0}/{1}/{2} : {3}'.format(x.day,x.month,x.year,
y),ResponseCodesPerDay.take(15)))

days = ResponseCodesPerDay.map(lambda (x,y):x.day).collect()
Errors=ResponseCodesPerDay.map(lambda (x,y):y).collect()
fig=plt.figure(figsize=(8,4.5) , facecolor='white' , edgecolor='white')
plt.axis([0,max(days),0,max(Errors)])
plt.grid(b=True,which='major',axis='y')
plt.xlabel('Days')
plt.ylabel('404 Errors')
plt.plot(days,Errors)
```

```
1/8/1995 : 243
3/8/1995 : 303
4/8/1995 : 346
5/8/1995 : 234
6/8/1995 : 372
7/8/1995 : 532
8/8/1995 : 381
9/8/1995 : 279
10/8/1995 : 314
11/8/1995 : 263
12/8/1995 : 195
13/8/1995 : 216
14/8/1995 : 287
15/8/1995 : 326
16/8/1995 : 258
```

Out[60]:

```
[<matplotlib.lines.Line2D at 0xb0862b2c>]
```