In [1]:

```
import pandas as pd
import numpy as np
from scipy import stats
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from tqdm.notebook import tqdm
import re
pd.set_option('max_columns', None)
pd.options.display.max_colwidth = 100
from pandas.api.types import CategoricalDtype
```

In [2]:

```
df = pd.read_csv('data.csv', index_col=0)
df.shape
```

Out[2]:

```
(6643221, 4)
```

First, tally the size distribution of all requested files. Then group the requests by file size and count the total number of requests and traffic for each group. By performing this analysis, we can gain insights into which types of files and how much resources are being devoted to serving these requests.

1 Number of files in each size range

In [3]:

```
df1 = df[['cid','bytes_returned']].groupby('cid').agg(['count','mean'])
df1.columns = df1.columns.get_level_values(1)
df1 = df1.reset_index()
df1['mean'] = df1['mean'].astype(int)
df1 = df1.rename(columns={"mean": "size"})
df1 = df1.sort_values(by=['size'])
df1.head()
```

Out[3]:

	cid	count	size
135800	QmZiZPaXaT4kSJq6gP3GJ8geNSHxEay8U8EigDhr4x39Gb	1	0
106116	QmXEg9JT6dVPMbmYpY8gWKbeD5fJHdUgcZWTHNLPXM9Vxx	1	0
106115	QmXEfjr121xgyXzU7Uu9U3kFeZh8tmThUvzQXtB94pfuAW	1	0
106105	QmXEdbeckJMpEQbpmsANxK7fPQ8LjYjQJA7ZJFRRPQ24ps	1	0
106103	QmXEc8dmxfTBXrUJaYJJEkiwpGXUDifnKQkrkPwv4cUgkY	1	0

In [4]:

```
df1.shape
```

Out[4]:

```
(254573, 3)
```

In [5]:

```
df1['size'] = df1['size']/1024
df1.describe()
```

Out[5]:

	count	size
count	254573.000000	2.545730e+05
mean	26.095544	8.049844e+02
std	625.841787	7.575891e+03
min	1.000000	0.000000e+00
25%	1.000000	4.121094e-01
50%	1.000000	1.076953e+01
75%	2.000000	2.827988e+02
max	101717.000000	2.702699e+06

In [6]:

```
df2 = pd.DataFrame(columns = ['size', 'count'])

def addRow(df2, l, r, name):
    df_temp = df1[(df1['size'] >= l) & (df1['size'] < r)]
    c = df_temp.count()[0]
    df2 = df2.append({'size':name, 'count':c}, ignore_index = True)
    return df2

df2 = addRow(df2, 0, 1, '<1KB')
df2 = addRow(df2, 1, 4, '1~4KB')
df2 = addRow(df2, 4, 64, '4~64KB')
df2 = addRow(df2, 64, 256, '64KB~256KB')
df2 = addRow(df2, 256, 1024, '256KB~1MB')
df2 = addRow(df2, 1024, 1024*16, '1MB~16MB')
df2 = addRow(df2, 1024*16, 10000000, '>16MB')

# df2.replace('[1,2)', '1')
# df2.replace('[10000,100000)', '[10000,+∞)')

total = df2['count'].sum()
df2['percentage'] = df2['count']/total
df2
```

Out[6]:

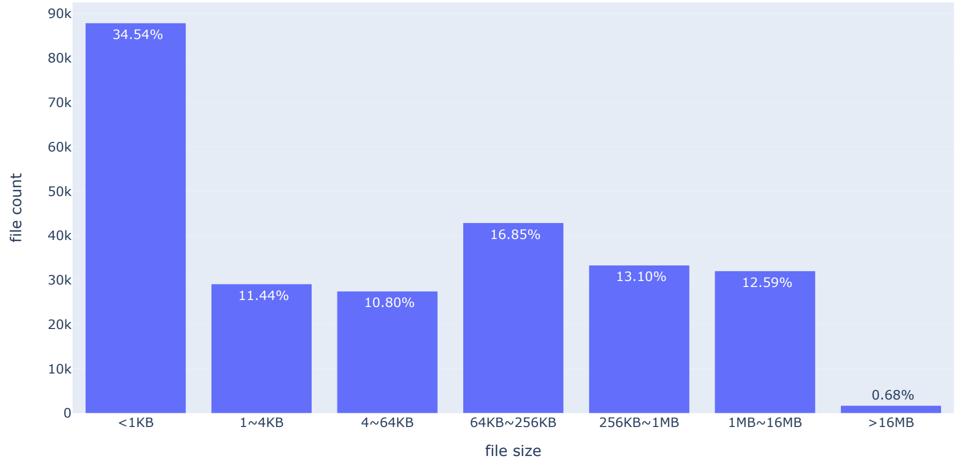
	size	count	percentage
0	<1KB	87919	0.345359
1	1~4KB	29117	0.114376
2	4~64KB	27490	0.107985
3	64KB~256KB	42902	0.168525
4	256KB~1MB	33350	0.131004
5	1MB~16MB	32060	0.125936
6	>16MB	1735	0.006815

In [7]:

```
fig = px.bar(df2, x='size', y='count', text=['\n{0:1.2f}%'.format(x*100) for x in df2['percentage']])

fig.update_xaxes(title='file size')
fig.update_yaxes(title='file count')

fig.show()
```



2 Number of requests and traffic in each size range

In [8]:

```
df3 = df[['cid','bytes_returned']]
df3 = df3.rename(columns={"bytes_returned": "size"})
df3['size'] = df3['size']/1024 #KB
df3.head()
```

Out[8]:

	cid	size
0	QmewCrTqsMECeYcX2etcuRAi2G37yNrL1QBsjxjAgZSwfy	0.413086
1	QmSoLuCB7xeFD5vf8pYnzoBhRFfnnM41nPy4zBnSqmjH7J	181.578125
2	bafybeifyvews52mcsuqfbeoxxlzv5lewk37jc43b5tpbd3gzs3rvcktpaa	453.484375
3	bafybeifqhn5mwknicly5hb72bgs4m2674xu24kxjt7j25ebw2tej5wiiqy	1592.687500
4	QmewCrTqsMECeYcX2etcuRAi2G37yNrL1QBsjxjAgZSwfy	0.402344

In [9]:

```
df3['size_type'] = ''

def addSizeType(l, r, name):
    df3.loc[(df3['size'] >= l) & (df3['size'] < r), 'size_type'] = name

addSizeType(0, 1, '<1KB')
addSizeType(1, 4, '1~4KB')
addSizeType(4, 64, '4~64KB')
addSizeType(64, 256, '64KB~256KB')
addSizeType(256, 1024, '256KB~1MB')
addSizeType(1024, 1024*16, '1MB~16MB')
addSizeType(1024*16, 1024*1024*16, '>16MB')

df3.head()
```

Out[9]:

	cid	size	size_type
0	QmewCrTqsMECeYcX2etcuRAi2G37yNrL1QBsjxjAgZSwfy	0.413086	<1KB
1	QmSoLuCB7xeFD5vf8pYnzoBhRFfnnM41nPy4zBnSqmjH7J	181.578125	64KB~256KB
2	bafybeifyvews52mcsuqfbeoxxlzv5lewk37jc43b5tpbd3gzs3rvcktpaa	453.484375	256KB~1MB
3	bafybeifqhn5mwknicly5hb72bgs4m2674xu24kxjt7j25ebw2tej5wiiqy	1592.687500	1MB~16MB
4	QmewCrTqsMECeYcX2etcuRAi2G37yNrL1QBsjxjAgZSwfy	0.402344	<1KB

In [10]:

```
df4_1 = df3[['size','size_type']].groupby('size_type').agg('sum')
df4_1['size'] = df4_1['size']/pow(1024,2) # GB
df4_1 = df4_1.reset_index()

df4_2 = df3[['cid','size_type']].groupby('size_type').agg('count')
df4_2 = df4_2.reset_index()
df4_2 = df4_2.rename(columns={"cid": "count"})

df4 = df4_1.set_index('size_type').join(df4_2.set_index('size_type'))
df4 = df4.reset_index()

cat_size_order = CategoricalDtype(
    ['<1KB', '1~4KB', '4~64KB', '64KB~256KB', '256KB~1MB', '1MB~16MB', '>16MB'],
    ordered=True
)
df4['size_type'] = df4['size_type'].astype(cat_size_order)
df4 = df4.sort_values('size_type')
df4.head()
```

Out[10]:

	size_type	size	count
5	<1KB	0.228333	1283298
1	1~4KB	0.267443	122601
3	4~64KB	6.692936	306100
4	64KB~256KB	109.824447	756815
2	256KB~1MB	1158.624712	1995383

In [11]:

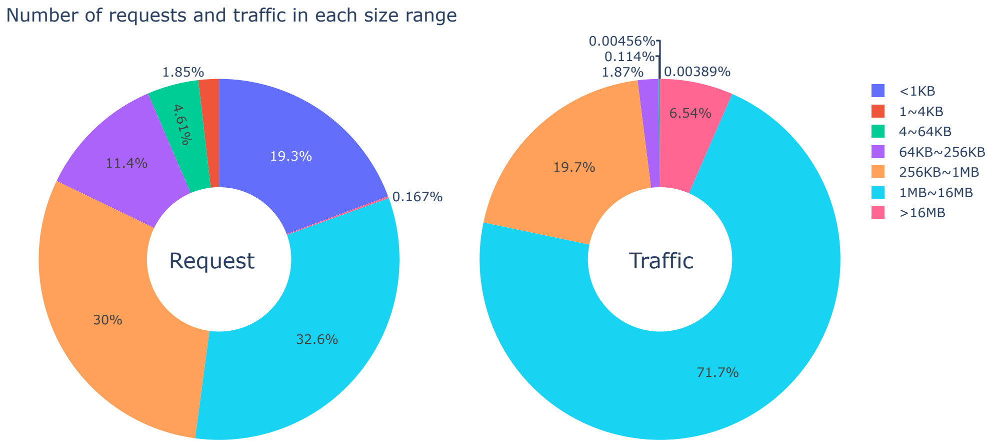
```
# create subplots: use 'domain' type for Pie subplot
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]])

fig.add_trace(go.Pie(
        labels=df4['size_type'],
        values=df4['size'],
        sort=False),1,2
    )
fig.add_trace(go.Pie(
        labels=df4['size_type'],
        values=df4['count'],
        sort=False),1,1
    )


# use `hole` to create a donut-like pie chart
fig.update_traces(hole=.4, hoverinfo="label+percent+name")

fig.update_layout(
    title_text="Number of requests and traffic in each size range",
    # add annotations in the center of the donut pies.
    annotations=[dict(text='Request', x=0.16, y=0.5, font_size=20, showarrow=False),
                 dict(text='Traffic', x=0.82, y=0.5, font_size=20, showarrow=False)])

fig.show()
```


Number of requests and traffic in each size range
Request
Traffic
<1KB
1~4KB
4~64KB
64KB~256KB
256KB~1MB
1MB~16MB
>16MB