Baleen数据集分析

任务1：负载总的请求数量和读写比例

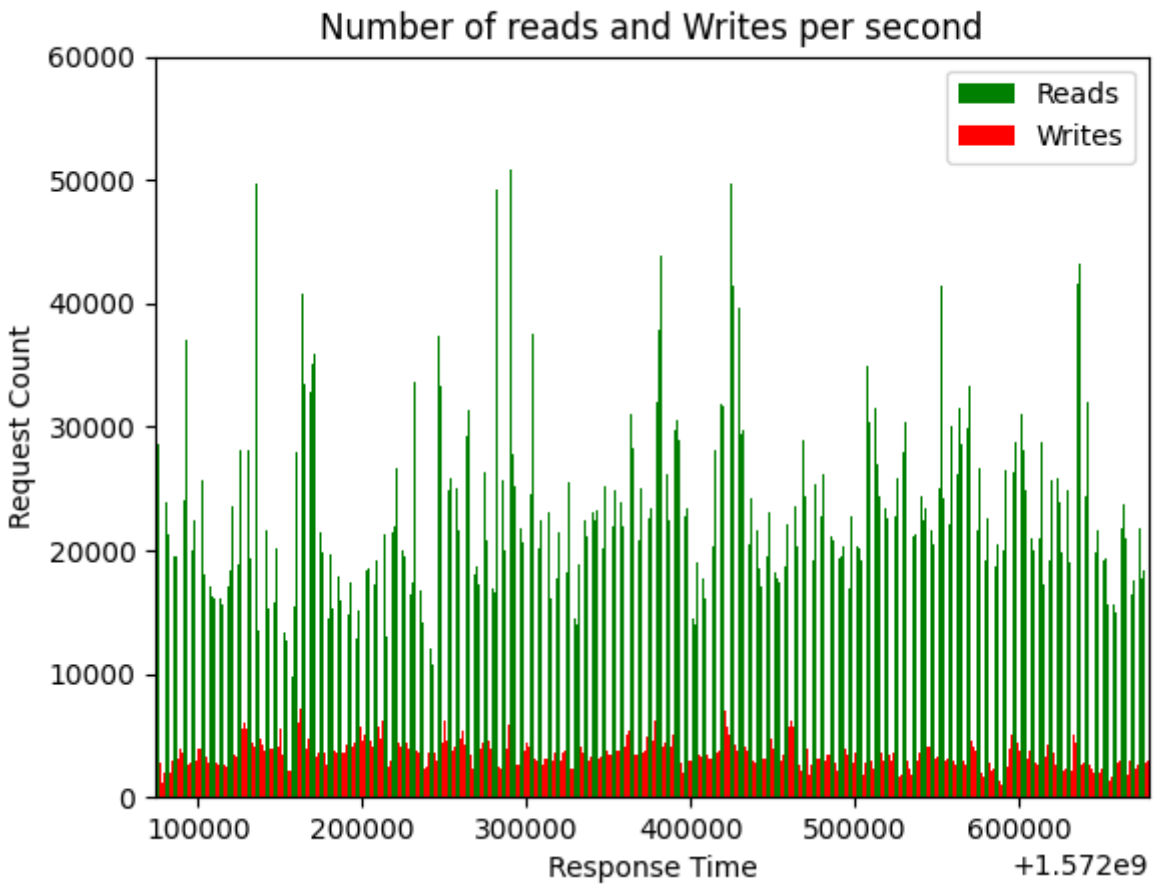
代码：

```
import os
import pandas as pd
import matplotlib.pyplot as plt

# 定义函数来解析每行数据并计算指标
def process_trace_file(file_path):
    total_io_size = 0
    read_count = 0
    write_count = 0
    io_count = 0
    min_time = float('inf')
    max_time = 0

    with open(file_path, 'r') as file:
        lines = file.readlines()[2:]
        for line in lines:

            # 解析每行数据
            parts = line.split()
            io_size = int(parts[2])
            time = float(parts[3])
            op_name = parts[4]

            # 计算指标
            max_time = max(max_time, time)
            min_time = min(min_time, time)
            total_io_size += io_size
            io_count += 1
            if op_name == '1' or op_name == '2' or op_name == '5':
                read_count += 1
            elif op_name == '3' or op_name == '4' or op_name == '6':
                write_count += 1

    return {
        'Name': os.path.basename(file_path),
        'Number of IOs': io_count,
        'IOPS': round(io_count / (max_time - min_time), 2),
        'RD/WT Ratio': round(read_count / write_count, 1) if write_count != 0 else "N/A",
        'Req. Size': str(round(total_io_size / io_count / 1000, 1)) + 'KB',
    }

data_directorys = ['C:\\users\\CC507\\Desktop\\storage\\20230325\\Region7\\']
results = []
```

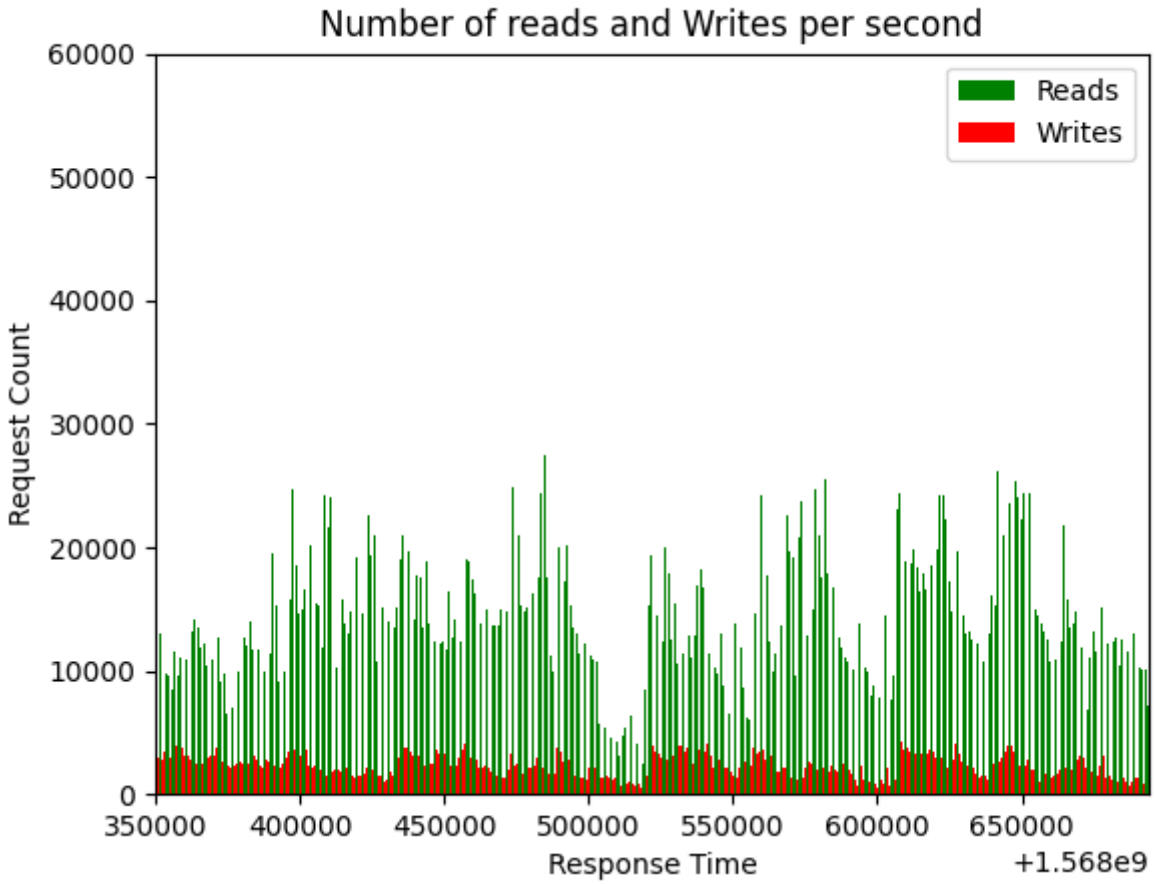
```
for data_directory in data_directorys:
    for data_set in os.listdir(data_directory):
        if data_set.endswith(".trace"):
            trace_file = os.path.join(data_directory, data_set)
            result = process_trace_file(trace_file)
            results.append(result)

# 绘制表格
table_data = []
table_headers = ['Name', 'Number of IOs', 'IOPS', 'RD/WT Ratio', 'Req. Size']

# 转换结果为DataFrame
df = pd.DataFrame(results, columns=['Name', 'Number of IOs', 'IOPS', 'RD/WT Ratio', 'Req.
Size'])
# 输出表格
fig, ax = plt.subplots()
ax.axis('off')
ax.table(cellText=df.values, colLabels=df.columns, loc='center')
# 保存为PDF格式
plt.savefig('C:\\users\\CC507\\Desktop\\table.pdf', format='pdf', bbox_inches='tight')
plt.show()
```

任务2：负载的读写请求数随时间变化示意图

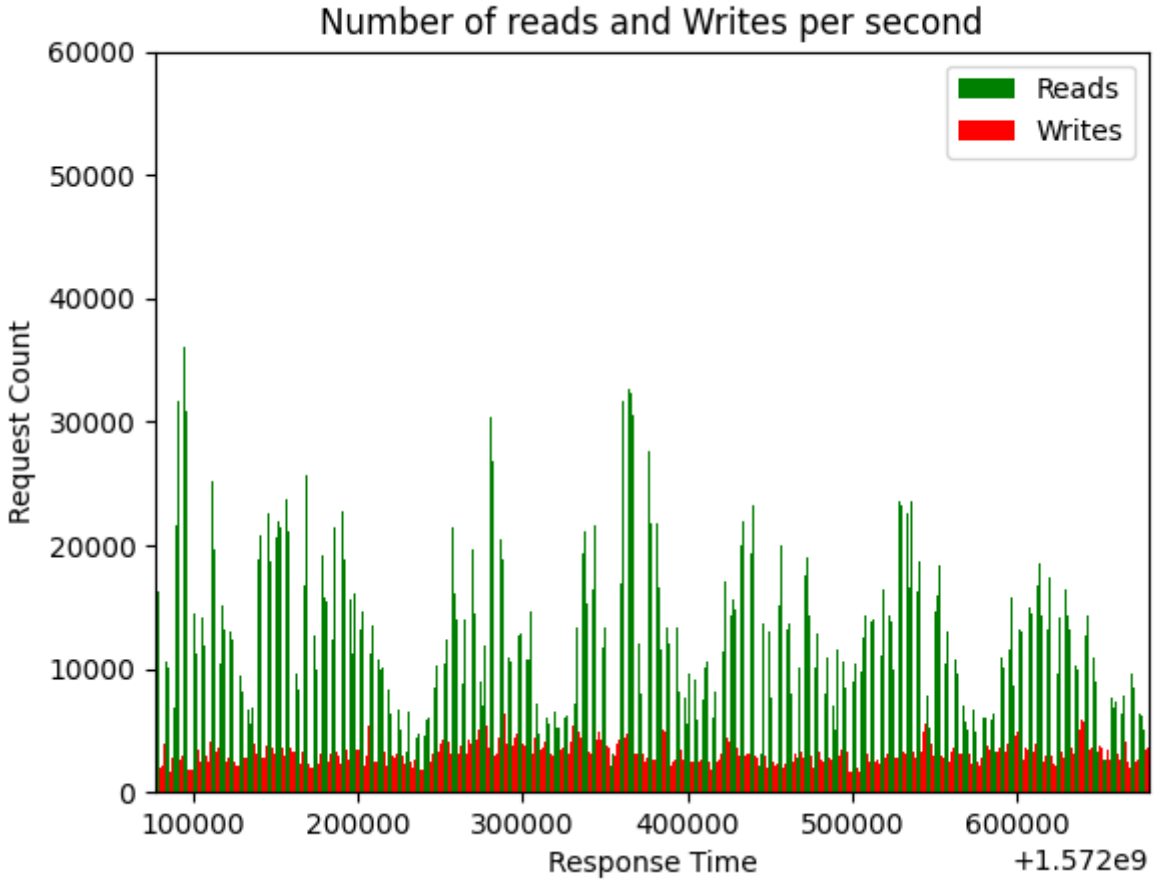
Region1



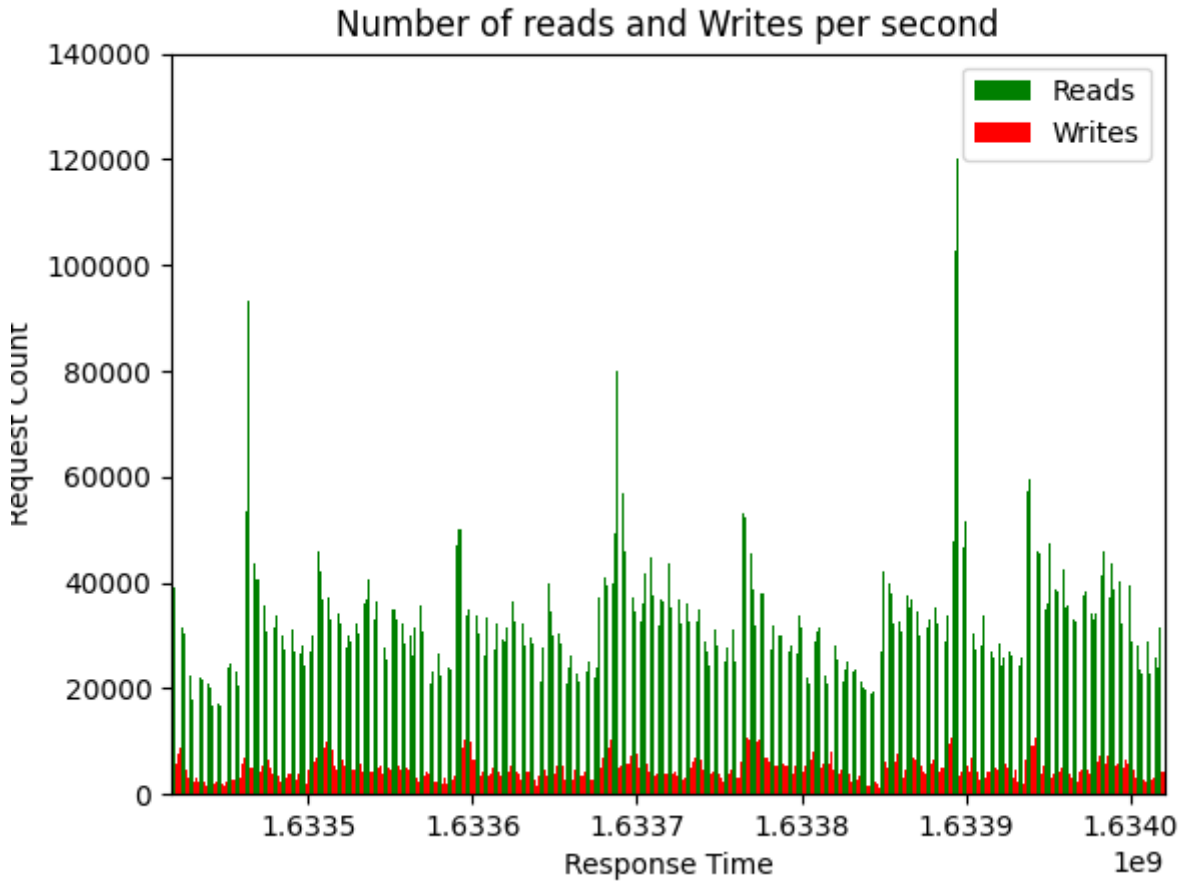
Region2



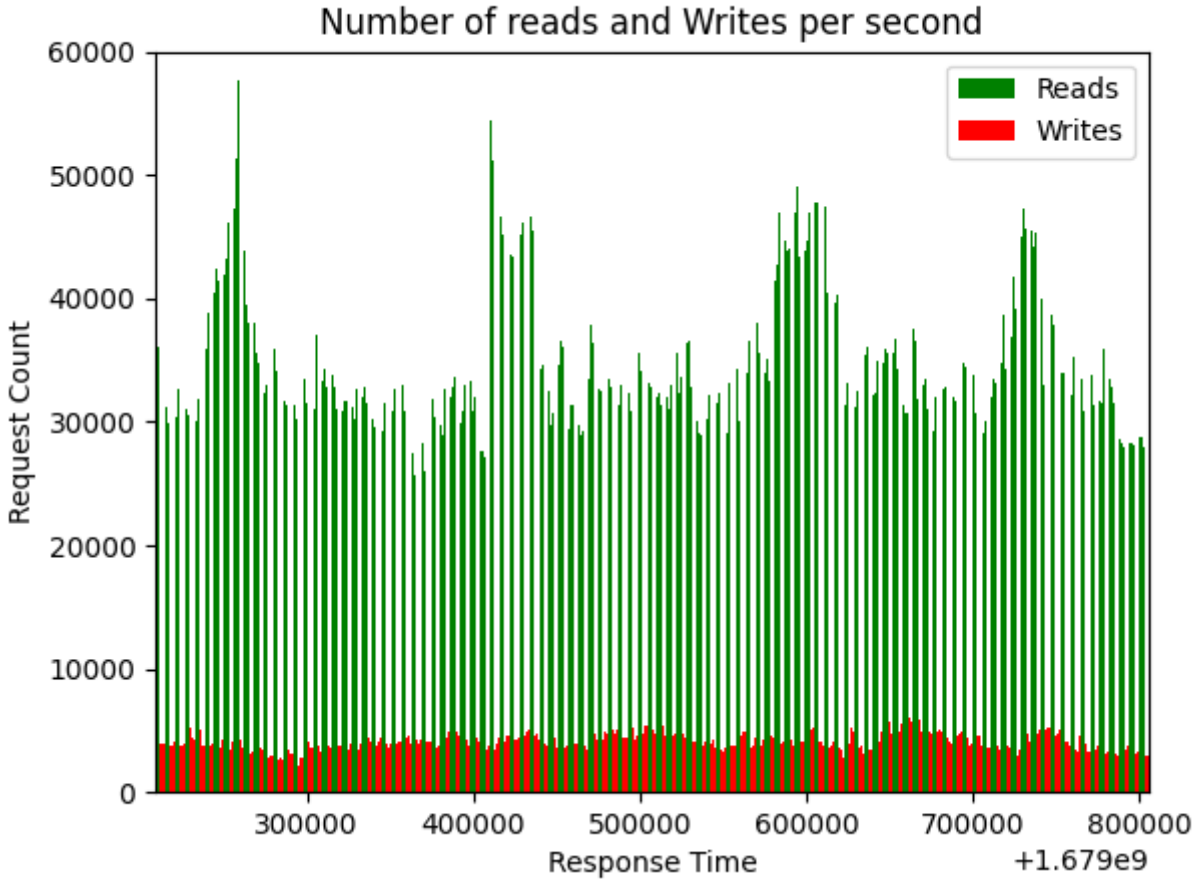
Region3



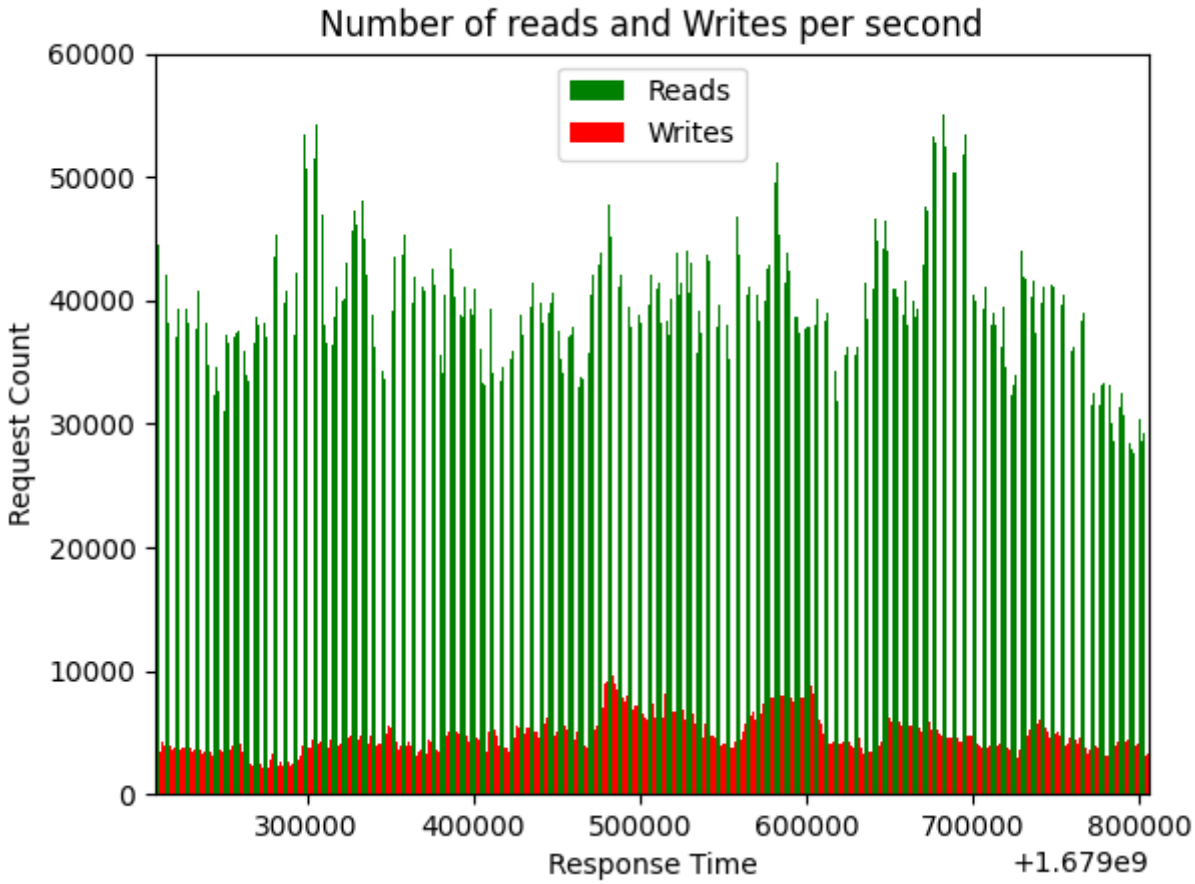
Region4



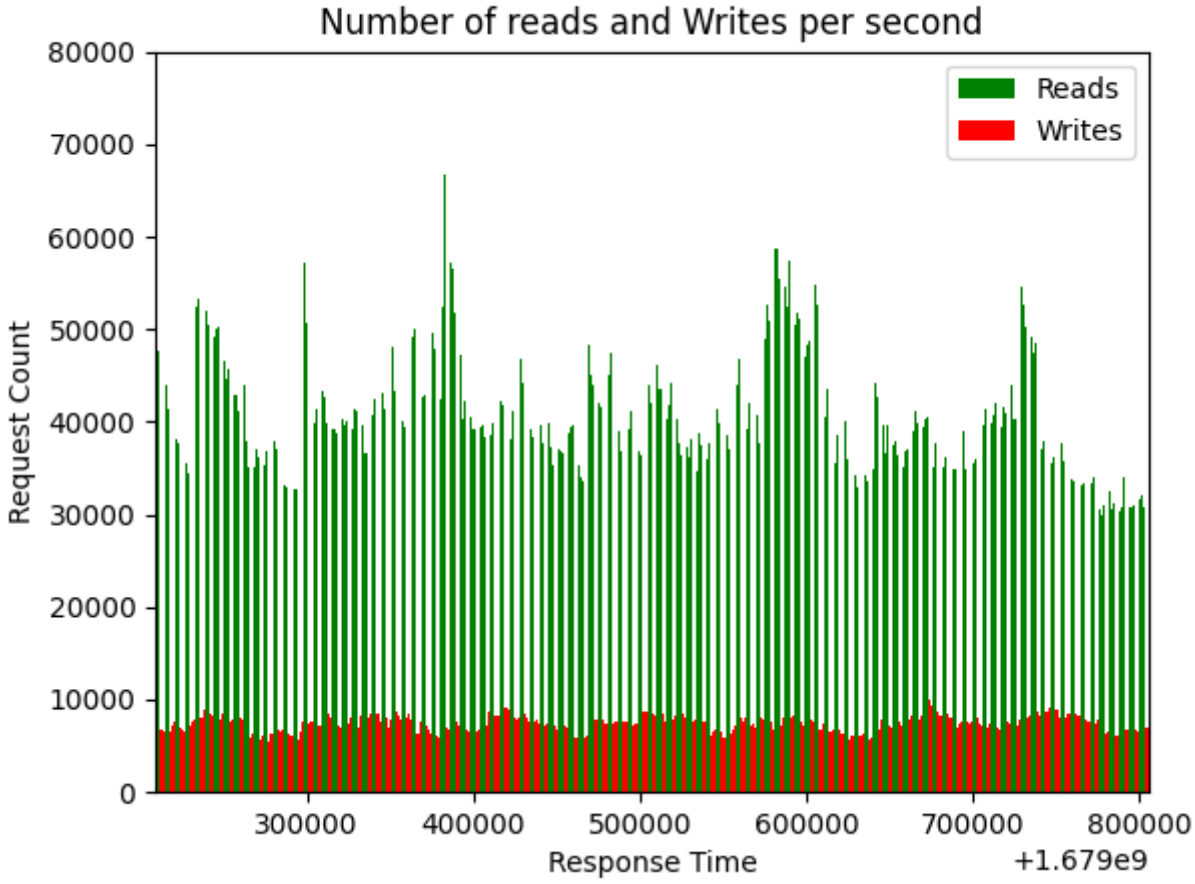
Region5



Region6


Number of reads and Writes per second
Reads
Writes
Request Count
60000
50000
40000
30000
20000
10000
0
300000
400000
500000
600000
700000
800000
Response Time
+1.679e9

Region7


Number of reads and Writes per second
Reads
Writes
Request Count
80000
70000
60000
50000
40000
30000
20000
10000
0
300000
400000
500000
600000
700000
800000
Response Time
+1.679e9

代码：

```python
import numpy as np
import matplotlib.pyplot as plt
import os

directorys = ['C:\\users\\CC507\\Desktop\\storage\\20230325\\Region7\\']  # 数据集所在目录

# 初始化计数器
read_count = np.zeros(1000)
write_count = np.zeros(1000)
time_range = np.arange(0, 1000, 1)
offset = 1679000000
compress_ratio = 1000
min_time = float('inf')
max_time = 0

for directory in directorys:
    for filename in os.listdir(directory):
        if filename.endswith(".trace"):
            file_path = os.path.join(directory, filename)
            with open(file_path, 'r') as data_file:
                lines = data_file.readlines()[2:]  # 从第二行开始读取
                for line in lines:
                    fields = line.split()
                    timestamp = float(fields[3])
                    operation_type = fields[4]
                    # 判断读写类型，并根据时间戳确定时间范围
                    index = int((timestamp - offset) / compress_ratio)
                    max_time = max(max_time, index)
                    min_time = min(min_time, index)
                    if operation_type == '1' or operation_type == '2' or operation_type == '5':
                        read_count[index] += 1
                    if operation_type == '3' or operation_type == '4' or operation_type == '6':
                        write_count[index] += 1

# 绘制柱状图
plt.bar(time_range * compress_ratio + offset - compress_ratio / 4, read_count,
width=compress_ratio/2, color='green', label='Reads', bottom=0)
plt.bar(time_range * compress_ratio + offset + compress_ratio / 4, write_count,
width=compress_ratio/2, color='red', label='Writes', bottom=0)
plt.xlabel('Response Time')
plt.ylabel('Request Count')
plt.title('Number of reads and Writes per second')
plt.legend()
# 设置横坐标刻度
plt.ylim(0, 80000)
plt.xlim(min_time * compress_ratio + offset, max_time * compress_ratio + offset)
plt.show()
```

任务3：负载的平均IOPS?

Region1

Average IOPS：26.417220075202227

Region2

Average IOPS：16.568953264403344

Region3

Average IOPS：15.899508172060026

Region4

Average IOPS：37.76018409228419

Region5

Average IOPS：38.84084977380411

Region6

Average IOPS：44.538175703936695

Region7

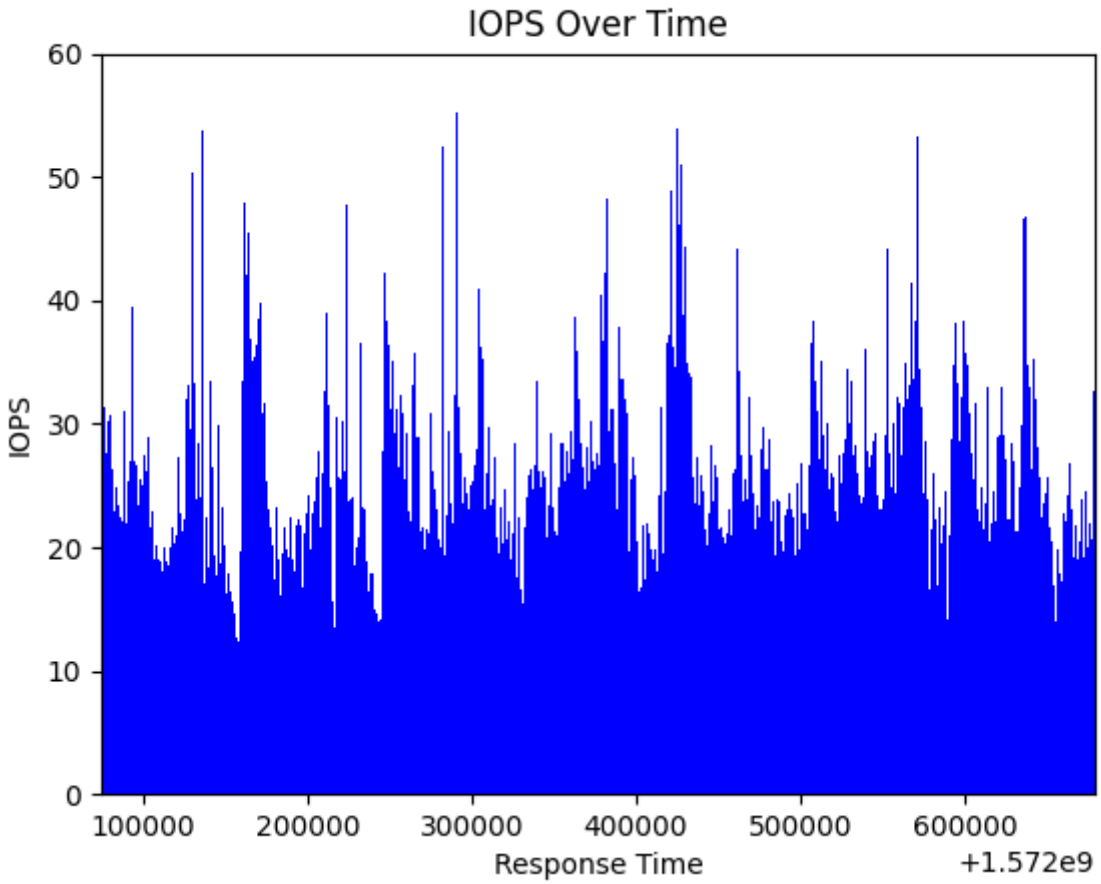
Average IOPS：48.256225406215705

代码：

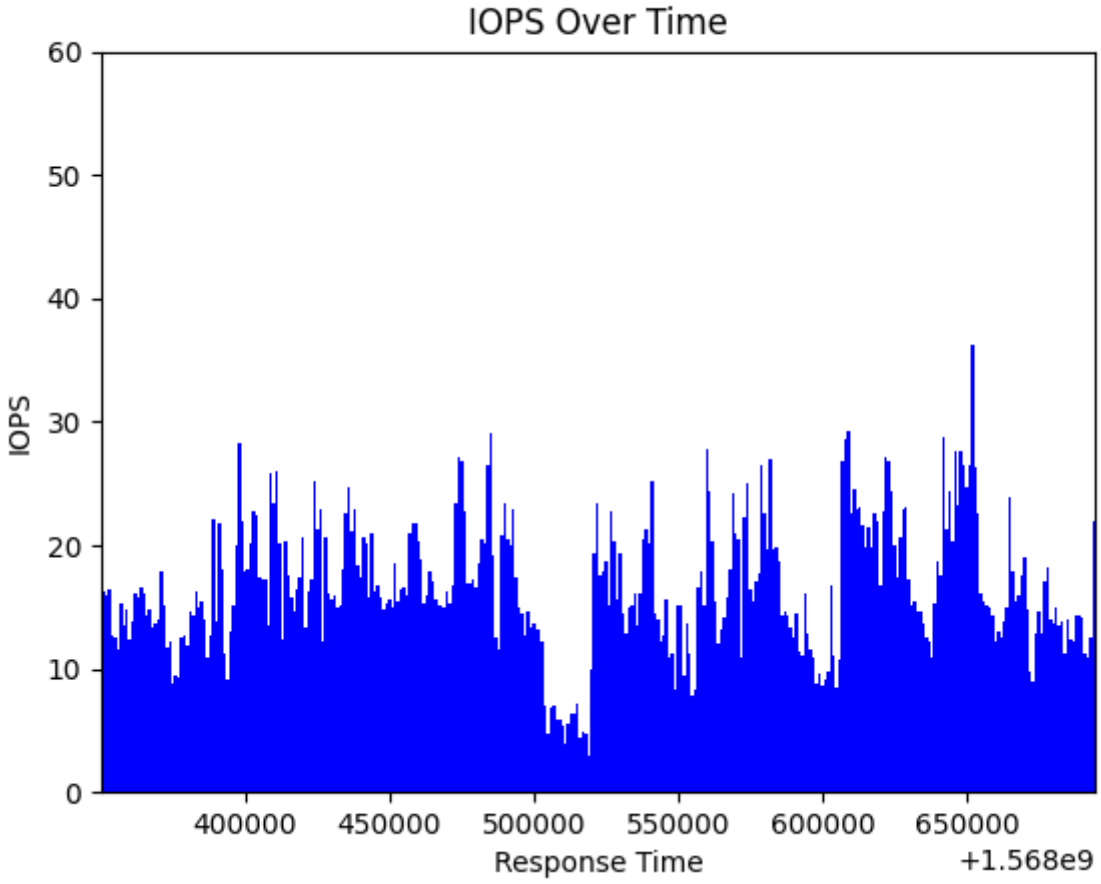
```
import os

io_count = 0
max_time = 0
min_time = float('inf')

data_directorys = ['C:\\users\\CC507\\Desktop\\storage\\20230325\\Region7\\']

for data_directory in data_directorys:
    for data_set in os.listdir(data_directory):
        if data_set.endswith(".trace"):
            trace_file = os.path.join(data_directory, data_set)
            with open(trace_file, 'r') as file:
                lines = file.readlines()[2:]  # 从第二行开始读取
                for line in lines:
                    parts = line.split()
                    time = float(parts[3])
                    # 计算指标
                    max_time = max(max_time, time)
                    min_time = min(min_time, time)
                    io_count += 1

print(io_count / (max_time - min_time))
```

任务4：负载的IOPS随时间变化示意图?

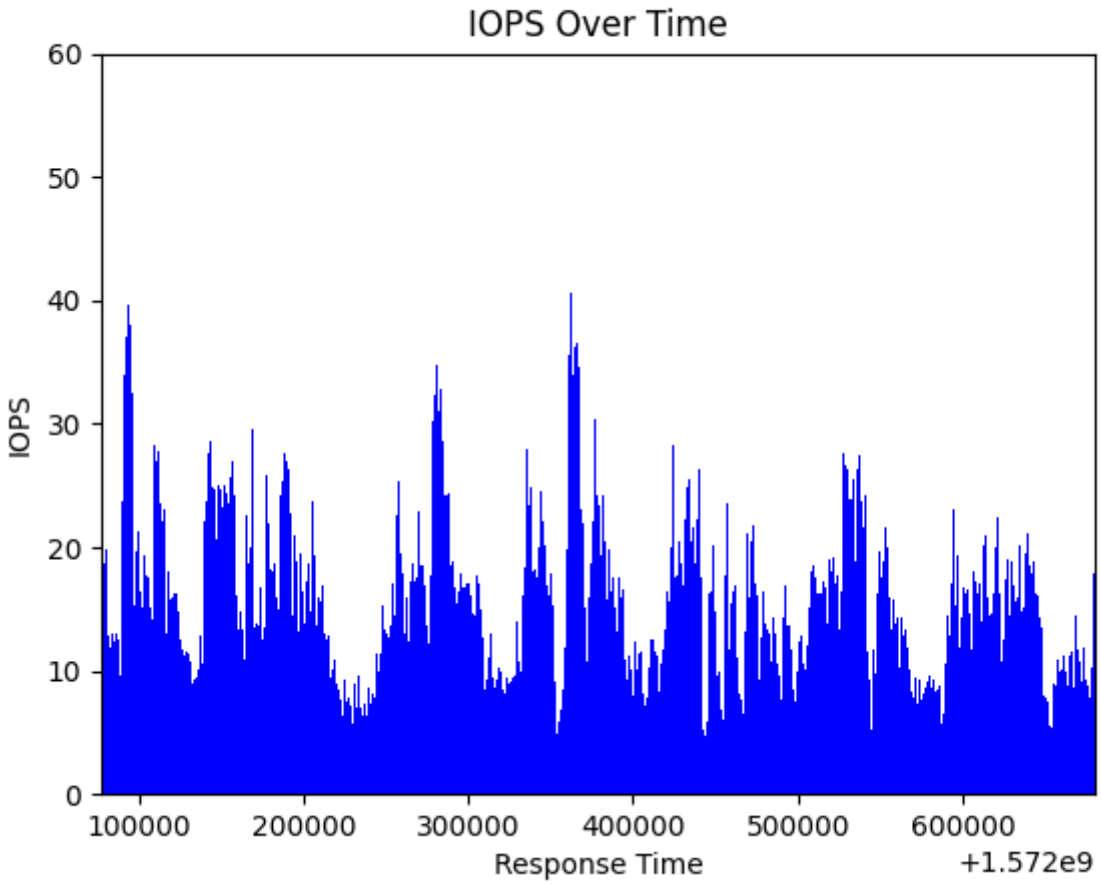
Region1



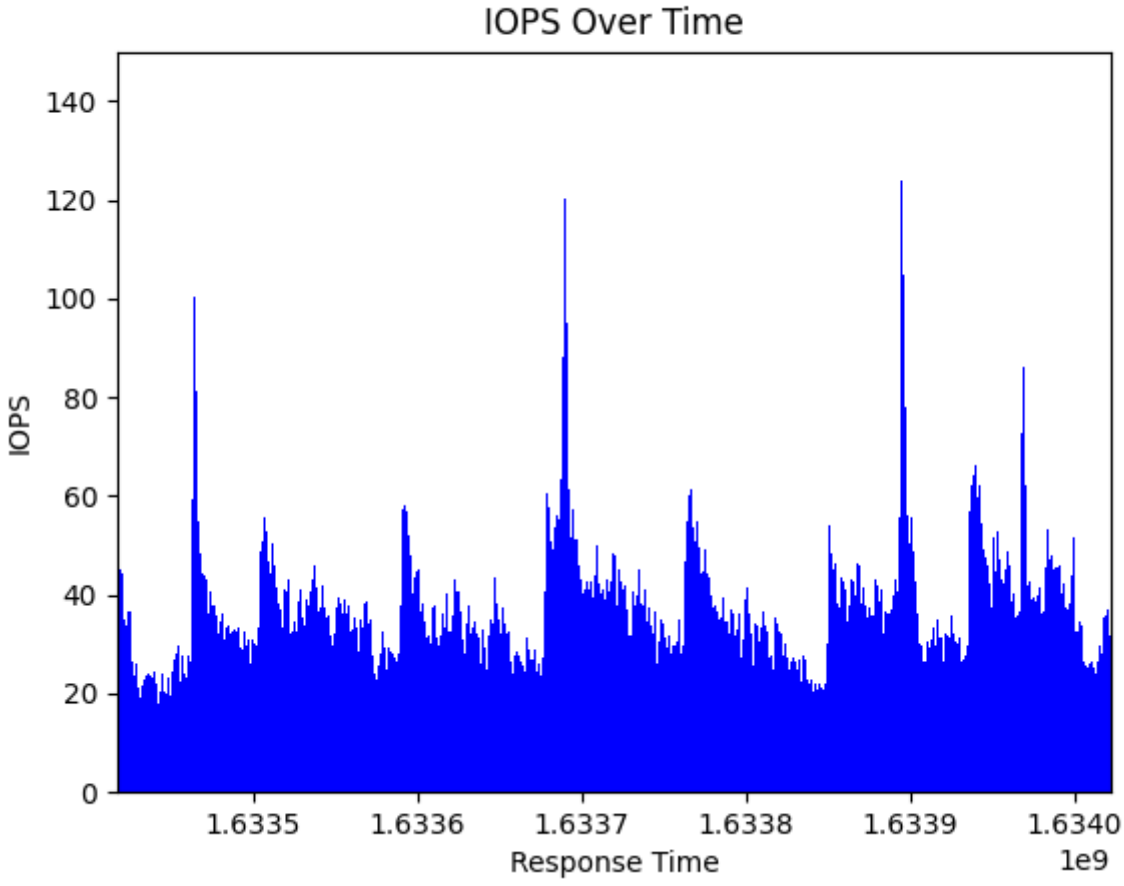
Region2



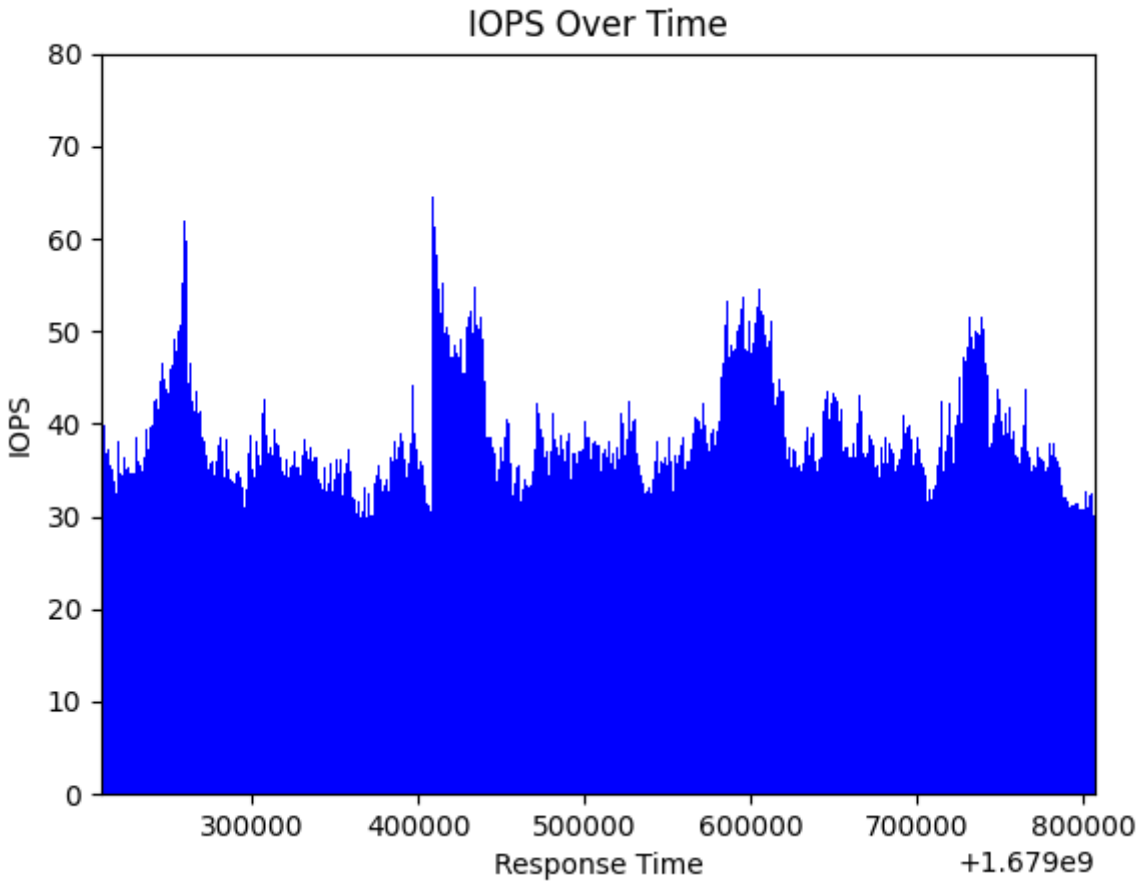
Region3



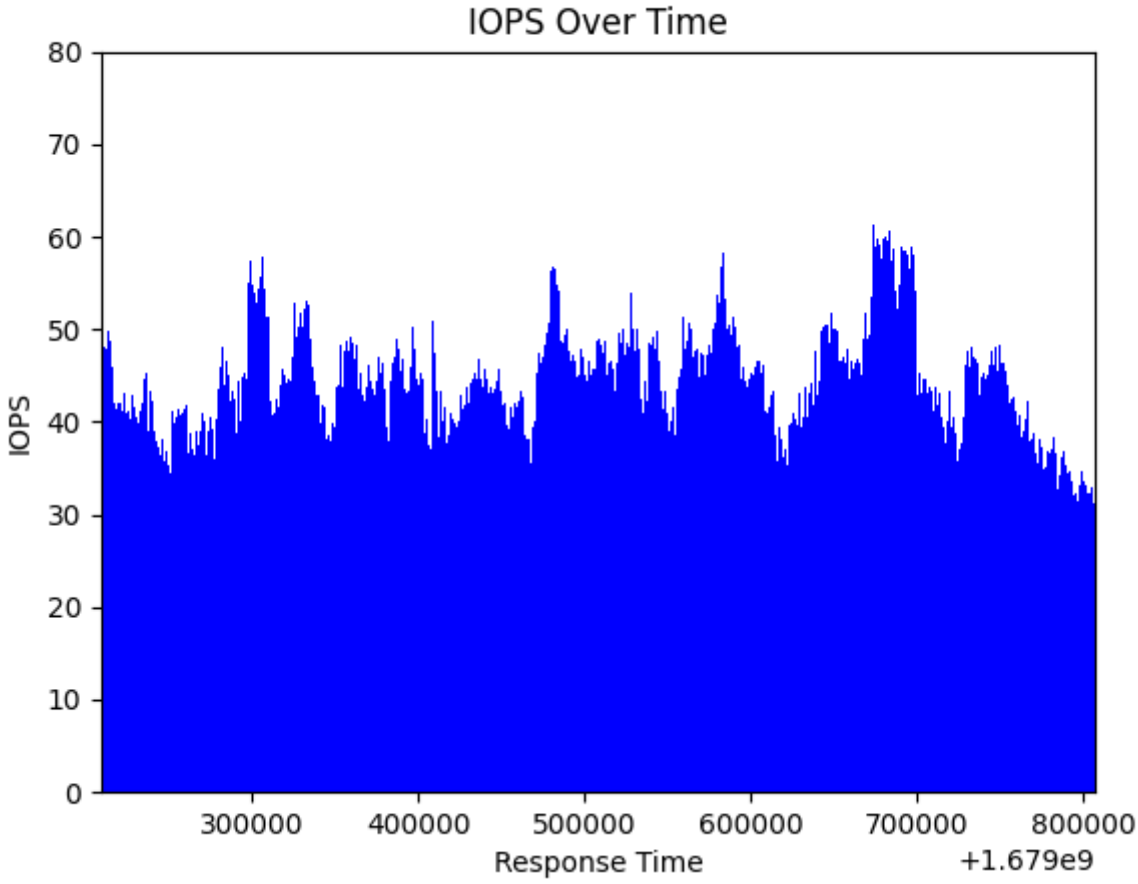
Region4



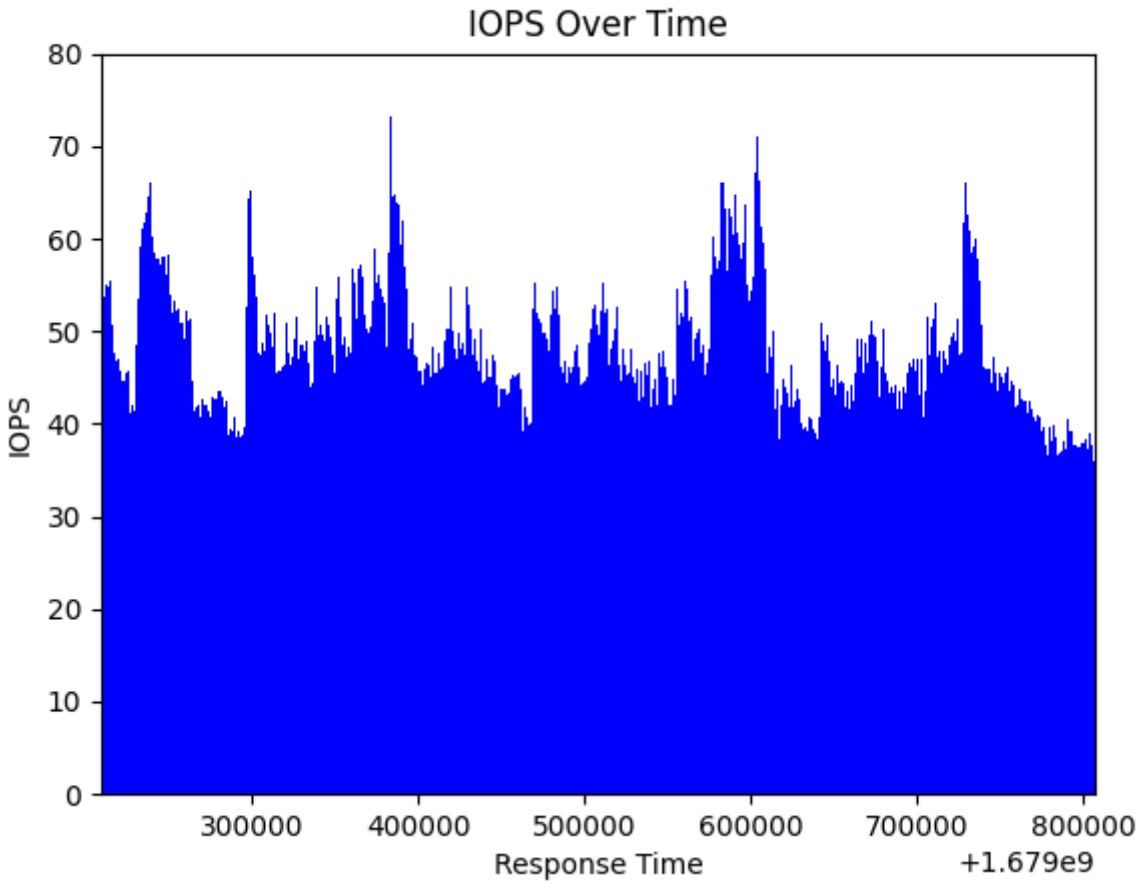
Region5



Region6



Region7



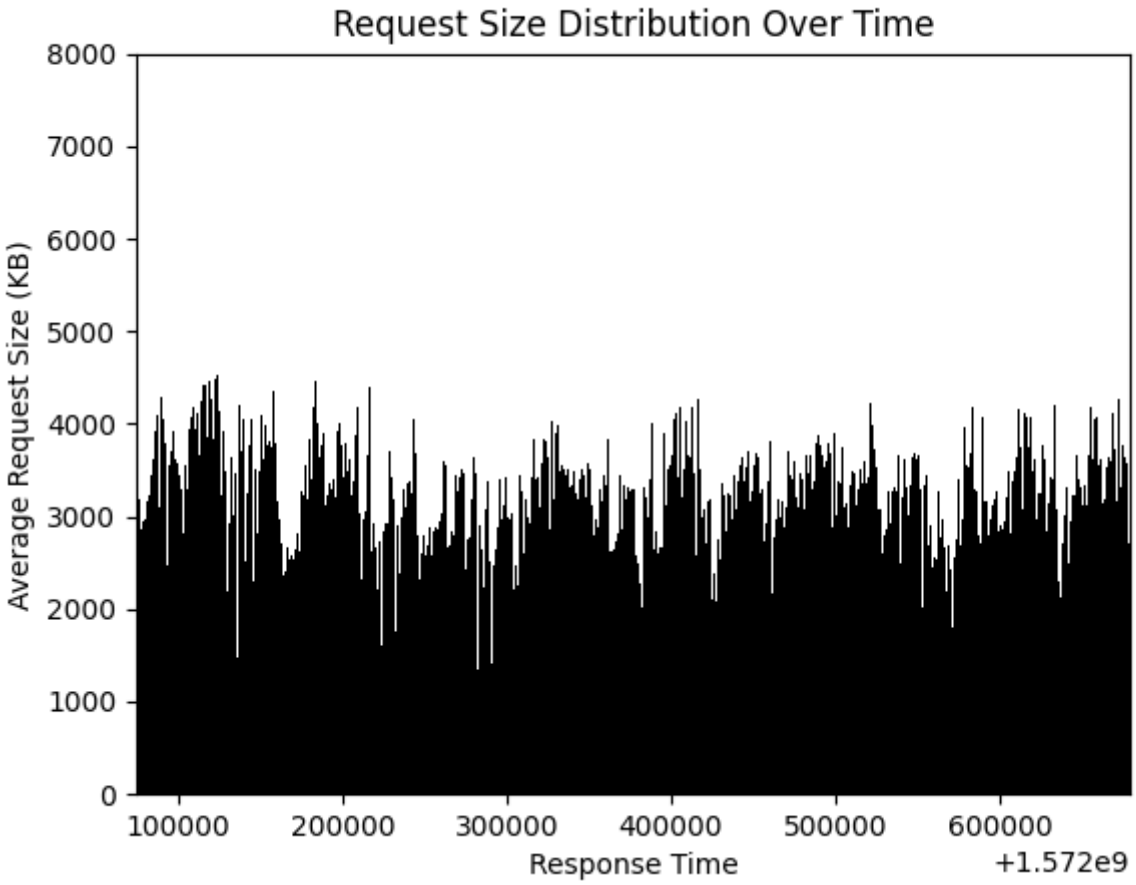
代码:

```
import numpy as np
import matplotlib.pyplot as plt
import os

directorys = ['C:\\users\\CC507\\Desktop\\storage\\20230325\\Region7\\']  # 数据集所在目录

# 初始化计数器
arr_size = 1100
iops = np.zeros(arr_size)
time_range = np.arange(0, arr_size, 1)
offset = 1679000000
compress_ratio = 1000
min_time = float('inf')
max_time = 0
min_time_temp = np.full(arr_size, float('inf'))
max_time_temp = np.zeros(arr_size)
io_count = np.zeros(arr_size)

for directory in directorys:
    for filename in os.listdir(directory):
        if filename.endswith(".trace"):
            file_path = os.path.join(directory, filename)
            with open(file_path, 'r') as data_file:
                lines = data_file.readlines()[2:]  # 从第二行开始读取
                for line in lines:
                    fields = line.split()
                    timestamp = float(fields[3])
                    max_time = max(max_time, timestamp)
```

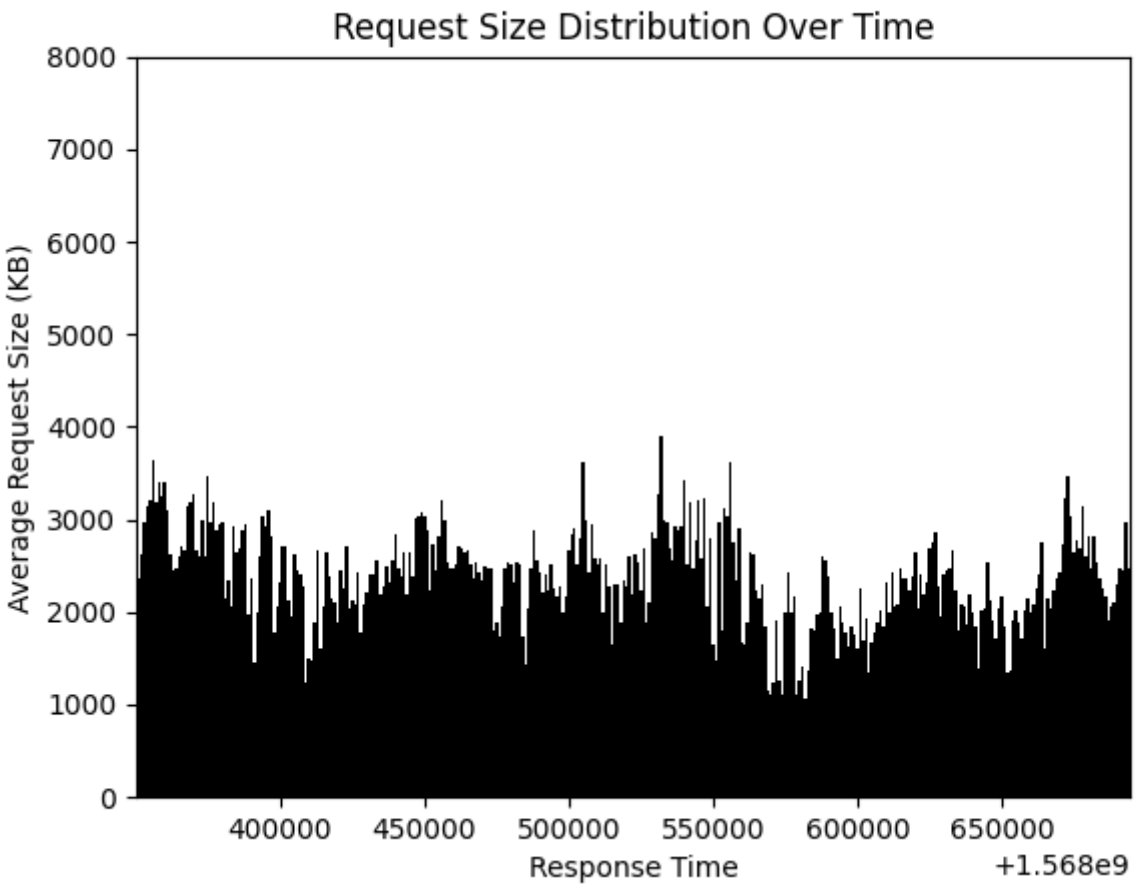
```
                    min_time = min(min_time, timestamp)
                    index = int((timestamp - offset) / compress_ratio)
                    max_time_temp[index] = max(float(max_time_temp[index]), timestamp)
                    min_time_temp[index] = min(float(min_time_temp[index]), timestamp)
                    io_count[index] += 1

for i in range(arr_size):
    iops[i] = io_count[i] / (max_time_temp[i] - min_time_temp[i] + 1)

# 绘制柱状图
plt.bar(time_range * compress_ratio + offset, iops, width=compress_ratio, color='blue',
label='IO Size', bottom=0)
plt.xlabel('Response Time')
plt.ylabel('IOPS')
plt.title('IOPS Over Time')
# 设置横坐标刻度
plt.ylim(0, 80)
plt.xlim(min_time, max_time)
plt.show()
```

任务5：负载的请求大小分布

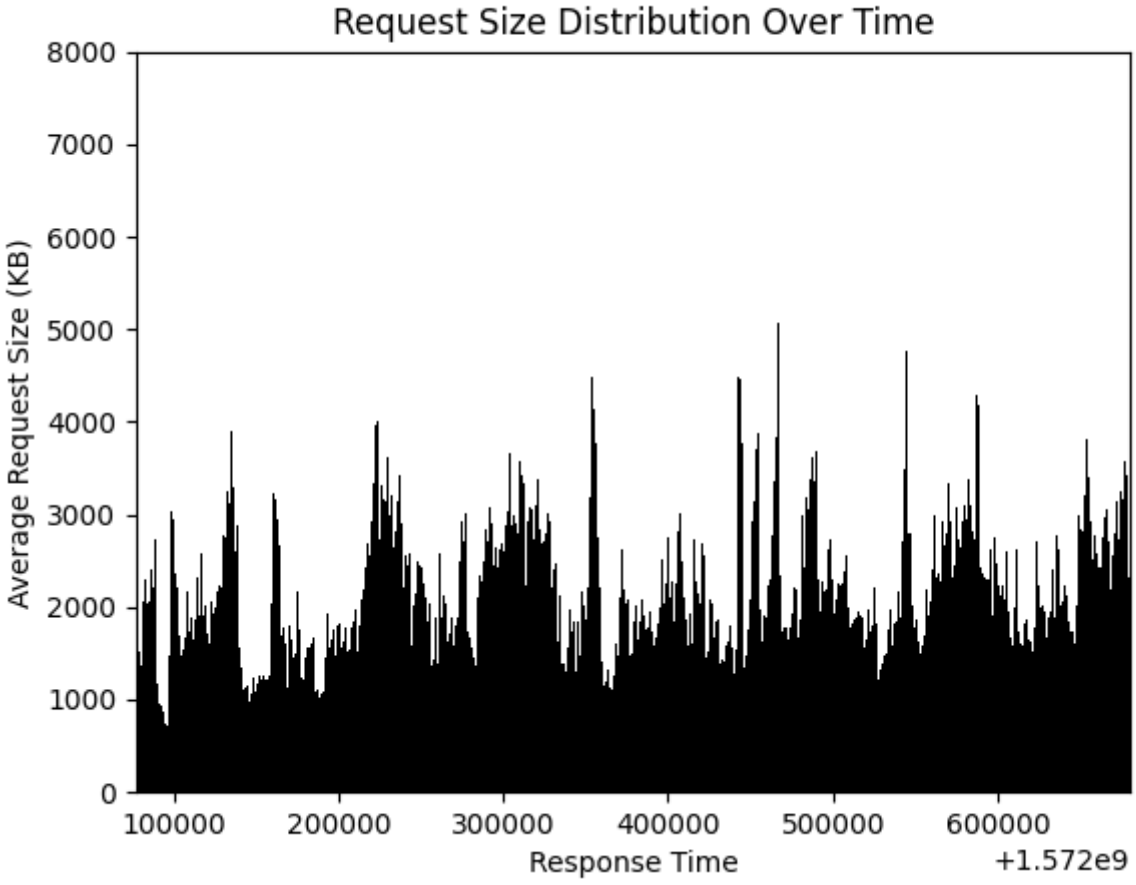
Region1



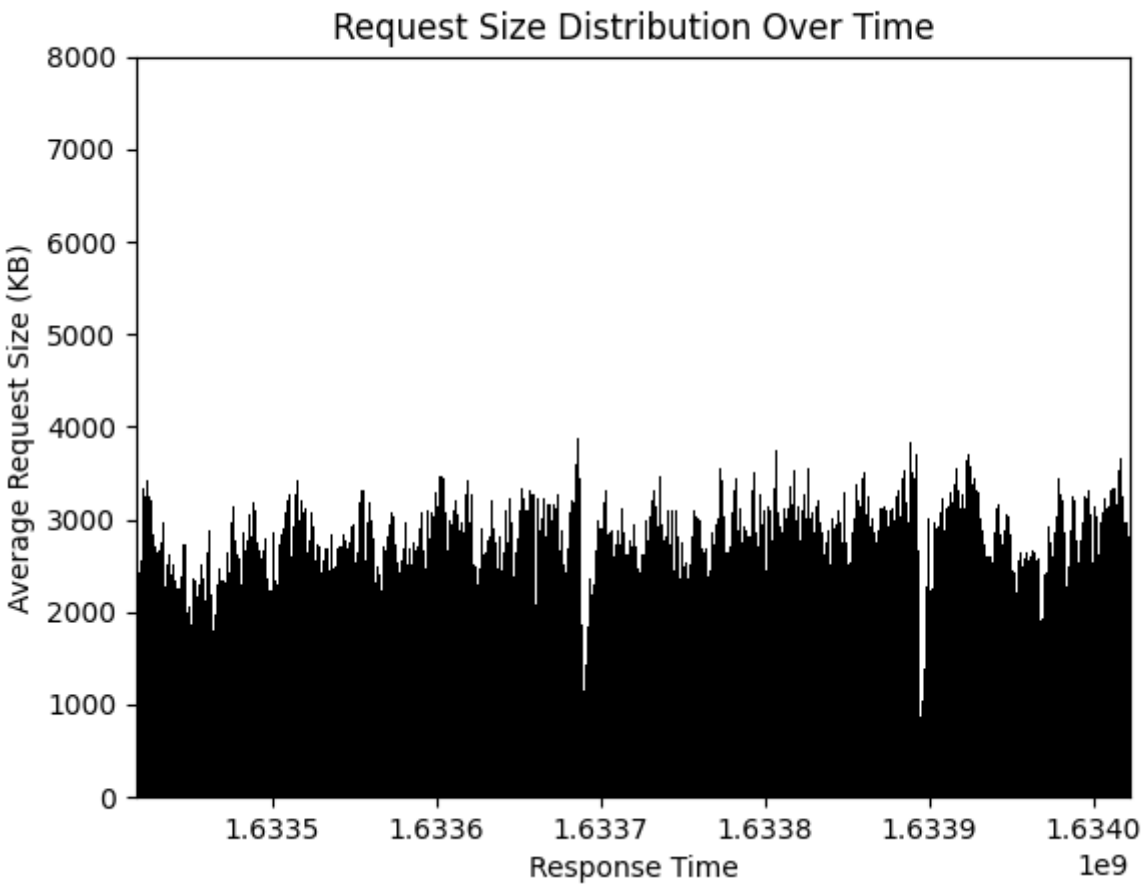
Region2



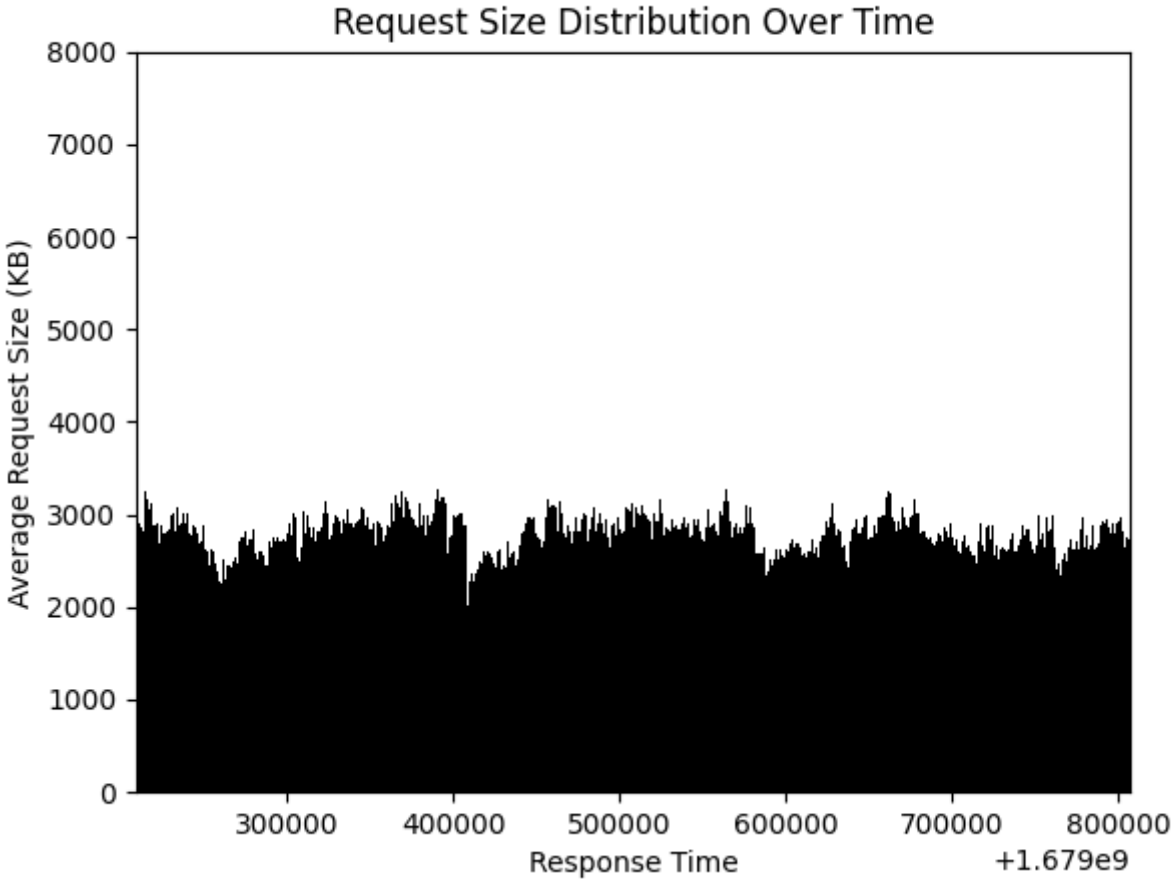
Region3



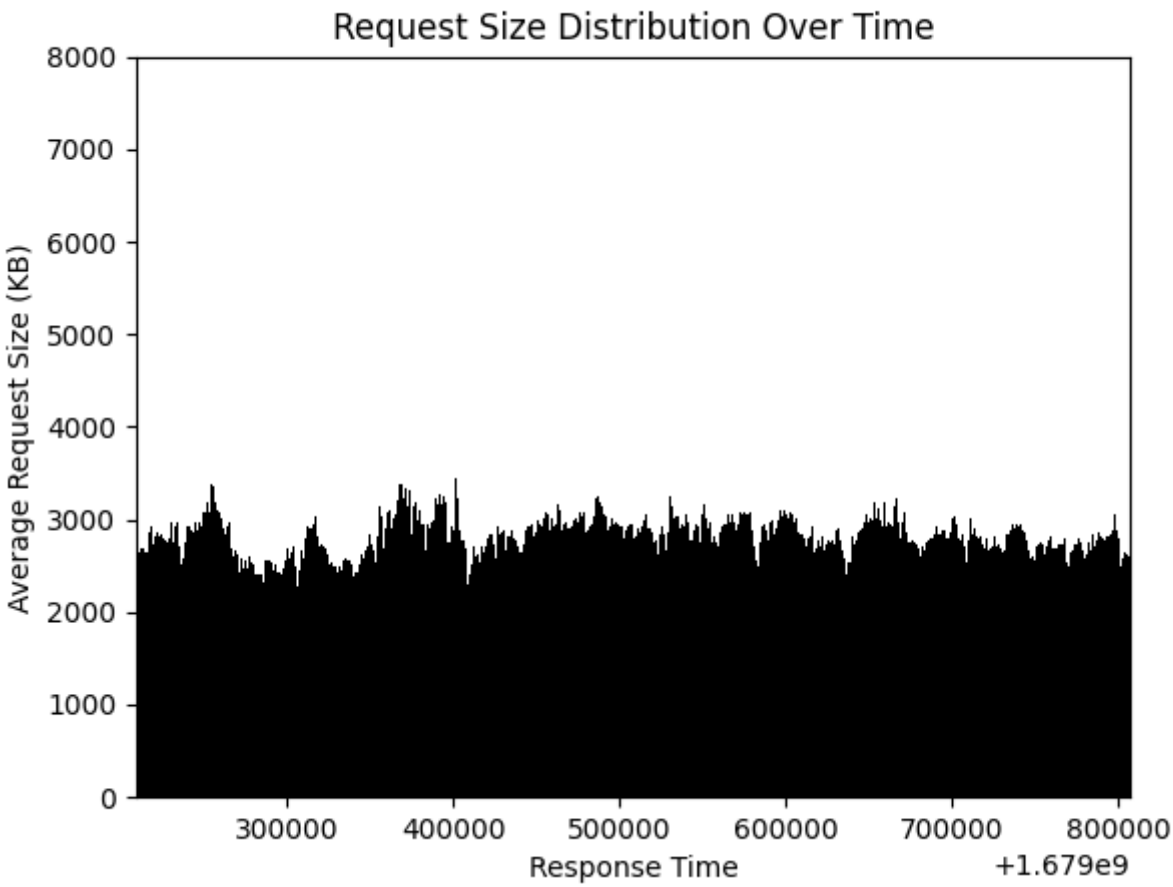
Region4



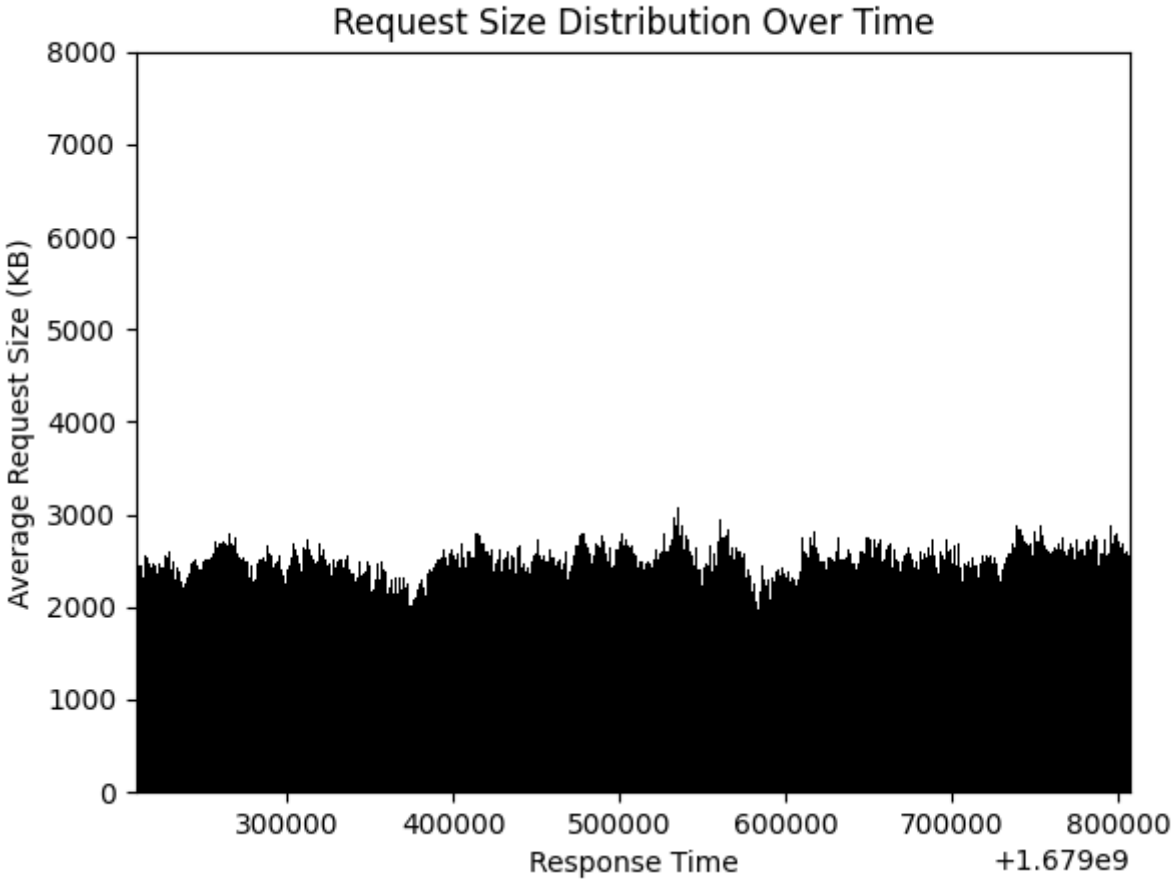
Region5



Region6



Region7



```python
import numpy as np
import matplotlib.pyplot as plt
import os

directorys = ['C:\\users\\CC507\\Desktop\\storage\\20230325\\Region7\\']  # 数据集所在目录

# 初始化计数器
arr_size = 1100
io = np.zeros(arr_size)
time_range = np.arange(0, arr_size, 1)
offset = 1679000000
compress_ratio = 1000
min_time = float('inf')
max_time = 0
io_count = np.zeros(arr_size)
io_size_count = np.zeros(arr_size)

for directory in directorys:
    for filename in os.listdir(directory):
        if filename.endswith(".trace"):
            file_path = os.path.join(directory, filename)
            with open(file_path, 'r') as data_file:
                lines = data_file.readlines()[2:]  # 从第二行开始读取
                for line in lines:
                    fields = line.split()
                    timestamp = float(fields[3])
                    io_size = float(fields[2])
                    index = int((timestamp - offset) / compress_ratio)
                    max_time = max(max_time, timestamp)
                    min_time = min(min_time, timestamp)
                    io_size_count[index] += io_size
                    io_count[index] += 1

for i in range(arr_size):
    io[i] = io_size_count[i] / io_count[i] / 1024

# 绘制柱状图
plt.bar(time_range * compress_ratio + offset, io, width=compress_ratio, color='black', label='IO
Size', bottom=0)
plt.xlabel('Response Time')
plt.ylabel('Average Request Size (KB)')
plt.title('Request Size Distribution Over Time')
# 设置横坐标刻度
plt.ylim(0, 8000)
plt.xlim(min_time, max_time)
plt.show()
```