

# 软件测试 实验报告—JProfile

## 1、实验目的

学习性能监测工具 JProfiler 的使用，测试的程序可以采用视频中给出的（测试时的参数另行选择，比如网页访问次数以及 jsp 里面内嵌的循环次数等），同时添加附件里给出的测试文件；也可以自行选择合适的程序。要求：

- （1）提交 word 版实验报告，模板采用超星上给出的。
- （2）报告中要给出 CPU、Memory 等监测的设计过程以及对应的分析结果截图。
- （3）根据检测分析结果，指出如果要优化程序代码，应着重优化哪一部分。

## 2、测试范围

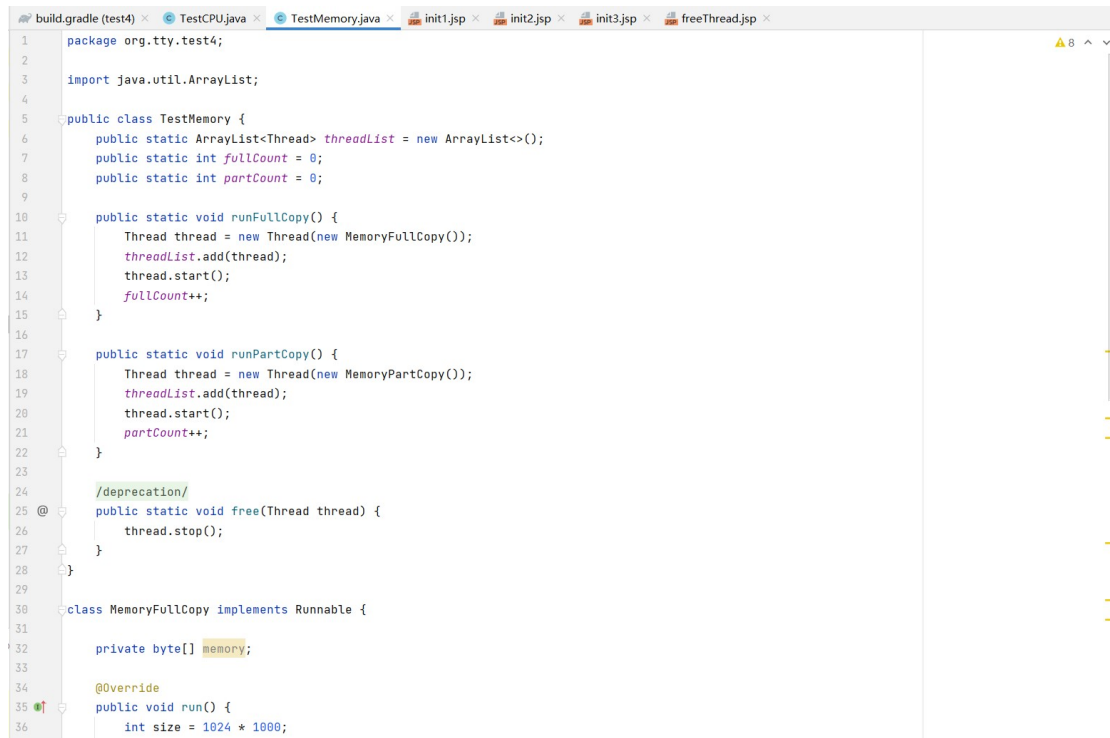
对 CPU、Memory 的使用情况进行监测。

## 3、测试过程

我们对性能测试进行模拟，模拟文件的传输工作，其中文件的 size 控制在 1G 以内，输入参数为?MB。并使用两种方式进行，第一种方式直接载入所有的文件到内存，然后下载。第二种方式使用缓冲区依次读取块进行下载，进行性能上的比较。为了进行性能上的模拟，我们并不进行下载模拟，而是模拟内存块的拷贝工作来实现。

### 3.1 编写模拟内存写入覆盖的代码

TestMemory.java



```
1 package org.tty.test4;
2
3 import java.util.ArrayList;
4
5 public class TestMemory {
6     public static ArrayList<Thread> threadList = new ArrayList<>();
7     public static int fullCount = 0;
8     public static int partCount = 0;
9
10    public static void runFullCopy() {
11        Thread thread = new Thread(new MemoryFullCopy());
12        threadList.add(thread);
13        thread.start();
14        fullCount++;
15    }
16
17    public static void runPartCopy() {
18        Thread thread = new Thread(new MemoryPartCopy());
19        threadList.add(thread);
20        thread.start();
21        partCount++;
22    }
23
24    /deprecation/
25    @Deprecated
26    public static void free(Thread thread) {
27        thread.stop();
28    }
29
30    class MemoryFullCopy implements Runnable {
31
32        private byte[] memory;
33
34        @Override
35        public void run() {
36            int size = 1024 * 1000;
```

```

38     try {
39         while(true) {
40             memory = new byte[size];
41             Thread.sleep(100);
42             memory = null;
43         }
44     } catch (InterruptedException e) {
45         e.printStackTrace();
46     }
47 }
48 }
49 }
50
51 class MemoryPartCopy implements Runnable {
52     private byte[] memory;
53
54     public void run() {
55         int size = 1024;
56
57         try {
58             while (true) {
59                 memory = new byte[size];
60                 Thread.sleep(10);
61                 memory = null;
62             }
63         } catch (InterruptedException e) {
64             e.printStackTrace();
65         }
66     }
67 }

```

## 3.2 编写 jsp 代码

init1.jsp

```

1  <%@ page import="org.tty.test4.TestMemory" %>
2  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
3  <html>
4  <head>
5      <title>TestMemory</title>
6  </head>
7  <body>
8      <%TestMemory.runFullCopy();%>
9      FullCount:<%=TestMemory.fullCount%>,PartCount:<%=TestMemory.partCount%>
10 </body>
11 </html>
12

```

init2.jsp

```

1  <%@ page import="org.tty.test4.TestMemory" %><!--
2      Created by IntelliJ IDEA.
3      User: cht
4      Date: 2021/4/14
5      Time: 15:56
6      To change this template use File | Settings | File Templates.
7  -->
8  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
9  <html>
10 <head>
11     <title>TestMemory</title>
12 </head>
13 <body>
14     <%TestMemory.runPartCopy();%>
15     FullCount:<%=TestMemory.fullCount%>,PartCount:<%=TestMemory.partCount%>
16 </body>
17 </html>

```

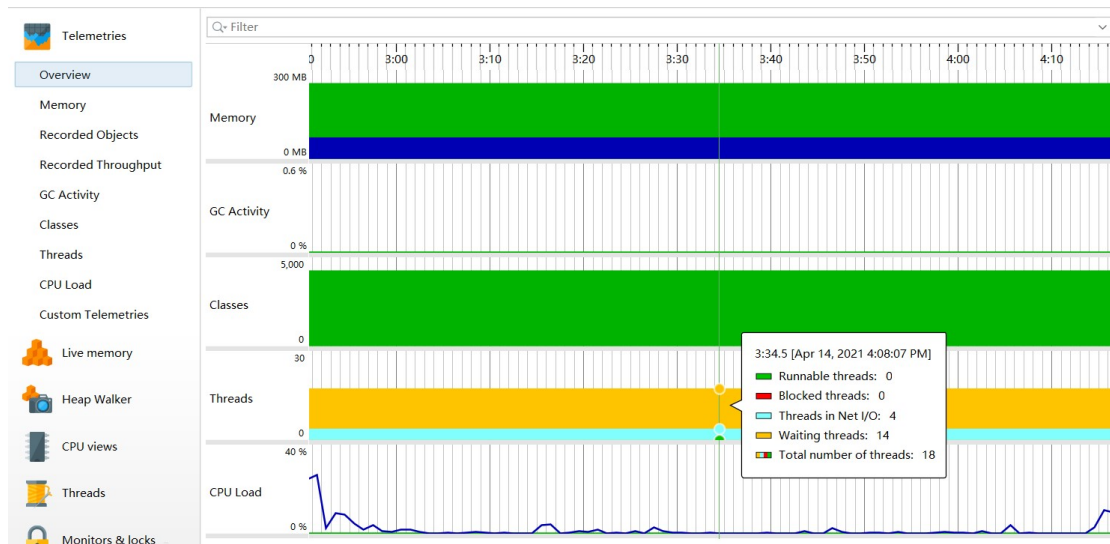
freeThread.jsp

```
1 <%@ page language="java" import="org.tty.test4.*" pageEncoding="ISO-8859-1"%>
2 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
3 <html>
4 <head>
5 <title>free</title>
6 </head>
7 <body>
8 <%
9 for(int i=0; i<TestCPU.threadList.size(); i++){
10 TestCPU.free(TestCPU.threadList.get(i));
11 }
12 TestCPU.threadList.clear();
13 for (int i =0; i < TestMemory.threadList.size(); i++) {
14 TestMemory.free(TestMemory.threadList.get(i));
15 }
16 TestMemory.threadList.clear();
17 %>
18 All threads killed! <br />
19 Thread Counter:<%=TestCPU.threadList.size()%>
20 TestMemory Thread Count:<%=TestMemory.threadList.size()%>
21 </body>
22 </html>
```

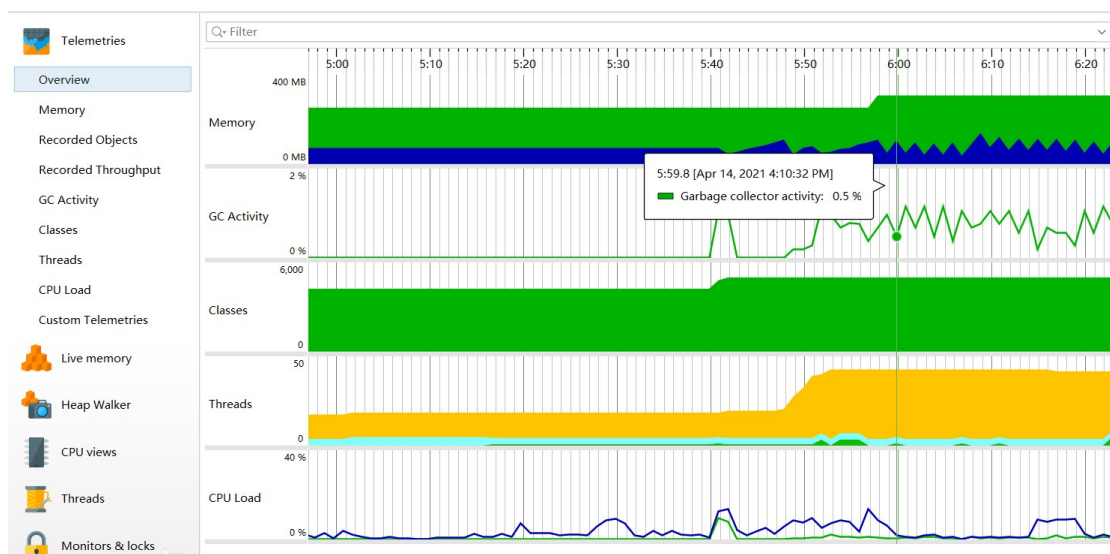
### 3.3 使用 JProfile 进行测试

我们在 IntelliJ IDEA 链接上 JProfile 的插件之后，运行内存的模式。

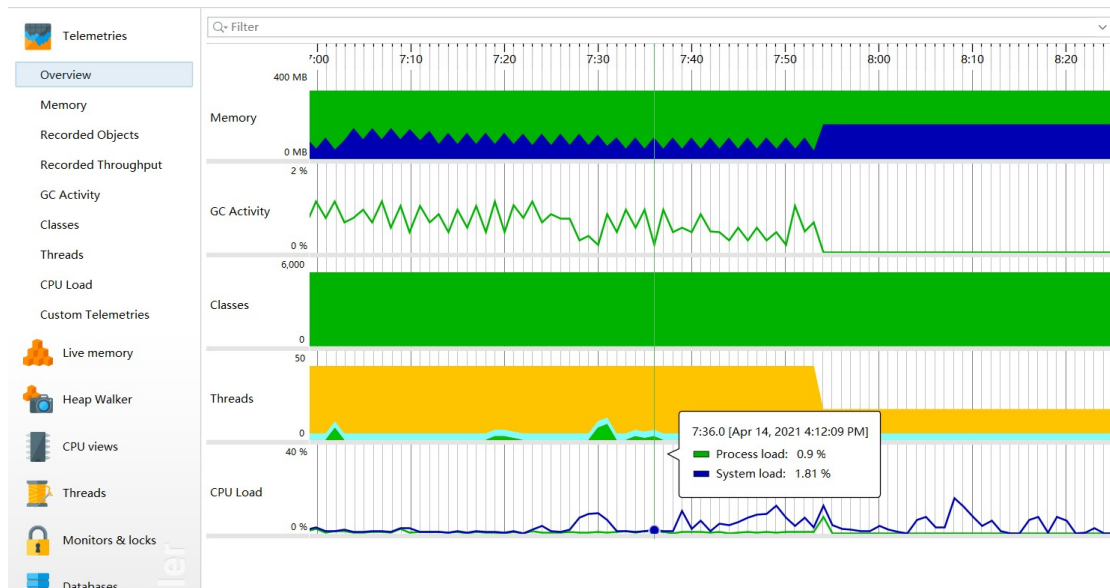
这是新建项目之后的监测情况：



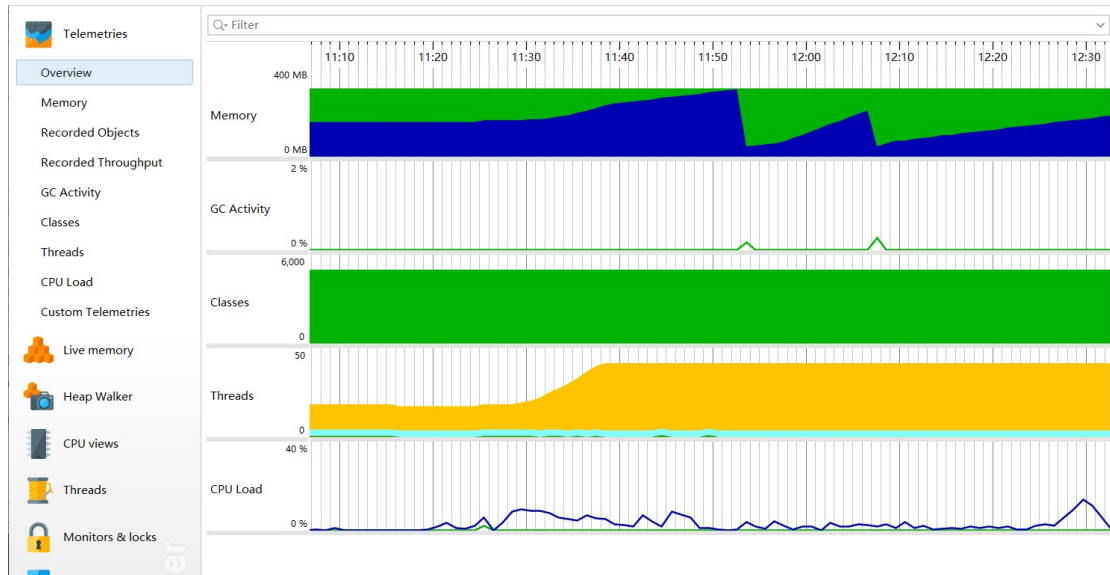
执行 25 次 init1.jsp 之后的结果。



之后执行 freeThread.jsp 清除所有线程



之后执行 25 次 init2.jsp 的结果



#### 4、总结与分析

在进行大内存区块的拷贝时，缓冲区的大小设计将会影响到 GC 的次数和 CPU 的运行效率。在实际项目中，选择缓冲区的大小将是影响效率的一个关键。在进行优化时，可以采用较大的缓冲区（例如 1MB）来降低内存分配和释放的频次从而提高效率。