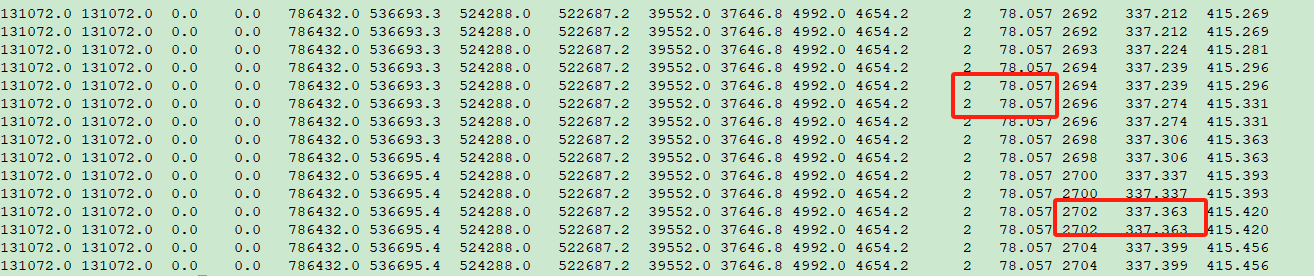
## 前言

目前在学习jvm调优，根据诸葛老师的工程jvm-full-gc做了以下修改(详情看**附录**)，最后导致大量full gc。

## 表现

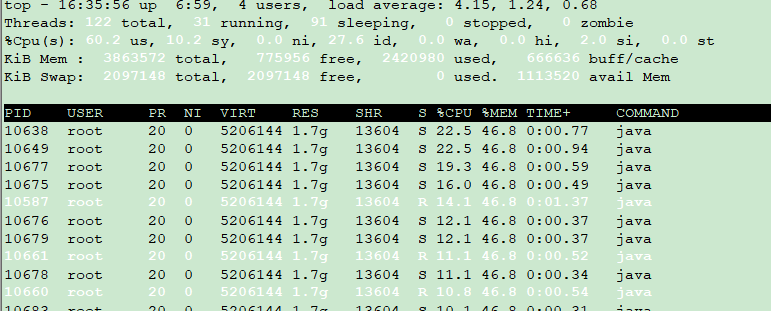
### jstat

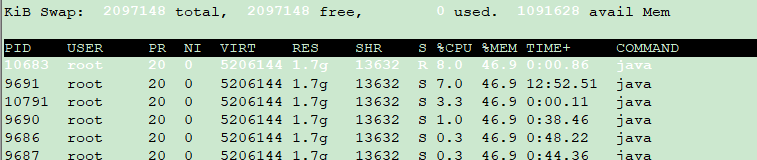
jstat -gc pid表现如下：大量full gc，少量young gc。



### **Top**

Top -p pid，然后输入大写H，查看应用内线程情况如下:

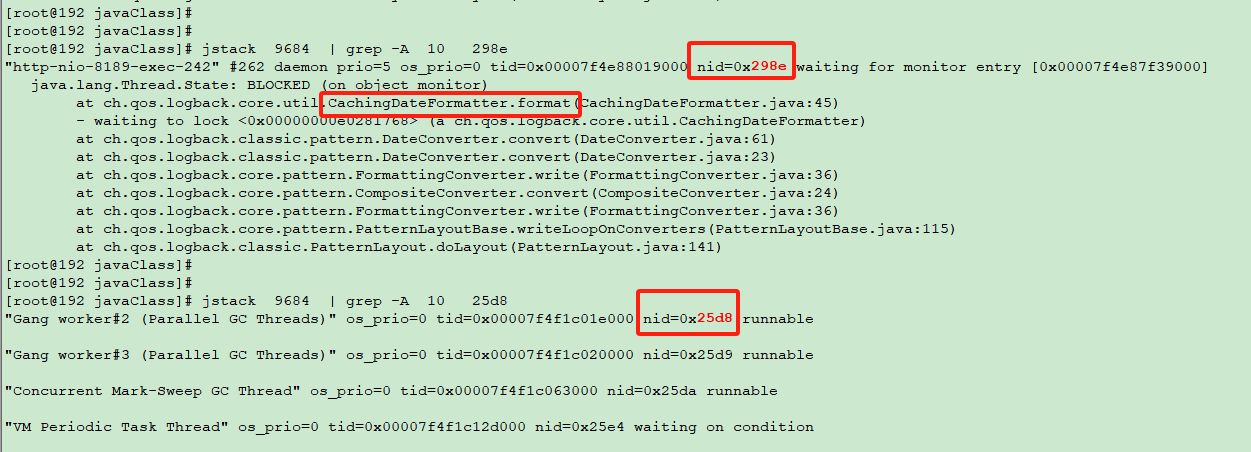




### **Jstack**

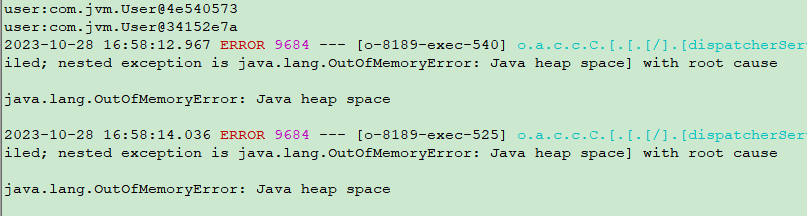
其中10638转成16进制298e ，jstack pid | grep -A 10 298e 发现是log线程；

其中 9691转成16进制25db ，jstack pid | grep -A 10 25db发现是gc线程；



### **服务器后台**

后台日志也显示 java heap space



## 问题

我该如何才能正确定位是由于**并请求processUserData()**导致的问题？

## 附录

### 服务端修改

@RequestMapping("/user/process")

public String processUserData() throws InterruptedException {

ArrayList<User> users = queryUsers();

for (User user: users) {

//TODO 业务处理

System.out.println("user:" + user.toString());

}

Thread.sleep(10000);// 新增模拟等待10S

return "end";

}

### 测试端修改

@Test

public void test() throws Exception {

for (int i = 0; i < 10000; i++) {

new Thread(()-> { //改成并发每500ms发送一次请求

String result = restTemplate.getForObject

("http://192.168.59.4:8189/user/process", String.class);

}).start();

Thread.sleep(500);

}

}