Tur.周四作业

W02Tur01

使用 GCLogAnalysis.java 自己演练一遍串行 / 并行 /CMS/G1 的案例。

打印命令：

```
public static void main(String[] args) {
        String[] gcType = new String[]{"UseSerialGC", "UseParallelGC", "UseConcMarkSweepGC", "UseG1GC"};
        String[] memSize = new String[]{"256m", "512m", "768m", "1g","1536m", "2g", "3g", "4g"};
        String commandTemplate = "java -XX:+%s -Xms%s -Xmx%s -XX:+PrintGCDateStamps -XX:+PrintGCDetails -Xloggc:./%s_%s_gclogs GCLogAnalysis";
        for (String gc: gcType) {
            for (String mem : memSize) {
                System.out.printf(commandTemplate, gc, mem, mem, gc, mem);
                System.out.println("");
            }
        }
}
```

执行结果：

生成对象次数	SerialGC	ParallelGC	CMS	G1
256m	OOM	OOM	4520	OOM
512m	11996	10103	12202	12240
768m	15500	16358	16746	16772
1g	15903	18517	16470	16215
1536m	14427	19444	15826	16398
2g	14123	19713	15448	14316
3g	12892	19577	15417	18149
4g	12493	19032	15544	19303

W02Tur02

使用了Superbenchmarker，演练 gateway-server-0.0.1-SNAPSHOT.jar 示例。

```
  90%   below 0ms
  95%   below 3ms
  98%   below 6ms
  99%   below 9ms
99.9%   below 21ms
PS C:\Users\xubuj> sb -u http://localhost:8088/api/hello -c 40 -N 60
Starting at 2020/10/28 23:20:02
[Press C to stop the test]
307535  (RPS: 4805.4)
---------------Finished!---------------
Finished at 2020/10/28 23:21:06 (took 00:01:04.0981419)
Status 200:    307535

RPS: 5035.8 (requests/second)
Max: 260ms
Min: 0ms
Avg: 0.4ms

  50%   below 0ms
  60%   below 0ms
  70%   below 0ms
  80%   below 0ms
  90%   below 0ms
  95%   below 3ms
  98%   below 6ms
  99%   below 9ms
99.9%   below 23ms
PS C:\Users\xubuj> sb -u http://localhost:8088/api/hello -c 20 -N 60
Starting at 2020/10/28 23:21:57
[Press C to stop the test]
332841  (RPS: 5201)
---------------Finished!---------------
Finished at 2020/10/28 23:23:01 (took 00:01:04.2444287)
Status 200:    332842

RPS: 5436.6 (requests/second)
Max: 219ms
Min: 0ms
Avg: 0.1ms

  50%   below 0ms
  60%   below 0ms
  70%   below 0ms
  80%   below 0ms
  90%   below 0ms
  95%   below 0ms
  98%   below 1ms
  99%   below 2ms
99.9%   below 6ms
PS C:\Users\xubuj>
```

并发数20，持续时间60秒的测试结果：

并发20持续时间60S	SerialGC	ParallelGC	CMS	G1
512m	Requests: 361332 RPS: 5903.2 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms			
1g	Requests: 345707 RPS: 5647.8 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms	Requests: 340904 RPS: 5570 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms	Requests: 349993 RPS: 5716 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms	Requests: 337148 RPS: 5524.1 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms
2g	Requests: 361953 RPS: 5913.6 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms			
4g	Requests: 330112 RPS: 5392.5 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms	Requests: 345608 RPS: 5646.6 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms	Requests: 349337 RPS: 5707.5 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms	Requests: 332842 RPS: 5436.6 90th Percentile: 0ms 95th Percentile: 0ms 99th Percentile: 2ms Average: 0.1ms

并发数40持续时间60S测试结果：

并发40持续时间60S	SerialGC	ParallelGC	CMS	G1
512m	Requests: 314895 RPS: 5155.9 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 8ms Average: 0.4ms			
1g	Requests: 307107 RPS: 5027.7 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 9ms Average: 0.4ms	Requests: 308389 RPS: 5048.6 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 8ms Average: 0.4ms	Requests: 295969 RPS: 4846.7 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 9ms Average: 0.4ms	Requests: 305285 RPS: 4997.8 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 9ms Average: 0.4ms
2g	Requests: 321863 RPS: 5266.6 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 8ms Average: 0.4ms			
4g	Requests: 312189 RPS: 5110.1 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 8ms Average: 0.4ms	Requests: 308389 RPS: 5048.6 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 8ms Average: 0.4ms	Requests: 290953 RPS: 4762.1 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 10ms Average: 0.5ms	Requests: 307535 RPS: 5035.8 90th Percentile: 0ms 95th Percentile: 3ms 99th Percentile: 9ms Average: 0.4ms

W02Tur03

...

W02Tur04

周四01、02作业总结：

当回收较为频繁时，串行回收器在最大堆内存较小时，创建回收对象效率较高；当最大堆内存较大时，串行回收器回收慢，影响整体使用效率。

并行回收器在最大堆内存较小时，并行效率并不比串行明显高；但随着最大堆内存设置变大，并行回收效率显著提升，并较为稳定。

CMS回收器随着最大堆内存变大，基本创建效率也趋于稳定，但稳定值明显与并行回收器有差距，可见为了减少用户线程停顿，而牺牲了一定回收效率。

G1回收器的效率随着最大堆内存的提升一直在提升，可见在内存较大时，G1回收器是首要考虑配置。

在作业02的并发测试中，可能因为测试方法较为简单，没有创建对象的操作，各回收器的性能差距并不明显。仅可以观察到，并发数提升后，增加最大堆内存并不能提升对并发任务的处理速度。

Sat.周六作业

W02Sat01

...

W02Sat02

见Homework_Sat02。

```
public class HttpClientDemo {
    public static void main(String[] args) {
        try (CloseableHttpClient httpclient = HttpClients.createDefault()) {
            HttpGet httpGet = new HttpGet("http://localhost:8801");

            try (CloseableHttpResponse response =
httpclient.execute(httpGet)) {
                System.out.println(response.getCode() + " " +
response.getReasonPhrase());
                HttpEntity entity1 = response.getEntity();
                EntityUtils.consume(entity1);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```