

系统性能测试方案以及测试结果

测试环境

硬件配置：4核 16G 2020款MacBook Pro

操作系统: MacOS



JDK版本: JDK8

测试工具

jmeter作为压力测试工具 jconsole作为JVM运行监控工具

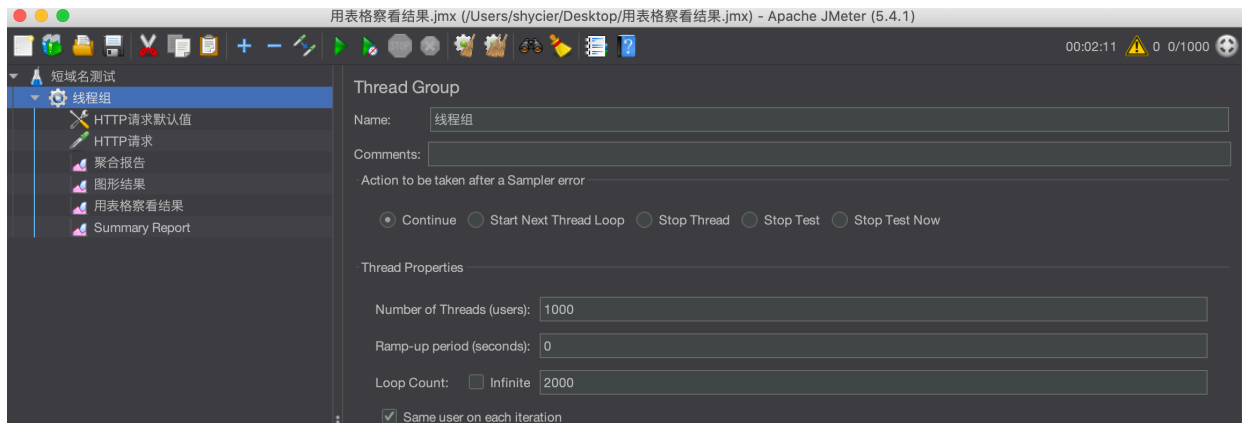
测试方案

启动服务，设置JVM最大内存1GB，其他参数默认，设置KV淘汰的maxMemory为1000MB略小于JVM最大内存，maxSize为-1不按照数量淘汰数据。

测试一 线程数1000 循环次数 2000 共200万个KV

A screenshot of the JMeter Aggregate Report window. The window has a dark theme. On the left is a sidebar with a tree view showing the test plan structure: '短域名测试' (Subdomain Test) -> '线程组' (Thread Group) -> 'HTTP请求默认值' (HTTP Request Defaults) -> 'HTTP请求' (HTTP Request) -> '聚合报告' (Aggregate Report) -> '图形结果' (Graph Results) -> '用表格察看结果' (View Results in Table) -> 'Summary Report'. The '聚合报告' (Aggregate Report) is selected. The main area shows the 'Aggregate Report' for the selected test. It includes fields for 'Name' (聚合报告), 'Comments', and 'Write results to file / Read from file' (Filename). There are buttons for 'Browse...', 'Log/Display Only:', 'Errors', 'Successes', and 'Configure'. Below these is a table with the following data:

Label	# Sampl...	Average	Median	90% ... ↑	95% Line	99% Line	Min	Maximum	Error %	Throug...	Receive...	Sent KB...
HTTP请...	2100000	61	56	92	122	255	0	947	0.00%	2639.8/...	437.07	605.81
TOTAL	2100000	61	56	92	122	255	0	947	0.00%	2639.8/...	437.07	605.81



如上图平均响应时间61ms,因为此阶段处于数据填充阶段，老年代GC不多，1000个并发的情况下也表现尚可

测试二 线程数1000 循环次数 2000 共200万个KV

Summary Report

Name:

Summary Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only:

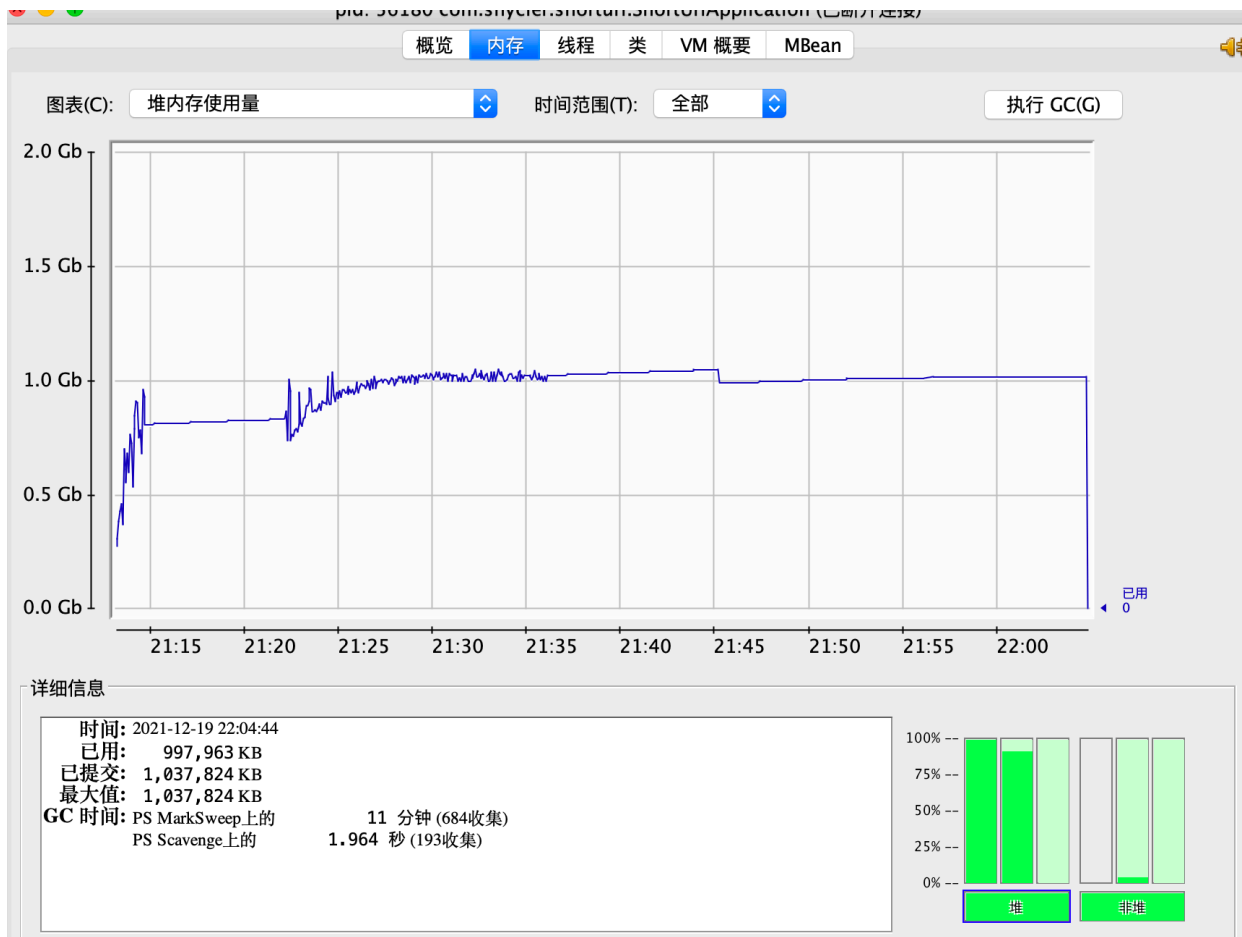
☐ Errors

☐ Successes

Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/sec	Avg. Bytes
HTTP请求	2000000	402	0	8966	512.25	0.00%	2410.1/sec	399.61	553.09	169.8
TOTAL	2000000	402	0	8966	512.25	0.00%	2410.1/sec	399.61	553.09	169.8

如上图，和测试一不同的是此时内存已经接近上限，虽然有缓存淘汰不至于OOM，但是频繁的FullGC导致平均延时达到400ms



从上面jconsole截图可以看到21:15分之前这段是第一个200万填充阶段，很快就完成了。21:20~21:35这段是第二个200数据稳定阶段，延时比较高，从下面PS MarkSweep也可以看出FullGC 684次共消耗11分钟。虽然没有OOM但是这样的延迟是不能接受的。

分析：应该是缓存淘汰的MaxMemory与JVM的Xmx设置的太接近导致FullGC频繁，这里将Xmx保持1GB，将maxMemory设置为800M(80%) 重启服务。

测试三 线程数1000 循环次数 2000 共200万个KV

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

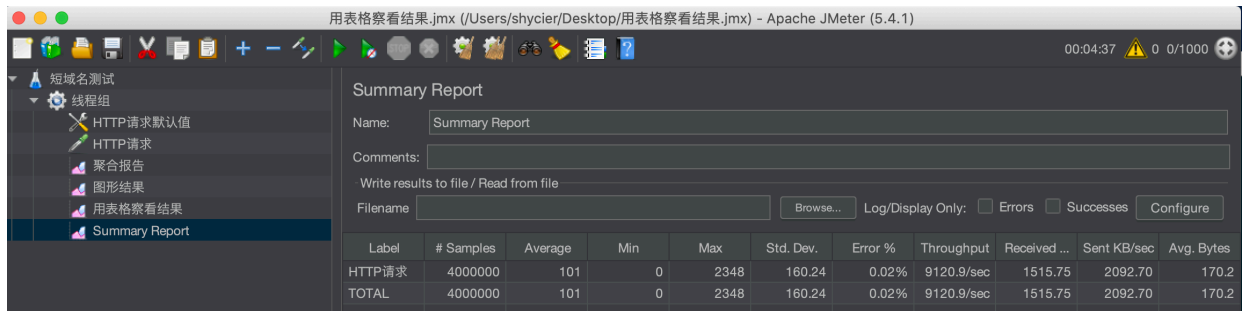
Filename: Browse...

Log/Display Only: ☐ Errors ☐ Successes

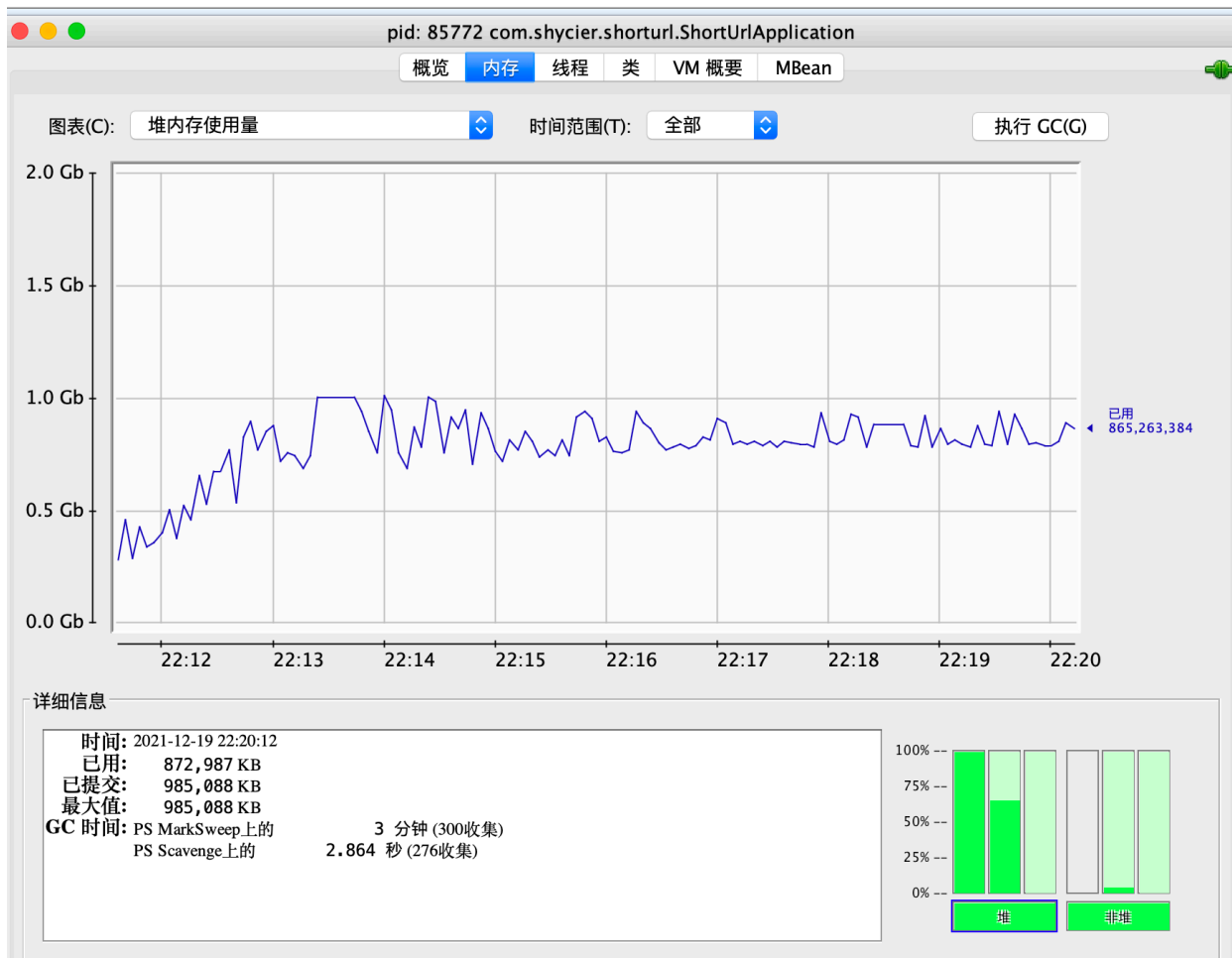
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/sec	Avg. Bytes
HTTP请求	2000000	68	0	2348	54.35	0.04%	14319.9/s...	2384.08	3285.01	170.5
TOTAL	2000000	68	0	2348	54.35	0.04%	14319.9/s...	2384.08	3285.01	170.5

这里重新填充数据，效果与测试一一致

测试四 线程数1000 循环次数 2000 共200万个KV

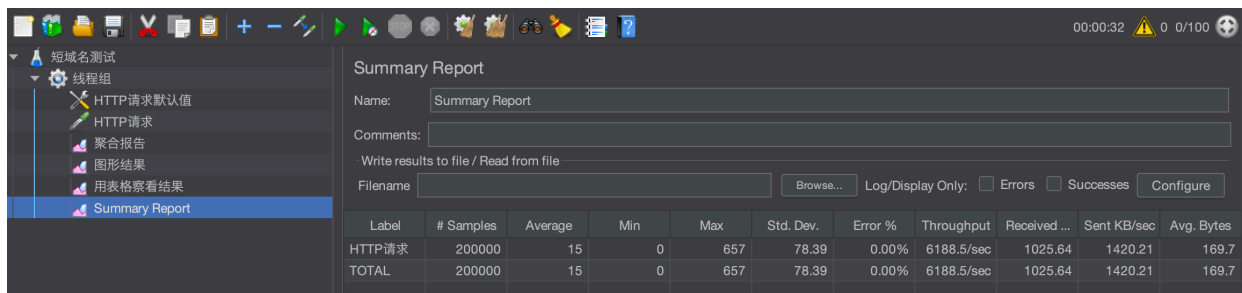


如上图可以看到平均延时101比之前的400好了很多



FullGC从之前的11分钟降到了3分钟，次数也减少了一倍。

tomcat的默认线程数是200,单个服务能扛住1000并发且延迟可以控制在100以内应该算达标了，下面附上100/250/500并发下的测试数据。



!截屏2021-12-19 下午10.32.49[/Users/shycier/Desktop/截屏2021-12-19 下午10.32.49.png)

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/sec	Avg. Bytes
HTTP请求	172668	40	0	743	123.07	0.00%	6182.8/sec	1024.83	1418.91	169.7
TOTAL	172668	40	0	743	123.07	0.00%	6182.8/sec	1024.83	1418.91	169.7

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received ...	Sent KB/sec	Avg. Bytes
HTTP请求	334129	82	0	1336	174.09	0.00%	5989.7/sec	992.82	1374.59	169.7
TOTAL	334129	82	0	1336	174.09	0.00%	5989.7/sec	992.82	1374.59	169.7

测试结论

结论一：该服务通过缓存淘汰策略可以避免OOM的情况产生。

结论二：服务在数据空载的情况下1000个并发填充200万数据，耗时2分钟，平均延时60ms。

结论三: 服务在数据满载的情况下100个并发，平均延时15ms。

结论三: 服务在数据满载的情况下250个并发，平均延时40ms。

结论三: 服务在数据满载的情况下500个并发，平均延时80ms。

结论三: 服务在数据满载的情况下1000个并发，平均延时100ms。