JMeterPerformance

_ _

JMeter Performance evolution across versions (>= 2.5.1)

This article is the record of a Test done in the following conditions:

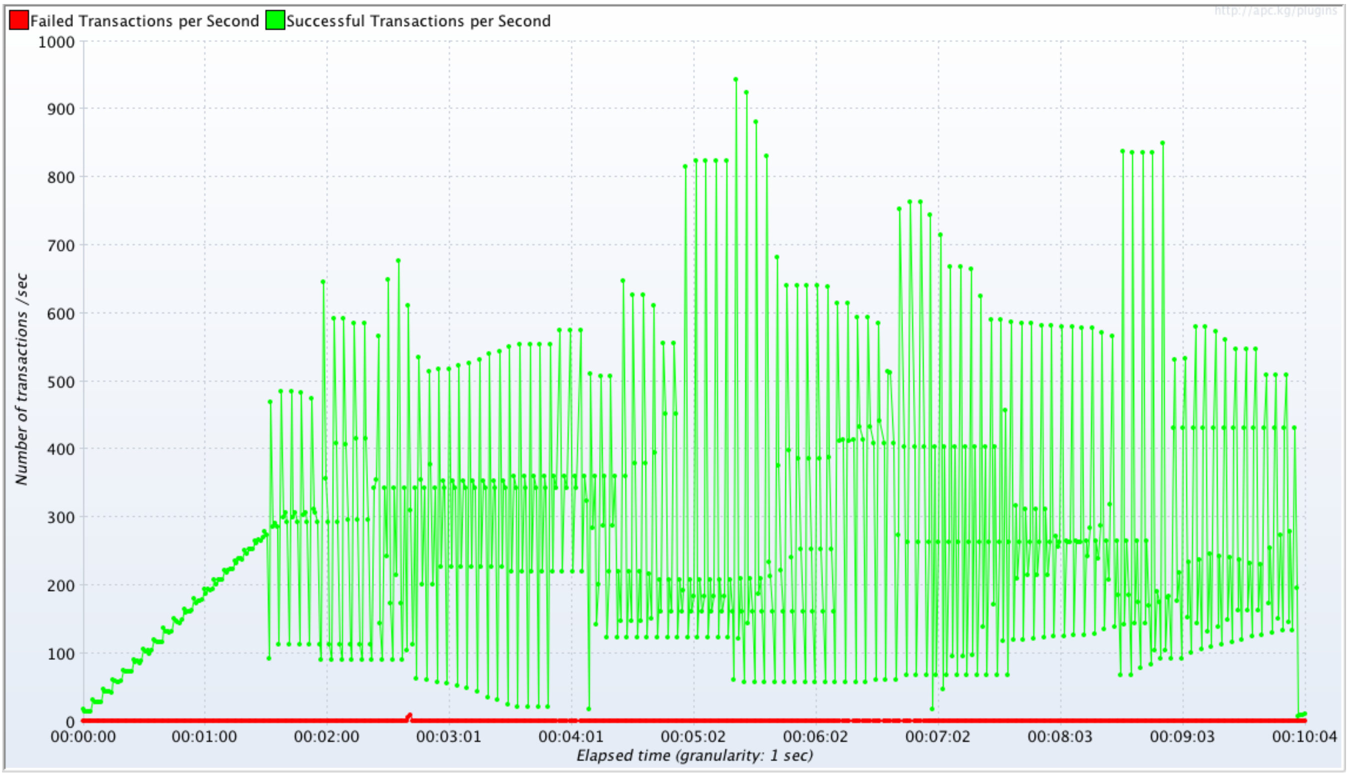
- Tomcat version 6.0.24
- Tomcat JVM : -Xms256m -Xmx1024m
- JMeter JVM : -Xmx512m
- Set session timeout in web.xml to 1 minute
- Java version "1.6.0_29"
- Java(TM) SE Runtime Environment (build 1.6.0_29-b11-402-10M3527)
- Java HotSpot(TM) 64-Bit Server VM (build 20.4-b02-402, mixed mode)
- Mac OS 10.6.8
- JMeter and Tomcat on same machine
- No particular OS Tuning
- RAM : 8 Go 1067 MHz DDR3
- Processor : 3.06 GHz Intel Core 2 Duo
- No swap
- Simple Test plan using Tomcat examples
- Test Plan:
    - Ramp-up period : 100 seconds
    - Number of Threads : 1500
    - Scheduler checked:
        - Duration: 10 minutes (So test will run just 10 minutes forcing stop at 10th minute)
        - Startup Delay : 7 seconds
- CSV output

Test Plan:

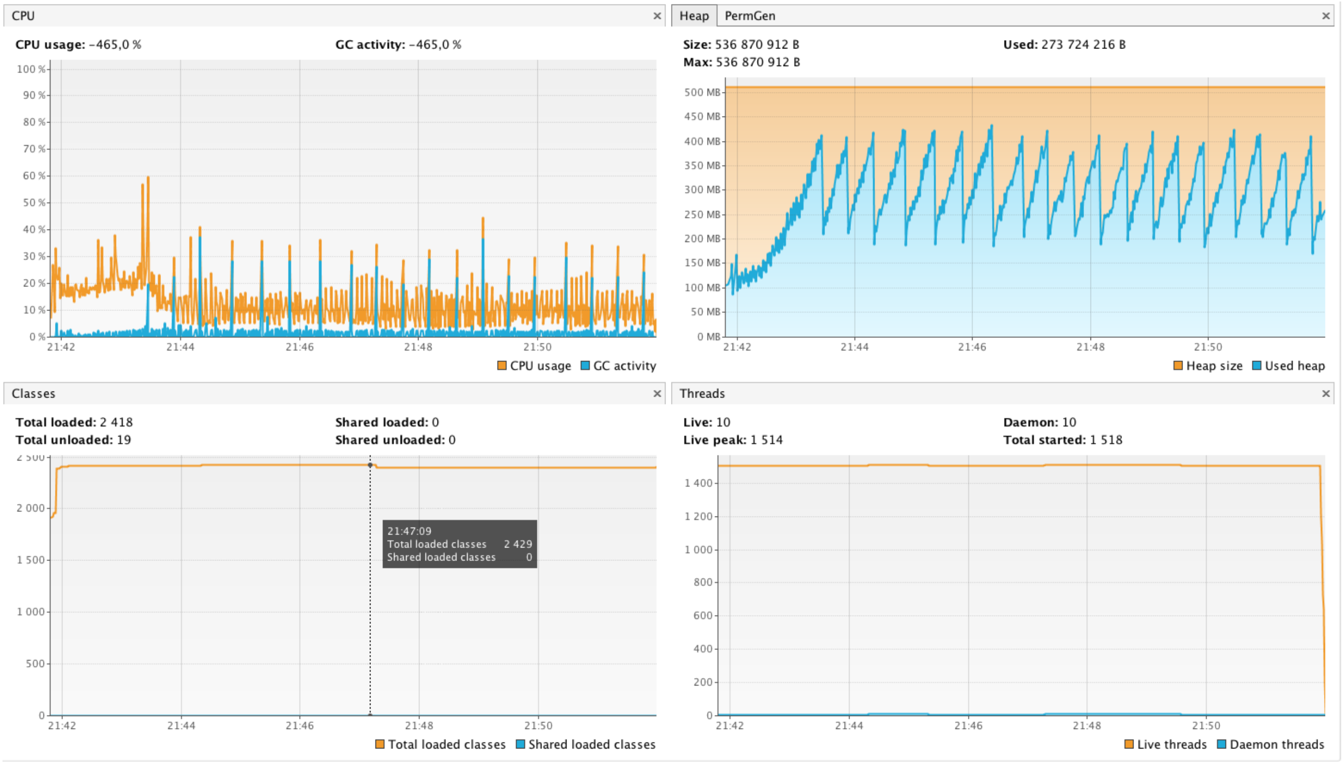
Test Plan used : TestPlan.jmx

JMETER 2.5.1

Transactions:



JVM Behaviour:



CPU

CPU usage: –465,0 % **GC activity:** –465,0 %

Heap | PermGen

Size: 536 870 912 B **Used:** 273 724 216 B
Max: 536 870 912 B

Classes

Total loaded: 2 418 **Shared loaded:** 0
Total unloaded: 19 **Shared unloaded:** 0

Threads

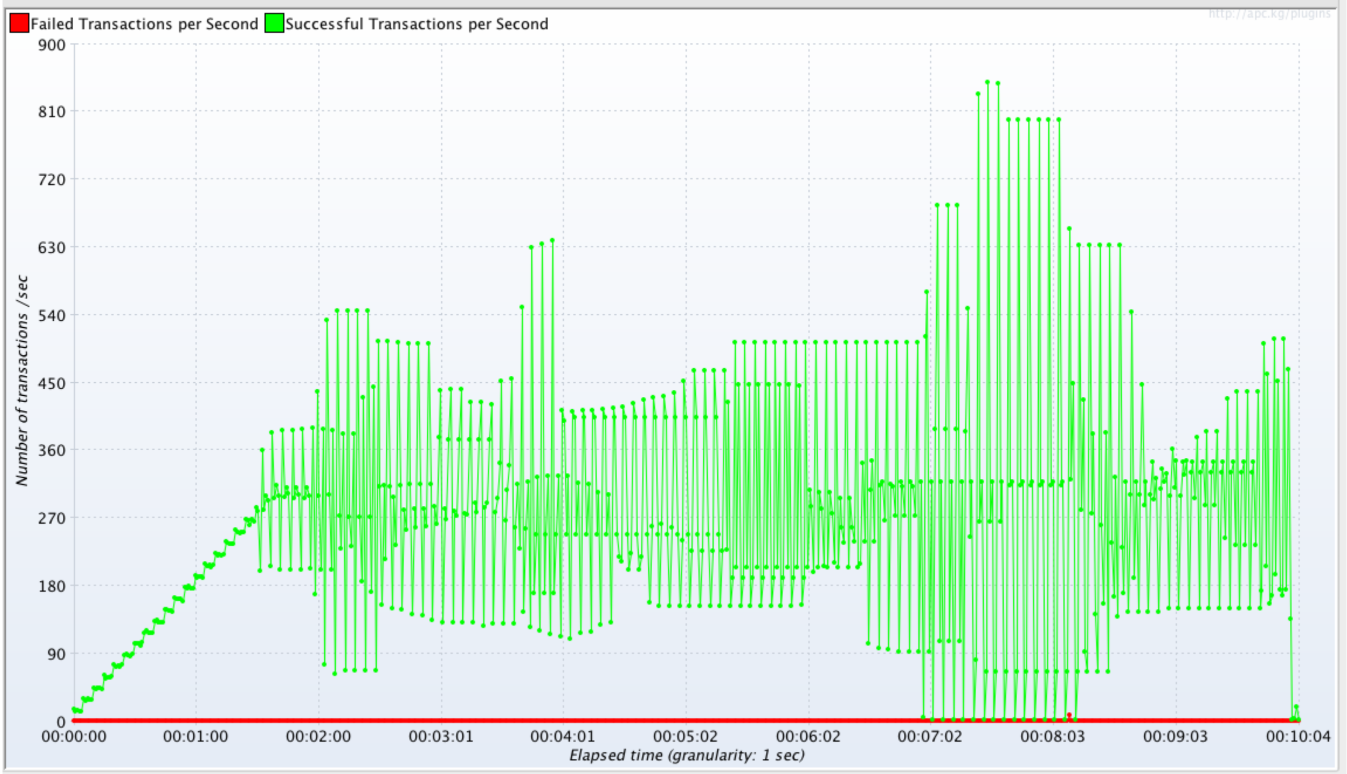
Live: 10 **Daemon:** 10
Live peak: 1 514 **Total started:** 1 518

GC Activity:

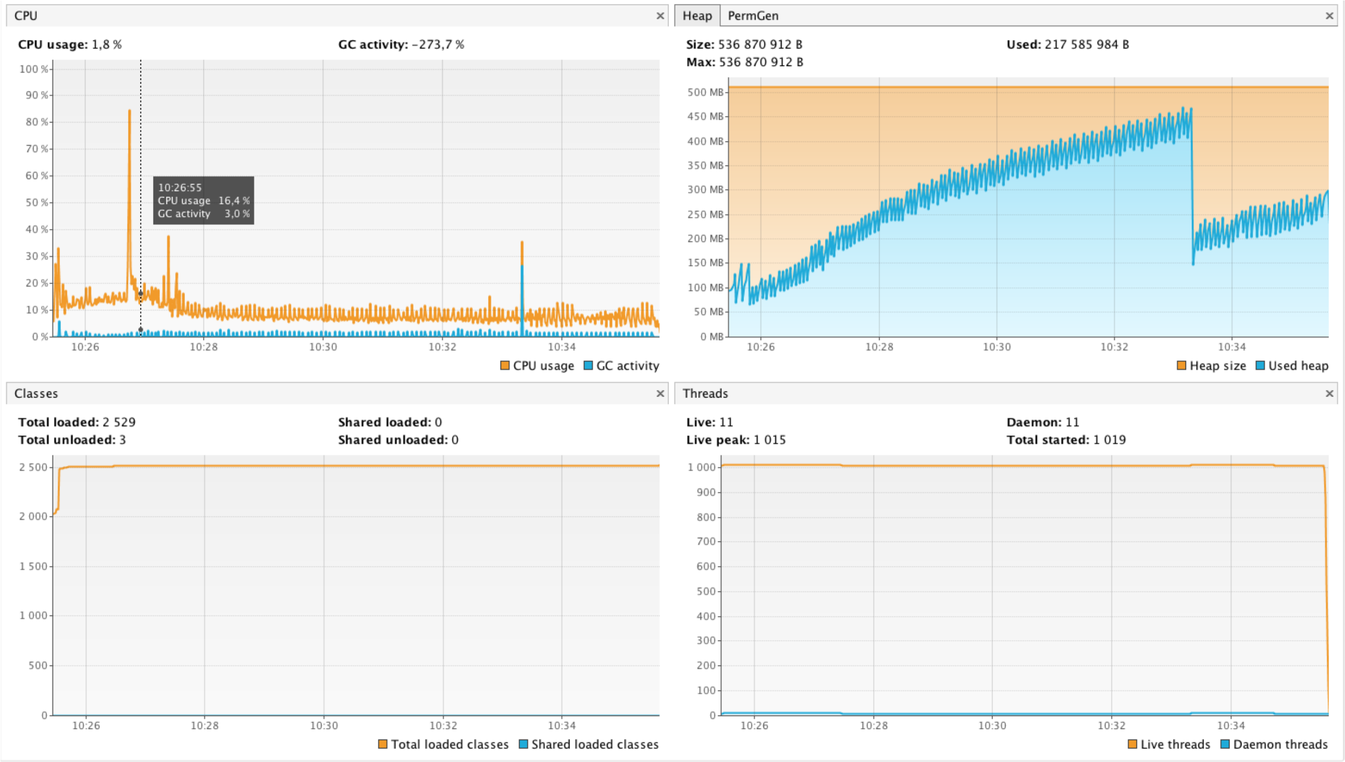
- GC activity is much higher with around 5 GC CPU peaks every 2 minutes
- and 20 FULL GC of 700 to 800 ms each
- Throughput: 97,71%
- Pauses : 13,69s
- Memory Cleaned : 391m/min
- Full GC tend to be much more frequent at end of test

JMETER 2.6:

Transactions:



JVM Behaviour:



CPU

CPU usage: 1,8 % **GC activity:** -273,7 %

10:26:55
CPU usage 16,4 %
GC activity 3,0 %

Heap PermGen

Size: 536 870 912 B **Used:** 217 585 984 B
Max: 536 870 912 B

Classes

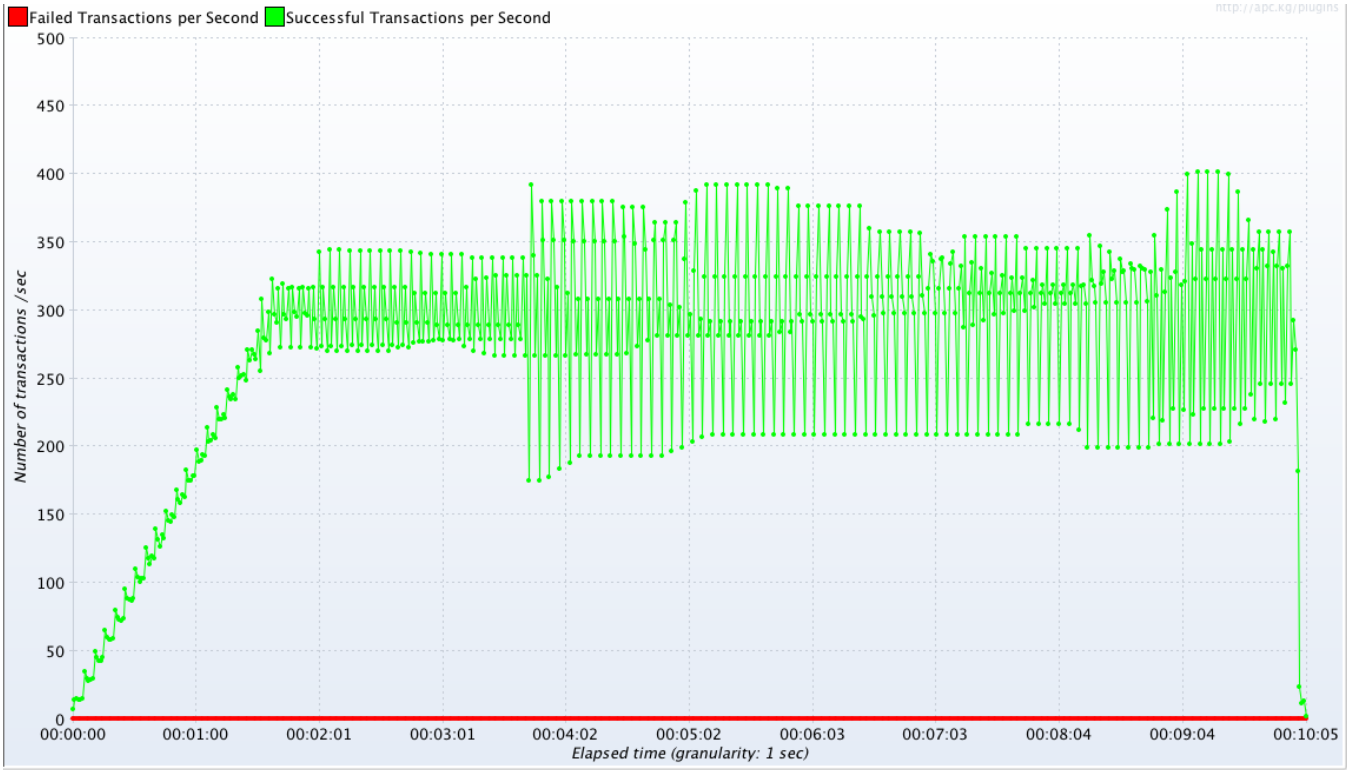
Total loaded: 2 529 **Shared loaded:** 0
Total unloaded: 3 **Shared unloaded:** 0

Threads

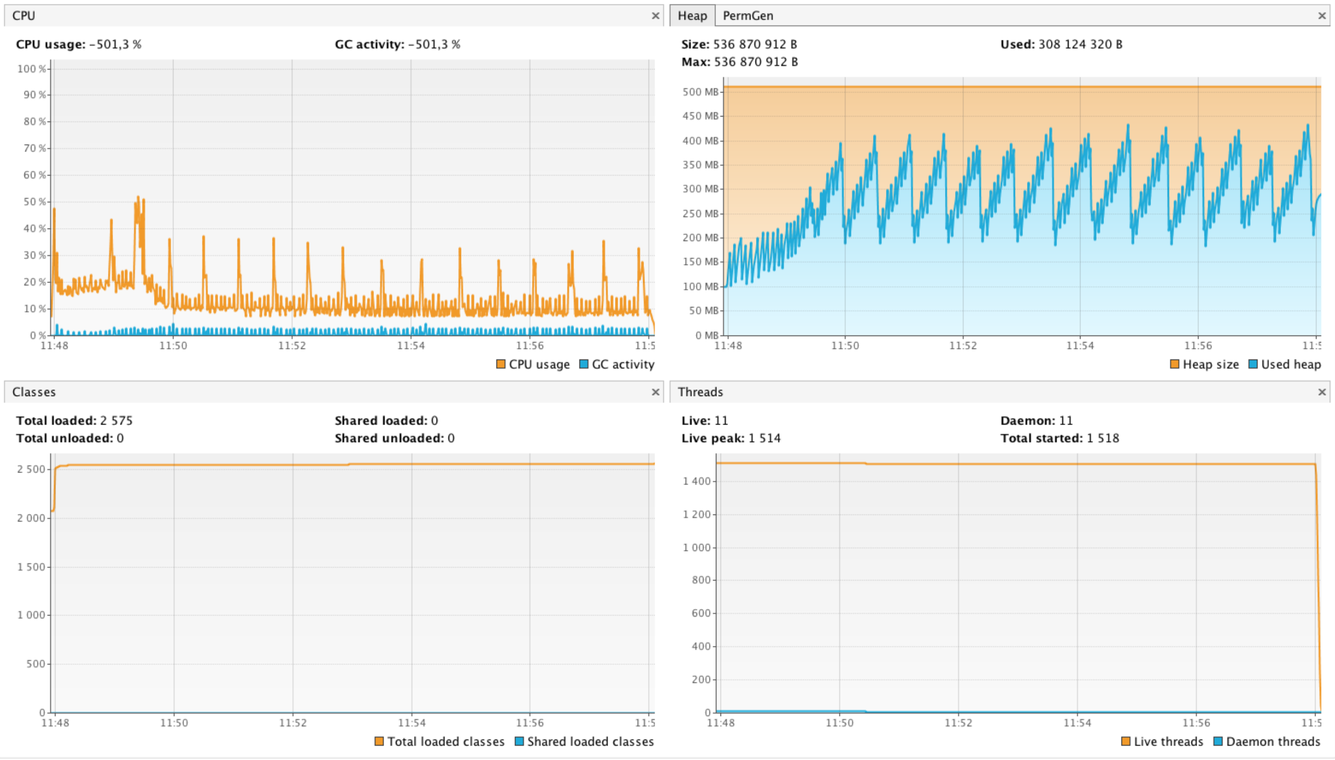
Live: 11 **Daemon:** 11
Live peak: 1 015 **Total started:** 1 019

JMETER 2.7:

Transactions:



JVM Behaviour:



CPU usage: –501,3 %

GC activity: –501,3 %

Size: 536 870 912 B

Max: 536 870 912 B

Used: 308 124 320 B

Total loaded: 2 575

Total unloaded: 0

Shared loaded: 0

Shared unloaded: 0

Live: 11

Live peak: 1 514

Daemon: 11

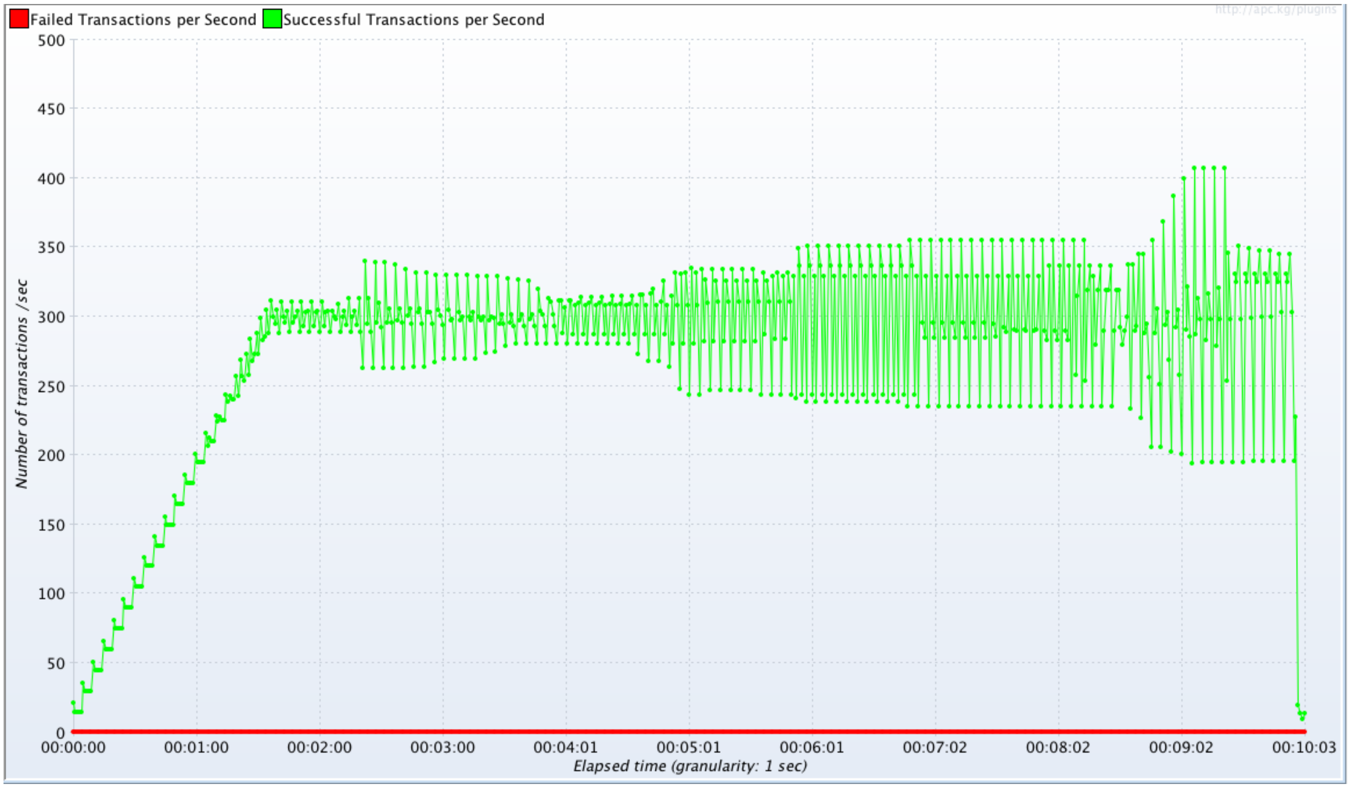
Total started: 1 518

GC Activity:

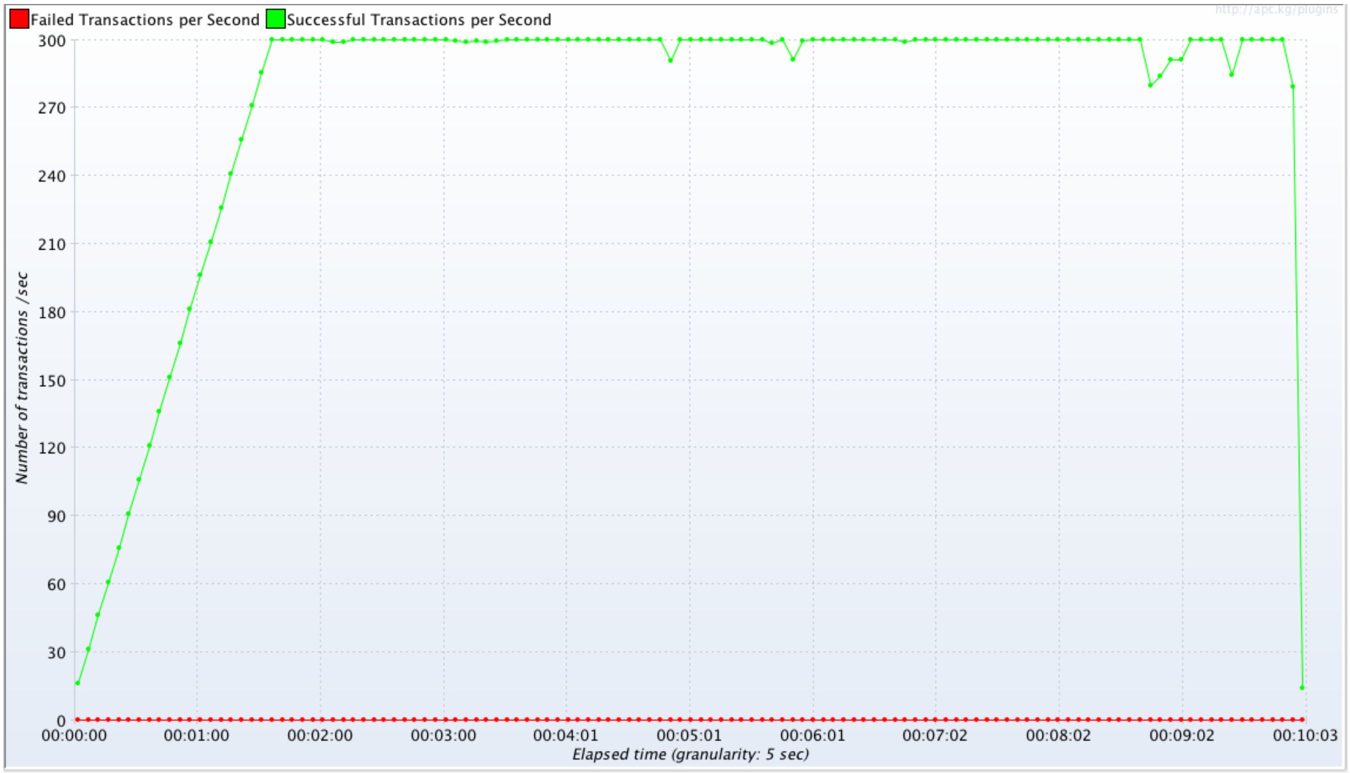
- No GC CPU peak,
- 1 FULL GC
- Throughput:98.54
- Pauses : 8.9s
- Memory cleaned: 1108m /min

JMETER 2.8:

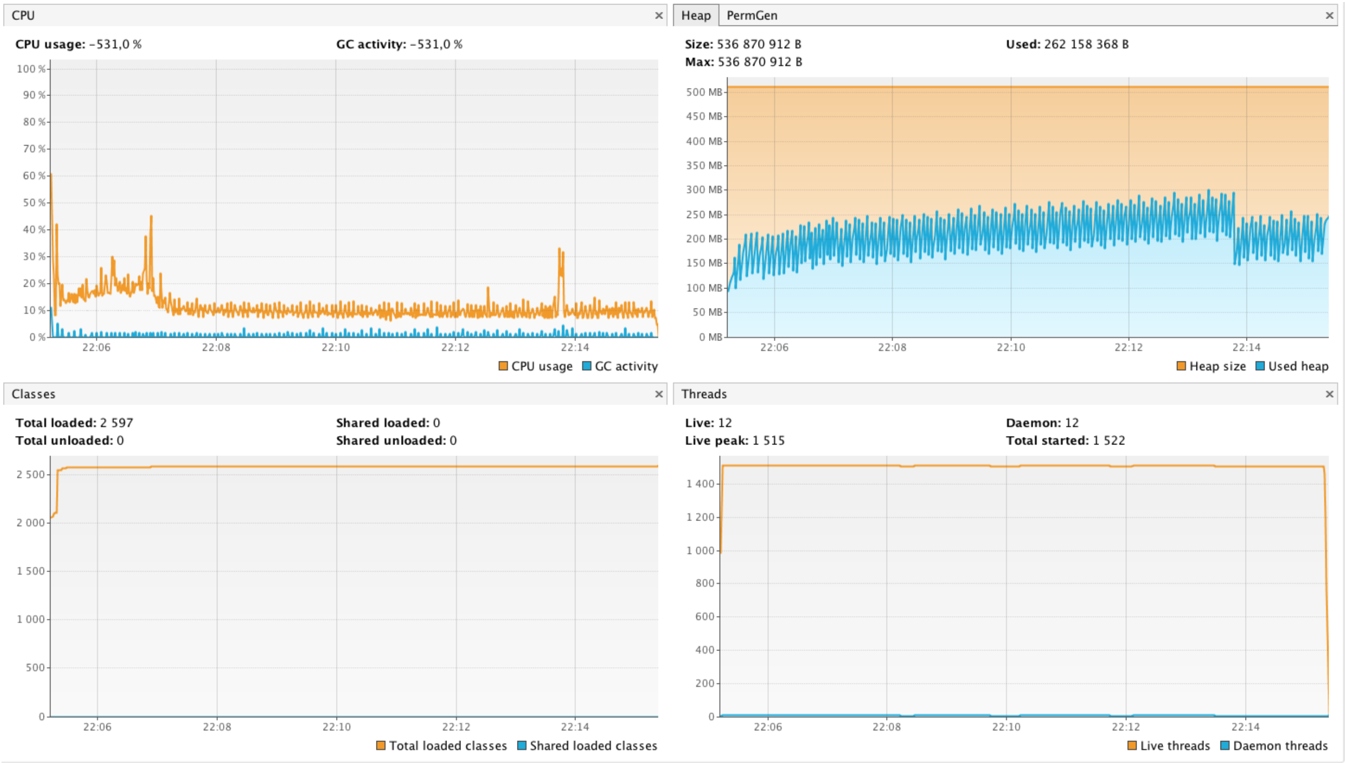
Transactions:



Graph Limiting 150 points in row:



JVM Behaviour:



CPU usage: ~531,0 %
GC activity: ~531,0 %

Size: 536 870 912 B
Max: 536 870 912 B
Used: 262 158 368 B

Total loaded: 2 597
Total unloaded: 0
Shared loaded: 0
Shared unloaded: 0

Live: 12
Live peak: 1 515
Daemon: 12
Total started: 1 522

GC Activity:

- No GC CPU peak,
- No FULL GC
- Throughput:98.89%
- Pauses : 6.72s
- Memory cleaned: 1391m /min

Conclusion

- No significant improvement between 2.6 and 2.5.1
- Significant improvement between 2.7 and 2.6
- Significant improvement between 2.8 and 2.7
- Better memory behaviour
- More accurate response times with High Load

JMeter 2.8 Additional Test (to check time taken is fine)

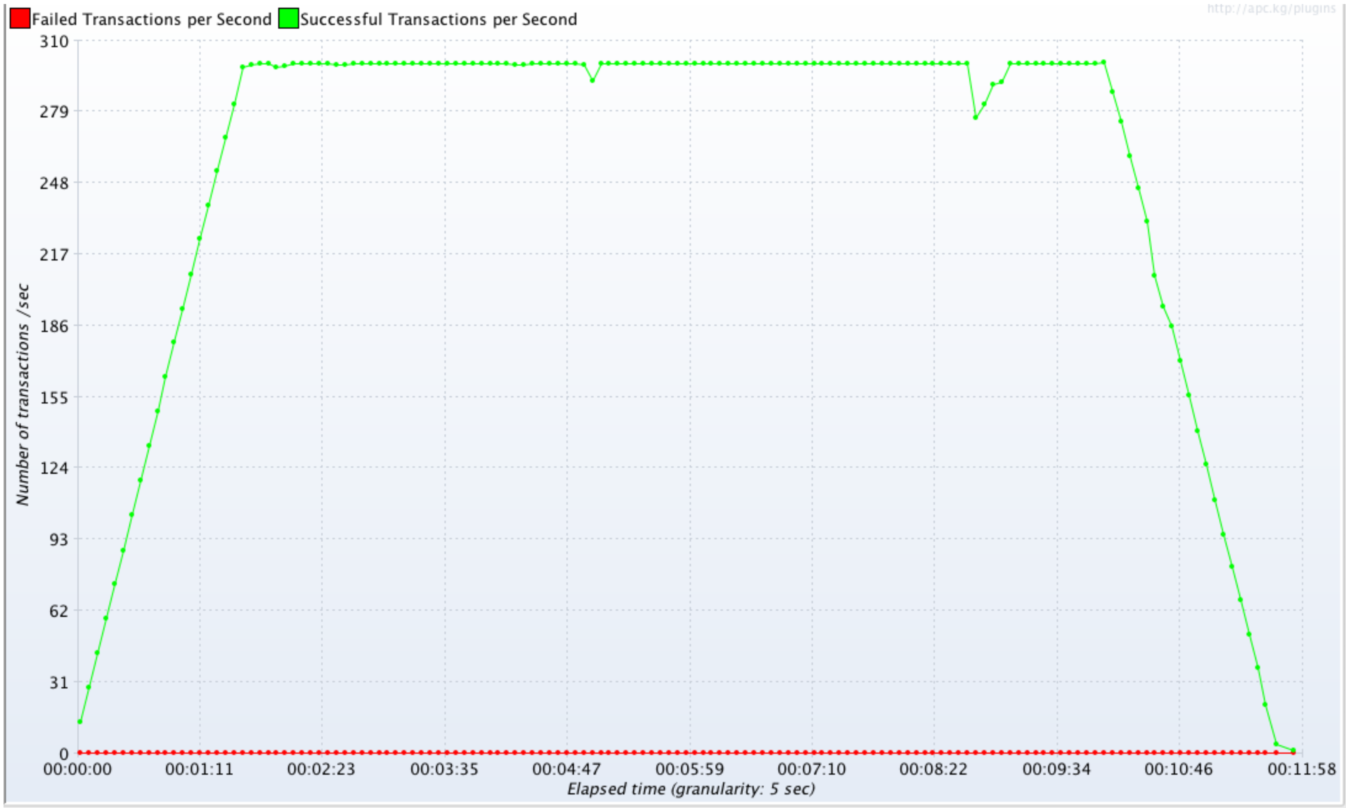
- In this test, to let threads complete, we increase Test Duration to check no delay happens which can reveal abnormal behaviour.
- Test Plan same as before with following changes:
  - Ramp-up period : 100 seconds
  - Number of Threads : 1500
  - Scheduler checked:
    - Duration: 800 seconds (So all threads should complete before that time)
    - Startup Delay : 7 seconds

Test Plan for additional Test:

Test Plan used :

PerformancePlan-2.8.jmx

Transactions:



Conclusion

No delay, everything is OK.