项目性能测试报告

01-测试目的

搭建小型的 spring-boot 项目 coffee,通过对项目的性能测试，了解项目的性能瓶颈，通过不同的优化手段，使得项目最终的 tps 达到一个可接受的程度。

02-测试工具

Jmeter5.4.1+prometheus2.15.1+fluxDB

03-测试环境

指标	参数
机器	4C8G
集群规模	单机
数据库	4C8G

04-测试接口

模拟高延时场景，

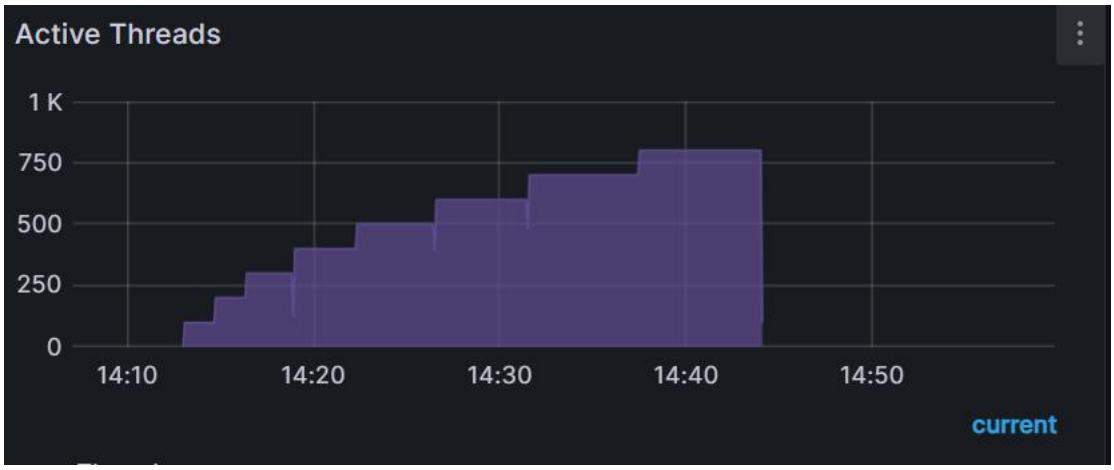
获取 coffee 商品信息接口，http://8.137.37.26:8080/slowShowCoffee?id=10

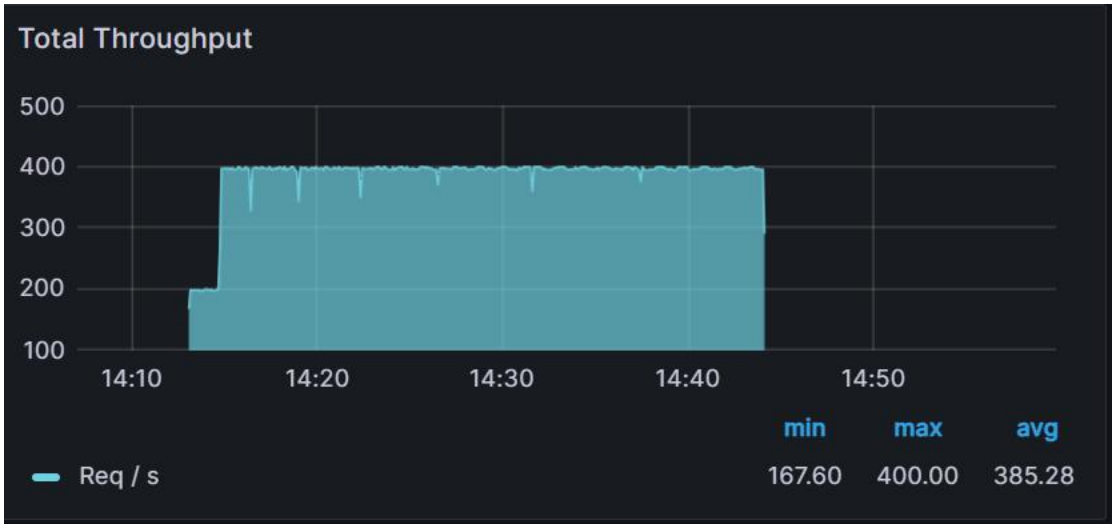
接口响应时间为 550ms,线程梯度:100、200、300、400、500、600、700、800 个线程，200 次，Ramp-up 为 1s.

05-测试结果

验证 coffee 服务获取商品信息接口能力性能。

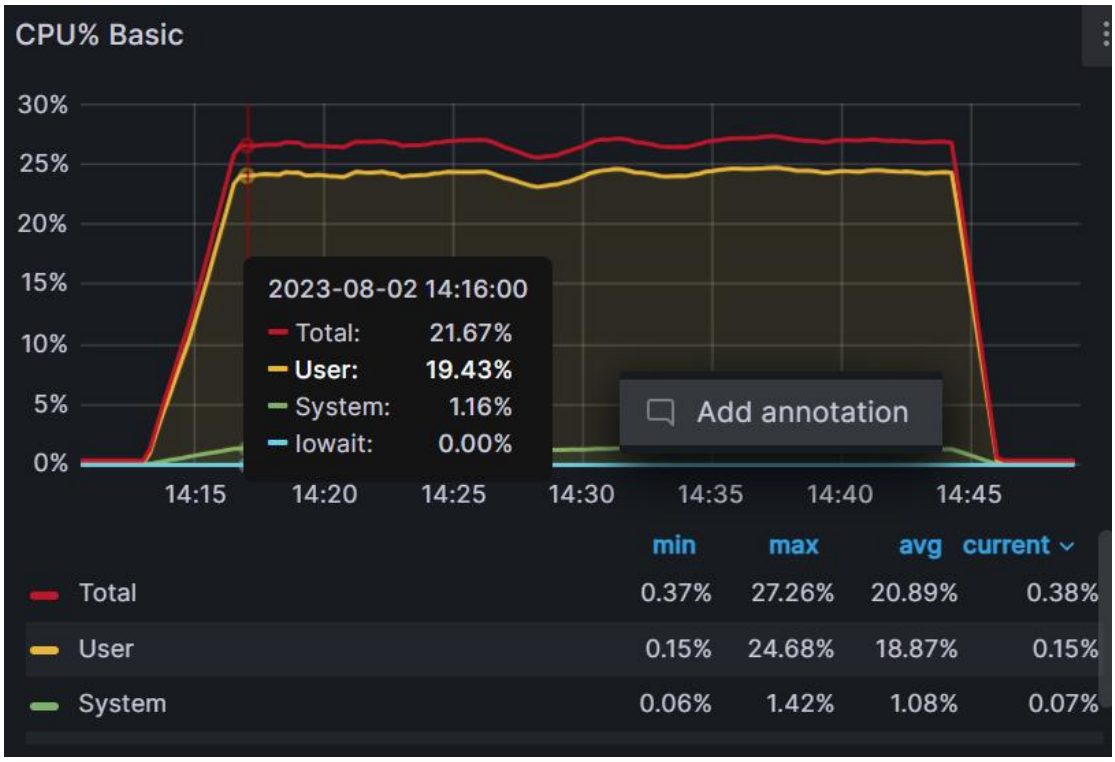
ActiveThread、TPS、RT

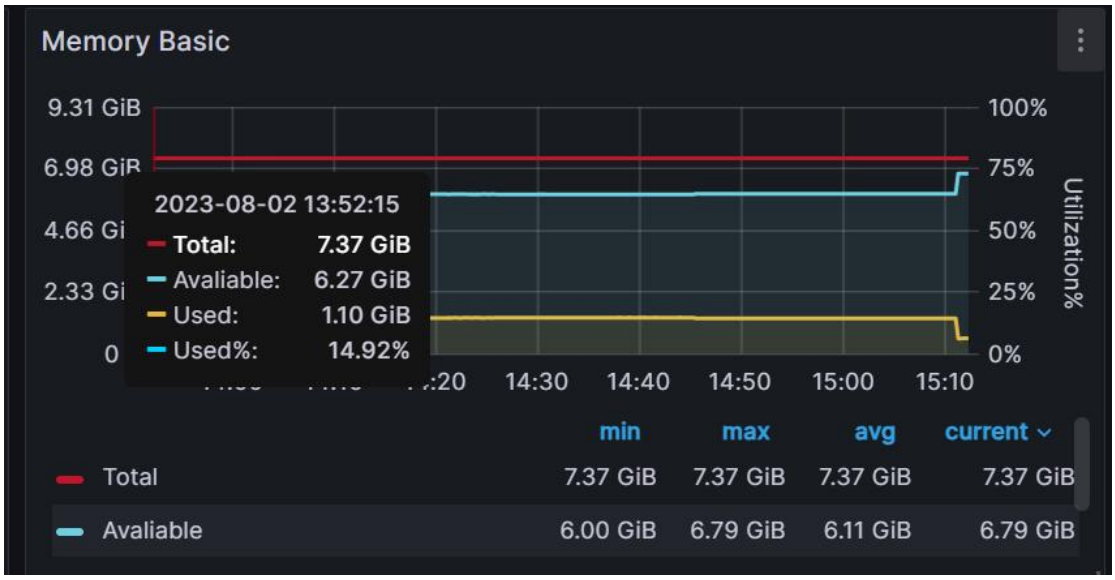

Active Threads
1 K
750
500
250
0
14:10
14:20
14:30
14:40
14:50
current

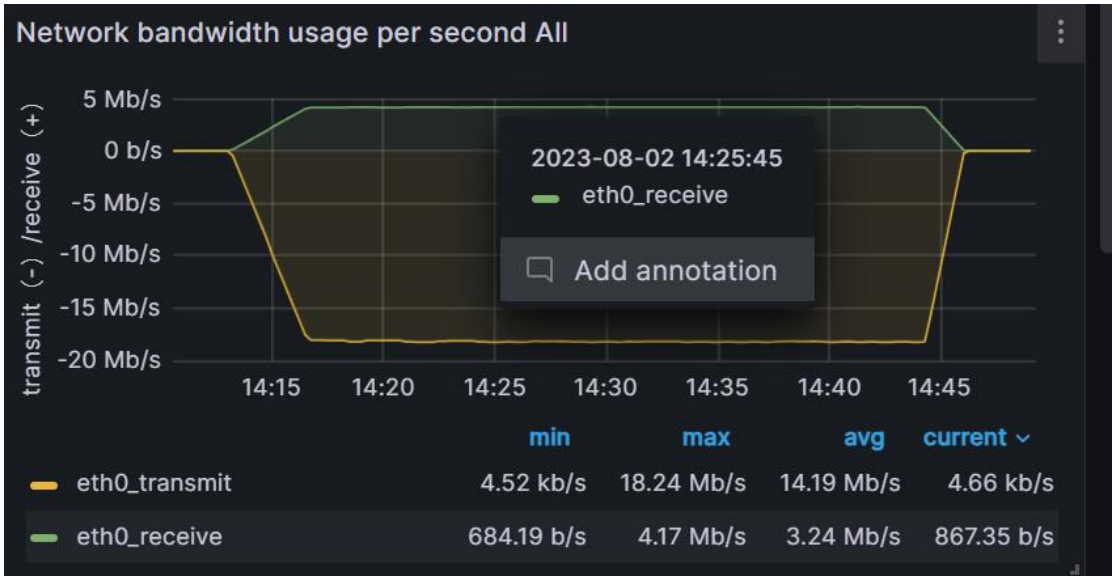

Total Throughput
500
400
300
200
100
14:10
14:20
14:30
14:40
14:50
min
max
avg
Req / s
167.60
400.00
385.28


Transactions Response Times (99th pct)
2.50 s
2 s
1.50 s
1 s
500 ms
0 ms
2023-08-02 13:59:45
300Thread-HTTP请求 815.00 ms 785.70 ms
400Thread-HTTP请求 1.09 s 1.02 s
500Thread-HTTP请求 1.37 s 1.32 s
600Thread-HTTP请求 1.69 s 1.51 s
700Thread-HTTP请求 1.90 s 1.83 s
800Thread-HTTP请求 2.19 s 2.03 s
all 2.19 s 1.44 s

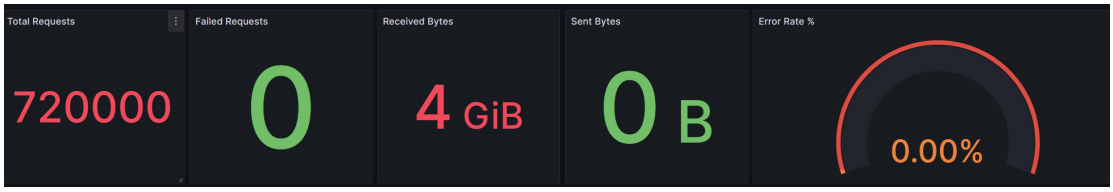
cpu、内存、网络


CPU% Basic
30%
25%
20%
15%
10%
5%
0%
2023-08-02 14:16:00
Total: 21.67%
User: 19.43%
System: 1.16%
Iowait: 0.00%
Add annotation
14:15
14:20
14:25
14:30
14:35
14:40
14:45
min max avg current
Total 0.37% 27.26% 20.89% 0.38%
User 0.15% 24.68% 18.87% 0.15%
System 0.06% 1.42% 1.08% 0.07%


Memory Basic
9.31 GiB
6.98 GiB
4.66 Gi
2.33 Gi
0
100%
75%
50%
25%
0%
Utilization%
2023-08-02 13:52:15
Total: 7.37 GiB
Avaliable: 6.27 GiB
Used: 1.10 GiB
Used%: 14.92%
14:30
14:40
14:50
15:00
15:10
min max avg current
Total 7.37 GiB 7.37 GiB 7.37 GiB 7.37 GiB
Avaliable 6.00 GiB 6.79 GiB 6.11 GiB 6.79 GiB

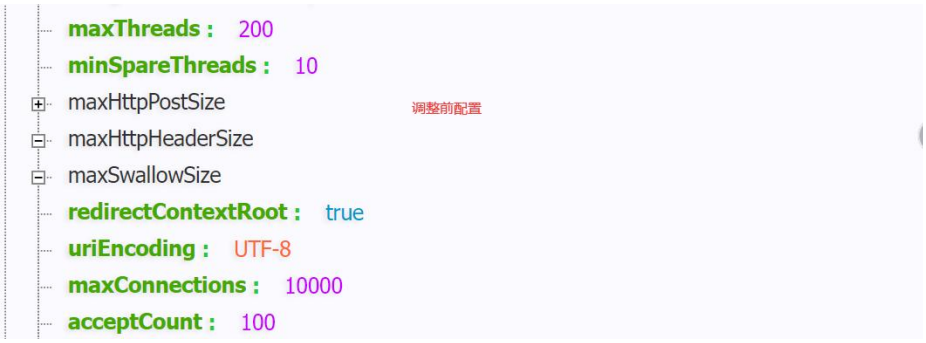


总的请求次数与错误率：



如图所示：以上测试总共请求 720000 次，0%的错误率，平均的 tps 在 385 左右，rt 在 p99 分位平均为 1.44s 左右。此时应用服务器的 cpu 使用率在 25%左右，内存使用率在 15%左右，网络上传平均在 3Mb/s，下载平均在 14Mb/s。因此硬件资源上没有达到瓶颈，尝试调整 tomcat 服务器配置。

Tomcat 调整为最大 800 线程数，相关配置前后对比图如下：

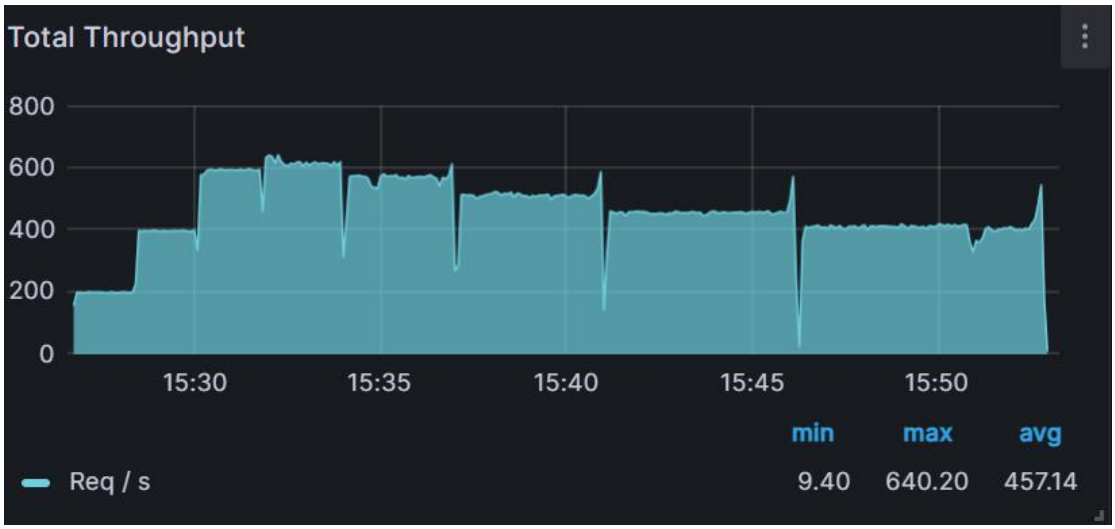


```
maxThreads : 800
minSpareThreads : 100
maxHttpPostSize
maxHttpHeaderSize
maxSwallowSize
redirectContextRoot : true
uriEncoding : UTF-8
maxConnections : 2000
acceptCount : 1000
additionalTldSkipPatterns :
```

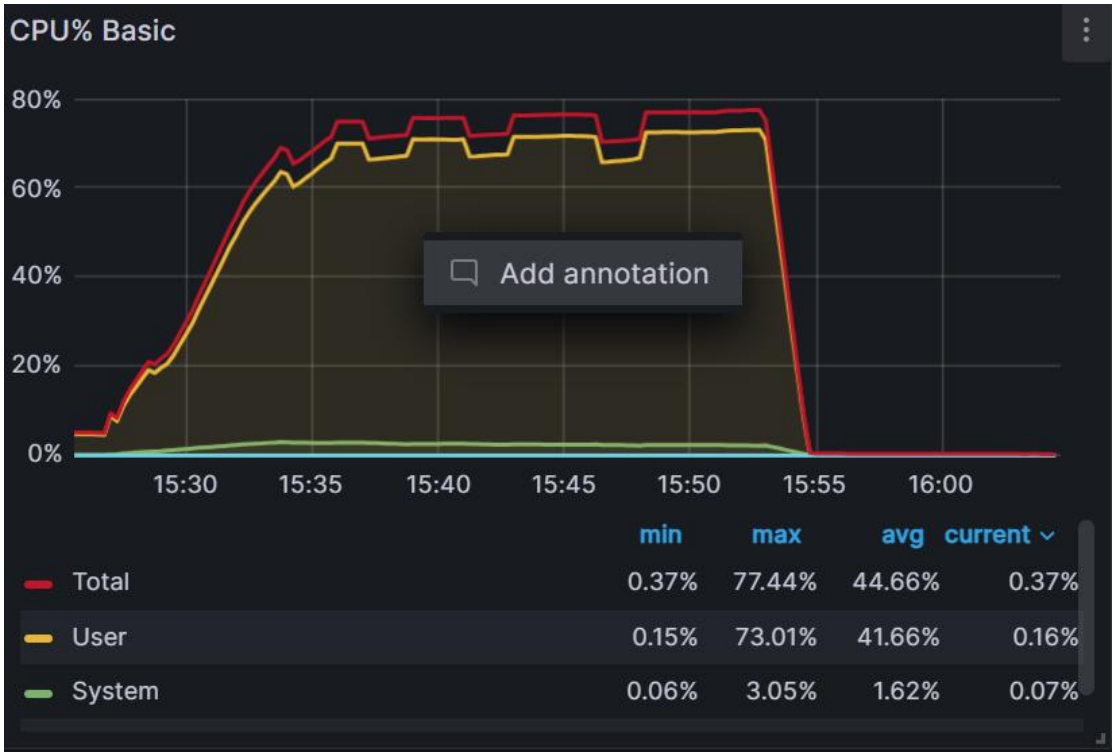
调整后配置

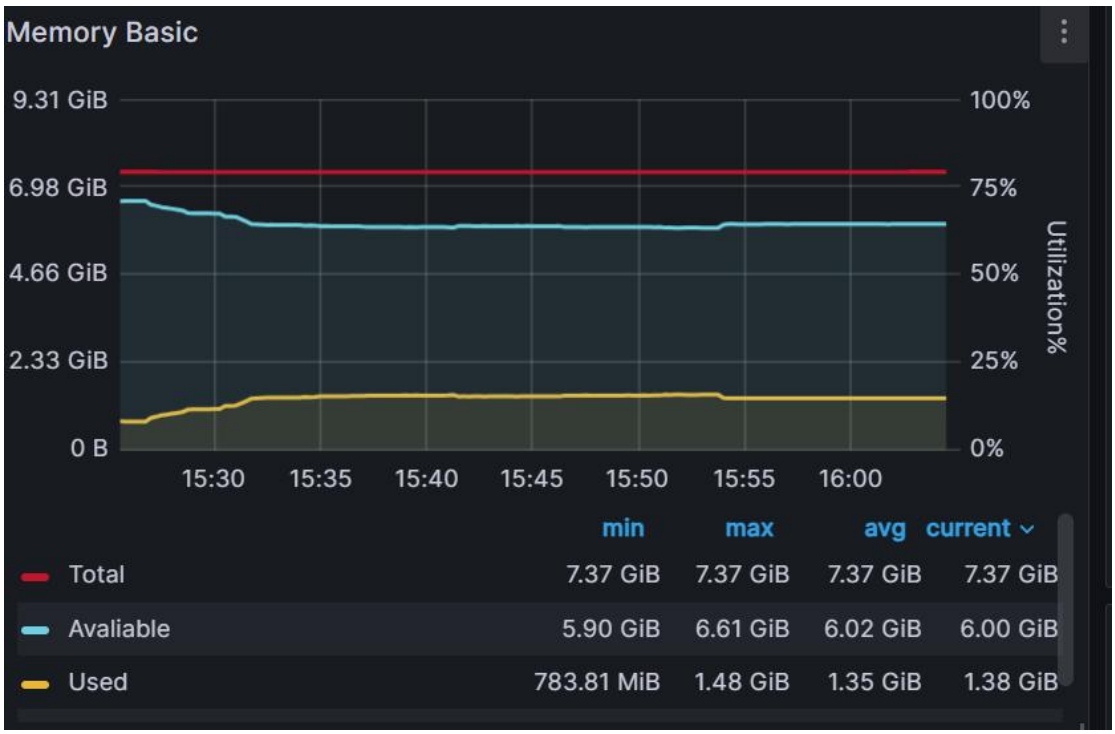
执行计划不变，再次通过梯度压测获取压测结果

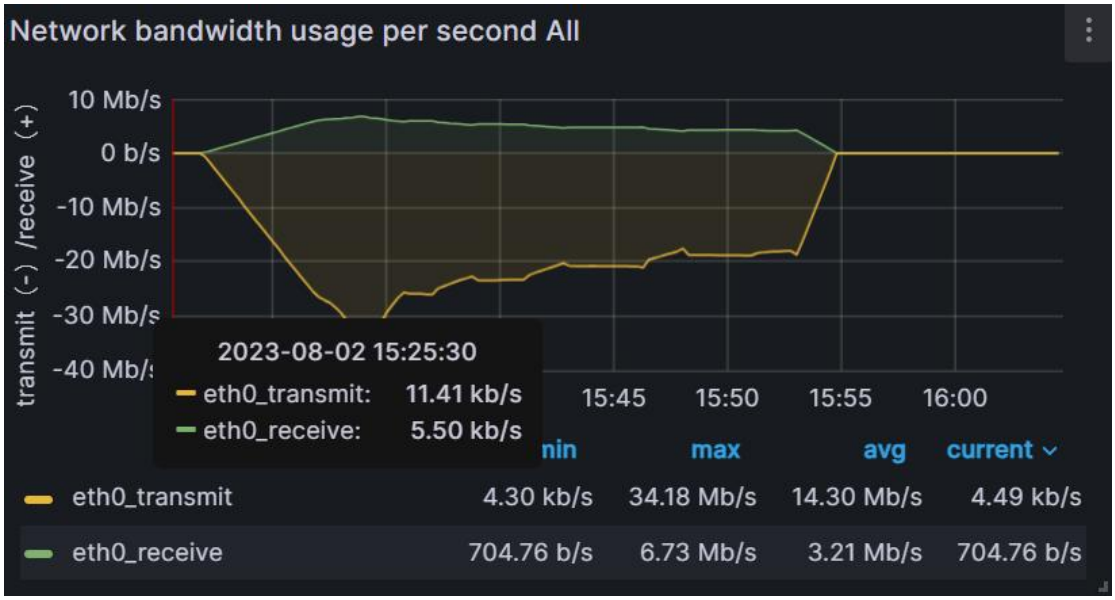
tps、rt



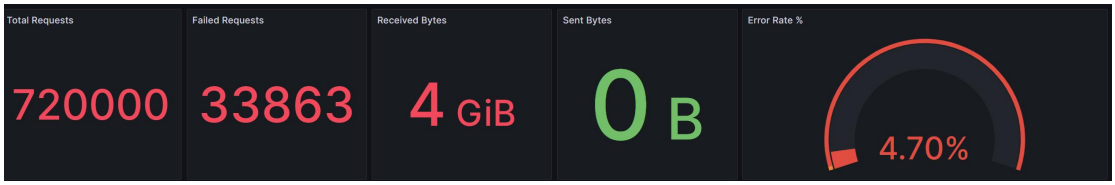
cpu、内存、网络：


CPU% Basic
80%
60%
40%
20%
0%
Add annotation
15:30
15:35
15:40
15:45
15:50
15:55
16:00
min
max
avg
current
Total 0.37% 77.44% 44.66% 0.37%
User 0.15% 73.01% 41.66% 0.16%
System 0.06% 3.05% 1.62% 0.07%


Memory Basic
9.31 GiB
6.98 GiB
4.66 GiB
2.33 GiB
0 B
100%
75%
50%
25%
0%
Utilization%
15:30
15:35
15:40
15:45
15:50
15:55
16:00
min
max
avg
current
Total 7.37 GiB 7.37 GiB 7.37 GiB 7.37 GiB
Avaliable 5.90 GiB 6.61 GiB 6.02 GiB 6.00 GiB
Used 783.81 MiB 1.48 GiB 1.35 GiB 1.38 GiB



总的请求次数与错误占比:



通过结果分析,发现在服务端提高 tomcat 的最大线程数后,tps 显著提高到了 457 左右,rt 在 p99 分位反而增加到了 1.98s 左右，并且错误率从 0 提高到了 4.7%左右，此时服务器资源内存平均使用在 1.35G 左右,网络上传平均在 3.21Mb/s,下载平均在 14.3Mb/s,但此时 cpu 使用率平均在 74%，发现增大线程数后主要限制于 cpu，导致出现了错误。此时查看系统的负载:

发现 cpu 负载在一段时间超过了 4，这应该就是 rt 上升，出现错误的原因。

后续可以通过增加 cpu 资源或者通过 openResty 反向代理应用集群的方式，进一步解决 cpu 资源受限的问题。