交大交交

测试评估报告

版本 1.0

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 28 /06/2020 | 1.0 | 性能测试报告 | 林江浩 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2. 测试过程明细 5

2.1 关于自主测试框架的搭建 5

2.2 Content微服务模块的测试 5

2.3 Message微服务模块的测试 6

2.4 SellInfo和File微服务模块的测试 8

3. 测试结果概述 11

4. 测试结果展示图 11

测试评估报告

# 简介

本测试报告是针对交大交交二手交易APP的测试文档。交大交交二手交易APP能让交大内的同学在平台上进行二手交易信息的发布。本次测试为性能测试，针对并发性、吞吐量、响应时间三个评价指标设计了三种测试场景，并试图探索不同场景下，整体系统的性能瓶颈所在，最后尝试给出可行的优化方案。本次测试采用python locust作为测试框架，采用promethues和grafana进行数据采集和可视化工作，使用VsCode编写和运行测试用例。

## 目的

本测试报告覆盖了针对并发性、吞吐量、响应时间这三个性能指标的性能测试，并在不同的测试场景上分析系统的性能瓶颈所在，最后给出相应的优化方案。

## 范围

本次性能测试针对三个不同的测试指标（并发性、吞吐量、响应时间）设计了三个测试场景，并对三种场景进行资源利用率的分析，测试场景大致情况如下：

1. 针对并发性，我们选取的测试场景是大量用户对读取头像的过程，这个操作是加载二手交易信息和查用用户个人信息时的必要操作，因此对系统并发性的要求较高。
2. 针对吞吐量，我们选取的测试场景是大量用户上传二手交易信息与相应的文件（包括图片和视频），因为这个过程涉及到大型文件的传输，对系统吞吐量的要求较高。
3. 针对响应时间，我们选取的测试场景是大量用户对系统发起不同关键词的整站二手交易信息检索请求，因为整站检索功能是一个复杂的关键词匹配与查找功能，涉及到对数据库的访问和读取，对系统响应时间的要求较高。

针对上述三种测试场景，我们也会通过对其资源利用率的使用情况，探索系统的性能瓶颈所在，并咋子分析后给出可行的优化方案。

## 定义、首字母缩写词和缩略语

|  |  |
| --- | --- |
| 缩写词 | 含义 |
| 交大交交 | 本次测试所针对的接口类和软件的名称 |
| 性能测试 | 针对系统整体性能的测试过程 |
| Locust | 本次测试使用性能测试框架 |

## 参考资料

《Software Testing - A Craftsman’s Approach》

## 概述

本次测试报告包含针对测试用例设计的说明，测试结果和分析讨论，以及对被测试软件改进的建议

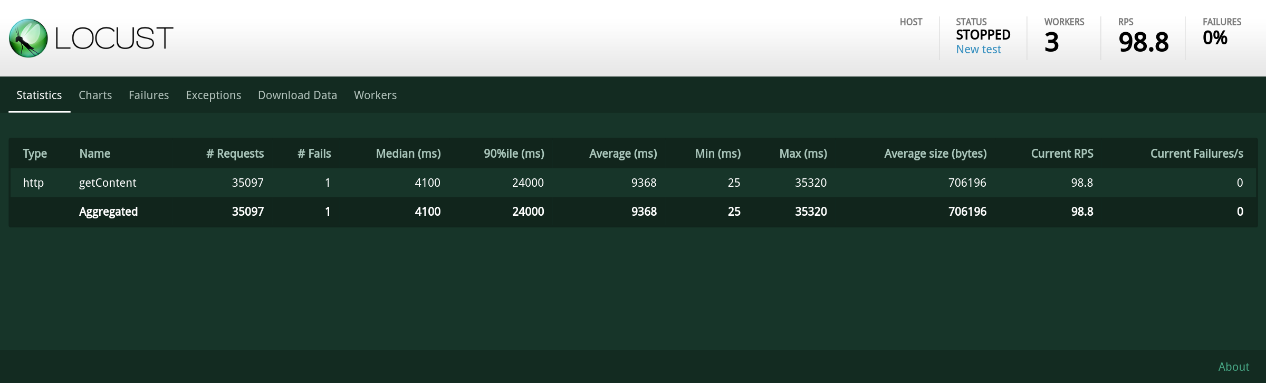
# 测试过程与结果分析

## 针对并发性的性能测试

**2.1.1 测试过程与结果概要**

并发性的性能集中体现在一些高概率同时触发的应用场景中，而如果这一操作能够触发小批量的吞吐则更容易出现瓶颈。在交大交交的整体系统下，查看二手交易信息和查看用户个人信息都会直接触发对用户头像的读取，而头像图片作为一个小文件也体现了一定的吞吐要求，因此本轮性能测试，我们设置的场景式多用户同时读取不同用户的个人头像。

在具体测试用例上，我们设置模拟用户数目为1000，并设置并发用户数量从零开始匀速爬坡，通过用户上传新头像并读取来测试系统整体的并发性。测试结果概要如下：



**图2.1.1 针对并发性的性能测试结果概要**

除此之外，我们记录了系统每秒请求数（Total Request per Second）、响应时间（Response Time）和用户数量（Number of Users）随时间的变化，具体如下图所示：



**图2.1.2 每秒请求数的变化**



**图2.1.3 响应时间的变化**



**图2.1.4 模拟用户数的变化**

可以发现，随着模拟用户数量的上升，系统的每秒请求数和响应时间也在慢慢攀升，当用户数量稳定在1000后，RPS稳定在100附近，也没有任何错误的返回，并发性并不高。

**2.1.2 分析与定位性能瓶颈**

接下来我们尝试进行性能瓶颈的分析，通过Grafana，我们首先观察网络带宽的相关情况：



**图2.1.5 网络带宽情况**

可以发现，网络带宽并没有达到对应的性能瓶颈（950MB/s），与测试实际用于上传/下载的头像文件是一个小文件的事实相符合。更进一步，我们考虑是在当请求进入到application，因为memory和cpu的使用导致的性能瓶颈，但如下图所示，我们实际观察到的memory和cpu的使用量并不高。

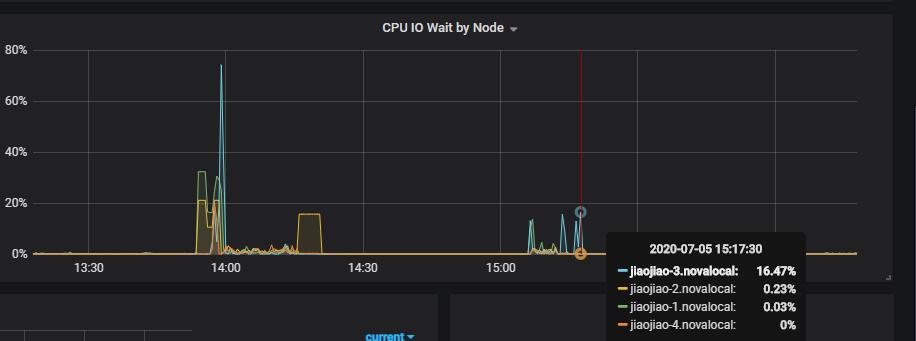


**图2.1.6 CPU使用量**



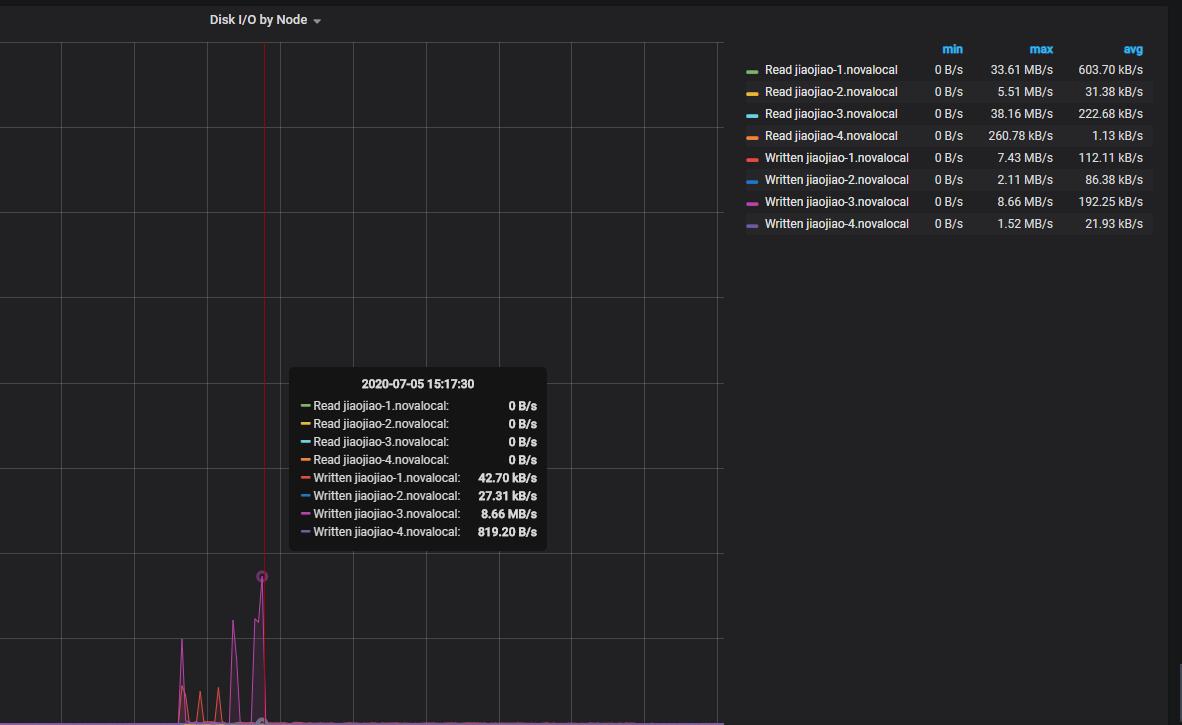
**图2.1.7 内存使用量**

因此我们，我们进一步考查磁盘存在的性能瓶颈：



**图2.1.8 磁盘IO的使用情况**

由于mongodb服务在jiaojiao-3.novalocal中，file srv使用mongodb存储文件，我们观察到CPU 等待 IO的时间较长，且对比其他的node异常显著。如下图所示，按照计算，我们上传的图片大小为700kb左右，在100 RPS下，会产生70MB/s的数据流量远远大于这里的峰值8.66MB/s（mongodb有写入memory buffer的相关优化，不再赘述）。进一步了解到其底层使用机械硬盘，写入速率大约在10MB/s，与数据实际情况符合，结合CPU IO wait time，可以认定磁盘IO为性能瓶颈。



**图2.1.9 性能瓶颈计算数据图**

**2.1.3 优化方案**

针对上述磁盘IO的性能瓶颈，我们提出以下三种可能的优化方案：

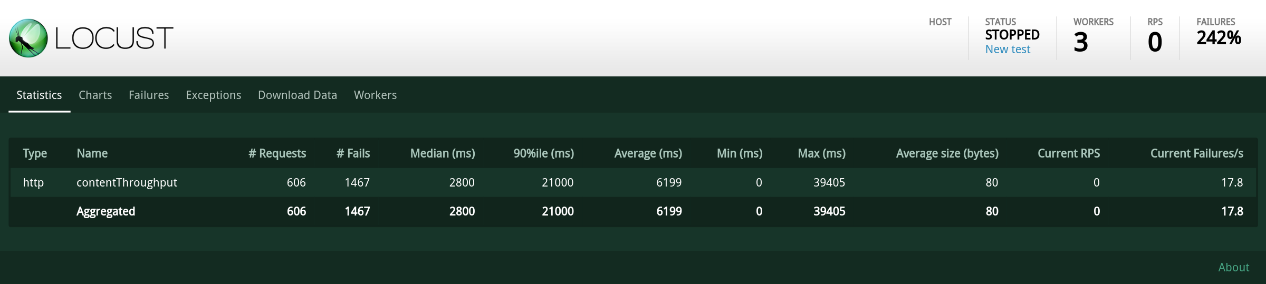
1. 使用SSD或RAID10来提供存储IOPS能力。
2. mongodb层面上可做的优化较多，例如分离journal和数据的disk，调整日志的刷新时间周期。
3. mongodb 进行分片，使得能够同时享有多个磁盘的I/O速度，注意这里如果是使用到了同一个物理机上的虚拟机，无法获得性能上的提升。

## 针对吞吐量的性能测试

**2.2.1 测试过程与结果概要**

在交大交大系统中，用户在创建二手交易信息时会涉及到对图片、视频等大文件的上传，而对大文件的传输过程往往涉及到系统整体的吞吐量的需求，因此针对吞吐量的性能指标，我们选取的测试场景式大量用户并发上传大文件并创建商品交易信息，其中文件内容为20~50MB的随机网络包。

测试用例上，我们设置模拟30个用户并发执行文件上传与创建工作，测试结果概要如下：



**图2.2.1 针对吞吐量的测试结果概要**



**图2.2.2 每秒请求数的变化**



**图2.2.3 响应时间的变化**



**图2.2.4 并发用户数量的变化**

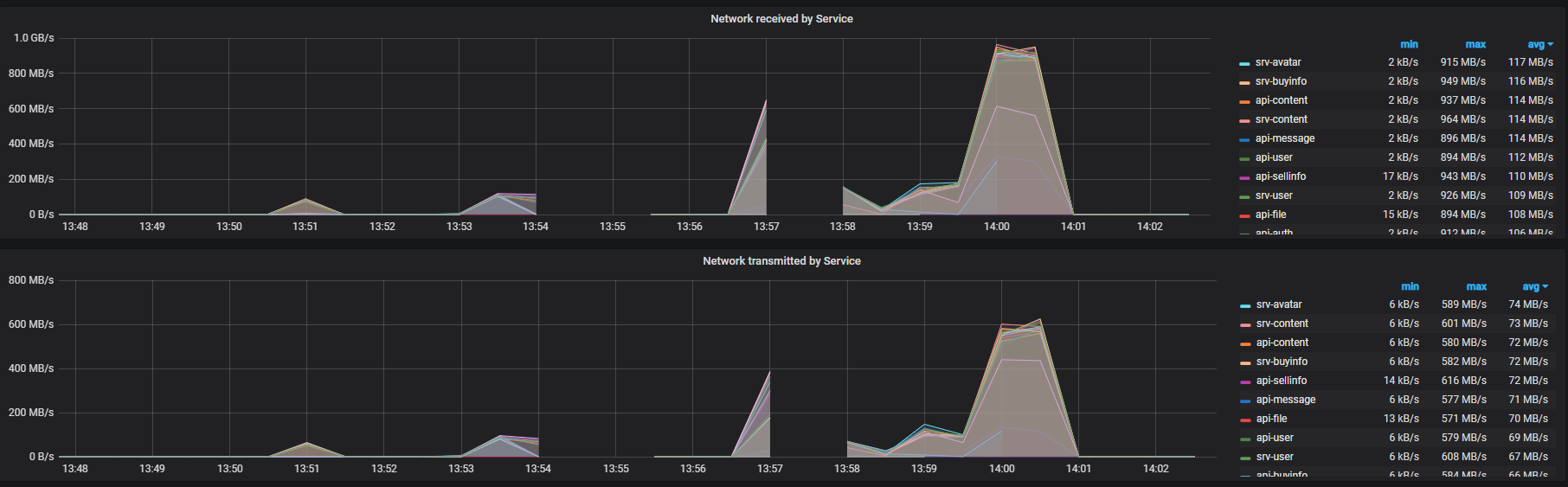
**2.2.2 测试过程与结果概要**

可以很明显发现的是，此次压力测试的错误率相当之高，这表明系统并没有扛住这一波请求负载，下面我们来详细分析一下其中的原因。

可以看到整个测试结果可以明显划分为三个阶段：

1. 第一阶段，RPS随并发请求数量的增加，出现了快速上升，在达到高峰后迅速下降。
2. 第二阶段，RPS和响应时间呈现较低水平的起起落落。
3. 第三阶段，RPS直接清零，响应时间达到较高水平。

接着，我们观察网络带宽使用情况：

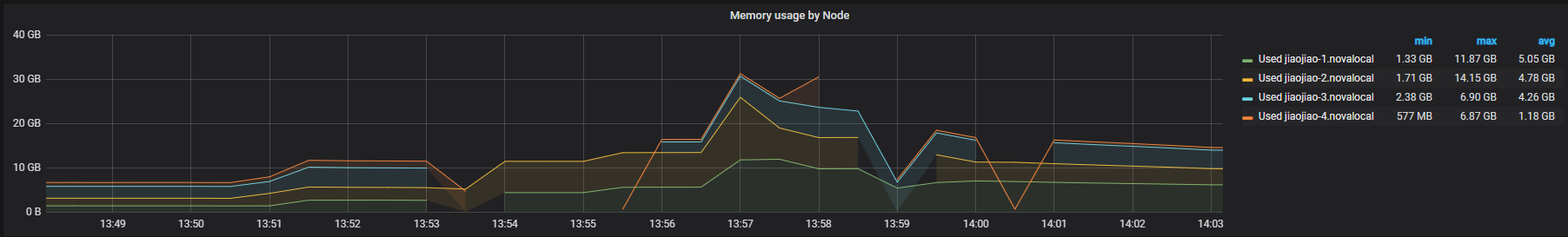


**图2.2.5 网络带宽使用情况**

单个服务网络带宽使用最高已经达到964MB/s，所有服务总和加起来的带宽使用将更加庞大。已知被测试服务器安装网卡是万兆网卡，即当前网络带宽使用已经几乎达到网卡的收发极限速度。甚至可以看到，由于对网络带宽的极度挤占，监控程序所显示的图形也断断续续。

因此，在运行初期，整个系统的性能瓶颈在于网卡速度。

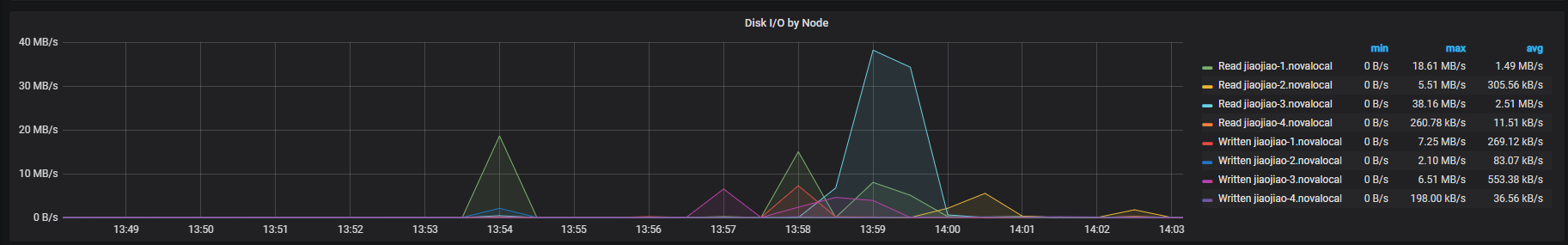
然后，我们观察内存使用情况：



**图2.2.6 内存使用变化情况**

可以发现，自测试启动开始，内存使用率就开始急速飙升，直接占满整个32GB的内存空间。由于网卡高速收发包来不及处理，都临时缓存到内存中，这阶段系统仍然有机会提供正常的服务，RPS也在这阶段迅速增加因此内存迅速占满。在内存占满之后，没有地方能够缓存网络包，因此RPS开始迅速下降了。

之后，系统必须及时将所有缓存在内存中的数据刷入磁盘，而磁盘的IO速度是极慢的，仅为几十兆每秒，因此磁盘IO成为系统第二个阶段的瓶颈，情况如下图所示：



**图2.2.7 磁盘IO使用变化情况**

在此之后，RPS进入低水平状态持续。后期，突然降为零，经查证属于mongoDB挂了。由于内存占满溢出，Docker异常退出，自此停止服务。

综上，在本轮针对吞吐量的性能测试中，系统一共遇到了两个性能瓶颈：

1. 网卡速度瓶颈
2. 磁盘IO瓶颈

**2.2.2 优化方案**

针对上述性能瓶颈，我们提出以下三条优化措施：

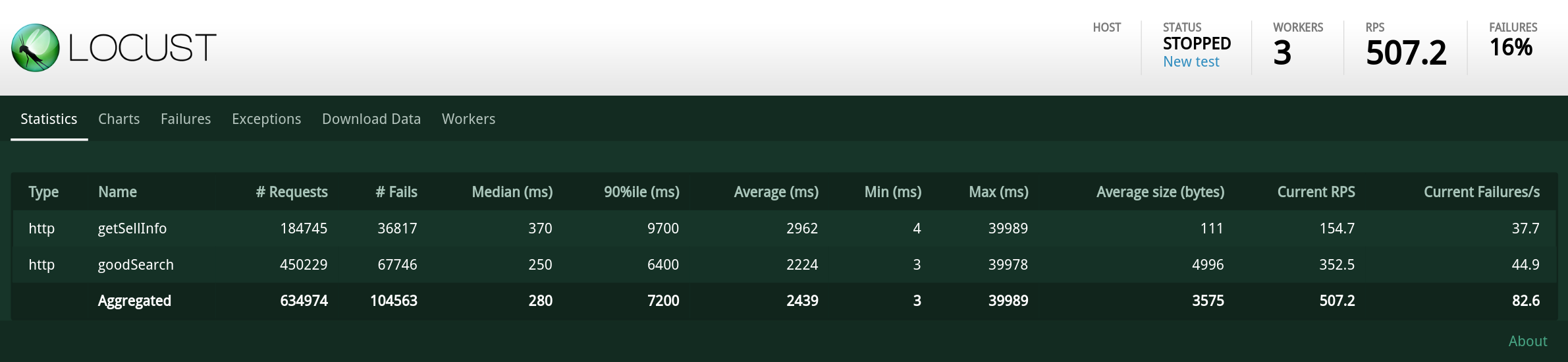
1. 换用更先进的网卡
2. 使用DPDK进行网卡收发加速，定制化服务。
3. 数据库分片分区，实现多地写，提高IO性能。

## 针对响应时间的性能测试

**2.3.1 测试过程与结果概要**

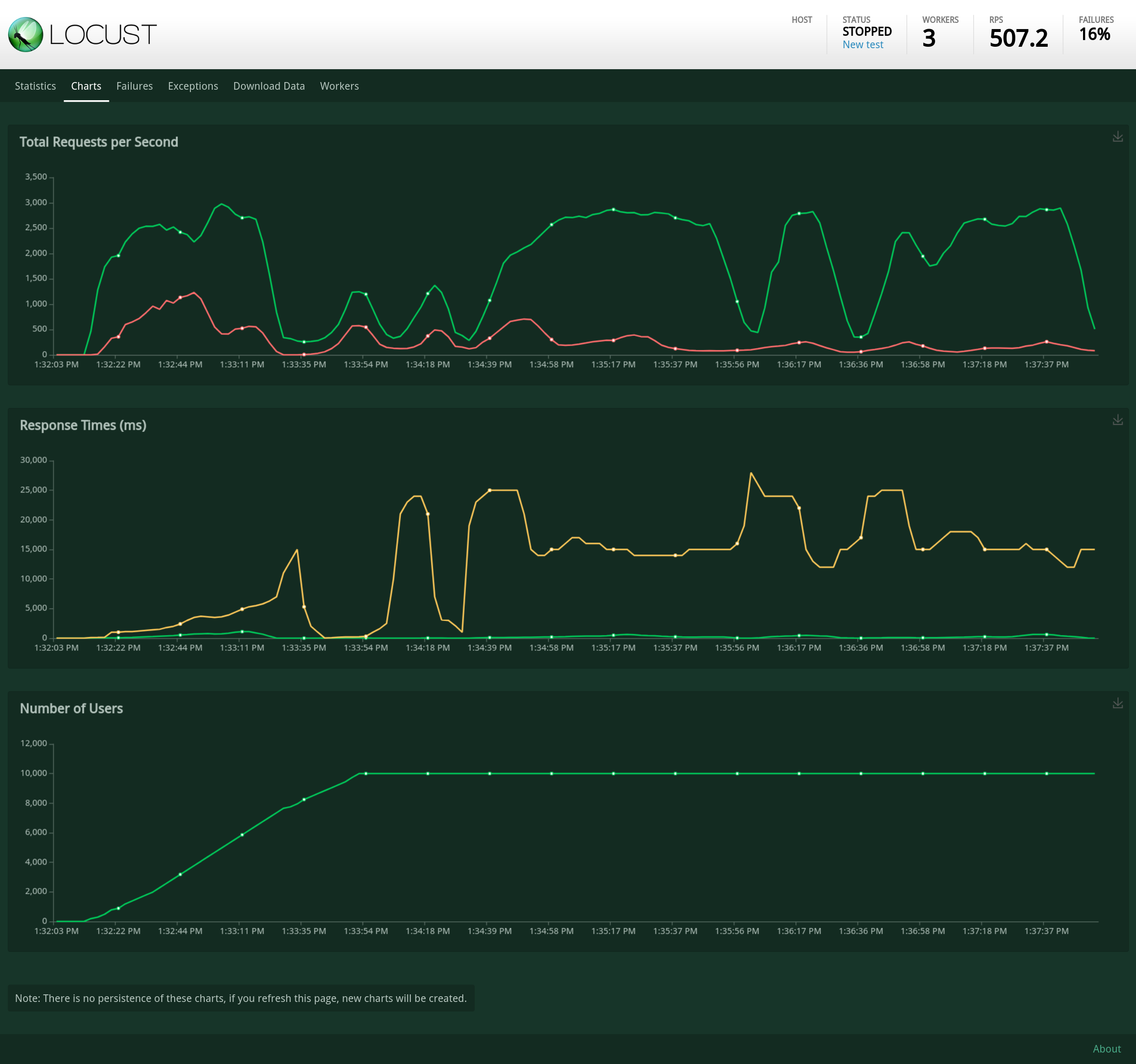
响应时间较高的服务请求往往集中于要进行复杂运算或者大量磁盘IO的任务中，为了令测试场景区分于2.1和2.2，我们选取的测试场景是大量请求信息检索的场景，交大交交中的信息检索采用的是全文关键词匹配，会涉及到较为复杂的字符匹配算法和数据库遍历操作，适合于本轮性能测试。

在具体测试用例上，我们设置模拟用户数目为10000，并设置并发用户数量从零开始以每秒100并发量的增速进行匀速爬坡，通过不断触发整站搜索请求来观察响应时间的变化。测试结果概要如下：

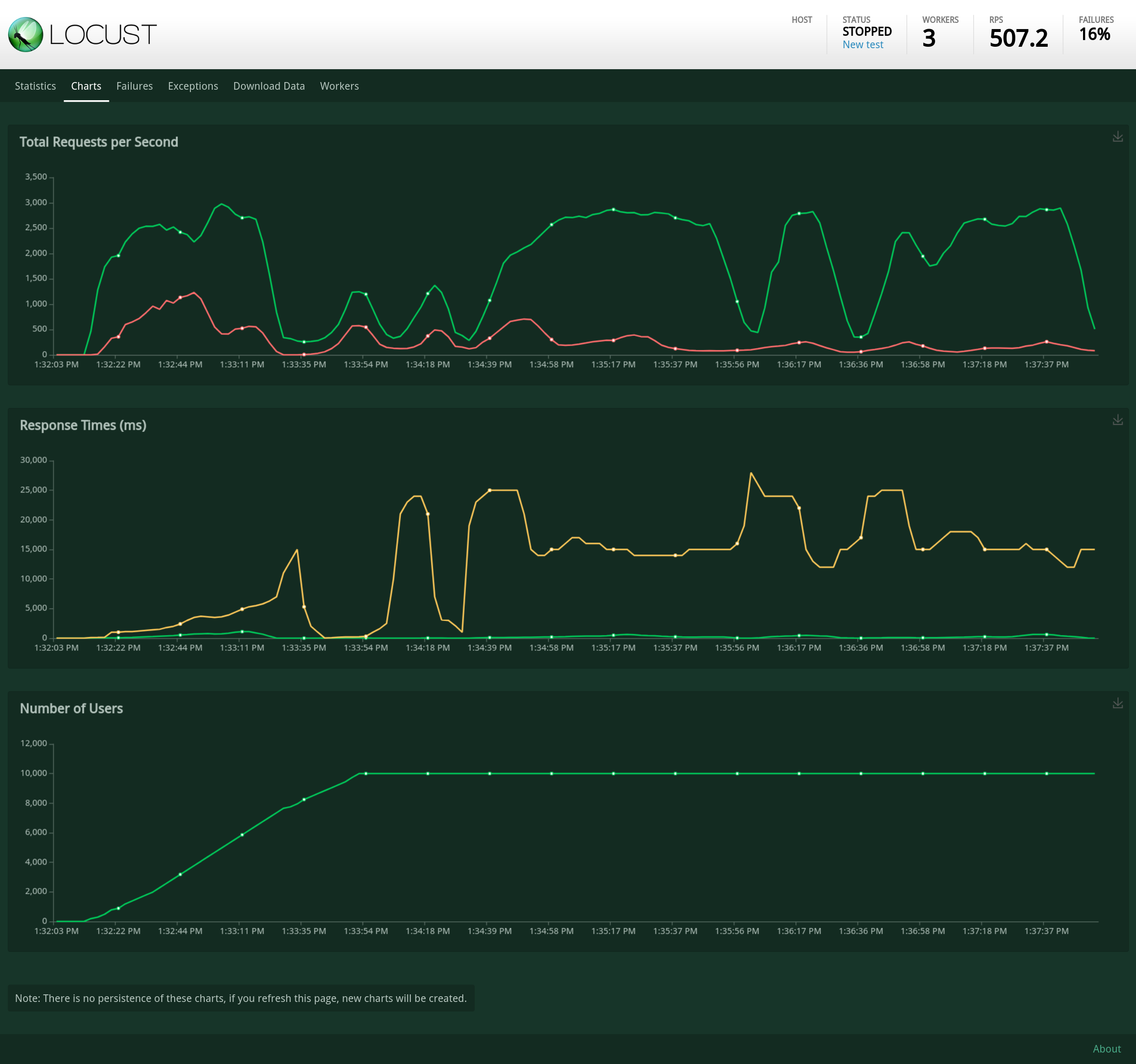


**图2.3.1 针对响应时间的性能测试结果概要**

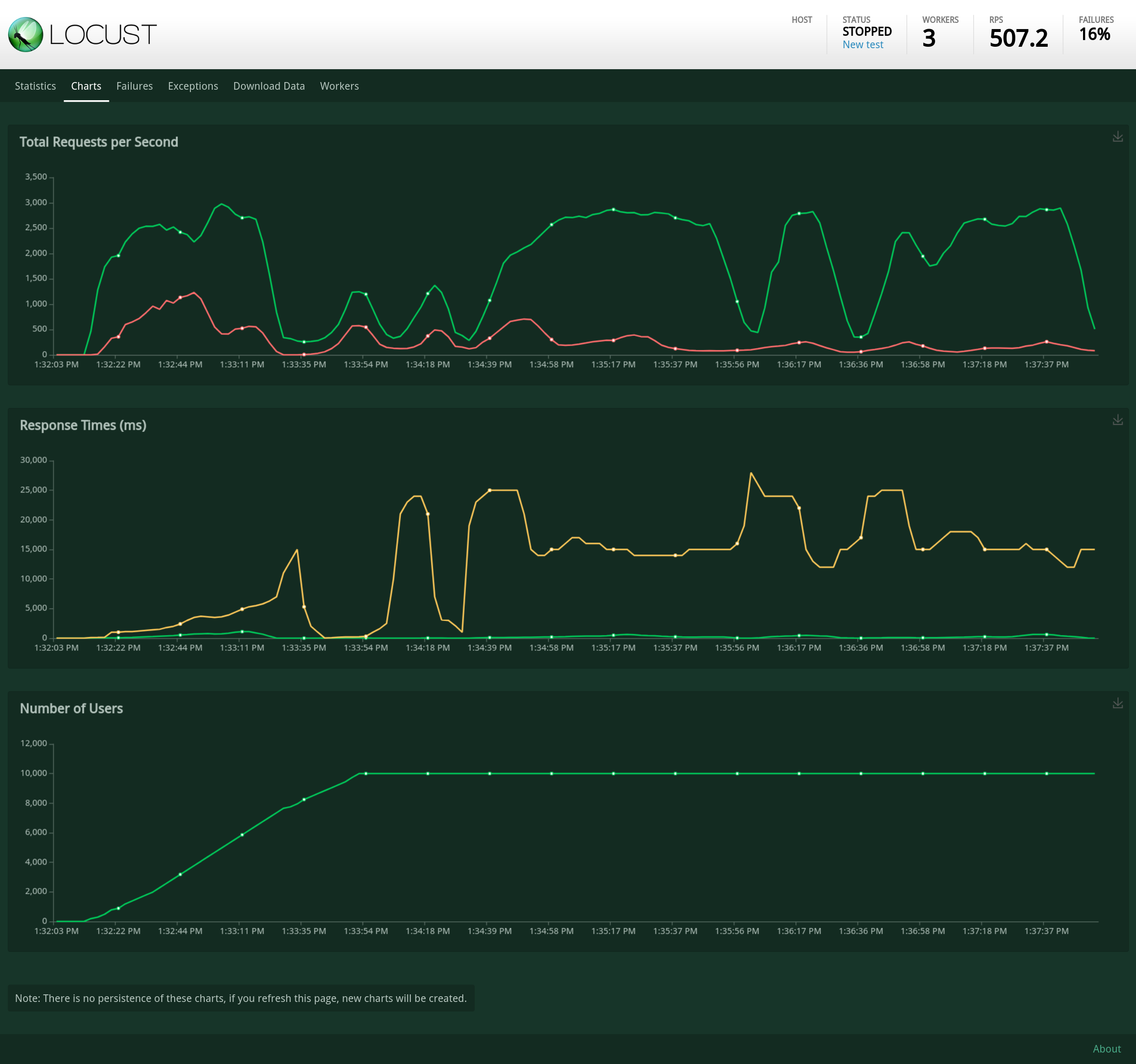
除此之外，我们记录了系统每秒请求数（Total Request per Second）、响应时间（Response Time）和用户数量（Number of Users）随时间的变化，具体如下图所示：



**图2.3.2 每秒请求数的变化**



**图2.3.3 响应时间的变化**



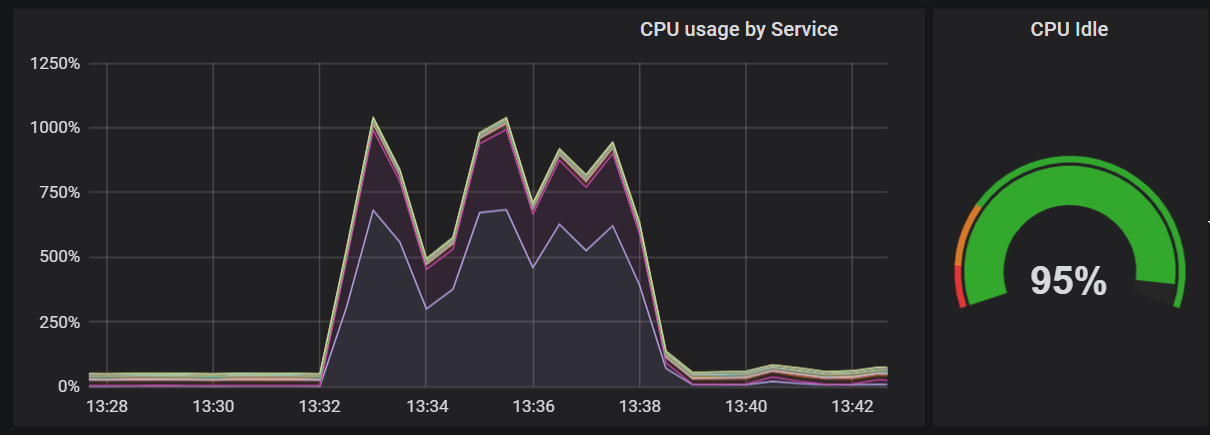
**图2.3.4 模拟用户数的变化**

可以发现，随着模拟用户数量的上升，系统的每秒请求数和响应时间也在慢慢攀升，当用户数量稳定在10000后，RPS和响应时间却开始围绕1500Req/s和15s做周期性波动，同时并且存在一定的请求失败的情况。这是因为过高的用户并发数量超出了系统支持的TCP量，而系统在取消TCP上需要一定的时延，因此导致测试过程中RPS伴随着TCP用量被占满又释放而产生了周期性的波动。测试可得，在该场景下，交大交交系统可以较好地应对5000以下的并发请求，之后响应时间和RPS将随着请求数量的增加而产生性能断崖。

**2.2.2 分析与定位性能瓶颈**

接下来我们尝试进行性能瓶颈的分析。首先，不同于2.1与2.2的测试场景，对于交易信息的关键词检索服务不涉及到对文件的操作，因此基本不会对磁盘IO、内存占用和网络带宽产生很大的压力，而对Grafana的观察也印证了我们的猜想：网络占用仅波动在25MB/s上下，磁盘IO的使用仅波动在2MB/s上下，内存占用率峰值仅为40%。

进一步思考我们的测试场景，因为没有涉及对文件读写的操作，因此大规模的运算需求大概率会成为系统的性能瓶颈，我们通过观察CPU使用量可以印证我们的猜想，并将性能瓶颈定位在CPU上：



**图2.3.5 CPU使用率**

**2.3.3 优化方案**

针对上述CPU的性能瓶颈，我们提出以下三种可能的优化方案：

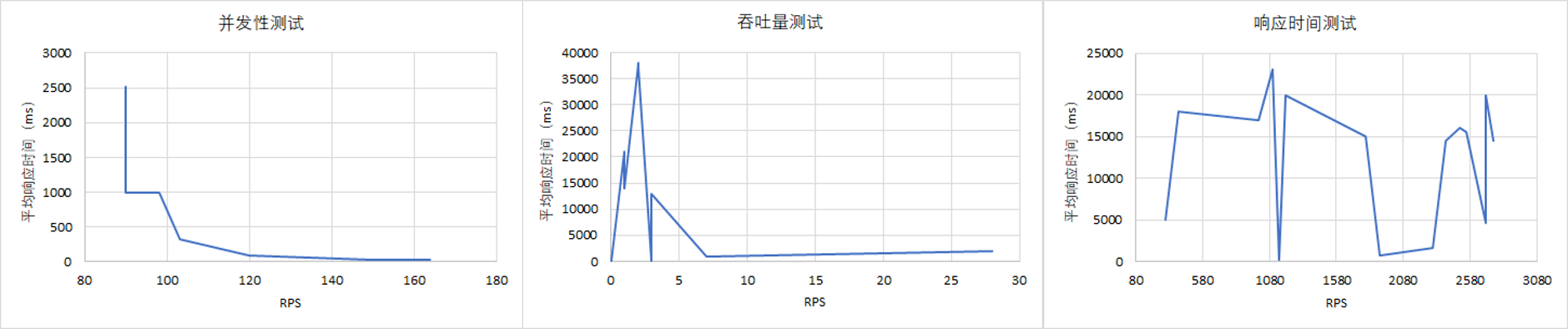
1. 通过优化关键词匹配的算法（目前交大交交系统的关键词检索仍然是采用最naïve的全文遍历检索方式），减少单次检索请求对CPU的占用。
2. 另外也可以考虑对系统整体的硬件资源进行升级，加钱换新永远是一个解决性能的方法。

# 关联用例与分析

本段说明补于2020.07.05，在截止提交日期之前，我们和其他小组在一个大群中交流时才发现我们对“关联的测试用例”的认知出现了偏差，在询问助教后被告知两种理解都可以进行，因此本次测试实验中，我们对“关联的测试用例”的理解是对上述三种场景的测试结果进行关联分析（correlation）。

针对上述三个场景下的关联用例与分析结果呈现出一定的共性与差别，因此我们将其单独作为一个模块进行展示，并进行一些横向的比较与分析。

## 响应时间与每秒请求数的关联

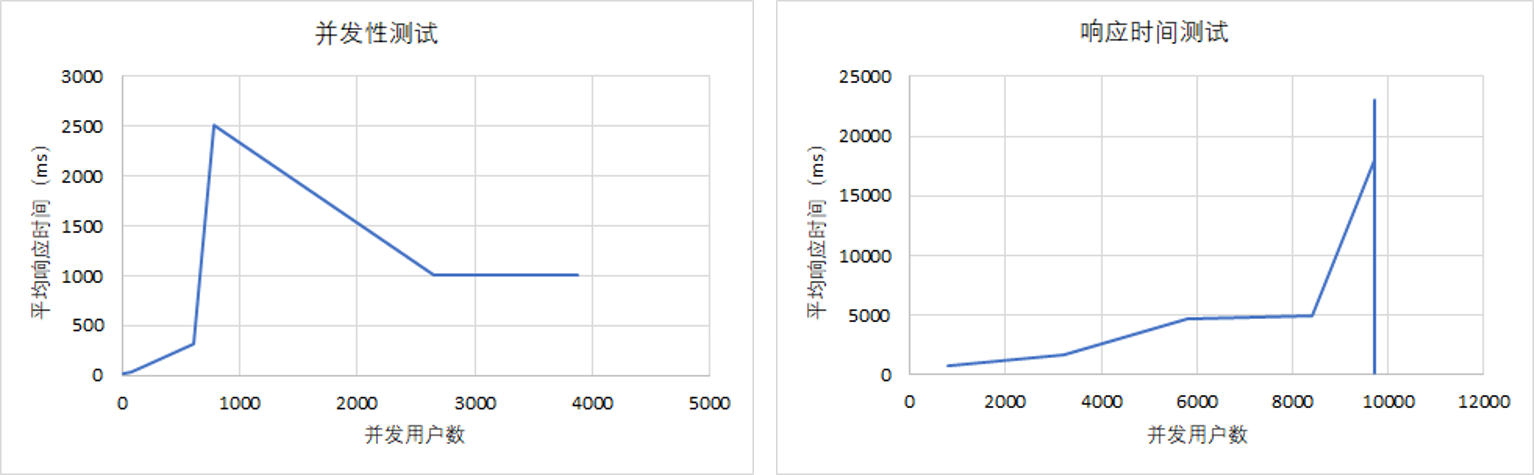


**图3.1.1 响应时间与每秒请求数的关联图**

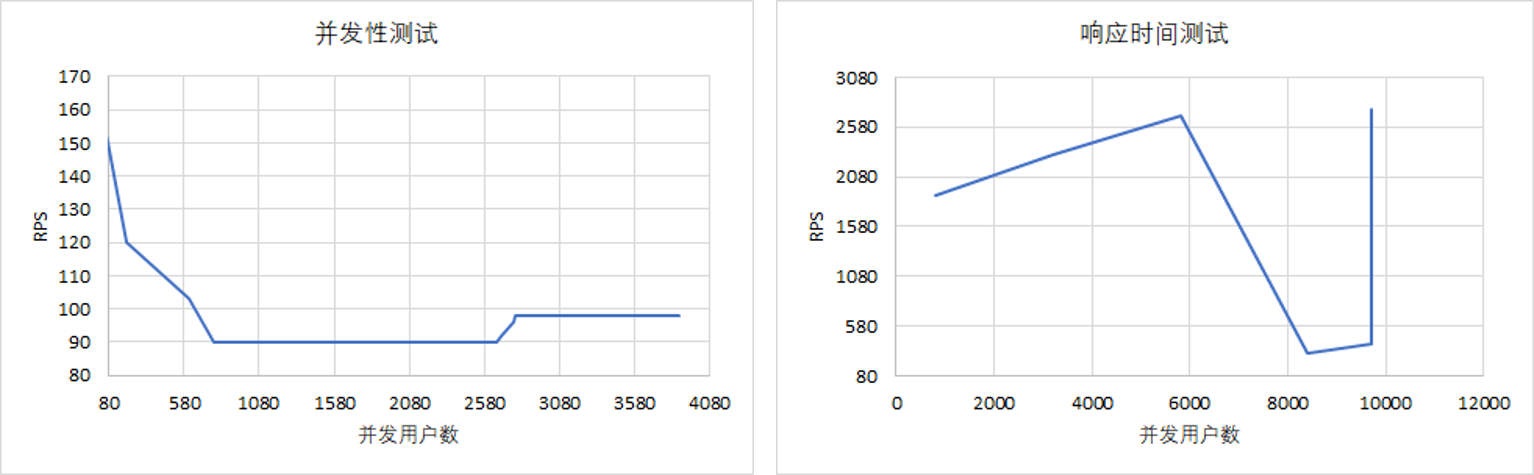
可以发现，在并发性测试与吞吐量测试中，平均响应时间与RPS保持基本的变化关系，即平均响应时间随着系统RPS的提升而下降，符合预期。但是在针对响应时间的测试场景中，平均响应时间关于RPS的关系呈现出无规律的波动。我们分析认为，这个和请求包的大小有关，因为在前两种测试场景中，请求都会涉及到对文件的读写，而最后一场测试场景只是关键词匹配，并无文件级别的读写，因此单个网络包较小，数量较多，容易受到CPU占用率的影响；再加上测试过程中，若中途网络包传输卡壳会导致包的到达速度波动，进而导致CPU占用也出现随机波动，进而有了上图中的结果。

## 响应时间/RPS与并发用户数的关联

首先，因为在针对吞吐量的测试中，我们设置的并发用户数量上限为30，因此本身并发用户数没有爬坡上升的空间，基本就是一步到位，我们更多的是考虑后期用户数稳定在30后的系统运转情况，因此短时间爬坡过程中的测量结果不够精准（做出的图基本就是一条垂直的线），这里不予考量。



**图3.2.1 响应时间与并发用户数的关联图**



**图3.2.2 RPS与并发用户数的关联图**

在并发性测试中，可以发现，随着并发用户数的增加，平均响应时间会先增后减最终稳定，RPS则逐渐减低并最终稳定，前期的系统尚未触及性能瓶颈，在并发用户数达到约2580后，系统整体的RPS与平均响应时间保持稳定，进入瓶颈期。

不同于针对并发性的测试，在针对响应时间的测试中，根据2.3中的分析，因为系统最终的性能瓶颈落在了CPU上，因此当并发用户数突破6000后，因为CPU占用问题，后来的网络包难以被即使处理和相应，因此系统的整体性能呈现出一个断崖式的下降（RPS骤降与平均响应时间骤升）。随后，当并发用户数稳定在10000后的持续测试中，RPS和平均响应时间开始呈现无规则的数值波动，与我们在2.3中的观察结果一致，分析原因可见2.3。