**EMQ X 千万连接性能测试报告**

2019/6

关于 EMQ 2

特别致谢 2

测试结果概述 3

测试场景 4

千万连接测试 4

Pub消息测试 5

50万主题使用测试 6

消息广播测试 7

测试环境 8

测试工具 10

测试结果 10

千万级连接测试 10

Pub消息测试 12

50万主题使用测试 15

消息广播测试 18

附录 21

系统优化 21

测试脚本 21

## 关于 EMQ

EMQ (<https://www.emqx.io>) 是家面向 5G 和物联网市场的消息与流处理开源软件公司。

2013年，EMQ 作为一个开源项目在 Github上发布，目前已成为全球市场广泛应用的开源 MQTT 消息服务器。 2017年，在客户需求快速增长的驱动下，我们创立公司并推出 EMQ X 企业版，以加快开源项目的产品化部署应用，并为大型企业客户提供专业技术支持与服务。

EMQ 创始团队主要来自华为、IBM、亚马逊和苹果，拥有多年的工作经验和对行业的深刻理解，并对开源软件产业的未来抱有坚定的信念。

## 特别致谢

本次测试得到了华为云和 XMeter 的大力帮助。华为云为本次测试提供了所有测试所需的机器和网络资源；XMeter 为本次测试提供了支撑千万 MQTT 连接所需的专业性能测试工具平台。

华为云（<https://www.huaweicloud.com>）：华为云是华为公司倾力打造的云战略品牌，致力于为全球客户提供领先的公有云服务，包含弹性云服务器、云数据库、云安全等云计算服务，软件开发服务，面向企业的大数据和人工智能服务，以及场景化的解决方案。

XMeter (<https://www.xmeter.net>)：云端SaaS性能测试产品XMeter, 让性能测试变得简单易行。XMeter致力于在软件性能测试领域为客户提供高效便捷的性能测试生产工具及技术领先的低成本性能测试解决方案。 XMeter 在开源社区提供的 MQTT JMeter 插件，在全球有众多用户。

## 测试结果概述

本次测试在华为云上进行，主要进行了千万级的 MQTT 连接测试，以及多种组合的消息吞吐量测试，相关的测试结果如下所示。

**1000万连接测试**：该测试只测试 MQTT 的连接，除了固定时间的Ping 包，每个客户端不会pub/sub 任何MQTT消息：

* 90%平均响应时间< 2ms；最大响应时间为 2.3秒
* 每个节点CPU平均使用率大约为8%左右，最高不超过20%
* 每个节点平均内存使用约9GB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **最大 响应时间** | **最小 响应时间** | **前90% 平均 响应时间** | **EMQ X  CPU 使用** | **EMQ X  内存使用** |
| 2.308s | 0s | 0.0015s | 8%-20% | 9GB |

**消息吞吐测试**：背景连接900万，用于模拟具有连接和发送 Ping 包的客户端，而剩下的100万则模拟用于发送实际业务数据的客户端

* 消息体为88字节
* 90% 平均响应时间 <70ms
* CPU使用随着消息吞吐量增加变大
* 每个节点平均内存使用在9 - 10GB

**消息吞吐测试类型1：Pub消息测试**

该测试类型下除了900万的背景连接，剩下的100万业务连接只做 pub 消息的操作。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Pub**  **(k/s)** | **Qos** | **最大 响应时间** | **最小 响应时间** | **前90% 平均响应时间** | **EMQ X CPU 使用** | **EMQ X 内存使用** |
| 200 | 0 | 6.173s | 0s | 0.0001s | 15%-20% | 9GB |
| 200 | 1 | 7.484s | 0s | 0.0087s | 15%-25% | 9GB |

**消息吞吐测试类型2： 50万主题使用**

该测试类型下除了900万的背景连接，剩下的100万连接主要将大量的主题用于消息转发的测试。该测试结果表明在使用50万主题的情况下，QoS 在0的时候，Pub和Sub的吞吐量分别都可以稳定达到20万/秒；而 QoS 在 1 的时候，Pub和Sub的吞吐量分别都可以稳定达到10万/秒。在大量主题使用的场景下，EMQ X 表现良好。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Pub /Sub**  **(k/s)** | **Qos** | **最大 响应时间** | **最小 响应时间** | **前90% 平均响应时间** | **EMQ X CPU 使用** | **EMQ X 内存使用** |
| 200 | 0 | 12.014s | 0s | 0.0353s | 25%-30% | 9GB |
| 100 | 1 | 5.477s | 0.0183s | 0.0063s | 20%-25% | 9GB |

**消息吞吐测试类型3： 消息广播测试**

该测试类型下除了900万的背景连接，剩下的100万连接主要做消息广播的测试。该测试场景下，主要是用少量的 Pub 端往主题上发送消息，大量的 Sub 端在订阅相关的主题，实现消息在大量 Sub 端的广播。测试结果显示，在 QoS 为0 的情况下，广播可以达到180万/秒的消息吞吐量；而 QoS 为1的情况下，广播可以达到90万/秒的消息吞吐量。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pub**  **（k/s）** | **Sub**  **(k/s)** | **Qos** | **最大 响应时间** | **最小 响应时间** | **前90% 平均响应时间** | **EMQ X CPU 使用** | **EMQ X 内存使用** |
| 4 | 1800 | 0 | 159.561s | 0s | 0.0654s | 30%-45% | 9.5G |
| 2 | 900 | 1 | 140.271s | 0s | 0.0500s | 35%-55% | 9.8G |

## 测试场景

### 千万连接测试

该测试场景只测试客户端到服务器端的MQTT协议连接，验证千万级并发连接下EMQ X的表现。关注的主要参数有连接响应时间、EMQ X 服务器的 CPU、内存、磁盘和网络等使用情况。

**消息吞吐类测试**

### Pub消息测试

该测试类型下除了900万的背景连接，剩下的100万业务连接只做 pub 消息的操作。该测试主要关注的目标是在 Pub 大量消息的情况下，EMQ系统的表现。关注的主要参数有：消息 Pub 响应时间、EMQ X 服务器的 CPU、内存、磁盘和网络等使用情况。

#### QoS 0 \* 20万/秒

* 连接
  + 900万背景连接
  + 100万 Pub 端连接
* Pub: 20万/秒，100万Pub 客户端，每间隔5秒Pub一次消息
* QoS: 0
* 主题数: 1000
* 消息体：88字节

#### QoS 1 \* 20万/秒

* 连接
  + 900万背景连接
  + 100万 Pub 端连接
* Pub: 20万/秒，100万Pub 客户端，每间隔5秒Pub一次消息
* QoS: 1
* 主题数: 1000
* 消息体：88字节

### 50万主题使用测试

该测试类型下除了900万的背景连接，剩下的100万连接主要将大量的主题用于消息转发的测试。该测试主要关注的目标是在使用大量主题的情况下，EMQ系统的表现。关注的主要参数有：消息 Pub 、Sub响应时间、EMQ X 服务器的 CPU、内存、磁盘和网络等使用情况。

#### Pub: QoS 0 \* 20 万/秒 + Sub: QoS 0 \* 20万/秒

* 连接
  + 900万背景连接
  + 50万 Pub 端连接
  + 50万 Sub 端连接
* Pub: 20万/秒，每客户端间隔2.5秒Pub一次消息
* Sub: 20万/秒，每客户端间隔2.5秒收到一次消息
* QoS: 0
* 主题数: 50万
* 消息体：88字节

#### Pub: QoS 1 \* 10 万/秒 + Sub: QoS 1 \* 10万/秒

* 连接
  + 900万背景连接
  + 50万 Pub 端连接
  + 50万 Sub 端连接
* Pub: 10万/秒，每客户端间隔5秒Pub一次消息
* Pub: 10万/秒，每客户端间隔5秒收到一次消息
* QoS: 1
* 主题数: 50万
* 消息体：88字节

### 消息广播测试

该测试类型下除了900万的背景连接，剩下的100万连接主要做消息广播的测试。该测试场景下，主要是用少量的 Pub 端往主题上发送消息，大量的 Sub 端在监听相关的主题，实现消息在大量 Sub 端的广播。该测试主要关注的目标是在使用消息广播场景下，EMQ系统的表现。关注的主要参数有：消息 Pub 、Sub响应时间、EMQ X 服务器的 CPU、内存、磁盘和网络等使用情况。

#### Pub: QoS 0 \* 4 k/秒 + Sub: QoS 0 \* 180万/秒

* 连接
  + 900万背景连接
  + 10万 Pub 端连接
  + 90万 Sub 端连接
* Pub: 4千/秒，每客户端间隔25秒Pub一次消息
* Sub: 180万/秒
* QoS: 0
* 主题数: 2k
* 消息体：88字节

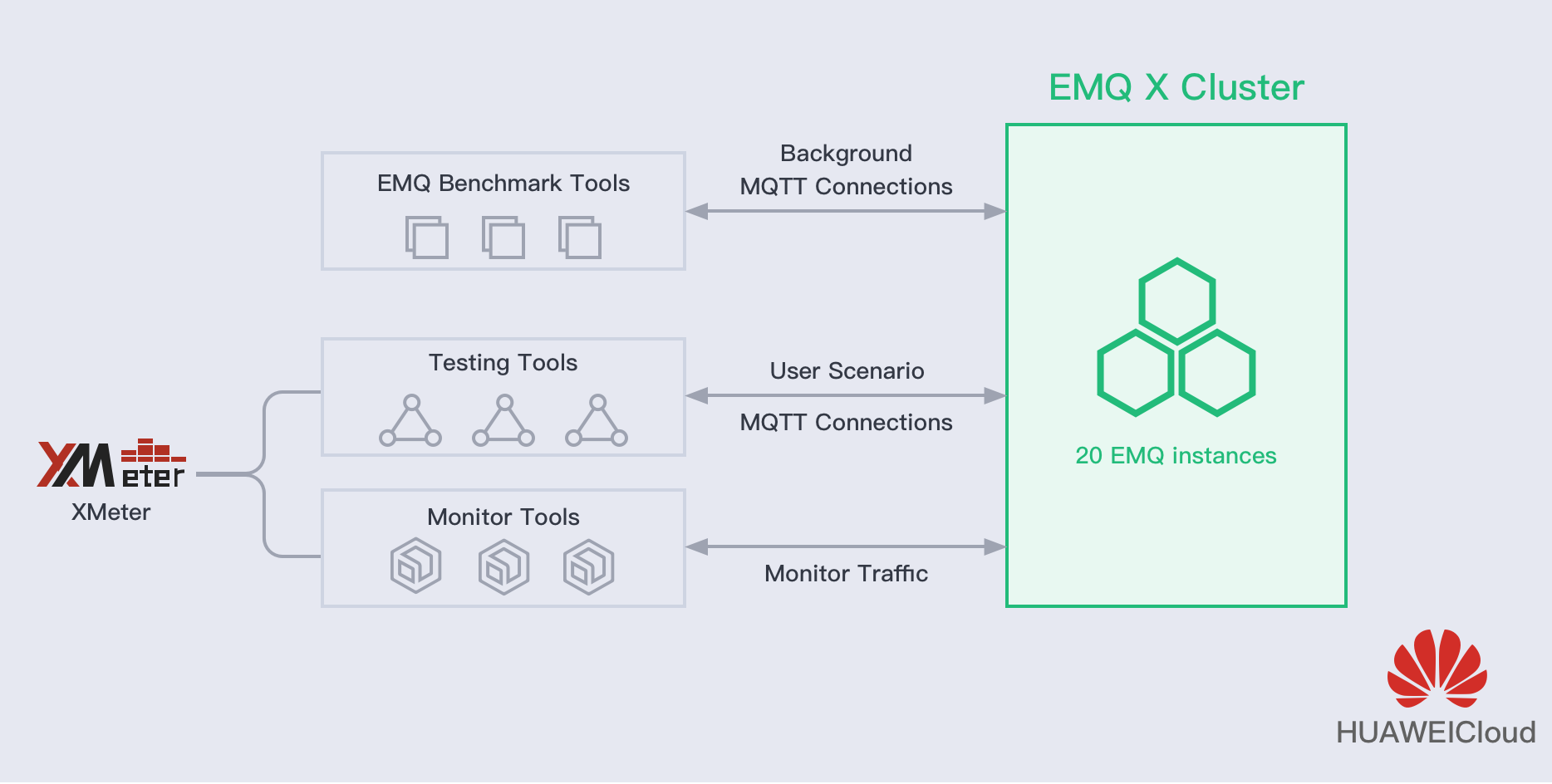
#### Pub: QoS 1 \* 2 k/秒 + Sub: QoS 1 \* 90万/秒

* 连接
  + 900万背景连接
  + 10万 Pub 端连接
  + 90万 Sub 端连接
* Pub: 2千/秒，每客户端间隔50秒Pub一次消息
* Sub: 90万/秒
* QoS: 1
* 主题数: 2k
* 消息体：88字节

## 测试环境

本册测试过程中由于资源协调的问题，中间调整过一次，因此有两套测试环境，本章后面将列出测试环境与测试场景的对应表。

华为云中的测试部署图如下所示，本次测试中使用了 EMQ Benchmark Tools 来模拟大量的 MQTT 连接；而 XMeter 提供的基于 JMeter MQTT 插件的测试工具来模拟业务测试场景；XMeter 内置支持的监控工具用于监控 20 台被测 EMQ X 集群的资源使用情况。



**服务器环境1**

* EMQ X版本: 3.1.1
* 被测试机（20台）
  + 操作系统：CentOS Linux release 7.4.1708
  + CPU：16核
  + 内存：32 G
  + 磁盘：120GB
* 压力机
  + 20台：用于执行Benchmark Tools模拟背景连接，每台主机10个网卡，每个网卡模拟5万Client连接EMQX
    - 操作系统：CentOS Linux release 7.4.1708
    - CPU：16核
    - 内存：32 G
    - 磁盘：120GB
  + 25台用于XMeter连接EMQX，模拟业务数据收发
    - 操作系统：CentOS Linux release 7.4.1708
    - CPU：8核
    - 内存：16 G
    - 磁盘：120GB
* 网络连接：华为云内网连接

此处列一下在此环境中测试的场景

千万连接测试

Pub消息测试

50万主题测试使用测试

**服务器环境2**

* EMQ X版本: 3.1.1
* 被测试机: 20台
  + 操作系统：CentOS Linux release 7.4.1708
  + CPU：8核
  + 内存：16 G
  + 磁盘：120GB
* 压力机： 70台
  + 操作系统：CentOS Linux release 7.4.1708
  + CPU：8核
  + 内存：16 G
  + 磁盘：120GB
* 网络连接：华为云内网连接

此处列一下在此环境中测试的场景

消息广播测试

EMQ 安装成功后，使用命令行显示的结果，总共安装了20台 EMQ X的一个集群。

|  |
| --- |
| Cluster status: [{running\_nodes, ['emqx@172.16.156.129', 'emqx@172.16.155.227', 'emqx@172.16.154.60', 'emqx@172.16.153.8', 'emqx@172.16.153.117', 'emqx@172.16.149.98', 'emqx@172.16.148.111', 'emqx@172.16.143.85', 'emqx@172.16.141.174', 'emqx@172.16.137.124' 'emqx@172.16.136.184', 'emqx@172.16.135.58', 'emqx@172.16.135.110', 'emqx@172.16.134.203', 'emqx@172.16.132.229', 'emqx@172.16.131.175', 'emqx@172.16.128.109', 'emqx@172.16.144.37', 'emqx@172.16.143.140', 'emqx@172.16.130.177']}] |

## 测试工具

* XMeter企业版2.0.1： <https://www.xmeter.net>

XMeter 是一个性能测试管理平台，基于开源的 JMeter 性能测试工具。XMeter可以支持大规模、高并发的性能测试，比如实现千万级别的MQTT 并发连接测试。除了测试 MQTT 协议之外，还可以支持 HTTP/HTTPS 等主流的应用的测试。

* JMeter-MQTT插件：mqtt-xmeter-1.13 – <https://github.com/emqx/mqtt-jmete>r

由 XMeter 实现的开源MQTT 性能测试插件，在众多的项目中得到了使用，目前是JMeter 社区中流行度最高的MQTT插件。

* JMeter版本: JMeter5.0 – https://jmeter.apache.org
* emqtt-benchmark-tools: <https://github.com/emqtt/emqtt_benchmark>

由EMQ 实现的用于模拟大量 MQTT 连接的测试工具。

## 测试结果

### 千万级连接测试

测试方法描述：TODO – 使用emqtt-benchmark-tools 的配置

emqtt\_bench\_conn -h ${emqx\_node\_ip} -c 50000 -n 1 -i 20 --ifaddr ${ips}

Usage: emqtt\_bench\_conn [--help <help>] [-h [<host>]] [-p [<port>]]

[-c [<count>]] [-n [<startnumber>]]

[-i [<interval>]] [-u <username>] [-P <password>]

[-k [<keepalive>]] [-C [<clean>]] [-S [<ssl>]]

[--certfile <certfile>] [--keyfile <keyfile>]

[--ifaddr <ifaddr>]

--help help information

-h, --host mqtt server hostname or IP address [default:

localhost]

-p, --port mqtt server port number [default: 1883]

-c, --count max count of clients [default: 200]

-n, --startnumber start number [default: 0]

-i, --interval interval of connecting to the broker [default: 10]

-u, --username username for connecting to server

-P, --password password for connecting to server

-k, --keepalive keep alive in seconds [default: 300]

-C, --clean clean session [default: true]

-S, --ssl ssl socoket for connecting to server [default: false]

--certfile client certificate for authentication, if required by

server

--keyfile client private key for authentication, if required by

server

--ifaddr local ipaddress or interface address

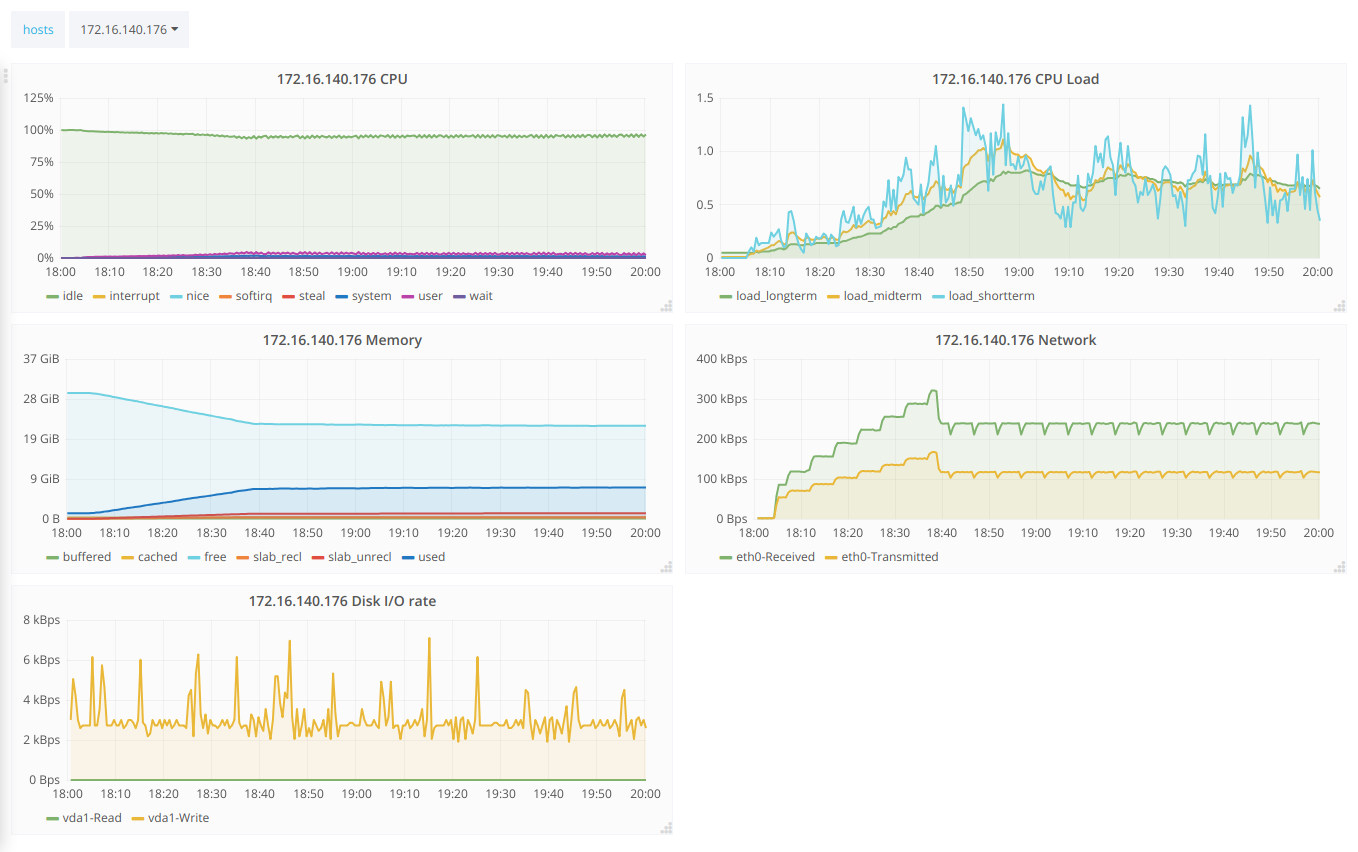
|  |  |
| --- | --- |
|  |  |

如上截图为成功达到了1000万连接后，EMQ X Dashboard 的连接截图。每台机器连接50万\*20台 = 1000万连接。

EMQ X 各个实例资源使用均衡，如下图所示是选择了其中二台机器的资源监控。 CPU、内存、磁盘和网络的使用情况都非常稳健。

被测系统报告解释（使用collectd+influxdb+grafana工具监控，以下测试报告中对机器监控的图表都类似，如果没有特别之处，我们不再赘述。）：

1. CPU 使用：监控项为idle,interrupt,nice,softirq,steal,system,user,wait;报告中数值均采用user项；
2. CPU 负载：load\_longterm,load\_midterm,load\_shorterm分别对应CPU负载15分钟，5分钟，1分钟；本报告中均采用1分钟值
3. 内存使用：监控项为buffered,cached,free,slab\_recl,slab\_ubrecl,used;本报告值均采用used
4. 网络使用：网卡收到与发出流量
5. 磁盘:基本为系统操作磁盘及日志操作磁盘，本报告不关注此项



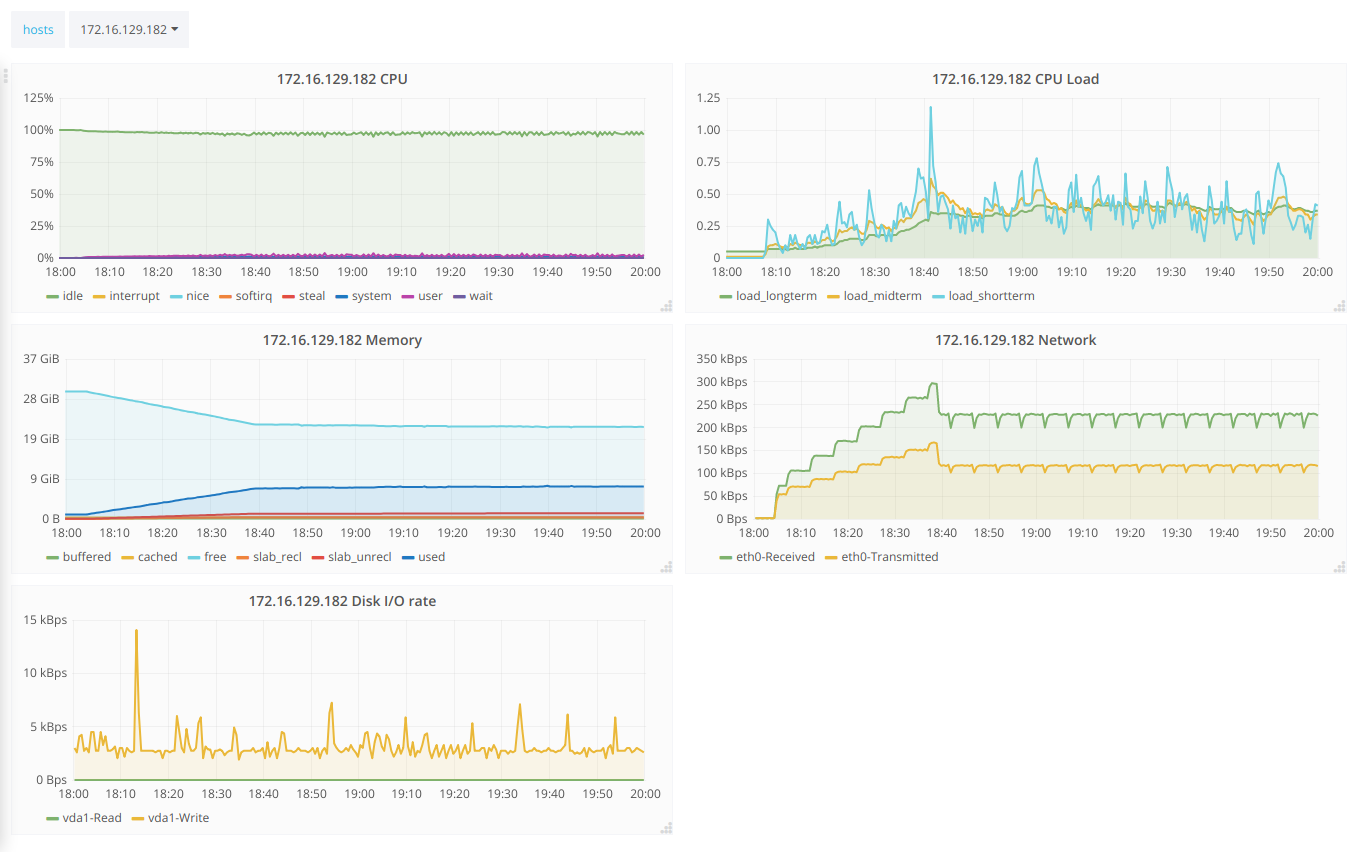
1

2

3

4

5



### Pub消息测试

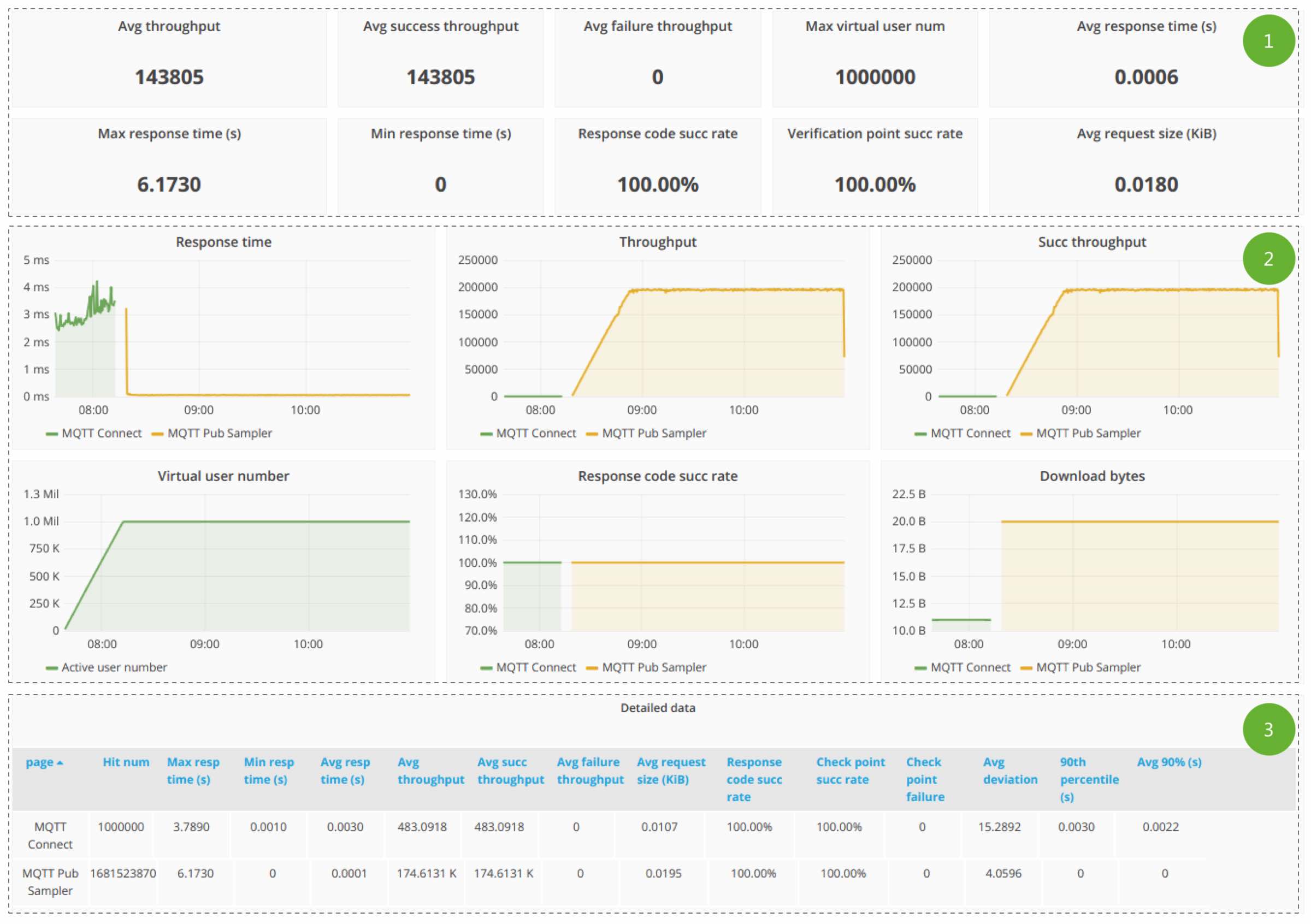
该测试类型下除了900万的背景连接，剩下的100万业务连接只做 pub 消息的操作。

#### QoS 0 \* 20万/秒

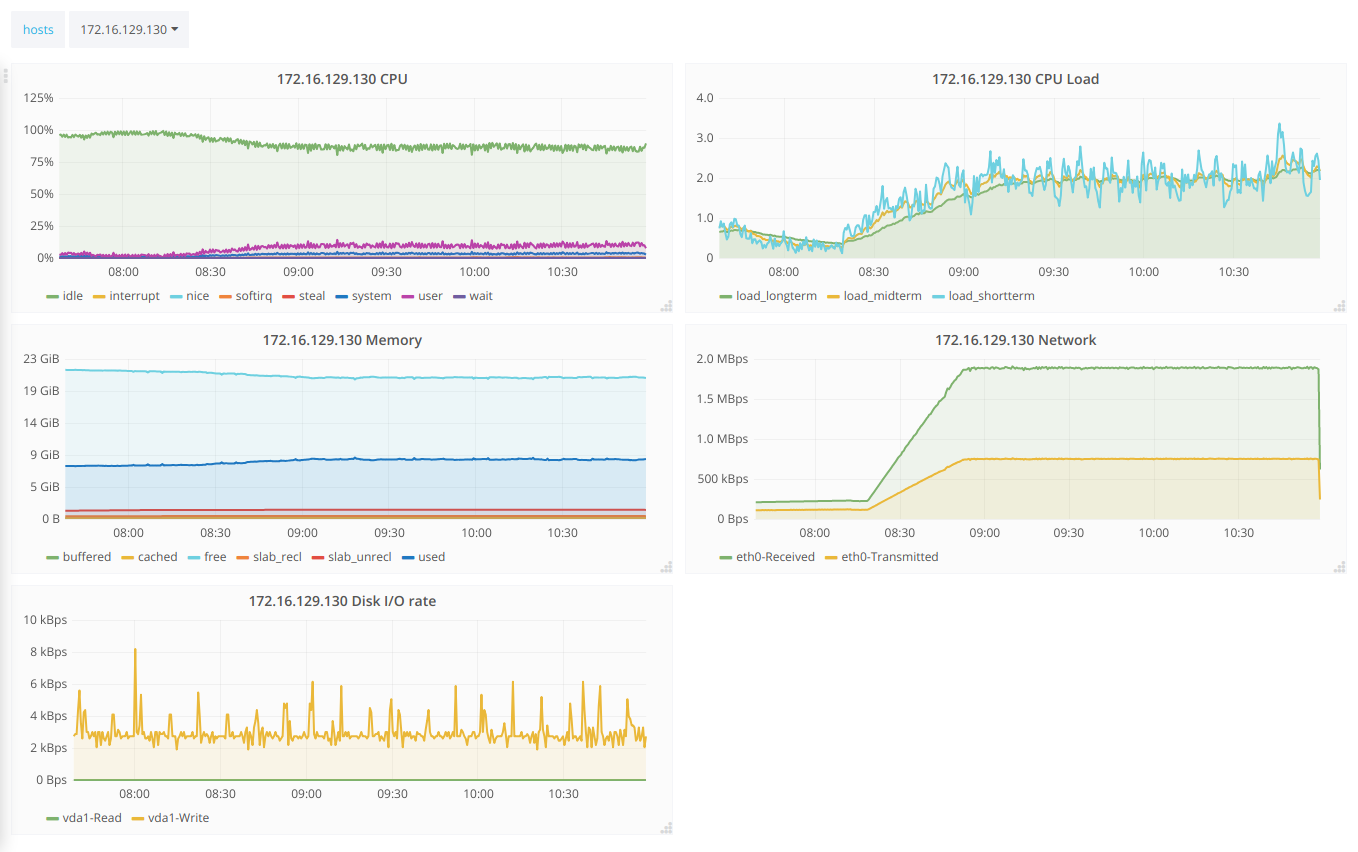
**报告解析**

如下图所示，XMeter 的性能测试报告分成三大部分。后面报告的结果解读方式都类似，如果没有特别之处，我们不再赘述。

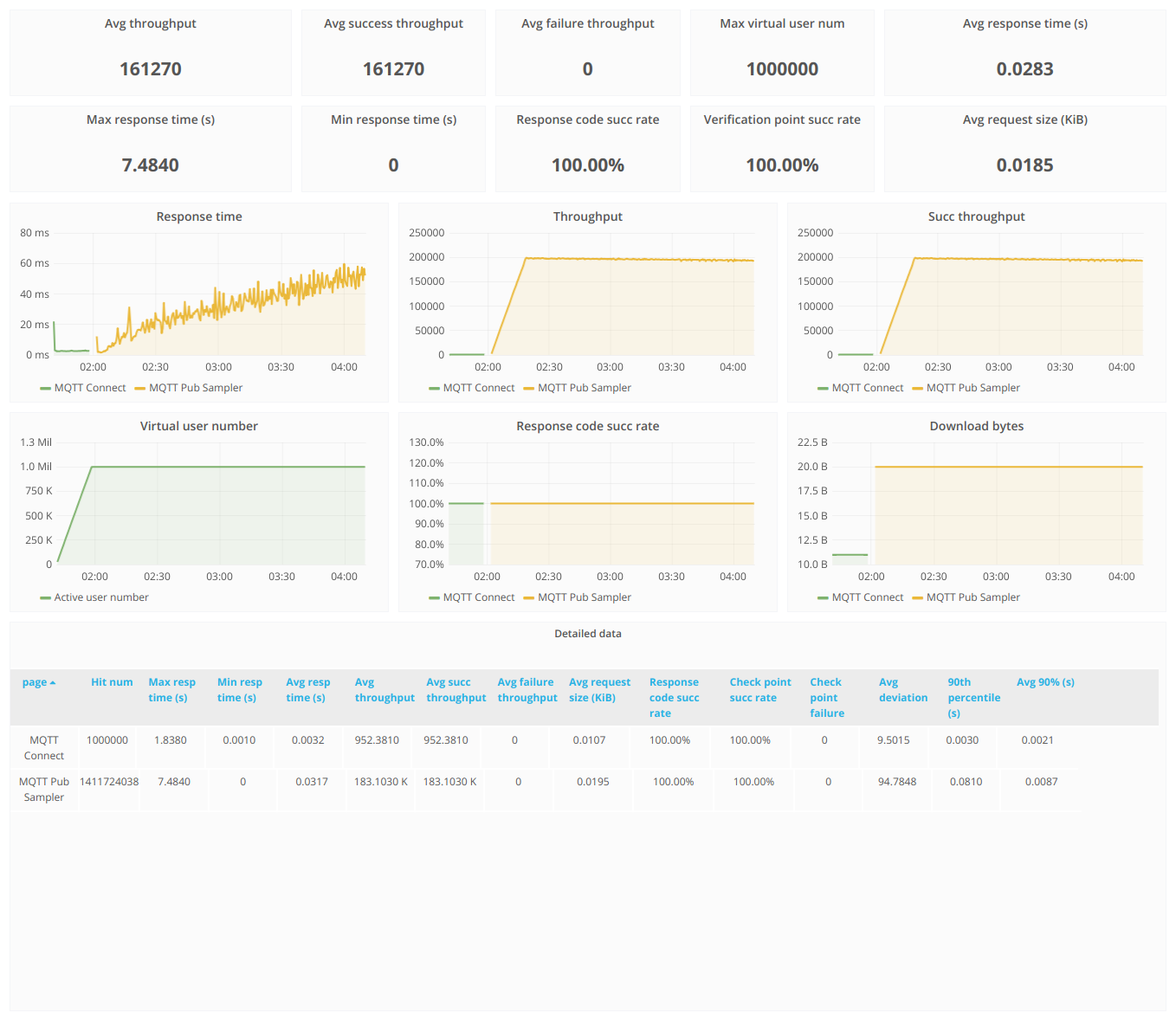
1. 概览
   * 1. Avg throughput 所有请求平均吞吐量
     2. Avg success throughput 所有请求平均成功吞吐量
     3. Avg failed throughput 所有请求平均失败吞吐量
     4. Max virtual user num 最大虚拟用户数
     5. Avg response time(s) 所有请求平均响应时间
     6. Max response time(s) 所有请求最大响应时间
     7. Min response time(s) 所有请求最小请求时间
     8. Response code succ rate 所有请求成功率
     9. Verification point succ rate 所有请求验证点成功率
     10. Avg request size(KiB) 所有请求平均大小
2. 图表区域
   * 1. Response time 每个请求响应时间
     2. Throughput 每个请求所有吞吐量
     3. Succ Throughput 每个请求成功吞吐量
     4. Virtual User 虚拟用户数
     5. Response code succ rate 响应码成功率
     6. Download bytes 响应数据大小
3. 详细数据（Detailed data 表格）
   * 1. Page 请求名称
     2. Hit num 运行次数
     3. Max resp time (s) 最大响应时间
     4. Min resp time (s) 最小响应时间
     5. Avg resp time(s) 平均响应时间
     6. Avg throughput 所有平均吞吐量
     7. Avg succ throughput 成功平均吞吐量
     8. Avg failure throughput 失败平均吞吐量
     9. Avg request size(KiB) 平均大小
     10. Response code succ rate 响应码成功率
     11. Check point succ rate 验证点成功率
     12. Check point failure 验证点错误数
     13. Avg deviation 平均标准差
     14. 90th percentile(s) 90分位响应时间
     15. Avg 90%(s) 90%平均响应时间



如上图所示，脚本的运行过程是在一定时间内完成连接建立（如上图绿色折线“MQTT Connect”所示），然后开始进行消息发送。测试在进入20万/秒吞吐量之后，持续运行了2个小时。吞吐量非常稳定，所有响应都成功。

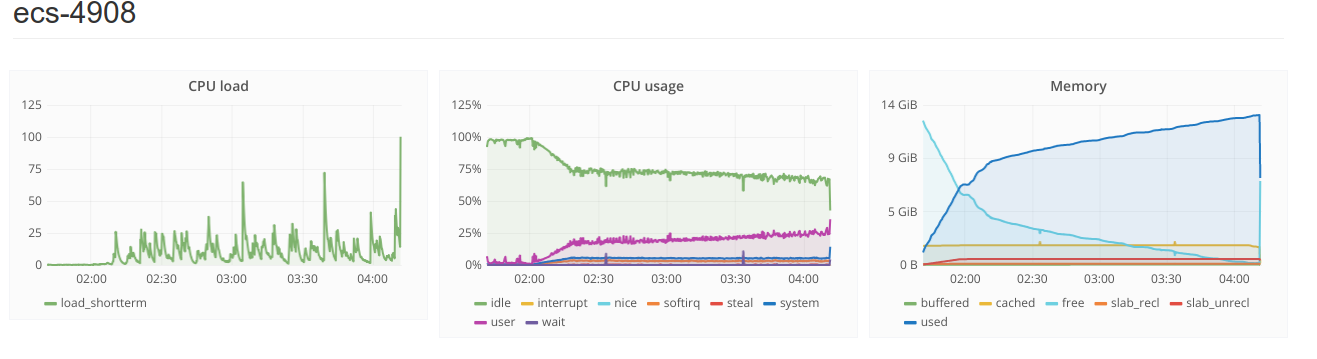


#### QoS 1 \* 20万/秒

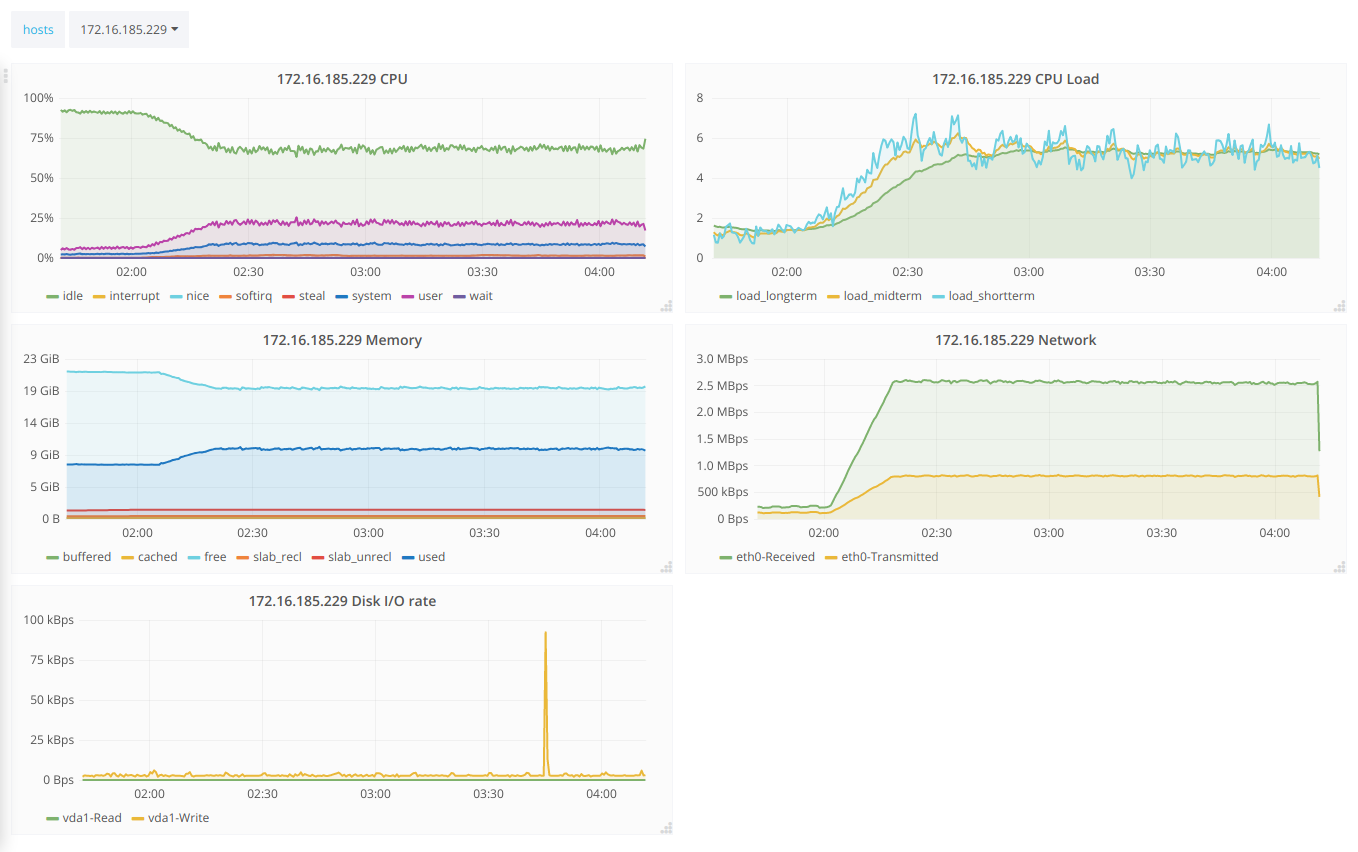


如上图所示，脚本的运行过程是在一定时间内完成连接建立（如上图绿色折线“MQTT Connect”所示），然后开始进行消息发送。测试在进入20万/秒吞吐量之后，持续运行了2个小时。吞吐量非常稳定，所有响应都成功。Qos1情况压力机处理ack等待线程较多压力机CPU负载较高，造成响应时间统计不够稳定。

压力机监控如下：



EMQX监控：



### 50万主题使用测试

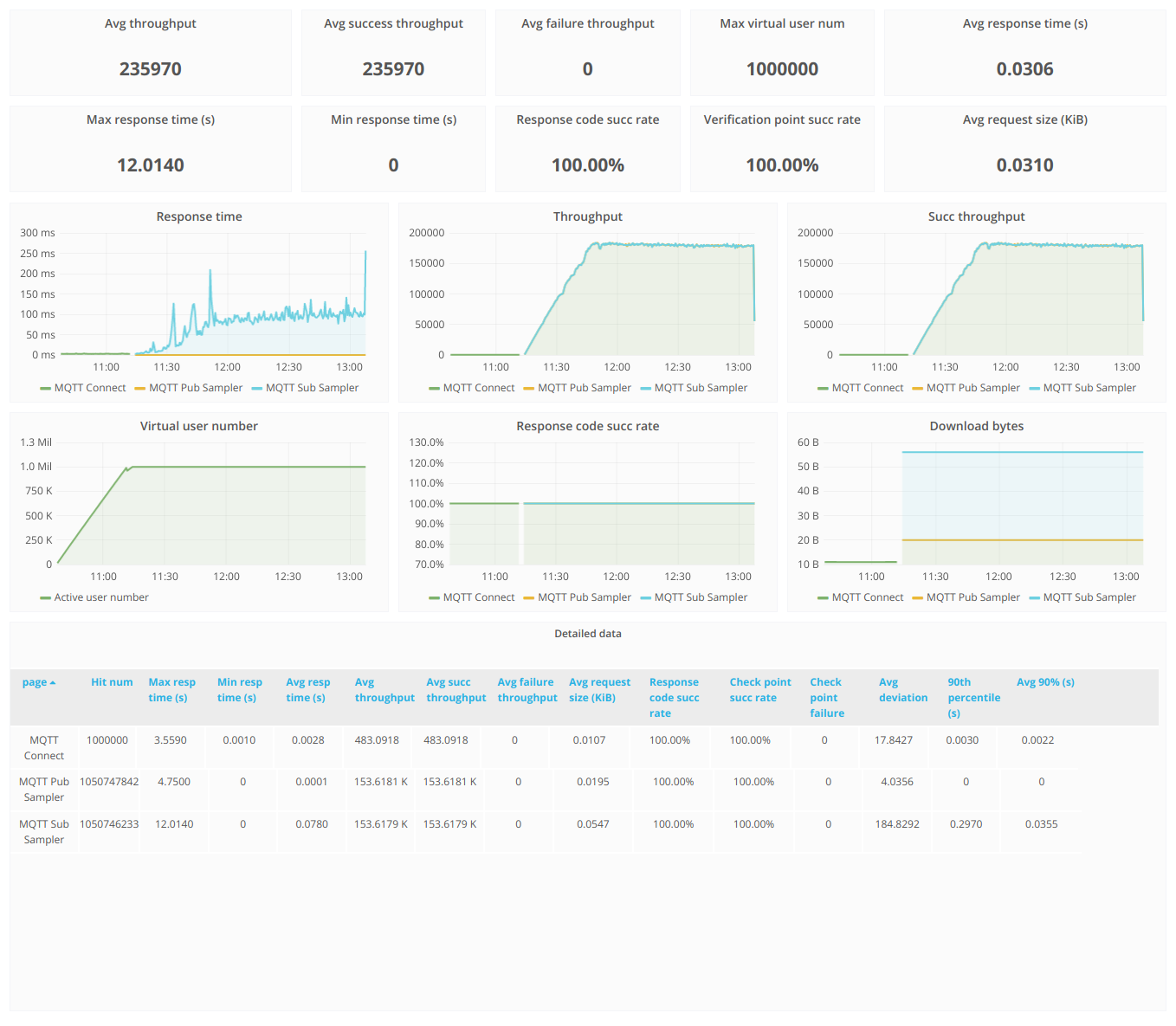
背景连接数：45万\*20台 = 900万

模拟Sub Client数：25000\*20台=50万

模拟Pub Client数：25000\*20台=50万

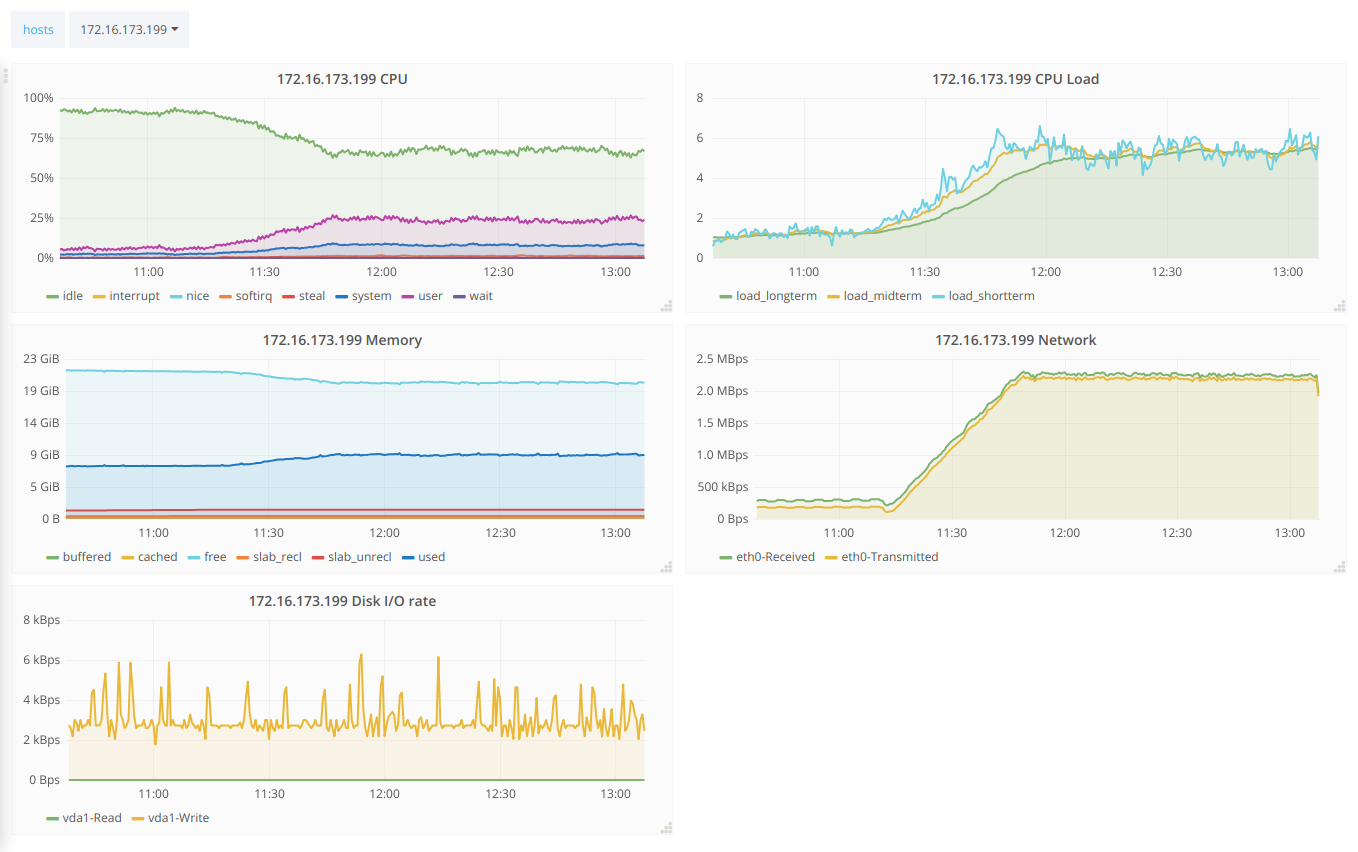
每个Sub Client订阅不同topic，同时有对应Pub Client，Topics=50万

#### Pub: QoS 0 \* 20 万/秒 + Sub: QoS 0 \* 20万/秒



如上图所示，脚本的运行过程是在一定时间内先完成连接建立，同时完成订阅，然后开始进行消息发送。测试在进入18万/秒Pub+18万/秒Sub之后，持续运行了1个小时。吞吐量非常稳定，所有响应都成功。

EMQX资源监控：

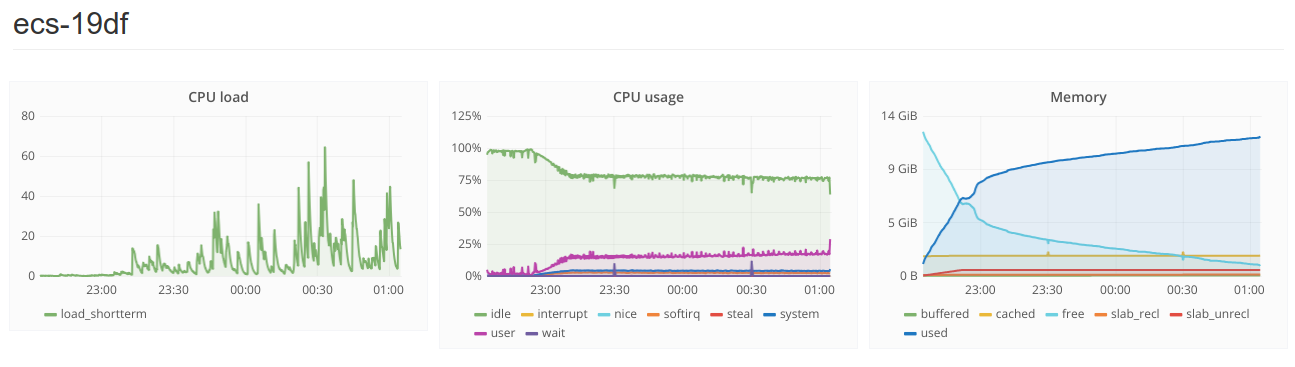


#### Pub: QoS 1 \* 10 万/秒 + Sub: QoS 1 \* 10万/秒

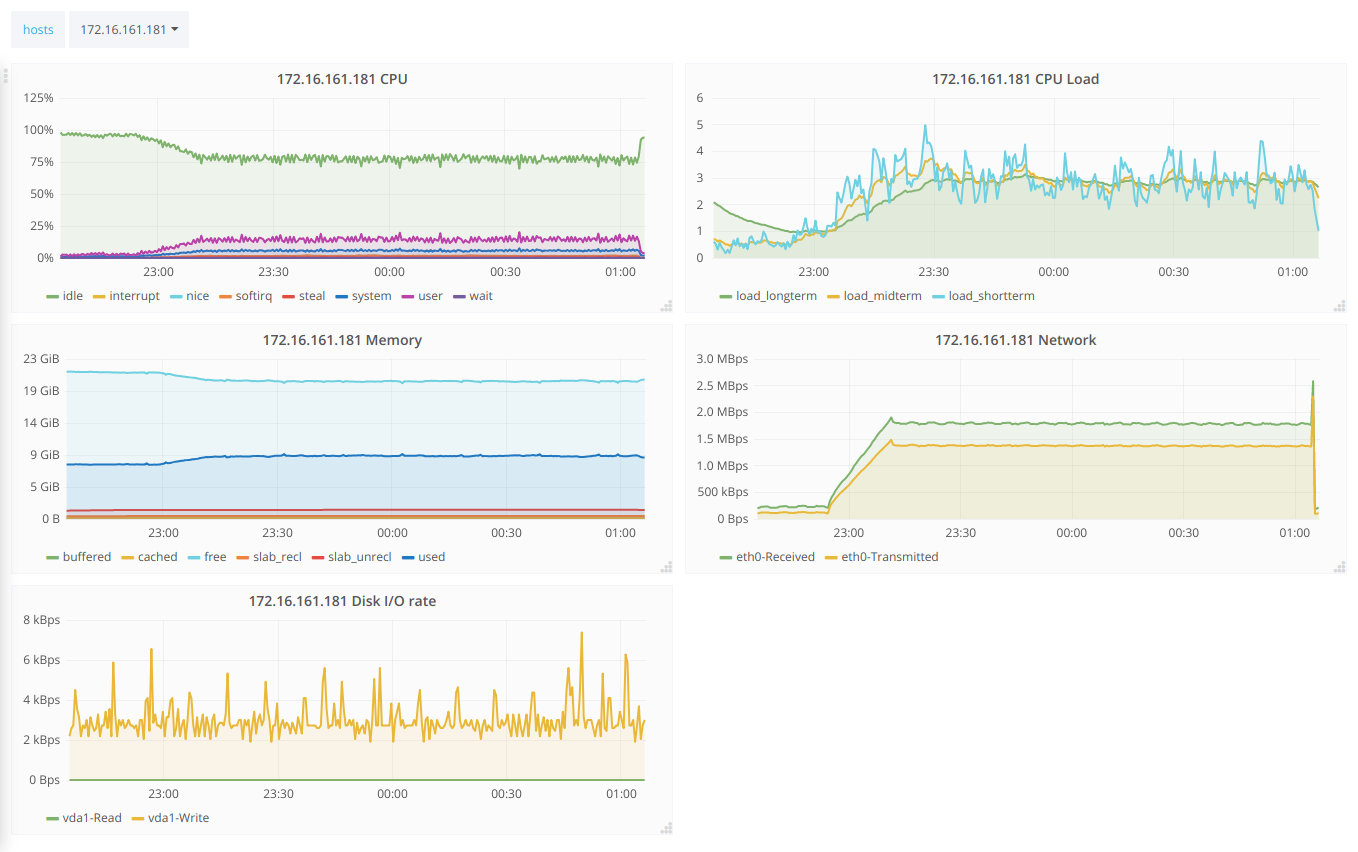


如上图所示，脚本的运行过程是在一定时间内先完成连接建立，同时完成订阅，然后开始进行消息发送。测试在进入10万/秒Pub+10万/秒Sub之后，持续运行了1个小时。吞吐量非常稳定，所有响应都成功。Qos1情况压力机处理ack等待线程较多压力机CPU负载较高，造成响应时间统计不够稳定。

压力机资源监控：



EMQX资源监控：



### 消息广播测试

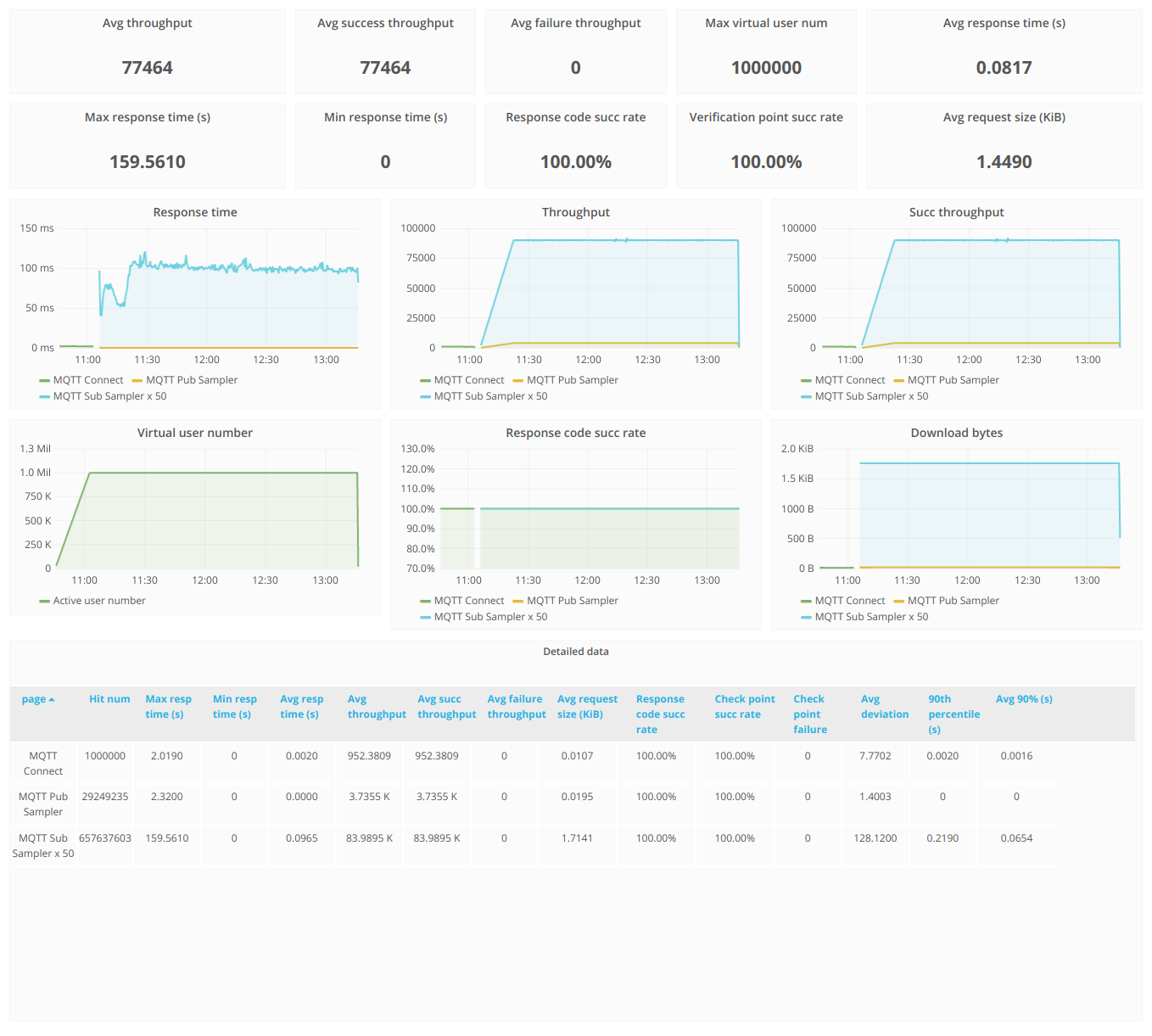
背景连接：45万\*20台=900万

Sub Client：45000\*20台=90万

Pub Client：5000\*20台=10万

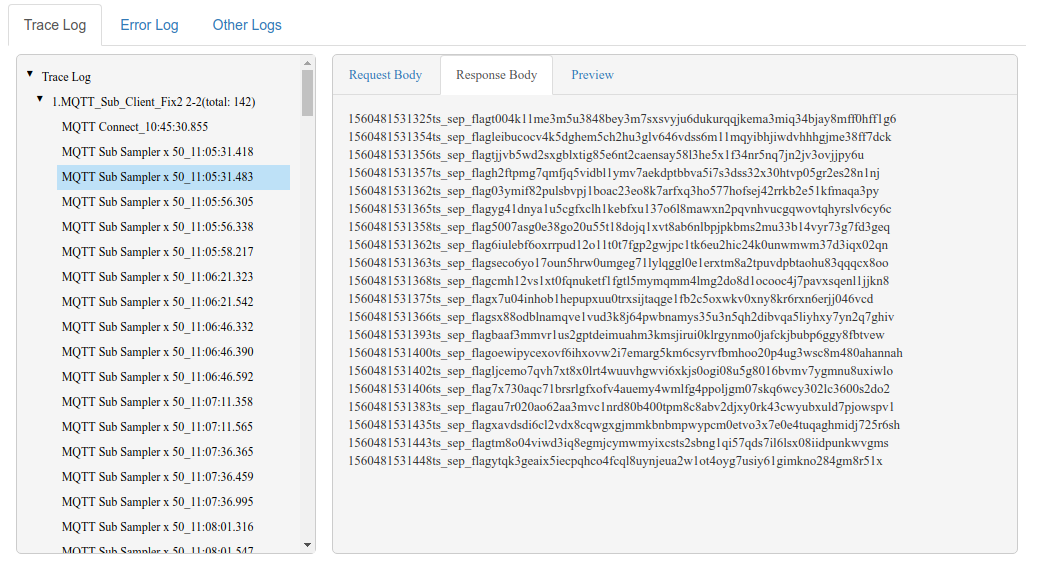
Sub Client 分别Sub 5000个topic,Pub client 分别pub 消息到5000个topic，设置不同Pub间隔时间控制吞吐量

#### Pub: QoS 0 \* 4 k/秒 + Sub: QoS 0 \* 180万/秒

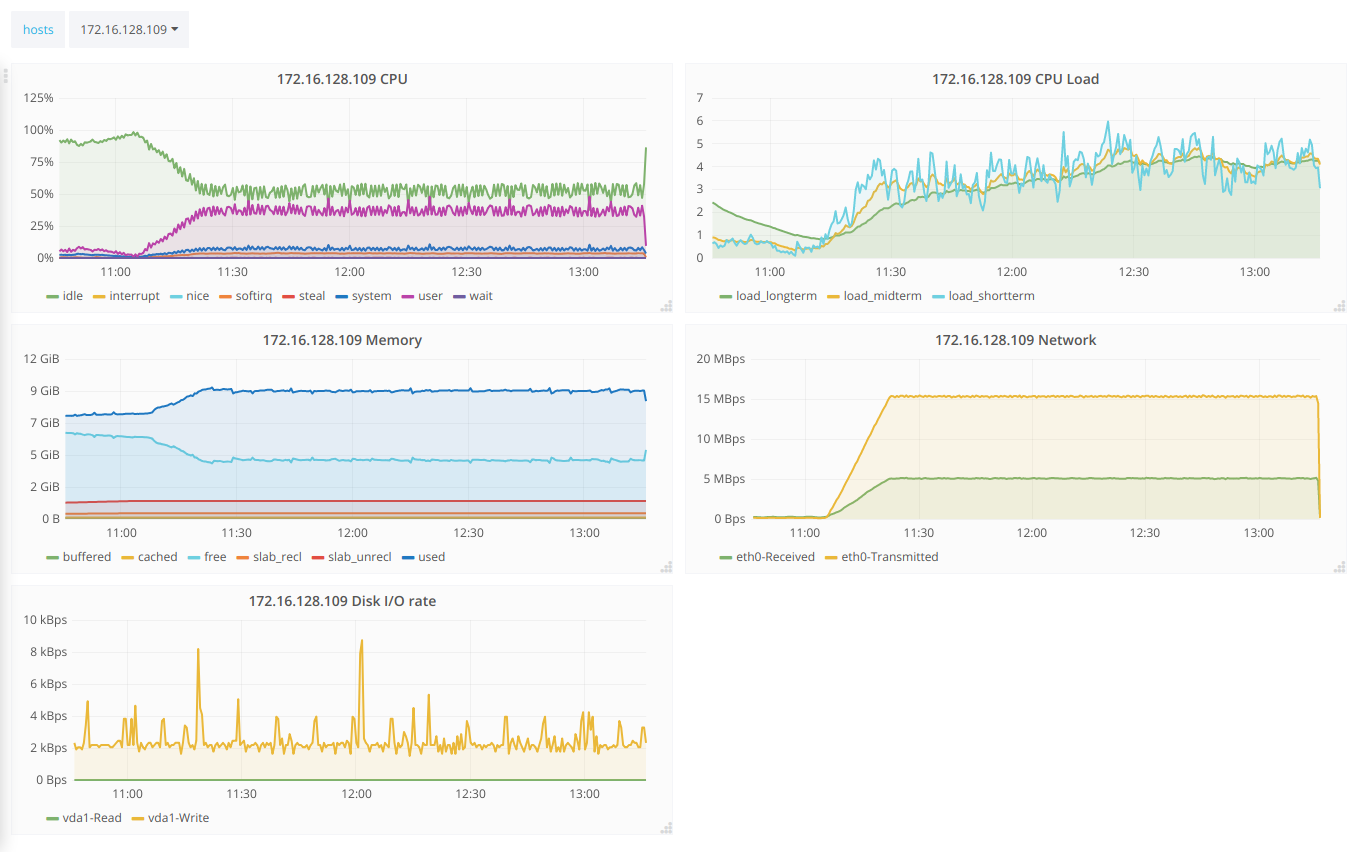


如上图所示，脚本的运行过程是在一定时间内先完成连接建立，同时完成订阅，然后开始进行消息发送。测试在进入4000/秒Pub+180万/秒Sub之后，持续运行了2个小时。吞吐量非常稳定，所有响应都成功。本测试Sub Sampler收到20次响应统计一次，图表显示吞吐量×20为实际吞吐量。

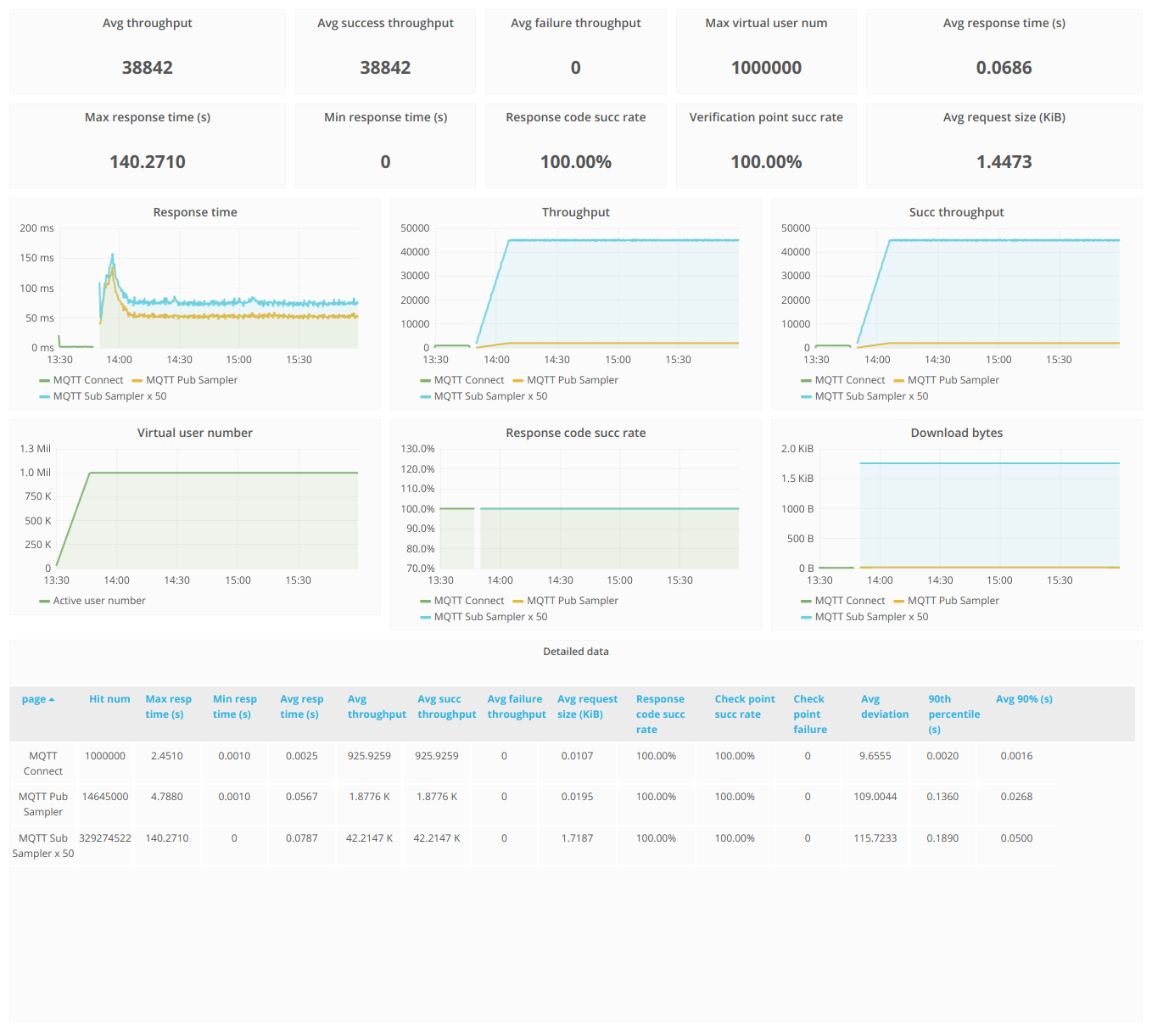
响应消息如下图：



EMQX资源监控：

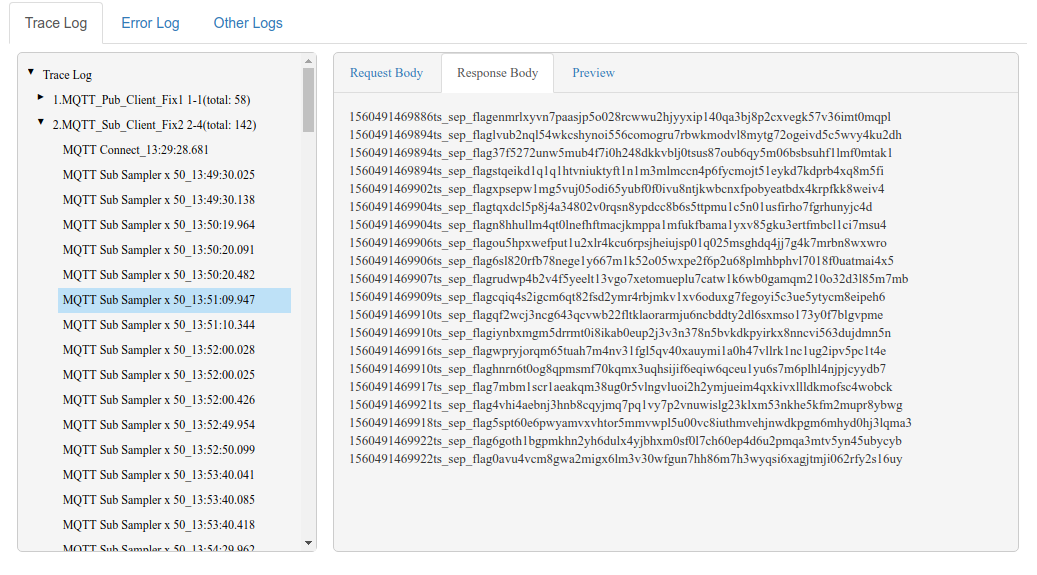


#### Pub: QoS 1 \* 2 k/秒 + Sub: QoS 1 \* 90万/秒

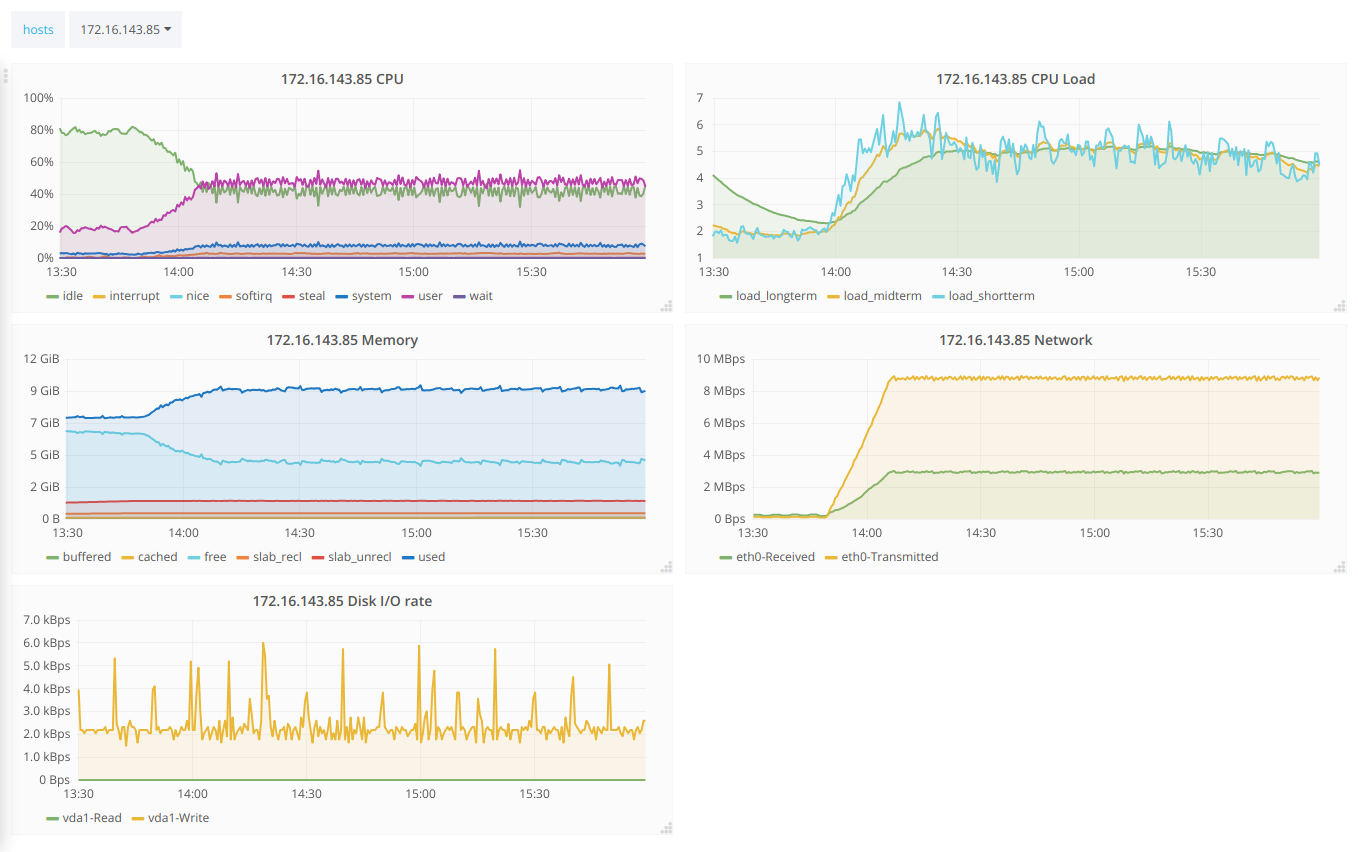


如上图所示，脚本的运行过程是在一定时间内先完成连接建立，同时完成订阅，然后开始进行消息发送。测试在进入2000/秒Pub+90万/秒Sub之后，持续运行了2个小时。吞吐量非常稳定，所有响应都成功。本测试Sub Sampler收到20次响应统计一次，图表显示吞吐量×20为实际吞吐量。

响应消息如下图：



EMQX资源监控：



## 附录

### 系统优化

参考：<https://developer.emqx.io/docs/broker/v3/en/tune.html>

### 测试脚本

测试脚本：script.zip – 上传到公开的 github 地址，以供下载