单体应用与微服务应用的效率对比

摘　要:　本实验旨在通过对比单体应用与微服务应用的效率，掌握微服务体系结构，并分析不同调用情况下的性能差异。实验在 Ubuntu 18.04 服务器上进行，涉及 JDK 11、Maven、git、Redis 6.2.4、JMeter 5.4.1 和 MySQL 9.0 等软件。通过 JMeter 测试 OOMALL product 模块中的 API，分析了不调用商铺模块、调用一次和调用两次商铺模块的性能数据。实验结果表明，微服务架构在横向扩展方面具有优势，但在调用次数增加时性能下降。
关键词:　微服务，单体应用，性能对比，横向扩展

Based on Mongo exp Plan

Abstract:　This experiment aims to compare the efficiency of monolithic and microservices applications, master the microservices architecture, and analyze performance differences under various calling conditions. The experiment was conducted on Ubuntu 18.04 servers, involving software such as JDK 11, Maven, git, Redis 6.2.4, JMeter 5.4.1, and MySQL 9.0. By testing the OOMALL product module's API with JMeter, we analyzed the performance data when not calling the shop module, calling once, and calling twice. The results indicate that microservices architecture has advantages in horizontal scaling but experiences performance degradation as the number of calls increases.
Key words:　Microservices, Monolithic Application, Performance Comparison, Horizontal Scaling

在现代软件开发中，微服务架构因其灵活性和可扩展性而受到青睐。然而，与单体应用相比，微服务架构在性能上的表现如何，尤其是在不同的调用情况下，是一个值得探讨的问题。本实验旨在通过对比单体应用和微服务应用在处理相同业务逻辑时的效率，来评估微服务架构的性能，并探讨其在实际应用中的适用性。

主要问题：

1.0 次、1 次、2 次调用商铺模块方案的效率差距

2.各方案的 CPU 和内存使用情况

1　使用 Jmeter 测试方案的速度差异

1.1　测试条件：

固定时间，保证相同线程与循环状态，进行比较

1.2　测试指标：

Response Times Over Times、Active Threads Over Times、Response Time Percentiles、平均响应时间

1.3　关键指标：

Response Time Percentiles，响应速度前 80%的请求所用的时间

2　使用华为云云监控服务监控服务器的负载情况

2.1　测试条件：

固定实验时间，通过华为云的云监控服务，比较相同线程与循环状态下 0 次、1 次、2 次调用商铺模块方案的服务器的负载情况

Github仓库链接：https://github.com/X-Believer/JavaEE-2-4/tree/master/Exp8

2.2 方案设计

（1）UnAuthorizedController 配置

```
@GetMapping("/products/{id}/{type}")  mingqiu@xmu.edu.cn <mingqiu@xmu.edu.cn> *
@Transactional(propagation = Propagation.REQUIRED)
public ReturnObject findProductById(@PathVariable("id") Long id, @PathVariable("type") Integer type)
    Product product = this.productService.findProductById(PLATFORM, id);
    return new ReturnObject(new ProductVo(product,type));
}
```

在原 API 后增加一个路径参数“type”，表示调用[type]次 shop 模块

（2）ProductVo 配置

```
public ProductVo(Product product, Integer type) {  2 个用法  mingqcn/mingqcn *
    super();
    CloneFactory.copy(this, product);
    if(type == 1 || type == 2) CloneFactory.copy(this, product.getValidOnsale());
    if(type == 2) this.setfreightTemplate(IdNameTypeVo.builder().id(product.getTemplate().getId()).name(
}
```

在 vo 对象构造函数中使用参数 type，设置不同[type]下使用的调用其他模块方法数

（3）运行效果

调用 0 次


```
C:\Users\ynkmx>curl 1.92.159.229:8081/products/3548/0
{"errmsg":"成功","data":{"id":3548,"name":"","status":1,"originalPrice":55430,"weight":200,"barcode":"6904032020071","un
it":"","originPlace":"","otherProduct":[{"id":1605,"name":"","price":13069,"status":1,"quantity":51},{"id":2263,"name":"
","price":9482,"status":1,"quantity":31},{"id":3548,"name":"","price":8939,"status":1,"quantity":81},{"id":4381,"name":"
","price":53946,"status":1,"quantity":44}],"shop":{"id":1,"name":"OOMALL"},"category":{"id":274,"name":""},"freeThreshol
d":-1},"errno":0}
```


调用 1 次，多查询到 actList（最后一行）


```
C:\Users\ynkmx>curl 1.92.159.229:8081/products/3548/1
{"errmsg":"成功","data":{"id":3548,"name":"","status":1,"originalPrice":55430,"weight":200,"barcode":"6904032020071","un
it":"","originPlace":"","otherProduct":[{"id":1605,"name":"","price":13069,"status":1,"quantity":51},{"id":2263,"name":"
","price":9482,"status":1,"quantity":31},{"id":3548,"name":"","price":8939,"status":1,"quantity":81},{"id":4381,"name":"
","price":53946,"status":1,"quantity":44}],"price":8939,"quantity":81,"maxQuantity":50,"beginTime":"2021-11-11T06:38:20"
,"endTime":"2027-02-19T06:38:20","shop":{"id":1,"name":"OOMALL"},"category":{"id":274,"name":""},"freeThreshold":-1,"act
List":[]},"errno":0}
```


调用 2 次，多查询到 freightTemplate（最后一行）


```
C:\Users\ynkmx>curl 1.92.159.229:8081/products/3548/2
{"errmsg":"成功","data":{"id":3548,"name":"","status":1,"originalPrice":55430,"weight":200,"barcode":"6904032020071","un
it":"","originPlace":"","otherProduct":[{"id":1605,"name":"","price":13069,"status":1,"quantity":51},{"id":2263,"name":"
","price":9482,"status":1,"quantity":31},{"id":3548,"name":"","price":8939,"status":1,"quantity":81},{"id":4381,"name":"
","price":53946,"status":1,"quantity":44}],"price":8939,"quantity":81,"maxQuantity":50,"beginTime":"2021-11-11T06:38:20"
,"endTime":"2027-02-19T06:38:20","shop":{"id":1,"name":"OOMALL"},"category":{"id":274,"name":""},"freeThreshold":-1,"act
List":[],"freightTemplate":{"id":1,"name":""}},"errno":0}
```

（4）日志输出

调用 0 次，只在 product 模块进行查询

```
2024-12-15 13:00:57.117 [http-nio-8080-exec-12] DEBUG c.e.x.o.product.dao.onsale.ValidOnSaleExecutor - execute: productId = 3548
2024-12-15 13:00:57.118 [http-nio-8080-exec-12] DEBUG cn.edu.xmu.oomall.product.dao.onsale.OnSaleDao - findLatestValidOnsale: id =3548
2024-12-15 13:00:57.119 [http-nio-8080-exec-12] DEBUG cn.edu.xmu.oomall.product.dao.onsale.OnSaleDao - findById: id =1999
2024-12-15 13:00:57.120 [http-nio-8080-exec-12] DEBUG cn.edu.xmu.javaee.core.aop.DaoAspect - doAround: obj = cn.edu.xmu.oomall.product.dao.on
sale.OnSaleDao@664129ab, method = findLatestValidOnsaleByProductId
2024-12-15 13:00:57.120 [http-nio-8080-exec-12] DEBUG cn.edu.xmu.oomall.product.dao.bo.Product - getValidOnsale: validOnsale = OnSale(super=O
OMallObject(id=1999, creatorId=1, creatorName=admin, modifierId=null, modifierName=null, gmtCreate=2021-11-11T06:38:20, gmtModified=null), pr
ice=8939, beginTime=2021-11-11T06:38:20, endTime=2027-02-19T06:38:20, quantity=81, maxQuantity=50, shopId=1, type=0, productId=3548, payTime=
null, deposit=0)
2024-12-15 13:00:57.120 [http-nio-8080-exec-12] DEBUG cn.edu.xmu.oomall.product.dao.bo.Product - getValidOnsale: validOnsale = OnSale(super=O
OMallObject(id=1999, creatorId=1, creatorName=admin, modifierId=null, modifierName=null, gmtCreate=2021-11-11T06:38:20, gmtModified=null), pr
ice=8939, beginTime=2021-11-11T06:38:20, endTime=2027-02-19T06:38:20, quantity=81, maxQuantity=50, shopId=1, type=0, productId=3548, payTime=
null, deposit=0)
2024-12-15 13:00:57.120 [http-nio-8080-exec-12] DEBUG cn.edu.xmu.oomall.product.dao.bo.Product - getValidOnsale: validOnsale = OnSale(super=O
OMallObject(id=1999, creatorId=1, creatorName=admin, modifierId=null, modifierName=null, gmtCreate=2021-11-11T06:38:20, gmtModified=null), pr
ice=8939, beginTime=2021-11-11T06:38:20, endTime=2027-02-19T06:38:20, quantity=81, maxQuantity=50, shopId=1, type=0, productId=3548, payTime=
null, deposit=0)
2024-12-15 13:00:57.123 [http-nio-8080-exec-12] DEBUG cn.edu.xmu.javaee.core.aop.ControllerAspect - doAround: jp = findProductById, code = OK
```

调用 1 次，跨模块查询到活动列表

```
2024-12-15 13:03:55.038 [http-nio-8080-exec-14] INFO  c.e.x.j.core.config.OpenFeignHeaderInterceptor - apply: feign interceptor.....
2024-12-15 13:03:55.038 [http-nio-8080-exec-14] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = host, values= [1.92.
159.229:8081]
2024-12-15 13:03:55.038 [http-nio-8080-exec-14] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = user-agent, values=
[curl/8.9.1]
2024-12-15 13:03:55.039 [http-nio-8080-exec-14] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = accept, values= [*/*
]
2024-12-15 13:03:55.039 [http-nio-8080-exec-14] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = Content-Type, values
= [application/json;charset=UTF-8]
2024-12-15 13:03:55.039 [http-nio-8080-exec-14] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - apply: add admin token...
2024-12-15 13:03:55.039 [http-nio-8080-exec-14] DEBUG cn.edu.xmu.javaee.core.util.JwtHelper - createToken:
2024-12-15 13:03:55.039 [http-nio-8080-exec-14] DEBUG cn.edu.xmu.javaee.core.util.JwtHelper - createToken: userId = 1 userName=admin departId
=0 tokenId:20241215130355005A
2024-12-15 13:03:55.040 [http-nio-8080-exec-14] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = authorization, value
s= [eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQiOjEsImRlcGFydElkIjowLCJ0b2tlbklkIjoiMjAyNDEyMTUxMzAzNTUwTzVBIiwidXNlck5hbWUiOiJhZG1pbiIs
InVzZXJMZXZlbCI6MSwiaXNzIjoiT09BRCIsInN1YiI6InRoaXMgaXMgYSB0b2tlbiIsImF1ZCI6Ik1JTklBUFAiLCJpYXQiOjE3MzQyNjc4MzUsImV4cCI6MTczNDI3MTQzNX0.r8jQh
U5qwL9oyC631DDlUriwn5kY6sZ8cu67_vNXWyU]
2024-12-15 13:03:55.044 [http-nio-8080-exec-14] DEBUG cn.edu.xmu.javaee.core.aop.DaoAspect - doAround: obj = cn.edu.xmu.oomall.product.dao.op
enfeign.ShopDao@72d9bd09, method = findById
2024-12-15 13:03:55.046 [http-nio-8080-exec-14] DEBUG cn.edu.xmu.javaee.core.aop.DaoAspect - doAround: obj = cn.edu.xmu.oomall.product.dao.ac
tivity.ActivityDao@274029ca, method = retrieveByOnsaleId
2024-12-15 13:03:55.046 [http-nio-8080-exec-14] DEBUG cn.edu.xmu.oomall.product.dao.bo.OnSale - getActList: actList = []
2024-12-15 13:03:55.047 [http-nio-8080-exec-14] DEBUG cn.edu.xmu.oomall.product.dao.bo.OnSale - getActList: actList = []
2024-12-15 13:03:55.048 [http-nio-8080-exec-14] DEBUG cn.edu.xmu.javaee.core.aop.ControllerAspect - doAround: jp = findProductById, code = OK
```

```
2024-12-15 13:13:08.628 [http-nio-8080-exec-22] DEBUG cn.edu.xmu.oomall.product.dao.bo.OnSale - getActList: actList = []
2024-12-15 13:13:08.629 [http-nio-8080-exec-22] DEBUG cn.edu.xmu.oomall.product.dao.bo.OnSale - getActList: actList = []
2024-12-15 13:13:08.629 [http-nio-8080-exec-22] INFO  c.e.x.j.core.config.OpenFeignHeaderInterceptor - apply: feign interceptor.....
2024-12-15 13:13:08.629 [http-nio-8080-exec-22] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = host, values= [1.92.
159.229:8081]
2024-12-15 13:13:08.629 [http-nio-8080-exec-22] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = user-agent, values=
[curl/8.9.1]
2024-12-15 13:13:08.629 [http-nio-8080-exec-22] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = accept, values= [*/*
]
2024-12-15 13:13:08.629 [http-nio-8080-exec-22] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = Content-Type, values
= [application/json;charset=UTF-8]
2024-12-15 13:13:08.629 [http-nio-8080-exec-22] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - apply: add admin token...
2024-12-15 13:13:08.629 [http-nio-8080-exec-22] DEBUG cn.edu.xmu.javaee.core.util.JwtHelper - createToken:
2024-12-15 13:13:08.630 [http-nio-8080-exec-22] DEBUG cn.edu.xmu.javaee.core.util.JwtHelper - createToken: userId = 1 userName=admin departId
=0 tokenId:2024121513130860LA
2024-12-15 13:13:08.630 [http-nio-8080-exec-22] DEBUG c.e.x.j.core.config.OpenFeignHeaderInterceptor - addHeader: name = authorization, value
s= [eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQiOjEsImRlcGFydElkIjowLCJ0b2tlbklkIjoiMjAyNDEyMTUxMzEzMDg2MExBIiwidXNlck5hbWUiOiJhZG1pbiIs
InVzZXJMZXZlbCI6MSwiaXNzIjoiT09BRCIsInN1YiI6InRoaXMgaXMgYSB0b2tlbiIsImF1ZCI6Ik1JTklBUFAiLCJpYXQiOjE3MzQyNjgzODgsImV4cCI6MTczNDI3MTk4OH0.9hvB_
MxnYRW5panUTzUTmbNyHr19P_IkymDH5iSmmdE]
2024-12-15 13:13:08.634 [http-nio-8080-exec-22] DEBUG cn.edu.xmu.javaee.core.aop.DaoAspect - doAround: obj = cn.edu.xmu.oomall.product.dao.op
enfeign.TemplateDao@728ad254, method = findById
2024-12-15 13:13:08.637 [http-nio-8080-exec-22] DEBUG cn.edu.xmu.javaee.core.aop.ControllerAspect - doAround: jp = findProductById, code = OK
```

3 结果分析与讨论

3.1 0次调用-600-60-4

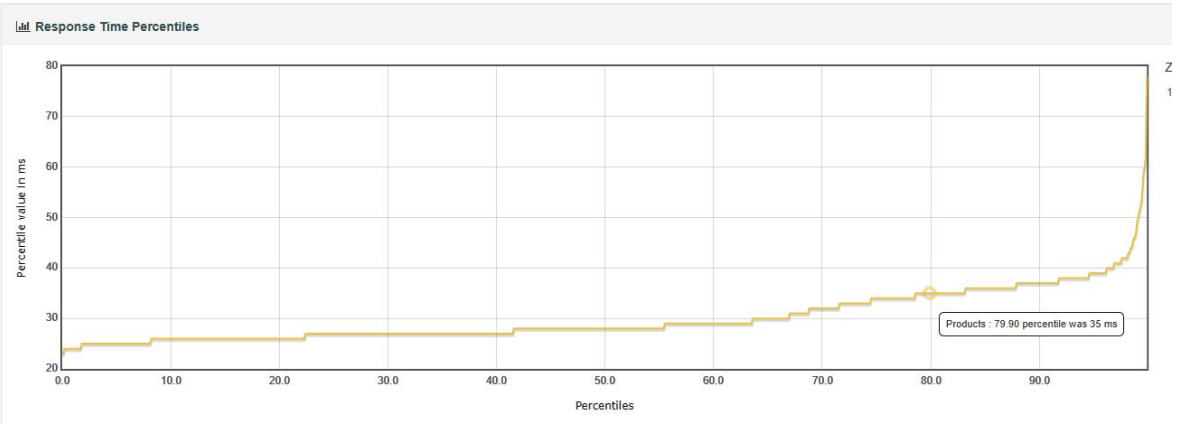

图 1 0 次调用-600-60-4 Response Time Percentiles

80%的请求在 35ms 内响应

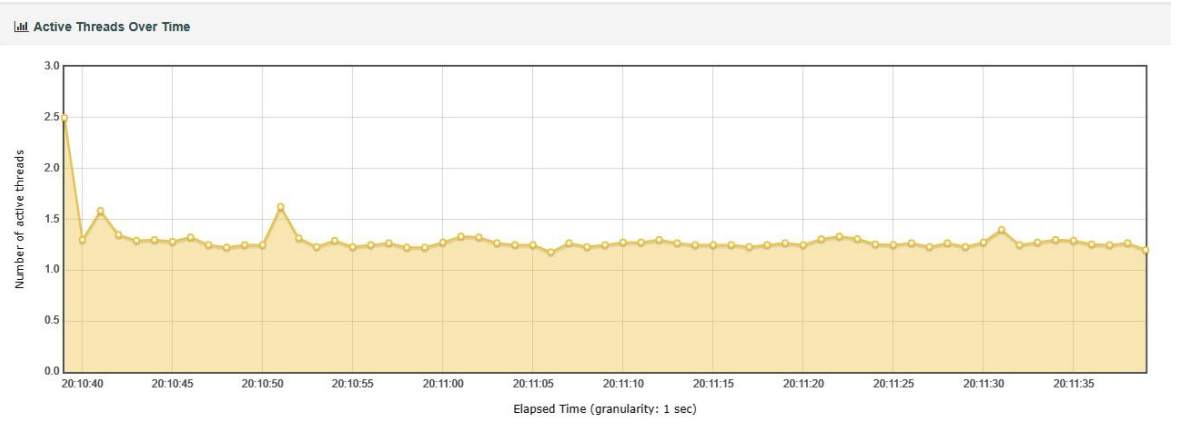

图 2 0 次调用-600-60-4 Active Threads Over Time

Github仓库链接：https://github.com/X-Believer/JavaEE-2-4/tree/master/Exp8

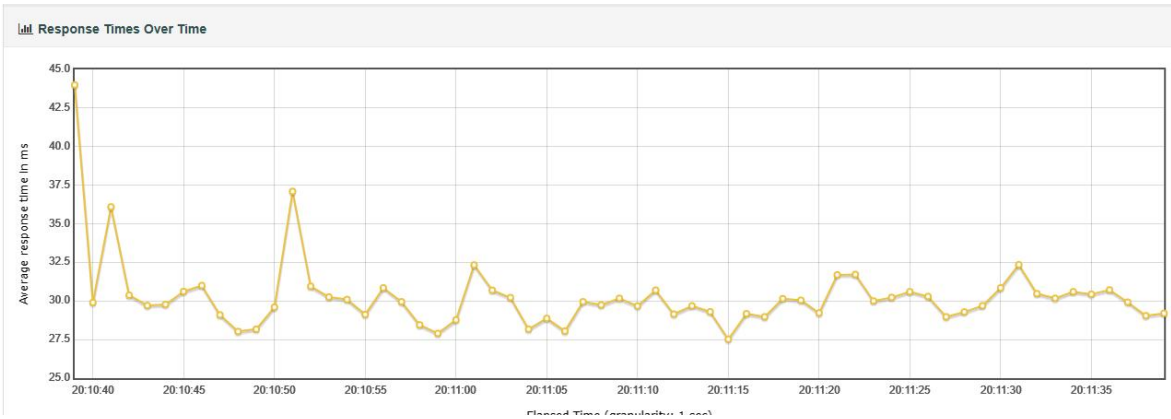

图 3 0 次调用-600-60-5 Response Times Over Time

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**2400**	**0**	**0.00%**	**30.18**	**23**	**88**	**28.00**	**37.00**	**39.00**	**49.00**	**40.09**	**49.76**	**7.09**
Products	2400	0	0.00%	30.18	23	88	28.00	37.00	39.00	49.00	40.09	49.76	7.09

图 4 0 次调用-600-60-5 Statistics

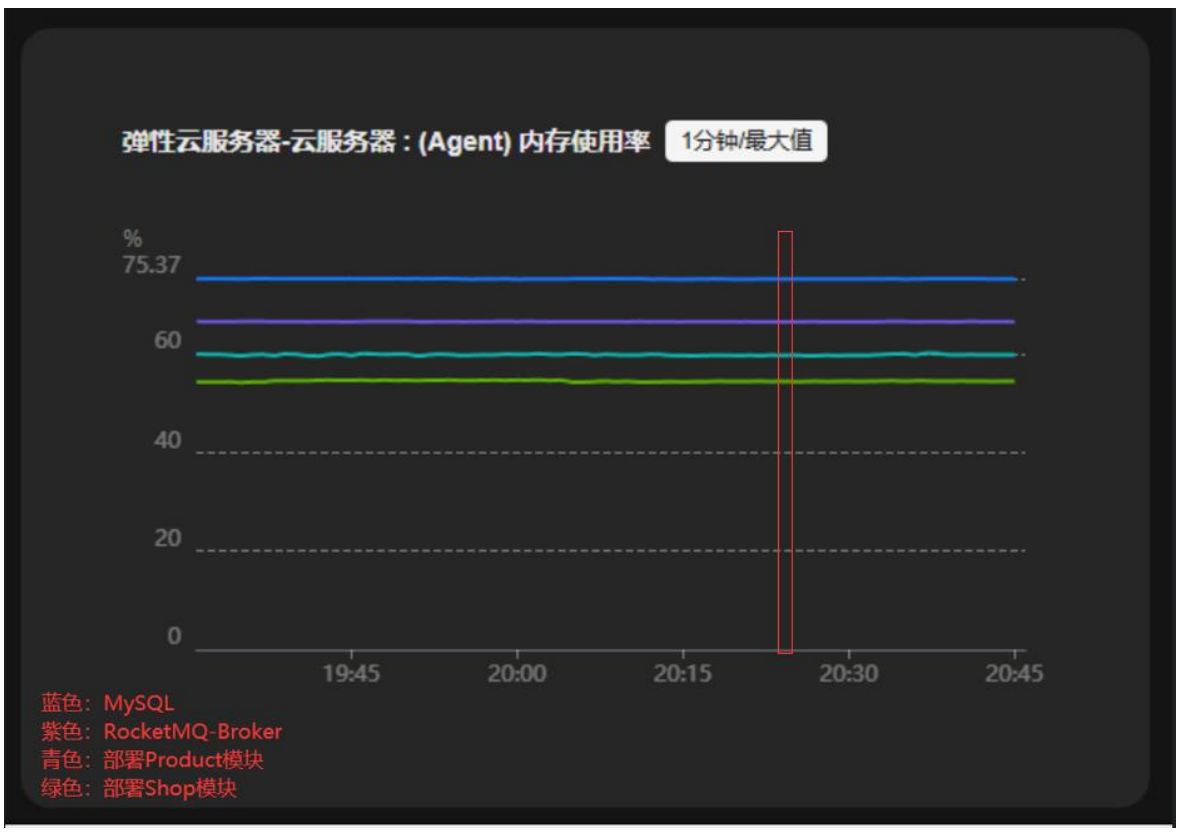

图 5 0 次调用-600-60-5 内存使用率

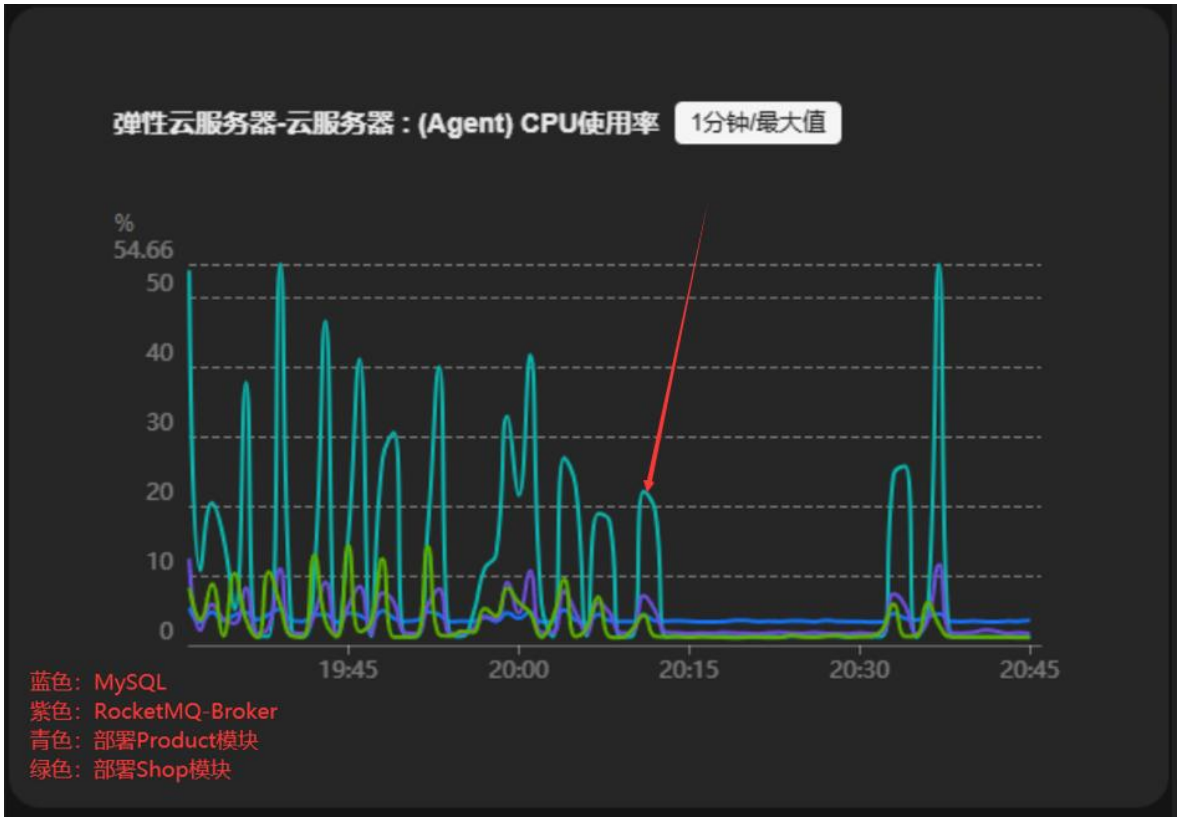

图 6 0 次调用-600-60-5 CPU 使用率

3.2 0次调用-600-60-5（60s 0次调用商铺模块达到的峰值）

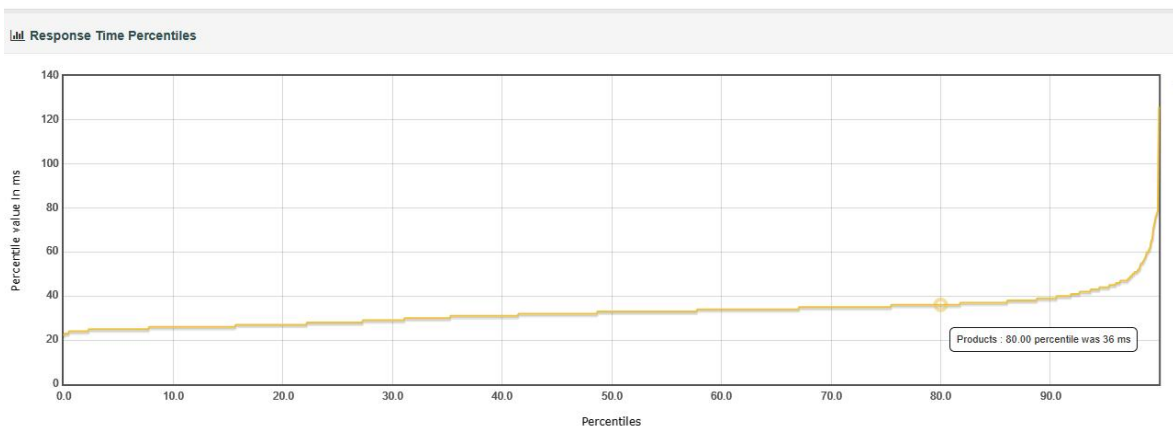

图 1 0 次调用-600-60-5 Response Time Percentiles

80%的请求在 36ms 内响应

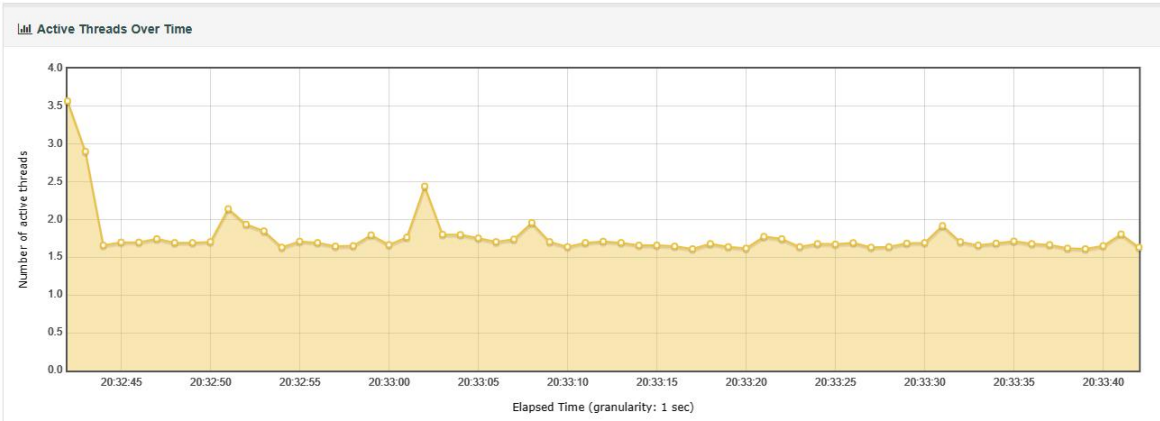

图 2 0 次调用-600-60-5 Active Threads Over Time

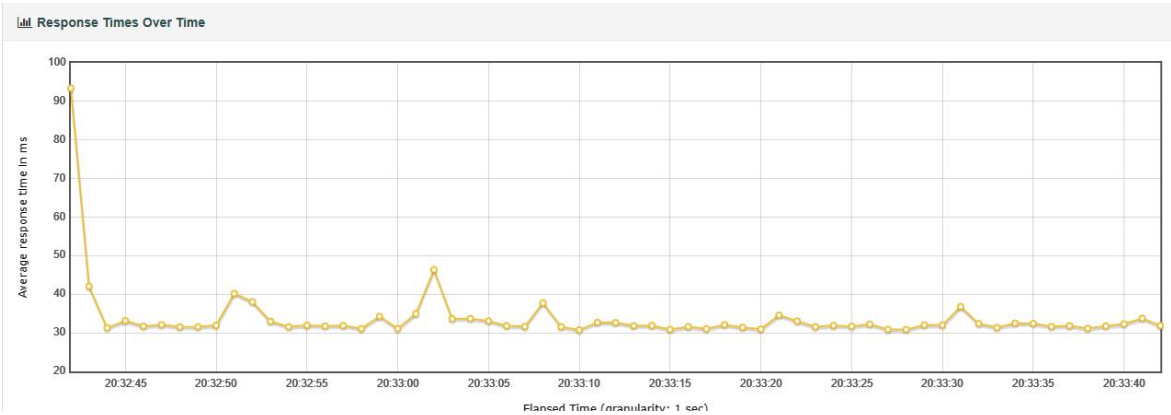

图 3 0 次调用-600-60-5 Response Times Over Time

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**3000**	**0**	**0.00%**	**33.06**	**22**	**172**	**33.00**	**39.00**	**44.00**	**61.00**	**50.06**	**62.14**	**8.85**
Products	3000	0	0.00%	33.06	22	172	33.00	39.00	44.00	61.00	50.06	62.14	8.85

图 4 0 次调用-600-60-5 Statistics

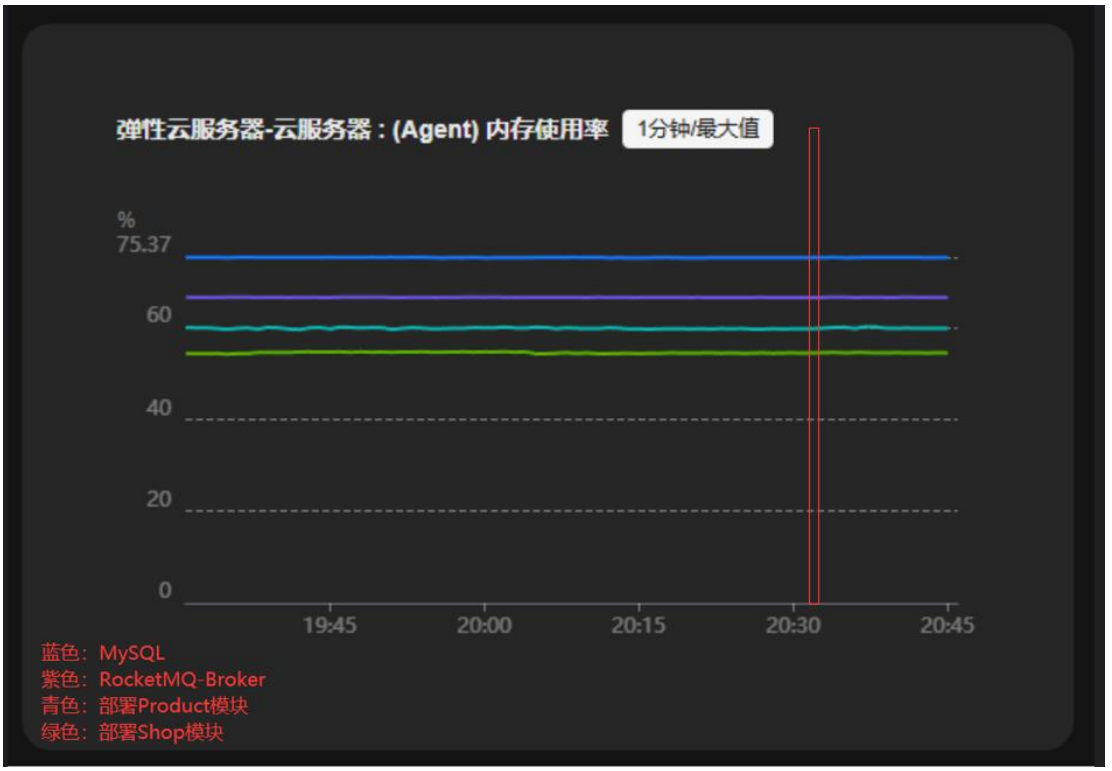

图 5 0 次调用-600-60-5 内存使用率

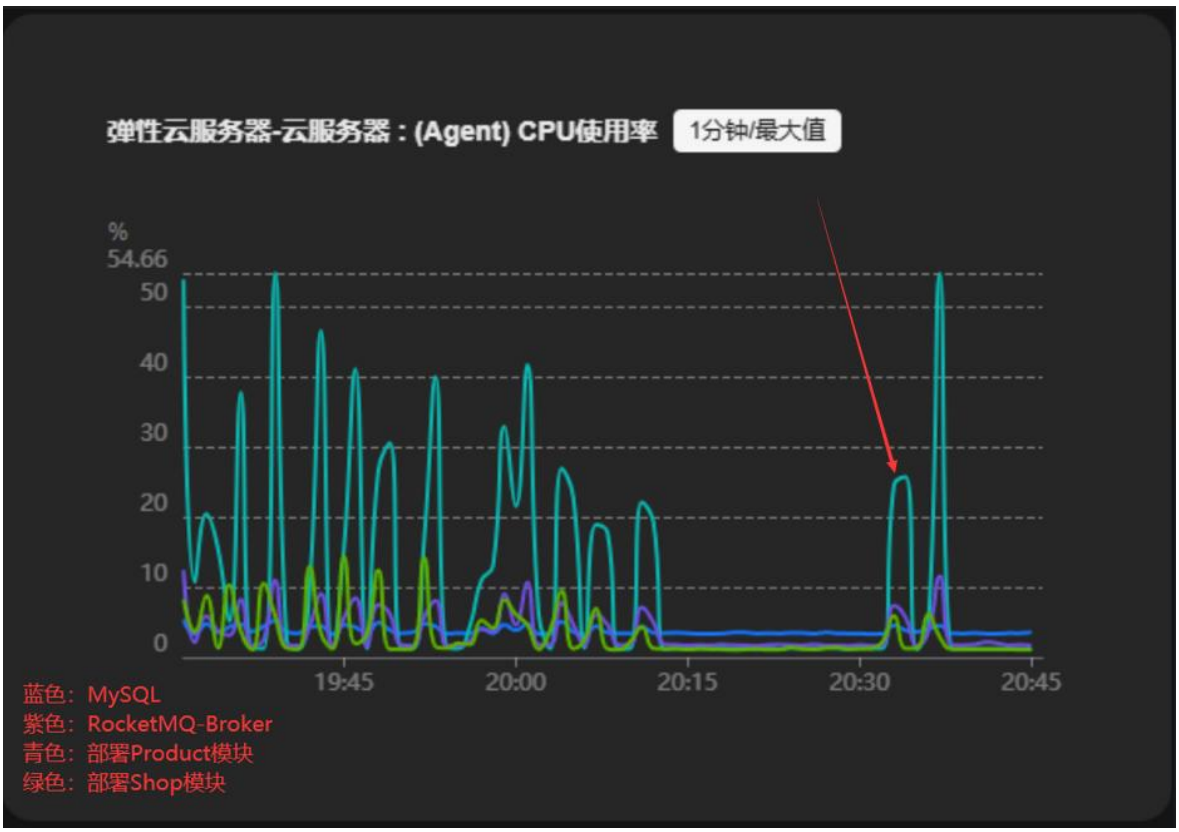

图 6 0 次调用-600-60-5 CPU 使用率

3.3 0次调用-600-60-6（60s 0次调用商铺模块阻塞）

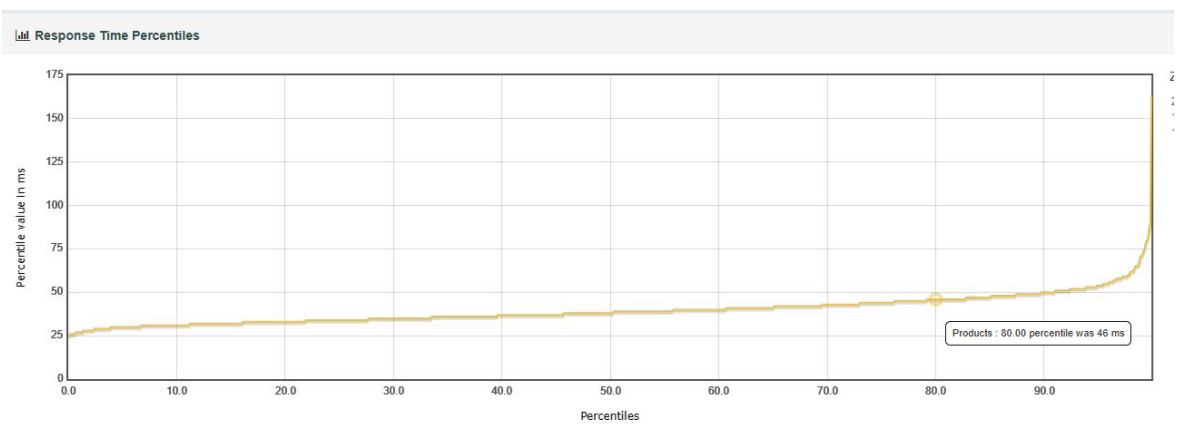

图 7 0 次调用-600-60-6 Response Time Percentiles

80%的请求在 46ms 内响应

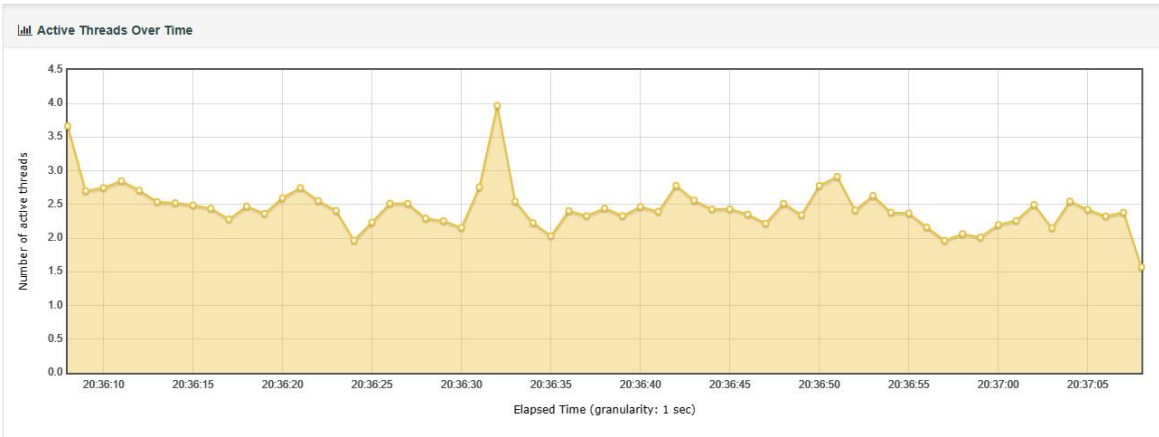

图 8 0 次调用-600-60-6 Active Threads Over Time

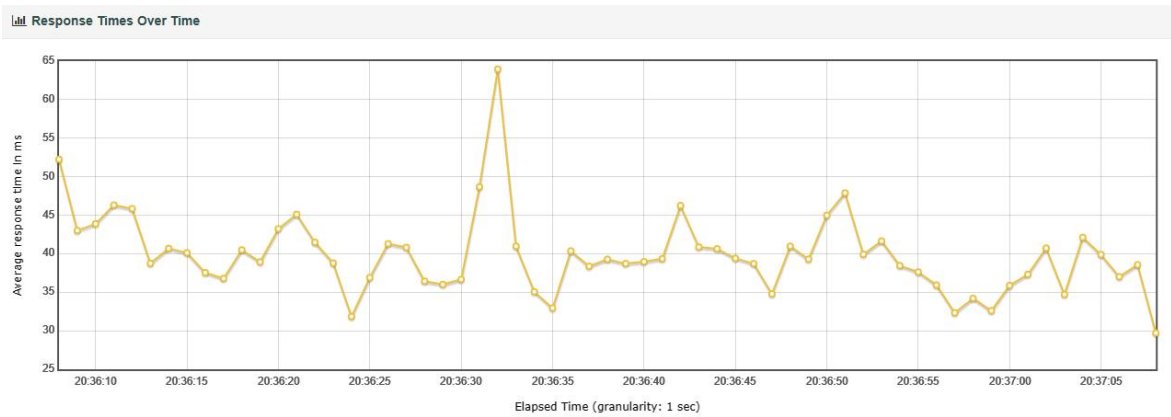

图 9 0 次调用-600-60-6 Response Times Over Time

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**3600**	**0**	**0.00%**	**40.03**	**25**	**214**	**38.00**	**50.00**	**54.00**	**71.00**	**60.09**	**74.59**	**10.62**
Products	3600	0	0.00%	40.03	25	214	38.00	50.00	54.00	71.00	60.09	74.59	10.62

图 10 0 次调用-600-60-6 Statistics

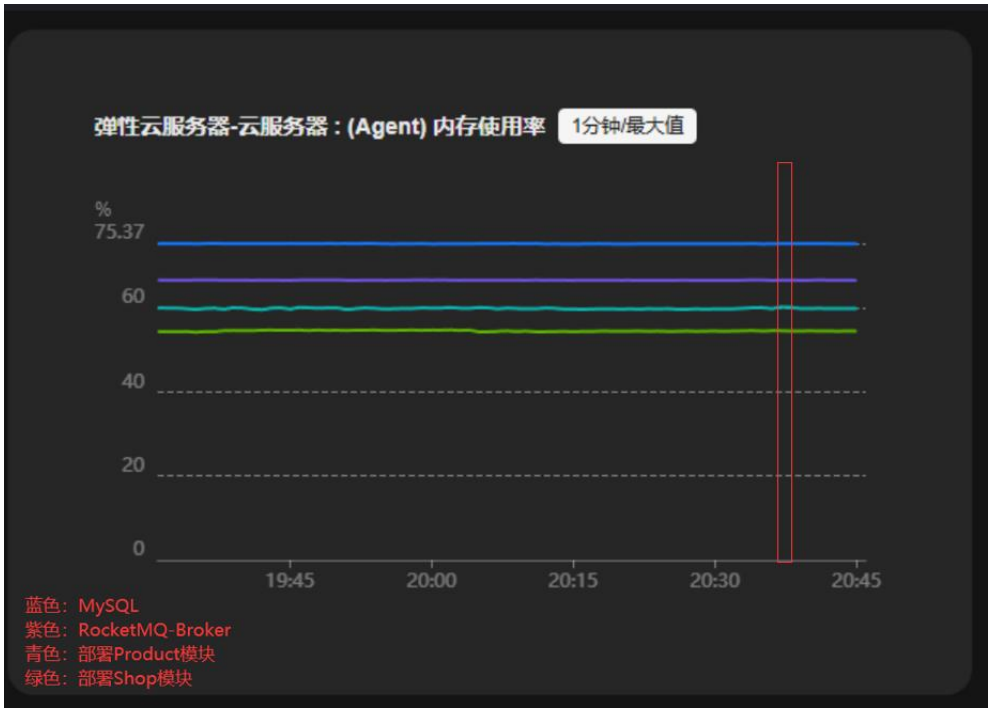

图 11 0 次调用-600-60-6 内存使用率

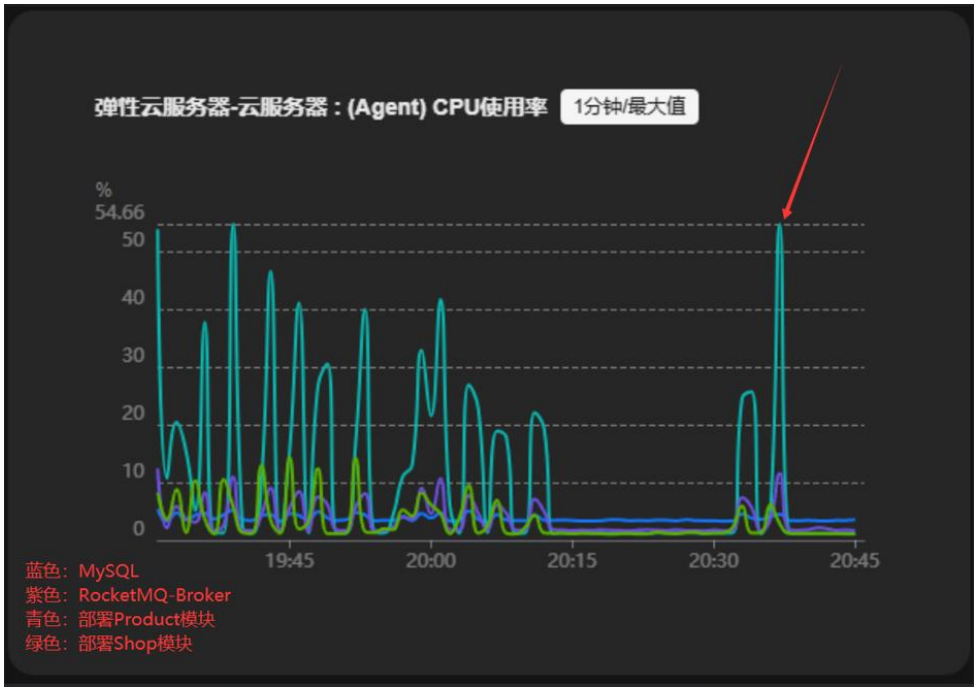

图 12 0 次调用-600-60-6 CPU 使用率

3.4 1次调用-600-60-3（60s 1次调用商铺模块达到的峰值）

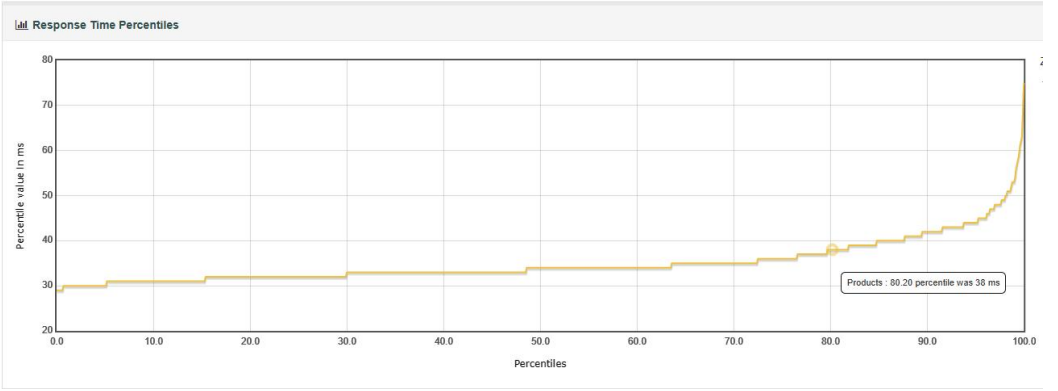

图 1 1 次调用-600-60-3 Response Time Percentiles

80%的请求在 38ms 内响应

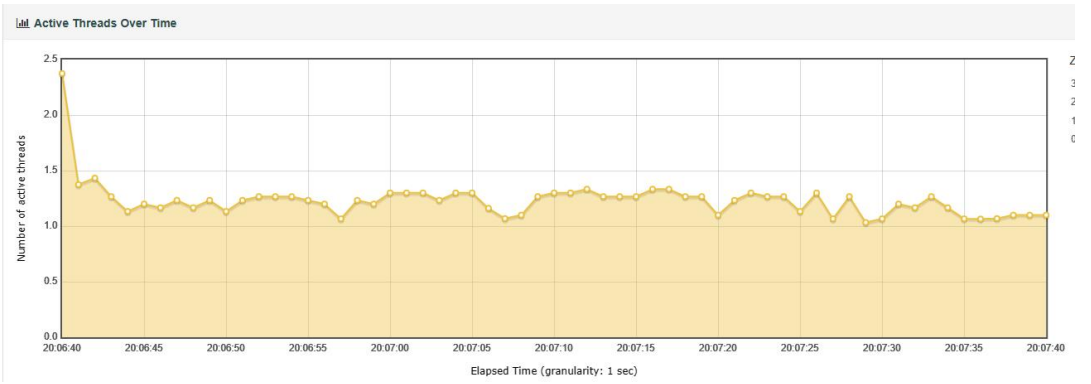

图 2 1 次调用-600-60-3 Active Threads Over Time

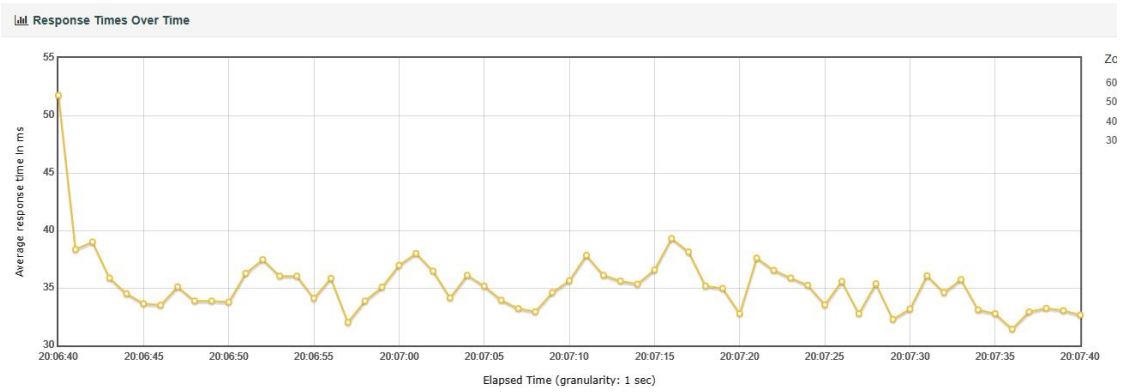

图 3 1 次调用-600-60-3 Response Times Over Time

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	1800	0	0.00%	35.10	29	75	34.00	42.00	44.00	53.00	30.07	41.58	5.32
Products	1800	0	0.00%	35.10	29	75	34.00	42.00	44.00	53.00	30.07	41.58	5.32

图 4 1 次调用-600-60-3 Statistics

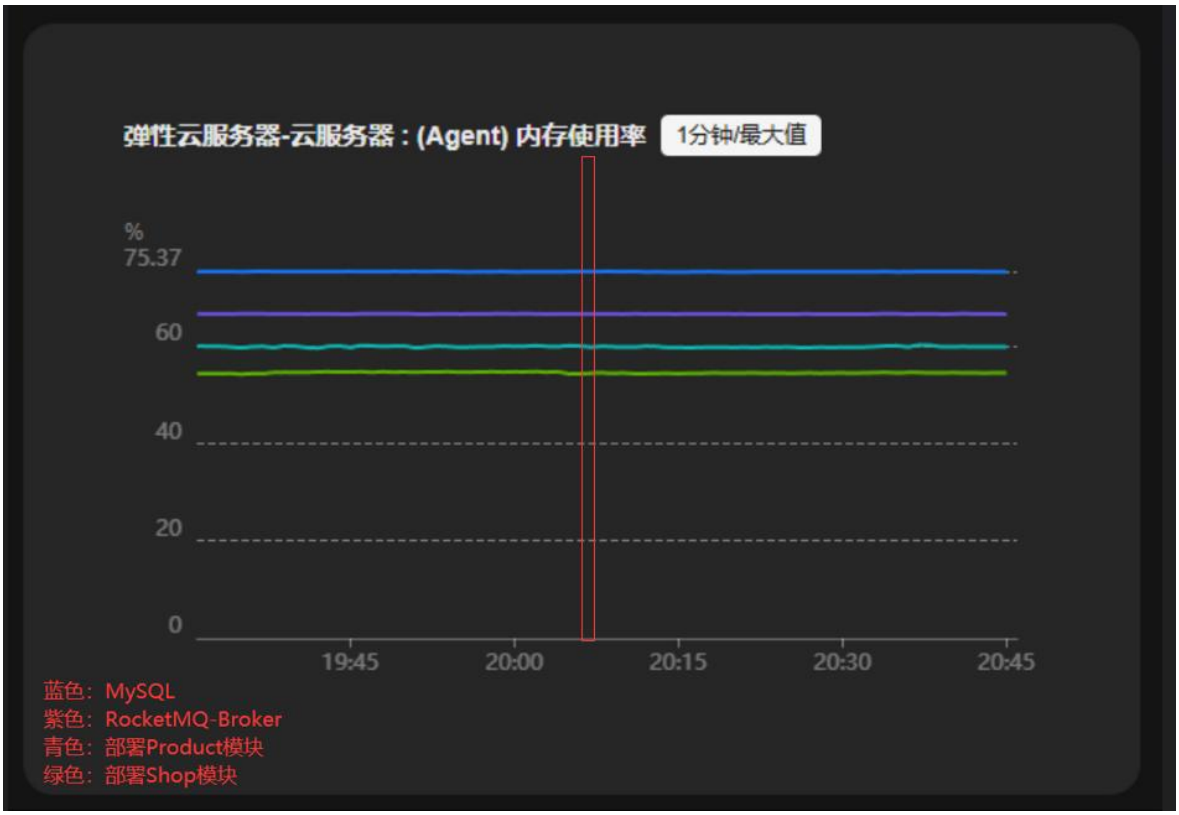

图 5 1 次调用-600-60-3 内存使用率

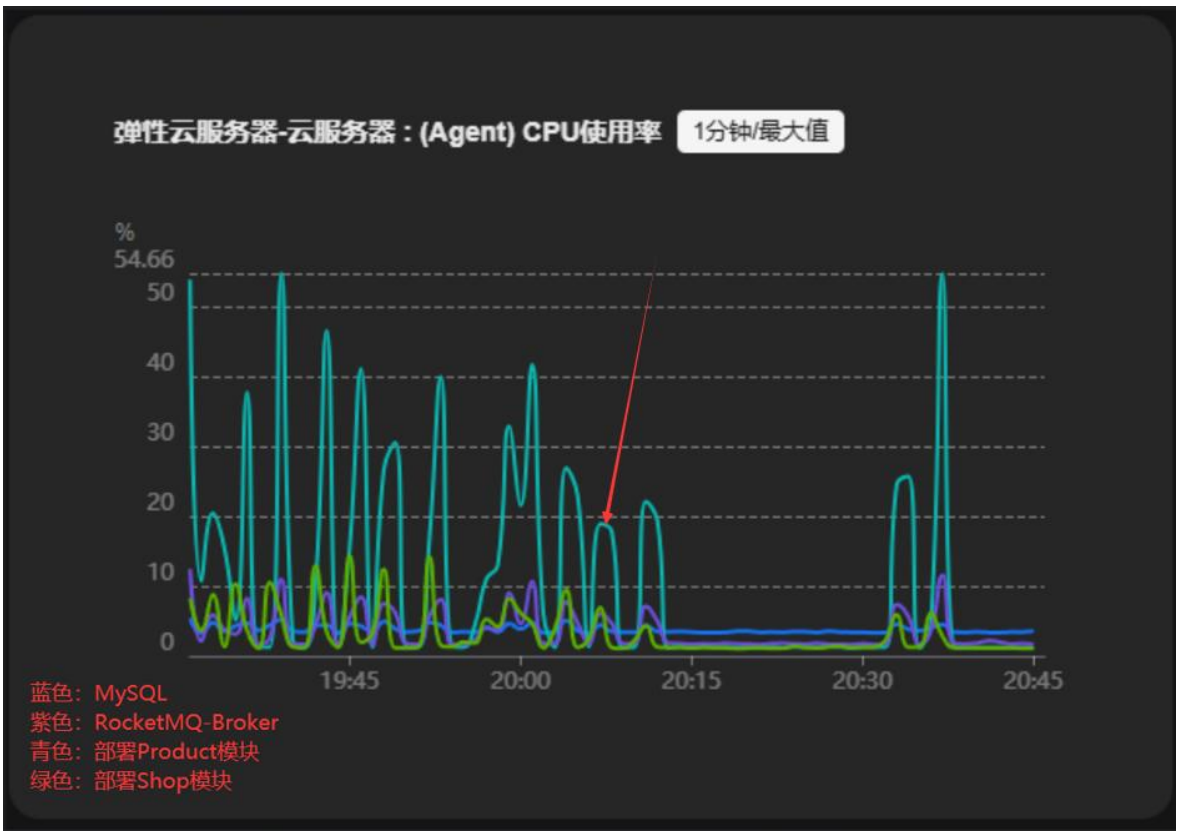

图 6 1 次调用-600-60-3 CPU 使用率

3.5 1次调用-600-60-4（60s 1次调用商铺模块阻塞）

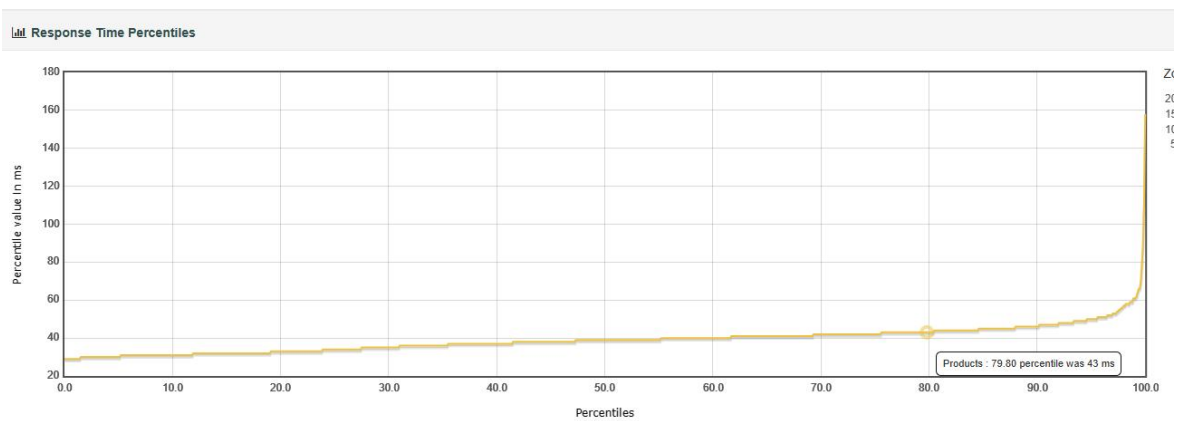

图 7 1 次调用-600-60-4 Response Time Percentiles

80%的请求在 43ms 内响应

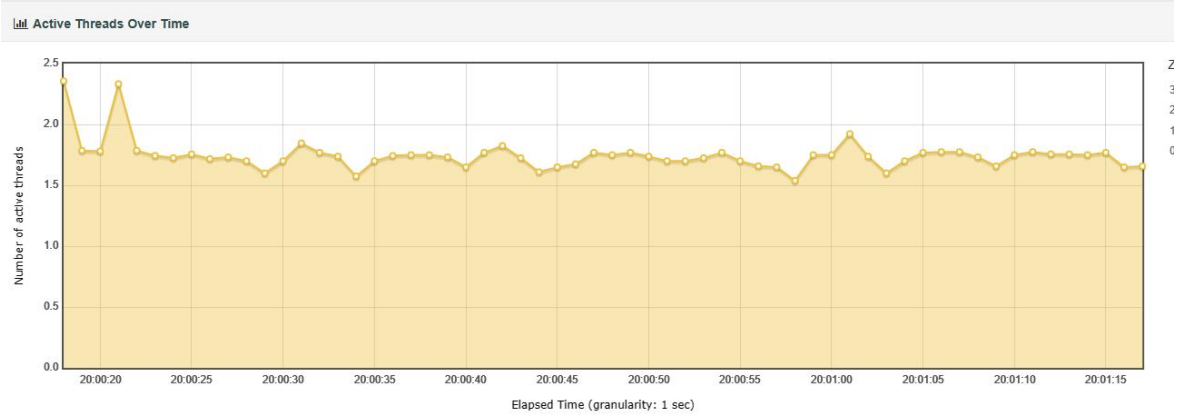

图 8 1 次调用-600-60-4 Active Threads Over Time

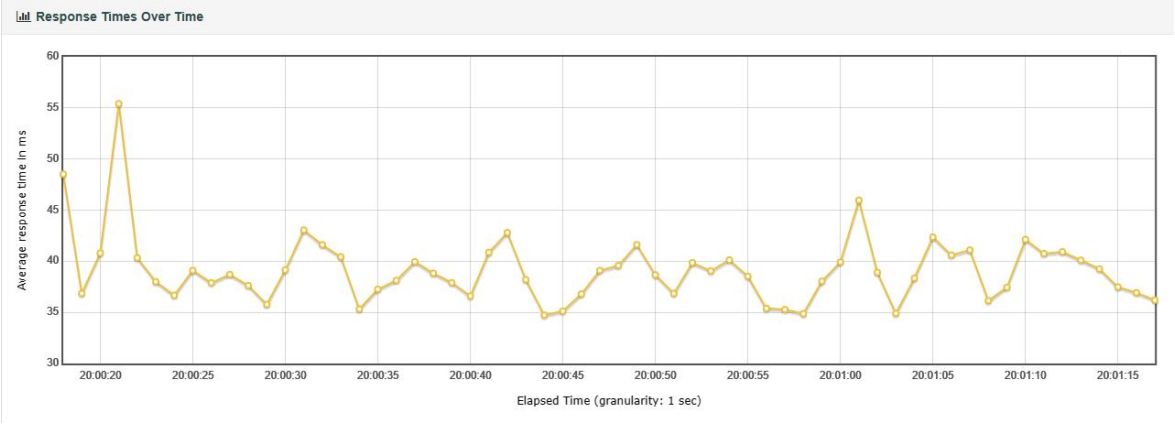

图 9 1 次调用-600-60-4 Response Times Over Time

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**2400**	**0**	**0.00%**	**39.15**	**29**	**158**	**39.00**	**46.00**	**50.00**	**61.00**	**40.09**	**55.44**	**7.09**
Products	2400	0	0.00%	39.15	29	158	39.00	46.00	50.00	61.00	40.09	55.44	7.09

图 10 1 次调用-600-60-4 Statistics

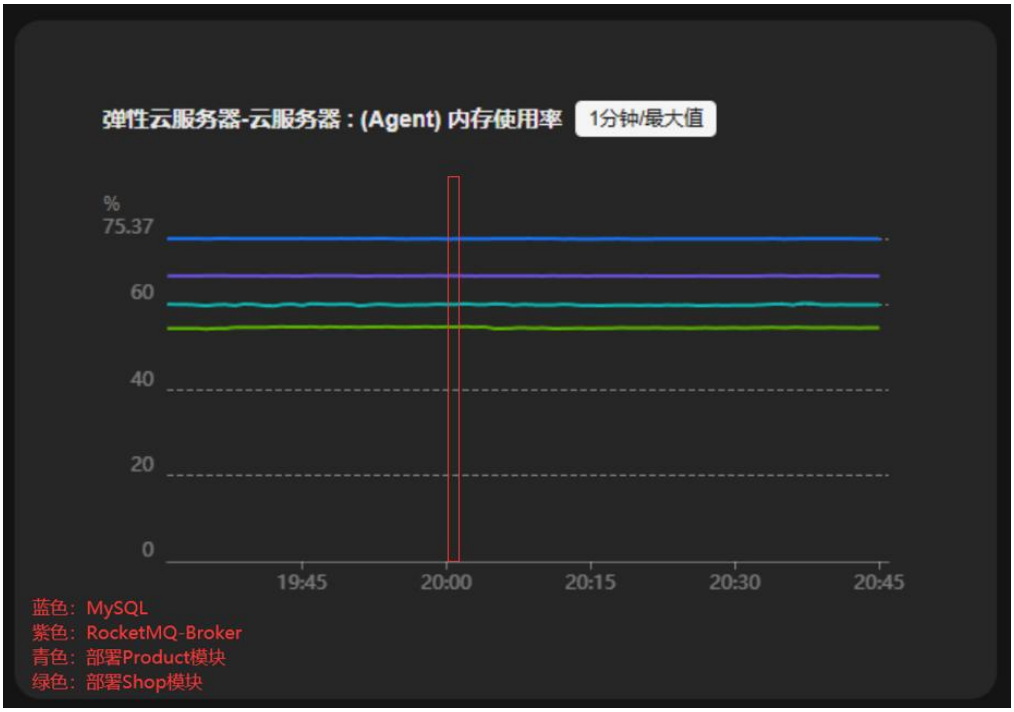

图 11 1 次调用-600-60-4 内存使用率

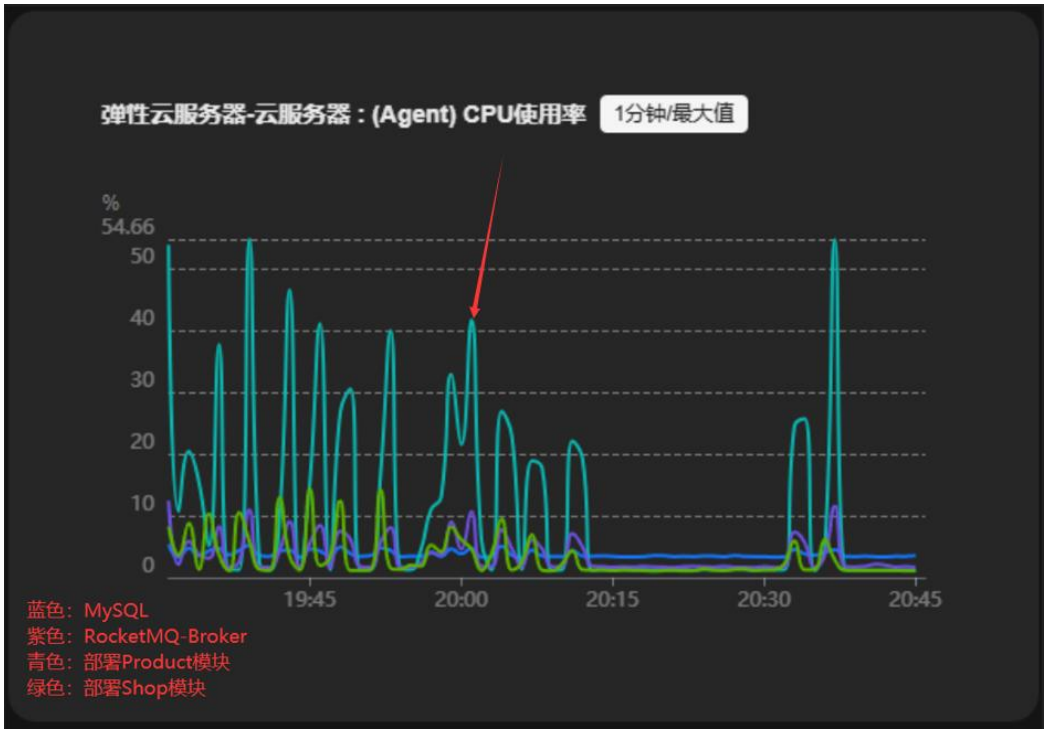

图 12 1 次调用-600-60-4 CPU 使用率

3.6 2次调用-600-60-3（60s 1次调用商铺模块达到的峰值）

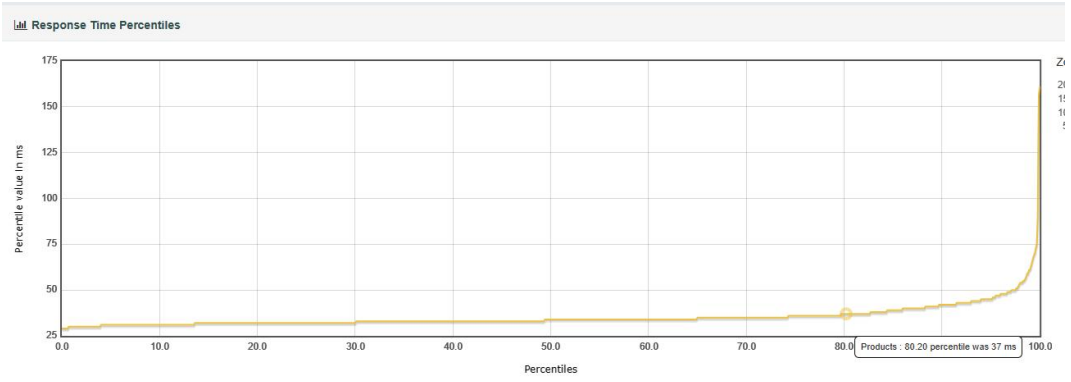

图 1 2 次调用-600-60-3 Response Time Percentiles

80%的请求在 37ms 内响应

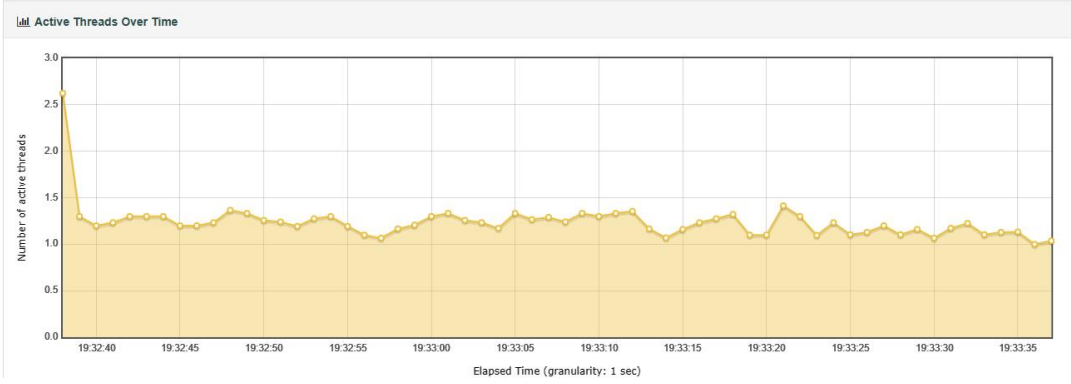

图 2 2 次调用-600-60-3 Active Threads Over Time

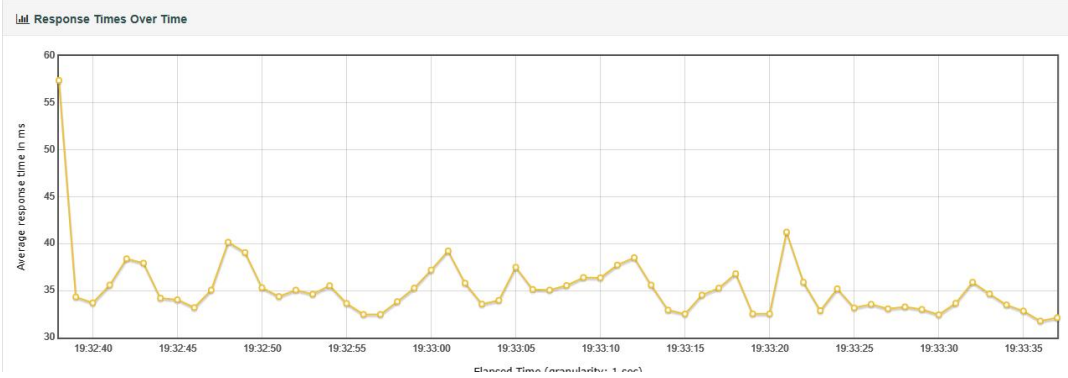

图 3 2 次调用-600-60-3 Response Times Over Time

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	1800	0	0.00%	35.34	29	161	34.00	42.00	45.00	61.00	30.08	38.98	5.32
Products	1800	0	0.00%	35.34	29	161	34.00	42.00	45.00	61.00	30.08	38.98	5.32

图 4 2 次调用-600-60-3 Statistics

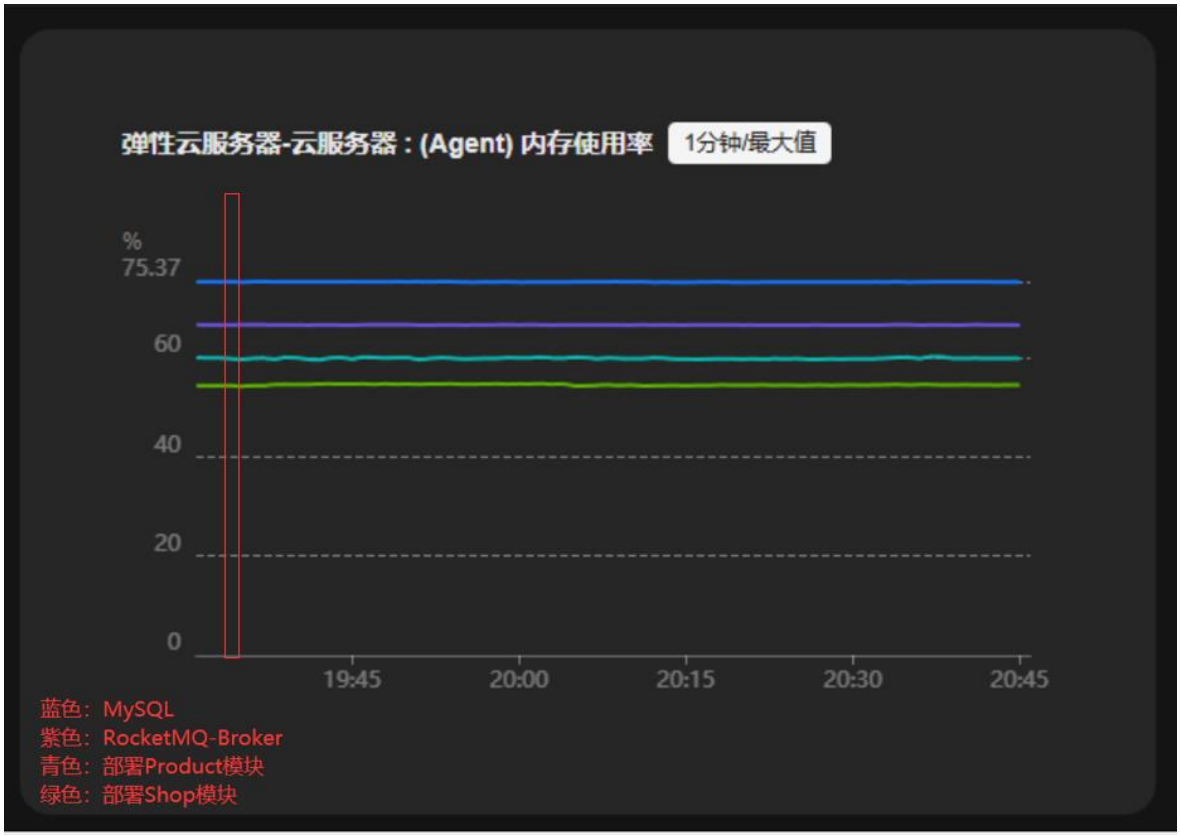

图 5 2 次调用-600-60-3 内存使用率

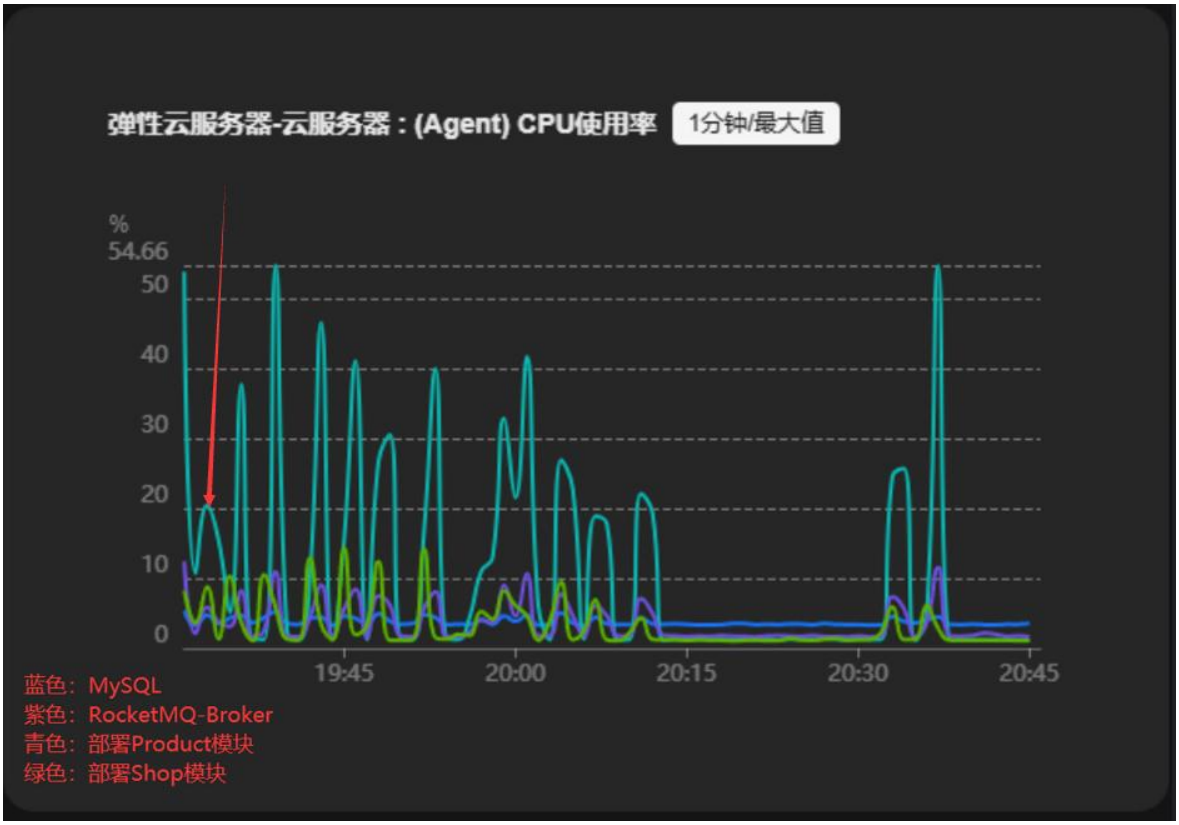

图 6 2 次调用-600-60-3 CPU 使用率

3.7 2次调用-600-60-4（60s 1次调用商铺模块阻塞）

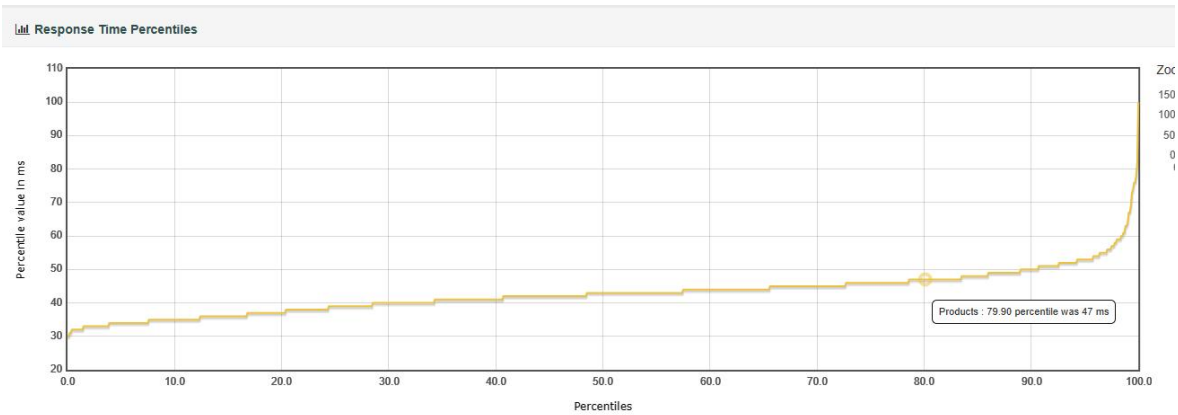

图 7 2 次调用-600-60-4 Response Time Percentiles

80%的请求在 47ms 内响应

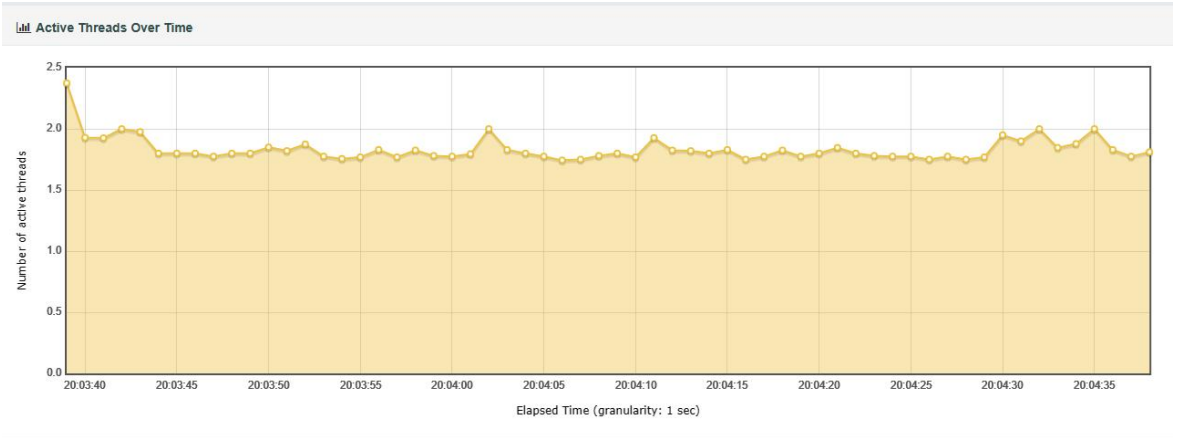

图 8 2 次调用-600-60-4 Active Threads Over Time

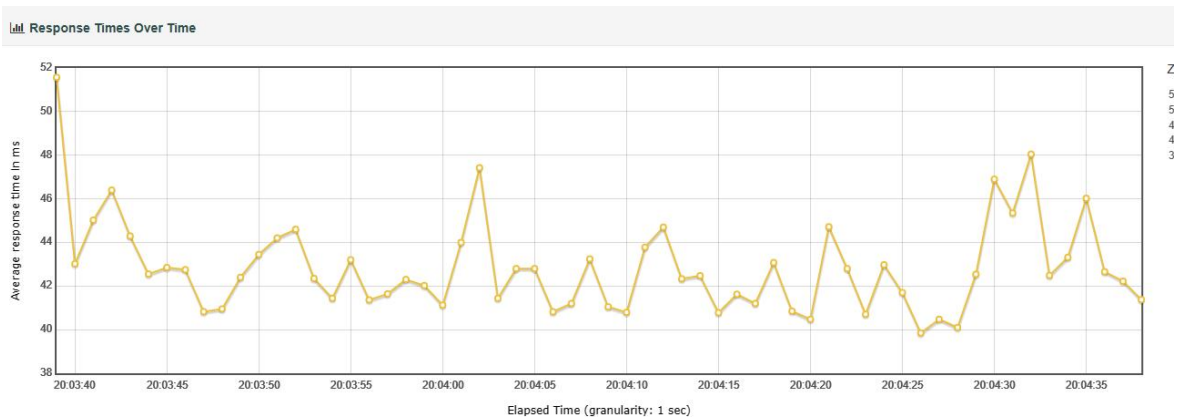

图 9 2 次调用-600-60-4 Response Times Over Time

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**2400**	**0**	**0.00%**	**42.86**	**30**	**100**	**43.00**	**50.00**	**53.00**	**64.00**	**40.05**	**56.87**	**7.08**
Products	2400	0	0.00%	42.86	30	100	43.00	50.00	53.00	64.00	40.05	56.87	7.08

图 10 2 次调用-600-60-4 Statistics

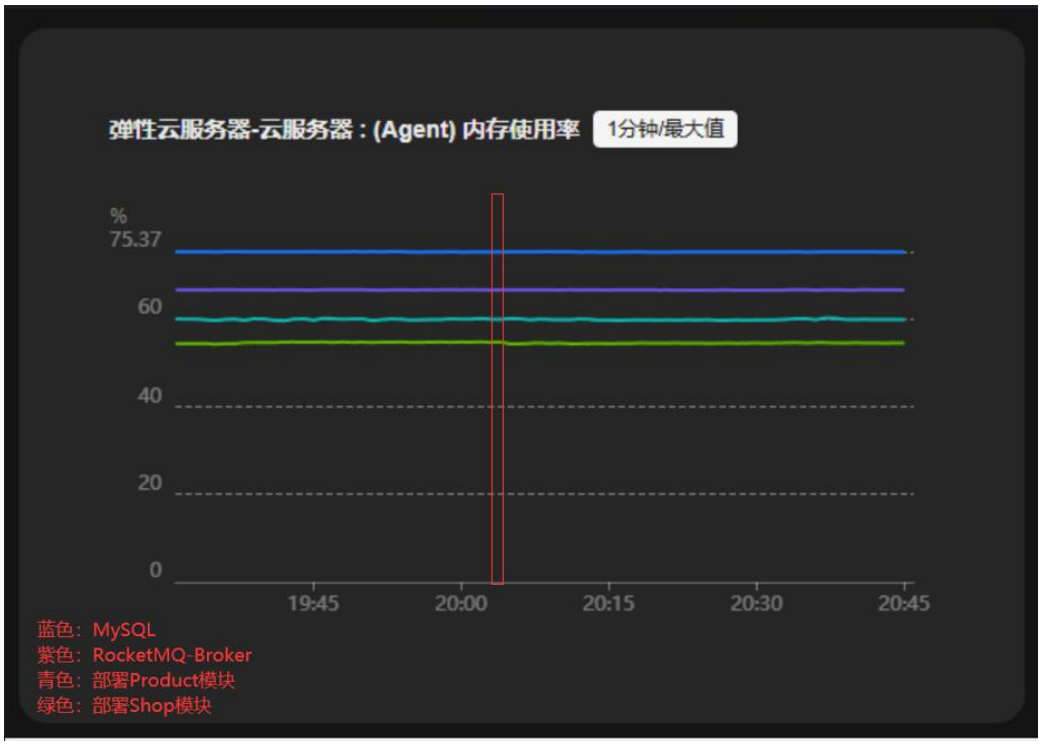

图 11 2 次调用-600-60-4 内存使用率

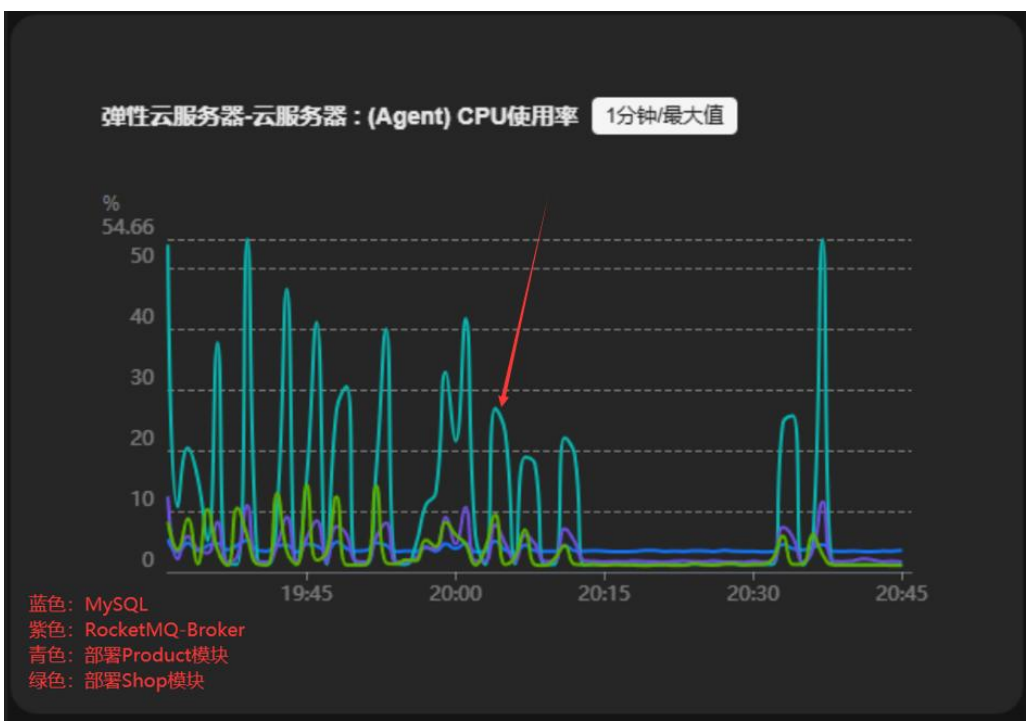

图 12 2 次调用-600-60-4 CPU 使用率

4 总 结

4.1 性能表现：

响应速度前80%的请求所用的时间(ms)			
测试条件	0次调用	1次调用	2次调用
600-60-3	-	38	37
600-60-4	35	43	47
600-60-5	36	-	-
600-60-6	46	-	-

实验结果显示，我们发现随着调用次数的增加，尤其是在多次调用商铺模块时，性能明显下降。

4.2 服务器负载：

通过对服务器的负载情况进行监控，虽然几组测试数据已经出现较高的延迟，但是测试过程中却没有任何一台服务器出现 CPU 占用高的情况。因此，我们猜测响应返回的时间可能受到网络带宽限制影响。根据上图中服务器入网带宽/出网带宽监控情况，我们发现：部署 Product 的服务器在测试时入网带宽较高，部署 RocketMQ-Broker 的服务器出网带宽较高，这说明 RocketMQ 在工作时可能使用了公网进行传输，这也能解释为什么测试性能在相对较低的时候就达到了瓶颈。

参考文献:

[1] 8 场 5 胜，微服务 VS 单体架
https://blog.csdn.net/uxiAD7442KMy1X86DtM3/article/details/107828703?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522de4287f82e95d3025ce967789bedaa16%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=de4287f82e95d3025ce967789bedaa16&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-107828703-null-null.142^v100^pc_search_result_base3&utm_term=%E5%8D%95%E4%BD%93%E5%BA%94%E7%94%A8%E4%B8%8E%E5%BE%AE%E6%9C%8D%E5%8A%A1%E5%BA%94%E7%94%A8%E7%9A%84%E6%95%88%E7%8E%87%E5%AF%B9%E6%AF%94&spm=1018.2226.3001.4187

[2] 微服务架构与传统单体结构项目的比较和应用
https://blog.csdn.net/LaotLisp/article/details/133339344?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522de4287f82e95d3025ce967789bedaa16%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=de4287f82e95d3025ce967789bedaa16&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-133339344-null-null.142^v100^pc_search_result_base3&utm_term=%E5%8D%95%E4%BD%93%E5%BA%94%E7%94%A8%E4%B8%8E%E5%BE%AE%E6%9C%8D%E5%8A%A1%E5%BA%94%E7%94%A8%E7%9A%84%E6%95%88%E7%8E%87%E5%AF%B9%E6%AF%94&spm=1018.2226.3001.4187