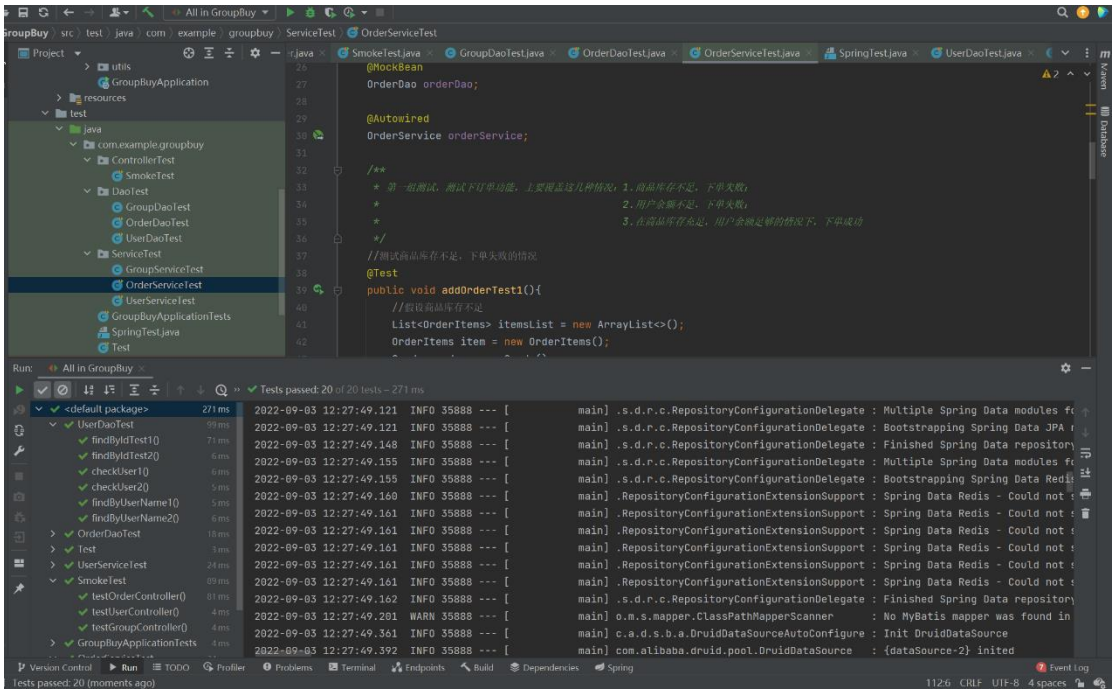
测试与改进报告

一． 在小学期的两次迭代中，我们小组的精力主要花在完成基本功能上，在这一阶段，我们对前后端代码做了很多测试，保证代码的正确性和可靠性。

下图是对后端的单元测试截图：



二． 在暑假中，我们为了达到并发要求和实现好的性能，使用 Redis 和 Rabbitmq 等对许多功能进行了重写或优化，以下订单为例。

1.初代下订单的性能测试（第二次迭代后的版本）：

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.025	500 ms	1 sec	Total
0.012	500 ms	1 sec	addToOrder
0.018	500 ms	1 sec	setNewAddress
0.024	500 ms	1 sec	addToCart
0.029	500 ms	1 sec	getGroupById
0.044	500 ms	1 sec	userAuth

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	10000	0	0.00%	422248.55	14	1599688	291791.00	1029068.20	1100617.55	1205320.62	2.81	1.22	0.57
addToCart	2000	0	0.00%	468201.09	118	1599688	470592.00	969003.00	1059004.55	1180913.05	0.94	0.31	0.17
addToOrder	2000	0	0.00%	771969.39	162	1361166	835384.50	1151760.80	1194748.65	1264734.49	0.56	0.18	0.11
getGroupById	2000	0	0.00%	163520.19	48	799014	133754.50	370279.10	415759.50	491040.43	1.99	1.67	0.28
setNewAddress	2000	0	0.00%	663720.58	16	1524916	736948.00	1079948.70	1099361.10	1235960.41	0.62	0.19	0.17
userAuth	2000	0	0.00%	43831.49	14	214927	38300.00	96460.20	116303.80	141294.86	7.64	2.99	1.86

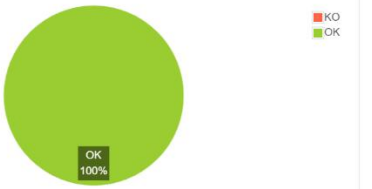
可以看出，性能是很糟糕的，虽然代码正确，但离 2000 高并发的要求还很远。

2.用 Redis 和 Rabbitmq 优化后的性能测试（第三次迭代后的版本）：

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	500 ms	1 sec	Total
1.000	500 ms	1 sec	secKill
1.000	500 ms	1 sec	getAddressId
1.000	500 ms	1 sec	userAuth

Requests Summary



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	6000	0	0.00%	18.00	3	255	8.00	51.00	60.00	67.00	30.12	11.30	7.31
getAddressId	2000	0	0.00%	5.23	3	32	5.00	6.00	7.00	8.00	10.06	4.16	1.43
secKill	2000	0	0.00%	39.80	10	127	39.00	63.00	66.00	70.99	10.05	3.22	3.44
userAuth	2000	0	0.00%	8.96	6	255	8.00	10.00	10.00	12.00	10.04	3.94	2.45

可以看出，性能大大的提高了。

但是回顾了小学期老师讲的课后，我们认为这一版本的代码虽然性能快，但由于是单体代码，如果一个服务出了问题整个软件将不可用，而且缺乏安全验证，可靠性并不是特别好，为此，我们小组认为可以适当牺牲一些性能来换取可靠性。

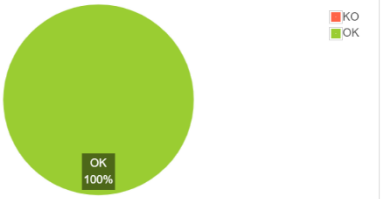
于是，我们设计了第四次迭代，采用微服务架构，并设计了网关和 token 验证

3.采用微服务架构：

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	500 ms	1 sec	Total
1.000	500 ms	1 sec	secKill
1.000	500 ms	1 sec	getAddressId
1.000	500 ms	1 sec	userAuth

Requests Summary



Statistics

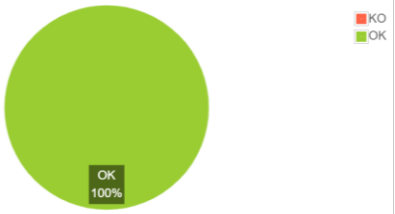
Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	6000	0	0.00%	26.14	3	297	18.00	67.00	79.00	90.00	30.13	7.68	7.30
getAddressId	2000	0	0.00%	5.82	3	40	6.00	7.00	7.00	8.00	10.05	2.94	1.43
secKill	2000	0	0.00%	53.73	14	297	52.00	83.00	88.00	97.00	10.05	2.01	3.42
userAuth	2000	0	0.00%	18.88	12	128	18.00	21.00	23.00	25.00	10.05	2.73	2.45

4.采用了网关和 token 验证：

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	**500 ms**	**1 sec**	**Total**
1.000	500 ms	1 sec	secKill
1.000	500 ms	1 sec	getAddressId
1.000	500 ms	1 sec	userAuth

Requests Summary



Statistics

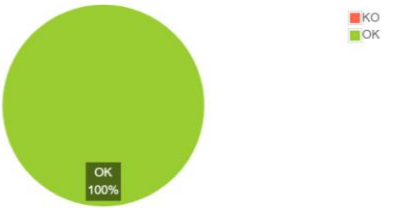
Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**6000**	**0**	**0.00%**	**28.20**	**4**	**514**	**19.00**	**71.00**	**83.00**	**94.00**	**30.11**	**7.67**	**7.29**
getAddressId	2000	0	0.00%	6.52	4	59	6.00	8.00	8.00	12.00	10.07	2.95	1.43
secKill	2000	0	0.00%	57.06	14	402	55.00	87.00	91.00	102.98	10.06	2.01	3.43
userAuth	2000	0	0.00%	21.02	14	514	19.00	23.00	24.00	27.00	10.04	2.72	2.45

3.采用网关但未用 token 验证：

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	**500 ms**	**1 sec**	**Total**
1.000	500 ms	1 sec	secKill
1.000	500 ms	1 sec	getAddressId
1.000	500 ms	1 sec	userAuth

Requests Summary



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**6000**	**0**	**0.00%**	**27.12**	**4**	**382**	**19.00**	**70.00**	**81.00**	**92.00**	**30.20**	**7.69**	**7.31**
getAddressId	2000	0	0.00%	6.20	4	104	6.00	7.00	7.00	9.00	10.08	2.95	1.43
secKill	2000	0	0.00%	55.29	15	382	54.00	85.00	89.00	97.00	10.08	2.02	3.43
userAuth	2000	0	0.00%	19.86	13	203	19.00	22.00	24.00	26.00	10.07	2.73	2.45

性能：单体>微服务>微服务和采用网关但未用 token 验证>微服务和什么都加

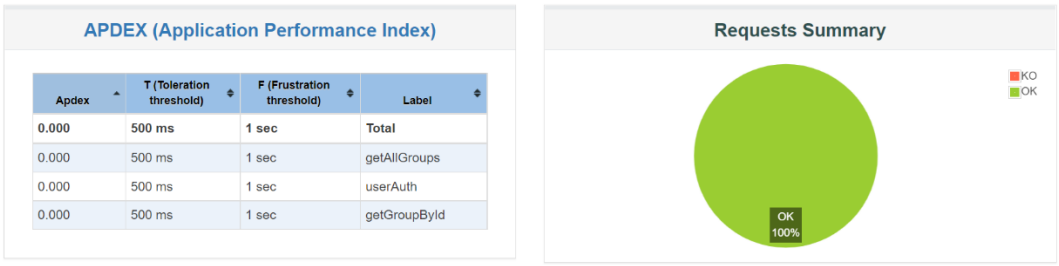
可以看出，性能略有下降，但我们小组的同学认为这种代价的付出是值得的，因为微服务等技术的应用极大的提高了我们程序的可靠性。

0905 https 与 http

单元测试都通过了，正确性方面应该没什么问题；性能测试比较奇怪.

一是性能下降的很多，二是 https 与 http 测出来没有什么差距,以下面的为例：

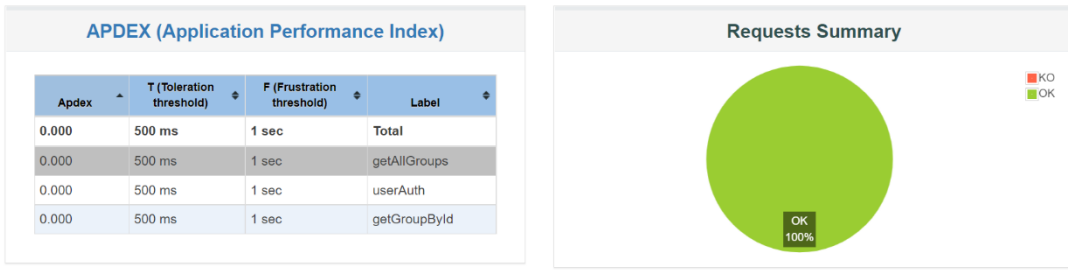
1. https 版本的登录→查看订单→查看订单详情



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**60**	**0**	**0.00%**	**10074.28**	**2312**	**24244**	**6146.00**	**22180.10**	**23435.10**	**24244.00**	**0.70**	**0.42**	**0.14**
getAllGroups	20	0	0.00%	20379.25	13086	24244	20754.00	23464.30	24205.15	24244.00	0.26	0.18	0.06
getGroupById	20	0	0.00%	3900.10	2312	6985	3326.00	6276.40	6951.20	6985.00	0.30	0.21	0.04
userAuth	20	0	0.00%	5943.50	3803	8727	5433.50	8411.40	8712.35	8727.00	0.30	0.12	0.07

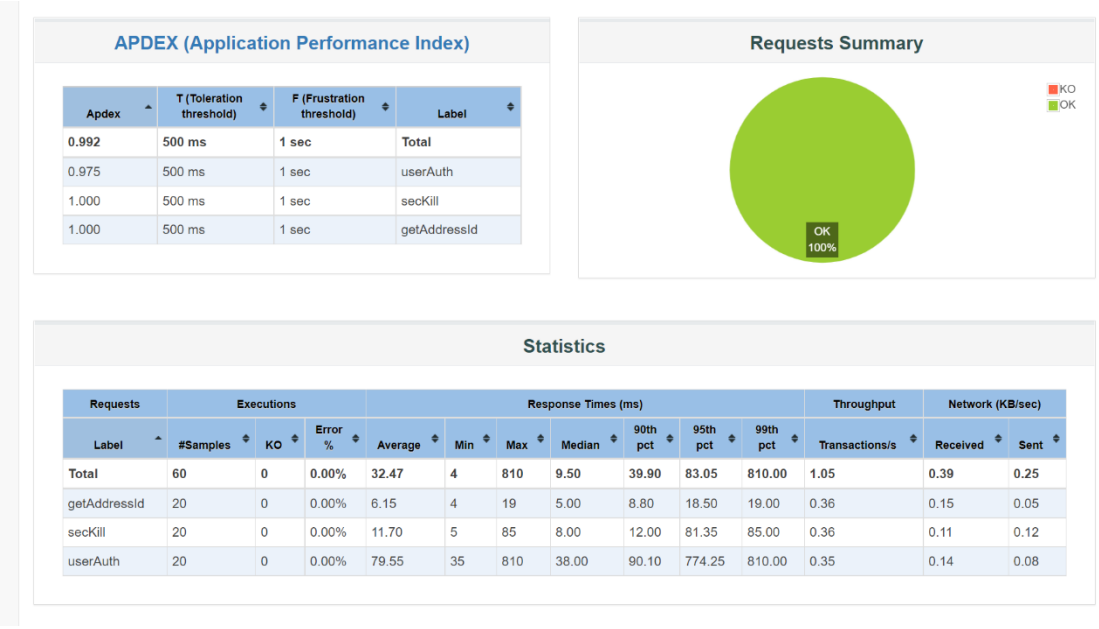
2. http 版本的登录→查看订单→查看订单详情



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**60**	**0**	**0.00%**	**10040.72**	**2252**	**25639**	**6219.50**	**23106.00**	**24212.80**	**25639.00**	**0.70**	**0.41**	**0.14**
getAllGroups	20	0	0.00%	20491.50	12917	25639	20687.00	24830.40	25602.00	25639.00	0.25	0.17	0.05
getGroupById	20	0	0.00%	4076.95	2252	7732	3220.00	6931.10	7692.50	7732.00	0.29	0.20	0.04
userAuth	20	0	0.00%	5553.70	3469	9264	4985.00	7920.90	9199.65	9264.00	0.30	0.12	0.07

3. https 版本的秒杀



APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.992	500 ms	1 sec	Total
0.975	500 ms	1 sec	userAuth
1.000	500 ms	1 sec	secKill
1.000	500 ms	1 sec	getAddressId

Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	60	0	0.00%	32.47	4	810	9.50	39.90	83.05	810.00	1.05	0.39	0.25
getAddressId	20	0	0.00%	6.15	4	19	5.00	8.80	18.50	19.00	0.36	0.15	0.05
secKill	20	0	0.00%	11.70	5	85	8.00	12.00	81.35	85.00	0.36	0.11	0.12
userAuth	20	0	0.00%	79.55	35	810	38.00	90.10	774.25	810.00	0.35	0.14	0.08

4. http 版本的秒杀

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	500 ms	1 sec	Total
1.000	500 ms	1 sec	secKill
1.000	500 ms	1 sec	getAddressId
1.000	500 ms	1 sec	userAuth

Requests Summary

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	60	0	0.00%	15.55	6	206	10.00	16.00	18.95	206.00	1.05	0.39	0.25
getAddressId	20	0	0.00%	7.15	6	19	6.50	7.90	18.45	19.00	0.35	0.14	0.05
secKill	20	0	0.00%	15.10	6	138	9.00	11.00	131.65	138.00	0.35	0.11	0.12
userAuth	20	0	0.00%	24.40	13	206	15.00	17.80	196.60	206.00	0.35	0.14	0.08

http 和 https 没有什么区别让我疑惑（可能是我不懂 https 的实现，我只发现了配置文件上面的变化，代码逻辑上的变化我没有找到）

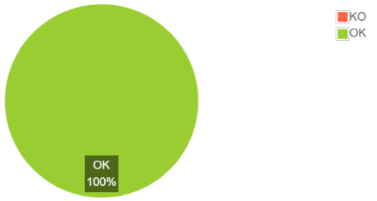
然后就是性能下降的非常大，这次测试我将并发线程降到了 20 个(因为太慢了，2000 并发可能一个测试得跑一天)，但还是很慢。

附上之版本的登录→查看订单→查看订单详情这一条流的性能(2000 并发):

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	500 ms	1 sec 500 ms	Total
1.000	500 ms	1 sec 500 ms	getAllGroups
1.000	500 ms	1 sec 500 ms	userAuth
1.000	500 ms	1 sec 500 ms	getGroupById

Requests Summary



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	6000	0	0.00%	14.72	8	284	13.00	19.00	21.00	39.00	100.17	86.05	20.02
getAllGroups	2000	0	0.00%	14.06	9	179	12.00	17.00	19.00	36.99	33.49	45.20	7.23
getGroupById	2000	0	0.00%	11.56	8	106	11.00	13.00	16.00	29.99	33.59	27.91	4.66
userAuth	2000	0	0.00%	18.54	12	284	16.00	21.00	26.00	82.81	33.40	13.24	8.19

0907 Security

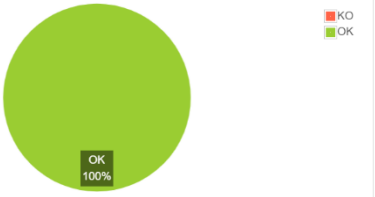
正确性方面没什么问题，像重复注册，用户名或密码错误等异常也能返回正确的结果。

依旧是性能有点差：

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.055	500 ms	1 sec	Total
0.041	500 ms	1 sec	login
0.070	500 ms	1 sec	Register

Requests Summary

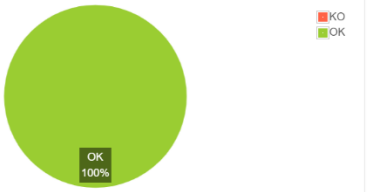


Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	4000	0	0.00%	4616.34	182	16314	4579.00	7525.50	8517.75	10749.80	56.78	48.44	13.64
login	2000	0	0.00%	4901.15	200	16314	4921.50	7916.20	8769.75	10669.23	28.51	32.47	6.22
Register	2000	0	0.00%	4331.53	182	16202	4257.50	7030.00	8129.00	10796.54	28.47	16.15	7.46

附上之前版本的注册登录：

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	**500 ms**	**1 sec**	**Total**
1.000	500 ms	1 sec	Register
1.000	500 ms	1 sec	userAuth
1.000	500 ms	1 sec	getGroupById



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**6000**	**0**	**0.00%**	**11.15**	**6**	**107**	**12.00**	**14.00**	**15.00**	**17.00**	**50.39**	**30.82**	**10.83**
getGroupById	2000	0	0.00%	13.22	11	51	13.00	15.00	16.00	18.00	16.82	17.70	2.33
Register	2000	0	0.00%	12.70	9	107	12.00	14.00	15.00	17.00	16.80	6.55	4.40
userAuth	2000	0	0.00%	7.54	6	27	7.00	9.00	9.00	10.00	16.82	6.59	4.10

多了一张表要插入或查询感觉跑成这样比较正常。