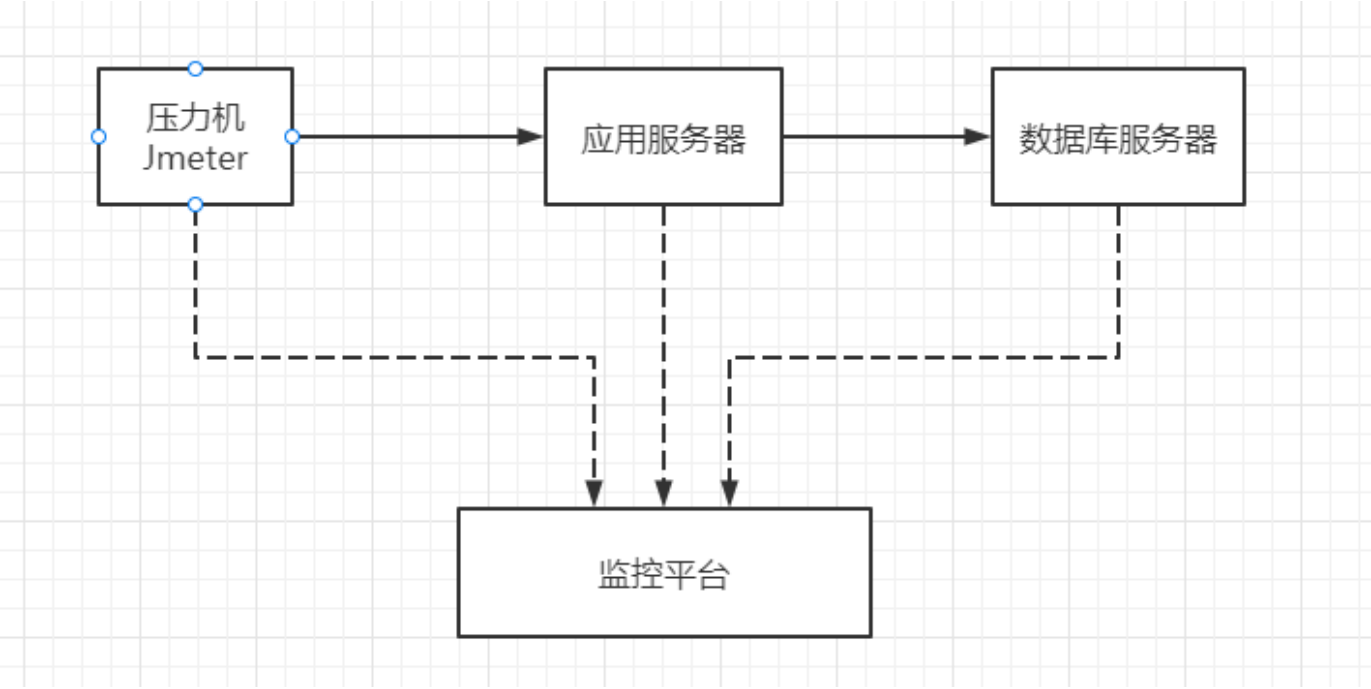
项目性能测试报告

一、测试目的

主要是让开发者对 hero_mall 项目的性能负载和容量有个准确的认知。同时，协助技术管理者更好的管理业务系统性能质量，科学评估业务系统的负荷，拒绝盲目上线。

二、测试工具



三、测试环境

3.1 环境

指标	参数
CPU、内存	4C 8G
集群规模	单机
hero_mall_one版本	hero_web-1.0-SNAPSHOT-default.jar
数据库	4C 8G
网络带宽	100Mbps

3.2 设置启动参数

Shell | 复制代码

```shell
export JAVA_HOME=/usr/local/hero/jdk1.8.0_261
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH

#================================================================================
# init
#================================================================================

export SERVER="hero_web"
export JAVA_HOME
export JAVA="$JAVA_HOME/bin/java"
# 获取当前目录
export BASE_DIR=`cd $(dirname $0)/.; pwd`
# 默认加载路径
export DEFAULT_SEARCH_LOCATIONS="classpath:/,classpath:/config/,file:./,file:./config/"
# 自定义默认加载配置文件路径
export CUSTOM_SEARCH_LOCATIONS=${DEFAULT_SEARCH_LOCATIONS},file:${BASE_DIR}/conf/


#================================================================================
# JVM Configuration
#================================================================================
JAVA_OPT="${JAVA_OPT} -server -Xms512m -Xmx512m -Xmn256 -XX:MetaspaceSize=128m -XX:MaxMetaspaceSize=320m"
JAVA_OPT="${JAVA_OPT} -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=${BASE_DIR}/logs/java_heapdump.hprof"
JAVA_OPT="${JAVA_OPT} -XX:-UseLargePages"
JAVA_OPT="${JAVA_OPT} -jar ${BASE_DIR}/${SERVER}-*.jar"
JAVA_OPT="${JAVA_OPT} ${JAVA_OPT_EXT}"
JAVA_OPT="${JAVA_OPT} --spring.config.location=${CUSTOM_SEARCH_LOCATIONS} --spring.profiles.path=/root/application-dev.yml"
# 创建日志文件目录
if [ ! -d "${BASE_DIR}/logs" ]; then
  mkdir ${BASE_DIR}/logs
fi

# 输出变量
echo "$JAVA ${JAVA_OPT}"
```

```
# 检查start.out日志输出文件
if [ ! -f "${BASE_DIR}/logs/${SERVER}.out" ]; then
  touch "${BASE_DIR}/logs/${SERVER}.out"
fi
#===========================================================================
# 启动服务
#===========================================================================
# 启动服务
echo "$JAVA ${JAVA_OPT}" > ${BASE_DIR}/logs/${SERVER}.out 2>&1 &
nohup $JAVA ${JAVA_OPT} hero_web.hero_web >> ${BASE_DIR}/logs/${SERVER}.out 2>&1 &
echo "server is starting, you can check the ${BASE_DIR}/logs/${SERVER}.out"
```

四、测试场景

测试场景一般情况下是都是最重要接口：验证hero_mall服务获取商品信息接口在不同并发规模的表现。

本次压力测试测试单接口：获取商品信息接口，在不同场景下的性能状况并进行分析优化。

情况1 – 模拟低延时场景

用户访问接口并发逐渐增加的过程。接口的响应时间为20ms，线程梯度：5、10、15、20、25、30、35、40个线程，5000次;

- 时间设置：Ramp–upperiod(inseconds)的值设为对应线程数
- 测试总时长：约等于20msx5000次 x8 = 800s = 13分

情况2 – 模拟高延时场景

用户访问接口并发逐渐增加的过程。接口的响应时间为500ms，线程梯度：100、200、300、400、500、600、700、800个线程，200次;

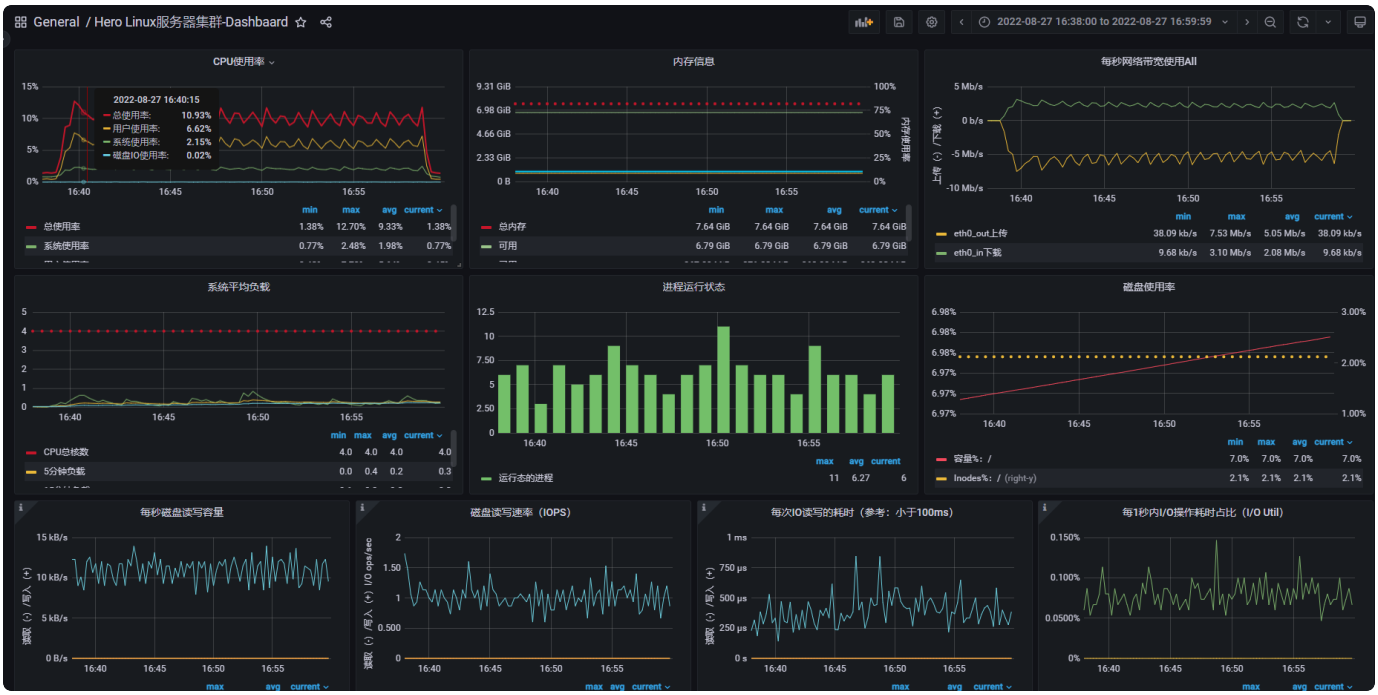
- 时间设置：Ramp–upperiod(inseconds)的值设为对应线程数的1/10；
- 测试总时长：约等于500msx200次 x8 = 800s = 13分

五、测试结果

5.1 低延时场景下，不同传输大小对性能的影响

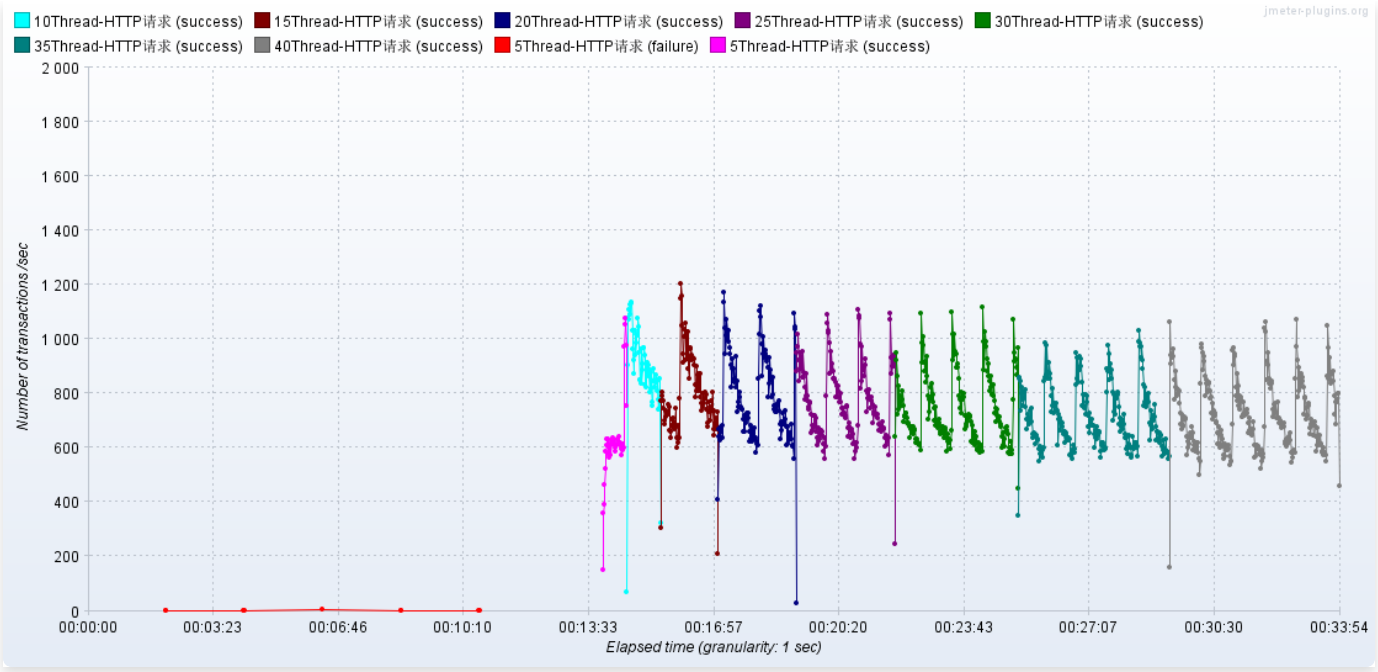
使用的脚本：03-jmeter-example-低延迟20ms-响应1.1k.jmx

下图为监控平台对应用服务器的监控结果图：



聚合报告结果：

Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
5Thread-HT...	25025	134	7	11	13	18	1	127329	0.10%	28.6/sec	30.32	0.00
10Thread-H...	50000	10	10	16	17	22	1	37	0.00%	914.2/sec	970.20	0.00
15Thread-H...	75000	18	18	28	31	39	1	112	0.00%	799.3/sec	848.30	0.00
20Thread-H...	100000	25	23	39	44	56	1	128	0.00%	789.2/sec	837.55	0.00
25Thread-H...	125000	31	29	52	61	83	1	177	0.00%	775.4/sec	822.93	0.00
30Thread-H...	150000	39	34	69	84	117	1	277	0.00%	749.0/sec	794.87	0.00
35Thread-H...	175000	48	41	88	108	154	1	385	0.00%	713.5/sec	757.20	0.00
40Thread-H...	200000	54	44	104	130	193	1	575	0.00%	725.0/sec	769.40	0.00
总体	900025	41	30	73	95	148	1	127329	0.00%	442.4/sec	469.53	0.00

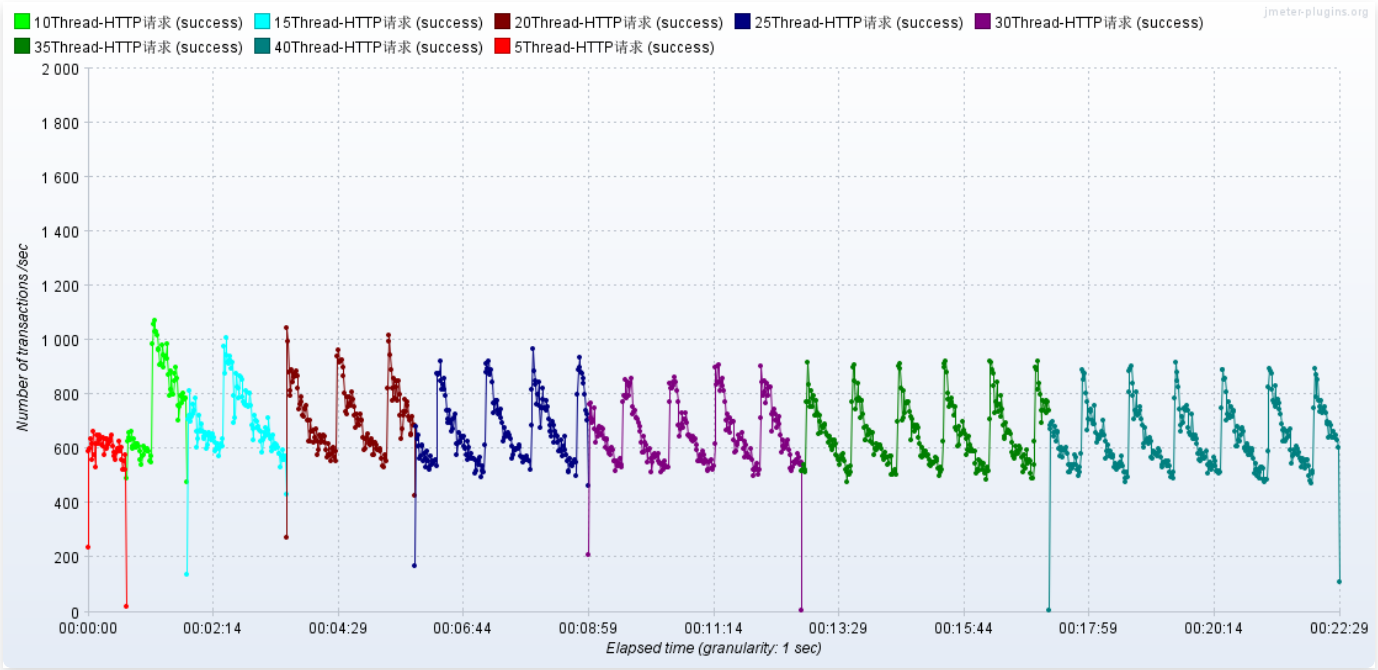


使用的脚本：03-jmeter-example-低延迟20ms-响应3.8k.jmx

下图为监控平台对应用服务器的监控结果图：

聚合报告结果：

Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
5Thread-HT...	25000	8	7	12	13	16	1	27	0.00%	598.9/sec	2202.33	0.00
10Thread-H...	50000	12	12	19	21	26	1	102	0.00%	761.7/sec	2801.27	0.00
15Thread-H...	75000	21	20	31	35	42	1	112	0.00%	701.5/sec	2579.78	0.00
20Thread-H...	100000	27	26	42	47	59	1	143	0.00%	719.1/sec	2644.49	0.00
25Thread-H...	125000	37	34	59	70	95	2	210	0.00%	668.6/sec	2458.61	0.00
30Thread-H...	150000	45	40	77	93	130	2	290	0.00%	654.0/sec	2405.07	0.00
35Thread-H...	175000	52	44	95	117	168	1	429	0.00%	655.6/sec	2410.79	0.00
40Thread-H...	200000	61	50	116	144	209	2	532	0.00%	640.5/sec	2355.24	0.00
总体	900000	42	34	81	104	161	1	532	0.00%	667.2/sec	2453.59	0.00



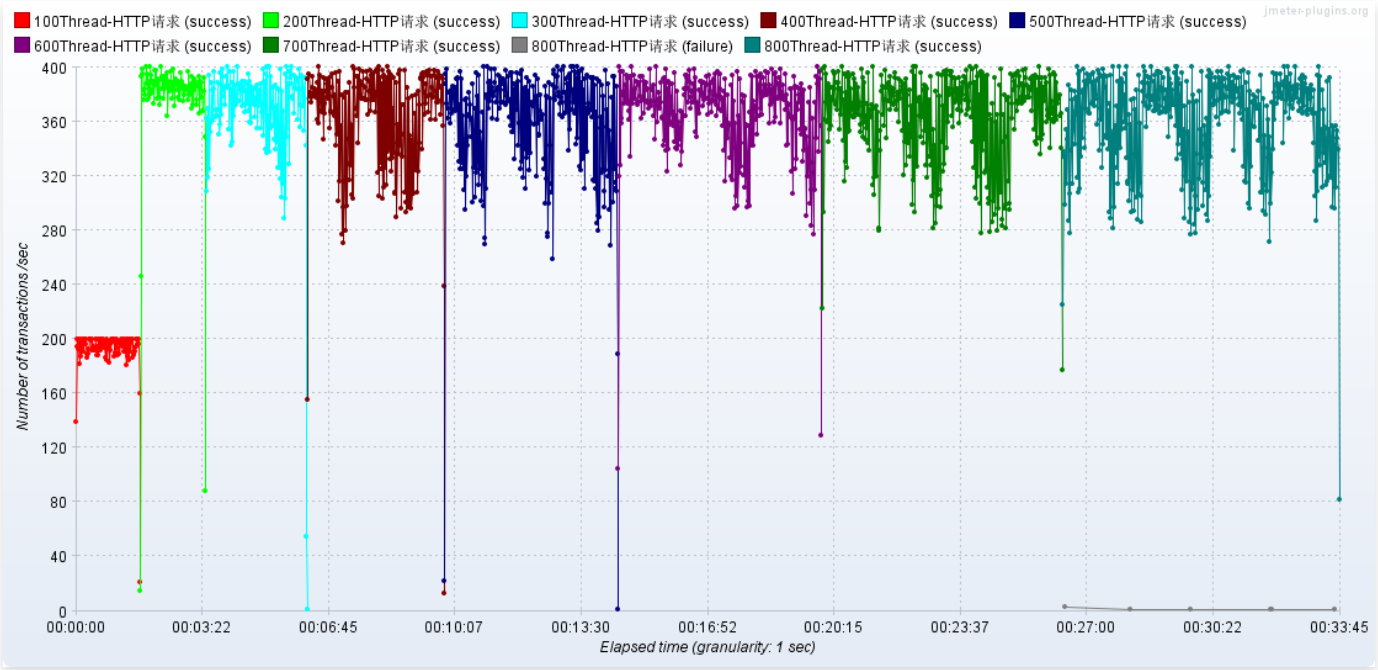
5.2 高延时场景下，不同线程数对性能的影响

使用的脚本：04-jmeter-example-高延迟500ms-200线程.jmx

下图为监控平台对应用服务器的监控结果图：

聚合报告结果：

Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
100Thread-...	20000	513	512	523	526	534	501	585	0.00%	192.6/sec	204.44	0.00
200Thread-...	40000	519	517	535	543	569	501	844	0.00%	380.1/sec	403.44	0.00
300Thread-...	60000	801	797	879	915	997	502	1454	0.00%	369.5/sec	392.13	0.00
400Thread-...	80000	1090	1059	1237	1314	1445	502	1705	0.00%	362.9/sec	385.19	0.00
500Thread-...	100000	1377	1367	1507	1558	1676	502	2072	0.00%	359.9/sec	381.99	0.00
600Thread-...	120000	1620	1595	1765	1845	2021	502	2370	0.00%	367.6/sec	390.09	0.00
700Thread-...	140000	1914	1888	2098	2166	2295	502	2664	0.00%	363.7/sec	385.95	0.00
800Thread-...	160000	2207	2158	2445	2526	2683	502	3048	0.01%	360.5/sec	382.57	0.00
总体	720000	1555	1596	2196	2328	2538	501	3048	0.00%	355.6/sec	377.35	0.00

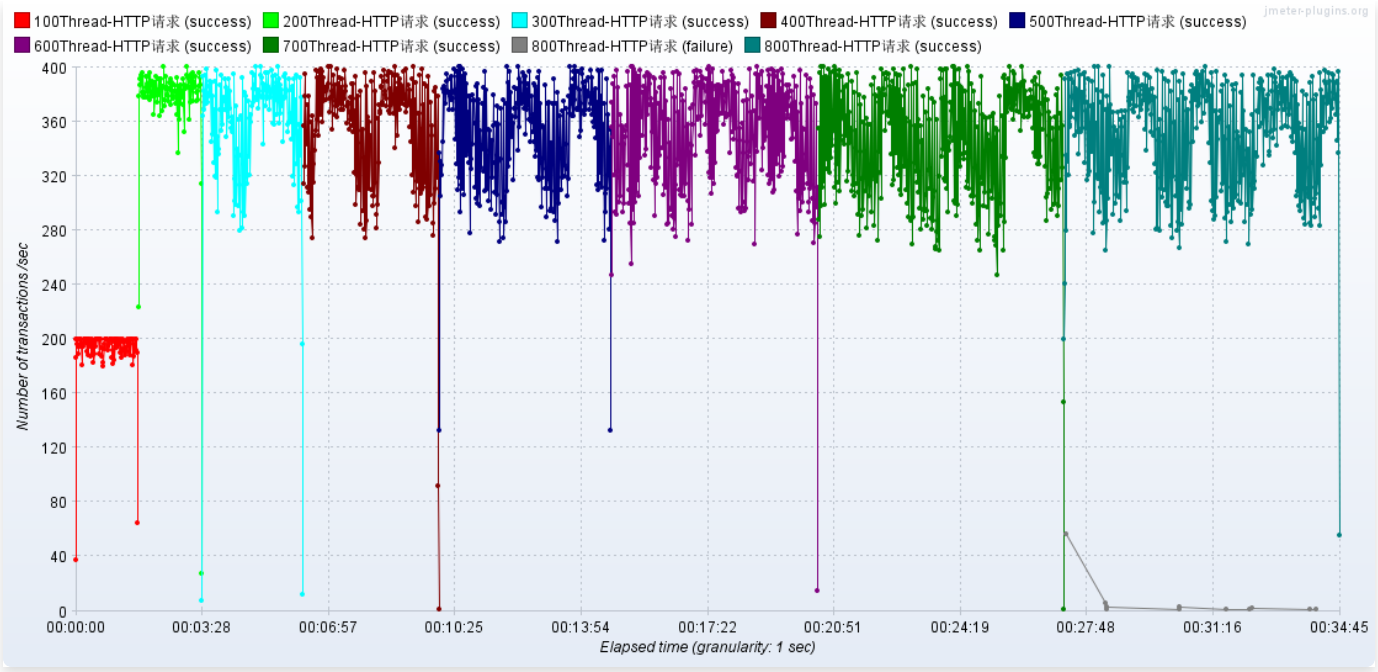


使用的脚本：04-jmeter-example-高延迟500ms-800线程.jmx

下图为监控平台对应用服务器的监控结果图：

聚合报告结果：

Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %
100Thread-HTTP...	20000	511	512	520	523	529	501	550	0.00%
200Thread-HTTP...	40000	522	519	538	548	579	501	782	0.00%
300Thread-HTTP...	60000	819	802	930	991	1122	502	1510	0.00%
400Thread-HTTP...	80000	1110	1068	1291	1370	1494	502	1745	0.00%
500Thread-HTTP...	100000	1407	1398	1556	1603	1710	502	2094	0.00%
600Thread-HTTP...	120000	1694	1663	1909	1995	2152	502	2429	0.00%
700Thread-HTTP...	140000	2014	2003	2223	2295	2424	502	2756	0.00%
800Thread-HTTP...	160000	2259	2235	2511	2588	2750	503	3623	0.05%
总体	720000	1606	1661	2290	2410	2599	501	3623	0.01%



六、测试结论

本次压力测试是针对单功能，单机单节点进行压测，可以通过监控平台查看到对应机器资源在压测中的变化，为优化提供可靠的指标信息。本测试供给参考，如有不足或偏差或错误，请指正！本次是测试是单节点梯度压测，如果对性能有其他需求，可以进行集群扩容。例如:3节点、10 节点、100节点...等多节点部署，然后进行分布式压测即可。