### 儒猿：10万级并发演示

* 部署环境：

|  |  |  |  |
| --- | --- | --- | --- |
| * 主机名 | * 配置信息 | * 服务说明 | * 备注 |
| * rapid-1 | * 16C/32G | * 网关服务 |  |
| * rapid-2 | * 16C/32G | * 网关服务 |  |
| * rapid-3 | * 16C/32G | * 网关服务 |  |
| * rapid-4 | * 16C/32G | * 网关服务 |  |
| * rapid-5 | * 16C/32G | * 网关服务 |  |
| * service-1 | * 8C/16G | * dubbo/http |  |
| * service-2 | * 8C/16G | * dubbo/http |  |
| * service-3 | * 8C/16G | * dubbo/http |  |
| * service-4 | * 8C/16G | * dubbo/http |  |
| * service-5 | * 8C/16G | * dubbo/http |  |
| * base-r1 | * 4C/8G | * 基础配置：etcd/zookeeper |  |
| * base-r2 | * 4C/8G | * 基础配置：etcd/zookeeper |  |
| * base-r3 | * 4C/8G | * 基础配置：etcd/zookeeper |  |
| * wrk | * 16C/32G | * 压测请求服务器 |  |

* linux性能调优：
* ulimit -n   
  ## 默认为65536  
    
  ## 禁用swap  
  sudo swapoff -a  
    
  ## 修改linux内核参数  
  sudo vim /etc/security/limits.conf  
    
  ## 添加如下内容:  
  \* soft nofile 655350  
  \* hard nofile 655350  
  \* soft memlock unlimited  
  \* hard memlock unlimited  
    
  ## 修改虚拟内存空间及swap使用率  
  cd /proc/sys/vm  
    
  ## 设置如下:  
  sysctl -w vm.max\_map\_count=655350  
  sysctl -w vm.swappiness=1  
    
  ## 查看  
  cat /proc/sys/vm/swappiness  
  cat /proc/sys/vm/max\_map\_count  
    
  #设置永久有效需要把配置信息添加到sysctl.conf文件中去   
  ## 命令：  
  vim /etc/sysctl.conf  
    
  ## 添加：  
  vm.max\_map\_count=655360  
  vm.swappiness=1  
    
  ## 执行命令：使配置生效   
  sudo sysctl -p
* 基础设置服务说明：
* ## zk服务状态：  
  zkServer.sh status  
  ## etcd服务状态：  
  /usr/local/etcd/etcdctl --write-out=table --endpoints=172.18.135.228:2379,172.18.135.229:2379,172.18.135.227:2379 endpoint status  
    
  ## etcd查询：  
  etcdctl get --prefix /rapid-dev/services  
  etcdctl get --prefix /rapid-dev/rules  
  etcdctl get --prefix /rapid-dev/instances
* 压测准备：
* ## wrk只能运行于 Unix 类的系统上，也只能在这些系统上便宜，所以我们需要一个Linux或者macOs。不得不说，使用了 Win10之后方便很多。  
    
  sudo yum groupinstall 'Development Tools'  
  sudo yum install -y openssl-devel git   
  git clone https://github.com/wg/wrk.git wrk  
  cd wrk  
  make  
  ## 将可执行文件移动到 /usr/local/bin 位置  
  sudo cp wrk /usr/local/bin
* wrk使用简介：
* ## 这条命令表示，利用 wrk 对 www.baidu.com 发起压力测试，线程数为12，模拟400个并发请求，持续30秒；  
  wrk -t12 -c400 -d30s http://www.baidu.com  
    
  ## 除了上面简单示例中使用到的子命令参数，wrk 还有其他更丰富的功能，命令行中输入 wrk --help, 可以看到支持以下子命令：  
  使用方法: wrk <选项> <被测HTTP服务的URL>   
   Options:   
   -c, --connections <N> 跟服务器建立并保持的TCP连接数量   
   -d, --duration <T> 压测时间   
   -t, --threads <N> 使用多少个线程进行压测   
     
   -s, --script <S> 指定Lua脚本路径   
   -H, --header <H> 为每一个HTTP请求添加HTTP头   
   --latency 在压测结束后，打印延迟统计信息   
   --timeout <T> 超时时间   
   -v, --version 打印正在使用的wrk的详细版本信息   
   <N>代表数字参数，支持国际单位 (1k, 1M, 1G)  
   <T>代表时间参数，支持时间单位 (2s, 2m, 2h)  
     
  ## PS: 关于线程数，并不是设置的越大，压测效果越好，线程设置过大，反而会导致线程切换过于频繁，效果降低，一般来说，推荐设置成压测机器 CPU 核心数的 2 倍到 4 倍就行了。
* 压测脚本lua：
* wrk.method = "POST"  
  wrk.body = '["xiaoxiao"]'  
  wrk.headers["Content-Type"] = "application/json"  
  wrk.headers["uniqueId"] = "say:1.0.0"  
    
  function request()  
   return wrk.format('POST', nil, nil, body)  
  end
* 网关服务部署运行：
* java -jar rapid-core-1.0.0-no.jar &  
  java -jar rapid-core-1.0.0-no-false.jar &  
  java -jar rapid-core-1.0.0-flusher.jar &  
  java -jar rapid-core-1.0.0-mpmc.jar &
* 业务服务部署运行：
* ## http启动命令  
  java -jar rapid-test-mvc-8083.jar &  
  ## dubbo启动命令  
  java -jar rapid-test-dubbo-1.0.0-8085.jar &
* 阿里云单点基础测试：
* ## http服务单点网关服务压测   
  cd /root/wrk && wrk -t 16 -c 400 -d 60s --latency --timeout 5s -s /usr/local/http.lua http://172.18.135.223:8888/testPost  
    
  ## 压测结果：  
  Running 1m test @ http://172.18.135.223:8888/testPost  
   18 threads and 1000 connections  
   Thread Stats Avg Stdev Max +/- Stdev  
   Latency 19.14ms 33.14ms 1.26s 99.70%  
   Req/Sec 3.13k 223.47 12.15k 79.36%  
   Latency Distribution  
   50% 17.50ms  
   75% 19.07ms  
   90% 20.80ms  
   99% 26.16ms  
   3364249 requests in 1.00m, 465.22MB read  
  Requests/sec: 55978.49  
  Transfer/sec: 7.74MB  
    
    
  ## dubbo服务单点网关服务压测：  
  cd /root/wrk && wrk -t 16 -c 400 -d 60s --latency --timeout 5s -s /usr/local/dubbo.lua http://172.18.135.223:8888/sayHelloUser/c  
    
  ## 压测结果：  
    
  Running 1m test @ http://172.18.135.223:8888/sayHelloUser/c  
   16 threads and 400 connections  
   Thread Stats Avg Stdev Max +/- Stdev  
   Latency 5.07ms 4.24ms 47.82ms 80.48%  
   Req/Sec 5.58k 410.08 7.55k 70.39%  
   Latency Distribution  
   50% 4.18ms  
   75% 7.24ms  
   90% 10.78ms  
   99% 18.90ms  
   5342895 requests in 1.00m, 845.83MB read  
  Requests/sec: 88899.94  
  Transfer/sec: 14.07MB
* 网关多模式压测：
* ## 压测服务参数命令：  
  cd /root/wrk && wrk -t 16 -c 1000 -d 60s --latency --timeout 5s -s /usr/local/dubbo.lua http://172.18.135.223:8888/sayHelloUser/c
* TCP(dubbo)压测结果：MPMC > 无缓存队列模式 > 双异步模式 > Fluster

|  |  |  |
| --- | --- | --- |
| * 网关服务模式 | * 压测方式 | * 压测结果 |
| * 单点：无缓存队列模式：原生Netty(Reactor模型) | * -t 16 -c 1000 | * QPS: 84927 |
| * 单点：无缓存队列，双异步模式：（请求异步、响应异步） | * -t 16 -c 1000 | * QPS: 77910 |
| * 单点：开启Fluster模式：disruptor | * -t 16 -c 1000 | * QPS: 77535 |
| * 单点：开启MPMC模式： | * -t 16 -c 1000 | * QPS: 85391 |

* SLB集群负载压测：
* ## SLB服务压测：  
  SLB服务地址：172.18.135.230  
    
  ## SLB负载 4层地址：172.18.135.230:80  
    
  ## 压测命令：  
  cd /root/wrk && wrk -t 16 -c 1000 -d 60s --latency --timeout 5s -s /usr/local/dubbo.lua http://172.18.135.230:80/sayHelloUser/c  
    
  ## 压测结果：  
  Running 1m test @ http://172.18.135.230:80/sayHelloUser/c  
   16 threads and 1000 connections  
   Thread Stats Avg Stdev Max +/- Stdev  
   Latency 29.31ms 74.02ms 908.05ms 89.31%  
   Req/Sec 20.19k 3.61k 60.63k 65.75%  
   Latency Distribution  
   50% 2.08ms  
   75% 6.36ms  
   90% 118.73ms  
   99% 331.57ms  
   19299899 requests in 1.00m, 2.98GB read  
  Requests/sec: 321132.65  
  Transfer/sec: 50.84MB  
    
    
  ## SLB负载 7层地址：172.18.135.230:8888  
    
  ## 压测命令：  
  cd /root/wrk && wrk -t 16 -c 1000 -d 60s --latency --timeout 5s -s /usr/local/dubbo.lua http://172.18.135.230:8888/sayHelloUser/c  
    
  ## 压测结果：  
  Running 1m test @ http://172.18.135.230:8888/sayHelloUser/c  
   16 threads and 1000 connections  
   Thread Stats Avg Stdev Max +/- Stdev  
   Latency 17.11ms 101.26ms 1.60s 98.31%  
   Req/Sec 14.79k 5.73k 31.58k 60.90%  
   Latency Distribution  
   50% 3.12ms  
   75% 6.52ms  
   90% 13.64ms  
   99% 579.86ms  
   13859227 requests in 1.00m, 4.38GB read  
   Non-2xx or 3xx responses: 10495682  
  Requests/sec: 230609.99  
  Transfer/sec: 74.61MB