3 K8S iptables/ipvs 压力测试

1.测试目的

在1 IPVS 调研和2 kube-proxy iptables 原理分析已经分析了iptables和ipvs，也指出了为什么要用ipvs, 接下来要在CCE环境下测试真实的数据，比较iptables/ipvs的性能。

2. 测试方法及过程

2.1 环境搭建

1. 集群：CCE 创建cluster, 一个work 节点， 配置： **4核(CPU) / 8GB(内存) / 40GB(系统盘)/ 普通型III**
2. k8s version:**v1.11.5**
3. 修改kubelet启动参数，vi /etc/systemd/system/kubelet.service, 添加**--max-pods=10000**，默认是110，如果用默认值的话，会发现当pod数量大于110的时候，集群中的新的pod会处于pending状态，会无法完成准确的测试。
4. 后端服务，采用**nginx:1.14.2** 版本。后端只有一个Ngxin pod, 通过不断累加service 数量来实现增加iptables规则，这样每增加**一个service，同样会增加7条rules**。 本来计划通过deploy service + pod 形式，后来实际测试发现，当k8s调度时，很快running的pod数量从0到64个，然后再以每分钟大概增加20个running的pod的速率增加。当一个node节点上的数量调度的pods 到了**241** 之后，不再增加，等了大概10分钟，为了节省时间，因此放弃这种测试。
5. 修改master 节点，node-port range，CCE采用默认值：-service-node-port-range， Default: 30000-32767， 如果用默认值，最多部署2767个node port 。在每个master 节点上 vi/etc/systemd/system/kube-apiserver.service, 添加--service-node-port-range=20000-65535 。 然后重启kube-apiserver服务

2.2 测试脚本

1.deploy script

```
#!/bin/bash
# MKDIR TMP
mkdir -p tmp
# actually the num of iptable rules is: iptables_num - 2*(services num)
# init iptables num
init_iptables_num=$(ssh root@192.168.3.250 "iptables-save | wc -l")
for i in {1000..2000}
do
    echo "${i}"
    # CREATE SERVICE
    service_name="nginx-test"-${i}
    tmp_service_yml=tmp/${service_name}.yml
    sed "s/test_name/${service_name}/g" nginx.yaml > ${tmp_service_yml}
    kubectl create -f ${tmp_service_yml} &>/dev/null
done
```

2. delete script:

```
#!/bin/bash
for i in {1000..2000}
do
    # remove SERVICE
    service_name="nginx-test"-${i}
    kubectl delete svc ${service_name}
    #kubectl delete deployments ${service_name}
done
```

2.3 测试方法

通过以下方法进行测试：

1. QPS: ab −n2000 −c100 http://192.168.3.233:node_port/

```
[root@instance-xbwb7at5-1 test]# ab -n2000 -c100 http://192.168.3.250:
31588/ | grep "Requests per second:"
Completed 200 requests
Completed 400 requests
Completed 600 requests
Completed 800 requests
Completed 1000 requests
Completed 1200 requests
Completed 1400 requests
Completed 1600 requests
Completed 1800 requests
Completed 2000 requests
Finished 2000 requests
Requests per second:    4589.65 [#/sec] (mean)
[root@instance-xbwb7at5-1 test]# ab -n2000 -c100 http://192.168.3.250:
32259/ | grep "Requests per second:"
Completed 200 requests
Completed 400 requests
Completed 600 requests
Completed 800 requests
Completed 1000 requests
Completed 1200 requests
Completed 1400 requests
Completed 1600 requests
Completed 1800 requests
Completed 2000 requests
Finished 2000 requests
Requests per second:    7651.37 [#/sec] (mean)
```

2. 规则匹配第一个Service和最后一个Service的时延：time curl 192.168.3.233:30956，根据： iptables −L KUBE−NODEPORTS −t nat −n −−line−number 去判断对应的nodePort是第几个service。

```
[root@instance-xbwb7at5-1 test]# time curl 192.168.3.250:31588 | grep
"real"
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
100   612  100   612    0     0   366k      0 --:--:-- --:--:-- --:--:
--  597k

real        0m0.008s
user        0m0.004s
sys        0m0.006s
[root@instance-xbwb7at5-1 test]# time curl 192.168.3.250:32259 | grep
"real"
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
100   612  100   612    0     0   310k      0 --:--:-- --:--:-- --:--:
--  597k

real        0m0.008s
user        0m0.003s
sys        0m0.006s
```

3. iptables 增加一条规则：time iptables −t nat −A POSTROUTING −s 172.8.8.9/32 −o eth0 −j MASQUERADE

```
[root@instance-nafy32t0 ~]# time iptables -t nat -A POSTROUTING -s
172.8.8.9/32 -o eth0 -j MASQUERADE

real        0m0.042s
user        0m0.005s
sys        0m0.032s
```

4. iptables 删除一条规则：timeiptables −t nat −D POSTROUTING3 ; (sudo iptables −L POSTROUTING −t nat −n −−line−number)

```
[root@instance-nafy32t0 ~]# time sudo iptables -t nat -D POSTROUTING 3

real 0m0.031s
user 0m0.004s
sys 0m0.025s
```

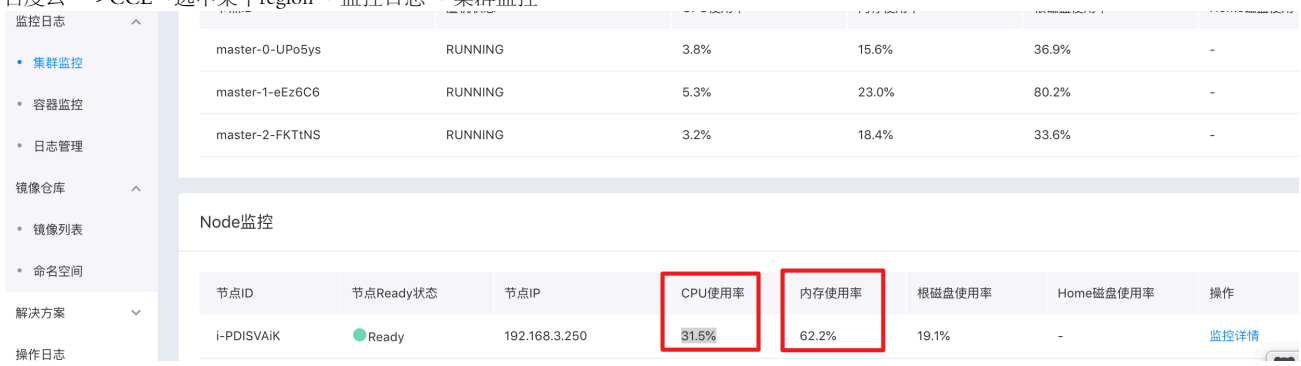
5. ipvs 增加一条规则

```
[root@instance-nafy32t0 ~]# time ipvsadm -a -t 127.0.0.1:30823 -r
172.18.4.14:80 -m

real 0m0.003s
user 0m0.001s
sys 0m0.001s
```

6. ipvs删除一条规则

```
[root@instance-nafy32t0 ~]# time ipvsadm -d -t 127.0.0.1:30823 -r
172.18.4.14:80

real        0m0.003s
user        0m0.003s
sys        0m0.000s
```

7. Node 节点 CPU/Memory 利用率
百度云-->CCE→选中某个region→ 监控日志→ 集群监控



每deploy一个service新增的iptables rules

```
-A KUBE-NODEPORTS -p tcp -m comment --comment "default/nginx-test-2:" -m
tcp --dport 30056 -j KUBE-MARK-MASQ
-A KUBE-NODEPORTS -p tcp -m comment --comment "default/nginx-test-2:" -m
tcp --dport 30056 -j KUBE-SVC-4AVFSJ4PJY464RGV
-A KUBE-SVC-4AVFSJ4PJY464RGV -m comment --comment "default/nginx-test-2:" -
j KUBE-SEP-SGTEPJLULDKV56EU
-A KUBE-SERVICES ! -s 172.18.0.0/16 -d 172.16.52.209/32 -p tcp -m comment
--comment "default/nginx-test-2: cluster IP" -m tcp --dport 80 -j KUBE-
MARK-MASQ
-A KUBE-SERVICES -d 172.16.52.209/32 -p tcp -m comment --comment "default
/nginx-test-2: cluster IP" -m tcp --dport 80 -j KUBE-SVC-4AVFSJ4PJY464RGV
-A KUBE-SEP-SGTEPJLULDKV56EU -s 172.18.4.14/32 -m comment --comment
"default/nginx-test-2:" -j KUBE-MARK-MASQ
-A KUBE-SEP-SGTEPJLULDKV56EU -p tcp -m comment --comment "default/nginx-
test-2:" -m tcp -j DNAT --to-destination 172.18.4.14:80
```

每deploy 一个service 新增5条rules, 分布针对，localhost, cluster ip, docker0, cbr0, eth0

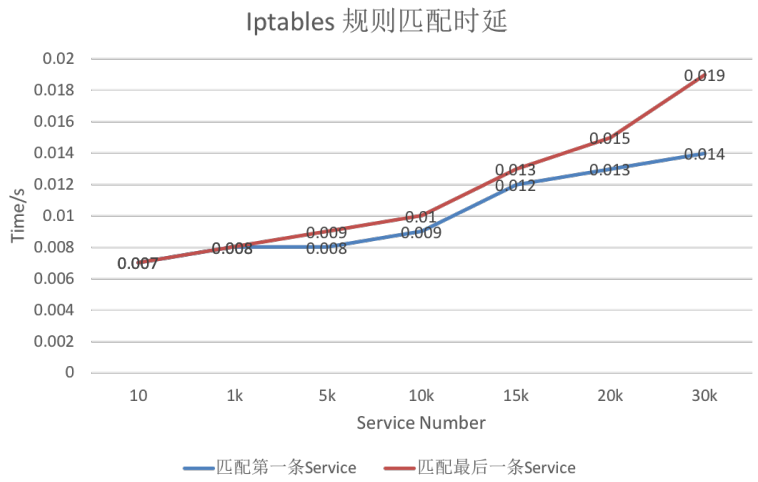
```
TCP  127.0.0.1:41113 rr
  -> 172.18.4.20:80               Masq    1      0          0
TCP  172.16.194.97:80 rr
  -> 172.18.4.20:80               Masq    1      0          0
TCP  172.17.0.1:41113 rr
  -> 172.18.4.20:80               Masq    1      0          0
TCP  172.18.4.1:41113 rr
  -> 172.18.4.20:80               Masq    1      0          0
TCP  192.168.3.250:41113 rr
  -> 172.18.4.20:80               Masq    1      0          0
```

2.4 测试结果及分析

2.4.1 Iptables 结果

service数量	10	1k	5k	10k	15k	20k	30k	结果分析
Iptables 规则数目	142	7145	35145	70145	105145	140145	210145	
Iptables 新增条数	1*7	3500	14000	14000	35000	35000	70000	
Node Memory usage	16%	20%	32%	47.1%	56.1%	62.2%	81.6%	逐渐增加
Node CPU usage	7%	7%	17%	21.2%	25.2%	31.5%	45.7%	逐渐增加
Qps(fist service)	7780	4589	2140	2566	668	723	318	逐渐减小
Qps(last service)	7782	7651	2451	2066	393	252	236	逐渐减少要比匹配fist service还要小
iptables规则匹配时延(first service)	0.007	0.008	0.008	0.009	0.012	0.013	0.014	增加的幅度非常小
iptables规则匹配时延(last service)，每秒	0.007	0.008	0.009	0.010	0.013	0.015	0.019	last service的匹配时延相比较first service要大，但是差距不是很明显差距。大概每增加10k service 时延增加4ms
Iptables 规则增加时延(增加一条规则)，每秒	0.010	0.113	0.587	3.621	5.432	7.081	11.201	当iptables增加较多时，每增加一次iptables规则花费时间较多
Iptables 规则删除时延(删除一条规则)，每秒	0.010	0.132	0.611	3.471	5.478	7.012	15.510	同上

iptables 匹配时延 (fist service 和 last service) 趋势图如下，匹配最后一条Service 数量的延时，大概每增加**10K service时延增加4ms**。这个数量其实不大，但对于QPS（每秒处理的请求数），会比较大。

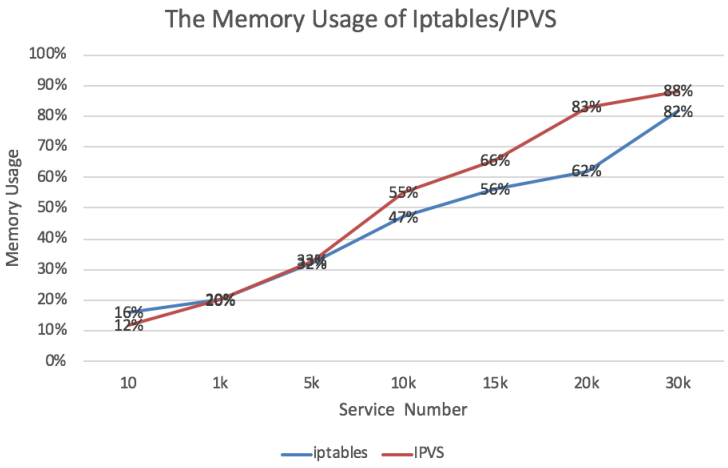


2.4.2 ipvs 结果

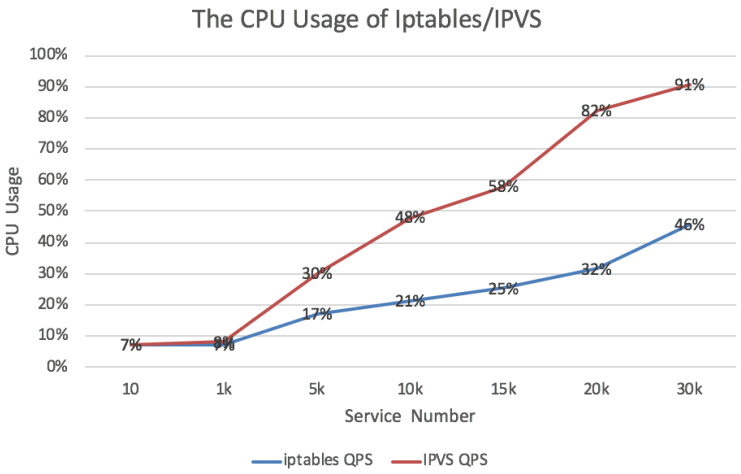
Service 数量	10	1k	5000	10k	15k	20k	30k	结果分析
Memory usage	11.8%	20.2%	33.4%	55.1%	65.5%	83.0%	88.3%	逐渐增加
CPU usage	7%	8%	40.5%	48.1%	57.9%	82.2%	90.5%	逐渐增加，消耗的比较多
Qps(match last service)	7085	7030	7109	6348	5700	5435	4006	QPS减少幅度不大
ipvs规则匹配时延，每秒	0.006	0.006	0.007	0.007	0.008	0.008	0.011	基本不增加，在30K的时候，是因为CPU比较高，导致时延大
ipvs规则增加时延，每秒	0.002	0.02	0.002	0.002	0.002	0.002	0.011	基本不增加，30K的时候，是因为CPU比较高，导致增加多
ipvs规则删除时延，每秒	0.002	0.02	0.002	0.007	0.002	0.003	0.013	同上

2.4.2 iptables/ipvs结果比较分析

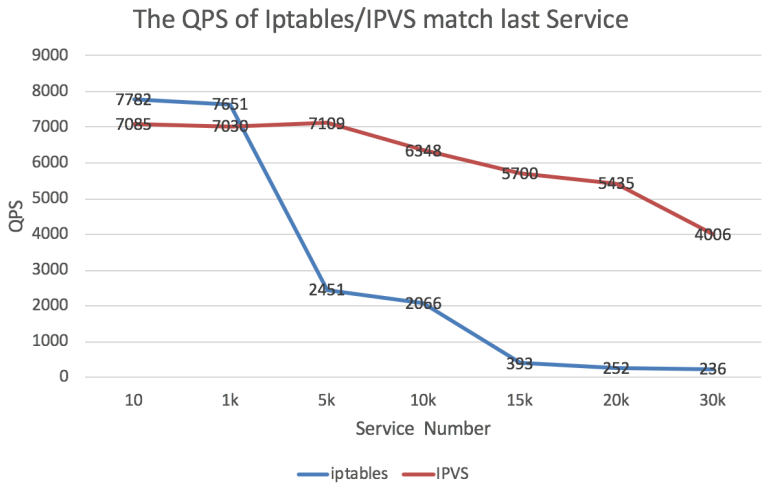
1. Memory 比较。随着Service数量增加，递增趋势，整体上ipvs消耗的memory要大。



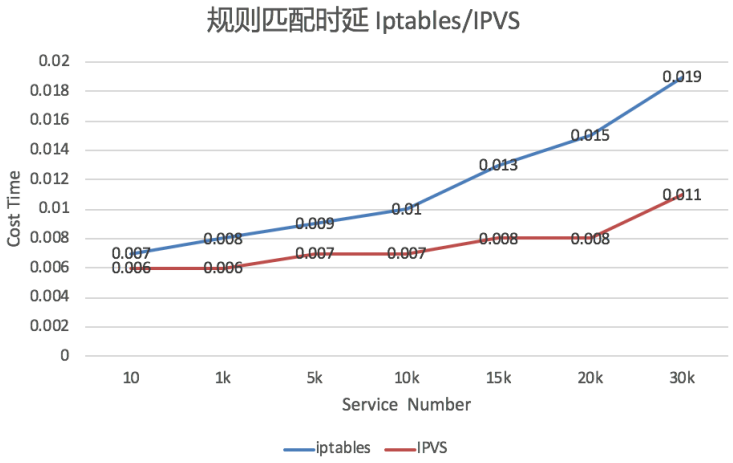
2. CPU 比较。随着Service数量增加，递增趋势，ipvs消耗的CPU要大很多，当serivce数量是30k的时候，CPU Usage 打到90%



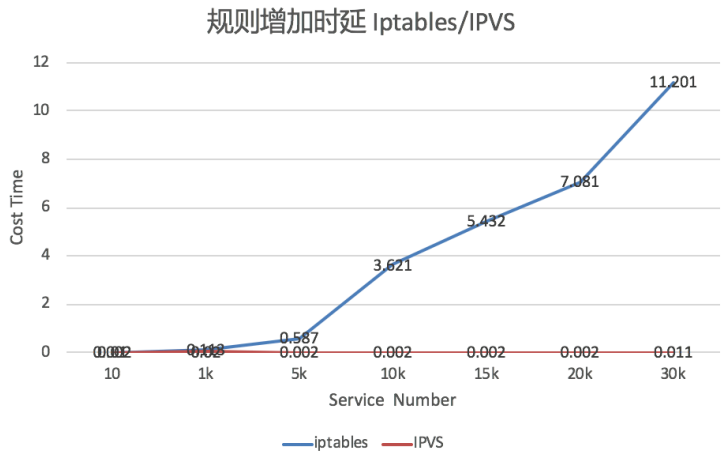
3. QPS last service比较。显然随着Service增加，下降趋势。但是IPVS要好过iptables 20倍左右。



4. 规则匹配时延比较。匹配时延，增加趋势，iptables要大于ipvs。



5. 规则增加时延比较，iptables是增加趋势，并且现象比较明显，k8s iptables 的更新会花费大部分时间。



3 测试中发现的问题

1. 创建了1000个pods后，当删除pods时，不能删除成功，所有的pod处于terminating 状态。不得不用命令：
kubectl delete --all pods --grace-period=0 --force, 但是这种方式不会完全删除container。最后强制删除container。
2. 当service数量大于10k之后，发现添加，删除规会经常出现如下，原因是默认每隔30s 更新一次iptables规则，后面改成了每隔--iptables-sync-period=180， 但是改成这个之后会出现iptables的数量增加非常缓慢。

```
[root@instance-nafy32t0 ~]# time iptables -t nat -A POSTROUTING -s
172.8.8.9/32 -o eth0 -j MASQUERADE
Another app is currently holding the xtables lock. Perhaps you want
to use the -w option?

real 0m0.002s
user 0m0.001s
sys 0m0.000s
```

3. 当service数量大于10k的时候，当从iptables模式切换到Ipvs模式时，ipvsadm的rules的规则需要话费大量的时候才能同步完。大概要3个小时左右。

4 总结

1. 从数据测试结果来看，随着service数量增加，ipvs的QPS性能远远大于iptables，但同时IPVS 消耗的CPU也随之增加。但是在service数量小于1k，iptables/ipvs的性能相差不是很大。
2. 从自己测试的数据上来看，测试结果和华为测试结果有些不同，华为结果。

测试内容	华为结果	自己测试结果	分析
20k Service（160k 规则 增加一条iptables规则）耗时时间	5h	7s	差距很大，实际测试来看，并没有花费很长时间
20k Service 增加一条ipvs 规则耗时时间	70us	0.002s	
20K Service CPU 利用率 IPVS	0%	ipvs模式下CPU利用率要比iptables模式下高50%	华为的测试结果是0%, 非常让人费解。实际测测试结果来看，ipvs模式下消耗的CPU要更高。

3. 建议，在创建CCE集群时，可以提供给用户选择kube-proxy的proxy-mode时iptables/ipvs，默认还是iptables模式。
4. k8s 建议的 services 数量是10K, refer:k8s