# 双主keydb节点的HA读写测试

# 背景

## 目标

使用keydb来替代redis构建主-主replication或HA redis集群的两个或三个服务器的中小型项目

## Redis的缺点

Redis社区版本的多主机架构有点复杂，它确实需要配置许多主节点和从节点。此外，所有好的额外功能也仅在商业Redis版本中可用。

## Keydb的简介

Keydb是Redis的高性能分支，与Redis完全兼容，但也有一些额外的功能：

多线程，内存效率和高吞吐量。具有仅在Redis Enterprise中可用的功能，例如Active Replication，FLASH存储支持以及一些根本不可用的功能，例如直接备份到AWS S3。

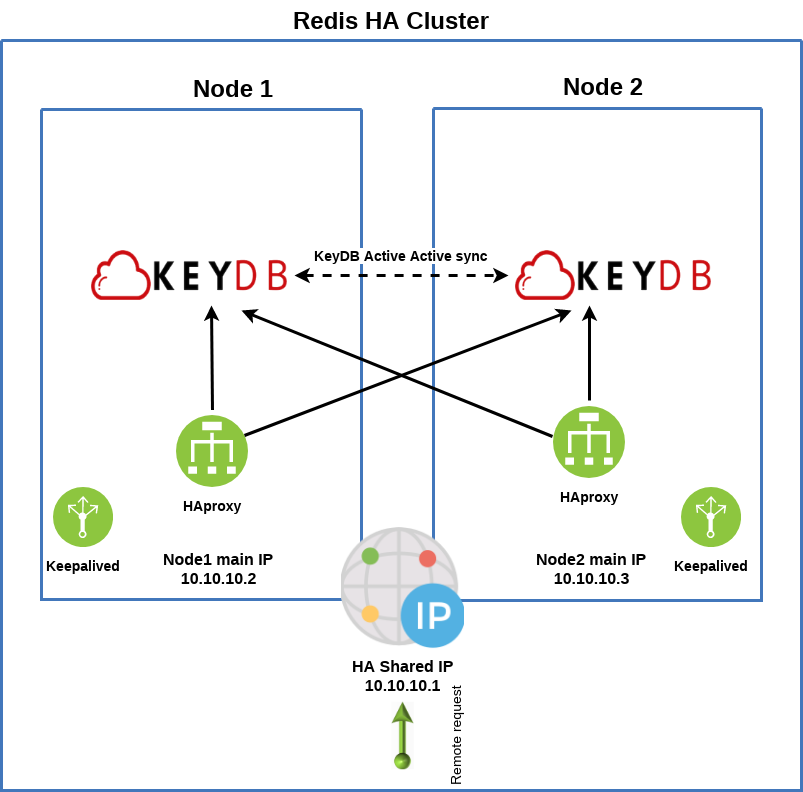
性能上：在相同的硬件上，KeyDB每秒可以执行的查询数量是Redis的两倍，而延迟却降低了60％。

## 架构示例

两个主节点，完全同步、高可用性数据，并具有负载均衡

即使一个节点完全丢失，该集群也能继续工作，并该节点的修复过程对于外部应用程序是透明的。

### 架构图示例



两节点上安装运行keydb的docker容器、HAProxy以及keepalived

### 架构要点

Keepalived负责其中两个节点之间的一个共享IP、该IP讲进行故障转移。如果一个节点宕机，则将使用该IP并继续处理请求。

下一级中，HAProxy将侦听此故障转移IP（示例中为10.10.10.1）.首先将获得该故障转移IP的节点的HAProxy会作为LB服务所有请求，，而第二个将处于热备用状态。然后，某些外部应用程序将使用此故障转移IP作为Redis密钥存储。

活动的HAproxy将使用简单的循环负载平衡在两个节点上的两个KeyDB服务之间分发请求。最后，KeyDB将以“ active-replica yes ”参数启动，这会将KeyDB服务配置为彼此的活动副本。

*默认情况下，KeyDB与Redis一样，仅允许从主数据库到副本数据库的单向通信。添加了新的配置选项“ active-replica”，当设置为true时，还意味着“ replica-read-only no”。在这种模式下，即使KeyDB与主数据库的连接断开，它也将接受副本。它还将允许在两个节点彼此为主的情况下进行循环连接。*

这意味着,比如第一个请求服务于Node1上的KeyDB，然后第二个请求将服务于Node2上的KeyDB，依此类推。同样，KeyDB服务将在两者之间具有完全复制的数据库。如果Node1发生故障，Node2将在Keepalived的帮助下获得10.10.10.1 IP，Node2上的HaProxy将继续向Node2上的活动KeyDB发送请求，直到Node1回来。

因此，配置的环境是：两个keyDB服务且在master-master模式下运行相同的数据库

### 配置说明

在root下运行所有命令、centOS7 x64系统

Node1 192.168.174.11

Node2 192.168.174.19

Node3 192.168.174.20

Vip 192.168.174.99

Test\_machine 192.168.174.10

# 安装keepalived

node1# yum -y install keepalived

node2# yum -y install keepalived

node3# yum -y install keepalived

## 配置keepalived

node1# vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

root@localhost

}

notification\_email\_from root@localhost

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

vrrp\_skip\_check\_adv\_addr

vrrp\_strict

vrrp\_garp\_interval 0

vrrp\_gna\_interval 0

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33

virtual\_router\_id 101

priority 101

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.174.99

}

}

virtual\_server 192.168.174.99 32 {

delay\_loop 6

lb\_algo rr

lb\_kind NAT

persistence\_timeout 50

protocol TCP

real\_server 192.168.174.11 32 {

weight 1

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 32

}

real\_server 192.168.174.19 32 {

weight 1

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 32

}

real\_server 192.168.174.20 32 {

weight 1

connect\_timeout 3

nb\_get\_retry 3

delay\_before\_retry 3

connect\_port 32

}

}

在第二个节点上，配置几乎相同，但是我们需要将优先级参数设置得较低，因此node1首先将成为主节点。同样重要的是，两个节点的virtual\_router\_id应该相同。

## 在两节点上启动keepalived

node1# systemctl enable keepalived

node1# systemctl start keepalived

node2# systemctl enable keepalived

node2# systemctl start keepalived

### 检查状态

然后检查服务状态，并确保在第一个节点上也添加了故障转移IP。

[root@localhost ~]# systemctl status keepalived

● keepalived.service - LVS and VRRP High Availability Monitor

Loaded: loaded (/usr/lib/systemd/system/keepalived.service; enabled; vendor preset: disabled)

Active: active (running) since Tue 2020-02-18 16:44:03 CST; 21s ago

Process: 41453 ExecStart=/usr/sbin/keepalived $KEEPALIVED\_OPTIONS (code=exited, status=0/SUCCESS)

Main PID: 41454 (keepalived)

CGroup: /system.slice/keepalived.service

├─41454 /usr/sbin/keepalived -D

├─41455 /usr/sbin/keepalived -D

└─41456 /usr/sbin/keepalived -D

Feb 18 16:44:08 localhost.localdomain Keepalived\_vrrp[41456]: Sending gratuitous ARP on ...

…

[root@localhost ~]# ip a

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000

link/ether 00:0c:29:12:56:70 brd ff:ff:ff:ff:ff:ff

inet 192.168.174.15/24 brd 192.168.174.255 scope global noprefixroute dynamic ens33

valid\_lft 1335sec preferred\_lft 1335sec

inet 192.168.174.99/32 scope global ens33

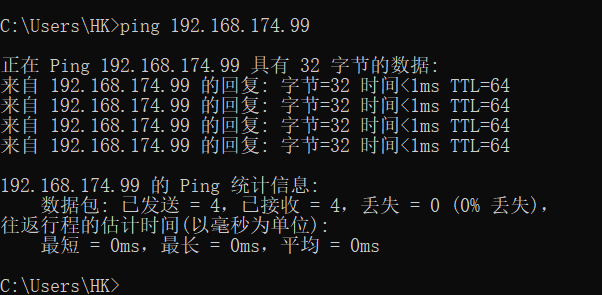
valid\_lft forever preferred\_lft forever

inet6 fe80::e725:5e85:ca98:deb1/64 scope link noprefixroute

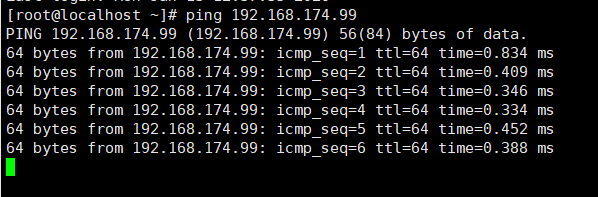
valid\_lft forever preferred\_lft forever

### 检查vip

1. 在物理机上可以ping通



1. 在另外的虚机(test\_machine)上ping



然后重启node1的keepalived服务，发现ping没有中断

# 安装keydb

可以选择像redis那样源码安装，也可以选择采用docker方式安装

这里采用dockers方式

## 安装docker



## 安装keydb

docker pull eqalpha/keydb

下载[初始keydb.conf](https://github.com/JohnSully/KeyDB/blob/unstable/keydb.conf)文件到/var/lib/docker目录下，并修改：

[docker install keydb](https://docs.keydb.dev/docs/docker-basics/)提示如果您使用自己的配置文件，请记住将“ bind 127.0.0.1”注释掉，将“ protected-mode”从“ yes”更改为“ no”。

### 运行eqalpha/keydb容器

#### 以master-master方式运行，并让彼此互为replica

node1# docker run --name keydb -v /var/lib/docker/keydb.conf:/etc/keydb/keydb.conf --restart=always -d -p 6380:6379 eqalpha/keydb keydb-server /etc/keydb/keydb.conf --active-replica yes --replicaof 192.168.174.19 6380

node2# docker run --name keydb -v /var/lib/docker/keydb.conf:/etc/keydb/keydb.conf --restart=always -d -p 6380:6379 eqalpha/keydb keydb-server /etc/keydb/keydb.conf --active-replica yes --replicaof 192.168.174.11 6380

参数项说明：

--name 表示容器的别名

-v pathA:pathB表示挂载目录映射，前者为主机部分，后者为容器部分

-p 6380:6379 端口映射，前表示主机部分，：后表示容器部分

-d 后台启动keydb

keydb-server /etc/keydb/keydb.conf 以配置文件启动keydb，加载容器内的conf文件，最终找到的是挂载的目录/usr/local/docker/keydb.conf

active-replica yes --replicaof 192.168.174.11 6380，启用active-replica，副本IP为 192.168.174.19(1) 端口为6380

#### 查看进程

[root@localhost bin]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

0a5b2d881f95 eqalpha/keydb "docker-entrypoint.s…" 2 hours ago Up 2 hours 0.0.0.0:6380->6379/tcp keydb

按以上方式启动后，keydb将在公用IP（192.168.174.11）上通过端口6380访问，因为与redis兼容，所以能够通过redis-cli来访问keydb，以下先确保两节点上安装有redis，再测试master-master的读写。

## 在两节点上安装redis

以node1为例

1. [获取redis安装包](https://redis.io/download)，假设下载至/usr/local目录下，并解压
2. 必要的工具gcc，yum -y install gcc
3. 进入redis-xxx目录里，执行编译命令

make

如果遇到以下错误

fatal error: jemalloc/jemalloc.h: No such file or directory

#include <jemalloc/jemalloc.h>

^

compilation terminated.

make[1]: \*\*\* [adlist.o] Error 1

make[1]: Leaving directory `/usr/local/redis-5.0-rc3/src'

make: \*\*\* [all] Error 2

解决：

cd /usr/local/redis-xxx/deps;

make hiredis lua jemalloc linenoise

# 重新返回redis-xxx目录下编译

cd ..

make

1. 编译完成之后，将redis安装到指定目录

make PREFIX=/usr/local/redis install

1. 启动
2. 前台启动方式

前台启动,不推荐使用，进入/usr/local/redis/bin里执行启动命令(默认端口号:6379)

./redis-server

1. 后台启动方式

后台启动，推荐使用，将redis-xxx目录下的redis.conf文件复制到 /usr/local/redis/bin 下。修改redis.conf 设置为后台启动，将daemonize no改为daemonize yes即可

【可选】允许远程连接Redis

redis 默认只允许自己的电脑（127.0.0.1）连接。如果想要其他电脑进行远程连接，将配置文件 redis.conf 中的 bind 127.0.0.1 注释掉(之前没注释，需要改为将其注释掉，默认只能连接本地)。同时需要找到配置文件redis.conf中protected mode，默认protected mode yes，需要将其改为protected mode no。

1. 防火墙

参考自[How to Install and Configure Redis on CentOS 7](https://linuxize.com/post/how-to-install-and-configure-redis-on-centos-7/)

sudo firewall-cmd --new-zone=redis --permanent

sudo firewall-cmd --zone=redis --add-port=6379/tcp --permanent

sudo firewall-cmd --zone=redis --add-source=xxx.xxx.xxx.0/24 --permanent

sudo firewall-cmd –reload

1. 测试master-master读写

假设两节点上的redis已安装

首先

node1# redis-cli -p 6380

127.0.0.1:6380> set k1 1

OK

node2# redis-cli -p 6380

127.0.0.1:6380> get k1

"1"

然后

node2# 127.0.0.1:6380> set k2 2

OK

node1# 127.0.0.1:6380> get k2

"2"

使用HAProxy为两个KeyDB服务器配置负载平衡

在两节点上安装HAPorxy

1. 修改/etc/sysctl.conf

node1＃vi /etc/sysctl.conf

net.ipv4.ip\_nonlocal\_bind = 1

node2＃vi /etc/sysctl.conf

net.ipv4.ip\_nonlocal\_bind = 1

# 然后

node1＃sysctl -p

node2＃sysctl -p

1. 安装
2. 配置HAProxy