**高并发下，如何保证接口的幂等性？**

**幂等性**就是同一个操作执行多次，产生的效果一样。如 http 的 get 请求，数据库的 select 请求就是幂等的

在分布式系统中，保证接口的幂等性非常重要，如提交订单，扣款等接口都要保证幂等性，不然会造成重复创建订单，重复扣款，那么如何保证接口的幂等性呢？

                      ​**前端保证幂等性的方法**

**按钮只能点击一次**

用户点击按钮后将按钮置灰，或者显示 loading 状态

**RPG 模式**

即 Post-Redirect-Get，当客户提交表单后，去执行一个客户端的重定向，转到提交成功页面。避免用户按 F5 刷新导致的重复提交，也能消除按浏览器后退键导致的重复提交问题。目前绝大多数公司都是这样做的，比如淘宝，京东等

**后端保证幂等性的方法**

​**使用唯一索引**

对业务唯一的字段加上唯一索引，这样当数据重复时，插入数据库会抛异常

**状态机幂等**

如果业务上需要修改订单状态，例如订单状态有待支付，支付中，支付成功，支付失败。设计时最好只支持状态的单向改变。这样在更新的时候就可以加上条件，多次调用也只会执行一次。例如想把订单状态更新为支持成功，则之前的状态必须为支付中

updatetable\_namesetstatus=支付成功 wherestatus=支付中  
**乐观锁实现幂等**

查询数据获得版本号  
通过版本号去更新，版本号匹配则更新，版本号不匹配则不更新  
--假如查询出的 version 为 1  
selectversionfromtable\_namewhereuserid=10;  
--给用户的账户加 10  
updatetable\_namesetmoney=money-10,version=version+1whereuserid=10andversion=1  
也可以通过条件来实现乐观锁，如库存不能超卖，数量不能小于 0

updatetable\_namesetnum=num-10wherenum-10>=0

**防重表**

增加一个防重表，业务唯一的 id 作为唯一索引，如订单号，当想针对订单做一系列操作时，可以向防重表中插入一条记录，插入成功，执行后续操作，插入失败，则不执行后续操作。本质上可以看成是基于 MySQL 实现的分布式锁。根据业务场景决定执行成功后，是否删除防重表中对应的数据

**分布式锁实现幂等**

执行方法时，先根据业务唯一的 id 获取分布式锁，获取成功，则执行，失败则不执行。分布式锁可以基于 redis，zookeeper，mysql 来实现，分布式锁的细节就不介绍了

**select+insert**

先查询一下有没有符合要求的数据，如果没有再执行插入。没有并发的系统中可以保证幂等性，高并发下不要用这种方法，也会造成数据的重复插入。我一般做消息幂等的时候就是先 select，有数据直接返回，没有数据加分布式锁进行 insert 操作

**全局唯一号实现幂等**

通过 source（来源）+seq（序列号）来判断请求是否重复，重复则直接返回请求重复提交，否则执行。如当多个三方系统调用服务的时候，就可以采用这种方式

# LINUX 从网卡入手限制带宽

网卡限速

这个服务的名称叫做：wondershaper

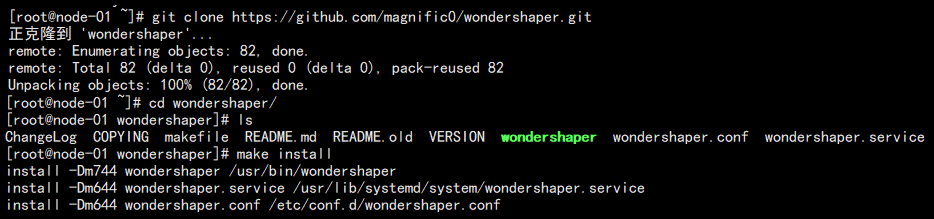
<https://github.com/magnific0/wondershaper>

安装：

git clone https://github.com/magnific0/wondershaper.git

cd wondershaper/

make install



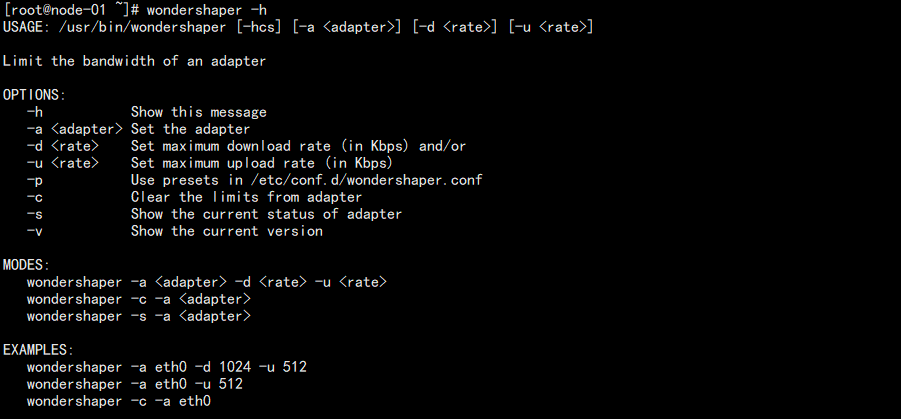
启动服务并设置开机启动：

systemctl start wondershaper.service

systemctl enable wondershaper.service

查看该服务能做啥：

wondershaper -h



可以看到 -d 是下载限速，-u 是上传限速，-a 是指定网卡，-c 清除限速。这就是我们最常用的三个参数。

然后做个限速测试，我的网卡是 ens33：

wondershaper -a ens33 -d 100 -u 100

除了使用命令配置限速，也可以修改配置文件：

vim /etc/conf.d/wondershaper.conf

[wondershaper]

# Adapter

#

IFACE="ens33"

# Download rate in Kbps

#

DSPEED="100"

# Upload rate in Kbps

#

USPEED="100"

当然，也可以解除限速：

wondershaper -c -a ens33