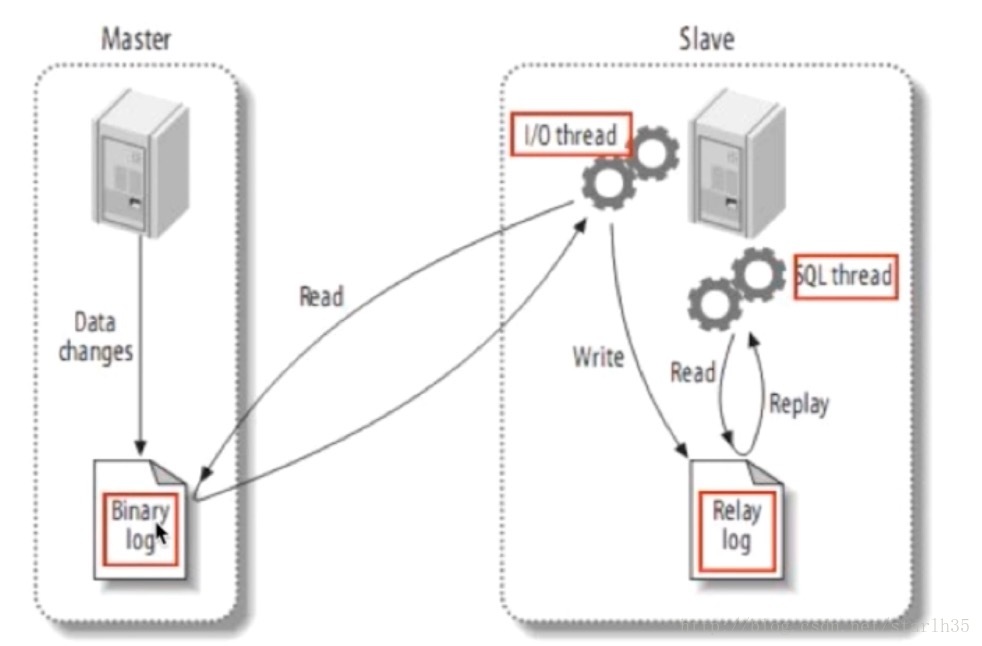
**主从同步原理**

主服务器master记录数据库操作日志到Binary log，从服务器开启i/o线程将二进制日志记录的操作同步到relay log（存在从服务器的缓存中），另外sql线程将relay log日志记录的操作在从服务器执行。

****

**主从配置**

#### **.在主服务器master上配置开启Binary log，主要是在[mysqld]下面添加：**

[root@host51 ~]# vim /etc/my.cnf

[mysqld]

validate\_password\_policy=0

validate\_password\_length=6

server\_id=51

log-bin=master51         -----开启binglog日志

binlog-format="mixed"

#### **配置从服务器的 my.cnf**

从库配置详细过程：

[root@host52 ~]# vim /etc/my.cnf

[mysqld]

validate\_password\_policy=0

validate\_password\_length=6

server\_id=52

[root@host52 ~]# mysql -h 192.168.4.51 -urepluser -p123456

mysql: [Warning] Using a password on the command line interface can be insecure.

Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 4

Server version: 5.7.17-log MySQL Community Server (GPL)

mysql> change master to

    -> master\_host="192.168.4.51",

    -> master\_user="repluser",

    -> master\_password="123456",

    -> master\_log\_file="123456",

    -> master\_log\_pos=452;

Query OK, 0 rows affected, 2 warnings (0.35 sec)

**1..一主多从，主库宕机，如何切换到从库，其他的从库如何处理？**

1.确保所有的relay log全部更新完毕，在每个从库上执行stop slave io\_thread; show processlist;直到看到Has read all relay log,则表示从库更新都执行完毕了

2.登陆所有从库，查看master.info文件，对比选择pos最大的作为新的主库。

3.登陆192.168.1.102，执行stop slave; 并进入数据库目录，删除master.info和relay-log.info文件, 配置my.cnf文件，开启log-bin,如果有log-slaves-updates和read-only则要注释掉，执行reset master

4.创建用于同步的用户并授权slave，同第五大步骤

5.登录另外一台从库，执行stop slave停止同步

6.根据第七大步骤连接到新的主库

7.执行start slave;

8.修改新的master数据，测试slave是否同步更新

**2.IDC 机房带宽突然从平时 100M 增加到 400M，请你分析问题所在，并解决。**

.真实遭受DDOS攻击（遇到过几次，造成影响的不多见，其中还有黑客勒索的案例）。

b.内部服务器中毒，大量外发流量（这个问题老男孩接警5次以上）

c.网站元素（如图片）被盗连，在门户页面被推广导致大量流量产生（接警3次以上）

d.合作公司来抓数据，如：对合作单位提供了API数据接口（有合作的公司的朋友了解这个）

e.购买了CDN业务，CDN猛抓源站（这个次数也不少）。

**3..apache 服务的常用工作模式及对应特点，企业如何选择对应模式。**

apache的工作模式有：beos，event，worker，prefork，mpmt\_os2。

    查看：http –l  
          apachectl –l

**beos工作模式（跟linux关系不大，或者暂时用不上）**  
    在Beos系统上的工作模式，使用一个单独的控制线程来创建和控制处理请求的工作线程。

**event工作模式（不太稳定，或者说暂时用不上）**    event 模式由于把服务进程从链接中分离出来，在开启KeepAlive场合下相对worker模式能够承受的了更高的负载。event模式为 worker开发的变种模式，配置以及指令与worker完全相同。不过event模式不能很好的支持https的访问，有时还会出现一系列的问题。

**worker工作模式（与php配合不好）**  
     worker模式由于使用线程来进行处理请求，所以能够处理海量请求，而系统资源的开销要小于基于进程的服务器。同时worker模式也使用了多进程，每个进程又有着多个线程，以获得基于进程服务器的稳定性。     
    **mpmt\_os2工作模式（很少用，或者说暂时用不上）**  
    mpmt\_os2是专门针对OS/2优化过的混合多进程多线程多路处理模块(MPM) 。

**重点：prefork工作模式**。  
    1、编译的时候使用#–with-mpm=prefork对应的工作模式名称来修改工作模式。  
    2、prefork工作模式是linux下apache安装时候的默认工作模式，是使用最普遍的工作模式。  
    3、原理：有一台正在运行的apache服务器，用户A访问该apache的时候apache建立一个新的进程1处理用户A的请求。这时又有一个用户B访问该apache，apache又建立一个新的进程2处理用户B的请求。后来又有用户C，D，E访问该apache，apache又建立三个进程3，4，5处理他们的请求。如果每当一个新用户访问该apache，apache再建立一个新的进程处理用户的请求，是不是太慢了呢？所以apache的prefork模式在apache第一次启动的时候就建立5个进程，等待用户的连接请求，有一个用户访问，就有一个进程处理他的请求。那么如果有5个用户同时访问apache，apache第一次建立的5个进程全部用光了，所以apache就再从新在建立5个进程，等待下一批用户的请求。 prefork模式会根据服务器的硬件情况，设定apache最多只能同时建立256个进程。再多的请求就只能等待前面的进程处理完毕在进行处理。