一，客户需求背景：

> maglev平台中应用udc使用了mysql存储，当前是部署在k8s集群中（operator部署1主1从），经过测试，当master出现异常的时候，有可能会导致数据库变成单点而存在风险。所以需要将Mysql实例迁移到虚机部署，方便进行维护。

二，客户需求分析：

2.1 原mysql版本为mysql-5.7.35版本，部署在k8s集群中；

2.2 新的mysql架构为1主1从架构，采用主从复制，需要部署在虚拟机上；

2.3 虚拟机配置要求：

cpu	内存	磁盘
2	4G	100G

2.4 需要将原mysql的数据导入到新的mysql主从集群；

2.5 备份周期要求为每天全量备份；

2.6 需要对mysql的运行状态进行监控。

三，msyql主从搭建步骤

3.1 在stack上开通两台虚机,虚拟机的情况如下：

虚机编号	虚拟机系统	ip(假设)	cpu	内存	磁盘
mysql-master	centos7.9	172.16.10.10	2	4G	100G
mysql-slave	centos7.9	172.16.10.11	2	4G	100G

3.2 对linux虚拟机进行性能优化：

3.2.1 关闭linux防火墙

3.2.2 修改系统资源限制

```
vim /etc/security/limits.conf

* soft nofile 65535
* hard nofile 65535
* soft nproc 65535
* hard nproc 65535
```

3.2.3 内核优化

```
# vim /etc/sysctl.conf

#关闭ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1

# 避免放大攻击
net.ipv4.icmp_echo_ignore_broadcasts = 1

# 开启恶意icmp错误消息保护
net.ipv4.icmp_ignore_bogus_error_responses = 11

#决定检查过期多久邻居条目
net.ipv4.neigh.default.gc_stale_time=120

#使用arp_announce / arp_ignore解决ARP映射问题
net.ipv4.conf.default.arp_announce = 2
net.ipv4.conf.all.arp_announce=2
net.ipv4.conf.lo.arp_announce=2 # 避免放大攻击
net.ipv4.icmp_echo_ignore_broadcasts = 1 # 开启恶意icmp错误消息保护
net.ipv4.icmp_ignore_bogus_error_responses = 1

#处理无源路由的包
net.ipv4.conf.all.accept_source_route = 0
```

```
net.ipv4.conf.default.accept_source_route = 0

#core文件名中添加pid作为扩展名
kernel.core_uses_pid = 1

#修改消息队列长度
kernel.msgmnb = 65536
kernel.msgmax = 65536

#开启反向路径过滤
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

#设置最大内存共享段大小bytes
kernel.shmmax = 68719476736
kernel.shmall = 4294967296

#timewait的数量，默认180000
net.ipv4.tcp_max_tw_buckets = 6000
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_rmem = 4096 87380 4194304
net.ipv4.tcp_wmem = 4096 16384 4194304
net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216

#每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许送到队列的数据包的最大数目。
net.core.netdev_max_backlog = 32768

#记录的那些尚未收到客户端确认信息的连接请求的最大值。对于有128M内存的系统而言，缺省值是1024
net.ipv4.tcp_max_syn_backlog = 65536

#web应用中listen函数的backlog默认会给我们内核参数的net.core.somaxconn限制到128，而nginx定义的NGX_LISTEN_BACKLOG默认为511，所以有必要调整这个值。
net.core.somaxconn = 32768

#限制仅仅是为了防止简单的DoS 攻击
net.ipv4.tcp_max_orphans = 3276800

#未收到客户端确认信息的连接请求的最大值
net.ipv4.tcp_max_syn_backlog = 262144
net.ipv4.tcp_timestamps = 0

#内核放弃建立连接之前发送SYNACK 包的数量
net.ipv4.tcp_synack_retries = 1

#内核放弃建立连接之前发送SYN 包的数量
net.ipv4.tcp_syn_retries = 1

#启用timewait 快速回收
net.ipv4.tcp_tw_recycle = 1

#开启重用。允许将TIME-WAIT sockets 重新用于新的TCP 连接
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_mem = 94500000 915000000 927000000
net.ipv4.tcp_fin_timeout = 1

# 开启SYN洪水攻击保护(防范少量SYN攻击)
net.ipv4.tcp_syncookies = 1

#允许系统打开的端口范围
net.ipv4.ip_local_port_range = 1024    65000


sysctl -p使参数生效
```

3.2.4 同步内网系统时钟

```
设置定时任务，同步内网时钟
*/5 * * * * /usr/sbin/ntpdata -u ntp.addpchina.com
```

3.2.5 修改主机名称

```
hostnamectl set-hostname mysql-master
hostnamectl set-hostname mysql-slave
```

3.3 mysql主从安装

3.3.1从MySQL官网下载和客户使用相对应的版本

> 官网：https://downloads.mysql.com/archives/community/
> 下载Compressed TAR Archive版本：https://downloads.mysql.com/archives/get/p/23/file/mysql-5.7.39-el7-x86_64.tar.gz

3.3.2 安装mysql相关依赖

```
yum install curl policycoreutils openssh-server openssh-clients postfix libaio -y
```

3.3.3 创建mysql数据目录并将下载的mysql文件进行解压

```
mkdir /data/mysql
tar xvf mysql-5.7.39-el7-x86_64.tar.gz -C /usr/local/mysql
```

3.3.4 创建mysql用户和组,修改相关文件夹的权限

```
useradd  mysql  -s /sbin/nologin
chown  -R mysql.mysql  /usr/local/mysql/*
chown  -R mysql.mysql  /data/mysql
```

3.3.5编辑mysql配置文件 /etc/my.cnf

> 注意：配置文件需要和客户所使用mysql保持一致，主从的server-id不能一样

- mysql-master /etc/my.cnf配置

```
[mysqld]
expire-logs-days               = 14
query-cache-size               = 0
query-cache-type               = 0

sql-mode                       =
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_AUTO_VALUE_ON_ZERO,NO_ENGINE_SUBSTITUTION,NO_ZERO_DATE,NO_ZERO_IN_DATE,ONL
Y_FULL_GROUP_BY

skip-host-cache
skip-name-resolve
binlog-format                  = ROW
character-set-server           = utf8mb4
collation-server               = utf8mb4_unicode_ci
default-storage-engine         = InnoDB
enforce-gtid-consistency       = on
gtid-mode                      = on
innodb-file-per-table          = 1
innodb-flush-log-at-trx-commit = 2
innodb-flush-method            = O_DIRECT
innodb-log-files-in-group      = 2
key-buffer-size                = 32M

server_id                      = 1
log-bin                        = /data/mysql/mysql-bin
log-slave-updates              = on
master-info-repository         = TABLE
max-allowed-packet             = 16M
max-connect-errors             = 1000000
max-connections                = 500
max-heap-table-size            = 32M
myisam-recover-options         = FORCE,BACKUP
open-files-limit               = 65535
relay-log-info-repository      = TABLE
relay-log-recovery             = on
skip-slave-start               = on
sync-binlog                    = 1
sysdate-is-now                 = 1
table-definition-cache         = 4096
table-open-cache               = 4096
thread-cache-size              = 50
tmp-table-size                 = 32M
binlog-space-limit             = 50G
innodb-buffer-pool-size        = 384M
innodb-log-file-size           = 128M
max-binlog-size                = 1G
socket                         = /data/mysql/mysql.sock

[client]
port=3306
```

```
socket=/data/mysql/mysql.sock
```

- mysql-slave /etc/my.cnf配置

```
[mysqld]
expire-logs-days               = 14
query-cache-size               = 0
query-cache-type               = 0

sql-mode                       =
STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_AUTO_VALUE_ON_ZERO,NO_ENGINE_SUBSTITUTION,NO_ZERO_DATE,NO_ZERO_IN_DATE,ONL
Y_FULL_GROUP_BY

skip-host-cache
skip-name-resolve
binlog-format                  = ROW
character-set-server           = utf8mb4
collation-server               = utf8mb4_unicode_ci
default-storage-engine         = InnoDB
enforce-gtid-consistency       = on
gtid-mode                      = on
innodb-file-per-table          = 1
innodb-flush-log-at-trx-commit = 2
innodb-flush-method            = O_DIRECT
innodb-log-files-in-group      = 2
key-buffer-size                = 32M

server_id                      = 2
log-bin                        = /data/mysql/mysql-bin
log-slave-updates              = on
master-info-repository         = TABLE
max-allowed-packet             = 16M
max-connect-errors             = 1000000
max-connections                = 500
max-heap-table-size            = 32M
myisam-recover-options         = FORCE,BACKUP
open-files-limit               = 65535
relay-log-info-repository      = TABLE
relay-log-recovery             = on
skip-slave-start               = on
sync-binlog                    = 1
sysdate-is-now                 = 1
table-definition-cache         = 4096
table-open-cache               = 4096
thread-cache-size              = 50
tmp-table-size                 = 32M
binlog-space-limit             = 50G
innodb-buffer-pool-size        = 384M
innodb-log-file-size           = 128M
max-binlog-size                = 1G
socket                         = /data/mysql/mysql.sock

[client]
port=3306
socket=/data/mysql/mysql.sock
```

3.3.6 初始化数据库

```
su - mysql
cd /usr/local/mysql/bin
./mysqld --initialize --user=mysql  --basedir=/usr/local/mysql --datadir=/data/mysql
```

> 注意：需要记录初始化打印最后面的密码

3.3.7 登录mysql并进行配置

```
ln -s /usr/local/mysql/support-files/mysql.server  /etc/init.d/mysqld

ln -s  /usr/local/mysql/bin/mysql /usr/bin/mysql

#添加开机启动
chmod +x /etc/init.d/mysqld

#添加服务
chkconfig --add mysqld

#启动mysql
service mysqld start
```

```
#登录mysql
mysql -uroot -p(系统初始化密码)

#修改root密码：
mysql>alter user 'root'@'localhost' identified by 'xxxxx';('你的密码');
mysql>flush privileges;

#msyql-master创建主从复制账号：
mysql> grant replication slave on *.* to repluser(主从复制账号)@'172.16.10.11' identified by  'xxxxx'(主从复制密码);
mysql>flush privileges;

#查看Mysql-master日志状态，并记录Position值：
mysql>show master status;

#mysql-slave上执行：
mysql>CHANGE MASTER TO
MASTER_HOST='172.16.10.10',
MASTER_USER='repluser',
MASTER_PASSWORD='xxxxx',
MASTER_PORT=3306,
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=120;
mysql> start slave;

检查主从状态：
mysql>show slave status\G

如果Slava_IO_running:Yes
    Slave_SQL_Running:Yes
表示主从同步配置成功
```

3.4 mysql数据备份导入

3.4.1 备份客户原数据库的所有数据

```
mysqldump -uroot -p --all-databases > /backup/mysqldump/all.db
```

3.4.2 将备份数据导入mysql主从上

```
mysql>source /backup/mysqldump/all.db
```

3.4.3 验证数据是否正确

3.5 对主从数据库进行全量备份

3.5.1 脚本如下：

```
#!/bin/bash
# mysql备份脚本，每天备份一次，删除4天前备份(根据客户需求)
#backdir
backupDir=/data/mysql/dback
#mysqlDump
mysqldump=/usr/local/mysql/bin/mysqldump
#ip address
host=127.0.0.1
#username && password
username=root
password=xxxxx

#今天日期
today=`date +%Y%m%d`
#3天前的日期
timeFourDaysAgo=`date -d -4day +%Y%m%d`

echo '开始备份mysql'
$mysqldump -h$host --port 3306 -u$username -p$password --all-databases | gzip > $backupDir/mysqlback-$today.sql.gz
echo '成功备份'$database'到'$backupDir/mysqlback-$today.sql.gz

if [ ! -f "$backupDir/mysqlback-$timeFourDaysAgo.sql.gz" ]; then
          echo '4天前备份不存在，无需删除'
      else
          rm $backupDir/mysqlback-$timeFourDaysAgo.sql.gz
          echo '删除4天前的备份文件'$backupDir/$database-$timeFourDaysAgo.sql.gz
fi
```

3.5.2 设置定时任务(设置每周天1:30进行备份)

```
crontab -e

30 1 1 * *  /home/mysql/mysql_back.sh
```

3.6 对mysql所在主机进行监控

3.6.1 用我们自建的prometheuse使用node_exporter进行统一监控。

3.6.2 prometheus node-exporter

> 下载地址：https://github.com/prometheus/node_exporter/releases/download/v1.4.0/node_exporter-1.4.0.linux-amd64.tar.gz

> node_exporter：用于监控Linux系统的指标采集器。常用的指标如下：

- CPU
- 内存
- 硬盘
- 网络流量
- 系统负载
- 数据接口：http://IP:9100/metrics
- 使用文档：https://prometheus.io/docs/guides/node-exporter/
- 部署步骤如下（maser和slave都要部署）：

```
# 下载解压
wget  https://github.com/prometheus/node_exporter/releases/download/v1.4.0/node_exporter-1.4.0.linux-amd64.tar.gz
tar -zvxf node_exporter-1.4.0.linux-amd64.tar.gz
cd node_exporter-1.4.0.linux-amd64/

#设置为系统服务
vi /etc/systemd/system/node_exporter.service

Unit]
Description=node-exporter

[Service]
Type=simple
Restart=on-failure
RestartSec=5
ExecStart=/usr/local/node_exporter/node_exporter

[Install]
WantedBy=multi-user.target

#启动
 systemctl start node_exporter
 systemctl status node_exporter
 systemctl enable node_exporter

 #配置prometheus,在prometheus.yml上新增监控节点:
   - job_name: 'my-home-linux'
    static_configs:
      - targets:
          - '172.16.10.10:9100','172.16.10.11:9100'
```

3.7 对mysql主从业务进行监控

3.7.1 用我们自建的prometheuse使用mysql_exporter进行统一监控。

3.6.2 prometheus mysql_exporter

> 下载地址：https://github.com/prometheus/mysqld_exporter/releases/download/v0.14.0/mysqld_exporter-0.14.0.linux-amd64.tar.gz

> mysql_exporter：用于监控mysql的指标采集器。常用的指标如下

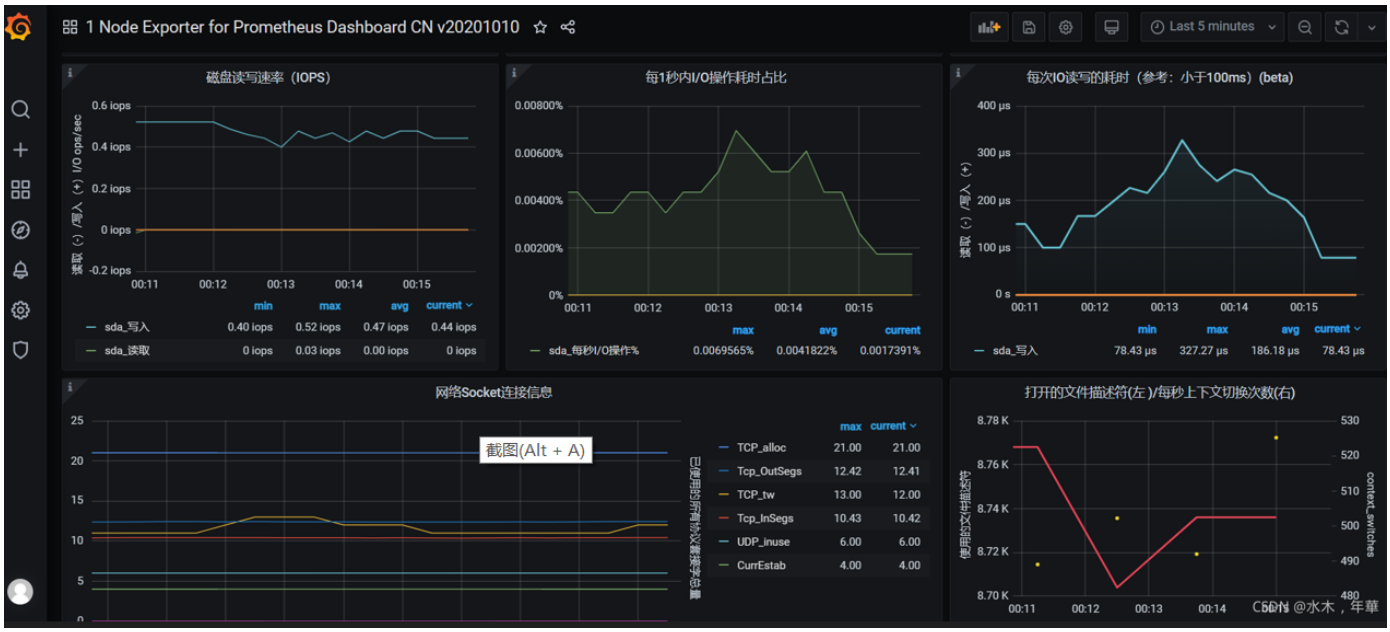
- Mysql监控agent存活
- Mysql监控连接数

- Mysql监控主从延迟
- Mysql监控SQL线程
- Mysql监控IO线程
- MySQL可用连接数
- 部署步骤如下（maser和slave都要部署）：

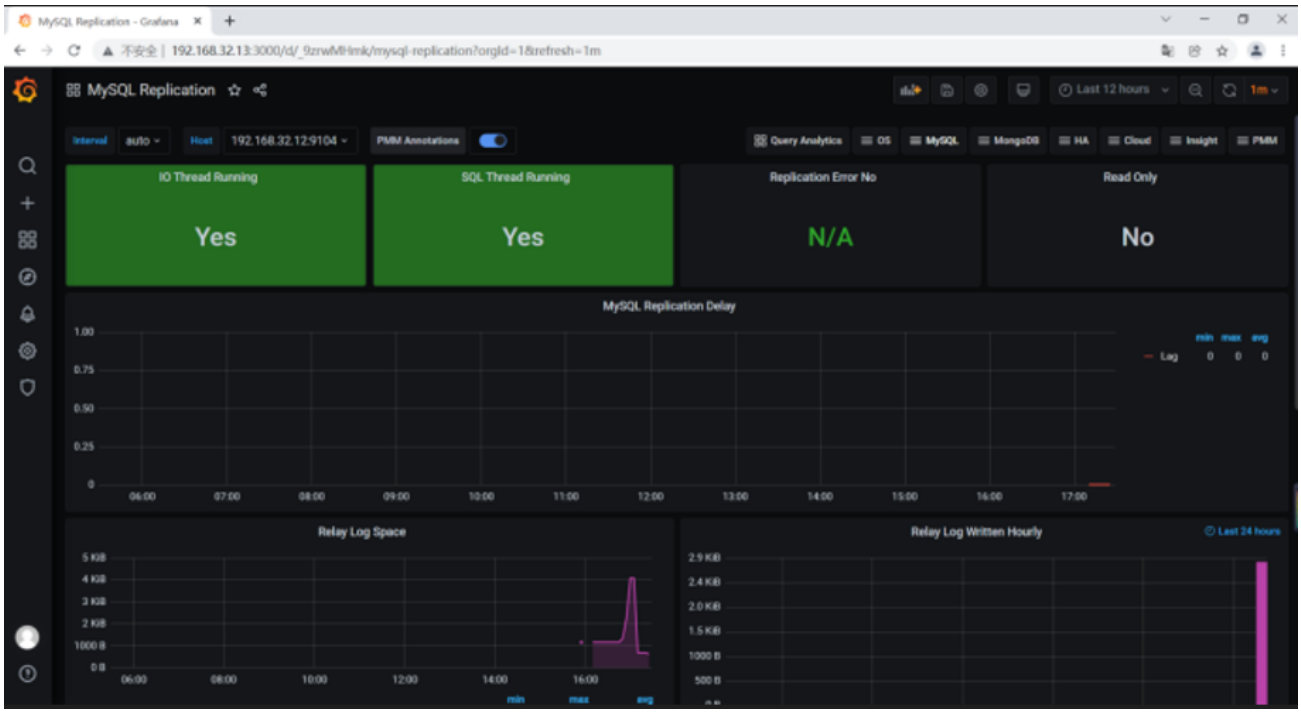
```
#下载并解压
wget https://github.com/prometheus/mysqld_exporter/releases/download/v0.14.0/mysqld_exporter-0.14.0.linux-amd64.tar.gz
tar -zxvf mysqld_exporter-0.14.0.linux-amd64.tar.gz -C /usr/local/

#mysql添加授权账户给exporter使用
mysql> create user 'exporter'@'127.0.0.1' identified by 'xxxxx'(密码);
mysql> grant process,replication client,select on *.* to 'exporter'@'127.0.0.1';
mysql> flush privileges;

#在mysqld_exporter路径下创建my.cnf，添加刚才创建的exporter用户和密码
[root@master mysqld_exporter-0.14.0.linux-amd64]# pwd
/usr/local/mysqld_exporter-0.14.0.linux-amd64
[root@master mysqld_exporter-0.14.0.linux-amd64]# cat my.cnf  //文件需创建
[client]
user=exporter
password=xxxxx

#添加system系统服务
 vim /usr/lib/systemd/system/mysqld_exporter.service

 [Unit]
Description=mysqld_exporter
After=network.target

[Service]
User=root
Type=simple
ExecStart=/usr/local/mysqld_exporter-0.14.0.linux-amd64/mysqld_exporter \
--config.my-cnf /usr/local/mysqld_exporter-0.14.0.linux-amd64/my.cnf \
--collect.info_schema.processlist

Restart=on-failure

[Install]
WantedBy=multi-user.target

#启动服务
systemctl daemon-reload
systemctl start mysqld_exporter.service
netstat -lntup | grep "9104" #查看服务是否启动

#配置prometheus,在prometheus.yml上新增监控节点:
- job_name: 'mysql-mater-slave'
  scrape_interval: 5s
  static_configs:
  - targets: ['172.16.10.10:9104','172.16.10.11:9104']
```

3.8 使用grafana添加相对应的模板来展示以上监控项

3.8.1 mysql node_exporter监控展示：



3.8.2 mysql_exporter监控展示



3.9使用prometheus altermanager组件实现短信告警

3.9.1通过修改prometheus-alert-sms.yaml文件，可以将一些紧急告警如上面所监控的主机down、mysql down、mysql主从复制失败等通过短信及时告知。

四、客户系统访问mysql地址的切换

4.1 部署完成并验证无问题后，可以重新导入最新的mysql备份数据，将mysql访问地址修改为mysql-mater的地址，并验证结果。