生产环境中，为了尽可能地让u8server服务高可用，除了u8server本身对分布式部署的支持。数据库层面，也需要满足稳定和高可用性。本文中，我们使用Mycat这一开源数据库中间件，实现基本的主从读写分离，主备自动切换。

同时对u8server中核心业务数据用户表和订单表进行了数据的水平切分，实现分库分表。

采用数据库中间件，最大限度地避免了业务层代码的修改。当数据量达到一定规模之后，数据库层面的运维会变得越来越关键，也越来越繁重。可能需要专门的运维团队和DBA来负责这些基础设施和日常运维工作。

下面记录了本次实时测试的完整步骤：

多mysql实例安装

1、拷贝mysql目录

2、拷贝my\_default.ini-》重命名为my.ini

3、修改my.ini配置，在[mysqld]下面添加：

basedir = I:/Program Files (x86)/MySQL Server 5.7-3307datadir = I:/Program Files (x86)/MySQL Server 5.7-3307/dataport = 3307character-set-server=utf8

4、添加mysql3307系统服务，执行命令：

I:\Program Files (x86)\MySQL Server 5.7-3307\bin>mysqld install mysql3307 --defaults-file="I:\Program Files (x86)\MySQL Server 5.7-3307\my.ini"

5、初始化数据库，执行命令：

I:\Program Files (x86)\MySQL Server 5.7-3307\bin>mysqld --defaults-file="I:\Program Files (x86)\MySQL Server 5.7-3307\my.ini" --initialize --explicit\_defaults\_for\_timestamp

6、启动服务

net start mysql3307

7、修改初始密码，在data目录下，找到[user]-err.txt，打开，在里面搜索password关键字：

找到初始密码

8、修改初始密码，执行命令：

I:\Program Files (x86)\MySQL Server 5.7-3307\bin>mysql -u root -p -hlocalhost -P3307

输入初始密码，进入mysql

继续执行命令：

set password = password('your\_password');

然后exit 退出

按照上面步骤，安装一台端口3309的mysql

配置mysql主从

3307为主，3309为从

1、配置3307主服务器：

1.1关闭 3307 服务： net stop mysql3307

1.2修改配置,my.ini里面 [mysqld]下面增加两行配置：

log-bin=mysql-binserver\_id = 1

1.3重启mysql服务：net start mysql3307

1.4 连上mysql3307，创建用户slave同步数据使用的用户和密码：

mysql> create user repl;mysql> grant replication slave on \*.\* to 'repl'@'localhost' identified by 'slave@123';mysql> show master status;+------------------+----------+--------------+------------------+------------------+| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB | Executed\_GtidSet |+------------------+----------+--------------+------------------+------------------+| mysql-bin.000001 | 643 | | | |+------------------+----------+--------------+------------------+------------------+1 row in set (0.00 sec)

2、配置3309从服务器：

2.1 关闭 3309 服务： net stop mysql3309

2.2 修改配置，my.ini里面[mysqld]下面增加配置：

server\_id = 2

2.3 重启mysql服务： net start mysql3309

2.4 连上mysql3309， 修改指向的master：

mysql> change master to master\_host='localhost',master\_port=3307,master\_user='repl',master\_password='slave@123',master\_log\_file='mysql-bin.000001',master\_log\_pos=643;

启动slave:

mysql> start slave;

查看slave 状态：

mysql>show slave status;

配置mycat

我们使用mycat数据库中间件来实现读写分离，主从切换，以及分库分表。mycat如何使用，建议先阅读《Mycat权威指南》，文档最后有链接，本文不再赘述。

3.1、配置逻辑库和表信息，以及配置读写分离和自动主从切换

u8server中， 游戏，渠道商，渠道，管理员，权限等数据都属于“字典型”数据。这类数据查询平凡，和其他数据耦合度相对较高，同时又很少发生变动，单表几乎不可能有超过百万数据量的规模。

对于这部分数据表，我们将其设置为mycat中的全局表。

schema.xml:

show slave status

3.2、配置mycat 用户权限

server.xml中添加user配置：

xiaohei udb\_mycat

3.3、设置mycat全局序列号方式为 数据库方式：

分库分表之后，原有的数据库表主键id自增长方式可能就不能满足需求了，这里Mycat提供了多种方式来实现全局唯一序列号。我们这里采用单独数据库表的方式，来生成全局唯一序列号。

server.xml中搜索sequnceHandlerType， 然后将其值设置为1， 表示，全局序列号采用数据库方式。

3.4、创建全局序列号数据表和对应函数。 我们在dn1上面建立。

DROP TABLE IF EXISTS `mycat\_sequence`;CREATE TABLE `mycat\_sequence` ( `name` varchar(50) NOT NULL, `current\_value` int(11) NOT NULL, `increment` int(11) NOT NULL DEFAULT '100', PRIMARY KEY (`name`)) ENGINE=InnoDB DEFAULT CHARSET=utf8;DROP FUNCTION IF EXISTS `mycat\_seq\_currval`;DELIMITER ;;CREATE DEFINER=`root`@`localhost` FUNCTION `mycat\_seq\_currval`(seq\_name VARCHAR(50)) RETURNS varchar(64) CHARSET utf8DETERMINISTICBEGINDECLARE retval VARCHAR(64);SET retval="-999999999,null";SELECT concat(CAST(current\_value AS CHAR),",",CAST(increment AS CHAR) ) INTO retval FROM MYCAT\_SEQUENCE WHERE name = seq\_name;RETURN retval ;END;;DELIMITER ;-- 设置sequence值DROP FUNCTION IF EXISTS `mycat\_seq\_nextval`;DELIMITER ;;CREATE DEFINER=`root`@`localhost` FUNCTION `mycat\_seq\_nextval`(seq\_name VARCHAR(50)) RETURNS varchar(64) CHARSET utf8DETERMINISTICBEGINUPDATE MYCAT\_SEQUENCESET current\_value = current\_value + increment WHERE name = seq\_name;RETURN mycat\_seq\_currval(seq\_name);END;;DELIMITER ;-- 获取下一个sequence值DROP FUNCTION IF EXISTS `mycat\_seq\_setval`;DELIMITER ;;CREATE DEFINER=`root`@`localhost` FUNCTION `mycat\_seq\_setval`(seq\_name VARCHAR(50), value INTEGER) RETURNS varchar(64) CHARSET utf8DETERMINISTICBEGINUPDATE MYCAT\_SEQUENCESET current\_value = valueWHERE name = seq\_name;RETURN mycat\_seq\_currval(seq\_name);END;;DELIMITER ;INSERT INTO `mycat\_sequence` VALUES ('GLOBAL', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('UCHANNEL', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('UCHANNELMASTER', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('UGAME', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('USUBCHANNEL', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('USYSMENU', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('UADMIN', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('UADMINROLE', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('UUSER', '100000', '100');INSERT INTO `mycat\_sequence` VALUES ('UDEVICE', '100000', '100');

3.5、指定全局序列号数据所在节点

sequence\_db\_conf.properties文件中，增加如下配置：

#sequence stored in datanodeGLOBAL=dn1UCHANNEL=dn1UCHANNELMASTER=dn1UGAME=dn1USUBCHANNEL=dn1USYSMENU=dn1UADMIN=dn1UADMINROLE=dn1UUSER=dn1UDEVICE=dn1

3.6、重启mycat

Mycat分库分表配置

在U8Server中，数据量最可能超出单机极限的是uuser用户表和uorder订单表。这个时候，就需要对这两张表做分库分表，对数据进行水平切分。分库分表之后，最可能引入的问题是，跨表Join查询和分布式事务问题。

好在u8server中，我们几乎没有用到join查询，事务处理也基本很简单，基本处于单机事务中，并不会涉及到复杂的分布式事务。 所以看，对于这两张表的切分， 可以正常进行，而不用调整业务逻辑部分。

分库分表，就涉及到另一个严肃的问题：分库分表规则。规则有很多，但是每种都有其优势和弊端。常用的规则有取余算法，范围算法，hash算法等。但是为了将数据能够尽可能均匀地划分到各个存储节点中，我们一般采用取余算法(mod)。

选定分库分表规则之后，另一个需要考虑的问题，是扩容减容问题。比如一开始我们有两个存储节点，按照id mod 2进行分库分表。

当数据量进一步提升，我们需要扩容一个新的存储节点。这个时候，我们按照id mod 3规则进行分库分表之后，需要对原有数据按照新的规则，进行迁移。当数据节点过多时，数据迁移可能会花费大量时间。

针对这个难题，一致性hash算法规则能最大限度地减少扩容之后需要迁移的数据量，同时又能够尽可能让数据均匀地分布在各个存储节点。mycat中已经有对一致性hash算法的支持。

这里，我们演示下，如何对uuser表和uorder表进行分库分表切分。 我们这里采用采用两个存储节点，使用uuser的id mod 2 规则进行数据的切分。同时，基于Mycat的ER表特性，我们将uorder分库分表规则，依赖uuser。也就是，让同一个用户的所有订单数据，和uuser记录划分到同一个存储节点。

以下是实施步骤：

4.1、新增两个mysql实例

按照一中所说步骤， 我们增加两个mysql实例， 端口号分别为 3311和 3313。3311和3313设定主从关系。

4.2、修改mycat 配置

增加一个dataHost——localhost2，然后增加一个dataNode——dn2，并将dn2的存储节点指向localhost2。 然后指定uuser表的dataNode为dn1,dn2。rule（分库分表规则）指定为mod-int-two。uorder表通过childTable特性，配置到uuser中，作为uuser的子表。

修改之后，完整的schema.xml配置如下：

show slave status show slave status

测试

将U8Server和U8ServerManager中jdbc.properties中，数据库ip和端口，改为mycat所在服务器的ip，mycat默认端口为8066。然后重启U8Server和U8ServerManager实例。

1、测试读写分离。

读写分离测试时，还没有进行分库分表配置。

从mycat日志中，查看数据读取和插入路由日志，确认数据从从库中读取，向主库中写入，写入的数据可以成功同步到从库中。

我们插入一条用户记录，然后读取该用户记录。通过mycat路由日志，可以看到数据从hostS1读取。插入时，数据会插入到hostM1。

2、测试全局序列号。

后台管理中，插入一条新的渠道数据。看到插入的数据中，主键ID已经变成了全局序列号生成的id值。

3、测试分库分表逻辑。

插入两条uuser记录， 查看hostM1和hostM3中，uuser表中的记录， 发现hostM1和hostM3中，uuser表各一条数据。

插入四条uorder记录， 其中三条的userID和hostM3中的uuser记录对应，查看其是否也被写入到hostM3中。其中一条的userID和hostM1中的uuser记录对应，查看其是否会被写入到hostM1中。

遇到的问题

1、代码中，之前企业版分离U8SeverManager时，对于uchannelmaster和ugame两个实体类中，没有定义自增长，而是采用启动时，加载全局数据，取出最大ID，然后内存中每次递增ID。这个属于代码历史遗留问题。这次一并将其改为主键自增长方式，不再采用老的方式。

2、数据库中，uchannelmaster和ugame两张表的主键，加上自动递增选项。

总结

数据量大到一定规模，引入分库分表之后，有一些问题需要解决或者注意。

1、数据统计需要重构。

之前数据统计，是采用数据库中定时任务来完成的。 定时任务每天凌晨，会从日志表、用户表、订单表中分析原始数据，然后将分析得到的统计数据插入到统计表中。用户表和订单表分库分表之后， 原有逻辑已经不能满足现在的部署架构。

同时，数据量达到一定规模之后， 单台数据库中执行的速度和效率都会大打折扣，还会影响到数据库本身的处理性能。

这里，未来我们计划将数据统计这块，全部走日志，后端输出到大数据计算平台来完成数据的异步计算和统计分析。结合大数据计算平台的分布式计算能力，来尽可能快速地完成统计分析任务。

当前阶段，如果你的数据到了必须分库分表的阶段，那么请关闭u8server的数据统计功能。客户端打包工具中，将config/local/local.properties中u8\_analytics设置为false即可。

2、扩容问题。

分库分表之后， 不可避免地会遇到扩容问题。 扩容目前，需要提前规划，还需要写数据迁移脚本来完成数据的迁移。分库分表算法，建议使用一致性hash算法，来尽可能地缩短扩容带来的数据迁移时间。

3、mycat本身的高可用性。

Mycat本身几乎是无状态的，所以可以参考Mycat的高可用性部署方案。具体部署方案，可以参考Mycat权威指南。

相关资料

[Mycat官方网站](http://www.mycat.org.cn/)

[一致性hash算法](http://www.cnblogs.com/rainwang/p/4309102.html)