**后台服务nosql数据库mongodb：高可用以及安装**

#### 一、架构介绍

### mongodb有几种部署方式，这里采用的是副本集架构（Replica Set）。

### 为了防止单点故障就需要引副本（Replication），当发生硬件故障或者其它原因造成的宕机时，可以使用副本进行恢复，最好能够自动的故障转移（failover）。有时引入副本是为了读写分离，将读的请求分流到副本上，减轻主（Primary）的读压力。而Mongodb的Replica Set都能满足这些要求。

### Replica Set的一堆mongod的实例集合，它们有着同样的数据内容。包含三类角色：

### 主节点（Primary）：接收所有的写请求，然后把修改同步到所有Secondary。一个Replica Set只能有一个Primary节点，当Primary挂掉后，其他Secondary或者Arbiter节点会重新选举出来一个主节点。默认读请求也是发到Primary节点处理的，需要转发到Secondary需要客户端修改一下连接配置。

### 副本节点（Secondary）：与主节点保持同样的数据集。当主节点挂掉的时候，参与选主。

### 仲裁者（Arbiter）：不保有数据，不参与选主，只进行选主投票。使用Arbiter可以减轻数据存储的硬件需求，Arbiter跑起来几乎没什么大的硬件资源需求，但重要的一点是，在生产环境下它和其他数据节点不要部署在同一台机器上。

### 注意，一个自动failover的Replica Set节点数必须为奇数，目的是选主投票的时候要有一个大多数才能进行选主决策。

### 

### 由图可以看到客户端连接到整个副本集，不关心具体哪一台机器是否挂掉。主服务器负责整个副本集的读写，副本集定期同步数据备份，一但主节点挂掉，副本节点就会选举一个新的主服务器，这一切对于应用服务器不需要关心。我们看一下主服务器挂掉后的架构：

### 

副本集中的副本节点在主节点挂掉后通过心跳机制检测到后，就会在集群内发起主节点的选举机制，自动选举一位新的主服务器。

#### 二、安装部署

选择三台服务器：192.168.40.172（主节点） 192.168.40.167（副本节点） 192.168.40.158（副本节点）

下载mongodb：wget <https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-3.4.7.tgz>

解压到：/data/program/software

文件夹重命名为mongodb

进入mongodb目录：cd /data/program/software/mongodb

新建两个文件夹：mkdir db

mkdir logs

进入bin目录cd /data/program/software/mongodb/bin

新建配置文件：touch mongodb.conf

dbpath=/data/program/software/mongodb/db

logpath=/data/program/software/mongodb/logs/mongodb.log

port=27017

fork=true

nohttpinterface=true

分别三台服务器上启动mongodb：

/data/program/software/mongodb/bin/mongod --replSet repset -f /data/program/software/mongodb/bin/mongodb.conf

/java/mongodb-linux-x86\_64-3.4.7/bin/mongod --replSet repset -f /java/mongodb-linux-x86\_64-3.4.7/bin/mongodb.conf

各个服务器查看，都已经启动：

[root@bigdata3 bin]# ps -ef|grep mongodb

root 2574 1 1 22:09 ? 00:00:00 /data/program/software/mongodb/bin/mongod --replSet repset -f /data/program/software/mongodb/bin/mongodb.conf

在三台机器上任意一台机器登陆mongodb：

/data/program/software/mongodb/bin/mongo

使用admin数据库

use admin

定义副本集配置变量，这里的\_id:”repset”和上面命令参数--replSet repset保持一致

|  |
| --- |
| config = { \_id:"repset", members:[{\_id:0,host:"192.168.40.172:27017"},{\_id:1,host:"192.168.40.167:27017"},{\_id:2,host:"192.168.40.158888:27017"}]}  初始化副本集群：rs.initiate(config);  表示成功  查看集群节点的状态：  rs.status();  如下结果：  {  "set" : "repset",  "date" : ISODate("2017-09-21T14:30:17.190Z"),  "myState" : 1,  "term" : NumberLong(1),  "heartbeatIntervalMillis" : NumberLong(2000),  "optimes" : {  "lastCommittedOpTime" : {  "ts" : Timestamp(1506004207, 1),  "t" : NumberLong(1)  },  "appliedOpTime" : {  "ts" : Timestamp(1506004207, 1),  "t" : NumberLong(1)  },  "durableOpTime" : {  "ts" : Timestamp(1506004207, 1),  "t" : NumberLong(1)  }  },  "members" : [  {  "\_id" : 0,  "name" : "10.211.55.7:27017",  "health" : 1,  "state" : 2,  "stateStr" : "SECONDARY",  "uptime" : 111,  "optime" : {  "ts" : Timestamp(1506004207, 1),  "t" : NumberLong(1)  },  "optimeDurable" : {  "ts" : Timestamp(1506004207, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2017-09-21T14:30:07Z"),  "optimeDurableDate" : ISODate("2017-09-21T14:30:07Z"),  "lastHeartbeat" : ISODate("2017-09-21T14:30:16.070Z"),  "lastHeartbeatRecv" : ISODate("2017-09-21T14:30:16.887Z"),  "pingMs" : NumberLong(0),  "syncingTo" : "10.211.55.9:27017",  "configVersion" : 1  },  {  "\_id" : 1,  "name" : "10.211.55.8:27017",  "health" : 1,  "state" : 2,  "stateStr" : "SECONDARY",  "uptime" : 111,  "optime" : {  "ts" : Timestamp(1506004207, 1),  "t" : NumberLong(1)  },  "optimeDurable" : {  "ts" : Timestamp(1506004207, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2017-09-21T14:30:07Z"),  "optimeDurableDate" : ISODate("2017-09-21T14:30:07Z"),  "lastHeartbeat" : ISODate("2017-09-21T14:30:16.070Z"),  "lastHeartbeatRecv" : ISODate("2017-09-21T14:30:16.877Z"),  "pingMs" : NumberLong(0),  "syncingTo" : "10.211.55.9:27017",  "configVersion" : 1  },  {  "\_id" : 2,  "name" : "10.211.55.9:27017",  "health" : 1,  "state" : 1,  "stateStr" : "PRIMARY",  "uptime" : 163,  "optime" : {  "ts" : Timestamp(1506004207, 1),  "t" : NumberLong(1)  },  "optimeDate" : ISODate("2017-09-21T14:30:07Z"),  "infoMessage" : "could not find member to sync from",  "electionTime" : Timestamp(1506004115, 1),  "electionDate" : ISODate("2017-09-21T14:28:35Z"),  "configVersion" : 1,  "self" : true  }  ],  "ok" : 1  } |

#### 三、测试集群功能

主节点连接到终端：

mongo 127.0.0.1

连接test数据库：

use test；

往testdb表里插入数据

db.testdb.insert({"test1":"testval1"})

在副本节点连接查询：

./mongo 10.211.55.7:27107

使用test数据库：user test;

查询表格：show tables;

报错如下：



mongodb默认是从主节点读写数据，副本节点上不允许读，设置副本节点可读。

db.getMongo().setSlaveOk();

然后就可以查询复制过来的数据了

repset:SECONDARY> db.testdb.find();

{ "\_id" : ObjectId("59c3cf8bf21850357d5392ba"), "test1" : "testval1" }

测试集群恢复功能，去停掉主节点：

然后查看节点状态，发现7和8中有一台变为了PRIMARY，然后再启动主节点观察状态。

https://www.cnblogs.com/clsn/p/8214345.html