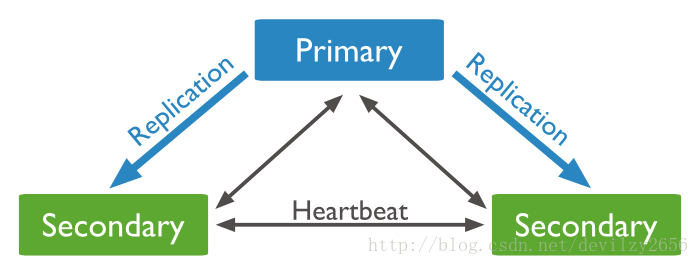
**副本集**：（主从的替代方式）

mongoDB官方已经不建议使用主从模式了，替代方案是采用副本集的模式，如图：



**1、准备两台机器** 192.168.1.136、192.168.1.137、192.168.1.138。 192.168.1.136 当作副本集**主节点**，192.168.1.137、192.168.1.138作为副本集**副本节点**。

**2、分别在每台机器上建立mongodb副本集测试文件夹**

#存放整个mongodb文件

mkdir -p /data/mongodbtest/replset

#存放mongodb数据文件

mkdir -p /data/mongodbtest/replset/data

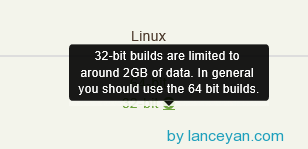
#进入mongodb文件夹

cd /data/mongodbtest

**3、下载mongodb的安装程序包**

wget <a href="http://fastdl.mongodb.org/linux/mongodb-linux-x86\_64-2.4.8.tgz">http://fastdl.mongodb.org/linux/mongodb-linux-x86\_64-2.4.8.tgz</a>

注意linux生产环境不能安装32位的mongodb，因为32位受限于操作系统最大2G的文件限制。

[](http://www.lanceyan.com/wp-content/uploads/2013/12/mongorep4.png)

#解压下载的压缩包

tar xvzf mongodb-linux-x86\_64-2.4.8.tgz

**4、分别在每台机器上启动mongodb**

/data/mongodbtest/mongodb-linux-x86\_64-2.4.8/bin/mongod --dbpath /data/mongodbtest/replset/data --replSet repset

可以看到控制台上显示副本集还没有配置初始化信息。

**[plain]** [view plaincopy](http://www.csdn.net/article/2014-04-09/2819221-build-high-avialable-mongodb-cluster-part-1/2)

1. Sun Dec 29 20:12:02.953 [rsStart] replSet can't get local.system.replset config from self or any seed (EMPTYCONFIG)
2. Sun Dec 29 20:12:02.953 [rsStart] replSet info you may need to run  replSetInitiate -- rs.initiate() in the shell -- if that is not already done

**5、初始化副本集**

在三台机器上任意一台机器登陆mongodb

/data/mongodbtest/mongodb-linux-x86\_64-2.4.8/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” 要保持一样。

config = { \_id:"repset", members:[

... {\_id:0,host:"192.168.1.136:27017"},

... {\_id:1,host:"192.168.1.137:27017"},

... {\_id:2,host:"192.168.1.138:27017"}]

... }

#输出

**[plain]** [view plaincopy](http://www.csdn.net/article/2014-04-09/2819221-build-high-avialable-mongodb-cluster-part-1/2)

[在CODE上查看代码片](https://code.csdn.net/snippets/415579)

1. {
2. "\_id" : "repset",
3. "members" : [
4. {
5. "\_id" : 0,
6. "host" : "192.168.1.136:27017"
7. },
8. {
9. "\_id" : 1,
10. "host" : "192.168.1.137:27017"
11. },
12. {
13. "\_id" : 2,
14. "host" : "192.168.1.138:27017"
15. }
16. ]
17. }

#初始化副本集配置

rs.initiate(config);

#输出成功

**[plain]** [view plaincopy](http://www.csdn.net/article/2014-04-09/2819221-build-high-avialable-mongodb-cluster-part-1/2)

1. {
2. "info" : "Config now saved locally.  Should come online in about a minute.",
3. "ok" : 1
4. }

#查看日志，副本集启动成功后，138为主节点**PRIMARY**，136、137为副本节点**SECONDARY**。

#查看集群节点的状态 rs.status();

#输出

**[plain]** [view plaincopy](http://www.csdn.net/article/2014-04-09/2819221-build-high-avialable-mongodb-cluster-part-1/2)

1. {
2. "set" : "repset",
3. "date" : ISODate("2013-12-29T12:54:25Z"),
4. "myState" : 1,
5. "members" : [
6. {
7. "\_id" : 0,
8. "name" : "192.168.1.136:27017",
9. "health" : 1,
10. "state" : 2,
11. "stateStr" : "SECONDARY",
12. "uptime" : 1682,
13. "optime" : Timestamp(1388319973, 1),
14. "optimeDate" : ISODate("2013-12-29T12:26:13Z"),
15. "lastHeartbeat" : ISODate("2013-12-29T12:54:25Z"),
16. "lastHeartbeatRecv" : ISODate("2013-12-29T12:54:24Z"),
17. "pingMs" : 1,
18. "syncingTo" : "192.168.1.138:27017"
19. },
20. {
21. "\_id" : 1,
22. "name" : "192.168.1.137:27017",
23. "health" : 1,
24. "state" : 2,
25. "stateStr" : "SECONDARY",
26. "uptime" : 1682,
27. "optime" : Timestamp(1388319973, 1),
28. "optimeDate" : ISODate("2013-12-29T12:26:13Z"),
29. "lastHeartbeat" : ISODate("2013-12-29T12:54:25Z"),
30. "lastHeartbeatRecv" : ISODate("2013-12-29T12:54:24Z"),
31. "pingMs" : 1,
32. "syncingTo" : "192.168.1.138:27017"
33. },
34. {
35. "\_id" : 2,
36. "name" : "192.168.1.138:27017",
37. "health" : 1,
38. "state" : 1,
39. "stateStr" : "PRIMARY",
40. "uptime" : 2543,
41. "optime" : Timestamp(1388319973, 1),
42. "optimeDate" : ISODate("2013-12-29T12:26:13Z"),
43. "self" : true
44. }
45. ],
46. "ok" : 1
47. }

整个副本集已经搭建成功了。

**6、测试副本集数据复制功能**

#在主节点192.168.1.138 上连接到终端：

mongo 127.0.0.1

#建立test 数据库。

use test;

往testdb表插入数据。

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

#在副本节点 192.168.1.136、192.168.1.137 上连接到mongodb查看数据是否复制过来。

/data/mongodbtest/mongodb-linux-x86\_64-2.4.8/bin/mongo 192.168.1.136:27017

#使用test 数据库。

repset:SECONDARY> use test;

repset:SECONDARY> show tables;

#输出

**[plain]** [view plaincopy](http://www.csdn.net/article/2014-04-09/2819221-build-high-avialable-mongodb-cluster-part-1/2)

[在CODE上查看代码片](https://code.csdn.net/snippets/415579)

1. Sun Dec 29 21:50:48.590 error: { "$err" : "not master and slaveOk=false", "code" : 13435 } at src/mongo/shell/query.js:128

#mongodb默认是从主节点读写数据的，副本节点上不允许读，需要设置副本节点可以读。

repset:SECONDARY> db.getMongo().setSlaveOk();

#可以看到数据已经复制到了副本集。

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

**[plain]** [view plaincopy](http://www.csdn.net/article/2014-04-09/2819221-build-high-avialable-mongodb-cluster-part-1/2)

1. #输出
2. { "\_id" : ObjectId("52c028460c7505626a93944f"), "test1" : "testval1" }

**7、测试副本集故障转移功能**

先停掉主节点mongodb 138，查看136、137的日志可以看到经过一系列的投票选择操作，137 当选主节点，136从137同步数据过来。

查看整个集群的状态，可以看到138为状态不可达。

/data/mongodbtest/mongodb-linux-x86\_64-2.4.8/bin/mongo 192.168.1.136:27017

repset:SECONDARY> rs.status();

再启动原来的主节点 138，发现138 变为 SECONDARY，还是137 为主节点 PRIMARY。

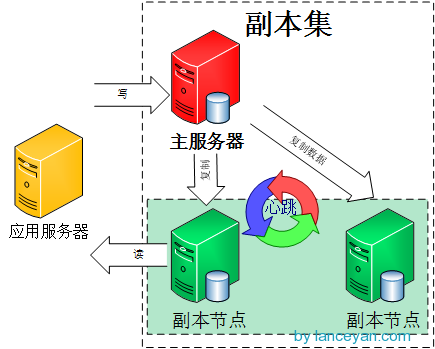
**8、java程序连接副本集测试。**三个节点有一个节点挂掉也不会影响应用程序客户端对整个副本集的读写！

**[java]** [view plaincopy](http://blog.csdn.net/devilzy2656/article/details/36629663)[在CODE上查看代码片](https://code.csdn.net/snippets/415579)

1. List<ServerAddress> addresses = new ArrayList<ServerAddress>();
2. ServerAddress address1 = new ServerAddress("192.168.0.180" , 27017);
3. ServerAddress address2 = new ServerAddress("192.168.0.181" , 27017);
4. ServerAddress address3 = new ServerAddress("192.168.0.182" , 27017);
5. addresses.add(address1);
6. addresses.add(address2);
7. addresses.add(address3);
8. Mongo mg = new MongoClient(addresses);
9. DB db = mg.getDB("qg\_cache");

目前看起来支持完美的故障转移了，这个架构是不是比较完美了？其实还有很多地方可以优化，比如开头的第二个问题：主节点的读写压力过大如何解决？常见的解决方案是读写分离，mongodb副本集的读写分离如何做呢？

看图说话：

[](http://www.lanceyan.com/wp-content/uploads/2013/12/mongorep5.png)

常规写操作来说并没有读操作多，所以一台主节点负责写，两台副本节点负责读。

1、设置读写分离需要先在副本节点SECONDARY 设置 setSlaveOk。   
2、在程序中设置副本节点负责读操作，如下代码：

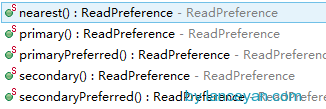
**[java]** [view plaincopy](http://www.csdn.net/article/2014-04-09/2819221-build-high-avialable-mongodb-cluster-part-1/2)

1. **public** **class** TestMongoDBReplSetReadSplit {
3. **public** **static** **void** main(String[] args) {
5. **try** {
6. List<ServerAddress> addresses = **new** ArrayList<ServerAddress>();
7. ServerAddress address1 = **new** ServerAddress("192.168.1.136" , 27017);
8. ServerAddress address2 = **new** ServerAddress("192.168.1.137" , 27017);
9. ServerAddress address3 = **new** ServerAddress("192.168.1.138" , 27017);
10. addresses.add(address1);
11. addresses.add(address2);
12. addresses.add(address3);
14. MongoClient client = **new** MongoClient(addresses);
15. DB db = client.getDB( "test" );
16. DBCollection coll = db.getCollection( "testdb" );

19. BasicDBObject object = **new** BasicDBObject();
20. object.append( "test2" , "testval2" );
22. //读操作从副本节点读取
23. ReadPreference preference = ReadPreference. secondary();
24. DBObject dbObject = coll.findOne(object, **null** , preference);
26. System. out .println(dbObject);

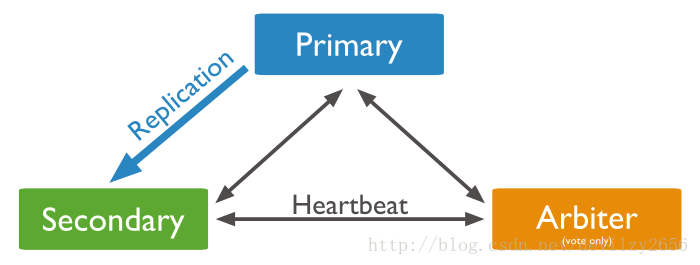
29. } **catch** (Exception e) {
30. e.printStackTrace();
31. }
32. }
33. }

读参数除了secondary一共还有五个参数：primary、primaryPreferred、secondary、secondaryPreferred、nearest。

[](http://www.lanceyan.com/wp-content/uploads/2013/12/mongorep6.png)

**primary:**默认参数，只从主节点上进行读取操作；   
**primaryPreferred:**大部分从主节点上读取数据,只有主节点不可用时从secondary节点读取数据。   
**secondary:**只从secondary节点上进行读取操作，存在的问题是secondary节点的数据会比primary节点数据“旧”。   
**secondaryPreferred:**优先从secondary节点进行读取操作，secondary节点不可用时从主节点读取数据；   
**nearest:**不管是主节点、secondary节点，从网络延迟最低的节点上读取数据。

好，读写分离做好我们可以数据分流，减轻压力解决了“主节点的读写压力过大如何解决？”这个问题。不过当我们的副本节点增多时，主节点的复制压力会加大有什么办法解决吗？mongodb早就有了相应的解决方案。

看图：   


其中的仲裁节点不存储数据，只是负责故障转移的群体投票，这样就少了数据复制的压力。是不是想得很周到啊，一看mongodb的开发兄弟熟知大数据架构体 系，其实不只是主节点、副本节点、仲裁节点，还有Secondary-Only、Hidden、Delayed、Non-Voting。

**Secondary-Only:**不能成为primary节点，只能作为secondary副本节点，防止一些性能不高的节点成为主节点。   
**Hidden:**这类节点是不能够被客户端制定IP引用，也不能被设置为主节点，但是可以投票，一般用于备份数据。   
**Delayed：**可以指定一个时间延迟从primary节点同步数据。主要用于备份数据，如果实时同步，误删除数据马上同步到从节点，恢复又恢复不了。   
**Non-Voting：**没有选举权的secondary节点，纯粹的备份数据节点。

到此整个mongodb副本集搞定了两个问题：

* 主节点挂了能否自动切换连接？目前需要手工切换。
* 主节点的读写压力过大如何解决？

还有这两个问题后续解决：

* 从节点每个上面的数据都是对数据库全量拷贝，从节点压力会不会过大？
* 数据压力大到机器支撑不了的时候能否做到自动扩展？

做了副本集发现又一些问题：

* 副本集故障转移，主节点是如何选举的？能否手动干涉下架某一台主节点。
* 官方说副本集数量最好是奇数，为什么？
* mongodb副本集是如何同步的？如果同步不及时会出现什么情况？会不会出现不一致性？
* mongodb的故障转移会不会无故自动发生？什么条件会触发？频繁触发可能会带来系统负载加重