

Seata-Seata的高可用异地容灾架构搭建

转载

mb5fe32930661bd 2021-10-14 16:51:00

文章标签 seata 异地容灾 seata 高可用 seata seata 搭建 seata 安装 文章分类 架构 后端开发

阅读数 471

1.模拟异地容灾的TC集群

计划启动两台seata的tc服务节点：

1. | 节点名称 | ip地址(具体IP) | 端口号 | 集群名称 |
2. | ----- | ----- | ----- | ----- |
3. | seata | 192.168.8.118 | 8091 | SH |
4. | seata2 | 192.168.8.118 | 8092 | HZ |

之前我们已经启动了一台seata服务，端口是8091，集群名为SH(注：192.168.8.118用tjia-nacos替代，在host配置)。

现在，将seata目录复制一份，起名为seata-server-2

修改seata-server-2/conf/registry.conf内容如下：

1. registry {
2. # tc服务的注册中心类，这里选择nacos，也可以是eureka、zookeeper等
3. # file、nacos、eureka、redis、zk、consul、etcd3、sofa
4. type = "nacos"
5.
6. nacos {
7. application = "seata-tc-server"
8. serverAddr = "tjia-nacos:8848"
9. group = "SEATA\_GROUP"
10. namespace = ""
11. cluster = "HZ"
12. username = "nacos"
13. password = "nacos"
14. }
15. }
16.
17. config {
18. # 读取tc服务端的配置文件的方式，这里是从nacos配置中心读取，这样如果tc是集群，可以共享配置
19. # file、nacos、apollo、zk、consul、etcd3
20. type = "nacos"
21. # 配置nacos地址等信息
22. nacos {
23. serverAddr = "tjia-nacos:8848"
24. namespace = ""
25. group = "SEATA\_GROUP"
26. username = "nacos"
27. password = "nacos"
28. dataId = "seataServer.properties"
29. }
30. }

进入seata2/bin目录，然后运行命令：

1. D:/dev/seata-server-2/bin/seata-server.bat -p 8092



mb5fe3293066

181 151.6万  
原创 人气

0 2407  
翻译 转载



+ 关注

—— 大咖讲师直播

博客用户专

课程亮点

- 企业级项目实
- 热门高薪技能
- IT证书考点解
- IT技术人职业

0元点击领取

近期文章

- 1.JS加密/解密之那些不为
- 2.跨语言的GRPC协议
- 3.西安王晓楠：大健康产品
- 4.理解Postgres的IOPS：为
- 5.默认为false导致实例创建



Seata-Seaata的高可用异地容灾架构搭建

服务名	分组名称	集群数目	实例数	健康实例数	触发保护阈值	操作
seata-tc-server	SEATA_GROUP	2	2	2	false	<a href="#">详情</a> <a href="#">示例代码</a> <a href="#">删除</a>

点进详情查看：

服务详情

服务名: seata-tc-server

分组: SEATA\_GROUP

保护阈值: 0

元数据: 1

服务路由类型: none

集群: HZ

元数据过滤

keyvalue

添加过滤

IP	端口	临时实例	权重	健康状态	元数据
192.168.8.118	8092	true	1	true	

集群: SH

元数据过滤

keyvalue

添加过滤

IP	端口	临时实例	权重	健康状态	元数据
192.168.8.118	8091	true	1	true	

2.将事务组映射配置到nacos

接下来，我们需要将tx-service-group与cluster的映射关系都配置到nacos配置中心。

新建配置

Data ID: client.properties

Group: DEFAULT\_GROUP

更多高级选项

描述:

配置格式: ☐ TEXT ☐ JSON ☐ XML ☐ YAML ☐ HTML ☒ Properties

配置内容: 1

配置的内容如下(client.properties):

📄

1.

# 事务组映射关系SH、HZ
2.

service.vgroupMapping.seata-demo=SH
3.
4.

service.enableDegrade=false
5.

service.disableGlobalTransaction=false
6.

# 与TC服务的通信配置
7.

transport.type=TCP
8.

transport.server=NIO



Seata-Seata的高可用异地容灾架构搭建

```
11. transport.threadFactory.bossThreadPrefix=NettyBoss
12. transport.threadFactory.workerThreadPrefix=NettyServerNIOWorker
13. transport.threadFactory.serverExecutorThreadPrefix=NettyServerBizHandler
14. transport.threadFactory.shareBossWorker=false
15. transport.threadFactory.clientSelectorThreadPrefix=NettyClientSelector
16. transport.threadFactory.clientSelectorThreadSize=1
17. transport.threadFactory.clientWorkerThreadPrefix=NettyClientWorkerThread
18. transport.threadFactory.bossThreadSize=1
19. transport.threadFactory.workerThreadSize=default
20. transport.shutdown.wait=3
21. # RM配置
22. client.rm.asyncCommitBufferLimit=10000
23. client.rm.lock.retryInterval=10
24. client.rm.lock.retryTimes=30
25. client.rm.lock.retryPolicyBranchRollbackOnConflict=true
26. client.rm.reportRetryCount=5
27. client.rm.tableMetaCheckEnable=false
28. client.rm.tableMetaCheckerInterval=60000
29. client.rm.sqlParserType=druidd
30. client.rm.reportSuccessEnable=false
31. client.rm.sagaBranchRegisterEnable=false
32. # TM配置
33. client.tm.commitRetryCount=5
34. client.tm.rollbackRetryCount=5
35. client.tm.defaultGlobalTransactionTimeout=60000
36. client.tm.degradeCheck=false
37. client.tm.degradeCheckAllowTimes=10
38. client.tm.degradeCheckPeriod=2000
39.
40. # undo日志配置
41. client.undo.dataValidation=true
42. client.undo.logSerialization=jackson
43. client.undo.onlyCareUpdateColumns=true
44. client.undo.logTable=undo_log
45. client.undo.compress.enable=true
46. client.undo.compress.type=zip
47. client.undo.compress.threshold=64k
48. client.log.exceptionRate=100
```

View Code

3.微服务读取nacos配置

接下来，需要修改每一个微服务的application.yml文件，让微服务读取nacos中的client.properties文件：

```
1. seata:
2.   config:
3.     type: nacos
4.   nacos:
5.     server-addr: tajia-nacos:8848
6.     username: nacos
7.     password: nacos
8.     group: SEATA_GROUP
9.     data-id: client.properties
```

seata完整配置如下：

```
1. seata:
2.   registry: # TC服务注册中心的配置，微服务根据这些信息去注册中心获取tc服务地址
3.     # 参考tc服务自己的registry.conf中的配置
4.     type: nacos
```



## Seata-Seata的高可用异地容灾架构搭建

```
namespace:
8.      group: SEATA_GROUP
9.      application: seata-tc-server # tc服务在nacos中的服务名称
10.     tx-service-group: seata-demo # 事务组，根据这个获取tc服务的cluster名称
11.     #service:
12.     # vgroup-mapping: # 事务组与TC服务cluster的映射关系
13.     # seata-demo: SH
14.     config:
15.     type: nacos
16.     nacos:
17.     server-addr: tajia-nacos:8848
18.     username: nacos
19.     password: nacos
20.     group: SEATA_GROUP
21.     data-id: client.properties
```

重启微服务，现在微服务到底是连接tc的SH集群，还是tc的HZ集群，统一由nacos的client.properties来决定。

nacos配置的service.vgroupMapping.seata-demo是SH集群，此时观察微服务和seata-server控制台的日志信息，切换service.vgroupMapping.seata-demo，微服务和seata-server控制台是否也跟着变化。

storage-service、account-service、order-service控制台日志已连接到8091：

```
1. 10-14 16:32:10:625 INFO 10816 --- [eoutChecker_1_1] i.s.c.r.netty.NettyClientChannelManager : will connect to 192.168.
2. 10-14 16:32:10:628 INFO 10816 --- [eoutChecker_1_1] i.s.core.rpc.netty.NettyPoolableFactory : NettyPool create channe
3. 10-14 16:32:10:649 INFO 10816 --- [eoutChecker_1_1] i.s.c.rpc.netty.TmNettyRemotingClient : register TM success. clie
4. 10-14 16:32:10:649 INFO 10816 --- [eoutChecker_1_1] i.s.core.rpc.netty.NettyPoolableFactory : register success, cost 12
```

端口8091的seata-server打印如下信息：

```
1. 16:32:10.646 INFO --- [ttyServerNIOWorker_1_4_32] i.s.c.r.processor.server.RegTmProcessor : TM register success,mes
2. 16:32:14.902 INFO --- [ttyServerNIOWorker_1_5_32] i.s.c.r.processor.server.RegTmProcessor : TM register success,mes
3. 16:32:55.114 INFO --- [ttyServerNIOWorker_1_6_32] i.s.c.r.processor.server.RegTmProcessor : TM register success,mes
```

**编辑nacos的client.properties配置文件，把service.vgroupMapping.seata-demo设置为HZ，微服务和seata-server控制台是否连接到8092这台seata-server。**

storage-service、account-service、order-service控制台日志已连接到8092：

```
1. 10-14 16:32:10:625 INFO 10816 --- [eoutChecker_1_1] i.s.c.r.netty.NettyClientChannelManager : will connect to 192.168.
2. 10-14 16:32:10:628 INFO 10816 --- [eoutChecker_1_1] i.s.core.rpc.netty.NettyPoolableFactory : NettyPool create channe
3. 10-14 16:32:10:649 INFO 10816 --- [eoutChecker_1_1] i.s.c.rpc.netty.TmNettyRemotingClient : register TM success. clie
4. 10-14 16:32:10:649 INFO 10816 --- [eoutChecker_1_1] i.s.core.rpc.netty.NettyPoolableFactory : register success, cost 12
5. 10-14 16:33:27:778 INFO 10816 --- [h_RMROLE_1_1_32] i.s.c.r.p.client.RmUndoLogProcessor : rm handle undo log pro
6. 10-14 16:33:27:778 WARN 10816 --- [h_RMROLE_1_1_32] io.seata.rm.RMHandlerAT : Failed to get dataSourceProxy fo
7. 10-14 16:45:30:628 INFO 10816 --- [eoutChecker_1_1] i.s.c.r.netty.NettyClientChannelManager : will connect to 192.168.
8. 10-14 16:45:30:629 INFO 10816 --- [eoutChecker_1_1] i.s.core.rpc.netty.NettyPoolableFactory : NettyPool create channe
9. 10-14 16:45:30:636 INFO 10816 --- [eoutChecker_1_1] i.s.c.rpc.netty.TmNettyRemotingClient : register TM success. clie
10. 10-14 16:45:30:636 INFO 10816 --- [eoutChecker_1_1] i.s.core.rpc.netty.NettyPoolableFactory : register success, cost 5 i
11. 10-14 16:45:30:683 INFO 10816 --- [eoutChecker_2_1] i.s.c.r.netty.NettyClientChannelManager : will connect to 192.168.
12. 10-14 16:45:30:683 INFO 10816 --- [eoutChecker_2_1] i.s.c.rpc.netty.RmNettyRemotingClient : RM will register :deduct,jc
13. 10-14 16:45:30:684 INFO 10816 --- [eoutChecker_2_1] i.s.core.rpc.netty.NettyPoolableFactory : NettyPool create channe
14. 10-14 16:45:30:697 INFO 10816 --- [eoutChecker_2_1] i.s.c.rpc.netty.RmNettyRemotingClient : register RM success. clie
15. 10-14 16:45:30:698 INFO 10816 --- [eoutChecker_2_1] i.s.core.rpc.netty.NettyPoolableFactory : register success, cost 7 i
```

端口8092的seata-server打印如下信息：

Seata-Seata的高可用异地容灾架构搭建



3. 16:45:24.977 INFO --- [rverHandlerThread\_1\_1\_500] i.s.c.r.processor.server.RegRmProcessor : RM register success,me:  
4. 16:45:25.112 INFO --- [ttyServerNIOWorker\_1\_3\_32] i.s.c.r.processor.server.RegTmProcessor : TM register success,mes  
5. 16:45:25.177 INFO --- [rverHandlerThread\_1\_2\_500] i.s.c.r.processor.server.RegRmProcessor : RM register success,me:  
6. 16:45:30.635 INFO --- [ttyServerNIOWorker\_1\_5\_32] i.s.c.r.processor.server.RegTmProcessor : TM register success,mes  
7. 16:45:30.696 INFO --- [rverHandlerThread\_1\_3\_500] i.s.c.r.processor.server.RegRmProcessor : RM register success,me:

本文章为转载内容，我们尊重原作者对文章享有的著作权。如有内容错误或侵权问题，欢迎原作者联系我们进行内容更正或删除文章。

赞

收藏

2评论

分享

举报

上一篇：实验5：开源控制器实践——POX

下一篇：今天开始学Pattern Recognition and Machine L...



提问和评论都可以，用心的回复会被更多人看到

评论

全部评论 (2)

🕒 最新 🔥 最热



e2da8adf6dfb

2年前

挺好的，点赞！

💬 回复 👍 点赞



9460078285dd

2年前

有幸看到博主的博文！感谢博主的细致介绍！

💬 回复 👍 点赞

相关文章

mongodb异地容灾

# MongoDB异地容灾实现流程## 1. 引言在开发过程中，数据库的容灾备份是非常重要的的一环，可以保证数据的安全性和可用性...

数据库 数据库恢复 数据

MySQL基操---高可用架构MMM搭建与容灾测试

MMM介绍MMM（Master-Master replication manager for MySQL）是一套支持双主故障切换和双主日常管理的脚本程序。MMM...

测试 MySQL

【Seata】初识Seata

Seata是 2019 年 1 月份蚂蚁金服和阿里巴巴共同开源的分布式事务解决方案。致力于提供高性能和简单易用的分布式事务服务...

Java 分布式事务 Seata 微服务 全局事务