

表广播机制在 MyCat 中的实现

王 锦, 梁正和, 王法强

(河海大学 计算机与信息学院, 江苏 南京 211100)

摘 要:海量数据存储与访问是系统设计与使用的瓶颈问题,利用开源的分布式存储数据库 MyCat,通过对数据进行水平切分,将不同的表映射到不同的数据引擎上,从逻辑上实现了数据库容量的扩充。但是,利用数据操纵语言 DML 访问 MyCat 的全局表时,如果遇到异常情况致使某个节点断开连接,可能导致出现多个不同分库数据不一致的问题。针对 MyCat 分布式数据库中间件的这一弱分布式事务问题,提出一种表广播机制的解决方案。该方案通过改变原始 MyCat 在 DML 访问时对全局表的执行逻辑,将作用于所有节点的操作改成对主节点的操作,再通过表广播机制完成各节点间的数据同步,实现各节点间的数据一致性。并利用通用的 DML 访问实现表广播机制的 MyCat 数据库中的数据对 MyCat 进行了性能测试,测试结果表明,基于 MyCat 的表广播机制在 SQL 高效率执行的同时,也保证了各数据节点间的数据一致性。

关键词:表广播; MyCat; 分布式数据库; 数据库中间件

中图分类号: TP311.133.1

文献标识码: A

文章编号: 1673-629X(2018)03-0042-05

doi:10.3969/j.issn.1673-629X.2018.03.009

Realization of Table Broadcasting Mechanism in MyCat

WANG Jin, LIANG Zheng-he, WANG Fa-qiang

(School of Computer and Information, Hohai University, Nanjing 211100, China)

Abstract: The storage and access of massive data is the bottleneck of system design and use. The MyCat, an open source distributed storage database, makes the different tables mapped to different data engines through the horizontal segmentation of the data, which realizes the capacity extension of database logically. However, when using the data manipulation language (DML) to access the global table of MyCat, if a node is disconnected due to an abnormal situation, it may lead to the inconsistencies in the number of different sub-library data. For the problem of weak distributed transaction in MyCat distributed database middleware, we put forward a kind of solution about table broadcast mechanism. By changing the execution logic of the global table while the original MyCat accessing to the DML, the program changes the operation of all nodes to that of the master node, and implements the data consistency between the nodes by data synchronization between the nodes of the table broadcast mechanism. Then the MyCat is tested by common DML accessing to achieve the data in MyCat database of table broadcast mechanism, which shows that the table broadcast mechanism based on the MyCat is efficient in SQL and also guarantees the data consistency between the nodes.

Key words: table broadcast mechanism; MyCat; distributed database; database middleware

0 引言

随着互联网技术的飞速发展,移动网络、局域网、广域网、Internet 得到了巨大的发展,集中部署的单机数据库系统已经不能够适应这样的环境^[1],使用服务器集群来解决海量数据^[2]成为必然趋势,分布式数据库应运而生。MyCat 分布式数据库中间件^[3]由阿里巴巴的开源项目 cobar^[4]发展而来,其本身提供了分布式资源整合方案。该产品具有双机热备份、负载均衡、SQL 语句重写、读写分离^[5]、查询路由、多台数据库并

行处理以及结果集合并等功能^[6],并提供了对 MySQL、PostgreSQL 等应用广泛的开源关系型数据库的分布式支持。该分布式数据库中间件为用户提供廉价的数据库集群,平滑地将现有的单机集中式数据库和应用迁移到“云”^[7]端,同时保留原有数据库的查询能力。

通过 MyCat 可以搭建以 MySQL 为底层节点的分布式数据库系统,系统通过 MySQL 的通信协议^[8]与用户以及底层数据库通信。对于用户,通过该中间件

收稿日期:2017-04-19

修回日期:2017-08-23

网络出版时间:2017-12-05

基金项目:国家自然科学基金(61272543)

作者简介:王 锦(1990-),女,硕士研究生,研究方向为分布式系统、大数据;梁正和,博士,副教授,研究方向为 EPR、分布式系统。

网络出版地址: <http://kns.cnki.net/kcms/detail/61.1450.TP.20171205.0906.052.html>

能够透明地访问整个集群,故而原有的基于 MySQL 开发的应用均无需修改,直接迁移到分布式系统中。该系统在中间件中建立了完整的 Schema(模式)、Table(表)、User(用户)的逻辑模型,将整套模型通过逻辑规则映射到每个数据节点上的 MySQL 实例中,并可以实现事务处理。

面对一些数据量巨大的表,可以通过 MyCat 对数据进行水平切分^[9],在保证每个分片节点的表结构一致的情况下,将表映射到不同的数据引擎上,从逻辑上实现数据库容量的扩充。MyCat 不支持跨分片连接查询,需要将其关联的表设置为全局表分布在各个分片上。由于 MyCat 的弱 Xa 事务^[10]机制,频繁操纵全局表可能会导致异常情况的发生:某个用户在修改好全局表提交的一瞬间,忽然某个节点出错了,可能会出现某些节点已成功修改而出错节点数据无变更,使得各个节点数据不一致的情况。对于一个大企业,该类问题是难以忍受的。MyCat 中的表广播机制改变了原始的对全局表的操作逻辑,把原来的同时向多个节点发送的 DML^[11]消息,修改为只向全局表中的主节点发送广播消息,待主节点接收消息后再向其他节点发送广播,通知其他节点有新消息到达,实现各节点间的数据同步。若主节点在修改过程中突遇异常情况致使操作中中断,则所有节点均无修改操作;若其他节点在同步过程中出现意外中断,待故障排除后,继续未完成的同步工作,成功解决了多个节点数据不一致的问题。

1 MyCat 中的表广播机制

1.1 MyCat 中的弱 XA 事务

MyCat 中的事务主要包括 SQL 不跨分片事务和 SQL 跨分片事务^[3]。对于 SQL 不跨分片这种情况,SQL 仅仅是在一个数据节点上执行,此时 MyCat 事务模式同标准的数据库事务模式^[12]完全一致,或提交、或回滚,能够保证强一致性^[13];对于 SQL 跨分片的事务,首先事务内的 SQL 在各自的分片上执行并返回状态码,若某个分片上的返回码为 ERROR,则 MyCat 认为事务失败,应用端只能回滚(rollback)事务,MyCat 收到回滚指令后,依次回滚事务中涉及到的所有分片;若事务中的所有 SQL 的执行都返回成功(OK)的返回码,则应用程序提交事务,由 MyCat 同时向事务中涉及到的节点发送提交事务的指令(commit),在 commit 的时候,若某个节点出错,MyCat 也无法等节点恢复后重新 commit,出现部分节点 commit 成功而部分节点没有 commit 的情况。由于跨分片事务的第二阶段无法保证强一致性,称 MyCat 是一种弱 XA 事务模式。

1.2 表广播机制的基本原理

表广播机制的实现主要分为三个步骤:

(1)用户发送 DML(数据操作语言)请求,主机接收到命令,向主节点发送广播。主节点接收到消息,将消息记录到二进制文件,并将消息发送给其他从节点。

(2)从节点读取消息,并将消息写入到自己的中继日志。

(3)从节点重做中继日志中的事件,并改变自己的数据。

如图 1 所示,主机接受请求后向主节点发送广播,主节点接收消息并记录二进制文件。在每个事务更新数据完成之前,主节点在二日志文件中记录这些改变。MySQL 线程将事务串行地写入二进制日志,在事件写入完成后,主节点通知存储引擎提交事务并向其他从节点发送广播。

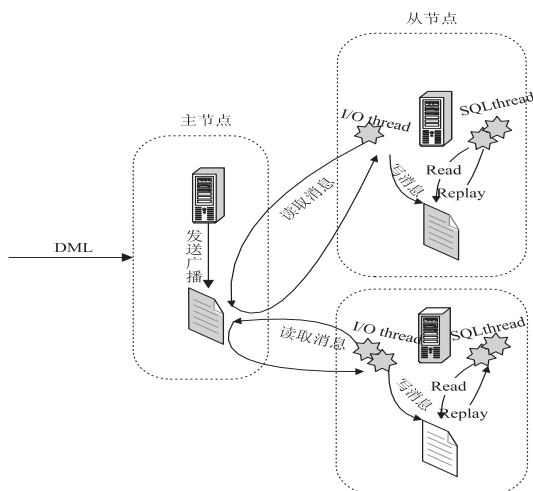


图 1 表广播机制原理图

从节点接收到消息,将主节点的二进制日志拷贝到自己的中继日志。首先,从节点开始一个工作线程(I/O 线程),I/O 线程在主节点上打开一个普通的连接,然后开始二进制转存。二进制转存程序从主节点的二进制日志中读取消息事件,如果没有新的消息,它会睡眠并等待主节点发送新的广播。

SQL 从线程从中继日志读取消息事件,并重放其中的事件以更新从节点中的数据,使其与主节点中的数据一致。

1.3 表广播机制在 MyCat 中的应用

MyCat 是一个开源的分布式数据库系统,是一个实现了 MySQL 协议的 server,前端用户可以把它看作一个数据库代理^[14],用 MySQL 客户端工具和命令行访问,而其后端可以用 MySQL 原声(Native)协议^[15]与多个 MySQL 服务器通信,也可以用 JDBC 协议与大多数主流数据库服务器^[16]通信。MyCat 中有两种典型类型的表,一种是按照某种给定的分片规则,将数据进行水平切分,把一个大表水平分割成 N 个小表,存储在后端 MySQL 服务器或者其他不同的数据库;另一种是全局表,MyCat 接收外界发送来的 SQL 语句,

将 DML SQL 语句分别发送到各个数据库节点上依次执行,图 2 形象地描述了全局表的 DML 操作过程。若对全局表频繁地进行增删改操作,由于 MyCat 的弱 XA 机制,很容易发生各节点数据不一致的情况。

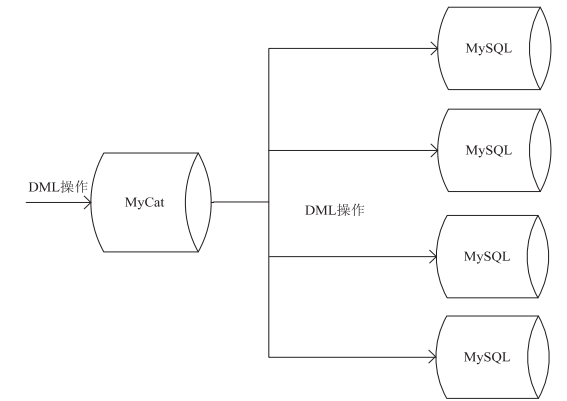


图 2 原始 MyCat 全局表的 DML 操作过程

MyCat 中的表广播机制是针对全局表的,为全局表新增了一个属性,当该属性值为 true 时,改变原始 SQL 语句的执行逻辑。MyCat 接收到 DML 命令,向主节点发送广播,主节点执行命令并保存二进制消息文件,同时向从节点推送消息,从节点接收到广播,同步主节点的数据。实现了表广播机制的 MyCat 即使在修改数据时出现节点异常的情况,待异常解除后,会主动读取存放消息的日志文件、同步数据,保证了强一致性。

图 3 形象展示了 DML 操作在表广播机制下的执行过程,即当 MyCat 接收到插入类型的 SQL 语句时,只向主节点发送相应的 SQL 语句,再通过日志文件,同步到其他节点。

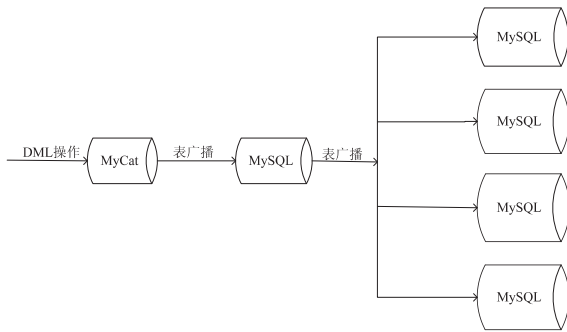


图 3 表广播机制下的 DML 操作过程

2 表广播机制在 MyCat 中的实现

2.1 配置文件的修改

MyCat 启动时,首先读取配置文件,根据其中 table 标签的设置信息,判断各张表的类型。在配置文件 schema.xml 的 table 标签中增加一个属性“writeOneNode”作为启用表广播机制的标志,当逻辑表的类型是全局表即“type = goble”时,才启用表广播功能机制。 万方数据

新增的属性为布尔类型,默认值为“false”。“writeOneNode = true”时,过滤 DML 操作的 SQL 语句,通过表广播作用于主节点;当“writeOneNode = false”时,DML 操作的 SQL 语句按照原始逻辑作用于所有节点。

2.2 配置文件的初始化

通过 Java 应用程序访问 XML 时,要先把 Java 对象转化成 XML 文件(称作 marshal),再将 XML 文件中的内容转化成相应的 Java 对象(称作 unmarshal),该对象需要用 JAXB(Java architecture for XML binding,XML 绑定的 Java 体系结构)注解来标注^[17]。因此需要修改注解类 Schemas.java,在其内部类中添加“writeOneNode”的属性设置,使配置文件“schema.xml”中的内容 unmarshal 成 Table 类型的对象时,包含“writeOneNode”字段的相应信息。

初始化 XML 配置文件类 XmlToYaml.java 中,增加对参数“writeOneNode”的初始化功能,在启动程序时加载配置文件,通过解析 XML 配置文件,对配置文件中设置的相关参数进行初始化,在 Schemas 的各个内部静态类中实例化本类对象,最后再通过 HashMap 存储好所需要配置的 Key-value 键值对,对外提供本类实例化对象。

2.3 MyCat 中表广播机制的执行流程

MyCat 中的表广播机制是针对于全局表的 DML 操作而提出的,且只有在 writeOneNode = true 时其功能才会奏效,因此需要增加新的 SQL 执行逻辑。如图 4 所示,MyCat 对接收到的 sql 语句进行以下判断:

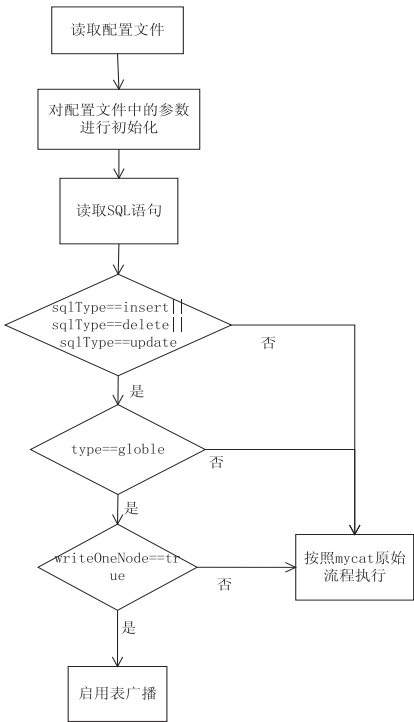


图 4 MyCat 中的表广播机制执行流程

- (1)sql 语句的类型是否为 insert 或 update 或 delete;
 - (2)待插入的表是否为全局表;
 - (3)是否写入单节点,即 writeOneNode=true。
- 只有三个条件同时满足,才能将原始的多节点插入功能修改为单节点插入,启用表广播功能。

3 实验与分析

为了分析表广播机制在 MyCat 中的实现方案的可行性,在真实环境下进行实验分析。

3.1 实验设置

实验环境:三台配置完全相同的服务器。配置如下:操作系统为 RedHat7.2;1*6 Core 的 CPU;内存 16 G;硬盘 16*1 T;千兆网卡。

步骤 1:新建数据库和数据表。

实验中的全局表“teacher”具有 4 个分片,分别位于 4 个不同的数据库中,数据库名称分别为 MyCat1, MyCat2, MyCat3, MyCat4, 所属服务器 ip 为 172.172.0.104,并在各个数据库中分别新建表“teacher”(含有字段:tid,name,sex,class;tid 为关键字);在另外一台 ip 为 172.172.0.106 的服务器上也新建 4 个数据库:broadcastMyCat1、broadcastMyCat2、broadcastMyCat3、broadcastMyCat4,每个数据库中也分别包含同样表结构的“teacher”表,并配置 broadcastMyCat1 为 broadcastMyCat2、broadcastMyCat3、broadcastMyCat4 的主数据库,broadcastMyCat2、broadcastMyCat3、broadcastMyCat4 为 broadcastMyCat1 的从数据库,启用表广播功能。

步骤 2:安装 MyCat。

在 ip 为 172.172.0.107 的服务器上分别安装官方 MyCat(端口号为 8068,命名为 MyCat_8068)和实现表广播机制的 MyCat(端口号为 8069,命名为 broadcast-MyCat_8069),进行正确的配置文件设置,两者的区别如下:

MyCat_8068 的 schema.xml 部分配置如下:

```
<schema name="STUDENTDB" checkSQLschema="false" sqlMaxLimit="100">
  <table name="teacher" primaryKey="tid" dataNode="dn1, dn2, dn3, dn4" type="global" />
</schema>
```

broadcastMyCat_8069 中的 schema.xml 配置如下:

```
<schema name="STUDENTDB" checkSQLschema="false" sqlMaxLimit="100">
  <table name="teacher" primaryKey="tid" dataNode="dn1, dn2, dn3, dn4" type="global" writeOneNode="true"/>
</schema>
```

完成配置文件的设置后,分别启动 MyCat_8068

和 broadcastMyCat_8069。

3.2 实验结果分析

实验 1:为验证 MyCat 中的表广播机制,实现了原始 MyCat 中全局表功能,通过程序分别向 MyCat_8068 和 broadcastMyCat_8069 中的 teacher 表中连续作插入操作。一段时间后,分别到本地的各个数据库中查询 teacher 表中的数据,对各个表中的记录总数和数据内容进行对比,结果如表 1 所示。

表 1 正常插入时的数据对比表

数据库名称	起始 tid	结束 tid	记录条数
MyCat1	1	2 581	2 581
MyCat2	1	2 581	2 581
MyCat3	1	2 581	2 581
MyCat4	1	2 581	2 581
broadcastMyCat1	1	2 637	2 637
broadcastMyCat2	1	2 637	2 637
broadcastMyCat3	1	2 637	2 637
broadcastMyCat4	1	2 637	2 637

由表 1 可知,各个表中的记录数量几乎相同,同一个 MyCat 下各个节点中的数据均一致。结果表明,在各个数据节点运行正常的情况下,原始的 MyCat 和实现表广播机制的 MyCat 均可以做到数据的高效插入,且保证了各个节点间的数据一致性。

实验 2:为验证在 DML 操作过程中出现节点异常后实现表广播机制的 MyCat 仍旧能保证各节点间的数据一致性,清空各个数据表中的数据后,通过程序分别向 MyCat_8068 和 MyCat_8069 中的 teacher 表中连续作插入操作,在插入过程中,关闭 MyCat2 和 broadcastMyCat2 两台数据库,一段时间后恢复两台数据库,多次尝试此操作,并对比各个 MySQL 数据库中的数据状况。

表 2 记录了各个节点异常消除后的数据情况,原始 MyCat 下的异常节点中的数据明显少于其他节点中的数据,而实现表广播机制的 MyCat 下的四个数据节点中的数据始终一致。

表 2 异常解除后的数据对比表

数据库名称	起始 tid	结束 tid	记录条数
MyCat1	1	5 332	5 332
MyCat2	1	5 207	5 207
MyCat3	1	5 332	5 332
MyCat4	1	5 332	5 332
broadcastMyCat1	1	5 491	5 491
broadcastMyCat2	1	5 491	5 491
broadcastMyCat3	1	5 491	5 491
broadcastMyCat4	1	5 491	5 491

由实验 2 结果可以看出,MyCat 的弱 XA 机制,确

实导致了节点间的数据不一致,而实现了表广播机制的 MyCat,在异常节点恢复正常后,成功同步数据,保证与主节点的数据同步。

以上实验表明,实现表广播机制的 MyCat 的可靠性较高,在保证数据高效插入的同时,也保证了数据一致性。

4 结束语

提出了一种表广播机制在 MyCat 中的实现方案,通过对 MyCat 提供的操作全局表的相关功能进行优化,把对多个节点的操作修改为对单个节点的操作,有效解决了 MyCat 面临的弱 XA 问题。实验结果表明,该方案在实现全局表原有功能的同时,有效提高了全局数据表中各节点数据的一致性。

目前,对于分库表的查询操作,MyCat 只会返回各分库合并后的数据,在排序、分页等功能上做得还不够完善,因此,未来计划运用其他的方法解决这类问题。

参考文献:

- [1] 庄天红. 常用数据库系统性能解析[J]. 微型电脑应用, 1999,15(2):52-54.
- [2] BOYD D, CRAWFORD K. Critical questions for big data [J]. Information Communication & Society, 2012, 15(5): 662-679.
- [3] 项 凯. 面向海量高并发数据库中间件的研究与应用 [D]. 上海:上海交通大学, 2015.
- [4] 邱 硕. Cobar 的架构与实践[J]. 程序员, 2012(9):90-93.
- [5] 祝雄锋. 数据库集群中间件 MySQL Proxy 研究与分析 [D]. 武汉:武汉理工大学, 2011.
- [6] 王 葱. 基于 MyCAT 的分布式数据存储研究与应用 [D]. 上海:东华大学, 2016.
- [7] HAYES B. Cloud computing [J]. Communications of the ACM, 2008, 51(7):9-11.
- [8] 安延文. 数据库审计系统中 MySQL 协议的研究与解析 [D]. 北京:华北电力大学, 2016.
- [9] 杨 晶, 刘天时, 马 刚. 分布式数据库数据分片与分配 [J]. 现代电子技术, 2006, 29(18):119-121.
- [10] 赵 艳, 李 钧. 异构数据源分布式事务处理研究[J]. 计算机工程, 2009, 35(4):69-71.
- [11] 胡百敬, 陈俊宇, 杨先民, 等. SQL Server 2005 T-SQL 数据库设计[M]. 北京:电子工业出版社, 2008.
- [12] 黄雅萍, 刘晓强, 吴成义. 基于 MySQL 和 PHP 的分布式事务处理[J]. 东华大学学报:自然科学版, 2011, 37(1):81-85.
- [13] 张旭刚, 李东辉, 俞 俊, 等. 基于 zookeeper 和强一致性复制实现 MySQL 分布式数据库集群[J]. 微型电脑应用, 2016, 32(1):77-80.
- [14] 徐 恪, 刘亚霄, 刘卫东. 数据库应用系统中的安全访问代理的设计与实现[J]. 计算机工程与应用, 2000, 36(1):105-107.
- [15] 李曙强, 蒋树春, 吕 兵. 一种基于 mysql 数据库的 sql 信息采集审计系统:CN, CN103488797A[P]. 2013-10-14.
- [16] 黄春华. 常用数据库的比较[J]. 科技信息, 2011(14):200.
- [17] 李占波, 李 娜. XML 数据在关系数据库中的存储[J]. 微机计算机信息, 2007, 23(27):192-194.
- [18] DC [J]. Information and Software Technology, 2006, 48(7): 433-440.
- [9] MALEVRIS N, YATES D F. The collateral coverage of data flow criteria when branch testing[J]. Information and Software Technology, 2006, 48(8):676-686.
- [10] WU X, LI J J, WEISS D, et al. Coverage-based testing on embedded systems[C]//Proceedings of the 29th international conference on software engineering workshops. Minneapolis, Minnesota, US: IEEE, 2007:204-209.
- [11] 郭 锐, 李 博, 彭宝新. 用于覆盖测试的代码插桩程序设计与实现[J]. 科学技术与工程, 2013, 13(30):9072-9077.
- [12] 张 荣, 王曙燕. 基于插桩技术的动态测试研究与实现 [J]. 现代电子技术, 2011, 34(4):50-52.
- [13] 路 翠. 嵌入式软件白盒测试中插桩技术的研究与应用 [D]. 北京:北京工业大学, 2010.
- [14] 王 伟. 基于多 Agent 的嵌入式软件测试系统研究与实现 [D]. 南京:南京航空航天大学, 2011.
- [15] 王叔娥. 嵌入式软件动态分析技术的研究[D]. 成都:电子科技大学, 2011.
- [16] 朱 丽. 嵌入式软件动态测试平台的研究与实现[D]. 福州:福建师范大学, 2013.

(上接第 41 页)

- [1] ting [M]. 3rd ed. Hoboken, New Jersey, U. S: John Wiley & Sons, 2012.
- [2] 赵 翀, 孙 宁. 软件测试技术:基于案例的测试[M]. 北京:机械工业出版社, 2013.
- [3] 姜 文, 刘立康. 基于 SVN 的应用软件持续集成[J]. 计算机测量与控制, 2016, 24(3):109-113.
- [4] DUVAL P M, MATYAS S, GLOVE A. 持续集成软件质量改进和风险降低之道[M]. 北京:电子工业出版社, 2012.
- [5] 姜 文, 刘立康. 软件配置管理中的基线问题研究[J]. 计算机技术与发展, 2016, 26(6):6-10.
- [6] 于全喜. 嵌入式软件路径覆盖测试数据采集研究与实现 [D]. 西安:西安理工大学, 2009.
- [7] DELAMARO M E, VINCENZI A M R, MALDONADO J C. A strategy to perform coverage testing of mobile applications [C]//Proceedings of the 2006 international workshop on automation of software test. New York, NY, USA: ACM, 2006:118-124.
- [8] WOODWARD M R, HENNEL M A. On the relationship between two control-flow coverage criteria all JJ-paths MC-