MariaDB新特性剖析

京东云——张金鹏

DTCC

2015中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2015

大数据技术探索和价值发现

内容提要

1. MariaDB发展历程
2. MariaDB新特性概要
3. 线程池技术剖析
4. binlog group commit技术剖析
5. 多源复制剖析
6. MariaDB面临的挑战

1、MariaDB发展历程

1. 2008年1月， Sun以10亿美元收购MySQL AB。
2. 2009年4月，Oracle以74亿美元收购Sun。
3. 2009年，MySQL创始人Monty出于以下几个原因创立了MySQL的分支MariaDB：
   - MySQL核心开发团队是封闭的，完全没有Oracle之外的成员参加。
   - MySQL新版本的发布速度，在Oracle收购Sun之后大为减缓，有很多bugfix和新的feature，都没有及时加入到发布版本之中。
4. 维基百科、Google、Red Hat、SUSE等从MySQL迁移到MariaDB。

2、MariaDB扩展和新特性概要

1. 更多的存储引擎。除标准存储引擎外还包含Aria、XtraDB、SphinxSE、FederatedX、TokuDB、Cassandra、CONNECT、Sequence、Spider等存储引擎。
2. 线程池技术。
3. binlog group commit技术。
4. 支持微秒级别的时间精度。
5. 虚拟列、动态列。
6. 用户统计功能。
7. Kill某个用户的所有Query。
8. Query的执行进度提示。
9. 多源复制。
10. 子查询优化。
11. ……

- 虚拟列

1. CREATE TABLE example_virtual_columns(
   a INT(11) PRIMARY KEY，
   b VARCHAR(32),
   c INT(11) AS (a mod 10) VIRTUAL,
   d VARCHAR(5) as (left(b, 5)) PERSISTENT);
   虚拟列c的值将会在查询时计算，而虚拟列d的值被存储在表中，查询的时候直接从表里取出。

2. 在执行插入操作时，虚拟列使用default关键字代替就可以。如果为虚拟列指定值，将会导致错误的发生：

```
mariadb[dbtest]> INSERT INTO example_virtual_columns VALUES (16, "abcdefghijkl", default, default);
Query OK, 1 row affected (0.01 sec)

mariadb[dbtest]> SELECT * FROM example_virtual_columns;
+-------+------------------+--------+------------------+
| a     | b                | c      | d                |
+-------+------------------+--------+------------------+
| 16    | abcdefghijkl     | 6      | abcde            |
+-------+------------------+--------+------------------+
1 row in set (0.00 sec)

mariadb[dbtest]> INSERT INTO example_virtual_columns VALUES (17, "abcdefghijkl", default, "there");
ERROR 1906 (HY000): The value specified for computed column 'd' in table 'example_virtual_columns' ignored
```

- 动态列（1）

1. 动态列适合于某些不确定的场景，例如某个商品的属性个数不确定并且将来可能还会添加。
2. 想要使用动态列，首先表中必须包含blob类型的列：

```
create table assets (
item_name varchar(32) primary key,
dynamic_cols blob
);
```

3. 接下来，就可以使用MariaDB定义的动态列操作函数对动态列进行存取操作：

```
INSERT INTO assets VALUES ("MariaDB T-shirt", COLUMN_CREATE("color", "blue", "size", "XL"));
INSERT INTO assets VALUES ("Thinkpad Laptop", COLUMN_CREATE ("color", "black", "price", 500));
```

以上两条语句往assets表中插入了两行记录，接下来查询商品的颜色情况：

```
mariadb[dbtest]> SELECT item_name, COLUMN_GET(dynamic_cols, "color" as char) AS color FROM
assets;
+--------------------------+--------+
| item_name                | color |
+--------------------------+--------+
| MariaDB T-shirt          | blue  |
| Thinkpad Laptop          | black |
+--------------------------+--------+
```

- 动态列（2）

1. 此外，还可以动态删除或者增加某行的动态列：

```
mariadb[dbtest]> UPDATE assets SET dynamic_cols = COLUMN_DELETE(dynamic_cols, "price") WHERE
COLUMN_GET(dynamic_cols,"color" as char)= "black";

mariadb[dbtest]> UPDATE assets SET dynamic_cols = COLUMN_ADD(dynamic_cols, "warranty", "3 years") WHERE
item_name="Thinkpad Laptop";
```

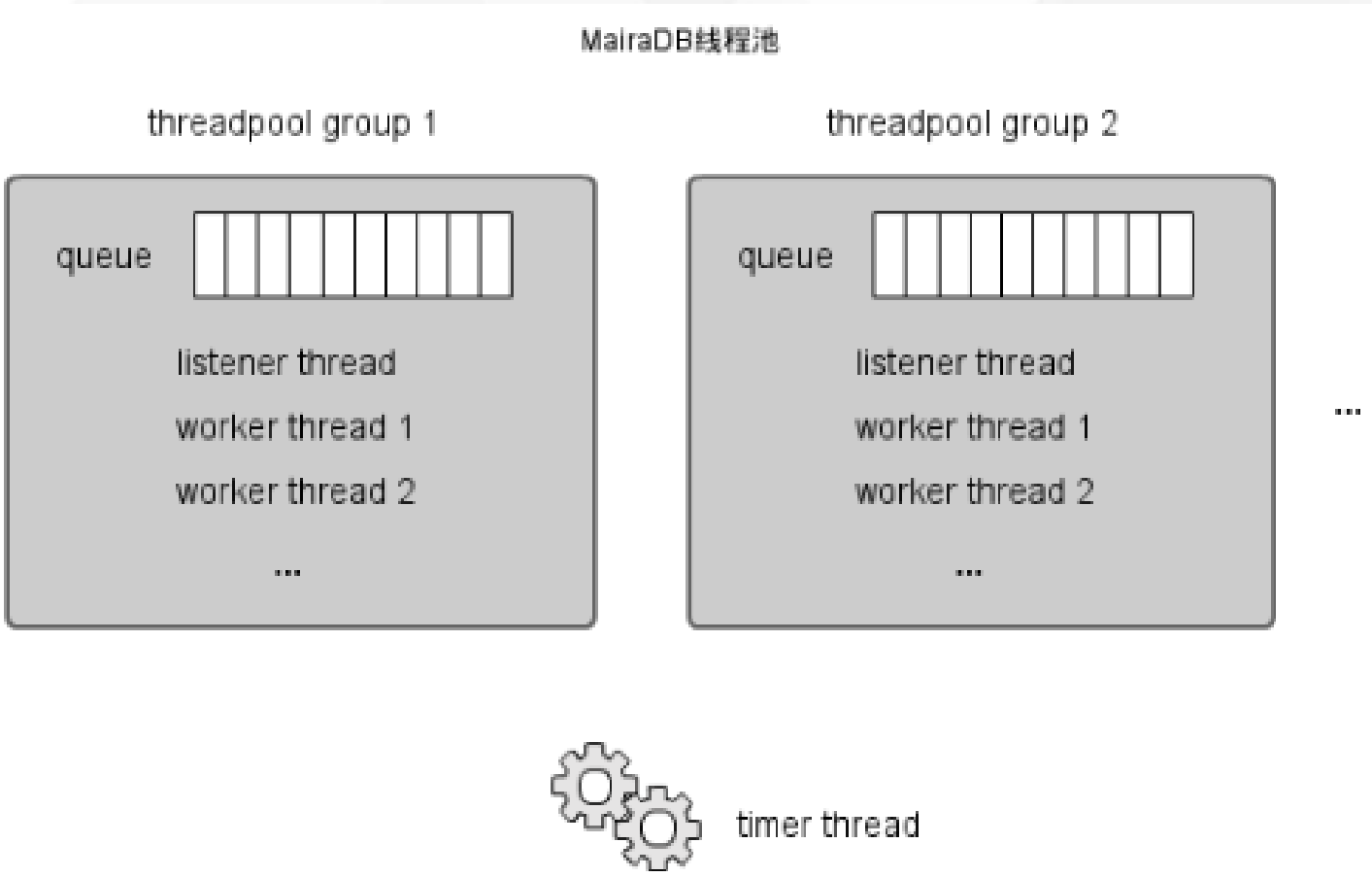
2. 你可以通过调用COLUMN_LIST函数来查看动态列的情况，或者使用COLUMN_JSON函数以JSON的格式来查看动态列以及它们对应的值：

```
mariadb[dbtest]> SELECT item_name, COLUMN_LIST (dynamic_cols) FROM assets;
+--------------------------+--------------------------------------+
| item_name                | column_list(dynamic_cols)     |
+--------------------------+--------------------------------------+
| MariaDB T-shirt          | "size","color“                       |
| Thinkpad Laptop          | "color","warranty“                    |
+--------------------------+--------------------------------------+

mariadb[dbtest]> SELECT item_name, COLUMN_JSON(dynamic_cols) FROM assets;
+--------------------------+-------------------------------------------------------------+
| item_name                | COLUMN_JSON(dynamic_cols)                                   |
+--------------------------+-------------------------------------------------------------+
| MariaDB T-shirt          | {"size":"XL","color":"blue"}                                |
| Thinkpad Laptop          | {"color":"black","warranty":"3 years"}                      |
+--------------------------+-------------------------------------------------------------+
```

3、MariaDB线程池技术

- MySQL每连接每线程模式的局限性：
  A. 最大连接数限制。参数max_connections。
  B. 随着连接数的上升，线程数上升。每个线程会占用一定系统资源的，线程数多了占用的系统资源也就多了。
  C. 线程的创建和销毁是有一定开销的。
  D. 当线程数过多时，如果其中大部分线程都处于活跃状态，将会导致频繁的上下文切换，从而造成巨大的系统开销。

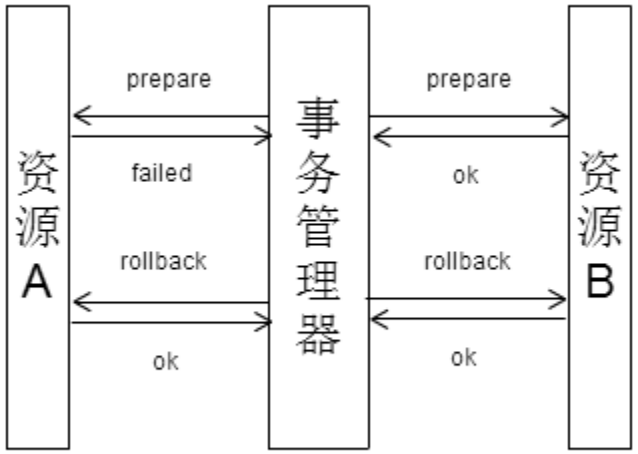

MairaDB线程池
threadpool group 1
queue
listener thread
worker thread 1
worker thread 2
...
threadpool group 2
queue
listener thread
worker thread 1
worker thread 2
...
...
timer thread

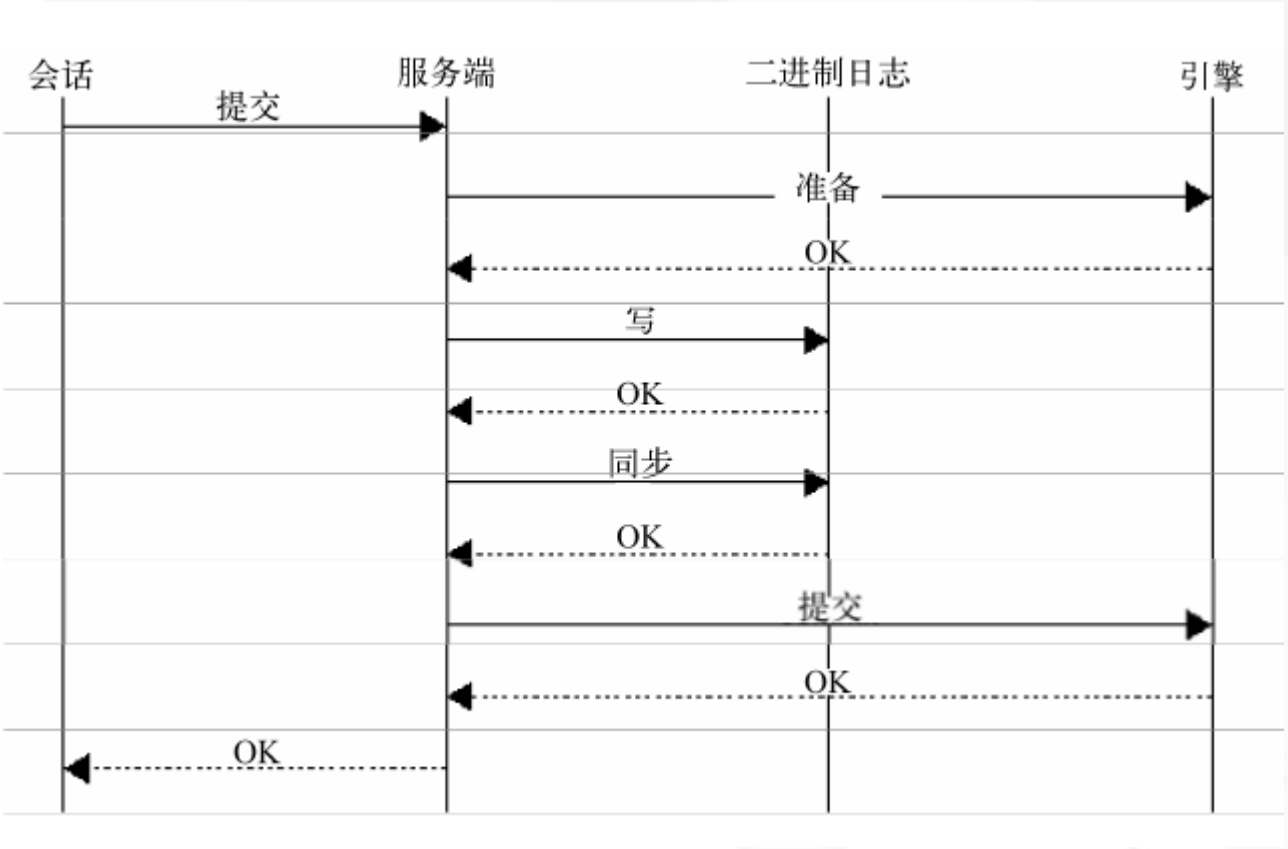
1. 由多个分组组成。参数thread_pool_size指定了线程池分组的个数，默认值为当前机器CPU的核心数。
2. 新来的连接会根据id的不同分配到不同的分组。
3. 每个分组有一个任务队列，存储待处理的连接。
4. 每个分组有一个listener线程，监听对应分组所有连接的网络事件，将有事件的连接添加到任务队列中。
5. worker线程负责处理任务队列中待处理的连接。
6. timer线程定期清理超时的客户端连接，并且检查各个分组是否处于“停滞”状态。
7. worker线程的数量会动态伸缩。各个分组的worker线程的总数最大不超过thread_pool_max_threads设置的值。
8. listener线程和worker线程之间可以进行相互转换。正常情况下当listener线程监听到分组中的某个连接需要处理的时候，会将其加入任务队列，并且唤醒worker线程进行处理。而在任务队列没有任务积压并且最近一段时间网络事件较少的情况下，listener线程不会唤醒worker线程去处理监听到的连接，而是自己作为worker线程处理该链接。worker线程在执行完任务队列的所有任务之后，在进入睡眠之前会判断当前分组是否存在listener线程，如果不存在listener线程那么自己就转化为listener线程。
9. 200,000+ connections

4、MariaDB binlog group commit技术

- 在开启binlog的情况下，MySQL/MariaDB事务的提交包括在binlog中的提交以及在存储引擎内部的提交。MySQL/MariaDB采用2PC保证事务的完整性。

- 2PC

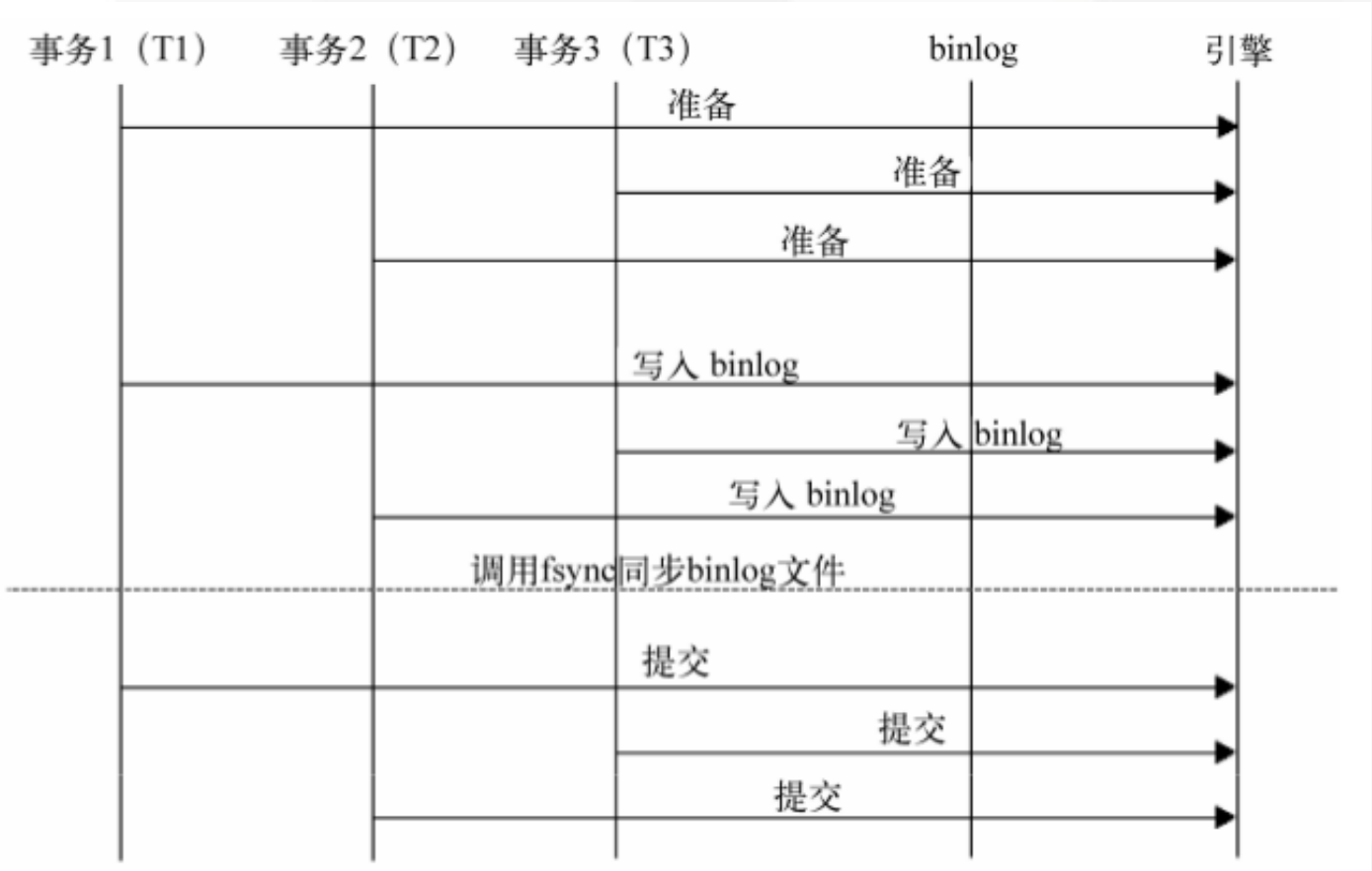




MySQL/MariaDB事务两阶段提交

1. 事务在存储引擎内部prepare（binlog也有prepare方法，只不过该方法什么也不做）。
2. 事务提交到binlog中。
3. 事务在存储引擎内部进行提交。

- 当系统在1和2期间崩溃了，事务仅仅在存储引擎内部prepare了，当MySQL重新开启的时候，会将其进行回滚。
- 当系统在2和3期间崩溃了，事务在存储引擎内部prepare了，在binlog中提交了，当MySQL重新启动的时候，会把该事务在引擎内部进行commit。因为该事务很可能已经通过binlog传播到了从库上。



多个并发提交的事务之间共享一次fsync操作对binlog进行持久化

- 多个并发需要提交的事务之间共享一次fsync操作对binlog文件进行持久化。
- 多个并发提交的事务在写binlog之前会被加入到一个队列中，位于队列头部的事务所在的线程称为leader线程，其他事务所在的线程称为follower线程。leader线程负责为队列中所有的事务进行写binlog操作，此时所有的follower线程处于等待状态，然后leader线程调用一次fsync操作，将binlog持久化，最后通知所有的follower线程可以继续往下执行。

1. 事务1
   - 获取binlog锁。
   - 将事务写入到binlog。
   - 调用fsync将对binlog的修改进行持久化。
2. 事务2
   - 获取binlog锁。
   - 将事务写入到binlog。
   - 调用fsync将对binlog的修改进行持久化。
3. 事务3
   - 获取binlog锁。
   - 将事务写入到binlog。
   - 调用fsync将对binlog的修改进行持久化。

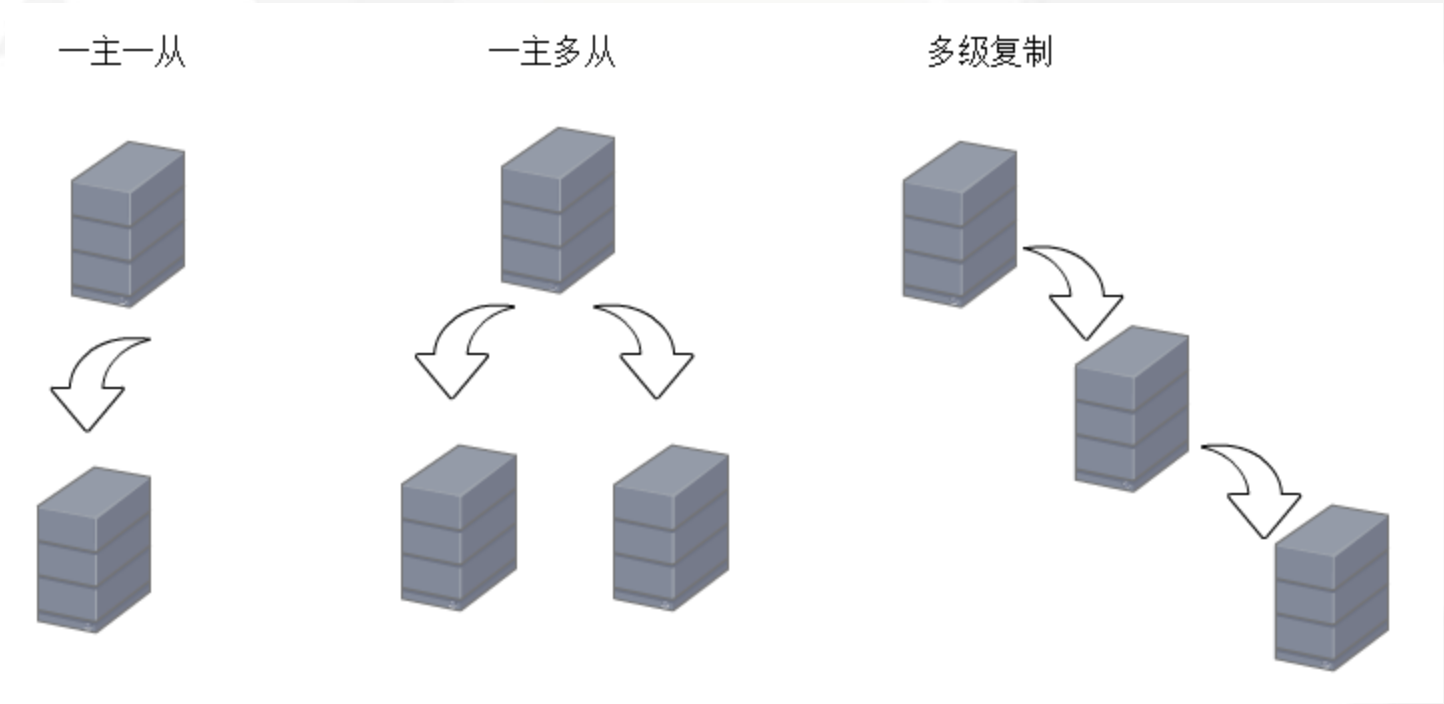
1. 事务1进入队列，事务2进入队列，事务3进入队列。
2. 事务1作为leader获取binlog锁，同时事务2和事务3作为follower进入等待状态。
3. leader（事务1）将队列中所有事务（事务1、2、3）写入到binlog。
4. leader（事务1）调用一次fsync将对binlog的修改进行持久化。
5. leader（事务1）唤醒所有等待的follower（事务2、3）。

传统写binlog流程与binlog group commit流程对比

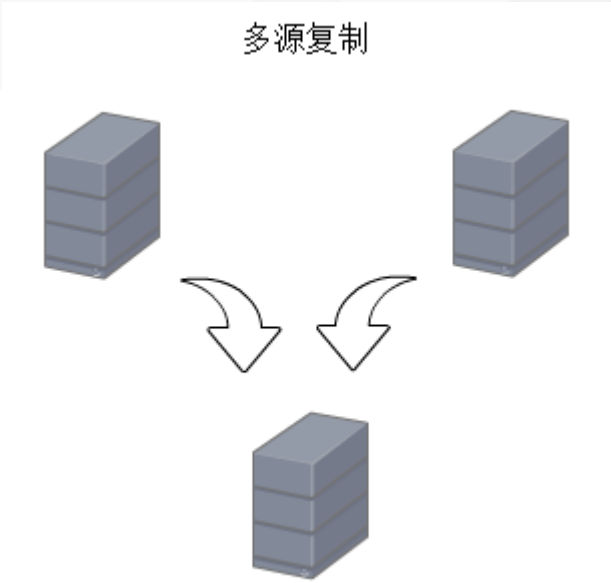
- binlog group commit技术只有在有足够多并发的需要提交的事务时，fsync操作成为事务的提交瓶颈的情况下才能带来性能的提升。

5、MariaDB多源复制

- MySQL传统复制方式

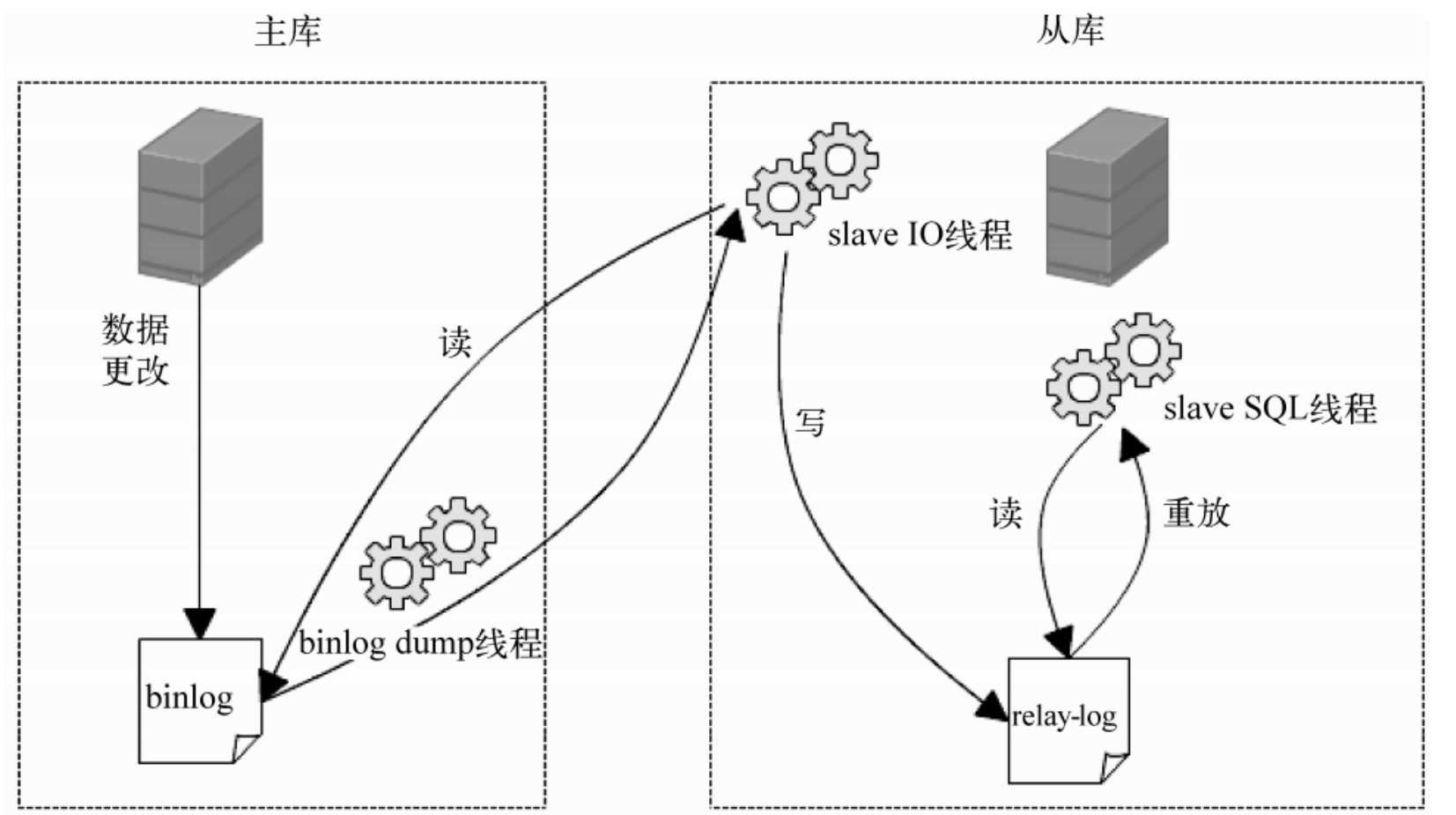


- MariaDB多源复制



- 多源复制的作用：
  1. 由于某些原因，数据被分片到多个数据库实例上，你想把这些数据聚集到一块进行数据分析等工作。
  2. 你有多个数据库实例，想把这些实例的数据使用一台机器进行备份。

- MariaDB多源复制相关的命令：
  1. CHANGE MASTER ["connection-name"] TO …
  2. START SLAVE ["connection-name"]
  3. START ALL SLAVE
  4. STOP SLAVE ["connection-name"]
  5. STOP ALL SLAVE
  6. RESET SLAVE ["connection-name"]
  7. SHOW RELAYLOG ["connection-name"] EVENTS
  8. SHOW SLAVE ["connection-name"] STATUS
  9. SHOW ALL SLAVE STATUS


主库
从库
数据
更改
读
slave IO线程
写
slave SQL线程
读
重放
binlog dump线程
binlog
relay-log

普通复制：

1. 在主库上会开启一个dump线程，在从库上开启了一个IO线程，一个SQL线程，。
2. 从库上有一个master.info文件，存储了连接到主库所需的信息以及slave IO线程获取主库binlog的进度。
3. 从库上有一组relay-log，有一个relay_log.info文件，记录了slave SQL线程重放的进度。

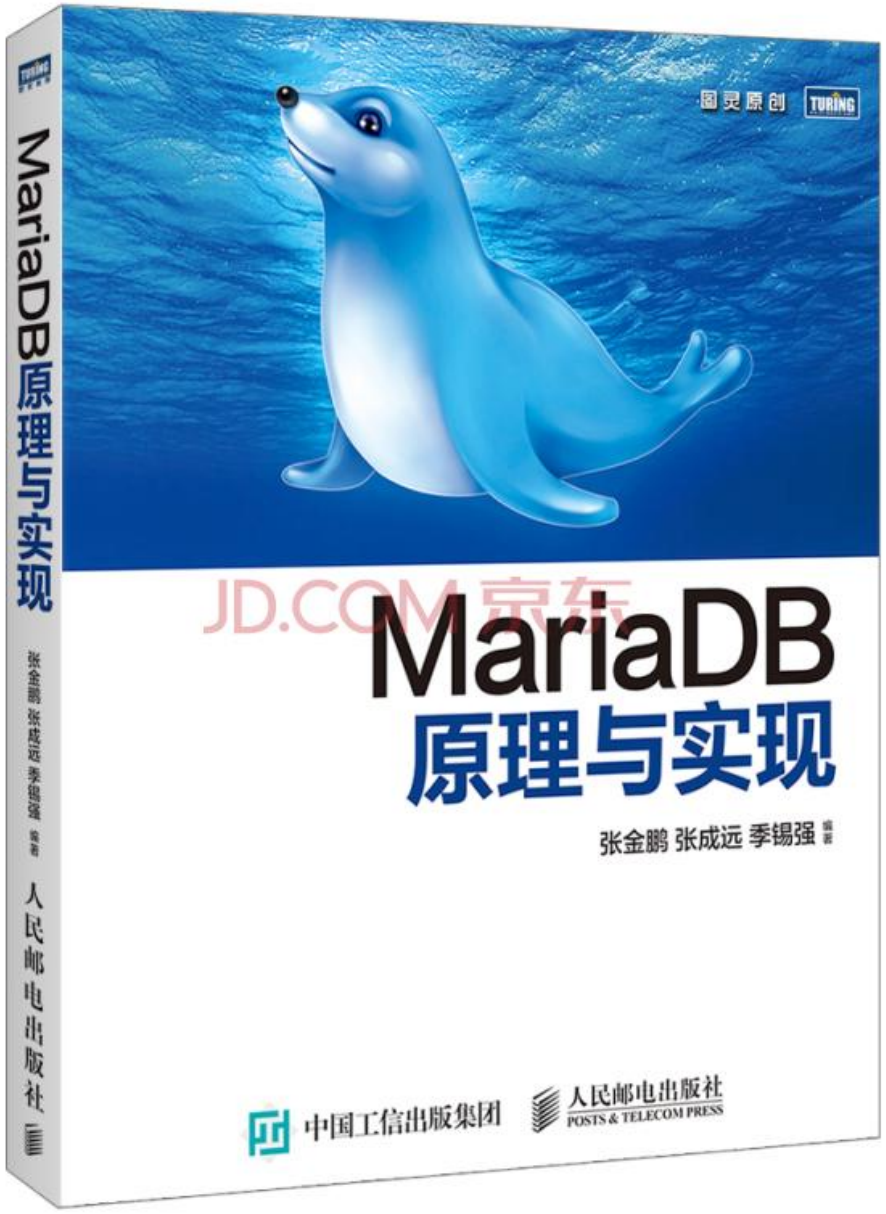
MariaDB多源复制

1. 每增加一个主库，从库上会创建一个IO线程和一个SQL线程。
2. 从库上有多个master-[connection].info文件，存储了连接到对应主库所需的信息以及slave IO线程获取主库binlog的进度。一个multi-master.info文件，存储了所有连接名。
3. 从库上有N组relay-log，有多个relay_log-[connection].info文件，记录了对应slave SQL线程重放的进度。

6、MariaDB面临的挑战

1. 虽然MariaDB拥有众多的特性，但很多特性的使用场景比较有限，例如多源复制。
2. 同时，MySQL也在不断发展中，不断吸收一些好的特性。例如MySQL从5.6开始也引入了binlog group commit技术，支持并发复制等等。
3. 虽然MariaDB有自己的Aria存储引擎，相比MyISAM性能更好，且具有崩溃恢复功能能。但目前InnoDB存储引擎才是默认的标配，MariaDB想要占据重要位置，必须推出自己重量级的存储引擎。
4. MariaDB与MySQL形成了相互竞争的状态，这对双方都有一定的促进作用。最终受益的还是数据库使用者。

More?


图灵原创
TURING
MariaDB原理与实现
张金鹏 张成远 季锡强 编著
人民邮电出版社
JD.COM 京东
MariaDB
原理与实现
张金鹏 张成远 季锡强 编著
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

《MariaDB原理与实现》简介

1. 剖析MariaDB的binlog group commit技术、线程池技术的实现。
2. 教你阅读MySQL和MariaDB的源代码。
3. binlog和复制相关内容：讲解复制、半同步的原理和实现，GTID实现等等。
4. 剖析ORDER BY和JOIN的实现。
5. 讲解京东分布式数据库架构。
6. 容器核心技术Cgroup机制的剖析。

新浪微博 @弓长金鹏

京东云招聘

- SDN/网络工程师
- Golang工程师
- 容器工程师
- 存储工程师
- 云数据库工程师

zhangjinpeng1@jd.com


ChinaUnix
ITPUB
THANKS