# 概述

参考：<https://zhuanlan.zhihu.com/p/47418626/>

Spider是为MySQL/MariaDB开发的一个特殊引擎，具有内嵌分片功能。现在它已经被集成到MariaDB10.0及以上版本中，作为MariaDB的一个新的主要特性。Spider的主要功能是将数据分散到多个后端节点，它的作用类似于一个代理。

## 功能特性

## 特点

### 优点

### 缺点

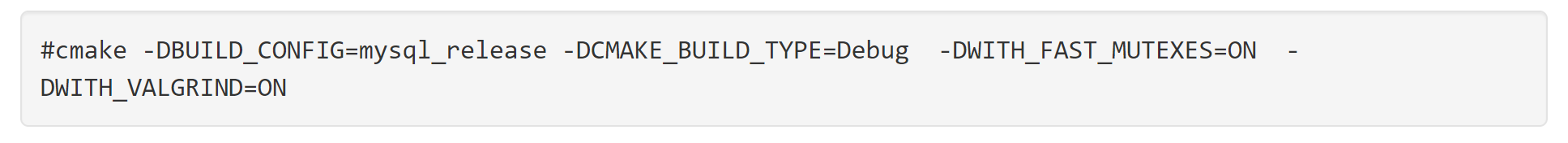
## 区别

# 编译安装

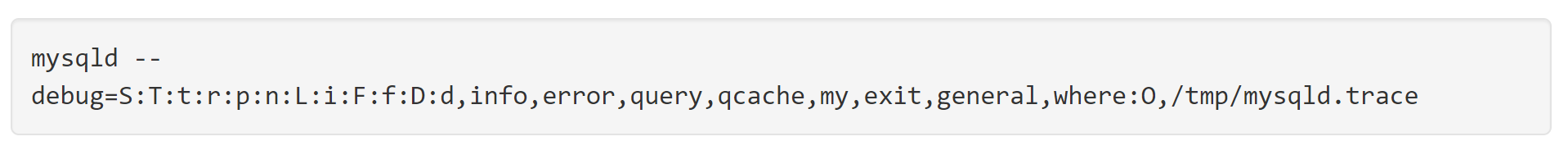
## 编译

### Debug模式编译

Debug模式编译MariaDB：



通过以下方式运行 MariaDB 可以得到详细的命令跟踪文件：



或者通过加载valgrind获取crash的完成堆栈：



### 静态编译

## 安装部署

Spider存储引擎支持分区和XA事务，并允许不同MariaDB实例的表像在同一个实例上一样被处理。

要使用Spider，您需要两个或多个MariaDB实例，通常在不同的主机上运行。Spider节点是从您的应用程序接收查询的MariaDB服务器。然后它处理这些查询，连接到一个或多个数据节点。数据节点是实际存储表数据的MariaDB服务器。

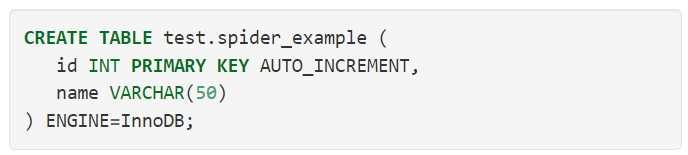
为了使其工作，您需要配置数据节点以接受来自Spider节点的查询，并且您需要将Spider节点配置为使用数据节点作为远程存储。

你不需要安装任何额外的包来使用它，但它确实需要一些配置。

### 配置数据节点

Spider部署使用数据节点来存储实际的表数据。为了让MariaDB服务器作为Spider的数据节点运行，需要创建一个或多个表来存储数据，并将服务器配置为接受来自Spider节点的客户端连接。

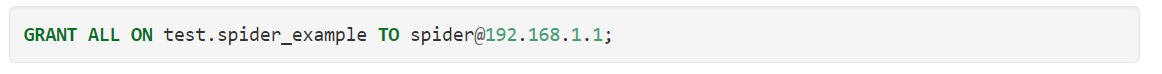
例如，首先创建表：



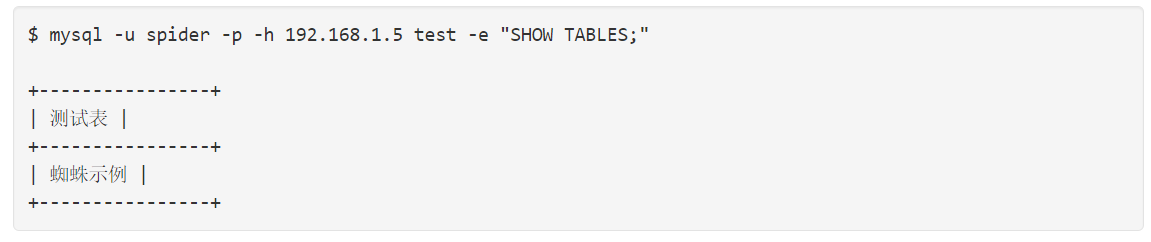
接下来，为Spider节点创建一个用户并为该用户设置一个密码。假设Spider节点的IP地址为192.168.1.1：



然后授予spider用户对示例表的权限。



数据节点现在可以使用了。可以通过尝试将MariaDB客户端连接到来自Spider节点的数据来测试它。例如，假设数据节点的IP地址为192.168.1.5，SSH进入Spider节点然后尝试建立客户端连接。



### 在Spider节点安装Spider

Spider存储引擎必须安装在Spider节点上。Spider节点是接收表查询的MariaDB服务器（在本例中为test.spider\_example）。然后它使用Spider存储引擎连接到数据节点上的表以检索数据并返回结果集。

要安装Spider存储引擎，请完成如下所示的安装过程。

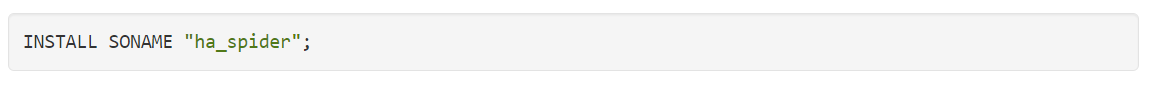
1、安装Spider软件包 (Debian/Ubuntu)

在Debian和Ubuntu上，Spider存储引擎是通过一个单独的mariadb-plugin-spider包安装的。要通过APT安装软件包，请执行以下命令：

2.1）、加载Spider插件（MariaDB 10.4及更高版本）

在MariaDB 10.4及更高版本中，Spider存储引擎可以作为普通插件加载，Spider会自动创建其依赖项。加载插件有两种主要方式。

插件可以动态加载而无需通过执行INSTALL SONAME或重新启动服务器INSTALL PLUGIN：

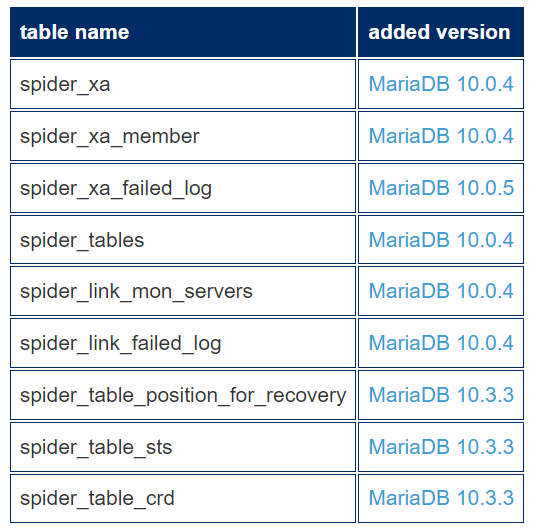


或者，可以通过添加plugin\_load\_add=ha\_spider到配置文件来加载插件：



如果插件加载在配置文件中，则服务器将在服务器重新启动后加载插件。

加载插件还会在数据库中创建一系列新表mysql，包括：

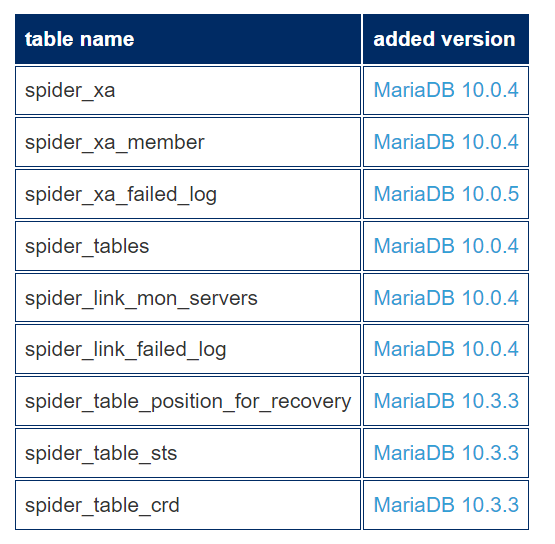


2.2）加载Spider插件（MariaDB 10.3及之前版本）

对于MariaDB 10.3及之前的版本，可以通过执行包含的脚本来加载Spider存储引擎install\_spider.sql：

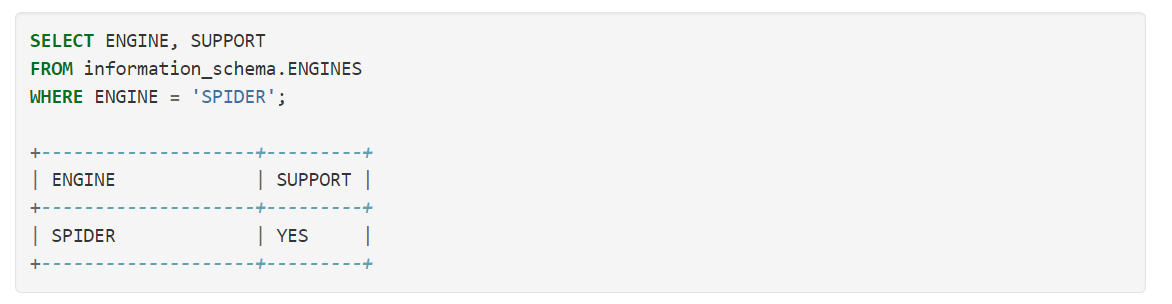
$ mysql --user root --password < /usr/share/mysql/install\_spider.sql

运行此配置脚本还会在数据库中创建一系列新表mysql，包括：



3、验证Spider插件的加载

可以通过查询表来验证是否已加载Spider插件information\_schema.ENGINES：



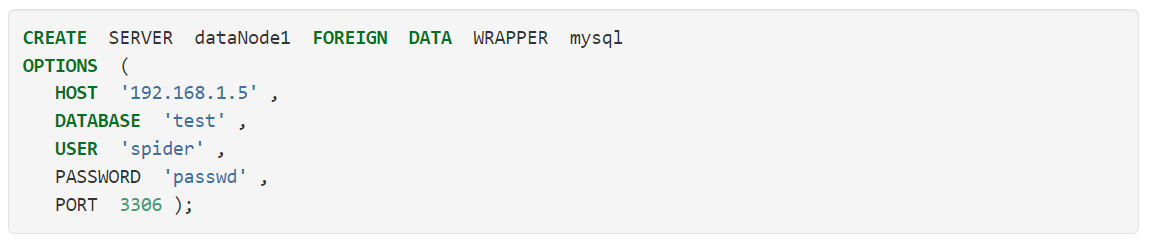
如果没有加载Spider插件，那么查询将不会返回任何结果。

### 配置Spider节点

配置数据节点或数据节点后，可以设置Spider节点以使用它们。Spider节点是接收表查询的MariaDB服务器（在本例中为test.spider\_example）。然后它使用Spider存储引擎连接到数据节点上的表以检索数据并返回结果集。

#### 配置服务器

为了将Spider节点连接到数据节点，需要CREATE SERVER为每个数据节点发出一条语句。然后可以使用服务器定义来创建Spider表。

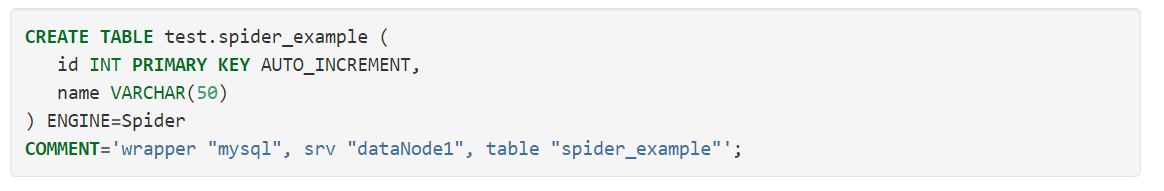


如果在设置Spider表后需要修改或替换此服务器，请记住发出FLUSH更新服务器定义的语句。

FLUSH TABLES;

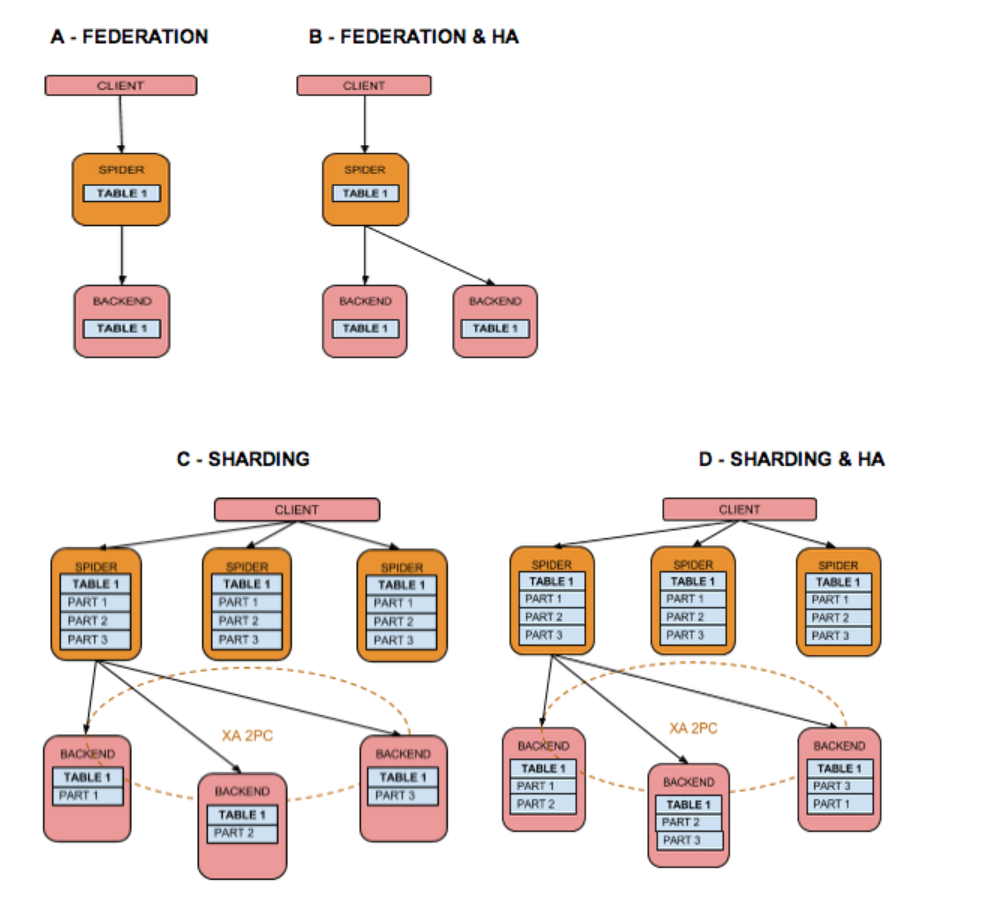
#### 创建表

设置数据节点并配置Spider节点以供使用后，您可以创建Spider表。Spider表必须与数据节点上的InnoDB表具有相同的列定义。Spider是通过传递给选项的表系统变量来配置的COMMENT。



这将Spider配置为使用server dataNode1，（上面定义的）作为远程表。写入此表的任何数据实际上存储在192.168.1.5的MariaDB服务器上。

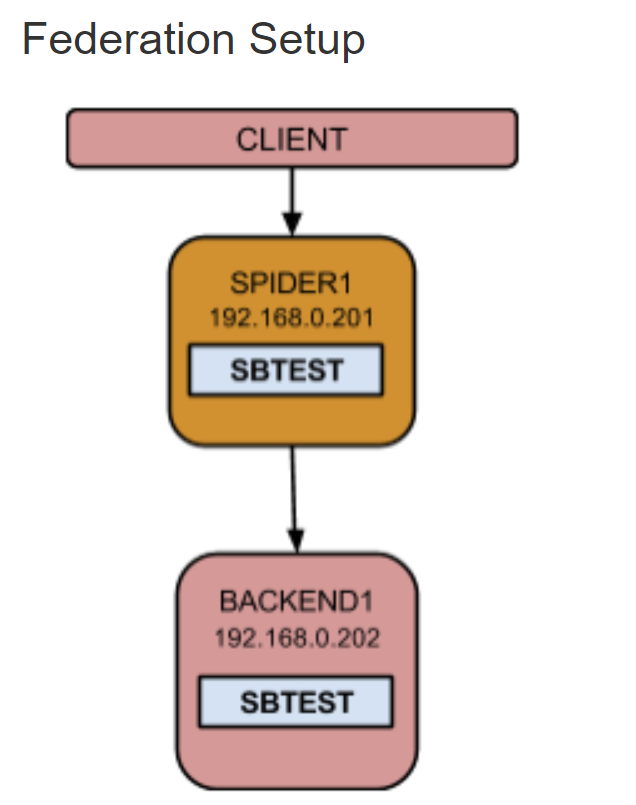
## 部署方式

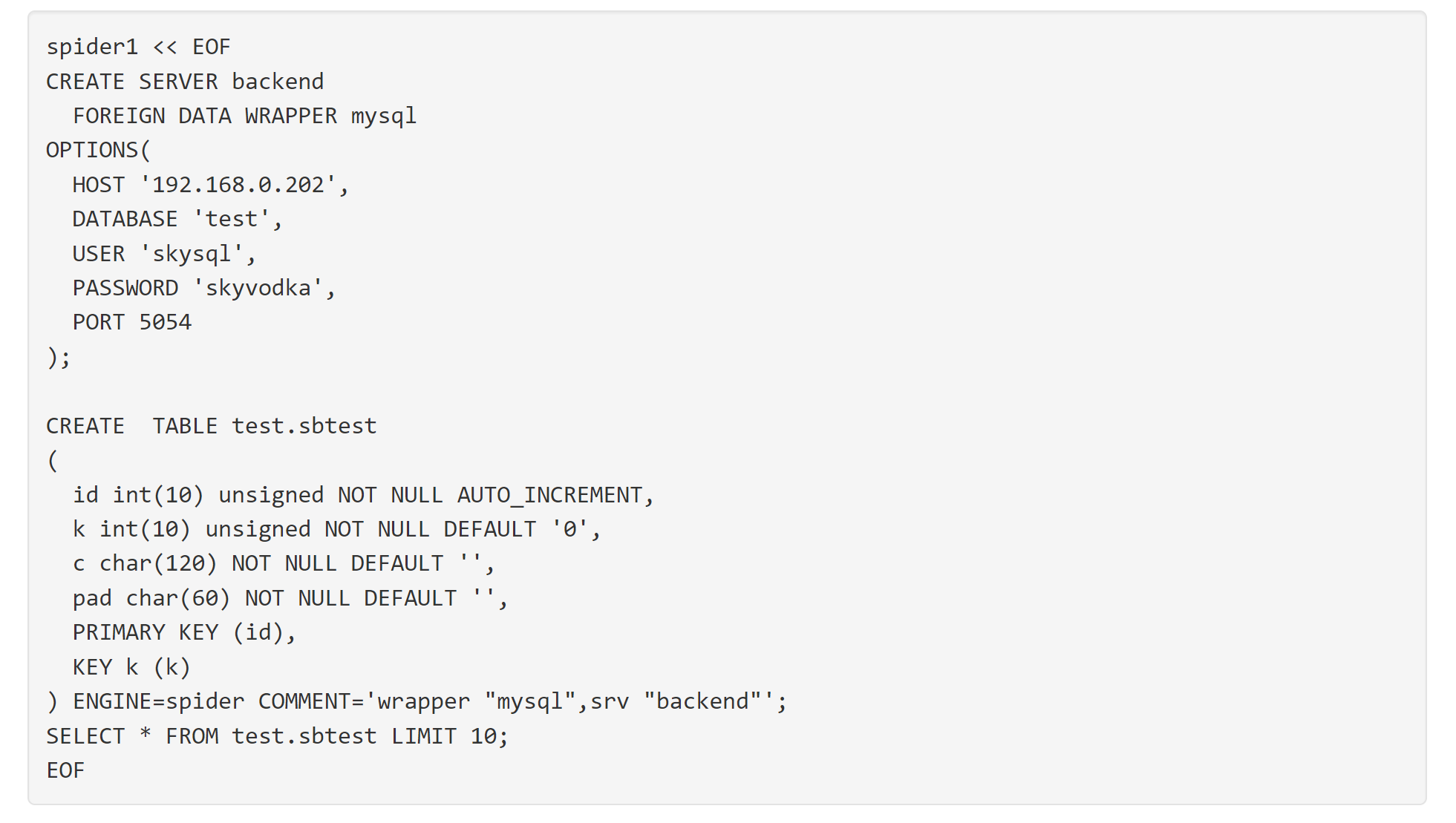


在默认的高可用性设置中，当后端服务器没有响应时，#Spider Nodes#会产生SQL错误。可以设置每表监控以在后端无响应monotoring\_bg\_kind=1或monotoring\_bg\_kind=2。监控Spider节点将与系统表的使用互连mysql.link\_mon\_servers以管理网络分区。

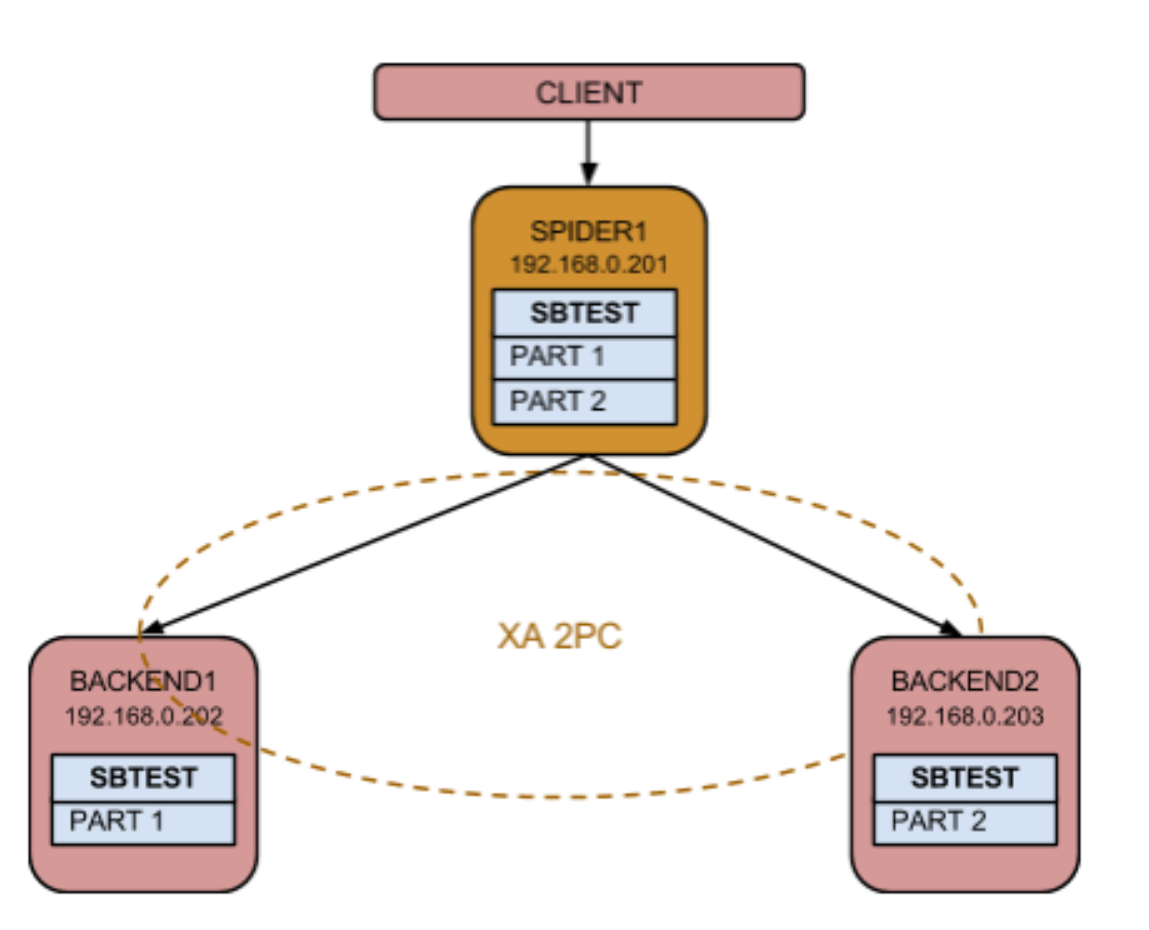
由于存在Spider Monitor Nodes裂脑，应该设置偶数以允许基于多数的共识。而是一个单独的共享Monitoring Node实例或最少3个实例Spider Nodes。

### Federation Setup





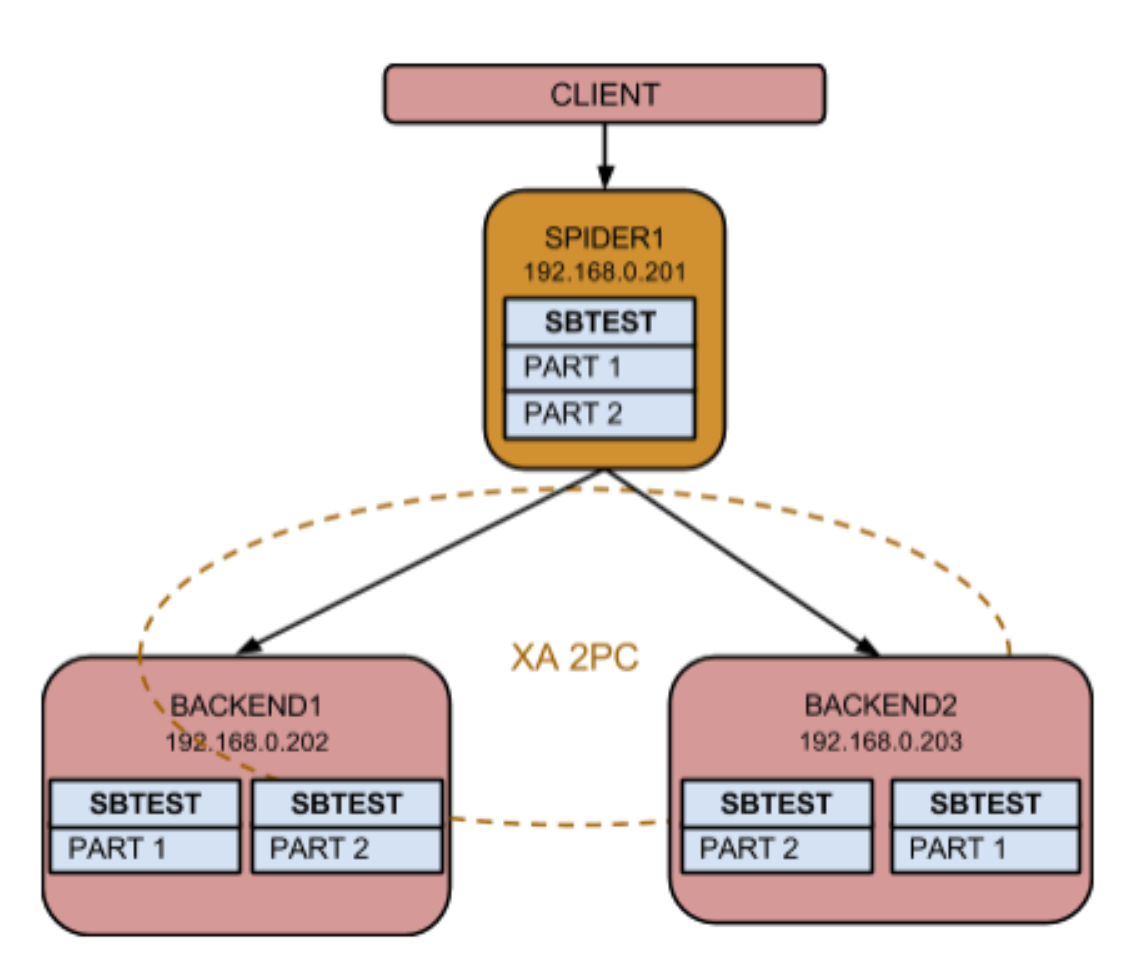
### Sharding Setup





### Background Setup

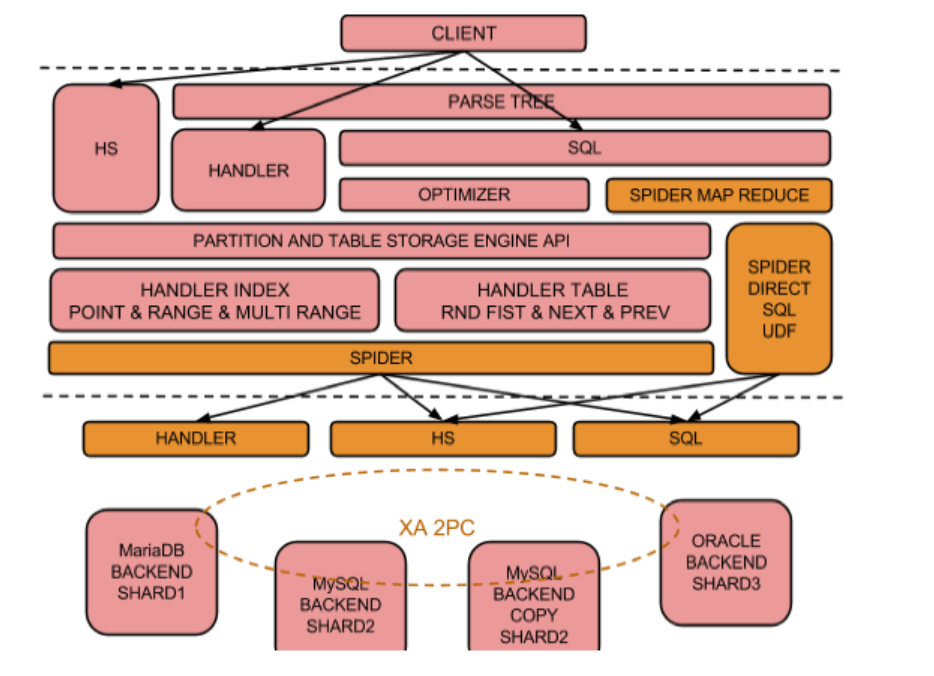
### High Availability Setup



# 架构

## Spider存储引擎联邦

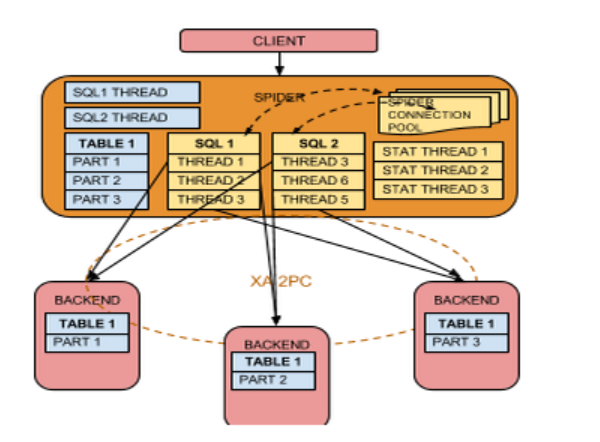
Spider是一个可插入的存储引擎，充当优化器和远程后端之间的代理。当优化器请求多次调用存储引擎时，Spider会使用2阶段提交协议向后端强制执行一致性，并通过在后端创建事务来为单个SQL执行保留原子操作。在执行期间保留原子操作用于体系结构的多个级别。对于常规optimize plan，它指的multiple split reads是并发分区扫描，它指的是semi transactions。



当可以在每个后端完全下推部分执行计划并在之后减少结果时，成本高昂的查询会更有效率。Spider使用一些直接执行快捷方式来启用此类执行。

## Spider线程模型

Spider使用per partitions和per table模型来并发访问远程后端节点。对于内存工作负载，该属性可用于在单个远程后端节点上定义多个分区，以更好地使并发适应硬件中的可用CPU。



Spider维护一个基于分离线程的表和索引统计的内部字典。默认情况下，统计信息是按时间线提取的，并参考crd基数和sts表状态。

## Spider内存模型

Spider将结果集存储到内存中，但如果结果集大于quick\_table\_size，则spider\_quick\_mode =3将结果集存储到内部临时表中。

# 原理

## 表连接

## 事务

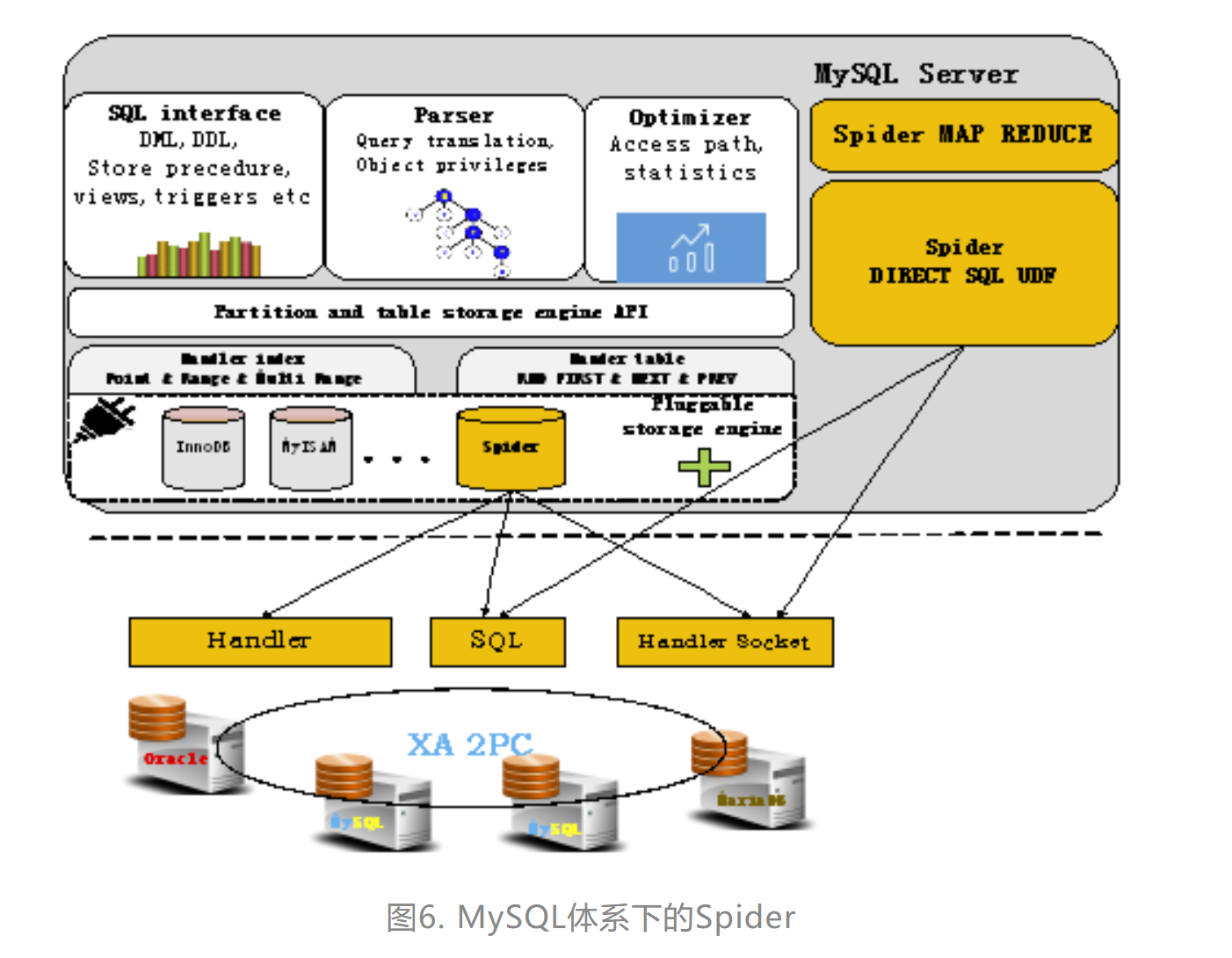
## 插拔式引擎

MySQL最强大的功能之一，以及区别于其它关系型数据库系统的一个主要的特色是不同的表能够采用不同的存储引擎。每一个存储引擎都有其优缺点，用户能够根据自己的需要定制MySQL的存储引擎。存储引擎能够控制在哪里以及如何存放、获取数据。它代表了下面物理层提供的抽象逻辑接口，也是数据库执行实际I/O操作的地方。这是一个组件体系结构。在这个结构中，handler类定义了存储引擎提供的接口和功能。因为所有的存储引擎从基类handler继承而来，所以它们能够提供相同的功能。

总的来说，handler类和handlerton结构在整个体系结构中扮演了中间层的角色。你所编写的存储引擎只有满足了handler的要求后，才能顺利插入到运行的MySQL服务器中。所有的网络连接、安全认证、解析和优化由MySQL服务器本身完成，与存储引擎无关。

Spider作为MySQL的一个可插拔引擎，实现了handler类定义的相应的存取方法。Spider本身并不存放数据，而是类似一个代理的功能将访问请求路由到后端的数据节点。Spider提供了两种途径访问后端节点存储的数据。如图6所示，Spider可以遵循MySQL传统的查询处理流程来访问数据，也开发了自有的一套来加速数据访问。在传统的查询处理方式下，SQL查询请求经过查询解析、查询重写、查询优化等步骤。按照生成的查询执行计划，Spider从后端节点拉取数据，交给MySQL服务器处理。Spider在这种查询处理框架之下的一个缺点是不能很好地利用后端节点可并行化特性，同时需要对SQL查询进行两次解析，带来的性能损耗问题比较严重。

在我们的测试中，性能损耗约50%左右。基于这个原因，为了加速聚集、统计等查询，Spider开发团队提供了DirectSQL方式执行查询。DirectSQL的原理类似于Map Reduce方案，将查询直接下发到后端节点，无需在MySQL服务器层进行解析（Map阶段）；后端节点将结果返回给Spider，由Spider合并结果集（Reduce阶段）。这个方式很好地利用后端节点可并行处理查询的特点，消除重复解析SQL语句的行为。



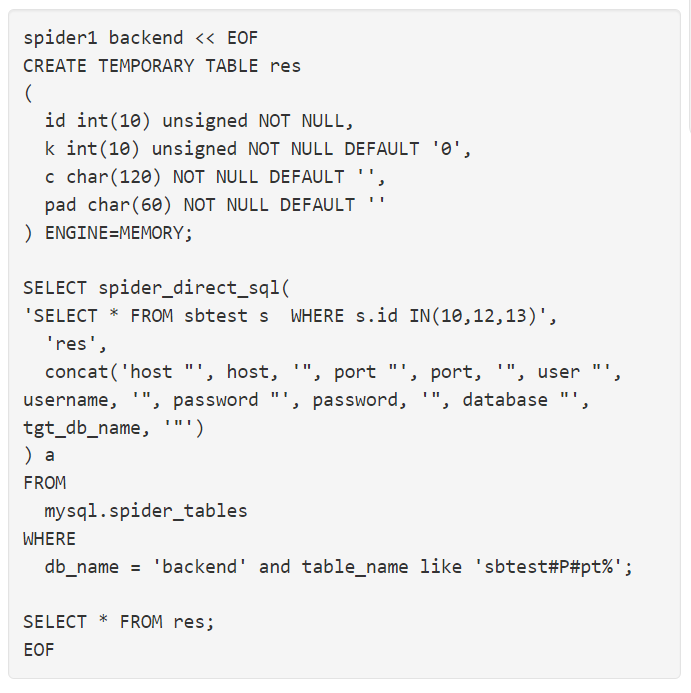
上面已经谈到，Spider本身并不存储数据，因此需要将数据访问请求转换成其它方式，例如Handler、Handler Socket以及SQL方式。前面两种访问方式更像是一种NoSQL的数据访问方式，允许查询绕过SQL layer层。Spider允许后端的数据节点可以是不同的数据库系统，通过2PC保证事务提交的原子性。

## 读写流程

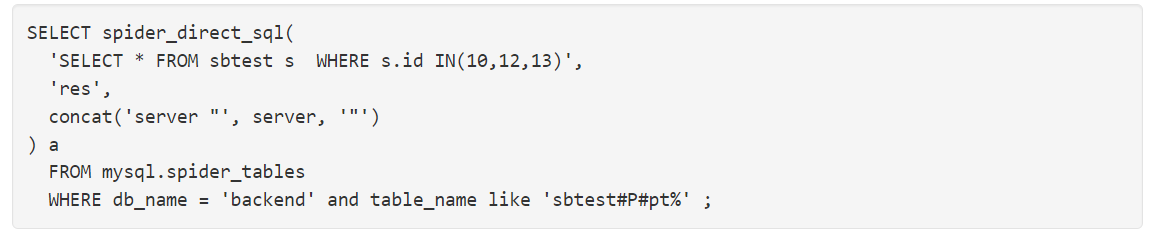
# 集群管理

## Direct SQL

Direct SQL是一种将减少的执行映射到远程后端并将结果存储在本地表中的方法。这可以是顺序的，使用UDF函数[spider\_direct\_sql](https://mariadb.com/kb/en/spider_direct_sql/)，或者同时使用[spider\_bg\_direct\_sql](https://mariadb.com/kb/en/spider_bg_direct_sql/)。



或者，如果您使用的是SERVER：



spider\_bg\_direct\_sql的默认设置是同时访问所有后端。如果您有多个分区存储在单个后端中，您仍然可以增加影响每个分区的不同通道的并行性。

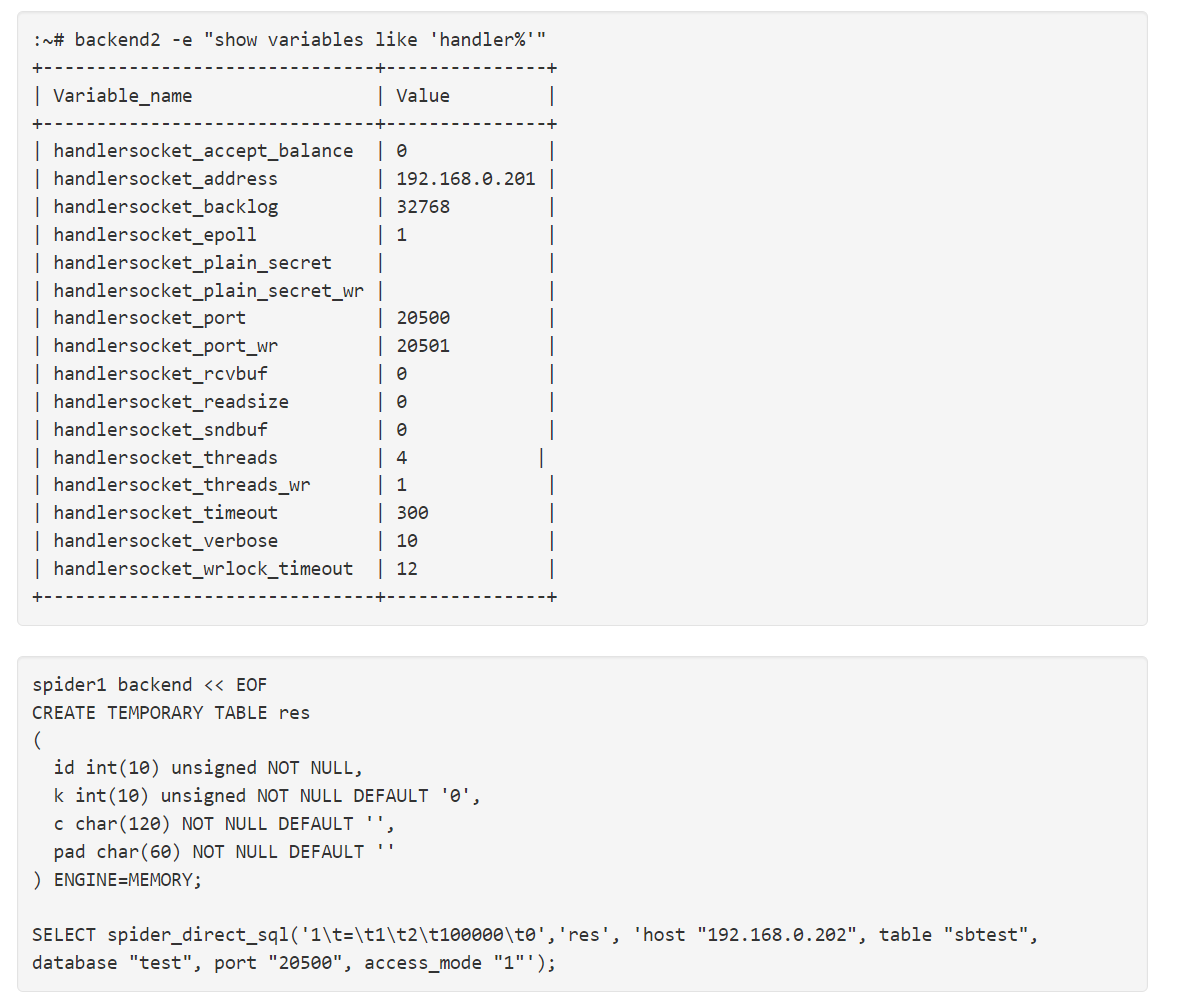


### spider\_direct\_sql

### spider\_bg\_direct\_sql

## Direct Handler Socket

检查Handler Socket是否在后端节点上运行：



### Handler

Handler声明提供一种直接访问读取存储引擎行的方法。这样比传统的SELECT这种SQL方式要快，因为这样会减少parse，且没有optimizer。

### HandlerSocket

HandlerSocket提供一种直接访问InnoDB和Spider的方法。

HandlerSocket是MariaDB的一种NoSQL插件。它在mysql进程内，接收客户端TCP连接，并且执行请求。HandlerSocket并不支持SQL请求。它支持简单的表CRUD操作。

HandlerSocket要比mysqld/libmysql要快，因为它占用更少的CPU、磁盘和网络：

1. 不解析SQL，降低CPU
2. 批量处理，进一步减少CPU和磁盘使用；
3. 最后，C/S协议非常契合mysql/libmysql，进一步减少CPU和磁盘利用。

#### HandlerSocket Installation

#### HandlerSocket Configuration Options

##### handlersocket\_accept\_balance

##### handlersocket\_address

##### handlersocket\_backlog

##### handlersocket\_epoll

##### handlersocket\_plain\_secret

##### handlersocket\_plain\_secret\_wr

##### handlersocket\_port

##### handlersocket\_port\_wr

##### handlersocket\_rcvbuf

##### handlersocket\_readsize

##### handlersocket\_sndbuf

##### handlersocket\_threads

##### handlersocket\_threads\_wr

##### handlersocket\_timeout

##### handlersocket\_verbose

##### handlersocket\_wrlock\_timeout

#### HandlerSocket Client Libraries

#### Testing HandlerSocket in a Source Distribution

#### HandlerSocket External Resources

## 节点间复制表

UDF函数spider\_copy\_tables可用于将表数据从源链接ID复制到目标链接ID列表，而无需停止复制服务。

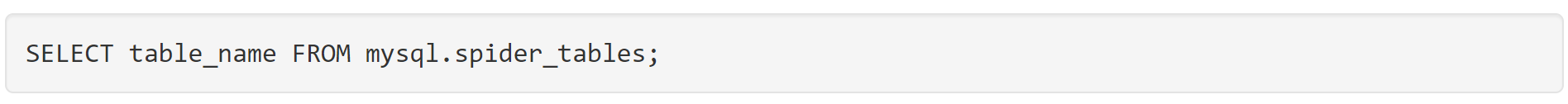
spider\_copy\_tables(Spider table name, source link ID, destination link ID list[, parameters])

returns 1如果复制数据成功。

returns 0如果复制数据失败。

如果Spider表是分区的，则必须为“Spider 表名”设置分区名称，例如“table\_name#P#part\_name”。

您可以使用如下SQL检查表名和带有分区号的link ID：

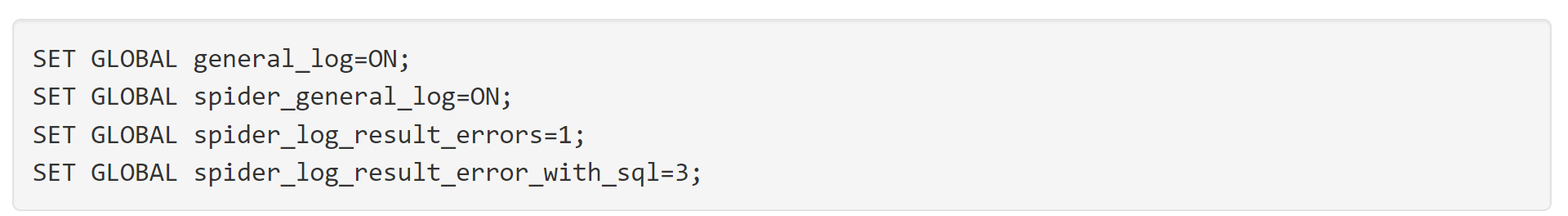


### SPIDER\_COPY\_TABLES

## 重新分片/resharding

## General log

要捕获发送到远程后端的所有查询Spider Node：



# 系统变量

## spider\_auto\_increment\_mode

## spider\_bgs\_first\_read

## spider\_bgs\_mode

## spider\_bgs\_second\_read

## spider\_bka\_engine

## spider\_bka\_mode

## spider\_bka\_table\_name\_type

## spider\_block\_size

## spider\_buffer\_size

## spider\_bulk\_size

## spider\_bulk\_update\_mode

## spider\_bulk\_update\_size

## spider\_casual\_read

## spider\_conn\_recycle\_mode

## spider\_conn\_recycle\_strict

## spider\_conn\_wait\_timeout

## spider\_connect\_error\_interval

## spider\_connect\_mutex

## spider\_connect\_retry\_count

## spider\_connect\_retry\_interval

## spider\_connect\_timeout

## spider\_crd\_bg\_mode

## spider\_crd\_interval

## spider\_crd\_mode

## spider\_crd\_sync

## spider\_crd\_type

## spider\_crd\_weight

## spider\_delete\_all\_rows\_type

## spider\_direct\_dup\_insert

## spider\_direct\_order\_limit

## spider\_dry\_access

## spider\_error\_read\_mode

## spider\_error\_write\_mode

## spider\_first\_read

## spider\_force\_commit

## spider\_general\_log

## spider\_index\_hint\_pushdown

## spider\_init\_sql\_alloc\_size

## spider\_internal\_limit

## spider\_internal\_offset

## spider\_internal\_optimize

## spider\_internal\_optimize\_local

## spider\_internal\_sql\_log\_off

## spider\_internal\_unlock

## spider\_internal\_xa

## spider\_internal\_xa\_id\_type

## spider\_internal\_xa\_snapshot

## spider\_load\_crd\_at\_startup

## spider\_load\_sts\_at\_startup

## spider\_local\_lock\_table

## spider\_lock\_exchange

## spider\_log\_result\_error\_with\_sql

## spider\_log\_result\_errors

## spider\_low\_mem\_read

## spider\_max\_connections

## spider\_max\_order

## spider\_multi\_split\_read

## spider\_net\_read\_timeout

## spider\_net\_write\_timeout

## spider\_ping\_interval\_at\_trx\_start

## spider\_quick\_mode

## spider\_quick\_page\_byte

## spider\_quick\_page\_size

## spider\_read\_only\_mode

## spider\_remote\_access\_charset

## spider\_remote\_autocommit

## spider\_remote\_default\_database

## spider\_remote\_sql\_log\_off

## spider\_remote\_time\_zone

## spider\_remote\_trx\_isolation

## spider\_remote\_wait\_timeout

## spider\_reset\_sql\_alloc

## spider\_same\_server\_link

## spider\_second\_read

## spider\_select\_column\_mode

## spider\_selupd\_lock\_mode

## spider\_semi\_split\_read

## spider\_semi\_split\_read\_limit

## spider\_semi\_table\_lock

## spider\_semi\_table\_lock\_connection

## spider\_semi\_trx

## spider\_semi\_trx\_isolation

## spider\_skip\_default\_condition

## spider\_skip\_parallel\_search

## spider\_slave\_trx\_isolation

## spider\_split\_read

## spider\_store\_last\_crd

## spider\_store\_last\_sts

## spider\_strict\_group\_by

## spider\_sts\_bg\_mode

## spider\_sts\_interval

## spider\_sts\_mode

## spider\_sts\_sync

## spider\_support\_xa

## spider\_sync\_autocommit

## spider\_sync\_sql\_mode

## spider\_sync\_time\_zone

## spider\_sync\_trx\_isolation

## spider\_table\_crd\_thread\_count

## spider\_table\_init\_error\_interval

## spider\_table\_sts\_thread\_count

## spider\_udf\_ct\_bulk\_insert\_interval

## spider\_udf\_ct\_bulk\_insert\_rows

## spider\_udf\_ds\_bulk\_insert\_rows

## spider\_udf\_ds\_table\_loop\_mode

## spider\_udf\_ds\_use\_real\_table

## spider\_udf\_table\_lock\_mutex\_count

## spider\_udf\_table\_mon\_mutex\_count

## spider\_use\_all\_conns\_snapshot

## spider\_use\_cond\_other\_than\_pk\_for\_update

## spider\_use\_consistent\_snapshot

## spider\_use\_default\_database

## spider\_use\_flash\_logs

## spider\_use\_handler

## spider\_use\_pushdown\_udf

## spider\_use\_snapshot\_with\_flush\_tables

## spider\_use\_table\_charset

## spider\_version

## spider\_wait\_timeout

## spider\_xa\_register\_mode

# 表参数

## access\_balances

## active\_link\_count

## casual\_read

## database

## default\_file

## default\_group

## delete\_all\_rows\_type

## host

## idx000

## internal\_delayed

## link\_status

## monitoring\_bg\_interval

## monitoring\_bg\_kind

## monitoring\_kind

## monitoring\_limit

## monitoring\_server\_id

## password

## port

## priority

## query\_cache

## read\_rate

## scan\_rate

## server

## socket

## ssl\_ca

## ssl\_capath

## ssl\_cert

## ssl\_cipher

## ssl\_key

## ssl\_verify\_server\_cert

## table

# 状态变量

## Spider\_direct\_aggregate

## Spider\_direct\_delete

## Spider\_direct\_order\_limit

## Spider\_direct\_update

## Spider\_mon\_table\_cache\_version

## Spider\_mon\_table\_cache\_version\_req

## Spider\_parallel\_search

# 函数

## SPIDER\_BG\_DIRECT\_SQL

## SPIDER\_COPY\_TABLES

## SPIDER\_DIRECT\_SQL

## SPIDER\_FLUSH\_TABLE\_MON\_CACHE

# 系统表

## mysql

### mysql.spider\_link\_failed\_log

### mysql.spider\_link\_mon\_servers

### mysql.spider\_tables

### mysql.spider\_table\_crd

### mysql.spider\_table\_position\_for\_recovery

### mysql.spider\_table\_sts

### mysql.spider\_xa

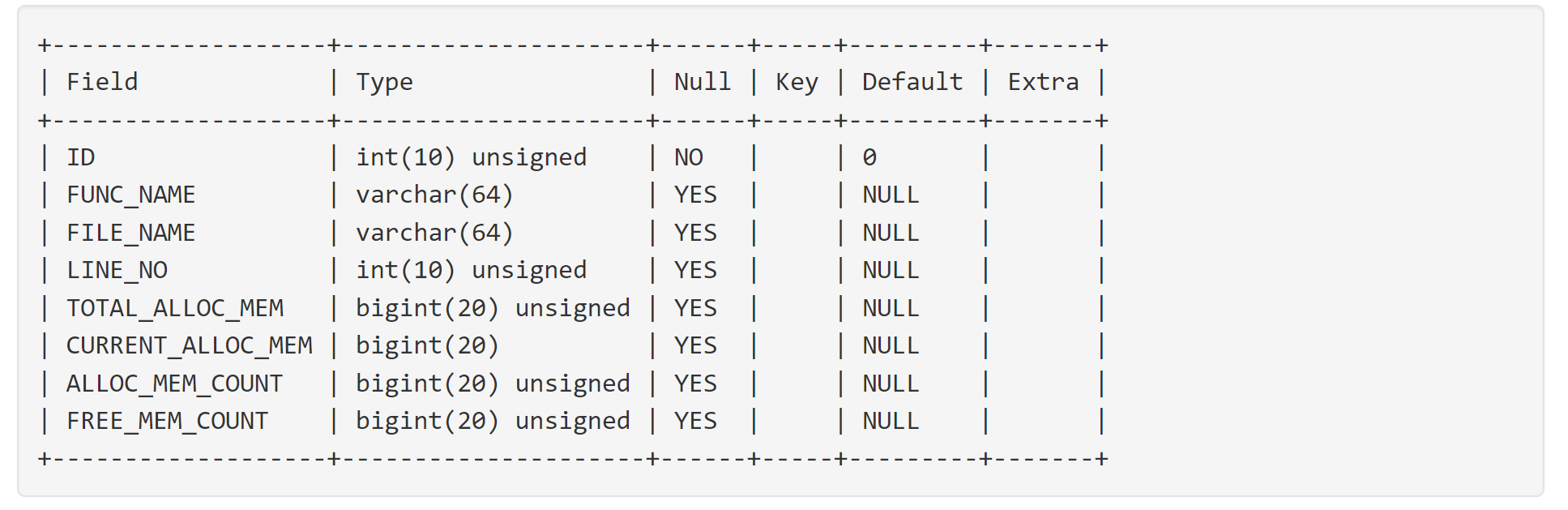
### mysql.spider\_xa\_failed\_log

### mysql.spider\_xa\_member

## information\_schema

### SPIDER\_ALLOC\_MEM

增加一个新的information\_schema的表SPIDER\_ALLOC\_MEM：



### SPIDER\_WRAPPER\_PROTOCOLS

## performance\_schema

# 源码分析

# 应用场景