# ****背景****

对现有的**[数据库](http://lib.csdn.net/base/14" \o "MySQL知识库" \t "/home/zhangxue/Documents\\x/_blank)**连接池做调研对比，综合性能，可靠性，稳定性，扩展性等因素选出推荐出最优的**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "/home/zhangxue/Documents\\x/_blank)**连接池 。

****NOTE: 本文所有**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "/home/zhangxue/Documents\\x/_blank)**均是**[MySQL](http://lib.csdn.net/base/14" \o "MySQL知识库" \t "/home/zhangxue/Documents\\x/_blank)**库****

# 测试结论

   1：性能方面 hikariCP>druid>tomcat-jdbc>dbcp>c3p0 。hikariCP的高性能得益于最大限度的避免锁竞争。

   2：druid功能最为全面，sql拦截等功能，统计数据较为全面，具有良好的扩展性。

   3：综合性能，扩展性等方面，可考虑使用druid或者hikariCP连接池。

   4：可开启prepareStatement缓存，对性能会有大概20%的提升。

# 功能对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能 | dbcp | druid | c3p0 | tomcat-jdbc | HikariCP |
| 是否支持PSCache | 是 | 是 | 是 | 否 | 否 |
| 监控 | jmx | jmx/log/http | jmx,log | jmx | jmx |
| 扩展性 | 弱 | 好 | 弱 | 弱 | 弱 |
| sql拦截及解析 | 无 | 支持 | 无 | 无 | 无 |
| 代码 | 简单 | 中等 | 复杂 | 简单 | 简单 |
| 更新时间 | 2015.8.6 | 2015.10.10 | 2015.12.09 |  | 2015.12.3 |
| 特点 | 依赖于common-pool | 阿里开源，功能全面 | 历史久远，代码逻辑复杂，且不易维护 |  | 优化力度大，功能简单，起源于boneCP |
| 连接池管理 | LinkedBlockingDeque | 数组 |  | FairBlockingQueue | threadlocal+CopyOnWriteArrayList |

* 由于boneCP被hikariCP替代，并且已经不再更新，boneCP没有进行调研。
* proxool网上有评测说在并发较高的情况下会出错，proxool便没有进行调研。
* druid的功能比较全面，且扩展性较好，比较方便对jdbc接口进行监控跟踪等。
* c3p0历史悠久，代码及其复杂，不利于维护。并且存在deadlock的潜在风险。

# ****性能测试****

****环境配置：****

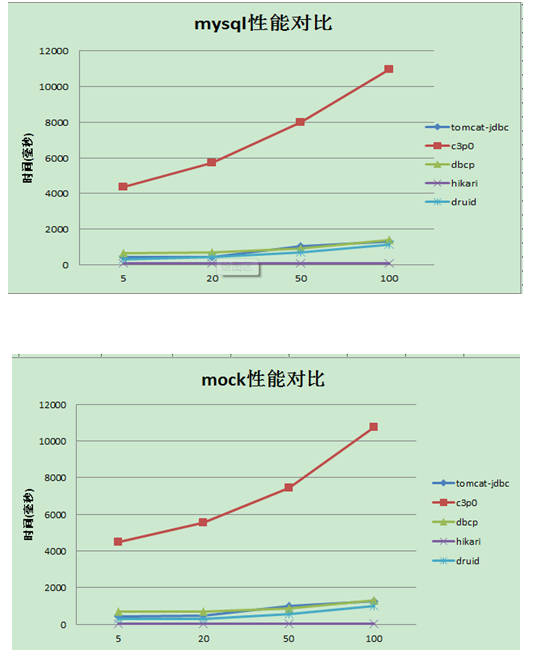
|  |  |
| --- | --- |
| CPU | Intel(R) Xeon(R) CPU E5-2430 v2 @ 2.50GHz，24core |
| msyql version | 5.5.46 |
| tomcat-jdbc version | 8.0.28 |
| HikariCP version | 2.4.3 |
| c3p0 Version | 0.9.5-pre8 |
| dbcpVersion | 2.0.1 |
| druidVersion | 1.0.5 |

****1：获取关闭连接性能测试****

       测试说明：

* 初始连接和最小连接均为5，最大连接为20。在borrow和return均不心跳检测
* 其中打开关闭次数为: 100w次
* 测试用例和mysql在同一台机器上面，尽量避免io的影响
* 使用mock和连接mysql在不同线程并发下的响应时间

     图形：



   mock性能数据 (单位:ms)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 5 | 20 | 50 | 100 |
| tomcat-jdbc | 442 | 447 | 1,013 | 1,264 |
| c3p0 | 4,480 | 5,527 | 7,449 | 10,725 |
| dbcp | 676 | 689 | 867 | 1,292 |
| hikari | 38 | 33 | 38 | 30 |
| druid | 291 | 293 | 562 | 985 |

**[MySQL](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "/home/zhangxue/Documents\\x/_blank)**性能数据 (单位:ms)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 5 | 20 | 50 | 100 |
| tomcat-jdbc | 436 | 453 | 1,033 | 1,291 |
| c3p0 | 4,378 | 5,726 | 7,975 | 10,948 |
| dbcp | 671 | 679 | 897 | 1,380 |
| hikari | 96 | 82 | 87 | 78 |
| druid | 304 | 424 | 690 | 1,130 |

测试结果：

* mock和mysql连接性能表现差不多，主要是由于初始化的时候建立了连接后期不再建立连接，和使用mock连接逻辑一致。
* 性能表现：hikariCP>druid>tomcat-jdbc>dbcp>c3p0。
* hikariCP 的性能及其优异。hikariCP号称java平台最快的数据库连接池。
* hikariCP在并发较高的情况下，性能基本上没有下降。
* c3p0连接池的性能很差，不建议使用该数据库连接池。

   hikariCP性能分析：

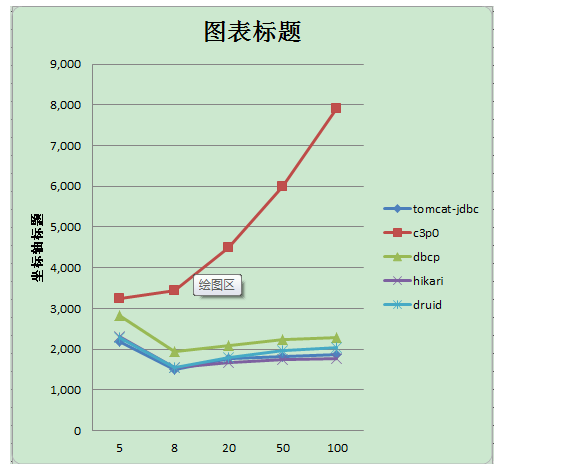
* hikariCP通过优化(concurrentBag，fastStatementList )集合来提高并发的读写效率。
* hikariCP使用threadlocal缓存连接及大量使用CAS的机制，最大限度的避免lock。单可能带来cpu使用率的上升。
* 从字节码的维度优化代码。 (default inline threshold for a JVM running the server Hotspot compiler is 35 bytecodes ）让方法尽量在35个字节码一下，来提升jvm的处理效率。

****2：查询一条语句性能测试****

     测试说明：

* 初始连接和最小连接均为8，最大连接为8。在borrow和return均不心跳检测
* 查询的次数为10w次，查询的语句为 1：打开连接 2：执行 ：select 1 3：关闭连接
* 测试用例和mysql在同一台机器上面，尽量避免io的影响

图形：



 测试数据：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 5 | 8 | 20 | 50 | 100 |
| tomcat-jdbc | 2,178 | 1,495 | 1,769 | 1,818 | 1,858 |
| c3p0 | 3,237 | 3,451 | 4,488 | 5,994 | 7,906 |
| dbcp | 2,816 | 1,935 | 2,097 | 2,243 | 2,280 |
| hikari | 2,299 | 1,546 | 1,682 | 1,751 | 1,772 |
| druid | 2,297 | 1,551 | 1,800 | 1,977 | 2,032 |

测试结果：

* 在并发比较少的情况下，每个连接池的响应时间差不多。是由于并发少，基本上没有资源竞争。
* 在并发较高的情况下，随着并发的升高，hikariCP响应时间基本上没有变动。
* c3p0随着并发的提高，性能急剧下降。

****3：pscache性能对比****

   测试说明：

* 通过druid进行设置pscache和不设置pscache的性能对比
* 初始连接和最小连接均为8，最大连接为8。在borrow和return均不心跳检测。并且执行的并发数为8.
* 查询10w次。查询流程为：1：建立连接，2：循环查询preparestatement语句 3：close连接
* 测试用例和mysql在同一台机器上面，尽量避免io的影响

   测试数据：

|  |  |
| --- | --- |
| cache | 1,927 |
| not cache | 2,134 |

  测试结果：

* 开启psCache缓存,性能大概有20%幅度的提升。可考虑开启pscache.

  测试说明：

* psCache是connection私有的，所以不存在线程竞争的问题，开启pscache不会存在竞争的性能损耗。
* psCache的key为prepare执行的sql和catalog等，value对应的为prepareStatement对象。开启缓存主要是减少了解析sql的开销。

# **[数据库连接池优化配置(druid,dbcp,c3p0)](http://blog.csdn.net/hetaohappy/article/details/51861015" \t "/home/zhangxue/Documents\\x/_blank)**

主要描述了**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "/home/zhangxue/Documents\\x/_blank)**连接池参数配置的准则，针对常用的数据库连接池(c3p0,dbcp,druid)给出推荐的配置。

## **考虑因素**

        1：当前连接DB的规模   2：并发情况 3：执行db的响应时间

# **配置考虑**

       1：初始化连接：可考虑设置为3个连接 。对于db规模特别大的情况下可考虑设置为1个。避免启动时间过长；

       2：最小连接：可考虑该值的设置和初始化连接保持一致；

       3：最大连接：对于有较大DB规模，最大连接不要设置过大，避免本地维护的db太大。 如果对应到数据源的并发数过高，可考虑增大最大连接数。

       4：获取连接的超时时间：如果连接全部被占用，需要等待的时间。可以根据当前系统的响应时间判定，如果容忍度较高，可以大点。容忍度较低，设置小点。

       5：当获取连接和释放连接心跳检测：建议全部关闭，否则每个数据库访问指令会对数据库生产额外的两条心跳检测的指令，增加数据库的负载。连接有效性的检查改用后台空闲连接检查。

       6：连接有效性检测时间：该值需要结合数据库的wait\_timeout,interactive\_timeout值进行设置。假如数据库为120s，则心跳检测时间在120s以内越大越好。如果太小，心跳检测时间会比较频繁。建议设置为90s。

       7：最大空闲时间：如果连接超过该时间没有使用过，则会进行close掉。 该值不要太小，避免频繁的建立连接关闭连接。也不要太大，导致一直无法关闭。

       8：心跳检查的sql语句：尽量使用ping命令，ping的性能较查询语句高。大部分的数据库连接池不配置query语句，便会调用ping命令。

       9：prepareStatement缓存：可以根据自己的业务来判定是否开启。开启后对性能的影响依赖于具体业务和并发情况。可考虑暂时不开启。

       10：连接使用超时：业务拿到一个连接，如果超过指定的时间未归还，是否把该连接给给回收掉。超时时间等和具体的业务关联。暂时建议先不开启。

下面主要给出：druid,dbcp,c3p0 三种连接池的推荐配置

# **druid配置**

 介绍：[https://github.com/alibaba/druid](https://github.com/alibaba/druid" \t "/home/zhangxue/Documents\\x/_blank)

 推荐配置：

|  |  |  |
| --- | --- | --- |
| initialSize | 3 | 初始化配置 |
| minIdle | 3 | 最小连接数 |
| maxActive | 15 | 最大连接数 |
| maxWait | 5000 | 获取连接超时时间（单位：ms） |
| timeBetweenEvictionRunsMillis | 90000 | 连接有效性检测时间(单位:ms) |
| testOnBorrow | false | 获取连接检测 |
| testOnReturn | false | 归还连接检测 |
| minEvictableIdleTimeMillis | 1800000 | 最大空闲时间(单位ms) |
| testWhileIdle | true | 在获取连接后，确定是否要进行连接空间时间的检查 |

配置说明：

           1：minEvictableIdleTimeMillis(最大空闲时间)：默认为30分钟，配置里面不进行设置。

           2：testOnBorrow ,testOnReturn 默认为关闭，可以设置为不配置。

           3：testWhileIdle(在获取连接后，确定是否要进行连接空闲时间的检查)。默认为true。配置里面不再进行设置。

流程说明：

          1：在第一次调用connection的时候，才会进行 initialSize的初始化。

          2：心跳检测时间线程，会休眠timeBetweenEvictionRunsMillis时间，然后只对(没有borrow的线程 减去 minIdle)的线程进行检查，如果空闲时间大于minEvictableIdleTimeMillis则进行close。

          3：testWhileIdle必须设置为true，在获取到连接后，先检查testOnBorrow，然后再判定testwhileIdle，如果连接空闲时间大于timeBetweenEvictionRunsMillis，则会进行心跳检测。

          4：不需要配置validationQuery，如果不配置的情况下会走ping命令，性能更高。

          5：连接保存在数组里面，获取连接的时候，获取数组的最后一位。在timeBetweenEvictionRunsMillis时是从前往后进行检查连接的有效性。

# **dbcp配置**

 介绍：[http://commons.apache.org/proper/commons-dbcp/configuration.html](http://commons.apache.org/proper/commons-dbcp/configuration.html" \t "/home/zhangxue/Documents\\x/_blank)

 推荐配置：

|  |  |  |
| --- | --- | --- |
| 参数 | 配置 | 说明 |
| initialSize | 3 | 初始化配置 |
| minIdle | 3 | 最小连接数 |
| maxIdle | 15 | 最大空闲连接 |
| maxTotal | 15 | 最大连接数 |
| maxWaitMillis | 5000 | 获取连接超时时间(单位ms) |
| timeBetweenEvictionRunsMillis | 90000 | 心跳检测时间(单位ms) |
| minEvictableIdleTimeMillis | 1800000 | 最大空闲时间(单位ms) |
| testOnBorrow | FALSE | 获取连接检测 |
| testOnReturn | FALSE | 归还连接检测 |
| numTestsPerEvictionRun | -1 | 空闲连接检查的个数 |
| testWhileIdle | TRUE | 是否开启对空闲连接的检查 |

 配置说明：

         1：关于maxidle和maxTotal尽量保持一致。

         2：numTestsPerEvictionRun 设置为-1，代表对所有的连接均进行检查。默认值为3。-1代表对全部idle的连接检查有效性。 否则有可能造成部分连接的有效性未进行检查。

         3：testWhileIdle 也必须为true，代表需要检查有效性。

         4：minEvictableIdleTimeMillis默认值为30分钟，可以不用进行设置。

        5：testOnReturn默认值为false，可以不用设置。但是testOnBorrow必须进行设置为false，默认值为true。

        6：validationQuery不配置默认走ping命令

 流程说明：

     1：在第一次调用connection的时候，才会进行 initialSize的初始化。

     2：不需要配置validationQuery，如果不配置的情况下会走ping命令，性能更高。

     3：连接保存在LinkedBlockingDeque 中。来做并发的控制。

     4：后端会有一个定时任务，间隔为timeBetweenEvictionRunsMillis，先确定需要对多少线程进行检测(numTestsPerEvictionRun控制)，然后判定是否超过minEvictableIdleTimeMillis，如果超过则close掉。没有超过，则判定testWhileIdle为true的话，进行心跳检查。如果检查失败则关闭连接。

     5：在return连接的时候会判定maxIdle，如果当前空闲连接是否大于maxIdle，则会关闭掉连接。

# **c3p0配置**

  介绍： [http://www.mchange.com/projects/c3p0/](http://www.mchange.com/projects/c3p0/" \t "/home/zhangxue/Documents\\x/_blank)

  推荐配置：

|  |  |  |
| --- | --- | --- |
| 参数 | 推荐值 | 说明 |
| initialPoolSize | 3 | 初始化配置 |
| minPoolSize | 3 | 最小连接数 |
| maxPoolSize | 15 | 最大连接数 |
| acquireIncrement | 1 | 每次获取的个数 |
| checkoutTimeout | 5000 | 获取连接超时时间(单位ms) |
| idleConnectionTestPeriod | 90 | 心跳检测时间(单位 s) |
| maxIdleTime | 1800 | 最大空闲时间(单位 s) |
| testConnectionOnCheckout | FALSE | 获取连接检测 |
| testConnectionOnCheckin | FALSE | 归还连接检测 |
| numHelperThreads | 1 |  |

 配置说明：

         1：testConnectionOnCheckout和testConnectionOnCheckin默认为false，可不用配置

        2：preferredTestQuery不用配置，默认走ping命令。

        3：numHelperThreads 默认是开启3个线程。如果数据源较多，这里会存在较多线程。 这里设置为1，避免线程较多的情况。

  流程说明：

      1：在第一次调用connection的时候，才会进行 initialPoolSize的初始化。

      2：在进行心跳检测的时候，会对所有的空闲连接进行心跳检测。如果发现总连接小于最小连接数，则会创建连接，保持最小的连接数。