# RestSql性能测试文档

测试内容：  
该部分采用Apache Jmeter 对该RestSQL 系统进行系统性能测试，从而发现和分析该系统软件存在的性能缺陷，具体测试内容如下：  
系统负载测试，针对于不同的数据源查询（其中可分为Postgre数据源测试，Druid数据源测试），并对每一个数据源在不同的负载条件下（如100-1000并发用户发起restsql查询），测试其返回查询结果的并发性能指标数据（响应时间，吞吐率，通过率等)

实验设备及环境  
PC计算机： CPU i59400H ,16GB内存，500GB磁盘  
操作系统： Windows10  
被测系统： RestSQL  
性能测试工具： Apache Jmeter

测试用例设计及结果

一．Postgre SQL数据源性能测试

1. Postgres 数据源查询测试

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO01 | | |
| 业务名称 | 测试使用正确查询时postgre的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性） | | |
| 场景设计 | | | |
| 测试场景 | 200个并发线程发起查询请求，在20秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | 9ms | <25ms | >10.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO02 | | |
| 业务名称 | 测试使用正确查询时postgre的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性） | | |
| 场景设计 | | | |
| 测试场景 | 400个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | 9ms | <25ms | >125.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO03 | | |
| 业务名称 | 测试使用正确查询时postgre的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性） | | |
| 场景设计 | | | |
| 测试场景 | 700个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <180ms | <564ms | >215.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO04 | | |
| 业务名称 | 测试使用正确查询时postgre的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性） | | |
| 场景设计 | | | |
| 测试场景 | 1000个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <800ms | <1700ms | >169.0/sec |

总结：在使用正确无误的restful对postgre数据源进行查询，该条件下能够支持约500-600之间用户的低延迟并发查询，查询用户数里进一步增加，那么响应时间将大幅度延长。当1000用户查询时，出现高延迟状况，该情况耗时一大部分出现在数据库查询数据传输这一部分上面

（2）使用错误的restsql查询（但能够被restsql服务端识别出错误格式并返回）

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO05 | | |
| 业务名称 | 测试使用错误查询时postgre的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday----","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性,此处在时间查询字段中加入非法注入字符，测试被服务端检查并错误返回的情况） | | |
| 场景设计 | | | |
| 测试场景 | 200个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <4ms | <9ms | >66.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO06 | | |
| 业务名称 | 测试使用错误查询时postgre的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday----","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性,此处在时间查询字段中加入非法注入字符，测试被服务端检查并错误返回的情况） | | |
| 场景设计 | | | |
| 测试场景 | 400个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <4ms | <9ms | >125.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO07 | | |
| 业务名称 | 测试使用错误查询时postgre的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday----","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性,此处在时间查询字段中加入非法注入字符，测试被服务端检查并错误返回的情况） | | |
| 场景设计 | | | |
| 测试场景 | 800个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <20ms | <25ms | >249.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO08 | | |
| 业务名称 | 测试使用错误查询时postgre的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday----","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性,此处在时间查询字段中加入非法注入字符，测试被服务端检查并错误返回的情况） | | |
| 场景设计 | | | |
| 测试场景 | 1000个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <100ms | <150ms | >316.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO09 | | |
| 业务名称 | 测试使用错误查询时postgre的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday----","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性,此处在时间查询字段中加入非法注入字符，测试被服务端检查并错误返回的情况） | | |
| 场景设计 | | | |
| 测试场景 | 1400个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <1000ms | <2000ms | >250.0/sec |

总结：

在该部分下，由于错误查询会被Restsql服务端检测到，并进行错误返回，该过程不涉及到数据库查询，因此该部分相比正确查询时涉及到数据库查询情况，延迟时间降低，在用户数1000以下时，都能保持低延迟时间处理性能。

（3）Postgre 错误查询（在数据库引擎发生错误，并错误返回）

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO010 | | |
| 业务名称 | 测试使用错误查询时postgre的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。此处使用错误字段进行查询，测试数据库引擎发生查询错误并返回的情况） | | |
| 场景设计 | | | |
| 测试场景 | 50个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <1500ms | <2500ms | >10.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | PostgreO010 | | |
| 业务名称 | 测试使用错误查询时postgre的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "pg\_test.people","time": {"begin": "2018-06-01 00:00:00", "end": "2021-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "birthday","timeShift": 0}, "select": [{"column": "age", "alias": "${alias}", "metric": "count"}], "where": [{"column": "grade", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。此处使用错误字段进行查询，测试数据库引擎发生查询错误并返回的情况） | | |
| 场景设计 | | | |
| 测试场景 | 200个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <4500ms | <5000ms | >25.0/sec |

总结： 在该情况下，该系统查询当发生远程数据库错误时，该查询性能效果不佳，延迟高。该解决方案需要进行进一步讨论

二. Druid 性能测试：

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid001 | | |
| 业务名称 | 测试使用正确查询时Druid的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLength", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。） | | |
| 场景设计 | | | |
| 测试场景 | 100个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <200ms | <300ms | >54.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid002 | | |
| 业务名称 | 测试使用错误查询时Druid的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLength", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。） | | |
| 场景设计 | | | |
| 测试场景 | 400个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <2000ms | <3000ms | >60.0/sec |

总结：该情况下，用户在200-300之间，低延迟查询，当用户在400时，受查询时数据库数据传输延迟影响，耗时提高

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid002 | | |
| 业务名称 | 测试使用错误查询时Druid的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLength", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。） | | |
| 场景设计 | | | |
| 测试场景 | 400个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <2000ms | <3000ms | >60.0/sec |

总结：该情况下，用户在200-300之间，低延迟查询，当用户在400时，受查询时数据库数据传输延迟影响，耗时提高

（2）使用错误查询测试Druid模块错误处理性能

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid003 | | |
| 业务名称 | 测试使用错误查询时Druid的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLength----", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。 使用非法字符进行查询，测试服务器处理错误并返回延迟） | | |
| 场景设计 | | | |
| 测试场景 | 400个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <10ms | <15ms | >60.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid004 | | |
| 业务名称 | 测试使用错误查询时Druid的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLength----", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。 使用非法字符进行查询，测试服务器处理错误并返回延迟） | | |
| 场景设计 | | | |
| 测试场景 | 800个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <10ms | <15ms | >200.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid005 | | |
| 业务名称 | 测试使用错误查询时Druid的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLength----", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。 使用非法字符进行查询，测试服务器处理错误并返回延迟） | | |
| 场景设计 | | | |
| 测试场景 | 1200个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <400ms | <900ms | >170.0/sec |

总结： 在该情况下，测试性能如前面Postgre查询模块类似，可以支持1200用户并发量

（3）使用错误查询测试Druid引擎错误返回性能

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid006 | | |
| 业务名称 | 测试使用错误查询时Druid的查询性能(低并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLengthtest", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。修改错误字段，测试该情况下服务器处理性能） | | |
| 场景设计 | | | |
| 测试场景 | 200个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <180ms | <200ms | >62.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid007 | | |
| 业务名称 | 测试使用错误查询时Druid的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLengthtest", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。修改错误字段，测试该情况下服务器处理性能） | | |
| 场景设计 | | | |
| 测试场景 | 500个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <500ms | <600ms | >145.0/sec |

|  |  |  |  |
| --- | --- | --- | --- |
| 用例ID | Druid008 | | |
| 业务名称 | 测试使用错误查询时Druid的查询性能(高并发条件下) | | |
| URL | http://127.0.0.1:8000/query | | |
| 前置条件 | RestSQL系统启动 | | |
| 测试用例 | {"refId": "${refId}", "from": "RestSQL.wikipedia", "time": {"begin": "2016-06-01 00:00:00", "end": "2016-07-01 23:59:59", "interval": "${\_\_Random(1,20,)}${timeden}", "column": "\_\_time"}, "select": [{"column": "commentLengthtest", "alias": "${alias}", "metric": "avg"}], "where": [{"column": "added", "op": ">", "value": "0"}], "group": [], "limit": ${limit}}  （使用jmeter从csv文件自动导入样例数据进行测试，其中${}为待填入的参数，保证每次用例查询的随机性。修改错误字段，测试该情况下服务器处理性能） | | |
| 场景设计 | | | |
| 测试场景 | 800个并发线程发起查询请求，在3秒钟内完成。记录各项指标数据。 | | |
| 测试指标 | 平均响应时间 | 90%响应时间 | 吞吐量 |
|  | <2400ms | <2800ms | >130.0/sec |

总结：Druid引擎当发生查询错误时，不同意postgre使用重启连接对象需要消耗时间，该延迟相对于postgre数据源低。

性能测试总结：

在正常查询时，postgre sql查询延迟低于druid查询，但在数据库查询发生错误时，该部分由于postgre sql需要进行回滚事务或者重启connect对象,影响性能，该情况下，druid在错误处理优于postgre sql。在另一方面，在服务端进行检查错误查询并及时错误返回，该过程不涉及到数据库引擎错误返回然后处理，可以提高查询性能，降低查询延迟时间。