测试环境：

Node(MySQL)

Cpu: Intel(R) Xeon(R) CPU E5620 @ 2.40GHz

Cpu Core: 16

OS-Version: RedHat6.3

Memory: 94.47 GB

Disk: Raid10 (4块1w转磁盘)

Server(MyCat)

Cpu: Intel(R) Xeon(R) CPU E5606 @ 2.13GHz

Cpu Core: 8

OS-Version: RedHat6.3

Memory: 47.12 GB

MyCat Version: 1.3.0.3

MySQL Version: 5.5.31

Tool：

Sysbench

测试目的：

对比单机MySQL与通过MyCat分表后的对比情况，并验证MyCAT比较适合的业务场景。

测试方法：

简单分表测试，16张表分别放在单台MySQL对比两台机器四个实例每个实例4张表的MySQL（使用mycat作为中间件），3600秒的时间分别在16、32、64、128的并发情况下的QPS、TPS的表现。

测试数据如下：

第一组

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **线程数** | **MyCat QPS** | **MyCat TPS** | **MySQL QPS** | **MySQL TPS** |
| **16** | **12625.5** | **4591.09** | **17331.5** | **6302.35** |
| **32** | **12373.6** | **4499.49** | **27328** | **9937.34** |
| **64** | **34076.2** | **12391** | **36449.3** | **13253** |
| **128** | **46335.3** | **16848** | **30180.3** | **10948.5** |

第二组

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **线程数** | **MyCat QPS** | **MyCat TPS** | **MySQL QPS** | **MySQL TPS** |
| **16** | **12610.1** | **4585.5** | **17323.3** | **6299.36** |
| **32** | **12633.6** | **4594.01** | **27310.3** | **9930.91** |
| **64** | **34750.3** | **12636.2** | **36716.2** | **13350.1** |
| **128** | **44386.7** | **16139.4** | **29687.7** | **10769.5** |

结论：

从两组测试中可以发现单机mysql发生资源耗尽的情况，但是通过中间件拆分后的mysql依然可以得到比较高的性能表现。

mycat 在高并发简单sql纯oltp环境下可以发挥很大的作用，来帮助提高mysql的吞吐量。

附录：

表结构

CREATE TABLE `sbtestX` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

`k` int(10) unsigned NOT NULL DEFAULT '0',

`c` char(120) NOT NULL DEFAULT '',

`pad` char(60) NOT NULL DEFAULT '',

PRIMARY KEY (`id`),

KEY `k\_1` (`k`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

Sysbench 测试用Perl

pathtest = string.match(test, "(.\*/)") or ""

dofile(pathtest .. "common.lua")

function thread\_init(thread\_id)

set\_vars()

if (db\_driver == "mysql" and mysql\_table\_engine == "myisam") then

begin\_query = "LOCK TABLES sbtest WRITE"

commit\_query = "UNLOCK TABLES"

--else

--begin\_query = "BEGIN"

--commit\_query = "COMMIT"

end

end

function event(thread\_id)

local rs

local i

local table\_name

local range\_start

local c\_val

local pad\_val

local query

table\_name = "sbtest".. sb\_rand\_uniform(1, oltp\_tables\_count)

if not oltp\_skip\_trx then

db\_query(begin\_query)

end

for i=1, oltp\_point\_selects do

rs = db\_query("SELECT c FROM ".. table\_name .." WHERE id=" .. sb\_rand(1, oltp\_table\_size))

end

for i=1, oltp\_simple\_ranges do

range\_start = sb\_rand(1, oltp\_table\_size)

rs = db\_query("SELECT c FROM ".. table\_name .." WHERE id BETWEEN " .. range\_start .. " AND " .. range\_start .. "+" .. oltp\_range\_size - 1)

end

if not oltp\_read\_only then

for i=1, oltp\_index\_updates do

rs = db\_query("UPDATE " .. table\_name .. " SET k=k+1 WHERE id=" .. sb\_rand(1, oltp\_table\_size))

end

for i=1, oltp\_non\_index\_updates do

c\_val = sb\_rand\_str("###########-###########-###########-###########-###########-###########-###########-###########-###########-###########")

query = "UPDATE " .. table\_name .. " SET c='" .. c\_val .. "' WHERE id=" .. sb\_rand(1, oltp\_table\_size)

rs = db\_query(query)

if rs then

print(query)

end

end

i = sb\_rand(1, oltp\_table\_size)

rs = db\_query("DELETE FROM " .. table\_name .. " WHERE id=" .. i)

c\_val = sb\_rand\_str([[

###########-###########-###########-###########-###########-###########-###########-###########-###########-###########]])

pad\_val = sb\_rand\_str([[

###########-###########-###########-###########-###########]])

rs = db\_query("INSERT ignore INTO " .. table\_name .. " (id, k, c, pad) VALUES " .. string.format("(%d, %d, '%s', '%s')",i, sb\_rand(1, oltp\_table\_size) , c\_val, pad\_val))

end -- oltp\_read\_only

if not oltp\_skip\_trx then

db\_query(commit\_query)

end

end

MyCAT相关配置：

schema.xml 

server.xml



wrapper.conf

