文章编号: 1000-5277(2013)06-0049-06

高并发环境下 MySQL 软硬件配置优化

范新民
(福建师范大学网络与继续教育学院,福建 福州 350007)

摘要: 通过分析生产环境中的测试结果,给出了 MySQL 关键配置的优化策略: 存储引擎使用 InnoDB、表空间配置单表拆分以及写入策略采用非阻塞磁盘缓存机制. 同时,给出了这些配置相对应的服务器硬件需求: 硬盘使用固态 SSD 硬盘,内存使用支持 ECC 校验的企业级内存.

关键词: MySQL; InnoDB; SSD; ECC 校验
中图分类号: TP391 **文献标志码:** A

The Optimization of MySQL Software and Hardware Configurations under the Circumstance of Overload

FAN Xin-min
(*Online and Continuing Education College, Fujian Normal University, Fuzhou* 350007, *China*)

Abstract: Through the analysis of test results under the production environment, the optimal strategies of MySQL's critical configurations have been drawn out as follows: InnoDB is taken as storage engine, and non-blocking disk-cache system is taken for splitting the single table of the configuration of table space and writing access strategy. At the same time, the corresponding server hardware of these critical configurations has been required as follows: take the solid state SSD as rigid disk, and the enterprise-class RAM supporting the verification of ECC as memory.

Key words: MySQL; InnoDB; SSD; the verification of ECC

当今大数据、高并发的环境下,MySQL 仍然作为主流数据存储手段之一,活跃在互联网的舞台上[1]. 但是随着数据存储规模的不断升级,MySQL 已经由传统意义上的 CPU 密集型应用,转型为专注于高速读写的 IO 密集型应用[2]. 对于 MySQL 的性能优化,核心思想是优化它的 IO 性能.

1 MySQL 软件层配置优化

1.1 MySQL 架构简介

MySQL 是一种关系型数据库,其 SQL 语言是目前用于访问数据库的最常用标准化语言. MySQL 具有体积小、速度快、成本低、开源等特点,得到了互联网与软件行业内的广泛关注. MySQL 架构图如图 1 所示. MySQL 架构主要由 SQL 接口、解析器、优化器、缓存、存储引擎这几个部分组成[3],具体为:

(1) Connectors: 管理客户端与 MySQL 服务器的交互.
(2) Management Serveices & Utilities: 对 MySQL 系统的管理和控制.
(3) Connection Pool: 连接池,用于缓冲用户连接、线程处理等.
(4) SQL Interface: SQL 接口,接受来自客户端的 SQL 指令,并且返回查询结果.
(5) Parser: 解析器,SQL 命令被传递到解析器时,由解析器进行验证和解析.

收稿日期: 2013-09-03
通信作者: 范新民(1970-),男,高级工程师,研究方向为软件工程、网络教育技术应用. xmfan@mail.fjtu.com.cn

(6) Cache&Buffer: 查询缓存，如果查询缓已经保存有之前命中的查询结果，新的查询语句就可以直接从查询缓存中取结果，而不用重新查询.

(7) Engine: 存储引擎，存储引擎是 MySQL 的具体与文件系统交互的子系统.

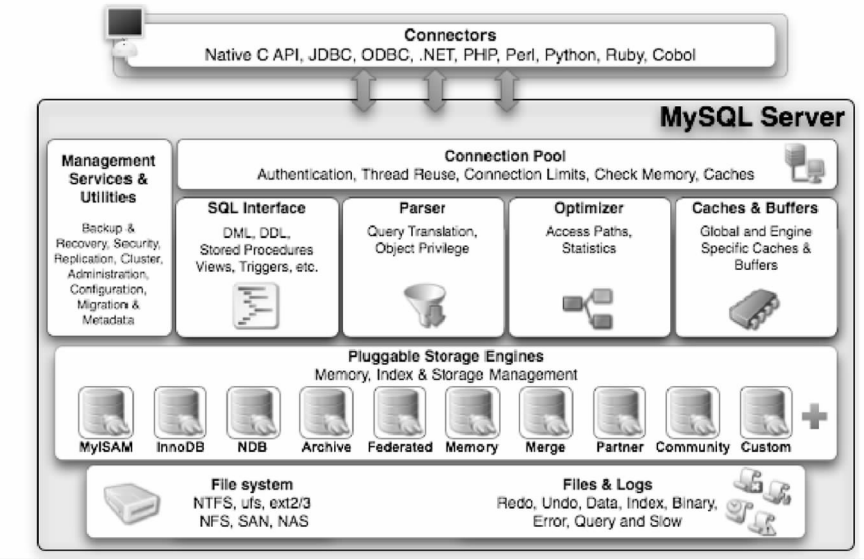

图 1 MySQL 架构

Fig. 1 MySQL Architecture

1.2 存储引擎的选择

MySQL 最常用的的存储引擎有两种，MyISAM 和 InnoDB[3]. 这两种存储引擎各有优劣，主要的差异如下:

(1) MyISAM 类型引擎不支持事务、外键等高级处理，而 InnoDB 类型引擎支持.

(2) MyISAM 类型引擎不支持行级锁，而 InnoDB 类型引擎支持行级锁.

(3) InnoDB 中不保存表的具体行数，每次 select count(*)，都会进行全表扫描，而 MyISAM 保存表的行数，select count(*) 会立刻返回; 当然，如果语句中包含 where 条件时，两者是一样的.

(4) InnoDB 的查询速度要慢于 MyISAM 引擎.

在大数据、高并发的生产环境中，InnoDB 相对于 MyISAM 能够提供更加灵活的锁机制，以及更加稳定与安全的外键与事务支持，表 1 给出了 MyISAM 引擎与 InnoDB 引擎的主要参数对比[4]. 虽然在查询性能上 InnoDB 要弱于 MyISAM，不过考虑到在数据量大的网站或应用中对高可扩展性、高性能存储和高并发的需求，推荐使用 InnoDB 作为 MySQL 的存储引擎.

表 1 MyISAM 引擎与 InnoDB 引擎的主要参数对比

Tab. 1 The comparison of MyISAM and InnoDB engine's main parameters

对比项目	MyISAM	InnoDB
事务	不支持	支持
外键	不支持	支持
行级锁	不支持	支持
单表行数	保存	不保存
查询速度	快	略慢

1.3 InnoDB 的表空间拆分

my. cnf 中有一个配置 innodb_file_per_table，其默认值为 0，意思是 InnoDB 引擎默认会将所有的数据库数据存储在一个共享空间(即一个唯一的物理文件 ibdata1). 这样做的一个明显的问题是，当增删数据库数据的时候，ibdata1 文件不会自动进行大小收缩，单个数据库的备份或者复制也将成为问题，通常只能使用 mysqldump 导出数据，然后再导入来解决这个问题. 可以修改 InnoDB 为独立表

空间模式，即 innodb_file_per_table = 1，数据库的每个表都会生成一个数据空间.

将 InnoDB 配置为独立表空间模式，除了数据备份方面的优势以外，还有两个好处. 首先，拆分表空间，可以让每一个单表对应一个独立的操作系统中的表空间物理文件，方便操作系统将这些相对小的，独立的文件放入系统缓存以加速访问，提供 MySQL 服务器整体的 IO 性能; 其次，拆分表空间后，可以把不同的表空间文件放到不同的硬盘上，分散每块硬盘的 IO 压力，尤其存储介质是某种磁盘阵列的时候，这个优势将更加明显. 共享表空间虽然在 Insert 操作上相对于独立表空间稍微有些优势，但是在其它方面，都没独立表空间表现好. 因此，推荐将 MySQL 配置为独立表空间，但是在实际配置时需合理调整下 innodb_open_files 的参数值，如果库里的表特别多的情况，可增加这个参数的值（默认值是 300）. 通过实际测试，发现这个参数可将 MySQL 的 IO 性能提高 50% 以上.

1.4　InnoDB 的内存优化

关于 MySQL 内存的优化，比较关键的一个参数是 innodb_buffer_pool_size，它和 MyISAM 的 key_buffer_size 有相似之处，但也是有差别的. innodb_buffer_pool_size 定义了 InnoDB 存储引擎的表数据和索引数据的最大内存缓冲区大小，为 Innodb 加速优化首要参数. InnoDB 引擎既可以缓存表的索引键，也可以缓存表中数据. 适当的增加这个参数的大小，可以有效的减少 InnoDB 类型表的磁盘 I/O，这个参数不能动态更改，所以分配需多考虑[5]. 在一个以 InnoDB 为主要存储引擎的专用 MySQL 数据库服务器上，推荐把该参数设置为物理内存大小的 60% ~70% ，例如，一个 10G 内存的服务器运行 MySQL，那么可以将这个参数配置为 innodb_buffer_pool_size = 6G.

1.5　InnoDB 日志控制参数的优化

参数 innodb_log_file_size 用于指定系统的日志组中每个日志文件的大小，在高写入负载尤其是大数据集的情况下很重要，这个值分配的大小和数据库写入速度、事务大小、异常重启后的恢复有很大关系. 一般来说，这个值越大则性能相对越高，但是要注意到可能会增加恢复时间. 要掌握的分配原则为，几个日志文件大小之和可设置为前述 innodb_buffer_pool_size 值的 25% ~100% ，一般控制在几个 LOG 文件相加大小在 2 G 以内为佳. 具体情况还需要考虑到事务大小和数据大小.

参数 innodb_log_files_in_group 用于指定日志组里文件的数目，一般来说这个值可设为 2 ~3 个，默认为 2 个.

参数 innodb_log_buffer_size 用于指定用来缓冲日志数据的缓冲区大小，一般控制在 2 ~8 M，如果它的值设置太高，可能导致内存浪费，因为它每 s 都会刷新一次，所以无需设置超过 1 s 所需的内存空间.

1.6　InnoDB 引擎的写操作磁盘缓存策略优化

InnoDB 写入操作的缓存策略，由唯一参数 innodb_flush_log_at_trx_commit 控制. 这个参数默认值为 1，表示每一次事务提交或写入指令都需要把日志 flush 到硬盘，在确认硬盘写入成功后，才认为这个写入操作成功完成，这会造成 IO 阻塞，不仅费时，也耗费资源，尤其是没有电池备用缓存时.

设置为 2 的意思是，不写入硬盘而是写入操作系统缓存. 操作系统缓存内的日志仍然会每 s flush 到硬盘，但是这个过程中 MySQL 写操作不会在磁盘 IO 的环节阻塞，而是在写入缓存后就认为写入成功且返回. 相对于默认配置，innodb_flush_log_at_trx_commit = 2 的配置会大大提高 MySQL 的 IO 性能. 但是如果服务器突然宕机或者断电，那么会损失 1 ~2 s 左右的写入数据.

这个参数设成 0，MySQL 读写会更快一点，但安全方面比较差，因为即使仅仅 MySQL 进程突然中止（服务器本身正常），也会丢失事务的数据.

从以上分析可以看出，这个参数的设置对 InnoDB 的性能有很大的影响，当这个值不为 1 时，可以取得较好的性能，但遇到异常会有损失，所以需要根据应用的情况去衡量.

表 2 给出了 MySQL 在配置层面的关键优化总结.

表 2 MySQL 在 InnoDB 引擎配置层面的关键优化

Tab. 2 The optimization of MySQL configurations under InnoDB engine

优化参数	推荐值	默认值
innodb_file_per_table	1	0
innodb_buffer_pool_size	系统最大内存的 60% 以上	1 G 或者 2 G
innodb_flush_log_at_trx_commit	0 或者 2	1

1.7 其他 MySQL 配置优化

character-set: 如果是单一语言，那么推荐使用简单的 character set，比如 latin1. 尽量不用 Utf-8，因为 Utf-8 字符占用物理空间较多，在高并发下，会对内存和硬盘造成更大负担.

max_allowed_packet: 这个参数一定要足够大，以满足比较大规模的 SQL 查询需求. 这个参数其实对性能没有太大的影响，主要是用来规避 packet error 类错误.

max_connections: MySQL server 允许同时保持的的最大连接数量. 这个参数的配置需要慎重，如果太小，那么可能导致数据库使用端无法连接数据库，配置太大又会导致 MySQL 服务器端的 out of memory 错误. 因为对于普通服务器，推荐这个参数配置在 1 000 左右.

table_cache: 这个参数定义了 MySQL 在同一时间内允许保持打开的 table 数量. 由于打开一个 table 对于 server 的资源开销还是比较大的，因此推荐这个参数设为不超过 512.

query_cache_size: 这个参数定义了每个查询的缓存内存大小，对于那些复杂查询，尤其是结果集或者临时表的内容很多的查询，这个参数设置的越大，查询的速度就越快，推荐配置为 64 M.

datadir: 这个字符串定义了 MySQL 存放数据的物理路径. 将这个路径和 MySQL 安装文件分开放在不同的磁盘上，可以提高一些 MySQL 的 IO 性能.

上述内容总结如表 3.

表 3 MySQL 通用参数优化

Tab. 3 The optimization of MySQL general parameters

优化参数	推荐值
character-set	latin1
max_allowed_packet	16 M
max_connections	1 000
table_cache	512
query_cache_size	64 M
datadir	与安装文件隔离在不同硬盘

2 MySQL 配置优化对应的硬件需求

在实际运营环境的测试结果说明 innodb_flush_log_at_trx_commit 这个参数配置，对于 MySQL 服务器的硬件是有相应的匹配要求的，主要体现在两个方面: 硬盘和内存.

2.1 InnoDB 引擎对硬盘的要求

对于普通机械硬盘以及通常意义上的企业级服务器机械硬盘，innodb_flush_log_at_trx_commit 配置从默认的 1 修改成 0 或者 2，对于 IO 性能的提升虽然比较显著，但也是有限的，性能提升大概在 50% ~100%. 近年固态硬盘(SSD 硬盘) 在生产环境中的应用越来越广泛[6-7]，使用 SSD 硬盘替换传统机械硬盘后发现，innodb_flush_log_at_trx_commit 配置从 1 修改成 0 或者 2，其性能提升幅度，远远高于普通硬盘在这个参数修改后的性能提升幅度. 经过测试，在 SSD 固态硬盘上，修改 innodb_flush_log_at_trx_commit 从 1 到 2，得到的 IO 随机读写吞吐性能提高，可以达到 6 000% ~7 000%，即 60 ~70 倍左右性能提升，因此在使用 SSD 硬盘后进行 innodb_flush_log_at_trx_commit 参数的优化，得到的 MySQL 的性能收益是最大的. 表 4 给出了 MySQL 在各种硬盘下，修改 innodb_flush_log_at_trx_commit 参数，得到的性能提升对比.

表 4　MySQL 各种硬盘下性能提升对比

Tab. 4　The comparison of MySQL performance improvement on different hard disk

硬盘类型	innodb_flush_log_at_trx_commit 从 1 修改 2 的性能提升幅度
普通台式机 7200 转机械硬盘	50% ~100%
服务器	15 000
转机械硬盘	50% ~100%
SSD 硬盘	6 000% ~7 000%

2.2　InnoDB 引擎对内存的要求

除了硬盘的要求外，innodb_flush_log_at_trx_commit 这个参数，如果修改成 2 或者 0，对服务里的内存规格也是有要求的. 其中必须使用支持 ECC 校验功能的服务器内存[8]，才能在 innodb_flush_log_at_trx_commit 参数配置成 0 或者 2 的情况下，保证 MySQL 的稳定运行. 测试表明，使用不带 ECC 校验功能的普通内存，在参数 innodb_flush_log_at_trx_commit 配置成 0 或者 2 的时候，在每 s 100 次左右的查询压力下，InnoDB 引擎大约每工作 1 ~3 d，就会出现一次严重错误，导致 MySQL 服务中断，在将内存换成支持 ECC 校验的企业级内存后，这个现象消失，MySQL 连续运行数月未出现问题.

3　MySQL 读写分离的集群配置

MySQL 通过主从复制（Master-Slave）的方式来同步数据[9]，在此基础上，MySQL 通过软件层实现的读写分离（MySQL-Proxy）程序或者中间件，将读写请求分发到不同服务器上，来提升数据库的并发负载能力[10]，读写业务逻辑如图 2 所示.

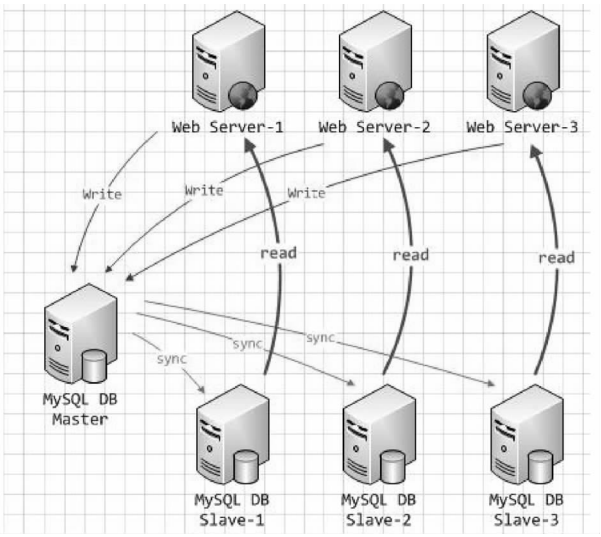

图 2　MySQL 主从复制、读写分离的原理图

Fig. 2　The schematic diagram of MySQL Master-Slave and MySQL-Proxy

MySQL 读写分离，对于数据库整体读性能的提升时非常显著的，其读性能可以随着集群里面从 MySQL 数据库数量的增加实现线性扩展，同时读写分离还在很大程度上减少了 MySQL 死锁发生的概率. 表 5 给出了一些经验数据.

表 5　MySQL 读写分离的性能

Tab. 5　The performance of MySQL-Proxy

MySQL 从数量	MySQL 读性能提高水平
1	30 ~40%
2	100% ~150%
3	200% ~300%

4　生产环境中的测试

通过一个短期在线全国性远程培训项目的相关数据，用于说明本文提及的各种优化方案、参数对 MySQL 的实际性能的提升效果.

在这个案例中，共需支撑约 15 000 人同时在线访问，每日的 UV 量为 5 000 以上，每日的 PV 量为 2 000 万次. 平台数据库选用 MySQL，通过 MySQL 主从结构实现读写分离；且由于业务的特殊性，在线用户与数据库交互高度活跃，数据库实时并发读写请求总量在高峰期达到每 s 1 000 次以上.

在这个项目的实施过程中，随着用户在线人数、在线活动的增长，数据业务密集程度的提高，在

运维过程中不断地遇到 MySQL 数据库层面的性能瓶颈，需要不断的分析和优化 MySQL 的软硬件结构. 总结起来，路径大致分为 7 个阶段:

(1) MySQL 主从结构为 1 主 3 从，配置基本采用默认配置，硬件方面、内存和硬盘均采用相对廉价的普通设备.

(2) 将部分表的存储引擎，由 MyISAM 修改成 InnoDB，对所有数据库慢查询进行优化，将所有表必要的单列索引、联合索引补充完整，同时，优化 1.5 与 1.7 节中所提及的 MySQL 配置参数.

(3) MySQL 主从架构升级为 1 主 7 从.

(4) 完成对数据库的表空间拆分.

(5) 完成 innodb_buffer_pool_size，innodb_flush_log_at_trx_commit 这两个参数的优化.

(6) 升级所有 MySQL 服务器的硬件性能，将普通硬盘升级为 SSD 硬盘，将普通内存升级为带 ECC 校验的企业级内存.

表 6 给出了上述各个阶段，MySQL 集群的性能表现.

表 6 MySQL 读写分离的性能

Tab. 6 The performance of MySQL-Proxy

阶段	服务器平均 CPU 占用	服务器平均 IO 占用	描述
1	100%	100%	大量业务由于数据库瓶颈执行超时，且部分业务造成 MySQL 锁表，写入操作过多，主从 MySQL 跟踪复制，无法同步
2	80 ~ 90%	100%	MySQL 锁表情况明显好转，但是数据库相关操作仍然很慢，主从 MySQL 仍然不同步
3	50 ~ 60%	90 ~ 95%	数据库读请求的执行明显变快，但是主从 MySQL 仍然不同步
4	40 ~ 50%	70 ~ 80%	主从 MySQL 复制不同步的时间长度缩短，但是仍然很难完全同步
5	40 ~ 50%	60 ~ 70%	主从 MySQL 复制不同步的时间长度进一步缩短，部分从库可以保持同步. 但是 MySQL 服务经常出现未知异常导致服务中断
6	10 ~ 20%	5 ~ 10%	系统运转稳定、性能良好

5 结语

本文通过在实际项目环境中的应用与测试，给出了 MySQL 关键配置的优化策略，存储引擎使用 InnoDB、表空间配置采用单表拆分以及写入策略采用非阻塞磁盘缓存机制，其中，写入策略采用非阻塞磁盘缓存机制. 对 MySQL 服务器硬件有一定要求，硬盘使用固态 SSD 硬盘，内存使用支持 ECC 校验的企业级内存.

参考文献:

[1] 唐汉明，翟振兴，兰丽华. 深入浅出 MySQL 数据库开发、优化与管理维护 [M]. 北京: 人民邮电出版社，2008.

[2] 薛军超. MySQL 网络数据库开发 [M]. 北京: 人民邮电出版社，2001.

[3] 罗凡，裴士辉，张雪松. MySQL 中 InnoDB 引擎的动态存储管理 [J]. 东北师大学报: 自然科学版，2006，38(1): 22 - 26.

[4] 顾治华，忽朝俭. MySQL 存储引擎与数据库性能 [J]. 计算机时代，2006，10: 8 - 10.

[5] 吴炳锡. Mysql Innodb 引擎优化 [EB/OL]. (2009 - 03 - 23). http://imysql.cn/blog/3208.

[6] 陈明达. 固态硬盘(SSD) 产品现状与展望 [J]. 移动通信，2009(11): 29 - 31.

[7] 彭觅. 固态硬盘 SSD 的性能分析和组建方案设计 [J]. 硅谷，2008(20): 25.

[8] 杨孝光. ECC 校验的算法分析和程序实现 [J]. 实验科学与技术，2004(3): 13 - 16.

[9] 方丹辉，张狄. MySQL 主从服务器数据库同步的实现 [J]. 计算机应用，2002，22(7): 11 - 13.

(责任编辑: 陈 静)