数据库参数配置优化

作者介绍

张小虎，甜橙金融持久化组负责人，多年**[数据库](http://lib.csdn.net/base/14" \o "MySQL知识库" \t "_blank)**优化、运维及开发经验。擅长不同业务场景下的数据库**[架构](http://lib.csdn.net/base/16" \o "大型网站架构知识库" \t "_blank)**设计及持久化解决方案。

前言

日常的**[MySQL](http://lib.csdn.net/base/14" \o "MySQL知识库" \t "_blank)**运维中说起调优，MySQL的配置文件my.cnf是不可忽略的。MySQL的默认参数并不能满足我们日常线上业务的需求，因此对参数进行优化也是不可缺少的环节。这里不想列出my.cnf配置中有多少项和每一项的意思，这些都可以在官方文档上查到。以下仅对日常工作用应该注意的一些参数进行说明。

下面针对一些参数进行说明。当然还有其它的设置可以起作用，取决于你的负载或硬件：在慢内存和快磁盘、高并发和写密集型负载情况下，你将需要特殊的调整。然而这里的目标是让你可以快速地获得一个稳健的MySQL配置，而不用花费太多时间在调整一些无关紧要的MySQL设置或读文档，找出哪些设置对你来说是重要的。

InnoDB配置

## 

从MySQL 5.5版本开始，InnoDB就是默认的存储引擎并且它比任何其它存储引擎的使用要多得多。那也是为什么它需要小心配置的原因。

**1innodb\_file\_per\_table**

表的数据和索引存放在共享表空间里或者单独表空间里。我们的工作场景安装是默认设置了innodb\_file\_per\_table = ON，这样也有助于工作中进行单独表空间的迁移工作。MySQL 5.6中，这个属性默认值是ON。

**2innodb\_flush\_log\_at\_trx\_commit**

默认值为1，表示InnoDB完全支持ACID特性。当你的主要关注点是数据安全的时候这个值是最合适的，比如在一个主节点上。但是对于磁盘（读写）速度较慢的系统，它会带来很巨大的开销，因为每次将改变flush到redo日志都需要额外的fsyncs。

如果将它的值设置为2会导致不太可靠（unreliable）。因为提交的事务仅仅每秒才flush一次到redo日志，但对于一些场景是可以接受的，比如对于主节点的备份节点这个值是可以接受的。如果值为0速度就更快了，但在系统崩溃时可能丢失一些数据：只适用于备份节点。说到这个参数就一定会想到另一个sync\_binlog。

**3innodb\_flush\_method**

这项配置决定了数据和日志写入硬盘的方式。一共有三种方式，我们默认使用O\_DIRECT 。O\_DIRECT模式：数据文件的写入操作是直接从mysql innodb buffer到磁盘的，并不用通过操作系统的缓冲，而真正的完成也是在flush这步，日志还是要经过OS缓冲。

**4innodb\_log\_buffer\_size**

这项配置决定了为尚未执行的事务分配的缓存。其默认值（1MB）一般来说已经够用了，但是如果你的事务中包含有二进制大对象或者大文本字段的话，这点缓存很快就会被填满并触发额外的I/O操作。看看Innodb\_log\_waits状态变量，如果它不是0，增加innodb\_log\_buffer\_size。

**5innodb\_buffer\_pool\_size**

这个参数应该是运维中必须关注的了。缓冲池是数据和索引缓存的地方，它属于MySQL的核心参数，默认为128MB，正常的情况下这个参数设置为物理内存的60%~70%。（不过我们的实例基本上都是多实例混部的，所以这个值还要根据业务规模来具体分析。）

**6innodb\_log\_file\_size**

这是redo日志的大小。redo日志被用于确保写操作快速而可靠并且在崩溃时恢复。如果你知道你的应用程序需要频繁地写入数据并且你使用的是MySQL 5.6，那么你可以一开始就把它这是成4G。（具体大小还要根据自身业务进行适当调整）

**7innodb\_support\_xa**

innodb\_support\_xa可以开关InnoDB的XA两段式事务提交。默认情况下，innodb\_support\_xa=true，支持XA两段式事务提交。由于XA两段式事务提交导致多余flush等操作，性能影响会达到10%，所有为了提高性能，有些DBA会设置innodb\_support\_xa=false。这样的话，redolog和binlog将无法同步，可能存在事务在主库提交，但是没有记录到binlog的情况。这样也有可能造成事务数据的丢失。

**8innodb\_additional\_mem\_pool\_size**

该参数用来存储数据字段信息和其他内部**[数据结构](http://lib.csdn.net/base/31" \o "算法与数据结构知识库" \t "_blank)**。表越多，需要在这里分配的内存越多。如果InnoDB用光了这个池内的内存，InnoDB开始从操作系统分配内存，并且往MySQL错误日志写警告信息，默认8MB。一般设置16MB。

**9max\_connections**

MySQL服务器默认连接数比较小，一般也就100来个最好把最大值设大一些。一般设置500~1000即可每一个链接都会占用一定的内存，所以这个参数也不是越大越好。有的人遇到too many connections会去增加这个参数的大小，但其实如果是业务量或者程序逻辑有问题或者sql写的不好，即使增大这个参数也无济于事，再次报错只是时间问题。在应用程序里使用连接池或者在MySQL里使用进程池有助于解决这一问题。

* Seesion级的内存分配

max\_threads(当前活跃连接数)\* (

read\_buffer\_size-- 顺序读缓冲，提高顺序读效率

+read\_rnd\_buffer\_size-- 随机读缓冲，提高随机读效率

+sort\_buffer\_size-- 排序缓冲，提高排序效率

+join\_buffer\_size-- 表连接缓冲，提高表连接效率

+binlog\_cache\_size-- 二进制日志缓冲，提高二进制日志写入效率ß

+tmp\_table\_size-- 内存临时表，提高临时表存储效率

+thread\_stack-- 线程堆栈，暂时寄存SQL语句/存储过程

+thread\_cache\_size-- 线程缓存，降低多次反复打开线程开销

+net\_buffer\_length-- 线程持连接缓冲以及读取结果缓冲

+bulk\_insert\_buffer\_size-- MyISAM表批量写入数据缓冲

)

* global级的内存分配

global buffer(全局内存分配总和) =

innodb\_buffer\_pool\_size

-- InnoDB高速缓冲，行数据、索引缓冲，以及事务锁、自适应哈希等

+ innodb\_additional\_mem\_pool\_size

-- InnoDB数据字典额外内存，缓存所有表数据字典

+innodb\_log\_buffer\_size

-- InnoDB REDO日志缓冲，提高REDO日志写入效率

+key\_buffer\_size

-- MyISAM表索引高速缓冲，提高MyISAM表索引读写效率

+query\_cache\_size

--查询高速缓存，缓存查询结果，提高反复查询返回效率+table\_cahce -- 表空间文件描述符缓存，提高数据表打开效率

+table\_definition\_cache

--表定义文件描述符缓存，提高数据表打开效率

参数的优化最终目的是让MySQL更好地利用资源通过合理地控制内存的分配，合理的CPU使用建议降低Session的内存分配。

**10server-id**

复制架构时确保 server-id 要不同，通常主ID要小于从ID。

**11log\_bin**

如果你想让数据库服务器充当主节点的备份节点，那么开启二进制日志是必须的。如果这么做了之后，还别忘了设置server\_id为一个唯一的值。就算只有一个服务器，如果你想做基于时间点的数据恢复，这（开启二进制日志）也是很有用的：从你最近的备份中恢复（全量备份），并应用二进制日志中的修改（增量备份）。

二进制日志一旦创建就将永久保存。所以如果你不想让磁盘空间耗尽，你可以用 PURGE BINARY LOGS 来清除旧文件，或者设置expire\_logs\_days 来指定过多少天日志将被自动清除。记录二进制日志不是没有开销的，所以如果你在一个非主节点的复制节点上不需要它的话，那么建议关闭这个选项。

**12skip\_name\_resolve**

当客户端连接数据库服务器时，服务器会进行主机名解析，并且当DNS很慢时，建立连接也会很慢。因此建议在启动服务器时关闭skip\_name\_resolve选项而不进行DNS查找。唯一的局限是之后GRANT语句中只能使用IP地址了，因此在添加这项设置到一个已有系统中必须格外小心。

**13sync\_binlog**

sync\_binlog 的默认值是0，像操作系统刷其他文件的机制一样，MySQL不会同步到磁盘中去而是依赖操作系统来刷新binary log。

当sync\_binlog =N (N>0) ，MySQL 在每写N次二进制日志binary log时，会使用fdatasync()函数将它的写二进制日志binary log同步到磁盘中去。当innodb\_flush\_log\_at\_trx\_commit和sync\_binlog  都为 1 时是最安全的，在mysqld服务崩溃或者服务器主机crash的情况下，binary log只有可能丢失最多一个语句或者一个事务。但是鱼与熊掌不可兼得，双1会导致频繁的IO操作，因此该模式也是最慢的一种方式。出于我们的业务考虑在业务压力允许的情况下默认的都是双1配置。

**14log\_slave\_update**

当业务中需要使用级联架构的时候log\_slave\_update = 1这个参数必须打开，否者第三级可能无法接收到第一级产生的binlog，从而无法进行数据同步。

**15tmpdir**

如果内存临时表超出了限制，MySQL就会自动地把它转化为基于磁盘的MyISAM表，存储在指定的tmpdir目录下.因此尽可能将tmpdir配置到性能好速度快的存储设备上。

**16慢日志相关**

slow\_query\_log = 1   #打开慢日志

slow\_query\_log\_file = /mysql/log/mysql.slow

long\_query\_time = 0.5  #设置超过多少秒的查询会入慢日志

## 

其他问题

## 

**1SSD对参数的影响**

随着科学技术的发展，越来越多的存储设备开始由传统的机械组件转向由电子元件组成的永久存储，且价钱越来越能让企业接受。存储组件速度提升后，再用传统机械组件的DB配置就显得浪费了，所以就需要针对不同的存储技术对MySQL配置作出调整，比如 innodb\_io\_capacity需要调大， 日志文件和redo放到机械硬盘， undo放到SSD， atomic write不需要Double Write Buffer， InnoDB压缩， 单机多实例+cgroup等等。分析 I/O 情况，动态调整 innodb\_io\_capacity 和 innodb\_max\_dirty\_pages\_pct；试图调整 innodb\_adaptive\_flushing，查看效果。

**2线程池设置**

针对innodb\_write\_io\_threads 和 innodb\_read\_io\_threads 的调优我们目前没有做，但我相信调整为8或者16，系统 I/O 性能会更好。还有，需要注意以下几点：任何一个调整，都要建立在数据的支撑和严谨的分析基础上，否则都是空谈； 这类调优是非常有意义的，是真正能带来价值的，所以需要多下功夫，并且尽可能地搞明白为什么要这么调整。

**3CPU相关**

* Innodb\_thread\_concurrency=0
* Innodb\_sync\_spin\_loops=288
* table\_definition\_cache=2000

**4IO相关的**

* Innodb\_flush\_method 建议用O\_DIRECT
* Innodb\_io\_capacity 设置成磁盘支持最大IOPS
* Innodb\_wirte\_io\_threads=8
* Innodb\_read\_io\_threads=8
* Innodb\_purge\_threads=1
* Innodb的预读方面，如果基于主建或是唯一索引的系统，建议禁用预读
* Innodb\_random\_read\_ahead = off