Atitit 数据库提升性能的机制总结db perf enhance v2 ubb.docx

Atitit 数据库提升性能的机制总结

目录

[1. 三大方法 通过架构 配置 代码调整 2](#_Toc24053)

[1.1. 上云 云虚拟机性能更高容易提升 2](#_Toc28302)

[1.2. 上ssd 可以立即带来一个数据级别的提升 2](#_Toc23064)

[1.3. 分库 2](#_Toc10667)

[1.4. 适当的数据库端代码，减少网络往返io与等待时间 2](#_Toc1486)

[2. 数据库本身提供的优化机制 2](#_Toc19588)

[3. 配置化解决的性能 2](#_Toc29686)

[3.1. 调整隔离级别为 可提交读read commit 2](#_Toc9232)

[3.2. 查询缓存query cache 调优数据库自带缓存机制 3](#_Toc5340)

[3.3. 非范围查询适当使用Hash索引比btree更快 3](#_Toc8670)

[3.4. 内存表机制 适当使用零时表机制 3](#_Toc29848)

[3.5. 使用好索引机制 hash btree 全文索引 3](#_Toc1880)

[3.6. Join表可以使用视图机制进行 sql重写 3](#_Toc1467)

[3.7. 子查询机制 提前过滤 4](#_Toc8283)

[3.8. Cte缓存机制 （mysql需要版本8） 4](#_Toc8722)

[3.9. 下列方法有助于最大限度地降低死锁： 低隔离级别 4](#_Toc12118)

[4. Other 4](#_Toc31425)

[4.1. 行锁变表锁问题 多实例 分库解决 分区 4](#_Toc32649)

[4.2. 集群 负载均衡 读写分离 5](#_Toc14860)

[4.3. 分区机制 5](#_Toc30238)

[4.4. 物化视图机制（mysql需要使用timer机制实现刷新） 6](#_Toc27779)

[4.5. NoSQL功能 支持json kv存储 6](#_Toc309)

[4.6. 换用更加大型数据库 6](#_Toc13419)

[4.7. 位图索引出马（msyql暂不支持，需自己实现） 6](#_Toc31767)

[4.8. 并可以建立业务索引 6](#_Toc8603)

[4.9. 适当使用数据库的拦截器机制事件机制 trigger触发器 7](#_Toc6694)

[4.10. 适当使用外键等约束检查，减少网络往返io性能消耗 7](#_Toc6446)

[4.11. Timer事件机制也可以适当使用 提升性能 延后异步处理一些事物 7](#_Toc13905)

[4.12. Json数据类型 减少join以及字段扩展提升 7](#_Toc21417)

[4.13. 适当使用sp，减少大量网络io 7](#_Toc4875)

[4.14. 窗口函数（Window functions）。 7](#_Toc28053)

[4.15. 适当使用 更新级联模式 7](#_Toc14104)

[5. 可能需要开发修改代码来达到的效果 7](#_Toc9624)

[5.1. 小事务范围 7](#_Toc13745)

[5.2. 将事务保持较短，并减小锁定占用率较高的查询 8](#_Toc31881)

[6. 批量数据写入模式性能提升 8](#_Toc25801)

[6.1. Ref 8](#_Toc20259)

# 三大方法 通过架构 配置 代码调整

## 上云 云虚拟机性能更高容易提升

## 上ssd 可以立即带来一个数据级别的提升

Iops更快，差距相当大。。大部分项目是属于iops类型，sql语句往往每次读写少量数据，但高频

## 分库

## 适当的数据库端代码，减少网络往返io与等待时间

把某些需要往返大量数据的代码（包括程序语言代码 和sql代码）部署到数据库服务器本机，这样可以节约大量网络io消耗与等待，提升应用响应速度。。缩短了调用链与层次

# 数据库本身提供的优化机制

# 配置化解决的性能

## 调整隔离级别为 可提交读read commit

锁变表锁问题解决

****总结：InnoDB的行锁是针对索引加的锁，不是针对记录加的锁。并且该索引不能失效，否则都会从行锁升级为表锁。****

****注意：行级锁都是基于索引的，如果一条SQL语句用不到索引是不会使用行级锁的，会使用表级锁。****

## 查询缓存query cache 调优数据库自带缓存机制

## 非范围查询适当使用Hash索引比btree更快

适用于非范围查询。。很多查询属于非范围查询

## 内存表机制 适当使用零时表机制

## 使用好索引机制 hash btree 全文索引

## Join表可以使用视图机制进行 sql重写

对于一些关联表的复杂查询，使用视图有时候会大大简化问题，因此在许多场合下都可以看到视图的身影， 它和直接使用JOIN的SQL语句有何区别？视图背后的原理又了解多少？

视图本身是一个虚拟表，不存放任何数据，查询视图的数据集由其他表生成。MySQL底层通过两种算法来实现视图：临时表算法（TEMPTABLE）和合并算法（MERGE）。

所谓临时表算法就是将SELECT语句的结果存放到临时表中，当需要访问视图的时候，直接访问这个临时表即可。

而合并算法则是重写包含视图的查询，将视图定义的SQL直接包含进查询SQL中

大部分情况下，尽量使用合并算法会有更好的性能。相当于sql重写

可能临时表上没有索引，则可以使用sql增加索引也是一种不错的方法

视图的实现算法是视图本身的属性决定的，跟作用在视图上的SQL没有任何关系。

那什么时候视图采用临时表算法，什么时候采用合并算法呢？一般来说，只要原表记录和视图中的记录无法建立一一映射的关系时，MySQL都将使用临时表算法来实现视图。

比如创建视图的SQL中包含GROUP BY、DISTINCT、UNION、聚合函数、子查询的时候，视图都将采用临时表算法（这些规则在以后的版本中，可能会发生改变，具体请参考官方手册）。

## 子查询机制 提前过滤

## Cte缓存机制 （mysql需要版本8）

可以自己使用临时表机制实现即可

## ****下列方法有助于最大限度地降低死锁：**** 低隔离级别

1. 按同一顺序访问对象。
2. 避免事务中的用户交互。
3. 保持事务简短并在一个批处理中。
4. 使用低隔离级别。
5. 使用绑定连接。

# Other

## 行锁变表锁问题 多实例 分库解决 分区

[mysql](https://m.yisu.com/mysql/" \o "mysql" \t "https://m.yisu.com/zixun/_blank):  
[Mysql](https://m.yisu.com/mysql/" \o "Mysql" \t "https://m.yisu.com/zixun/_blank)以表级锁为主，对资源锁定的粒度很大，如果一个session对一个表加锁时间过长，会让其他session无法更新此表中的数据。  
虽然InnoDB引擎的表可以用行级锁，但这个行级锁的机制依赖于表的索引，如果表没有索引，或者sql语句没有使用索引，那么仍然使用表级锁。

oracle:  
oracle使用行级锁，对资源锁定的粒度要小很多，只是锁定sql需要的资源，并且加锁是在数据库中的数据行上，不依赖与索引。所以oracle对并发性的支持要好很多。

## 集群 负载均衡 读写分离

## 分区机制

一般使用userid来隔离数据，rang范围模式，方便扩容无缝

## 物化视图机制（mysql需要使用timer机制实现刷新）

## NoSQL功能 支持json kv存储

## 换用更加大型数据库

大型 中型 小型数据库性能差距非常明显

## **位图索引出马（msyql暂不支持，需自己实现）**

如果用户查询的列的基数非常的小， 即只有的几个固定值，如性别、婚姻状况、行政区等等。要为这些基数值比较小的列建索引，就需要建立位图索引。

对于性别这个列，位图索引形成两个向量，男向量为10100...，向量的每一位表示该行是否是男，如果是则位1，否为0，同理，女向量位01011

位图索引的适用条件

　　上面讲了，位图索引适合只有几个固定值的列，如性别、婚姻状况、行政区等等，而身份证号这种类型不适合用位图索引。

## 并可以建立业务索引

业务索引可使用触发器或定时器实现触发

## 适当使用数据库的拦截器机制事件机制 trigger触发器

可能可以减少大量网络io往返，与连接建立的资源消耗，提升响应速度。。适用于多个修改的sql语句场景

## 适当使用外键等约束检查，减少网络往返io性能消耗

提升应用响应速度

## Timer事件机制也可以适当使用 提升性能 延后异步处理一些事物

通常可以替换一些触发器

## Json数据类型 减少join以及字段扩展提升

## 适当使用sp，减少大量网络io

特别是纯sql 数据 拿过来修改后又返回去的场景，比较常见

网络传输与交互，速度快

## 窗口函数（Window functions）。

## 适当使用 更新级联模式

# 可能需要开发修改代码来达到的效果

## 小事务范围

大力使用Json字段结构减少join关联查询消耗性能

## 将事务保持较短，并减小锁定占用率较高的查询

# 批量数据写入模式性能提升

临时禁用索引，批量插入修改后重建索引

事务批量提交模式

临时禁用触发器等

启用延迟insert cache等

## Ref

硬件提升性能的道路

单体性能，并行 多路，，cache ，缩短调用链距离

Atitit mysql adv fun feature MySQL高级特性与玩法

数据库性能对比 SQL Feature Comparison

https://www.sql-workbench.eu/dbms\_comparison.htmlhttps://www.sql-workbench.eu/dbms\_comparison.html