

百度二面，有点小激动！附面试题

=====

前几天刚面完百度，这不，没两天就收到二面邀请了，还有点小激动呢！来看看这次都问了哪些面试题吧，附答案仅供参考。

ConcurrentHashMap如何计算下标？

ConcurrentHashMap 计算下标和 HashMap 类似，它的主要执行流程有以下两步：

1. **计算 key 哈希值**：

1. **JDK 1.7**：key.hashCode()。

2. **JDK 1.8+**：((h=key.hashCode()) ^ (h>>>16)) -> 算法更均匀，哈希冲突越少。

2. **计算下标**：hash & (table.length-1)。

说说MVCC机制？

MVCC (Multi-Version Concurrency Control) 是一种并发控制机制，用于解决数据库并发访问中，数据一致性问题。它通过在读写操作期间保存多个数据版本，以提供并发事务间的隔离性，从而避免了传统的锁机制所带来的资源争用和阻塞问题。

在 MVCC 机制中，每个事务的读操作都能看到事务开始之前的一致性数据快照，而不受其他并发事务的修改的影响。核心思想是**通过创建多个数据版本，保持事务的一致性和隔离性**。

MVCC 主要是依靠以下两部分实现的：

1. Undo Log 链

2. Read View (读视图或者叫一致性视图)

Undo Log 链

我们知道 Undo Log 主要是用于数据库中事务回滚的，但在 MVCC 机制中也发挥着重要的作用，那什么是 Undo Log 链呢？

Undo Log 链是指在每个数据对象上维护的 Undo Log 记录链表。每张表都会有与之相对应的 Undo Log 链，用于记录修改前的数据信息（以方便数据进行回滚）。

![image.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/65330d42188347a682f04989c4eae902~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=860&h=902&s=581959&e=png&b=fd9f9)

Read View

Read View（读视图）用于管理事务之间数据可见性的一种机制。Read View 在特定时刻为事务创建的一个快照，该快照包含了在该时刻所有未提交事务的事务标识符，以及其他一些辅助信息。

在 Read View 中包含了以下 4 个主要的字段：

1. m_ids：当前活跃的事务编号集合。
2. min_trx_id：最小活跃事务编号。
3. max_trx_id：预分配事务编号，当前最大事务编号+1。
4. creator_trx_id：ReadView 创建者的事务编号。

RC 级别中，每次快照读都会生成一个全新的 Read View，而 RR 级别中同一个事务会复用一個 Read View。

有了 Read View 和 Undo Log 链之后，并发事务在查询时就知道要读取那些数据了。

判断方法

判断方法是根据 Read View 中的 4 个重要字段，先去 Undo Log 中最新的数据行进行比对，如果满足下面 Read View 的判断条件，则返回当前行的数据，如果不满足则继续查找 Undo Log 的下一行数据，直到找到满足的条件数据为止，如果查询完没有满足条件的数据，则返回 NULL。

判断规则

1. ****trx_id==creator_trx_id****: 先将 Undo Log 最新数据行中的 trx_id 和 ReadView 中的 creator_trx_id 进行对比, 如果他们两个值相同, 则说明是在同一个事务中执行, 那么直接返回当前 Undo Log 的数据行即可, 如果不相等, 则继续下面流程。
2. ****trx_id<min_trx_id****: 如果 trx_id 小于 min_trx_id, 则说明在执行查询时, 其他事务已经提交此行数据了, 那么直接返回此行数据即可, 如果大于等于, 则继续下面流程。
3. ****trx_id>max_trx_id****: 如果 trx_id 如果大于等于 max_trx_id, 则说明该行数据比当前操作执行的晚, 当前行数据不可见, 继续执行后续流程。
4. ****min_trx_id<=trx_id<max_trx_id****: trx_id 在 min_trx_id 和 max_trx_id 之间还分为以下两种情况:
 1. ****trx_id 在 m_ids 中****: 说明事务尚未执行完, 该行数据不可被访问。
 2. ****trx_id 未在 m_ids 中****: 说明事务已经执行完, 可以返回该行数据。

以上判断规则从 Undo Log 最新的行数据, 逐行对比, 直到找到匹配的数据, 否则查询完未匹配上, 则返回 NULL。

说说读已提交和可重复读各自创建ReadView的时机?

创建 ReadView 时机如下:

*** **读已提交 (Read Committed) ****: 在读已提交隔离级别下, MySQL 为每一个****读取操作****创建一个新的 ReadView。这意味着每次执行 SELECT 查询时, 都会根据当前活跃事务的状态重新计算可见性视图。这样做的结果是, 同一个事务内的连续两次查询可能会看到不同的数据, 因为第二次查询可能会看到第一次查询后其他事务提交的新数据。因此, 在这个隔离级别下, 事务能看到其他事务已经提交的修改。

*** **可重复读 (Repeatable Read) ****: 在可重复读隔离级别下, MySQL为****整个事务****而不是单个查询创建一个 ReadView。也就是说, 当一个事务开始时, MySQL 会为该事务创建一个快照 (Snapshot), 这个快照包含了数据库在事务开始时刻的所有数据的一个一致性视图。在整个事务的生命周期内, 不论执行多少次查询操作, 都是基于这个初始创建的 ReadView 来决定数据的可见性, 确保事务内多次相同的查询结果是一致的, 即“可重复读”。因此, 在这个隔离级别下, 事务开始后, 不会看到其他事务后续提交的修改。

你知道哪些常用的Linux命令?

Linux 常用的命令有以下这些：

1. **ls**：用于列出目录中的文件和子目录。
2. **pwd**：显示当前工作目录的路径。
3. **cd**：切换到指定工作目录。
4. **mkdir**：创建一个新的目录。
5. **rmdir**：删除一个空目录。
6. **rm**：删除文件或目录。
7. **cp**：复制文件或目录。
8. **mv**：移动或重命名文件或目录。
9. **touch**：创建空文件或更新文件的时间戳。
10. **less**：分页显示文件内容。
11. **tail**：显示文件的开头或结尾部分的内容（可查看动态日志）。
12. **cat**：查看文件内容或将多个文件内容合并输出。
13. **grep**：在文件中搜索指定的文本模式。
14. **ps**：显示系统中的进程信息。
15. **kill**：终止指定进程。
16. **ifconfig/ip**：查看和配置网络接口信息。
17. **ping**：测试网络连接。
18. **wget/curl**：从网络下载文件。
19. **chmod**：改变文件或目录的权限。

如何排查CPU占用比较高的问题？

以 Linux 系统为例，排查 CPU 飙升问题的实现步骤如下：

1. 使用 top 命令，查询占用 CPU 最高的进程 ID。
2. 查询该进程 ID 中，哪个线程占用的 CPU 资源最多。
3. 将占用 CPU 资源最多的线程 ID 转换成 16 进制。
4. 使用 Java 自带的工具 jstack 查询该线程的详细信息，定位到问题代码，分析代码和解决 CPU 飙升的问题。

说说Top命令各个指标的具体含义？

使用 top 命令查询 cpu 占用最高的进程 ID (PID) ，如下图所示：

![https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/778297aca656401686c7c8427e416ac2~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1720&h=573&s=107489&e=png&b=0e1019)

> 可以使用 shift+P 快捷键进行 CPU 占用率排序（从高到低）。

指标含义：包括进程ID (PID)、用户 (USER)、优先级 (PR)、nice 值 (NI)、虚拟内存使用量 (VIRT)、物理内存使用量 (RES)、共享内存大小 (SHR)、进程状态 (S)、CPU 使用率 (%CPU)、内存使用率 (%MEM)、进程累计 CPU 时间 (TIME+) 和命令名 (COMMAND)。

其中：

1. ****优先级 (PR)****：优先级是进程在内核中的实际调度优先级，也称为动态优先级。它反映了进程被调度占用 CPU 的实际顺序。在 top 命令中，PR 列显示的是进程的调度优先级，对于实时进程，使用“RT”标记；对于非实时进程（普通用户进程），其取值范围是 0 至 39，值越小优先级越高。
2. ****nice值 (NI)****：nice 值是进程的用户态优先级，也称为静态优先级。它是一个从 -20 到 19 的数值，其中负数表示较高的优先级，正数表示较低的优先级。默认情况下，大多数进程的 nice 值为0。通过 nice 和 renice 命令，用户可以调整进程的 nice 值，从而影响其优先级。

Redis为什么快？

Redis 运行比较快的原因主要有以下几点：

1. ****纯内存操作****：Redis 将所有数据存储在内存中，这意味着对数据的读写操作直接在内存中进行，而内存的访问速度远远高于磁盘。这种设计使得 Redis 能够以接近硬件极限的速度处理数据读写。
2. ****单线程模型****：Redis 使用单线程模型来处理客户端请求。这可能听起来似乎效率不高，但实际上，这种设计避免了多线程频繁切换和过度竞争所带来的性能开销。Redis 每个请求的执行时间都很短，因此在单线程下，也能够处理大量的并发请求。
3. ****I/O多路复用****：Redis 使用了 I/O 多路复用技术，可以在单个线程中同时监听多个客户端连接，只有当有网络事件发生时才会进行实际的 I/O 操作。这样有效地利用了 CPU 资源，减少了无谓的等待和上下文切换。
4. ****高效数据结构****：Redis 提供了多种高效的数据结构，如哈希表、有序集合等。这些数据结构的实现都经过了优化，使得 Redis 在处理这些数据结构的操作时非常高效。

Redis有哪些持久化方式？

Redis 4.0 之后支持以下 3 种持久化方案：

1. ****RDB (Redis DataBase) 持久化****：快照方式持久化，将某一个时刻的内存数据，以二进制的方式写入磁盘。占用空间小，恢复快，可能存在数据丢失。
2. ****AOF (Append Only File) 持久化****：文件追加持久化，记录所有非查询操作命令，并以文本的形式追加到文件中。数据通常不易丢失，但占用空间大。
3. ****混合持久化****：RDB + AOF 混合方式的持久化，Redis 4.0 之后新增的方式，混合持久化是结合了 RDB 和 AOF 的优点，在写入的时候，先把当前的数据以 RDB 的形式写入文件的开头，再将后续的操作命令以 AOF 的格式存入文件，这样既能保证 Redis 重启时的速度，又能减低数据丢失的风险。

说说AOF中的写时复制技术？

① AOF 重写背景

随着 Redis 运行，AOF 文件会不断增长，因为每次写操作都被追加到文件中。为了减小文件体积、提高恢复速度，Redis 提供了 AOF 重写功能，它可以创建一个新的、更紧凑的 AOF 文件，仅包含重建当前数据集所需的最小命令序列。

② 写时复制在 AOF 中的应用

1. ****子进程写时复制****：Redis 在执行 AOF 重写 (bgrewriteaof) 时，会 fork 出一个子进程 (bgsave 子进程) 来负责 AOF 文件的重写，主进程依然执行 Redis 的业务命令。
2. ****AOF 重写遇到写操作****：在 bgsave 子进程运行期间，如果主进程有写操作（如修改 key-value），主进程会采用写时复制机制。具体来说，主进程会把这个新写或修改的数据写入到一个新的物理地址中，并修改自己的页表映射。这样，虚拟页和物理页的关系在子进程中保持不变，而主进程中的数据已经被更新。
3. ****AOF 重写缓冲区****：为了解决主进程在 AOF 重写过程中修改数据导致的数据不一致问题，Redis 设置了一个 AOF 重写缓冲区。在重写 AOF 期间，当 Redis 执行完一个写命令之后，它会同时将这个写命令写入到 AOF 缓冲区和 AOF 重写缓冲区。当 AOF 重写完成后，Redis 会将 AOF 重写缓冲区中的命令追加到新的 AOF 文件中，以确保数据的一致性。

手撕算法：三数之和？

* **题目**：[leetcode.cn/problems/3sum/](http://cxyroad.com/"https://leetcode.cn/problems/3sum/description/")

* **解题思路**：双指针+数组变量。

* **解题思路推荐**：[leetcode.cn/problems/3sum/](http://cxyroad.com/"https://leetcode.cn/problems/3sum/solutions/12307/hua-jie-suan-fa-15-san-shu-zhi-he-by-guanpengchn/")

> 本文已收录到我的面试小站 [www.javacn.site](http://cxyroad.com/"https://www.javacn.site"), 其中包含的内容有：Redis、JVM、并发、并发、MySQL、Spring、Spring MVC、Spring Boot、Spring Cloud、MyBatis、设计模式、消息队列等模块。

原文链接: <https://juejin.cn/post/7381855857124147200>