

XXP7 笔试题

1. *junit* 用法, *before*, *beforeClass*, *after*, *afterClass* 的执行顺序
2. 分布式锁
3. *nginx* 的请求转发算法, 如何配置根据权重转发
4. 用 *hashmap* 实现 *redis* 有什么问题 (死锁, 死循环, 可用 *ConcurrentHashMap*)
5. 线程的状态
5. 线程的阻塞的方式
6. *sleep* 和 *wait* 的区别
7. *hashmap* 的底层实现
8. 一万个人抢 100 个红包, 如何实现 (不用队列), 如何保证 2 个人不能抢到同一个红包, 可用分布式锁
9. *java* 内存模型, 垃圾回收机制, 不可达算法
10. 两个 *Integer* 的引用对象传给一个 *swap* 方法在方法内部交换引用, 返回后, 两个引用的值是否会发现变化
11. *aop* 的底层实现, 动态代理是如何动态, 假如有 100 个对象, 如何动态的为这 100 个对象代理
12. 是否用过 *maven install*。 *maven test*。 *git* (*make install* 是安装本地 *jar* 包)
13. *tomcat* 的各种配置, 如何配置 *docBase*

14. *spring* 的 *bean* 配置的几种方式
15. *web.xml* 的配置
16. *spring* 的监听器。
17. *zookeeper* 的实现机制，有缓存，如何存储注册服务的
18. *IO* 会阻塞吗？*readLine* 是不是阻塞的
19. 用过 *spring* 的线程池还是 *java* 的线程池？
20. 字符串的格式化方法 （20，21 这两个问题问的太低级了）
21. 时间的格式化方法
22. 定时器用什么做的
23. 线程如何退出结束
24. *java* 有哪些锁？乐观锁 悲观锁 *synchronized* 可重入锁 读写锁,用过 *reentrantlock* 吗？*reentrantlock* 与 *synmchronized* 的区别
25. *ThreadLocal* 的使用场景
26. *java* 的内存模型，垃圾回收机制
27. 为什么线程执行要调用 *start* 而不是直接 *run*（直接 *run*，跟普通方法没什么区别，先调 *start*，*run* 才会作为一个线程方法运行）
28. *qmq* 消息的实现机制(*qmq* 是去哪儿网自己封装的消息队列)
29. 遍历 *hashmap* 的三种方式
30. *jvm* 的一些命令

31. *memcache* 和 *redis* 的区别
32. *mysql* 的行级锁加在哪个位置
33. *ConcurrentHashMap* 的锁是如何加的？是不是分段越多越好
34. *myisam* 和 *innodb* 的区别（*innodb* 是行级锁，*myisam* 是表级锁）
35. *mysql* 其他的性能优化方式
36. *linux* 系统日志在哪里看
37. 如何查看网络进程
38. 统计一个整数的二进制表示中 *bit* 为 *1* 的个数
39. *jvm* 内存模型，*java* 内存模型
40. 如何把 *java* 内存的数据全部 *dump* 出来
41. 如何手动触发全量回收垃圾，如何立即触发垃圾回收
42. *hashmap* 如果只有一个写其他全读会出什么问题
43. *git rebase*
44. *mongodb* 和 *hbase* 的区别

45. 如何解决并发问题

46. *volatile* 的用途

47. *java* 线程池（好像之前我的理解有问题）

48. *mysql* 的 *binlog*

49. 代理模式

50. *mysql* 是如何实现事务的

51. 读写分离何时强制要读主库，读哪个从库是通过什么方式决定的，从库的同步 *mysql* 用的什么方式

52. *mysql* 的存储引擎

53. *mysql* 的默认隔离级别，其他隔离级别

54. 将一个链表反转（用三个指针，但是每次只发转一个）

55. *spring Aop* 的实现原理，具体说说

56. 何时会内存泄漏，内存泄漏会抛哪些异常

57. 是否用过 *Autowired* 注解

58. *spring* 的注入 *bean* 的方式

59. *sql* 语句各种条件的执行顺序，如 *select*，*where*，*order by*，*group by*

60. *select xx from xx where xx and xx order by xx limit xx*；如何优化这个（看 *explain*）

61. 四则元算写代码

62. 统计 100G 的 ip 文件中出现 ip 次数最多的 100 个 ip

63. zookeeper 的事物，结点，服务提供方挂了如何告知消费方

64. 5 台服务器如何选出 leader(选举算法)

65. 适配器和代理模式的区别

66. 读写锁

67. static 加锁

68. 事务隔离级别

69. 门面模式，类图(外观模式)

70. mybatis 如何映射表结构

71. 二叉树遍历

72. 主从复制

73. mysql 引擎区别

74. 静态内部类加载到了哪个区？方法区

75. class 文件编译后加载到了哪

76. web 的 http 请求如何整体响应时间变长导致处理的请求数变少，该如何处理？用队列，当处理不了那么多 http 请求时将请求放到队列

中慢慢处理，web 如何实现队列

77. 线程安全的单例模式

78. 快速排序性能考虑

79. *volatile* 关键字用法

80. 求表的 *size*，或做数据统计可用什么存储引擎

81. 读多写少可用什么引擎

82. 假如要统计多个表应该用什么引擎

83. *concurrenthashmap* 求 *size* 是如何加锁的，如果刚求完一段后这段发生了变化该如何处理

84. 1000 个苹果放 10 个篮子，怎么放，能让我拿到所有可能的个数

85. 可重入的读写锁，可重入是如何实现的？

86. 是否用过 *NIO*

87. *java* 的 *concurrent* 包用过没

88. *sting s=new string("abc")* 分别在堆栈上新建了哪些对象

89. *java* 虚拟机的区域分配，各区分别存什么

90. 分布式事务 (*JTA*)

91. *threadlocal* 使用时注意的问题（*ThreadLocal* 和 *Synchronized* 都用于解决多线程并发访问。但是 *ThreadLocal* 与 *synchronized* 有本质的区别。*synchronized* 是利用锁的机制，使变量或代码块在某一时刻只能被一个线程访问。而 *ThreadLocal* 为每一个线程都提供了变量的副本，使得每个线程在某一时间访问到的并不是同一个对象，这样就隔离了多个线程对数据的数据共享。而 *Synchronized* 却正好相反，它用于在多个线程间通信时能够获得数据共享）

92. java 有哪些容器(集合，*tomcat* 也是一种容器)

93. 二分查找算法

94. *myisam* 的优点，和 *innodb* 的区别

95. *redis* 能存哪些类型

96. *http* 协议格式，*get* 和 *post* 的区别

97. 可重入锁中对应的 *wait* 和 *notify*

98. *redis* 能把内存空间交换进磁盘中吗(这个应该是可以的，但是那个面试官非跟我说不可以)

99. java 线程池中基于缓存和基于定长的两种线程池，当请求太多时分别是如何处理的？定长的事用的队列，如果队列也满了呢？交换进磁盘？基于缓存的线程池解决方法呢？

100. *synchronized* 加在方法上用的什么锁

101. 可重入锁中的 *lock* 和 *trylock* 的区别

102. *innodb* 对一行数据的读会枷锁吗？不枷锁，读实际读的是副本

103. *redis* 做缓存是分布式存的？不同的服务器上存的数据是否重复？*guava cache* 呢？是否重复？不同的机器存的数据不同

104. 用 *awk* 统计一个 *ip* 文件中 *top10*

105. 对表做统计时可直接看 *schema info* 信息，即查看表的系统信息

106. *mysql* 目前用的版本

107. 公司经验丰富的人给了什么帮助？(一般 *boss* 面会问这些)

108. 自己相对于一样的应届生有什么优势

109. 自己的好的总结习惯给自己今后的工作带了什么帮助，举例为证

110. 原子类，线程安全的对象，异常的处理方式

111. 4 亿个 *int* 数，如何找出重复的数（用 *hash* 方法，建一个 2 的 32 次方个 *bit* 的 *hash* 数组，每取一个 *int* 数，可 *hash* 下 2 的 32 次方找到它在 *hash* 数组中的位置，然后将 *bit* 置 1 表示已存在）

112. 4 亿个 *url*，找出其中重复的（考虑内存不够，通过 *hash* 算法，将 *url* 分配到 1000 个文件中，不同的文件间肯定就不会重复了，再分别找出重复的）

有 1 万个数组，每个数组有 1000 个整数，每个数组都是降序的，从中找出最大的 *N* 个数， $N < 1000$

113. *LinkedHashMap* 的底层实现

114. 类序列化时类的版本号的用途，如果没有指定一个版本号，系统是怎么处理的？如果加了字段会怎么样？

115. *Override* 和 *Overload* 的区别，分别用在什么场景

116. java 的反射是如何实现的