



持续更新中

回复“1024”获取Java架构师资源



微信搜一搜

Java研发军团

Java研发军团《Java面试手册》V1.0
公众号后台回复“面试手册”

Memcached面试题

1、Memcached 是什么，有什么作用？

Memcached 是一个开源的，高性能的内存缓存软件，从名称上看 Mem 就是内存的意思，而 Cache 就是缓存的意思。Memcached 的作用：通过在事先规划好的内存空间中临时缓存数据库中的各类数据，以达到减少业务对数据库的直接高并发访问，从而达到提升数据库的访问性能，加速网站集群动态应用服务的能力。

2、memcached 服务在企业集群架构中有哪些应用场景？

一、作为数据库的前端缓存应用

a、完整缓存（易），静态缓存

例如：商品分类（京东），以及商品信息，可事先放在内存里，然后再对外提供数据访问，这种先放到内存，我们称之为预热，（先把数据缓存中），用户访问时可以只读取 memcached 缓存，不读取数据库了。

b、热点缓存（难）

需要前端 web 程序配合，只缓存热点的数据，即缓存经常被访问的数据。先预热数据库里的基础数据，然后在动态更新，选读取缓存，如果缓存里没有对应的数据，程序再去读取数据库，然后程序把读取的新数据放入缓存存储。

特殊说明：

□ 如果碰到电商秒杀等高并发的业务，一定要事先预热，或者其它思想实现，例如：秒杀只是获取资格，而不是瞬间秒杀到手商品。

那么什么是获取资格？

- 就是在数据库中，把 0 标成 1.就有资格啦。再慢慢的去领取商品订单。因为秒杀过程太长会占用服务器资源。
- 如果数据更新，同时触发缓存更新，防止给用户过期数据。
- 对于持久化缓存存储系统，例如：redis，可以替代一部分数据库的存储，一些简单的数据业务，投票，统计，好友关注，商品分类等。nosql= not onlysql

二、作业集群的 session 会话共享存储。

- Memcached 服务在不同企业业务应用场景中的工作流程

- 当 web 程序需要访问后端数据库获取数据时会优先访问 Memcached 内存缓存，如果缓存中有数据就直接获取返回前端服务及用户，如果没有数据（没有命中），在由程序请求后端的数据库服务器，获取到对应的数据后，除了返回给前端服务及用户数据外，还会把数据放到 Memcached 内存中进行缓存，等待下次请求被访问，Memcache 内存始终是数据库的挡箭牌，从而大大的减轻数据库的访问压力，提高整个网站架构的响应速度，提升了用户体验。
- 当程序更新，修改或删除数据库中已有的数据时，会同时发送请求通知 Memcached 已经缓存的同一个 ID 内容的旧数据失效，从而保证 Memcache 中数据和数据库中的数据一致。
- 如果在高并发场合，除了通知 Memcached 过程的缓存失效外，还会通过相关机制，使得在用户访问新数据前，通过程序预先先把更新过的数据推送到 memcache 中缓存起来，这样可以减少数据库的访问压力，提升 Memcached 中缓存命中率。
- 数据库插件可以再写入更新数据库后，自动抛给 MC 缓存起来，自身不 Cache。

2、Memcached 服务分布式集群如何实现？

特殊说明：Memcached 集群和 web 服务集群是不一样的，所有 Memcached 的数据总和才是数据库的数据。每台 Memcached 都是部分数据。（一台 memcached 的数据，就是一部分 mysql 数据库的数据）

a、程序端实现

程序加载所有 mc 的 ip 列表，通过对 key 做 hash（一致性哈希算法）

例如：web1 (key)===>对应 A,B,C,D,E,F,G.....若干台服务器。（通过哈希算法实现）

b、负载均衡器

通过对 key 做 hash（一致性哈希算法）一致哈希算法的目的是不但保证每个对象只请求一个对应的服务器，而且当节点宕机，缓存服务器的更新重新分配比例降到最低。

3、Memcached 服务特点及工作原理是什么？

- a、完全基于内存缓存的
- b、节点之间相互独立
- c、C/S 模式架构，C 语言编写，总共 2000 行代码。
- d、异步 I/O 模型，使用 libevent 作为事件通知机制。
- e、被缓存的数据以 key/value 键值对形式存在的。
- f、全部数据存放于内存中，无持久性存储的设计，重启服务器，内存里的数据会丢失。
- g、当内存中缓存的数据容量达到启动时设定的内存值时，就自动使用 LRU 算法删除过期的缓存数据。
- h、可以对存储的数据设置过期时间，这样过期后的数据自动被清除，服务本身不会监控过期，而是在访问的时候查看 key 的时间戳，判断是否过期。
- j、memcache 会对设定的内存进行分块，再把块分组，然后再提供服务

4、简述 Memcached 内存管理机制原理？

早期的 Memcached 内存管理方式是通过 malloc 的分配的内存，使用完后通过 free 来回收内存，这种方式容易产生内存碎片，并降低操作系统对内存的管理效率。加重操作系统内存管理器的负担，最坏的情况下，会导致操作系统比 memcached 进程本身还慢，为了解决这个问题，Slab Allocation 内存分配机制就诞生了。

现在 Memcached 利用 Slab Allocation 机制来分配和管理内存。

Slab Allocation 机制原理是按照预先规定的大小，将分配给 memcached 的内存分割成特定长度的内存块 (chunk)，再把尺寸相同的内存块，分成组 (chunks slab class),这些内存块不会释放，可以重复利用。

而且，slab allocator 还有重复使用已分配的内存的目的。也就是说，分配到的内存不会释放，而是重复利用。

Slab Allocation 的主要术语Page分配给 Slab 的内存空间，默认是 1MB。分配给 Slab 之后根据 slab 的大小切分成chunk。

Chunk

用于缓存记录的内存空间。

SlabClass

特定大小的 chunk 的组

5、memcached 是怎么工作的？

Memcached 的神奇来自两阶段哈希 (two-stage hash)。Memcached 就像一个巨大的、存储了很多 <key,value>对的哈希表。通过 key，可以存储或查询任意的数据。

客户端可以把数据存储在多台 memcached 上。当查询数据时，客户端首先参考节点列表计算出 key 的哈希值 (阶段一哈希)，进而选中一个节点；客户端将请求发送给选中的节点，然后 memcached 节点通过一个内部的哈希算法 (阶段二哈希)，查找真正的数据 (item)

6、memcached 最大的优势是什么？

Memcached 最大的好处就是它带来了极佳的水平可扩展性，特别是在一个巨大的系统中。由于客户端自己做了一次哈希，那么我们很容易增加大量 memcached 到集群中。memcached 之间没有相互通信，因此不会增加 memcached 的负载；没有多播协议，不会网络通信量爆炸 (implode)。memcached 的集群很好用。内存不够了？增加几台 memcached 吧；CPU 不够用了？再增加几台吧；有多余的内存？在增加几台吧，不要浪费了。

基于 memcached 的基本原则，可以相当轻松地构建出不同类型的缓存架构。除了这篇 FAQ，在其他地方很容易找到详细资料的。

7、memcached 和 MySQL 的 query

cache 相比，有什么优缺点？

把 memcached 引入应用中，还是需要不少工作量的。MySQL 有个使用方便的query cache，可以自动地缓存 SQL 查询的结果，被缓存的 SQL 查询可以被反复地快速执行。Memcached 与之相比，怎么样呢？MySQL 的 query cache 是集中式的，连接到该 query cache 的 MySQL 服务器都会受益。

□ 当您修改表时，MySQL 的 query cache 会立刻被刷新 (flush)。存储一个 memcached item 只需要很少的时间，但是当写操作很频繁时，MySQL的 query cache 会经常让所有缓存数据都失效。

□ 在多核 CPU 上，MySQL 的 query cache 会遇到扩展问题 (scalabilityissues)。在多核 CPU 上，query cache 会增加一个全局锁 (global lock)，由于需要刷新更多的缓存数据，速度会变得更慢。

□ 在 MySQL 的 query cache 中，我们是不能存储任意的数据的 (只能是SQL 查询结果)。而利用 emcached，我们可以搭建出各种高效的缓存。比如，可以执行多个独立的查询，构建出一个用户对象 (user object)，然后将用户对象缓存到 memcached 中。而 query cache 是 SQL 语句级别的，不可能做到这一点。在小的网站中，query cache 会有所帮助，但随着网站规模的增加，query cache 的弊将大于利。

□ query cache能够利用的内存容量受到MySQL服务器空闲内存空间的限制。给数据库服务器增加更多的内存来缓存数据，固然是很好的。但是，有了memcached，只要您有空闲的内存，都可以用来增加memcached 集群的规模，然后您就可以缓存更多的数据。

8、memcached 和服务器的 local cache (比如 PHP 的 APC、mmap 文件等) 相比，有什么优缺点？

首先，local cache 有许多与上面(query cache)相同的问题。local cache 能够利用的内存容量受到 (单台) 服务器空闲内存空间的限制。不过，local cache 有一点比 memcached 和 query cache 都要好，那就是它不但可以存储任意的数据，而且没有网络存取的延迟。

□ local cache 的数据查询更快。考虑把 highly common 的数据放在 localcache 中吧。如果每个页面都需要加载一些数量较少的数据，考虑把它们放在local cached 吧。

□ local cache 缺少集体失效 (groupinvalidation) 的特性。在 memcached 集群中，删除或更新一个 key 会让所有的观察者觉察到。但是在 local cache 中，我们只能通知所有的服务器刷新 cache (很慢，不具扩展性)，或者仅仅依赖缓存超时失效机制。

□ local cache 面临着严重的内存限制，这一点上面已经提到。

9、memcached 的 cache 机制是怎样的？

Memcached 主要的 cache 机制是 LRU (最近最少用) 算法 + 超时失效。当您存数据到 memcached 中，可以指定该数据在缓存中可以呆多久 Which is forever, or some time in the future。如果 memcached 的内存不够用了，过期的 slabs 会优先被替换，接着就轮到最老的未被使用的 slabs。

10、memcached 如何实现冗余机制？

不实现！我们对这个问题感到很惊讶。Memcached 应该是应用的缓存层。它的设计本身就不带有任何冗余机制。如果一个 memcached 节点失去了所有数据，您应该可以从数据源 (比如数据库) 再次获取到数据。您应该特别注意，您的应用应该可以容忍节点的失效。不要写一些糟糕的查询代码，寄希望于 memcached 来保证一切！如果您担心节点失效会大大加重数据库的负担，那么您可以采取一些办法。比如您可以增加更多的节点 (来减少丢失一个节点的影响)，热备节点 (在其他节点 down 了的时候接管 IP)，等等

11、memcached 如何处理容错的？

不处理！在 memcached 节点失效的情况下，集群没有必要做任何容错处理。如果发生了节点失效，应对的措施完全取决于用户。节点失效时，下面列出几种方案供您选择：

□ 忽略它！在失效节点被恢复或替换之前，还有很多其他节点可以应对节点失效带来的影响。

□ 把失效的节点从节点列表中移除。做这个操作千万要小心！在默认情况下 (余数式哈希算法)，客户端添加或移除节点，会导致所有的缓存数据不可用！因为哈希参照的节点列表变化了，大部分 key 会因为哈希值的改变而被映射到 (与原来) 不同的节点上。

□ 启动热备节点，接管失效节点所占用的 IP。这样可以防止哈希紊乱 (hashing chaos)。

□ 如果希望添加和移除节点，而不影响原先的哈希结果，可以使用一致性哈希算法（consistent hashing）。您可以百度一下一致性哈希算法。支持一致性哈希的客户端已经很成熟，而且被广泛使用。去尝试一下吧！

□ 两次哈希（reshing）。当客户端存取数据时，如果发现一个节点 down 了，就再做一次哈希（哈希算法与前一次不同），重新选择另一个节点（需要注意的时，客户端并没有把 down 的节点从节点列表中移除，下次还是有可能先哈希到它）。如果某个节点时好时坏，两次哈希的方法就有风险了，好的节点和坏的节点上都可能存在脏数据（stale data）。

12、如何将 memcached 中 item 批量导入导出？

您不应该这样做！Memcached 是一个非阻塞的服务器。任何可能导致 memcached 暂停或瞬时拒绝服务的操作都应该值得深思熟虑。向 memcached 中批量导入数据往往不是您真正想要的！想象看，如果缓存数据在导出导入之间发生了变化，您就需要处理脏数据了；

13、如果缓存数据在导出导入之间过期了，您又怎么处理这些数据呢？

因此，批量导出导入数据并不像您想象中的那么有用。不过在一个场景倒是很有用。如果您有大量的从不变化的数据，并且希望缓存很快热（warm）起来，批量导入缓存数据是很有帮助的。虽然这个场景并不典型，但却经常发生，因此我们会考虑在将来实现批量导出导入的功能。如果一个 memcached 节点 down 了让您很痛苦，那么您还会陷入其他很多麻烦。您的系统太脆弱了。您需要做一些优化工作。比如处理“惊群”问题（比如 memcached 节点都失效了，反复的查询让您的数据库不堪重负...这个问题在 FAQ 的其他提到过），或者优化不好的查询。记住，Memcached 并不是您逃避优化查询的借口。

14、memcached 是如何做身份验证的？

没有身份认证机制！memcached 是运行在应用下层的软件（身份验证应该是应用上层的职责）。memcached 的客户端和服务端之所以是轻量级的，部分原因就是完全没有实现身份验证机制。这样，memcached 可以很快地创建新连接，服务端也无需任何配置。如果您希望限制访问，您可以使用防火墙，或者让 memcached 监听 unix domain socket

15、memcached 的多线程是什么？如何使用它们？

线程就是定律（threads rule）！在 Steven Grimm 和 Facebook 的努力下，memcached 1.2 及更高版本拥有了多线程模式。多线程模式允许 memcached 能够充分利用多个 CPU，并在 CPU 之间共享所有的缓存数据。memcached 使用一种简单的锁机制来保证数据更新操作的互斥。相比在同一个物理机器上运行多个

memcached 实例，这种方式能够更有效地处理 multi gets。

如果您的系统负载并不重，也许您不需要启用多线程工作模式。如果您在运行一个拥有大规模硬件的、庞大的网站，您将会看到多线程的好处。

简单地总结一下：命令解析（memcached 在这里花了大部分时间）可以运行在多线程模式下。memcached 内部对数据的操作是基于很多全局锁的（因此这部分工作不是多线程的）。未来对多线程模式的改进，将移除大量的全局锁，提高 memcached 在负载极高的场景下的性能。

16、memcached 能接受的 key 的最大长度是多少？

key 的最大长度是 250 个字符。需要注意的是，250 是 memcached 服务器端内部的限制，如果您使用的客户端支持“key 的前缀”或类似特性，那么 key（前缀+原始 key）的最大长度是可以超过 250 个字符的。我们推荐使用使用较短的 key，因为可以节省内存和带宽。

17、memcached 对 item 的过期时间有什么限制？

过期时间最大可以达到 30 天。memcached 把传入的过期时间（时间段）解释成时间点后，一旦到了这个时间点，memcached 就把 item 置为失效状态。这是一个简单但 obscure 的机制。

18、memcached 最大能存储多大的单个 item？

1MB。如果你的数据大于 1MB，可以考虑在客户端压缩或拆分到多个 key 中。

为什么单个 item 的大小被限制在 1M byte 之内？

啊...这是一个大家经常问的问题！

简单的回答：因为内存分配器的算法就是这样的。

详细的回答：Memcached 的内存存储引擎（引擎将来可插拔...），使用 slabs 来管理内存。内存被分成大小不等的 slabs chunks（先分成大小相等的 slabs，然后每个 slab 被分成大小相等 chunks，不同 slab 的 chunk 大小是不相等的）。chunk 的大小依次从一个最小数开始，按某个因子增长，直到达到最大的可能值。

19、memcached 能够更有效地使用内存吗？

Memcache 客户端仅根据哈希算法来决定将某个 key 存储在哪个节点上，而不考虑节点的内存大小。因此，您可以在不同的节点上使用大小不等的缓存。但是一般都是这样做的：拥有较多内存的节点上可以运行多个 memcached 实例，每个实例使用的内存跟其他节点上的实例相同。

20、什么是二进制协议，我该关注吗？

关于二进制最好的信息当然是二进制协议规范：

二进制协议尝试为端提供一个更有效的、可靠的协议，减少客户端/服务器端因处理协议而产生的 CPU 时间。

根据 Facebook 的测试，解析 ASCII 协议是 memcached 中消耗 CPU 时间最多的环节。所以，我们为什么不改进 ASCII 协议呢？

21、memcached 的内存分配器是如何工作的？为什么不适用 malloc/free！？为何要使用 slabs？

实际上，这是一个编译时选项。默认会使用内部的 slab 分配器。您确实应该使用内建的 slab 分配器。最早的时候，memcached 只使用 malloc/free 来管理内存。然而，这种方式不能与 OS 的内存管理以前很好地工作。反复地 malloc/free 造成了内存碎片，OS 最终花费大量的时间去查找连续的内存块来满足 malloc 的请求，而不是运行 memcached 进程。如果您不同意，当然可以使用 malloc！

只是不要在邮件列表中抱怨啊

slab 分配器就是为了解决这个问题而生的。内存被分配并划分成 chunks，一直被重复使用。因为内存被划分成大小不等的 slabs，如果 item 的大小与被选择存放它的 slab 不是很合适的话，就会浪费一些内存。Steven Grimm 正在这方面已经做出了有效的改进。

22、memcached 是原子的吗？

所有的被发送到 memcached 的单个命令是完全原子的。如果您针对同一份数据同时发送了一个 set 命令和一个 get 命令，它们不会影响对方。它们将被串行化、先后执行。即使在多线程模式，所有的命令都是原子的，除非程序有 bug:)

命令序列不是原子的。如果您通过 get 命令获取了一个 item，修改了它，然后想把它 set 回 memcached，我们不保证这个 item 没有被其他进程（process，未必是操作系统中的进程）操作过。在并发的情况下，您也可能覆写了一个被其他进程 set 的 item。

memcached 1.2.5 以及更高版本，提供了 gets 和 cas 命令，它们可以解决上面的问题。如果您使用 gets 命令查询某个 key 的 item，memcached 会给您返回该 item 当前值的唯一标识。如果您覆写了这个 item 并想把它写回到 memcached 中，您可以通过 cas 命令把那个唯一标识一起发送给 memcached。如果该 item 存放在 memcached 中的唯一标识与您提供的一致，您的写操作将会成功。如果另一个进程在这期间也修改了这个 item，那么该 item 存放在 memcached 中的唯一标识将会改变，您的写操作就会失败

23、如何实现集群中的 session 共享存储？

Session 是运行在一台服务器上的，所有的访问都会到达我们的唯一服务器上，这样我们可以根据客户端传来的 sessionId，来获取 session，或在对应 Session 不存在的情况下（session 生命周期到了/用户第一次登录），创建一个新的 Session；但是，如果我们在集群环境下，假设我们有两台服务器 A，B，用户的请求会由

Nginx 服务器进行转发（别的方案也是同理），用户登录时，Nginx 将请求转发至服务器 A 上，A 创建了新的 session，并将 SessionID 返回给客户端，用户在浏览其他页面时，客户端验证登录状态，Nginx 将请求转发至服务器 B，由于 B 上并没有对应客户端发来 sessionId 的 session，所以会重新创建一个新的 session，并且再将这个新的 sessionId 返回给客户端，这样，我们可以想象一下，用户每一次操作都有 1/2 的概率进行再次的登录，这样不仅对用户体验特别差，还会让服务器上的 session 激增，加大服务器的运行压力。

为了解决集群环境下的 session 共享问题，共有 4 种解决方案：

1.粘性 session

粘性 session 是指 Nginx 每次都把同一用户的所有请求转发至同一台服务器上，即将用户与服务器绑定。

2.服务器 session 复制

即每次 session 发生变化时，创建或者修改，就广播给所有集群中的服务器，使所有的服务器上的 session 相同。

3.session 共享

缓存 session，使用 redis，memcached。

4.session 持久化

将 session 存储至数据库中，像操作数据一样才做 session

24、memcached 与 redis 的区别

- 1、Redis 不仅仅支持简单的 k/v 类型的数据，同时还提供 list，set，zset，hash 等数据结构的存储。而 memcache 只支持简单数据类型，需要客户端自己处理复杂对象
- 2、Redis 支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用（PS：持久化在 rdb、aof）。
- 3、由于 Memcache 没有持久化机制，因此宕机所有缓存数据失效。Redis 配置为持久化，宕机重启后，将自动加载宕机时刻的数据到缓存系统中。具有更好的灾备机制。
- 4、Memcache 可以使用 Magent 在客户端进行一致性 hash 做分布式。Redis 支持在服务器端做分布式（PS:Twemproxy/Codis/Redis-cluster 多种分布式实现方式）
- 5、Memcached 的简单限制就是键（key）和 Value 的限制。最大键长为 250 个字符。可以接受的储存数据不能超过 1MB（可修改配置文件变大），因为这是典型 slab 的最大值，不适合虚拟机使用。而 Redis 的 Key 长度支持到 512k。
- 6、Redis 使用的是单线程模型，保证了数据按顺序提交。Memcache 需要使用 cas 保证数据一致性。CAS（Check and Set）是一个确保并发一致性的机制，属于“乐观锁”范畴；原理很简单：拿版本号，操作，对比版本号，如果一致就操作，不一致就放弃任何操作cpu 利用。由于 Redis 只使用单核，而 Memcached 可以使用多核，所以平均每一个核上 Redis 在存储小数据时比 Memcached 性能更高。而在 100k 以上的数据中，Memcached 性能要高于 Redis。
- 7、memcache 内存管理：使用 Slab Allocation。原理相当简单，预先分配一系列大小固定的组，然后根据数据大小选择最合适的块存储。避免了内存碎片。（缺点：不能变长，浪费了一定空间）memcached 默认情况下下一个 slab 的最大值为前一个的 1.25 倍。
- 8、redis 内存管理：Redis 通过定义一个数组来记录所有的内存分配情况，Redis 采用的是包装的 malloc/free，相较于 Memcached 的内存管理方法来说，要简单很多。由于 malloc 首先以链表的方式搜索已管理的内存中可用的空间分配，导致内存碎片比较多



持续更新中

回复“1024”获取Java架构师资源



微信搜一搜



Java研发军团

Java研发军团《Java面试手册》V1.0
公众号后台回复“面试手册”