redis 从入门到实战（4）-redis 主从复制原理

作者：SmiteLi
原文链接：https://ld246.com/article/1571724954978
来源网站：链滴

<h2 id="复写">复写</h2>
在 Redis 复制的基础上（不包括 Redis Cluster 或 Redis Sentinel 作为附加层提供的高可用性能），使用和配置领导者跟随者（主从）复制非常简单：它允许从属 Redis 实例精确实例的副本。每次链接断开时，从站将自动重新连接到主站，并且无论主站发生什么况，它都会尝试作为它的精确副本。
该系统使用三种主要机制工作：

当主实例和从属实例连接良好时，主实例通过向从属实例发送命令流来保持从属实例的更新，以复制由于以下原因而对主节点上发生的数据集产生的影响：客户端写入，密钥已过期或驱逐任何其他改主数据集的操作。
当主从服务器之间的链接中断时，由于网络问题或由于在主服务器或从服务器中检测到超时，从务器将重新连接并尝试进行部分重新同步：这意味着它将尝试仅获取部分断开期间错过的命令流的大。
如果无法进行部分重新同步，则从站将要求完全重新同步。这将涉及一个更复杂的过程，在此过中，主服务器需要创建其所有数据的快照，将其发送到从服务器，然后在数据集更改时继续发送命令。

默认情况下，Redis 使用异步复制（低延迟和高性能）是绝大多数 Redis 用例的自然复制模式。是，Redis 从站异步地确认它们定期与主站接收的数据量。因此，主机不会每次都等待从机处理命令但是，如果需要，它知道哪个从机已经处理了什么命令。这允许具有可选的同步复制。
客户端可以使用 WAIT 命令请求某些数据的同步制。但是，WAIT 仅能确保其他 Redis 实例中具有指定量的已确认副本，它不会将一组 Redis 实例转换为具有高度一致性的 CP 系统：在故障转移期间，仍会丢失已确认的写入，具体取决于关于 Redis 持久性的确切配置。但是，使用 WAIT，在发生故障事件后丢失写操作的可能性会大大降低，从而很难触发某些障模式。
您可以查看 Sentinel 或 Redis 群集文档，以获取有关高可用性和故障转移的更多信息。本文档其余部分主要描述 Redis 基本复制的基本特征。
以下是有关 Redis 复制的一些非常重要的事实：

Redis 使用异步复制，异步从属到主对处理的数据量进行确认。
一个主机可以有多个从机。
从站能够接受其他从站的连接。除了将多个从站连接到同一主站之外，从站还可以以级联结构连到其他从站。从 Redis 4.0 开始，所有从属服务器都将从主服务器接收完全相同的复制流。
Redis 复制在主端无阻塞。这意味着当一个或多个从属设备执行初始同步或部分重新同步时，主设备将继续处理查询。
复制在从属端也基本上没有阻塞。当从属服务器执行初始同步时，假定您在 redis.conf 中配置了edis，则它可以使用旧版本的数据集处理查询。否则，您可以配置 Redis 从属服务器，以在复制流关时将错误返回给客户端。但是，在初始同步之后，必须删除旧的数据集，并且必须加载新的数据集。设备将在此短暂的窗口内阻止传入的连接（对于非常大的数据集，该连接可能长达几秒钟）。从 Redis 4.0 开始，可以对 Redis 进行配置，以使旧数据集的删除发生在其他线程中，但是加载新的初始数据仍将在主线程中进行，并阻塞从属。
复制既可以用于可伸缩性，也可以用于多个从服务器进行只读查询（例如，可以将慢速的 O（N操作卸载到从服务器上），也可以仅用于提高数据安全性和高可用性。
可以使用复制来避免让主服务器将整个数据集写入磁盘的成本：一种典型的技术包括配置主服务器<code>redis.conf</code> 以避免完全保留到磁盘，然后连接配置为不时保存的从服务器，或者已用 AOF。但是，必须谨慎处理此设置，因为重启的主服务器将以一个空的数据集开始：如果从服务器试与其同步，则从服务器也将被清空。

<h2 id="主服务器上的持久性关闭时复制的安全性">主服务器上的持久性关闭时复制的安全性</h2>
在使用 Redis 复制的设置中，强烈建议在主服务器和从服务器中打开持久性。如果不可能（例如

由于磁盘速度非常慢而导致延迟问题），则应配置实例，以避免重启后<strong自动重启。</p>
为了更好地理解为什么将持久性已关闭的主服务器配置为自动重启是危险的，请检查以下故障模，其中擦除了主服务器及其所有从服务器上的数据：

我们有一个设置，其中节点 A 充当主节点，持久性被关闭，并且节点 B 和 C 从节点 A 复制。</l>
节点 A 崩溃，但是它具有一些自动重新启动系统，该系统可以重新启动进程。但是，由于关闭了久性，因此节点将使用空数据集重新启动。
节点 B 和 C 将从节点 A 复制，节点 A 为空，因此它们将有效销毁其数据副本。

当将 Redis Sentinel 用于高可用性时，关闭主服务器上的持久性以及自动重启该过程也是很危的。例如，主机可以足够快速地重新启动，以使 Sentinel 不会检测到故障，从而发生上述故障模式。/p>
每当数据安全性很重要，并且复制与配置了持久性的主服务器一起使用时，都应禁用实例的自动启。
<h2 id="Redis复制如何工作">Redis 复制如何工作</h2>
每个 Redis 主数据库都有一个复制 ID：它是一个较大的伪随机字符串，用于标记数据集的给定事。每个主服务器还采用一个偏移量，该偏移量会针对复制流中要发送给从服务器的每个复制字节增，以使用修改数据集的新更改来更新从服务器的状态。即使实际上没有连接任何从服务器，复制偏移会增加，因此基本上每个给定的一对：
<pre><code class="highlight-chroma">Replication ID, offset

</code></pre>
标识母版数据集的确切版本。
当从属服务器连接到主服务器时，它们使用 PSYNC</> 命令来发送其旧的主服务器复制 ID 和到目前为止已处理的偏移量。这样主机可以只发送所需的增部分。但是，如果主缓冲区中的积压不足，或者从属正在引用的历史记录（复制 ID）再为人所知，则会发生完全重新同步：在这种情况下，从属将获得数据集的完整副本， 从头开始。
完整同步的工作方式如下：
主机开始后台保存过程以生成 RDB 文件。同时，它开始缓冲从客户端收到的所有新写命令。后保存完成后，主服务器将数据库文件传输到从服务器，从服务器将其保存在磁盘上，然后将其加载到存中。然后，主机将所有缓冲命令发送到从机。这是作为命令流完成的，并且与 Redis 协议本身的格相同。
您可以自己通过 telnet 尝试。服务器正在执行某些工作时，请连接到 Redis 端口并发出 <a href "https://ld246.com/forward?goto=https%3A%2F%2Fredis.io%2Fcommands%2Fsync" target=" blank" rel="nofollow ugc">SYNC 命令。您将看到批量传输，然后主服务器收到的每个命令会在 telnet 会话中重新发出。实际上，SYNC 是一个旧的协议，不再供较新的 Redis 实例使用，但仍然存在，以实现向后兼容性：它不允许部分重新同步因此现在改用 PSYNC。
如前所述，当主从链路由于某种原因断开时，从属能够自动重新连接。如果主服务器收到多个并的从服务器同步请求，它将执行一次后台保存，以便为所有请求提供服务。
<h2 id="复制ID说明">复制 ID 说明</h2>
在上一节中，我们说过，如果两个实例具有相同的复制 ID 和复制偏移，则它们具有完全相同的据。但是，了解复制 ID 到底是什么以及为什么实例实际上具有两个复制 ID（主要 ID 和辅助 ID）很用。
复制 ID 基本上标记了数据集的给定历史记录。每次实例作为主服务器从头重新启，或从服务器升级为主服务器时，都会为此实例生成一个新的复制 ID。与主服务器连接的从服务器将握手后继承其复制 ID。因此，具有相同 ID 的两个实例通过以下事实相关联：它们拥有相同的数据，

是可能在不同的时间。对于拥有最新数据集的给定历史记录（复制 ID），这是理解逻辑时间的偏移量
例如，如果两个实例 A 和 B 具有相同的复制 ID，但是一个实例的偏移量为 1000，另一个实例偏移量为 1023，则意味着第一个实例缺少应用于数据集的某些命令。这也意味着，通过仅应用一些令，A 可能会达到与 B 完全相同的状态。
Redis 实例具有两个复制 ID 的原因是由于从属服务器被提升为主服务器。故障转移后，升级的属服务器仍然需要记住其过去的复制 ID，因为该复制 ID 是以前的主服务器之一。这样，当其他从属务器将与新的主服务器同步时，它们将尝试使用旧的主服务器复制 ID 执行部分重新同步。这将按预工作，因为当从属设备升级为主设备时，它会将其辅助 ID 设置为其主要 ID，并记住发生此 ID 切换的偏移量。稍后它将选择一个新的随机复制 ID，因为新的历史记录开始了。处理新的从站连接时，主将其 ID 和偏移量与当前 ID 和辅助 ID 相匹配（为安全起见，直到给定的偏移量）。简而言之，这意着在故障转移之后，连接到新提升的主节点的从节点不必执行完全同步。
如果您想知道为什么升级为主服务器的从属服务器在故障转移后需要更改其复制 ID：由于某些络分区，旧的主服务器仍可能作为主服务器工作：保留相同的复制 ID 会违反以下事实：任何两个随实例的相同 ID 和相同偏移量意味着它们具有相同的数据集。
<h2 id="无盘复制">无盘复制</h2>
通常，完全重新同步需要在磁盘上创建 RDB 文件，然后从磁盘重新加载相同的 RDB，以便为从提供数据。
对于慢速磁盘，这对于主服务器而言可能是非常压力的操作。Redis 2.8.18 版是第一个支持无盘制的版本。在此设置中，子进程直接通过电线将 RDB 发送到从属服务器，而无需使用磁盘作为中间储。
<h2 id="组态">组态</h2>
要配置基本的 Redis 复制很简单：只需将以下行添加到从属配置文件中：
<pre><code class="highlight-chroma">slaveof 192.168.1.1 6379

</code></pre>
当然，您需要用主 IP 地址（或主机名）和端口替换 192.168.1.1 6379。或者，您可以调用 SLAVEOF 命令，主主机将开始与从机同步。
还有一些参数可用于调整主服务器在内存中执行的复制积压，以执行部分□□重新同步。有关 <cod>redis.conf</code> 更多信息，请参见 Redis 发行版附带的示例 。
可以使用 <code>repl-diskless-sync</code> 配置参数启用无盘复制。启动传输的延迟，以便 <code>repl-diskless-sync-delay</code> 参数控制第一个从机之后等待更多从机到达。请参阅 <cde>redis.conf</code> Redis 发行版中的示例文件以获取更多详细信息。
<h2 id="只读从站">只读从站</h2>
从 Redis 2.6 开始，从站支持默认情况下启用的只读模式。此行为由 <code>slave-read-only</ode> redis.conf 文件中的选项控制，并且可以在运行时使用 CONFIG SET 启用和禁用。
只读从站将拒绝所有写入命令，因此由于错误而无法写入从站。这并不意味着该功能旨在将从属例公开到 Internet，或更普遍地说是公开存在不信任客户端的网络，因为仍启用 <code>DEBUG</cde> 或类似的管理命令 <code>CONFIG</code>。但是，可以通过使用 <code>rename-commad</code> 伪指令禁用 redis.conf 中的命令来提高只读实例的安全性。
您可能想知道为什么可以还原只读设置并具有可被写操作作为目标的从属实例。如果从设备和主备重新同步或从设备重新启动，这些写入将被丢弃，但是有一些合法的用例将临时数据存储在可写的设备中。
例如，计算慢速 Set 或 Sorted set 操作并将其存储到本地密钥中是多次观察到的可写 Slave 的例。
但是请注意，版本 4.0 之前的可写从属无法使密钥到期并具有生存时间。意味着，如果您使用 EXPIRE 或其他为密钥设置最大TL 的命令，则密钥将泄漏，并且当您使用读取命令访问它时可能不再看到它时，您将在密钥计数中到它。仍会使用内存。因此，通常将可写的从属服务器（4.0 版之前的版本）和密钥与 TTL 混合会造

问题。
Redis 4.0 RC3 和更高版本完全解决了这个问题，现在可写的从属服务器能够像主服务器一样用 TL 驱逐密钥，但以 DB 编号大于 63 的密钥除外（但默认情况下，Redis 实例只有 16 个数据库）。</>
还要注意，由于 Redis 4.0 从属写入仅在本地进行，因此不会传播到连接到该实例的子从属。子属将始终接收与顶级主服务器发送给中间从属的复制流相同的复制流。因此，例如在以下设置中：</>
<pre><code class="highlight-chroma">A ---> B ---> C

</code></pre>
即使 <code>B</code> 是可写的，C 也不会看到 <code>B</code> 写入，而是具有与主实相同的数据集 <code>A</code>。
<h2 id="设置从属服务器向主服务器进行身份验证">设置从属服务器向主服务器进行身份验证</h2>
如果您的主服务器通过拥有密码 <code>requirepass</code>，那么配置从属服务器以在所有步操作中使用该密码很简单。
要在正在运行的实例上执行此操作，请使用 <code>redis-cli</code> 并键入：
<pre><code class="highlight-chroma">config set masterauth <password>

</code></pre>
要永久设置，请将其添加到您的配置文件中：
<pre><code class="highlight-chroma">masterauth <password>

</code></pre>
<h2 id="只允许写入N个附加副本">只允许写入 N 个附加副本</h2>
从 Redis 2.8 开始，可以将 Redis 主服务器配置为仅在当前至少有 N 个从服务器连接到主服务时才接受写查询。
但是，由于 Redis 使用异步复制，因此无法确保从站实际收到给定的写操作，因此始终存在数据失的窗口。
该功能的工作方式如下：

Redis 从站每秒对主站执行一次 ping 操作，确认已处理的复制流数量。
Redis 主机会记住上一次它收到每个从机的 ping 命令。
用户可以配置滞后不大于最大秒数的最小数量的从站。

如果至少有 N 个从站，且延迟小于 M 秒，则将接受写入。
您可能将其视为一种尽力而为的数据安全机制，其中不能确保给定写入的一致性，但至少将数据失的时间窗限制为给定的秒数。通常，绑定数据丢失比未绑定数据丢失更好。
如果不满足条件，则主机将返回一个错误，并且将不接受写入。
此功能有两个配置参数：

最小从动写 <code><number of slaves></code>
最小 - 最大 - 从层 <code><number of seconds></code>

有关更多信息，请检查 <code>redis.conf</code> Redis 源分发附带的示例文件。
<h2 id="Redis复制如何处理密钥失效">Redis 复制如何处理密钥失效</h2>
Redis 过期使密钥有有限的生存时间。这种功能取决于实例计算时间的能力，但是 Redis 从属会确复制带有过期的密钥，即使使用 Lua 脚本更改了此类密钥也是如此。
要实现这种功能，Redis 不能依赖于主时钟和从时钟具有同步时钟的能力，因为这是一个无法解的问题，将导致竞争状况和数据集的差异，因此 Redis 使用三种主要技术来使过期密钥的复制能够正工作：

从站不使密钥失效，而是等待主服务器使密钥失效。当主服务器使密钥过期（或由于 LRU 而将退出）时，它会合成一个 DEL 命令，该命令将传输到所从服务器。
但是，由于主机驱动的过期，有时从机可能仍在逻辑上已过期的内存密钥中，因为主机无法及时供 DEL 命令。为了处理从属服务器使用其逻辑时钟，以报告仅在不违反数据集一致性的读操作中不存在密钥（因将收到来自主机的新命令）。这样，从站避免报告仍然存在逻辑过期的密钥。实际上，使用从属进行放的 HTML 片段缓存将避免返回早于所需生存时间的项目。
在 Lua 脚本执行期间，不会执行任何密钥过期。当 Lua 脚本运行时，从概念上讲，主服务器中时间将被冻结，因此在脚本运行的所有时间内给定密钥将存在或不存在。这样可以防止密钥在脚本中过期，并且这是将密钥发送到从属服务器所必需的，以确保在数据集中具有相同的效果。

一旦将奴隶提升为主人，它将开始独立地失效密钥，并且不需要其旧主人的任何帮助。
<h2 id="在Docker和NAT中配置复制">在 Docker 和 NAT 中配置复制</h2>
当使用端口转发或网络地址转换的 Docker 或其他类型的容器时，Redis 复制需要格外小心，特是在使用 Redis Sentinel 或其他扫描主 INFO 或 ROLE 命令输出以发现从属地址的系统时，尤其如此。
问题在于，ROLE 命令和 INFO 输出的复制部分在发布到主实例后，将显示从属具有用于连接到主的 IP 地址在使用 NAT 的环境中，从属与到从属实例的逻辑地址（客户端应使用该逻辑地址连接到从属实例）
同样，将列出从属服务器，并将侦听端口配置为 <code>redis.conf</code>，如果重新映射端，则该端口可能与转发端口不同。
为了解决这两个问题，从 Redis 3.2.2 开始，可以强制从服务器向主服务器声明任意一对 IP 和端。使用的两个配置指令是：
<pre><code class="highlight-chroma">slave-announce-ip 5.5.5.5
slave-announce-p rt 1234

</code></pre>
并且在 <code>redis.conf</code> 最近的 Redis 发行示例中有记录。
<h2 id="INFO和ROLE命令">INFO 和 ROLE 命令</h2>
有两个 Redis 命令可提供有关主实例和从实例的当前复制参数的大量信息。一种是 INFO。如果使用 <code>replication</code> 参数作为参数调用命令则 <code>INFO replication</code> 仅显示与复制相关的信息。另一个对计算机更友好的命令是 ROLE，它提供主服务器和从服务器的复制状态以及它们的制偏移量，已连接的从服务器的列表等。
<h2 id="重新启动和故障转移后的部分重新同步">重新启动和故障转移后的部分重新同步</h2>
从 Redis 4.0 开始，在故障转移后将实例提升为主节点时，它仍将能够与旧主节点的从节点执行分重新同步。为此，从属服务器会记住旧的复制 ID 及其以前的主服务器的偏移量，因此，即使连接从属服务器要求提供旧的复制 ID，也可以将部分积压提供给正在连接的从属服务器。
但是，升级后的从属的新复制 ID 将有所不同，因为它构成了数据集的不同历史记录。例如，主务器可以返回可用状态，并且可以继续接受写入一段时间，因此在升级的从服务器中使用相同的复制 I 会违反复制 ID 和偏移对 a 仅标识单个数据集的规则。
此外，从属设备在轻轻关闭电源并重新启动后，能够在 <code>RDB</code> 文件中存储 与主

备重新同步所需的信息。这在升级时很有用。如果需要，最好使用 SHUTDOWN 命令在 <code>save & quit</code> 从站上执行操作。
无法部分重新同步通过 AOF 文件重新启动的从站。但是，可以在关闭实例之前将实例转换为 RDB 持久性，然后再重新启动，最后可以再次启用 AOF。