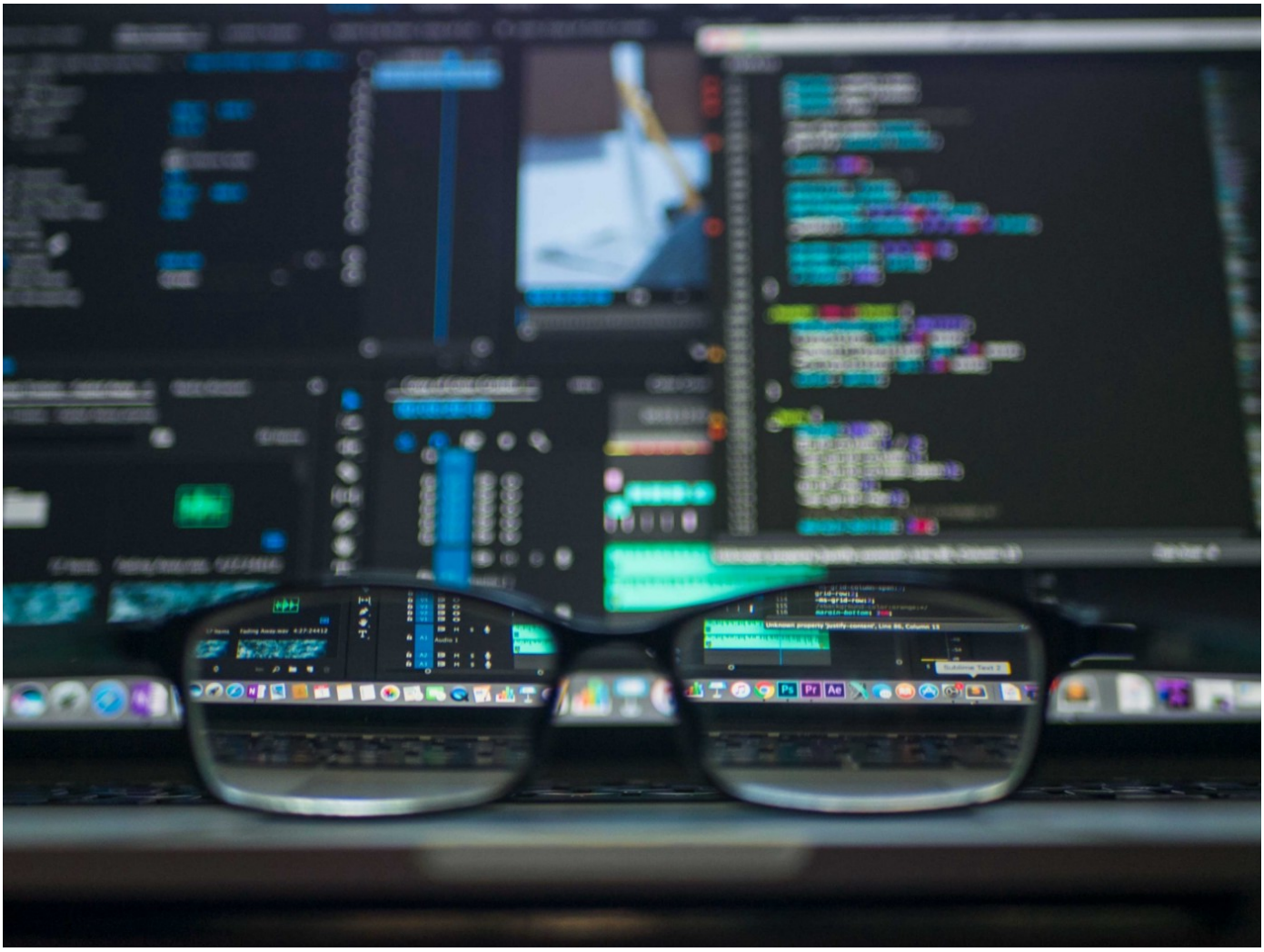
出版于 创业公司

梅格的科技角 跟随

2020 年 8 月 11 日 · 6分钟阅读 · 听

节省

# 亚马逊 DynamoDB 的架构及其性能为何如此之高



DynamoDB 是 Amazon Web Service (AWS) 提供的 NoSQL 数据库。它可以提供极高的性

在本文中，我们将从简要介绍 DynamoDB 的 API 开始。然后我们将研究 DynamoDB 的架构，并详细解释为什么 DynamoDB 可以提供如此高的性能。

从 10,000 英尺的角度来看，DynamoDB 基本上是一个键值存储。它可以被认为是由一些持久存储支持的哈希映射。DynamoDB 支持的两个最重要的操作是 Get 和 Put。

不出所料，Get 和 Put 操作的语义与我们对 hash-map 或 key-value 存储的理解是一致的。例如，我们可以使用 Put 操作将键值对持久化到 DynamoDB，随后的 Get 操作将返回我们之前存储的值。

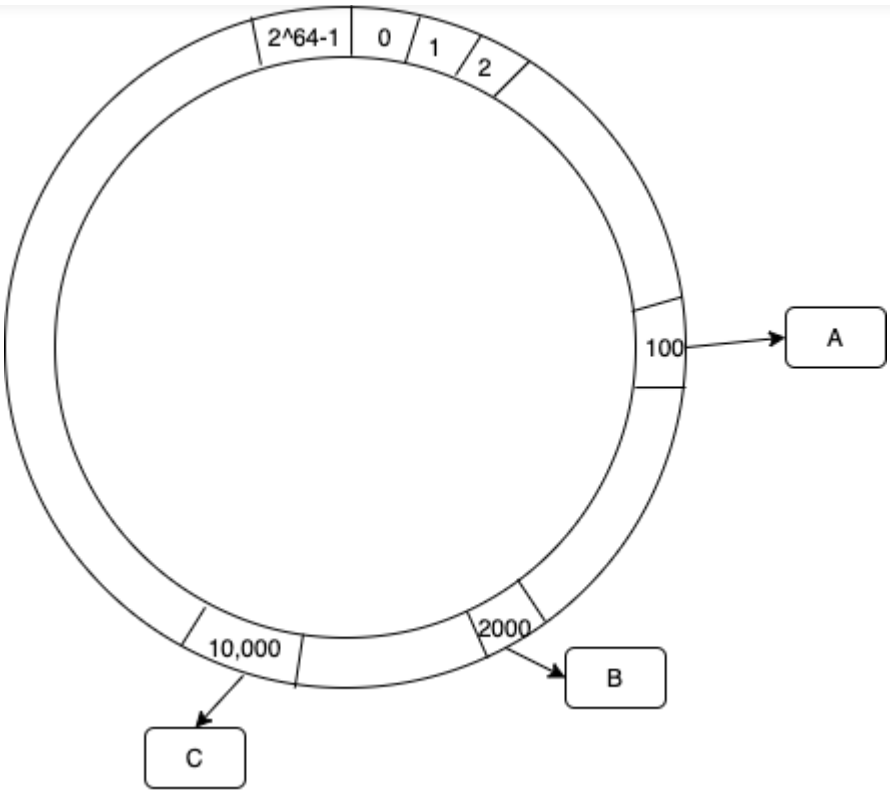
> 放（“键”，“值”）；
>
> 获取（“键”）→返回“值”

DynamoDB 的不同之处在于其极高的性能。在接下来的部分中，我们将仔细研究 DynamoDB 的架构，并了解 DynamoDB 为何如此具有可扩展性。

- 分区

为了实现水平扩展，DynamoDB 将数据分配到由不同机器托管的不同分区。当有更多数据时，DynamoDB 可以创建更多分区并使用更多机器来托管这些额外的分区。它的工作原理如下。

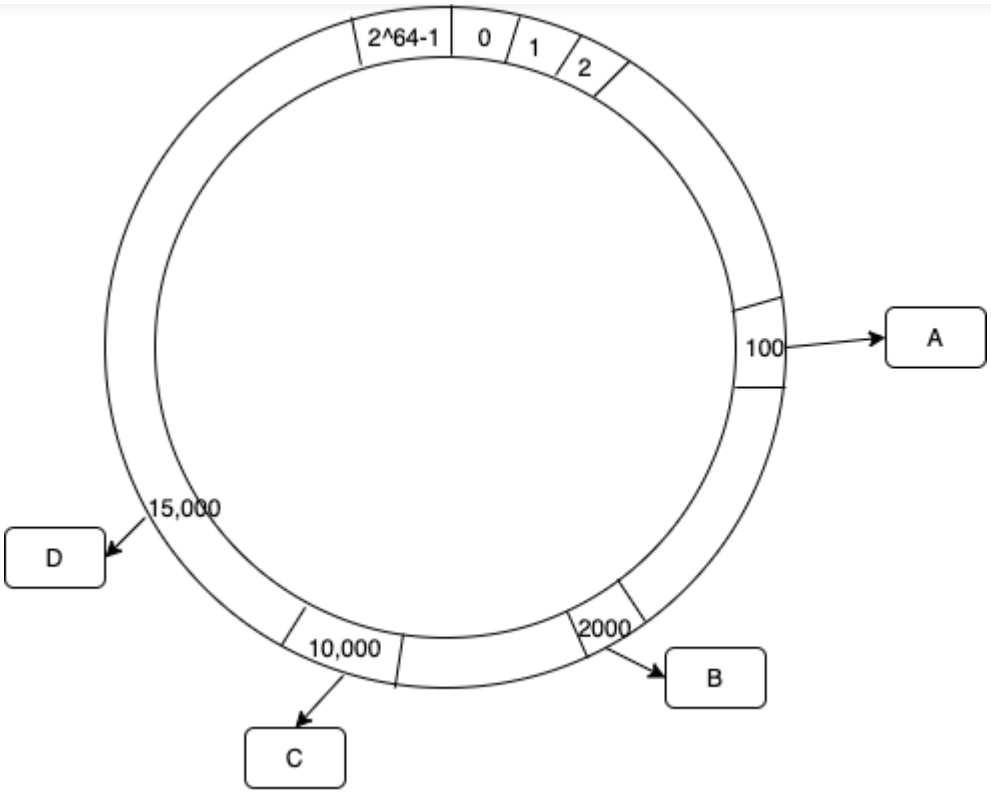
DynamoDB 使用机器集群，每台机器负责将部分数据存储在其本地磁盘中。当一台机器被添加到 DynamoDB 机器集群时，它会被随机分配一个整数值，我们在文章中称之为 token。在我们的示例中，我们假设 DynamoDB 有 3 台机器。A 被分配了一个 100、B 2000 和 C 10,000 的代币。




当我们向 DynamoDB 中插入一个键值对时，该键首先被散列为一个整数 I。然后，该键值对被存储在我们将首先遇到的机器上，如果我们从 I 开始顺时针绕环行走. 因此散列到 $(1000, 2^{64}-1]$ 和 [0, 100] 的键存储在机器 A 上。因此，机器 B 存储散列值在 100 和 2000 之间的键。其余的存储在机器 C



如果我们收到大量哈希值在（10,000, 30,000] 之间的数据，那么机器 A 可能会过载。我们现在可以做的是添加另一台机器 D，并将 A 上存储的数据在 A 和 D 之间拆分。鉴于数据大部分在 (10,000, 30,000] 之间，我们可以为机器 D 分配一个 15,000 的令牌。然后负载或多或少均匀地分布在机器 A 和 D 之间。




当某个范围变热时，即我们收到了大量哈希值在该范围内的数据，我们可以添加更多机器并为这些机器分配该范围内的令牌值。通过这种更改，负载和数据分布在更多机器上。如果数据被删除并且两个连续的范围变冷，我们可以合并它们。例如，如果哈希值在 (100, 2000] 和 (2000, 10,000] 内的数据很少，我们可以从集群中移除机器 B 并使用 C 来托管这些数据。

这种设计至少有两个好处。

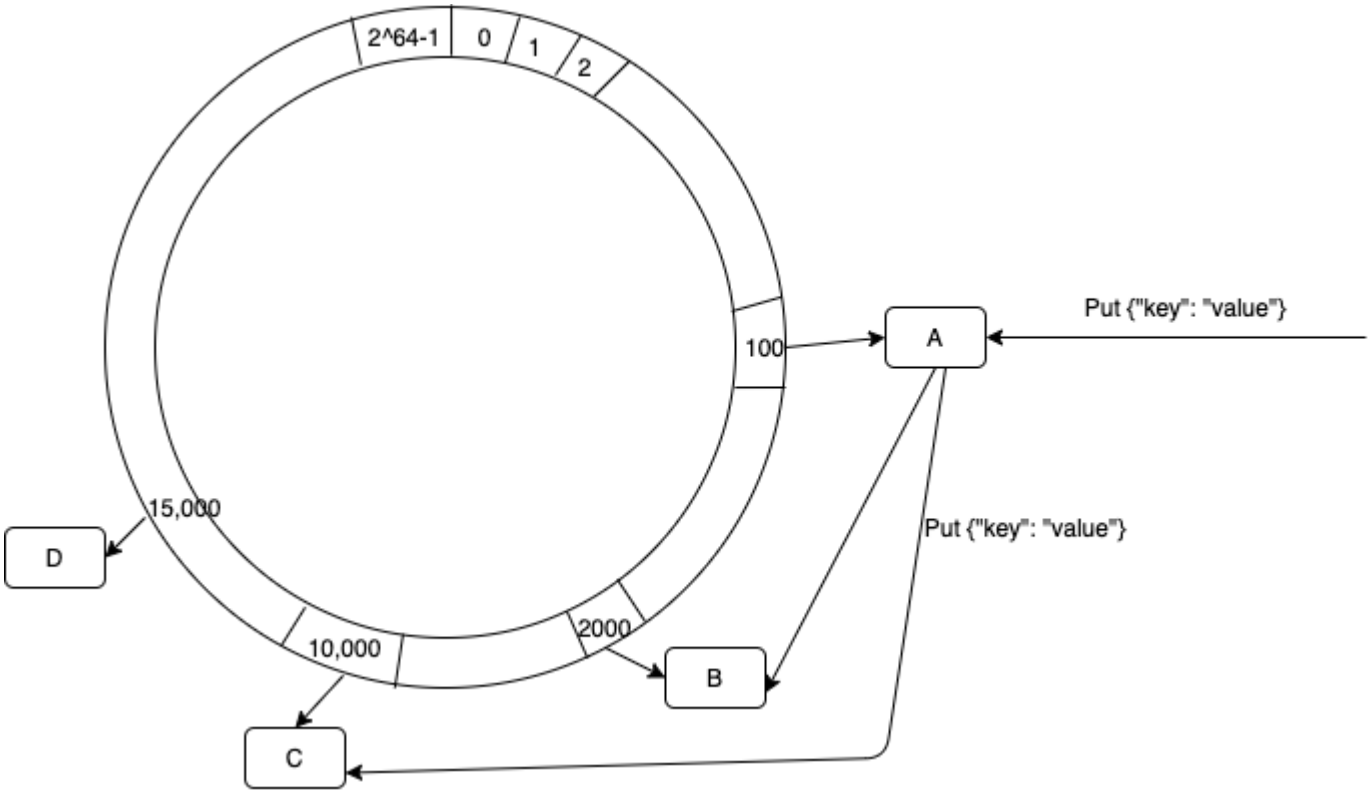
- 首先，基本上没有缩放限制。我们总是可以通过添加更多机器来扩展 DynamoDB 以存储更多数据。
- 其次，这可以实现增量可扩展性。当我们第一次启动 DynamoDB 时，我们不需要分配很多机器。当我们有更多数据时，我们可以添加更多机器。这样可以大大节省资源。

请注意，不同的键可能会散列为相同的值。这很好，因为哈希值只决定哪台机器托管数据。我们根据键而不是其哈希值从该机器检索值。

如果数据只存储在一台机器上，它就不是很可靠也不可用。当机器崩溃或磁盘损坏时，数据就会丢失。当机器停机时，我们无法访问数据。为解决此问题，DynamoDB 将数据复制 N 次。我们将在我们的文章中假设 N=3，这在 AWS 中很常用。

每台机器都完全了解集群内所有机器的令牌。例如，所有机器都知道 A 的令牌值为 100。

当一台机器接收到持久化数据的请求时，它会将请求转发给另外两台拥有接下来两个更大令牌的机器。例如，当机器 A 收到存储 {“key”: “value”} 的请求时，它会首先找出哪两台机器有接下来的两个更大的令牌，在我们的例子中，它是机器 B 和 C。然后向机器 B 和 C 发送持久化键值对的请求。




机器 A 在收到机器 B 和 C 的响应后认为数据是持久的。然后它将响应发送回客户端。但是，这有一个问题。如果没有复制，只有机器 A 慢时请求处理才会慢。现在，当 3 台机器中的任何一台运行缓慢时，处理速度就会很慢。

为了解决这个问题，DynamoDB 不需要在返回客户端之前接收来自所有 N 台机器的响

- 写入 A 内的磁盘成功
- 收到 B 的回复
- 收到 C 的回复

请注意，如果磁盘过载，机器 A 在完成写入本地磁盘之前实际上可以收到来自 B 和 C 的响应。

这种方法减少了写入延迟，但是它引入了另一个问题。您可能无法读取刚刚写入数据库的值！想象一下机器 A 收到 B 和 C 的响应并返回给客户端的情况。但是，写入其本地磁盘实际上失败了。然后当客户端想要获取该值时，A 会返回错误的结果，因为它的本地磁盘没有最近更新的数据。

因此，DynamoDB 进一步要求机器 A 获取 R 个数据副本并将最新的副本返回给客户端。R+W 应该大于 N，我们可以保证用户总是看到最新的数据。R 通常设置为 2。在我们的例子中，机器 A 需要从机器 B 和 C 中的至少一个获取数据的副本，并将最新的值返回给客户端。因此将返回正确的值。

因此，通过分区和复制，DynamoDB 能够提供可扩展且可靠的服务！

> 如果您对 *AWS* 的另一个特色数据库 *AuroraDB* 的架构感兴趣，可以阅读我的另一篇博客 *[点击这里]*。