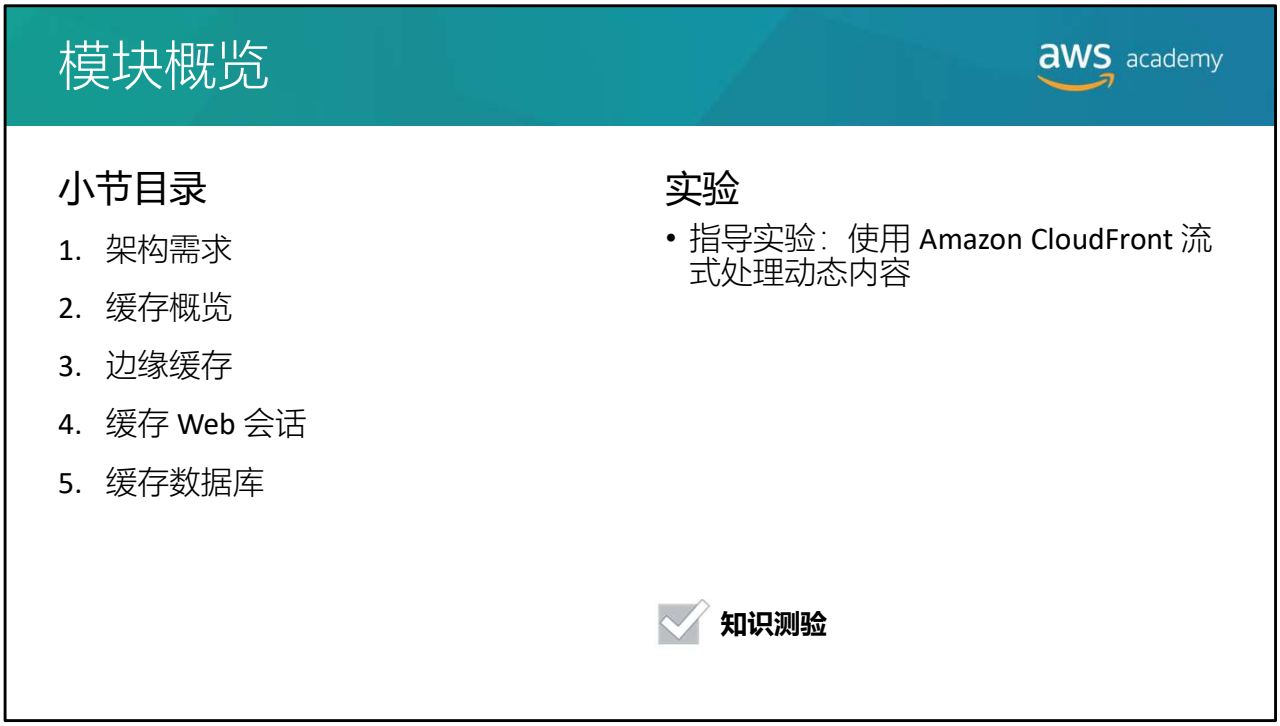
欢迎学习“模块 11：缓存内容”。



本模块包含以下章节：

1. 架构需求
2. 缓存概览
3. 边缘缓存
4. 缓存 Web 会话
5. 缓存数据库

该模块还包括一个指导实验，您将在实验中学习如何使用 Amazon CloudFront 流式传输动态内容。

最后，您需要完成一个知识测验，以测试您对本模块中涵盖的关键概念的理解程度。

模块目标

学完本模块后，您应该能够：

- 确定如何通过缓存内容提升应用程序性能并减少延迟
- 确定如何设计通过边缘站点实现分配和分布式拒绝服务 (DDoS) 保护的架构
- 创建使用 Amazon CloudFront 来缓存内容的架构
- 识别会话管理与缓存的关系
- 描述如何设计使用 Amazon ElastiCache 的架构

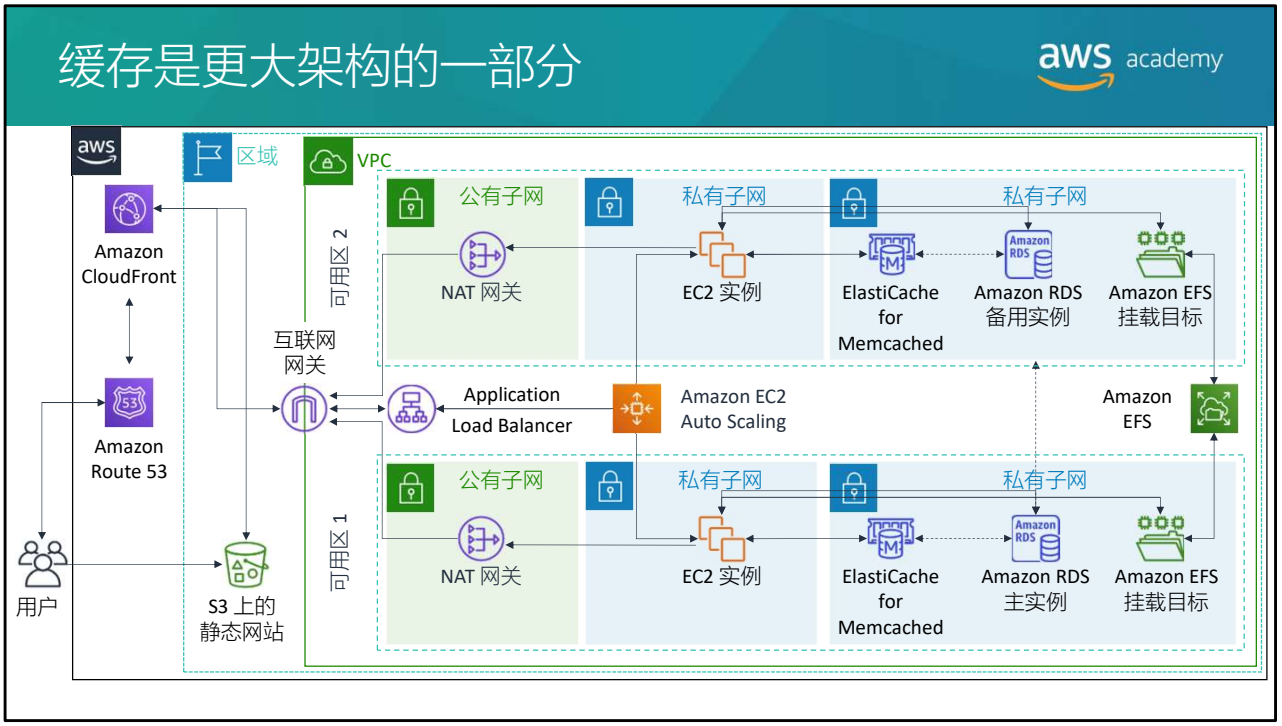
学完本模块后，您应该能够：

- 确定如何通过缓存内容提升应用程序性能并减少延迟
- 确定如何设计通过边缘站点实现分配和分布式拒绝服务 (DDoS) 保护的架构
- 创建使用 Amazon CloudFront 来缓存内容的架构
- 识别会话管理与缓存的关系
- 描述如何设计使用 Amazon ElastiCache 的架构

模块 11：缓存内容

第 1 节：架构需求

介绍第 1 节：架构需求。



在本模块中，您将学习如何在网络环境中实施缓存。本模块将介绍架构图的最后两个组件（Amazon ElastiCache 和 Amazon CloudFront），已在图中展示。本模块还将介绍使用 Amazon DynamoDB 进行数据库缓存。

咖啡馆业务要求

咖啡馆基础设施的容量经常因同等量级的请求而超载。这不仅效率低下，还增加了成本和延迟。

咖啡馆基础设施的容量经常因对静态内容（如菜单项图像）同等量级的请求而超载。这种情况增加了成本和延迟。Sofía 和 Nikhil 希望识别和缓存经常访问的静态内容，以减少延迟并改善客户体验。每次客户加载菜单时，都会从缓存中获取。但是，当菜单项更改时，请求将路由到数据库，并更新缓存。

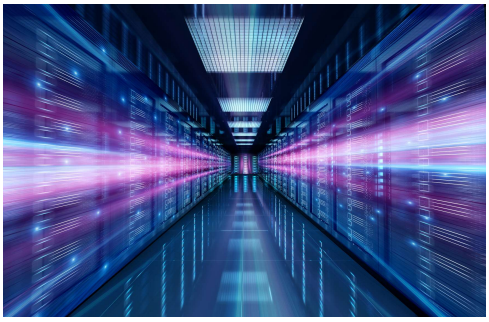
此外，当地名人开始通过视频推荐支持咖啡馆。Sofía 和 Nikhil 必须使用边缘缓存来流式传输数据，以根据加载站点的用户所在的地理位置提供适当且具有区域吸引力的内容。

模块 11：缓存内容

第 2 节：缓存概览

介绍第 2 节：缓存概览。

缓存：牺牲容量换取速度



- 缓存是高速数据存储层
- 缓存能存储数据子集
- 缓存能提高数据检索性能
- 缓存能减少对底层速度较慢的存储层的访问需求

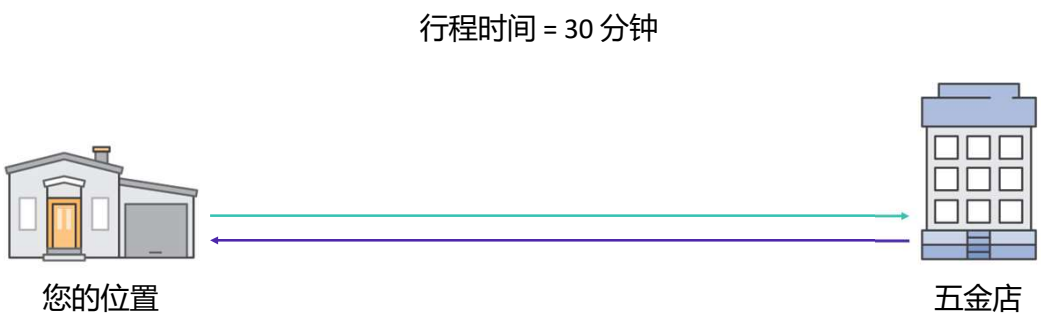
无论您的应用程序是用于提供最新新闻、前十位的排行榜、产品目录还是销售活动门票，它的速度都很重要。应用程序成功与否受内容分发速度的影响。当某人需要数据时，无论是网页数据还是推动业务决策的报告数据，如果此等数据已经缓存，您便能更快地交付数据。

在计算中，*缓存*是高速数据存储层。与通常以完整且持久的形式存储数据的数据库不同，缓存会临时存储数据子集。缓存的主要目的是减少对底层速度较慢的存储层的访问需求，以此来提高数据检索性能。与访问数据主存储位置的请求相比，未来对缓存数据的请求的处理速度更快。

缓存牺牲了容量来换取速度。通过缓存，您可以高效地重新使用之前检索或计算的数据。

缓存中的数据通常存储在快速访问硬件中，例如随机访问存储器 (RAM)。

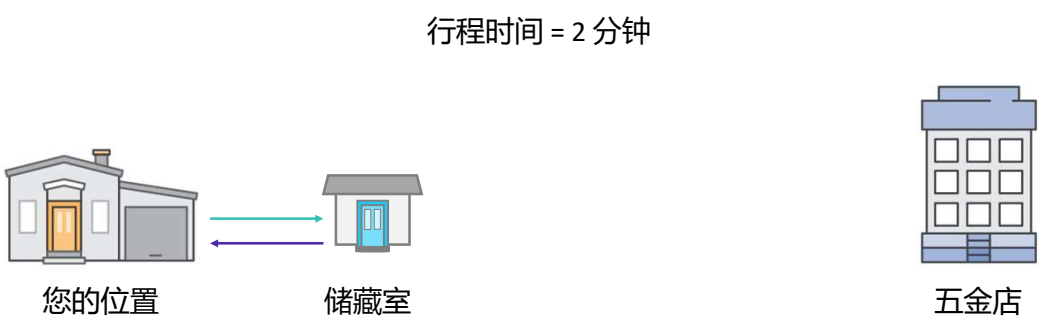
缓存示例 (1/2)



为了说明缓存如何提高性能，不妨考虑前往五金店购物的例子。

如果商店在数英里之外，那么每次需要购物时前往商店都需要花费很大的精力。

缓存示例 (2/2)



您可以将日常使用的物品存放在靠近您家的储藏室内。因此，获取这些物品所需的时间比前往五金店所需的时间少。

但您仍可以前往商店更新物品。

在本例中，储藏室类似于缓存。

为什么要缓存?

需要借助速度慢且成本高的查询才能获取的数据

相对静态且访问频繁的数据，例如社交媒体网站的用户资料

可能在一段时间内过时的信息，如公开交易的股票价格

在决定要缓存的数据时，请考虑以下因素：

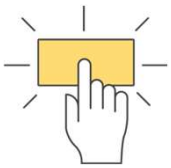
速度和费用 – 耗时的数据库查询和经常使用的复杂查询通常会在应用程序中造成瓶颈。通常，如果要使用速度慢且成本高的查询来获取数据，则可以选择缓存数据。例如，在多个表上执行联接的查询比简单的单个表查询要慢且成本更高。但即使数据所需的查询相对快速且简单，这种数据也可能是缓存的候选项，具体取决于其他因素。

数据和访问模式 – 在确定要缓存的内容时，还需了解数据本身及其访问模式。例如，对返回的搜索结果本质上动态变化程度异常高的网页进行缓存并无意义。为了使缓存更有意义，数据应相对静态且访问频繁，例如社交媒体站点上的个人资料。相反，如果缓存没有任何速度或成本优势，那么您也无需缓存数据。例如，缓存返回搜索结果的网页没有意义，因为这些查询和结果几乎都是唯一的。

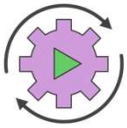
过时 – 根据定义，缓存的数据是过时的数据。即使缓存数据在特定环境中是未过时的，也应始终将其视为过时数据。在确定数据是否需要缓存时，还必须确定应用程序对过时数据的容忍度。您的应用程序可能会在某种情况下容忍过时数据，但其他情况下不行。例如，假设某应用程序在网站上提供公开交易的股票价格。对于该应用程序，如果有免责声明，表示价格可能会延迟长达 *n* 分钟，则短暂延迟是可以接受的。但是，当应用程序必须将股票价格提供给进行买卖的经纪人时，您需要使用实时数据。

缓存的优势

提高应用程序速度



降低响应延迟



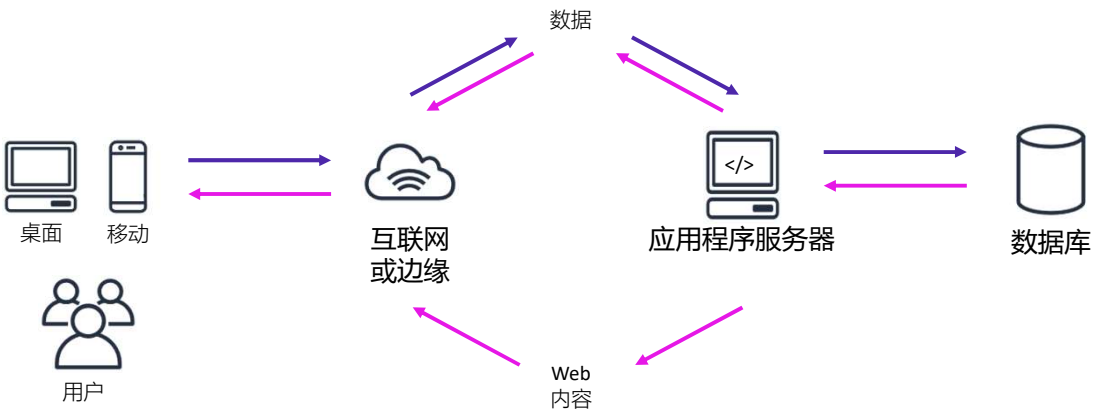
缩短数据库访问时间

缓存通过将数据存储在内存中，实现以高吞吐量、低延迟的方式访问常用的应用程序数据。

缓存可以：

- 提高应用程序的速度。
- 降低用户在应用程序中遇到的响应延迟。
- 缩短应用程序处理时间和读取密集型工作负载（如社交网络、游戏、媒体共享和问答门户）的数据库访问时间。缓存为写入密集型应用程序带来的优势通常并不明显。但即使是写入密集型应用程序，其读/写比率通常也大于 1，这意味着读取缓存仍然是有益的。

在整个数据传输过程中缓存



这个简单的 Web 应用程序架构展示了数据流出和流向用户的情况。您可以在每个层使用缓存来提高应用程序的整体性能和可用性。这些层包括操作系统、网络层（如内容分发网络 (CDN) 和域名系统 (DNS)）、Web 应用程序和数据库。

在这种架构中，您可以使用缓存来：

- 加快从网站检索信息
- 存储域名到 IP 地址的映射
- 加快从 Web 服务器或应用程序服务器检索 Web 内容
- 提升应用程序性能并加快数据访问速度
- 降低与数据库查询请求相关的延迟

在本模块中，您将了解不同的 AWS 服务如何支持在不同的层进行缓存。

第 2 节要点

- 缓存通过将数据存储在内存中，实现以高吞吐量、低延迟的方式访问常用的应用程序数据
- 在决定要缓存哪些数据时，需考虑速度和费用、数据和访问模式以及应用程序对过时数据的容忍度
- 缓存可以在各个技术层中应用和使用，包括操作系统、网络层、Web 应用程序和数据库

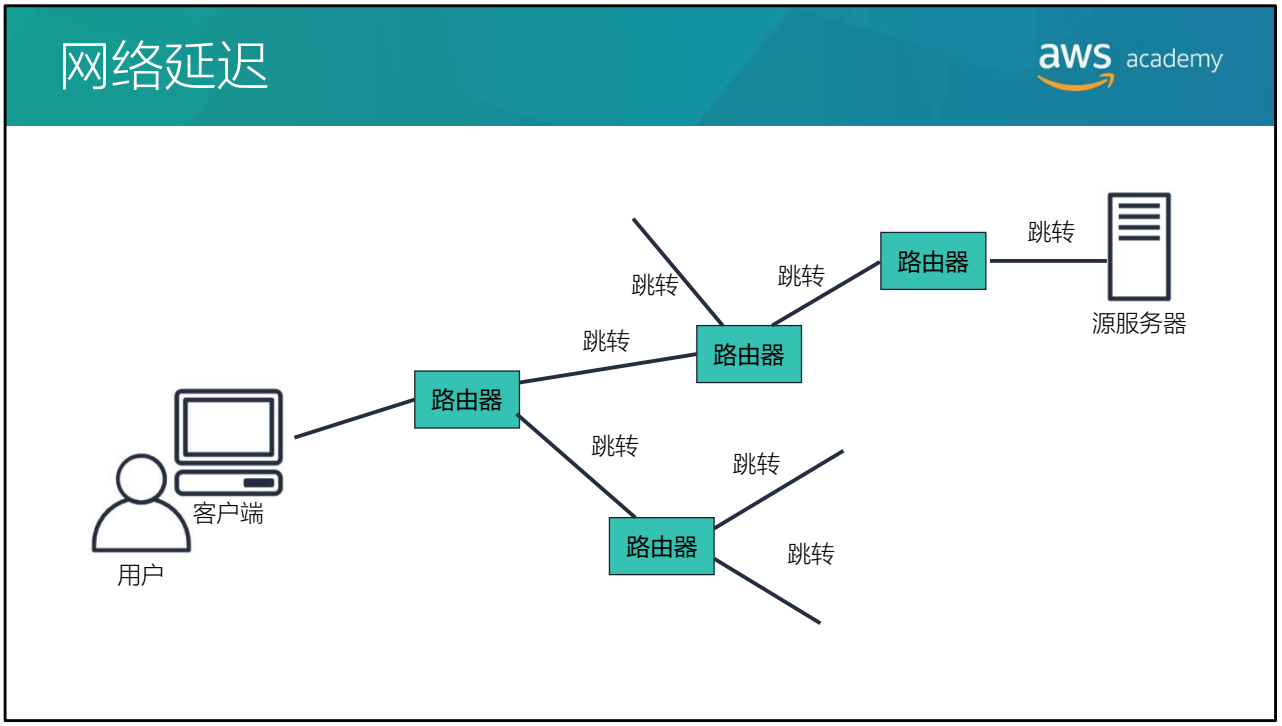
本模块中这节内容的要点包括：

- 缓存通过将数据存储在内存中，实现以高吞吐量、低延迟的方式访问常用的应用程序数据
- 在决定要缓存哪些数据时，需考虑速度和费用、数据和访问模式以及应用程序对过时数据的容忍度
- 缓存可以在各个技术层中应用和使用，包括操作系统、网络层、Web 应用程序和数据库

模块 11：缓存内容

第 3 节：边缘缓存

介绍第 3 节：边缘缓存。



当某人浏览您的网站或使用您的应用程序时，其请求将通过许多不同的网络路由到您的源服务器。源服务器（也称为*源*）存储对象的原始最终版本（例如，Web 对象、图像和媒体文件）。网络跳转次数和请求必须经过的距离会显著影响网站的性能和响应速度。

此外，网络延迟还取决于源服务器的地理位置。如果您的 Web 流量分散在不同的地理位置，那么在全球范围内复制整个基础设施有时并不可行（或经济高效）。在这种情况下，内容分发网络 (CDN) 可能非常有用。

内容分发网络 (CDN)

- 缓存服务器的全球分布式系统
- 缓存经常请求的文件（静态内容）的副本
- 从附近的缓存边缘或接入点提供所请求内容的本地副本
- 提高应用程序性能和扩展性

内容分发网络 (CDN) 是缓存服务器的全球分布式系统。CDN 会缓存应用程序源服务器上托管的常见请求文件的副本。这些文件可以包含静态内容，例如 HTML、CSS、JavaScript、图像和视频文件。CDN 从缓存边缘或接入点 (PoP) 分发所请求内容的本地副本，从而以最快的速度向请求者分发内容。

有关使用 CDN 进行缓存的更多信息，请参阅内容分发网络 (CDN) 缓存。

Amazon CloudFront

- 为 Amazon 全球 CDN
- 针对具有多层缓存（默认）和广泛灵活性的所有分发使用案例进行了优化
- 为您的架构提供一层额外的安全保护
- 支持 WebSocket 和 HTTP 或 HTTPS 方法

Amazon CloudFront 是一项全球 CDN 服务，可加速向用户分发内容。此类内容可能是静态和动态内容、使用 HTTP 或 HTTPS 的媒体文件以及流媒体视频（包括视频点播和直播）。 与其他 AWS 服务一样，CloudFront 是一种按使用量付费的自助服务，不需要长期承诺，也无最低消费。

默认情况下，CDN 提供多层缓存。区域性边缘缓存可在对象尚未缓存在边缘时减少延迟并降低源服务器上的负载。此外，CDN 还提供多种媒体流式传输选项，包括预先录制的文件和直播活动。它可以为全球受众提供 4K 传输所需的持续高吞吐量。

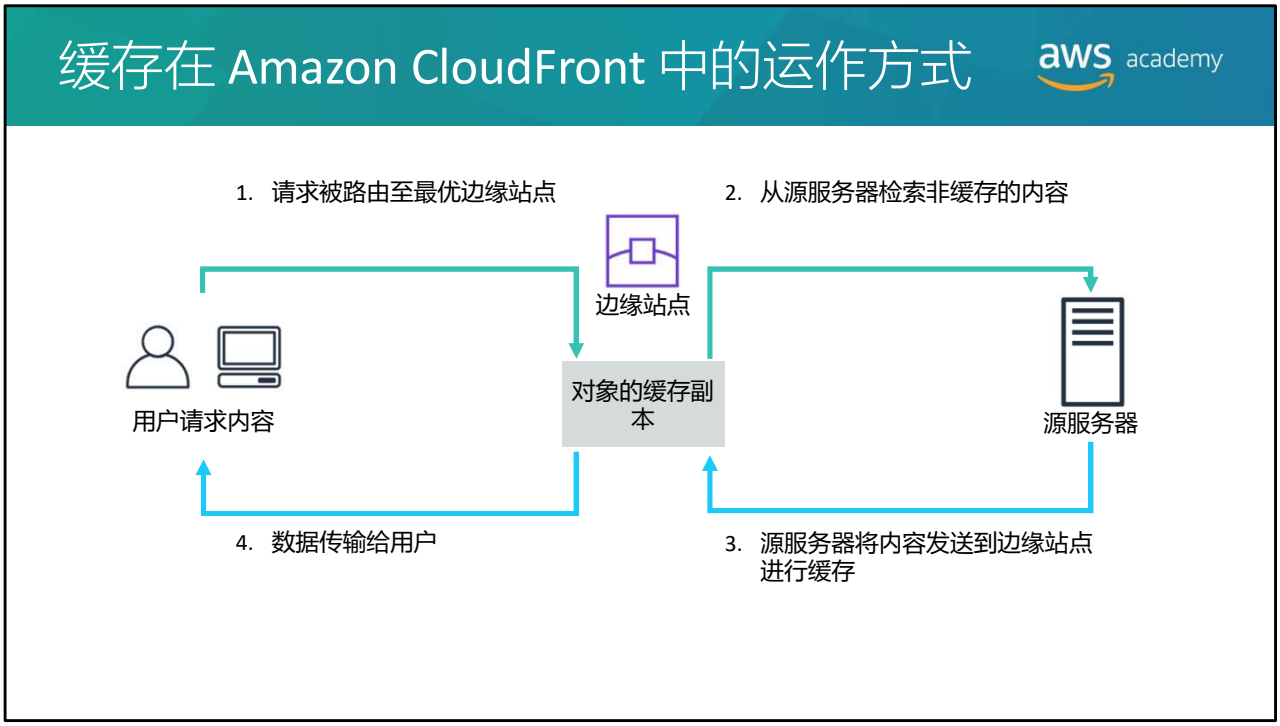
CloudFront 提供网络级别和应用程序级别保护。您的流量和应用程序可从 AWS Shield Standard 等各种内置保护功能受益，无需额外付费。您还可以使用可配置的功能（例如 AWS Certificate Manager (ACM)）来创建和管理自定义 SSL 证书而无需支付额外费用。CloudFront 支持安全套接字层/传输层安全性 (SSL/TLS) 协议。

CloudFront 支持使用 WebSocket 协议进行实时双向通信。这种持久连接使客户端和服务器可以互相发送实时数据，避免了重复打开连接的开销。这对聊天、协作、游戏和金融贸易等通信应用程序尤其有用。CloudFront 还支持 HTTP 方法（DELETE、GET、HEAD、OPTIONS、PATCH POST、PUT），这些方法可提高动态网站的性能。这些网站具有 Web 表单、评论和登录框、加入购物车按钮以及其他上传用户数据的功能。因此，您可以使用单个域名通过 CloudFront 交付整个网站，从而加快网站各部分的下载和上传速度。

该 Amazon.com 网页示例展示了静态和动态内容如何构成动态 Web 应用程序。该 Web 应用程序通过 HTTPS 协议提供，用于对用户页面请求和从 Web 服务器返回的页面进行加密。您可以使用 CDN 或边缘缓存来缓存静态内容。此类内容可能包括 Web 对象（例如，HTML 文档、CSS 样式表或 JavaScript 文件）、图像文件和视频文件。

您无法缓存动态生成的内容或用户生成的数据。但是，您可以将 CloudFront 配置为从自定义源上运行的应用程序提供这些信息。例如，它可能是 EC2 实例或 Web 服务器。

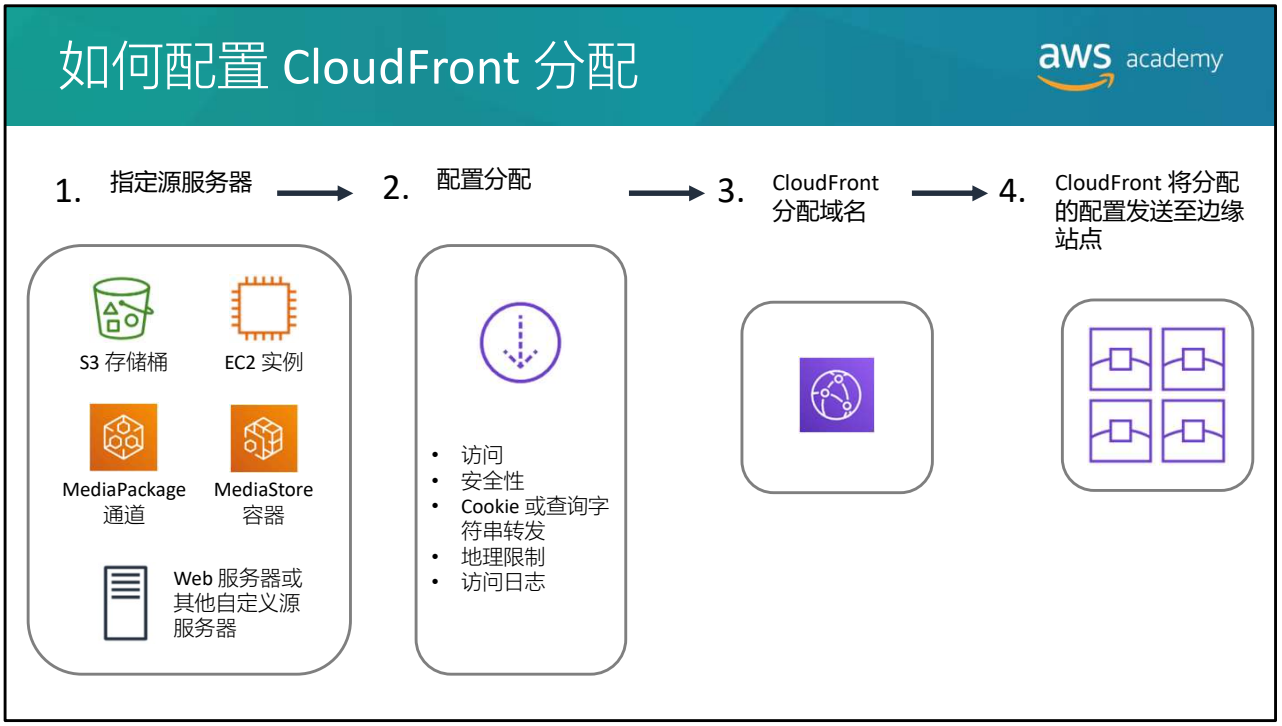
此外，您可以将 CloudFront 配置为要求查看器使用 HTTPS 请求您的对象，以便在 CloudFront 与查看器通信时加密连接。您也可以将 CloudFront 配置为使用 HTTPS 从源获取对象，以便在 CloudFront 与源通信时加密连接。



Amazon CloudFront 通过全球数据中心网络（称作*边缘站点*）分发内容。

当用户请求通过 CloudFront 提供的内容时，DNS 会将请求路由到最能满足该请求需要的边缘站点。通常，它是最近的边缘站点，具有最低的延迟。CloudFront 在缓存中检查请求内容（第 1 步）。如果内容在缓存中，CloudFront 会立即将其分发给用户（第 4 步）。内容当前可能不在缓存中。如果不在，CloudFront 会将请求转发到您标识为内容最终版本来源的源服务器（第 2 步）。源服务器将内容发回边缘站点（第 3 步），然后 CloudFront 将内容转发给用户（第 4 步）。它还将内容添加到边缘站点的缓存中，以备下次有人请求这些内容。

如果对象热度下降，各个边缘站点可以移除这些对象，从而为更热门的内容腾出空间。对于热度下降的内容，CloudFront 使用*区域性边缘缓存*。区域性边缘缓存是在全球范围内部署且靠近您的受众的 CloudFront 站点。它们位于源服务器和全球边缘站点之间，直接向受众提供内容。区域性边缘缓存比单个边缘站点的缓存空间更大，因此对象在区域性边缘缓存中保留的时间更长。这种安排有助于让更多的内容更靠近受众。它减少了 CloudFront 需要返回源服务器的次数，为受众提升了整体性能。



如果您想使用 CloudFront 来分配内容，需要创建*分配*。

1. 您可以指定托管文件的源服务器。您的源服务器可以是 S3 存储桶、AWS Elemental MediaPackage 通道、AWS Elemental MediaStore 容器或自定义源服务器。例如，自定义源服务器可以是 EC2 实例或您自己的 Web 服务器。

2. 然后，指定有关如何跟踪和管理内容分发的详细信息。例如，您可以指定是希望文件对所有人都可用还是仅供特定用户使用。您还可以指定是否希望 CloudFront 执行以下功能：创建显示用户活动的访问日志、将 Cookie 或查询字符串转发到源服务器或要求用户使用 HTTPS 访问您的内容。

3. CloudFront 将域名分配到您的新分配。

4. CloudFront 会将您的分配配置发送至所有的边缘站点，但不会发送内容。

如何使内容过期

- 生存时间 (TTL) –
  - 固定时间（有效期）
  - 由您设置
  - 从 CloudFront 到源服务器的 GET 请求使用 If-Modified-Since 标头
- 更改对象名称 –
  - Header-v1.jpg 改为 Header-v2.jpg
  - 新名称强制立即刷新
- 使对象失效 –
  - 不得已的手段：低效且成本高昂

您可以通过三种方式使缓存内容过期：

- 生存时间 (TTL) – 使用这种方法，在 CloudFront 将另一个请求转发到您的源服务器之前，您可以控制文件在 CloudFront 缓存中保留的时间。减少持续时间让您可以提供动态内容。增加持续时间意味着您的用户将获得更高的性能，因为直接从边缘缓存提供文件的可能性更大。较长的持续时间还会减轻源服务器的负载。如果您将特定源服务器的 TTL 设置为 *0*，CloudFront 仍会从该源服务器缓存内容。然后它将使用 If-Modified-Since 标头发出 GET 请求。因此，如果缓存的内容在源服务器中未发生更改，源服务器可以发出信号，指示 CloudFront 可以继续使用缓存内容。如果不需要立即替换，TTL 是一种很好的方法。

- 更改对象名称 – 此方法需要更多的工作，但可以立即替换。尽管您*可以*在 CloudFront 分配中更新现有对象，并使用相同的对象名称，但不建议这样做。仅当请求对象时（而不是当您在源服务器中放入新的对象或更新的对象时），CloudFront 才会将对象分配到边缘站点。例如，您可以使用具有相同名称的更新版本来更新源服务器中的现有对象。在这种情况下，在两个列出的事件发生之后，边缘站点才会从您的源服务器获取新版本。

- 使对象失效 – 这种方法不是一个好的解决方案，因为系统必须强制与所有边缘站点交互。您应该谨慎使用此方法，并且只能用于个别对象。

有关如何使缓存内容过期的详细信息，请参阅管理内容在边缘缓存中保留的时长（有效期）。

正如您所学到的，您可以使用 CloudFront 来分发流媒体视频，包括视频点播和直播流。

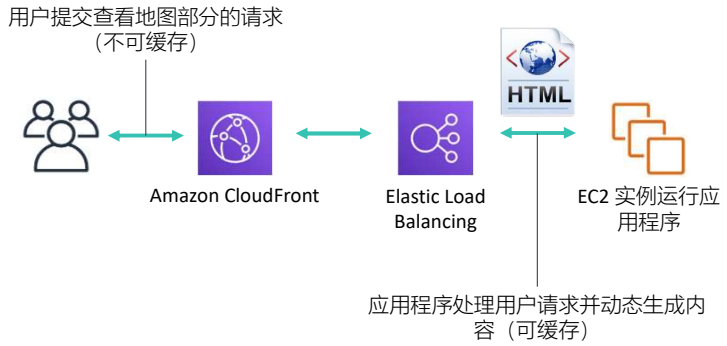
对于点播视频流，必须先使用编码器对视频内容进行格式设置和打包，然后 CloudFront 才能分配视频内容。例如，可以使用 AWS Elemental MediaConvert 和 Amazon Elastic Transcoder 编码器。打包过程会创建*分段*，这些分段是静态文件，其中包含音频、视频和字幕内容。它还会生成清单文件，这些文件描述了要播放哪些分段，以及播放它们的具体顺序。打包格式包括基于 HTTP 的动态自适应流（DASH 或 MPEG-DASH）、Apple HTTP Live Streaming (HLS)、Microsoft Smooth Streaming 以及通用媒体应用程序格式 (CMAF)。

将视频转换为输出格式后，您将转换后的内容托管在 S3 存储桶，也即您的源服务器中。然后，您可以使用 CloudFront 将分段文件分发给世界各地的用户。

有关使用 CloudFront 进行视频流式传输的更多信息，请参阅使用 CloudFront 进行视频点播和视频直播。

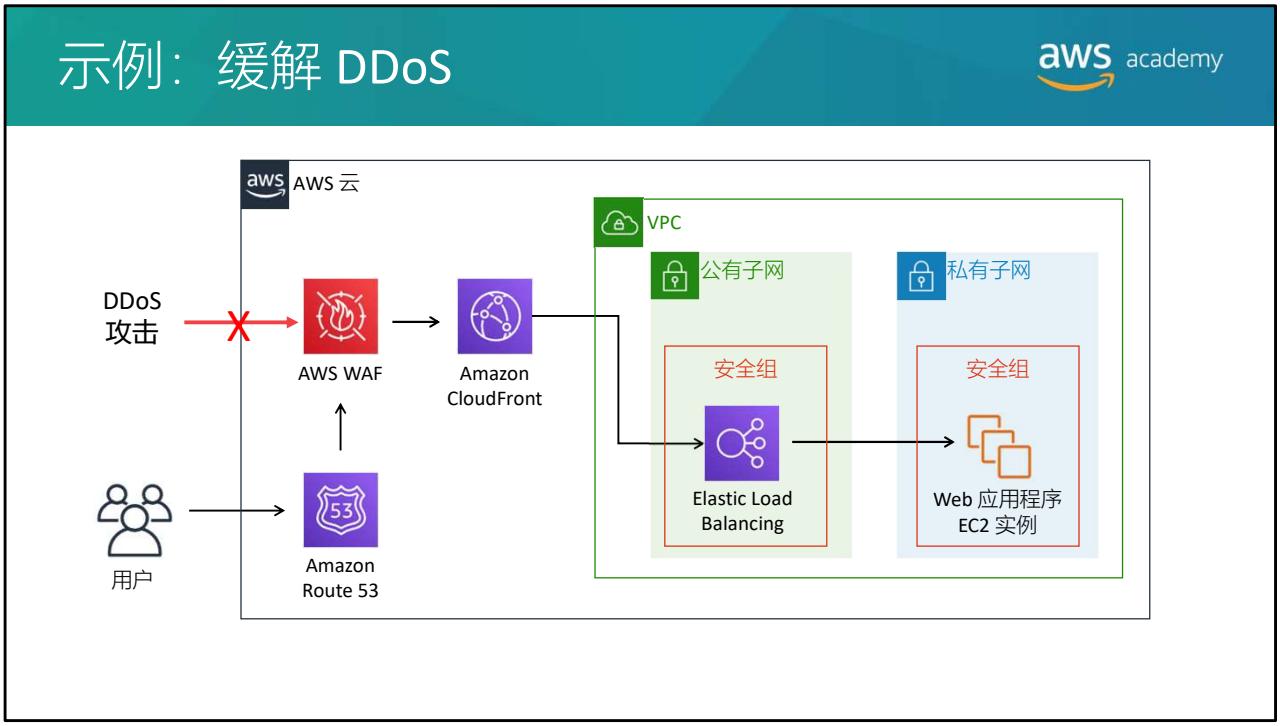
示例：动态生成的内容

使用案例：地图图块
问题：需要更快的数据库响应速度



通常，您只缓存静态内容。但是，您可以让内容*看起来*是静态的（因为它的 URL），但在首次需要它时，它是动态构建的。当内容可重复使用时，这种动态构建可能很有用，但创建成本可能很高，而且可能不经常更改。

地图图块是比较典型的例子。在此类例子中，您可以缓存人们经常查看的地点（如主要城市），而不是偏远地区之类的地方。生成所有可能的图块组合非常昂贵和浪费，因为大多数图块几乎永远都不需要。每个图块 URL 的路径部分都可以包含生成图块所需的参数。如果图块已存在于特定的 CloudFront 边缘站点，则直接提供图块。否则，将生成图块并返回边缘站点，以满足将来的请求。



您可以使用 CloudFront 提高在 AWS 上运行的应用程序在遭受分布式拒绝服务 (DDoS) 攻击后的恢复能力。DDoS 攻击是蓄意为之，意在使您的网站或应用程序无法供用户使用，例如，通过网络流量使其泛洪。为了实现这一目标，攻击者使用多个来源编排针对目标的攻击。这些来源可能包括受恶意软件感染的计算机、路由器、物联网 (IoT) 设备和其他终端节点的分布式组。

以下示例显示了一个弹性架构，该架构有助于防止或缓解 DDoS 攻击。

DNS 服务（如 Amazon Route 53）可以有效地将用户的请求连接到 CloudFront 分配。然后，CloudFront 分配会将动态内容请求代理到托管应用程序终端节点的基础设施。对通过 CloudFront 路由的 Route 53 DNS 请求和后续应用程序流量进行内联检查。Route 53 和 CloudFront 都内置了始终开启的监控、异常检测和针对常见基础设施 DDoS 攻击的缓解措施。

常见的基础设施攻击包括同步/确认 (SYN/ACK) 泛洪、用户数据报协议 (UDP) 泛洪和反射攻击。当超过 SYN 泛洪攻击阈值时，系统会激活 SYN Cookie 以避免丢弃来自合法客户端的连接。确定性数据包筛选将丢弃格式错误的 TCP 数据包和无效的 DNS 请求，并仅在流量适用于服务时才允许流量通过。启发式异常检测可评估流量的类型、来源和组成等属性。在多个维度对流量进行评分，只有最可疑的流量才会被丢弃。

该方法使您能够在保护应用程序可用性的同时避免误报。Route 53 还可以抵御 DNS 查询泛洪。DNS 查询泛洪实际是可持续数小时并尝试耗尽 DNS 服务器资源的 DNS 请求。Route 53 使用随机分片和 Anycast 条带化将 DNS 流量分布到边缘站点，并保护服务的可用性。

AWS WAF 是 Web 应用程序防火墙。它允许您监控转发到 Amazon API Gateway API、Amazon CloudFront 或 Application Load Balancer 的 HTTP 和 HTTPS 请求。AWS WAF 还可让您控制对内容的访问。例如，您可以指定条件，例如请求来自的 IP 地址或查询字符串的值。基于这些条件，API Gateway、CloudFront 或 Application Load Balancer 以请求的内容或 HTTP 403 状态代码（禁止）进行响应。您还可以将 CloudFront 配置为在请求被阻止时返回自定义错误页面。

如需深入了解如何提高在 AWS 上运行的应用程序抵御 DDoS 攻击的弹性，请参阅以下资源：

- 实现 DDoS 弹性的 AWS 最佳实践 AWS 白皮书
- How to Help Protect Dynamic Web Applications Against DDoS Attacks by Using Amazon CloudFront and Amazon Route 53 AWS 安全性博客文章

第 3 节要点

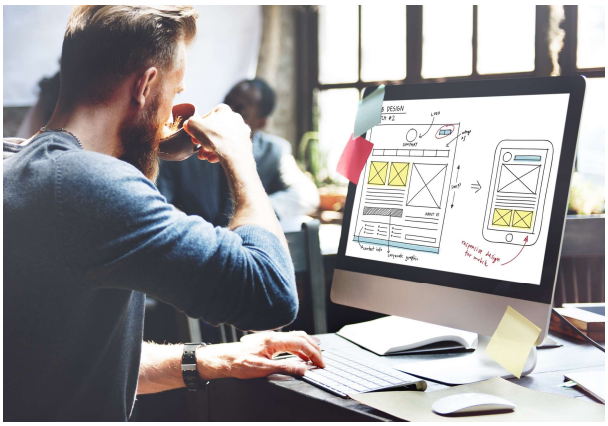
aws academy

- Amazon CloudFront 是一项 全球 CDN服务，可加速向用户分发内容（包括静态内容和视频），无最低使用承诺。
- CloudFront 使用由边缘站点和区域边缘缓存 组成的全球网络向用户分发内容。
- 要使用 CloudFront 分发内容，请指定源服务器 并配置 CloudFront 分配。CloudFront 将分配域名，并将您的分配配置发送至其所有边缘站点。
- 您可以使用 Amazon CloudFront 提高在 AWS 上运行的应用程序在遭受 DDoS 攻击后的恢复能力 。

本模块中这节内容的要点包括：

- Amazon CloudFront 是一项全球 CDN 服务，可加速向用户分发内容（包括静态内容和视频），无最低使用承诺。
- CloudFront 使用由边缘站点和区域边缘缓存组成的全球网络向用户分发内容。
- 要使用 CloudFront 分发内容，请指定源服务器并配置 CloudFront 分配。CloudFront 将分配域名，并将您的分配配置发送至其所有边缘站点。
- 您可以使用 Amazon CloudFront 提高在 AWS 上运行的应用程序在遭受 DDoS 攻击后的恢复能力。

模块 11 – 指导实验：使用 Amazon CloudFront 流式处理动态内容



您现在将完成“模块 11 – 指导实验：使用 Amazon CloudFront 流式处理动态内容”。

指导实验：场景

在此实验中，您将使用 Amazon Elastic Transcoder 将源视频转换为多比特率。然后，使用 Amazon CloudFront 通过 Apple HTTP Live Streaming (HLS) 协议将动态多比特率流传输到连接的设备。

Amazon Elastic Transcoder

Amazon CloudFront

在此实验中，您将使用 Amazon Elastic Transcoder 将源视频转换为多比特率。然后，使用 Amazon CloudFront 通过 Apple HTTP Live Streaming (HLS) 协议将动态多比特率流传输到连接的设备。此流可以在任何支持 HLS 协议的浏览器上播放。

Apple HLS 可通过普通 Web 服务器动态调整影片播放质量，以匹配有线或无线网络的适用速度。这一过程通过创建不同质量的流实现。每个流再分解为不同的块，按顺序传输到客户端设备。在客户端，您可以选择不同比特率的流，以使流式传输会话能够适应不同的网络速度。

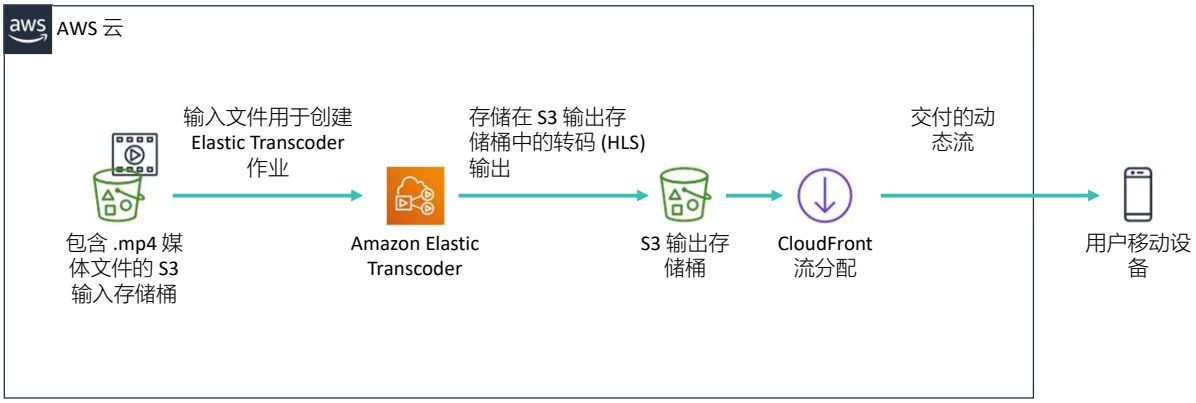
指导实验：任务

1. 创建 Amazon CloudFront 分配
2. 创建 Amazon Elastic Transcoder 管道
3. 测试动态（多比特率）流的播放

在本指导实验中，您将完成以下任务：

1. 创建 Amazon CloudFront 分配
2. 创建 Amazon Elastic Transcoder 管道
3. 测试动态（多比特率）流的播放

指导实验：最终产品



该图总结了您完成实验后将会构建的内容。

现在可以开始指导实验了。

指导实验总结：要点

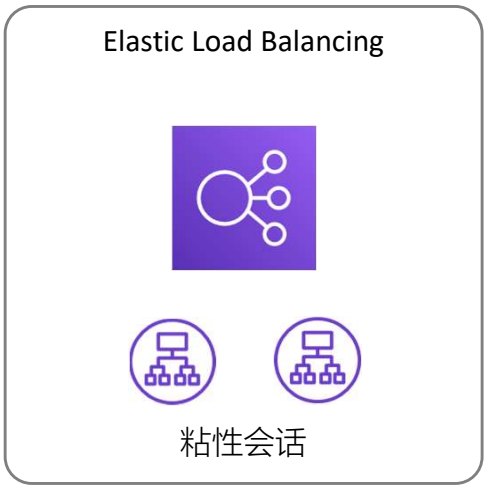
完成这个指导实验之后，您的讲师可能会带您讨论此指导实验的要点。

模块 11：缓存内容

第 4 节：缓存 Web 会话

介绍第 4 节：缓存 Web 会话。

会话管理：粘性会话



使负载均衡器能够将请求路由到管理用户会话的特定服务器的功能。

- 使用客户端 Cookie
- 经济高效
- 加快会话检索速度
- 存在缺点 –
  - 实例失败时会话丢失
  - 限制可扩展性：负载分配不均衡和延迟增加

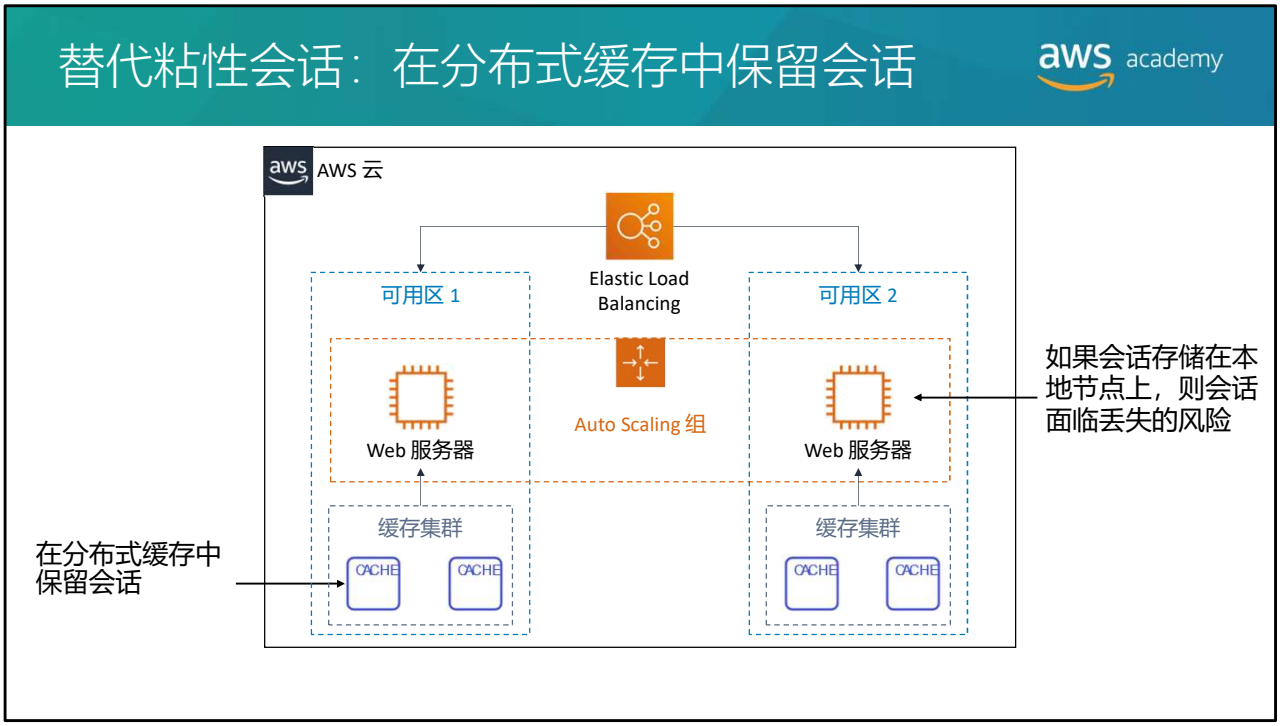
当用户或服务与 Web 应用程序交互时，它会发送 HTTP 请求，应用程序返回响应。这样的一系列事务称为*会话*。每个请求都独立于之前的事务。因此，会话用于管理用户身份验证，并在用户与应用程序交互时存储用户数据。例如，通过使用会话，用户不需要为他们向服务器发出的每个请求发送凭证。

您可以通过多种方式管理用户会话。（默认情况下，负载均衡器会将每个请求单独路由到负载最小的注册实例。）要使用粘性会话，客户端必须支持 Cookie。

粘性会话具有成本效益，因为会话存储在运行应用程序的 Web 服务器上。因此，粘性会话可以消除网络延迟并加快这些会话的检索速度。但是，如果实例发生故障，您可能会丢失存储在该实例上的会话。

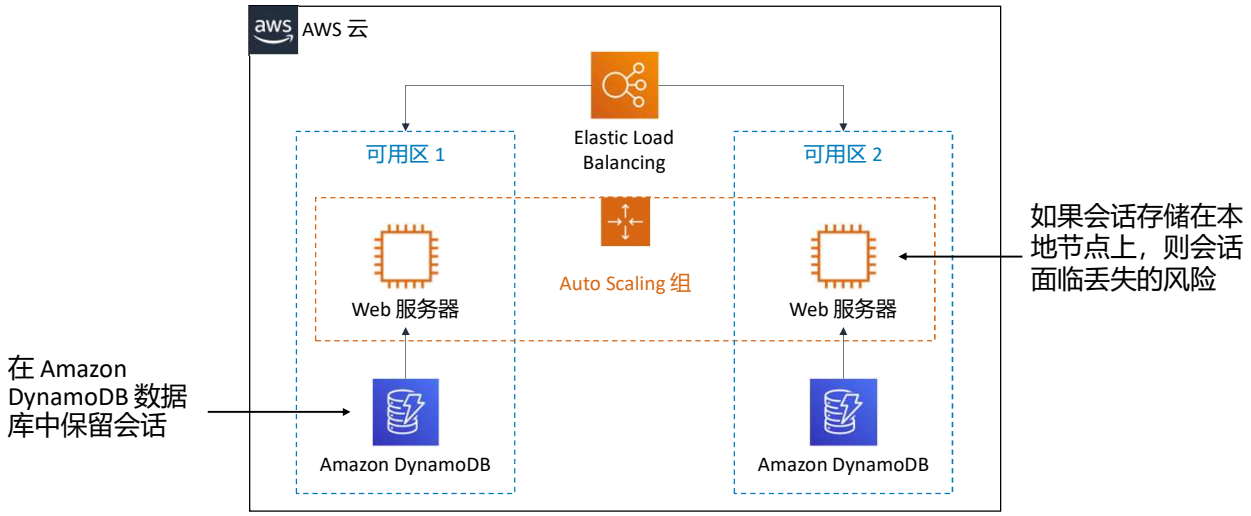
粘性会话的另一个缺点是它们可能会限制应用程序的可扩展性。使用粘性会话，负载均衡器无法在每次收到来自客户端的请求时真正均衡负载。粘性会话会强制负载均衡器将所有请求发送到创建会话状态的源服务器。如果该服务器过载，那么接收如此多的请求可能会导致服务器之间的负载不均衡，并影响用户响应时间。

替代粘性会话：在分布式缓存中保留会话



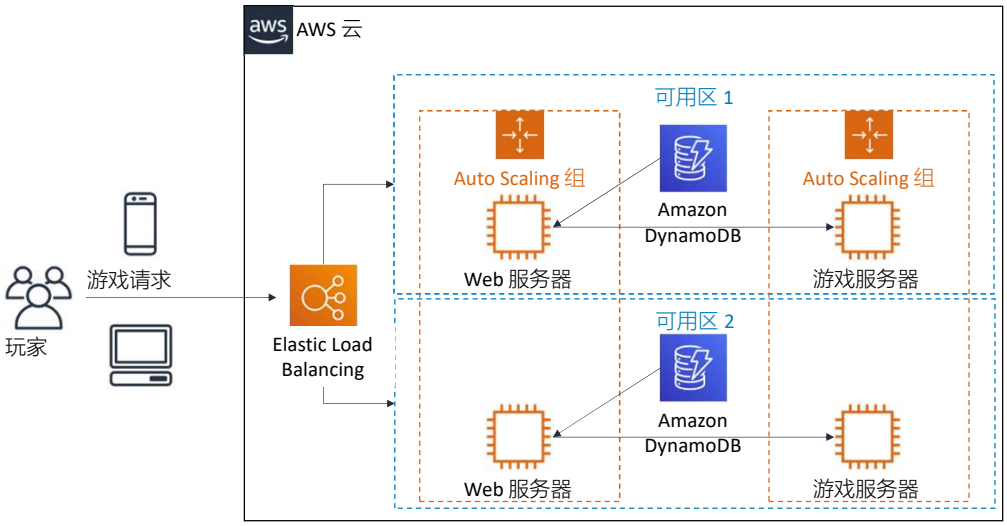
您可以在架构中指定一个层，在实例外部以可扩展和可靠的方式存储这些会话，而不是使用粘性会话。 一种选择是将会话数据保存在分布式缓存中，如该架构图所示。当 Web 服务器数量发生变化以适应负载，并且您不希望冒丢失会话的风险时，在动态环境中实施此架构是有意义的。

替代粘性会话：在 DynamoDB 表中保留会话



另一种选择是将会话数据保存在 Amazon DynamoDB 数据库中，如该图所示。通过 Amazon DynamoDB，您可以设置扩缩策略，并根据需要扩展和缩减 DynamoDB 容量。

示例：存储在线游戏应用程序的会话状态



该架构可能支持在线游戏应用程序，在这种应用程序中，必须要快速检索会话。随着玩家收集物品、击败敌人、获得金币、解锁关卡以及完成成就，游戏数据将持续更新。必须将每个会话事件写入数据库层，以免丢失。游戏制作者将会话历史记录和其他面向时间的数据存储在 DynamoDB 中，以便按玩家、日期和时间快速查找。

每个 DynamoDB 数据库表都与吞吐容量关联。您可以指定每秒 1000 次写入，DynamoDB 会在后台扩缩数据库。随着需求的变化，您可以更新容量，然后，Amazon DynamoDB 会根据需要重新分配资源。这种弹性对游戏开发商颇有助益：如果您的游戏越来越受欢迎，您的玩家数量可能会突然从几千名扩展到数百万名。如果需要，您可以快速轻松地缩小规模。

DynamoDB 可以在任何规模上保持可预测的低延迟性能，如果您的游戏发展到数百万对延迟敏感的客户，这一点至关重要。您不会花时间优化 DynamoDB 的性能。

要查看包含 DynamoDB 的类似游戏架构，请参阅此 AWS 大数据博客文章。

第 4 节要点

aws academy

- 会话用于管理用户身份验证，并在用户与应用程序交互时存储用户数据。
- 您可以使用粘性会话管理会话，这是 Elastic Load Balancing 负载均衡器的一项功能。粘性会话将请求路由到管理用户会话的特定服务器。
- 您还可以通过在 Web 服务器实例外部保留会话数据来管理会话，例如，保留在分布式缓存或 DynamoDB 表中。

本模块中这节内容的要点包括：

- 会话用于管理用户身份验证，并在用户与应用程序交互时存储用户数据。
- 您可以使用粘性会话管理会话，这是 Elastic Load Balancing 负载均衡器的一项功能。粘性会话将请求路由到管理用户会话的特定服务器。
- 您还可以通过在 Web 服务器实例外部保留会话数据来管理会话，例如，保留在分布式缓存或 DynamoDB 表中。

模块 11：缓存内容

第 5 节：缓存数据库

介绍第 5 节：缓存数据库。

应该在什么时候缓存数据库？

您很担忧对客户的响应时间。

大量的请求让您的数据库难以招架。

您希望降低数据库成本。

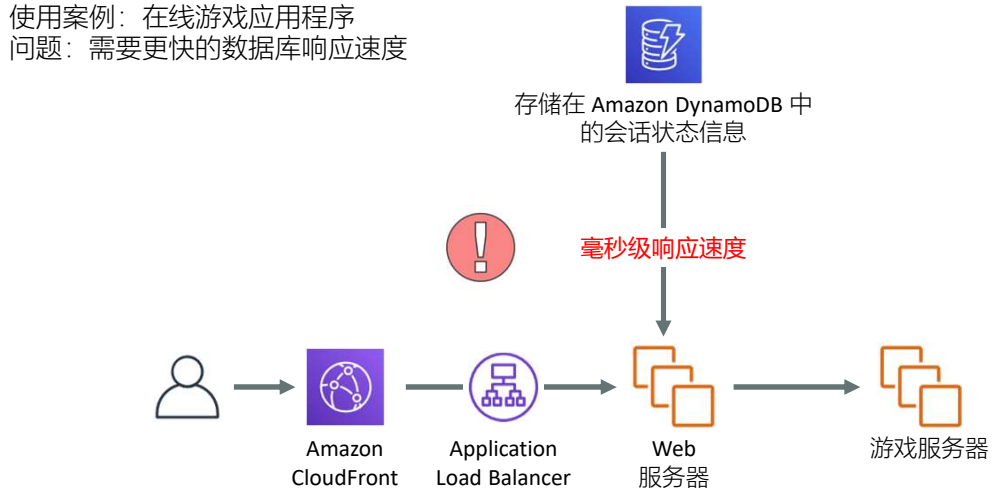
正如您在本模块前面所学到的，耗时的数据库查询和复杂的查询会在应用程序中造成瓶颈。

在以下情况下，应考虑缓存数据库：

- 您担心对客户的响应时间。您可能需要加速延迟敏感型工作负载。缓存有助于提高吞吐量并减少数据检索延迟，从而提高应用程序的性能。
- 大量的请求让您的数据库难以招架。您可能拥有大量流量，因而无法获得该等工作负载所需的吞吐量。在数据库旁边放置缓存层可以提高吞吐量并实现更高的性能。
- 希望降低数据库成本。无论数据是分布在基于磁盘的 NoSQL 数据库中还是在关系数据库中纵向扩展，通过扩展来实现高读取可能成本高昂。可能需要多个数据库只读副本才能匹配单个内存中缓存节点每秒的请求数。

数据库缓存可消除主数据库面临的不必要压力（通常是频繁访问的读取数据），以此为其提供助益。缓存本身可存在于数据库、应用程序等许多区域中，也可作为独立层存在。

使用 DynamoDB 获取状态信息



假设有一个在线游戏应用程序架构的简化版本，您在 DynamoDB 中存储会话状态信息。在某些情况下，您可能会发现毫秒级的响应速度对于您的应用程序来说不够快。数据库缓存可以帮助解决这个问题。

Amazon DynamoDB Accelerator

适用于 DynamoDB 的完全托管、高度可用的内存中缓存

- 极致的性能（微秒级响应时间）
- 高度可扩展
- 完全托管
- 与 DynamoDB 集成
- 灵活
- 安全

Amazon DynamoDB Accelerator (DAX) 是适用于 DynamoDB 的完全托管且高度可用的内存中缓存。即使每秒数百万个请求，它仍可提供高达 10 倍的性能提升（从毫秒到微秒）。DAX 将承担向 DynamoDB 表添加内存中加速所需的所有繁重工作。您无需管理缓存失效、数据填充或集群管理。

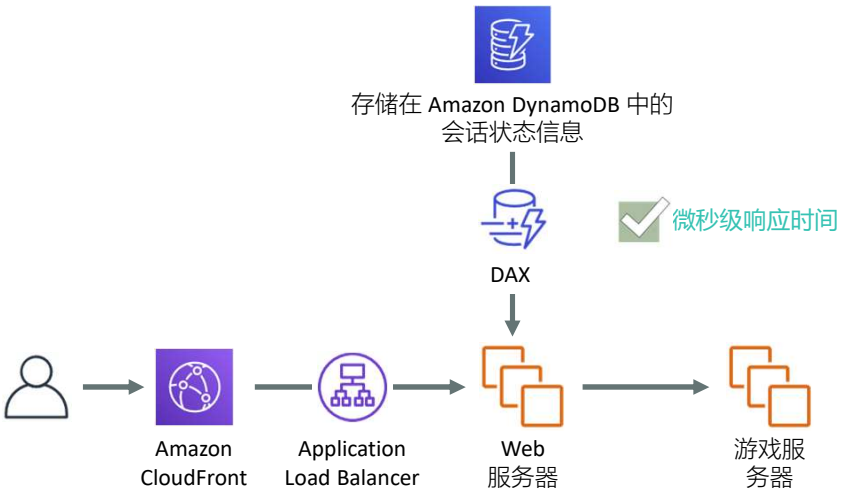
DAX 具有以下优势：

- **极致的性能** – DynamoDB 提供一致的个位数毫秒延迟。当 DynamoDB 和 DAX 一起使用时，对于每秒数百万个请求的读取密集型工作负载，可实现微秒级的响应速度。
- **高度可扩展** – DAX 具有按需扩展功能。您可以先使用三节点 DAX 集群，并根据需要添加容量，扩展到十节点集群。
- **完全托管** – 和 DynamoDB 一样，DAX 为完全托管。DAX 负责处理管理任务，包括预置、设置和配置、软件修补以及在扩展操作期间通过节点复制数据。DAX 可自动处理常见的管理任务，如故障检测、故障恢复和软件修补。
- **与 DynamoDB 集成** – DAX 的 API 与 DynamoDB 兼容，无需更改任何功能型应用程序代码。预置 DAX 集群，使用 DAX 客户端开发工具包 (SDK) 指向 DAX 集群中的现有 DynamoDB API 调用。由 DAX 负责处理其余操作。
- **灵活** – 您可以为多个 DynamoDB 表预置一个 DAX 集群，或者为单个 DynamoDB 表预置多个 DAX 集群，也可以两种预置方法结合使用。
- **安全** – DAX 与多项 AWS 服务完全集成，增强了安全性。您可以使用 AWS Identity and Access Management (IAM) 为每个用户分配唯一的安全凭证，并控制每个用户对服务和资源的访问。使用 Amazon CloudWatch，您可以全面了解资源使用率、应用程序性能和运行状况。通过与 AWS CloudTrail 集成，您可以轻松记录和审计集群配置的更改。DAX 支持 Amazon Virtual Private Cloud (Amazon VPC)，可从现有应用程序安全轻松地访问。标记可让您进一步了解 DAX 集群，帮助您进行相应管理。

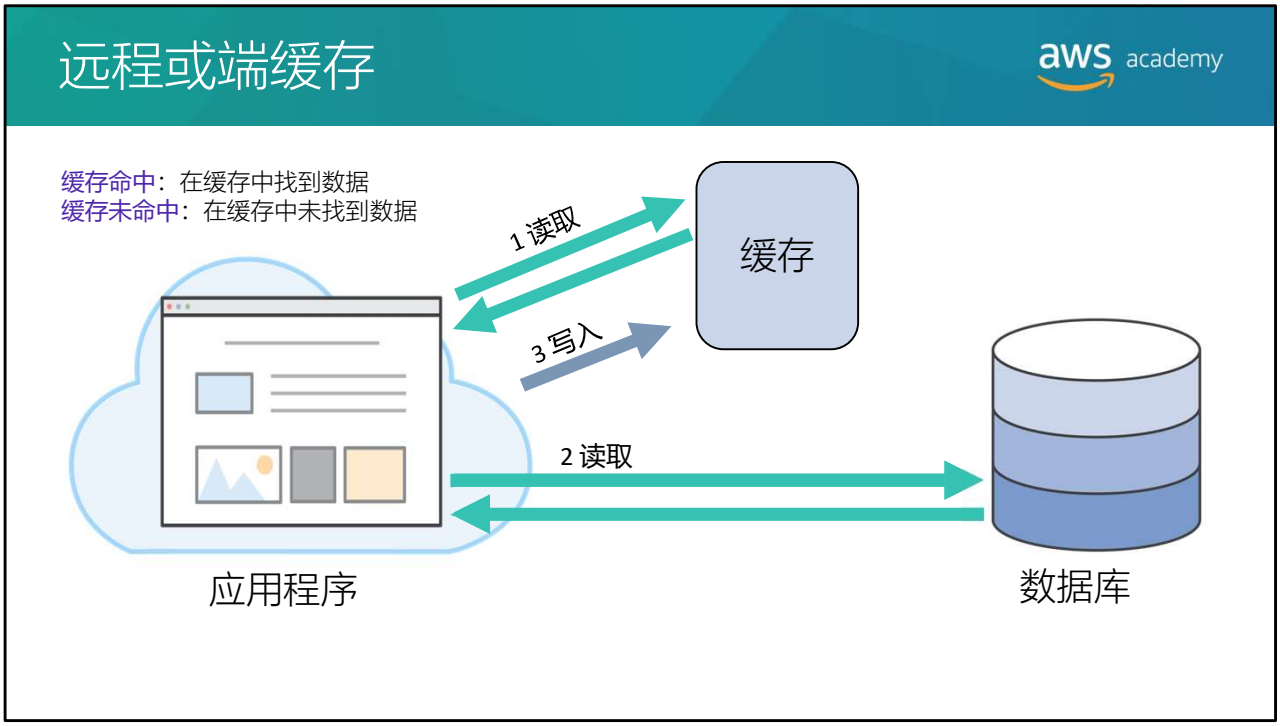
对缓存数据的检索减少了现有 DynamoDB 表的读取负载。因此，它可能会减少预置读取容量，降低总体运营成本。

有关 DAX 的更多信息，请参阅 AWS 数据库博客文章。

将 DynamoDB 与 DAX 结合使用以加快响应速度



在游戏示例中，通过将 DAX 添加到架构中，无需在游戏代码中进行任何重大更改，即可而获得加速。通过这种方式，可以简化对架构的部署。您必须要做的一点是使用指向 DAX 的新终端节点重新初始化 DynamoDB 客户端。无需对代码的其余部分进行任何更改。DAX 会处理缓存失效和数据填充，无需您进行任何干预。当您举办可能导致玩家数量激增的活动时，缓存有助于加快响应速度。此类活动的一个例子是季节性可下载内容 (DLC) 产品或发布新的补丁。



DAX 是透明的缓存。实施数据库缓存部署的另一种方法是使用远程缓存或端缓存。端缓存不直接连接到数据库，而是与数据库相邻使用。端缓存通常基于键值 NoSQL 存储（如 Redis 或 Memcached）构建。其中的每个缓存节点每秒可提供数十万到百万个请求。

端缓存通常用于读取密集型工作负载。其工作方式如下：

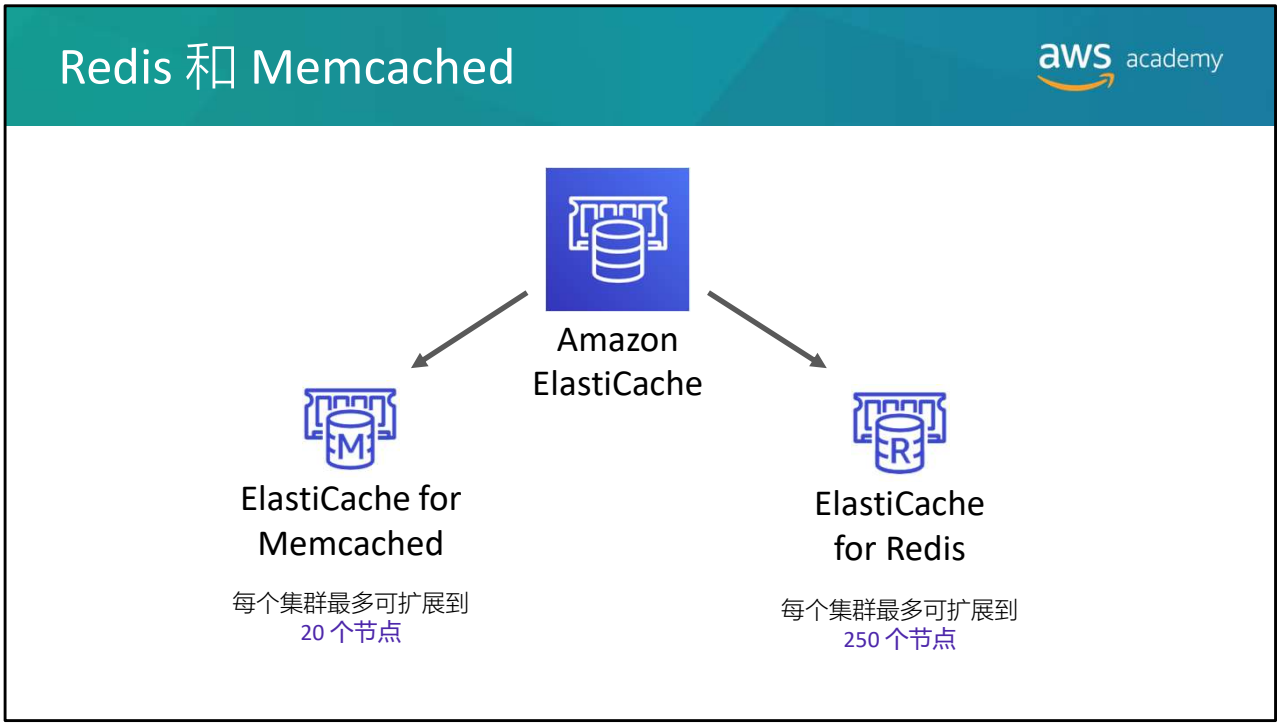
1. 对于给定的键值对，应用程序首先尝试从缓存中读取数据。如果缓存包含相应数据（称为*缓存命中*），则返回相关值。
2. 如果在缓存中找不到所需的键值对（称为*缓存未命中*），应用程序会从底层数据库中获取数据。
3. 当应用程序再次需要数据时，数据必须存在，这一点非常重要。为了确保这一点，从数据库获取的键值对随后将写入缓存。

Amazon ElastiCache

ElastiCache 为 Web 应用程序提供云中内存数据存储。

- 使用内存中数据存储和缓存
- 提供高性能
- 完全托管
- 可扩展
- 支持 Redis 和 Memcached

Amazon ElastiCache 是可以用作内存中数据存储的端缓存，支持要求最严苛且需要亚毫秒级响应速度的应用程序。借助 ElastiCache，您无需执行硬件预置、软件修补、设置、配置、监控、故障恢复和备份等管理任务。ElastiCache 会持续监控您的集群以保证工作负载正常运行，使您可以专注于价值更高的应用程序开发工作。Amazon ElastiCache 支持横向扩展、缩减及纵向扩展，可满足不断变化的应用程序需求。通过分片实现写入扩展和内存扩展。副本提供读取扩展。ElastiCache 支持两种开源的内存中数据库：Redis 和 Memcached。



ElastiCache for Memcached 最多可以扩展到每个集群 20 个节点。相比之下，ElastiCache for Redis 可以扩展到 250 个节点，从而提高数据访问性能。ElastiCache 支持 Amazon VPC，使您能够将集群隔离在您为节点选择的 IP 范围内。

ElastiCache 在高度可靠的基础设施上运行，其可靠性与其他 AWS 服务使用的基础设施相同。ElastiCache for Redis 可通过具有自动故障转移功能的多可用区部署实现高可用性。对于 Memcached 工作负载，数据将在集群中的所有节点之间进行分区。因此，当需求增长时，您可以横向扩展以更好地处理更多数据。

Memcached 与 Redis 比较

特征	Memcached	Redis
亚毫秒级延迟	是	是
能够横向扩展以进行写入和存储	是	否
多线程性能	是	否
高级数据结构	否	是
对数据集进行排序和排名	否	是
发布/订阅消息收发服务	否	是
具有自动故障转移功能的多可用区部署	否	是
持久性	否	是

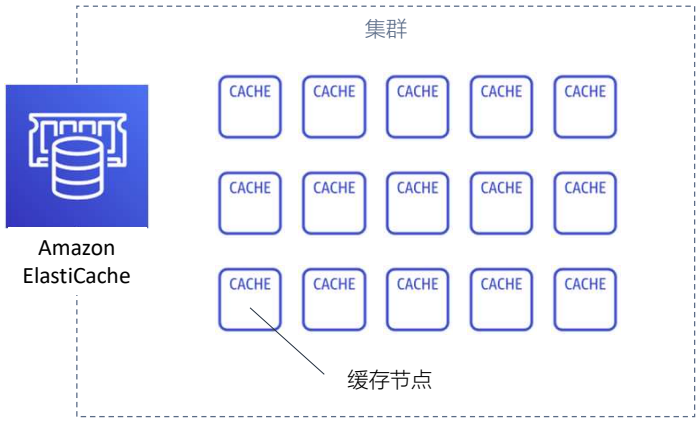
Memcached 和 Redis 引擎是简单的缓存，可用于减轻数据库负担。每种引擎都具有某些优势。此表比较了 Memcached 和 Redis 之间的一些关键特征。

- **亚毫秒级延迟** – 两种引擎都提供亚毫秒级响应速度。相较基于磁盘的数据库，将数据存储在内存中可以更快地读取数据。
- **横向扩缩能力** – Memcached 使您能够随着系统需求的增加和减少而扩展和缩减，从而添加和删除节点。
- **多线程性能** – 由于 Memcached 是多主题的，它可以使用多个处理内核，这意味着您可以通过纵向扩展计算容量来处理更多操作。
- **高级数据结构** – Redis 支持复杂的数据类型，如字符串、哈希、列表、集、排序集和位图。
- **对数据集进行排序或排名** – 您可以使用 Redis 对内存中数据集进行排序或排名。例如，您可以使用 Redis 排序集来实施游戏排行榜，用于保留按排名排序的玩家列表。
- **发布/订阅消息** – Redis 支持具有模式匹配功能的发布/订阅消息服务，可将其用于高性能聊天室、实时评论流、社交媒体源和服务器间通信。

- **具有自动故障转移功能的多可用区部署** – ElastiCache for Redis 通过具有自动故障转移功能的多可用区部署提供高可用性，以防主节点发生故障。

- **持久性** – Redis 使您能够持续保留密钥存储。相比之下，Memcached 引擎不支持持久性。例如，节点出现故障并被新的空节点替换；或者终止节点或缩小节点。在这种情况下，您将丢失存储在缓存内存中的数据。

ElastiCache 组件

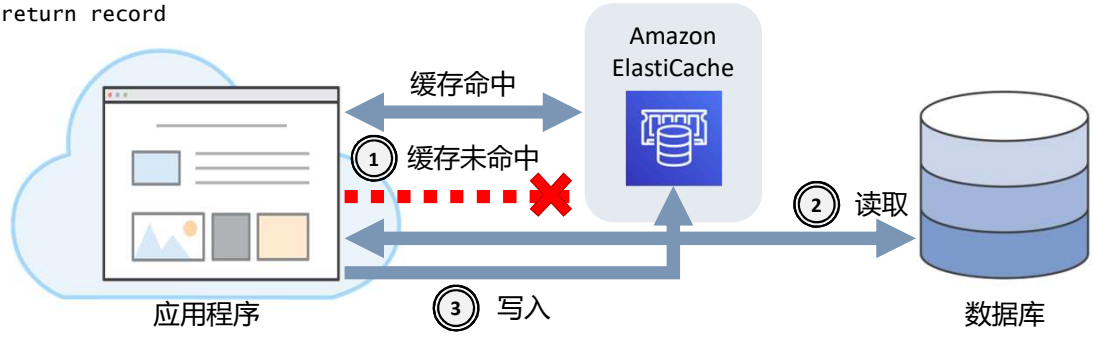


- 节点是 ElastiCache 部署的最小数据块
- 每个节点都有自己的 DNS 名称和端口
- 集群是一个或多个节点的逻辑分组

*缓存节点是 ElastiCache 部署的最小构建块。*它是固定大小、与网络连接的安全 RAM 区块。每个节点都运行创建或最后一次修改集群或复制组时选择的引擎。每个节点都有自己的 DNS 名称和端口。它可独立于其他节点存在，也可以与其他节点组成分组（也称为*集群*）。

缓存策略：延迟加载

```
def get_user(user_id):
    # Check the cache
    record = cache.get(user_id)
    if record is None:
        # Run a DB query
        record = db.query("select * from users where id = ?", user_id)
        # Populate the cache
        cache.set(user_id, record)
    return record
```



您可以使用 ElastiCache 部署两种缓存策略。

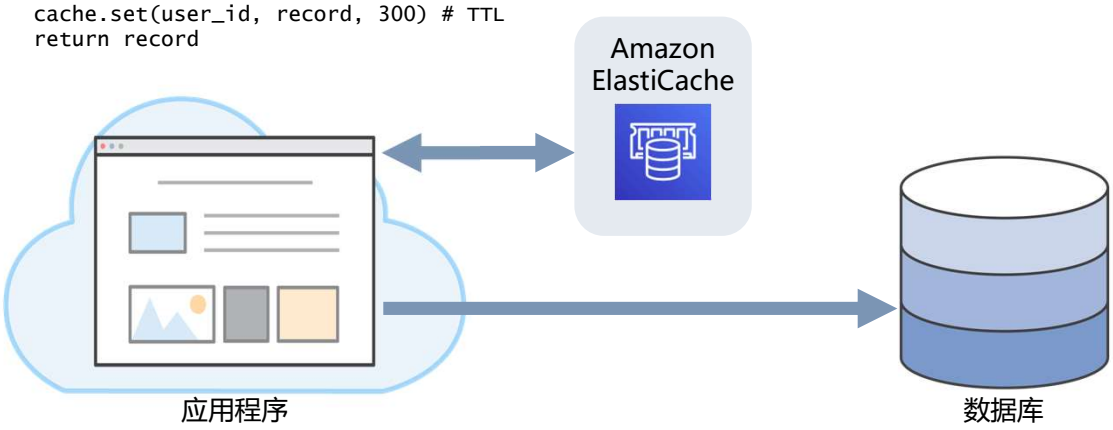
第一种策略是*延迟加载*，这是一种缓存策略，仅在必要时才将数据加载到缓存中。在此情况下，当应用程序请求数据时，它会先向 ElastiCache 缓存发出请求。如果数据存在于缓存中且为最新状态，则发生缓存命中，ElastiCache 会将数据返回给您的应用程序。否则（如果缓存未命中），您的应用程序会从数据存储中请求数据，然后将数据返回给应用程序。然后，您的应用程序会将数据写入缓存，以便在下次请求时更快地检索到。

当您拥有经常读取但很少写入的数据时，请使用延迟加载。例如，在典型的 Web 或移动应用程序中，用户个人资料很少发生变化，但需要在整个应用程序中访问。一个人每年可能只对自己的个人资料进行几次更新。但是，根据用户的不同，个人资料每天可能会被访问数十次或数百次。

延迟加载的优点是只缓存请求的数据。由于大部分数据从未被请求，所以延迟加载避免了向缓存中填入不必要的数据。但是，缓存未命中会带来不利影响。每次出现缓存未命中都会造成三次往返，这可能会导致数据在到达应用程序时出现明显延迟。另外，如果仅在存在缓存未命中时将数据写入缓存，则缓存中的数据可能会过时。当数据库中的数据发生更改时，延迟加载不会向缓存提供更新。

缓存策略：直写

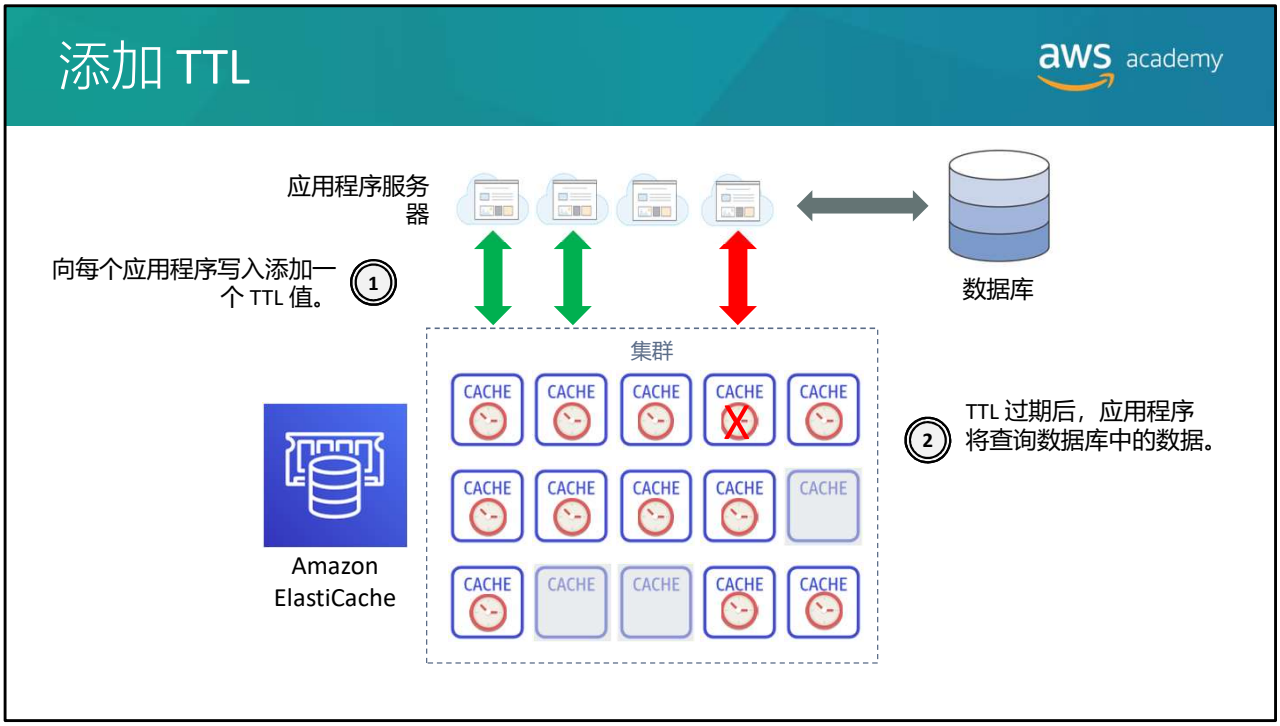
```
def save_user(user_id, values):
    # Save to DB
    record = db.query("update users…where id = ?", user_id, values)
    # Push into cache
    cache.set(user_id, record, 300) # TTL
    return record
```



第二种缓存策略是*直写策略*。如果将数据写入数据库中，则会在缓存中添加数据或更新数据。当您具有必须实时更新的数据时，请使用直写缓存策略。这种方法具有主动性：如果您知道哪些数据会被访问到，就可以避免不必要的缓存未命中情况。一个很好的例子是：任何类型的聚合都适合采用这种方法，例如前 100 名游戏排行榜、最热门的 10 大新闻报道或推荐内容。由于此类数据通常由特定的应用程序或后台作业代码更新，可直接更新缓存。

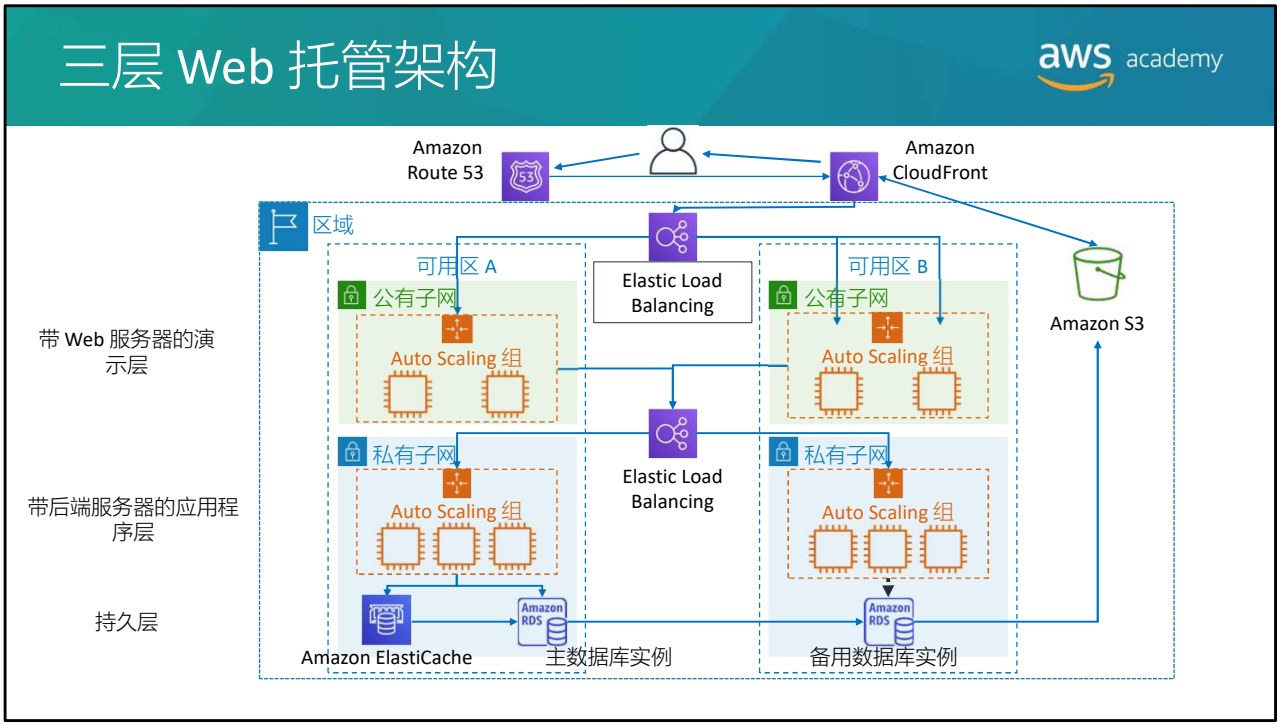
这种方法的优点是，它增加了应用程序在查找该值时在缓存中找到该值的可能性。缺点是，您可能缓存的是不需要的数据，因此这种方法可能会增加成本。

实际上，这两种方法可结合使用，因此了解数据更改的频率并使用适当的 TTL 对您来说非常重要。



延迟加载允许数据过时。而直写可确保数据始终是最新状态，但这种方法可能会使用不必要的数据填充缓存。

通过向每次写入添加 TTL 值，您可以获得每种策略的优点，并避免因数据造成的缓存混乱。TTL 是一个整数值或键，它指定了键过期之前的秒数或毫秒数，具体取决于内存引擎。当应用程序尝试读取过期键时，则视为在缓存中找不到数据。因此，系统会查询数据库并更新缓存。这样可以防止数据过于陈旧，并且需要从数据库中偶尔刷新缓存中的值。



您可能希望使用 Amazon ElastiCache 的一种场景是在传统的三层 Web 托管架构中。在这种场景下，您希望在私有子网中维护私有后端服务器的同时运行公共 Web 应用程序。您可以为 Web 服务器创建一个可访问互联网的公有子网。同时，您可以将后端基础设施放置在无法访问互联网的私有子网中。后端基础设施的数据库层可能包括 Amazon Relational Database Service (Amazon RDS) 数据库实例和提供内存中层的 ElastiCache 集群。

在此 Web 托管架构图中：

- *Amazon Route 53* 使您能够将顶级域名（例如 *example.com*）DNS 名称映射到负载均衡器 DNS 名称。
- *Amazon CloudFront* 提供边缘缓存，支持大量内容。
- 负载均衡器在表示层的 Auto Scaling 组中跨 Web 服务器分配流量。
- 另一个负载均衡器在应用程序层的 Auto Scaling 组中跨后端应用程序服务器分配流量。
- *Amazon ElastiCache* 为应用程序提供了内存中数据缓存，从数据库层中删除负载。

第 5 节要点

aws academy

- 数据库缓存可消除主数据库面临的不必要压力（通常是频繁访问的读取数据），以此为其提供助益
- DAX 是适用于 DynamoDB 的完全托管且高度可用的内存中缓存，可实现高达 10 倍的性能提升 – 从毫秒到微秒
- Amazon ElastiCache 是可以用作内存中数据存储的端缓存，支持要求最严苛且需要亚毫秒级响应速度的应用程序

本模块中这节内容的要点包括：

- 数据库缓存可消除主数据库面临的不必要压力（通常是频繁访问的读取数据），以此为其提供助益
- DAX 是适用于 DynamoDB 的完全托管且高度可用的内存中缓存，可实现高达 10 倍的性能提升 – 从毫秒到微秒
- Amazon ElastiCache 是可以用作内存中数据存储的端缓存，支持要求最严苛且需要亚毫秒级响应速度的应用程序

模块 11：缓存内容

模块总结

现在来回顾下本模块，并对知识测验和对实践认证考试问题的讨论进行总结。

模块总结

aws academy

总体来说，您在本模块中学习了如何：

- 确定如何通过缓存内容提升应用程序性能并减少延迟
- 创建使用 Amazon CloudFront 来缓存内容的架构
- 确定如何设计通过边缘站点实现分配和分布式拒绝服务 (DDoS) 保护的架构
- 识别会话管理与缓存的关系
- 描述如何设计使用 Amazon ElastiCache 的架构

总体来说，您在本模块中学习了如何：

- 确定如何通过缓存内容提升应用程序性能并减少延迟
- 创建使用 Amazon CloudFront 来缓存内容的架构
- 确定如何设计通过边缘站点实现分配和分布式拒绝服务 (DDoS) 保护的架构
- 识别会话管理与缓存的关系
- 描述如何设计使用 Amazon ElastiCache 的架构

完成知识测验

现在可以完成本模块的知识测验。

样题

某家公司正在开发使用无状态 Web 服务器的高度可用的 Web 应用程序。哪些服务适合存储会话状态数据？（请选择两项。）

A. Amazon CloudWatch
B. Amazon DynamoDB
C. Elastic Load Balancing
D. Amazon ElastiCache
E. AWS Storage Gateway

请查看答案选项，并根据之前突出显示的关键字排除错误选项。

正确答案是 B (Amazon DynamoDB) 和 D (Amazon ElastiCache)： DynamoDB 和 ElastiCache 均可提供键值对高性能存储。CloudWatch 和 Elastic Load Balancing 都不是存储服务。AWS Storage Gateway 虽然是存储服务，不过它是一种混合存储服务，本地应用程序可以借助它来使用云存储。

感谢您完成本模块的学习。