讨论课2--复用产品方案

1352871 康慧琳

讨论内容

> 参考业界架构，讨论程序的扩展
>
> 1. 分布式：高可用性、高吞吐量
> 2. 数据存储，分区，一致性，缓存
> 3. 负载均衡
> 4. ID分配
> 5. 通信可靠高效
> 6. 协议
> 7. 消息队列
> 8. 垃圾消息过滤
> 9. 安全 ...
>
> 参考架构：Amazon Dynamo

Amazon平台概述

Amazon平台是一个由数百服务组成的面向服务的架构，其秉承高度去中心化、松散耦合、完全分布式的原则。在这种环境中，尤其需要一个始终可用的存储系统，由此，Dynamo诞生了。

Dynamo概述

Dynamo是Amazon提供的一款高可用的分布式Key-Value存储系统，其满足可伸缩性、可用性、可靠性。

CAP原理满足：通过一致性哈希满足P，用复制满足A，用对象版本与向量时钟满足C。用牺牲C来满足高可用的A，但是最终会一致。但是，是牺牲C满足A，还是牺牲A满足C，可以根据NWR模型来调配，以达到收益成本平衡。

Dynamo内部有3个层面的概念：

Key-Value：Key唯一标识一个数据对象，Value标识数据对象实体，通过对Key来完成对数据对象的读写操作。

节点node：节点是指一个物理主机。在每个节点上，会有3个必备组件：请求协调器（request coordination）、成员与失败检测、本地持久引擎（local persistence engine），这些组件都由Java实现。本地持久引擎支持不同的存储引擎，最主要的引擎是Berkeley Database Transactional Data Store（存储数百K的对象更合适），其它还有BDB Java Edtion、MySQL以及一致性内存Cache。本地持久化引擎组件是一个可插拔的持久化组件，应用程序可以根据需要选择最合适的存储引擎，比如：如果存储对象的通常为数千字节则可以选择BDB，如果是更多尺寸则可以选择MySQL。生产中，Dynamo通常使用BDB事物数据存储。

实例instance：从应用的角度来看就是一个服务，提供IO功能。每个实例由一组节点组成，这些节点可能位于不同的IDC，这样IDC出现问题也不会导致数据丢失，这样会有更好的容灾和可靠性。

核心技术

数据分区

Hash算法：使用MD5对Key进行Hash以产生一个128位的标示符，以此来确定Key的存储节点。

为了达到增量可伸缩性的目地，Dynamo采用一致性哈希来完成数据分区。在一致性哈希中，哈希函数的输出范围为一个圆环，如图2所示，系统中每个节点映射到环中某个位置，而Key也被Hash到环中某个位置，Key从其被映射的位置开始沿顺时针方向找到第一个位置比其大的节点作为其存储节点，换个角度说，就是每个系统节点负责从其映射的位置起到逆时针方向的第一个系统节点间的区域。

一致性哈希最大的优点在于节点的扩容与缩容，只影响其直接的邻居节点，而对其它节点没有影响。这样看似很完美了，但是亚马逊没有因此而停止脚本，这是其伟大之处，其实还存在两个问题：节点数据分布不均匀和无视节点性能的异质性。为了解决这两个问题，Dynamo对一致性哈希进行了改进而引入了虚拟节点，即每个节点从逻辑上切分为多个虚拟节点，每个虚拟节点从逻辑上看像一个真实节点，这样每个节点就被分配到环上多个点而不是一个单点。

数据复制

为了实现高可用，Dynamo将每个数据复制到N台主机上，其中N是每个实例（per-instance）的配置参数，建议值为3。每个Key被分配到一个协调器（coordinator）节点，协调器节点管理其负责范围内的复制数据项，其除了在本地存储其责任范围内的每个Key外，还复制这些Key到环上顺时针方向的N-1个后继节点。这样，系统中每个节点负责环上从其自己位置开始到第N个前驱节点间的一段区域。

为了保证复制时数据副本的一致性，Dynamo采用类似于Quorum系统的一致性协议实现。这里涉及到三个关键参数(N, R, W)，其中，N是指数据对象复制到N台主机，协调器负责将数据复制到N-1个节点上，亚马逊建议N配置为3，R代表一次成功的读取操作中最小参与节点数量，W代表一次成功的写操作中最小参与节点数量。R+W>N，则会产生类似于Quorum的效果。该模型中，读（写）延迟由最慢的R(W)复制副本决定，为了得到比较小的延迟，R和W通常配置为小于N。亚马逊建议(N, R, W)设置为(3, 2, 2)以兼顾性能与可用性。R和W直接影响性能、扩展性和一致性，如果W设置为1，则一个实例中只要有一个节点可用，也不影响写操作，如果R设置为1，只要有一个节点可用，也不会影响读请求，R和W值过小则影响一致性，过大则可用性，因此，需要在R和W两个值之间平衡，这也是Dynamo的一个亮点之一。

版本合并

由前文可知，Dynamo为了保证高可用，对每份数据都复制了多份（建议3份），在数据没有被异步复制到所有副本前，如果有get操作会取到不一致的数据，但是Dynamo提供最终一致性。在亚马逊平台中，购物车就是这种情况的典型应用，为了保证购物车永远可用，对任何一个副本的任何一次更改操作的结果都会当做一个数据版本存储起来，这样当用户get时就会取到多个版本，这样也就需要做数据版本合并了。Dynamo将合并工作推给应用程序，在这里就是购物车get时处理。

故障检测

（1）Ring Membership

每个节点启动时存储自己在环上的映射信息并持久化到磁盘上，然后每个节点每隔一秒随机选择一个对等节点，通过Gossip协议传播节点的映射i信息，最终每个节点都知道对等节点所处理范围，即每个节点都可以直接转发一个key的读/写操作到正确的数据集节点，而不需要经过中间路由或者跳。

（2）External Discovery

如果人工分别往Dynamo环中加入节点A和B，则Ring Membership不会立即检测到这一变化，而出现暂时逻辑分裂的Dynamo环（A和B都认为自己在环中，但是互相不知道对方存在）。Dynamo用External Discovery来解决这个问题，即有些Dynamo节点充当种子节点的角色，在非种子节点中配置种子节点的IP，所有非种子节点都与种子节点协调成员关系 。

（3）Failure Detection

Dynamo采用类Gossip协议来实现去中心化的故障检测，使系统中的每个节点都可以了解其它节点的加入和likai

故障处理

传统的Quorum，在节点故障或者网络故障情况下，系统不可用。为了提高可用性，Dynamo采用Sloppy Quorum和Hinted Handoff，即所有读写操作由首选列表中的前N个健康节点执行，而发往故障节点的数据做好标记后被发往健康节点，故障节点重新可用时恢复副本。

如图2所示中Dynamo配置N为3。如果在写过程中节点A暂时不可用（Down或无法连接），则发往A的副本将被发送到节点D，发到D的副本在其原始数据中有一个hint以表明节点A才是副本的预期接收者，D将副本数据保存在一个单独的本地存储中，在检测到A可用时，D尝试将副本发到A，如果发送成功，D会将数据从本地存储中删除而不会降低系统中的副本总数。

一个高可用的存储系统具备处理整个IDC故障（断电、自然灾害、网络故障灯）的能力是非常重要的，Dynamo就具备此能力。Dynamo可以配置成跨多个IDC复制对象，即key的首选列表由跨多个IDC的节点组成，这些IDC之间由高速专线连接，跨多个IDC的复制方案使得Dynamo能够处理整个IDC故障。

此外，为了处理在hinted副本移交会预期节点之前该副本不可用的情况，Dynamo实现了anti-entropy协议来保持副本同步，为了更快递检测副本之间的不一致性并减少传输量，Dynamo采用MerkleTree。

解决问题

负载均衡

Dynamo采用一致性的散列将key space(键空间)分布在其所有的副本上，并确保负载均匀分布。假设对key的访问分布不会高度偏移，一个统一的key分配可以帮助我们达到均匀的负载分布。特别地, Dynamo设计假定，即使访问的分布存在显着偏移，只要在流行的那端(popular end)有足够多的keys，那么对那些流行的key的处理的负载就可以通过partitioning均匀地分散到各个节点。

可用性

完全去中心化，无单点，永远可写。

伸缩性

带虚拟机节点的一致性hash：一致性hash解决扩容/缩容问题，虚拟节点解决机器异质性问题。

可靠性

数据复制多份副本，用向量时钟解决版本合并问题。

可配置

平衡性可调，即根据（N,W,R）模型平衡可用性和一致性，建议模型参数为（3,2,2）。

Dynamo架构优点

Dynamo的主要优点是，它提供了使用三个参数的(N，R，W)，根据自己的需要来调整它们的实例。不同于流行的商业数据存储，Dynamo将数据一致性与协调的逻辑问题暴露给开发者。开始，人们可能会认为应用程序逻辑会变得更加复杂。然而，从历史上看，Amazon平台都为高可用性而构建，且许多应用内置

了处理不同的失效模式和可能出现的不一致性。因此，移植这些应用程序到使用Dynamo是一个相对简单的任务。对于那些希望使用Dynamo的应用，需要开发的初始阶段做一些分析，以选择正确的冲突的协调机制以适当地满足业务情况。最后，Dynamo采用全成员(full membership)模式，其中每个节点都知道其对等节点承载的数据。要做到这一点，每个节点都需要积极地与系统中的其他节点Gossip完整的路由表。这种模式在一个包含数百个节点的系统中运作良好，然而，扩展这样的设计以运行成千上万节点并不容易，因为维持路由表的开销将随着系统的大小的增加而增加。克服这种限制可能需要通过对 Dynamo 引入分层扩展。

Dynamo架构总结

Dynamo，一个高度可用和可扩展的数据存储系统，被Amazon.com电子商务平台用来存储许多核心服务的状态。Dynamo已经提供了所需的可用性和性能水平，并已成功处理服务器故障，数据中心故障和网络分裂。Dynamo是增量扩展，并允许服务的拥有者根据请求负载按比例增加或减少。Dynamo让服务的所有者通过调整参数N,R和W来达到他们渴求的性能，耐用性和一致性的SLA。