Lecture 23 Amazon云

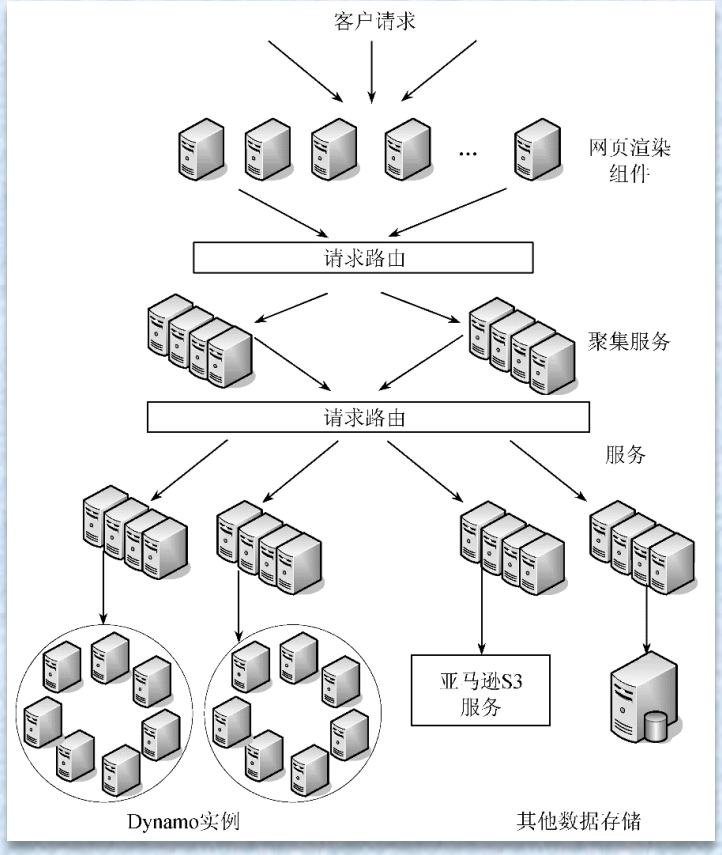
目录

目录

23.1 基础存储架构Dynamo

23.1 基础存储架构Dynamo

- Dynamo概况



为了保证其稳定性，Amazon的系统采用完全的**分布式、去中心化的架构**

作为底层存储架构的Dynamo也同样采用了无中心的模式

Dynamo只支持简单的键/值（key/value）方式的数据存储，不支持复杂的查询

Dynamo中存储的是数据值的原始形式，即按位存储，并不解析数据的具体内容

面向服务的Amazon平台基本架构

目录

23.1 基础存储架构Dynamo

23.1 基础存储架构Dynamo

Dynamo需要解决的主要问题及解决方案

- Dynamo在设计时被定位为一个基于分布式存储架构的，高可靠、高可用且具有良好容错性的系统。下图列举了Dynamo设计时面临的主要问题及所采取的解决方案。

问　题	采取的相关技术
数据均衡分布	改进的一致性哈希算法
数据备份	参数可调的弱quorum机制
数据冲突处理	向量时钟（Vector Clock）
成员资格及错误检测	基于Gossip协议的成员资格和错误检测
临时故障处理	Hinted handoff（数据回传机制），
永久故障处理	Merkle哈希树

23.1 基础存储架构Dynamo

- Dynamo的存储节点

Dynamo中的存储节点呈无中心的环状分布。

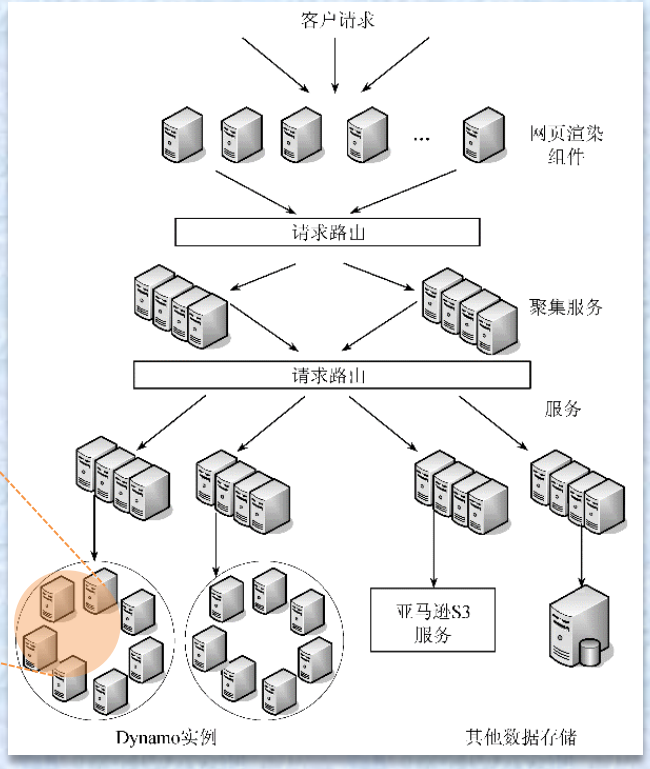
两个基本概念

preference list

存储与某个特定键值相对应的数据的节点列表

coordinator

执行一次读或写操作的节点



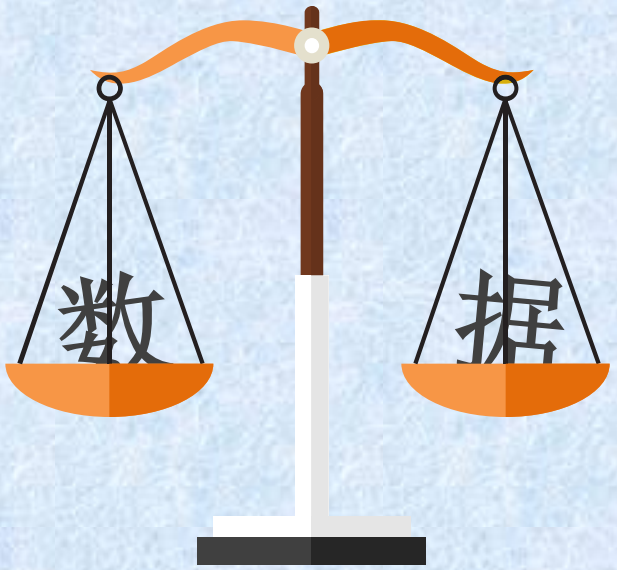
通常，coordinator 是 preference list 上的第一个节点

23.1 基础存储架构Dynamo

- 数据均衡分布的问题

Dynamo采用了分布式的数据存储架构，均衡的数据分布可以保证负载平衡和系统良好的扩展性。

因此，如何在各个节点上数据的均衡性是影响Dynamo性能的关键问题。



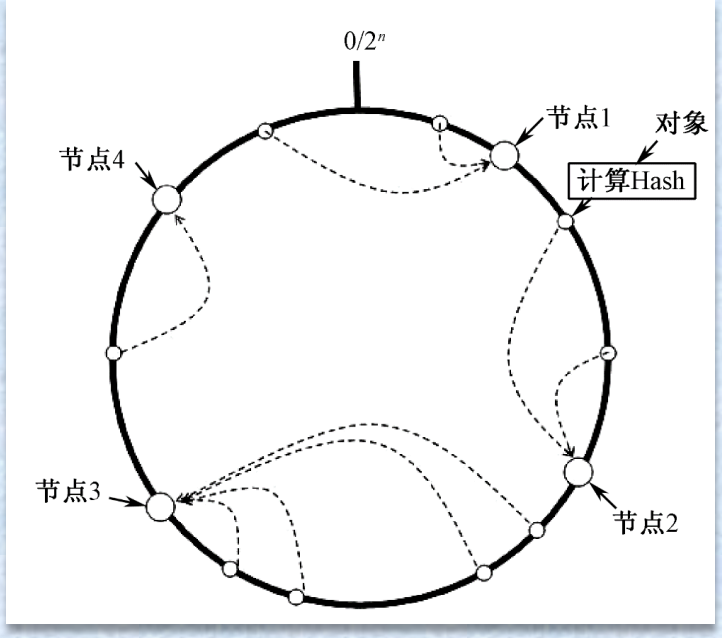
Dynamo中使用改进后的一致性哈希算法，并在此基础上进行数据备份，以提高系统的可用性。

23.1 基础存储架构Dynamo

- 数据均衡分布的问题　**一致性哈希算法**

一致性哈希算法是目前主流的**分布式哈希表**（Distributed Hash Table，DHT）协议之一，于**1997年**由**麻省理工学院**提出。

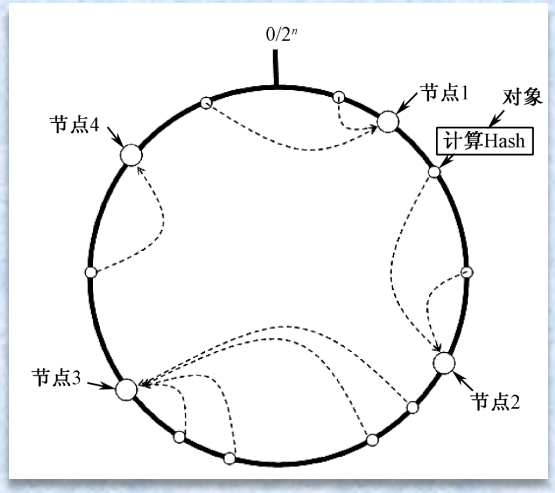
一致性哈希算法通过修正简单哈希算法，解决了网络中的**热点问题**，使得**DHT**可以真正地应用于**P2P环境**中。

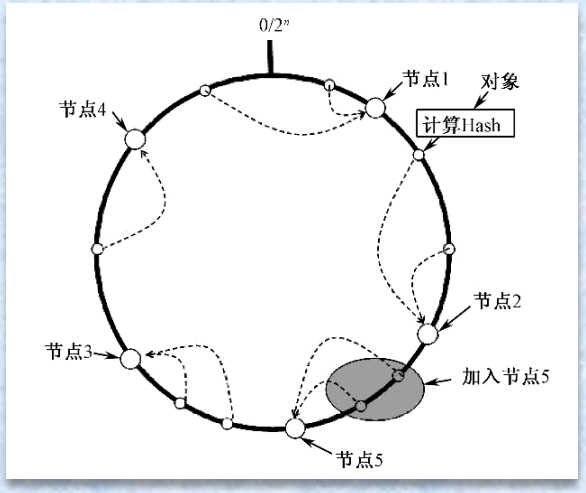


23.1 基础存储架构Dynamo

- 数据均衡分布的问题

一致性哈希算法除了能够保证哈希运算结果充分分散到整个环上外，还能保证在添加或删除设备节点时只会影响到其在哈希环中的前驱设备节点，而不会对其他设备节点产生影响。





一致性哈希算法可以大大降低在添加或删除节点时引起的节点间的数据传输开销

23.1 基础存储架构Dynamo

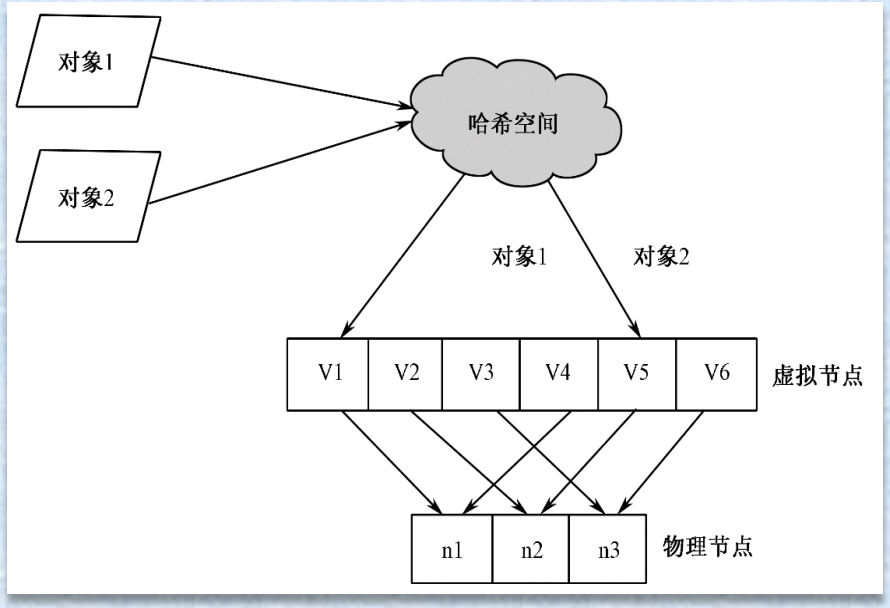
- 数据均衡分布的问题

改进的一致性哈希算法

Dynamo中引入了**虚拟节点（计算资源抽象模型）**的概念

每个虚拟节点都**隶属于**某一个实际的**物理节点**，一个物理节点根据其性能的差异被分为**一个或多个虚拟节点**。

各个虚拟节点的**能力基本相当**，并**随机分布**在哈希环上。



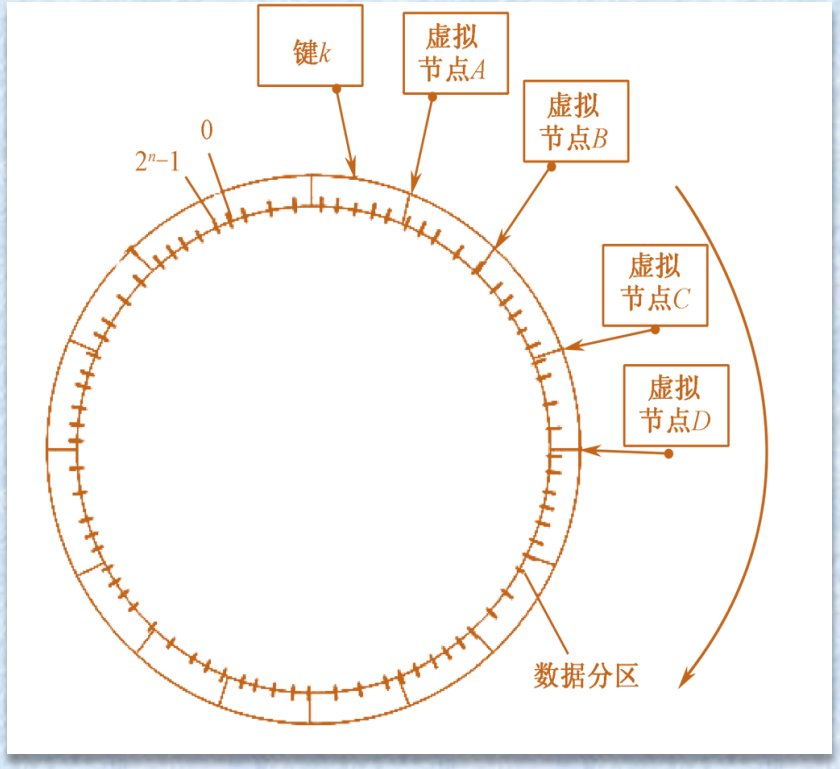
- 数据均衡分布的问题

Dynamo将整个哈希环划分成Q等份，每个等份称为一个数据分区（Partition）再将虚拟节点映射到分区上

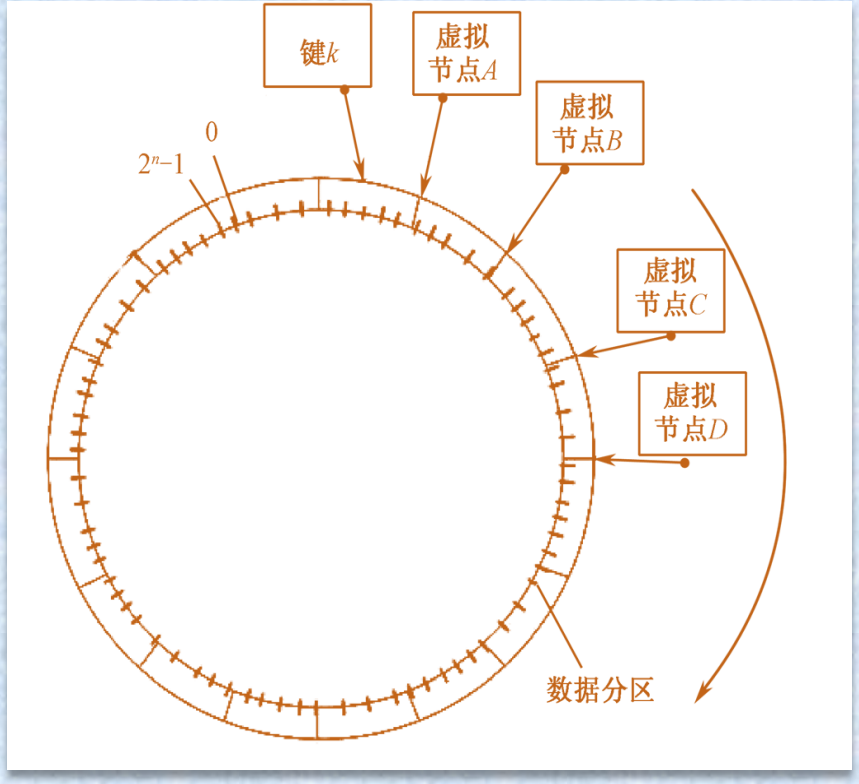
存储数据时，每个数据会被先分配到某个数据分区，再根据负责该数据分区的虚拟节点，最终确定其所存储的物理节点。

数据分区的好处

1. 减小数据分布不均衡的可能性
2. 添加或删除设备节点时引起较小的数据传输



数据备份



在Dynamo中，每个数据的副本备份存储在哈希环顺时针方向上该数据所在虚拟节点的后继节点中。

Dynamo对数据进行冗余备份，一般副本数N为3。键k数据保存在虚拟节点A中，其副本保存在虚拟节点B，C中。

Dynamo对数据写进行了优化：保证一个副本写入硬盘，其他副本写入内存即返回成功。

Dynamo还保证相邻的节点位于不同的地区区域，如此某个数据中心瘫痪，保证数据在其他数据中心有副本。

数据备份在存储数据的同时进行，会使每次写操作的延时变长。

23.1 基础存储架构Dynamo

- 数据冲突问题

Dynamo选择通过牺牲一致性来保证系统的可靠性和可用性，
没有采用强一致性模型而采用了最终一致性模型。

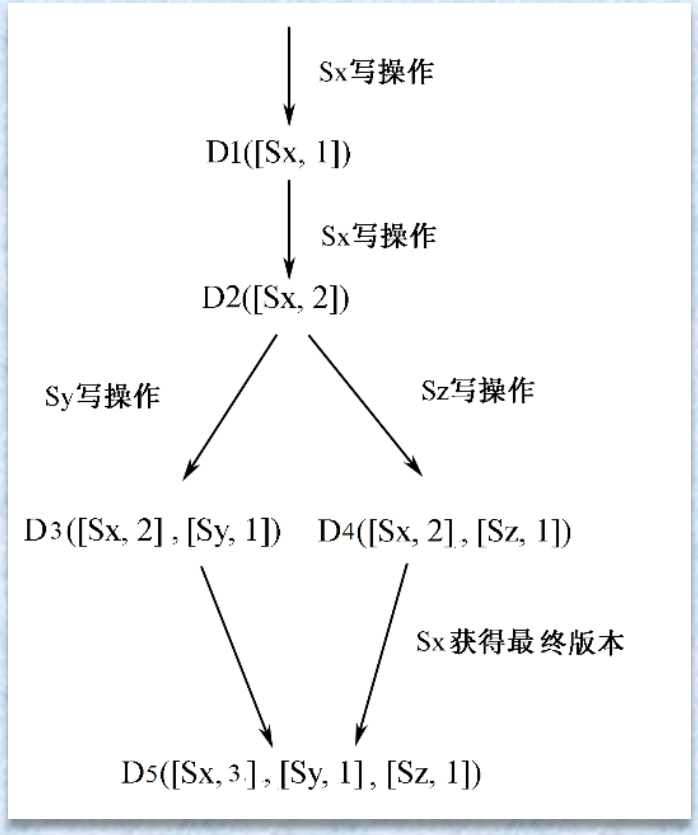
由于Dynamo中可能出现同一个数据被多个节点同时更新的情况，
且无法保证数据副本的更新顺序，这有可能会导致数据冲突。

目录

数据冲突问题

如何解决

- 数据冲突问题



Dynamo中采用了**向量时钟技术**

（Vector Clock）

Dynamo中的向量时钟通过

[node, counter] 对来表示。

node表示**操作节点**

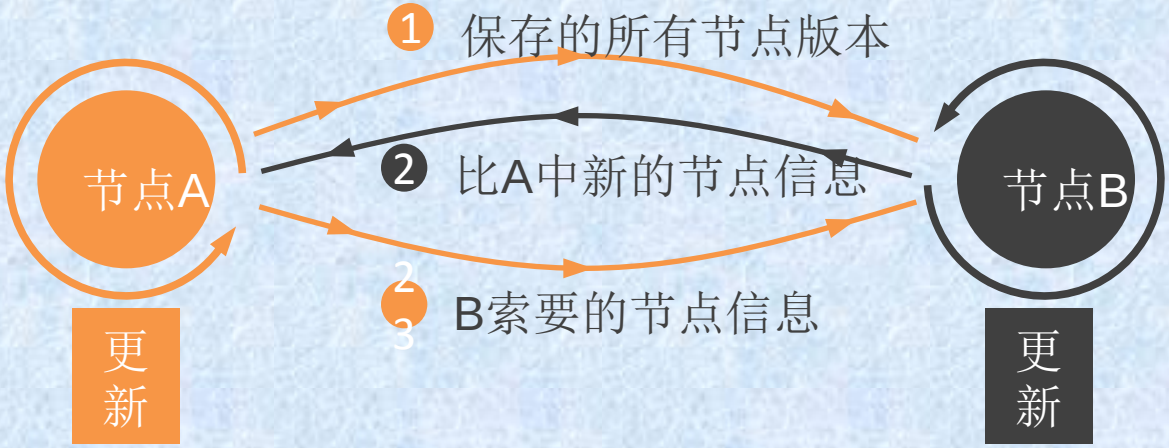
counter是其**对应的计数器，初始值为 0**

节点每进行一次更新操作则**计数器加 1**

• 成员资格及错误检测

由于Dynamo采用了无中心的架构，每个成员节点都需要保存其他节点的路由信息

为了保证每个节点都能拥有最新的成员节点信息，Dynamo中采用了一种类似于Gossip（闲聊）协议的技术

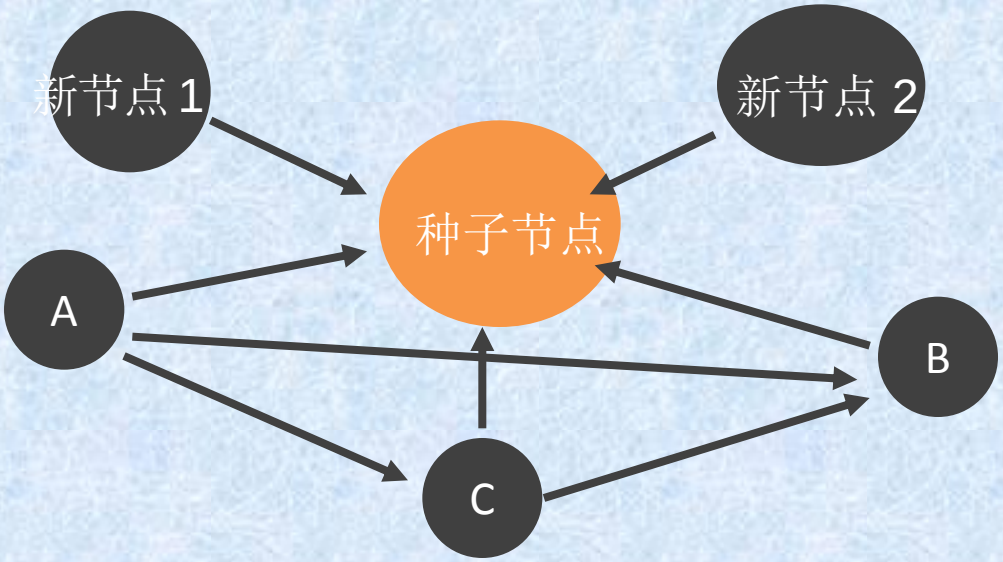


Dynamo中还通过Gossip来实现错误检测任何节点向其他节点发起通信后，如果对方没有回应，则认为对方节点失效

23.1 基础存储架构Dynamo

- 成员资格及错误检测

为了避免新加入的节点之间不能及时发现其他节点的存在，Dynamo中设置了一些种子节点（Seed Node）。种子节点和所有的节点都有联系。当新节点加入时，它扮演一个中介的角色，使新加入节点之间互相感知。



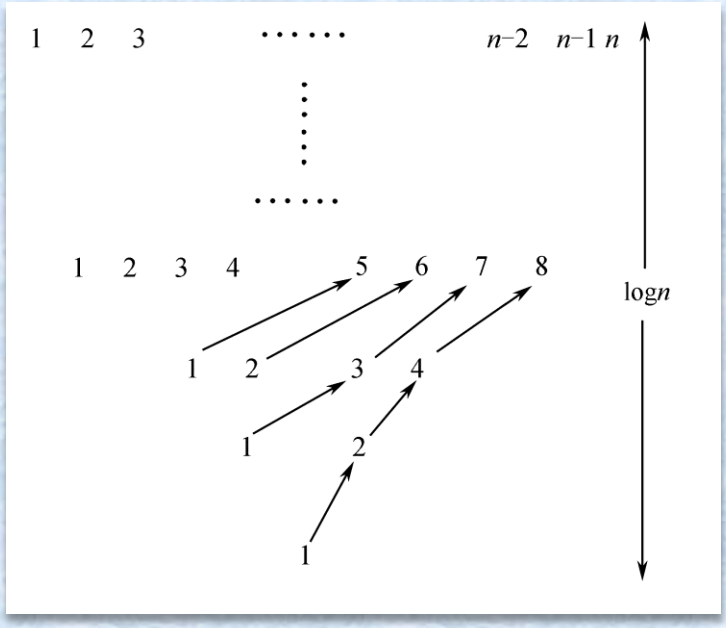
23.1 基础存储架构Dynamo

- 成员资格及错误检测

- 自底向上每一层代表一次随机通信
- 第一层节点1将信息交换给节点2
- 第二层节点1和2同时开始随机选择其他节点交换信息
- 直到N个节点全部传遍

结论：

- Dynamo中的节点数不能太多
- Amazon采用了分层Dynamo结构来解决该问题



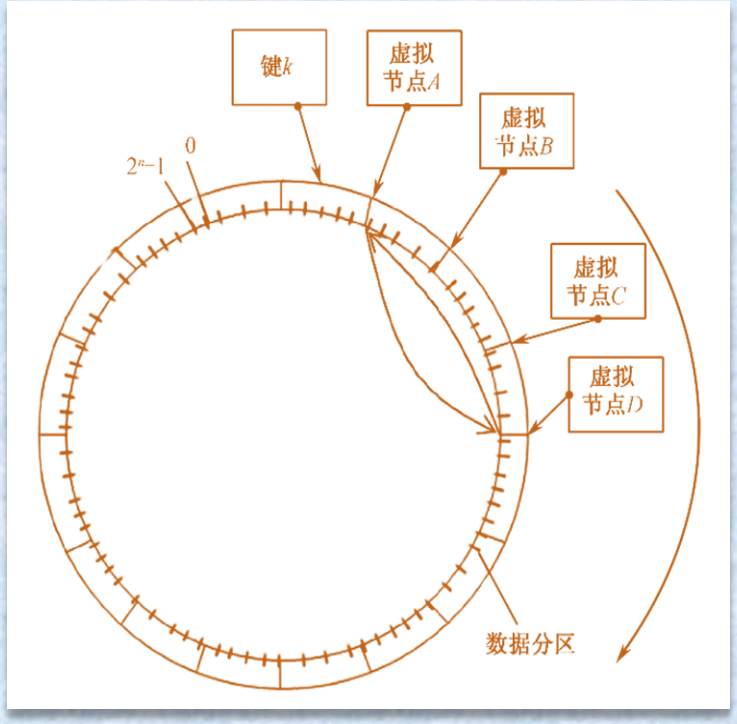
23.1 基础存储架构Dynamo

- 容错机制

临时故障处理机制

-为了处理临时失效的节点，Dynamo中采用了一种带有监听的数据回传机制（Hinted Handoff）

-当虚拟节点A失效后，会将数据临时存放在节点D的临时空间中，并在节点A重新可用后，由节点D将数据回传给节点A。

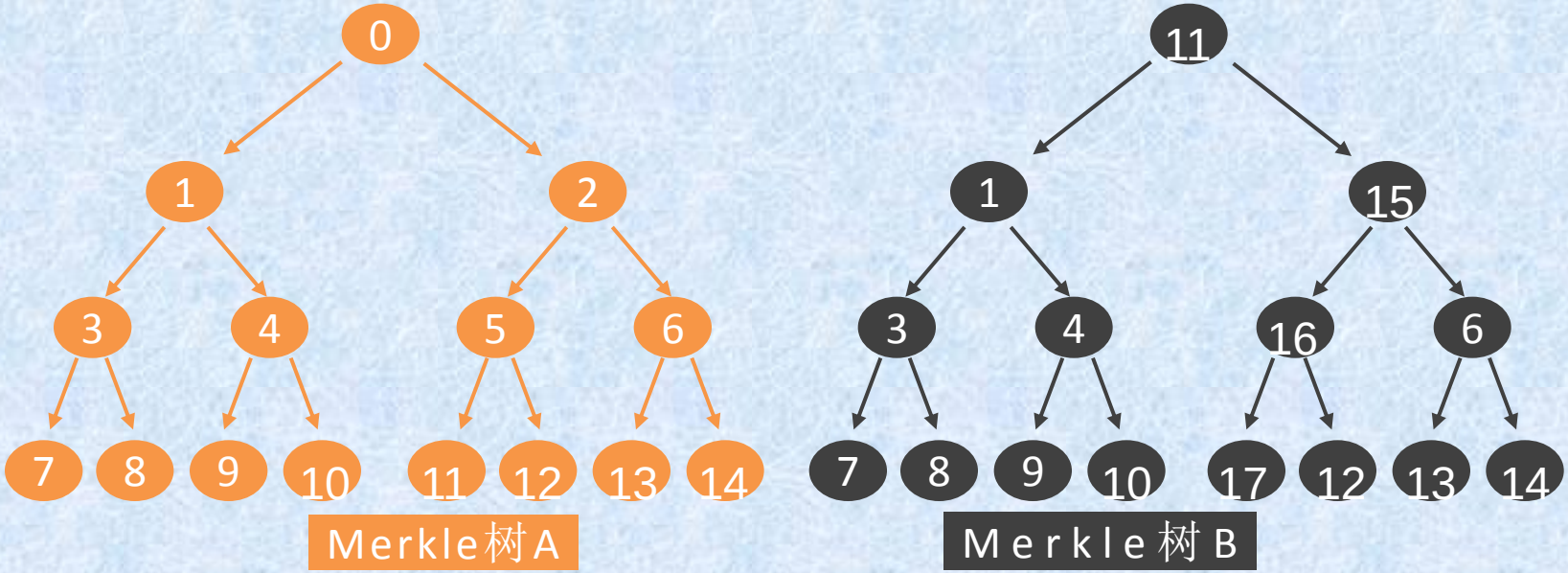


23.1 基础存储架构Dynamo

- 容错机制

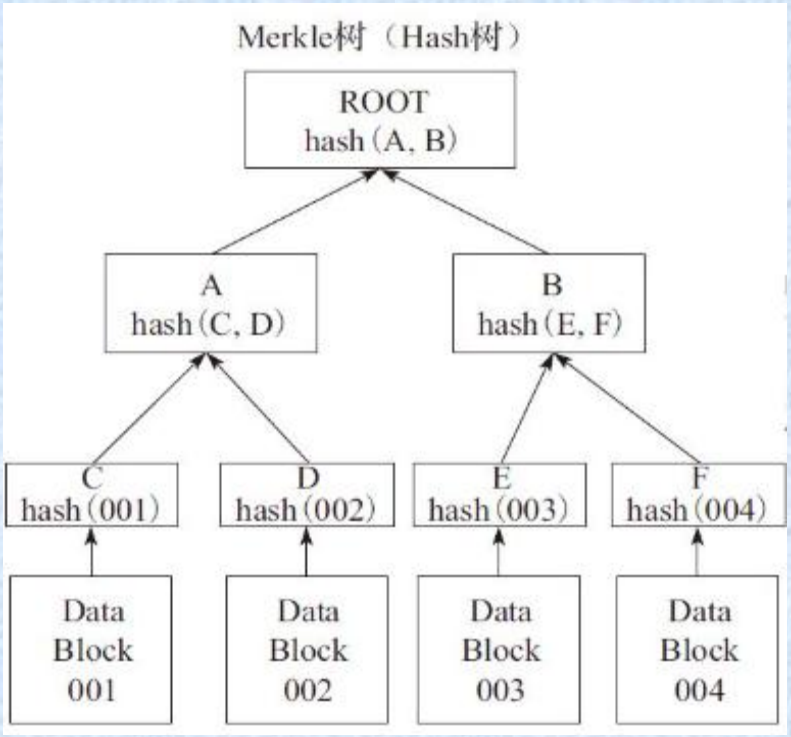
永久性故障处理机制

Dynamo采用**Merkle哈希树**技术来加快检测和减少数据传输量

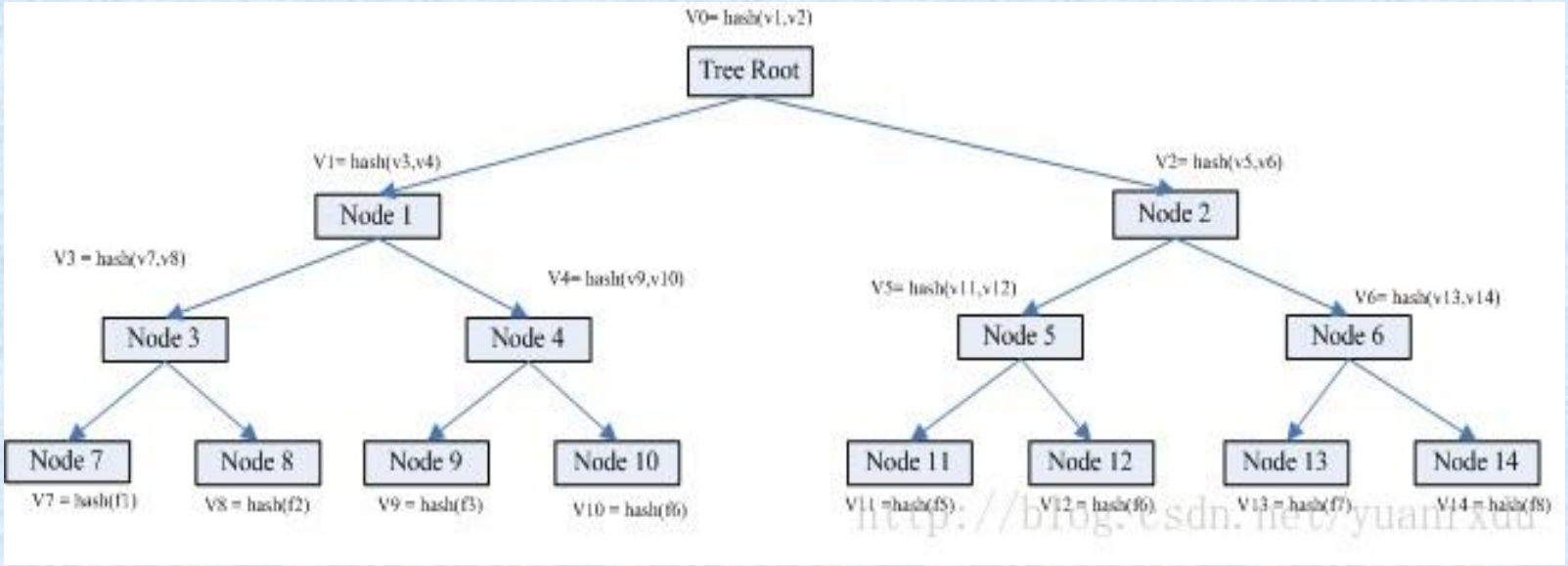


Merkle哈希树是一类基于哈希值的二叉树或多叉树，其叶子节点上的值通常为数据块的哈希值，而非叶子节点上的值是将该节点的所有子节点的组合结果的哈希值。

如下图所示为一个Merkle哈希树，节点A的值必须通过节点C、D上的值计算而得到。叶子节点C、D分别存储数据块001和002的哈希值，而非叶子节点A存储的是其子节点C、D的组合的哈希值，这类非叶子节点的哈希值被称作路径哈希值，而叶子节点的哈希值是实际数据的哈希值。



为了更好理解Merkle Tree，我们假设有A和B两台机器，A需要与B相同目录下有8个文件，文件分别是f1 f2 f3 ....f8。假设我们在文件创建的时候每个机器都构建了一个Merkle Tree。具体如下图:



假如A上的文件5与B上的不一样。我们怎么通过两个机器的Merkle Tree信息找到不相同的文件? 这个比较检索过程如下:

Step1. 首先比较v0是否相同,如果不同，检索其孩子node1和node2.

Step2. v1 相同，v2不同。检索node2的孩子node5 node6;

Step3. v5不同，v6相同，检索比较node5的孩子node 11 和node 12

Step4. v11不同，v12相同。node 11为叶子节点，获取其目录信息。

Step5. 检索比较完毕。

以上过程的理论复杂度是Log(N)。

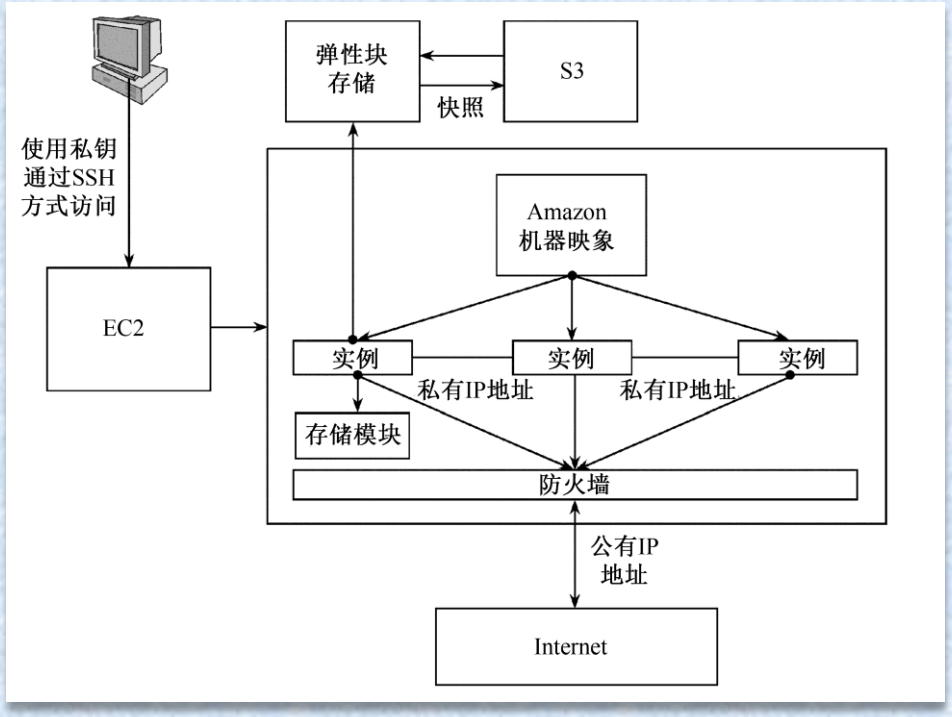
目录

目录

23.2 弹性计算云EC2

23.2 弹性计算云EC2

- EC2的基本架构



主要包括了Amazon机器映象、实例、存储模块等组成部分，并能与S3等其他Amazon云计算服务结合使用。

23.2 弹性计算云EC2

- **Amazon机器映象（AMI）**

Amazon机器映像（Amazon Machine Image，AMI）是包含了操作系统、服务器程序、应用程序等软件配置的模板

当用户用EC2服务创建自己的应用程序时，首先需要构建或获取相应的AMI

构建好的AMI分为Amaznon EBS支持和实例存储支持两类

23.2 弹性计算云EC2

- 实例（Instance）

-EC2中实例由**AMI启动**，可以像传统的主机一样提供服务。同一个AMI可以用于创建具有**不同计算和存储能力的实例**。

-Amazon提供了多种不同类型的实例，分别在**计算**、**GPU**、**内存**、**存储**、**网络**、**费用**等方面进行了优化

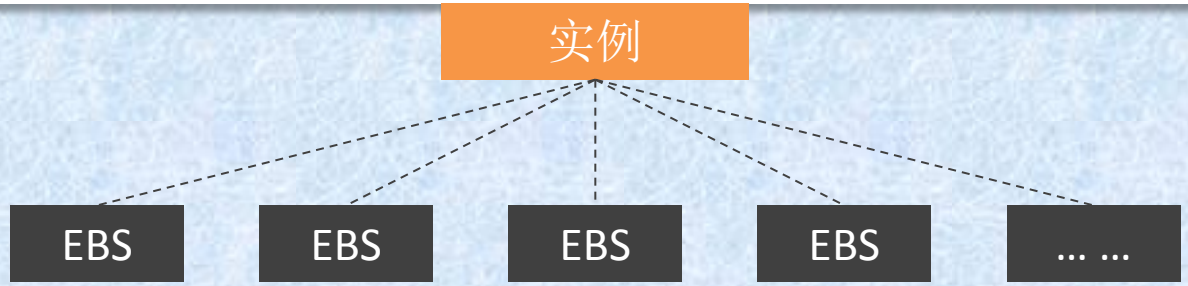
-Amazon还允许用户在应用程序的需求发生变更时，对**实例的类型进行调整**，从而实现按需付费。

-Amazon EC2还为实例提供了许多**附加功能**，帮助用户更好地**部署和管理应用程序。**

23.2 弹性计算云EC2

- 弹性块存储（EBS）

EBS存储卷的设计与**物理硬盘相似**，其大小由用户设定，目前提供的容量从**1GB到1TB**不等。



EBS存储卷适用于数据需要**细粒度**地频繁访问并持久保存的情形，适合作为文件系统或数据库的**主存储**。

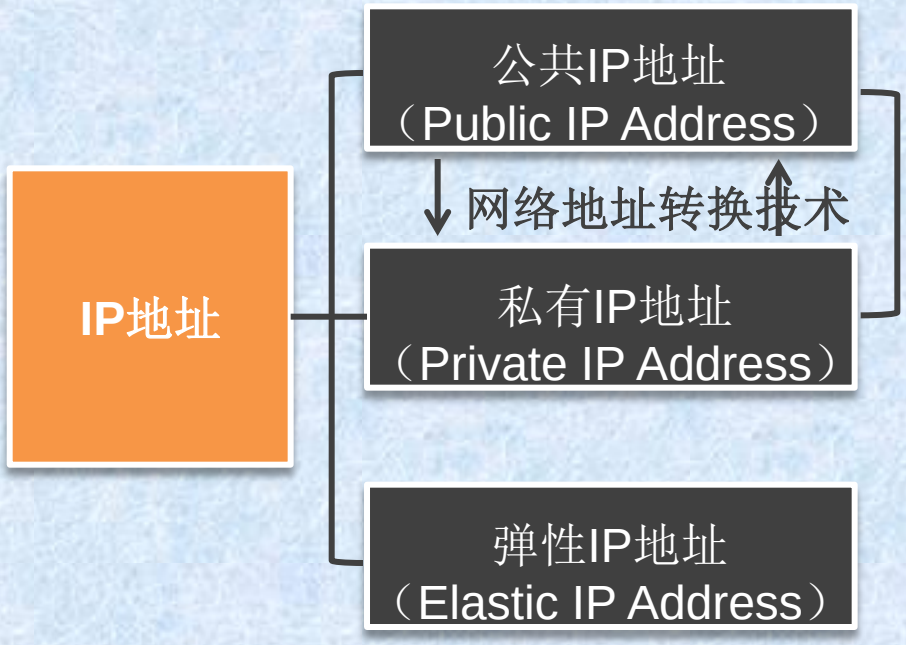
快照功能是EBS的特色功能之一，用于在S3中存储Amazon EBS卷的时间点副本。

目录

23.2 弹性计算云EC2

23.2 弹性计算云EC2

- EC2的通信机制



EC2的**实例**一旦被**创建**就会动态地分配公共IP地私有IP地址由动态主机配置协议（DHCP）分配产生

23.2 弹性计算云EC2

- 弹性负载平衡

弹性负载平衡功能允许**EC2实例自动分发应用流量**，从而保证工作负载不会超过现有能力，并且在一定程度上**支持容错**。

弹性负载平衡功能可以**识别**出应用实例的**状态**，当一个应用运行不佳时，它会自动将流量路由到状态较好的实例资源上，直到前者恢复正常才会**重新分配流量**到其实例上。

23.2 弹性计算云EC2

- 监控服务

Amazon CloudWatch提供了AWS资源的可视化检测功能

用户只需要选择EC2实例，设定监视时间，CloudWatch就可以自动收集和存储检测数据

23.2 弹性计算云EC2

- 自动缩放

自动缩放可以按照用户自定义的条件，**自动调整EC2的计算能力**：

需求高峰期	确保EC2实例的处理能力无缝增大
需求下降时	自动缩小EC2实例规模以降低成本

自动缩放功能特别适合**周期性变化**的应用程序，它由**CloudWatch**自动启动。

23.2 弹性计算云EC2

- 服务管理控制台

各项技术通过互相配合来实现EC2的可扩展性和可靠性

目录

23.2 弹性计算云EC2

23.2 弹性计算云EC2

- EC2的安全及容错机制

安全组是一组规则，用户利用这些规则来决定哪些网络流量会被实例接受，其他则全部拒绝。

当用户的实例被创建时，如果没有指定安全组，则系统自动将该实例分配给一个默认组。

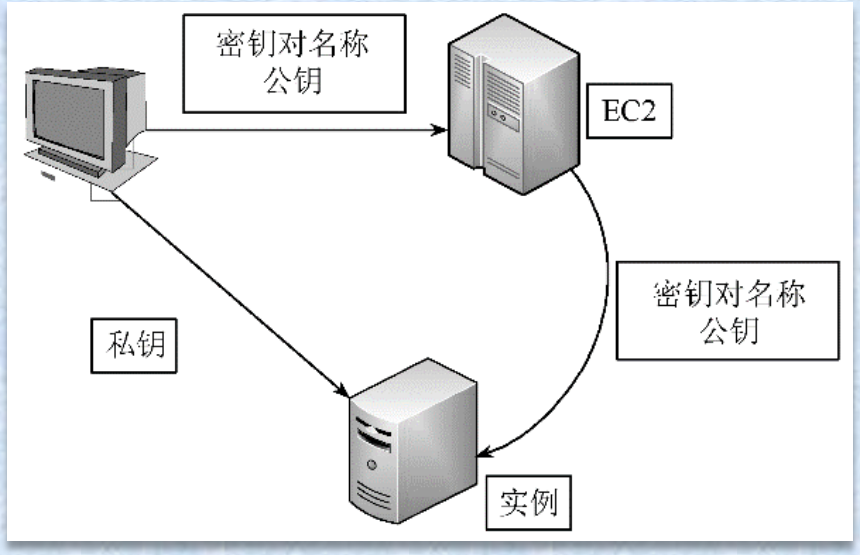
默认组只接受组内成员的消息，拒绝其他消息

当一个组的规则改变后，改变的规则自动适用于组中所有的成员。

23.2 弹性计算云EC2

- **EC2的安全及容错机制**

  SSH是目前对网络上传输的数据进行加密的一种很可靠的协议，当用户创建一个密钥对时，密钥对的名称（Key Pair Name）和公钥（Public Key）会被存储在EC2中



用户使用密钥对登录服务

23.2 弹性计算云EC2

- **EC2的安全及容错机制**

EC2引入了弹性IP地址的概念

- 弹性IP地址和**用户账号绑定**而不是和某个特定的实例绑定
- 当系统正在使用的实例出现故障时，用户只需要将弹性IP地址通过网络地址转换**NAT**转换为新实例所对应的**私有IP地址**
- 通过弹性IP地址改变映射关系总可以**保证有实例可用**

目录

目录

23.3 简单存储服务S3

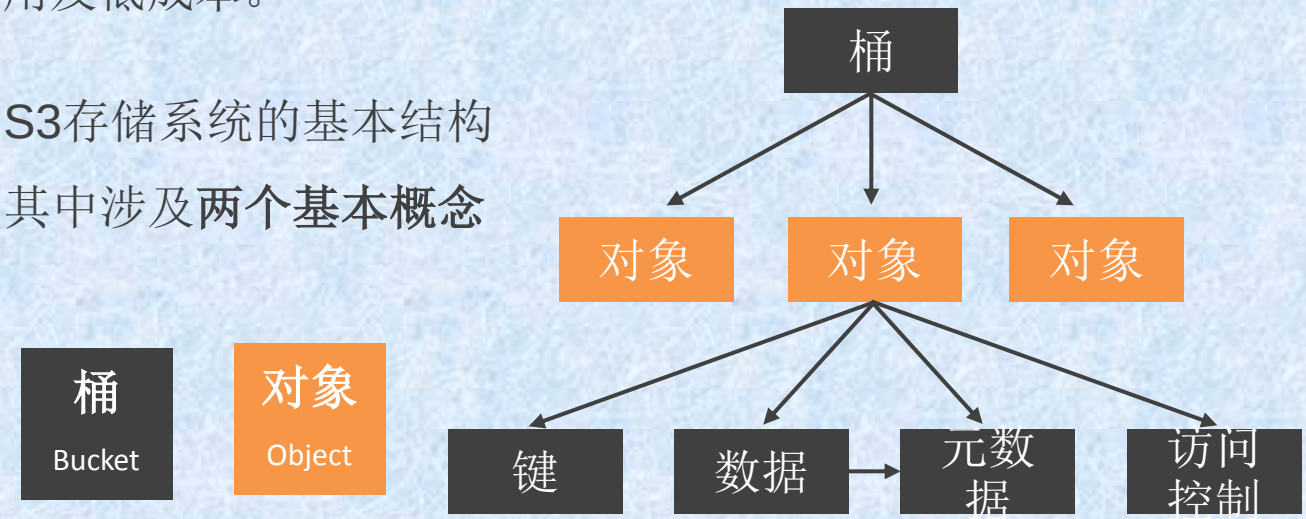
23.3 简单存储服务S3

- **S3的基本概念和操作**

简单存储服务（Simple Storage Services，S3）构架在Dynamo之上，用于提供任意类型文件的临时或永久性存储。S3的总体设计目标是可靠、易用及低成本。

S3存储系统的基本结构

其中涉及**两个基本概念**



23.3 简单存储服务S3

- S3的基本概念和操作

桶

Bucket

桶是用于存储对象的容器，其作用类似于文件夹，但桶不可以被嵌套，即在桶中不能创建桶。

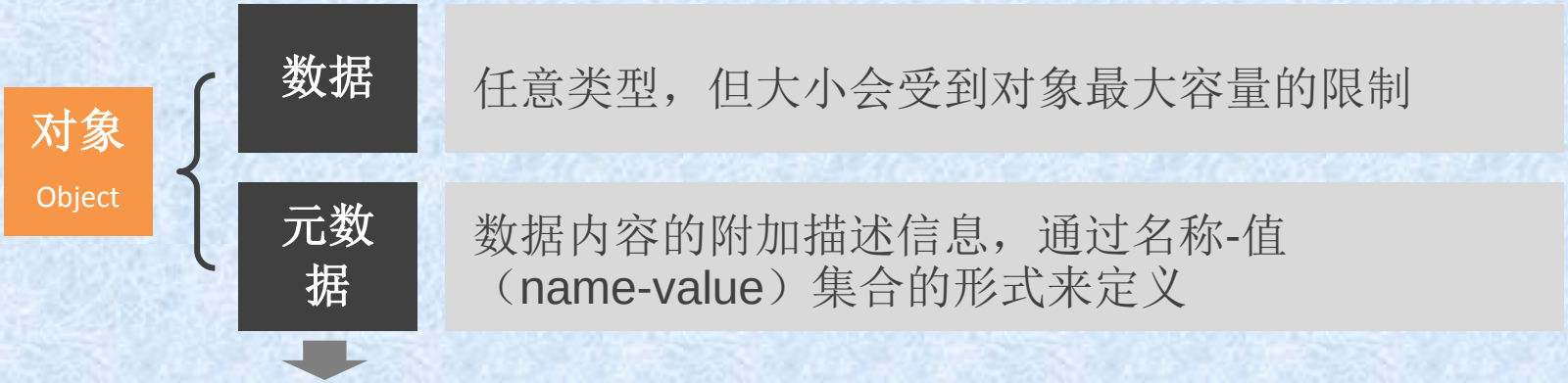
目前，Amazon限制了每个用户创建桶的数量，但没有限制每个桶中对象的数量。

桶的名称要求在整个Amazon S3的服务器中是全局唯一的，以避免在S3中数据共享时出现相互冲突的情况

在对桶命名时，建议采用符合DNS要求的命名规则，以便与CloudFront等其他AWS服务配合使用。

23.3 简单存储服务S3

- S3的基本概念和操作



元数据名称	名称含义
last-modified	对象被最后修改的时间
ETag	利用MD5哈希算法得出的对象值
Content-Type	对象的MIME（多功能网际邮件扩充协议）类型，默认二进制/八位组
Content-Length	对象数据长度，以字节为单位

23.3 简单存储服务S3

- **S3的基本概念和操作**

S3中支持对桶和对象的操作，主要包括：**Get**、**Put**、**List**、**Delete**和**Head**。下图列出了五种操作的主要内容。

操作目标	Get	Put	List	Delete	Head
桶	获取桶中对象	创建或更新桶	列出桶中所有键	删除桶	—
对象	获取对象数据和元数据	创建或更新对象	—	删除对象	获取对象元数据

表3-3 S3的主要操作

目录

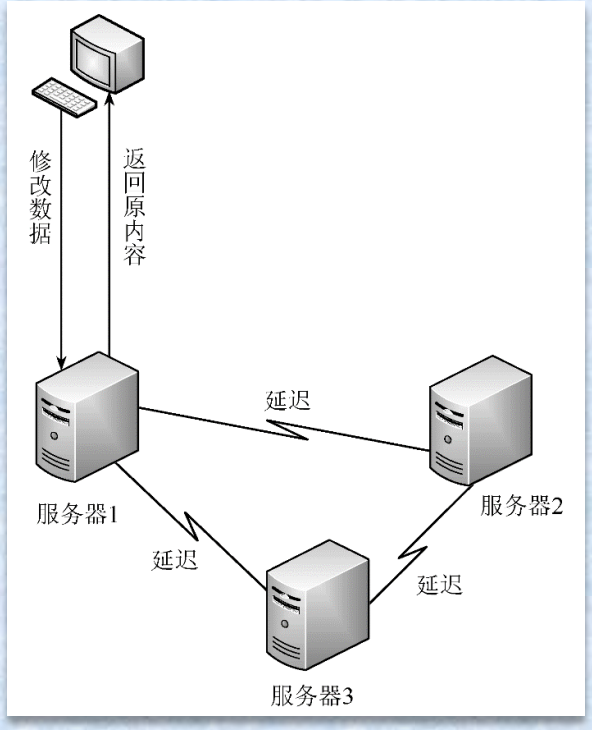
23.3 简单存储服务S3

23.3 简单存储服务S3

- S3的数据一致性模型

与其构建的基础Dynamo相同，S3中采用了最终一致性模型。

在数据被充分传播到所有的存放节点之前，服务器返回给用户的仍是原数据，此时用户操作可能会出现后面几种情况：



23.3 简单存储服务S3

- S3的数据一致性模型

	用户操作	结果
1	写入一个新的对象并立即读取它	服务器可能返回“键不存在”
2	写入一个新的对象并立即列出桶中已有的对象	该对象可能不会出现在列表中
3	用新数据替换现有的对象并立即读取它	服务器可能返回原有的数据
4	删除现有的对象并立即读取它	服务器可能返回被删除的数据
5	删除现有的对象并立即列出桶中的所有对象	服务器可能列出被删除的对象

23.3 简单存储服务S3

- S3的数据一致性模型

最终一致性读	一致性读
可能读到过时数据	没有过时的读取
最低的读延迟	潜在的更高读延迟
最高的读吞吐量	潜在的较低的读吞吐量

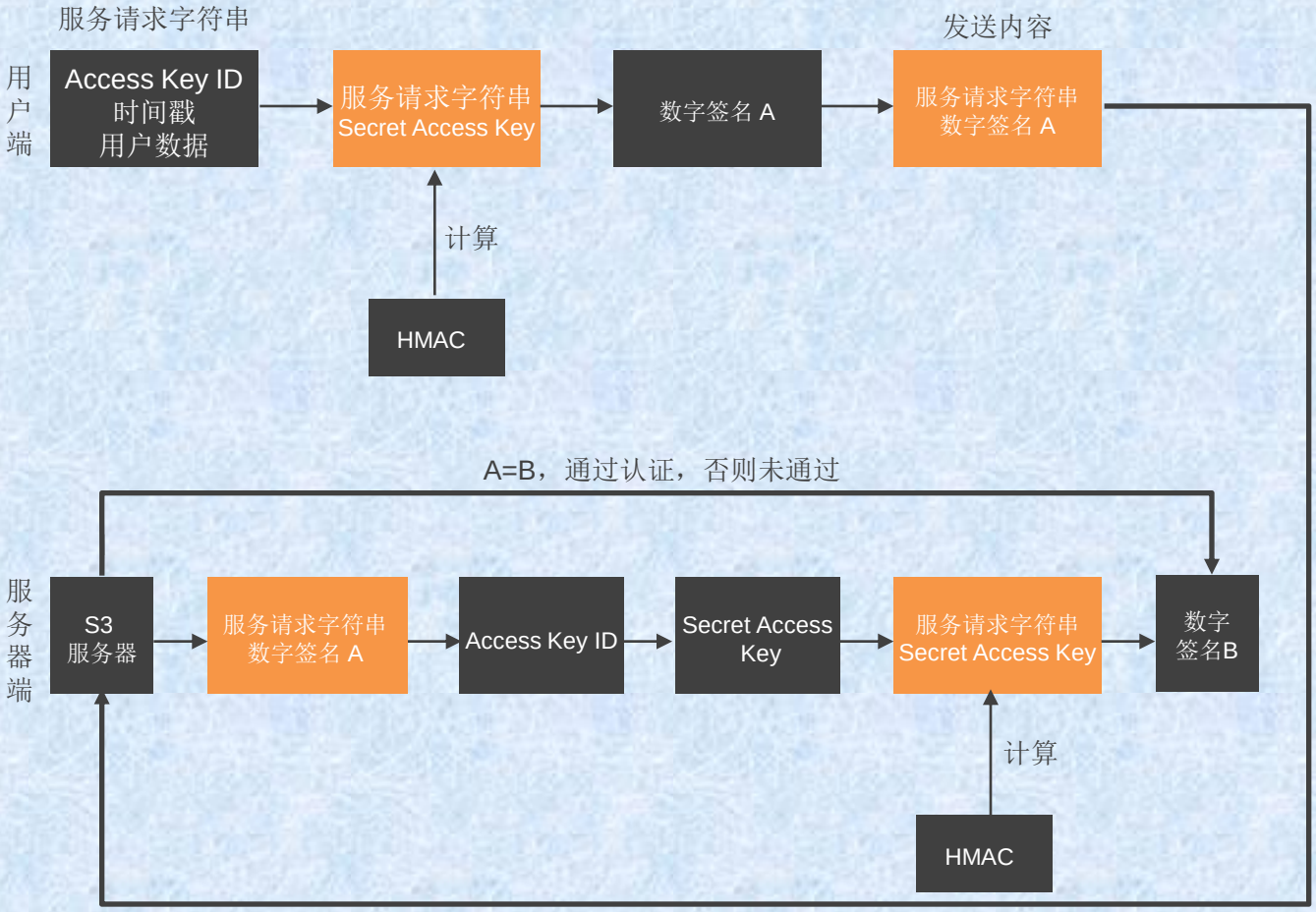
目录

23.3 简单存储服务S3

23.3 简单存储服务S3

- S3的安全措施

身份认证（Authentication）


服务请求字符串
用户端
Access Key ID
时间戳
用户数据
服务请求字符串
Secret Access Key
数字签名 A
发送内容
服务请求字符串
数字签名 A
计算
HMAC
A=B，通过认证，否则未通过
服务器端
S3
服务器
服务请求字符串
数字签名 A
Access Key ID
Secret Access Key
服务请求字符串
Secret Access Key
数字
签名B
计算
HMAC

23.3 简单存储服务S3

- **S3的安全措施**

访问控制列表（Access Control List）

访问控制列表是S3提供的可供用户自行定义的访问控制策略列表。S3的访问控制策略（ACP）提供如下所列的五种访问权限。

权　限	允许操作目标	具体权限内容
READ	桶	列出已有桶
	对象	读取数据及元数据
WRITE	桶	创建、覆写、删除桶中对象
READ_ACP	桶	读取桶的ACL
	对象	读取对象中的ACL
WRITE_ACP	桶	覆写桶的ACP
	对象	覆写对象的ACP
FULL_CONTROL	桶	允许进行以上所有操作，是S3提供的最高权限
	对象	

23.3 简单存储服务S3

- **S3的安全措施**

S3中有三大类型的授权用户

所有者（Owner）

所有者是桶或对象的创建者，默认具是WRITE_ACP权限。
所有者默认就是最高权限拥有者。

个人授权用户（User）

两种授权方式，一种是通过电子邮件地址授权的用户，
另一种是通过用户ID进行授权。

组授权用户（Group）

一种是AWS用户组，它将授权分发给所有AWS账户拥有者
另一种是所有用户组，这是一种有着很大潜在危险的授权方式。

目录

目录

23.4 非关系型数据库服务SimpleDB和DynamoDB

23.4 非关系型数据库服务SimpleDB和DynamoDB

- 非关系型数据库与传统关系数据库的比较

	传统的关系数据库	非关系型数据库
数据模型	对数据有严格的约束	key和value可以使用任意的数据类型
数据处理	满足CAP原则的C和A，在P方面很弱	满足CAP原则的A和P，而在C方面比较弱
接口层	以SQL语言对数据进行访问的，提供了强大的查询功能，并便于在各种关系数据库间移植	通过API操作数据，支持简单的查询功能，且由于不同数据库之间API的不同而造成移植性较差

23.4 非关系型数据库服务SimpleDB和DynamoDB

- 非关系型数据库与传统关系数据库的比较

总结：

关系型数据库	优点	具有高一致性，在ACID方面很强，移植性很高
	缺点	可扩展性方面能力较弱
非关系型数据库	优点	具有很高的可扩展性，具有很好的并发处理能力
	缺点	缺乏数据一致性保证，处理事务性问题能力较弱 难以处理跨表、跨服务器的查询

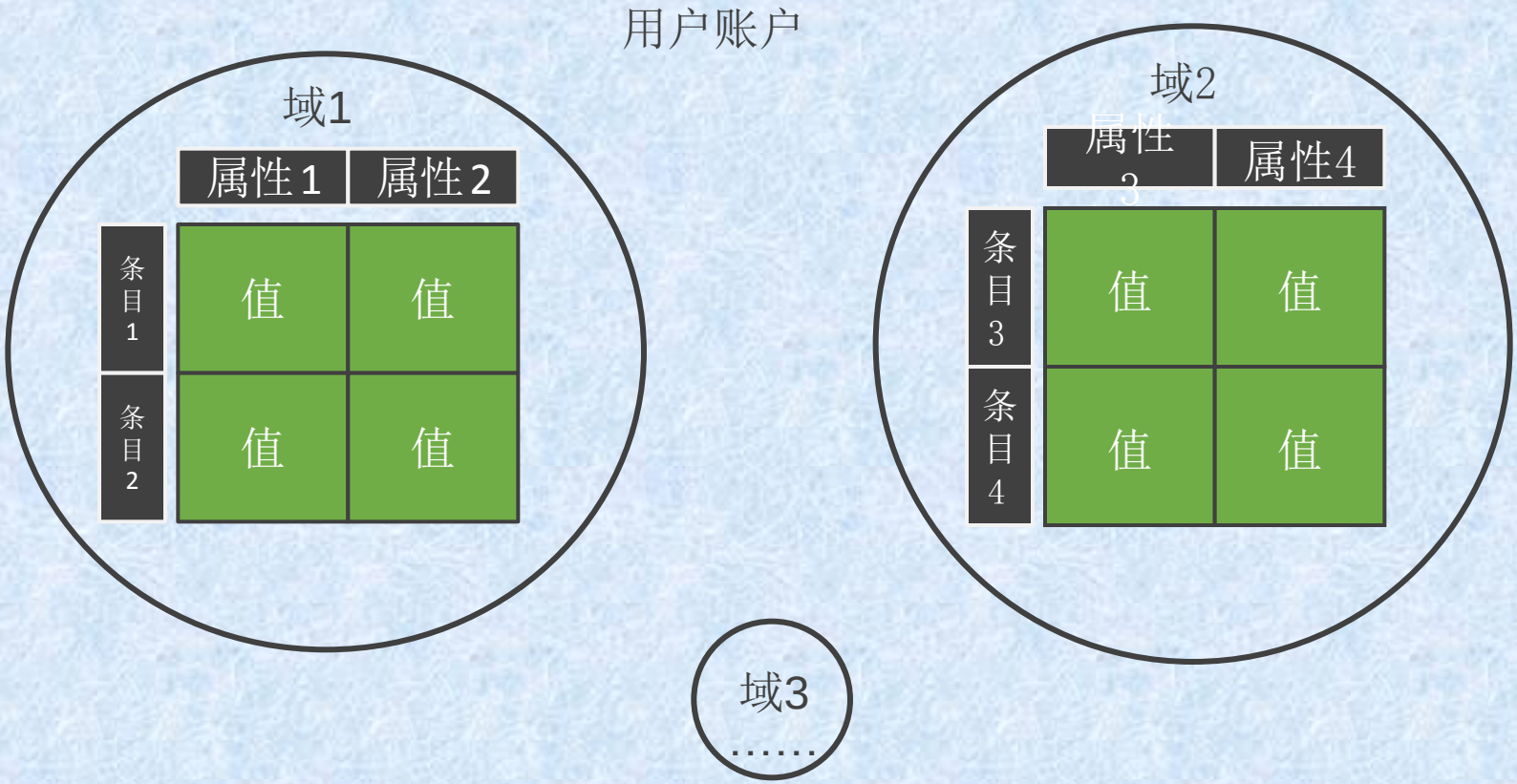
目录

23.4 非关系型数据库服务SimpleDB和DynamoDB

23.4 非关系型数据库服务SimpleDB和DynamoDB

- **SimpleDB**

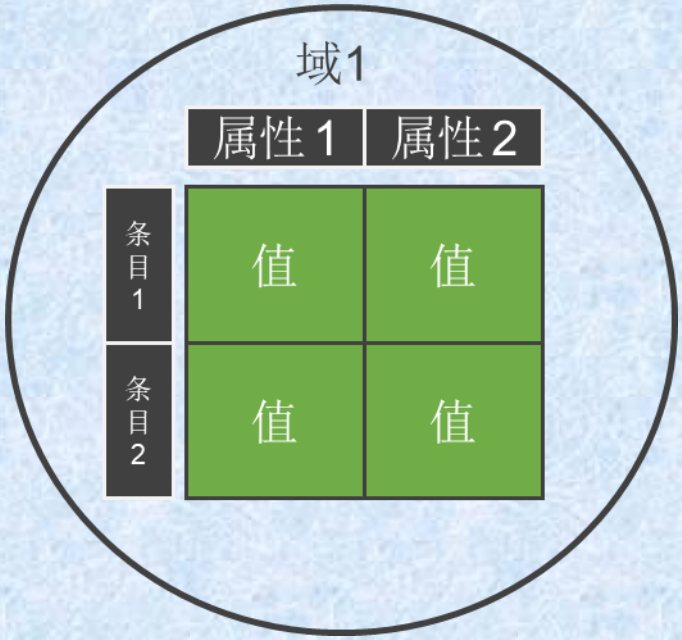
SimpleDB基本结构图如下，包含了域、条目、属性、值等概念。



23.4 非关系型数据库服务SimpleDB和DynamoDB

- SimpleDB

1.域（Domain）



域是用于存放具有一定关联关系的数据的**容器**，其中的数据以**UTF-8编码**的字符串形式存储。

每个用户账户中的**域名**必须是**唯一**的，且域名长度为**3～255个字符**。

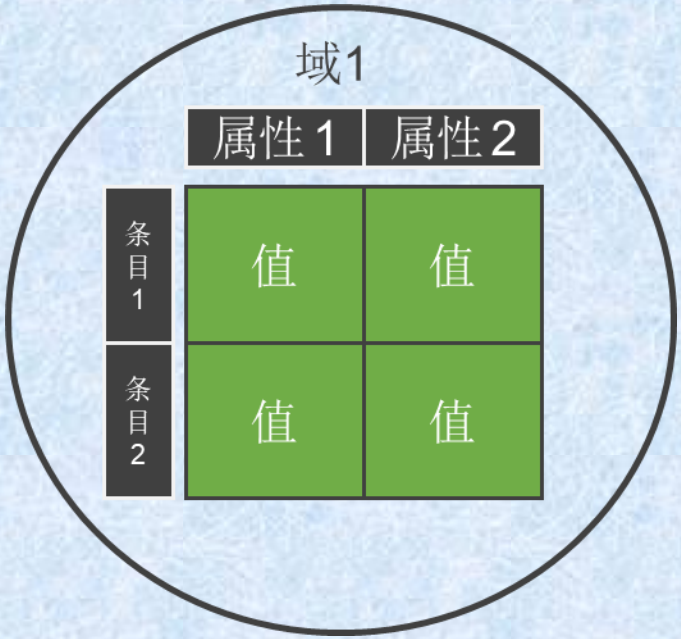
每个域中**数据的大小**具有一定的**限制**。

但域的划分也会为**数据操作**带来一些**限制**，是否划分域需要**综合多种因素考虑**。

23.4 非关系型数据库服务SimpleDB和DynamoDB

- SimpleDB

2.条目（Item）



条目对应着一条**记录**，通过一系列属性来描述，即条目是**属性的集合**。

在每个域中，**条目名**必须是**唯一**的

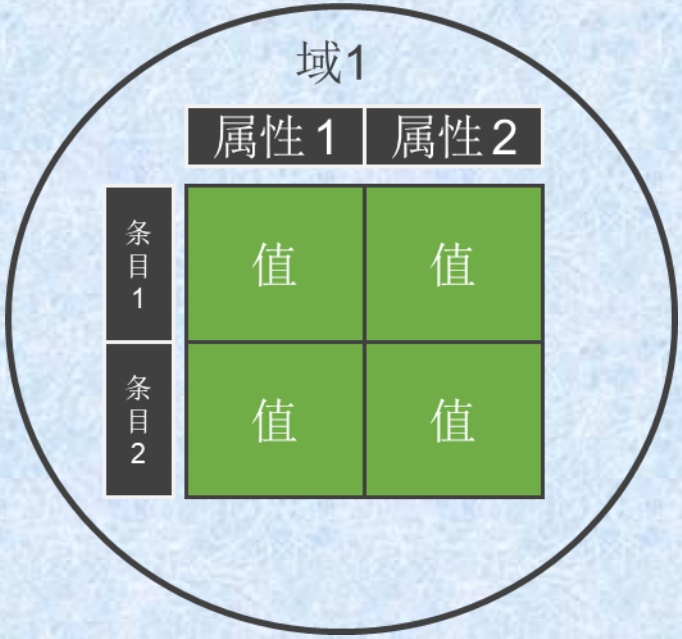
与关系数据库不同，SimpleDB中**不需要事先定义条目的模式**，即条目由哪些属性来描述。

操作上具有极大的**灵活性**，用户可以随时**创建**、**删除**以及**修改条目**的内容

23.4 非关系型数据库服务SimpleDB和DynamoDB

- SimpleDB

3.属性（Attribute）



属性是条目的特征，每个属性都用于对条目某方面特性进行**概括性描述**。

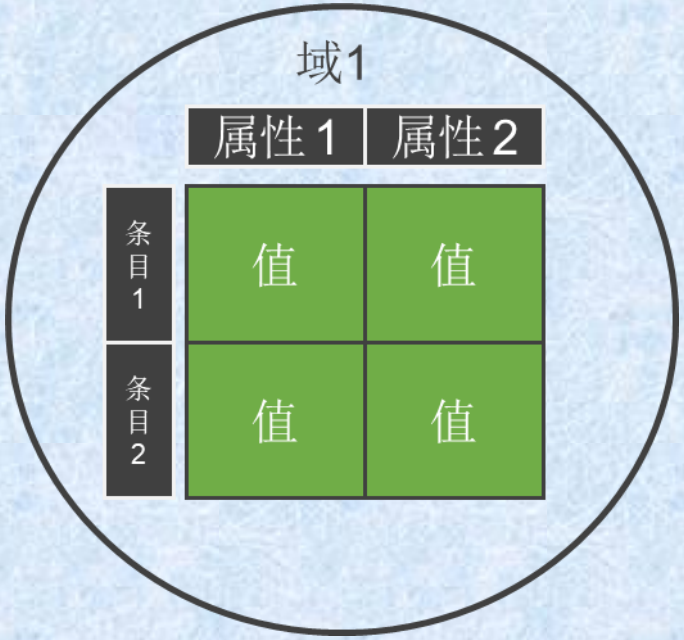
每个条目可以有**多个属性**。

属性的操作**相对自由**，不用考虑该属性是否与域中的其他条目相关。

23.4 非关系型数据库服务SimpleDB和DynamoDB

- SimpleDB

4.值（Value）



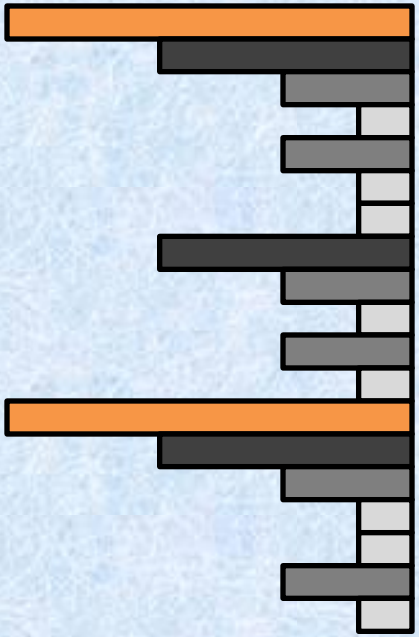
值用于描述某个条目在某个属性上的具体内容

一个条目的一个属性中可以有多个值。

例如：某类商品除颜色外其他参数完全一致，此时可以通过在颜色属性中存放多个值来使用一个条目表示该商品，而不需要像关系数据库中那样建立多条记录。

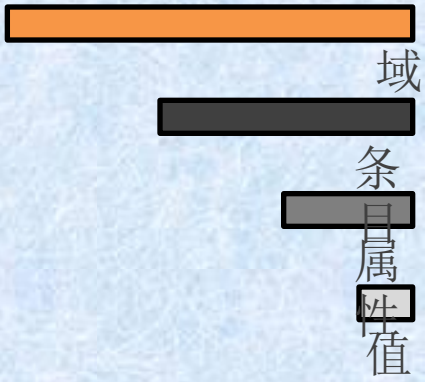
23.4 非关系型数据库服务SimpleDB和DynamoDB

- SimpleDB



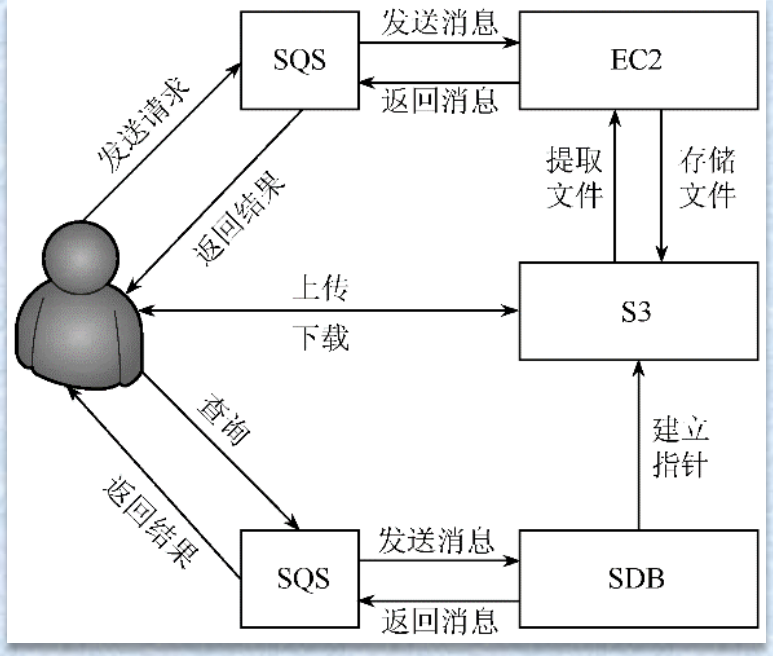
如图显示了SimpleDB的树状组织方式，

其中可以看出SimpleDB对多值属性的支持。



23.4 非关系型数据库服务SimpleDB和DynamoDB

- SimpleDB



SimpleDB与其他AWS组件综合使用的方式

限制 SimpleDB中每个属性值的大小不能超过**1KB**

导致 SimpleDB存储的**数据范围**极其**有限**

解决 将相对大的数据存储在S3中，在SimpleDB中只保存指向某个特定文件位置的**指针**

目录

23.4 非关系型数据库服务SimpleDB和DynamoDB

23.4 非关系型数据库服务SimpleDB和DynamoDB

- **DynamoDB**

DynamoDB的特点：

1. DynamoDB以**表**为基本单位，表中的条目同样不需要预先定义的模式
2. DynamoDB中**取消**了对表中**数据大小的限制**，用户设置任意大小，并由系统自动分配到多个服务器上。
3. DynamoDB不再固定使用最终一致性数据模型，而是允许用户选择**弱一致性**或者**强一致性**。
4. DynamoDB还在硬件上进行了优化，采用**固态硬盘**作为支撑，并根据用户设定的**读/写流量**限制预设来确定数据分布的硬盘数量。

目录

23.4 非关系型数据库服务SimpleDB和DynamoDB

23.4 非关系型数据库服务SimpleDB和DynamoDB

- SimpleDB和DynamoDB的比较

SimpleDB和DynamoDB都是Amazon提供的非关系型数据库服务。

SimpleDB	限制了每张表的大小，更适合于小规模复杂的工作。自动对所有属性进行索引，提供了更加强大的查询功能。
DynamoDB	支持自动将数据和负载分布到多个服务器上，并未限制存储在单个表中数据量的大小，适用于较大规模负载的工作。

目录

目录

23.5 关系数据库服务RDS

23.5 关系数据库服务RDS

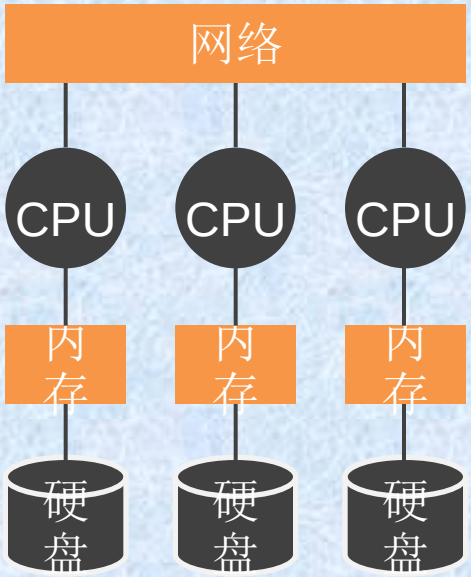
- **RDS的基本原理**

Amazon RDS将MySQL数据库移植到集群中，在一定的范围内解决了关系数据库的可扩展性问题。

-MySQL集群方式采用了**Share-Nothing**架构。

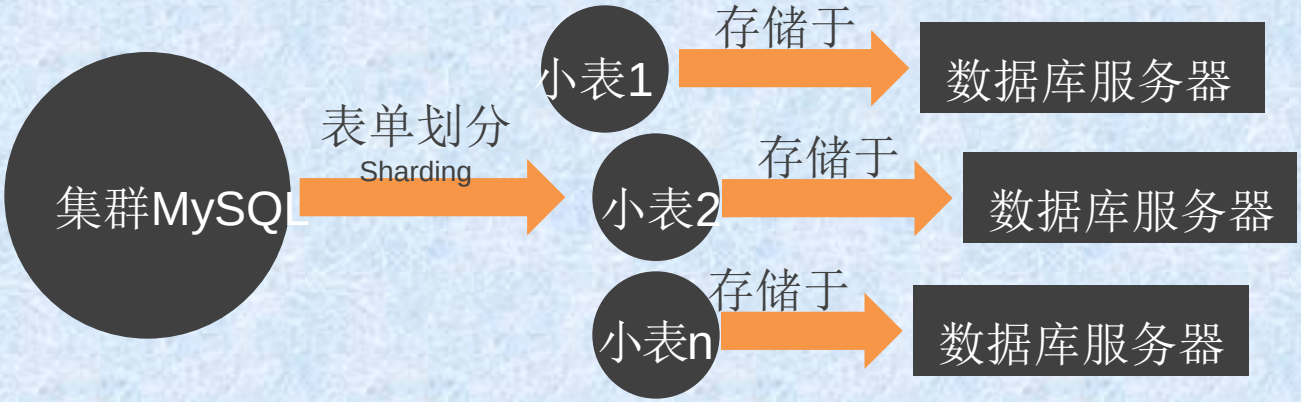
-每台数据库服务器都是完全独立的计算机系统，通过**网络相连**，**不共享任何资源**。

-这是一个具有**较高可扩展性**的架构，当数据库处理能力不足时，可以通过**增加服务器数量**来提高处理能力，同时多个服务器也增加了**数据库并发访问的能力**。



23.5 关系数据库服务RDS

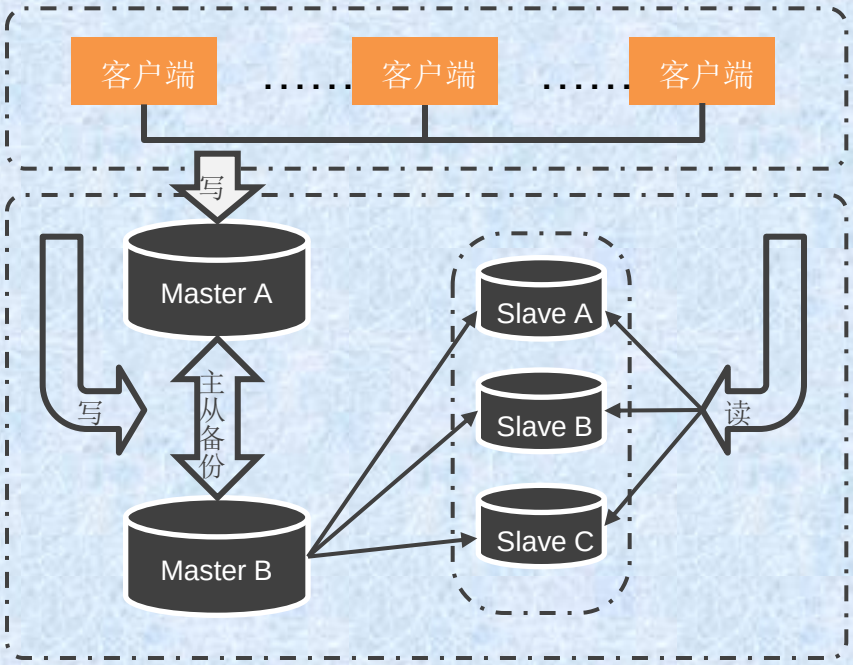
- RDS的基本原理



表单主要根据业务的需要进行针对性的划分，对数据库的管理人员提出了非常高的要求，如果划分得不科学，则查询经常会跨表单和服务器，性能就会严重下降。

23.5 关系数据库服务RDS

- RDS的基本原理



集群MySQL通过主从备份和读副本技术提高可靠性和数据处理能力。

瘫痪

升级

并发处理

目录

23.5 关系数据库服务RDS

23.5 关系数据库服务RDS

- RDS的使用

-从用户和开发者的角度来看，RDS和一个远程MySQL关系数据库没什

-Amazon将RDS中的MySQL服务器实例称做DB Instance，通过基于Web的API进行创建和管理，其余的操作可以通过标准的MySQL通信协议完成。

-创建DB Instance时还需要定义可用的存储，存储范围为5GB到1024GB，RDS数据库中表最大可以达到1TB。

-可以通过两种工具对RDS进行操作：命令行工具和兼容的MySQL客户端

-命令行工具是Amazon提供的Javamazon网站下载。MySQL客户端是可以与MySQL服务器进行通信的应用程序