《云计算导论》复习课

考试安排

时间：2021年12月21日
15:20----17:00　共100分钟

地点：瀚学楼101、瀚学楼107

考试形式：开卷

题型：单选题、不定项选择题、判断题、问答题

注意：

（1）携带身份证或学生证备查

（2）不允许使用手机、电脑等电子设备！！！

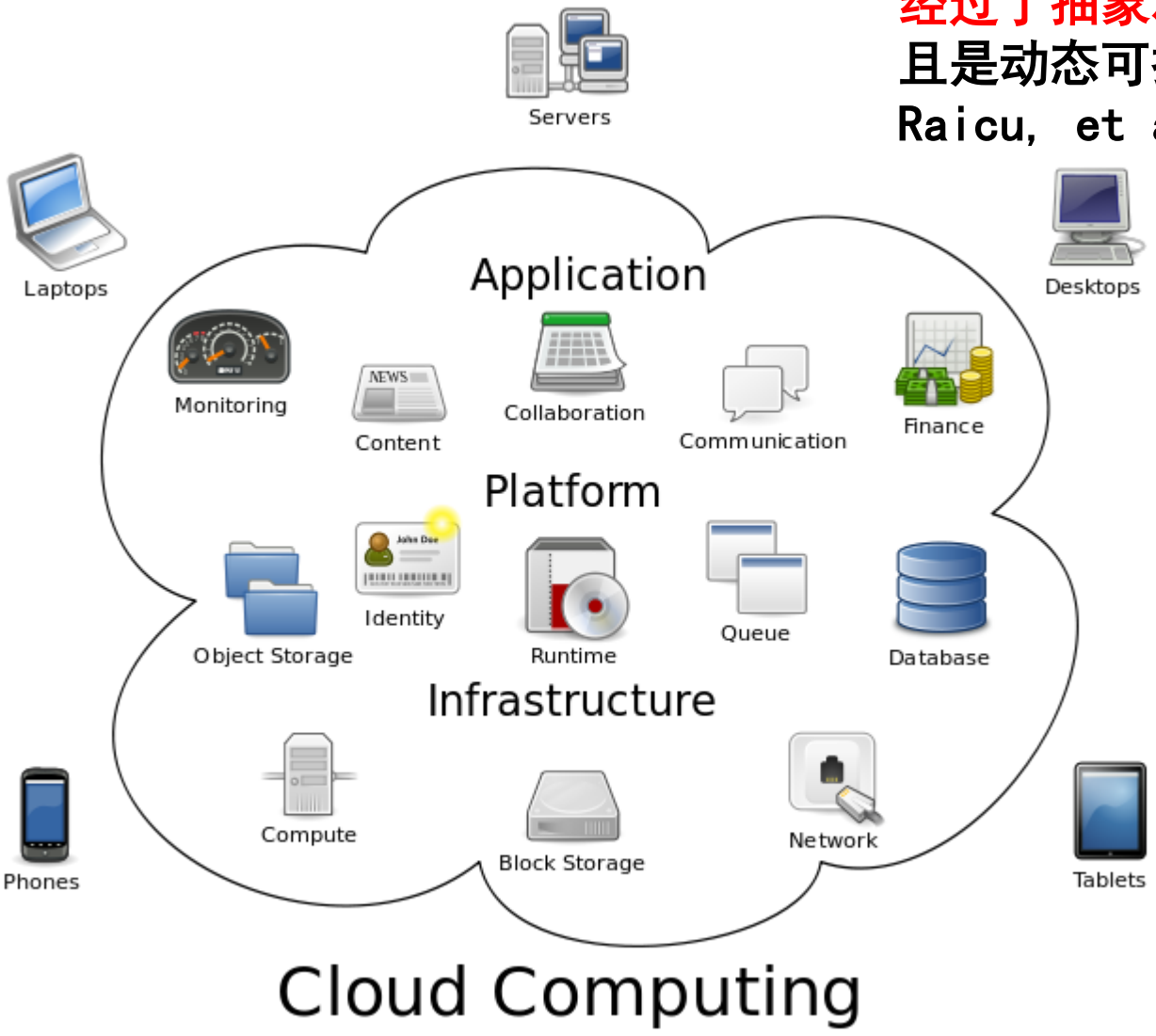
提纲

- 云计算
- 虚拟化
- 分布式文件系统
- NoSQL数据库
- MapReduce并行计算框架

(1) 云计算

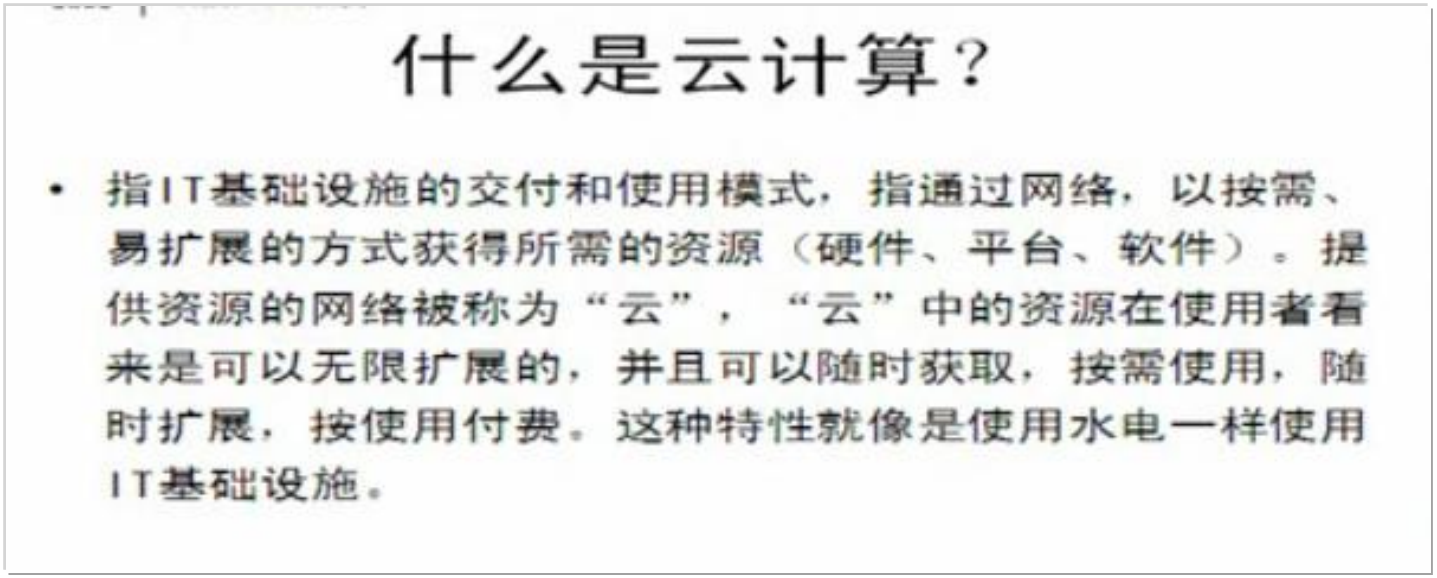
“云”是什么?

在用户看来，云计算提供了一种大规模的资源池，资源池管理的资源包括计算、存储、平台和服务等各种资源，资源池中的资源经过了抽象和虚拟化的处理，并且是动态可扩展的(Foster, Zhao, Raicu, et al., 2008)。



云计算

云计算的官方定义



通俗解释：大量it基础设施（服务器，硬件资源，软件资源，网络， 存储，运算设备）集中起来完成很多以前靠单个服务器完成不了的工作，这种it部署和应用的模式，叫云计算模式

云计算是什么?

云计算＝

互联网:((Σ资源(设备、数据、软件))×服务

我们的定义:

云计算是分布式计算的一种形式，它强调在互联网上建立大规模数据中心等IT基础设施，通过面向服务的商业模式为各类用户提供基础设施能力，是建造和运维互联网分布系统相关技术的总称。

云计算的典型特征

规模经济驱动的资源集中共享

资源虚拟化

系统动态可伸缩

面向服务的商业模式

云计算的服务模型

基础设施即服务（IaaS）

- 向客户提供处理、存储、网络以及其他基础计算资源，客户可以在上运行任意软件，包括操作系统和应用程序。
- 用户不管理或者控制底层的云基础架构，但是可以控制操作系统、存储、发布应用程序，以及可能限度的控制选择的网络组件（例如，防火墙）。
- 仅提供计算和存储等基础功能，应用的实现需用户自行完成

平台即服务（PaaS）

- 客户使用云供应商支持的开发语言和工具，开发出应用程序，发布到云基础架构上。
- 客户不管理或者控制底层的云基础架构，包括网络、服务器、操作系统或者存储设备，但是能控制发布应用程序和可能的应用程序运行环境配置

软件即服务（SaaS）

- 客户所使用的服务商提供的这些应用程序运行在云基础设施上。这些应用程序可以通过各种各样的客户端设备访问。
- 客户不管理或者控制底层的云基础架构，包括网络、服务器、操作系统、存储设备

公有云和私有云

什么是公有云？

公有云：是指为外部客户提供服务的云，他所有的服务是供别人使用的，而不是自己使用。

什么是私有云？

私有云：是指企业自己使用的云，他所有的服务不是供别人使用，而是公司及内部人员或分支机构使用。

(2) 虚拟化

虚拟化（Virtualization）

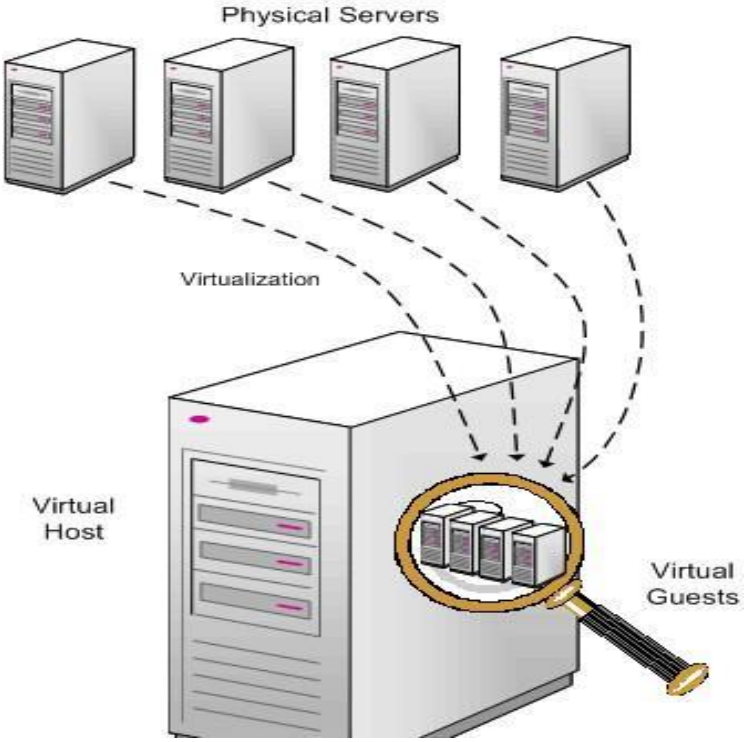
虚拟化的本质是建模问题

虚拟化是资源的逻辑表示，它不受物理限制的约束

1.虚拟化的对象是各种各样的资源

2.经过虚拟化后的逻辑资源对用户隐藏了不必要的细节

3.用户可以在虚拟环境中实现其在真实环境中的部分或全部功能



定义：虚拟化是一种仿真建模

- 一个变多个
- 多个像一个

虚拟化的分类

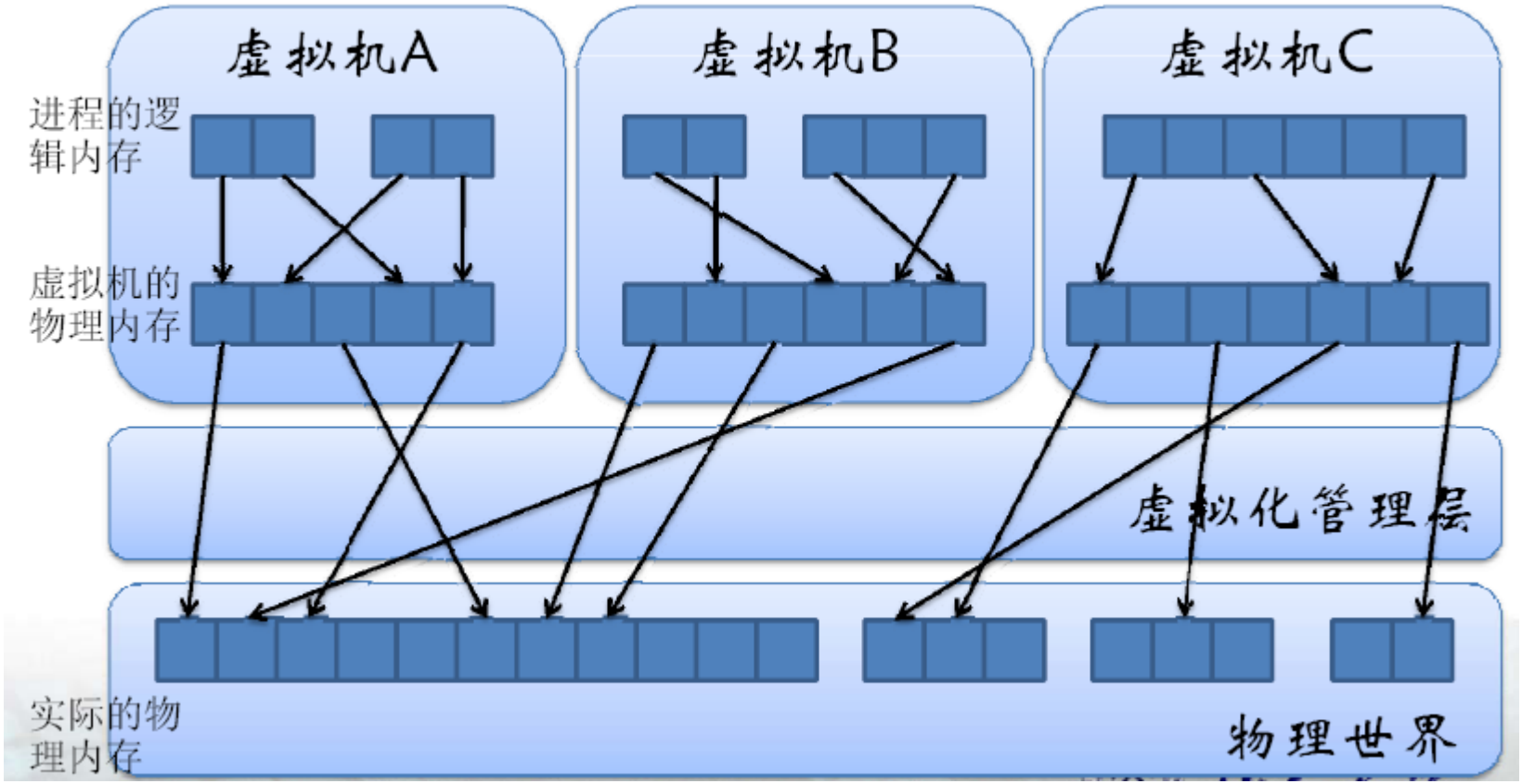
分层、抽象，实现模拟效果 ——》利用率、使用效率

- **计算虚拟化，比如一台机器上同时运行多个操作系统，每个OS的Kernel都以为独享硬件**
- **存储虚拟化：我的文件可能分散存放到了不同位置**
- **应用程序虚拟化：例如在同一台机器上同时运行office2003和office2010**

“软件定义（Software-defined）存储”（存储虚拟化最新发展趋势）

- Windows Server 2012 R2
- 提供RAID配置
- 存储分层等服务
- 软件定义网络：在软件里定义公司或机构的所有的网络（浅）；分层、建抽象、通过补充额外的智能算法等定义附加智能（深）

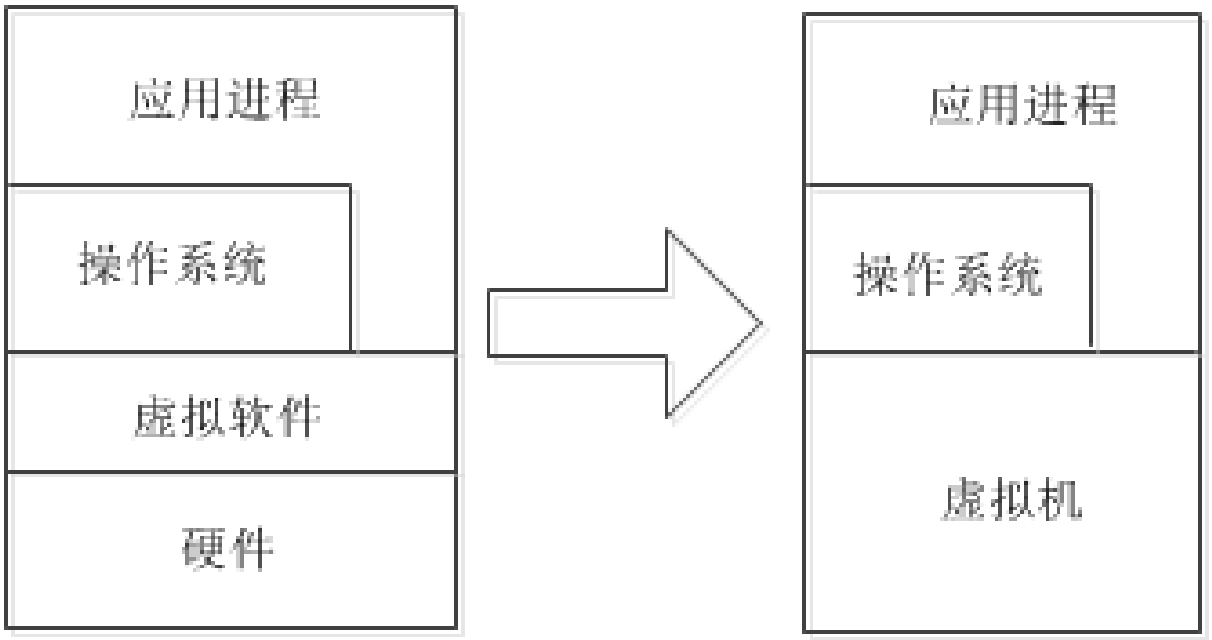
虚拟机



计算虚拟化

虚拟机是什么？

- 虚拟机：是指通过虚拟化软件套件模拟的、具有完整硬件功能的、运行在一个隔离环境中的逻辑计算机系统。



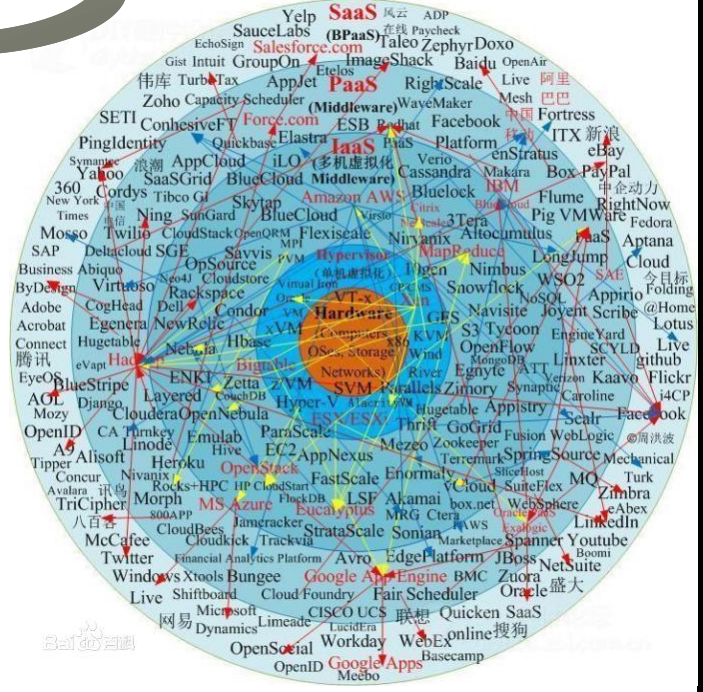
(3) 分布式文件系统

什么是大数据？

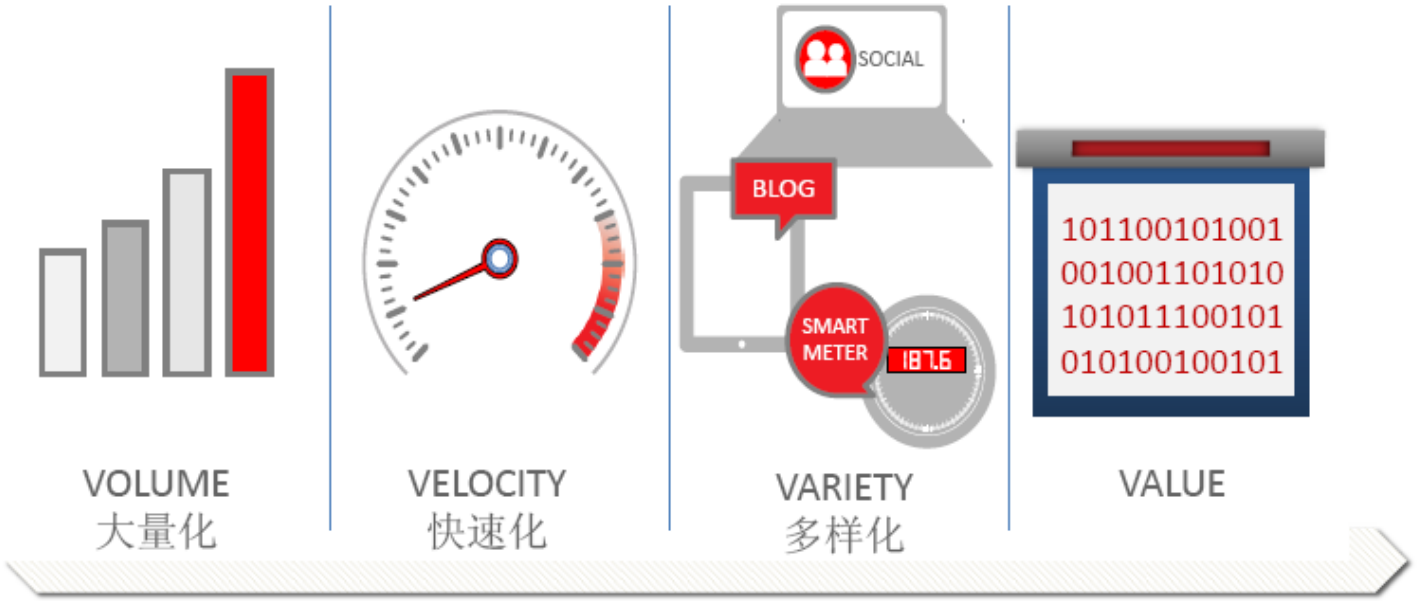
•目前对大数据尚未有一个公认的定义，不同的定义基本上是从特征出发，试图给出大数据的定义。

什么是“大数据”？

- 大数据很抽象，表示数据规模的庞大。
- 大数据泛指巨量的数据集，因可从中挖掘出有价值的信息而受到重视。
- 《维基百科》将**大数据是指利用常用软件工具捕获、管理和处理数据所耗时间超过可容忍时间的数据集。**。



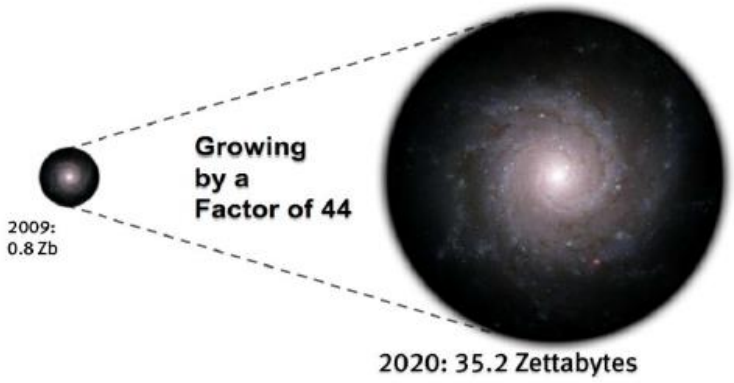
大数据特征



大数据不仅仅是数据的“大量化”，而是包含“快速化”、“多样化”和“价值化”等多重属性。

1. 数据量大

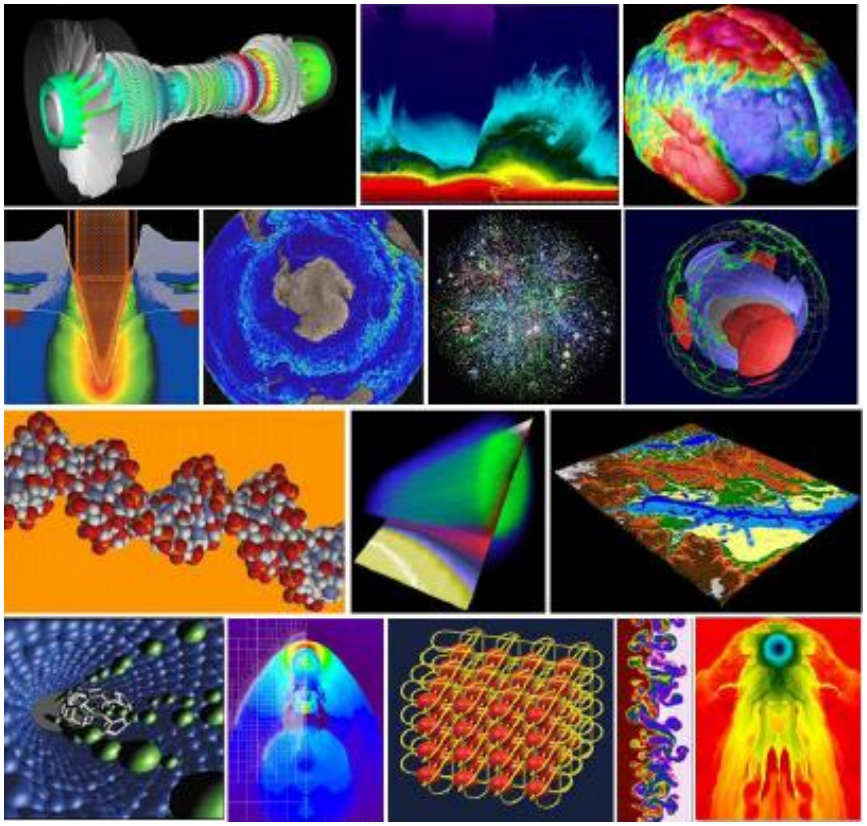
- 根据IDC作出的估测，数据一直都在以每年50%的速度增长，也就是说每两年就增长一倍（大数据摩尔定律）
- 人类在最近两年产生的数据量相当于之前产生的全部数据量
- 预计到2020年，全球将总共拥有35ZB的数据量，相较于2010年，数据量将增长近30倍



TERABYTE	10 的 12 次方	一块 1TB 硬盘		200,000 照片或 mp3 歌曲
PETABYTE	10 的 15 次方	两个数据中心机柜		16 个 Blackblaze pod 存储单元
EXABYTE	10 的 18 次方	2,000 个机柜		占据一个街区的 4 层数据中心
ZETTABYTE	10 的 21 次方	1000 个数据中心		纽约曼哈顿的 1/5 区域
YOTTABYTE	10 的 24 次方	一百万个数据中心		特拉华州和罗德岛州

2. 数据类型繁多

- 大数据是由结构化和非结构化数据组成的
  - 10%的结构化数据，存储在数据库中
  - 90%的非结构化数据，它们与人类信息密切相关



- 科学研究
  - 基因组
  - LHC 加速器
  - 地球与空间探测
- 企业应用
  - Email、文档、文件
  - 应用日志
  - 交易记录
- 物联网数据
  - 交通
  - 智能制造
  - 智慧城市
- 互联网数据
  - 查询日志/点击流
  - Twitter/ Blog / SNS
  - Wiki

3. 处理速度快

- 从数据的生成到消耗，时间窗口非常小，可用于生成决策的时间非常少
- 1秒定律：这一点也是和传统的数据挖掘技术有着本质的不同

4 价值密度低

- **价值密度低，商业价值高**
- 以视频为例，连续不间断监控过程中，可能有用的数据仅仅有一两秒，但是具有很高的商业价值

分布式文件系统

发展背景：互联网的发展，海量数据的存储

文件系统是操作系统用来组织磁盘文件的方法和数据结构。它通过对操作系统所管理的存储空间的抽象，向用户提供统一的、抽象化的访问接口，屏蔽对物理设备的直接操作和资源管理。

分布式文件系统（Distributed File System）是指文件系统管理的物理存储资源不一定直接连接在本地计算机上，而是通过计算机网络与计算机相连。从内部实现来看，分布式文件系统不再和普通文件系统一样负责管理本地磁盘，它的文件内容和目录结构也不存储在本地磁盘上，而是通过网络传输到远端系统上。

分布式文件系统关键问题

命名空间

命名空间使整个存储空间有全局一致的逻辑视图

元数据管理

文件元数据用来描述文件的属性信息，是文件管理的核心数据

故障和错误处理

大量商用计算机和其他网络设备会引入各种故障和错误，是分布式系统必须解决的问题

性能和效率

系统必须保证文件的处理时间在用户可接受的范围内，否则系统的应用价值将大打折扣

命名空间

单个的服务器想要组成一个统一的存储空间就需要在逻辑上有一个一致的视图。这个一致的视图由命名系统来实现，通过命名空间的全局一致来构建集群存储空间的全局一致。

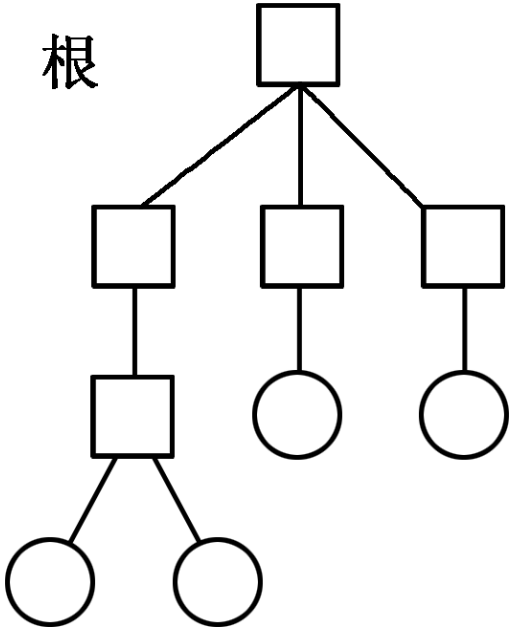
命名空间（Namespace）：一个系统中实体名称的集合，命名空间指定了名称构造的规则，界定了名称的范围，在命名空间中名称唯一地标识一个实体，名称不允许重复。

实体名称的构造方法

无结构命名：

- 命名空间的命名构造规则除了指定构成名称的符号集合之外，并没有指定名称的结构，对组成名称的单个符号并没有指定特殊的含义
- 如全局唯一标识符（Globally Unique Identifier，GUID；或者 Universally Unique Identifier，UUID），全局唯一标识符是由128位二进制符号组成的，用于在全球范围内唯一标识一个数字对象，比如一个注册表项、一个软件组件、一个Word文档等。
- GUID 的格式为“xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx”，其中每个 x 是 0-9 或 a-f 范围内的一个十六进制数。例如：6F9619FF-8B86-D011-B42D-00C04FC964FF 即为有效的 GUID 值。

实体名称的构造方法



结构化命名

- 结构化命名方法除了指定名称的符号集之外，还对名称的结构做了规定
- 最常见的结构化命名方式是分层表示，比如计算机文件系统的命名和互联网的域名系统。

以C:\Program Files\Office\word.exe为例，C:为根节点，Program Files为其子树节点，而Office是Program Files的子树节点，文件名word.exe是Office的叶节点。

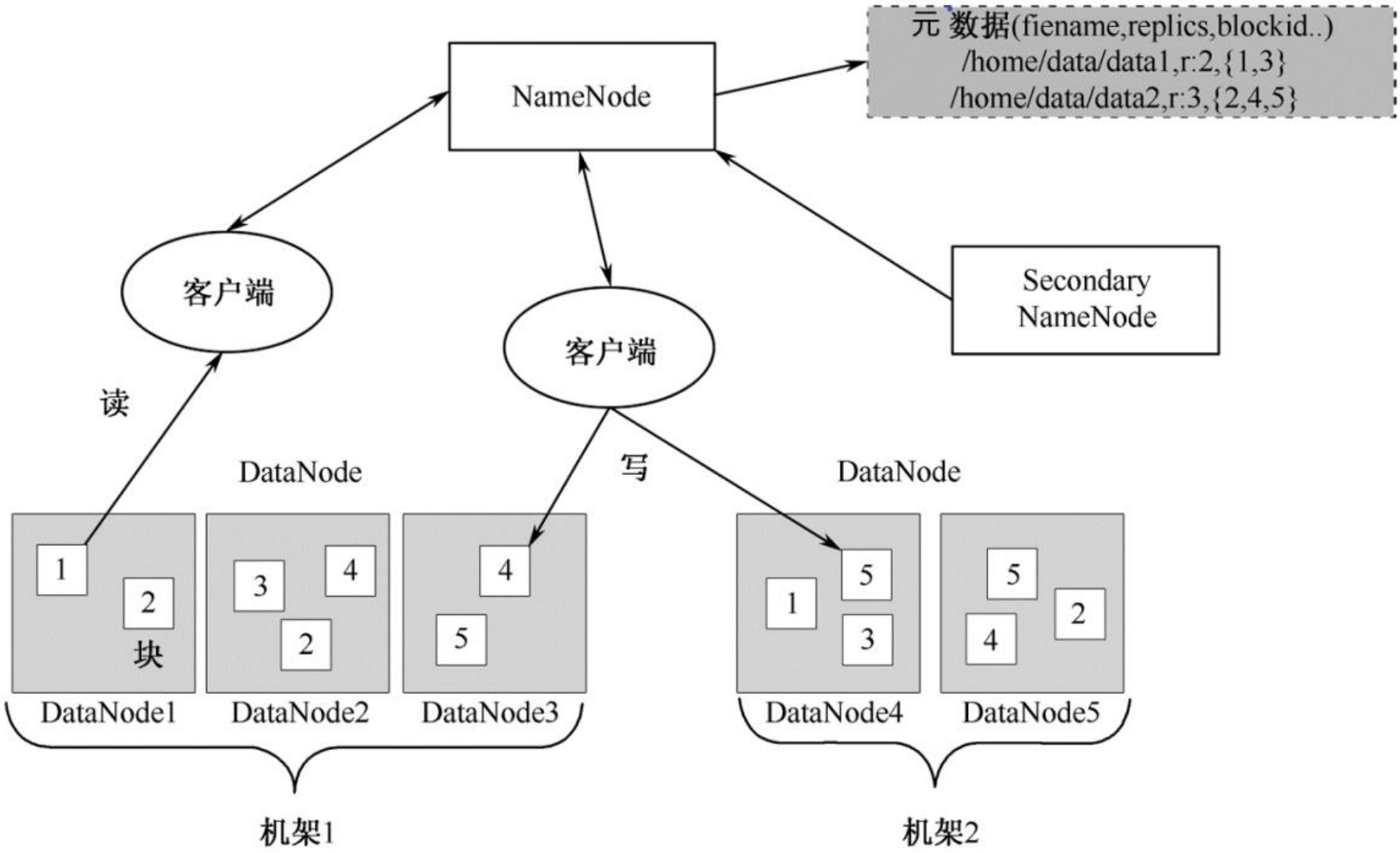
云文件系统的元数据类型

文件和数据块的命名空间：这部分数据主要记录目录结构、文件名称和数据块的名称。

文件到数据块的映射信息。一个文件通常由多个数据块组成，这些数据块按存储策略会比较均匀地分布在整个集群的存储空间中。这部分元数据记录组成一个文件的所有数据块以及文件大小等信息。

每个数据块的备份的位置。记录每个数据块的存储位置，以及它的备份的位置。

HDFS的体系结构


元 数据(fiename,replics,blockid..)
/home/data/data1,r:2,{1,3}
/home/data/data2,r:3,{2,4,5}
NameNode
客户端
客户端
Secondary
NameNode
读
写
DataNode
DataNode
1
2
块
3
4
2
4
5
1
5
3
5
2
4
DataNode1
DataNode2
DataNode3
DataNode4
DataNode5
机架1
机架2

HDFS架构——概念

- **Block： 一个文件分块，默认64M**
- **NameNode：保存整个文件系统的目录信息，文件信息及文件相应的分块信息**
- **DataNode：用于存储Blocks**
- **HDFS的HA策略：NameNode一旦宕机，整个文件系统将无法工作。如果NameNode中的数据丢失，整个文件系统就丢失了。**

NAMENODE和DATANODE

HDFS集群有两类节点，以管理者—工作者(master-worker)模式运行

Namenode名称节点：管理者

●namenode管理文件系统的命名空间。它维护着文件系统树以及整颗树内所有的文件和目录。执行命名空间操作，如打开、关闭、重命名文件或目录

●namenode也记录着每个文件中各个块所在的数据节点信息。

Datanode数据节点：工作者

datanode是文件系统的工作节点。存储并检索数据块（受客户端或namenode调度），处理客户端的文件读写请求，并且定期向namenode发送它们所存储的块的列表。

云存储

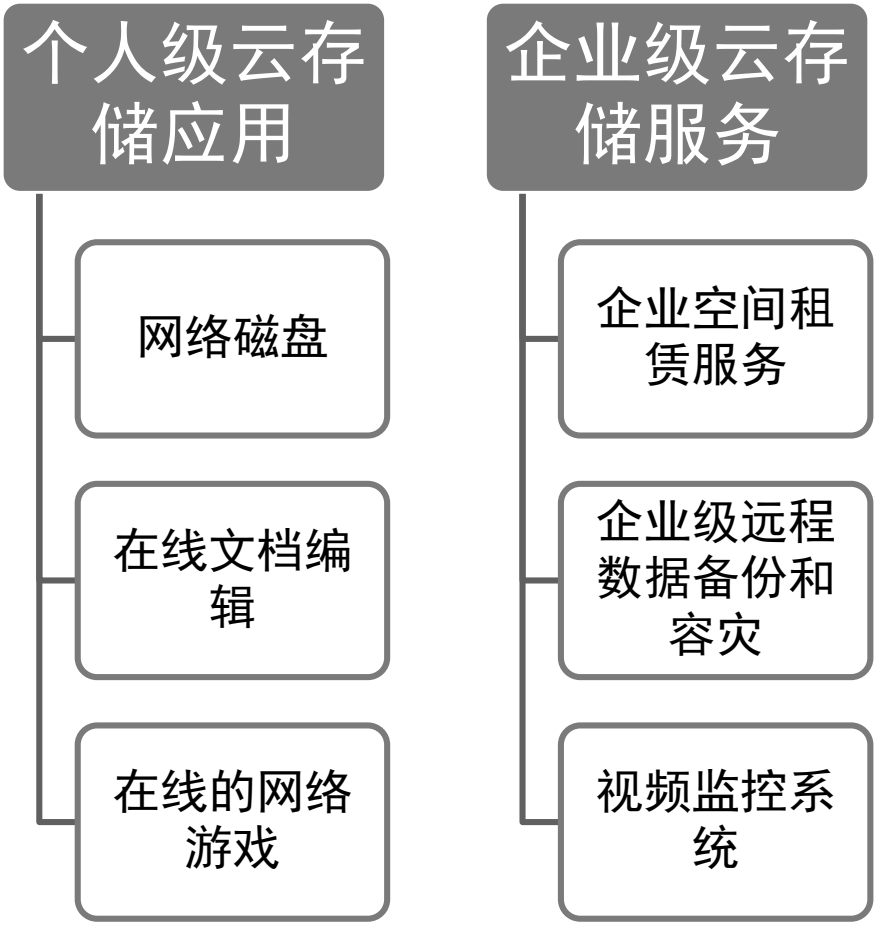
云存储的概念与云计算类似，它是指通过集群应用、网格技术或分布式文件系统等功能，将网络中大量各种不同类型的存储设备通过应用软件集合起来协同工作，共同对外提供数据存储和业务访问功能的一个系统。

云存储主要特点：

- 超强可扩展性
- 不受具体地理位置所限
- 基于商业组件
- 按照使用收费（如每G 收15 美分）
- 可跨不同应用等。

云存储不是存储，而是服务！

云存储应用


个人级云存储应用
网络磁盘
在线文档编辑
在线的网络游戏
企业级云存储服务
企业空间租赁服务
企业级远程数据备份和容灾
视频监控系统

(4) NoSQL数据库

NoSQL数据库类型–基于数据模型分类

- **KEY/VALUE数据模型**
  - 键值对数据库
- **KEY/COLUMN数据模型**
  - 列存储数据库
- **KEY/DOCUMENT数据模型**
  - 文档型数据库
- **图数据模型**
  - 图关系数据库

NOSQL数据库的基本原理

CAP定理

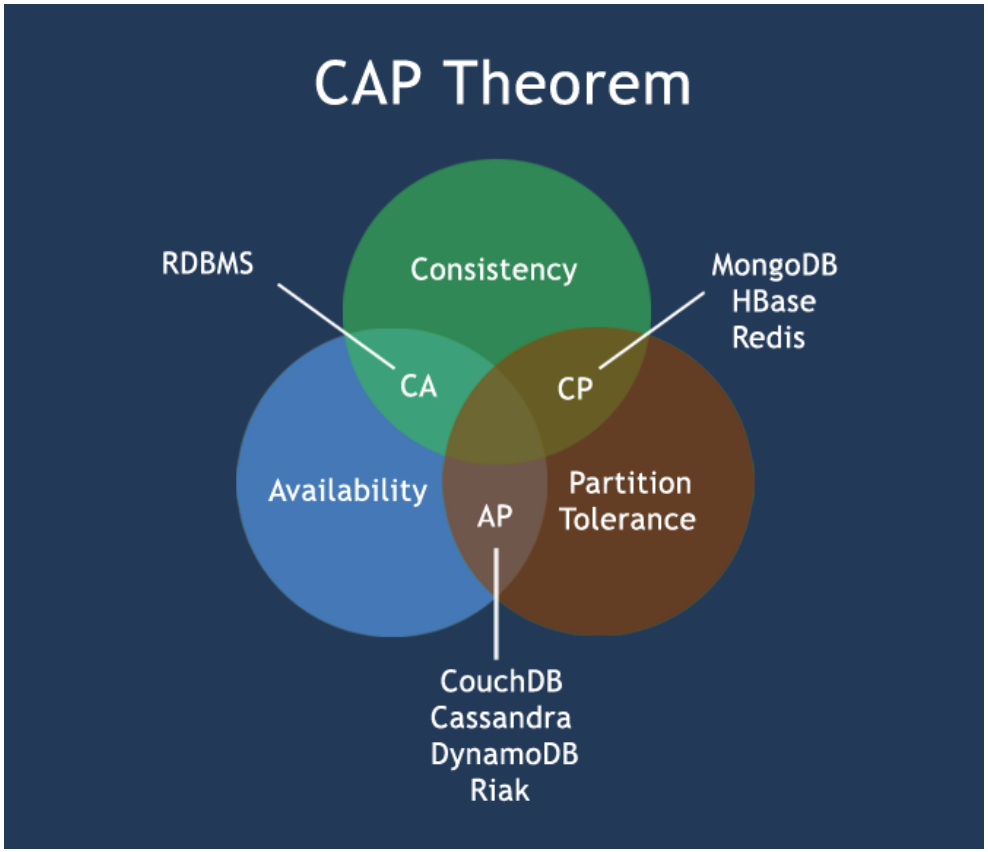
- CAP理论起源于Eric Brewer在2000年的分布式计算原则研讨会（Symposium on Principles of Distributed Computing（PODC））上提出的一个猜想。
- 2002年，麻省理工学院（MIT）的Seth Gilbert和Nancy Lynch发表了Brewer猜想的证明，使之成为一个定理。

Consistency	Availability	Tolerance of Network Partition
• 分布式环境多个节点上副本的数据是一致	• 可以在确定的时间内返回操作结果	• 系统中的一部分节点无法和其他节点进行通信时，系统仍然正常工作

“鱼和熊掌不可兼得”

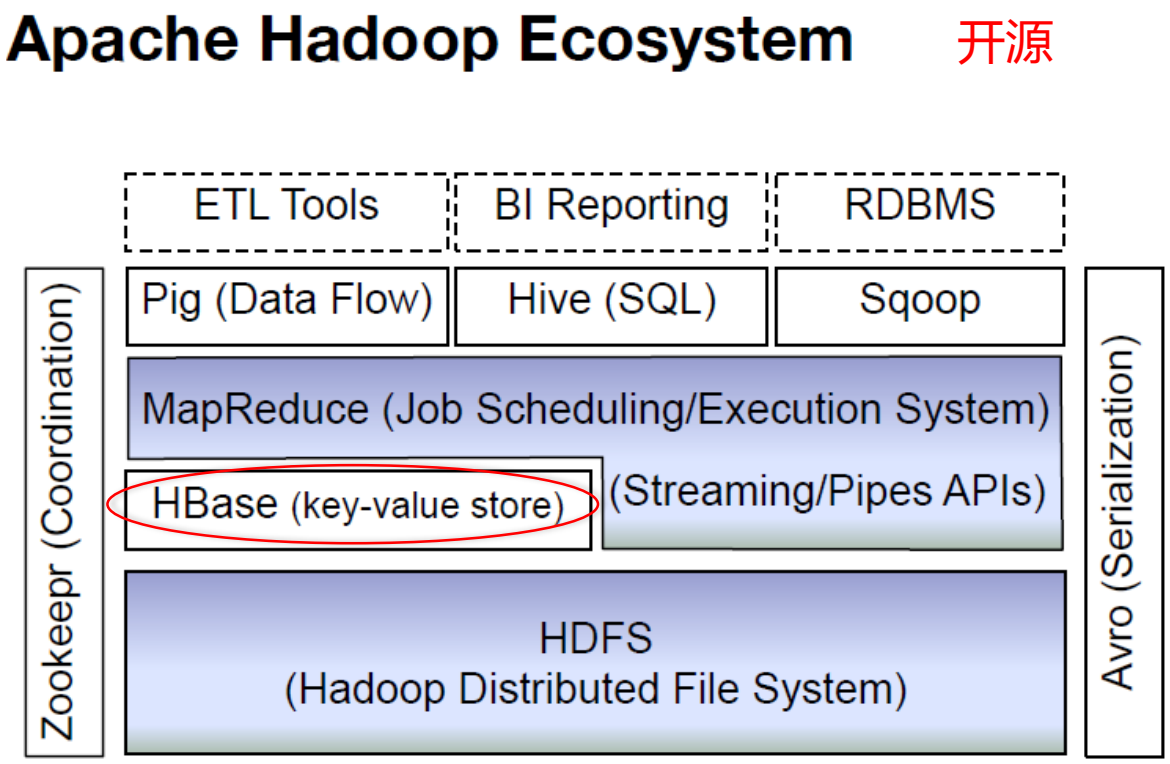
CAP定理

一个分布式系统不可能同时满足一致性、可用性和分区容忍性这三个需求，最多只能同时满足其中两个。



HBASE

HBase在Apache Hadoop生态中的位置



HBASE

HBase特点

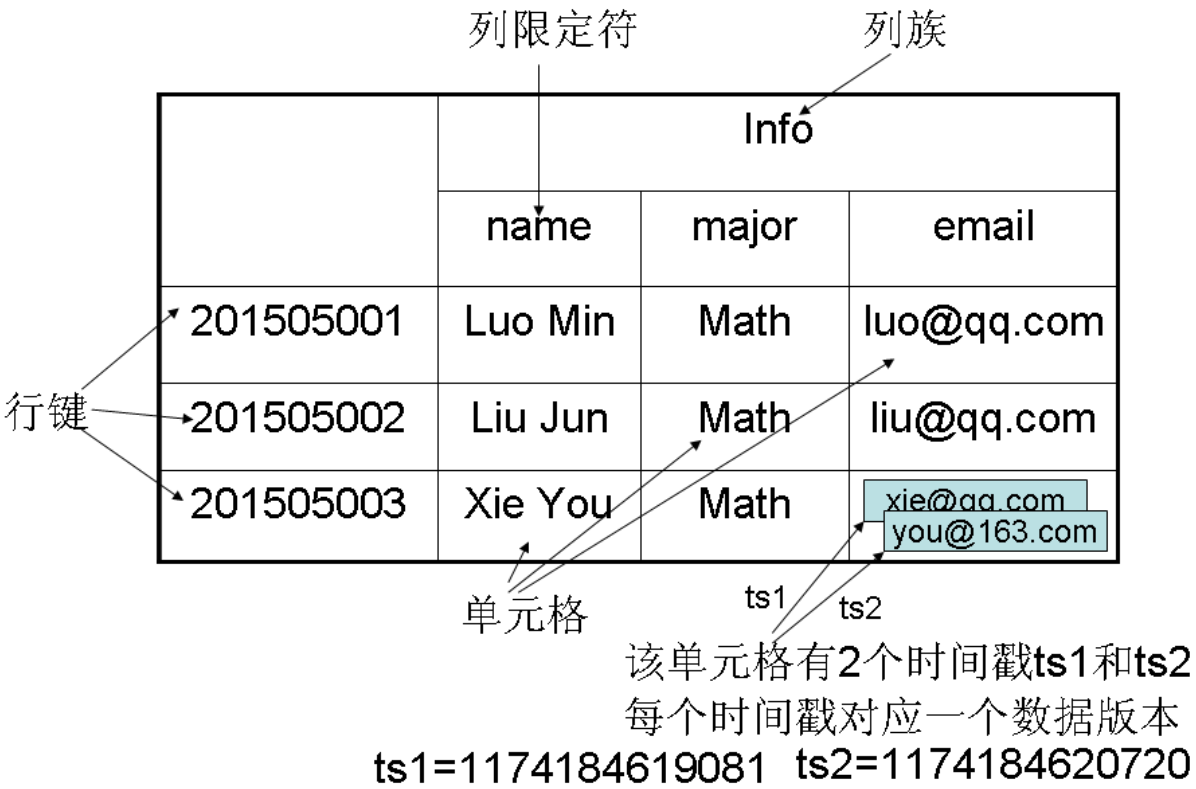
- HBase(Hadoop Database)是谷歌BigTable的开源实现，一个分布式存储系统。
  - ✓ 面向列（Column-Oriented）
  - ✓ 适合存储海量非结构化数据或半结构化数据
  - ✓ 具备高可靠性
  - ✓ 可灵活扩展伸缩
  - ✓ 高性能(支持实时数据读写)
  - ✓ 依赖于HDFS

HBase逻辑模型

HBase Table			
	Column Family A		Column Family B
	Qualifier 1	Qualifier 2	Qualifier 3
Row Key 1	*Value*		
Row Key 2		*Value*	*Value*
Row Key 3	*Value*		

HBASE

HBase 逻辑模型



(5) MAP REDUCE 并行计算框架

MAPREDUCE

问题与需求：

如何对巨量的Web文档建立索引、根据网页链接计算网页排名，从上百万文档中训练垃圾邮件过滤器，运行气象模拟，数十亿字符串的排序？

解决方案：

MapReduce将为你提供一个强大的分布式计算环境和构架，让你仅需关注你的应用问题本身，编写很少的程序代码即可完成看似难以完成的任务！

什么是MapReduce？

MapReduce是Google公司发明的一种面向大规模海量数据处理的高性能并行计算平台和软件编程框架，是目前最为成功和最易于使用的大规模海量数据并行处理技术，广泛应用于搜索引擎（文档倒排索引，网页链接图分析与页面排序等）、Web日志分析、文档分析处理、机器学习、机器翻译等各种大规模数据并行计算应用领域

MapReduce简介

基于集群的高性能并行计算平台(Cluster Infrastructure)

允许用市场上现成的普通PC或性能较高的刀架或机架式服务器，构成一个包含数千个节点的分布式并行计算集群

并行程序开发与运行框架(Software Framework)

提供了一个庞大但设计精良的并行计算软件构架，能自动完成计算任务的并行化处理，自动划分计算数据和计算任务，在集群节点上自动分配和执行子任务以及收集计算结果，将数据分布存储、数据通信、容错处理等并行计算中的很多复杂细节交由系统负责处理，大大减少了软件开发人员的负担

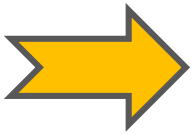
并行程序设计模型与方法(Programming Model & Methodology)

借助于函数式语言中的设计思想，提供了一种简便的并行程序设计方法，用Map和Reduce两个函数编程实现基本的并行计算任务，提供了完整的并行编程接口，完成大规模数据处理

典型的批量大数据问题的特征

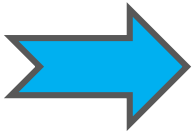
- 大量数据记录/元素进行重复处理
- 对每个数据记录/元素作特定处理

获取感兴趣的中间结果信息



Map

- 排序和整理中间结果以利后续处理
- 收集整理中间结果
- 产生最终结果输出



Reduce

关键思想：为大数据处理过程中的两个主要处理操作提供一种抽象机制

MAP & REDUCE

MapReduce借鉴了函数式程序设计语言Lisp中的思想，定义了如下的Map和Reduce两个抽象的编程接口，由用户去编程实现:

- map: (k1; v1) → [(k2; v2)]
- 输入：键值对(k1; v1)表示的数据
- 处理：文档数据记录(如文本文件中的行，或数据表格中的行)将以“键值对”形式传入map函数；map函数将处理这些键值对，并以另一种键值对形式输出处理的一组键值对中间结果[(k2; v2)]
- 输出：键值对[(k2; v2)]表示的一组中间数据

MAP & REDUCE

- **reduce**: (k2; [v2]) → [(k3; v3)]
- 输入： 由map输出的一组键值对[(k2； v2)] 将被进行合并处理将同样主键下的不同数值合并到一个列表[v2]中，故reduce的输入为(k2; [v2])
- 处理：对传入的中间结果列表数据进行某种整理或进一步的处理, 并产生最终的某种形式的结果输出[(k3; v3)] 。
- 输出：最终输出结果[(k3; v3)]

- Map和Reduce为程序员提供了一个清晰的操作接口抽象描述

WordCount：文档词频统计

设有4组原始文本数据：

Text 1: the weather is good　　Text 2: today is good

Text 3: good weather is good　　Text 4: today has good weather

传统的串行处理方式(Java)：

```
String[] text = new String[]
 { "hello world", "hello every one", "say hello to everyone in the world"
 } ;
HashTable ht = new HashTable();
for(i=0; i<3; ++i)
{  StringTokenizer st = new StringTokenizer(text[i]);
   while (st.hasMoreTokens())
   {  String word = st.nextToken();
      if(!ht.containsKey(word)) {  ht.put(word, new Integer(1));  }
      else { int wc = ((Integer)ht.get(word)).intValue() +1;// 计数加1
             ht.put(word, new Integer(wc));
           }
   }
for (Iterator itr=ht.KeySet().iterator();  itr.hasNext(); )
```

输出：　good: 5;　has: 1;　is: 3;　the: 1;　today: 2;　weather: 3

WordCount：文档词频统计

使用四个map节点：

map节点1:

输入：(text1, “the weather is good”)

输出：(the, 1), (weather, 1), (is, 1), (good, 1)

map节点2:

输入：(text2, “today is good”)

输出：(today, 1), (is, 1), (good, 1)

map节点3:

输入：(text3, “good weather is good”)

输出：(good, 1), (weather, 1), (is, 1), (good, 1)

map节点4:

输入：(text3, “today has good weather”)

输出：(today, 1), (has, 1), (good, 1), (weather, 1)

WordCount：文档词频统计

使用三个reduce节点：

reduce节点1:

输入：(good, 1), (good, 1), (good, 1), (good, 1), (good, 1)

输出：(good, 5)

reduce节点2:

输入：(has, 1), (is,1), (is,1), (is, 1),

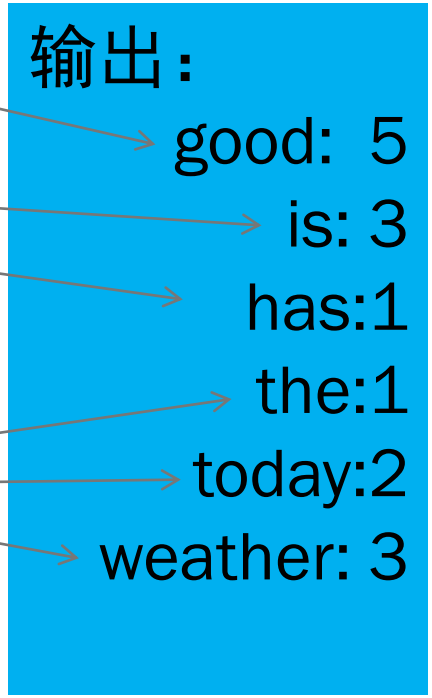
输出：(has, 1), (is, 3)

reduce节点3:

输入：(the, 1), (today, 1), (today, 1)

(weather, 1), (weather,1), (weather, 1)

输出：(the, 1), (today, 2), (weather, 3)



WordCount：文档词频统计

MapReduce伪代码（实现map和reduce两个函数）

```
Class Mapper
  method map(String input_key, String
  input_value):
    // input_key: text document name
    // input_value: document contents
```

```
Class Reducer
  method reduce(String output_key,
                Iterator
  intermediate_values):
    // output_key: a word
    // output_values: a list of counts
    int result = 0;
    for each v in intermediate_values:
      result += ParseInt(v);
```