第3章

网格计算和云计算

网络计算模式

- 课程主要内容
  - 概述
  - 企业计算
  - 网格计算和云计算
  - P2P网络、CDN网络和物联网
  - 社会计算

云计算

典型的云计算产品

- 云计算的研究吸引了不同技术领域巨头，Amazon、Google、IBM、微软和Yahoo等大公司是云计算的先行者。不同的产品对云计算理论及实现架构也有所不同。
- 云计算的先行者Google的云计算平台能实现大规模分布式计算和应用服务程序，平台包括Hadoop框架、 MapReduce分布式处理技术、分布式的文件系统GFS、结构化的BigTable存储系统以及Google其他的云计算支撑要素。
- **MapReduce分布式处理技术、分布式的文件系统GFS、结构化的BigTable存储系统**是Google的三个核心技术。

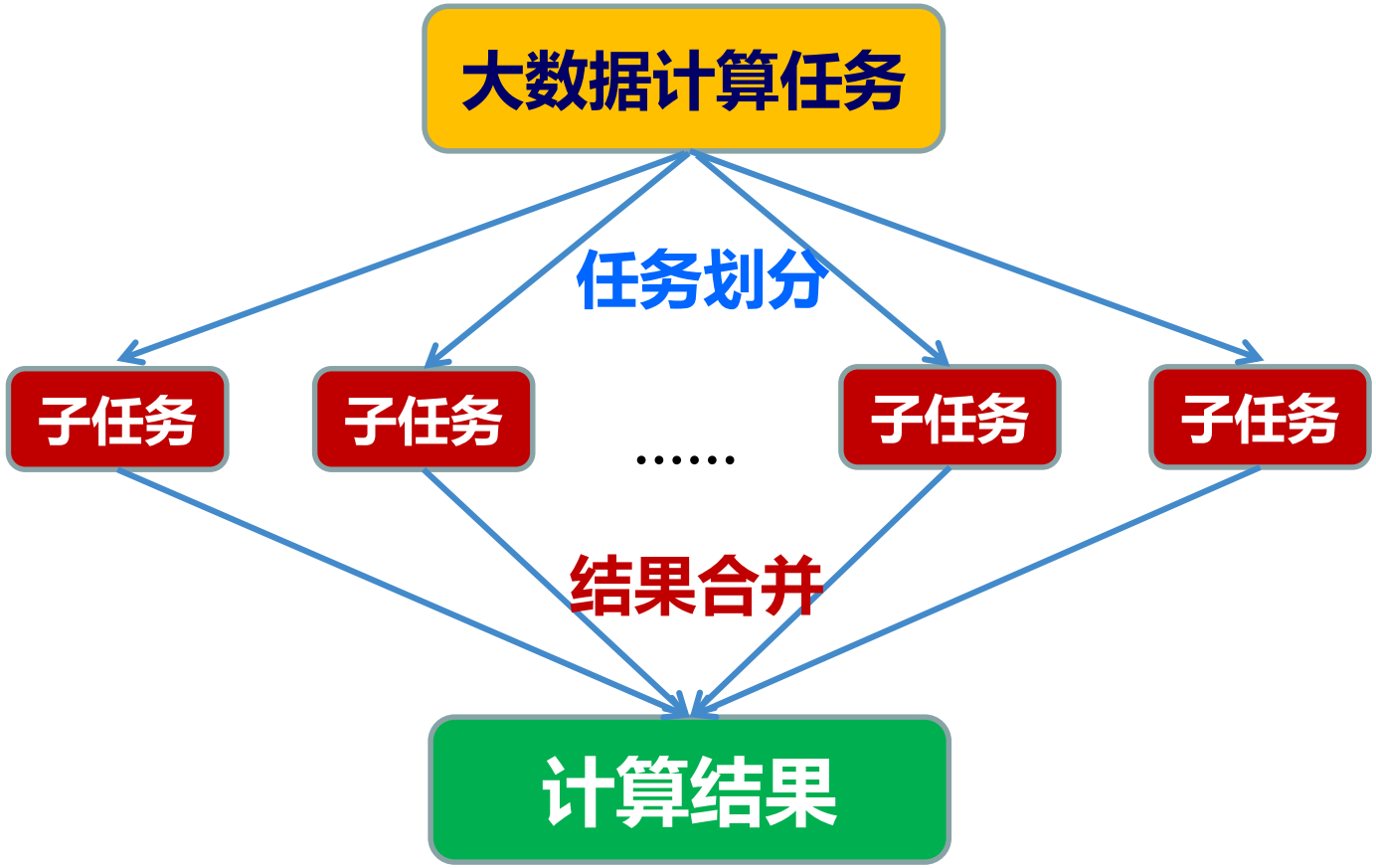
云计算

MapReduce分布式处理技术

- MapReduce是Google开发的Java、Python、C++编程工具，用于大规模数据集（大于1TB）的并行运算，也是云计算的核心技术。
- MapReduce模式的思想是将要执行的问题拆解成Map（映射）和Reduce（化简）的方式，先通过Map程序将数据切割成不相关的区块，分配（调度）给大量计算机处理达到分布运算的效果，再通过Reduce程序将结果汇整，输出开发者需要的结果。

云计算

MapReduce的编程思想——分而治之



云计算

MapReduce可解决哪些算法问题

- MapReduce待处理的数据集可以分解成许多小的数据集,而且每一个小数据都可以完全并行地进行处理，因此**不能解决不可分拆的计算任务，或者相互间有依赖关系任务**，如Fibonacci函数: $F_{k+2} = F_k + F_{k+1}$。
- MapReduce可解决的基本算法：**各种全局数据相关性小、能适当划分数据的计算任务。**如：分布式排序、关系代数操作（选择，投影，求交集、并集，连接，成组，聚合等）、矩阵向量相乘、矩阵相乘、词频统计(word count)、文档倒排索引等。

云计算

MapReduce可解决哪些算法问题

- MapReduce可解决的复杂算法：Web搜索引擎（网页爬取、网页排序、搜索算法）、Web访问日志分析（分析和挖掘用户在Web上的行为）、数据/文本统计分析（如专利文献引用分析和统计）、图算法并行化宽度优先搜索（最短路径问题）、机器学习、数据挖掘等。

云计算

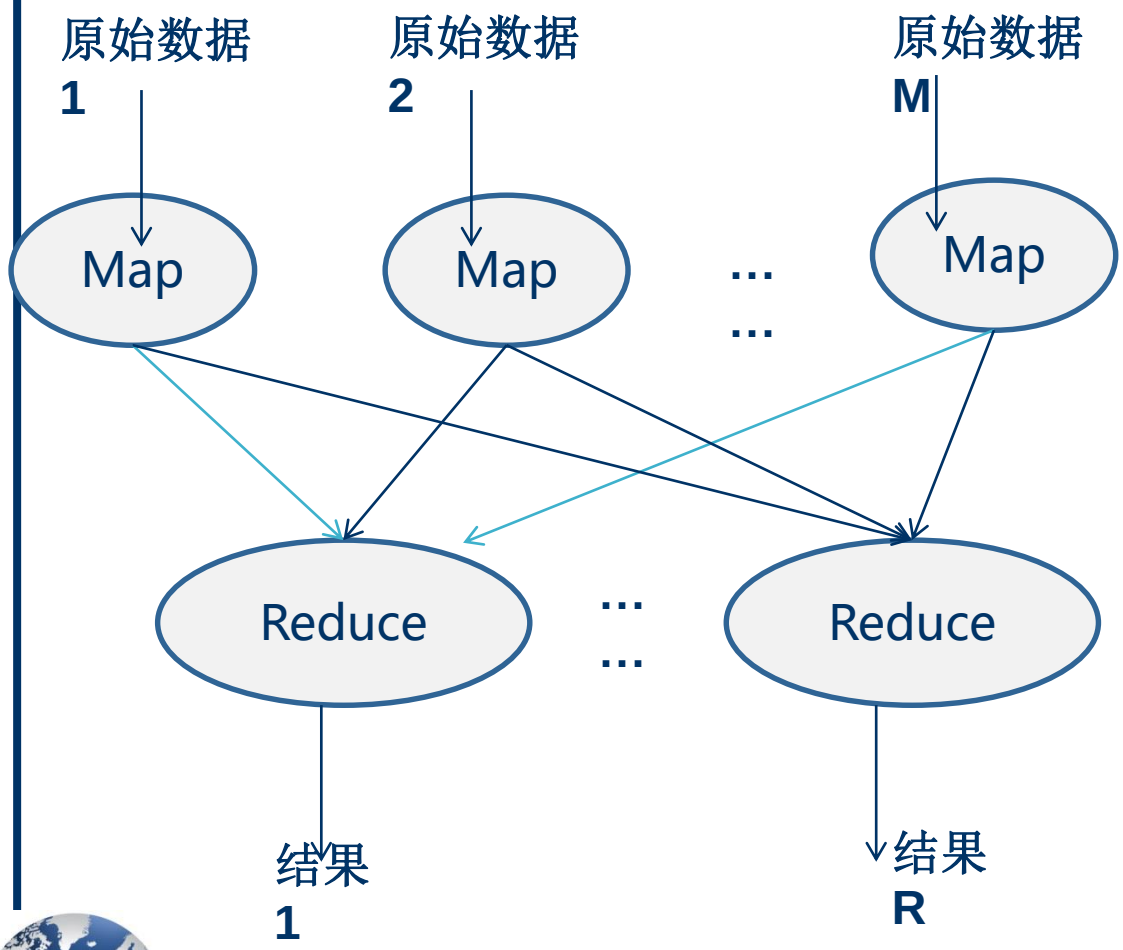
MapReduce分布式处理技术

MapReduce定义了Map和Reduce两个抽象的编程接口，由用户编程实现：

- Map: 对一组数据元素进行某种重复式的处理
- Reduce: 对Map的中间结果进行某种进一步的结果整理

云计算

MapReduce分布式处理技术



1、在编程的时候，开发者需要编写两个函数：

Map:(in_key, in_value)

Reduce:(key,[value 1,value 2…])

2、Map操作产生结果是<key,value>对

3、在Map，Reduce之间系统把同一Key归类到Reduce

3、Reduce操作对相同的Key进行归类处理

云计算

MapReduce分布式处理技术——Map端

map: (k1, v1) → list(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，并把处理的一组键值对中间结果list(k2, v2）以另一种键值对形式输出。
- 输出：键值对(k2, v2)表示的一组中间数据
- 备注: list(k2, v2) 表示有一个或多个键值对组成的列表

云计算

MapReduce分布式处理技术——Reduce端

reduce: (k2, list(v2)) → list(k3, v3)

- 输入：由map输出的一组键值对list(k2, v2) 将被进行合并处理，同样主键下的不同数值合并会到一个list(v2)中，故reduce的输入为(k2, list(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list(k3, v3)。
- 输出：最终输出结果list(k3, v3)。

云计算

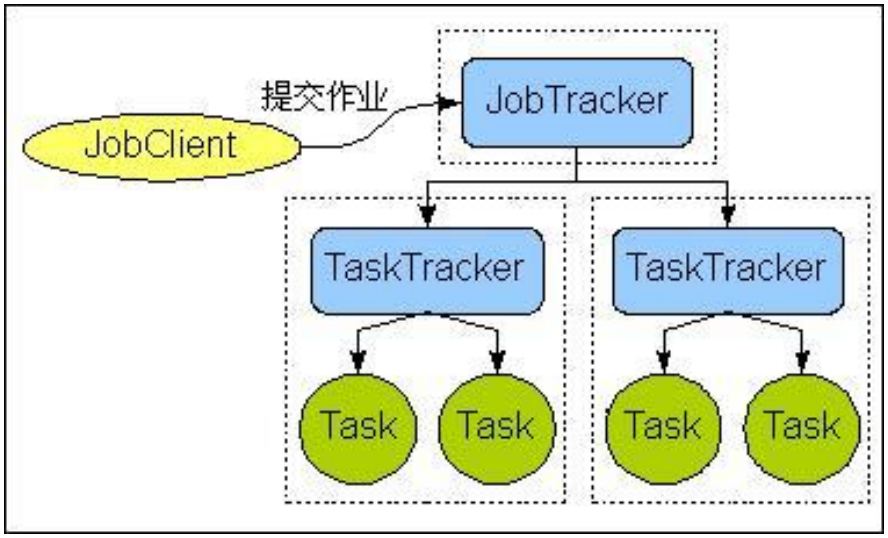
MapReduce分布式处理技术

函数	输入	输出
Map	(k1, v1)	List(k2, v2)
Reduce	(k2, List(v2))	List(k3, v3)

- 各个map函数对所划分的数据并行处理，从不同的输入数据产生不同的中间结果输出；
- 各个reduce各自并行计算，各自负责处理不同的中间结果数据集合；
- 进行reduce处理之前，须等到所有的map函数做完，并且在进入reduce前会对map的中间结果数据进行整理(Shuffle)，保证将map的结果发送给对应的reduce；
- 最终汇总所有reduce的输出结果即可获得最终结果

云计算

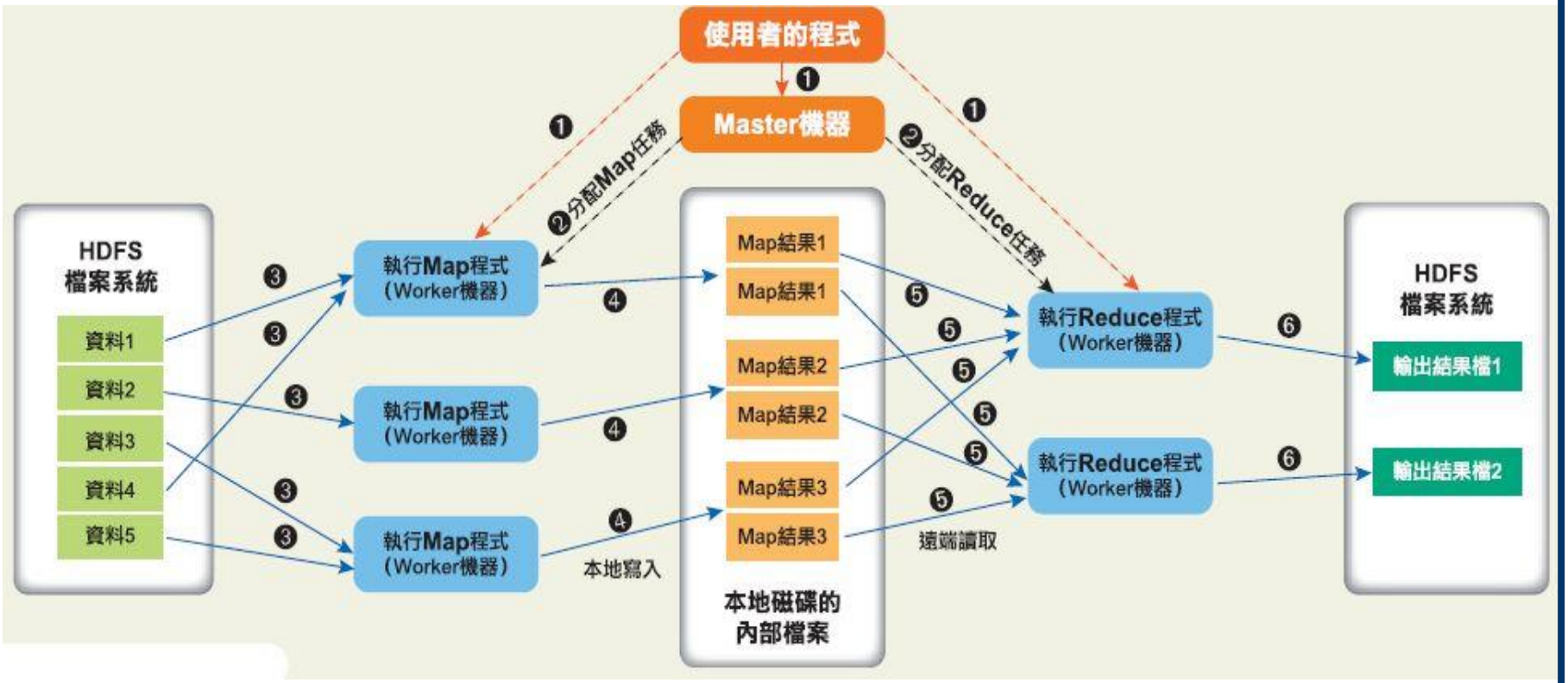
MapReduce分布式处理技术——框架



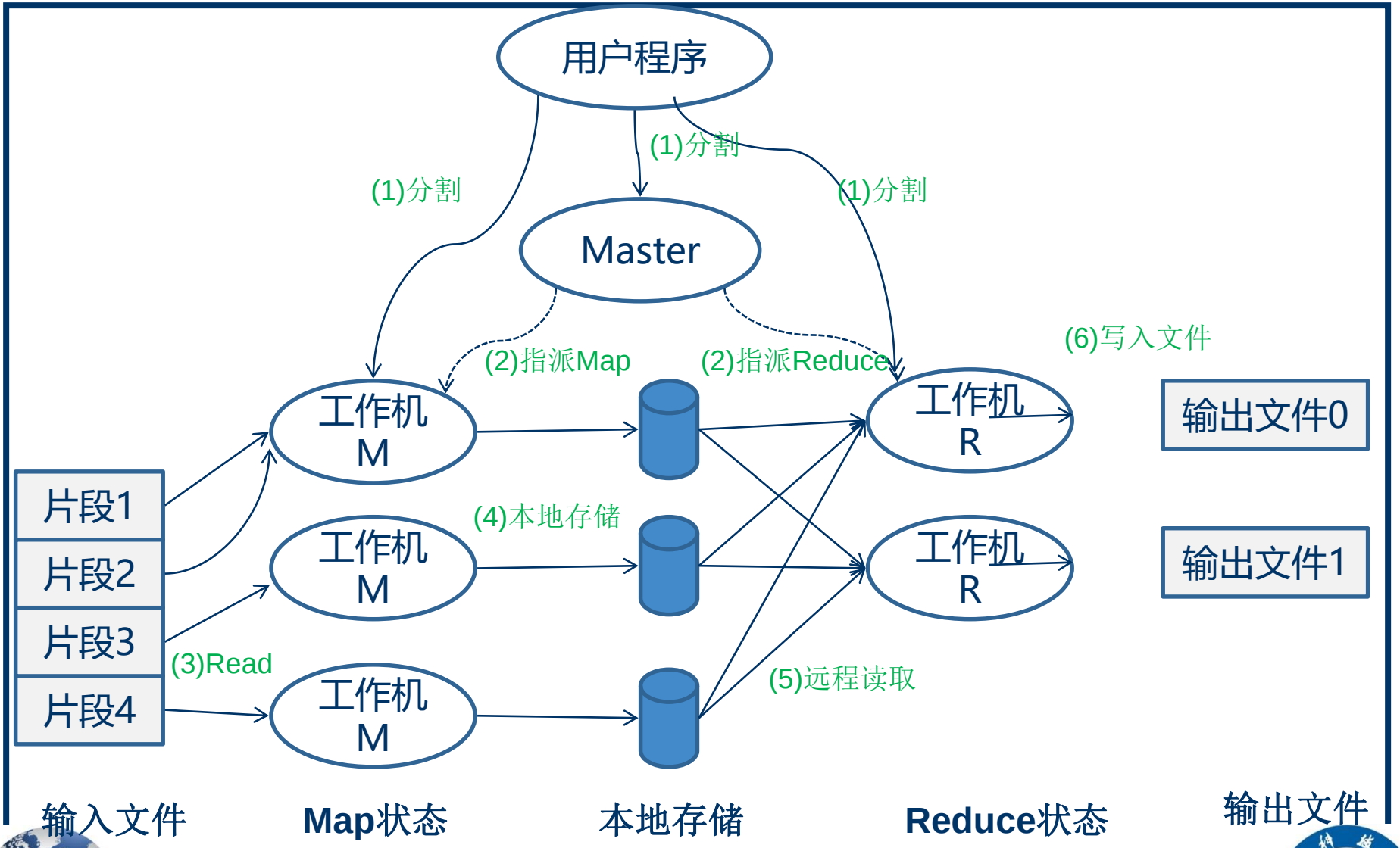
- MapReduce框架是由一个单独运行在主节点上的JobTracker 和运行在每个集群从节点上的TaskTracker共同组成的。
- 主节点负责调度构成一个作业的所有任务，这些任务分布在不同的从节点上。主节点监控它们的执行情况，并且重新执行之前失败的任务。

云计算

MapReduce分布式处理技术——实现机制



云计算


用户程序
(1)分割
(1)分割
(1)分割
Master
(2)指派Map
(2)指派Reduce
(6)写入文件
工作机
M
工作机
M
工作机
M
工作机
R
工作机
R
输出文件0
输出文件1
片段1
片段2
片段3
片段4
(3)Read
(4)本地存储
(5)远程读取
输入文件
Map状态
本地存储
Reduce状态
输出文件

云计算

MapReduce分布式处理技术——实现机制

- (1) MapReduce函数库首先把输入文件分成M块，每块大概16MB到64MB。接着在集群的机器上执行处理程序。
- (2) MapReduce算法运行过程中有一个主控程序，称为master。主控程序会产生很多作业程序，称为worker。并且把M个map任务和R个reduce任务分配给这些worker，让它们去完成。

云计算

MapReduce分布式处理技术——实现机制

- (3) 被分配了map任务的worker读取并处理相关的输入(这里的输入是指已经被切割的输入小块splite)。它处理输入的数据，并且将分析出的键/值(key/value)对传递给用户定义的reduce()函数。map()函数产生的中间结果键/值(key/value)对暂时缓冲到内存。
- (4) map()函数缓冲到内存的中间结果将被定时刷写到本地硬盘，这些数据通过分区函数分成R个区。这些中间结果在本地硬盘的位置信息将被发送回master，然后这个master负责把这些位置信息传送给reduce()函数的worker。

云计算

MapReduce分布式处理技术——实现机制

- (5) 当master通知了reduce()函数的worker关于中间键/值(key/value)对的位置时，worker调用远程方法从map()函数的worker机器的本地硬盘上读取缓冲的中间数据。当reduce()函数的worker读取到了所有的中间数据，它就使用这些中间数据的键(key)进行排序，这样可以使得相同键(key)的值都在一起。
- (6) reduce()函数的worker根据每一个中间结果的键(key)来遍历排序后的数据，并且把键(key)和相关的中间结果值(value)集合传递给reduce()函数。reduce()函数的worker最终把输出结果存放在master机器的一个输出文件中。

云计算

MapReduce分布式处理技术——实现机制

- (7) 当所有的map任务和reduce任务都已经完成后，master激活用户程序。在这时，MapReduce返回用户程序的调用点。
- (8) 当以上步骤成功结束以后，MapReduce的执行数据存放在总计R个输出文件中(每个输出文件都是由reduce任务产生的，这些文件名是用户指定的)。通常，用户不需要将这R个输出文件合并到一个文件，他们通常把这些文件作为输入传递给另一个MapReduce调用，或者用另一个分布式应用来处理这些文件，并且这些分布式应用把这些文件看成为输入文件由于分区(partition)成为的多个块文件。

MapReduce分布式处理技术——实例

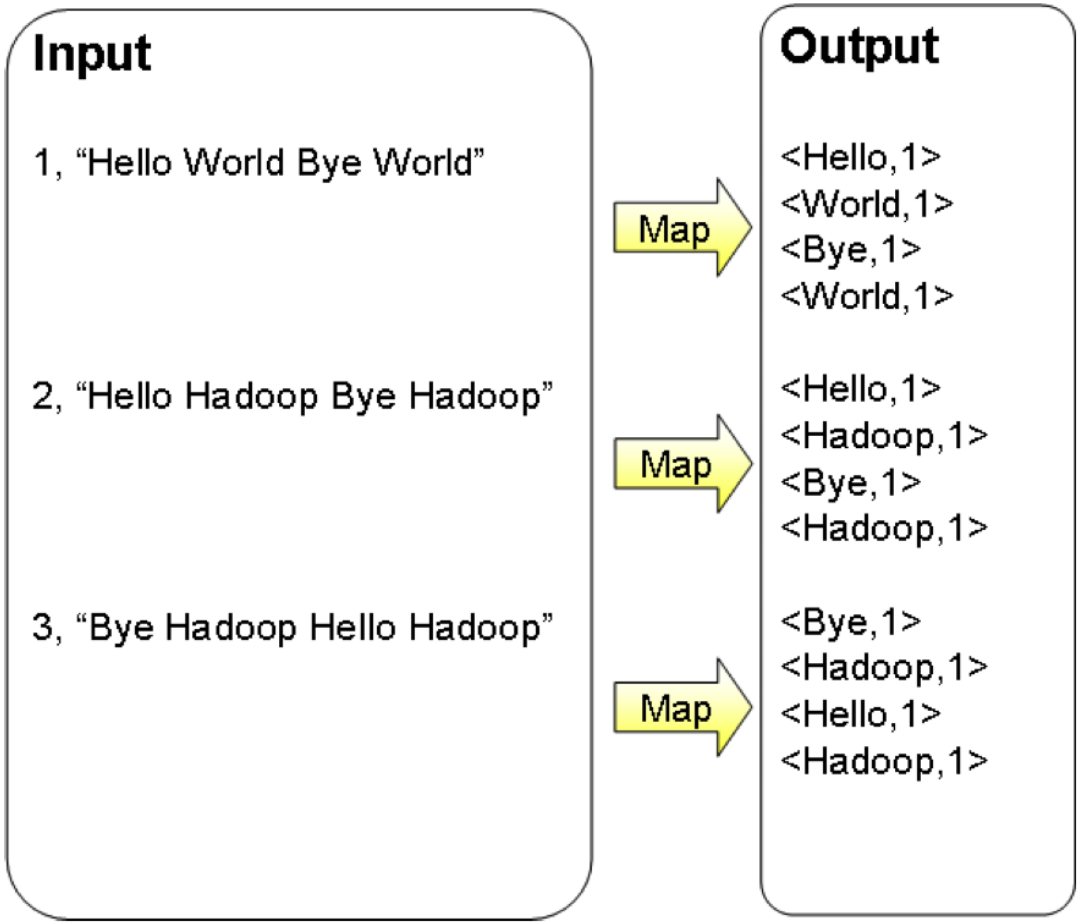
单词统计WordCount：

- 问题描述：统计文本中所出现单词的次数。
- 解决思路：需符合Map、Reduce各自的输入、输出格。
- Map端：输入为(k1, v1), 以文本行号为k1，以行内容为v1；输出为list(k2, v2)，每有一个单词就输出一个(word, 1)。
- Reduce端：输入为(k2, list(v2))，将list(v2)中所有数字相加即可得到单词次数；输出为list(k3, v3)，即最终的结果：(单词, 单词次数)。

MapReduce分布式处理技术——实例

单词统计WordCount：

- Map：



MapReduce分布式处理技术——实例

单词统计WordCount：

- Map函数

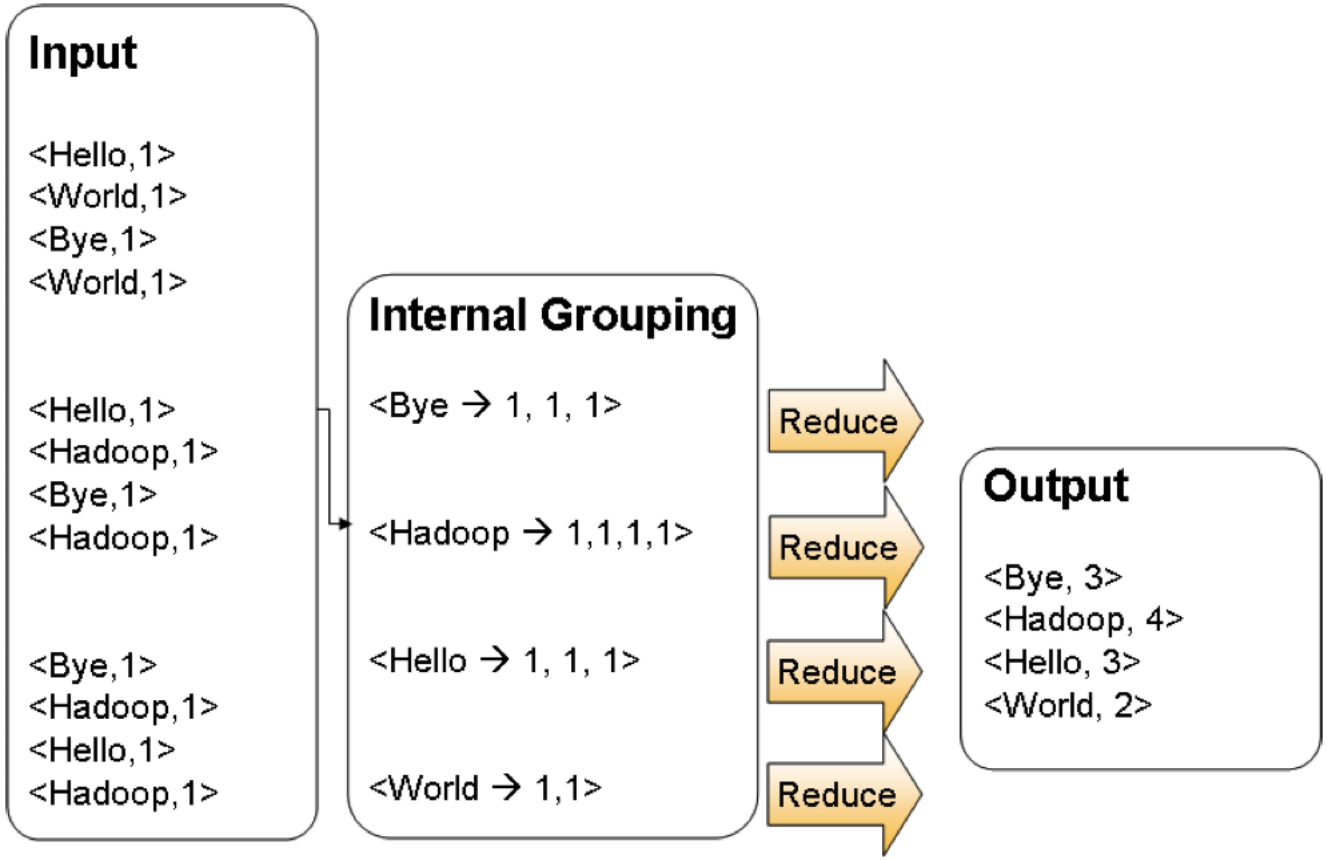
```
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
}
```

云计算

MapReduce分布式处理技术——实例

单词统计WordCount：

➢ **Reduce：**



MapReduce分布式处理技术——实例

单词统计WordCount：

- Reduce函数

```
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

云计算

MapReduce分布式处理技术——实例

气象站数据：

- 问题描述：给出气象站历年每天的数据，要提取出每年的最高气温。数据示例：

  0067011990999991950051507004…..99999N9+00221+99999…

- 解决思路-Map端： 输入为(k1, v1): (行号，内容)；输出为list(k2, v2): (年份，温度)，例如(1950, 22)。
- 解决思路-Reduce端： 输入为(k2, list(v2): 如(1950, [22, -11])，对list(v2)排序可得到高气温；输出为list(k3, v3)，即最终的结果：(年份, 最高气温)。

MapReduce分布式处理技术——实例

文档倒排索引算法：

- 问题描述：Inverted Index(倒排索引)是目前几乎所有支持全文检索的搜索引擎都要依赖的一个数据结构。基于索引结构，给出一个词(term)，能取得含有这个term的文档列表(the list of documents)

云计算

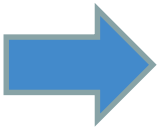
MapReduce分布式处理技术——实例

文档倒排索引算法：

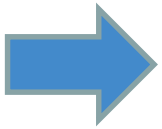
doc1：
one fish
two fish

doc2：
red fish
blue fish

doc3：
one red bird



倒排索引：
one: doc1, doc3
fish: doc1, doc2
two: doc1
red: doc2, doc3
blue: doc2
bird: doc3



搜索：
fish → doc1, doc2
red → doc2, doc3
red fish → doc2

Map：输入 (文档名+行号, 内容); 输出list(单词, 文档名)
Reduce: 输入(单词, list(文档名)); 输出list(单词, 文档列表)

云计算

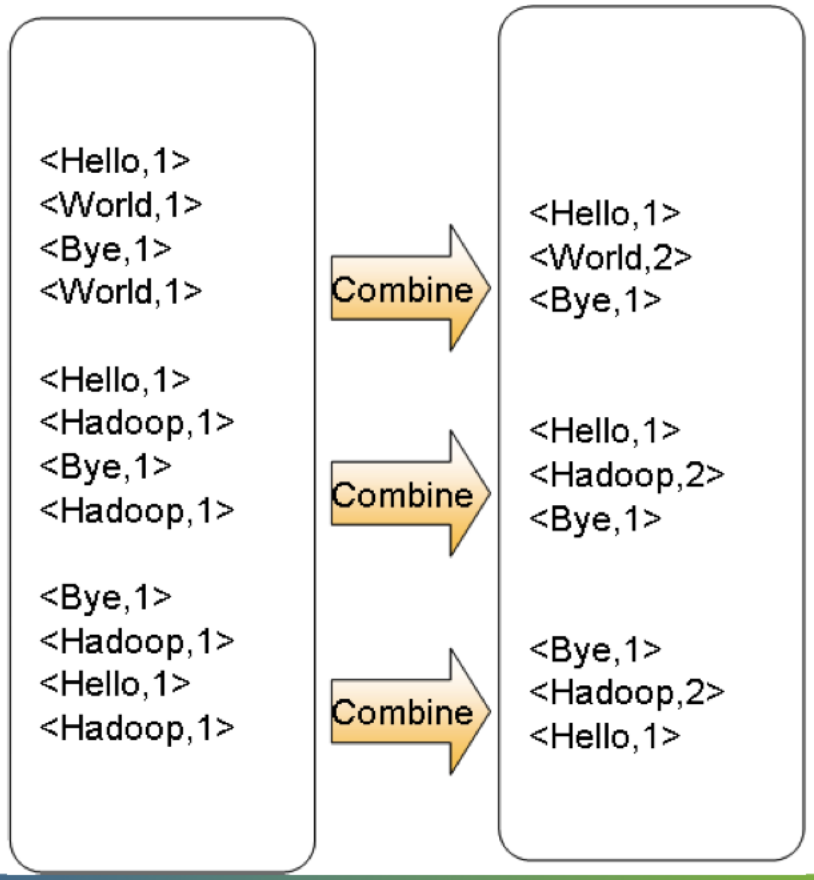
MapReduce分布式处理技术——改进

如单词统计问题，若有1亿个单词，那么就会传输1亿个键值对。合理的使用Combiner可以减少键值对的网络传输。减少数据网络传输也就意味着提升效率。

Combiner发生在Map输出时，通常与Reduce有相同的实现。

Combiner一般适用于求和，求最大/最小的实例中。

map: (K1, V1) → list(K2,V2)
combine: (K2, list(V2)) → list(K3, V3)
reduce: (K3, list(V3)) → list(K4, V4)



云计算

MapReduce分布式处理技术——容错机制

- 由于MapReduce函数库是设计用于在成百上千台机器上处理海量数据的，所以这个函数库必须考虑到机器故障的容错处理。
- master会**定期发送命令轮询**每一台worker机器。如果在一定时间内有一台worker机器一直没有响应，master就认为这个worker失效了。所有这台worker完成的map任务都被设置成为它们的初始空闲状态，并且因此可以被其他worker调度执行。类似的，所有这个机器上正在处理的map任务或者reduce任务都被设置成为空闲状态，被其他worker重新执行。

云计算

MapReduce分布式处理技术——容错机制

- 在失效机器上的已经完成的map任务还需要再次重新执行，这是因为中间结果存放在这个失效的机器上，所以导致中间结果无法访问。已经完成的reduce任务无需再次执行，因为它们的结果已经保存在全局的文件系统中了。
- 当map任务首先由A worker执行，随后被B worker执行的时候(因为A机器失效了)，所有执行reduce任务的worker都会被通知。所有还没有来得及从A上读取数据的worker都会从B上读取数据。

云计算

MapReduce分布式处理技术——容错机制

- 大多数MapReduce函数库都能够有效地支持很多worker失效的情况。比如，在一个网络例行维护时，可能会导致每次大约有80台机器在几分钟之内不能访问。master会简单地使这些不能访问的worker上的工作再执行一次，并且继续调度进程，直到所有任务都完成。

云计算

分布式的文件系统GFS

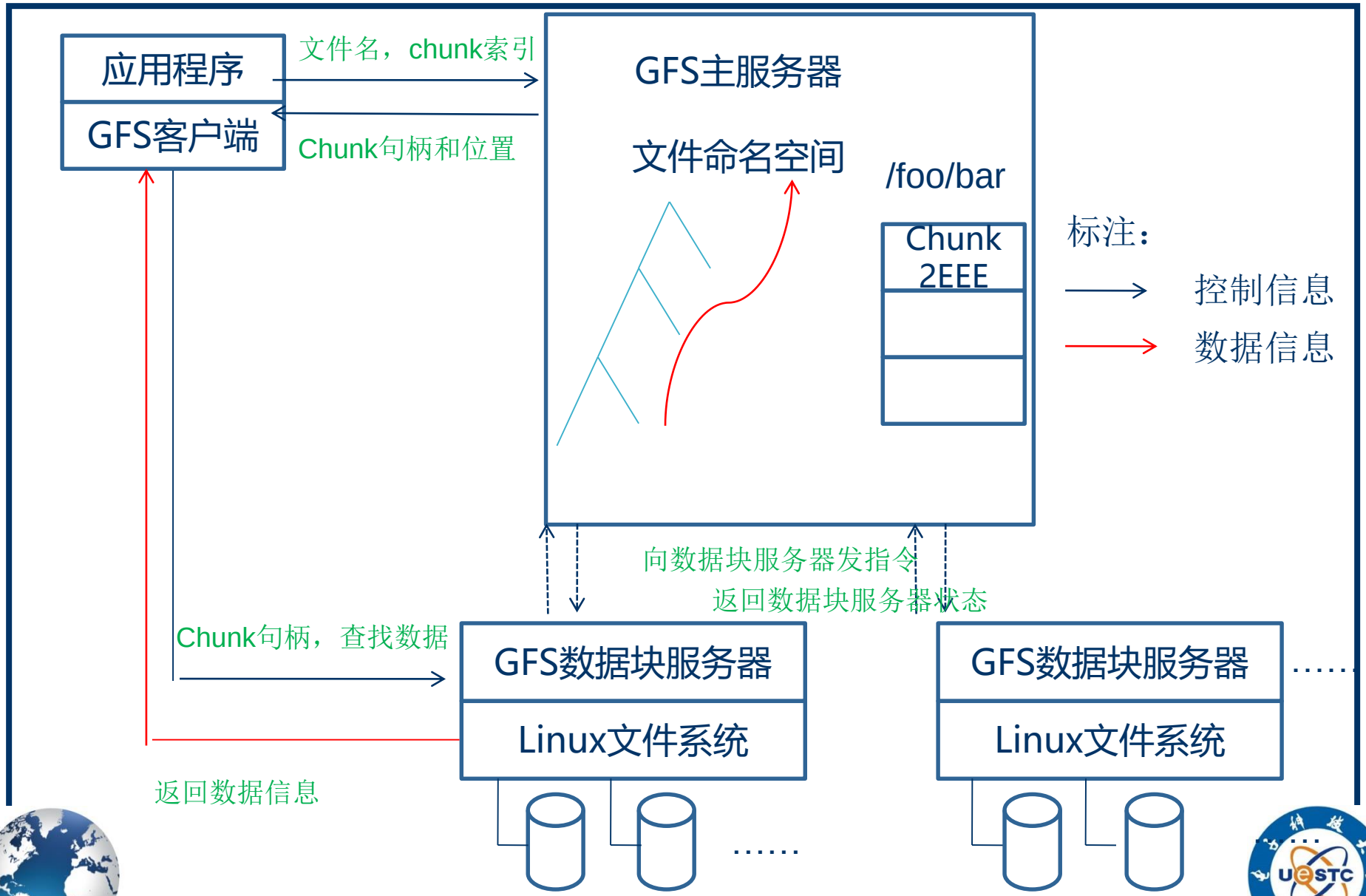
- GFS是一个适用于**大规模分布式数据处理**相关应用的，可扩展的分布式文件系统。
- GFS基于普通的不算昂贵的硬件设备，实现了容错的设计，并且为大量客户端提供极高的聚合处理性能。
- **GFS集群由一个单个的master和多个chunkserver（块服务器）组成，还有相应的很多客户端client。**

云计算

分布式的文件系统GFS——设计动机

- Google需要一个支持海量存储的文件系统。
- 购置昂贵的分布式文件系统与硬件?
- **是否可以在一堆廉价且不可靠的硬件上构建可靠的分布式文件系统?**
- **GFS**将容错任务交给文件系统完成，利用软件的方法解决系统可靠性问题，使存储的成本成倍下降。 **GFS**将服务器故障视为正常现象，并采用多种方法，从多个角度，使用不同的容错措施，确保数据存储的安全、保证提供不间断的数据存储服务。

云计算—GFS 体系结构


应用程序
GFS客户端
文件名，chunk索引
Chunk句柄和位置
GFS主服务器
文件命名空间
/foo/bar
Chunk 2EEE
标注：
控制信息
数据信息
向数据块服务器发指令
返回数据块服务器状态
Chunk句柄，查找数据
GFS数据块服务器
Linux文件系统
GFS数据块服务器
Linux文件系统
返回数据信息

云计算

分布式的文件系统GFS——GFS体系结构

- GFS将整个系统划分为三类角色：Client，Master，Chunk Server。
- Client提供**应用程序的访问接口**（是一组专用接口），以库文件的形式提供。
- Master是**GFS的管理节点**。在逻辑层面只有一个，保存整个系统的元数据。负责整个系统的文件管理。
- Chunk server 负责**存储工作**。数据主要以文件的形式存储在Chunk server上。
- GFS将文件划分为大小为64M数据块chunk，每一个数据块（chunk）都有唯一的index索引号。

云计算

分布式的文件系统GFS——GFS体系结构

GFS实现机制：

- 客户端首先访问Master节点，获取交互的Chunk Server信息，然后访问这些Chunk Server，完成数据存取工作。这种设计方法实现了**控制流和数据流的分离。**
- Client与Master之间只有控制流，而无数据流，极大地**降低了Master的负载。**
- Client与Chunk Server之间直接传输数据流，同时由于文件被分成多个Chunk进行分布式存储，Client可以同时访问多个Chunk Server，从而使得整个系统的I/O高度并行，系统**整体性能得到提高。**

云计算

分布式的文件系统GFS——GFS体系结构

GFS体系结构-Chunk Server：

- 在GFS下，每一个文件都拆成固定大小的chunk(块)。每一个块都由master根据块创建的时间产生一个全局唯一的以后不会改变的64位的chunk handle标志。
- Chunk servers在本地磁盘上用Linux文件系统保存这些块，并且根据chunk handle和字节区间，通过Linux文件系统读写这些块的数据。
- 出于可靠性的考虑，每一个块都会在不同的chunk server上保存备份。缺省情况下，保存3个备份。

分布式的文件系统GFS——GFS体系结构

GFS体系结构-Master：

- Master负责管理所有的文件系统的元数据，包括文件和chunk的namespace、访问控制信息、文件到chunk的映射关系、当前chunk的位置等等。
- Master控制系统级别的活动，包括chunk的分配管理、孤点chunk的垃圾回收机制、chunk在chunk server之间的移动。
- Master周期地跟每个chunk server通信，给他们以指示并收集他们的状态。
- 单一Master设计必须减小master的读/写操作，以避免它成为集群瓶颈。

云计算

分布式的文件系统GFS——GFS体系结构

GFS体系结构-Client：

- Client代码包含了google文件系统的API，并且会和master和chunk server进行通信。
- client和master通信——**交互元数据信息**：client会缓存从master获取的元数据信息，以便对同一块的操作不在通过client-master交互。(除非cache过期或这个文件被再次打开)
- client和chunk server通信——**交互文件数据**：client所有的数据相关的通信是直接和chunk server进行，但不会缓存文件数据。

分布式的文件系统GFS——GFS体系结构

GFS体系结构-Client读取数据的操作顺序：

- Client把应用要读取的文件名和偏移量，根据固定的chunk大小，转换成为文件的chunk index。
- 向master发送这个包含了文件名和chunk index的请求。
- Master返回相关的chunk handle以及对应的位置，client缓存这些信息。
- Client就向最近的对应位置的chunk server发起请求，请求包含chunk handle以及一个在这个chunk内需要读取的字节区间。
- Chunk server返回给client要读取的chunk data 。

云计算

分布式的文件系统GFS——GFS体系结构

GFS体系结构-Chunk size：

- Chunk size是设计的关键参数(64M)。
- 采用很大的chunk size优点：它减少了客户端和master的交互；减少网络管理量；减少了元数据在master上的大小。
- 采用很大的chunk size缺点：小型文件包含较少数目的chunk，也许只有一个chunk。保存这些文件的chunk server就会在大量客户端访问的时候就会成为焦点，导致系统局部过载。

云计算—GFS特点

采用中心服务器模式

- ▶可以方便地增加Chunk Server
- ▶ Master掌握系统内所有Chunk Server的情况，方便进行负载均衡
- ▶不存在元数据的一致性问题

不缓存数据

- ●文件操作大部分是流式读写，不存在大量重复读写，使用Cache对性能提高不大
- ● Chunk Server上数据存取使用本地文件系统，若读取频繁，系统具有Cache
- ●从可行性看，Cache与实际数据的一致性维护也极其复杂

在用户态下实现

- □利用POSIX编程接口存取数据降低了实现难度，提高通用性
- □POSIX接口提供功能更丰富
- □用户态下有多种调试工具
- □Master和Chunk Server都以进程方式运行，单个进程不影响整个操作系统
- □GFS和操作系统运行在不同的空间，两者耦合性降低

只提供专用接口

- ○降低实现的难度
- ○对应用提供一些特殊支持
- ○降低复杂度

云计算

分布式的文件系统GFS——GFS容错机制

Master容错：

- 单个Master，对于前两种元数据，GFS通过**操作日志**来提供容错功能。

云计算

分布式的文件系统GFS——GFS容错机制

Master容错：

- 第三种元数据信息保存在各个Chunk Server上，Master故障时，**磁盘恢复**。
- GFS还提供了Master远程的**实时备份**，防止Master彻底死机。

分布式的文件系统GFS——GFS容错机制

Chunk Server容错：采用副本方式

- 每一个Chunk有多个存储副本（默认为三个副本），分布存储在不同的Chunk Server上。
- 副本的分布策略需要考虑多种因素，如网络的拓扑、机架的分布、磁盘的利用率等。
- 对于每一个Chunk，必须将所有的副本全部写入成功，才视为成功写入。

云计算

分布式的文件系统GFS——GFS容错机制

Chunk Server容错：采用副本方式

尽管一份数据需要存储三份，好像磁盘空间的利用率不高，但综合比较多种因素，加之磁盘的成本不断下降，采用副本无疑是最简单、最可靠、最有效，而且实现的难度也最小的一种方法。

云计算

分布式的文件系统GFS——GFS容错机制

Chunk Server容错：采用副本方式

- GFS中的每一个文件被划分成多个Chunk，Chunk的默认大小是64MB。
- Chunk Server存储的是Chunk的副本，副本以文件的形式进行存储。
- 每个Chunk又划分为若干Block（64KB），每个Block对应一个32bit的校验码，保证数据正确（若某个Block错误，则转移至其他Chunk副本）。

云计算

分布式的文件系统GFS——GFS系统管理技术

云计算

结构化的BigTable存储系统

为什需要设计BigTable?

✦**Google需要存储的数据种类繁多**: Google目前向公众开放的服务很多，需要处理的数据类型也非常多。包括URL、网页内容、用户的个性化设置在内的数据都是Google需要经常处理的。

✦**海量的服务请求**： Google运行着目前世界上最繁忙的系统，它每时每刻处理的客户服务请求数量是普通的系统根本无法承受的。

✦**商用数据库无法满足Google的需求**：一方面现有商用数据库设计着眼点在于通用性，根本无法满足Google的苛刻服务要求；另一方面对于底层系统的完全掌控会给后期的系统维护、升级带来极大的便利。

云计算

☙结构化的BigTable存储系统——目标

广泛的适用性

Bigtable是为了满足系列Google产品而非特定产品存储要求

很强的可扩展性

根据需要随时可以加入或撤销服务器

高可用性

Bigtable设计的重要目标之一就是确保几乎所有的情况下系统都可用

简单性

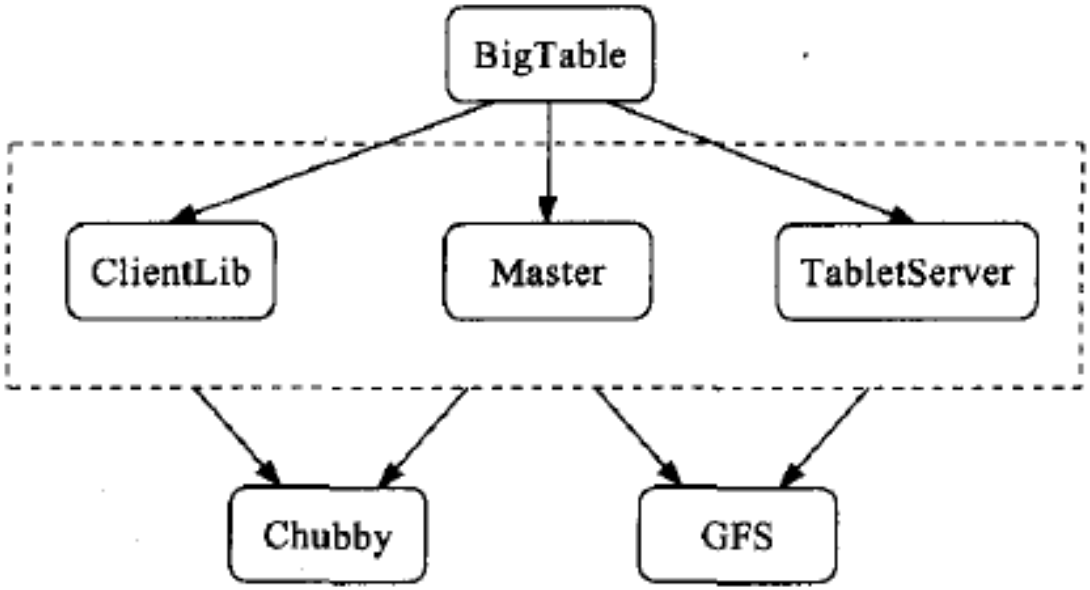
底层系统简单性既可减少系统出错概率，也为上层应用开发带来便利

云计算

结构化的BigTable存储系统

✦GFS 用于底层数据存储的分布式文件系统，Bigtable 系统运行在GFS上实现对网络数据的**结构化管理**。

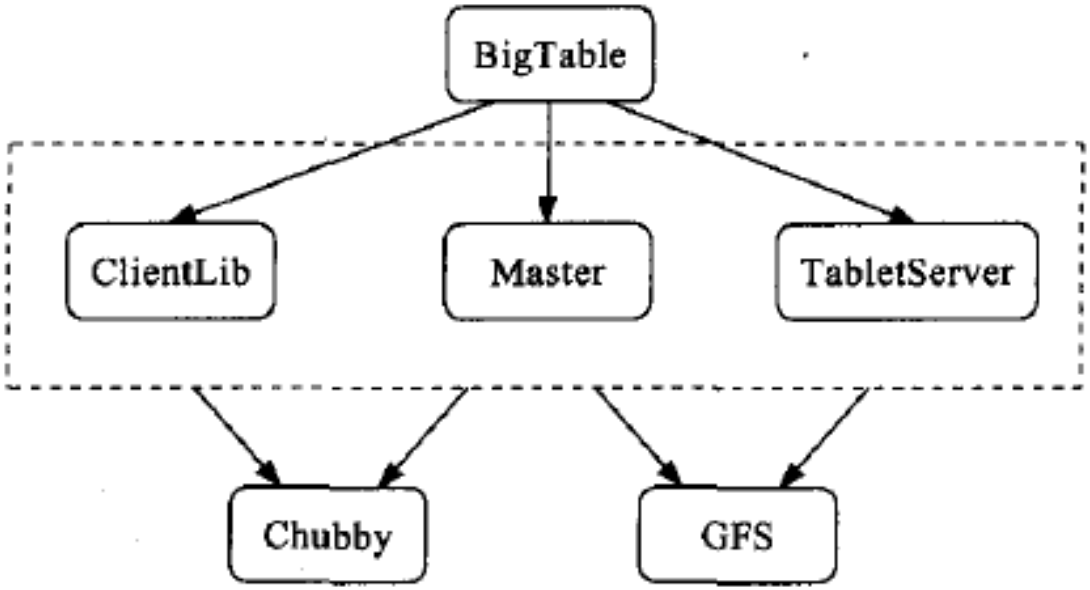
✦Bigtable系统的运行环境是普通微机组成**集群**环境，实现了数据的**分布式**管理和结构的组织。



云计算

☙结构化的BigTable存储系统

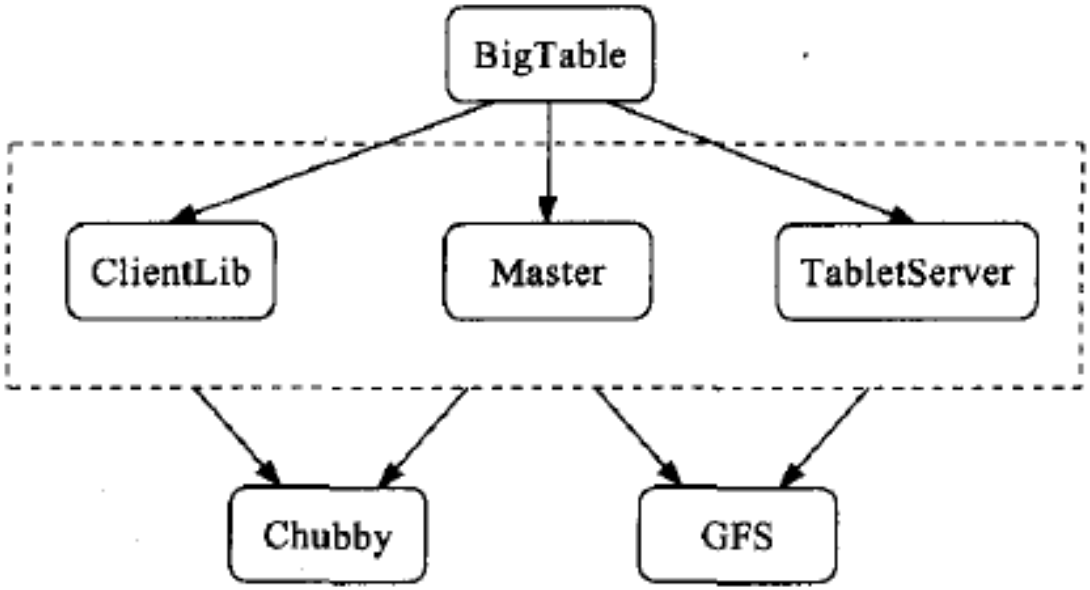
✦Bigtable主要由三个部分组成：客户端程序库（Client Library）、一个主服务器（Master Server）和多个子表服务器（Tablet Server）。



云计算

结构化的BigTable存储系统

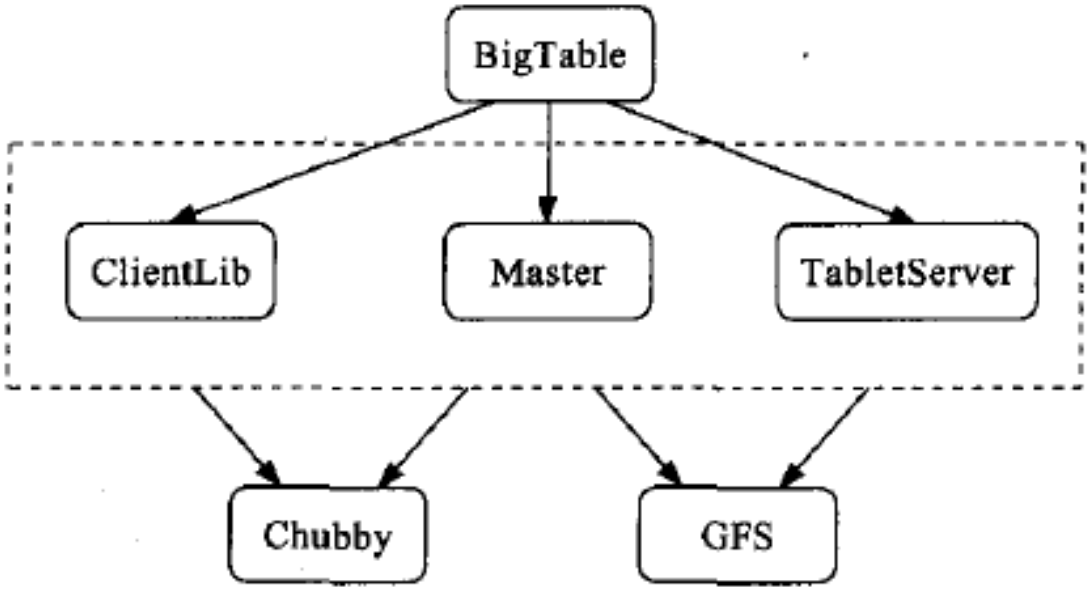
✦海量的数据分布存储在底层文件系统GFS中，逻辑相邻的数据由子表服务器分块管理，Master服务器实现对多台子表服务器的协调和调度。



云计算

结构化的BigTable存储系统

✦为了保证多台服务器共享数据的安全性，此系统采用Chubby提供锁服务。Chubby是一个分布式的锁服务器，利用小型锁文件的管理实现**分布式数据的共享（涉及数据一致性理论）**，同时具有表结构数据的存储能力。



云计算

结构化的BigTable存储系统

✦客户端程序库（Client Library）访问Bigtable服务时，首先要利用其库函数执行Open()操作来打开一个锁（实际上就是获取了文件目录），锁打开以后客户端就可以和子表服务器进行通信。

✦和许多具有单个主节点分布式系统一样，客户端程序库主要与子表服务器通信，几乎不和主服务器进行通信，这使得主服务器的负载大大降低。

✦主服务主要进行一些元数据操作以及子表服务器之间负载调度问题，实际数据是存储在子表服务器上 。

云计算

Google Hadoop架构

✦在Google发表MapReduce后，2004年开源社群用Java搭建出一套Hadoop框架，用于实现MapReduce 算法，能够把应用程序分割成许多很小的工作单元，每个单元可以在任何集群节点上执行或重复执行。

✦此外，Hadoop 还提供一个分布式文件系统GFS（Google file system），支持大型、分布式大数据量的读写操作，其容错性较强。而分布式数据库（BigTable）是一个有序、稀疏、多维度的映射表，有良好的伸缩性和高可用性，用来将数据存储或部署到各个计算节点上。

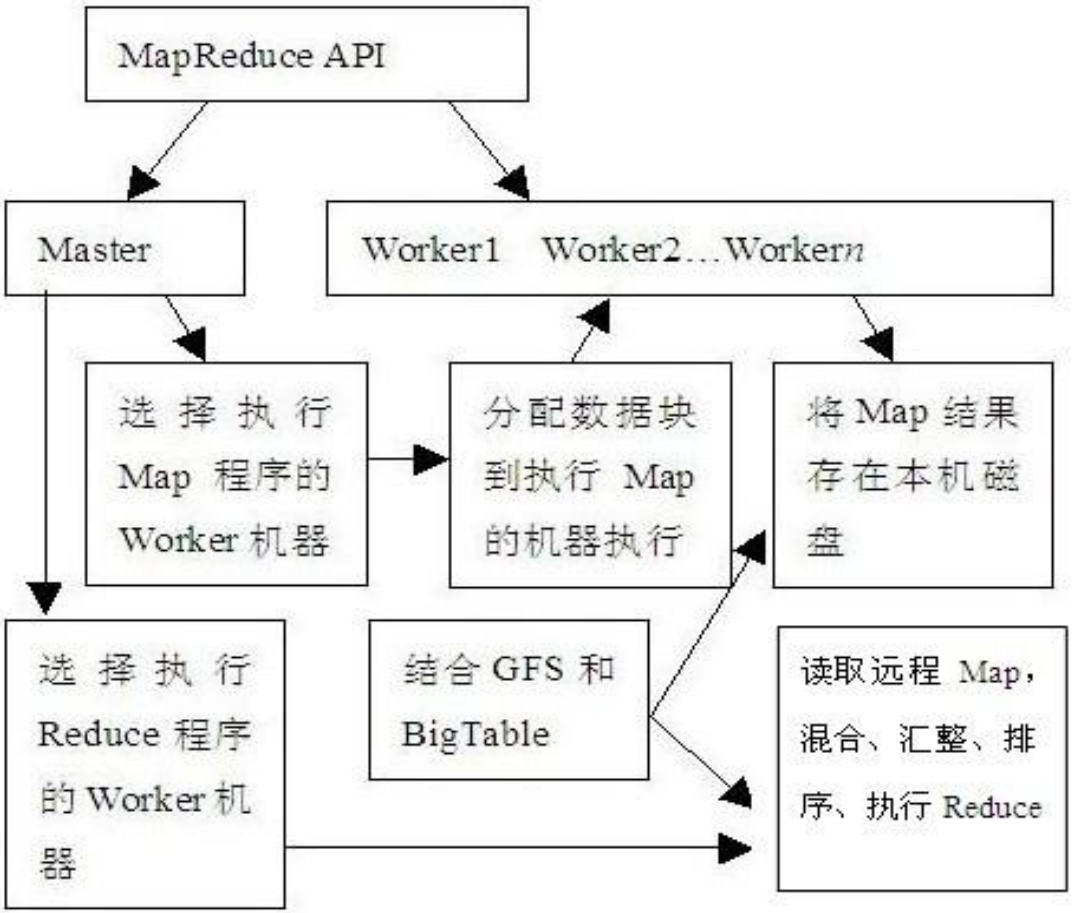
云计算

Google Hadoop架构

在架构中MapReduce API提供Map和Reduce处理、GFS分布式文件系统和BigTable分布式数据库提供数据存取。基于Hadoop可以非常轻松和方便完成处理海量数据的分布式并行程序，并运行于大规模集群上。

云计算

Google云计算执行过程



云计算

其他云计算产品

- Amazon使用弹性计算云（EC2）和简单存储服务（S3）为企业提供计算和存储服务
- IBM在2007年11月推出了 “蓝云”计算平台
- 中国电信e云
- 中国移动 “大云”

云计算

☙云计算发展障碍

✦标准不统一：Google、Amazon、 IBM、微软等的平台互不兼容

✦数据安全：云服务提供商的信誉——留后门？！ 面临着全世界的黑客——需要高强度的安全系统。

✦耗电量巨大：节能减排、绿色云计算