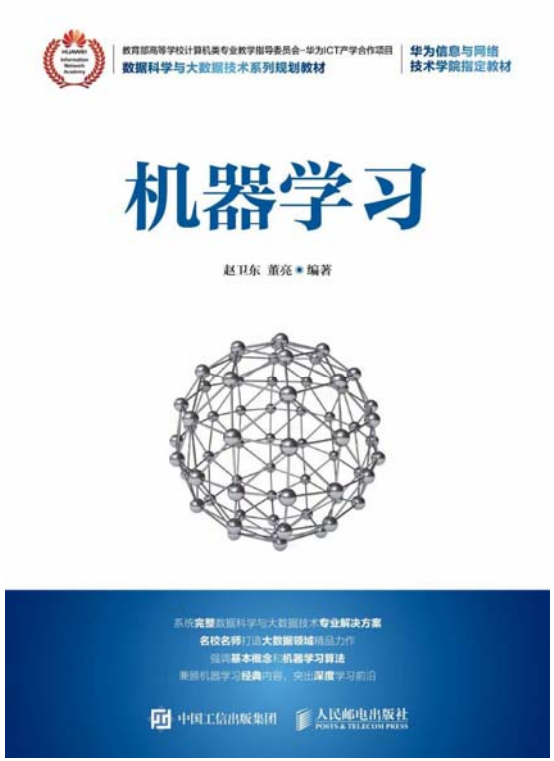


机器学习
分布式机器学习

复旦大学 **赵卫东** 博士

wdzhao@fudan.edu.cn

章节介绍

- 机器学习方法是计算机利用已有的数据生成某种模型，并利用此模型预测的一种方法。在确定模型结构之后，根据已知模型寻找模型参数的过程就是训练，训练过程中不断依据训练数据来迭代调整模型的参数值，从而使模型的预测结果更为准确。在现实应用中，要达到好的效果，训练数据集可能很大，模型参数量剧增，会带来很多性能和算法设计问题，单台机器难以胜任，需要分布式的机器学习架构。本章主要介绍分布式机器学习基础知识，并介绍主流的分布式机器学习框架，结合实例介绍一些机器学习算法。

章节结构

- 分布式机器学习基础
  - 参数服务器
  - 分布式并行计算类型
- 分布式机器学习框架
  - MapReduce 编程模型
  - Hadoop MapReduce 框架
  - Spark
  - PMLS
  - TensorFlow
- 并行决策树
- 并行k-均值算法

分布式机器学习基础

- 分布式机器学习中的一些核心问题：
  - 如何提高各分布式任务节点之间的网络传输效率；
  - 如何解决参数同步问题，传统训练模型是采用同步方法，如果机器性能不统一，必然会产生训练任务之间的协作；
  - 分布式环境下如何提高容错能力，需要避免单点故障，并能合理处理异常，训练子节点出错不影响全局任务。

参数服务器

- 应用传统的大数据处理框架训练大型的机器学习模型时，由于数据量比较大并且训练方法多样，存在着一致性、扩展性、稳定性的问题。较大的模型也意味着参数较多，因而需要实现分布式并行训练，参数服务器是分布式并行训练框架之一，存储着模型的参数和状态。参数服务器具有如下特点：
  - 高效通信
  - 宽松一致性
  - 灵活可扩展
  - 容错能力强
  - 易用

灵活可扩展　　容错能力强　　易用

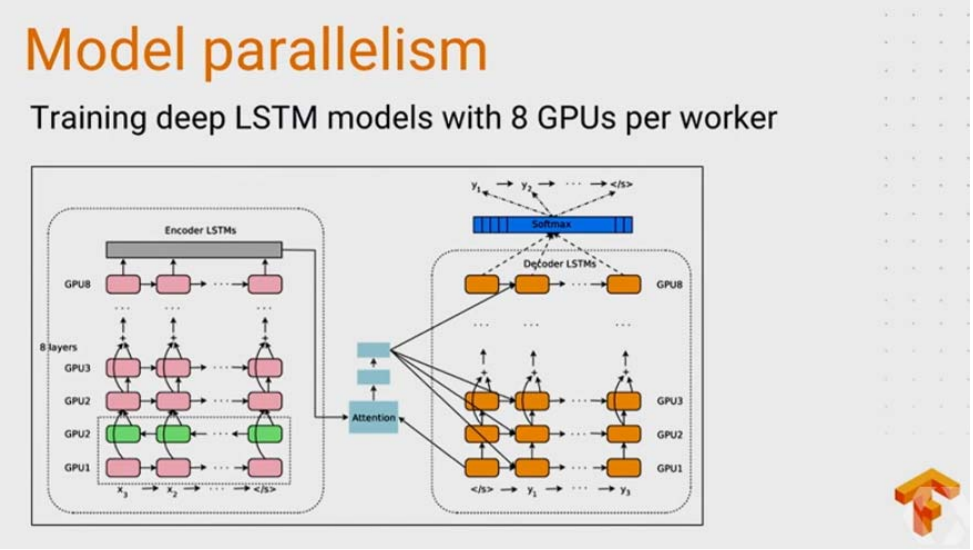
- 训练过程中支持动态扩展节点，不需要重启训练任务就可以动态插入新节点到集合中，这一特性无疑有利于那些训练周期较长（长达数天或数周）的机器学习项目，可节省大量训练时间。
- 在大型服务器集群中，由于节点较多，小概率故障往往常态化，所以需要节点的恢复（状态清理、任务重启）时间要短，而且不能中断训练过程，这就要求并行化系统具有较强的容错能力。
- 目前机器学习项目开发者数量较少，为了减少学习难度，需要尽可能的使用常用语言或将参数表示成通用的形式，如向量、矩阵等，并与现有机器学习框架无缝拼接。

分布式并行计算框架

- 分布式并行计算的类型一般分为三种：
  - 模型并行
  - 数据并行
  - 混合并行

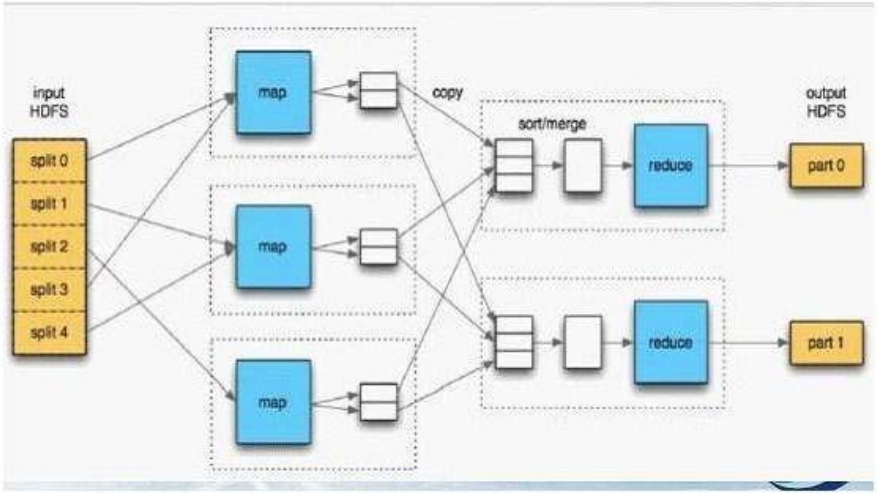
模型并行

- 模型并行是指将模型按照其结构放在不同的分布式机器上进行训练，一般用在那些内存要求较高的机器学习项目，例如，单机训练一个1000层的DNN网络，内存容易溢出，而使用模型并行，用不同的机器负责不同的层进行训练，通过维护各层间参数同步实现整个DNN网络的并行训练。



数据并行

- 数据并行是指各机器上的模型相同，对训练数据进行分割，并分配到各机器上，最后将计算结果按照某种方式合并。该方法主要应用在海量训练数据的情况，数据以并行化方式训练，训练过程中组合各工作节点的结果，实现模型参数的更新。参数并行常用的方法有参数平均和异步梯度下降的方法。



参数平均

- 参数平均是在每次训练迭代完成后计算各节点各模型参数平均值，这一方法操作简单，主要依赖网络同步更新，如果更新频率较慢会导致参数差别较大，平均之后的模型参数的局部差异化被抵消，效果较差，影响模型的精确性。反之，如果更新较快，对网络压力较大，通信和同步的成本较高，所以在应用中需要结合模型复杂度和优化方法进行平衡。

异步梯度下降

- 异步梯度下降是一种基于更新的数据并行化，它传递的是模型训练过程中的梯度、动量等信息，而没有直接传递参数值，这样一方面可以减少传输数据量，提高网络传输效率；另一方面不同计算节点通过共享梯度，可以提高模型收敛速度。该方法的不足之处在于会随着引入参数数量的增多出现梯度值过时的问题。

混合并行

- 混合并行的方式是指综合应用模型并行和数据并行，在训练集群的设计中，将上述两种方式进行合并，各取所长，形成互补。例如，可以在同一台机器上采用模型并行化，在GPU和CPU之间使用模型并行。然后在机器之间采用数据并行化，将数据分配在不同的机器上，既实现了计算资源利用的最大化，也减少了数据分发的压力。

分布式机器学习框架

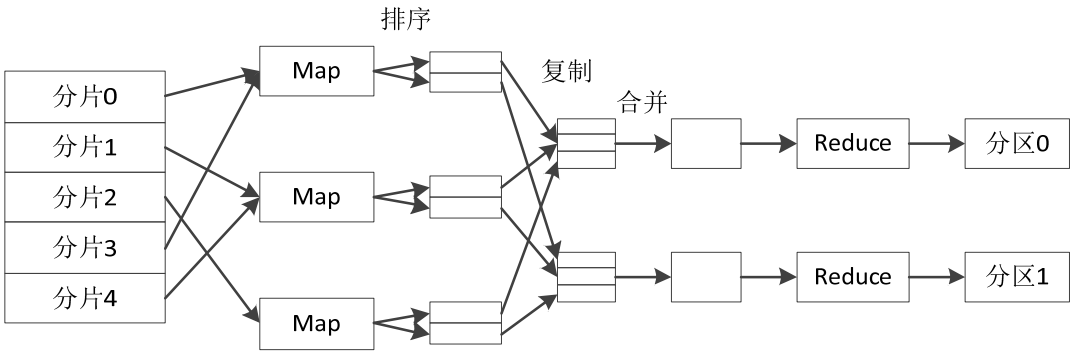
- 分布式机器学习是机器学习领域的一大主要研究方向，其中MapReduce适合做离线计算，Storm适合做流式计算，Spark是内存计算框架，能快速得到计算结果。分布式机器学习平台归类为三种基本设计方法：基本数据流、参数服务器模型以及高级数据流。基于这三种方法来介绍分布式机器学习框架。

MapReduce编程模型

- MapReduce是一个能处理和生成超大数据集的算法模型，该架构能够在大量硬件配置不高的计算机上实现并行化处理，这一编程模型结合用户自定义的Map和Reduce函数。Map函数处理一个输入的基于<Key,value>对的集合，输出中间基于<Key,value>对的集合，Reduce函数是将所有具有相同key值的value值进行合并，将数据集合进行压缩。

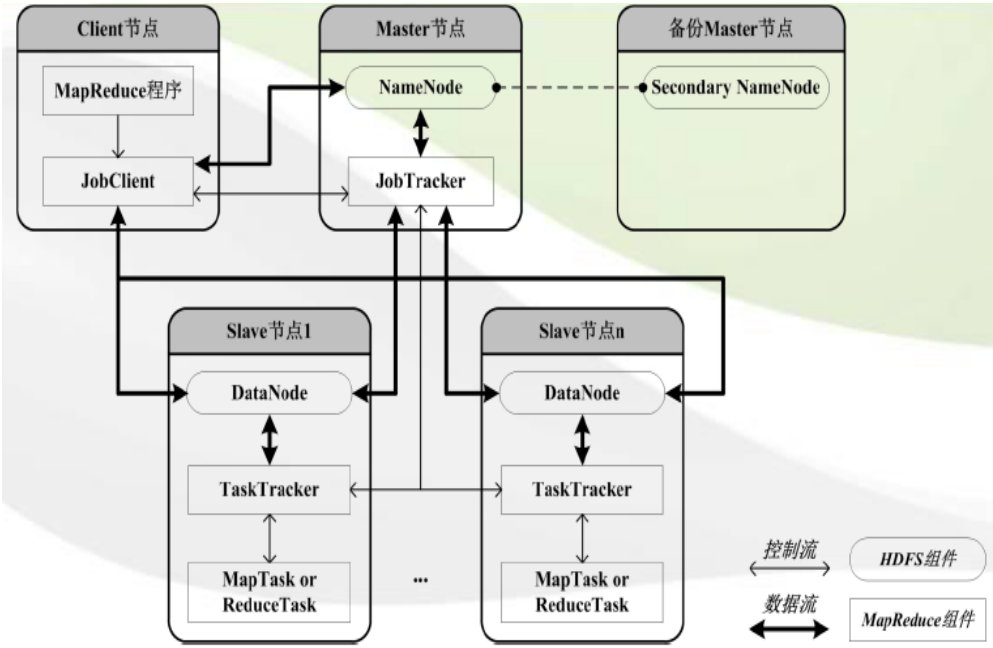
MapReduce编程模型

- 一个典型的MapReduce程序的执行流程如下图所示。



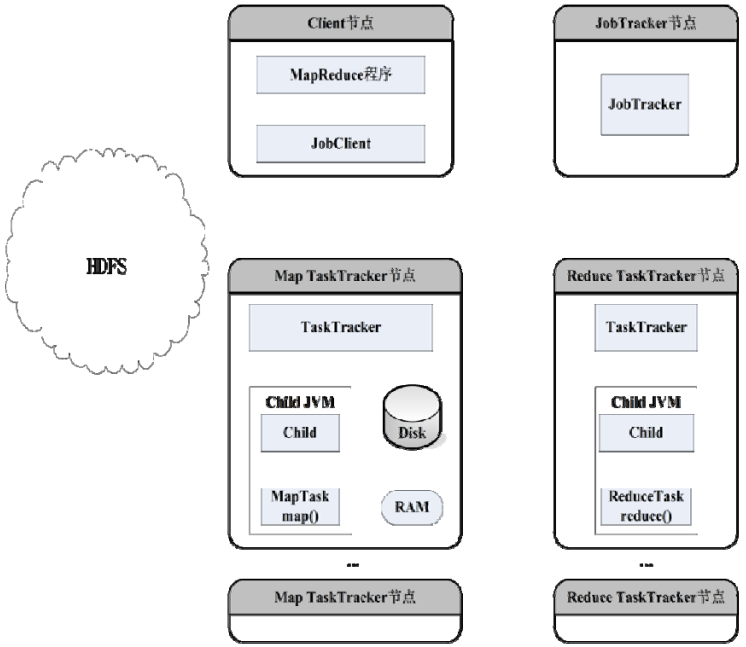
Hadoop MapReduce框架

- Hadoop MapReduce是Hadoop三大组件之一，包括JobTracker和一定数量的TaskTracker。JobTracker负责任务分配和调度，一个MapReduce作业通常会把输入的数据集切分为若干独立的数据块，由Map任务以并行方式处理它们，框架会对Map的输出先进行排序，然后把结果输入到Reduce任务中。通常作业的输入和输出都会被存储在文件系统HDFS中，由JobTracker负责任务的调度和监控，以及重新执行已经失败的任务。

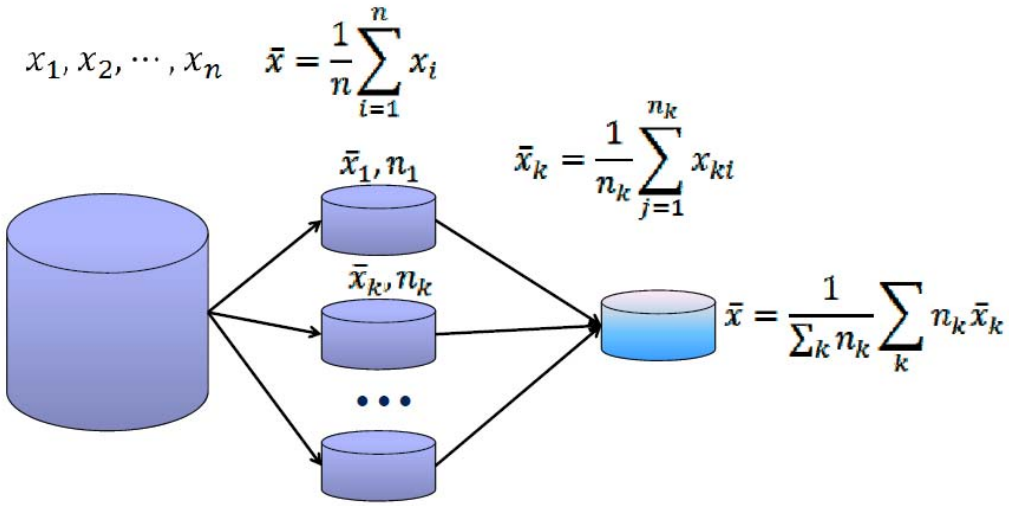


Hadoop MapReduce框架

- Hadoop MapReduce框架由一个单独的主JobTracker和每个集群节点对应一个备TaskTracker组成。JobTracker负责调度作业的所有任务，并监控它们的执行，这些任务分布在不同的备TaskTracker上。如果TaskTracker上的任务执行失败，还会调度其重新执行。而TaskTracker仅负责执行指派的任务。

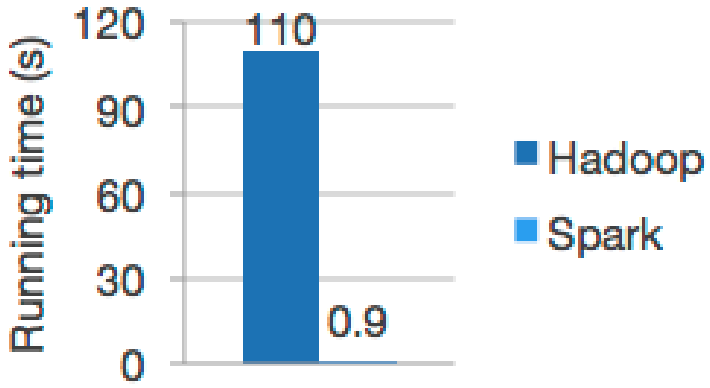


Hadoop MapReduce框架

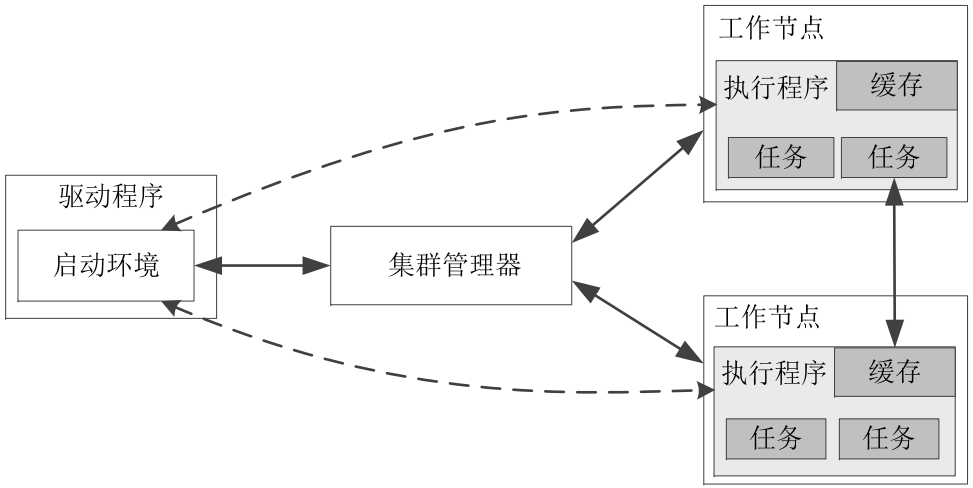


Spark

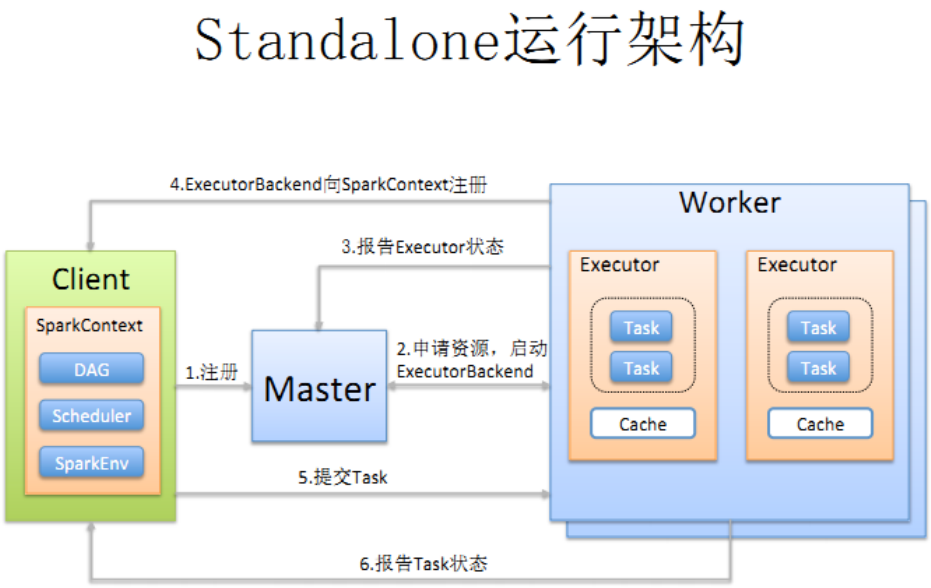
- 与Hadoop MapReduce相比，Spark的优势在于处理迭代计算的机器学习任务，尤其是内存要求小的应用，性能提升很大，Spark还可以进行批处理、实时数据处理、机器学习以及图算法等计算模块。使用Spark平台无需关心分布式并行计算的细节，可以智能地进行数据切分、算法复制、分布执行、结果合并，以支持数据分析人员快速开发分布式应用。



- Spark的基本框架如下图所示



- Spark应用核心由启动环境和执行程序两部分组成，其中执行程序负责执行任务，运行执行程序的机器是工作节点，而启动环境由用户程序启动，通过集群管理器与各个执行程序进行通信。集群管理器主要负责集群的资源管理和调度，目前支持Standalone、Apache Mesos和YARN三种类型的管理器。



Spark

- Spark使用弹性分布式数据集（RDD）抽象分布式计算，RDD是Spark并行数据处理的基础，它是一种只读的分区记录的集合，用户可以通过RDD对数据显示地控制存储位置和选择数据的分区。RDD主要通过转换和动作操作来进行分布式计算，转换是根据现有数据集创建新数据集，动作是在数据集上进行计算后返回值给Driver程序。使用RDD可以用基本一致的方式应对不同的大数据处理场景，还能够提高分布式计算的容错性。

```
val sc = new SparkContext("local", "WordCountApp")

val file = sc.textFile("hdfs://...", 5)

val words = file.flatMap(line => line.split(" "))
                .map(word => (word, 1))

words.cache()

val counts = words.reduceByKey(_ + _)

val top = counts.top(10)(Ordering.by(f => f._2))

println(top.mkString("\n"))
```

Spark

- Spark是一种粗粒度、基于数据集的并行计算框架。其计算范式是数据集上的计算，在使用Spark的时候，要按照这一范式编写算法。所谓的数据集操作，就是成堆的数据，如果源数据集是按行存储的话，就需要对其进行适配，将若干记录组成一个集合。因此在提交给Spark任务时，需要先构建数据集，然后通过数据集的操作，实现目标任务。

TensorFlow

- TensorFlow为用户封装了底层的分布式操作，使其可以专注于编写机器学习代码。使用数据流图进行数值计算，用有向图中的节点表示，节点的状态是可变的，边是张量，对应为多维数组。TensorFlow中数据并行化的方式由In-graph、Between-graph、异步训练、同步训练几种方式，通过将模型训练分配给不同的工作节点，并使用参数服务器共享参数。

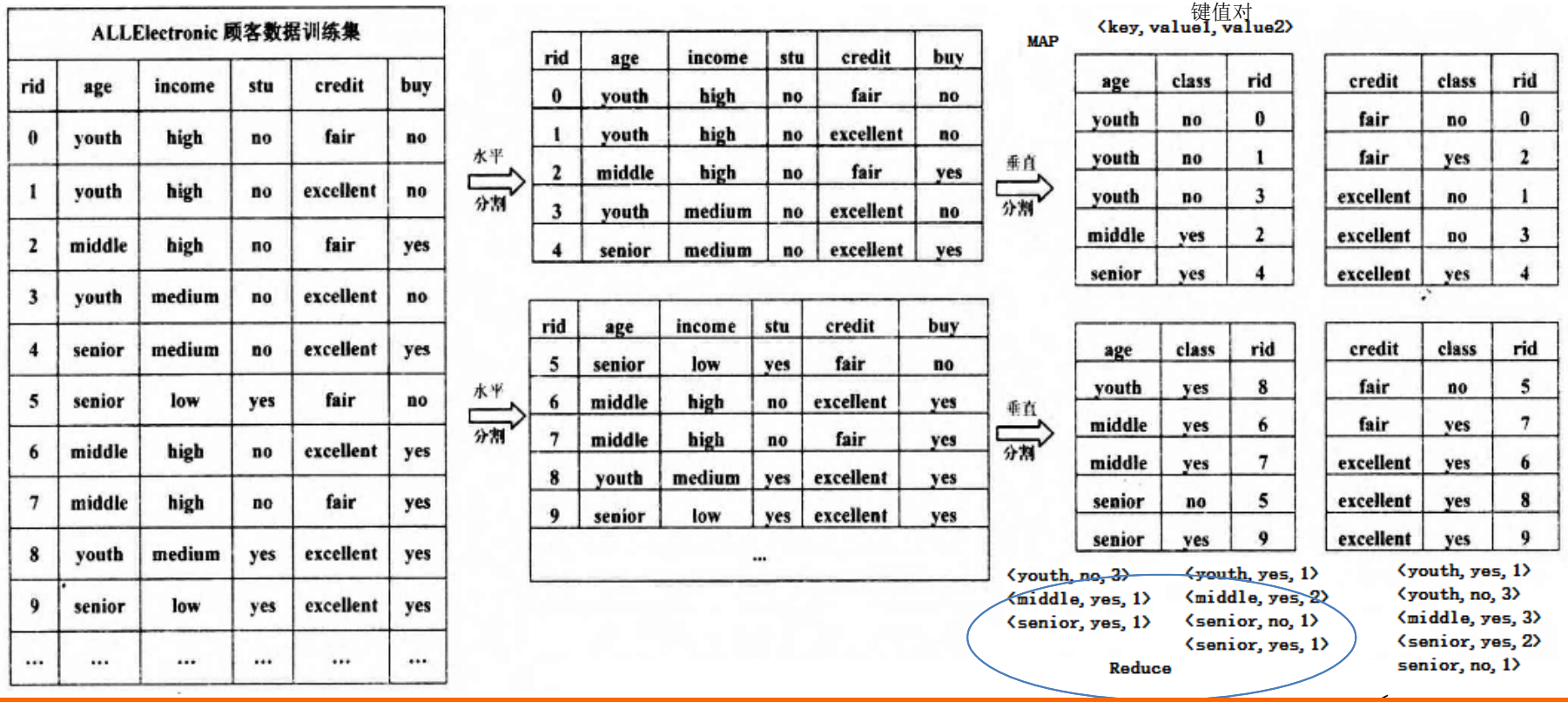
并行决策树

- 随着大数据时代的到来，算法需要处理的数据量急剧增加，仅依靠原始的决策树算法进行分类无论在效率上还是准确性上都不足以满足需求。高效出色的在大数据量下使用决策树算法，需要将决策树算法并行化。

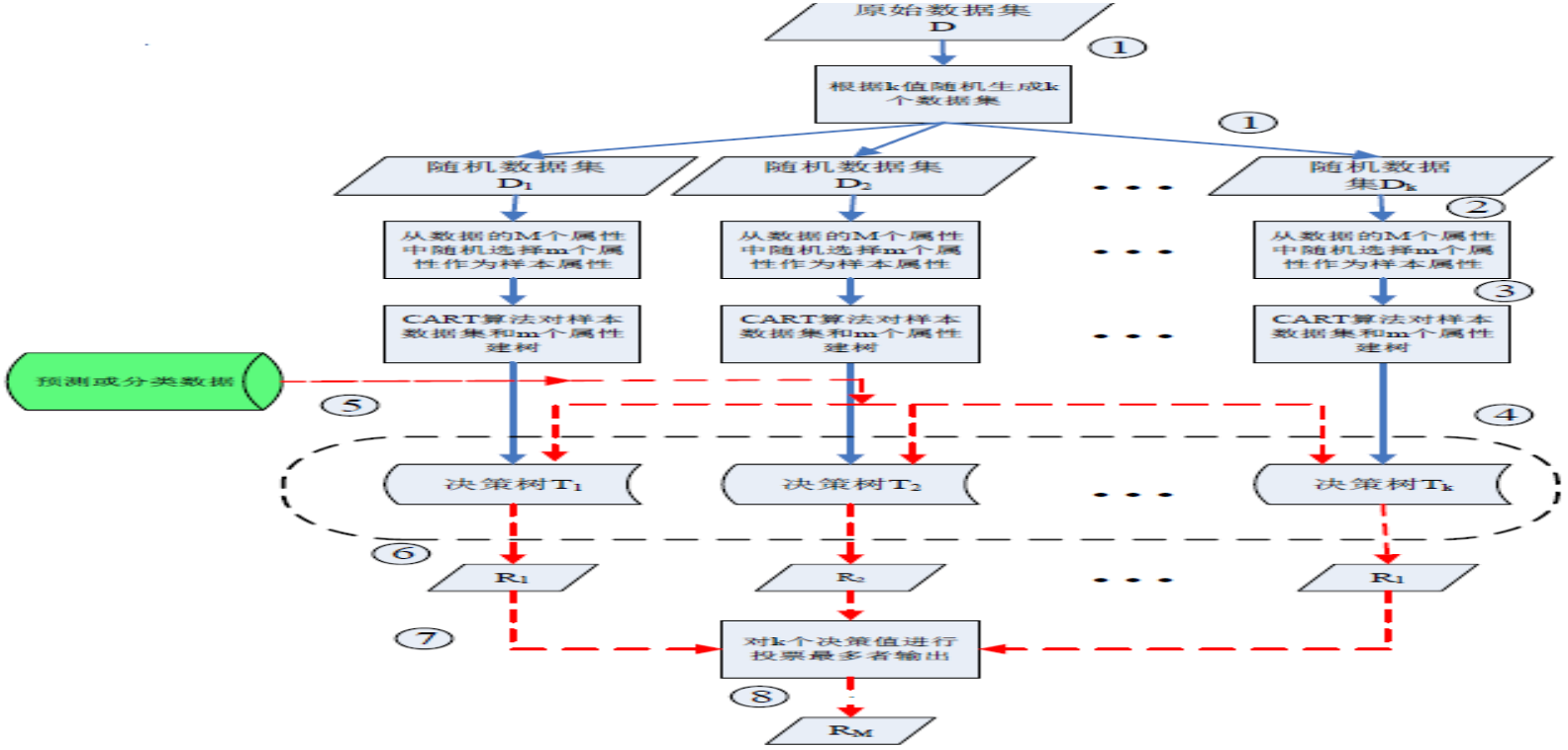
并行决策树

- 并行决策树算法基于MapReduce框架，核心思想是分而治之的策略。MapReduce通过将海量数据集分割成多个小数据集交给多台不同计算机进行处理，实现并行化数据处理。应用到决策树算法上，通过MapReduce将决策树算法并行处理，将耗时的属性相似度计算的步骤并行执行。Map阶段，以单元组形式分解数据，计算属性相似度，以<属性名，相似度>形式输出。Reduce阶段，汇总所有局部结果，找到最大相似度属性名，以这个属性作为测试节点，若是叶子节点，则返回，否则执行分裂，将其录入待计算数据库中进行存储。不断重复上述过程完成决策树的构建。

并行决策树算法


ALLElectronic 顾客数据训练集
rid age income stu credit buy
0 youth high no fair no
1 youth high no excellent no
2 middle high no fair yes
3 youth medium no excellent no
4 senior medium no excellent yes
5 senior low yes fair no
6 middle high no excellent yes
7 middle high no fair yes
8 youth medium yes excellent yes
9 senior low yes excellent yes
... ...
水平分割
水平分割
垂直分割
垂直分割
MAP
键值对
〈key, value1, value2〉
age class rid
youth no 0
youth no 1
youth no 3
middle yes 2
senior yes 4
credit class rid
fair no 0
fair yes 2
excellent no 1
excellent no 3
excellent yes 4
age class rid
youth yes 8
middle yes 6
middle yes 7
senior no 5
senior yes 9
credit class rid
fair no 5
fair yes 7
excellent yes 6
excellent yes 8
excellent yes 9
〈youth, no, 3〉
〈middle, yes, 1〉
〈senior, yes, 1〉
〈youth, yes, 1〉
〈middle, yes, 2〉
〈senior, no, 1〉
〈senior, yes, 1〉
Reduce
〈youth, yes, 1〉
〈youth, no, 3〉
〈middle, yes, 3〉
〈senior, yes, 2〉
senior, no, 1〉

并行化的随机森林——并行CART决策树算法



并行k-均值算法

- k-均值算法是应用最广泛的聚类算法之一，随着大数据的发展，在实际使用过程中如何提升该算法的性能成为了一个有挑战性的任务。可以基于Map Reduce实现k-均值算法，在Hadoop环境中并行运行，能够高效且廉价的处理大型数据集。

并行k-均值算法

- 在具体实现该算法时，将输入数据集存储在分布式文件系统HDFS中，作为<key,value>的序列文件，每个键值对代表数据集的一条记录，其中key记录的是数据文件距离起始位置的偏移量，value是该条记录的内容。将迭代后或初始化后的k个聚类中心放到Configuration中，然后在Mapper的setUp计算读取这k个聚类中心。Mapper会将同一类的数据发送至同一个Reducer。在Reducer中，只需要根据数据重新计算聚类中心即可。

并行k-均值算法

- 使用MapReduce框架实现k-均值聚类算法时，需要将每一次迭代作为一个MapReduce Job进行计算，通过多次运行该Job达到迭代的效果，最终得到k个聚类中心。基于MapReduce的并行k-均值算法，可以在廉价机器上有效处理大型数据集。

k-均值算法算例

- 进行k-均值聚类的数据如下表：

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11
1	2	2	3	9	10	10	11	15	16	16
2	2	5	3	14	13	15	16	6	5	8

k-均值算法算例

- 将x1—x6分配给node1，将x7-x11分配给node2，选择k=3，在开始阶段，创建一个如下表的全局文件。

迭代次数	0	
簇ID	簇中心	样本点数目
1	（1，2）	0
2	（2，2）	0
3	（2，5）	0

k-均值算法算例

- Map阶段对于数据集中的每一个节点，读取全局文件，获得上一轮迭代生成的簇中心信息，计算样本点到簇中心的距离。在Reduce阶段每个reduce收到关于某一个簇的信息。包括该簇的ID和簇的中心以及包含的样本个数。具体如下表。

迭代次数	1	
簇ID	簇中心	样本点数目
1	（1，2）	1
2	（2，2）	1
3	（10.22，9.44）	9

k-均值算法算例

- 一次迭代完成后，进入下一次迭代，直到聚类结果不再发生变化，输出最终得到的聚类结果如右表。

样本点	簇ID
x1	簇1
x2	簇1
x3	簇2
x4	簇2
x5	簇3
x6	簇3
x7	簇3
x8	簇3
x9	簇3
x10	簇3
x11	簇3

多元线性回归模型

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + \epsilon_i$$

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} \quad \text{and} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$L = \sum_{i=1}^{n} \epsilon_i^2 = \boldsymbol{\epsilon}'\boldsymbol{\epsilon} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \mathbf{0} \Rightarrow \mathbf{X}'\mathbf{X}\boldsymbol{\beta} = \mathbf{X}'\mathbf{y}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

并行多元线性回归模型

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1' \\ \vdots \\ \mathbf{x}_n' \end{bmatrix} \quad \mathbf{X}' = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$$

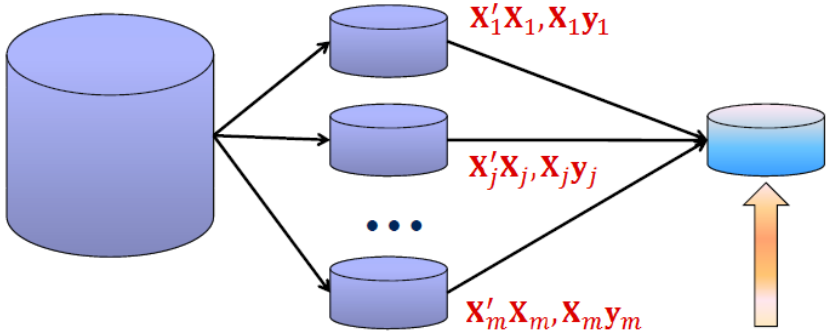
$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$\mathbf{X}'\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \begin{bmatrix} \mathbf{x}_1' \\ \vdots \\ \mathbf{x}_n' \end{bmatrix} = \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i'$$

$$\mathbf{X}'\mathbf{y} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} \mathbf{x}_i y_i$$

$$\mathbf{X}' = [\mathbf{x}_1, \cdots, \mathbf{x}_{n_1}, \mathbf{x}_{n_1+1}, \cdots, \mathbf{x}_{n_2}, \mathbf{x}_{n_2+1}, \cdots, \mathbf{x}_n]$$

$$\mathbf{y}' = [y_1, \cdots, y_{n_1}, y_{n_1+1}, \cdots, y_{n_2}, y_{n_2+1}, \cdots, y_n]$$



$$\hat{\boldsymbol{\beta}} = \left(\sum_{j=1}^{m} \mathbf{X}_j'\mathbf{X}_j \right)^{-1} \left(\sum_{j=1}^{m} \mathbf{X}_j\mathbf{y}_j \right)$$


THANK
YOU