数据库系统概论新技术篇
大数据处理平台Hadoop & Spark及其生态系统

覃雄派

中国人民大学信息学院

2016年12月

大数据处理平台Hadoop & Spark及其生态系统

- 本讲，包括3个部分的内容
  - 1.一个工具不能解决所有问题 (One Size Cannot Fit All)
  - 2. Hadoop 1.0及其生态系统
  - 3. Hadoop 2.0及Spark

1. One Size Cannot Fit All

- 第1部分的具体内容，包括
- 1.1关系数据库管理系统 (RDBMS)的巨大成功
- 1.2 大数据时代的新挑战
- 1.3 RDBMS的局限性

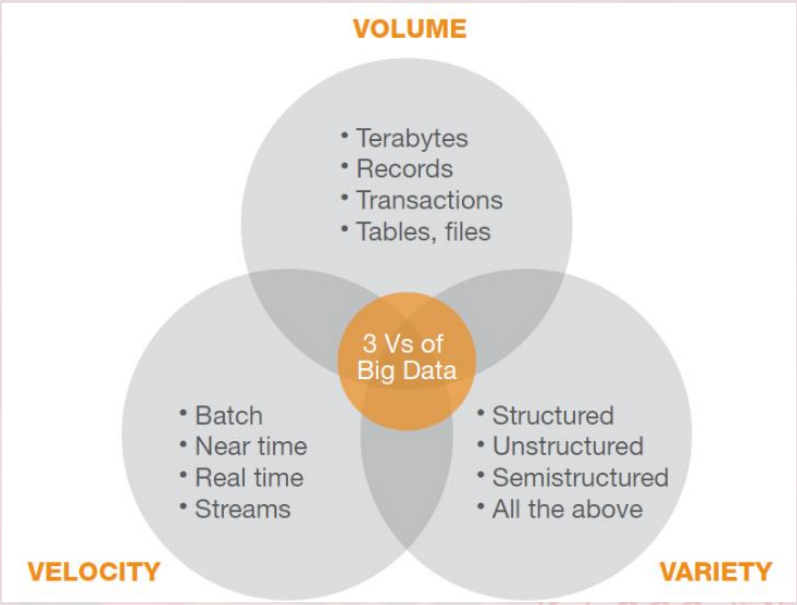
1.1 RDBMS的巨大成功

- 20世纪70年代以来，关系数据库技术由产生到成熟
- 关系数据库系统的优势，包括
  - 易于理解的关系模型
  - 成熟的事务处理技术，保证系统的可靠性
  - 先进的查询优化技术，保证系统的性能
  - 简单易用的SQL语言

- 关系数据库系统在市场上，取得了巨大的成功，成为主流的数据库技术
- 典型的系统包括
  - Oracle、IBM DB2、Sybase、微软SQL Server等商用数据库
  - MySQL、PostgreSQL等开源数据库

1.2 大数据时代的新挑战

- 大数据时代已经来临
- 大数据的来源丰富多样
  - 电子商务、互联网、社交媒体与社交网络、科学计算、物联网与传感器网络等
- 大数据具有3个关键的特点
  - 数据规模大Volume
  - 数据产生的速度快Velocity
  - 数据类型/格式多样Variety



1.3 RDBMS的局限性

- 面向大数据的处理要求，RDBMS具有一些局限性
  - Volume，为了对大数据进行及时有效处理，一般需要建立大型的集群，通过并行计算，分摊数据和计算任务。目前为止，没有RDBMS达到超过1000节点的扩展能力
  - Variety，RDBMS在管理多样的数据方面能力不足，比如对于文本数据、图数据， RDBMS不能有效地管理，需要专门的文档数据库和图数据库
- 一个工具不能解决所有问题(One size cannot fit all), 促使人们研究各种专门的基于非关系数据模型的各类数据库系统，这些数据库统称NoSQL
  - 其中，Hadoop平台是面向大数据批处理的工具

2. Hadoop 1.0及其生态系统

- 第2部分的具体内容，包括
- 2.1 GFS 以及HDFS
- 2.2 MapReduce 运行时Runtime
- 2.3 MapReduce 计算模型 &实例
- 2.4 Hadoop 1.0生态系统
- 2.5 Hadoop 1.0的应用场景
- 2.6 Hadoop 1.0的持续改进
- 2.7 Hadoop 1.0的优势、所获奖项、和局限

2.1 GFS 以及HDFS

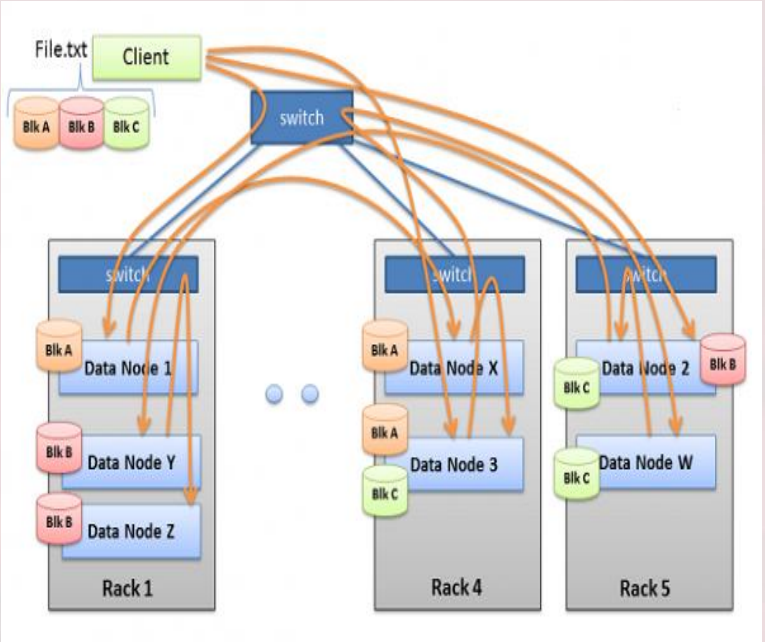
- Hadoop平台的诞生，受到了Google三大论文的启发，包括GFS分布式文件系统、MapReduce计算模型、和Big Table数据库
  - GFS (Google File System)，是运行于大型集群的分布式文件系统，数据以数据块为基本单元进行组织，每个数据块有三个副本，存放于不同节点，实现容错
  - MapReduce，是运行于GFS之上的计算模型
  - Big Table，是基于GFS的、面向结构化数据存储处理的列分组(Column Family)数据库

2.1 GFS 以及HDFS

- Hadoop平台及外围工具，是GFS、MapReduce、Big Table的开源实现
  - 与GFS对应的，是HDFS，即Hadoop分布式文件系统(Hadoop Distributed File System)
  - 与MapReduce对应的，是HDFS上的MapReduce计算模型的实现
  - 与Big Table数据库对应的，是HBase数据库

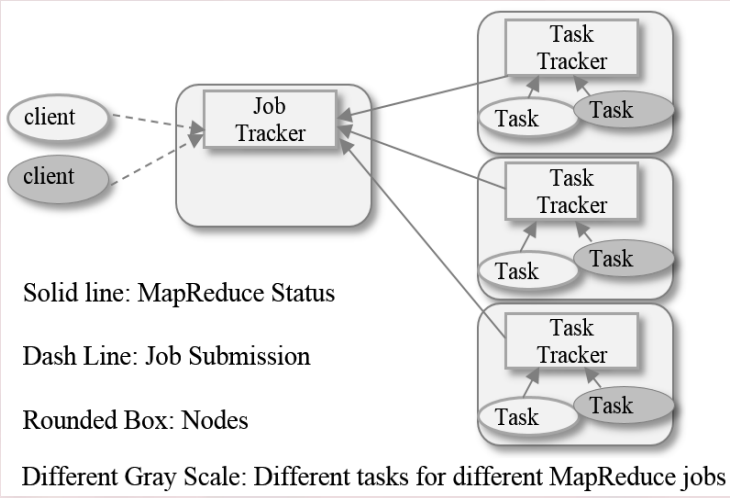
2.1 GFS 以及HDFS

- **HDFS 采用主从系统架构(master/slave).**
  - **NameNode是主节点，负责管理元信息，比如数据块的位置信息**
  - **DataNode是从节点，负责数据块的具体存储和存取操作**
- **数据以64MB/128MB的大小，组织成数据块，每个数据块在集群里不同节点上保存3个副本(Replicas)，以实现存储级的容错**



2.2 MapReduce 运行时runtime

- MapReduce 运行时，运行在HDFS上，包括两个主要的组件，分别是JobTracker和TaskTracker，分别运行在NameNode和各个DataNode上
  - 用户编写MapReduce应用程序，以MapReduce作业(Jobs)的形式，提交给JobTracker运行
  - Job Tracker运行MapReduce作业(Job)，分配Map/Reduce任务(Task), 监控任务执行情况
  - TaskTracker运行Map/Reduce 任务(Task)，对数据进行处理

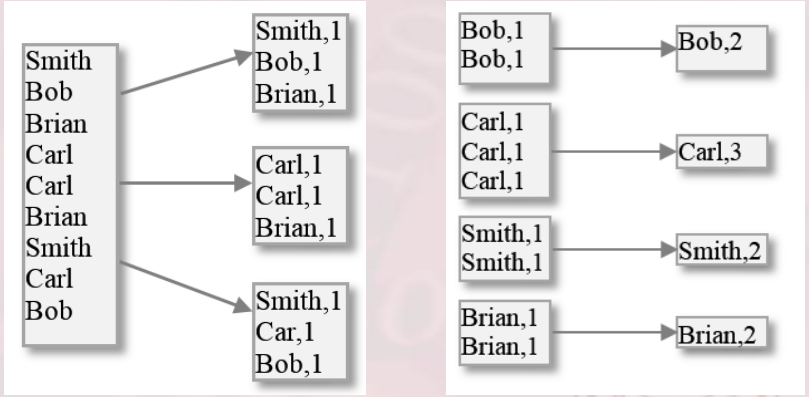


2.2 MapReduce 运行时runtime

- MapReduce作业的运行过程，包括Map和Reduce两个阶段
  - 用户只需编写Map函数和Reduce函数即可，作业和任务的调度、容错等功能，由系统负责
  - 我们通过一个数据分析实例，来了解Map函数和Reduce函数的功能，以及MapReduce作业的调度过程

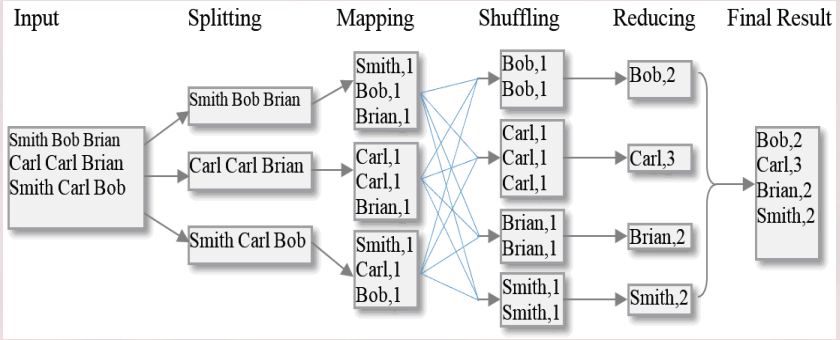
2.3 MapReduce 计算模型 &实例

- 数据分析实例Word Count
  - 对分布式文件系统里的一个文本文件里的不同单词的出现次数进行统计
    - **Map 函数：每个w, 生成<w,1>**
    - **Reduce函数：收集key值**
    - **相同的<w,1>，进行汇总**



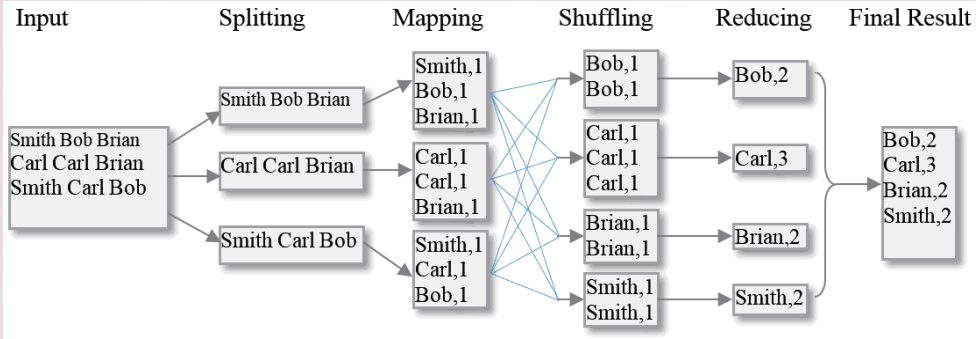
2.3 MapReduce 计算模型 &实例

- 数据分析实例Word Count的执行过程
  - 1. 文件的Load & splitting
  - 2. 用户提交 Word Count
  - 3. Job Tracker启动Map Tasks，Map Task对各个数据块进行操作，中间结果存盘
    - 任务(Task)靠近数据进行调度
    - 所有的数据块都经过Map Task处理以后，Map阶段结束



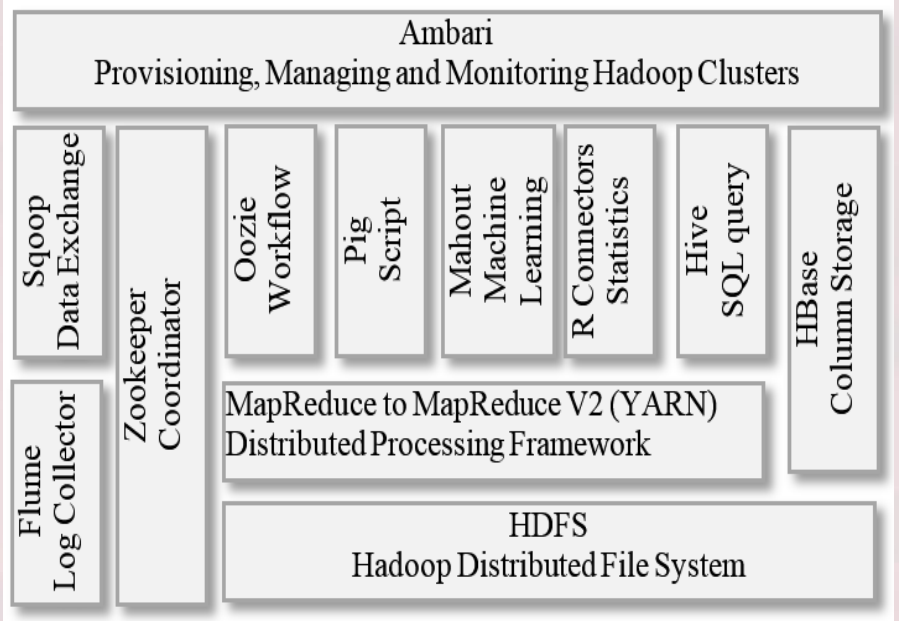
2.3 MapReduce 计算模型 &实例

- 数据分析实例Word Count的执行过程
  - 4. Job Tracker启动Reduce任务
    - 每个Reduce任务从所有Map Task的中间结果里，把应该由它处理的部分拉过来，然后进行合并处理
    - 这个过程称为Shuffle
  - 5. 所有Reduce任务结束后，它们的输出，一起构成结果集



2.4 Hadoop 1.0生态系统

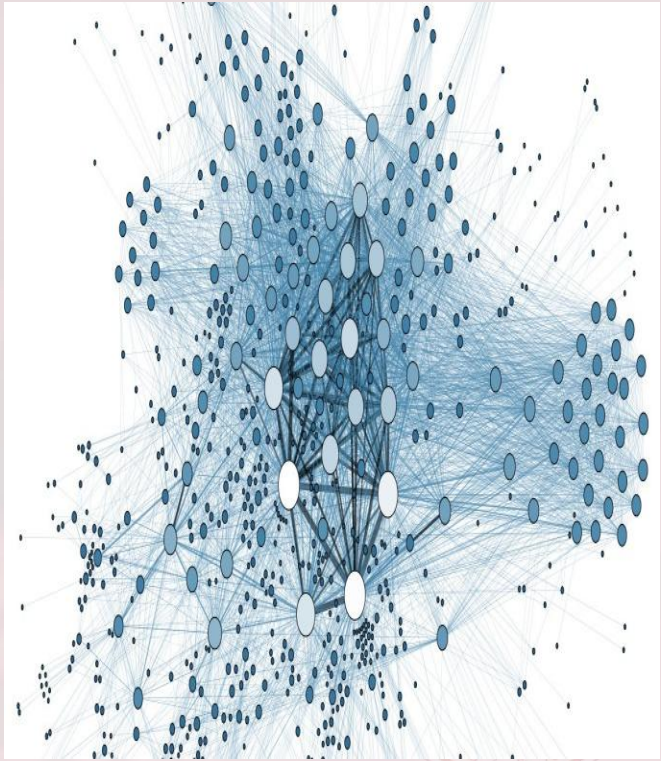
- 基于HDFS以及MapReduce计算模型的若干外围工具，一起构成了Hadoop1.0的生态系统。这些外围工具包括
  - Hive – 结构化数据的SQL查询系统
  - HBase – Big Table的替代者，用于数据服务应用场景
  - Pig – 过程性语言Pig Latin，及其执行环境
  - Flume – 日志数据收集工具
  - Sqoop – 从RDBMS进行数据迁移的工具
  - Mahout – 机器学习软件包
  - Oozie - 工作流调度软件
  - Zookeeper - Google Chubby(分布式锁服务)的开源实现.
    - 提供诸如命名服务、状态同步、配置管理、集群管理等功能。



2.5 Hadoop 1.0 的应用场景

- 除了SQL 统计汇总之外
- 人们已经把很多复杂的算法移植到了Hadoop平台(利用MapReduce计算模型)
  - 包括在线分析处理OLAP、信息检索、数据挖掘、机器学习、科学数据处理、图数据处理等

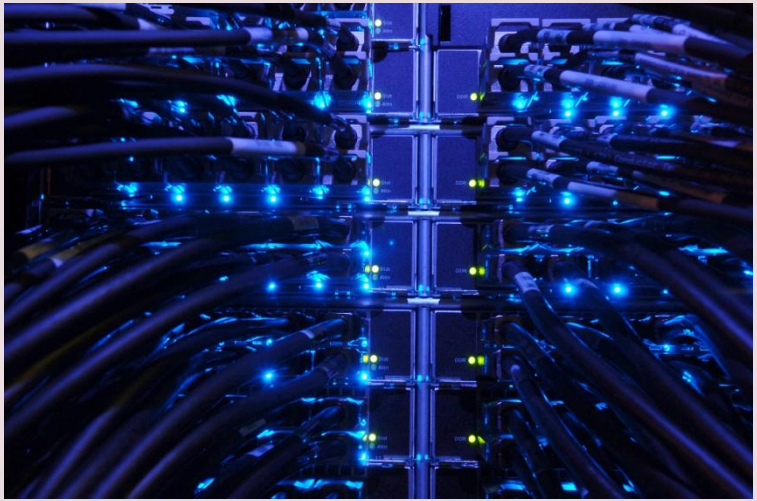
- HDFS并未规定数据块的内部结构
  - 当我们对数据进行适当地组织
  - MapReduce不仅可以对非结构化数据进行处理，也可以对结构化、半结构化数据进行处理



2.6 Hadoop 1.0的持续改进

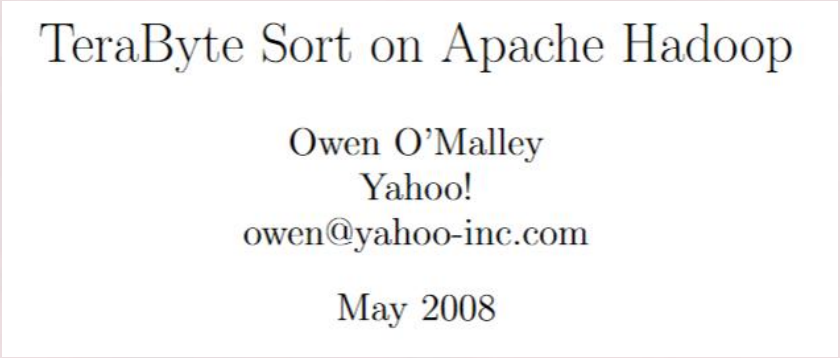
- 研发人员对Hadoop进行了持续的改进，这些改进包括
- 存储层
  - 数据存储格式和放置策略(storage layout & data placement)的优化、对数据倾斜的处理、支持索引、支持多样化的数据等
- 扩展MapReduce计算模型
  - 支持流数据处理、增量式处理、迭代式处理等
- 数据的连接操作Join、以及数据挖掘机器学习算法的实现和优化
  - 两个数据集的连接操作、多个数据集的连接操作、Sita连接操作
- 调度策略
  - 在多核CPU和GPU 环境下的调度策略、异构环境的调度策略、云平台的调度策略
- 易用的用户接口和编程语言
  - Hive用于SQL查询、Pig编写脚本、 System ML编写机器学习程序
- 面向Hadoop平台的节能技术、隐私保护和安全技术

2.7 Hadoop 1.0的优势、所获奖项、和局限



- Hadoop已经成为大数据处理的事实上的标准工具
  - 一说大数据，人们就想到Hadoop，就像一说数据库，人们就想到RDBMS
- Hadoop平台的最大优势是其强大的扩展能力Scalability(可以部署到1000+节点构成的集群上)
  - 对系统进行扩展的成本降低了(Cost to Scale)：Hadoop运行在大量廉价机器构成的集群上，实现对大数据的有效处理
  - I/O效率提高了：利用大规模集群的聚集I/O带宽，可以实现大数据的并行装载和快速处理(数据一定要送入内存，才能进行后续操作)

2.7 Hadoop 1.0的优势、所获奖项、和局限


TeraByte Sort on Apache Hadoop

Owen O'Malley
Yahoo!
owen@yahoo-inc.com

May 2008

- **2008年，Yahoo用910个节点的Hadoop集群，在209秒里，完成了Terabyte数据的排序,打破了以前的排序记录(297s)。**
  - **这件事的意义在于，开源的Java语言编写的软件，第一次赢得这个Sort Benchmark**

2.7 Hadoop 1.0的优势、所获奖项、和局限

- **2011年3月，Hadoop获得Media Guardian 年度创新大奖。**
  - 评审委员会认为， **Hadoop** 是“**21**世纪的瑞士军刀”。
  - **Hadoop**成为大数据处理的事实标准工具。
  - 它的能力、重要性，获得了更多人的认识。

Apache Hadoop takes top prize at Media Guardian Innovation Awards

Data management software described by judges as 'Swiss army knife of the 21st century' wins innovator of the year award

The full list of winners

2.7 Hadoop 1.0的优势、所获奖项、和局限

- Hadoop1.0具有两个明显的局限
- (1) 它仅仅支持一种计算模型，那就是MapReduce，表达能力有限。
  - 复杂的数据操作，比如Join连接操作，需要转换成一系列的MapReduce作业，一个接一个运行，才能得到最终结果，这个过程不是这么直观。
- (2) 在MapReduce计算模型中，Map阶段和Reduce阶段之间，以及MapReduce作业之间，中间结果要持久化到磁盘上，实现了计算上的容错。
  - 这给系统的性能造成了较大的影响，即造成比较大的延迟，很难支持交互式的应用。

3. Hadoop 2.0 and Spark

- 第3部分的具体内容，包括
- 3.1业务 (商业)需求推动技术创新
  - 业务需求推动Hadoop的发展和Spark的诞生
- 3.2 Hadoop 2.0 及其生态系统
  - Hadoop 2.0 运行时runtime
  - Hadoop 2.0支持的计算模型
  - Hadoop 2.0 的主要优势
  - Hive on MapReduce、到Tez 以及 Hive on Tez
- 3.3 Spark及其生态系统
  - Spark 生态系统及其组件 - Spark Core, Spark SQL, GraphX, Streaming, MLLib
  - Spark 技术细节：RDD, TRANSFORMATION 和 ACTION, DAG, 窄依赖和宽依赖, DAG调度器 (DAGScheduler)，DataFrame
  - Spark 的性能评测结果及所获奖项
- 3.4 Hadoop and Spark – 共存及竞争关系
  - Hadoop/Spark在未来数据仓库系统中的角色

3.1 商业需求

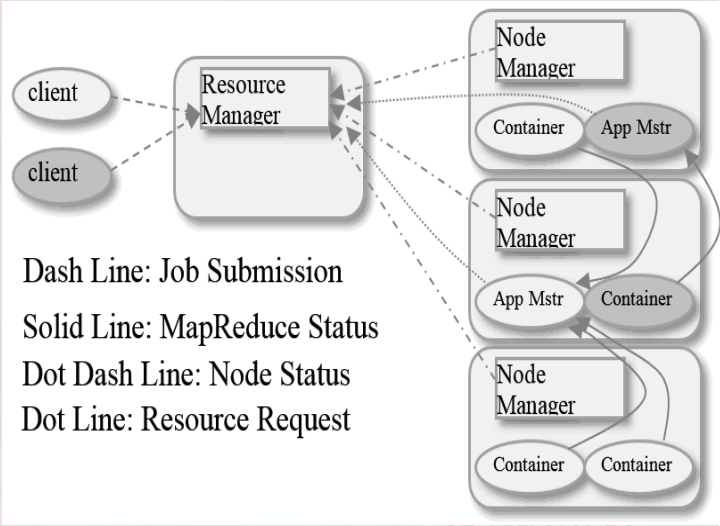
- **Hadoop1.0**展示了其处理大数据的强大能力。
- 但是它在如下两个方面，先天不足。
- 交互式查询
  - 人们希望尽快获得对数据的理解。但是**Hadoop 1.0(MapReduce**计算模型**)**不支持对数据进行交互式的查询分析。
- 迭代处理
  - 此外，**MapReduce**计算模型也不能很好地支持迭代式计算，这是机器学习算法所必须的。
  - 机器学习算法，需要对数据进行多次迭代处理，才能得到结果。

3.2 Hadoop 2.0 及其生态系统

- **Hadoop 2.0 运行时runtime**
  - 商业需求的推动，促使Hadoop开发者改进和提升Hadoop平台
  - 他们推出了Hadoop 2.0(关键技术是YARN资源管理器，Yet Another Resource Negotiator)
  - YARN最大的特点是，它把在Hadoop1.0中紧密绑定的MapReduce计算模型和资源管理功能，进行了分离。

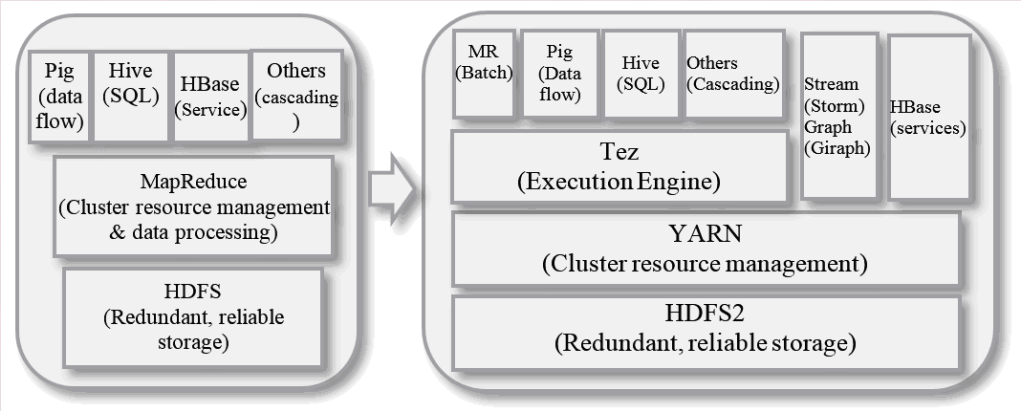
3.2 Hadoop 2.0 及其生态系统

- Hadoop 2.0 运行时runtime，主要包括如下组件
  - ResourceManager 运行于Master Node
    - 由Scheduler 和ApplicationsManager 构成
    - Scheduler 负责为应用程序分配资源
    - ApplicationsManager 接受用户提交的作业，申请第一个容器container资源，运行该应用程序的ApplicationMaster
    - ApplicationMaster继续和 Scheduler 谈判，获取容器container资源，然后和NodeManager配合运行应用程序
  - NodeManagers 运行于Slave Node
    - 启动应用程序的container容器，监控其运行，报告给ResourceManager



3.2 Hadoop 2.0 及其生态系统

- **Hadoop2.0支持的计算模型**
  - **Hadoop2.0中，YARN实现了资管管理，在此之上，可以运行若干数据处理模型(或者计算模型)，包括传统的MapReduce作业(批处理)、Tez(交互式查询处理)、Spark(内存迭代式处理)、Storm(数据流的实时处理)、以及GraphLab/Giraph(图数据处理)等。**
  - **简言之，YARN把Hadoop从一个支持单一计算模型(主要进行数据的批处理)的平台，转变成支持多种计算模型的平台，包括交互式查询处理。**



3.2 Hadoop 2.0 及其生态系统

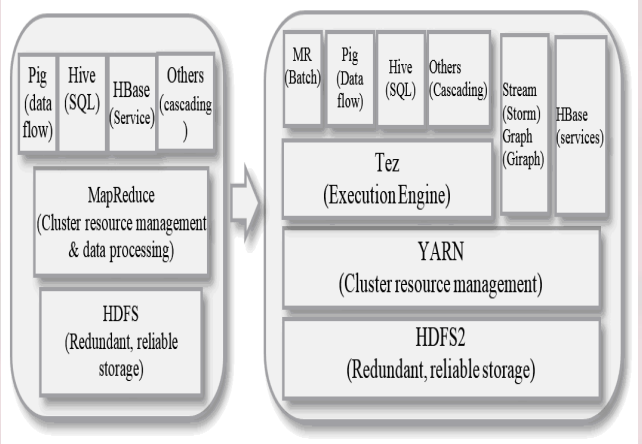
- **Hadoop 2.0 的主要优势**
  - **(1) 更高的扩展能力 – 达到上10,000节点规模**
  - **(2) 资源利用效率提高 – ResourceManager可以根据各个应用的资源需求、公平性、SLA (service level agreements)协议等要求，优化资源调度，提高资源利用效率**
  - **(3)多样化的负载 – 除了MapReduce计算模型，Hadoop2.0支持更多的负载类型，包括交互式查询、流数据处理、迭代式处理、图数据处理等**
  - **(4)灵活性 - Hadoop2.0具有后向的兼容性，也就是早先为Hadoop1.0平台编写的MapReduce程序，无需修改，就可以在Hadoop2.0上运行。**

3.2 Hadoop 2.0 及其生态系统

- ❖ **Hive on MapReduce、到Tez 以及 Hive on Tez**
  - ■ **Hive on MapReduce**
    - **Hive是Hadoop平台上的数据仓库系统**
    - **SQL查询被转换成一系列的MapReduce Job，依次执行**
    - **Hive利用HDFS的扩展能力、以及MapReduce计算模型的并行性，实现大数据的查询处理**
    - **受限于MapReduce计算模型，Hive的性能比MPP数据库差很多**

3.2 Hadoop 2.0 及其生态系统

- Hive on MapReduce、到Tez 以及 Hive on Tez
- Tez及其特点
  - Tez是Hadoop2.0的新的查询执行引擎，Tez运行在YARN之上。
  - MapReduce作业、Hive查询、Pig程序等，运行在Tez上，获得更高的性能。
  - Tez把数据的分析处理工作，表达成DAG(directed acyclic graph)形式。
  - 通过把一系列的MapReduce Job以一个Tez Job来运行，通过消除不必要的任务、任务同步机制(synchronization barriers)、以及中间结果存盘等，Tez提高了数据处理的速度。
  - 数据处理任务，包括小规模数据上的低延迟查询，以及大量数据上的高吞吐量负载，都能够从Tez获得性能收益。



3.2 Hadoop 2.0 及其生态系统

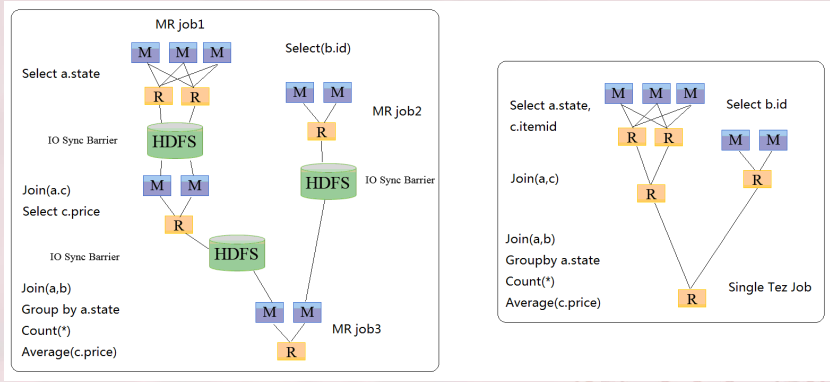
- ❖ **Hive on MapReduce、到Tez 以及 Hive on Tez**
  - ■ **Hive on Tez**
  - ■ **Tez 是 Hive 1.3 的新的执行引擎**

3个MapReduce Job　1个Tez Job

```
SELECT a.state, COUNT (*), AVERAGE(c.price)

FROM a JOIN b ON (a.id = b.id)
JOIN c ON (a.itemId = c.itemId)

GROUP BY a.state
```

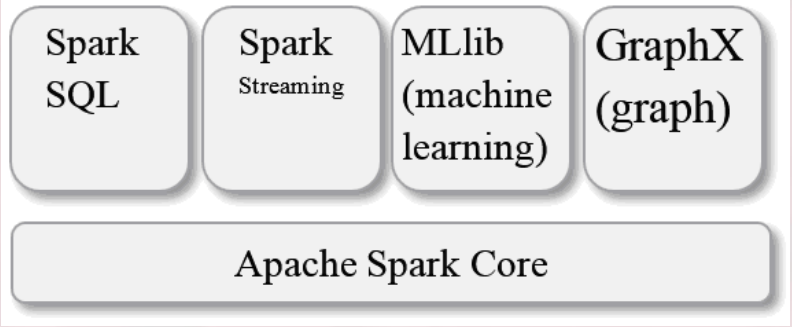


3.3 Spark 及其生态系统

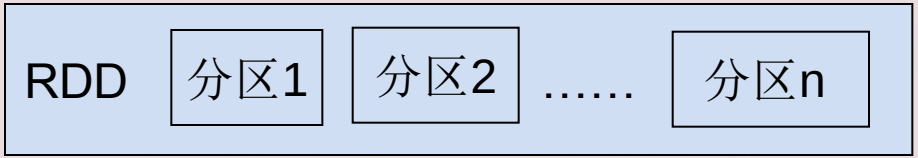
- **Spark 生态系统及其组件**
  - **Spark大数据处理软件，于2010年由 University of California at Berkeley研发和发布。**
  - **目前， Spark 及其生态系统(相关工具)，成为Hadoop生态系统的一个 竞争者和替换者(alternative)。虽然Hadoop平台试图把Spark作为整个平台的一个组件，实际上Spark及其生态系统，提供了Hadoop及其生态系统所提供的功能。**
  - **目前，最大的Spark平台，部署在超过8000节点的集群上。**

3.3 Spark 及其生态系统

- **Spark 生态系统，包含如下的主要组件**
  - **Spark Core是核心模块**
  - **Spark SQL支持结构化数据的SQL查询**
  - **Spark Streaming支持流数据处理**
  - **MLlib是一个机器学习软件包**
  - **GraphX是一个图数据处理和分析软件包**



3.3 Spark 及其生态系统



- **Spark 技术细节：RDD**
  - **Spark Core使用一种特殊的内存数据结构RDD (Resilient Distributed Datasets)，进行数据管理。**
  - **RDD 是不可更新的、分区的数据集，方便利用集群，进行并行处理。**
  - **任何数据可以保存在RDD中，包括结构化数据和非结构化数据。**
  - **RDD通过装载HDFS/HBase的数据而创建，或者从其它的RDD经过转换(transform)操作得来。**
  - **RDD使用基于血缘关系的技术，实现容错保证。当下游RDD损坏或者丢失，那么从上游RDD，经过一系列的转换操作(transform)，重建丢失的RDD。**
  - **这种上下游RDD之间的转换生成关系，也就是数据的处理流程，通过DAG (Directed Acyclic Graph有向无环图)来描述。**

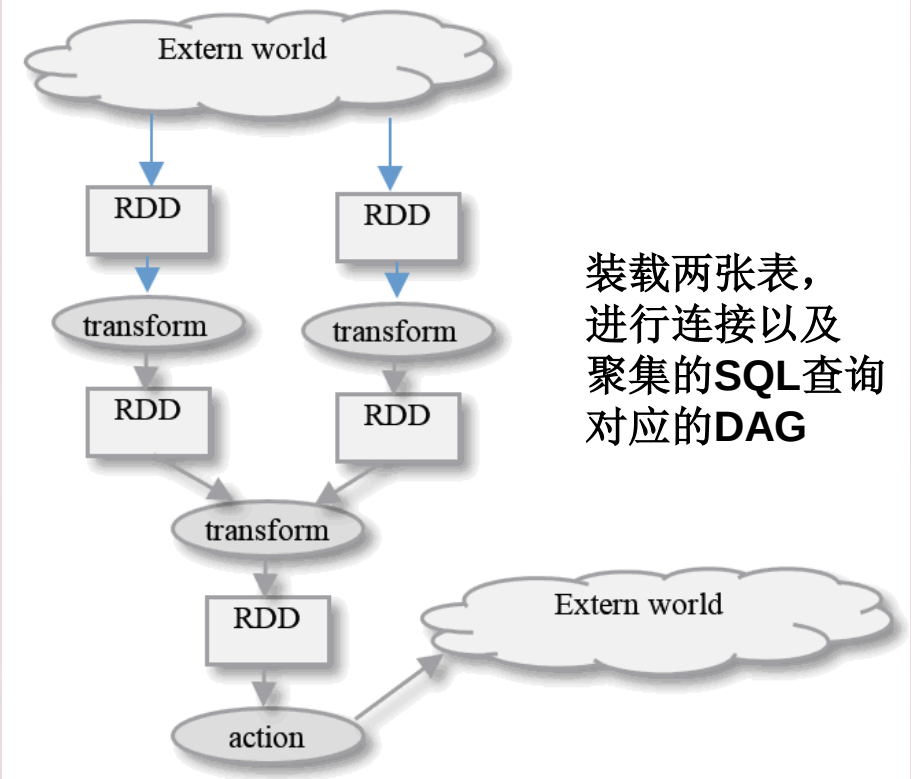
3.3 Spark 及其生态系统

❖ Spark 技术细节：Transformation & Action

- 数据处理的流程，一般表达成依次施加于**RDD**的一系列的转换*TRANSFORMATION*和动作*ACTION* 。
  - 所谓转换**TRANSFORMATION**操作，它施加于一个或多个**RDD**，产生一个新的**RDD**，比如**Join**、**Union**、**filter** 、**map** 等。它是一种粗粒度的操作。
  - 所谓动作**ACTION**操作，它应用到**RDD**上，产生一些结果值，比如**count**、**first**、**Reduce**等操作。

3.3 Spark 及其生态系统

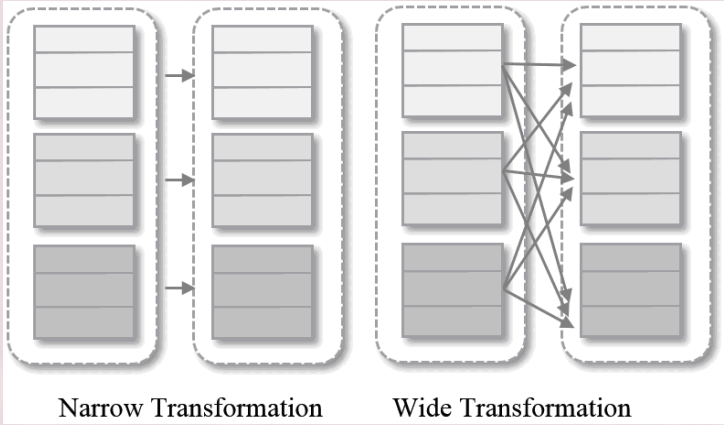
- **Spark 技术细节：DAG**
  - 数据处理流程，一般表达成一系列的施加于RDD的转换transformation操作、动作action操作。
  - 转换transformations操作，表达的是如何从已有的RDD，经过处理，生成新的RDD。
  - 动作actions操作，表达的是如何产生最终结果。
  - 一系列RDD、 transformation、和action，构成了一个DAG，表达了数据处理的流程。



装载两张表，进行连接以及聚集的SQL查询对应的DAG

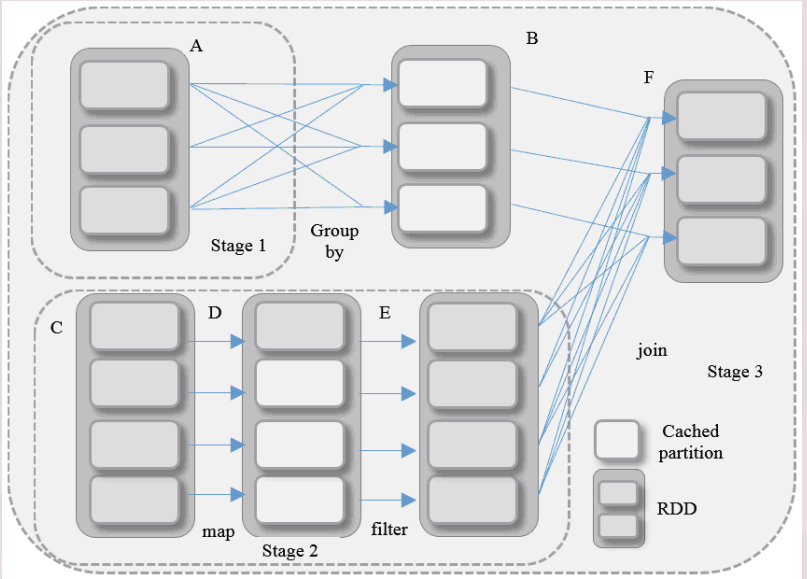
3.3 Spark 及其生态系统

- Spark 技术细节：宽依赖与窄依赖
  - 父RDD和子RDD的分区之间，有两种依赖关系，分别是宽依赖和窄依赖。
  - 窄依赖
    - 父RDD的每个分区，最多被一个子RDD分区用到
  - 宽依赖
    - 多个子RDD分区，依赖于同一个父RDD分区
    - group-by-keys, reduce-by-keys, sort-by-keys 等操作，需要宽依赖才能获得正确结果
  - 对窄依赖的处理(从父RDD产生子RDD)，可以在各个节点上独立完成，无需数据Shuffle。而宽依赖，则需要通过网络，实现数据的Shuffle。



3.3 Spark 及其生态系统

- Spark 技术细节：DAG调度器(DAGScheduler)
  - DAGScheduler 是Spark的作业调度软件模块，它实现了面向阶段的调度策略。
  - 用户提交的一个数据处理程序(比如一个查询)，称为一个作业Job。
  - DAGScheduler检查RDD之间的依赖关系，把所有相邻的窄依赖组织成一个阶段。而宽依赖，则需要跨越不同的阶段。
  - 一个阶段Stage，是并行任务的集合。每个任务是对RDD的一个分区的处理任务，是数据处理的一个基本单元，作业job的一部分。



实现了两个表的连接、和聚集的查询作业Job，对应的3个阶段Stage

3.3 Spark 及其生态系统

- **Spark 技术细节：DataFrame**
  - 在Spark的早期版本中，DataFrame称为SchemaRDD
  - 本质上，它是在RDD的上面增加了一个数据模式的描述，对数据集的各个列给出了名称、数据类型等元信息描述
  - DataFrame给RDD增加了一个抽象描述层，为SparkSQL服务
  - DataFrame API以一种延迟的方式(Lazy manner)，对数据上的操作进行计算(Evaluate)，使得优化器有机会对关系运算和整个数据处理流程进行优化
  - 开发人员，可以在Scala和Java程序里，调用DataFrame API，把过程性语言和关系操作结合起来，对数据进行处理

3.3 Spark 及其生态系统

- **Spark 的性能评测结果及所获奖项**
  - **University of California at Berkley 的AMP Lab对Redshift, Hive (v0.12), Shark, Impala(v1.2.3), and Stinger/Tez(v0.2.0) 等系统进行了性能评测**
    - **Redshift 是Amazon.com提供的一个MPP 数据库(基于ParAccel 数据仓库)。**
    - **Shark是SparkSQL的前身。**
    - **Impala是Cloudera提供的HDFS之上的结构化数据处理引擎，它采用类似MPP数据库的查询处理技术。**
    - **Shark和Impala都和Hive数据仓库兼容。**

3.3 Spark 及其生态系统

❖Spark 的性能评测结果及所获奖项

- 数据集包括三张表。**Ranking** 表保存每个网页的**rank**。**UserVisits**表保存每个网页的访问日志。**Documents**表保存非结构化的**HTML**网页内容。
- 负载包括4个查询。
  - **Query 1 (**扫描查询**scan query) 和 Query 2 (**聚集查询 **aggregation query)** 属于探索类**SQL**查询**(exploratory SQL queries)**。
  - **Query 3** 是一个连接查询 **join query**，结果集是一个很小的集合。
  - **Query 4** 是一个大批量数据处理的带**UDF(**用户自定义函数**)**的查询**(bulk UDF query)**。它对**Common Crawl** 数据集**(**http://commoncrawl.org/**)**的一个采样**(sample)**，运行简化版本的**PageRank**算法，计算**PageRank**。

3.3 Spark 及其生态系统

❖Spark 的性能评测结果及所获奖项

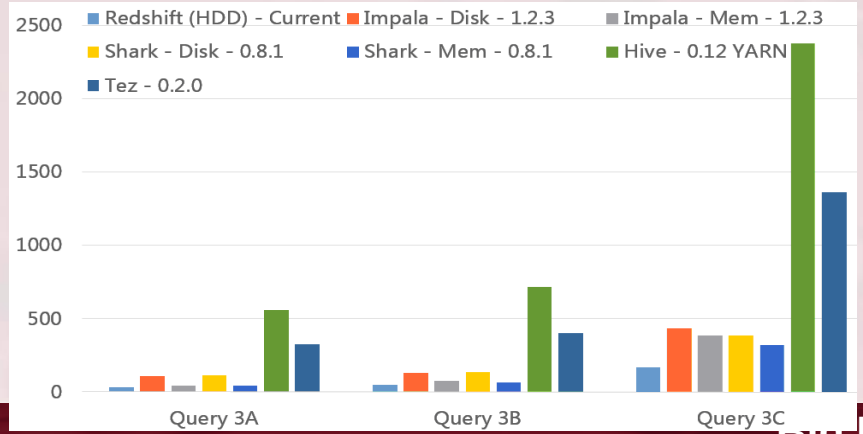
- 对于Query1(扫描查询scan query)，当整个数据集保存在内存中，对于所有的选择率(selectivity)，Spark获得最好的性能。
- 对于Query2(聚集查询aggregation query)，运行于内存数据集上的Spark，获得比其它系统更高的性能(RedShift除外)。

3.3 Spark 及其生态系统

❖ Spark 的性能评测结果及所获奖项

- 对于 Query 3 (连接查询join query)，当选择率较低的时候，Spark (in-memory)获得和Redshift以及Impala类似的性能，并且比其它配置或者系统获得的性能好得多。
  - 当选择率较高的时候， Spark (in-memory)获得比其它系统或者配置更好的性能(除了Redshift，它的性能总是比Spark要好) 。(in-memory表示整个数据集驻留在内存中)
- 对于 Query 4 (包含UDF的查询), Spark (in-memory) 比 Hive on MapReduce,和Hive on Tez的性能都好很多。

各个系统或者配置的
Query 3查询性能



3.3 Spark 及其生态系统

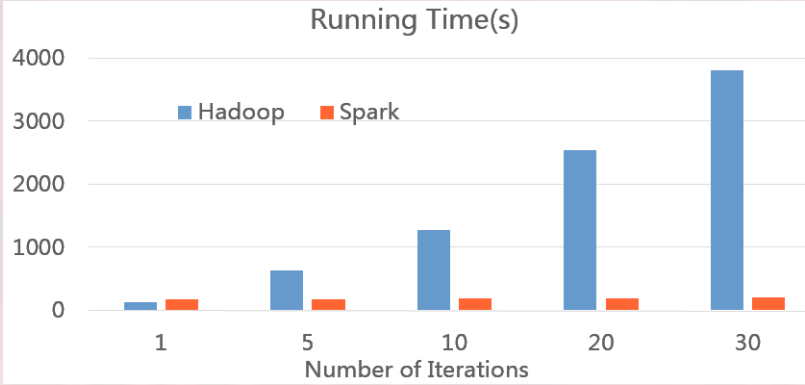
❖Spark 的性能评测结果及所获奖项

- Spark 的开发者，还对Spark和Hadoop进行逻辑回归(logistic regression)的性能进行了比较。
- 他们在20个Amazon“m1. xlarge” EC2虚拟节点构成的集群上，进行实验，每个节点有4个内核(Cores)。
  - EC2是amazon云服务的计算节点。
- 他们使用的数据集大小是29GB。

3.3 Spark 及其生态系统

❖Spark 的性能评测结果及所获奖项

- Hadoop的每一次迭代需要127秒，因为每次迭代都是独立的MapReduce作业。
- Spark的第一次迭代，耗时174秒(实验者认为，原因可能是因为使用Scala而不是Java)，后续的迭代，使用了缓存的数据，每次仅仅耗时6秒，使得整个逻辑回归的速度提高10倍以上。



3.3 Spark 及其生态系统

❖Spark 的性能评测结果及所获奖项

- 2014年, Spark赢得了 Daytona GraySort contest (http://sortbenchmark.org/)。
- Databricks公司的工程师，在206个EC2节点构成的集群上，耗时23分钟，完成100TB磁盘数据的排序。
  - 早先的由Hadoop创造的世界纪录是，在2100个节点构成的集群上，耗时72分钟，完成100TB数据的排序。
- 这个结果显示，Spark可以使用更少的硬件资源(10X fewer machines)，更快地(3X faster)完成等量数据的排序工作。
  - 这个排序是对存储在磁盘上的HDFS文件进行的，并没有使用Spark的内存缓冲(in-memory cache)。

3.4 Hadoop and Spark – 共存或者竞争

- **Hadoop/Spark在未来数据仓库系统中的角色**
  - **Hadoop和Spark两者之间，有功能上的重叠，以及相互之间的依赖。**
    - **Hadoop和Spark都能够管理大型的数据集，上层工具提供类似的分析能力。**
    - **Spark本身没有自己的文件系统，它依赖于分布式文件系统比如Hadoop的HDFS和某种存储格式，比如Parquet (HDFS上的一种列存储结构)。**
      - 备注：**RDD是面向内存的存储结构。**

3.4 Hadoop and Spark – 共存或者竞争

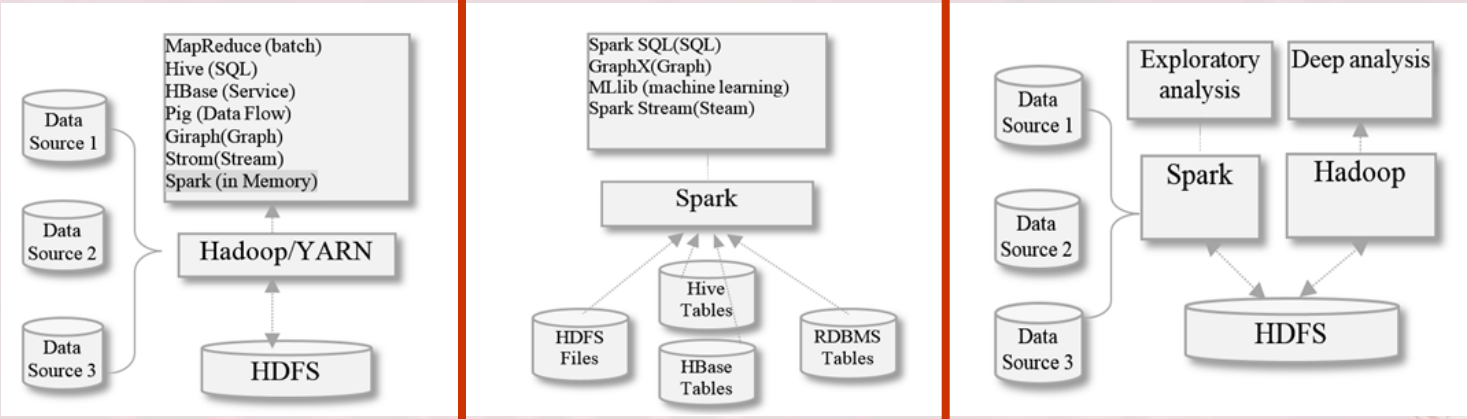
- **Hadoop/Spark在未来数据仓库系统中的角色**
  - **Spark的目标是成为统一的数据集线器(Data Hub),它不仅能够处理内生(Native)的数据集(比如从磁盘Parquet列存储文件里装载的RDD内存数据集)，而且它能够接受任何来源的数据，进行处理，包括来自RDBMS比如MySQL的数据、来自流数据处理系统比如Storm的数据、消息队列比如Kafka的数据、以及其它大数据处理系统的数据比如Hive和HBase。**
  - **对于Hadoop来讲，YARN的引入，把资源管理和编程模型两个功能模块分开了，它支持批处理MapReduce作业、交互式查询、迭代式的机器学习算法、流数据处理、图数据处理等。**
    - **Yarn把Hadoop从一个大数据批处理工具，变成一个支持多种处理模型的工具，包括支持交互式查询处理。**
    - **Spark可以作为一个模块，嵌入到Hadoop平台中，作为一种应用程序类型(Application Type),利用内存处理技术，提供大数据的交互式查询能力。**

3.4 Hadoop and Spark – 共存或者竞争

- **Hadoop/Spark在未来数据仓库系统中的角色**
  - **Hadoop和Spark及其生态系统(相关工具)，将在长期的时间里共存，并且各自发展成更加先进的平台和工具。**
  - **但是，不可否认的是，两者具有竞争的关系。**
    - **近年来Spark获得了比Hadoop更多的注意力。Hadoop平台上的机器学习软件包Mahout，在2009年迁移到Spark平台上，以获得更高的性能。**
    - **当然，Hadoop社区也在持续地改进Hadoop系统，以降低查询延迟，这是Spark的强项。**
  - **于是，两个平台将继续共存和竞争。**

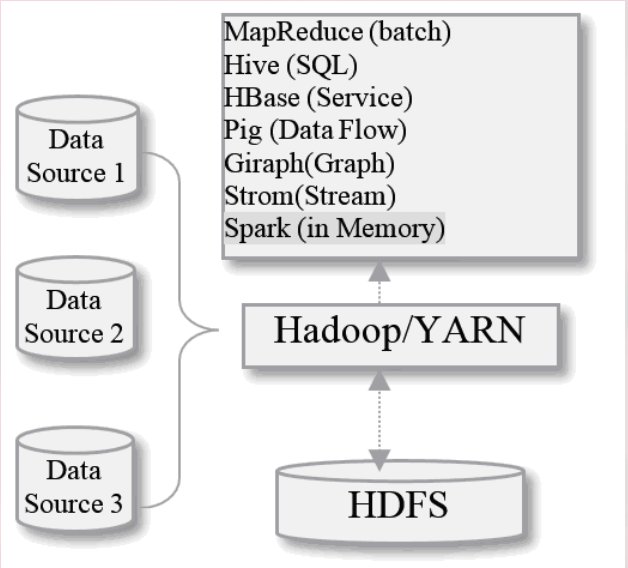
3.4 Hadoop and Spark – 共存或者竞争

- **Hadoop/Spark在未来数据仓库系统中的角色**
  - **Hadoop和Spark有三种共存的方式，即 *Hadoop dominant*, *Spark dominant*, 以及 *Co-Exist***



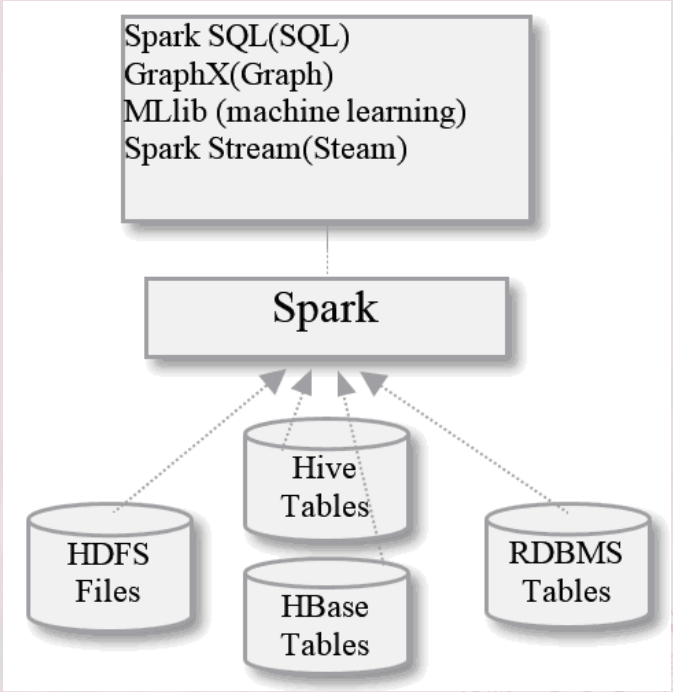
3.4 Hadoop and Spark – 共存或者竞争

- **Hadoop/Spark在未来数据仓库系统中的角色 - Hadoop dominant**
  - **Hadoop**是集中所有数据的后台系统。
  - 数据可以是结构化的，也可以是半结构化的或者非结构化的数据。
  - 在**Hadoop**平台上，人们可以进行简单的统计汇总、以及执行复杂的机器学习和数据挖掘算法。
    - 在这种场景下，**Spark**作为**Hadoop**的一个组件，通过基于内存的数据处理，提供交互式分析的能力。
    - **Spark**的优势是对大小适中的数据集，进行快速的分析处理。



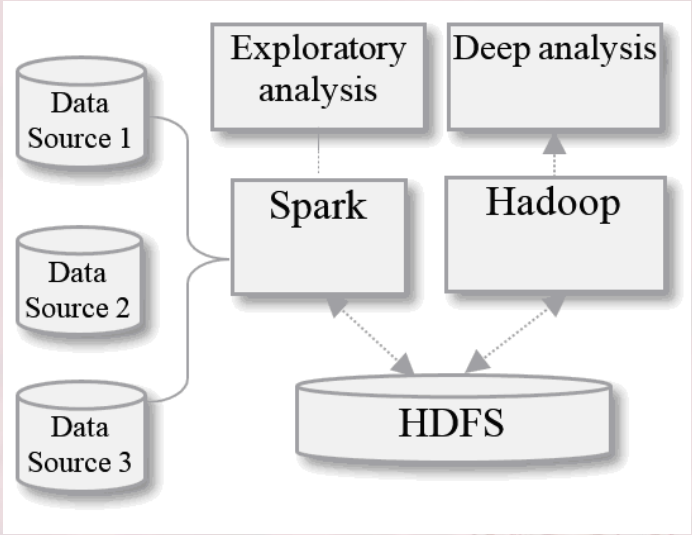
3.4 Hadoop and Spark – 共存或者竞争

- **Hadoop/Spark在未来数据仓库系统中的角色 - Spark dominant**
  - **Spark社区试图把Spark开发成为一个功能全面的数据处理框架或平台，在数据仓库中占据主导地位。**
  - **Spark作为数据集线器(Data Hub)，接受来自各种来源的数据，包括流数据、文件、以及来自关系数据库 RDBMS的数据。**
  - **它把各种数据加载到数据分析处理流水线上，Spark是集中所有相关数据的目的地，人们将依赖于Spark获得对数据的洞察。**
    - **在和Hadoop的关系方面，Spark可以从HDFS文件系统、Hive数据库、HBase数据库等系统抽取数据。**
    - **在这样的架构中，Spark统治一切。**



3.4 Hadoop and Spark – 共存或者竞争

- **Hadoop/Spark在未来数据仓库系统中的角色 - *Co-Exist***
  - 首先数据被送到**Spark**，人们在那里对数据进行探索式分析。
  - 接着数据被转交给**Hadoop**，以便进行进一步的分析(深入分析)。
  - 对数据进行深入的分析需要跨越较长时间的数据(更大时间范围)，需要更多的时间来完成。
  - 人们充分利用其各自的长处，**Spark**和**Hadoop**在数据处理的不同阶段，发挥不同的作用。



大数据处理平台Hadoop & Spark及其生态系统

- 回顾
  - 1.一个工具不能解决所有问题 (One Size Cannot Fit All)
  - 2. Hadoop 1.0及其生态系统
  - 3. Hadoop 2.0及Spark

The End!