[前言: 3](#_Toc410241814)

[**一、Hadoop生态圈：** 3](#_Toc410241815)

[*Hadoop* 4](#_Toc410241816)

[*HBase* 5](#_Toc410241817)

[*Hive* 6](#_Toc410241818)

[*Apache Pig:* 6](#_Toc410241819)

[*Impala：* 7](#_Toc410241820)

[*Flume：* 7](#_Toc410241821)

[*Sqoop:* 8](#_Toc410241822)

[*Chukwa：* 8](#_Toc410241823)

[*Mahout:* 8](#_Toc410241824)

[*Hama:* 8](#_Toc410241825)

[*Giraph：* 9](#_Toc410241826)

[*Storm：* 9](#_Toc410241827)

[*ZooKeeper：* 9](#_Toc410241828)

[*Ambari:* 9](#_Toc410241829)

[*Oozie：* 9](#_Toc410241830)

[*Cloudera Hue：* 9](#_Toc410241831)

[**二、Spark生态圈：** 10](#_Toc410241832)

[*Spark：* 10](#_Toc410241833)

[*Spark SQL：* 11](#_Toc410241834)

[*Spark Streaming：* 12](#_Toc410241835)

[*MLLib：* 13](#_Toc410241836)

[*GraphX ：* 14](#_Toc410241837)

[*SparkR ：* 14](#_Toc410241838)

[*Tachyon：* 15](#_Toc410241839)

[*Mesos:* 16](#_Toc410241840)

[*Yarn:* 16](#_Toc410241841)

[*BlinkDB ：* 17](#_Toc410241842)

[**三、结构化数据生态圈：** 18](#_Toc410241843)

[*DBSync：* 18](#_Toc410241844)

[*OLAP* 19](#_Toc410241845)

[*HANA* 19](#_Toc410241846)

[**Spark与Hadoop的对比** 19](#_Toc410241847)

[**Spark与Hadoop的结合** 20](#_Toc410241848)

[**Spark的适用场景** 20](#_Toc410241849)

[**案例：** 20](#_Toc410241850)

**大数据分析系统架构之探讨**

### 前言:

对于大数据平台，本人也没实际实践过，所以，做为一个初学者的身份与大家探索这个问题，如有欠妥之处，请多多包涵！

首先，先让我们来看看大数据平台架构的集装箱里可有哪些零件。

**一、Hadoop生态圈：**

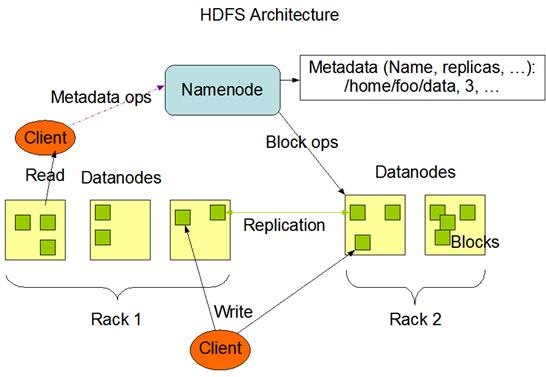
**数据计算平台：**

### Hadoop

Hadoop是Apache软件基金会所开发的并行计算框架与分布式文件系统。最核心的模块包括Hadoop Common、HDFS与MapReduce。HDFS是Hadoop分布式文件系统（Hadoop Distributed File Sys

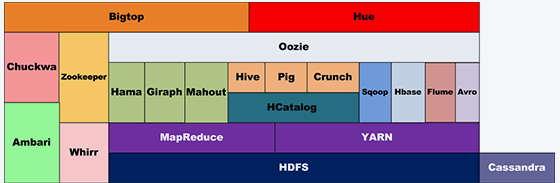
tem）的缩写，为分布式计算存储提供了底层支持。采用Java语言开发，可以部署在多种普通的廉价机器上，以集群处理数量积达到大型主机处理性能。HDFS采用master/slave架构。一个HDFS集群包含一个单独的NameNode和多个DataNode。NameNode作为master服务，它负责管理文件系统的命名空间和客户端对文件的访问。NameNode会保存文件系统的具体信息，包括文件信息、 文件被分割成具体block块的信息、以及每一个block块归属的DataNode的信息。对于整个集群来说，HDFS通过NameNode对用户提供 了一个单一的命名空间。DataNode作为slave服务，在集群中可以存在多个。通常每一个DataNode都对应于一个物理节点。DataNode负责管理节点上它们拥有 的存储，它将存储划分为多个block块，管理block块信息，同时周期性的将其所有的block块信息发送给NameNode。

  下图为HDFS系统架构图，主要有三个角色，Client、NameNode、DataNode。



在Hadoop的系统中，会有一台master，主要负责NameNode的工作以及JobTracker的工作。JobTracker的主要职责就是启 动、跟踪和调度各个Slave的任务执行。还会有多台slave，每一台slave通常具有DataNode的功能并负责TaskTracker的工作。 TaskTracker根据应用要求来结合本地数据执行Map任务以及Reduce任务。

MapReduce用于大规模数据集群分布式运算。任务的分解 （Map）与结果的汇总（Reduce）是其主要思想。Map就是将一个任务分解成多个任务，Reduce就是将分解后多任务分别处理，并将结果汇总为最 终结果。



**数据存储：**

### HBase

HBase是基于HDFS存储的一个分布式的、面向列的开源数据库。它是Apache Hadoop在HDFS基础上提供的一个类Bigatable。 是一个高[可靠](http://baike.baidu.com/view/190360.htm)性、高性能、面向列、可伸缩的[分布式存储系统](http://baike.baidu.com/view/1911305.htm)。可以这么理解，在 HDFS上，我们看到的是一些非结构，零散的文件数据，透过HBase可以将这些零散的、非结构文件数据结构化。从而可以进行一些高层次的操作，例如建表、增加、删除、更改、查找等，与传统的数据库不同的是HBase采用的是列式存储而不是行式存储。

其特点：

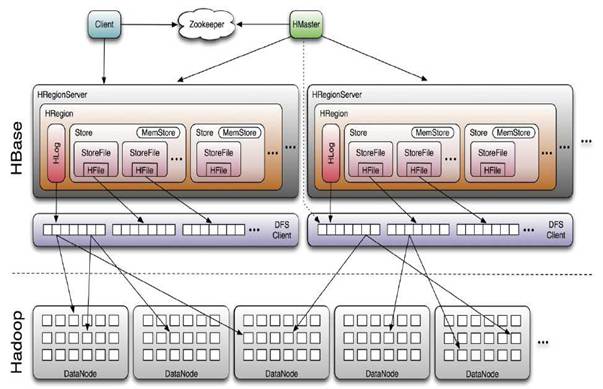
1 高可靠性

2 高效性

3 面向列

4 可伸缩

5 可在廉价PC Server搭建大规模结构化存储集群



**数据提取与分析：**

### Hive

Hive是建立在 Hadoop 上的数据仓库基础构架。它提供了一系列的工具，可以用来进行数据提取转化加载（ETL），这是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制。Hive 定义了简单的类 SQL 查询语言，称为 HQL，它允许熟悉 SQL 的用户查询数据。同时，这个语言也允许熟悉 MapReduce 开发者的开发自定义的 mapper 和 reducer 来处理内建的 mapper 和 reducer 无法完成的复杂的分析工作。

Hive 构建在基于静态批处理的Hadoop 之上，Hadoop 通常都有较高的延迟并且在作业提交和调度的时候需要大量的开销。因此，Hive 并不能够在大规模数据集上实现低延迟快速的查询，例如，Hive 在几百MB 的数据集上执行查询一般有分钟级的时间延迟。因此，

Hive 并不适合那些需要低延迟的应用，例如，联机事务处理（OLTP）。Hive 查询操作过程严格遵守Hadoop MapReduce 的作业执行模型，Hive 将用户的HiveQL 语句通过解释器转换为MapReduce 作业提交到Hadoop 集群上，Hadoop 监控作业执行过程，然后返回作业执行结果给用户。Hive 并非为联机事务处理而设计，Hive 并不提供实时的查询和基于行级的数据更新操作。Hive 的最佳使用场合是大数据集的批处理作业，例如，网络日志分析。

### [Apache Pig](http://pig.apache.org/):

是一个基于Hadoop的大规模数据分析工具，它提供的SQL-LIKE语言叫Pig Latin，该语言的编译器会把类SQL的数据分析请求转换为一系列经过优化处理的MapReduce运算。

### Impala：

Impala是Cloudera公司主导开发的新型查询系统，它提供SQL语义，能够查询存储在Hadoop的HDFS和HBase中的PB级大数据。已 有的Hive系统虽然也提供了SQL语义，但是由于Hive底层执行使用的是MapReduce引擎，仍然是一个批处理过程，难以满足查询的交互性；相比 之下，Impala的最大特点就是快速。

优点：

Impala的查询效率相比Hive，有数量级的提升。从技术角度上来看，Impala之所以能有好的性能，主要有如下几方面的原因：

 1） Impala不需要把中间结果写入磁盘，省掉了大量的I/O开销。

2） 省掉了MapReduce作业启动的开销。MapReduce启动task的速度是很慢的（默认每个心跳间隔是3秒钟），Impala直接通过相应的服务进程来进行作业调度，速度快了很多。

3） Impala完全抛弃了MapReduce这个不太适合做SQL查询的范式，而是像Dremel一样借鉴了MPP并行数据库的思想，从新另起炉灶，因此可以做更多的查询优化，从而能省掉不必要的shuffle，sort等开销；

4） 通过使用LLVM来统一编译运行时代码，避免了为支持通用编译而带来的不必要开销；

5） 用C++实现，做了很多有针对性的硬件优化，例如使用SSE指令；

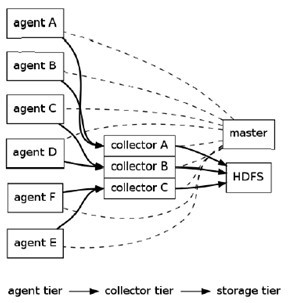
6） 使用了支持Data locality的I/O调度机制，尽可能的将数据和计算分配在同一台机器上进行，减少了网络开销；

**日志类收集工具：**

### Flume：

Flume是Cloudera提供的一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统，Flume支持在日志系统中定制各类数据发送方，用于收集数据；同时，Flume提供对数据进行简单处理，并写到各种数据接受方（可定制）的能力。

flume的逻辑架构：

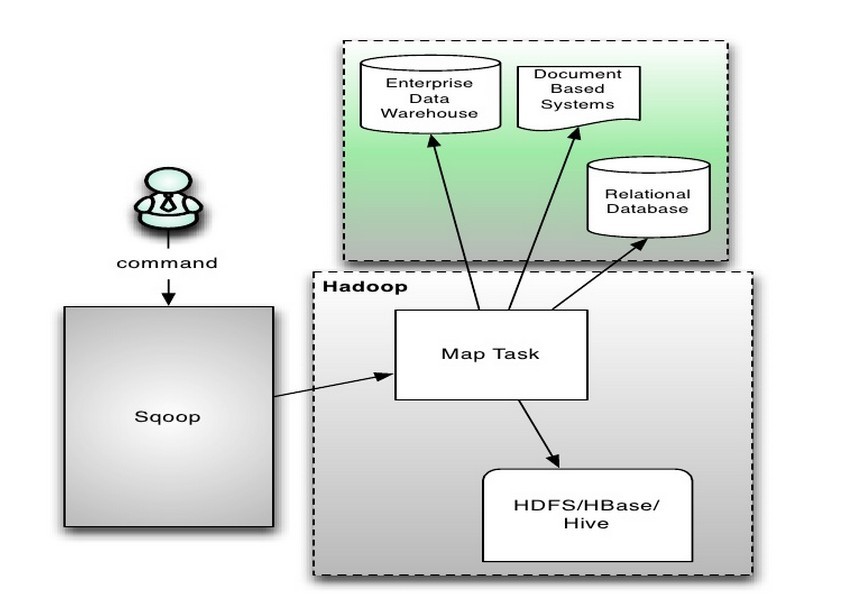


其中，storage是存储系统，可以是一个普通file，也可以是HDFS，HIVE，HBase，分布式存储等。

### Sqoop:

Sqoop是一个用来将Hadoop和关系型[数据库](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D6842%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%CA%FD%BE%DD%BF%E2&k0=spark&kdi0=8&k1=sql&kdi1=8&k2=%CA%FD%BE%DD%BF%E2&kdi2=8&k3=%CA%FD%BE%DD%B4%E6%B4%A2&kdi3=8&k4=twitter&kdi4=8&sid=d2b63361b043e59f&ch=0&tu=u1692056&jk=22a86577588c1a38&cf=29&fv=15&stid=9&urlid=0&luki=3&seller_id=1&di=128)中的数据相互转移的工具，可以将一个关系型数据库中数据导入Hadoop的HDFS中，也可以将HDFS中数据导入关系型数据库中。

sqoop架构：



### Chukwa：

Chukwa是一个开源的用于监控大型分布式系统的数据收集系统，它可以将各种各样类型的数据收集成适合 Hadoop 处理的文件保存在 HDFS 中供 Hadoop 进行各种 MapReduce 操作。

数据计算：

### Mahout:

[Apache Mahout](http://mahout.apache.org/)是基于Hadoop的机器学习和数据挖掘的一个分布式框架。Mahout用MapReduce实现了部分数据挖掘算法，解决了并行挖掘的问题.

### Hama:

是一个基于HDFS的BSP（Bulk Synchronous Parallel)并行计算框架, Hama可用于包括图、矩阵和网络算法在内的大规模、大数据计算。

### Giraph：

Giraph是一个可伸缩的分布式迭代图处理系统， 基于Hadoop平台，灵感来自 BSP (bulk synchronous parallel) 和 Google 的 Pregel。

**实时流数据处理：**

### Storm：

Storm是一个基于内存的实时流处理系统。适合于大批量小型数据的处理，实时性较好，基本上是毫秒级级别。

**资源管理与调度：**

### ZooKeeper：

Zookeeper是Google的Chubby一个开源的实现。它是一个针对大型分布式系统的可靠协调系统，提供的功能包括：配置维护、名字服务、 分布式同步、组服务等。ZooKeeper的目标就是封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

### Ambari:

[Ambari](http://incubator.apache.org/ambari/)是一种基于Web的工具，支持Hadoop集群的供应、管理和监控。

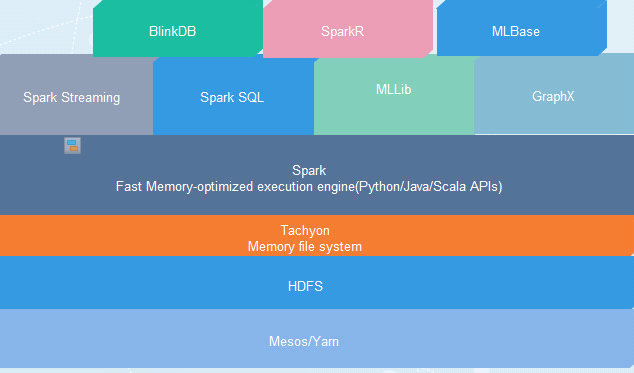
### Oozie：

Oozie: 是一个工作流引擎服务器, 用于管理和协调运行在Hadoop平台上（HDFS、Pig和MapReduce）的任务。

### [Cloudera Hue](http://cloudera.github.io/hue/)：

[Cloudera Hue](http://cloudera.github.io/hue/)是一个基于WEB的监控和管理系统，实现对HDFS，MapReduce/YARN, HBase, Hive, Pig的web化操作和管理。

**二、Spark生态圈：**



### Spark：

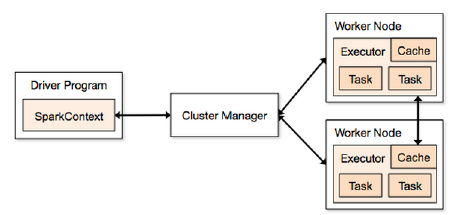
Spark是基于内存分布式的计算框架。spark立足于内存计算，从多迭代批量处理出发，兼收并蓄数据仓库、流处理和图计算等多种计算范式，是罕见的全能选手。

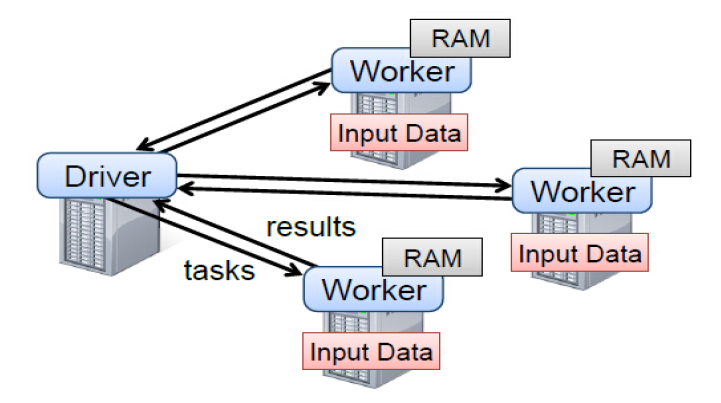
Spark 启用了内存分布数据集，除了能够提供交互式查询外，它还可以优化迭代工作负载。Spark 是在 Scala 语言中实现的，它将 Scala 用作其应用程序框架，而Scala的语言特点也铸就了大部分Spark的成功。与 Hadoop 不同，Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据集。尽管创建 Spark 是为了支持分布式数据集上的迭代作业，但是实际上它是对 Hadoop 的补充，可以在 Hadoop文件系统中并行运行。通过名为 Mesos 的第三方集群框架可以支持此行为。

虽然 Spark 与 Hadoop 有相似之处，但它提供了具有有用差异的一个新的集群计算框架。首先，Spark 是为集群计算中的特定类型的工作负载而设计，即那些在并行操作之间重用工作数据集（比如机器学习算法）的工作负载。为了优化这些类型的工作负 载，Spark 引进了内存集群计算的概念，可在内存集群计算中将数据集缓存在内存中，以缩短访问延迟。

Spark 还引进了名为 弹性分布式数据集 (RDD) 的抽象。RDD 是分布在一组节点中的只读对象集合。这些集合是弹性的，如果数据集一部分丢失，则可以对它们进行重建。重建部分数据集的过程依赖于容错机制，该机制可以维 护 “血统”（即充许基于数据衍生过程重建部分数据集的信息）。RDD 被表示为一个 Scala 对象，并且可以从文件中创建它；一个并行化的切片（遍布于节点之间）；另一个 RDD 的转换形式；并且最终会彻底改变现有 RDD 的持久性，比如请求缓存在内存中。

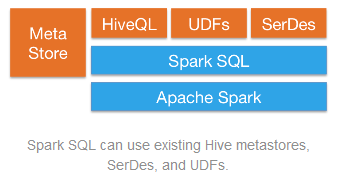
Spark 中的应用程序称为驱动程序，这些驱动程序可实现在单一节点上执行的操作或在一组节点上并行执行的操作。与 Hadoop 类似，Spark 支持单节点集群或多节点集群。对于多节点操作，Spark 依赖于 Mesos 集群管理器。Mesos 为分布式应用程序的资源共享和隔离提供了一个有效平台。该设置充许 Spark 与 Hadoop 共存于节点的一个共享池中。





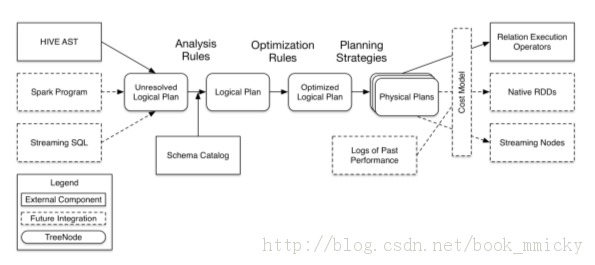
### Spark SQL：

Spark SQL允许在Spark中执行使用SQL，HiveQL或Scala表示的关系型查询，其中的核心组件SchemaRDD，SchemaRDDs由行对象以及用来描述每行中各列数据类型的模式组成，每个SchemaRDD类似于关系数据库中的一个表。



Spark SQL的特点:

* 引入了新的RDD类型SchemaRDD，可以象传统数据库定义表一样来定义SchemaRDD，SchemaRDD由定义了列数据类型的行对象构成。
* SchemaRDD可以从RDD转换过来，也可以从Parquet文件读入，也可以使用HiveQL从Hive中获取。
* 在应用程序中可以混合使用不同来源的数据，如可以将来自HiveQL的数据和来自SQL的数据进行join操作。
* 内嵌catalyst优化器对用户查询语句进行自动优化



### Spark Streaming：

Spark Streaming是Spark核心API的一种扩展，它实现了对实时流数据的高吞吐量，低容错率的流处理。数据可以有许多来源，如Kafka，Flume，Twitter，ZeroMQ或传统的TCP Socket，可以使用复杂算法对其处理实现高层次的功能，如map,reduce,join和window.



SparkStreaming流式处理系统特点有：

* 将流式计算分解成一系列短小的批处理作业
* 将失败或者执行较慢的任务在其它节点上并行执行
* 较强的容错能力(基于RDD继承关系Lineage)
* 使用和RDD一样的语义

### MLLib：

MLLib是Spark机器学习算法库，由通用的机器学习算法和工具组成，包括分类，回归，聚类，协同过滤，降维以及底层的优化组件等。

分类和回归：

* 线性模型（支持向量机，逻辑回归，线性回归）
* 决策树
* 朴素贝叶斯

协同过滤：

* 交替最小二乘法（ALS）

聚类：

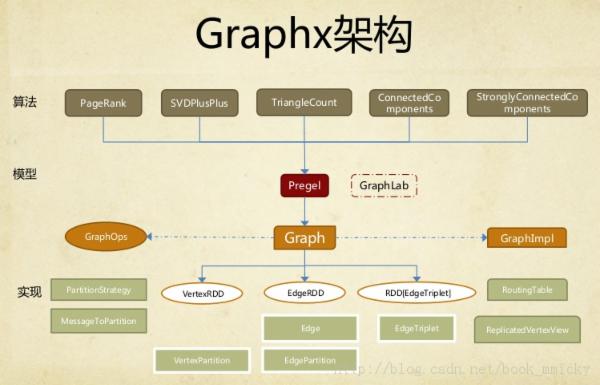
* K-means聚类

降维

* 奇异值分解（SVD）
* 主成分分析（PCA）

### GraphX ：

  GraphX是基于Spark的图处理和图并行计算API。GraphX定义了一个新的概念：弹性分布式属性图，一个每个顶点和边都带有属性的定向多重 图；并引入了三种核心RDD：Vertices、Edges、Triplets；还开放了一组基本操作（如subgraph, joinVertices, and mapReduceTriplets），并且在不断的扩展图形算法和图形构建工具来简化图分析工作。

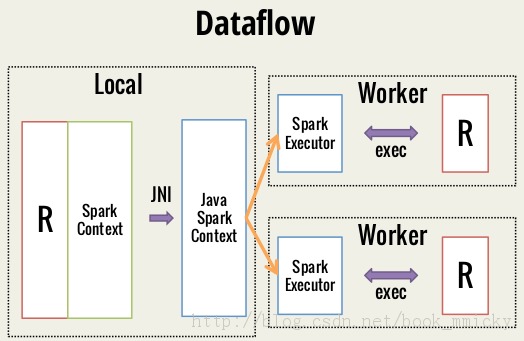


### SparkR ：

   SparkR是AMPLab发布的一个R开发包，使得R摆脱单机运行的命运，可以作为Spark的job运行 在集群上，极大得扩展了R的数据处理能力。

      Spark的几个特性：

* 提供了Spark中弹性分布式数据集（RDD）的API，用户可以在集群上通过R shell交互性的运行Spark job。
* 支持序化闭包功能，可以将用户定义函数中所引用到的变量自动序化发送到集群中其他的机器上。
* SparkR还可以很容易地调用R开发包，只需要在集群上执行操作前用includePackage读取R开发包就可以了，当然集群上要安装R开发包。

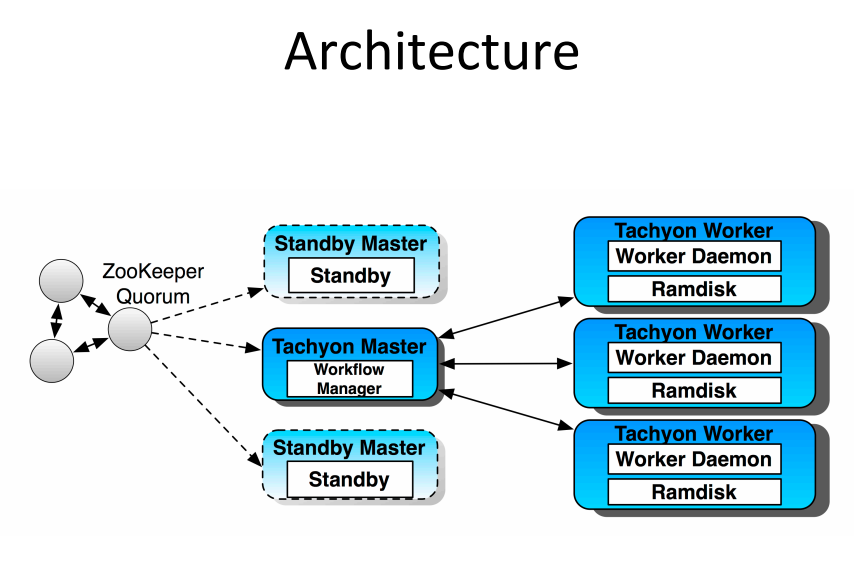


### Tachyon：

**Tachyon**是一个分布式内存文件系统，可以在集群里以访问内存的速度来访问存在tachyon里的文件。把 Tachyon是架构在最底层的分布式文件存储和上层的各种计算框架之间的一种中间件。主要职责是将那些不需要落地到DFS里的文件，落地到分布式内存文 件系统中，来达到共享内存，从而提高效率。同时可以减少内存冗余，GC时间等。

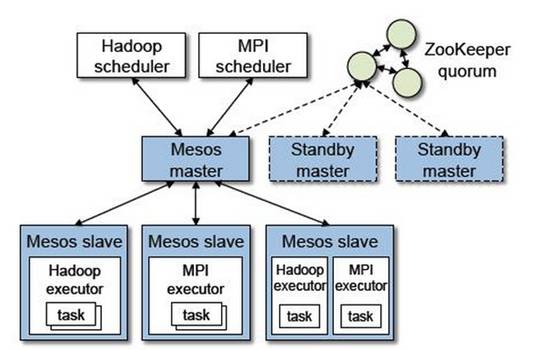
**Tachyon架构**

       Tachyon的架构是传统的Master—slave架构，这里和Hadoop类似，TachyonMaster里WorkflowManager是 Master进程，因为是为了防止单点问题，通过Zookeeper做了HA，可以部署多台Standby Master。Slave是由Worker Daemon和Ramdisk构成。这里个人理解只有Worker Daemon是基于JVM的，Ramdisk是一个off heap memory。Master和Worker直接的通讯协议是Thrift。



### Mesos:

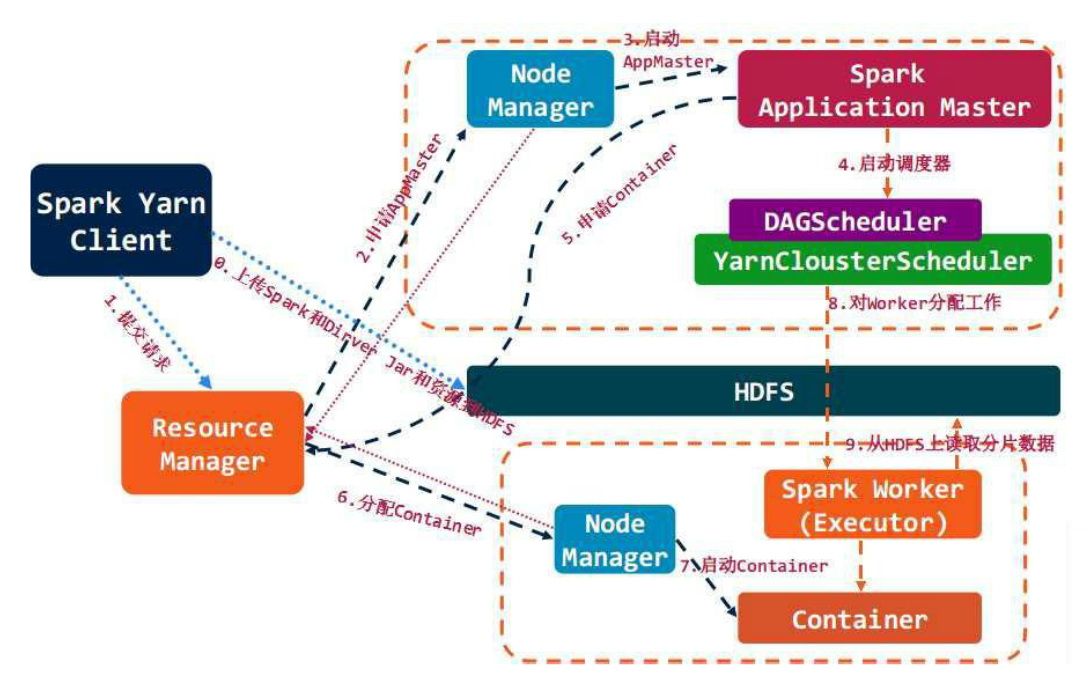
Mesos master是一个分布式集群资源调度器，采用某种策略将某个slave上的空闲资源分配给某一个framework，各种framework通过自己的 调度器向Mesos master注册，以接入到Mesos中；而Mesos slave主要功能是汇报任务的状态和启动各个framework的executor（比如Hadoop的excutor就是TaskTracker）。



### Yarn:

Yarn是一个实现分布式集群资源管理和调度的框架。

Yarn调度器根据容量，队列等限制条件（如每个队列分配一定的资源，最多执行一定数量的作业等），将系统中的资源分配给各个正在运行的应用。这里的调度器是一个 “纯调度器”，因为它不再负责监控或者跟踪应用的执行状态等，此外，他也不负责重新启动因应用执行失败或者硬件故障而产生的失败任务。调度器仅根据各个应 用的资源需求进行调度，这是通过抽象概念“资源容器”完成的，资源容器（Resource Container）将内存，CPU，磁盘，网络等资源封装在一起，从而限定每个任务使用的资源量。

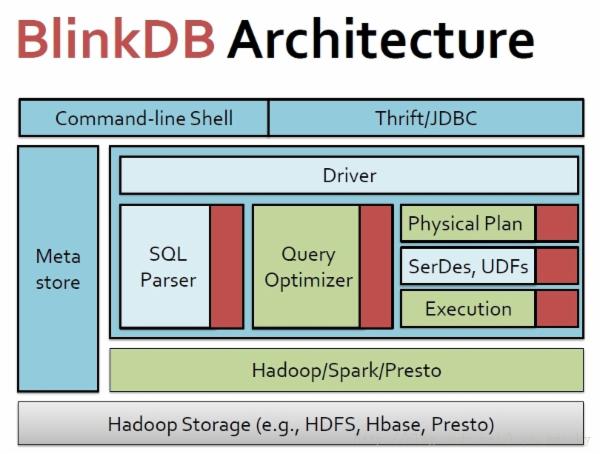


### BlinkDB ：

  BlinkDB是一个很有意思的交互式查询系统，就像一个跷跷板，用户需要在查询精度和查询时间上做一权衡；如果用户想更快地获取查询结果，那么将牺牲查 询结果的精度；同样的，用户如果想获取更高精度的查询结果，就需要牺牲查询响应时间。用户可以在查询的时候定义一个失误边界。

 BlinkDB的设计核心思想：

* 通过采样，建立并维护一组多维度样本
* 查询进来时，选择合适的样本来运行查询



**三、结构化数据生态圈：**

* Oracle
* Greenplum
* Mysql
* MongoDB
* SQL Server

数据同步：

### DBSync：

DBSync数据库同步备份工具是一款[异构数据库](http://baike.baidu.com/view/78075.htm)之 间同步的工具，支持市面上大多数主流数据库,主要有：SqlServer、ORACLE、 DB2、Sybase,Access，该软件提供的ODBC的同步功能，可以间接实现对MYSQL、SYBASE、INTERBASE等其他数据库的支 持。DBSync可以实现计划、增量、两表记录一致等方式的同步操作，利用该软件，可以实现企业内部应用系统数据的互通互联。该软件性能稳定，能提供 7\*24小时不间断同步的支持，具备单表千万级记录甚至更多记录的同步能力。该软件的专业版提供局域网或企业内部网之间数据库的同步（数据库都具备独立的 IP);企业版则提供集团在世界范围内的各分支机构的数据库同步。

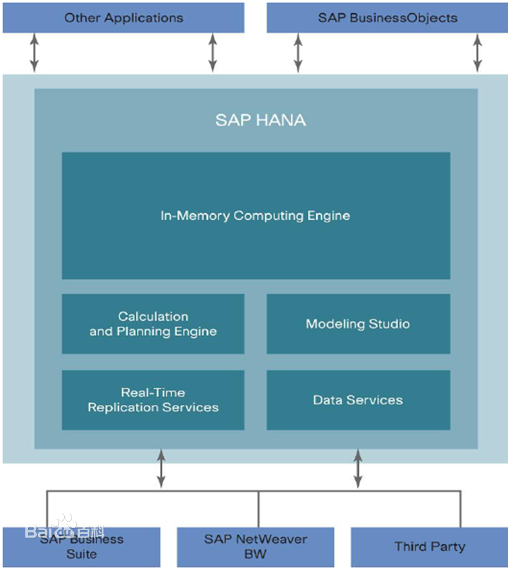
**数据分析处理：**

### OLAP

联机分析处理（OLAP）系统是[数据仓库](http://baike.baidu.com/view/19711.htm)系 统最主要的应用，专门设计用于支持复杂的分析操作，侧重对决策人员和高层管理人员的决策支持，可以根据分析人员的要求快速、灵活地进行大数据量的复杂查询 处理，并且以一种直观而易懂的形式将查询结果提供给决策人员，以便他们准确掌握企业（公司）的经营状况，了解对象的需求，制定正确的方案。

### HANA

HANA是一个软硬件结合体，提供高性能的数据查询功能，用户可以直接对大量实时业务数据进行查询和分析，而不需要对业务数据进行建模、聚合等。



对于作为大数据计算的两个主流平台Hadoop与Spark，其实在比较hadoop与spark之前，心里还是有所偏向的，做为spark的粉丝，很想多为spark美言几句，但由于spark还是个新生事物，成功以及大集群的案例还不是很多，所以，我们只能抱着中立的态度来学习spark与hadoop。

**Spark与Hadoop的对比**

* Spark的中间数据放到内存中，对于迭代运算效率更高。
  + Spark更适合于迭代运算比较多的ML和DM运算。因为在Spark里面，有RDD的抽象概念。
* Spark比Hadoop更通用。
  + Spark 提供的数据集操作类型有很多种，不像Hadoop只提供了Map和Reduce两种操作。比如 map, filter, flatMap, sample, groupByKey, reduceByKey, union, join, cogroup, mapValues,sort,partionBy 等多种操作类型，Spark把这些操作称为Transformations。同时还提供 Count,collect, reduce, lookup, save等多种actions操作。
  + 这些多种多样的数据集操作类型，给给开发上层应用的用户提供了方便。各个处理节点之间的通信模型不再像Hadoop那样就是唯一的Data Shuffle一种模式。用户可以命名，物化，控制中间结果的存储、分区等。可以说编程模型比Hadoop更灵活。
  + 不过由于RDD的特性，Spark不适用那种异步细粒度更新状态的应用，例如web服务的存储或者是增量的web爬虫和索引。就是对于那种增量修改的应用模型不适合。
* 容错性。
  + 在分布式数据集计算时通过checkpoint来实现容错，而checkpoint有两种方式，一个是checkpoint data，一个是logging the updates。用户可以控制采用哪种方式来实现容错。
* 可用性。
  + Spark通过提供丰富的Scala, Java，Python API及交互式Shell来提高可用性。

**Spark与Hadoop的结合**

* Spark可以直接对HDFS进行数据的读写，同样支持Spark on YARN。Spark可以与MapReduce运行于同集群中，共享存储资源与计算，数据仓库Shark实现上借用Hive，几乎与Hive完全兼容。

**Spark的适用场景**

* Spark是基于内存的迭代计算框架，适用于需要多次操作特定数据集的应用场合。需要反复操作的次数越多，所需读取的数据量越大，受益越大，数据量小但是计算密集度较大的场合，受益就相对较小
* 由于RDD的特性，Spark不适用那种异步细粒度更新状态的应用，例如web服务的存储或者是增量的web爬虫和索引。就是对于那种增量修改的应用模型不适合。
* 总的来说Spark的适用面比较广泛且比较通用。

**案例：**

**大数据解决的命题：**

从各式各样的数据源中找出一种提取与计算数据并最终得出有效数据的解决方案。

特点：

1. 数据源的多样化
2. 计算平台的分布式化
3. 离线与实时综合化

**数据源种类：**

1、结构化数据：

数据库数据

2、非结构化数据：

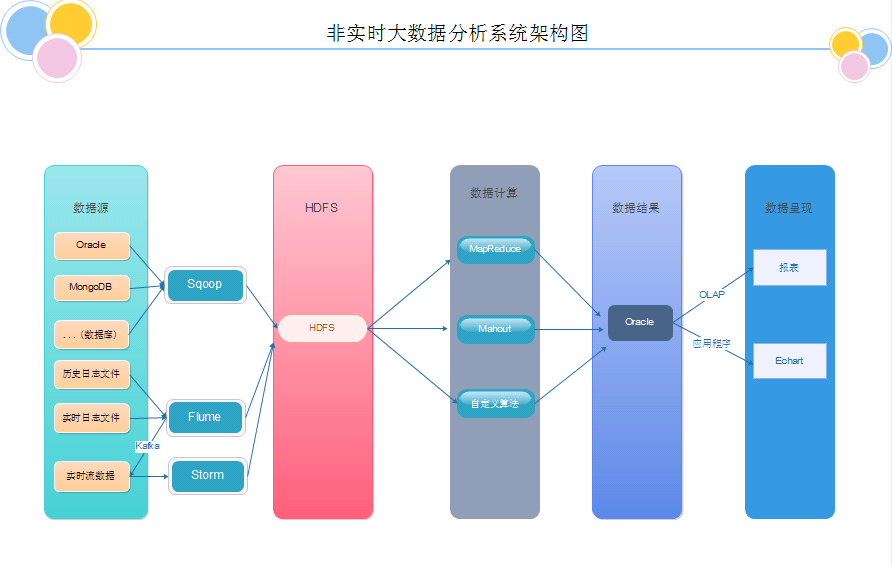
文本，视频，音频，图片，邮件，传感器，GPS

3、流式数据

实时流数据

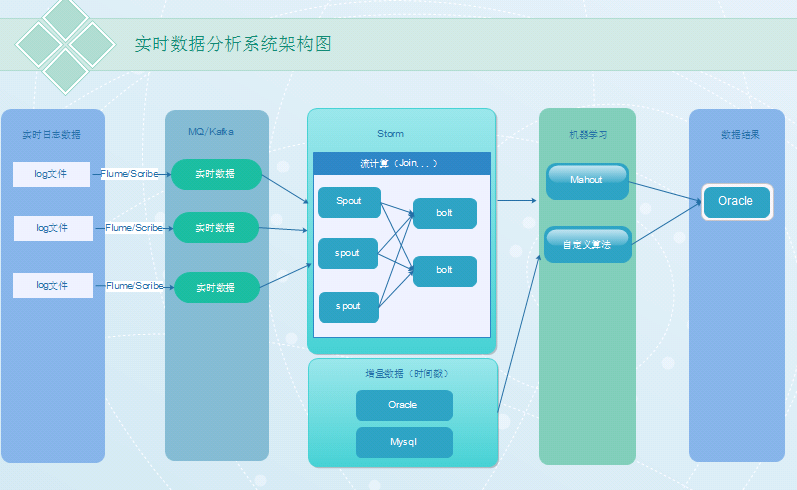
在讨论Hadoop系统架构之前，首先，我们应该明确一件事，Hadoop是做什么用的，简单的说，Hadoop就是一个基于大量数据进行计算的一个平台，它的计算的核心就是MapReduce，也就是去繁从简，剔除无用数据把有用的留下。HDFS是它的存储组件，也是它计算最直接的数据源。我们的目的就是，首先，将原始的数据，如数据库中数据或日志文件数据提取并按一定格式写入Hadoop计算的数据中心——HDFS，然后再通过一系列的计算算法，计算出有用的数据，再将小量的结果数据存入到结果数据库中。

**方案一：**



对于该架构，只适合于较为传统的数据处理，着重考虑的是将各种数据，如关系型数据库的数据，NoSQL数据库的数据，Log形式的数据统一到一种格式并存入分布式HDFS上，然后通过一系列的机器算法，计算出想要的结果。其中，暂时未考虑像网页，图像，音频，视频等类型的数据源，对于其他类型的数据，也许还需要在前端做一些处理，将它们转换成可分析的数据。另外，该架构，对数据处理的实效性也暂未考虑，虽引入了实时计算框架Storm，但这里仅将其作为实时流处理的一个中间平台，其结果存入HDFS后参与后面的计算，因此，其实时性也是不高的。如果对数据实时性要求较高的话，可以将storm等独立出来，不参与全局的计算，只作为独立的模块，读取数据-->计算数据-->存入数据.结构如下：

方案二：



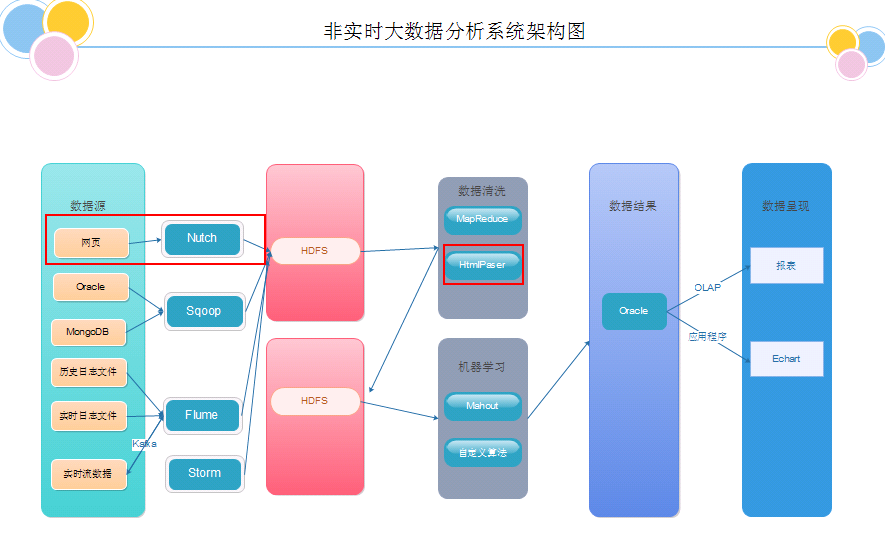
以上架构思想：只针对那些实时的数据，其中，Storm结合Kafka处理实时的日志文件数据，同时，对于数据库的数据源，并不提取全量数据，而只针对那些增量的数据，该架构适合于那些动态性，实时性较强的数据，例如：实时路况分析。

当然，对于一个分析系统，更多时候是历史数据和实时数据的统一，那么可以结合以上的两种架构，前种处理全量的，历史的离线数据，后种处理增量的，实时的线上数据。

下面让我们对数据源展开看看，对于传统的数据源，也许以上架构就已满足，但在当今的网络时代，对于一个普普通通的网站，就已经不仅仅局限于数据库以及日志这两种数据类型了。网页，图片，音频，视频等都是网络中必不可少的组成元素，那么大数据时代，也当然少不了对这些元素进行解读。

1、网页数据。

对于网页爬虫框架，Nutch也许是不错的选择，通过Nutch对网站的网页进行爬取，再结合HtmlParser和MapReduce对网页数据进行分析提取。然后再将结果存入HDFS，以备后续的机器学习所用。新的架构图如下：



2、图像数据：

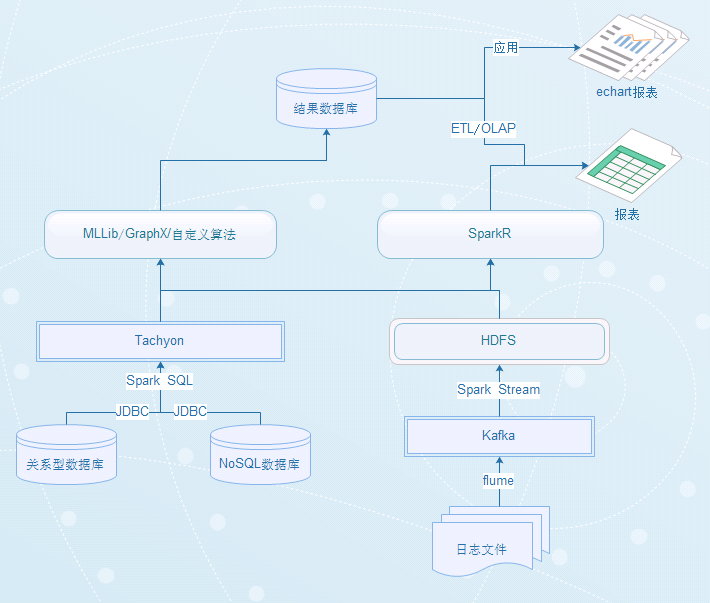
3、音频数据：

4、视频数据：

对于以上三种数据的分析涉及到一些较为前沿和高深的技术，如图像识别，语言识别，视频处理，每一个模块也许就是一个大型的系统，由于本人才疏学浅，暂时略过，不做展开。

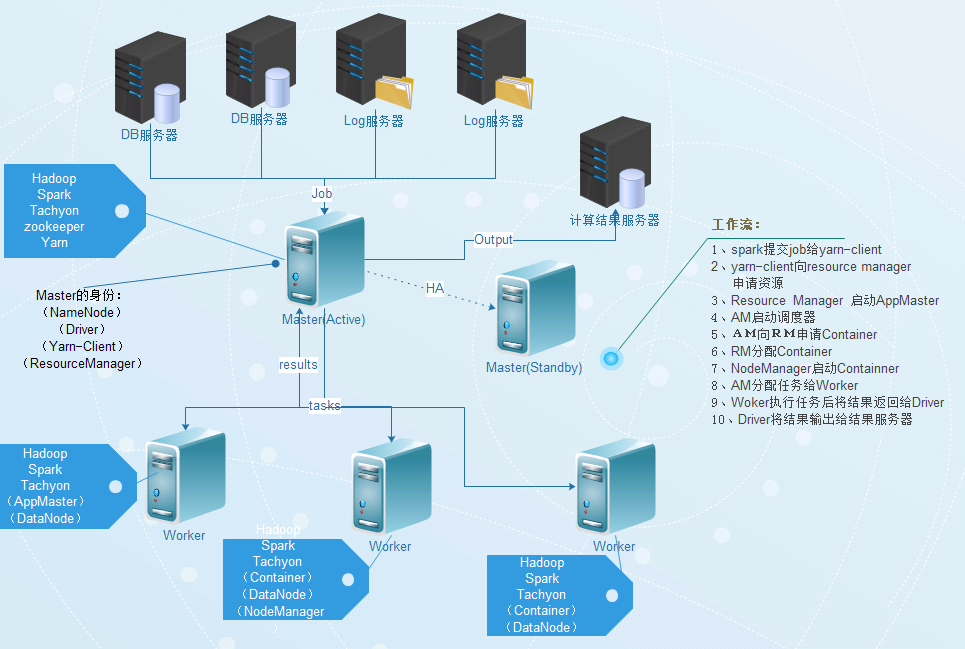
下面让我们重磅推出我们的全能选手Spark，让我们来看看spark在大数据中的应用。

先让我们来看这样一张图：



同样的，对于数据源，同样有关系型数据库，非关系型数据库，以及日志文件等。对于数据库的数据，由于已经有规范的物理存储，我们就无需将其再次存入HDFS，可以考虑在spark与数据库间加入一层内存存储——Tachyon,先将参与计算的数据提到分布式的内存中，这样提高计算的效率。而对于日志文件，由于其本身为非结构化的数据，可能需要经过一定的加工处理，因此，有必要先将其统一结构化，所以可以考虑将结构化后的结果先存入HDFS，再参与计算。另外，考虑到业务需求，有的系统可能需要将结果存入到物理的存储空间，有的可能希望直接得到例如报表的结果，所以这里分别针对这两种需求给出方案，一、将Tachyon与HDFS上的数据通过机器学习后存入数据库，以便后续加工处理；二、引入SparkR统计分析工具，通过SparkR结合R算法、MLLib算法对数据进行学习，分析，计算后直接得出结果。

网络架构图如下：



【后话】

以上是我的大数据分析系统架构一点初浅的理解，也许有不成熟和欠妥的地方，本文只做为个人的一点心得体会的分享，欢迎指正与建议。

参考：

<http://www.linuxidc.com/Linux/2013-08/88606.htm>

<http://www.open-open.com/lib/view/open1416213856742.html>

<http://www.aboutyun.com/thread-6842-1-1.html>

<http://www.open-open.com/lib/view/open1384084364227.html>

<http://blog.fens.me/hadoop-family-roadmap/>

<http://blog.csdn.net/zhouleilei/article/details/8568147>

<http://blog.csdn.net/yfkiss/article/details/8700480>

<http://www.aboutyun.com/thread-6855-1-1.html>

<http://www.open-open.com/lib/view/open1409754088791.html>

<http://www.open-open.com/lib/view/open1402227705525.html>

<http://sofar.blog.51cto.com/353572/1362240>