## 1 原理

Apache Spark是一个新兴的大数据处理的引擎，主要特点是提供了一个集群的分布式内存抽象，以支持需要工作集的应用。

弹性分布式数据集RDD是Apache Spark的核心理念。它是由数据组成的不可变分布式集合，其主要进行两个操作：transformation和action。Transformation是类似在RDD上做 filter()、map()或union() 以生成另一个RDD的操作，而action则是count()、first()、take(n)、collect() 等促发一个计算并返回值到Master或者稳定存储系统的操作。Transformations一般都是lazy的，直到action执行后才会被执行。Spark Master/Driver会保存RDD上的Transformations。这样一来，如果某个RDD丢失(也就是salves宕掉)，它可以快速和便捷地转换到集群中存活的主机上。这也就是RDD的弹性所在。

尽管MapReduce适用大多数批处理工作，并且在大数据时代成为企业大数据处理的首选技术，但由于以下几个限制，它对一些场景并不是最优选择：

缺少对迭代计算以及DAG运算的支持

Shuffle过程多次排序和落地，MR之间的数据需要落地Hdfs文件系统

Spark在很多方面都弥补了MapReduce的不足，比MapReduce的通用性更好，迭代运算效率更高，作业延迟更低，它的主要优势包括：

提供了一套支持DAG图的分布式并行计算的编程框架，减少多次计算之间中间结果写到Hdfs的开销

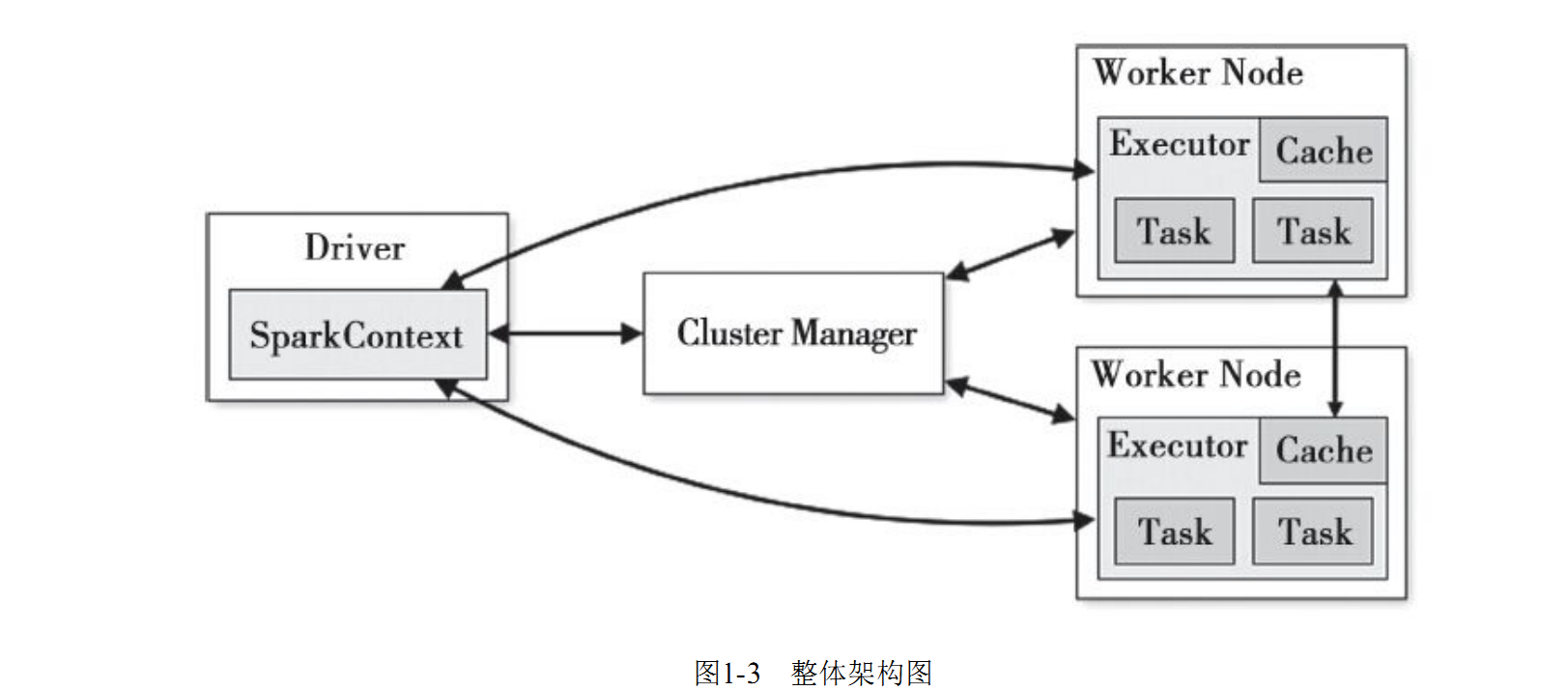
提供Cache机制来支持需要反复迭代计算或者多次数据共享，减少数据读取的IO开销

使用多线程池模型来减少task启动开稍，shuffle过程中避免不必要的sort操作以及减少磁盘IO操作

广泛的数据集操作类型

MapReduce由于其设计上的约束只适合处理离线计算，在实时查询和迭代计算上仍有较大的不足，而随着业务的发展，业界对实时查询和迭代分析有更多的需求，单纯依靠MapReduce框架已经不能满足业务的需求了。Spark由于其可伸缩、基于内存计算等特点，且可以直接读写**[Hadoop](http://lib.csdn.net/base/hadoop" \t "_blank" \o "Hadoop知识库)**上任何格式的数据，成为满足业务需求的最佳候选者。

Spark运行框架如下图所示，首先有集群资源管理服务(Cluster Manager)和运行作业任务的结点(Worker Node)，然后就是每个应用的任务控制结点Driver和每个机器节点上有具体任务的执行进程(Executor)。



与MR计算框架相比，Executor有二个优点：一个是多线程来执行具体的任务，而不是像MR那样采用进程模型，减少了任务的启动开稍。二个是Executor上会有一个BlockManager存储模块，类似于KV系统(内存和磁盘共同作为存储设备)，当需要迭代多轮时，可以将中间过程的数据先放到这个存储系统上，下次需要时直接读该存储上数据，而不需要读写到hdfs等相关的文件系统里，或者在交互式查询场景下，事先将表Cache到该存储系统上，提高读写IO性能。另外Spark在做Shuffle时，在Groupby，Join等场景下去掉了不必要的Sort操作，相比于MapReduce只有Map和Reduce二种模式，Spark还提供了更加丰富全面的运算操作如filter,groupby,join等。

Spark的Application在运行时，首先在Driver程序中创建SparkContext，将其作为调度的总入口，在其初始化的过程中会分别创建DAGSchedule(进行Stage调度)和TaskSchedule(进行Task调度)两个模块。DAGSchedule模块是基于Stage的调度模块，它为每个Spark job计算具有依赖关系的多个Stage任务阶段然后将每个Stage划分为一组具体的任务以TaskSet的

形式提交给底层的TaskSchedule模块来具体执行。TaskSchedule负责具体启动任务，监控和汇报任务的运行情况。而任务运行所需的资源需要向Cluster Manager申请。

Spark体系架构包括如下三个主要组件：

* 数据存储
* API
* 管理框架

Spark生态系统

除了Spark核心API之外，Spark生态系统中还包括其他附加库，可以在大数据分析和机器学习领域提供更多的能力。

这些库包括：

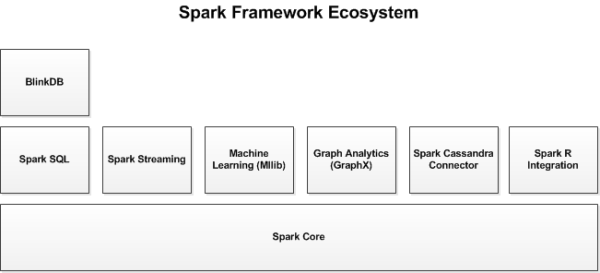
* **Spark Streaming:**
  + [Spark Streaming](https://spark.apache.org/streaming/)基于微批量方式的计算和处理，可以用于处理实时的流数据。它使用DStream，简单来说就是一个弹性分布式数据集（RDD）系列，处理实时数据。
* **Spark SQL:**
  + [Spark SQL](https://spark.apache.org/sql/)可以通过JDBC API将Spark数据集暴露出去，而且还可以用传统的BI和可视化工具在Spark数据上执行类似SQL的查询。用户还可以用Spark SQL对不同格式的数据（如JSON，Parquet以及数据库等）执行ETL，将其转化，然后暴露给特定的查询。
* **Spark MLlib:** 
  + [MLlib](https://spark.apache.org/mllib/)是一个可扩展的Spark机器学习库，由通用的学习算法和工具组成，包括二元分类、线性回归、聚类、协同过滤、梯度下降以及底层优化原语。
* **Spark GraphX:** 
  + [GraphX](https://spark.apache.org/graphx/)是用于图计算和并行图计算的新的（alpha）Spark API。通过引入弹性分布式属性图（Resilient Distributed Property Graph），一种顶点和边都带有属性的有向多重图，扩展了Spark RDD。为了支持图计算，GraphX暴露了一个基础操作符集合（如subgraph，joinVertices和aggregateMessages）和一个经过优化的Pregel API变体。此外，GraphX还包括一个持续增长的用于简化图分析任务的图算法和构建器集合。

除了这些库以外，还有一些其他的库，如BlinkDB和Tachyon。

[BlinkDB](http://blinkdb.org/)是一个近似查询引擎，用于在海量数据上执行交互式SQL查询。BlinkDB可以通过牺牲数据精度来提升查询响应时间。通过在数据样本上执行查询并展示包含有意义的错误线注解的结果，操作大数据集合。

[Tachyon](http://tachyon-project.org/index.html)是一个以内存为中心的分布式文件系统，能够提供内存级别速度的跨集群框架（如Spark和MapReduce）的可信文件共享。它将工作集文件缓存在内存中，从而避免到磁盘中加载需要经常读取的数据集。通过这一机制，不同的作业/查询和框架可以以内存级的速度访问缓存的文件。  
此外，还有一些用于与其他产品集成的适配器，如Cassandra（[Spark Cassandra 连接器](http://www.datastax.com/dev/blog/accessing-cassandra-from-spark-in-java)）和R（SparkR）。Cassandra Connector可用于访问存储在Cassandra数据库中的数据并在这些数据上执行数据分析。

下图展示了在Spark生态系统中，这些不同的库之间的相互关联。



**图1. Spark框架中的库**

## 2 业务场景

目前spark在互联网公司主要应用在广告、报表、推荐系统等业务上。在广告业务方面需要大数据做应用分析、效果分析、定向优化等，在推荐系统方面则需要大数据优化相关排名、个性化推荐以及热点点击分析等。

这些应用场景的普遍特点是计算量大、效率要求高。

## 3运行环境配置

硬件环境：Centos 6.5 服务器4台（一台为Master节点，三台为Slave节点）   
软件环境：jdk-1.7、hadoop-2.6 spark-1.6

## 4开发环境配置：

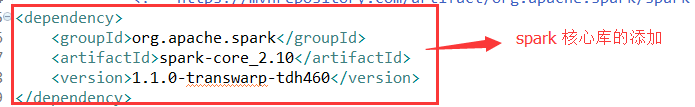
### 一： 安装JDK与eclipse

### 二 构建spark maven工程

#### 3.1 新建java工程



#### 3.2 引用相关jar包

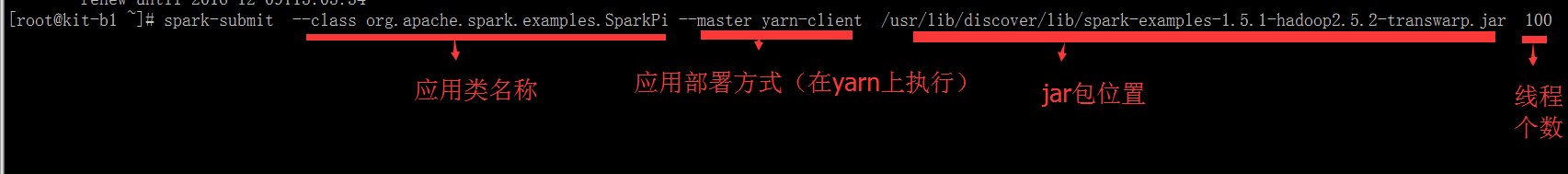


## 5 代码演示



## 6部署运行

spark-submit --class org.apache.spark.examples.SparkPi --master yarn-client /usr/lib/discover/lib/spark-examples-1.5.1-hadoop2.5.2-transwarp.jar 100



## 7结果查看

