Spark 学习笔记

harli

2014/12/9

shanghai-pd

[Scala语言 4](#_Toc405898066)

[Spark平台的API 4](#_Toc405898067)

[Spark内核 4](#_Toc405898068)

[Spark上的核心框架 4](#_Toc405898069)

[商业级别的Spark项目 4](#_Toc405898070)

[Spark解决方案 4](#_Toc405898071)

[Spark内核 4](#_Toc405898072)

[Spark核心框架 5](#_Toc405898073)

[Spark Streaming 5](#_Toc405898074)

[Spark SQL 5](#_Toc405898075)

[MLlib(Machine Learning library) 5](#_Toc405898076)

[GraphX(Graph Processing) 5](#_Toc405898077)

[Bagel (Pregel on Spark) 5](#_Toc405898078)

[SparkR 5](#_Toc405898079)

[Spark开发与应用 5](#_Toc405898080)

[部署 5](#_Toc405898081)

[提交应用程序 5](#_Toc405898082)

[Amazon EC2 5](#_Toc405898083)

[Standalone模式 6](#_Toc405898084)

[Mesos 6](#_Toc405898085)

[YARN 6](#_Toc405898086)

[监控 6](#_Toc405898087)

[配置 6](#_Toc405898088)

[调优 6](#_Toc405898089)

[作业调度 6](#_Toc405898090)

[安全 6](#_Toc405898091)

[硬件配置 6](#_Toc405898092)

[技巧、FQA 6](#_Toc405898093)

# Spark学习阶段篇

<http://book.51cto.com/art/201409/451443.htm>

Spark 的学习阶段篇包含六大部分，从Scala语言到spark内核、框架，以及spark在商业上的企业级实践等。

# Scala语言

1，Spark框架是采用Scala语言编写的，精致而优雅。要想成为Spark高手，你就必须阅读Spark的源代码，就必须掌握Scala,；

2，虽然说现在的Spark可以采用多语言Java、Python等进行应用程序开发，但是最快速的和支持最好的开发API依然并将永远是Scala方式的API，所以你必须掌握Scala来编写复杂的和高性能的Spark分布式程序；

3，尤其要熟练掌握Scala的trait、apply、函数式编程、泛型、逆变与协变等；

推荐课程：”精通Spark的开发语言：Scala最佳实践”

# Spark平台的API

1，掌握Spark中面向RDD的开发模式，掌握各种transformation和action函数的使用；

2，掌握Spark中的宽依赖和窄依赖以及lineage机制；

3，掌握RDD的计算流程，例如Stage的划分、Spark应用程序提交给集群的基本过程和Worker节点基础的工作原理等

推荐课程：“18小时内掌握Spark：把云计算大数据速度提高100倍以上!”

# Spark内核

Spark的内核部分的学习笔记，在***学习路线篇***中对应章节记录。

此阶段主要是通过Spark框架的源码研读来深入Spark内核部分：

1，通过源码掌握Spark的任务提交过程；

2，通过源码掌握Spark集群的任务调度；

3，尤其要精通DAGScheduler、TaskScheduler和Worker节点内部的工作的每一步的细节；

推荐课程：“Spark 1.0.0企业级开发动手：实战世界上第一个Spark 1.0.0课程，涵盖Spark 1.0.0所有的企业级开发技术”

# Spark上的核心框架

Spark上的核心框架部分的学习笔记，在***学习路线篇***中对应章节记录。

Spark作为云计算大数据时代的集大成者，在实时流处理、图技术、机器学习、NoSQL查询等方面具有显著的优势，我们使用Spark的时候大部分时间都是在使用其上的框架例如Shark、Spark Streaming等：

1，Spark Streaming是非常出色的实时流处理框架，要掌握其DStream、transformation和checkpoint等；

2，Spark的离线统计分析功能，Spark 1.0.0版本在Shark的基础上推出了Spark SQL，离线统计分析的功能的效率有显著的提升，需要重点掌握；

3，对于Spark的机器学习和GraphX等要掌握其原理和用法；

推荐课程：“Spark企业级开发最佳实践”

# 商业级别的Spark项目

通过一个完整的具有代表性的Spark项目来贯穿Spark的方方面面，包括项目的架构设计、用到的技术的剖析、开发实现、运维等，完整掌握其中的每一个阶段和细节，这样就可以让您以后可以从容面对绝大多数Spark项目。

推荐课程：“Spark架构案例鉴赏：Conviva、Yahoo！、优酷土豆、网易、腾讯、淘宝等公司的实际Spark案例”

# Spark解决方案

1，彻底掌握Spark框架源码的每一个细节；

2，根据不同的业务场景的需要提供Spark在不同场景的下的解决方案；

3，根据实际需要，在Spark框架基础上进行二次开发，打造自己的Spark框架；

推荐课程：“精通Spark：Spark内核剖析、源码解读、性能优化和商业案例实战”

前面所述的成为Spark高手的六个阶段中的第一和第二个阶段可以通过自学逐步完成，随后的三个阶段最好是由高手或者专家的指引下一步步完成，最后一个阶段，基本上就是到”无招胜有招”的时期，很多东西要用心领悟才能完成。

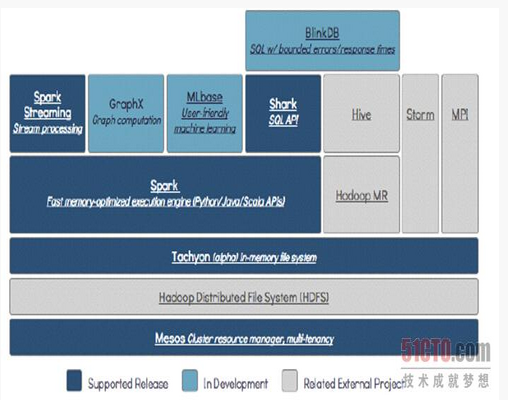
# Spark的学习路线篇

Spark学习路线篇，基于spark的官网文档组织结构，分别对各个部分进行学习，并记录各个部分学习笔记。

# 生态圈

## BDAS

Spark源于BDAS:



## Spark与Hadoop的比较

<http://book.51cto.com/art/201409/451446.htm>

各公司对spark与Hadoop的选择：

1，原先支持Hadoop的四大商业机构纷纷宣布支持Spark；

2，Mahout前一阶段表示从现在起他们将不再接受任何形式的以MapReduce形式实现的算法，另外一方面，Mahout宣布新的算法基于Spark；

3，Cloudera的机器学习框架Oryx的执行引擎也将由Hadoop的MapReduce替换成Spark；

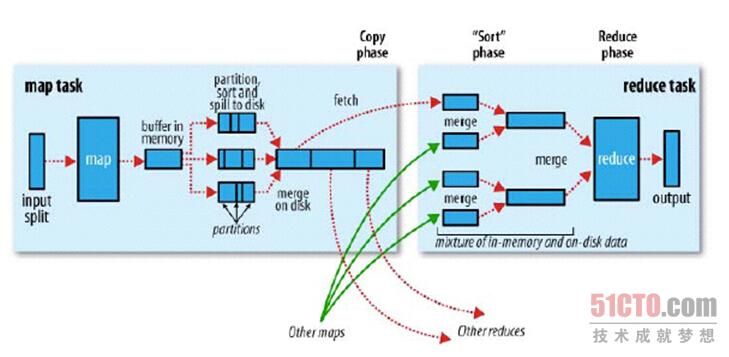
4，Google已经开始将负载从MapReduce转移到Pregel和Dremel上；

5，FaceBook则将负载转移到Presto上；

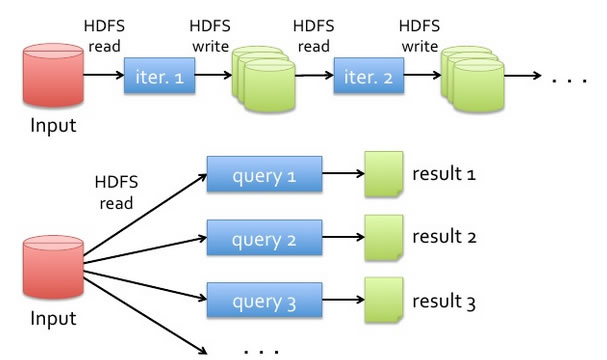
### Hadoop的MapReduce计算模型

Hadoop本身的计算模型决定了Hadoop上的所有工作都要转化成Map、Shuffle和Reduce等核心阶段，由于每次计算都要从磁盘读或者写数据，同时真个计算模型需要网络传输，这就导致了越来越不能忍受的延迟性，同时在前一个任务运行完之前，任何一个任务都不可以运行，这直接导致了其无力支持交互式应用；

首先我们看一下Hadoop经典的处理过程：

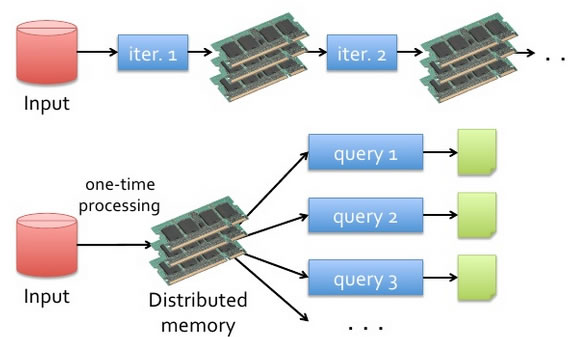
[](http://s8.51cto.com/wyfs02/M01/49/AF/wKiom1QZA-Kx2E5iAACCOw5Z3jo227.jpg)

MapReduce在每次执行的时候都要从磁盘读数据，计算完毕后都要把数据存放到磁盘上：

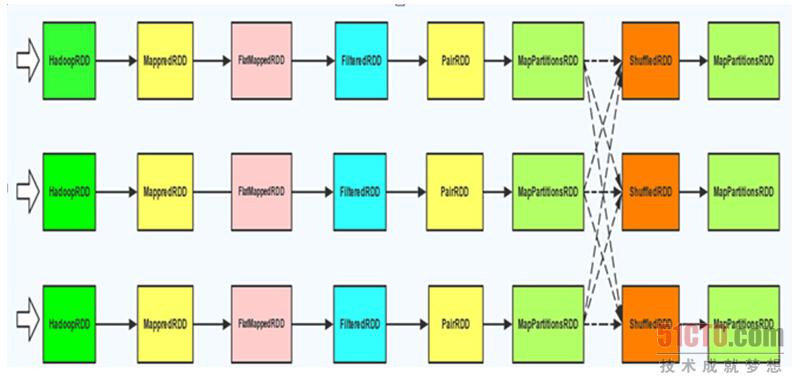


### Spark的计算模型

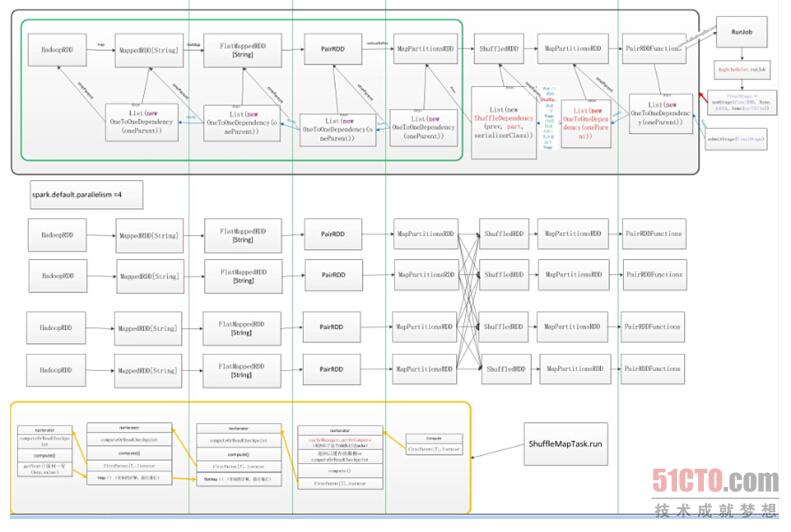
而在Spark中，使用内存替代了使用HDFS存储中间结果：



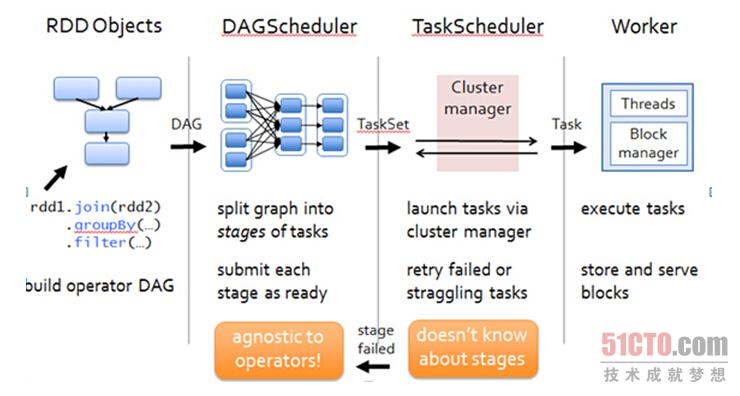
另外一方面，DAG也是Spark快的极为重要的原因，下面是一张DAG图的示例：

[](http://s5.51cto.com/wyfs02/M00/49/B1/wKioL1QZBAvzkg6dAACVeKealQc484.jpg)

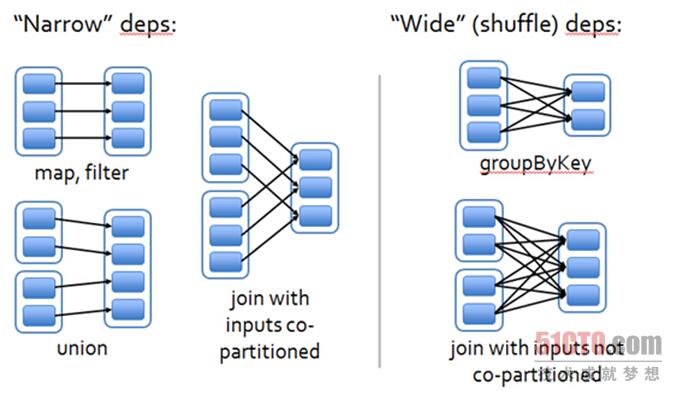
大家也可以看一下网络上一张描述DAG更多细节的图片：

[](http://s3.51cto.com/wyfs02/M00/49/AF/wKiom1QZBAaDh4bFAADd6A7hits354.jpg)

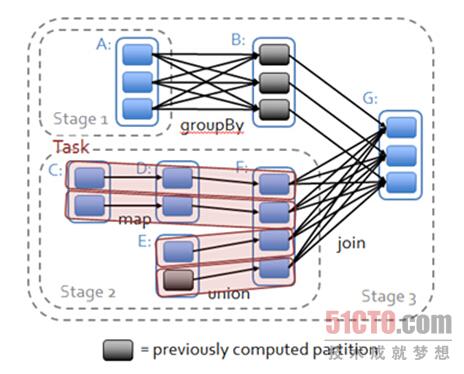
基于DAG，Spark具备了非常精致的作业调度系统：

[](http://s4.51cto.com/wyfs02/M01/49/B1/wKioL1QZBCuyVPxaAACdnxSZEyM489.jpg)

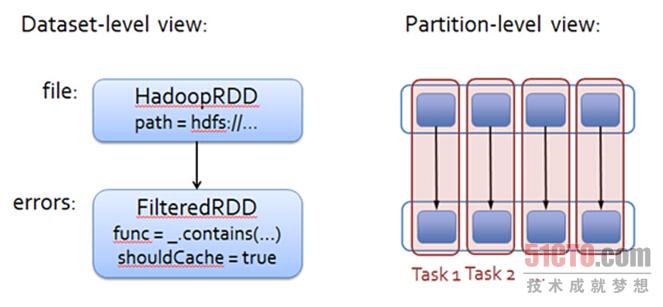
DAG中的依赖有宽依赖和窄依赖之分：

[](http://s6.51cto.com/wyfs02/M02/49/AF/wKiom1QZBCGjW-JaAACUgui8TKY273.jpg)

在DAG图中可以根据依赖对pipeline等优化操作：

[](http://s6.51cto.com/wyfs02/M00/49/B1/wKioL1QZBESB00coAAB8EujG9YY249.jpg)

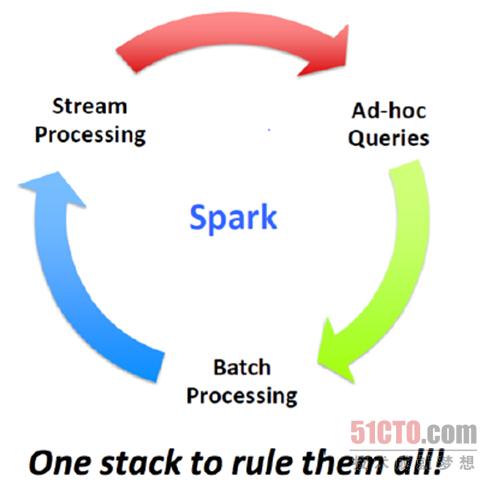
基于RDD和DAG，并行计算整个Job：

[](http://s5.51cto.com/wyfs02/M00/49/AF/wKiom1QZBDqBolUVAABsotUtKaM866.jpg)

Spark之所以快，还有一个原因就是其容错机制，这个我们会在本讲的后面和大家分享。

### Spark的特性

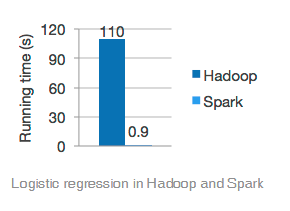
Spark的“One stack to rule them all”的特性，Spark采用一个统一的技术堆栈解决了云计算大数据的如流处理、图技术、机器学习、NoSQL查询等方面的所有核心问题，具有完善的生态系统，这直接奠定了其一统云计算大数据领域的霸主地位；



### 对比实例

<http://spark.apache.org/>

**Apache Spark™** is a fast and general engine for large-scale data processing.



Speed

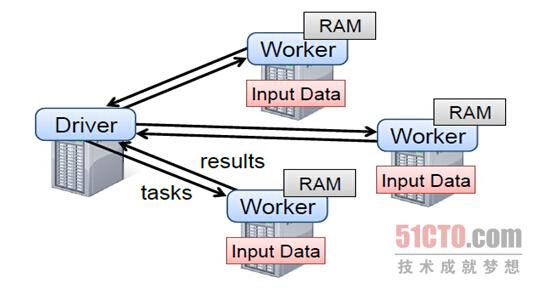
Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.

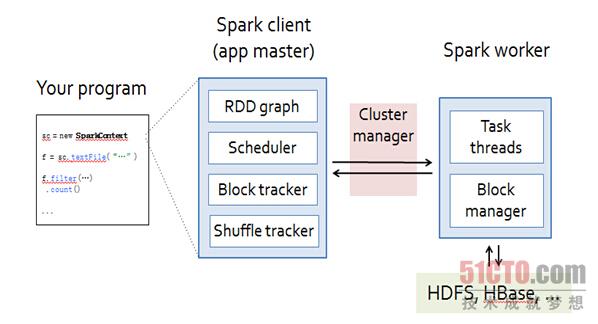
# Spark架构设计与编程模型

## Spark架构设计

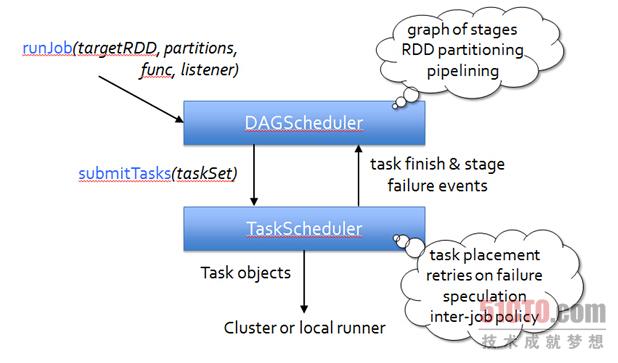
正如所有的分布式结构一样，Spark分布式集群也是主从结构的：

[](http://s8.51cto.com/wyfs02/M00/49/A5/wKiom1QX1vjDc1LgAABZYY1wZtU595.jpg)

Spark运行时组件如下所示：

[](http://s3.51cto.com/wyfs02/M01/49/A7/wKioL1QX1x2TlbcZAABm0_1xrPo716.jpg)

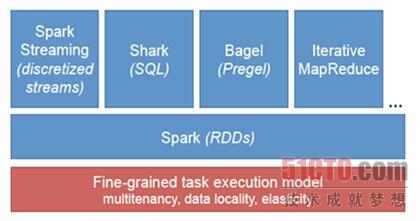
Spark运行时候的事件流如下所示：

[](http://s1.51cto.com/wyfs02/M01/49/A5/wKiom1QX1xayHKacAAB6bg-lKGA055.jpg)

## Spark编程模型

Spark是大数据时代通用而高效的计算平台，基于RDD成功实现了“One stack to rule them all”理念。在Spark中一切都是以RDD为基础和核心。

以RDD为基石的Spark编程模型。



### RDD

RDD本身是一个抽象类，具有很多具体的实现子类，每个RDD都至少有以下三个函数实现，

RDD都会基于Partition进行计算：

默认的Partitioner如下所示：

另外有两个特殊的RDD:

[](http://s9.51cto.com/wyfs02/M02/49/AF/wKiom1QZBP6Ba_qxAAAW12cp5t4430.jpg)

他们都是controlling operations：

[](http://s5.51cto.com/wyfs02/M02/49/B1/wKioL1QZBUrxSvD2AABRqQUGmfo748.jpg)

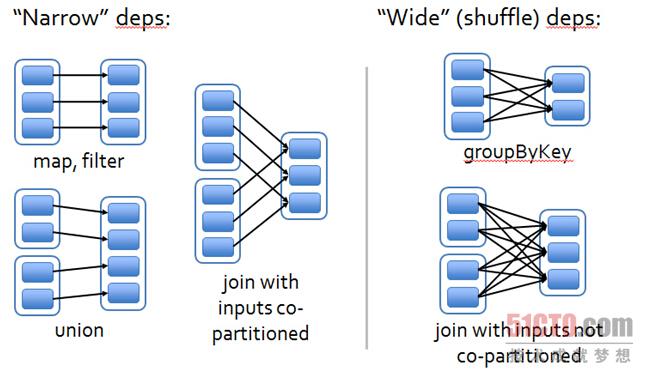
### RDD中的操作

分为transformations和actions两种

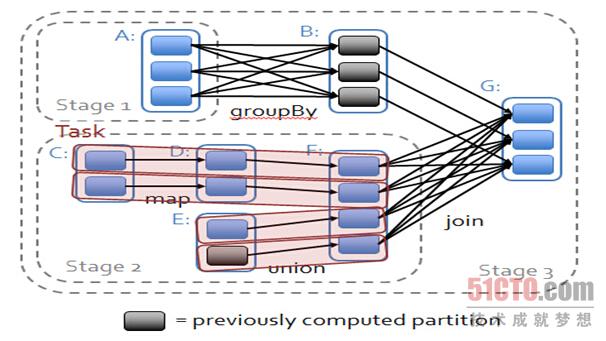
其中的transformations是lazy execution的，需要具体的action去触发，每个action操作都是一个单独的job；

### RDD的依赖

在Spark中RDD是具备依赖关系的，而依赖分为两种：

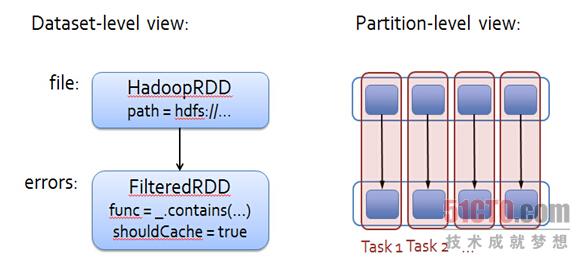
[](http://s8.51cto.com/wyfs02/M00/49/A5/wKiom1QX1seBZgvlAACcdmWVUf4081.jpg)

“Narrow”依赖的一个好处就是可以进行内部的pipeline操作：

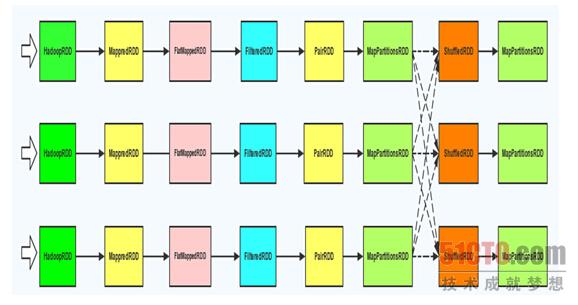
[](http://s3.51cto.com/wyfs02/M02/49/A7/wKioL1QX1uTjo6DMAACfyI_BJzU555.jpg)

### RDD执行框架

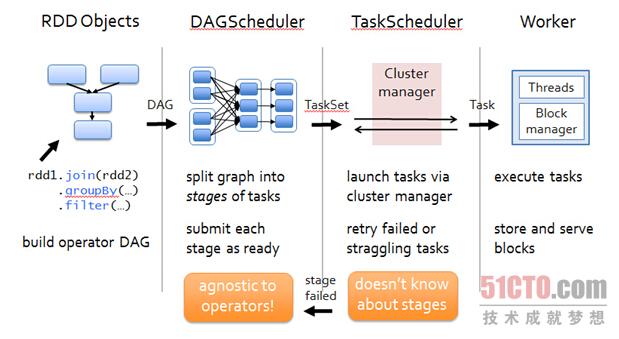
运行的时候是以RDD为统一抽象并行化运行：

[](http://s7.51cto.com/wyfs02/M02/49/A5/wKiom1QX1uHQiSh_AABdq_Oy4tE861.jpg)

更进一步的详细RDD并行化计算过程如下所示：

[](http://s1.51cto.com/wyfs02/M00/49/A7/wKioL1QX1v3RVxuwAABsO4PxdvM244.jpg)

RDD在产生作业调用的时候，经典的过程如下所示：

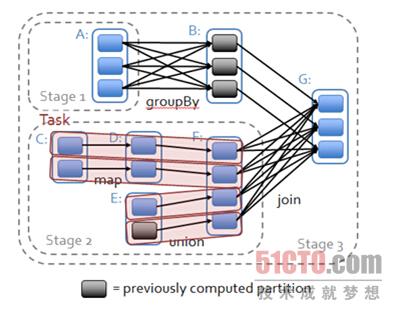
[](http://s8.51cto.com/wyfs02/M01/49/A6/wKiom1QX2l6SgaISAACN5rHw90c511.jpg)

### RDD的持久化

RDD在持久化的需要考虑内存策略

## Spark的高容错机制lineage

基于DAG图，lineage是轻量级而高效的，操作之间相互具备lineage的关系，每个操作只关心其父操作，各个分片的数据之间互不影响，出现错误的时候只要恢复单个Split的特定部分即可：



# Spark内核

Spark内核揭秘共分四个部分：

第一部分：Spark内核初探

第二部分：Spark内核核心源码解析

第三部分：Job全生命周期源码解读

第四部分：Spark性能优化

## Spark内核初探

本讲是Spark内核揭秘的第一部分：Spark内核初探，具体内容如下所示：

1，Spark内核核心术语解析；

2，Spark集群概览；

3，Spark核心组件；

4，Spark任务调度系统初见；

### Spark内核核心术语解析

#### Application

Application是创建了SparkContext实例对象的Spark用户，包含了Driver程序。

Spark-shell是一个应用程序，因为spark-shell在启动的时候创建了SparkContext对象，其名称为sc。

#### Job

和Spark的action相对应，每一个action例如count、savaAsTextFile等都会对应一个Job实例，该Job实例包含多任务的并行计算。

#### Driver Program

运行main函数并且新建SparkContext实例的程序。

#### Cluster Manager

集群资源管理的外部服务，在Spark上现在主要有Standalone、Yarn、Mesos等三种集群资源管理器，Spark自带的Standalone模式能够满足绝大部分纯粹的Spark计算环境中对集群资源管理的需求，基本上只有在集群中运行多套计算框架的时候才建议考虑Yarn和Mesos。

#### Worker Node

集群中可以运行应用程序代码的工作节点，相当于Hadoop的slave节点。

#### Executor

在一个Worker Node上为应用启动的工作进程，在进程中负责任务的运行，并且负责将数据存放在内存或磁盘上，必须注意的是，每个应用在一个Worker Node上只会有一个Executor，在Executor内部通过多线程的方式并发处理应用的任务。

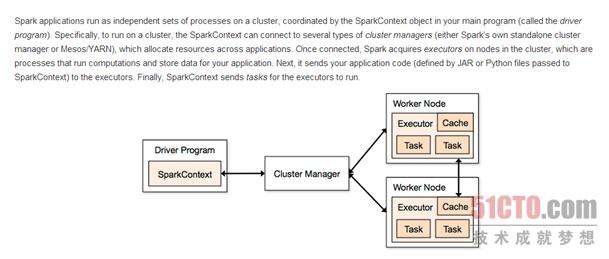
#### Task

被Driver送到executor上的工作单元，通常情况下一个task会处理一个split的数据，每个split一般就是一个Block块的大小：

#### Stage

一个Job会被拆分成很多任务，每一组任务被成为Stage，这个MapReduce的map和reduce任务很像，划分Stage的依据在于：Stage开始一般是由于读取外部数据或者Shuffle数据、一个Stage的结束一般是由于发生Shuffle（例如reduceByKey操作）或者整个Job结束时例如要把数据放到hdfs等存储系统上。

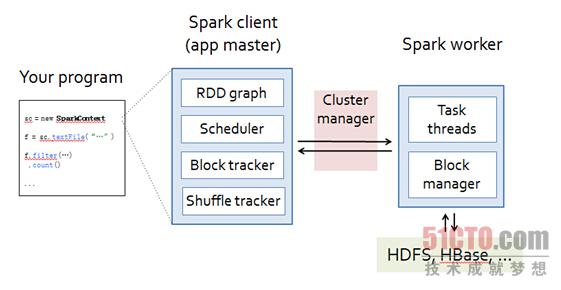
## Spark集群概览



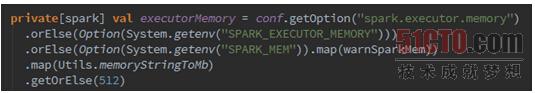
需要注意的是Spark Driver所在的机器需要和Spark集群最好位于同一个网络环境中，因为Driver中的SparkContext实例要要发送任务给不同Worker Node的Executor并接受Executor的一些执行结果信息，一般而言，在企业实际的生产环境中Driver所在机器的配置往往都是比较不错的，尤其是其CPU的处理能力往往都很强悍。

## Spark核心组件

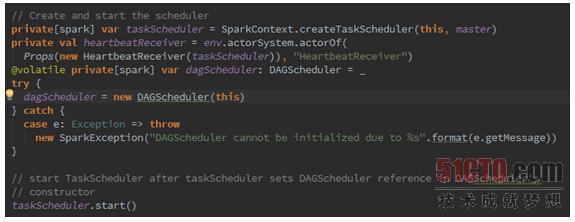
Spark核心组件如下所示：

[](http://s9.51cto.com/wyfs02/M02/4B/4D/wKioL1QpPcGytxZoAABc0lnRzzA334.jpg)

在初始话SparkContext的时候会初始化一系列的内容，例如会查看内存的设置情况：

[](http://s4.51cto.com/wyfs02/M02/4B/4D/wKioL1QpPcqiWS2KAAAxbfwyBas120.jpg)

SparkContext在初始化的时候也会创建和启动scheduler：

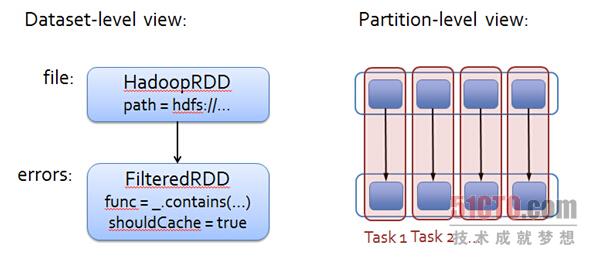
[](http://s5.51cto.com/wyfs02/M02/4B/4B/wKiom1QpPanhXG1AAABWRCj1G3U061.jpg)

集群核心组件中的Block tracker是用于Block和partition对应关系的管理器。

集群核心组件中的Shuffle tracker是用于记录Shuffle操作过程细节的。

从集群中也可以清晰的看出，Executor在执行任务的时候是采用多线程的方式执行的并能够在HDFS或者HBase等系统上存取数据。

在实际的Driver Program运行的时候每个Partition都会由一个Task负责运行：

[](http://s4.51cto.com/wyfs02/M00/4B/4D/wKioL1QpPevzSmJoAABf694Jo1U497.jpg)

也就是说有多少partition就会有多少task在运行，而这些task是并发的运行在executor中的。

## Spark任务调度系统初见

<http://book.51cto.com/art/201409/453029.htm>

## Spark内核核心源码解析

## Job全生命周期源码解读

## Spark性能优化

# Spark核心框架

## Spark Streaming

## Spark SQL

## MLlib(Machine Learning library)

## GraphX(Graph Processing)

## Bagel (Pregel on Spark)

## SparkR

# Spark开发与应用

## Spark-shell

# 部署

## 提交应用程序

## Amazon EC2

## Standalone模式

## Mesos

## YARN

# 监控

# 配置

# 调优

# 作业调度

# 安全

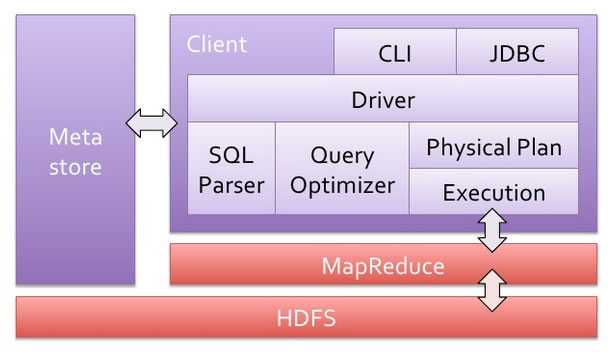
# 硬件配置

# 技巧、FQA

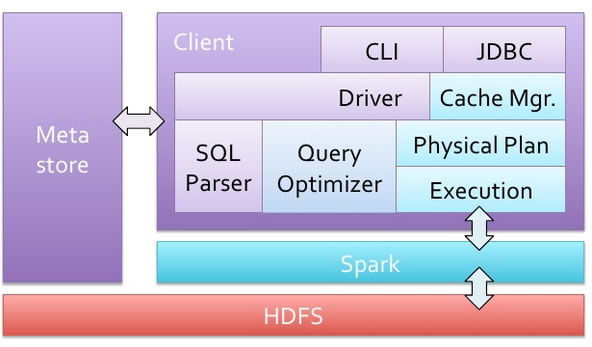
# 参考资料

### Shark

Shark是基于Spark上的“Hive”，看看基于hadoop的Hive：



而Shark的结构图：



Hive是记录每行记录一个对象，而shark是每列记录：



执行SQL时间对比：

