# 一 MapReduce流程

[link\MapReduce过程详解(基于hadoop2.x架构) - py小杰 - 博客园.mhtml](link/MapReduce过程详解(基于hadoop2.x架构) - py小杰 - 博客园.mhtml)





# 二 hive

## hive简介

## Hive的sql与传统sql区别

## Hive体系架构

## Hive数据管理

### 内部表和外部表

### Partition（分区）

### Bucket（分桶）

### 数据类型

### Hive的窗口函数

[link\Hive 之 分析窗口函数 - happy19870612's blog - CSDN博客.mhtml](link/Hive 之 分析窗口函数 - happy19870612's blog - CSDN博客.mhtml)

#### 5.1分析函数定义

分析函数又被称为窗口函数。在联机分析(OLAP)的时候，传统是SQL可能需要进行多个自连接，从而导致查询新能很差。又或者有时候需要多维分析需要在不同层级进行聚合，并且聚合后数据行和密钥聚合的数据行都必须返回。传统是SQL都有一定的局限性。分析函数很好的可以解决这些问题，可以不使用自联结就能在一行中取出聚合和非聚合的行

#### 5.2分析函数组成

function(argument1, argument2…)over([partition-by-cluase][order-by-clasue][windowing-clause])

分析函数有三个部分组成：分区子句，排序子句，以及窗口子句。

分区子句： 其实可以理解为根据什么分组

排序子句：对每一个分区的结果集排序

窗口子句：

[ROWS|RANGE] BETWEEN <开始表达式> AND <结束表达式>

<开始表达式>:[UNBOUNDED PRECEDING| CURRENT ROW|

nRECEDING | n FOLLOWING]

<结束表达式>:[UNBOUNDED FOLLOWING|CURRENT ROW|

nRECEDING | n FOLLOWING]

UNBOUNDEDPRECEDING：针对当前分区里面的当前行而言的前面所有行

UNBOUNDEDFOLLOWING：针对当前分区里面的当前行而言的后面所有行

CURRENTROW： 当前行

nRECEDING： 针对当前行而言，前n行

nFOLLOWING：针对当前行而言，后n行

## hive的优化

# 三 Hadoop1和2的区别

## （一）slot的含义

1）slot就是槽的意思，是一个资源单位，只有给task分配了一个slot之后，这个task才可以运行。slot分两种，map slot沪蓉reduce slot。另外，slot是一个逻辑概念，一个数据节点的slots数量既不是CPU的核数，也不是memory chip，一个节点的slot数量用来表示此节点的资源容量或是计算计算能力的大小，也就是说slot是hadoop的资源单位。

2）系统中map slots总数与reduce slots总数的计算方式如下:

map slots 总数=集群节点数\*mapred.tasktracker.map.tasks.maximum（默认是2）;

reduce slots 总数=集群节点数\*mapred.tasktracker.reduce.tasks.maximum（默认是2）;

# 四 yarn

# 五spark

## （一）批处理和流处理

### 1批处理系统

***批处理***在大数据世界有着悠久的历史。批处理主要操作大容量静态数据集，并在计算过程完成后返回结果。

#### 批处理模式中使用的数据集通常符合下列特征：

有界：批处理数据集代表数据的有限集合

持久：数据通常始终存储在某种类型的持久存储位置中

大量：批处理操作通常是处理极为海量数据集的唯一方法

Apache Hadoop是一种专用于批处理的处理框架，Hadoop的处理功能来自MapReduce引擎。MapReduce的处理技术符合使用键值对的map、shuffle、reduce算法要求。

#### 使用场景：

需要处理大量数据的任务通常最适合用批处理操作进行处理（准确性要求较高），大量数据的处理需要付出大量时间，因此批处理不适合实时性要求较高的场合

### 2流处理系统

流处理中的数据集是“无边界”的，流处理方式无需针对整个数据集执行操作，而是对通过系统传输的每个数据项执行操作，即处理瞬时数据，实时性要求较高

## spark之批处理和流处理模式

### 1批处理模式

与MapReduce不同，Spark的数据处理工作全部在内存中进行，只在一开始将数据读入内存，以及将最终结果持久存储时需要与存储层交互。所有中间态的处理结果均存储在内存中。

为了实现内存中批计算，Spark会使用一种名为Resilient Distributed Dataset（弹性分布式数据集），即RDD的模型来处理数据。这是一种代表数据集，只位于内存中，永恒不变的结构。针对RDD执行的操作可生成新的RDD。每个RDD可通过世系（Lineage）回溯至父级RDD，并最终回溯至磁盘上的数据。Spark可通过RDD在无需将每个操作的结果写回磁盘的前提下实现容错

### 2流处理模式

流处理能力是由Spark Streaming实现的。Spark本身在设计上主要面向批处理工作负载，为了弥补引擎设计和流处理工作负载特征方面的差异，Spark实现了一种叫做微批（Micro-batch）\*的概念

Spark Streaming会以亚秒级增量对流进行缓冲，随后这些缓冲会作为小规模的固定数据集进行批处理。这种方式的实际效果非常好，但相比真正的流处理框架在性能方面依然存在不足

### 3优势和使用场景

#### 3.1优势

1）使用Spark而非Hadoop MapReduce的主要原因是速度。在内存计算策略和先进的DAG调度等机制的帮助下，Spark可以用更快速度处理相同的数据集。

2）Spark的另一个重要优势在于多样性。该产品可作为独立集群部署，或与现有Hadoop集群集成。该产品可运行批处理和流处理，运行一个集群即可处理不同类型的任务。

3）除了引擎自身的能力外，围绕Spark还建立了包含各种库的生态系统，可为机器学习、交互式查询等任务提供更好的支持。相比MapReduce，Spark任务更是“众所周知”地易于编写，因此可大幅提高生产力。

#### 3.2使用场景

1. 为流处理系统采用批处理的方法，需要对进入系统的数据进行缓冲。缓冲机制使得该技术可以处理非常大量的传入数据，提高整体吞吐率，但等待缓冲区清空也会导致延迟增高。这意味着Spark Streaming可能不适合处理对延迟有较高要求的工作负载。

（对于重视吞吐率而非延迟的工作负载，则比较适合使用Spark Streaming作为流处理解决方案）

2）由于内存通常比磁盘空间更贵，因此相比基于磁盘的系统，Spark成本更高。然而处理速度的提升意味着可以更快速完成任务，在需要按照小时数为资源付费的环境中，这一特性通常可以抵消增加的成本。

3）Spark内存计算这一设计的另一个后果是，如果部署在共享的集群中可能会遇到资源不足的问题。相比Hadoop MapReduce，Spark的资源消耗更大，可能会对需要在同一时间使用集群的其他任务产生影响。从本质来看，Spark更不适合与Hadoop堆栈的其他组件共存一处。