# 20200704第2课

## 模板设计模式

## HDFS API 入口类 FileSystem

# 20200705第3课

## 引入MapReduce

## 引入Shuffle

相同的key分发到同一个reduce进行处理，就是shuffle

## MR中常用的数据序列化类型

## 词频统计、八股文编程

## Debug Hadoop源码

# 20200708第4课

## 序列化和反序列化

内存的对象转成字节数组，以便于存储或者网络的数据传输

## 自定义序列化类

## InputFormat

### getSplits()

### createRecordReader()

## 切片的源码讲解

## KeyValueTextInputFormat(不常用)

## 作业

Debug源码找到Hadoop在LocalFileSystem中的blockSize为32M的源码

# 20200711第5课

## NLineInputFormat(不常用)

## DBInputFormat

## 自定义Partitioner

## Combiner

## 自定义OutputFormat

## 作业

将MR读取MySQL的代码在服务器上跑起来，这里的MySQL驱动包是需要通过****-libjars****传入的

具体查看：

<https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

# 20200712第6课

## 排序

### 全排序

保证输出是全局有序的;只有1个reduce; Hive中的order by就是全排序

### 分区/局部排序

保证每个reducer的输出是有序的; Hive中的sort by

### 二次排序

## 分组TopN问题

## Join（重点）

### Reduce Join/Shuffle Join

以join条件作为map数据的key，不同来源的数据需要打上一个标签，经过shuffle，相同的key的数据会落入到同一个reduce中，我们的join操作就是在reduce端完成的

### Map Join

join真正是在map端完成的，也就是没有reducer,必然就没有shuffle；

前提:只适合大表join小表；

实现原理:把小表的数据加到缓存中，在读取大表的一行数据时，直接根据join的条件，去缓存中匹配

## 作业

变形的wc解决shuffle时的数据倾斜问题(双层group by,先打散,再汇总)

# 20200715第7课

## Shell

### awk

### Sed

## 标准的脚本格式

## 监控HDFS HA状态并邮件报警脚本

# 20200717

## 邮件配置

## Spark提交脚本

## Spark提交脚本中带check

# 20200718

## 讲解MR的官网教程

<https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

## Shuffle 执行过程

## Shuffle 执行优化

1. 备用

# 20200719

## Hive常用命令

### hive -e

### hive -f

## show functions;查看Hive所有内置函数

## FunctionRegistry是Hive内置函数的注册入口类

## 定义UDF

## 查看具体的命令 desc function extended upper;

## Hive排序

### 全排序 order by

reducer只有1个，保证了全局有序，但是牺牲了速度

Hive.mapred.mode=strict的模式下，order by 一定要与limit配合使用

### 局部排序

每个reducer中有序，但并不能全局有序；

### 二次排序

## 决定reduce的个数的参数

mapred.reduce.tasks:默认的reduce的个数

## distributed by 和 cluster by

distributed by a 相当于按照a来进行分区相当于Hadoop代码中指定partitioner

cluster by a 等价于 distributed by a sort by a

## 讲解将两个MR的job连接在一起写在同一个Driver中，就是Hadoop为了解决数据倾斜，可以先拼随机数，再去掉随机数。也就是Hive中双层group by的代码实现

先打散再收敛

# 20200722

## Hive优化

### 是否起MR hive.fetch.task.conversion

### 是否跑本地 hive.exec.mode.local.auto

hive.exec.mode.local.auto.inputbytes.max

hive.exec.mode.local.auto.tasks.max

hive.exec.mode.local.auto.input.files.max

### 严格模式hive.mapred.mode

### 推测式执行 hive.mapred.reduce.tasks.speculative.execution

### 列裁剪hive.optimize.cp

### 谓词下推hive.optimize.ppd

### 调整reduce个数

hive.exec.reducers.bytes.per.reducer

hive.exec.reducers.max

1. 窗口函数

<https://databricks.com/blog/2015/07/15/introducing-window-functions-in-spark-sql.html>

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+WindowingAndAnalytics>

# 20200725

# Hive函数

## collect\_set

## collect\_lsit

## lateral view

## nvl

## concat

## concat\_ws

## case when

## if

# 窗口函数

## cume\_dist

## percent\_rank

## lag

## lead

## first\_value

## last\_value

1. UDTF函数编写

## 入口类Operator

### TableScanOperator

### ReduceSinkOperator

### JoinOperator

### MapJoinOperator

### SelectOperator

### FileSinkOperator

### FilterOperator

### GroupByOperator

### LimitOperator

## 查看执行计划

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Explain

## hive.auto.convert.join

## hive.smalltable.filesize

## hive.mapjoin.smalltable.filesize

## 数据倾斜

### shuffle后，某个/些task处理的数据不均衡，本质上是key的分布不均匀

### 引起shuffle的操作：group by/common join/count(distinct)

### select \* from a join b on a.id = b.id;假设id有为null的情况

#### case when a.id is null then concat(xxx,rand()) else xxx end

#### 如果求count(distinct id) 直接求count(distinct id) where id is not null,对结果值进行加1即可

### group by 倾斜，调整hive.groupby.skewindata

这个参数底层就是双重group by的思路，先打散，再收敛

# 20200730 ——28-Scala高级

## 隐式转换

### 隐式转换方法

### 隐式转换类

### 隐式转换参数

## 隐式转换作用域

### 当前作用域

### 相关类及伴生对象中

## Scala IO 读文件

### Source

### StdIn

## Scalike JDBC

CRUD

## Scala泛型

### 上界 <:

### 下界 >:

### 视图界定 <%

### 协变 父类引用指向子类对象

### 逆变 子类引用指向父类对象

## Scala比较

### Ordering 类似于Compartor

### Ordered 类似于Comparable

# 20200808

## Hadoop日志离线项目

### 1.1离线项目架构图

## Flume架构