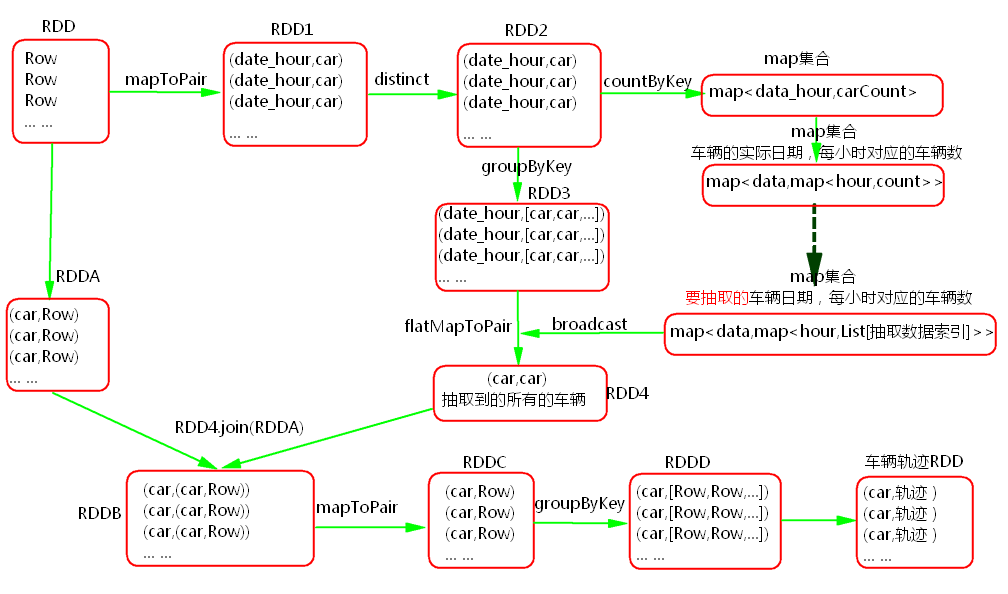
1. 区域多卡扣碰撞分析

查询两个区域同时出现的车辆和经过的卡扣信息。

1. 随机抽取N辆车的信息

假设一天的车流量是10000，在7~8点之间车流量为2000，在13~14点车流量是3000，那么我们想要在一天中随机抽取100辆车辆信息，那么，应该在7~8点抽取2000/10000\*100=20辆，在13~14点之间抽取3000/10000\*100=30辆。



1. Spark调优
2. 资源调优
3. .搭建Spark集群的时候要给Spark集群足够的资源（core，memory）

在spark安装包的conf下spark-env.sh

SPARK\_WORKER\_CORES

SPARK\_WORKER\_MEMORY

SPARK\_WORKER\_INSTANCE

1. .在提交Application的时候给Application分配更多的资源。

提交命令选项：（在提交Application的时候使用选项）

--executor-cores

--executor-memory

--total-executor-cores

配置信息：（在Application的代码中设置

在Spark-default.conf中设置）

spark.executor.cores

spark.executor.memory

spark.max.cores

1. 并行度调优

原则：一个core一般分配2~3个task,每一个task一般处理1G数据（task的复杂度类似wc）

提高并行度的方式：

1. .如果读取的数据在HDFS上，降低block块的大小
2. .sc.textFile(path,numPartitions)
3. sc.parallelize(list,numPartitions) 一般用于测试
4. coalesce、repartition可以提高RDD的分区数。
5. 配置信息：

spark.default.parallelism not set (默认executor core的总个数)

spark.sql.shuffle.partitions 200

1. 自定义分区器
2. 代码调优
3. 避免创建重复的RDD

val rdd1 = sc.textFile(“xxx”)

val rdd2 = sc.textFile(“xxx”)

在执行效率上没有区别，但是代码乱。

1. 在其他的job中对于重复使用的RDD要使用持久化算子

cache:

MEMORY\_ONLY

persist：

MEMORY\_ONLY

MEMORY\_ONLY\_SER

MEMORY\_AND\_DISK\_SER

一般不要选择带有\_2的持久化级别。

checkpoint:

1. 如果一个RDD的计算时间比较长或者计算起来比较复杂，一般将这个RDD的计算结果保存到HDFS上，这样数据会更加安全。
2. 如果一个RDD的依赖关系非常长，也会使用checkpoint,会切断依赖关系，提高容错的效率。
3. 尽量使用广播变量

使用广播变量可以大大的降低集群中变量的副本数。

不使用广播变量：变量的副本数和task数一致。

使用广播变量:变量的副本数与Executor数一致。

广播变量最大可以是多大?

ExecutorMemory\*60%\*90%\*80%

1. Spark相关问题
2. Spark比MR快的原因？
3. Spark的计算结果可以放入内存，支持基于内存的迭代，MR不支持。
4. Spark有DAG有向无环图，可以实现pipeline的计算模式。
5. 资源调度模式：Spark粗粒度资源调度，MR是细粒度资源调度。

资源复用：Spark中的task可以复用同一批Executor的资源。

MR里面每一个map task对应一个jvm，不能复用资源。

1. Spark中主要进程的作用？

Driver进程：负责任务的分发和结果的回收。

Executor进程：负责具体任务的执行。

Master进程：Spark资源管理的主进程，负责资源调度。

Worker进程：Spark资源管理的从进程，woker节点主要运行Executor