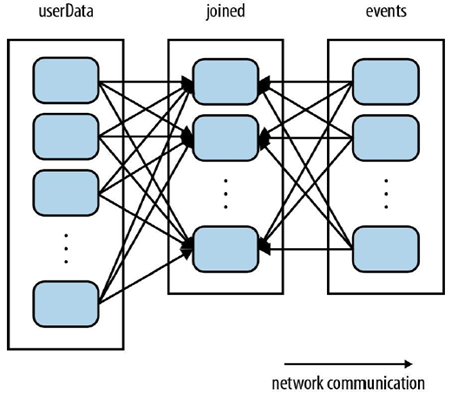
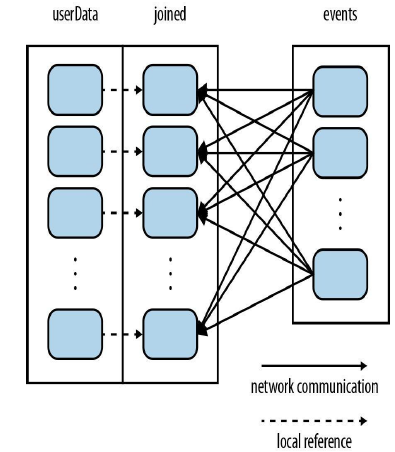
Spark Using Optimization

1:减少Shuffle网络数据传输量，可以大大加速计算效率。

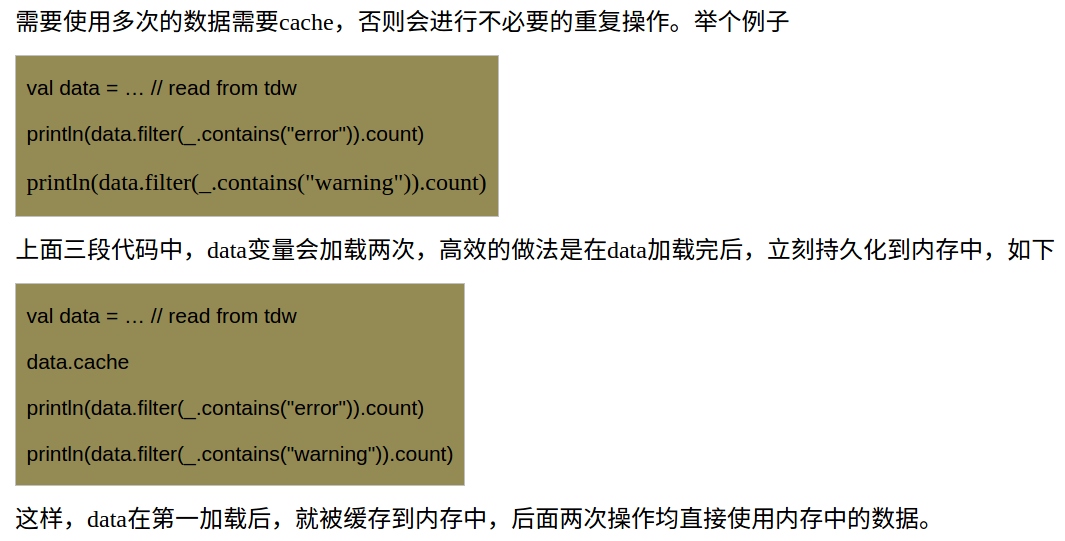
一般来讲，shuffle的过程如下：可能存在两次网络传输



如果我们多告诉Spark 一些信息，比如制定UserData的Partitioner,那么就会现在UserData所在的节点进行分区，请求events的数据直接shuffle到对应的userData partition中，可以有效减少数据穿数量。

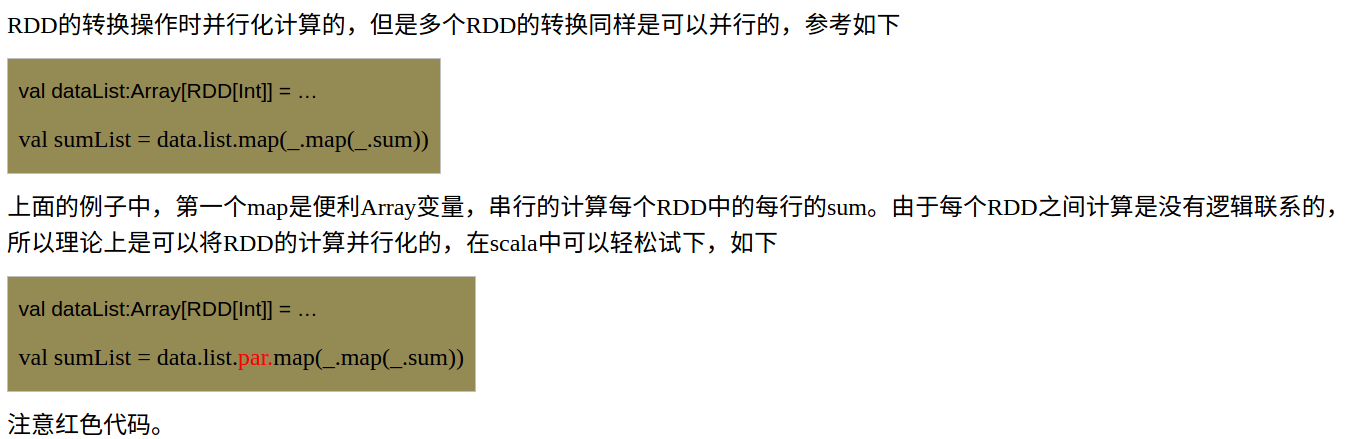


**2：RDD缓存:transformation是不会有cache 效果的，**

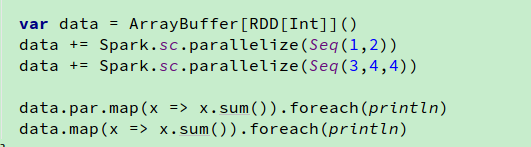


**3：转换并行化:TODO**

**这里主要使用了scala 为了现代计算机多核架而设计的parallelize collection的特性。具体可以参考scala 官方文档，附录中也有一些我自己的理解。**



实验如下：



附录：

实验发现确实parallelize collection 确实有一定的加速效果，但是应该注意避免“side-effect”，比如 sum, reduce 等操作

\* sum = 0

\* val list = (1 to 1000).toList

\* list.foreach(sum + \_)

每次运行结果都不一致。

List.par.map()

