

The frozen part of the MapReduce framework is a large distributed sort. The hot spots, which the application defines, are:

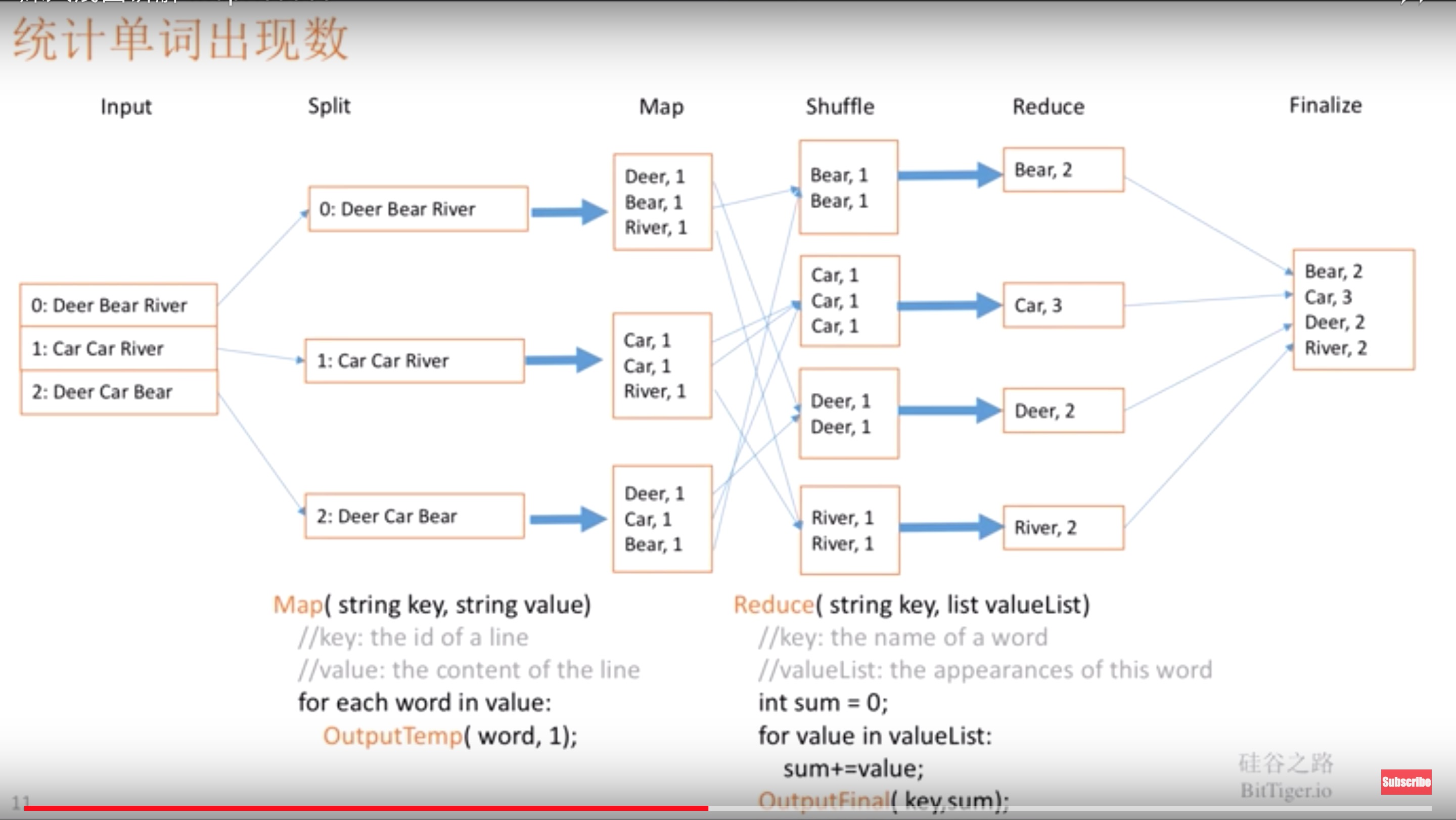
1. an input reader
2. a Map function
3. a partition function
4. a compare function
5. a Reduce function
6. an output writer

A MapReduce framework (or system) is usually composed of three operations (or steps):

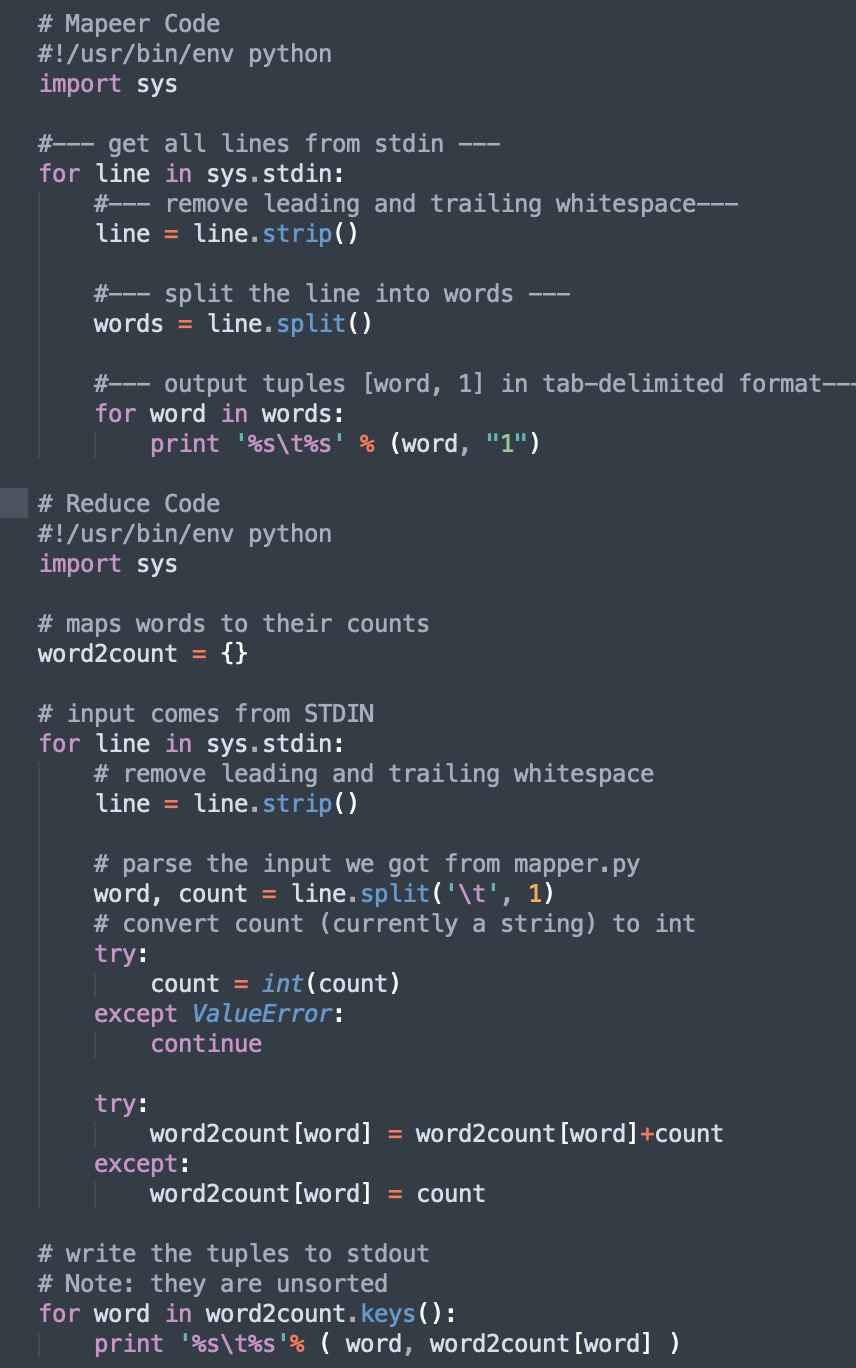
**Map**: each worker node applies the map function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of redundant input data is processed.

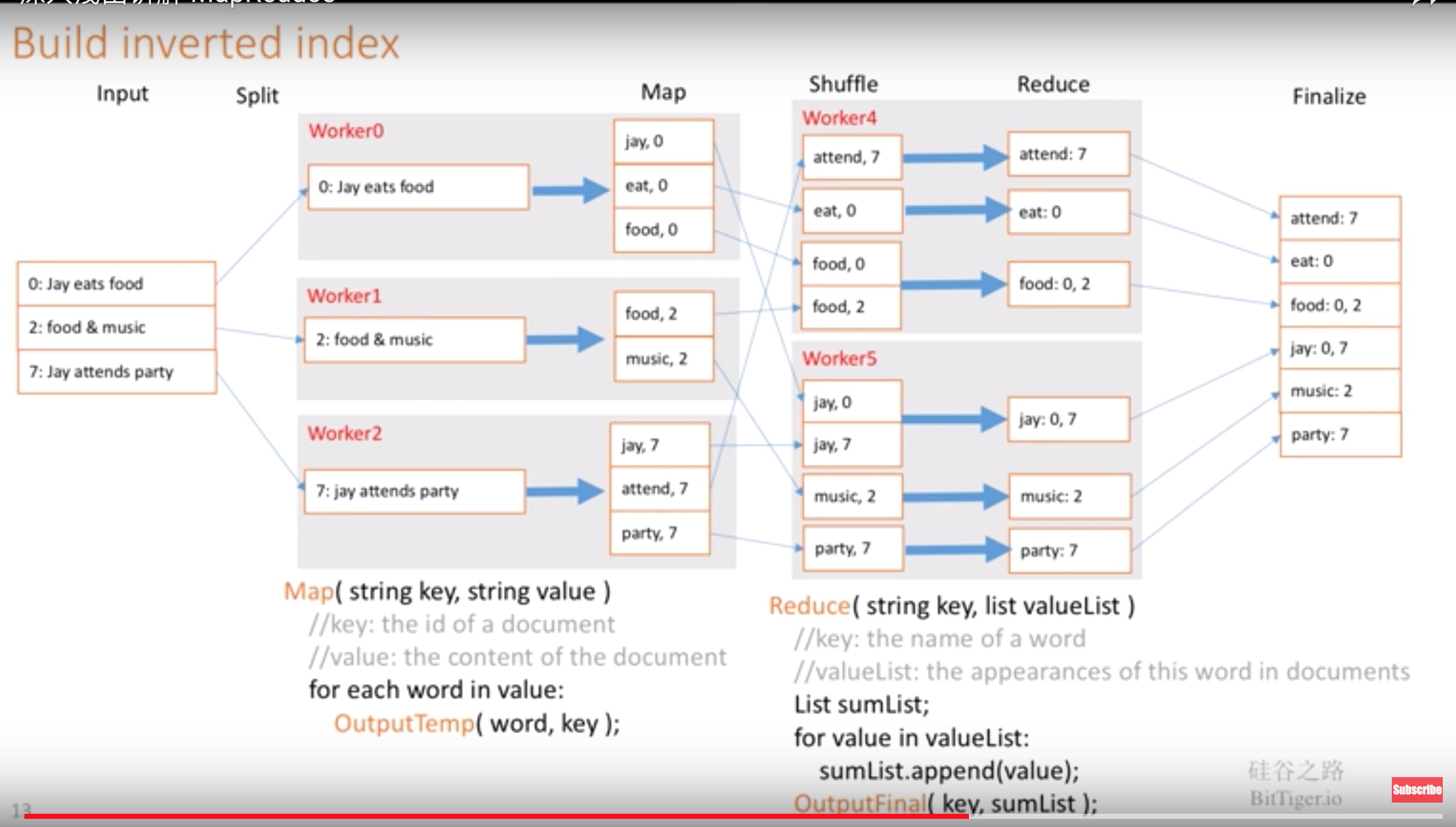
**Shuffle**: worker nodes redistribute data based on the output keys (produced by the map function), such that all data belonging to one key is located on the same worker node.

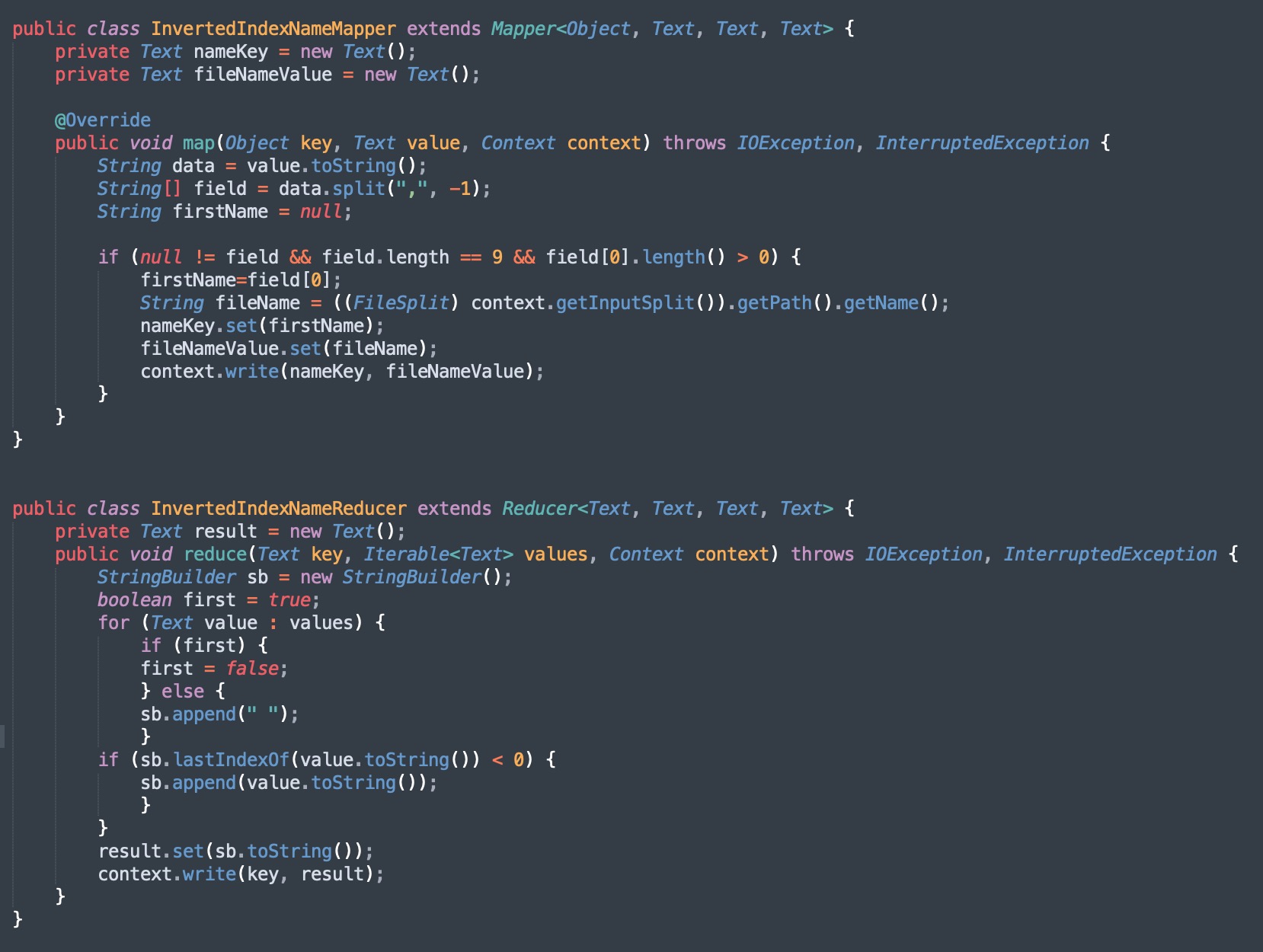
**Reduce**: worker nodes now process each group of output data, per key, in parallel.

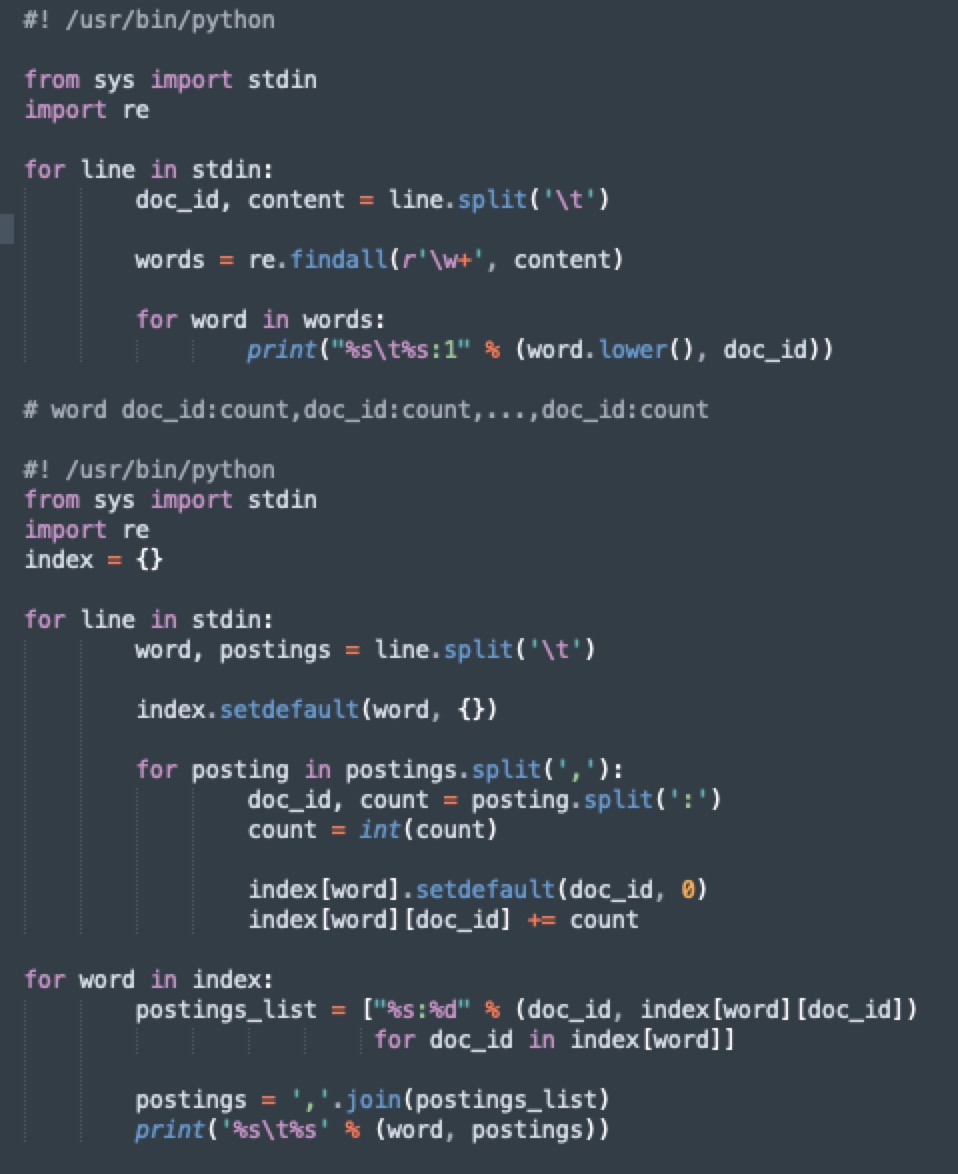


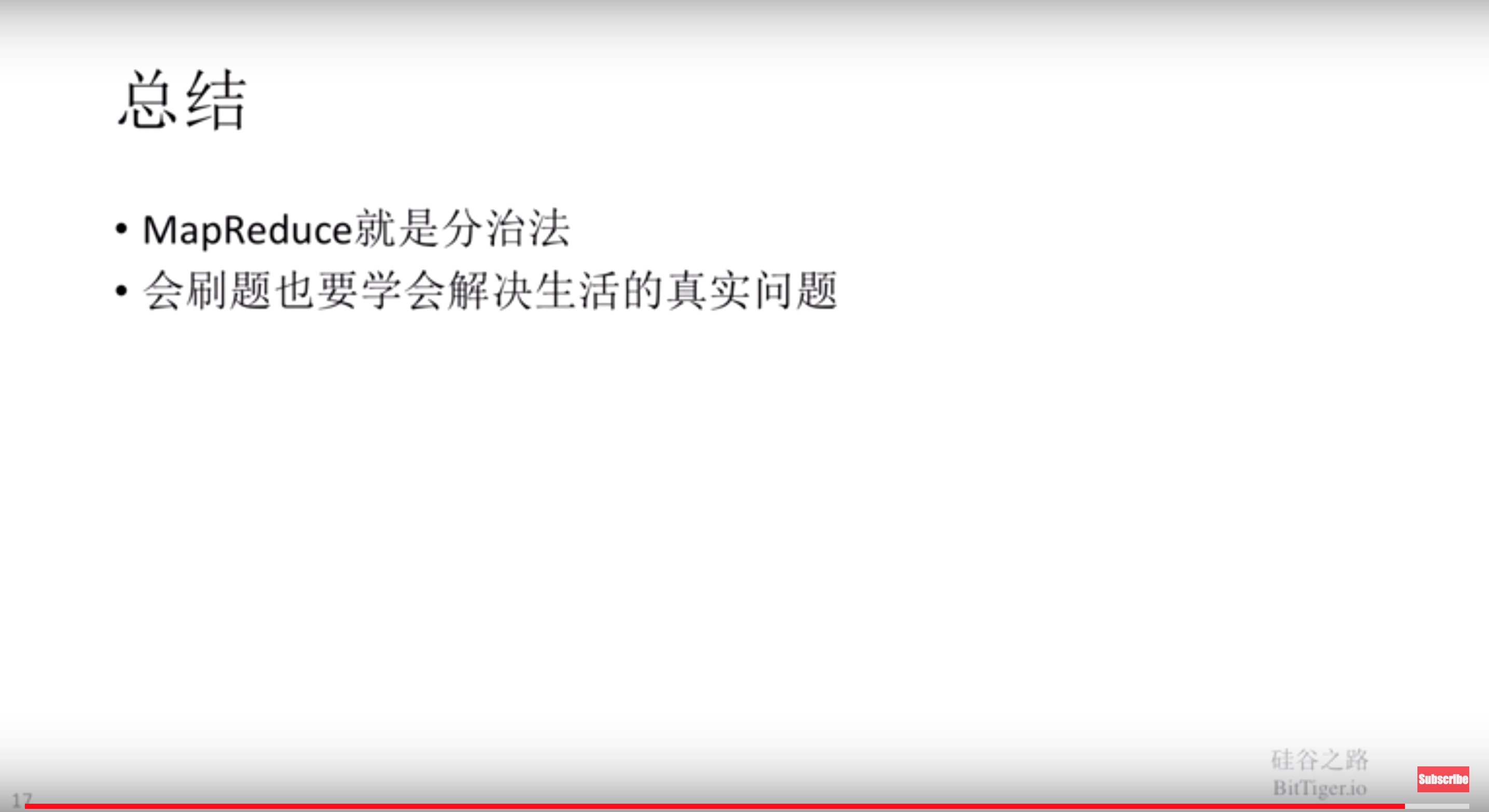


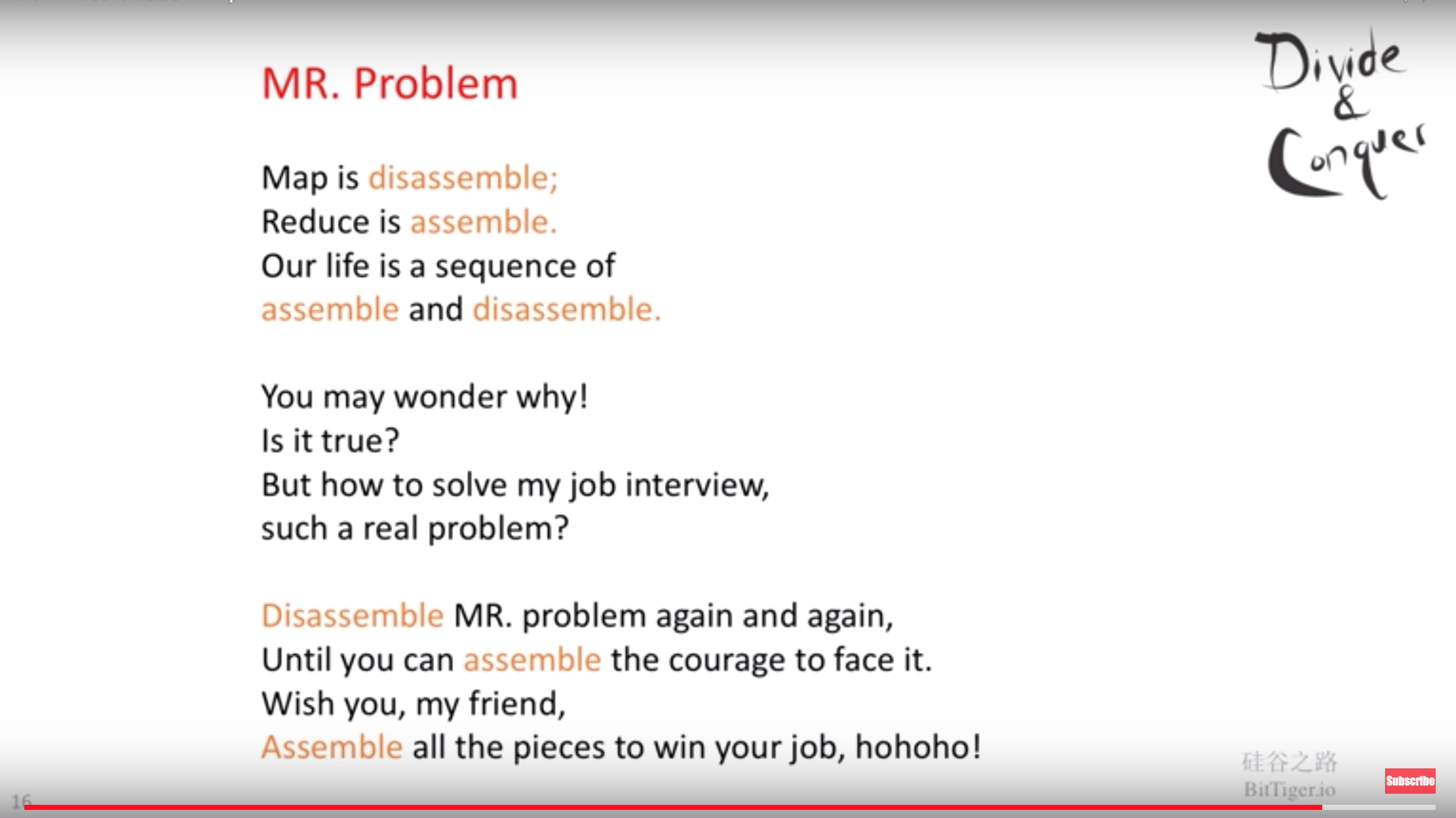












协同过滤

1）基于用户的协同过滤算法 (User Based Collaborative filtering algorithm)

给用户推荐和他兴趣相似的其他用户喜欢的物品

2）基于物品的协同过滤算法 (Item Based Collaborative filtering algorithm)

给用户推荐和他之前喜欢的物品相似的物品

1. 基于用户的协同过滤算法

Step1: 找到和目标用户兴趣相似的用户集合

Step2: 找到这个集合中的用户喜欢的，且用户没有听说过得物品推荐给目标用户

模拟情景:

用户 A B C D E F

商品 1 2 3 4 5 6

行为 点击 1.0分 搜索 2.0分 收藏 5.0分 付款 10.0分

用户行为列表如下

用户    物品    行为 用户    物品    行为

A        1        点击 A        3        收藏

A        4        搜索 B        2        搜索

B         5        搜索 C        1        收藏

C         6        付款 D        1        付款

D        5        收藏 E        3        收藏

E        4        点击 F        2        收藏

F        3        搜索 F        6        点击

算法步骤

1.根据用户行为列表计算物品、用户的评分矩阵

                1                2                3              4            5            6

A              1                0                5              3            0            0

B              0                3                0              0            3            0

C              5                0                0              0            0            10

D              10              0                0              0            5            0

E               0                0                5              1            0            0

F               0                5                3              0            0            1

2.根据评分矩阵计算用户与用户相似度矩阵

将所有用户两两计算相似度

                A                B                C                D                E                F

A              1                0              0.08            0.15            0.93            0.43

B              0                1                 0                 0.32            0                0.6

C             0.08             0                1                0.4              0                0.15

D             0.15             0.32         0.4              1                 0                0

E              0.93             0               0                 0                 1                0.5

F              0.43             0.6           0.15             0                 0.5              1

3. 相似度矩阵X评分矩阵=推荐列表

                1                2                 3                 4                5                6

A             2.9              2.2              11.0             3.9              0.8             1.2

B             3.2              6.0              1.8               0                 4.6             0.6

C             9.1              0.8              0.9               0.2              2.0             10.2

D             12.2            1.0              0.8               0.5              6.0              4.0

E              0.9              2.5              11.2             3.8              0                0.5

F              1.2              6.8              7.7               1.82            1.8             2.5

4.在推荐列表中，将之前产生过操作的物品（即评分矩阵中有过评分的物品）置零

                1                2                 3                 4                5                6

A             0                2.2               0                 0                0.8             1.2

B             3.2              0                 1.8               0                0                0.6

C             0                 0.8              0.9               0.2              2.0             0

D             0                1.0              0.8               0.5              0                4.0

E              0.9              2.5              0                 0                 0                0.5

F              1.2              0                0                 1.82             1.8              0

MapReduce步骤

1.根据用户行为列表构建评分矩阵

输入：用户ID，物品ID，分值

输出：用户ID（行）——物品ID（列）——分值

2.利用评分矩阵构建用户与用户的相似度矩阵

输入：步骤1输出

缓存：步骤1输出

（输出和缓存是相同的文件）

输出：用户ID（行）——用户ID（列）——相似度

3.将评分矩阵转置

输入：步骤1输出

输出：物品ID（行）——用户ID（列）——分值

4.用户与用户相似度矩阵X评分矩阵

输入：步骤2输出

缓存：步骤3输出

输出：用户ID（行）——物品ID（列）——分值

5.根据评分矩阵，将步骤4的输出中，用户已经有过行为的商品评分置零

输入：步骤4输出

缓存：步骤1输出

输出：用户ID（行）——物品ID（列）——分值（最终推荐列表）

1.ItermCF的基本思想

基于物品相似度的协同过滤推荐的思想大致可分为两部分：

1.计算物与物之前的相似度

2.根据用户的行为历史，给出和历史列表中的物品相似度最高的推荐

数据集字段：

1． User\_id: 用户ID

2． Item\_id: 物品ID

3． preference:用户对该物品的评分

算法的思想：

1． 建立物品的同现矩阵A，即统计两两物品同时出现的次数

数据格式：Item\_id1:Item\_id2 次数

2． 建立用户对物品的评分矩阵B，即每一个用户对某一物品的评分

数据格式：Item\_id user\_id:preference

3． 推荐结果=物品的同现矩阵A \* 用户对物品的评分矩阵B

数据格式：user\_id item\_id,推荐分值

4. 过滤用户已评分的物品项

5.对推荐结果按推荐分值从高到低排序

原始数据：

1,101,5.0 1,102,3.0 1,103,2.5

2,101,2.0 2,102,2.5 2,103,5.0 2,104,2.0

3,101,2.0 3,104,4.0 3,105,4.5 3,107,5.0

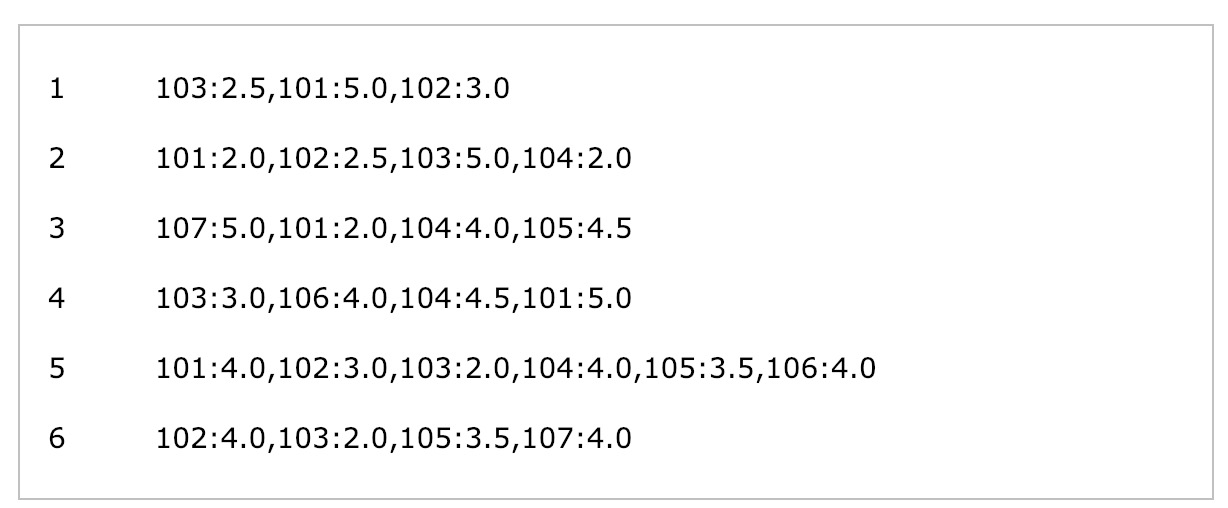
4,101,5.0 4,103,3.0 4,104,4.5 4,106,4.0

5,101,4.0 5,102,3.0 5,103,2.0 5,104,4.0 5,105,3.5 5,106,4.0

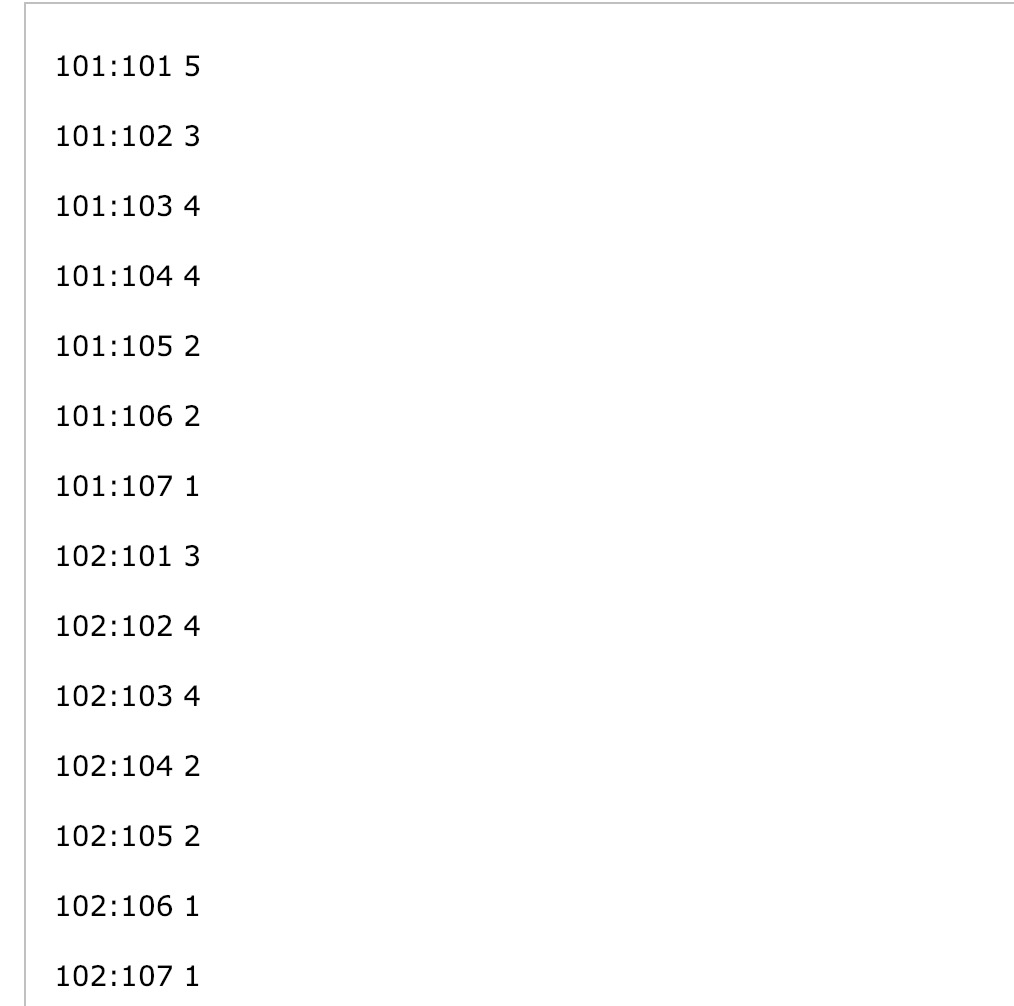
6,102,4.0 6,103,2.0 6,105,3.5 6,107,4.0

Hadoop MapReduce程序分为四步：

第一步： 读取原始数据，按用户ID分组，输出文件数据格式为

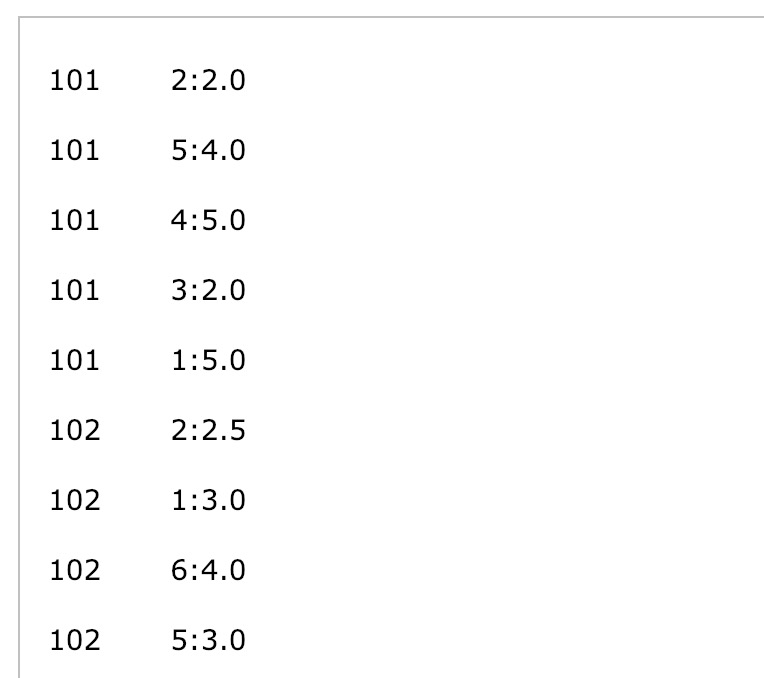


第二步：统计两两物品同时出现的次数，输出文件数据格式为

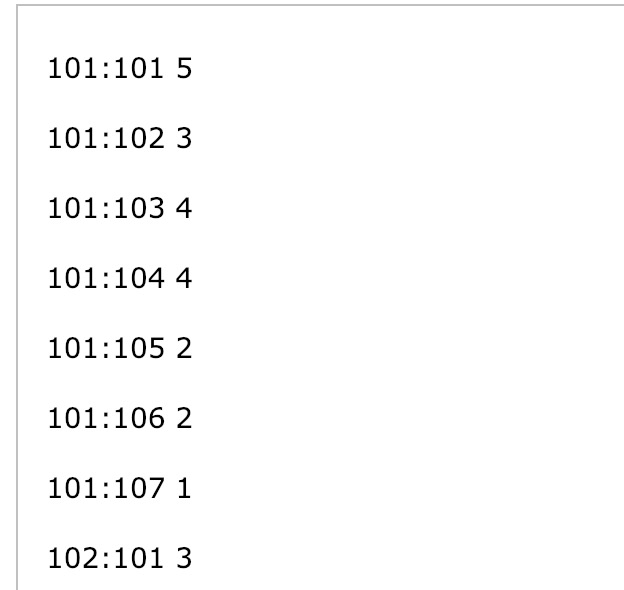


第三步：生成用户评分矩阵和物品同现矩阵

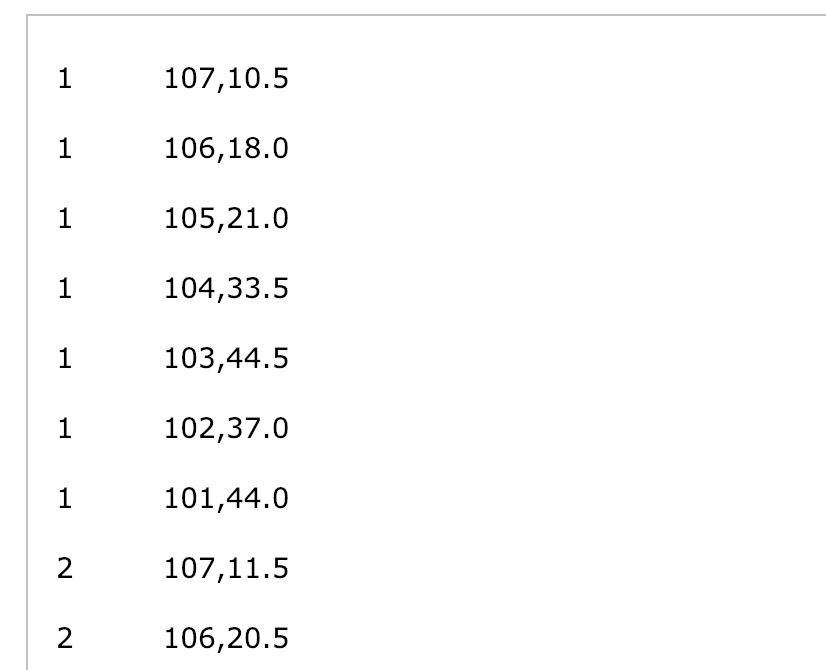
第一个mapper结果为用户评分矩阵，结果如下：

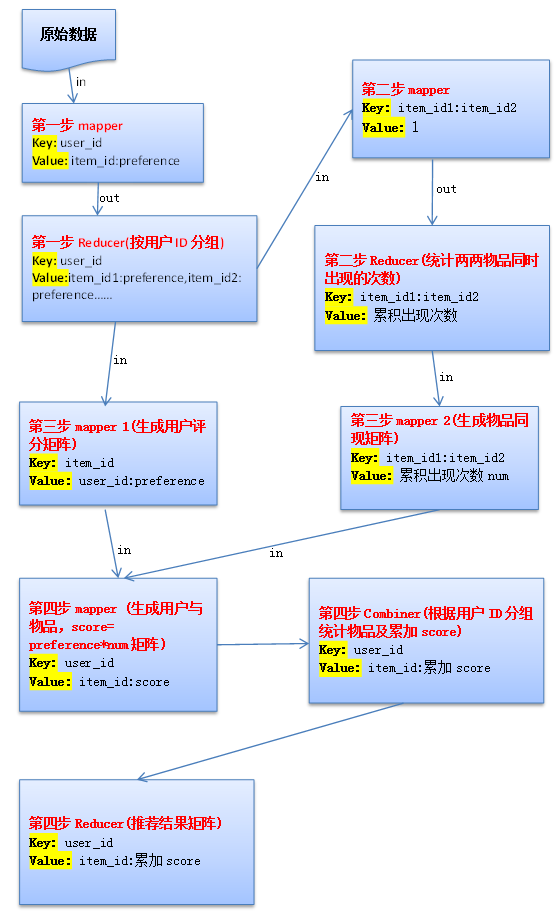


第二个mapper生成物品同现矩阵，结果如下：



第四步：做矩阵乘法，推荐结果=物品的同现矩阵A \* 用户对物品的评分矩阵B





1,Hadoop是离线计算，基于磁盘，每次运算之后的结果需要存储在HDFS里面，下次再用的话，还需要读出来进行一次计算，磁盘IO开销比较大。底层基于HDFS存储文件系统。适用于离线数据处理和不需要多次迭代计算的场景,并且Hadoop只有Map和Reduce两种接口，相对于Spark来说太少了。

2,Spark是内存计算框架，适用于多次迭代的计算模型，诸如各种机器学习算法 ,Spark里面的一个核心的概念就是RDD，弹性分布式数据集。Spark支持内存计算模型，用户可以指定存储的策略，当内存不够的时候，可以放置到磁盘上。并且Spark提供了一组RDD的接口，Tran敏感词ormations和Action。Tran敏感词ormations是把一个RDD转换成为另一个RDD以便形成Lineage血统链，这样当数据发生错误的时候可以快速的依靠这种继承关系恢复数据。Action操作是启动一个Job并开始真正的进行一些计算并把返回的结果可以给Driver或者是缓存在worker里面。

3,MPI是消息传递接口，可以理解为是更原生的一种分布式模型