

**课 程 实 验 报 告**

**课程名称： 大数据算法综合实践**

**计算机科学与技术学院**

**目录**

[实验1： 3](#_Toc163668107)

# 实验1： 大规模图数据中三角形计数算法的设计与性能优化

**1.1 实验概述**

本实验旨在通过大规模图数据中三形计数算法的设计与性能优化，帮助学生深⼊理解图计算系统的工作原理和性能优化机制，并学会使图计算框架进行大规模图数据分析和处理。通过此实验，学生将能够掌握图计算的基本概念、编写比较复杂的图算法程序并进⾏性能调优。

**1.2 实验内容**

1.2.1算法一

本实验要求在给定服务器平台，以及数据集上实现三角形计数（Triangle

Counting，TC）算法，调试并获得最高的性能。三角形的定义是一个包含三个顶点的子图，其中顶点两两相连。算法应着重讨论简单无向图的情形，即将重边（multi-edge）看成一条边，同时应不考虑 loop（顶点指向自己的边）。如在上图中，若存在顶点 b 到 c 的两条边，则应忽略其中一条，这样，结果仍然是找到 2 个三角形。在本实验的数据集中，也应将有向图看成是无向图。在本实验中，同学们需要重点考虑如何在大规模图数据中，精确计算三角形的个数。

我们这里在大数据处理实验的虚拟机基础上，尝试使用scala来编写程序，借助hadoop大数据平台来处理数据，使用graphx库来实现图计算。

对于最基础的TC算法，我们主要是要将文件内的数据转换成无向图。这里我们使用以下代码来实现无向图的创建。如图1.1所示：

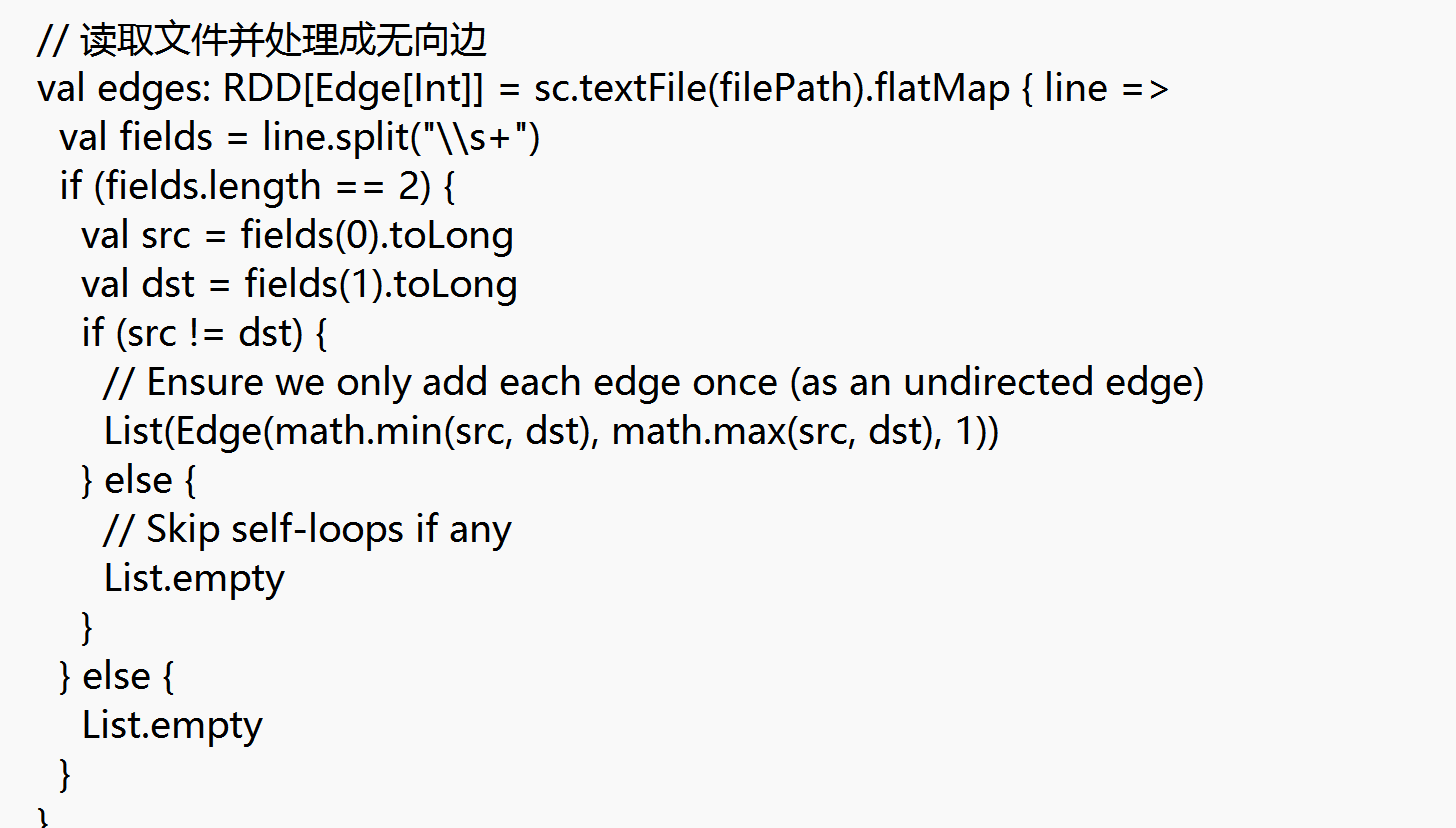


图1.1 无向图的创建

在flatMap中处理每一行的数据。将每行拆分为src和dst，并将它们转换为Long类型。使用math.min和math.max确保每条边都只添加一次，无论其顺序如何，从而确保图中的边是无向的。如果src等于dst，则跳过自环（自己连接自己的边）。接下来我们创建Edge RDD，将处理后的边列表作为flatMap的输出，最终形成RDD[Edge[Int]]，其中每个Edge对象代表了图中的一条无向边。

在转换完无向图后，我们使用Graph.fromEdges(edges, 1)根据处理后的边数据构建图数据结构。接着直接调用graphx库中的graph.triangleCount().vertices计算每个顶点参与的三角形数量。最后将答案写入result.txt文件中即可。如图1.2所示。由于我们每个三角形被计算了三次，所以我们需于/3来求出正确答案。

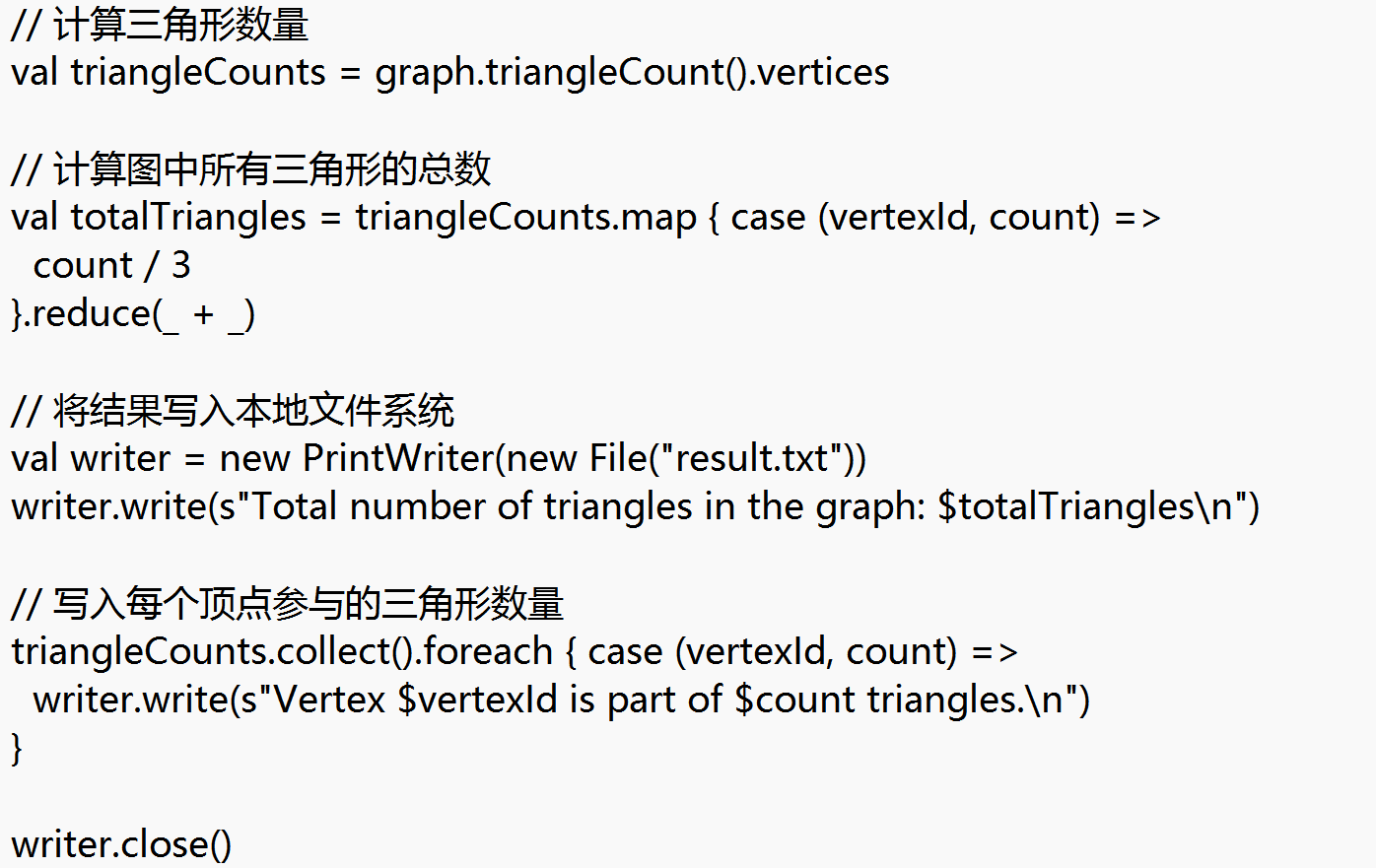


图1.2 计算三角形数量

1.2.2优化算法

优化想法1：我们需于对数据预处理部分进行优化。首先我们可以将这些数据分配到的不同节点上的这个算法进行优化，保证数据均匀分布在集群之中。这里我们使用repartition方法来根据并行度重新分区数据，并且使用cache来缓存RDD，以便后续来快速访问数据。由于我们需要将数据在存储空间上多次计算使用，所以我们还需要来压缩数据，以实现数据在磁盘存储传输时加快速度，这里我们使用mapPartitions函数。

优化想法2：我们对图处理算法进行优化。在构建图的时候，我们采取二维边分区，使用partitionBy的函数来减少跨分区的通信开销和优化计算效率。

优化想法3：我们使用并行处理来通过进程调度池以加快程序的计算速度。这里可以使用sc.setLocalProperty("spark.scheduler.pool", "pool1")

整体优化代码部分如图1.3所示：



图1.3 优化代码

我们将上述代码打包运行，会有以下的结果：

没有优化的结果如下所示：

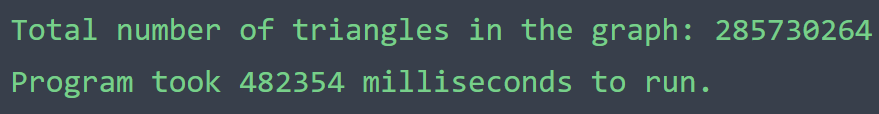


图1.4 未优化的结果

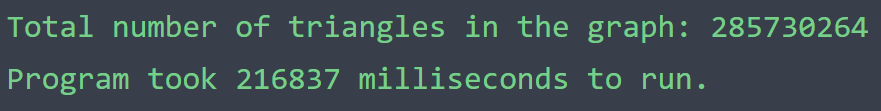


图1.5 优化后的结果

比较发现，我们可以知道这里的优化效率大概是55.05%，优化效果显著

**1.3 实验小结**

在本次实验中，我们主要用scala来实现了tc算法，由于是借助graphx库中的triangleCounts函数来计算的，所以我们对程序优化主要集中在spark程序运行资源的分配，数据分配到节点的调度算法，以及对图数据划分的处理和并行处理这几部分内容。在自己的虚拟机上运行中，我发现对于第一个小的数据集可以正确的运行，但是对于大的数据由于数据文件过大，虚拟机会运行错误。这就要求我们对自己的程序进行优化，这也就是本次实验的主要内容。

本次实验启发我，在今后的大数据项目中，我们会遇到同类型的问题。我们不仅需要更好的服务器，更好的运行资源来实现大数据算法，还需要我们对大数据中的算法来进行优化。从内存调度，从并行处理，从图数据处理，从数据分配的多个角度来进行优化算法。

# 