计算机体系结构课程实验

2020.4

1. 实验目的

(1). 学习并行计算引擎 Spark，了解编程语言 Scala 和编程环境 IDEA。

(2). 学习使用开源图计算平台 GraphX，了解 Pagerank 算法的实现流程与优化方式。

(3). 练习测量 GraphX 平台下的内存/缓存数据分析，如内存使用率等。

2. 实验要求

使用 GraphX API，将源数据集数据抽象为图数据格式（实验 1），分别使用 SSSP 算法和 pagerank 算法处理 **Wikipedia 图和** google 图（实验 2）。在这个过程中，尝试分时记录内存使用率、cache 命中率，并作图表分析相关现象（实验 3）。

3. 实验环境配置

(1). 搭建 Java 开发环境，配置环境变量

http://www.oracle.com/technetwork/java/javase/downloads/index.html

(2). 安装 hadoop，Spark 并配置

Ubuntu 环境搭建：

Hadoop 配置：http://www.cnblogs.com/kinglau/p/3794433.html

Spark 配置：https://www.cnblogs.com/lijingchn/p/5573898.html

Windows 环境搭建：

https://blog.csdn.net/u013963380/article/details/72677212

(3). 安装 IDE 环境 Intellij IDEA

https://www.jetbrains.com/idea/download/

(4). 引入 graphX，学习 graphX API 的使用方式

https://spark.apache.org/graphx/

https://spark.apache.org/docs/latest/graphx-programming-guide.html

4. 实验数据说明

(1). 数据集来自 **SNAP networks**，网址为 http://snap.stanford.edu/data/index.html，数据集下载解压后得到 txt 文件。

(2). **Wikipedia 图**是由 7115 个节点和 103689 条边组成，**Google 图**是由 875713 个节点和 5105039 条边组成。

(3). 每一行代表一条边，FromNodeId 是边的起始节点 ID，ToNodeId 是边的终止节点 ID，方向代表他们之间的引用关系，A->B 代表 A 投票给 B，或 A 引用了 B，即 A 认为 B 很重要，将 A 的一部分重要性赋予 B。

5. 实验流程

(1). 实验 1(20%)：graphX API 练习

- 使用 Pregel 实现 SSSP(single source shortest path)：参考 Pregel API, GraphOps
- 数据预处理：参考 Vertex and Edge RDDs，GraphLoader.edgeListFile

(2). 实验 2(40%)：使用 SSSP 和 PageRank 处理 Wikipedia 图并测量内存使用率和 cache 命中率等内存参数。

- 下载数据集 http://snap.stanford.edu/data/wiki-Vote.html
- 下载数据集 http://snap.stanford.edu/data/web-Google.html
- 按照**实验 1** 中的数据预处理方法预处理所给的实验数据
- 参考使用 aggregateMessages，org.apache.spark.graphx.lib
- **可使用性能检测工具，Spark 的 SparkBench，Windows 下如 perfmon 等，ubuntu 下如 nmon 等，工具没有限制。**
- **根据实际情况测量内存使用率和 cache 命中率等内存参数，作分时图并分析图表，多关注峰值，低估，快速升高，快速下降等。**

(3). 实验 3(20%)：处理 Google 图并测量内存使用率和 cache 命中率等内存参数。

- 在实验 2 中之上进行重运行，内存不够可能涉及到数据分割。
- 参考 PartitionStrategy，Graph.partitionBy，了解 graphlab, powergraph 原理
- 参考 https://www.cnblogs.com/wei-li/p/graphx.html
- **其他与实验 2 相似，加入不同数据集的对比实验。**

(4). 问答题(20%)：

叙述实验 2 和**实验 3** 代码运行流程与算法设计，**比较实验 2** 和**实验 3** 在运行时间、空间占用方面的差异，并结合自己的实际作图进行综合阐述。最后，总结自己在实验过程中遇到的难题和解决方式。

6. 实验提交与评分标准

(1). 需要将代码(scala 文件和 readme.txt)和实验报告(2 页及以上)一并打包(zip 或 rar)并提交到 canvas，每人提交一份。**打包文件命名方式（课程序号_作者学号_作者名称.rar）**。

(2). 评分依据：报告整洁度 30%，实验设计的清晰度 30%，实验结果对比分析的完整度 40%（图表、解释说明等）。

7. 一些参考：

(1). Pagerank 的原理

PageRank,即网页排名，又称网页级别、Google 左侧排名或佩奇排名。是 Google 创始人拉里·佩奇和谢尔盖·布林于 1997 年构建早期的搜索系统原型时提出的链接分析算法，目前很多重要的链接分析算法都是在 PageRank 算法基础上衍生出来的。

对于某个网页 A 来说，该网页 PageRank 的计算基于以下两个基本假设：**①数量假设**：在 Web 图模型中，如果一个页面节点接收到的其他网页指向的入链数量越多，那么这个页面越重要。**②质量假设**：指向页面 A 的入链质量不同，质量高的页面会通过链接向其他页面传递更多的权重。所以越是质量高的页面指向页面 A，则页面 A 越重要。

执行步骤：1).在初始阶段：网页通过链接关系构建起 Web 图，每个页面设置相同的 PageRank 值，通过若干轮的计算，会得到每个页面所获得的最终 PageRank 值。随着每一轮的计算进行，网页当前的 PageRank 值会不断得到更新。2).在一轮中更新页面 PageRank 得分的计算方法：在一轮更新页面 PageRank 得分的计算中，每个页面将其当前的 PageRank 值平均分配到本页面包含的出链上，这样每个链接即获得了相应的权值。而每个页面将所有指向本页面的入链所传入的权值求和，即可得到新的 PageRank 得分。当每个页面都获得了更新后的 PageRank 值，就完成了一轮 PageRank 计算。

简单计算：

Arvind Arasu 在《Junghoo Cho Hector Garcia - Molina, Andreas Paepcke, Sriram Raghavan. Searching the Web》更加准确的表达为：

$$\mathrm{PageRank}(p_i) = \frac{1-q}{N} + q\sum_{p_j} \frac{\mathrm{PageRank}(p_j)}{L(p_j)}$$

$p_1, p_2, \ldots, p_N$是被研究的页面，$M(p_i)$是p_i链入页面的数量，$L(p_j)$是p_j链出页面的数量，而N是所有页面的数量。

PageRank值是一个特殊矩阵中的特征向量。这个特征向量为：

$$\mathbf{R} = \begin{bmatrix} \mathrm{PageRank}(p_1) \\ \mathrm{PageRank}(p_2) \\ \vdots \\ \mathrm{PageRank}(p_N) \end{bmatrix}$$

R是如下等式的一个解：

$$\mathbf{R} = \begin{bmatrix} (1-q)/N \\ (1-q)/N \\ \vdots \\ (1-q)/N \end{bmatrix} + q \begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & \cdots & \ell(p_1,p_N) \\ \ell(p_2,p_1) & \ddots & & \\ \vdots & & \ell(p_i,p_j) & \\ \ell(p_N,p_1) & & & \ell(p_N,p_N) \end{bmatrix} \mathbf{R}$$

如果网页i有指向网页j的一个链接，则

$$\sum_{i=1}^{N} \ell(p_i,p_j) = 1,$$

否则$\ell(p_i,p_j) = 0$。

摘自 https://blog.csdn.net/hguisu/article/details/7996185

(2). 图计算框架 GraphX

GraphX 是 Spark 中用于图(graph)和图并行计算的一个新组件。在高层次上，GraphX 通过引入一个新的图抽象，扩展了 Spark RDD:一个有向多重图，属性被附加到每个顶点和边缘。为了支持图计算，GraphX 公开了一组基本操作符(例如，subgraph、joinVertices 和 aggregateMessages)，以及 Pregel API 的一个优化变体。此外，GraphX 还包含了越来越多的图算法和图构建器，用于简化图计算任务。

摘自 https://spark.apache.org/docs/latest/graphx-programming-guide.html