哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

大数据分析

实验三

(2019 年度春季学期)

姓　　名　朱明彦
学　　号　1160300314
学　　院　计算机学院
教　　师　杨东华、王金宝

计算机科学与技术学院

目录

实验三 图数据分析

第 1 章 实验目的

掌握大图数据计算平台的原理、架构和工作机制，理解大图计算平台的各项功能，包括数据载入、大图数据划分原理和大图计算。能够编写大图数据计算平台中的常用图分析方法 (PageRank, SSSP) 程序。

第 2 章 实验环境

- Ubuntu 16.04.5
- Java 1.8.0_181

第 3 章 实验过程及结果

3.1 项目架构

3.1.1 `Master` 及其子类

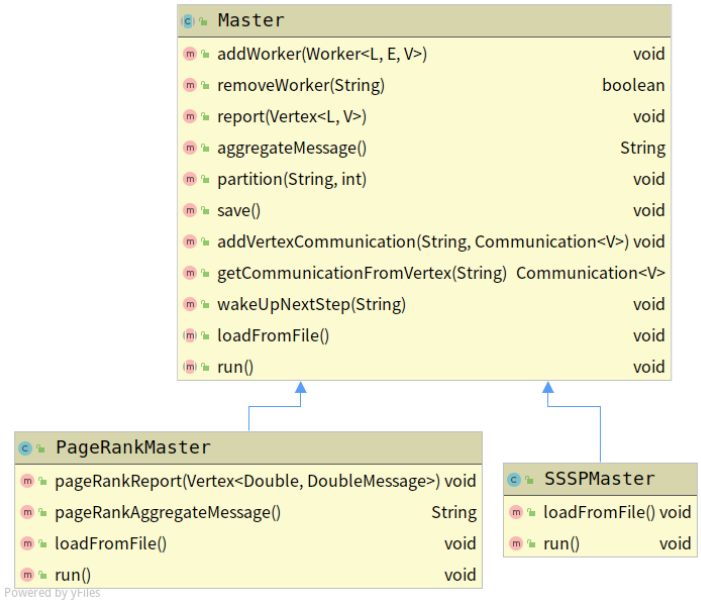

图 3.1: `Master` 及其子类 UML 图

对于 `Master` 与其子类，模拟的是 BSP 模型中 Master 节点。其中需要处理的问题主要有完成所有 worker 中每个顶点的计算、在所有的顶点之间进行通信两个功能。在实现时以 `Master` 为抽象类，其中最为重要的抽象函数即 `run()`，对于不同的应用如 SSSP 和 PageRanking，只需要重写 `run()` 函数来完成其相应逻辑即可，`master` 包中的继承关系如图3.1所示。

对于 `Master` 类，其定义时共有三个泛型参数，分别为 `L`，顶点类型；`E`，边类型；`V`，消息类型，对于 SSSP 问题，其只需将对应的泛型参数改为 `Integer`，`Integer`，`IntMessage` 即可。对于 `Master` 类中设计的函数及其功能如下：

- `addWorker`，`removeWorker` 分别用于在 `Master` 类中增加删除 `worker`。
- `report`，`aggregateMessage` 主要用于全局的消息的 aggregate，具体参见节。
- `partition` 用于将图进行划分，其中的两个参数分别为图源文件路径以及划分的数量。
- `save` 用于将 `partition` 的结果存储到外存中，每个划分一个源文件，每一条输出类似 $u, v_0, v_1 \ldots, v_n$，其中 u 为源点，v_i 为 u 的所有出边的目的点，分隔符为 `tab`。
- `addVertexCommunication`，`getCommunicationFromVertex` 分别用于增删每个顶点对应的 `communication`，关于 `communication` 详见3.1.4节。
- `wakeUpNextStep` 是将发送消息的接收节点在下一轮唤醒（转为 active 状态）。

对于 `SSSPMaster` 和 `PageRankMaster` 而言，其 `run` 函数具有很大的相似性（**即进行 BSP 过程的模拟**），本质上都是一轮轮执行 Worker，直到所有的 Worker 都结束，再将每个顶点的计算结果输出至外存中进行保存；**不同在于对于 PageRanking 应用，只需要将所有的 `Vertex` 保持 active 状态，执行指定轮数即可，而对于 SSSP 应用则对每个 Worker 的状态进行判断，并在下一轮开始之前将利用 `wakeUpNextStep` 唤醒的 worker 转入工作状态**。

3.1.2 `Worker` 及其子类

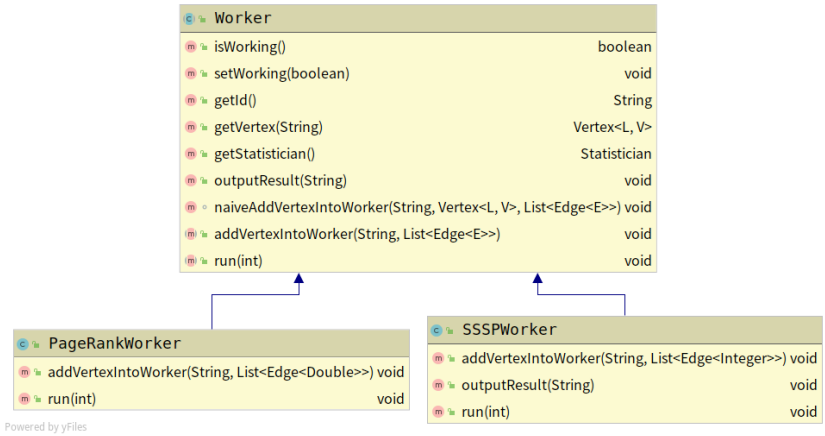

图 3.2: `Worker` 及其子类 UML 图

对于 `Worker` 及其子类，模拟的是 BSP 模型中的 Worker 节点，在其中需要维护每个 Vertex 的出边、消息队列（包括两个接收队列和一个发送队列，在本项目中利用 `Communication` 来实现，详见3.1.4节）。并且在接收到 Master 发送的消息后，串行在每个节点上执行 `compute` 函

数，并将有更新的 Vertex 的产生的消息发送到对应的出边目的顶点即可。worker 包中的继承关系如图3.2所示。

对于 Worker 类，其定义时共有三个泛型参数，分别为 L，顶点类型；E，边类型；V，消息类型，对于 SSSP 问题，其只需将对应的泛型参数改为 Integer，Integer，IntMessage 即可，PageRanking 问题同理。在 Worker 类中设计函数及其功能如下所示：

- run(int) 用于在 Worker 上对其中的每个 Vertex 串行的执行 compute 函数。
- 一系列的 getter 和 setter 函数。

对于 SSSPWorker 而言，其 run 函数需要对其中每个当前接收队列不为空的 Vertex 执行 compute 函数，并将执行后为 active 的顶点，将更新后的信息发送至所有的出边的目的点；而对于 PageRankWorker 而言，其 run 函数相对简单，只需要在第 0 轮 super step 利用对应的 aggregator 统计顶点的数目，在剩余轮数中对所有的顶点执行 compute。

3.1.3 Vertex 及其子类

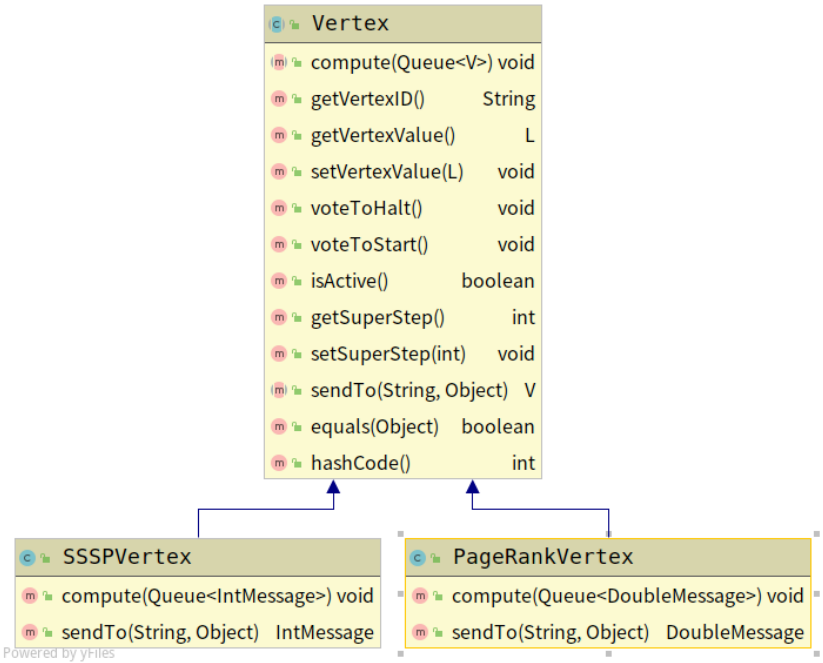

图 3.3: Vertex 及其子类 UML 图

对于 Vertex 及其子类，模拟的是 BSP 模型中的 Vertex，其中基本类（抽象类）Vertex 不参与运算，对于不同的应用分别重写 compute 函数即可。其中 Vertex 有 2 个泛型参数，分别是 L，顶点类型；V，消息类型。vertex 包中的继承结构如图3.3所示。

Vertex 中设计的函数及其功能如下：

- compute 即该顶点的需要执行的计算函数，对于不同的应用需要重写。

- `sendTo` 用于产生该顶点需要发送的消息并将其返回，使用对应的 worker 进行发送。
- 一系列 `getter` 和 `setter` 函数等。

对于 `SSSPVertex` 的 `compute` 函数需要比较所有的接收消息中，是否存在比当前顶点的属性更小的信息，如果是则更新并将该顶点转为 active 状态，否则则将该顶点转为 inactive 状态。对于 `PageRankVertex` 的 `compute` 计算，只需要在其中计算所有接收消息中值的和，然后在 worker 中计算 $value = 0.15/verticesNumber + 0.85 \times sum$ [1]。

3.1.4 `Communication`

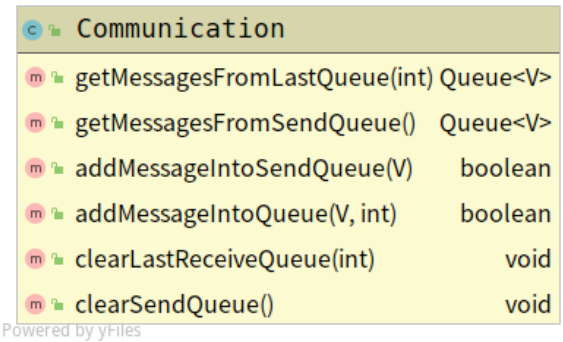

图 3.4: `Communication`UML 图

项目中实现 Communication 功能使用 `Communication` 类实现，其中有一个泛型参数 `V`，消息类型，有两个接收队列和一个发送队列，在使用时根据当前 superStep 的轮数确定使用哪一个队列作为当前的接收队列（模 2 的结果进行选择）。**通过 Master 和 Worker 维护每个 Vertex 和其对应的 `Communication` 实例，来实现对于发送和接收消息过程的模拟。**

3.1.5 `Combiner` 类及其子类

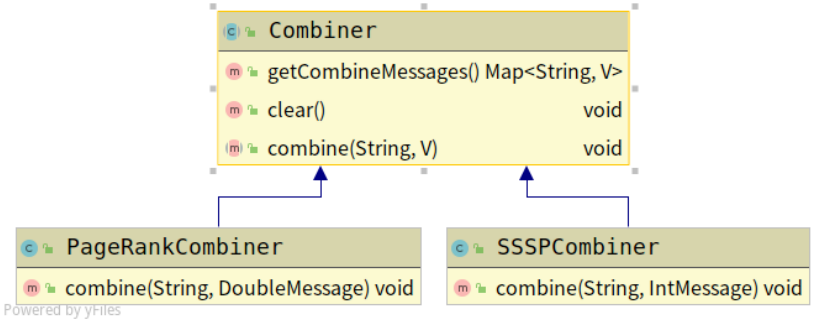

图 3.5: `Combiner` 及其子类 UML 图

`Combiner` 及其子类用于将多条发送至同一 Vertex 的信息进行合并，对于不同的应用可以有不同的合并方式，在相应的子类中重写 `combine` 函数即可，然后利用其中的维护的接收 Vertex

和信息的 Map 来实现减少通信数量的作用。`Combiner` 与 `Worker` 结合使用，每个 `Worker` 中可以有至多一个 `Combiner`，其继承关系如图3.5所示。

对于不同应用的 `combine` 函数的作用不同，分别有如下功能，

- 对于 `SSSPCombiner` 而言，`combine` 函数对于发往同一个目的顶点的信息，**取其中较小的信息**保留。
- 对于 `PageRankCombiner` 而言，`combine` 函数对于发往同一目的地的信息**取和**保留。

3.1.6 `Aggregator` 及其子类

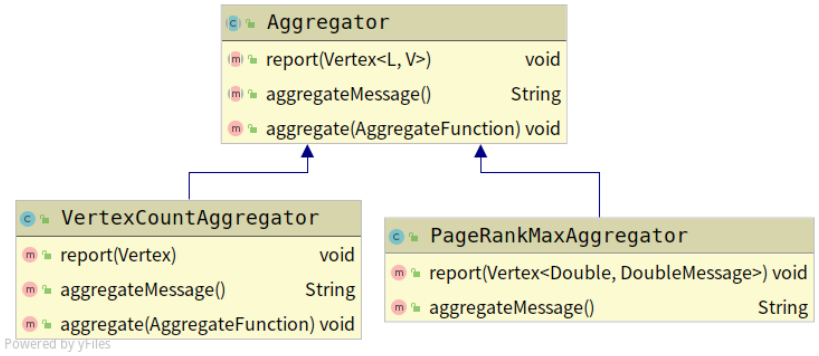

图 3.6: `Aggregator` 及其子类 UML 图

`Aggregator` 及其子类用作统计全局信息，在本项目中主要使用其进行全局节点数目的统计和在执行完 PageRanking 后进行 Rank 的 TopK 统计。继承关系如图3.6所示，由于使用到全局信息，所以 `Aggregator` 与 `Master` 节点结合使用，并且对于每个 `Master` 至多与 1 个 `Aggregator` 结合使用。

对于两种不同的应用，分别有不同的函数实现，

- `VertexCountAggregator` 用于统计全局的节点数目，故在 `report` 对其维护的 Vertex 数量进行自增即可。
- `PageRankMaxAggregator` 用于全局的 PageRank 中 TopK 的统计，所以在其中维护了一个大小为 K 的小顶堆，在不足 K 个时将每个顶点均加入堆中，达到 K 个时则判断当前堆顶元素的 Rank 是否比新来的 Vertex 的 Rank 高，如果新来的更高则弹出堆顶元素，加入新来元素，否则直接舍弃堆顶元素即可。

3.2 图分析算法实现

SSSP 的实现和 PageRank 的实现区别主要在 `Worker` 和 `Vertex` 实现的区别，接下来分别进行说明。

3.2.1 SSSP 的实现

`SSSPVertex` 在接收到消息时将自身转为 active 状态，但通过消息队列计算当前的最短距离没有发生改变，就再将自身转为 inactive 状态。关于 `SSSPWorker` 的区别，在3.1.2节有详细说明。

3.2.2 PageRank 的实现

`PageRankWorker` 只需在指定轮数之内保持 active 状态，将更新后的信息发送给所有的出边目的顶点即可，到达制定轮数之后转为 inactive 即可。关于 `PageRankWorker` 的具体实现，见3.1.2节说明。

第 4 章 实验心得

总的来说，模拟 Pregel [1] 的难度不大，而在于理清各个类之间的关系，以及模拟 BSP 模型时注意模拟各个方法的调用顺序。建议在之后的实验里可以适当增大该实验的难度，比如多线程并行模拟 BSP 模型，线程数为 worker 数量。

参考文献

[1] Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N., & Czajkowski, G. (2010, June). Pregel: a system for large-scale graph processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (pp. 135-146). ACM.