2018 Introduction to Massive Data Analysis

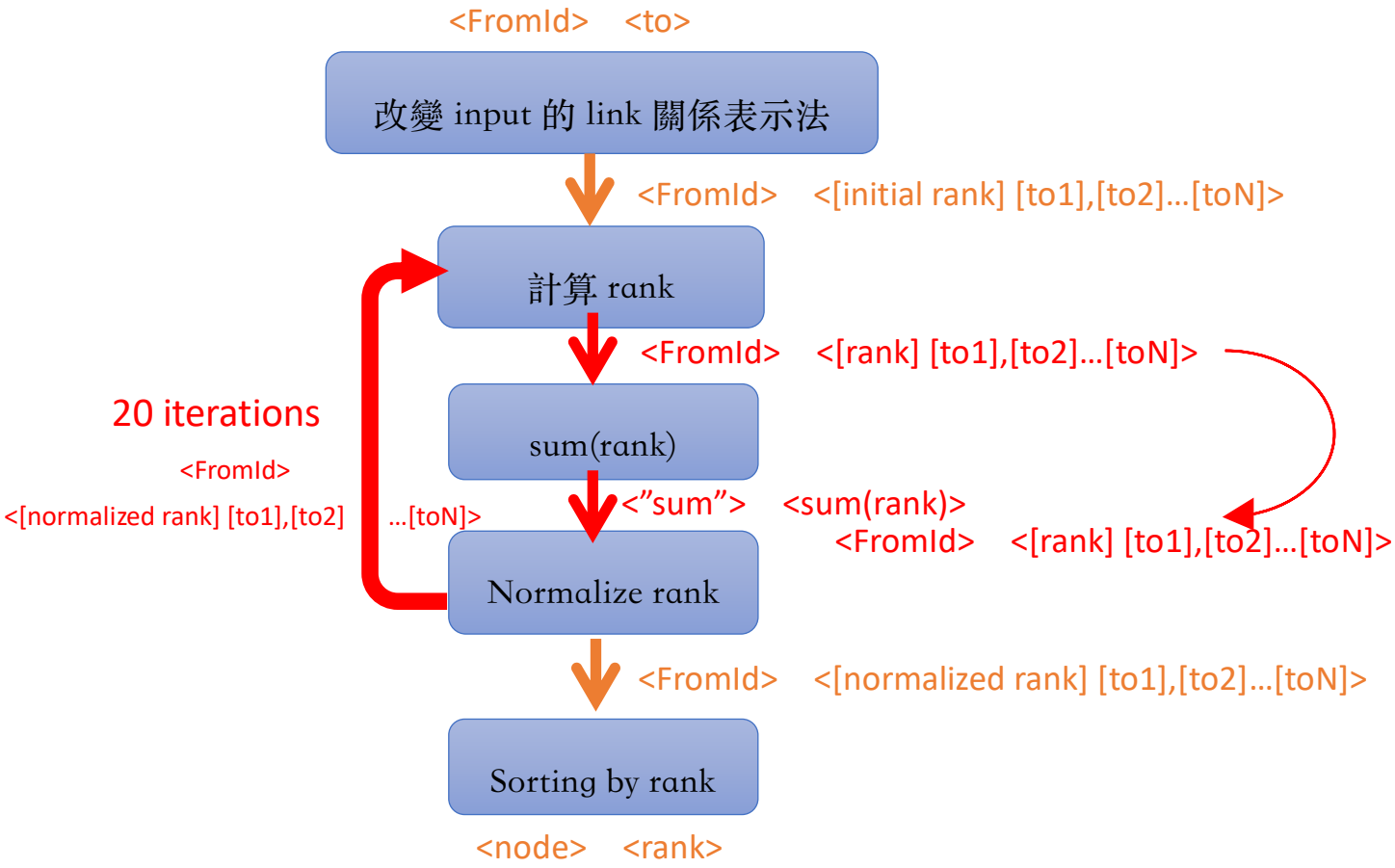
Assignment 1

107062558 賴怡惠

Flow chart

[noted]一個框框是一組 mapreduce

[noted] <key><value>

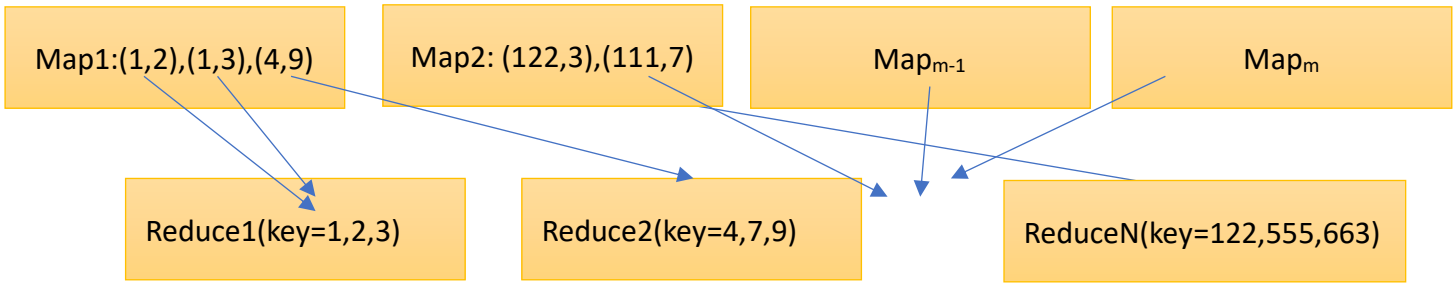


Concept Explanation

1. **改變 input 的 link 表示法**

   根據公式 $\sum_{i\to j} \beta \frac{ri}{di}$ 需要知道來源的 rank，也需要知道來源 rank 共有多少

outlinks。而一開始的 p2p-Gnutella04.txt 把同個來源去到不同 node 的資訊分成一行一行寫，如此一來就必須讀完整個檔案才可以知道有多少某個來源有多少 outlinks。但是 map-reduce 的設計是完全平行化的，也就是說，每個 map 無法讀到完整檔案，所以有可能同個來源去到不同 node 的資訊被分到兩個 file 內，故得到的 outlink 數不完整，所以我們首先需要把資料整合成: [fromId] [rank] [to1],[to2],[to3]…[toN]的形式。



以下步驟開始 iterate 20 次

2. **計算 rj = $\sum_{i\to j} \beta \frac{ri}{di}$ + (1- β)/N**

雖然公式是合在一起的，但是若在 mapper 兩項一起做，無法知道**(1- β)/N** 是不是第一次加到 rj (由上個步驟的 output 可知此步驟的 input 格式是一個 fromId 對應到多個 to，故有可能 to 的其他"來源"被分到其他 mapper，而 mapreduce 的架構是完全平行，所以 mapper 之間是不會溝通的！故我們無法在 mapper 直接加上第二項)。Mapper 的 input 也就是步驟一的 output ; Mapper 的 output 是 ri 對於 rj 的貢獻值，output 形式如下：[to] [rank contribution from fromId]。Reducer 的 input 需要有兩種，第一種是 [to] [rank contribution from fromId]幫助我們計算 rank，第二種是[fromId] [to1],[to2],[to3]…[toN]，因為此步驟開始是 iteration，若在某一個步驟遺失 rank 或 link 之間的資訊，會造成 mapreduce 的 input out 格式不固定，就無法 iterate，所以計算 rank 時，也需要 link 的資訊！故不僅要在 mapper 加入 link 間關係，reducer 也需要 output 此資訊，所以 reduce 的 output 為[selfId] [selfRank] [to1],[to2],[to3]…[toN]。

3. **計算$\sum_{\forall j} rj$是否為 1**

為符合 stochastic 原則，$\sum_{\forall j} rj$不為 1 表示有 dead end，我們必須把公式轉換成$\forall j \;\; rj' = \text{rj} + \frac{1-S}{N} \; where \; S = \sum_{\forall j} rj$。為了計算 S 以改變上個步驟的 selfRank，首先我建了一個 mapper， input 即為上一個步驟的 output : [selfId] [selfRank] [to1],[to2],[to3]…[toN]，在此不需要再存進 link 間的關係，只需要 rank 即可，而 map 的方式為利用一個指定的 key，在此為一個 text 固定叫 sum ，然後其 value 為 rj : output ["sum"] [rj]，由此一來在 reduce 階段，每個 key 都會被 map 到同一個 reducer，藉由此方法幫助我

aggregate rj 的值，而 reduce 的 output 只有一個，sum S。

4. **Normalize**

   由於此份 data 有存在 dead end，所以需要 normalize，利用我們剛算好的 S，對步驟二 output file 把 rank value 的那一項做 $\forall j\ rj' = \text{rj} + \frac{1-S}{N}\ where\ S = \sum_{\forall j} rj$ 的轉換。Mapper 的 input 即是步驟二的 Output，每個 mapper 對該項加$\frac{1-S}{N}$即可，而資料已經是一個 node 就存成一行，不需要再做 reduce，所以此步驟不需要寫 reducer，只需要意思一下在 job 裡定義好 reducer 的 output key,value 即可。

   到此步驟為止是一個 iteration 要做的 map reduce

5. **Sort**

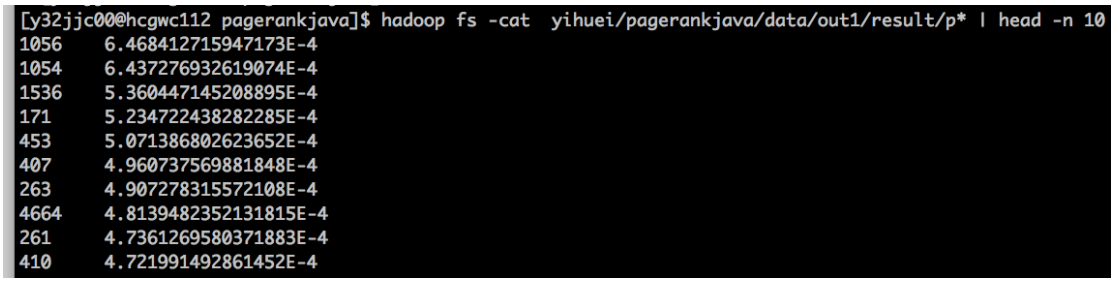
   由於 spec 要求需要依照 rank 做由高到低的排序，在此我針對第 20 個 iteration normalized output 做 sort，而 sort 是需要全部的 data 一起排序才有用，不能根據某個 worker 拿到的那一組資料進行 sort 而已，所以我在 job 的地方設定只有一個 reducer!

   mapper 的工作是把資料切割，因為上個步驟的 output 是 [selfId] [selfRank] [to1],[to2],[to3]…[toN]，我們現在不需要 link 間的關係了，所以我就把 selfId 和 selfRank 切出來，作為 value 和 key 傳給 reducer。

   Job 內我設了 setSortComparatorClass，所以 mapper 會先把資料傳給 comparator，sorting 完才會傳給 reducer。

   而 Reducer 的工作就很簡單了！只需要把 comparator 的資料 output 寫成檔案即可！需要注意的是 mapper 傳給 comparator 是利用 rank 做排序，所以 rank 是 key，而我們需要的 output 形式是[node] [rank]，所以要把 node 當 key;rank 當 value 再輸出檔案。

Result



[noted]資料夾內有附完整排序過後的 output