HW2 Report

105062581 蕭文逸

• Top Ten vertices sorted by rank: (20 iteration)

```
1056
0.000632

1054
0.000629

1536
0.000524

171
0.000512

453
0.000496

407
0.000485

263
0.000480

4664
0.000470

261
0.000463

410
0.000462
```

• Implementation:

基本上分成兩步驟(兩次 map reduce),第一次是整合 Page Rank 值和 Link 關係,第二次時才會有矩陣運算。

== Phase1 ==

- Mapper1

把 edge 和 vertex 關係改寫成需要的模式。以範例來看,例如:

- 1 2
- 1 3
- 2 4
- 3 1
- 3 4
- 3 5
- 5 1
- 5 4

把它寫成這樣的 pair: <k, v> = <1, (2, 3) >, <2, (4) >, <3, (1,4,5) >等等

同時把 Page Rank 值也 MAP 出去: <k, v> = <0, PR>

這樣設計,在下一階段時就可以知道 Out Degree 是多少,才能計算 M 的值。

- Reducer1

不做任何事

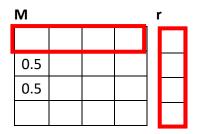
- Mapper2

上一階段是類似預處理的步驟,這一階段,是要模擬矩陣乘法,把需要用到的 值賦予相同 key,方便在 Reducer 計算。

以剛剛第一個 pair 為例,這一階段的 mapper 接收到<1, (2, 3) >時,會多分出兩條 pair: <2,(1, Value)>和<3,(1, Value)>,其中 Value 值是 1 除以 2(Out-degree)。 然後剛剛的 Pagerank pair<0, PR>則會變成

<1, (1, PR)>, <1, (2, PR)>, <1, (3, PR)>, <1, (4, PR)>

這是因為新的 Pager rank 值所用到的東西如下圖:



新的 PR 要用到的是("從 i 射進來的 1/degree" * PR(i))

其中 A 是由 M 矩陣來的,在這邊是先為了把 M 矩陣的值求出,等等在 Reducer 階段會求出 A 並相加。

- Reducer2

 $A = beta * M + (1-beta) * (1/N)_{NxN}$

所以這一階段所以 key 相同都 group 在一起。接下來就是要總和算出這一輪 PR。以上個例子繼續來看。

新的 PR(1)將會是

[beta * <1, (1, Value)> + (1-beta) * (1/N)] * <1, (1, PR)>

Δ

r

由於 piar 會少(即 M==0 的地方),所以要寫一些判斷式補足。

<< Deadend>>

預設是 1(即有 Deadend 手動),所以算完 PR 後還要在算 S 並重新校正,才是真正的 PR

== Iteration ==

上面兩 PHASE 會 Iteration 20 次。