CprE 419 Lab 3: Graph Processing using MapReduce

**Shuo Wang**

# Experiment 1 (40 points):

# output the top ten patents and their significance

In this task I used three map-reduce rounds:

Round 1: for each A, get all the patents A cites and all the patents A is cited by.

Input: V1 V2

After map: <V1, V2 cite>, <V2, V1 citedby>

After reduce: <V1, cite V2 V3 …|citedby V4 V5 …>



Round 2: find all the 1-hop and 2-hop citations for A

Input: V1, cite V2 V3 …|citedby V4 V5 …

After map: <V2, V1 citedby V4 V5 …>

<V3, V1 citedby V4 V5 …>

….

<A, B citedby C D…>

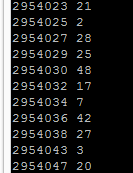
// B is the 1-hop citation of A and C D … are the 2-hop citations of A

After reduce: <A, number of 1-hop and 2-hop citations>

// put all the 1-hop and 2-hop citations of A together

// remove the duplicates

// count the number



Round 3: find the top 10 patents with most citations

Input: A, number of 1-hop and 2-hop citations

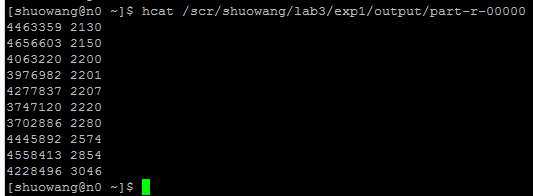
After map: <uniform key, A number of 1-hop and 2-hop citations >

// push everything to a single reducer

After reduce: top ten patents

// while scanning all the patents,

// use a TreeMap to keep tracking the top 10 largest frequencies



So the most frequent cited patent is ID 4228496: Multiprocessor system, invented by Katzman; James A. in 1976.

# Experiment 2 (40 points):

# compute the global clustering coefficient

In this task I used three map-reduce rounds:

Round 1: get all the neighbors of A and the number of triplets with A in the middle

Input: A B

After map: <A, B >, <B, A>

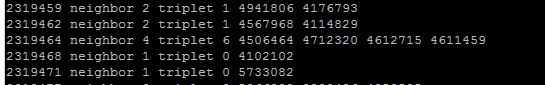
After reduce: <A, neighbor 2 triplet 1 B C>

// A has 2 neighbors: B C, and there is one triplet with A in the middle: BAC

// The number of triplets = neighbor #\*( neighbor #-1)/2;

// for example: B,C,D are the 3 neighbors of A,

// then there are 3\*(3-1)/2=3 triplets with A in the middle are BAC,BAD,CAD



Round 2: finds all the triangles having A for each A

Input: A, neighbor 2 triplet 1 B C

After map: <B, A neighbor 2 triplet 1 B C >

<C, A neighbor 2 triplet 1 B C >

….

<A, B neighbor n triplet m C D…>

// B is the neighbor of A and C D … are the neighbors of B

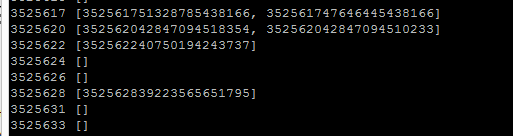
After reduce: <A, the triangles having A>

// for each <A, B neighbor n triplet m C D…>

// check whether the other neighbors of A (except B),

// are the also the neighbors of B

// remove the duplicates



Round 3: counts all the triangles in the network

Input: A, [ABC, ACD]

After map: <uniform key, [ABC, ACD]>

// push all the triangles to a single reducer

After reduce: number of triangles

// put all the triangles together

// remove the duplicates



Round 4: counts all the triplets in the network

Input: A, neighbor 2 triplet 1 B C

// the output of round 1 is the input of round 4

After map: <uniform key, number of triplets>

// push all the numbers to a single reducer

// there is no duplicates

After reduce: number of triplets

// sum them all



The final answer:

the global clustering coefficient = 3 \* 2111096 / 335781273 = **0.018861**

Communication complexity:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Exp# | Round | Total map cost | Per reducer cost | Total reducer cost | Total M-R communication |
| Exp1 | 1 | Ө(#edges) | Ө(#edges **/** #patents) | Ө(#edges) | Ө(#edges) |
| 2 | Ө(#patents) | Ө(1) | Ө(#patents) | Ө(#patents) |
| 3 | Ө(#patents) | Ө(#patents) | Ө(#patents) | Ө(#patents) |
| Exp2 | 1 | Ө(#edges) | Ө(#edges **/** #patents) | Ө(#edges) | Ө(#edges) |
| 2 | Ө(#patents) | Ө(1) | Ө(#patents) | Ө(#patents) |
| 3 | Ө(#patents) | Ө(#patents) | Ө(#patents) | Ө(#patents) |
| 4 | Ө(#patents) | Ө(#patents) | Ө(#patents) | Ө(#patents) |

Exp1:

Round 1: The number of input atoms = #edges, and the in map-reduce process no information is missed. So the total maps cost, total reduce cost and total M-R communication are all linear functions of #edges. Each patent goes to a different reducer, so there are #patents reducers and the per reducer cost = Total reducer cost / number of reducers = Ө (#edges **/** #patents).

Round 2: The number of input atoms = #patents, and the in map-reduce process no information is missed. So the total maps cost, total reduce cost and total M-R communication are all linear functions of #patents. Each patent goes to a different reducer, so there are #patents reducers and the per reducer cost = Total reducer cost / number of reducers = Ө (#patents **/** #patents) = Ө (1).

Round 3: The number of input atoms = #patents, and the in map-reduce process no information is missed. So the total maps cost, total reduce cost and total M-R communication are all linear functions of #patents. There is only one reducer, so the per reducer cost = Total reducer cost / number of reducers = Ө (#patents **/** 1) = Ө (#patents).

Exp2:

Round 1: similar to Exp1.round 1

Round 2: similar to Exp1.round 2

Round 3: similar to Exp1.round 3

Round 4: similar to Exp1.round 3