CS7960 L19 : MapReduce | triangle count

MapReduce

M = Massive Data

Mapper(M) -> {(key,value)}

Shuffle({(key,value)}) -> group by "key"

Reducer ({"key,value_i}) -> ("key, f(value_i))

Can repeat, constant # of rounds

Given graph G=(V,E)

```
Assume |V|=n and |E| = m = n^{1+c}
  typical large graphs have c in [0.08, 0.5]
```

N(v) = neighbors of v

```
cluster coefficient cc(V)
  = fraction N(v), neighbors themselves
How dense a subgraph is
```

** need to find all triangles for each v in V**

```
(sequential)
for each v in V
 for each (u,w) in N(v)
  if (u,w) in E  -> Triangle[v]++
```

```
T = sum_{v in V} |N(v)|^2
 O(n^2) if some v N(v) = O(n)
```

(parallel)

Map 1: G=(V,E) -> (v,u),(u,v) for (v,u) in E

Reduce 1: (v, N(v)) -> ((u,w),v) s.t. u,w in N(v)

Map 2: -> ((u,w),v) (output of R1)

```
       -> ((u,w),$)  for (u,w) in E

Reduce 2: ((u,w),{v1,v2,v3,...vt,$?}
  iff $, then -> (vi,1/3)

Map 3: identity
Red 3: aggregate

:(  running time still max_{v in V} |N(v)|^2

--------
LiveJournal
80% reducers done in 5 min
99% reducers done in 35 min
some 60 minutes

---------------------------
Idea 1: count each triangle once, with lowest degree

(sequential)
for each v in V
 for each (u,w) in N(v)
  if deg(u) > deg(v) && deg(w) > deg(v)
   if (u,w) in E  -> {Tri[v]++,Tri[u]++,Tri[w]++}

In Reduce 1, add if condition.
In Reduce 2, -> (vi,1)
             -> (u,t) , (w,t)

Works better!
two types of nodes:
L = {v | N(v) <= sqrt{m} }
H = {v | N(v) > sqrt{m} }

|L| <= n -> produce O(m) paths
|H| <= 2sqrt{m} -> produce O(m) paths
if m = O(n^2)  (very dense)
 n ~ sqrt{m}
-> O(m^{3/2}) work (optimal!)


------------------
Idea 2 : Graph Split
 partition V into p equal-size sets {V1,V2,...,Vp}
 For triples (Vi,Vj,Vk) -> subgraph G_{ijk} = G[Vi + Vj + Vk]
                computer triangles on G_{ijk}
 triangles counted {1,p-2, or p^2} times
```

figure out and adjust

subgraph has O(m/p^2) edges in expectation
work: p^3 * O((m/p^2)^{3/2}) = O(m^{3/2})

p about 20 worked best on LiveJournal graph