CS7960 L17 : MapReduce | Matrix Multiply + Relational Algebra

MapReduce

M = Massive Data

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

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

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

Can repeat, constant # of rounds

Word Count "Hello, World!" of MapReduce
 -> count occurrences of each word

Input, M = large corpus of text.

Mapper:
 each word in M -> ("word", 1)

Reducer:
 for all ("word", v_i)
  -> ("word", sum_i v_i)
"aggregate" in Hadoop.

Matrix-Vector Multiply

Input M = n x n matrix [square] (sparse!)
 V = n x 1 matrix [column]

X = M * V

Output
x_i = sum_{j=1}^n m_{ij} * v_j

Mapper:
 (i,m_{ij} * v_j) for all m_{ij}

```
Reducer:
  "aggregate"

--------------
What if v does not fit on mapper?
 ->  Stripe v

Sets of columns of M + Sets of rows of V
Now, each mapper can do m_{ij}v_j with just part of V.


------------------------------------------
Matrix-Matrix Multiply

Input M = n x n matrix    [square]   (sparse!)
      B = n x n matrix    [square]   (sparse!)

X = M*B

Output
x_ik = <m_{i*} , v_{*k}>
     = sum_{j=1}^n m_{ij}*v_{jk}

Mapper:
  ((i,k) , <m_{i*}, v_{*k}>)

Reducer:
  "aggregate"

--------
Again, requires mapper to store too much:

Mapper:
  ((i,k) , m_{ij}*v_{ik})

where each mapper gets squares
  [i_1,i_2] x [j_3,j_4]  of M
and
  [j_3,j_4] x [k_5,k_6]  of B
(so j's align)


Problem:  js may not be aligned...(fix later)

-----------------------------
```

Relational Algebra?

Selection: Find all tuples that satisfy sigma( )
  MR = OVERKILL

Mapper: If sigma(T) true -> (T,T)

Reducer: all (T,T) -> (T,T)
   "identity"

Projection: For tuple, produce subset S or attributes pi_S( )

Mapper: T -> (pi_S(T), pi_S(T))
  Let T' = pi_S(T)

Reducer: All {(T',T'), (T',T'), ...} -> (T',T')
    "remove-duplicate"

Union: For sets R,S return union

Mapper: tuple T -> (T,T)
  given chunks of R and/or S

Reducer: if {(T,T) , (T,T)} -> (T,T) <appears twice> -> <appears-once>
         if {(T,T)} -> (T,T)

Intersection: For sets R,S return intersection

Mapper: tuple T -> (T,R) if in R, and -> (T,S) if in S
  given chunks of R and/or S

Reducer: if {(T,R) , (T,S)} <appears twice> -> (T,T)
         otherwise -> NULL

Join: R(A,B) >< S(B,C) -> H(A,C) if agree on B

Mapper: (a,b) in R -> (b,(R,a))

```
        (b,c) in S  ->  (b,(S,c))

Reducer: {(b,(R,a1)),(b,(R,a2)),...} {(b,(S,c1)),(b,(S,c2)),...}
  ->  (b,{(a1,b,c1),(a1,b,c2),...,(a2,b,c1),(a2,b,c2),...})


-------
Grouping + Aggregation:  R(A,B,C) -> sigma_{A,theta(B)}(R)
    where grouped by A, theta over B
    where theta = {SUM, PRODUCT, COUNT, MIN, MAX}

Mapper: (a,b,c) -> (a,b)

Reducer: {(a,b1),(a,b2),...} -> (a,theta(b1,b2,...))

for R(words, 1) and theta=SUM, then := word-count!


------
Matrix Multiplication, Revisited:
 Input M (nxn) and N (nxn)   both sparse!

2-rounds

Mapper1:  m_{ij} -> (j,(M,i,m_{ij}))
          n_{jk} -> (j,(N,k,n_{jk}))

Reducer1:
  for some j, for each pair (M,i,m_{ij}) , (N,k,n_{jk})
    ->  (j,(i,j,m_{ij}* n_{jk}))

Mapper2:
   {(j,(i1,k1,v1)) , (j,(i2,k2,v2)), ... (j,(ip,kp,vp))}
 -> {((i1,k1),v1) , ((i2,k2),v2), ... ((ip,kp),vp)}

Reducer2: "aggregate" on keys (i,k)

------

1-round

Mapper:  m_{ij} -> ((i,k),(M,j,m_{ij})
         n_{jk} -> ((j,k),(N,j,n_{jk})

Reducer:  For all (i,k):
  sort (M,j,m_{ij}) by j    and
  sort (M,j,n_{jk}) by j
```

For pairs with same j
-> $((i,k), \mathrm{sum}_j(m_{ij} * n_{jk}))$

Sort may be too big to fit on single Reducer.
But if sparse, then may be ok.

PageRank

Web, basically, a nxn big matrix M

each row represents a webpage, and if it links to L pages, then it has L non-zero entries with a value $1/L$.

What to compute the position of an random web browser v. $v(i)$ probability browser reaches page i. Once at page i, goes to random page linked to by i.

In the limit $v = M v$

So v is the top eigen-vector of M.

Solving for v, takes $O(n^3)$ time, too long!

Instead, can approximate v, by starting at any v_0 (e.g. $v_0(i) = 1/I$) for I webpages.
Then Let $v_1 = M*v_0$...
- in general $v_{i+1} = M*v_i$

and in the limit (about v_{50}), $v_i = v$.

So we need to execute matrix-vector multiply about 50 times (with M very sparse).

Could also compute M^{50}, and set $v = M^{50} * v$, but M^{50} no longer sparse...

Much more in "Data Mining" next semester...