MapReduce

CS5300

Hugo Hache, Chet Mancini, Sean Ogden

Summary

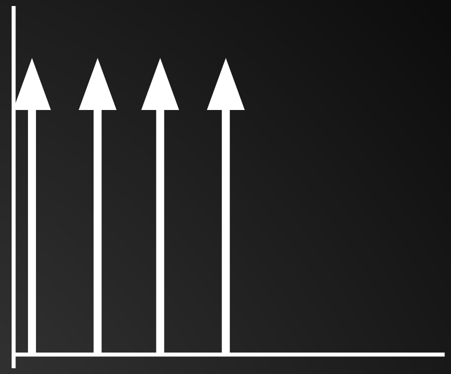
Challenges:

Hadoop setup
and conventions

Current State:

- Working locally and on Amazon EMR
- Extra credit completed
- Code includes comprehensive JUnit test cases.

Indexing Format



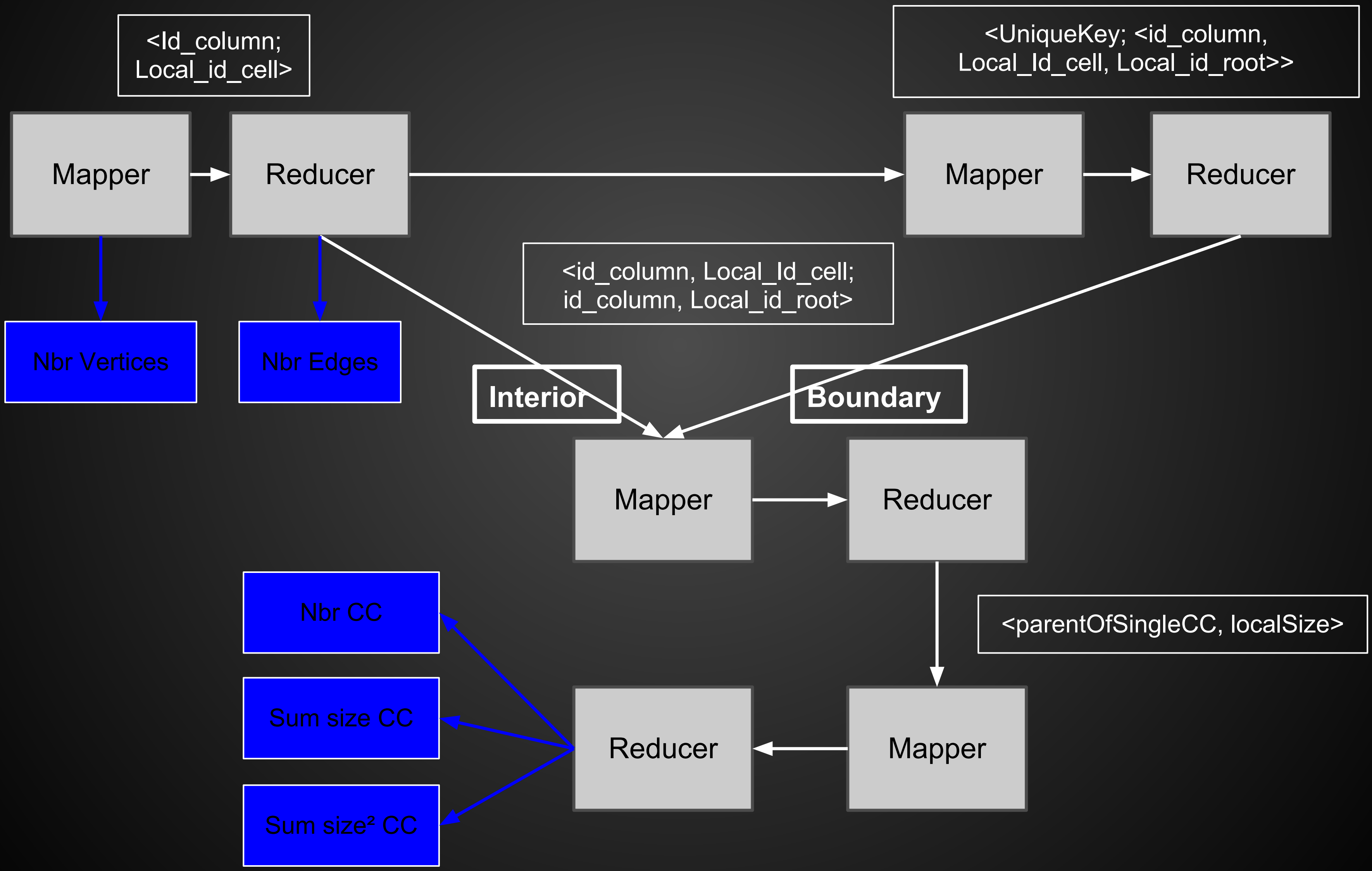
Line Number:
transform line # from input to (i, j) components

Global:
Start at southwest and count up each column.

```
   index = i + (N * j)
```

Local:
Same scheme as global, but specific to column group. Column groups also have index from 0.

A Computation in Four Passes



<Id_column; Local_id_cell>
<UniqueKey; <id_column, Local_Id_cell, Local_id_root>>
Mapper
Reducer
Mapper
Reducer
Nbr Vertices
Nbr Edges
<id_column, Local_Id_cell; id_column, Local_id_root>
Interior
Boundary
Mapper
Reducer
<parentOfSingleCC, localSize>
Nbr CC
Sum size CC
Sum size² CC
Reducer
Mapper

Formula of statistics

- Average Size CC

  sum size CC / nbr CC

- Weighted Average Size CC =

  sum size^2 CC / sum size CC

- Average Burn Count =

  (# vertices / # cells)
      * weighted avg size CC

The Union Find Algorithm

- Mapped all global indices to complex numbers <G, i>
- Each column group gets G=group number, i = 0, ... , N where N is the number of positions in a group
- Points are indexed from bottom to top, left to right.
- If two points overlap, like <0,30> and <1,0>, then the first one is strictly less than the second, so standard union find works without modification in the second pass

Example

First phase: partitioning

<0,4>	<0,9>	<0,14>/<1,4>	<1,9>	<1,14>
<0,3>	<0,8>	<0,13>/<1,3>	<1,8>	<1,13>
<0,2>	<0,7>	<0,12>/<1,2>	<1,7>	<1,12>
<0,1>	<0,6>	<0,11>/<1,1>	<1,6>	<1,11>
<0,0>	<0,5>	<0,10>/<1,0>	<1,5>	<1,10>

Example

Second phase: boundary columns only

<0,4>	<0,14>	<1,4>	<1,14>
<0,3>	<0,13>	<1,3>	<1,13>
<0,2>	<0,12>	<1,2>	<1,12>
<0,1>	<0,11>	<1,1>	<1,11>
<0,0>	<0,10>	<1,0>	<1,10>

- 0,10 is to the right of <1,0>, so algorithm will catch if root is lower in group 0 for that vertex

Example

Third phase: partitioning with adjusted boundaries

<0,4>	<0,9>	<1,4>	<1,9>	<1,14>
<0,3>	<0,8>	<1,3>	<1,8>	<1,13>
<0,2>	<0,7>	<1,2>	<1,7>	<1,12>
<0,1>	<0,6>	<1,1>	<1,6>	<1,11>
<0,0>	<0,5>	<1,0>	<1,5>	<1,10>

- When there is overlap, choose the _higher_ value (for example, <1,0> instead of <0,10>)

Union Find: 1st Pass

- Start from bottom left point, working up to top left
    - at each vertex, check if there is another vertex below or to the left of it
    - if so, merge their roots by walking up the tree until the parent of a vertex is itself
    - replace the root that has the higher number with the lower numbered root
- Proceed to next column and work bottom to top
- continue.

Union Find: 2nd Pass

- Start at the bottom left again
- repeat all steps from the first pass
- lowest index from every component will definitely be encountered before any other, so two passes are sufficient

Our Results (non-diagonal)

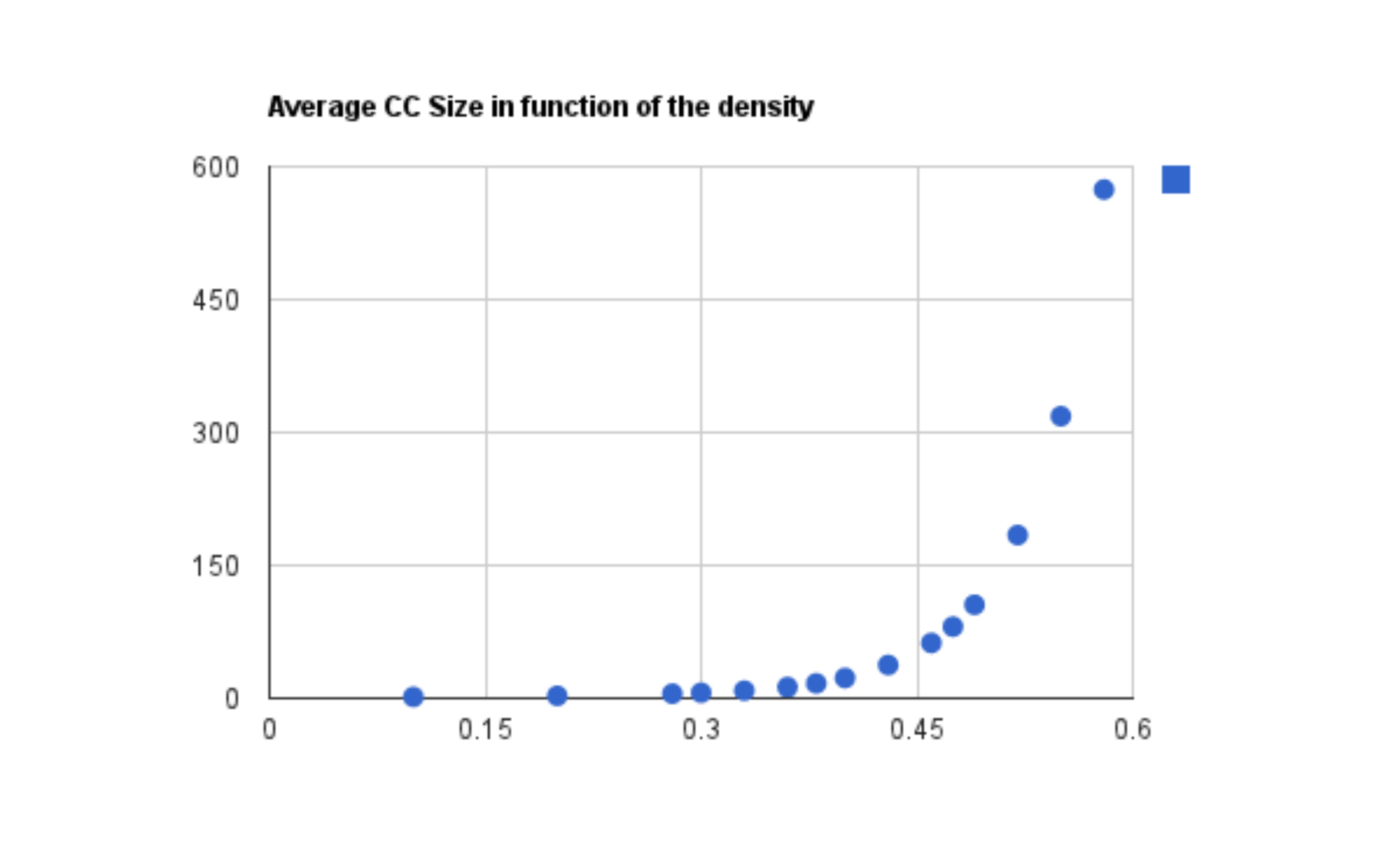
From NetID "974"
Our statistics on productions.txt with size 10000*10000:

# of vertices	58997294
# of edges	69608194
# CC	2897988
Avg CC Size	20.358019
Weighted Avg CC Size	7543.9851
Burn Count	4450.74707628

Performance

# Instances	Minutes
10	25
20	19
30	14

Results w/ Diagonal Added


Average CC Size in function of the density