Counting Triangles and the CURSE OF THE LAST REDUCER

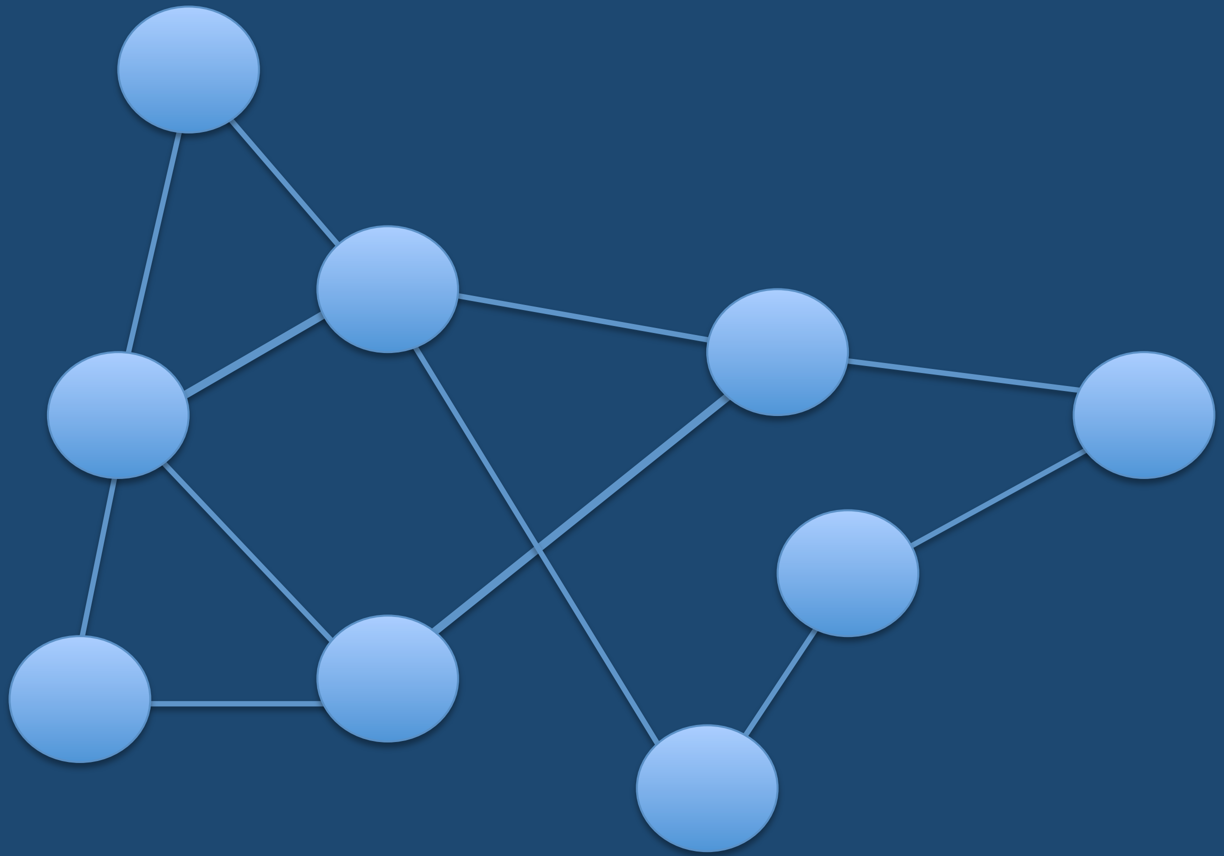
A paper by: Siddharth Suri and Sergei Vassilvitskii

Presented by: Ryan Rogers (with some slides from Sergei's Presentation)

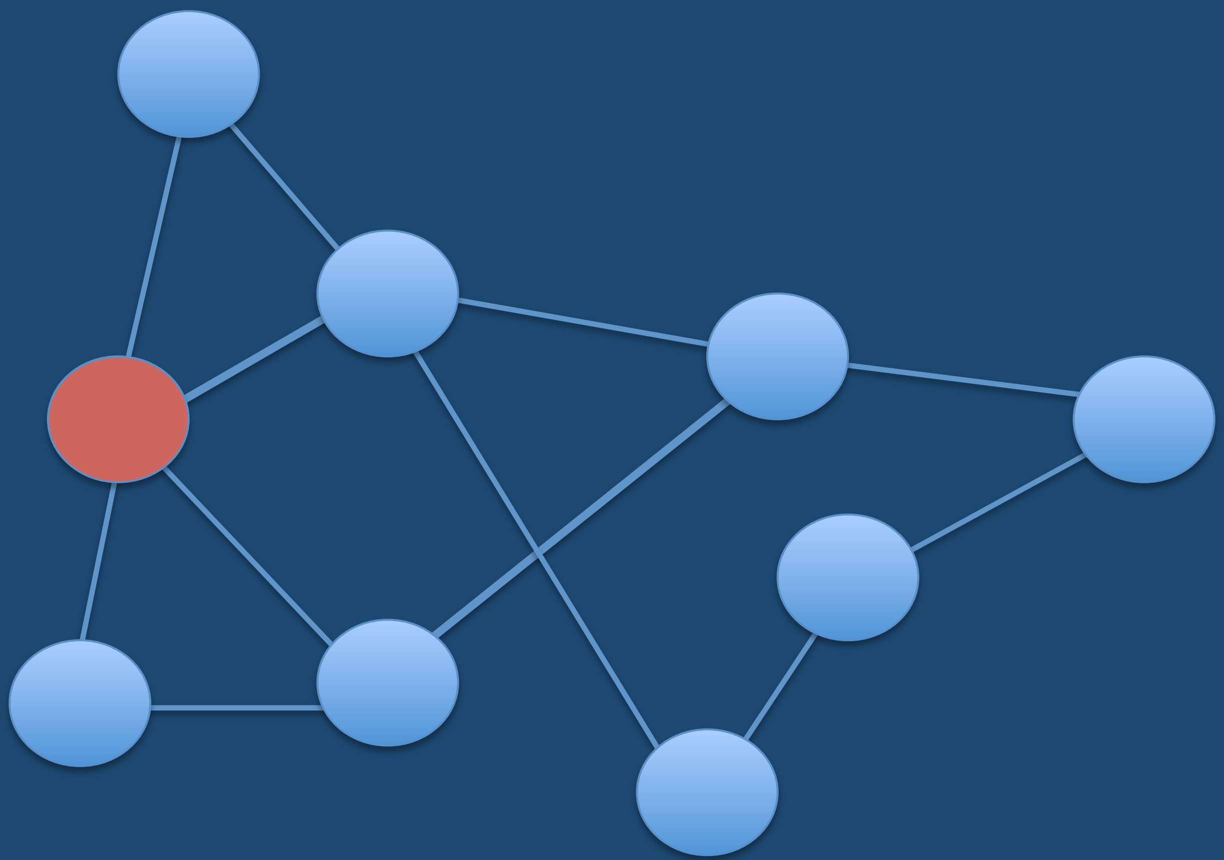
Introduction

- Study Social Networks
- Main metric for analyzing Social Networks: Clustering Coefficient of each node
- Problem of finding the Clustering Coefficient of a node is basically the same as counting the number of △'s incident to that node.

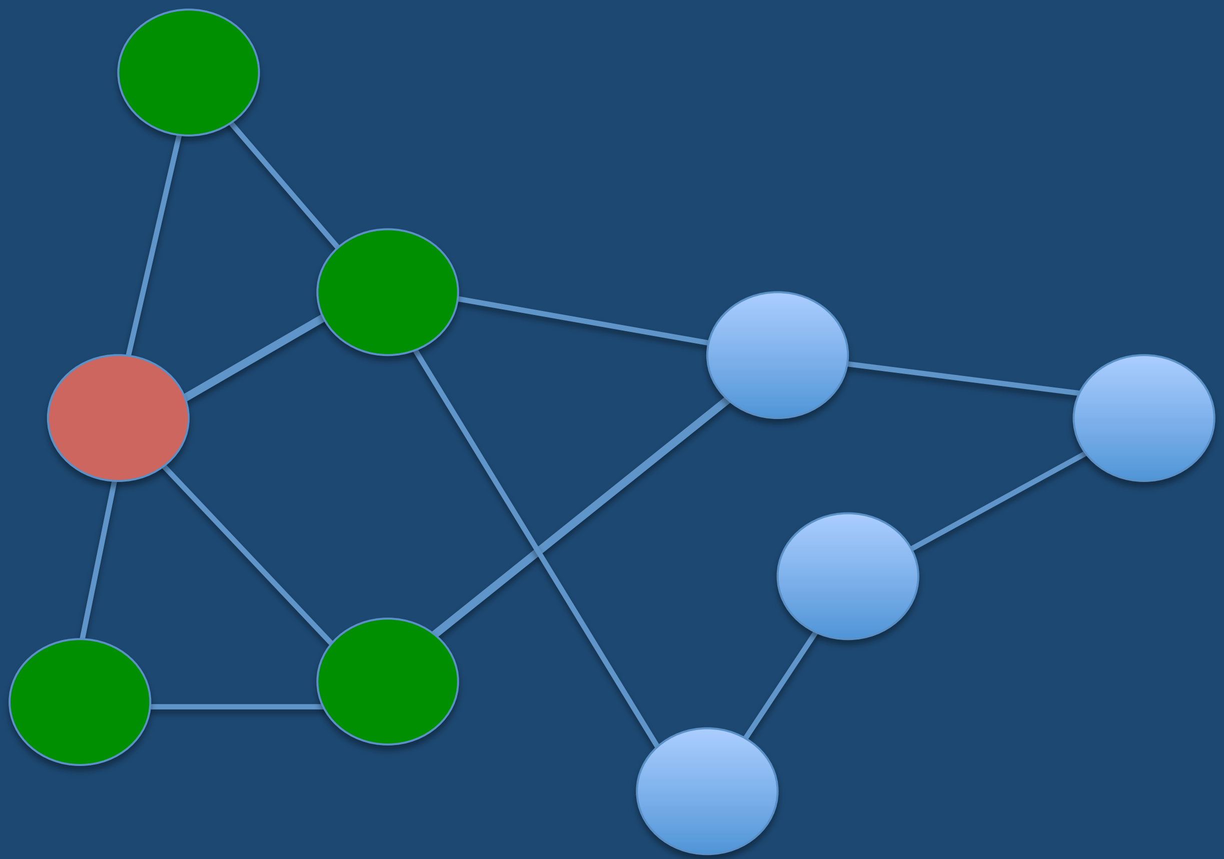
Clustering and Triangles



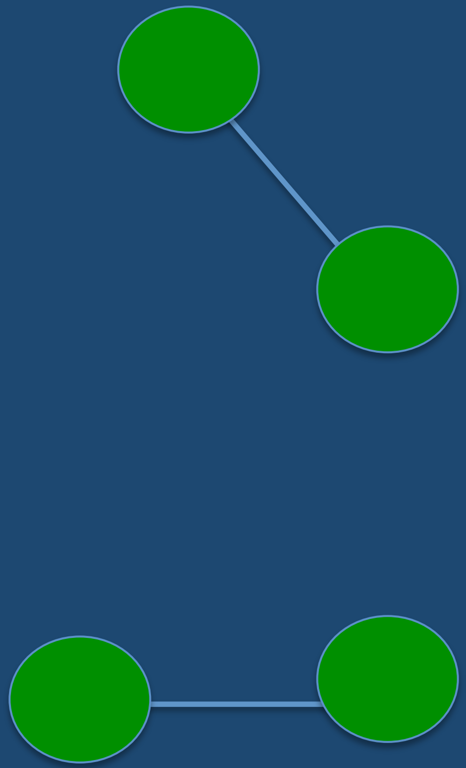
Clustering and Triangles



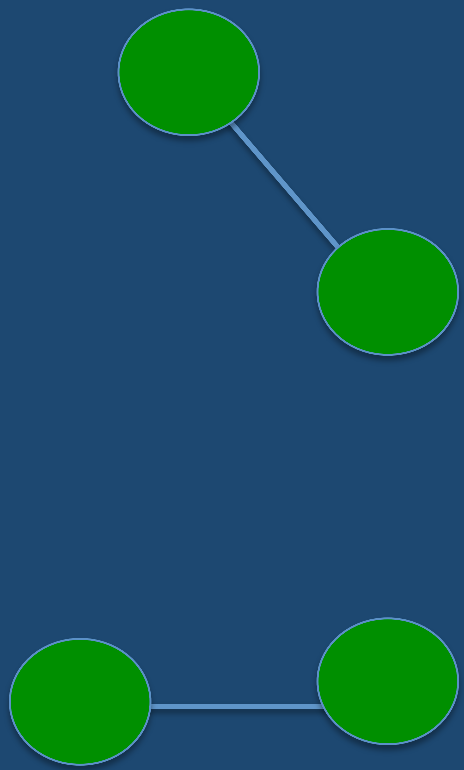
Clustering and Triangles

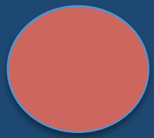


Clustering and Triangles



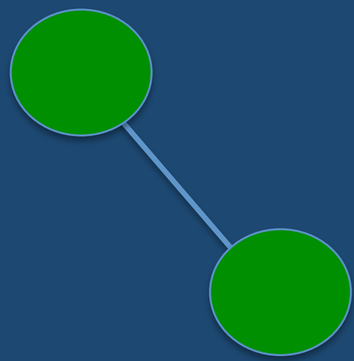
Clustering and Triangles

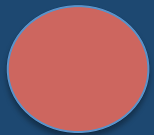


CC() $= 2/\binom{4}{2}$

$= 1/3$

Clustering and Triangles

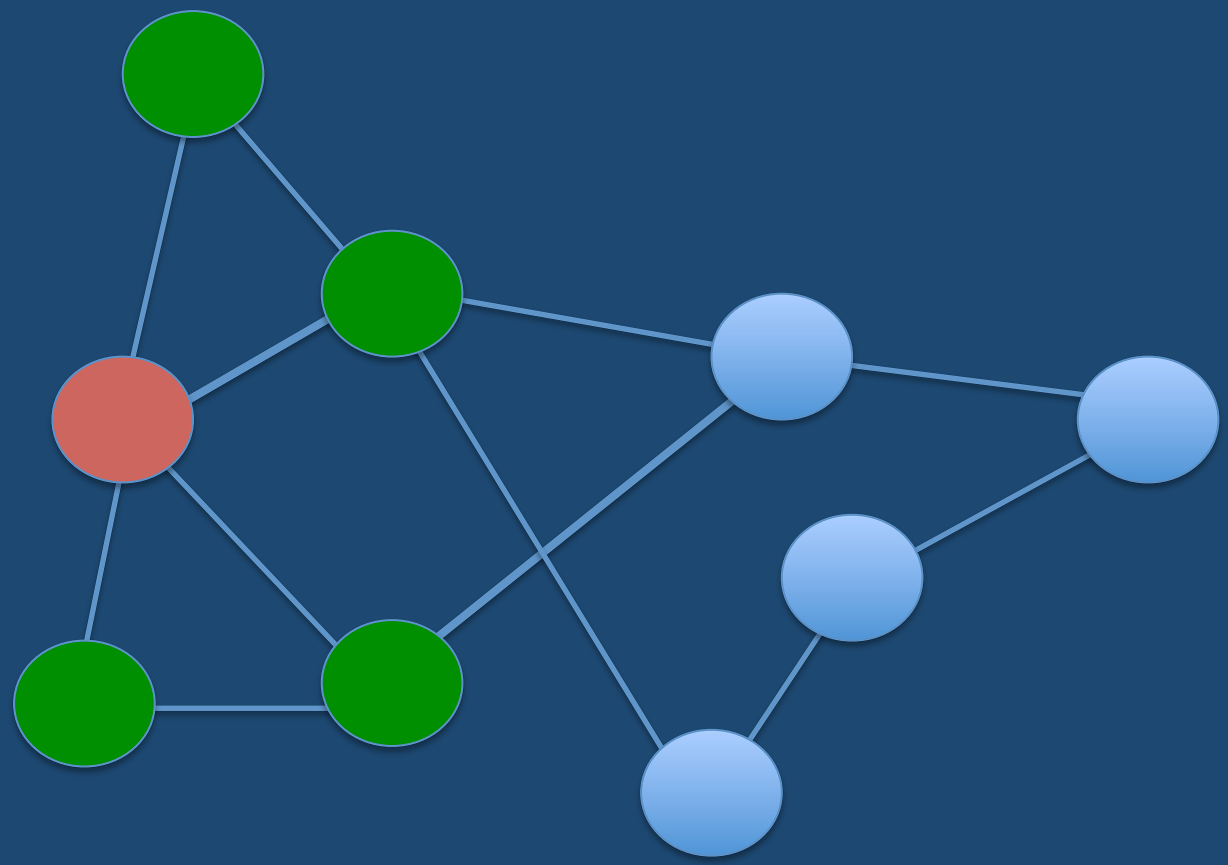


CC() $= 2/\binom{4}{2}$

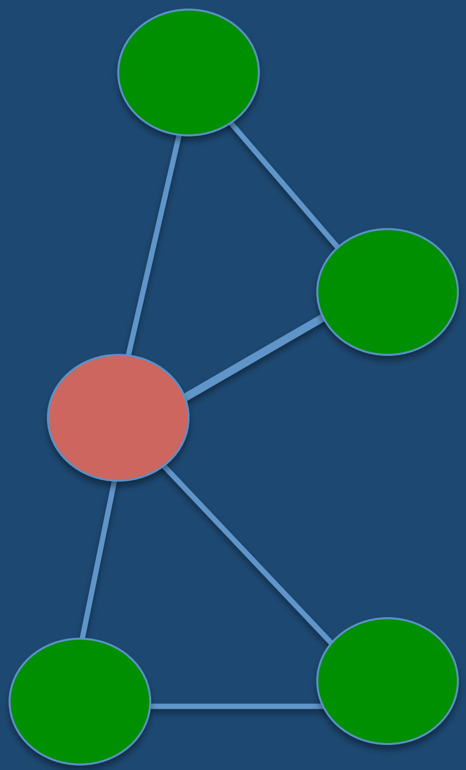
$= 1/3$

OR…

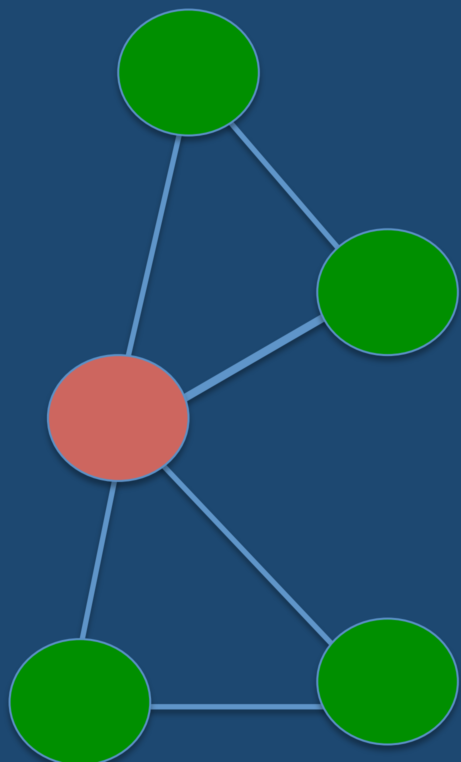
Clustering and Triangles

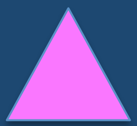


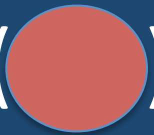
Clustering and Triangles



Clustering and Triangles



Number of  ‘s

= $\binom{d}{2}$ x CC()

Past Work

- Coppersmith and Kumar ('04) and Buriol et al. ('04): Streaming algorithms to find **total** number of triangles with **high accuracy**
- Becchetti et al. ('08): **Estimate** the number of triangles incident on **each node**.
- Tsourakakis et al. ('09): Randomized MapReduce procedure that gives the **total** number of triangles accurately in **expectation**.

Contributions

- Count the **exact** number of triangles
- Count the number of triangles incident on **each node**, exactly.
- Comparable speedup as the randomized MapReduce procedure.

Counting Triangles (Naïve)

- Let $T \longleftarrow 0$
  - for $v \in V$
    - for each $u \in \Gamma(v)$
      - for each $w \in \Gamma(v)$
        - » if $(u,w) \in E$

          $T \longleftarrow T + 1/2$

- Output $T \longleftarrow T/3$

RUN TIME

$$O\left(\sum_{u \in V} d_u^{\,2}\right)$$

MapReduce (Naïve)

- Map 1: $G=(V,E)$

$$< key,value >$$

MapReduce (Naïve)

- Map 1: $G = (V, E)$

$< v_1, \Gamma(v_1) >$ $< v_2, \Gamma(v_2) >$ $< v_3, \Gamma(v_3) >$ … $< v_n, \Gamma(v_n) >$

MapReduce (Naïve)

- Map 1:

$$G=(V,E)$$

$$<v_1,\Gamma(v_1)> \quad <v_2,\Gamma(v_2)> \quad <v_3,\Gamma(v_3)> \quad \cdots \quad <v_n,\Gamma(v_n)>$$

- Reduce 1:

$$<v,\Gamma(v)> \longrightarrow \left\{<(u_1,u_2),v>: u_1,u_2 \in \Gamma(v)\right\}$$

MapReduce (Naïve)

- Map 2: $< (u,w), v > + < (u,w), Edge >$

$$< (u,w), \left\{ v_1, \ldots, v_k, Edge? \right\} >$$

- Reduce 2:
  - If $Edge$ then.
    - For $v \in \left\{ v_1, \ldots, v_k \right\}$ emit $< v, 1 >$

What's Wrong with this?

- Does this improve our running time?
- There still may be a very high degree vertex in the network
- Thus, one machine may be stuck with a lot of data!

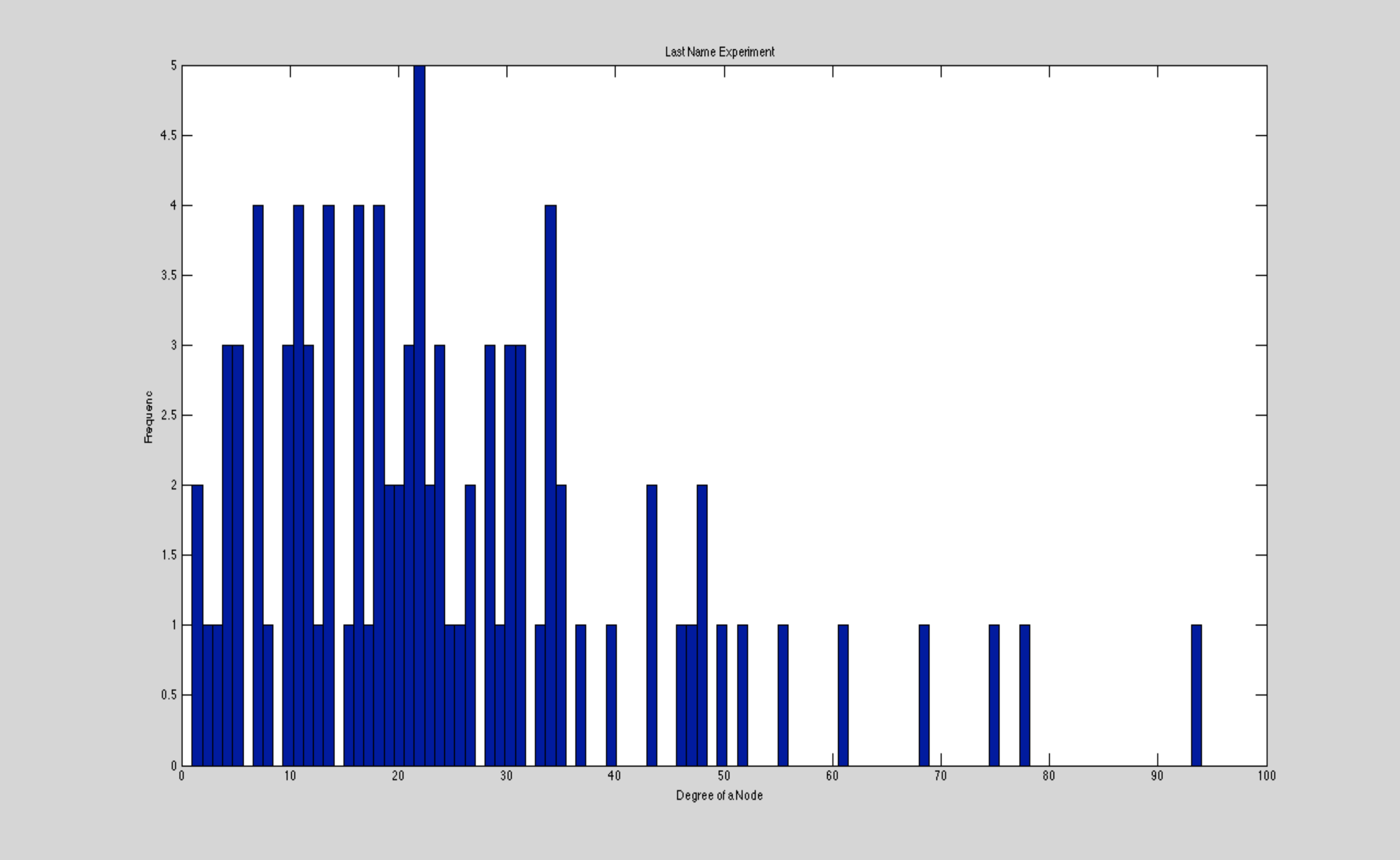
STILL HAS RUN TIME

$$O\left(d_{\max}{}^{2}\right)$$

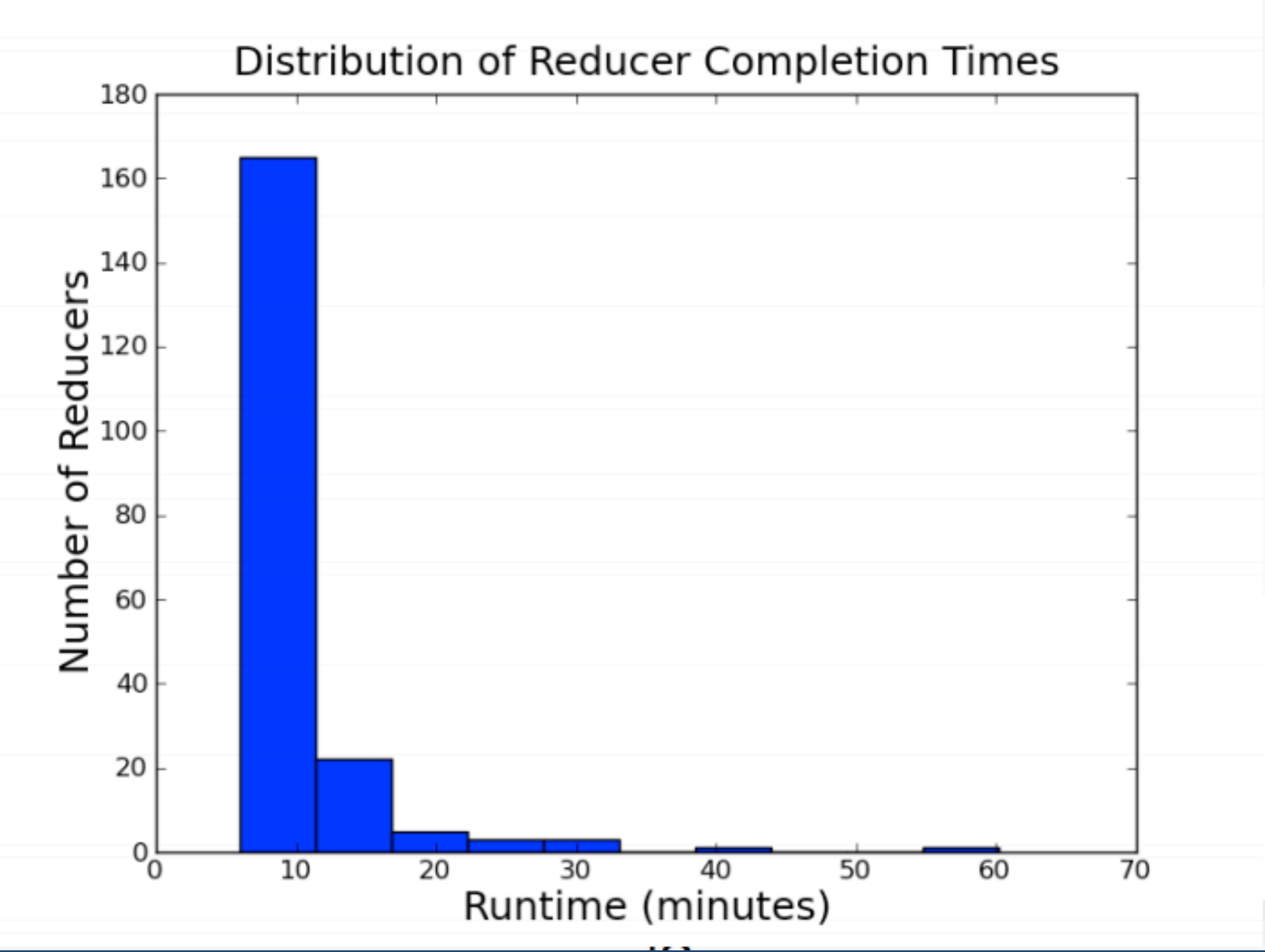
Reality

- Social Networks are typically sparse
- However, there may be few nodes with very high degree.

Reality


Last Name Experiment
Frequenc
Degree of a Node
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
10
20
30
40
50
60
70
80
90
100

Live Journal Data



THE CURSE OF THE LAST REDUCER

- The idea that 99% of the computation finishes quickly, but the last 1% takes a HUGE amount of time.

Possible Fixes

- Generating 2-paths around high-degree nodes is expensive – concentrate on low degree.

- Divide the graph into overlapping subgraphs and somehow account for the overlap.

Counting Triangles (Optimal)

- NodeIterator++ (V, E)
  - $T \longleftarrow 0$
  - For $v \in V$
    - For $u \in \Gamma(v)$ and $u \succ v$
      - for $w \in \Gamma(v)$ and $w \succ u$
        - »if $(u, w) \in E$
          - $T \longleftarrow T + 1$
- Return T

$d_u > d_v$

Properties of NodeIterator++

- Has running time $O(m^{3/2})$ and gives the exact number of triangles incident to each node [Schank '07]
- Best possible bound:

MR-NodeIterator++

- Map 1’:
  - If $v \succ u$
    - Emit $< u, v >$
- Reduce 1’:

$$< u, S \subseteq \Gamma(u) > \longrightarrow \left\{ < u, (v, w) >: v, w \in S \right\}$$

- Map 2, Reduce 2.

Memory Required per Machine

- Lemma: The input to any reduce instance in first round has $O\left(\sqrt{m}\right)$ edges (Sublinear space)
- Proof:

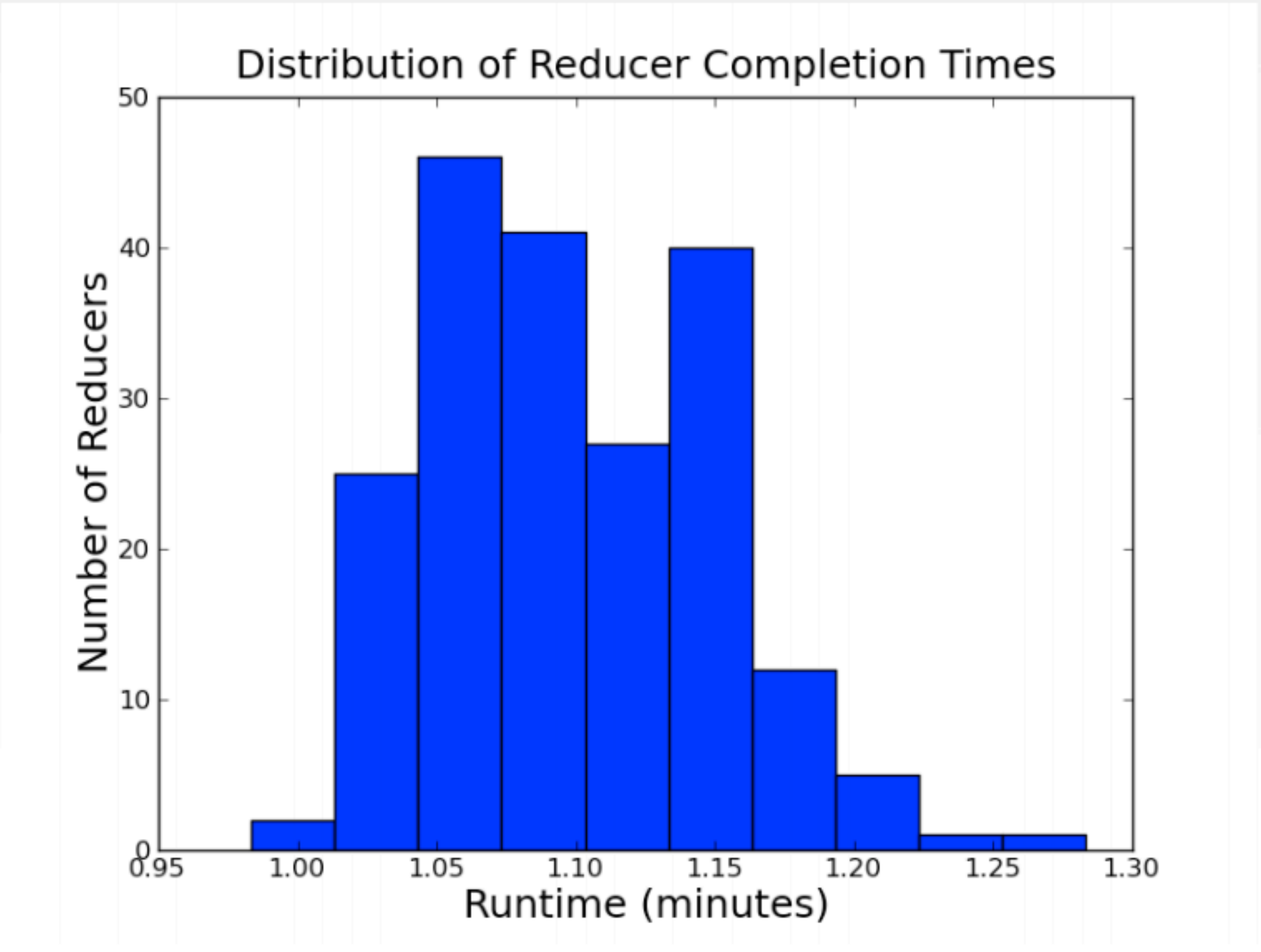
$$\mathcal{L} = \left\{ v \in V : d_v < \sqrt{m} \right\}$$

$$\mathcal{H} = \left\{ v \in V : d_v \geq \sqrt{m} \right\}$$

Size of Output after Round 1

- Lemma: The total number of records output at the end of the first reduce is $O\left(m^{3/2}\right)$
- Proof:
  - There are at most n= $O\left(m^{1/2}\right)$ machines with low degree nodes, and each machine produces an output of size $O\left(m\right)$
  - There are at most $O\left(m^{1/2}\right)$ machines with high degree nodes and each machine must output pairs with other high degree nodes => $O\left(m\right)$ output size

Did it Help?


Distribution of Reducer Completion Times
Number of Reducers
0
10
20
30
40
50
0.95
1.00
1.05
1.10
1.15
1.20
1.25
1.30
Runtime (minutes)

Possible Fixes

- Generating 2-paths around high-degree nodes is expensive – concentrate on low degree.

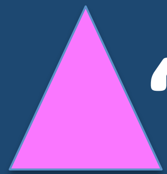
- Divide the graph into overlapping subgraphs and somehow account for the overlap.

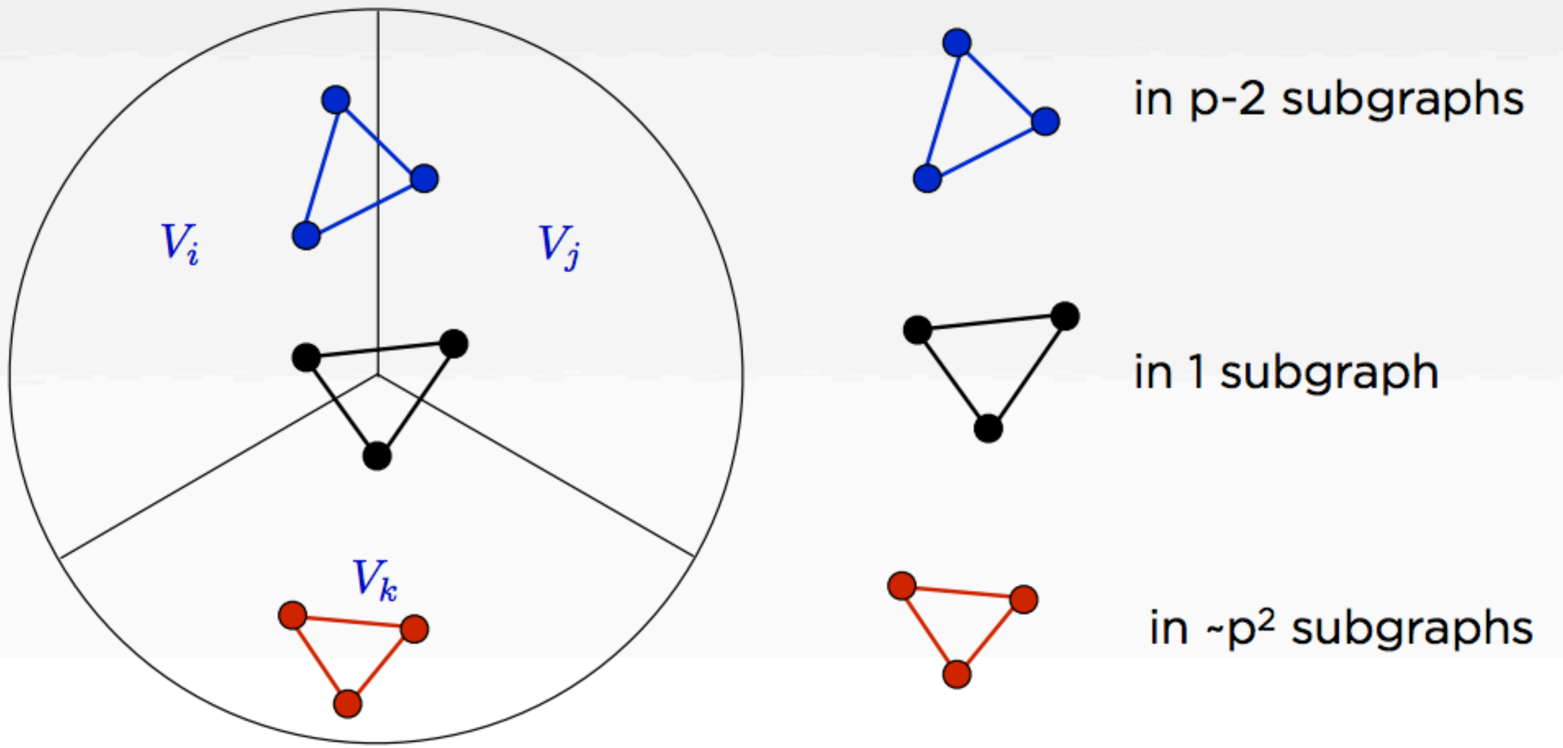
MR-GraphPartition

- Input: (V, E, ρ)
- Partition vertices into ρ equal sized $V_0, \ldots, V_{\rho-1}$
- Consider all triples (V_i, V_j, V_k) and the induced graph $G_{ijk} = G\left[V_i, V_j, V_k\right]$ for $i < j < k$
- Compute Triangles on each graph separately
  - You can use your favorite triangle counting algorithm on each!
- Map nodes to index *i* by using a universal hash

MR-GraphPartition

- Map 1": Input $<(u,v),1>$
  - for $a<b<c\le\rho-1$
    - if $\{h(u),h(v)\}\subseteq\{a,b,c\}$
      - emit $<(a,b,c),(u,v)>$
- Reduce 1": Input: $<(i,j,k),E_{ijk}>$
  - Count Triangles and weight accordingly.

May Over Count # of 's



Can count exactly how many subgraphs each triangle will be in

Analysis

- The expected size of the input to any machine instance is $O\left(m / \rho^2\right)$

- The expected total space used at the end of map phase is $O\left(m\rho\right)$

- Proof: SEE BOARD

Analysis (continued)

- Theorem: For $\rho \leq \sqrt{m}$, the amount of work done by all the machines is $O\left(m^{3/2}\right)$
- Proof:

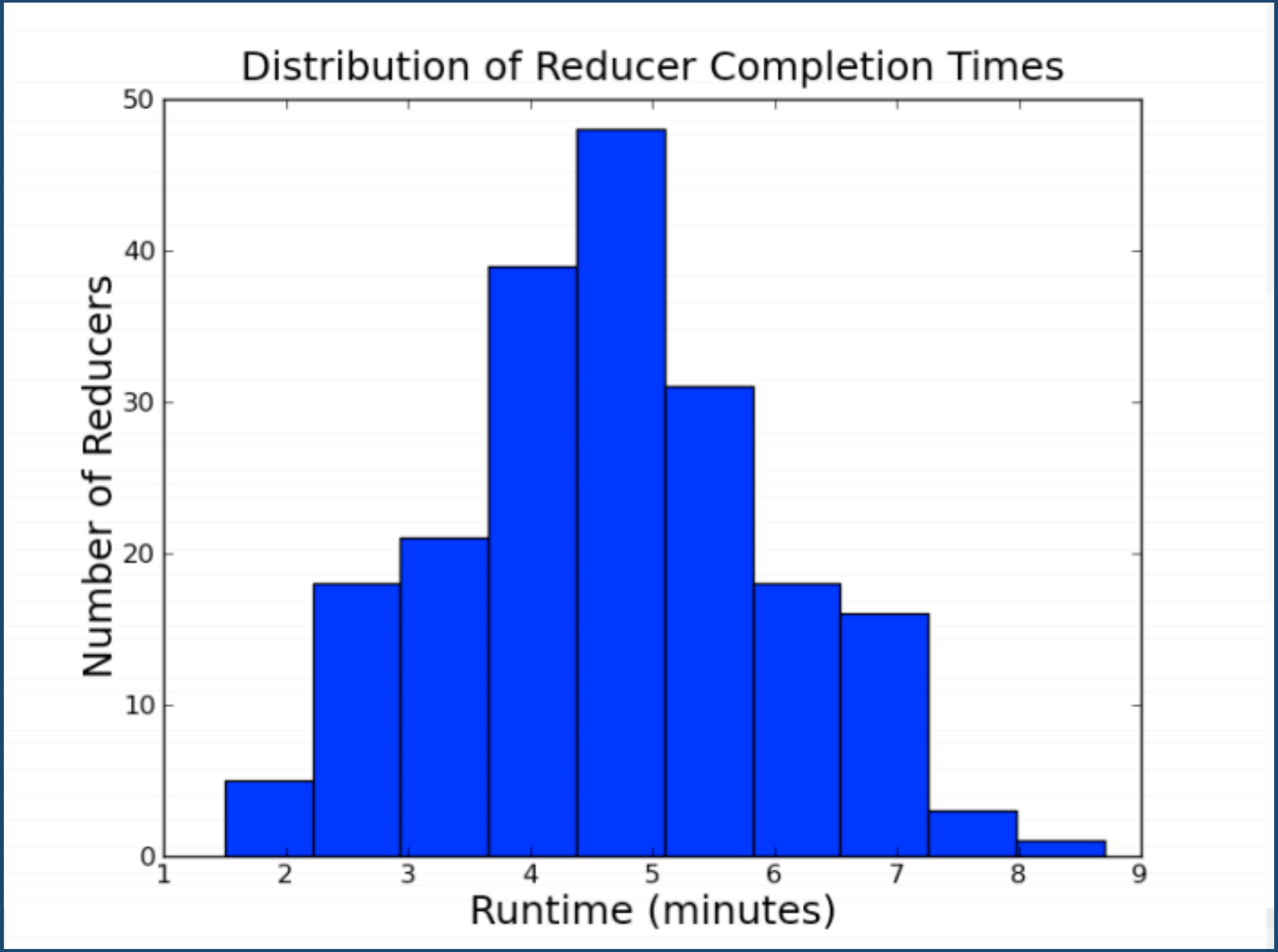
  $O\left(1\right)$ time per edge => $O\left(m\rho\right) = O\left(m^{3/2}\right)$ time for Map 2” phase.

  Partition input amongst $O\left(\rho^{3}\right)$ reducers.

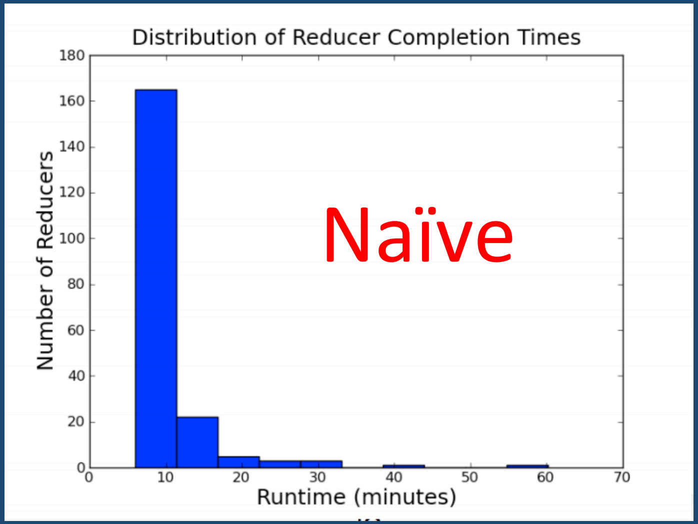
  Running Time per Reducer:

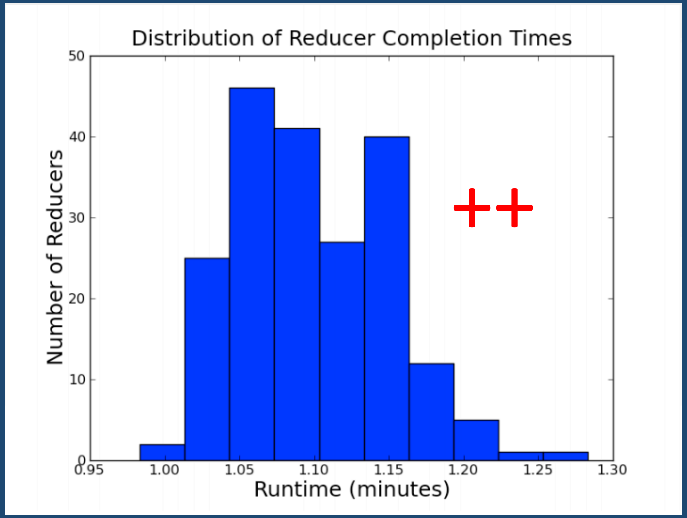
$$= O\left(\#Edges^{3/2}\right) = O\left(\left(\frac{m}{\rho^{2}}\right)^{3/2}\right)$$

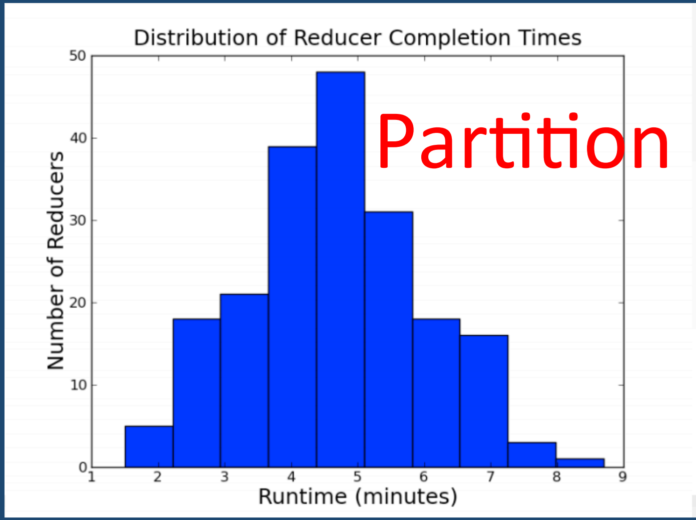
Results for Partition



Comparison of Results







THE CURSE OF THE LAST REDUCER

Questions???