Counting Triangles and Modeling MapReduce

Siddharth Suri

- Yahoo! Research

Outline

- Modeling MapReduce
  - How and why did we come up with our model?
  - [Karloff, Suri, Vassilvitskii SODA 2010]
- MapReduce algorithms for counting triangles in a graph
  - What do these algorithms say about the model?
  - [Suri, Vassilvitskii WWW 2011]
- Open research questions

MapReduce is Widely Used

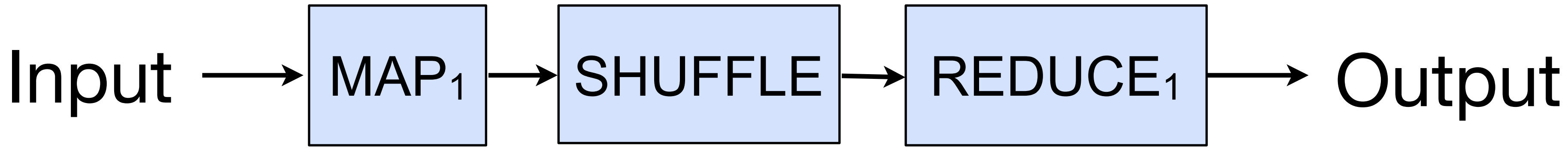
- MapReduce is a widely used method of parallel computation on massive data.
  - Yahoo! uses it to process 120 TB daily
  - facebook uses it to process 80 TB daily
  - Google uses it to process 20 petabytes per day
  - Also used at The New York Times amazon.com IBM ...
- Implementations: Hadoop, Amazon Elastic MapReduce
- Invented by [Dean & Ghemawat '08]

MapReduce: Research Question

- In practice MapReduce is often used to answer questions like:
  - What are the most popular search queries?
  - What is the distribution of words in all emails?
  - Often used for log parsing, statistics
- Massive input, spread across many machines, need to parallelize.
  - Moves the data, and provides scheduling, fault tolerance
- What is and is not efficiently computable using MapReduce?

Overview of MapReduce

- One round of MapReduce computation consists of 3 steps

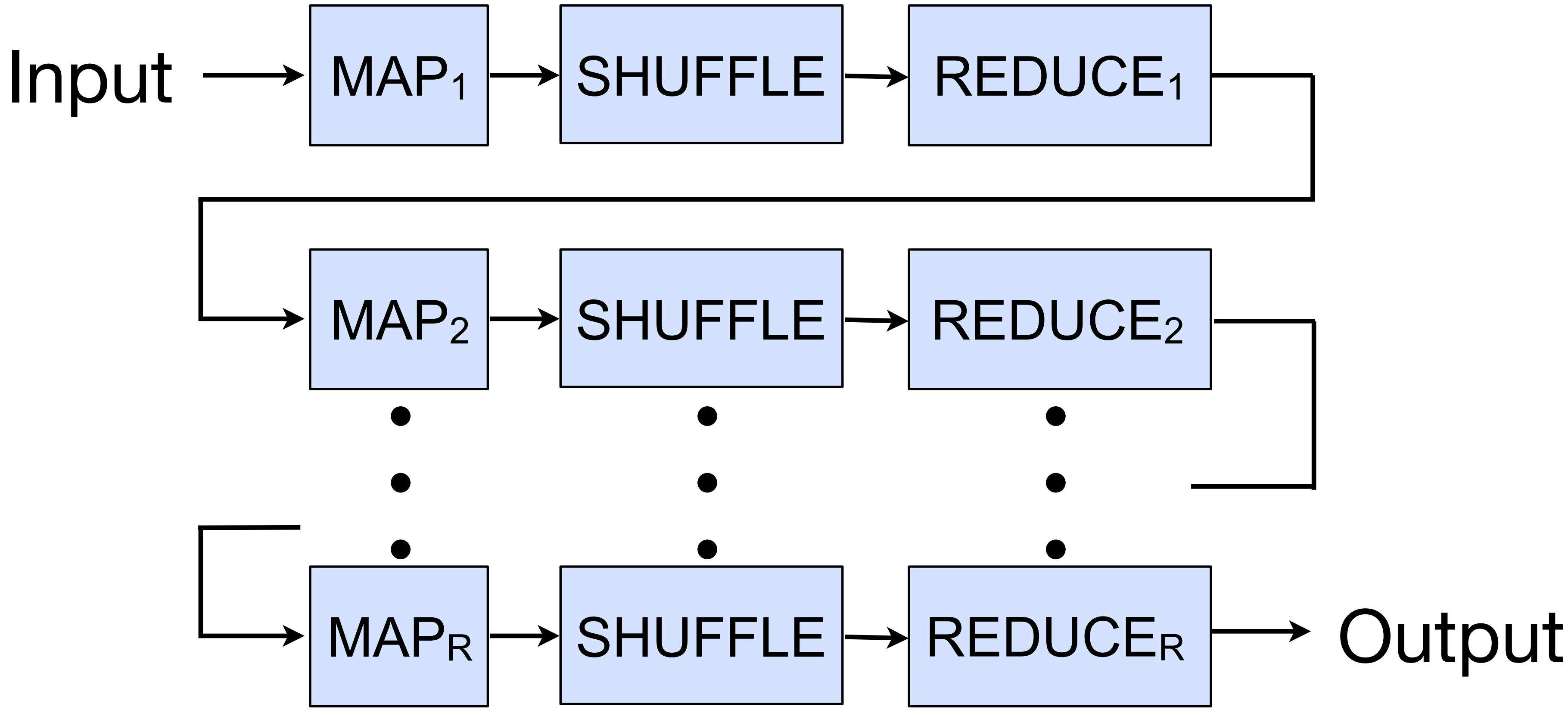


Overview of MapReduce

- One round of MapReduce computation consists of 3 steps

Overview of MapReduce

- One round of MapReduce computation consists of 3 steps



MapReduce Basics: Summary

- Data are represented as a <key, value> pair
- Map: <key, value> → multiset of <key, value> pairs
  - user defined, easy to parallelize
- Shuffle: Aggregate all <key, value> pairs with the same key.
  - executed by underlying system
- Reduce: <key, multiset(value)> → <key, multiset(value)>
  - user defined, easy to parallelize
- Can be repeated for multiple rounds

Building a Model of MapReduce

- The situation:
  - Input size, n, is massive
  - Mappers and Reducers run on commodity hardware
- Therefore:
  - Each machine must have $O(n^{1-\varepsilon})$ memory
  - $O(n^{1-\varepsilon})$ machines

Building a Model of MapReduce

- Consequences:
  - Mappers have $O(n^{1-\varepsilon})$ space
    - Length of a <key, value> pair is $O(n^{1-\varepsilon})$
  - Reducers have $O(n^{1-\varepsilon})$ space
    - Total length of all values associated with a key is $O(n^{1-\varepsilon})$
  - Mappers and reducers run in time polynomial in n
  - Total space is $O(n^{2-2\varepsilon})$
    - Since outputs of all mappers have to be stored before shuffling, total size of all <key, value> pairs is $O(n^{2-2\varepsilon})$

Definition of MapReduce Class (MRC)

- Input: finite sequence $<key_i, value_i>$, $n = \sum_i (|key_i| + |value_i|)$
- Definition: Fix an $\varepsilon > 0$. An algorithm in MRC^j consists of a sequence of operations $<map_1, red_1, \ldots, map_R, red_R>$ where:
  - Each map_r uses $O(n^{1-\varepsilon})$ space and time polynomial in n
  - Each red_r uses $O(n^{1-\varepsilon})$ space and time polynomial in n
  - The total size of the output from map_r is $O(n^{2-2\varepsilon})$
  - The number of rounds $R = O(\log^j n)$

Related Work

- Feldman et al. SODA ’08 also study MapReduce
  - Reducers access input as a stream and are restricted to polylog space
  - Compare to streaming algorithms
- Goodrich et al ’11
  - Comparing MapReduce with BSP and PRAM
  - Gives algorithms for sorting, convex hulls, linear programming

Outline

- Modeling MapReduce
  - How and why did we come up with our model?
  - [Karloff, Suri, Vassilvitskii SODA 2010]
- <u>MapReduce algorithms for counting triangles in a graph</u>
  - <u>What do these algorithms say about the model?</u>
  - [Suri, Vassilvitskii WWW 2011]
- Open research questions

Clustering Coefficient

- Given G=(V,E) unweighted, undirected
- cc(v) = fraction of v's neighbors that are neighbors

$$= \frac{|\{(u,w) \in E \mid u \in \Gamma(v) \text{ and } w \in \Gamma(v)\}|}{\binom{d_v}{2}}$$

$$= \frac{\text{\# triangles incident on v}}{\text{\# possible triangles incident on v}}$$

- Computing the clustering coefficient of each node reduces to computing the number of triangles incident on each node.

Related Work

- Estimating the global triangle count using sampling
  - [Tsourakakis et al '09]
- Streaming algorithms:
  - Estimating global count
    - [Coppersmith & Kumar '04, Buriol et al '06]
  - Approximating the number of triangles per node using O(log n) passes
    - [Becchetti et al '08]

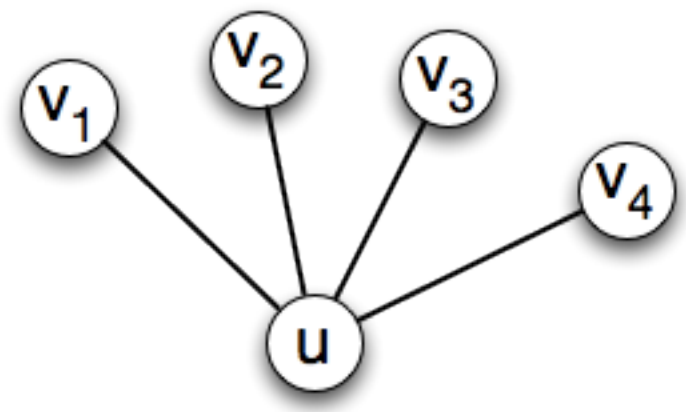
Why Compute the Clustering Coefficient?

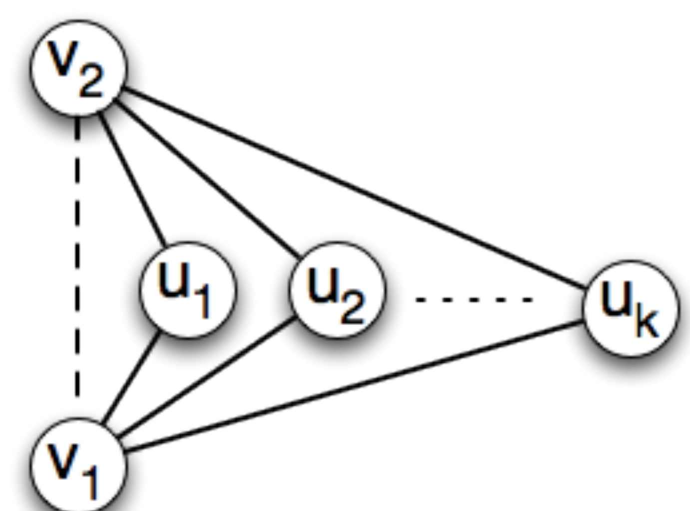
- Network Cohesion: Tightly knit communities foster more trust, social norms
  - More likely reputation is known
  - [Coleman ’88, Portes ’98]

- Structural Holes: Individuals benefit from bridging
  - Mediator can take ideas from both and innovate
  - Apply ideas from one to problems faced by another
  - [Burt ’04, ’07]

Naive Algorithm for Counting Triangles: NodeItr

- Map 1: for each $u \in V$, send $\Gamma(u)$ to a reducer
- Reduce 1: generate all 2-paths of the form $\langle v_1, v_2; u \rangle$, where $v_1, v_2 \in \Gamma(u)$
- Map 2
  - Send $\langle v_1, v_2; u \rangle$ to a reducer,
  - Send graph edges $\langle v_1, v_2; \$ \rangle$ to a reducer
- Reduce 2: input $\langle v_1, v_2; u_1, \ldots, u_k, \$? \rangle$
  - if \$ in input, then v_1, v_2 get $k/3$ Δ's each, and
  - $u_1, \ldots, u_k$ get $1/3$ Δ's each





NodeItr $\notin$ MRC

- Reduce 1: generate all 2-paths among pairs in $v_1, v_2 \in \Gamma(u)$
  - NodeItr generates $O(\sum_{v \in V} d_v^2)$ 2-paths which need to be shuffled
  - In a sparse graph, one linear degree node results in ~n^2 bits shuffled
  - Thus NodeItr is not in MRC, indicating it is not an efficient algorithm.
- Does this happen on real data?

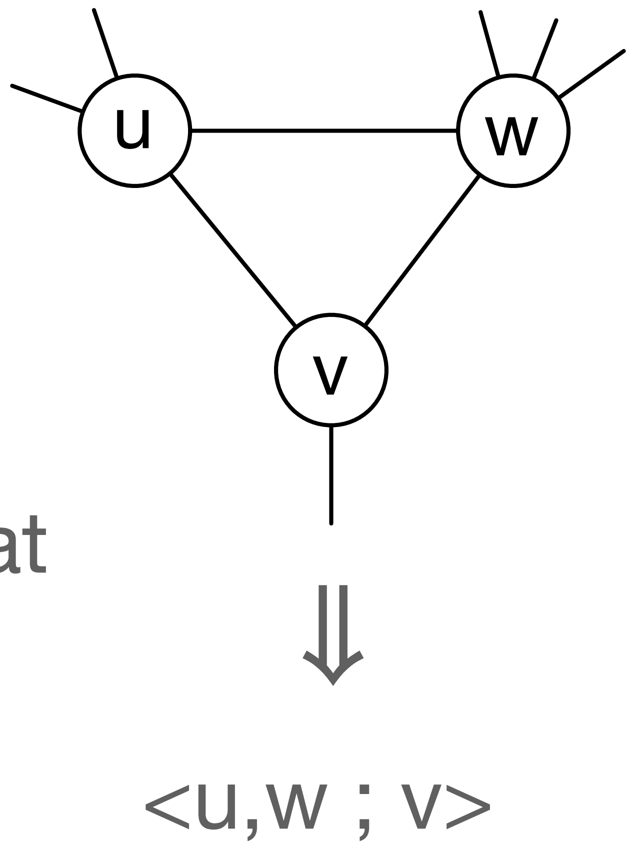
NodeItr Performance

Data Set	Nodes	Edges	# of 2-Paths	Runtime (min)
web-BerkStan	6.9×10^5	1.3×10^7	5.6×10^{10}	752
as-Skitter	1.7×10^6	2.2×10^7	3.2×10^{10}	145
Live Journal	4.8×10^6	8.6×10^7	1.5×10^{10}	59.5
Twitter	4.2×10^7	2.4×10^9	2.5×10^{14}	?

- Massive graphs have heavy tailed degree distributions [Barabasi, Albert '99]
- NodeItr does not scale, model gets this right

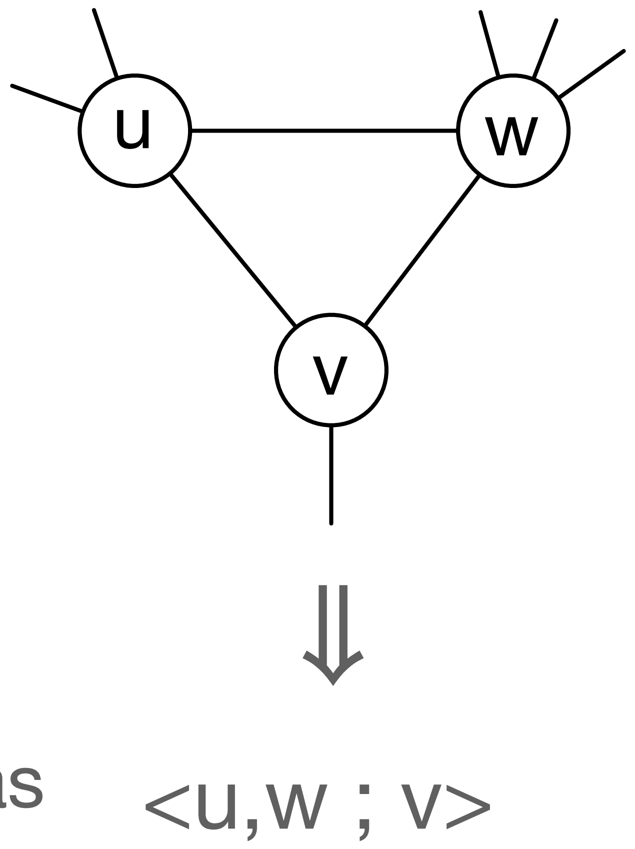
NodeItr++: Intuition

- Generating 2-paths around high degree nodes is expensive
- Make the lowest degree node “responsible” for counting the triangle
  - Let ≫ be a total order on vertices such that $v \gg u$ if $d_v > d_u$
  - Only generate 2-paths <u,w ; v> if $v \ll u$ and $v \ll w$
  - [Schank ’07]



NodeItr++: Definition

- Map 1: if v ≫ u emit <u; v>
- Reduce 1: Input <u; S ⊆ Γ(u)>
  generate all 2-paths of the form <v_1, v_2; u>, where $v_1, v_2 \in S$
- Map 2 and Reduce 2 are the same as before



- Thm: The input to any reducer in the first round has $O(m^{1/2})$ edges
- Thm (Shank '07): $O(m^{3/2})$ 2-paths will be output

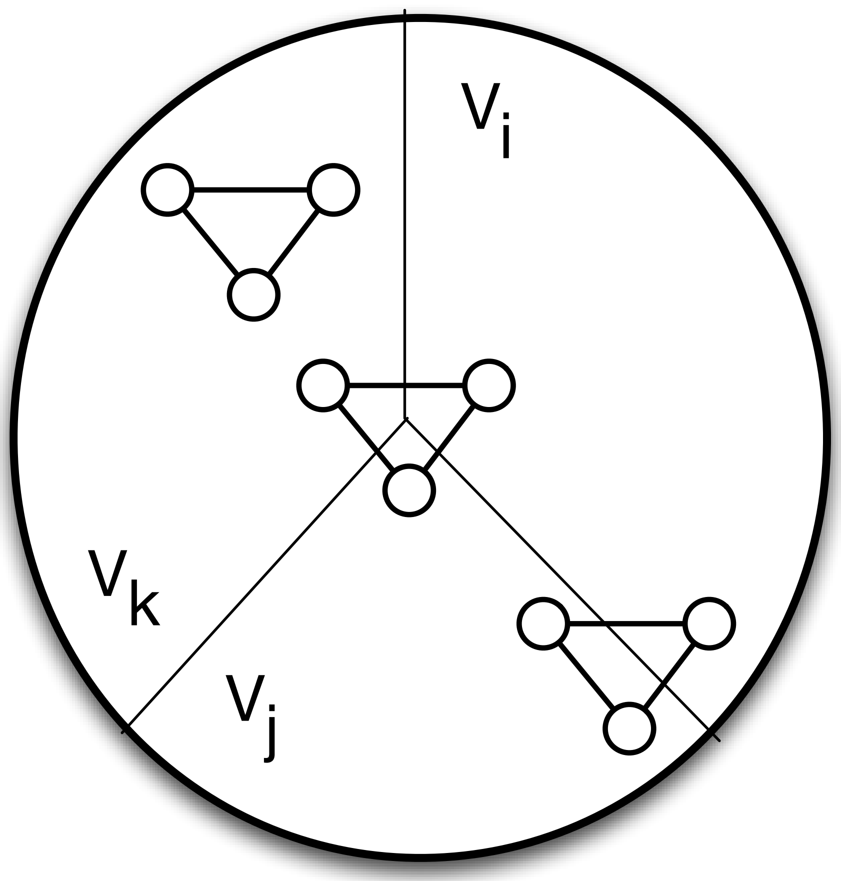
NodeItr Performance

Data Set	# of 2-Paths NodeItr	# of 2-Paths NodeItr++	Runtime (min) NodeItr	Runtime (min) NodeItr
web-BerkStan	5.6×10^{10}	1.8×10^{8}	752	1.8
as-Skitter	3.2×10^{10}	1.9×10^{8}	145	1.9
Live Journal	1.5×10^{10}	1.4×10^{9}	59.5	5.3
Twitter	2.5×10^{14}	3.0×10^{11}	?	423

- Model indicated shuffling m^2 bits is too much but $m^{1.5}$ bits is not

One Round Algorithm: GraphPartition

- Input parameter ρ: partition V into $V_1,\ldots,V_\rho$
- Map 1: Send induced subgraph on $V_i \cup V_j \cup V_k$ to reducer (i,j,k) where $i < j < k$.
- Reduce 1: Count number of triangles in subgraph, weight accordingly



GraphPartition $\in$ MRC^0

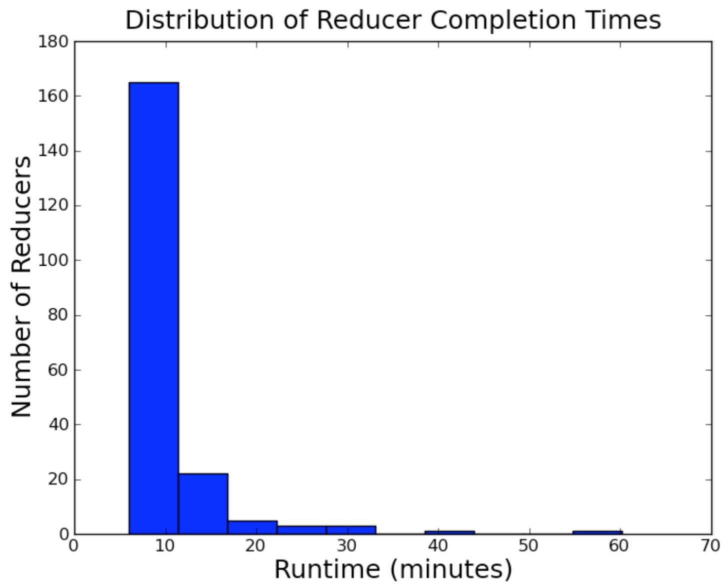
- Lemma: The expected size of the input to any reducer is $O(m/\rho^2)$.
  - $9/\rho^2$ chance a random edge is in a partition
- Lemma: The expected number of bits shuffled is $O(m\rho)$.
  - $O(\rho^3)$ partitions, combined with previous lemma
- Thm: For any $\rho < m^{1/2}$ the total amount of work performed by all machines is $O(m^{3/2})$.
  - ρ^3 partitions, $(m/\rho^2)^{3/2}$ complexity per reducer

Runtime of Algorithms

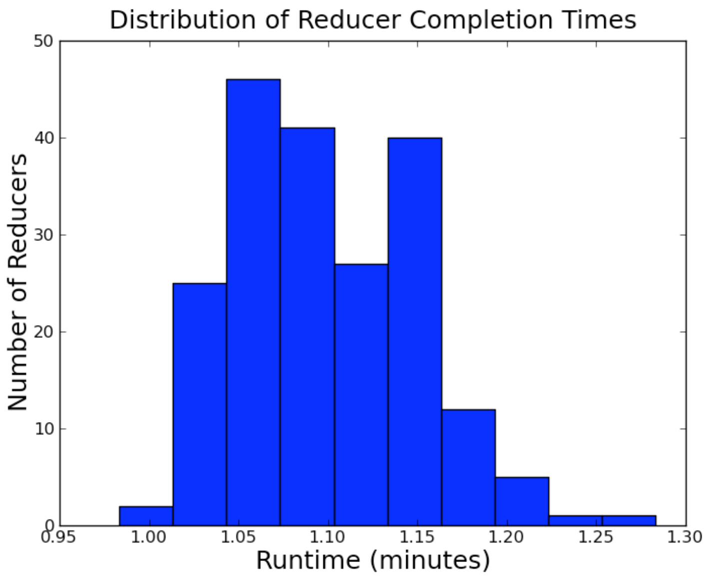
Data Set	Runtime (min) NodeItr	Runtime (min) NodeItr++	Runtime (min) GraphPartition
web-BerkStan	752	1.8	1.7
as-Skitter	145	1.9	2.1
Live Journal	59.5	5.3	10.9
Twitter	?	423	483

- Model does not differentiate between rounds when they are both constants.

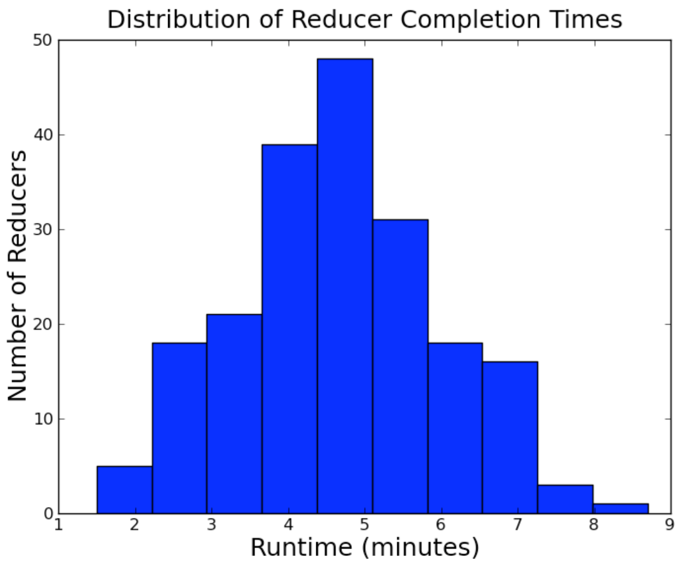
The Curse of the Last Reducer



NodeItr



NodeItr++



GraphPartition

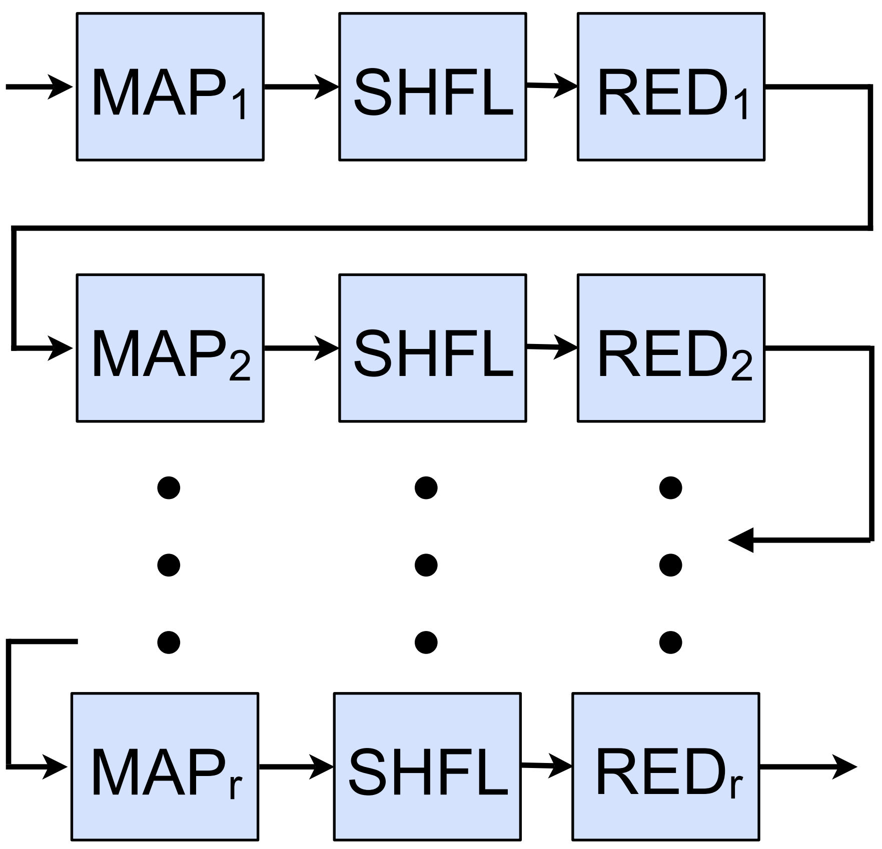
- LiveJournal data
- NodeItr++ and GraphPartition deal with skew much better then NodeItr

What do Algorithms Say About MRC?

- Model indicated shuffling m^2 bits is too much but $m^{1.5}$ bits is not, this was accurate
- Rounds can take a long time
  - GraphPartition only had a constant factor blow up in amount shuffled, still took 8 hours on Twitter
  - Need to strive for constant round algorithms
- Two round algorithm took as long as one round algorithm
  - Streaming on the reducers can be more efficient then loading subgraph into memory
  - Differentiating between constants is too fine grained for model

MapReduce: Future Directions

- Lower bounds: show that a certain problem requires Ω(log n) rounds
  - What is the structure of problems solvable using MapReduce?
- Space-time tradeoffs
  - time: number of rounds
  - space: number of bits shuffled
- MapReduce is changing, can theorists inform its design?



Thank You!

Siddharth Suri

- Yahoo! Research