A Study of Skew in MapReduce Applications

YongChul Kwon

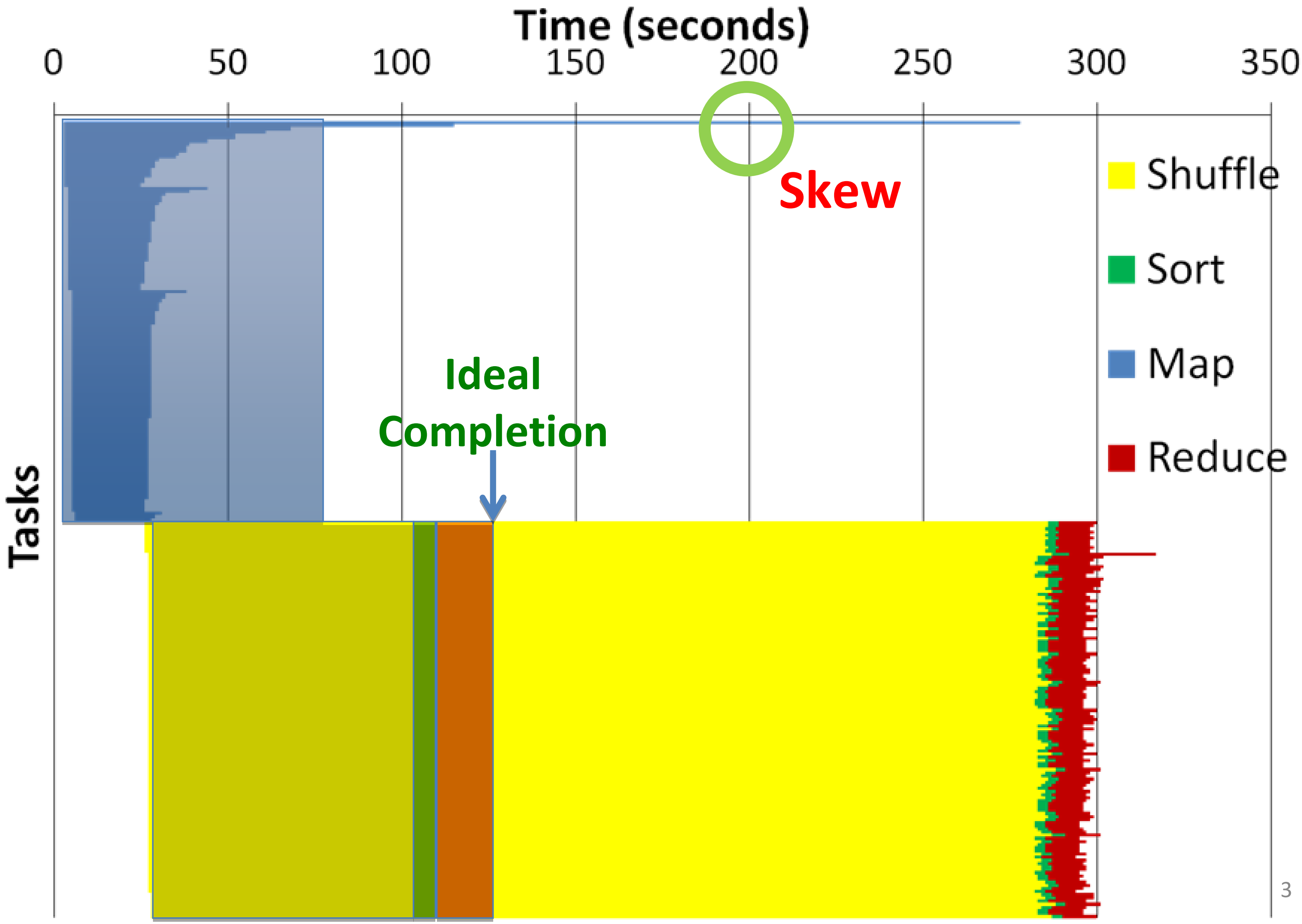
Magdalena Balazinska, Bill Howe, Jerome Rolia*

University of Washington, *HP Labs

Motivation

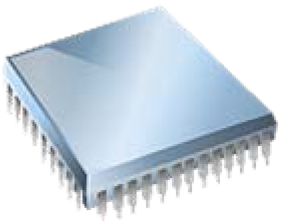
- MapReduce is great
  - Hides details of distributed execution
  - Simplifies writing distributed tasks
- Democratizes large scale data analysis
  - Domain experts (scientists, business analysts, …)
- Difficult to optimize MapReduce applications
  - Skew is one of such challenges

Problem: Skew in PageRank

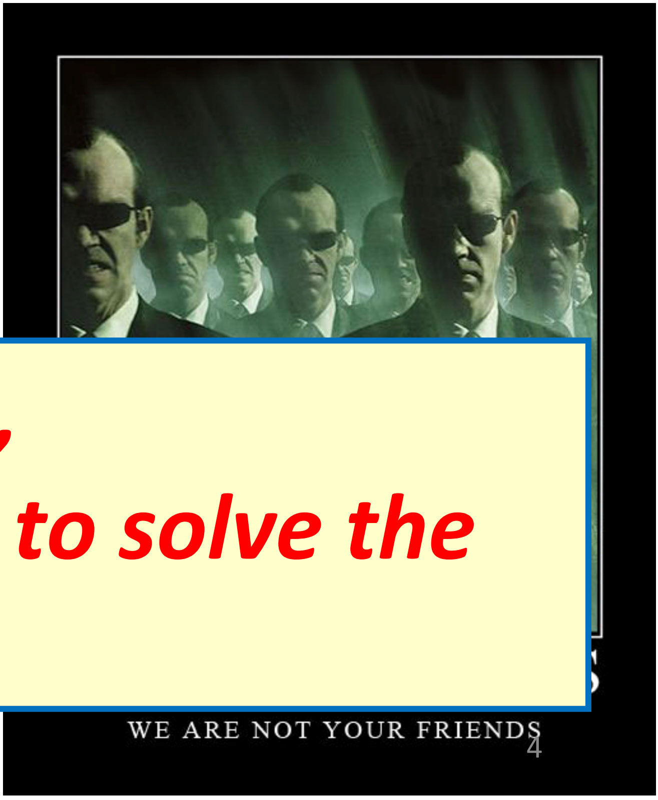


Why does it run slow?

- H/W problem



- Workload interference
  - –Your friendly neighbor random Joe



Solution: Speculative Execution

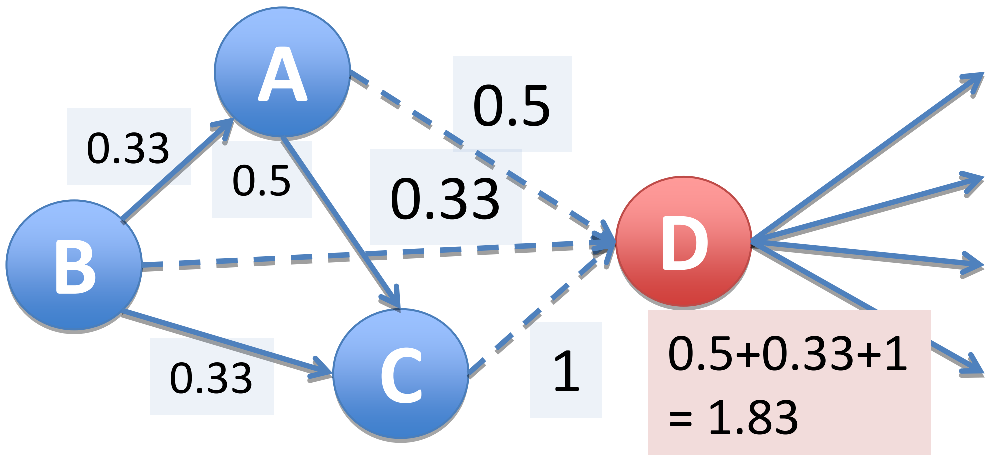
But sometimes,
speculative execution fails to solve the problem!

Survey of Skew

- Map-side
  1. Expensive Record (e.g., PageRank)
  2. Heterogeneous Map (e.g., CloudBurst)
  3. Non-homomorphic Map (e.g., Friends of Friends)

- Reduce-side
  4. Partitioning Skew (e.g., CloudBurst)
  5. Expensive Input (e.g., CloudBurst)

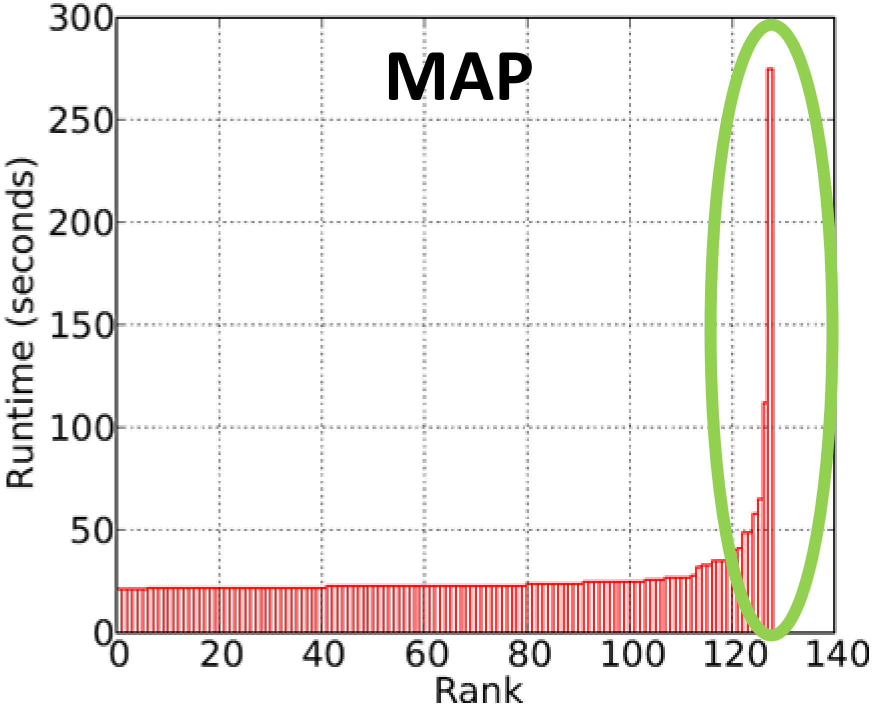
Case Study: PageRank

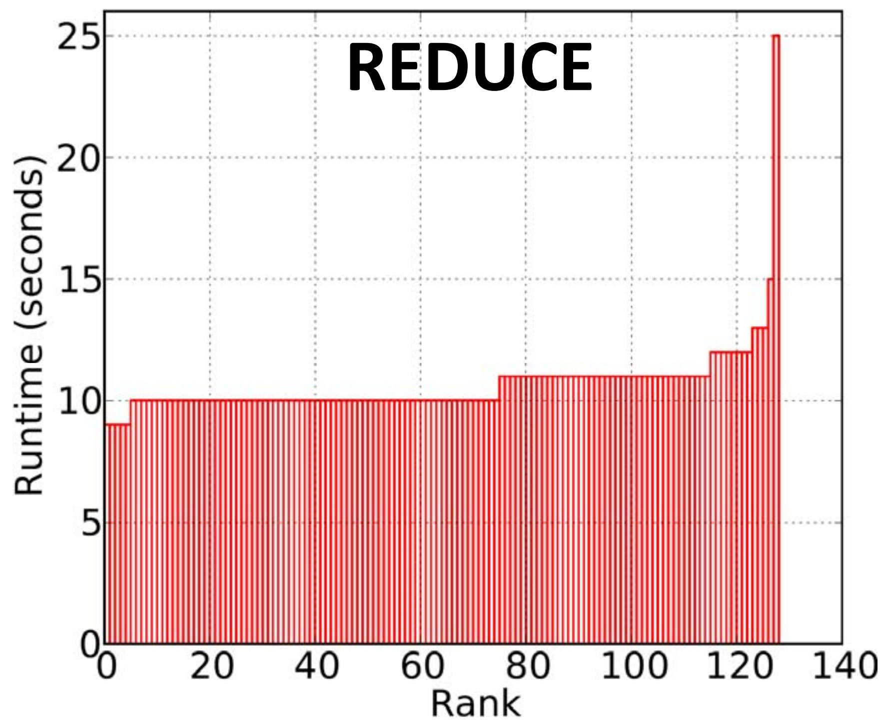


- Famous link analysis algorithm
  - Cast weighted vote along outgoing edges
  - Aggregate the votes and update the rank
- MapReduce conversion
  - Map: send out fractional PageRank along all out edges
  - Reduce: aggregation and update PageRank

Implementation: Cloud 9 – http://www.umiacs.umd.edu/~jimmylin/Cloud9/docs/index.html

PageRank: Task Runtime





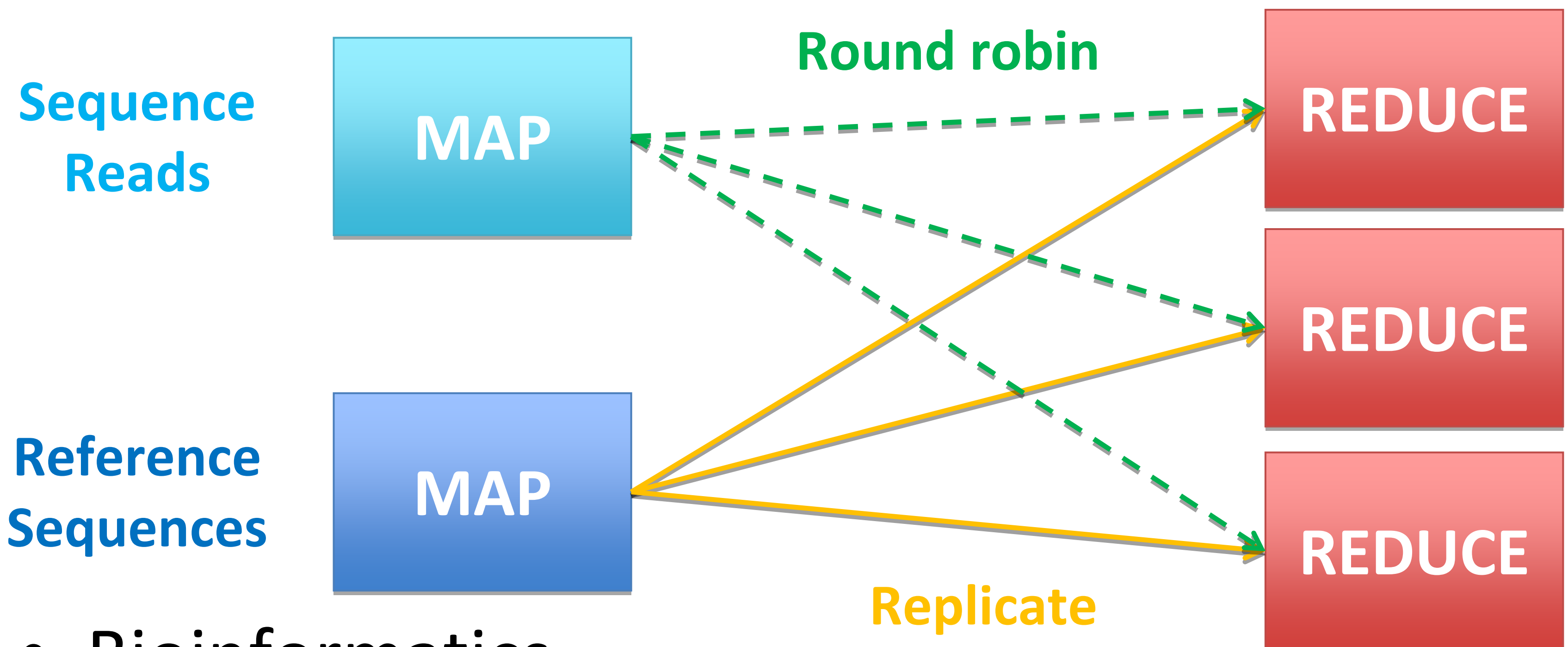
- Some records take longer to process
  - A large number of outgoing edges
  - Yields large output, more spills to disk

Configuration: 32 machines, 32/32 map/reduce slots, 128/128 map/reduce tasks

Skew Type 1: Expensive Record

- Cause
  - Some input record is taking longer to process than others
- Best practice
  - Use domain knowledge
    - Which record is expensive?
  - Pre-process input and partition
    - Isolate expensive records

Case Study: CloudBurst



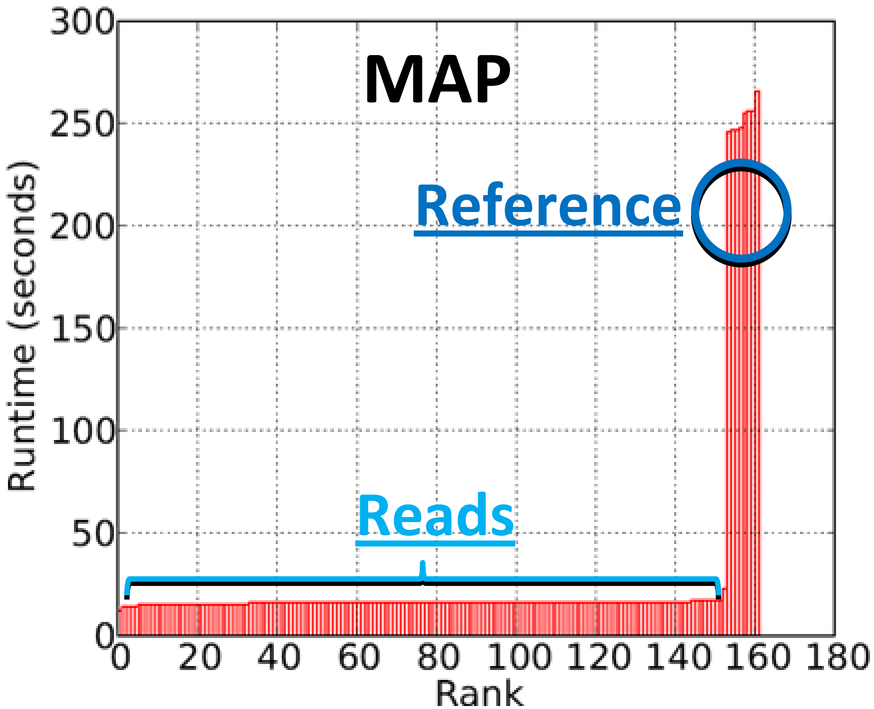
- Bioinformatics
  - ***Approximately align*** genome **sequence reads** along known **reference sequences**

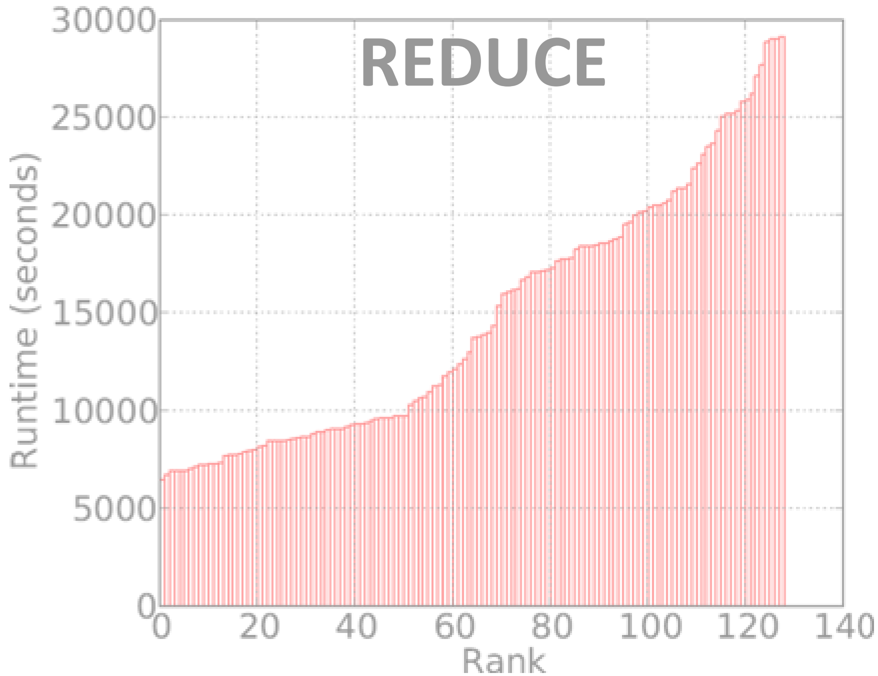
Similarity string matching

Two Input Datasets

Implementation: CloudBurst – http://cloudburst-bio.sourceforge.net

CloudBurst: Task Runtime





- Two code paths to process two datasets in map()
- Within a dataset, there is no skew

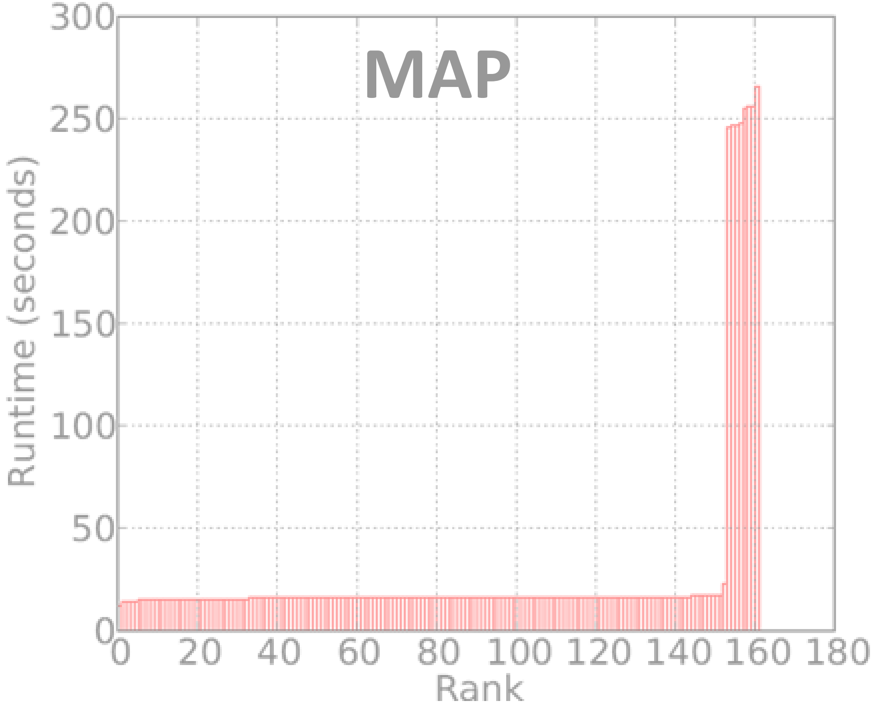
Configuration: 32 machines, 32/32 map/reduce slots, 160 map tasks, 128 reduce tasks

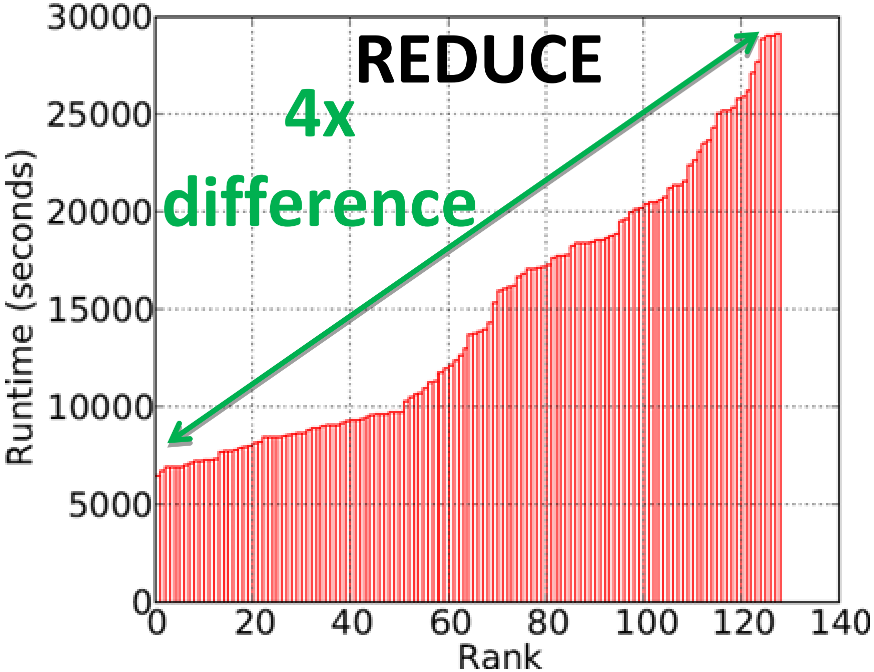
Skew Type 2: Heterogeneous Map

- Cause
  - More than one map() function in a job
  - Each map() function has different performance characteristics

- Best practice
  - Use domain knowledge
    - Determine appropriate # of map tasks per map()
  - Pre-process input and partition
    - If necessary

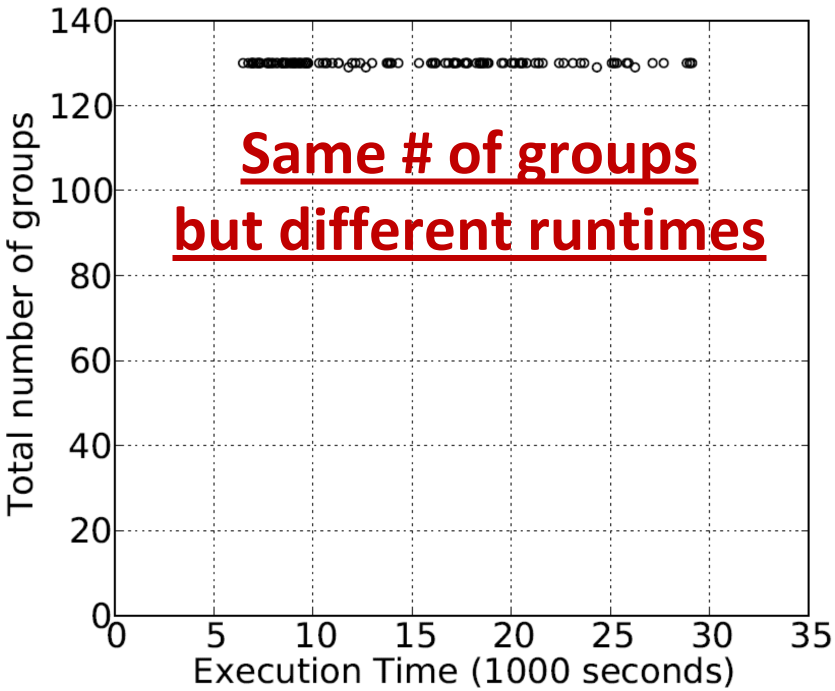
CloudBurst: Task Runtime

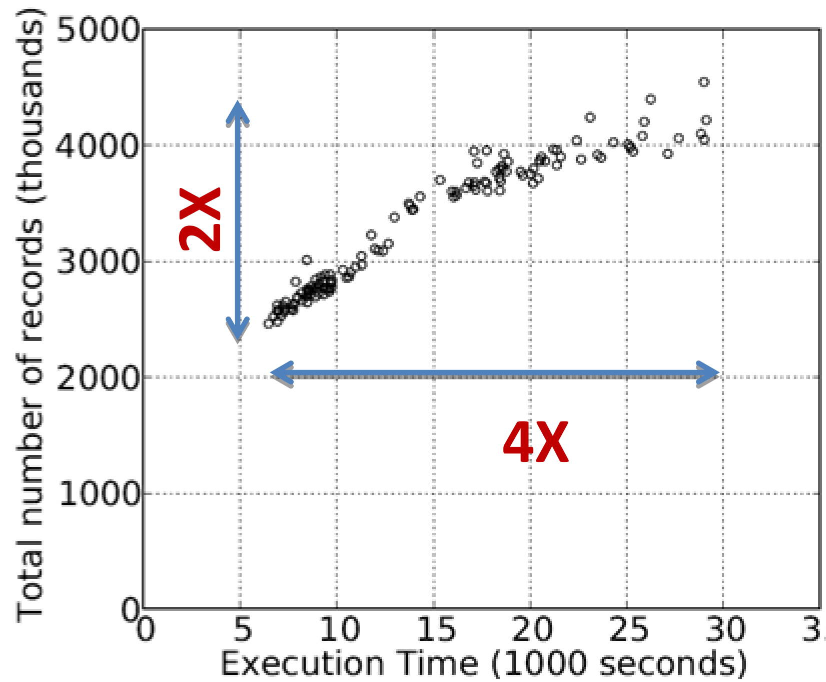




- Smooth distribution of task runtime
- Factor of 4 difference between the fastest and the slowest

CloudBurst: What's Happening in Reduce?





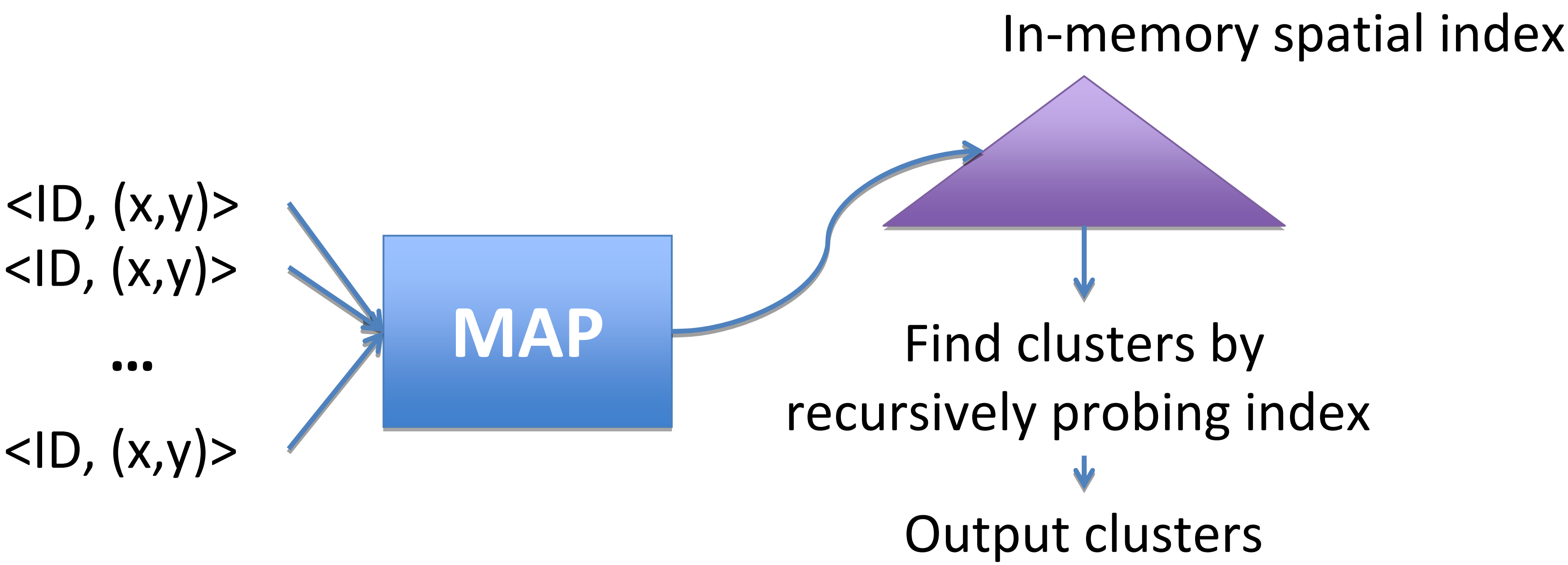
- # of reduce keys
  - Same number
  - Factor of 4 difference in runtime
- # of input records
  - Factor of 2 difference
  - ***Does not account for 4x difference in runtime!***

Reduce Skews

- Skew Type 4: Partitioning Skew
  - Cause: Some reduce tasks receive more input data
- Skew Type 5: Expensive Input Skew
  - Cause: Some reduce() take longer than others

- Best Practice
  - Use domain knowledge
  - Try different partitioning
  - Implement combiner

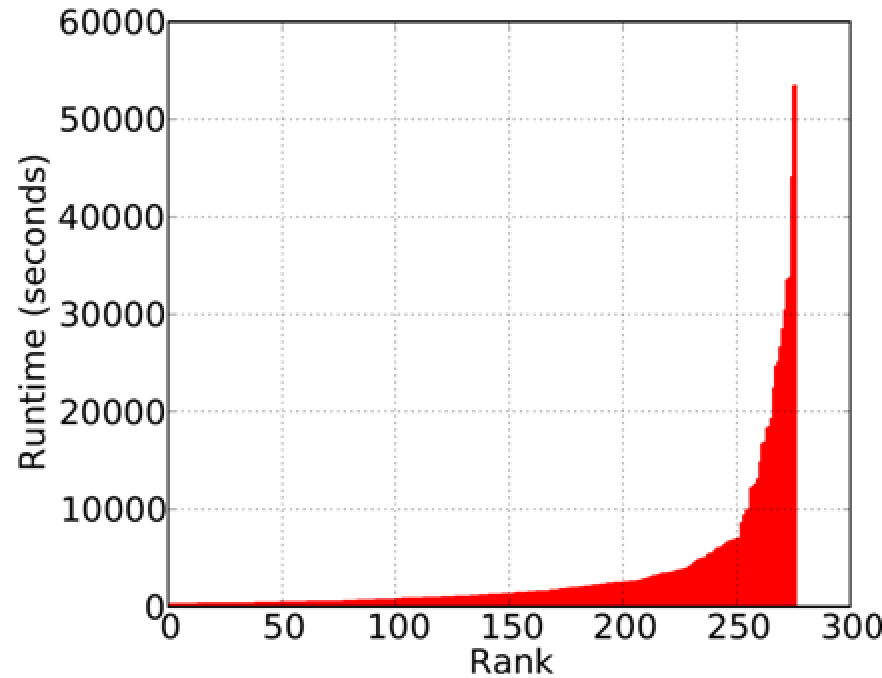
Case Study: Friends of Friends



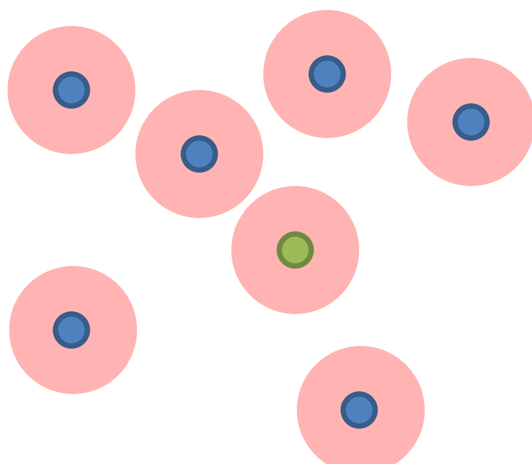
- Clustering algorithm used by astronomers
  - Friend: a point within distance threshold
  - Cluster: transitive closure of Friend from seed point
  - Requires spatial index for efficient execution

Friends of Friends: Task Runtime

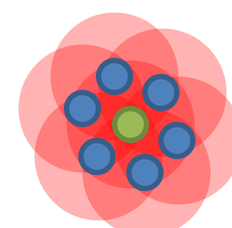
- No previous types of skew
  - Same amount of data
  - No expensive record
  - No heterogeneity



Data



O(N log N)



~ $O(N^2)$

Configuration: 32 machines, 32/32 map/reduce slots, 276 map tasks

Skew Type 3: Non-homomorphic Map

- Cause
  - The map() processes contiguous blocks of records
  - Each map task runtime depends on data value or distribution
- Best practice
  - Use domain knowledge
  - Pre-process input and partition
  - Redesign algorithm

Summary: Survey of Skew

- Map-side
  1. Expensive Record (e.g., PageRank)
  2. Heterogeneous Map (e.g., CloudBurst)
  3. Non-homomorphic Map (e.g., Friends of Friends)

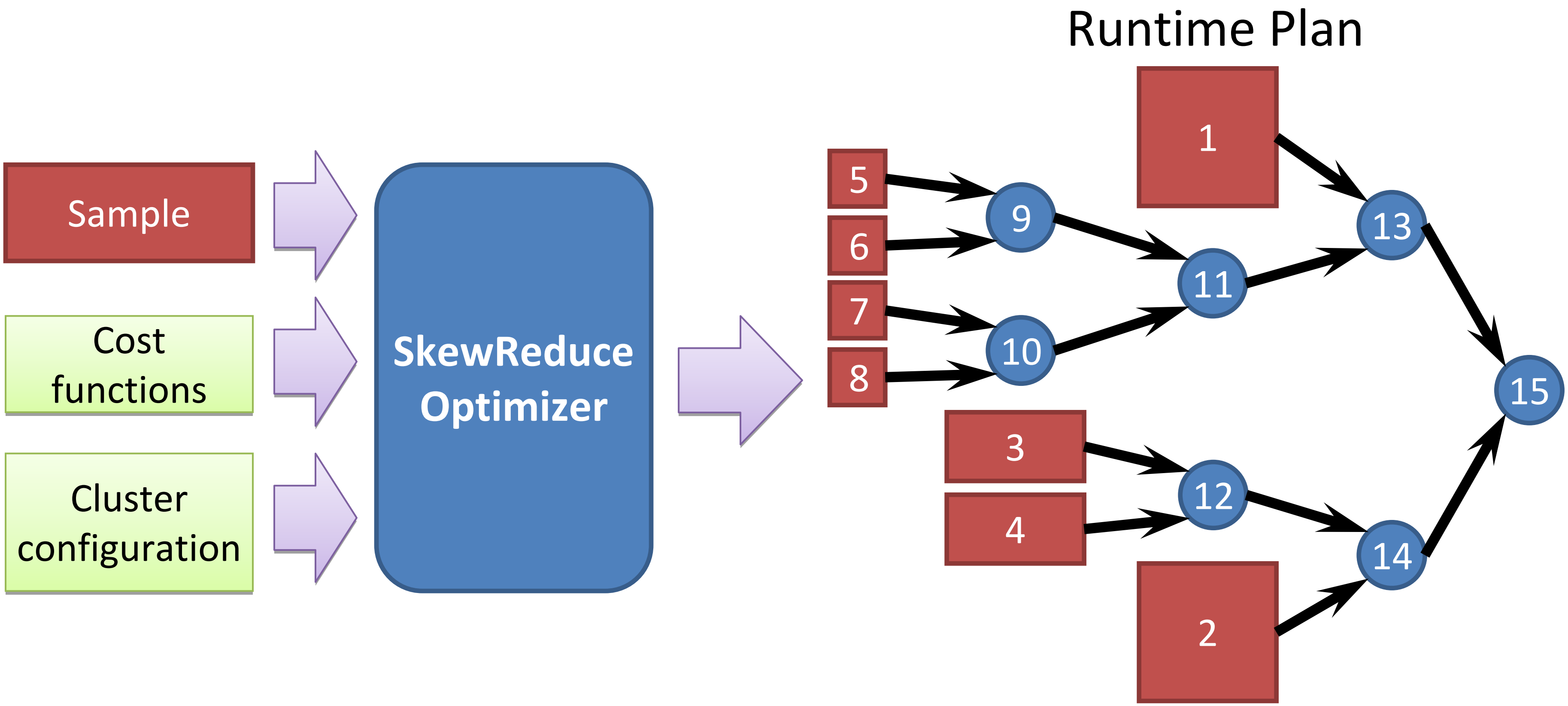
- Reduce-side
  4. Partitioning Skew (e.g., CloudBurst)
  5. Expensive Input (e.g., CloudBurst)

[SOCC2010]

SkewReduce

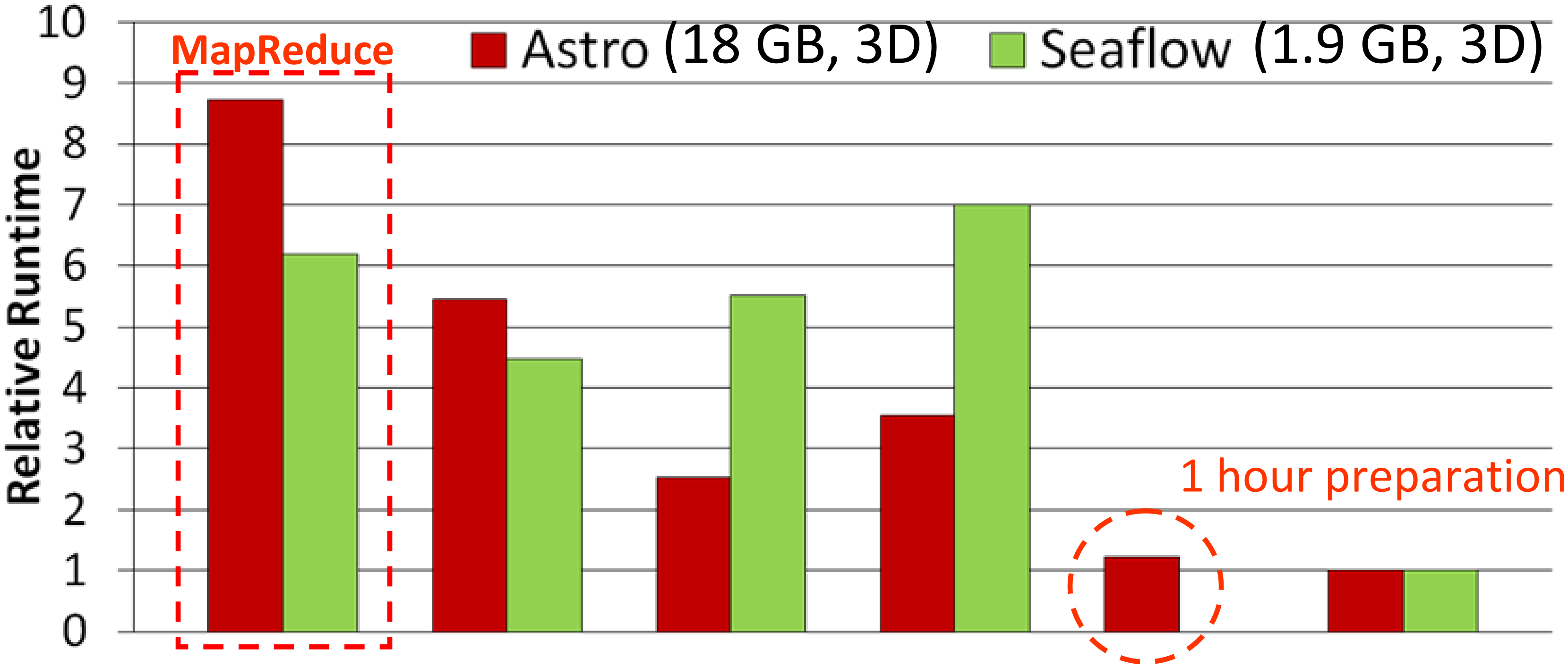
- Can a system automatically derive a good data partitioning?
- Domain: feature extracting application
  - But applicable if the computation could be hierarchically decomposable
- Optimizer + Runtime
- **http://code.google.com/p/skewreduce**

SkewReduce: Approach



- ***Goal**: minimize expected total runtime*
- SkewReduce runtime plan
  - Bounding boxes for data partitions
  - Schedule

Does SkewReduce work?



128 MB	16 MB	4 MB	2 MB	Manual	SkewReduce	
14.1	8.8	4.1	5.7	2.0	1.6	Hours
87.2	63.1	77.7	98.7	-	14.1	Minutes

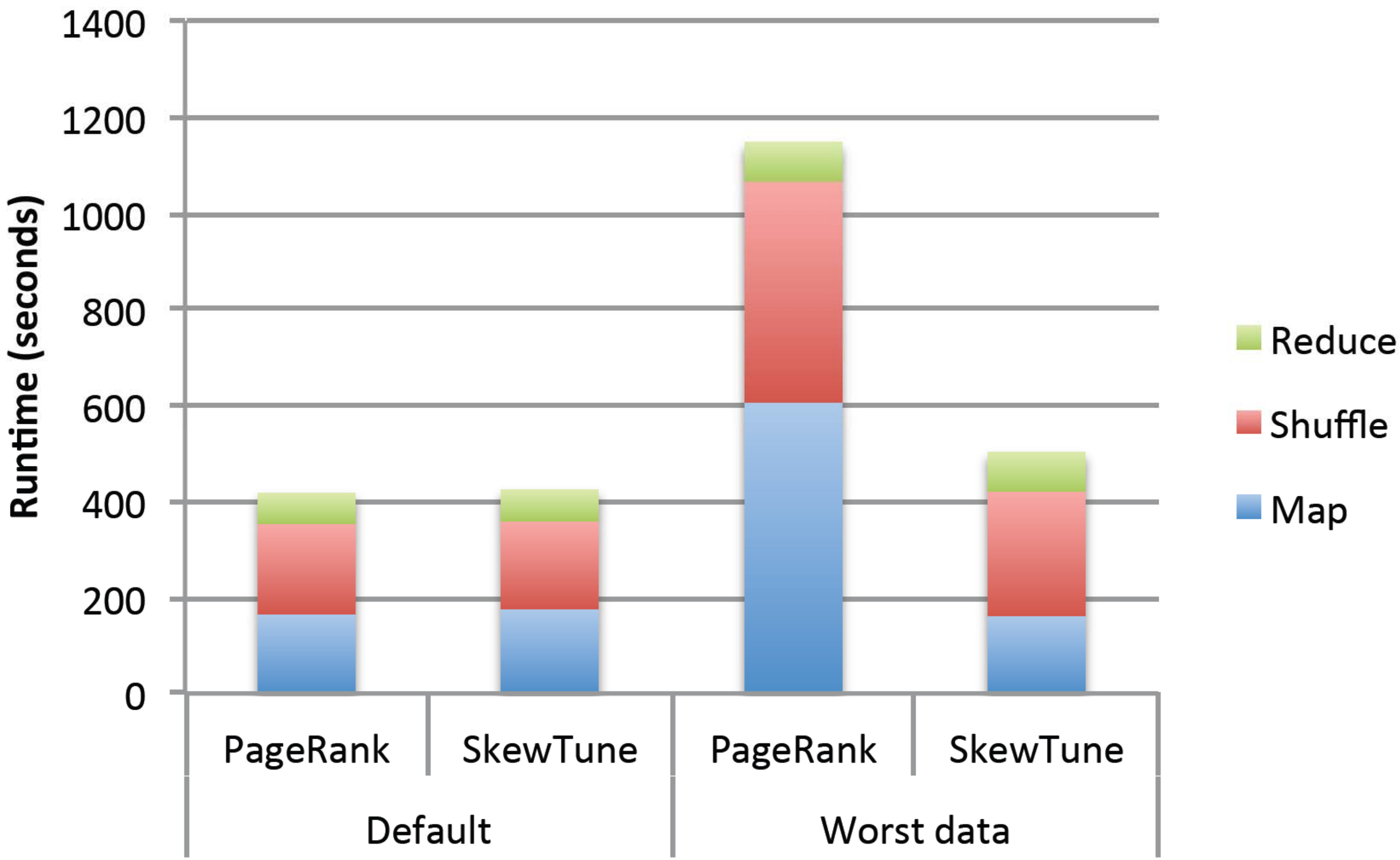
- SkewReduce plan yields 2 ~ 8 times faster running time

[Submitted to SOCC2011]

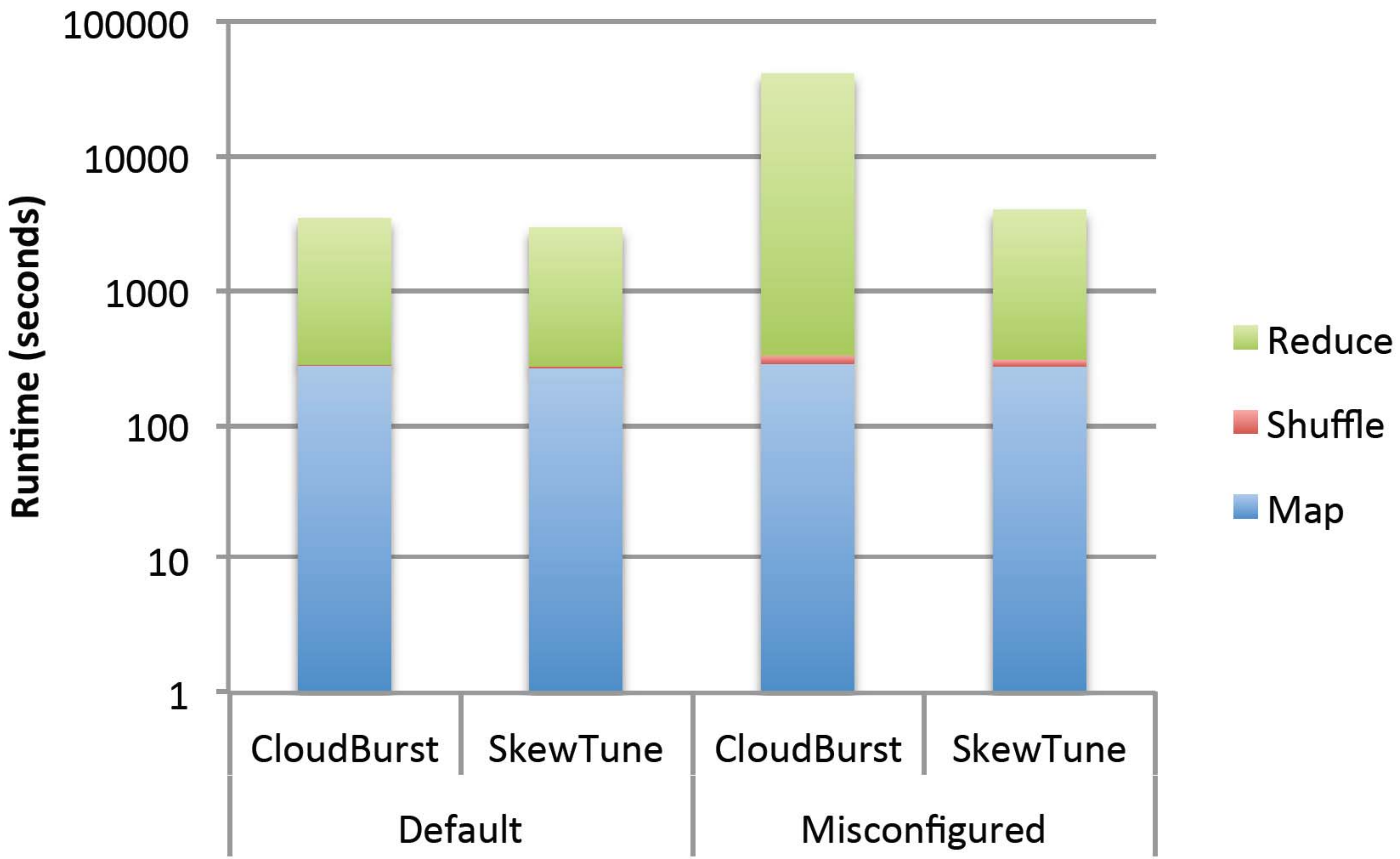
Future Work: SkewTune

- How far can we automate?
- Analyze a MapReduce application
  - Is map/reduce prone to skew?
  - Is map/reduce repartitionable?
- Accelerate the slowest task
  - Aggressively repartitioning the input data

SkewTune: PageRank



SkewTune: CloudBurst



Conclusion

- Grand Vision
  - Open-up large-scale data analysis to domain experts
- In this talk
  - Showed skew problems in MapReduce applications
  - Our efforts to mitigate the impact of skew
- If you have an interesting application, please let us know!