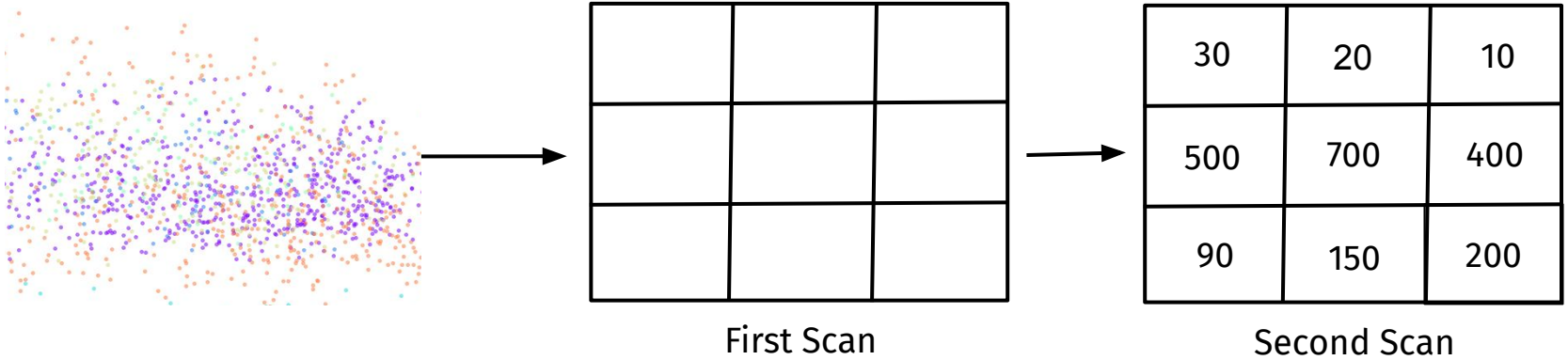
Flexible Computation of Multidimensional Histograms

Samriddhi Singla, Ahmed Eldawy
University of California, Riverside

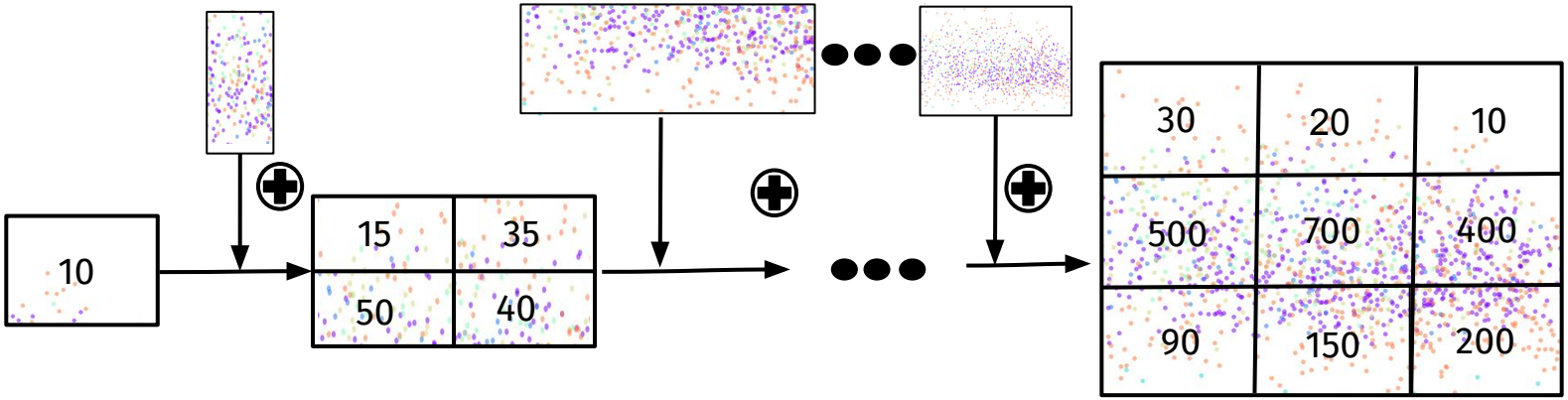
Spatial Gems 2020, November 3

Motivation



- Histograms are used to facilitate
  - query optimization,
  - approximate query processing, and
  - load balancing.
- Computed by scanning the data twice.
  - First scan to compute the minimum bounding box (MBB) of the data and the bucket boundaries
  - Second scan to assign the data points to their respective buckets.

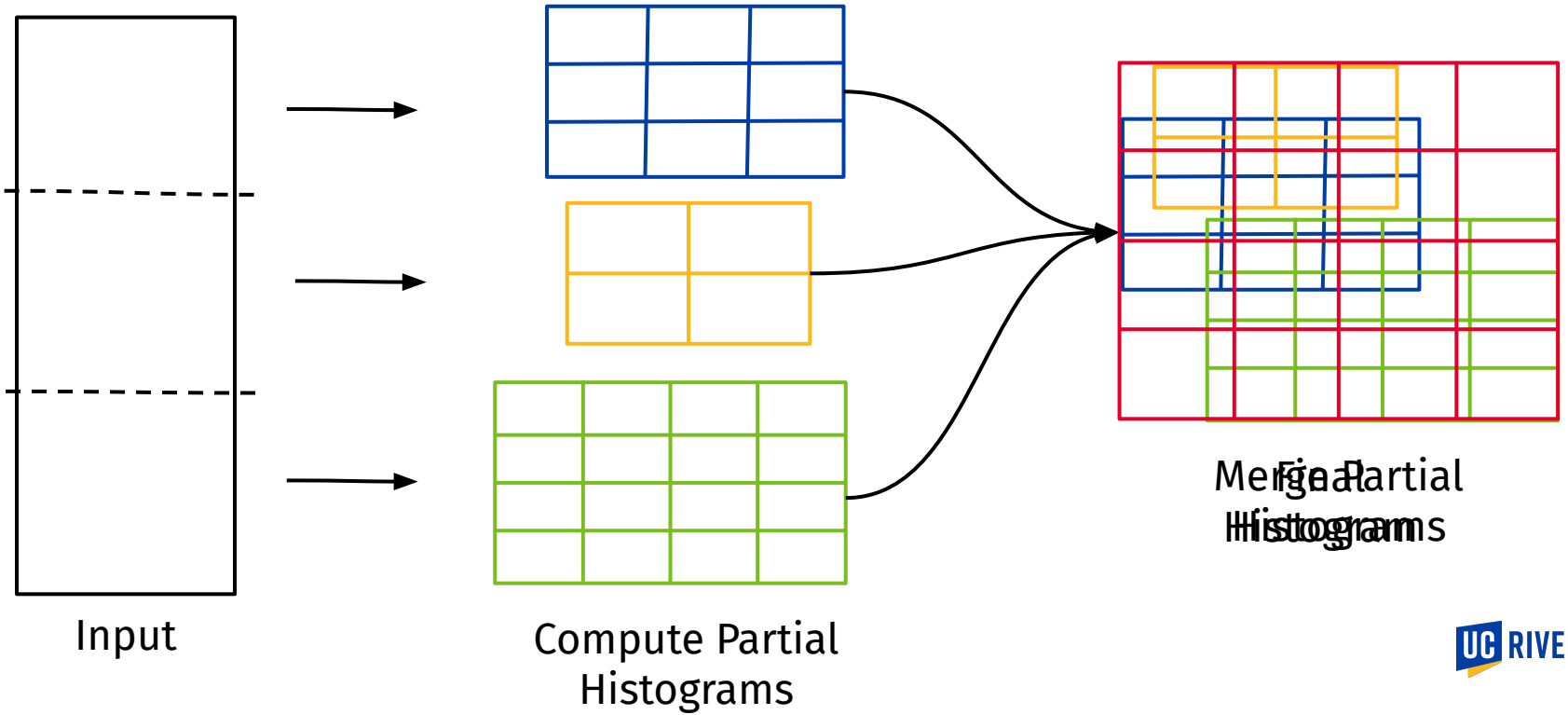
Motivation



- Difficult to compute histogram for streaming applications
  - data distribution changes frequently
  - histogram requires to be computed from scratch repeatedly
- For big data applications that use HDFS or LSM-based systems
  - data is stored in batches
  - data needs to be scanned twice to compute a histogram.

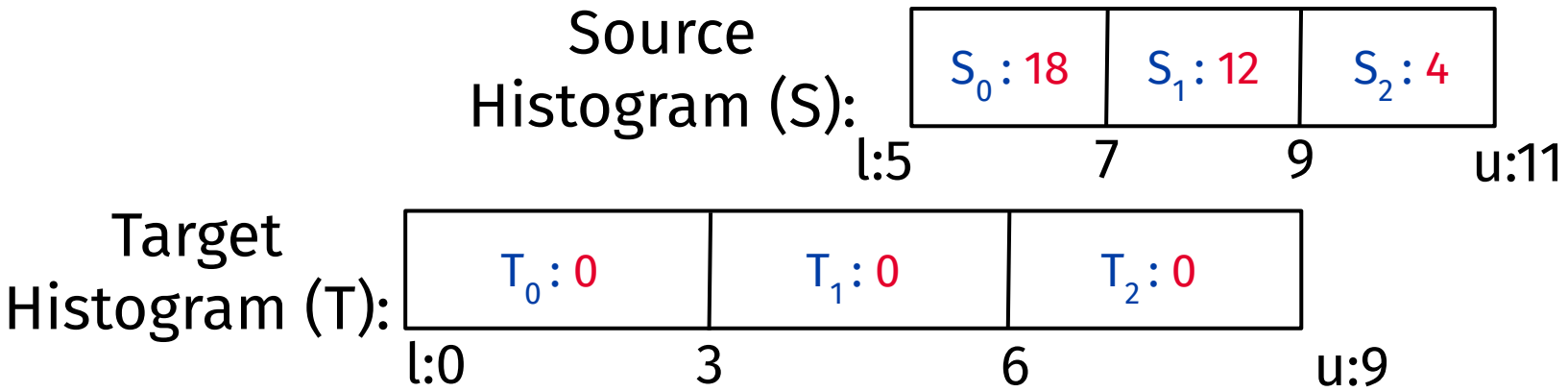
Proposed Algorithm

Computes approximate multidimensional histograms in only one scan.

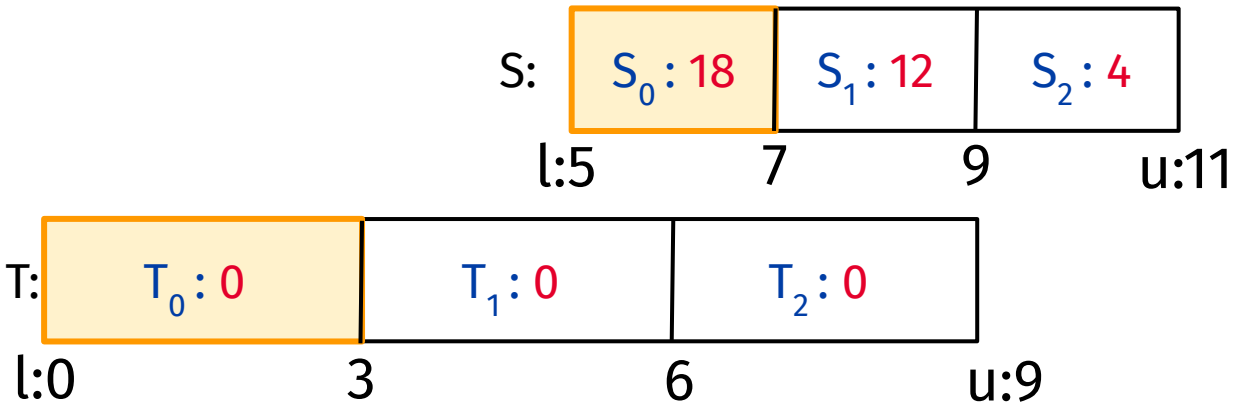


Proposed Algorithm

- Sort-Merge-like algorithm
- Histogram buckets have a bounding box
  - two corner points l and u
  - row-order-like sort
- Uses overlap volume as a weight
- Demonstrate it using 1-D histograms (for simplicity)

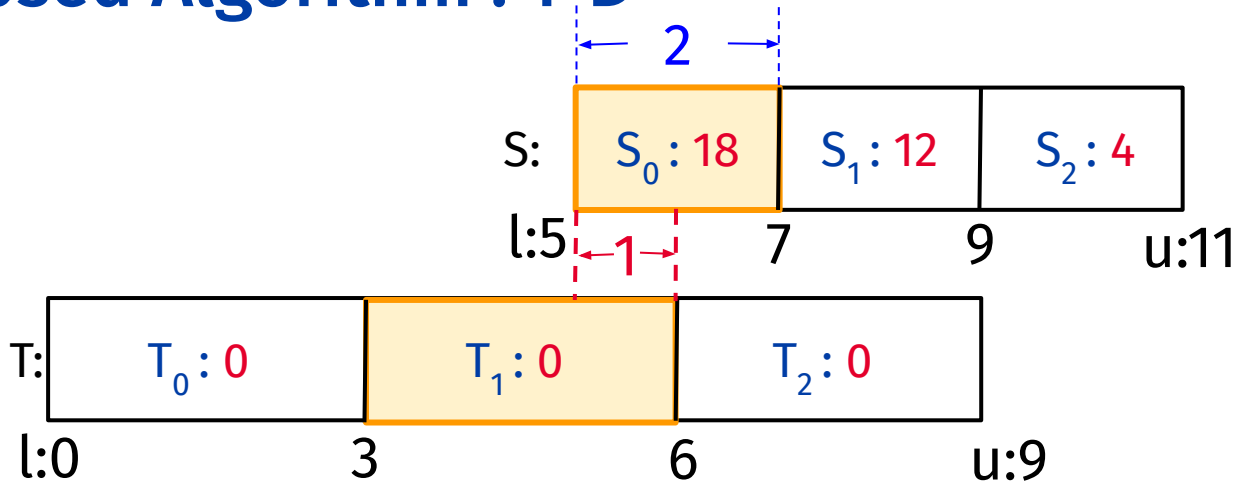


Proposed Algorithm : 1-D



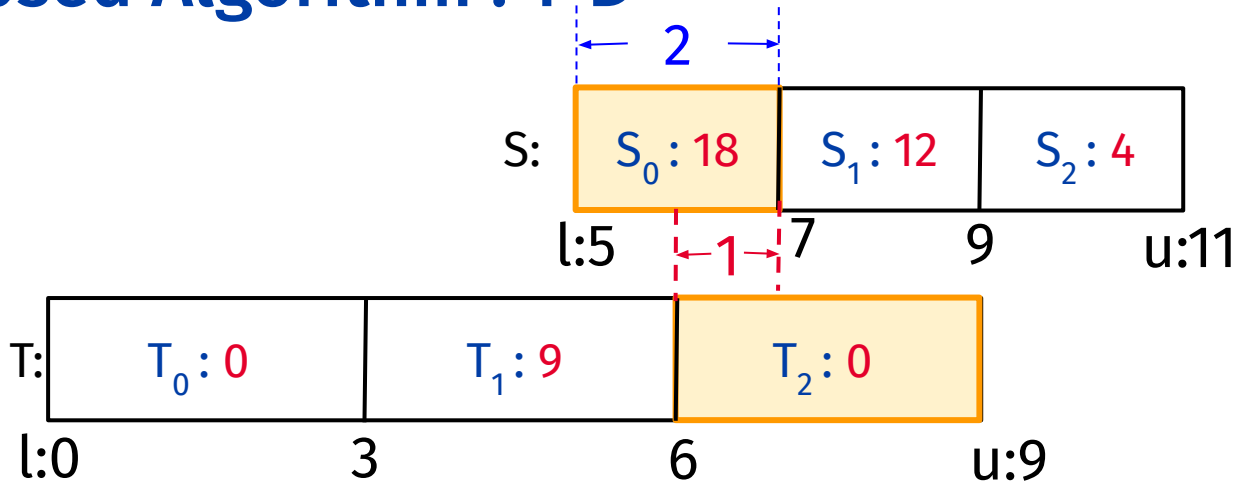
Overlap = 0, No change

Proposed Algorithm : 1-D



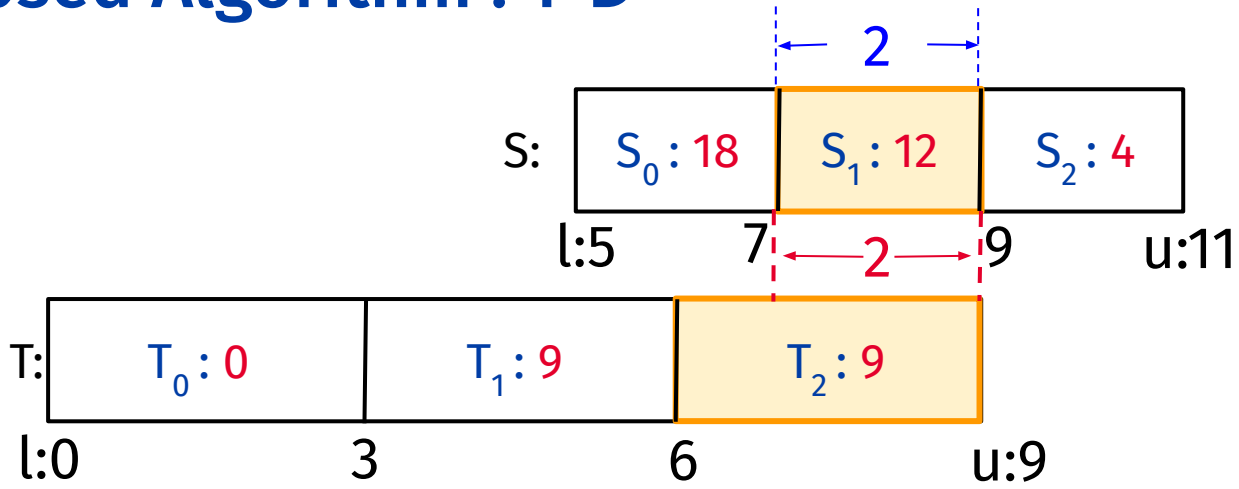
Overlap = 1/2

Proposed Algorithm : 1-D



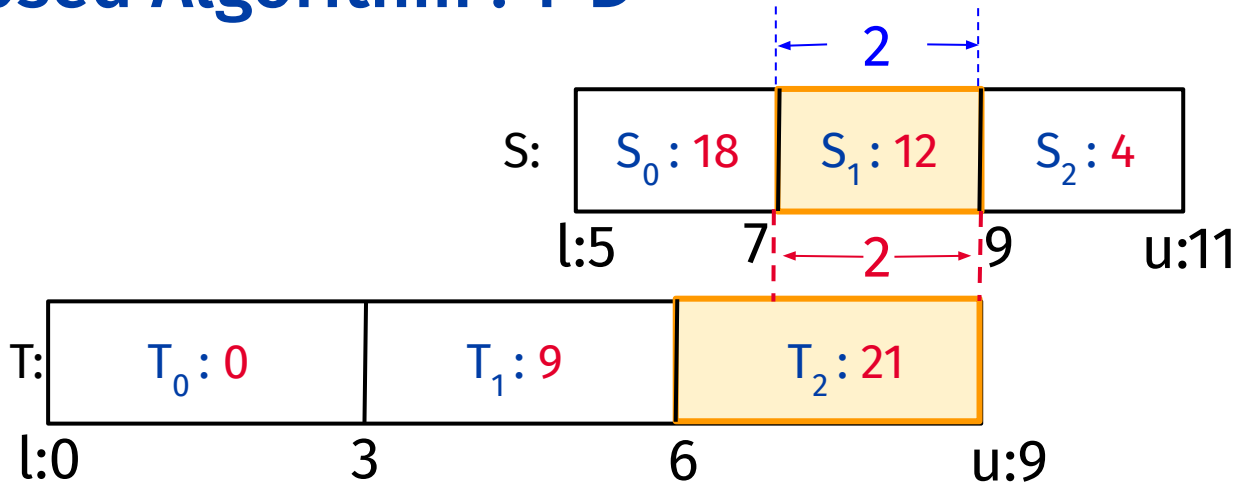
Overlap = 1/2

Proposed Algorithm : 1-D



Overlap = 2/2 = 1

Proposed Algorithm : 1-D



Overlap = 2/2 = 1

Proposed Algorithm : Multidimensional

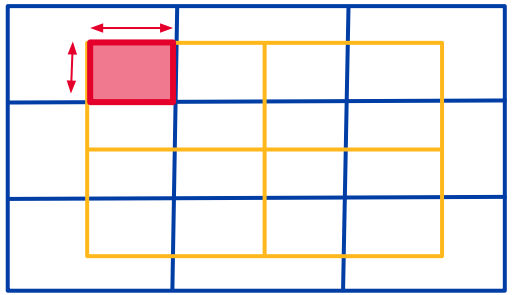
(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

- Buckets are iterated over using an array instead of a single value.
- Overlap is calculated along each dimension.

$$Overlap = \prod_{k=1}^{d} \frac{Min(u(S,i_k)_k, u(T,j_k)_k) - Max(l(S,i_k)_k, l(T,j_k)_k)}{u(S,i_k)_k - l(S,i_k)_k}$$

- For more details, please refer to the paper.

Proposed Algorithm : Multidimensional

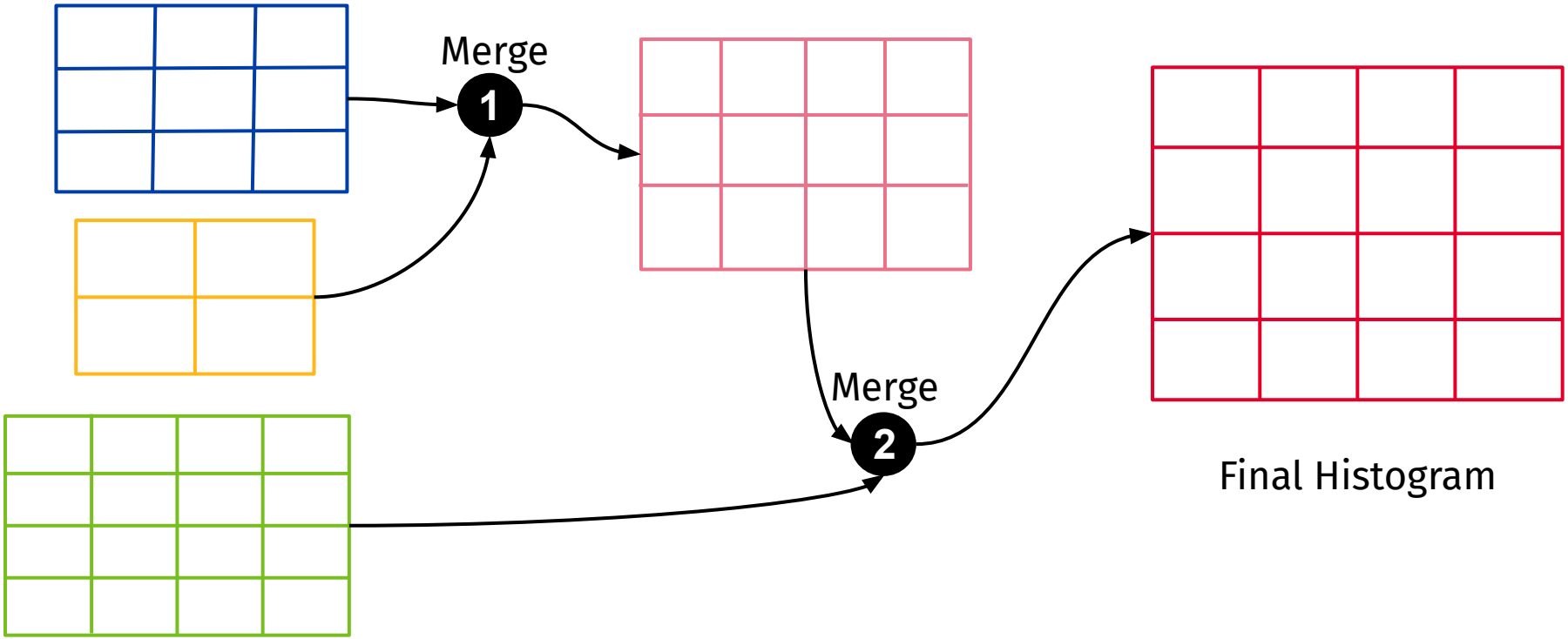


- Buckets are iterated over using an array instead of a single value.
- Overlap is calculated along each dimension.

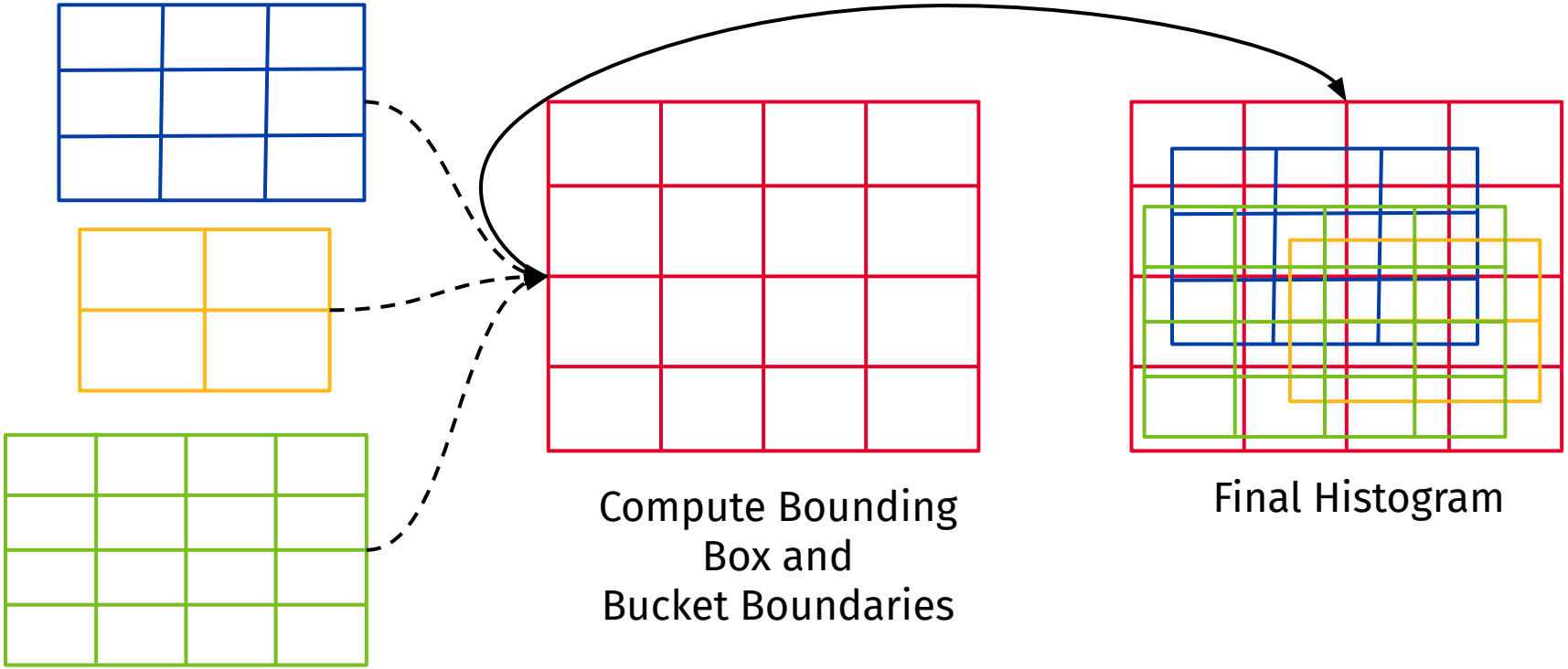
$$Overlap = \prod_{k=1}^{d} \frac{Min(u(S,i_k)_k, u(T,j_k)_k) - Max(l(S,i_k)_k, l(T,j_k)_k)}{u(S,i_k)_k - l(S,i_k)_k}$$

- For more details, please refer to the paper.

One-pass Merge for Partial Histograms



One-1/2-pass Merge for Partial Histograms

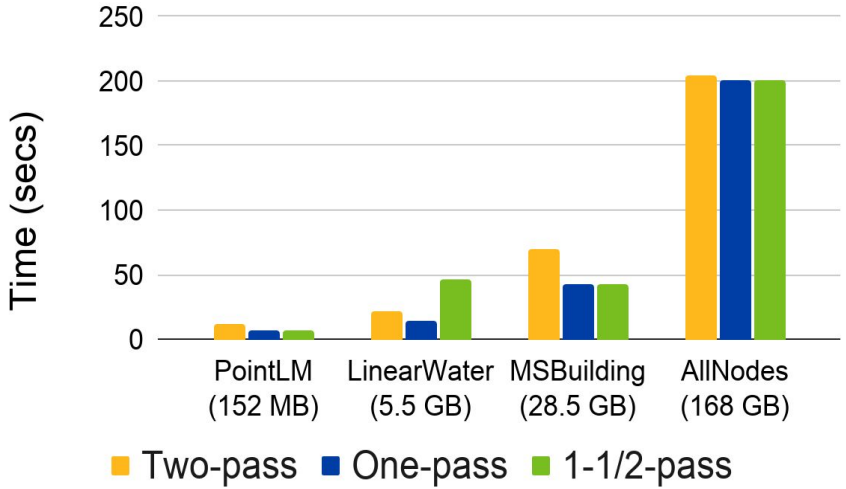


Experimental Setup

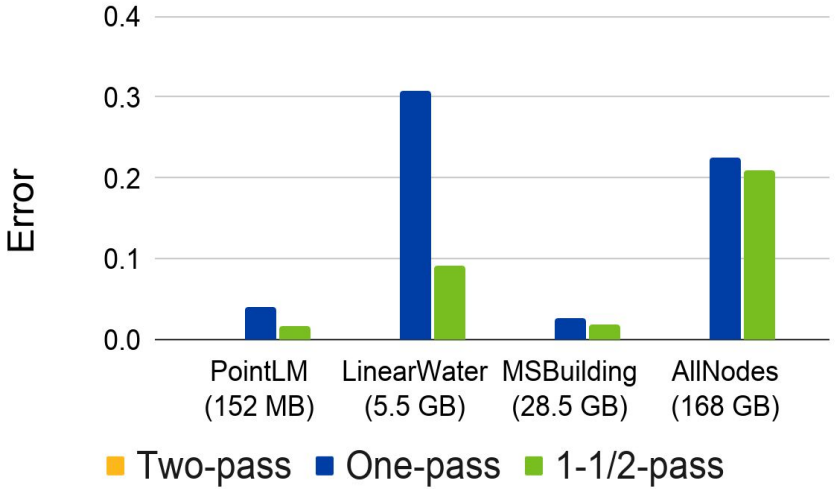
- Implemented in Spark
- Use real datasets of upto 168 GB in size
- To observe accuracy and execution time.
  - Error metric: normalized SAE (sum of absolute errors)
- Runs on a cluster of 12 machines
  - Each machine has 64 GB of RAM 2x6-cores
- Code is publicly available at : https://bitbucket.org/eldawy/beast/
  Class: UniformHistogram

Experiments

Running Time



Error



Reviewer's Comments

- The algorithm computes an approximate histogram, which needs to be made clear in the introduction.
- Add an example for non-overlapping or partially overlapping buckets.
- Theoretical bound on error. (Future Work)


Thank You!
Questions?
UC RIVERSIDE