Groupwise Analytics via Adaptive MapReduce

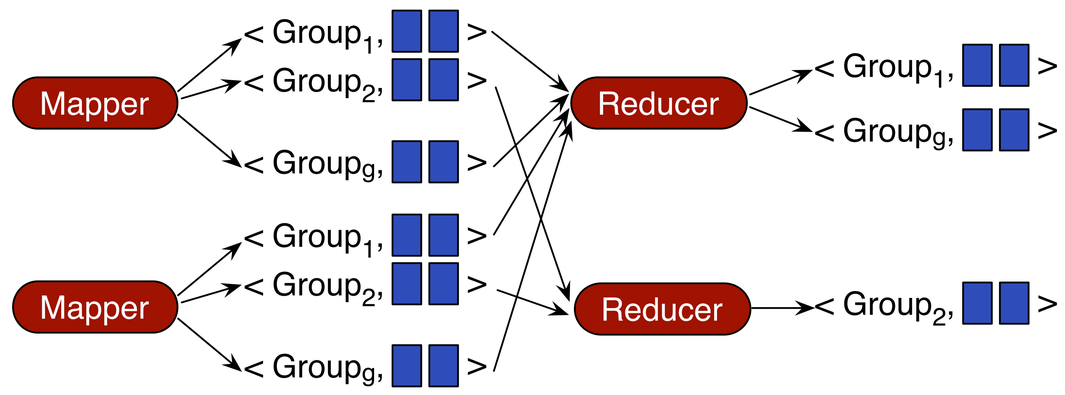
Liping Peng, Kai Zeng, Andrey Balmin, Vuk Ercegovac, Peter J. Haas, Yannis Sismanis

Groupwise Set-Valued Analytics

- Micromarketing, fraudulent transaction detection, .etc
- SELECT synopsis(*k*)
  FROM dataset
  GROUP BY strata;
  - Stratified top-k
  - Stratified bottom-k
  - Stratified sampling

Buffered MapReduce for Bottom-*k* Query

- Mapper: run bottom-*k* algorithm and emit a local synopsis of *k* records per group
- Reducer: collect all synopses of the same group and merge into a global synopsis

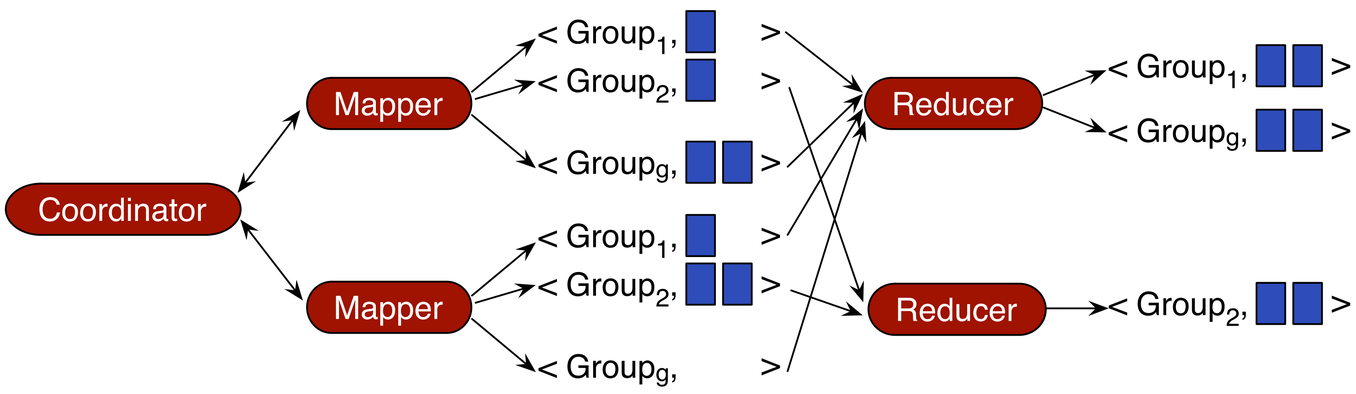


- O(*gkm*) records are shuffled
- Memory consumed at each mapper is O(*gk*)

g = # groups
m = # mappers

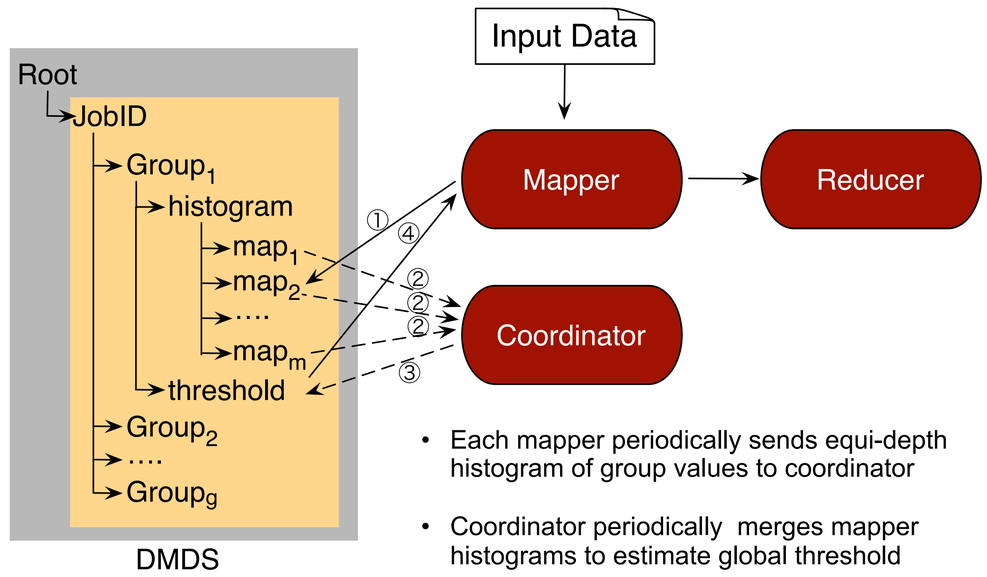
Groupwise Analytics Running on Adaptive MapReduce

- Global threshold $w_{i,(k)}$
  - The *k*-th smallest weight for all records in group G_i
  - Each mapper maintains the set of records with weights no larger than $w_{i,(k)}$
  - The number of shuffled records can be reduced from O(*gkm*) to O(*gk*)



- The coordinator communicates with all mappers, thus has a more accurate view of the global threshold than mappers
- The coordinator periodically tells the mappers its view of the global threshold which mappers can use to pre-filter local samples

Asynchronous Coordination with DMDS

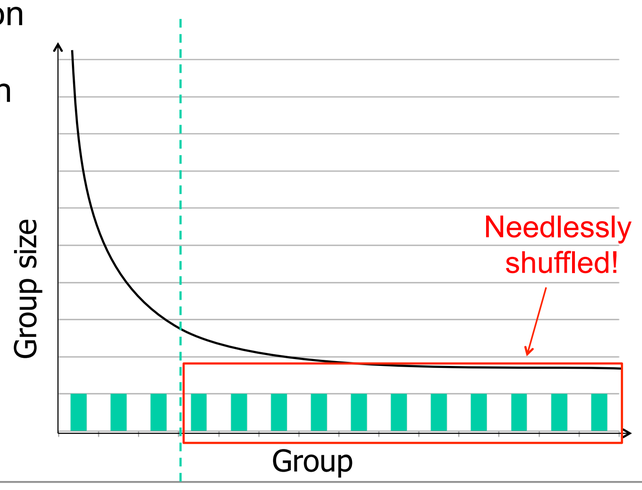


- Each mapper periodically sends equi-depth histogram of group values to coordinator
- Coordinator periodically merges mapper histograms to estimate global threshold

Top-r Stratified Sampling

Maintain a uniform random sample of size *k* for each of the *r* largest groups in {G_1, …, G_g} as in GARAM

- In micro-marketing applications, focus on the *r* largest (age, zip code) customer groups due to resource or time limitation
- SELECT sample(*k*)
  FROM dataset
  GROUP BY strata
  ORDER BY count(*)
  LIMIT *r* ;
- **Key idea:**
  GARAM + approximate thresholding



- Track running top-*r* groups using distributed algorithm of Babcock & Olston
- For running top-*r* groups, use threshold sequence generated by GARAM
- For running non-top-*r* groups, combine GARAM threshold sequence with a set of estimated thresholds {*q*}

$$\mathrm{Beta}\left(q; k, n^*_{[r]} - k + 1\right) = 1 - \epsilon$$

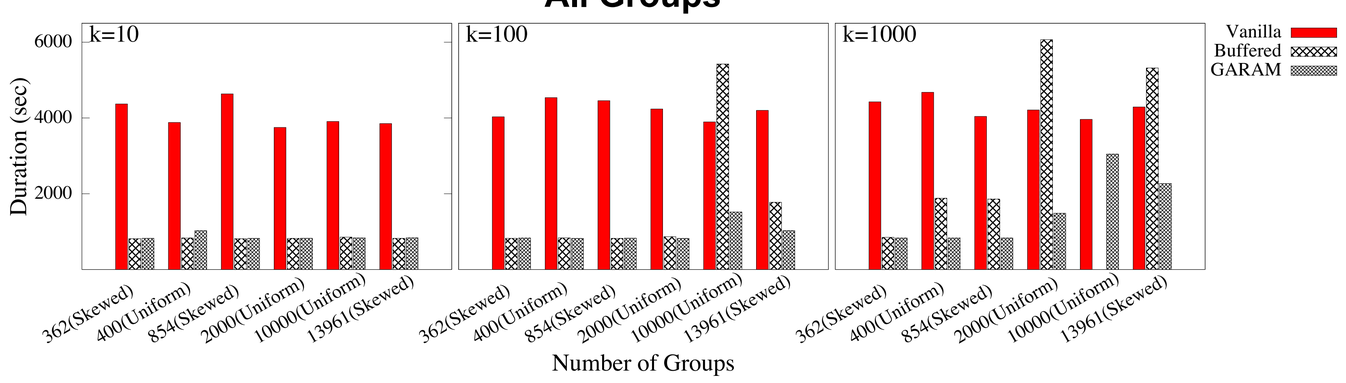
$n^*_{[r]}$ = current size of rth currently largest group

➢ For a final top-*r* group G_i that was once not in the running top-*r* groups:
- Algorithm produces a statistically correct sample for G_i
- The sample size may be less than *k* but with probability less than *ε*

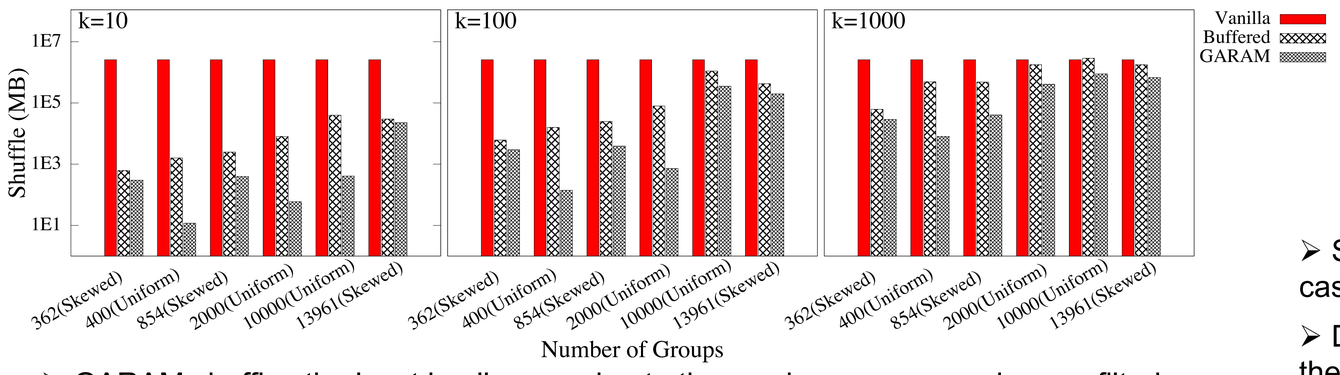
Evaluation: Stratified Sampling

- Data:
  - A table in SDSS
  - 245 columns
  - 586 million records
  - 2.45TB in HDFS
- Cluster:
  - 12 nodes with 10Gbs Ethernet
  - 12-core Intel Xeon 64-bit CPU @2.2GHz, 96GB RAM and 12 SATA disks per node
  - Hadoop v1.1.2
  - 1 node for Hadoop JobTracker for HDFS
  - 1 node for ZooKeeper Server
  - 10 workers, each with 8 map slots and 4 reduce slots

All Groups

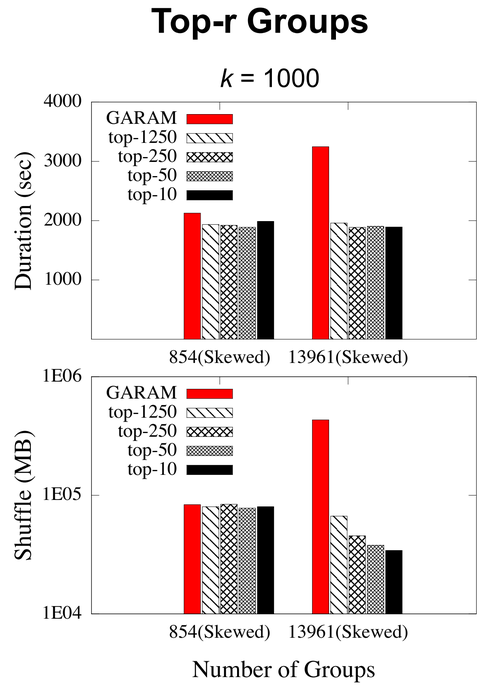


➢ GARAM is close to Buffered for small *gk* and performs the best when *gk* is large



➢ GARAM shuffles the least in all cases due to the much more aggressive pre-filtering in the mappers based on the coordination

Top-r Groups



➢ Significantly outperforms GARAM in the case of 13961 strata: 20x less data, 2x faster

➢ Does not improve much over GARAM in the case of 854 strata, but shorter duration indicates little overhead relative to GARAM