

Beyond Online Aggregation: Parallel and Incremental Data Mining with MapReduce

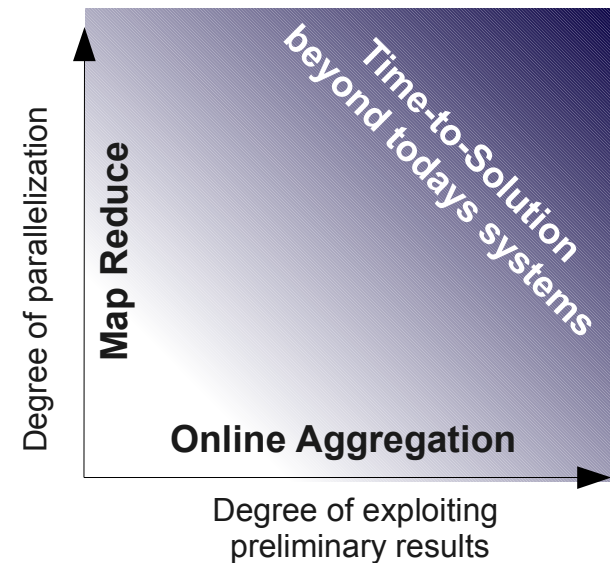
Joos-Hendrik Böse^{*}, Artur Andrzejak[♦], Mikael Höggqvist[♦]

^{*}ICSI Berkeley

[♦]Zuse Institut Berlin

Motivation & Idea

- **Exploratory data studies with MapReduce are tedious:**
 - MR Jobs are typically long running
 - User is forced to wait for final result of job before she can decide on next query
 - **But:** “speed of thought“ performance required for interactive analysis!
DSS and OLAP users are demanding
 - Exact (final) result is rarely required in non-lookup queries!
- **How to reduce Time-to-Solution?**
 - i. Increase degree of parallelization
 - ii. Exploit preliminary results, (as seen in DBMS [**Online Aggregation**])



- **Idea**
 - **Combine MR parallelization and Online Aggregation to achieve Time-to-Solutions beyond todays systems!**

Agenda

- **Online Aggregation**
- **Streaming MapReduce Framework**
- **Assess Quality of Preliminary Results**
- **Experiments for Single-Phase ML Algorithms**

- **How to implement Online Aggregation for Multi-Phase ML Algorithms**
- **Reducing K-Means to a Single-Phase Algorithm**
- **Experimental results for Online K-Means**

Online Aggregation

- Developed for large-scale data analysis on RDBMS [Hellerstein et. al 1997]
- For an SQL aggregate ***progress***, ***preliminary results***, and ***confidence intervals*** are provided during query execution
- **Puts user in the “driver seat“:**
 - can cancel futile queries prematurely
 - can stop execution if preliminary result is precise enough
- Often aggregate estimations are close to the final result after processing only a small fraction of the input data

Select **AVG(grade)** from **ENROLL**
 GROUP BY major;



Postgres95 Online Aggregation Interface

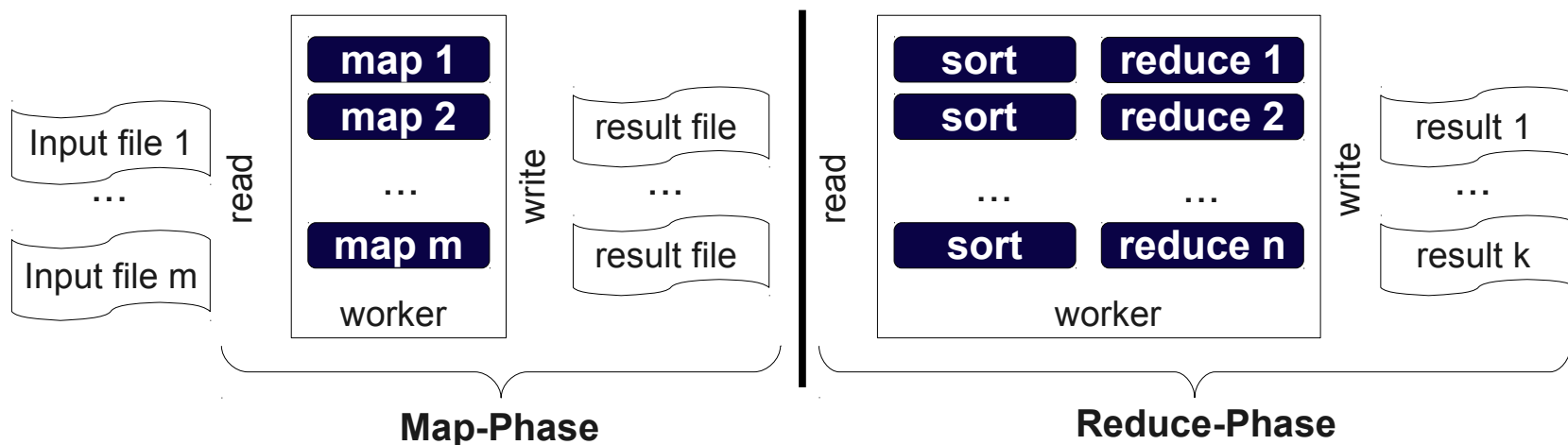
	major	AVG	Confidence	Interval
⬇	1	2.27216	95	0.160417
⬇	2	2.50148	95	0.160417
⬇	3	2.66702	95	0.160417
⬇	4	2.86235	95	0.160417
⬇	5	3.12048	95	0.160417
⬇	9	2.89645	95	0.160417

Cancel All 14% done

Online Aggregation in MapReduce

- How to implement online aggregation?
 - **Standard MapReduce system model is batch-oriented!**
 - Each operator executes completely before producing any output
 - Map-Phase must be finished before Reduce-Phase is started

Standard MapReduce system model

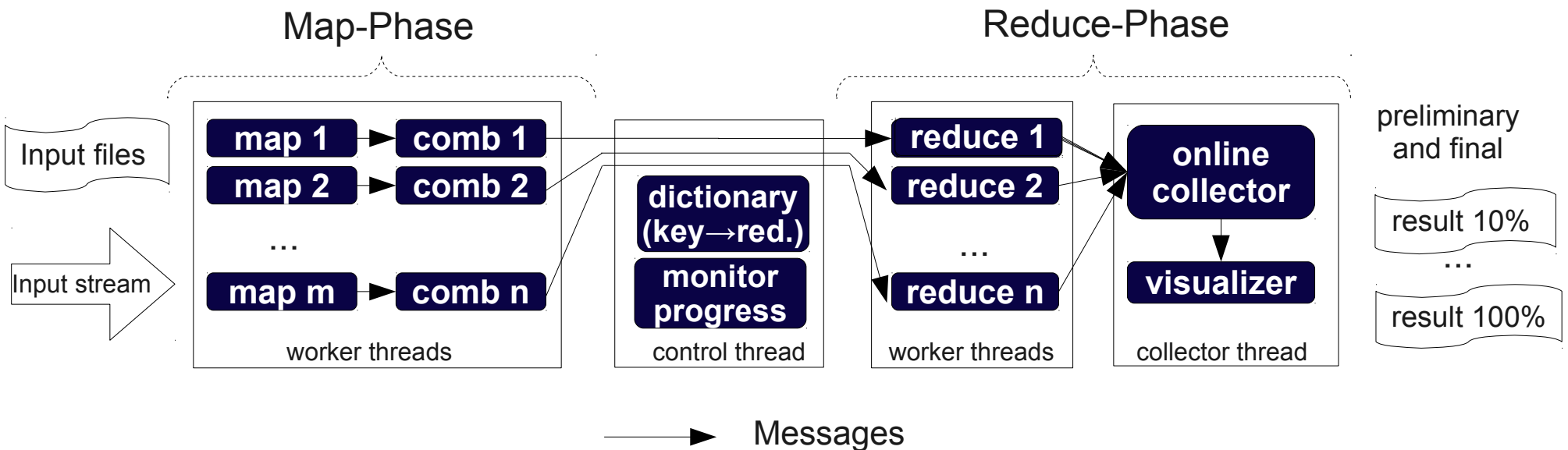


- **To implement online aggregation, results must be continuously streamed from mapper to reducers!**

Streaming MapReduce

- **Mappers send $\langle k, v \rangle$ directly to the reducer responsible for k**
 - Progress information is added to each $\langle k, v \rangle$ pair by a control component
- Reducer output their results regularly to a collector component
- **Collector computes preliminary (and final) results**
 - Preliminary results are used for convergence estimation and visualization
- Currently implemented as shared-memory framework

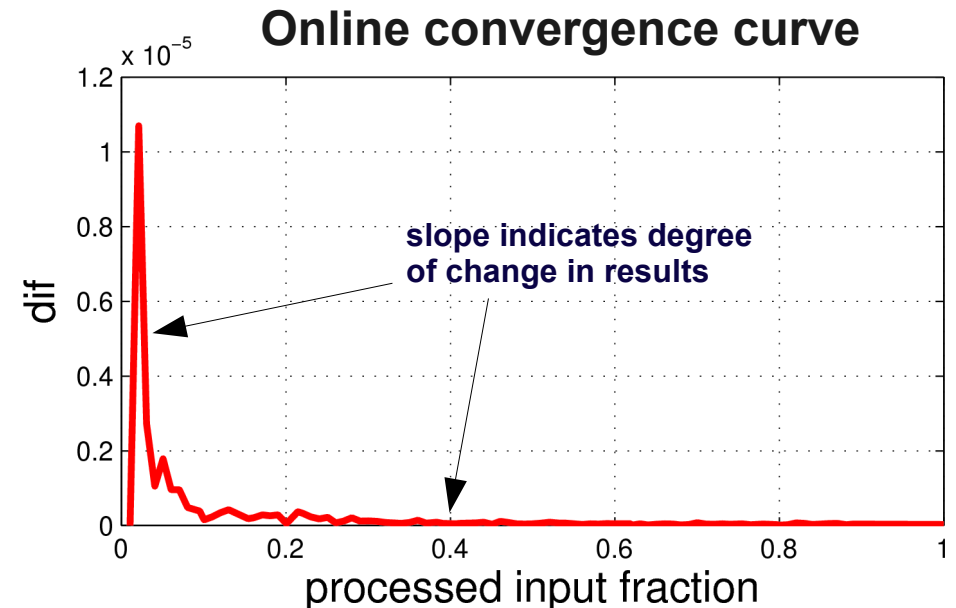
Streaming MapReduce system model



Convergence Estimation

- **Collector component saves and visualizes intermediate results using signatures of intermediate results**
 - A signature $sig(R)$ of each intermediate result R is saved to plot a “convergence graph”
 - $sig(R)$ is either the full result R , a compact representation of R , or the quality of R
- **A convergence graph depicts the “distance” between intermediate results using a specific metric $dif(sig(R_i), sig(R_j))$**

- **Example: relative word freq.**
 - R is a histogram(word->rel. freq)
 - $sig(R) = R$
 - $dif()$ is Mean Squared Error
- **Online convergence curve** plots $dif(sig(R_{i-1}), sig(R_i))$



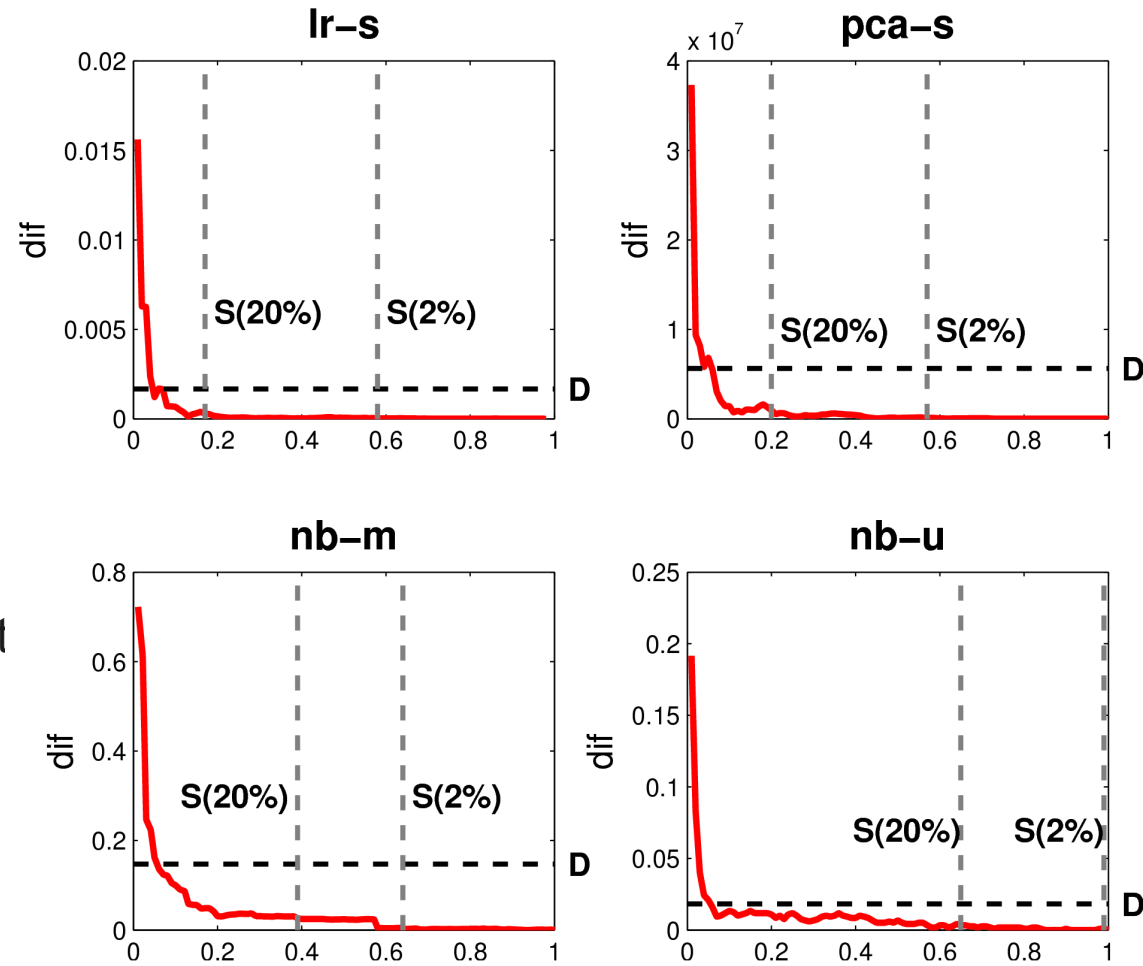
Experiments: 1-Phase Algorithms

- **1-Phase Algorithms:**

- **word count (wc)**, texts of gutenber project
- **linear regression (lr)**, synthetic dataset (s)
- **PCA (pca)**, synthetic dataset (s)
- **Naive Bayes (nb)**,
 (i) spam dataset (m),
 (ii) url dataset (u)

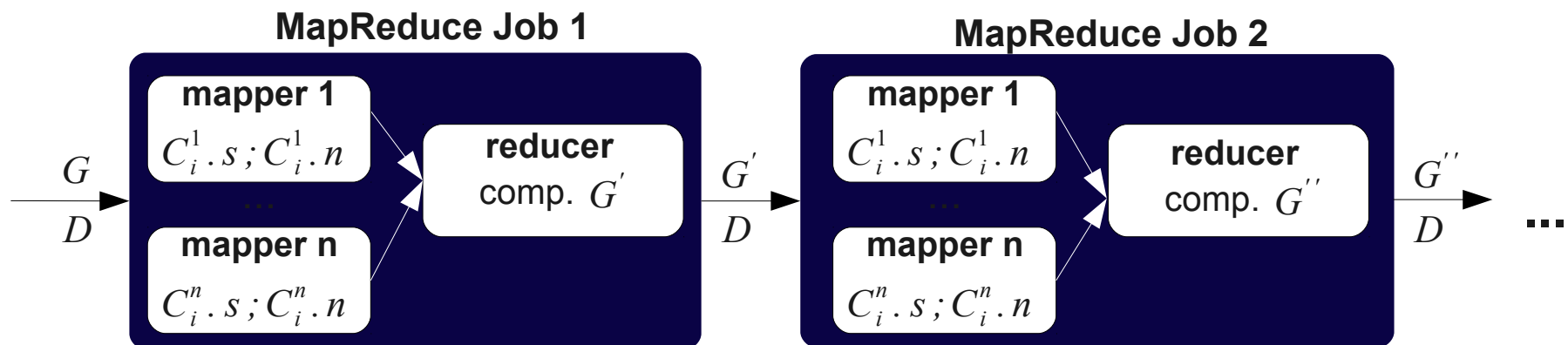
- **Offline convergence curve:**
 $dif(sig(R_i), sig(R_f))$ with R_f =final result
- D is the 95-percentile of dif values
- $S(y)$ is the smallest fraction of input data that all values of dif encountered afterwards are smaller than $y\%$ of D

Offline convergence curves



Multi-Phase Algorithms

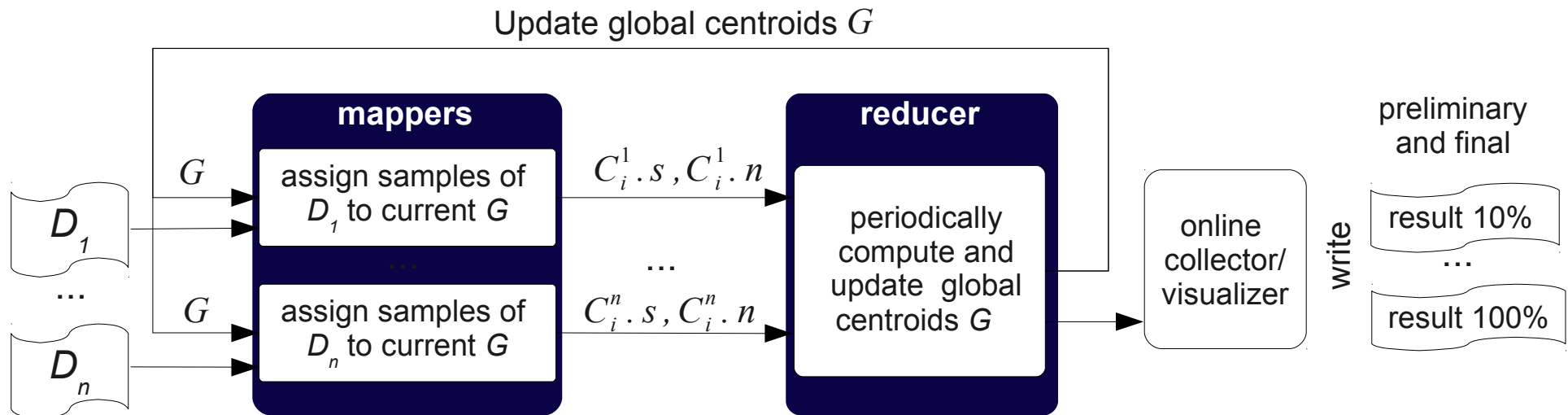
- Iterative algorithms like k-means, EM, SVM, require multiple MR phases
- Example: multi-phase k-means clustering on MR
 - Input: k cluster centroids (cluster center) G_i ($i=1..k$) and partitioned input data D
 - Map task m computes $C_i^m \cdot s$, the sum of sample vectors assigned to G_i , and $C_i^m \cdot n$
 - Reducer can compute new centroids as $G'_i = \frac{1}{\sum_{m=1}^n C_i^m \cdot n} \sum_{m=1}^n C_i^m \cdot s$



- How to derive preliminary results?

Single-Phase K-Means

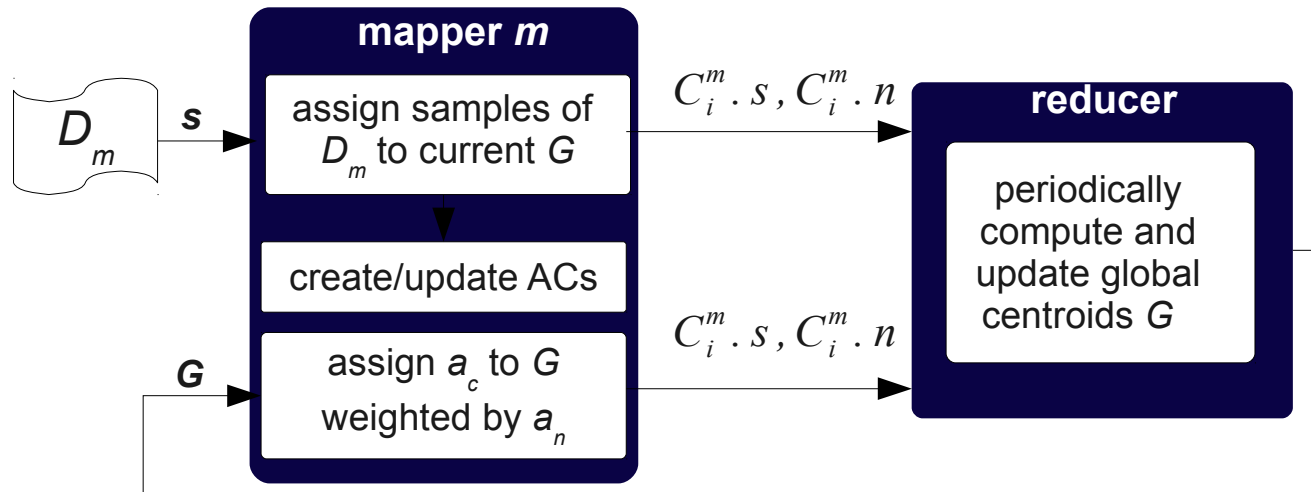
- **Idea:** re-submit preliminary results G to mappers
 - G is updated via event mechanism



- **Problem:** mapper has to revisit samples already processed when an update for global centroids G is received
 - **Not all samples may fit into memory!**

Auxiliary Clusters

- **Each sample s is stored in an auxiliary cluster (AC)**
 - ACs represent a group of similar samples by two values
 - a_c : centroid of AC and a_n : number of samples represented by AC
 - a_c are treated as data points weighted by a_n



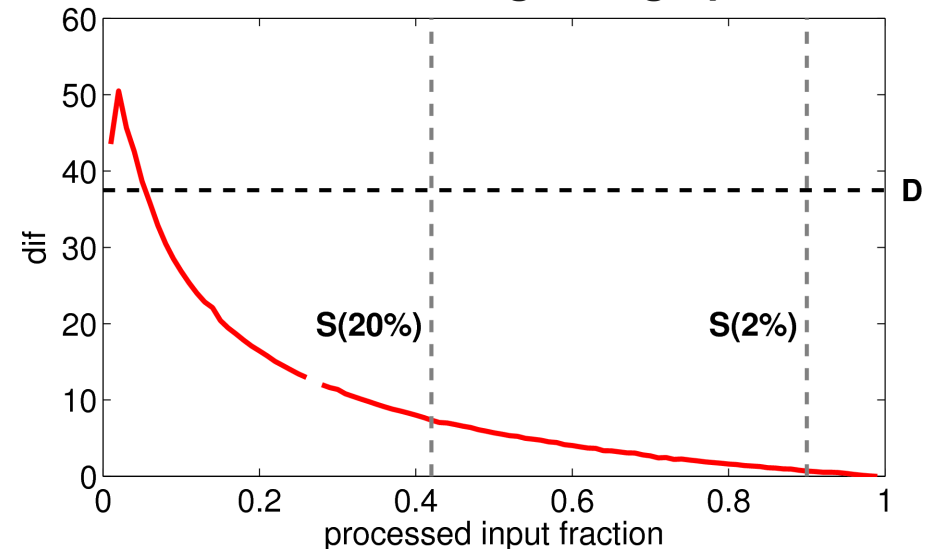
Update and creation of ACs:

- A new AC is created for each s until max number L of ACs is reached
- s is added to the closest AC if its dist. is smaller than the largest dist. between two ACs
- Otherwise the two ACs with the smallest dist. are merged and a new AC is created for s

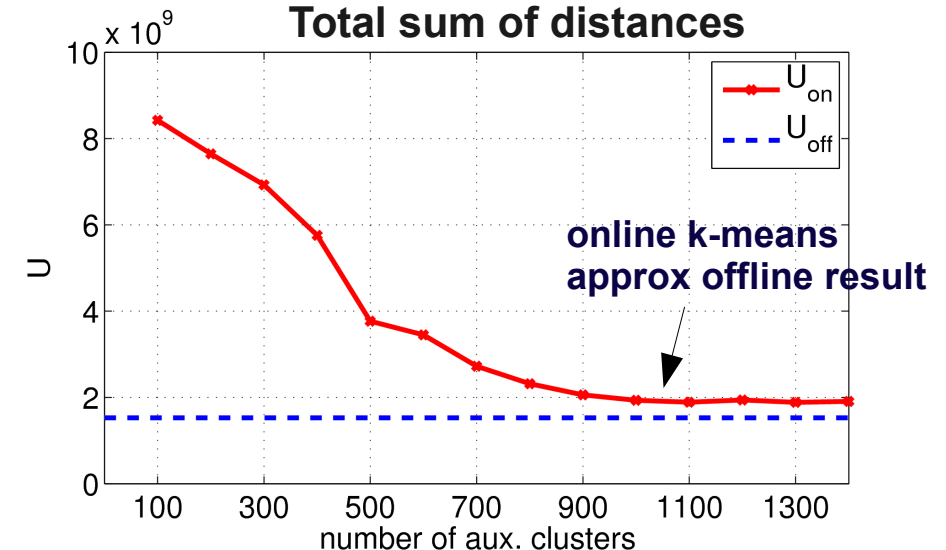
Experiments: Online K-Means

- **Dataset:** US Census (1990) with 2.4Mio samples
- **Parameter:**
 - $k = 5$;
 - number of ACs $L = 1000$
 - $dif = \text{MSE}$
- R_f of online k-means differs from standard k-means
- Accuracy is evaluated using total sum of distances U
 - Standard k-means: $U_{off} = 1.52 \cdot 10^9$
 - Online k-means: $U_{on} = 1.98 \cdot 10^9$
- Accuracy increases with L

Offline convergence graph



Total sum of distances



THANK YOU !

Summary & Conclusion

- We presented a system model and approaches to exploit the potential of online aggregation in MapReduce
- We presented an approach to handle iterative algorithms which need multiple MapReduce phases
- Experiments show that our algorithms are capable to combine benefits of online aggregation with parallelism in a streaming MapReduce framework
- **Future research includes:**
 - adapting other iterative data mining algorithms such as SVN or EM
 - porting extensions for online aggregation to Hadoop Online

Progress Estimation

- **To estimate progress we monitor:**
 - i. **The size of all input data processed by all mappers**
 - ii. **For each k how many pairs are emitted by mappers per input byte**
- **Progress is monitored per key k, i.e. how much input contributed to a <k,v> pair emitted by a reducer**
 - Allows for algorithm specific progress metrics, if progress of reducer is heterogeneous.

Example:



Experimental Results 1Ph. k-means

