

# Estimating Quantiles from the Union of Historical and Streaming Data

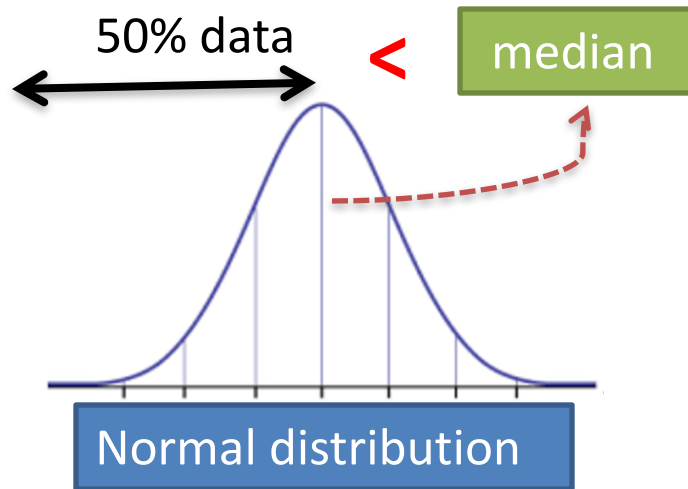
**Sneha Aman Singh**, *Iowa State University*

**Divesh Srivastava**, *AT&T Labs - Research*

**Srikanta Tirthapura**, *Iowa State University*

# Quantiles

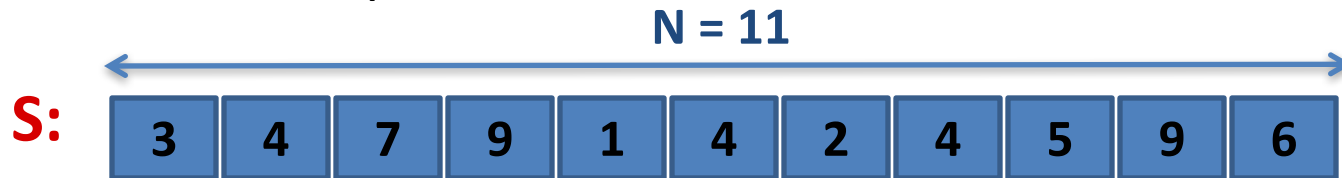
- **Quantile** – A statistical measure used to characterize distribution of data sets
- Indicates a point in a data distribution at or below which a given fraction of the entire data set falls



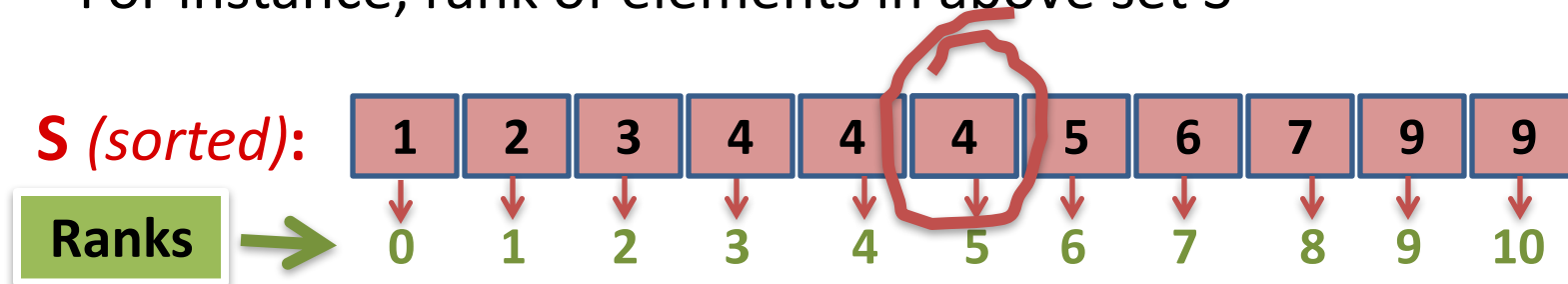
- Fundamental database query.
- An operator in many big data tools, such as **R** and **Apache Spark**

# $\phi$ – Quantile of a Set

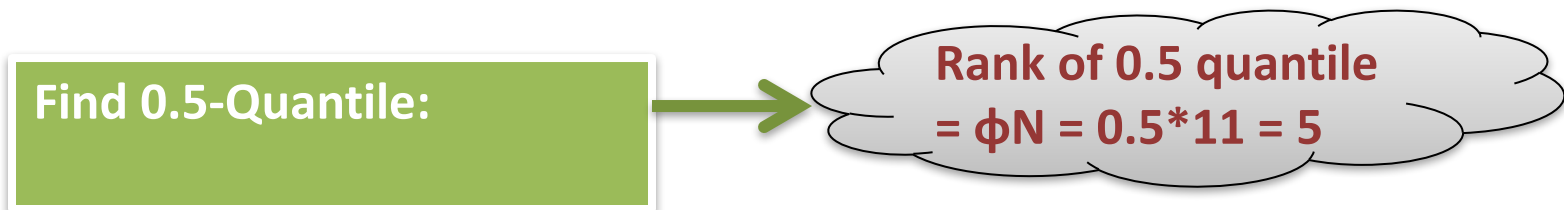
- **Rank** of an element  $x$  in a set  $S$  is the number of elements with value less than or equal to  $x$



- For instance, rank of elements in above set  $S$



- $\phi$ -quantile is an element of rank  $\phi N$ , for  $0 < \phi < 1$

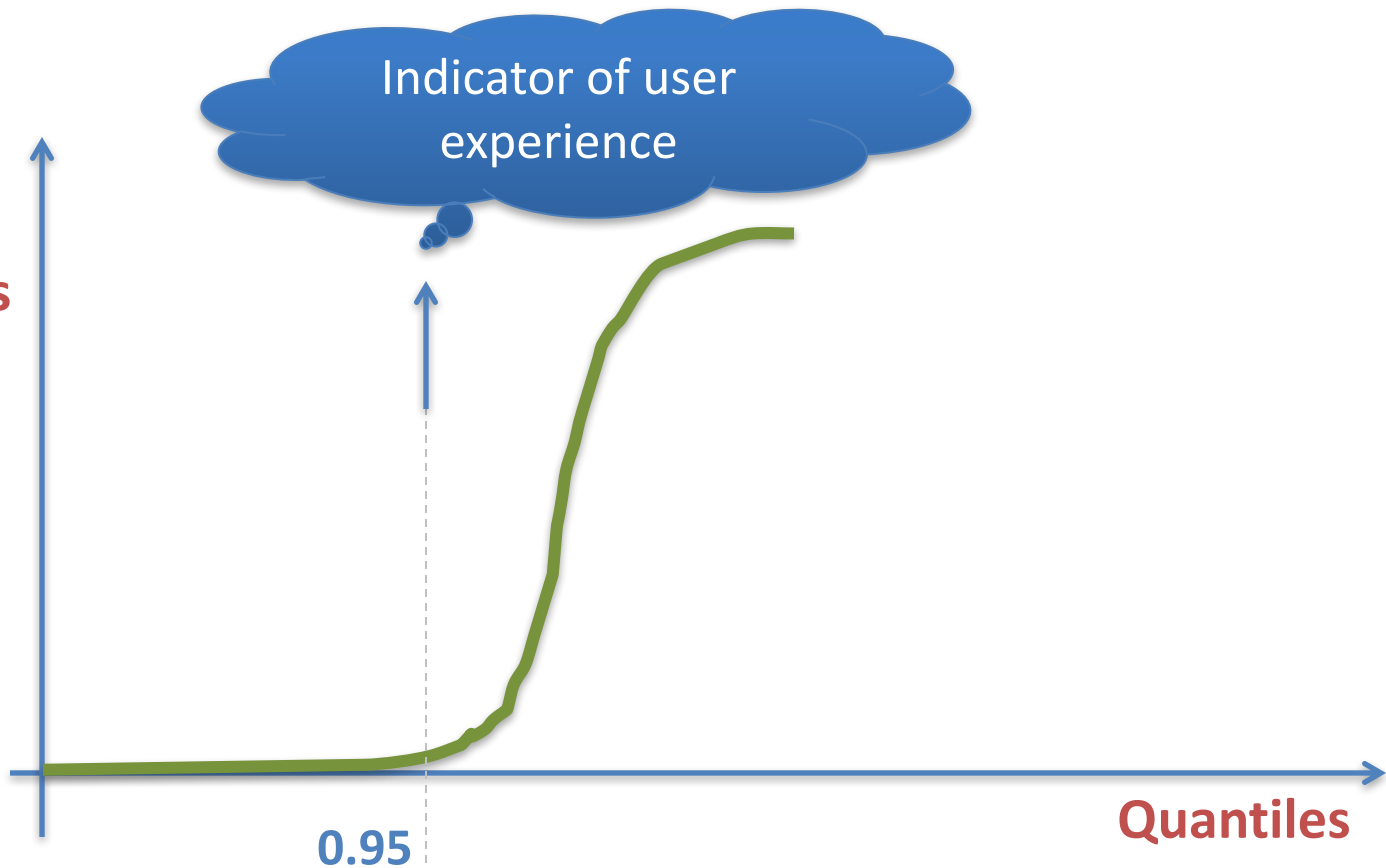


- **0-quantile** is the minimum and **1-quantile** is the maximum of  $S$ .

# Application – Web Latency

**Web Latency** – delay experienced by a user between sending a request to the web server and getting back a response.

Latency  
Experienced  
by All the Users



**0.95 quantile > Threshold latency -- Track poor network performance**

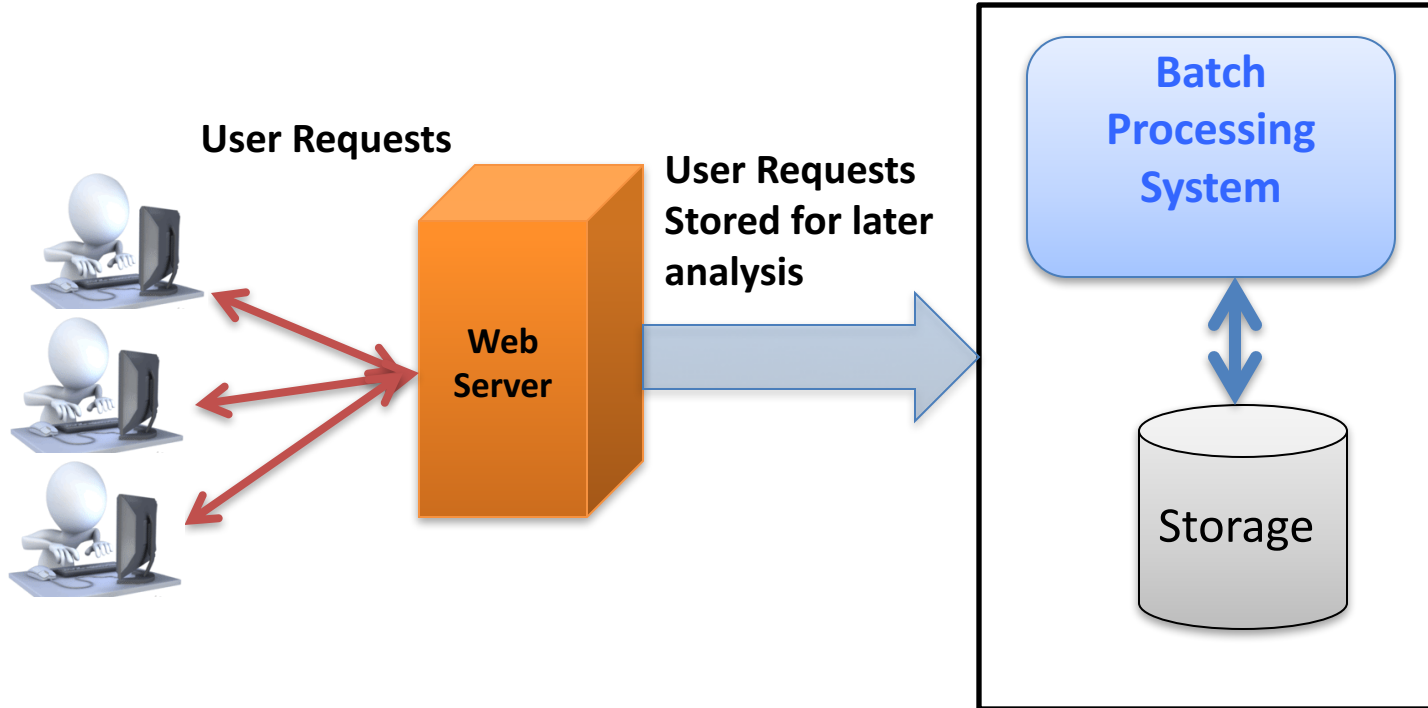
# Outline

1. Description of quantile
- 2. Historical and Streaming Data**
3. Contribution
4. High Level Algorithm Description
5. Experimental Evaluation

# Historical Data

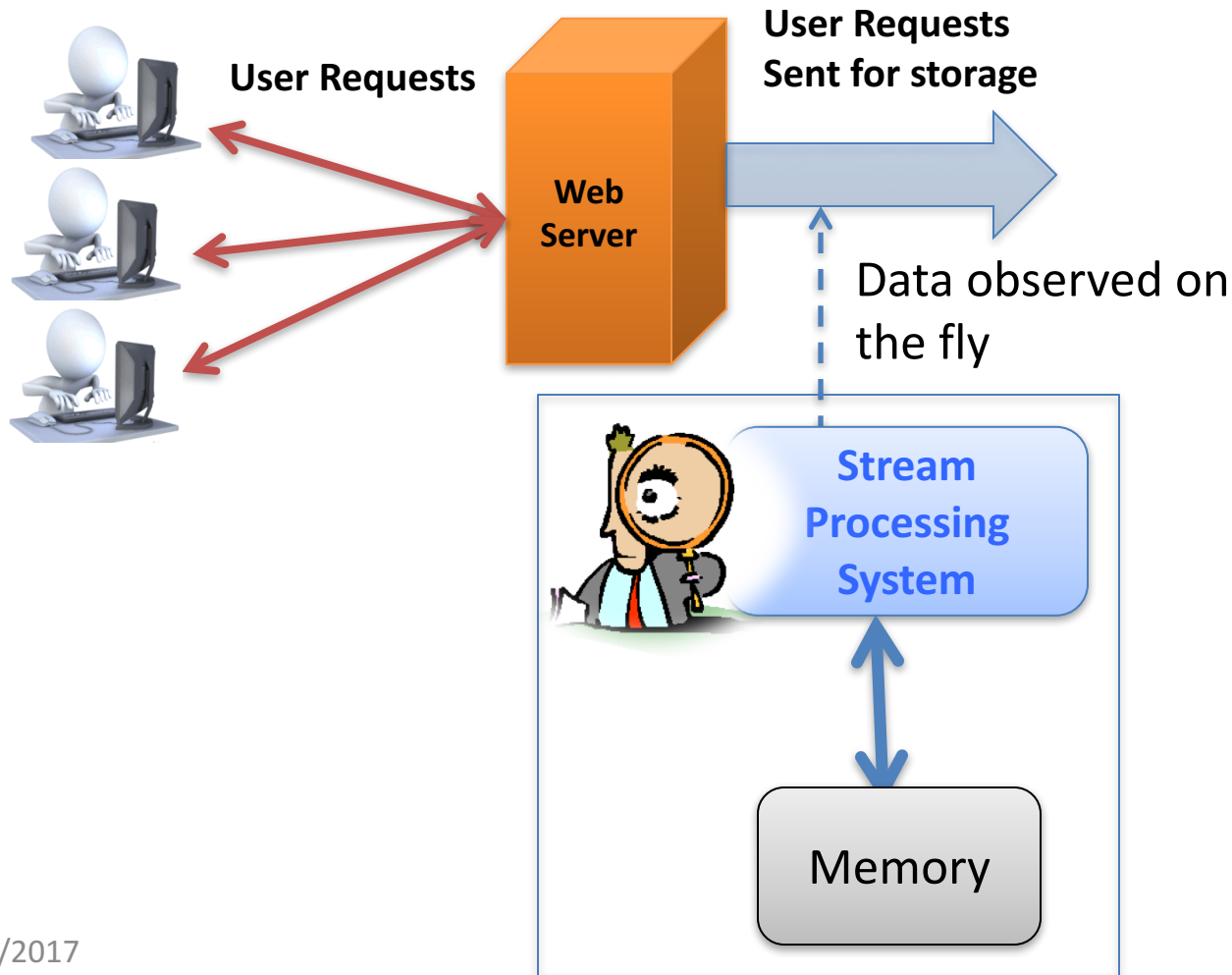
- **Historical Data** – Data collected and stored for later analysis.

Web user requests collected at the server end: to analyze user trends / behavior and web site performance

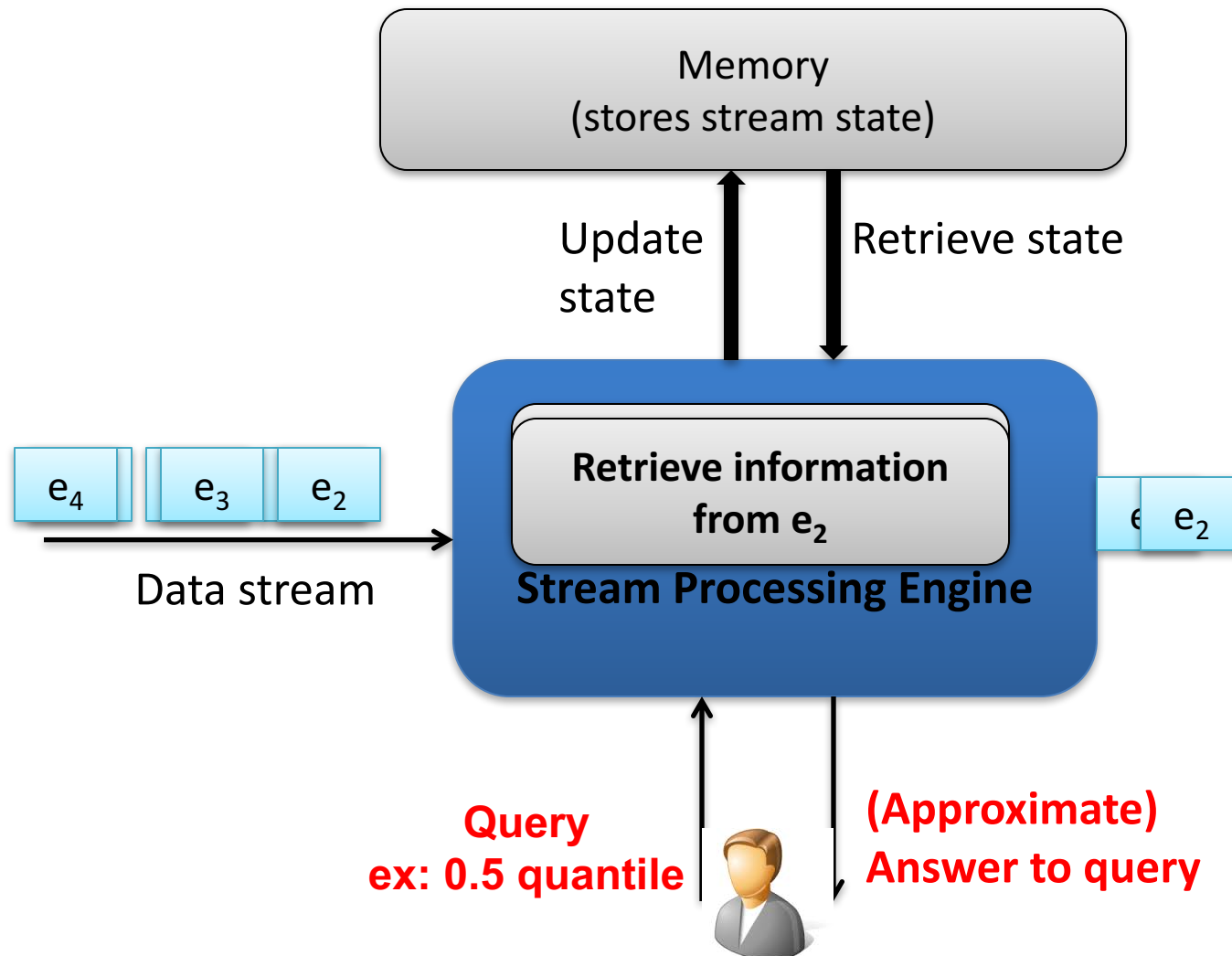


# Streaming Data

**Data Stream:** Continuous sequence of data observed in the form of events, messages, tuples, in a single pass or limited number of pass.



# Stream Processing System

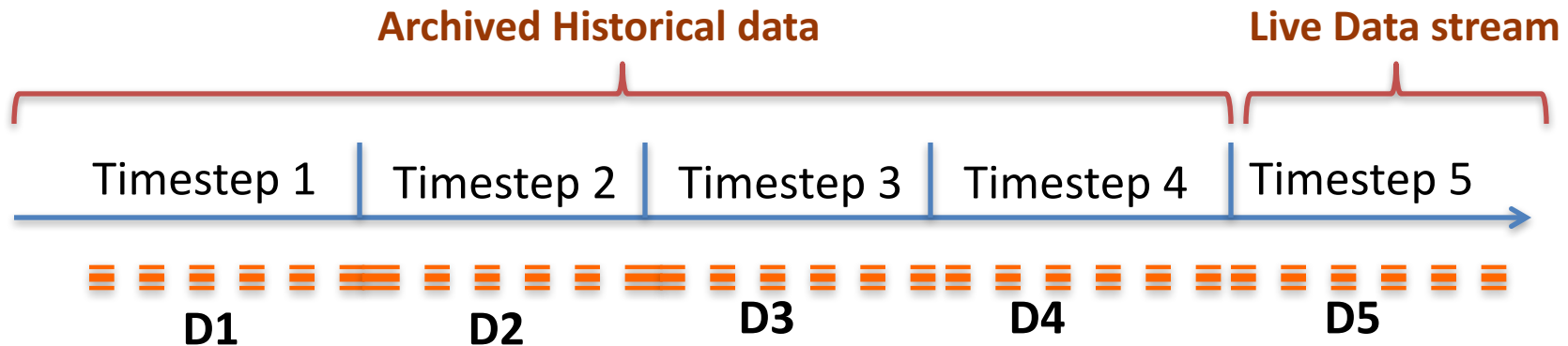


Continuous update of the state of the data, with every incoming element







# Integrated Processing of Historical and Streaming Data

- Long term data analysis with higher accuracy in real time



- Correlation of historical data with live streaming data
- Predictive analysis

# Historical vs Stream Processing System

| Batch Processing System   | Stream Processing System   |
|---|--|
| Data stored and can be traversed multiple times   | Single or limited pass over data   |
| Accurate results of user query<br> | Approximate results of user query<br> |
| Latency due to disk I/O<br>       | Fast processing time<br>             |

**Can we have our cake and eat it too?**

# Our Focus: (Approximate) Quantile Computation

- $N$  : size of data stream
- **$\epsilon$ -approximate quantiles** – Find an element of rank  $\rho$  from a dataset of size  $N$ , such that,

$$(\phi - \epsilon)N \leq \rho \leq (\phi + \epsilon)N, \quad 0 < \epsilon \ll \phi < 1$$

- $\epsilon$  : approximation parameter
- rank of element  $e$  in  $D$  :  $|\{x \in D \mid x < e\}|$

# Prior Work on Approximate Quantile Computation

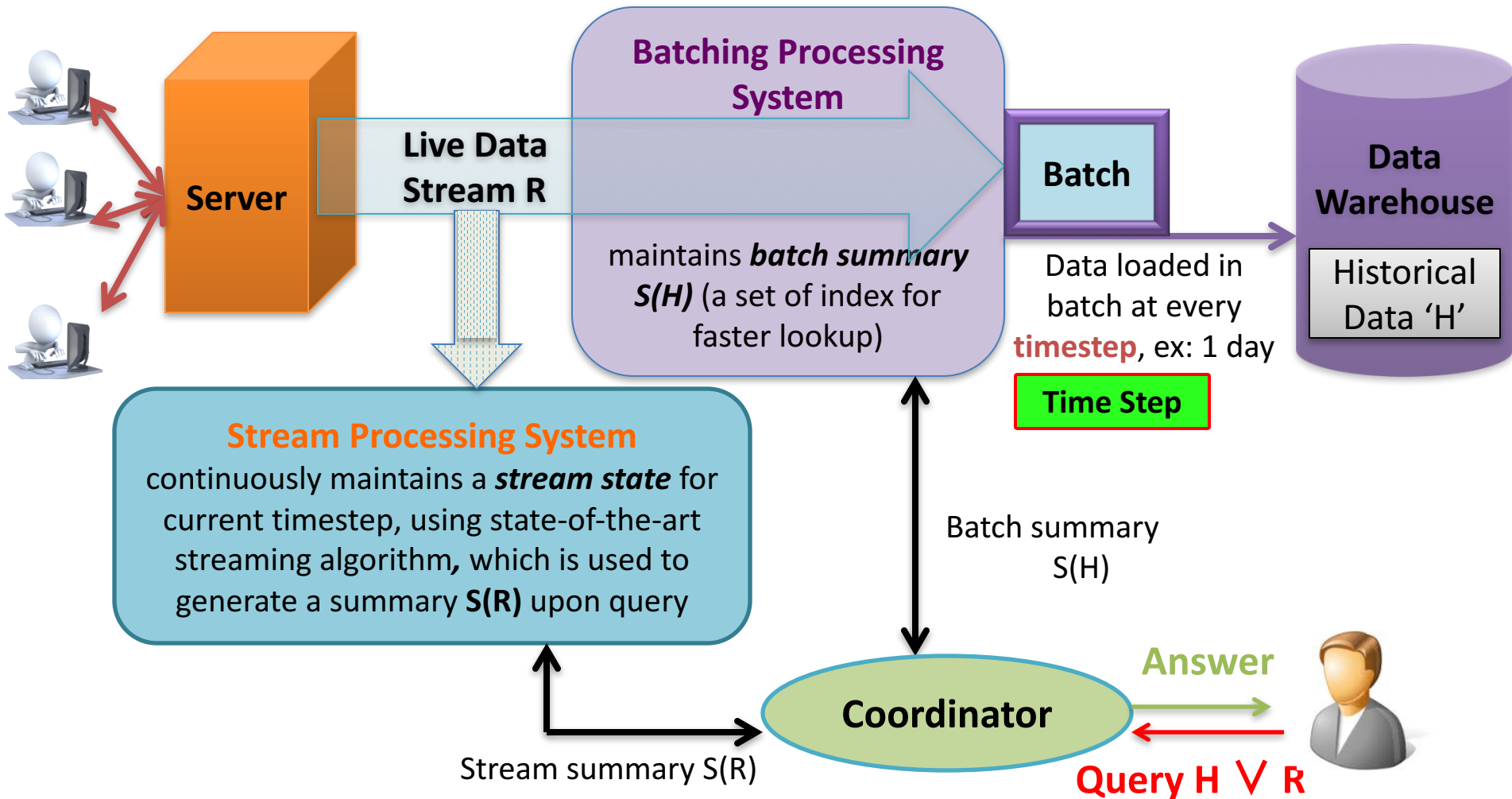
Space efficient  **$\epsilon$ -approximate** streaming algorithms:

- Manku et al [MRL98] with space cost  $O(1/\epsilon \log^2(\epsilon N))$
- Greenwald-Khanna [GK01] with space cost  $O(1/\epsilon \log(\epsilon N))$
- Qdigest [SBAS04] with space cost  $O(1/\epsilon \log(U))$ 
  - $N$ : size of data stream,
  - $U$ : universe size,
  - $\epsilon$ : approximation parameter,  $0 < \epsilon < 1$

# Outline

1. Description of quantile
2. Historical and Streaming Data
- 3. Contribution**
4. High Level Algorithm Description
5. Experimental Evaluation

# Processing Historical Data + Streaming Data



# Contribution

- An algorithm that returns an element of rank  $\rho$  as a  $\phi$ -quantile from a union of historical and streaming data, such that

$$\phi N - \epsilon m \leq \rho \leq \phi N + \epsilon m,$$

$$m \ll N$$

compared to  
 $\phi N - \epsilon N \leq \rho \leq \phi N + \epsilon N$  by  
streaming algorithms

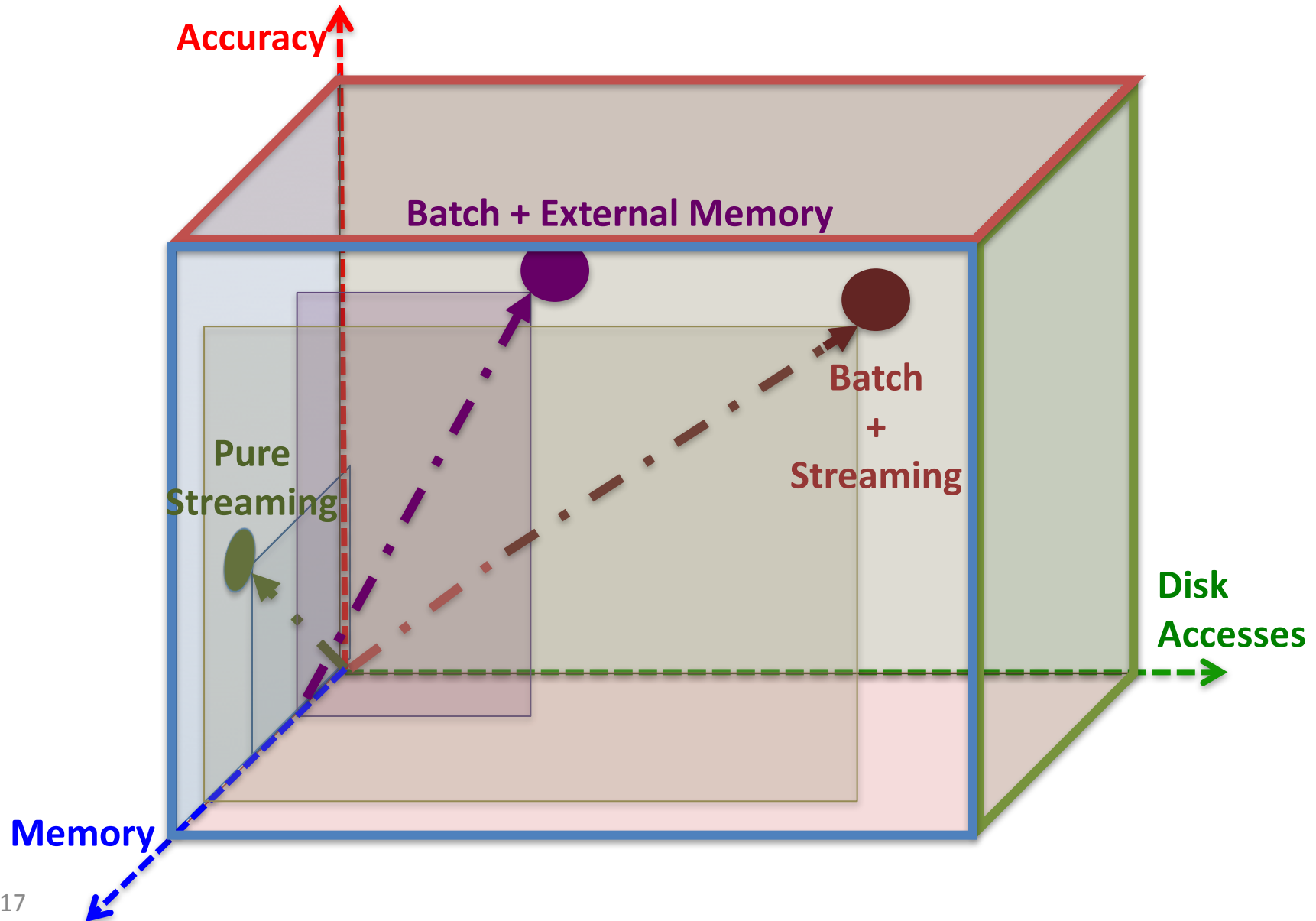
- The algorithm is memory efficient requiring  $O\left(\frac{\log(\epsilon N)}{\epsilon}\right)$  memory
  - $m$  : size of data stream,
  - rank of an element  $e$  in  $S$  :  $|\{x \in S \mid x < e\}|$
  - $\epsilon$  : approximation parameter,  $0 < \epsilon < 1$
  - $T$  : no of time steps

# Contribution

- Small processing time for updating historical data in the warehouse at every timestep, using amortized  $O((m / B)\log(n / B))$  disk accesses per timestep,
- Real time querying using amortized  $O(\log^2(\epsilon m / B)\log(T))$  disk accesses
  - $n$  : size of historical data
  - $B$  : block size of the data warehouse
  - $T$  : no of time steps
  - $\epsilon$  : approximation parameter,  $0 < \epsilon < 1$



# 3-way Tradeoff between Accuracy, Disk Accesses & Memory



# Comparison of Our Algorithm with Pure Streaming Algorithms

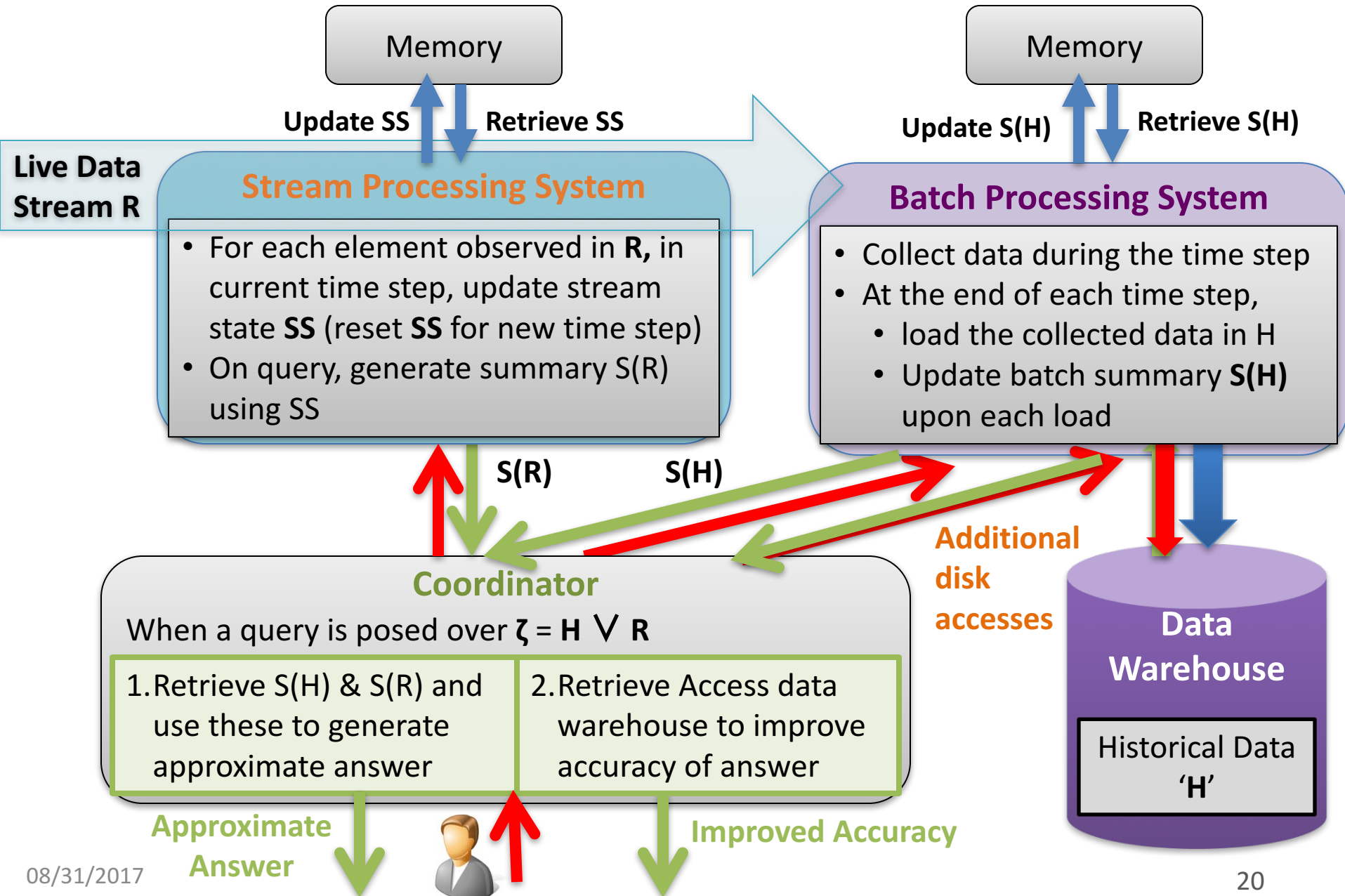
|                | GK (streaming)                                    | Qdigest (streaming)                      | Our Algorithm                                     |
|----------------|---|--|---|
| Relative Error | $\epsilon$  | $\epsilon$                               | $\frac{\epsilon m}{N}$                            |
| Memory         | $O\left(\frac{\log(\epsilon N)}{\epsilon}\right)$ | $O\left(\frac{\log(U)}{\epsilon}\right)$ | $O\left(\frac{\log(\epsilon N)}{\epsilon}\right)$ |

- N: total size of historical and streaming data
- m: data stream size,  $m \ll N$
- T: no of time steps
- U: universe size

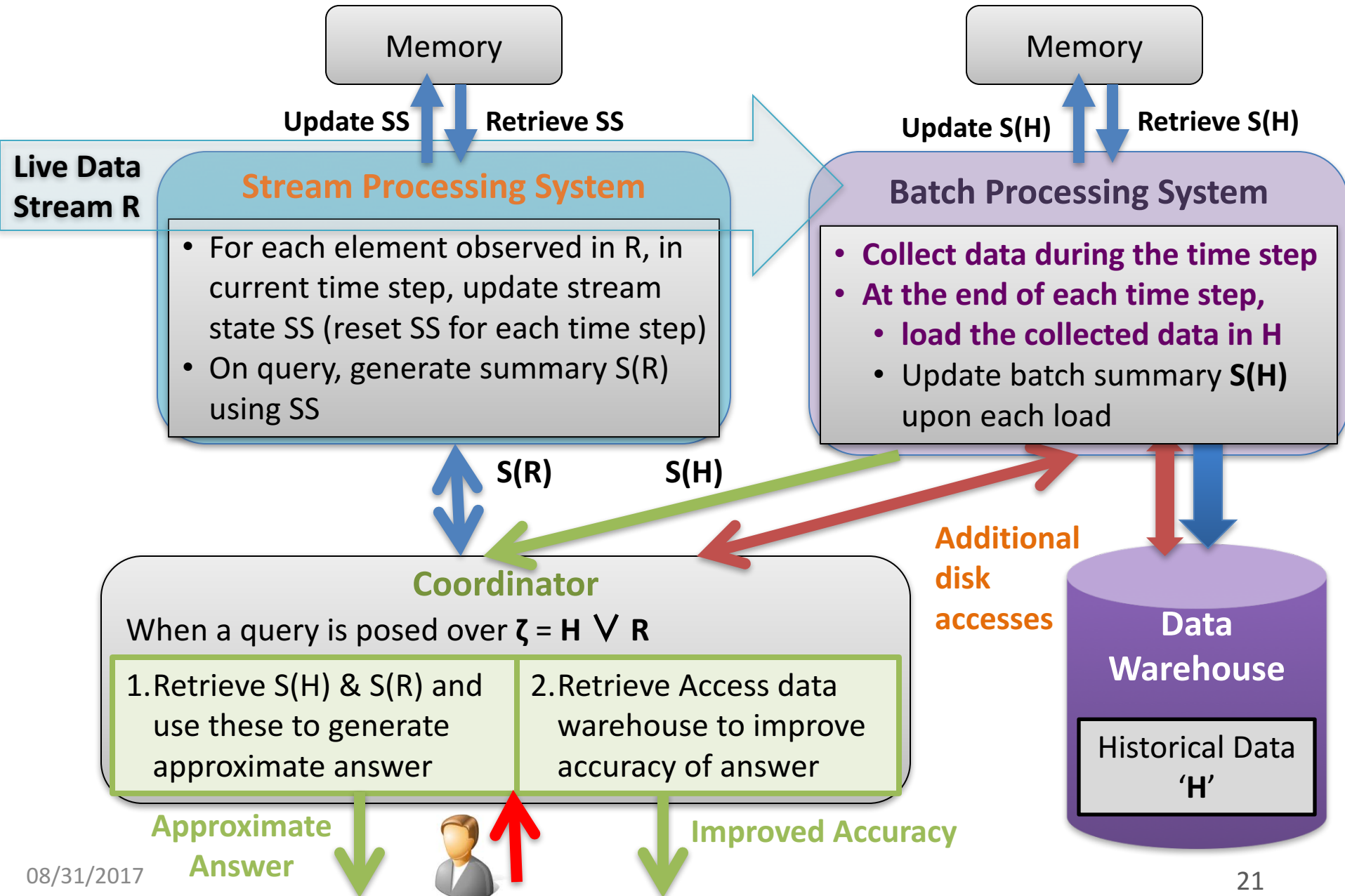
# Outline

1. Description of quantile
2. Historical and Streaming Data
3. Contribution
- 4. High Level Algorithm Description**
5. Experimental Evaluation

# High Level Algorithm Steps

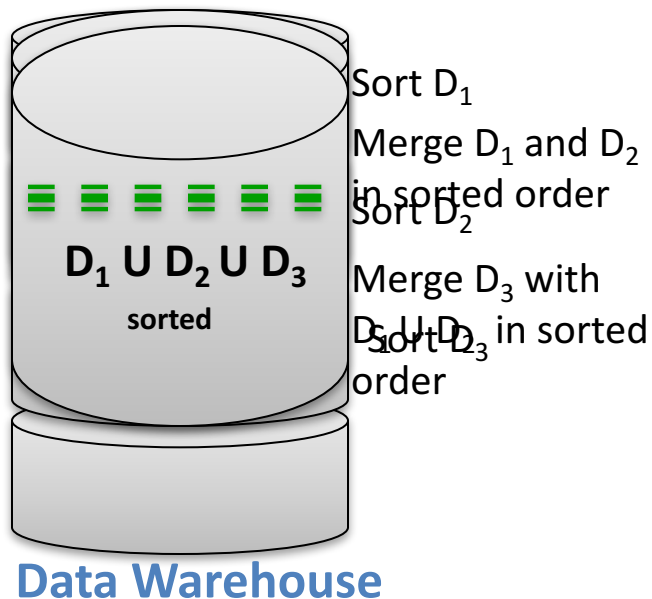
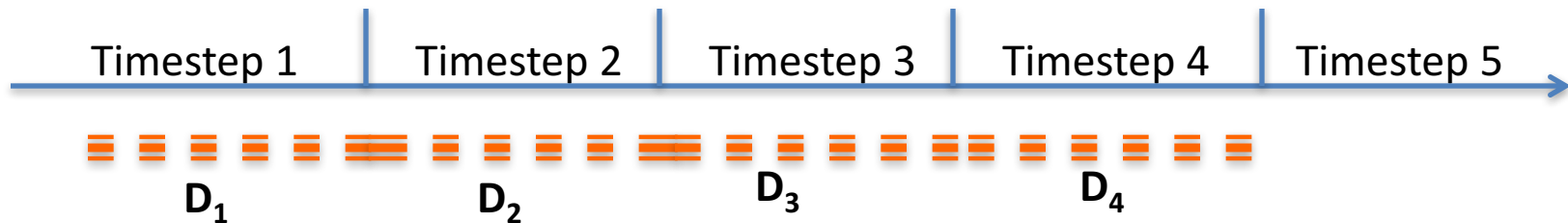


# High Level Algorithm Steps



# Historical Data Update at Each Time Step: Approach 1

Merge the newly arrived dataset with older historical data at every time step

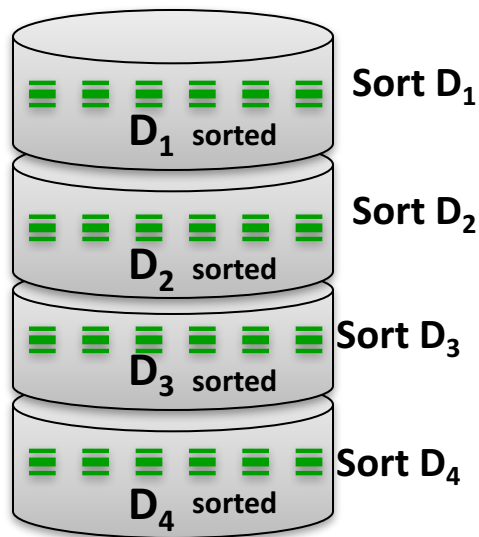
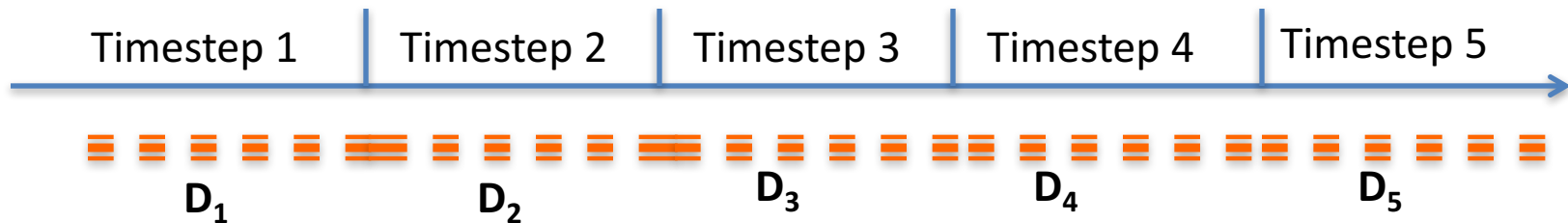


## Disadvantages :

- Large disk access at each timestep  $i$  :
  - Sort  $D_i$  of size  $n_i$  :  $O((n_i/B)\log(n_i/B))$
  - Merge with historical data of size  $n$  :  $O(n/B)$ ,  $B$ - block size

# Historical Data Update at Each Time Step: Approach 2

Add the new arrived dataset at each time step as a new partition in data warehouse



**Data Warehouse**

## Advantage :

- Small update cost at each timestep  $i$ ,
  - Sort and store  $D_i$  of size  $n_i$  :  $O((n_i/B)\log(n_i/B))$

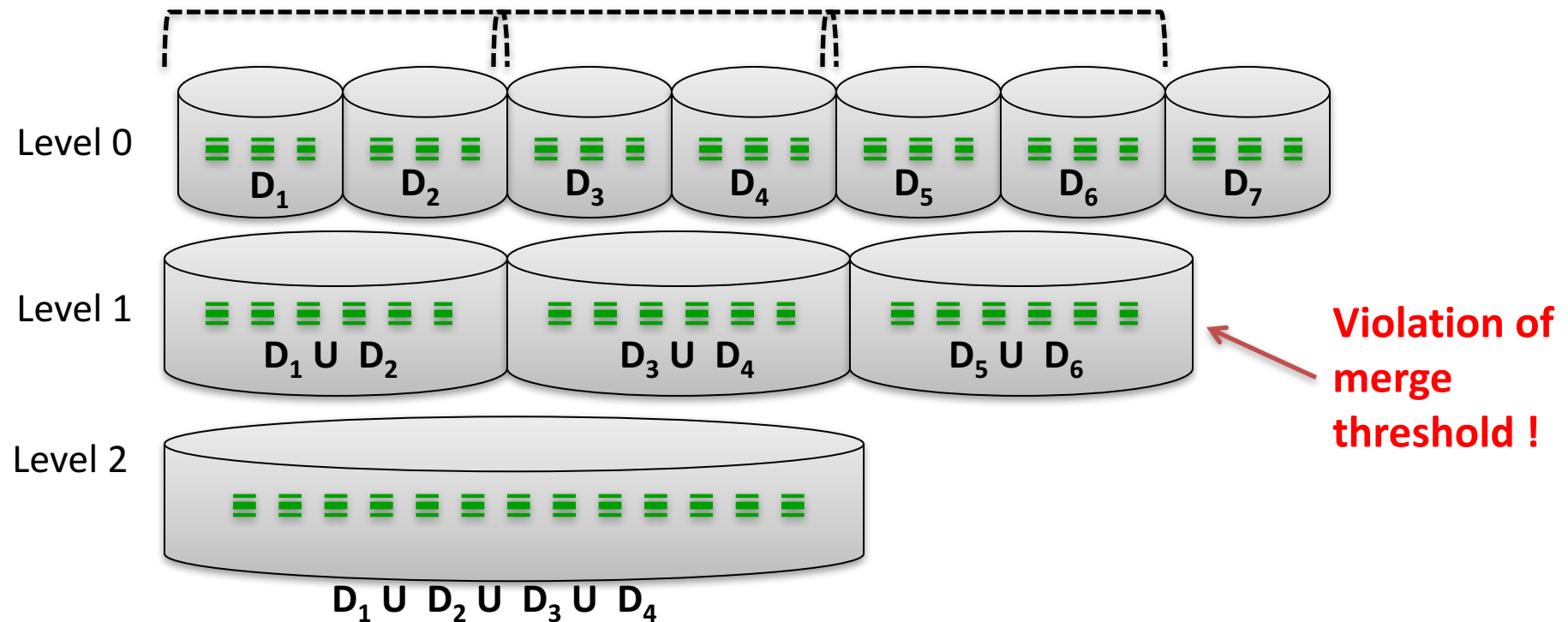
## Disadvantage :

- Maintenance of several partitions
- Expensive to answer query -  $O(d*T)$ 
  - $d$ : no of disk access / timestep to answer a query
  - $T$ : no of timesteps

# Historical Data Update: Our Approach

- Dataset  $D_i$  arrives at timestep  $i$
- Data stored in **sorted** format in data partitions using **levels**

Merge Threshold  $\kappa$  Merge Threshold  $\kappa$  Merge Threshold  $\kappa = 2$

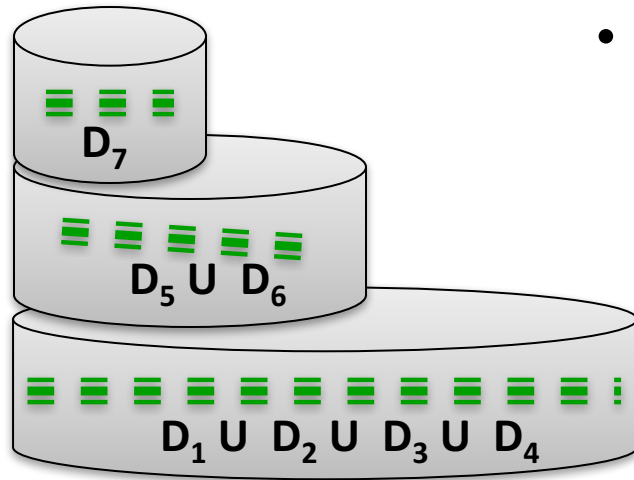


## Data Warehouse Partitions

- Each level has at most  $\kappa$  data partitions. In the figure,  $\kappa=2$ ,  $\kappa$  – **merge threshold**
- At most  $\log_{\kappa} T$  data partitions at the end of  $T$  time steps.

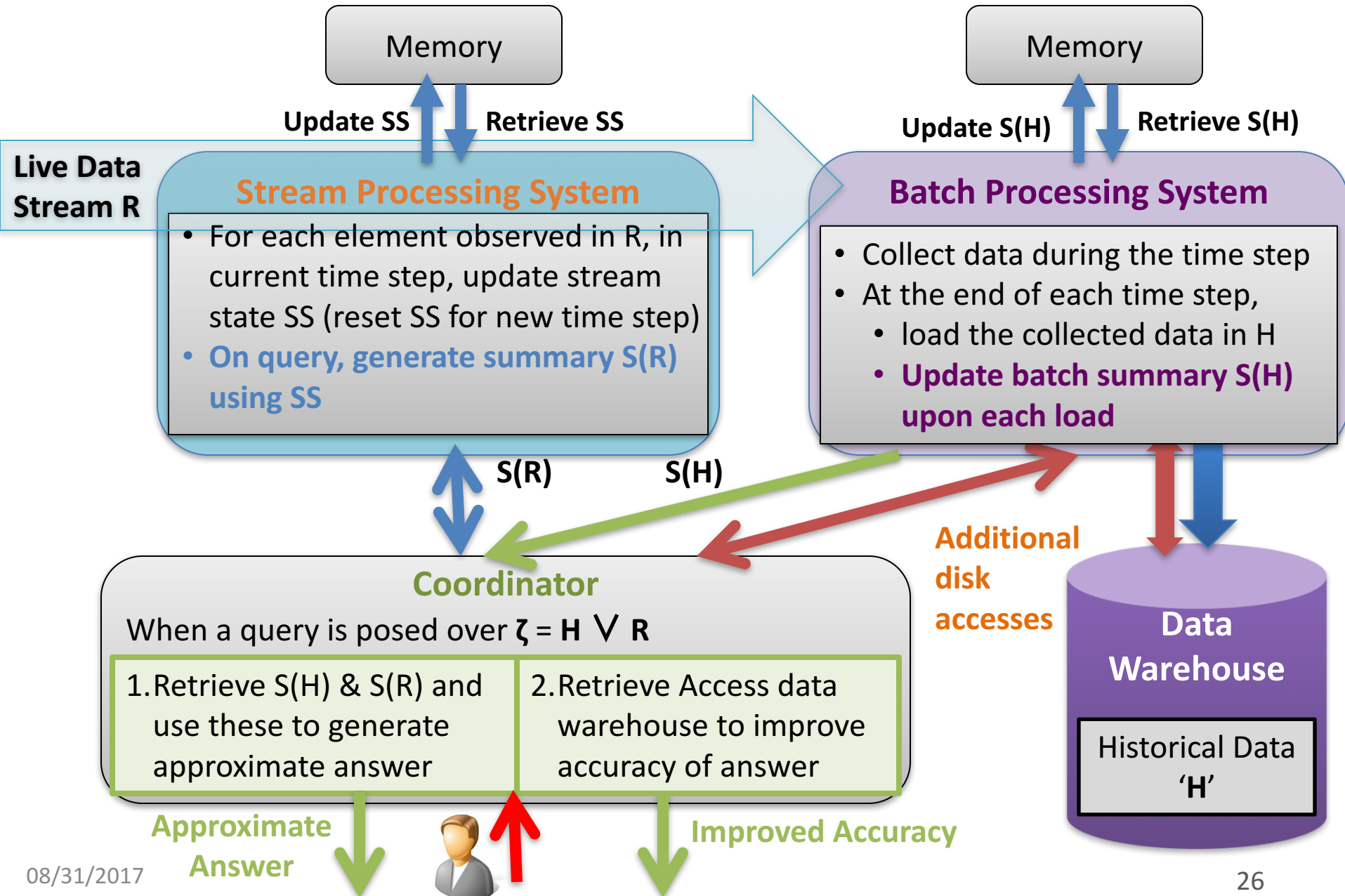


# Historical Data Update at Each Time Step: Our Approach



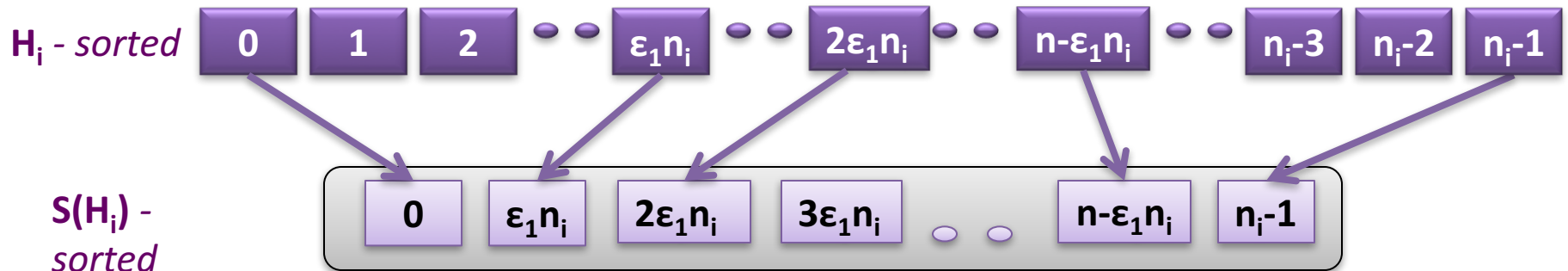
- Advantages:
  - Relatively small update cost since the entire historical data is not accessed at each time step
  - Small query time

# High Level Algorithm Steps

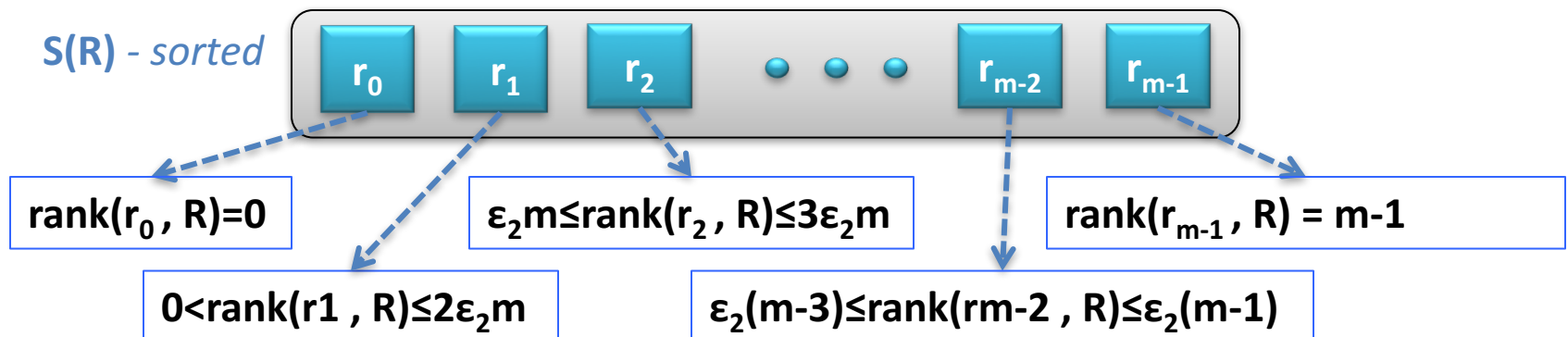


# Batch and Stream Summary

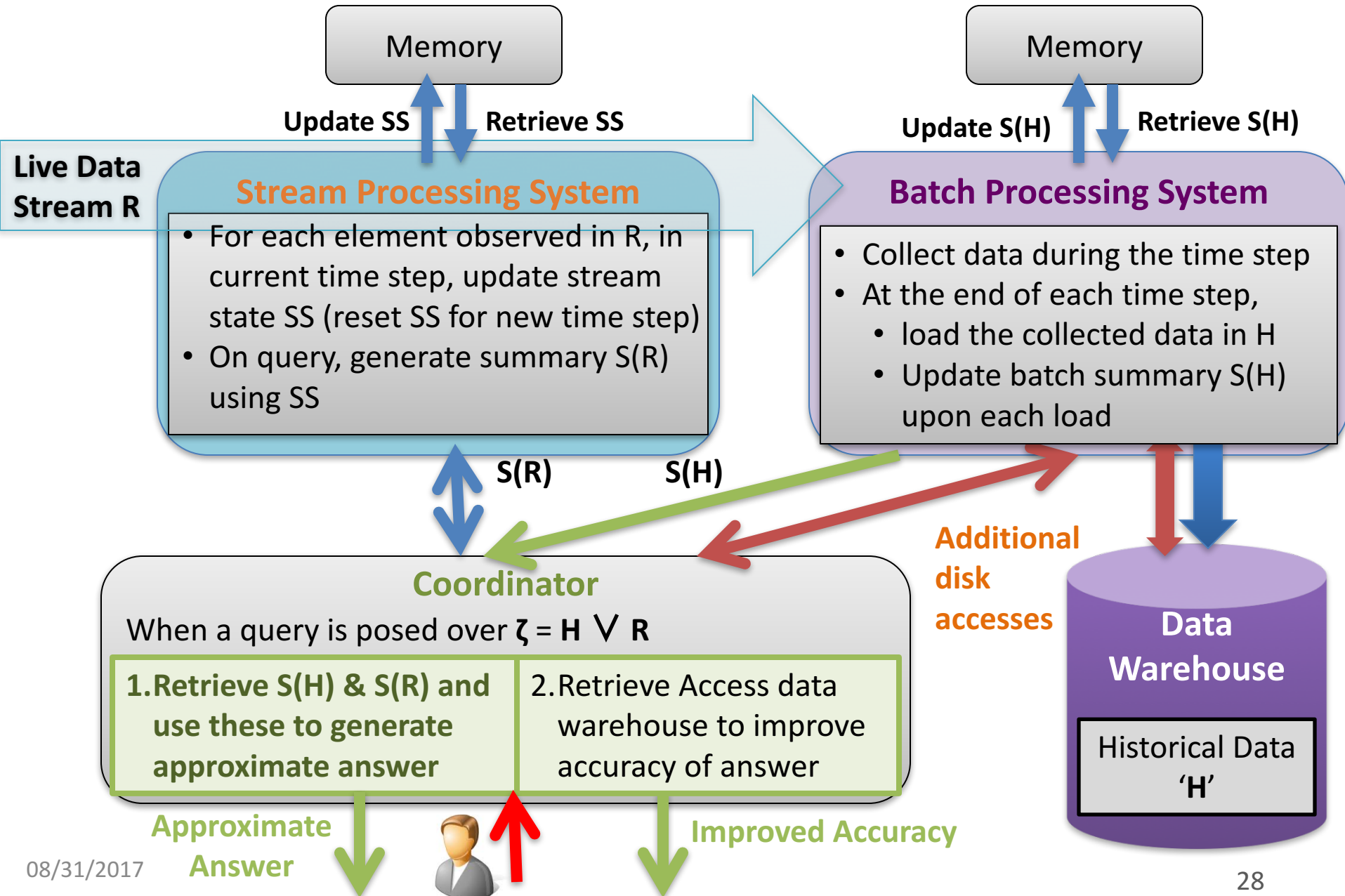
- Batch Summary  $S(H_i)$**  –  $((1/\epsilon_1)+1)$  elements and their database index, for each disk partition  $H_i$  of size  $n_i$ , updated at the end of each time step



- Stream Summary  $S(R)$**  – use stream state  $SS$  to get  $((1/\epsilon_2)+1)$  elements of approximate ranks  $\epsilon_2 m$ ,  $2\epsilon_2 m$ ,  $3\epsilon_2 m$  and so on, with a max error of  $\epsilon_2 m$ , created when a query is made



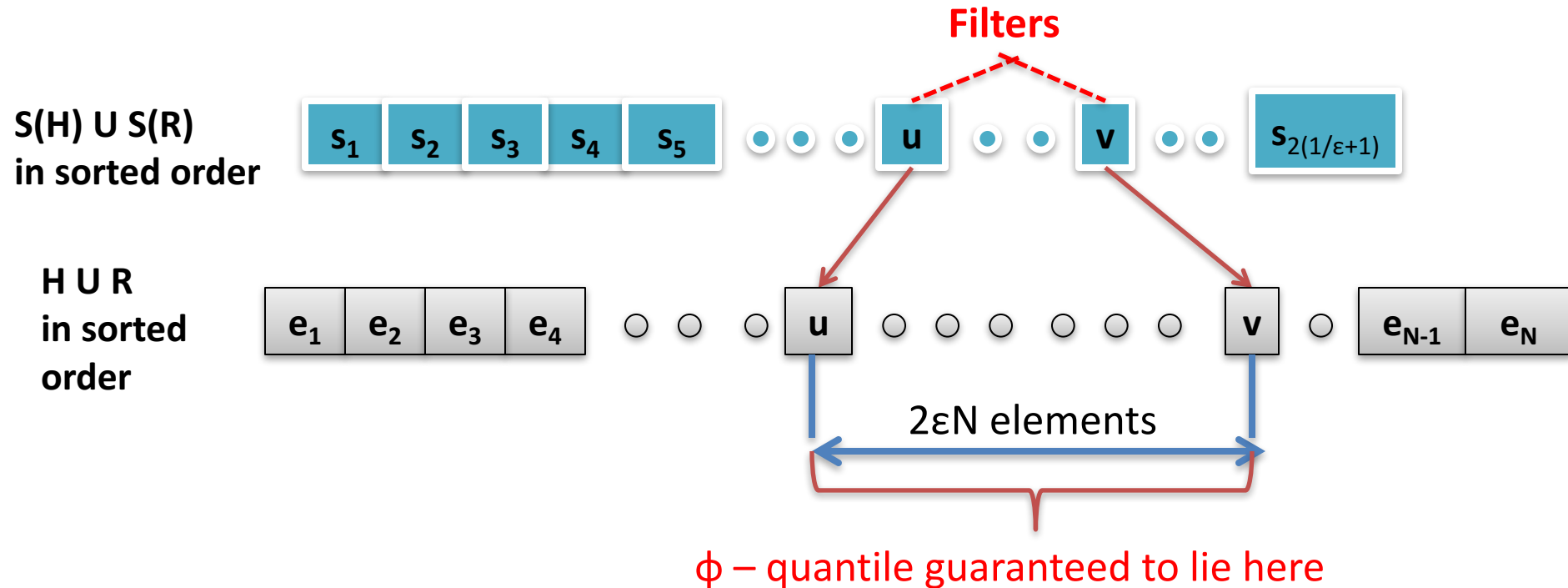
# High Level Algorithm Steps



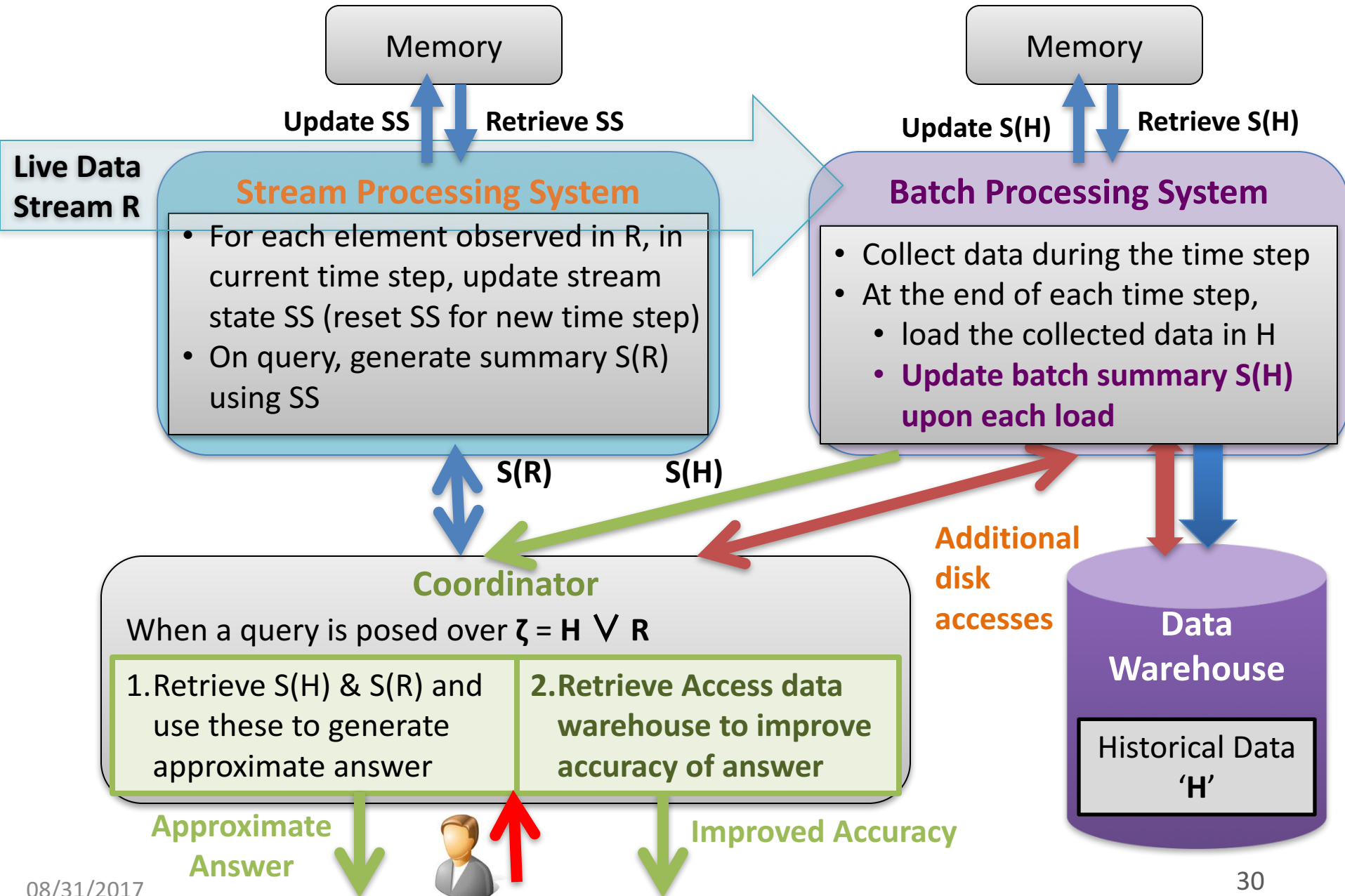
# Answer a Query : Find a Range within which Quantile is Guaranteed to lie

$H$  : historical data of size  $n$ ;  $R$  : live data stream of size  $m$ ;  $N=(n+m)$ ;

Find elements  $u$  and  $v$  from  $S(H) \cup S(R)$ , such that the quantile is guaranteed to lie between  $[u, v]$  in the union of historical and streaming data



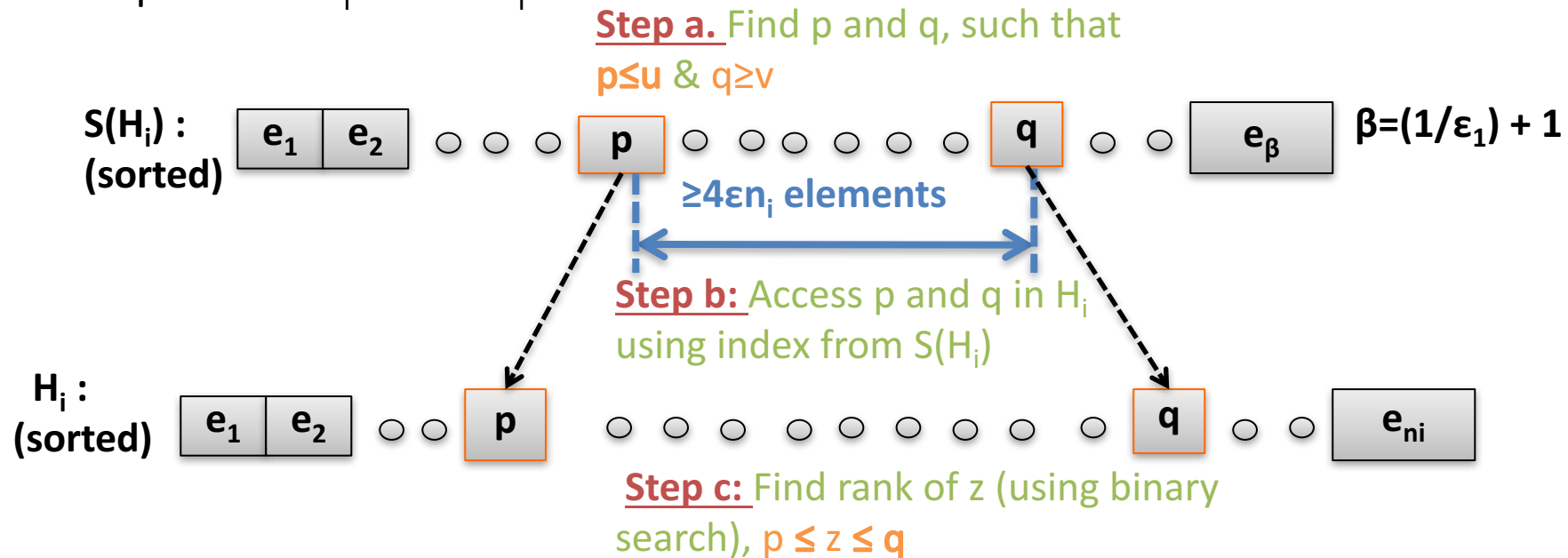
# High Level Algorithm Steps



# Additional Disk Accesses to Improve Accuracy

The filter range  $[u, v]$  is narrowed down recursively to zero down on error due to batch.

- At each recursion, Compute rank of  $z = (u+v)/2$  in all data partitions and stream
- For each partition  $H_i$  of size  $n_i$



- Rank of  $z = \sum(\text{rank of } z \text{ in each partition}) + \text{rank of } z \text{ in data stream}$
- Choose  $[u, z]$  or  $[z, v]$  as the new filter for next recursion, depending on which contains the quantile

# Outline

1. Description of quantile
2. Historical and Streaming Data
3. Problem Statement and Motivation
4. Contribution
5. High Level Algorithm Description
- 6. Experimental Evaluation**



# Experimental Setup

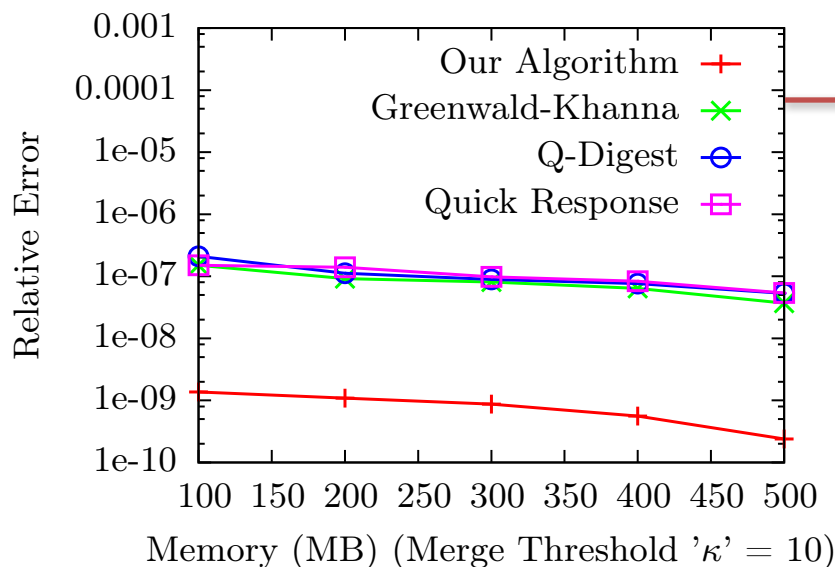
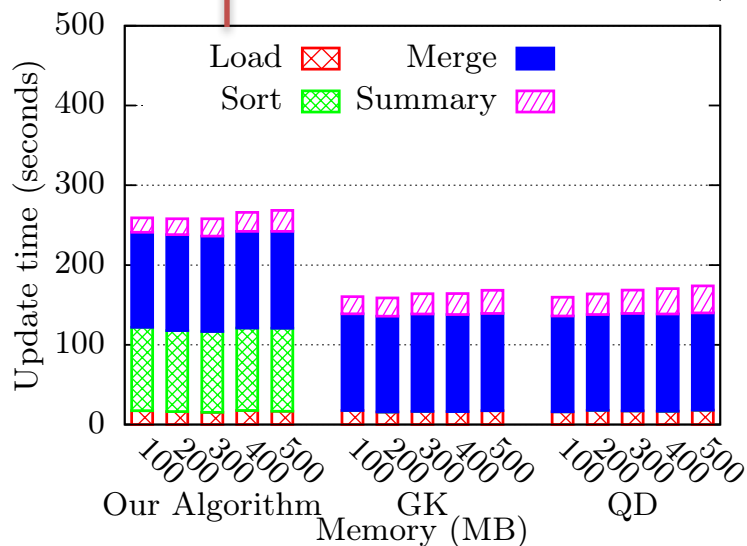
| Datasets                     | Wikipedia | Network traces | Uniform Random | Normal |
|------------------------------|-----------|----------------|----------------|--------|
| Data Size (GB)               | 58.5      | 60             | 100            | 100    |
| Tuples / time step (million) | 50        | 60             | 100            | 100    |

| Algorithms | Greenwald-Khanna | Qdigest | Our |
|------------|------------------|---------|-----|
|------------|------------------|---------|-----|

| Performance Metric | Accuracy  | Processing time                  | Query time                       |
|--------------------|---|----------------------------------|----------------------------------|
| Description        | Measure relative error :<br>$ \phi N - r  / \phi N$ | Runtime and no. of disk accesses | Runtime and no. of disk accesses |

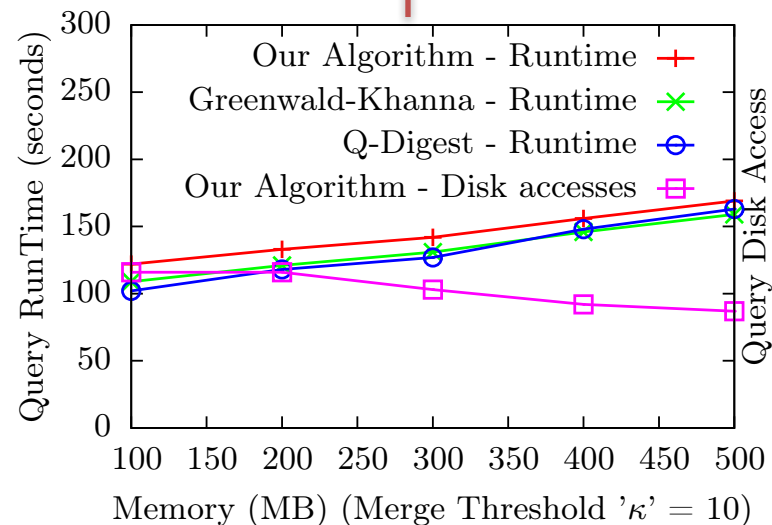
# Dependence On Memory

Update time  
vs  
Memory



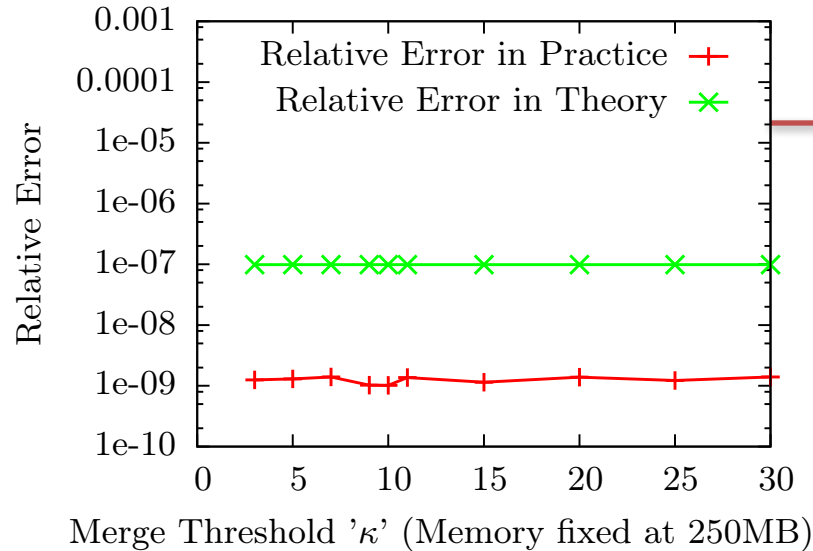
Accuracy vs Memory

Query time vs Memory



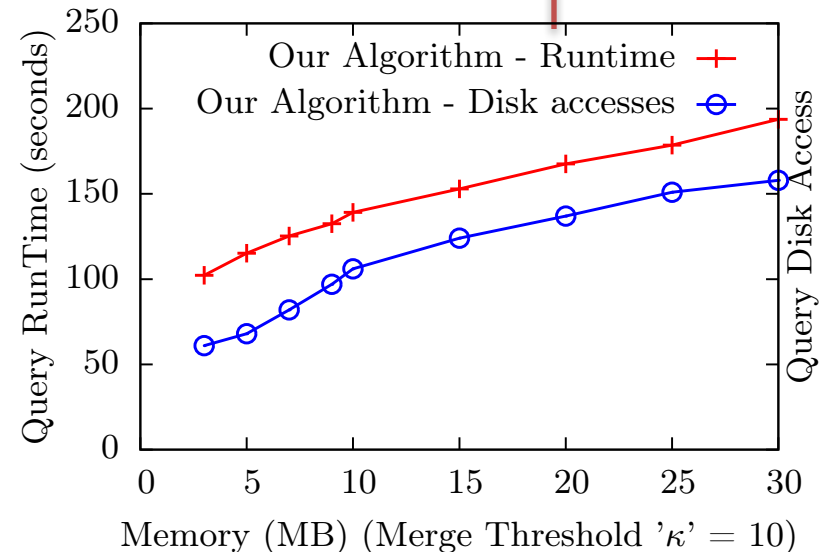
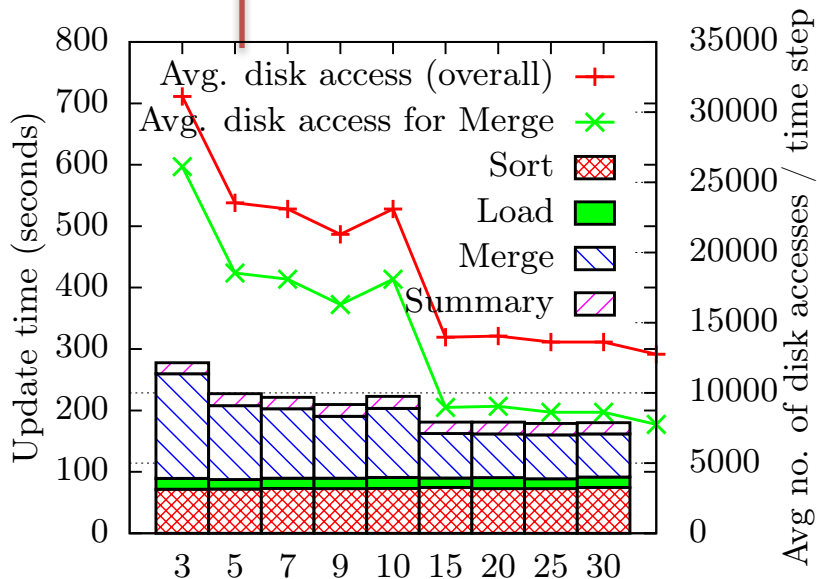
# Dependence On Merge Threshold

Update time  
vs  
Merge Threshold



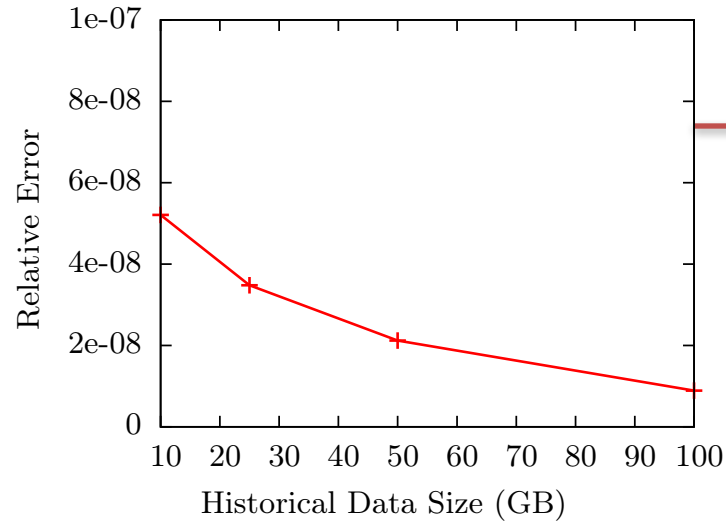
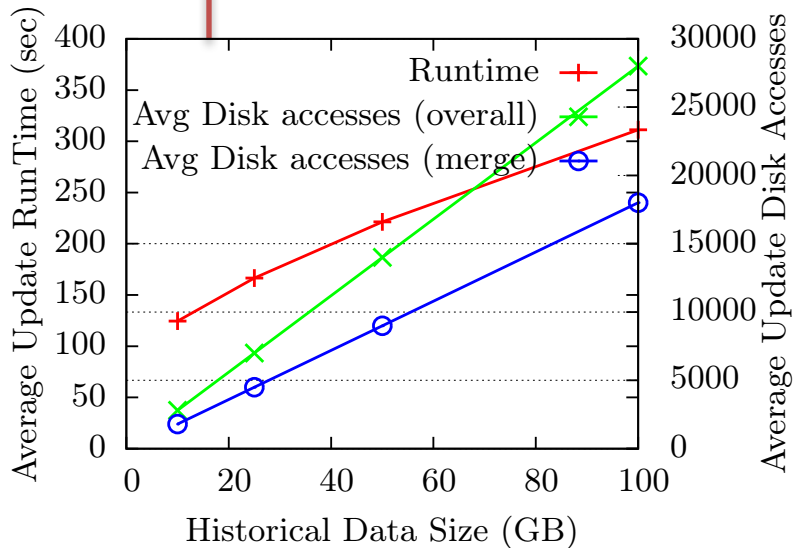
Accuracy  
vs  
Merge Threshold

Query time  
vs  
Merge Threshold



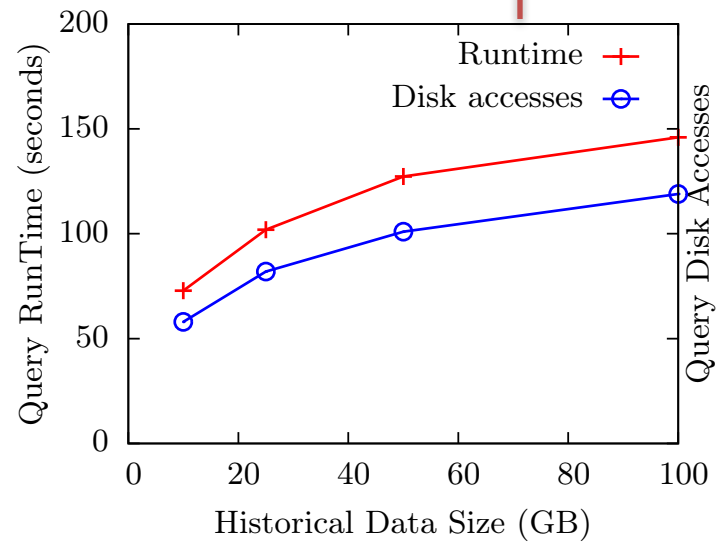
# Dependence On Historical Data Size

**Update time  
vs  
Historical Data Size**



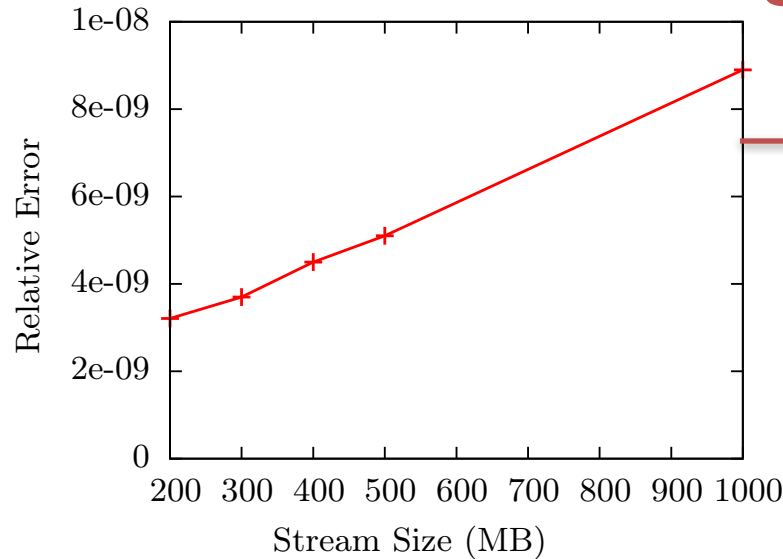
**Accuracy  
vs  
Historical Data Size**

**Query time  
vs  
Historical Data Size**



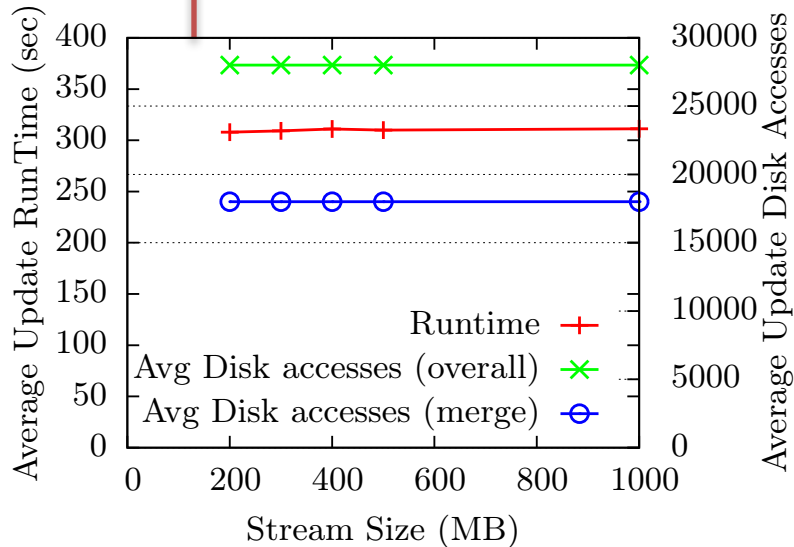
**Normal Dataset (streamind data size: 1GB, memory: 250 MB, merge threshold: 10)**

# Dependence On Streaming Data Size

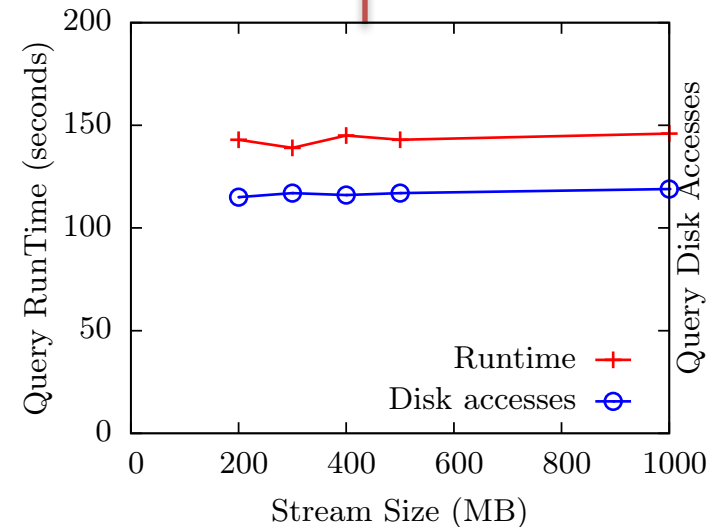


Accuracy  
vs  
Data Stream Size

Update time  
vs  
Data Stream Size



Query time  
vs  
Data Stream Size



Normal Data (historical data size: 100GB, memory: 250 MB, merge threshold: 10)

# Conclusion

- We present an algorithm that identifies quantile from a union of historical and streaming data in real time
  - with accuracy significantly better than pure streaming algorithms
  - using memory similar to pure streaming algorithms
  - using few additional disk accesses

## Future Work

- Analysis on the tradeoff between accuracy and disk accesses
- Improve the window adaptation of algorithm
- Parallel methods for processing historical data
- Other aggregates over a union of historical and streaming data

# Thank You