Lecture 11

Real-time Data Warehousing

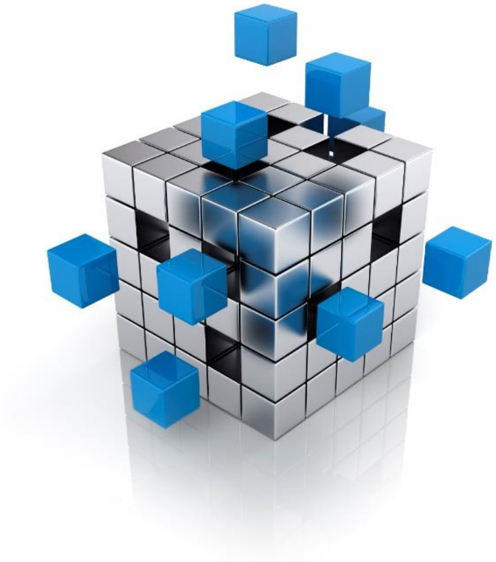
Summary – last week

- How to build a DW
  - The DW Project: usual tasks, hardware, software, timeline (phases)
  - Data Extract/Transform/Load (ETL):
    - Data storage structures, extraction strategies (e.g., scraping, sniffing)
    - Transformation: data quality, integration
    - Loading: issues, and strategies, (bulk loading for fact data is a must)
  - Metadata:
    - Describes the contents of a DW, comprises all the intermediate products of ETL,
    - Helps for understanding how to use the DW

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

This week

- Real-time Data Warehousing
  - Real-time Data Processing Challenges
  - Real-time Join Algorithms



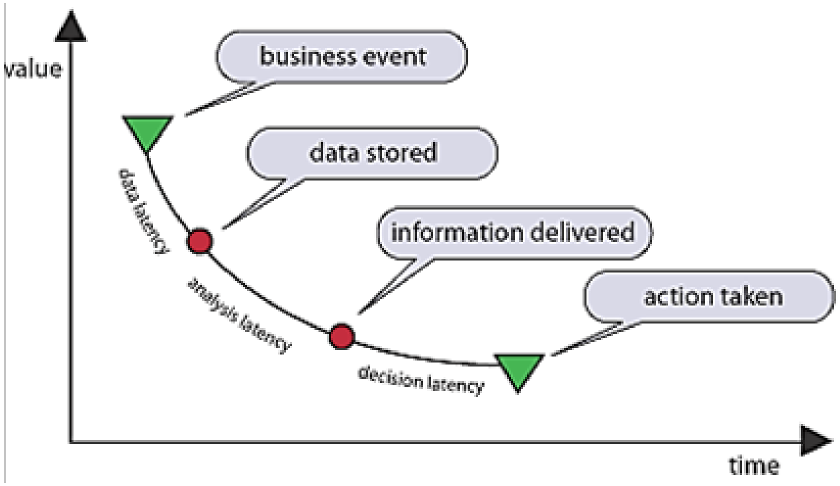
Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Real-time DW – Why?

- **Customer business scenario:** a utility company owns plants generating energy
- Existing DW supports planning by recommending:
  - The production capacity
  - The reserve capacity
  - When to buy supplemental energy, as needed

Real-time DW – Why?

- Each day is pre-planned on **historical behavior**
  - Peak demand periods are somewhat predictable
- Good planning is important because:
  - Expensive to have unused capacity!
  - Cheaper to buy energy ahead!
- Planning on last week's average is not enough



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Real-time DW – Why?

- Getting more **in-time** accuracy enhances operational business
  - Compare **today's** plant output and customer consumption volumes to yesterday's or last week's average
  - Know when to purchase additional options or supplies
- CustomerTarget: have the actual data from the operational environment available for analytics within a **5 minute lag**
- **Real-time ≠ fast**
  - Real time DW has the capability to **enforce** time constraints

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Real-time DW – Why?

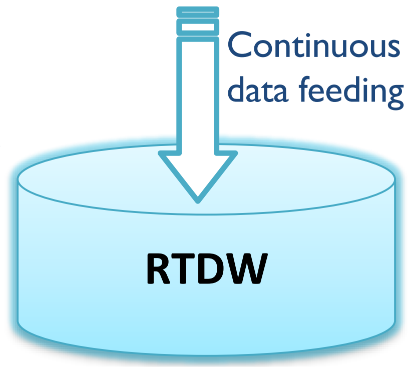
- The most difficult part for complying with the 5 minutes time constraint is the **ETL process**
  - ETL tools usually operate in batch mode on a certain schedule nightly, weekly or monthly
  - ETL typically involves **downtime** for the DW during the loading step (usually happens over night)
  - Data is extracted into flat files in the staging area **outside the DBMS,** where it is not available for querying
  - ETL may take **hours** to finish
- The ETL process needs to be re-designed to meet real-time constraints

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Real-time Data Warehousing

- Real-Time Data Warehouse (RTDW)

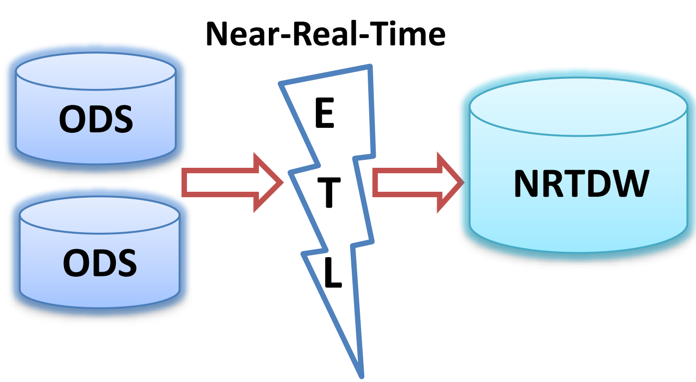
Individual changes occurring on the source systems are immediately forwarded to data warehouse in a best effort strategy.



- RTDW needs Real-Time ETL (Extract, Transform, Load)

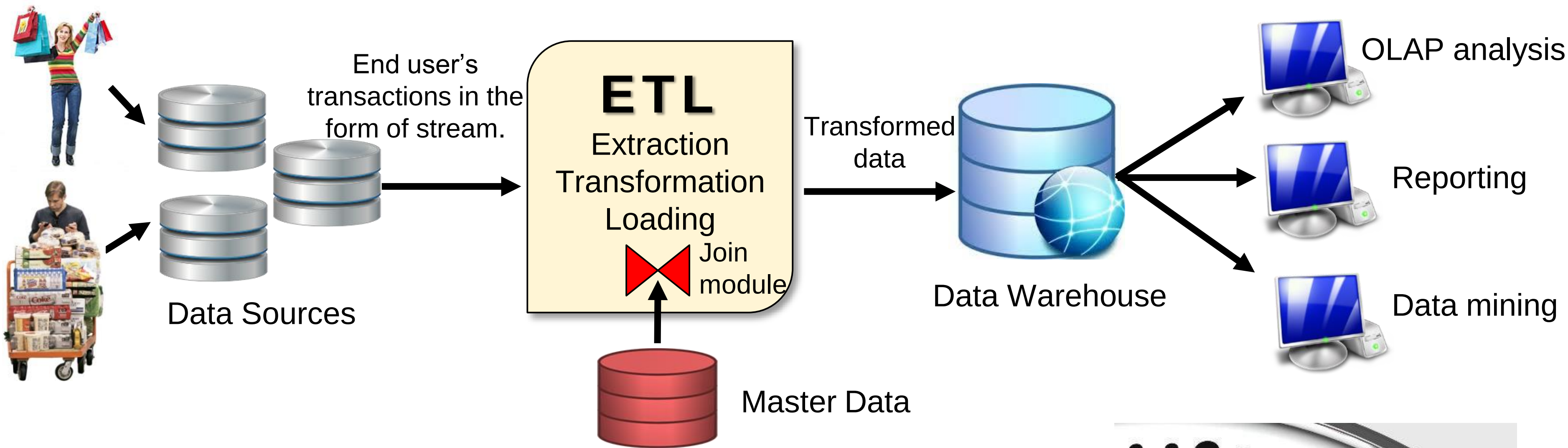
– ETL is a data integration layer between multi-data sources and data warehouse.

– Perform Extraction, Transformation and Loading tasks on Near-Real-Time basis



Real-time ETL

Real-time transformation is an important phase in an ETL layer where incoming source updates are transformed into warehouse format in an online fashion.



Real-time ETL (cont'd.)

Transformation examples: Key replacement, enrichment of data

DS$_1$

id	name
01	Sugar
02	Milk

DS$_2$

id	name
01	Milk
02	Butter

Data Sources

Join operator

id	Source	warehouse key
01	DS$_1$	101
02	DS$_1$	102
01	DS$_2$	102
02	DS$_2$	103

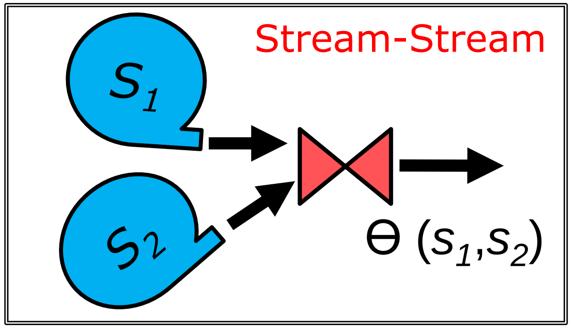
Master Data

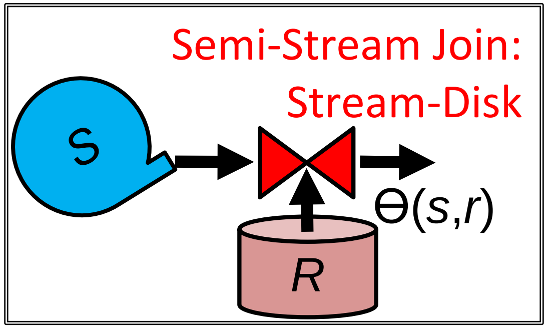
w_key	name
101	Sugar
102	Milk
103	Butter

Data Warehouse

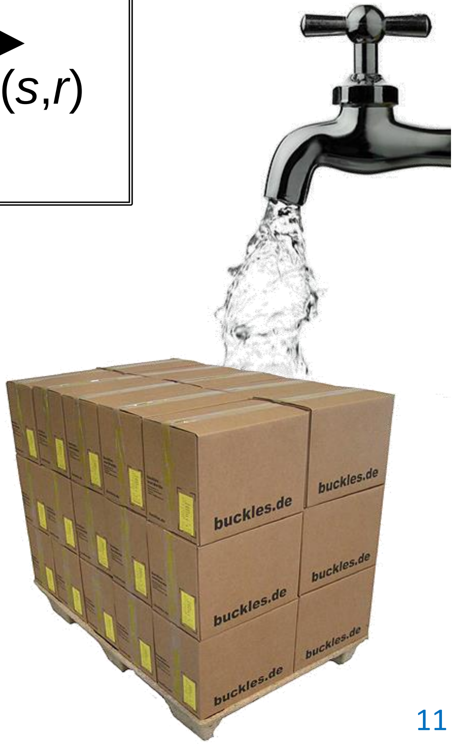
What is Stream-based Join?

- Stream-based join is an operation to combine the information coming from two or more data sources.
- Data sources can be in the form of streams or persistent data.





- Applications of stream-based join.
  - Enrichment of stream data with master data.
  - Key replacement in data warehouse.
  - Identification of duplicate records.
  - Merging of two or more data streams.



Research Challenges

- Challenge 1
  - Both inputs of the join operator have different arrival rates.
    - The stream input is fast, high volume and has an intermittent nature.
    - Master data input is comparatively slow due to the disk I/O cost.
  - It creates a bottleneck in join processing. The challenge here is to eliminate this bottleneck.
- Challenge 2:
  - Stream data is non-uniform therefore, an efficient approach is required to retrieve master data.

Existing Approaches

Stream-based joins

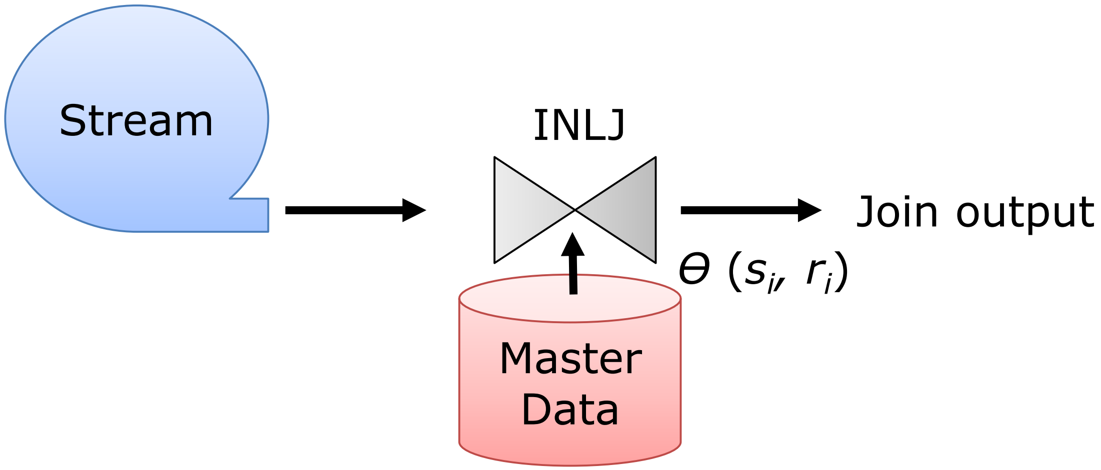
Full stream joins

- Symmetric Hash Join (SHJ)
- Early Hash Join (EHJ)
- X-Join
- Double Pipeline Hash Join (DPHJ)
- Hash Merge Join (HMJ)
- MJoin

Semi-stream joins

- Index Nested Loop Join (INLJ)
- Mesh Join (MESHJOIN)
- Partitioned-based Join
- Semi Stream Index Join (SSIJ)
- Reduced Mesh Join (R-MESHJOIN)
- Hybrid Join (HYBRIDJOIN)
- Cache Join (CACHEJOIN)

Index Nested Loop Join (INLJ)



Issues in INLJ

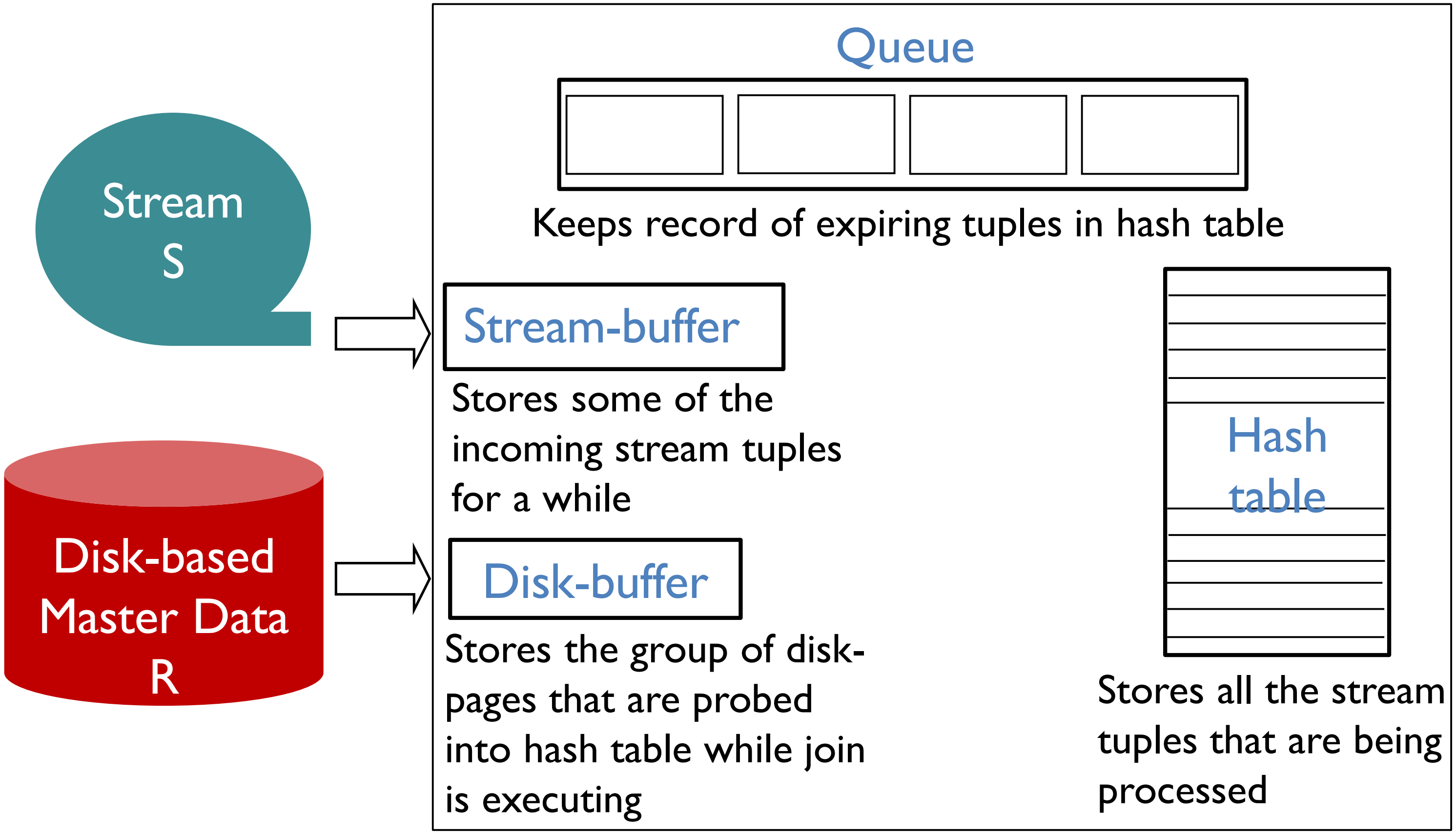
- INLJ processes only one stream tuple per iteration therefore solution is not practical for fast and huge volume stream data.
- INLJ does not amortize expensive disk reading cost on stream data.
- INLJ does not take into account some common characteristics of stream data e.g. skew in stream data.

MESHJOIN (Mesh Join)

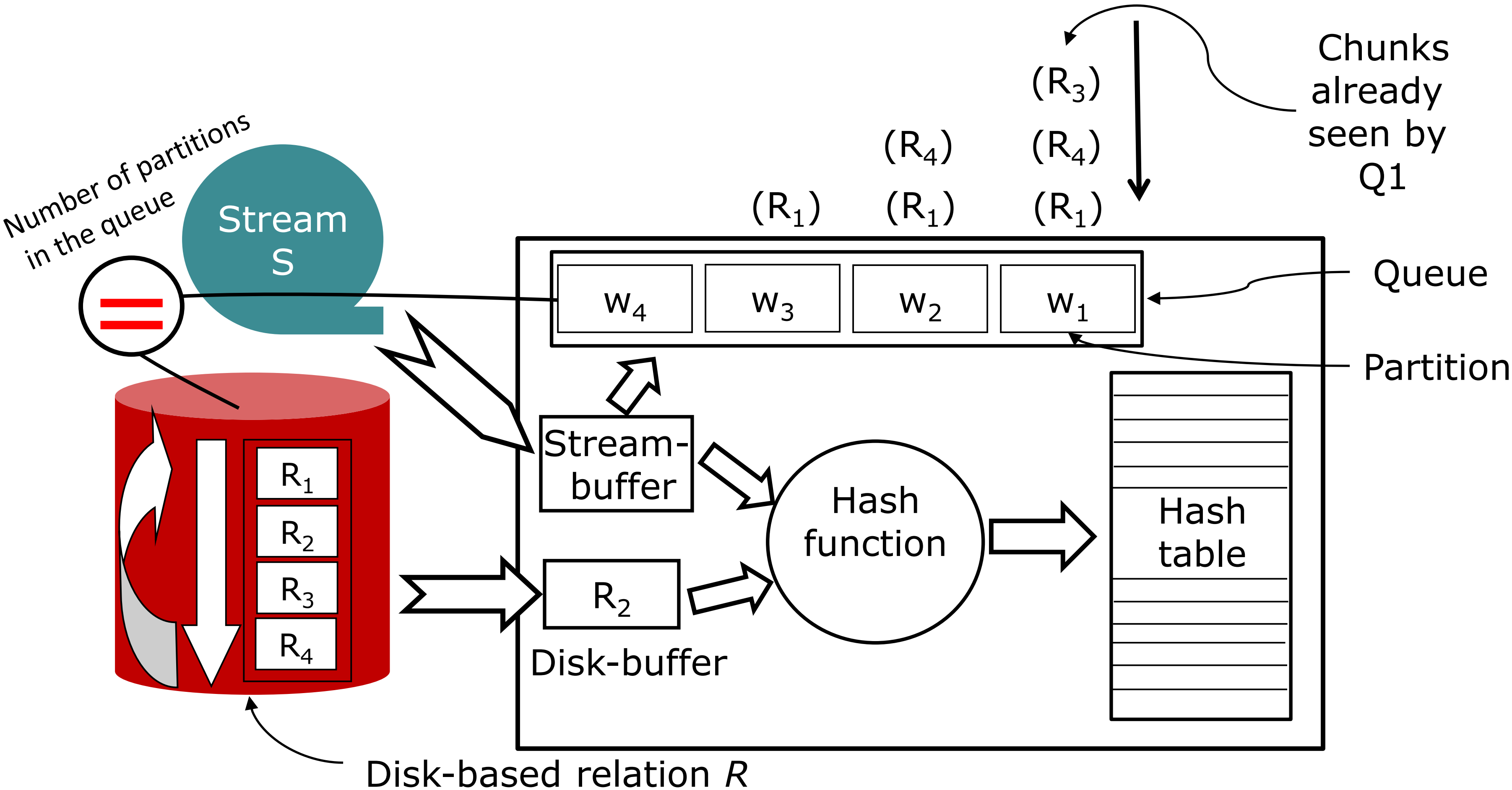
Features

- MESHJOIN (Mesh Join) has been proposed for processing stream data with master data.
- Designed for joining a stream S with disk-based relation R.
- Uses limited memory budget.
- Does not need an index.
- Works for any equijoin.

MESHJOIN Components



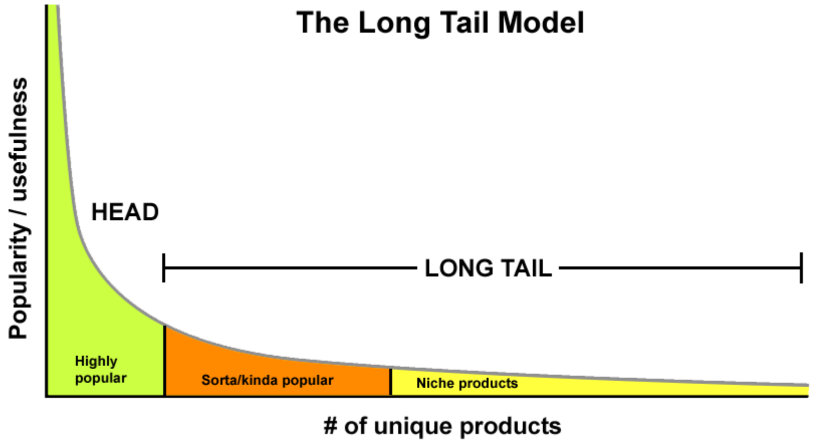
MESHJOIN Operation



- Size of stream-buffer = w tuples
- Size of disk-buffer = b pages
- Iterations required to bring all of R into memory = k (in this example k=4)

Problem in MESHJOIN

- Problem 1
  - Due to unnecessary dependency, memory distribution among the join components is not optimal.
- Problem 2
  - The performance of the algorithm is inversely proportional to the size of master data.
- Problem 3
  - Can not deal with intermittency in stream data.
- Problem 4
  - Typical characteristics of stream data such as non-uniform data are not considered.

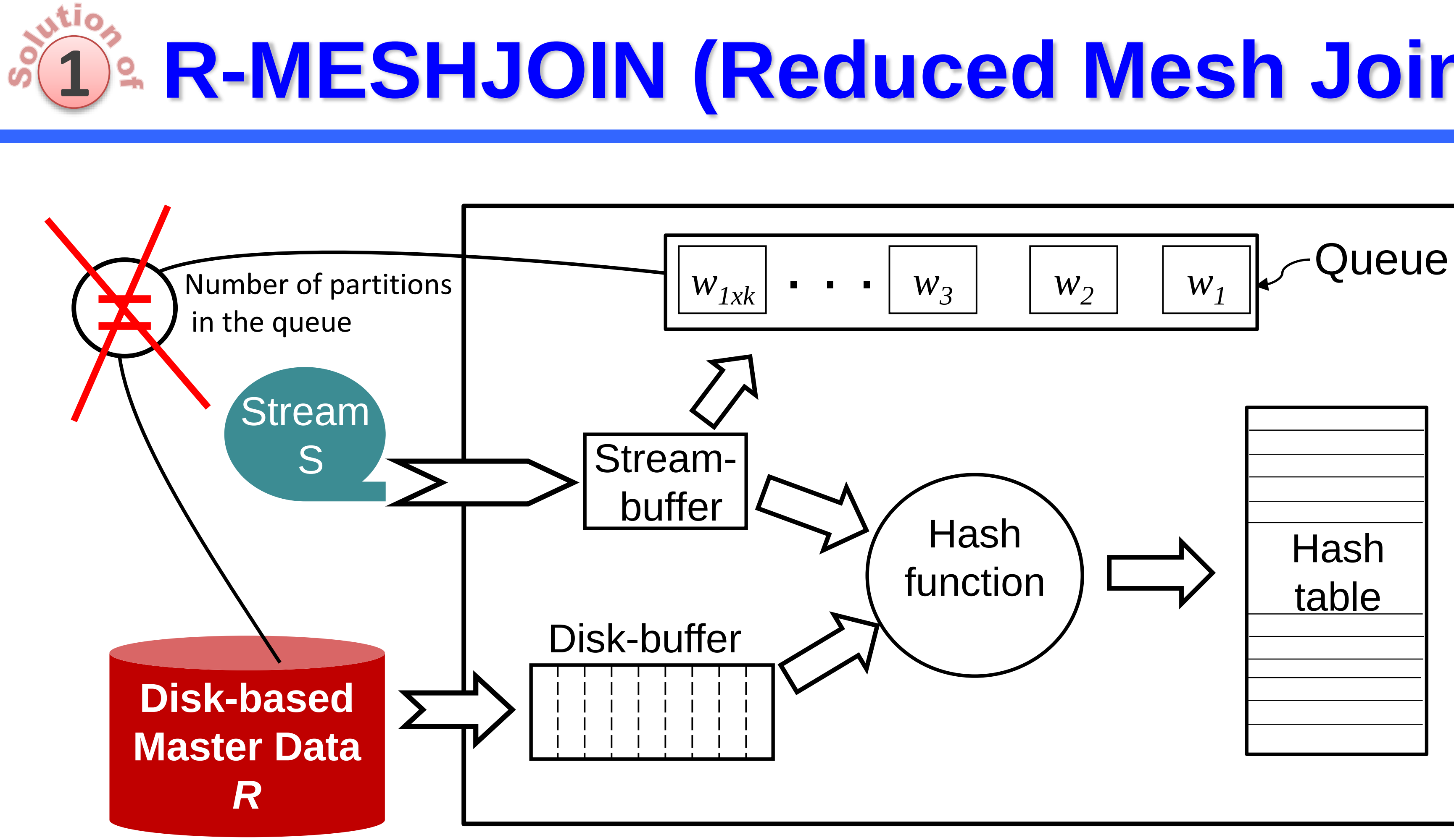


Solutions

We propose following novel algorithms to solve the highlighted problems.

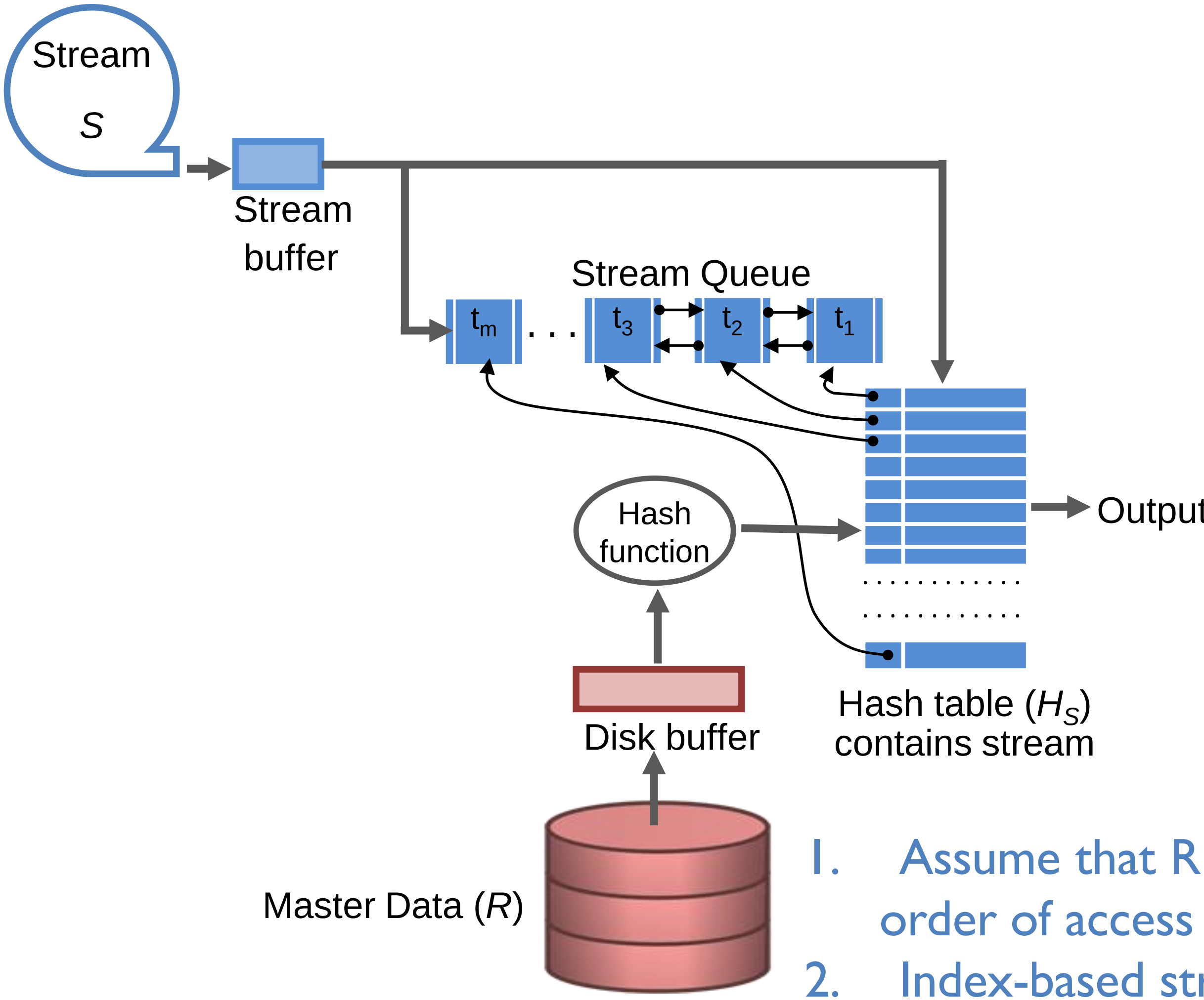
- **Reduced Mesh Join (R-MESHJOIN):-** clarifies the dependency among the components more appropriately.
- **Hybrid Join (HYBRIDJOIN):-** introduces an index-based strategy to access the master data.
- **Cache Join (CACHJOIN):-** considers non-uniform characteristic in stream data.

R-MESHJOIN (Reduced Mesh Join)



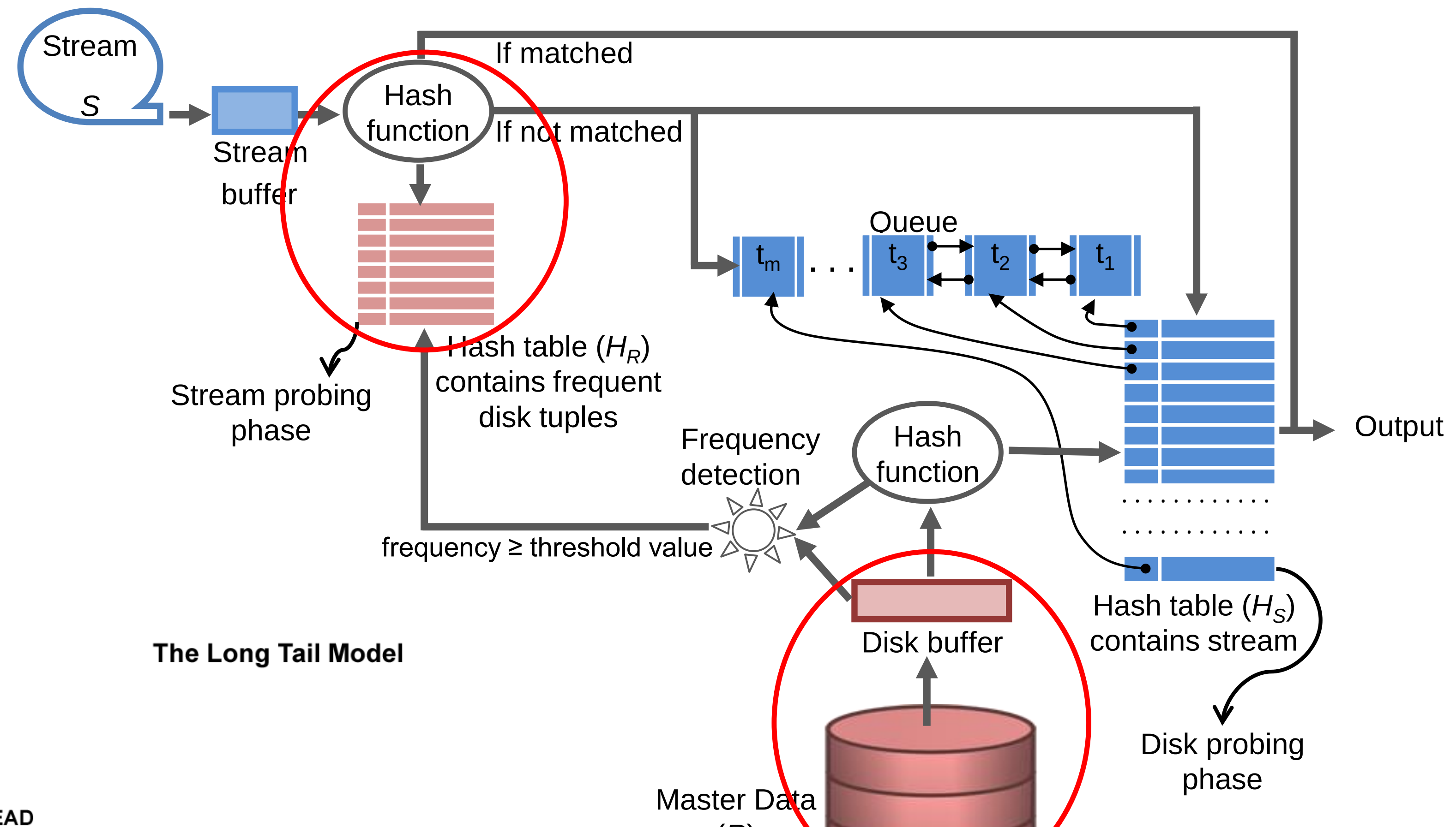
- Number of logical partitions in disk-buffer= *l*
- Size of each logical partition (pages) = b_p
- Size of disk-buffer (pages)= *b*
- Iteration required to load entire disk-based relation into memory=*k*

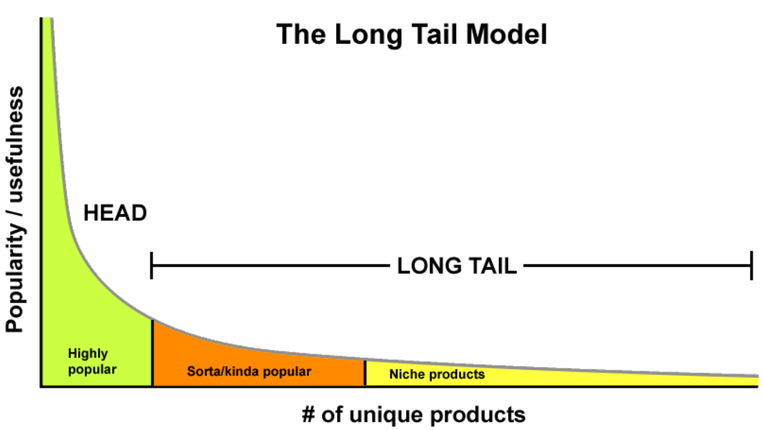
HYBRIDJOIN (Hybrid Join)



1. Assume that R is sorted in the order of access frequency.
2. Index-based strategy is used to access R.

CACHEJOIN (Cache Join)





Experimental Setup

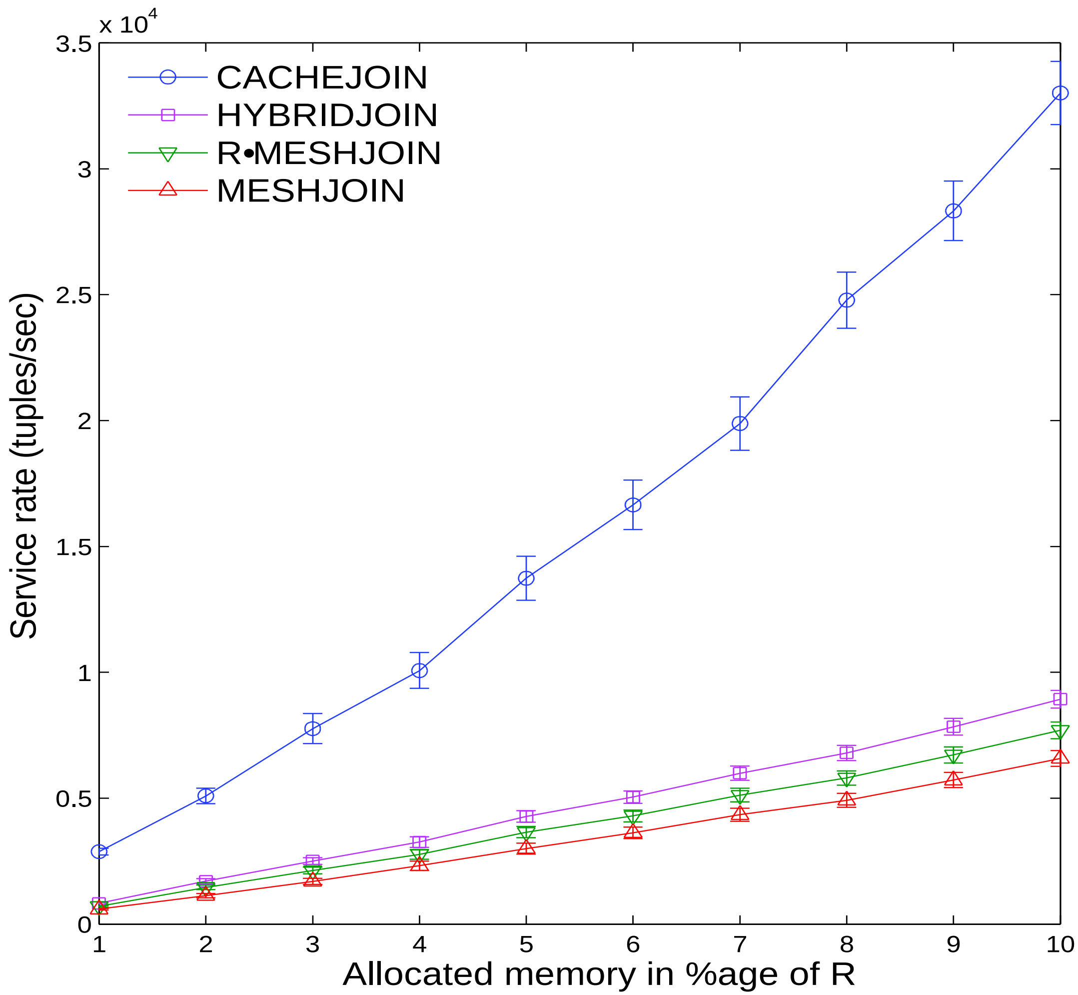
- Hardware specifications
  - Pentium-core-i5, 8G main memory, 500G HDD
- Synthetic dataset
  - Size of master data R, 100 million tuples (~11.18GB)
  - Size of each disk tuple, 120 bytes
  - Size of each stream tuple, 20 bytes
  - Size of each pointer in queue, 12 bytes
  - Based on Zipf's Law with skew of 0 to 1

Experimental Setup (cont'd.)

- TPC-H dataset
  - Create the datasets using a scale factor of 100
  - Uses table Customer as our master data table with each tuple o 223 bytes
  - Uses table Order as our stream data table with each tuple of 138 bytes
- Real dataset
  - Size of master data, 20 million tuples
  - Size of each master as well as stream tuple is 128 bytes
  - Source url: http://cdiac.ornl.gov/ftp/ndp026b/
- Evaluation metrics
  - We calculate confidence interval by considering 95% accuracy rate.

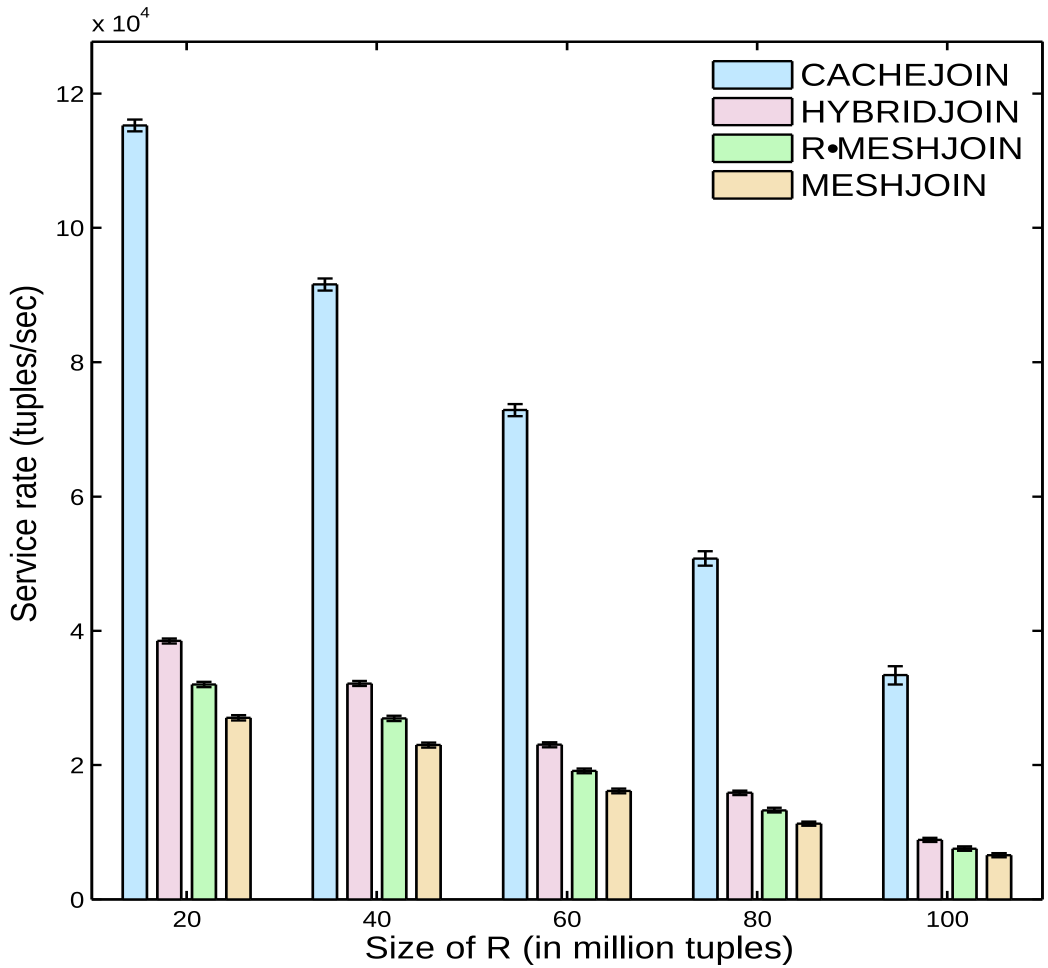
Performance Comparisons

Performance comparisons with 95% confidence interval while R=100 million tuples and **M varies in percentage of R**.



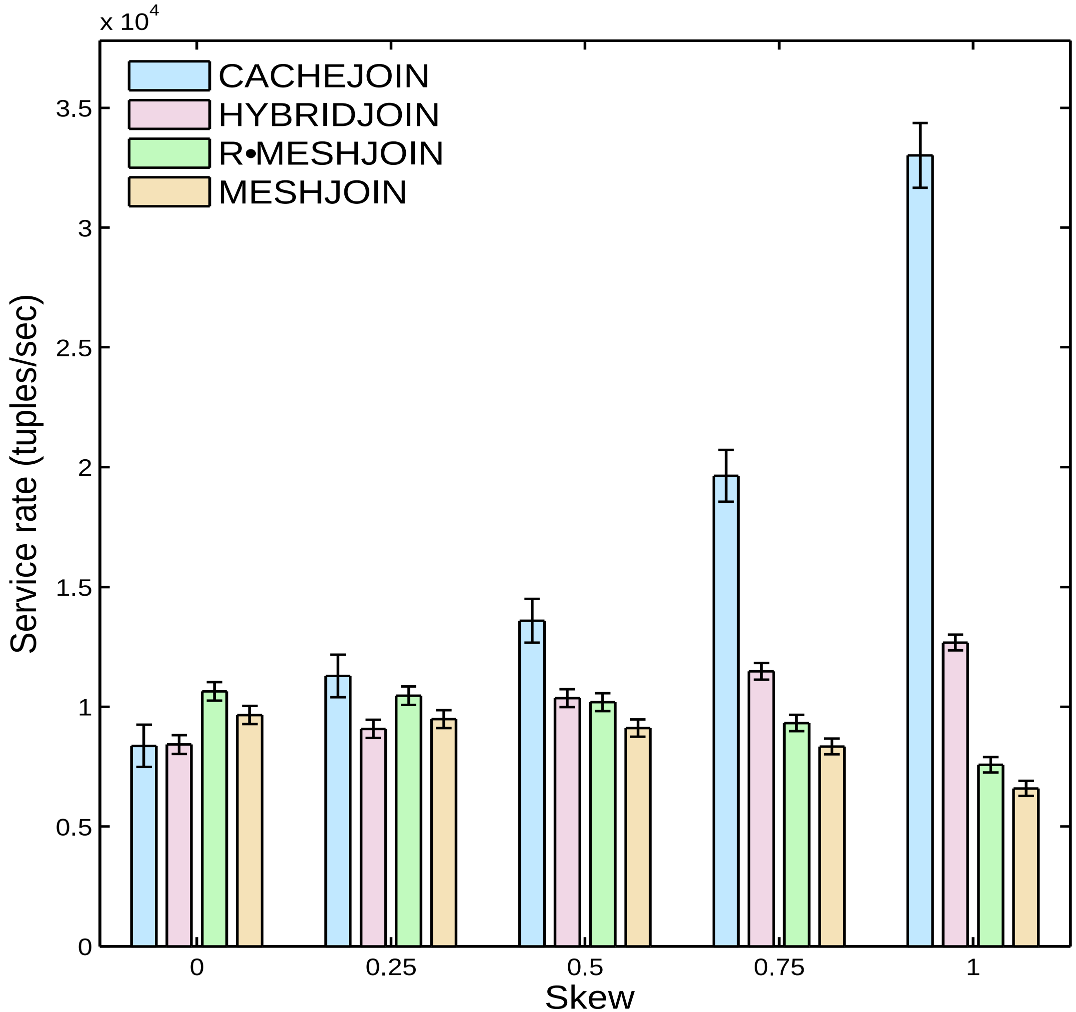
Performance Comparisons (cont'd.)

Performance comparisons with 95% confidence interval while M~1.2G and **R varies.**



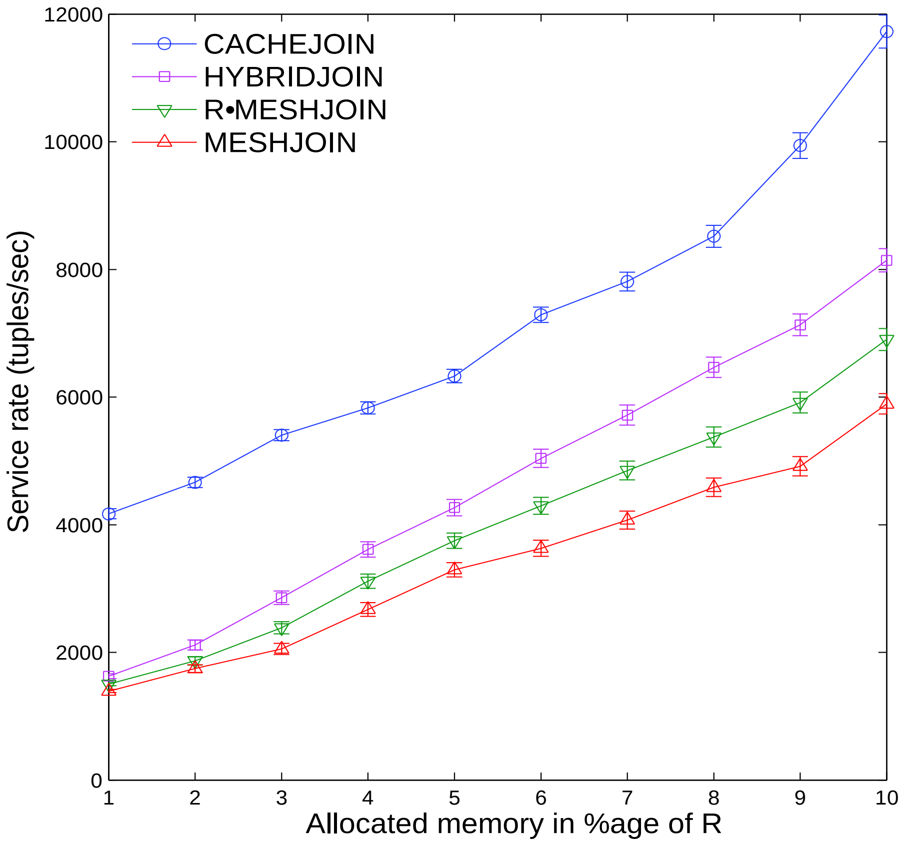
Performance Comparisons (cont'd.)

Performance comparisons with 95% confidence interval while M~1.2G, R=100 million tuples and **skew varies.**

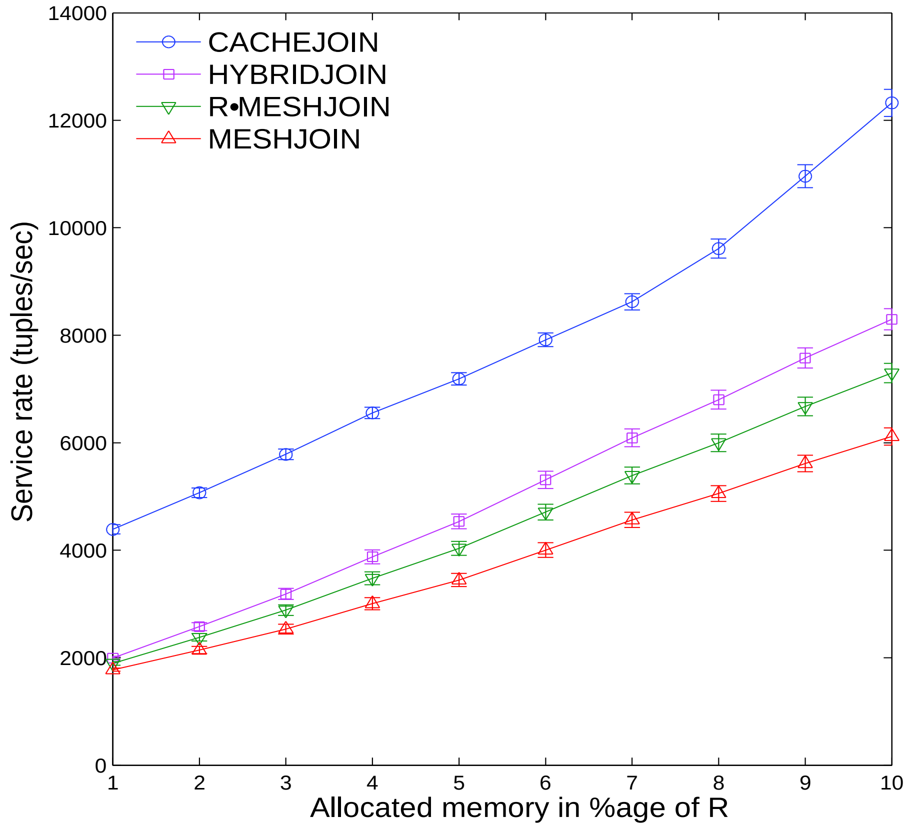


Performance Comparisons (cont'd.)

TPC-H datasets

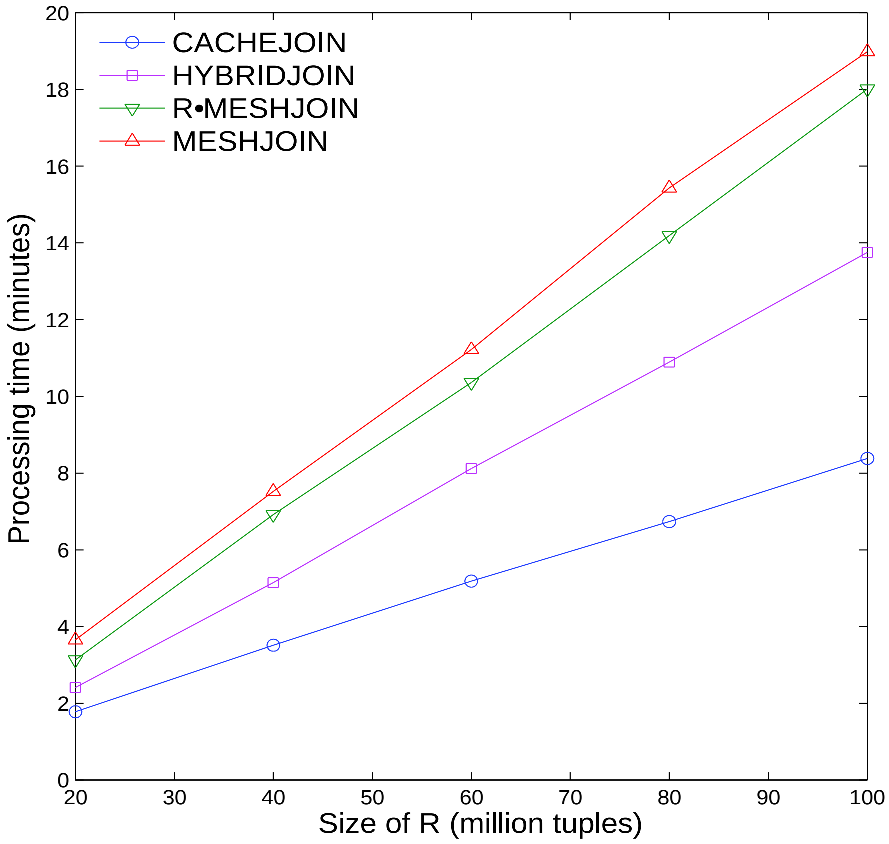


Real datasets

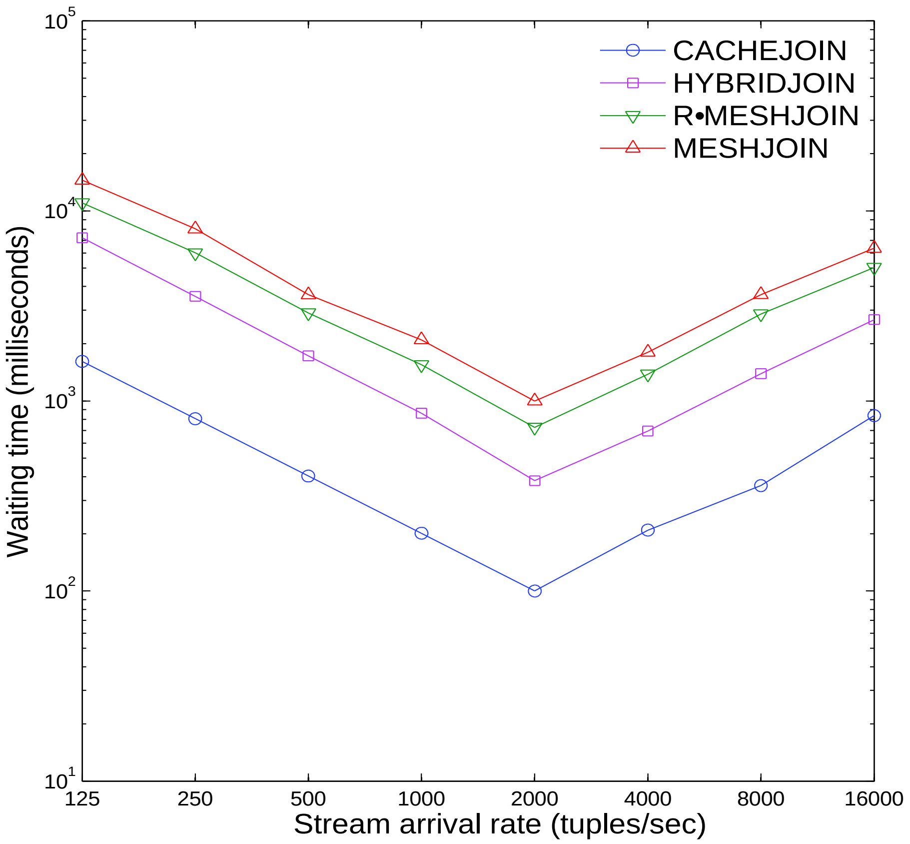


Performance Comparisons (cont'd.)

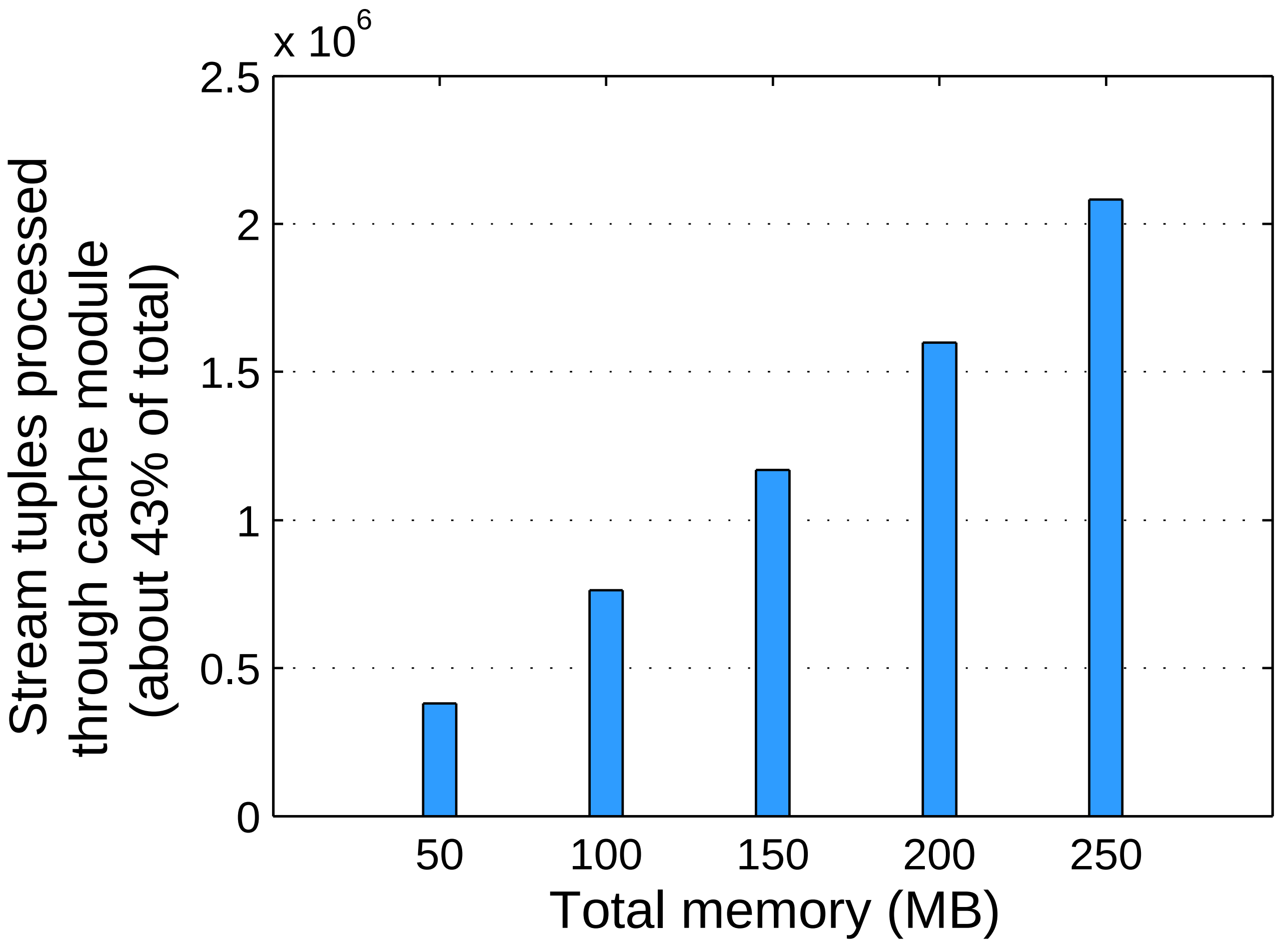
Processing time



Waiting time



Role of Cache Module in Performance



Total number of stream tuples processed through cache module in 4000 iterations.

Conclusions

- Stream processing has become a novel field in the area of data management due to its infinite characteristics.
- Stream-based join operators perform a key role in processing of stream data.
- A number of algorithms were designed to process semi-stream data however, they suffer with some limitations.
- We addressed these limitations in our research by presenting three novel algorithms.
- Our experimental evaluation proved the contribution of each algorithm in terms of performance.

About Me

- Publications
  - Published **one** book, **six** peer-reviewed journal articles, about **thirty** core-ranked conference and workshop papers, and **three** book chapters
- Professional Activities.
  - Keynote speaker, ICDIM'13
  - General track chair, ICDIM'13
  - Editorial member in IJES & JCI
  - PC member in ACSE, ADC, DOLAP, and DASFAA

- Student's Supervision
  - Supervising **three** Master and **two** PhD students

My Research

- Data stream processing in real-time Databases and Data Warehouses

Research in AUT
Prof. Albert Yeap, Dr. Russel Pears
Shoba Tegginmath

- Stream processing in Cloud Computing
- Big Data Management

Research with collaboration of AUT & UoA
Prof. Gillian Dobbie
Dr. Gerald Weber
Dr. Christof Lutteroth

- Natural Language Processing and Object Modelling

Research with collaboration of AUT & Birmingham University, UK
Dr. Behzad Bordbar, Dr. Imran S. Bajwa


Albert Yeap


Gillian Dobbie


Gerald Weber


Christof Lutteroth


Behzad Bordbar


Imran S. Bajwa


Shoba Tegginmath

Open Research Issues

- Data Stream Processing
  - Consider many-to-many equijoins
  - Dealing with non-equijoins
  - Parallelization of semi-stream joins
  - Semi-stream joins in cloud computing
  - Semi-stream joins on mobile and embedded platforms
- Other Research Areas
  - Big Data Management and Knowledge Engineering
  - Data Mining

Summary

- Real-time DW
  - Real-time ETL
  - Joins to process stream data
    - MESHJOIN
    - Problems in MESHJOIN
    - R-MESHJOIN
    - HYBRIDJOIN
    - CACHEJOIN
    - Performance comparisons
  - My research

Next Lecture

- Big Data
  - Real-time Data Processing Challenges
  - Real-time Join Algorithms