Outline

- What is a data stream?
- Applications of data stream management
- Models for data streams
- Data stream management systems
- Data stream mining
- Synopses structures
- Conclusion
What is a data stream?

- Golab & Oszu (2003): “A **data stream** is a **real-time**, **continuous**, ordered (implicitly by arrival time or explicitly by timestamp) **sequence of items**. It is impossible to control the order in which items arrive, nor is it feasible to locally **store** a stream in its entirety.”

- Structured records ≠ audio or video data

- Massive volumes of data, records arrive at a high rate

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Puis. A (kW)</th>
<th>Puis. R (kVAR)</th>
<th>U 1 (V)</th>
<th>I 1 (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>16/12/2006-17:26</td>
<td>5,374</td>
<td>0,498</td>
<td>233,29</td>
<td>23</td>
</tr>
<tr>
<td>16/12/2006-17:27</td>
<td>5,388</td>
<td>0,502</td>
<td>233,74</td>
<td>23</td>
</tr>
<tr>
<td>16/12/2006-17:28</td>
<td>3,666</td>
<td>0,528</td>
<td>235,68</td>
<td>15,8</td>
</tr>
<tr>
<td>16/12/2006-17:29</td>
<td>3,52</td>
<td>0,522</td>
<td>235,02</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
What is a data stream?

- Golab & Oszu (2003): “A data stream is a real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items. It is impossible to control the order in which items arrive, nor is it feasible to locally store a stream in its entirety.”

- Structured records ≠ audio or video data

- Massive volumes of data, records arrive at a high rate

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source</th>
<th>Destination</th>
<th>Duration</th>
<th>Bytes</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>12342</td>
<td>10.1.0.2</td>
<td>16.2.3.7</td>
<td>12</td>
<td>20K</td>
<td>http</td>
</tr>
<tr>
<td>12343</td>
<td>18.6.7.1</td>
<td>12.4.0.3</td>
<td>16</td>
<td>24K</td>
<td>http</td>
</tr>
<tr>
<td>12344</td>
<td>12.4.3.8</td>
<td>14.8.7.4</td>
<td>26</td>
<td>58K</td>
<td>http</td>
</tr>
<tr>
<td>12345</td>
<td>19.7.1.2</td>
<td>16.5.5.8</td>
<td>18</td>
<td>80K</td>
<td>ftp</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Outline

- What is a data stream?
- Applications of data stream processing
- Models for data streams
- Data stream management systems
- Data stream mining
- Synopses structures
- Conclusion
Applications of data stream processing

Data stream processing

- Process queries (compute statistics, activate alarms)
- Apply data mining algorithms
  - Real-time processing
  - One-pass processing
  - Bounded storage (no complete storage of streams)
  - Possibly consider several streams
Applications of data stream processing

Applications

- Real-time monitoring/supervision of IS (Information Systems) generating large amounts of data
  - Computer network management
  - Telecommunication calls analysis (BI)
  - Internet applications (eBay, Google, recommendation systems, click stream analysis)
  - Monitoring of power plants

- Generic software for applications where basic data is streaming data
  - Finance (fraud detection, stock market information)
  - Sensor networks (environment, road traffic, weather forecast, electric power consumption)
## Applications of data stream processing

### Standard data processing versus data stream processing

<table>
<thead>
<tr>
<th></th>
<th>Standard data processing technology</th>
<th>Data stream processing technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring, Business Intelligence applications</td>
<td>Data warehouses (unscalable)</td>
<td>Querying and mining ‘on the fly’ (scalable)</td>
</tr>
<tr>
<td>Applications with basic streaming data</td>
<td>Specific development without database technology</td>
<td>Generic tools for processing data</td>
</tr>
</tbody>
</table>
Applications of data stream processing

Let’s go deeper into some examples

- Network management
- Stock monitoring
- Linear road benchmark
Applications of data stream processing

Network management

- Supervision of a computer network
  - Improvement of network configuration (hardware, software, architecture)
  - Measurements made on routers (Cisco Netflow)

![Network supervision center diagram]
Applications of data stream processing

Network management

▷ Information about IP sessions going through a router
▷ Huge amounts of data (300 Go/day, 75000 records/second when sampling 1/100)
▷ Typical queries:
  • 100 most frequent (@S, @D) on router R1 …
  • How many different (@S, @D) seen on R1 but not R2 …
    … during last month, last week, last day, last hour?

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Duration</th>
<th>Bytes</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>10.1.0.2</td>
<td>16.2.3.7</td>
<td>12</td>
<td>20K</td>
<td>http</td>
</tr>
<tr>
<td>18.6.7.1</td>
<td>12.4.0.3</td>
<td>16</td>
<td>24K</td>
<td>http</td>
</tr>
<tr>
<td>12.4.3.8</td>
<td>14.8.7.4</td>
<td>26</td>
<td>58K</td>
<td>http</td>
</tr>
<tr>
<td>19.7.1.2</td>
<td>16.5.5.8</td>
<td>18</td>
<td>80K</td>
<td>ftp</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Applications of data stream processing

Stock monitoring

- Stream of price and sales volume of stocks over time
- Technical analysis/charting for stock investors
- Support trading decisions

- Notify me when the price of IBM is above $83, and the first MSFT price afterwards is below $27.
- Notify me when some stock goes up by at least 5% from one transaction to the next.
- Notify me when the price of any stock increases monotonically for ≥30 min.
- Notify me whenever there is double top formation in the price chart of any stock.
- Notify me when the difference between the current price of a stock and its 10 day moving average is greater than some threshold value.

Source: Gehrke 07 and Cayuga application scenarios (Cornell University)
Applications of data stream processing

Linear Road Benchmark
Benchmark to compare Data Stream Management Systems

Linear City

- Imaginary city: 100 miles x 100 miles
- 10 parallel express ways: 2 x (3 lanes + access ramp), cut into segments
- Vehicles send their position every 30’
- Unique clock, no delay on data transmission
- Random generator of vehicle traffic, one accident every 20 minutes

Source: Linear Road: A Stream Data Management Benchmark, VLDB 2004
Applications of data stream processing

Linear Road Benchmark

- Position reports (Time, VID, Spd, Xway, Lane, Dir, Pos)

- Queries issued by vehicles:
  - Account balance
  - Daily expenditures over the last 10 weeks
  - Time and price estimation for a trip, given day of week and time

Source: Linear Road: A Stream Data Management Benchmark, VLDB 2004
Applications of data stream processing

**Linear Road Benchmark**

**Toll depending on traffic**

- Notification of a price when entering a new segment, billing when leaving a segment
- Notification within 5’ after reception of position reports corresponding to a segment change
- Latest Average Velocity (LAV): average speed of vehicles in a segment and a direction for the last 5 minutes
- Toll:
  - Free if LAV > 40 MPH or if less than 50 vehicles in the segment
  - Free if detected accident in the next 4 segments
  - 2 * (numvehicules – 50)$^2$
- An accident is detected if at least 2 vehicles are stopped in the segment and lane for 4 position reports
- Accidents are notified to vehicles (they can react and change their route)
Applications of data stream processing

Where is the problem?

Example:

- Computation of daily electric power consumption by customer market segment, from customer meter data
  - Join between several streams
  - Join between stream data and customer database

Generic tools for processing streams
Avoid the ‘Store’, ‘Compute’, ‘Delete’ approach
Solution: incremental computation and definition of temporal windows for joins

Example:

- 100 most frequent @S IP addresses on a router
  - Maintain a table of IP addresses with frequencies?
  - Sampling the stream?

Face high (and varying) rate of arrivals
Exact versus approximate answers
Outline

- What is a data stream?
- Applications of data stream processing
- Models for data streams
- Data stream management systems
- Data stream mining
- Synopses structures
- Conclusion
Models for data streams

Structure of a stream

- Infinite sequence of items (elements)
- One item: structured information, i.e. tuple or object
- Same structure for all items in a stream
- Timestamping
  - « explicit » (date field in data)
  - « implicit » (timestamp given when items arrive)
- Representation of time
  - « physical » (date)
  - « logical » (integer)
## Models for data streams

### Protocol Streams

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source</th>
<th>Destination</th>
<th>Duration</th>
<th>Bytes</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>12342</td>
<td>10.1.0.2</td>
<td>16.2.3.7</td>
<td>12</td>
<td>20K</td>
<td>http</td>
</tr>
<tr>
<td>12343</td>
<td>18.6.7.1</td>
<td>12.4.0.3</td>
<td>16</td>
<td>24K</td>
<td>http</td>
</tr>
<tr>
<td>12344</td>
<td>12.4.3.8</td>
<td>14.8.7.4</td>
<td>26</td>
<td>58K</td>
<td>http</td>
</tr>
<tr>
<td>12345</td>
<td>19.7.1.2</td>
<td>16.5.5.8</td>
<td>18</td>
<td>80K</td>
<td>ftp</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Electrical Power Data

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Puis. A (kW)</th>
<th>Puis. R (kVAR)</th>
<th>U 1 (V)</th>
<th>I 1 (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>16/12/2006-17:26</td>
<td>5,374</td>
<td>0,498</td>
<td>233,29</td>
<td>23</td>
</tr>
<tr>
<td>16/12/2006-17:27</td>
<td>5,388</td>
<td>0,502</td>
<td>233,74</td>
<td>23</td>
</tr>
<tr>
<td>16/12/2006-17:28</td>
<td>3,666</td>
<td>0,528</td>
<td>235,68</td>
<td>15,8</td>
</tr>
<tr>
<td>16/12/2006-17:29</td>
<td>3,52</td>
<td>0,522</td>
<td>235,02</td>
<td>15</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Models for data streams

Model of a stream

- Contents of the stream (observed values)
- Underlying signal
- Model:

  relationship between observed values and an underlying signal
Models for data streams

Contents of a stream

- Infinite sequence of items $x_i = (t_i, m_i)$
  - Observation time: $t_i = i$ if logical time
  - Observed descriptive values $m_i$ (numerical, symbolic, ID’s)

- Example:
  - Observation at $t_i = 12342$
    - $m_i = (10.1.0.2, 16.2.3.7, 12, 20K, http)$
    - (@S, @D, Duration, Volume, Protocol)

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source</th>
<th>Destination</th>
<th>Duration</th>
<th>Bytes</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>12342</td>
<td>10.1.0.2</td>
<td>16.2.3.7</td>
<td>12</td>
<td>20K</td>
<td>http</td>
</tr>
<tr>
<td>12343</td>
<td>18.6.7.1</td>
<td>12.4.0.3</td>
<td>16</td>
<td>24K</td>
<td>http</td>
</tr>
<tr>
<td>12344</td>
<td>12.4.3.8</td>
<td>14.8.7.4</td>
<td>26</td>
<td>58K</td>
<td>http</td>
</tr>
<tr>
<td>12345</td>
<td>19.7.1.2</td>
<td>16.5.5.8</td>
<td>18</td>
<td>80K</td>
<td>ftp</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Models for data streams

Modeling the stream

- Observation m: \( N \rightarrow A \times R \)
  - \( A \) = set of ID's, \( m(i) = (a, v) \)
  - Example:
    - \( A \) = set of IP addresses sending packets
    - \( m(i) = (@S, volume) \)

- Underlying signal M: \( A \times T \rightarrow R \)
  - \( T \): time (implicit or explicit)
  - \( M(●, t_i) \leftarrow \text{function}(M(●, t_{i-1}), m_i) \)
  - Reconstruction of the signal from the stream
  - Difficult if \(|A|\) is large

- Not a unique model for a stream
Models for data streams

Some canonical models of streams

- Time series
- Increments
  - "cash register"
    - Incrementing by positive values
  - "turnstile"
    - Incrementing by positive/negative values
Models for data streams

Examples:

- **Time series model:**
  - Observation of $m_i = (@S, v) \rightarrow M(i) = v$
  - *Time series representing the volume transmitted between $t_{i-1}$ and $t_i$ (for all IP addresses)*

- **Cash register model:**
  - Observation of $m_i = (a, v) \rightarrow M(a, t_i) = v$
  - *Volume transmitted by sending IP address*
Models for data streams

Windowing

Applying queries/mining tasks to the whole stream (from beginning to current time)

Applying queries/mining to a portion of the stream
Models for data streams

Windowing

Definition of windows of interest on streams

- Fixed windows: September 2007
- Sliding windows: last 3 hours
- Landmark windows: from September 1st, 2007

Window specification

- Physical time: last 3 hours
- Logical time: last 1000 items

Refreshing rate

- Rate of producing results (every item, every 10 items, every minute, …)
Models for data streams

Sliding window

Results

Beginning of the stream

$t_c$

Refreshment time

$t'_{c}$

$t$

Results
Outline

- What is a data stream?
- Applications of data stream processing
- Models for data streams
- Data stream management systems
- Data stream mining
- Synopses structures
- Conclusion
DSMS outline

Data Stream Management System (DSMS)

- The user point of view
  - Definition of a DSMS
  - DSMS data model
  - Queries in a DSMS
  - STREAM example with « Linear Road »

- The computer scientist point of view
  - Main architecture of DSMS
  - Approximate answers to queries

- Main existing DSMS
DSMS: definition

**DBMS - Data Base Management System**
- Data model (relational)
- Data is stored on disk
- SQL language
  - Creating structures
  - Inserting/updating/deleting data
  - Retrieving data (query)
- Good performance even with large volumes of data

**DSMS - Data Stream Management System**
- Data model (streams and permanent relations)
- Permanent relations are stored on disk but streams are processed on the fly
- SQL like query language
  - Standard SQL on permanent relations
  - Extended SQL on streams with windowing features
  - New paradigm of queries (continuous queries)
- Tools for capturing input streams and producing output streams
- Good performance: optimization of computer resources
  - Several streams
  - Several queries
  - Ability to face variations in arrival rates without any crash
DSMS outline

Data Stream Management System (DSMS)

- The user point of view
  - Definition of a DSMS
  - DSMS data model
  - Queries in a DSMS
  - STREAM example with « Linear Road »

- The computer scientist point of view
  - Main architecture of DSMS
  - Approximate answers to queries

- Main existing DSMS
DSMS: data model

- Permanent relation (table)
  - Tuple (row)
  - Attribute (column)

- Stream
  - Tuple (row), Attribute (column), Stream of tuples
DSMS: data model

- DSMS input
  - Standard permanent tables, for instance:
    - Meter-customer correspondence
    - Hourly normal consumption at 20°C
  - One or several data streams, for instance:
    - Electric power consumption (several customers)
    - Hourly outdoor temperatures by region

- DSMS output
  - Updates on standard permanent tables, for instance:
    - Hourly electric power consumption, aggregated by city, for the last 24 hours
  - One or several output streams, for instance:
    - Alarms to customers with an abnormal consumption during the last 24 hours
    - 10 customers with the highest consumption during the last 24 hours, sent every hour
DSMS outline

Data Stream Management System (DSMS)

- The user point of view
  - Definition of a DSMS
  - DSMS data model
  - Queries in a DSMS
  - STREAM example with « Linear Road »

- The computer scientist point of view
  - Main architecture of DSMS
  - Approximate answers to queries

- Main existing DSMS
DSMS: queries

- Concept of **continuous queries**

  - Standard query in a DBMS (**one-time query**)
    - Defined and executed once on data stored in the database
    - Data are persistent and queries are transient
    - Data are accessed on demand of a query
    - A query is finished when the last tuple has been produced

  - Queries in a DSMS: standard/continuous queries
    - Standard queries on standard tables
    - **Continuous queries** when a stream is involved:
      - Defined before the beginning of the stream and executed continuously
      - Permanent queries, transient data
      - Arriving records are pushed to queries
      - Result: output streams or updates on permanent tables
      - Incremental computation of queries (no storage of the whole streams)
        - $A(Q,t+1) \text{ can be computed from } A(Q,t), \text{ new records arrived between } t \text{ and } t+1$, and some temporary limited storage of context data
Main querying approaches for continuous queries
- Graphical combination of operators on streams
- Extensions of SQL to continuous queries
DSMS: queries

Graphical combination of operators on streams

![Aurora system model diagram]

**Fig. 1.** Aurora system model

Examples of operators:
- Filter, Map, Union, Join, ...

**Source:** Aurora: a new model and architecture for data stream management, VLDB Journal 2003
DSMS: queries

Extensions of SQL to continuous queries

- Querying streams in SQL like permanent tables
- Example

```
ORDERS ( DATE, ID_ORDER, ID_CUSTOMER, ID_DEPT, TOTAL_AMOUNT )
BILLS (DATE, ID_BILL, ID_ORDER, AMOUNT )

Several bills for 1 order

SELECT MONTH(ORDERS.DATE),ID_DEPT, SUM(TOTAL_AMOUNT) - SUM(AMOUNT) 
FROM ORDERS, BILLS 
WHERE ORDERS.ID_ORDER = BILL.ID_ORDER 
GROUP BY MONTH(ORDERS.DATE),ID_DEPT;
```

- Result(s?) of the query ?
DSMS: queries

Extensions of SQL to continuous queries

ORDERS (DATE, ID_ORDER, ID_CUSTOMER, ID_DEPT, TOTAL_AMOUNT)
BILLS (DATE, ID_BILL, ID_ORDER, AMOUNT)

SELECT MONTH(ORDERS.DATE), ID_DEPT, SUM(TOTAL_AMOUNT) - SUM(AMOUNT)
FROM ORDERS, BILLS
WHERE ORDERS.ID_ORDER = BILL.ID_ORDER
GROUP BY MONTH(ORDERS.DATE), ID_DEPT;

- Blocking operations:
  - Specification of windows

SELECT MONTH(ORDERS.DATE), ID_DEPT, SUM(TOTAL_AMOUNT) - SUM(AMOUNT)
FROM ORDERS [LAST 10 DAYS], BILLS [LAST DAY]
WHERE ORDERS.ID_ORDER = BILL.ID_ORDER
GROUP BY MONTH(ORDERS.DATE), ID_DEPT;

- Ponctuations
DSMS: queries

Extensions of SQL to continuous queries

```
ORDERS ( DATE, ID_ORDER, ID_CUSTOMER, ID_DEPT, TOTAL_AMOUNT )
BILLS (DATE, ID_BILL, ID_ORDER, AMOUNT )

SELECT MONTH(ORDERS.DATE), ID_DEPT, SUM(TOTAL_AMOUNT) - SUM(AMOUNT)
FROM ORDERS [LAST 10 DAYS], BILLS [LAST DAY]
WHERE ORDERS.ID_ORDER = BILL.ID_ORDER
GROUP BY MONTH(ORDERS.DATE), ID_DEPT;
```

- Incremental computation
DSMS outline

Data Stream Management System (DSMS)

- The user point of view
  - Definition of a DSMS
  - DSMS data model
  - Queries in a DSMS
  - STREAM example with « Linear Road »

- The computer scientist point of view
  - Main architecture of DSMS
  - Approximate answers to queries

- Main existing DSMS
DSMS: STREAM

STREAM project

- Stanford University
- General purpose DSMS
- New prototype built from scratch
- Several new ideas
- Two structures:
  - STREAMS: implicit logical timestamp
  - RELATIONS: tables with contents varying with time
- CQL Language (Continuous Query Language) based on SQL
- Specification of sliding windows
- Definition of several streams and queries
- Optimized execution plan for a set of queries (no new query)

- Demo site: http://www-db.stanford.edu/stream
- Project ended January 2006
DSMS: STREAM

Two structures

- STREAMS: implicit logical timestamp
- RELATIONS: tables with contents varying with time

Specification of sliding windows

- Physical sliding windows (time-based)
  - BILLS [NOW]
  - BILLS [RANGE UNBOUNDED]
  - BILLS [RANGE 5 MINUTES]

- Logical sliding windows (tuple-based)
  - BILLS [ROWS 10]

- Partitioned sliding windows
  - ORDERS [PARTITION BY ID_CUSTOMER ROWS 20]

Source: Talk from Jennifer Widom http://infolab.stanford.edu/stream/index.html#talks
**DSMS: STREAM**

STREAM – RELATION operators

**Streams**
- Window specification

**Relations**
- Special operators: *I*stream, *D*stream, *R*stream
- Any relational query language

**ISTREAM**: stream of inserted tuples

**DSTREAM**: stream of deleted tuples

**RSTREAM**: stream of all tuples at every instant

CarLocStr (car_id, speed, expr_way, lane, dir, x_pos)

CarSegStr (car_id, speed, expr_way, dir, seg)
   -- Computation of segment from position (stream)
SELECT car_id, speed, expr_way, dir, x_pos/5280
FROM CarLocStr;

CarSegEntryStr (car_id, expr_way, dir, seg)
   -- Current segment of a vehicle (insertion stream)
ISTREAM ( SELECT * FROM CurCarSeg );

CurCarSeg (car_id, expr_way, dir, seg)
   -- Current segment of a vehicle (relation)
SELECT car_id, expr_way, dir, seg
FROM CarSegStr [Partition By car_id Rows 1];

SegAvgSpeed (expr_way, dir, seg, speed)
   -- average speed of vehicles on each segment
   -- during the last 5 minutes (relation)
SELECT expr_way, dir, seg, AVG(speed)
FROM CarSegEntryStr [Range 5 Minutes]
GROUP BY expr_way, dir, seg;

SegVolume (expr_way, dir, seg, volume)
   -- instant number of car in each segment
   -- (relation)
SELECT expr_way, dir, seg, COUNT(*)
FROM CurCarSeg
GROUP BY expr_way, dir, seg;

SegToll (expr_way, dir, seg, toll)
   -- toll for each segment. No tuple for a segment if toll is free (relation)
SELECT S.expr_way, S.dir, S.seg, 2 * (V.volume – 150) * (V.volume – 150)
FROM SegAvgSpeed as S, SegVolume as V
WHERE S.expr_way = V.expr_way AND S.dir = V.dir AND S.seg = V.seg AND S.speed < 40.00;

Toll notification to each vehicle
RSTREAM ( SELECT E.car_id, E.seg, T.toll
FROM CarSegEntryStr [Now] as E, SegToll as T
WHERE E.expr_way = T.expr_way
AND E.dir = T.dir AND E.seg = T.seg);
DSMS outline

Data Stream Management System (DSMS)

- The user point of view
  - Definition of a DSMS
  - DSMS data model
  - Queries in a DSMS
  - STREAM example with « Linear Road »

- The computer scientist point of view
  - Main architecture of DSMS
  - Approximate answers to queries

- Main existing DSMS
Main architecture of DSMS

- Still very unstable
- One generic architecture proposed by Golab et Ozsu (2003):

Source: Golab & Özsu 2003
Main architecture of DSMS

Some general problems and solutions

- Computation of memory needs for a query
  - Exact or approximate result

- Generation of execution plans for queries
  - Combination of operators applied to streams + queuing files + temporary storage + scheduler
  - Optimization of use of memory and CPU:
    - Sharing of execution plans, queuing files, buffers, temporary storage
    - Index of queries
  - Dynamic change of execution plans (variations in streams, new queries)
  - Distribution of processing (sensor networks)

- Quality of service
  - Maintain service in case of scratch, recovery from scratch
  - Maintain service when arrival rates increase
Main architecture of DSMS

Fig. 2. A simple query plan illustrating operators, queues, and synopsis.

Source: STREAM (Arasu et al. 2004)

Fig. 3. A query plan illustrating synopsis sharing.
DSMS outline

Data Stream Management System (DSMS)

- The user point of view
  - Definition of a DSMS
  - DSMS data model
  - Queries in a DSMS
  - STREAM example with « Linear Road »

- The computer scientist point of view
  - Main architecture of DSMS
  - Approximate answers to queries

- Main existing DSMS
Approximate answers to queries

When?

- Queries needing unbounded memory
  - Ex: *10 most present IP addresses on a router*

- Too much queries/too rapid streams/too high response time requirements
  - CPU limit
  - Memory limit

**Solution:** approximate answers to queries

- Sliding windows
- Refreshment rate (*batch processing*)
- Sampling and load shedding
- Definition of synopses (summaries)
Approximate answers to queries

Load shedding

Goal
- Face (dynamically) high arrival rates in streams by sampling tuples
- Control the error using a quality of service function

Principle
- Set sampling operators in the data flow diagram
- Optimize dynamically the location/rate of sampling operators
Approximate answers to queries

Example of load shedding approach: Babcock, Datar and Motwani (STREAM Project)

- Aggregate queries:
  - SUM, COUNT
  - Intermediate selections
  - External joins with fixed relations by foreign keys

Figure 1. Data Flow Diagram
Approximate answers to queries

Parameters of the problem

- For each operator $O_i$: selectivity $s_i$, processing time of a tuple $t_i$
- For each terminal operator (SUM): result average $\mu_i$ and standard-deviation $\sigma_i$
- For each stream: $r_i$ arrival rate of tuples
- For each operator $O_i$: $p_i$ is the number of tuples to send to it by unit of time

Problem definition

- Determine $p_i$'s by **minimizing the maximum error** on terminal operators under the constraint of system max load
DSMS outline

Data Stream Management System (DSMS)

- The user point of view
  - Definition of a DSMS
  - DSMS data model
  - Queries in a DSMS
  - STREAM example with « Linear Road »

- The computer scientist point of view
  - Main architecture of DSMS
  - Approximate answers to queries

- Main existing DSMS
Main existing DSMS

References: Golab & Oszu 2003, Gobel & Plagemann 2004

Principal general-purpose DSMS’s

⇔ STREAM: Université de Stanford
  - CQL language
  - Query optimization with good memory management
  - Approximate answer with synopses management

⇔ TelegraphCQ: Université de Berkeley
  - Extension of PostgreSQL
  - Continuous queries of CQL type
  - New queries can be added dynamically

⇔ Aurora (Medusa, Borealis): Brandeis, Brown University, MIT
  - Combination of operators (data flow diagram)
  - Load shedding with explicit definition of quality of service
  - Medusa and Borealis for distributed architecture
Main existing DSMS

Principal specialized DSMS’s

- Gigascope and Hancock : AT&T
  - Network monitoring
  - Analysis of telecommunication calls

- NiagaraCQ : University of Wisconsin-Madison
  - Large number of continuous queries on web content (XML-QL)

- Tradebot (finance), Statstream (statistics)

Commercial DSMS’s

- Streambase (cf. Aurora)
- Aminsight (cf. TelegraphCQ)
- Aleri
Main existing DSMS

But also:

⇒ Sensor networks
  • **Cougar**: Cornell University
  • **TinyDB**: University of Berkeley

⇒ Event Processing Systems
  • **Cayuga**: Cornell University
  • **Sase**: University of Massachusetts
Outline

- What is a data stream?
- Applications of data stream processing
- Models for data streams
- Data stream management systems
- Data stream mining
- Synopses structures
- Conclusion
Data stream mining: outline

Data stream mining

- Definition
- Decision tree
- PCA
- Clustering
Data stream mining: definition

**Goal**

*Apply data mining algorithms to (one) stream(s)*

**Constraints**

- Limited memory
- Limited CPU
- One-pass

**Windowing**
Data stream mining: definition

Windowing

- Beginning of the stream
- Current date
- Application to the whole stream
- Application to a sliding window
- Application to any portion of the stream
Data stream mining: definition

Windowing

- Whole stream
  - incremental algorithms

- Sliding window
  - incremental algorithms + ability to forget the past

- Any past portion
  - incremental algorithms + conservation of summaries
Data stream mining: definition

**Whole stream**
- Neural networks
- Non-additive methods: ex. decision tree

**Sliding window**
- Additive methods: ex. PCA

**Any portion of the stream**
- Temporal summaries: ex. clustering
Data stream mining: outline

Data stream mining

- Definition
- Decision tree
- PCA
- Clustering
Data stream mining: decision tree

Non-additive methods: the example of decision trees

**VFDT**: Very Fast Decision Trees (Domingos & Hulten 2000)

- $X_1, X_2, \ldots, X_p$: discrete or continuous attributes
- $Y$: discrete attribute to predict
- Elements of the stream $(x_1, x_2, \ldots, x_p, y)$ are examples
- $G(X)$: measure to maximize to choose splits (ex. Gini, entropy, …)
Data stream mining: decision tree

Hoeffding trees

**Idea:** *not necessary to wait for all examples to choose a split*

\[ G(X_j) \xrightarrow{n \to +\infty} G(X_j) \]

⇒ Minimum number of examples

\[
\text{if } G(X_j) - G(X_{j'}) \geq \varepsilon \quad \text{with} \quad \varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}
\]

\[
P(G(X_j) - G(X_{j'}) \geq 0) = 1 - \delta
\]
Data stream mining: decision tree

Hoeffding trees

Algorithm

- Maintain $G(X_j)$
- Wait for a minimum number of examples
- $j, k$ the 2 variables with highest values of $G$
- Split on $X_j$ when $G(X_j) - G(X_k) \geq \varepsilon$
- Recursively apply the rule by pushing new examples in leaves of the tree

- Sufficient statistics: $n_{ijkl}$ # of items with value $i$ of variable $j$ in class $k$ for leaf $l$
- VFDT: refinements on this algorithm
Data stream mining: outline

- Definition
- Decision tree
- PCA
- Clustering
Data stream mining: additive methods

Additive methods: the example of PCA

- Principal Component Analysis
- Items are elements \((x_1, x_2, \ldots, x_n)\) of \(\mathbb{R}^p\)
- Covariance/correlation matrix \(p \times p\)
- Incremental maintenance of \(p(p+1)\) statistics:

\[
\sum_{i=1}^{n} x_{ij} \quad \sum_{i=1}^{n} x_{ij}x_{ij'}
\]

- Recomputation of PCA at refreshment rate
Data stream mining: additive methods

\[
\sum_{i=1}^{n} x_{ij} \quad \sum_{i=1}^{n} x_{ij} \quad \ldots \quad \sum_{i=1}^{n} x_{ij} \quad \sum_{i=1}^{n} x_{ij}
\]

Sliding window of 24h

Refreshment every 1h
Data stream mining: outline

Data stream mining

- Definition
- Decision tree
- PCA
- Clustering
Data stream mining: clustering

Two distinct clustering problems

- Maintain k centers of clusters (k-median, k-medoids algorithms)
- Maintain k clusters with statistics on their contents

Problem of concept drift

- Evolution of distributions of data over time
- Windowing is one solution

→ Presentation of a clustering approach for evolving DS
Data stream mining: clustering

**Clustream** (Aggarwal et al. 2003)

- **Numerical variables**
- **2 phases:**
  - **On-line phase:** maintenance of a large number of ‘micro-clusters’ described by statistics of their contents
  - **Off-line phase:** use of micro-clusters to produce a final clustering
- **Mecanism to keep track of micro-clusters history**
Data stream mining: clustering

Representation of micro-clusters

- CVF: Cluster Feature Vector (BIRCH)

\[(n, CF1(T), CF2(T), CF1(X_1), CF2(X_1), \ldots, CF1(X_p), CF2(X_p))\]

\[CF1(X_j) = \sum_{i=1}^{n} x_{ij} \]

\[CF2(X_j) = \sum_{i=1}^{n} x_{ij}^2 \]

- Supports union/difference by addition/subtraction
- Incremental computation (elements are discarded)
Data stream mining: clustering

Maintenance of micro-clusters

- Fixed number of micro-clusters
- Initial micro-clusters (off-line)
- Each new item:
  - Find closest micro-cluster
  - ‘affectation’ to a cluster and update of CFV
  - Creation of a new micro-cluster (deletion or merge to make room)
- List of items of each micro-cluster not maintained
- History of micro-clusters fusions kept
Data stream mining: clustering

Mecanism to keep track of micro-clusters history

- Snapshots at regular time intervals
- Logarithmic storage structure (bounded)
- Tilted time windows
Data stream mining: clustering

Reconstitution of micro-clusters from any past portion
- Use addition/subtraction properties of micro-clusters
- Less detail for older data
- Approximation of the past portion
Data stream mining: clustering

Final clustering

- Hierarchical clustering of micro-clusters
- Use of CFV
  - Weight of micro-cluster
  - Centroid of micro-cluster
Data stream mining: conclusion

Conclusion on data stream mining

- Bounded memory
- Limited CPU
- One-pass algorithm
- Windowing
  - Whole stream
  - Sliding window
  - Any portion

- Summarizing the whole stream with bounded storage
  → tilted time windows
Outline

- What is a data stream?
- Applications of data stream processing
- Models for data streams
- Data stream management systems
- Data stream mining
- Synopses structures
- Conclusion
Synopses structures

**Motivation**
- Keeping track of a maximum of items in bounded space
- Some operations may still be long even with windowing
  - Approximate result based on summarized information

**Temporal management approach**
- Tilted time windows

**Memory management approach**
- Random samples
- Histograms
- Micro-clusters
- Sketches
Synopses structures: random samples

**Problem:** maintaining a random sample from a stream

‘Reservoir’ sampling (Vitter 85)

- Random sample of size $M$
  - Fill the reservoir with the first $M$ elements of the stream
  - For element $n$ ($n > M$)
    - Select element $n$ with probability $M/n$
    - If element $n$ is selected pick up randomly an element in the reservoir and replace it by element $n$

Random sampling from a sliding window:

‘Chain’ sampling (Babcock et al. 2002)
Synopses structures

**Motivation**
- Keeping track of a maximum of items in bounded space
- Some operations may still be long even with windowing
  - Approximate result based on summarized information

**Temporal management approach**
- Tilted time windows

**Memory management approach**
- Random samples
- Histograms
- Micro-clusters
- Sketches
Synopses structures: sketches

**Sketch**
- Synopsis structure taking advantage of high volumes of data
- Provides an approximate result with probabilistic bounds
- Random projections on smaller spaces (hash functions)

**Many sketch structures:** usually dedicated to a specialized task

**Examples of sketch structures**
- **COUNT** (Flajolet 85)
- **COUNT SKETCH** (Charikar et al. 04)
**Synopses structures: sketches**

**COUNT** (Flajolet 85)

**Goal**
- Number $N$ of distinct values in a stream (for large $N$)
- Ex. number of distinct IP addresses going through a router

**Sketch structure**
- SK: L bits initialized to 0
- H: hashing function transforming an element of the stream into L bits

**Example**

| 18.6.7.1 | → |
| 0 0 0 0 0 0 0 0 0 |

**H** distributes uniformly elements of the stream on the $2^L$ possibilities
**Method**

- **Maintenance and update of SK**
  - For each new element \( e \)
  - Compute \( H(e) \)
  - Select the position of the leftmost 1 in \( H(e) \)
  - Force to 1 this position in SK

**SK**

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**H(18.6.7.1)**

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**New SK**

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Result

- Select the position $R (0...L-1)$ of the leftmost 0 in $SK$
- $E(R) = \log_2 (\varphi^*N)$ with $\varphi = 0.77351…$
- $\sigma(R) = 1.12$

For $n$ elements already seen, we expect:

- $SK[0]$ is forced to 1 $N/2$ times
- $SK[1]$ is forced to 1 $N/4$ times
- $SK[k]$ is forced to 1 $N/2^{k+1}$ times
Synopses structures: sketches

COUNT SKETCH ALGORITHM (Charikar et al. 2004)

Goal

- $k$ most frequent elements in a stream (for large number $N$ of distinct values)
- Ex. 100 most frequent IP addresses going through a router

Input stream: $2, 0, 1, 3, 1, 2, 4, \ldots$

Output:

- $f(0) = 1$
- $f(1) = 2$
- $f(2) = 2$
- $f(3) = 1$
- $f(4) = 1$

$N = 4$
Synopses structures: sketches

-1
+1
-1
+1

<table>
<thead>
<tr>
<th></th>
<th>+12</th>
<th>+7</th>
<th>+23</th>
<th>+15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-5</td>
<td>-12</td>
<td>-23</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>B</td>
<td>+78</td>
<td>+56</td>
<td>+66</td>
<td>+65</td>
</tr>
</tbody>
</table>

1  2  ...  t
Synopses structures: sketches

**Sketch structure**
- \( h \): hash function from \([0, \ldots, N-1]\) to \([0, 1, \ldots, B]\)
- \( s \): hash function from \([0, \ldots, N-1]\) to \(+1, -1\)
- Array of \(B\) counters: \(C_1, \ldots, C_B\) (with \(B << N\))

**Sketch maintenance**
- when \(e\) arrives: \(C_{h(e)} += s(e)\)

**Use of sketch**
- Estimation of frequency of object \(e\): \(n_e \approx C_{h(e)} \cdot s(e)\)
- Actually \(t\) hash function \(h\) and \(t\) hash function \(s\):
  \[n_e \approx \text{median}_{j \in [1\ldots t]} \left( C_{h_j(e)} \cdot s_j(e) \right)\]
- Theoretical results on error depending on \(N, t\) and \(B\).
Synopses structures: sketches

Algorithm

Maintenance of a list \((e_1, e_2, \ldots, e_k)\) of the current \(k\) most frequent elements

For a new arriving element \(e\)

- Add \(e\) to the sketch structure
- Estimate frequency of \(e\) from the sketch structure
- If \(f(e) > f(e_k)\), remove \(e_k\) and insert \(e\) into the list
Outline

- What is a data stream?
- Applications of data stream processing
- Models for data streams
- Data stream management systems
- Data stream mining
- Synopses structures
- Conclusion
Conclusion

Very active area of research

Many practical applications in various domains

DSMS are more mature than data stream mining

DSMS
  - First commercial efficient systems
  - Event processing systems
  - Distributed DSMS

Data stream mining
  - Already several results
  - Still much work to do:
    - Identification and modeling of concept drift
    - Summarizing data stream history (also for DSMS)
    - Distributed data stream mining
References: general

*Querying and Mining Data Streams: You Only Get One Look. A tutorial.*
M.Garofalakis, J.Gehrke, R.Rastogi, Tutorial SIGMOD'02, Juin 2002.


http://www.cs.brandeis.edu/~linearroad/
References: DSMS


Amalgamated Insight, http://www.aminsight.com

Streambase software, http://www.streambase.com
References: data stream mining


*Mining data streams: a review.* M.M.Medhat, A.Zaslavsky and S.Krishnaswamy, in SIGMOD Record, Vol.34, N°2, pp.18-26, June 2005.
QUESTIONS ?