DATA STREAM PROCESSING (DSP)
Overview

1. General
2. Stream
3. Aurora
4. Conclusion
1 General Overview

1. Motivation Applications
2. Definition of Data Streams
3. Data Base Management System (DBMS) vs. Data Stream Management System (DSMS)
4. Stream Projects
1.1 Motivation Applications

- interpreting of sensor information
- traffic monitoring
- environmental monitoring
- recording of telephone calls
- logging web servers
- analysing monetary flows
1.1 Social Networks, Status Feeds

- Facebook, Twitter streams
- Queries:
  - Status analysis
  - Useful for advertising
- Status feed update examples:
  - Comments added to threads at any time
  - Posts removed from threads at any time
Abbildung (1), „Study Projects Nearly 45-Fold Annual Data Growth by 2020“ EMC Press Release
1.8 Zettabytes
(1.8 trillion gigabytes) of information will be created and replicated in 2011 and growing fast (it has grown by a factor of 9 in just five years)

Last year WE cracked the 1 zettabyte barrier for the first time!

1.2 Zettabytes of information created and replicated in 2010

The amount of information managed by enterprise datacenters will grow by 50 times.

Over the next decade, the number of servers (virtual and physical) worldwide will grow by 10 times.

The Digital Universe Paradox: Falling Costs & Rising Investment

while the pool of IT staff available to manage them will grow only 1.5x slightly.

Abbildung (2), IDC 2011 Digital Universe Study
1.2 What is a Data Stream

- A sequence of digitally encoded coherent signals (packets of data or data packets) used to transmit or receive information that is in the process of being transmitted.

- In a formal way, a data stream is any ordered pair $(s, \delta)$ where:
  - $s$ is a sequence of tuples and
  - $\delta$ is a sequence of positive real time intervals.
# 1.3 DBMS vs. DSMS

<table>
<thead>
<tr>
<th>DBMS</th>
<th>DSMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistent data (relations)</td>
<td>volatile data streams</td>
</tr>
<tr>
<td>Random access</td>
<td>Sequential access</td>
</tr>
<tr>
<td>One-time queries</td>
<td>Continuous queries</td>
</tr>
<tr>
<td>(theoretically) unlimited secondary storage</td>
<td>limited main memory</td>
</tr>
<tr>
<td>Only the current state is relevant</td>
<td>Consideration of the order of the input</td>
</tr>
<tr>
<td>relatively low update rate</td>
<td>potentially extremely high update rate</td>
</tr>
<tr>
<td>Little or no time requirements</td>
<td>Real-time requirements</td>
</tr>
<tr>
<td>Assumes exact data</td>
<td>Assumes outdated/inaccurate data</td>
</tr>
<tr>
<td>Plannable query processing</td>
<td>Variable data arrival and data characteristics</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1.4 DSMS Projects

- Amazon/Cougar (Cornell) – sensors
- Aurora (Brown/MIT) – sensor monitoring, dataflow
- Hancock (AT&T) – telecom streams
- Niagara (OGI/Wisconsin) – Internet XML databases
- OpenCQ (Georgia) – triggers, incr. view maintenance
- Stream (Stanford) – general-purpose DSMS
- Tapestry (Xerox) – pub/sub content-based filtering
- Telegraph (Berkeley) – adaptive engine for sensors
- Tribeca (Bellcore) – network monitoring
2 Stream Overview

- General
- Continuous Query Language (CQL)
- Windows
- CQL Examples
- Query Plan
- Query Approximation
- Summarize
2.1 The Stanford Data Stream Management System (STREAM)

- originally part of the homonymous research project at Stanford University
2.2 Continuous Query Language (CQL)

- Expressive SQL-based declarative language
2.3 Windows

- Mechanism for extracting a finite relation from an infinite stream
- Various window proposals for restricting operator scope.
  - Windows based on ordering attribute (e.g. time)
  - Windows based on tuple counts
2.3 Windows

- **Terminology**

  - **Start time**
  - **Current time**

  - **Sliding Window**
  - **Tumbling Window**
Find all Fotos where the name is like “foo” and they are at most 1 day old

Select *
From Fotos Fo [Range 1 Day Precending]
Where Fo.name like ‘foo’
Take the names of the 5 most recent Fotos bigger than 4000 bytes

Select F.name
From Fotos F
[Partition BY F.name Rows 5]
Where Fgröße < 40000
Find all Fotos and Filme where the names are equals

Select *
From Fotos Fo, Filme Fi [Range 1 Day]
Where Fo.name = Fi.name
2.4 Query 4 (Sample)

Get random 30% of the Fotos and all Films that are not older than one Day where the names are equals

Select *
From Fotos Fo Sample(30), Filme Fi [Range 1 Day]
Where Fo.name = Fi.name
2.4 Query 5 (Istream)

Insert the name of every new foto to an Stream

Select Istream(F.name)
From F [Rows 100]
Group By F.name
## 2.4 CQL Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Operator Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>select</td>
<td>relation-to-relation</td>
<td>Filters elements based on predicate(s)</td>
</tr>
<tr>
<td>project</td>
<td>relation-to-relation</td>
<td>Duplicate-preserving projection</td>
</tr>
<tr>
<td>binary-join</td>
<td>relation-to-relation</td>
<td>Joins two input relations</td>
</tr>
<tr>
<td>mjoin</td>
<td>relation-to-relation</td>
<td>Multiway join from [22]</td>
</tr>
<tr>
<td>union</td>
<td>relation-to-relation</td>
<td>Bag union</td>
</tr>
<tr>
<td>except</td>
<td>relation-to-relation</td>
<td>Bag difference</td>
</tr>
<tr>
<td>intersect</td>
<td>relation-to-relation</td>
<td>Bag intersection</td>
</tr>
<tr>
<td>antisemijoin</td>
<td>relation-to-relation</td>
<td>Antisemijoin of two input relations</td>
</tr>
<tr>
<td>aggregate</td>
<td>relation-to-relation</td>
<td>Performs grouping and aggregation</td>
</tr>
<tr>
<td>duplicate-eliminate</td>
<td>relation-to-relation</td>
<td>Performs duplicate elimination</td>
</tr>
<tr>
<td>seq-window</td>
<td>stream-to-relation</td>
<td>Implements time-based, tuple-based, and partitioned windows</td>
</tr>
<tr>
<td>i-stream</td>
<td>relation-to-stream</td>
<td>Implements $Istream$ semantics</td>
</tr>
<tr>
<td>d-stream</td>
<td>relation-to-stream</td>
<td>Implements $Dstream$ semantics</td>
</tr>
<tr>
<td>r-stream</td>
<td>relation-to-stream</td>
<td>Implements $Rstream$ semantics</td>
</tr>
</tbody>
</table>

Abbildung (4), [9]
2.5 Query Plan - Composition

- Query-Operators

- Inter-Operator-Queues
  - Connections between the operators

- Synopsis
  - summarizes the tuples seen so far, as needed for future evaluation of that operator
2.5 Query Plan (example)

Select * From S1 [Range 15 Minutes], S2 [Rows 1000]
Where S1.A = S2.A And S1.A < 20
2.5 Query Plan - Optimizations

Abbildung (7), [9]
Novel notions of optimization:

- Stream rate based [e.g. NiagaraCQ]
- Resource based [e.g. STREAM]
- QoS based [e.g. Aurora]
2.6 Approximation

- Why we need Approximation?
- CPU-limited Approximation
- Memory-limited Approximation
- Static Approximation
- Dynamic Approximation
2.6 Why do we need approximation

- Handling load – streams comming too fast
- Avoid unbounded storage and computation
- 2 factors can become a constraint:
  - CPU
  - Memory
2.6 CPU-limited Approximation

- Data arrived too fast
- load-shedding - dropping elements from query plans and saving the CPU time that would be required to process them to completion
too many queries -> memory becomes a constraint -> results may partly disappear
memory usage can be reduced at the cost of accuracy by reducing the size of synopses
Optimization during submitting a new query to the system
- window reduction
- sampling rate reduction
2.6 Static Approximation – window reduction

- Reduction of the size of the window
  - Saving of computation time and storage

- Exception:
  - Elimination of duplicates
  - Negations

Select *
From Fotos Fo, Filme Fi
[Range 1 Day]
Where Fo.name = Fi.name
2.6 Static Approximation – Sampling Rate Reduction

- Minimization of the Input-Stream

Select *
From Fotos Fo Sample(30), Filme Fi [Range 1 Day]
Where Fo.name = Fi.name
2.6 Dynamic Approximation

- Optimization if the system is already running
  - Synopsis Compression
  - Sampling/Load Shedding
2.6 Dynamic Approximation - Synopsis Compression

- Reduction of the Synopsis Size
- Same approach as in Memory-Limited Approximation

Abbildung (6), [9]
2.6 Dynamic Approximation - Sampling/Load Shedding

- Reduction of the memory usage
- Sampling:
  - Undistorted result
- Load-Shedding:
  - Easier to implement
2.7 Summarizing of STREAM

- CQL an expressive SQL-based declarative language
- Query Plan
  - One Query Plan for each Query
- Approximation
  - CPU-limited
  - Memory-limited
  - Static approximation
  - Dynamic approximation
3 Aurora Overview

- general
- Aurora vs. Stream
- Query Plan
- Optimizations
- Quality of Service
3.1 AURORA

- developed in cooperation of the M.I.T., the Brandeis University and the Brown University
### 3.2 STREAM vs. Aurora

<table>
<thead>
<tr>
<th>STREAM</th>
<th>AURORA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evey query has it own query plan</td>
<td>One big query plan for all queries</td>
</tr>
<tr>
<td>Synopses and queues</td>
<td>„Boxes and Arrows“ Paradigm</td>
</tr>
<tr>
<td>Direct entry of plans</td>
<td>Ad Hoc queries and views</td>
</tr>
<tr>
<td></td>
<td>Qos for each output stream</td>
</tr>
</tbody>
</table>
3.2 Aurora

![Diagram of Aurora system](image)

Abbildung (8), [10]
Based on eight primitive operations:

- **Windowed operators:**
  - Slide
    - Advances a window by sliding it downstream by some tuples
  - Tumble
    - Resembles slide except that consecutive windows have no tuples in common
    - Partitions a stream into disjoint windows
  - Latch
    - Produces a partially synthetic stream by interpolating tuples between actual tuples of an input stream

- **non-windowed operators:**
  - Filter
  - Drop
  - Map
  - Groupby
  - Join

- **Other Operators:**
  - Resample
3.3 Query Plan

- Only one big plan (AURORA)
  - Easier to optimize
  - Adding or deleting queries always leads to bigger overhead

- One query plan for each query (STREAM)
  - Optimization is more difficult
  - Adding or deleting is quite easy
3.3 Query Plan

- 3 Modes
  - Continual queries
  - Views
  - Ad-hoc queries

Abbildung (9) [10]
3.4 Optimization

- Dynamic Continuous Query Optimization
3.4 Dynamic Continuous Query Optimization

- Map-Operator
- Combination of Boxes
- Reordering of Boxes
3.4 Map-Operator

- Inserting of the map Operator for eliminating unneeded tuples
- System must provide operator signatures
3.4 Combination of Boxes

- Combination of different boxes
- Saves overhead
- Reducing quantity of boxes
3.4 Reordering of Boxes

- For suspending tuples earlier in the query plan
- For example with pushing down a filter operator
3.5 Quality of Services (QoS)

- Aim: improve quality of the output
- Multidimensional function with 3 properties:
  - Delay
  - Tuples delivered
  - Output value
3.5 Quality of Service

(a) Delay-based
(b) Drop-based
(c) Value-based

Abbildung (11),[10]
4 Conclusion

- Data Stream Processing is getting more and more important
- So there are several projects to deal with it
- Between these projects you can find some differences
- Development is going on
THANK YOU

Any questions?