A DBMS kernel sailing streams

Erietta Liarou

INS1- Database Architectures

CWI, Amsterdam, The Netherlands
• Evaluate queries over the already stored data
• DB applications are everywhere!
**DBMS versus DSMS**

- Evaluate queries over the already stored data
- DB applications are everywhere!

- Continuous queries are waiting for the future data
- Applications where the databases are inefficient
Motivation

- Modern (stream) applications require both management of \textit{stored} and \textit{streaming} data

- Nowadays stream systems are built \textit{from scratch}

- Redesign operators and optimizations

- Relational Databases are considered inefficient and too complex
DataCell

- We design the DataCell on top of an existing Database Kernel

- Exploit database techniques, query optimization and operators

- Provide full language functionalities (SQL’03)

- It is possible!
  - We show that we can achieve high stream processing and scalable performance
  - A plethora of new research issues arises
    - real-time processing
    - multi-query processing/scheduling
The Basic Idea

- *Trick* the Database Kernel to consider a continuous queries as a normal one-time query.
  - Scheduling the trigger conditions
  - Wait to collect a few tuples and then evaluate the query

- Use the storage infrastructure to temporarily store the streaming data
  - Once a tuple is seen, it is *dropped*
Against a specialized steam engine

Against a specialized steam engine

```sql
SELECT max(s1.x1), avg(s2.x1) 
FROM stream1 s1, stream2 s2 
WHERE s1.x2 = s2.x2
```

IW | = 10^3 and IW | = 10^5 tuples

IWL = IW | /64 =~ 16 to IWL = IW | /64 =~ 1600 tuples

![Graph showing performance of different systems with varying window sizes](image-url)
Thank you!
The MonetDB/DataCell stack

SQL Query

Query parser + CQ

Query Optimizer + DC opt

Continuous Query Scheduler

MAL

MAL Interpreter

Query Executor