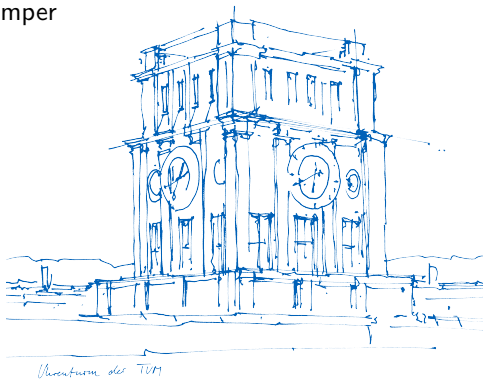
UMBRA

Relation-Based In-Database Stream Processing

Christian Winter, Thomas Neumann, Alfons Kemper

Technical University of Munich

CDMS @ VLDB 2023



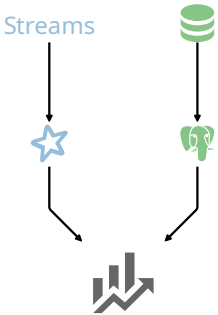
Motivation

UMBRA

- Stream data analytics is growing increasingly complex

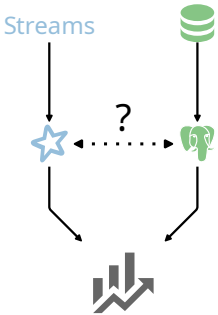
Motivation

- Stream data analytics is growing increasingly complex
- Streams are enriched and analyzed with historic data
  - Business reporting
  - Log monitoring
  - Logistics



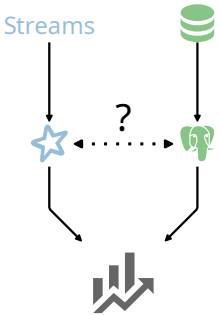
Motivation

- Stream data analytics is growing increasingly complex
- Streams are enriched and analyzed with historic data
  - Business reporting
  - Log monitoring
  - Logistics
- ⚠ Stream processing engines inefficient for historic data



Motivation

- Stream data analytics is growing increasingly complex
- Streams are enriched and analyzed with historic data
  - Business reporting
  - Log monitoring
  - Logistics
- Stream processing engines inefficient for historic data
- Database systems not built for ephemeral data

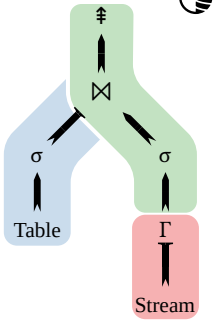


Complex stream analytics in a single system is hard

Current Solutions

UMBRA

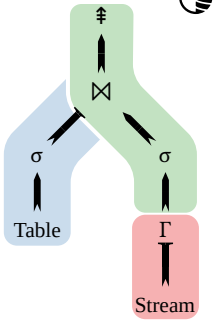
- Historic data in stream processing engines
  - Read-only
  - Optimized for filter-and-aggregate queries



Current Solutions

UMBRA

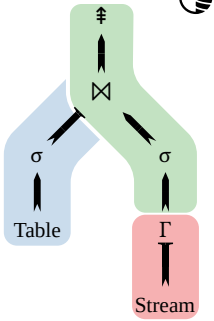
- Historic data in stream processing engines
  - ⚠ Read-only
  - ⚠ Optimized for filter-and-aggregate queries
- View-based stream processing
  - ✓ Highly performant
  - ⚠ Requires fine-tuned and invasive integration



Current Solutions

UMBRA

- Historic data in stream processing engines
  - ⚠ Read-only
  - ⚠ Optimized for filter-and-aggregate queries
- View-based stream processing
  - ✓ Highly performant
  - ⚠ Requires fine-tuned and invasive integration

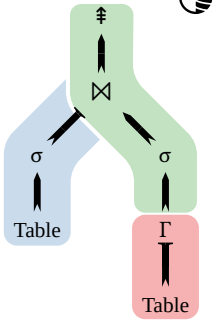


Current solutions are complex to implement and integrate

Relation-Based Integration

UMBRA

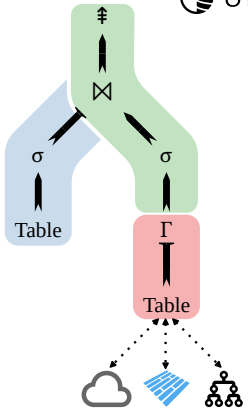
- Relations are entry point for all data in a DBMS



Relation-Based Integration

UMBRA

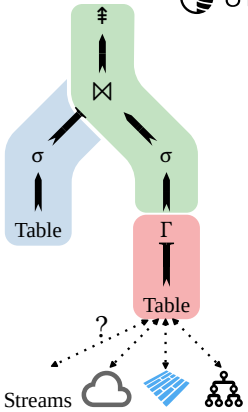
- Relations are entry point for all data in a DBMS
- Database systems utilize physical data independence
  - Masks remote data access
  - Independence of storage formats, e.g., Parquet
  - Optimized access paths for OLTP or OLAP



Relation-Based Integration

UMBRA

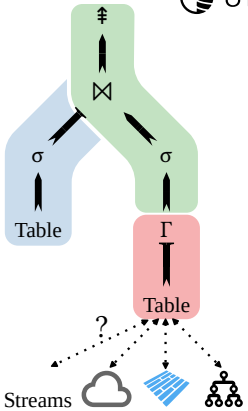
- Relations are entry point for all data in a DBMS
- Database systems utilize physical data independence
  - Masks remote data access
  - Independence of storage formats, e.g., Parquet
  - Optimized access paths for OLTP or OLAP
- New relation types transparent to other operators



Relation-Based Integration

UMBRA

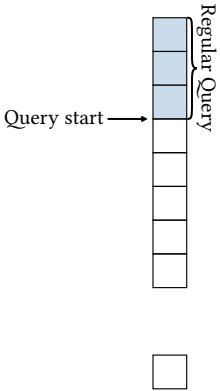
- Relations are entry point for all data in a DBMS
- Database systems utilize physical data independence
  - Masks remote data access
  - Independence of storage formats, e.g., Parquet
  - Optimized access paths for OLTP or OLAP
- New relation types transparent to other operators
- Streams are ephemeral in nature
- Scanned data is unknown at query start



Masking Unknown Scan Boundaries

UMBRA

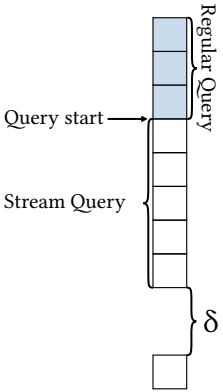
- Scans rely on fixed set of accessed data, i.e. TIDs



Masking Unknown Scan Boundaries

UMBRA

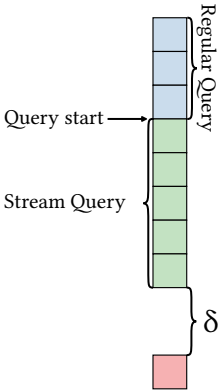
- Scans rely on fixed set of accessed data, i.e. TIDs
- ⚠ Stream data arrives during query processing



Masking Unknown Scan Boundaries

UMBRA

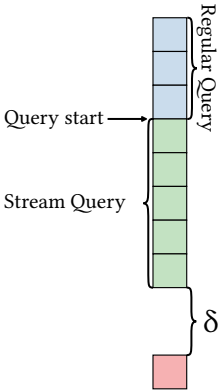
- Scans rely on fixed set of accessed data, i.e. TIDs
- ⚠ Stream data arrives during query processing
- Scan has to detect stream depletion
  - ⚠ Control messages would break SQL interface
  - ✔ Detect stream depletion from metadata



Masking Unknown Scan Boundaries

UMBRA

- Scans rely on fixed set of accessed data, i.e. TIDs
- Stream data arrives during query processing
- Scan has to detect stream depletion
  - Control messages would break SQL interface
  - Detect stream depletion from metadata

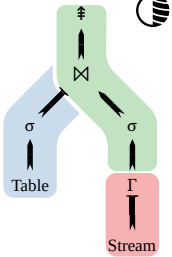


Queries with session-window semantics

Masking Data Ephemerality

UMBRA

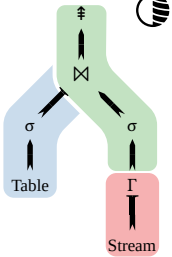
- Scans rely on data to be fully and durably materialized



Masking Data Ephemerality

UMBRA

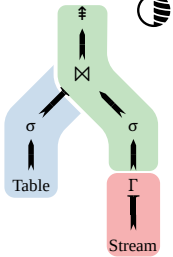
- Scans rely on data to be fully and durably materialized
- Streams are too voluminous be materialized fully

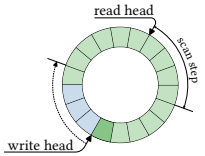


Masking Data Ephemerality

UMBRA

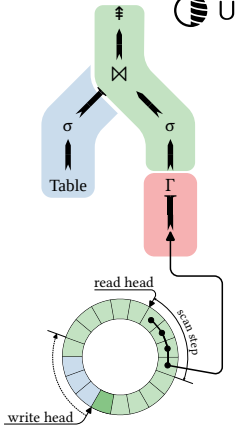
- Scans rely on data to be fully and durably materialized
- ⚠ Streams are too voluminous be materialized fully
- Scans have to deal with short-lived data
  - ✓ Ring buffer provides required abstraction
  - ⚠ Requires careful overflow checks for variable-sized data





Masking Data Ephemerality

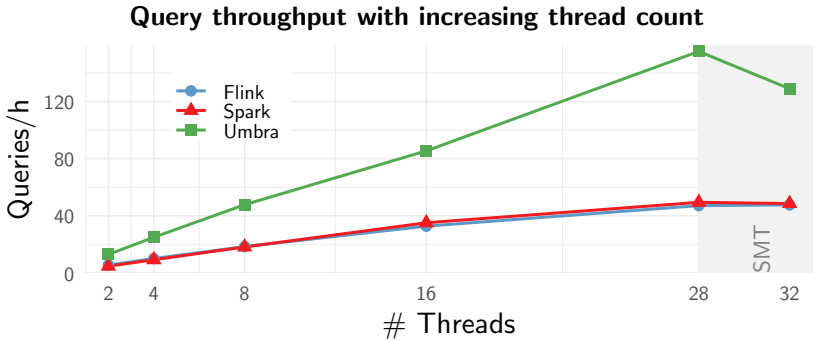
- Scans rely on data to be fully and durably materialized
- Streams are too voluminous be materialized fully
- Scans have to deal with short-lived data
  - Ring buffer provides required abstraction
  - Requires careful overflow checks for variable-sized data



Ring-buffered cache with overflow handling

Experimental Analysis - Analytical Performance

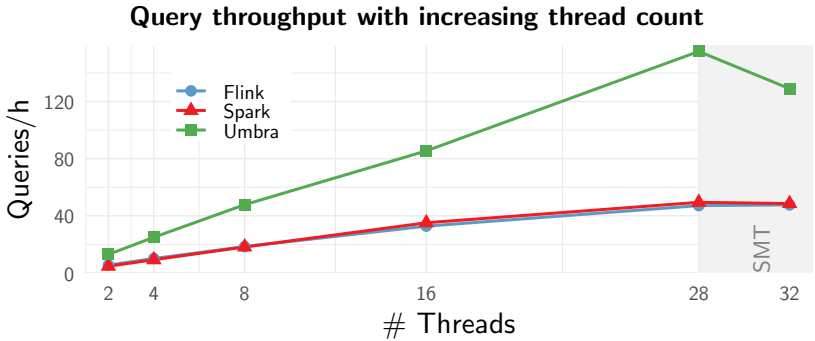
UMBRA



- TPC-H SF 100 where `lineitem` is treated as a stream

Experimental Analysis - Analytical Performance

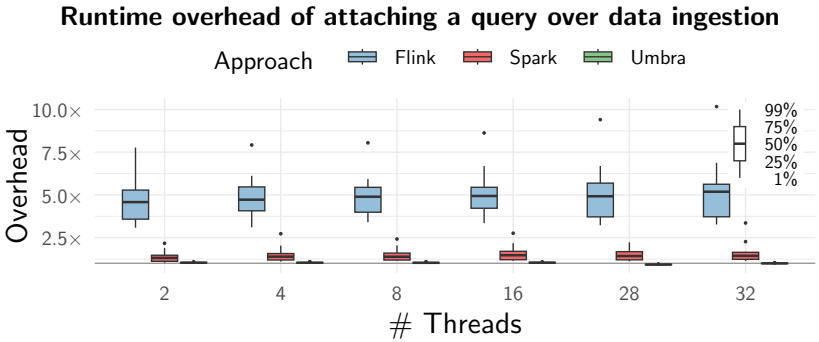
UMBRA



- TPC-H SF 100 where `lineitem` is treated as a stream
- Near-linear scaling to all physical cores
- **3**× speedup over established stream processing engines

Experimental Analysis - Query Overhead

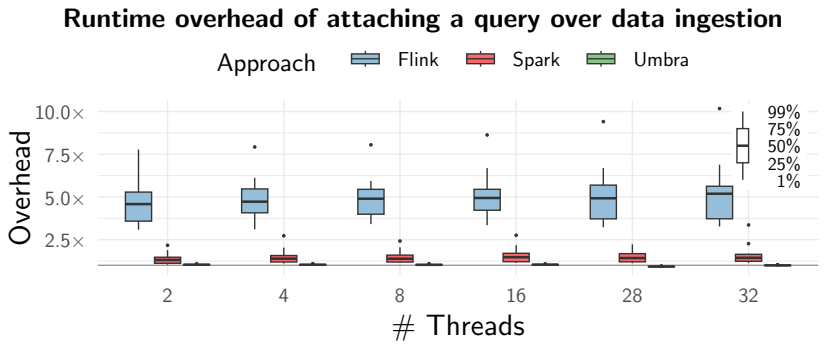
UMBRA



➔ Runtime difference of evaluating a query to parsing the data in each system

Experimental Analysis - Query Overhead

UMBRA

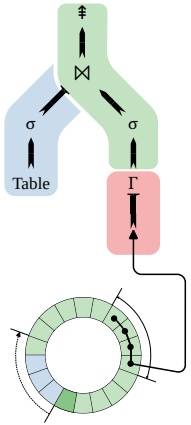


- Runtime difference of evaluating a query to parsing the data in each system
- ✓ Next to no overhead over insert processing
- ✓ Attaching multiple queries has no additional cost

Conclusion

UMBRA

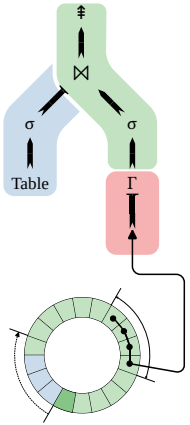
- Relation-based stream processing
  - Ring-buffered cache for ephemeral data
  - Session-windowed semantics masks unknown stream bounds
  - Minimally invasive, fully SQL-based integration
  - Full functionality for historic data



Conclusion

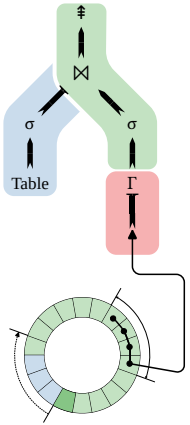
UMBRA

- Relation-based stream processing
  - Ring-buffered cache for ephemeral data
  - Session-windowed semantics masks unknown stream bounds
  - Minimally invasive, fully SQL-based integration
  - Full functionality for historic data
- Higher ease of integration than view-based solutions
- More performant than stream processing engines



Conclusion

- Relation-based stream processing
  - Ring-buffered cache for ephemeral data
  - Session-windowed semantics masks unknown stream bounds
  - Minimally invasive, fully SQL-based integration
  - Full functionality for historic data
- Higher ease of integration than view-based solutions
- More performant than stream processing engines



Thank you for your attention!