Continuous Analytics: Stream Query Processing in Practice

Michael J. Franklin
Professor, CS Dept., UC Berkeley
Founder and CTO, Truviso

LBNL
December 2, 2009
Stream Query Processing: The Basic Idea

**Traditional**
- Static Results
- Data Warehouse
- Batch Load Data
- Queries
- Results
- “Store First, Query Later”

**Stream Query Processing**
- Continuous, Visibility, Alerts
- Data Stream Live Data Processor
- Results
- Process “On-the-Fly”
A “Hot” Research Topic

Early: Active DBs, ECA rules, Triggers, Data Broadcast

Field took off in 2002

- ~10 SIGMOD/VLDB/ICDE “stream” papers thru 2001
- 275+ since then (holding steady at 40-50/yr)

Lots of Research Systems-Building Projects

- AT&T – Gigascope
- Berkeley – TelegraphCQ->HIFI
- Brandeis, Brown, MIT - Aurora->Borealis
- Purdue – Nile
- Stanford – STREAM
- Wisconsin, Portland State – NiagaraCQ

Lots of Technology Transfer:
Research Vision Meets Reality

Existing Technology

New Application

New Technology

Existing Technology

New Application

New Technology

Existing Application
But There is a Fundamental Problem

For data analytics, computers get slower every year.

Data volume (net-centric businesses)

Data volume (industry-wide)

Hardware capacity

http://www.b-eye-network.com/view/7188
It’s Getting Harder to Get Timely Information

Value of Data to Decision-Making

Time-critical Decisions

Information Half-Life In Decision-Making

Traditional Business Intelligence

Preventive/Predictive

Actionable

Reactive

Historical

Real-Time

Seconds

Minutes

Hours

Days

Months
Batch Processing: A Poor Fit for the Real-Time Web

The basic approach is the same for a legacy data warehouse cluster or a brand new Hadoop cluster.
Making the Database Elephant Dance

**Precomputation**
- Materialized Views
- Fancy Indexes
- Query Result Caching

**I/O Reduction**
- Main memory DB
- Column databases

**Shared Processing**
- Multi-Query Optimization
- Shared Scans

**Latency Reduction**
- Mini-batch, Trickle-feed

And of course: Throwing hardware at it! (i.e., MPP)
So what does Stream Query Processing have to do with the (batch-oriented) Data Overload problem?
The Key Insight: Decouple Processing and Delivery

All data starts as streams

On Demand/Interactive
- Instant ad-hoc Reports
- Visualization/Drill Down
- “Tivo” Replay

Decision Optimization
- Periodic Reporting
- Data Mining/OLAP
- Simulation/Back-Testing

“For Real-Time”
- Continuous Monitoring
- Automated Actions/Alerts
- Real-Time Dashboards

For Efficiency:
Process Streams in Real-Time.

But let Business Processes drive the use of the results.
Net-centric Analytics Workloads

- 80-90% of the workload is known ahead of time (i.e., not ad hoc)
  - Reports, KPIs, key metrics

- Applications are “additive”
  - Append-mostly
  - Time or order-oriented
  - Computation applied to new data

- Lots of Aggregation, often by time
- SQL is appreciated
Context Requires Stream/Relational

“Live” Data
- Network Traffic
- Message Buses
- Log Entries
- QoS Data

Staged Data
- Databases
- Documents/XML
- Files/Logs

Application Data
- Web Apps
- SOA/Web Services
- ERP/CRM/SCM
- Legacy Applications

Data Sources

Full SQL Interface

Algorithms + App Logic

PostgreSQL

Shared Stream Query Processor

Integrated Framework

Raw Data

Aggregates

Persistent Data Store

Can also directly link TruCQ instances

Immediate Insight
- TruView Web Dashboards
- On Demand Reports
- Transactional Drill Down

Output Streams
- Pre-Aggregates
- Message Buses
- External Data Storage

Actions
- Alerts/Events
- Algorithms
- Data Services
- Transactional Systems

Destinations
Goal: support the full SQL language and ecosystem

- A **stream** is an unbounded sequence of records
- A **table** is a set of records
- **Window operators** convert streams to tables
- **SQL queries** apply to tables

- Each “window” produces a set of records (a table)
- The TruCQ engine applies standard SQL to the results of window operators
- Results are continuously appended to the output stream
**Example: Window Operators**

**Chunking**  Non-overlapping windows (e.g. 5 second / record windows)

<SLICES ‘5 seconds’>

**Sliding**  Overlapping windows (e.g. 5 sec. windows move every 2 sec.)

<VISIBLE ‘5 seconds’ ADVANCE ‘2 seconds’>

**Landmark**  Growing windows (e.g., extend window every 2 seconds)

<LANDMARK RESET ‘1 day’ ADVANCE ‘2 seconds’>
Example: Partitioned Window
User Session Tracking

Partitioned Value-based “virtual” streams
<PER UserID VISIBLE “2 rows” ADVANCE “2 rows”>

Data Stream

User 1 Session

User 2 Session

User 3 Session
Creating and Querying Streams

```
CREATE STREAM url_stream
( url             varchar(1024),
  atime           timestamp  CQTIME USER,
  client_ip       varchar(50),
)
;

SELECT url, count(*) as url_count
FROM url_stream <VISIBLE '5 minutes'
  ADVANCE '1 minute'>
GROUP by url
ORDER by url_count desc
LIMIT    10;
```
CREATE STREAM urls_now AS
SELECT url, count(*) as scnt,
    cq_close(*)
FROM url_stream <VISIBLE '5 minutes' 
    ADVANCE '1 minute'>
GROUP by url;

CREATE TABLE urls_archive
(url varchar(1024),
    scnt integer,
    stime timestamp);

CREATE CHANNEL urls_channel
FROM urls_now
INTO urls_archive APPEND;

Stream/Table integration pushes Stream Processing beyond “real-time” applications

Derived stream (Can also define Views over streams)
SELECT c.scnt, h.scnt, c.stime
FROM (select sum(cnt) as scnt,
        cq_close(*) as stime
        from urls_now <slices 1 windows>) c,
      urls_archive h
WHERE c.stime - '1 week'::interval = h.stime

Added Benefit – you get to re-use
nice features like subqueries,
sophisticated date/time functions,
fancy query operators, etc.
Shared CQ Query Processing

- Start with a regular optimized plan
- New queries “folded” into the current shared plan
  [Krishnamurthy et al SIGMOD 06]
- Adaptive query plan – Tuple Router
  (i.e., “CQ-eddy” [Madden et al SIGMOD 02])
- Eliminates redundant work

```
SELECT   T.symbol, AVG(T.price*T.volume)
FROM     Trades T, SANDP500 S
WHERE    T.symbol = S.symbol AND T.volume > 5000
GROUP BY T.symbol
```
Unique User Tracking

Query:
How many unique users clicked this ad in the past 5 minutes who were males, 18-24
Meeting Data Volume and Latency Demands

Major ad network’s 40+ server data warehouse system

Continuous Analytics software on a single commodity server
The Latency/Performance Tradeoff: Gone

With Batch: reducing batch size reduces throughput.
With Continuous Analytics this tradeoff goes away.
Recall the tools used to make store-first systems perform in this environment:

1. I/O Reduction
2. Shared Processing
3. Latency Reduction
4. Precomputation
5. Throwing Hardware at it

Stream Query Processing inherently does #1, 2 and 3

An integrated Stream-Relational system enables #4

#5 still possible but much less necessary
The Future of Analytics

• “Barbarians at the Gate”
  • Procedural cloud-based approaches gaining interest
  • Proven scalability for massive data sets
  • But, we’ve seen this movie before!

• What is the minimal extension to SQL to support Continuous Analytics?
• Can we make persistence an orthogonal property of both queries and data?
Complex Event Processing (CEP)

- Related Technology
  - Distributed Simulation Background
  - Some overlap with Stream Query Proc.
- Oracle (BEA), TIBCO, and others have products
- Focused on real-time reaction
- Proprietary Languages (some SQL-ish)
- Currently suffering from the dual innovation dilemma discussed earlier.
Conclusions

• **Stream Query Processing enables “Continuous Analytics”**
  • Addresses a 30+ yr old legacy problem in Database Systems.
  • Crucial for coping with the ongoing Data Tsunami.

• **Key Insight: Decouple Processing from Delivery**
  • Process in real-time because it is orders of magnitude more efficient.

• Deliver results as dictated by users and business processes.

• Leveraging existing standards, tools, and business processes is key to deploying new technology.