Massive Scale-out of Expensive Continuous Queries

Erik Zeitler and Tore Risch
Department of Information Technology
Uppsala University

Presented by Haikal Pribadi
Massive Scale-out of Expensive Continuous Queries

Motivation and Contribution

Split stream functions

Stream processes in distributed environment

Evaluation

Related and future work
Motivation and Contribution
Motivation

Real time decision making

- Scientific computing, engineering, network traffic, phone conversations, ATM transactions, web searches, and sensor data

Queries of massive data streams

Expensive computation

Requires splitting stream into parallel substreams
Problem with stream splitting

Becomes a bottleneck for inputs streams of

- High volume
- Complex parallelization condition
- Massive parallelization of query operators
Contribution

Parasplit

A splitstream function
Eliminates bottleneck of stream splitting
- Parallelize stream splitting operator
Achieves max rate of network bound
Concept: data parallel stream processing

Input stream $S$

Split into $q$ parallel stream

Query operator $Q$ are executed on substreams

Substreams map to parallel CPUs: $PQ_j, j = 1..q$

Tuples in a split stream may be:
- partitioned
- replicated
Stream Functions
Splitstream function basic signature

\[ \text{Splitstream}(\text{stream } s, \text{ integer } q, \text{ function } rfn, \text{ funcion } bfn) \rightarrow \text{ vector of stream } sv \]

s: input stream
q: output split
rfn: routing function
bfn: broadcast function
Problem: routing and broadcast functions become bottlenecks on high volume streams
Parasplit function

\[
\text{Parasplit}(\text{stream } s, \text{ integer } q, \text{ function } rfn, \text{ funcion } bfn) \rightarrow \text{ vector of stream } sv
\]

Eliminates the bottleneck by scaling out execution of \( rfn \) and \( bfn \) in addition to \( Q \)

Dynamically creates distributed execution plan for stream processes
Parasplit function

Window router, $PR$: randomly splits stream windows into $p$ parallel substream
- Random routing eliminates delay
- Window size can be configured with high volume

Window splitter, $PS$: splits substreams according to split functions ($rfn$ and $bfn$)

Query processor $PQ_j$: merges all received substreams into a local stream where query operator $Q$ will be executed. Order of tuples are maintained through their timestamp.
Stream processes for distributed environment
Operators in a stream process

*Merge* several streams into one

*Compute* a continuous sub-plan over stream

*Split* stream into modules being partitioned or replicated
Cost model for Stream Processes

\[ C = cr + (cp + cm) \cdot u + cq + \sigma(cs(o + r + q \cdot b) + ce(r + q \cdot b)) \]

- \( cr \): reading an input tuple
- \( cp \): polling input streams
- \( cm \): merging input streams
- \( u \): number of input streams
- \( cq \): computation cost on merged stream
- \( \sigma \): selectivity of sub-plan
- \( cs \): splitting modules per tuple
- \( ce \): emitting a tuple to an output stream
Cost model for Stream processes

Window router (PR)

\[ C_{PR} = cr_w + cs_w + ce_w \]

Window Splitter

\[ C_{PS} = cr_w + cs(o + r + q.b) + ce(r + q.b) \]

Query Processor

\[ C_{PQ} = cr + p.(cp + cm)+O \]
Heuristic for automatic parallelization

\[ \Phi_{PR} : \text{max stream rate for PR} \]

\[ \Phi_{PS} : \text{max stream rate for PS (nb: parallelized)} \]

\[ \Phi_{PQ} : \text{max stream rate for PQ (nb: parallelized)} \]

\[ \Phi_{\text{PARASPLIT}} = \min(\Phi_{PR}, \Phi_{PS}, \Phi_{PQ}) \]
Heuristic for automatic parallelization

Window router
- Large window size, less communication
- Determine window size
- Profile the cluster with different window sizes

Window splitter
- Parallelization \times SP \text{ rate} \geq Desired rate
- Consider \( C_{PS} \) and \( C_{PQ} \) to calculate optimal parallelization over cost
Evaluation
Achieving network bound

Max stream rate [Mbps]

W [kB]
Scale-up comparison – wrt MaxTree

Max stream rate [Mbps] vs. $q$

- **Parasplit PR Tree**
- **Parasplit, single PR**
- **Cost model, single PR**
- **$p=q$, single PR**
- **$p=1$**
- **MaxTree**
Parasplit Efficiency

- Parasplit PR tree
- Parasplit, single PR
- Cost model, single PR
- p=q, single PR
- p=1

Graph showing efficiency vs. q.
## Linear Road Benchmark

### Table 1. LRB implementations.

<table>
<thead>
<tr>
<th>Name</th>
<th>year</th>
<th>L</th>
<th>#cores</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aurora [3]</td>
<td>2004</td>
<td>2.5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SPC [19]</td>
<td>2006</td>
<td>2.5</td>
<td>170</td>
<td>3GHz Xeon</td>
</tr>
<tr>
<td>XQuery [6]</td>
<td>2007</td>
<td>1.5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>scsq-lr [26]</td>
<td>2007</td>
<td>1.5</td>
<td>1</td>
<td>laptop</td>
</tr>
<tr>
<td>DataCell [22]</td>
<td>2009</td>
<td>1</td>
<td>4</td>
<td>1.4s average response time</td>
</tr>
<tr>
<td>stream schema [13]</td>
<td>2010</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>scsq-plr [32]</td>
<td>2010</td>
<td>64</td>
<td>48</td>
<td>maxtree</td>
</tr>
<tr>
<td>CaaaS [9]</td>
<td>2011</td>
<td>1</td>
<td>2</td>
<td>Streaming MapReduce</td>
</tr>
<tr>
<td>scsq-plr</td>
<td>2011</td>
<td>512</td>
<td>560</td>
<td>Parasplit. D disabled</td>
</tr>
</tbody>
</table>
Future work
Future Work

Scaling out parallel database
- Combine high volume of idle data

Adaptive parallelization

Scheduling of execution over streams
Comments and Criticisms

Cost model equation coefficients should be well defined and explained.

Optimization of cost model equation is a vital topic of discussion, more detail on the matter would be useful.
Thank you
Haikal Pribadi
hp356@cam.ac.uk