# Massive Scale-out of Expensive Continuous Queries

Erik Zeitler and Tore Risch

Presented by Brett Lagerwall

University of Cambridge

February 17, 2013

## Motivation

There are many real-world applications for processing continuous queries over streaming data.

Input steams need to be split into parallel sub-streams.

Stream splitting becomes a bottleneck if poorly implemented.

## Overview

Parasplit intends to reduce the bottleneck. In summary:

- Is a stream splitting algorithm
- Maximum stream rate bound by network
- Energy efficient

## Methodology and Contribution – Splitstream

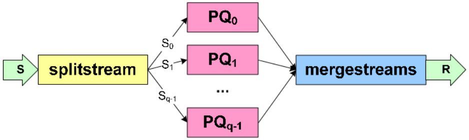


Figure: Source: Zeitler and Risch (2011)

A splitstream function splits a stream $S$ into $q$ parallel streams. Must answer these questions:

- For any given tuple, which substream must it be sent to?
- Does a large value of $q$ affect the cost of the split?

## Methodology and Contribution – Parasplit

Problem with split stream: functions rfn and bfn have non-zero cost. This causes a bottleneck.

A new split function called parasplit is defined:

- Scales out the execution of rfn and bfn.
- Has a window router PR which reads in entire physical windows.
- Has window splitters which unpack tuples and send them to query processors.

## Methodology and Contribution – Parasplit Diagram

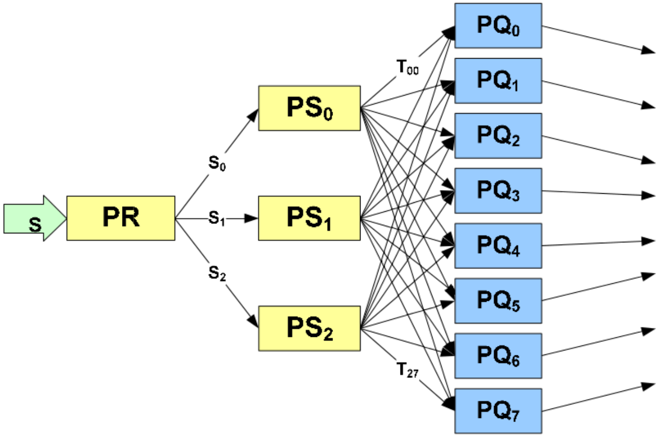


Figure: Source: Zeitler and Risch (2011)

## Methodology and Contribution – Cost

Formulas are derived for the cost of execution at each of the window router and window splitter stages.

They then need to show that the maximum stream rate at each of them is not a bottleneck.

- Window router – Large windows make the cost of execution negligible compared to the cost of communication.
- Window splitter – A formula is derived to show the number of window splitters required so that this is not a bottleneck.

## Results – Window Router Stream Rate

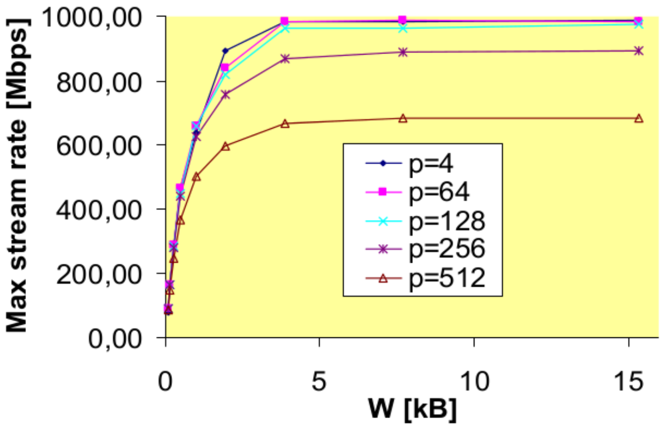


Figure: Source: Zeitler and Risch (2011)

## Results – Window Router Stream Rate

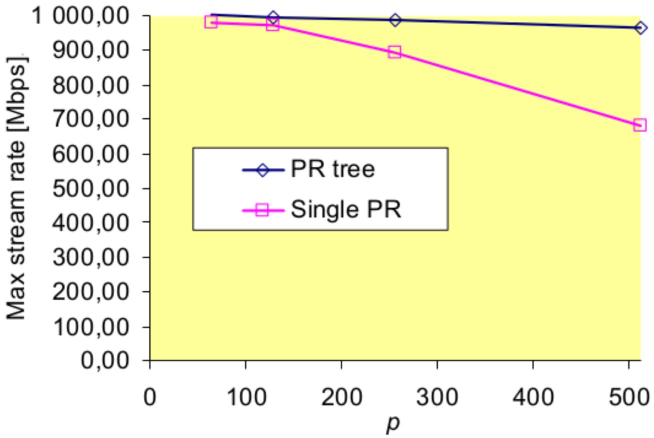


Figure: Source: Zeitler and Risch (2011)

## Results – Parasplit Scale-up

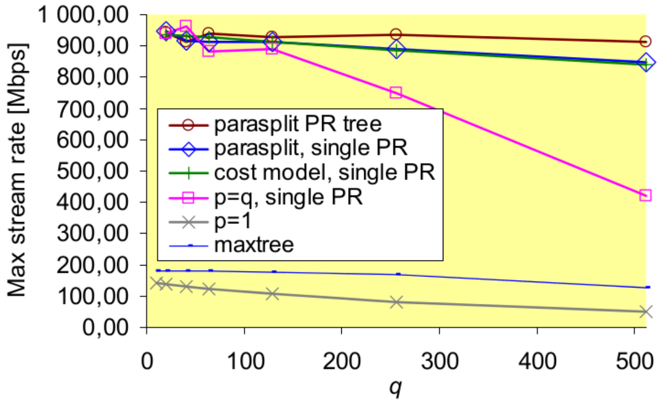


Figure: Source: Zeitler and Risch (2011)

## Results – Parasplit Efficiency

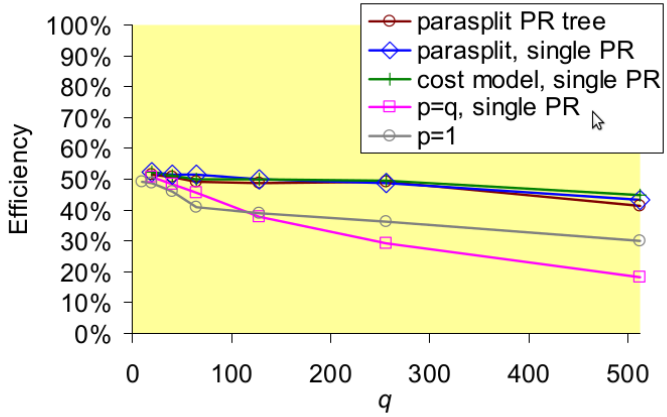
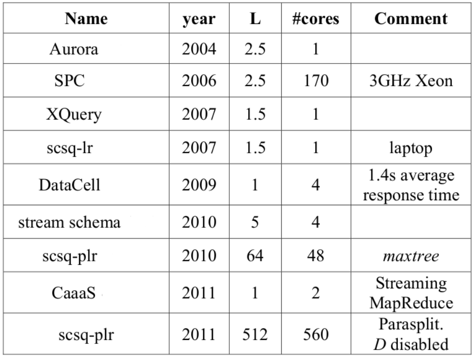


Figure: Source: Zeitler and Risch (2011)

## Results – LRB Experiment

The Linear Road Benchmark (LRB) is used to evaluate Stream Data Management Systems.

The goal is to see the number of expressways which a system can handle.



| Name | year | L | #cores | Comment |
|---|---|---|---|---|
| Aurora | 2004 | 2.5 | 1 | |
| SPC | 2006 | 2.5 | 170 | 3GHz Xeon |
| XQuery | 2007 | 1.5 | 1 | |
| scsq-lr | 2007 | 1.5 | 1 | laptop |
| DataCell | 2009 | 1 | 4 | 1.4s average response time |
| stream schema | 2010 | 5 | 4 | |
| scsq-plr | 2010 | 64 | 48 | *maxtree* |
| CaaaS | 2011 | 1 | 2 | Streaming MapReduce |
| scsq-plr | 2011 | 512 | 560 | Parasplit. *D* disabled |

Figure: Source: Zeitler and Risch (2011)

## Related Work

- Zeitler and Risch introduced a splitstream function called autosplit. This function was able to achieve an L-rating of 64.
- An earlier instance of stream splitting could be seen with SPADE.
- Gigascope provides basic stream operators, with the system later being upgraded by Johnson et al.

## Related Work

- SPC makes use of separate processing elements – each of which processes a different stream operator.
- Stream schema uses metadata to automatically parallelize stream queries.
- CaaaS is another example of a service which uses metadata to stage the data – rather than physically moving it.

## Future Work

Areas which can be addressed in the future:

- Investigate parasplit with higher network speeds and more cores.
- Determine whether hardware acceleration can improve performance.
- Adaptive parameter selection with regards to parallelization.

## Criticism

- In the evaluation, why were they testing their system with more cores than the others? Is this fair to the other systems?
- It did not appear as if they re-tested other systems and implementations. With newer network interfaces and faster cores might the others do better?
- There was no explanation of how to route data through multiple streams, but not broadcast it to all compute nodes.
- In the first test, they cannot answer why their performance degrades with more window splitters.

## Criticism

- What is a Pr tree and why are they testing it without explaining it?
- Why does the Pr tree generally tend to get better results than a single window router?
- If a tree of window routers worked better in the first 3 experiments, why did they just use a single one in the LRB benchmark? What are they not telling us?

## Conclusion

To conclude:

- There is a need.
- They appear to have a good solution.
- There are anomalies and unexplained events in the testing.