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

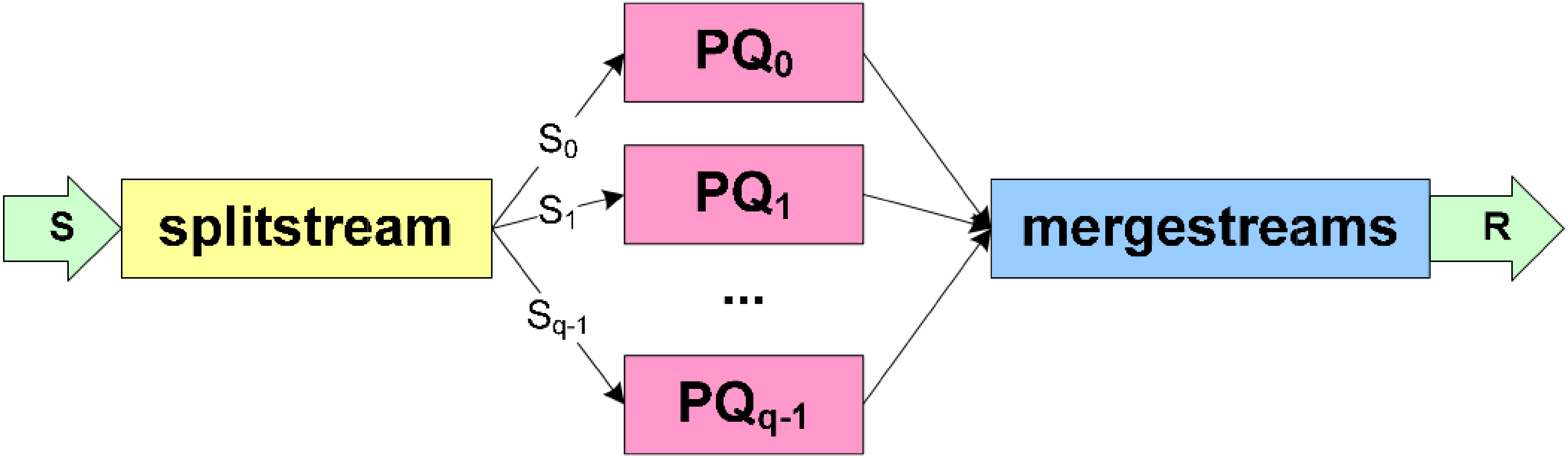
Splitstream(stream s, integer q, function rfn, funcion bfn) → vector of stream sv

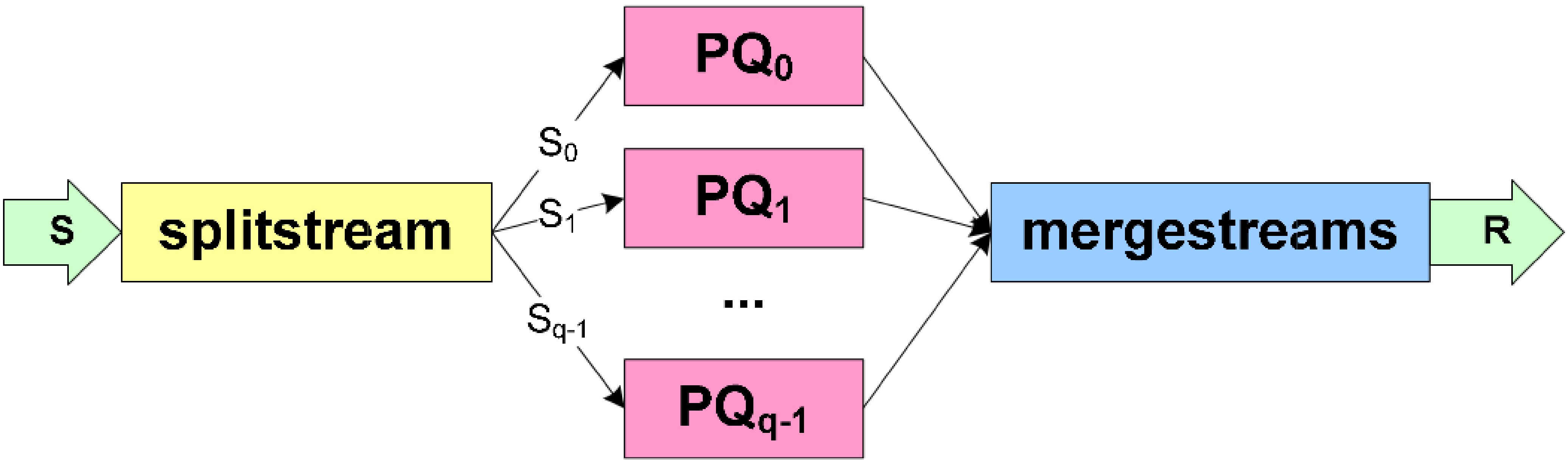
s: input stream

q: output split

rfn: routing function

bfn: broadcast function


S
splitstream
S0
S1
Sq-1
PQ0
PQ1
...
PQq-1
mergestreams
R



Problem: routing and broadcast functions become bottlenecks on high volume streams

Parasplit function

Parasplit(stream s, integer q, function rfn, funcion bfn) → vector of stream sv

Eliminates the the bottleneck by scaling out execution of *rfn* and *bfn* in addition to *Q*

Dynamically creates distributed execution plan for stream processes

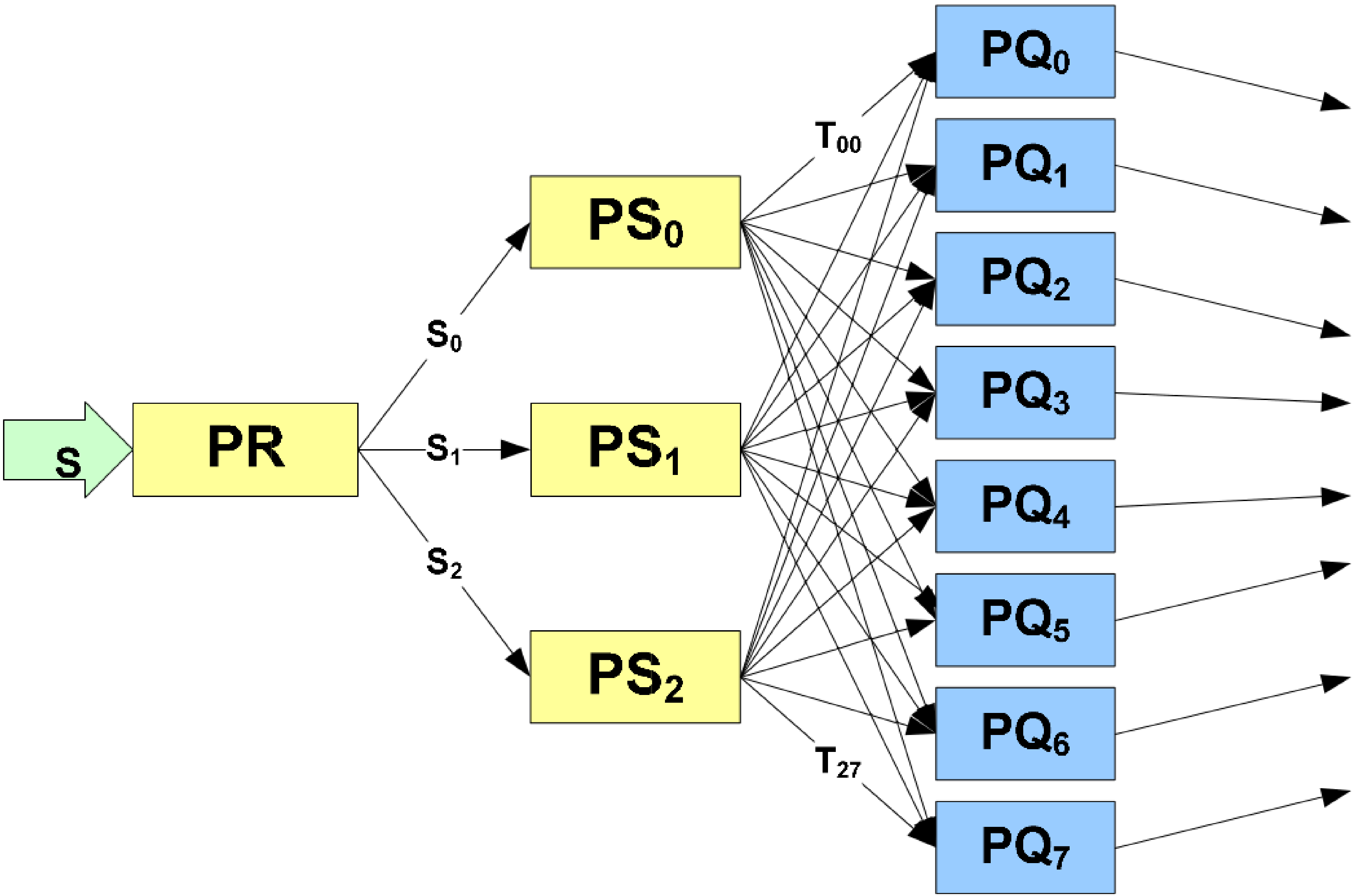
Parasplit function

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 *subtreams* 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.



S
PR
S0
S1
S2
PS0
PS1
PS2
T00
T27
PQ0
PQ1
PQ2
PQ3
PQ4
PQ5
PQ6
PQ7

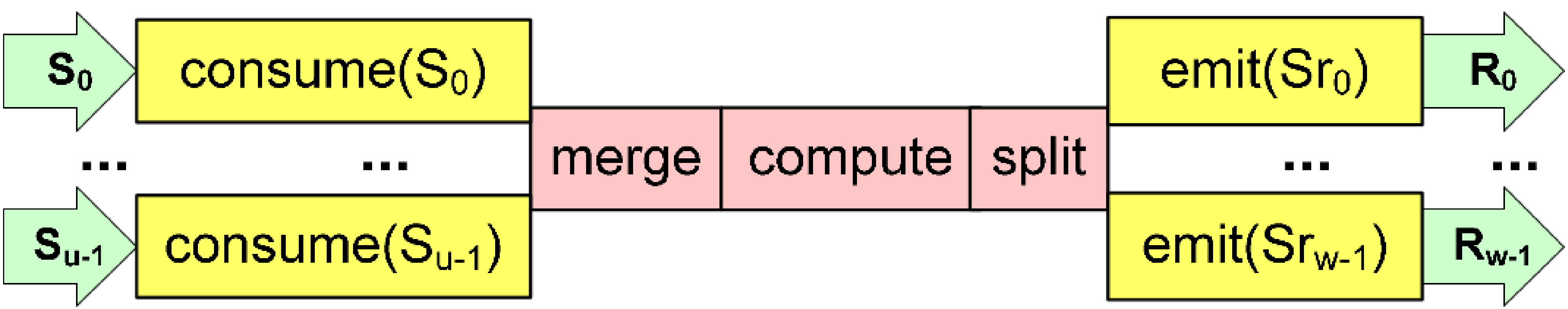
Stream processes for distributed environment

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

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

cr: reading an input tuple

cp: polling input streams

cm: merging input streams

u: number of input streams

cq: computation cost on merged stream

σ: 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

Φ_{PR} :max stream rate for PR

Φ_{PS} :max stream rate for PS (nb: parallelized)

Φ_{PQ} :max stream rate for PQ (nb: parallelized)

$\Phi_{PARASPLIT} = \min(\Phi_{PR}, \Phi_{PS}, \Phi_{PQ})$

Heuristic for automatic parallelization

Window router

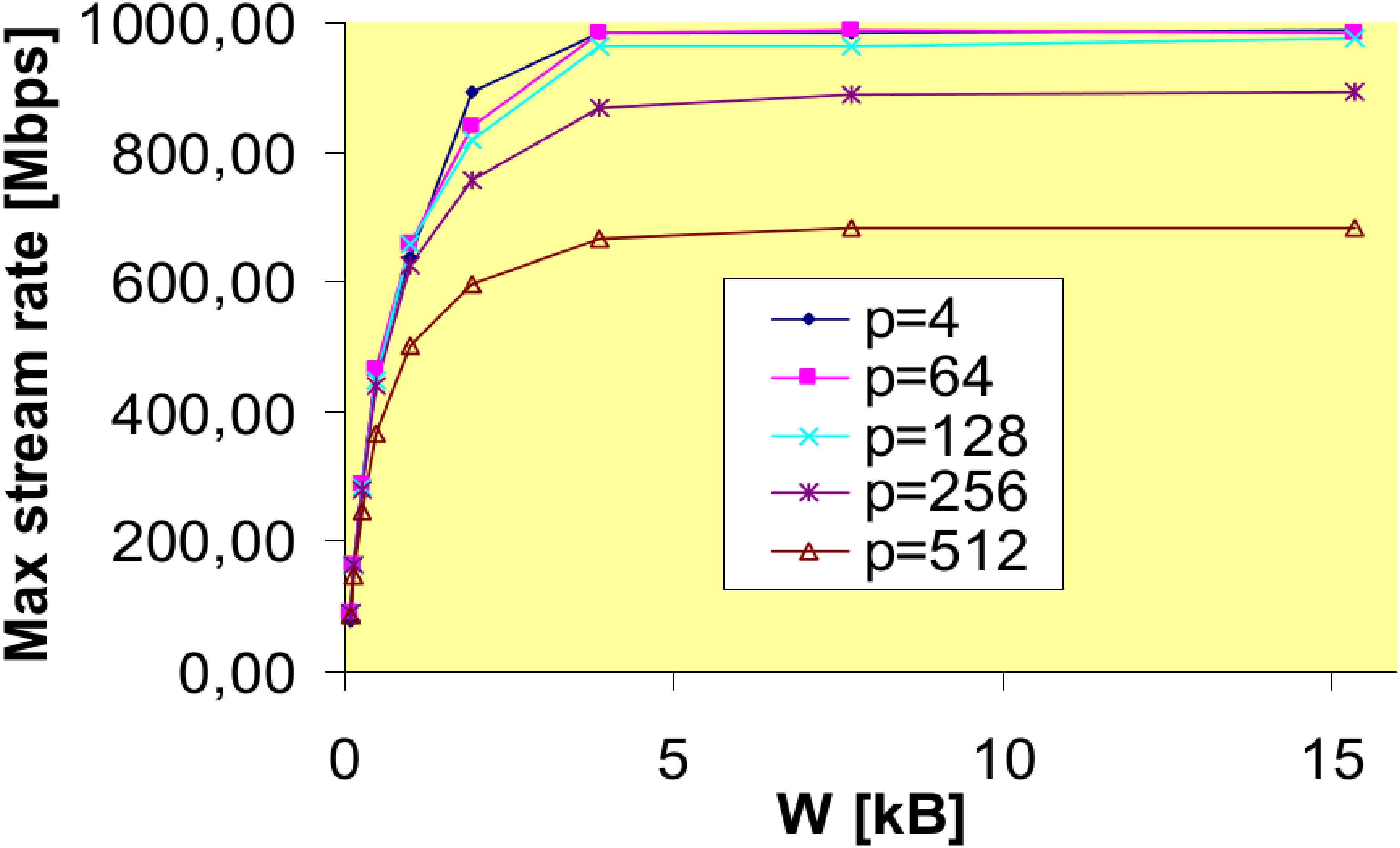
- Large window size, less communication
- Determine window size
- Profile the cluster with different window sizes

Window splitter

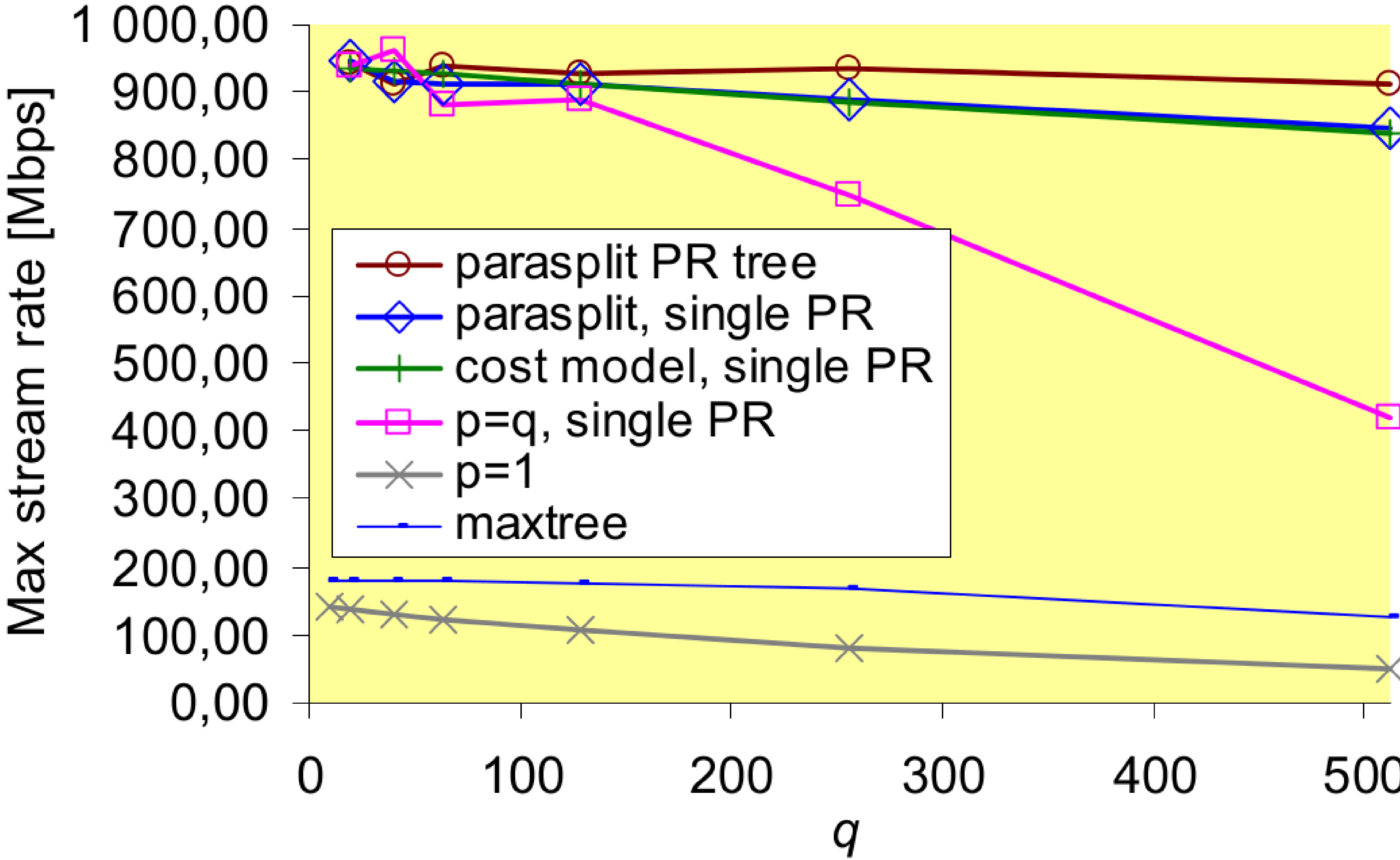
- Parallelization x SP rate >= Desired rate
- Consider C_{PS} and C_{PQ} to calculate optimal parallelization over cost

Evaluation

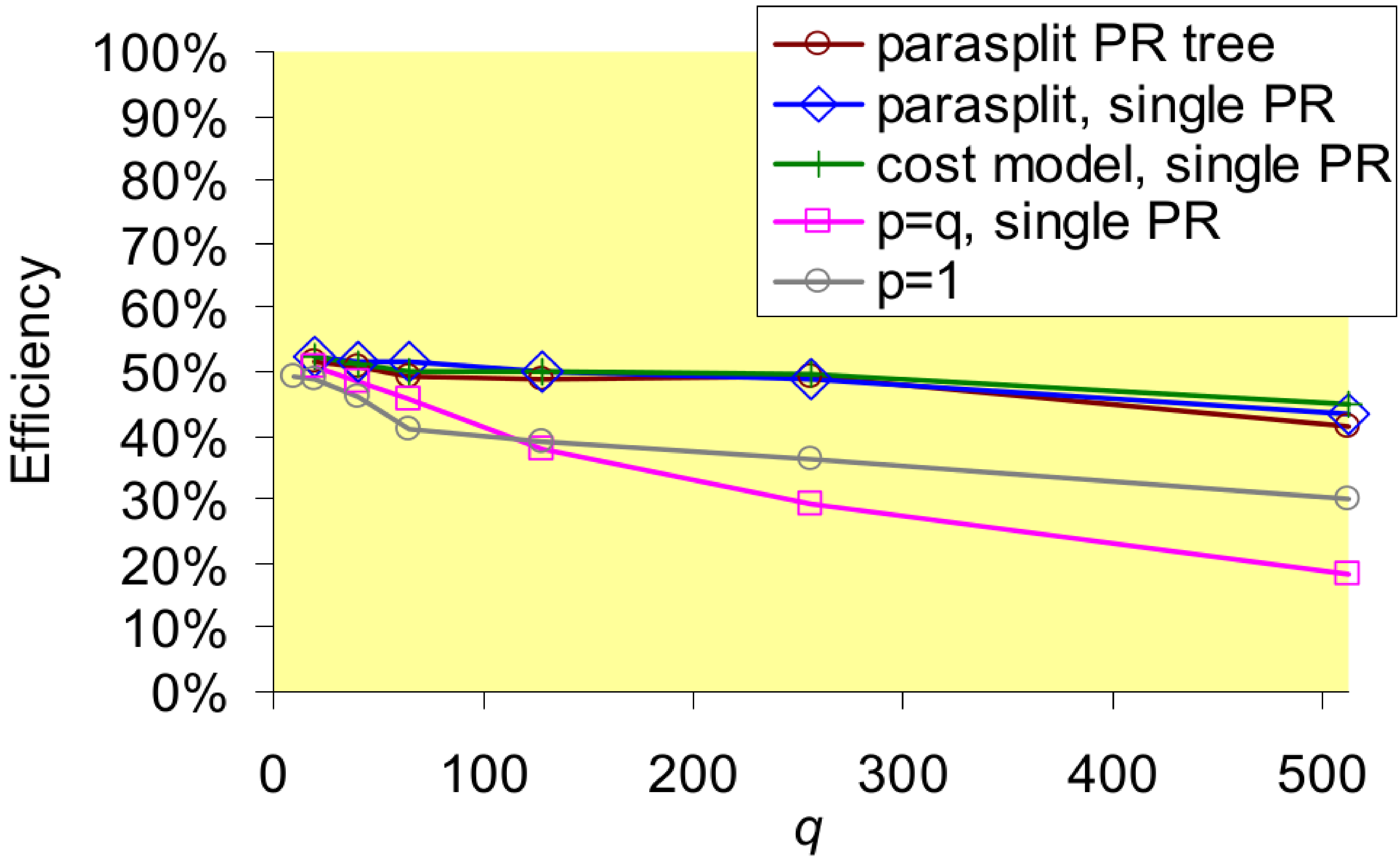
Achieving network bound



Scale-up comparison – wrt MaxTree



Parasplit Efficiency



Linear Road Benchmark

Table 1. LRB implementations.

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

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