Massive Scale-out of Expensive Continuous Queries

E. Zeiter, T. Risch, VLDB, 2011.

Review by Mariana Marasoiu for R212

Context

Data in stream form

- radio telescopes, sensor networks
- financial analysis

Continuous Queries

High data volume

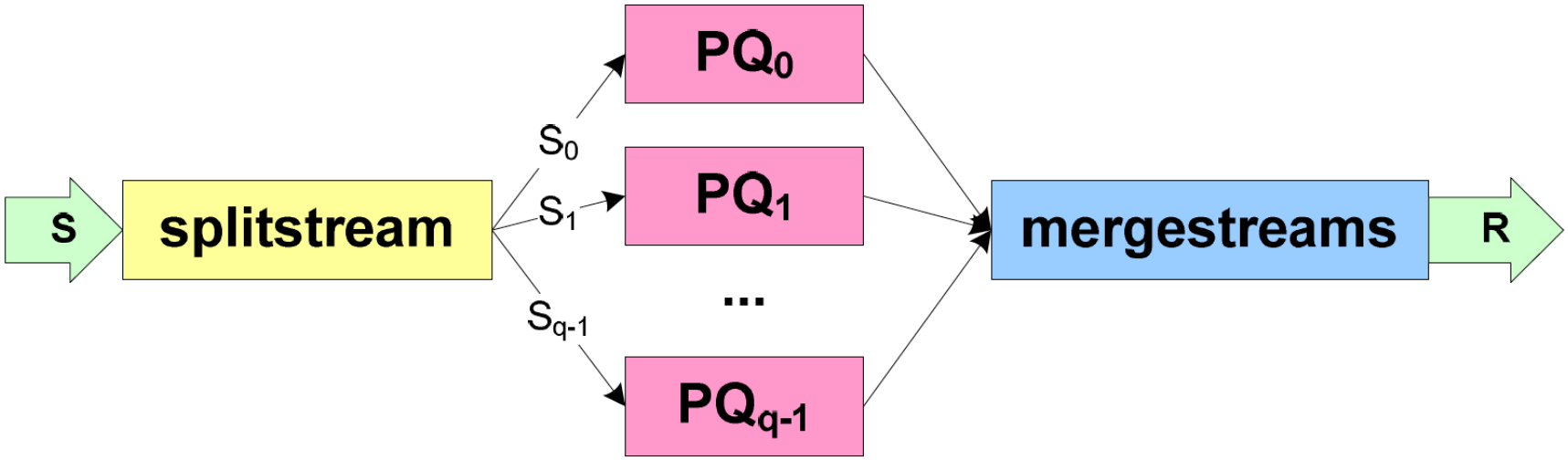
+

expensive computations

} Scalable stream processing

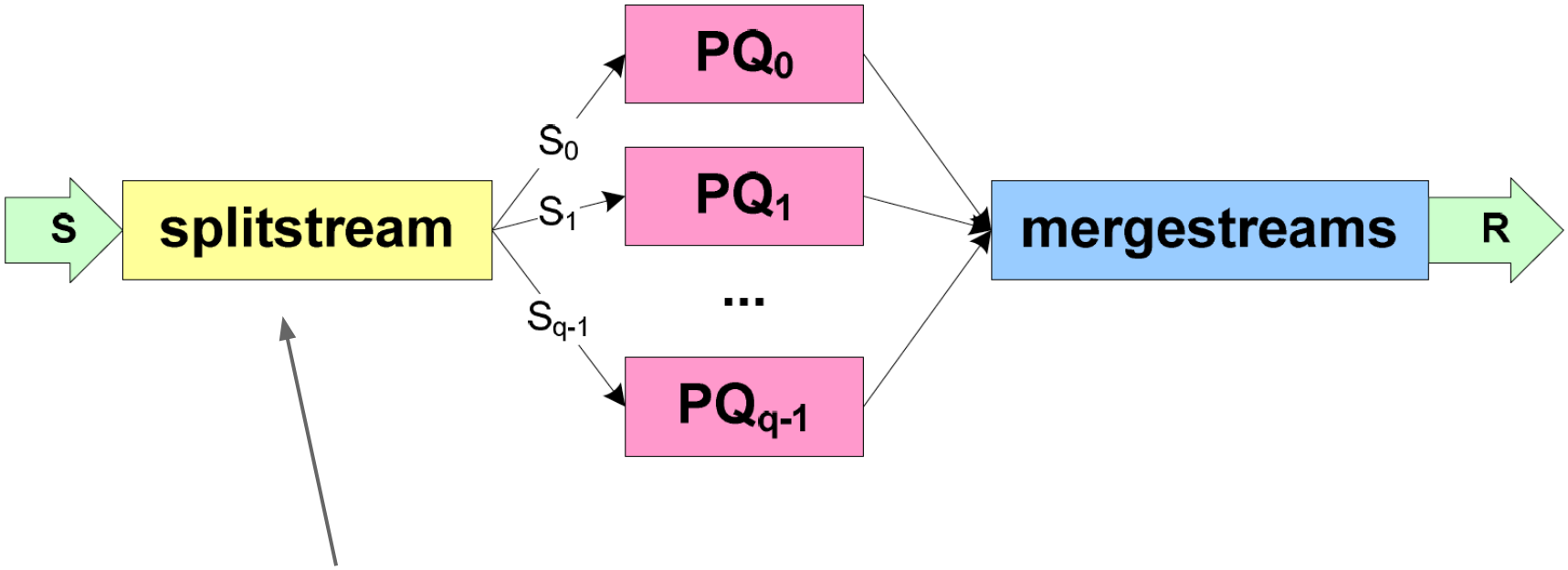
Streamed data parallelism

Parallelise query operators



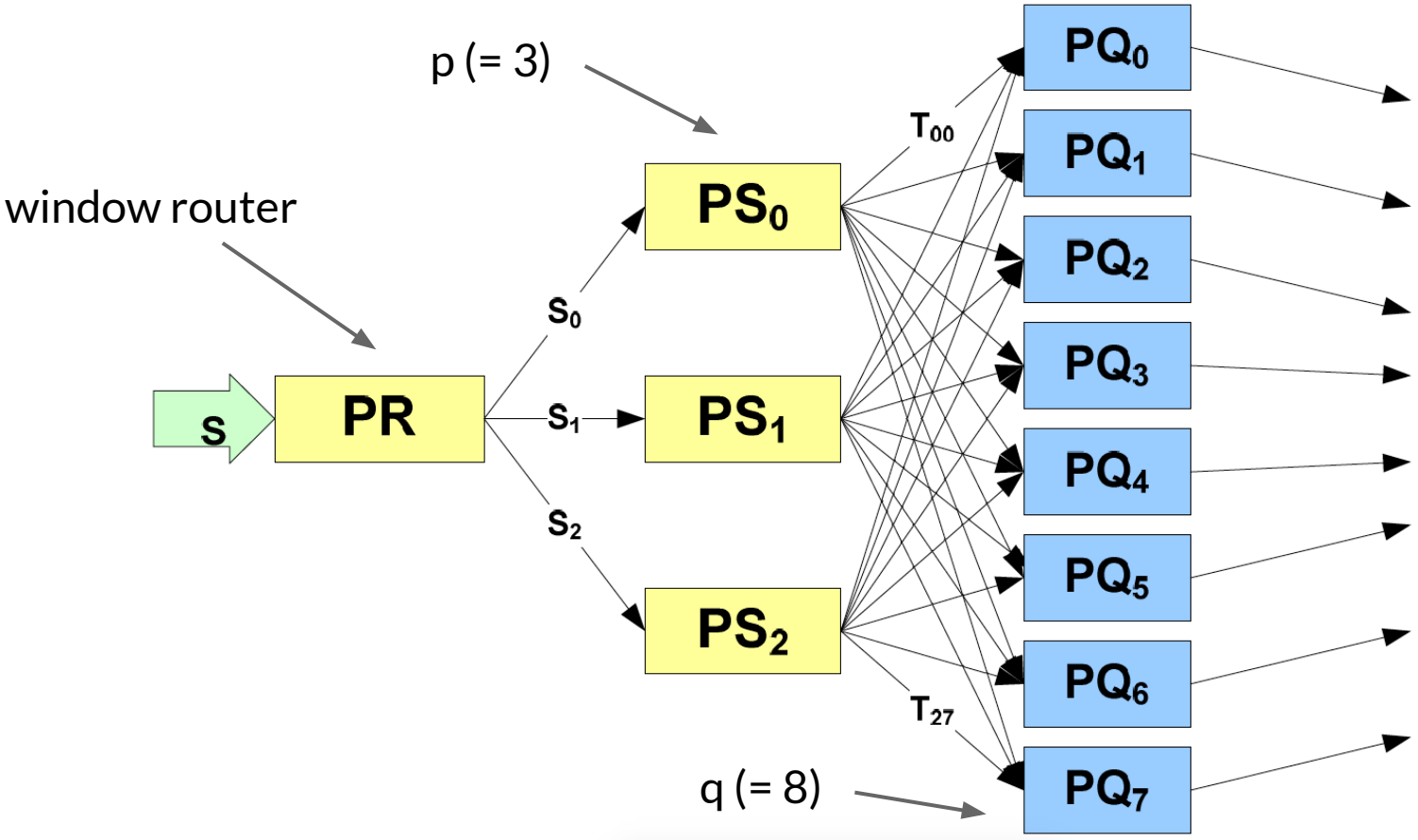
Streamed data parallelism

Parallelise query operators



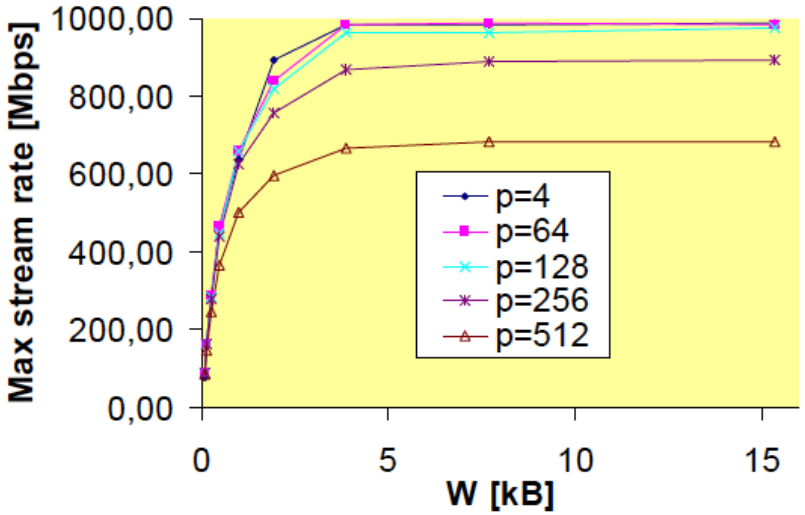
Solution? Parasplit.

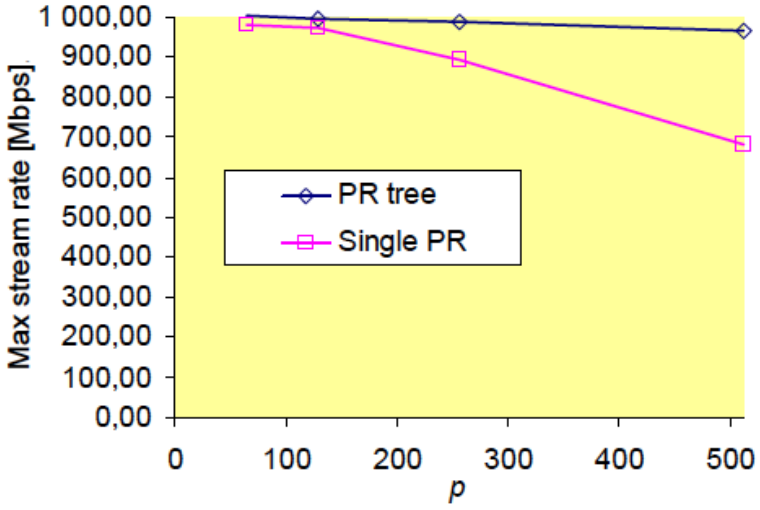
Parallelise query operators + stream splitting



Results

Window router stream rate

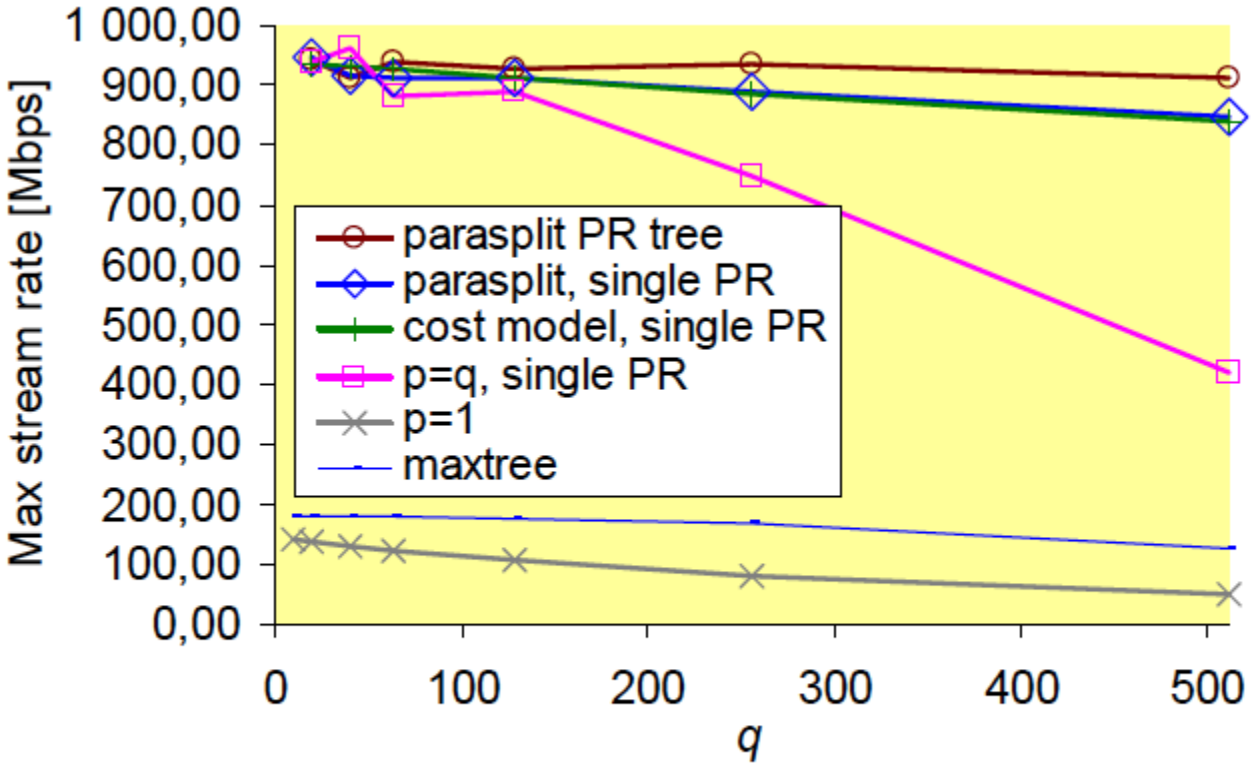




Performance degrades for p > 128,

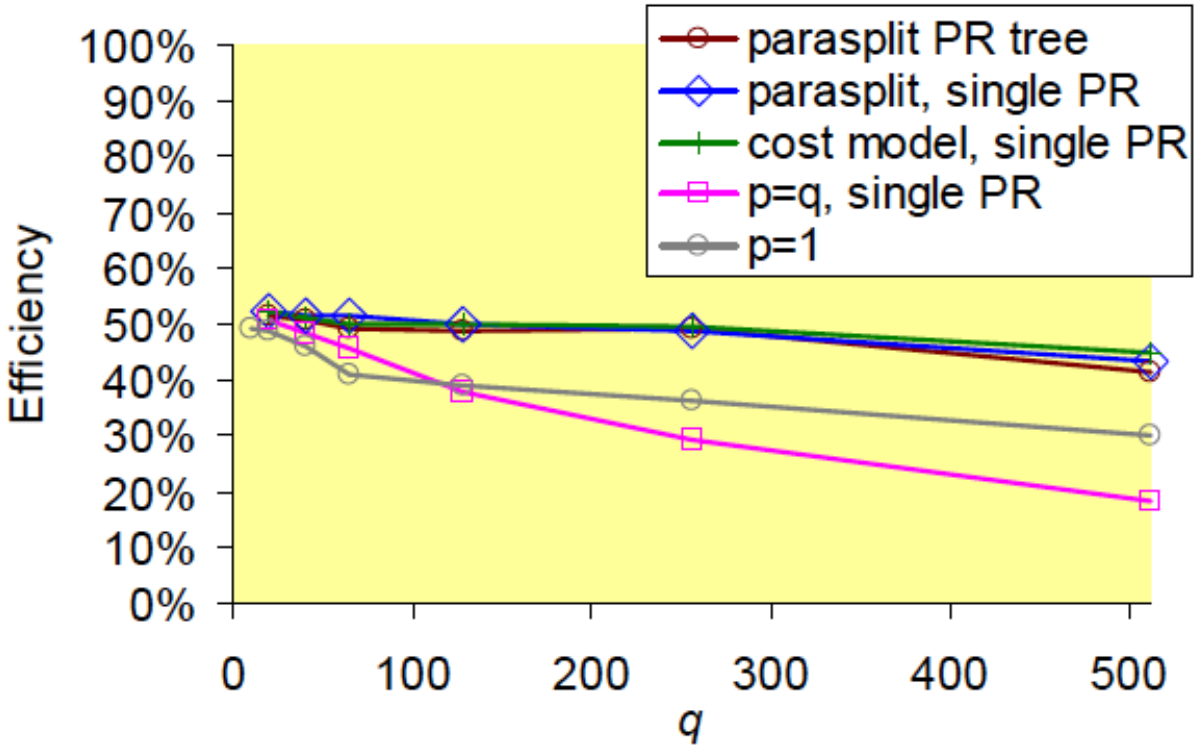
but with window router (PR) as tree, decrease is negligible

Parasplit scale-up



Parasplit efficiency

Measured as CPU overhead



Comparisons with 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

Higher L-rating is better

Strengths and weaknesses

Straightforward approach
Parallel splitting + parallel computing
Network bound efficiency

But…
Unclear how *p* is chosen based on cost/heuristic
Why does the performance degrade with high *p*?