

SQPR: Stream Query Planning with Reuse

Evangelia Kalyvianaki* Wolfram Wiesemann* Quang Hieu Vu+ Daniel Kuhn* Peter Pietzuch*

* Imperial College London, Dept. of Computing; + ETISALAT BT Innovation Center;

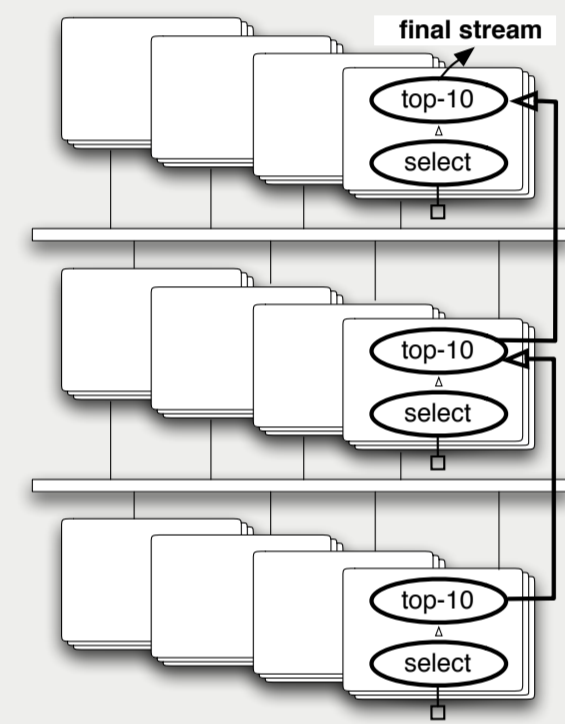
Data Stream Processing

Data streaming queries involve the processing of data from *sources* in near *real-time*. Stream processing appears in many application domains:

- ▶ environmental sensing
- ▶ healthcare systems
- ▶ financial trading
- ▶ network monitoring

Query Planning

Distributed Stream Processing System (DSPS)



Example Query:
Which are the top-10 locations, in the city center, with the highest concentration of CO?

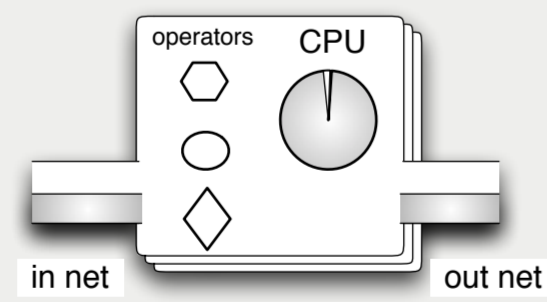
- Query plan involves:**
- ▶ operator placement
 - ▶ resource allocation: CPU, net
 - ▶ stream connections

Challenges in Query Planning

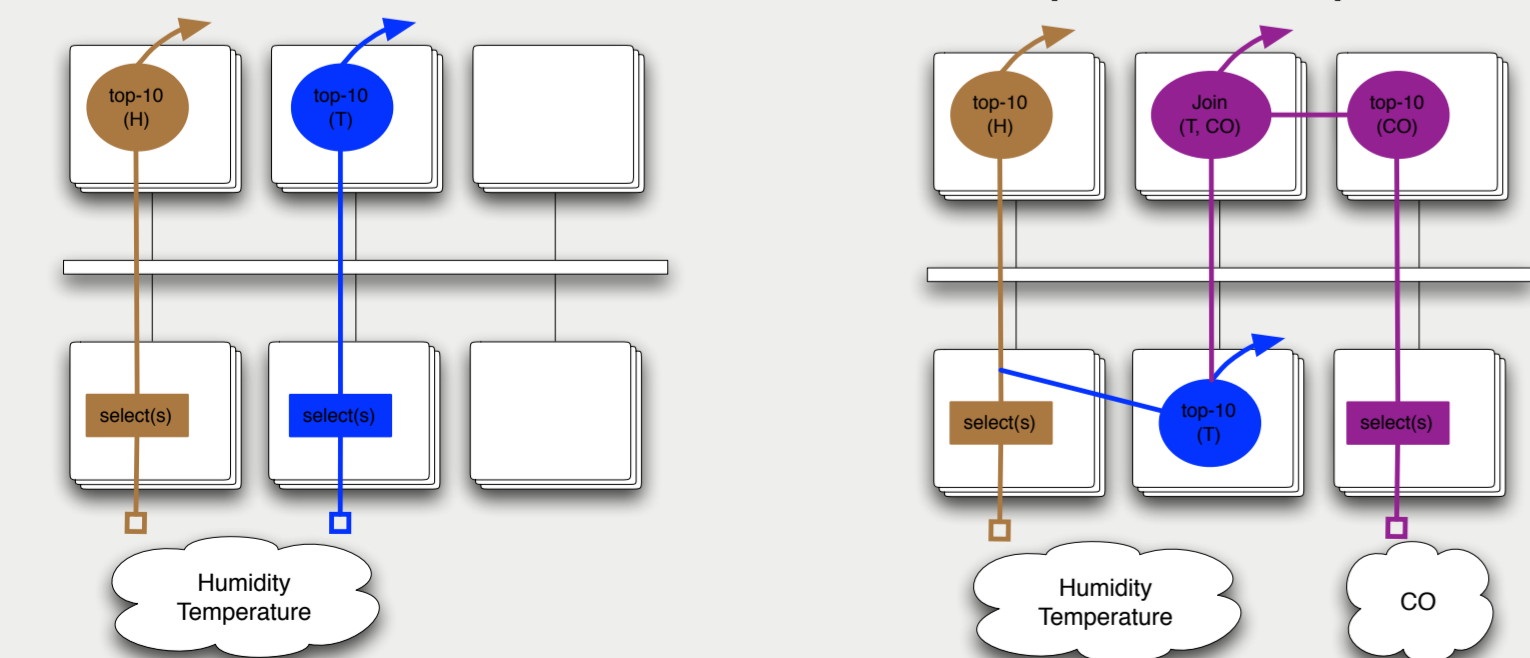
Previous approaches assume vastly provisioned DSPSs, or use heuristics. But, today, large volumes of queries soon exhaust DSPSs resources.

Pitfalls:

1. single resource exhaustion



2. waste of resources because of operator repetition



operator overlap

stream reuse

SQPR Overview

SQPR treats query admission, **multi-resource** operator allocation and **reuse** as a single optimisation model.

Optimisation Problem

maximise:

$$\lambda_1 * (\#queries) - \lambda_2 * (CPU\ usage) - \lambda_3 * (net\ usage) - \lambda_4 * (balance\ load)$$

subject to constraints:

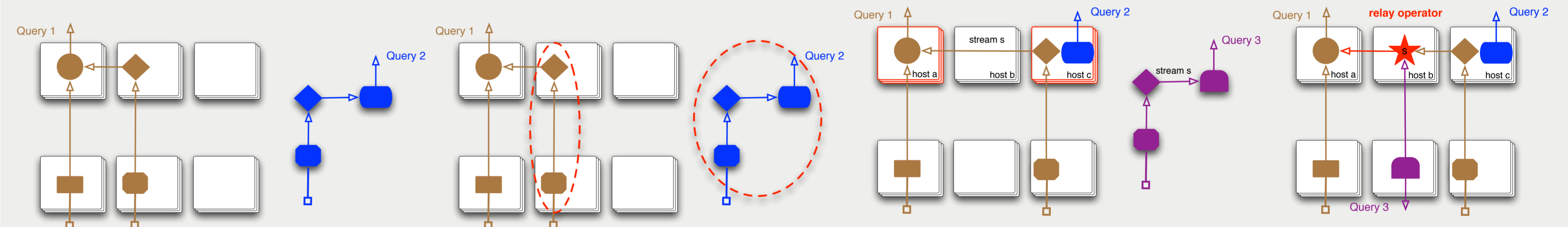
1. *availability*: streams for operators are present in hosts
2. *resources*: allocations are within the physical capacity
3. *demand*: a query is satisfied if final stream is generated
4. *acyclicity*: all streams come from a real source

→ This is an **NP-hard** problem when optimising for all existing and new queries.

SQPR Opts for *Tractable* Solutions by:

1. solving the optimisation problem for new queries
2. keeping existing queries running
3. optimising over new queries and re-planning common operators from old and new queries

SQPR Optimisation Example



Query (1) is running and (2) arrives

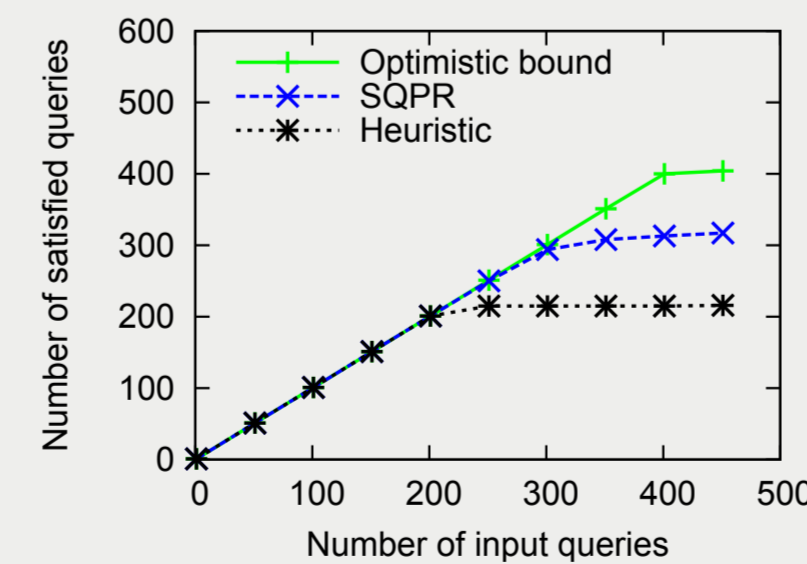
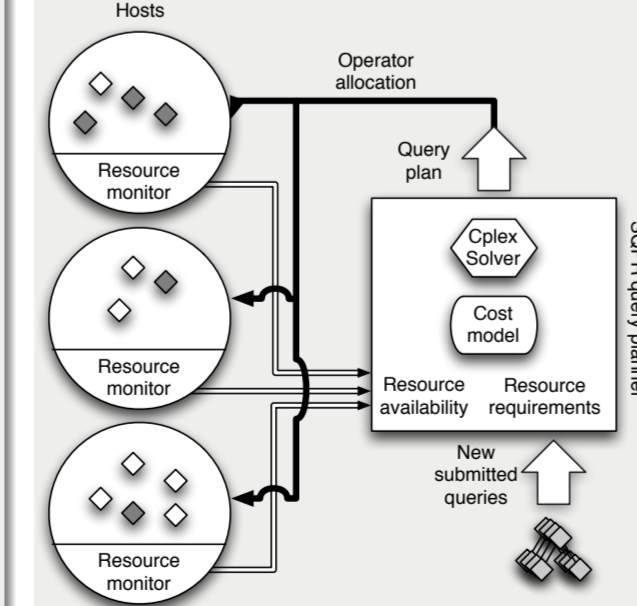
Planning new and common operators

Query (3) arrives; assume hosts a and b have used up their outgoing net bw

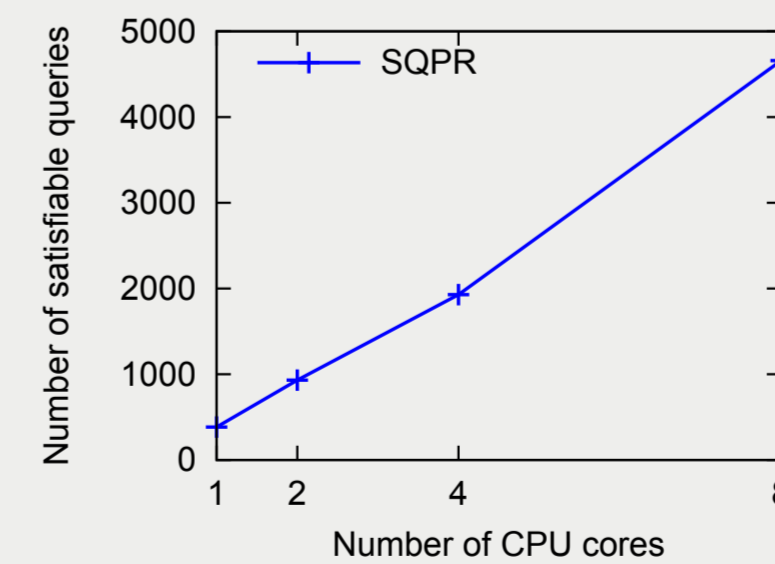
SQPR introduces **relay** operators to scale to resources/queries

Experimental Evaluation

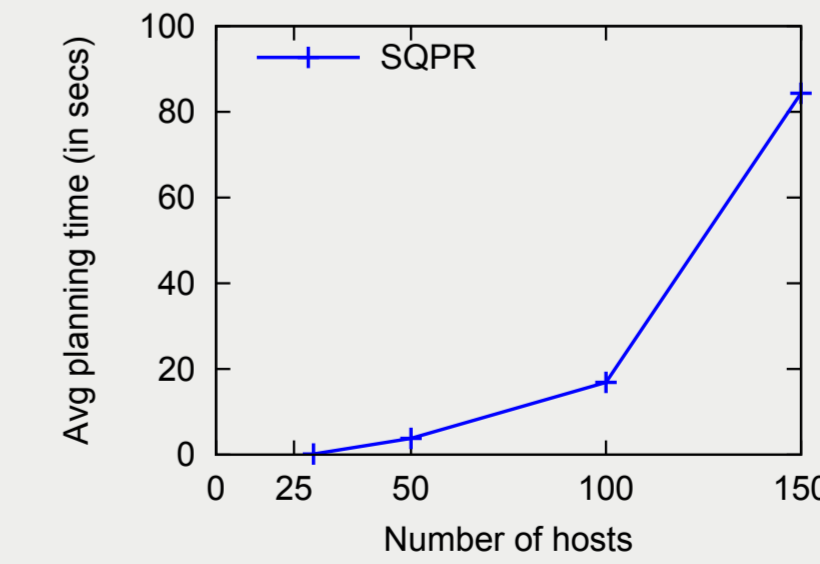
Architecture



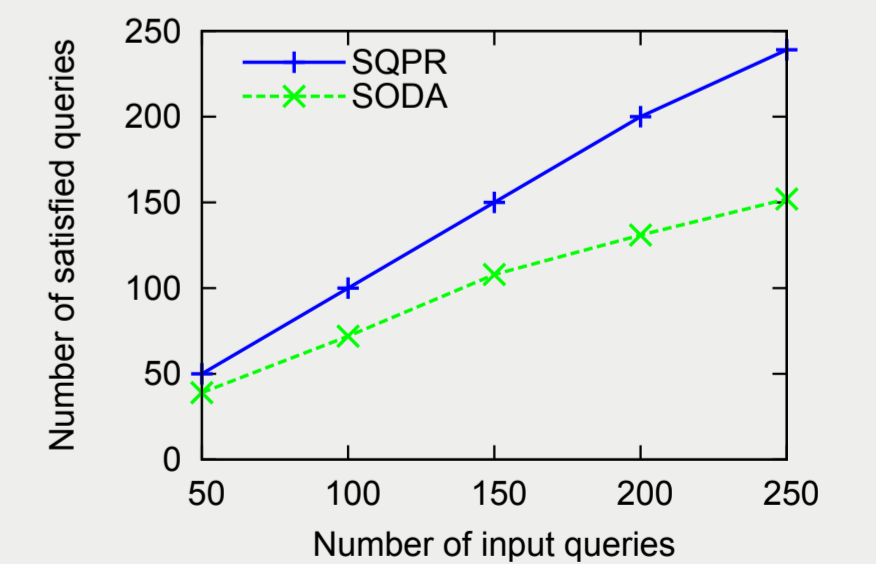
SQPR plans more queries than a heuristic and approaches the optimistic bound



In modern multi-core servers, SQPR scales to the number of cores per machine



With increasing number of nodes per data-center, SQPR has to solve larger problems



SQPR admits more queries than the SODA scheduler [1], because of relay operators