Eddies: Continuously Adaptive Query Processing

Ran Avnur, Joseph M. Hellestein
University of California, Berkeley

CPSC 504 Data Management
Presentation: Jonas Tai
Discussion Lead: Dorna Dehghani

Run-Time Fluctuations

Queries executions have dynamic run-time properties:

- Costs of operators
- Selectiveness of operators
- Arrival rate of inputs

Example: Selecting *salary > 100.000* on a table ordered by age

Static query plan inappropriate for systems with distributed data sources!

Goal: Dynamic reordering of query processing operators during run-time

How can we reorder operators and dynamically adjust input orders?

Synchronization Barrier:

Task needs to wait for another task to finish before it can continue

Example: Table scan of merge join needs to wait for smaller tuple from other scan

Moment of Symmetry:

Order of operators can be changed

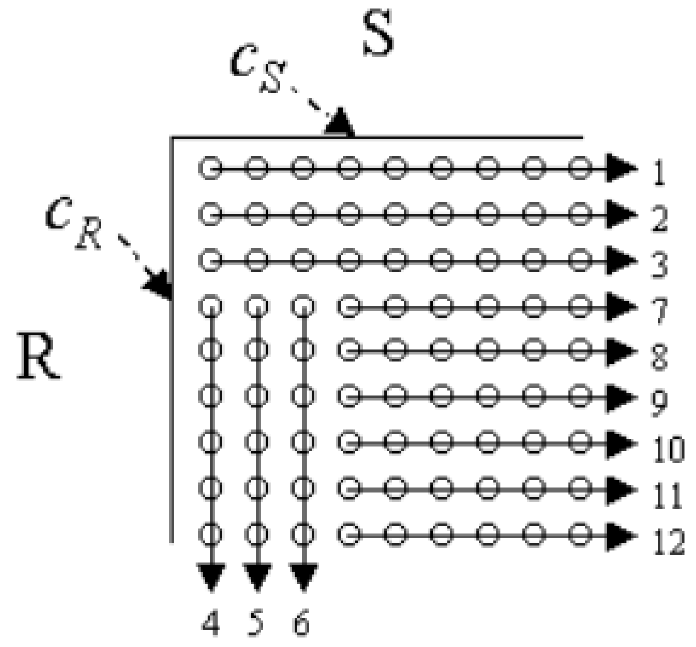
Moments of Symmetry – Example R ⋈ S

Nested Loop Join

- Asymmetric: Choice of inner relationship impacts performance
- Moment of symmetry - End of inner loop
  - Can reorder R ⋈ S to S ⋈ R
  - With commutativity, we can extend this to work on a tree of *n* binary joins

Merge Join:

- Symmetric operator, treats both inputs uniformly
- Synchronization barrier



Desirable Join Properties

Merge Join

Constrained by ordering condition and has imbalanced synchronization barriers

Nested Loop Join

Infrequent moments of symmetry

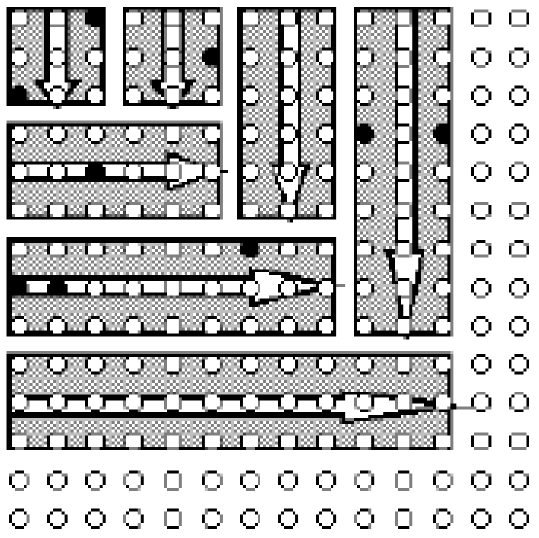
Hybrid Hash Join

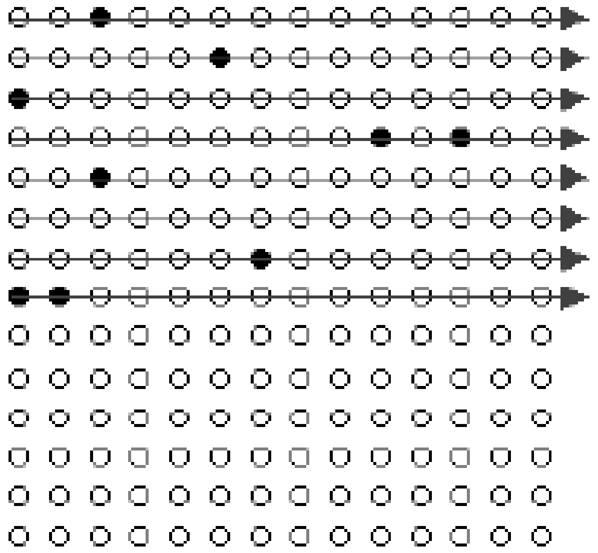
Large state that might have to be modified or recomputed on reordering

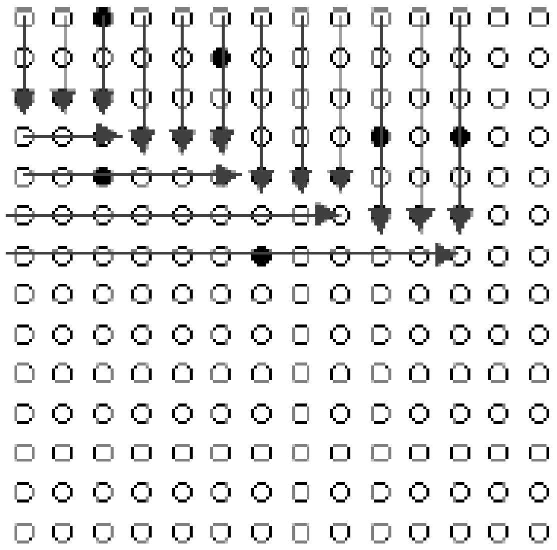
Eddies: Adaptivity > Best-case performance

Want to be able to reoptimize as often as possible!

Ripple Join Family


Block


Index


Hash

- Frequent Moment of symmetry at each corner / after each consumed tuple
- Next corner can be chosen adaptively

Discussion 4: Eddies philosophy

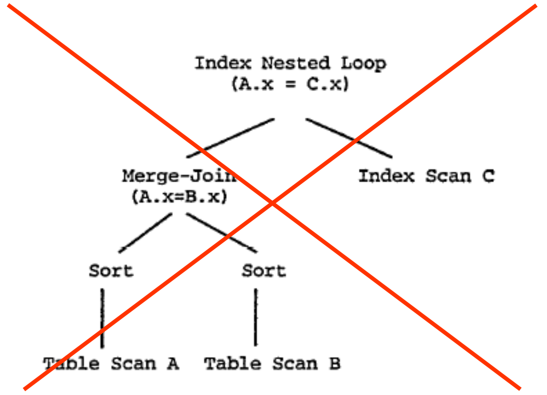
“We Favor Adaptivity Over Best-case Performance”

Consider if adaptivity is needed only when the best-case missing (unable established for lack of statistics, or non-existence because of changing environment) or could also be a general strategy in regular query processing.

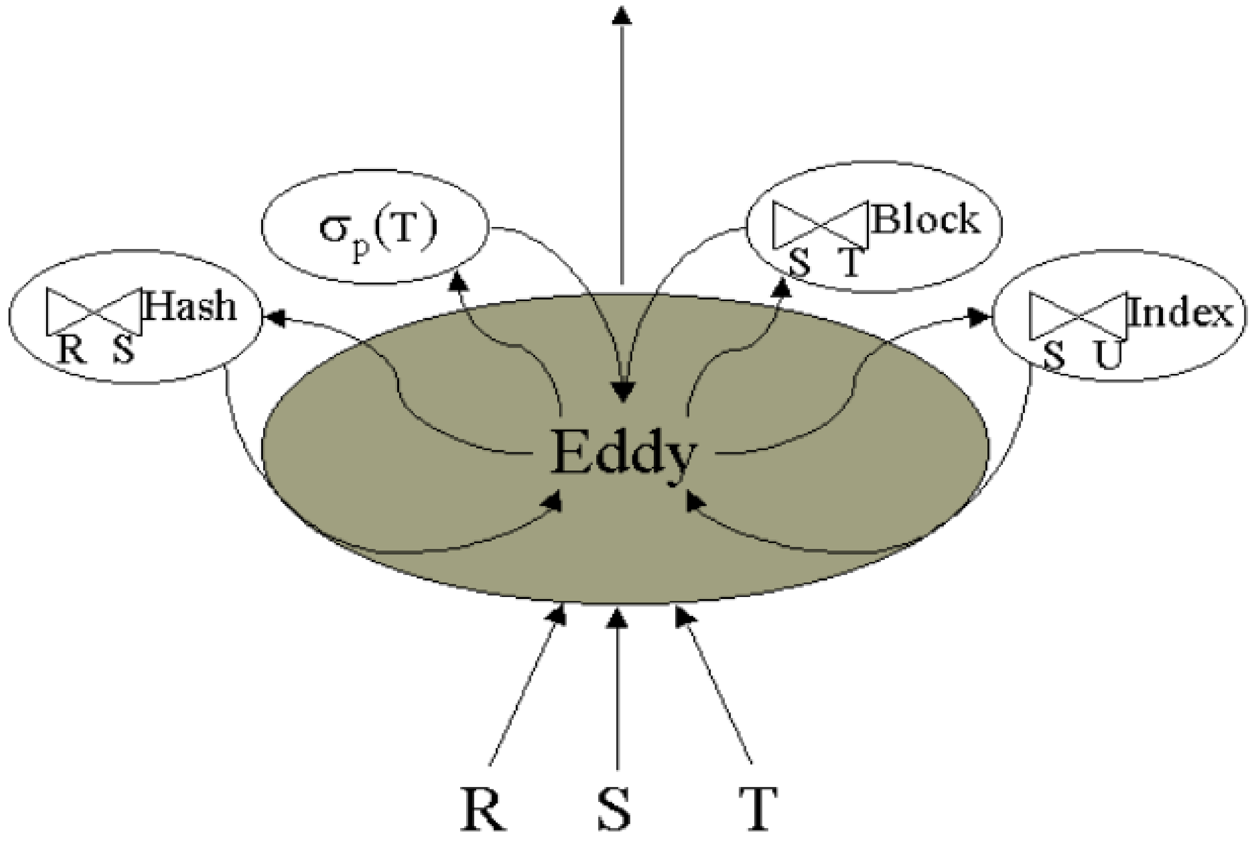
Do you think it is good or bad to apply it in the traditional query processing? Why? Please give reasons or use examples to support your opinions.

Discuss in groups of 2.

General Idea of Eddies



- Instead of static queries, the whole query or subtrees are combined in an Eddy
- The flow of tuples from *n* inputs is directed through operators to outputs
- Adaptive routing on a tuple basis guides performance
- Track which operators can process a tuple (data dependencies)

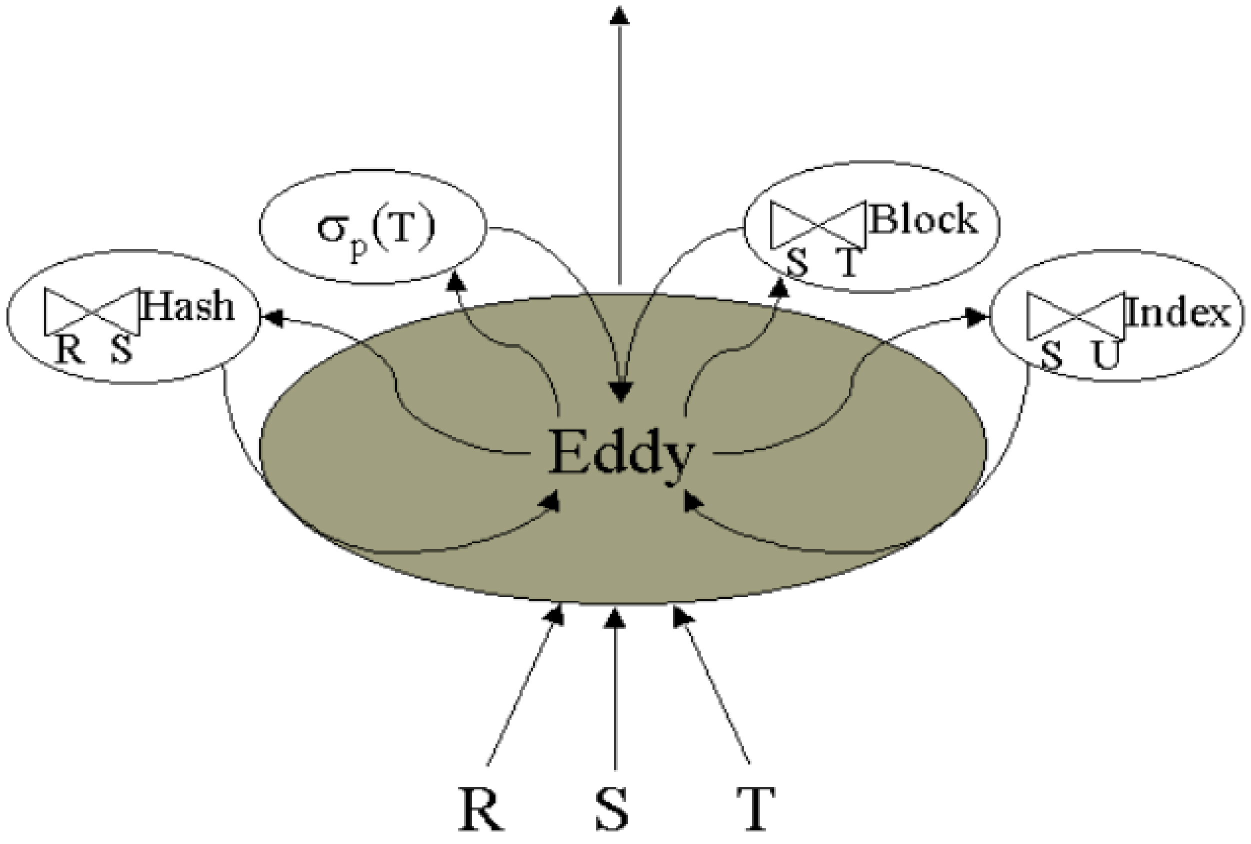


Routing Inside Eddies – Naive Eddy

Naïve Priority System:

- Assign incoming tuples low priority and returned tuples from operators high priority
- Encourage tuples to flow through the complete Eddy
- Cost-aware: Fixed-size queues limit consumption of slow operators and guide tuples to fast operators first

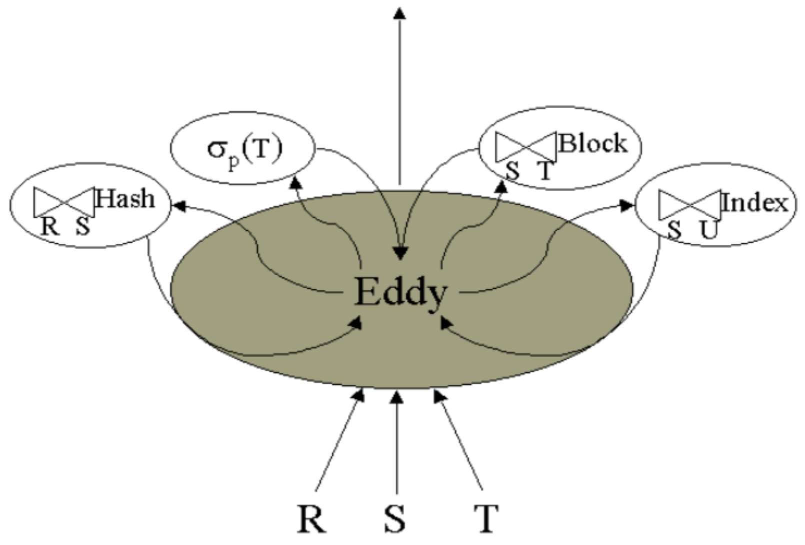
Problem: Does not factor in selectivity!



Routing Inside Eddies - Lottery Scheduling

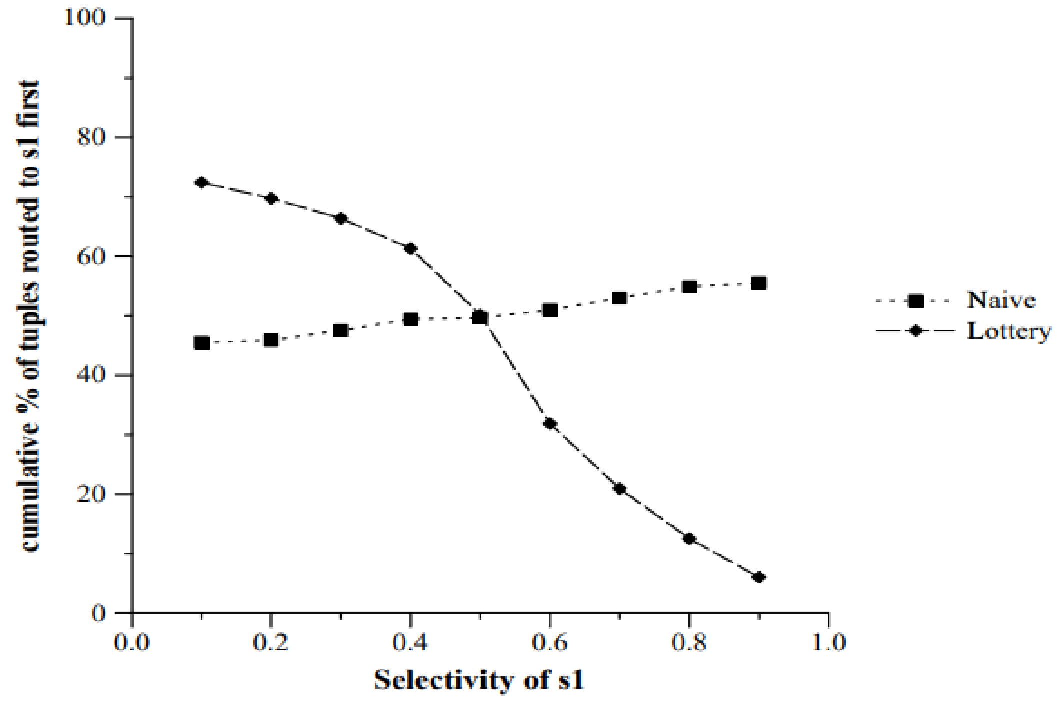
- Operators collect tickets:
  - Operators get one lottery ticket per assigned tuple
  - Eddy removes one ticket of the operator for each returned tuple
- Number of tickets = Efficiency of operator at removing tuples from the system
- Assignment of new tuples: Hold lottery with eligible operators

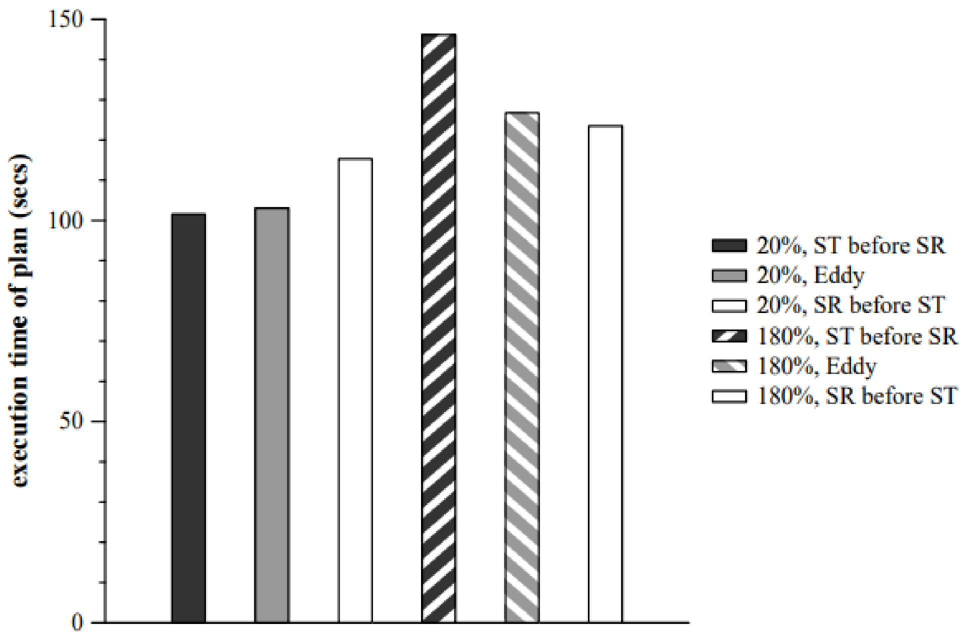
Extension: Use a window to avoid ticket inflation and to deal with non-static environments



No. of tickets = Tuples input – tuples output

Does this work?





Adaptivity > Best-case!

Discussion 4: Tukwila or Eddies?

- It's the year 2024, and there's a huge amount of data available. People expect to access it instantly through a vast global network of connected computers.

- Which one do we prefer, Tukwila or Eddies? Why?
- What’s the pros and cons of each method?

- Discuss in groups of 4.

Summary

- Distributed data sources require run-time reoptimization
- Aggressive dynamic per tuple routing through Eddies
- Lottery system to guide tuple routing
- Prioritize adaptivity over best-case performance for operators