Eddies: Continuously Adaptive Query Processing

CPSC 504 Presentation

Avnur, R. and Hellerstein, J. M. 2000. Eddies: continuously adaptive query processing. In *Proceedings of the 2000 ACM SIGMOD international Conference on Management of Data* (Dallas, Texas, United States, May 15 - 18, 2000). SIGMOD '00. ACM Press, New York, NY, 261-272.

Outline

- **What's the Problem?**
- Synchronization Barriers
- Moments of Symmetry
- Choosing a Join Algorithm
- Eddies
- Lottery Scheduling
- Results

What's the Problem?

- "optimize-then-execute" does not work well for data integration applications
- Why?

What's the Problem?

- Because of <u>unpredictable environment</u>:
  - Fluctuations in computing resources
    - Heterogeneous mix of hardware
      - bursty performance
  - Fluctuations in data characteristics
    - Federated data
      - Lack of statistics, complex data types
  - Fluctuations in user preferences
    - Queries run for a long time
      - User might want to control properties of queries while they execute (based on refining approximate results)

Discussion 1

- Their approach is to "favor adaptability over best-case performance."
  - Is this a good approach? Did they succeed?
  - How does this compare with other approaches we've seen?

Outline

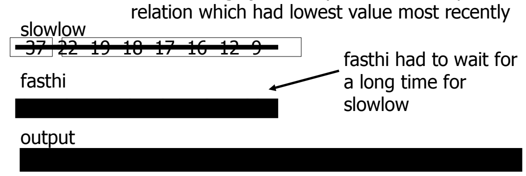
- What's the Problem?
- **Synchronization Barriers**
- Moments of Symmetry
- Choosing a Join Algorithm
- Eddies
- Lottery Scheduling
- Results

Synchronization Barriers

- What are synchronization barriers?
  - When table-scan A has to wait for table-scan B to finish or reach something before A can continue
- Synchronization barriers limit concurrency

Synchronization Barriers

- Example
  - Merge join - extreme case
    - Recall: merge join always takes next tuple from relation which had lowest value most recently



Outline

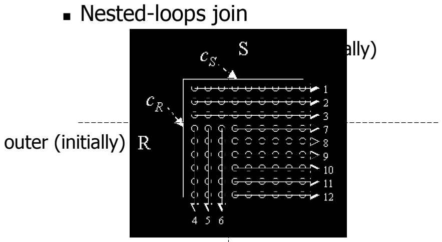
- What's the Problem?
- Synchronization Barriers
- Moments of Symmetry
- Choosing a Join Algorithm
- Eddies
- Lottery Scheduling
- Results

Moments of Symmetry

- What are moments of symmetry?
  - Moment of symmetry = synchronization barrier where the join algorithm can swap inputs without modifying any state in the join

Moments of Symmetry

- Example
  - Nested-loops join



Moments of Symmetry

- We switched roles of R and S two times during the join!
  - But we still end up "matching up" all the tuples (just in a different order)
  - And we didn't have to change any internal state (except for remembering C_R and C_S)

Outline

- What's the Problem?
- Synchronization Barriers
- Moments of Symmetry
- Choosing a Join Algorithm
- Eddies
- Lottery Scheduling
- Results

Choosing a Join Algorithm

- What do we want in a join algorithm?
  - Frequent moments of symmetry
  - Adaptive/Non-existent synchronization barriers
  - Minimal ordering constraints
- So we can't use hybrid hash join, merge join, nested loop join…

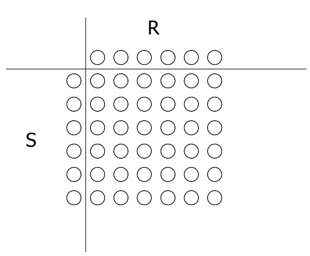
Choosing a Join Algorithm

- Instead, we choose the Ripple Join family:
  - Block Ripple Join
  - Index Ripple Join
  - Hash Ripple Join
  - …and relatives

Choosing a Join Algorithm

- Example
  - 1) Get tuple r from R, or tuple s from S
  - 2) Compare r and s with every tuple in r×s rectangle

Choosing a Join Algorithm



Choosing a Join Algorithm

- Ripple Join is basically like nested loops join except that moments of symmetry occur much more often
  - In fact, at every corner of a rectangle there is a moment of symmetry (i.e. between every tuple for hash and index ripple joins)
- <u>Offers adaptivity at modest overhead in performance and memory</u>

Outline

- What's the Problem?
- Synchronization Barriers
- Moments of Symmetry
- Choosing a Join Algorithm
- **Eddies**
- Lottery Scheduling
- Results

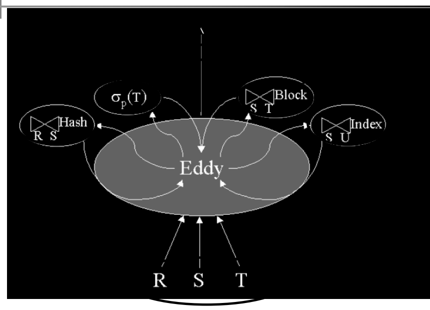
19

Eddies

- What are eddies?
  - Eddy = n-ary tuple router interposed between n data sources and a set of query processing operators
- Basically, eddy
  - 1) takes n tuples as input
  - 2) feeds tuples to operators
  - 3) operators feed result back to eddy
  - 4) eddy sends result
    - to output; or
    - to other operators

20

Eddies



s

21

Eddies

- Each tuple entering eddy has a tuple descriptor
  - Eddy only sends tuple to operators with corresponding Ready=1
  - Eddy sets corresponding Done=1 when operator sends tuple back, and updates Ready bits
  - When all Done=1, eddy sends tuple to output

	Join_1	Select_1	Select_2	Join_2	Select_3
Ready	0	1	1	0	0
Done	1	0	0	0	1

22

Eddies

- Naïve Eddy
  - Operator s1 lower-cost than s2
    - So s1 consumes input faster than s2
  - Equal selectivity
    - higher selectivity -> more likely to return tuples to eddy
- Back-pressure effect
  - s1 consumes fast, s2 produces slow, so most tuples end up being routed to s1 first
- Desired effect, even though costs were not explicitly exposed or tracked!

23

Outline

- What's the Problem?
- Synchronization Barriers
- Moments of Symmetry
- Choosing a Join Algorithm
- Eddies
- **Lottery Scheduling**
- Results

24

Lottery Scheduling

- Need a learning algorithm to track both consumption and production over time
  - I.e. take into account selectivity (as well as cost)
- Lottery Scheduling
  - Each time eddy gives tuple to operator, operator gets 1 ticket
  - Each time operator returns tuple to eddy, operator loses 1 ticket
  - Operator must use possessed tickets to win "lottery" to get new tuples
- Therefore more "efficient" operator -> more tickets -> more likely to win lottery -> more likely to get tuples

25

Outline

- What's the Problem?
- Synchronization Barriers
- Moments of Symmetry
- Choosing a Join Algorithm
- Eddies
- Lottery Scheduling
- Results

26

Results

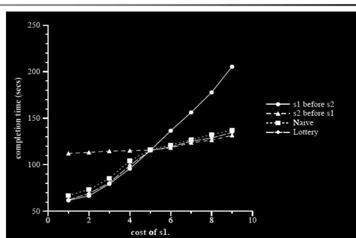


Figure 4: Performance of two 50% selections, **s2** has cost 5, **s1** varies across runs.

27

Results

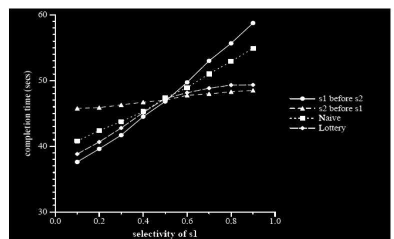


Figure 5: Performance of two selections of cost 5, **s2** has 50% selectivity, **s1** varies across runs.

28

Results

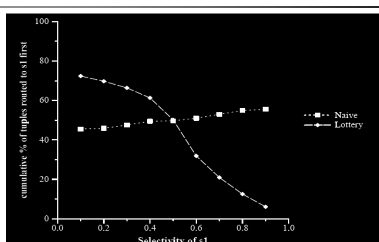


Figure 6: Tuple flow with lottery scheme for the variable-selectivity experiment(Figure 5).

29

Results

- Other experiments
  - Responding to dynamic fluctuations
    - Using window, banked vs. escrow tickets…
  - Delayed delivery
    - Handling initial delay of tuples from input relation

30

Discussion 2

- Compare and contrast this system to Mariposa's bidding approach.
  - Is one better than the other?
  - Could you combine the two?
- Compare and contrast Eddies and Tukwila.
  - Which is better?

31