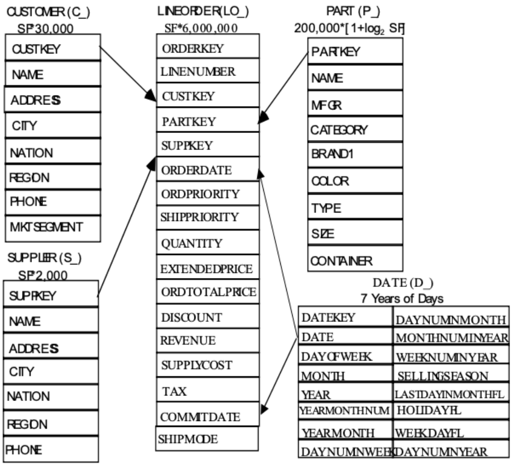
Accelerating Joins with Filters: Keeping a Limited Memory is Robust

Nicholas Corrado Xiating Ouyang

University of Wisconsin-Madison

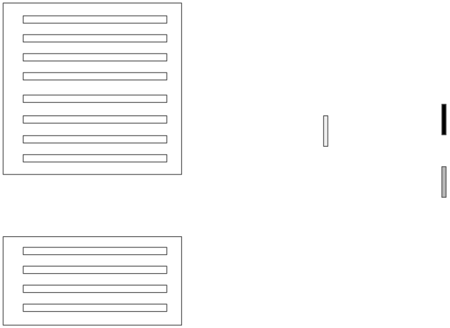
Star Schema



- If the query optimizer chooses a poor join order, intermediate join results may be unnecessarily large.
- Solution: try to filter out extraneous tuples before performing joins

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)



Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

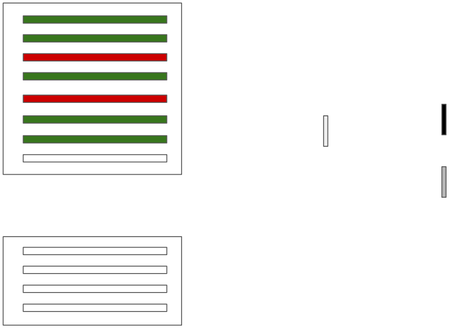
Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

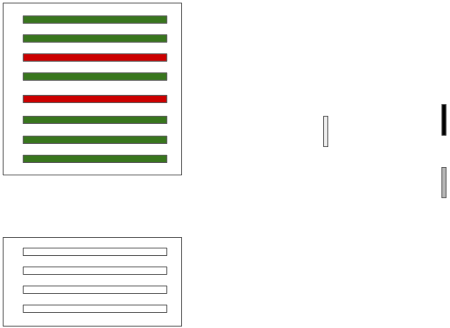
Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)



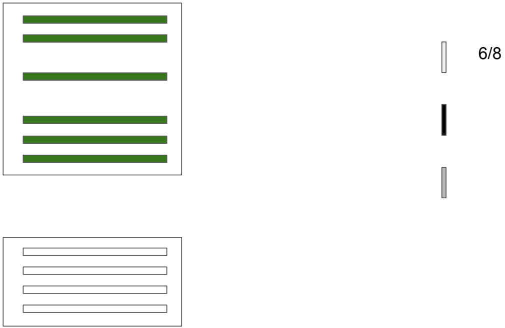
Lookahead Information Passing (LIP)



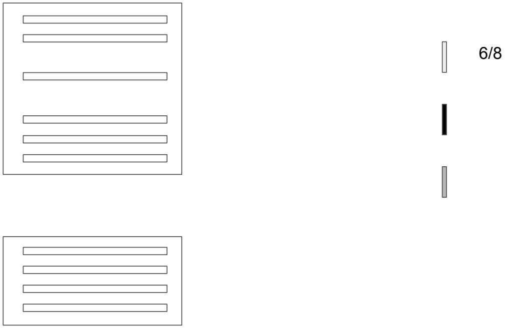
Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

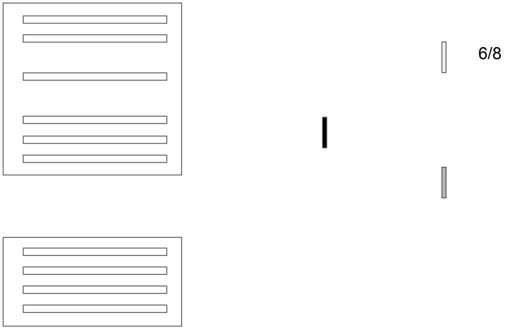
Lookahead Information Passing (LIP)



Lookahead Information Passing (LIP)



Lookahead Information Passing (LIP)



Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

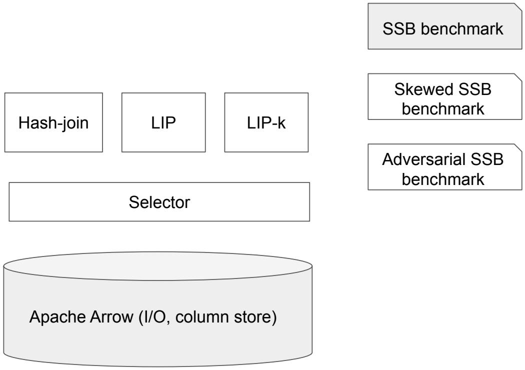
Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

Lookahead Information Passing (LIP)

- LIP uses statistics from all previous batches to compute σ
  - Slow response to local changes in key distributions
- **LIP-k**: Only use the previous k batches to compute σ

Implementation and benchmarking



An Example Experiment

Skewed Key	
$\sigma = 1$	
$\vdots$	50 batches
$\sigma = 1$	
$\sigma = 0$	
$\vdots$	50 batches
$\sigma = 0$	
$\sigma = 1$	
$\vdots$	50 batches
$\sigma = 1$	
$\vdots$	

- LIP-k perform performs better than LIP on some queries...
- ...but LIP performs better on others...

LIP is solving an online problem

- Tuples arriving one at a time
- Upon arrival, decide a sequence of filters
- Minimize the total probes
- Deterministic!

LIP is solving an online problem

- Tuples arriving one at a time
- Upon arrival, decide a sequence of filters
- Minimize the total probes
- Deterministic!

Theorem

Let n be the number of filters in the LIP problem. There is no deterministic mechanism $\mathcal{M}$ achieving a competitive ratio less than n for the LIP problem.

LIP is solving an online problem

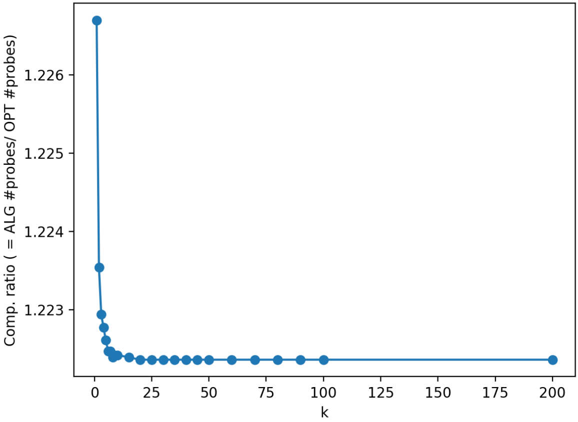
- Tuples arriving one at a time
- Upon arrival, decide a sequence of filters
- Minimize the total probes
- Deterministic!

Theorem

Let n be the number of filters in the LIP problem. There is no deterministic mechanism $\mathcal{M}$ achieving a competitive ratio less than n for the LIP problem.

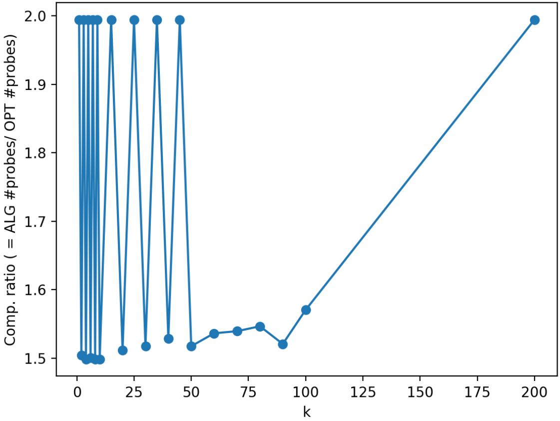
- Not observed in practice, but a theoretical lower bound
- Randomness?

Competitive Ratio vs. k on Uniform Data



Competitive Ratio vs. k on Adversarial Data

- Adversarial dataset constructed such that LIP-k has worst case performance for odd k



Conclusion

- Implemented LIP and its variant LIP-k
- LIP-k is better than LIP in the adversarial/skewed settings
- Randomness to achieve better robustness guarantee

Thank you!

Questions?