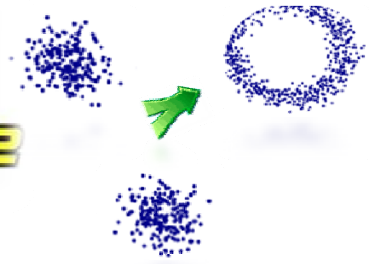
Toward Intersection Filter-Based Optimization for Big Joins



Thuong-Cang Phan (thuongcang.phan@isima.fr)
Laurent d'Orazio (laurent.dorazio@isima.fr)
Philippe Rigaux (philippe.rigaux@cnam.fr)

Context

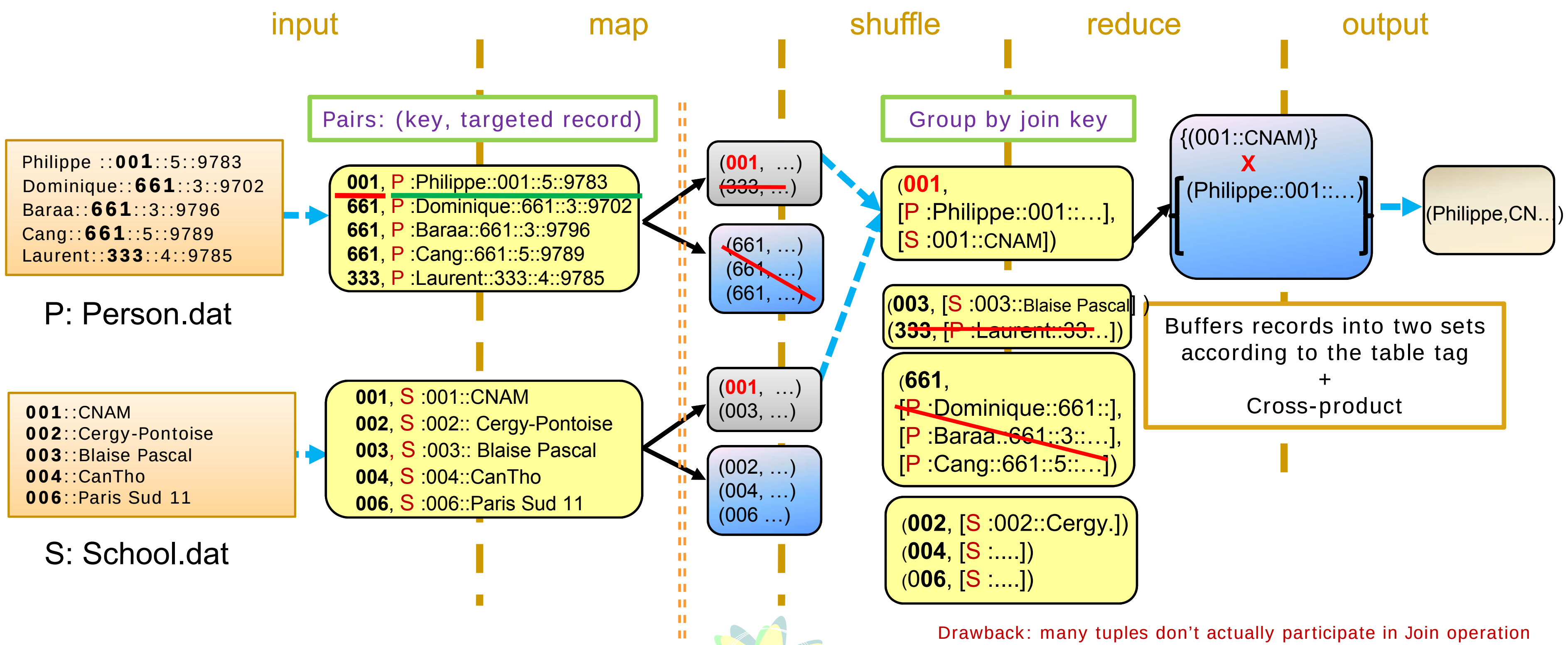
▪Mapreduce

- ✓ a popular big data processing framework
- ✓ its basic complex operations used extensively and expensively
  - ➢ join operations : $R_1(X_1) \bowtie R_2(X_2) \bowtie .... \bowtie R_n(X_n)$

▪Big join

- ✓ an important operation for efficient data analysis&query evaluation
- ✓ NOT a straightforward implementation in Mapreduce
- ✓ compiled to MapReduce job(s)
- ✓ Join algorithms:Map-side join,Reduce-side join,Broadcast join,etc.
  - ➢ Too much unnecessary intermediate data generated in the map phase

Problem: Intermediate data in Join



300,000,000,...

Don't be *wasting ...*

They significantly increase the costs :

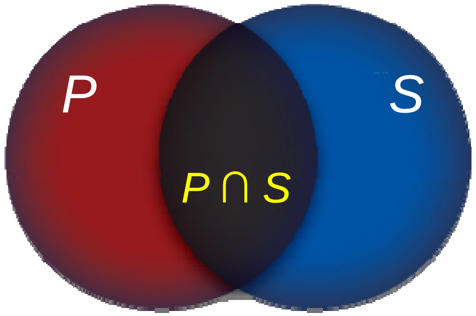
- **I / O operations for intermediate results**
- **Communication cost**

P(.., sch-id) ⋈ S(sch-id, ...)

Reduce-Side Join

Bloom Filters BF(S)

Proposed Solution



$P \cap S = (P \cup S) \setminus (P \,\Delta\, S)$

$P \cap S$ = {001}

P {(001, Philippe) , (661, Barra), (333, Laurent)}

S {(001, CNAM) , (002, Lyon I), (003, BP)}

{(001, Philippe,CNAM)}

The intersection filter

- **Contributions:**

  (a) three approaches of the intersection filter that approximates the intersection of datasets;

  (b) the feasibility of our approaches used in two-way joins

  (c) the advantage of the intersection filter for important join cases

  (d) The considerable efficiency of the intersection filter as compared with basic filters in join operations.

Content

- Join algorithms in MapReduce
- Modeling Intersection Filter (I.F)
- Optimization of two-way join using I.F
- Advantage of I.F for important join cases
- Cost analysis and experimental evaluation

Join Algorithms in MapReduce

- **Reduce-side Join**

  The actual join happens on the Reduce side of the framework. The ‘map’ phase only pre-processes the tuples of the two datasets to organize them in terms of the join key.

- **Map-side Join**

  It is carried out on Mapper nodes. Both the input datasets for each map task must be already partitioned and sorted by the same join key.

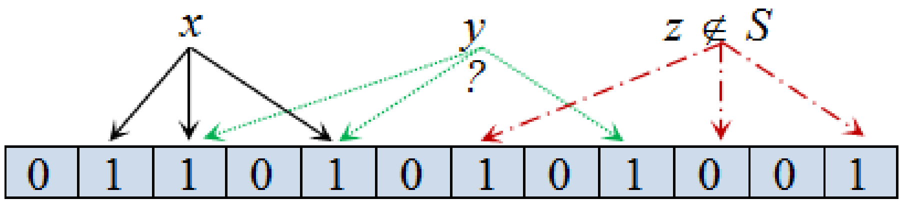
- **Broadcast Join**

  Mappers load the small dataset into memory and calls the map function for joining each tuple from the bigger dataset

Bloom Filter (BF)

✓ Bloom filter [Burton Howard Bloom in 1970] is a space-efficient probabilistic data structure used to test membership in a set with a small rate of false positives (a false positive probability).

✓ BF representing a static set $S = \{e_1, e_2, \ldots, e_n\}$ of n elements consists of an array of m bits and a group of k independent hash functions $h_1, \ldots, h_k$ with the range of $\{1, \ldots, m\}$.



A Bloom Filter.

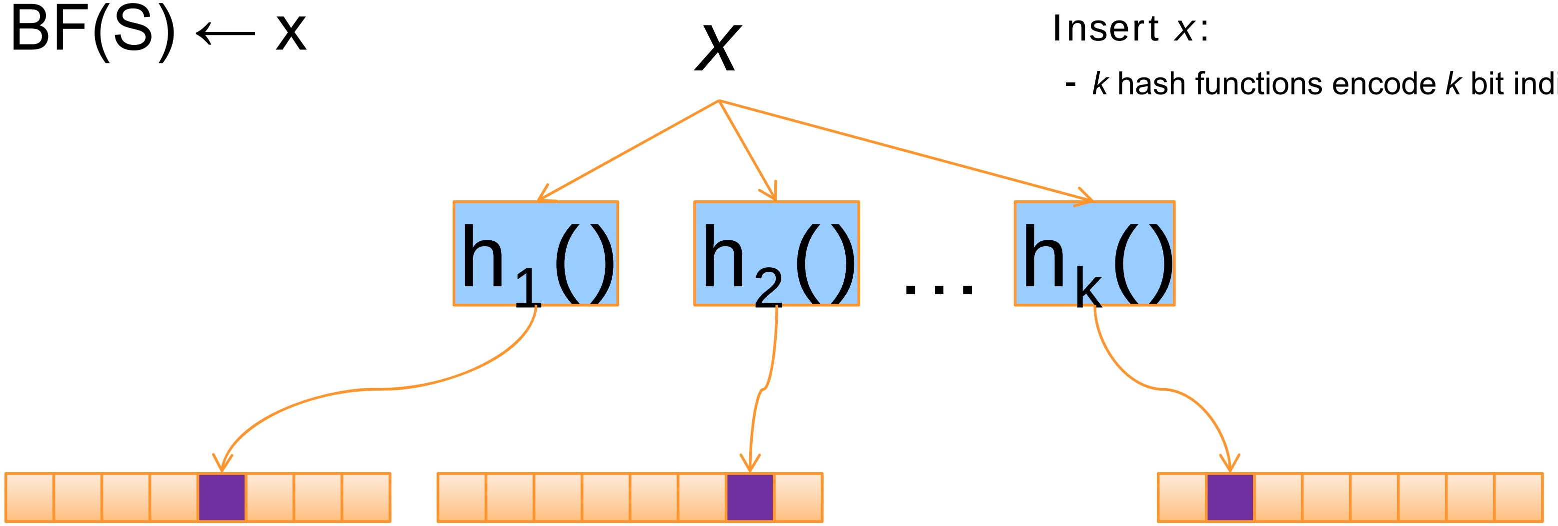
- No false negative
  z is definitely not a member.
- False positive
  y is probably a member; (may be wrong)
- Find optimal at $k = (\ln 2)m/n$, $p = 1/2$ by derivative of f

Partitioned Bloom Filter (PBF)

BF(S) ← x

Insert *x*:

- *k* hash functions encode *k* bit indices to set



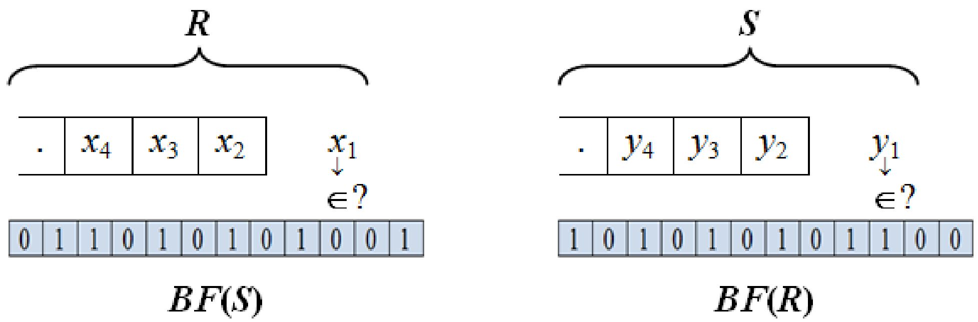
k partitions of length *m/k* bits

Content

- Join algorithms in MapReduce
- Modeling Intersection Filter (I.F)
- Optimization of two-way join using I.F
- Advantage of I.F for important join cases
- Cost analysis and experimental evaluation

Modeling Intersection Filter

- Three approaches to building the intersection filter

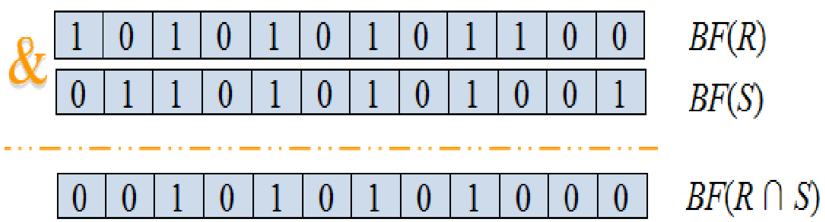


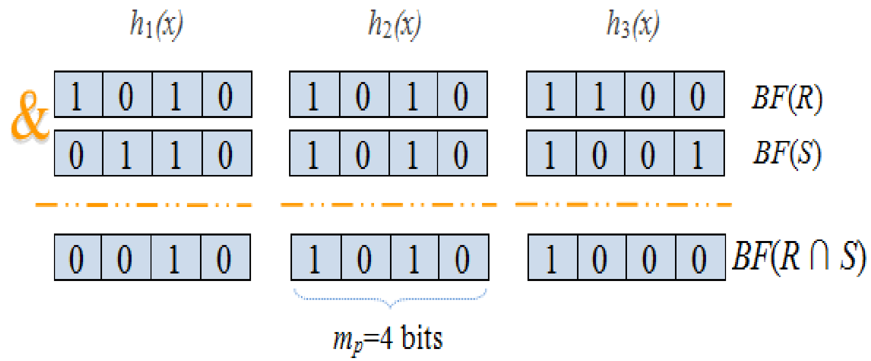
$R \cap S = (R \cup S) \setminus (R \Delta S)$
$= (R \cup S) \setminus ((R \setminus S) \cup (S \setminus R))$

$BF(S)$ $BF(R)$

(1) A pair of Bloom filters

$BF(R \cap S) = BF(R) \cap BF(S)$ with probability $(1-1/m)^{k|R-R\cap S|.k|S-R\cap S|}$





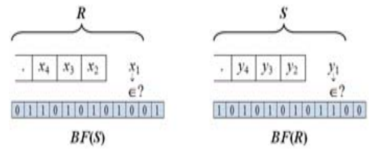
(2) Unpartitioned BF Intersection

(3) Partitioned BF Intersection

The false intersection probability

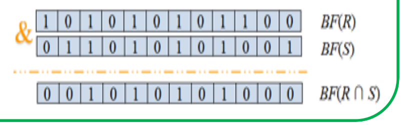
THEOREM 1. *A false intersection by a pair of Bloom filters is identified with one of probabilities*

$$f_{\cap pair(R)} = \left(1-\left(1-\frac{1}{m_1}\right)^{k_1|R|}\right)^{k_1} \qquad f_{\cap pair(S)} = \left(1-\left(1-\frac{1}{m_2}\right)^{k_2|S|}\right)^{k_2}$$



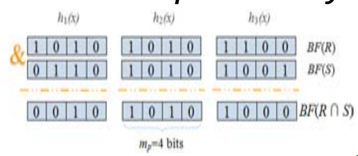
THEOREM 2. *A false intersection by intersecting* unpartitioned *filters is identified with probability*

$$f_{\cap BF} = \left(1-\left(1-\frac{1}{m}\right)^{k|R|}\right)^{k}\left(1-\left(1-\frac{1}{m}\right)^{k|S|}\right)^{k}$$



THEOREM 3. *A false intersection by intersecting partitioned filters is identified with probability*

$$f_{\cap PBF} = \left(1-\left(1-\frac{k}{m}\right)^{|R|}\right)^{k}\left(1-\left(1-\frac{k}{m}\right)^{|S|}\right)^{k}$$

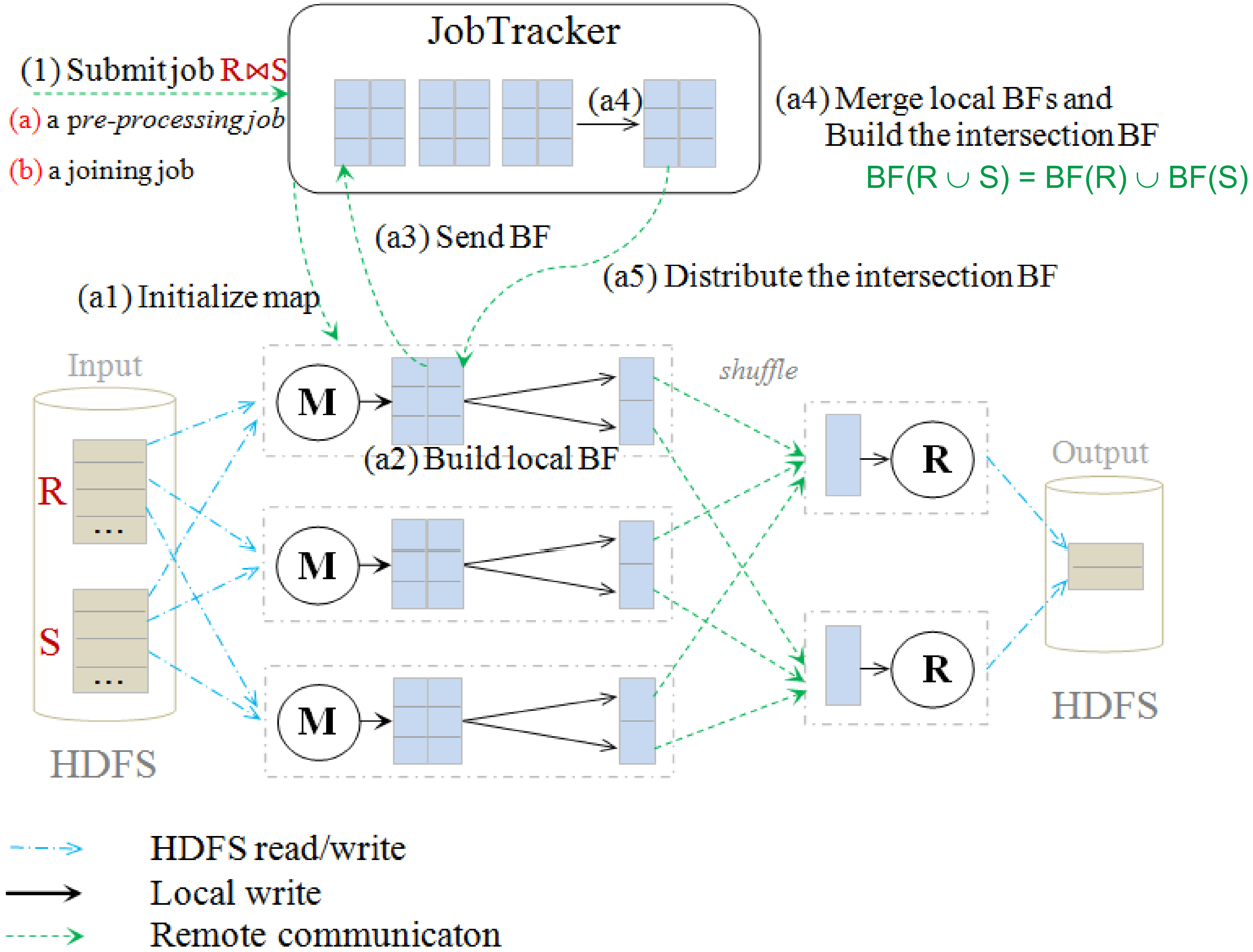


THEOREM 4. *The false intersection probability of the unpartitioned filter intersection is less than the false intersection probability of the partitioned filter intersection* $f_{\cap BF} < f_{\cap PBF}$

Content

- Join algorithms in MapReduce
- Modeling Intersection Filter (I.F)
- Optimization of two-way join using I.F
- Advantage of I.F for important join cases
- Cost analysis and experimental evaluation

Two-way join using Inter. Filter



Content

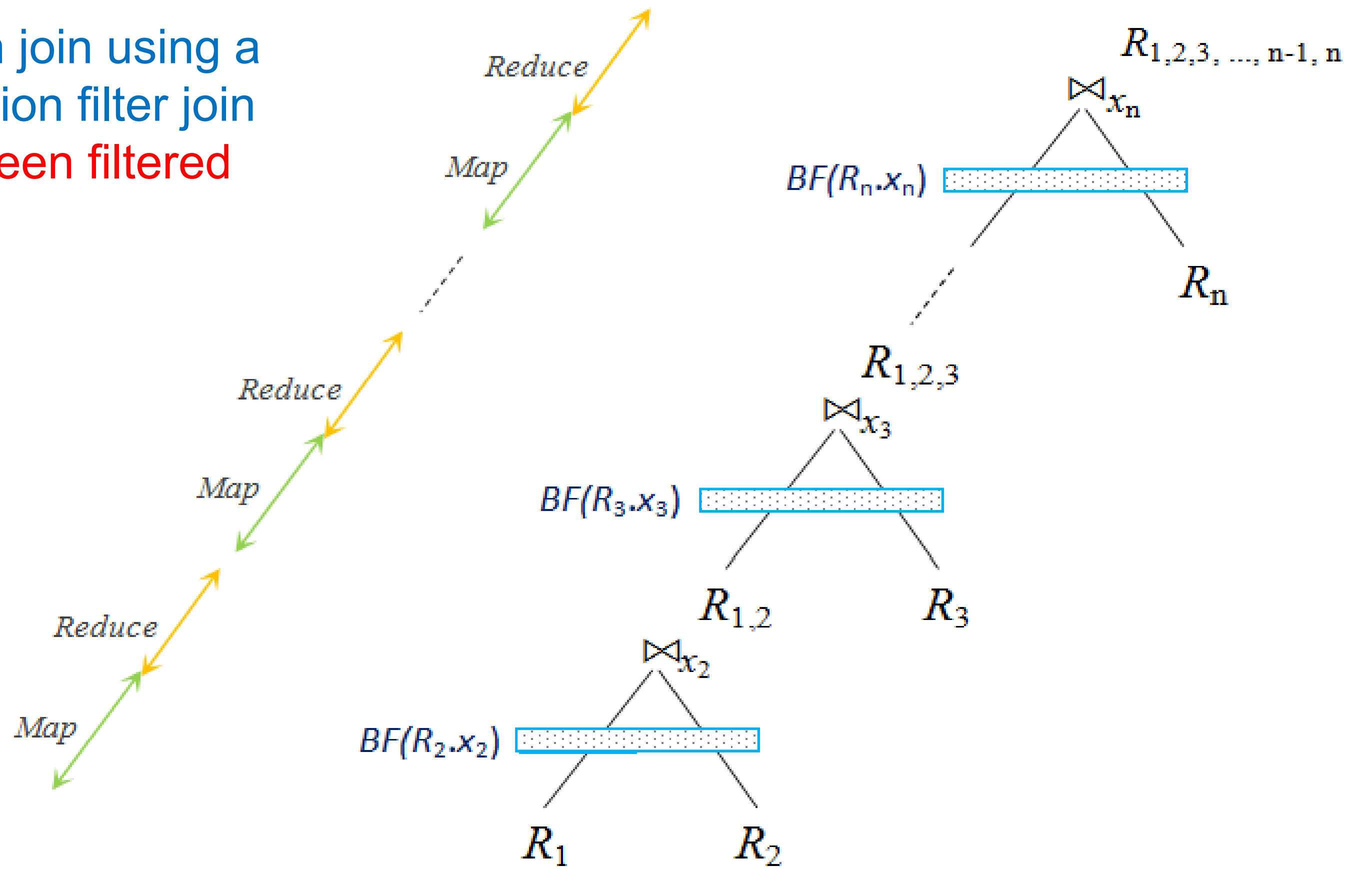
- Join algorithms in MapReduce
- Modeling Intersection Filter (I.F)
- Optimization of two-way join using I.F
- Advantage of I.F for important join cases
- Cost analysis and experimental evaluation

Advantage of I.F for important join cases

- **Chain Join** $R_1(x_1, x_2) \bowtie R_2(x_2, x_3) \bowtie R_3(x_3, x_4) \bowtie \ldots \bowtie R_n(x_n, x_{n+1})$

Execution of a chain join using a cascade of intersection filter join

$R_2, R_3, \ldots, R_n$ have been filtered

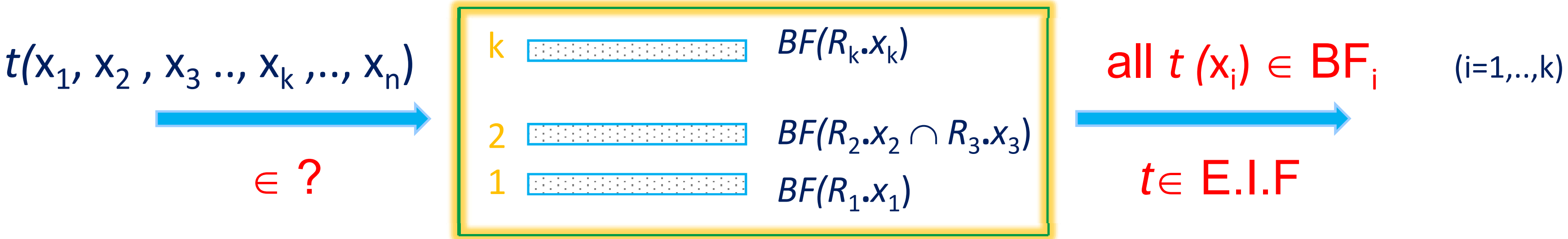


Advantage of I.F for important join cases

▪ I.F based optimization of a chain join

➢ Extended intersection filter (E.I.F)

includes an array of Bloom filters hashed on different join keys. Each tuple of a dataset may contain a few join keys linking to others. The tuple is eliminated if at least one of its join keys, x_i, is not a member of a component filter BF_i of the extended filter.



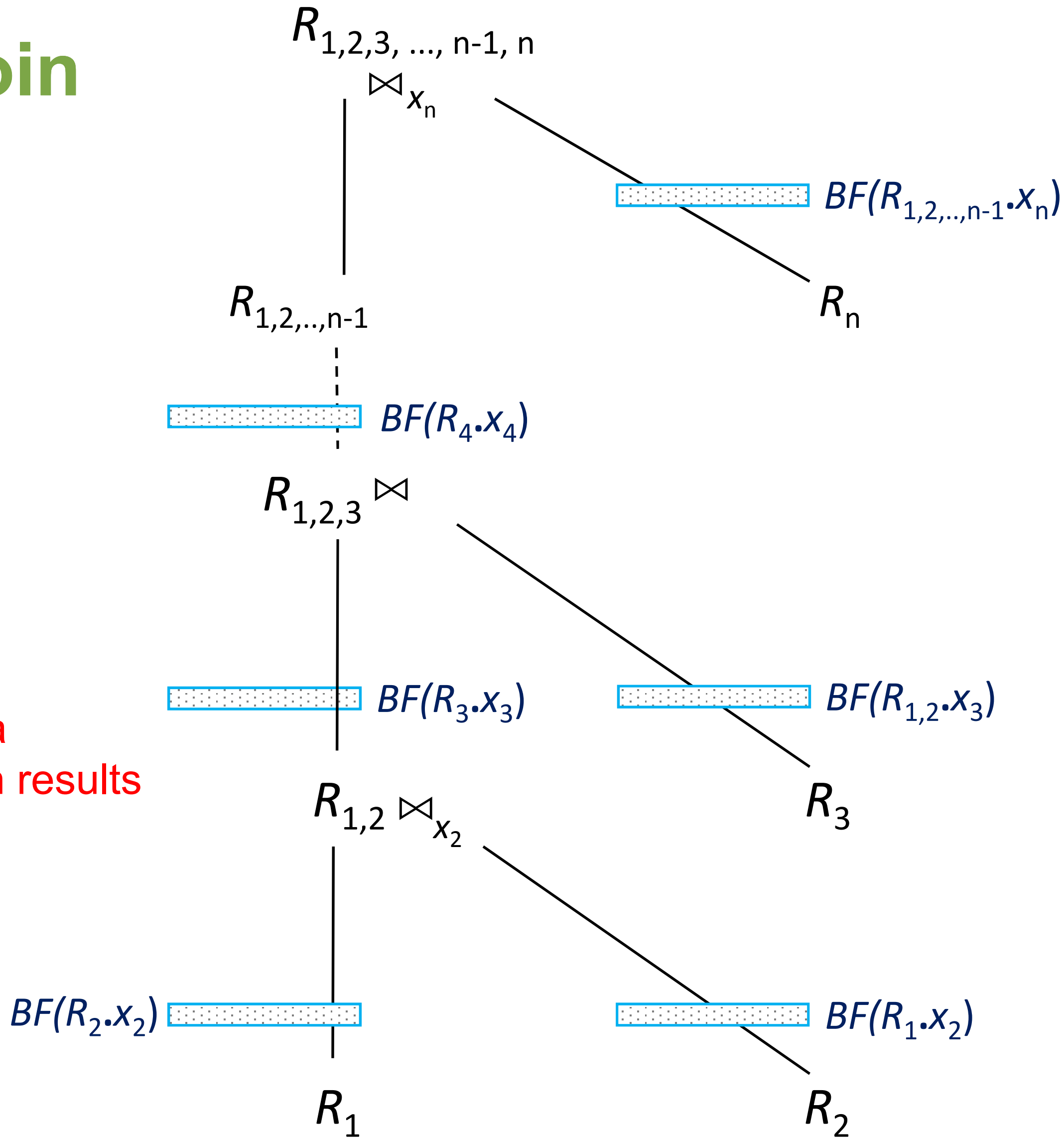
Extended intersection filter (E.I.F)

Advantage of I.F for important join cases

- **Chain Join**

Optimization of a chain join with extended intersection filters

NO redundant data
In intermediate join results

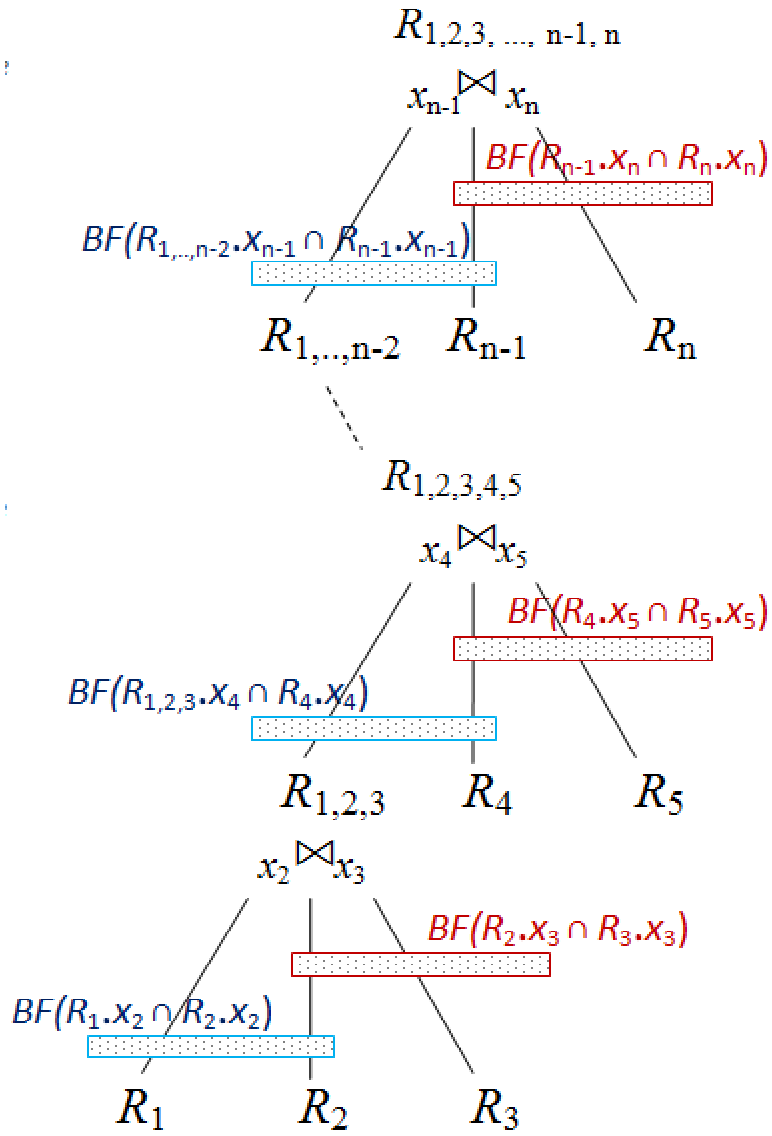


Advantage of I.F for important join cases

- **Chain Join**

Optimization of a chain join with extended intersection filters

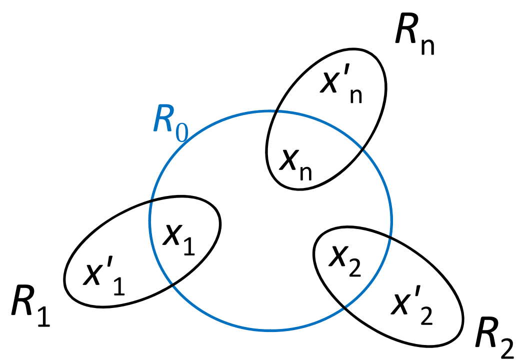
Three-way join
reduces the number of intermediate join jobs

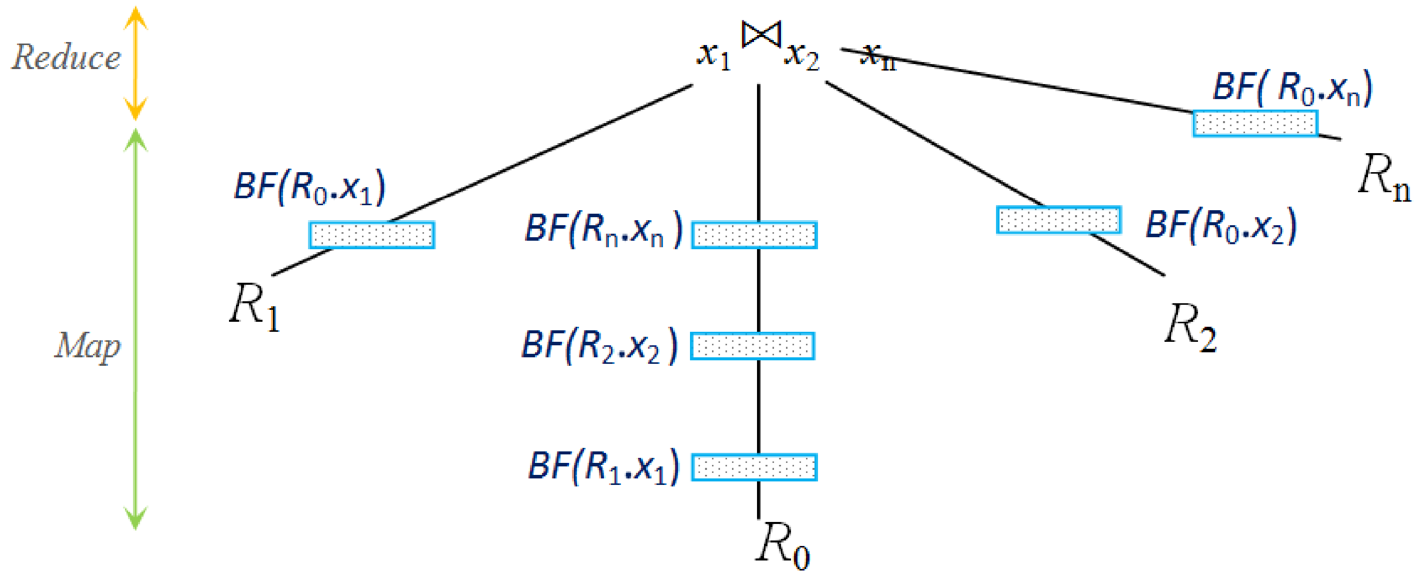


Advantage of I.F for important join cases

- **Star Join**

Optimization of a star join with extended intersection filters





E.I.F reduces the number of intermediate join jobs to zero, NO redundant data.

Content

- Join algorithms in MapReduce
- Modeling Intersection Filter (I.F)
- Optimization of two-way join using I.F
- Advantage of I.F for important join cases
- Cost analysis and experimental evaluation

Cost Analysis for Two-way Join

- **Cost model**

  The total cost of the join operation:

$$C = C_{pre} + C_{read} + C_{sort} + C_{tr} + C_{write}$$

where

$C_{read} = c_r \,.\, |R| + c_r \,.\, |S|$; $C_{write} = c_r \,.\, |O|$; $C_{tr} = c_t \,.\, |D|$

$C_{sort} = c_l|D|.2([\log_B|D| - \log_B(mp_1+mp_2)] + [\log_B(mp_1+mp_2)])$ [8]

$C_{pre} = C_{read} + 2 \,.\, c_t \,.\, m \,.\, t + c_t \,.\, m \,.\, r \,.\, t + a$

$a = c_t \,.\, m \,.\, r \,.\, t$ for the first approach, otherwise a = 0

Cost Analysis for Two-way Join

▪ Cost comparison of approaches

The size of intermediate data with the false intersection probability is

$$|D| = \begin{cases} \partial_S|R| + f_{\cap pair(S)} \cdot (1 - \partial_S)|R| + \partial_R|S| + f_{\cap pair(R)} \cdot (1 - \partial_R)|S| & (1) \\ \partial_S|R| + f_{\cap BF} \cdot (1 - \partial_S)|R| + \partial_R|S| + f_{\cap BF} \cdot (1 - \partial_R)|S| & (2) \\ \partial_S|R| + f_{\cap PBF} \cdot (1 - \partial_S)|R| + \partial_R|S| + f_{\cap PBF} \cdot (1 - \partial_R)|S| & (3) \\ |R| + \partial_R|S| + f_{\cap pair(R)} \cdot (1 - \partial_R)|S| & (4) \\ |R| + |S| & (5) \end{cases}$$

where
equation (1) for the pair of the filters (approach 1),
equation (2) for the unpartitioned intersection filter (approach 2),
equation (3) for the partitioned intersection filter (approach 3),
equation (4) for a filter *BF*(*R*), and
equation (5) in case without Bloom filter

Cost Analysis for Two-way Join

THEOREM 5. *The join operation using the intersection filter is more efficient than using a basic Bloom filter because it produces less redundant and intermediate data than the latter. Additionally, we can drive comparing equation for |D|*

$$|D|_1 \approx |D|_2 < |D|_3 < |D|_4 < |D|_5$$

where $|D|_i$ is the intermediate data size for equation i^{th} (i = 1..5).

THEOREM 6. The total cost of the join operation for our approaches is defined by

$$C_1 \approx C_2 < C_3 < C_4 < C_5$$

where C_i is the total cost in case of equation i^{th} (i = 1..5).

THEOREM 7. *The total cost to perform pre-processing step*

$$C_{pre} = \begin{cases} C_{read} + 2 \cdot c_t \cdot m \cdot t + 2 \cdot c_t \cdot m \cdot r \cdot t, & \text{in case of (1)} \\ C_{read} + 2 \cdot c_t \cdot m \cdot t + c_t \cdot m \cdot r \cdot t, & \text{in case of (2), (3), (4)} \\ 0 & \text{in case of (5)} \end{cases}$$

Conclusion

- Three approaches for building the intersection filter
- Their efficiency used in joins better than other solutions
- Their advantage for important join cases

- Although the intersection filter has false positives and an extra cost for the pre-processing step, its efficiency in space-saving and filtering often outweighs these drawbacks
- System will become inefficient if *t* and *r* is large or there is very little redundant data in the join operation.

Future work

- Implementation of general multiway joins, especially a cascade of map-side joins.
- Recursive joins.
- A complete optimizer for choosing the best join implementation in MapReduce.

References

[1] Bloom, B.H. 1970. *Space/time trade-offs in hash coding with allowable errors*. Commun. ACM.

[2] Broder, A. and Mitzenmacher, M. 2004. *Network Applications of Bloom Filters: A Survey*. Internet Mathematics.

[3] Lee, T., Kim, K. and Kim, H.-J. 2012. *Join processing using Bloom filter in MapReduce*. Proceedings of the 2012 ACM Research in Applied Computation Symposium (New York).

[4] Tom White's book 2010. *Hadoop: The Definitive Guide*, 2nd Edition. O'Reilly.

[5] PUMA: Purdue MapReduce Benchmarks Suite: *http://web.ics.purdue.edu/~fahmad/benchmarks.htm*.

[6] Foto N. Afrati and Jeffrey D. Ullman. 2010.*Optimizing joins in a map-reduce environment*. In *Proceedings of the 13th International Conference on Extending Database Technology* (EDBT '10).

[7] Michael, L., Nejd, W., Papapetrou, O. and Siberski, W. 2007. *Improving distributed join efficiency with extended bloom filter operations*. 21st International Conference on Advanced Information Networking and Applications, 2007. AINA '07.

[8] Nykiel, T., Potamias, M., Mishra, C., Kollios, G. and Koudas, N. 2010. *MRShare: sharing across multiple queries in MapReduce*. Proc. VLDB Endow. 3, 1-2 (Sep. 2010), 494–505.

Thank you for your attention !