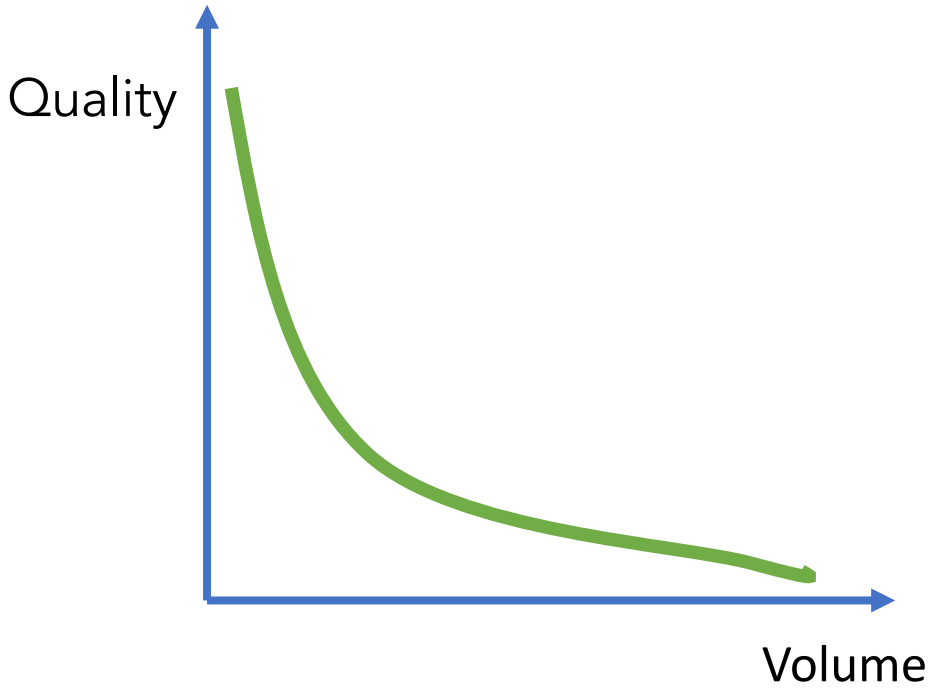


Balance-Aware Distributed String Similarity-Based Query Processing System

Ji Sun[+], Zeyuan Shang[*], Guoliang Li[+], Dong Deng[#], Zhifeng Bao[$]

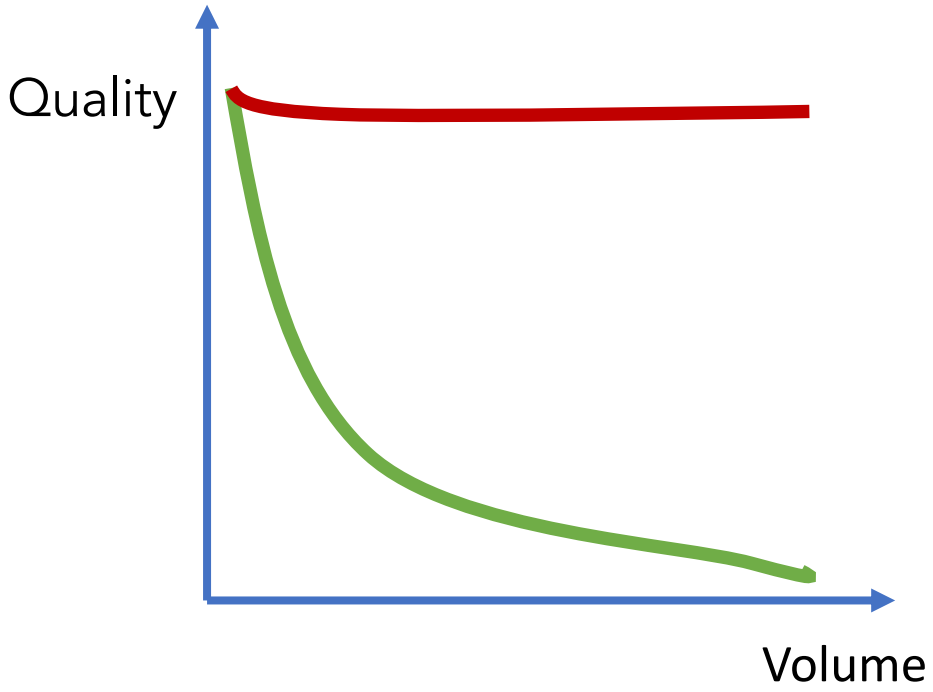
Tsinghua[+], MIT[*], Rutgers[#], RMIT[$]

Big Data Comes with Big Problem



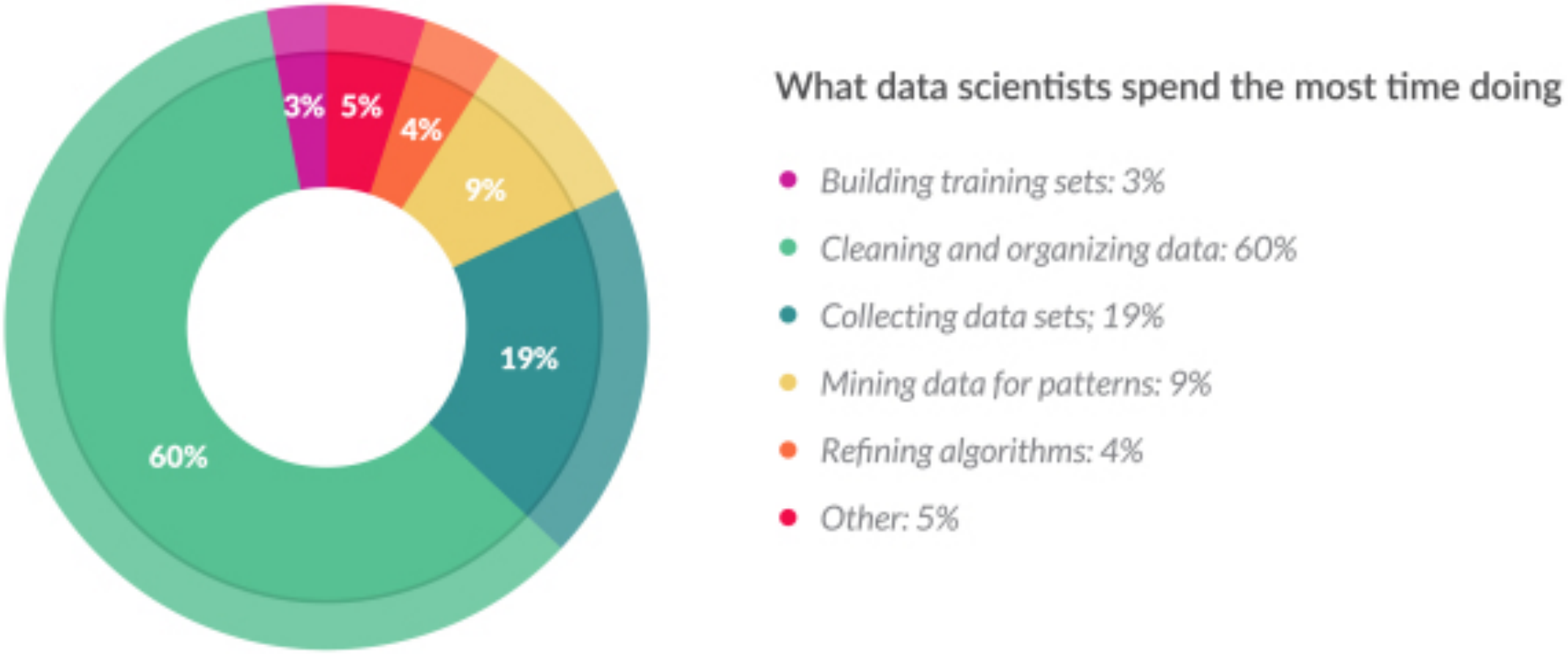
Reference: https://www.aibook.in/2018/12/data-quality-in-hindi.html

Cleaning and Integration becomes Necessary



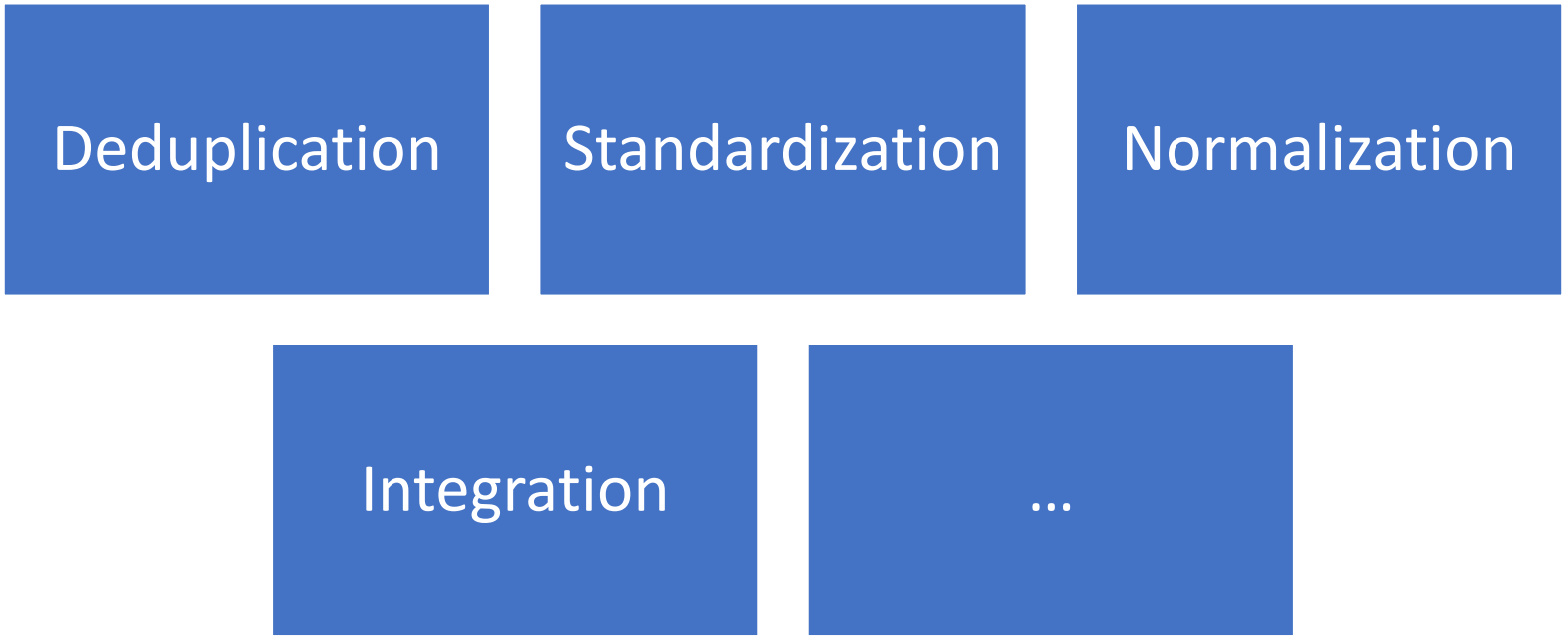
Reference: https://www.aibook.in/2018/12/data-quality-in-hindi.html

Cleaning and Integration is Expensive



Reference: https://heap.io/blog/data-stories/data-virtualization

String Similarity in Cleaning and Integration



Challenges of Large-Scale String Similarity

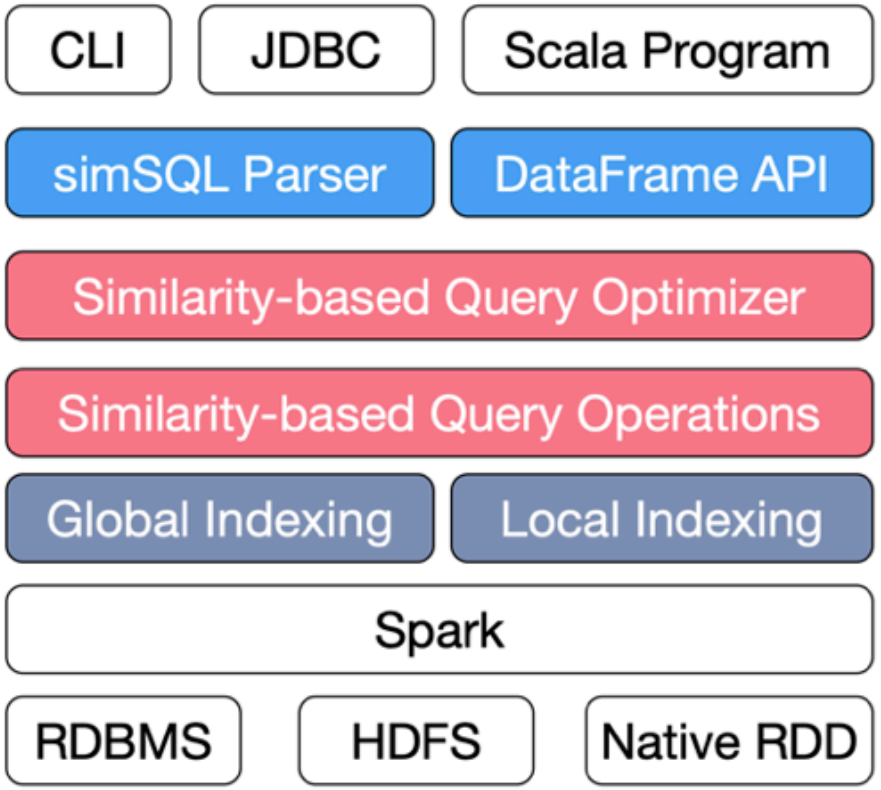
- Scalability
  - Serial algorithm is not efficient enough for large data
  - State-of-the-art parallel algorithms have **load balance** problem
- Flexibility
  - Hard to support different similarity functions
  - Operations not only include string similarity
- Easy-to-use interface (e.g., SQL or DataFrame)

Distributed-In-Memory Analytics for String Similarity

- Built over Spark
- Interfaces: SQL, DataFrame
- Supported Similarity Functions
  - Set-based Similarity Functions

$$\text{JAC}(r,s)=\frac{|r\cap s|}{|r\cup s|},\ \text{COS}(r,s)=\frac{|r\cap s|}{\sqrt{|r|\cdot|s|}},\ \text{DICE}(r,s)=\frac{2|r\cap s|}{|r|+|s|}$$

  - Character-based Similarity Functions
    - Edit distance
- Workflow



Overview of DIMA

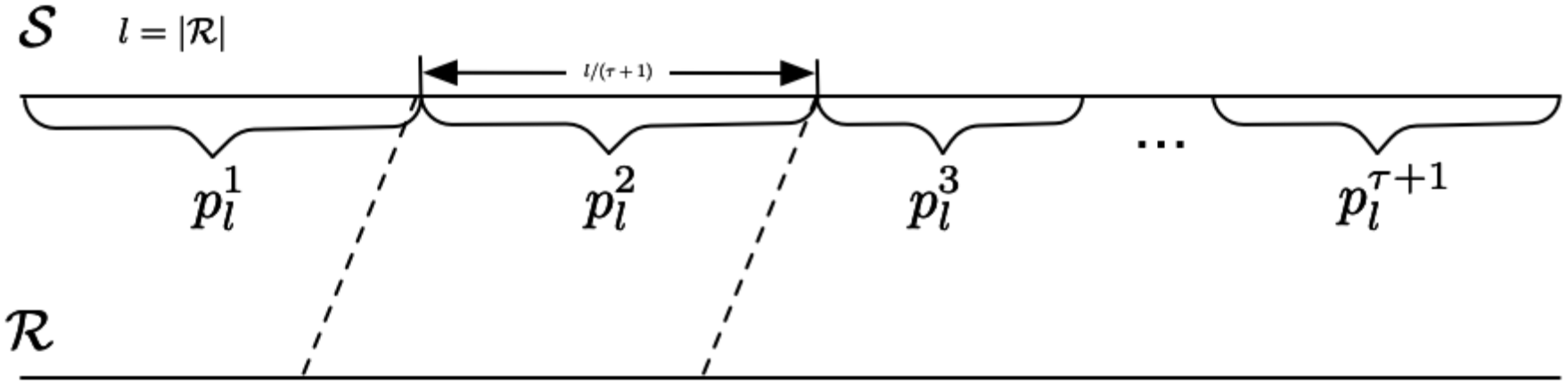
- Indexing: Global and Local index
- Balance-aware Search and Join
- Cost-based Optimizations
- Top-K Search and Join (not covered)

Overview of DIMA

- Indexing: Global and Local index
- Balance-aware Search and Join
- Cost-based Optimizations
- Top-K Search and Join (not covered)

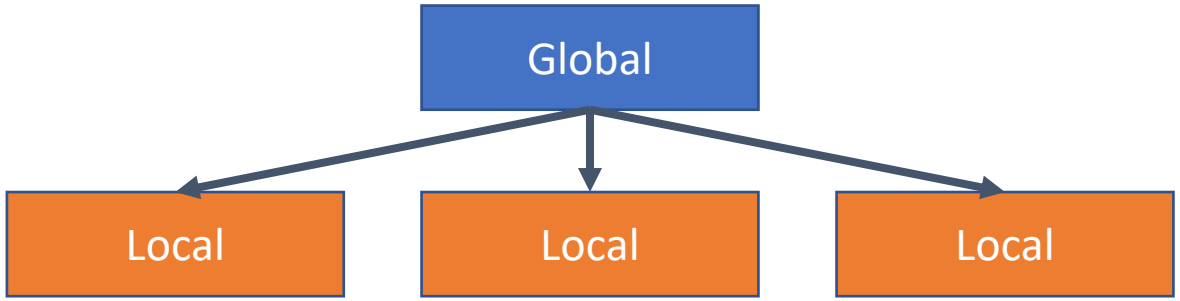
Segment-based Pruning

- Partition each string into **N** disjoint segments
- If two strings have more than **Θ** mis-matched (non-equal) segments, which means they have more than **Θ** mis-matched tokens
- **Θ** can be computed for each similarity function given the threshold **τ**, so if they have **Θ** mis-matched (non-equal) segments, they cannot be similar



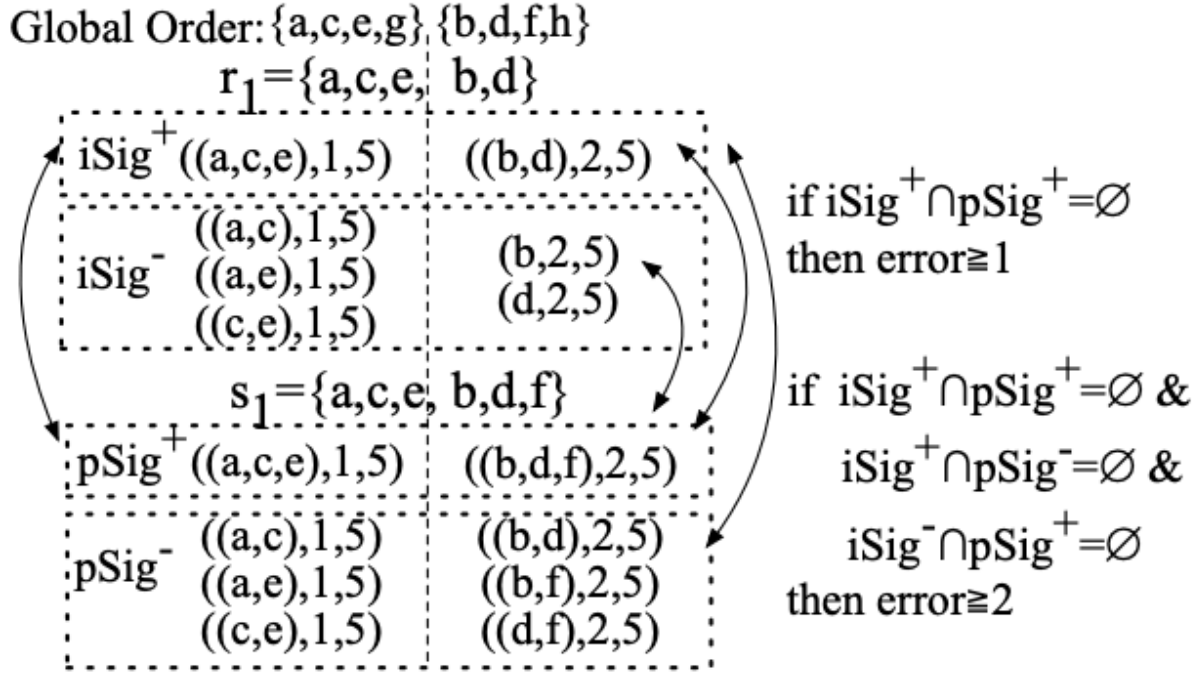
Two-layer Indexing

- Global Index (on each worker)
  - Mapping from the segment to the partitions containing this segment
  - Small so that it can be replicated to each worker
- Local Index (for each partition)
  - Segment and its inverted list
  - Reside in local memory



Deletion Segment

- Deletion segment: remove a token from the segment
  - Assume the original segment is {b, d, f}, then we will have a deletion segment {{b, d}, {b, f}, {d, f}}
- Derive similar lemma for number of mismatches of tokens, but **one** mismatch of segments means **two** mismatches of tokens.



Sig^+ : regular segment

Sig^- : deletion segment

Hybrid Pruning

- Regular Segment
- Deletion Segment
- We can compute the number of mismatched tokens based on mismatch of segments, to prune some candidate pairs.

- Length-based: lengths of two strings must be within some interval which can be derived from the given threshold

Overview of DIMA

- Indexing: Global and Local index
- **Balance-aware Search and Join**
- Cost-based Optimizations
- Top-K Search and Join (not covered)

Segment Generation

- Given a query, we compute its segments
- For each segment i, we can either (use Z[i] to denote) while $\sum Z[i] \geq n_l$ (n_l is the number of segments)
    - Don't use it (Z[i] = 0)
    - Use the regular segment (Z[i] = 1)
    - Use the deletion segment (Z[i] = 2)
- Based on the size of inverted list of each segment, we can compute the number of candidates (so this becomes an optimization problem to minimize it)

$$\sum_{i=1}^{\eta_l}(b_i \sum_{g \in \mathtt{pSig}^+_{s,l}} \mathcal{F}^+[g] + c_i(\sum_{g \in \mathtt{pSig}^-_{s,l}} \mathcal{F}^+[g] + \sum_{g \in \mathtt{pSig}^+_{s,l}} (\mathcal{F}^-[g] + \mathcal{F}^+[g])))$$

$$b_i = \begin{cases} 1 & Z[i] = 1 \\ 0 & Z[i] \neq 1 \end{cases} \quad c_i = \begin{cases} 1 & Z[i] = 2 \\ 0 & Z[i] \neq 2 \end{cases} \quad s.t. \sum_{i=1}^{\eta_l} Z[i] \geq \theta_{|s|,l}.$$

Balance-aware Segment Generation

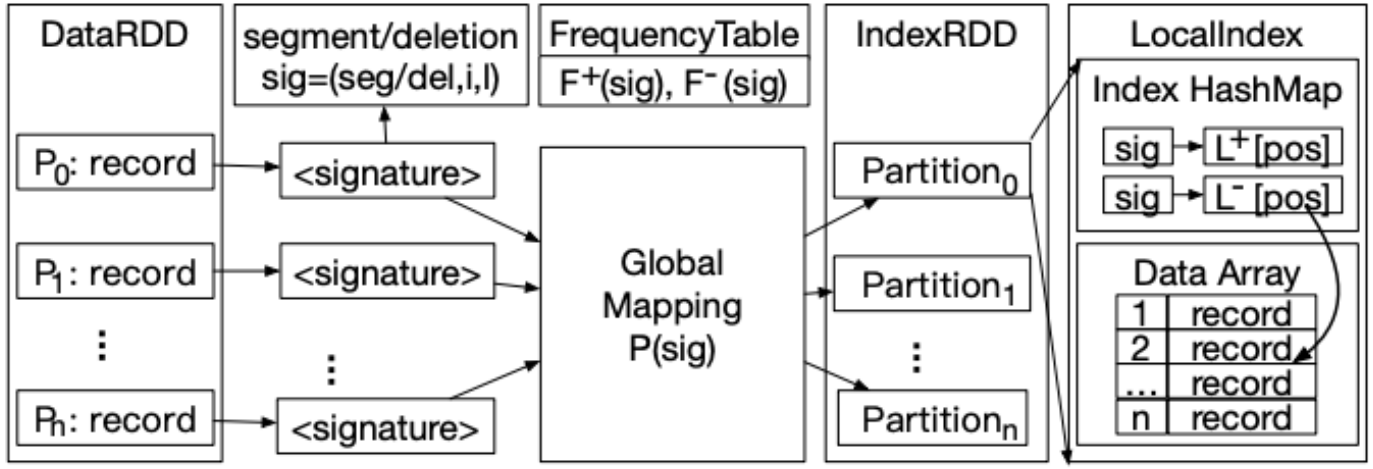
- Extend this optimization problem to a distributed setting
- Assume that there are **P** partitions, and each partition has a workload **W** (estimated by the number of candidates in the previous slide), we aim to minimize

$$\max(W_1, W_2, \ldots, W_P)$$

- This can be solved by dynamic programming

Search Procedure

- Generate segments for the given query using DP
- Global Search
  - Query the mappings to find relevant partitions
- Local Search
  - Query inverted list



Join Procedure

- Similar with search, but for many records (strings) now
- This becomes an NP-hard problem
- A greedy algorithm
  - Compute optimal segments for each record individually
  - Iterative algorithm
    - Compute the workload for each segment if we use it or use the deletion
    - Each round we take the segment with minimum workload, and we use corresponding segment strategy (use it directly or use the deletion segment), then the overall workload (the optimization goal) is decreased
    - Loop until all segments have been processed

Overview of DIMA

- Indexing: Global and Local index
- Balance-aware Search and Join
- Cost-based Optimizations
- Top-K Search and Join (not covered)

Cost-based Optimizations

- Cost model can be easily derived based on previous slides (based on the number of candidates)
- Select number of partitions
- Partition scheme (how to partition segments)

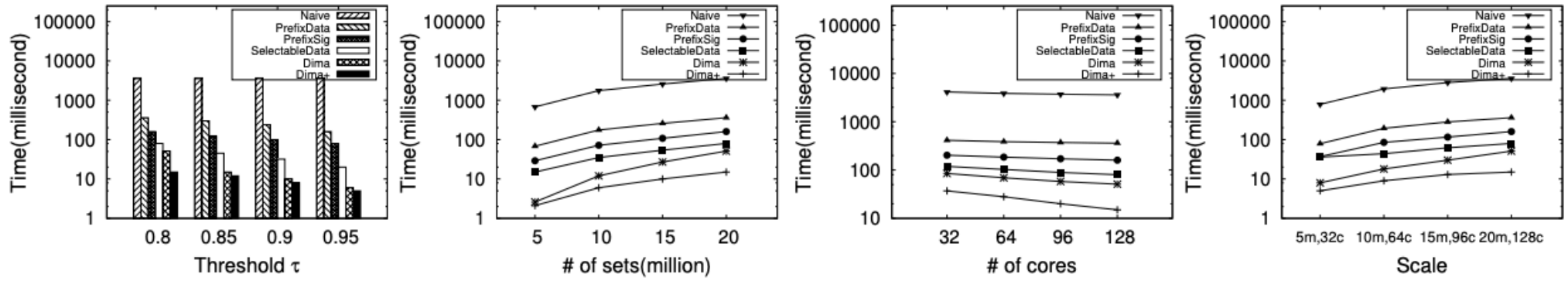
Experiments Setup

- Datasets

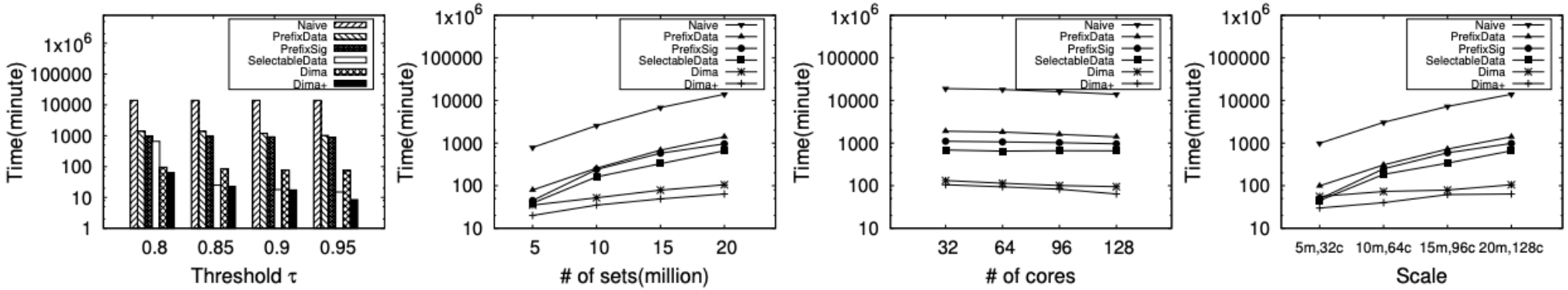
Datasets	Cardinality	AvgLen	MinLen	MaxLen	Size
Tweet	20,000,000	15.52	5	32	2.9GB
Review	20,000,000	10.3	1	399	1.5GB
ReviewBig	100,000,000	11.2	1	399	7.7GB
Dbpedia	20,000,000	18.43	1	61	1.5GB
Webtable	20,000,000	9.44	4	145	1.1GB
WebtableBig	100,000,000	10.57	4	148	5.5GB

- Jaccard similarity function (refer to paper for other functions)
- Baselines
  - Map-Reduce implementations
  - Partitioned-by-data methods
- 64-node cluster while each node has a 8-core Intel Xeon E5-2670 v3 2.30GHz and 48 GB RAM

Efficient Search and Join

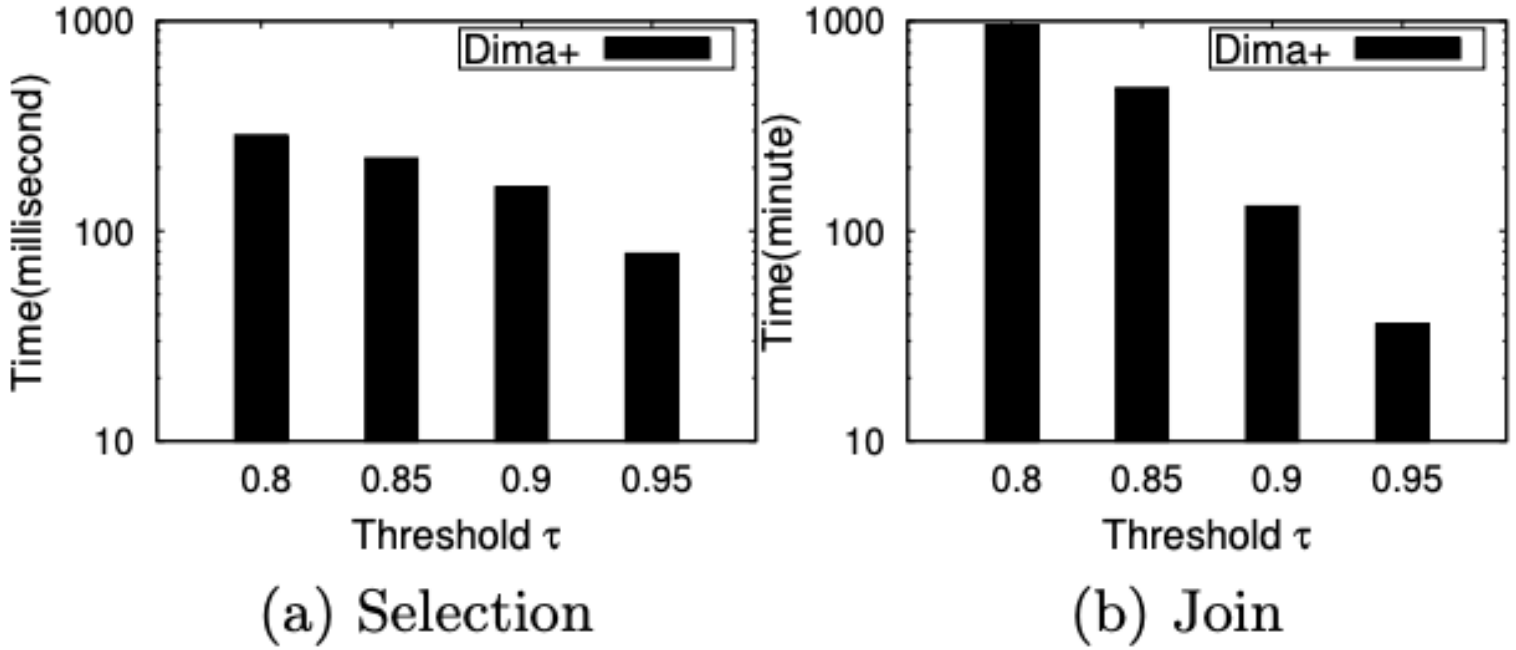


(a) Varying τ: `Tweet` (b) Scalability: `Tweet` (c) Scale-up: `Tweet` (d) Scale-out: `Tweet`



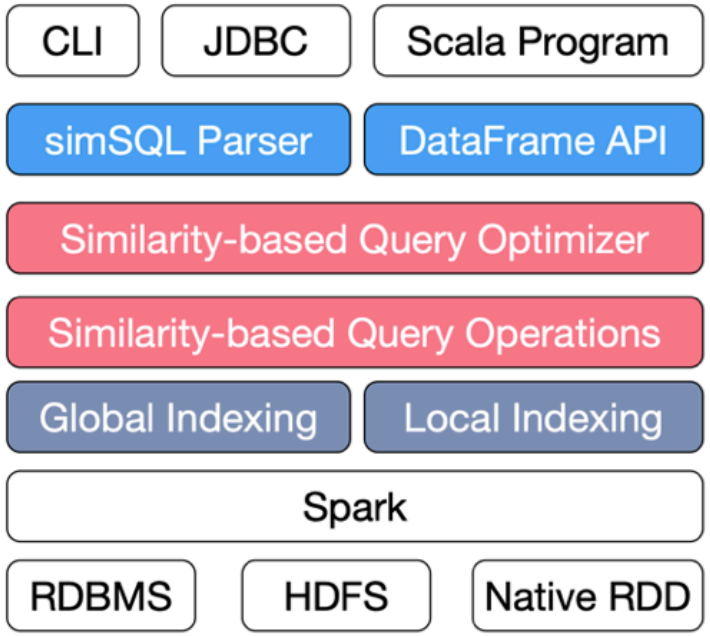
(a) Varying τ: `Tweet` (b) Scalability: `Tweet` (c) Scale-up: `Tweet` (d) Scale-out: `Tweet`

Evaluation on Big Dataset (100M records)



Zeyuan Shang
zeyuans@mit.edu

- DIMA: Distributed In-Memory Analytics for String Similarity Query
- Methods
  - Global and Local Indexing
  - Balance-aware Search & Join
  - Cost-based Optimizations
- Outperform state-of-the-art studies by several orders of magnitude with good scalability



Special Thanks to:

NSFC National Natural Science Foundation of China

HUAWEI

TAL 好未来