Efficient Parallel Partition based Algorithms for Similarity Search and Join with Edit Distance Constraints

Yu Jiang, Dong Deng, Jiannan Wang, Guoliang Li, and Jianhua Feng

Tsinghua University

Similarity Search&Join Competition on EDBT/ICDT 2013

Outline

1. Motivation
   - Problem Definition
   - Application
2. Our Approach
   - Pass Join Algorithm
   - Additional Filters
   - Parallel
3. Experiment
   - Evaluating Pruning Techniques
   - Evaluating Parallelism
   - Evaluating Scalability

Problem Definition

String Similarity Joins

Given a set of strings $\mathcal{S}$, the task is to find all pairs of τ-similar strings from $\mathcal{S}$. A program must output all matches with both string identifiers and distance τ.(Track II)

An Example

Table: A string dataset

ID	Strings	Length
s_1	vankatesh	9
s_2	avataresha	10
s_3	kaushic chaduri	15
s_4	kaushik chakrab	15
s_5	kaushuk chadhui	15
s_6	caushik chakrabar	17

Consider the string dataset in Table 1.
Suppose $\tau = 3$. $\langle s_4, s_6 \rangle$ is a similar pair as $ED(s_4, s_6) \leq \tau$

Application

- Data cleaning
- Information Extraction
- Comparison of biological sequences
- ...

Basic Idea

Lemma

Given a string r with $\tau + 1$ segments and a string s, if s is similar to r within threshold τ, s must contain a segment of r.

Example

$\tau = 1$, $r =$"EDBT" has two segments "ED" and "BT". $s =$"ICDT" cannot similar to r as s contains none of the two segemtns.

Even Partition Scheme

Definition

In even partition scheme, each segment has almost the same length. ($\lfloor \frac{|s|}{\tau+1} \rfloor$ or $\lceil \frac{|s|}{\tau+1} \rceil$)

Example

$\tau = 3$, we partition $s_1 =$"*vankatesh*" into four segments "*va*", "*nk*", "*at*", "*esh*".

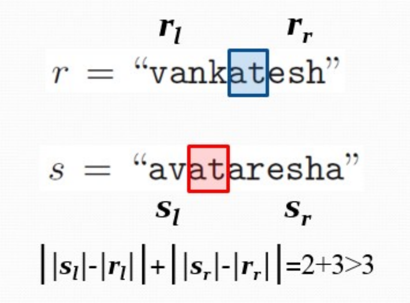
Substring Selection
Basic Methods

- Enumeration:
  Enumerate all substrings for each of the segment.
- Length-based:
  For each segment, only select substrings with same length.
- Shift-based:
  For segment with start position p_i, select substrings with start position in $[p_i - \tau, p_i + \tau]$

Substring Selection

Position-aware Substring Selection

Observation

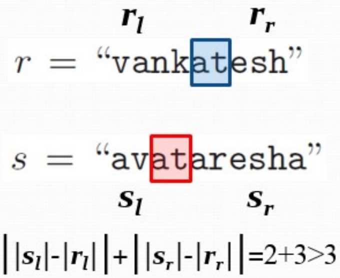


Theorem (Position-aware Substring Selection)

For segment with start position p_i, select substrings with start position in $[p_i - \lfloor\frac{\tau-\triangle}{2}\rfloor, p_i + \lfloor\frac{\tau+\triangle}{2}\rfloor]$ where $\triangle = |s| - |r|$.

Substring Selection
Position-aware Substring Selection

Observation



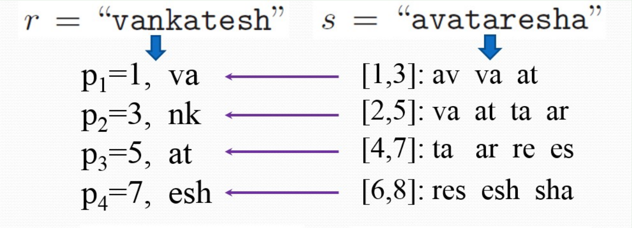
Theorem (Position-aware Substring Selection)

For segment with start position p_i, select substrings with start position in $[p_i - \lfloor\frac{\tau-\triangle}{2}\rfloor, p_i + \lfloor\frac{\tau+\triangle}{2}\rfloor]$ where $\triangle = |s| - |r|$.

Substring Selection

Position-aware Substring Selection

Example

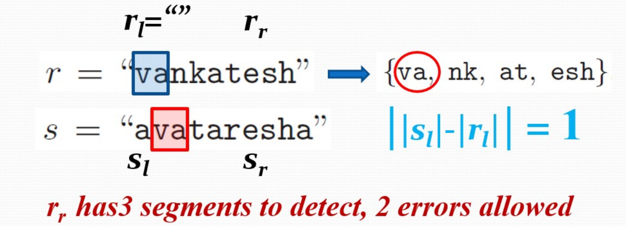


$\tau = 3, \triangle = 1, [p_i - \lfloor \frac{\tau - \triangle}{2} \rfloor, p_i + \lfloor \frac{\tau + \triangle}{2} \rfloor] = [p_i - 1, p_i + 2]$

Substring Selection

Multi-match-aware Substring Selection

Observation



There must be another matching between r_r and s_r.

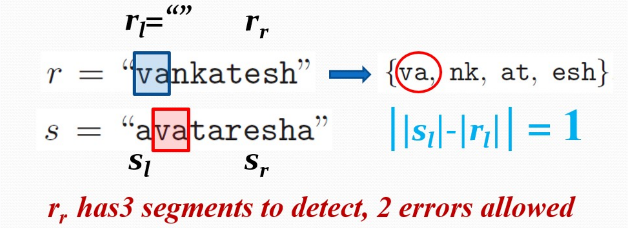
Theorem (Multi-match-aware Substring Selection)

For the i-th segment with start position p_i, select substrings within $[p_i - i, p_i + i] \cap [p_i + \triangle - (\tau + 1 - i), p_i + \triangle + (\tau + 1 - i)]$.

Substring Selection

Multi-match-aware Substring Selection

Observation



There must be another matching between r_r and s_r.

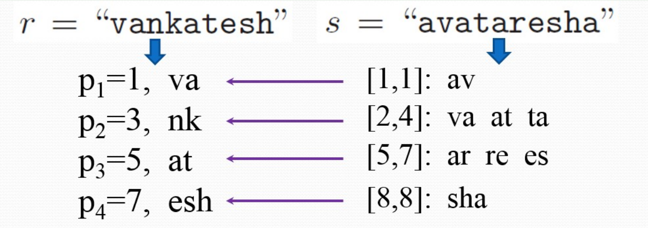
Theorem (Multi-match-aware Substring Selection)

For the i-th segment with start position p_i, select substrings within $[p_i - i, p_i + i] \cap [p_i + \triangle - (\tau + 1 - i), p_i + \triangle + (\tau + 1 - i)]$.

Substring Selection

Multi-match-aware Substring Selection

Example

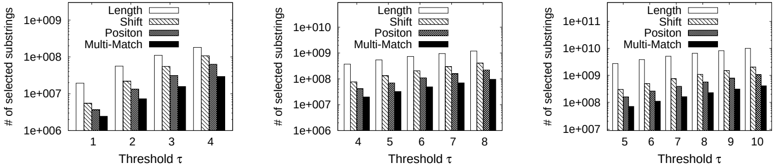


Substring Selection

Theoretical Results

1. The number of selected substrings by the multi-match-aware method is minimum.
2. For strings longer than $2 * (\tau + 1)$, our selection method is the only way to select minimum number of substrings.

Substring Selection
Experimental Results



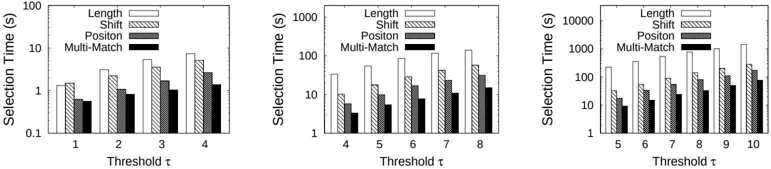
(a) Author Name (Avg Len = 15)

(b) Query Log (Avg Len = 45)

(c) Author+Title (Avg Len = 105)

Figure: Numbers of selected substrings

Substring Selection
Experimental Results



(a) Author Name (Avg Len = 15)

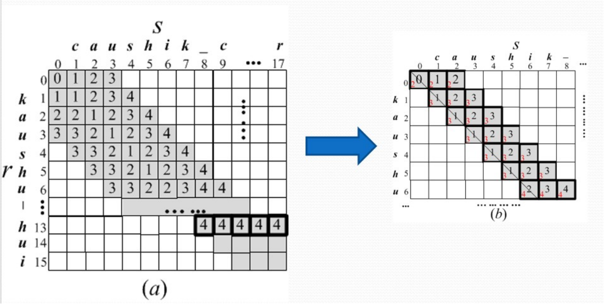
(b) Query Log (Avg Len = 45)

(c) Author+Title (Avg Len = 105)

Figure: Elapsed time for generating substrings

Verification

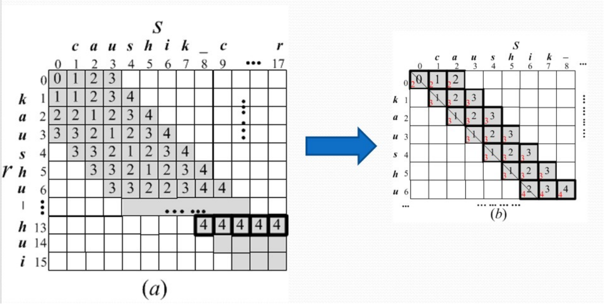
Length-aware Verification



- Inspired by the position-aware substring selection.
- Save at least half computation than traditional dynamic method.
- Save even more using improved early termination.

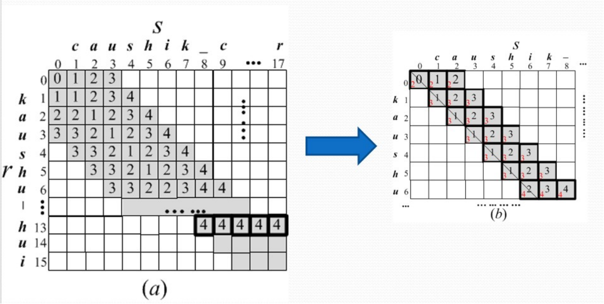
Verification

Length-aware Verification



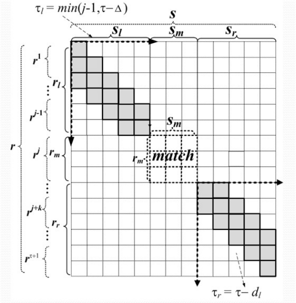
- Inspired by the position-aware substring selection.
- Save at least half computation than traditional dynamic method.
- Save even more using improved early termination.

Verification
Length-aware Verification



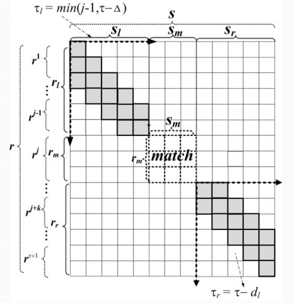
- Inspired by the position-aware substring selection.
- Save at least half computation than traditional dynamic method.
- Save even more using improved early termination.

Verification
Extension-based Verification



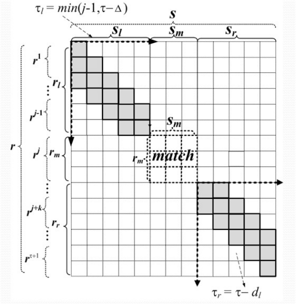
- Inspired by the multi-match-aware substring selection.
- Using tighter thresholds to verify the candidate pairs.
- Verify if $ED(r_r, s_r) \leq \tau + 1 - i$ and $ED(r_l, s_l) \leq i - 1$.

Verification
Extension-based Verification



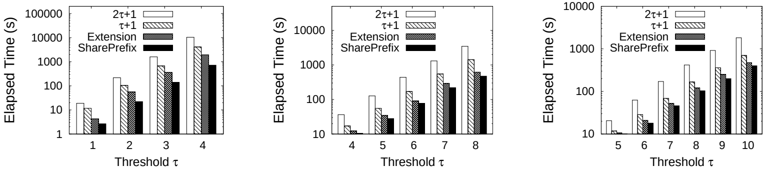
- Inspired by the multi-match-aware substring selection.
- Using tighter thresholds to verify the candidate pairs.
- Verify if $ED(r_r, s_r) \le \tau + 1 - i$ and $ED(r_l, s_l) \le i - 1$.

Verification
Extension-based Verification



- Inspired by the multi-match-aware substring selection.
- Using tighter thresholds to verify the candidate pairs.
- Verify if $\text{ED}(r_r, s_r) \leq \tau + 1 - i$ and $ED(r_l, s_l) \leq i - 1$.

Verification
Experimental Results



(a) Author Name (Avg Len 15)

(b) Query Log (Avg Len 45)

(c) Author+Title (Avg Len 105)

Figure: Elapsed time for verification

Additional Filters
Effective Indexing Strategy

- Partition longer strings into segments.
- Select substrings from shorter strings.
- Longer segments decrease the possibility of matching.
- Thus decrease the number of candidates.

Additional Filters

Effective Indexing Strategy

- Partition longer strings into segments.
- Select substrings from shorter strings.
- Longer segments decrease the possibility of matching.
- Thus decrease the number of candidates.

Additional Filters
Effective Indexing Strategy

- Partition longer strings into segments.
- Select substrings from shorter strings.
- Longer segments decrease the possibility of matching.
- Thus decrease the number of candidates.

Additional Filters

Effective Indexing Strategy

- Partition longer strings into segments.
- Select substrings from shorter strings.
- Longer segments decrease the possibility of matching.
- Thus decrease the number of candidates.

Additional Filters

Content Filter

Observation

- *Let $\mathcal{H}_r$ denote the character frequency vector of r.*
- $r =$"abyyyy", $s =$"axxyyyxy".
  $\mathcal{H}_r = \{\{a, 1\}, \{b, 1\}, \{y, 4\}\}$, $\mathcal{H}_s = \{\{a, 1\}, \{x, 3\}, \{y, 4\}\}$
- *Let $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s|$.*
- $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s| = ||1| + |-3|| = 4$.
- *A deletion or insertion changes $\mathcal{H}_\triangle$ by 1 at most.*
- *An substitution changes $\mathcal{H}_\triangle$ by 2 at most.*

Additional Filters

Content Filter

Observation

- *Let $\mathcal{H}_r$ denote the character frequency vector of r.*
- $r =$"abyyyy", $s =$"axxyyyxy".
  $\mathcal{H}_r = \{\{a, 1\}, \{b, 1\}, \{y, 4\}\}$, $\mathcal{H}_s = \{\{a, 1\}, \{x, 3\}, \{y, 4\}\}$
- *Let $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s|$.*
- $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s| = ||1| + | - 3|| = 4$.
- *A deletion or insertion changes $\mathcal{H}_\triangle$ by 1 at most.*
- *An substitution changes $\mathcal{H}_\triangle$ by 2 at most.*

Additional Filters

Content Filter

Observation

- *Let $\mathcal{H}_r$ denote the character frequency vector of r.*
- $r =$"abyyyy", $s =$"axxyyyxy".
  $\mathcal{H}_r = \{\{a, 1\}, \{b, 1\}, \{y, 4\}\}$, $\mathcal{H}_s = \{\{a, 1\}, \{x, 3\}, \{y, 4\}\}$
- *Let $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s|$.*
- $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s| = ||1| + |-3|| = 4.$
- *A deletion or insertion changes $\mathcal{H}_\triangle$ by 1 at most.*
- *An substitution changes $\mathcal{H}_\triangle$ by 2 at most.*

Additional Filters
Content Filter

Observation

- *Let $\mathcal{H}_r$ denote the character frequency vector of r.*
- $r =$"abyyyy", $s =$"axxyyyxy".
  $\mathcal{H}_r = \{\{a, 1\}, \{b, 1\}, \{y, 4\}\}$, $\mathcal{H}_s = \{\{a, 1\}, \{x, 3\}, \{y, 4\}\}$
- *Let $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s|$.*
- $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s| = ||1| + |-3|| = 4.$
- *A deletion or insertion changes $\mathcal{H}_\triangle$ by 1 at most.*
- *An substitution changes $\mathcal{H}_\triangle$ by 2 at most.*

Additional Filters
Content Filter

Observation

- *At most τ edit operations, $\mathcal{H}_\triangle \leq 2\tau$.*
- *At most $\tau - ||r| - |s||$ substitutions, $\mathcal{H}_\triangle \leq 2\tau - ||r| - |s||$.*
- *Group symbols to improve the content-filter running time.*
- *Integrate the content filter with the extension-based verification.*

Additional Filters

Content Filter

Observation

- *At most τ edit operations, $\mathcal{H}_\triangle \leq 2\tau$.*
- *At most $\tau - \big||r| - |s|\big|$ substitutions, $\mathcal{H}_\triangle \leq 2\tau - \big||r| - |s|\big|$.*
- *Group symbols to improve the content-filter running time.*
- *Integrate the content filter with the extension-based verification.*

Additional Filters

Content Filter

Observation

- *At most τ edit operations, $\mathcal{H}_{\triangle} \leq 2\tau$.*
- *At most $\tau - \big||r| - |s|\big|$ substitutions, $\mathcal{H}_{\triangle} \leq 2\tau - \big||r| - |s|\big|$.*
- *Group symbols to improve the content-filter running time.*
- *Integrate the content filter with the extension-based verification.*

Additional Filters
Content Filter

Observation

- *At most τ edit operations, $\mathcal{H}_{\triangle} \leq 2\tau$.*
- *At most $\tau - \big||r| - |s|\big|$ substitutions, $\mathcal{H}_{\triangle} \leq 2\tau - \big||r| - |s|\big|$.*
- *Group symbols to improve the content-filter running time.*
- *Integrate the content filter with the extension-based verification.*

1. Parallel Sorting. Group strings by lengths using existing parallel algorithm.
2. Parallel Building Indexes. Parallel building indexes for each group.
3. Parallel Joins. Parallel perform similarity joins on each groups.

Experiment Setup

Table: Datasets

Datasets	cardinality	average len	max len	min len
GeoNames	400,000	11.106	1	60
GeoNames Query	100,000	10.7	2	43
Reads	750,000	101.388	86	106
Reads Query	100,000	101.2	88	116

Experiment Setup

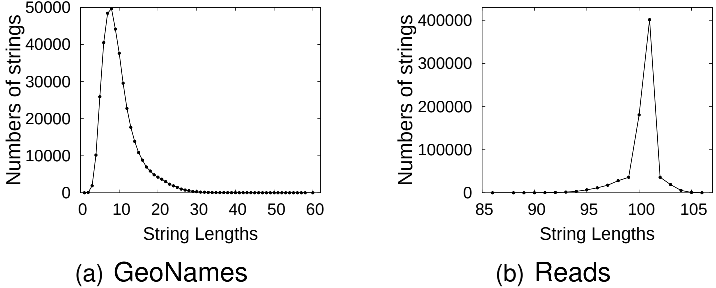


Figure: Length Distribution.

Evaluating Pruning Techniques

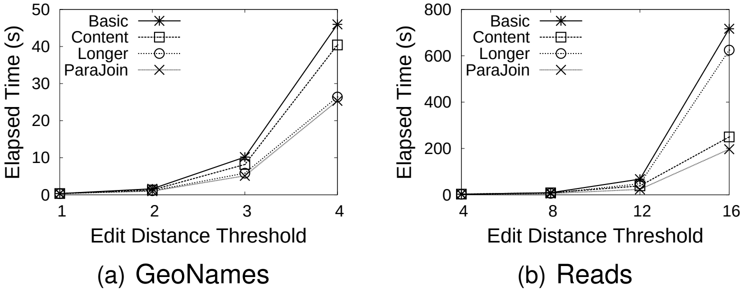


Figure: Evaluating pruning techniques for similarity joins(8 threads).

Evaluating Pruning Techniques

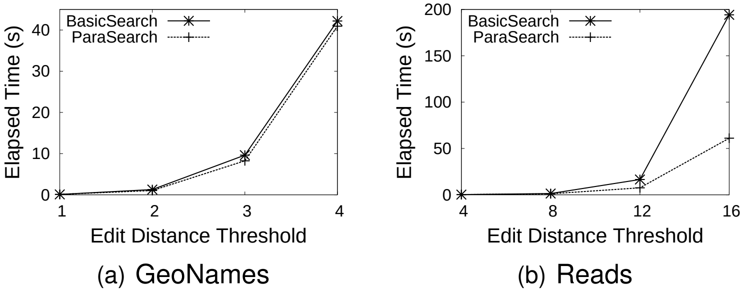


Figure: Evaluating pruning techniques for similarity search(8 threads).

Evaluating Parallelism

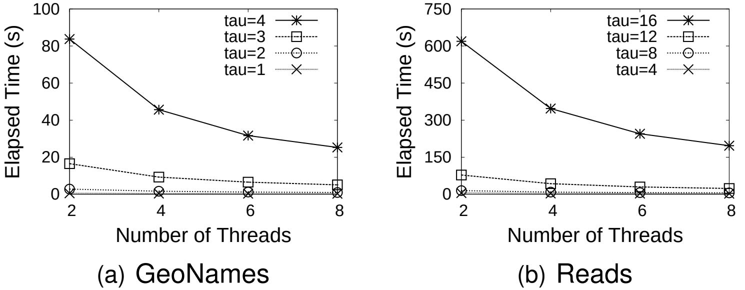


Figure: Evaluating running time of similarity join by varying number of threads.

Evaluating Speedup

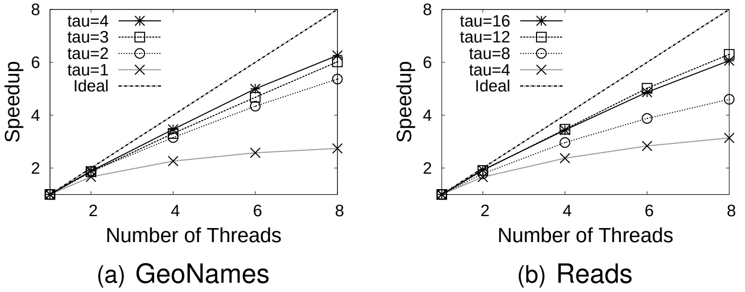
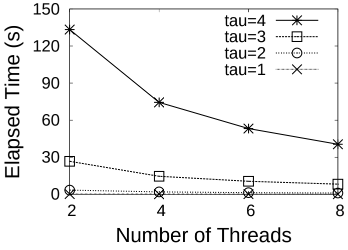
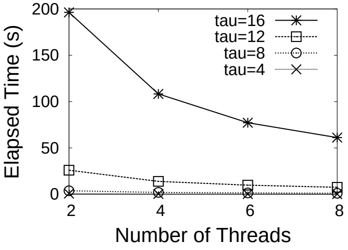


Figure: Evaluating speedup of similarity join.

Evaluating Parallelism



(a) GeoNames



(b) Reads

Figure: Evaluating running time of similarity search by varying number of threads.

Evaluating Speedup

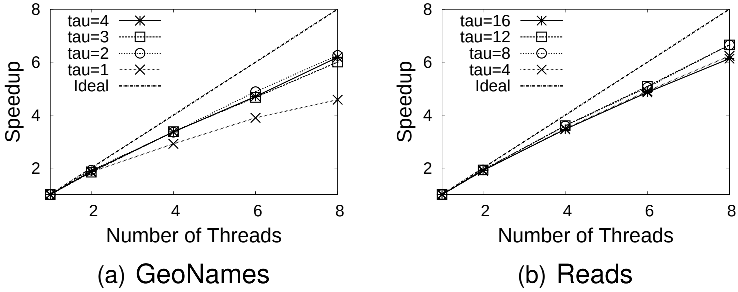
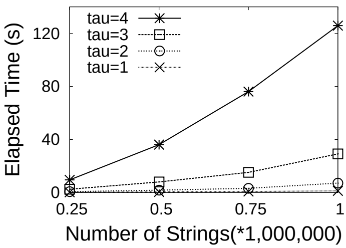
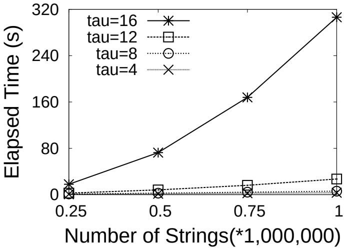


Figure: Evaluating speedup of similarity search.

Evaluating Scalability



(a) GeoNames



(b) Reads

Figure: Evaluating the scalability of the similarity join algorithm(8 threads).

Evaluating Scalability

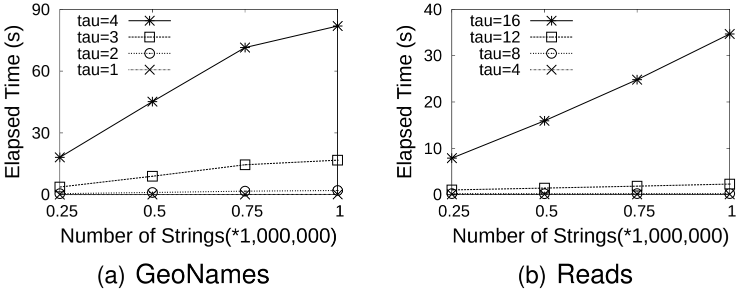
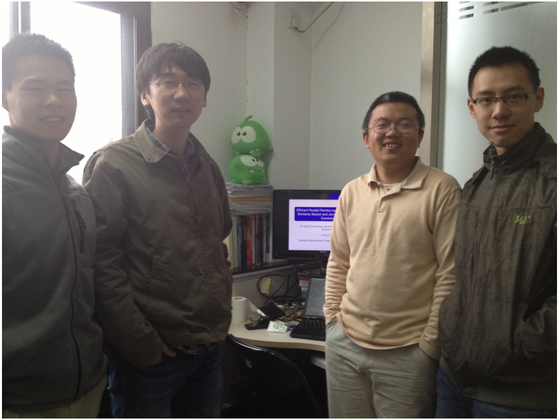


Figure: Evaluating the scalability of the similarity search algorithm(8 threads).

About our team I

- We are from Tsinghua University, Beijing, China.
- Yu Jiang, Jiannan Wang, Guoliang Li, Jianhua Feng and Dong Deng.

About our team II



Thank You
Q & A

http://dbgroup.cs.tsinghua.edu.cn/dd

Pass-Join: A Partition based Method for Similarity Joins. Guoliang Li, Dong Deng, Jiannan Wang, Jianhua Feng. VLDB 2012.