# Cache-Aware Parallel Approximate String Search and Join Algorithms Using BWT

Jiaying Wang, Xiaochun Yang, and Bin Wang

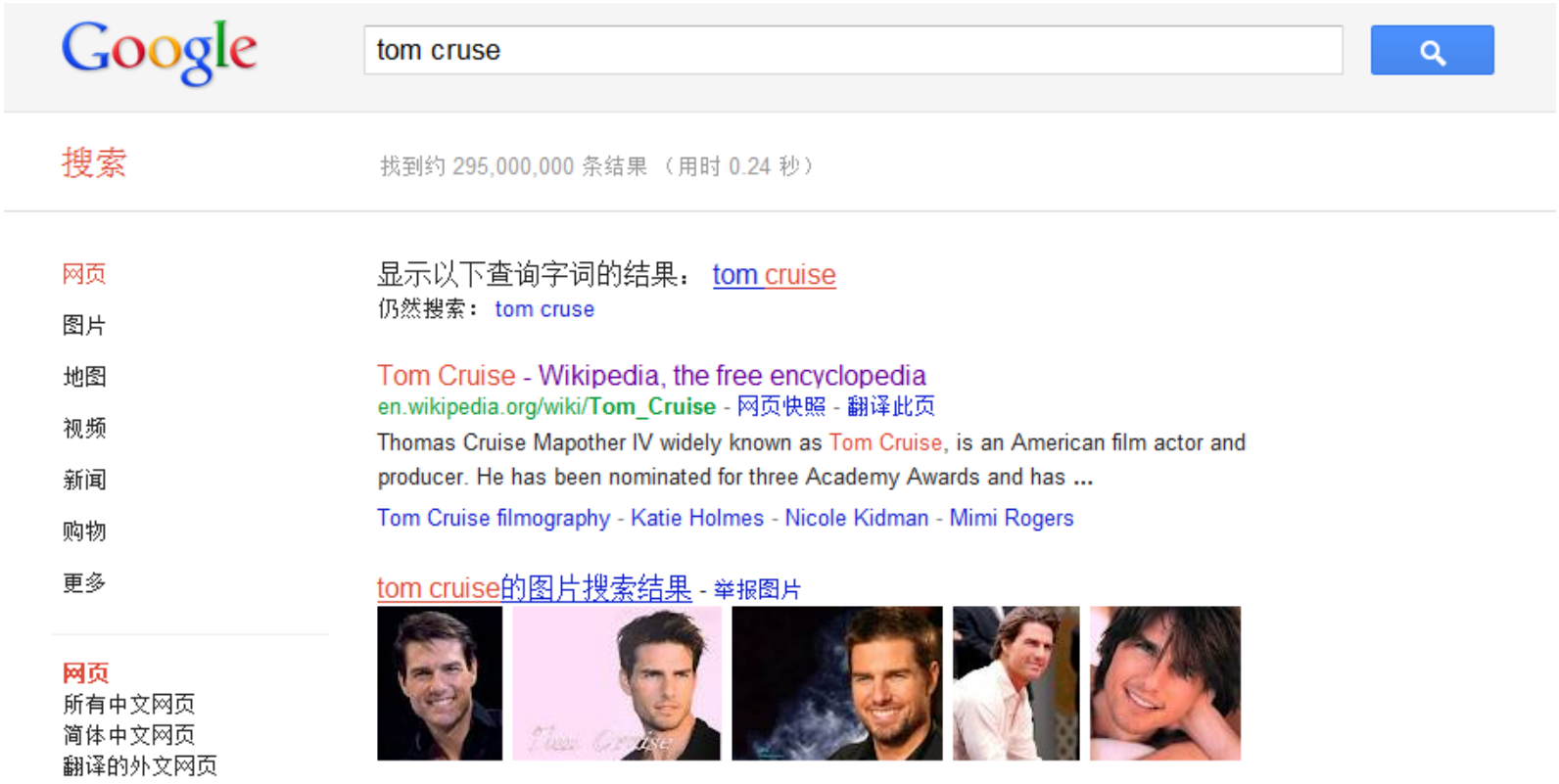
Mar 22, 2013


Northeastern University, China

# Outline

- Motivation
- Problem statement
- Approximate search method
  - Cache aware parallel framework
  - Pruning technique
  - Multi query optimization
  - Look ahead verification
- Approximate join method
- Experiment
- Conclusion

# Motivation


Google
tom cruse
搜索
找到约 295,000,000 条结果 （用时 0.24 秒）
网页
图片
地图
视频
新闻
购物
更多
网页
所有中文网页
简体中文网页
翻译的外文网页
显示以下查询字词的结果： tom cruise
仍然搜索： tom cruse
Tom Cruise - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Tom_Cruise - 网页快照 - 翻译此页
Thomas Cruise Mapother IV widely known as Tom Cruise, is an American film actor and producer. He has been nominated for three Academy Awards and has ...
Tom Cruise filmography - Katie Holmes - Nicole Kidman - Mimi Rogers
tom cruise的图片搜索结果 - 举报图片

# Motivation

- Searching dna sequence similar to "ACGTACAATATTAG" in genome database.
- Results:

ACGTACAATATTAG is similar to

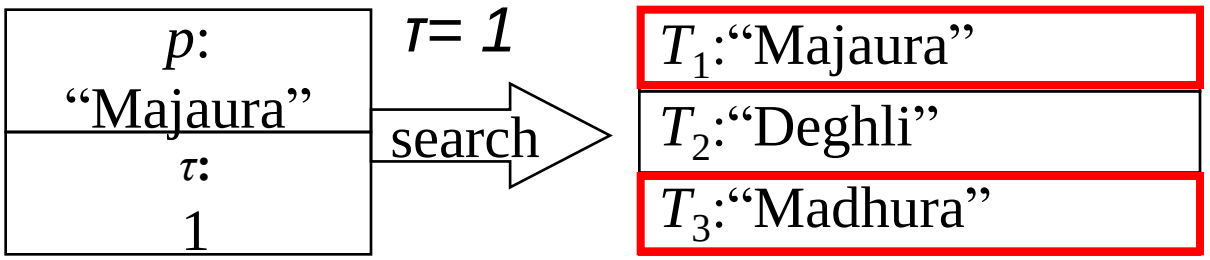
...ACGTACATTATTAG...

...ACGTAAAATATTAG...

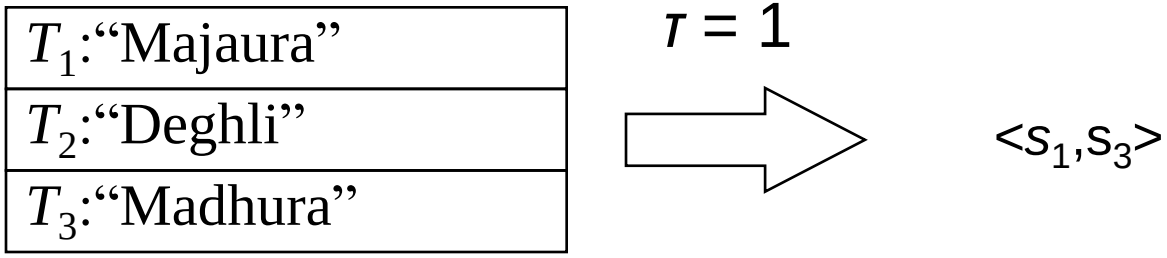
...ACGTACAAATTTAG...

# Problem Statement

- We have a string set $C$, for a given pattern $p$ and threshold $\tau$, return all answer $T_i \in C$ and $ED(p, T_i) \leq \tau$




- Find all pairs $(T_1, T_2) \in C \times C$ that $ed(T_1, T_2) \leq \tau$ as fast as possible.



# Current famous solutions

- Metric space-base method
- Signature based (q-gram, q-chunk+ q-gram)
  - Idea:

    Filter (fully filter, prefix filter) +Verify

    if s is similar to q, some (lower bound, LB) part (signature) of s and q must be identical.

  #gram = $|p|-q+1$

  LB = #gram $-q\times\tau$( prefix, PF = $q\times\tau+1$)

- Tree/Trie

# BWTPA Index

| |
|---|
| $T_1$:“Majaura” |
| $T_2$:“Deghli” |
| $T_3$:“Madhura” |

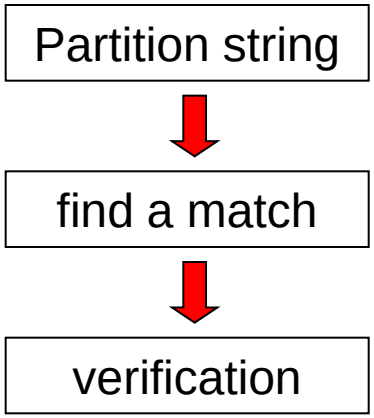
BWTPA index contains

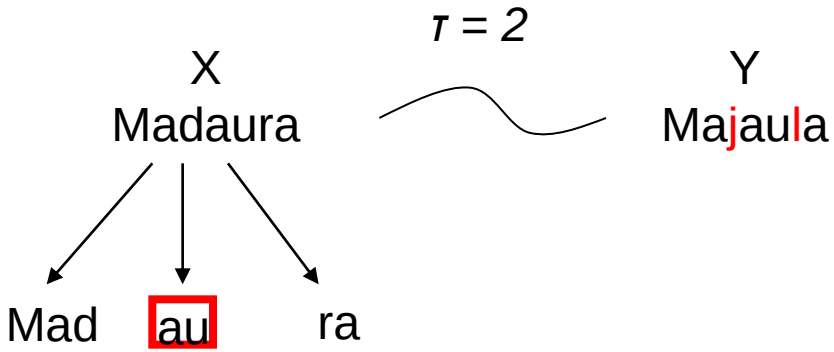
BWT:
Simulate suffix array (SA)
PA:
Record id of SA

| | | L | SA | PA |
|---|---|---|---|---|
| 1 | #Majaura$Deghli$Madhur | a | 22 | 3 |
| 2 | $Deghli$Madhura#Majaur | a | 7 | 1 |
| 3 | $Madhura#Majaura$Deghl | i | 14 | 2 |
| 4 | Deghli$Madhura#Majaura | $ | 8 | 2 |
| 5 | Madhura#Majaura$Deghli | $ | 15 | 3 |
| 6 | Majaura$Deghli$Madhura | # | 0 | 1 |
| 7 | a#Majaura$Deghli$Madhu | r | 21 | 3 |
| 8 | a$Deghli$Madhura#Majau | r | 6 | 1 |
| 9 | adhura#Majaura$Deghli$ | M | 16 | 3 |
| 10 | ajaura$Deghli$Madhura# | M | 1 | 1 |
| 11 | aura$Deghli$Madhura#Ma | j | 3 | 1 |
| 12 | dhura#Majaura$Deghli$M | a | 17 | 3 |
| 13 | eghli$Madhura#Majaura$ | D | 9 | 2 |
| 14 | ghli$Madhura#Majaura$D | e | 10 | 2 |
| 15 | hli$Madhura#Majaura$De | g | 11 | 2 |
| 16 | hura#Majaura$Deghli$Ma | d | 18 | 3 |
| 17 | i$Madhura#Majaura$Degh | l | 13 | 2 |
| 18 | jaura$Deghli$Madhura#M | a | 2 | 1 |
| 19 | li$Madhura#Majaura$Deg | h | 12 | 2 |
| 20 | ra#Majaura$Deghli$Madh | u | 20 | 3 |
| 21 | ra$Deghli$Madhura#Maja | u | 5 | 1 |
| 22 | ura#Majaura$Deghli$Mad | h | 19 | 3 |
| 23 | ura$Deghli$Madhura#Maj | a | 4 | 1 |

# Approximate String Search

- If a string x is similar to a string y, then a segment of x will match a substring of y exactly.
- Partition $p$ to $\tau+1$ same (almost) length partitions
- $r = p \% (\tau+1)$
- First $r$ partitions' length is $p/(\tau+1)+1$
- The left $\tau+1-r$ partitions will be $p/(\tau+1)$



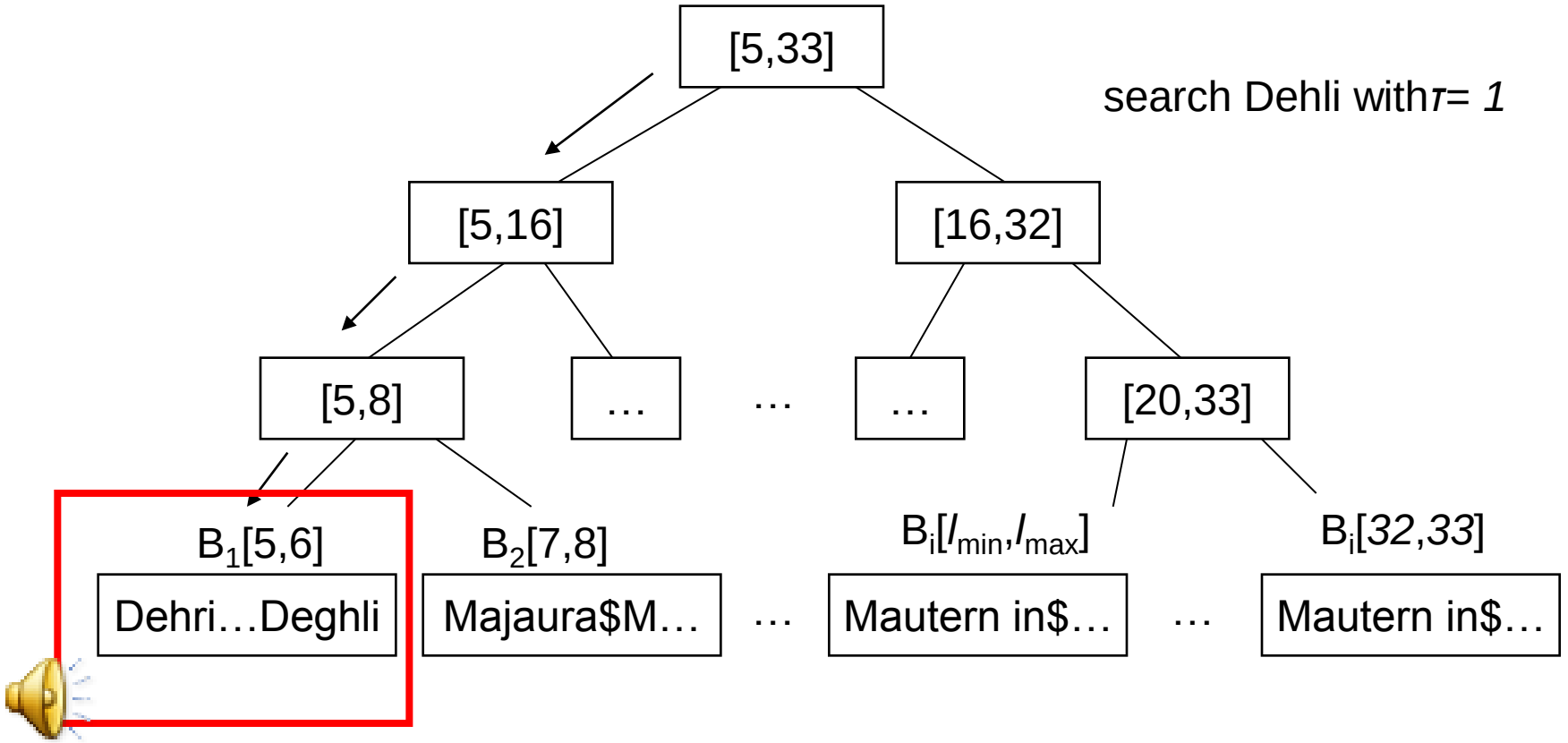
# Find Exact Substring

| | | L | SA | PA |
|---|---|---|---|---|
| 1 | #Majaura$Deghli$Madhur | a | 22 | 3 |
| 2 | $Deghli$Madhura#Majaur | a | 7 | 1 |
| 3 | $Madhura#Majaura$Deghl | i | 14 | 2 |
| 4 | Deghli$Madhura#Majaura | $ | 8 | 2 |
| 5 | Madhura#Majaura$Deghli | $ | 15 | 3 |
| 6 | Majaura$Deghli$Madhura | # | 0 | 1 |
| 7 | a#Majaura$Deghli$Madhu | r | 21 | 3 |
| 8 | a$Deghli$Madhura#Majau | r | 6 | 1 |
| 9 | adhura#Majaura$Deghli$ | M | 16 | 3 |
| 10 | ajaura$Deghli$Madhura# | M | 1 | 1 |
| 11 | aura$Deghli$Madhura#Ma | j | 3 | 1 |
| 12 | dhura#Majaura$Deghli$M | a | 17 | 3 |
| 13 | eghli$Madhura#Majaura$ | D | 9 | 2 |
| 14 | ghli$Madhura#Majaura$D | e | 10 | 2 |
| 15 | hli$Madhura#Majaura$De | g | 11 | 2 |
| 16 | hura#Majaura$Deghli$Ma | d | 18 | 3 |
| 17 | i$Madhura#Majaura$Degh | l | 13 | 2 |
| 18 | jaura$Deghli$Madhura#M | a | 2 | 1 |
| 19 | li$Madhura#Majaura$Deg | h | 12 | 2 |
| 20 | ra#Majaura$Deghli$Madh | u | 20 | 3 |
| 21 | ra$Deghli$Madhura#Maja | u | 5 | 1 |
| 22 | ura#Majaura$Deghli$Mad | h | 19 | 3 |
| 23 | ura$Deghli$Madhura#Maj | a | 4 | 1 |

# Pruning Techniques

- Length Filtering:

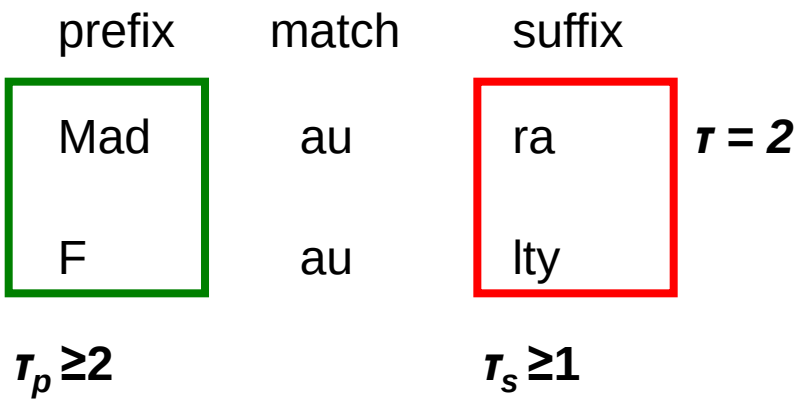
The possible length range of pattern *P* with is [|*P*|-*τ*,|*P*| + *τ* ].



# Pruning Techniques

- Position Filtering

Three parts: prefix, matched segment, and suffix.




$$\tau = \tau_p + \tau_s \geq 3$$

# Multiple Queries

| ID | Strings | $\tau$ |
|---|---|---|
| 1 | lodna | 1 |
| 2 | lodnna | 2 |
| 3 | Dehri | 1 |
| 4 | dehri | 1 |
| 5 | loddna | 2 |

lod na

lo dn na

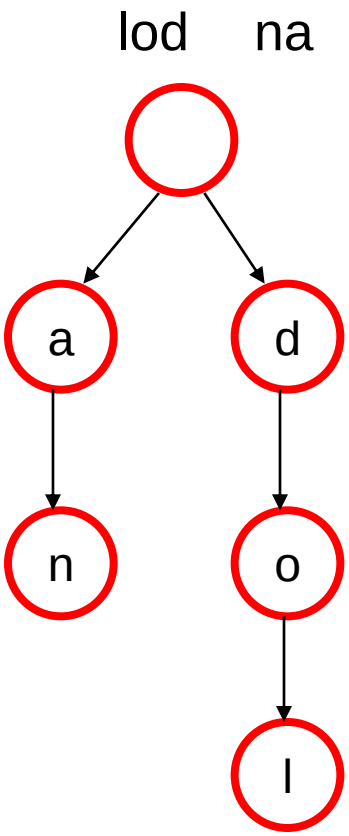
Deh ri

deh ri

lo dd na

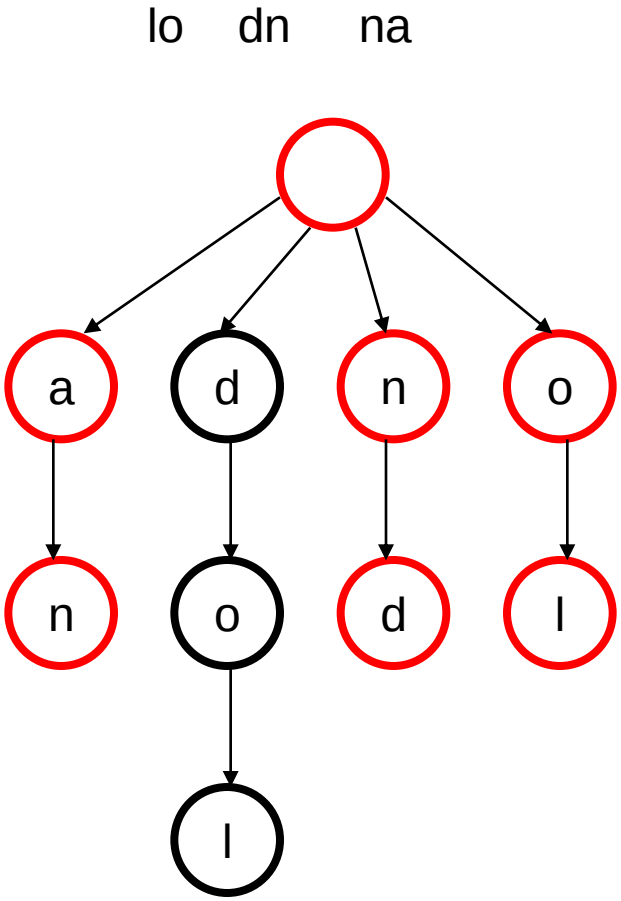
# Multiple Queries

| ID | Strings | *T* |
|---|---|---|
| 1 | lodna | 1 |
| 2 | lodnna | 2 |
| 3 | Dehri | 1 |
| 4 | dehri | 1 |
| 5 | loddna | 2 |



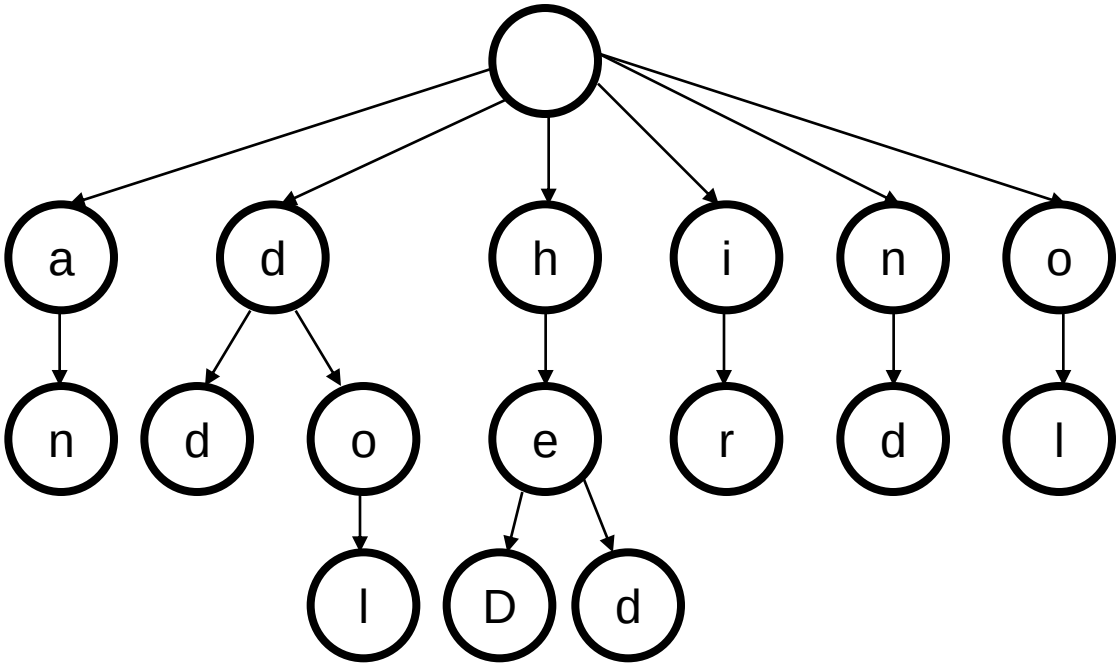

# Multiple Queries

| ID | Strings | *τ* |
|---|---|---|
| 1 | lodna | 1 |
| 2 | lodnna | 2 |
| 3 | Dehri | 1 |
| 4 | dehri | 1 |
| 5 | loddna | 2 |



# Multiple Queries

| ID | Strings | *T* |
|---|---|---|
| 1 | lodna | 1 |
| 2 | lodnna | 2 |
| 3 | Dehri | 1 |
| 4 | dehri | 1 |
| 5 | loddna | 2 |



# Look Ahead Verification

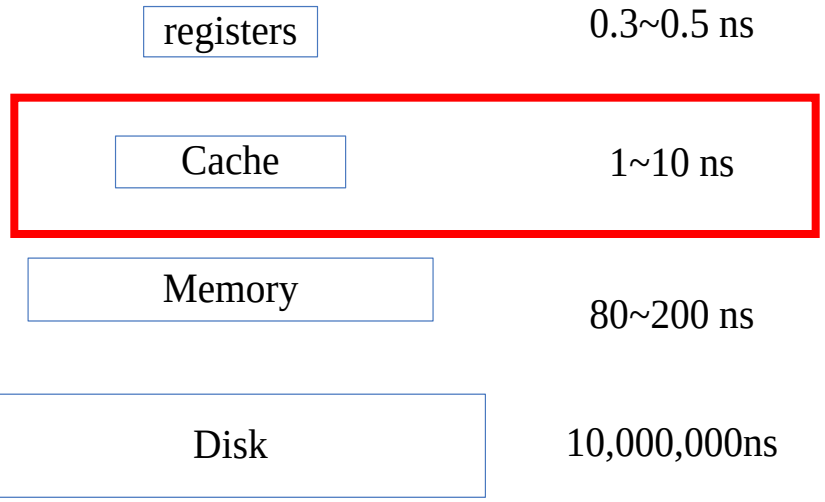
| | | s | i | m | i | l | a | r | l | y |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| s | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| i | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| m | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| i | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| l | 5 | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 |
| a | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 |
| l | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 1 | 1 | 2 |
| r | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 2 | 2 |
| y | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 3 | 2 | 2 |

case 1

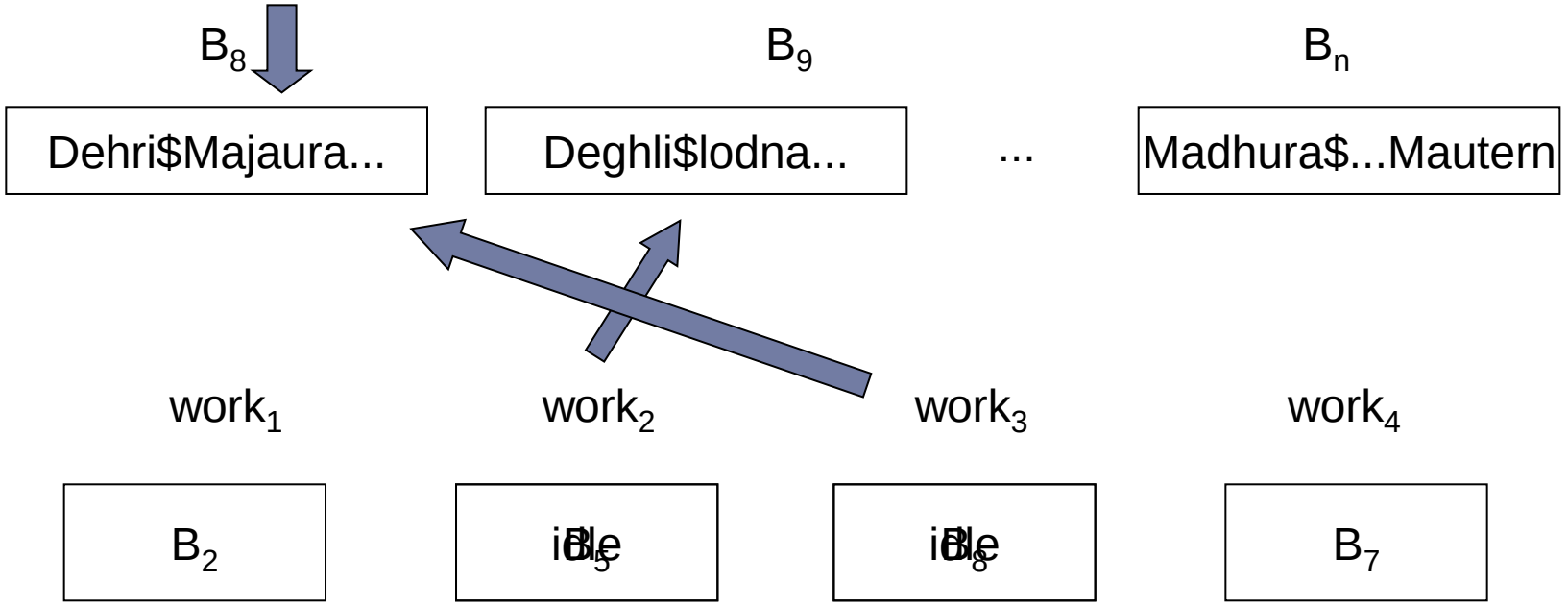
| | | r | e | f | e | r | e | n | c | e |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| d | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| i | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| f | 3 | 3 | 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| f | 4 | 4 | 4 | 3 | 3 | 4 | 5 | 6 | 7 | 8 |
| e | 5 | 5 | 4 | 4 | 3 | 4 | 4 | 5 | 6 | 7 |
| r | 6 | 5 | 5 | 5 | 4 | 3 | 4 | 5 | 6 | 7 |
| e | 7 | 6 | 5 | 6 | 5 | 4 | 3 | 4 | 5 | 6 |
| n | 8 | 7 | 6 | 6 | 6 | 5 | 4 | 3 | 4 | 5 |
| t | 9 | 8 | 7 | 7 | 7 | 6 | 5 | 4 | 4 | 5 |

case 2

# Cache-aware Parallel Optimization




# Cache-aware Parallel Optimization



# Approximate String Join

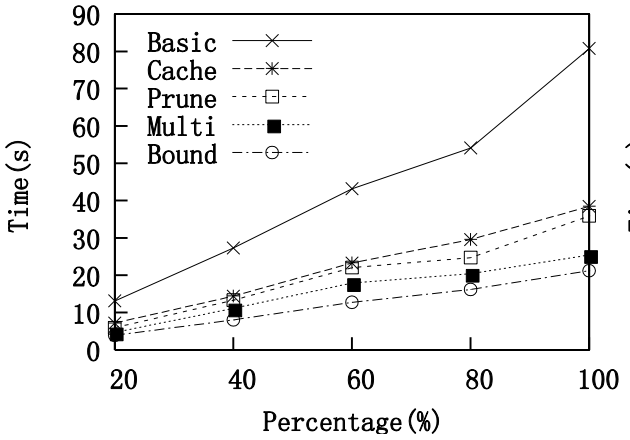
- Incremental Approximate String Join
  - ed（$S_1$, $S_2$）≤$\tau$ *also means* ed（$S_2$, $S_1$）≤$\tau$
  - remove the symmetrical case
  - stop the search when reach a ID ≥ current ID

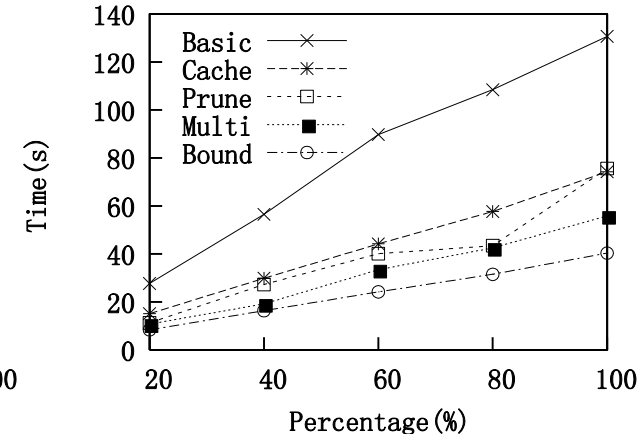
- Trie-based Approximate Join
  - build a reversed segment trie first
  - avoid the search processing for the duplicated segments
- Pruning techniques
  - count filter
  - stop the search for current segment when there is only one candidate, which will be itself.

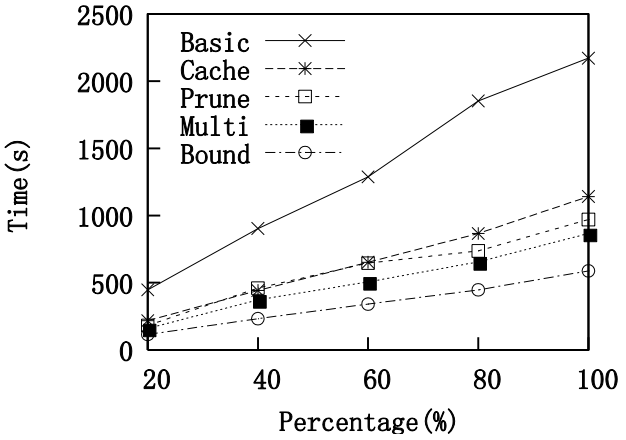
# Experiment

- Environment
  - C++ language
  - PC with 2.93 GHz Intel Core CPU
  - 4 GB main memory
  - Ubuntu operating system (Linux distribution).
- data sets
  - Geographical name
  - DBLP author
  - Human genome read

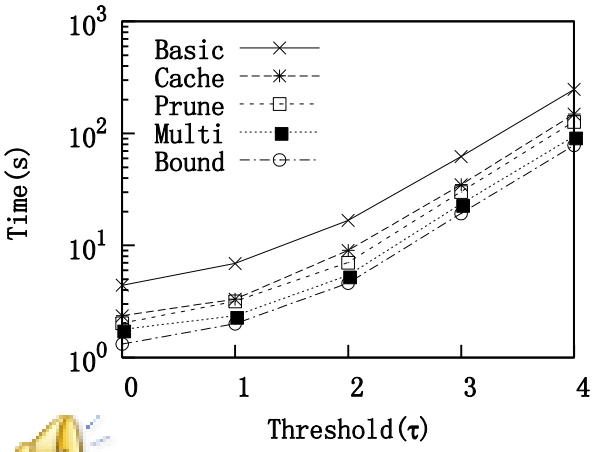
# Search Performance



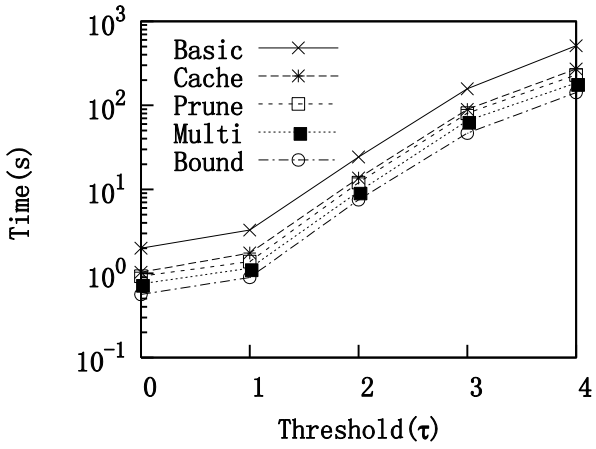



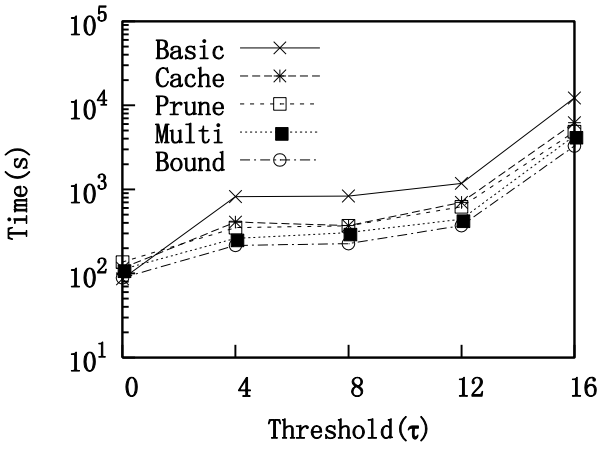

Geo



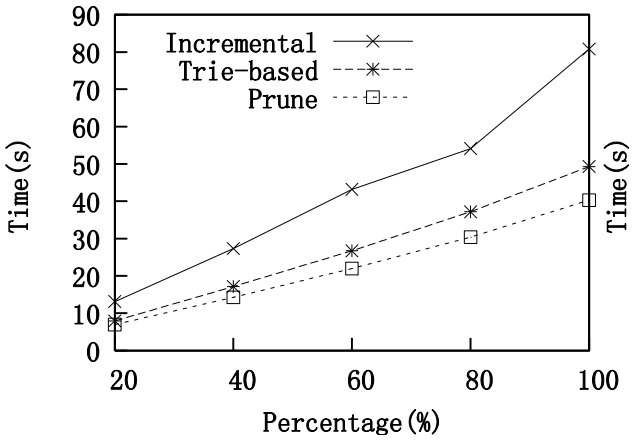

DBLP

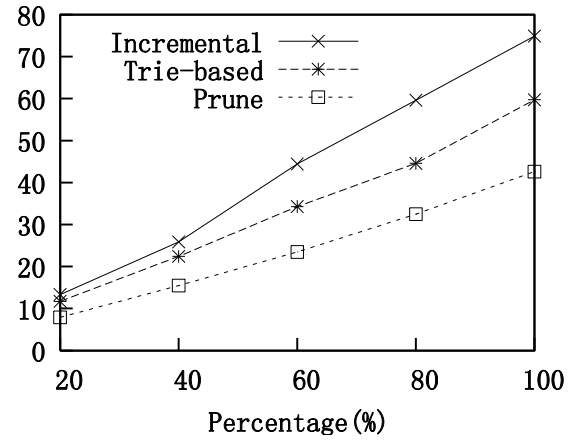


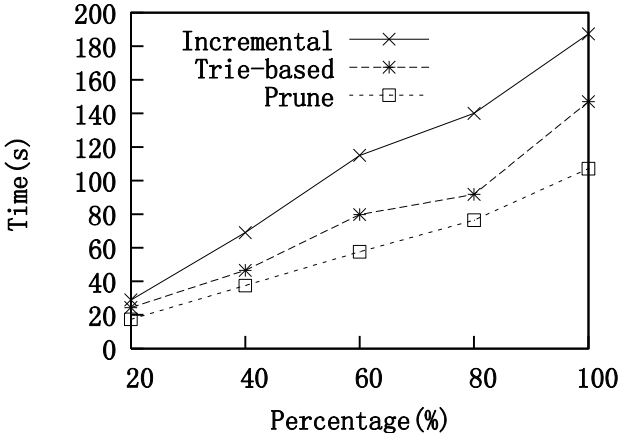

reads



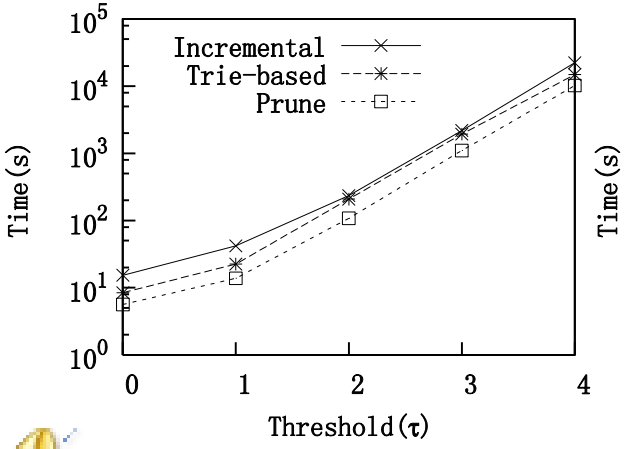

# Join Performance on DBLP



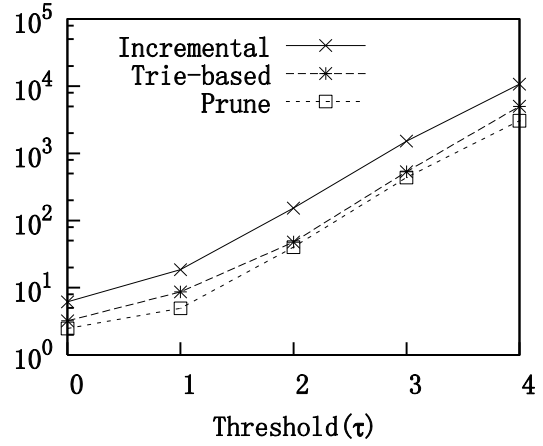



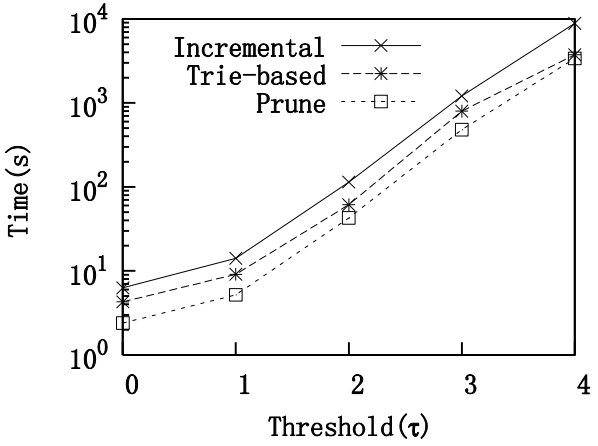

Geo




DBLP




reads




# Conclusions

- A new index BWTPA
- Cache-aware multi core framework
- Efficient pruning techniques
  - Length filter
  - Position filter
- Look ahead algorithm to improve edit distance
- Approximate string join
  - Incremental
  - Trie-based

# Thank you!