Mimir: Term-Distributed Indexing and Search for Secret Documents

Guoqiang Gao, Ruixuan Li, Xiwu Gu,
Kunmei Wen, Zhengding Lu, Kun Yan

Huazhong University of Science and Technology
Wuhan, China

Background

- Secret documents shared over government, army and enterprise intranets grow rapidly
- Most of the data are not well organized, but just being securely stored
  - It is inefficient that authorized users search secret documents through intranet

Background

- How to organize document data?
  - Inverted index is the most successful approach for management of web data
    - Widely used in many commercial search engines, such as Google
  - Building an inverted index over secret documents may be a good choice
    - High efficiency of search

Challenges and Solutions

- Scalability challenges
  - Document volume tends to grow huge
    - Using distributed and parallel techniques
    - Adding more nodes
- Security challenges
  - Security requirements of sensitive and secret documents grow rapidly
    - Access control
    - Encryption

Problems of Scalability Solutions

- Document-based distribution
  - Each node contains partial index with all terms
    - Documents and indexes can be distributed uniformly, and load is well balanced
  - High communication cost and low response
    - Each query should be processed on all nodes

Problems of Scalability Solutions

- Term-based distribution
  - Each node contains partial index with partial terms
  - Load Unbalance
    - Different size of term posting list
    - Different popularity of terms

Security Problems

- Security problems of public search engines
  - Unauthorized access
    - Everyone can use the search engines
  - Information leakage
    - Plain index can be used to obtain source documents once the index is illegally obtained

Contributions

- Mimir: term-distributed retrieval system
  - load balanced term-based distribution
- Using access control and encryption to protect sensitive information
  - random key encryption
  - partial key update technique
- Dynamically pipelined search
  - improve search efficiency for large document collection

Related Work

- Information security
  - Discretionary access control (DAC) and role-based access control (RBAC) are usually used for documents
  - Zerber proposes a user-group based access control for index
  - Encryption is a standard technique for storing data confidentially
    - Seitz proposed an architecture that allows users to store and share encrypted data

Related Work (contd.)

- Information retrieval
  - Gurajada proposes a new merge-based index maintenance strategy
  - Li et al. distribute index contains fuzzy keywords and encrypted files into the cloud servers
  - Google uses the cluster of servers to maintain a document-based distributed index
  - An approach to eliminate the bottleneck of the receptionist is to use pipelining

Outline

- Motivation
- Related Work
- Mimir
- Evaluation
- Conclusion

Mimir

- Architecture
- Strategies used in Mimir
  - Load balanced term distribution
  - Random key
  - Partial key update
  - Access control based on role and user
  - Dynamically pipelined search
- Utilizing Mimir

Architecture

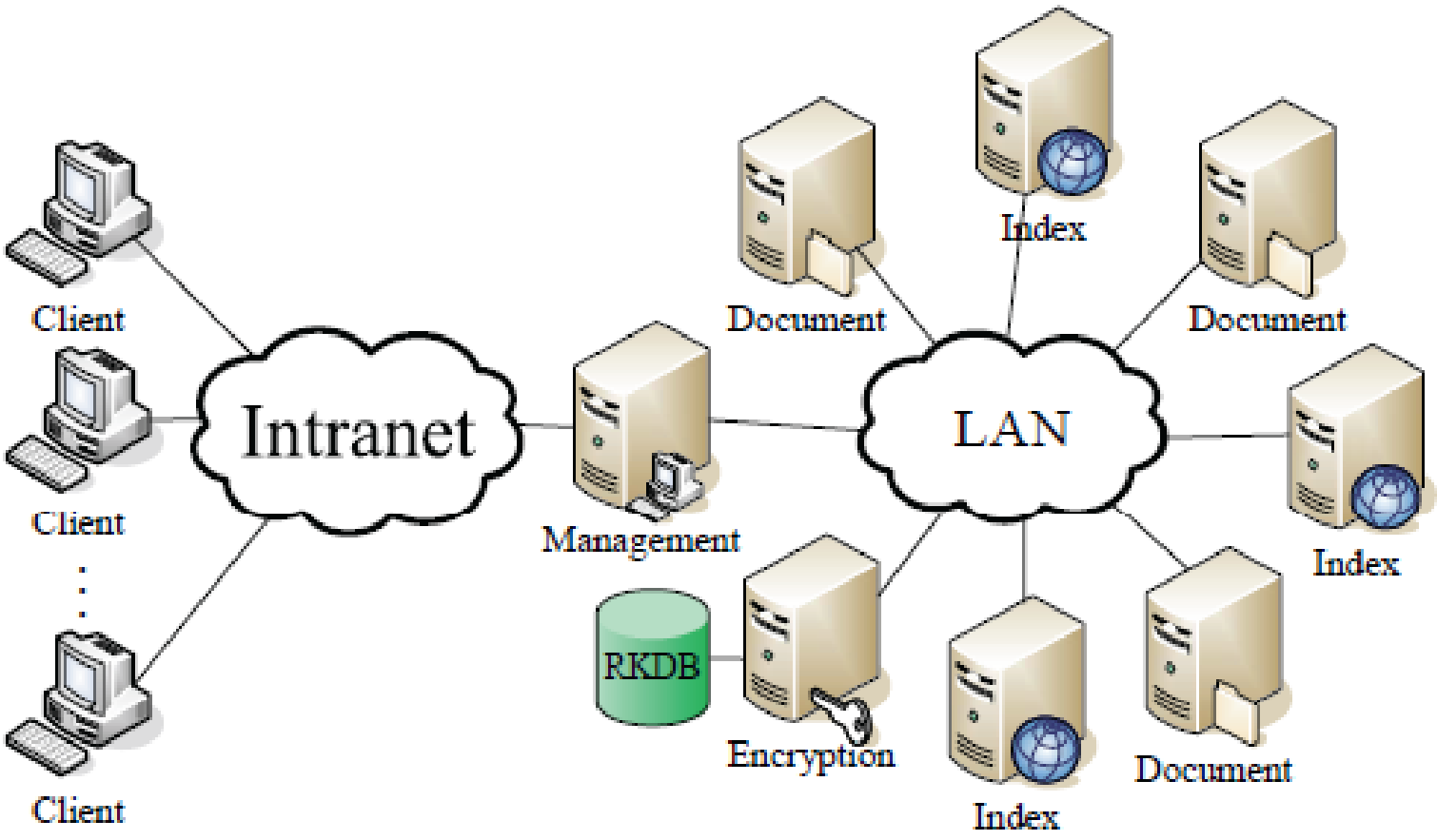


Fig. 1. The architecture of Mimir.

Term-based Distribution

- Example of distributed cipher index

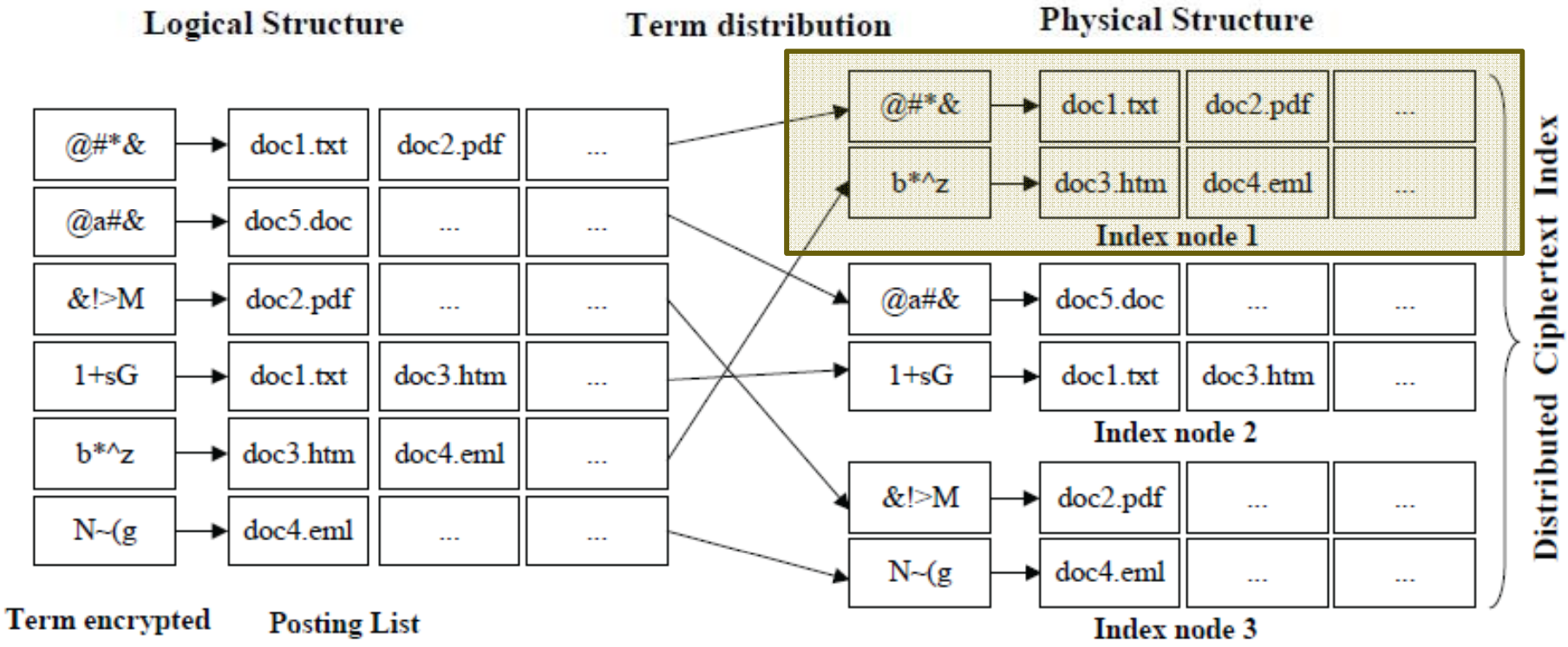


Fig. 2. The structure of the distributed cipher index with six terms.

Term-based Distribution

- Frequency of index server
  - $f(I_A) = \sum_{t_i \in T_A} f(t_i)$: I_A is index server A
  - f(t_i): frequency of term t_i
- Problem of term-based distribution
  - lack of load balance
    - Index servers with the same number of terms have different f(I)
    - Larger f(I_A), larger index and more load

Load Balanced Term Distribution

- Load Balanced Term Distribution
  - If term f(t) > $f_{threshold}$, choose server I which f(I) is minimum to allocate and record it; else randomly distribute to servers
  - Achieve both efficiency and load balance

Random Key Encryption

- Statistical attack
  - The attacker can obtain term frequency
    - The posting list is not encrypted in Mimir
    - The longer posting list, the higher frequency
  - Estimate the term plan text to obtain key
    - For example, “people” is the highest frequent term, so we can obtain it’s cipher text from index; use plan text and cipher text to crack key
  - Use the cracked key to decrypt the other terms

Random Key Encryption (contd.)

- Encrypt terms with random key, not using the same key
  - Use random key database (RKDB) to store keys
- Increase the difficulties of attack
  - If one term is cracked, the key belongs to it can not be used to decrypt other terms

Partial Key Update

- Updating key will result in index update for term encryption
  - Mimir doesn't update main index directly
- Randomly update 10% keys in RKDB each time
  - Confuse the correspondence between terms and keys
  - Destroy the keys obtained by attacker
- Build secondary index for updated terms
  - Indexes merge at idle time after secondary index reaches a threshold

Access Control Index

- Access control index (ACI) is based on roles and users
  - ACI is loaded into memory when system initializes
  - Use ACI to filter the query results
  - Flexible authorization management

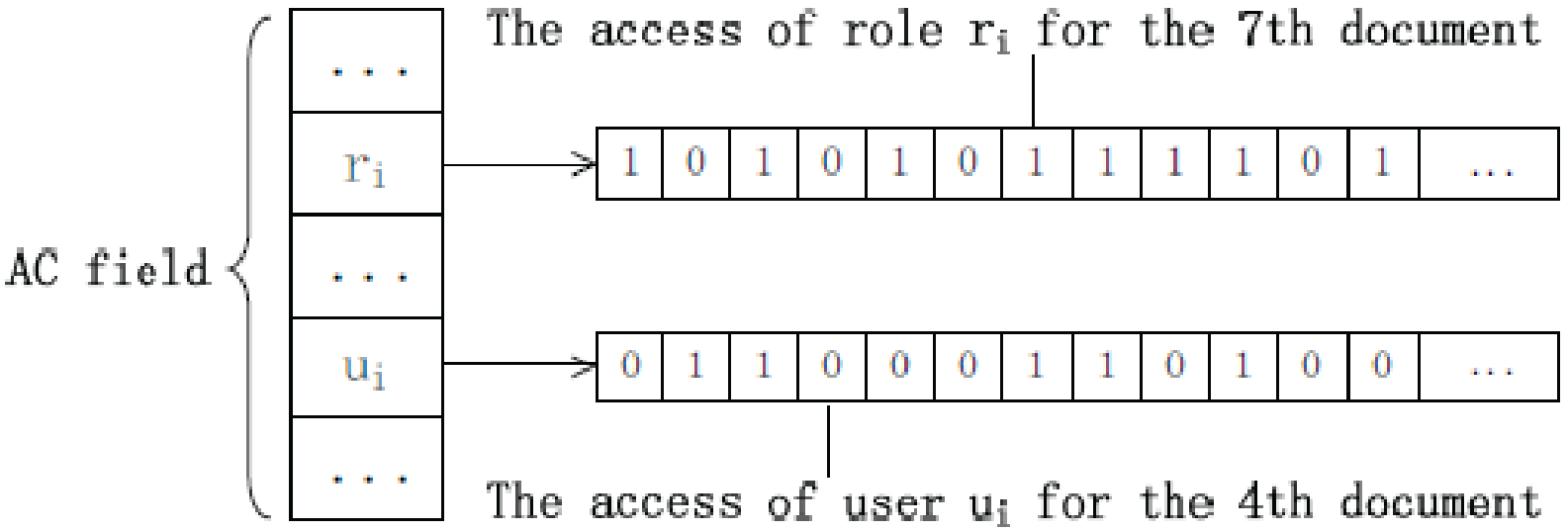


Fig. 3. Bitmap storage structure of ACI.

Dynamically Pipelined Search

- Why?
- Pipelined search
  - Search is done at index server one by one
  - Final result is returned to management server
  - Reduce the load of management server
- Dynamically pipelined search
  - If management server is low load, take full distribution search, else pipeline search
  - Trade-off between load and delay

Utilizing Mimir

- Distributed index
  - 1. clients upload documents to management server
  - 2. management server extracts the document's terms
  - 3. obtain terms' key from RKDB
  - 4. obtain the index server of terms according to load balanced term distribution
  - 5. encrypt terms and send them to index servers to build index
  - 7. encrypt documents and allocate them to document servers to store
    - Documents are distributed according to their names

Utilizing Mimir (contd.)

- Distributed search
  - 1. client submit query keywords to management server
  - 2. management server extracts terms from query keywords
  - 3. obtain terms' key from RKDB
  - 4. obtain the index server of terms according to load balanced term distribution
  - 5. encrypt terms and send them to index servers to query
  - 6. collect results from index servers and merge them
  - 7. return query result to client

Outline

- Motivation
- Related Work
- Mimir
- Evaluation
- Conclusion

Evaluation Setup

- Document data set
  - Reuter Corpus: volume is 1.55GB
- Beowulf-style cluster of 7 computers
  - 2.5GHz Intel Xeon with 2GB RAM and 1TB local SATA disk in a RAID-5 configuration
  - One acts as the management server in which encryption card and RKDB are installed
  - The other six computers are acted as both index and document servers

Security Analysis

- Security guarantees
  - Access control
    - Prevent unauthorized data access when using the system
  - Encryption
    - Protect data confidential even if the indexes and documents are stolen
  - Random key
    - Increase the difficulty of statistical attacks
  - Partial key update
    - Further increase system security

Load Balance

- General term-based distribution lacks load balance
- Both Mimir and document-based distribution have good performance on load balance, but Mimir has less communication cost

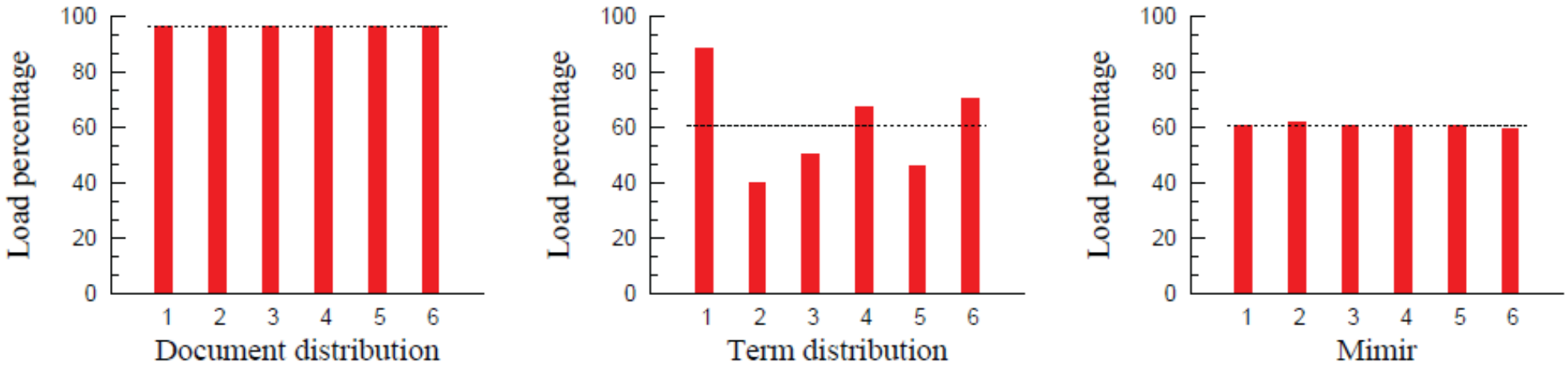
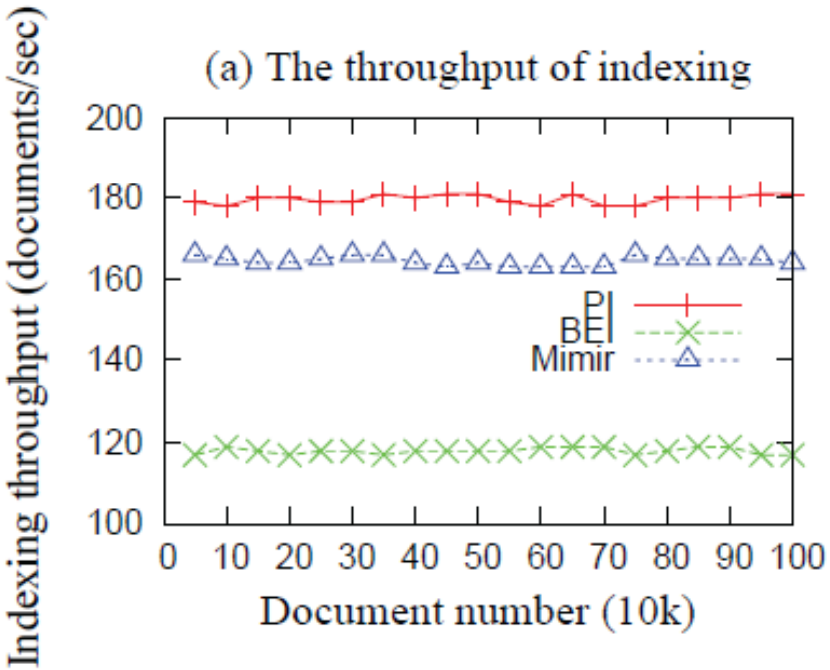
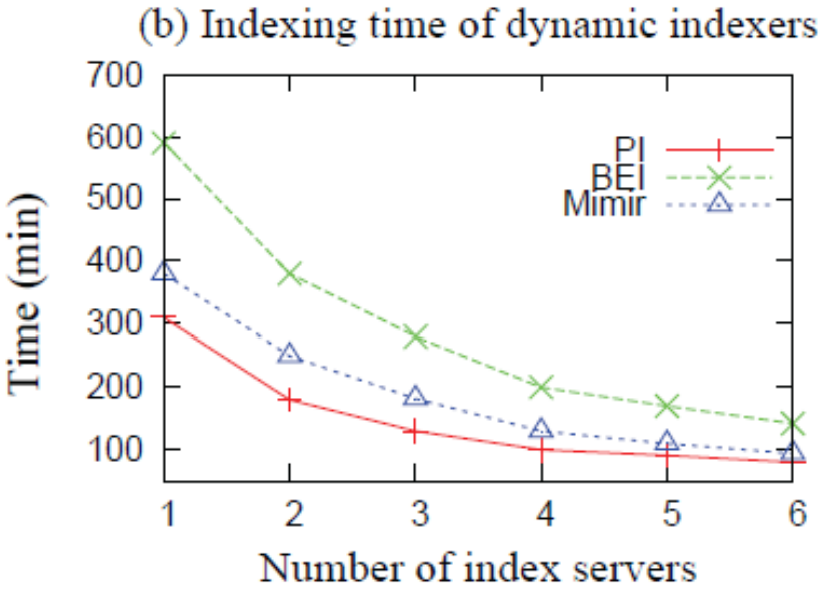


Fig. 4. Distribution of the average load per node in document distribution, term distribution and Mimir.

Indexing Performance

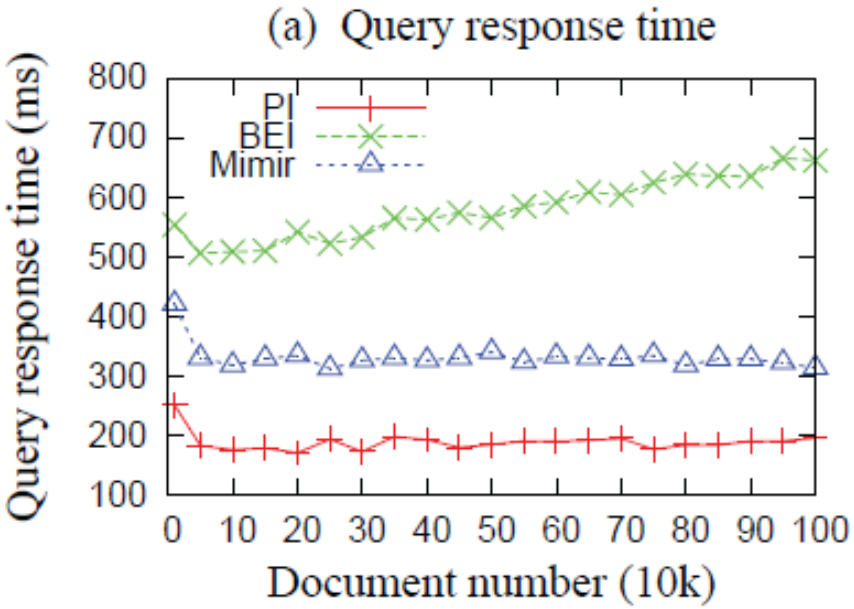
- PI is plain text index
- BEI is block encryption indexing
  - Encrypt index as a whole
- Mimir has better performance, close to PI





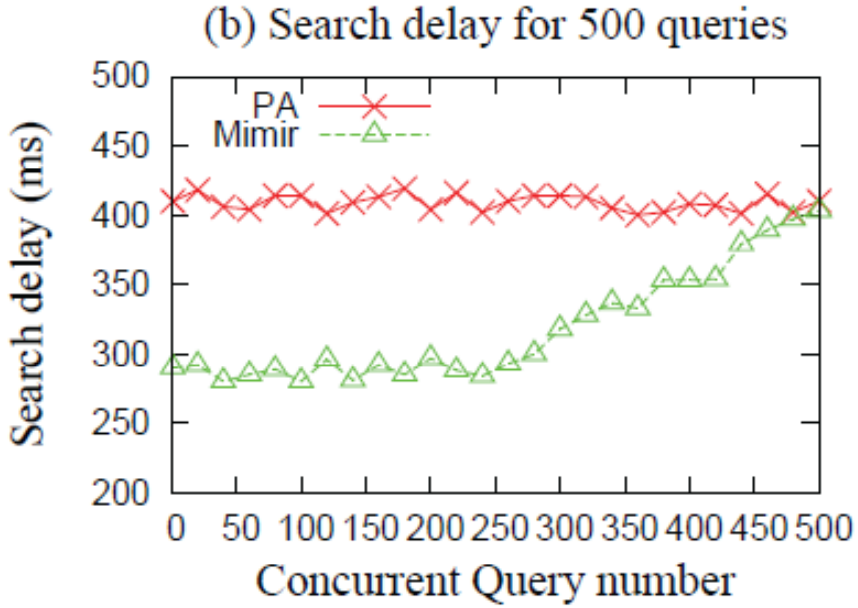
Query Performance

- Query response time
  - Mimir has the average response time 0.3s, close to PI



Query Performance (contd.)

- Efficiency of dynamically pipelined search
  - PA is general pipelined approach
  - When load is low, Mimir has better performance



Outline

- Motivation
- Related Work
- Mimir
- Evaluation
- Conclusion

Conclusion

- Summary of contributions
  - Load balanced term distribution
  - Random key encryption
  - Partial key update
  - Access control base on role and user
  - Dynamically pipelined search
  - Implement Mimir and test the performance

Questions?

- Ruixuan Li
- Huazhong University of Science and Technology
- rxli@hust.edu.cn
- http://idc.hust.edu.cn