ICDE 2020 International Conference on Data Engineering

Enabling Efficient Random Access to Hierarchically-Compressed Data

Feng Zhang †, Jidong Zhai ◇, Xipeng Shen #, Onur Mutlu ⋆, Xiaoyong Du †

†Renmin University of China
◇Tsinghua University
#North Carolina State University
⋆ETH Zürich

Outline

1. Background
2. Motivation
3. Operations to Support
4. Challenges
5. Our Solution
6. Evaluation
7. Conclusion

1. Background

- TADOC: Text Analytics Directly on Compression

Input:

file0: w1 w2 w3 w1 w2 w4
w1 w2 w3 w1 w2 w4

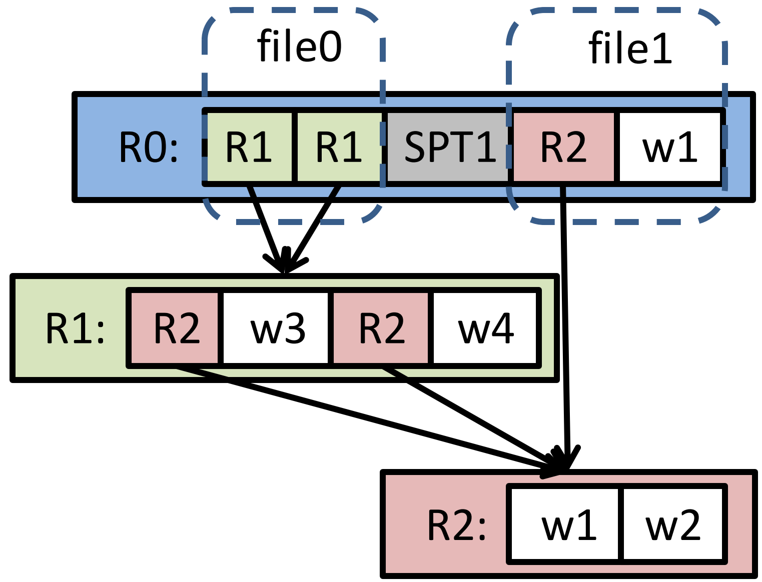
file1: w1 w2 w1

(a) Original data

Rules:

R0 → R1 R1 SPT1 R2 w1
R1 → R2 w3 R2 w4
R2 → w1 w2

(b) TADOC compressed data



(c) DAG Representation

w1: 0 w2: 1 w3: 2
w4: 3 **R0**: 4 **R1**: 5
R2: 6 SPT1: 7

(d) Numerical representation

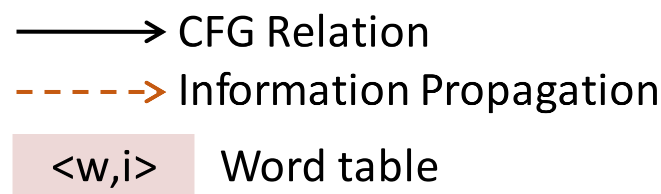
4 → **5** **5** **7** **6** 0
5 → **6** 2 **6** 3
6 → 0 1

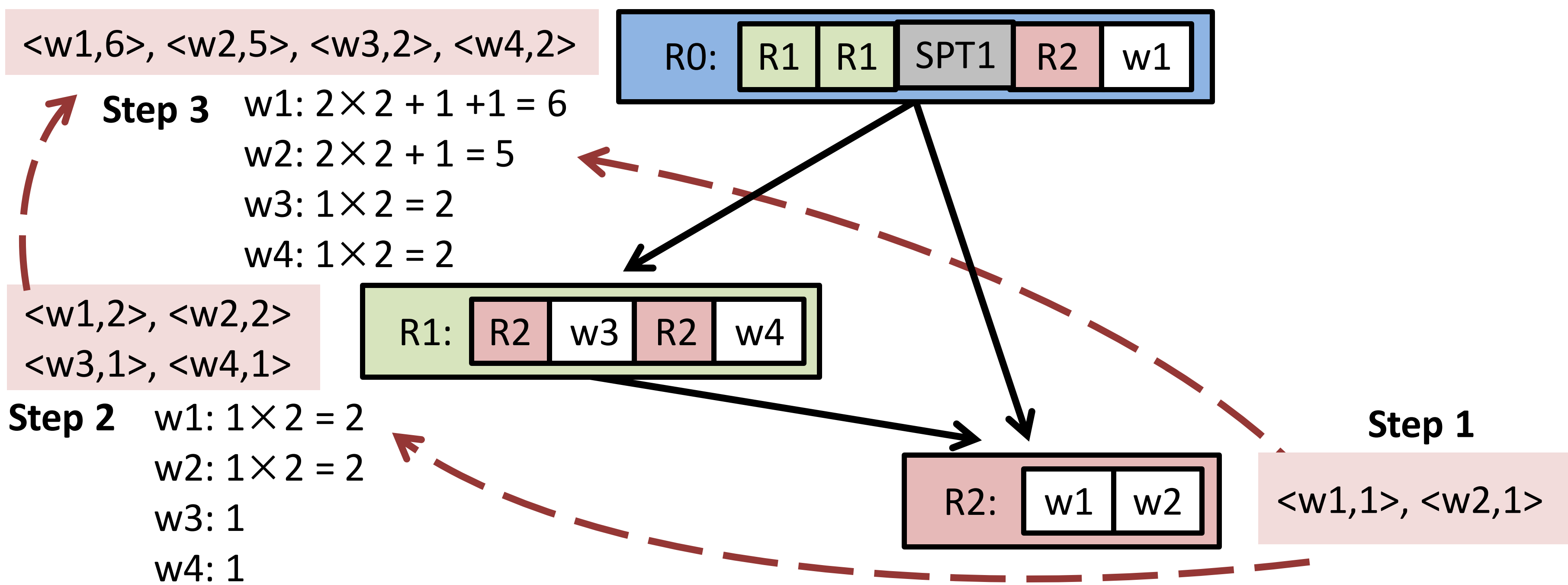
(e) Compressed data in numerical form

Zhang, F., Zhai, J., Shen, X., Mutlu, O., & Chen, W. (2018). Efficient document analytics on compressed data: Method, challenges, algorithms, insights. *Proceedings of the VLDB Endowment*, *11*(11), 1522-1535.

1. Background

- Example: word count

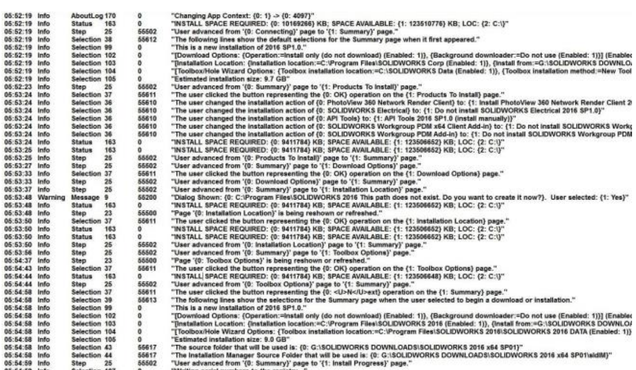




2. Motivation

News

Legal files



Logs

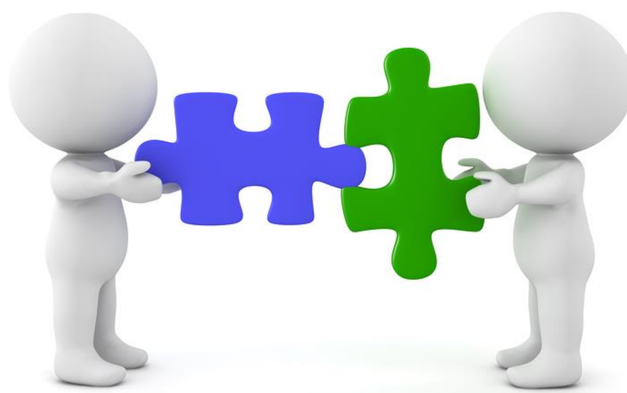
Random Access

- search
- extract
- count
- insert
- append

3. Operations to Support

- Random Access
  - **extract(file,offset,length)**
  - **search(file,word)**
  - **count(file,word)**
  - **insert(file,offset,string)**
  - **append(file,string)**

Locality

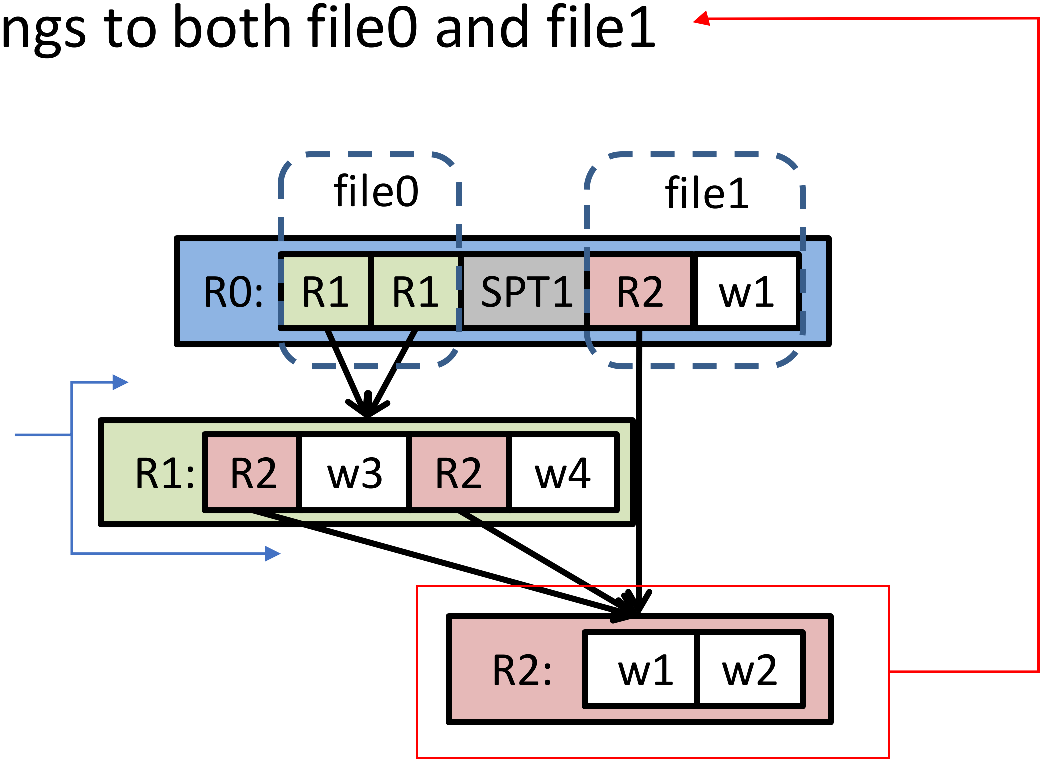


Compatibility

User Transparency

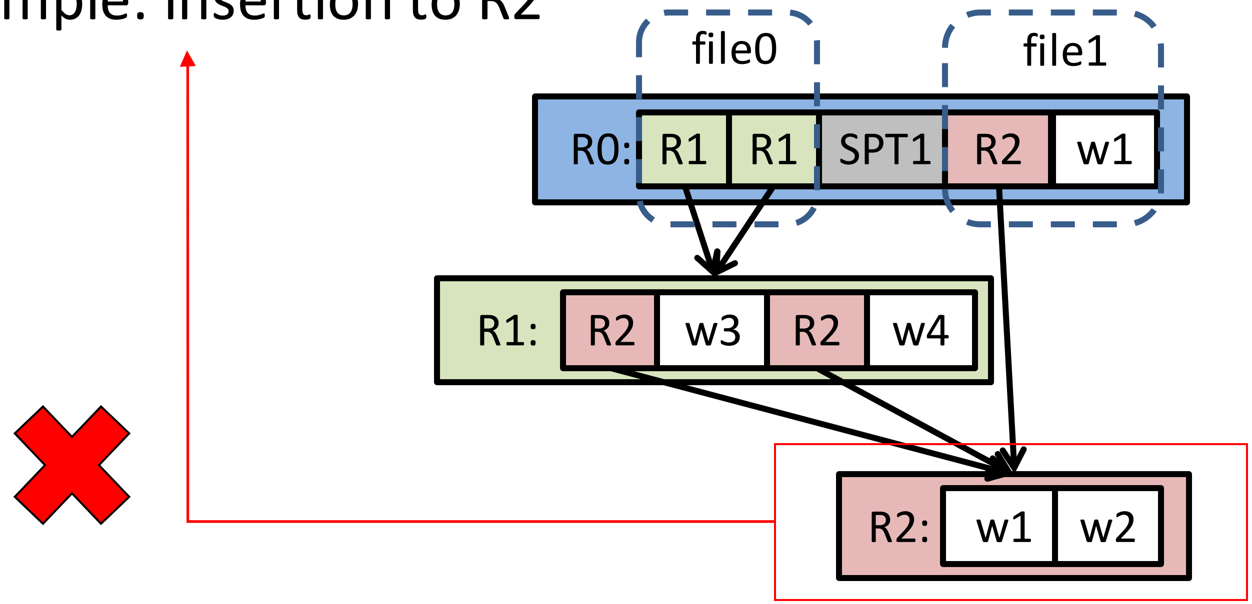
4. Challenges

- Hierarchical Structure of the DAG
  - Example: R2 belongs to both file0 and file1
- Uni-Directionality
  - Edges



4. Challenges

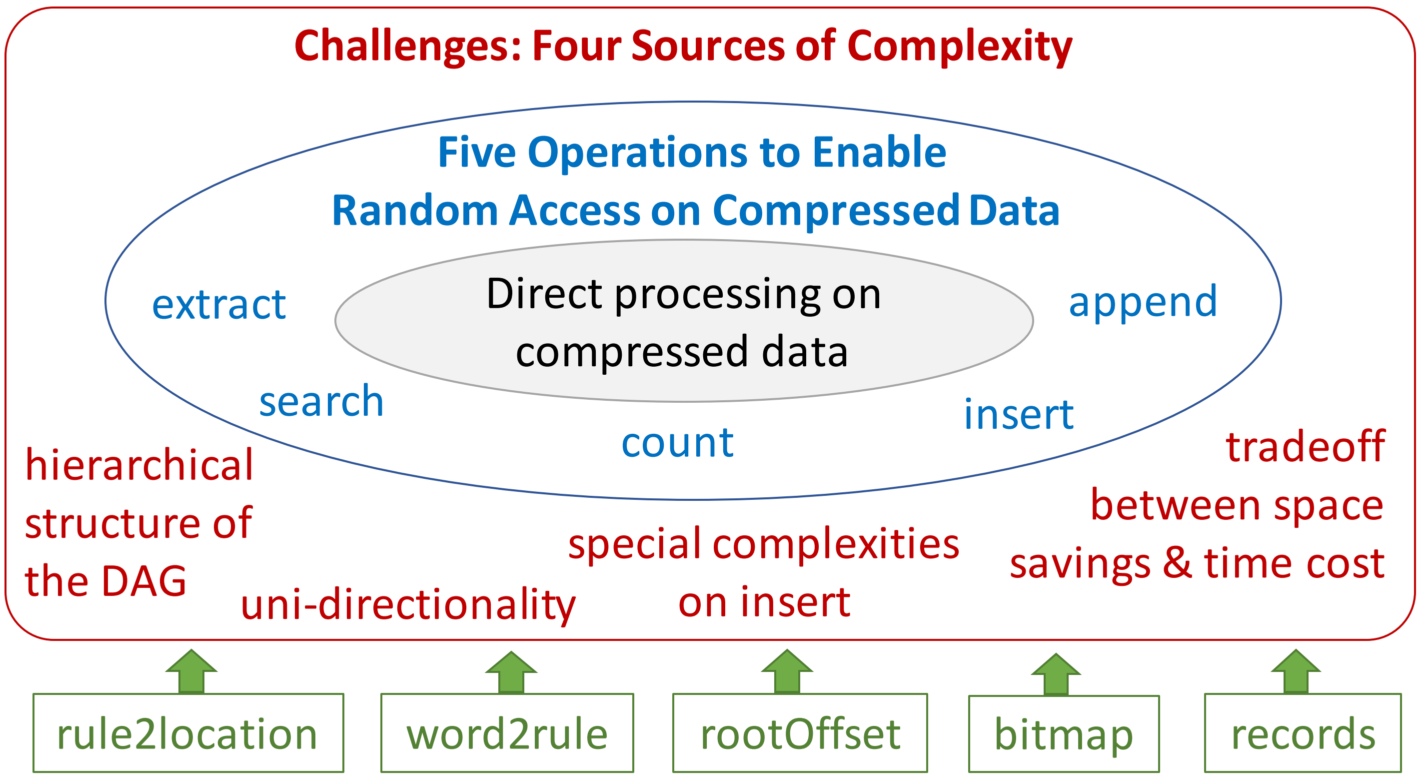
- Special Complexities on Insert
  - Example: insertion to R2



- Tradeoff between Space Savings and Time Cost
  - Index space cost

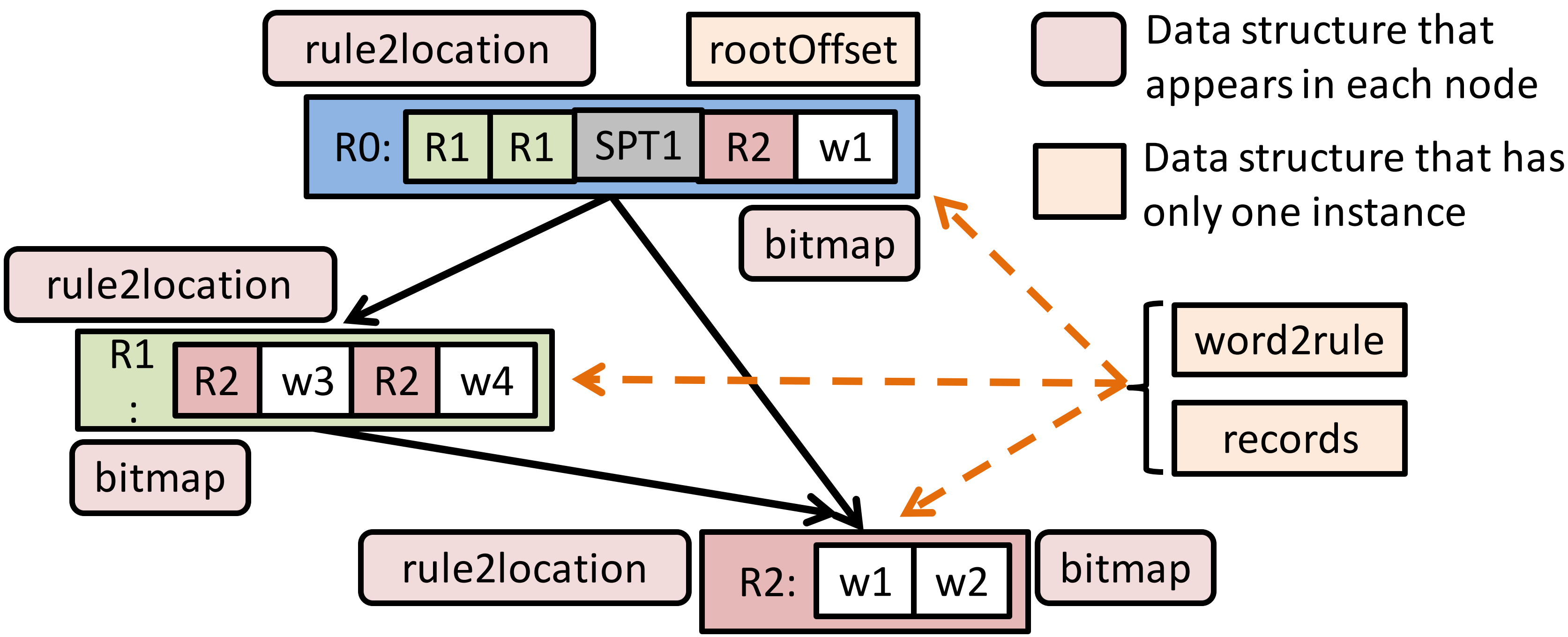
5. Our Solution

Solution Techniques: Five Data Structures



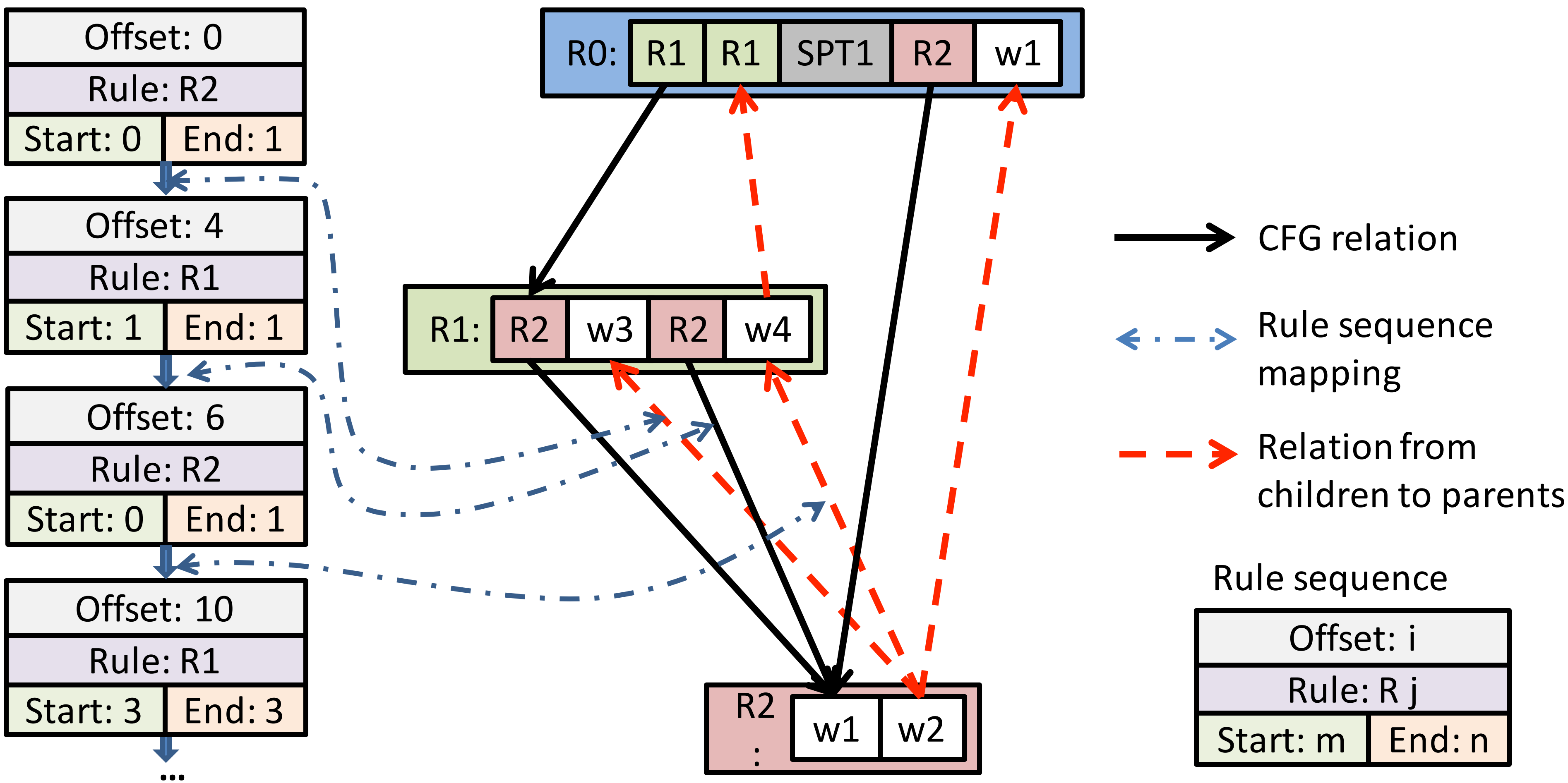
5. Our Solution

- Relations for data structures



5. Our Solution

- extract(file,offset,length)



5. Our Solution

- search(file,word)
  - word2rule
  - rule2location
- count(file,word)
  - word2rule
  - rule2location

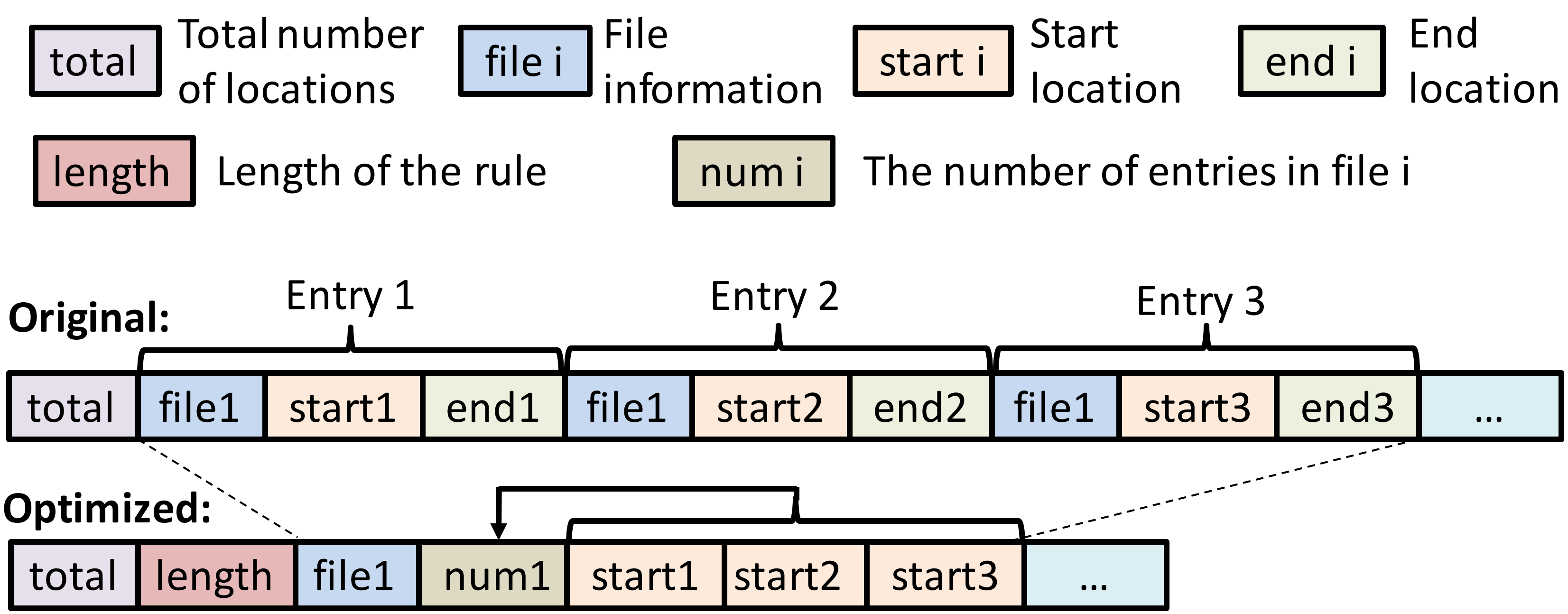


Illustration of *rule2location* optimization.

5. Our Solution

- insert(file,offset,string)
- append(file,string)

The Record Data Structure

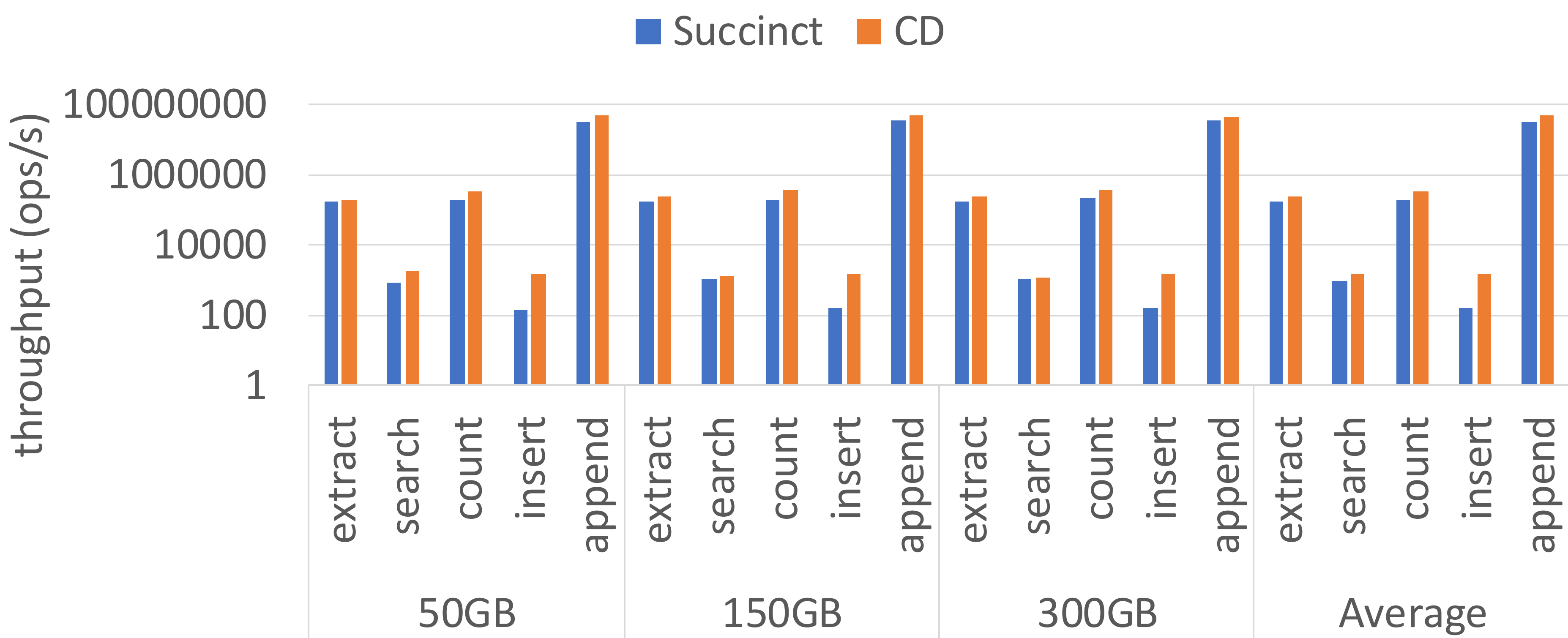
```
struct Record{
  int fileID; // file, such as file1
  int fileOffset; //file offset to insert, such as 100
  int ruleID; // the rule ID to insert, such as 0
  int ruleLocation; //the inserted location, such as 2
  int replaceWord; //the replaced word, such as w2
  string content; //content string
  int ptr; //the recordID inserted at the same place. Default is -1
  int ruleStartOffset; //the starting offset of the rule to insert, such as 0
};
```

6. Evaluation

- Five operations
  - search, extract, count, insert, append
- Five datasets
  - 580 MB ~ 300 GB
- Two platforms
  - Single node
  - Spark cluster (10 nodes on Amazon EC2)

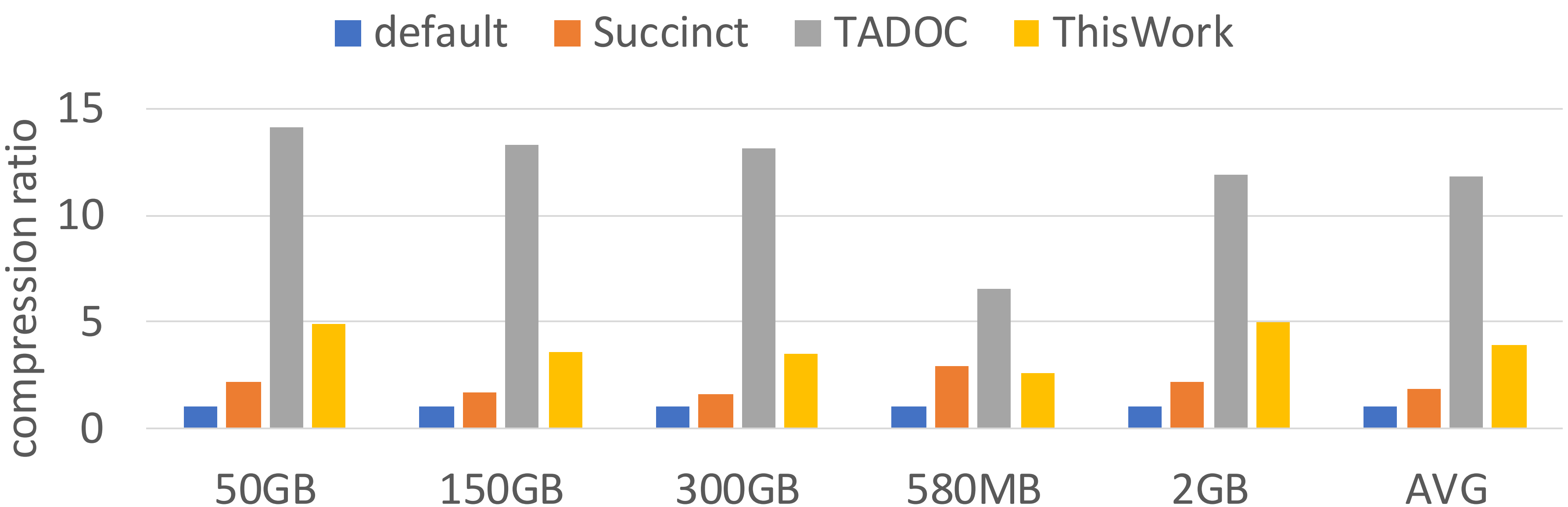
6. Evaluation

- On average, the overall throughput of our proposed techniques (CD) is **3.1×** of Succinct's throughput in a distributed environment.



6. Evaluation

- The average compression ratio we observe is 3.9, which is still much more compact than the 1.8 compression ratio of Succinct.
- compression ratio = original size / compressed data size



7. Conclusion

- We provide a set of new techniques that enable efficient random access operations on hierarchically-compressed data
- Compatible with TADOC: data traversal operations
- We remove a major barrier against practical adoption of direct text analytics on compressed data.

Thanks!

- Any questions?

Feng Zhang †, Jidong Zhai ◊, Xipeng Shen #, Onur Mutlu ⋆, Xiaoyong Du †

†Renmin University of China
◊Tsinghua University
#North Carolina State University
⋆ETH Zürich