VLDB2018

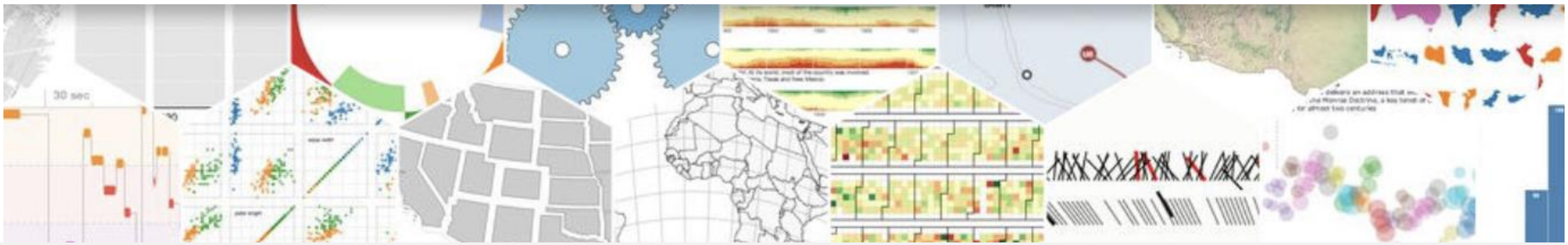
Efficient Document Analytics on Compressed Data: Method, Challenges, Algorithms, Insights

Feng Zhang [†◊], Jidong Zhai [◊], Xipeng Shen [#], Onur Mutlu [⋆], Wenguang Chen [◊]

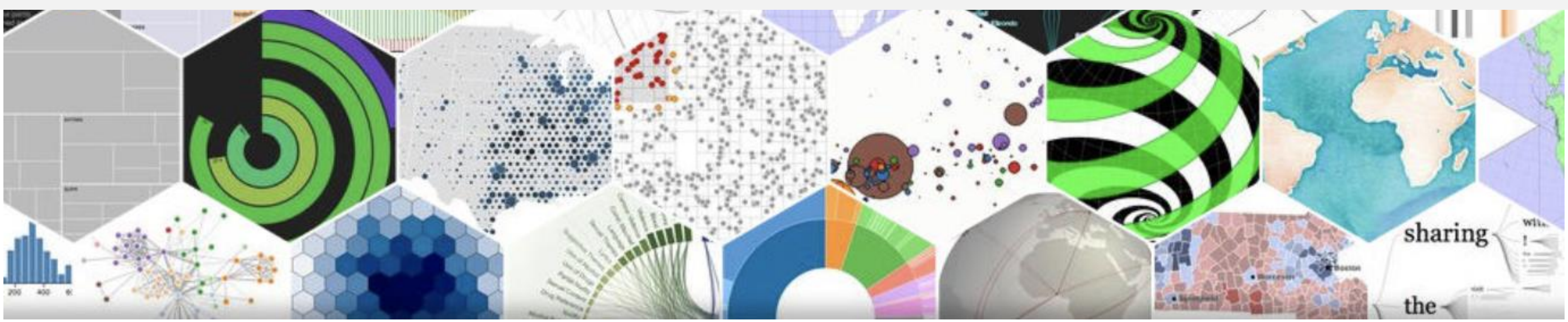
†Renmin University of China
◊Tsinghua University
#North Carolina State University
⋆ETH Zürich

Motivation

- Every day, 2.5 quintillion bytes of data created – 90% data in the world today has been created in the last two years alone[1].



How to perform efficient document analytics when data are extremely large?



[1] What is Big Data?
https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

Outline

- **Introduction**
  - **Motivation & Example**
  - **Compression-Based Direct Processing**
  - **Challenges**
- Guidelines and Techniques
  - Solution Overview
  - Guidelines
- Evaluation
  - Benchmarks
  - Results
- Conclusion

Motivation

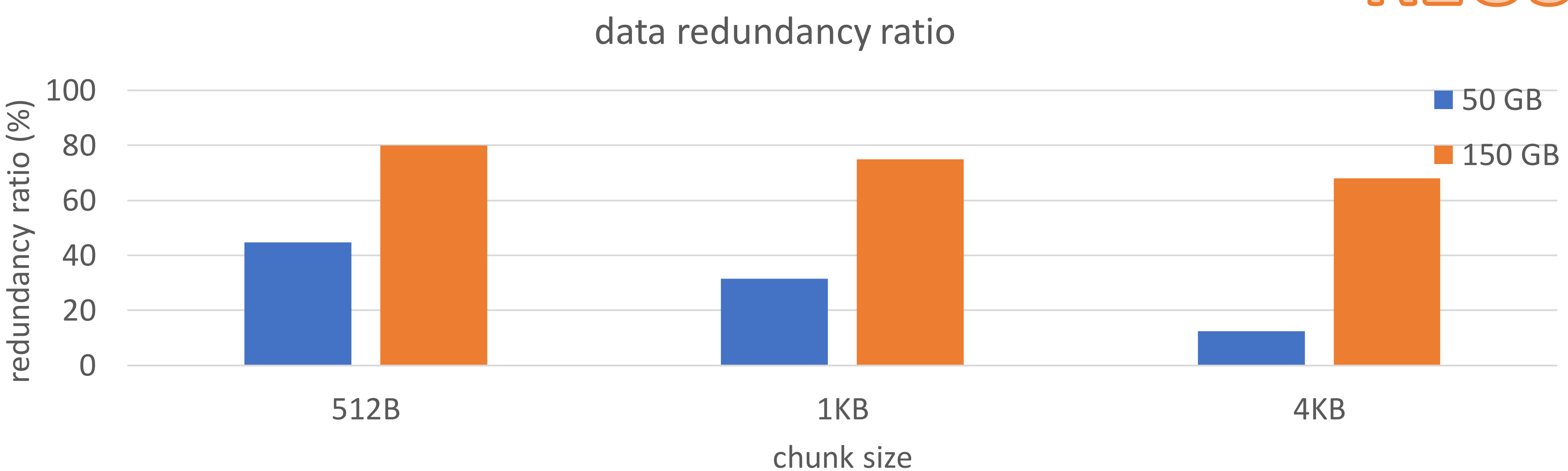
How to perform efficient document analytics when data are extremely large?

- Challenge 1:
  - SPACE: Large Space Requirement
- Challenge 2:
  - TIME: Long Processing Time

Motivation

- **Observation**
  - **Using Hash Table to check redundant content for Wikipedia dataset**

REUSE



Whether we can perform analytics directly on compressed data?

Our Idea

- Compression-based direct processing
- *Sequitur* algorithm meets our requirement

Input:

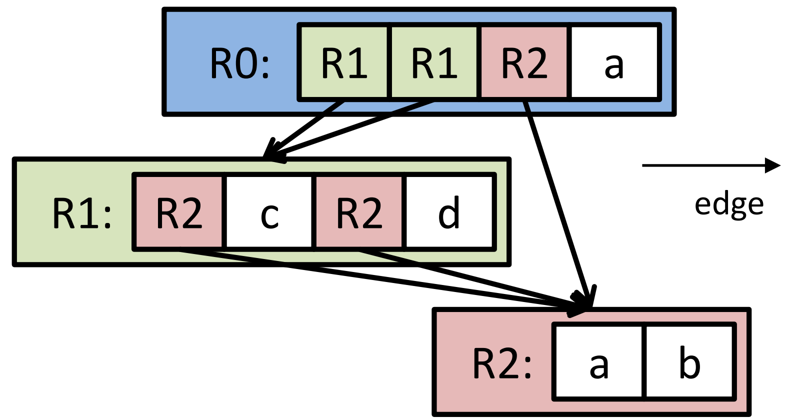
a b c a b d a b c a b d a b a

(a) Original data

Rules:

R0 → R1 R1 R2 a
R1 → R2 c R2 d
R2 → a b

(b) Sequitur compressed data



(c) DAG Representation

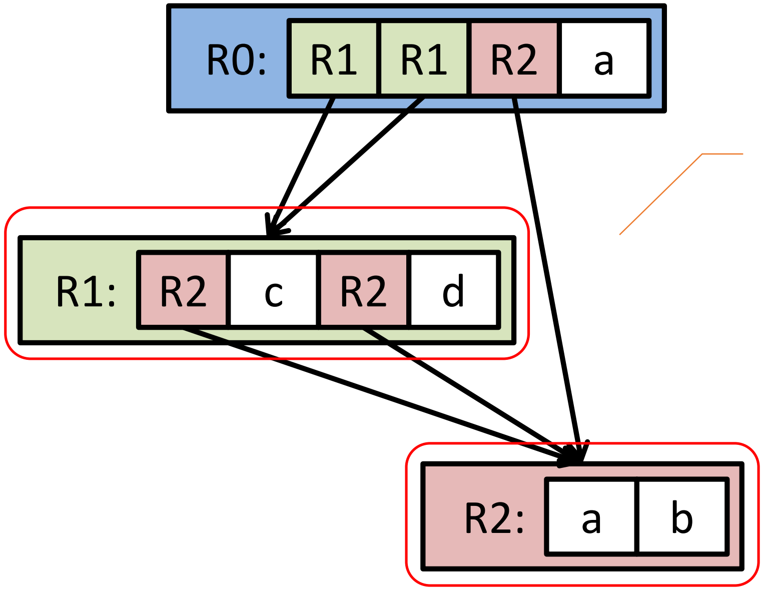
a: <u>0</u> b: <u>1</u> c: <u>2</u> d: <u>3</u>
R0: 4 R1: 5 R2: 6

(d) Numerical representation

4 → 5 5 6 <u>0</u>
5 → 6 <u>2</u> 6 <u>3</u>
6 → <u>0</u> <u>1</u>

(e) Compressed data in numerical ID

Double benefits



Appear more than once, but only **store** once!

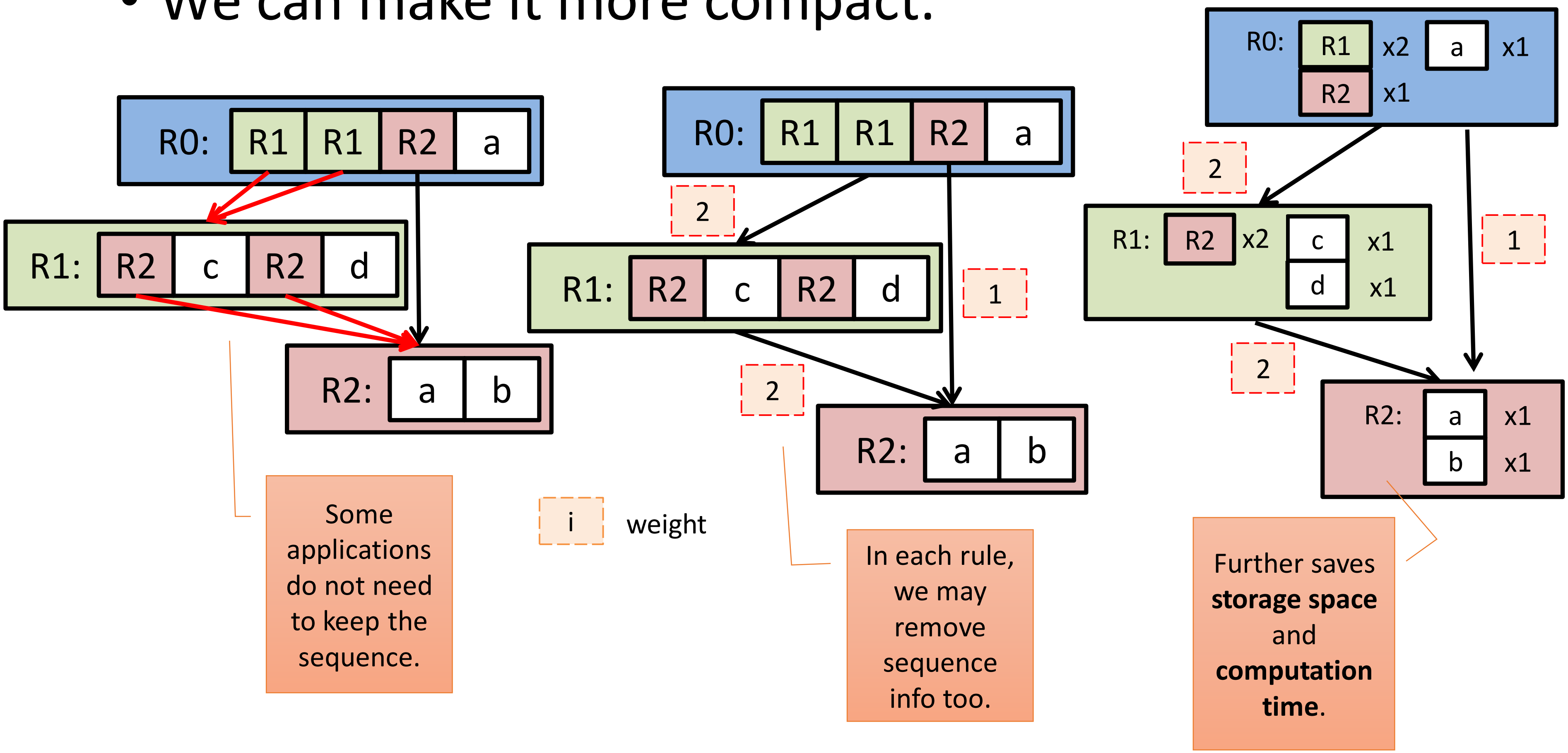
Challenge 1: Space

Appear more than once, but only **compute** once!

Challenge 2: Time

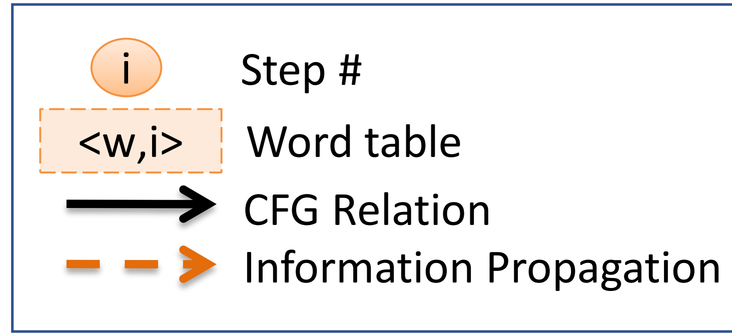
Optimization

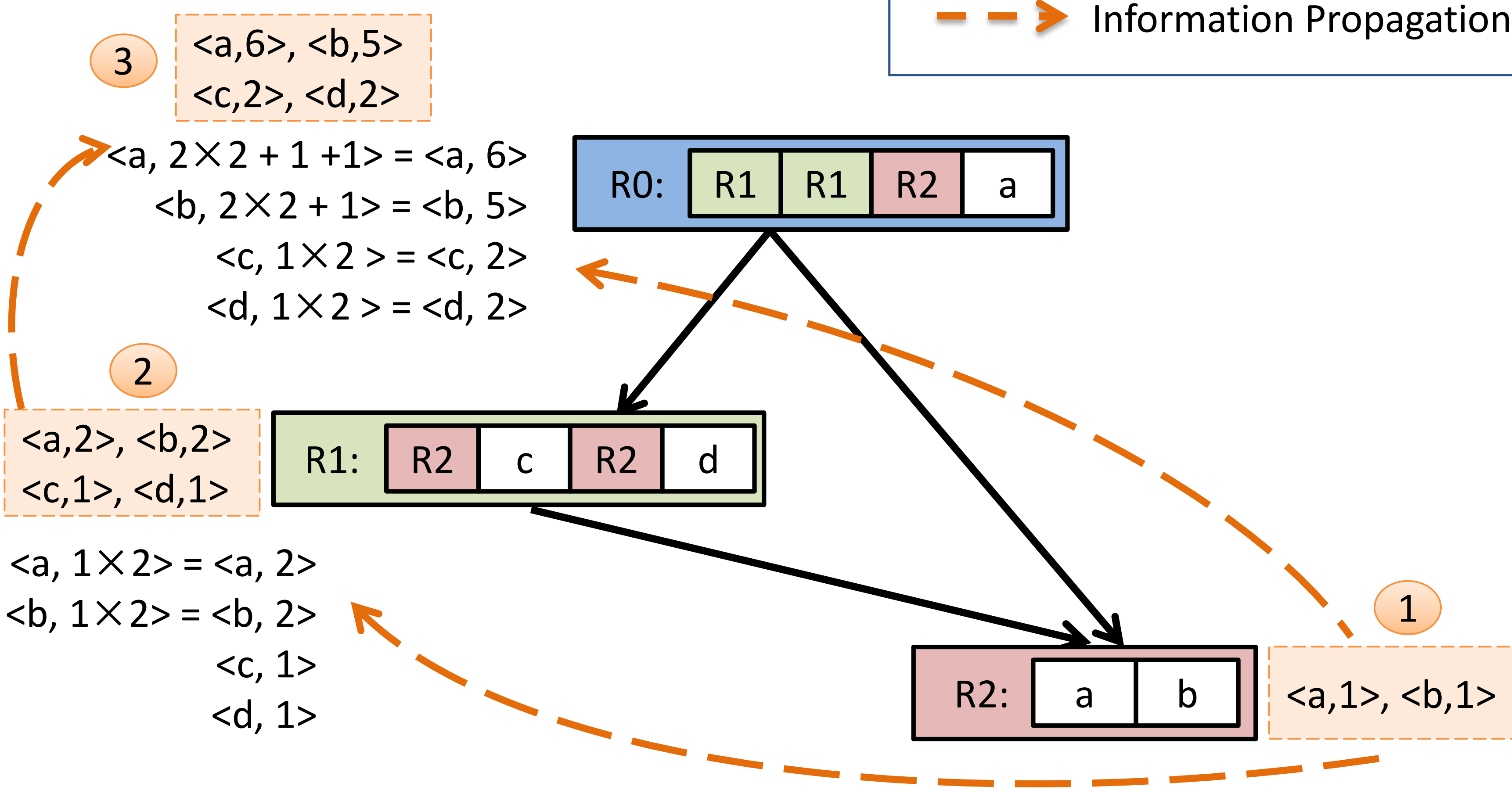
- We can make it more compact.



Example

- Word Count

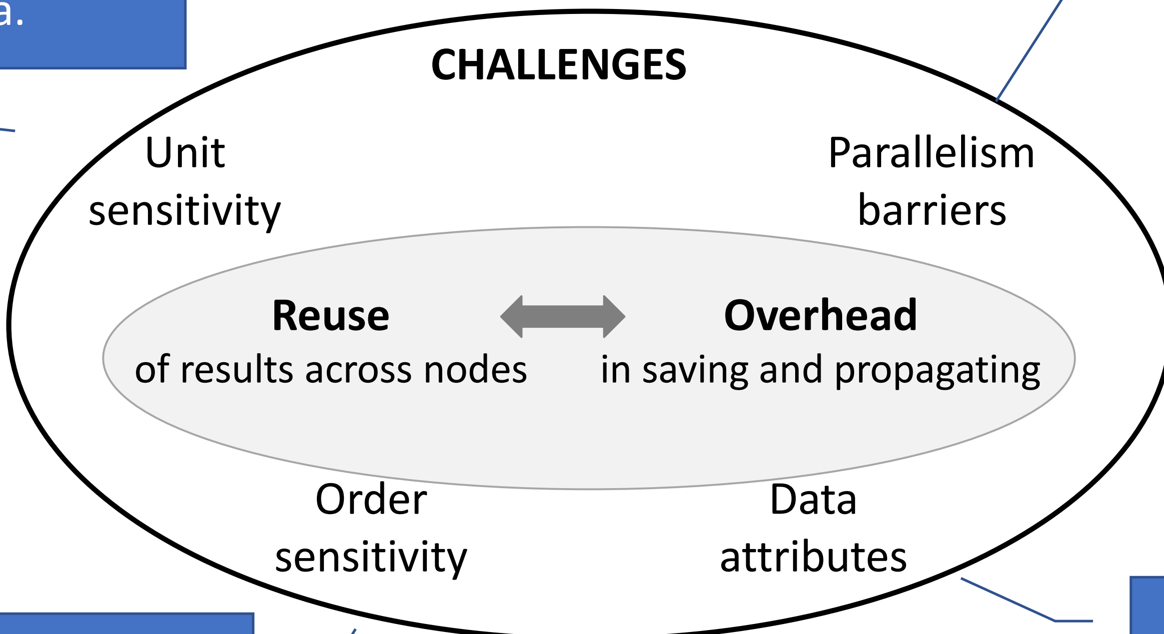




Challenges

How to organize data.

How to perform parallelism on large datasets.



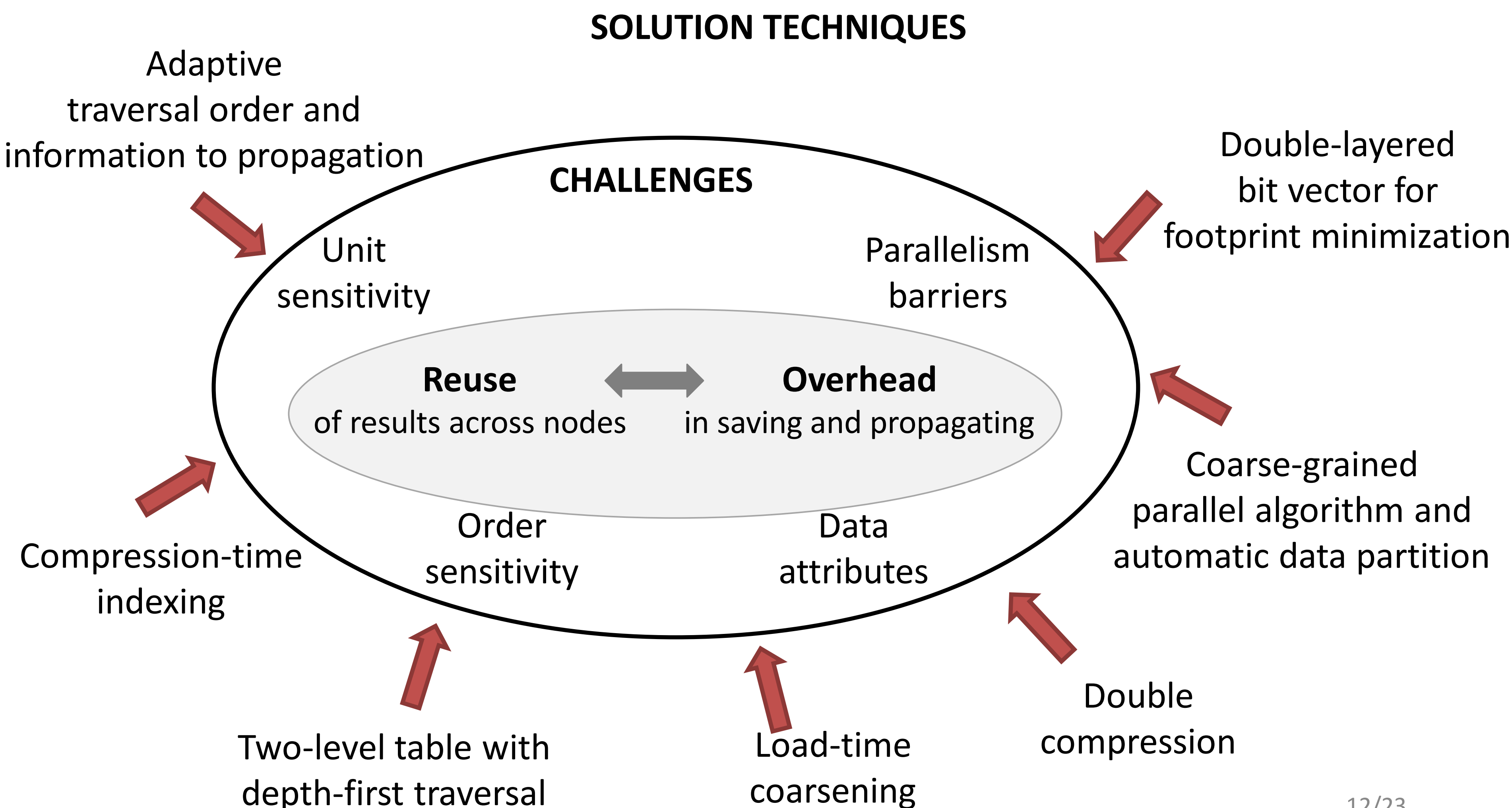
How to accommodate the order for applications that are sensitive to the order.

How to utilize the attributes of datasets.

Outline

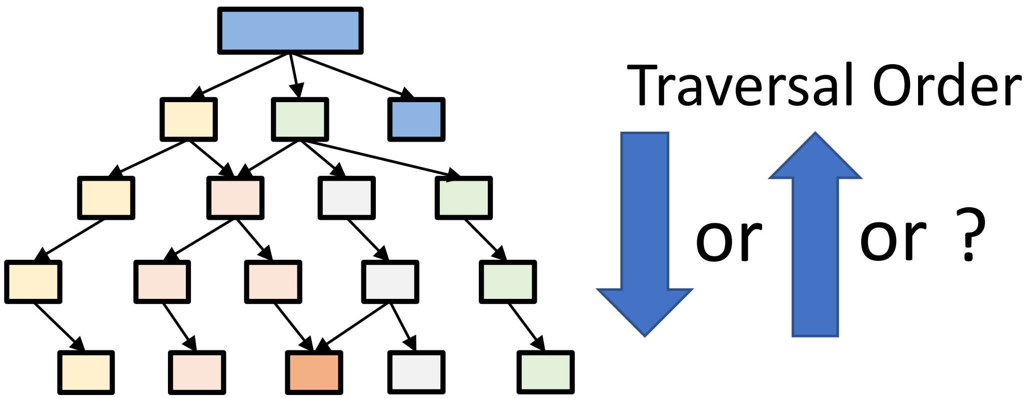
- Introduction
  - Motivation & Example
  - Compression-Based Direct Processing
  - Challenges
- **Guidelines and Techniques**
  - **Solution Overview**
  - **Guidelines**
- Evaluation
  - Benchmarks
  - Results
- Conclusion

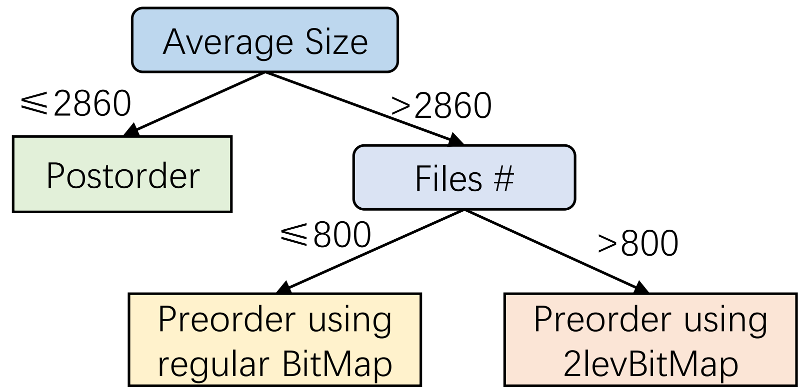
Solution Overview



Data Attributes

Problem: How to utilize the attributes of datasets



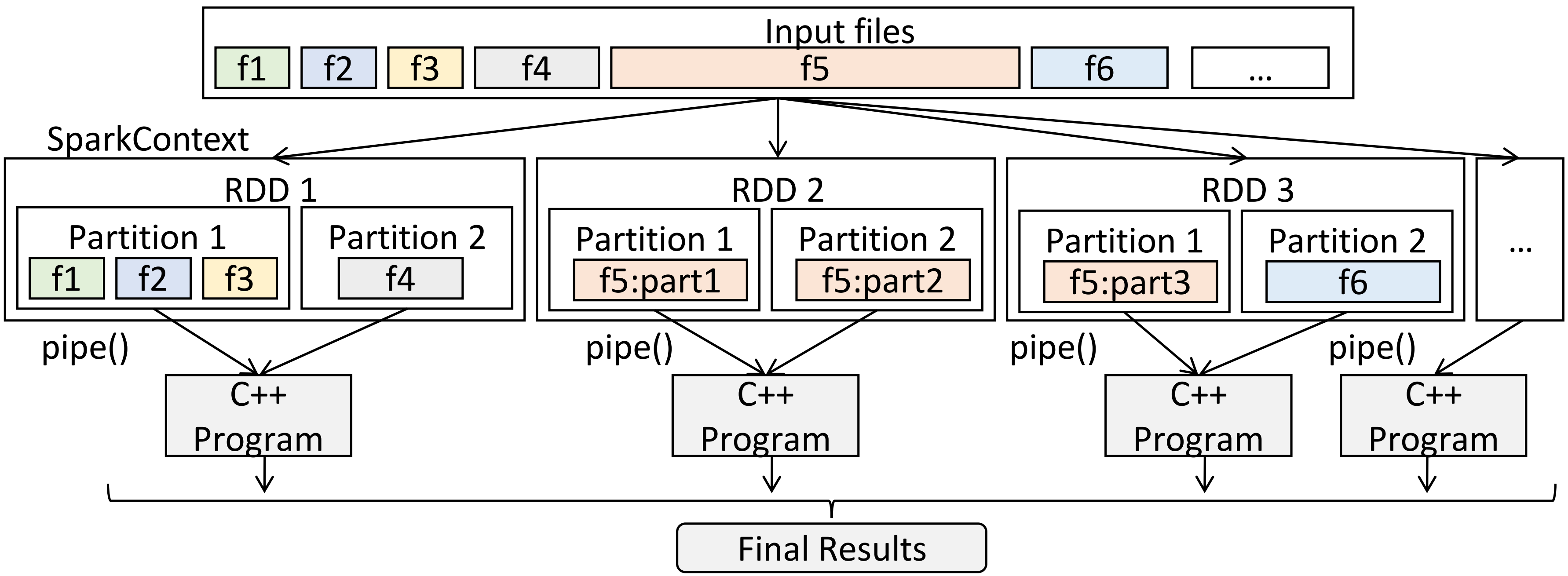


The best traversal order may depend on the data attributes of input.

- Guideline I: minimize the footprint size
- Guideline II: Traversal order is essential for the efficiency

Parallelism Barriers

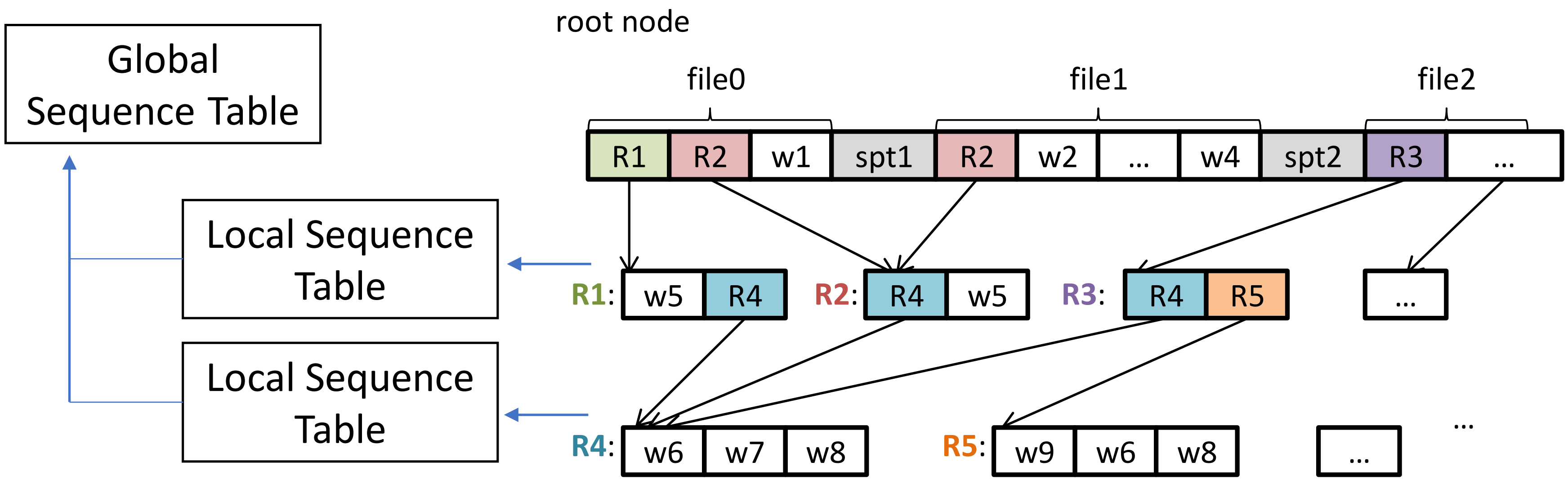
Problem: How to perform parallelism on large datasets



- **Guideline III: Coarse-grained distributed implementation is preferred**

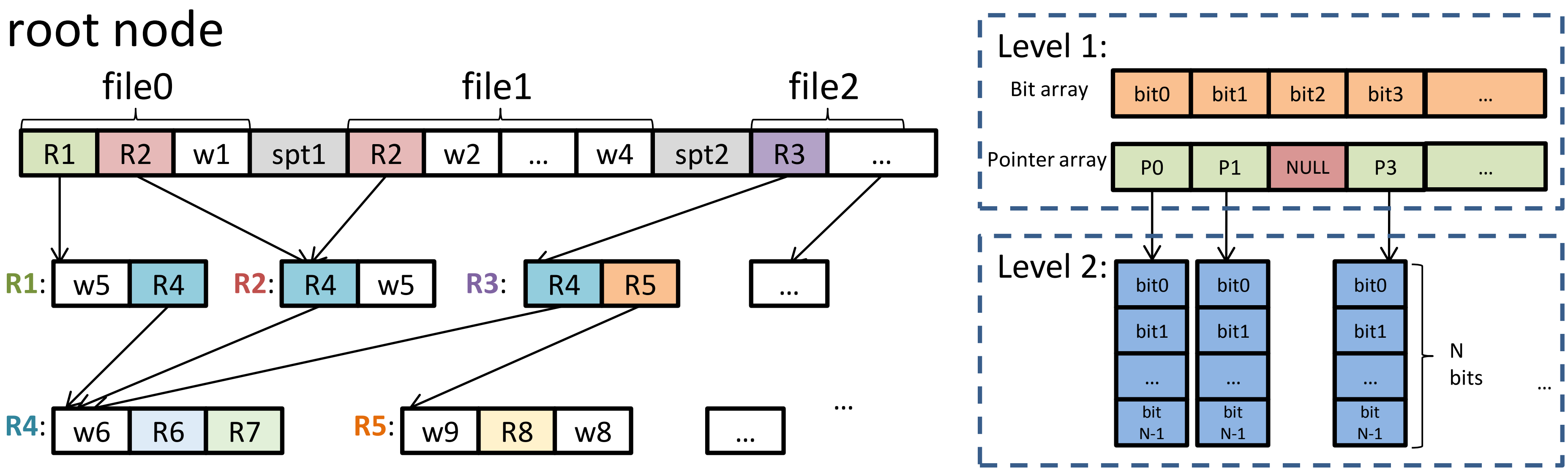
Order Sensitivity (detailed in paper)

Problem: Some applications are sensitive to the order



- Guideline IV: depth-first traversal and a two-level table design

Unit Sensitivity (detailed in paper)
Problem: How to organize data



- Guideline V: use of double-layered bitmap if unit information needs to be passed across the CFG

Short Summary for Six Guidelines

- Data Attributes Challenge
  - Guideline II
- Parallelism Barriers
  - Guideline III
- Order Sensitivity
  - Guideline IV
- Unit Sensitivity
  - Guideline V
- General insights and common techniques
  - Guideline I and Guideline VI

Outline

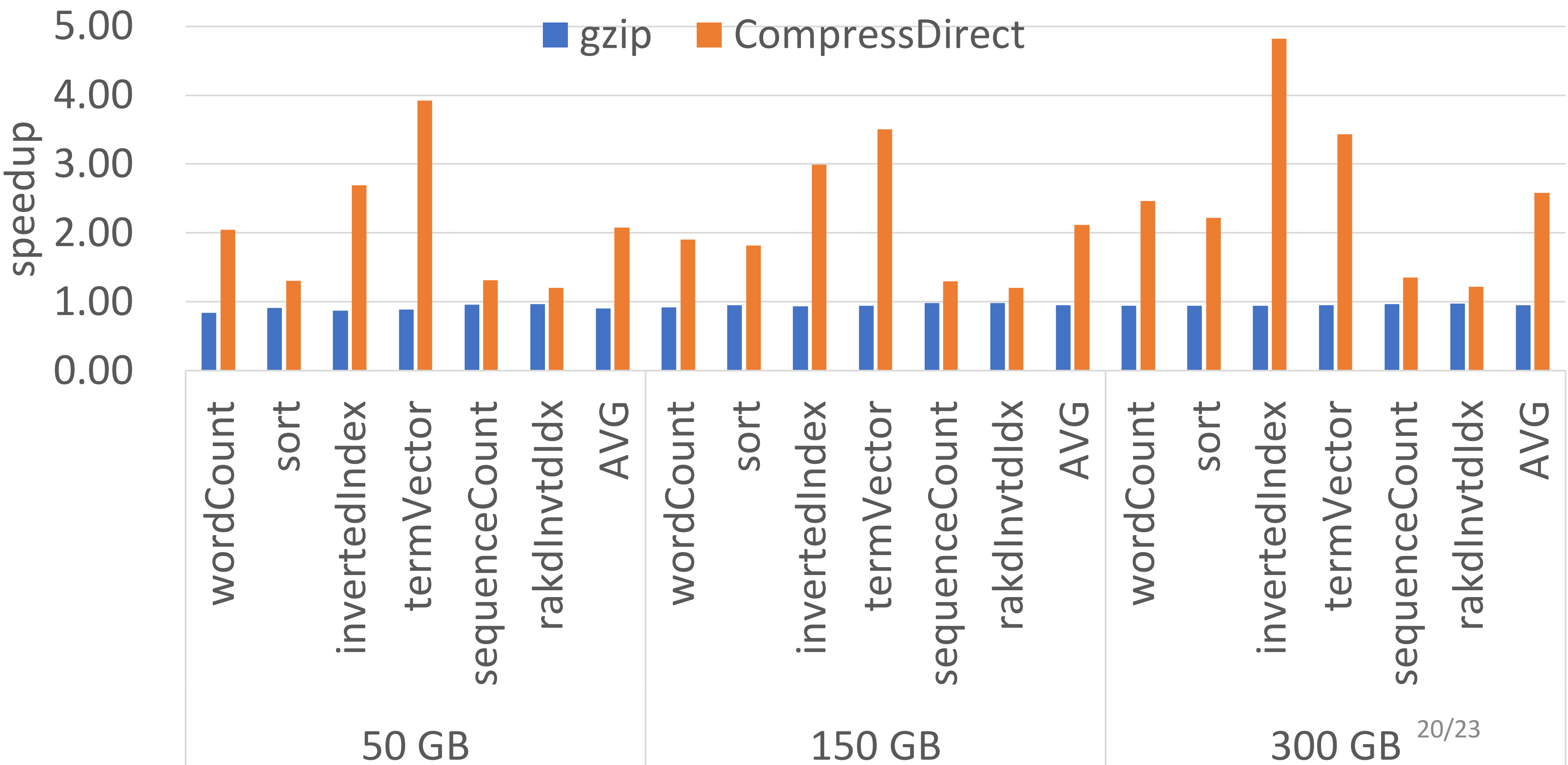
- Introduction
  - Motivation & Example
  - Compression-Based Direct Processing
  - Challenges
- Guidelines and Techniques
  - Solution Overview
  - Guidelines
- **Evaluation**
  - **Benchmarks**
  - **Results**
- Conclusion

Benchmarks

- Six benchmarks
  - Word Count, Inverted Index, Sequence Count, Ranked Inverted Index, Sort, Term Vector
- Five datasets
  - 580 MB ~ 300 GB
- Two platforms
  - Single node
  - Spark cluster (10 nodes on Amazon EC2)

Time Savings

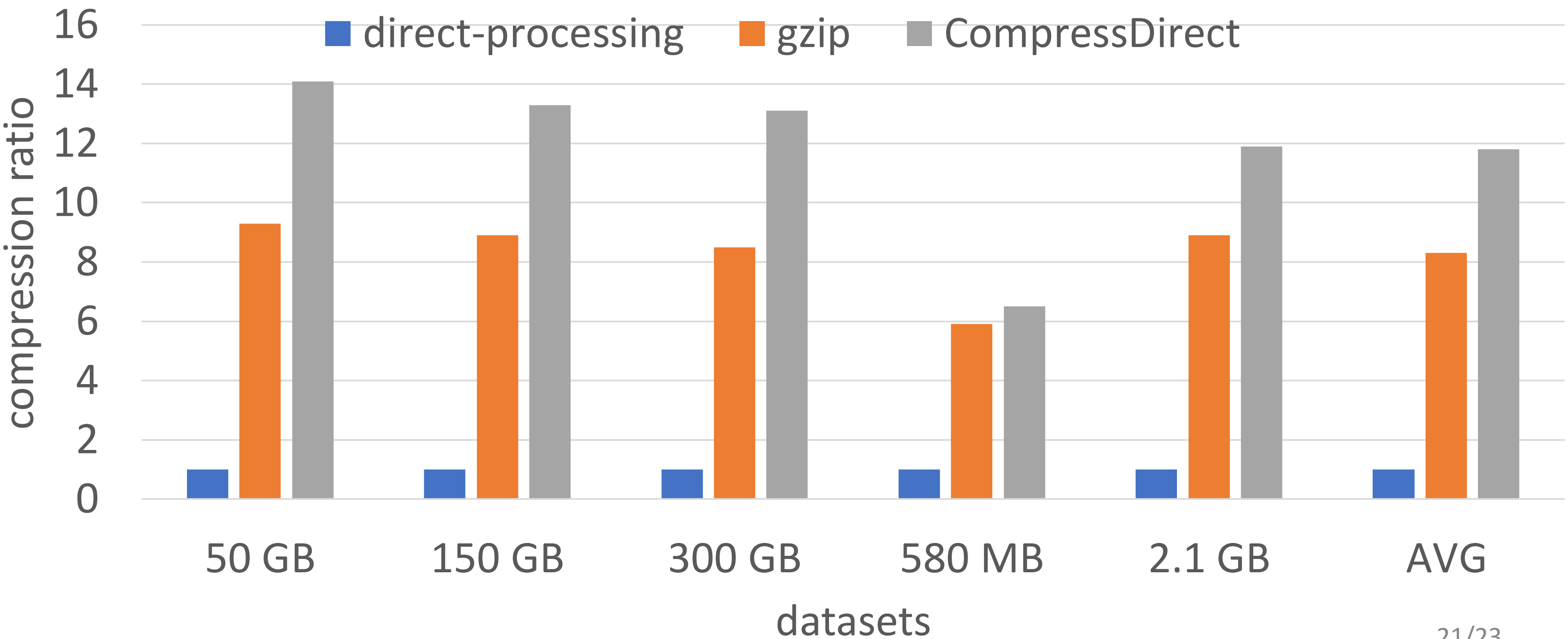
- *CompressDirect* yields 2X speedup, on average, over *direct processing*.



Space Savings

- *CompressDirect* achieves 11.8X compression ratio, even more than *gzip* does.

compression ratio = original size / compressed data size



Conclusion

- Our method, *compression-based direct processing*.
- How the concept can be materialized on Sequitur.
  - Major challenges.
  - Guidelines.
- Our library, *CompressDirect*, to help further ease the required development efforts.

Thanks!

- Any questions?

Feng Zhang †⋄, Jidong Zhai ⋄, Xipeng Shen #, Onur Mutlu ⋆, Wenguang Chen ⋄

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