CS SEMINAR

Research on Homomorphic Compression Database Technology

Feng Zhang

Renmin University of China

2024-5-28

About Myself

- Feng Zhang
  - Professor @ DB Group since 2023
  - Renmin University of China (RUC)
  - https://fengzhangcs.github.io/
- Short Bio
  - Visiting Scholar @ NUS (system) and NCSU (compiler)
  - Postdoc @ RUC, 2019 (database)
  - PhD @ Tsinghua, 2017 (HPC)

Outline

- Part 1: Application of homomorphic compression database technology in text data
- Part 2: Application of homomorphic compression database technology in database
- Part 3: Application of homomorphic compression database technology in graph data
- Part 4: Application of compressed database technology in stream data
- Part 5: Application of compressed database technology in ML data

Part 1: Application of homomorphic compression database technology in text data

- **1. Data Analytics directly on Compression**
  - 1.1 Background and Idea
  - 1.2 Example
  - 1.3 Compression-Based Direct Processing
- 2. Data Management directly on Compression
  - 2.1 Motivation
  - 2.2 Operations to Support
  - 2.3 Compression-Based Data Management
- 3. Homomorphic Compression
- 4. Hardware Acceleration

1.1 Background and Idea

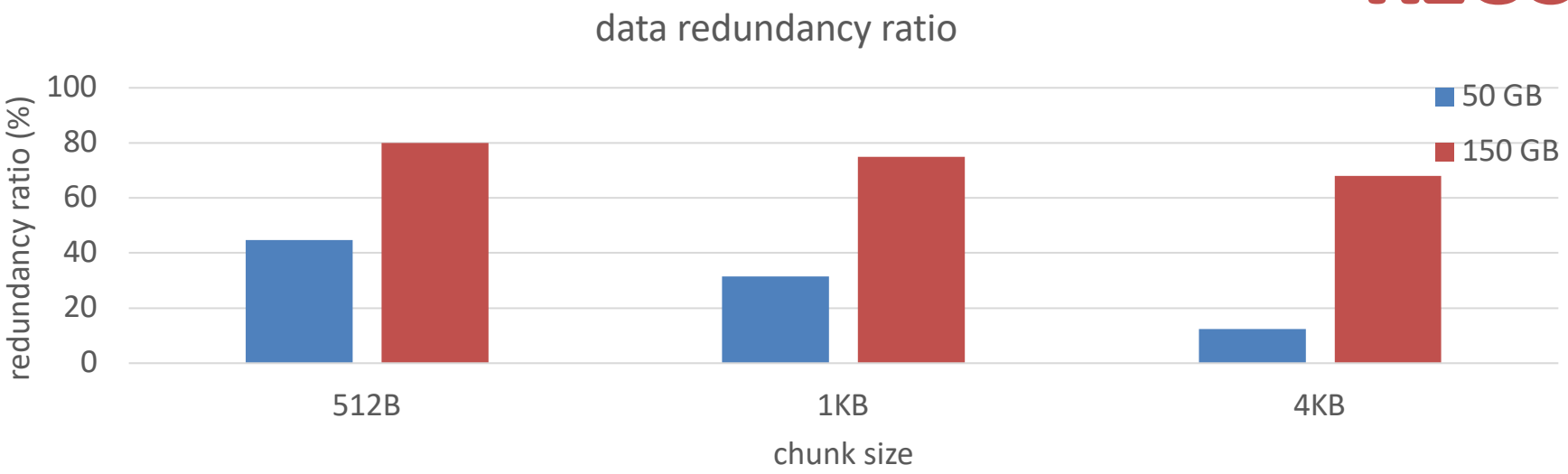
How to perform efficient document analytics when data are extremely large?

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

1.1 Background and Idea

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

REUSE



Whether we can perform analytics directly on compressed data?

1.1 Background and Idea

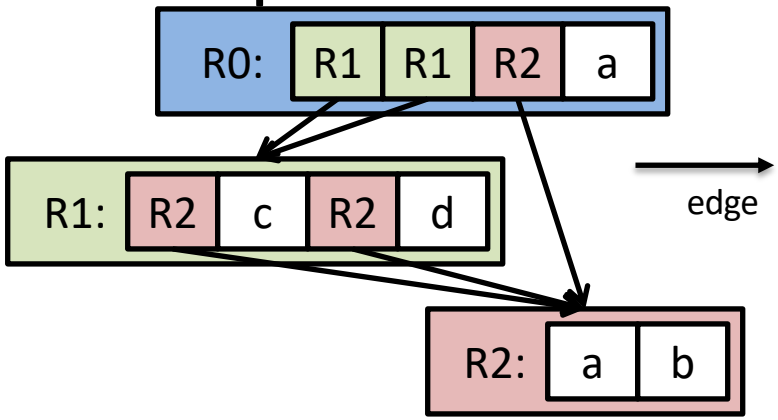
- Compression-based direct processing
- *Sequitur* algorithm meets our requirement

Input:

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

Rules:

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



(a) Original data (b) Rule-based compression (c) DAG Representation

a: 0 b: 1 c: 2 d: 3
R0: 4 R1: 5 R2: 6

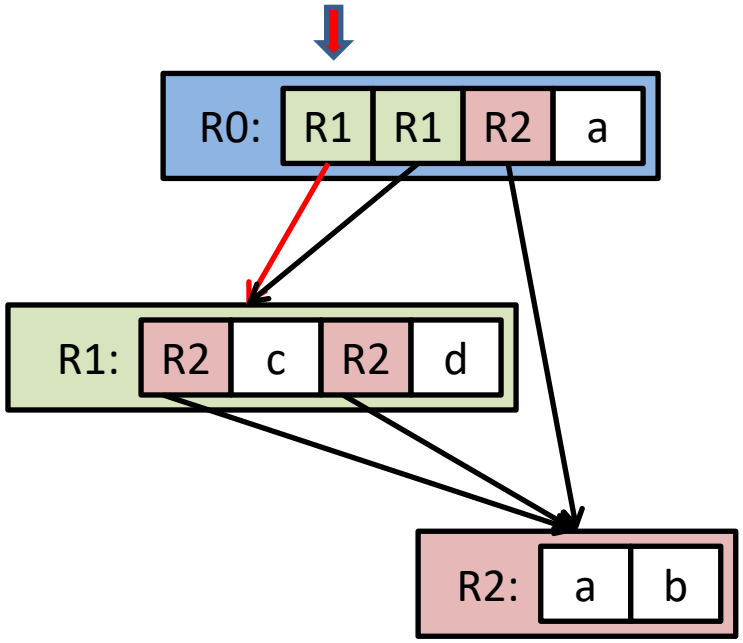
4 → 5 5 6 0
5 → 6 2 6 3
6 → 0 1

(d) Numerical representation (e) Compressed data in numerical ID

[**VLDB'18**] "Efficient Document Analytics on Compressed Data: Method, Challenges, Algorithms, Insights", Feng Zhang, Jidong Zhai, Xipeng Shen, Onur Mutlu, and Wenguang Chen. The 44th International Conference on Very Large Data Bases, Rio de Janeiro, Brazil, August 27-31, 2018.

1.2 Example

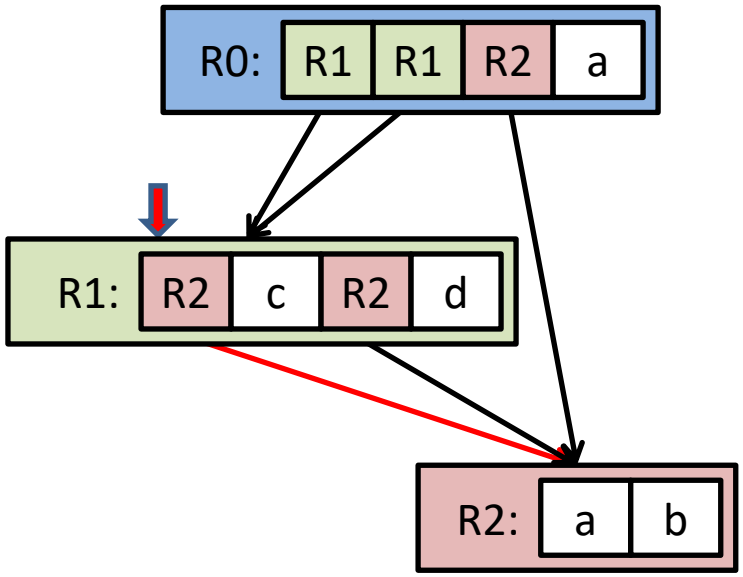
- How to go through data from DAG?



Output:

1.2 Example

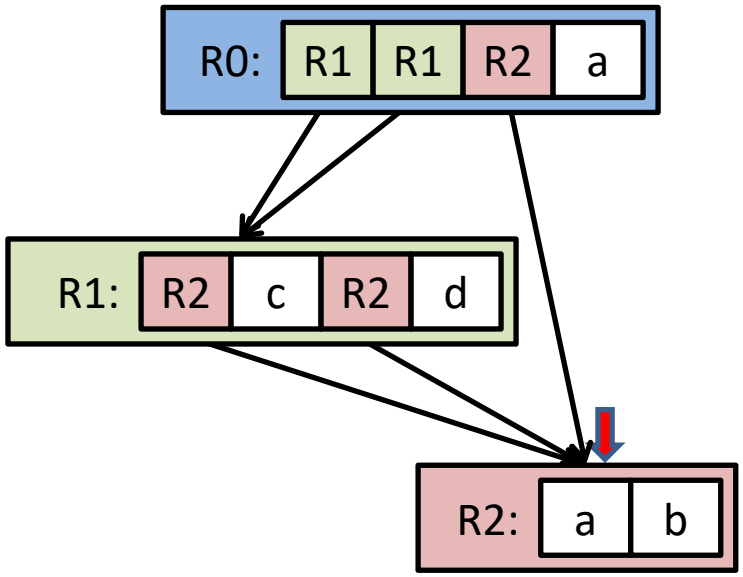
- How to go through data from DAG?



Output:

1.2 Example

- How to go through data from DAG?

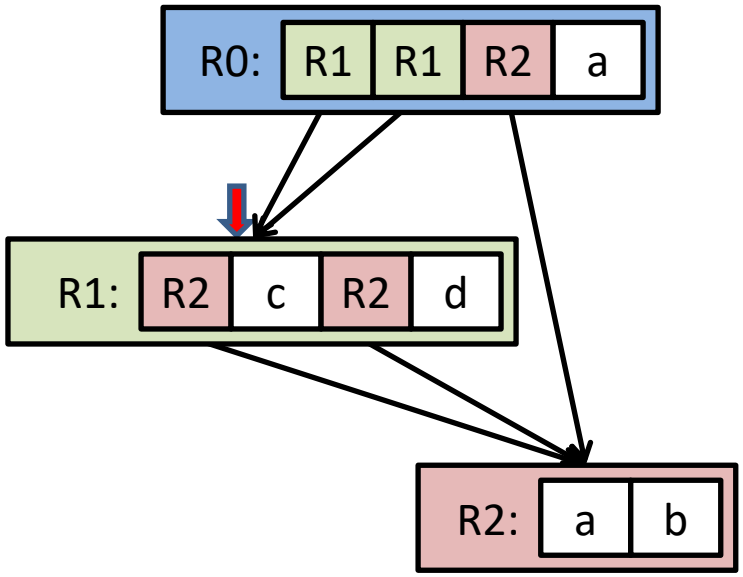


Output:

a b

1.2 Example

- How to go through data from DAG?

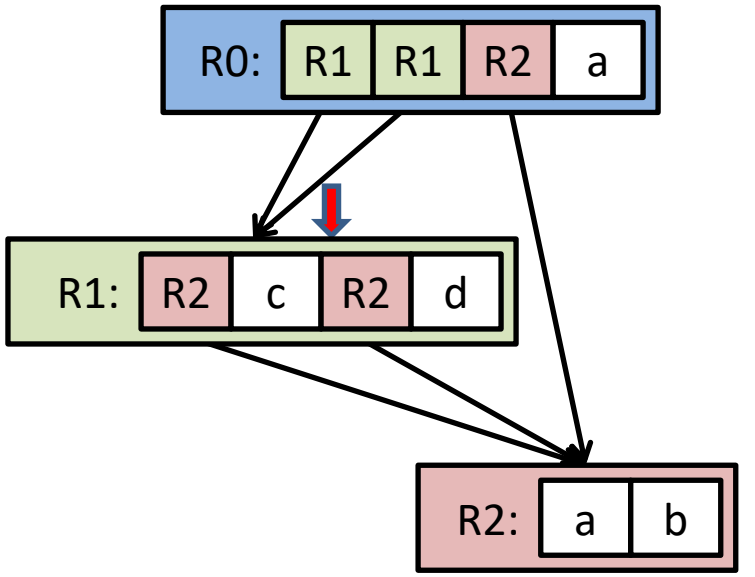


Output:

a b c

1.2 Example

- How to go through data from DAG?



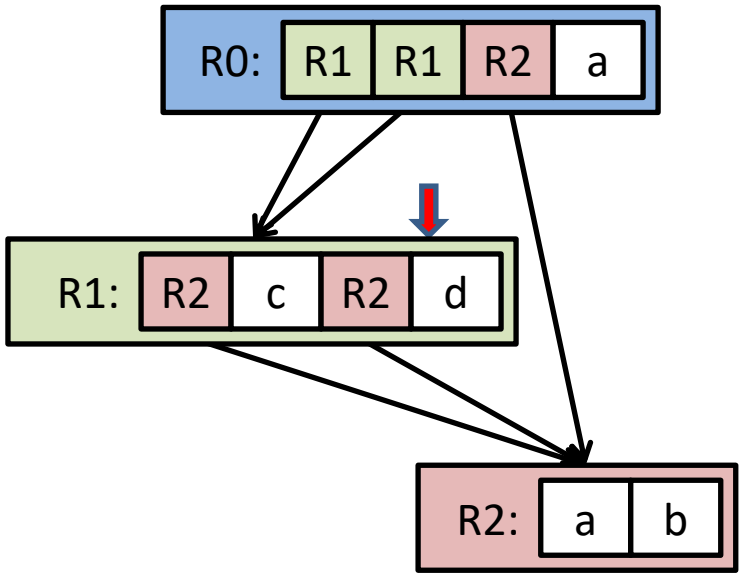
Output:

a b c a b

We do not need to go to R2!

1.2 Example

- How to go through data from DAG?

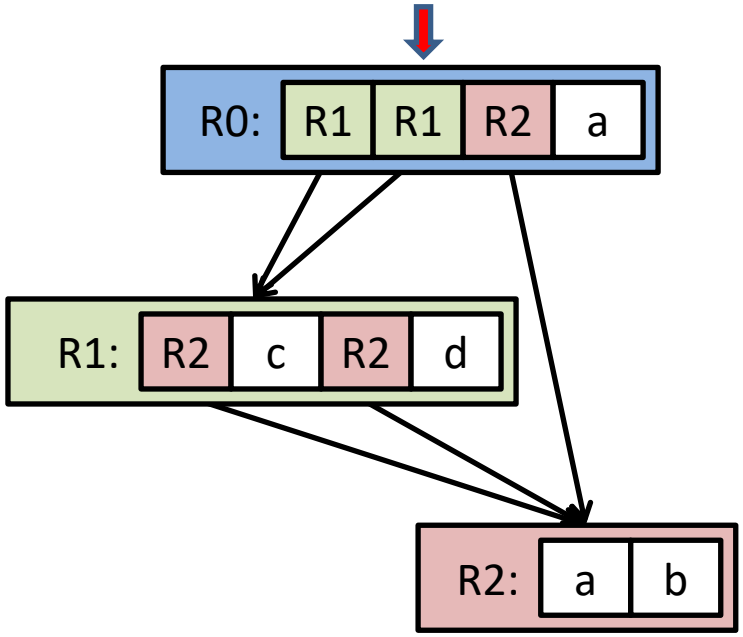


Output:

a b c a b d

1.2 Example

- How to go through data from DAG?



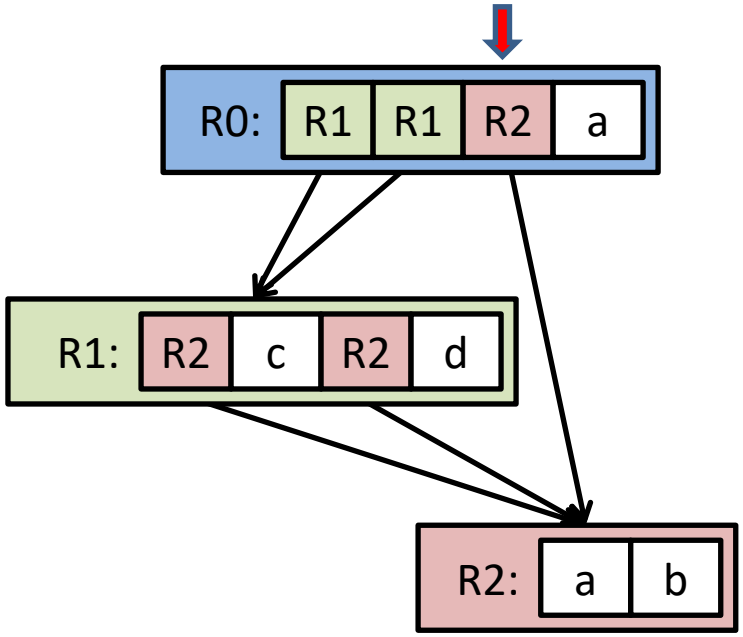
Output:

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

We do not need to go to R1!

1.2 Example

- How to go through data from DAG?



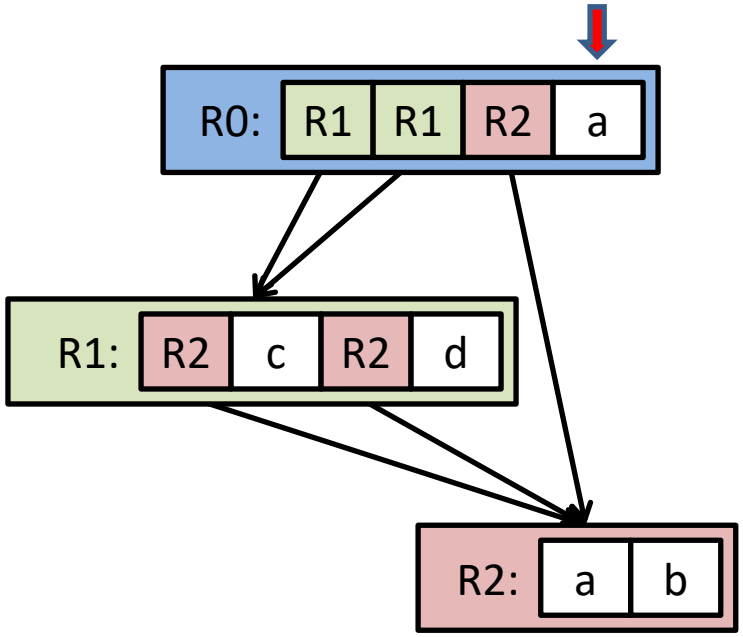
Output:

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

We do not need to go to R2!

1.2 Example

- How to go through data from DAG?

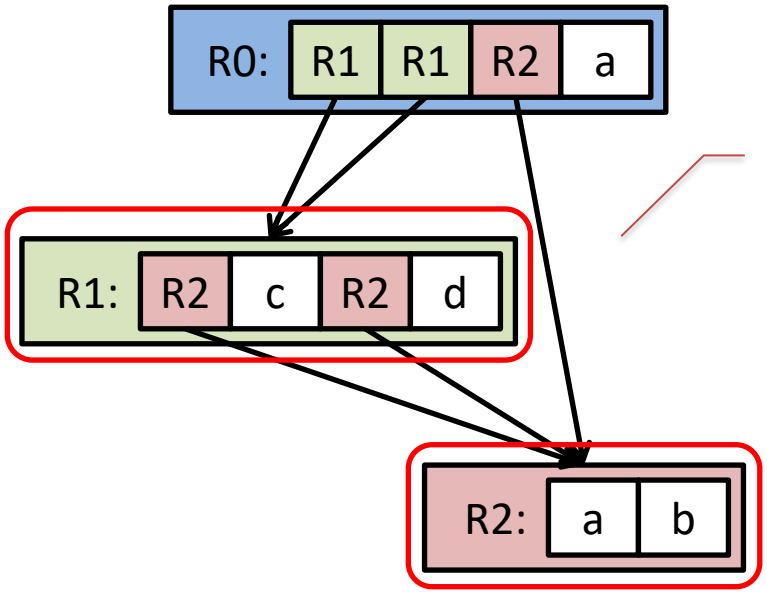


Output:

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

1.2 Example

- Double benefits



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

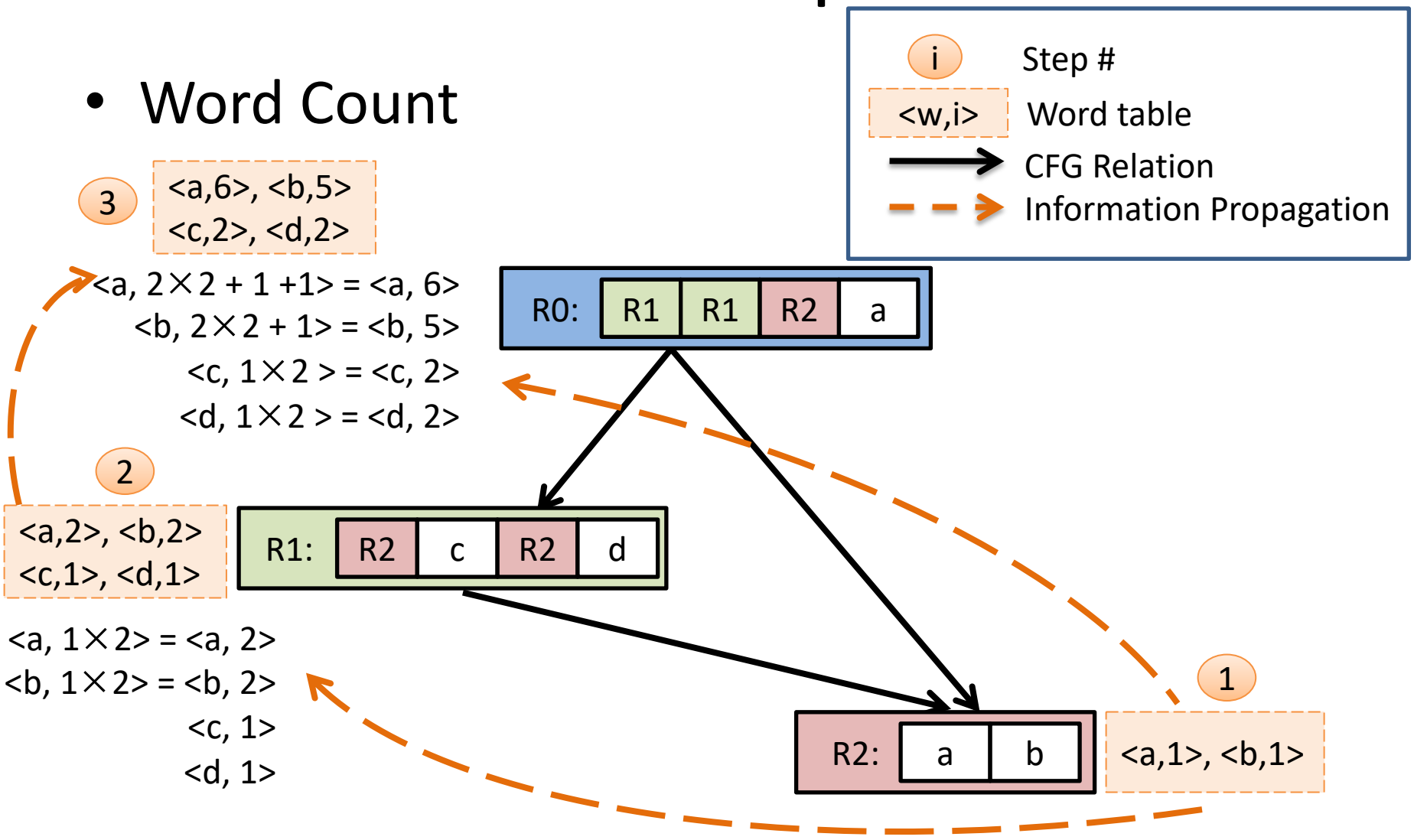
Challenge 1: Space

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

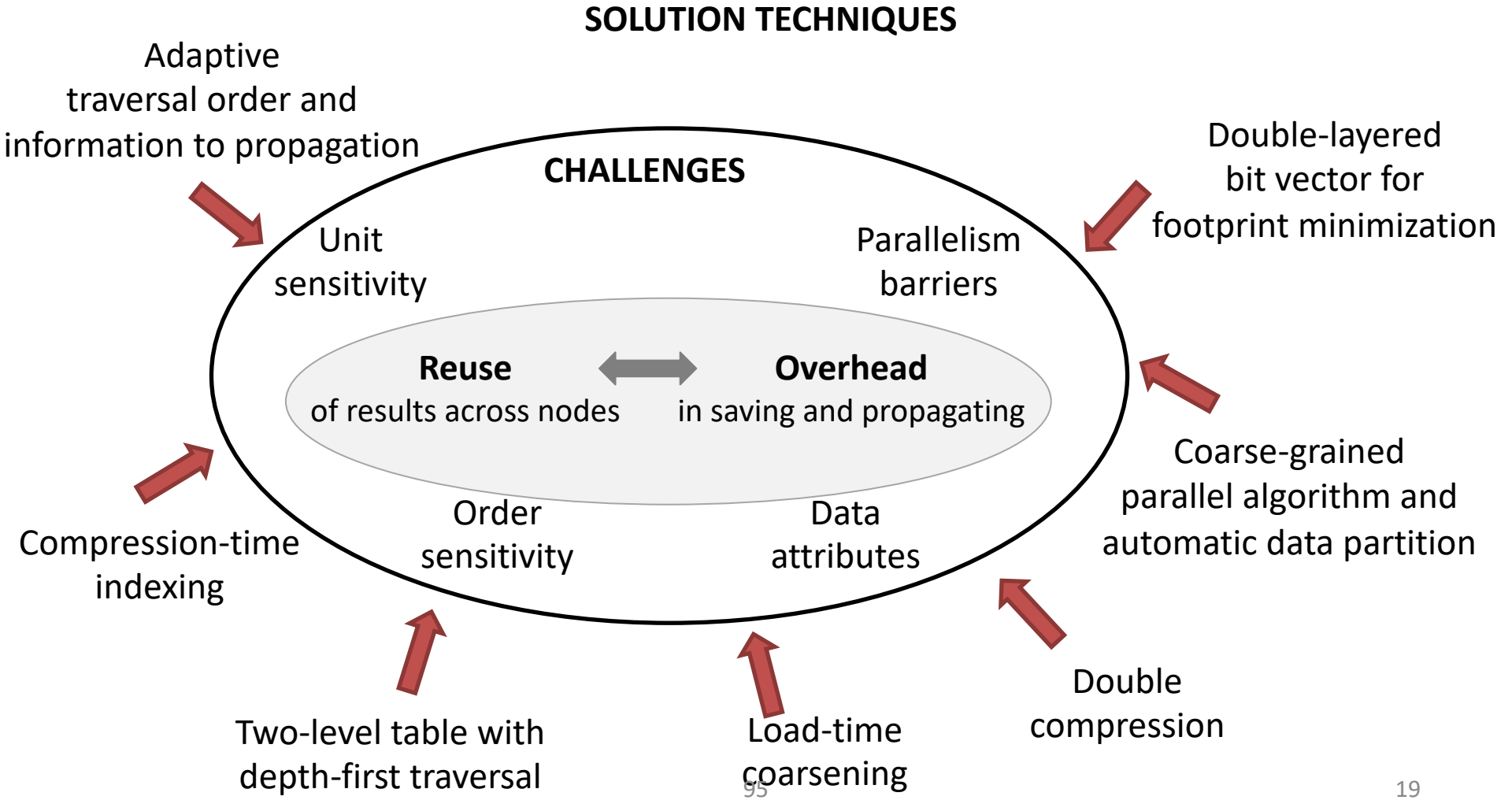
Challenge 2: Time

1.2 Example

- Word Count



1.3 Compression-Based Direct Processing

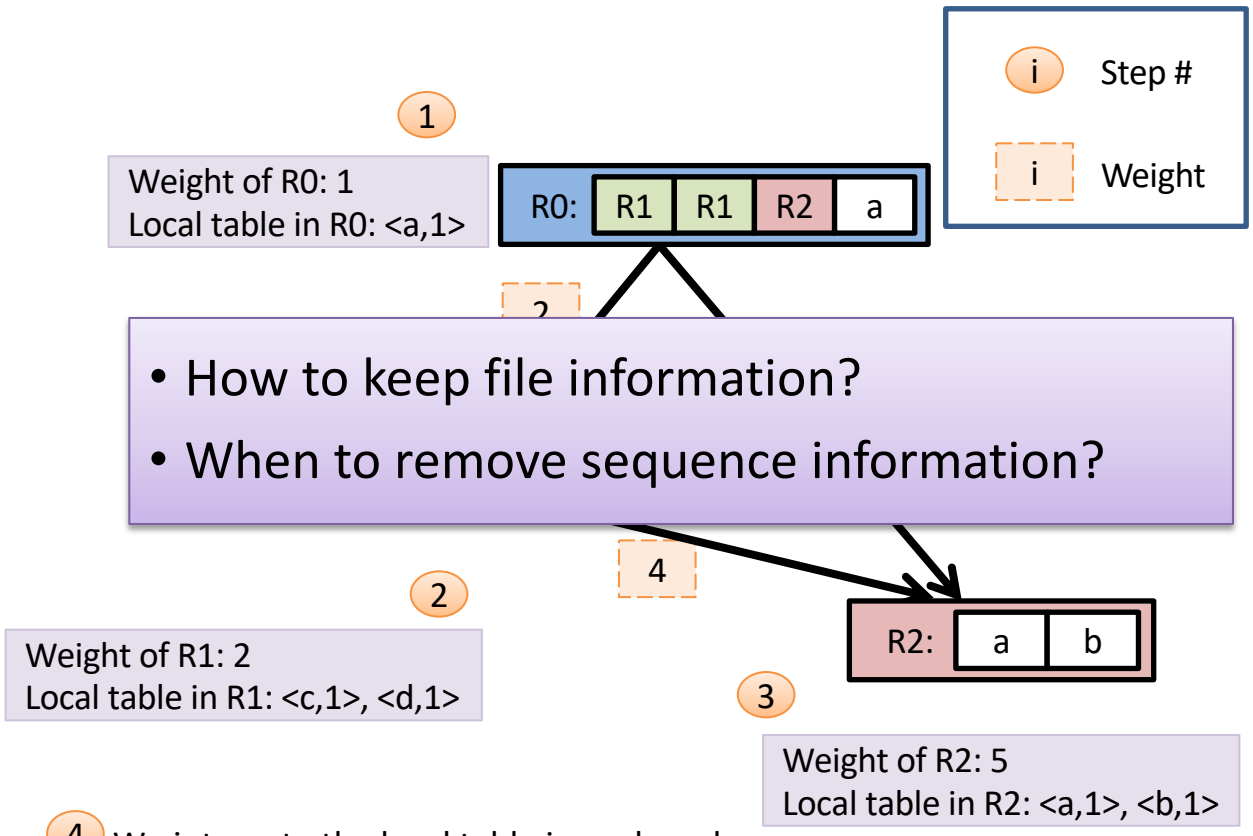


1.3.1 Programming Challenges

Problem dimension

Implementation dimension

Dataset dimension

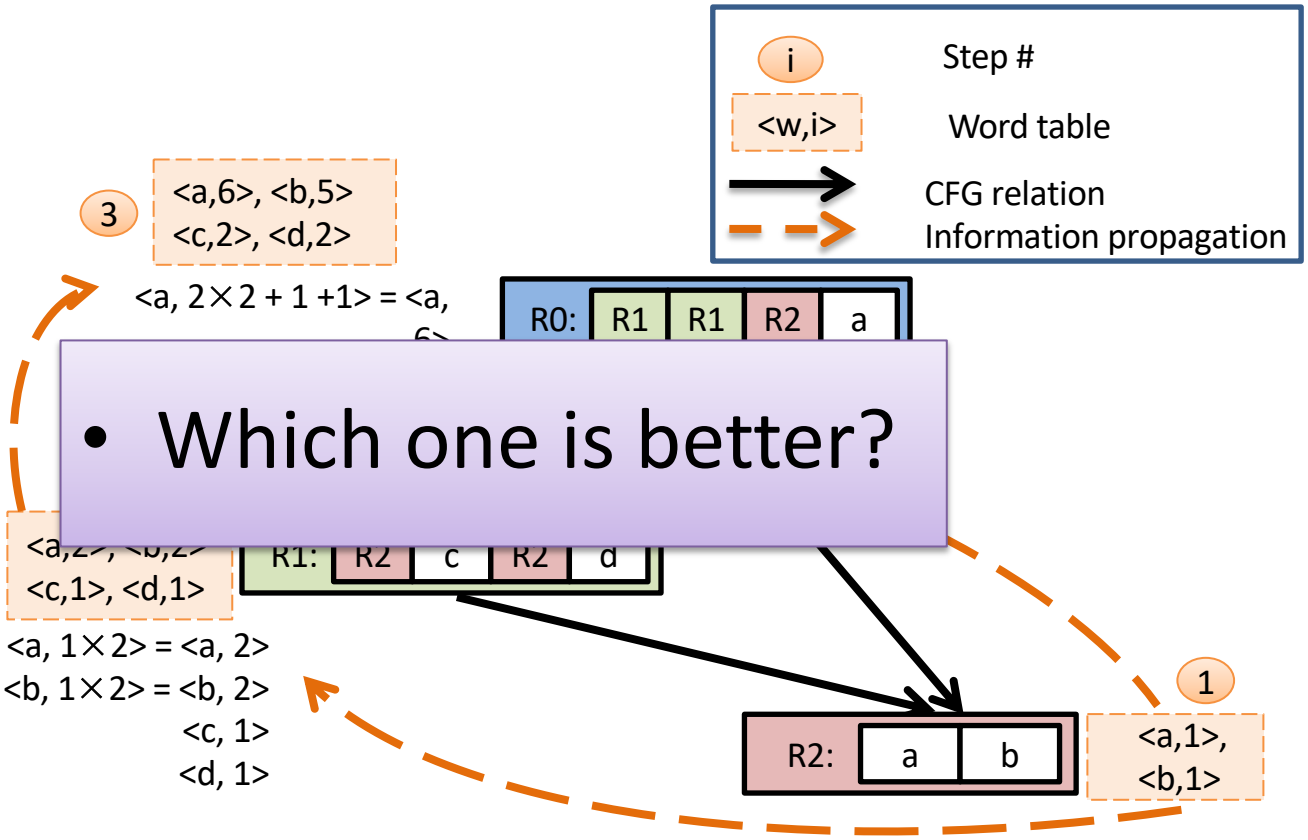


1.3.1 Programming Challenges

Problem dimension

Implementation dimension

Dataset dimension



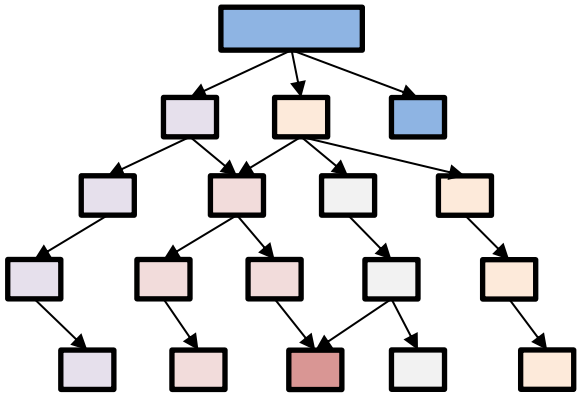
- Which one is better?

1.3.1 Programming Challenges

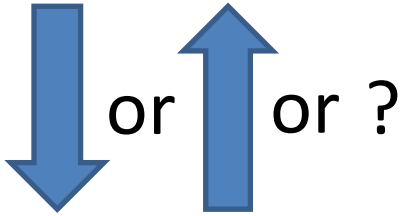
Problem dimension

Implementation dimension

Dataset dimension



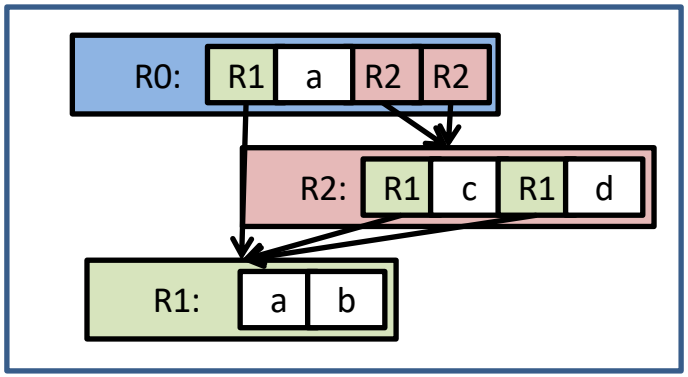
Traversal Order



The best traversal order may depend on input.

1.3.2 Solution

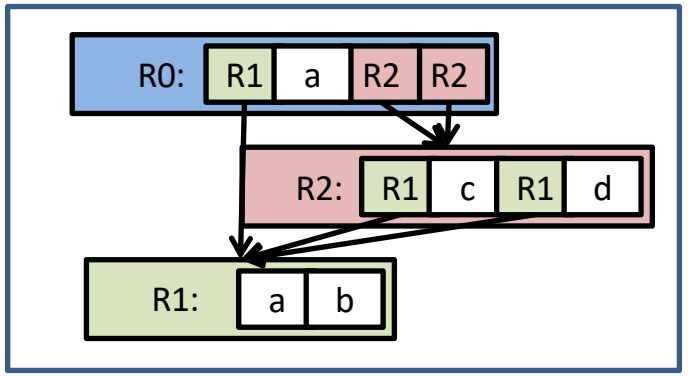
- A conceptual framework, a six-element tuple (G, V, F, ⋁,D, ⋀)
  - A graph G, DAG representation



- G, how to represent it.

1.3.2 Solution

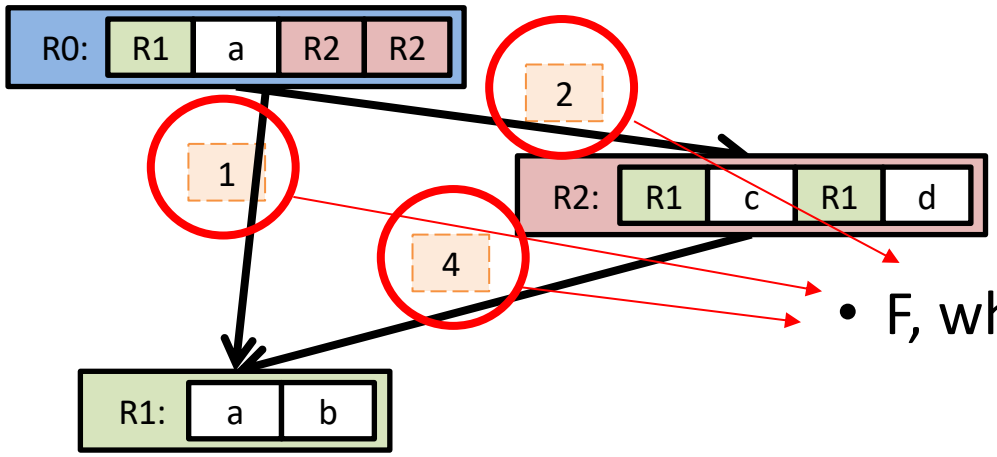
- A conceptual framework, a six-element tuple (G, V, F, V,D, Λ)
  - A domain of values V, possible values as the outputs



- V, what do we want to get from G?

1.3.2 Solution

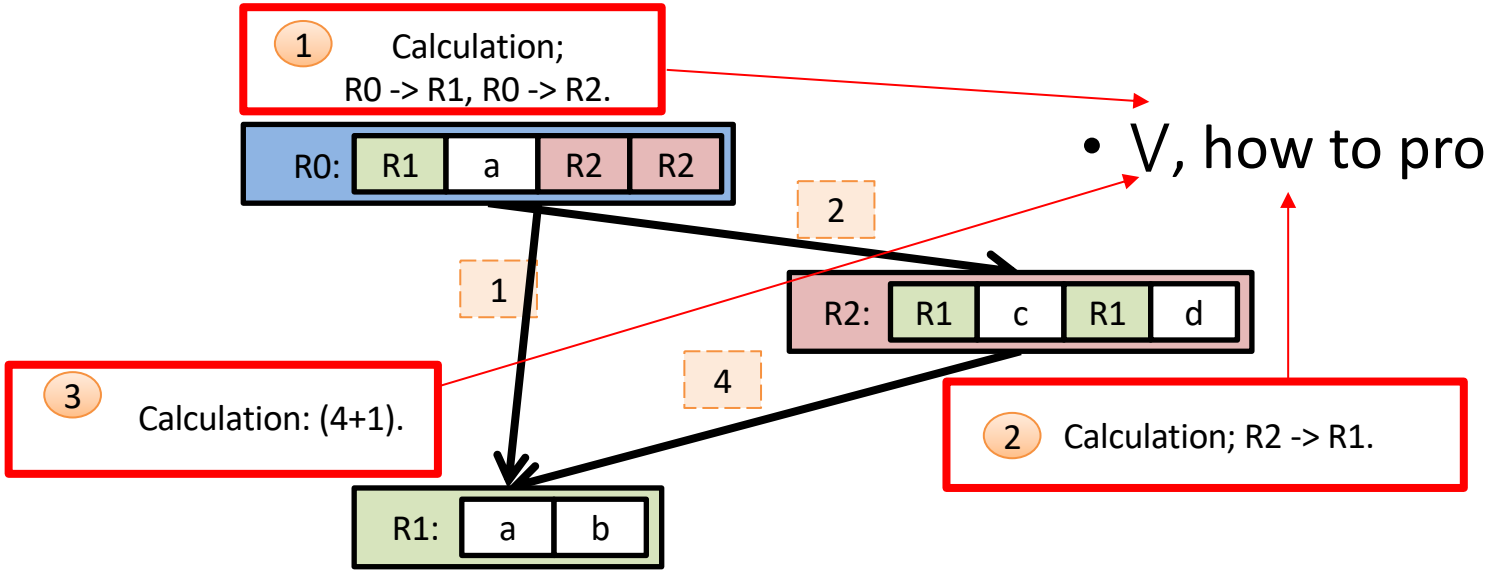
- A conceptual framework, a six-element tuple (G, V, F, ⋁,D, ∧)
  - A domain of values F, possible values to be propagated



- F, what to propagate?

1.3.2 Solution

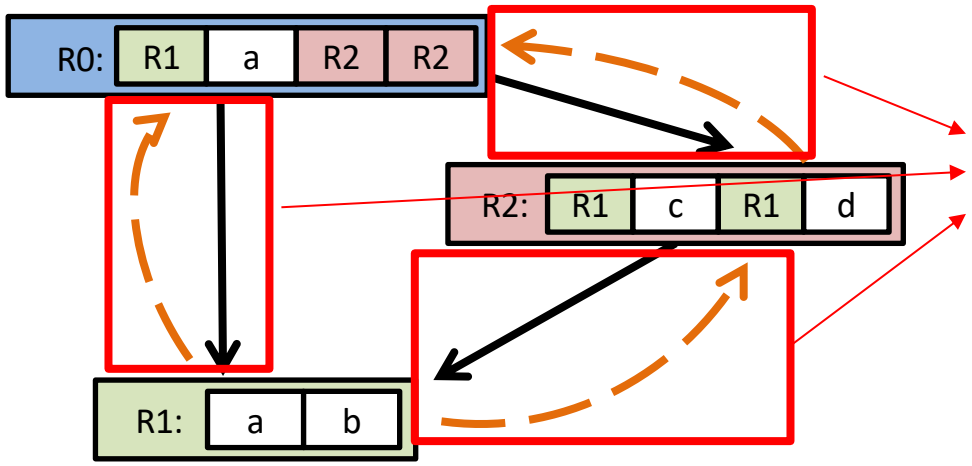
- A conceptual framework, a six-element tuple (G, V, F, ⋁,D, ∧)
  - A meet operator ⋁, how to transform values



- ⋁, how to propagate values?

1.3.2 Solution

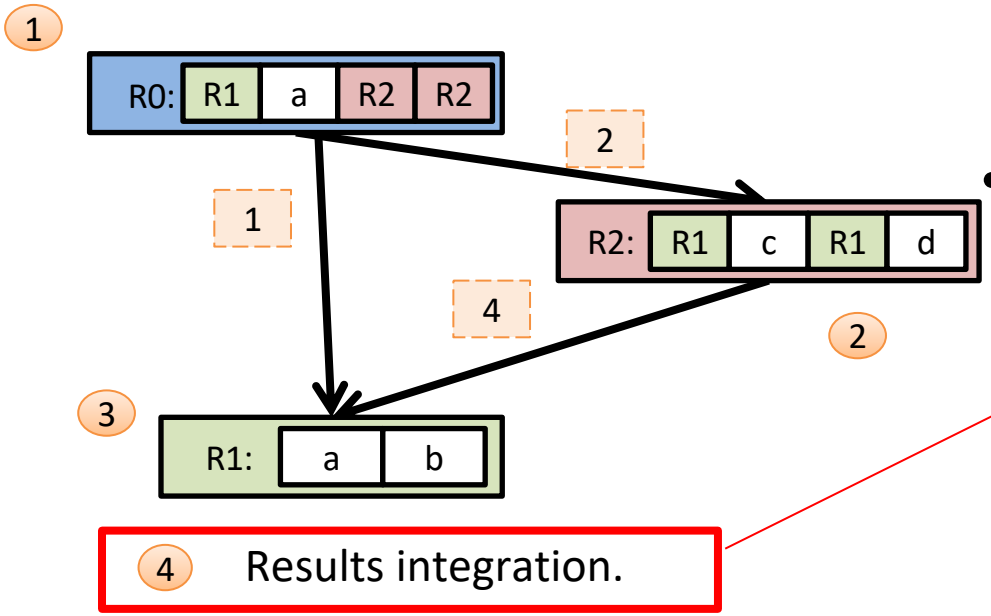
- A conceptual framework, a six-element tuple (G, V, F, ⋁,D, ⋀)
  - A direction of the value flow D



- D, what direction?

1.3.2 Solution

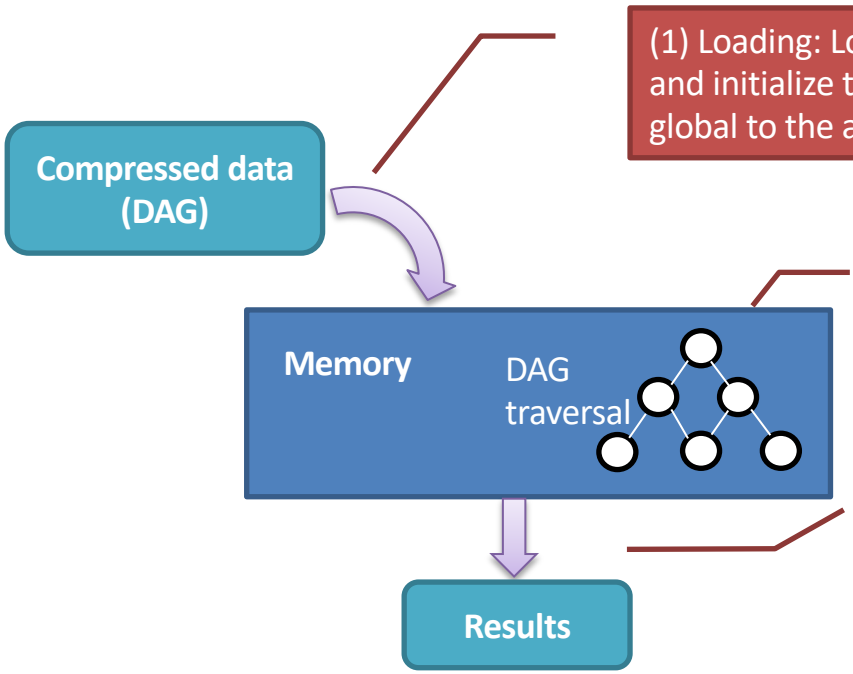
- A conceptual framework, a six-element tuple (G, V, F, ⋁,D, ⋀)
  - A gleaning operator ⋀, final stage of the analytics



- ⋀, the last step, how to generate results?

1.3.2 Solution

- High-level algorithm for TADOC:



(1) Loading: Load the grammar, build G (or its variants), and initialize the data structures local to each node or global to the algorithm.

(2) Propagation: Propagate information with the meet operator $\vee$ while traversing G in direction D.

(3) Gleaning: Glean information through the gleaning operator $\wedge$ and output the final results.

1.3.2 Solution

- How to generalize the idea of TADOC
- Zwift, the first programming framework for TADOC
  - The Zwift language
  - A compiler and runtime
  - A utility library

1.3.3 System Design and Implementation

- The Conceptual Framework

  (G, V, F, ⋁,D, ∧)
  - A graph G.
  - A domain of values V.
  - A domain of values F.
  - A meet operator ⋁.
  - A direction of the value flow D.
  - A gleaning operator ∧.

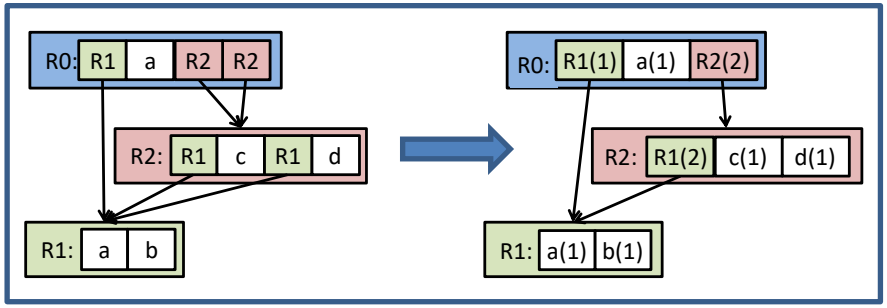
Zwift DSL template.

```
01 ELEMENT = LETTER/WORD/SENTENCE
02 USING_FILE = true/false
03 NodeStructure = {
04 //data structures in each node
05 }
06 Init = {
07 //initializations for each node in ZwiftDAG
08 }
09 Direction = FORWARD/BACKWARD/DEPTHFIRST
10 Action = {
11 //actions taken at each node during a traversal
12 }
13 Result = {
14 // result structure
15 }
16 FinalStage = {
17 // final operations to produce the results
18 }
```

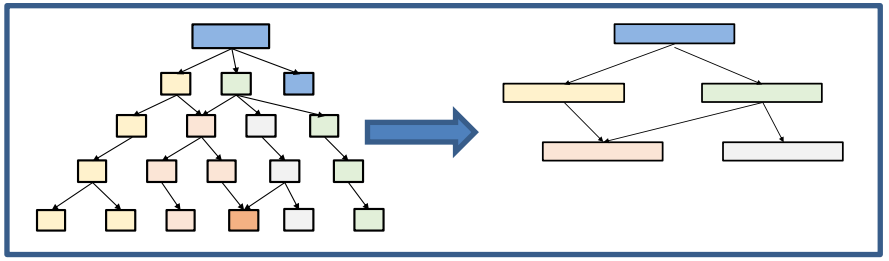
The Zwift Language

1.3.3 System Design and Implementation

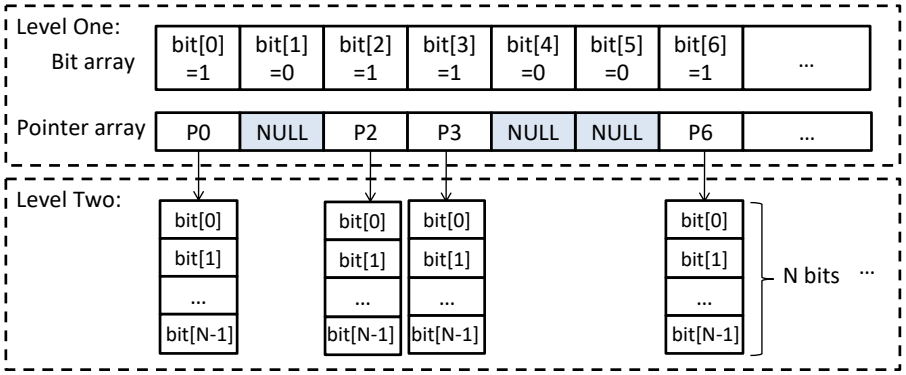
- Edge Merging



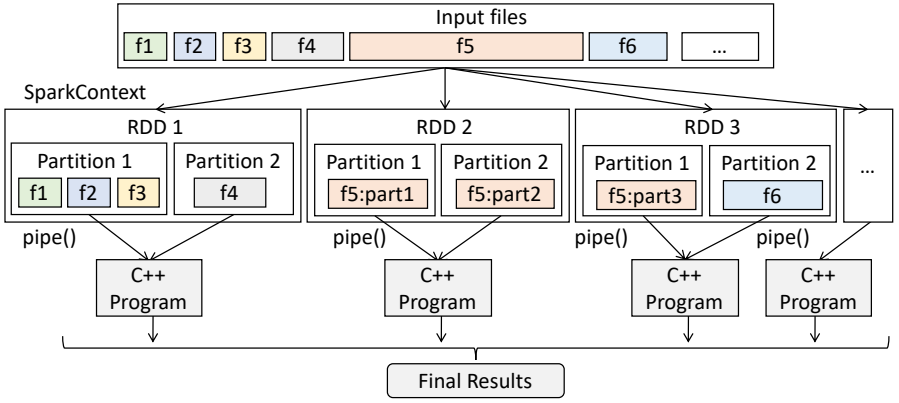
- Node Coarsening



- Two-Level Bit Vector



- Coarse-Grained Parallelism



1.3.3 System Design and Implementation

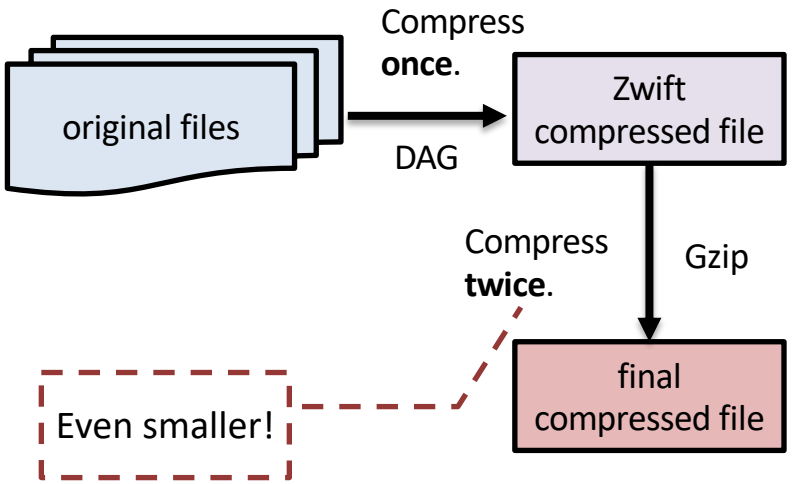
- Version Selection
  - Used to choose from different versions

```
...
35 Action[0] = {
36   for(ZwiftVec <FILEID,ZwiftBit>::iterator i=
37     PARENT.file.begin();
38   i!= PARENT.file.end(); i++){
39     CHILD.file[*i]=1;
40   }
41 }
42 Action[1] = {
43   for(ZwiftSet <ELEMENT>::iterator i=
44     CHILD.wordSet.begin();
45   i!= CHILD.wordSet.end(); i++){
46     PARENT.wordSet.insert(*i);
47   }
48 }
...
```

V1

V2

- Double Compression



[**ICS'18**] "Zwift: A Programming Framework for High Performance Text Analytics on Compressed Data", Feng Zhang, Jidong Zhai,Xipeng Shen, Onur Mutlu, and Wenguang Chen. The 32nd ACM International Conference on Supercomputing, Beijing, China, June 12-15, 2018.

2. Data Management directly on Compression

2.1 Motivation

News

Legal files

```
06:52:19  Info        AboutLog 170       0          "Changing App Context: {0: 1} -> {0: 4097}"
06:52:19  Info        Status   163       0          "INSTALL SPACE REQUIRED: {0: 10169266} KB; SPACE AVAILABLE: {1: 123510776} KB; LOC: {2: C:\}"
06:52:19  Info        Step     25        55502      "User advanced from '{0: Connecting}' page to '{1: Summary}' page."
06:52:19  Info        Selection 38       55612      "The following lines show the default selections for the Summary page when it first appeared."
06:52:19  Info        Selection 99       0          "This is a new installation of 2016 SP1.0."
06:52:19  Info        Selection 102      0          "{Download Options: {Operation:=Install only (do not download) (Enabled: 1)}, {Background downloader:=Do not use (Enabled: 1)}} (Enable
06:52:19  Info        Selection 103      0          "{Installation Location: {Installation location:=C:\Program Files\SOLIDWORKS Corp (Enabled: 1)}, {Install from:=G:\SOLIDWORKS DOWNLO
06:52:19  Info        Selection 104      0          "{Toolbox/Hole Wizard Options: {Toolbox installation location:=C:\SOLIDWORKS Data (Enabled: 1)}, {Toolbox installation method:=New Tool
06:52:19  Info        Selection 105      0          "Estimated installation size: 9.7 GB"
06:52:23  Info        Step     25        55502      "User advanced from '{0: Summary}' page to '{1: Products To Install}' page."
06:53:24  Info        Selection 37       55611      "The user clicked the button representing the {0: OK} operation on the {1: Products To Install} page."
06:53:24  Info        Selection 36       55610      "The user changed the installation action of {0: PhotoView 360 Network Render Client} to: {1: Install PhotoView 360 Network Render Client 2
06:53:24  Info        Selection 36       55610      "The user changed the installation action of {0: SOLIDWORKS Electrical} to: {1: Do not install SOLIDWORKS Electrical 2016 SP1.0}"
06:53:24  Info        Selection 36       55610      "The user changed the installation action of {0: API Tools} to: {1: API Tools 2016 SP1.0 (install manually)}"
06:53:24  Info        Selection 36       55610      "The user changed the installation action of {0: SOLIDWORKS Workgroup PDM x64 Client Add-in} to: {1: Do not install SOLIDWORKS Work
06:53:24  Info        Selection 36       55610      "The user changed the installation action of {0: SOLIDWORKS Workgroup PDM Add-in} to: {1: Do not install SOLIDWORKS Workgroup PDM
06:53:24  Info        Status   163       0          "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
06:53:25  Info        Status   163       0          "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
06:53:25  Info        Step     25        55502      "User advanced from '{0: Products To Install}' page to '{1: Summary}' page."
06:53:27  Info        Step     25        55502      "User advanced from '{0: Summary}' page to '{1: Download Options}' page."
06:53:33  Info        Selection 37       55611      "The user clicked the button representing the {0: OK} operation on the {1: Download Options} page."
06:53:33  Info        Step     25        55502      "User advanced from '{0: Download Options}' page to '{1: Summary}' page."
06:53:37  Info        Step     25        55502      "User advanced from '{0: Summary}' page to '{1: Installation Location}' page."
06:53:48  Warning     Message  9         55200      "Dialog Shown: {0: C:\Program Files\SOLIDWORKS 2016 This path does not exist. Do you want to create it now?}. User selected: {1: Yes}"
06:53:48  Info        Status   163       0          "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
06:53:48  Info        Step     23        55600      "Page '{0: Installation Location}' is being reshown or refreshed."
06:53:50  Info        Selection 37       55611      "The user clicked the button representing the {0: OK} operation on the {1: Installation Location} page."
06:53:50  Info        Status   163       0          "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
06:53:50  Info        Status   163       0          "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506652} KB; LOC: {2: C:\}"
06:53:50  Info        Step     25        55502      "User advanced from '{0: Installation Location}' page to '{1: Summary}' page."
06:53:56  Info        Step     25        55502      "User advanced from '{0: Summary}' page to '{1: Toolbox Options}' page."
06:54:37  Info        Step     23        55500      "Page '{0: Toolbox Options}' is being reshown or refreshed."
06:54:43  Info        Selection 37       55611      "The user clicked the button representing the {0: OK} operation on the {1: Toolbox Options} page."
06:54:44  Info        Status   163       0          "INSTALL SPACE REQUIRED: {0: 9411784} KB; SPACE AVAILABLE: {1: 123506648} KB; LOC: {2: C:\}"
06:54:44  Info        Step     25        55502      "User advanced from '{0: Toolbox Options}' page to '{1: Summary}' page."
06:54:58  Info        Selection 37       55611      "The user clicked the button representing the {0: <U+N<U>ext} operation on the {1: Summary} page."
06:54:58  Info        Selection 39       55613      "The following lines show the selections for the Summary page when the user selected to begin a download or installation."
06:54:58  Info        Selection 99       0          "This is a new installation of 2016 SP1.0."
06:54:58  Info        Selection 102      0          "{Download Options: {Operation:=Install only (do not download) (Enabled: 1)}, {Background downloader:=Do not use (Enabled: 1)}} (Enable
06:54:58  Info        Selection 103      0          "{Installation Location: {Installation location:=C:\Program Files\SOLIDWORKS 2016 (Enabled: 1)}, {Install from:=G:\SOLIDWORKS DOWNLO
06:54:58  Info        Selection 104      0          "{Toolbox/Hole Wizard Options: {Toolbox installation location:=C:\Program Files\SOLIDWORKS 2016\SOLIDWORKS 2016 DATA (Enabled: 1)}
06:54:58  Info        Selection 105      0          "Estimated installation size: 9.0 GB"
06:54:58  Info        Selection 43       55617      "The source folder that will be used is: {0: G:\SOLIDWORKS DOWNLOADS\SOLIDWORKS 2016 x64 SP01}"
06:54:58  Info        Selection 44       55617      "The installation Manager Source Folder that will be used is: {0: G:\SOLIDWORKS DOWNLOADS\SOLIDWORKS 2016 x64 SP01\sldIM}"
06:54:59  Info        Step     25        55502      "User advanced from '{0: Summary}' page to '{1: Install Progress}' page."
06:54:59  Info        Selection 107      0          "Writing serial numbers to the registry..."
```

Logs

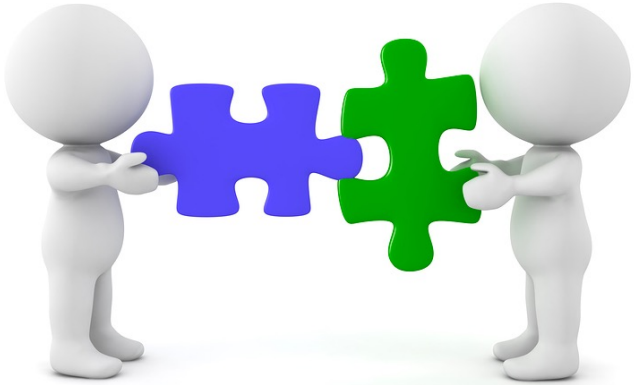
Random Access

- search
- extract
- count
- insert
- append

2.2 Operations to Support

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

Locality

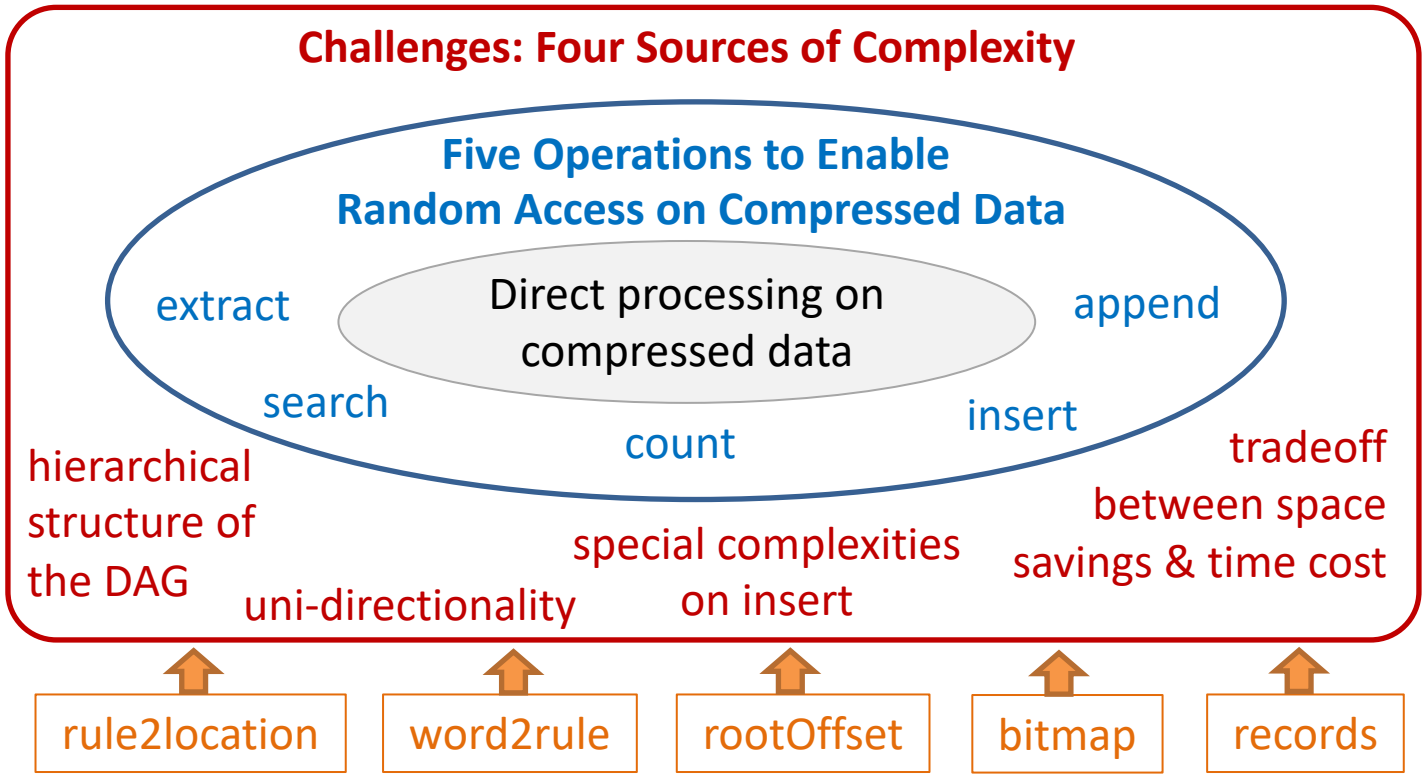


Compatibility

User Transparency

2.3 Compression-Based Data Management

Solution Techniques: Five Data Structures



3. Homomorphic Compression

- **Limitations of current "*operating on compressed data*" research**
  1. *Conception level:*
     - Lack of consensus and comprehensive *understanding* regarding the concept of *"operating on compressed data"*.
       - Definitions and properties.
  2. *System level:*
     - The *tight coupling* between upper-level application implementation and the internal mechanisms of underlying compression algorithms increases user operational costs.
  3. *Algorithm level:*
     - Lack support for *modification* tasks, e.g., insert and delete.
     - Record modifications in *separate data structures* instead of reflecting directly in the data.

Overview of Homomorphic Compression

1. We borrow the concept of *homomorphism* from algebra, and establish a **Homomorphic Compression Theory** (on text data currently). → ***Conception***

2. We present **Homomorphic Compression System (HOCO)**, a text data management system that provides isolation between upper-level applications and underlying compression algorithms. → ***System***

3. We implement in HOCO with unified and succinct **optimization techniques** to achieve high performance in tasks involving text random access, modification, and analysis. → ***Algorithm***

Homomorphic Compression Theory

- ***Homomorphism in algebra***

  Homomorphism is a mapping φ between two algebraic structures $(G, *)$ and $(H, \Diamond)$ of the same type, such that

$$\varphi(x*y) = \varphi(x)\Diamond\varphi(y),$$

  where $x,y \in G$ and $\varphi(x), \varphi(y) \in H$.

💡 **Similar concept!**

(De)compression functions are homomorphism.

- ***Operating on compressed data***

  The results of operating on compressed data = Operating on uncompressed data first, and then compress the data.

Homomorphic Compression Theory

Formalization 1: Homomorphic Compression

- Text algebraic structure

  Uncompressed: $(\mathbb{U}, \Pi)$, $\mathbb{U}$ refers to uncompressed text set.

  Compressed: $(\mathbb{C}, \Theta)$, $\mathbb{C}$ refers to compressed text set.

- Basic operations

  $\Pi = \{\mathcal{E}, \mathcal{I}, \mathcal{D}, \mathcal{S}\}$ and $\Theta = \{\mathfrak{E}, \mathfrak{I}, \mathfrak{D}, \mathfrak{S}\}$ represent four text processing operations, namely *extract*, *insert*, *delete*, and *symbol_compare*.

- Homomorphic compression (HC)

  A compression algorithm that satisfies *operating on compressed data* paradigm can be expressed as a mapping $\varphi : \mathbb{U} \rightarrow \mathbb{C}$, such that for any operation $\sigma_u \in \Pi$ with n operands (n = 1, 2) and its corresponding operation $\sigma_c \in \Theta$, we have

  $$\varphi\big(\sigma_u(u_1, \cdots, u_n)\big) = \sigma_c\big(\varphi(u_1), \cdots, \varphi(u_n)\big),$$

  where $u_1, \cdots, u_n \in \mathbb{U}$. We call φ ***homomorphic compression (HC)***.

Homomorphic Compression Theory

Formalization 2: Homomorphic Compression Scheme

Definition 1 (P-Evaluation Scheme). Let P be a non-empty set of permitted operations. A P-evaluation scheme is characterized by three polynomial-time algorithms $(Comp, Decomp, Eval)$:

- $c \leftarrow Comp(u)$. The compression algorithm. It takes the uncompressed text u as input and outputs the compressed text c.
- $u \leftarrow Decomp(c)$. The decompression algorithm. It takes the compressed text c as input and outputs the uncompressed text u.
- $(c', r_c) \leftarrow Eval(\sigma_u, c)$. The evaluation algorithm. Given a permitted operation $\sigma_u \in$ P defined in the uncompressed text set, it finds the corresponding operation σ_c defined in the compressed text set, and outputs the updated compressed text c' by executing σ_c on the given compressed text c, as well as possible query result r_c for querying operations like *extract*.

Correct decompression + Correct evaluation

→ Correct P-Evaluation scheme == Homomorphic compression scheme.

Homomorphic Compression Theory

Formalization 3: Properties of HC Schemes

- **Property 1 (Directness).** → **Avoid decompression in *Eval*.**
  - Let HCS be a HC scheme, and P is the permitted operation set that HCS can evaluate. HCS is direct if for all $P \in$ P, $c \in$ C, $Eval(P,c)$ does not involve any processing fragments in the form of P (u_s), where u_s can be any uncompressed text segment in the processing procedure.
- **Property 2 (Strong homomorphism).** → **Avoid recompression.**
  - A scheme HCS is strongly homomorphic on operation family P if it is direct and for all $P \in$ P and every $u \in$ U, the following two distributions are statistically close up to a negligible distance:

$$Fresh(P,u) \stackrel{def}{=} \left\{ (c,c') : \begin{array}{l} c \leftarrow Comp(u), (u', r_u) \leftarrow P(u) \\ c' \leftarrow Comp(u') \end{array} \right\},$$

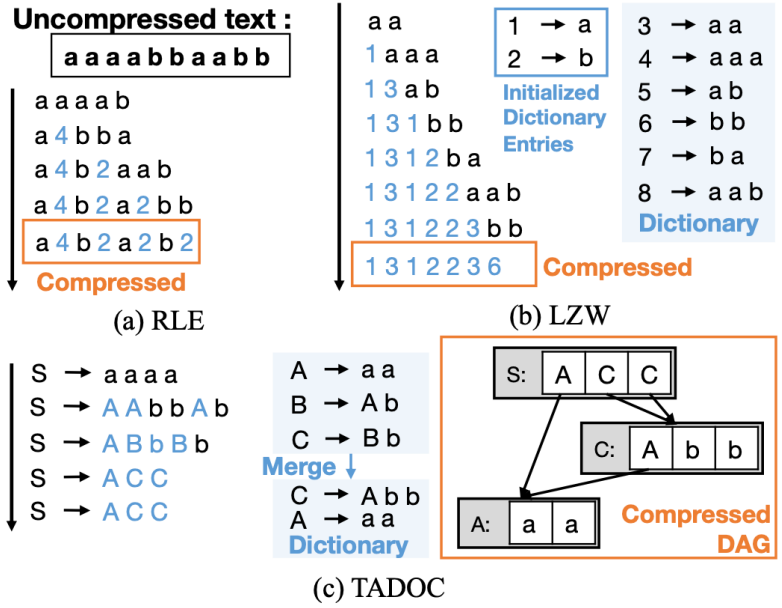
$$Evaluated(P,u) \stackrel{def}{=} \left\{ (c,c') : c \leftarrow Comp(u), (c', r_c) \leftarrow Eval(P,c) \right\}.$$

Homomorphic Compression Theory

Converting Compression Algorithms to HC Schemes

- Define (de)compression algorithms, and four basic operations.

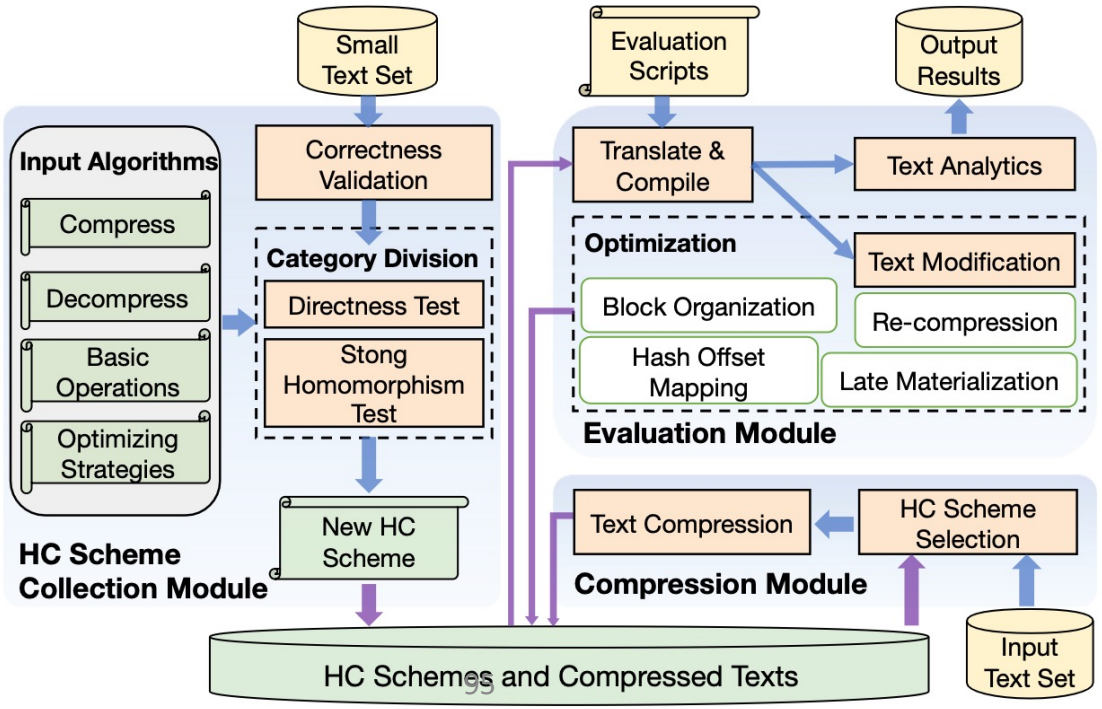
- Currently support three HC Schemes based on algorithms:
  - Run-length Encoding (RLE)
  - LZW
  - TADOC



- Suitable for different scenarios:
  - The RLE scheme satisfies directness and strong homomorphism, thus can support unlimited numbers of arbitrary operations.
  - LZW and TADOC schemes satisfy directness but are more efficient than RLE.

Homomorphic Compression System (HOCO)

HOCO is developed based on the HC theory, containing three key modules: HC scheme collection, compression, and evaluation.



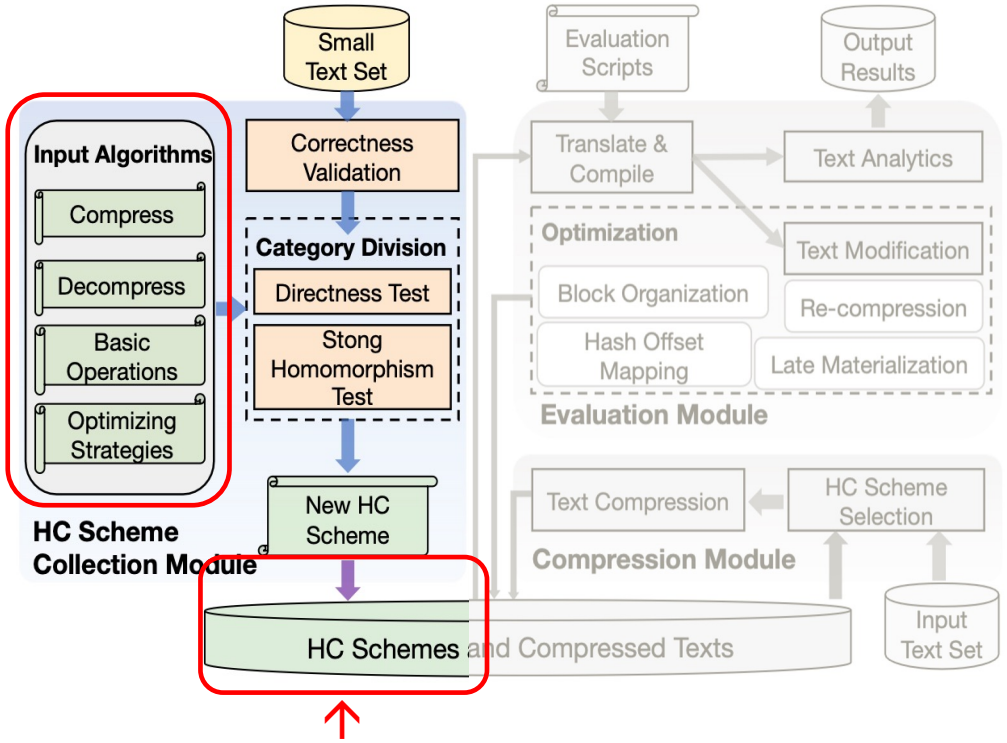
Homomorphic Compression System (HOCO)

1. HC Scheme Collection Module

- *Usage*: streamline the processes of *scheme registration*, *validation*, and *accessibility by other modules*.
- *Input*: user-defined HC scheme.

- HOCO structures each HC scheme as an object-oriented class.
- All schemes (including user input) are derived from a default base class, *BaseHC*.

```
1  template <typename Path, typename Offset,
2            typename Symbol, typename Rst>
3  class BaseHC {
4      public:
5          BaseHC(short _ctg) { HC_category = _ctg; }
6          virtual bool Compress(Path input_file,
7                                Path output_file) = 0;
8          virtual bool Decompress(Path output_file) = 0;
9          virtual bool Extract(Offset offset_start,
10                              Offset length, Rst* result);
11         virtual bool Insert(Offset offset_start,
12                             Symbol* str);
13         virtual bool Delete(Offset offset_start,
14                             Offset length);
15         virtual bool Symbol_cmp(Symbol s1, Symbol s2);
16     protected:
17         short HC_category;
18 };
```



- HOCO provides a repository for external access.

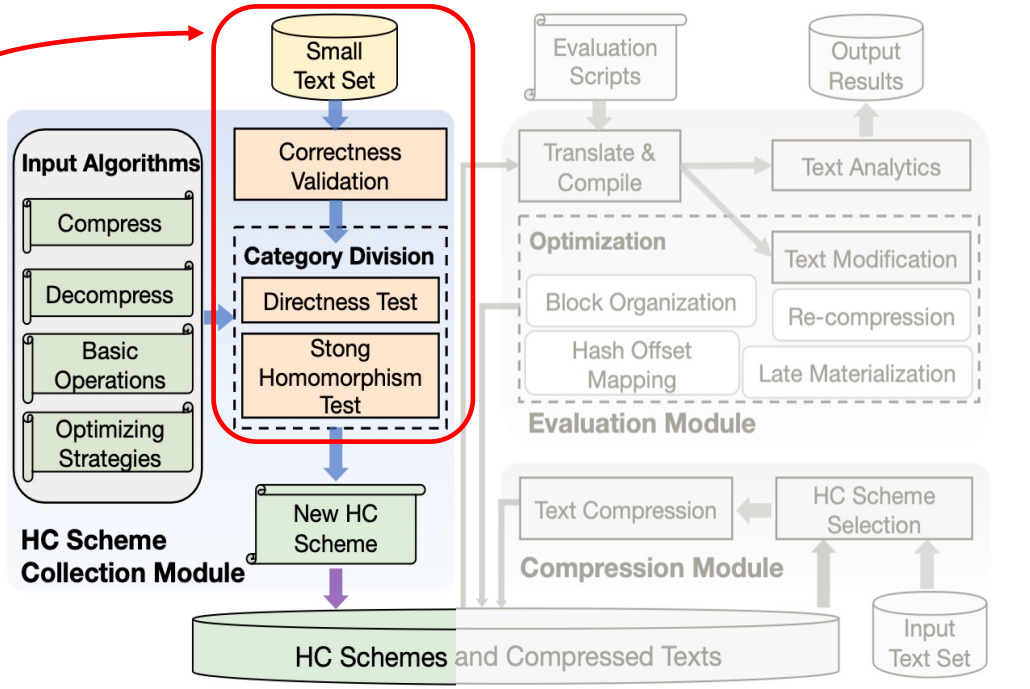
Homomorphic Compression System (HOCO)

1. HC Scheme Collection Module

- *Usage*: streamline the processes of *scheme registration*, *validation*, and *accessibility by other modules*.
- *Input*: user-defined HC scheme.
- *Property examination*: correctness → directness → strong homomorphism.

- *Correctness*: check if decompress successfully recovers the original text.
- *Directness*: check if homomorphic operations take shorter time than decompression + computation.
- *Strong homomorphism*: check if homomorphic operations perfectly retain the compression formats.

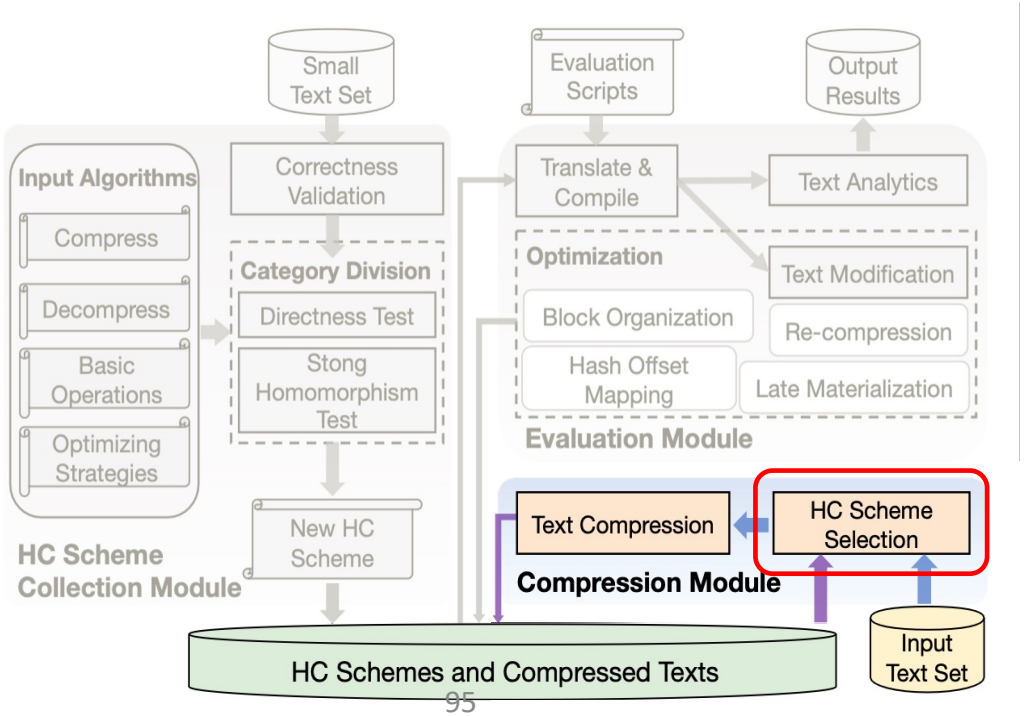
- *Classify* the newly added HC schemes based on the properties they satisfy for subsequent use.



Homomorphic Compression System (HOCC)

2. Compression Module

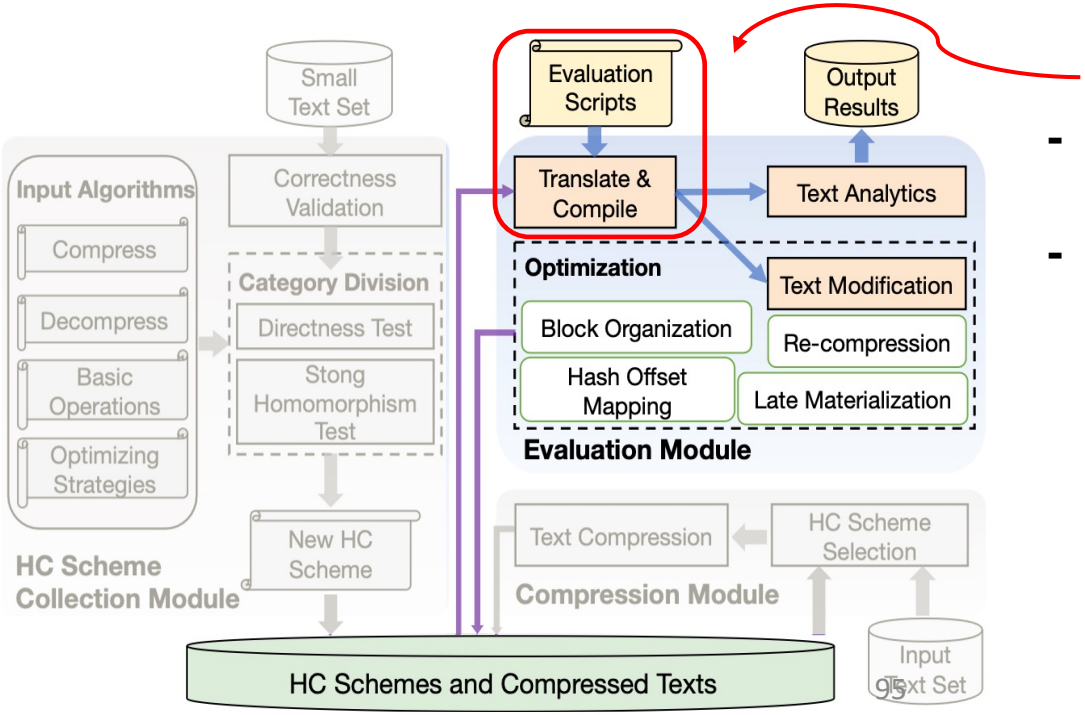
- *Usage*: select appropriate HC scheme, and use it to compress user-provided data.
- *Input*: uncompressed text.
- *Scheme selection*: use the scheme specified by user, or automatically determine a scheme based on its properties, input texts' attributes, and user requirements with a standard MLP.



Homomorphic Compression System (HOCO)

3. Evaluation Module

- *Usage*: receive text processing requests from users and translate them into corresponding efficient operations on the compressed text.
- *Input*: evaluation scripts that follow the coding logic on uncompressed data.
- *Homomorphic evaluation*: operates at runtime, invoked upon receipt of text processing requests from users, activating an *online instance*. Each instance involves three core stages: *data loading*, *code translation and compilation*, and *compressed text processing*.

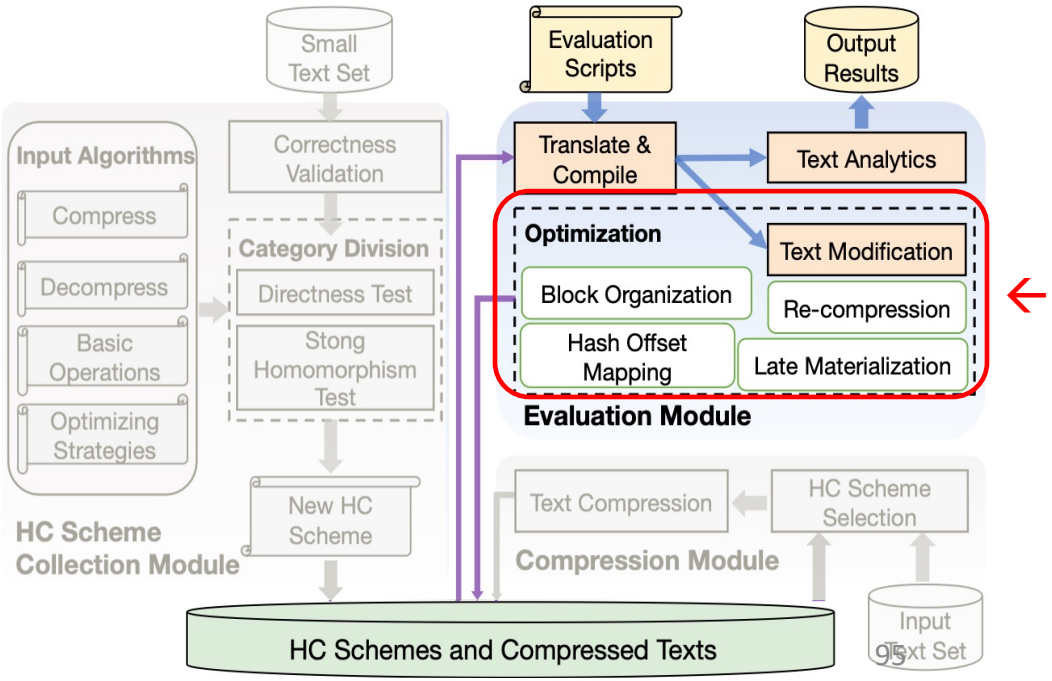


- HOCO provides *evaluation APIs* and *domain-specific language (DSL)*.
- *Parser & translator*: convert input scripts into executable C++ files, which relies on the Boost Spirit X3 C++ library.

Homomorphic Compression System (HOCO)

3. Evaluation Module

- *Usage*: receive text processing requests from users and translate them into corresponding efficient operations on the compressed text.
- *Input*: evaluation scripts that follow the coding logic on uncompressed data.
- *HOCO optimization*: uniformly abstract compressed texts of diverse formats, and construct unified and concise data structures to facilitate random access and modification operations on compressed data.



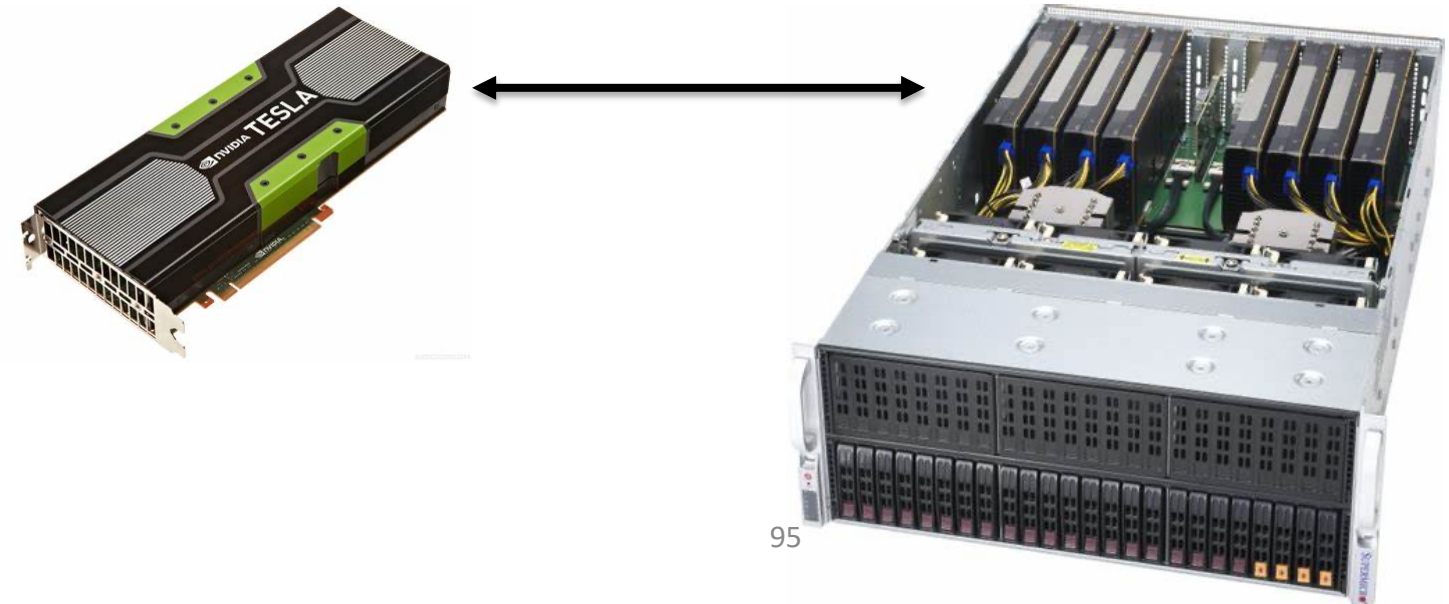
- *Text analytics*: reuse intermediate results on repetitive symbols (e.g., dictionary entries) to reduce computation.
- *Random access*: hash offset mapping + block organization.
- *Modification*: block reuse technique within block organization.

4. Hardware-accelerated compressed data direct processing

- **1. Motivation of GPUs**
- 2. Challenges
- 3. Our Solution
- 4. Application to Edge Devices

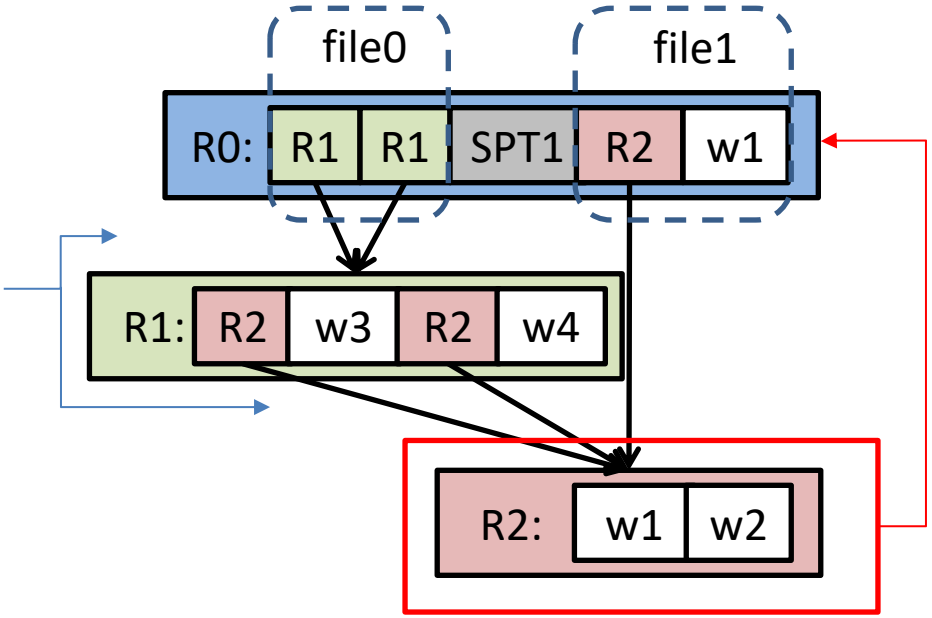
4.1 Motivation of GPUs

- GPU – popular in data science
- High throughput and computing power
- Different from CPUs
- Previous GPU-based methods does not apply



4.2 Challenges

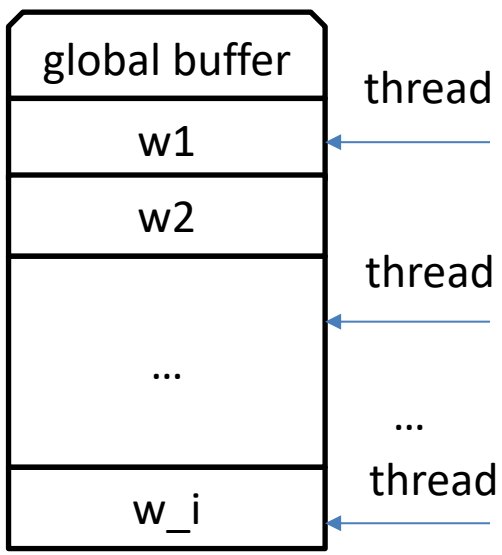
- Challenge 1: GPU parallelism for TADOC
  - Example: R2 depends on R0 and R1, and R1 depends on R0



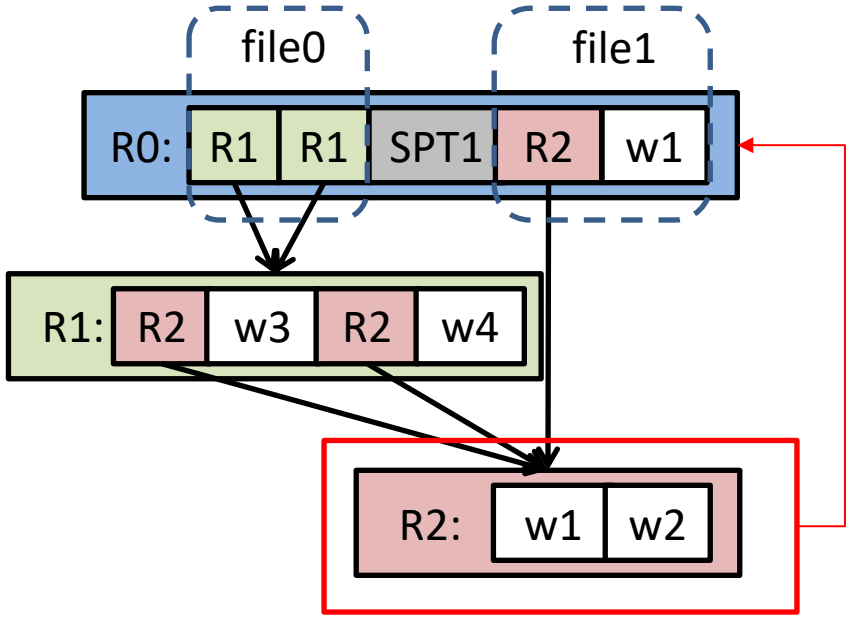
- dependencies
  - Edges

4.2 Challenges

- Challenge 2: TADOC final result update conflict of massive GPU threads
  - Example: writing to a global buffer

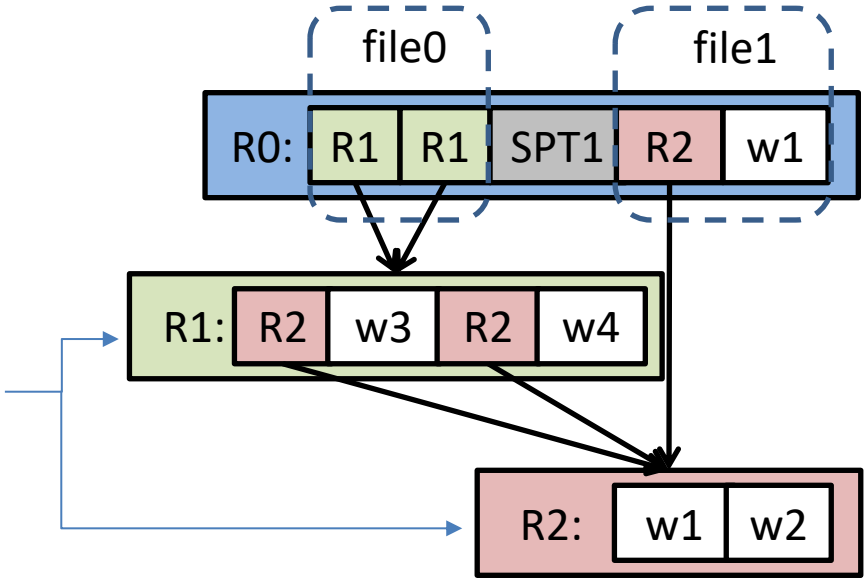


write conflicts



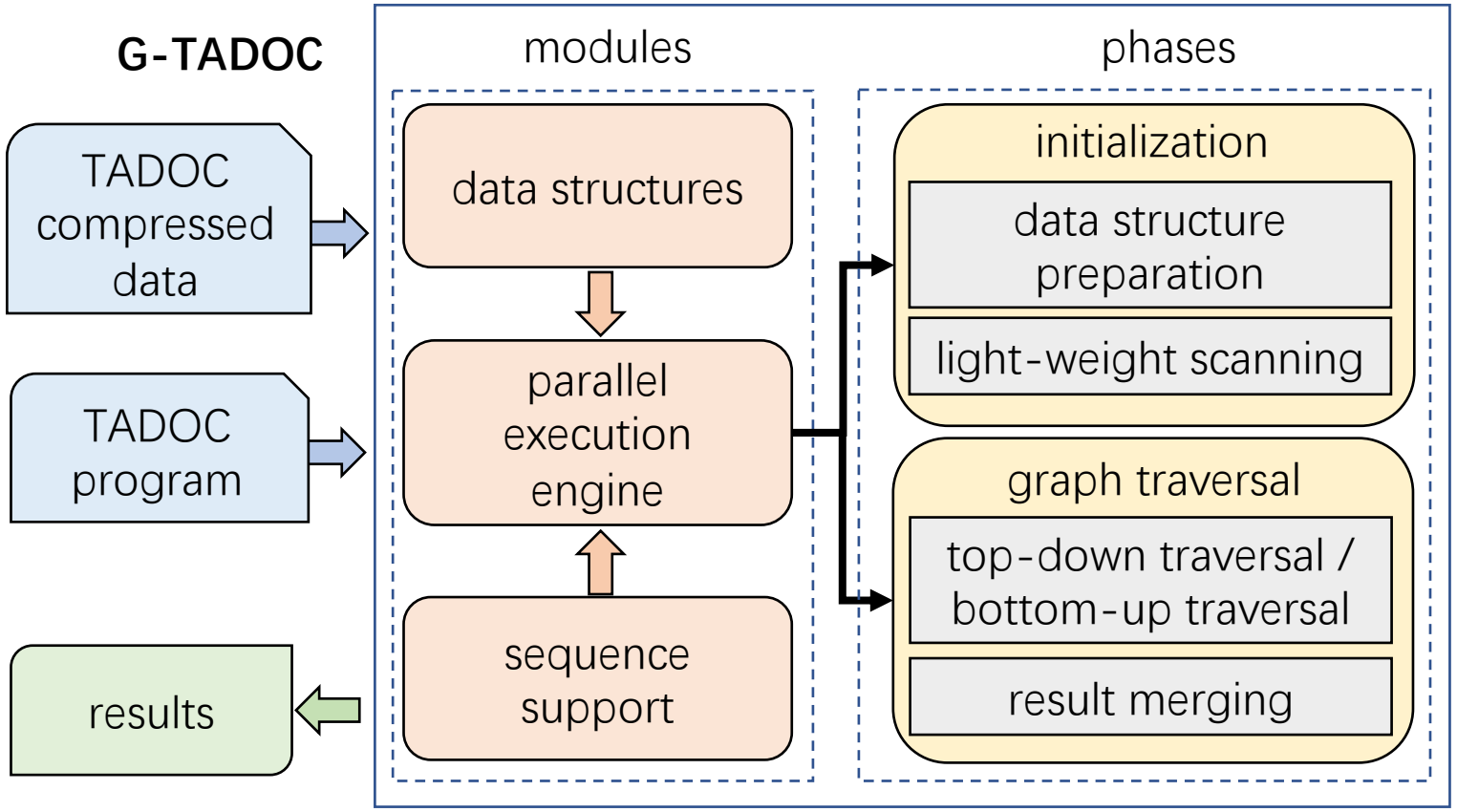
4.2 Challenges

- Challenge 3: sequence maintenance of TADOC compressed data on GPUs
  - Example: cross-rule sequence



- sequence:
  - w1-w2-w3
    - R1 and R2

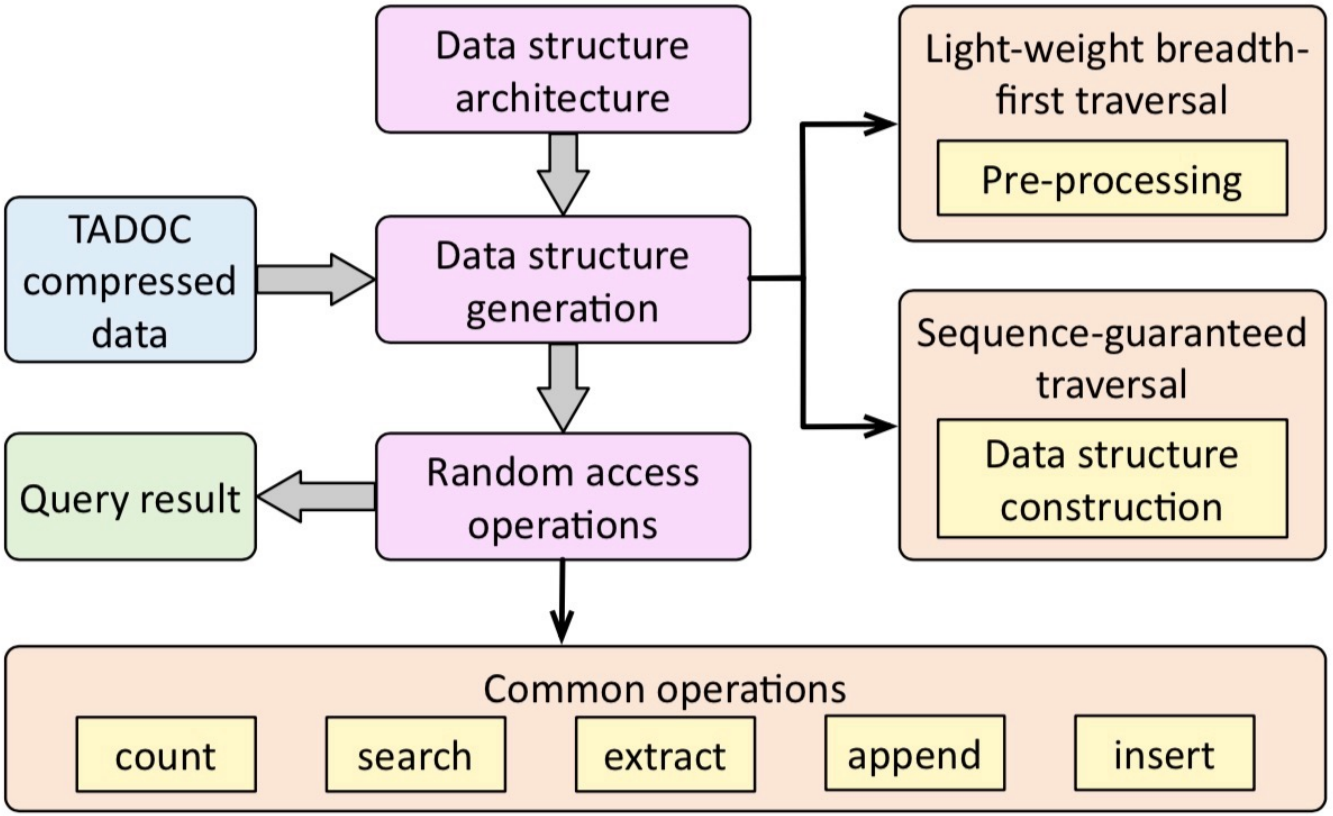
4.3 Our Solution



[**ICDE'21**] "G-TADOC: Enabling Efficient GPU-Based Text Analytics without Decompression" , Feng Zhang, Zaifeng Pan, Yanliang Zhou, Jidong Zhai, Xipeng Shen, Onur Mutlu, and Xiaoyong Du. IEEE ICDE, 2021.

4.3 Our Solution

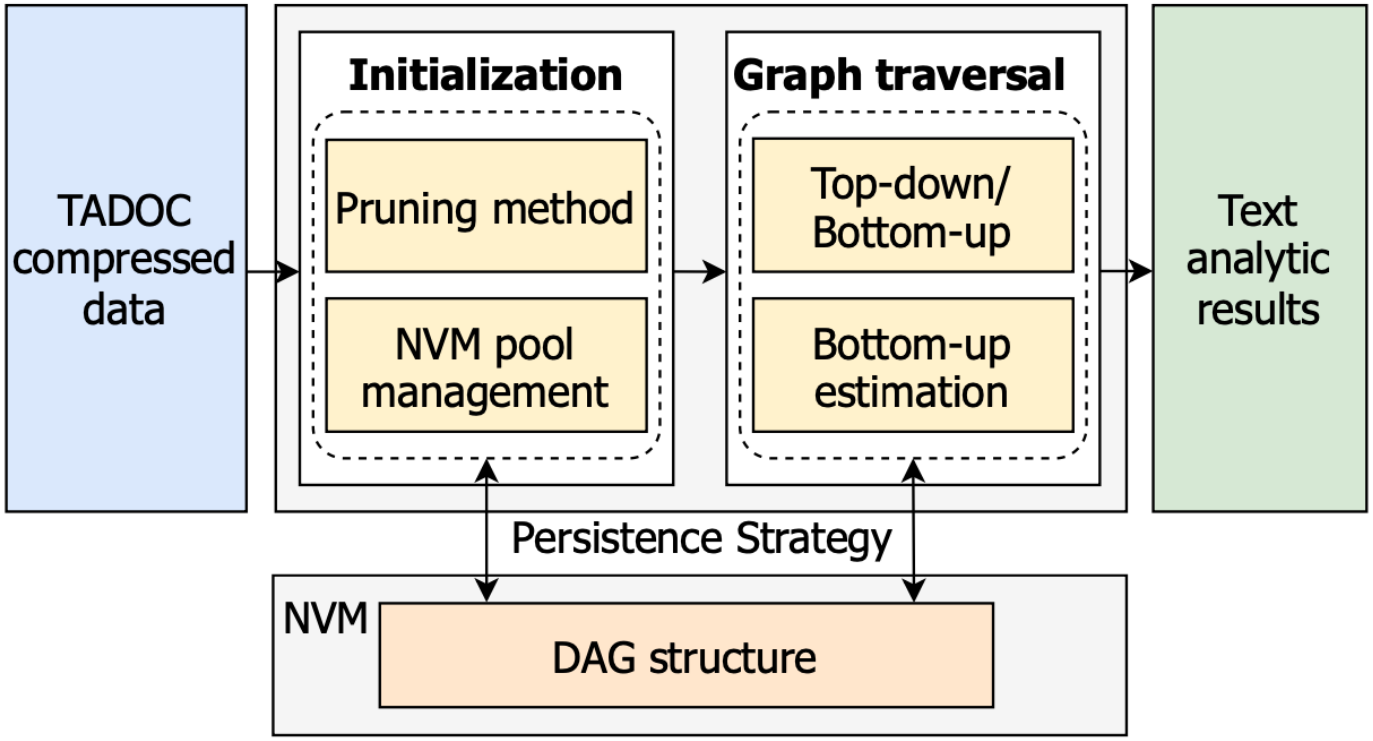
- Random Access



[SC'22] Optimizing Random Access to Hierarchically-Compressed Data on GPU, Feng Zhang, Yihua Hu, Haipeng Ding, Zhiming Yao, Zhewei Wei, Xiao Zhang, Xiaoyong Du. SC, 2022.

4.3 Our Solution

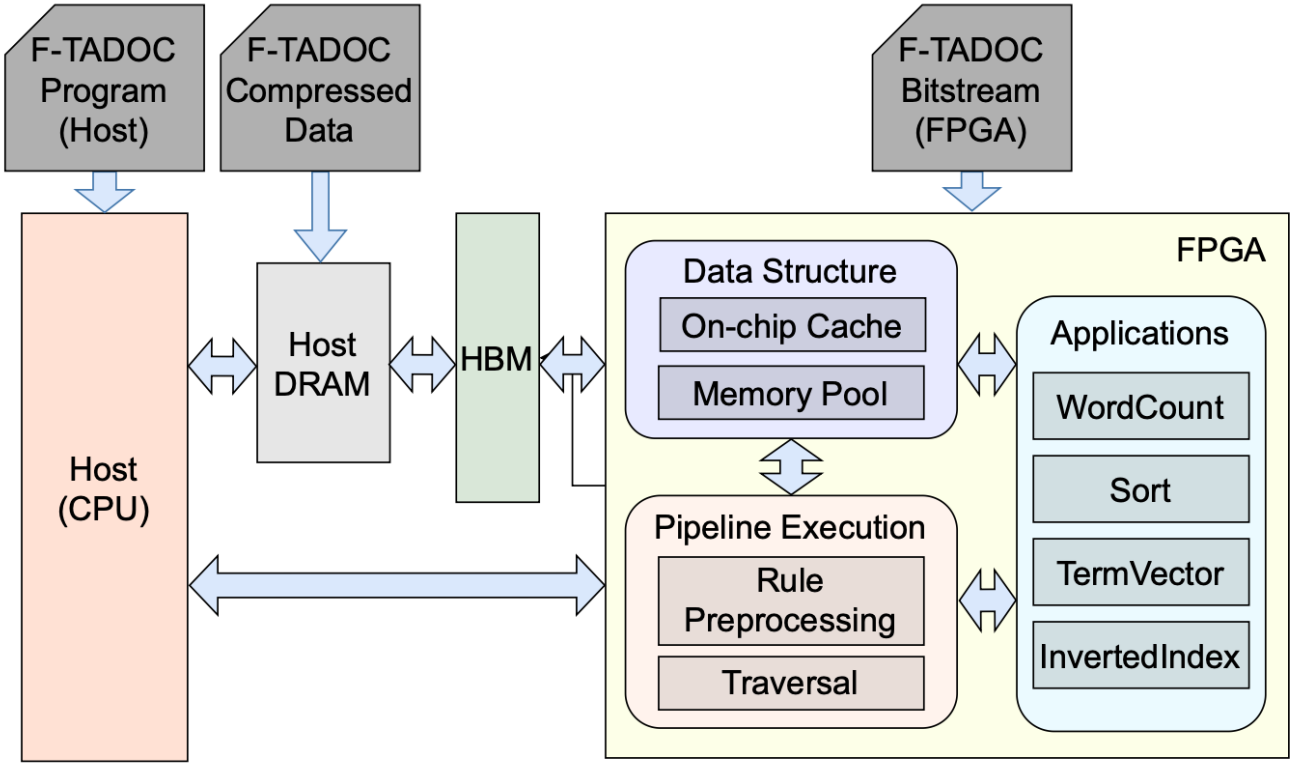
- NVM



[ICDE'24] "Enabling Efficient NVM-Based Text Analytics without Decompression", Xiaokun Fang, Feng Zhang, Junxiang Nong, Mingxing Zhang, Puyun Hu, Yunpeng Chai, Xiaoyong Du. ICDE 2024.

4.3 Our Solution

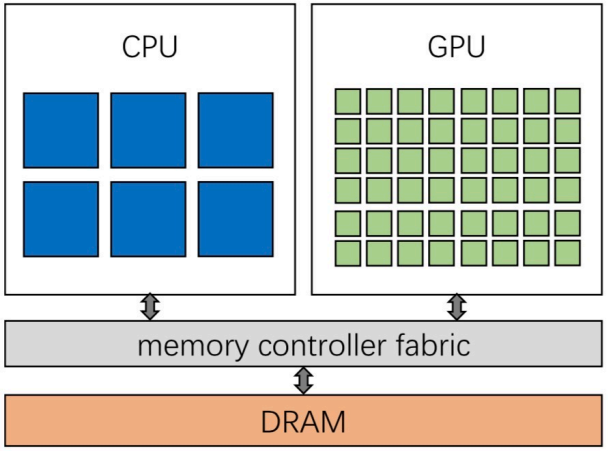
- FPGA

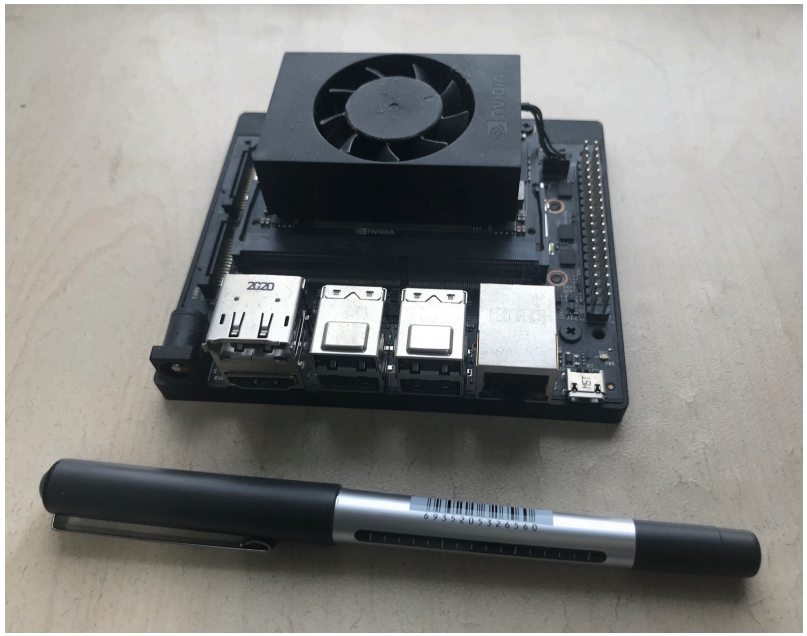


[ICDE'24] "F-TADOC: FPGA-Based Text Analytics Directly on Compression with HLS", Yanliang Zhou, Feng Zhang, Tuo Lin, Yuanjie Huang, Saiqin Long, Jidong Zhai, Xiaoyong Du. ICDE 2024.

4.4 Application to Edge Devices

- Edge computing is popular
- Low capacity
- Embedded GPU
- limited GPU memory





[**TPDS'21**] "Exploring Data Analytics without Decompression on Embedded GPU Systems", Zaifeng Pan, Feng Zhang, Yanliang Zhou, Jidong Zhai, Xipeng Shen, Onur Mutlu, Xiaoyong Du. TPDS, 2021.

Part 2: Application of homomorphic compression in database

- 1. Motivation
- 2. Orthogonal Processing on Compression
- 3. Idea
- 4. Overview
- 5. System Design
- 6. Implementation

1. Motivation – diverse systems

- Various databases

Traditional DBS

NOSQL DBS

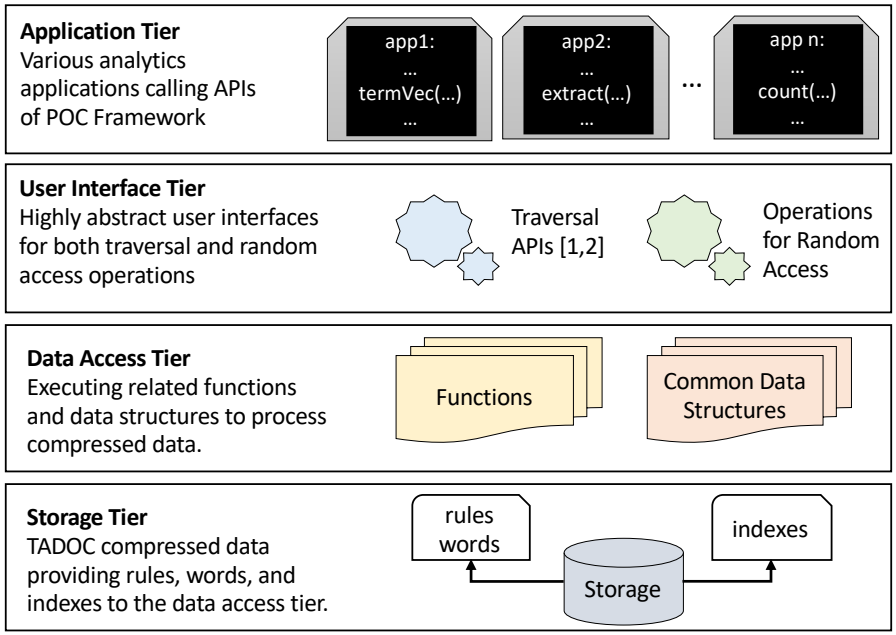
GRAPH DBS

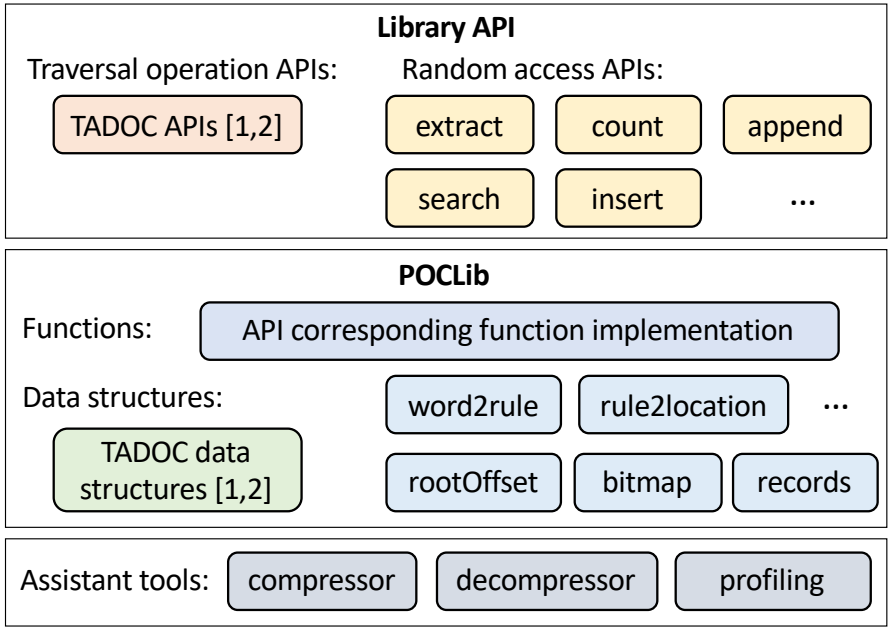
MPP DBS

CLOUD DBS

2. Orthogonal Processing on Compression

- Orthogonal POC
  - Analytics Orthogonality
  - POC Efficiency
  - Persistence Independence
- Near Orthogonal POC
  - Locality
  - Adaptability
  - Compatibility
  - Transparency

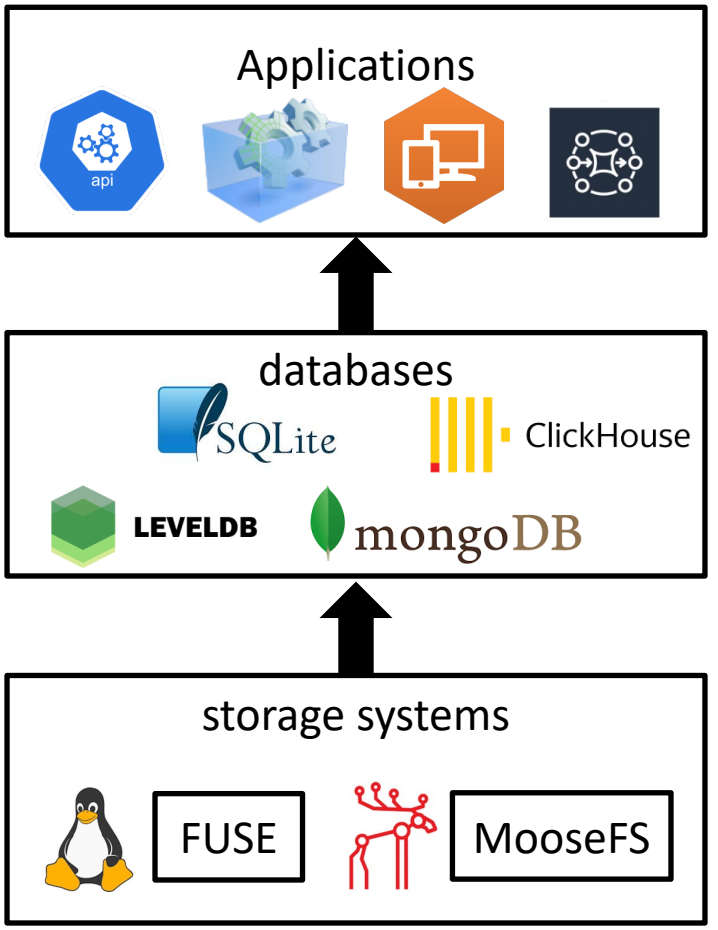




[**TPDS'21**] "POCLib: A High-Performance Framework for Enabling Near Orthogonal Processing on Compression", Feng Zhang, Jidong Zhai, Xipeng Shen, Onur Mutlu, and Xiaoyong Du. IEEE TPDS.

3. Idea

- Random update over compressed data in storage layer, supporting various databases

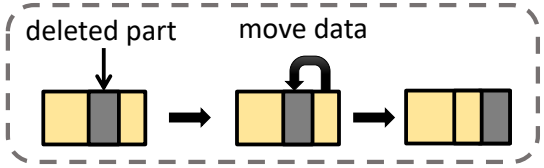


If we can integrate CompressDB to storage layer, then it can support diverse systems!

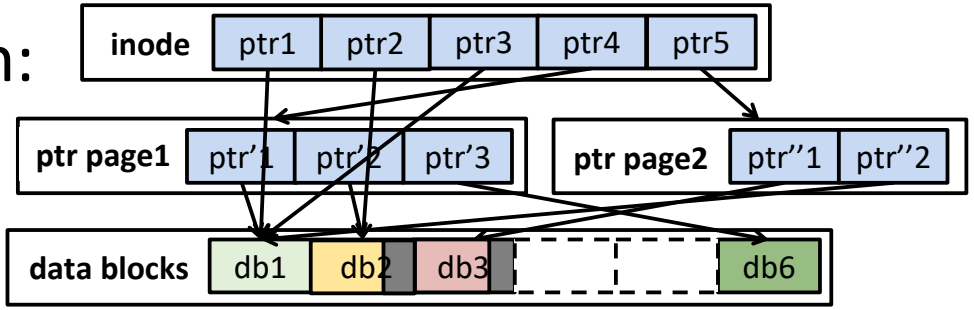
CompressDB

4. Overview

- Novelty in element level design:
  - Allowing data holes within rules.



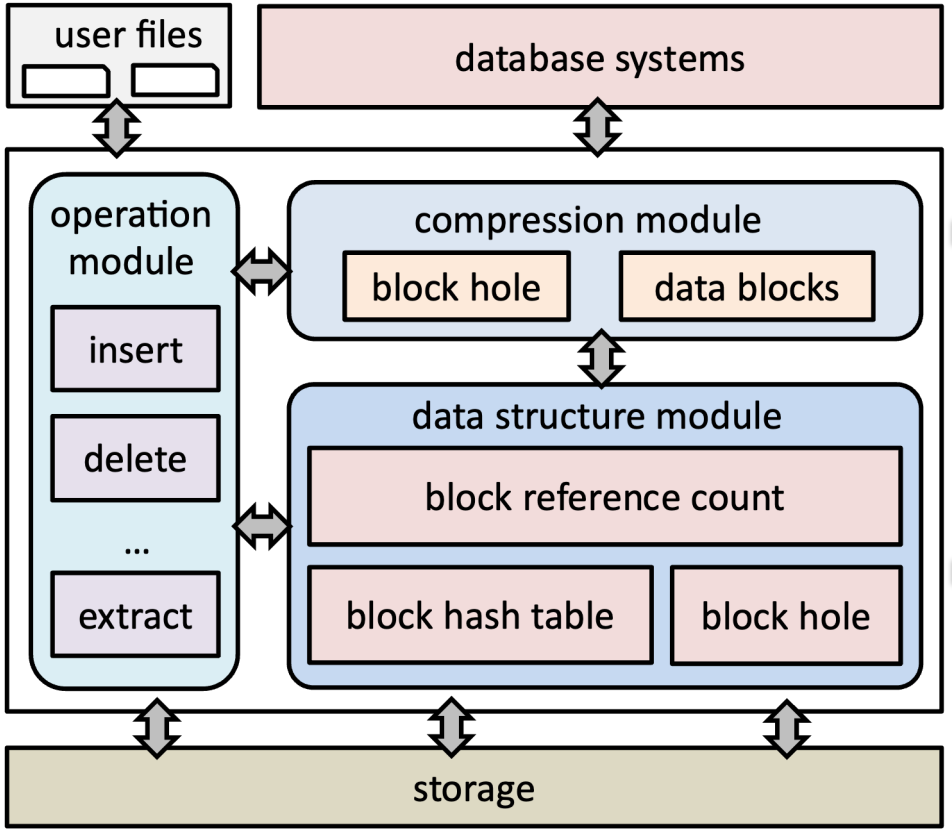
- Novelty in rule level design:
  - Efficiently locating and merging rules.

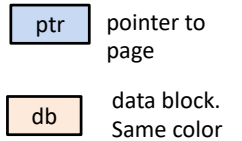


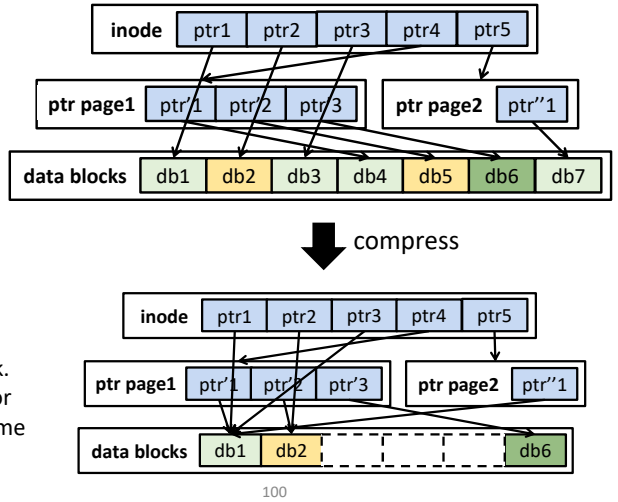
- Novelty in DAG level design:
  - Limiting the depth of rules for efficient random updates

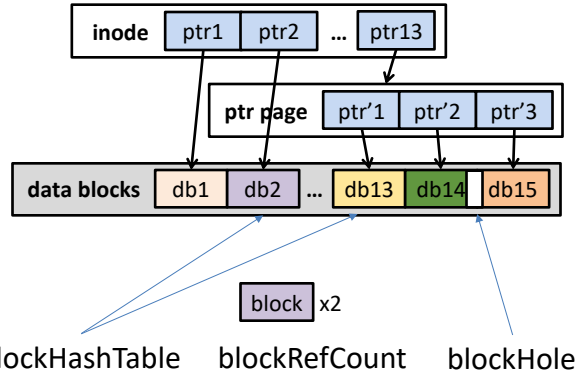
5. System Design

- CompressDB

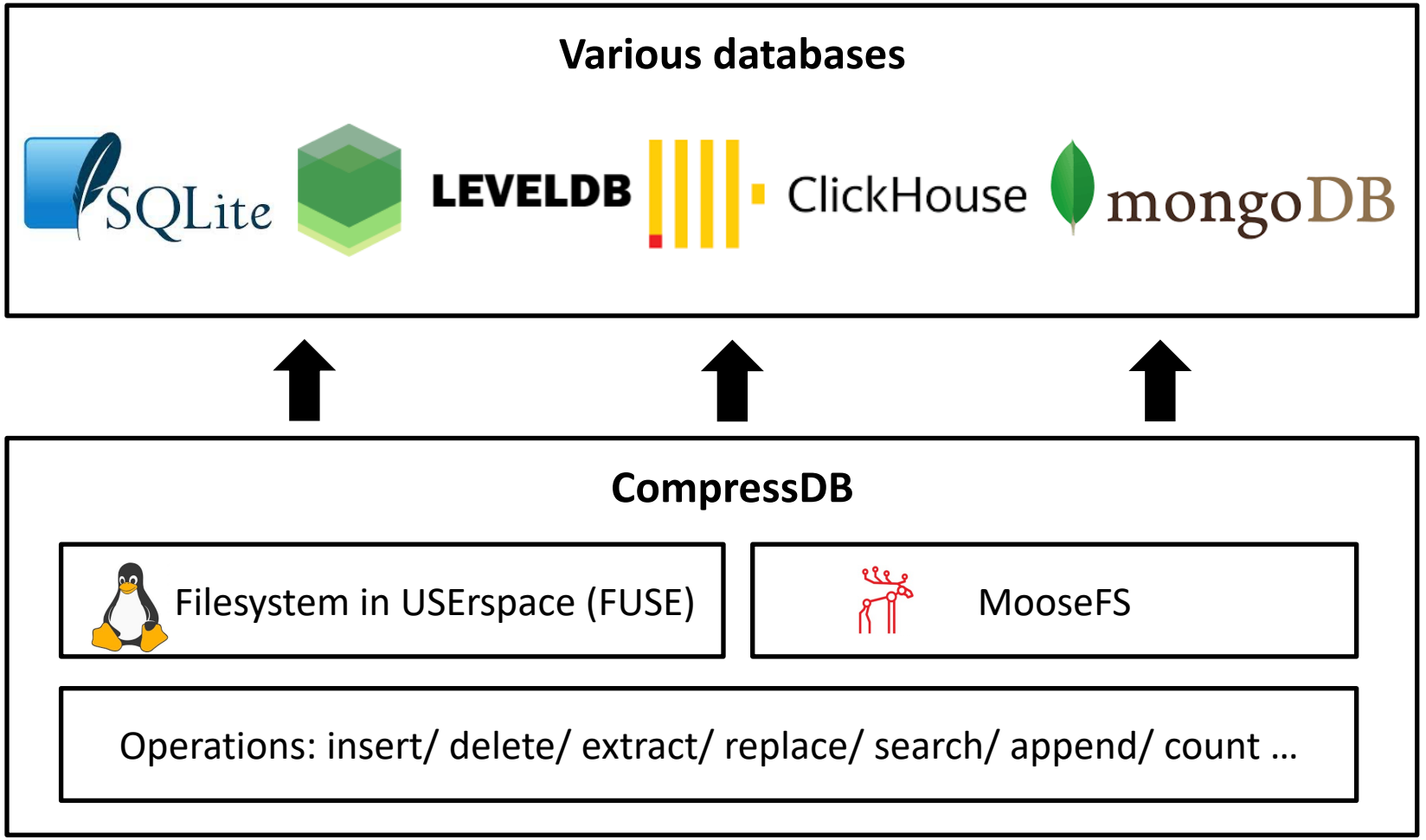








6. Implementation



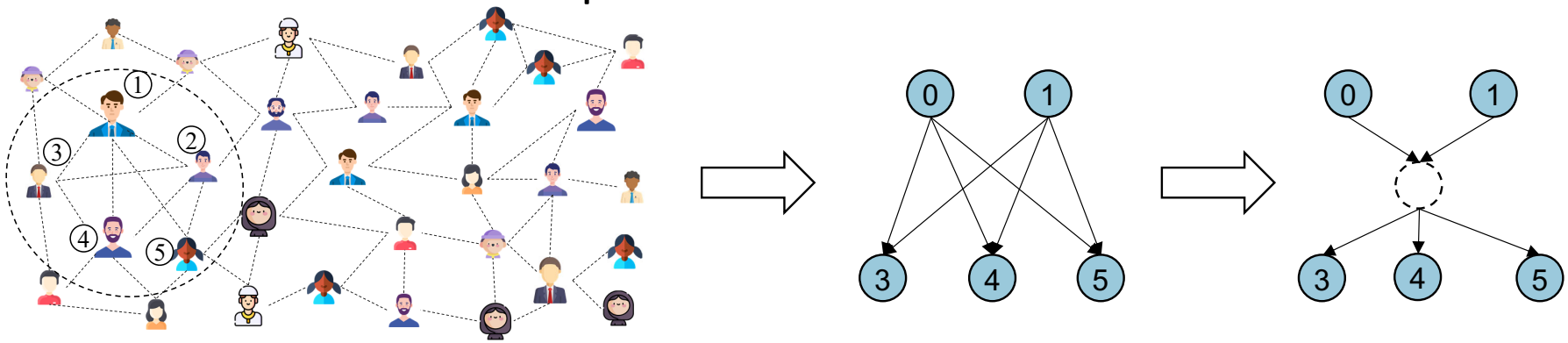
[SIGMOD'22] CompressDB: Enabling Efficient Compressed Data Direct Processing for Various Databases, Feng Zhang, Weitao Wan, Chenyang Zhang, Jidong Zhai, Yunpeng Chai, Haixiang Li, Xiaoyong Du, SIGMOD, 2022.

Part 3: Application of homomorphic compression database technology in graph data

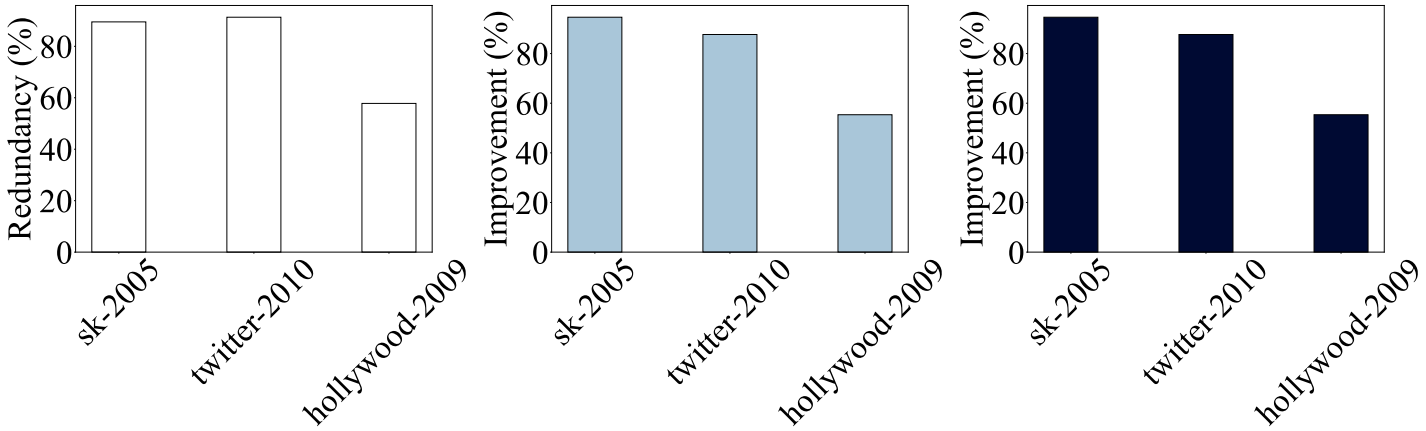
1. Motivation
2. Conceptual Framework
3. System Design

1. Motivation

- Rule-based compression



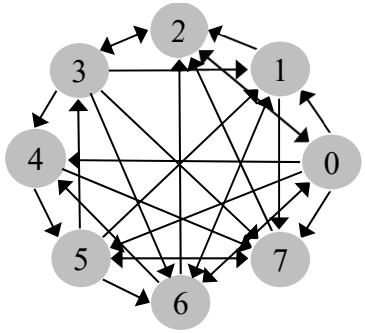
- Experiment analysis



(a) Redundancy occupancy. (b) CPU performance. (c) GPU performance.

2. Conceptual Framework

- Two-level traversal model

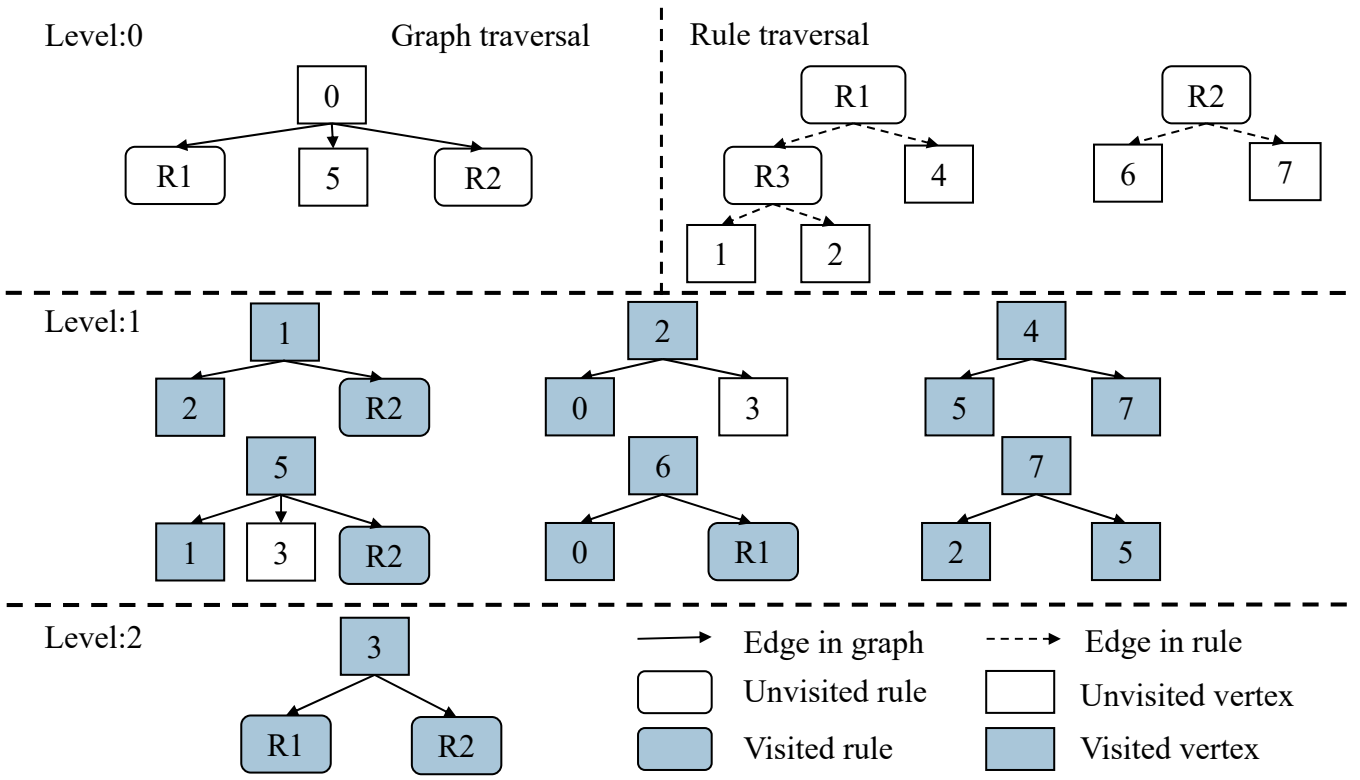


(a) Original graph.

Vertex	Neighbors
0	R1, 5, R2
1	2, R2
2	0, 3
3	R1, R2
4	5, 7
5	1, 3, R2
6	0, R1
7	2, 5

Rule	Content
R1	R3, 4
R2	6, 7
R3	1, 2

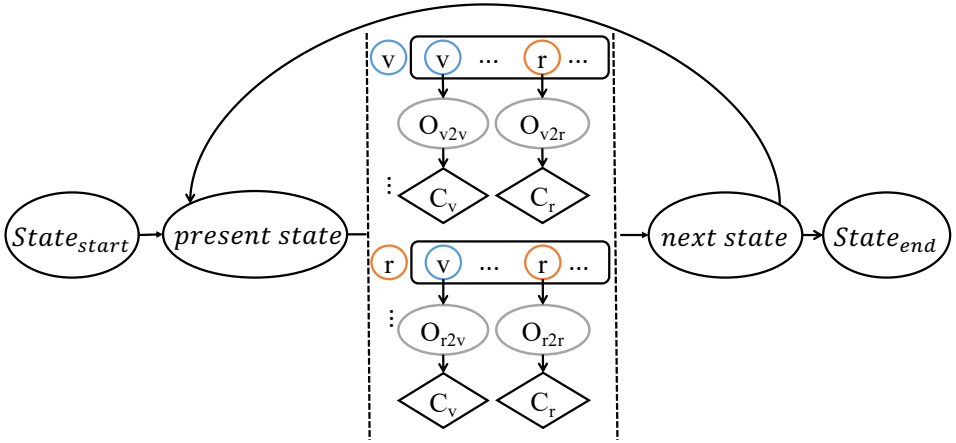
(b) Rule-based representation.



(c) BFS on CompresseGraph.

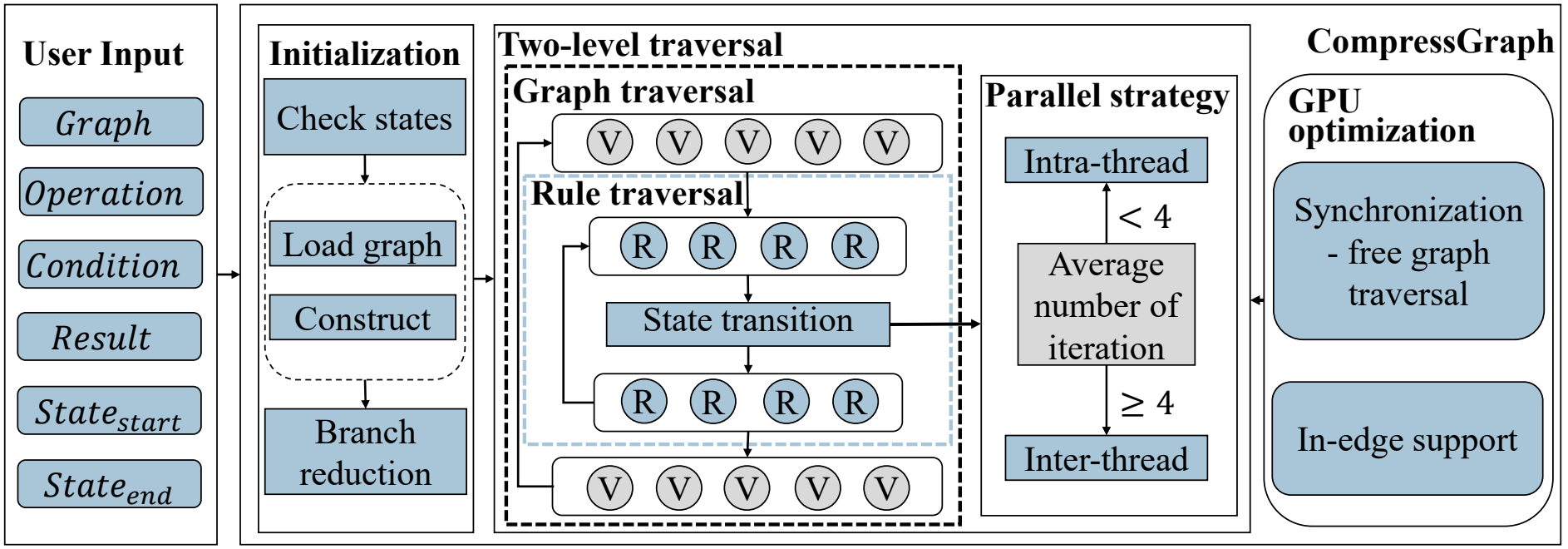
2. Conceptual Framework

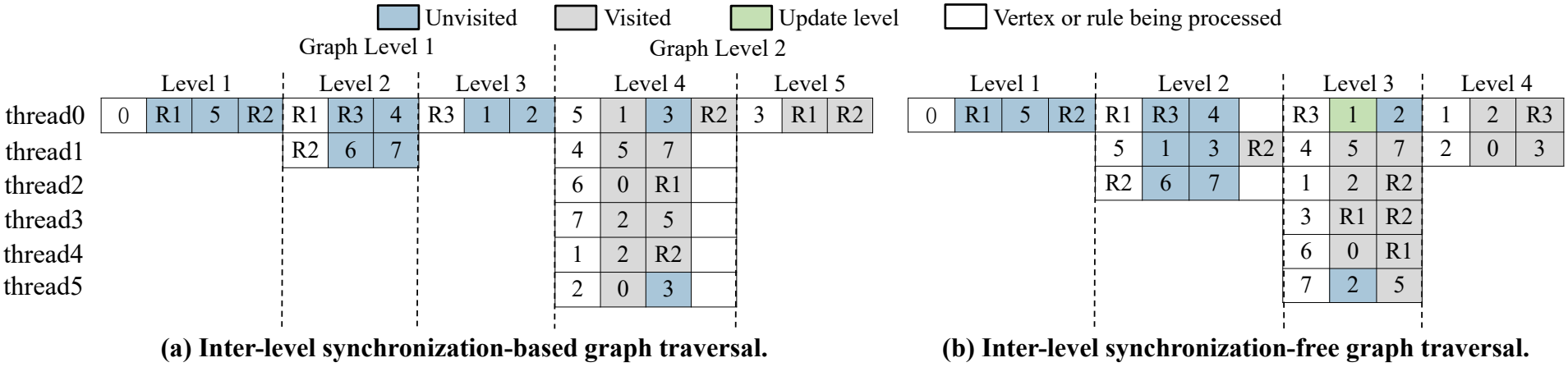
- Six-element tuple
  - Graph = <V, R, E>
  - Operation
  - Condition
  - Result
  - $\mathbf{State_{start}}$
  - $\mathbf{State_{end}}$




```
1   CompressGraph = {Graph; Operation; Condition; Result,
        State_start, State_end};
2   Graph = {V, R, E};
3   class Operation{
4       void Ov2v(vertex src, vertex dst){
5           if(dst.distance == INIT)
6               dst.distance = src.distance+1;
7       }
8       void Ov2r(vertex src, rule dst){
9           if(dst.distance == INIT)
10              dst.distance = src.distance+1;
11      }
12      void Or2v(rule src, vertex dst){
13          if(dst.distance == INIT)
14              dst.distance = src.distance;
15      }
16      void Or2r(rule src, rule dst){
17          if(dst.distance == INIT)
18              dst.distance = src.distance;
19      }
20  };
21  class Condition{
22      bool Cv(vertex V) { return V.distance == INIT; }
23      bool Cr(rule R) { return R.distance == INIT; }
24  };
25  class Result{
26      int distance;
27      Result(Graph G){
28          distance = INIT;
29      }
30  };
31  State_start = {V&R-{root}, {root}, null};
32  State_end = {U1, null, U2};
33  State_cur = State_start;
```

3. System Design





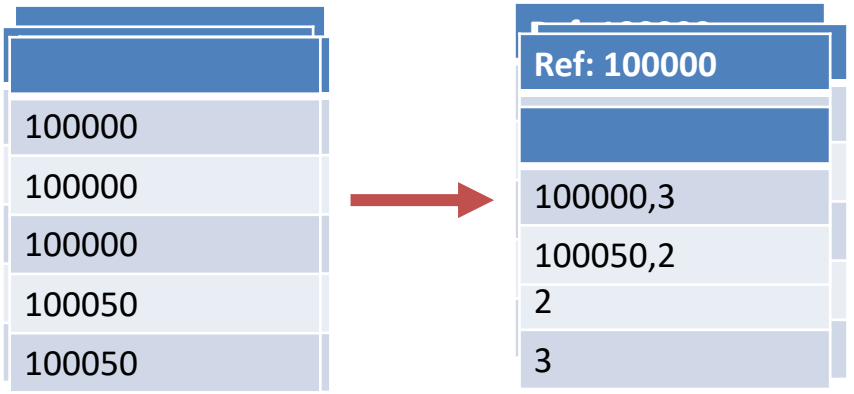
(a) Inter-level synchronization-based graph traversal. **(b) Inter-level synchronization-free graph traversal.**

Part 4: Application of compressed database technology in stream data

1. Background
2. Motivation
3. CompressStreamDB Framework
4. Selected Compression Algorithms

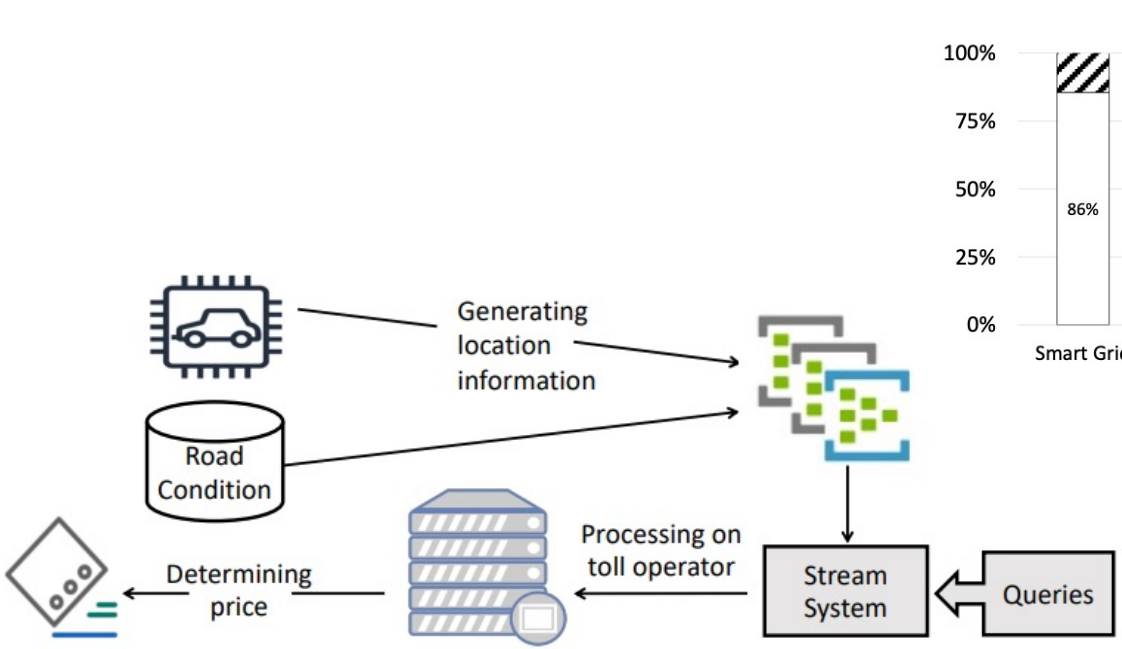
1. Background

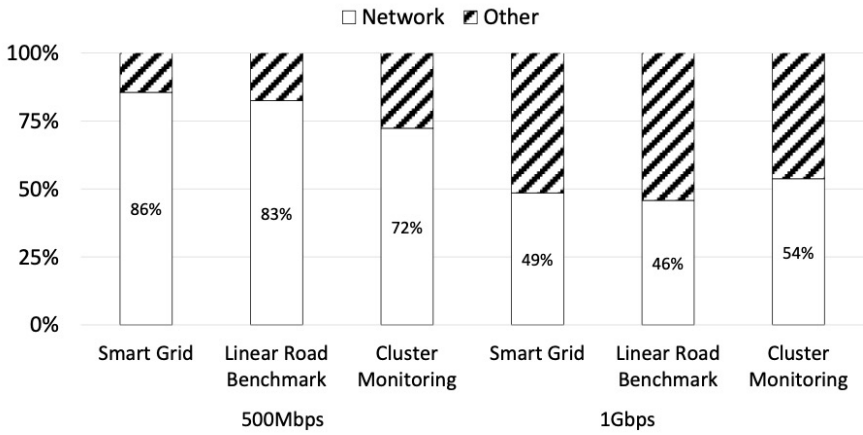
- **Light-weight compression method**
  - Frame-of-Reference（FOR）
  - Delta
  - Dictionary
  - Run Length Encoding(RLE)



2. Motivation

- The real-time requirement of stream processing

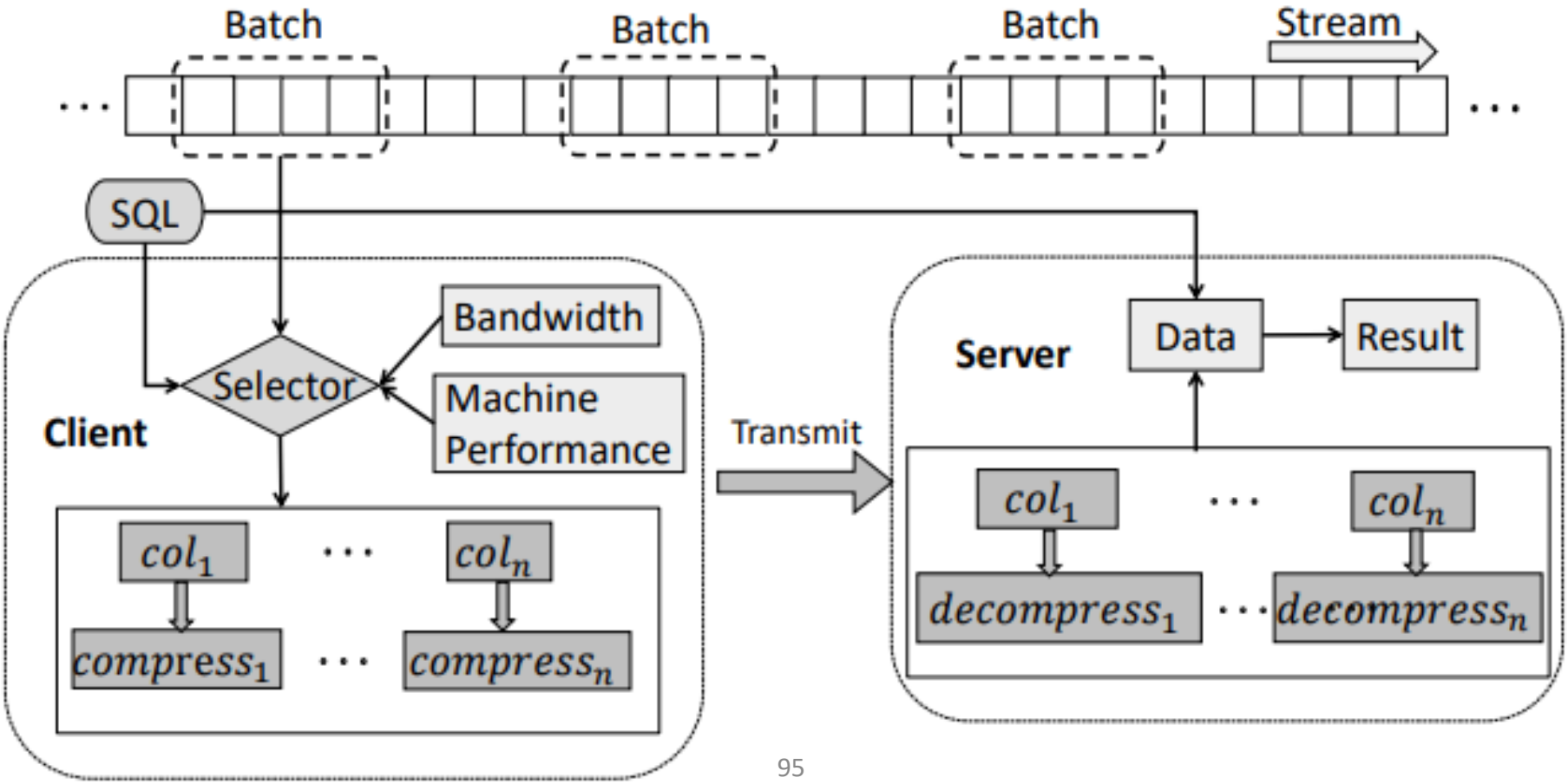




[2] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts, “Linear road: a stream data management benchmark,” in PVLDB, 2004

3. CompressStreamDB Framework

- Overview



4. Selected Compression Algorithms

TABLE I
EAGER AND LAZY COMPRESSION METHODS IN LIGHTWEIGHT COMPRESSION

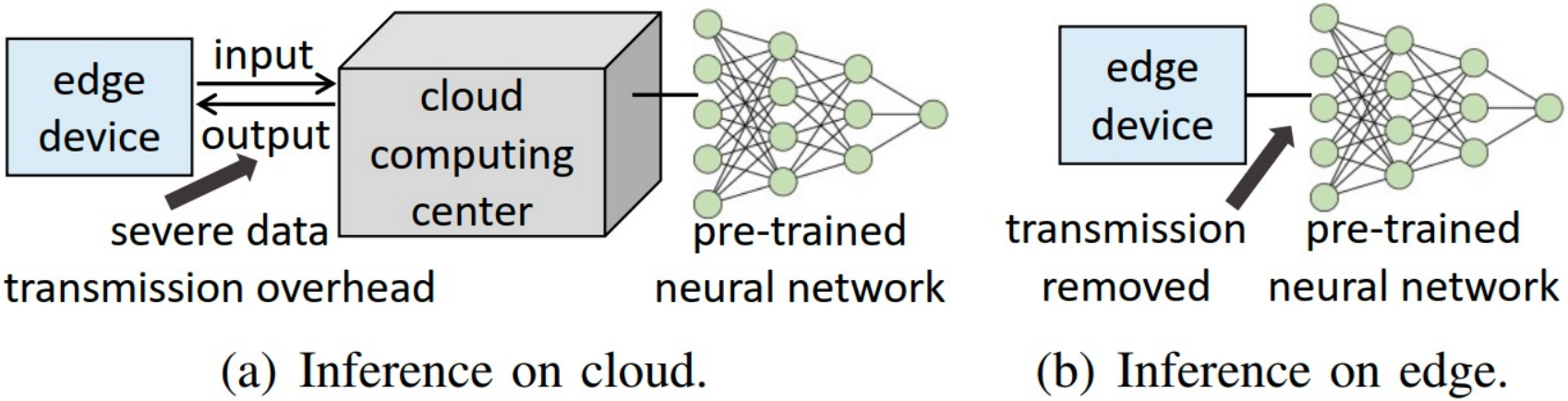
	Compression Method	Description
Eager	Elias Gamma Encoding [11]	Encode each value with unary and binary bits.
	Elias Delta Encoding [11]	Encode each value with unary and binary bits.
	Null Suppression with Fixed Length [36]	Delete leading zeros of each value with fixed bits.
	Null Suppression with Variable Length [36]	Delete leading zeros of each value with variable bits.
Lazy	Base-Delta Encoding [35]	Encode values as their delta values from base value.
	Run Length Encoding [37]	Encode values with their run lengths.
	Dictionary [38]	Maintain a dictionary of the distinct values.
	Bitmap [13, 41, 39, 40]	Encode each distinct value as a bit-string.

Part 5: Application of compressed database technology in ML data

1. Motivation
2. Background and Idea
3. Deep Reuse
4. Deep Reuse Winograd
5. TREC: Transient Redundancy Elimination-based Convolution

1 Motivation

- Data management on edge devices
- Widespread use of AIoT (artificial intelligence of things) in data management at the edge



(a) Inference on cloud. (b) Inference on edge.

[ICDE'23] "EdgeNN: Efficient Neural Network Inference for CPU–GPU Integrated Edge Devices", Chenyang Zhang, Feng Zhang, Kuangyu Chen, Mingjun Chen, Bingsheng He, Xiaoyong Du. ICDE, 2023.

1 Motivation

Are there any reusable computations in current DL models that can be exploited to further reduce computation overhead?

- Main concerns:
  - COMPUTATION: The inference process is more frequently done on low-power computing equipment.
  - SPEED: Faster DNN inference.

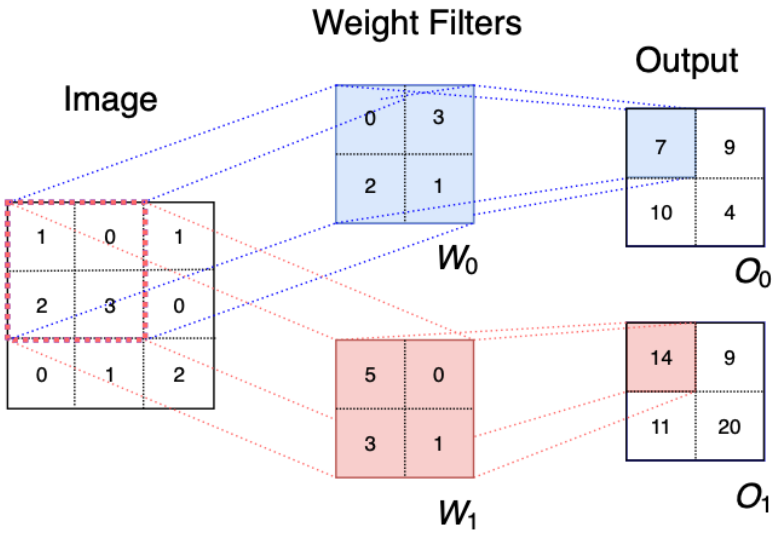
2 Background and Idea

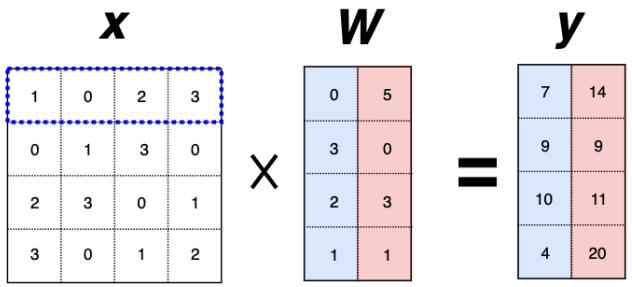
Observation

– Matrix multiplication tends to be the most computation-intensive operation.

Convolutional Neural Networks (CNNs):

– Im2col / Im2row based convolution

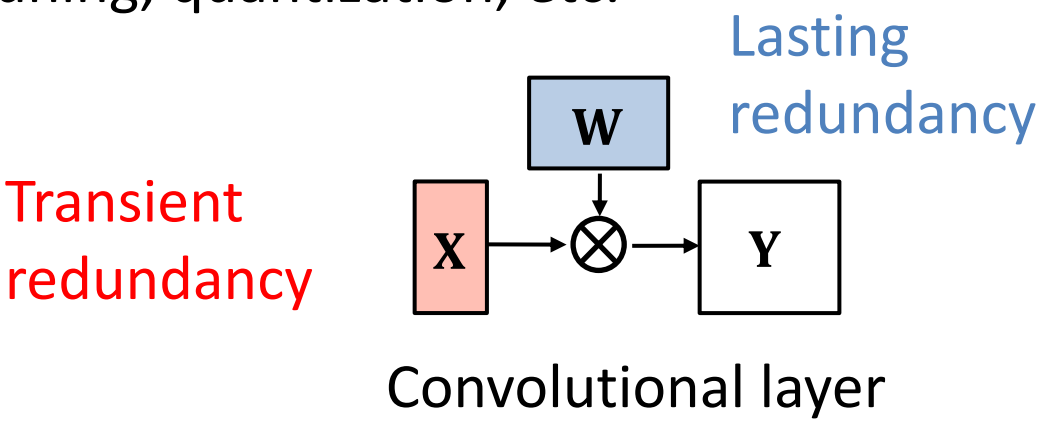




2 Background and Idea

Redundancy

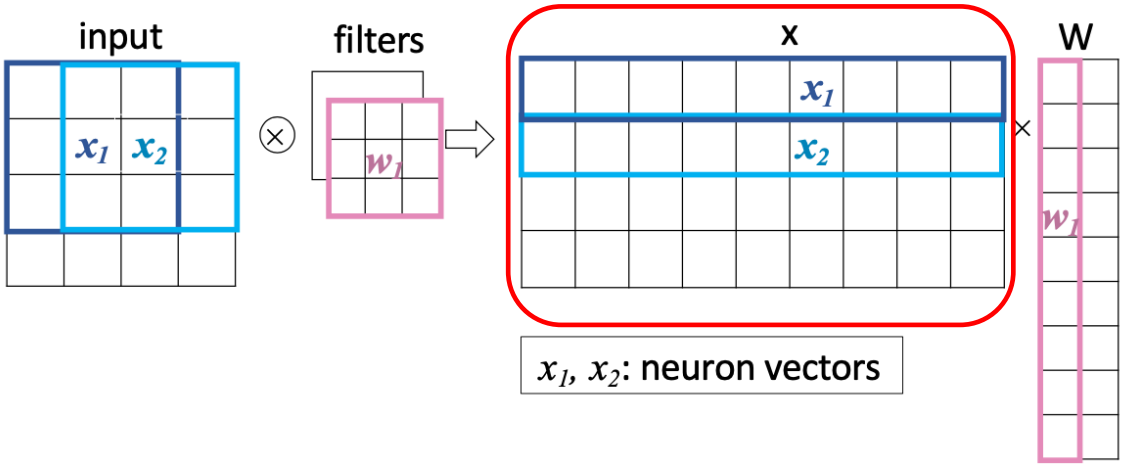
- **Lasting redundancy**
  - Arises from DNN parameters.
  - Removed during training mostly.
  - Pruning, quantization, etc.
- **Transient redundancy**
  - Arises from Input data / activation maps.
  - Removed during inference.



3 Deep Reuse

Key insight:

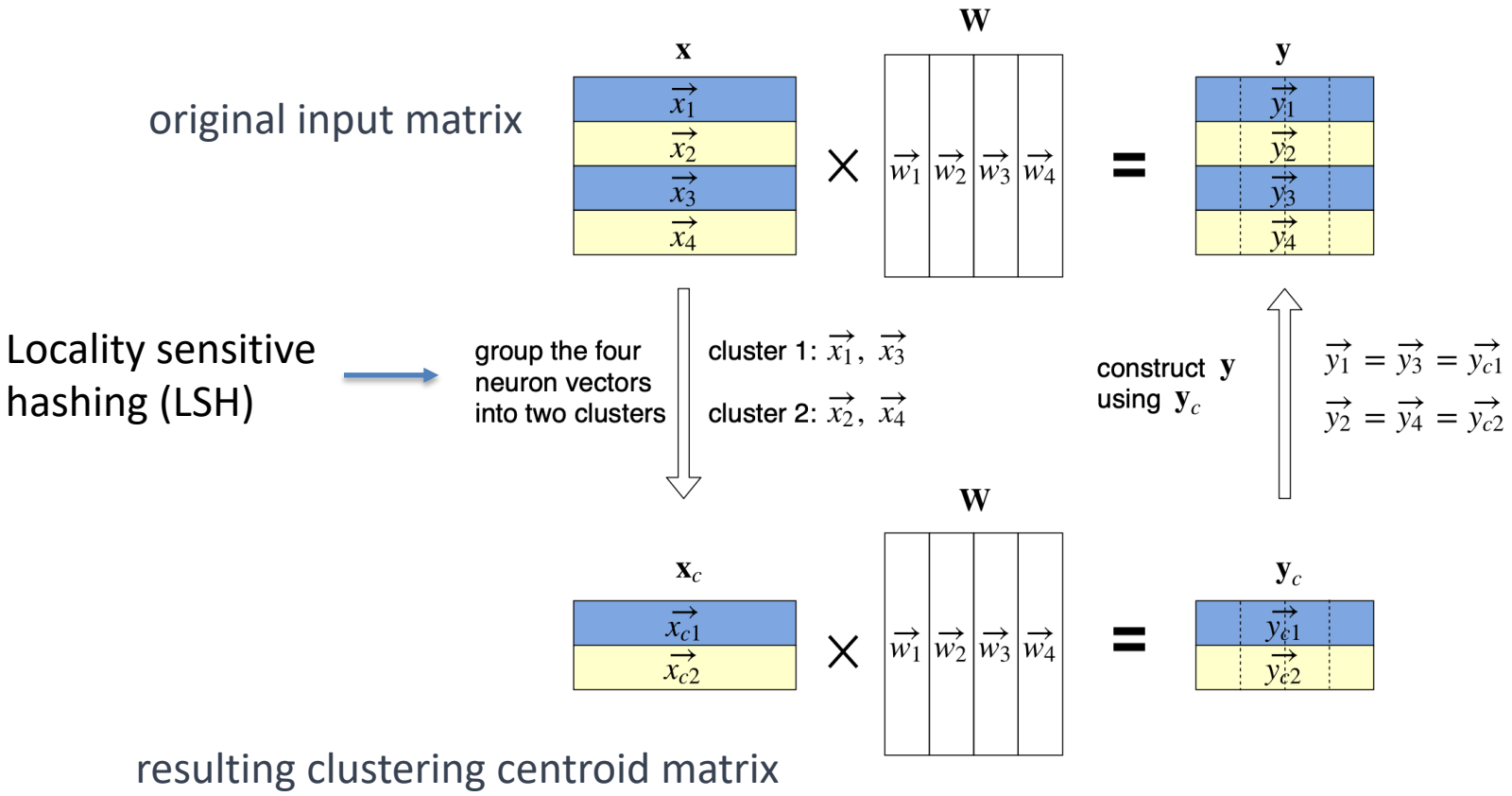
- Similarity exists among neuron vectors.
  - A *neuron vector* is made up of values carried by some consecutive neurons at a CNN layer.



[ICDE'2019] "Adaptive Deep Reuse: Accelerating CNN Training on the Fly", Lin Ning, Hui Guan, Xipeng Shen, International Conference on Data Engineering, Macau SAR, China, April, 2019.

3 Deep Reuse

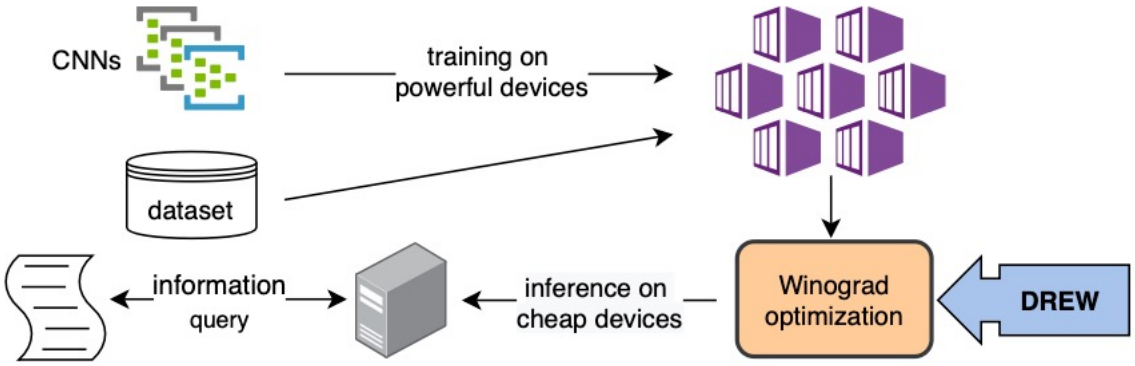
The basic idea of employing neuron vector similarity for deep reuse.



4 Deep Reuse Winograd

The key to improving the inference performance is to accelerate the convolutional layers.

– Fast convolution algorithms: Winograd, FFT, etc.



Is there an opportunity for data reuse in fast convolution algorithms?

4 Deep Reuse Winograd

Winograd convolution

1. Transformation

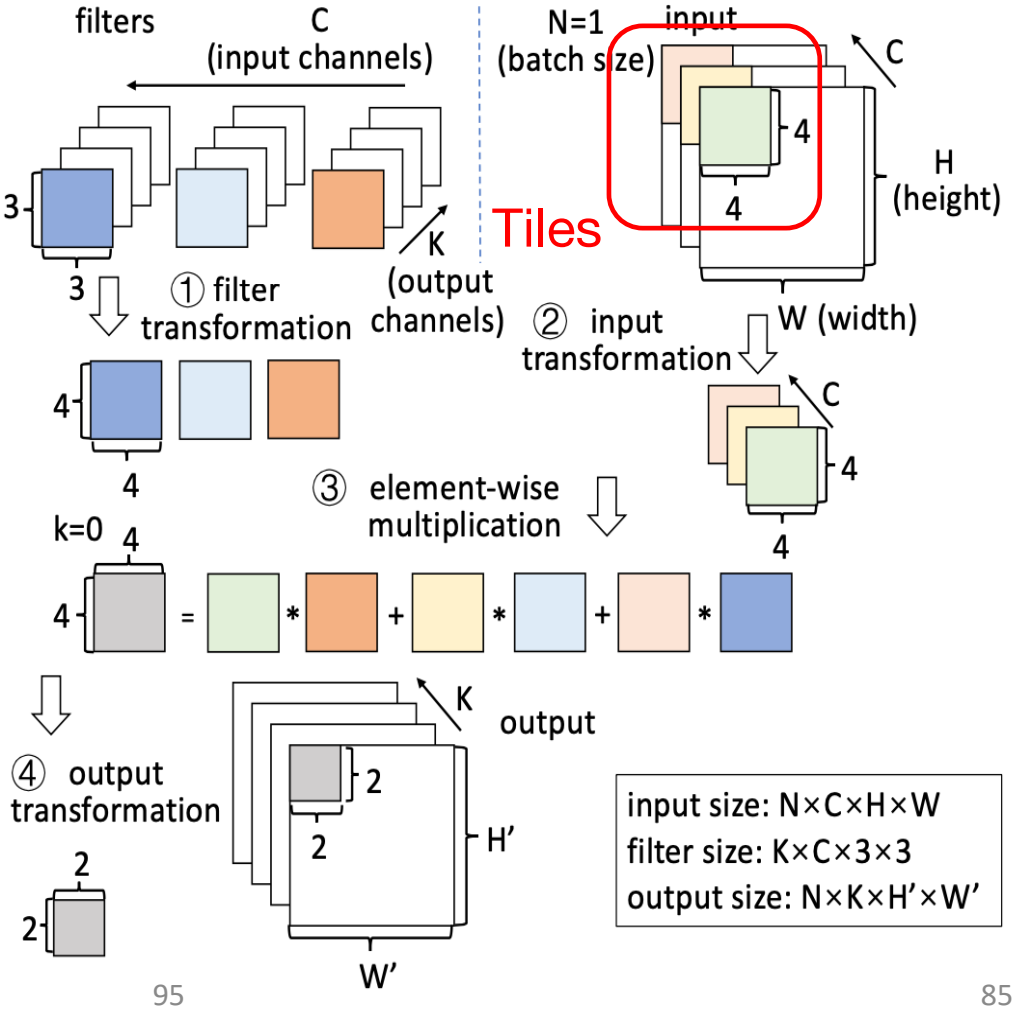
   → where multiplication ops are reduced.

2. Element-wise multiplication

3. Output transformation

Key Insight :
similarity exist in tiles.

The workflow of Winograd convolution.



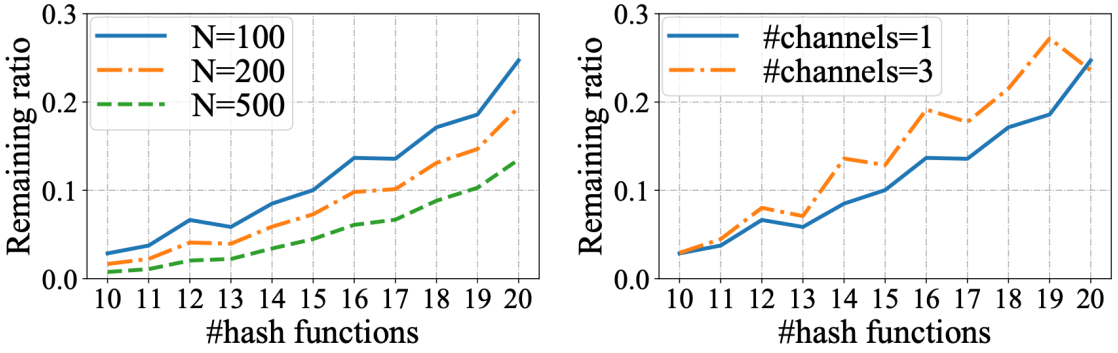
4 Deep Reuse Winograd

Opportunity: *Tiles* in Winograd convolution have strong similarities.

Observation: CifarNet on CIFAR-10

- Tiles have strong similarities

Importance: Deep reuse & Winograd: computation



(a) Single channel. (b) Multiple channels.

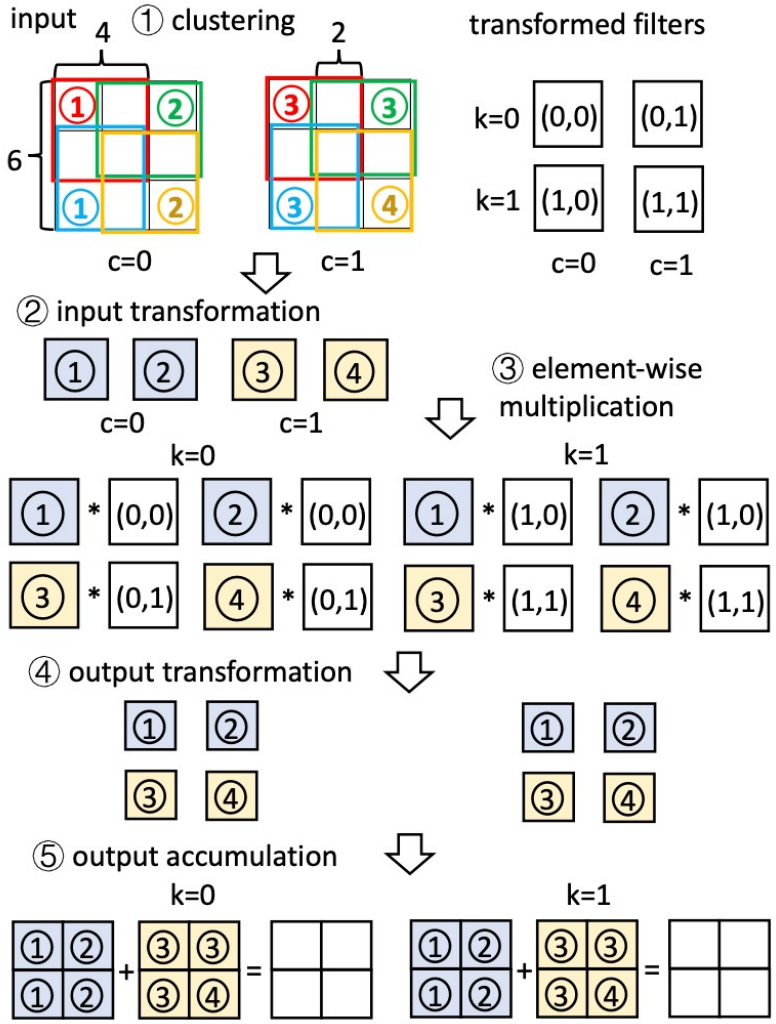
[PPoPP'21] "POSTER: Exploring Deep Reuse in Winograd CNN Inference", Ruofan Wu, Feng Zhang, Zhen Zheng, Xiaoyong Du, Xipeng Shen. 26th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, Seoul, S. Korea (PPoPP 2021).

4 Deep Reuse Winograd

Case study

1. Clustering

   → LSH
2. Transformation
3. Matrix multiplication
4. Output transformation
5. Output accumulation



The workflow of DREW.

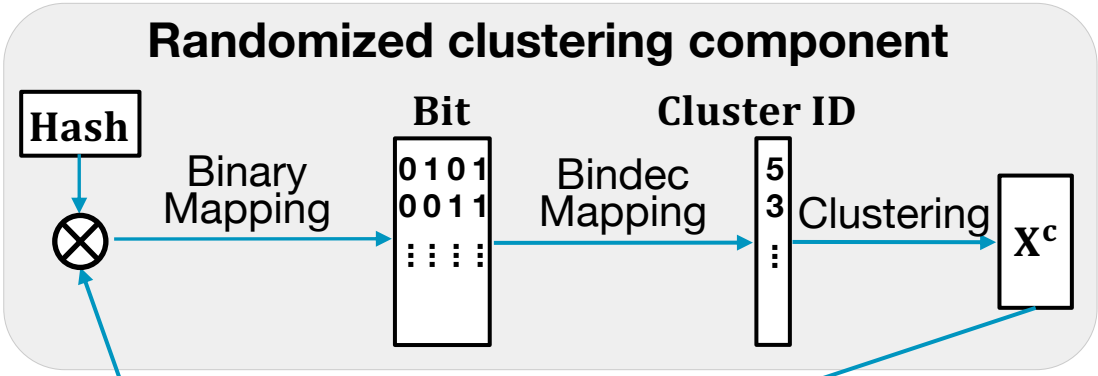
[WWW'22] "DREW: Efficient Winograd CNN Inference with Deep Reuse", Ruofan Wu, Feng Zhang, Jiawei Guan, Zhen Zheng, Xipeng Shen, Xiaoyong Du. The Web Conference, 2022.

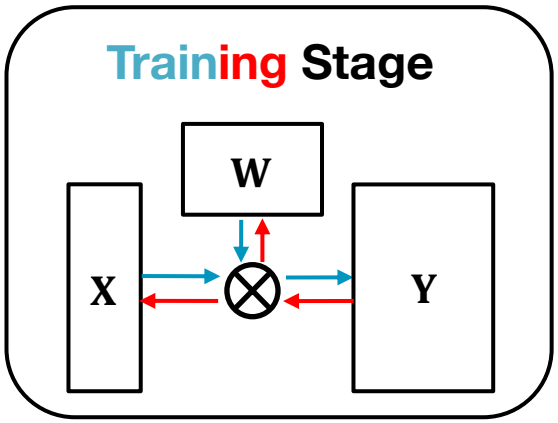
5 TREC: Transient Redundancy Elimination-based Convolution

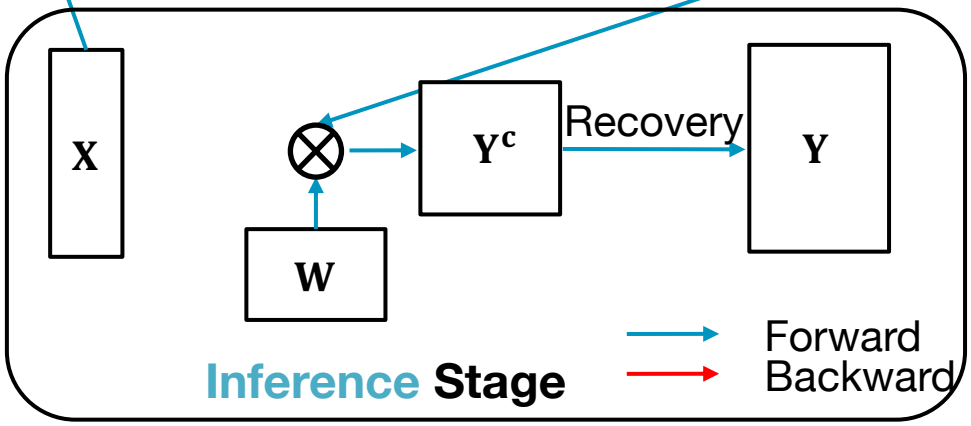
Problem: Over 5% accuracy fluctuations in different runs.

Cause:

1. Treat transient redundancy in an Ad-hoc manner.
2. Non-deterministic: random hash vectors.



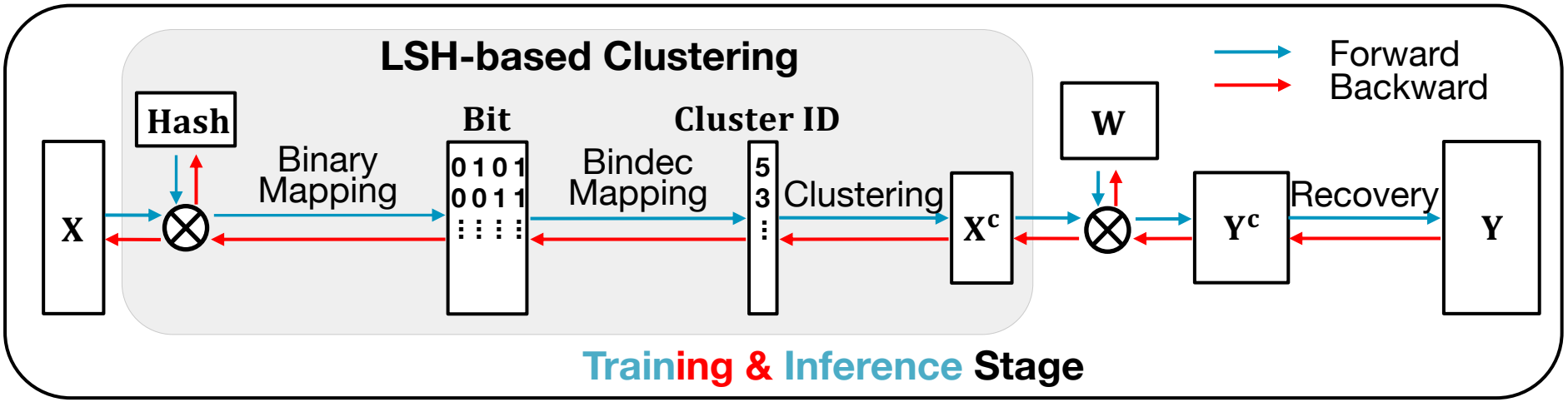




5 TREC: Transient Redundancy Elimination-based Convolution

- **Insight**: Can the detection and elimination of transient redundancy be integrated into the CNN training process, so that they become part of its inherent architecture?
- Can Hash vectors be jointly learnt?

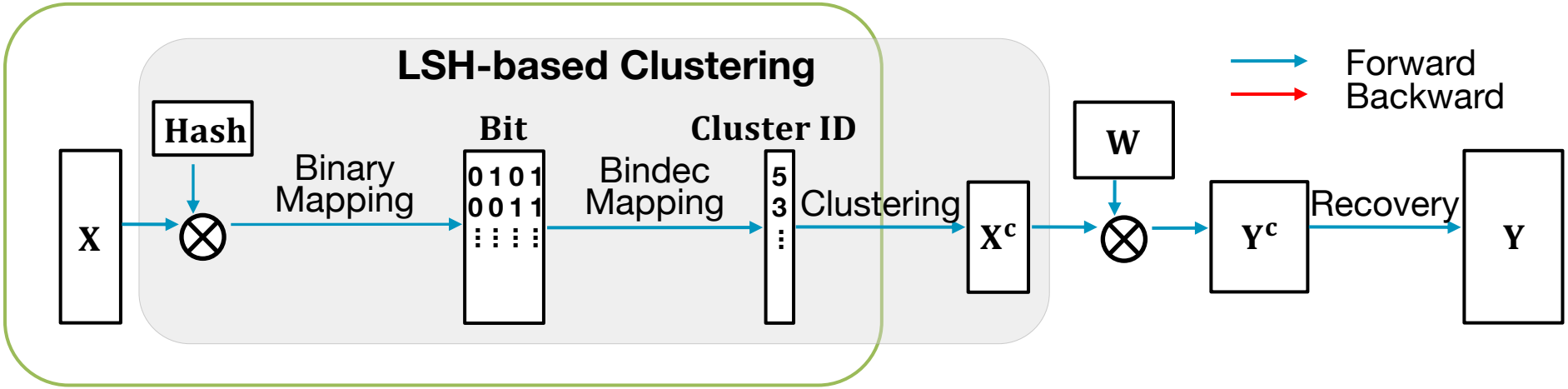
➢ TREC: Transient redundancy elimination-based convolution



[NeurIPS'22] " TREC: Transient Redundancy Elimination-based Convolution", Jiawei Guan, Feng Zhang, Jiesong Liu, Hsin-Hsuan Sung, Ruofan Wu, Xiaoyong Du, Xipeng Shen. NeurIPS, 2022.

5 TREC: Transient Redundancy Elimination-based Convolution

A closer look at redundancy elimination.



$$\underset{X}{\begin{pmatrix}1&2&3\\2&3&4\\1&2&1\\1&1&3\end{pmatrix}} \times \underbrace{\begin{pmatrix}1&1&-1&-1\\-1&0&1&-1\\0&1&-1&1\end{pmatrix}}_{\text{H=4}} = \underbrace{\begin{pmatrix}-1&4&-2&0\\-1&6&-3&-1\\-1&2&0&-2\\0&4&-3&1\end{pmatrix}}_{\text{H=4}} \xrightarrow{\text{Binary Mapping}} \underbrace{\begin{pmatrix}0&1&0&0\\0&1&0&0\\0&1&0&0\\0&1&0&1\end{pmatrix}}_{\text{H=4}} \xrightarrow{\text{Bindec Mapping}} \begin{pmatrix}4\\4\\4\\5\end{pmatrix}$$

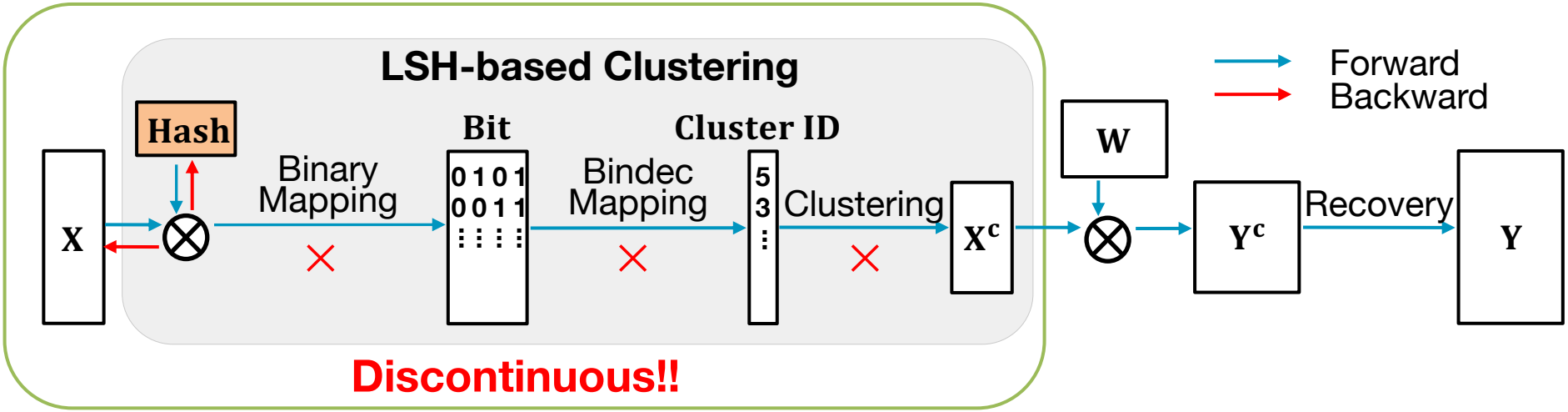
Hash Projected Bit Cluster ID

$$f(x) = \begin{cases}1, & x > 0\\0, & x \le 0\end{cases}$$

$2^{\mathrm{H}} = 16$ possible clusters

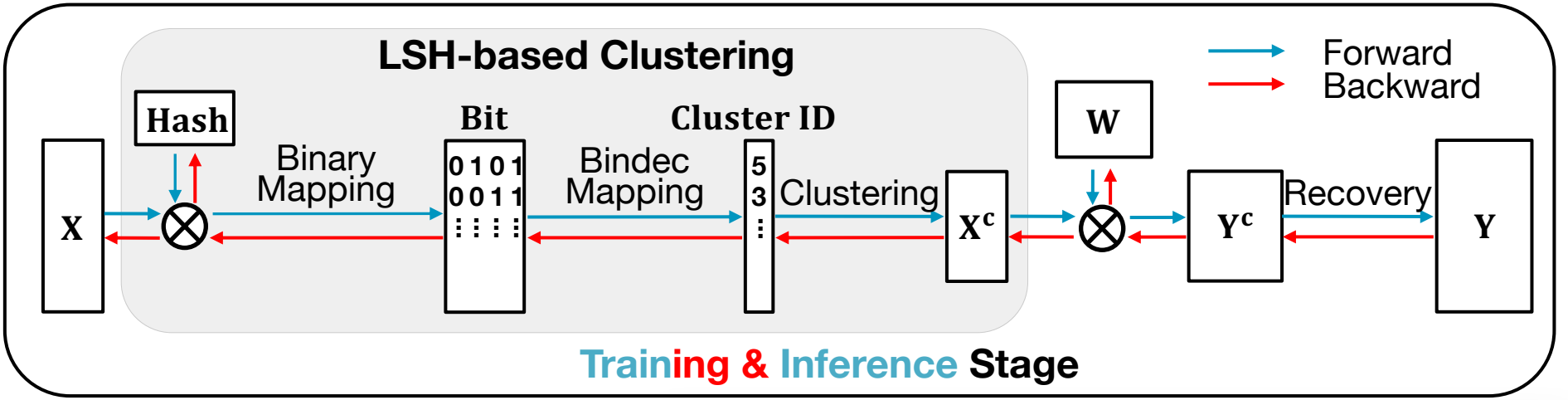
5 TREC: Transient Redundancy Elimination-based Convolution

Challenge: The discrete nature of LSH-based clustering.



all segmented mappings, which are discontinuous functions that cannot be derived.

5 TREC: Transient Redundancy Elimination-based Convolution

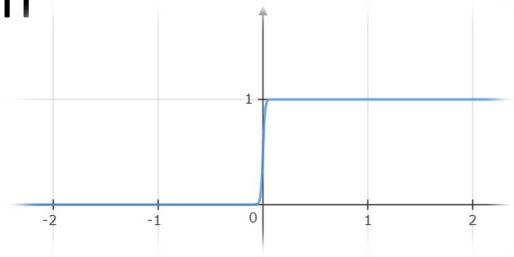


$$X \quad \text{Hash} \quad \text{Projected} \quad \text{Bit}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & -1 & -1 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 4 & -2 & 0 \\ -1 & 6 & -3 & -1 \\ -1 & 2 & 0 & -2 \\ 0 & 4 & -3 & 1 \end{pmatrix} \xrightarrow{\text{Binary Mapping}} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Example: Binary Mapping → Binary Approximation

$$f(x) = \begin{cases} 1, & x > 0 \\ 0, & x \le 0 \end{cases} \Longrightarrow sigmoid(x) = \frac{1}{1 + e^{-\alpha x}}$$



Summary

- Application of compressed data direct computing technology in text, database, graph, stream, and machine learning workloads

Thanks!
Any questions?

Part 1: Text

- [SIGMOD'24] "Homomorphic Compression: Making Text Processing on Compression Unlimited", JiaWei Guan, Feng Zhang, Siqi Ma, Kuangyu Chen, Yihua Hu, Yuxing Chen, Anqun Pan, Xiaoyong Du. SIGMOD 2024.
- [VLDBJ'20] "TADOC: Text Analytics Directly on Compression", Feng Zhang, Jidong Zhai, Xipeng Shen, Dalin Wang, Zheng Chen, Onur Mutlu, Wenguang Chen, Xiaoyong Du. VLDB Journal.
- [ICDE'20] "Enabling Efficient Random Access to Hierarchically-Compressed Data", Feng Zhang, Jidong Zhai, Xipeng Shen, Onur Mutlu, and Xiaoyong Du. IEEE ICDE, 2020.
- [VLDB'18] "Efficient Document Analytics on Compressed Data: Method, Challenges, Algorithms, Insights", Feng Zhang, Jidong Zhai, Xipeng Shen, Onur Mutlu, and Wenguang Chen. The 44th International Conference on Very Large Data Bases, Rio de Janeiro, Brazil, August 27-31, 2018.
- [TPDS'21] "Exploring Data Analytics without Decompression on Embedded GPU Systems", Zaifeng Pan, Feng Zhang, Yanliang Zhou, Jidong Zhai, Xipeng Shen, Onur Mutlu, Xiaoyong Du. TPDS, 2021.
- [ICDE'21] "G-TADOC: Enabling Efficient GPU-Based Text Analytics without Decompression" , Feng Zhang, Zaifeng Pan, Yanliang Zhou, Jidong Zhai, Xipeng Shen, Onur Mutlu, and Xiaoyong Du. IEEE ICDE, 2021.
- [SC'22] " Optimizing Random Access to Hierarchically-Compressed Data on GPU ", Feng Zhang, Yihua Hu, et al. SC, 2022.
- [ICDE'24] "Enabling Efficient NVM-Based Text Analytics without Decompression", Xiaokun Fang, Feng Zhang, Junxiang Nong, Mingxing Zhang, Puyun Hu, Yunpeng Chai, Xiaoyong Du. ICDE 2024.
- [ICDE'24] "F-TADOC: FPGA-Based Text Analytics Directly on Compression with HLS", Yanliang Zhou, Feng Zhang, Tuo Lin, Yuanjie Huang, Saiqin Long, Jidong Zhai, Xiaoyong Du. ICDE 2024.

Part 2: DB

- [TPDS'22] "POCLib: A High-Performance Framework for Enabling Near Orthogonal Processing on Compression", Feng Zhang, Jidong Zhai, Xipeng Shen, Onur Mutlu, Xiaoyong Du. TPDS, 2022.
- [SIGMOD'22] CompressDB: Enabling Efficient Compressed Data Direct Processing for Various Databases, Feng Zhang, Weitao Wan, Chenyang Zhang, Jidong Zhai, Yunpeng Chai, Haixiang Li, Xiaoyong Du, SIGMOD, 2022.
- [TPDS'23] Compressed Data Direct Processing for Databases, Weitao Wan, Feng Zhang, Chenyang Zhang, Jidong Zhai, Yunpeng Chai, Haixiang Li, Xiaoyong Du, TPDS, 2023.

Part 3: Graph

- [SIGMOD'23] " CompressGraph: Efficient Parallel Graph Analytics with Rule-Based Compression", Zheng Chen, Feng Zhang, et al. NeurIPS, 2022.

Part 4: Stream

- [ICDE'23] " CompressStreamDB: Fine-Grained Adaptive Stream Processing without Decompression", Yu Zhang, Feng Zhang, et al. ICDE, 2023.

Part 5: ML

- [ASPLOS'23] "Space-Efficient TREC for Enabling Deep Learning on Microcontrollers", Jiesong Liu, Feng Zhang, Jiawei Guan, Hsin-Hsuan Sung, Xiaoguang Guo, Xiaoyong Du, Xipeng Shen. ASPLOS 2023.
- [WWW'22] "DREW: Efficient Winograd CNN Inference with Deep Reuse", Ruofan Wu, Feng Zhang, Jiawei Guan, Zhen Zheng, Xipeng Shen, Xiaoyong Du. The Web Conference, 2022.
- [NeurIPS'22] " TREC: Transient Redundancy Elimination-based Convolution", Jiawei Guan, Feng Zhang, et al. NeurIPS, 2022.

Thank you!

Any questions?

95