Fast Window Aggregate on Array Database by Recursive Incremental Computation

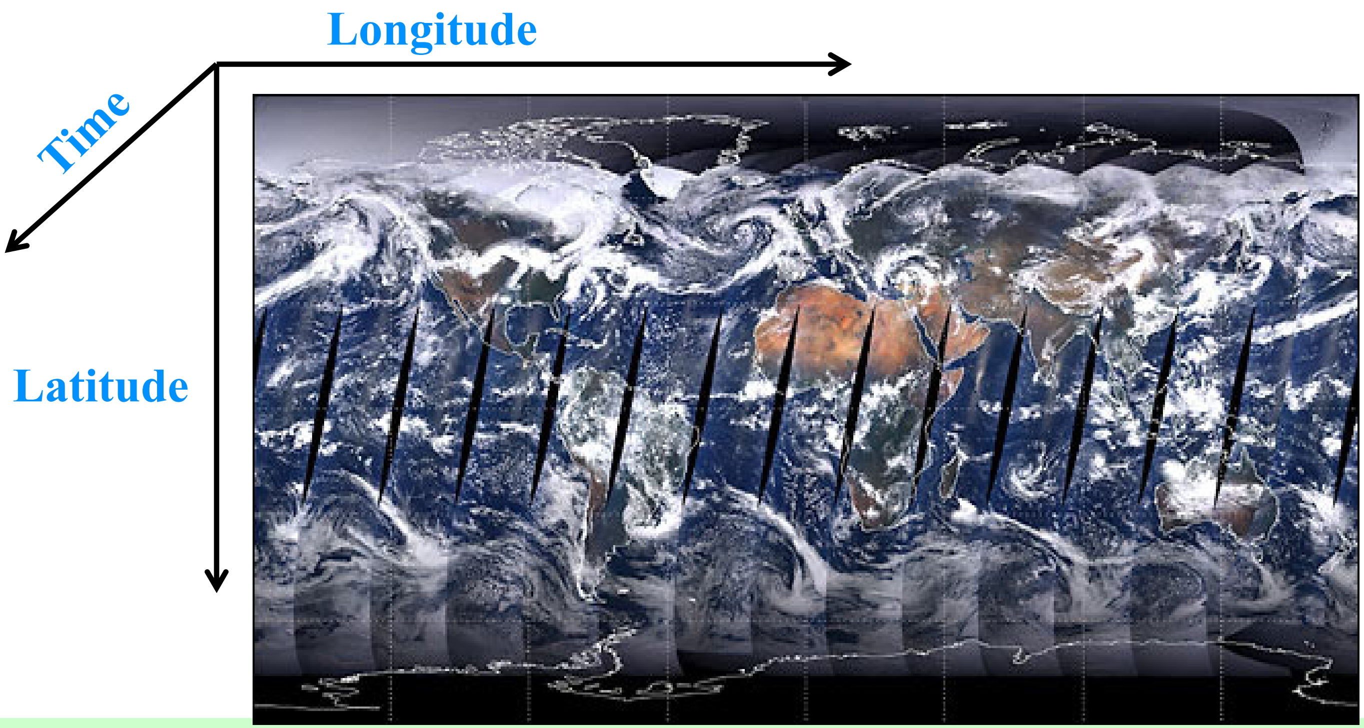
Li Jiang Hideyuki Kawashima Osamu Tatebe
University of Tsukuba, Japan

Agenda

- Background
- Proposed Method
- Evaluation
- Related Work
- Summary

Background: Big Scientific Data

- Huge multi-dimensional data is generated in many sciences (MODIS satellite, Subaru telescope, …)
- Naturally represented by array than relation



NASA Earth Science Data Product: MODIS Satellite Sensing Data

Credit: https://lpdaac.usgs.gov/dataset_discovery/modis

System – Array Database

- Array Database takes '**array**' instead of '**relation**' as basic data model [1,2,3].
- Elements
  - Dimensions: values determine coordinators of cells.
  - Attributes: same concept as in table, stored in cells.
- Advantages:
  - Suitable with multi-dimensional data.
  - Powerful data analysis tool for array data.

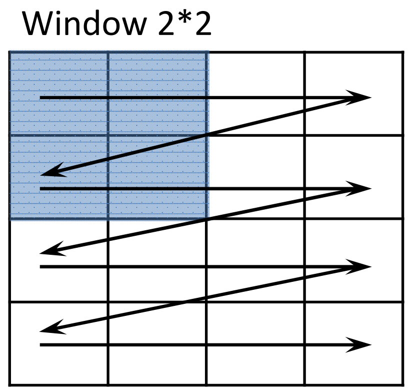
J \ I	[0]	[1]	[2]	[3]	[4]
[0]	(2, 0.7)	(5, 0.5)	(4, 0.9)	(2, 0.8)	(1, 0.2)
[1]	(5, 0.5)	(3, 0.5)	(5, 0.9)	(5, 0.5)	(5, 0.5)
[2]	(4, 0.3)	(6, 0.1)	(6, 0.5)	(2, 0.1)	(7, 0.4)
[3]	(4, 0.25)	(6, 0.45)	(6, 0.3)	(1, 0.1)	(0, 0.3)
[4]	(6, 0.5)	(1, 0.6)	(5, 0.5)	(2, 0.15)	(2, 0.4)

Array Data Model

Credit: the SciDB development team

[1] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann, "The multidimensional database system rasdaman," in SIGMOD Record, vol. 27, no. 2. ACM, 1998, pp. 575–577.
[2] M. Kersten, Y. Zhang, M. Ivanova, and N. Nes, "Sciql, a query language for science applications," in EDBT/ICDT Workshop on Array Databases.ACM, 2011, pp. 1–12.
[3] M. Stonebraker, J. Becla, D. J. DeWitt, K.-T. Lim, D. Maier, O. Ratzesberger, and S. B. Zdonik, "Requirements for science data bases and scidb." in CIDR, 2009, pp. 173–184.

Target Operator – Window Aggregates



- Application of window aggregate
  - Preprocess on raw data
  - Visualize results of other analysis tasks on purpose
- Task: compute aggregate functions over a moving window with given size.
  - **Arguments:**

Aggregate to compute → **max(v)**; **Source array** → ***arr***; **Window size** → **window (2,3)**

Query: select max(v) from *arr* grouping by window (2,3)

4	7	3	1	8
5	2	6	2	2
3	9	3	2	4
7	7	8	2	6

Source Array: *arr*

7	7	8	8	8
9	9	6	4	4
9	9	8	6	6
7	7	8	6	6

Result Array

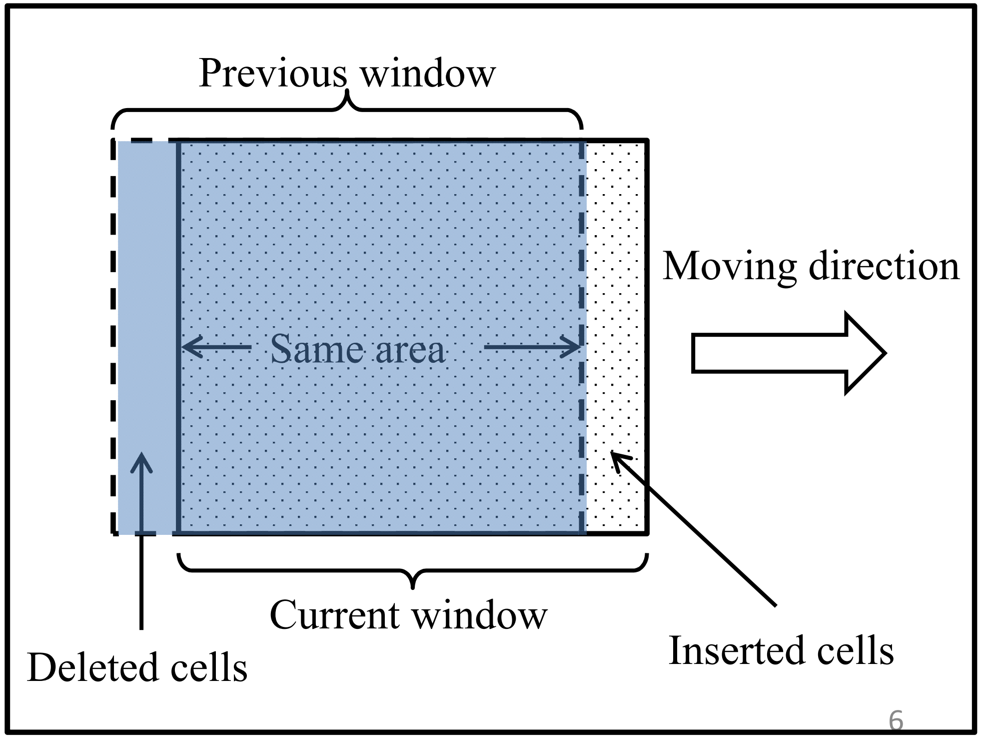
Aggregates: sum/avg, var/stdev, min/max

Naive Method – Inefficient

- Naive method
  - Scan all the elements in window, and compute its aggregate.
  - Inefficient: redundant calculation exists.

- Consider adjacent windows:
  - Large overlapping area.
  - Few cells are different.

- Large common area
  - Re-compute the same area ?
  - Waste of Resource.



Agenda

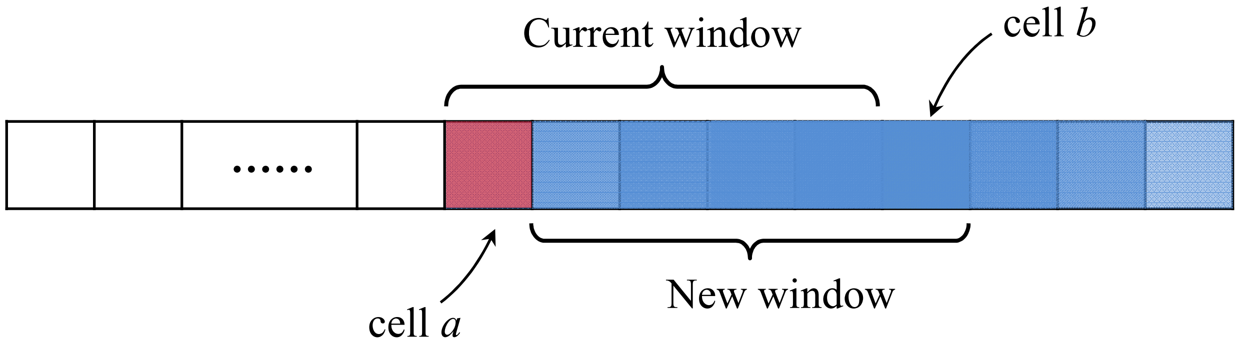
- Background
- Proposed Method
- Evaluation
- Related Work
- Summary

Proposal Overview

- Central Idea: Incremental Computation (IC) Scheme
  - Goal: eliminate redundant calculation
  - Simple trick: buffer and reuse previously computed intermediate aggregate values
- Previous Work
  - Basic IC method [4]: reduces redundant calculation in one dimension
- Proposal
  - Recursive IC method: eliminates all redundant calculation in every dimension
- Six aggregate functions improved
  - sum/avg, var/stdev, min/max

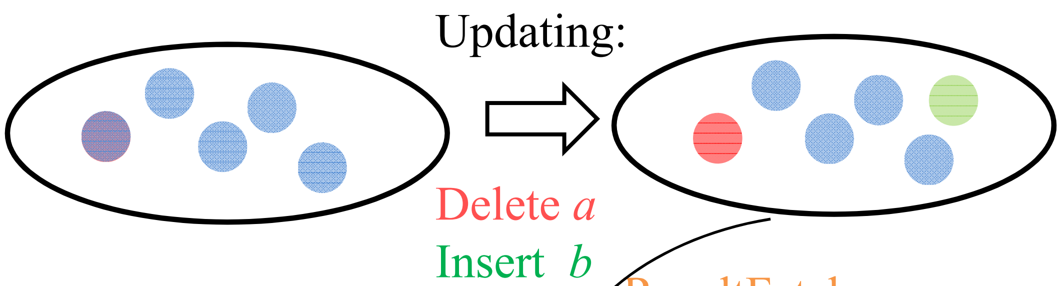
[4] Li Jiang, Hideyuki Kawashima, Osamu Tatebe: Incremental window aggregates over array database. IEEE International Conference on Big Data, pages 183–188, 2014.

Primary Task : 1-D IC process

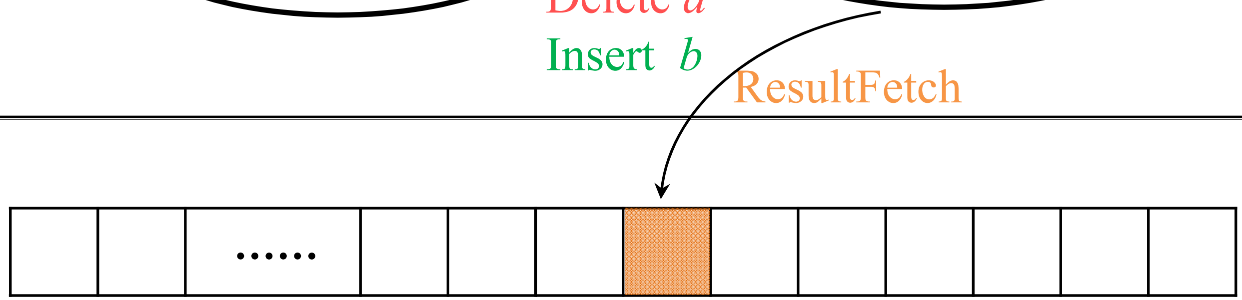


Source Array (1-D)

Buffer Tool (to buffer intermediate result and help achieve incremental computation)



Result Array



For different group of aggregate operator, different data structure is designed to achieve efficient IC.

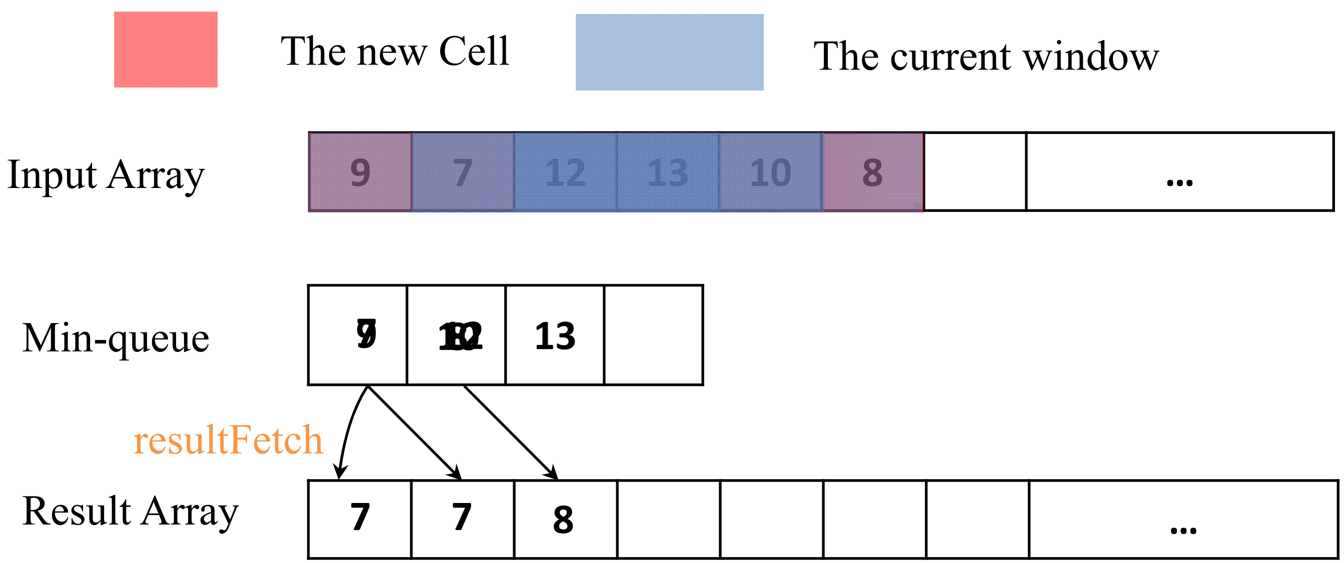
- Sum-list: sum/avg
- Var-list: var/stdev
- Queue: min/max

Buffer Tool Example: Min Queue

- Min Queue: un-decreasing circle queue
  - Updates: maintain the queue so that,

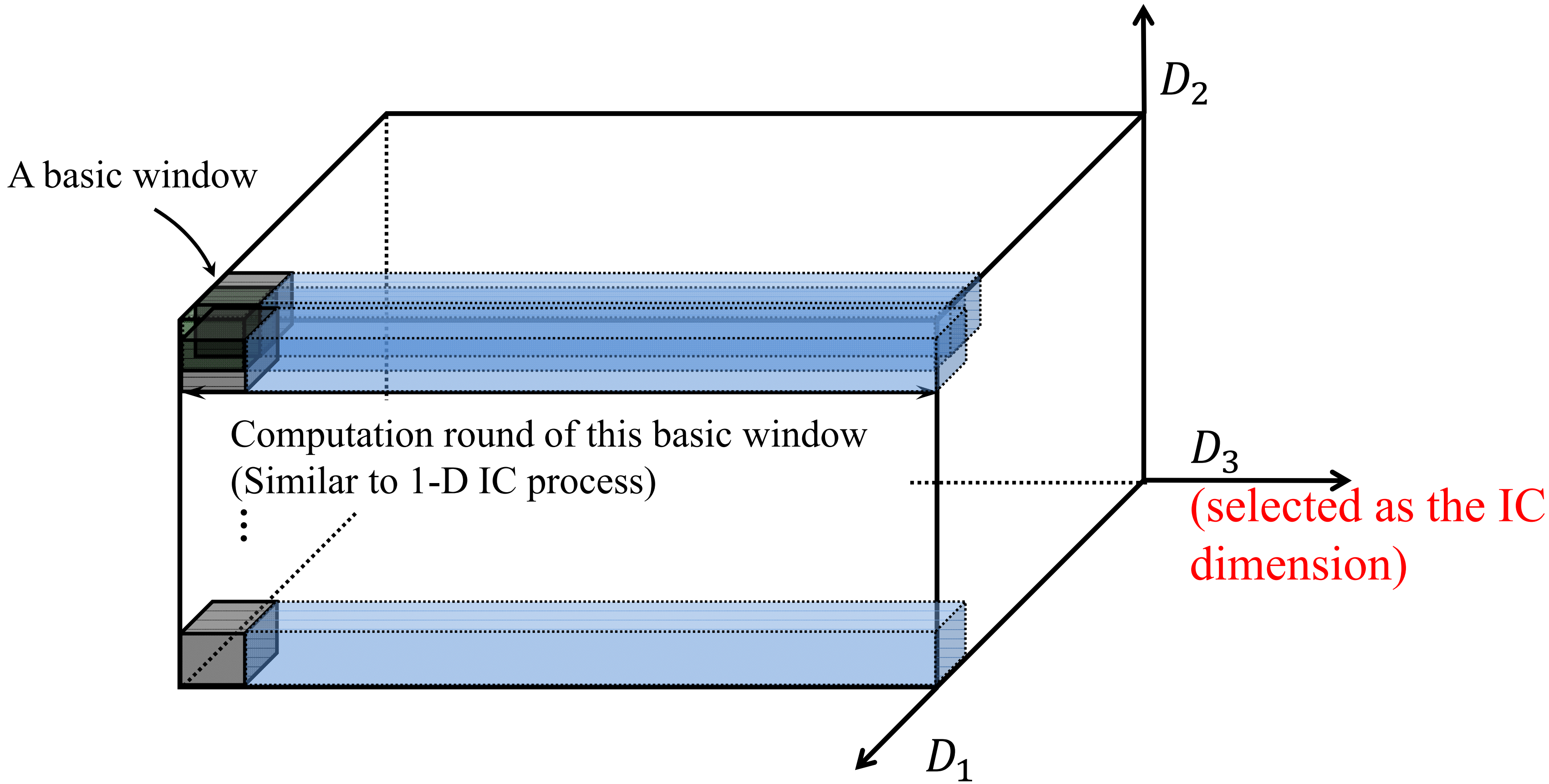
    For Queue$[a_1, a_2, a_3 \ldots, a_n]$, it satisfies:

    $\forall\ i, j\ \in [1, n]\ that\ \ i < j, a_i \leq a_j$
  - Result Fetch: return the head element (→ the smallest element)
- Example: window size = 4



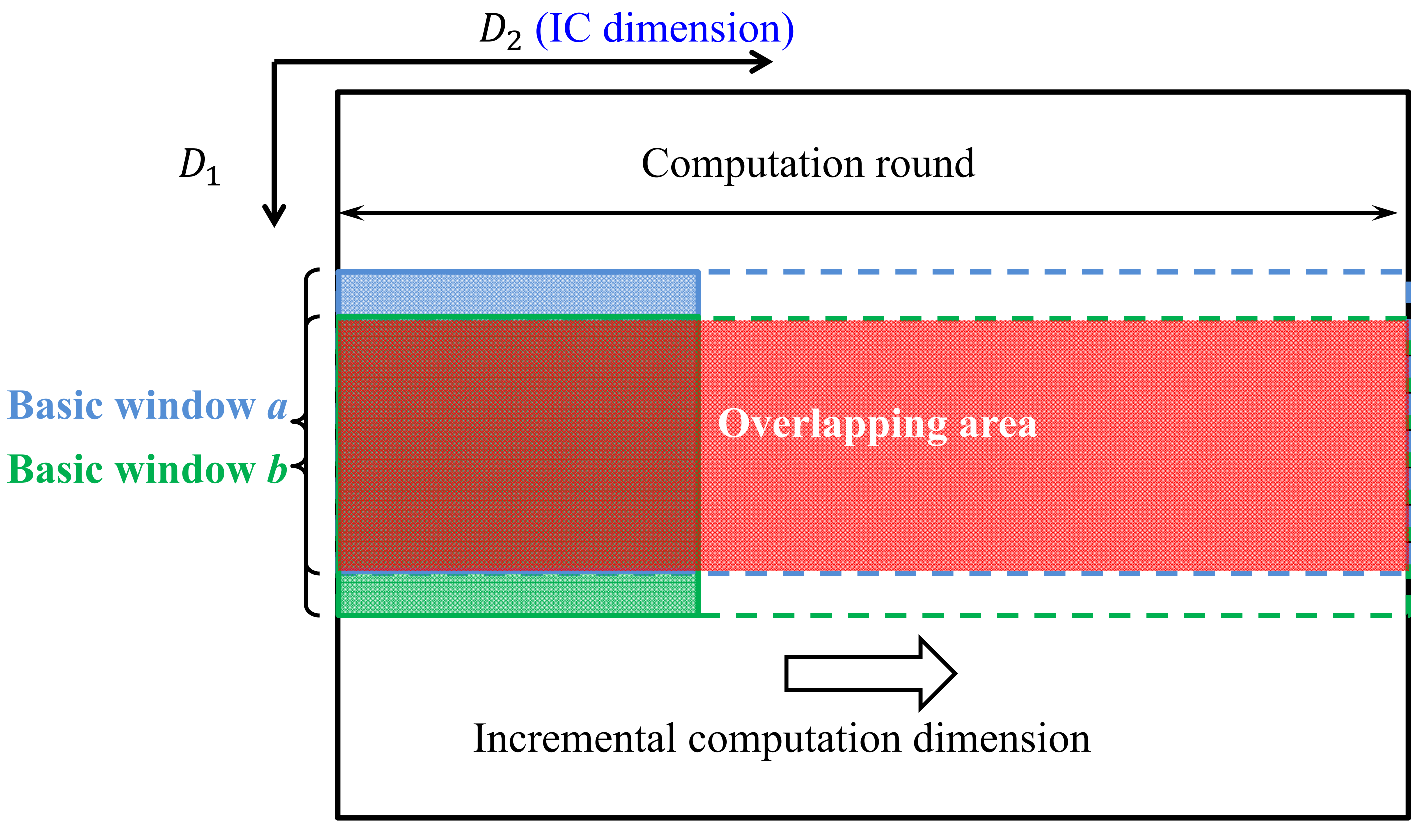
1-D to *n*-D: Basic IC Method

- To apply IC scheme from 1-D to n-D window aggregate.
- Process
  - Solve a n-D window aggregate task as in multiple 1-D subtasks.
  - For each 1-D subtask, borrow the 1-D IC process with little modification



Defect of basic IC method

Actually, redundant calculation still exist



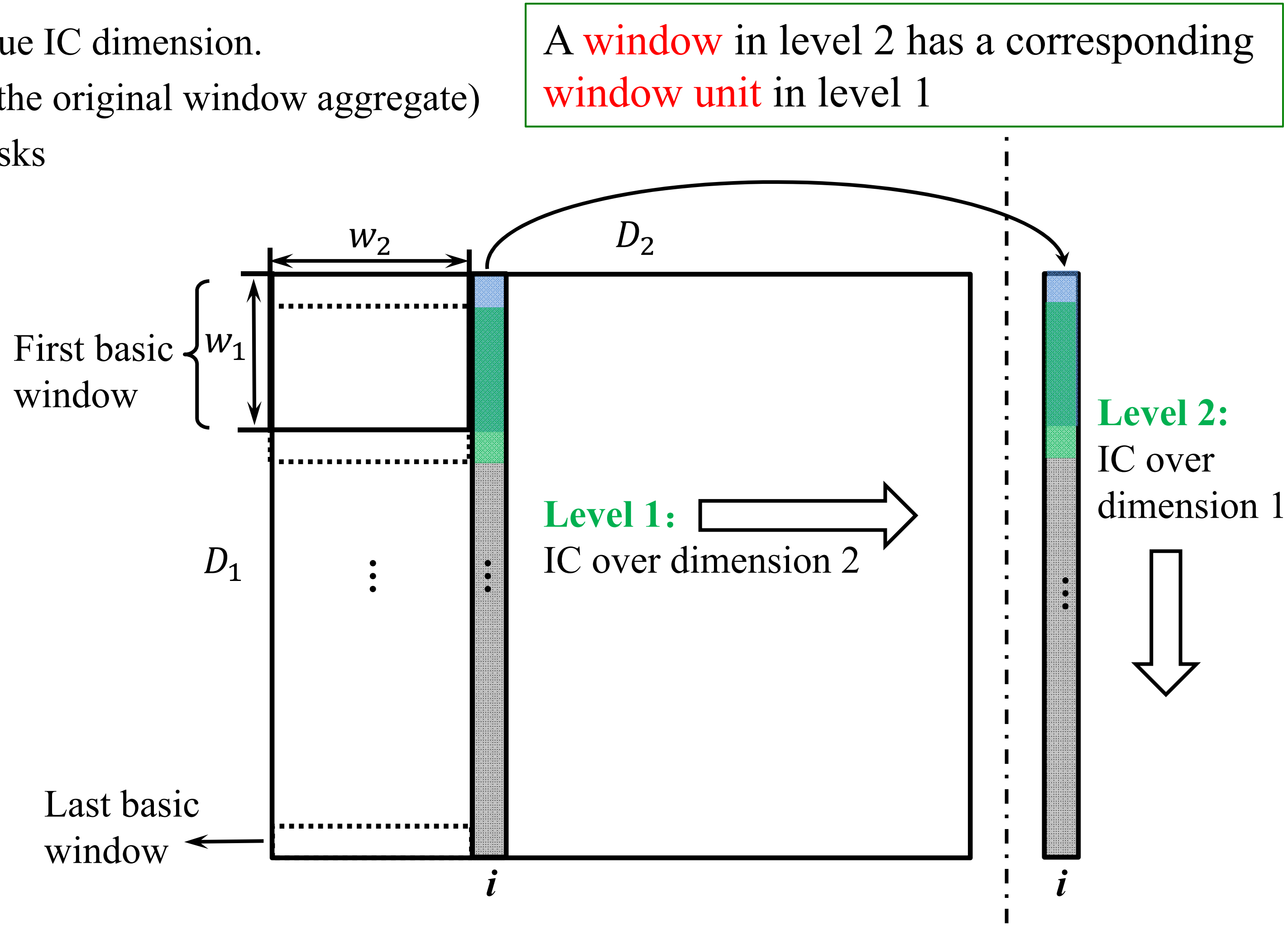
- Basic IC eliminates redundant works in IC dimension, but in other dimensions, unnecessary calculation still exists.

Proposal : Recursive IC Method

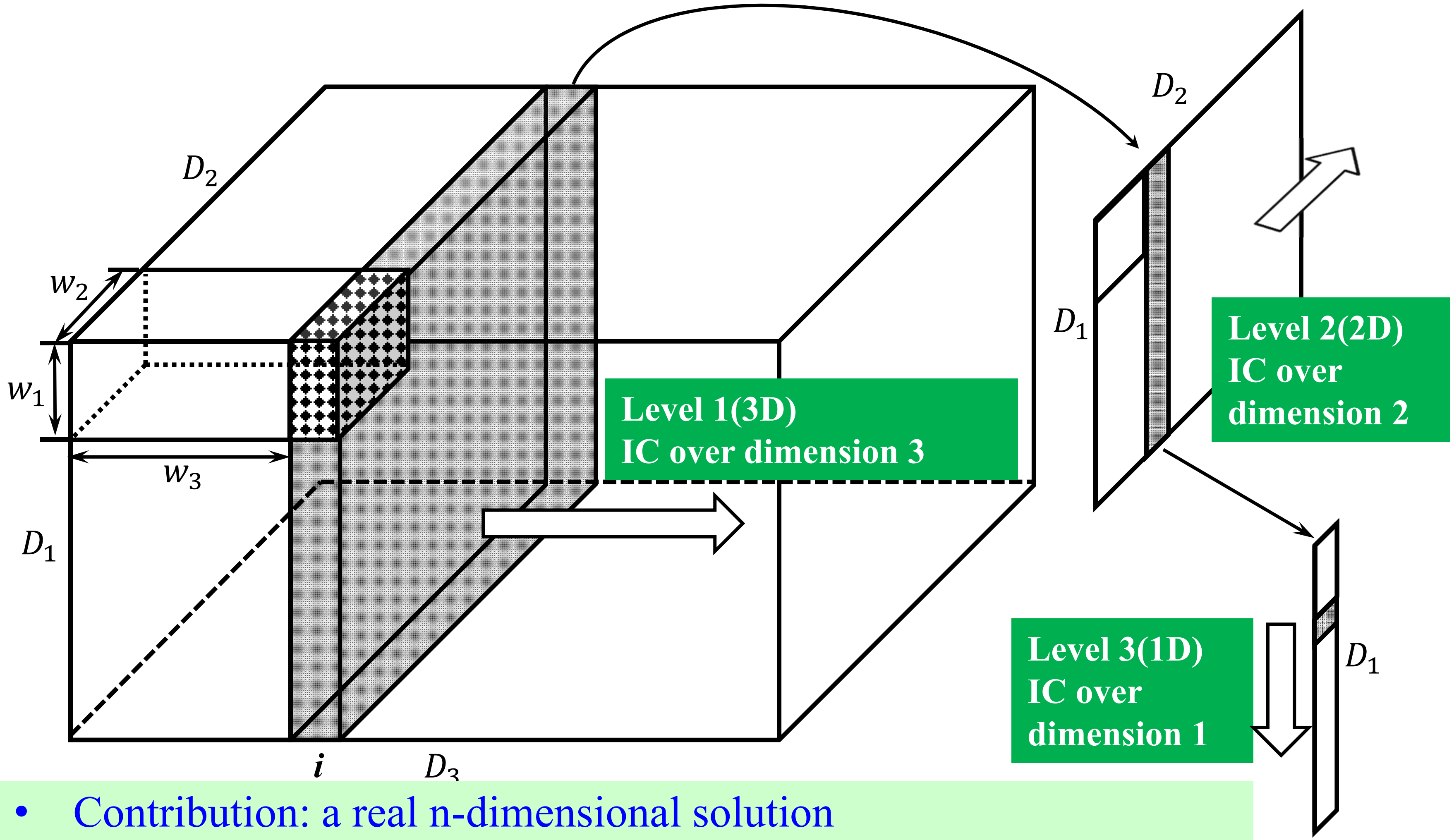
- **Recursive Dimensionality Reduction**
  - Keeping breaking a *n*-D window aggregate down to multiple smaller window aggregates.
- Multiple levels workflow

  Each level has its unique IC dimension.
  - Level 1: *n*-D task (the original window aggregate)
  - Level 2: (*n-1*)-D tasks

  ……
  - Level n: *1*-D tasks



Recursive IC Method (3D example)



- Contribution: a real n-dimensional solution
  - No redundant calculation during the whole process at all
- Tradeoff: more extra space cost, one buffer tool maintained for each computation round

Agenda

- Background
- Proposed Method
- Evaluation
  - Overall Comparison
  - Earth Science Benchmark
  - Synthetic Workload
- Related Work
- Summary

Evaluation

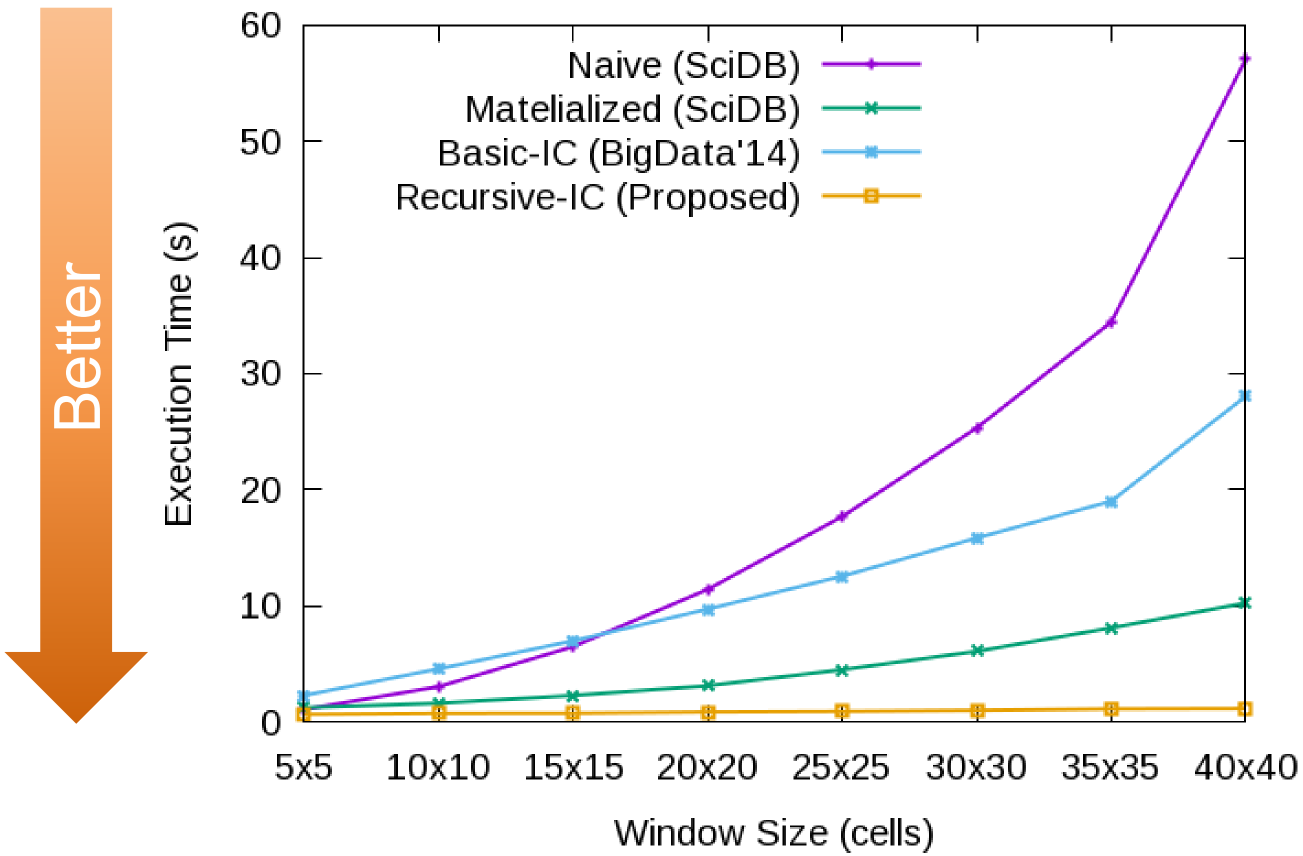
- SciDB
  - An open-source array database system
  - Version : 14.12
  - Proposed method implemented into SciDB and tested comparing with SciDB's built-in naive method

- Environment

  A SciDB cluster consists of 4 nodes, each node has the same setting as
  - Operating System : CentOS 6.5
  - CPU : Intel(R) Xeon(R) E5620 2.40GHz
  - Main Memory : 24GB

Overall Comparison

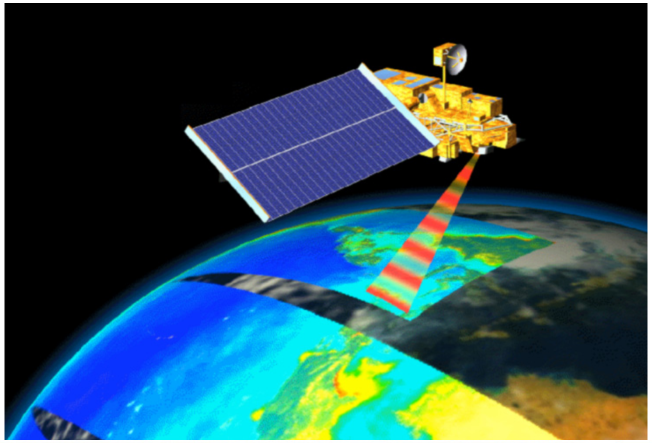
- Dimension: 2
- Array size: 1000 × 1000 (small)
- Operator: Variance (all 6 operator performs similar)
- Result: naïve (SciDB) and basic-IC are slow, will be omitted.

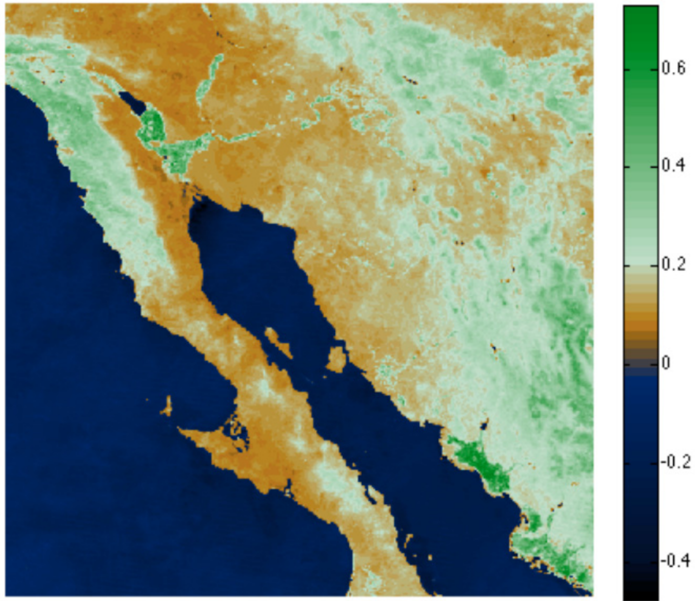


Earth Science Benchmark (1/3)

- A real application of earth scientific data analysis [5] [6]
  - Window average operator
  - Used to reduce resolution
  - On purpose of visualizing.
- Data: NASA MODIS product
  - 45 MODIS files downloaded (each 160MB)
  - Preprocessed, loaded into SciDB cluster
  - Sparse (a lot of empty cells, >30%)

Terra satellite scanning the Earth [5]



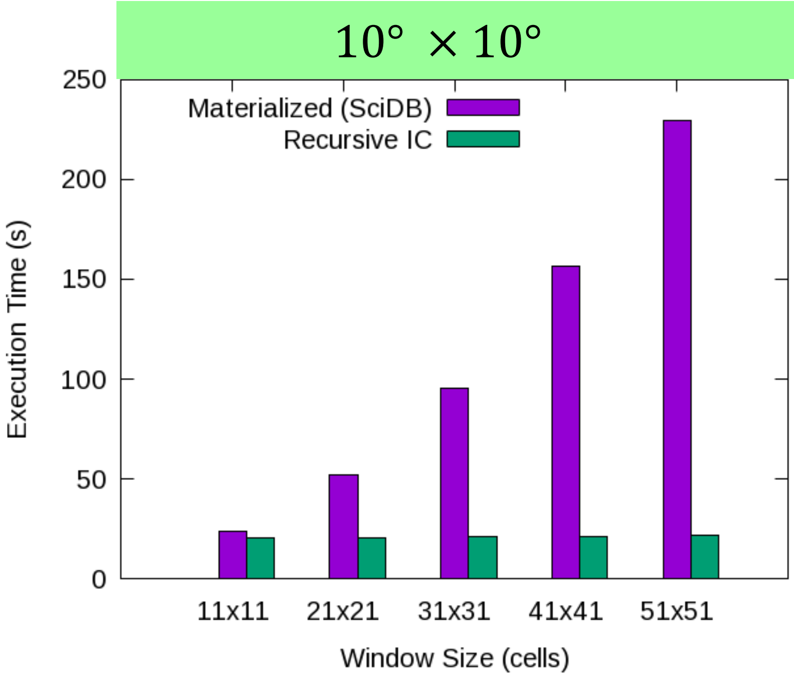


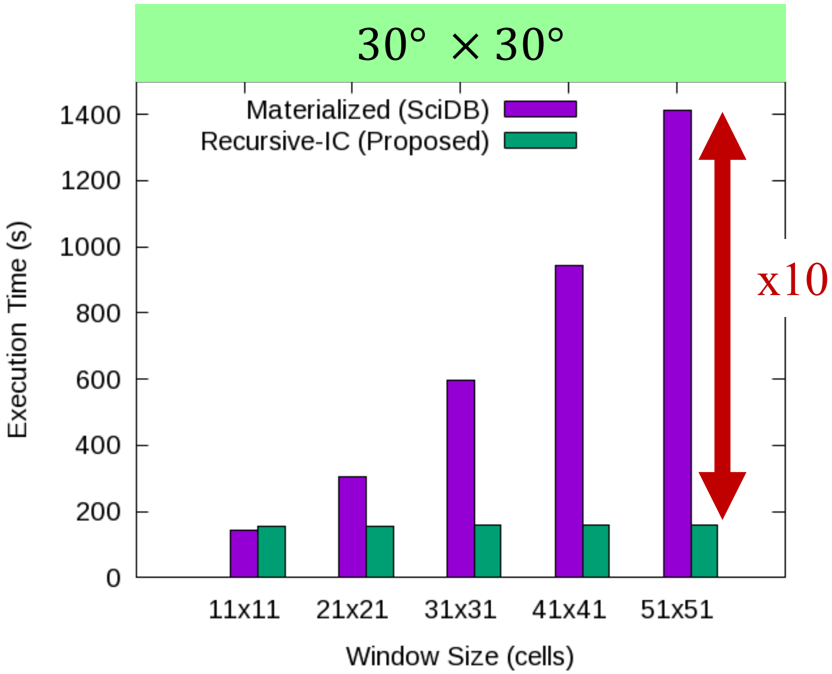
NDVI result visualized after window aggregate [6]

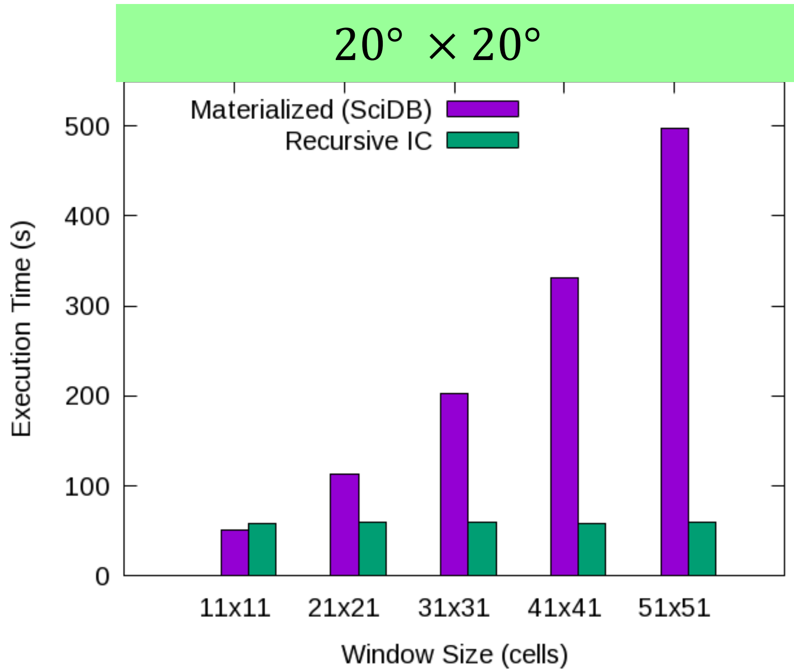
[5] Gary Lee Planthaber Jr. Modbase: A scidb-powered system for large-scale distributed storage and analysis of modis earth remote sensing data. PhD thesis, Massachusetts Institute of Technology, 2012.
[6] Earth science benchmark over modis data. http://people.csail.mit.edu/jennie/elasticity_benchmarks.html

Earth Science Benchmark (2/3)

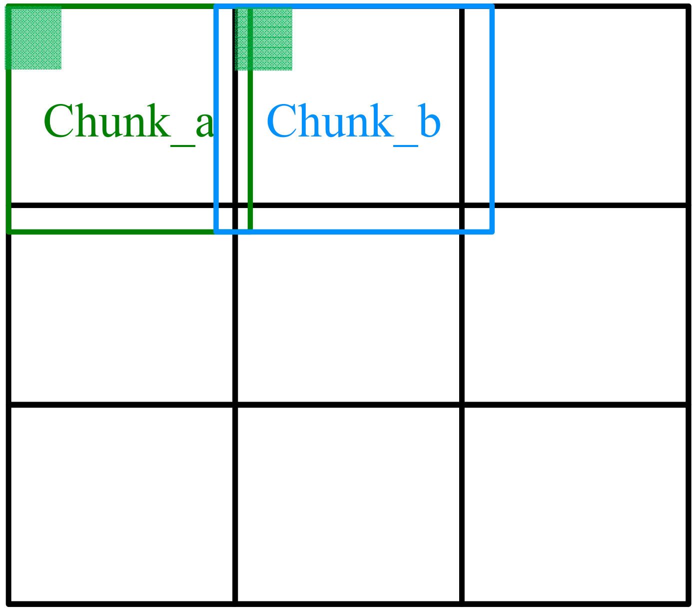
- Input: NDVI
- Window size: $0.05° \times 0.05°$
- Operator: average
- Result
  - For 30x30 case, x10 improvement.







Earth Science Benchmark (3/3)
Space Analysis



Extra Space Cost of Recursive IC

Extra Space (Array Scope)	
10° Granule	**19.47MB**
20° Granule	**77.90MB**
30° Granule	**175.27MB**

Extra Space(Chunk Scope)	**199KB**
Chunk Setting	1000×1000
Data Size Per Chunk	3.81MB

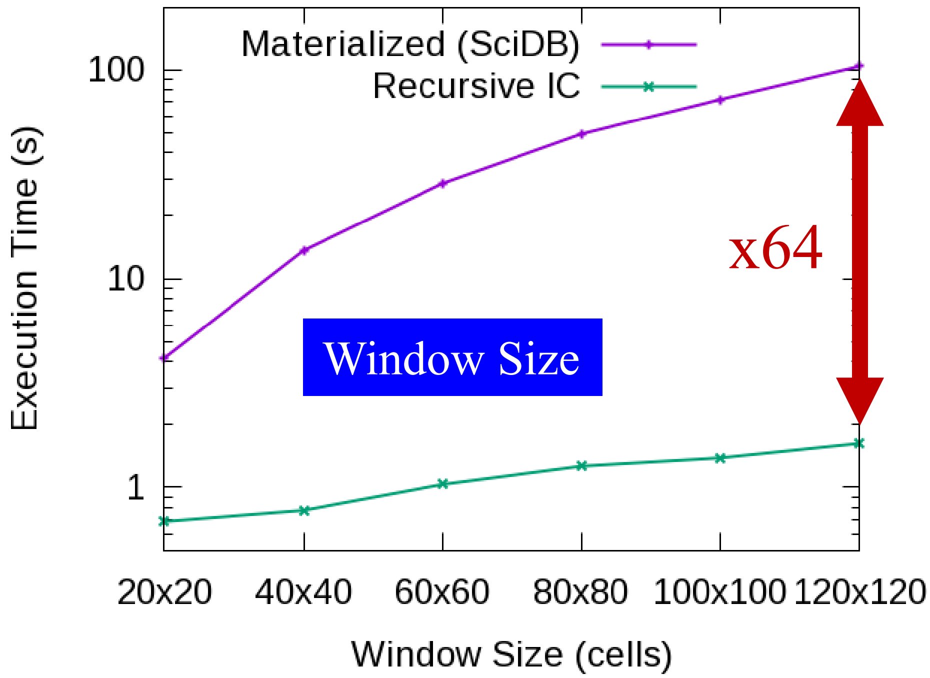
- Total Extra space cost of buffer tools seems big.
- Actually in SciDB, window aggregate is executed chunk by chunk
- Only one single chunk's buffer tools are maintained, totally acceptable.

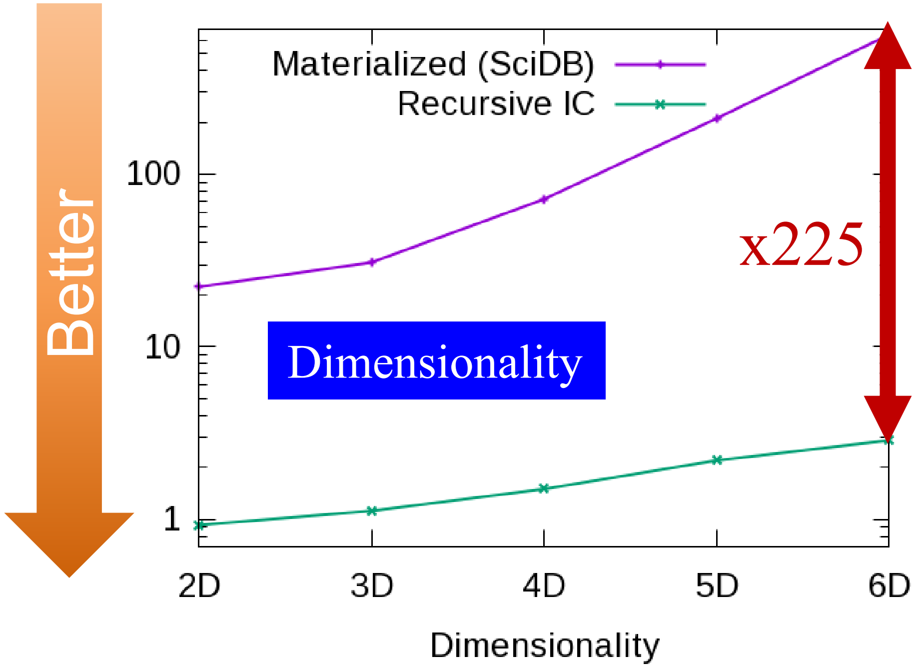
Area Size	**Array Size**	**Cells**	**Density**	**Data Size**
10° × 10°	10000 × 10000	28787550	28.79%	559MB
20° × 20°	20000 × 20000	90526766	22.63%	1.78GB
30° × 30°	30000 × 30000	240706765	26.75%	4.32GB

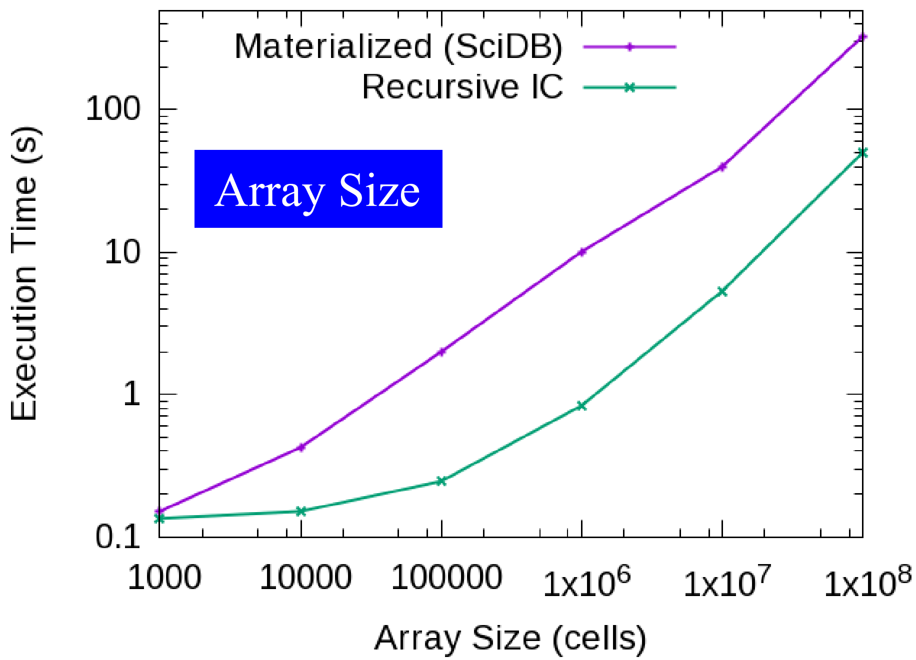
Synthetic Dataset

- Operator: variance
- Attribute values of the arrays were randomly generated in the range [0, 100,000].

Parameter	Window	Array	Dim.
Window		Fix	Fix
Array	Fix		Fix
Dim.	Fix	Fix	







Agenda

- Background
- Proposed Method
- Evaluation
- Related Work
- Summary

Related Work

- Incremental Computation of aggregates
  - Sliding window aggregate of stream data [7]
  - Temporal Aggregates of interval data [8]
  - → Similar basic ideas. Different targeting data types and queries. Hard to evaluate performance between their work with this one.
- Image processing
  - Similar incremental computation used to accelerate filter calculation
  - Difference: limited to 2 dimensions.
- Improving scientific features of array databases
  - Data versioning [9], Data uncertainty [10]

[7] Jin Li, David Maier etc. No Pane, No Gain: Efficient Evaluation of Sliding-Window Aggregates over Data Streams. SIGMOD Rec. 34, 1, 2005.
[8] Jun Yang, Jennifer Widom. Incremental computation and maintenance of temporal aggregates. VLDB J. Vol. 12, No. 3, pp. 262-283, 2003.
[9] A. Seering, P. Cudre-Mauroux, S. Madden, and M. Stonebraker, "Efficient versioning for scientific array databases," in ICDE, 2012, pp. 1013–1024.
[10] T. Ge and S. Zdonik, "Handling uncertain data in array database systems," in ICDE, 2008, pp. 140–1149.

Summary

- Proposal
  - Fast window aggregates with recursive incremental computation for sum/avg/var/stddev/min/max over array database.
- Result
  - Proposed recursive IC method is the fastest.
  - In sparse Earth science benchmark, recursive method is x10 faster.
  - In dense synthetic test, recursive method is x64 faster.
- Future direction
  - Find applications: dense data
    - Meteorological simulation
    - Cosmological simulation (with Subaru team)

Code is available on GitHib

https://github.com/ljiangjl/Recursive-IC-Window