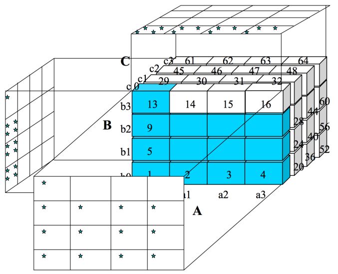
An Array-Based Algorithm for Simultaneous Multidimensional Aggregates

By Yihong Zhao, Prasad M. Desphande and Jeffrey F. Naughton

PRESENT BY CAROLYN LIANG
CMPT843



Recall: ROLAP vs MOLAP

Relational OLAP	Multi-dimensional OLAP
Stores as tuple e.g. (shoes, WestTown, 3-July-96, \$34)	Stores as sparse arrays e.g. Stores \$34 -The position in the array encodes(shoes, WestTown, 3-July-96)

Background and motivation

- Many efficient ROLAP algorithm.

- Naïve MOLAP Method is not good enough.

Want: MOLAP algorithm which is faster and less memory required.

Array Storage

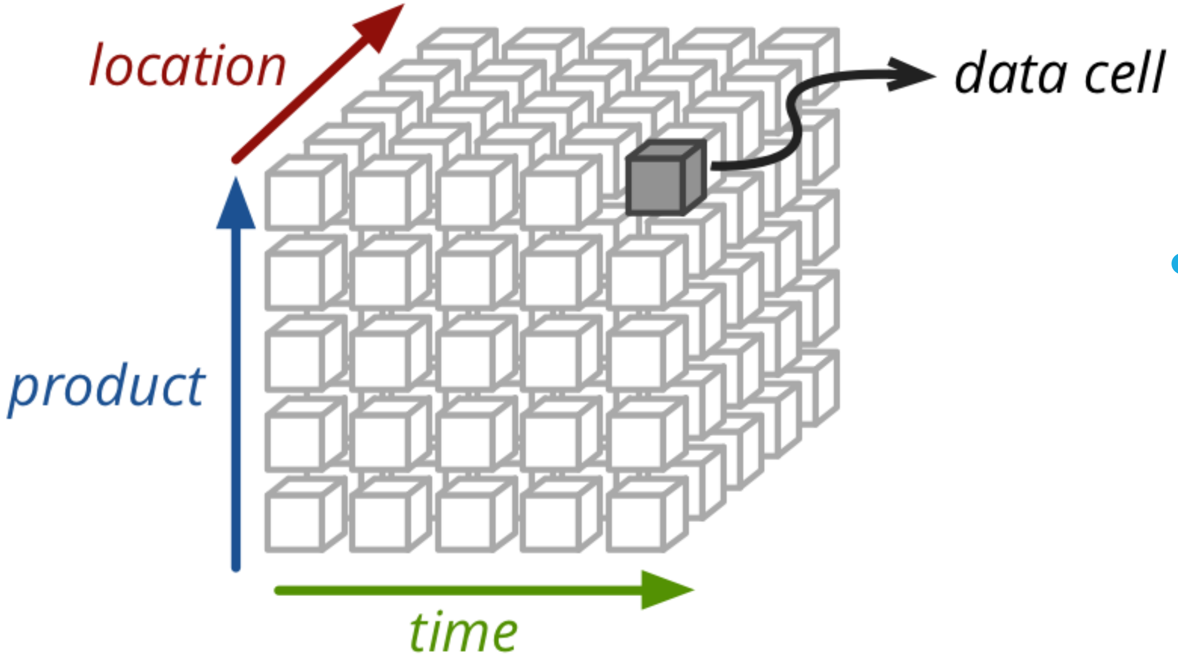
-Purpose: Load and store large, sparse arrays efficiently.

Main issues:

1. Do not fit in memory.

2. Many cells are empty.

Issue 1: Do not fit in memory



Chunking:

By Sarawagi: Efficient Organization of Large Multidimensional Arrays

- Divide n-dimensional arrays into smaller n-dimensional chunks, storing each chunk as one object on disk.

Issue 2: Many cells are empty

Compressing(if density <= 40%):

Apply “chunk-offset compression” to the sparse array.

For each valid array entry:

- **(offsetInChunk, data)**
  - offsetInChunk describes the <u>location</u> of this value in the sparse array
  - data stores <u>actual data</u> for the cell.

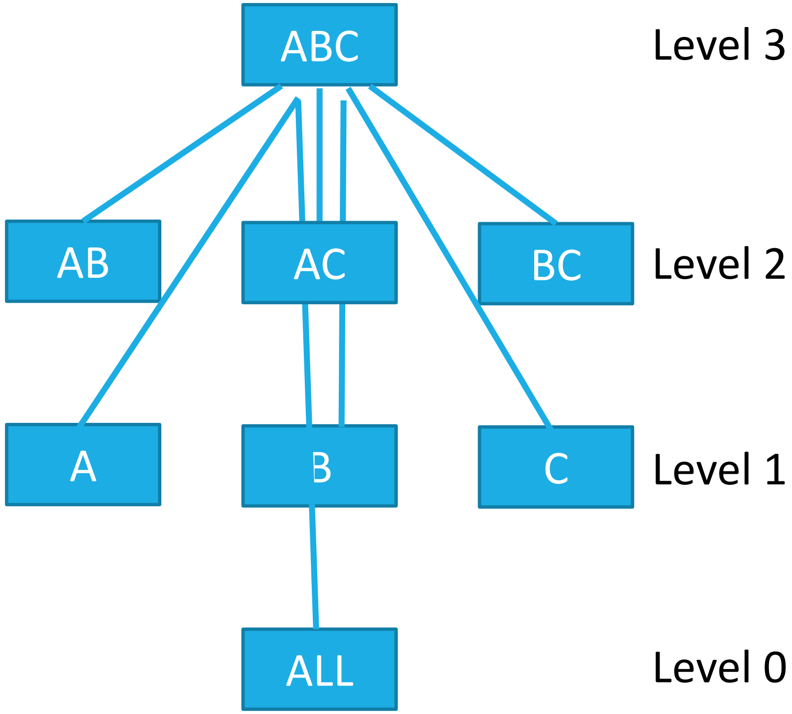
A Basic Array Cubing Algorithm

How to aggregate the cube over all dimensions?

For 3-D cube ABC, want AB, AC, BC, A, B, C and ALL.

Naïve approach:

aggregates from the initial ABC



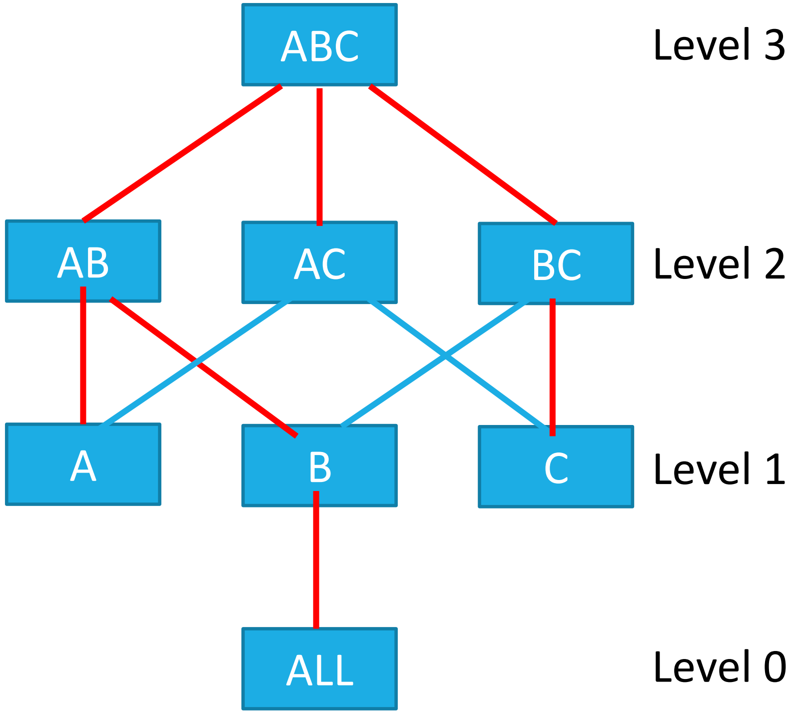
A Basic Array Cubing Algorithm-con't

How to aggregate the cube over all dimensions?

For 3-D cube ABC, want AB, AC, BC, A, B, C and ALL.

Another approach:

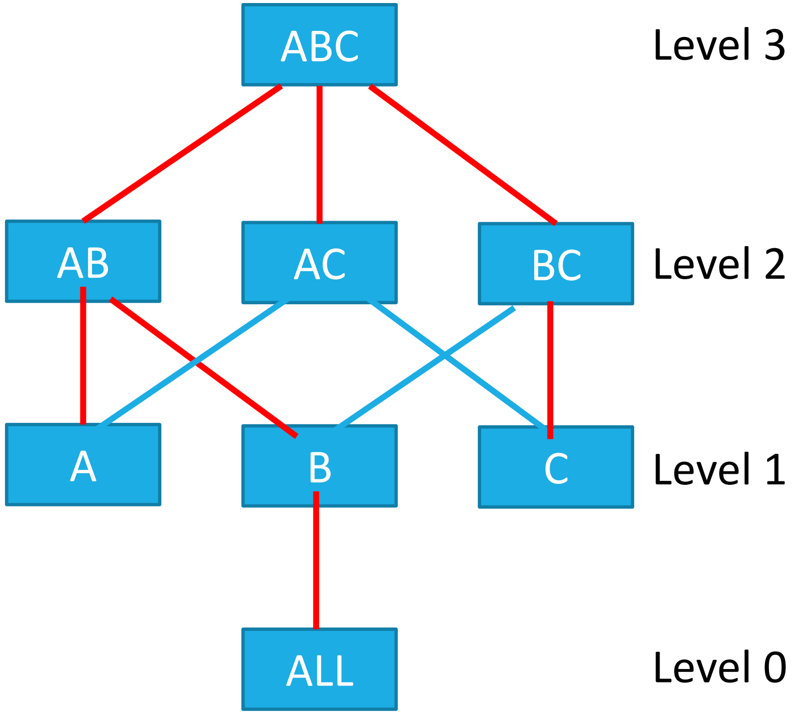
Minimum size spanning tree use hierarchical lattice representation of the data- min size parent.



A lattice view of the (A,B,C) database

A Basic Array Cubing Algorithm-con’t

Problem:
Suppose now we want to compute AB, AC and BC, how many times do we need to scan ABC?



A lattice view of the (A,B,C) database

The Single-Pass Multi-Way Array Cubing Algorithm

-Purpose: overlaps computation of different aggregations, avoiding the need for multiple scans over the data

- A **dimension order** $O = (D_{j1}, D_{j2}, \ldots D_{jn})$ defines the order in which dimensions are scanned.
- $|Di|$ = the size of dimension i
- $|Ci|$ = the size of the chunk for dimension i
- $|Ci| << |Di|$ in general

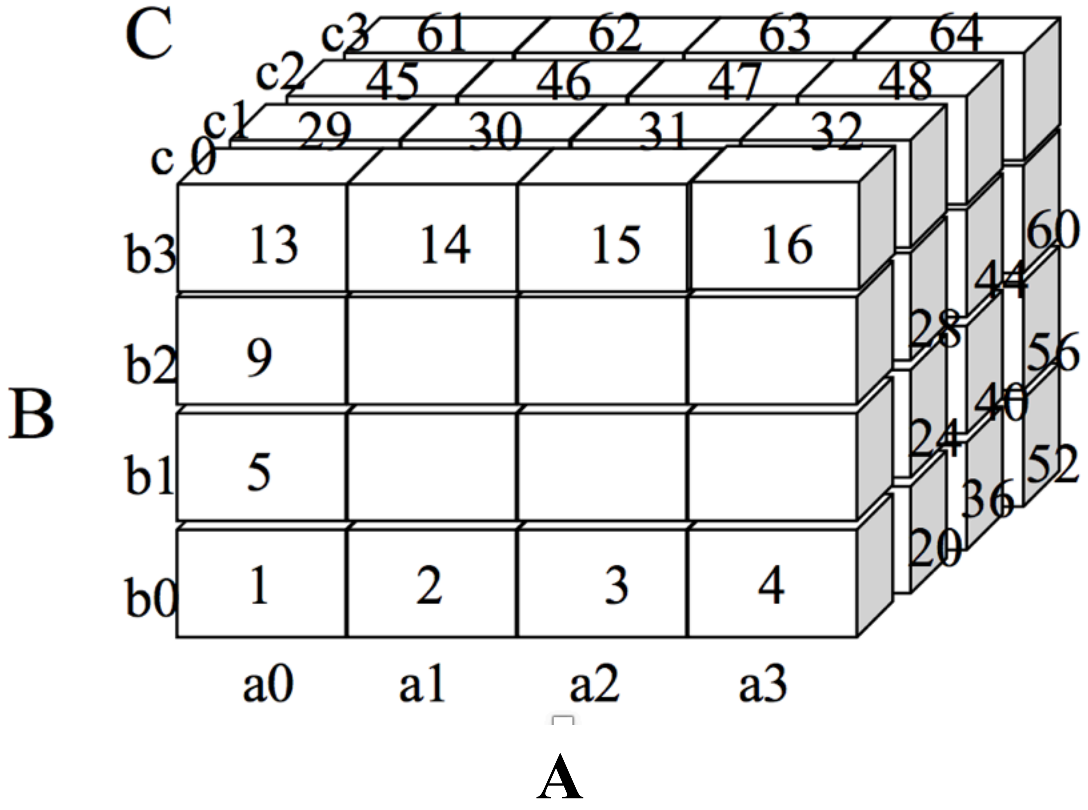
A

The Single-Pass Multi-Way Array Cubing Algorithm-con’t

Example:

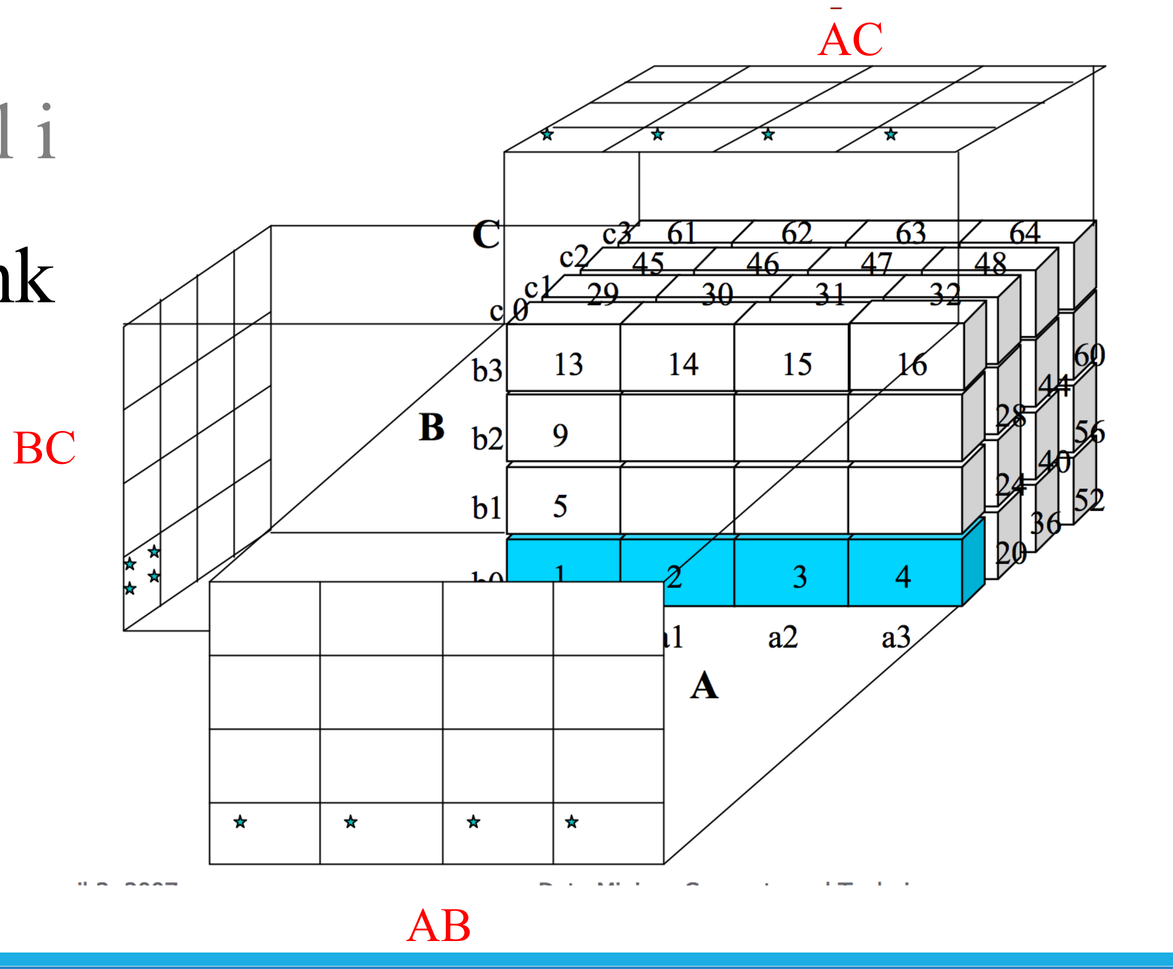
Read 3D array in dimension order ABC, from chunk 1 to 64.

- |Ci| = 4, |Di| = 16 for all i



The Single-Pass Multi-Way Array Cubing Algorithm-con't

- |Ci| = 4, |Di| = 16 for all i
- For BC, allocate 1 chunk memory -> (4*4)

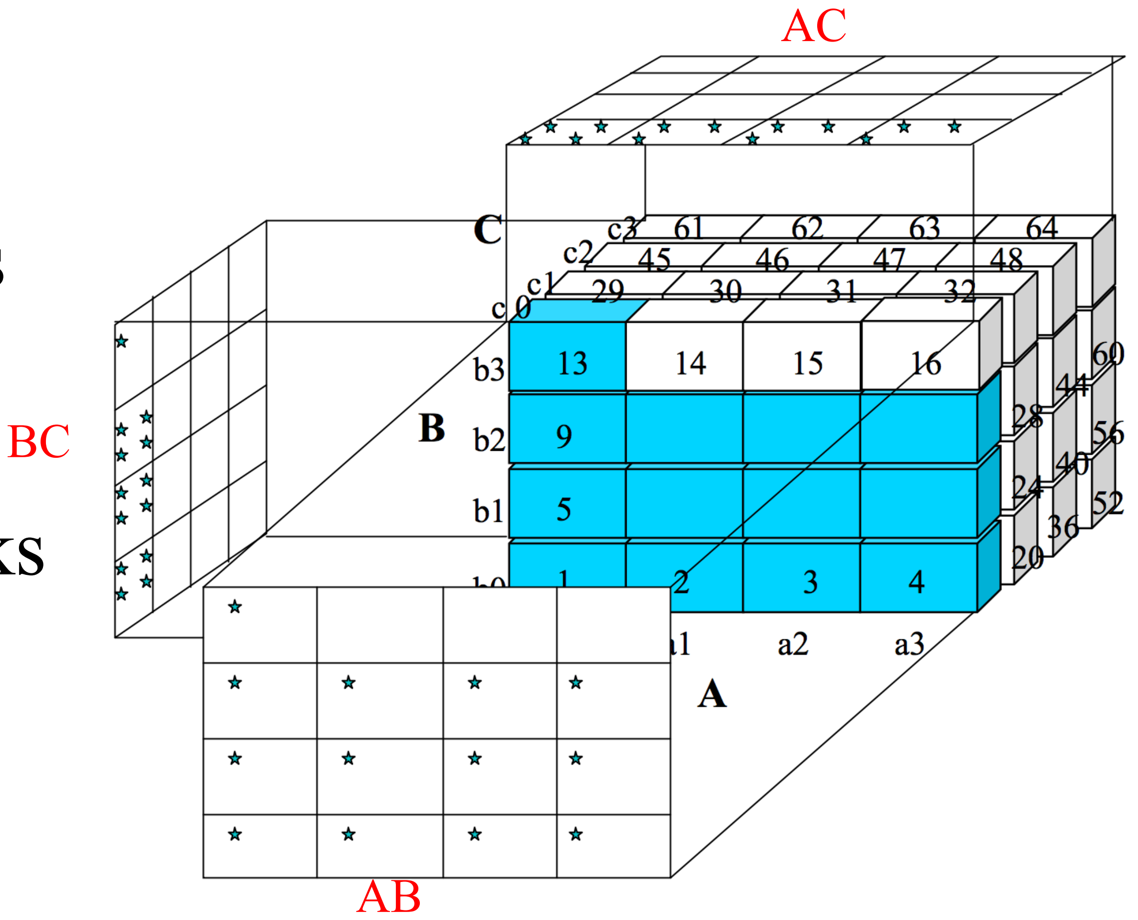


The Single-Pass Multi-Way Array Cubing Algorithm-con't

- |Ci| = 4, |Di| = 16 for all i
- For AC, allocate 4 chunks

  memory -> (16*4)
- For AB, allocate 16 chunks

  memory -> (16*16)

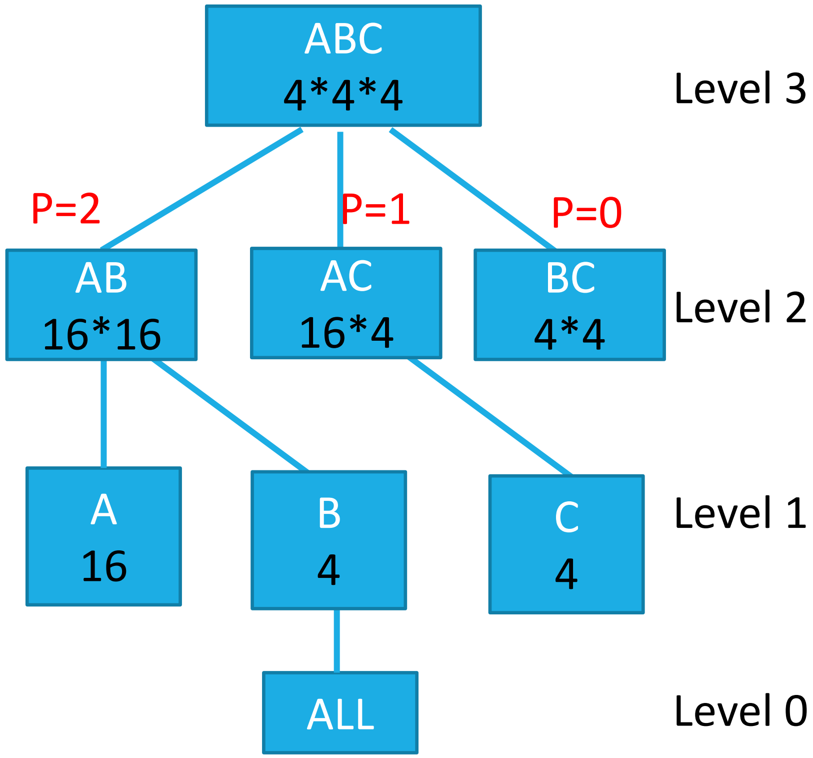


Minimum Memory Spanning Trees

P = size of the largest common prefix between the current group-by (size n-1) and its parent.

Rule 1:

$$\Pi_{i=1}^{p}|D_i| * \Pi_{i=p+1}^{n-1}|C_i|$$



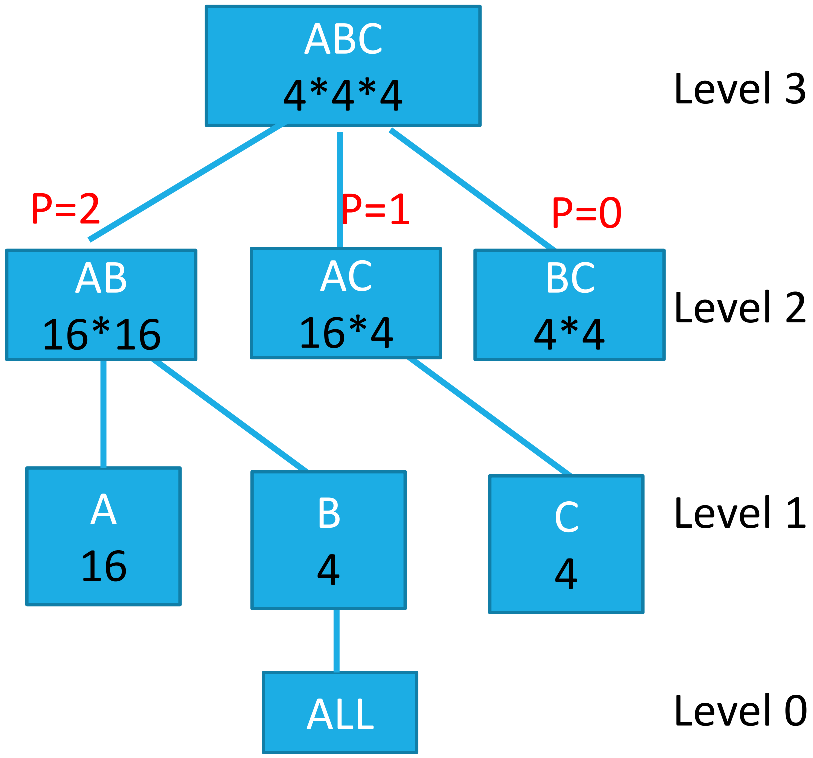
3-D array MMST in dimension order (A ,B, C)

Minimum Memory Spanning Trees

Rule 1:

$$\Pi_{i=1}^{p}|D_i| * \Pi_{i=p+1}^{n-1}|C_i|$$

- AB = |A| * |B| = 16 * 16
- AC = |A| * |Chunk| = 16 * 4
- BC = |Chunk| * |Chunk| = 4 * 4



3-D array MMST in dimension order (A ,B, C)

Effects of Dimension Order

Size of dimensions A, B , C and D are 10, 100, 1,000 and 10,000

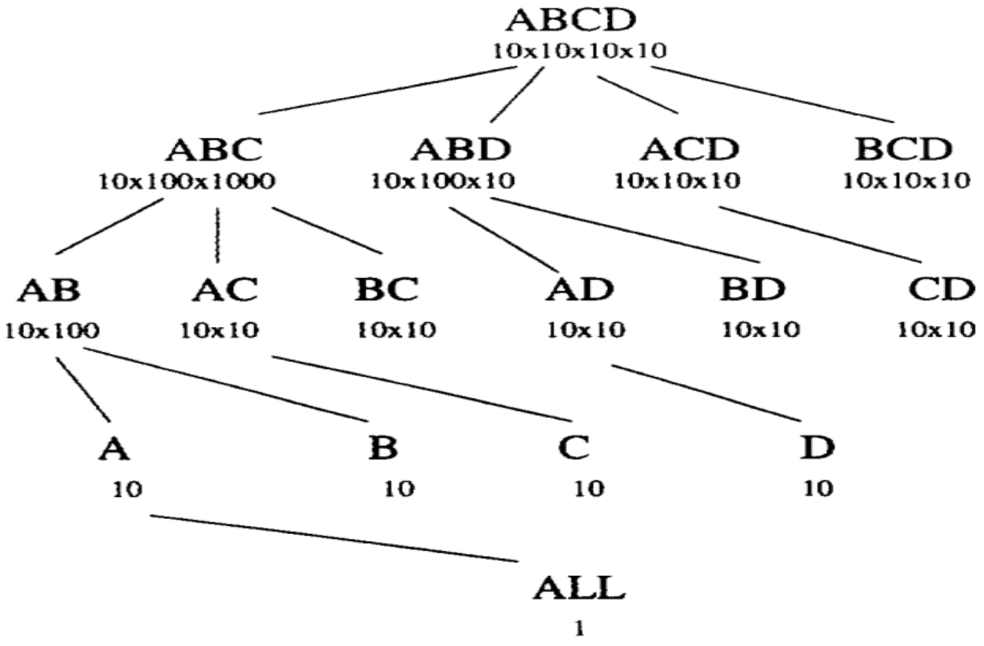


Figure 3: MMST for Dimension Order ABCD (Total Memory Required 4 MB)

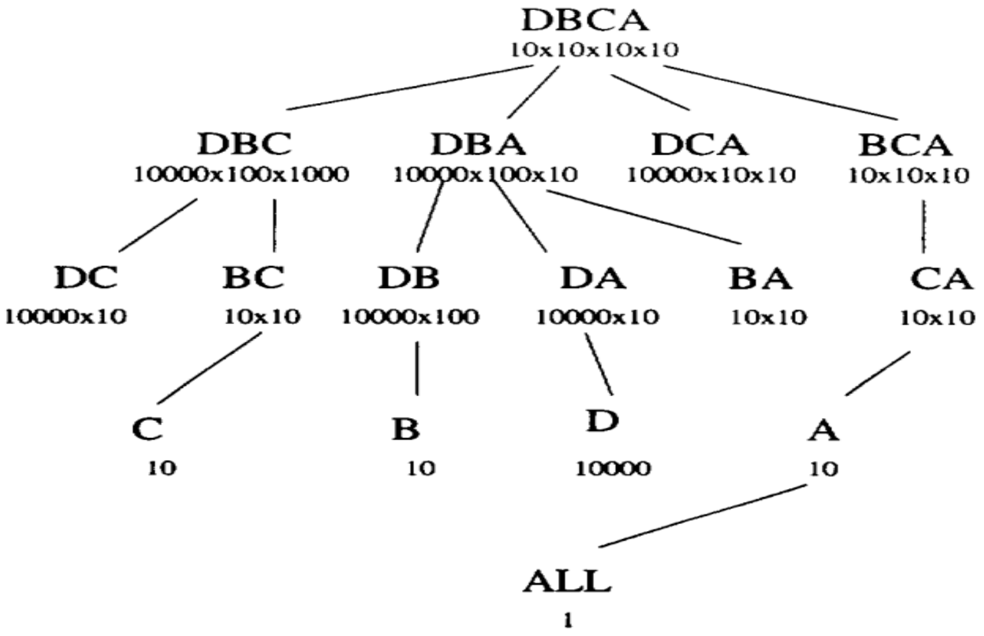


Figure 4: MMST for Dimension Order DBCA (Total Memory Required 4 GB)

Optimal Dimension Order

Theorem 1:

If we read the chunks in logical order O, where O = (D1, D2,…,Dn) and |D1| <= |D2| <= |D3| … <= |Dn|, the total amount of memory required to compute the cube of the array in one scan of A is minimum.

Performance-Naïve vs Multi-way

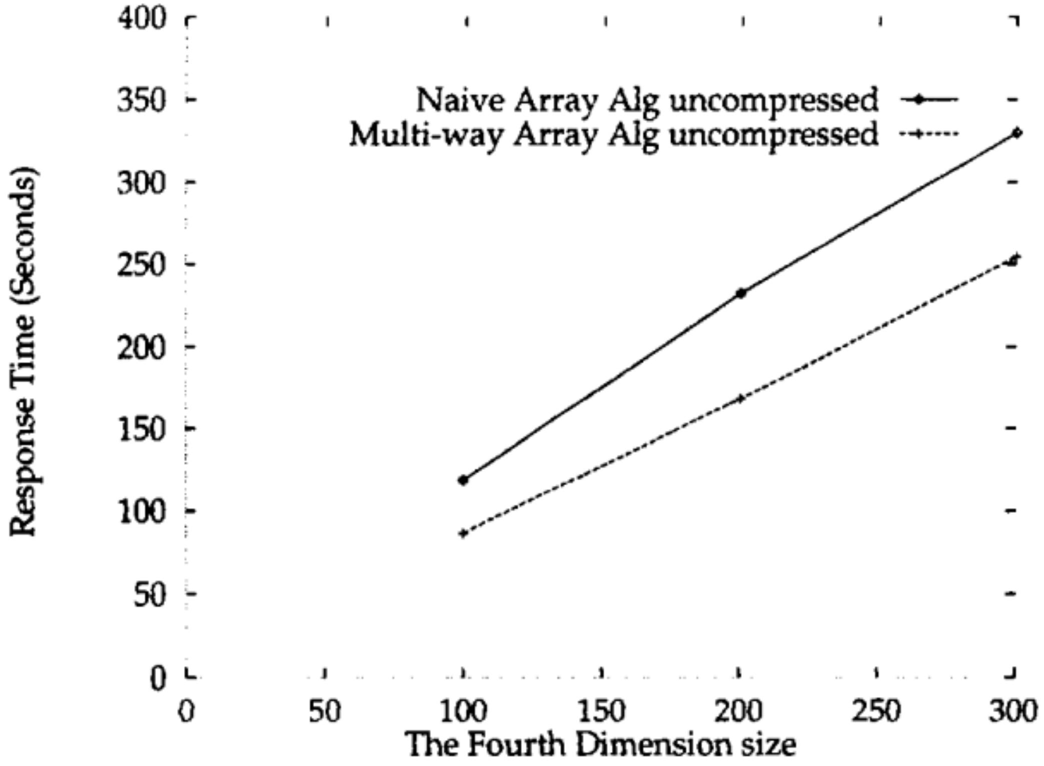
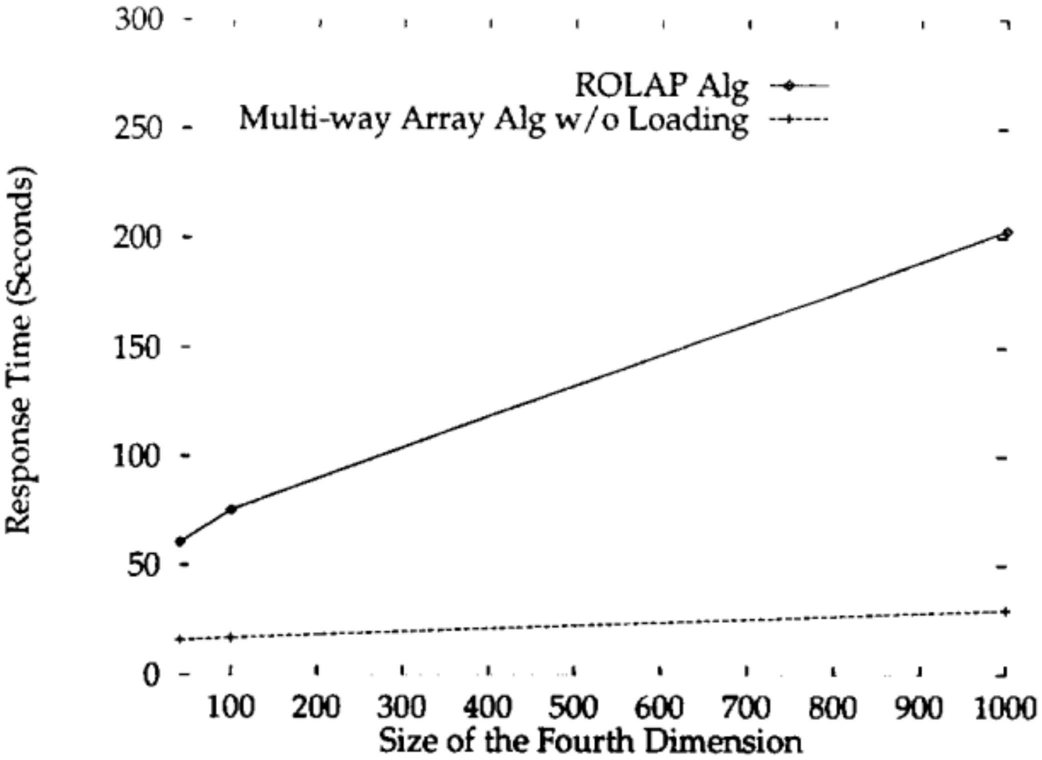
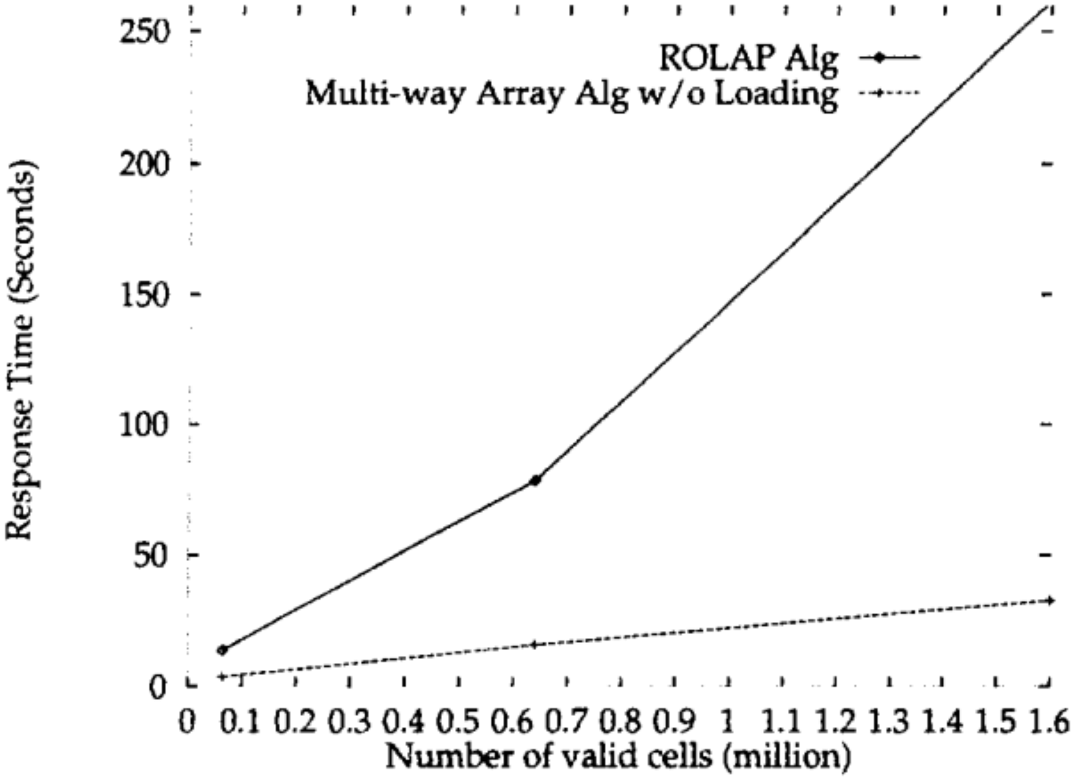


Figure 5: Naive vs. Multi-way Array Alg.

Performance-ROLAP vs MOLAP





Summary

- **The multidimensional array of MOLAP should be chunked and compressed.**

- **The Single-Pass Multi-Way Array Cubing Algorithm simultaneously updates all GROUP-BYs in the CUBE with single pass over the sparse array.**

  - Multiple passes if not enough memory.

- **Dimension Order is essential.**

- **A MMST can give a plan for computing the CUBE.**

Questions?