An Array-Based Algorithm for Simultaneous Multidimensional Aggregates

Pushkar Sinha

University	Nobel Prize Field	Number of Nobel prizes
SFU	Ph	2
UoT	Ch	1
UVic	Me	2
SFU	Li	2
UoT	Li	1
UoT	Me	3
McGill	Ph	4

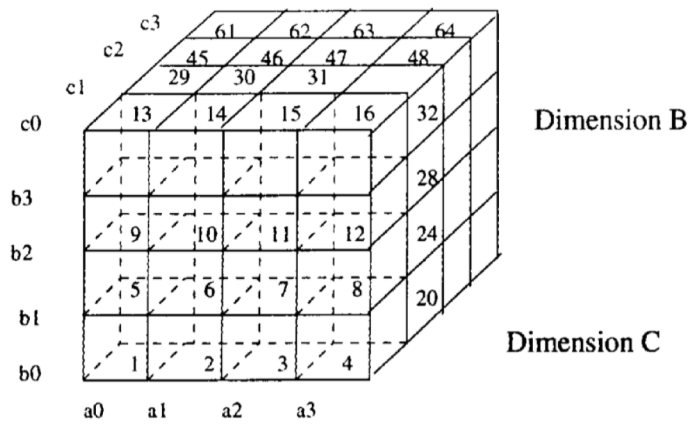

Figure 1: 3 D array

Cube Operator and Multidimensional Array

No of relational rows = 8, No of DQ rows = 4X4X4

Different Group-by and their Independence

- A chunk is 1 of the small 64 cubes in the pic. Total cube is 16X16X16 while a chuck is 4X4X4
- For group by AB we aggregates each cell of the front face for the whole C dimension and store it in the disk.
- Since these AB cells are independent we need to store on one AB chunk at a time.

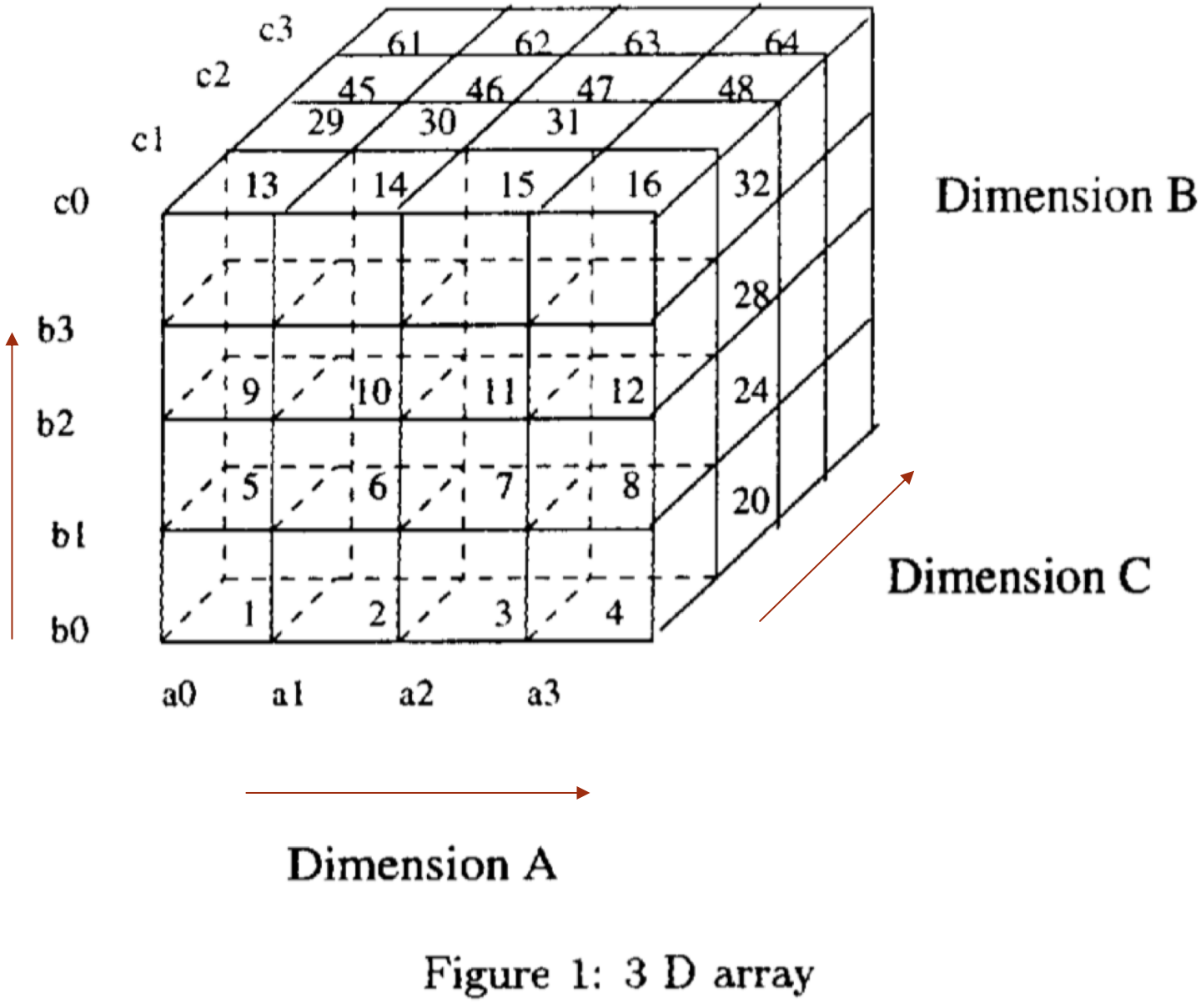

Figure 1: 3 D array

Reading order and the Key concept

- We want to read as many **less** times as possible (if enough memory, we read only once).
- Reading in dimensional order (A,B,C) mean reading the chunks linearly 1–64. You see that its group by BC (aggregating BC cells towards dimension A). For which we need only one chunk i.e. $a_0b_0 + a_1b_0 + a_2b_0 + a_3b_0$ each time to calculate each cell of BC.
- But to group by BC we need 4 chunks and similarly for group by AB we need 16 chunks.

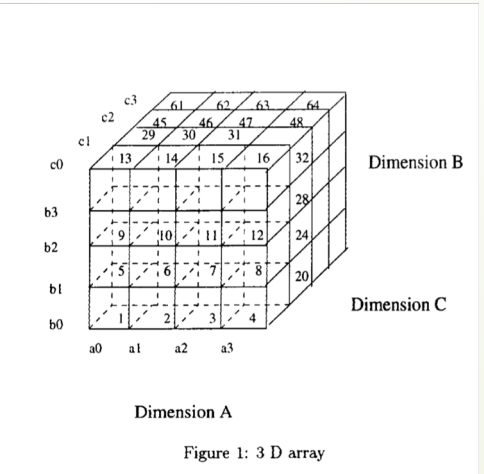

Figure 1: 3 D array

MMST

We have this tree as well as the required data structure so as we read and we can have the possible subtrees calculated, they will be such.

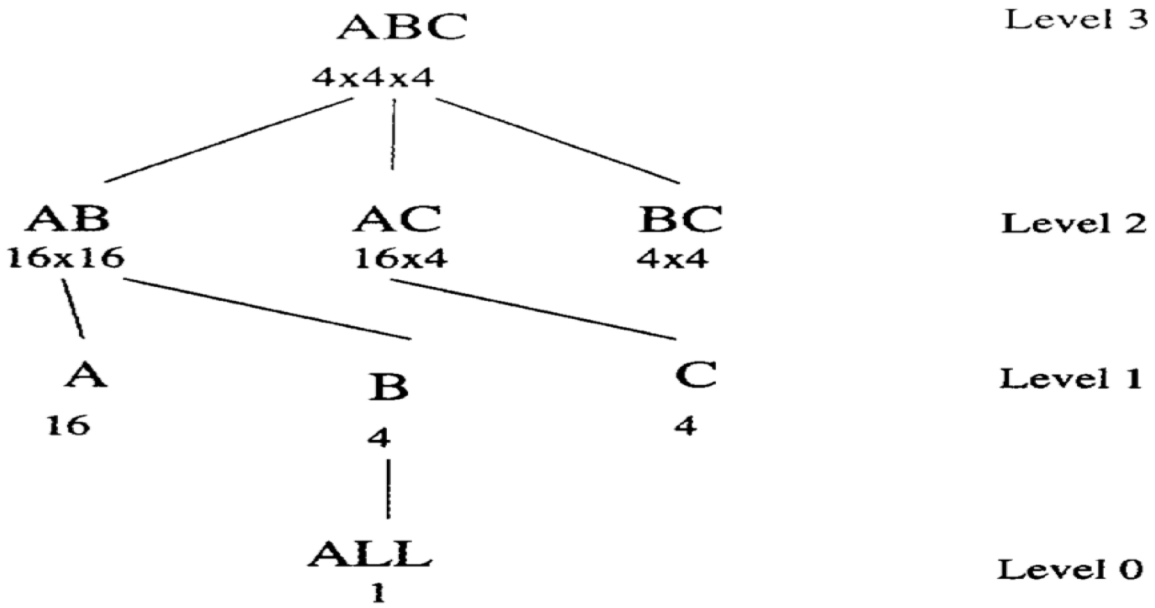

Figure 2: 3-D array MMST in dimension order (A, B, C)

Generalizing the memory requirement for n dimension

$$\prod_{i=1}^{n-1} |D_i| + (\prod_{i=1}^{n-2} |D_i|)c + (\prod_{i=1}^{n-3} |D_i|)c^2 + \cdots + c^{n-1}.$$

- We see that as the group bys get more sqewed corresponding to how we read the chunks in the first place the memory requirement increases (Right-to-left).

$$\prod_{i=1}^{n-2} |D_i| + C(2,1)(\prod_{i=1}^{n-3} |D_i|)c + C(3,2)(\prod_{i=1}^{n-4} |D_i|)c^2 + \cdots + C(n-1,n-2)c^{n-2}.$$

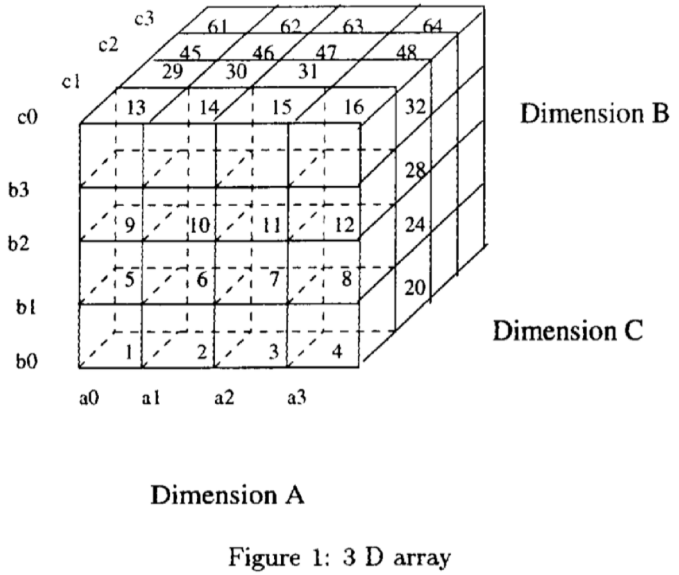

Figure 1: 3 D array

Reading order comparison

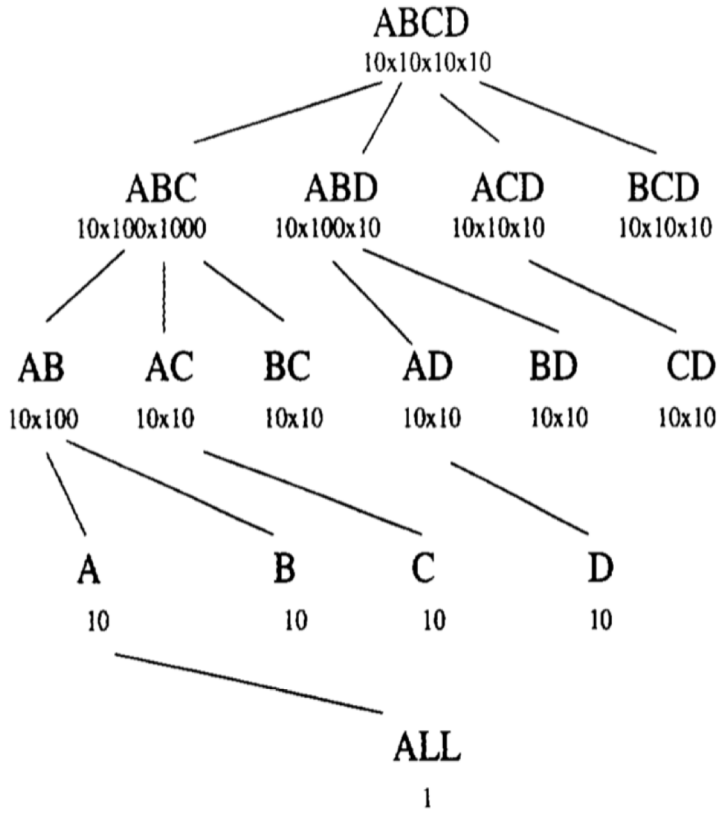

Figure 3: MMST for Dimension Order ABCD (Total Memory Required 4 MB)

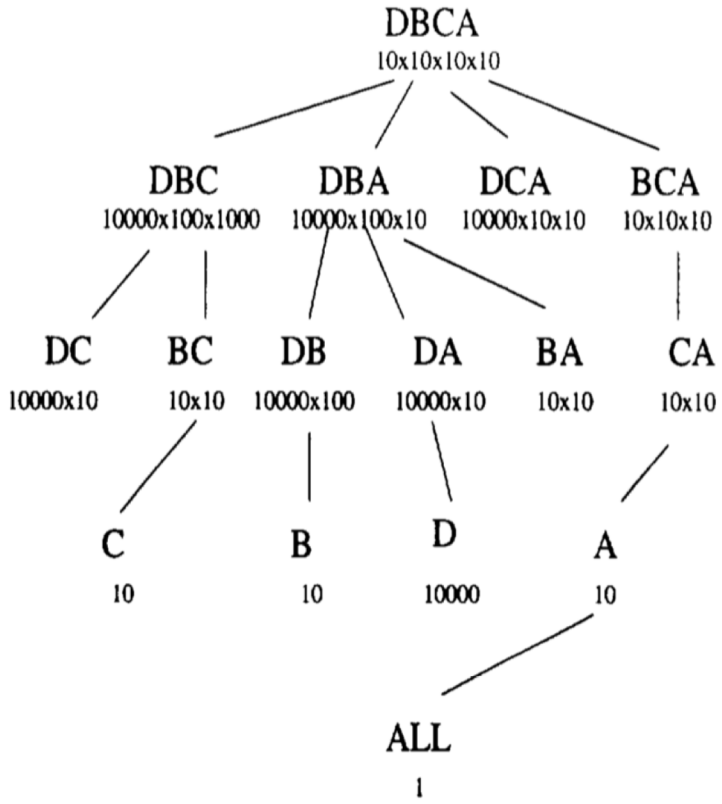

Figure 4: MMST for Dimension Order DBCA (Total Memory Required 4 GB)

When we don't have enough memory for the whole MMST

```
(1)   Create the MMST T for a dimension order O
(2)   Add T to the Tobecomputed list.
(3)   For each tree T' in Tobecomputed list
      {
(3.1)      Create the working subtree W and
                incomplete subtrees Is
(3.2)      Allocate memory to the subtrees
(3.3)      Scan the array chunk of the root of T'
                in the order O
           {
(3.3.1)         aggregate each chunk to the groupbys
                     in W
(3.3.2)         generate intermediate results for Is
(3.3.3)         write complete chunks of W to disk
(3.3.4)         write intermediate results to the
                     partitions of Is
           }
(3.4)      For each I
           {
(3.4.1)         generate the chunks from the
                     partitions of I
(3.4.2)         write the completed chunks of I to disk
(3.4.3)         Add I to Tobecomputed
           }
      }
```