**Objective →**
**Sparse Matrix- Vector Multiplication (SpMV)**
A(mXn matrix) . X(nX1 vector) = Y (mX1 vector)

**Algorithm:**

- Number of non zeros is known (nz)

**1-D Partitioning :**

- Input matrix in CRS format => three arrays: I, J, K
- Number of processes => p
  - [Process => block level => thread level]

- *Partitioning phase:*
  - Task : Multiply a non-zero element $K_e$ at ($I_e$ , $J_e$) with $X_{Je}$ and add to $Y_{Ie}$.
  - (nz) is the number of tasks to be completed.

- *Communication :*
  - We assume that matrix A and vector X are available in the shared memory of the cluster. No other communication is required between the processes for this strategy.

- *Agglomeration: Sparse Blocking*
  - Group non-zeros into blocks of size TxT. (we use, T = 256)
  - Rationale:
    - Operate on vectors X and Y with more locality => cache friendly.
    - Preprocessing => hashmap for T rows, each set of threads does lookup in it's hashmap (corresponding to the rows it processes)
    - Two levels:
      - Process **elements inside each block** *Hilbert Order* to maximise cache hits by preserving locality of input vector X.
      - Process **blocks in hilbert order.**

- *Mapping phase:*
  - Map blocks containing approximately **nz/p** elements (Tasks) to each process.
  - Process $p_i$ determines its partition [number of consecutive rows] by binary search:
    - Lower limit :- index containing i*(nz/p)th element
    - Upper limit :- index containing (i+1)*(nz/p)th element
  - Threads work on consecutive elements in the predetermined hilbert order.

- *Post-processing:*
  - Each process communicates its computed portion of output vector to the master process, which concatenates these.

**Approach 2: 2-dimensional Partitioning**

1. Model the input matrix (A) as a hypergraph.
2. Recursive, adaptive, bipartitioning
   - **Aim –**
     to obtain a p - way partition of the hypergraph ( if there are p processors in the system )
   - At each step of partitioning, divide into 2 partitions, either row-wise or column wise depending on whichever partition leads to lesser increase in communication volume.
3. Metric for partitioning – load imbalance
4. Input, output vectors are also partitioned.
5. Broadcasted list – storing mapping of partition number against each cell of x and y, for fetching x and storing y.
6. Steps:
   1. Fetch the part of x vector not local.
   2. Compute the assigned part of output vector.
   3. Send the calculated y to appropriate processor,