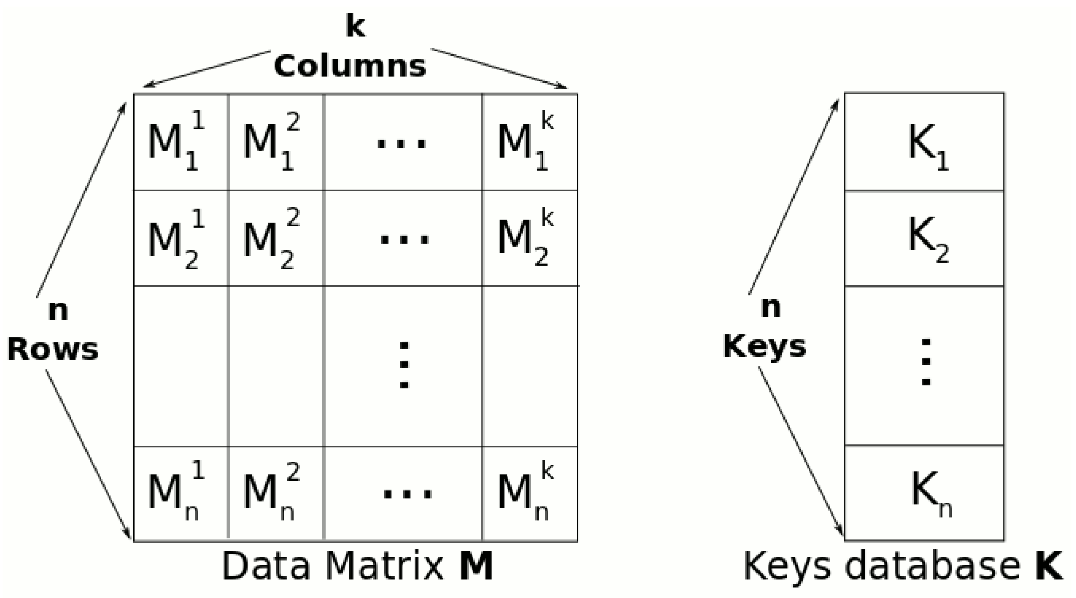
CITS3402 High Performance Computing Project 1

Project 1: Parallelization of column-wise matrix collisions

- This project carries 25% of the total marks for this unit for students enrolled in CITS3402 and CITS5507. It carries 10% of the total marks for students enrolled in SHPC4002.
- The project can be done either individually, or in a group of two.
- The submission deadline is 11:59 pm on October 5, through cssubmit.
- This project has to be implemented in OpenMP.
- You should submit C or Fortran code (well commented), a small document (in pdf) detailing the experiments you have done to improve speed-up and the best speed-up that you have obtained over your sequential code.
- You should submit through cssubmit by the deadline.

Problem Description:

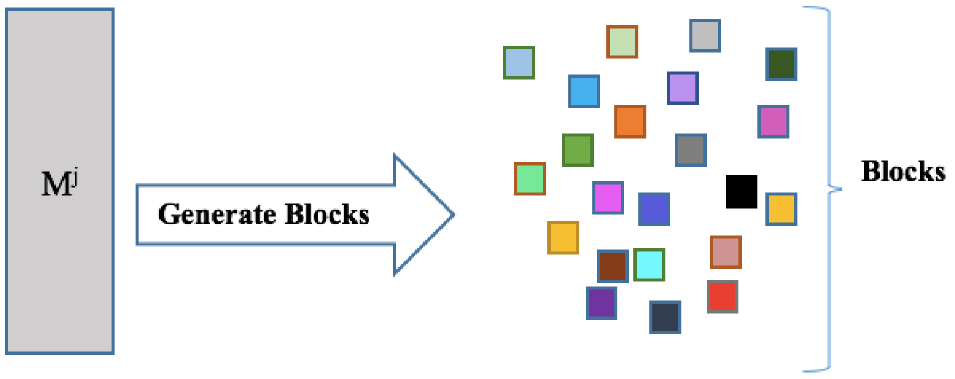
- **M** is a data matrix of *n* rows and *k* columns where, M_i represents i^{th} row-vector and M^j represents j^{th} column-vector as the figure below. Each row-vector is of size *k* and each column-vector is of size *n*. M_i^j represents a data element from row *i* and column *j*. The value of each data element M_i^j lies between 0 and 1. Assume that there is no missing value in this *n* x *k* matrix **M**.



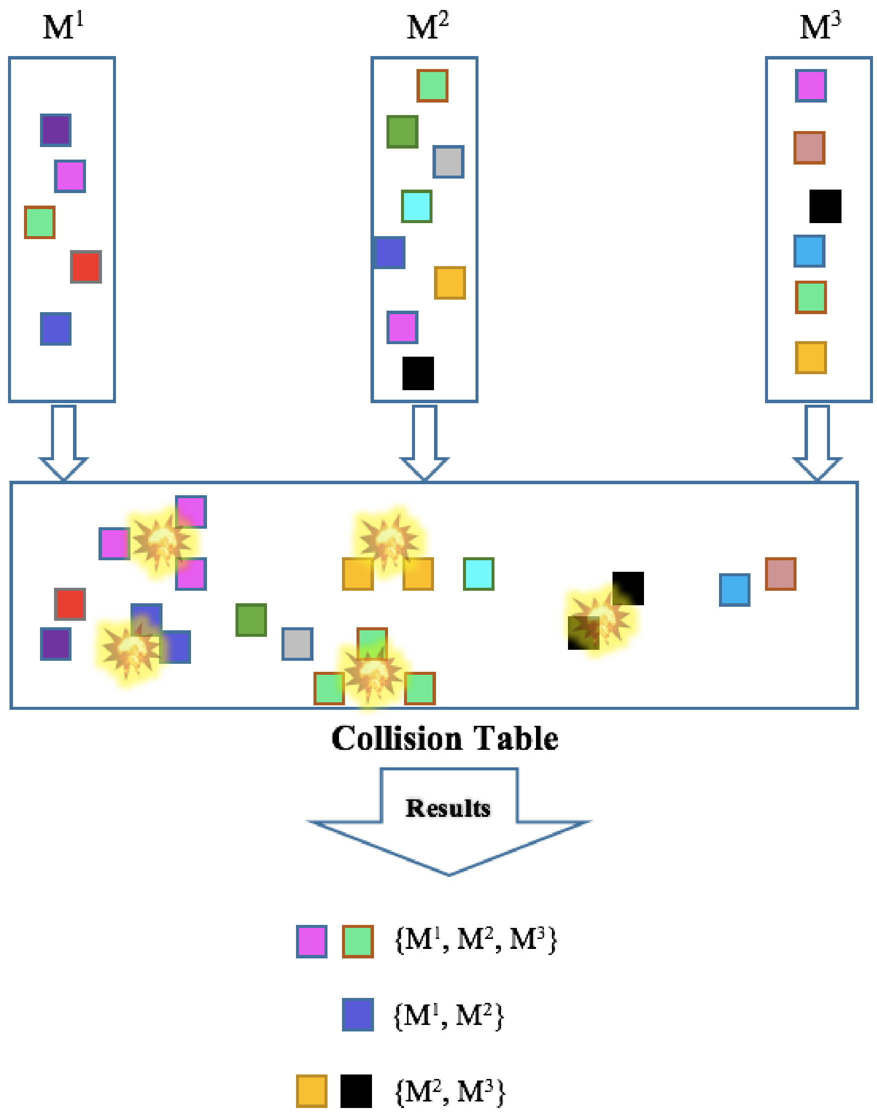
- **K** is a vector of random and unique *n* large integers called *keys*. Each row-ID *i* ($1 \le i \le n$) can be mapped to a separate key K_i from **K** which is also called *keys database*. In other words, you can choose, or associate a unique *key* from **K** for each row of **M**. No more than one row should be associated with a key in **K**, and no more than one key in **K** should be associated with a row in **M**. You can randomly choose a key for every row in **M**.

- You will break down each column M^j into a number of blocks (as in the figure below). Each column will/can result in different number of blocks. The process of generating blocks from a column is described later in this project sheet. Suppose there are

4 elements in a block (we are taking the number 4 arbitrarily here, this can vary, as explained later). All these elements will come from the same column of **M**. Let us call these 4 elements as M_u^j, M_v^j, M_w^j, and M_x^j. Also, these four elements are from the four rows *u, v, w, x* of the matrix *M*. Suppose the four keys that you have chosen for these four rows are K_u, K_v, K_w, K_x. The sum K_u+K_v+K_w +K_x is called the *signature* of this block.



- Eventually, we want to match these blocks by matching their signatures across columns to see which blocks from which columns are the same. We consider two blocks as same if their signatures match. For example, in a 3-column example below, two different blocks from columns M^1, M^2 and M^3 match with each other and appear in the final result. Also, we notice that columns M^1 and M^2 have one block in common, and columns M^2 and M^3 have two blocks in common. Let us say that when two blocks from different columns match with each other, a collision happens. We want to detect all such collisions and produce the results as fast as possible. The aim of this project is to speed-up the matching process through parallelization.



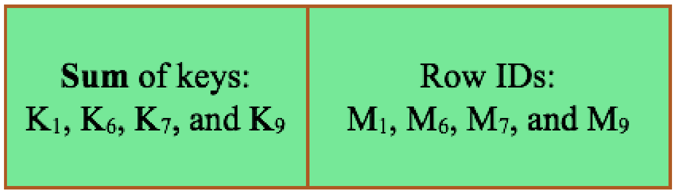
Generating blocks:

- Each block is generated from a unique combination of 4 neighbouring data elements from the same column. Two data elements M_x^j and M_y^j from a column M^j are within the same neighbourhood if $|M_x^j - M_y^j|$ < **dia**, where **dia** is a distance measure whose value lie between 0 and 1. Note that a block cannot have elements from different columns, all should be from the same column.
- In each column, you would need to find all possible combinations of four elements which are within **dia** distance of each other. Each such combination will be converted into a block.
  One way is to first find all **dia**-neighbourhoods in a column and then create 4-element combinations out of it. For example, if in a column M^j, we find that five elements M_1^j, M_6^j, M_7^j, M_9^j and M_{12}^j are within same **dia**-neighbourhood then following distinct combinations are possible:

  1. $\{M_1^j\ M_6^j, M_7^j, M_9^j\}$
  2. $\{M_1^j\ M_6^j, M_7^j, M_{12}^j\}$
  3. $\{M_1^j\ M_6^j, M_9^j, M_{12}^j\}$
  4. $\{M_1^j\ M_7^j, M_9^j, M_{12}^j\}$
  5. $\{M_6^j\ M_7^j, M_9^j, M_{12}^j\}$

However, a data element can belong to multiple neighbourhoods and therefore, redundant combinations can be generated from overlapping **dia**-neighbourhoods. Avoid generating two duplicate combinations with all same elements. The combinations can be only partially overlapping or totally disjoint with each other. You can think of other ways to generate such combinations from the columns.

- As explained above, each block actually represents a unique sum called *signature* which is calculated by adding the mapped keys of each data element from the block. In the example above, a block corresponding to the first combination: $\{M_1^j, M_6^j, M_7^j, M_9^j\}$ will contain the sum of 4 mapped keys: $K_1 + K_6 + K_7 + K_9$ (as the Figure below). Each rowID is mapped to a separate key.



Sum of keys: K_1, K_6, K_7, and K_9	Row IDs: M_1, M_6, M_7, and M_9

Typical information in a Block

- To match two blocks from different columns, we need to match their signatures. Collision occurs when two blocks with the same sum or signature value exists in different columns. As the same block can exist in 2 or more columns, we do not have prior information about these column ids. The only way is to generate blocks, find their sum and see if the same sum exists in other columns and which columns. You can either generate one block and then match, OR generate a couple of blocks, then do the matching, OR generate all possible blocks in all columns first and then do the matching. You will need to figure out the effective way of doing it in parallel.

- So the task is to output those blocks which exist in more than 1 column such that they represent same sum. And the task also involves identifying the relative columns for each such block. As a post-processing step, you can merge partially overlapping blocks which were found under the same subset of columns. For example, if blocks $\{M_1, M_2, M_3, M_4\}$ and $\{M_1, M_2, M_8, M_9\}$ were detected to be present in columns M^1, M^5 and M^7 and neither of these blocks exists in any other column, then they can be merged into $\{M_1, M_2, M_3, M_4, M_8, M_9\}$. Only in this last post processing step, a block is allowed to have more than 4 elements. Such merged blocks in their relevant subsets of columns should ideally form the final result.

Data and Input Parameters:

You should download the dataset `data.txt` and the keys keys.txt. The `data.txt` file contains 4400 x 500 matrix of real numbers between 0 and 1. There are 4400 keys in the `keys.txt` file and each key is 14-digit long. Remember, we are interested to know which rowIDs group together in a set of 4 (blocks) and under which subsets of columns. **dia** = 1.0e-6 which is 0.000001.

Amitava Datta and Amardeep Kaur
September 2016

School of Computer Science & Software Engineering
The University of Western Australia
Crawley, Western Australia, 6009.
Phone: +61 8 9380 2716 - Fax: +61 8 9380 1089.
CRICOS provider code 00126G