# MatMult

## Multiplication of two matrices with MapReduce

In examples package of Hama, we have introduced two MapReduce-based approaches and illustrated them on examples.

### Iterative Approach

The iterative approach is simple and naive. Initially, each map task receives a row index of B as a key, and the column vector of the row as a value. Then, it multiplies all columns of i-th row of A with the received column vector. Finally, a reduce task collects the i-th product into the result matrix.

```
For i = 0 step 1 until N -1
  Job: Computes the ith row of C = Matrix-Vector multiplication
```

### Block Approach

To multiply two dense matrices A and B, we should build the "collectionTable" in the pre-processing phase of MapReduce. The collectionTable is an 1-D representation, transformed from the original 2-D representation of two matrices. Each row of the collectionTable has two submatrices of A(i,k) and B(k,j) with the row index of (n2  i) + (n  j) + k, where n denotes the row size of matrix A and B. We call these submatrices a block. Each map task walks only on the collectionTable instead of the original matrices, and thus it significantly reduces required data movement over the network. The following code shows the block algorithm after pre-processing. Each map task receives a blockID as a key, and two submatrices of A and B as its value, and then multiplies two submatrices, A[i][j]  B[j][k]. Afterward, a reduce task computes the summation of blocks, s[i][k]+ = multipliedblocks.

```
Blocking jobs:

Collect the blocks to 'collectionTable' from A and B.

- A map task receives a row n as a key, and vector of each row as its value
   - emit (blockID, sub-vector) pairs
- Reduce task merges block structures based on the information of blockID

Multiplication job:

- A map task receives a blockID n as a key, and two sub-matrices of A and B as its value
   - Multiply two sub-matrices: A[i][j] * B[j][k]
- Reduce task computes sum of blocks
   - s[i][k] += multiplied blocks
```