Report

令 Input matrix M 為 m*n，N 為 n*p
　Output matrix O 為 m*p

(input size 的 m, n, p 必須在程式的第 28, 77, 29 行做更改)

Map

假設讀到的輸入為”M, i, j, M_ij”，則製造出的 Key-Value 為(i, 0)- [M, j, M_ij]、(i, 1)- [M, j, M_ij]、(i, 2)- [M, j, M_ij]、…、(i, p)-[M, j, M_ij]；若讀入的輸入為”N, j, k, N_jk” 則作與上述相對應的處理，由此得到所有的 Key-Value。

```
    {
        int m = 500;
        int p = 500;
        String line = value.toString();
        String[] words = line.split(",");
        Text outputKey = new Text();
        Text outputValue = new Text();
        if (words[0].equals("M"))
        {
            for (int k = 0; k < p; k++)
            {
                outputKey.set(words[1] + "," + k);
                outputValue.set(words[0] + "," + words[2] + "," + words[3]);
                context.write(outputKey, outputValue);
            }
        }
        else {
            for (int i = 0; i < m; i++)
            {
                outputKey.set(i + "," + words[2]);
                outputValue.set("N," + words[1] + "," + words[3]);
                context.write(outputKey, outputValue);
            }
        }
```

Reduce

對 Map 後的結果將 Key 相同的 Values 分為同一組，因此對於 Key (i, k) 所對應到的 Values 為{ [M, 0, M_i0]、[M, 1, M_i1] 、…、[M, 1, M_ip] 、[N, 0, N_0k] 、[N, 1, N_1k] 、…、[N, p, N_pk] }，以”M”，”N”為區別將 Values 分到 hashM 及 hashN 中，再將每個 Value 後面的兩個值存成新的 key-value，例如：[M, 0, M_i0]就會被分到 hashM 中，其新的 key-value 為(0)-(M_i0)。將 hashM 及 hashN 中以 key 做區分，將相對應的 value 相乘後取其總和即為 O_ik 的結果，EX: O_ik = M_i0*N_0k + M_i1*N_1k + … + M_ip*N_pk。

```
61          String[] value;
62          HashMap<Integer, Integer> hashM = new HashMap<Integer, Integer>();
63          HashMap<Integer, Integer> hashN = new HashMap<Integer, Integer>();
64          for (Text val : values)
65          {
66              value = val.toString().split(",");
67              if (value[0].equals("M"))
68              {
69                  hashM.put(Integer.parseInt(value[1]), Integer.parseInt(value[2]));
70              }
71              else
72              {
73                  hashN.put(Integer.parseInt(value[1]), Integer.parseInt(value[2]));
74              }
75          }
76
77          int n = 500;
78          int sum = 0;
79          int m_ij;
80          int n_jk;
81          for (int j = 0; j < n; j++)
82          {
83              m_ij = hashM.containsKey(j) ? hashM.get(j) : 0;
84              n_jk = hashN.containsKey(j) ? hashN.get(j) : 0;
85              sum += m_ij * n_jk;
86          }
87
88          Text out = new Text(key.toString() + "," + Integer.toString(sum));
89          context.write(null, out);
```