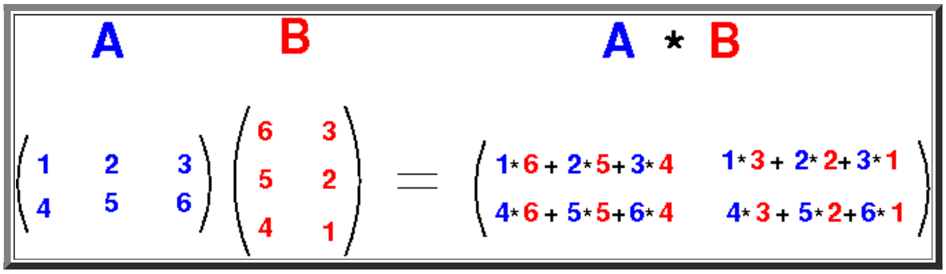
MapReduce Algorithm for Matrix Multiplication

- **Matrix Multiplication**
  - From **high school calculus**:

$$A \times B = C$$

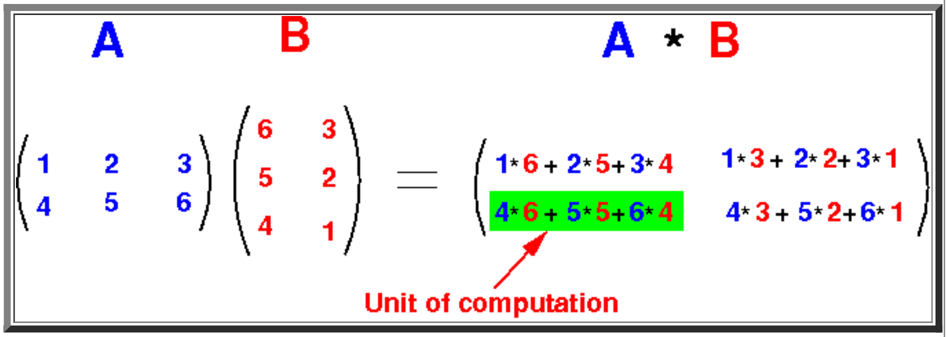
$$c_{ij} = \sum_{k=1,2,...,n} a_{ik} \times c_{kj}$$

  - **Example:**

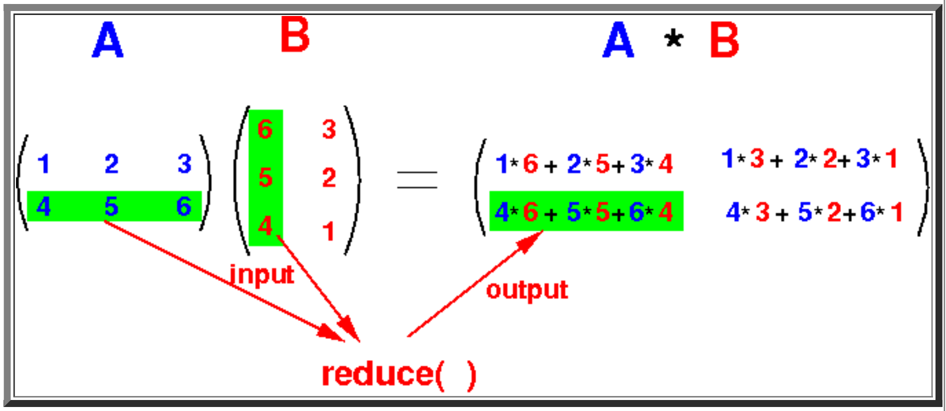


- **The reduce( ) step in the MapReduce Algorithm for matrix multiplication**
  - **Facts:**

1. The ***final* step** in the **MapReduce** algorithm is to produce the **matrix A × B**
2. The **unit** of **computation** of of **matrix A × B** is ***one* element** in the **matrix**:



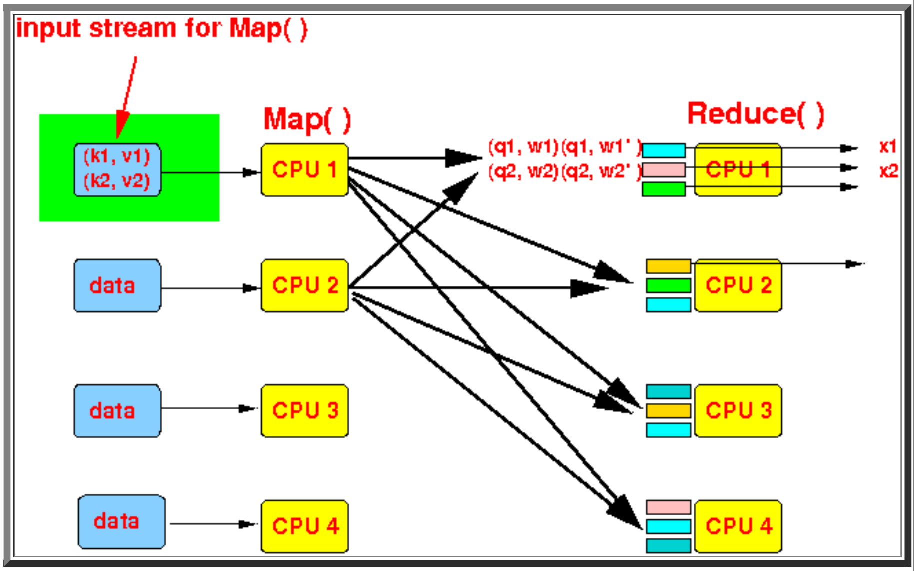
  - **Conclusion:**
    - The ***input* information** of the **reduce( )** step (function) of the **MapReduce algorithm** are:



1. **One row vector** from **matrix A**
2. **One column vector** from **matrix B**

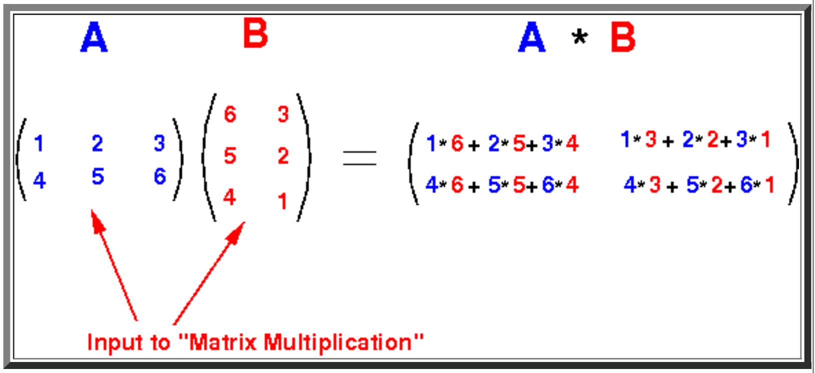
- The `reduce( )` function will **compute**:
  - The ***inner* product** of the
    - **One row vector** from **matrix A**
    - **One column vector** from **matrix B**

- **Preprocessing for the `map( )` function**
  - **Fact:**
    - The `map( )` function (really) ***only*** has ***one* input stream**:



of the **format** `( keyi , valuei )`

  - The **inputs** of the **matrix multiplication** are:
    - ***Tow* (2)** input **matrices**:

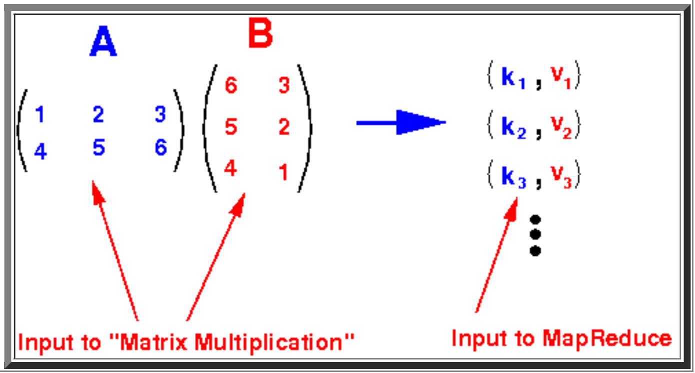


  - **Therefore:**
    - We must **insert** a ***pre-processing* step** to:

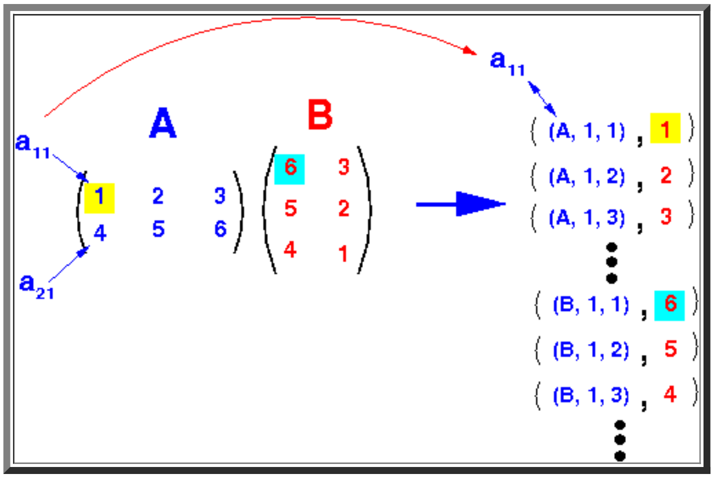
- **Convert** the **input matrices** to the **form**:

  (key_1 , $value_1$)
  (key_2 , $value_2$)
  (key_3 , $value_3$)
  ...

Graphically:



- **Pre-processing used** for **matrix multiplication**:

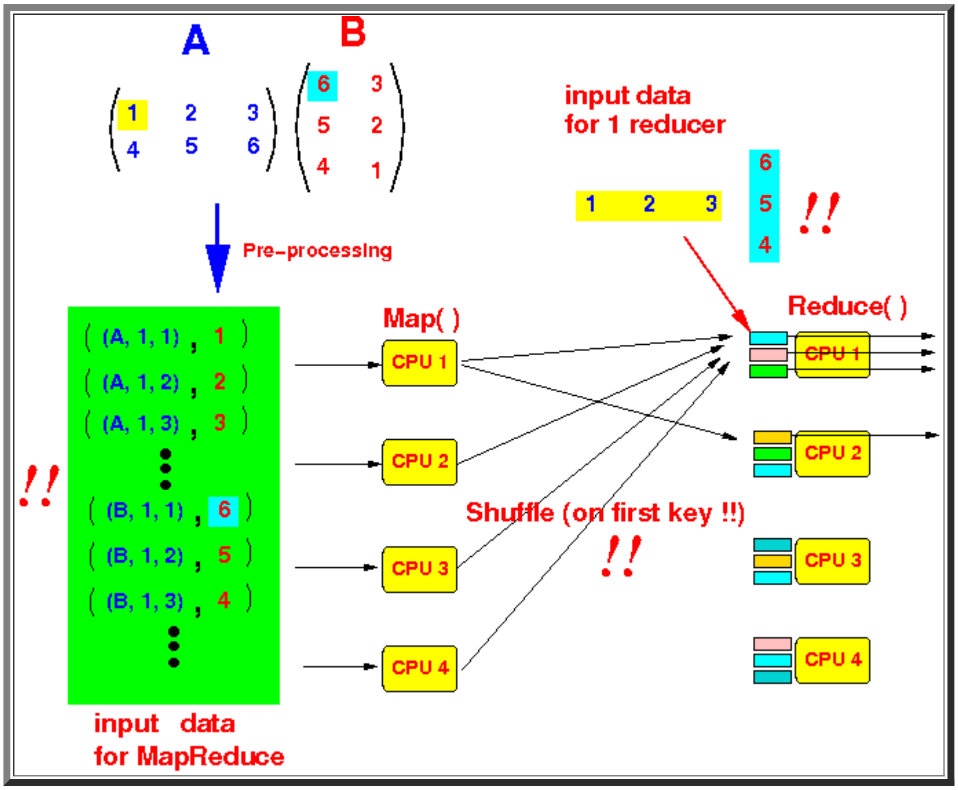


- **Overview of the MapReduce Algorithm for Matrix Multiplication**
  - **So far**, we have **discovered**:
    - The **input** to the **Map( )** is as **follows**:

      ((A, 1, 1) , a_{11})
      ((A, 1, 2) , a_{12})
      ((A, 1, 3) , a_{13})
      ...
      ((B, 1, 1) , b_{11})
      ((B, 1, 2) , b_{12})
      ((B, 1, 3) , b_{13})
      ...

    - The **input** to ***one*** **reduce( )** function is as **follows**:
      - A **row vector** from **matrix A**
      - A **column vector** from **matrix B**

- **Graphical summary:**



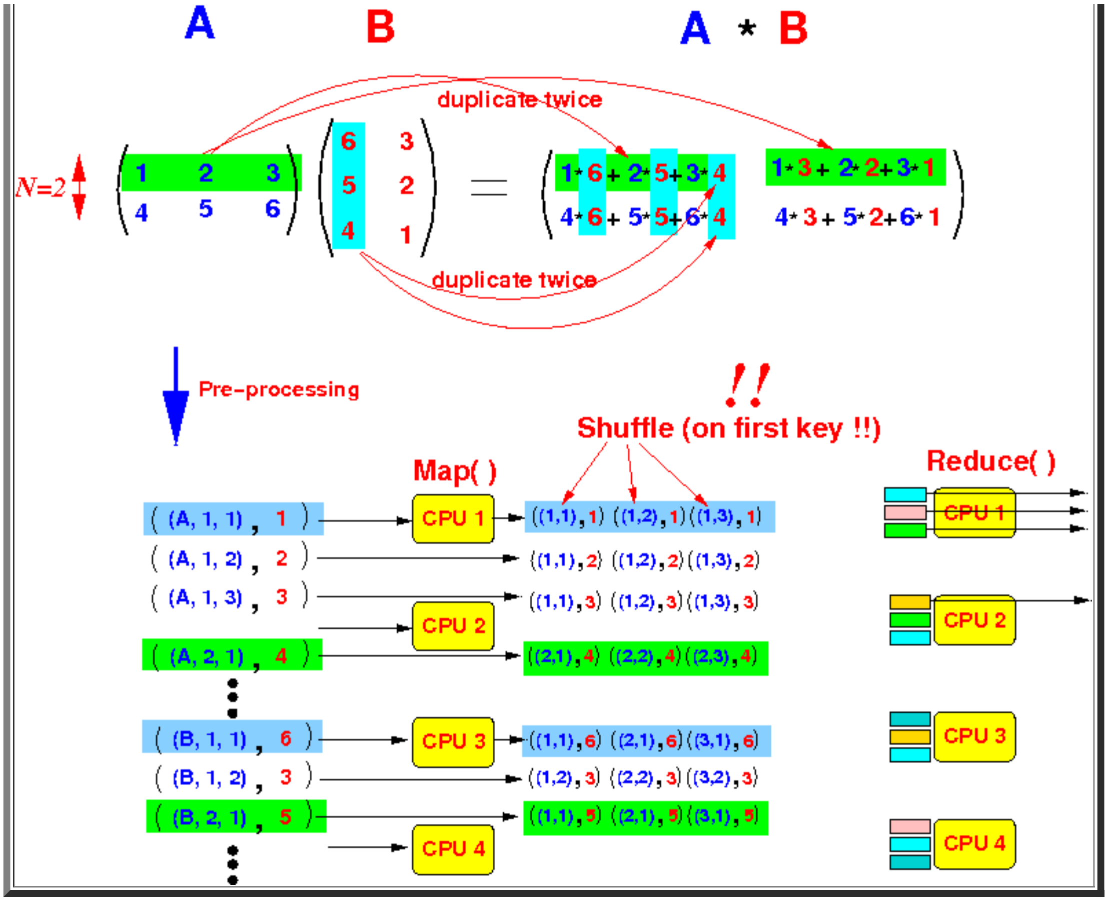
- **The MapReduce Algorithm for Matrix Multiplication**
  - The `map( )` function:

- The `map( )` will **duplicate** ***N* times** as **follows**:

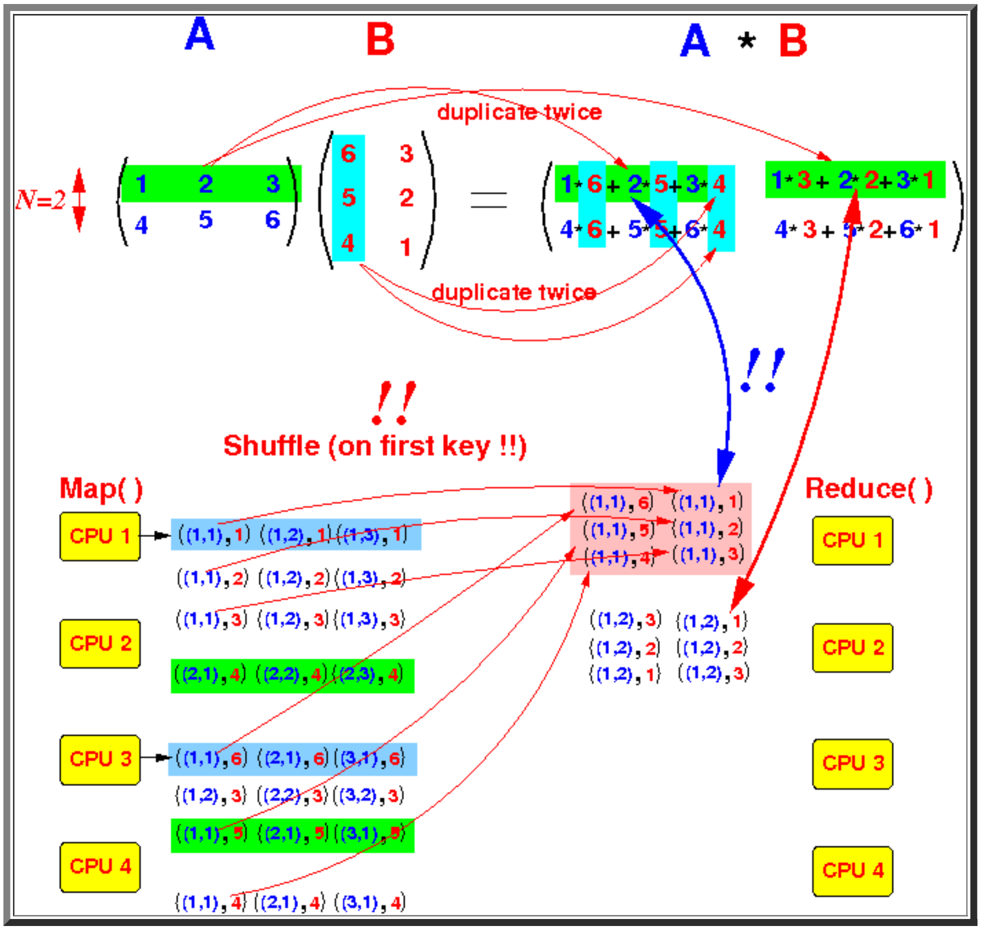
```
(A, i, j), x) --->  ( (i,1), x )  ( (i,2), x ) ....  ( (i,N), x )

(B, i, j), x) --->  ( (1,j), x )  ( (2,j), x ) ....  ( (N,j), x )
```

wher *N* = **# rows** in **matrix A** (= **# columnss** in **matrix B**)

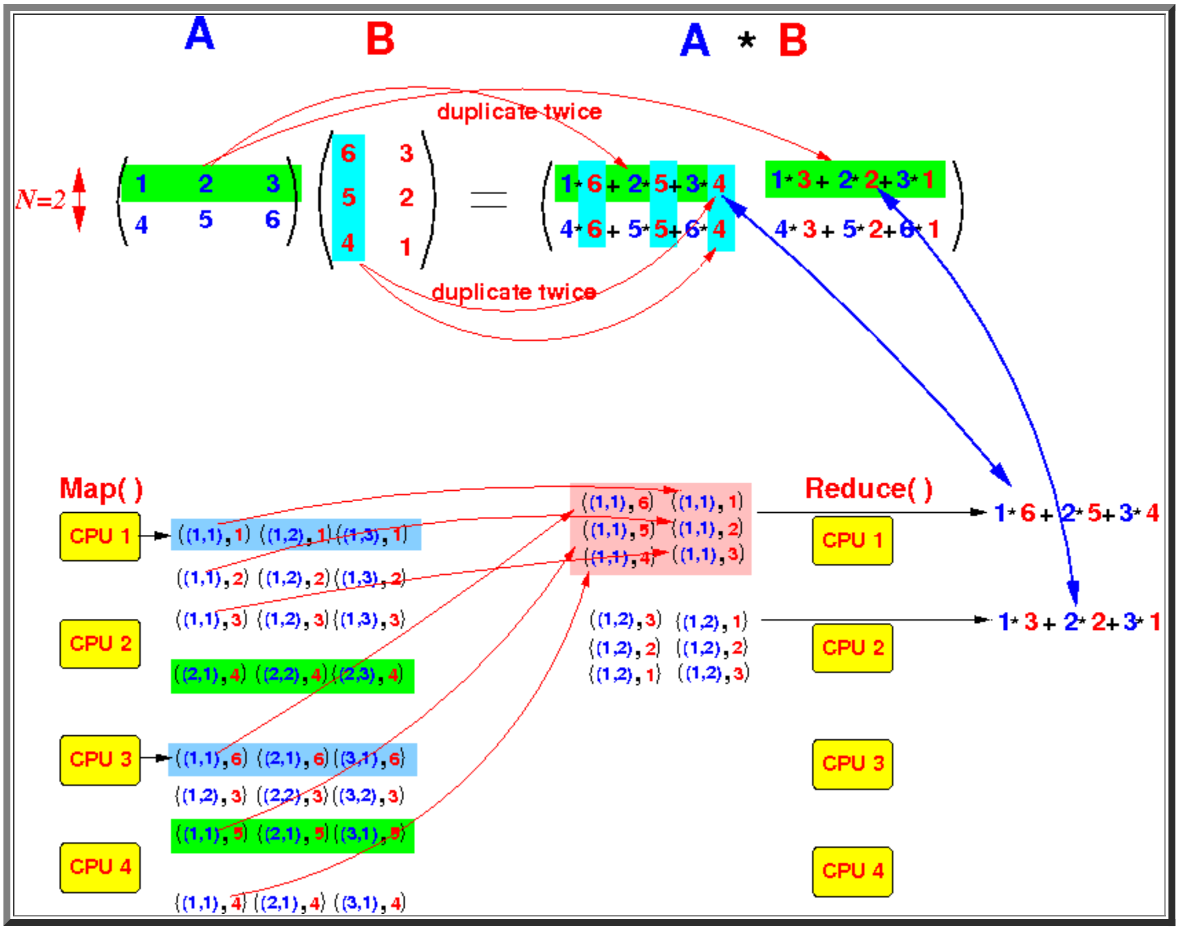
Example:



- The *shuffle* **mechanism** of **MapReduce** will *re-organize* **(group)** the `map( )` **output** as **follows**:



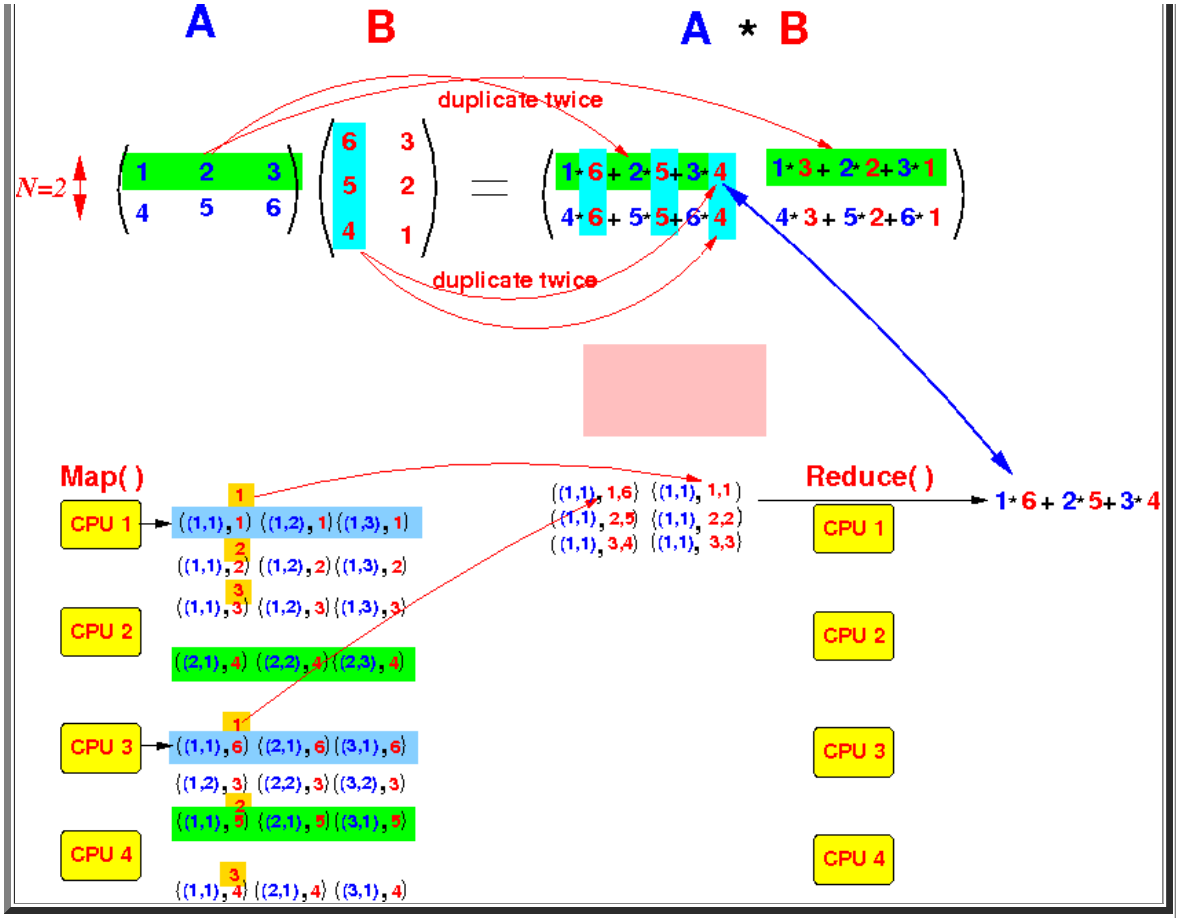
- The `reduce( )` **function** will **compute** the ***inner product*** of the **input vectors**



- **Postscript:**

  - We need to **tag** the **map( ) function output** with the **position** so the **reduce( ) function** can **identify** the ***components*** in the ***different*** **vectors**

    Example:



(This **detail** was **omitted** for ***brevity*** --- **figure** is kinda ***full***)

- The `reduce( )` function is as **follows**:

```
sum = 0;

for ( pos = 1 to N ) do
{
   x = first  value at position pos
   y = second value at position pos

   sum = sum + x*y;
}
```