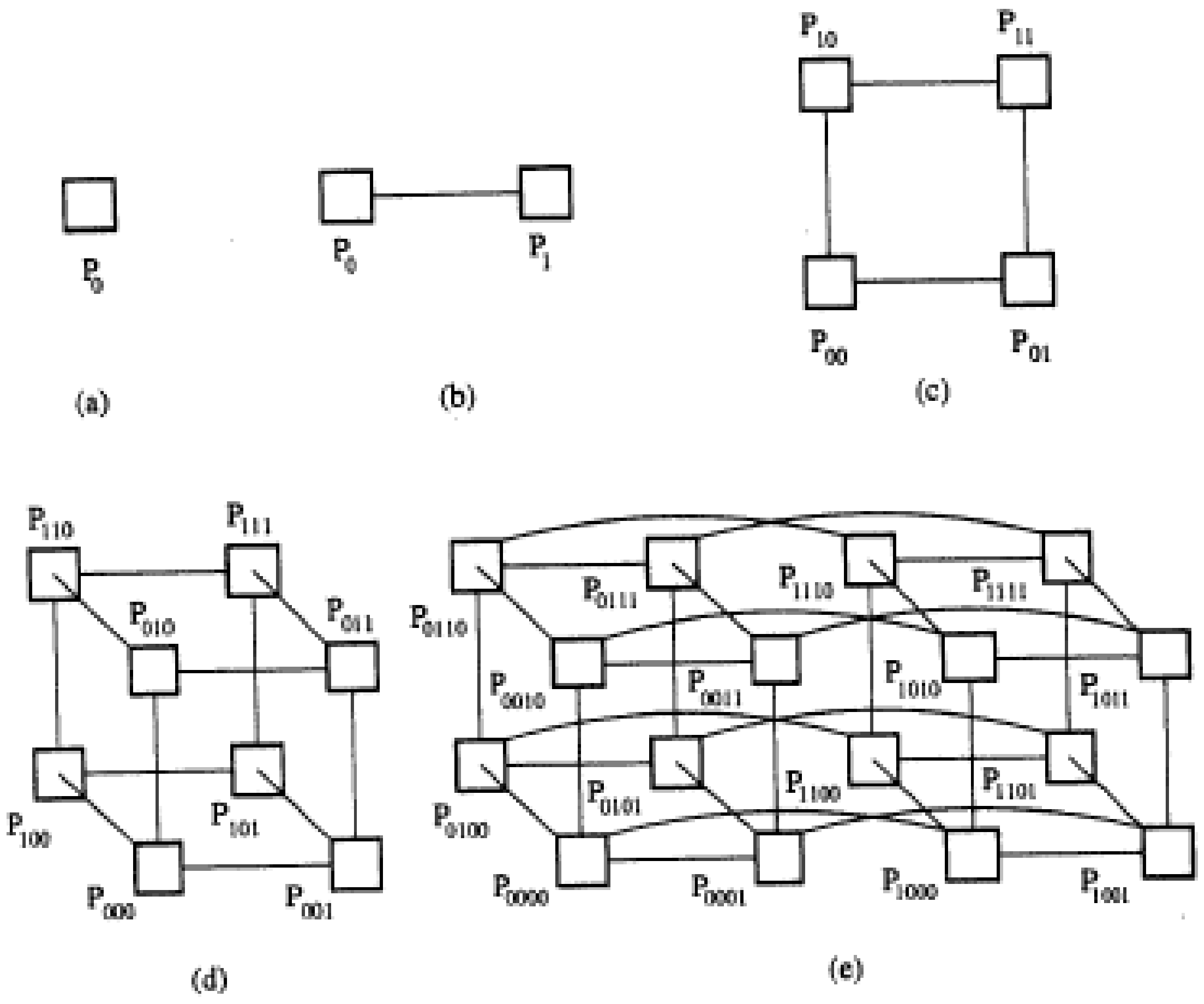
Matrix Multiplication & Transposition on Hypercube

Presented by Brett Duncan

Matrix Multiplication Applications

- Many problems can be solved using matrix multiplication or a variation of it. [1]
  - Finding the shortest distance between all pairs of vertices in a graph.
  - Transitive closure
  - Finding the radius, diameter, and centers of a graph.
  - Finding a breadth-first spanning tree
  - Topological sort

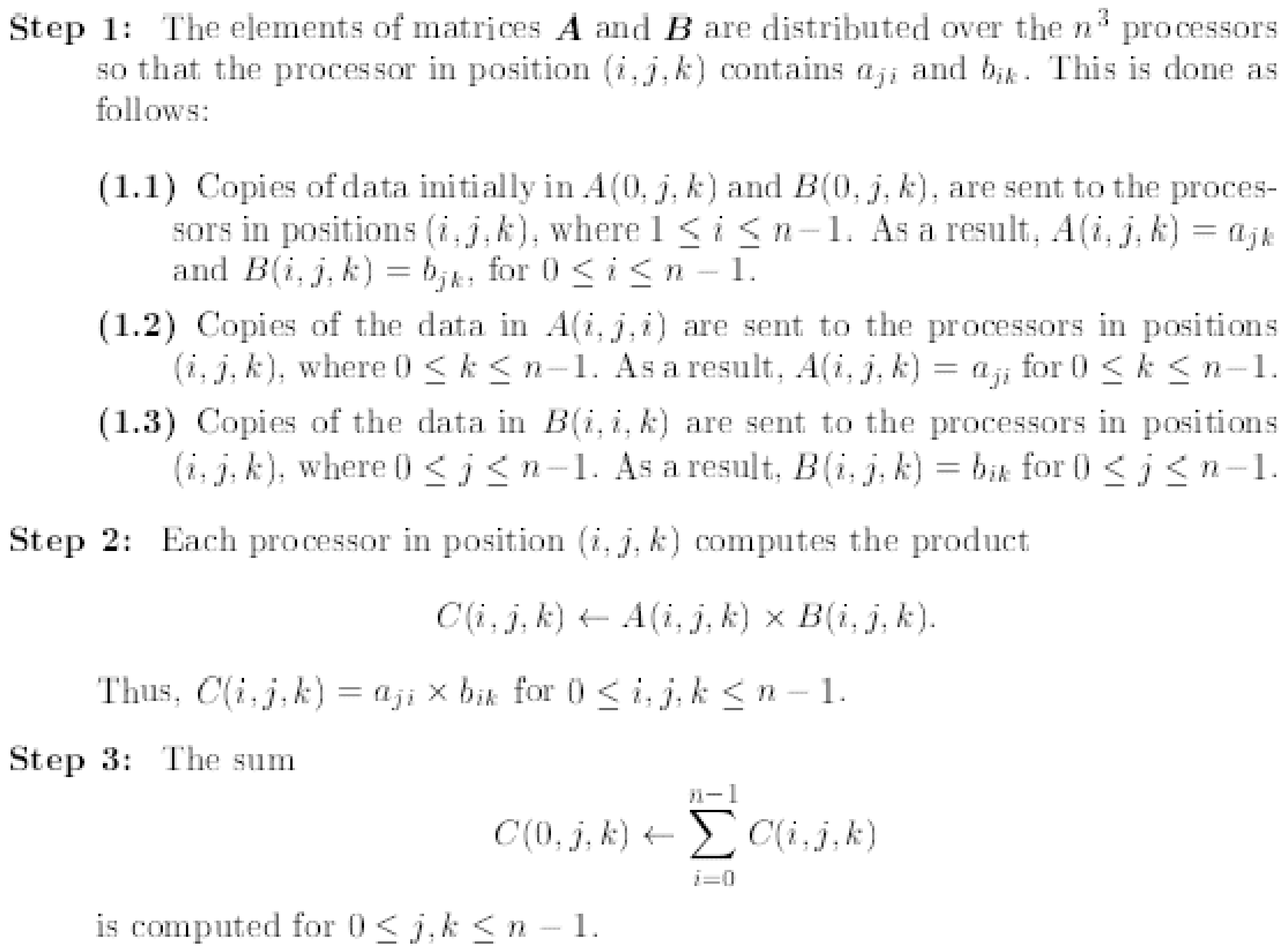
Hypercube Interconnection Network



Parallel Matrix Multiplication on Hypercube

- Use $N = n^3 = 2^{3q}$ processors, where $n = 2^q$.
- Can visualize the processors as being arranged in an $n \times n \times n$ array, with processor P_r occupying position (i, j, k), where $r = in^2 + jn + k$ and $0 \leq i, j, k \leq n - 1$
  - The binary representation of r is
  - $r_{3q-1}r_{3q-2} \dots r_{2q}r_{2q-1} \dots r_q r_{q-1} \dots r_0$
  - The binary representations of i, j, and k are
  - $r_{3q-1}r_{3q-2} \dots r_{2q}, \quad r_{2q-1}r_{2q-2} \dots r_q, \quad r_{q-1}r_{q-2} \dots r_0$
  - Processors agreeing on one or two of the coordinates (i, j, k) form a hypercube.
  - Processors agreeing on one coordinate form a hypercube with n^2 processors.
  - Processors agreeing on two coordinates form a hypercube with n processors.
- Each processor has registers A, B, and C, denoted A(i, j, k), B(i, j, k), and C(i, j, k) for local storage.

Parallel Matrix Multiplication on Hypercube



Step 1: The elements of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are distributed over the n^3 processors so that the processor in position (i, j, k) contains a_{ji} and b_{ik}. This is done as follows:

(1.1) Copies of data initially in $A(0, j, k)$ and $B(0, j, k)$, are sent to the processors in positions (i, j, k), where $1 \leq i \leq n-1$. As a result, $A(i, j, k) = a_{jk}$ and $B(i, j, k) = b_{jk}$, for $0 \leq i \leq n-1$.

(1.2) Copies of the data in $A(i, j, i)$ are sent to the processors in positions (i, j, k), where $0 \leq k \leq n-1$. As a result, $A(i, j, k) = a_{ji}$ for $0 \leq k \leq n-1$.

(1.3) Copies of the data in $B(i, i, k)$ are sent to the processors in positions (i, j, k), where $0 \leq j \leq n-1$. As a result, $B(i, j, k) = b_{ik}$ for $0 \leq j \leq n-1$.

Step 2: Each processor in position (i, j, k) computes the product

$$C(i, j, k) \leftarrow A(i, j, k) \times B(i, j, k).$$

Thus, $C(i, j, k) = a_{ji} \times b_{ik}$ for $0 \leq i, j, k \leq n-1$.

Step 3: The sum

$$C(0, j, k) \leftarrow \sum_{i=0}^{n-1} C(i, j, k)$$

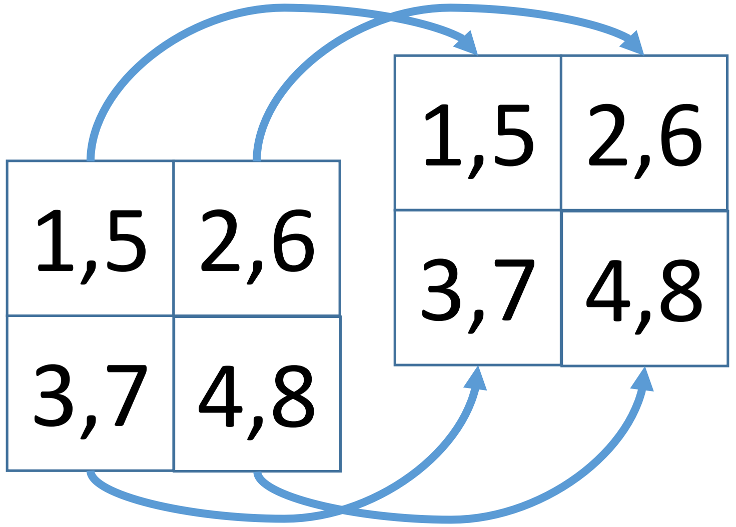
is computed for $0 \leq j, k \leq n-1$.

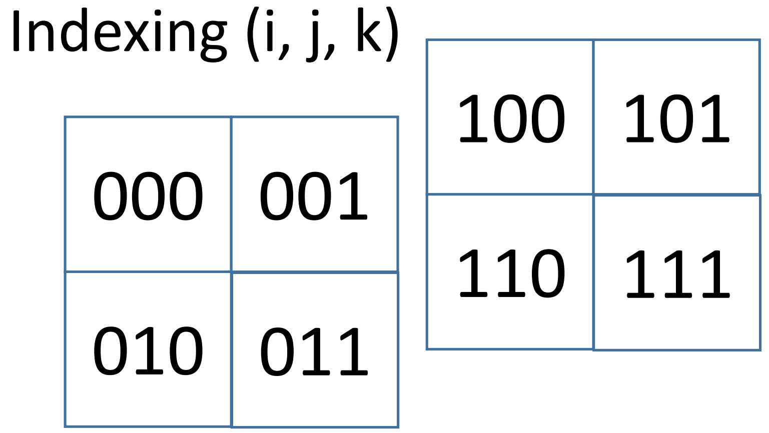
Parallel Matrix Multiplication on Hypercube (The idea)

Example: Multiplying 2×2 matrices

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

- Step 1.1: Data in $A(0, j, k)$ and $B(0, j, k)$ are sent to processors in positions (i, j, k), where $1 \le i \le n - 1$.

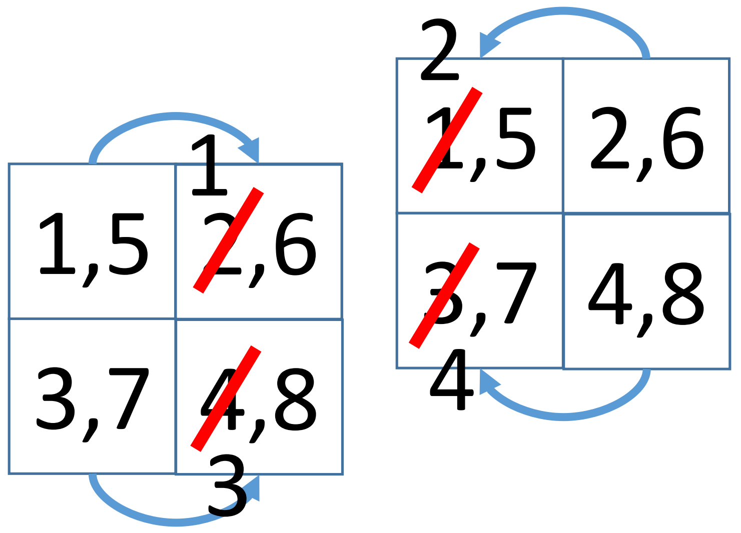


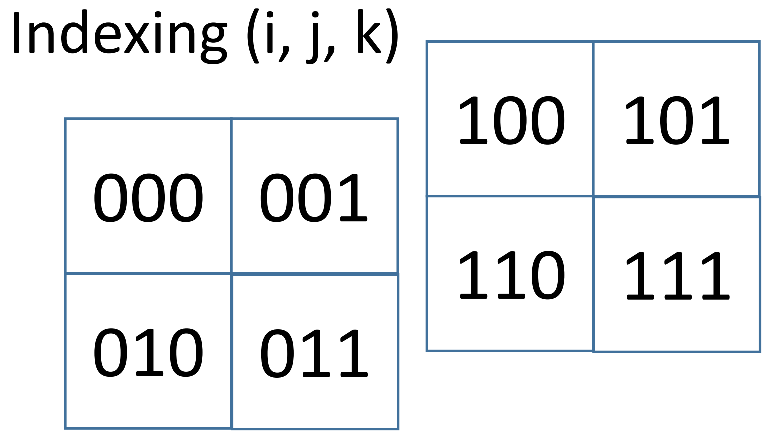


Parallel Matrix Multiplication on Hypercube (The idea)

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \times \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}19 & 22\\43 & 50\end{bmatrix}$$

- Step 1.2: Copies of data in $A(i,j,i)$ are sent to the processors in positions (i,j,k), where $0 \leq k \leq n-1$.

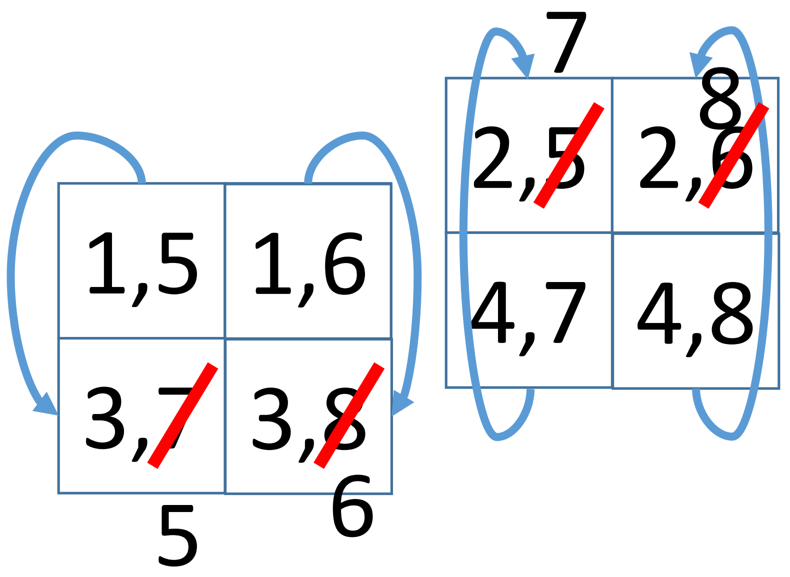


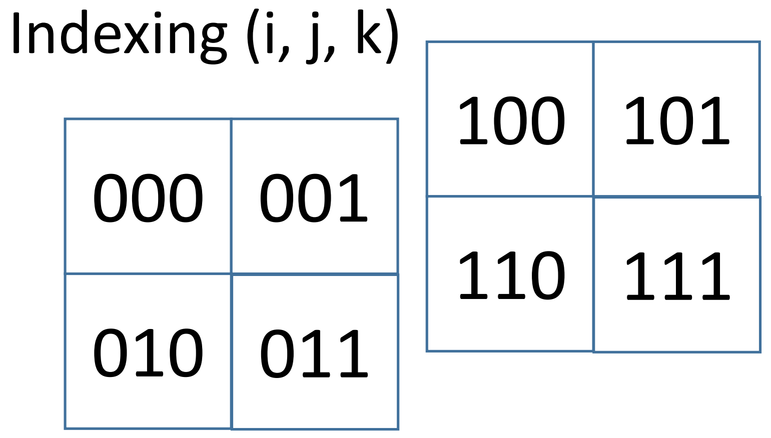


Parallel Matrix Multiplication on Hypercube (The idea)

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \times \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}19 & 22\\43 & 50\end{bmatrix}$$

- Step 1.3: Copies of data in $B(i,i,k)$ are sent to the processors in positions (i,j,k), where $0 \le j \le n-1$.

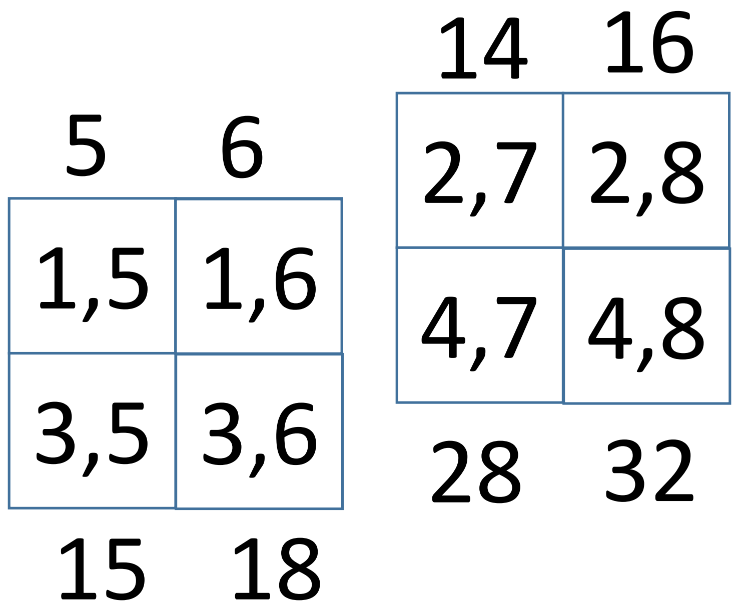


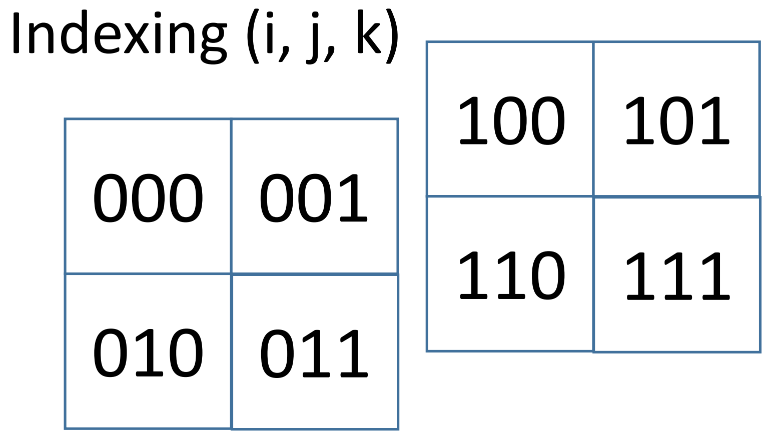


Parallel Matrix Multiplication on Hypercube (The idea)

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \times \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}19 & 22\\43 & 50\end{bmatrix}$$

- Step 2: Each processor computes the product of their local A and B registers and stores it in their C register.





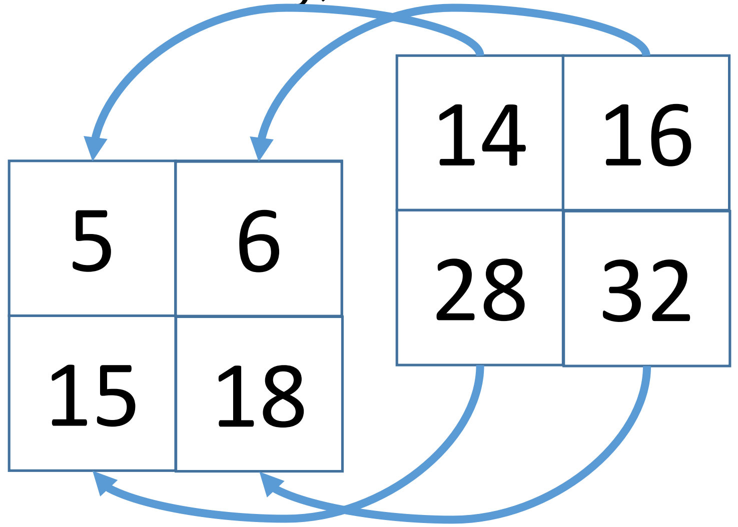
Parallel Matrix Multiplication on Hypercube (The idea)

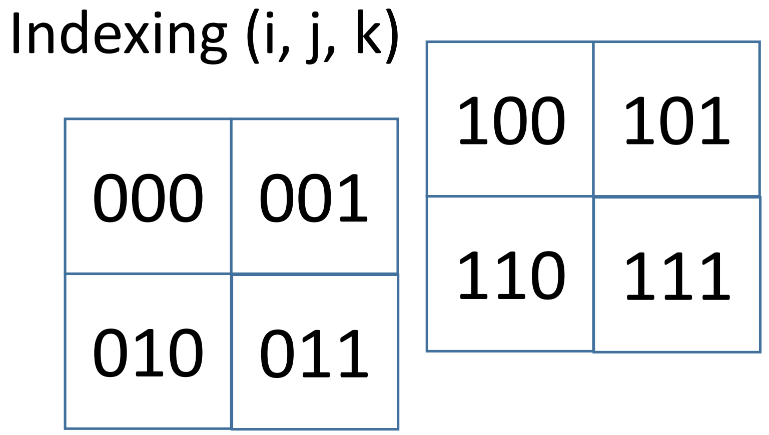
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

- Step 3: The sum

$$C(0,j,k) = \sum_{i=0}^{n-1} C(i,j,k)$$

is computed for $0 \leq j,k \leq n-1$.





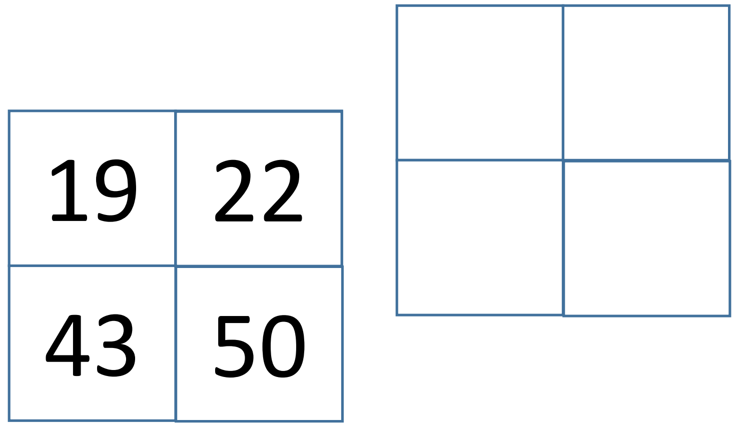
Parallel Matrix Multiplication on Hypercube (The idea)

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \times \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}19 & 22\\43 & 50\end{bmatrix}$$

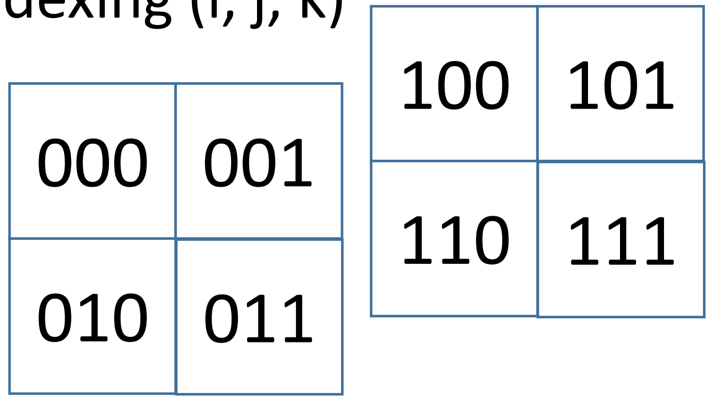
- Step 3: The sum

$$C(0,j,k) = \sum_{i=0}^{n-1} C(i,j,k)$$

is computed for $0 \le j, k \le n-1$.



Indexing (i, j, k)



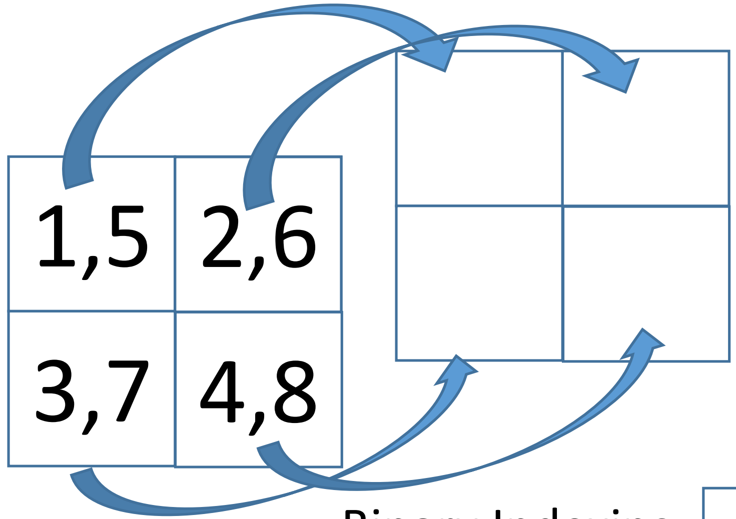
Parallel Matrix Multiplication on Hypercube

```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
  Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                    (i) $A_{r^{(m)}} \leftarrow A_r$
                    (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
          (1.2) for $m = q - 1$ downto $0$ do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                    $A_{r^{(m)}} \leftarrow A_r$
                  end for
                end for
          (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                    $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
  Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
          end for
  Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
              $C_r \leftarrow C_r + C_{r^{(m)}}$
            end for
          end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$

$m = 2$



Binary Indexing

000	001
010	011

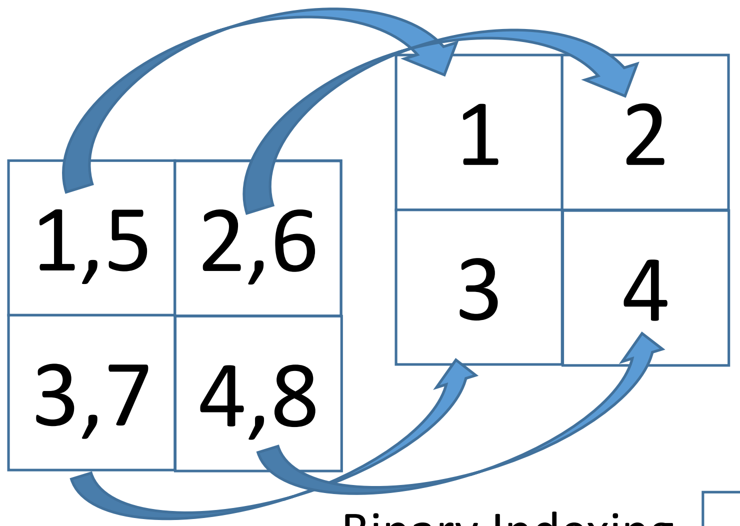
100	101
110	111

Parallel Matrix Multiplication on Hypercube

```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
  Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                      (i) $A_{r^{(m)}} \leftarrow A_r$
                      (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
          (1.2) for $m = q - 1$ downto $0$ do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                      $A_{r^{(m)}} \leftarrow A_r$
                  end for
                end for
          (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                      $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
  Step 2: for $r = 0$ to $N - 1$ do in parallel
              $C_r \leftarrow A_r \times B_r$
          end for
  Step 3: for $m = 2q$ to $3q - 1$ do
              for all $r \in N(r_m = 0)$ do in parallel
                  $C_r \leftarrow C_r + C_{r^{(m)}}$
              end for
          end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$
$m = 2$



Binary Indexing

000	001
010	011

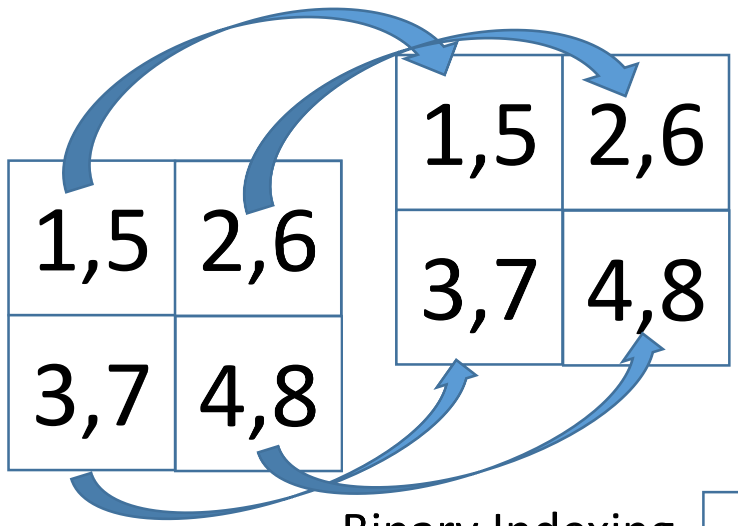
100	101
110	111

Parallel Matrix Multiplication on Hypercube

```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
 Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                     (i) $A_{r(m)} \leftarrow A_r$
                    (ii) $B_{r(m)} \leftarrow B_r$
                  end for
               end for
         (1.2) for $m = q - 1$ downto 0 do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                     $A_{r(m)} \leftarrow A_r$
                  end for
               end for
         (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                     $B_{r(m)} \leftarrow B_r$
                  end for
               end for
 Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
         end for
 Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
               $C_r \leftarrow C_r + C_{r(m)}$
            end for
         end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$
$m = 2$



Binary Indexing

000	001
010	011

100	101
110	111

Parallel Matrix Multiplication on Hypercube

Algorithm HYPERCUBE MATRIX MULTIPLICATION (A, B, C)

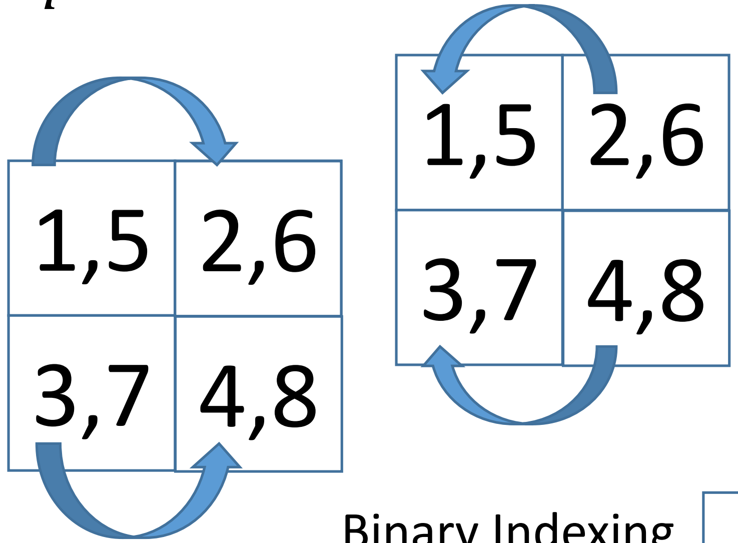
```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                     (i) $A_{r^{(m)}} \leftarrow A_r$
                     (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
              end for
        (1.2) for $m = q - 1$ downto $0$ do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                     $A_{r^{(m)}} \leftarrow A_r$
                  end for
              end for
        (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                     $B_{r^{(m)}} \leftarrow B_r$
                  end for
              end for
Step 2: for $r = 0$ to $N - 1$ do in parallel
           $C_r \leftarrow A_r \times B_r$
        end for
Step 3: for $m = 2q$ to $3q - 1$ do
           for all $r \in N(r_m = 0)$ do in parallel
              $C_r \leftarrow C_r + C_{r^{(m)}}$
           end for
        end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$

$m = 0$

$2q + m = 2$



Binary Indexing

000	001
010	011

100	101
110	111

Parallel Matrix Multiplication on Hypercube

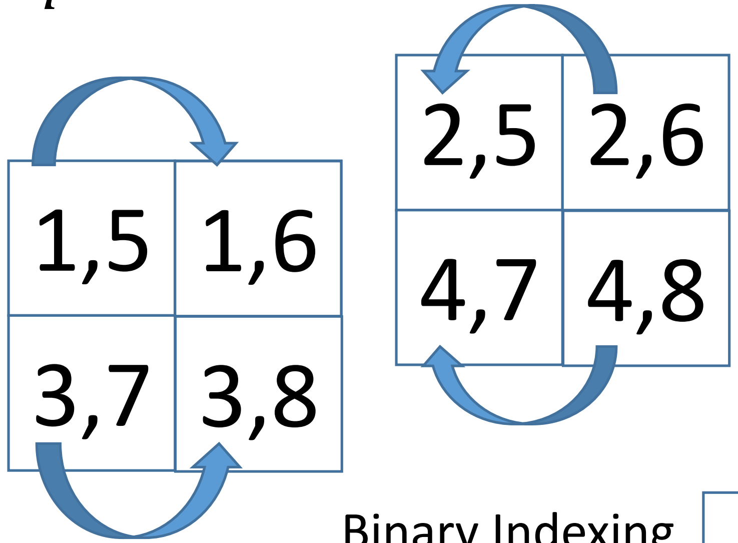
```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
  Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                    (i) $A_{r^{(m)}} \leftarrow A_r$
                    (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
          (1.2) for $m = q - 1$ downto $0$ do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                    $A_{r^{(m)}} \leftarrow A_r$
                  end for
                end for
          (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                    $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
  Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
          end for
  Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
              $C_r \leftarrow C_r + C_{r^{(m)}}$
            end for
          end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$

$m = 0$

$2q + m = 2$



Binary Indexing

000	001
010	011

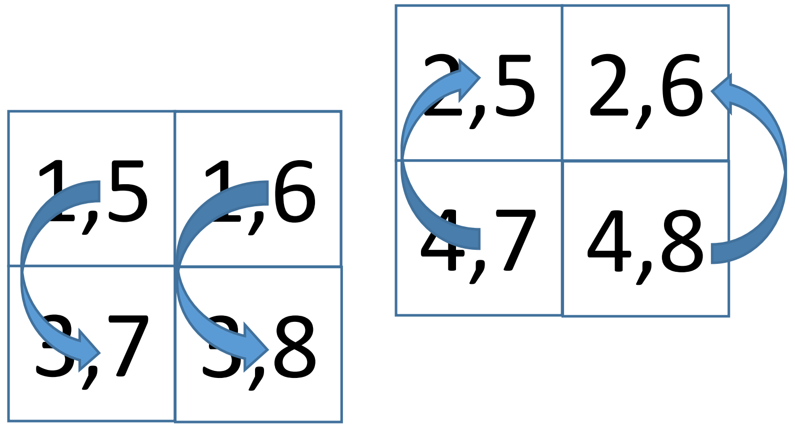
100	101
110	111

Parallel Matrix Multiplication on Hypercube

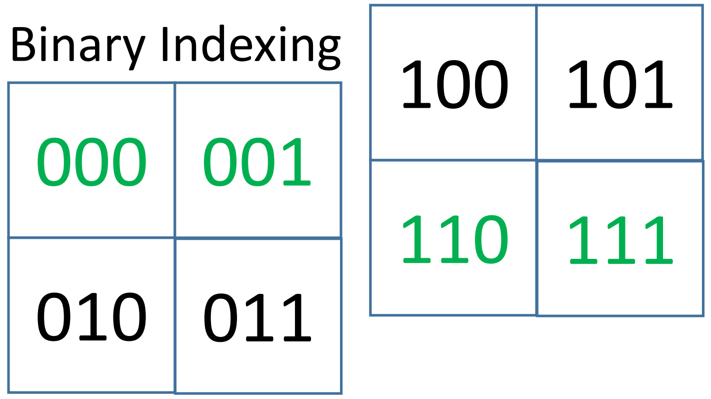
Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$
$m = 1$
$q + m = 2$

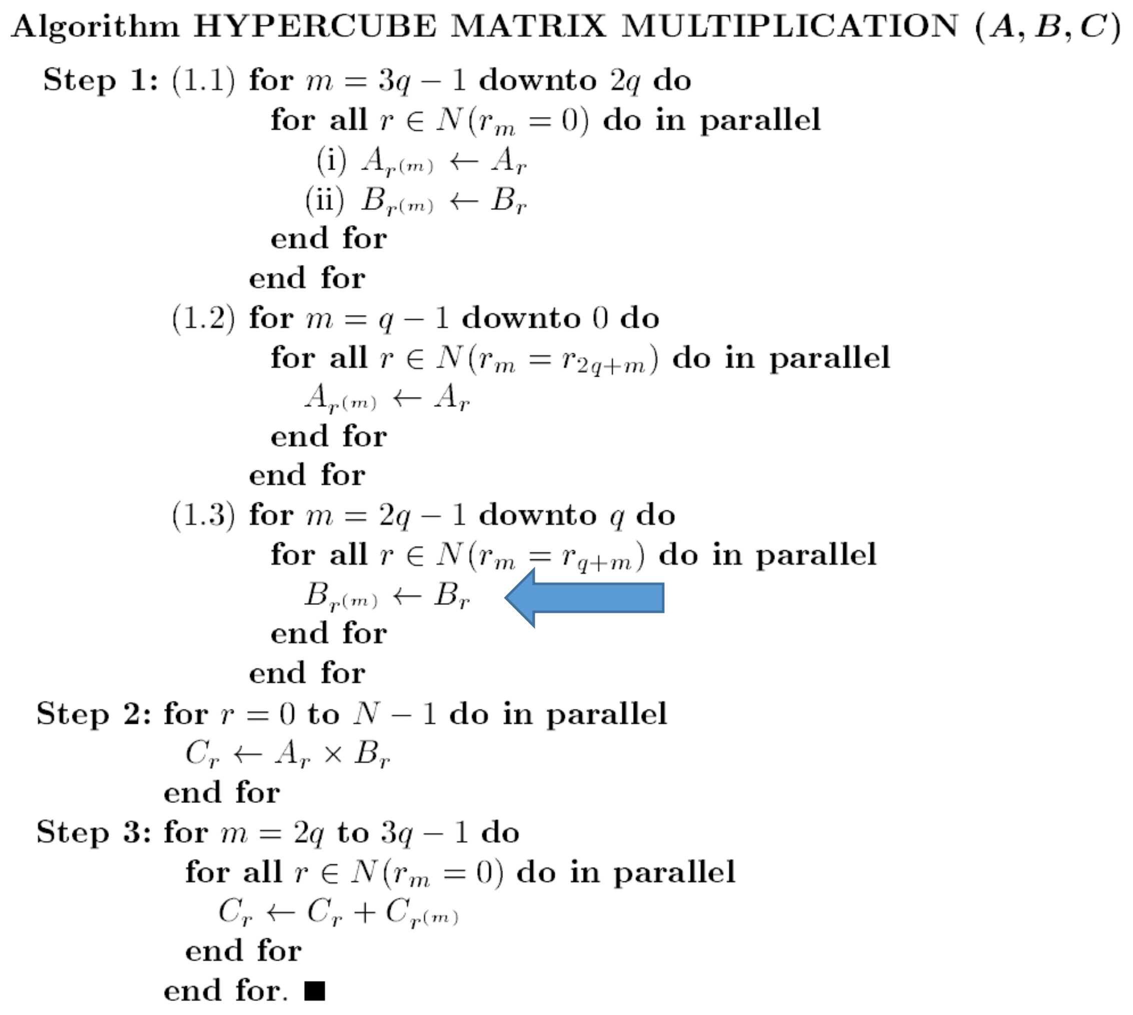
```
Algorithm HYPERCUBE MATRIX MULTIPLICATION ($A, B, C$)
  Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                    (i) $A_{r(m)} \leftarrow A_r$
                    (ii) $B_{r(m)} \leftarrow B_r$
                  end for
                end for
          (1.2) for $m = q - 1$ downto 0 do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                    $A_{r(m)} \leftarrow A_r$
                  end for
                end for
          (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                    $B_{r(m)} \leftarrow B_r$ ⟵
                  end for
                end for
  Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
          end for
  Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
              $C_r \leftarrow C_r + C_{r(m)}$
            end for
          end for. ■
```



Binary Indexing



Parallel Matrix Multiplication on Hypercube



```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
  Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                    (i) $A_{r^{(m)}} \leftarrow A_r$
                    (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
          (1.2) for $m = q - 1$ downto $0$ do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                    $A_{r^{(m)}} \leftarrow A_r$
                  end for
                end for
          (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                    $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
  Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
          end for
  Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
              $C_r \leftarrow C_r + C_{r^{(m)}}$
            end for
          end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$

$m = 1$

$q + m = 2$

2,7	2,8
4,7	4,8

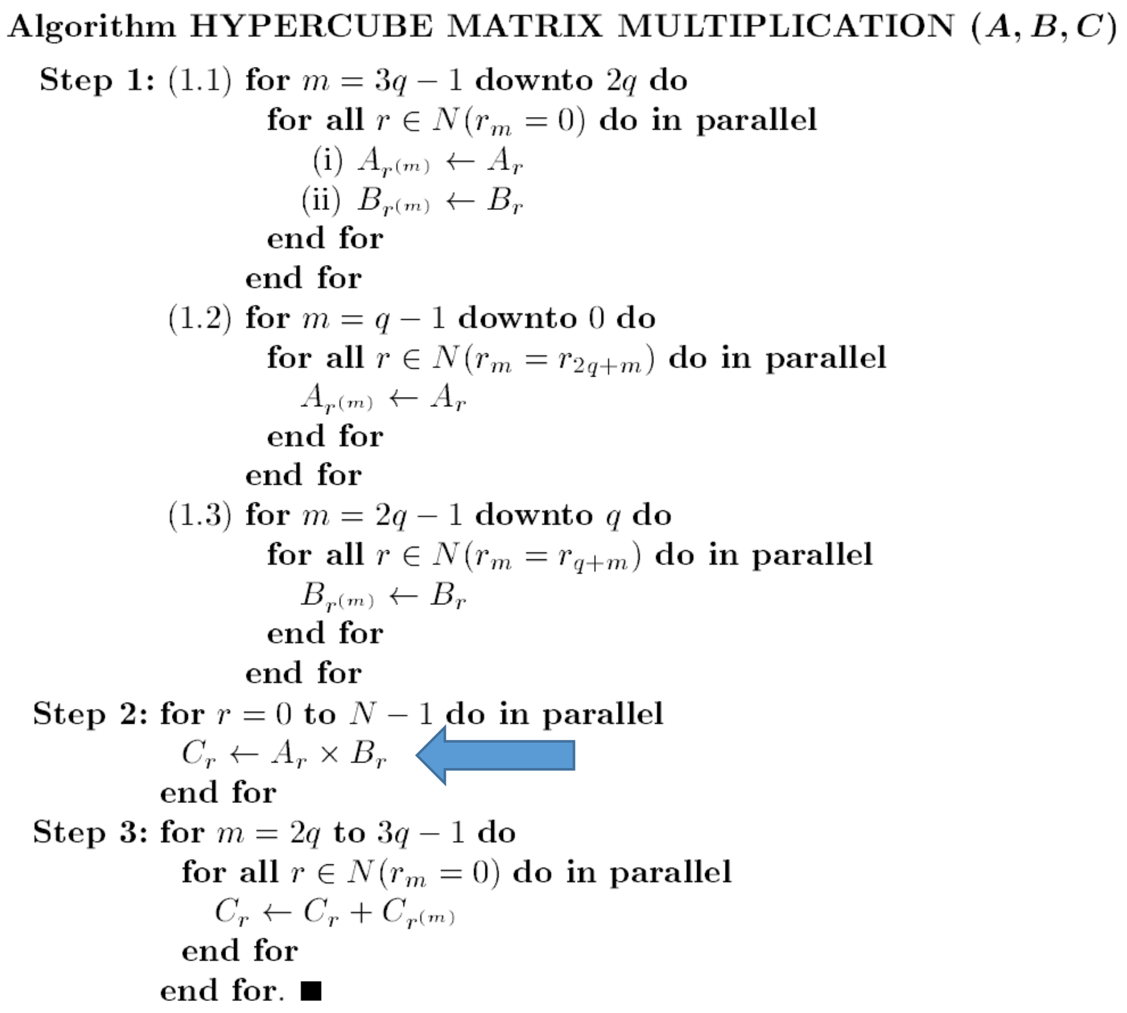
1,5	1,6
3,5	3,6

Binary Indexing

000	001
010	011

100	101
110	111

Parallel Matrix Multiplication on Hypercube



```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
 Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                 for all $r \in N(r_m = 0)$ do in parallel
                   (i) $A_{r(m)} \leftarrow A_r$
                   (ii) $B_{r(m)} \leftarrow B_r$
                 end for
               end for
         (1.2) for $m = q - 1$ downto $0$ do
                 for all $r \in N(r_m = r_{2q+m})$ do in parallel
                   $A_{r(m)} \leftarrow A_r$
                 end for
               end for
         (1.3) for $m = 2q - 1$ downto $q$ do
                 for all $r \in N(r_m = r_{q+m})$ do in parallel
                   $B_{r(m)} \leftarrow B_r$
                 end for
               end for
 Step 2: for $r = 0$ to $N - 1$ do in parallel
           $C_r \leftarrow A_r \times B_r$
         end for
 Step 3: for $m = 2q$ to $3q - 1$ do
           for all $r \in N(r_m = 0)$ do in parallel
             $C_r \leftarrow C_r + C_{r(m)}$
           end for
         end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$
$m = 1$

1x5	1x6
3x5	3x6

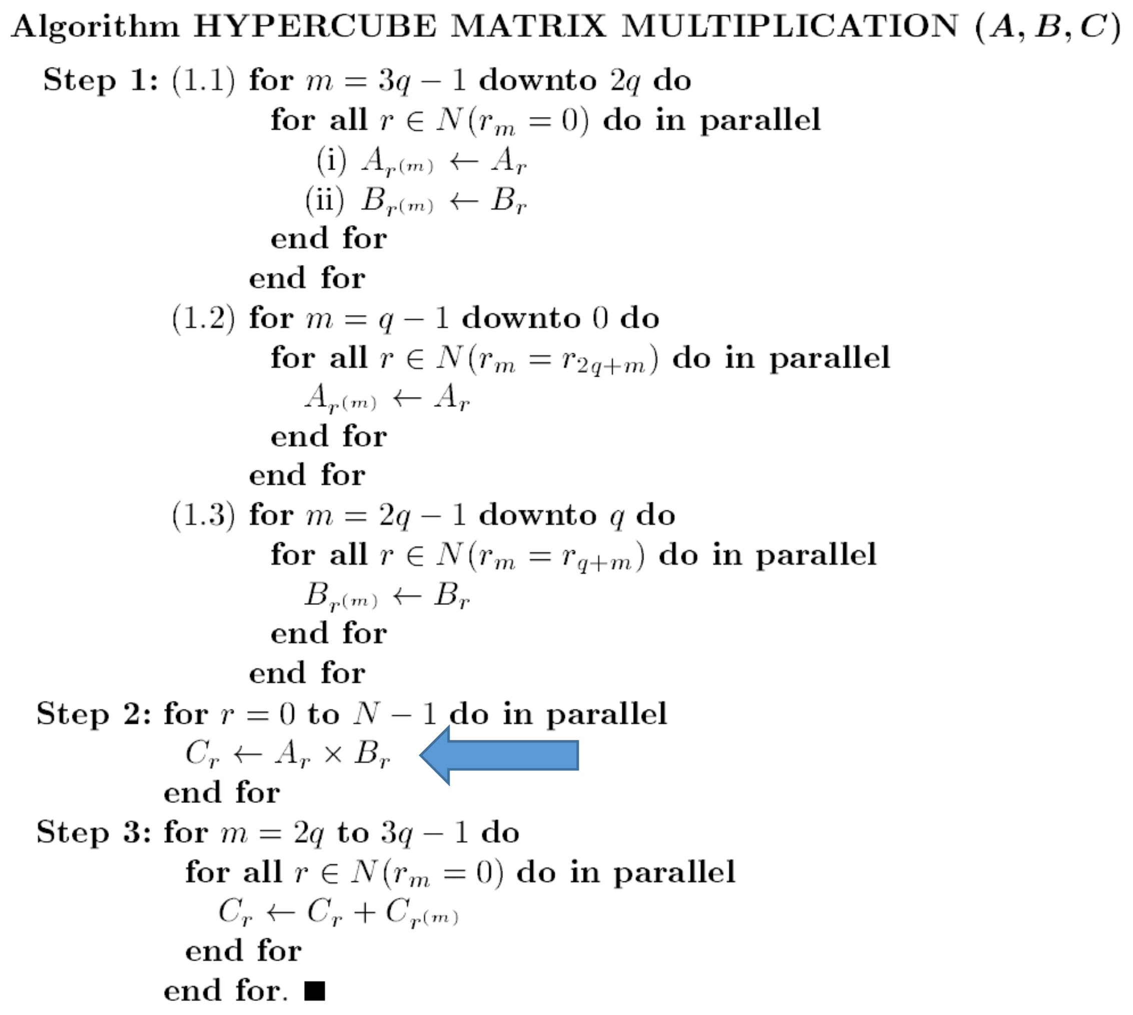
2x7	2x8
4x7	4x8

Binary Indexing

000	001
010	011

100	101
110	111

Parallel Matrix Multiplication on Hypercube



```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
  Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                    (i) $A_{r^{(m)}} \leftarrow A_r$
                    (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
          (1.2) for $m = q - 1$ downto $0$ do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                    $A_{r^{(m)}} \leftarrow A_r$
                  end for
                end for
          (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                    $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
  Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
          end for
  Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
              $C_r \leftarrow C_r + C_{r^{(m)}}$
            end for
          end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$

$m = 1$

14	16
28	32

5	6
15	18

Binary Indexing

100	101
110	111

000	001
010	011

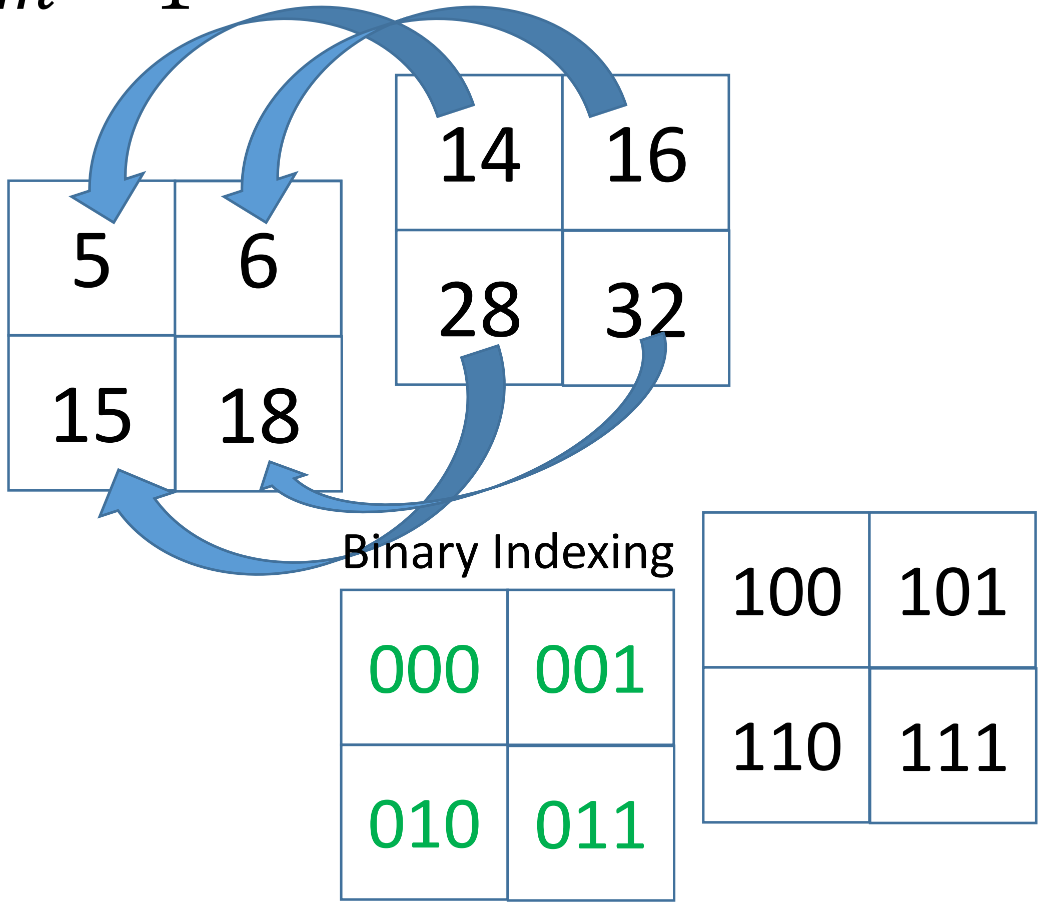
Parallel Matrix Multiplication on Hypercube

```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
  Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                    (i) $A_{r^{(m)}} \leftarrow A_r$
                    (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
          (1.2) for $m = q - 1$ downto $0$ do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                    $A_{r^{(m)}} \leftarrow A_r$
                  end for
                end for
          (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                    $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
  Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
          end for
  Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
              $C_r \leftarrow C_r + C_{r^{(m)}}$
            end for
          end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$

$m = 1$



Parallel Matrix Multiplication on Hypercube

Algorithm HYPERCUBE MATRIX MULTIPLICATION (A, B, C)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                      (i) $A_{r(m)} \leftarrow A_r$
                      (ii) $B_{r(m)} \leftarrow B_r$
                  end for
              end for
        (1.2) for $m = q - 1$ downto $0$ do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                      $A_{r(m)} \leftarrow A_r$
                  end for
              end for
        (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                      $B_{r(m)} \leftarrow B_r$
                  end for
              end for
Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
        end for
Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
                $C_r \leftarrow C_r + C_{r(m)}$ ⇐
            end for
        end for. ■
```

Use $N = n^3 = 2^{3q}$ processors.

$n = 2, q = 1$

$m = 1$

19	22
43	50

14	16
28	32

Binary Indexing

000	001
010	011

100	101
110	111

Parallel Matrix Multiplication on Hypercube

```
Algorithm HYPERCUBE MATRIX MULTIPLICATION $(A, B, C)$
  Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                    (i) $A_{r^{(m)}} \leftarrow A_r$
                    (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
          (1.2) for $m = q - 1$ downto 0 do
                  for all $r \in N(r_m = r_{2q+m})$ do in parallel
                    $A_{r^{(m)}} \leftarrow A_r$
                  end for
                end for
          (1.3) for $m = 2q - 1$ downto $q$ do
                  for all $r \in N(r_m = r_{q+m})$ do in parallel
                    $B_{r^{(m)}} \leftarrow B_r$
                  end for
                end for
  Step 2: for $r = 0$ to $N - 1$ do in parallel
            $C_r \leftarrow A_r \times B_r$
          end for
  Step 3: for $m = 2q$ to $3q - 1$ do
            for all $r \in N(r_m = 0)$ do in parallel
              $C_r \leftarrow C_r + C_{r^{(m)}}$
            end for
          end for. ■
```

Analysis:

Steps 1.1, 1.2, 1.3, and step 3 require q iterations.

Step 2 is in constant time.

Time complexity is $O(q) = O(\log n)$. (how?)

$P_n = n^3$, so cost is $n^3 \cdot \log n$. This is not cost optimal because the straightforward RAM algorithm takes $O(n^3)$ multiplications.

Parallel Matrix Multiplication on Hypercube

- The next slides will illustrate the $O(\log n)$ nature of data distribution for 2×2, 4×4, and 8×8 matrices.
- Only step 1.1 will be visualized, but a similar concept applies for steps 1.2, 1.3, and step 3.
- Observe that the distance between the senders and receivers is halved each iteration.
- Processors holding useful data are shaded in blue. Some processors send data even though they don't hold useful data yet, but this is not a problem.

Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
          for all $r \in N(r_m = 0)$ do in parallel
            (i) $A_{r^{(m)}} \leftarrow A_r$
            (ii) $B_{r^{(m)}} \leftarrow B_r$
          end for
        end for
```

For all where the mth bit is 0

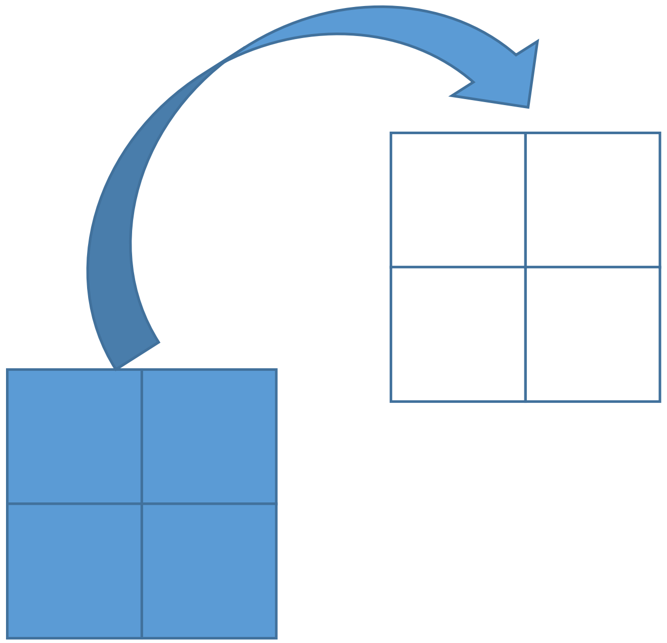
Send to neighboring processor by flipping bit m

2×2 distribution

q = 1

m starts at $3q - 1 = 2$

m ends at $2q = 2$



Binary coordinates:

000 001

010 011

Binary coordinates:

100 101

110 111

Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
          for all $r \in N(r_m = 0)$ do in parallel
              (i)  $A_{r^{(m)}} \leftarrow A_r$
              (ii) $B_{r^{(m)}} \leftarrow B_r$
          end for
        end for
```

For all where the mth bit is 0

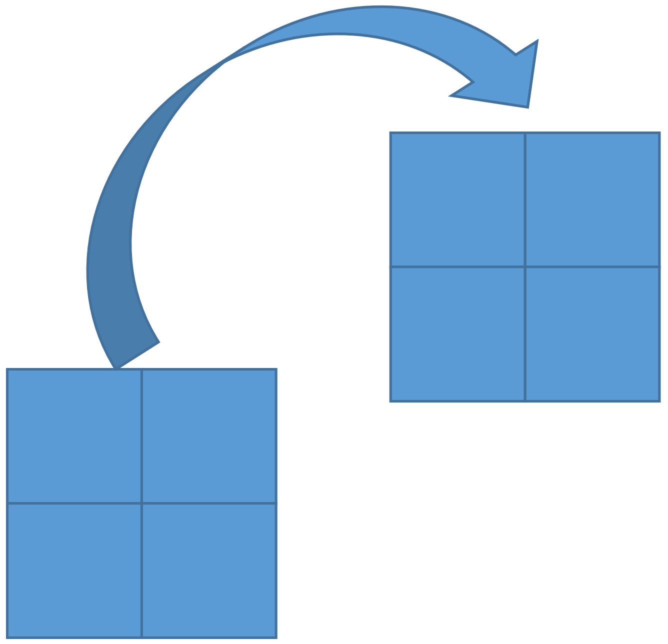
Send to neighboring processor by flipping bit m

2×2 distribution

q = 1

m starts at $3q - 1 = 2$

m ends at $2q = 2$



Binary coordinates:

100 101

110 111

Binary coordinates:

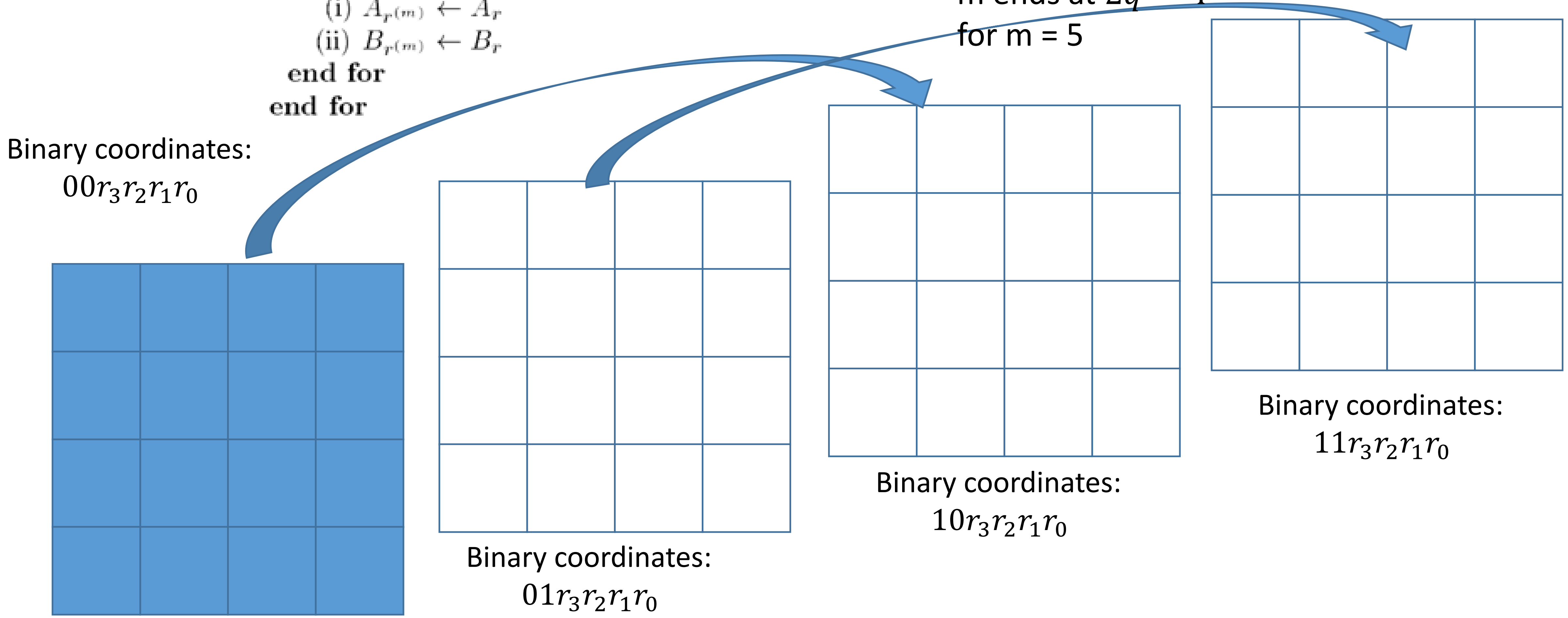
000 001

010 011

Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
            for all $r \in N(r_m = 0)$ do in parallel
                (i) $A_{r^{(m)}} \leftarrow A_r$
                (ii) $B_{r^{(m)}} \leftarrow B_r$
            end for
        end for
```

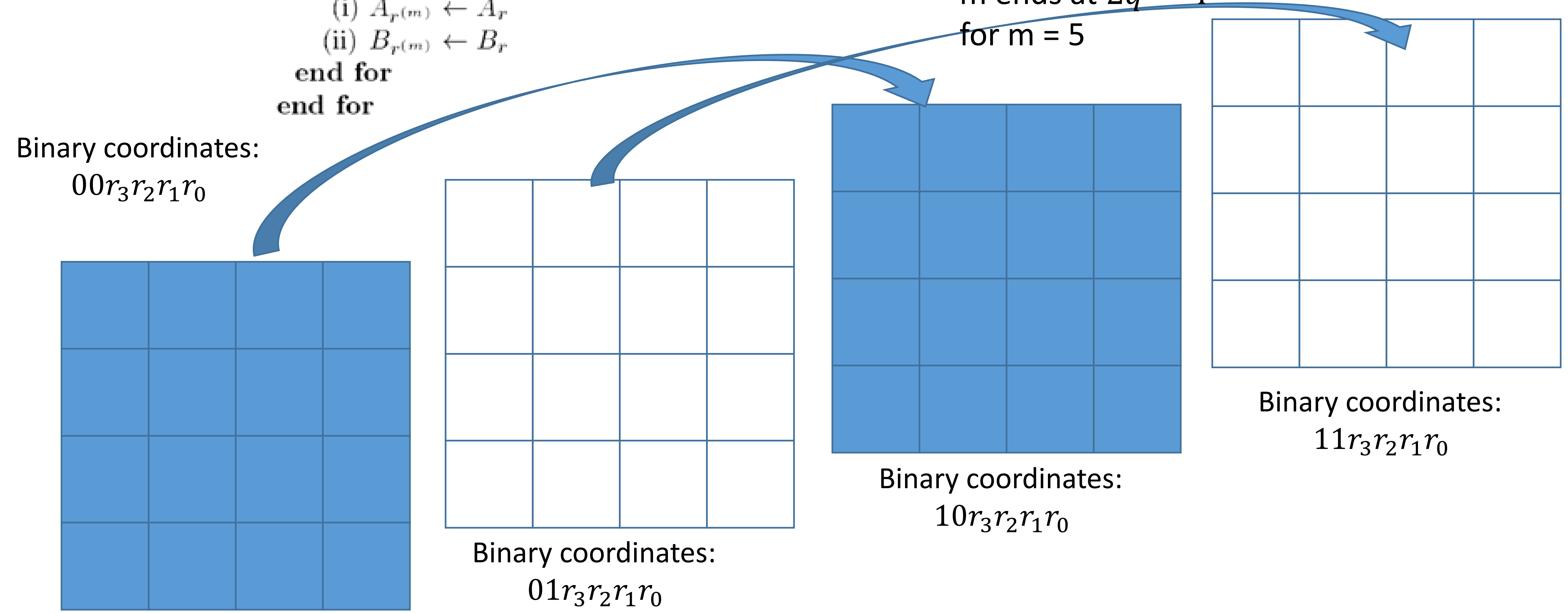
4×4 distribution
q = 2
m starts at $3q - 1 = 5$
m ends at $2q = 4$
for m = 5



Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                for all $r \in N(r_m = 0)$ do in parallel
                    (i) $A_{r^{(m)}} \leftarrow A_r$
                    (ii) $B_{r^{(m)}} \leftarrow B_r$
                end for
              end for
```

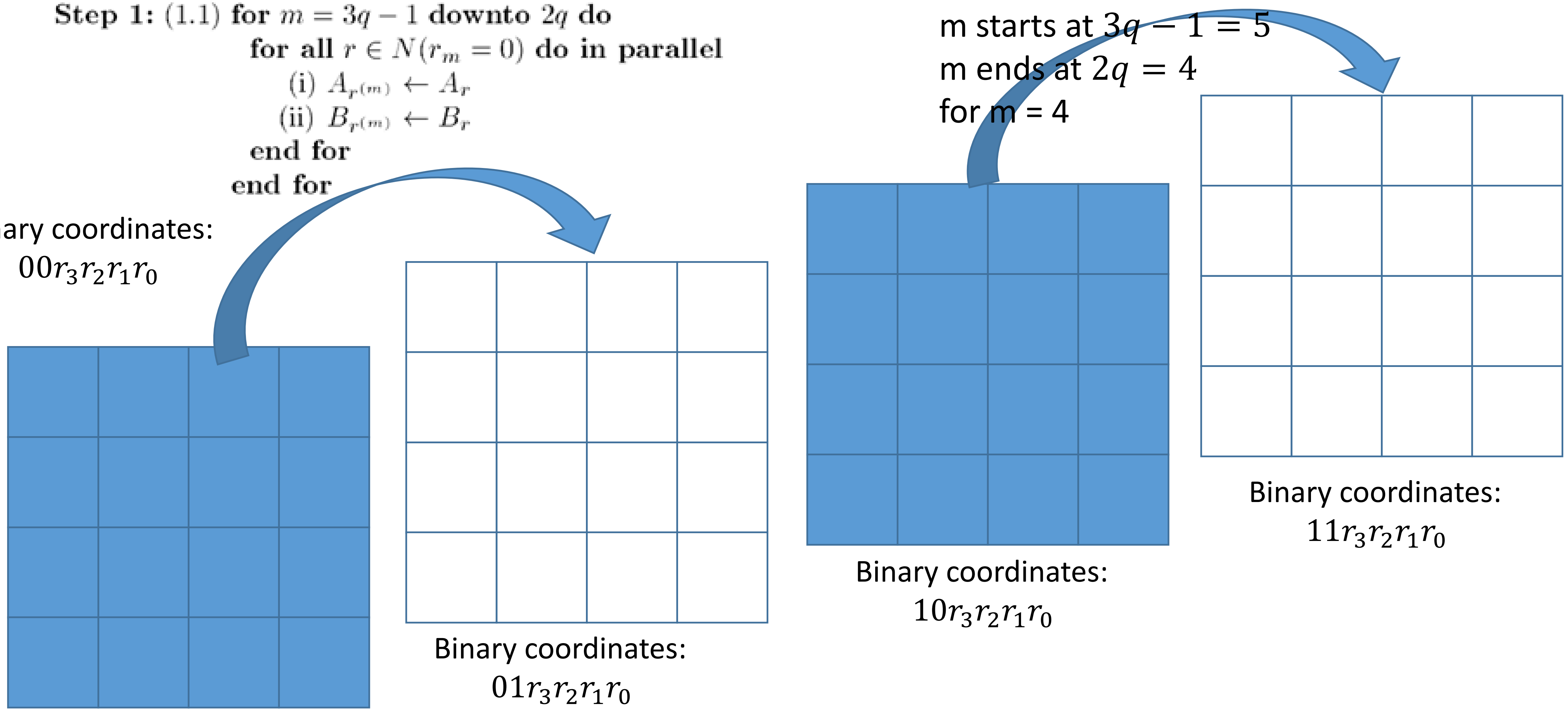
4×4 distribution
q = 2
m starts at $3q - 1 = 5$
m ends at $2q = 4$
for m = 5


Binary coordinates: $00r_3r_2r_1r_0$
Binary coordinates: $01r_3r_2r_1r_0$
Binary coordinates: $10r_3r_2r_1r_0$
Binary coordinates: $11r_3r_2r_1r_0$

Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                      (i) $A_{r^{(m)}} \leftarrow A_r$
                      (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
              end for
```

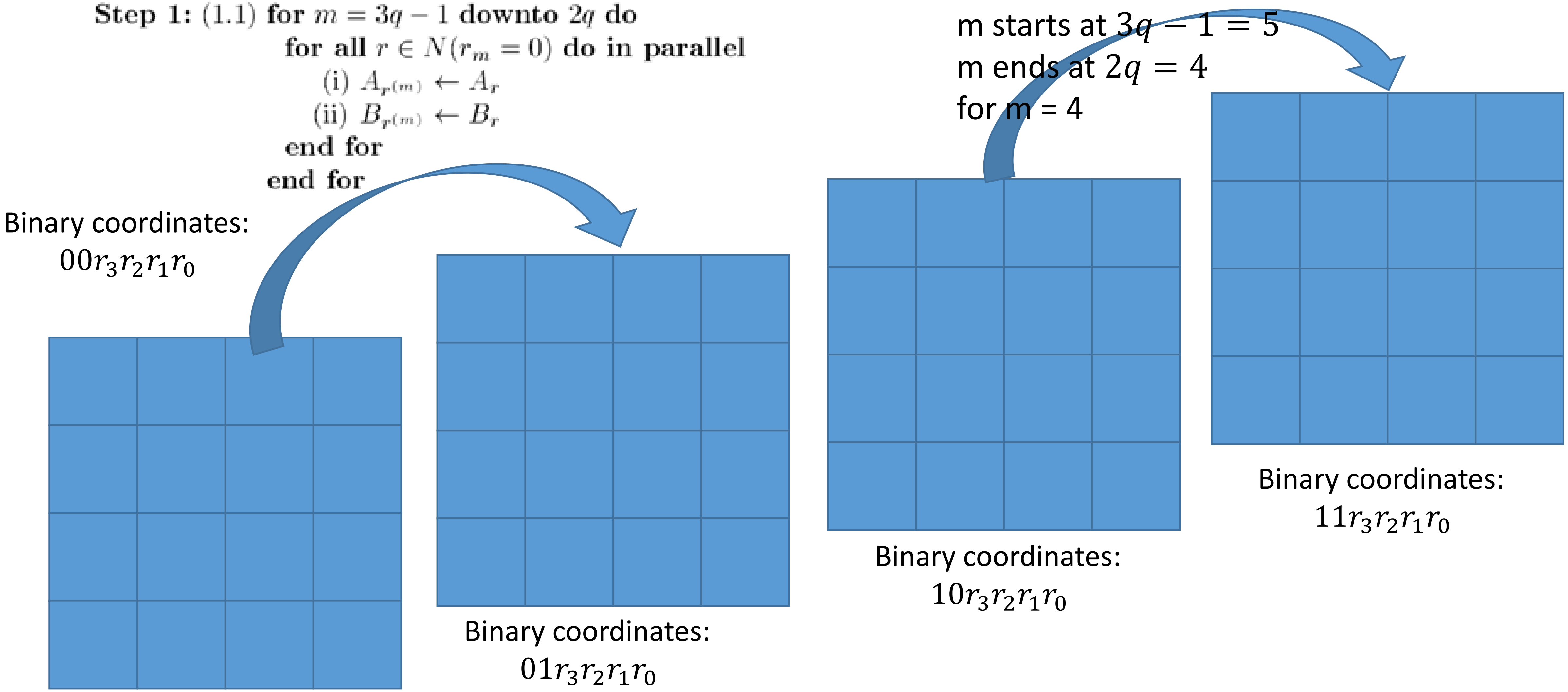
4×4 distribution
q = 2
m starts at $3q - 1 = 5$
m ends at $2q = 4$
for m = 4



Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                     (i) $A_{r^{(m)}} \leftarrow A_r$
                     (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
              end for
```

4×4 distribution
q = 2
m starts at $3q - 1 = 5$
m ends at $2q = 4$
for m = 4



Binary coordinates: $00r_3r_2r_1r_0$
Binary coordinates: $01r_3r_2r_1r_0$
Binary coordinates: $10r_3r_2r_1r_0$
Binary coordinates: $11r_3r_2r_1r_0$

Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
            for all $r \in N(r_m = 0)$ do in parallel
                (i) $A_{r^{(m)}} \leftarrow A_r$
                (ii) $B_{r^{(m)}} \leftarrow B_r$
            end for
        end for
```

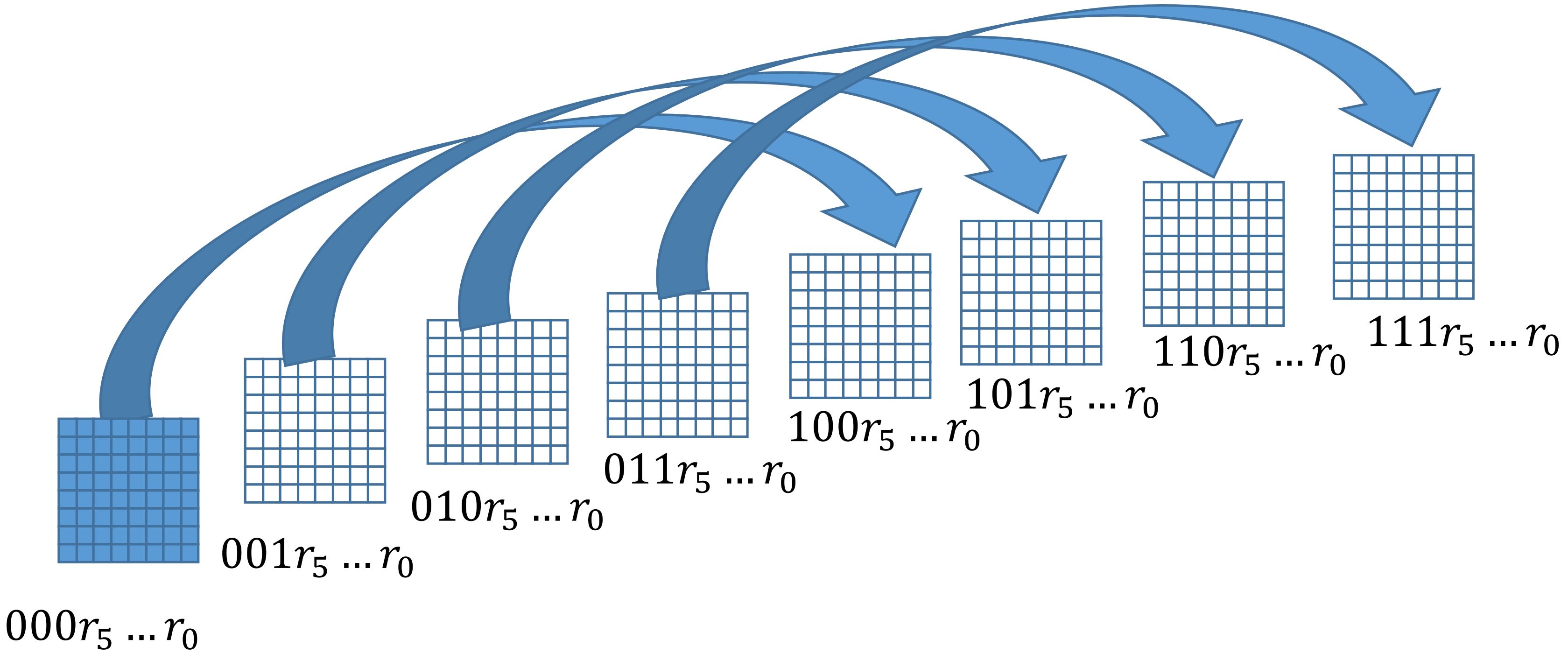
8×8 distribution

q = 3

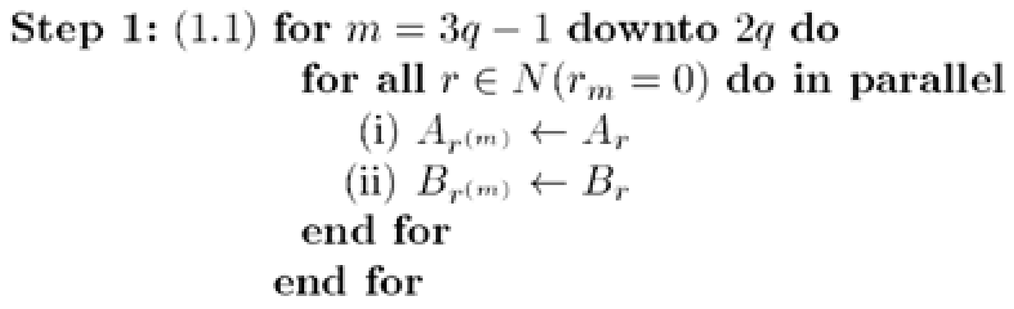
m starts at $3q - 1 = 8$

m ends at $2q = 6$

for m = 8



Parallel Matrix Multiplication on Hypercube (log n distribution example)



```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
            for all $r \in N(r_m = 0)$ do in parallel
              (i) $A_{r^{(m)}} \leftarrow A_r$
              (ii) $B_{r^{(m)}} \leftarrow B_r$
            end for
          end for
```

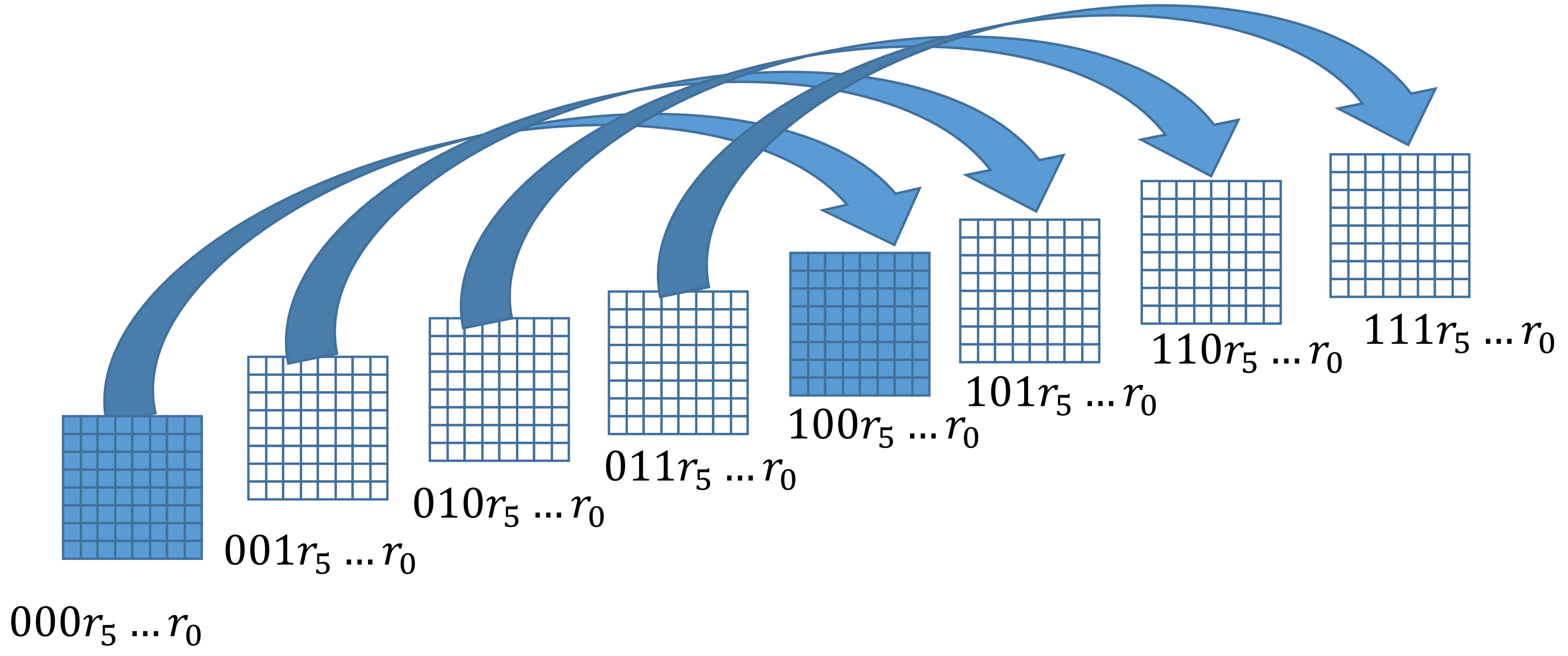
8×8 distribution

q = 3

m starts at $3q - 1 = 8$

m ends at $2q = 6$

for m = 8



Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                      (i) $A_{r^{(m)}} \leftarrow A_r$
                      (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
              end for
```

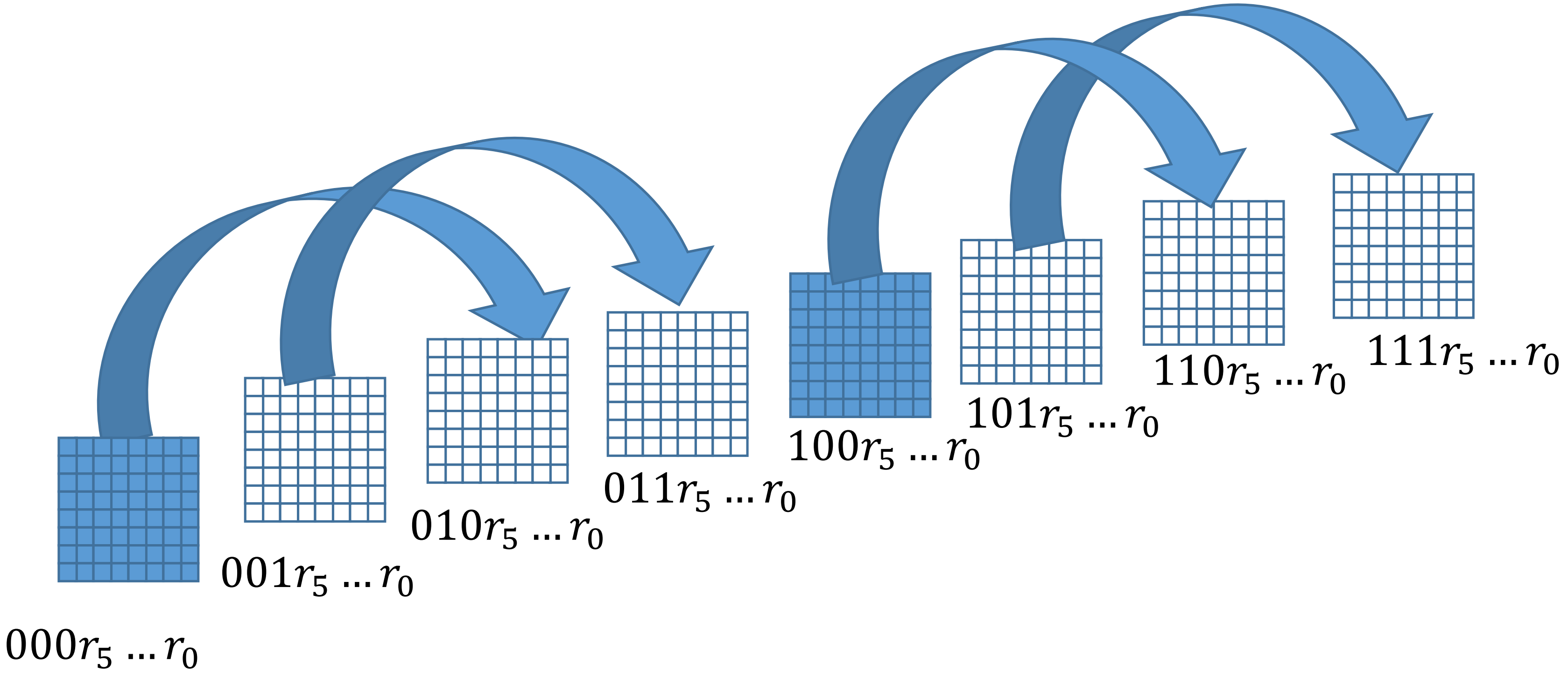
8×8 distribution

q = 3

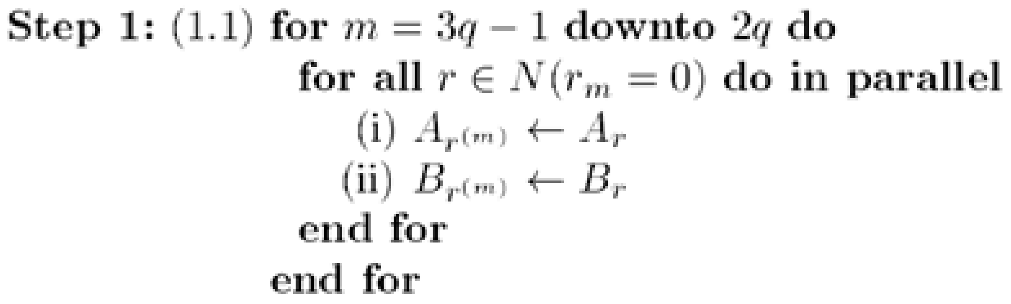
m starts at $3q - 1 = 8$

m ends at $2q = 6$

for m = 7



Parallel Matrix Multiplication on Hypercube (log n distribution example)



```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
            for all $r \in N(r_m = 0)$ do in parallel
                (i) $A_{r^{(m)}} \leftarrow A_r$
                (ii) $B_{r^{(m)}} \leftarrow B_r$
            end for
        end for
```

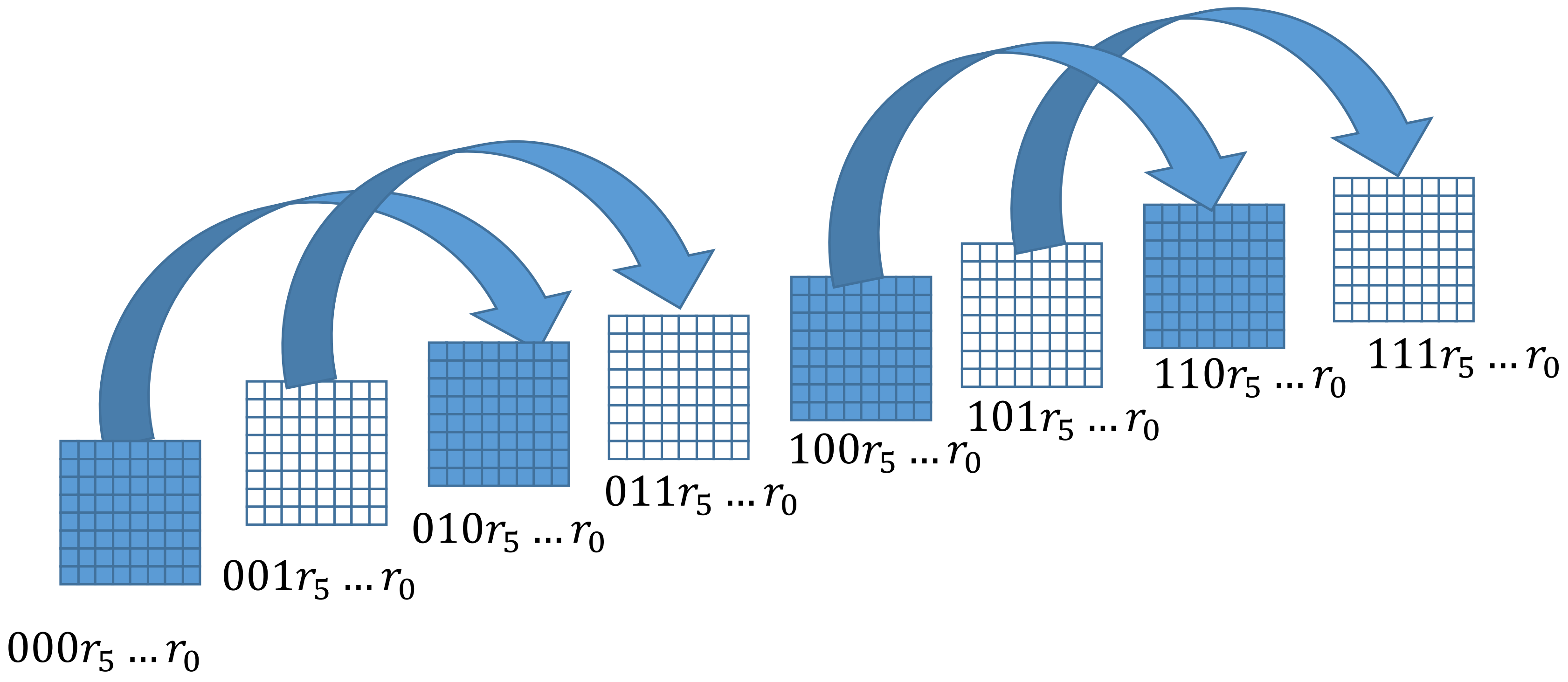
8×8 distribution

q = 3

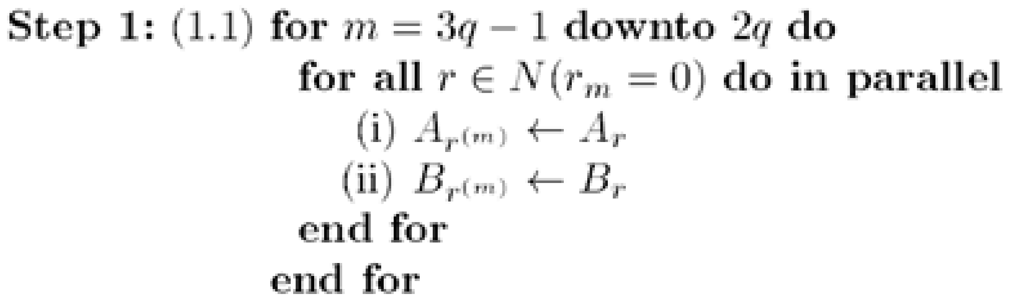
m starts at $3q - 1 = 8$

m ends at $2q = 6$

for m = 7



Parallel Matrix Multiplication on Hypercube (log n distribution example)



```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
            for all $r \in N(r_m = 0)$ do in parallel
                (i) $A_{r(m)} \leftarrow A_r$
                (ii) $B_{r(m)} \leftarrow B_r$
            end for
        end for
```

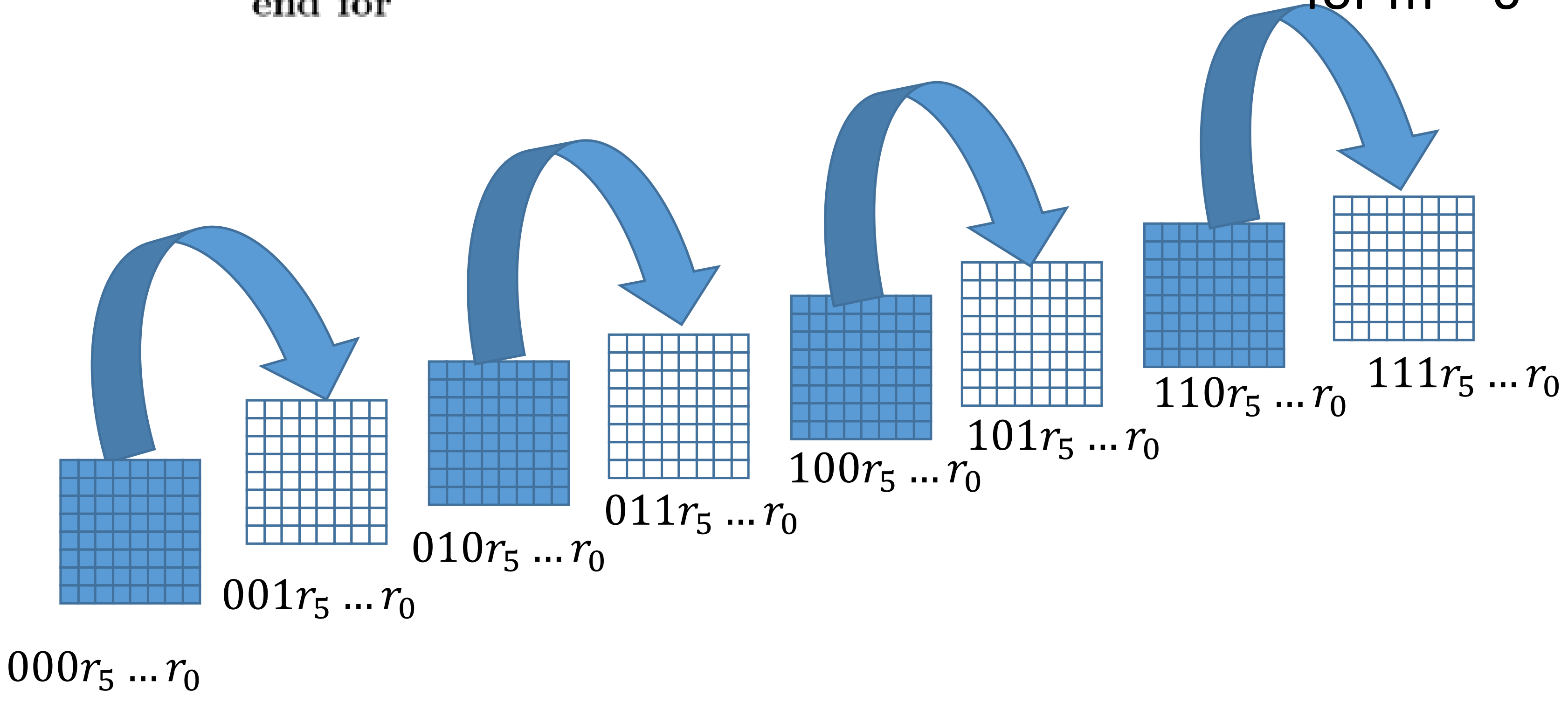
8×8 distribution

q = 3

m starts at $3q - 1 = 8$

m ends at $2q = 6$

for m = 6



Parallel Matrix Multiplication on Hypercube (log n distribution example)

```
Step 1: (1.1) for $m = 3q - 1$ downto $2q$ do
                  for all $r \in N(r_m = 0)$ do in parallel
                     (i) $A_{r^{(m)}} \leftarrow A_r$
                     (ii) $B_{r^{(m)}} \leftarrow B_r$
                  end for
              end for
```

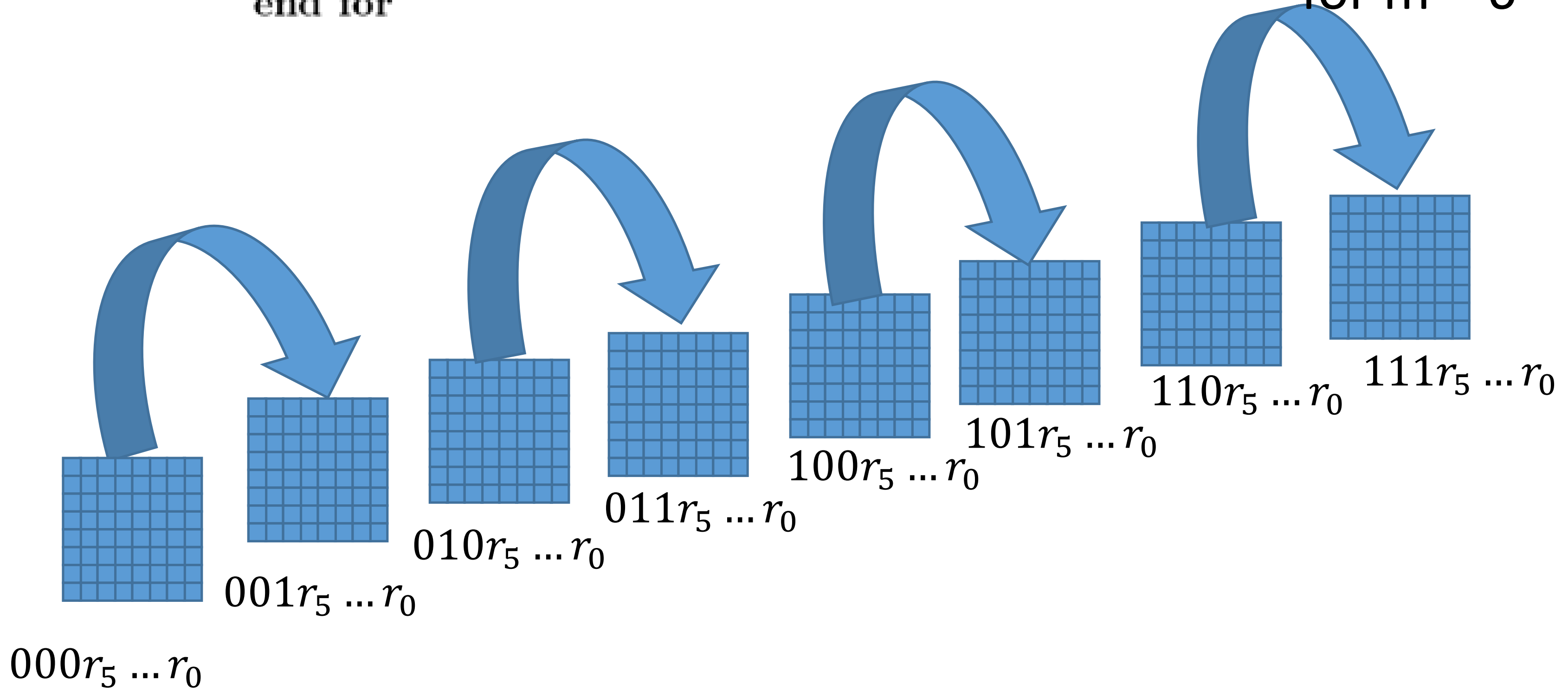
8×8 distribution

q = 3

m starts at $3q - 1 = 8$

m ends at $2q = 6$

for m = 6



Hypercube Matrix Transpose

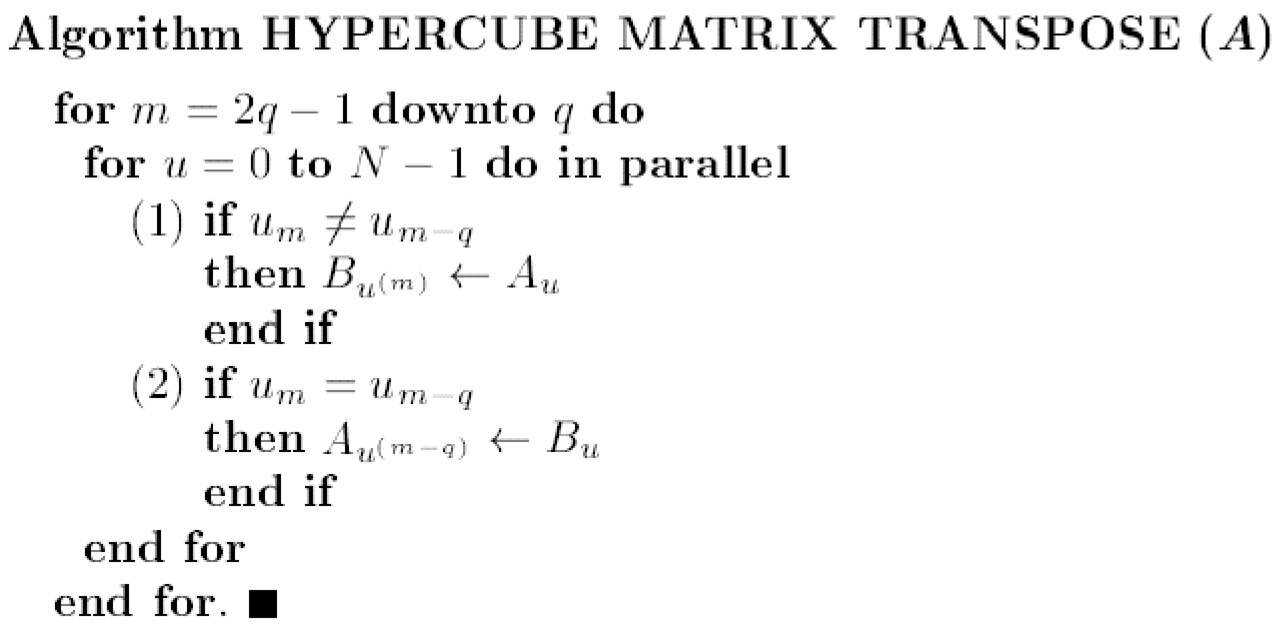
- A is an $n \times n$ matrix whose transpose is A^T.
  - Each element a_{ji}^T of A^T is such that $a_{ji}^T = a_{ij}$.
- The transpose can be computed on a hypercube with $N = n^2 = 2^{2q}$ processors.
- Can visualize the processors as being arranged in an $n \times n$ array in row-major order.
- Processor P_r holds position (i, j), where $r = in + j$ and $0 \leq i, j \leq n - 1$.

Hypercube Matrix Transpose

- Initially, every element a_{ij} of A is in processor P_r in a register A_r, where $r = in + j$. After the matrix transpose algorithm, element a_{ij} of A will be in processor P_s, in register A_s, where $s = jn + i$
- The binary representations of r and s are $r_{2q-1}r_{2q-2} \dots r_q r_{q-1} \dots r_1 r_0$ and $s_{2q-1}s_{2q-2} \dots s_q s_{q-1} \dots s_1 s_0$.
    - $r_{2q-1}r_{2q-2} \dots r_q$ is the binary representation for i.
    - $r_{q-1}r_{q-2} \dots r_1 r_0$ is the binary representation for j.
    - $s_{2q-1}s_{2q-2} \dots s_q$ is the binary representation for j.
    - $s_{q-1}s_{q-2} \dots s_1 s_0$ is the binary representation for i.
- Thus, $r_{2q-1}r_{2q-2} \dots r_q = s_{q-1}s_{q-2} \dots s_1 s_0$
- and $r_{q-1}r_{q-2} \dots r_1 r_0 = s_{2q-1}s_{2q-2} \dots s_q$
- Thus, an element a_{ij} can be routed from P_r to P_s in at most $2q$ steps.

Hypercube Matrix Transpose

- A_u of P_u is assumed to hold initially element a_{kl} of A, where $u = kn + l$.
- When the algorithm is done, A_u holds a_{kl}^T.
- An additional register B_u is used by P_u for routing data sent to it by other processors.



Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
     (1) if $u_m \neq u_{m-q}$
           then $B_{u^{(m)}} \leftarrow A_u$
           end if
     (2) if $u_m = u_{m-q}$
           then $A_{u^{(m-q)}} \leftarrow B_u$
           end if
  end for
end for. ■
```

Hypercube Matrix Transpose

- How it works:
- Suppose the $n \times n$ matrix is subdivided into four $(n/2) \times (n/2)$ submatrices.
- The elements of the bottom left submatrix are swapped with the corresponding elements of the top right submatrix. The other two submatrices are untouched.
- Next, the same step is applied recursively to each of the four $(n/2) \times (n/2)$ matrices. Each of the $(n/2) \times (n/2)$ matrices, are subdivided into four $(n/4) \times (n/4)$ matrices.

Hypercube Matrix Transpose (The idea)

- Example: Transposing a 4×4 matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

- Divide into four $(n/2) \times (n/2)$ or 2×2 submatrices.
- Swap the elements of the top right submatrix with the elements of the bottom left submatrix.

Hypercube Matrix Transpose (The idea)

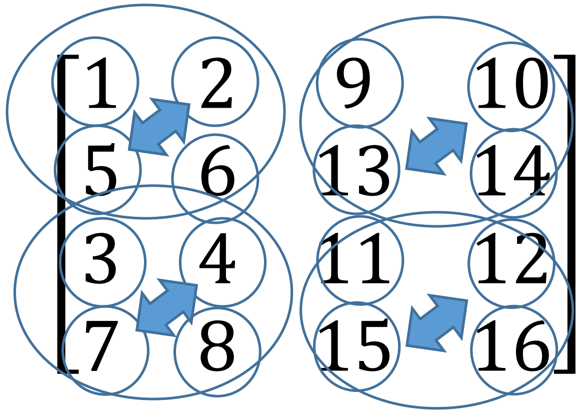
- Example: Transposing a 4×4 matrix

$$\begin{bmatrix} 1 & 2 & 9 & 10 \\ 5 & 6 & 13 & 14 \\ 3 & 4 & 11 & 12 \\ 7 & 8 & 15 & 16 \end{bmatrix}$$

- Divide into four $(n/2) \times (n/2)$ or 2×2 submatrices.
- Swap the elements of the top right submatrix with the elements of the bottom left submatrix.

Hypercube Matrix Transpose (The idea)

- Example: Transposing a 4×4 matrix

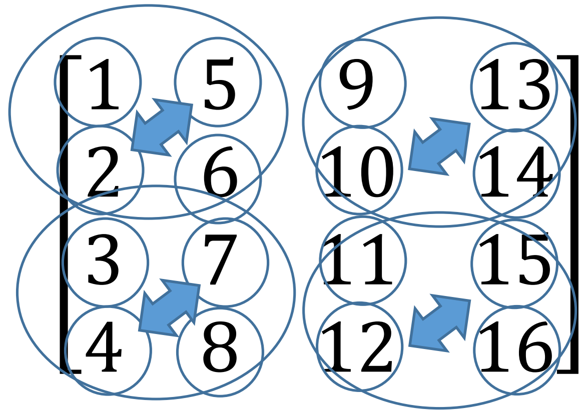


$$\begin{bmatrix} 1 & 2 & 9 & 10 \\ 5 & 6 & 13 & 14 \\ 3 & 4 & 11 & 12 \\ 7 & 8 & 15 & 16 \end{bmatrix}$$

- Recursively divide each submatrix into four submatrices.
- Swap the elements of the top right submatrix with the elements of the bottom left submatrix.

Hypercube Matrix Transpose (The idea)

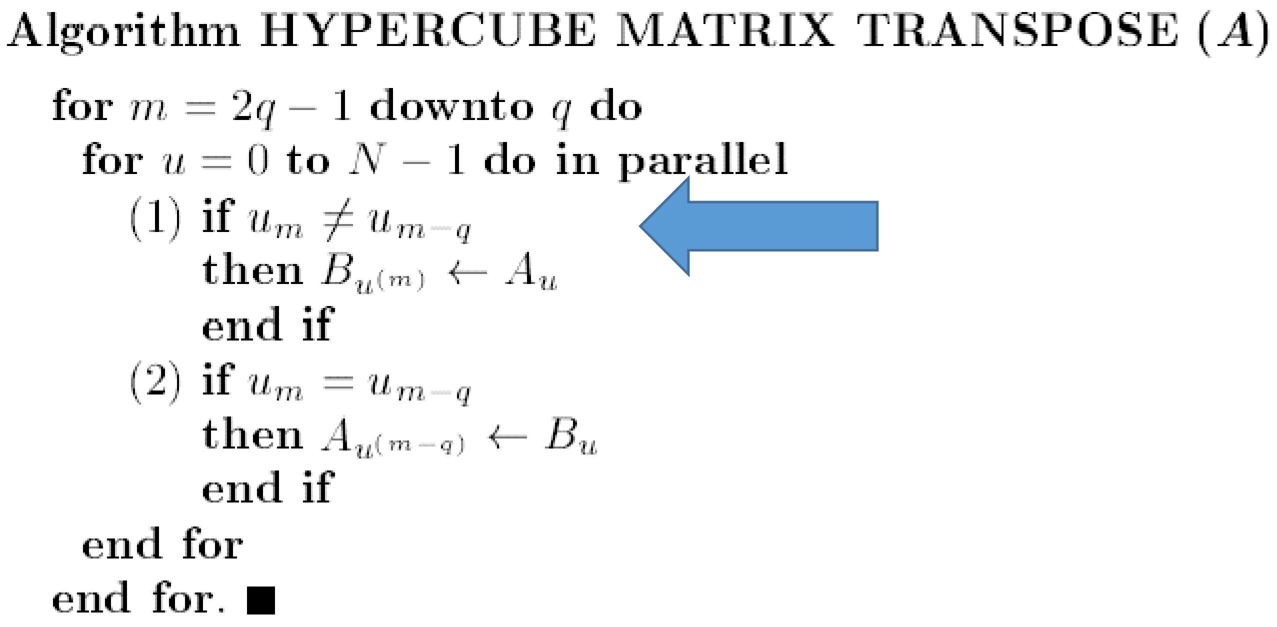
- Example: Transposing a 4×4 matrix



- Recursively divide each submatrix into four submatrices.
- Swap the elements of the top right submatrix with the elements of the bottom left submatrix.

Hypercube Matrix Transpose

- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.



Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
    (1) if $u_m \neq u_{m-q}$
        then $B_{u^{(m)}} \leftarrow A_u$
        end if
    (2) if $u_m = u_{m-q}$
        then $A_{u^{(m-q)}} \leftarrow B_u$
        end if
  end for
end for. ■
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

m=3

m-q=1

0000 0001 0010 0011
0100 0101 0110 0111
1000 1001 1010 1011
1100 1101 1110 1111

Hypercube Matrix Transpose

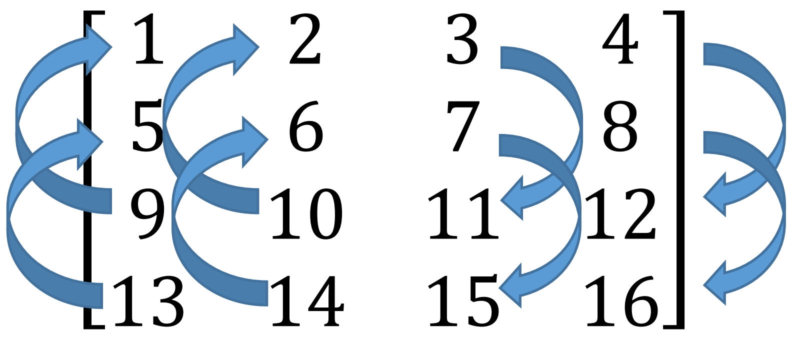
- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.

Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
      (1) if $u_m \neq u_{m-q}$
          then $B_{u^{(m)}} \leftarrow A_u$
          end if
      (2) if $u_m = u_{m-q}$
          then $A_{u^{(m-q)}} \leftarrow B_u$
          end if
  end for
end for. ■
```

m=3

m-q=1



0000 0001 **0010 0011**

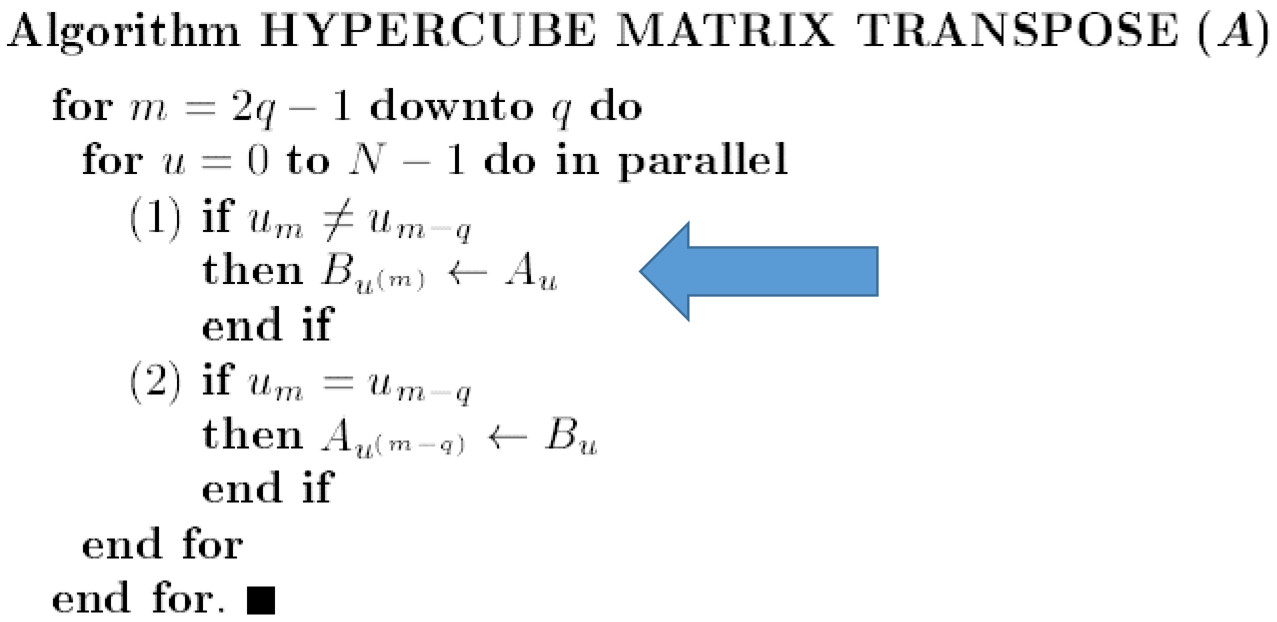
0100 0101 **0110 0111**

1000 1001 1010 1011

1100 1101 1110 1111

Hypercube Matrix Transpose

- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.



Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
    (1) if $u_m \neq u_{m-q}$
        then $B_{u^{(m)}} \leftarrow A_u$
        end if
    (2) if $u_m = u_{m-q}$
        then $A_{u^{(m-q)}} \leftarrow B_u$
        end if
  end for
end for. ■
```

$$\begin{bmatrix} 1,9 & 2,10 & 3 & 4 \\ 5,13 & 6,14 & 7 & 8 \\ 9 & 10 & 11,3 & 12,4 \\ 13 & 14 & 15,7 & 16,8 \end{bmatrix}$$

m=3

m-q=1

0000 0001 **0010 0011**

0100 0101 **0110 0111**

1000 1001 1010 1011

1100 1101 1110 1111

Hypercube Matrix Transpose

- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.

Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
    (1) if $u_m \neq u_{m-q}$
        then $B_{u^{(m)}} \leftarrow A_u$
        end if
    (2) if $u_m = u_{m-q}$
        then $A_{u^{(m-q)}} \leftarrow B_u$
        end if
  end for
end for. ■
```

$$\begin{bmatrix} 1,9 & 2,10 & 3 & 4 \\ 5,13 & 6,14 & 7 & 8 \\ 9 & 10 & 11,3 & 12,4 \\ 13 & 14 & 15,7 & 16,8 \end{bmatrix}$$

m=3

m-q=1

0000 0001 0010 0011
0100 0101 0110 0111
1000 1001 **1010 1011**
1100 1101 **1110 1111**

Hypercube Matrix Transpose

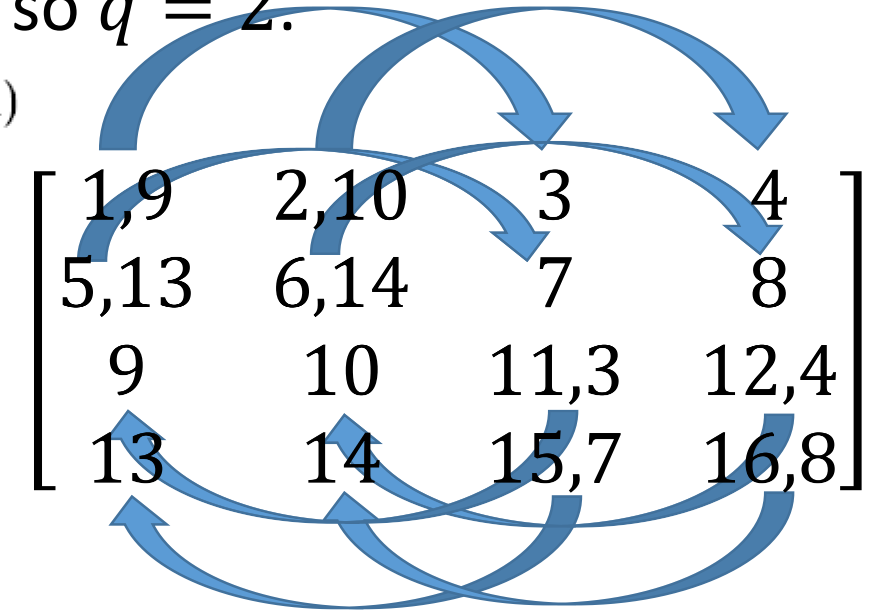
- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.

```
Algorithm HYPERCUBE MATRIX TRANSPOSE ($A$)

  for $m = 2q - 1$ downto $q$ do
    for $u = 0$ to $N - 1$ do in parallel
      (1) if $u_m \neq u_{m-q}$
          then $B_{u^{(m)}} \leftarrow A_u$
          end if
      (2) if $u_m = u_{m-q}$
          then $A_{u^{(m-q)}} \leftarrow B_u$
          end if
    end for
  end for. ■
```

m=3

m-q=1



$$\begin{bmatrix} 1,9 & 2,10 & 3 & 4 \\ 5,13 & 6,14 & 7 & 8 \\ 9 & 10 & 11,3 & 12,4 \\ 13 & 14 & 15,7 & 16,8 \end{bmatrix}$$

0000 0001 0010 0011

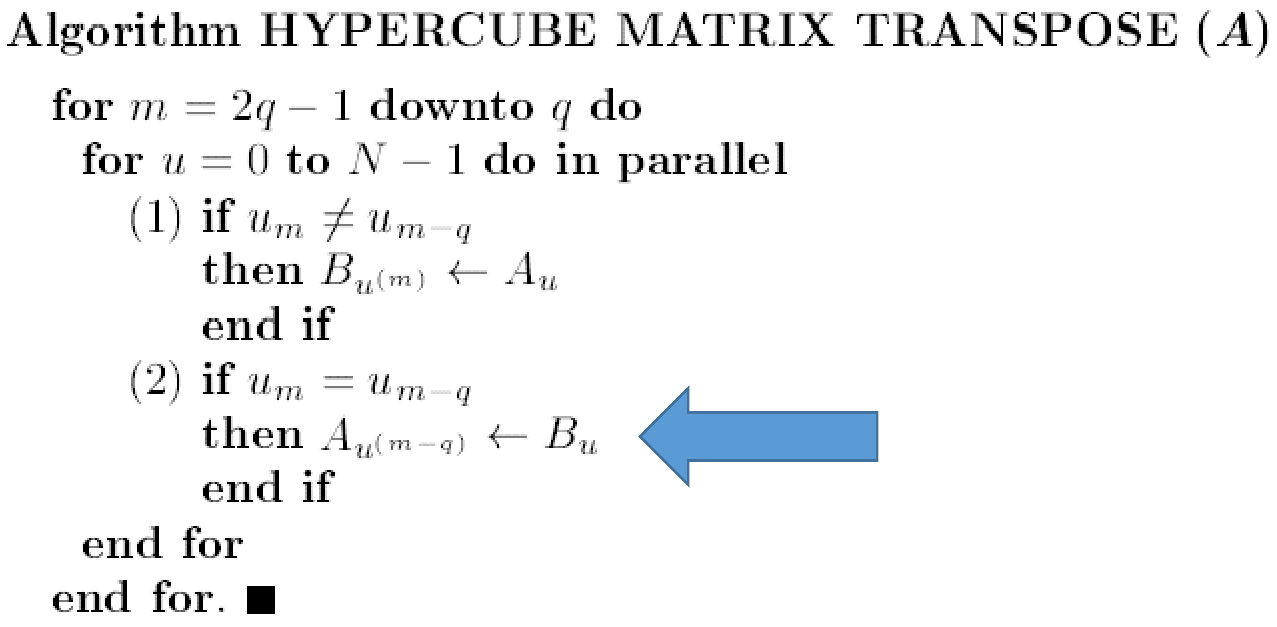
0100 0101 0110 0111

1000 1001 **1010 1011**

1100 1101 **1110 1111**

Hypercube Matrix Transpose

- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.



Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
    (1) if $u_m \neq u_{m-q}$
        then $B_{u^{(m)}} \leftarrow A_u$
        end if
    (2) if $u_m = u_{m-q}$
        then $A_{u^{(m-q)}} \leftarrow B_u$
        end if
  end for
end for. ■
```

$$\begin{bmatrix} 1 & 2 & 9 & 10 \\ 5 & 6 & 13 & 14 \\ 3 & 4 & 11 & 12 \\ 7 & 8 & 15 & 16 \end{bmatrix}$$

m=3

m-q=1

0000 0001 0010 0011
0100 0101 0110 0111
1000 1001 **1010 1011**
1100 1101 **1110 1111**

Hypercube Matrix Transpose

- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.

Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
    (1) if $u_m \neq u_{m-q}$
        then $B_{u^{(m)}} \leftarrow A_u$
        end if
    (2) if $u_m = u_{m-q}$
        then $A_{u^{(m-q)}} \leftarrow B_u$
        end if
  end for
end for. ■
```

$$\begin{bmatrix} 1 & 2 & 9 & 10 \\ 5 & 6 & 13 & 14 \\ 3 & 4 & 11 & 12 \\ 7 & 8 & 15 & 16 \end{bmatrix}$$

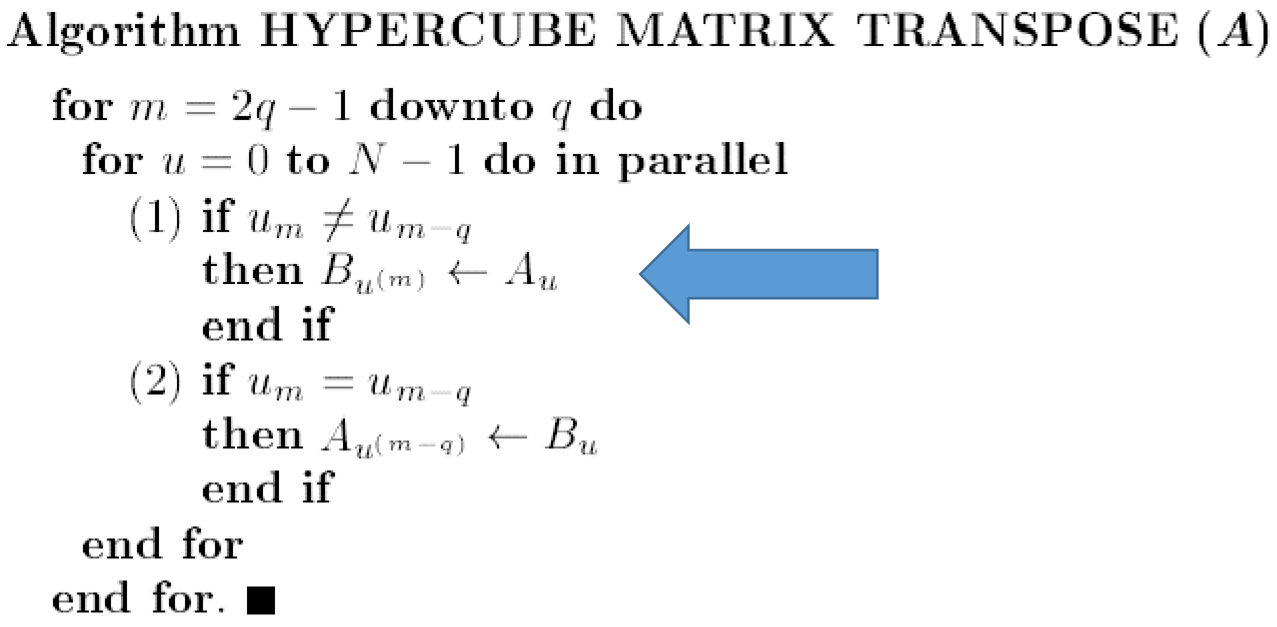
m=2

m-q=0

0000 **0001** 0010 **0011**
0100 0101 **0110** 0111
1000 **1001** 1010 **1011**
1100 1101 **1110** 1111

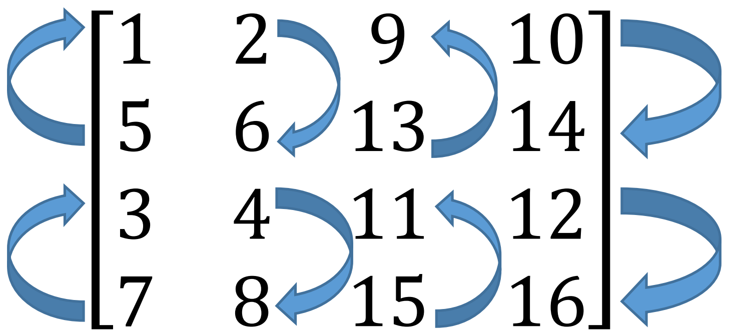
Hypercube Matrix Transpose

- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.



Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
    (1) if $u_m \neq u_{m-q}$
        then $B_{u^{(m)}} \leftarrow A_u$
        end if
    (2) if $u_m = u_{m-q}$
        then $A_{u^{(m-q)}} \leftarrow B_u$
        end if
  end for
end for. ■
```



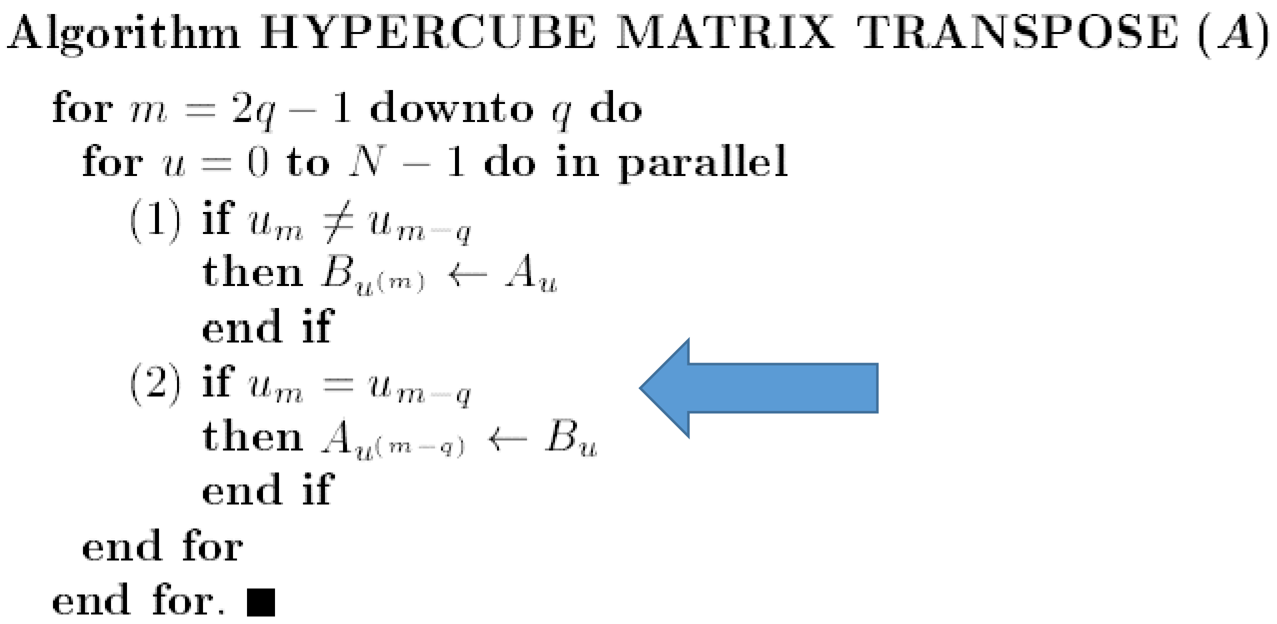
m=2

m-q=0

0000	**0001**	0010	**0011**
0100	0101	**0110**	0111
1000	**1001**	1010	**1011**
1100	1101	**1110**	1111

Hypercube Matrix Transpose

- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.



Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

```
for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
    (1) if $u_m \neq u_{m-q}$
        then $B_{u^{(m)}} \leftarrow A_u$
        end if
    (2) if $u_m = u_{m-q}$
        then $A_{u^{(m-q)}} \leftarrow B_u$
        end if
  end for
end for. ■
```

$$\begin{bmatrix} 1,5 & 2 & 9,13 & 10 \\ 5 & 6,2 & 13 & 14,10 \\ 3,7 & 4 & 11,15 & 12 \\ 7 & 8,4 & 15 & 16,12 \end{bmatrix}$$

m=2

m-q=0

0000 0001 0010 0011

0100 0101 0110 0111

1000 1001 1010 1011

1100 1101 1110 1111

Hypercube Matrix Transpose

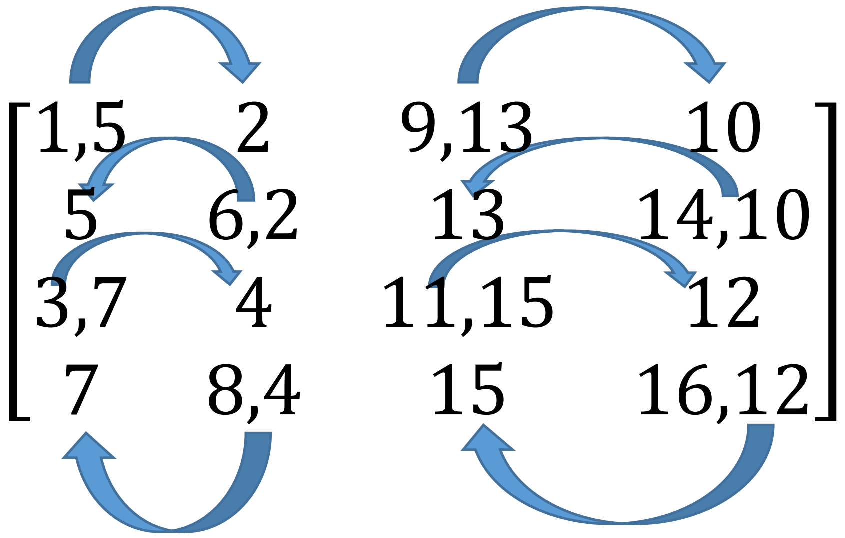
- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.

```
Algorithm HYPERCUBE MATRIX TRANSPOSE (A)

  for $m = 2q - 1$ downto $q$ do
    for $u = 0$ to $N - 1$ do in parallel
      (1) if $u_m \neq u_{m-q}$
          then $B_{u^{(m)}} \leftarrow A_u$
          end if
      (2) if $u_m = u_{m-q}$
          then $A_{u^{(m-q)}} \leftarrow B_u$
          end if
    end for
  end for. ■
```

m=2

m-q=0

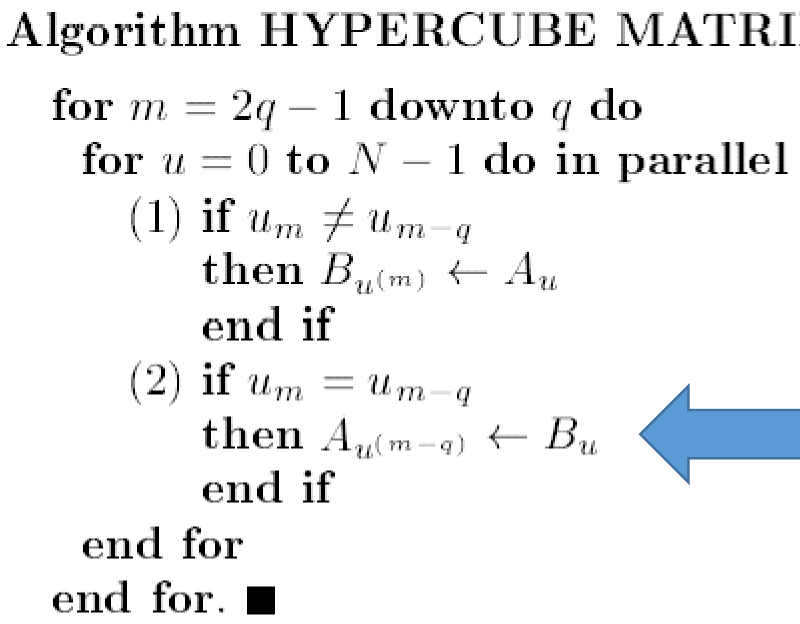


$$\begin{bmatrix} 1,5 & 2 & 9,13 & 10 \\ 5 & 6,2 & 13 & 14,10 \\ 3,7 & 4 & 11,15 & 12 \\ 7 & 8,4 & 15 & 16,12 \end{bmatrix}$$

0000 0001 **0010** 0011
0100 **0101** 0110 **0111**
1000 1001 **1010** 1011
1100 **1101** 1110 **1111**

Hypercube Matrix Transpose

- With the algorithm: $N = n^2 = 2^{2q}$, so $q = 2$.



```
Algorithm HYPERCUBE MATRIX TRANSPOSE ($A$)

for $m = 2q - 1$ downto $q$ do
  for $u = 0$ to $N - 1$ do in parallel
    (1) if $u_m \neq u_{m-q}$
          then $B_{u^{(m)}} \leftarrow A_u$
          end if
    (2) if $u_m = u_{m-q}$
          then $A_{u^{(m-q)}} \leftarrow B_u$
          end if
  end for
end for. ■
```

m=2

m-q=0

$$\begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}$$

0000 0001 **0010** 0011
0100 **0101** 0110 **0111**
1000 1001 **1010** 1011
1100 **1101** 1110 **1111**

Hypercube Matrix Transpose

```
Algorithm HYPERCUBE MATRIX TRANSPOSE ($A$)
  for $m = 2q - 1$ downto $q$ do
    for $u = 0$ to $N - 1$ do in parallel
      (1) if $u_m \neq u_{m-q}$
          then $B_{u^{(m)}} \leftarrow A_u$
          end if
      (2) if $u_m = u_{m-q}$
          then $A_{u^{(m-q)}} \leftarrow B_u$
          end if
    end for
  end for. ■
```

<u>Analysis</u>:

There are q constant time iterations. The run time is $O(q) = O(\log n)$.

$P_n = n^2$, so the cost is $n^2 \cdot \log n$. Not cost optimal because the RAM algorithm only needs $n(n-1)/2$ operations.

References

[1] E. Dekel, D. Nassimi and S. Sahni, "Parallel Matrix and Graph Algorithms," *SIAM Journal on Computing,* vol. 10, no. 4, pp. 657-819, 1981.

[2] S. G. Akl, Parallel Computation: Models and Methods, Prentice Hall, 1997.