CSE 638: Advanced Algorithms

Lectures 16 & 17
(Analyzing I/O and Cache Performance)

Rezaul A. Chowdhury
Department of Computer Science
SUNY Stony Brook
Spring 2013

Memory: Fast, Large & Cheap!

For efficient computation we need

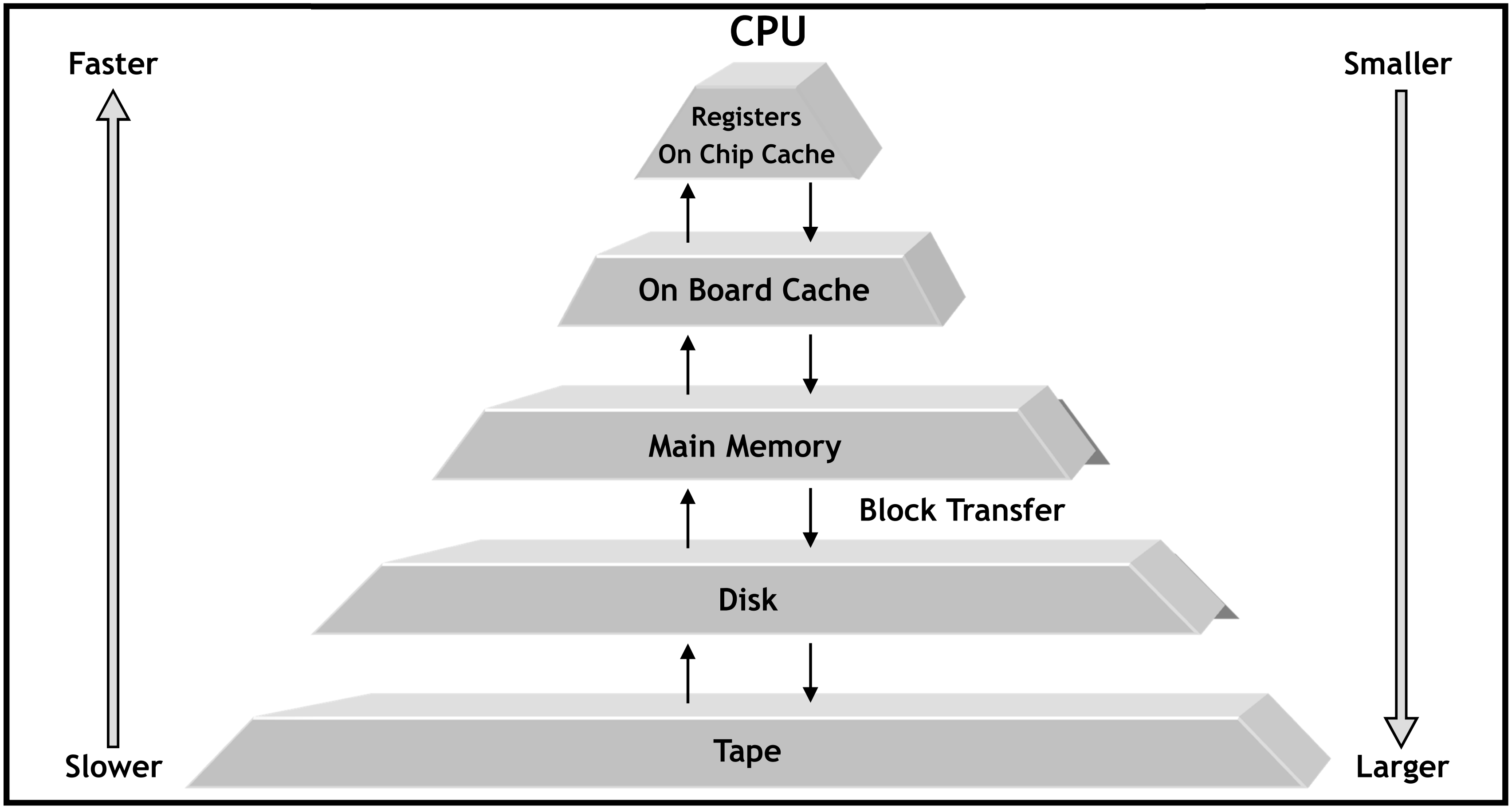
- fast processors
- fast and large (but not so expensive) memory

But memory cannot be cheap, large and fast at the same time, because of

- finite signal speed
- lack of space to put enough connecting wires

A reasonable compromise is to use a *memory hierarchy*.

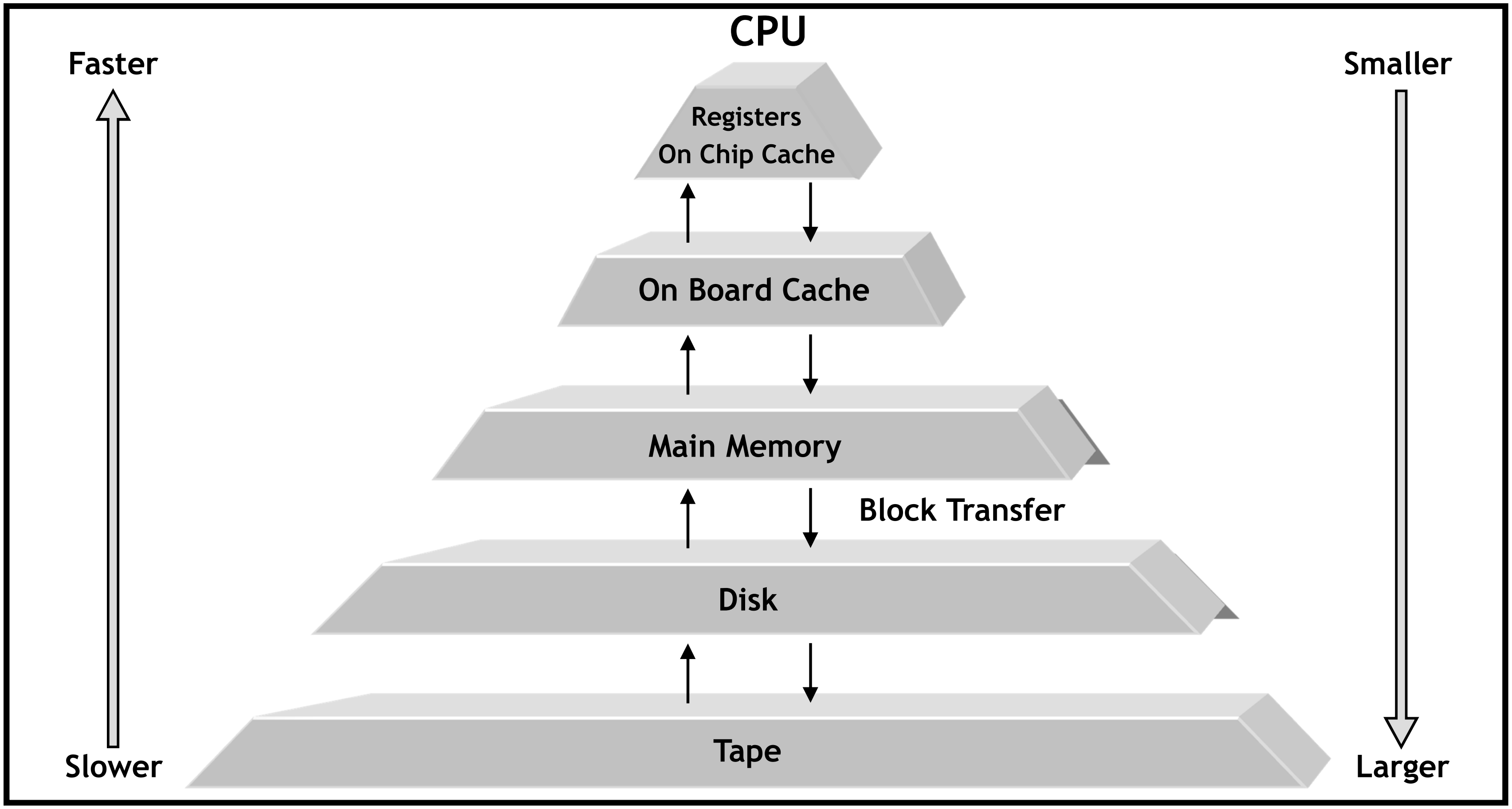
The Memory Hierarchy



A *memory hierarchy* is

- almost as fast as its fastest level
- almost as large as its largest level
- inexpensive

The Memory Hierarchy



To perform well on a memory hierarchy algorithms must have <u>high locality</u> in their memory access patterns.

Locality of Reference

Spatial Locality: When a block of data is brought into the cache it should contain as much useful data as possible.

Temporal Locality: Once a data point is in the cache as much useful work as possible should be done on it before evicting it from the cache.

CPU-bound vs. Memory-bound Algorithms

The Op-Space Ratio: Ratio of the number of operations performed by an algorithm to the amount of space (input + output) it uses.

Intuitively, this gives an upper bound on the average number of operations performed for every memory location accessed.

CPU-bound Algorithm:

- high op-space ratio
- more time spent in computing than transferring data
- a faster CPU results in a faster running time

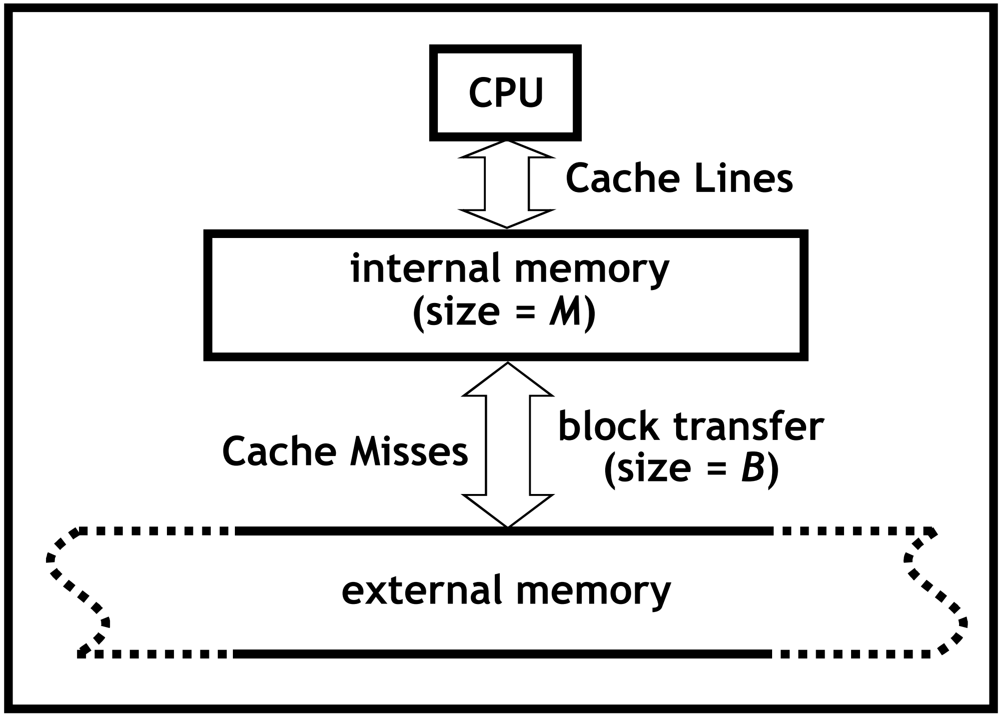
Memory-bound Algorithm:

- low op-space ratio
- more time spent in transferring data than computing
- a faster memory system leads to a faster running time

The Two-level I/O Model

The *two-level I/O model* [Aggarwal & Vitter, CACM'88] consists of:

- an *internal memory* of size M
- an arbitrarily large *external memory* partitioned into blocks of size B.



I/O complexity of an algorithm

= number of blocks transferred between these two levels

Basic I/O complexities: $scan(N) = \Theta\left(\frac{N}{B}\right)$ and $sort(N) = \Theta\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{B}\right)$

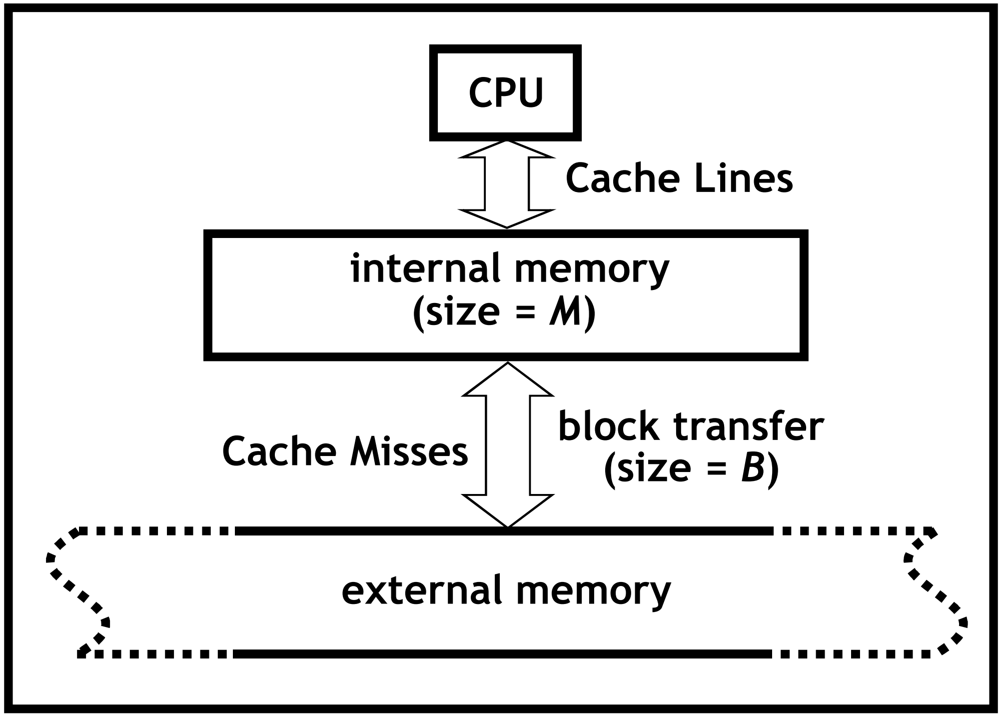
Algorithms often crucially depend on the knowledge of M and B

$\Rightarrow$ algorithms do not adapt well when M or B changes

The Ideal-Cache Model

The *ideal-cache model* [Frigo et al., FOCS'99] is an extension of the I/O model with the following constraint:

algorithms are not allowed to use knowledge of M and B.



Consequences of this extension

- algorithms can simultaneously adapt to all levels of a multi-level memory hierarchy
- algorithms become more flexible and portable

Algorithms for this model are known as *cache-oblivious algorithms*.

The Ideal-Cache Model: Assumptions

The model makes the following assumptions:

- Optimal offline cache replacement policy
- Exactly two levels of memory
- Automatic replacement & full associativity

The Ideal-Cache Model: Assumptions

The model makes the following assumptions:

- Optimal offline cache replacement policy
  - LRU & FIFO allow for a constant factor approximation of optimal [Sleator & Tarjan, JACM'85]
- Exactly two levels of memory
- Automatic replacement & full associativity

The Ideal-Cache Model: Assumptions

The model makes the following assumptions:

- ❑ Optimal offline cache replacement policy
- ❑ Exactly two levels of memory
  - – can be effectively removed by making several reasonable assumptions about the memory hierarchy [Frigo et al., FOCS’99]
- ❑ Automatic replacement & full associativity

The Ideal-Cache Model: Assumptions

The model makes the following assumptions:

- Optimal offline cache replacement policy
- Exactly two levels of memory
- Automatic replacement & full associativity
  - in practice, cache replacement is automatic
    (by OS or hardware)
  - fully associative LRU caches can be simulated in software with only a constant factor loss in expected performance [Frigo et al., FOCS’99]

The Ideal-Cache Model: Assumptions

The model makes the following assumptions:

- Optimal offline cache replacement policy
- Exactly two levels of memory
- Automatic replacement & full associativity

Often makes the following assumption, too:

- $M = \Omega\left(B^2\right)$, i.e., the cache is *tall*

The Ideal-Cache Model: Assumptions

The model makes the following assumptions:

- Optimal offline cache replacement policy
- Exactly two levels of memory
- Automatic replacement & full associativity

Often makes the following assumption, too:

- $M = \Omega\left(B^2\right)$, i.e., the cache is *tall*
  - most practical caches are tall

The Ideal-Cache Model: I/O Bounds

Cache-oblivious vs. cache-aware bounds:

- Basic I/O bounds (same as the cache-aware bounds):
  - $scan(N) = \Theta\left(\frac{N}{B}\right)$
  - $sort(N) = \Theta\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{B}\right)$
- Most cache-oblivious results match the I/O bounds of their cache-aware counterparts
- There are few exceptions; e.g., no cache-oblivious solution to the *permutation* problem can match cache-aware I/O bounds [Brodal & Fagerberg, STOC'03]

Some Known Cache Aware / Oblivious Results

Problem	Cache-Aware Results	Cache-Oblivious Results
Array Scanning (*scan(N)*)	$O\left(\frac{N}{B}\right)$	$O\left(\frac{N}{B}\right)$
Sorting (*sort(N)*)	$O\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{B}\right)$	$O\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{B}\right)$
Selection	$O(scan(N))$	$O(scan(N))$
B-Trees [Am] (*Insert*, *Delete*)	$O\left(\log_B\frac{N}{B}\right)$	$O\left(\log_B\frac{N}{B}\right)$
Priority Queue [Am] (*Insert*, *Weak Delete*, *Delete-Min*)	$O\left(\frac{1}{B}\log_{\frac{M}{B}}\frac{N}{B}\right)$	$O\left(\frac{1}{B}\log_{\frac{M}{B}}\frac{N}{B}\right)$
Matrix Multiplication	$O\left(\frac{N^3}{B\sqrt{M}}\right)$	$O\left(\frac{N^3}{B\sqrt{M}}\right)$
Sequence Alignment	$O\left(\frac{N^2}{BM}\right)$	$O\left(\frac{N^2}{BM}\right)$
Single Source Shortest Paths	$O\left(\left(V+\frac{E}{B}\right)\cdot\log_2\frac{V}{B}\right)$	$O\left(\left(V+\frac{E}{B}\right)\cdot\log_2\frac{V}{B}\right)$
Minimum Spanning Forest	$O\left(\min\left(sort(E)\log_2\log_2 V,\ V+sort(E)\right)\right)$	$O\left(\min\left(sort(E)\log_2\log_2\frac{VB}{E},\ V+sort(E)\right)\right)$

Table 1: N = #elements, V = #vertices, E = #edges, Am = Amortized.

Matrix
Multiplication

Iterative Matrix Multiplication

$$z_{ij} = \sum_{k=1}^{n} x_{ik} y_{kj}$$

$$\begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nn} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} \times \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{bmatrix}$$

Iter-MM (X, Y, Z, n)

1. *for* $i \leftarrow 1$ *to* n *do*
2. *for* $j \leftarrow 1$ *to* n *do*
3. *for* $k \leftarrow 1$ *to* n *do*
4. $z_{ij} \leftarrow z_{ij} + x_{ik} \times y_{kj}$

Iterative Matrix Multiplication

Iter-MM (X, Y, Z, n)

```
1. for i ← 1 to n do
2.     for j ← 1 to n do
3.         for k ← 1 to n do
4.             z_ij ← z_ij + x_ik × y_kj
```

store in **row-major** order (for X)

store in **row-major** order (for Y)

$$\begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & \boxed{z_{22}} & \cdots & z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nn} \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix} \times \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{pmatrix}$$

Each iteration of the *for* loop in line 3 incurs $O(n)$ cache misses.

I/O-complexity of *Iter-MM*, $Q(n) = O(n^3)$

Iterative Matrix Multiplication

Iter-MM (X, Y, Z, n)

1. *for* $i \leftarrow 1$ *to* n *do*
2. *for* $j \leftarrow 1$ *to* n *do*
3. *for* $k \leftarrow 1$ *to* n *do*
4. $z_{ij} \leftarrow z_{ij} + x_{ik} \times y_{kj}$

store in **row-major** order

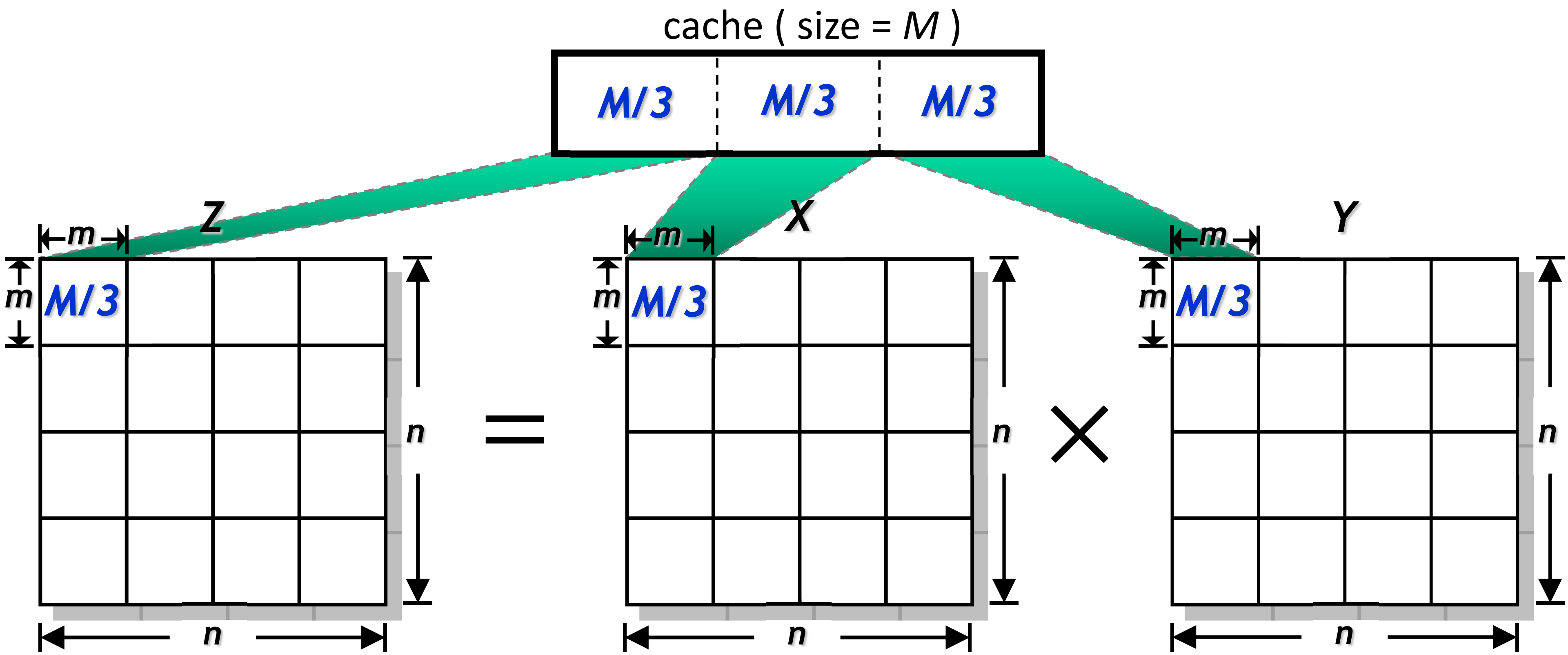
store in **column-major** order

$$\begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & \boxed{z_{22}} & \cdots & z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nn} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} \times \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{bmatrix}$$

Each iteration of the *for* loop in line 3 incurs $O\left(1 + \frac{n}{B}\right)$ cache misses.

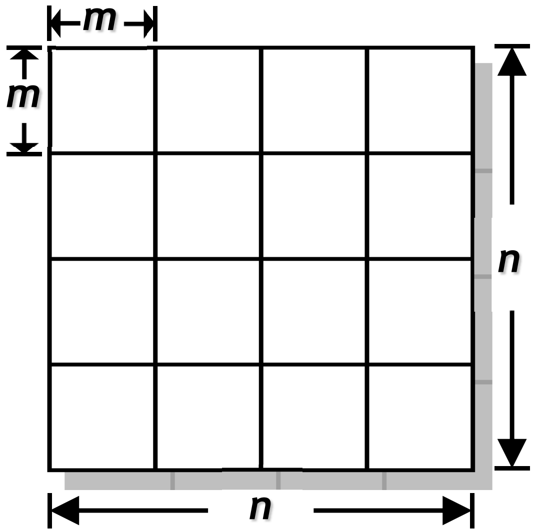
I/O-complexity of *Iter-MM*, $Q(n) = O\left(n^2\left(1 + \frac{n}{B}\right)\right) = O\left(\frac{n^3}{B} + n^2\right)$

Block Matrix Multiplication




```
Block-MM ( X, Y, Z, n )
    for i ← 1 to n / m do
        for j ← 1 to n / m do
            for k ← 1 to n / m do
                Iter-MM ( X_ik, Y_kj, Z_ij )
```

Block Matrix Multiplication




```
Block-MM ( X, Y, Z, n )
1.  for i ← 1 to n / m do
2.      for j ← 1 to n / m do
3.          for k ← 1 to n / m do
4.              Iter-MM ( X_ik, Y_kj, Z_ij )
```

Choose $m = \sqrt{M/3}$, so that X_{ik}, Y_{kj} and Z_{ij} just fit into the cache.

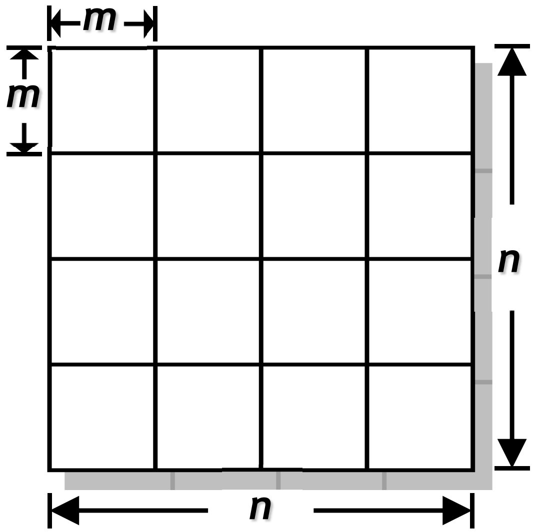
Then line 4 incurs $\Theta\left(m\left(1+\frac{m}{B}\right)\right)$ cache misses.

I/O-complexity of *Block-MM* [assuming a *tall cache*, i.e., $M = \Omega(B^2)$]

$$= \Theta\left(\left(\frac{n}{m}\right)^3\left(m+\frac{m^2}{B}\right)\right) = \Theta\left(\frac{n^3}{m^2}+\frac{n^3}{Bm}\right) = \Theta\left(\frac{n^3}{M}+\frac{n^3}{B\sqrt{M}}\right) = \Theta\left(\frac{n^3}{B\sqrt{M}}\right)$$

(Optimal: Hong & Kung, STOC'81)

Block Matrix Multiplication




```
Block-MM( X, Y, Z, n )
1.  for i ← 1 to n / m do
2.      for j ← 1 to n / m do
3.          for k ← 1 to n / m do
4.              Iter-MM( X_ik, Y_kj, Z_ij )
```

Choose $m = \sqrt{M/2}$ so that X, Y and Z just fit into the cache.

Optimal for any algorithm that performs the operations given by the following definition of matrix multiplication:

$$z_{ij} = \sum_{k=1}^{n} x_{ik} y_{kj}$$

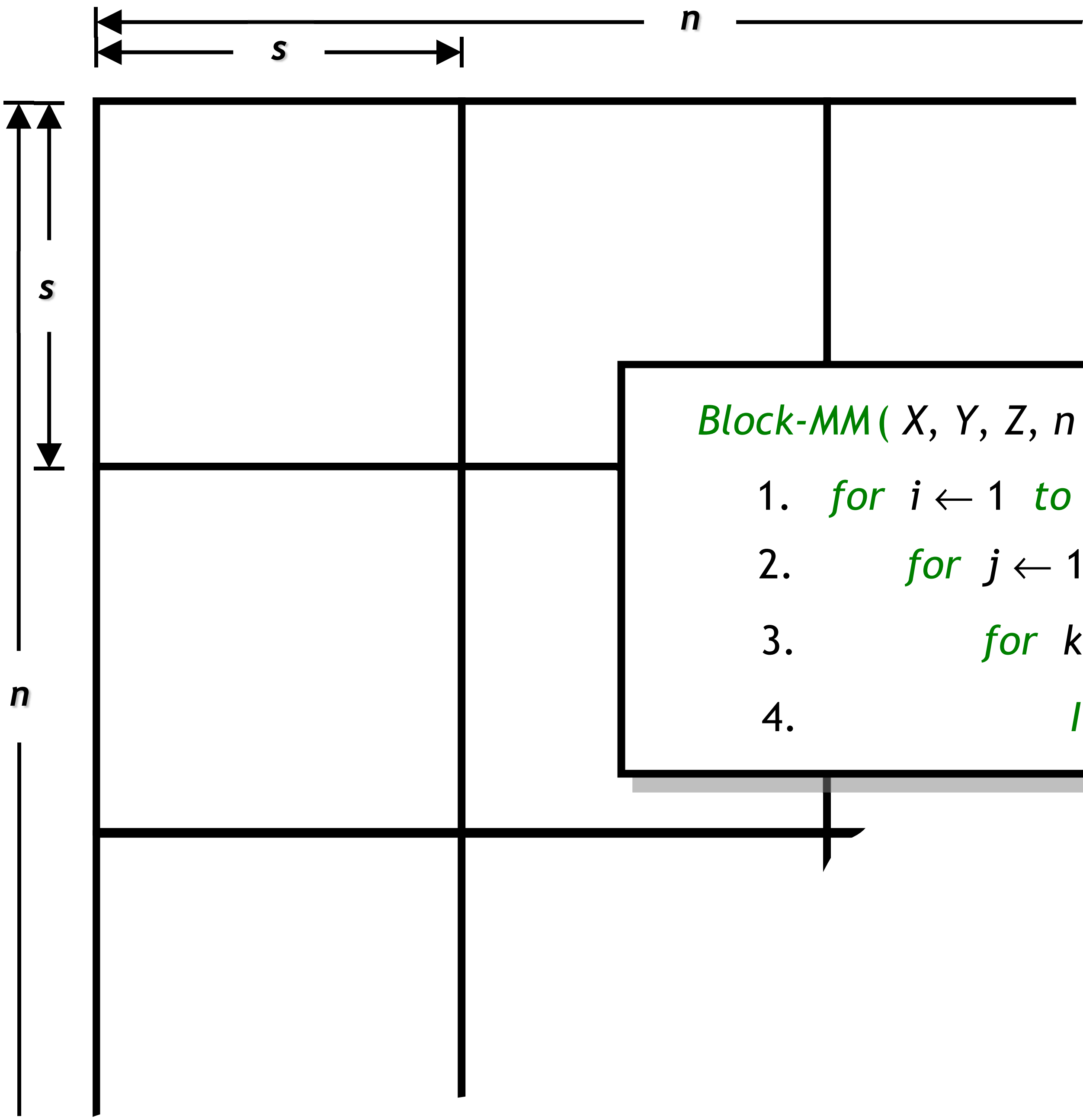
The … ses.

I/O- … *cache*, i.e., $M = \Omega(B^2)$]

$$= \Theta\left(\left(\frac{n}{m}\right)^3\left(m + \frac{m^2}{B}\right)\right) = \Theta\left(\frac{n^3}{m^2} + \frac{n^3}{Bm}\right) = \Theta\left(\frac{n^3}{M} + \frac{n^3}{B\sqrt{M}}\right) = \Theta\left(\frac{n^3}{B\sqrt{M}}\right)$$

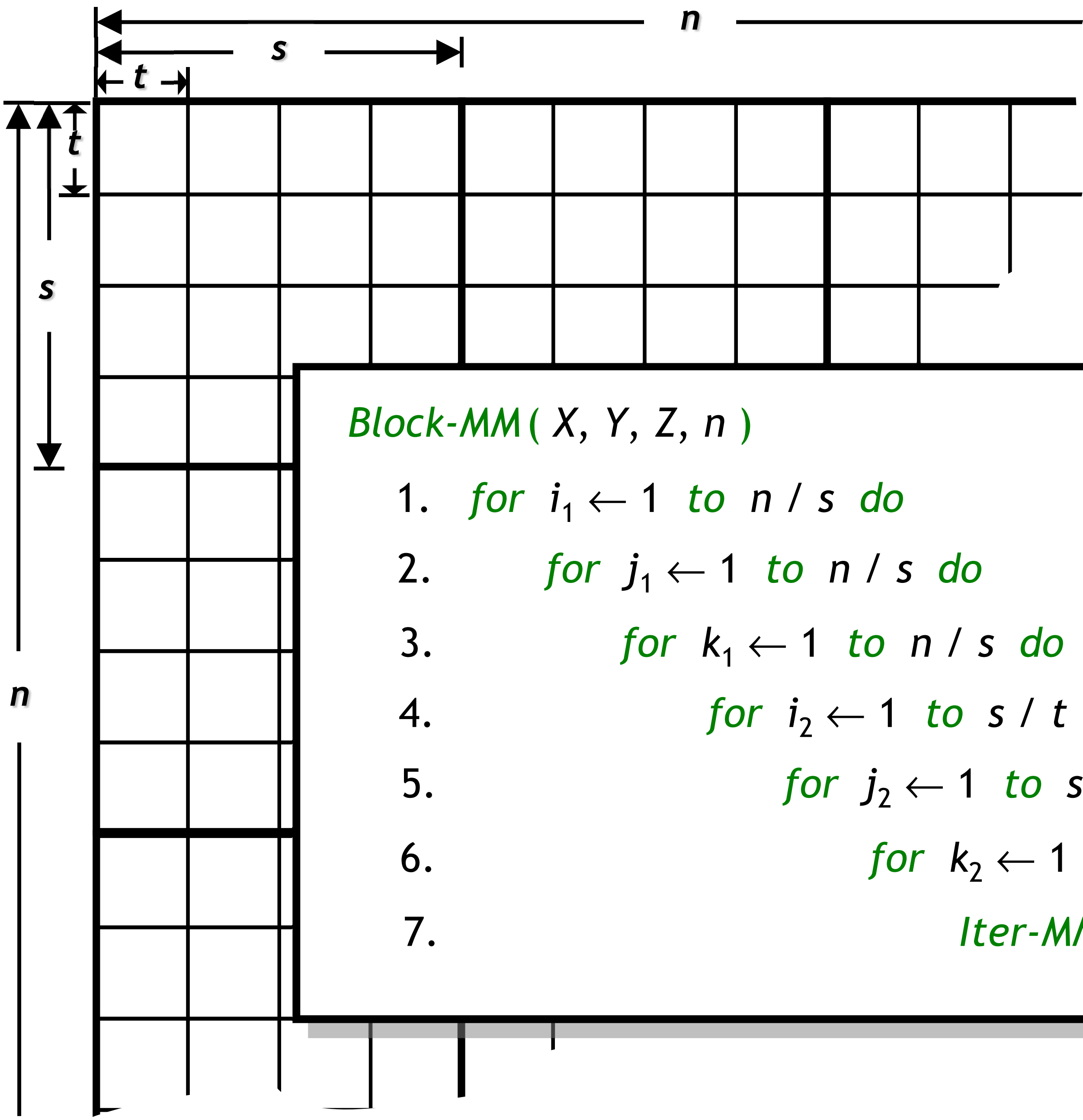
(Optimal: Hong & Kung, STOC'81)

Multiple Levels of Cache




```
Block-MM( X, Y, Z, n )
    for i ← 1 to n / s do
        for j ← 1 to n / s do
            for k ← 1 to n / s do
                Iter-MM( X_ik, Y_kj, Z_ij, s )
```

Multiple Levels of Cache



Block-MM (X, Y, Z, n)

for $i_1 \leftarrow 1$ *to* n / s *do*

  for $j_1 \leftarrow 1$ *to* n / s *do*

    for $k_1 \leftarrow 1$ *to* n / s *do*

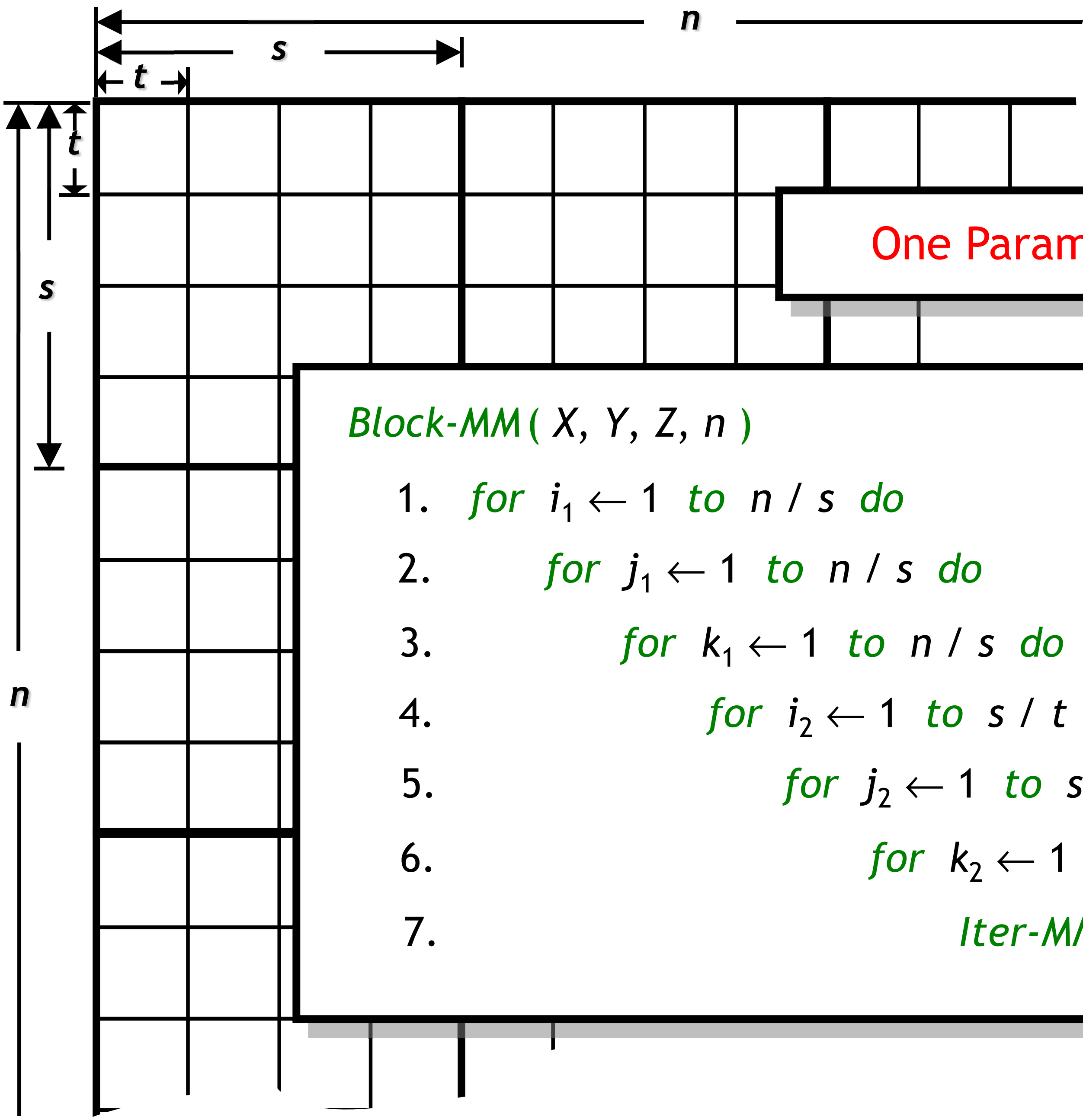
      for $i_2 \leftarrow 1$ *to* s / t *do*

        for $j_2 \leftarrow 1$ *to* s / t *do*

          for $k_2 \leftarrow 1$ *to* s / t *do*

            Iter-MM ($(X_{i_1 k_1})_{i_2 k_2}$, $(Y_{k_1 j_1})_{k_2 j_2}$, $(X_{i_1 j_1})_{i_2 j_2}$, t)

Multiple Levels of Cache



One Parameter Per Caching Level!

Block-MM(X, Y, Z, n)

for $i_1 \leftarrow 1$ *to* n / s *do*

    for $j_1 \leftarrow 1$ *to* n / s *do*

        for $k_1 \leftarrow 1$ *to* n / s *do*

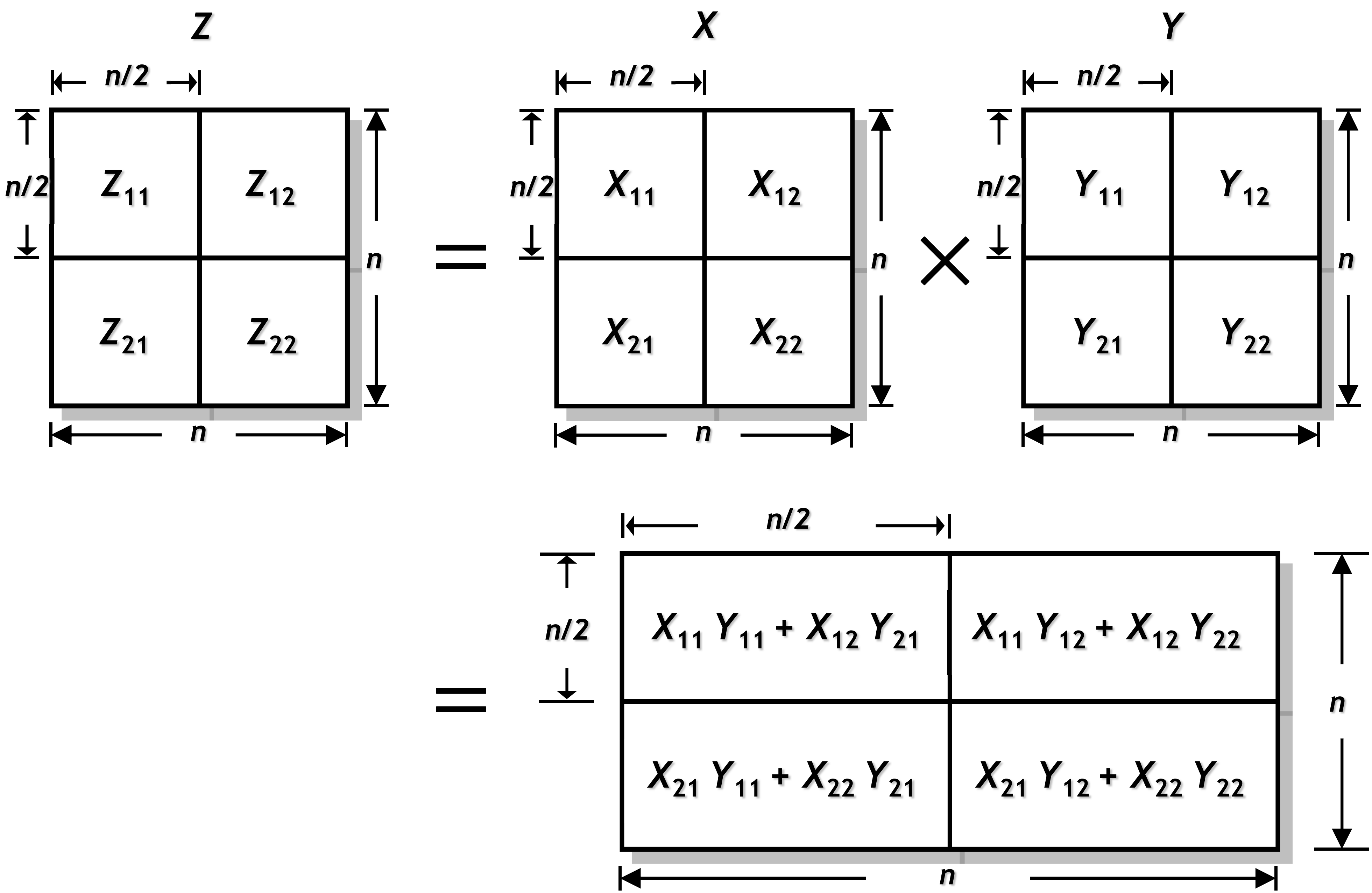
            for $i_2 \leftarrow 1$ *to* s / t *do*

                for $j_2 \leftarrow 1$ *to* s / t *do*

                    for $k_2 \leftarrow 1$ *to* s / t *do*

                        Iter-MM($(X_{i_1 k_1})_{i_2 k_2}, (Y_{k_1 j_1})_{k_2 j_2}, (X_{i_1 j_1})_{i_2 j_2}, t$)

Recursive Matrix Multiplication



Recursive Matrix Multiplication

$$\begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} = \begin{bmatrix} X_{11}Y_{11} + X_{12}Y_{21} & X_{11}Y_{12} + X_{12}Y_{22} \\ X_{21}Y_{11} + X_{22}Y_{21} & X_{21}Y_{12} + X_{22}Y_{22} \end{bmatrix}$$

Rec-MM (Z, X, Y)

1. *if* $Z \equiv 1 \times 1$ matrix *then* $Z \leftarrow Z + X \cdot Y$
2. *else*
3. *Rec-MM* (Z_{11}, X_{11}, Y_{11}), *Rec-MM* (Z_{11}, X_{12}, Y_{21})
4. *Rec-MM* (Z_{12}, X_{12}, Y_{12}), *Rec-MM* (Z_{12}, X_{12}, Y_{22})
5. *Rec-MM* (Z_{21}, X_{21}, Y_{11}), *Rec-MM* (Z_{21}, X_{22}, Y_{21})
6. *Rec-MM* (Z_{22}, X_{21}, Y_{12}), *Rec-MM* (Z_{22}, X_{22}, Y_{22})

Recursive Matrix Multiplication

Rec-MM (Z, X, Y)

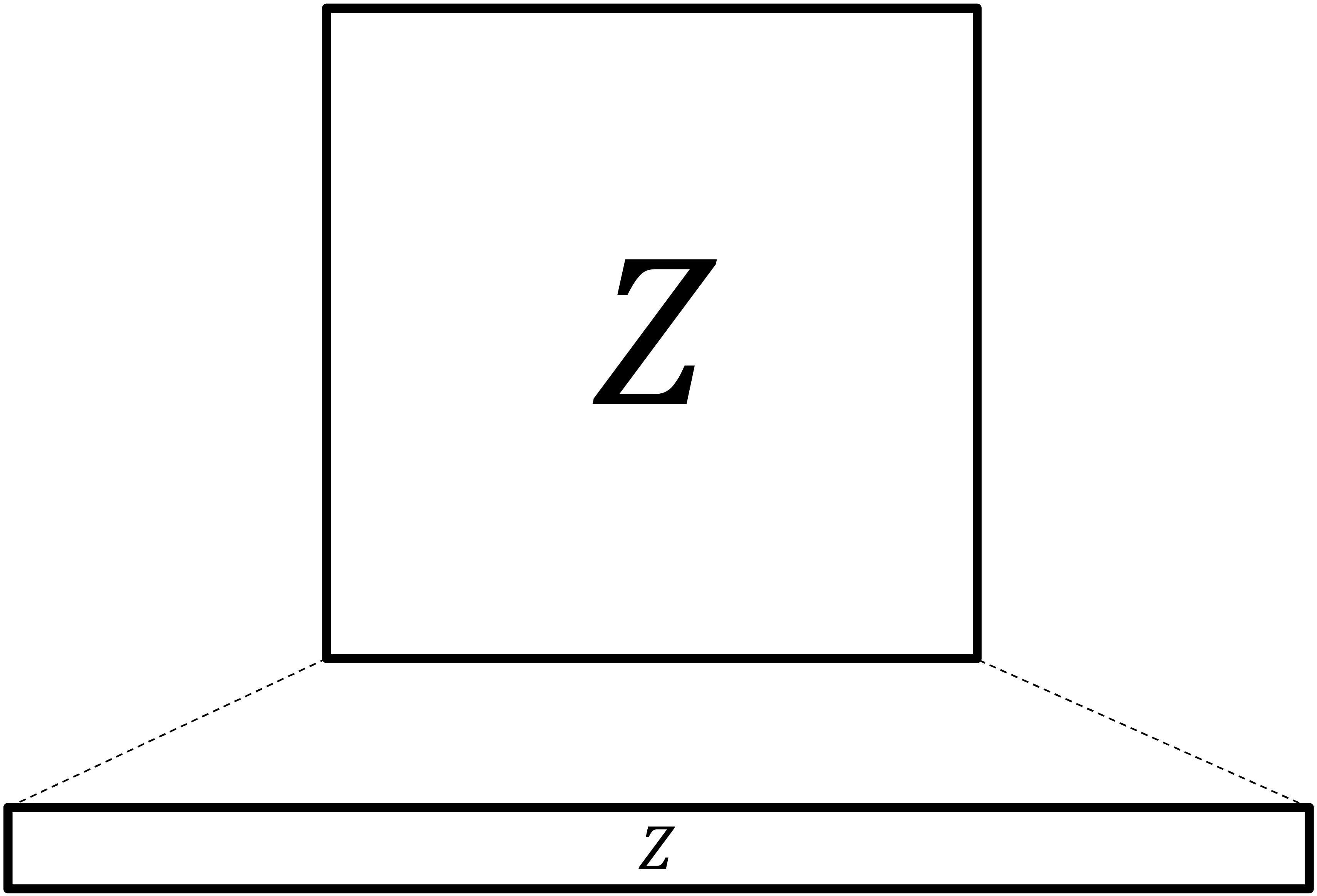
1. *if* $Z \equiv 1 \times 1$ matrix *then* $Z \leftarrow Z + X \cdot Y$
2. *else*
3. *Rec-MM* (Z_{11}, X_{11}, Y_{11}), *Rec-MM* (Z_{11}, X_{12}, Y_{21})
4. *Rec-MM* (Z_{12}, X_{12}, Y_{12}), *Rec-MM* (Z_{12}, X_{12}, Y_{22})
5. *Rec-MM* (Z_{21}, X_{21}, Y_{11}), *Rec-MM* (Z_{21}, X_{22}, Y_{21})
6. *Rec-MM* (Z_{22}, X_{21}, Y_{12}), *Rec-MM* (Z_{22}, X_{22}, Y_{22})

I/O-complexity (for $n > M$), $Q(n) = \begin{cases} O\left(n + \frac{n^2}{B}\right), & if\ n^2 \leq \alpha M \\ 8Q\left(\frac{n}{2}\right) + O(1), & otherwise \end{cases}$

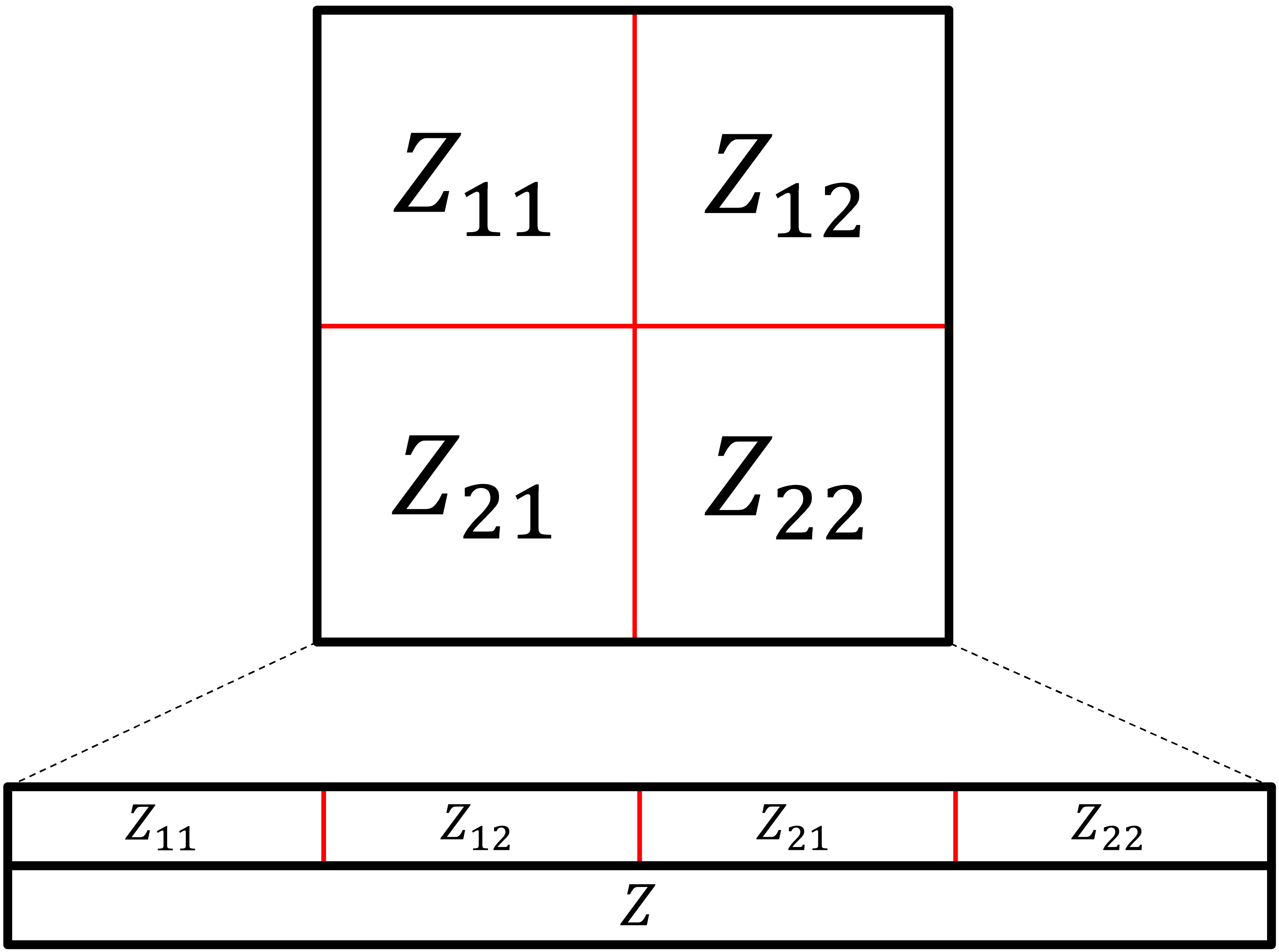
$$= O\left(\frac{n^3}{M} + \frac{n^3}{B\sqrt{M}}\right) = O\left(\frac{n^3}{B\sqrt{M}}\right), when\ M = \Omega(B^2)$$

I/O-complexity (for all n) $= O\left(\frac{n^3}{B\sqrt{M}} + \frac{n^2}{B} + 1\right)$ (why?)

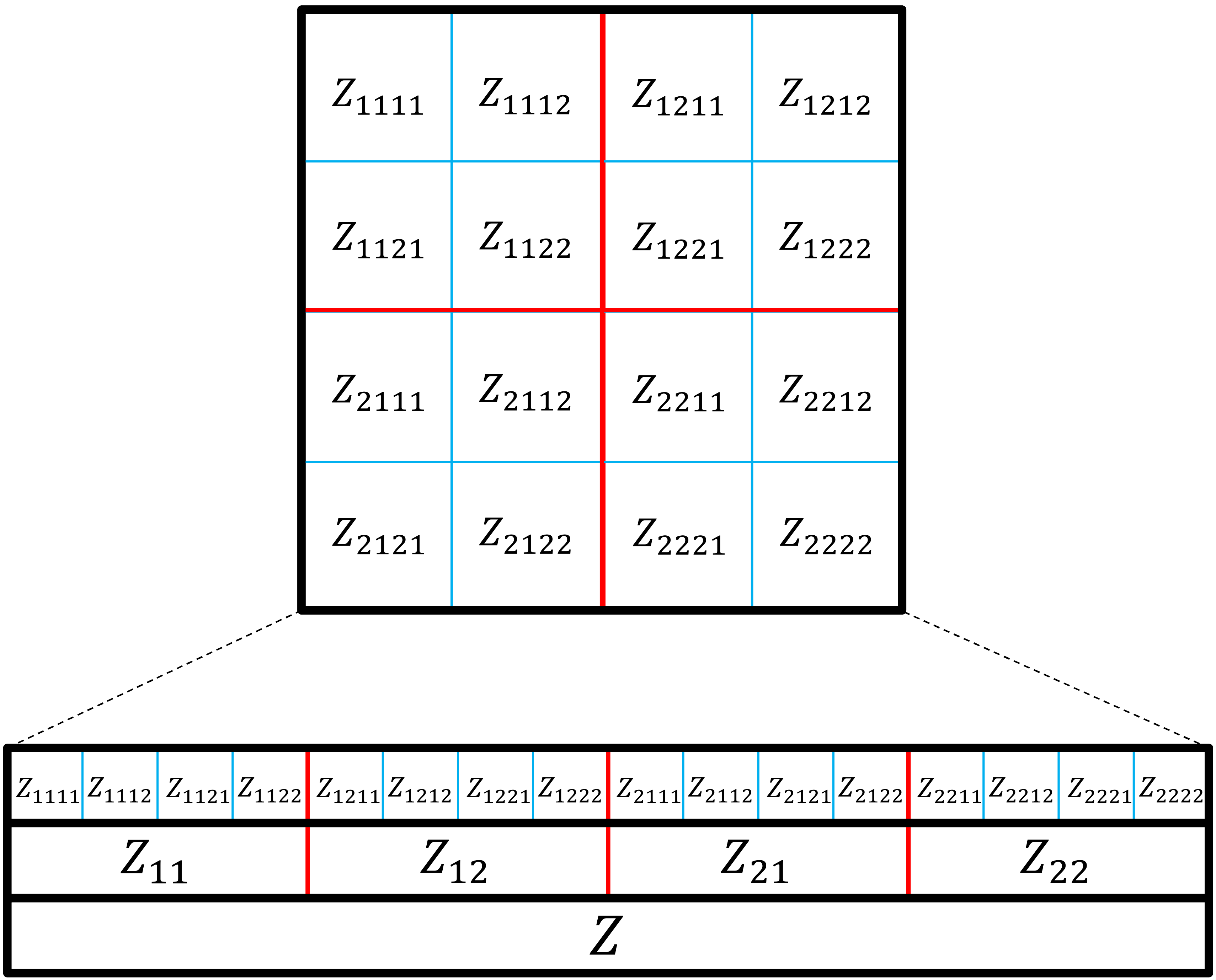
Recursive Matrix Multiplication with Z-Morton Layout



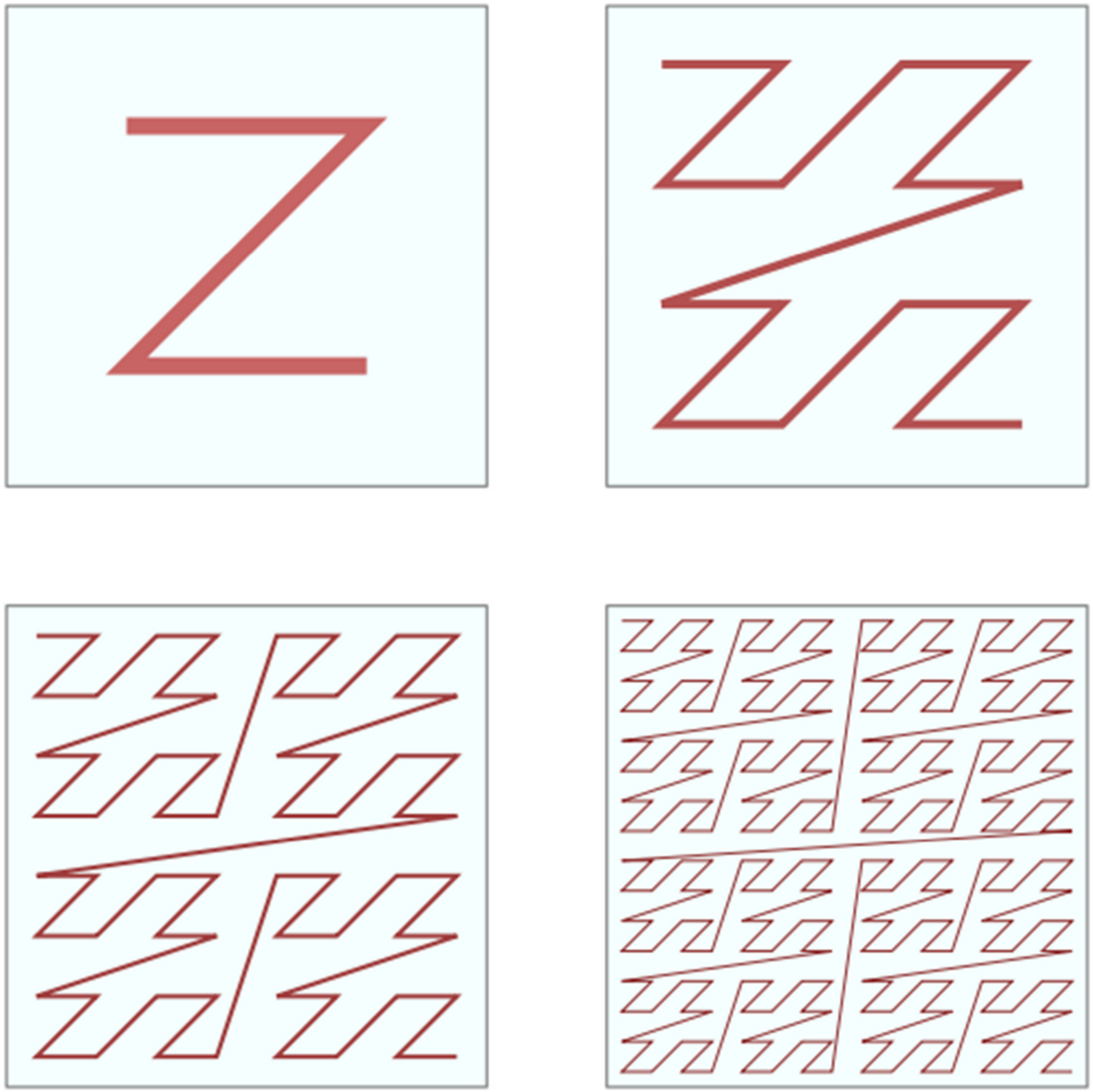
Recursive Matrix Multiplication with Z-Morton Layout



Recursive Matrix Multiplication with Z-Morton Layout



Recursive Matrix Multiplication with Z-Morton Layout



Source: wikipedia

Recursive Matrix Multiplication with Z-Morton Layout

Rec-MM (Z, X, Y)

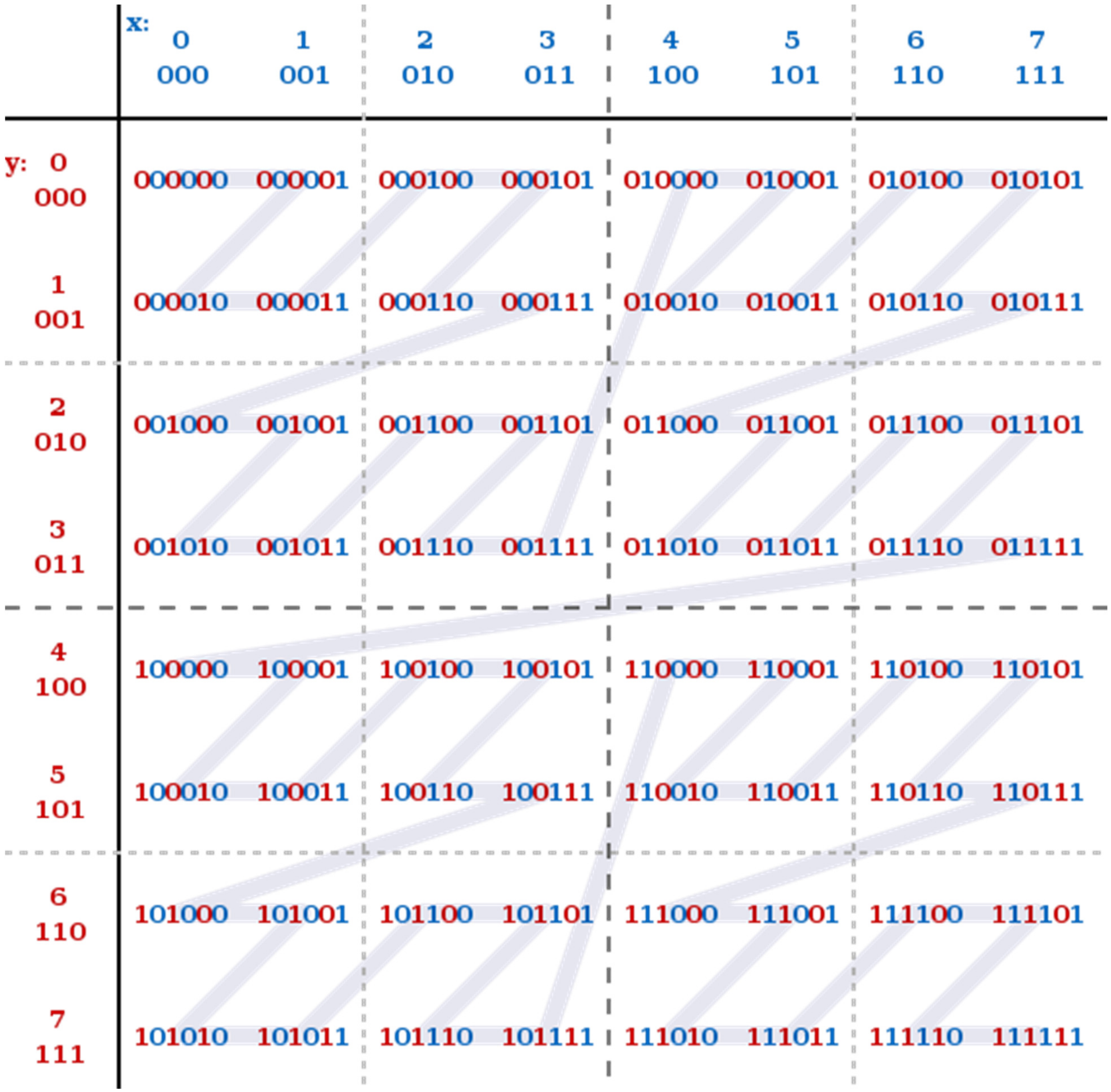
1. *if* $Z \equiv 1 \times 1$ matrix *then* $Z \leftarrow Z + X \cdot Y$
2. *else*
3. *Rec-MM* (Z_{11}, X_{11}, Y_{11}), *Rec-MM* (Z_{11}, X_{12}, Y_{21})
4. *Rec-MM* (Z_{12}, X_{12}, Y_{12}), *Rec-MM* (Z_{12}, X_{12}, Y_{22})
5. *Rec-MM* (Z_{21}, X_{21}, Y_{11}), *Rec-MM* (Z_{21}, X_{22}, Y_{21})
6. *Rec-MM* (Z_{22}, X_{21}, Y_{12}), *Rec-MM* (Z_{22}, X_{22}, Y_{22})

I/O-complexity (for $n > M$), $Q(n) = \begin{cases} O\left(1 + \frac{n^2}{B}\right), & if\ n^2 \le \alpha M \\ 8Q\left(\frac{n}{2}\right) + O(1), & otherwise \end{cases}$

$$= O\left(\frac{n^3}{M\sqrt{M}} + \frac{n^3}{B\sqrt{M}}\right) = O\left(\frac{n^3}{B\sqrt{M}}\right), when\ M = \Omega(B)$$

I/O-complexity (for all n) $= O\left(\frac{n^3}{B\sqrt{M}} + \frac{n^2}{B} + 1\right)$

Recursive Matrix Multiplication with Z-Morton Layout



Source: wikipedia