

CS 267
More on
Communication-optimal Matmul
(and beyond)

James Demmel
www.cs.berkeley.edu/~demmel

Outline

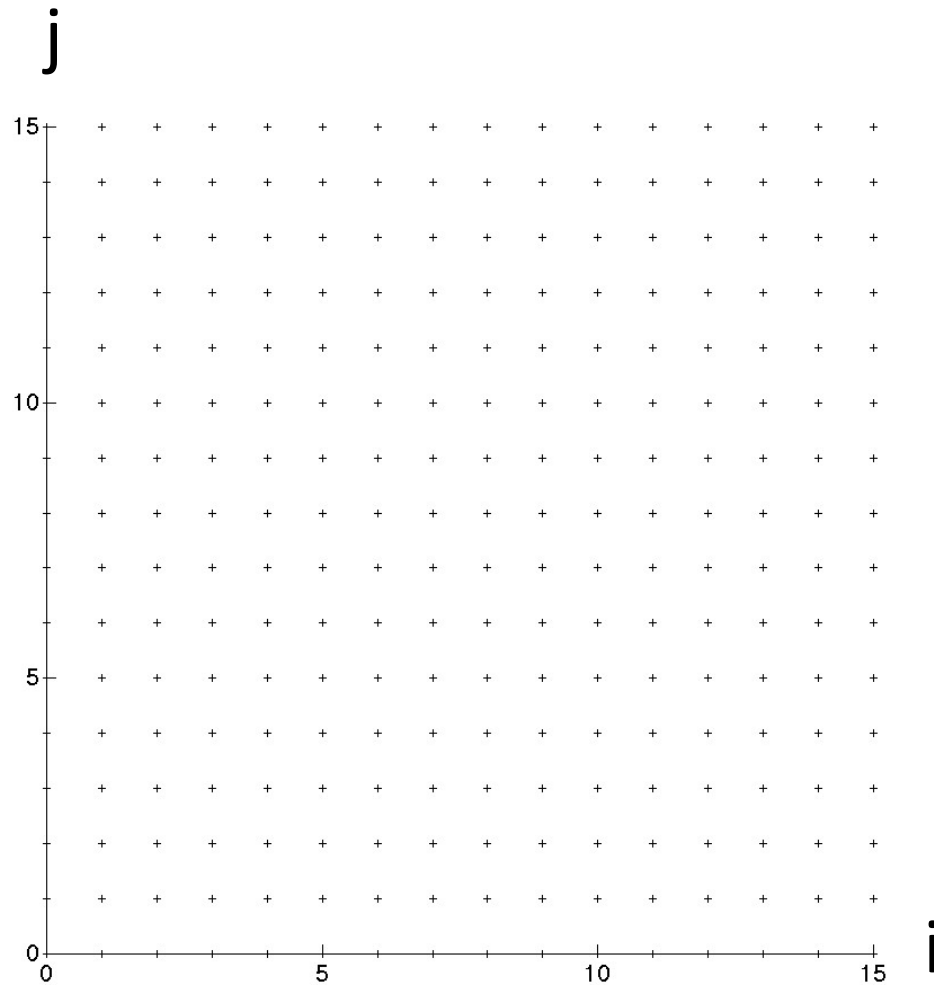
- Communication = moving data
 - Between main memory and cache
 - Between processors over a network
 - Most expensive operation (in time or energy)
- Goal: Provably minimize communication for algorithms that look like nested loops accessing arrays
 - Includes matmul, linear algebra (dense and sparse), n-body, convolutional neural nets (CNNs), ...
- Simple case: n-body (sequential, with main memory and cache)
 - Communication lower bound and optimal algorithm
- Extension to Matmul
- Extension to algorithms that look like nested loops accessing arrays, like CNNs (and open questions)

Data access for n-body

- $A()$ = array of structures
 - $A(i)$ contains position, charge on particle i
- Usual n-body
 - for $i = 1:n$, for $j = 1:n$ except i , $F(i) = F(i) + \text{force}(A(i), A(j))$
- Simplify to make counting easier
 - Let $B()$ = array of disjoint set of particles
 - for $i = 1:n$, for $j = 1:n$, $e = e + \text{potential}(A(i), B(j))$
- Simplify more
 - for $i = 1:n$, for $j = 1:n$, access $A(i)$ and $B(j)$

Data access for n-body

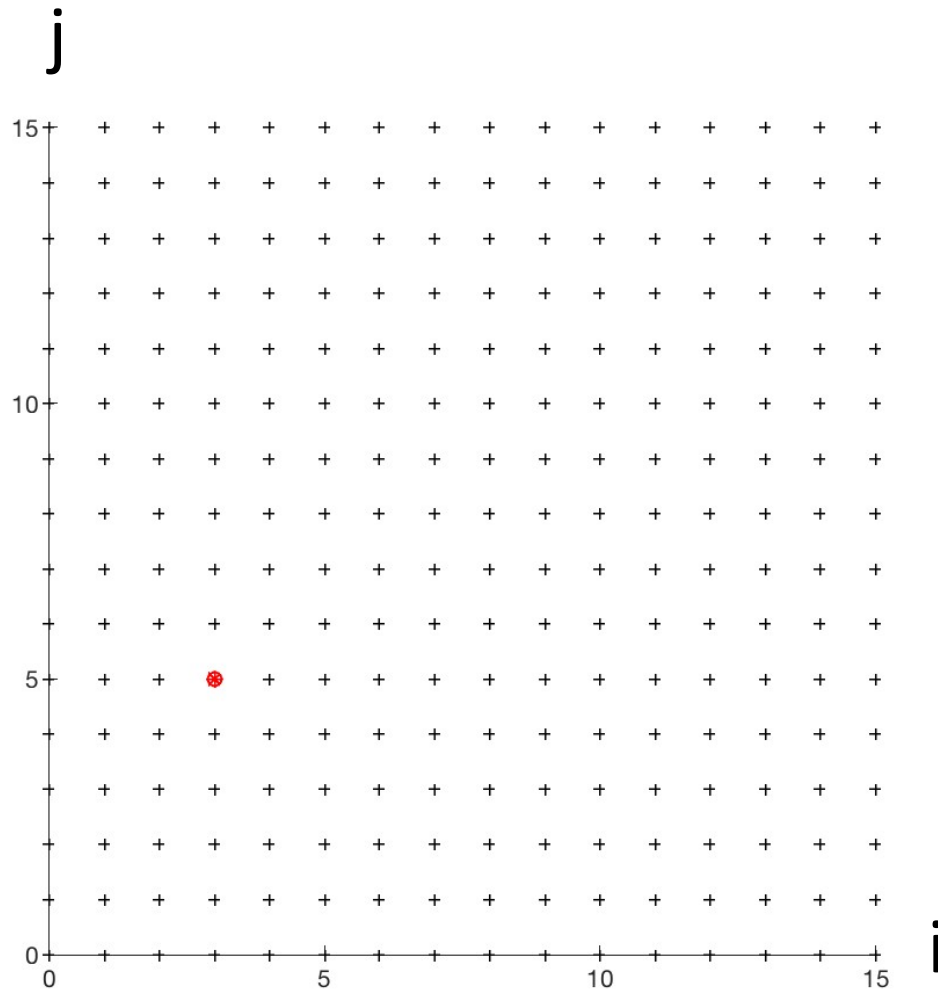
```
for i = 0:n  
  for j = 0:n  
    access A(i), B(j)
```



Data access for n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(j)
```

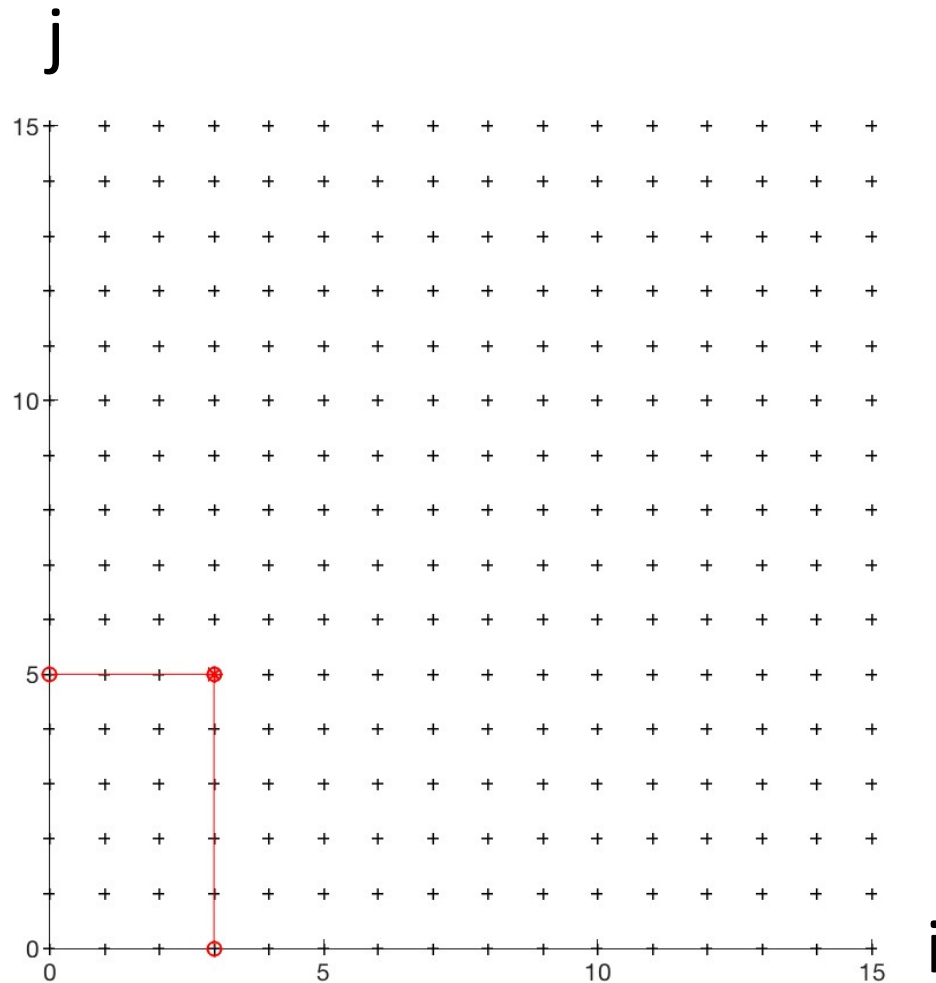
Ex: execute loop for
 $i = 3, j = 5$



Data access for n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(j)
```

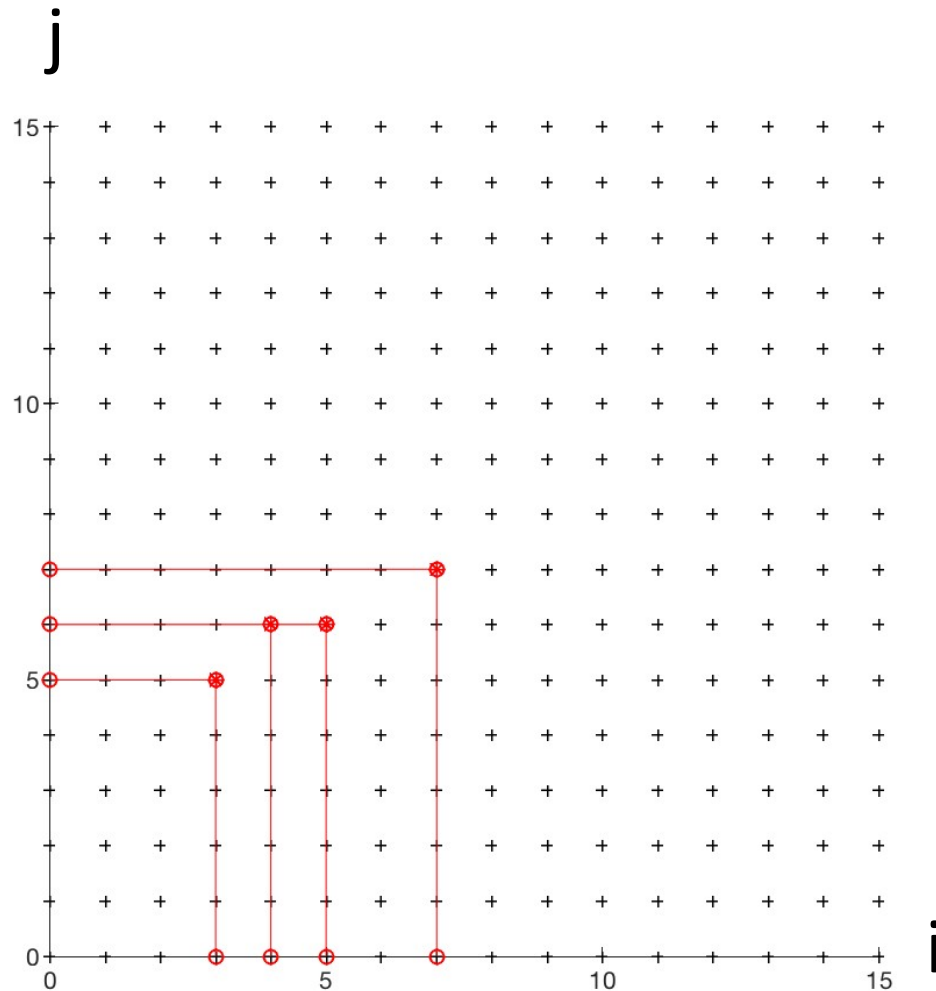
```
Ex: execute loop for  
    i = 3, j = 5  
access A(3), B(5)
```



Data access for n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(j)
```

Ex: execute loop for
multiple pairs (i,j),
access multiple
A(i), B(j)

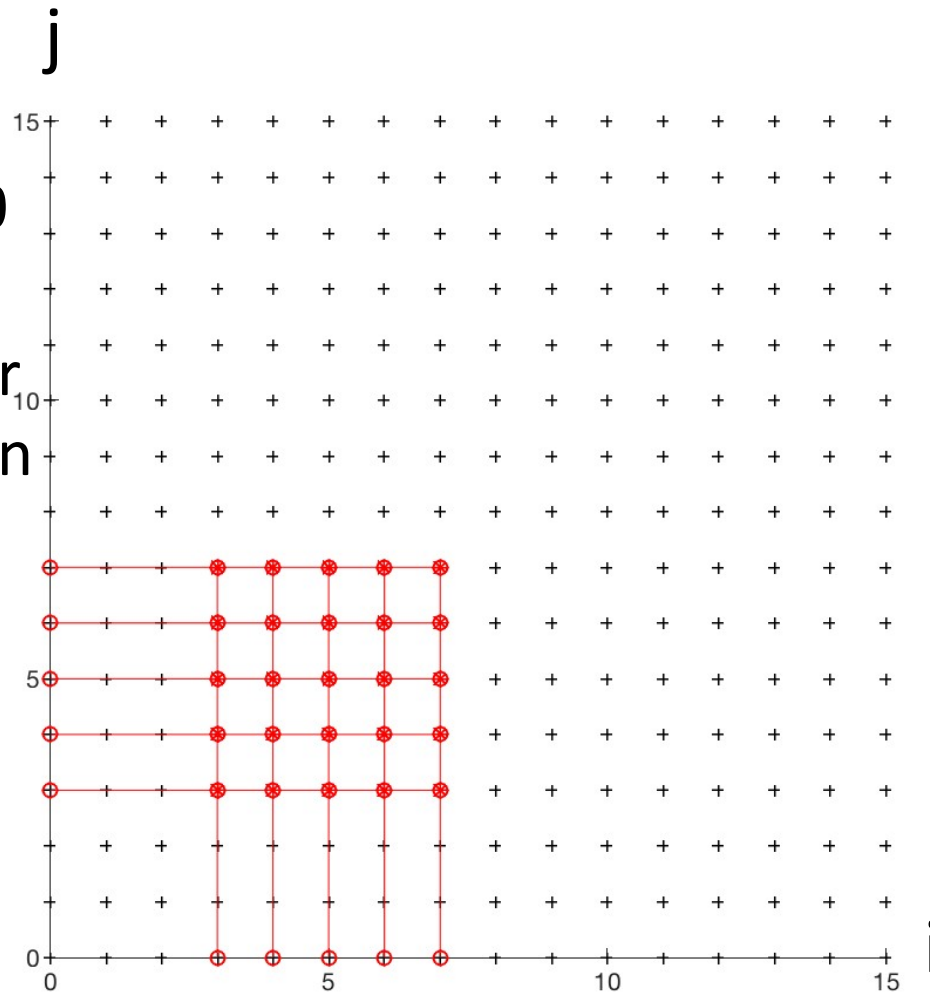


Data access for n-body

If we can only access 10 entries of $A(i)$ and $B(j)$, what is the max number of loop iterations we can do?

$$25 = 5 \times 5$$

If we can access M = cache size entries, then we can do $(M/2)^2 = M^2/4$ loop iterations



Communication lower bound for n-body (intuition)

- for $i=1:n$, for $j=1:n$, access $A(i)$, $B(j)$
- With a cache of size M full of data, can only perform $M^2/4$ loop iterations
- To perform all n^2 loop iterations, need to (re)fill cache $n^2/(M^2/4) = 4(n/M)^2$ times
- Filling cache costs M reads from slow memory
- Need to do at least $4(n/M)^2 * M = 4n^2 / M$ reads
 - Can improve constant slightly
 - Write as $\Omega(n^2/M) = \Omega(\text{\#loop iterations} / M)$

Optimal tiling for usual n-body

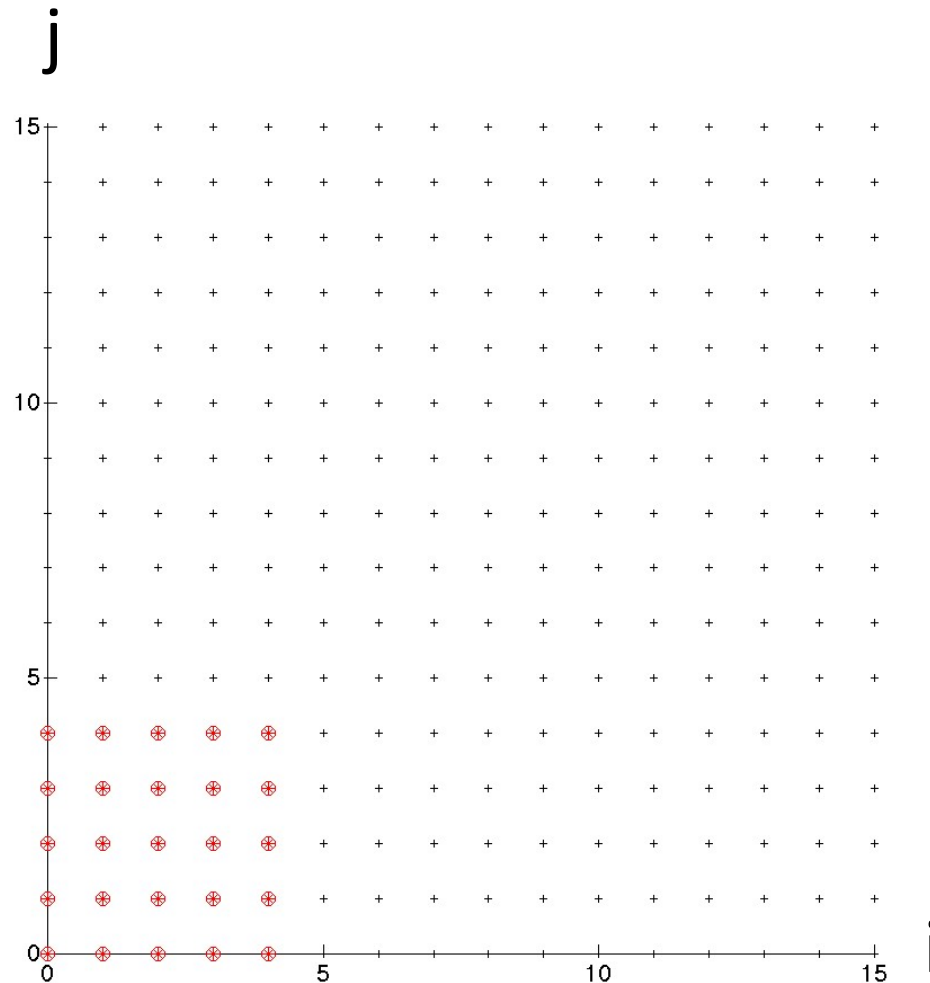
```
for i = 0:n
  for j = 0:n
    access A(i), B(j)
```

Tiling (M=10)

Read 5 entries of A:
A([0,1,2,3,4])

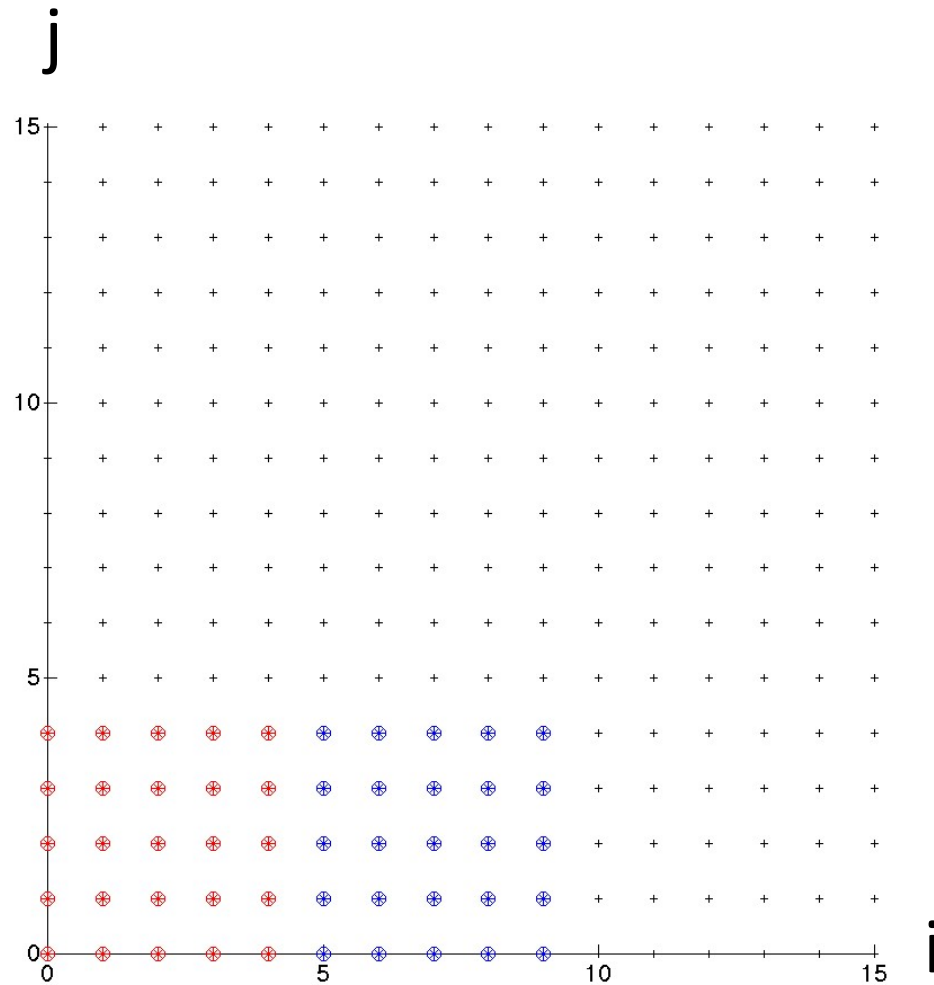
Read 5 entries of B:
B([0,1,2,3,4])

Perform $5^2 = 25$
loop iterations



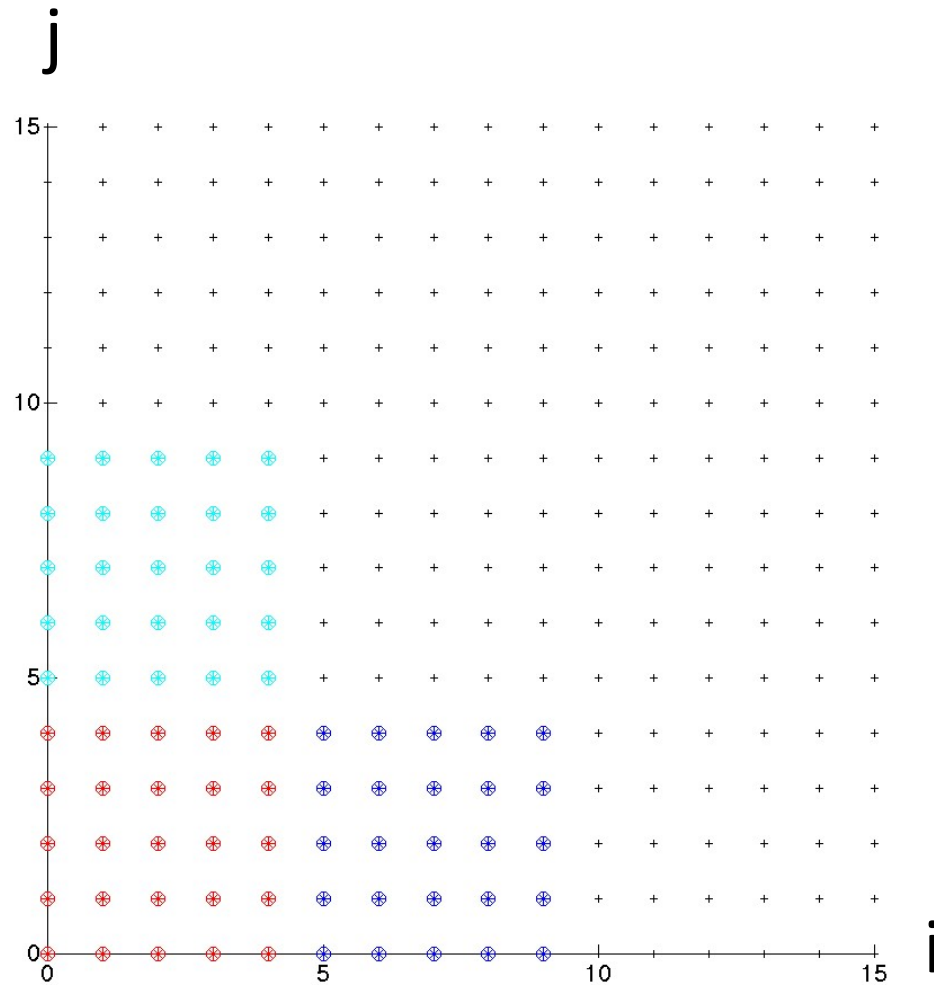
Optimal tiling for usual n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(j)
```



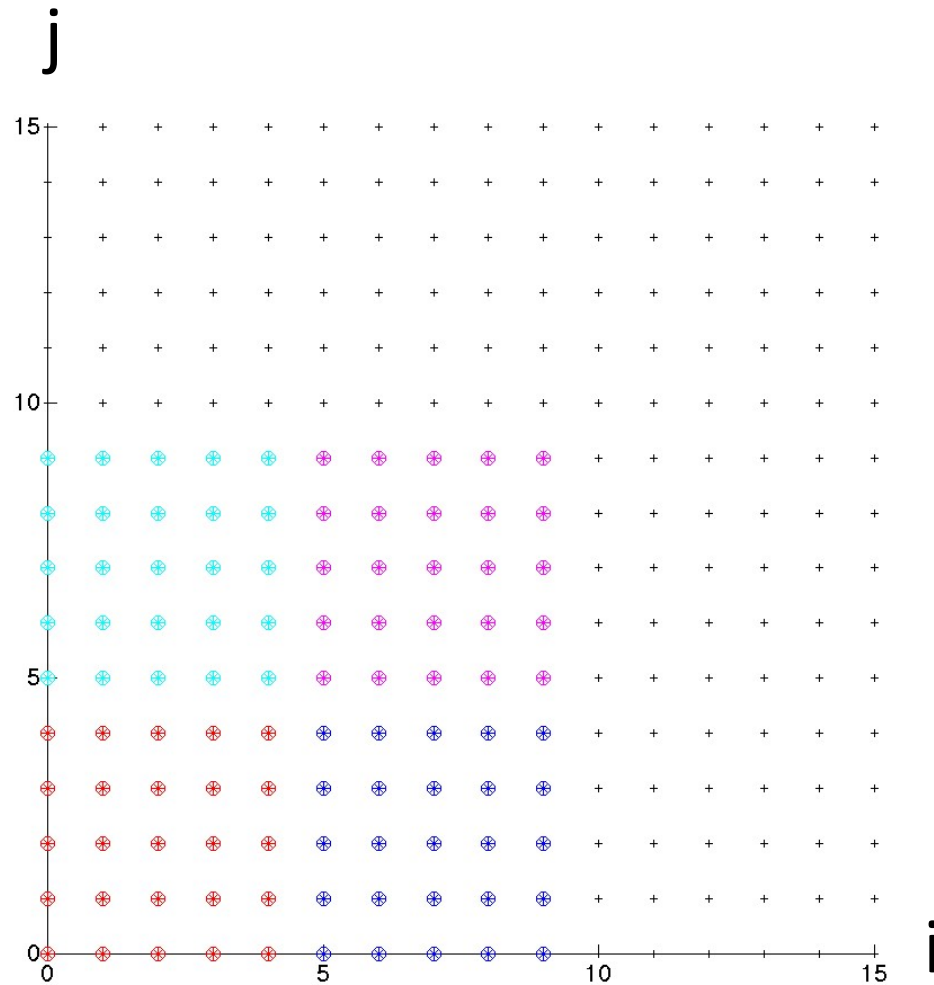
Optimal tiling for usual n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(j)
```



Optimal tiling for usual n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(j)
```



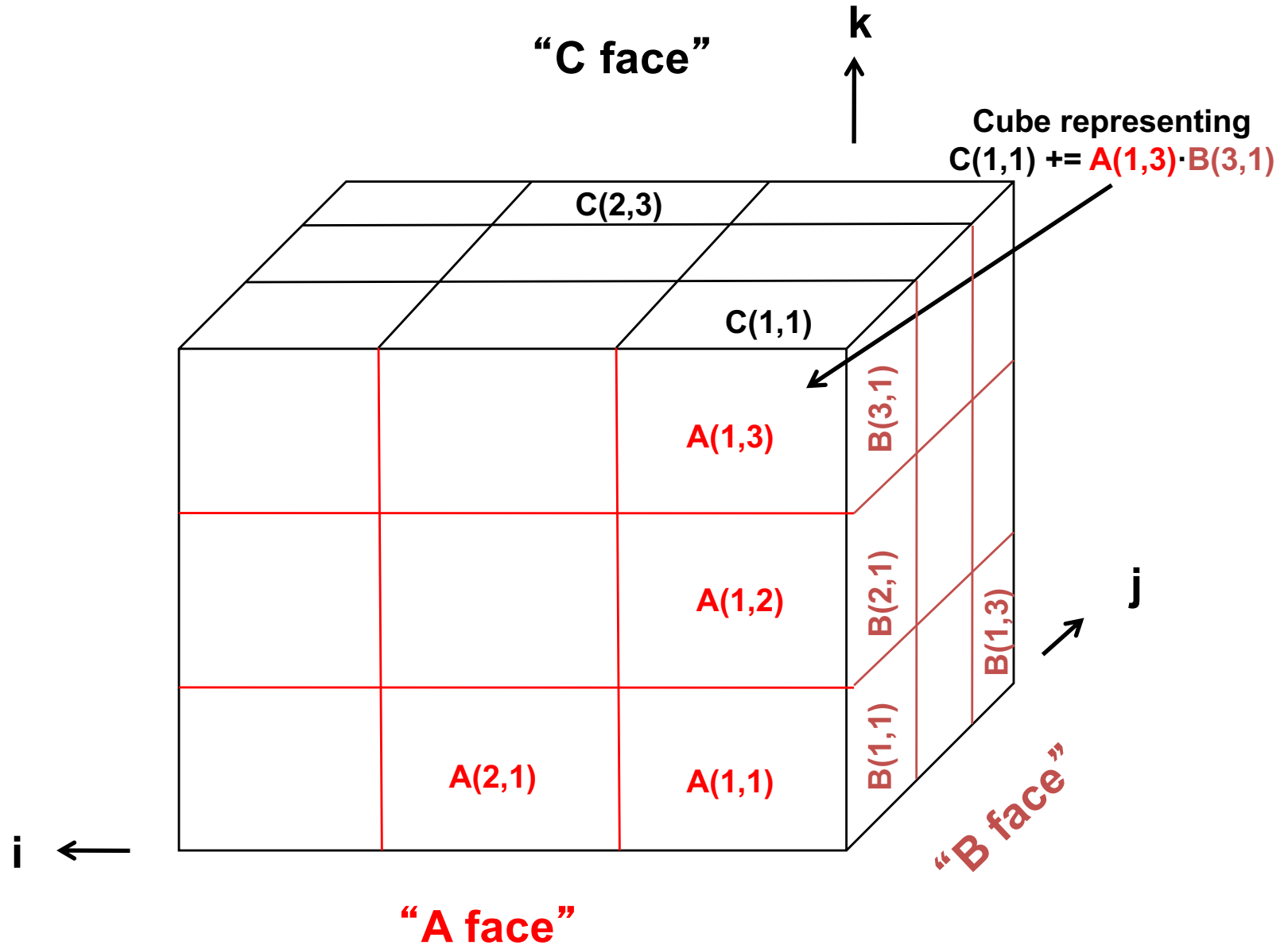
Generalizing to other algorithms

- Many algorithms look like nested loops accessing arrays
 - Linear Algebra (dense and sparse)
 - Grids (structured and unstructured)
 - Convolutional Neural Nets (CNNs) ...
- Matmul: $C = A * B$
 - for $i=1:n$, for $j=1:n$, for $k=1:n$
 $C(i,j) = C(i,j) + A(i,k) * B(k,j)$

Proof of Communication Lower Bound on $C = A \cdot B$ (1/4)

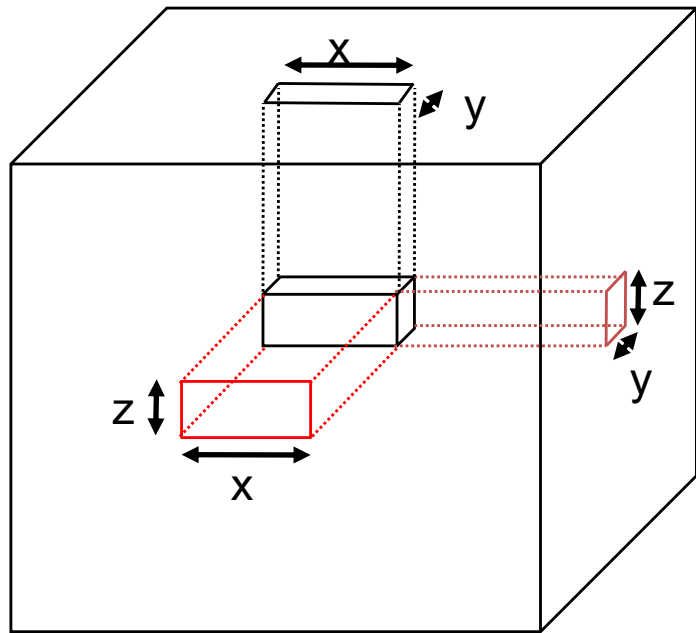
- Analogous to n-body:
 - Only M entries of A , B and C are available in cache
 - Find an upper bound F on the number of different iterations $C(i,j) = C(i,j) + A(i,k) \cdot B(k,j)$ we can perform
 - Need to refill cache n^3/F times to complete algorithm
 - Need to read/write at least $M n^3 / F$ words to/from cache
- Like n-body, represent iterations and data geometrically

Proof of Communication Lower Bound on $C = A \cdot B$ (2/4)

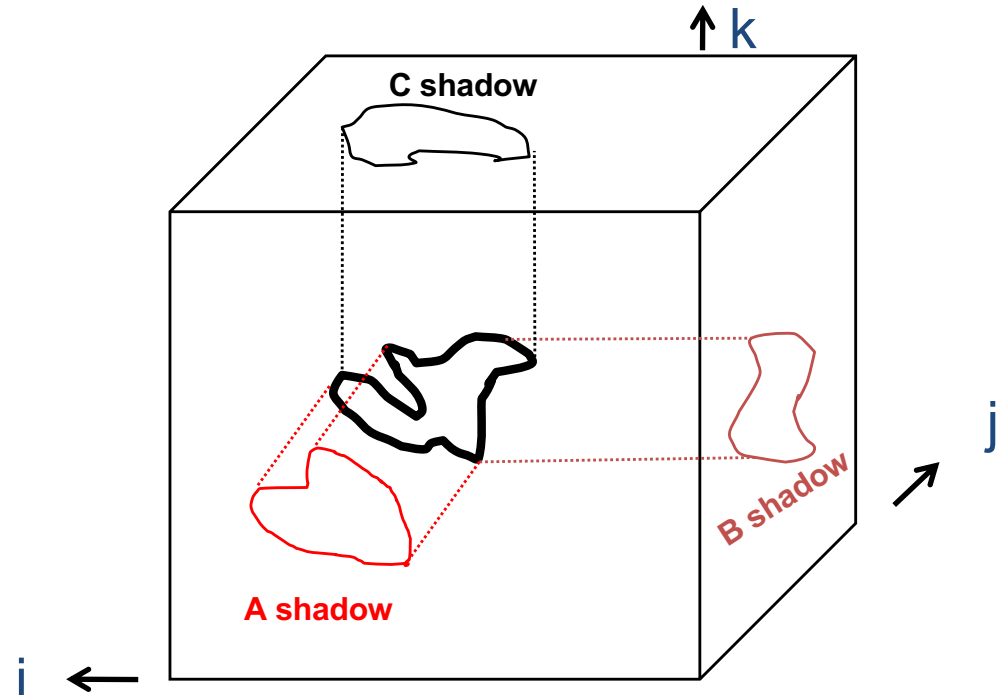


- If we have at most M "A squares", "B squares", and "C squares" on faces, how many cubes can we have?

Proof of Communication Lower Bound on $C = A \cdot B$ (3/4)



cubes in black box with
 side lengths x , y and z
 = Volume of black box
 = $x \cdot y \cdot z$
 = $(xz \cdot zy \cdot yx)^{1/2}$
 = $(\#A \square s \cdot \#B \square s \cdot \#C \square s)^{1/2}$



(i,k) is in **A shadow** if (i,j,k) in 3D set
 (j,k) is in **B shadow** if (i,j,k) in 3D set
 (i,j) is in **C shadow** if (i,j,k) in 3D set

Thm (Loomis & Whitney, 1949)

cubes in 3D set = Volume of 3D set
 $\leq (\text{area}(\text{A shadow}) \cdot \text{area}(\text{B shadow}) \cdot \text{area}(\text{C shadow}))^{1/2}$

Proof of Communication Lower Bound on $C = A \cdot B$ (4/4)

- # loop iterations doable with M words of data = #cubes
 $\leq (\text{area}(A \text{ shadow}) \cdot \text{area}(B \text{ shadow}) \cdot \text{area}(C \text{ shadow}))^{1/2}$
 $\leq (M \cdot M \cdot M)^{1/2} = M^{3/2} = F$
- Need to read/write at least $M n^3 / F = \Omega(n^3 / M^{1/2}) = \Omega(\text{\#loop iterations} / M^{1/2})$ words to/from cache

Recall optimal Matmul Algorithm

- Analogous to n-body:
 - What is the largest set of $C(i,j) += A(i,k) * B(k,j)$ we can perform given M entries $A(i,k)$, $B(k,j)$, $C(i,j)$?
 - What is the largest set of (i,j,k) we can have, given a bound M on the number of (i,k) , (k,j) , (i,j) ?
 - What is the shape of the largest 3D volume we can have, given a bound M on the area of its shadows in 3 directions?
 - Answer: A cube, with edge length $O(M^{1/2})$, volume $O(M^{3/2})$
 - Optimal "blocked" Algorithm: 6 nested loops, 3 innermost loops do $b \times b$ matmul with $b = O(M^{1/2})$

Proof of Communication Lower Bound on $C = A \cdot B$ (4/4)

- # loop iterations doable with M words of data = #cubes
 $\leq (\text{area}(A \text{ shadow}) \cdot \text{area}(B \text{ shadow}) \cdot \text{area}(C \text{ shadow}))^{1/2}$
 $\leq (M \cdot M \cdot M)^{1/2} = M^{3/2} = F$
- Need to read/write at least $M n^3 / F = \Omega(n^3 / M^{1/2}) = \Omega(\text{\#loop iterations} / M^{1/2})$ words to/from cache
- Parallel Case: apply reasoning to one processor out of P
 - "Fast memory" = local processor, "Slow memory" = other procs
 - Goal: lower bound # "reads/writes" = # words moved between one processor and others
 - # loop iterations = n^3 / P (load balanced)
 - $M = 3n^2 / P$ (each processor gets equal fraction of data)
 - # "reads/writes" $\geq M \cdot (n^3 / P) / (M)^{3/2} = \Omega(n^2 / P^{1/2})$

Approach to generalizing lower bounds

- Matmul
 - for $i=1:n$, for $j=1:n$, for $k=1:n$,
 $C(i,j) += A(i,k) * B(k,j)$
 - \Rightarrow for (i,j,k) in $S = \text{subset of } Z^3$
Access locations indexed by (i,j) , (i,k) , (k,j)
- General case
 - for $i_1=1:n$, for $i_2 = i_1:m$, ... for $i_k = i_3:i_4$
 $C(i_1+2*i_3-i_7) = \text{func}(A(i_2+3*i_4, i_1, i_2, i_1+i_2, \dots), B(\text{pnt}(3*i_4)), \dots)$
 $D(\text{something else}) = \text{func}(\text{something else}), \dots$
 - \Rightarrow for (i_1, i_2, \dots, i_k) in $S = \text{subset of } Z^k$
Access locations indexed by “projections”, eg
 $\phi_C(i_1, i_2, \dots, i_k) = (i_1+2*i_3-i_7)$
 $\phi_A(i_1, i_2, \dots, i_k) = (i_2+3*i_4, i_1, i_2, i_1+i_2, \dots), \dots$
- Goal: Communication lower bounds, optimal algorithms for *any* program that looks like this

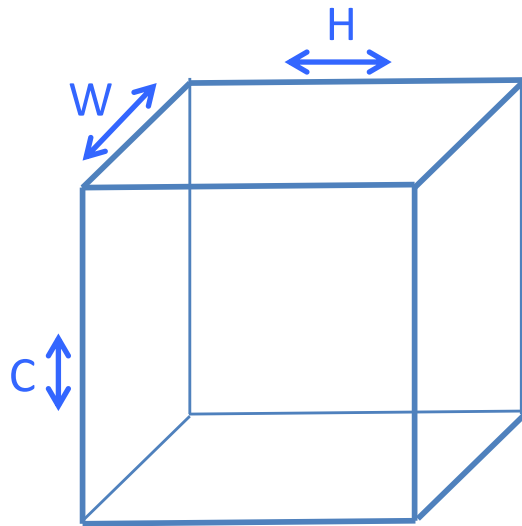
General Communication Lower Bound

- Thm: Given a program with array refs given by projections ϕ_j , then there is an $s_{\text{HBL}} \geq 1$ such that
$$\# \text{words_moved} = \Omega (\# \text{iterations} / M^{s_{\text{HBL}} - 1})$$
where s_{HBL} is the value of a linear program:
$$\begin{aligned} &\text{minimize } s_{\text{HBL}} = \sum_j e_j \text{ subject to} \\ &\text{rank}(H) \leq \sum_j e_j * \text{rank}(\phi_j(H)) \text{ for all subgroups } H < \mathbb{Z}^k \end{aligned}$$
- Proof depends on recent result in pure mathematics by Christ/Tao/Carbery/Bennett
 - Generalization of Hölder-Brascamp-Lieb (HBL) inequality to Abelian groups
 - HBL generalizes Cauchy-Schwartz, Loomis-Whitney, ...

Is this bound attainable?

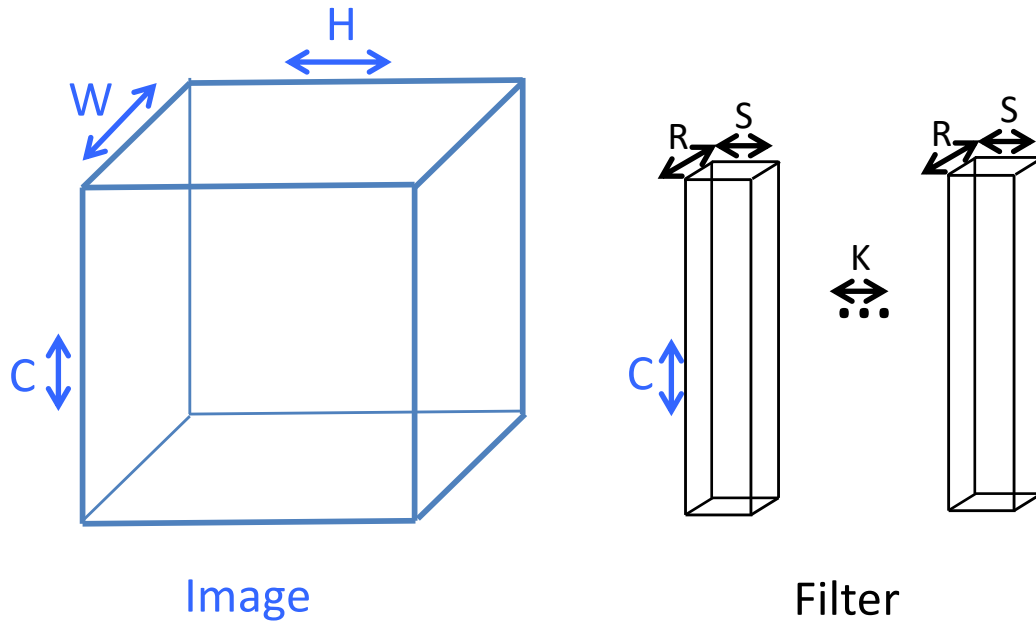
- Thm: We can always construct an optimal tiling, that attains the lower bound
- Assumptions/caveats/open questions
 - Attains lower bound $\Omega(\text{\#iterations}/M^{\text{sHBL}-1})$ in $O()$ sense
 - Depends on loop dependencies
 - Not all tilings may compute the right answer
 - Best case: no dependencies, or just reductions (like matmul)
 - Assumes loop bounds are large enough to fit tile
 - Ex: same lower bound for matmul applies to matrix-vector-multiply, but not attainable

What CNNs compute

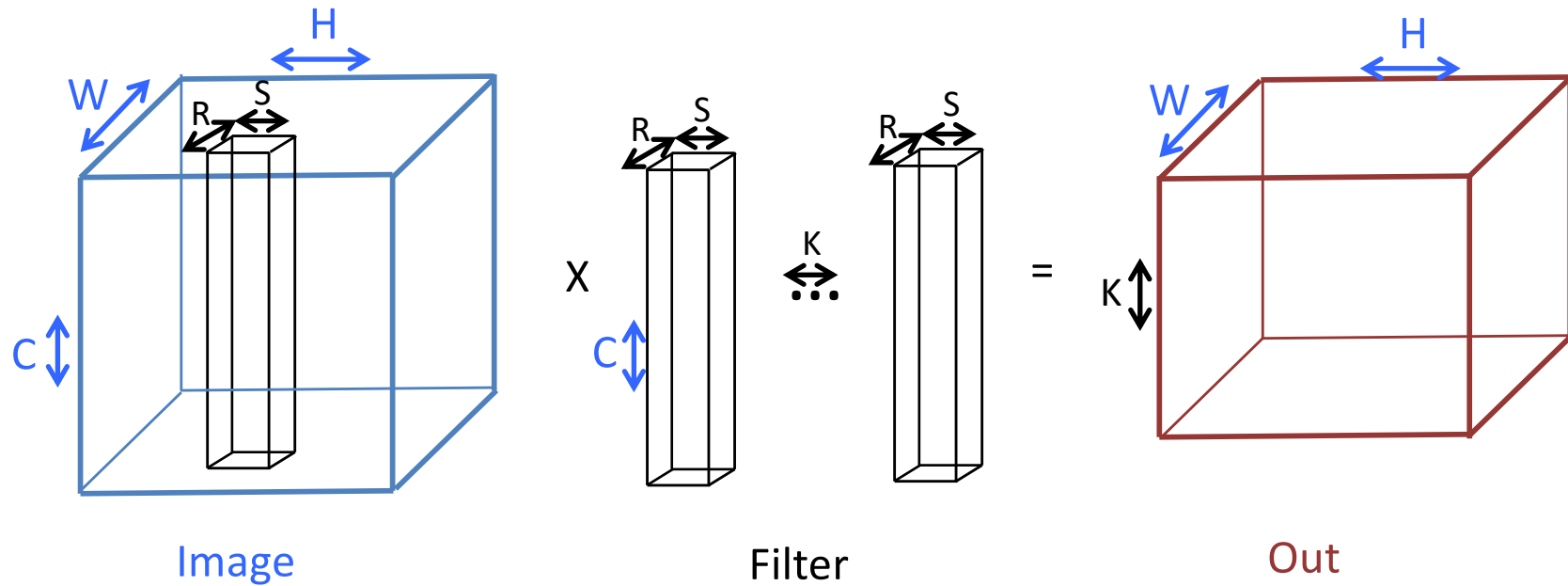


Image

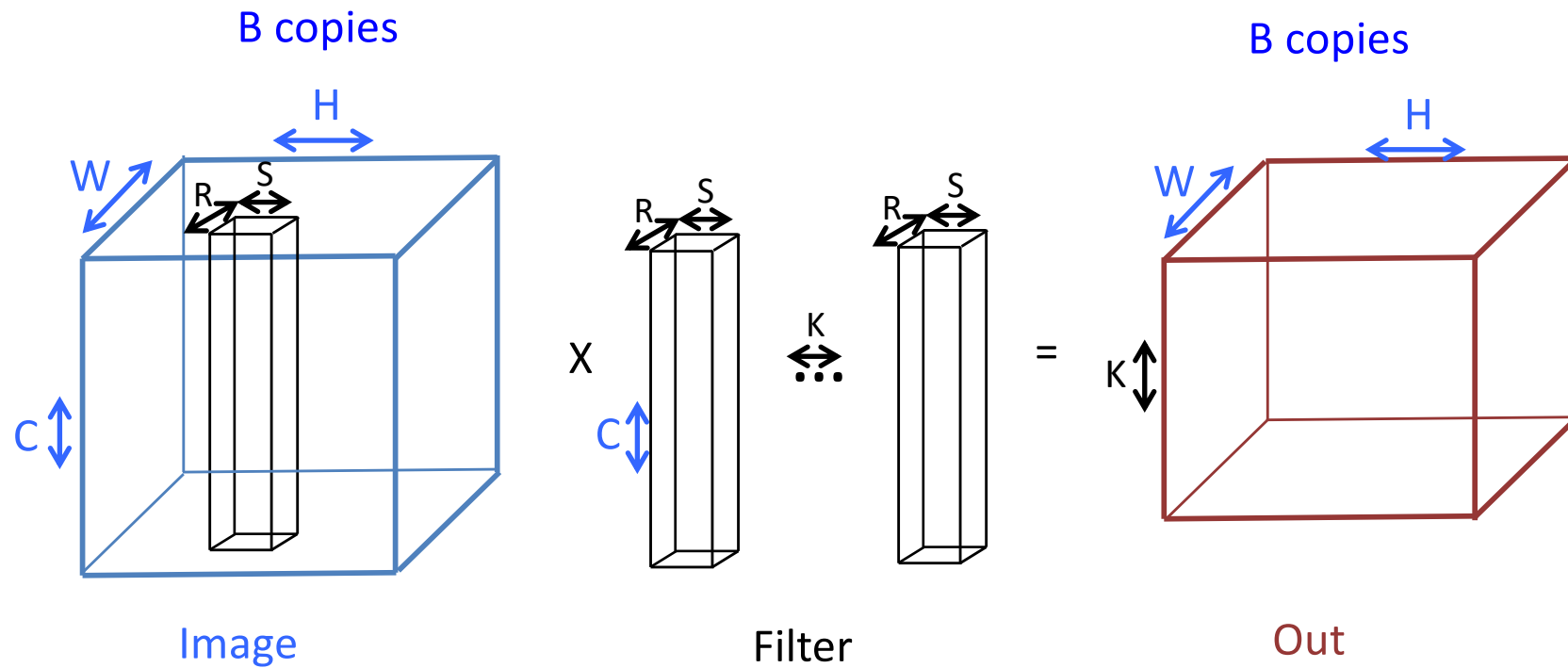
What CNNs compute



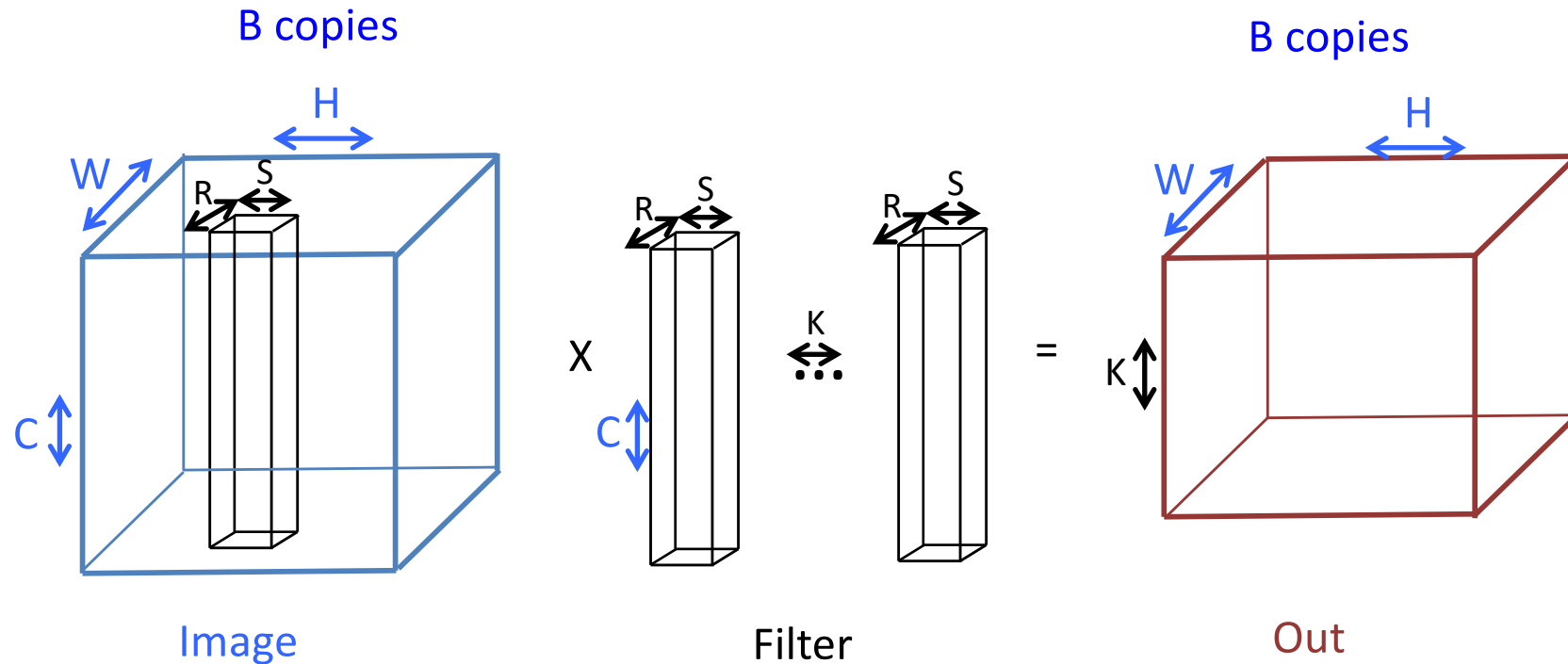
What CNNs compute



What CNNs compute



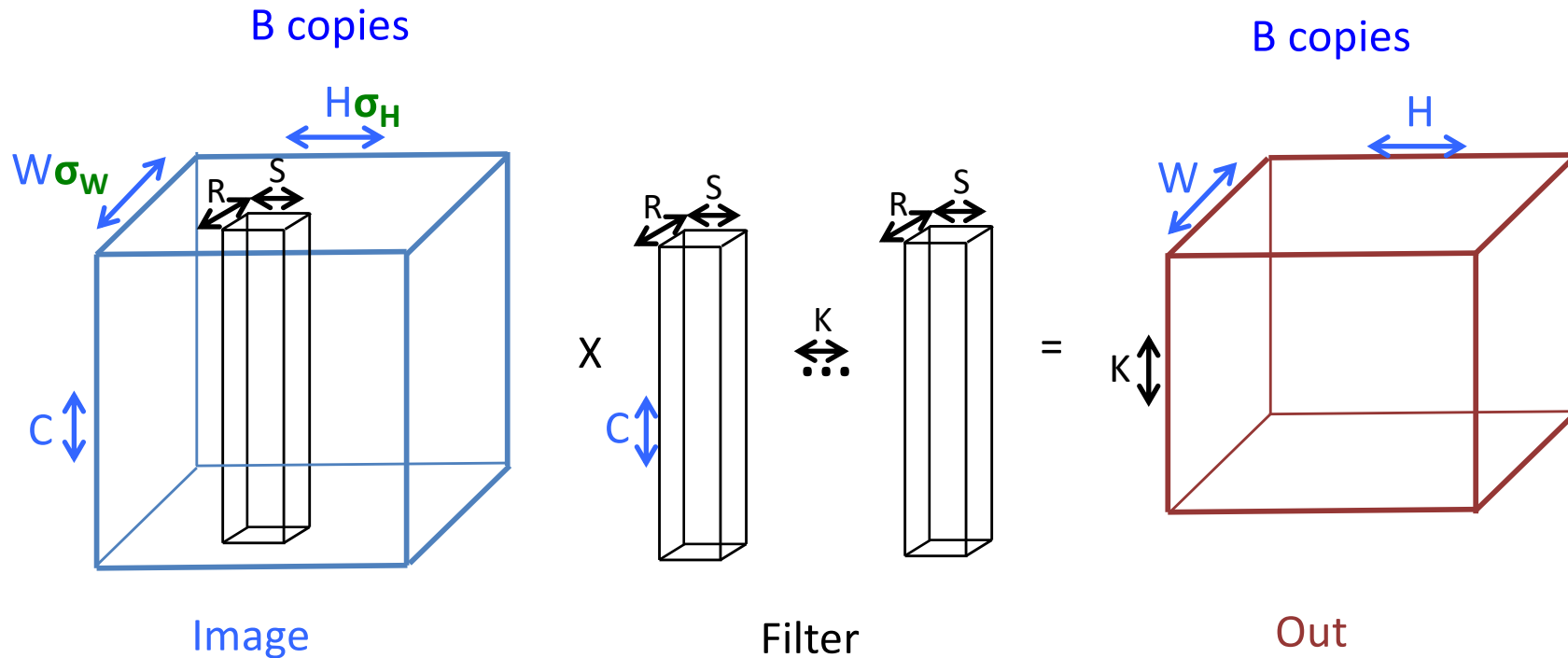
What CNNs compute



for $k=1:K$, for $h=1:H$, for $w=1:W$, for $r=1:R$,
for $s=1:S$, for $c=1:C$, for $b=1:B$

$\text{Out}(k, h, w, b) += \text{Image}(r+w, s+h, c, b) * \text{Filter}(k, r, s, c)$

What CNNs compute



for $k=1:K$, for $h=1:H$, for $w=1:W$, for $r=1:R$,
 for $s=1:S$, for $c=1:C$, for $b=1:B$

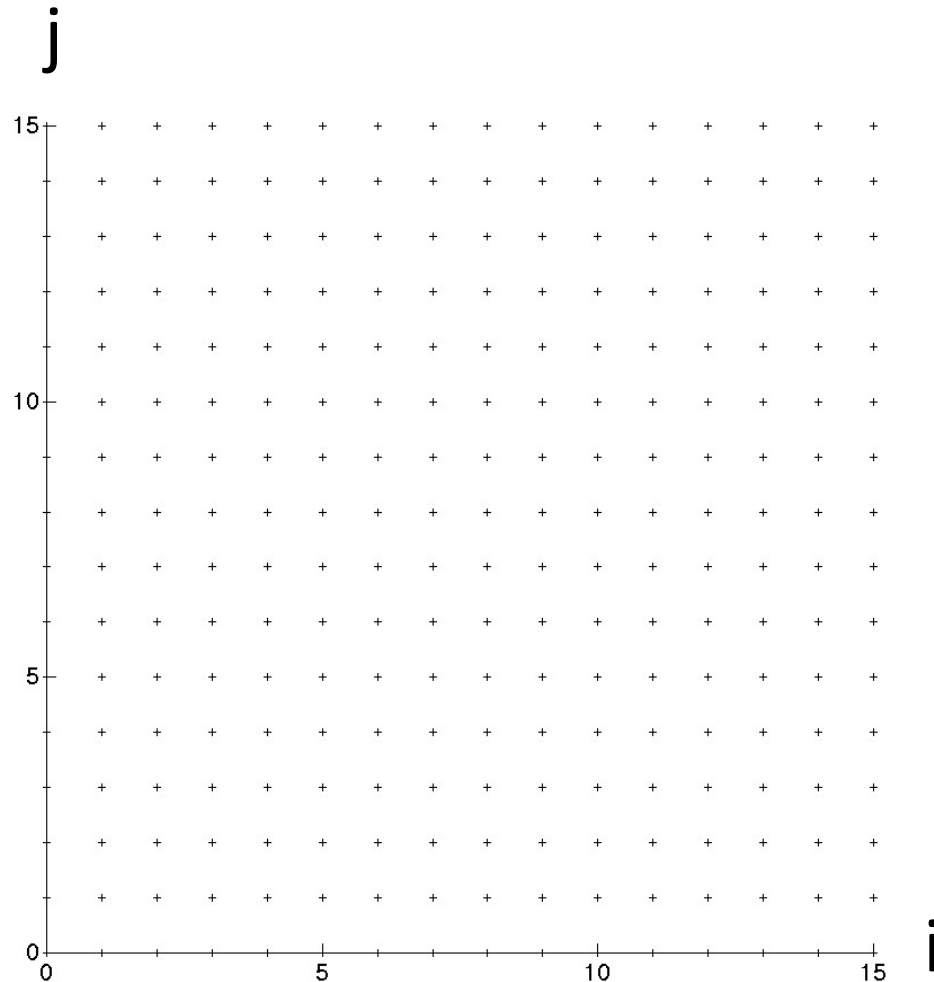
$$\text{Out}(k, h, w, b) += \text{Image}(r + \sigma_w w, s + \sigma_h h, c, b) * \text{Filter}(k, r, s, c)$$

Communication Lower Bound for CNNs

- Let $N = \text{\#iterations} = KHW RSCB$, $M = \text{cache size}$
- $\text{\#words moved} = \Omega(\max(\dots 5 \text{ terms}$
 - $BKHW, \dots \text{size of Out}$
 - $\sigma_H \sigma_W BCWH, \dots \text{size of Image}$
 - $CKRS, \dots \text{size of Filter}$
 - $N/M, \dots \text{lower bound from n-body}$
 - $N/(M^{1/2} (RS/(\sigma_H \sigma_W))^{1/2}) \dots \text{new lower bound}$
- New lower bound
 - Beats matmul by factor $(RS/(\sigma_H \sigma_W))^{1/2}$
 - Applies in common case when data does not fit in cache, but one $R \times S$ filter does
 - Tile needed to attain N/M too big to fit in loop bounds
- Attainable (many cases)

Optimal tiling for “slanted” n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(i+j)
```



Optimal tiling for “slanted” n-body

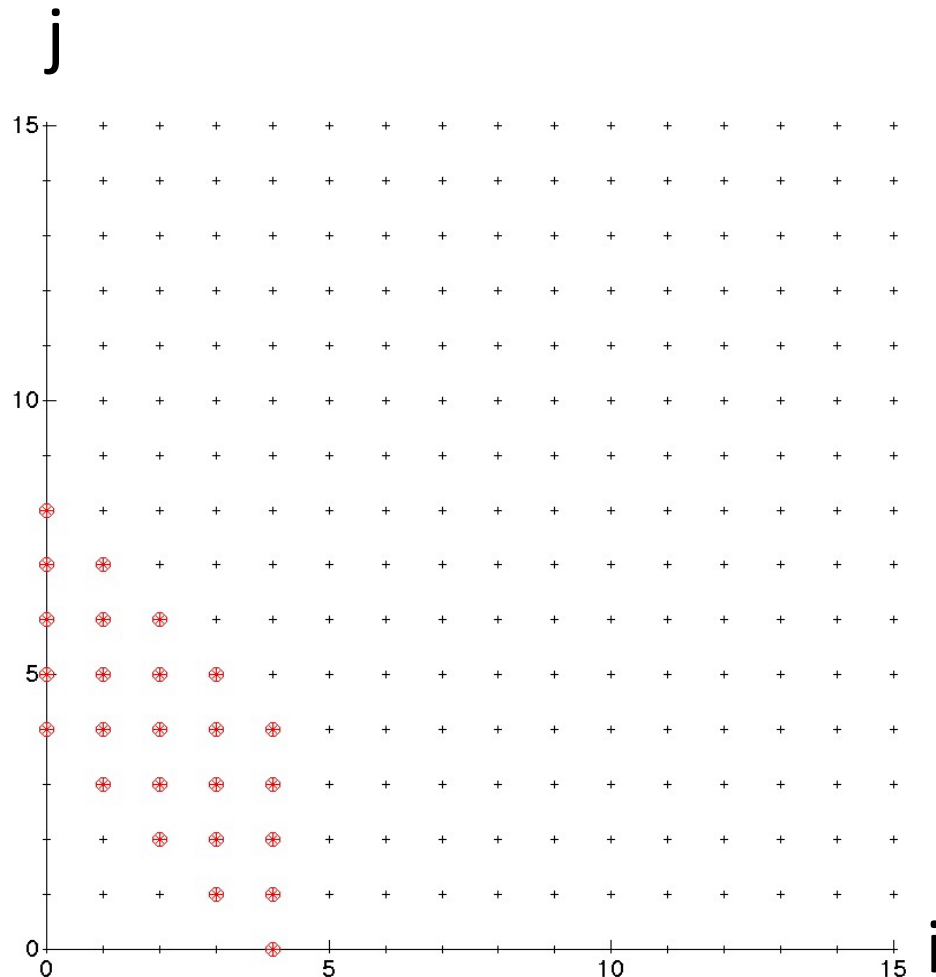
```
for i = 0:n
  for j = 0:n
    access A(i), B(i+j)
```

Tiling:

Read 5 entries of A:
A([0,1,2,3,4])

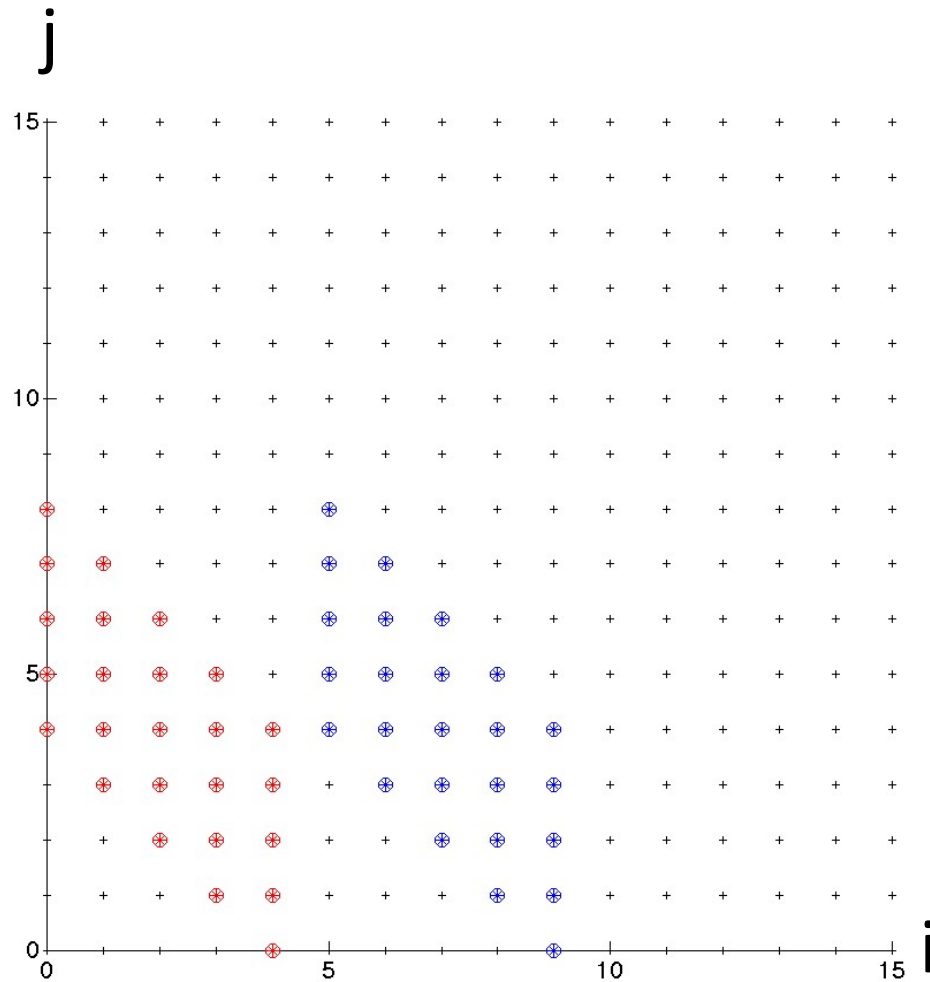
Read 5 entries of B:
B([4,5,6,7,8])

Perform $5^2 = 25$
loop iterations



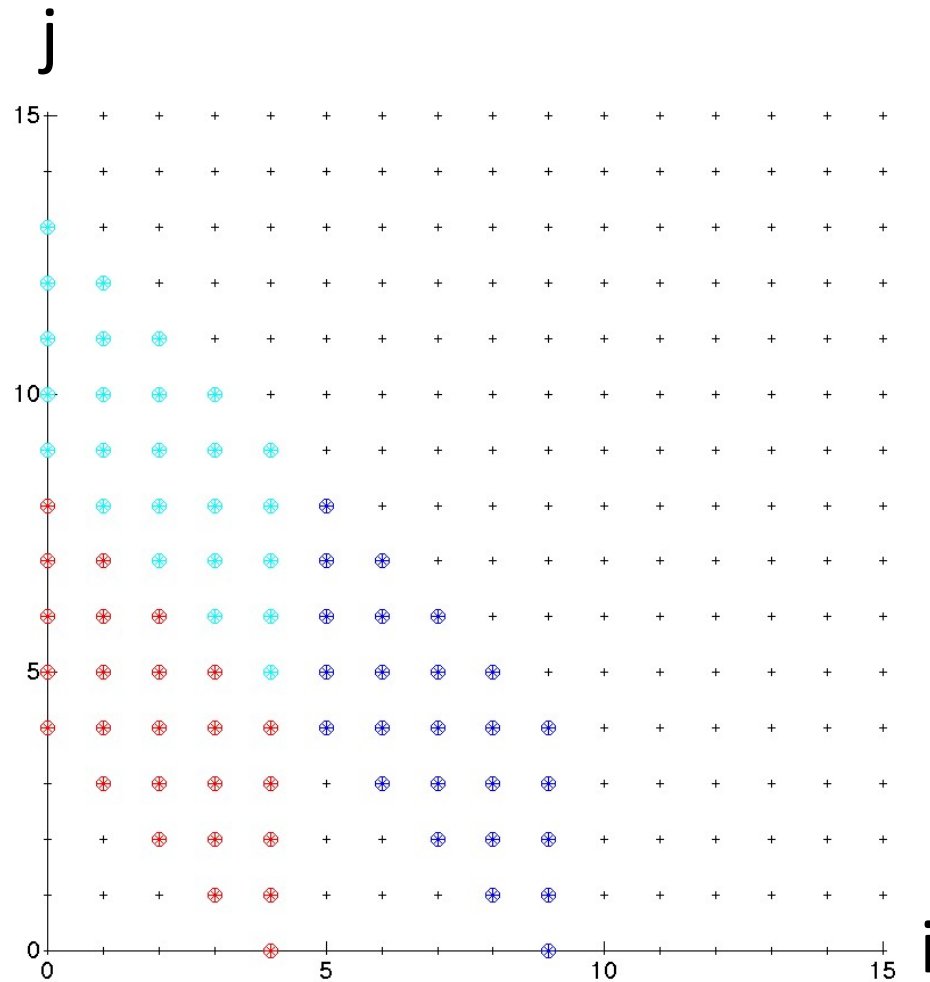
Optimal tiling for “slanted” n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(i+j)
```



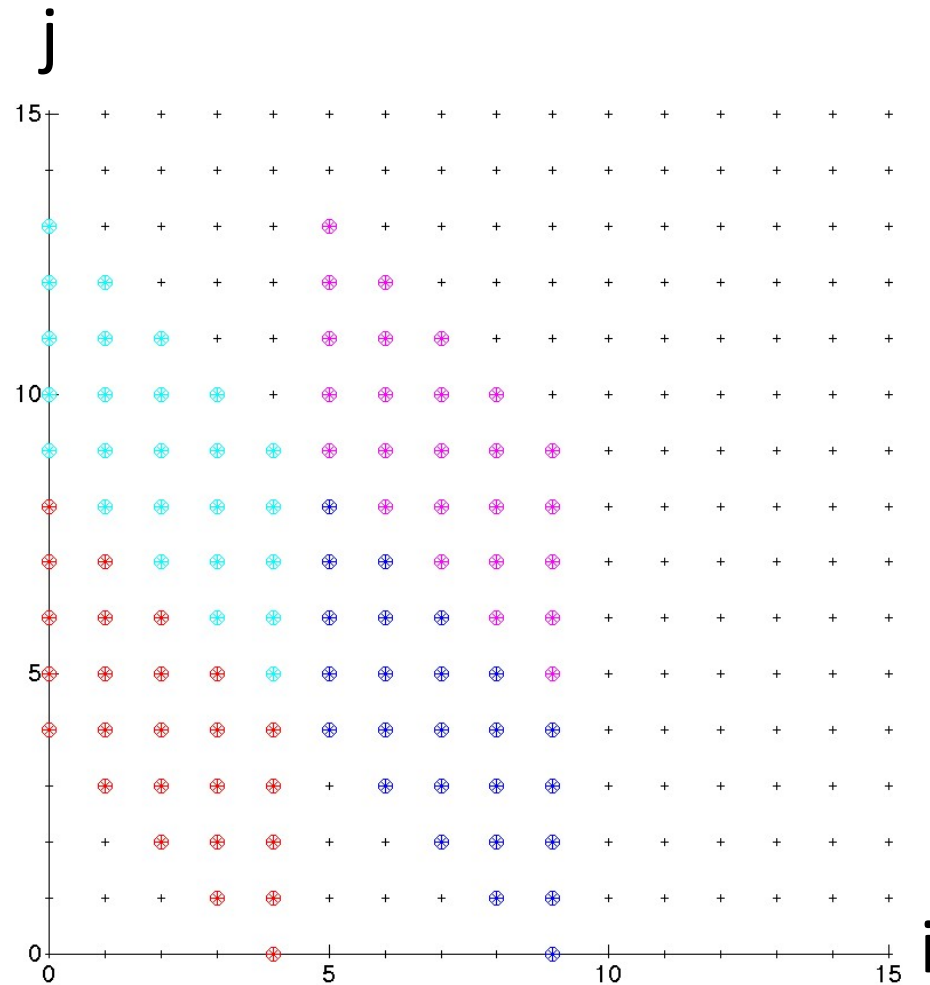
Optimal tiling for “slanted” n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(i+j)
```



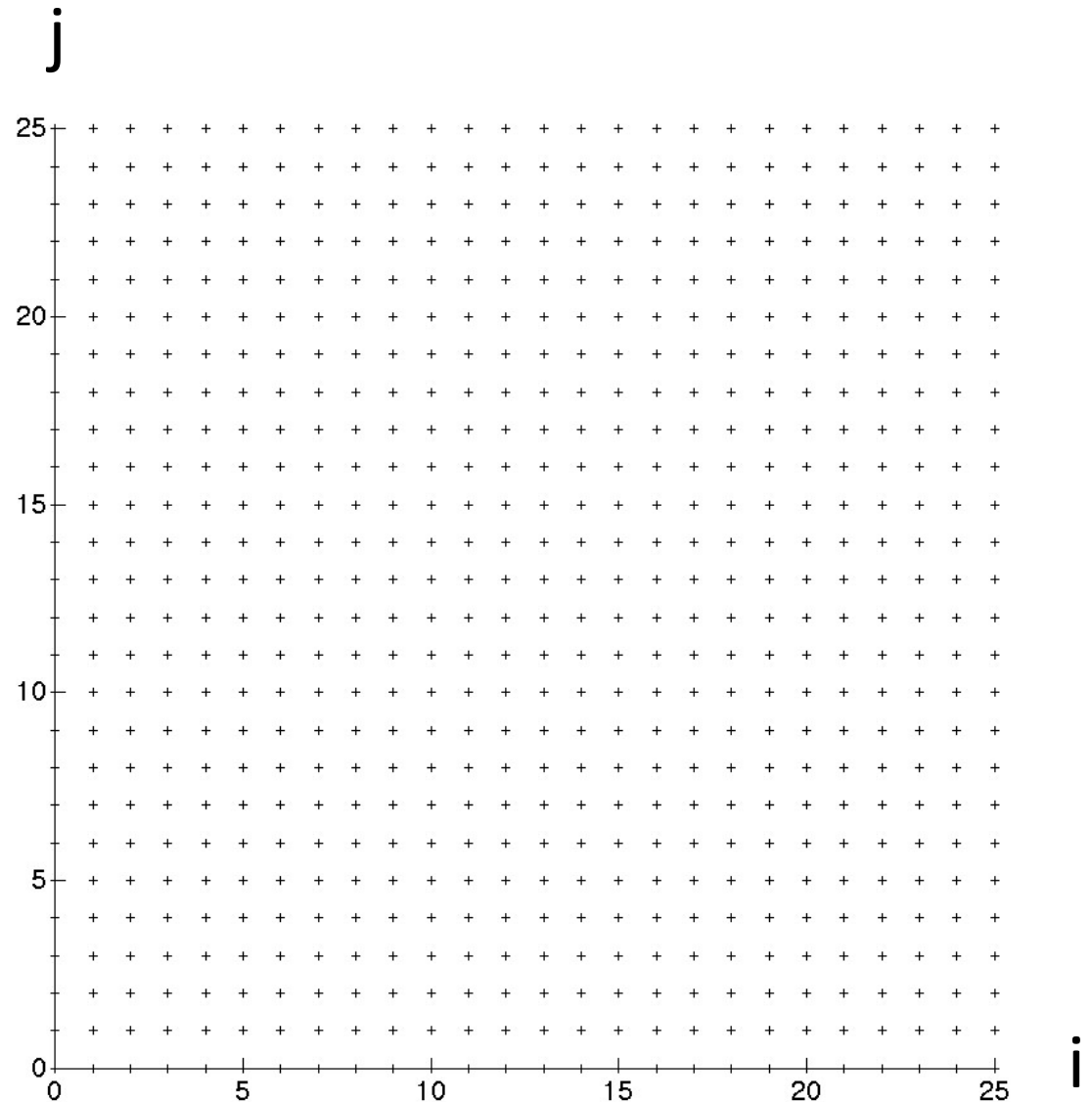
Optimal tiling for “slanted” n-body

```
for i = 0:n  
  for j = 0:n  
    access A(i), B(i+j)
```



Optimal tiling for “twisted” n-body

```
for i = 0:n  
  for j = 0:n  
    access A(3*i-j),  
           B(i-2*j)
```



Optimal tiling for “twisted” n-body

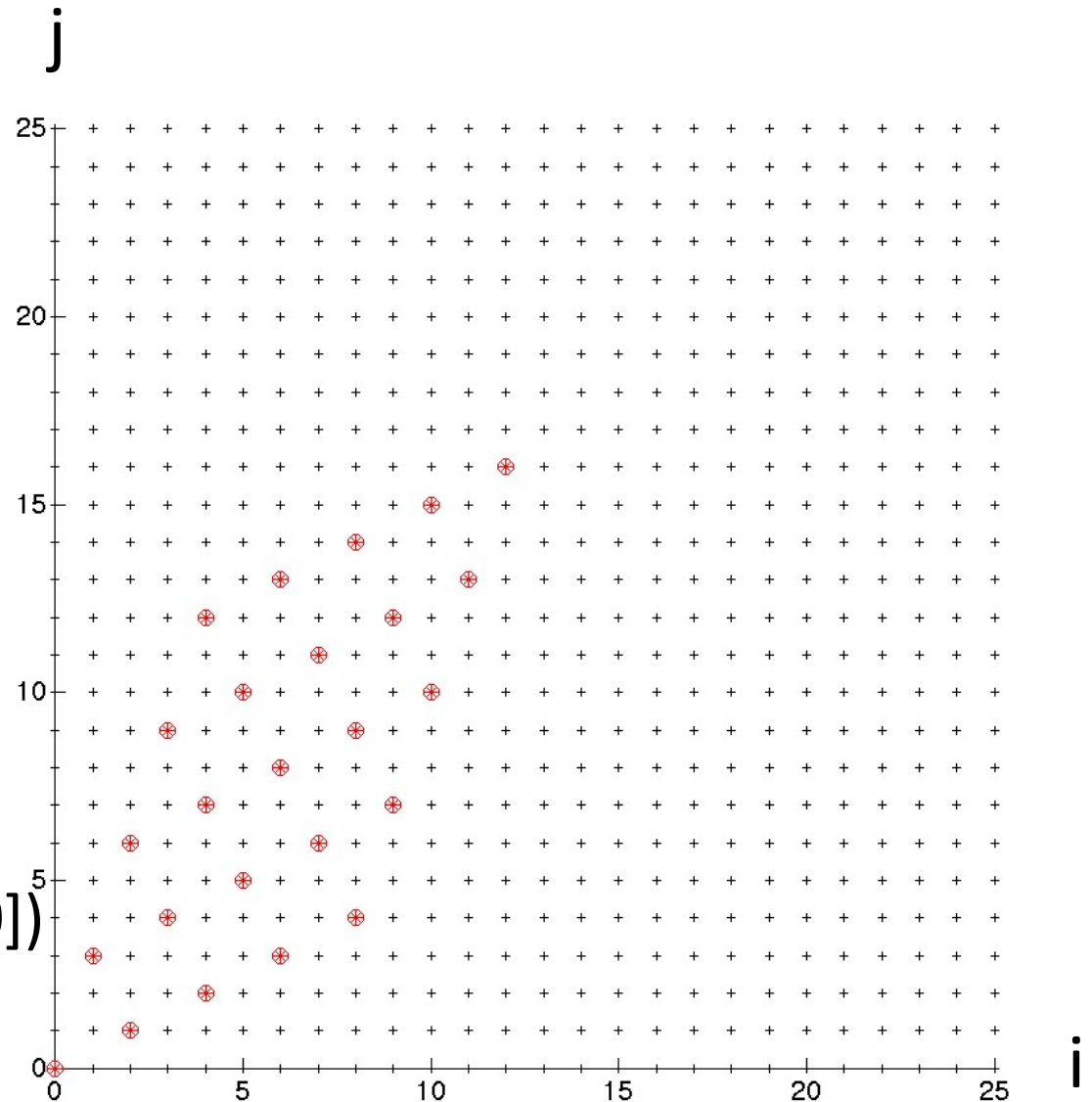
```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```

Tiling:

Read 5 entries of A:
A([0,5,10,15,20])

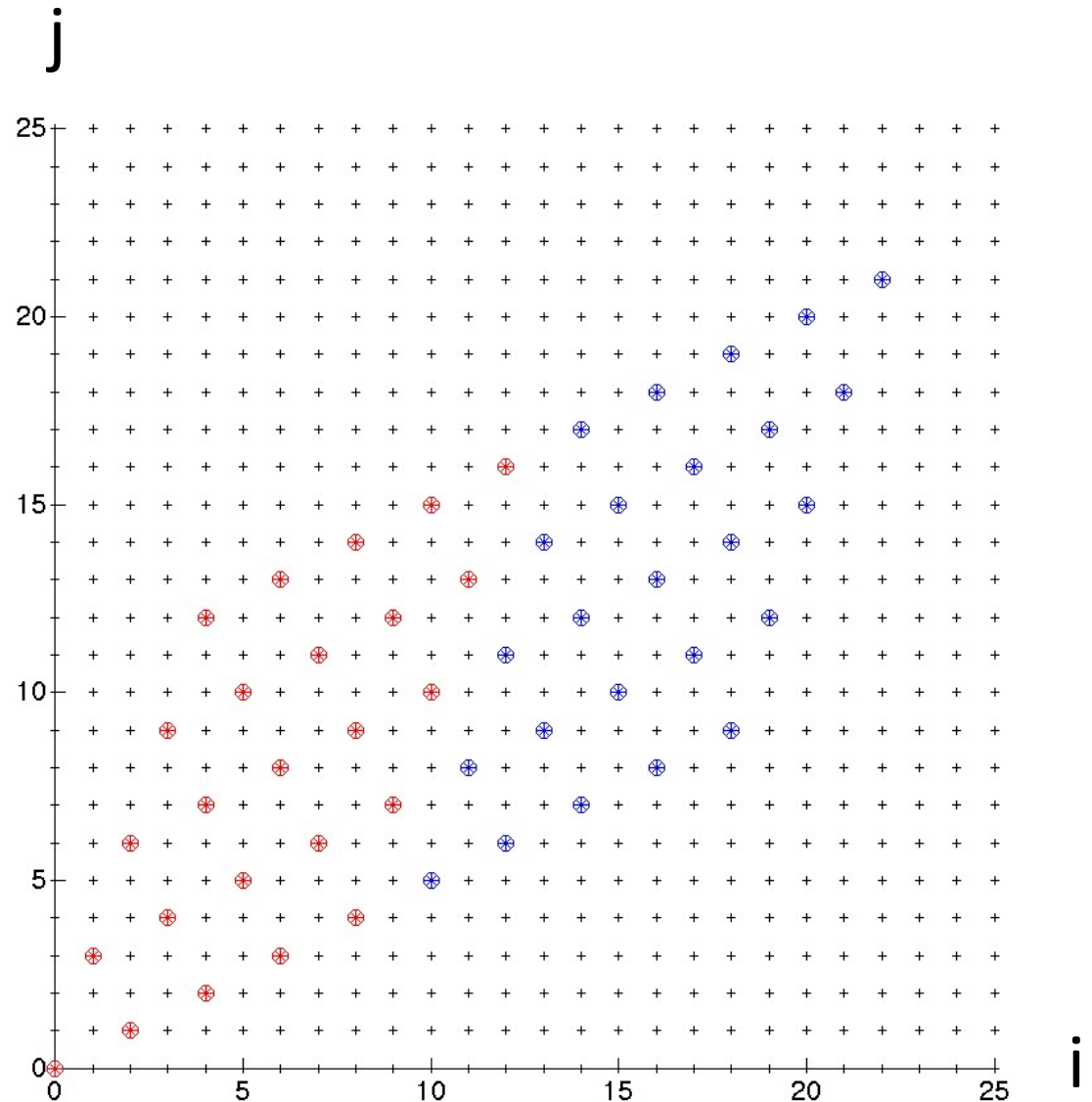
Read 5 entries of B:
B([0,-5,-10,-15,-20])

Perform $5^2 = 25$
loop iterations



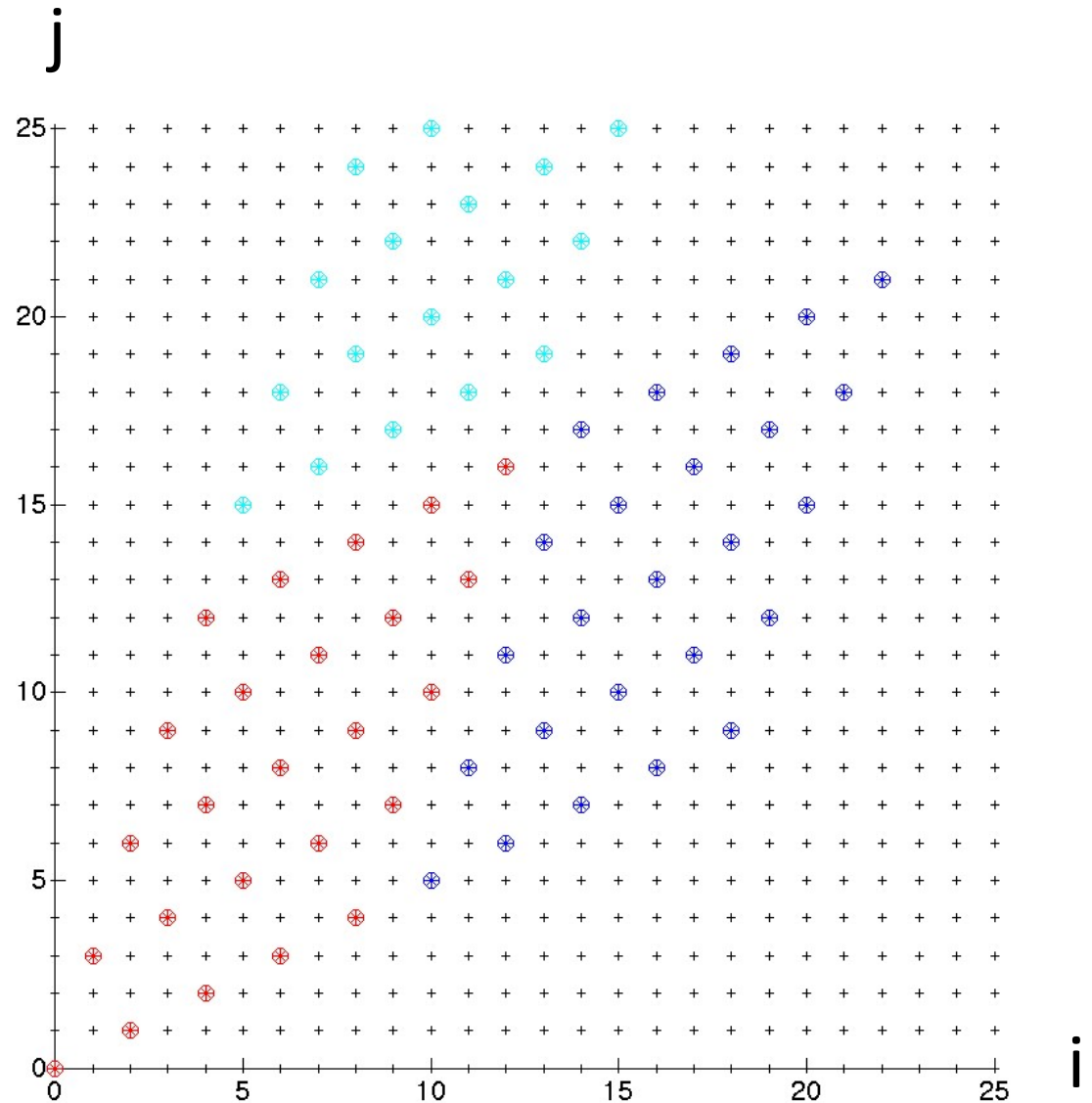
Optimal tiling for “twisted” n-body

```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```



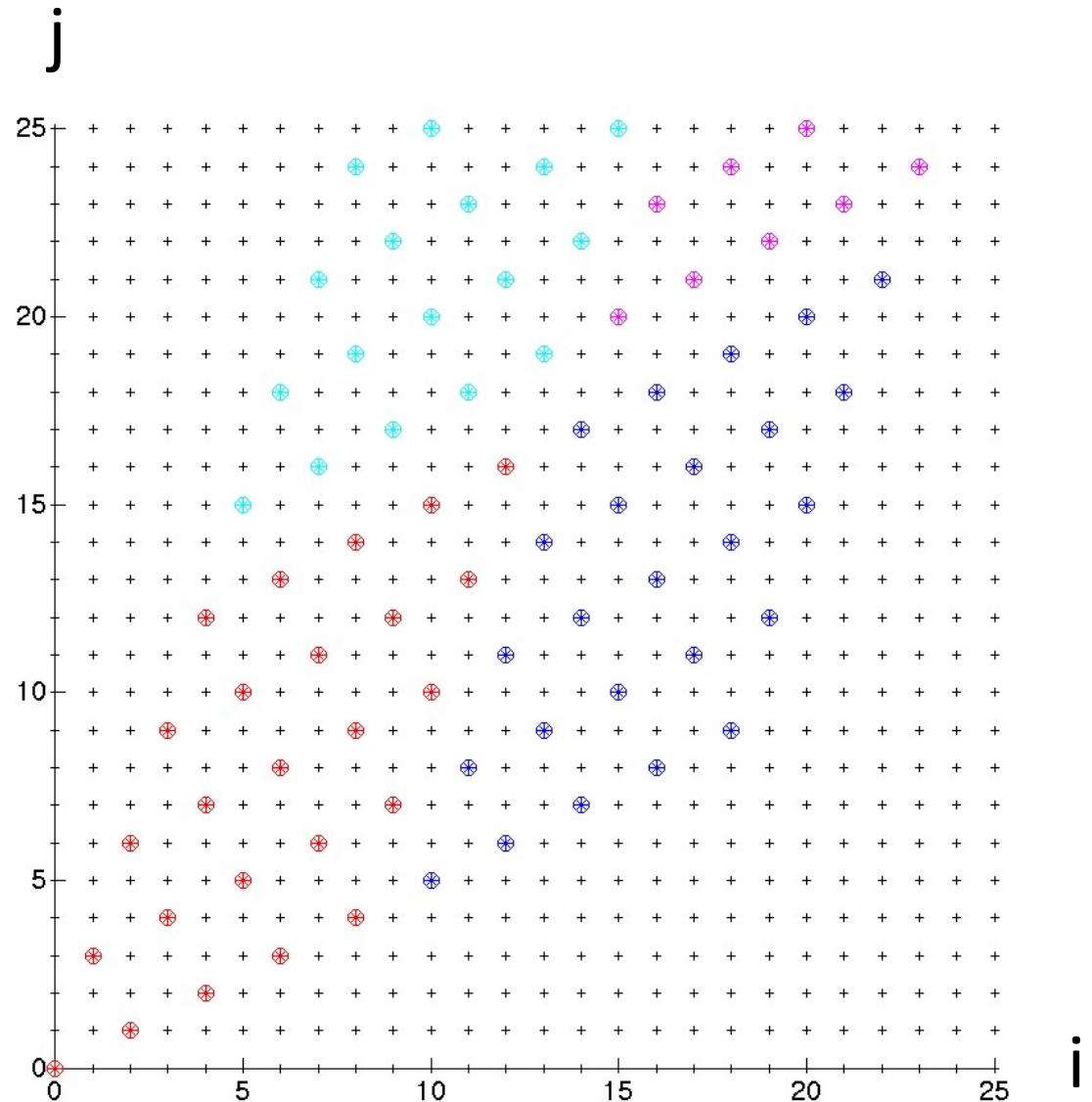
Optimal tiling for “twisted” n-body

```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```



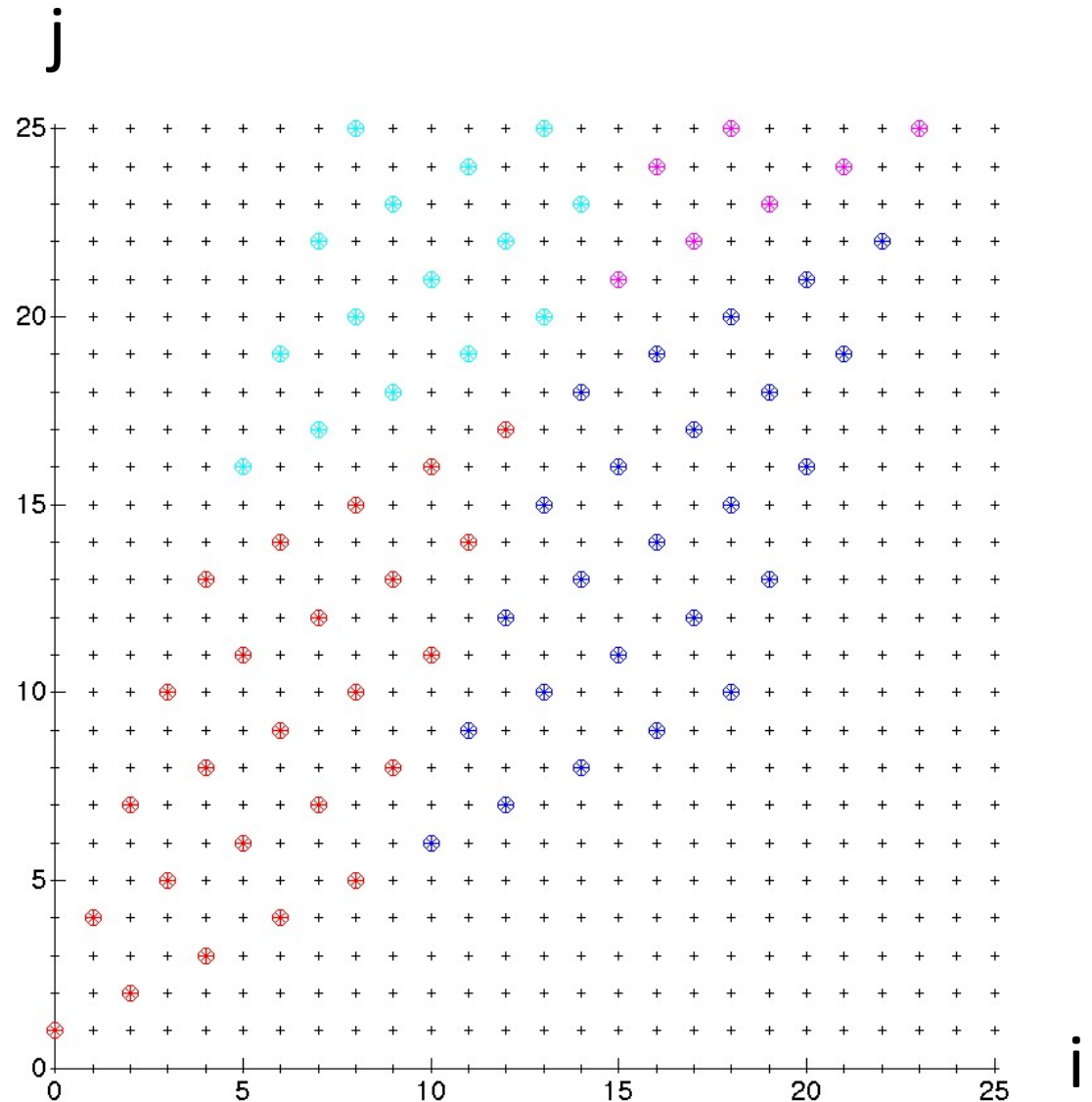
Optimal tiling for “twisted” n-body

```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```



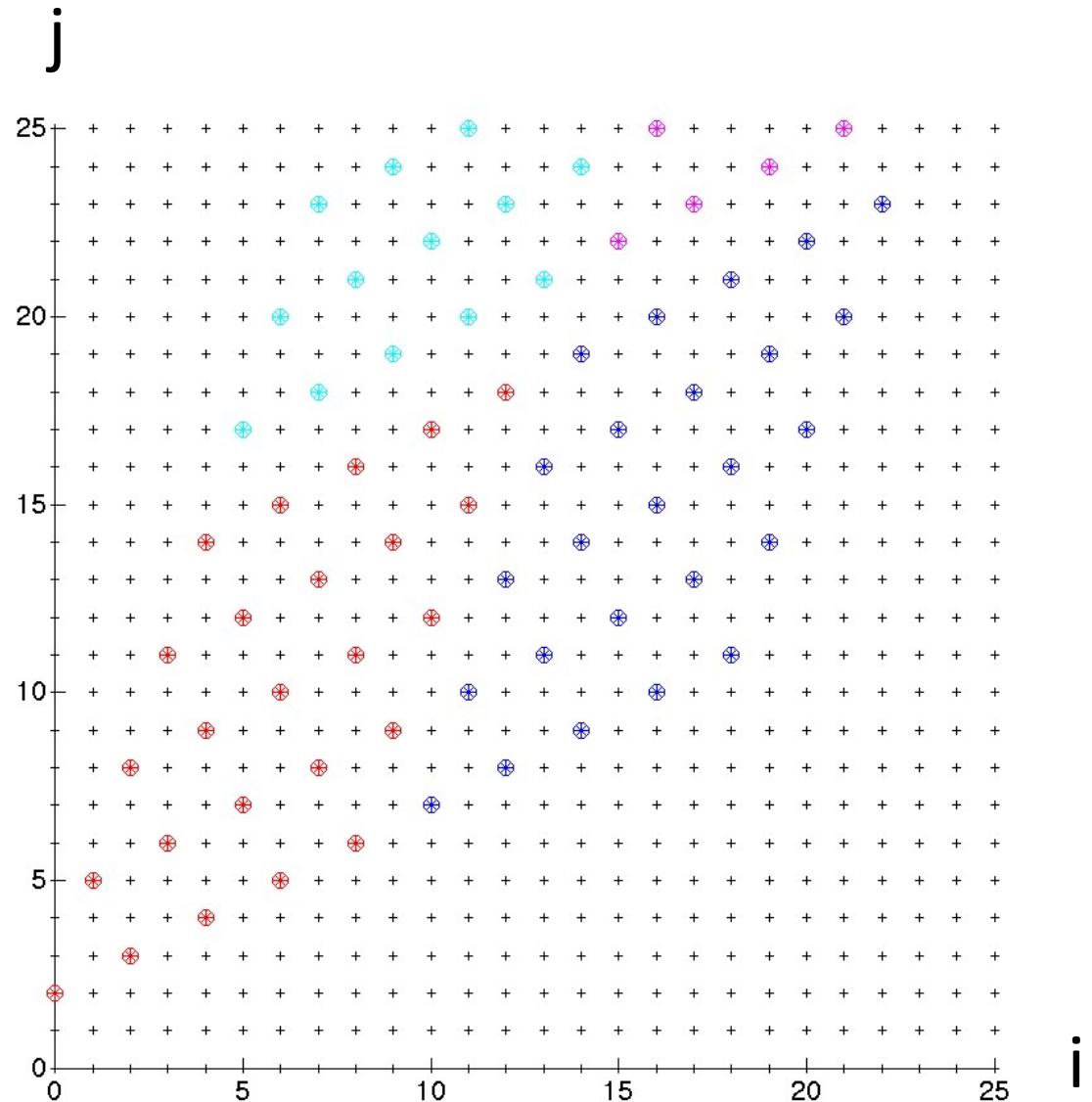
Optimal tiling for “twisted” n-body

```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```



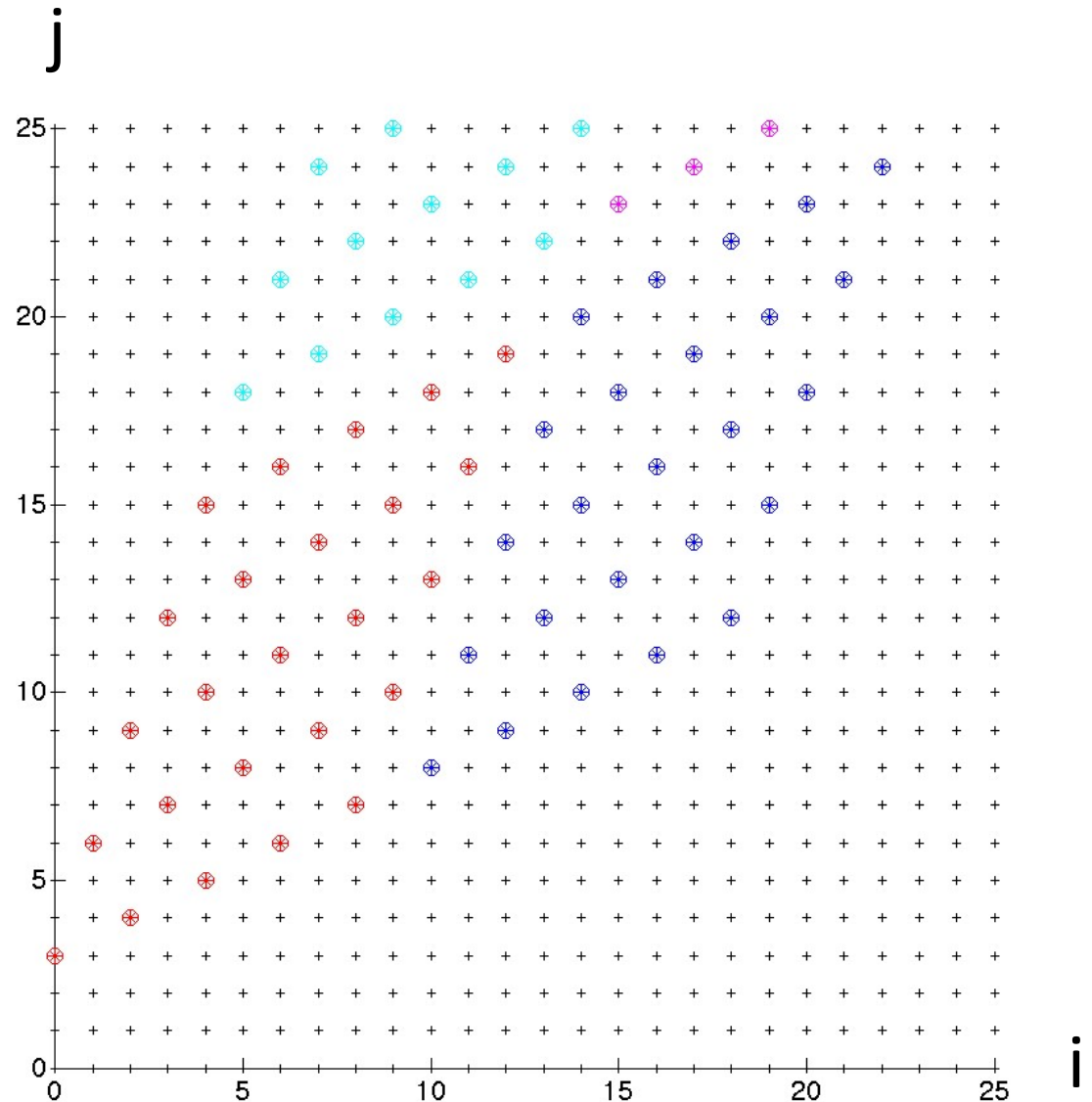
Optimal tiling for “twisted” n-body

```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```



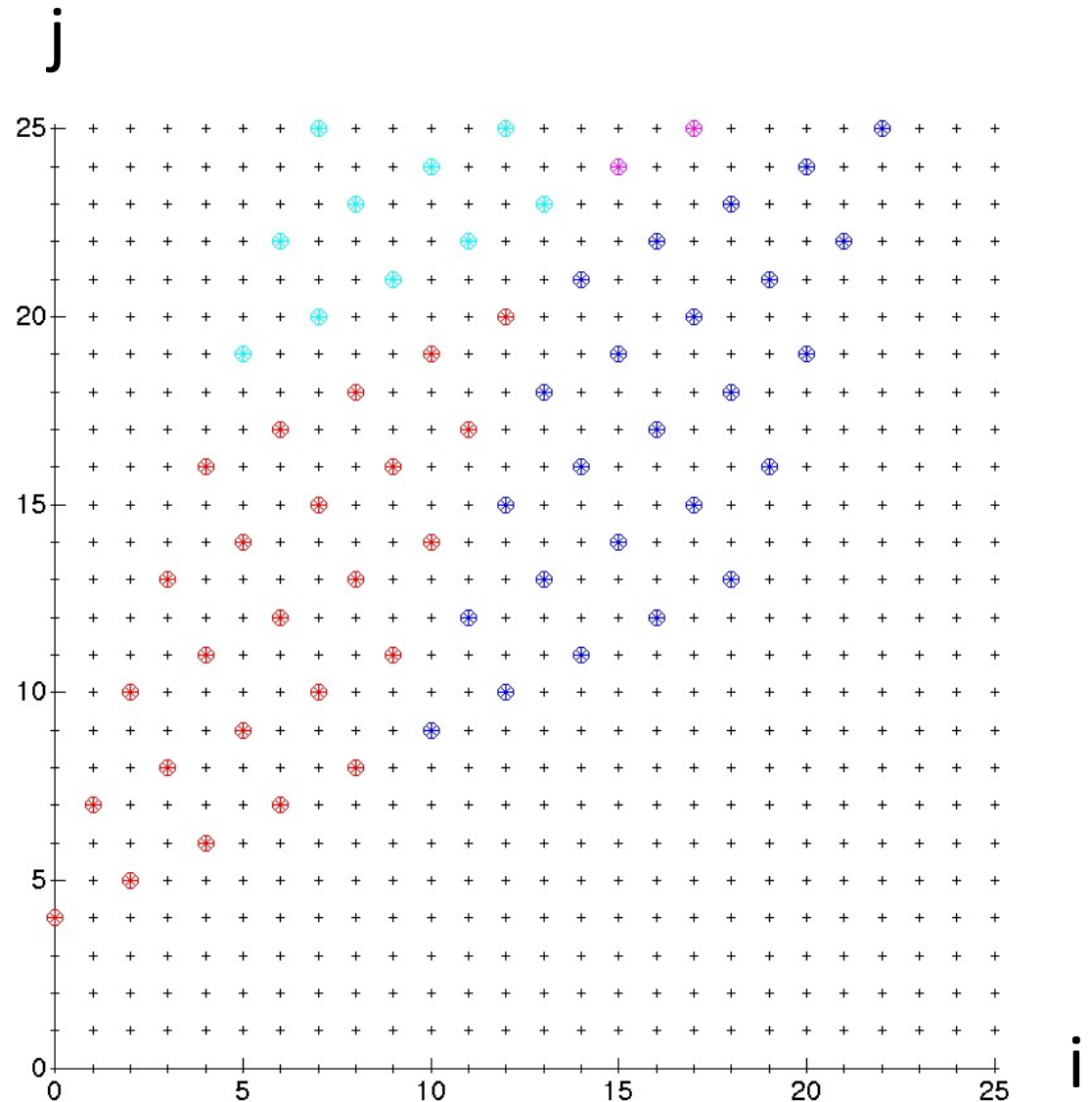
Optimal tiling for “twisted” n-body

```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```



Optimal tiling for “twisted” n-body

```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```



Optimal tiling for “twisted” n-body

```
for i = 0:n
  for j = 0:n
    access A(3*i-j),
           B(i-2*j)
```

