# PROGRAMMING FOR PARALLELISM AND LOCALITY WITH HTA's

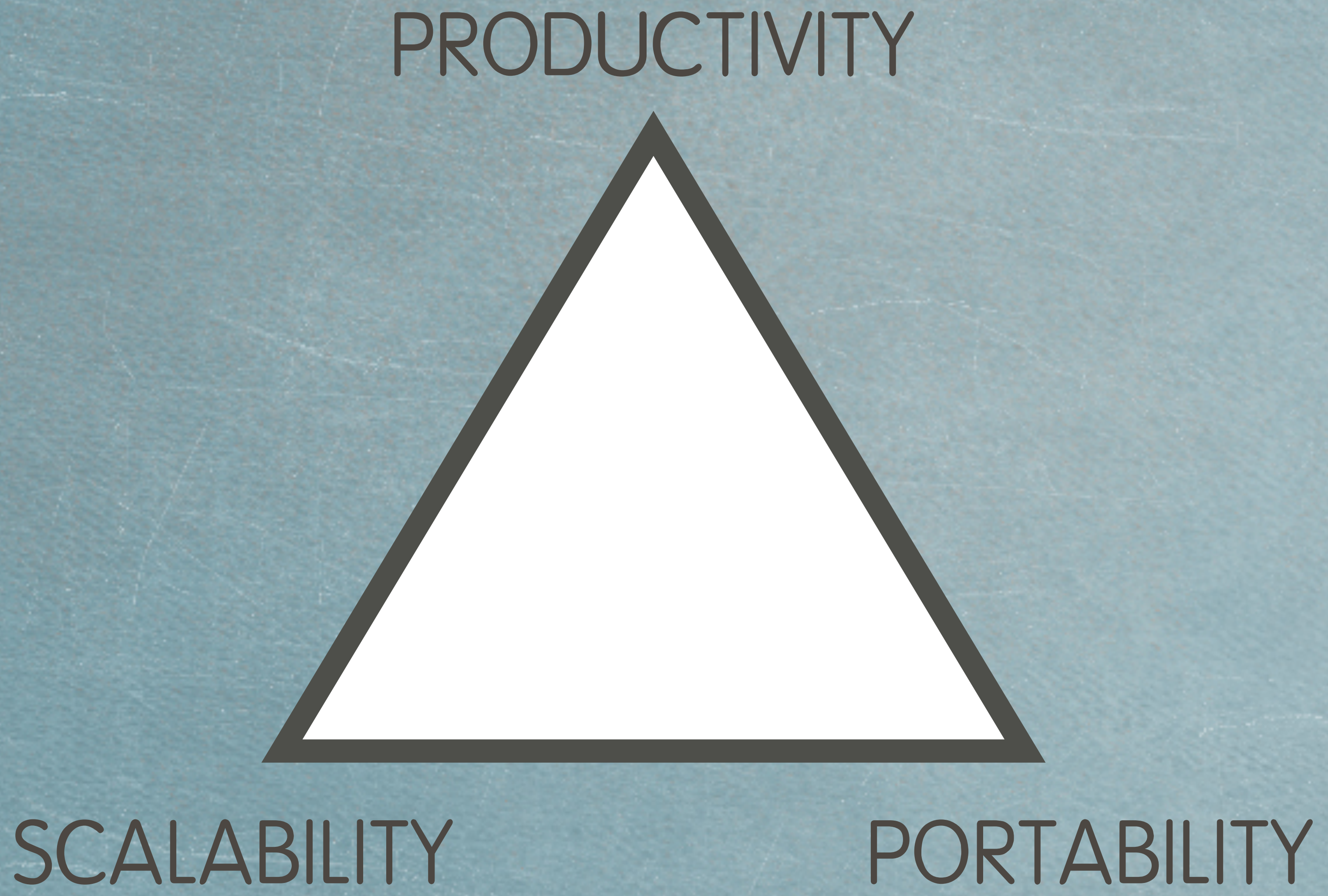
PAPER PUBLISHED AT PPOPP MARCH 2006

PRESENTATION BY ROMAN FRIGG

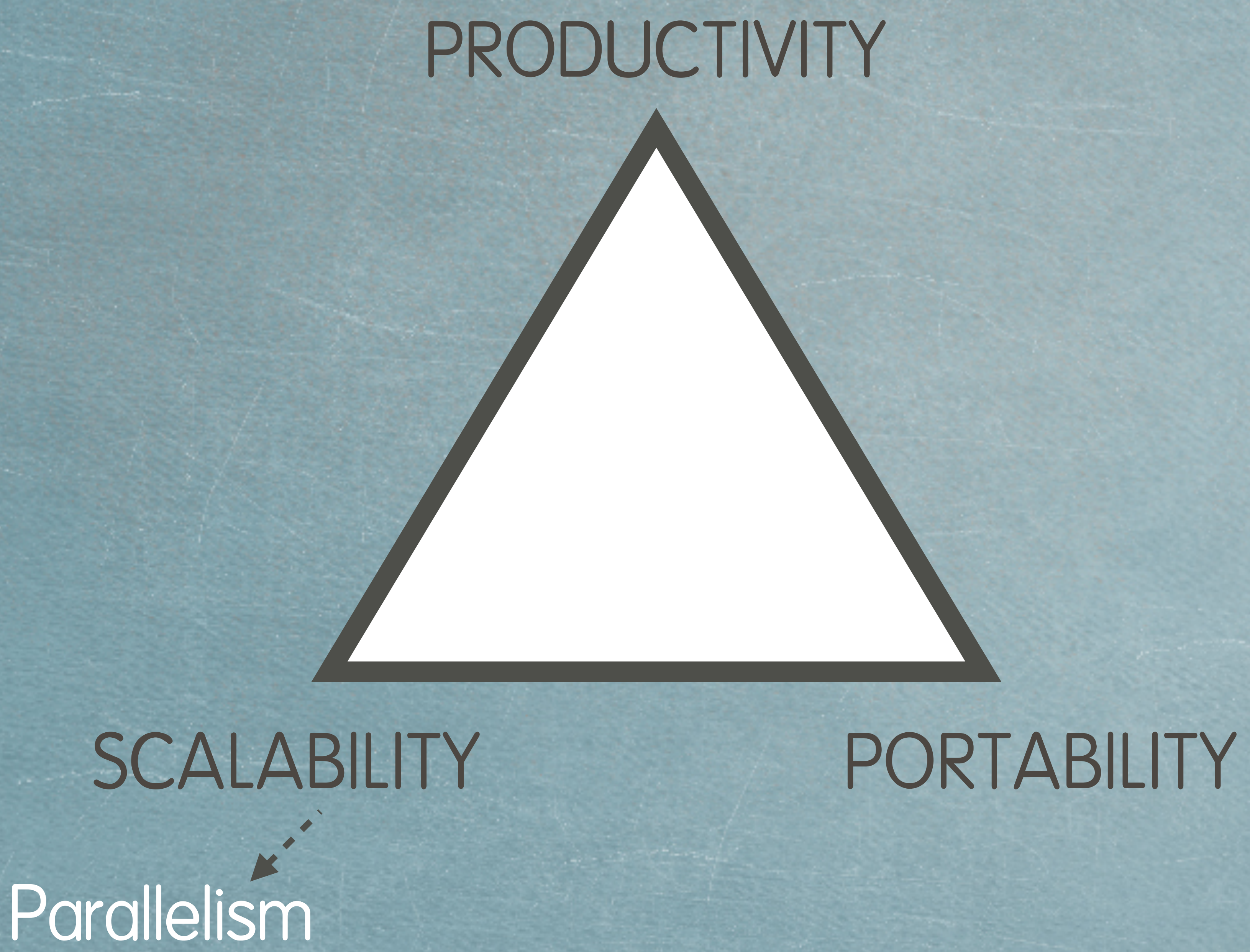
Written at UIUC[1], Universidade da Coruna[2] and IBM T.J. Watson Research Center[3] by Ganesh Bikshandi[1], Jia Guo, Daniel Hoeflinger[1], Gheorghe Almasi[3], Basilio B. Fraguela[2], María J. Garzarán[1], David Padua[1] and Christoph von Praun[3]

30 NOV

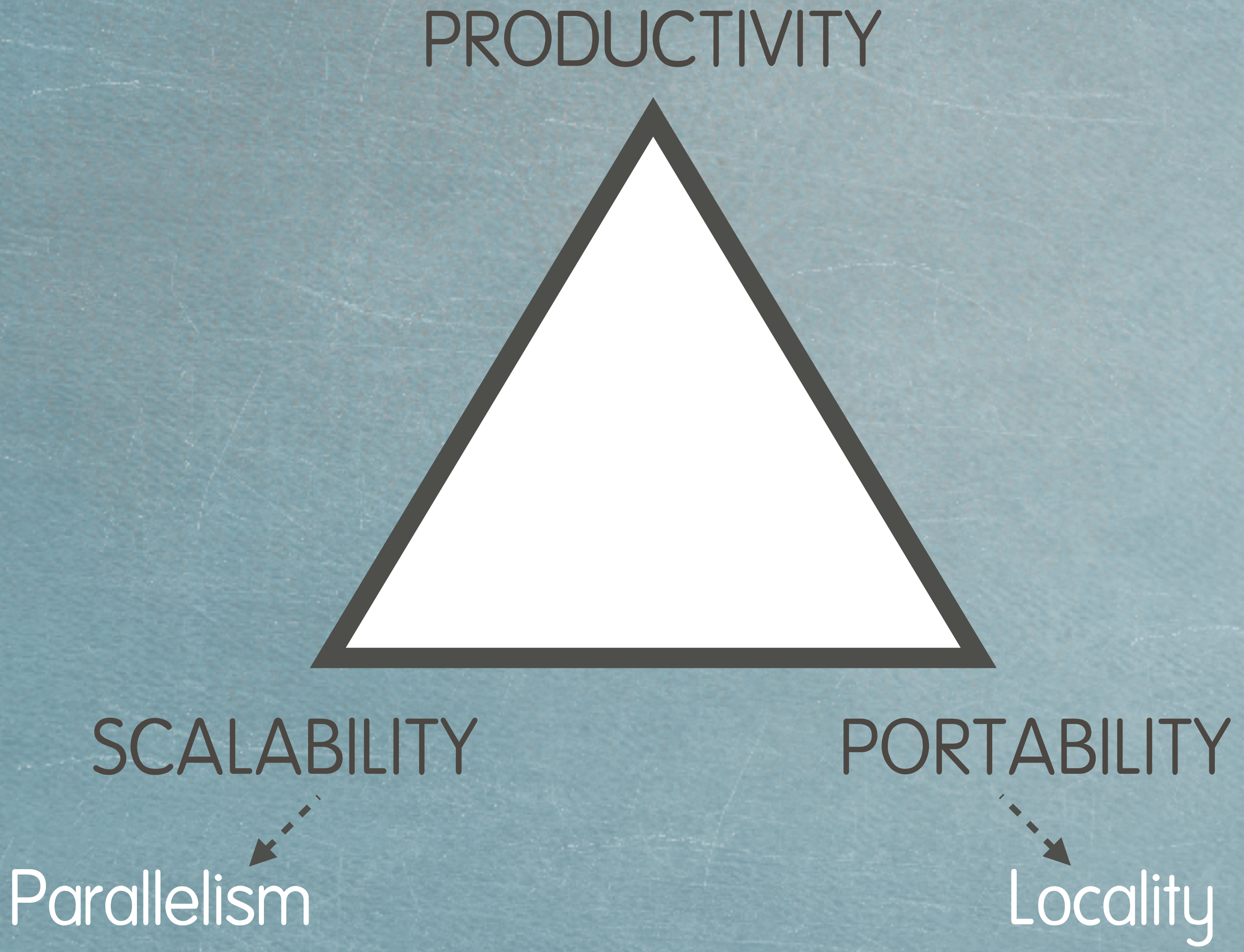
# PROGRAMMING TODAY'S SYSTEMS



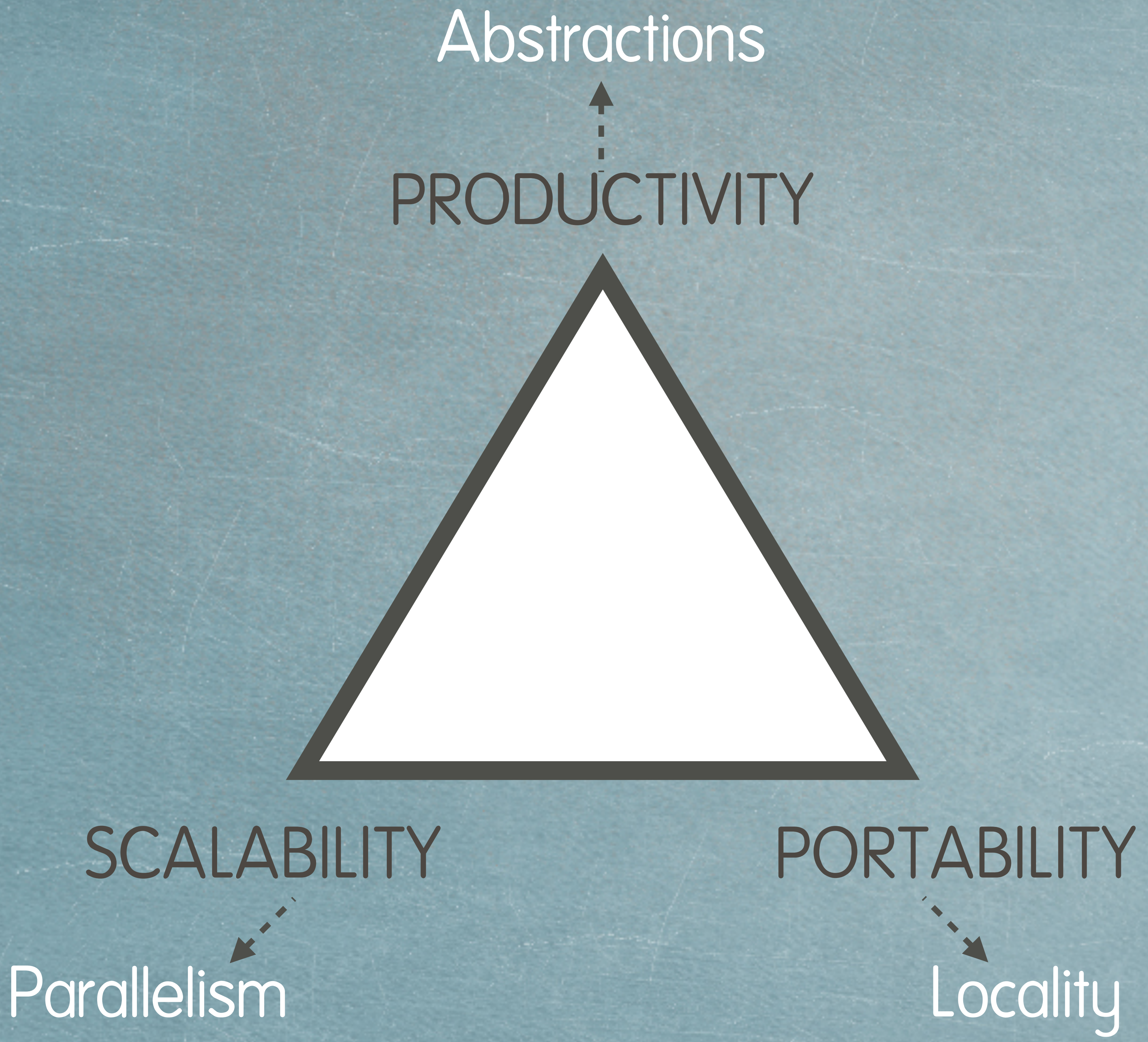
# PROGRAMMING TODAY'S SYSTEMS



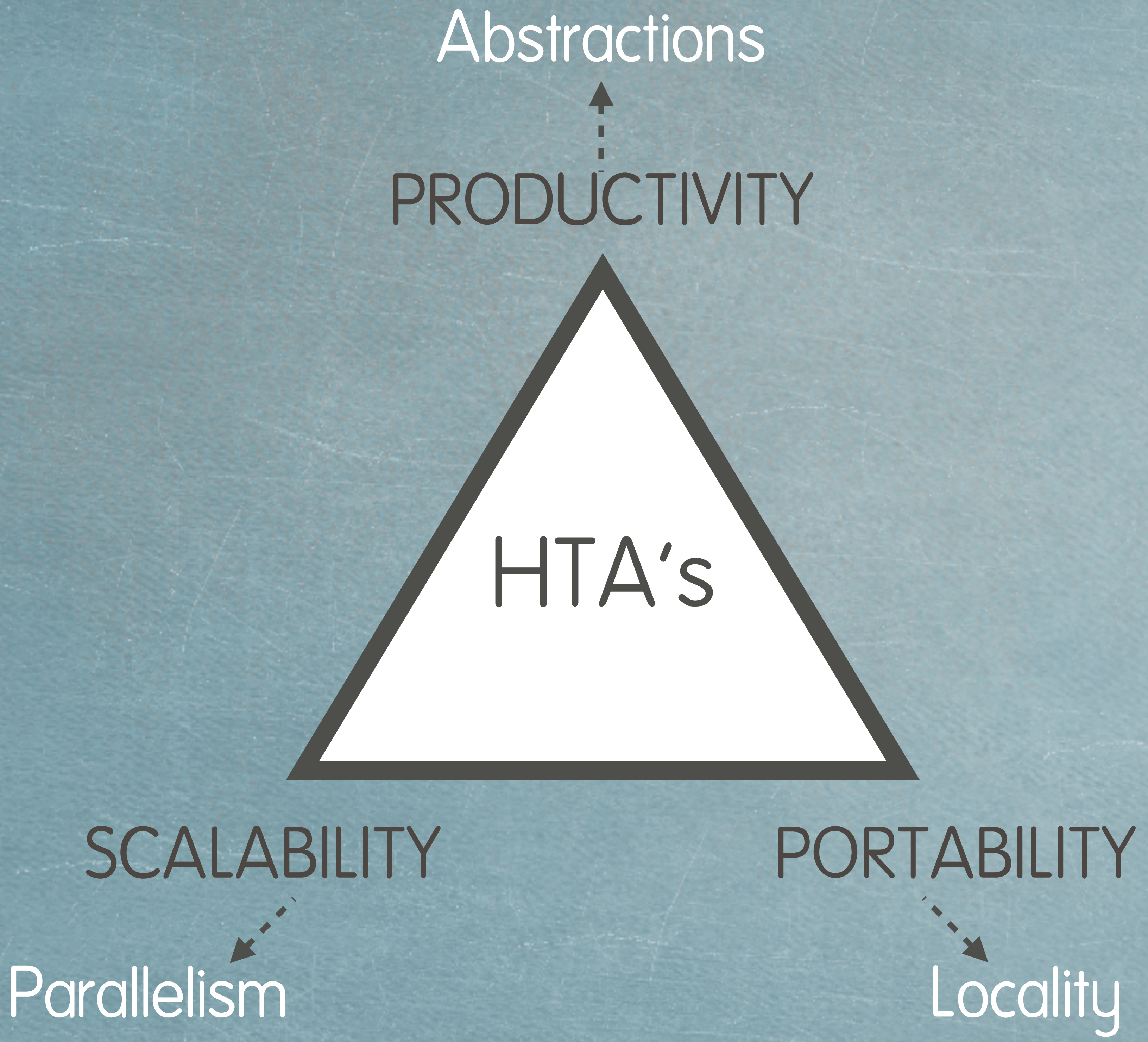
# PROGRAMMING TODAY'S SYSTEMS



# PROGRAMMING TODAY'S SYSTEMS

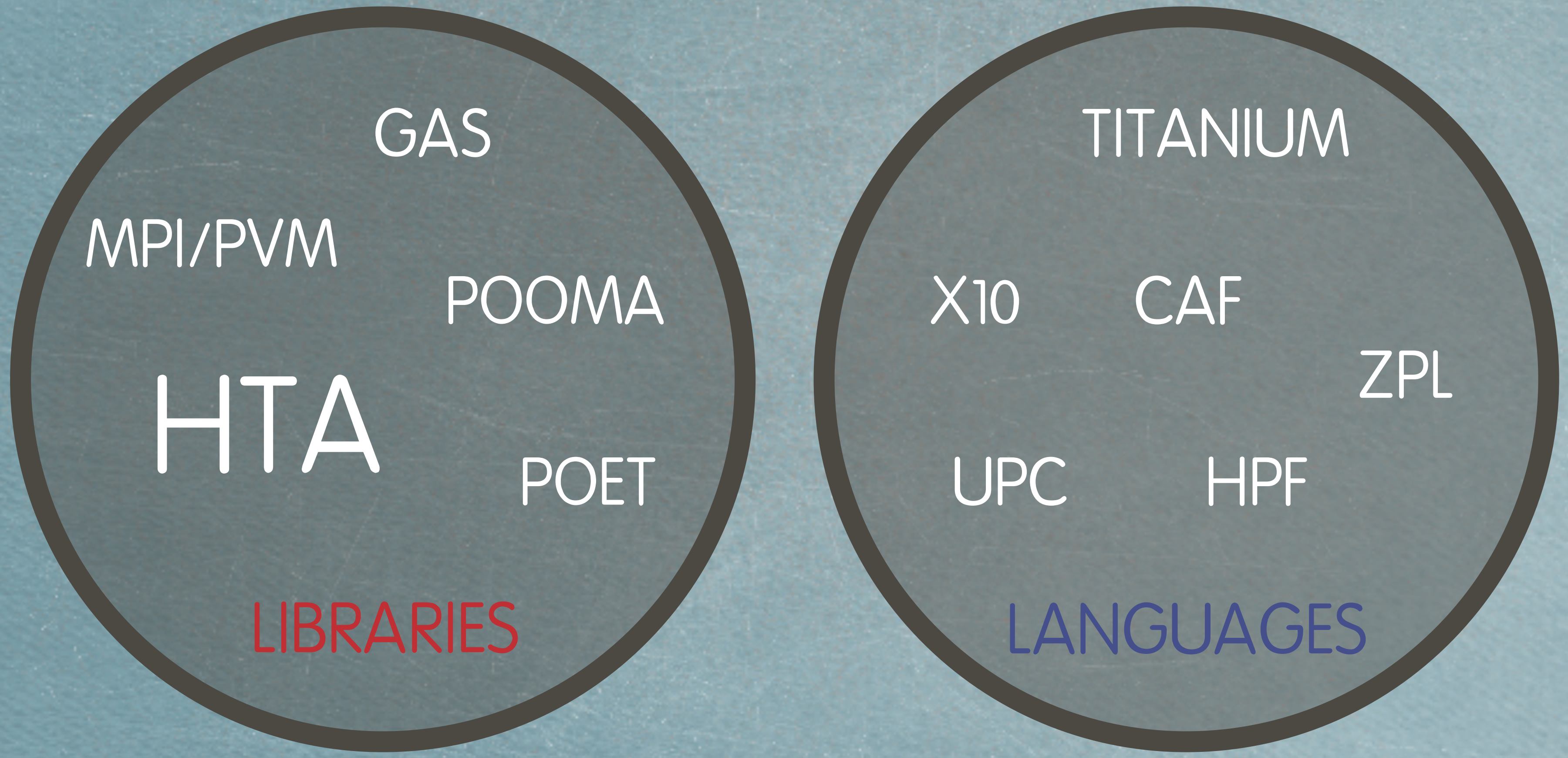


# PROGRAMMING TODAY'S SYSTEMS

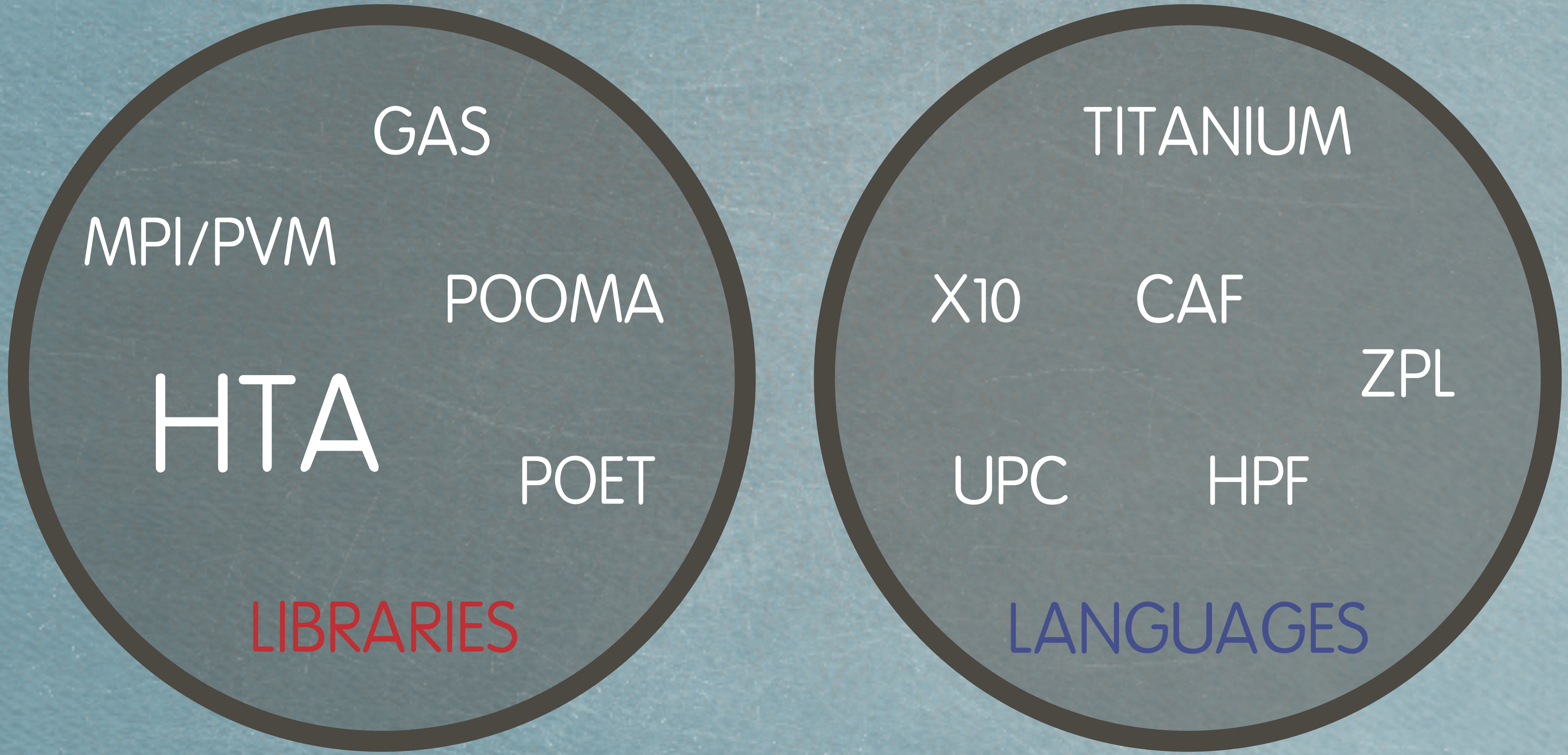


# CLASSIFICATION

# CLASSIFICATION



# CLASSIFICATION




- Library
- Matlab & C++
- Single threaded, global view

# TALK OVERVIEW

INTRO

1

# TALK OVERVIEW

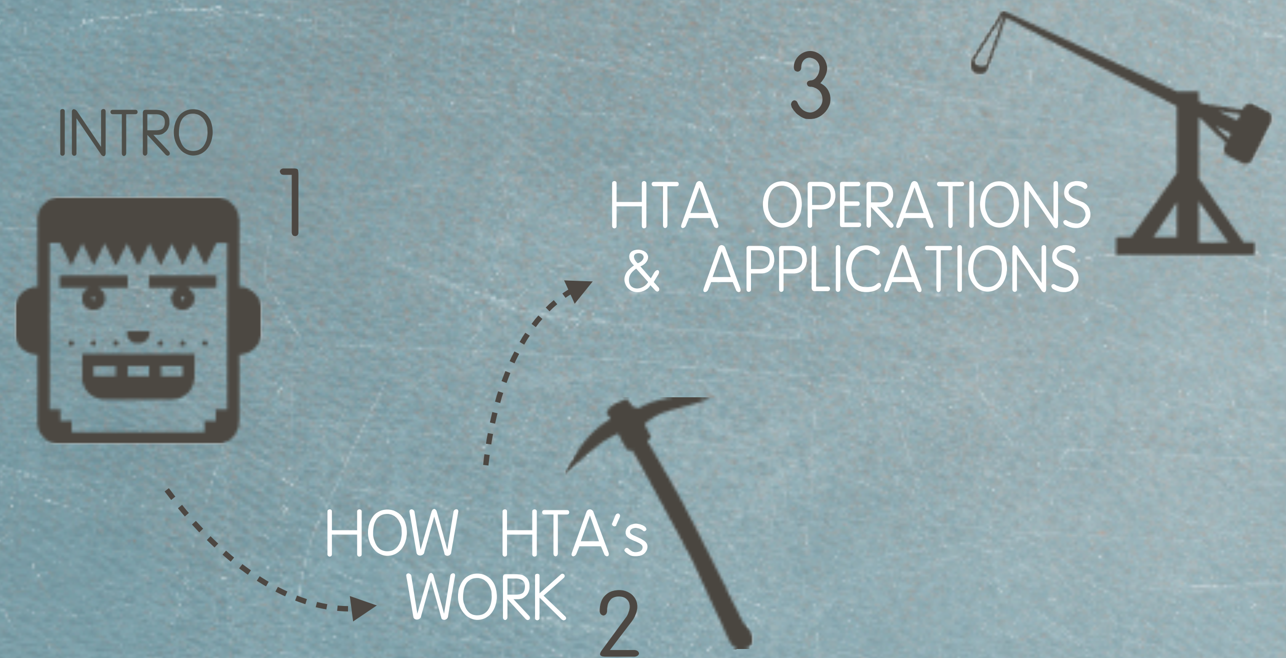
INTRO

1

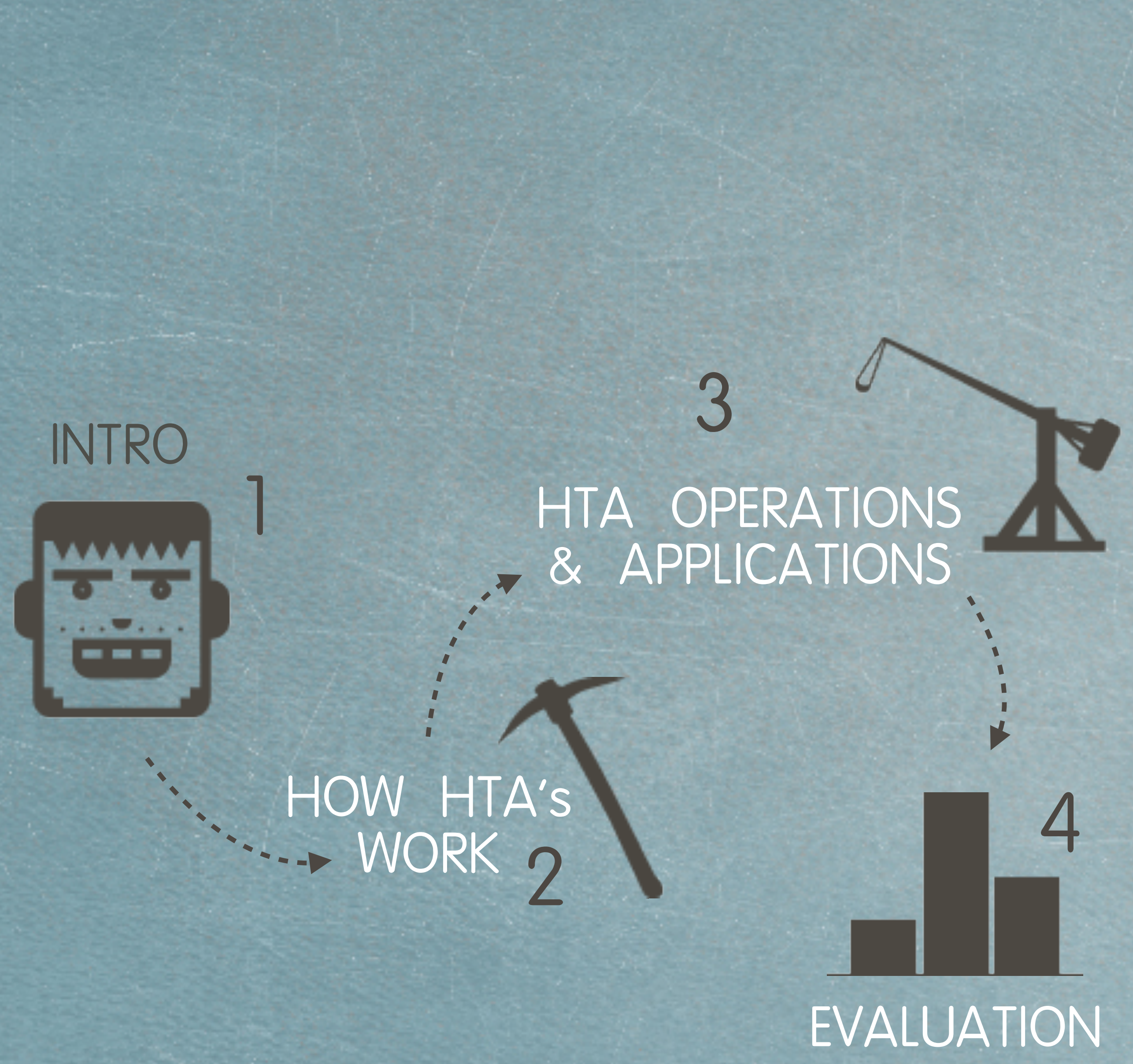
HOW HTA's WORK

2

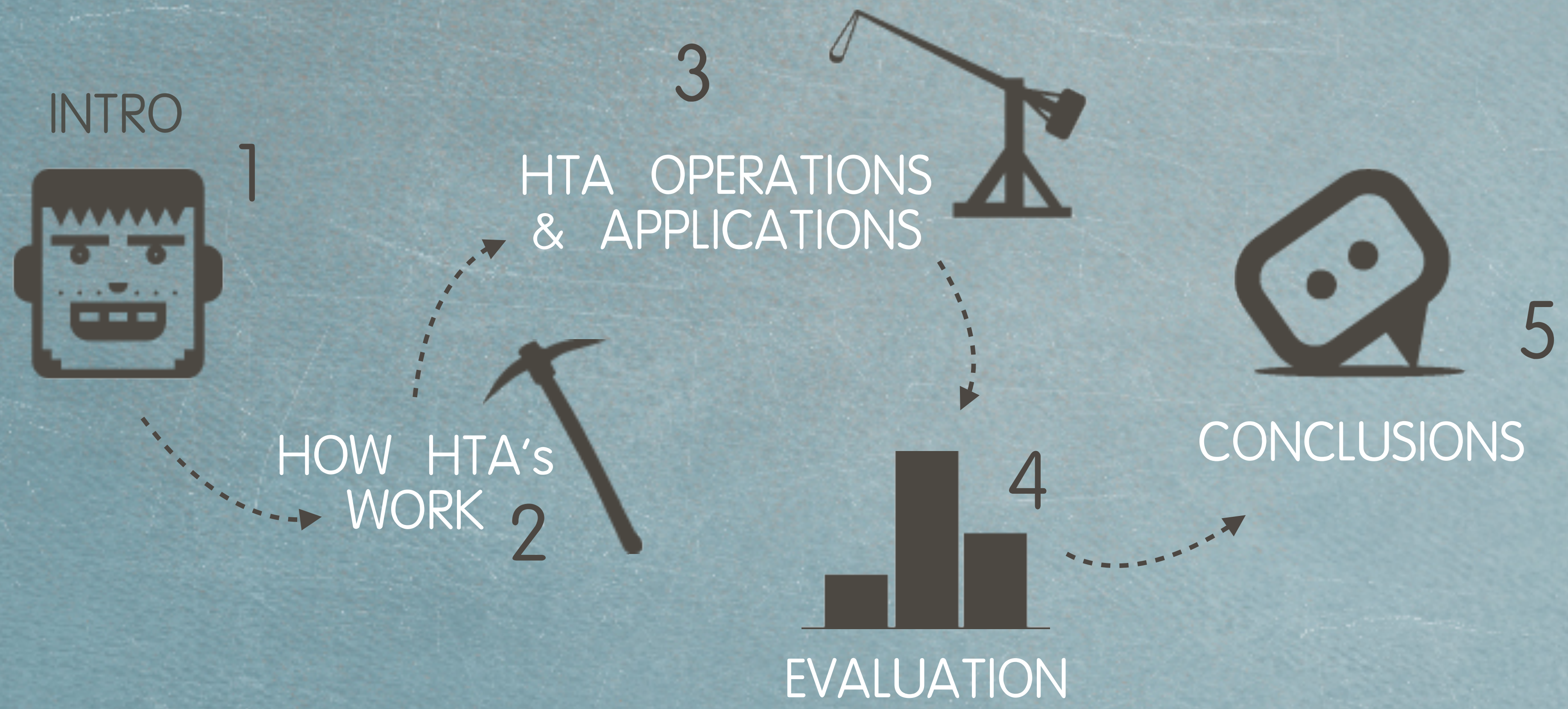
# TALK OVERVIEW



# TALK OVERVIEW



# TALK OVERVIEW



# RECURSIVE TILING

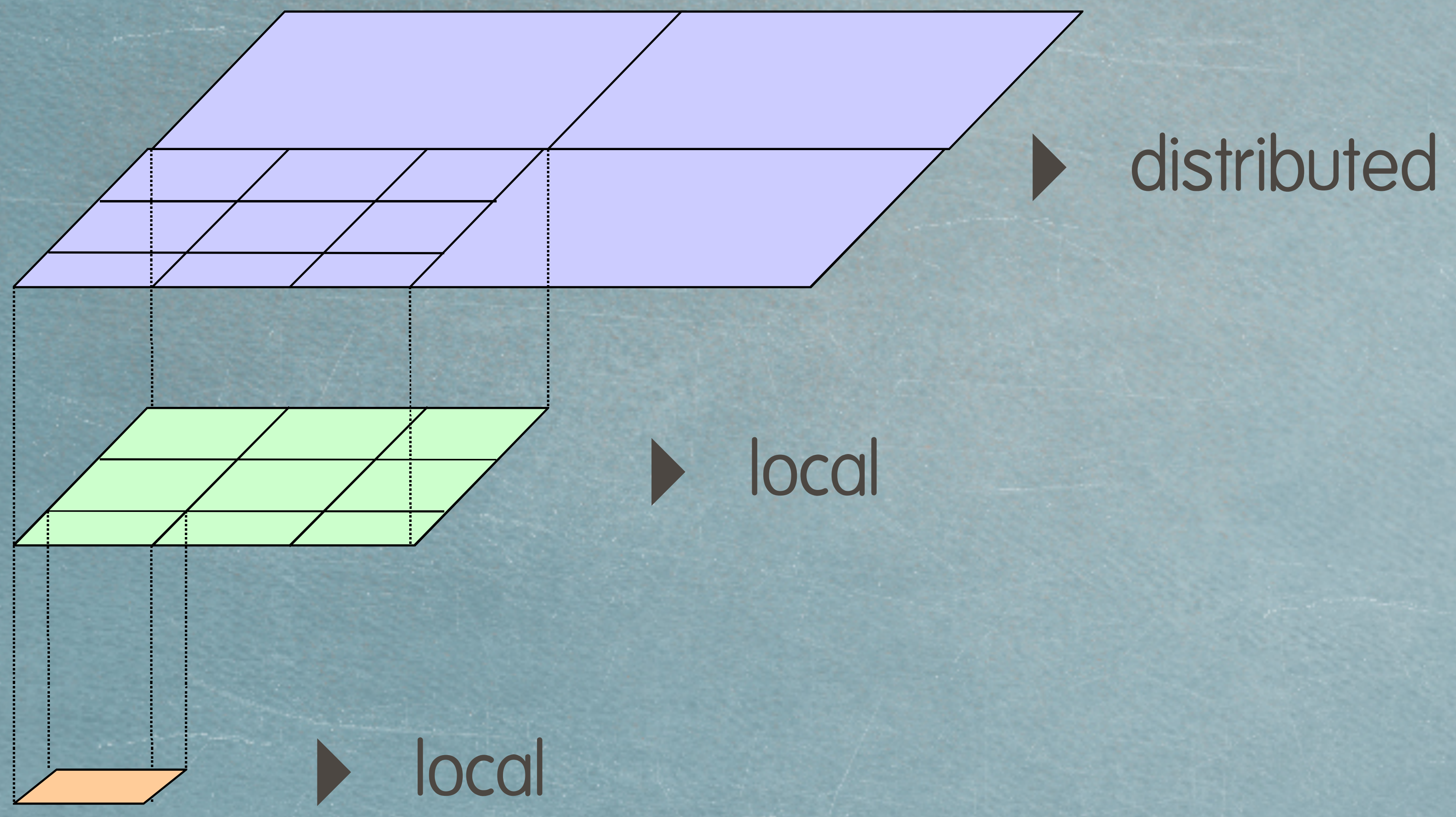



image source: paper

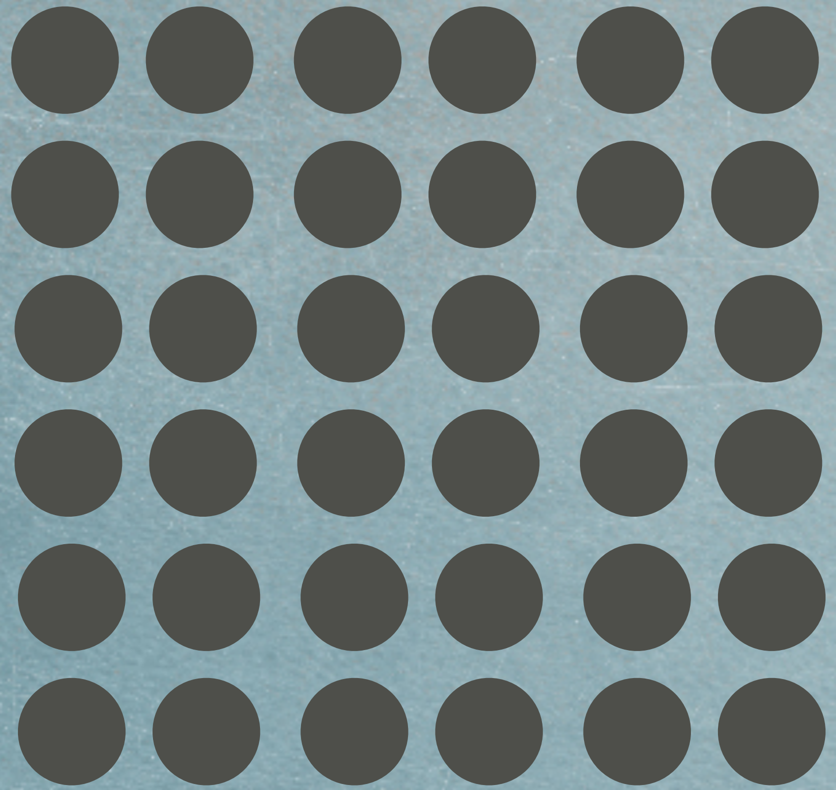
# CONSTRUCT HTA FROM 6x6 MATRIX

```
T1 = hta(   ,{        ,        } )
```

# CONSTRUCT HTA FROM 6x6 MATRIX

```
T1 = hta( M ,{          ,          } )
```

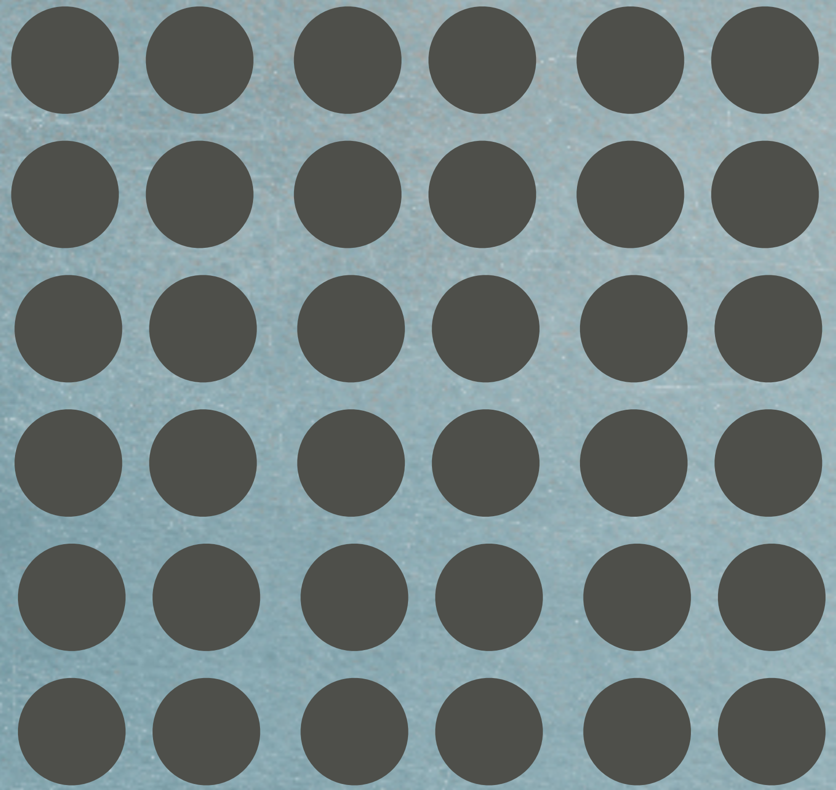
# CONSTRUCT HTA FROM 6x6 MATRIX



HOW HTA's WORK 2

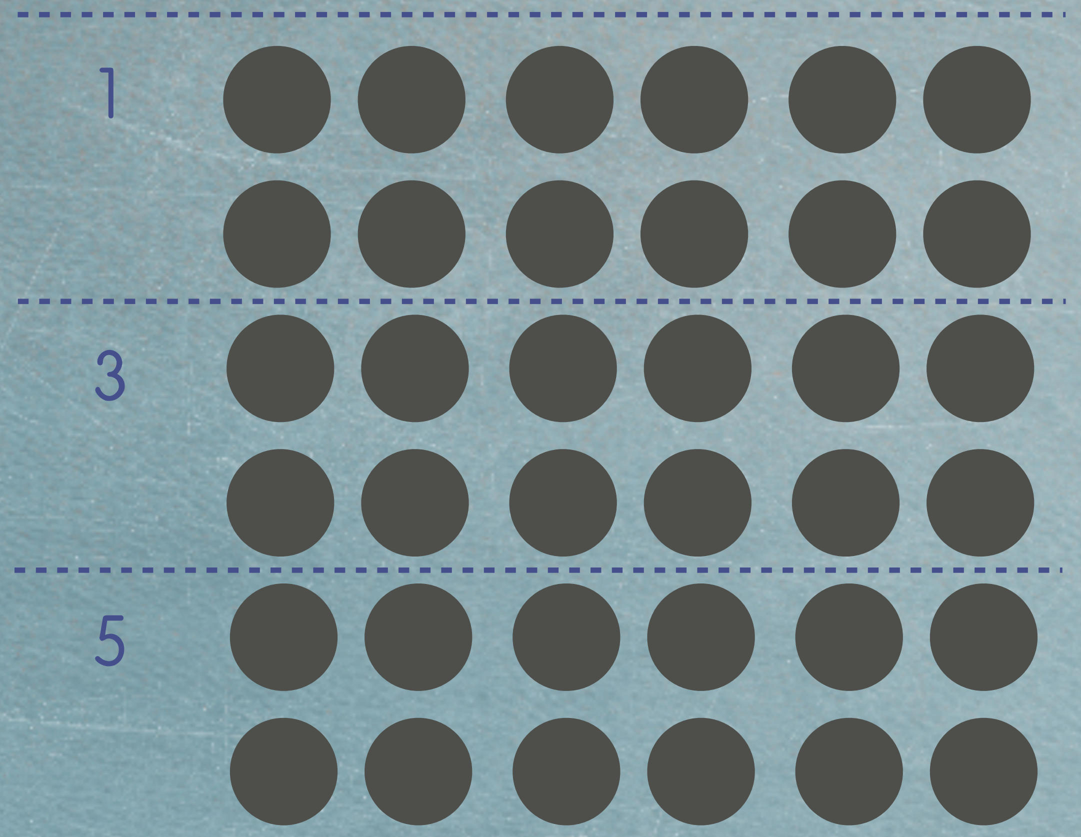
```
T1 = hta( M ,{[1 3 5],          } )
```

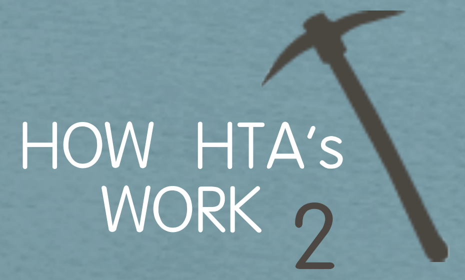
# CONSTRUCT HTA FROM 6x6 MATRIX



```
T1 = hta( M ,{[1 3 5],          } )
```

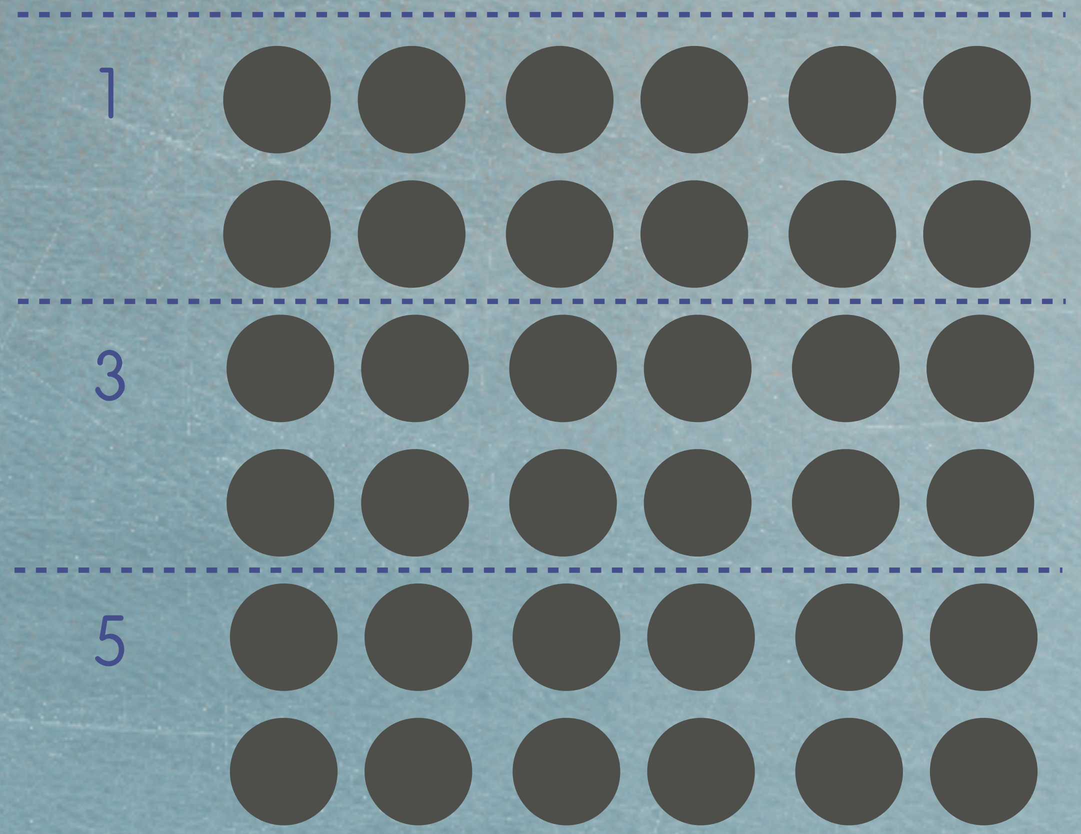
# CONSTRUCT HTA FROM 6x6 MATRIX



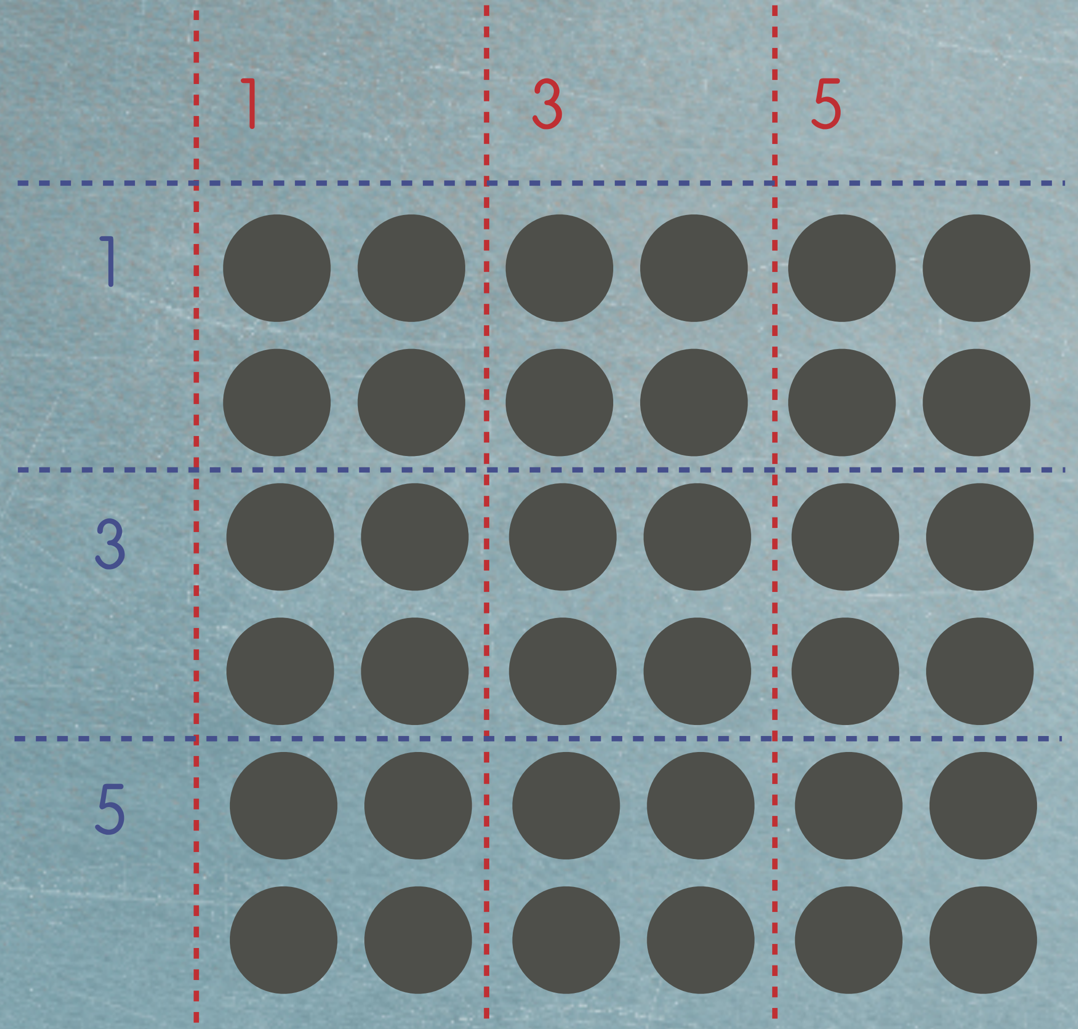
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
```

# CONSTRUCT HTA FROM 6x6 MATRIX



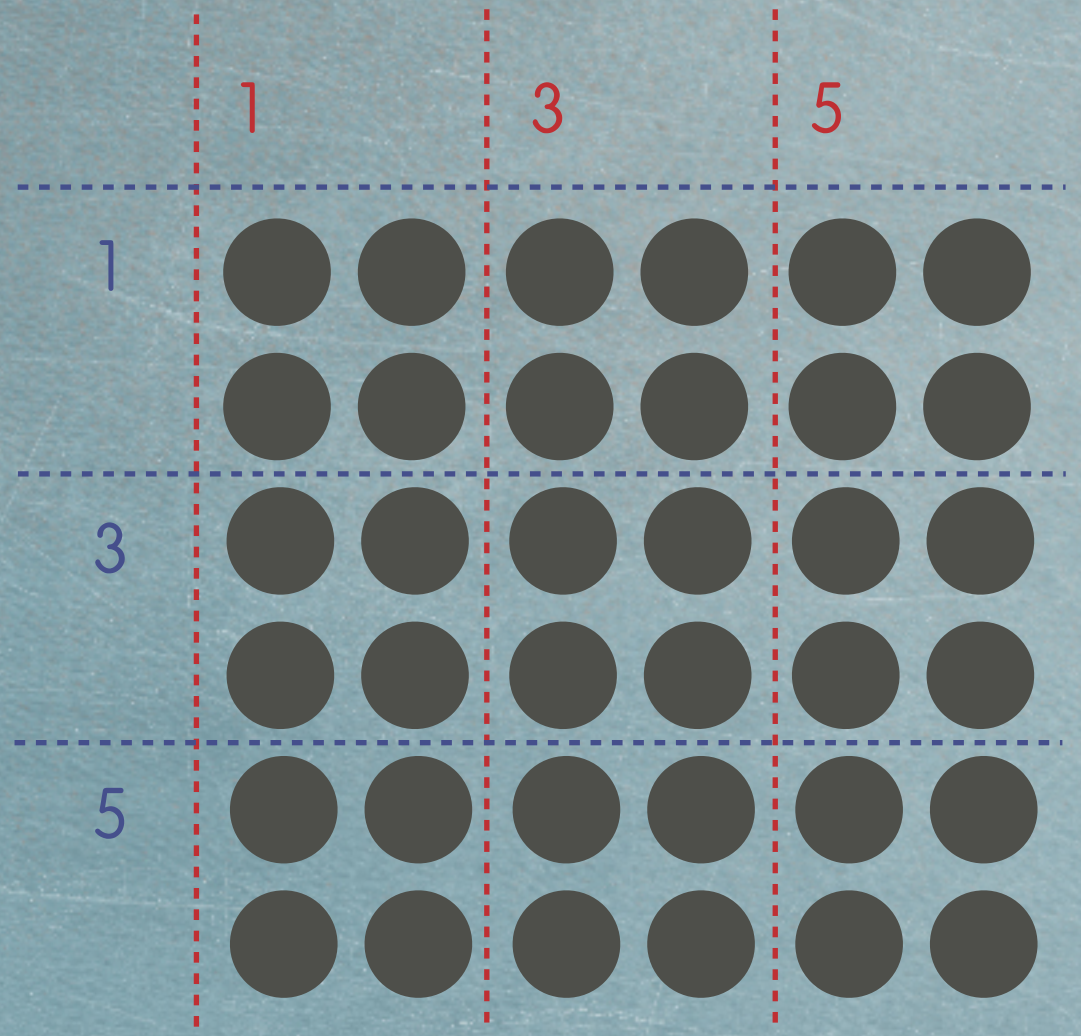
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
```

# CONSTRUCT HTA FROM 6x6 MATRIX



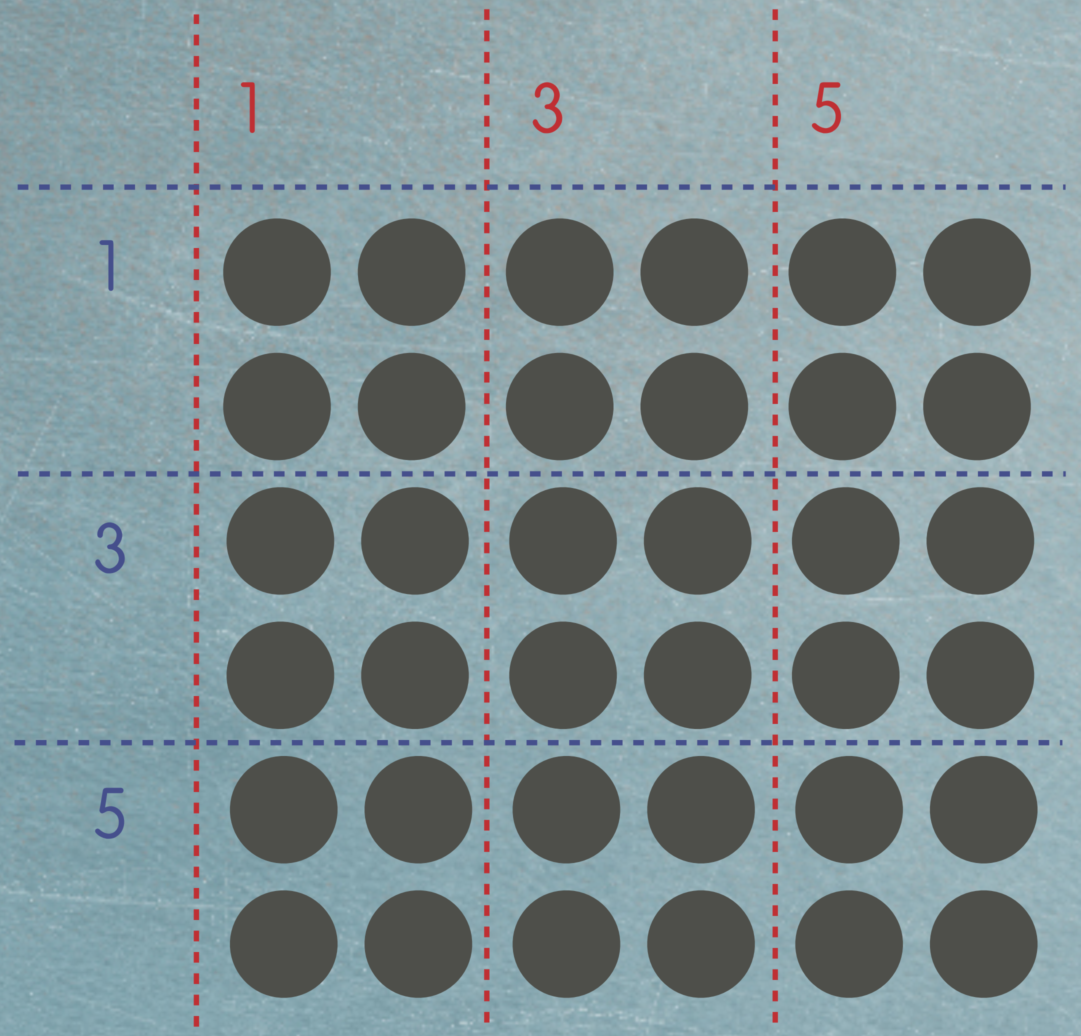
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
T2 = hta(   ,{        ,        },        )
```

# CONSTRUCT HTA FROM 6x6 MATRIX



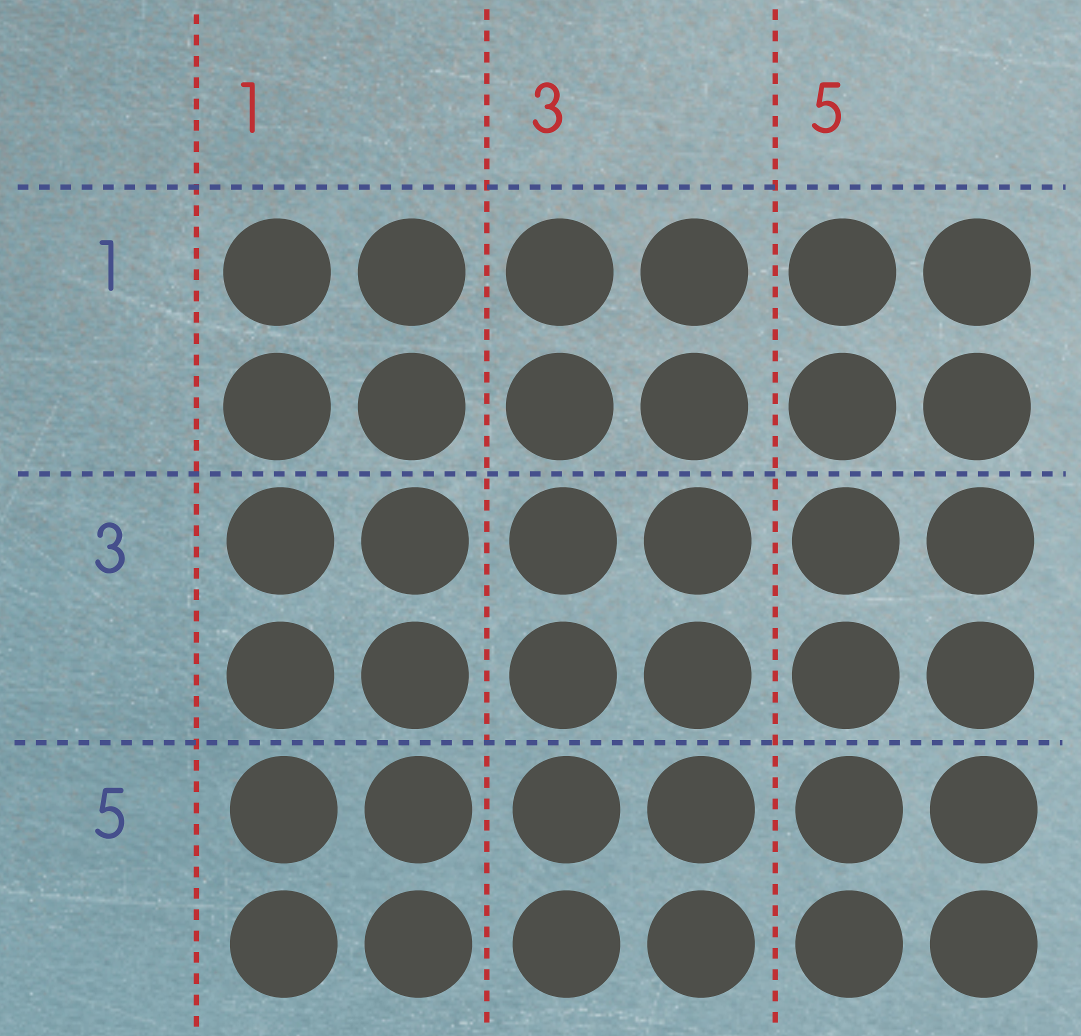
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
T2 = hta(T1,{      ,      },      )
```

# CONSTRUCT HTA FROM 6x6 MATRIX



# CONSTRUCT HTA FROM 6x6 MATRIX

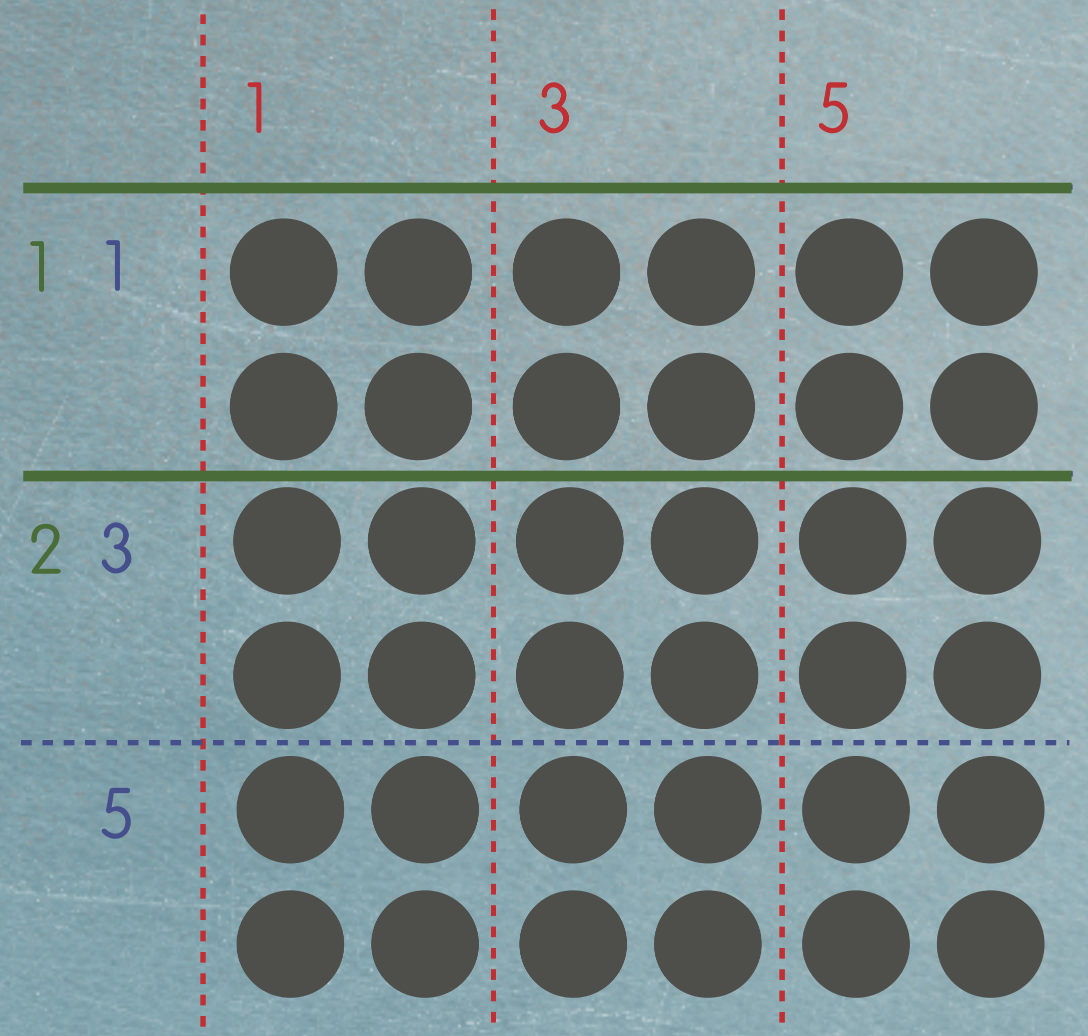
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
T2 = hta(T1,{[1 2],        },       )
```




HOW HTA's WORK 2

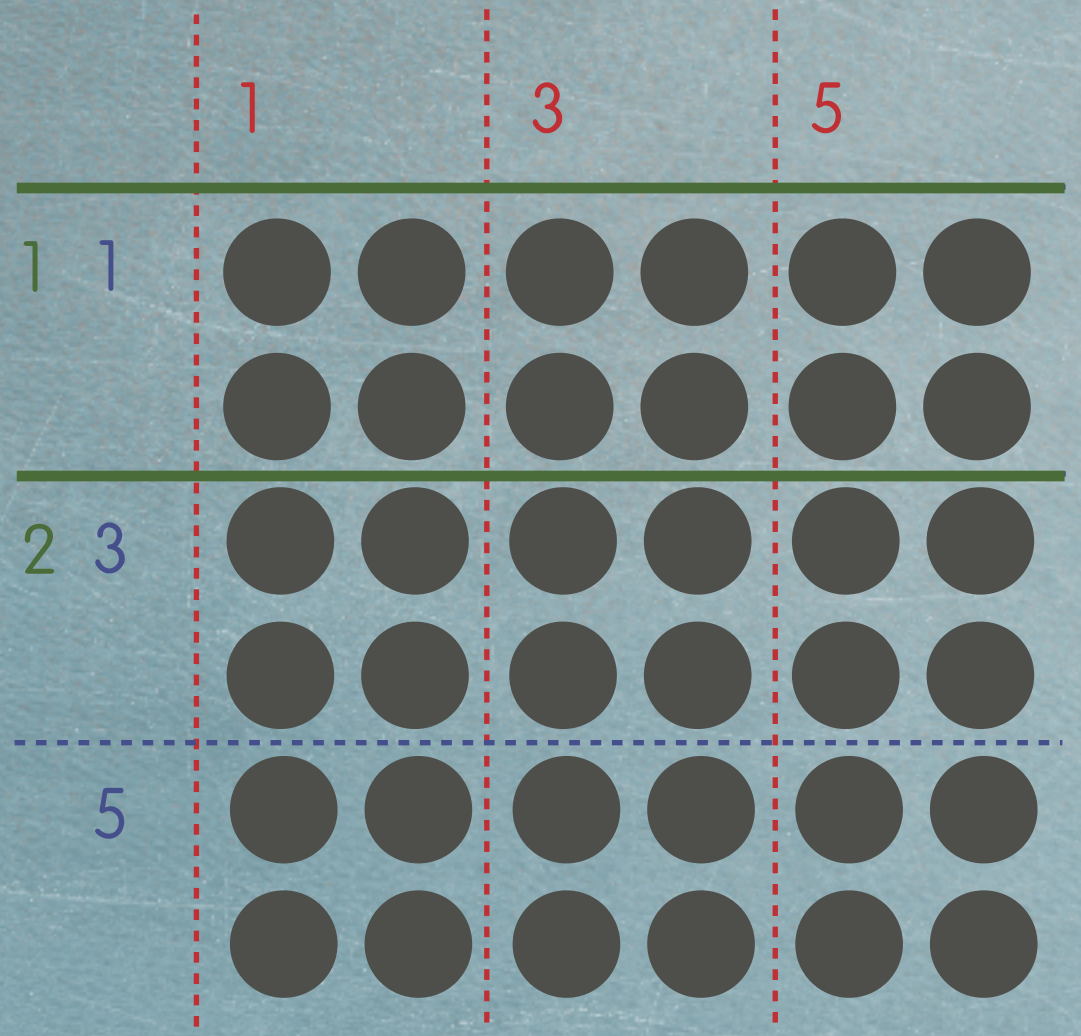
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
T2 = hta(T1,{[1 2],        },       )
```

# CONSTRUCT HTA FROM 6x6 MATRIX



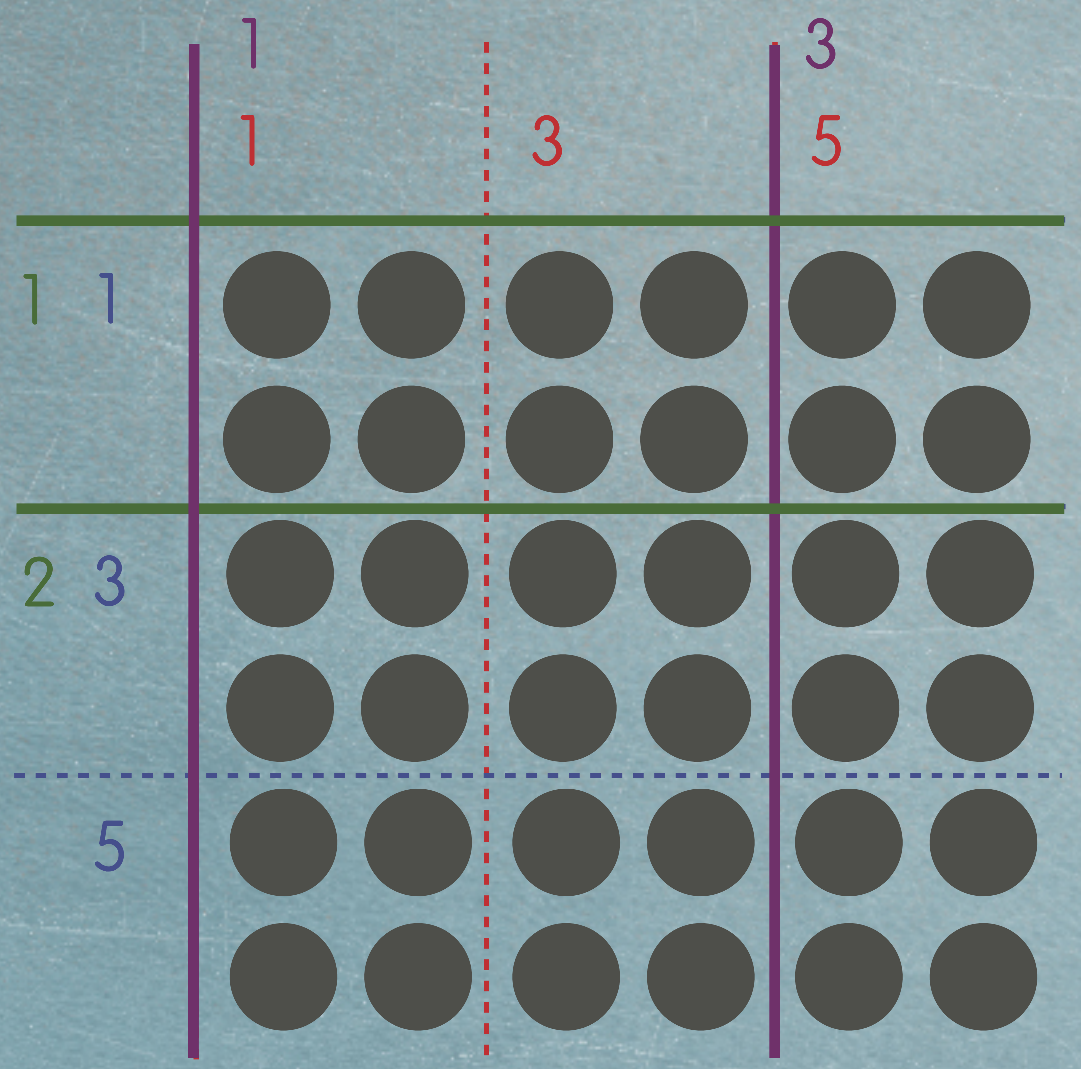
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
T2 = hta(T1,{[1 2],[1 3]},       )
```

# CONSTRUCT HTA FROM 6x6 MATRIX



```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
T2 = hta(T1,{[1 2],[1 3]},       )
```

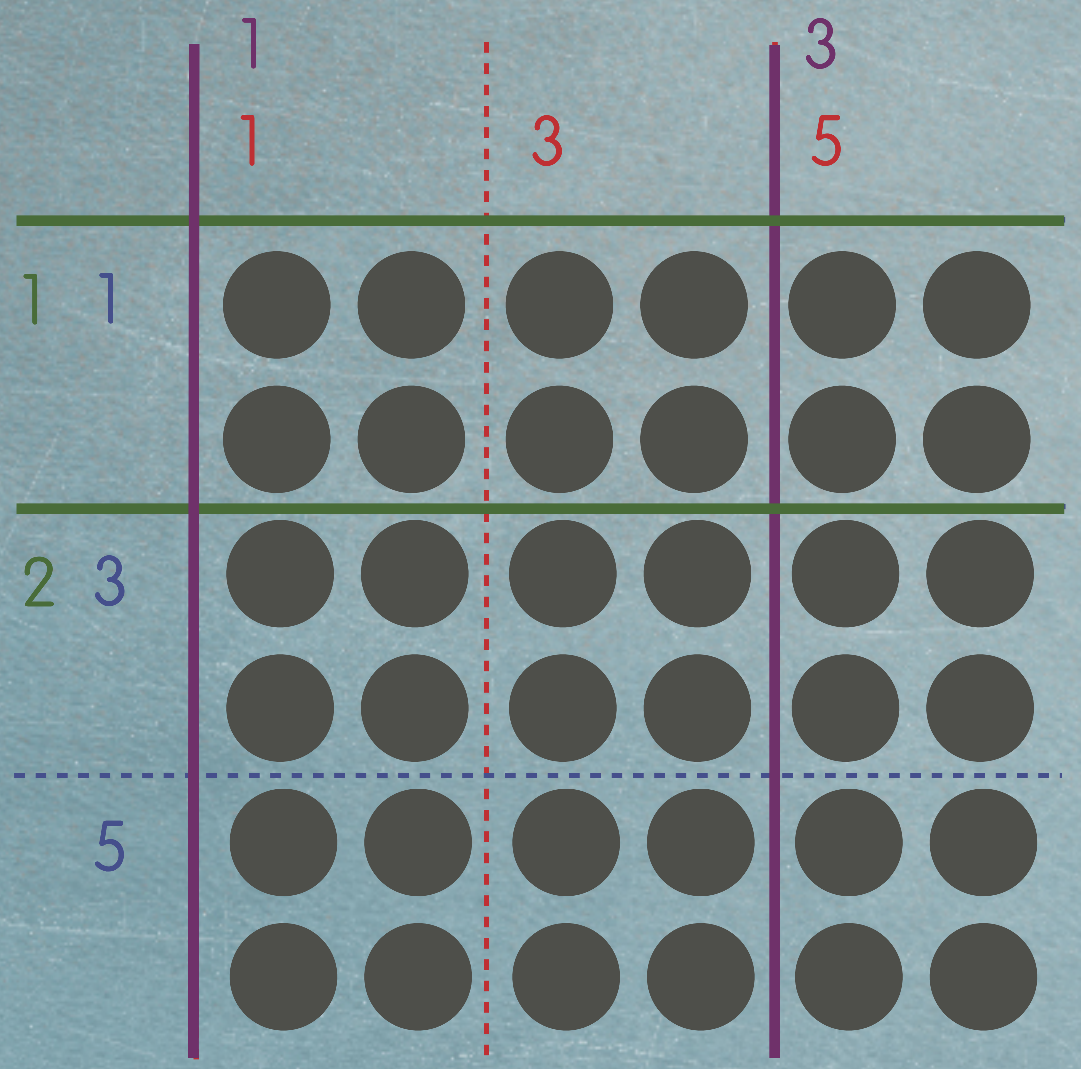
# CONSTRUCT HTA FROM 6x6 MATRIX




HOW HTA's WORK 2

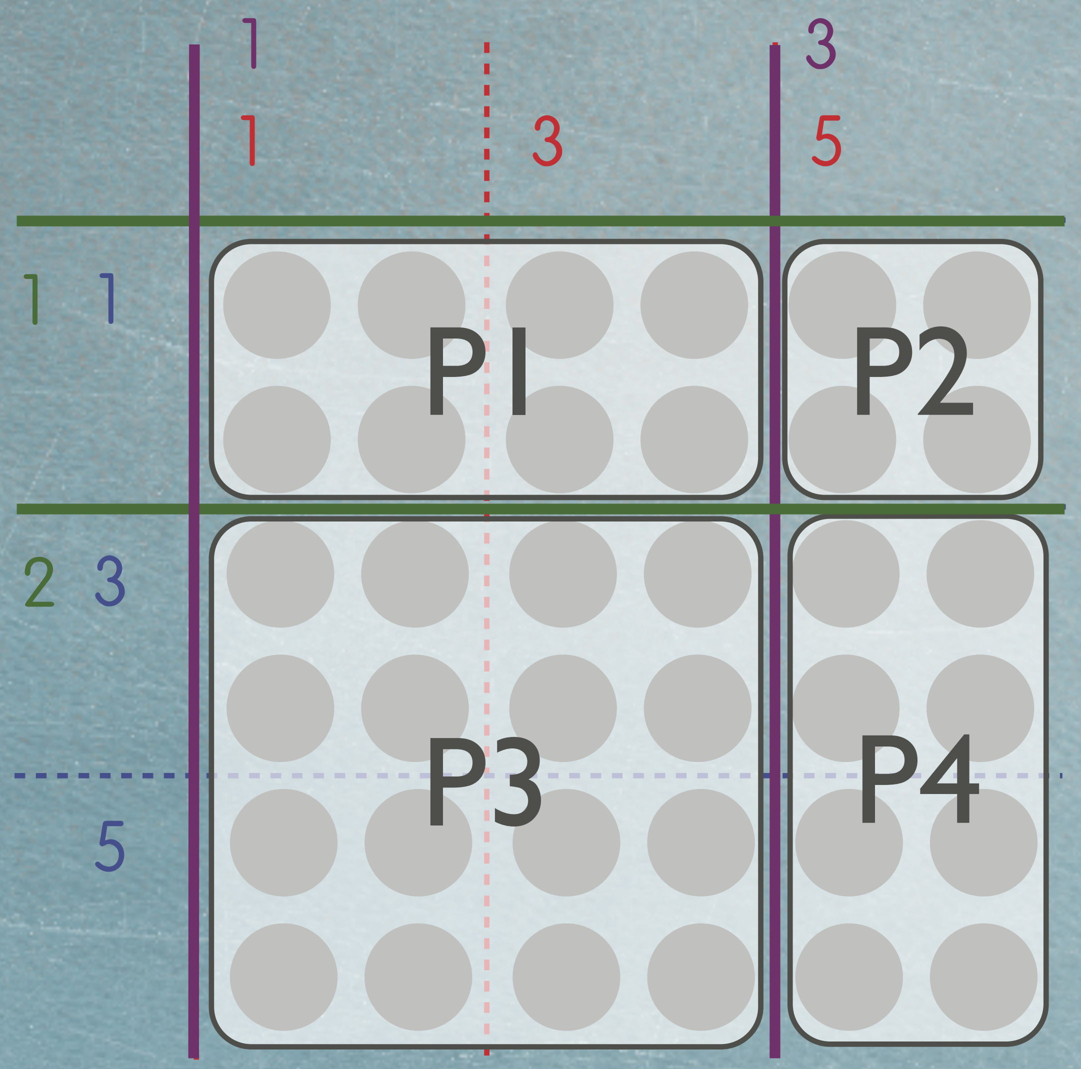
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
T2 = hta(T1,{[1 2],[1 3]},[2 2])
```

# CONSTRUCT HTA FROM 6x6 MATRIX



# CONSTRUCT HTA FROM 6x6 MATRIX

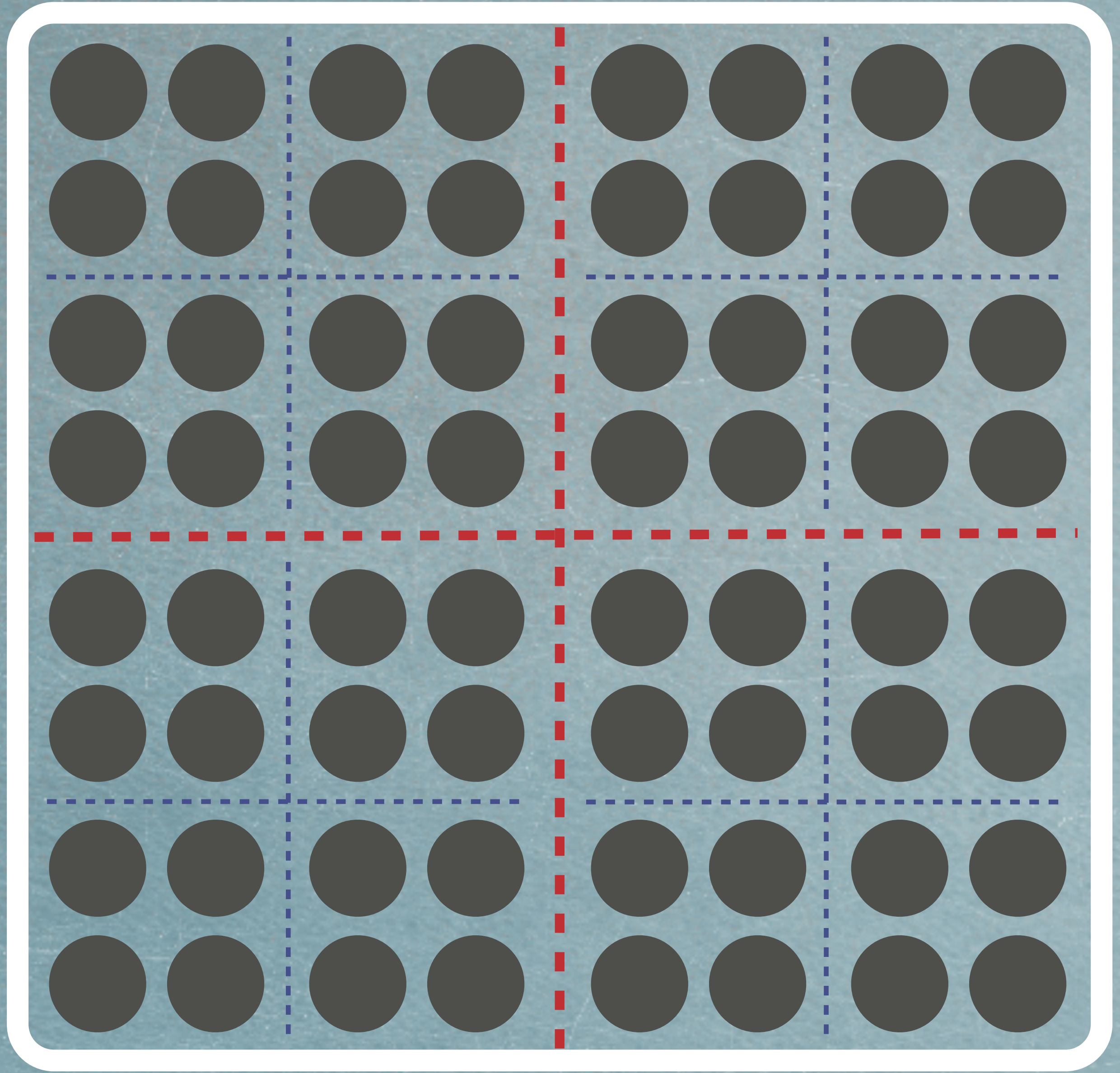
```
T1 = hta( M ,{[1 3 5],[1 3 5]} )
T2 = hta(T1,{[1 2],[1 3]},[2 2])
```




HOW HTA's WORK 2

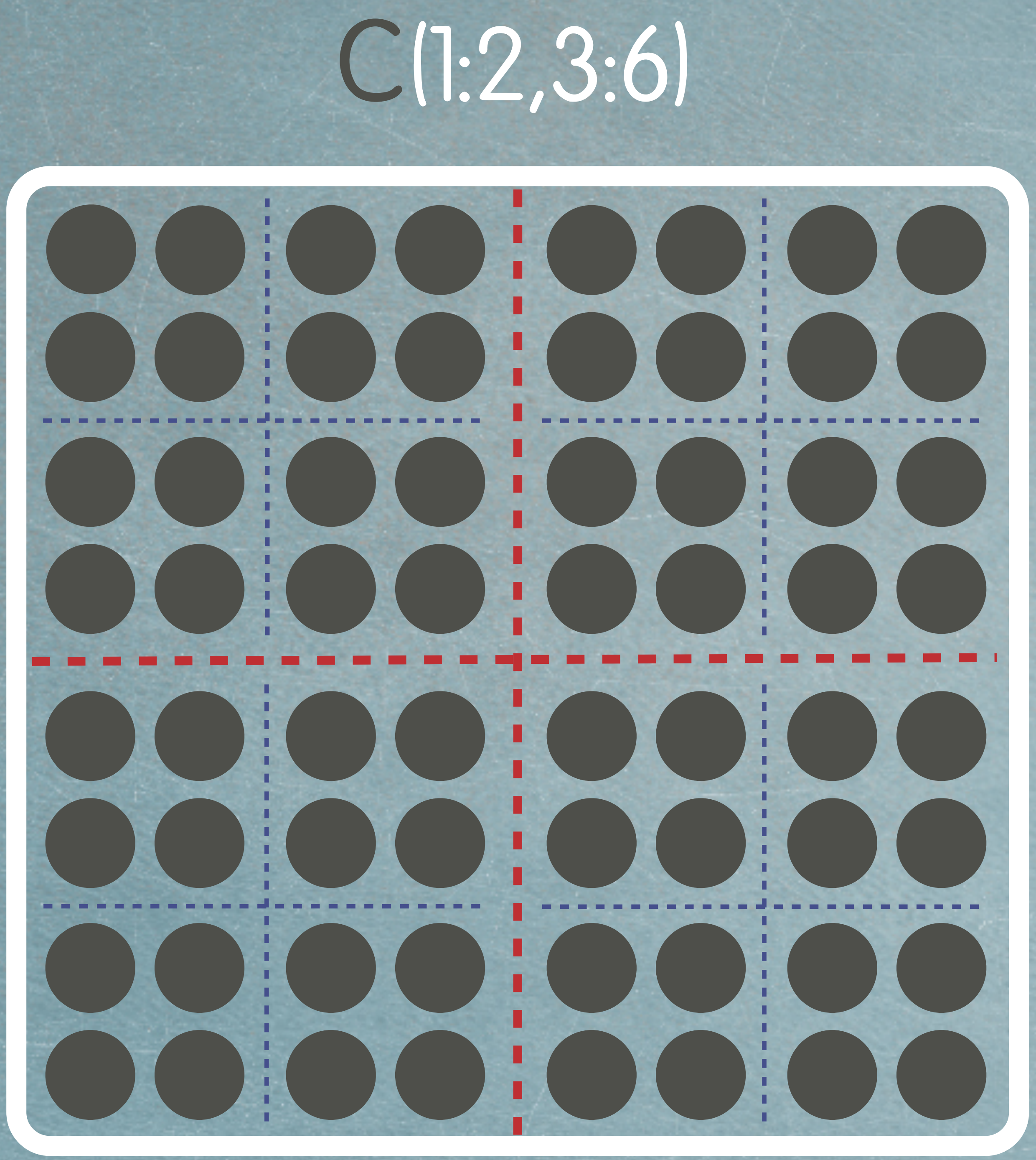
HTA
ACCESS

C =

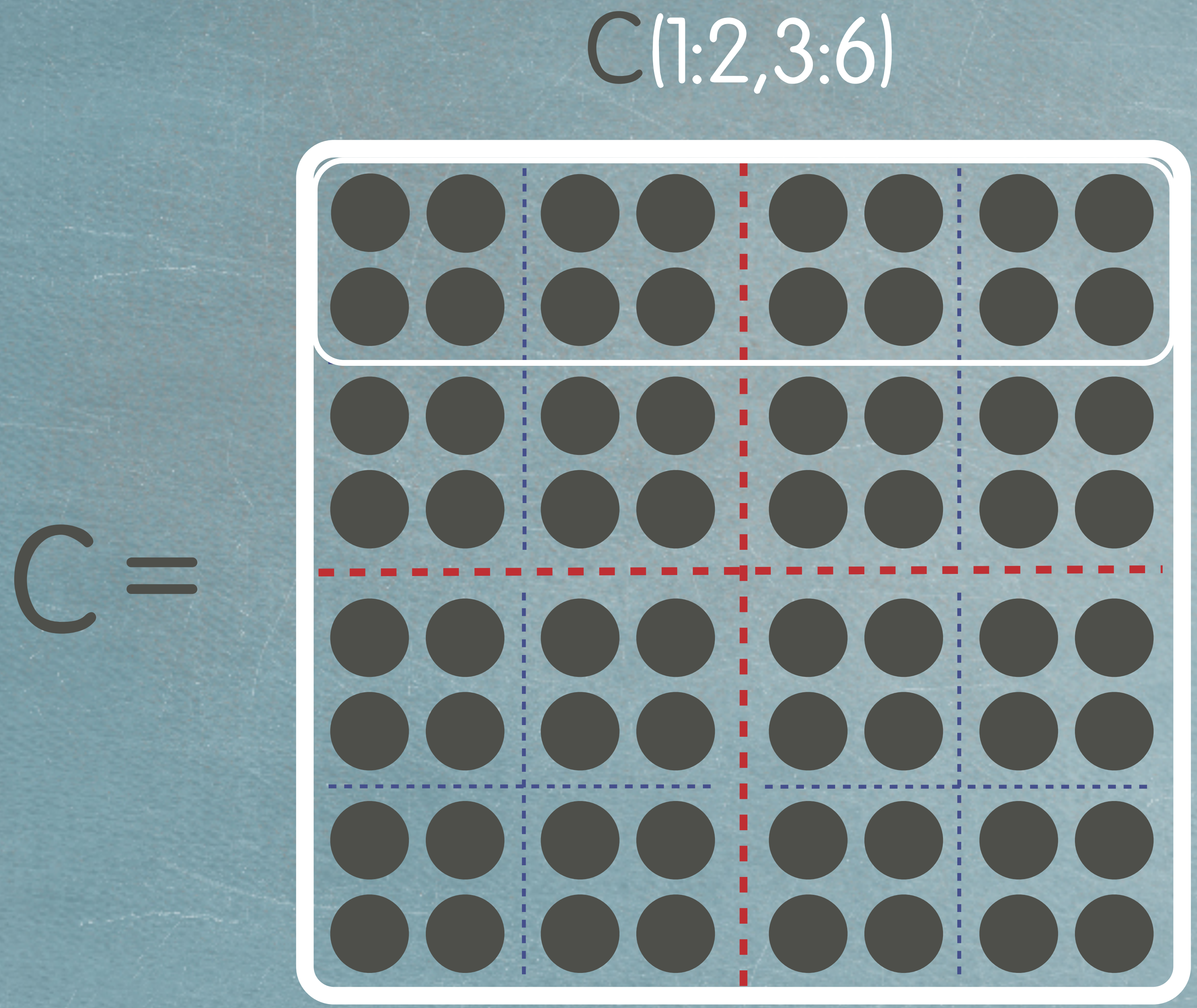


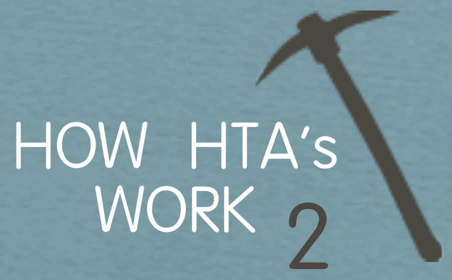
# HTA ACCESS

C =



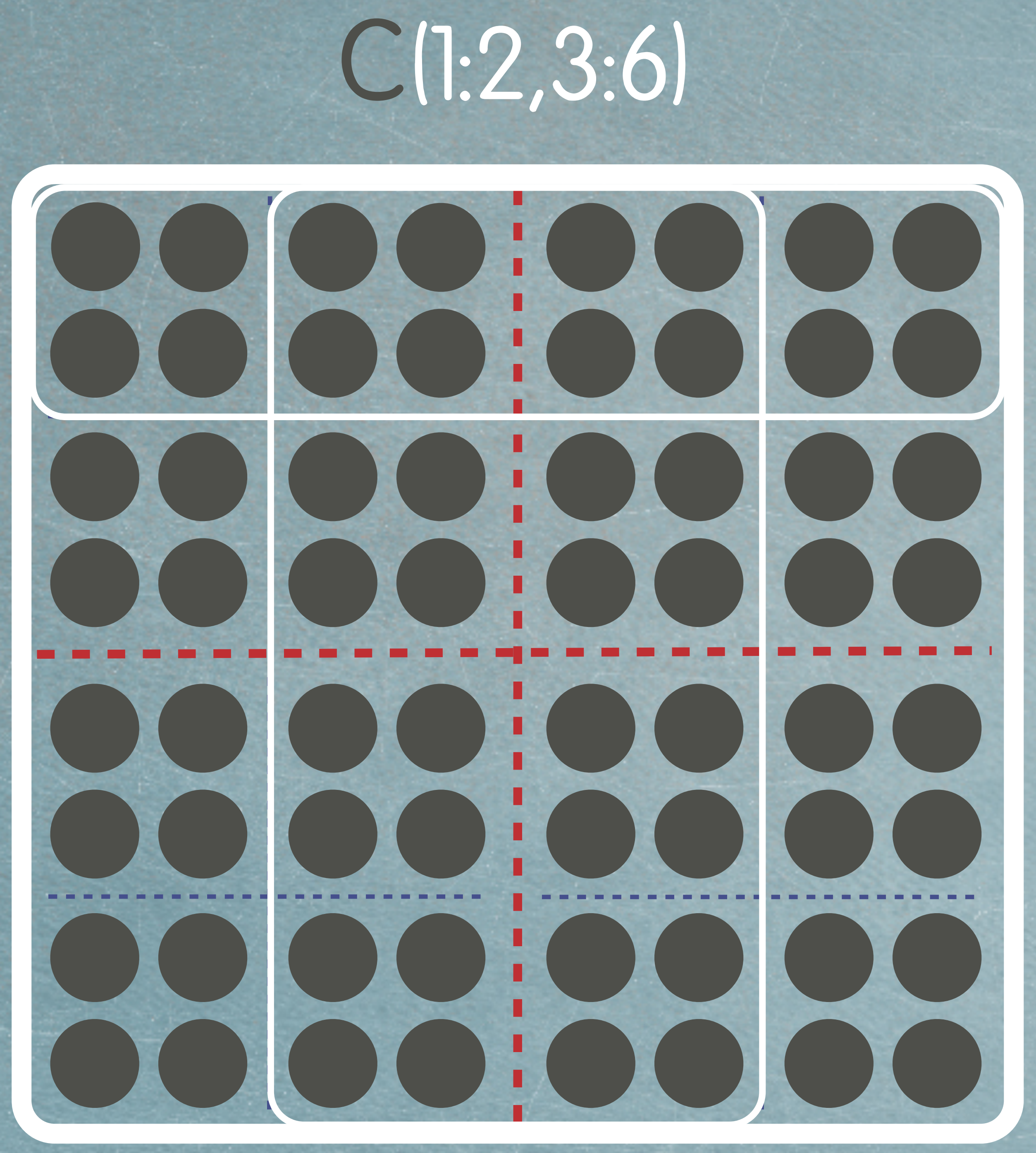
# HTA ACCESS



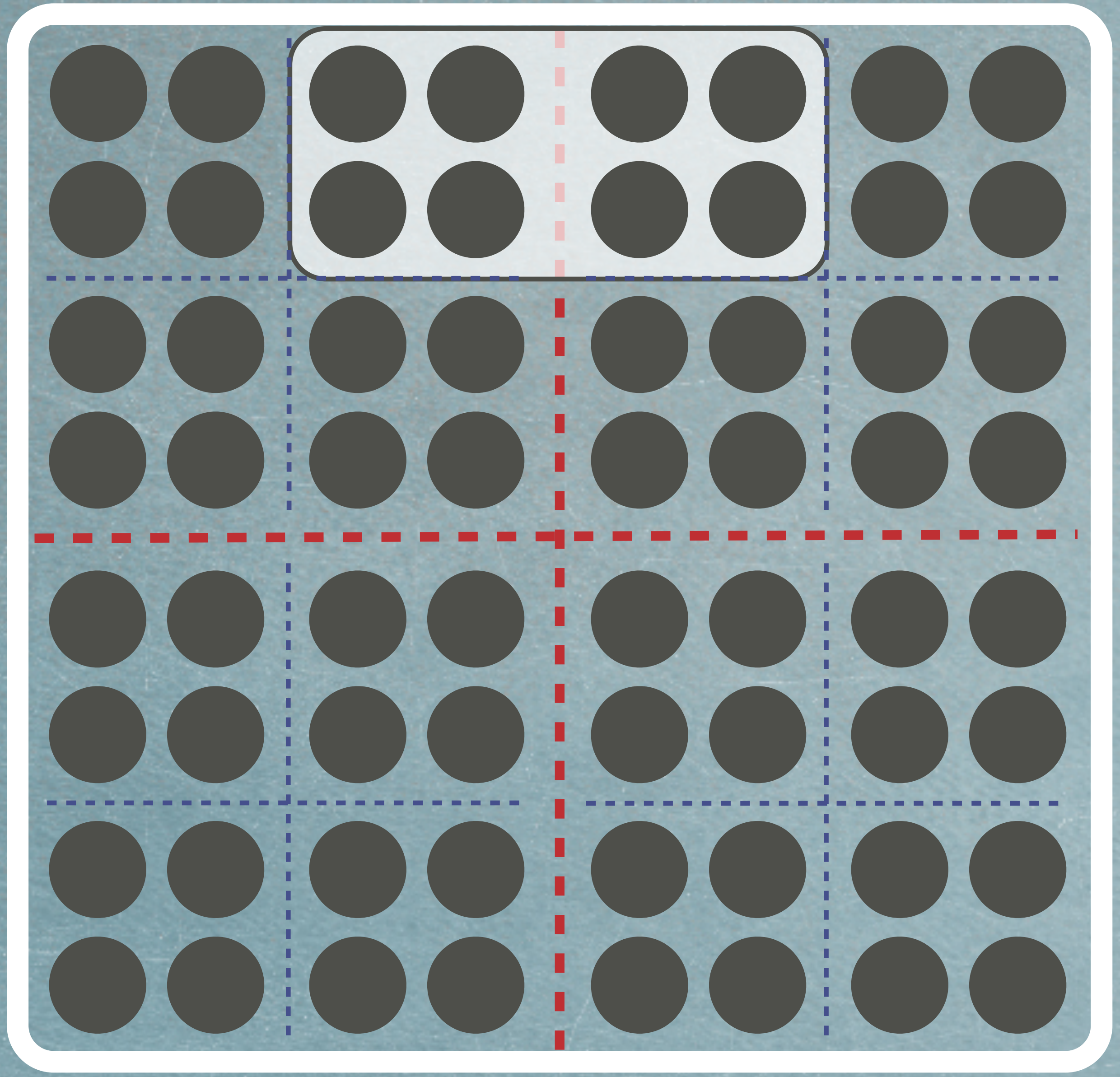
# HTA ACCESS

C =



# HTA ACCESS

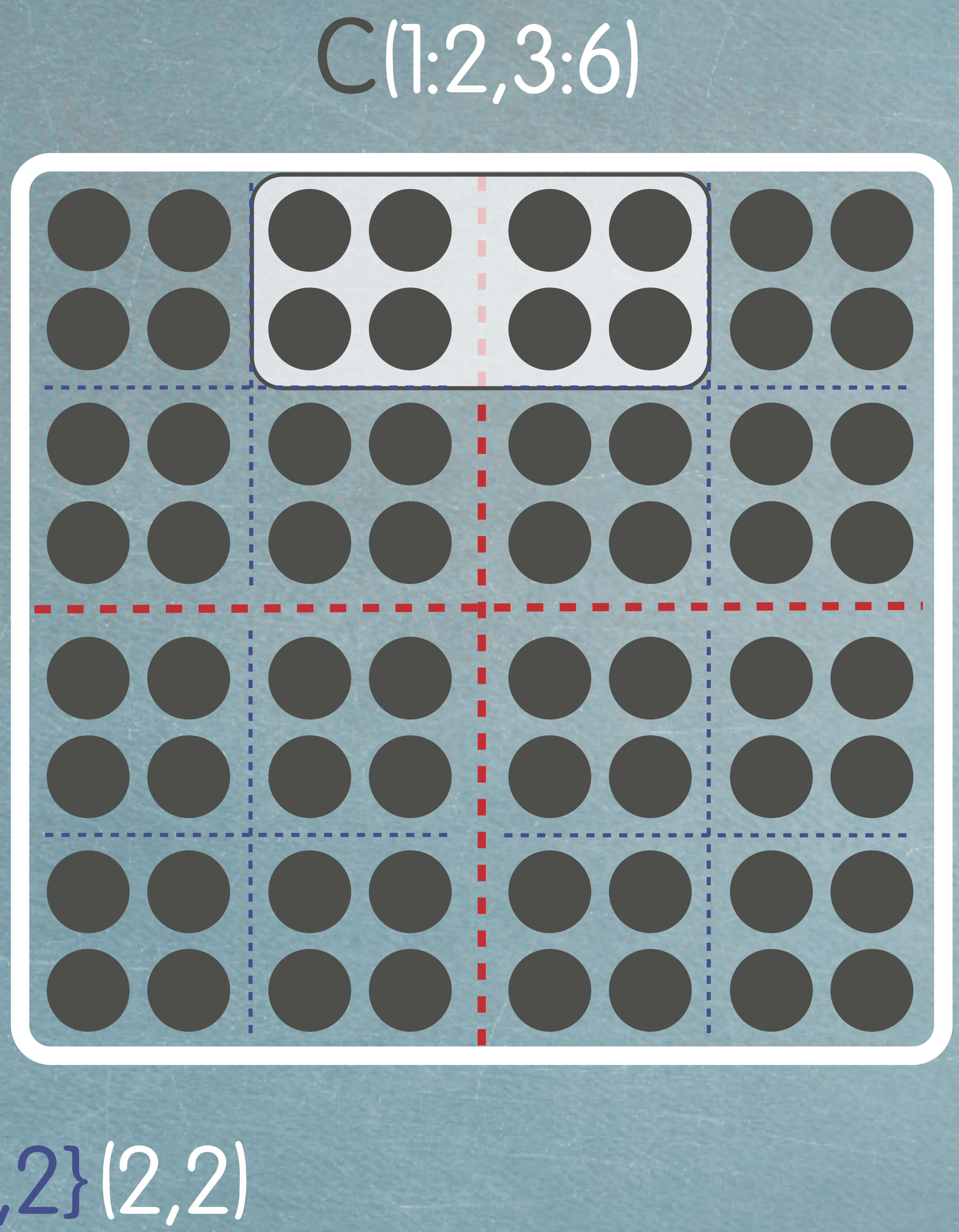
C =




HOW HTA's WORK 2

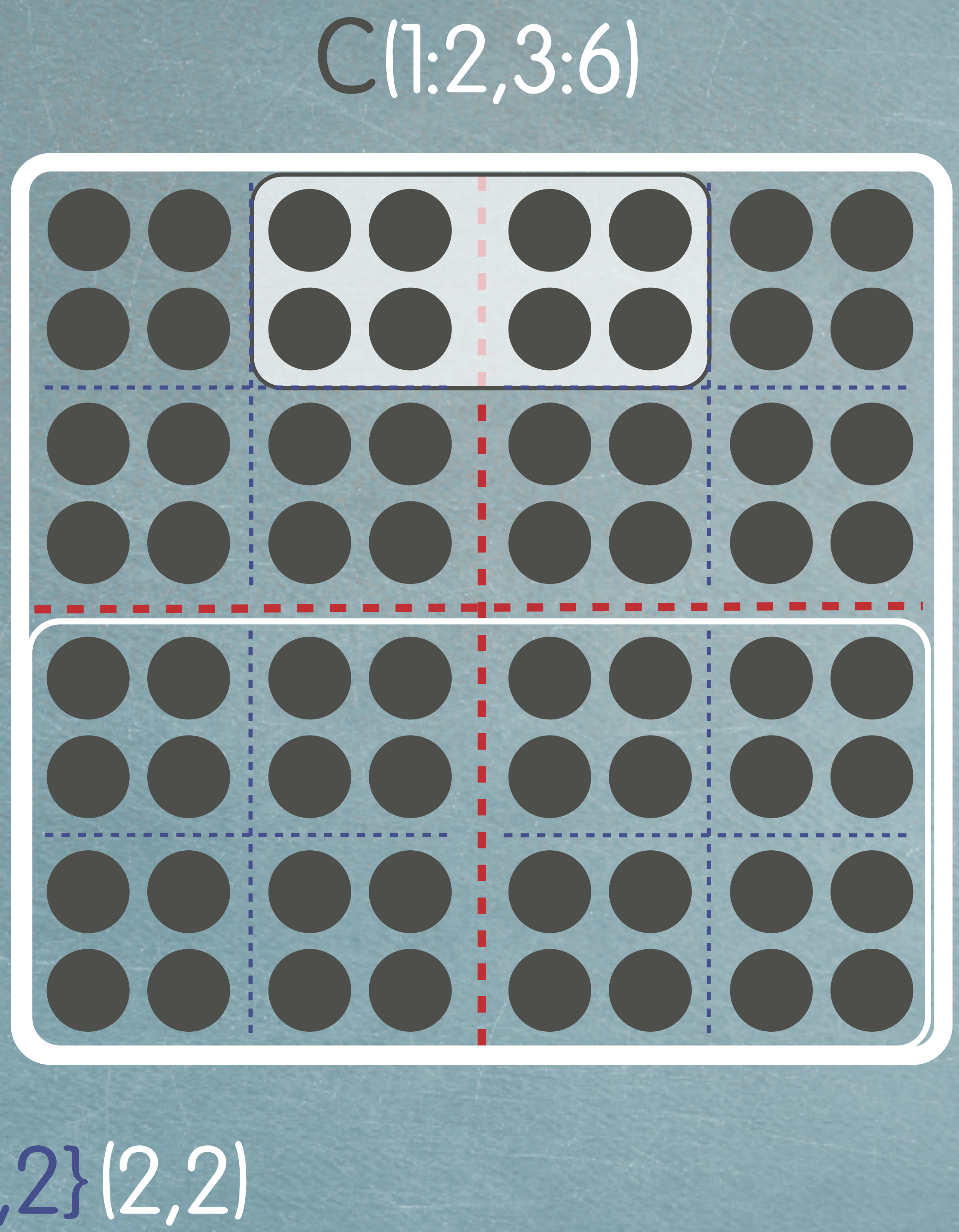
# HTA ACCESS

C =



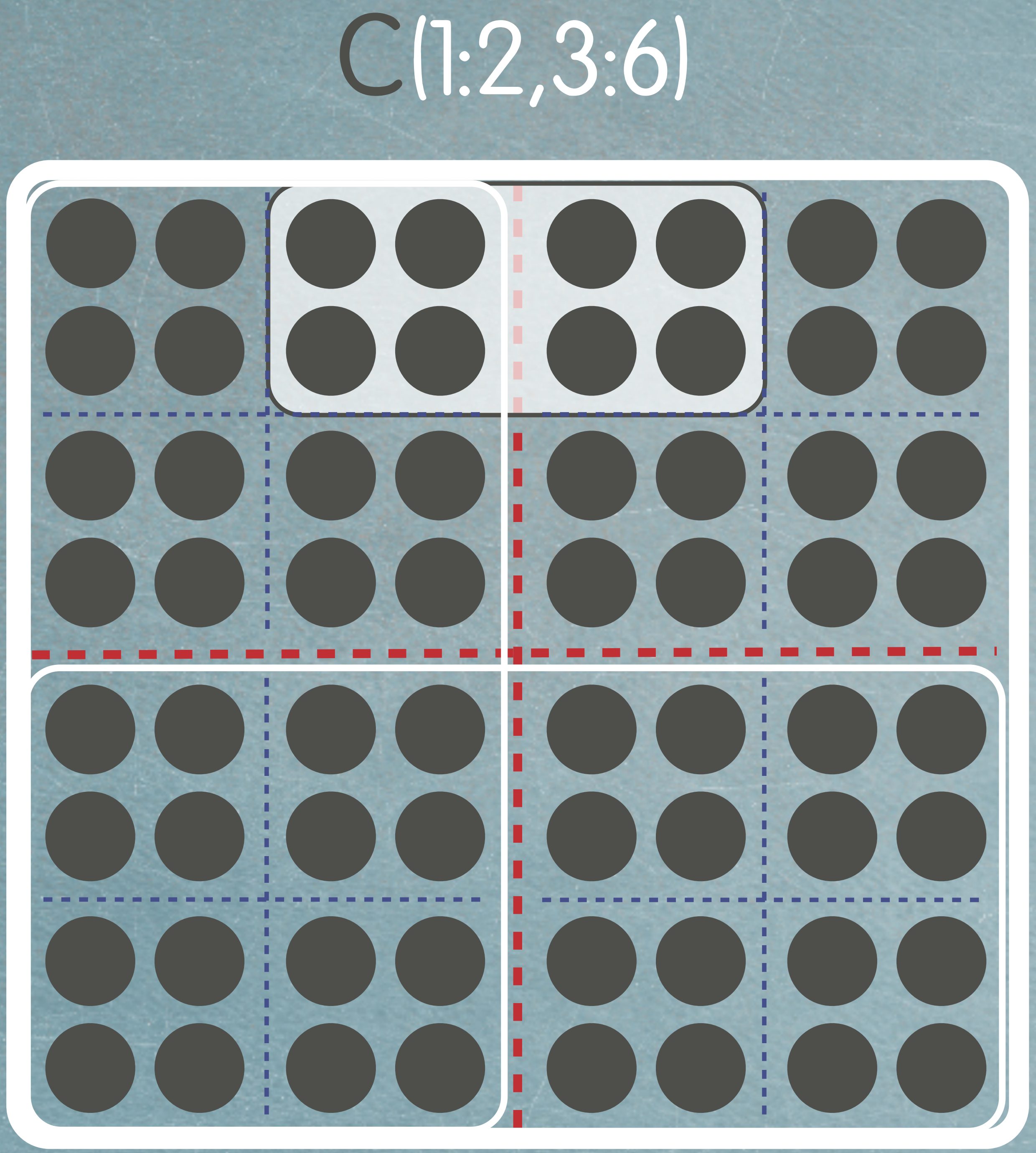
# HTA ACCESS

C =



# HTA ACCESS

C =

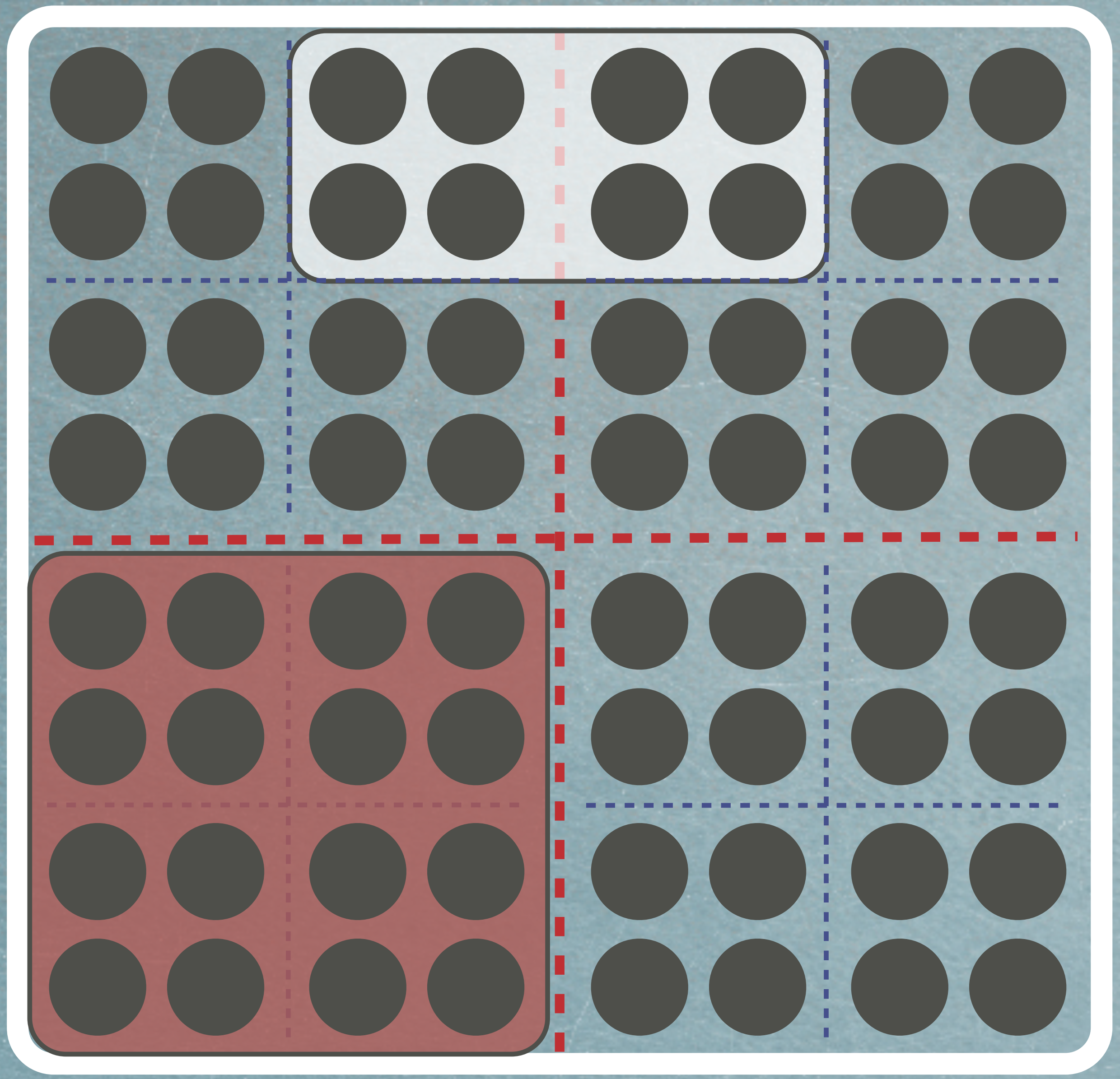



C{2,1}{1,2}(2,2)

# HTA ACCESS

C =

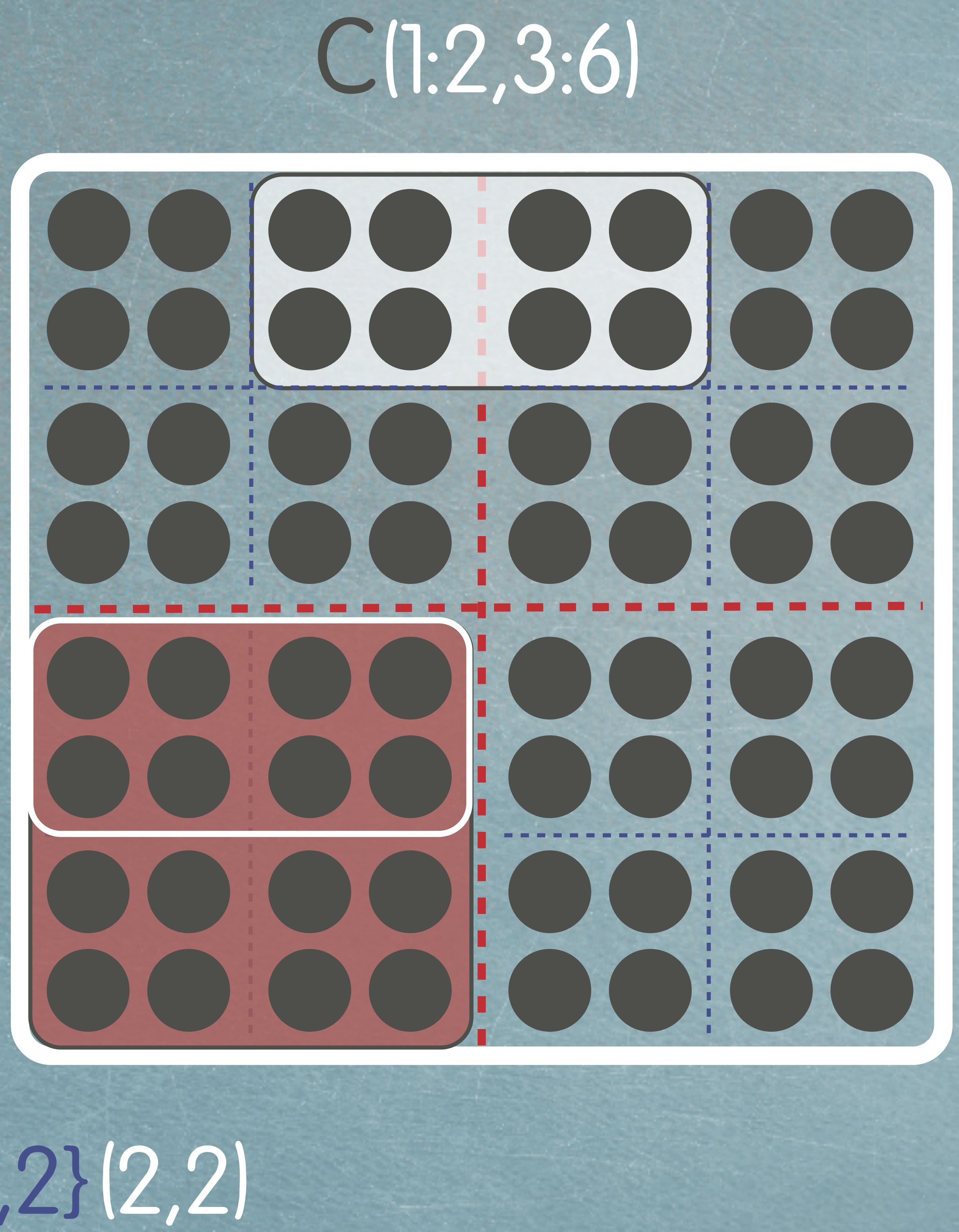
C(1:2,3:6)



C{2,1}{1,2}(2,2)

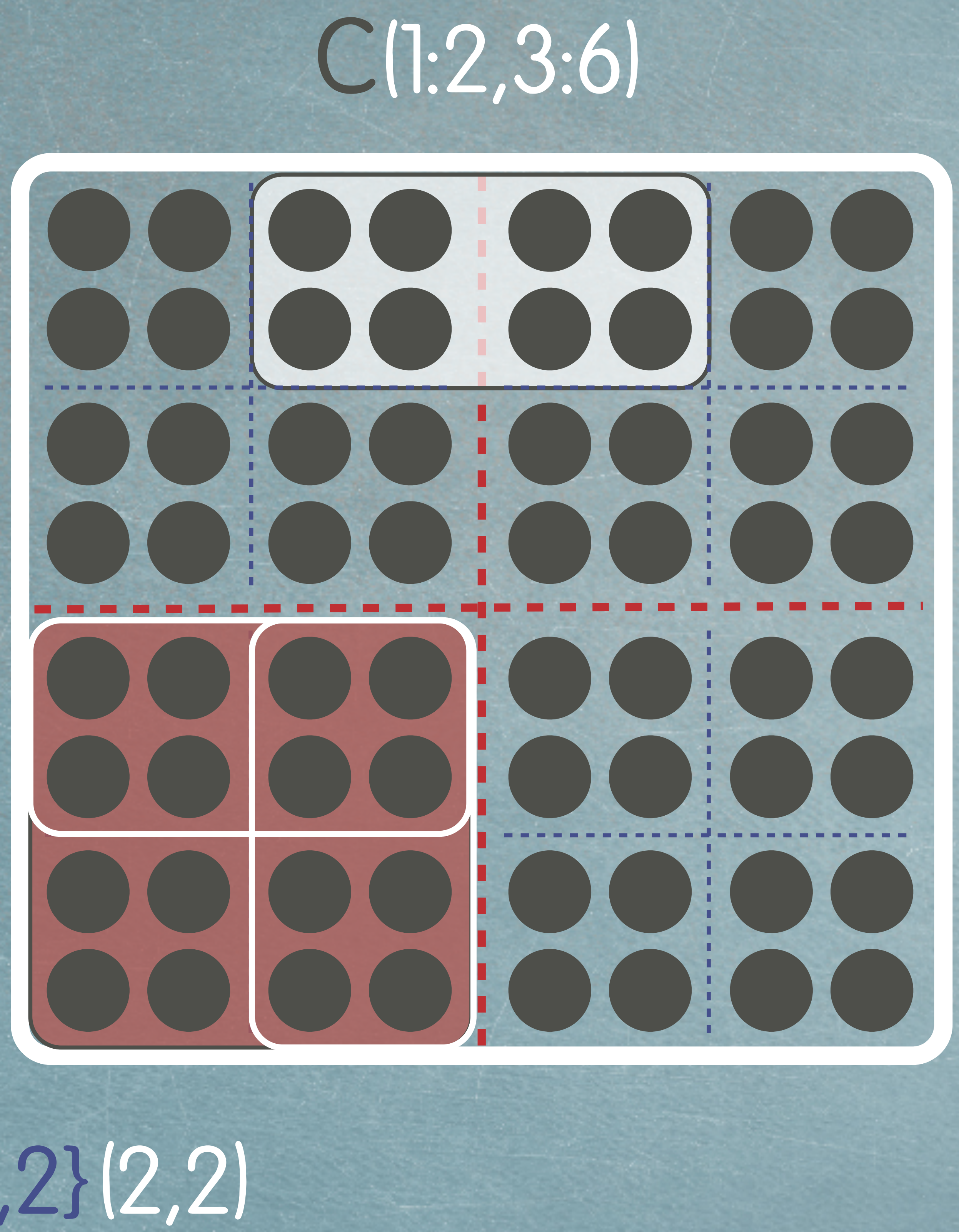
# HTA ACCESS

C =

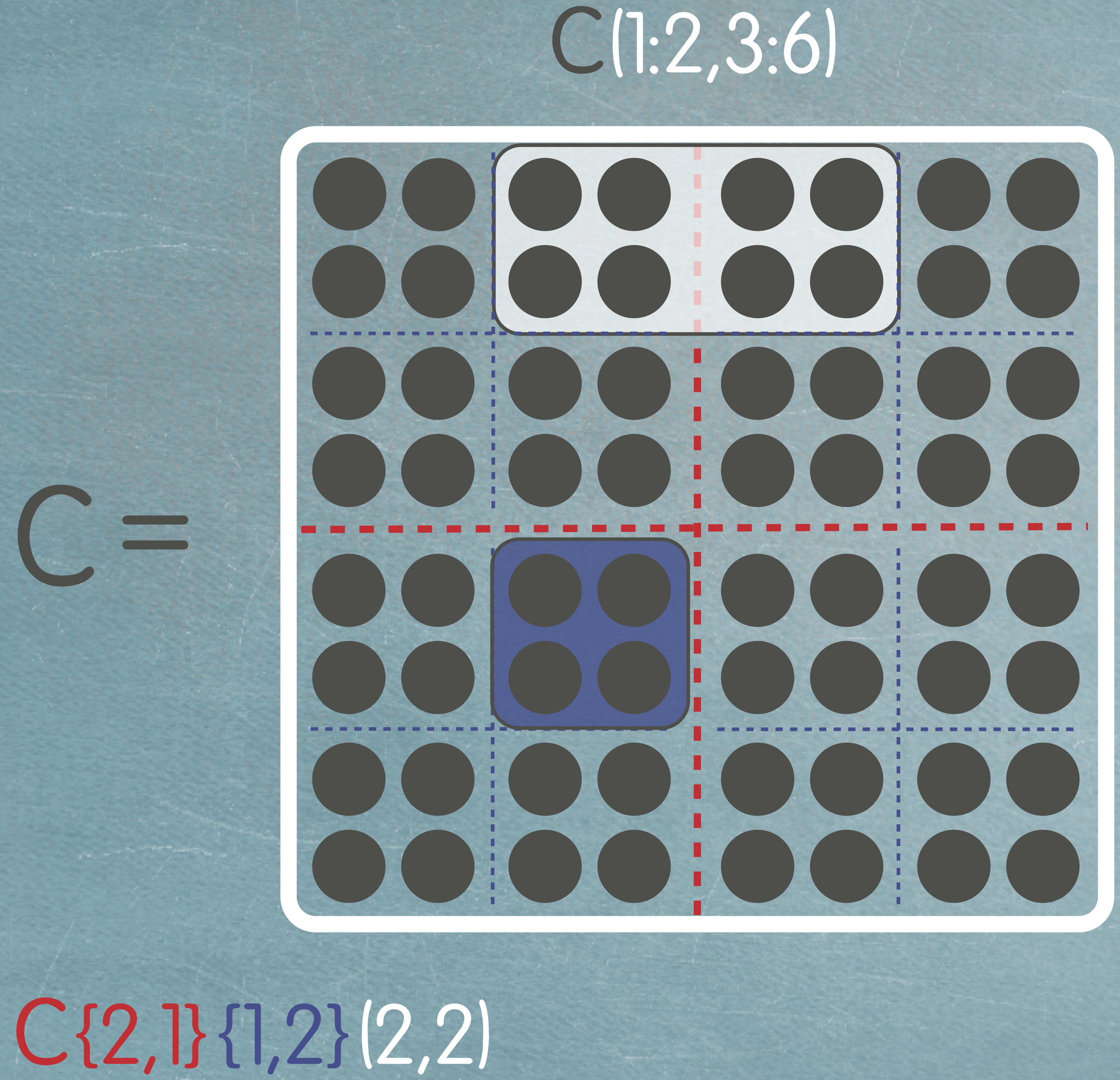


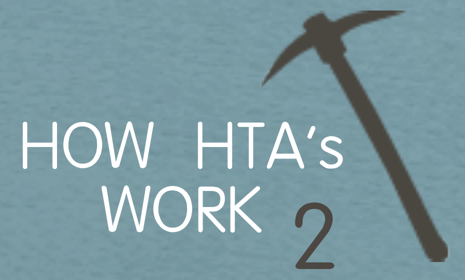
# HTA ACCESS

C =

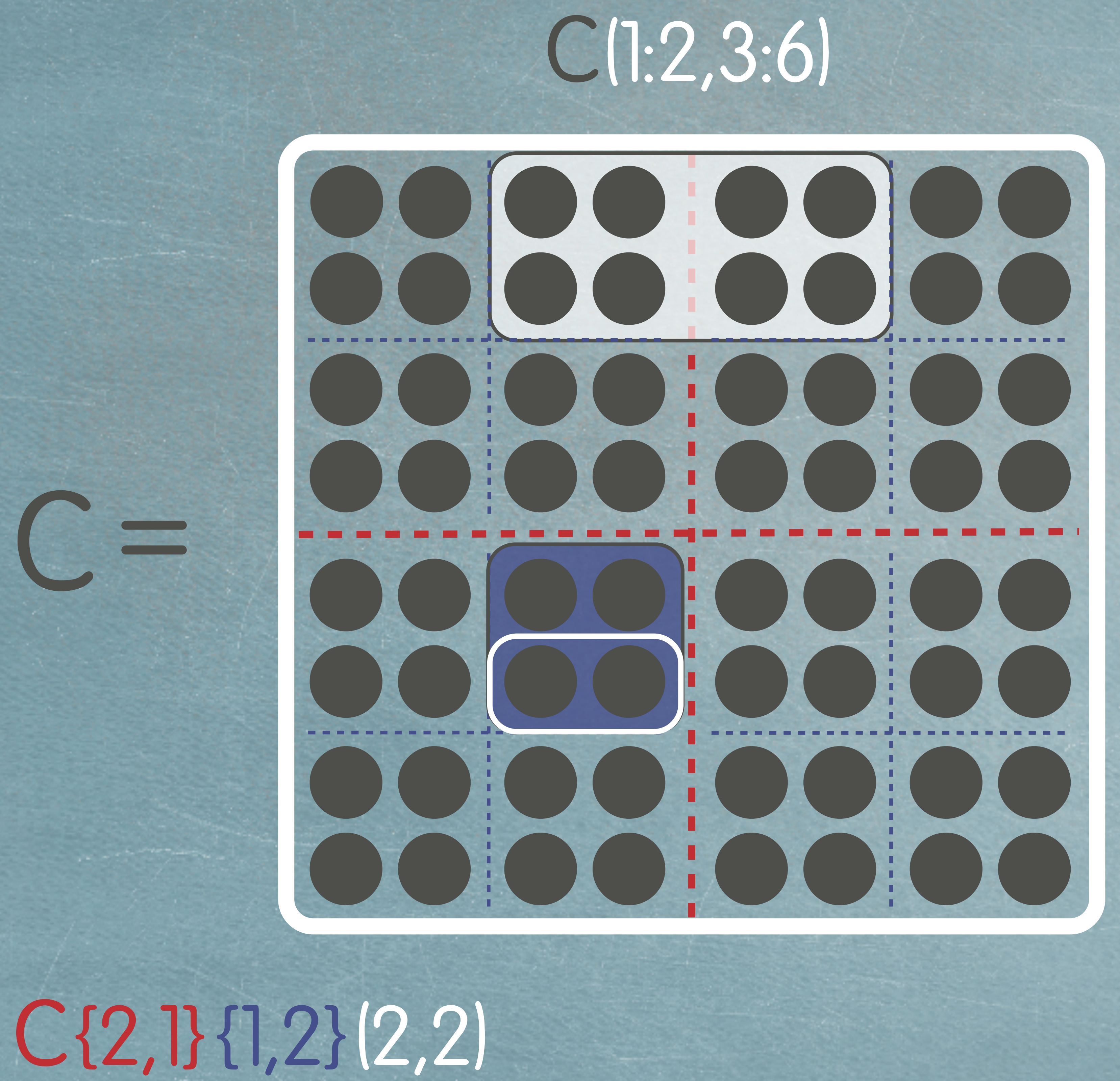


# HTA ACCESS



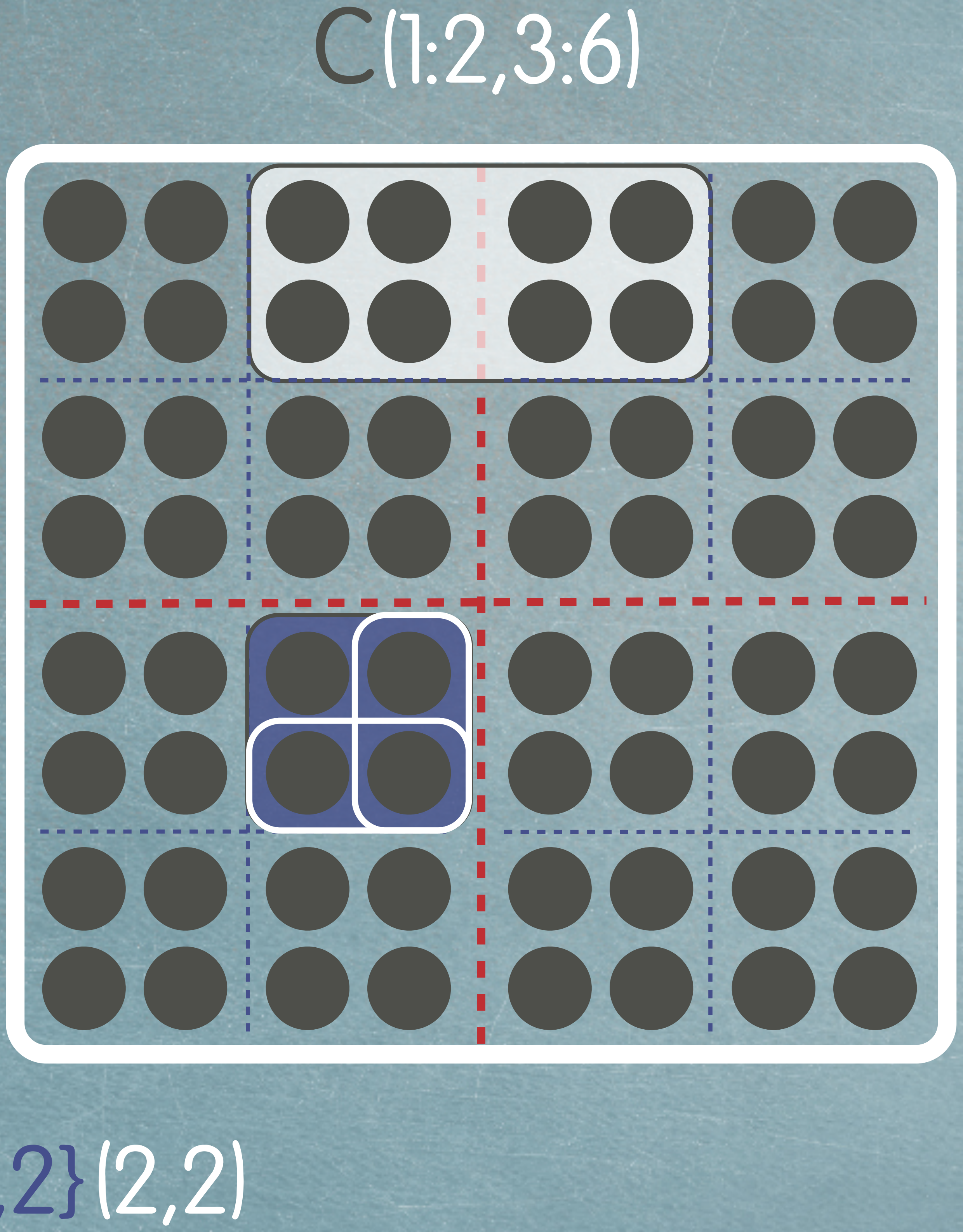



# HTA ACCESS



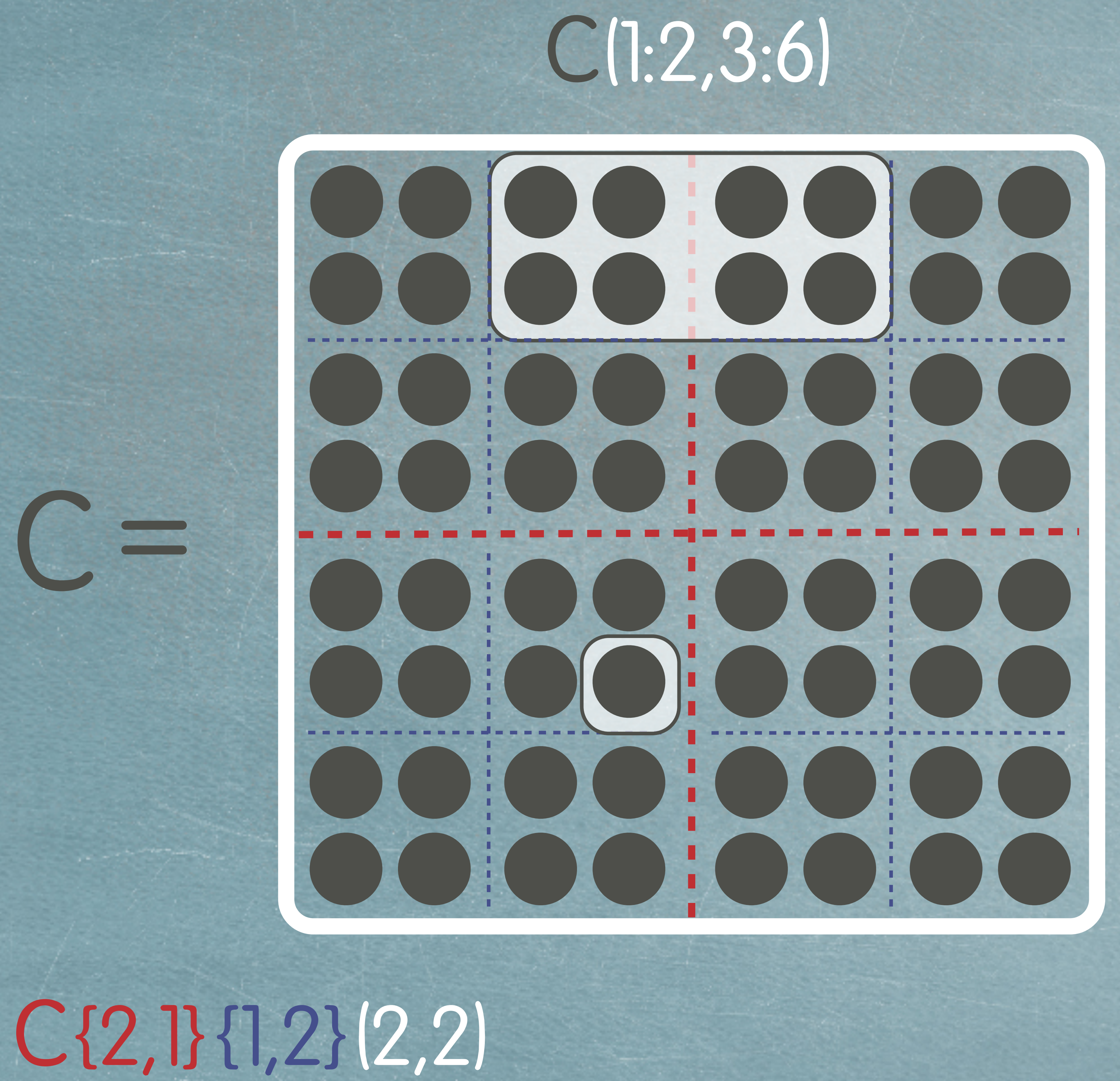
# HTA ACCESS

C =



# HTA ACCESS

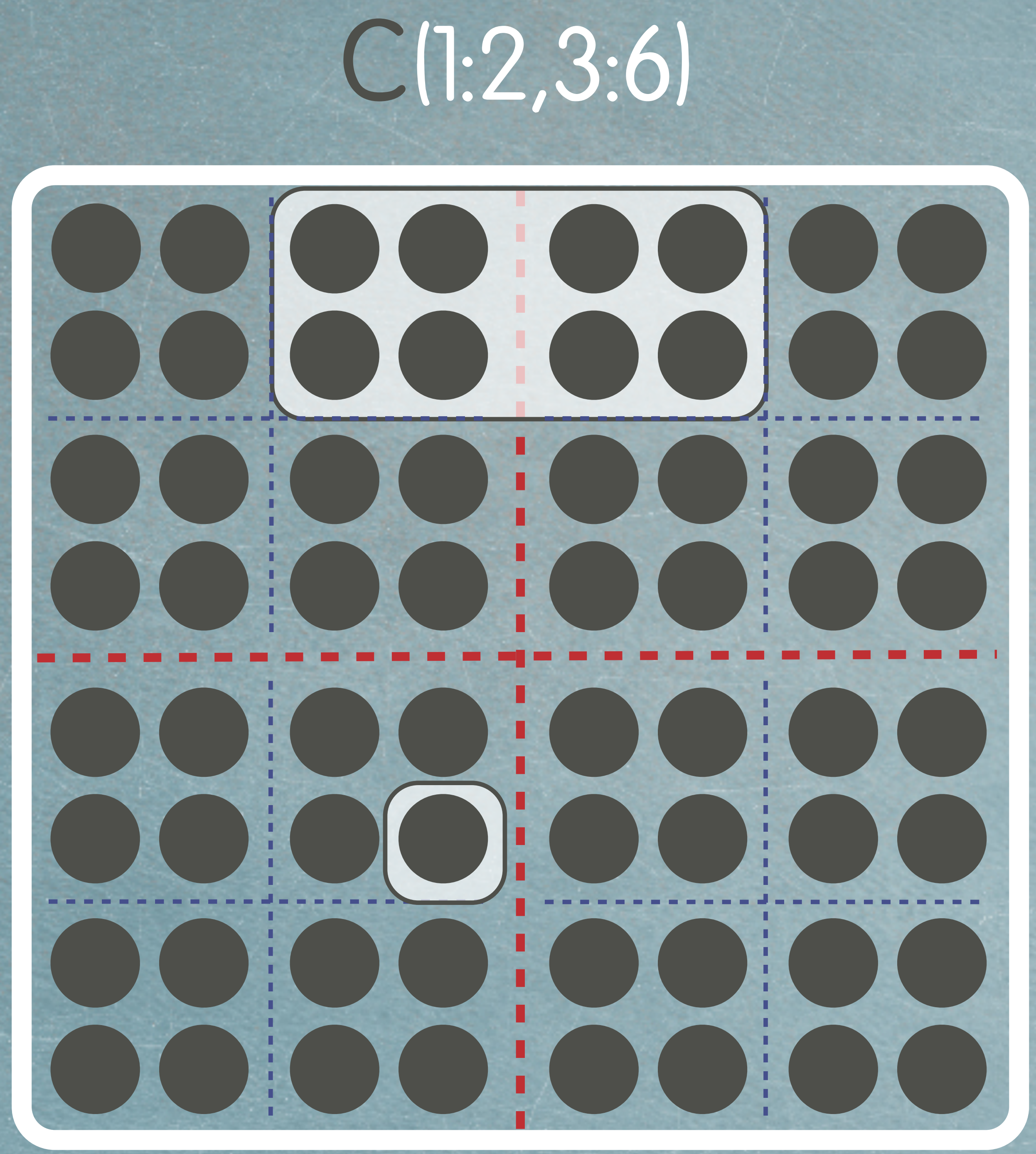
C =




HOW HTA's WORK 2

# HTA ACCESS

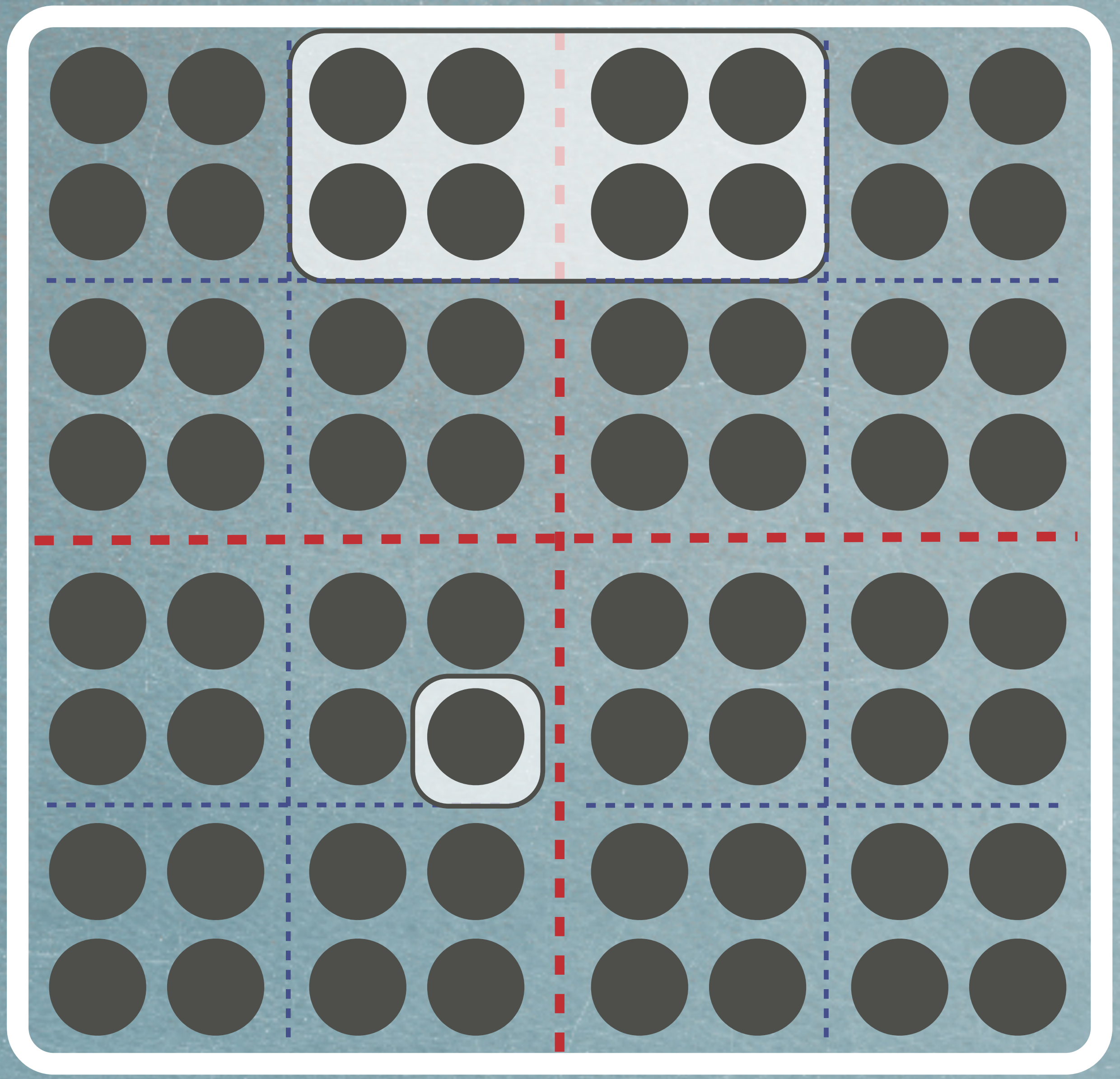
C =




C{2,1}{1,2}(2,2) = C(6,4)

# HTA ACCESS

C =

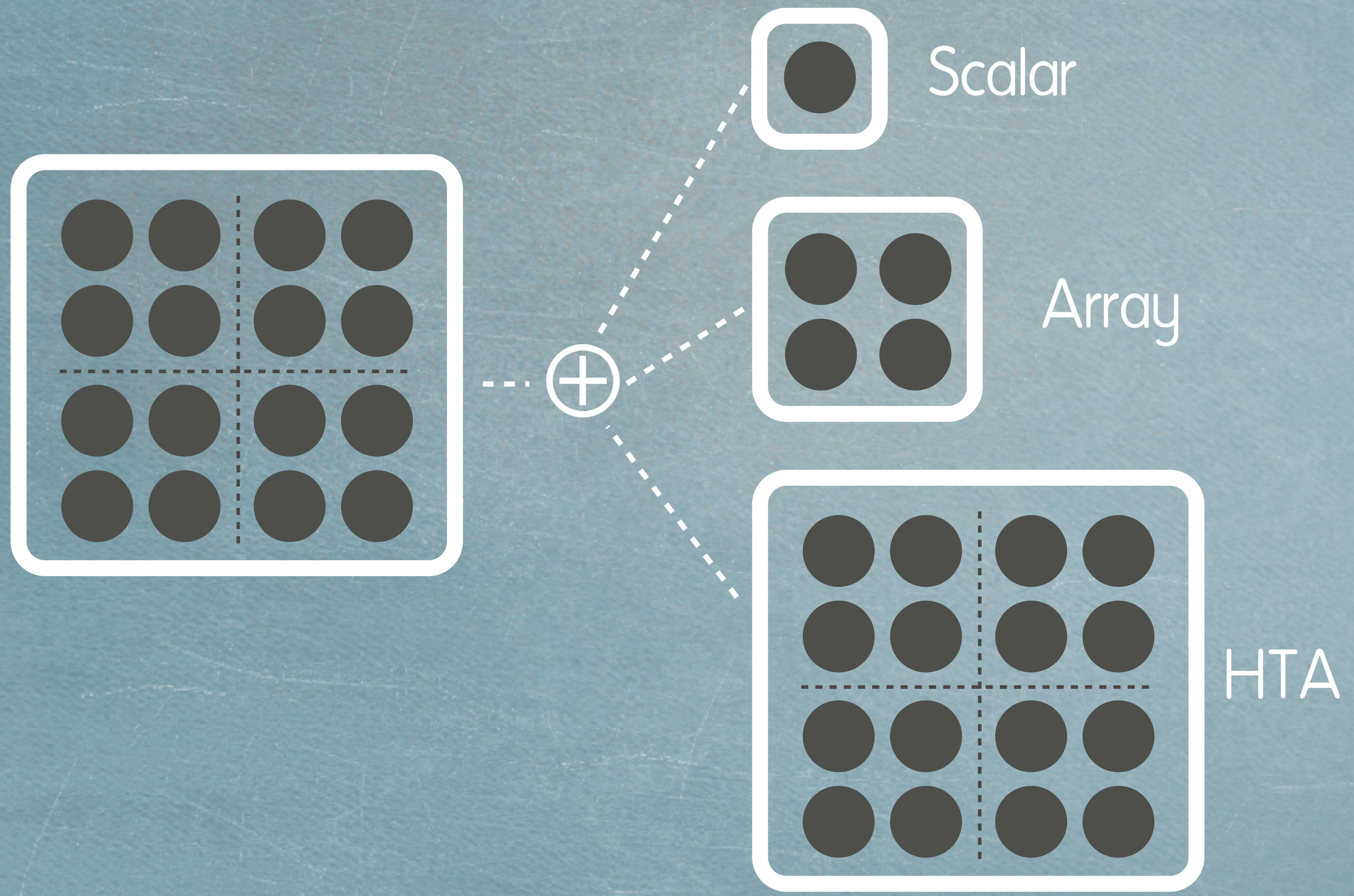



C{2,1}{1,2}(2,2) = C(6,4) = C{2,1}(2,4)

HOW HTA's WORK 2

# VALID OPERATIONS

# ASSIGNMENTS & BINARY OPERATORS

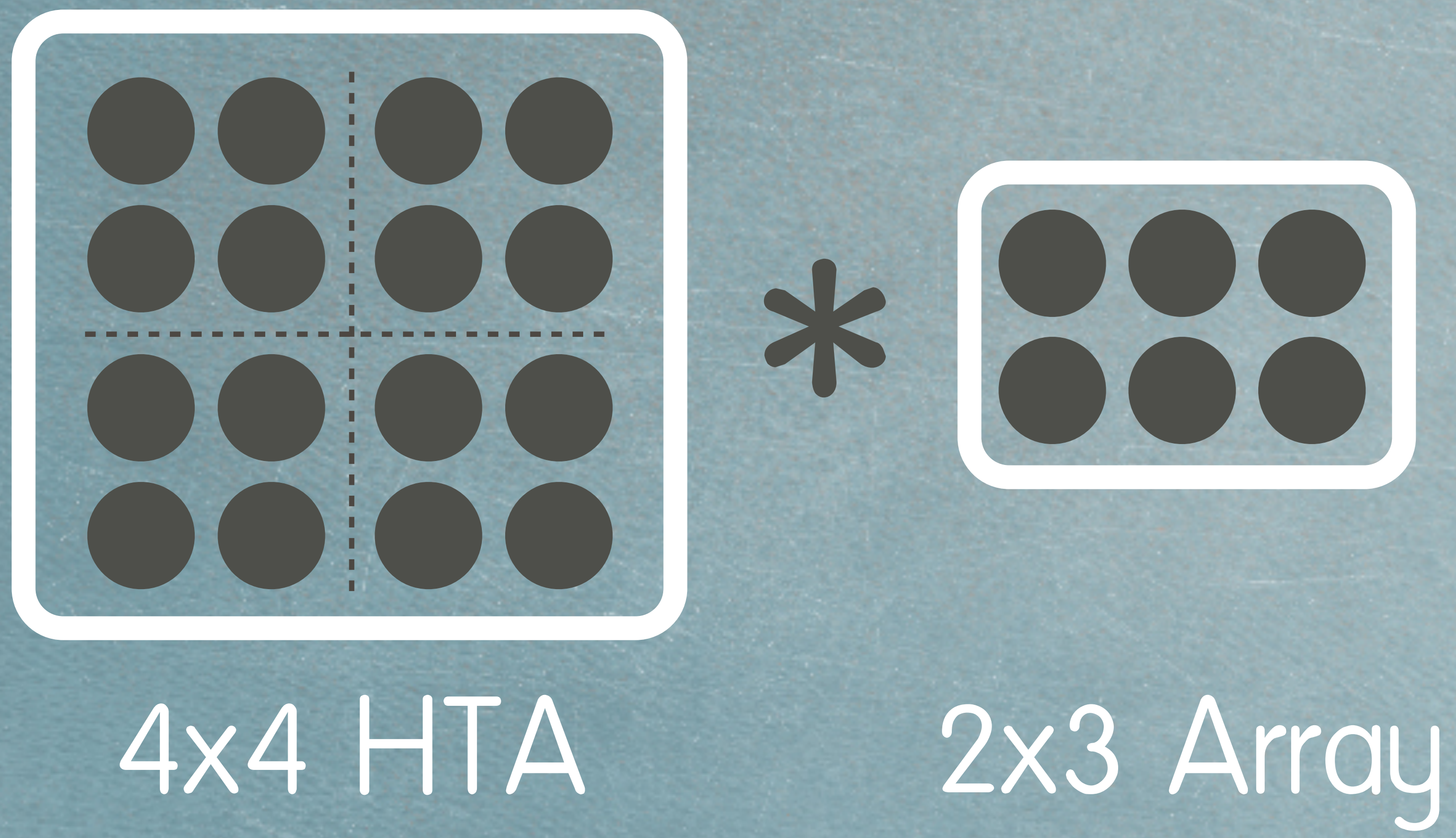


# VALID OPERATION ?

# ASSIGNMENTS & BINARY OPERATORS

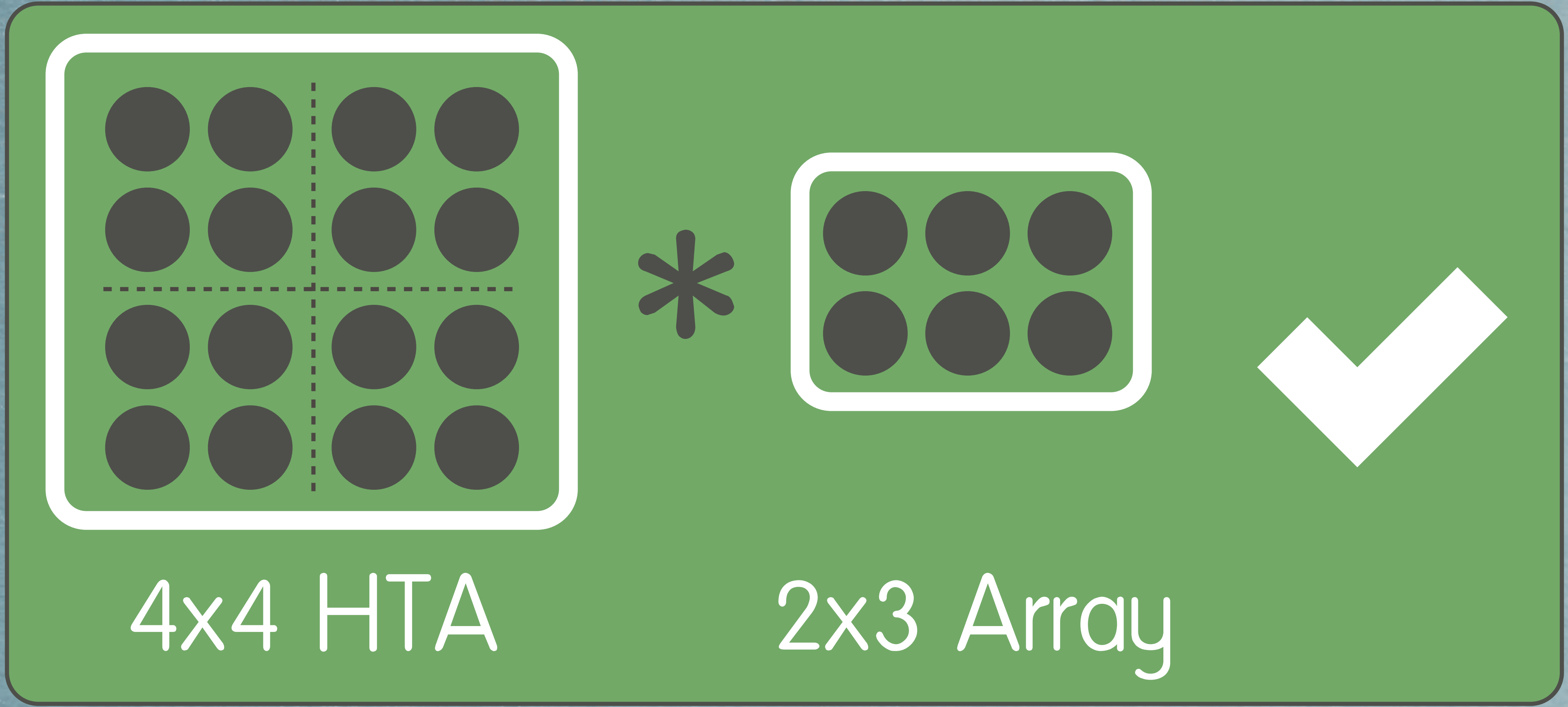
# ASSIGNMENTS & BINARY OPERATORS

VALID OPERATION ?



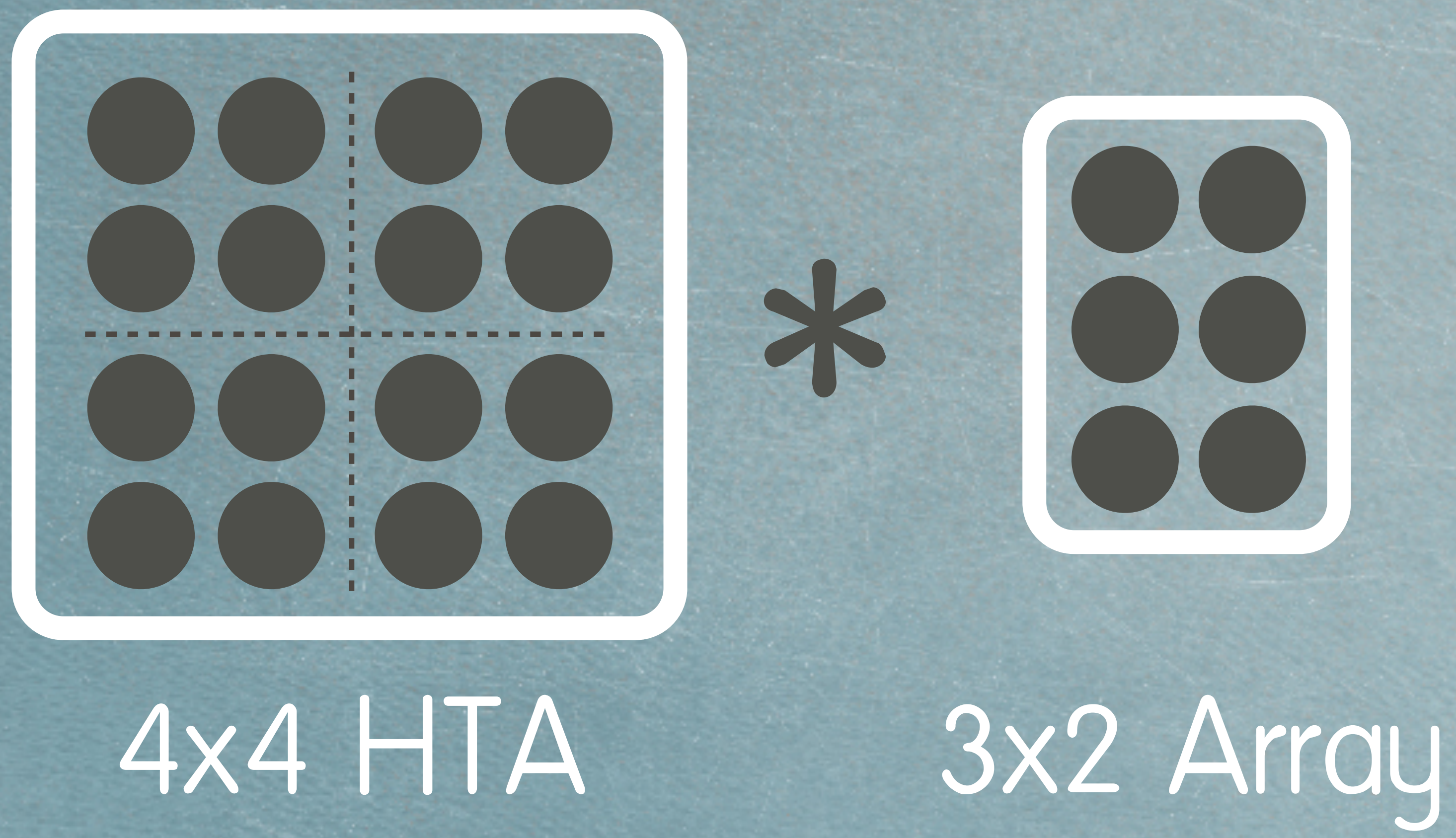
# VALID OPERATION ?

## ASSIGNMENTS & BINARY OPERATORS



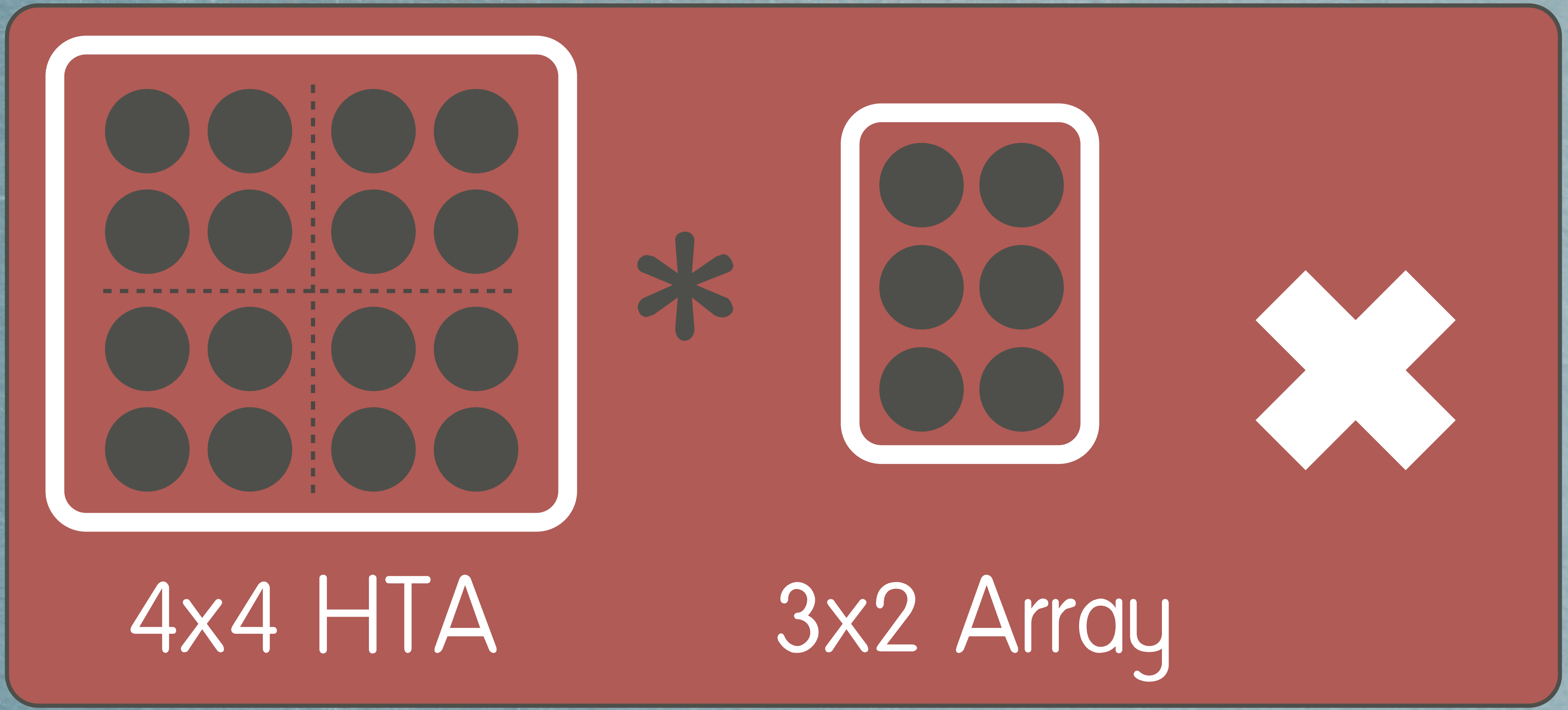
# VALID OPERATION ?

# ASSIGNMENTS & BINARY OPERATORS



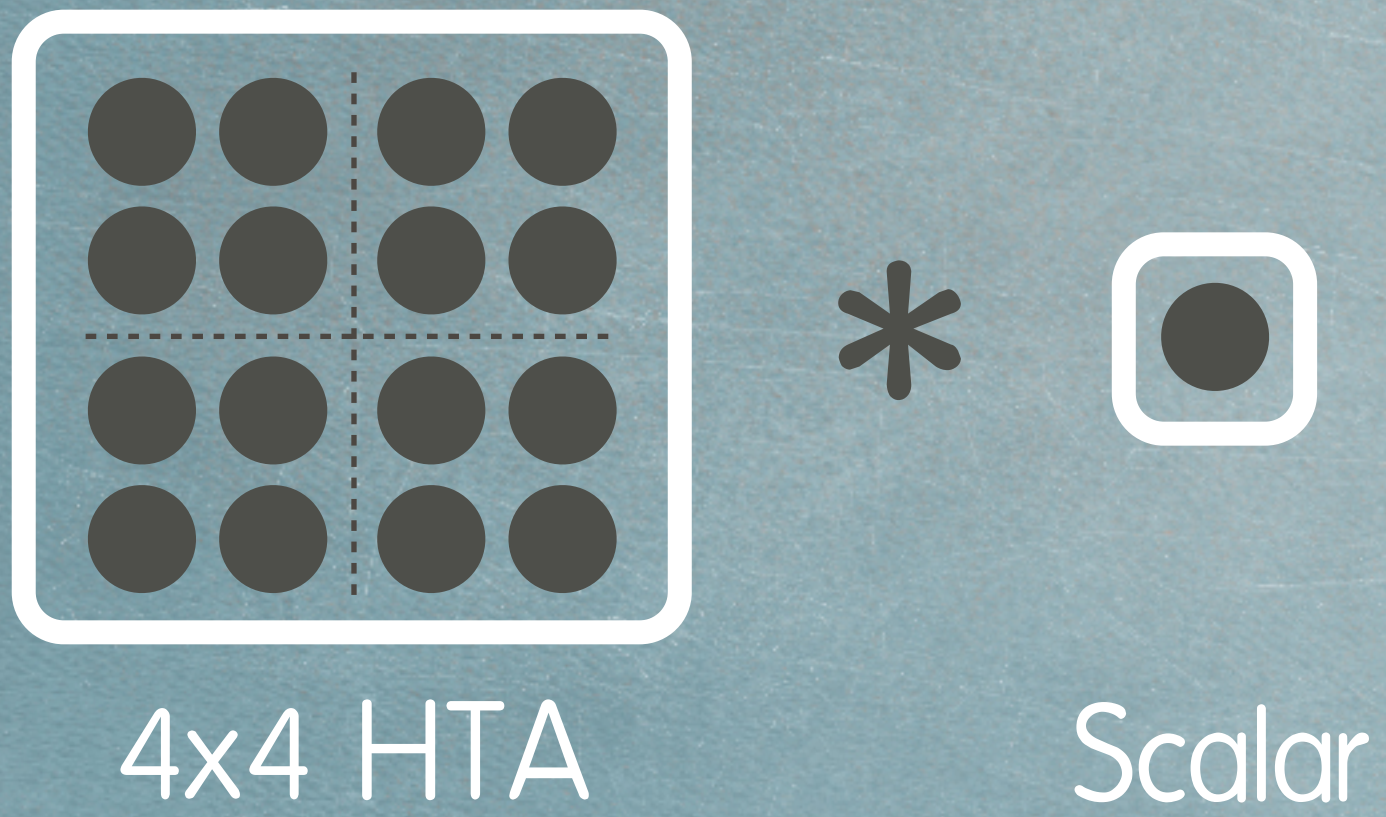
# VALID OPERATION ?

# ASSIGNMENTS & BINARY OPERATORS



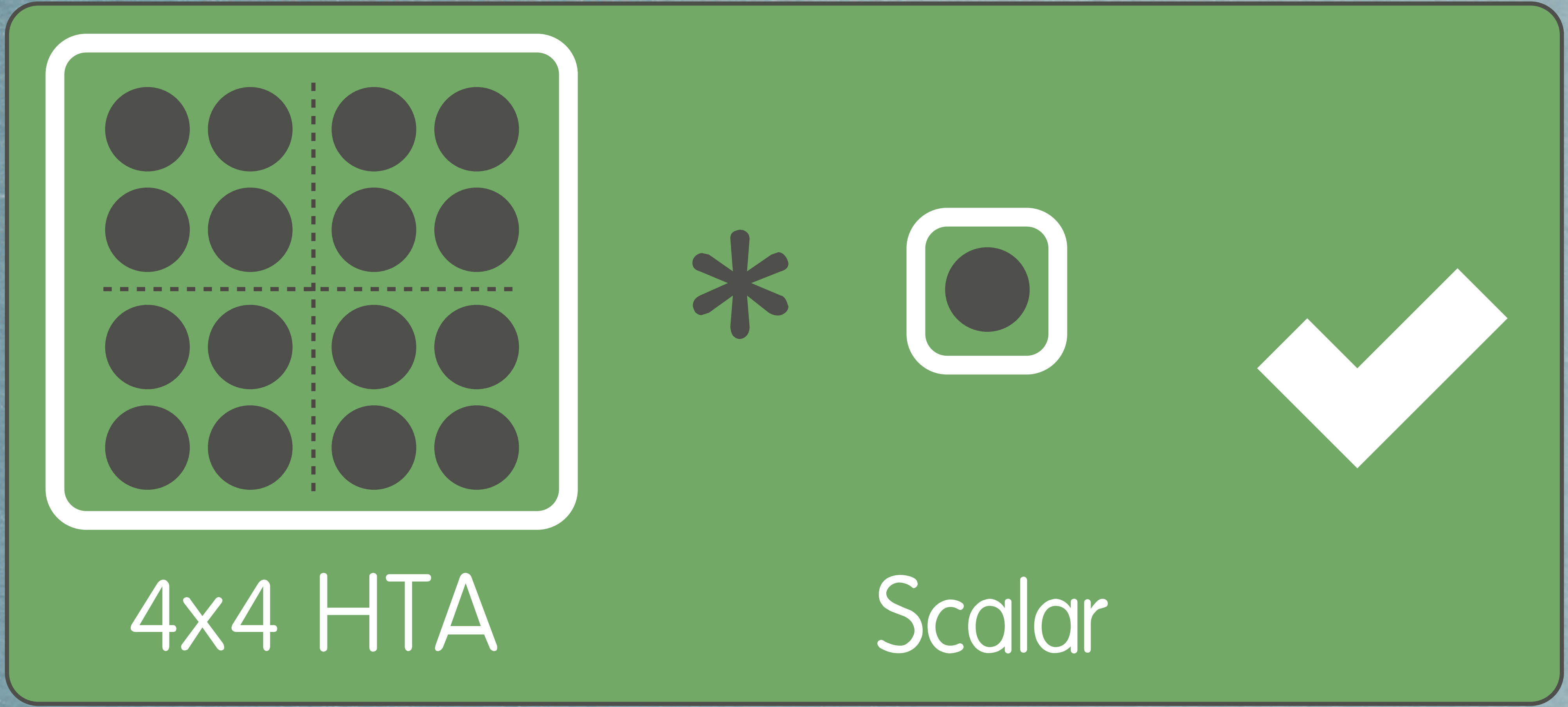
# VALID OPERATION ?

ASSIGNMENTS
& BINARY
OPERATORS



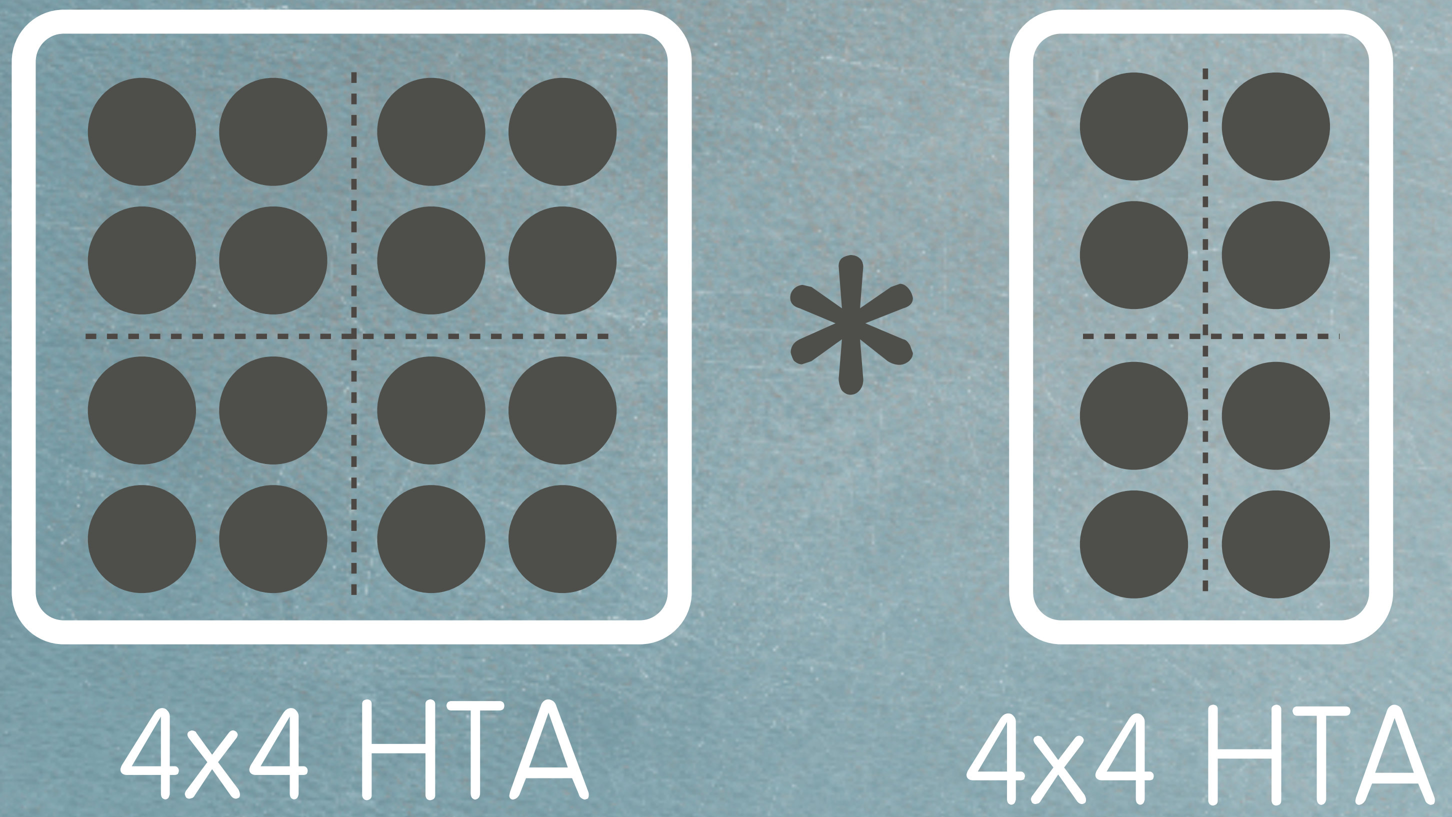
# VALID OPERATION ?

## ASSIGNMENTS & BINARY OPERATORS



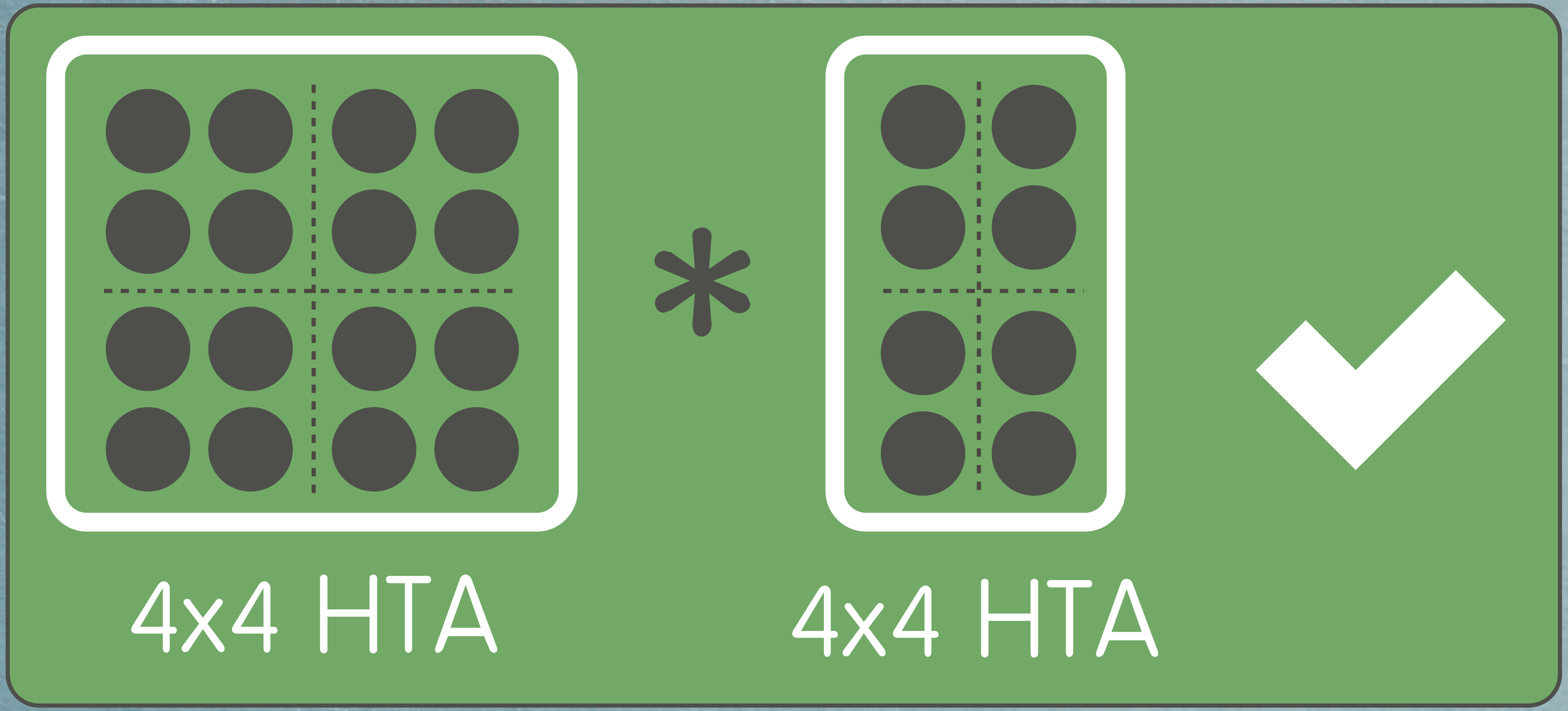
# ASSIGNMENTS & BINARY OPERATORS

## VALID OPERATION ?



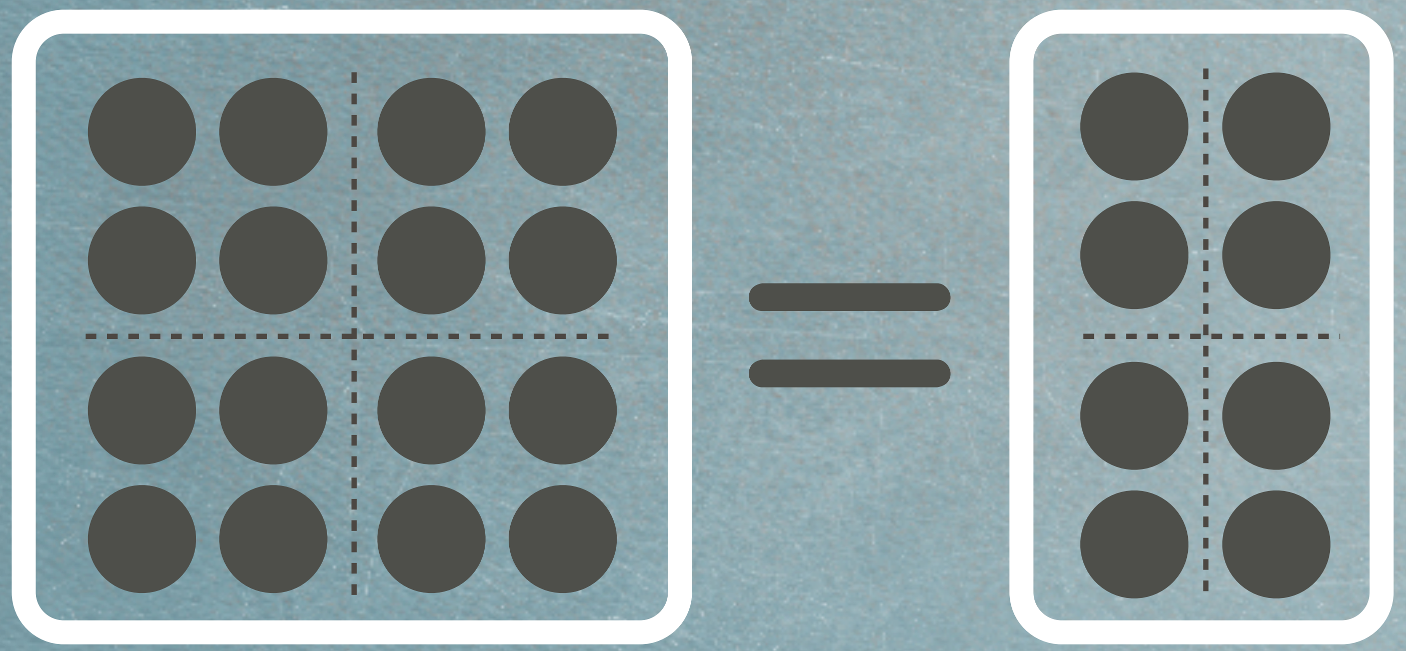
## VALID OPERATION ?

# ASSIGNMENTS & BINARY OPERATORS



## VALID OPERATION ?

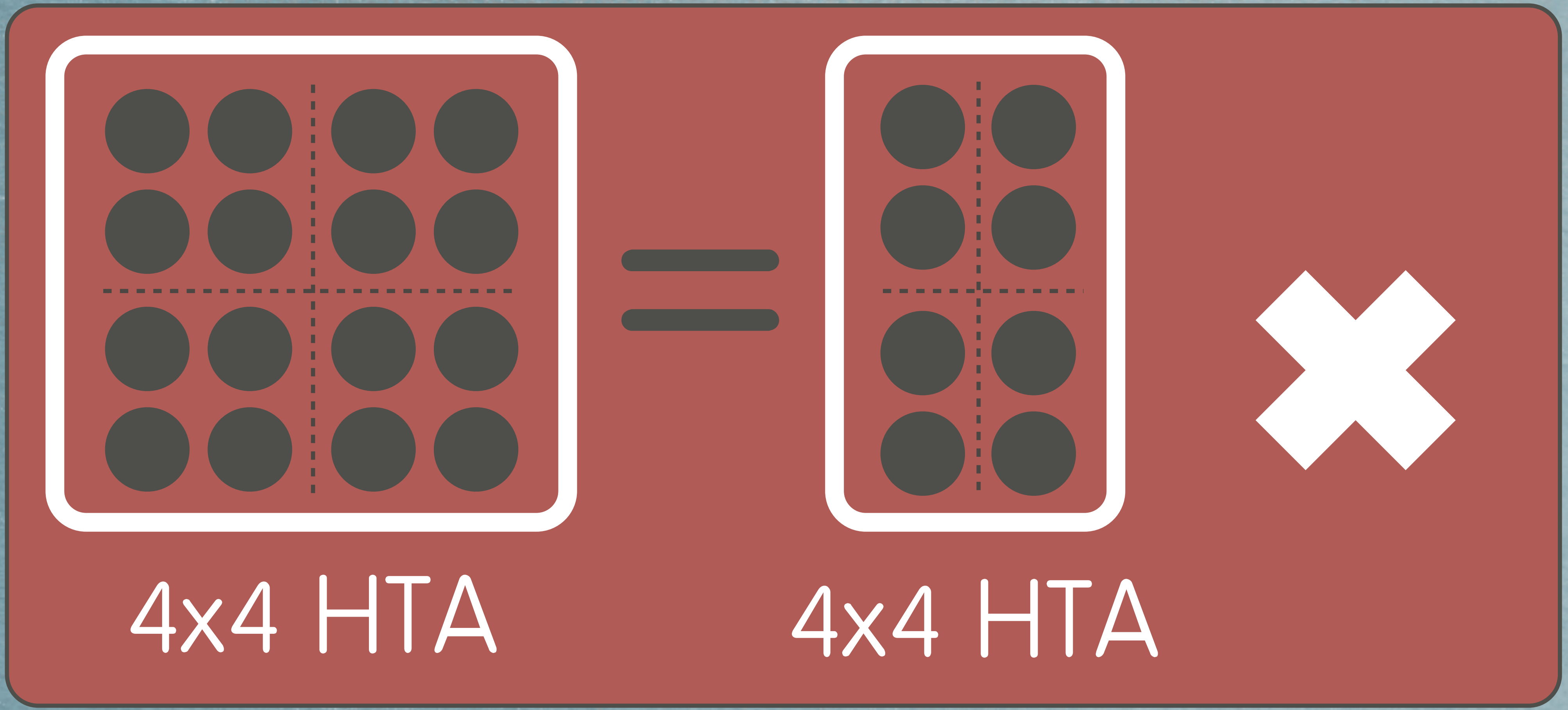
# ASSIGNMENTS & BINARY OPERATORS



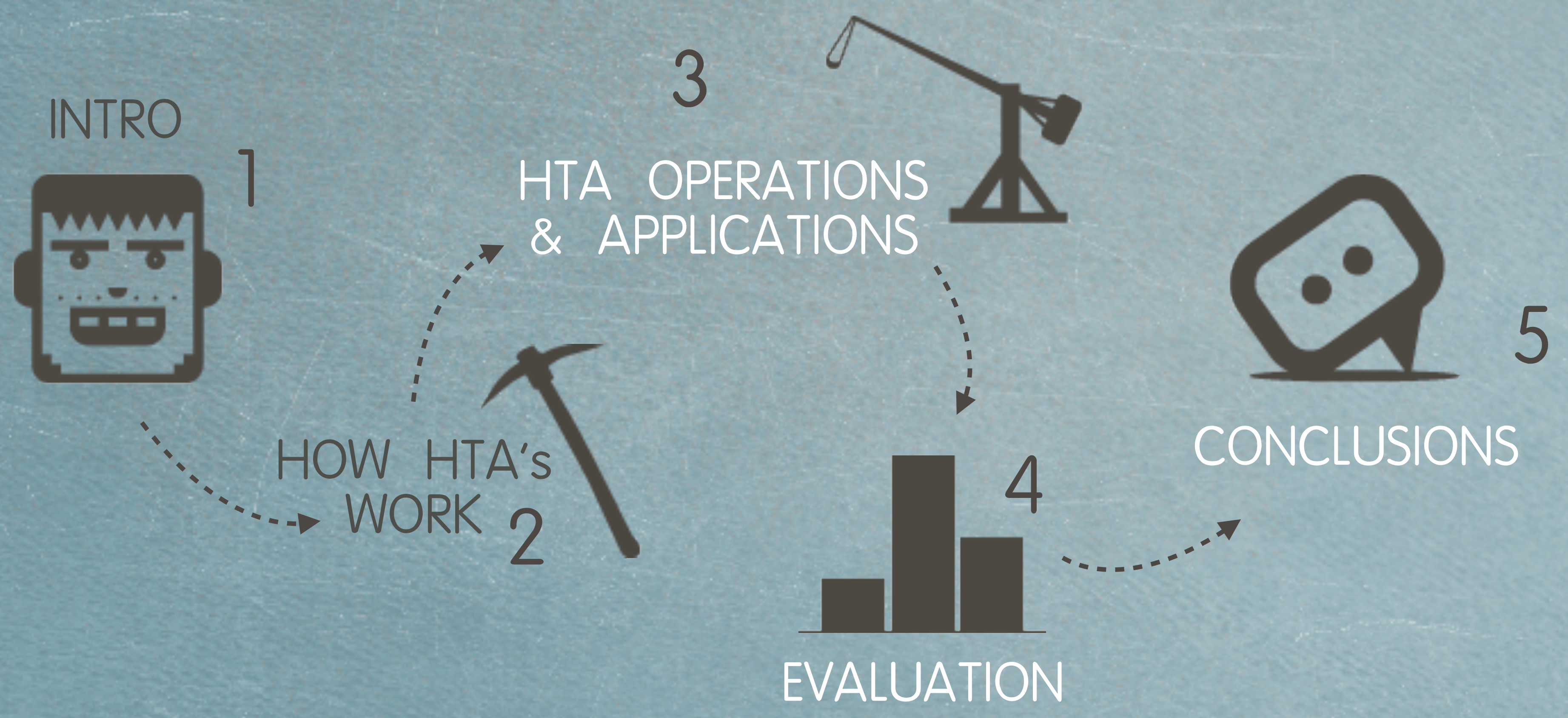
4x4 HTA 4x4 HTA

VALID OPERATION ?

# ASSIGNMENTS & BINARY OPERATORS

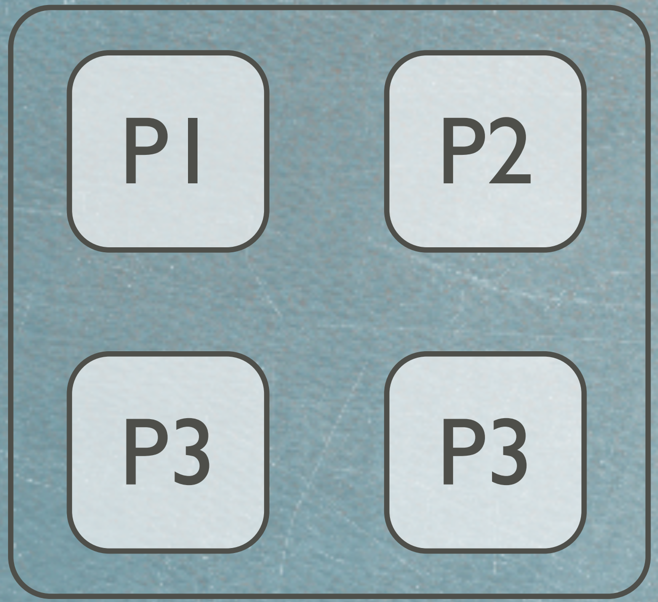


# TALK OVERVIEW

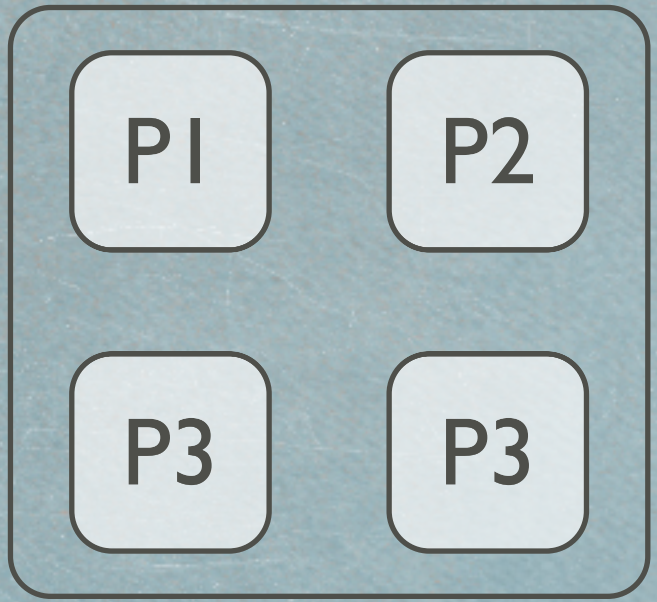


# TWO KINDS OF OPERATIONS

## COMMUNICATION OPERATIONS



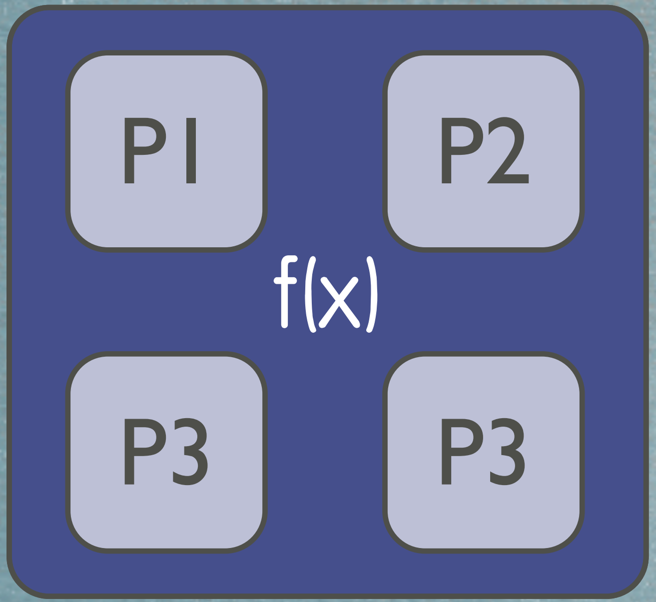

## GLOBAL COMPUTATIONS



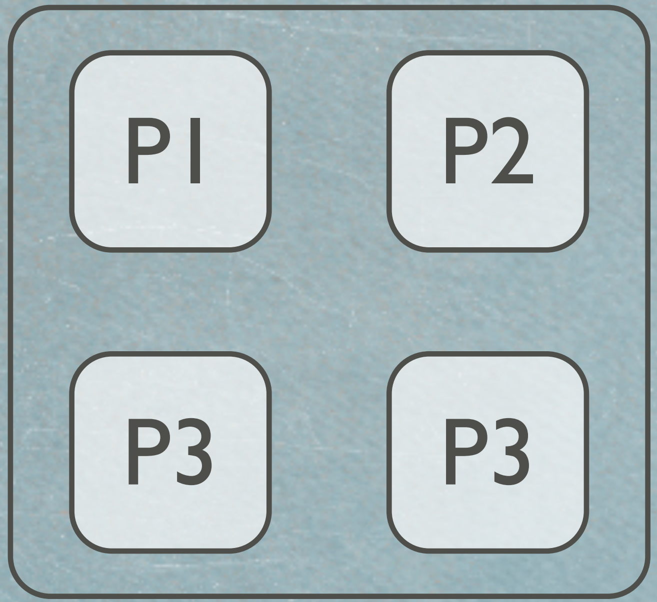



# TWO KINDS OF OPERATIONS

## COMMUNICATION OPERATIONS

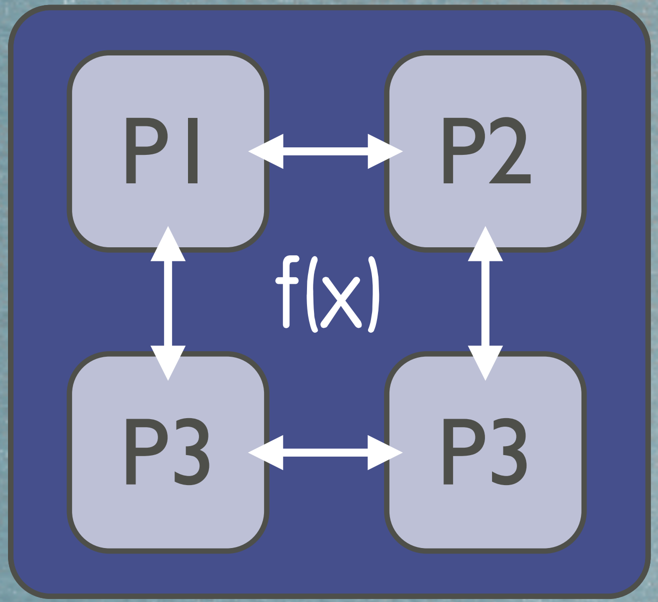



## GLOBAL COMPUTATIONS

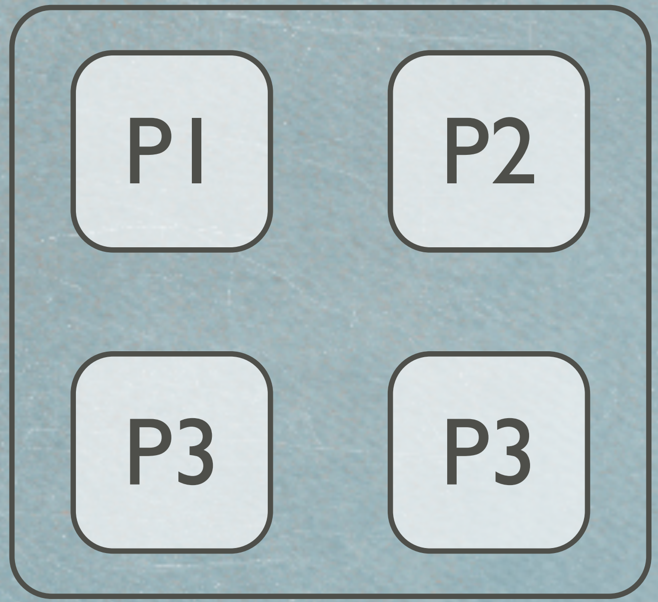


# TWO KINDS OF OPERATIONS

## COMMUNICATION OPERATIONS

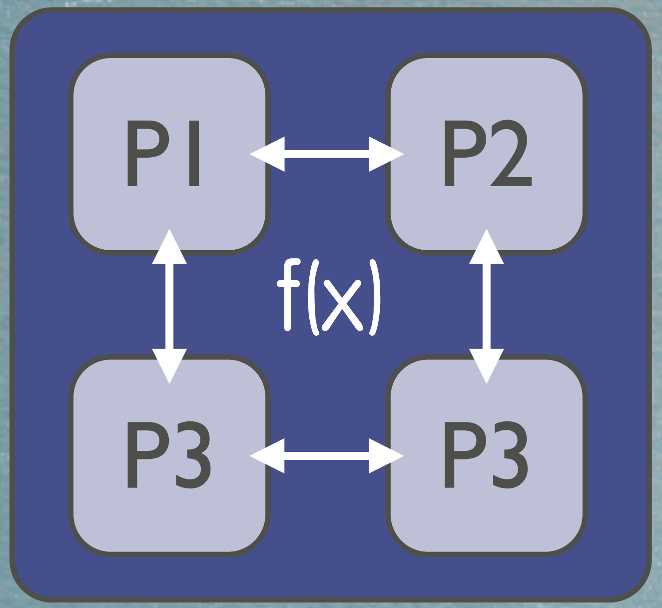



## GLOBAL COMPUTATIONS



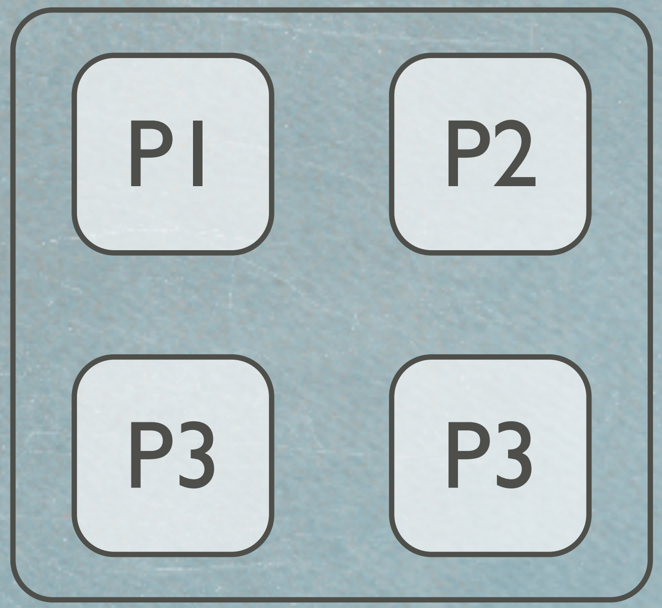
# TWO KINDS OF OPERATIONS

## COMMUNICATION OPERATIONS



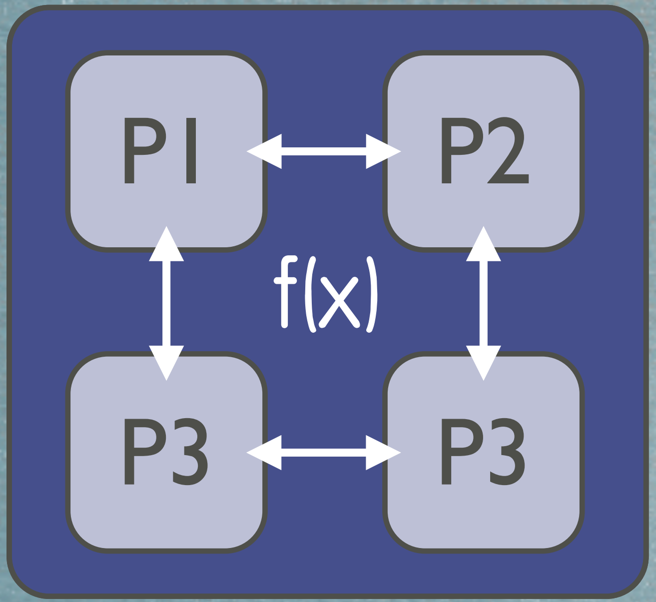

Assignments, repmat, circshift, permute

## GLOBAL COMPUTATIONS



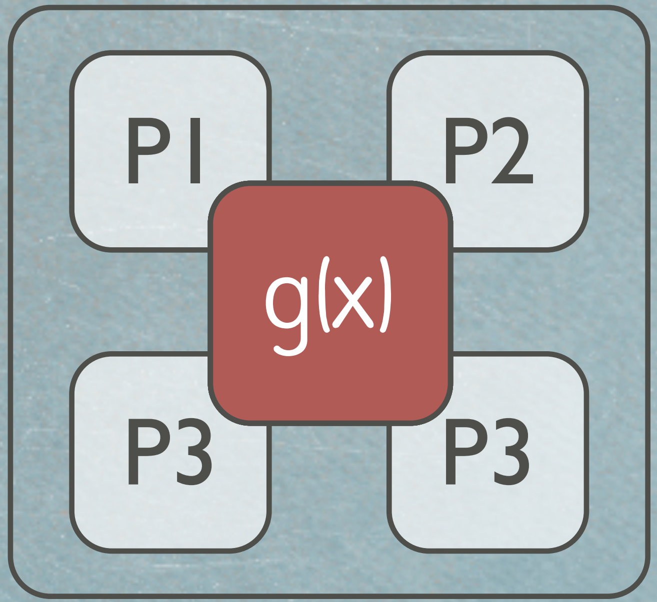
# TWO KINDS OF OPERATIONS

## COMMUNICATION OPERATIONS



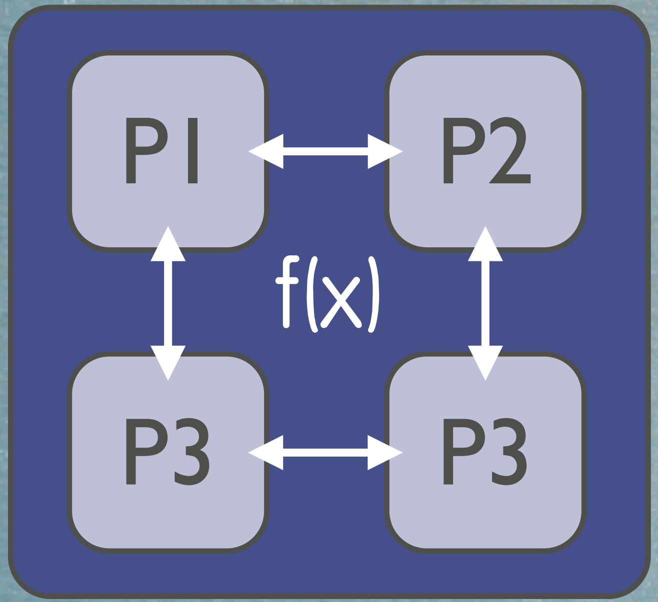

Assignments, repmat, circshift, permute

## GLOBAL COMPUTATIONS



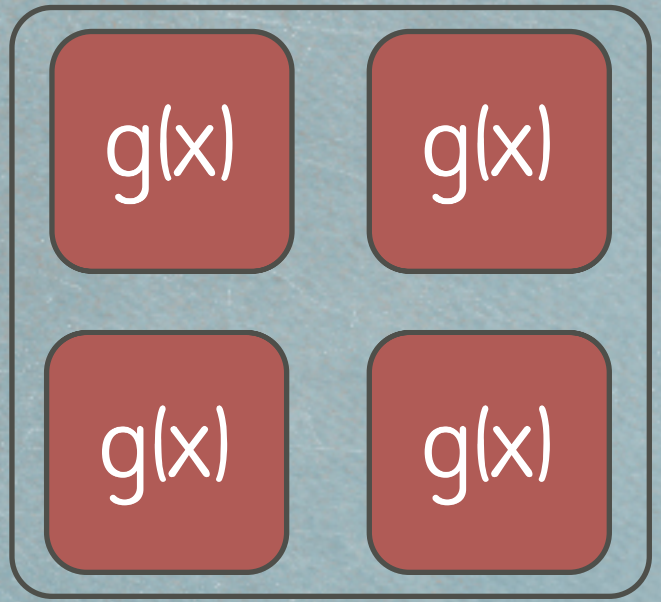
# TWO KINDS OF OPERATIONS

## COMMUNICATION OPERATIONS



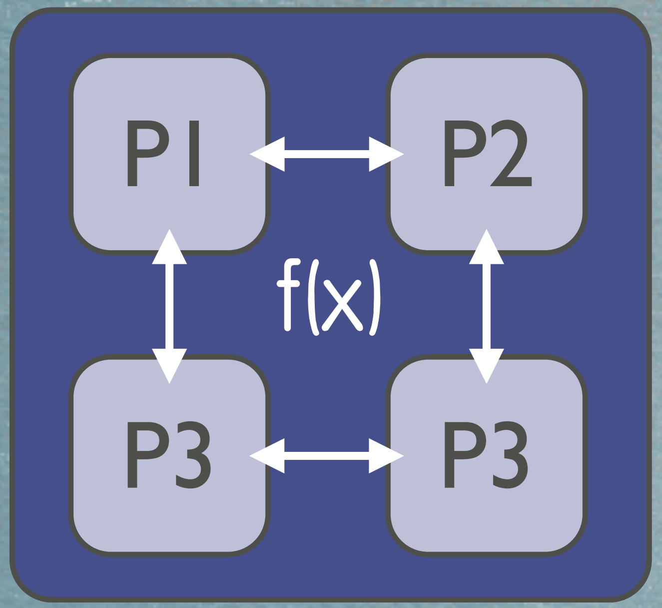

Assignments, repmat, circshift, permute

## GLOBAL COMPUTATIONS



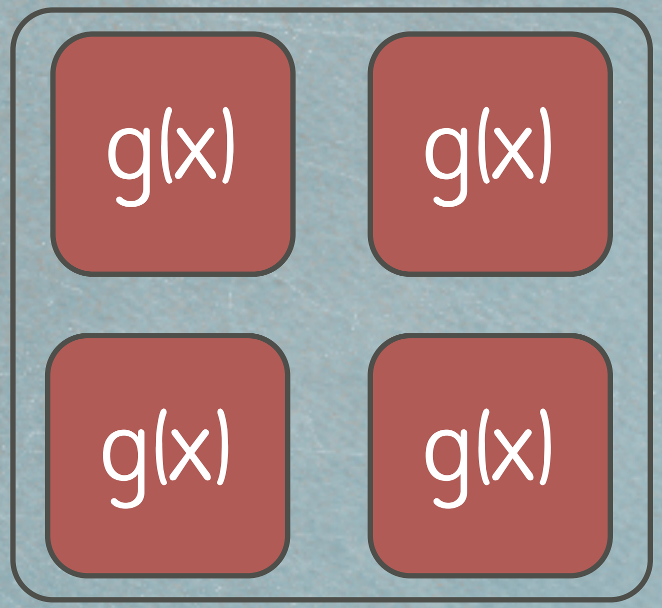
# TWO KINDS OF OPERATIONS

## COMMUNICATION OPERATIONS




Assignments, repmat, circshift, permute

## GLOBAL COMPUTATIONS




parHTA(@g(x), H)

# CANNON'S ALGORITHM

```
function C = cannon(A,B,C)
  for i=2:m
    A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
    B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
  end
  for k=1:m
    C = C + A * B;
    A = circshift(A, [0, -1]);
    B = circshift(B, [-1, 0]);
  end
```

# CANNON'S ALGORITHM

```
function C = cannon(A,B,C)
  for i=2:m
    A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
    B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
  end
  for k=1:m
    C = C + A * B;
    A = circshift(A, [0, -1]);
    B = circshift(B, [-1, 0]);
  end
```

# CANNON'S ALGORITHM

```
function C = cannon(A,B,C)
  for i=2:m
    A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
    B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
  end
  for k=1:m
    C = C + A * B;
    A = circshift(A, [0, -1]);
    B = circshift(B, [-1, 0]);
  end
```

# CANNON'S ALGORITHM

```
function C = cannon(A,B,C)
  for i=2:m
    A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
    B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
  end
  for k=1:m
    C = C + A * B;
    A = circshift(A, [0, -1]);
    B = circshift(B, [-1, 0]);
  end
```

# CANNON'S ALGORITHM

```
function C = cannon(A,B,C)
  for i=2:m
    A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
    B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
  end
  for k=1:m
    C = C + A * B;
    A = circshift(A, [0, -1]);
    B = circshift(B, [-1, 0]);
  end
```

# CANNON'S ALGORITHM

```
function C = cannon(A,B,C)
  for i=2:m
    A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
    B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
  end
  for k=1:m
    C = C + A * B;
    A = circshift(A, [0, -1]);
    B = circshift(B, [-1, 0]);
  end
```

Initialization

# CANNON'S ALGORITHM

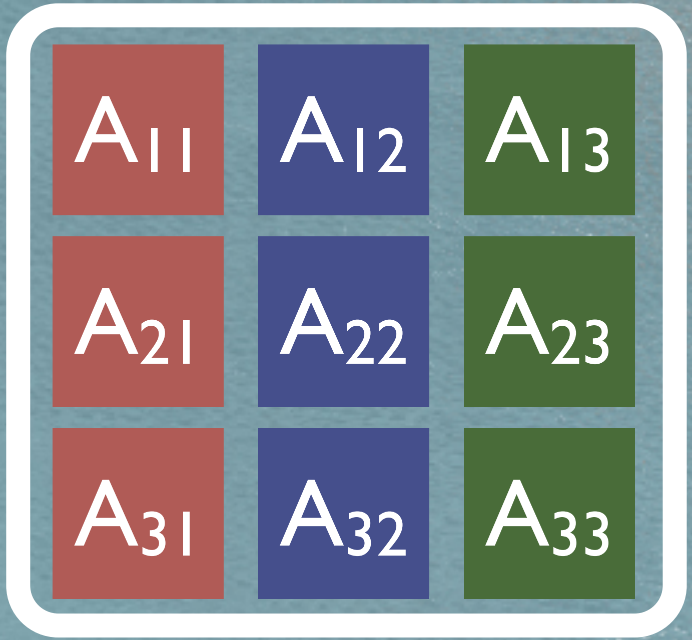
```
function C = cannon(A,B,C)
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```

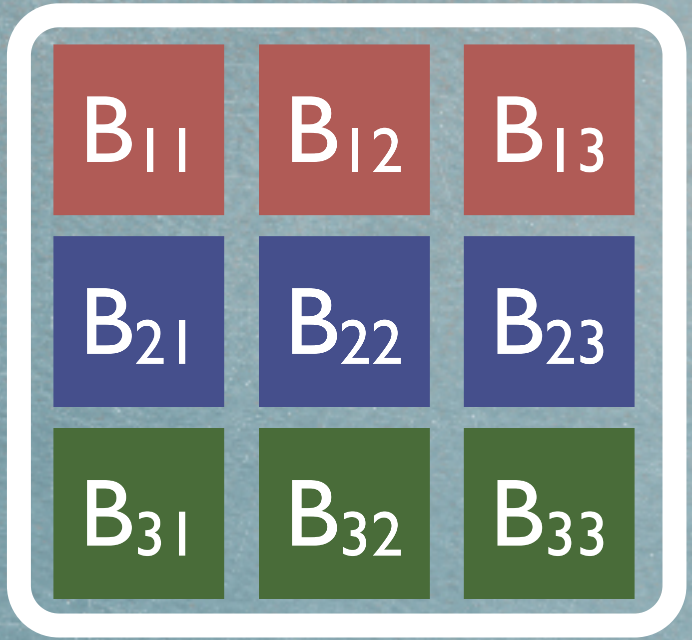
Initialization

Iteration

Initialization

```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```

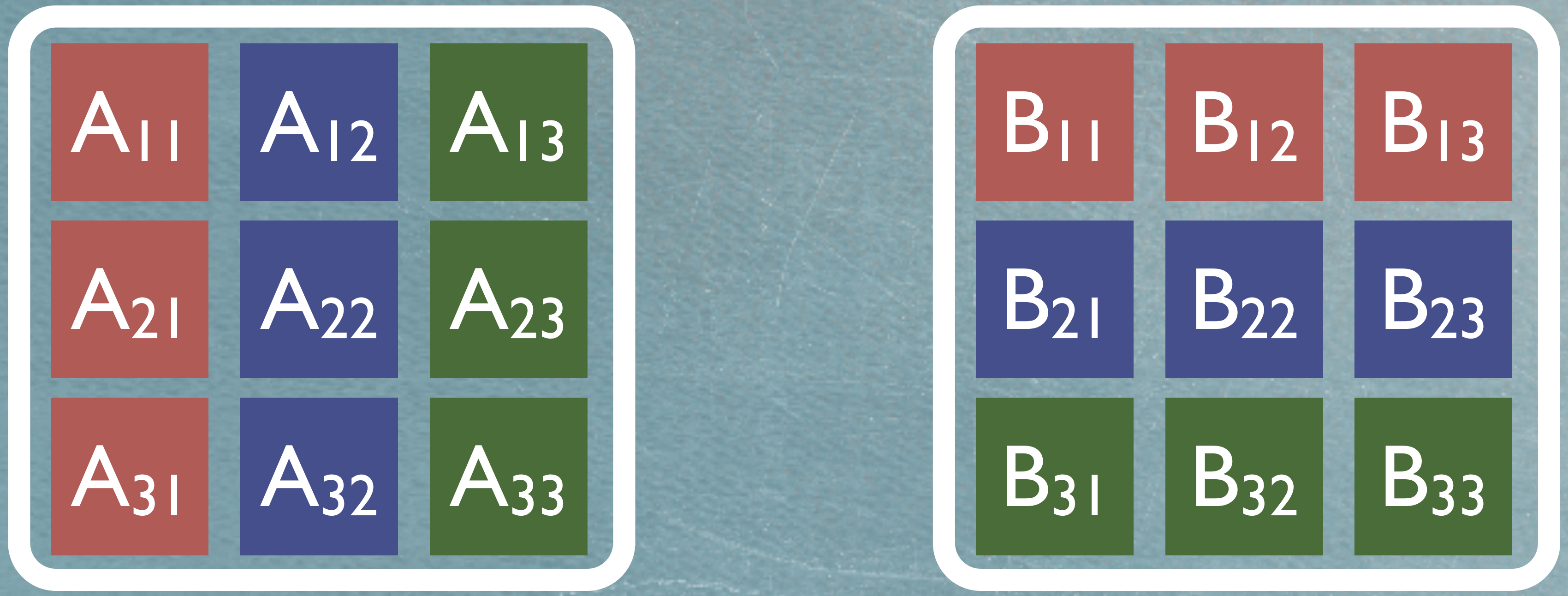


i=2

Initialization

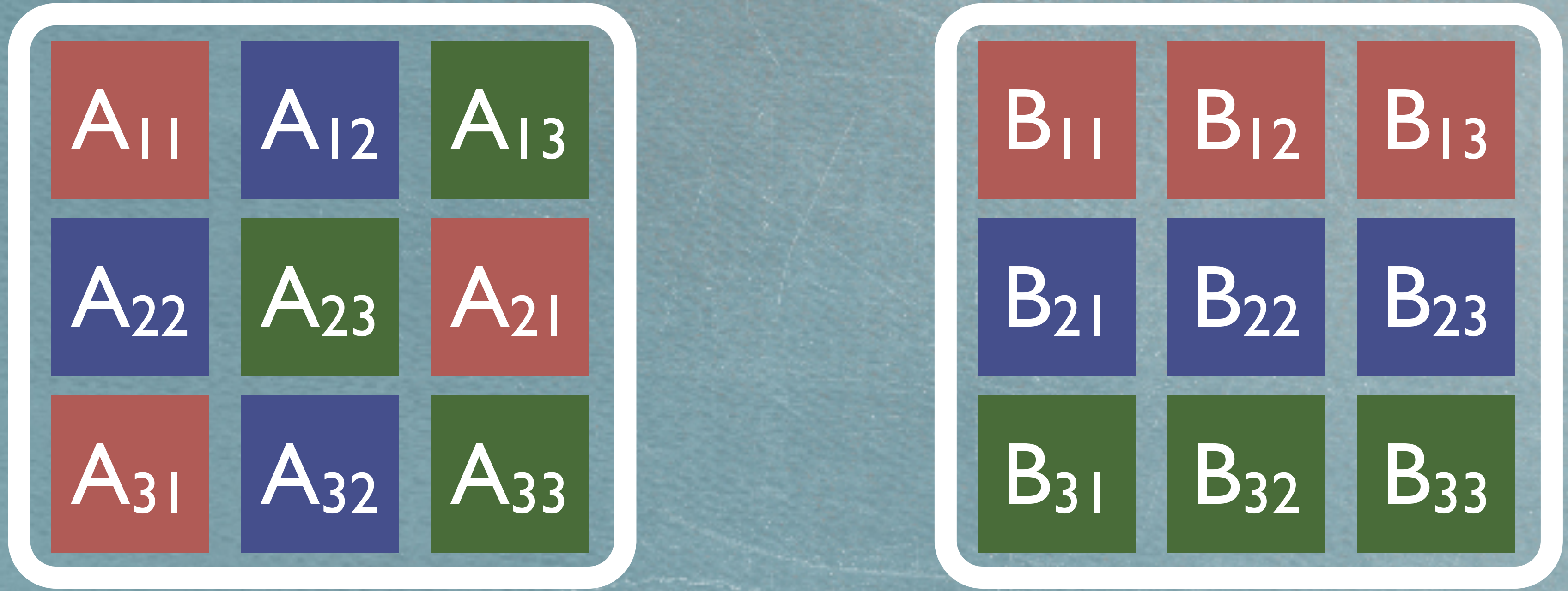
```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```



i=2

Initialization

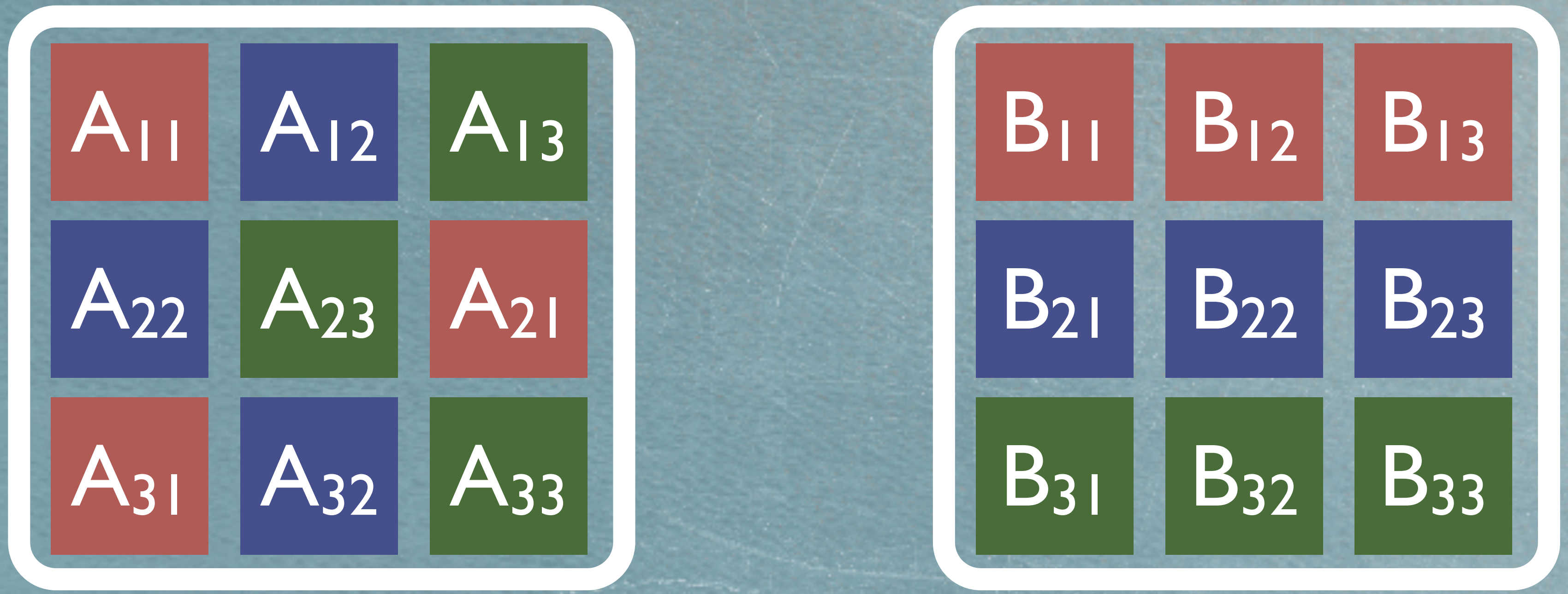
```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```



i=2

Initialization

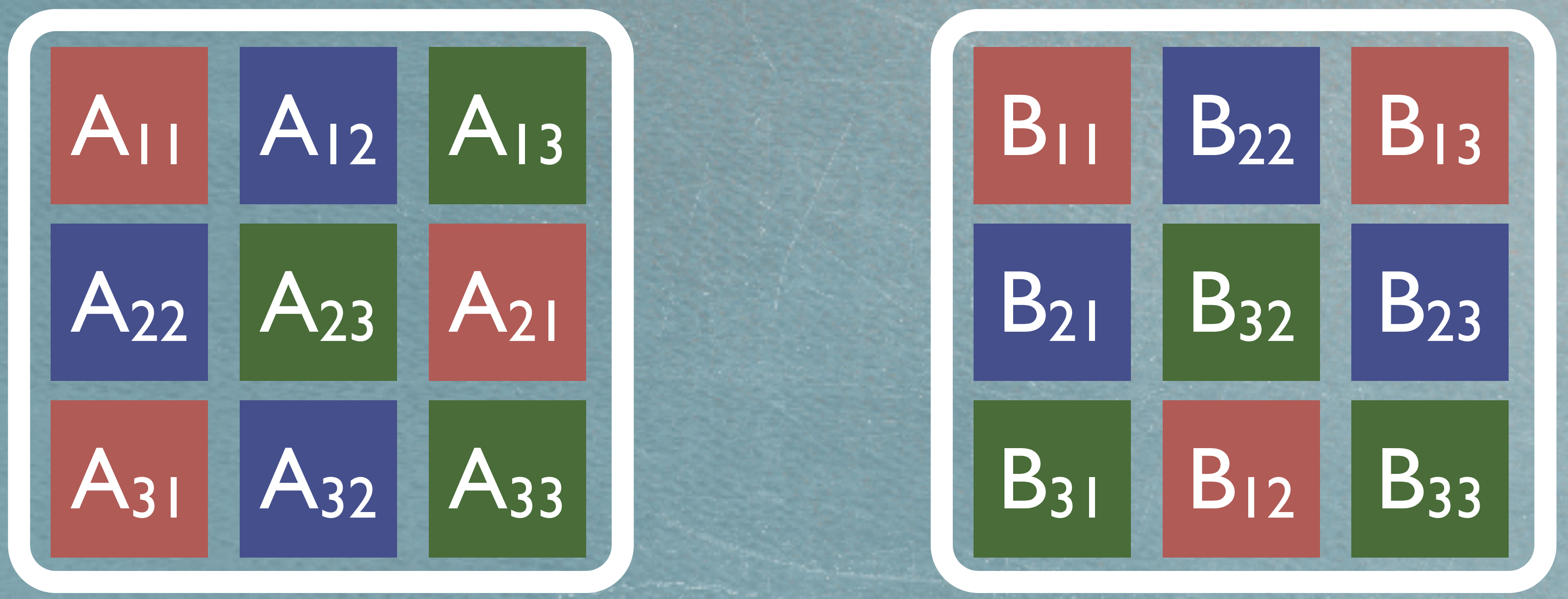
```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```



i=2

Initialization

```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```



i=3

Initialization

```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```

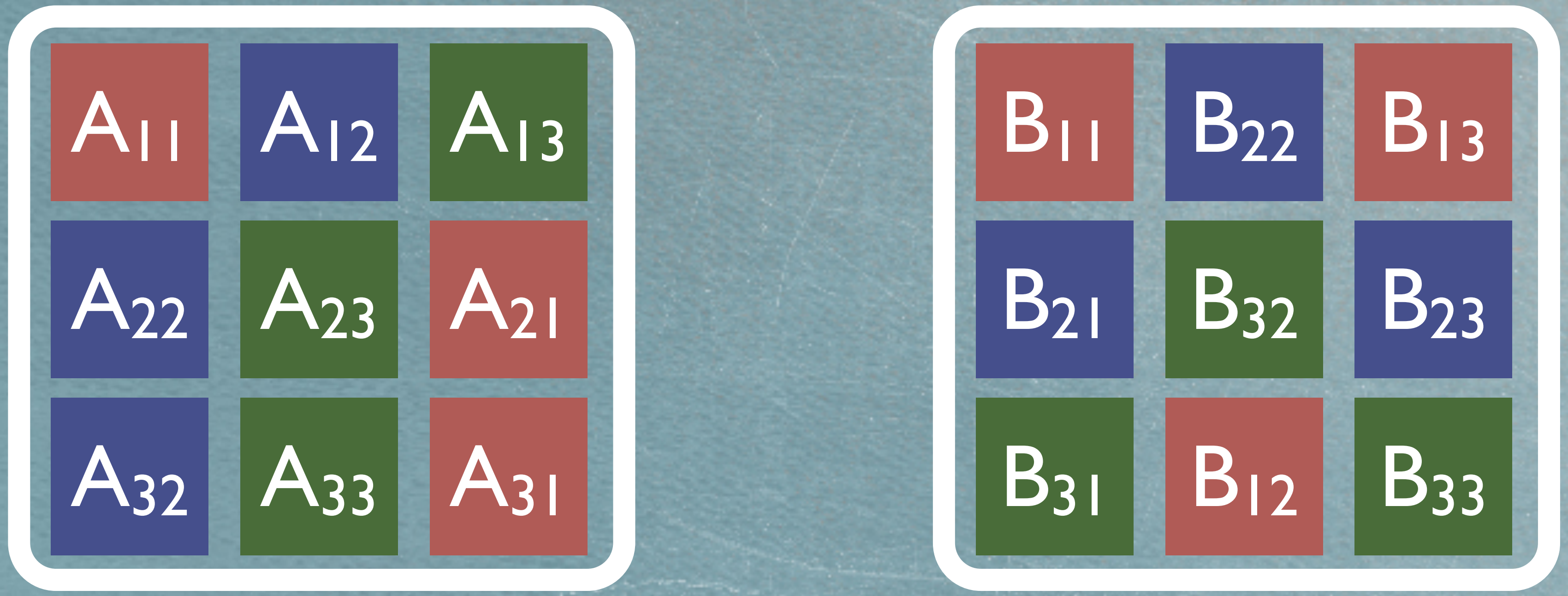
| | | |
|---|---|---|
| $A_{11}$ | $A_{12}$ | $A_{13}$ |
| $A_{22}$ | $A_{23}$ | $A_{21}$ |
| $A_{31}$ | $A_{32}$ | $A_{33}$ |

| | | |
|---|---|---|
| $B_{11}$ | $B_{22}$ | $B_{13}$ |
| $B_{21}$ | $B_{32}$ | $B_{23}$ |
| $B_{31}$ | $B_{12}$ | $B_{33}$ |

i=3

Initialization

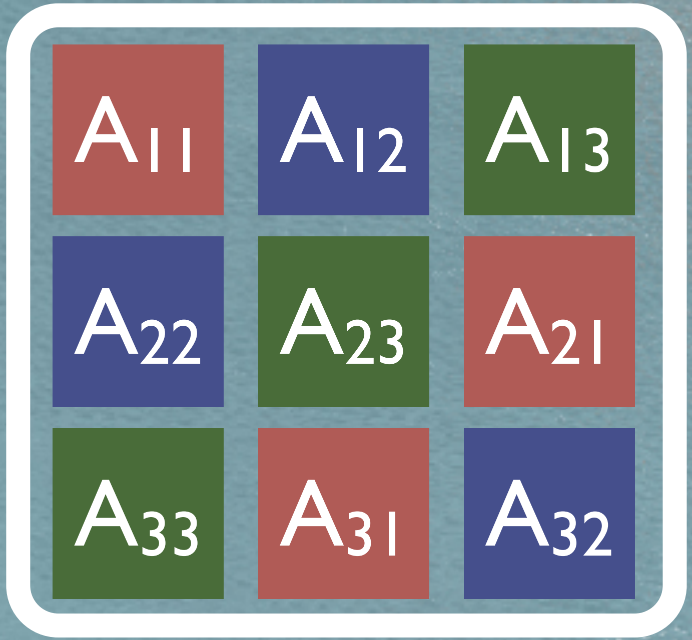
```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```

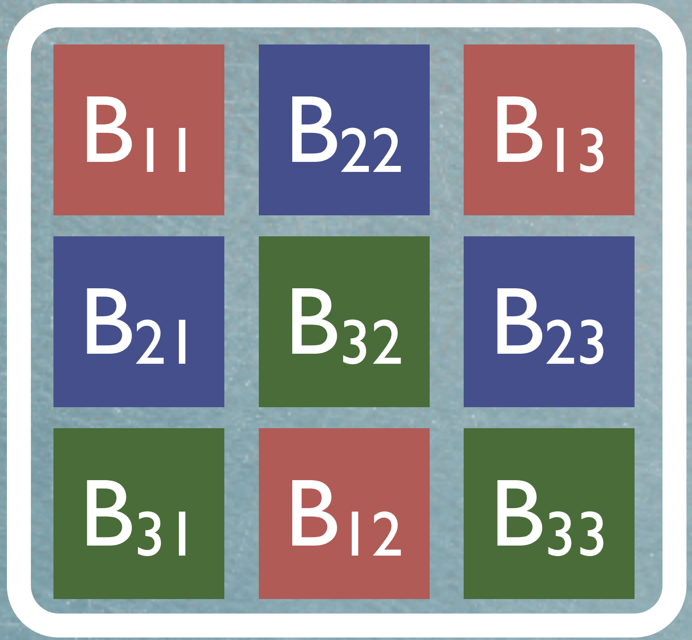


i=3

```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```

Initialization

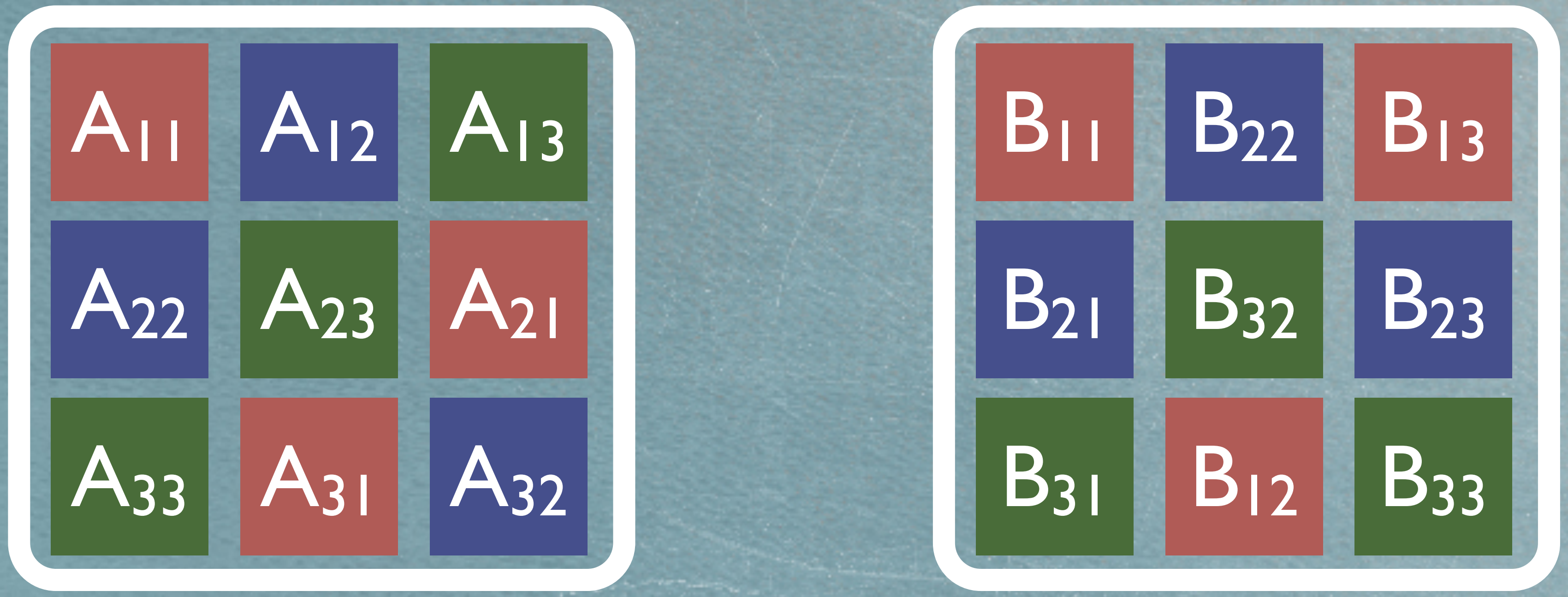


i=3

Initialization

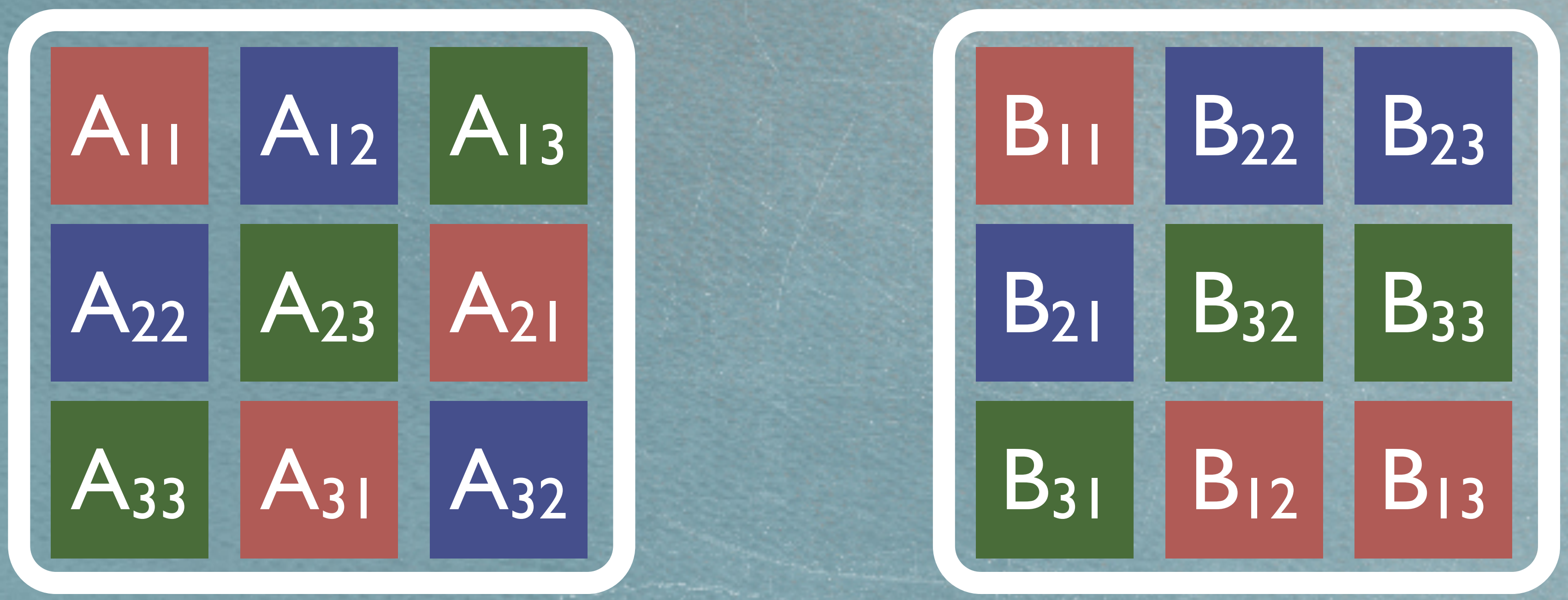
```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```



i=3

Initialization

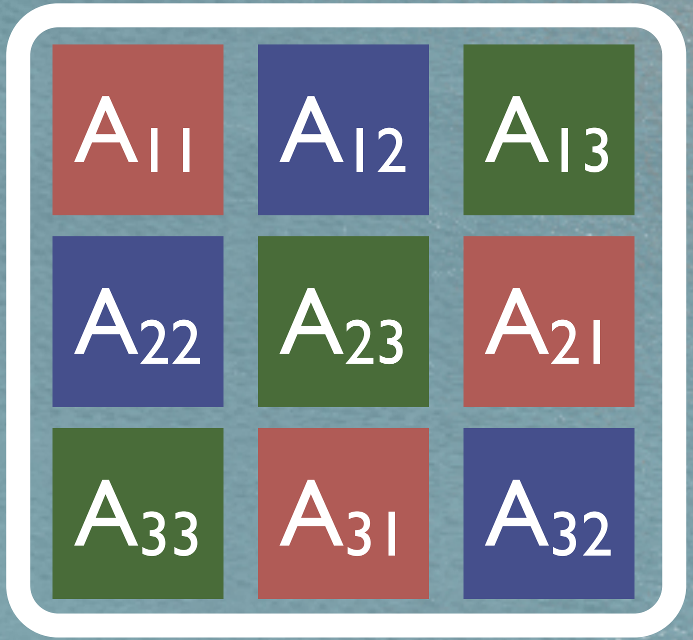
```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```

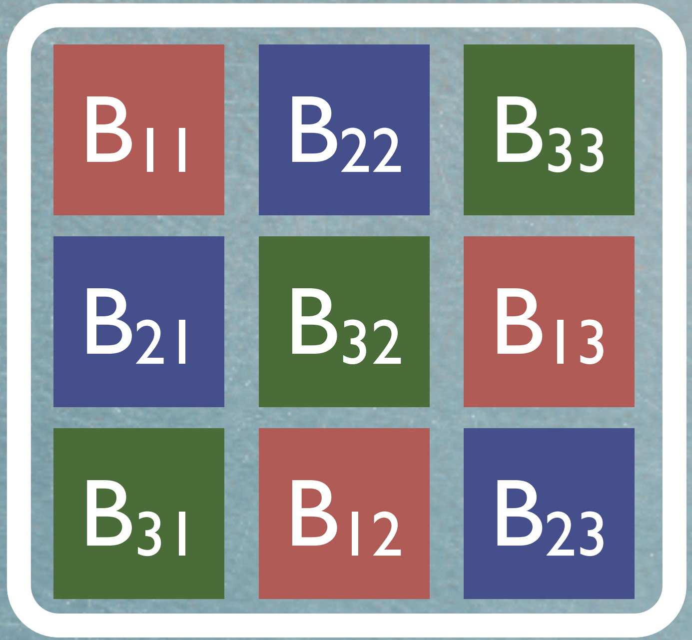


i=3

Initialization

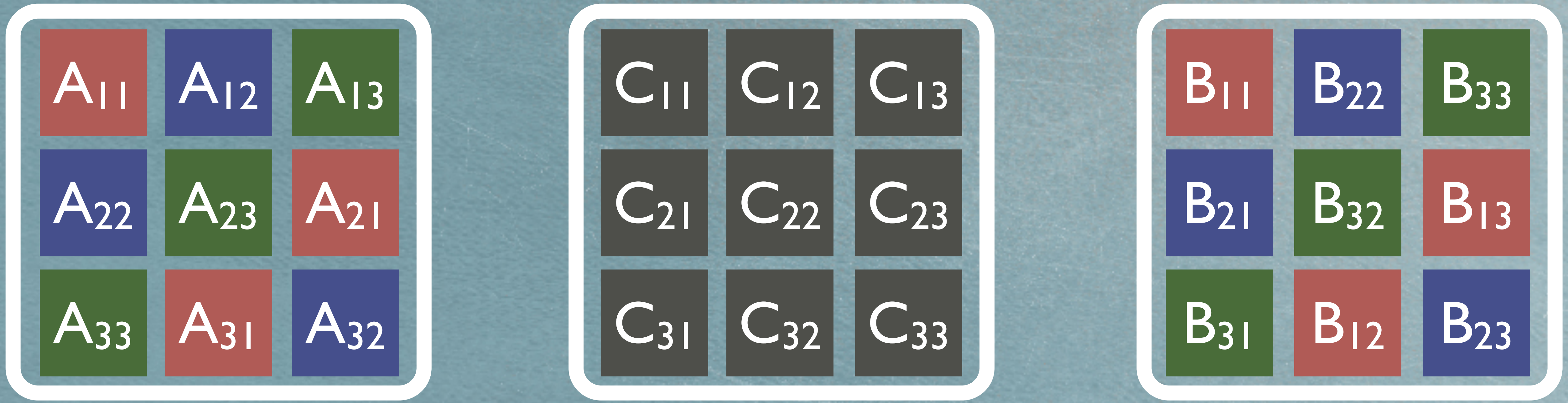
```
for i=2:m
  A{i,:} = circshift(A{i,:}, [0, -(i-1)]);
  B(:,i} = circshift(B{:,i}, [-(i-1), 0]);
end
```



Iteration

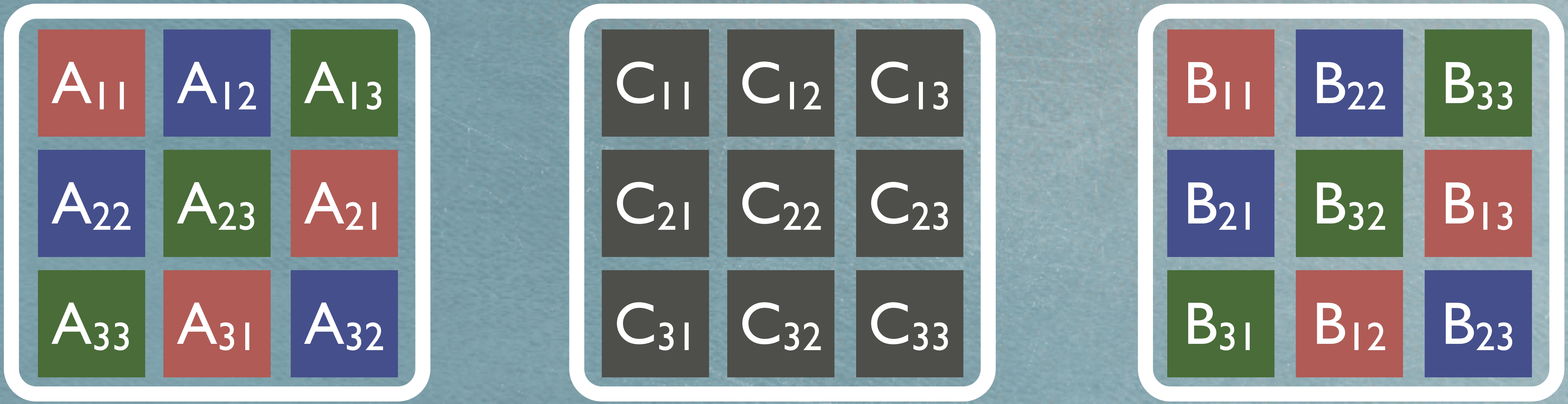
```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```



k=1

```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```

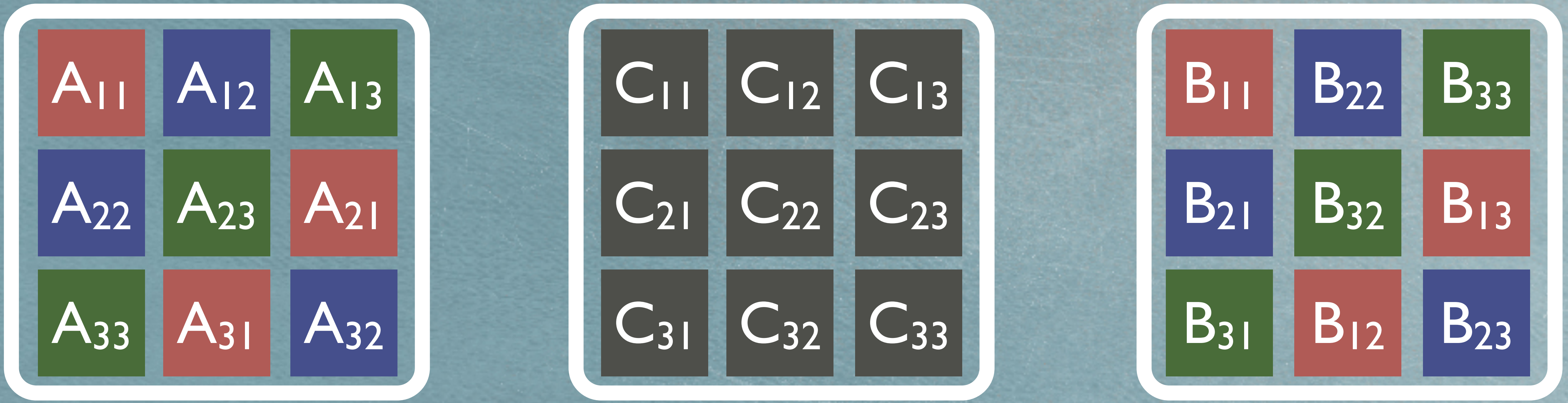
Iteration



k=1

Iteration

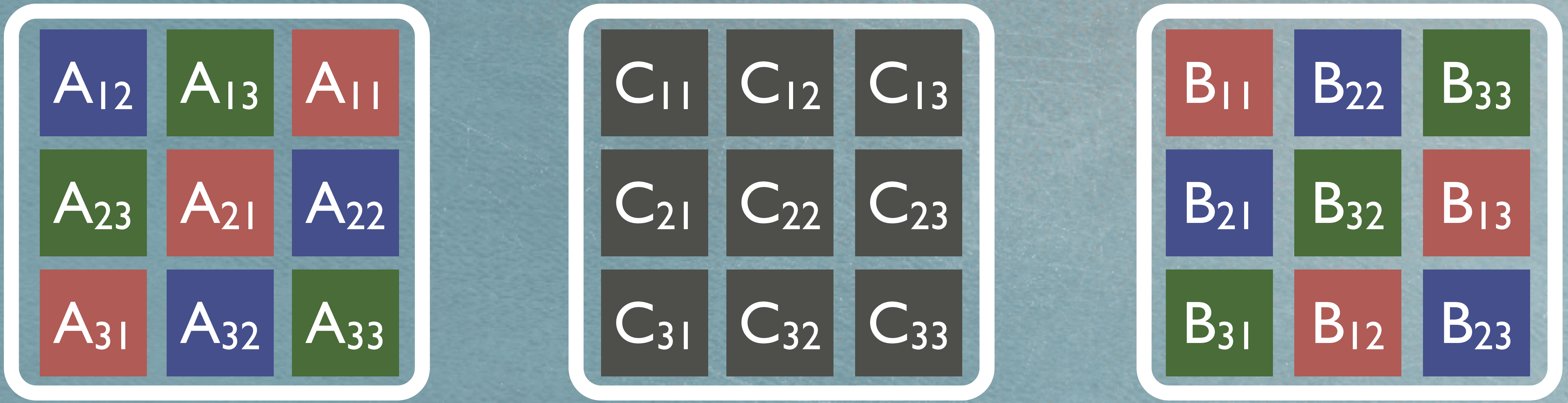
```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```



k=1

```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```

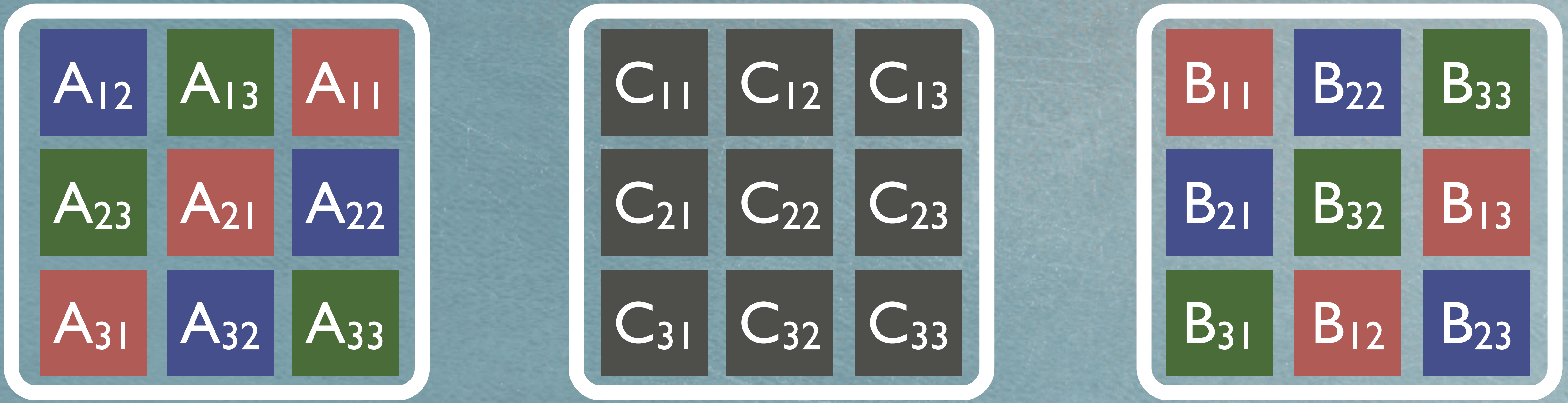
Iteration



k=1

Iteration

```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```



k=1

Iteration

```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
→ B = circshift(B, [-1, 0]);
end
```

| | | |
|---|---|---|
| $A_{12}$ | $A_{13}$ | $A_{11}$ |
| $A_{23}$ | $A_{21}$ | $A_{22}$ |
| $A_{31}$ | $A_{32}$ | $A_{33}$ |

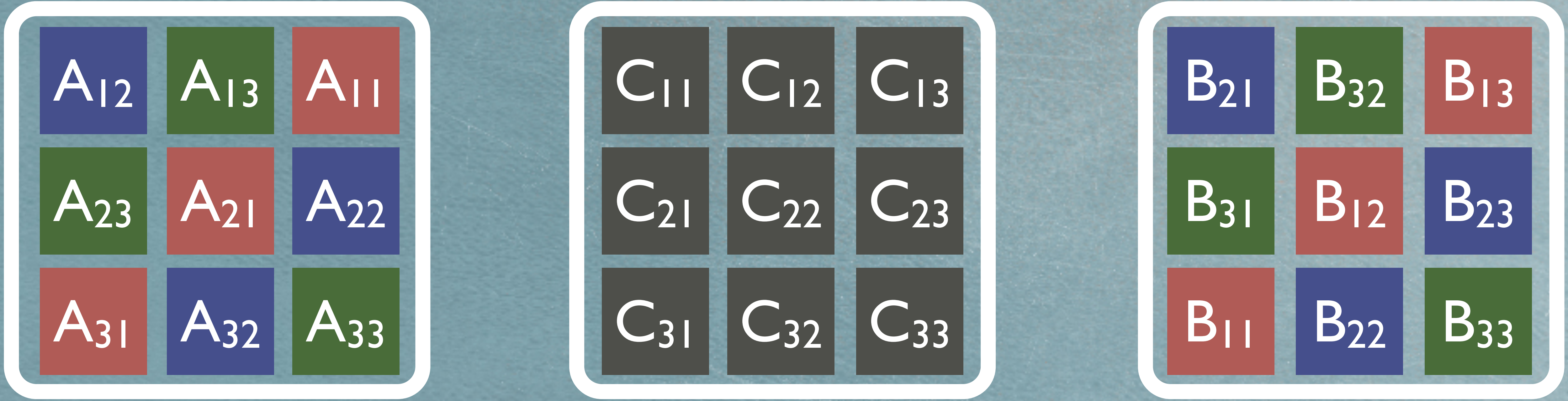
| | | |
|---|---|---|
| $C_{11}$ | $C_{12}$ | $C_{13}$ |
| $C_{21}$ | $C_{22}$ | $C_{23}$ |
| $C_{31}$ | $C_{32}$ | $C_{33}$ |

| | | |
|---|---|---|
| $B_{21}$ | $B_{32}$ | $B_{13}$ |
| $B_{31}$ | $B_{12}$ | $B_{23}$ |
| $B_{11}$ | $B_{22}$ | $B_{33}$ |

k=2

Iteration

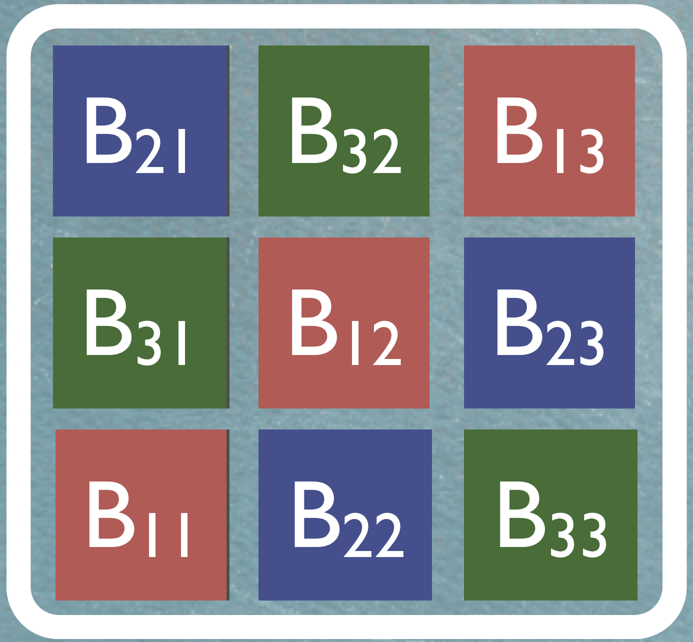
```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```



k=2

Iteration

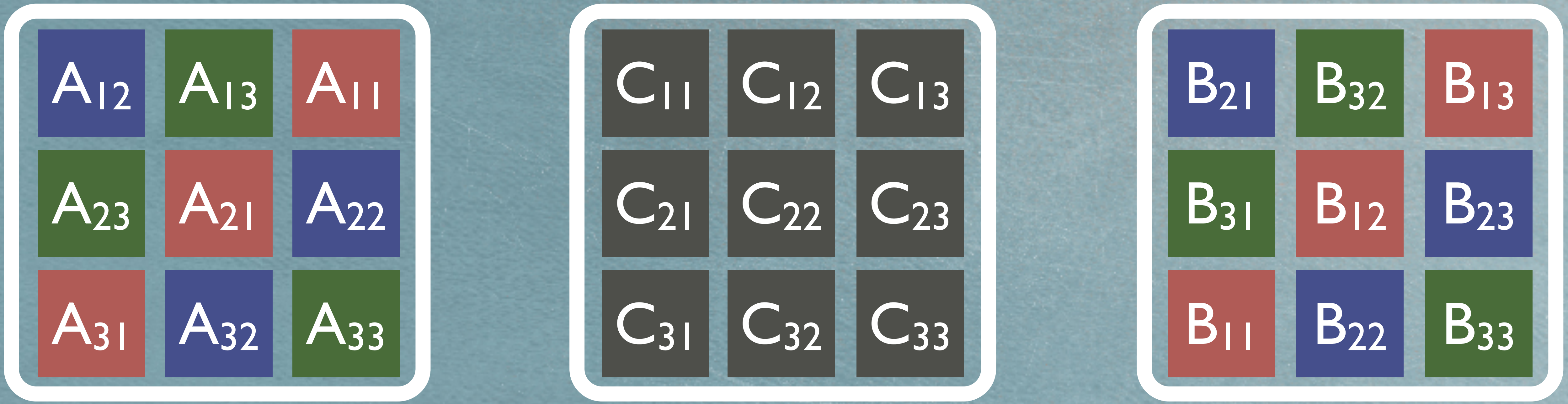
```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```



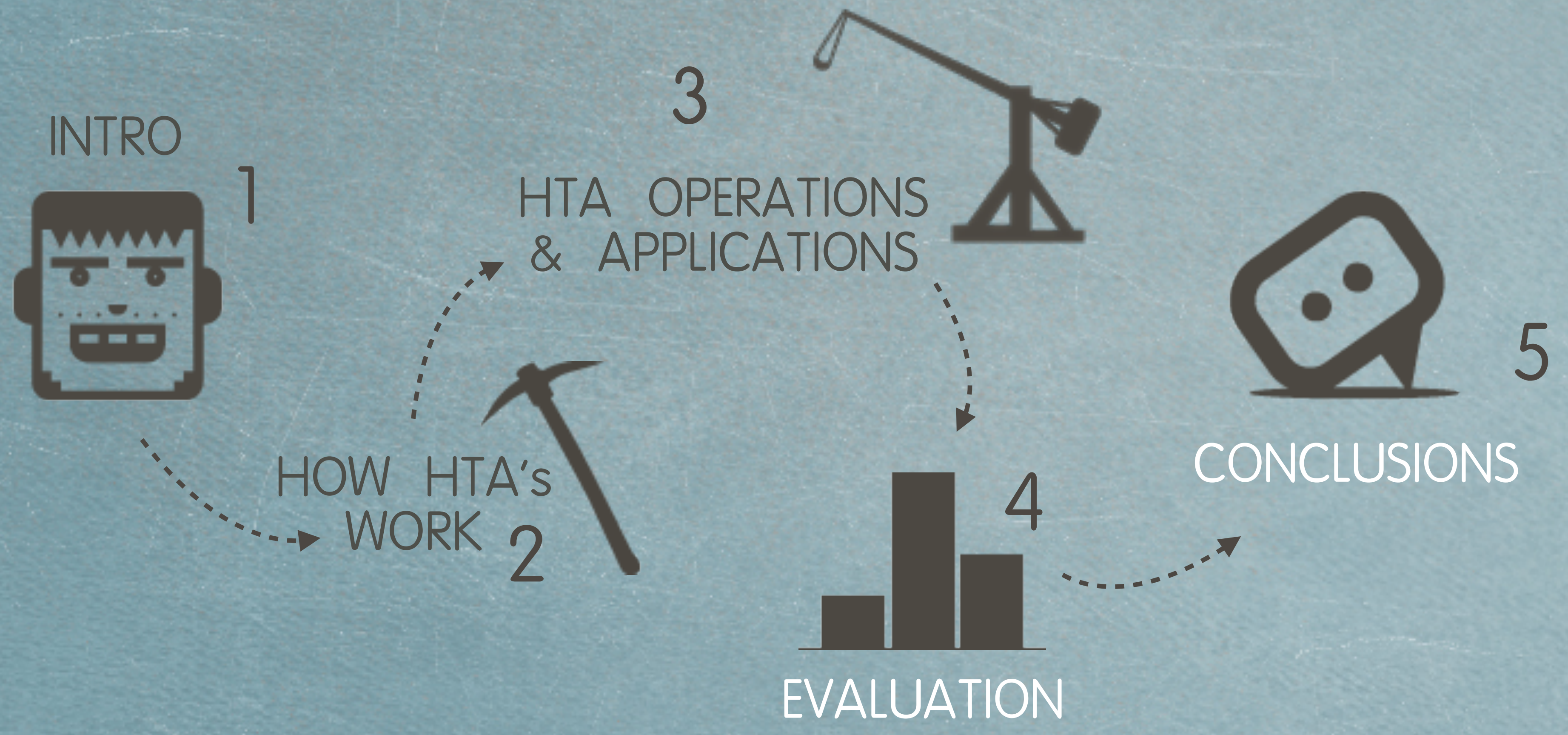
k=2

```
for k=1:m
  C = C + A * B;
  A = circshift(A, [0, -1]);
  B = circshift(B, [-1, 0]);
end
```

Iteration



# TALK OVERVIEW



# NASA ADVANCED SUPERCOMPUTING BENCHMARK

| Nprocs | EP (CLASS C) | | FT (CLASS B) | | CG (CLASS C) | | MG (CLASS B) | | LU (CLASS B) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Fortran+ MPI | Matlab + HTA | Fortran + MPI | Matlab + HTA | Fortran + MPI | Matlab + HTA | Fortran + MPI | Matlab + HTA | Fortran + MPI | Matlab + HTA |
| 1 | 901.6 | 3556.9 | 136.8 | 657.4 | 3606.9 | 3812.0 | 26.9 | 828.0 | 15.7 | 245.1 |
| 4 | 273.1 | 888.8 | 109.1 | 274.0 | 362.0 | 1750.9 | 17.0 | 273.8 | 6.3 | 60.5 |
| 8 | 136.3 | 447.0 | 65.5 | 159.3 | 123.4 | 823.6 | 9.6 | 151.3 | 2.9 | 29.9 |
| 16 | 68.6 | 224.8 | 37.2 | 87.2 | 89.5 | 375.2 | 4.8 | 87.0 | 1.2 | 16.0 |
| 32 | 34.7 | 112.0 | 20.7 | 42.9 | 48.4 | 250.3 | 3.3 | 54.9 | 1.1 | 9.8 |
| 64 | 17.1 | 56.7 | 10.4 | 24.0 | 44.5 | 148.0 | 1.6 | 50.4 | 1.3 | 7.1 |
| 128 | 8.5 | 29.1 | 5.9 | 15.6 | 30.8 | 123.0 | 1.4 | 38.5 | 1.6 | N/A |

image source: paper

# NASA ADVANCED SUPERCOMPUTING BENCHMARK

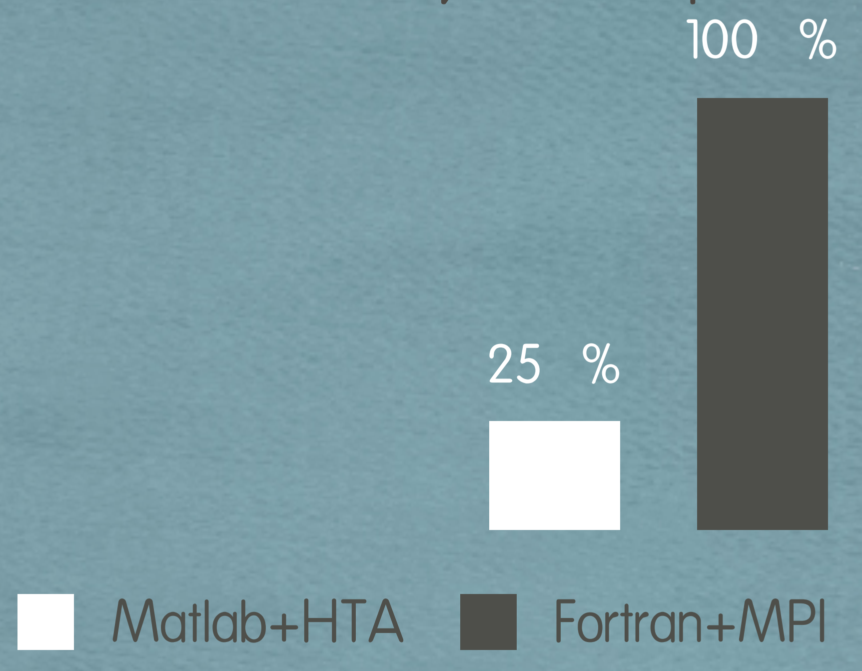
| Nprocs | EP (CLASS C) | | FT (CLASS B) | | CG (CLASS C) | | MG (CLASS B) | | LU (CLASS B) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Fortran+ MPI | Matlab + HTA | Fortran + MPI | Matlab + HTA | Fortran + MPI | Matlab + HTA | Fortran + MPI | Matlab + HTA | Fortran + MPI | Matlab + HTA |
| 1 | 901.6 | 3556.9 | 136.8 | 657.4 | 3606.9 | 3812.0 | 26.9 | 828.0 | 15.7 | 245.1 |
| 4 | 273.1 | 888.8 | 109.1 | 274.0 | 362.0 | 1750.9 | 17.0 | 273.8 | 6.3 | 60.5 |
| 8 | 136.3 | 447.0 | 65.5 | 159.3 | 123.4 | 823.6 | 9.6 | 151.3 | 2.9 | 29.9 |
| 16 | 68.6 | 224.8 | 37.2 | 87.2 | 89.5 | 375.2 | 4.8 | 87.0 | 1.2 | 16.0 |
| 32 | 34.7 | 112.0 | 20.7 | 42.9 | 48.4 | 250.3 | 3.3 | 54.9 | 1.1 | 9.8 |
| 64 | 17.1 | 56.7 | 10.4 | 24.0 | 44.5 | 148.0 | 1.6 | 50.4 | 1.3 | 7.1 |
| 128 | 8.5 | 29.1 | 5.9 | 15.6 | 30.8 | 123.0 | 1.4 | 38.5 | 1.6 | N/A |

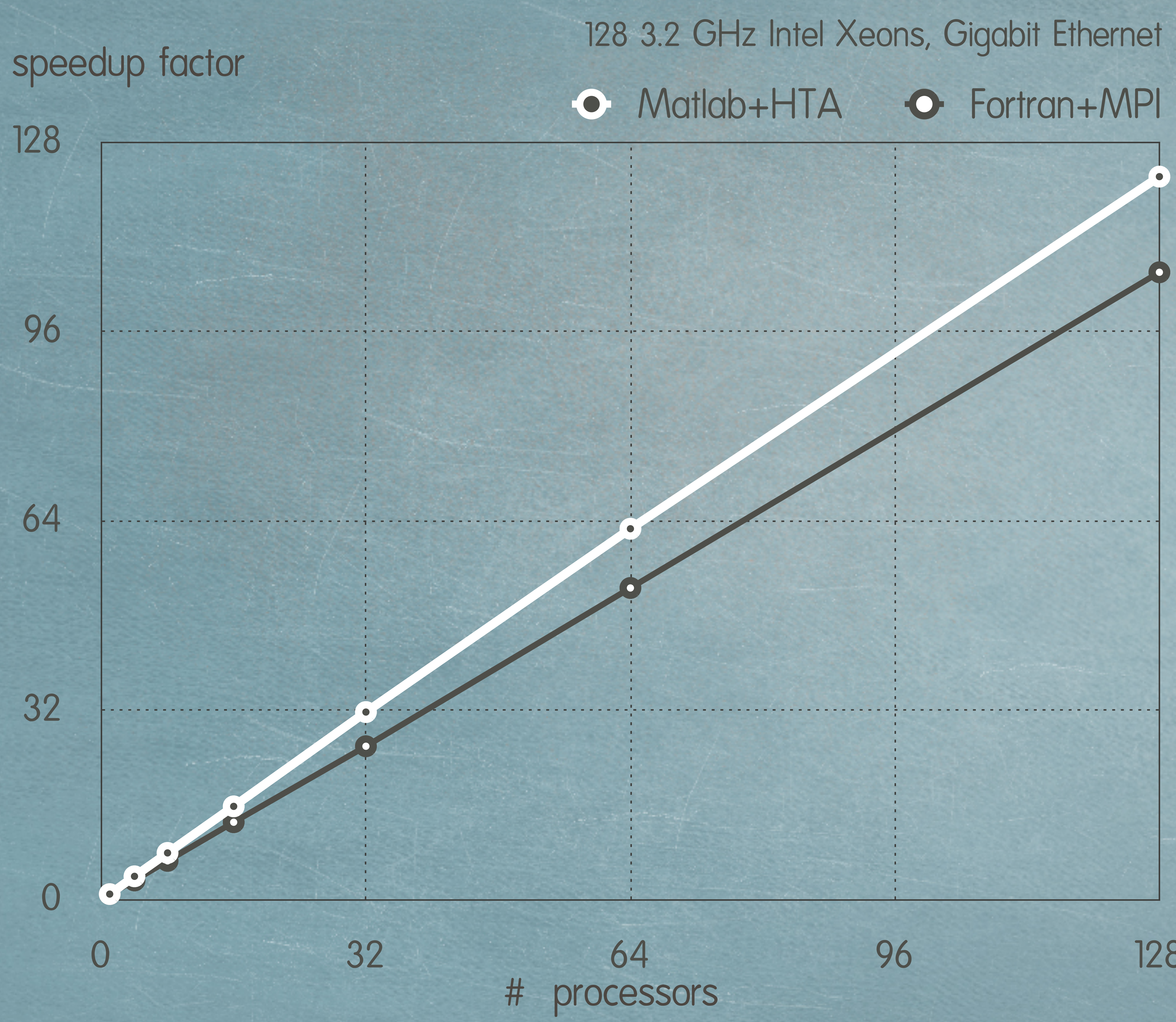
image source: paper


4
EVALUATION

## ebarassingly parallel EP

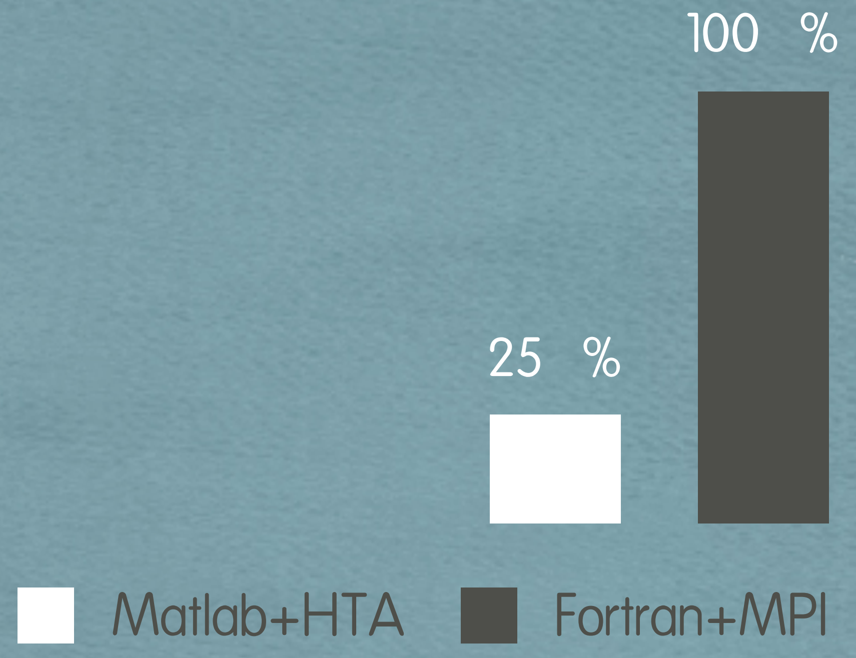
sequential speed

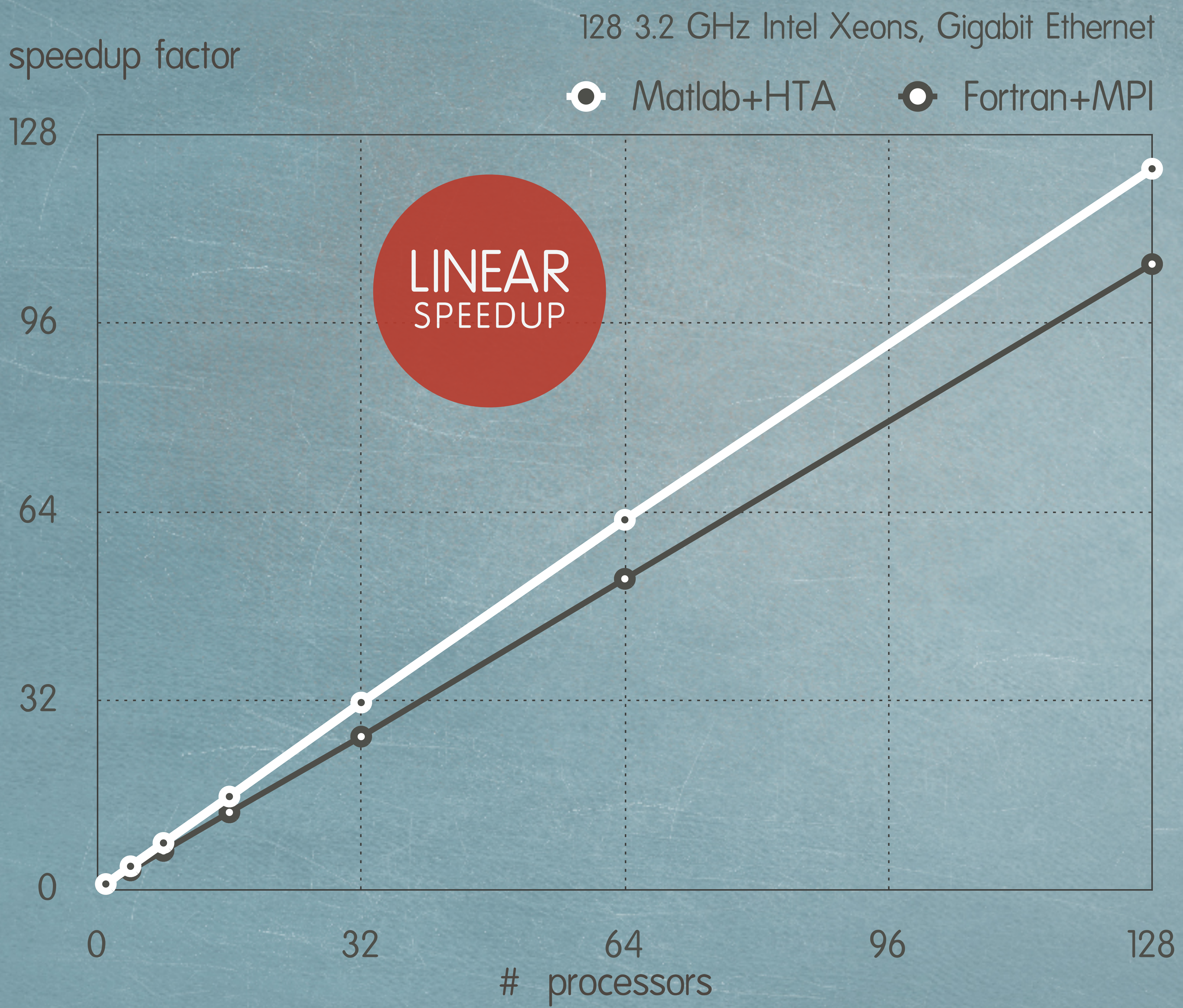


ebarassingly parallel **EP**

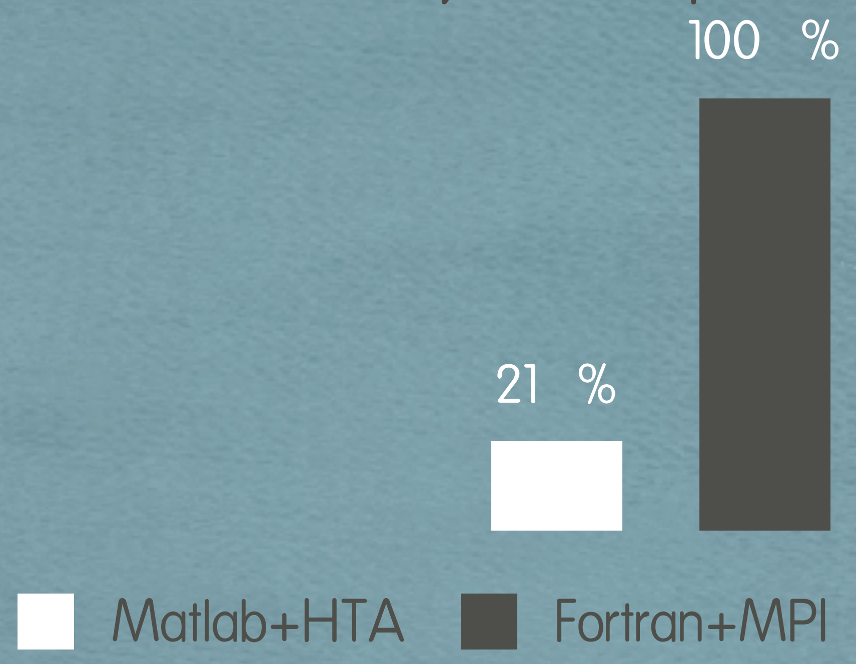
sequential speed

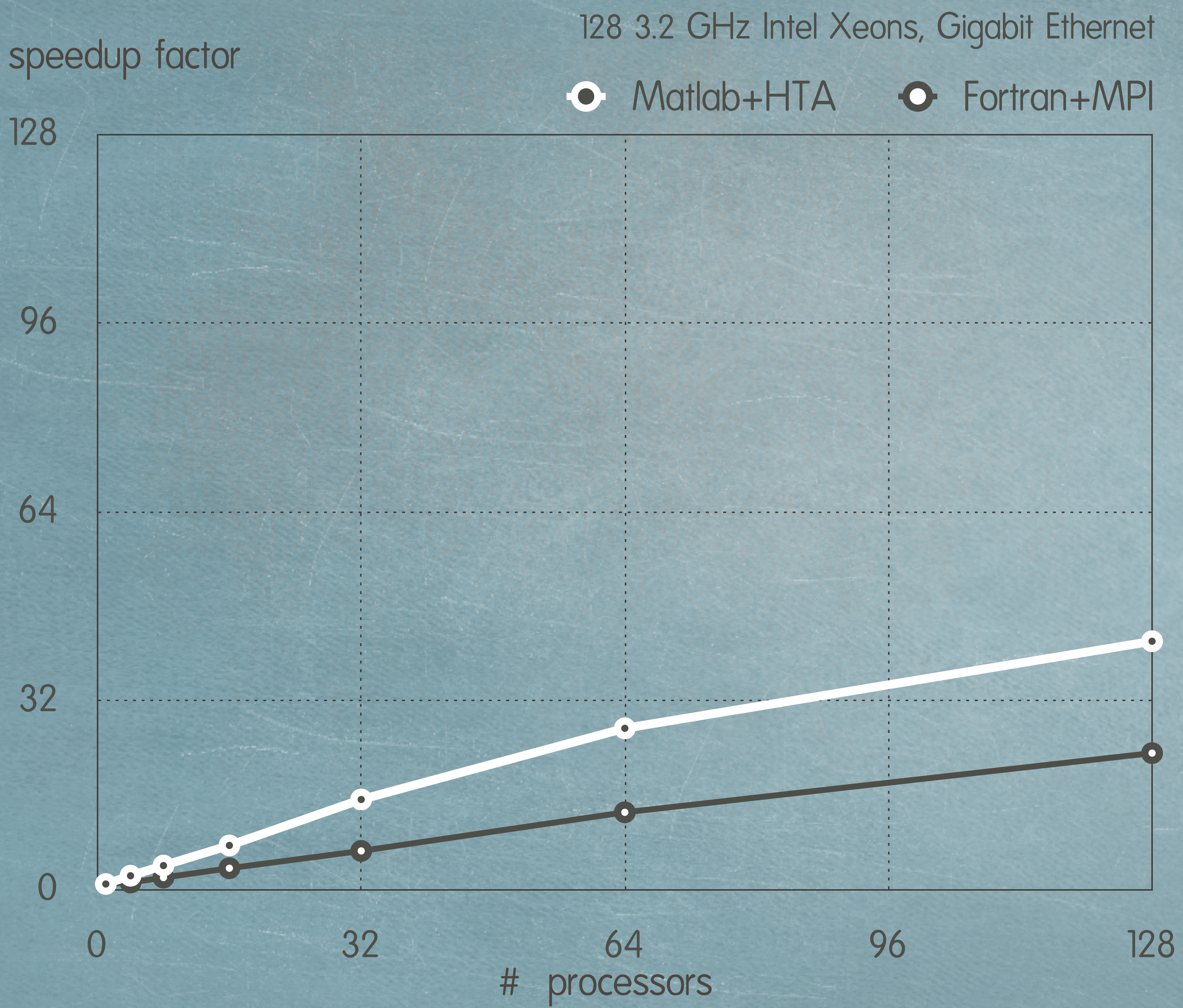


# fast fourier transform FFT

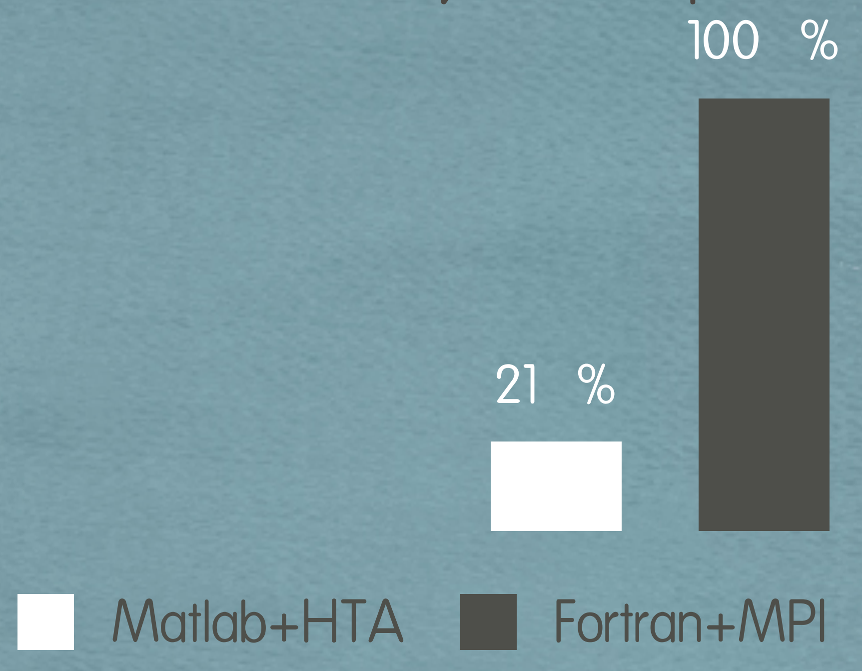
## sequential speed

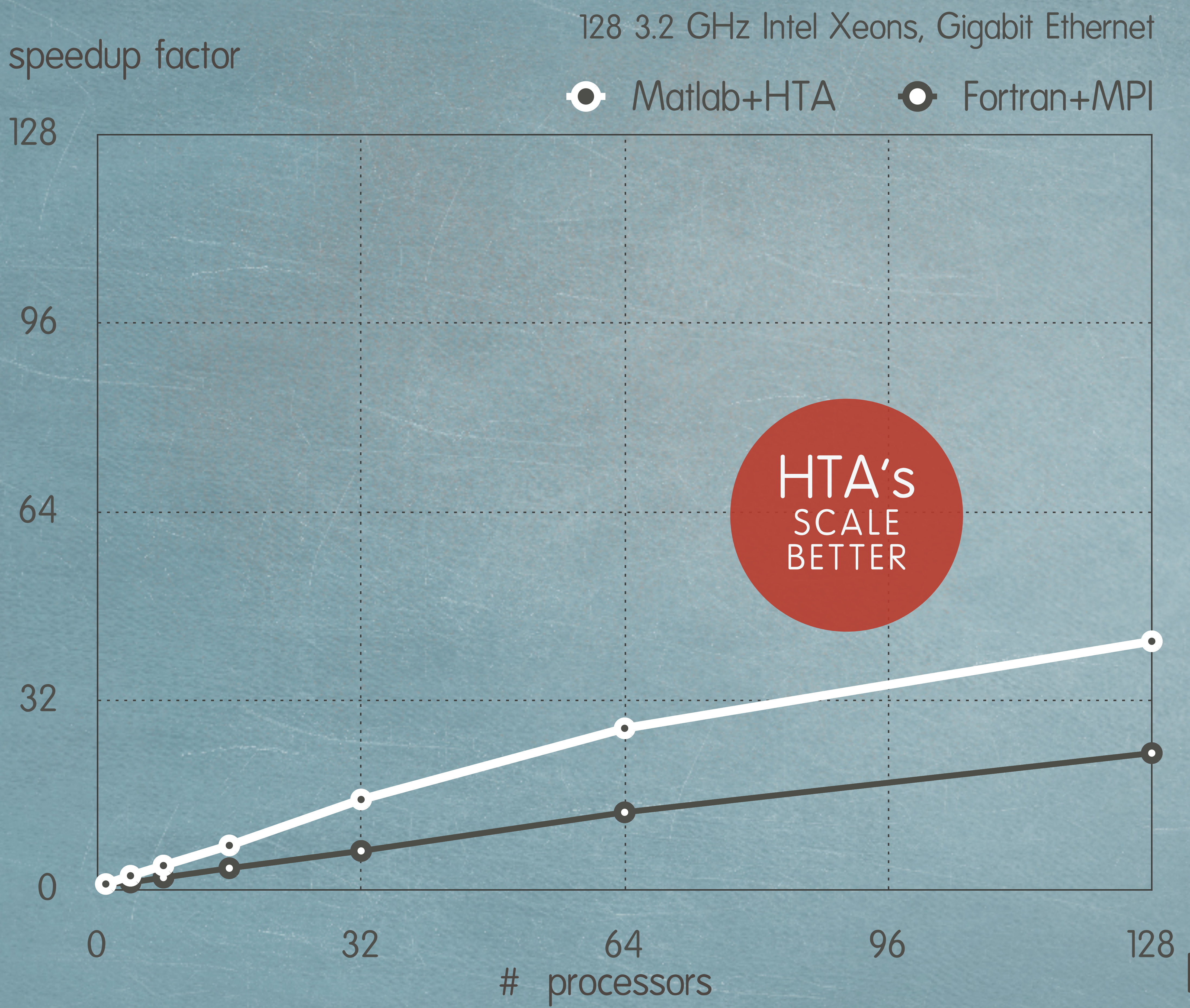


fast fourier transform FFT

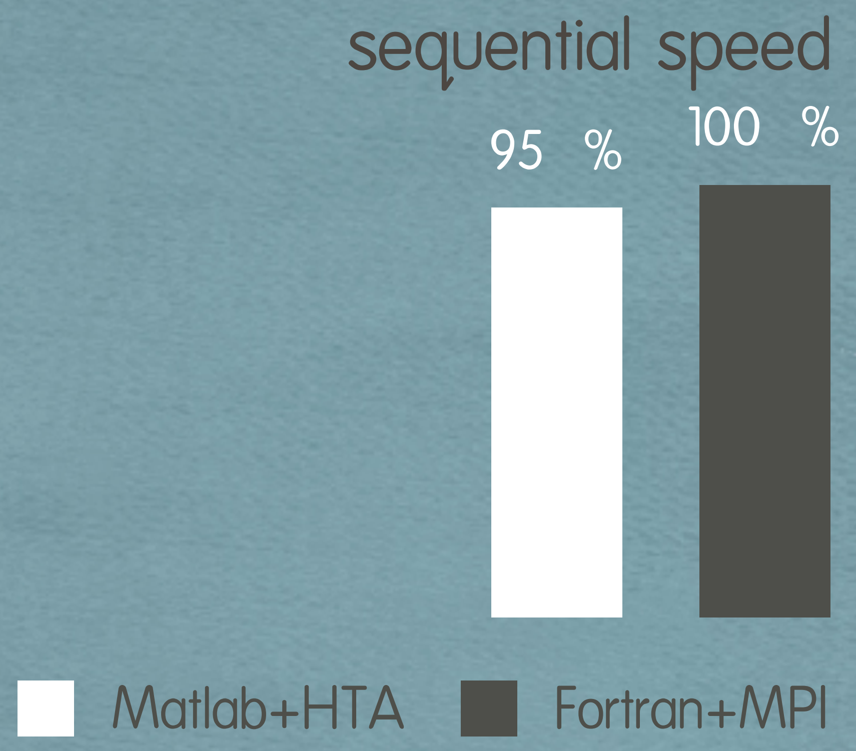
sequential speed

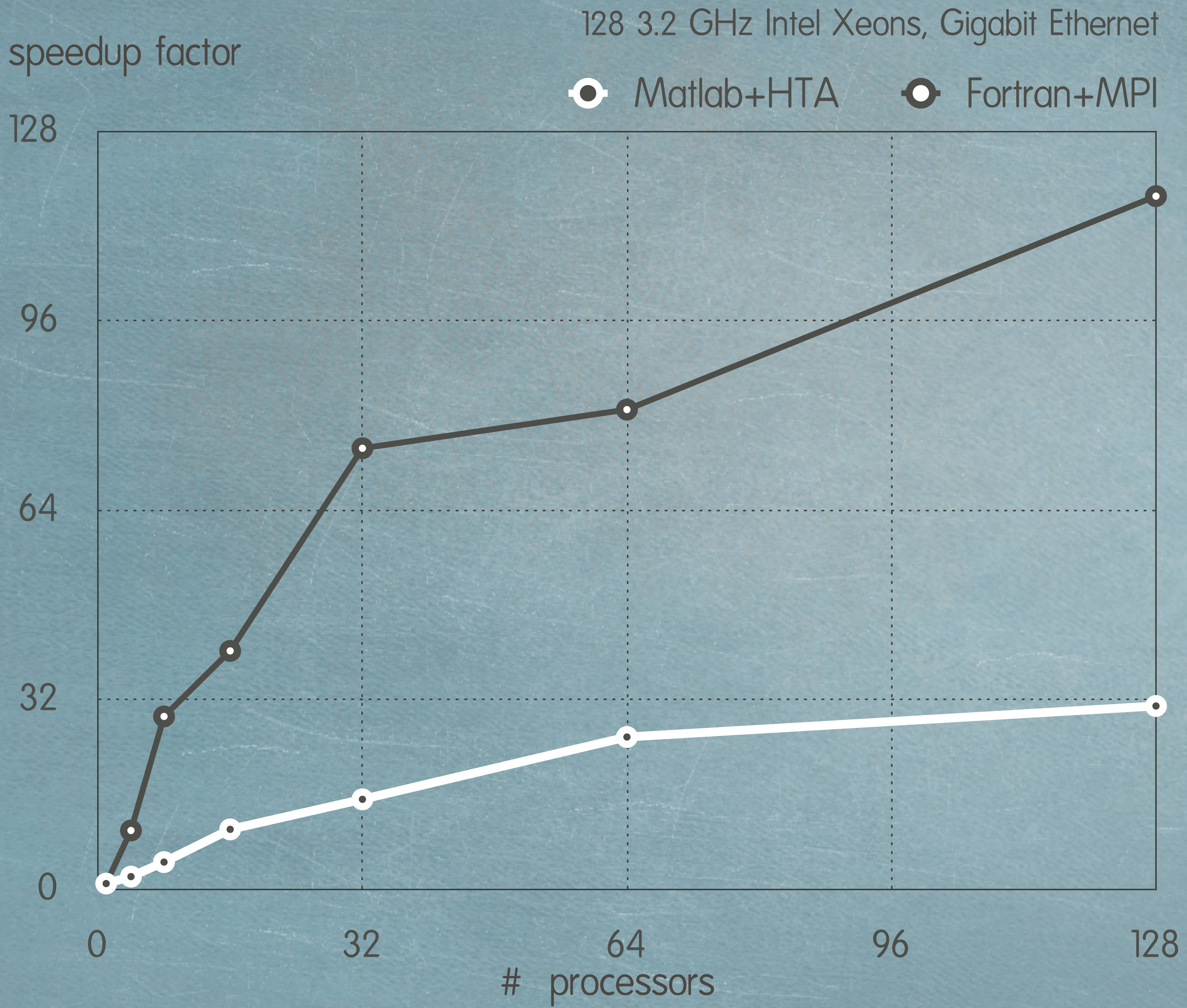



4
EVALUATION

# conjugate gradient CG

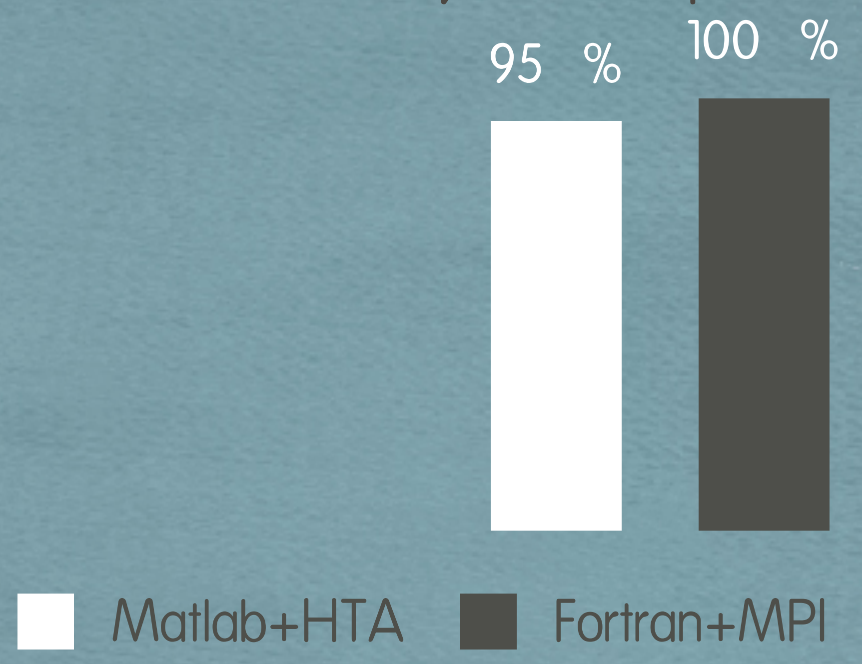


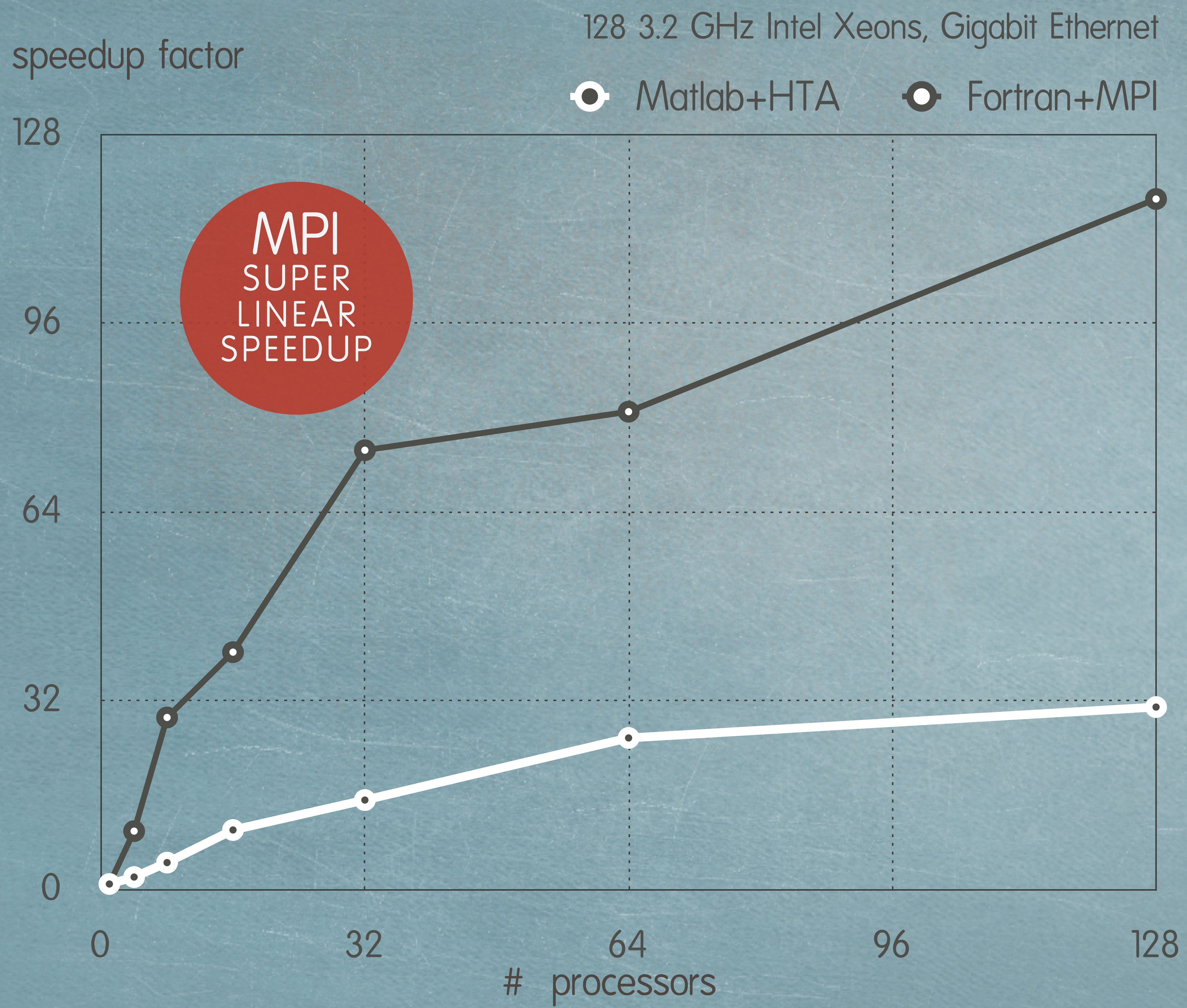




4
EVALUATION

# conjugate gradient CG

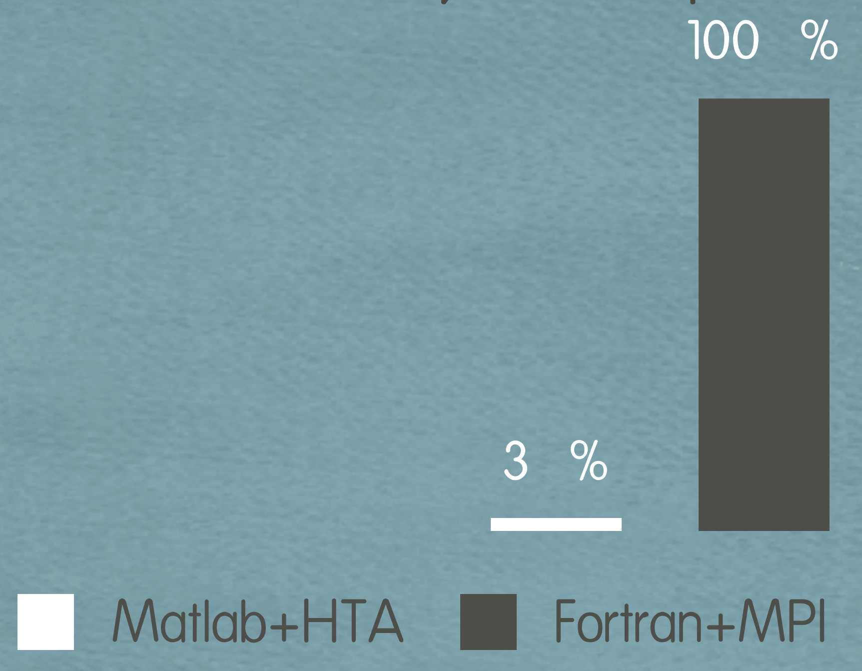
sequential speed



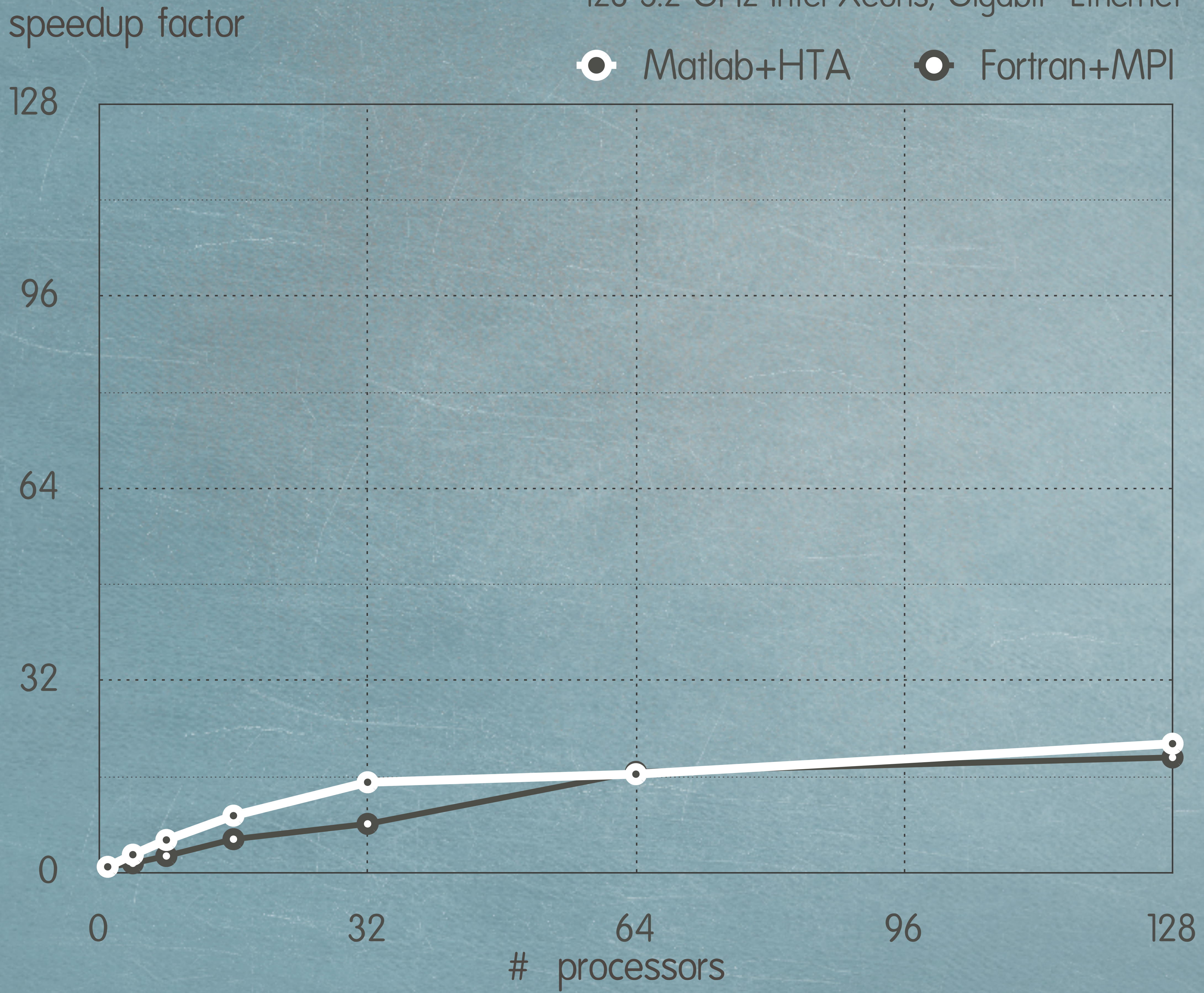



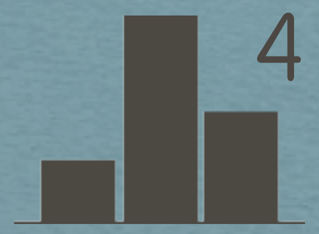
# multi grid MG

sequential speed

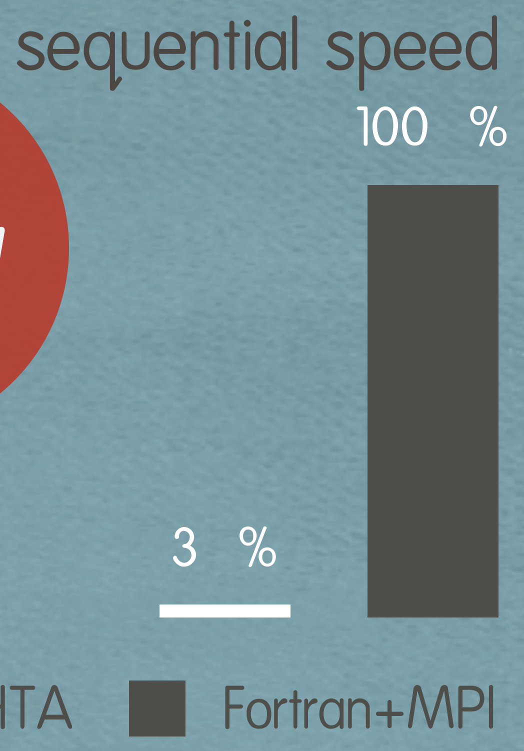



128 3.2 GHz Intel Xeons, Gigabit Ethernet



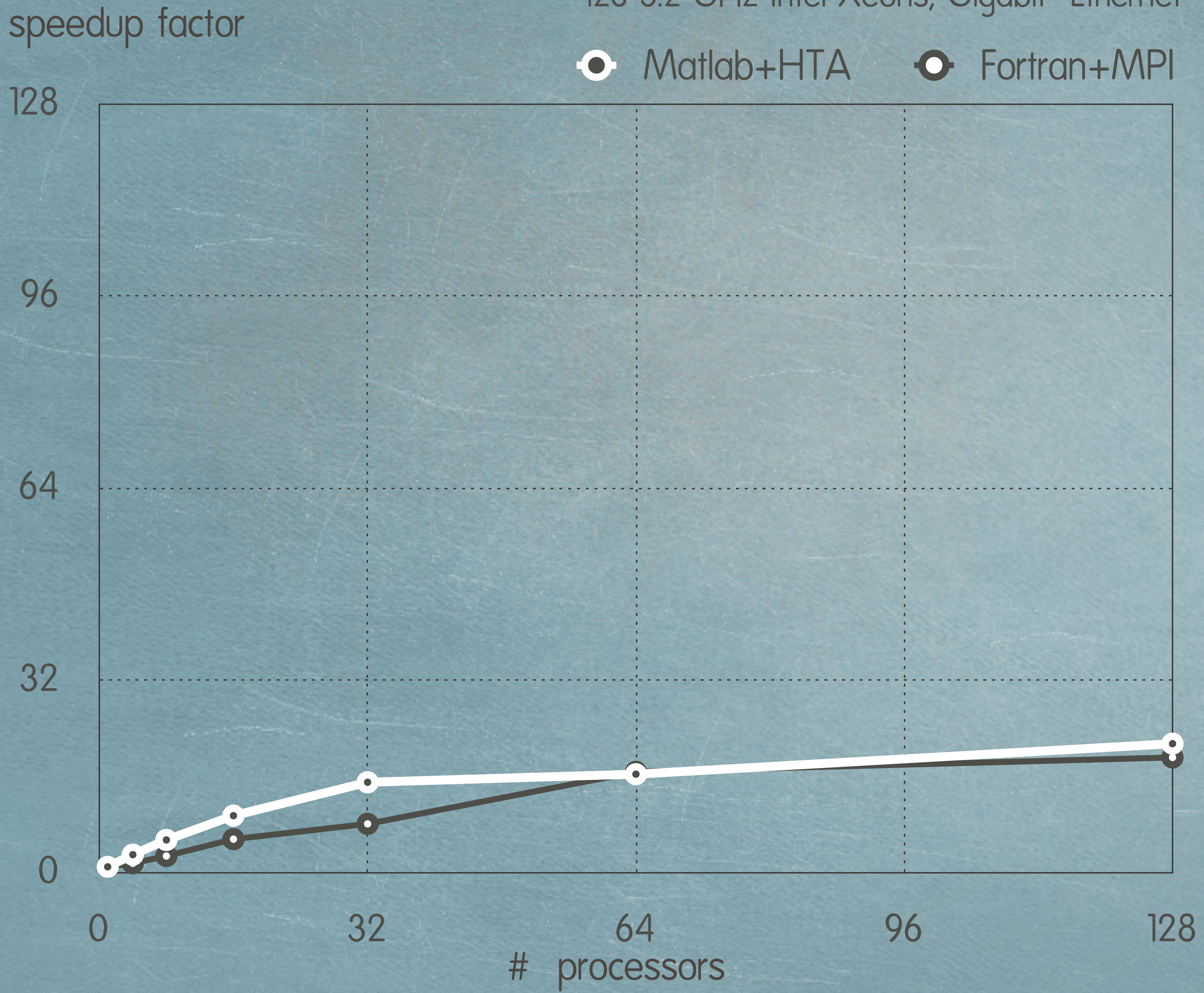



multi grid **MG**



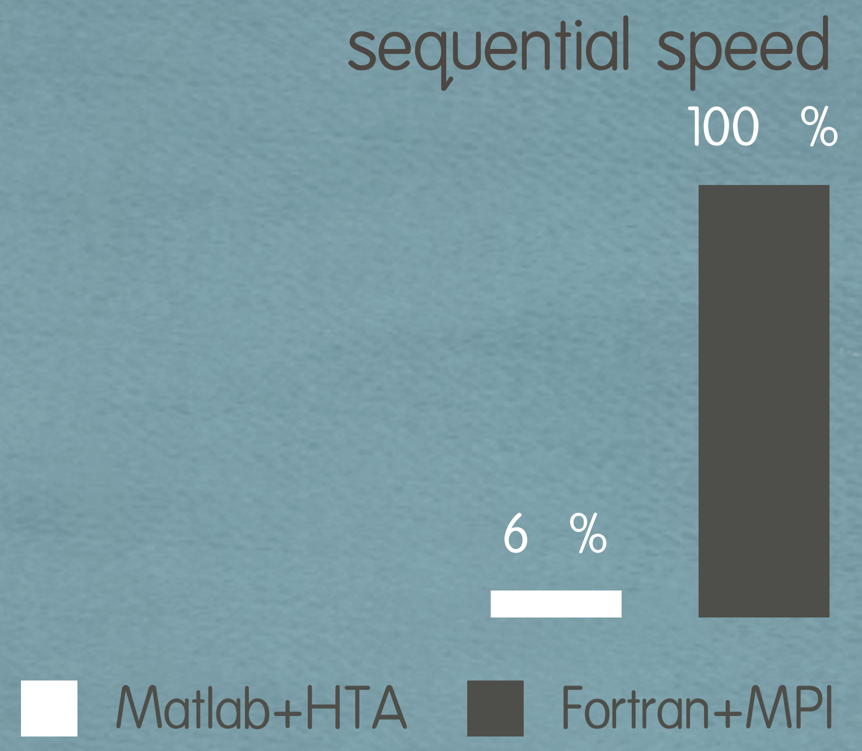

HTA's
SLOW

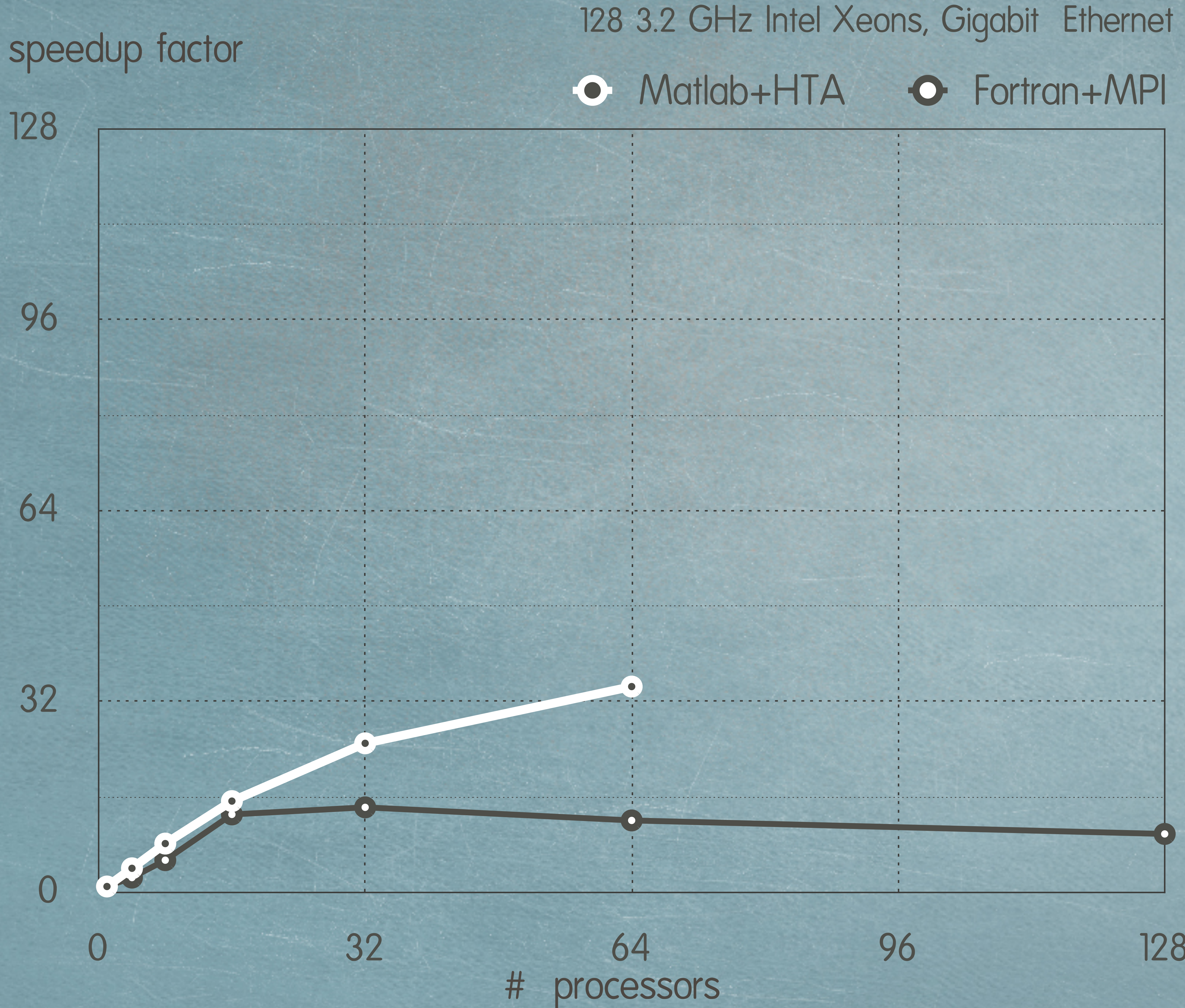
128 3.2 GHz Intel Xeons, Gigabit Ethernet




EVALUATION 4

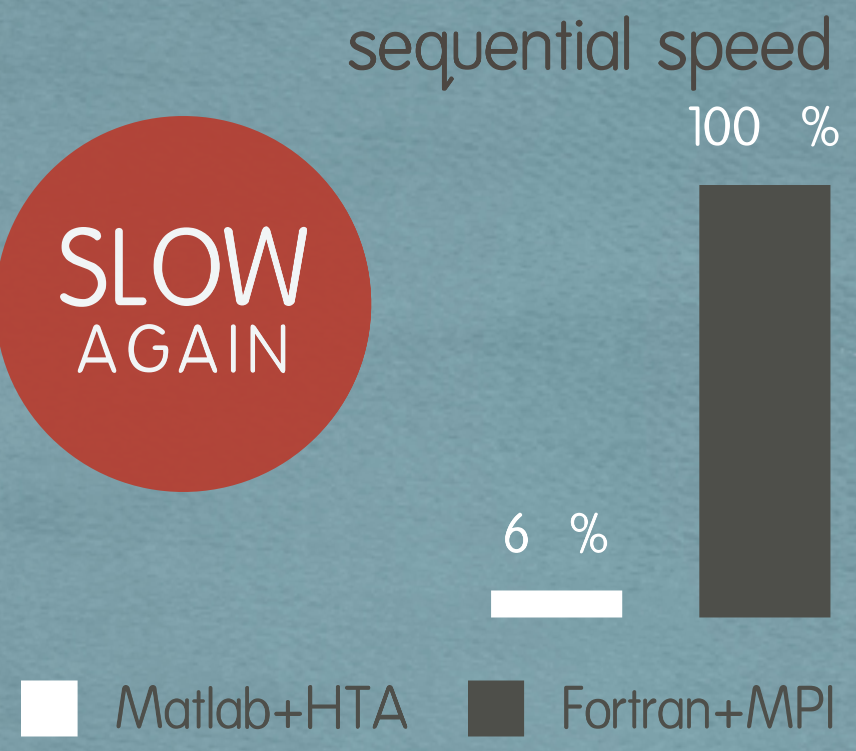
# lu factorization LU

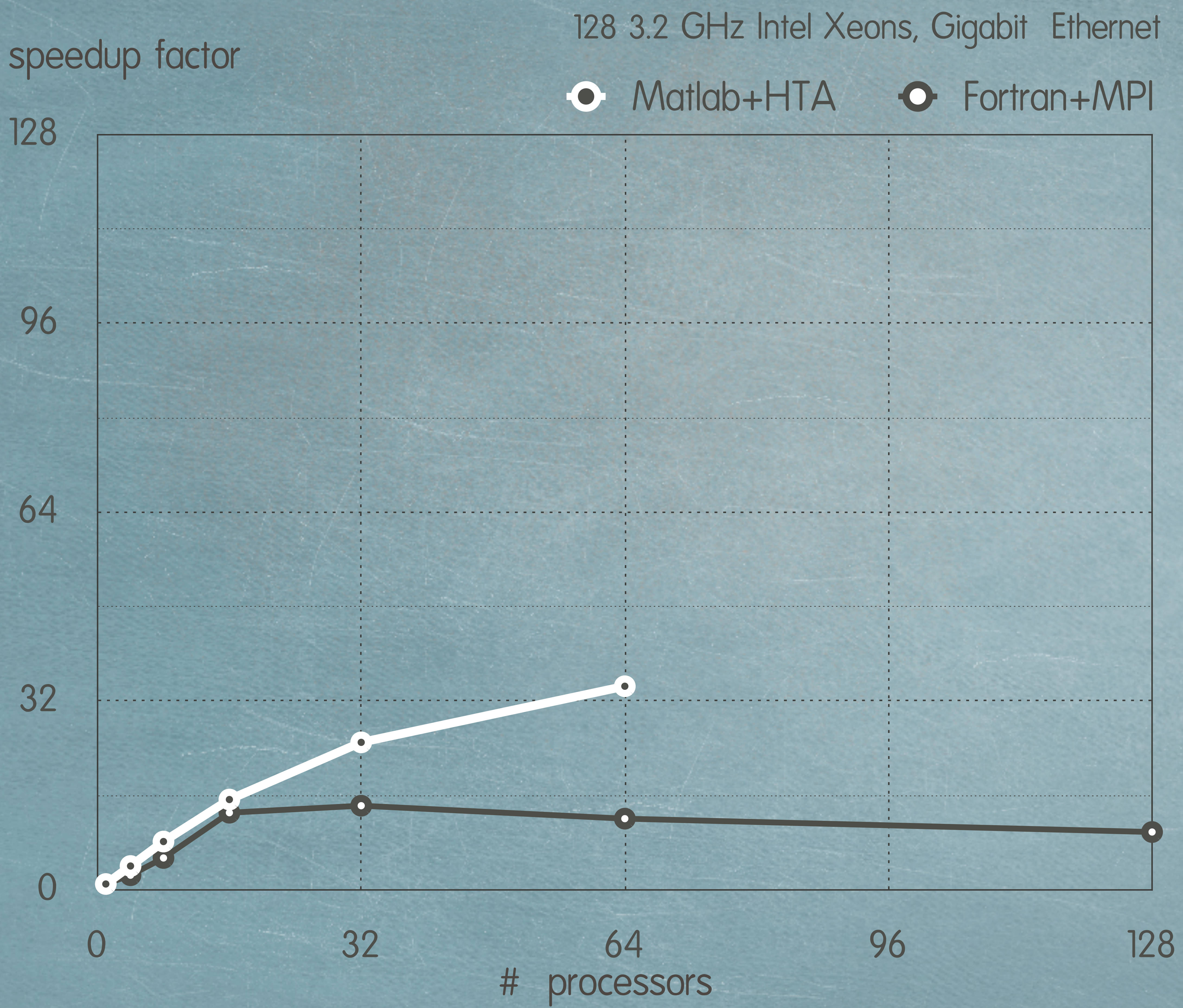


# lu factorization LU

SLOW AGAIN

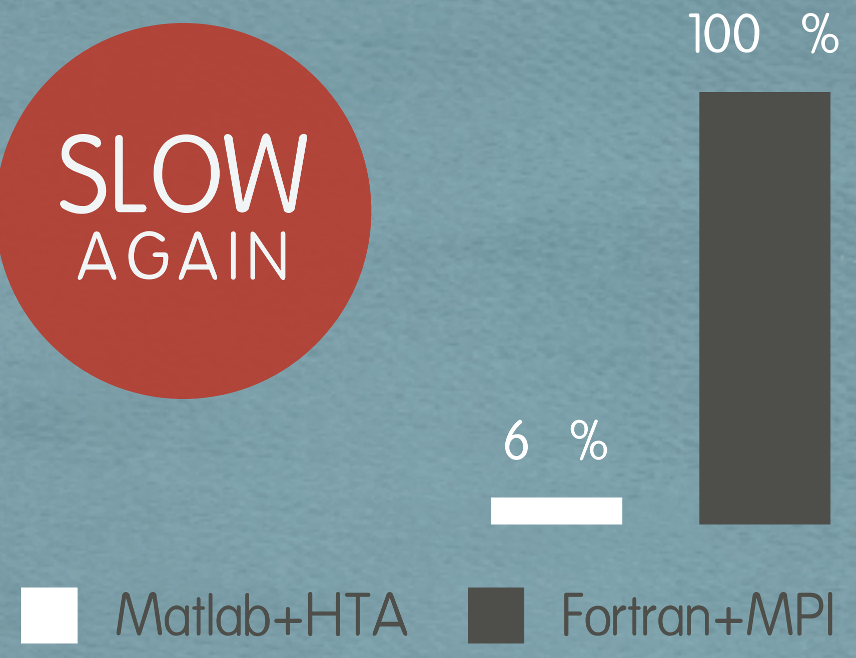


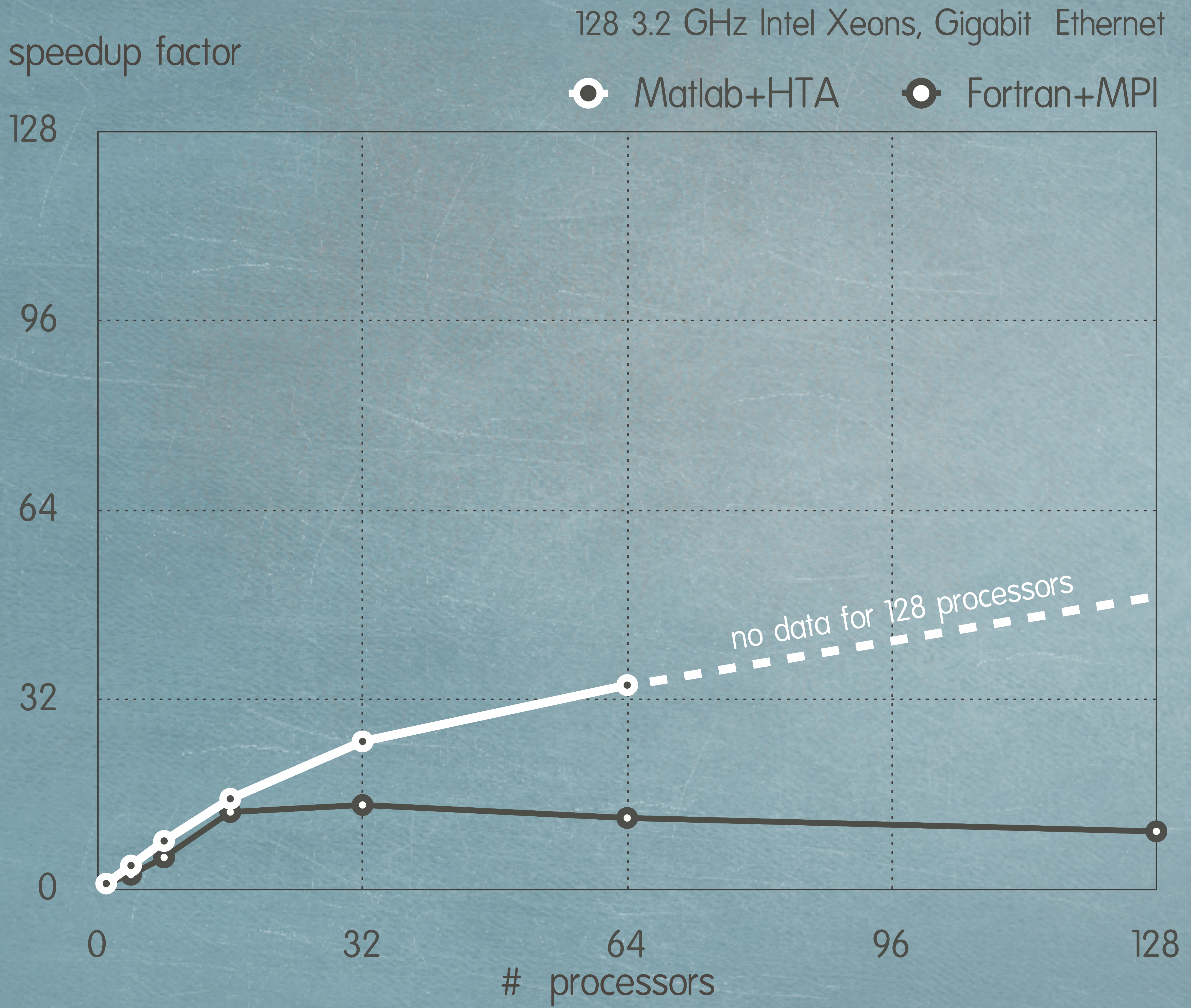

EVALUATION 4

# lu factorization LU

sequential speed

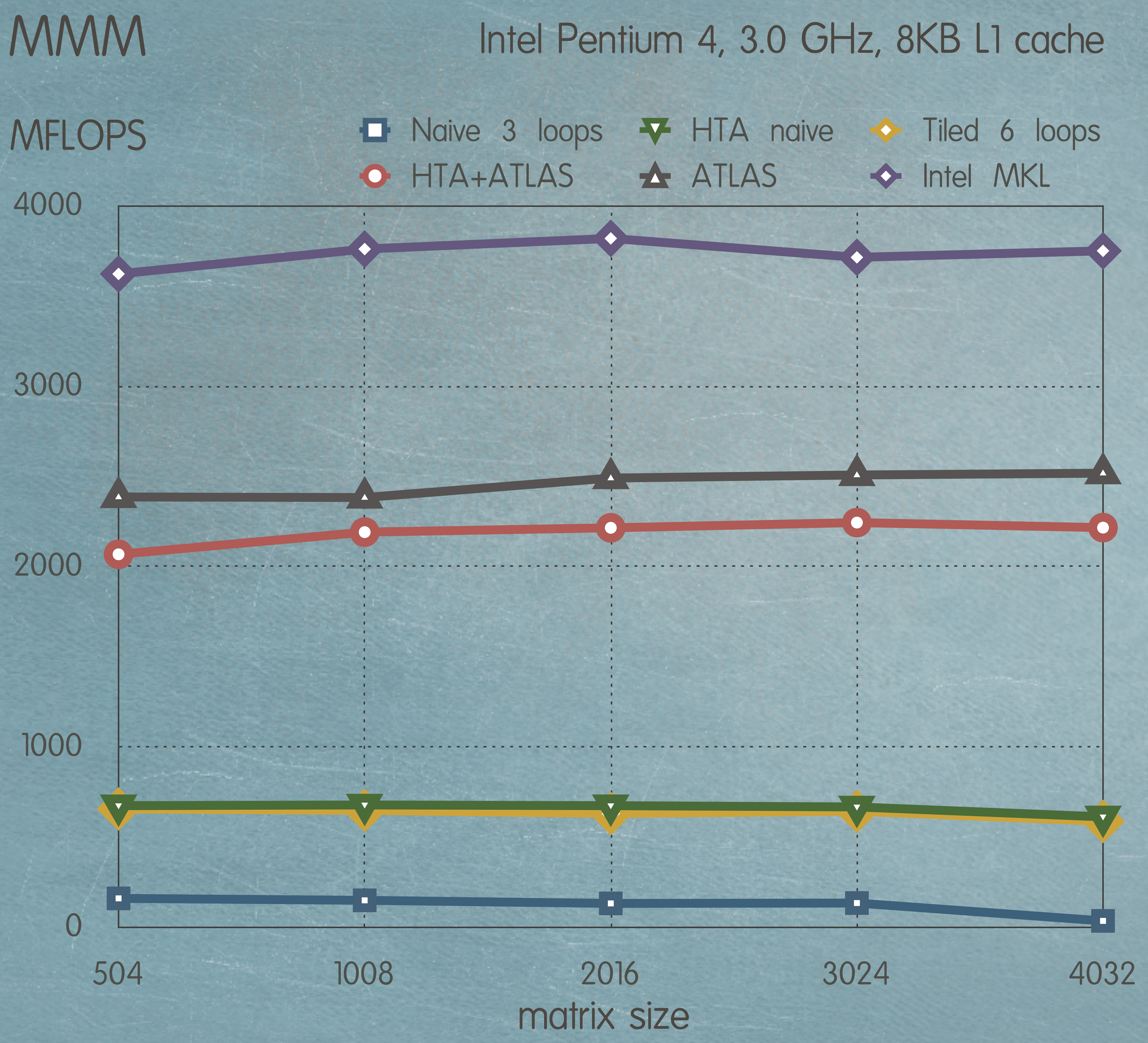



SLOW AGAIN

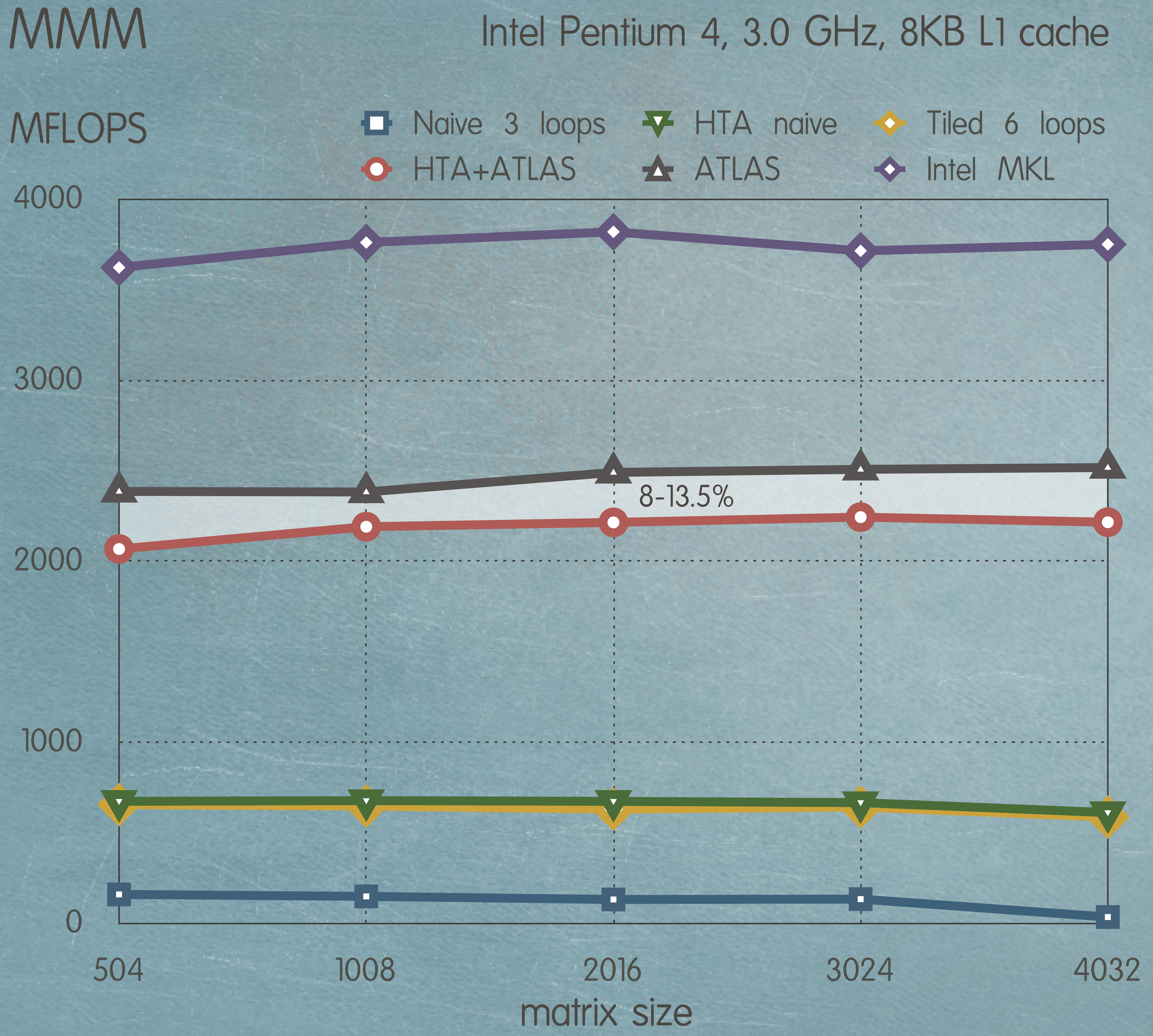



EVALUATION 4

# PERFORMANCE OF C++ HTA's




# PERFORMANCE OF C++ HTA's




EVALUATION 4

# LINES OF CODE COMPARISON

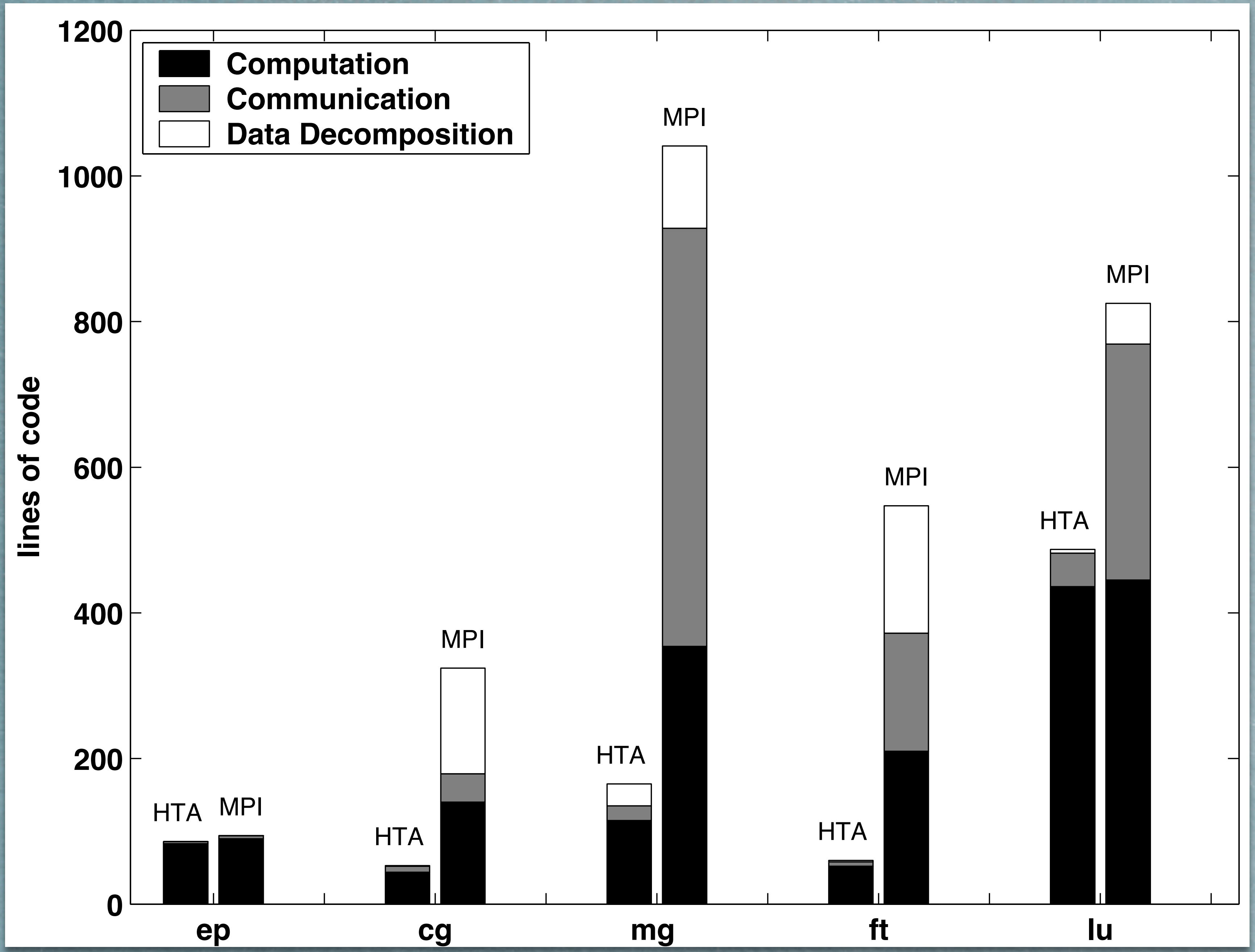
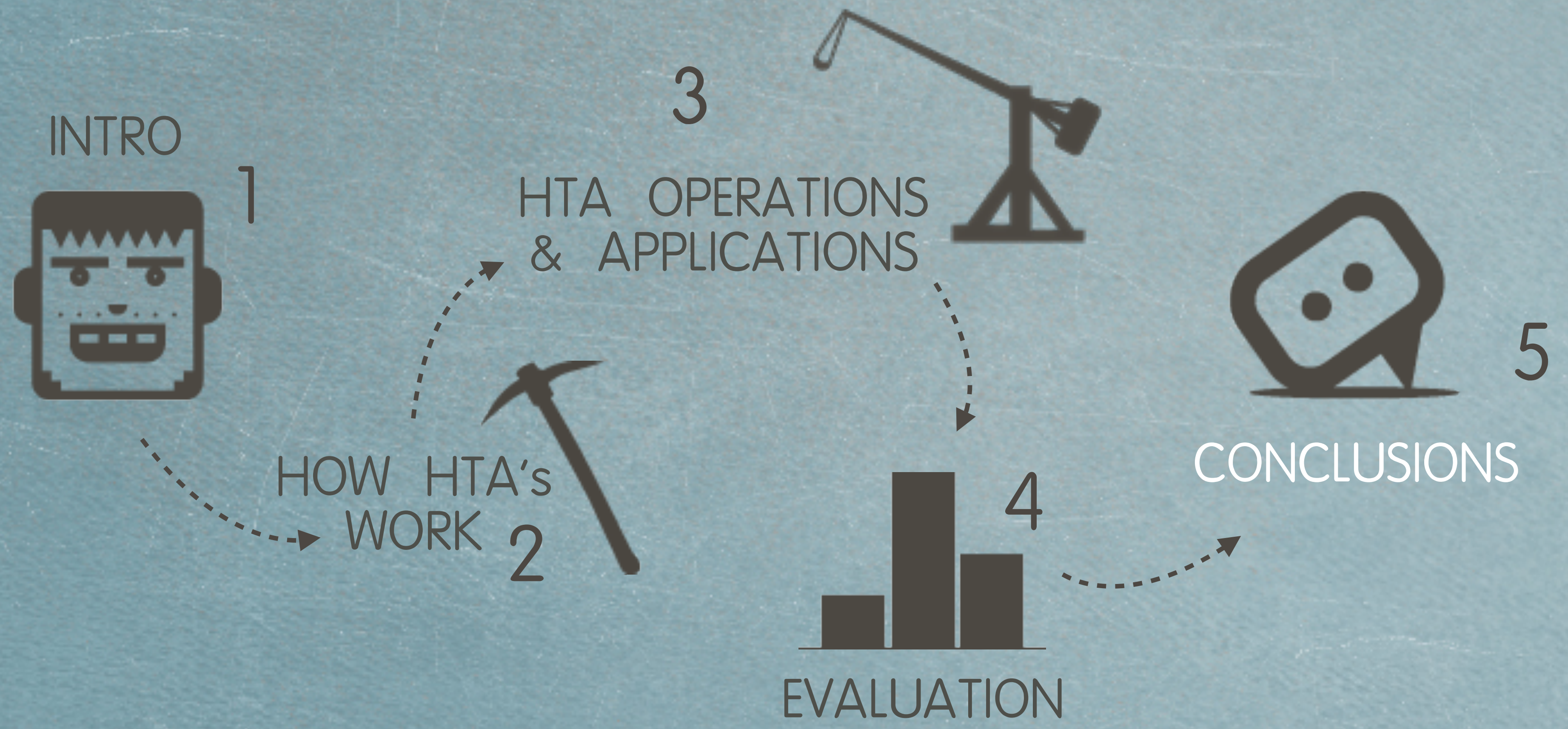


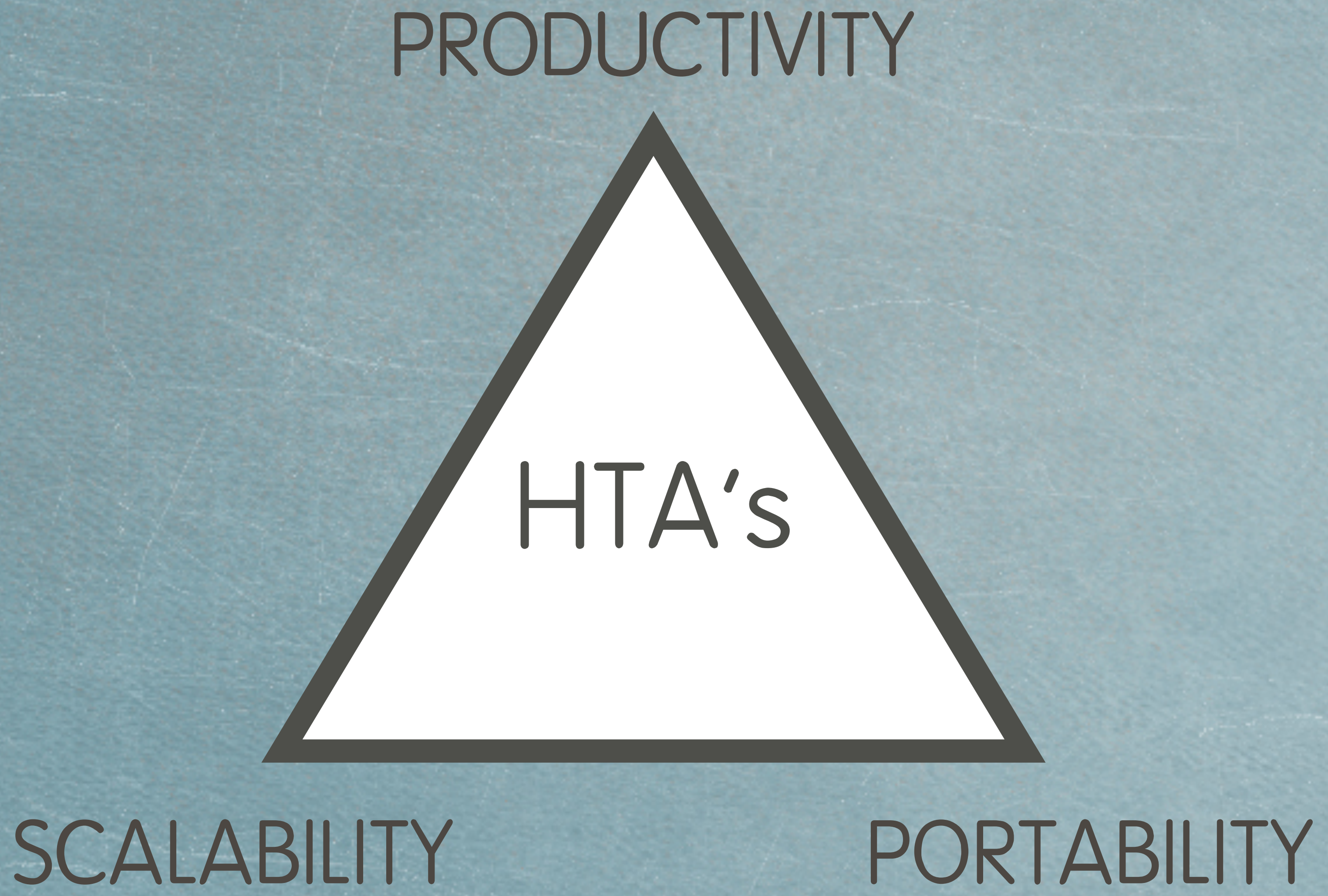

image source: paper

EVALUATION 4

# TALK OVERVIEW



# CONCLUSIONS



# FURTHER INFORMATION

http://polaris.cs.uiuc.edu/hta/

# THANKS.

FOR YOUR ATTENTION

# Q&A

PUT YOUR QUESTIONS