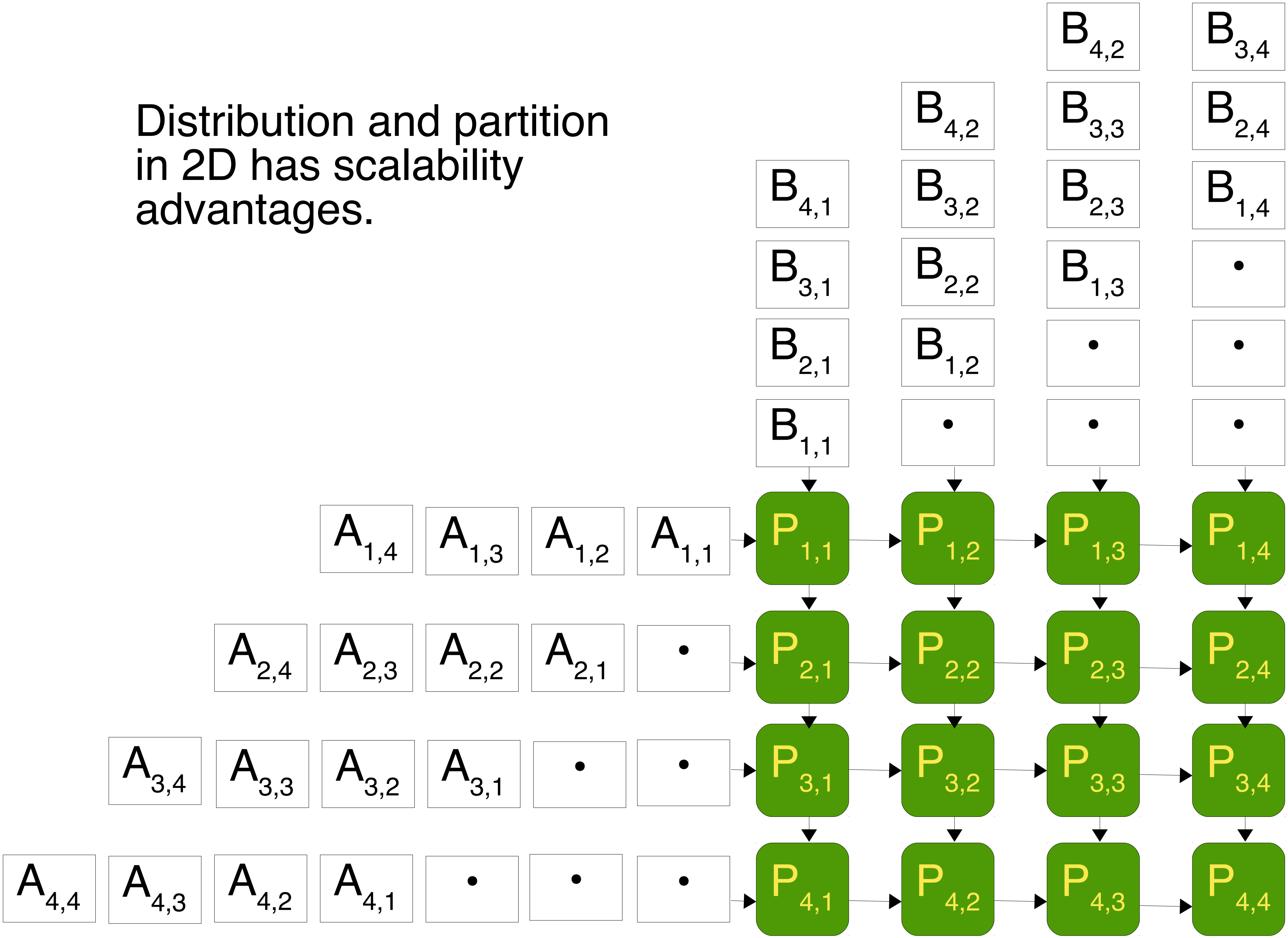
COSC 462

Solving Linear Systems

Piotr Luszczek

Cannon's (Systolic) Algorithm Recap

Distribution and partition in 2D has scalability advantages.



Applications Using Linear Solve

- Structural analysis (civil engineering)
- Heat transport (mechanical engineering)
- Analysis of power grids (electrical engineering)
- Production planning (economics)
- Regression analysis (statistics)
- Antenna/radar/stealth fighter design(electromagnetics)
- Plasma containment (physics)
- Benchmarking (TOP500, HPL)

Gaussian Elimination Example

$+x$	$-3y$	$+z$	$=$	$+4$
$+2x$	$-8y$	$+8z$	$=$	-2
$-6x$	$+3y$	$-15z$	$=$	9

*(-2)

*6

Could divide by 2 to get row values closer to 1

Could divide by 3 to get row values closer to 1

$+x$	$-3y$	$+z$	$=$	$+4$
$0x$	$-2y$	$+6z$	$=$	-10
$0x$	$-15y$	$-9z$	$=$	33

/2

$+x$	$-3y$	$+z$	$=$	$+4$
$0x$	$-y$	$+3z$	$=$	-5
$0x$	$-5y$	$-3z$	$=$	11

*5

$+x$	$-3y$	$+z$	$=$	$+4$
$0x$	$-y$	$+3z$	$=$	-5
$0x$	$0y$	$-18z$	$=$	36

$x = 3$

$y = -1$

$z = -2$

Back-substitution

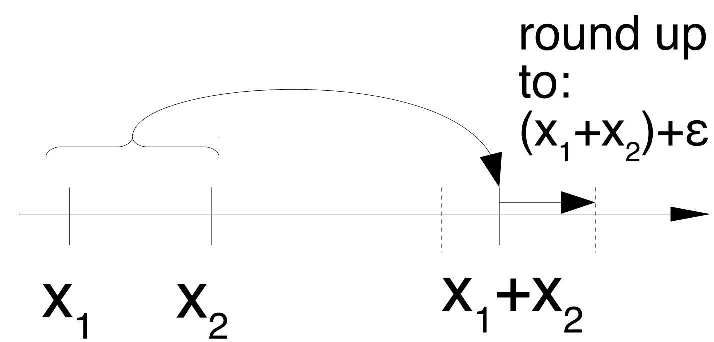
Reference Implementation

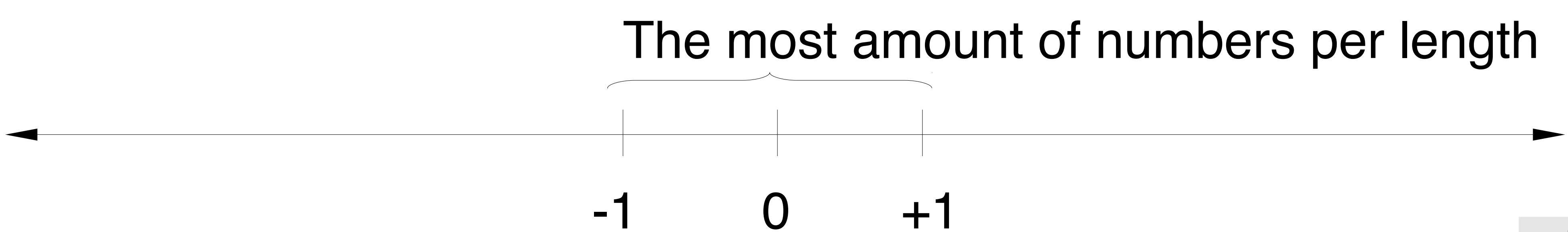
- Ax = b
  - A is N by N matrix
  - x, b are N by 1 vectors
- ```
  for (i = 0; i < N; ++i)
    pivot = A[max_loc( abs(A[:][i]) )][j]
    A[i+1:N][j] /= pivot

    for (j = i+1; j < N; ++j)
      for (k = i+1; j < N; ++j)
        A[j][k] -= A[k][i] * A[i][j]
  ```
- Complexity
  - $^{2}/_{3}N^{3}$
  ```

Floating Point Arithmetic Primer

- Floating point numbers in a computer are store in IEEE 754 standard (1985, 2008)
  - A subset of rational numbers and infinities, NaN's, -0
  - Binary representation is sign, mantissa, exponent
  - Multiple sizes available
    - 16-bits (half precision, storage only)
    - 32-bits (single precision)
    - 64-bits (double precision)
    - 80-bits (extended precision)
    - 128-bits (quad precision)





Row Pivoting for Numerical Stability

$+x$	$-3y$	$+z$	$=$	$+4$
$+2x$	$-8y$	$+8z$	$=$	-2
$-6x$	$+3y$	$-15z$	$=$	9

Row pivoting

$+x$	$-3y$	$+z$	$=$	$+4$
$+2x$	$-8y$	$+8z$	$=$	-2
$-6x$	$+3y$	$-15z$	$=$	9

swap

$-6x$	$+3y$	$-15z$	$=$	$+9$
$+2x$	$-8y$	$+8z$	$=$	-2
$+x$	$-3y$	$+z$	$=$	$+4$

/6

$-x$	$+1/2y$	$-5/2z$	$=$	$+3/2$
$+1/3x$	$-4/3y$	$+3/2z$	$=$	$+1/2$
$+1/6x$	$-1/2y$	$+z/6$	$=$	$+2/3$

\+

$-x$	$+1/2y$	$-5/2z$	$=$	$3/2$
$0x$	$-7/2y$	$+8/6z$	$=$	$-5/6$
$0x$	$-5/2y$	$-3/2z$	$=$	$+11/2$

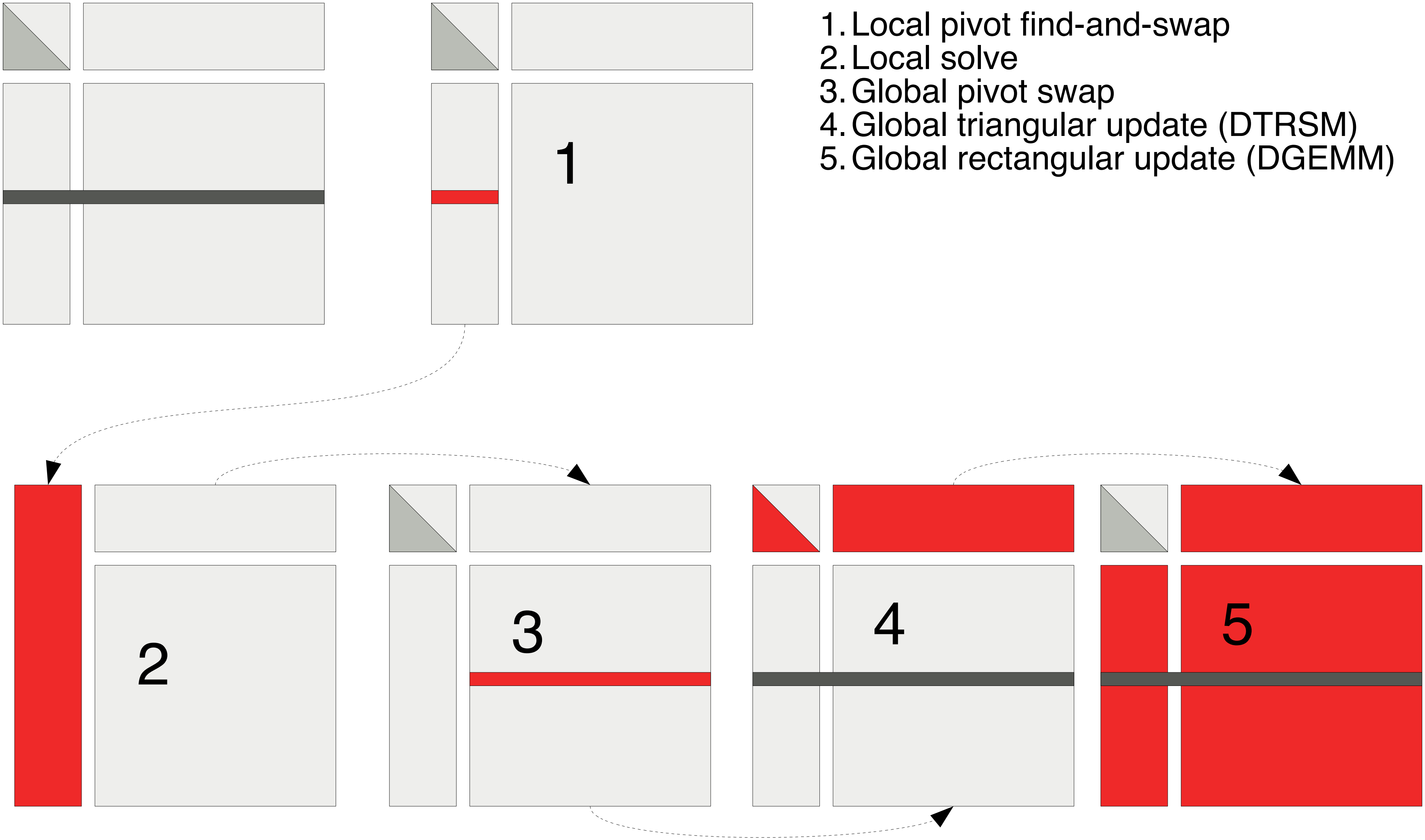
No pivoting

$+x$	$-3y$	$+z$	$=$	$+4$
$0x$	$-2y$	$+6z$	$=$	-10
$0x$	$-15y$	$-9z$	$=$	33

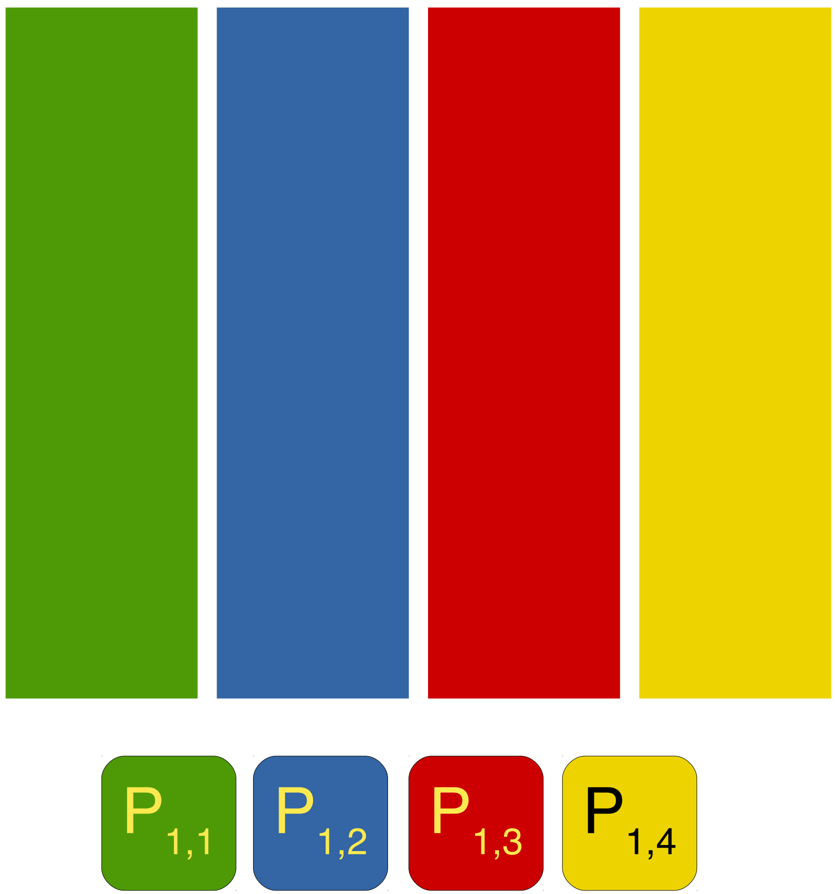
Without proper down-scaling, errors get multiplied and the magnitude of updated entries grows: phenomenon call pivot growth.

Agglomeration: Blocking

1. Local pivot find-and-swap
2. Local solve
3. Global pivot swap
4. Global triangular update (DTRSM)
5. Global rectangular update (DGEMM)

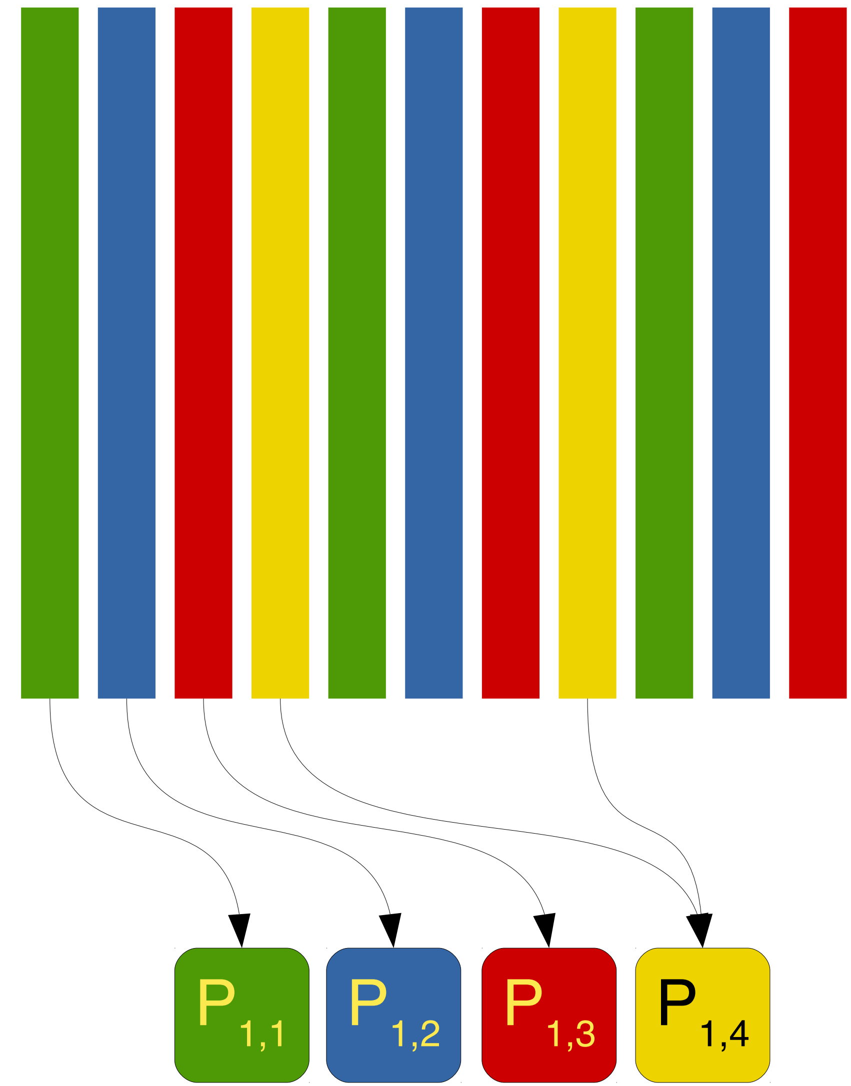


Block Distribution vs. Block Cyclic Distribution



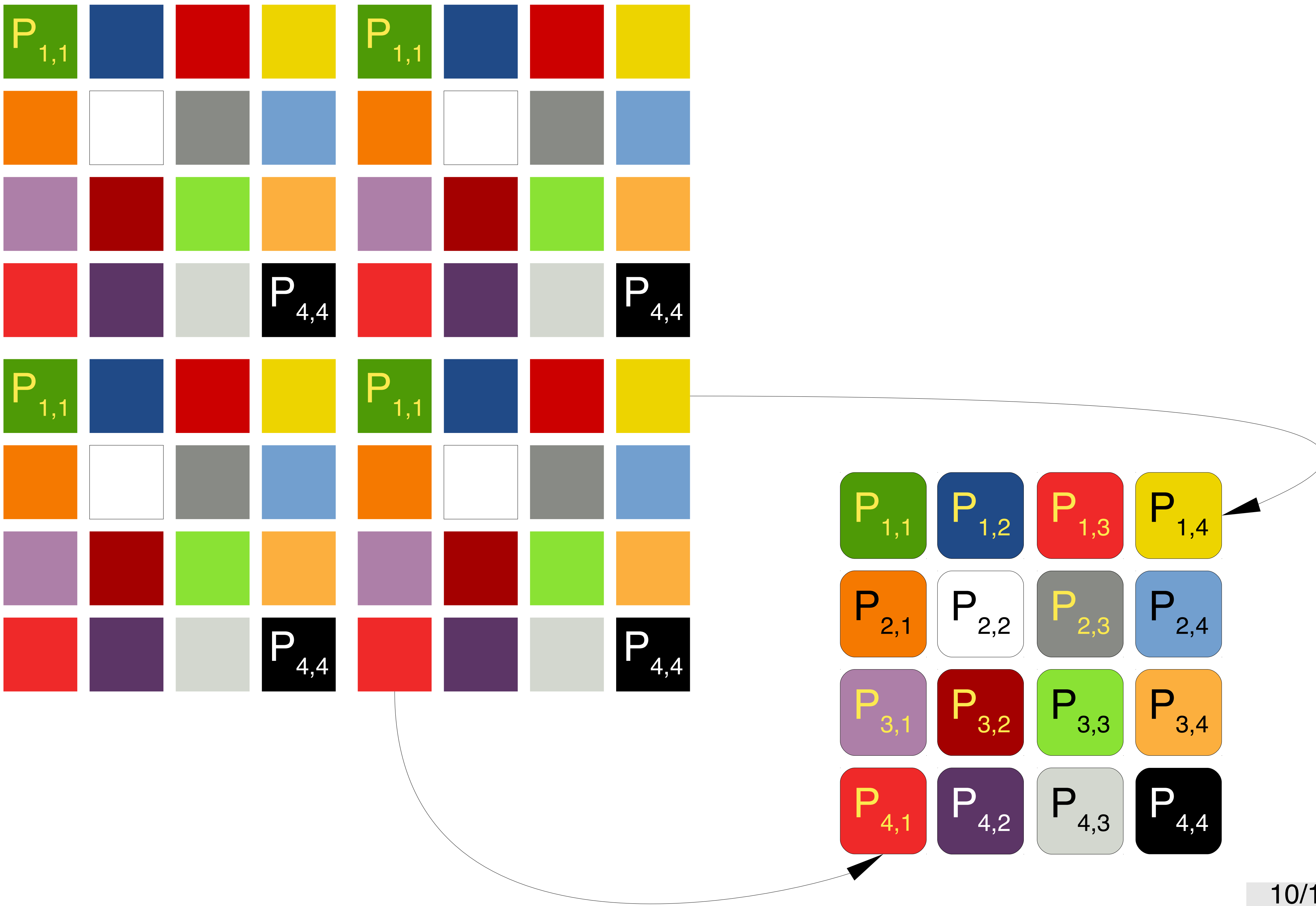
Non-cyclic distribution sequentializes the computational steps:

1. Solve first block
2. Wait for pivot information and scaling factors.



Cyclic distribution in both dimensions minimizes communication and improves scalability.

Mapping: 2D Block Cyclic Distribution



Divisibility and Padding

- If P is not a square of an integer
  - Use prime factors of P to form as square of the process grid as possible
  - If P is a prime then leave some of the processes out of the grid to get close-to-square grid
- If N is not divisible by P
  - Consider implementing clean-up code
  - If extra operations are OK, pad the matrix with 0's and 1's
    - Example: extend to 8

```
aaaaa000 }
aaaaa000 }
aaaaa000 }  padding
aaaaa000 }
aaaaa000 }
00000100
00000010    identity matrix extension
00000001
```

padding