Parallelizing Gaussian Elimination

- **Recall parallelization steps from earlier lecture**
  - **Decomposition:** identify enough parallel work, but not too much
  - **Assignment:** load balance work among threads
  - **Orchestrate:** communication and synchronization
  - **Mapping:** which processors execute which threads

- **Decomposition**
  - In BLAS 2 algorithm nearly each flop in inner loop can be done in parallel, so with n^2 processors, need 3n parallel steps

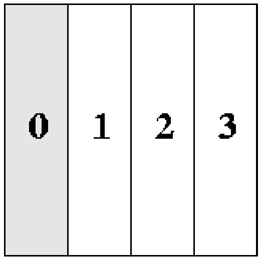
```
for i = 1 to n-1
    A(i+1:n,i) = A(i+1:n,i) / A(i,i)         … BLAS 1 (scale a vector)
    A(i+1:n,i+1:n) = A(i+1:n , i+1:n )   … BLAS 2 (rank-1 update)
           - A(i+1:n , i) * A(i , i+1:n)
```

  - This is too fine-grained, prefer calls to local matmuls instead
  - Need to discuss parallel matrix multiplication

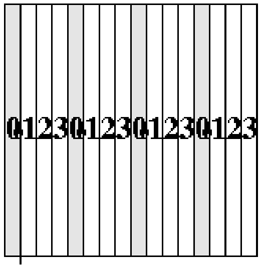
- **Assignment**
  - Which processors are responsible for which submatrices?

Different Data Layouts for Parallel GE (on 4 procs)

Bad load balance: P0 idle after first n/4 steps



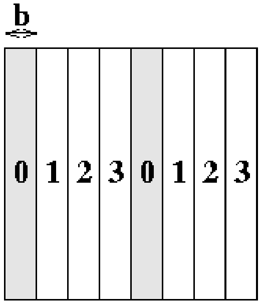
1) Column Blocked Layout



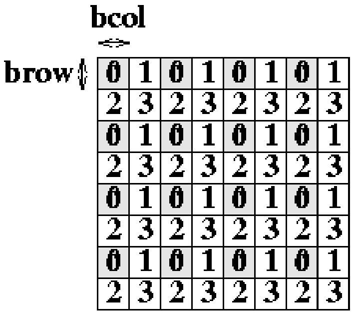
2) Column Cyclic Layout

Load balanced, but can't easily use BLAS2 or BLAS3

Can trade load balance and BLAS2/3 performance by choosing b, but factorization of block column is a bottleneck

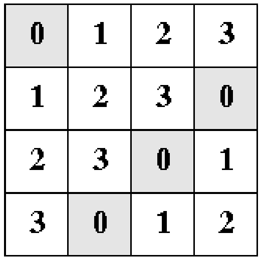


3) Column Block Cyclic Layout



4) Row and Column Block Cyclic Layout

The winner!



Complicated addressing

5) Block Skewed Layout

Performance of PBLAS

PDGEMM = PBLAS routine
for matrix multiply

Observations:

- For fixed N, as P increases Mflops increases, but less than 100% efficiency
- For fixed P, as N increases, Mflops (efficiency) rises

Speed in Mflops of PDGEMM

Machine	Procs	Block Size	N 2000	N 4000	N 10000
Cray T3E	4=2x2	32	1055	1070	0
	16=4x4		3630	4005	4292
	64=8x8		13456	14287	16755
IBM SP2	4	50	755	0	0
	16		2514	2850	0
	64		6205	8709	10774
Intel XP/S MP Paragon	4	32	330	0	0
	16		1233	1281	0
	64		4496	4864	5257
Berkeley NOW	4	32	463	470	0
	32=4x8		2490	2822	3450
	64		4130	5457	6647

DGEMM = BLAS routine
for matrix multiply

Maximum speed for PDGEMM
= # Procs * speed of DGEMM

Observations (same as above):

- Efficiency always at least 48%
- For fixed N, as P increases, efficiency drops
- For fixed P, as N increases, efficiency increases

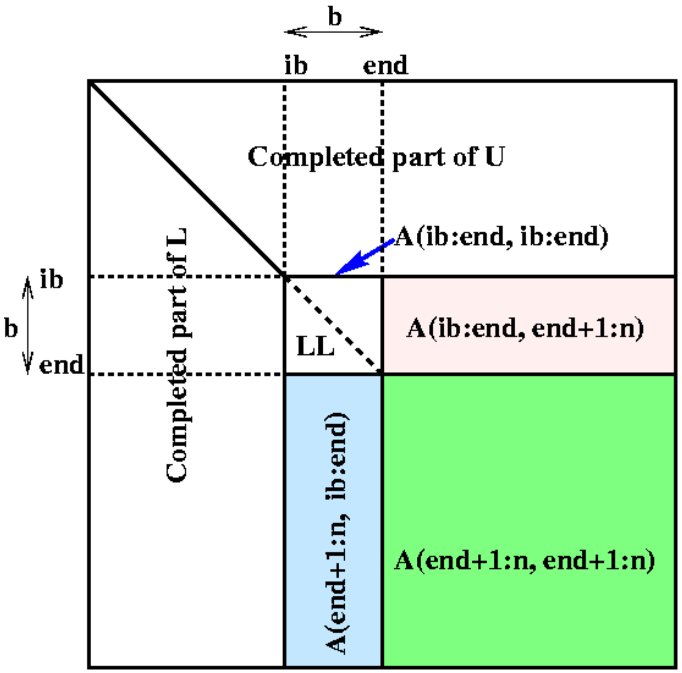
Efficiency = MFlops(PDGEMM)/(Procs*MFlops(DGEMM))

Machine	Peak/proc	DGEMM Mflops	Procs	N 2000	N 4000	N 10000
Cray T3E	600	360	4	.73	.74	
			16	.63	.70	.75
			64	.58	.62	.73
IBM SP2	266	200	4	.94		
			16	.79	.89	
			64	.48	.68	.84
Intel XP/S MP Paragon	100	90	4	.92		
			16	.86	.89	
			64	.78	.84	.91
Berkeley NOW	334	129	4	.90	.91	
			32	.60	.68	.84
			64	.50	.66	.81

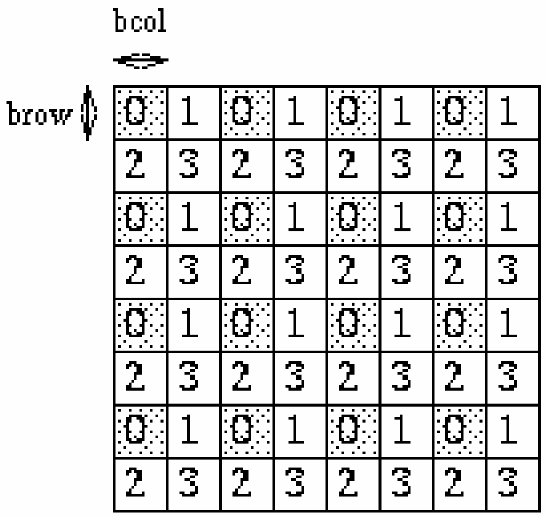
Review: BLAS 3 (Blocked) GEPP

```
for   ib = 1 to n-1 step b     … Process matrix b columns at a time
    end = ib + b-1                  … Point to end of block of b columns
    apply BLAS2 version of GEPP to  get A(ib:n , ib:end) = P' * L' * U'
    … let LL denote the strict lower triangular part of A(ib:end , ib:end) + I
        A(ib:end , end+1:n) = $LL^{-1}$ * A(ib:end , end+1:n)        … update next b rows of U
BLAS 3  A(end+1:n , end+1:n ) = A(end+1:n , end+1:n )
                  - A(end+1:n , ib:end) * A(ib:end , end+1:n)
                                        … apply delayed updates with single matrix-multiply
                                        … with inner dimension b
```

Gaussian Elimination using BLAS 3



Review: Row and Column Block Cyclic Layout



0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3

4) Row and Column Block Cyclic Layout

processors and matrix blocks are distributed in a 2d array

pcol-fold parallelism in any column, and calls to the BLAS2 and BLAS3 on matrices of size brow-by-bcol

serial bottleneck is eased

need not be symmetric in rows and columns

Distributed GE with a 2D Block Cyclic Layout

block size b in the algorithm and the block sizes brow and bcol in the layout satisfy b=brow=bcol.

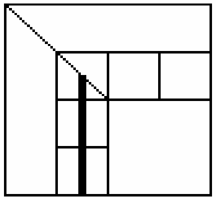
shaded regions indicate busy processors or communication performed.

unnecessary to have a barrier between each step of the algorithm, e.g.. step 9, 10, and 11 can be pipelined

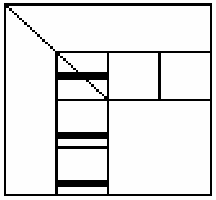
Distributed Gaussian Elimination with a 2D Block Cyclic Layout

```
for ib = 1 to n-1 step b
    end = min( ib+b-1, n )
    for i = ib to end
        (1)   find pivot row k, column broadcast
        (2)   swap rows k and i in block column, broadcast row k
        (3)   A( i+1:n , i) = A( i+1:n , i) / A( i , i)
        (4)   A( i+1:n , i+1:end) -= A( i+1:n , i) * A( i , i+1:end )
    end for
(5)   broadcast all swap information right and left
(6)   apply all rows swaps to other columns
```

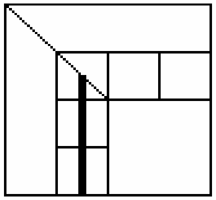
(1)



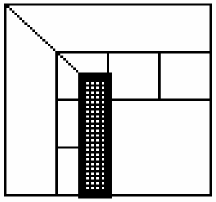
(2)



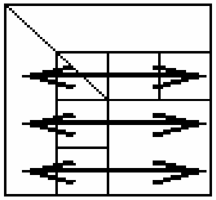
(3)



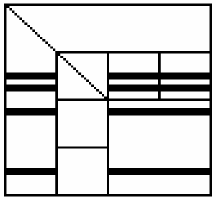
(4)



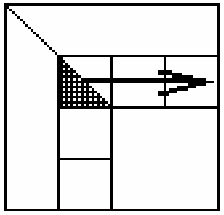
(5)



(6)

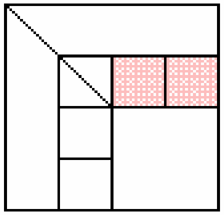


(7)



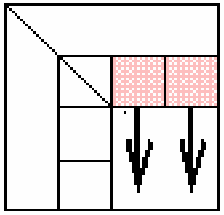
(7) Broadcast LL right

(8)



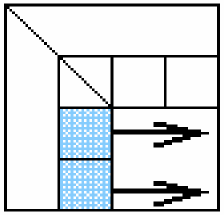
(8) A(ib:end , end+1:n) = LL \ A(ib:end , end+1:n)

(9)



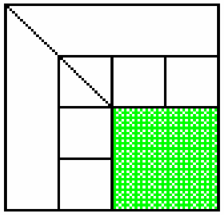
(9) Broadcast A(ib:end , end+1:n) down

(10)



(10) Broadcast A(end+1:n , ib:end) right

(11)



(11) Eliminate A(end+1:n , end+1:n)

Matrix multiply of
green = green - blue * pink

C

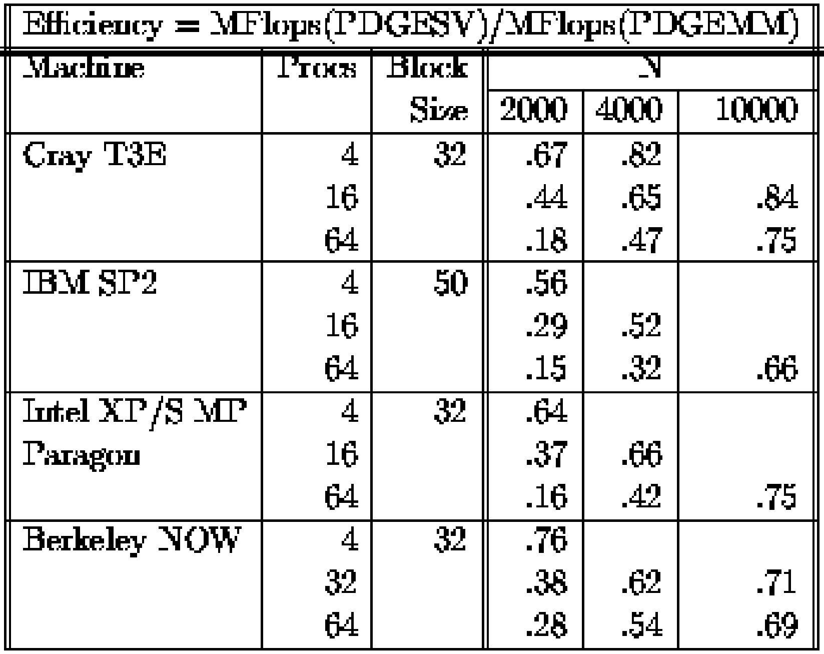
Performance of ScaLAPACK LU

PDGESV = ScaLAPACK parallel LU routine

Since it can run no faster than its inner loop (PDGEMM), we measure:
Efficiency = Speed(PDGESV)/Speed(PDGEMM)

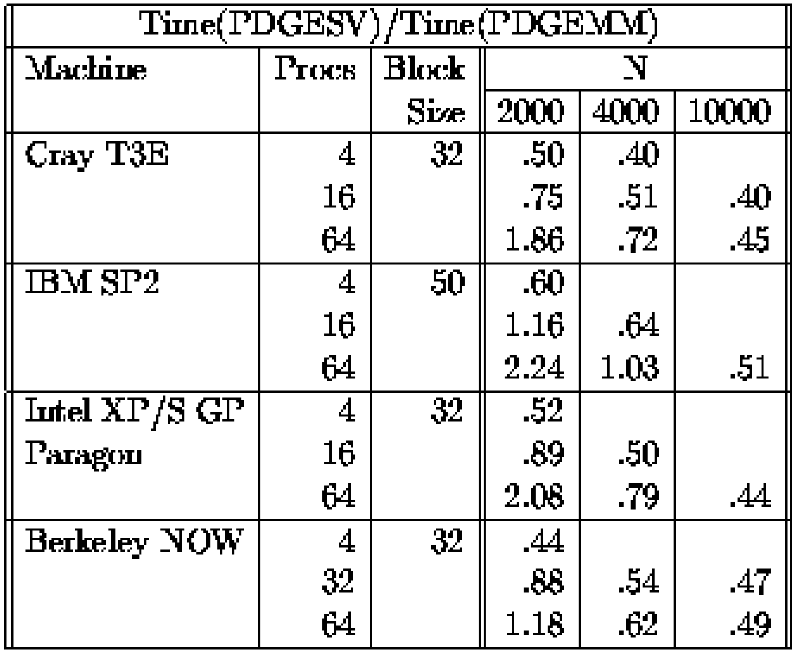
Observations:

- **Efficiency well above 50% for large enough problems**
- **For fixed N, as P increases, efficiency decreases (just as for PDGEMM)**
- **For fixed P, as N increases efficiency increases (just as for PDGEMM)**
- **From bottom table, cost of solving Ax=b about half of matrix multiply for large enough matrices. From the flop counts we would expect it to be $(2*n^3)/(2/3*n^3) = 3$ times faster, but communication makes it a little slower.**



Efficiency = MFlops(PDGESV)/MFlops(PDGEMM)

Machine	Procs	Block Size	N 2000	N 4000	N 10000
Cray T3E	4	32	.67	.82	
	16		.44	.65	.84
	64		.18	.47	.75
IBM SP2	4	50	.56		
	16		.29	.52	
	64		.15	.32	.66
Intel XP/S MP Paragon	4	32	.64		
	16		.37	.66	
	64		.16	.42	.75
Berkeley NOW	4	32	.76		
	32		.38	.62	.71
	64		.28	.54	.69



Time(PDGESV)/Time(PDGEMM)

Machine	Procs	Block Size	N 2000	N 4000	N 10000
Cray T3E	4	32	.50	.40	
	16		.75	.51	.40
	64		1.86	.72	.45
IBM SP2	4	50	.60		
	16		1.16	.64	
	64		2.24	1.03	.51
Intel XP/S GP Paragon	4	32	.52		
	16		.89	.50	
	64		2.08	.79	.44
Berkeley NOW	4	32	.44		
	32		.88	.54	.47
	64		1.18	.62	.49