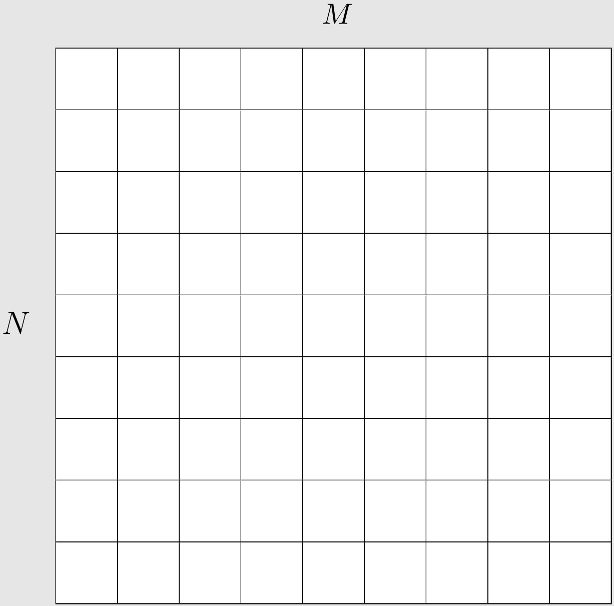
Mapping Dense LU Factorization on Multicore Supercomputer Nodes

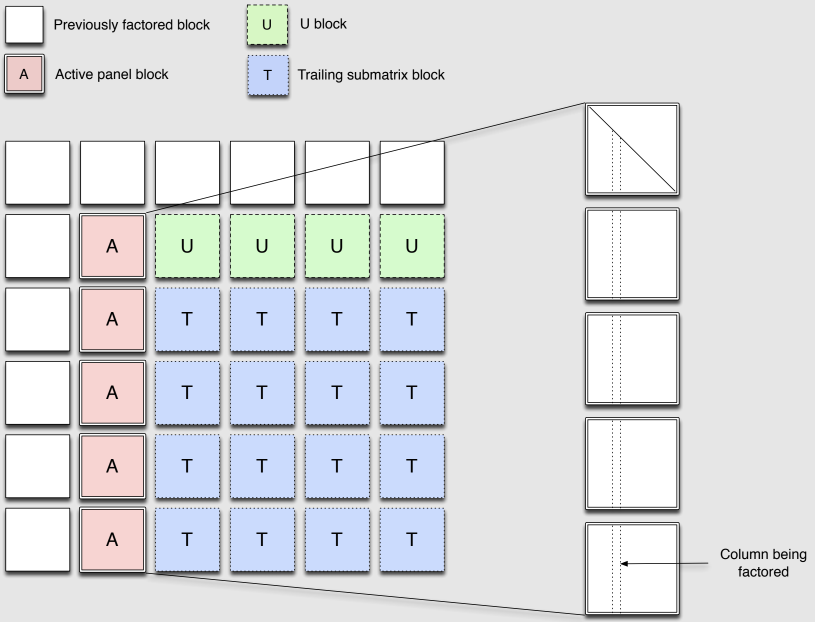
Jonathan Lifflander[†], Phil Miller[†], Ramprasad Venkataraman[†],
Anshu Arya[†], Terry Jones[‡], Laxmikant Kale[†]

[†]University of Illinois Urbana-Champaign
[‡]Oak Ridge National Laboratory

May 7, 2012



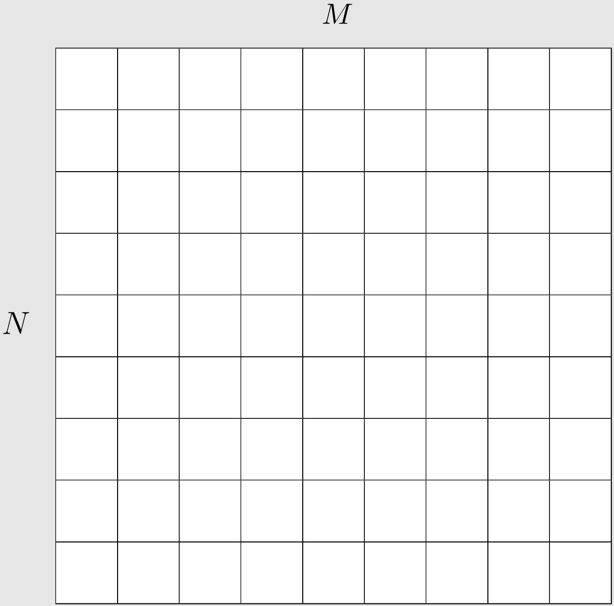
M
N


Previously factored block
U
U block
A
Active panel block
T
Trailing submatrix block
A U U U U
A T T T T
A T T T T
A T T T T
A T T T T
Column being factored

Block-cyclic process grid: 24 cores

0	4	8	12	16	20
1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23


M
N

Array Mapping in Charm++

```
opts.setMap(map);
CProxy_LUMgr mgr = CProxy_PrioLU::ckNew(blockSize, matrixSize);
```

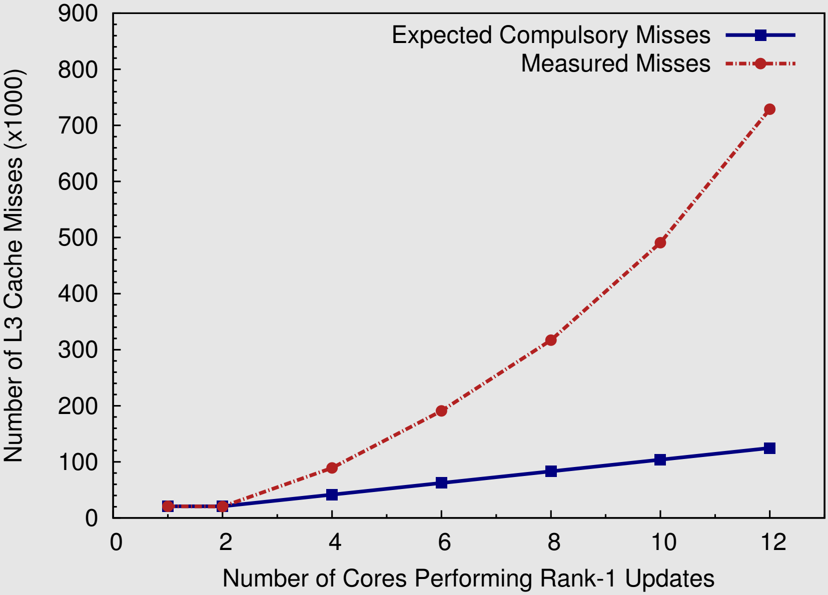
0	4	8	12	16	20	0	4	8
1	5	9	13	17	21	1	5	9
2	6	10	14	18	22	2	6	10
3	7	11	15	19	23	3	7	11
0	4	8	12	16	20	0	4	8
1	5	9	13	17	21	1	5	9
2	6	10	14	18	22	2	6	10
3	7	11	15	19	23	3	7	11
0	4	8	13	16	20	0	4	8

0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8
12	13	14	15	12	13	14	15	12
16	17	18	19	16	17	18	19	16
20	21	22	23	20	21	22	23	20
0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8

0-3	node 0	4-7	node 1	8-11	node 2
12-15	node 3	16-19	node 4	20-23	node 5

Rank-1 Update Times (ms)

Microarchitecture	Cores/socket performing updates						Efficiency
	1	2	3	4	5	6	
Intel Nehalem-EP	16	22	30	38			42%
AMD Istanbul	17	27	38	50	63	76	22%
IBM Blue Gene/P	19	20	20	22			86%


Expected Compulsory Misses
Measured Misses
Number of L3 Cache Misses (x1000)
0
100
200
300
400
500
600
700
800
900
0
2
4
6
8
10
12
Number of Cores Performing Rank-1 Updates

Striding: a range of values for u

Block-cylic:

0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8
12	13	14	15	12	13	14	15	12
16	17	18	19	16	17	18	19	16
20	21	22	23	20	21	22	23	20
0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8

Block-cylic with stride ($s = 2$):

0	1	12	13	0	1	12	13	0
2	3	14	15	2	3	14	15	2
4	5	16	17	4	5	16	17	4
6	7	18	19	6	7	18	19	6
8	9	20	21	8	9	20	21	8
10	11	22	23	10	11	22	23	10
0	1	12	13	0	1	12	13	0
2	3	14	15	2	3	14	15	2
4	5	16	17	4	5	16	17	4

Formulæ: augmenting block-cyclic with a stride s

Block-cylic with striding:

$$p = \left\lfloor \frac{y}{Q} \right\rfloor \qquad \text{(grid y index)}$$

$$m_3 = x \bmod P \qquad \text{(x in grid)}$$

$$n_3 = y \bmod s \qquad \text{(y in subgrid)}$$

$$q = \left\lfloor \frac{y \bmod Q}{s} \right\rfloor \qquad \text{(subgrid y)}$$

$$f_s(x, y, P, Q, s) = m_3 s + n_3 + Psq \qquad (3)$$

where $1 \leq s \leq Q$ and s is a factor of Q.

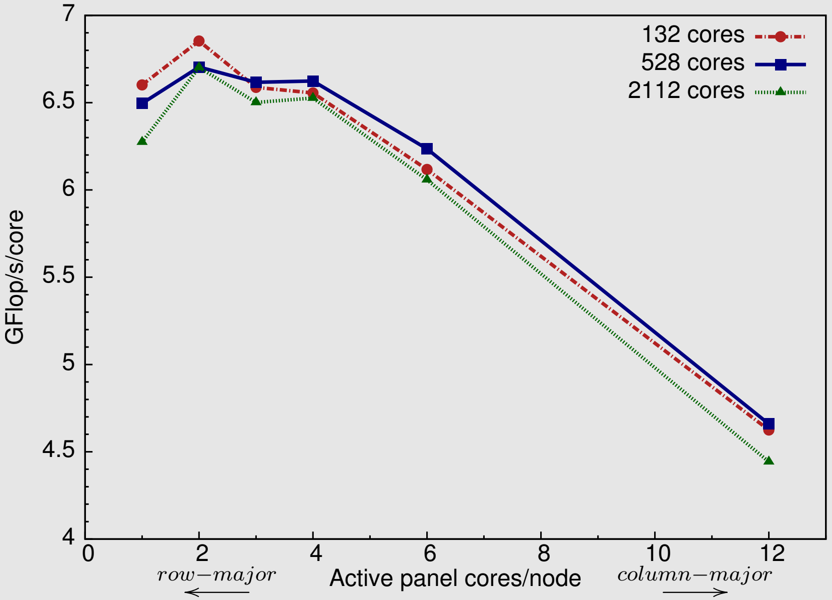
Block-cylic:

$$m_1 = x \bmod P \qquad \text{(x in grid)}$$

$$n_1 = y \bmod Q \qquad \text{(y in grid)}$$

$$f_r(x, y, P, Q) = m_1 Q + n_1 \qquad (1)$$

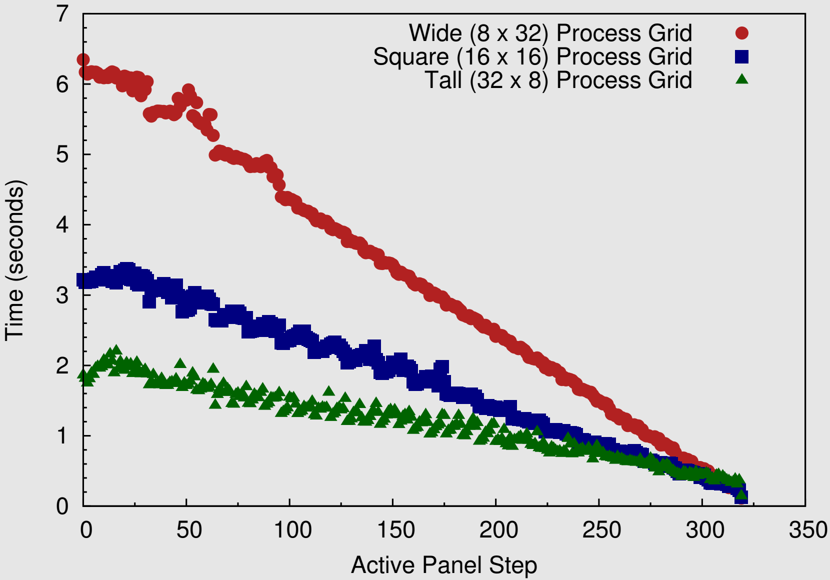
$$f_c(x, y, P, Q) = n_1 P + m_1 \qquad (2)$$

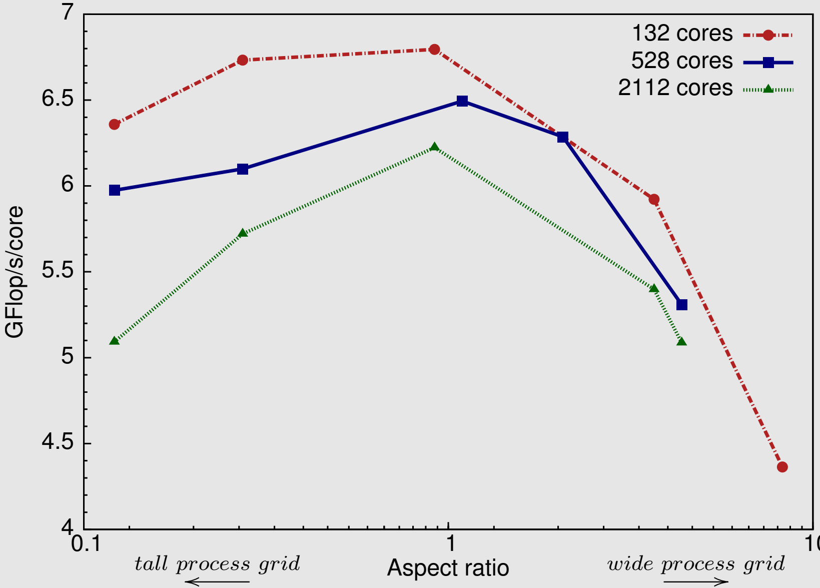


132 cores
528 cores
2112 cores
GFlop/s/core
7
6.5
6
5.5
5
4.5
4
0
2
4
6
8
10
12
row−major
Active panel cores/node
column−major

0	4	8	12	16	20	0	4	8
1	5	9	13	17	21	1	5	9
2	6	10	14	18	22	2	6	10
3	7	11	15	19	23	3	7	11
0	4	8	12	16	20	0	4	8
1	5	9	13	17	21	1	5	9
2	6	10	14	18	22	2	6	10
3	7	11	15	19	23	3	7	11
0	4	8	13	16	20	0	4	8

0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8
12	13	14	15	12	13	14	15	12
16	17	18	19	16	17	18	19	16
20	21	22	23	20	21	22	23	20
0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8


Wide (8 x 32) Process Grid
Square (16 x 16) Process Grid
Tall (32 x 8) Process Grid
Time (seconds)
Active Panel Step
0
1
2
3
4
5
6
7
0
50
100
150
200
250
300
350



7
6.5
6
5.5
5
4.5
4
GFlop/s/core
0.1
1
10
Aspect ratio
tall process grid
wide process grid
132 cores
528 cores
2112 cores

Rotation: increasing row parallelism

Block-cylic:

0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8
12	13	14	15	12	13	14	15	12
16	17	18	19	16	17	18	19	16
20	21	22	23	20	21	22	23	20
0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8

Block-cylic with rotation ($r = 2$):

0	1	2	3	8	9	10	11	16
4	5	6	7	12	13	14	15	20
8	9	10	11	16	17	18	19	0
12	13	14	15	20	21	22	23	4
16	17	18	19	0	1	2	3	8
20	21	22	23	4	5	6	7	12
0	1	2	3	8	9	10	11	16
4	5	6	7	12	13	14	15	20
8	9	10	11	16	17	18	19	0

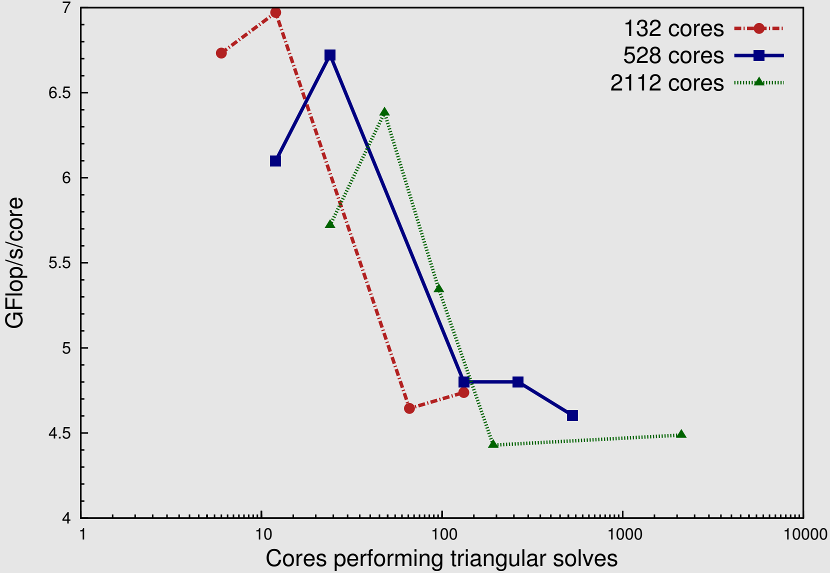
Formulæ: augmenting block-cyclic with a rotation r

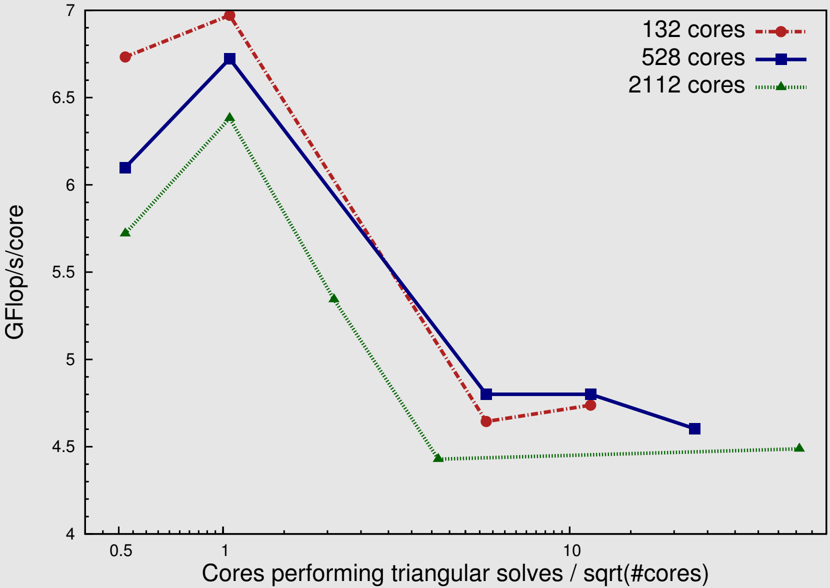
Block-cylic:

$$m_1 = x \bmod P \quad \text{(x in grid)}$$
$$n_1 = y \bmod Q \quad \text{(y in grid)}$$
$$f_r(x, y, P, Q) = m_1 Q + n_1 \qquad (4)$$
$$f_c(x, y, P, Q) = n_1 P + m_1 \qquad (5)$$

Block-cylic with rotation:

$$p = \left\lfloor \frac{y}{Q} \right\rfloor \quad \text{(grid y index)}$$
$$m_2 = (x + pr) \bmod P \quad \text{(x in grid)}$$
$$n_2 = y \bmod Q \quad \text{(y in grid)}$$
$$f_{rotRow}(x, y, P, Q, r) = m_2 Q + n_2 \qquad (6)$$
$$f_{rotCol}(x, y, P, Q, r) = n_2 P + m_2 \qquad (7)$$


132 cores
528 cores
2112 cores
GFlop/s/core
7
6.5
6
5.5
5
4.5
4
1
10
100
1000
10000
Cores performing triangular solves


132 cores
528 cores
2112 cores
GFlop/s/core
7
6.5
6
5.5
5
4.5
4
0.5
1
10
Cores performing triangular solves / sqrt(#cores)

Combining striding and rotation

Block-cylic:

0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8
12	13	14	15	12	13	14	15	12
16	17	18	19	16	17	18	19	16
20	21	22	23	20	21	22	23	20
0	1	2	3	0	1	2	3	0
4	5	6	7	4	5	6	7	4
8	9	10	11	8	9	10	11	8

Block-cylic with striding and rotation:

0	1	12	13	4	5	16	17	8
2	3	14	15	6	7	18	19	10
4	5	16	17	8	9	20	21	0
6	7	18	19	10	11	22	23	2
8	9	20	21	0	1	12	13	4
10	11	22	23	2	3	14	15	6
0	1	12	13	4	5	16	17	8
2	3	14	15	6	7	18	19	10
4	5	16	17	8	9	20	21	0

Formulæ: augmenting parameters r and s

Block-cylic:

$$m_1 = x \bmod P \quad \text{(x in grid)}$$
$$n_1 = y \bmod Q \quad \text{(y in grid)}$$
$$f_r(x, y, P, Q) = m_1 Q + n_1 \qquad (8)$$
$$f_c(x, y, P, Q) = n_1 P + m_1 \qquad (9)$$

Block-cylic with striding and rotation:

$$p = \left\lfloor \frac{y}{Q} \right\rfloor \quad \text{(grid y index)}$$
$$m_4 = (x + pr) \bmod P \quad \text{(x in grid)}$$
$$n_4 = y \bmod s \quad \text{(y in grid)}$$
$$q = \left\lfloor \frac{y \bmod Q}{s} \right\rfloor \quad \text{(subgrid y)}$$
$$f_{sr}(x, y, P, Q, r, s) = m_4 s + n_4 + Psq \qquad (10)$$

Stride and Rotation Mapping in Charm++

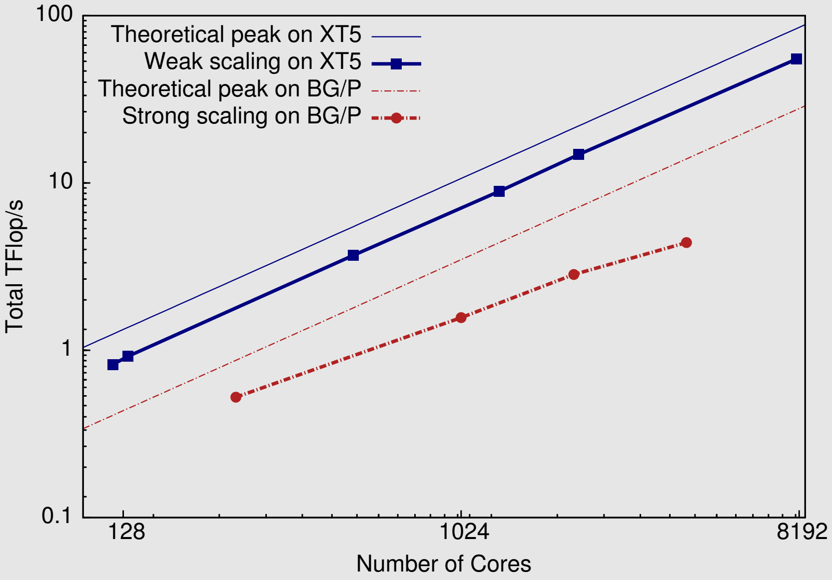
```
int map(const int coor[2]) {
  int tileYIndex = coor[1]  / peCols;
  int XwithinPEtile = (coor[0] + tileYIndex * peRotate) %
                      peRows;
  int YwithinPEtile = coor[1] % (peCols / peStride);
  int subtileY = (coor[1] % peCols) / (peCols / peStride);
  int peNum = XwithinPEtile * peStride + YwithinPEtile *
              peStride * peRows + subtileY
  CkAssert(peNum < CkNumPes());
  return peNum;
}
```

Data Movement Overhead

Number of Cores	528	2112
Factorization Time (seconds)	653.3	1427.4
Data Movement Time (seconds)	12.4	14.1
Total Time (seconds)	665.7	1441.5
Data Movement Percentage of Total	1.9%	1.0%

Conclusion

- Performance compared to best square grid
  - Striding: 8% increase
  - Rotation: 3% increase
  - Combined: 11% increase, corresponding to a 7% increase in peak performance
- With memory usage constant at about 75% on Cray XT5:
  - About 67% of peak from 120 cores to 8064 cores



Theoretical peak on XT5
Weak scaling on XT5
Theoretical peak on BG/P
Strong scaling on BG/P
Total TFlop/s
100
10
1
0.1
128
1024
8192
Number of Cores