Cholesky Factorization

Project of CS594
By: Amir Saadat
PhD student of Chemical and Biomolecular Eng. Dep.
University of Tennessee

Motivation:

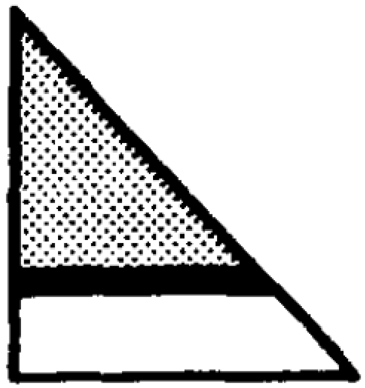
- Application in Monte Carlo simulation.
- Brownian Dynamic Simulation of Bead-rod and Bead-Spring polymers with Hydrodynamic Interaction .
- Solving linear system of equations (resulting in symmetric positive definite A in Ax=b)
- …

Definition:

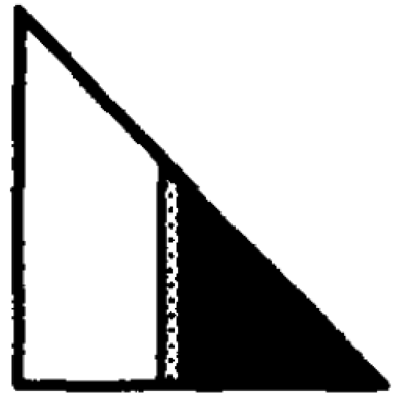
A = L . LT

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{21} & a_{22} & a_{32} & a_{42} \\ a_{31} & a_{32} & a_{33} & a_{43} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} * \begin{bmatrix} l_{11} & l_{21} & l_{31} & l_{41} \\ 0 & l_{22} & l_{32} & l_{42} \\ 0 & 0 & l_{33} & l_{43} \\ 0 & 0 & 0 & l_{44} \end{bmatrix}$$

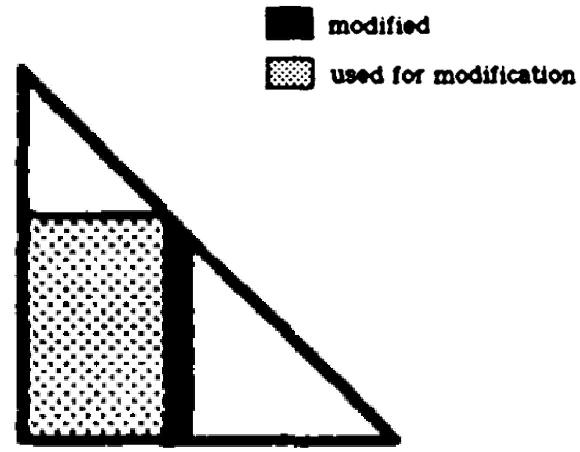
$$\begin{bmatrix} {l_{11}}^2 & l_{11}l_{21} & l_{11}l_{31} & l_{11}l_{41} \\ l_{11}l_{21} & {l_{21}}^2 + {l_{22}}^2 & l_{21}l_{31} + l_{22}l_{32} & l_{21}l_{41} + l_{22}l_{42} \\ l_{11}l_{31} & l_{21}l_{31} + l_{22}l_{32} & {l_{31}}^2 + {l_{32}}^2 + {l_{33}}^2 & l_{31}l_{41} + l_{32}l_{42} + l_{33}l_{43} \\ l_{11}l_{41} & l_{21}l_{41} + l_{22}l_{42} & l_{31}l_{41} + l_{32}l_{42} + l_{33}l_{43} & {l_{41}}^2 + {l_{42}}^2 + {l_{43}}^2 + {l_{44}}^2 \end{bmatrix}$$



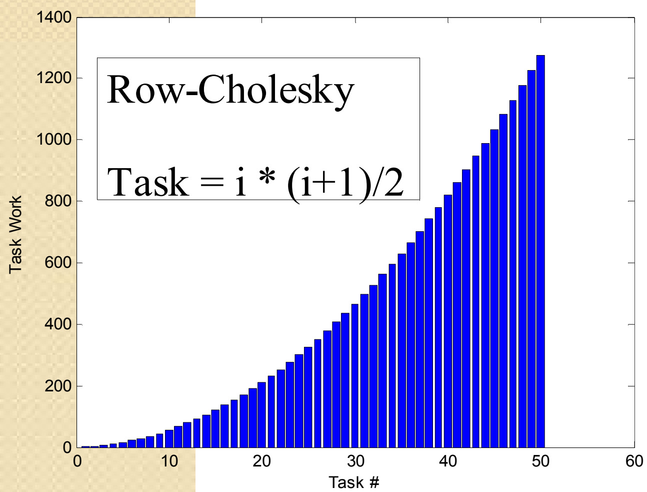
Row Cholesky

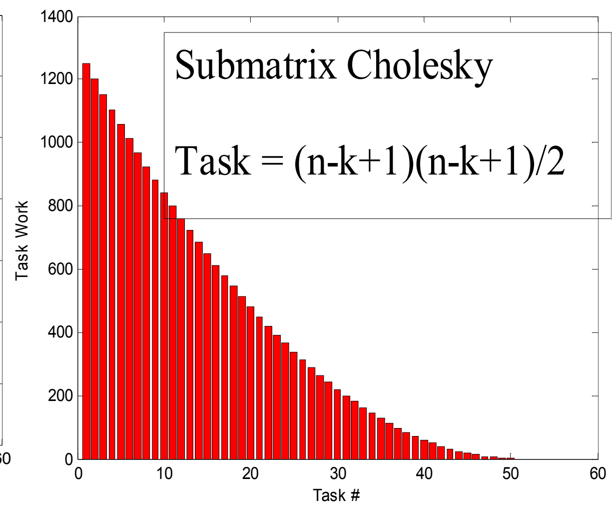


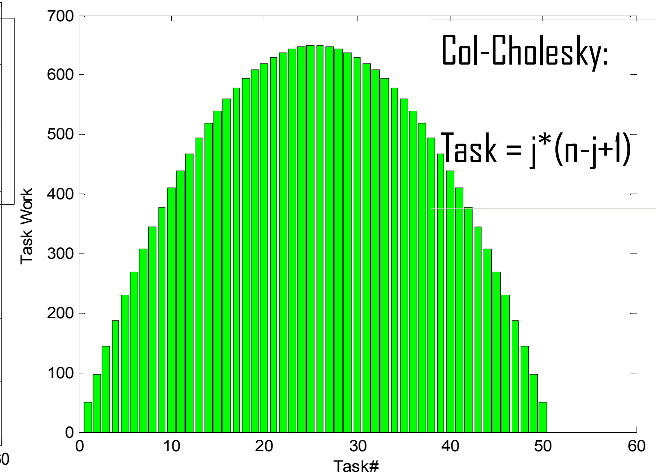
Submatrix Cholesky



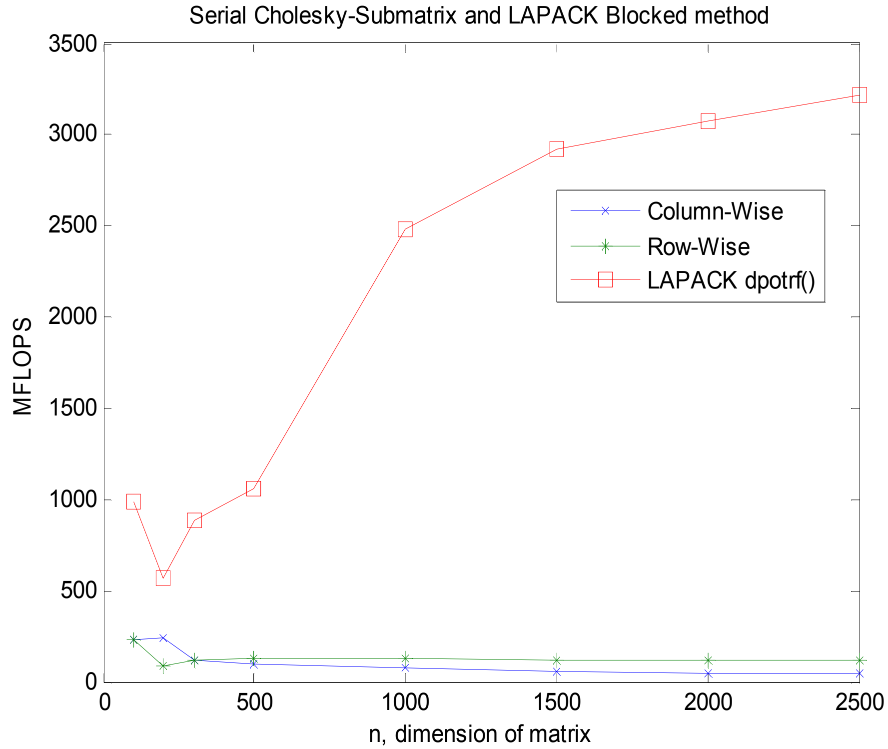
Column Cholesky







Using Row Cholesky in Serial
Comparison with dpotrf_() LAPACK



Data Storage

$\begin{pmatrix} a_{00} & & \\ a_{10} & a_{11} & \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$	$a_{00}\ a_{10}\ a_{20}\ *\ a_{11}\ a_{21}\ *\ *\ a_{22}$
$\begin{pmatrix} a_{00} & & \\ a_{10} & a_{11} & \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$	$a_{00}\ *\ *\ a_{10}\ a_{11}\ *\ a_{20}\ a_{21}\ a_{22}$

Advantages:
✓ Row major access pattern in doing Column Cholesky
✓Ease of access
✓ Decreasing the amount of cache misses

Disadvantage
- Increment in floating point operations

$$a_{i,j} \rightarrow a_{i,j*(2n-j-1)/2}$$

Parallelizing in Shared Memory using OpenMP

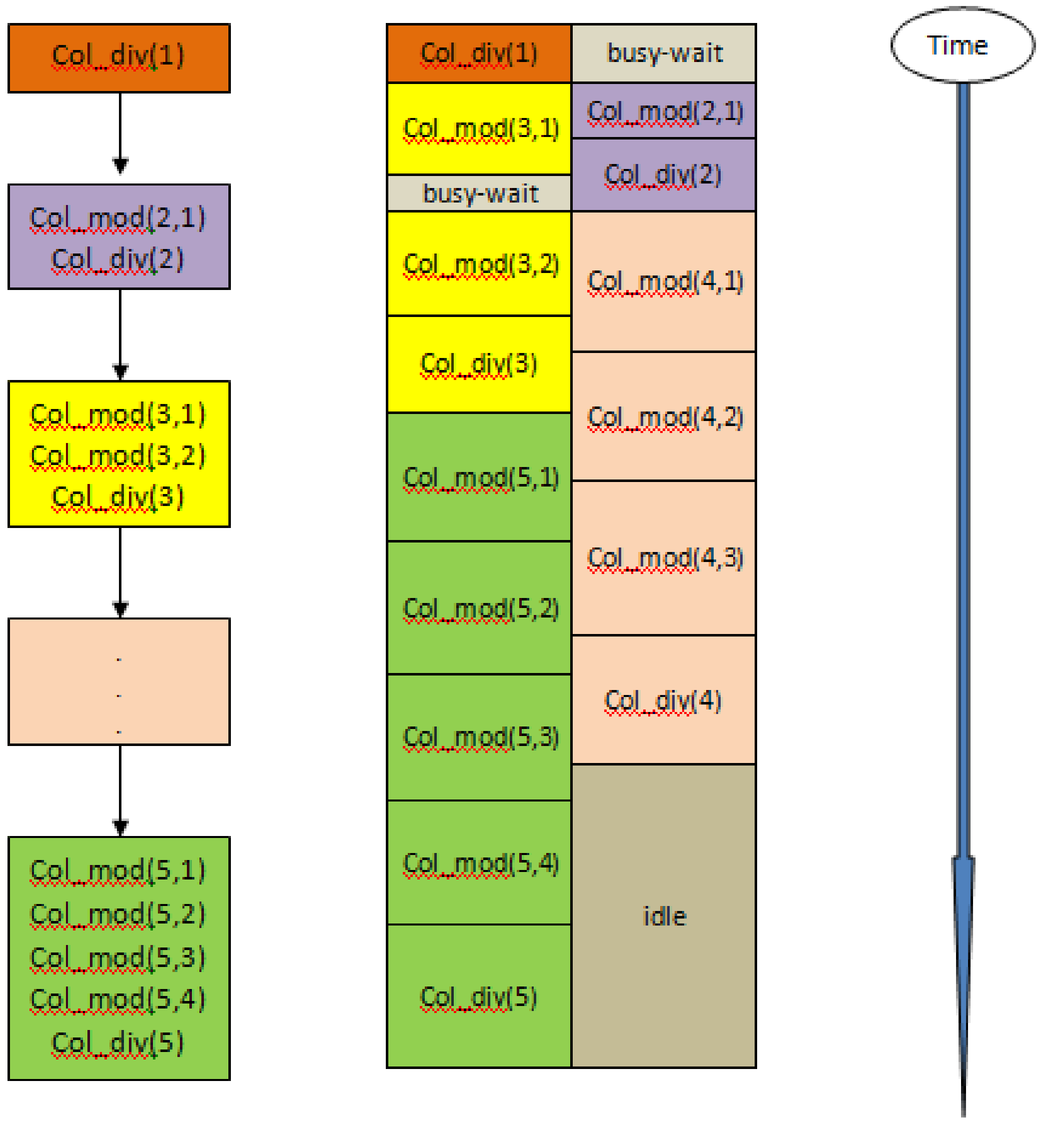
```
Cholesky_Parallel_OMP( ) {
   for j = 1 to n
   begin
      Pick up task Task_col(j) from task queue
   end

}
```

```
Col_mod( j, k ) {
        for i = j to n
            begin
                A [i, j] = A [i, j] – A [j, k] * A [i, k]
            end
                }
```

```
Task_col( j ) {

        for k = 1 to j-1

            begin

                Wait until the ready[k] flag has been set

                Perform Col_mod(j,k) operation

            end

            Perform Col_div(j) operation

            Set the ready[j] flag

              }
```

MKL_CBLAS calls:

- ✓ Cblas_daxpy()
- ✓ Cblas_dscal()

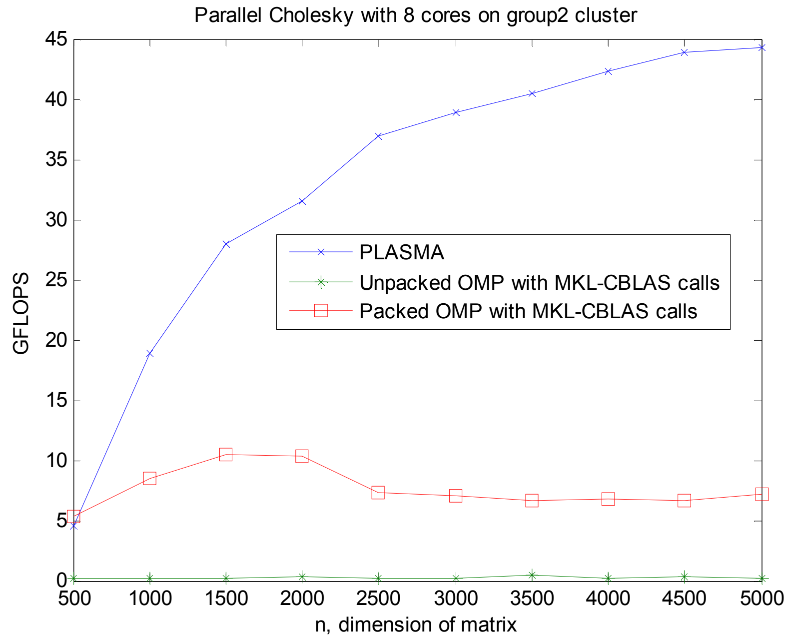

Col_div(1)
Col_mod(2,1)
Col_div(2)
Col_mod(3,1)
Col_mod(3,2)
Col_div(3)
.
.
.
Col_mod(5,1)
Col_mod(5,2)
Col_mod(5,3)
Col_mod(5,4)
Col_div(5)
Col_div(1)
busy-wait
Col_mod(3,1)
Col_mod(2,1)
Col_div(2)
busy-wait
Col_mod(3,2)
Col_mod(4,1)
Col_div(3)
Col_mod(4,2)
Col_mod(5,1)
Col_mod(4,3)
Col_mod(5,2)
Col_div(4)
Col_mod(5,3)
Col_mod(5,4)
idle
Col_div(5)
Time

UNPACKED

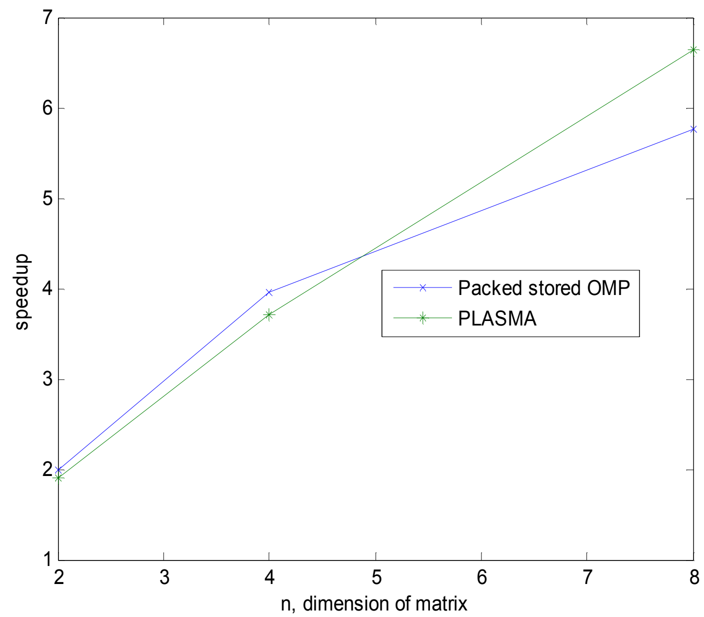
Order of matrix (n)	Sequential run-time (sec)	P = 2		p = 4		p = 8	
		Parallel run-time (sec)	Speedup	Parallel run-time (sec)	Speedup	Parallel run-time (sec)	Speedup
500	0.066575	0.573751	0.11	0.253307	0.26	0.180856	0.39
1500	2.741704	17.194054	0.15	5.325873	0.52	5.500979	0.5
2000	6.591150	15.413304	0.42	11.252013	0.58	17.690219	0.37
2500	14.660367	63.760863	0.23	22.191439	0.66	19.195476	0.76
3000	26.775183	44.543718	0.60	55.401160	0.48	29.904158	0.90
3500	40.449806	165.01936	0.24	53.391307	0.75	43.333751	0.93
5000	152.30613	345.0206	0.44	222.24456	0.68	121.26268	1.25

PACKED

Order of matrix (n)	Sequential run-time (sec)	P = 2		p = 4		p = 8		p = 8, PLASMA
		Parallel run-time (sec)	Speedup	Parallel run-time (sec)	Speedup	Parallel run-time (sec)	Speedup	Speedup
500	0.025132	0.072621	0.35	0.011709	2.14	0.005930	4.23	1.33
1500	0.504532	0.344995	1.46	0.168797	2.98	0.114359	4.41	5.01
2000	1.492682	0.885646	1.68	0.412234	3.62	0.277910	5.37	5.37
2500	3.784008	2.047774	1.85	1.042373	3.63	0.687795	5.50	6.09
3000	6.834918	3.703445	1.85	2.033217	3.36	1.268673	5.38	6.01
3500	8.773805	5.767410	1.52	3.036204	2.88	2.072095	4.23	6.21
5000	34.09648	16.31397	2.00	8.589718	3.96	5.906355	5.77	7.23

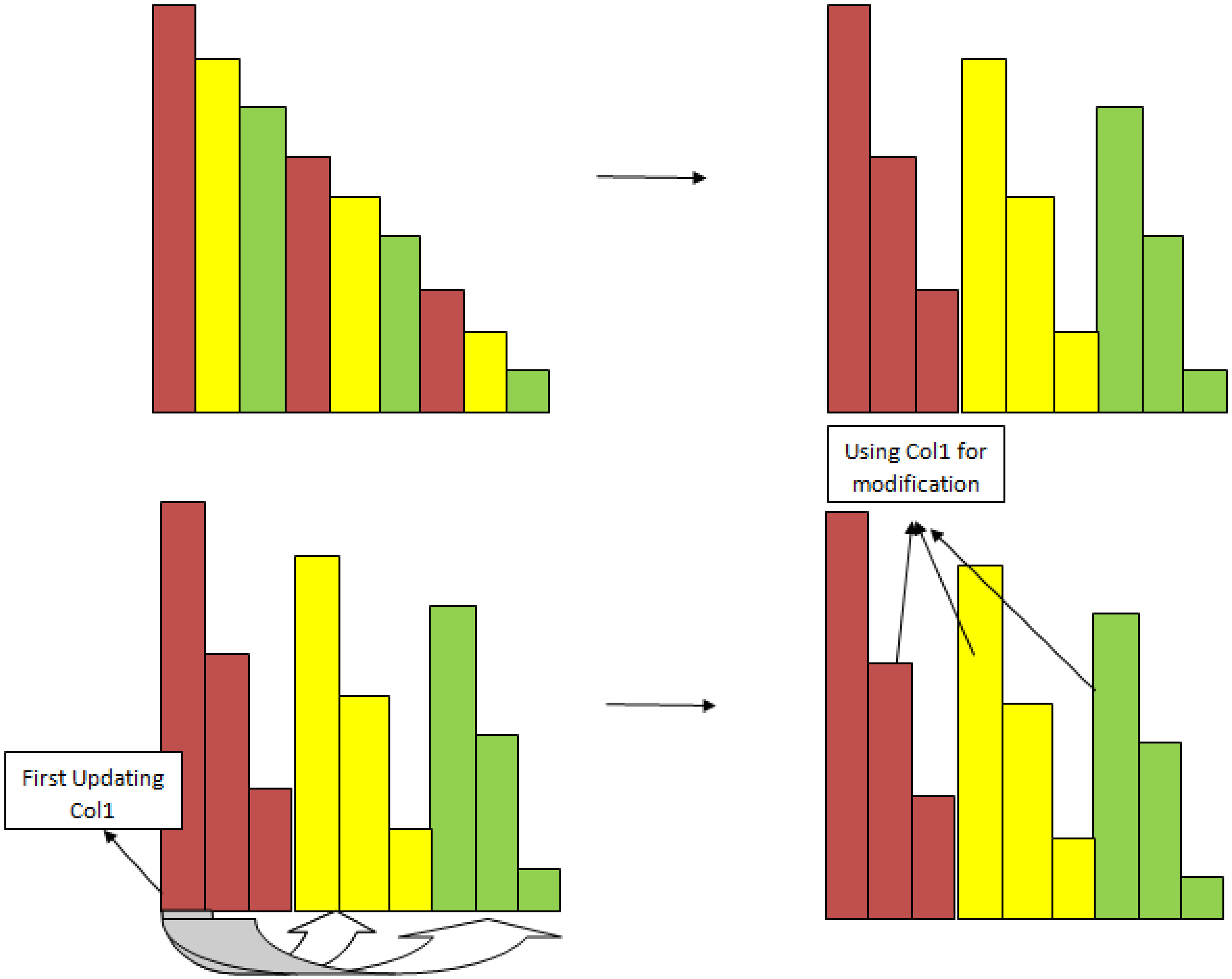


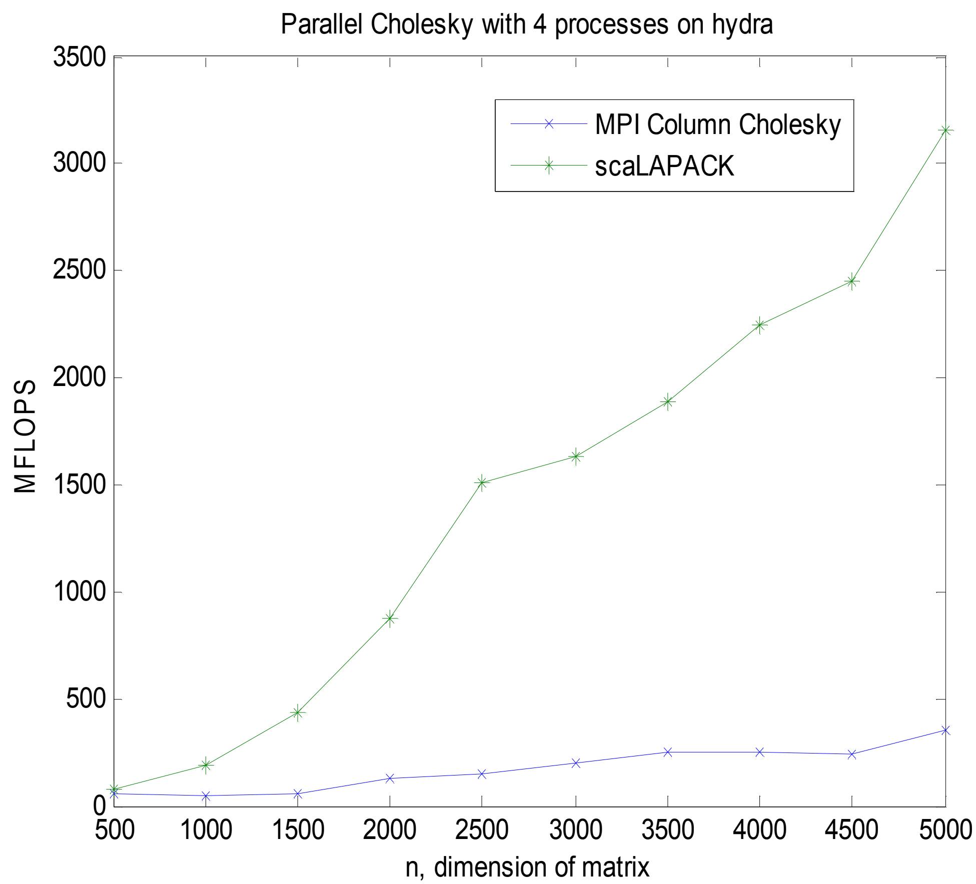
Parallel Cholesky with 8 cores on group2 cluster
GFLOPS
n, dimension of matrix
PLASMA
Unpacked OMP with MKL-CBLAS calls
Packed OMP with MKL-CBLAS calls
0
5
10
15
20
25
30
35
40
45
500
1000
1500
2000
2500
3000
3500
4000
4500
5000



speedup
n, dimension of matrix
Packed stored OMP
PLASMA
1
2
3
4
5
6
7
8

Parallelizing in Distributed Memory using MPI




Parallel Cholesky with 4 processes on hydra
MPI Column Cholesky
scaLAPACK
MFLOPS
n, dimension of matrix
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

Summary:

1. Powerful LAPACK dpotrf() in serial Cholesky.
2. Column Cholesky in Parallelization (shared Memory).
3. Significant improvement of parallel Cholesky by using packed storage.
4. Efficient PLASMA in multicore system.
5. Block Cyclic distribution and high performance as a result in scaLAPACK.

References:

1. A. George, M.T. Heath and J. Liu, Parallel Cholesky Factorization on Shared-Memory Multiprocessor, Linear Algebra and its Applications 77:165-187, 1986.

2. Ref[2]. http://www.netlib.org/blas/blast-forum/chapter2.pdf