SCA/HPCAsia 2026 (Osaka, Japan) Poster
Jan. 27th, 2026; PM13:30-16:00

Halve the Amount of Storage Transfer for LU Decomposition with Complete Pivoting

Hiroshi Murakami (Tokyo Metropolitan Univ. (Retiree Affilate))
Minami-Osawa, Hachi-Oji, Tokyo, Japan
email: mrkmhrsh at-mark tmu.ac.jp

ABSTRACT

We will show that the amount of memory transter to calculate the LU decomposition with complete pivoting can be halved by a simple coding modification. When we introduce the complete pivoting, the tiling method cannot be applied to make LU decomposition.

For the size n matrix A in an array, we sequentially at the k-th step iteration the pivot which has the largest magnitude is searched in the area `A(k:n,k:n)`, and the columns and rows are exchanged appropriately to make the pivot element be placed at the principal diagonal, and then the k-th step elimination is applied for the area

`A(k+1:n,k+1:n)`. If this trivial method of calculation is used, the area of two dimensional array which is gradually shrinking is swept twice in each step of iteration.

Our present method, by a simple coding modification, the order of the access to the storage of the array is changed so that while in the elimination the updating process of the matrix elements and the search process of the largest element in magnitude are merged together, which makes the sweep of the area `A(k+1:n,k+1:n)` not twice but once.

By our experiments, this reduction of the amount of storage transfer for the matrix A has an effect to reduce the elapsed time to calculate the LU decomposition with complete pivotting.

Background

We tried to reduce the elapsed time of the LU decomposition with complete pivoting. It is not possible to make a tiled algorithm for the LU decomposition when complete pivoting is made.

The advantage to use the complete pivoting is the best numerical stability and the best accuracy of the solution. The partial pivoting (row exchange) has a potential numerical instability by the growth of elements in the U matrix in magnitudes. The most disadvantage of the complete pivoting is, the increase of operations to make comparisons to find the pivot which is as many as operations used for LU eliminations. The overhead for the partial pivoting process can be ignored comparing with the amount of computation for the elimination process. However the partial pivoting has a potential numerical instability. Recently, there is the Rook pivoting method[1][2] whose pivoting process

overhead is a few times of that of the partial pivoting one, which usually can be ignored comparing with the amount of computations for the elimination process. The Rook pivoting has no numerical instability and the attainable accuracy of the solution is comparable with the one by the complete pivoting. However, today the complete pivoting is still included in numerical libraries and most text books do not mention the Rook pivoting but explains the partial pivoting and the complete pivoting only.

In present study, we introduce an easy method to reduce the elapsed time of the LU decomposition calculation with complete pivoting.

Methods

The trivial method to make LU decomposition with complete pivoting for the size n matrix A stored in an array is, at the k-th step's iteration the pivot is searched which has the largest in magnitude in the area `A(k:n,k:n)`, and the columns and rows are exchanged appropriately to place the pivot element at the k-th diagonal, and then the k-th step's elimination is applied for the area `A(k+1:n,k+1:n)`. By this trivial method of calculation, in each step of the iteration the area of two dimensional array whose area is gradually shrinking is swept twice.

Our present method, by a simple coding modification, changes the order of the access to the storage of the array so that the process of updating the matrix elements by eliminations and the one to search the largest element in magnitude are merged together, which reduces the number of sweeps of the area `A(k+1:n,k+1:n)` in the k-th iteration from twice to only once.

Sample algorithms (For sequential codes)

```
!==============================================================================
! NO PIVOTING LU ALGORITHM in pseudo Fortran code.  Array A(N,N).
!==============================================================================
DO  K=1,N-1
  DO  J=K+1,N
    C = A(K,J)/A(K,K)
    DO  I=K+1,N; A(I,J) = A(I,J) - A(I,K) * C; ENDDO
  ENDDO
ENDDO
!==============================================================================
! PARTIAL PIVOTING LU ALGORITHM in pseudo code. Arrays: A(N,N),IPIV(N-1).
!==============================================================================
DO  K=1,N-1
  ! Find the pivot in the "K"-th column "A(K:N,K)".
  AMX = -1.0
  DO  I=K,N; IF(ABS(A(I,K)) > AMX) THEN;AMX=ABS(A(I,K));IPIV(K)=I;ENDIF;ENDDO
  ! Row exchange and elimination, column by column.
  IF(IPIV(K) /= K) THEN; EXCHANGE ( A(K,K), A(IPIV(K),K) ); ENDIF
  DO  J=K+1,N
    IF(IPIV(K) /= K) THEN; EXCHANGE ( A(K,J), A(IPIV(K),J) ); ENDIF
    C = A(K,J)/A(K,K); DO  I=K+1,N; A(I,J) = A(I,J) - A(I,K) * C; ENDDO
  ENDDO
ENDDO
!==============================================================================
! TRIVIAL COMPLETE PIVOTING LU ALGORITHM. Arrays: A(N,N),IPIV(K-1),JPIV(K-1).
!==============================================================================
DO  K=1,N-1
  ! Find the pivot from "A(K:N,K:N)".
  AMX = -1.0
  DO  J=K,N
    T = AMX  ! "T = -1.0" when calculation is parallel.
    DO  I=K,N; IF(ABS(A(I,J)) > T) THEN; T = ABS(A(I,J)); IP = I; ENDIF; ENDDO
    IF(T > AMX) THEN; AMX = T; IPIV(K) = IP; JPIV(K) = J; ENDIF
  ENDDO
  IF(JPIV(K) /= K) THEN; EXCHANGE ( A(1:N,K), A(1:N,JPIV(K)) ); ENDIF
  IF(IPIV(K) /= K) THEN; EXCHANGE ( A(K,K), A(IPIV(K),K) ); ENDIF
  DO  J=K+1,N
    IF(IPIV(K) /= K) THEN; EXCHANGE ( A(K,J), A(IPIV(K),J) ); ENDIF
    C = A(K,J)/A(K,K); DO  I=K+1,N; A(I,J) = A(I,J) - A(I,K) * C; ENDDO
  ENDDO
ENDDO
!=============================================================================
! PRESENT COMPLETE PIVOTING LU ALGORITHM. Arrays: A(N,N), IPIV(N), JPIV(N).
!=============================================================================
! Find the pivot in "A(1:N,1:N)".
AMX = -1.0
DO  J=1,N
  T = AMX ! "T = -1.0" when calculation is made parallel.
  DO  I=1,N; IF(ABS(I,J) > T) THEN; T = ABS(A(I,J)); IP = I; ENDIF; ENDDO
  IF(T > AMX) THEN; AMX = T; IPIV(1) = IP; JPIV(1)= J; ENDIF
ENDDO
DO  K=1,N-1
  IF(JPIV(K) /= K) THEN; EXCHANGE ( A(1:N,K), A(1:N,JPIV(K) ); ENDIF
  IF(IPIV(K) /= K) THEN; EXCHANGE ( A(K,K), A(IPIV(K),K) ); ENDIF
  AMX = -1.0
  DO  J=K+1,N
    IF(IPIV(K) /= K) THEN; EXCHANGE( A(K,J),A(IPIV(K),J) ); ENDIF
    C = A(K,J)/A(K,K)
    T = AMX ! "T = -1.0" when calculation is parallel.
    DO  I=K+1,N ! Fused element update and search.
      A(I,J) = A(I,J) - A(I,K) * C
      IF(ABS(A(I,J)) > T) THEN; T = ABS(A(I,J)); IP = I; ENDIF
    ENDDO
    IF(T > AMX) THEN; AMX = T; IPIV(K+1) = IP; JPIV(K+1) = J; ENDIF
  ENDDO
ENDDO
```

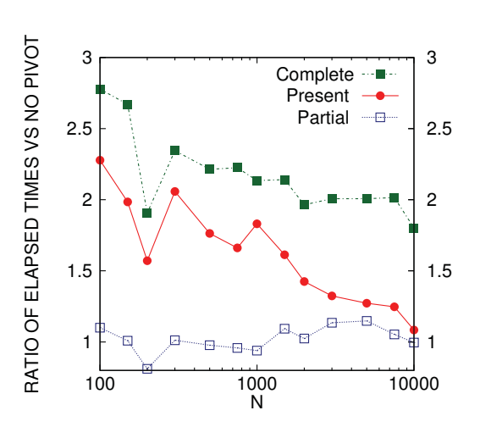
Experiments

We made experiments by using the following four systems:

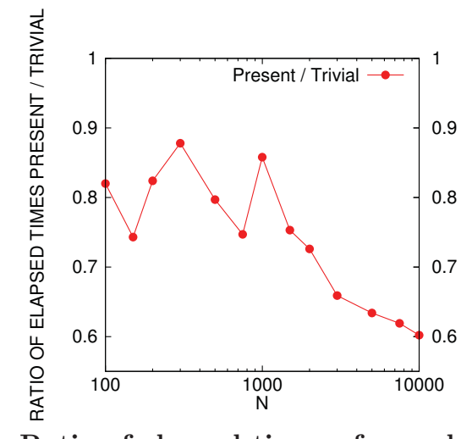
1. PC: Virtual machine (VMWare on Windows 11)
   CPU:intel Corei5-6400(2.7GHz), Allocated MEM:16GB.
2. Server: Dual CPU node (intel Xeon Platinum 8368),
   38 cores/CPU(16 Pcores(2.5GHz),22 Ecores(2.3GHz)),
   L3 cache=57MB, DDR4-3200 MEM 256GB/node. 8 MEM channels/CPU.
3. Vector Engine: NEC SX-Aurola TSUBASA Type20A (10 Cores, MEM 48GB).
   Vector Host: AMD EPIC 7402P(2.8GHz,24Cores) 1CPU, MEM 128GB.
4. GPU: NVIDIA A100 (MEM 80GB).
   GPU Host: A dual CPU node (intel Xeon Platinum 8368),
   38 cores/CPU(16 Pcores(2.5GHz),22 Ecores(2.3GHz)),
   L3 cache=57MB, DDR4-3200 MEM 512GB/node. 8 MEM channels/CPU.

Experiment 1 (PC, 1 Core, No Parallel)

- PC: A virtual machine (VMWare on Windows 11)
  CPU:intel Corei5-6400(2.7GHz), Allocated memory: 16GB. OS: AlmaLinux-9.
- Measured elapsed times (in sec) for LU decompositions.
- Compiler: Intel ifort ver 2021.10
  Compiler flags:`-Ofast -fma -align array256byte -xCORE-AVX2`
- FP64, Single thread (1 Core), No Parallel.



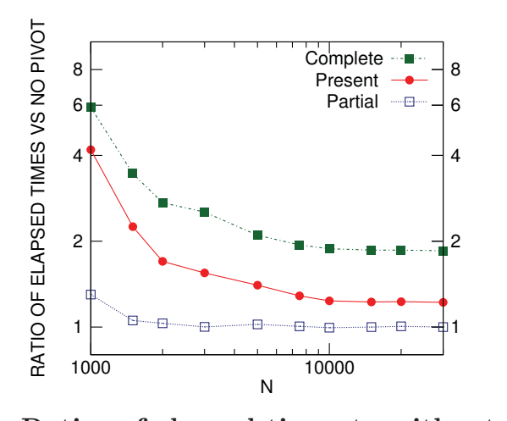
Ratios of elapsed times to the one without pivoting (PC, 1 thread)



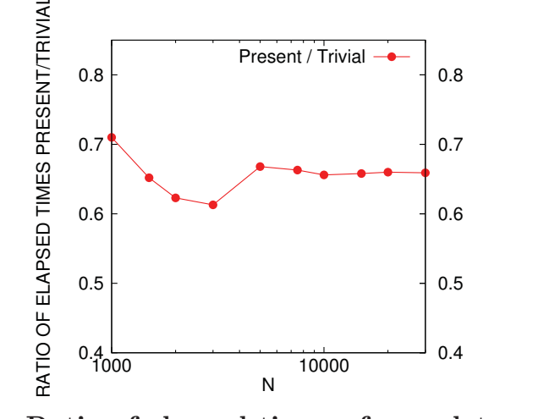
Ratio of elapsed times of complete pivotings (PC, 1 thread)

Experiment 2 (Server, OpenMP Parallel)

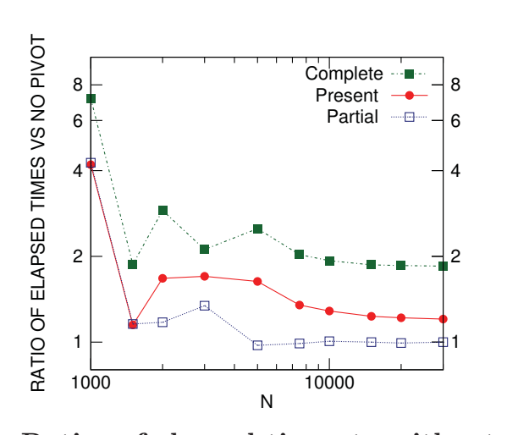
- Server: A dual CPU node(intel Xeon Platinum 8368) 38 cores/CPU (16 Pcores, 22 Ecores) L3=57MB, DDR4 3200 MEM 256GB/node, 8 MEM channels.
- Measured elapsed times (in sec) for LU decompositions.
- Compiler: intel ifort ver. 2021.10
  Compiler flags:`-Ofast -fma -align array256byte -qopenmp -xCORE-AVX2`
- FP64, OMP parallel (multi-core calculation)

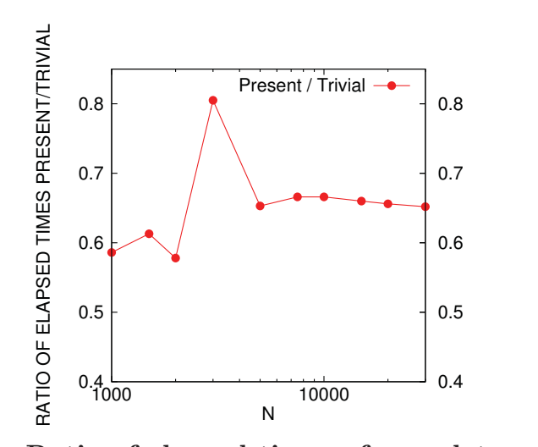


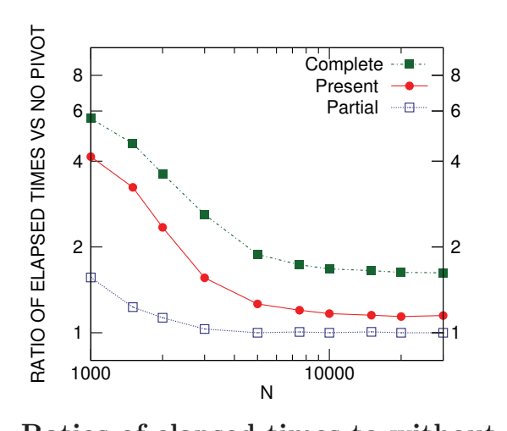
Ratios of elapsed times to without pivoting (Server, 8 threads, 1CPU)

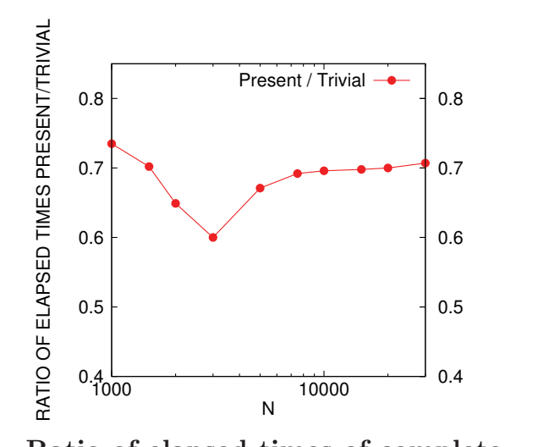


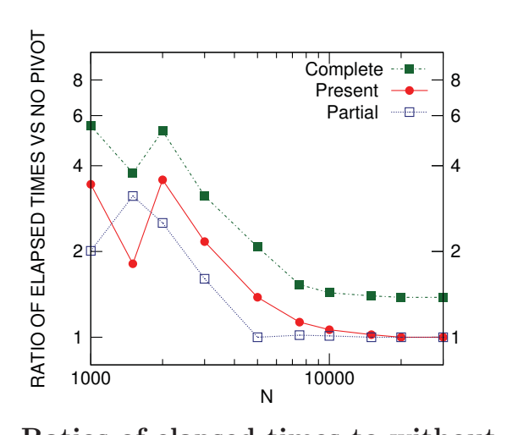
Ratio of elapsed times of complete pivotings (Server, 8 threads, 1CPU)

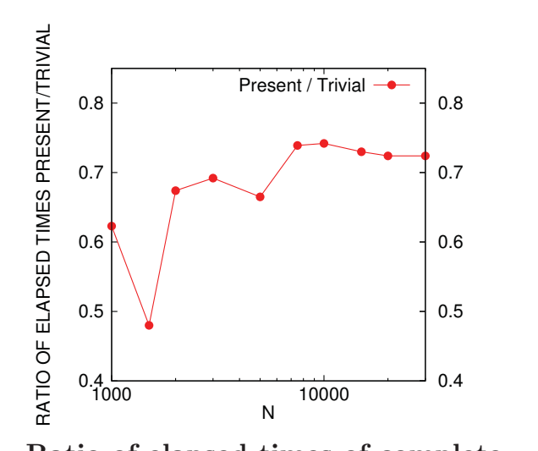

Ratios of elapsed times to without pivoting (Server, 8 threads, 2CPU)

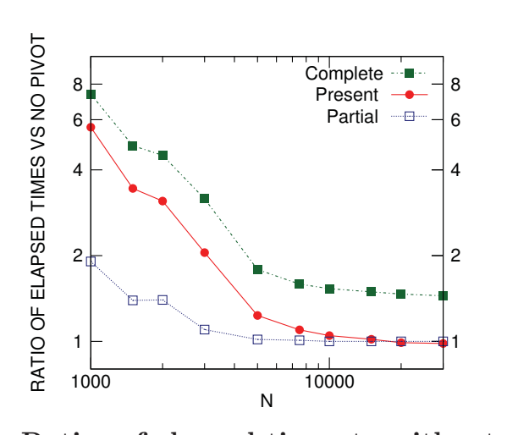

Ratio of elapsed times of complete pivotings (Server, 8 threads, 2CPU)

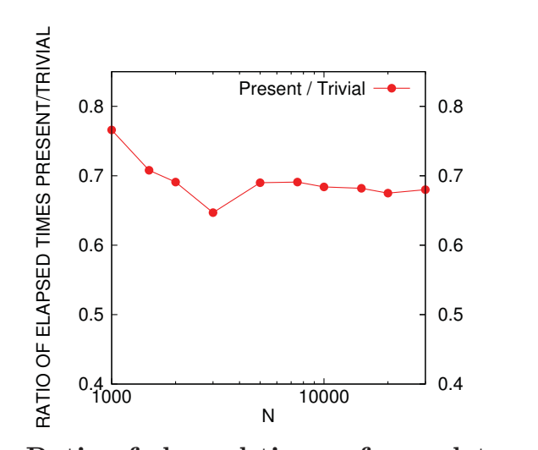

Ratios of elapsed times to without pivoting (Server, 16 threads, 1CPU)

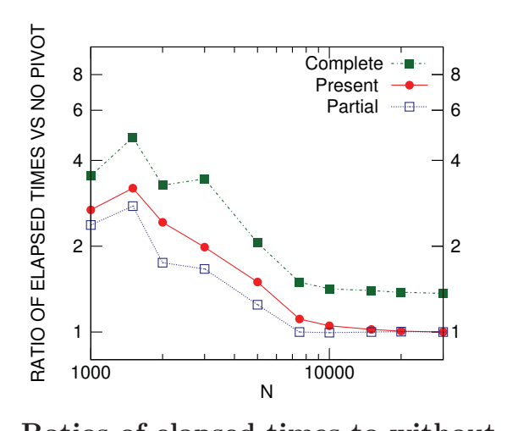

Ratio of elapsed times of complete pivotings (Server, 16 threads, 1CPU)

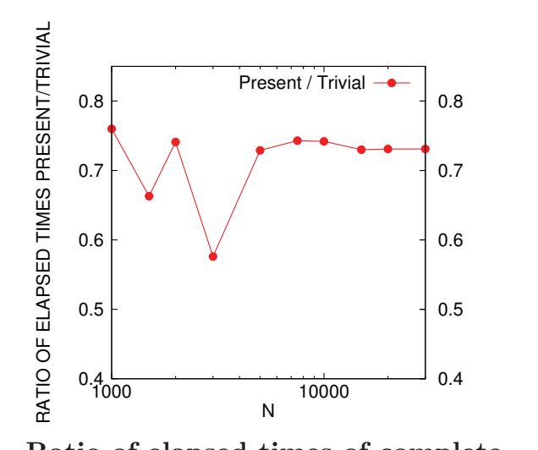

Ratios of elapsed times to without pivoting (Server, 32 threads, 2CPU)

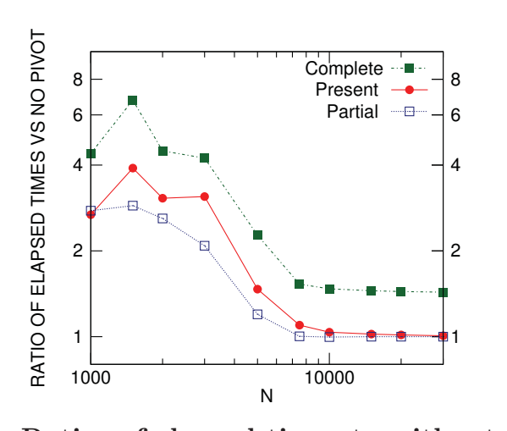

Ratio of elapsed times of complete pivotings (Server, 32 threads, 2CPU)

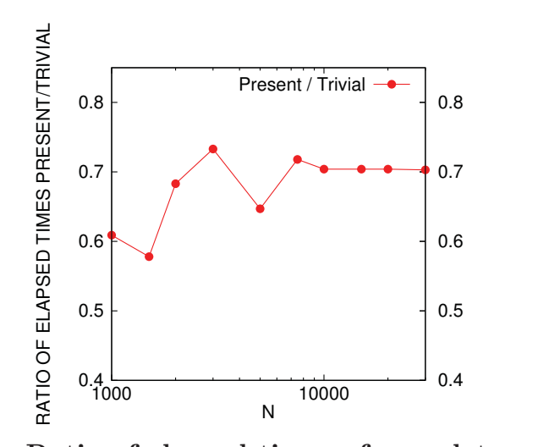

Ratios of elapsed times to without pivoting (Server, 38 threads, 1CPU)


Ratio of elapsed times of complete pivotings (Server 38 threads, 1CPU)


Ratios of elapsed times to without pivoting (Server, 38 threads, 2CPU)

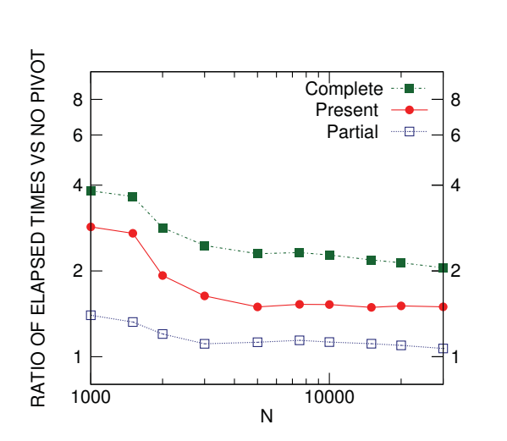

Ratio of elapsed times of complete pivotings (Server, 38 threads, 2CPU)

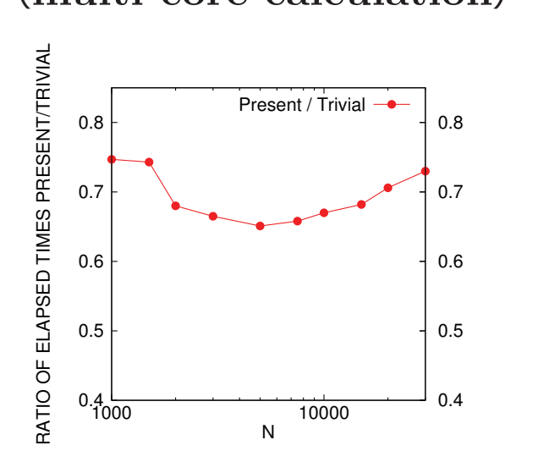

Ratios of elapsed times to without pivoting (Server, 76 threads, 2CPU)

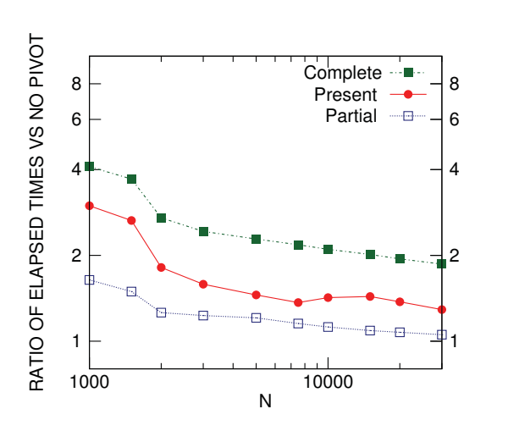

Ratio of elapsed times of complete pivotings (Server, 76 threads, 2CPU)

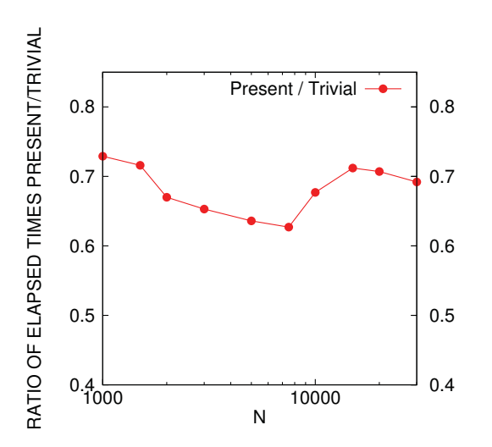
Experiment 3 (NEC Vector Engine, OpenMP Parallel)

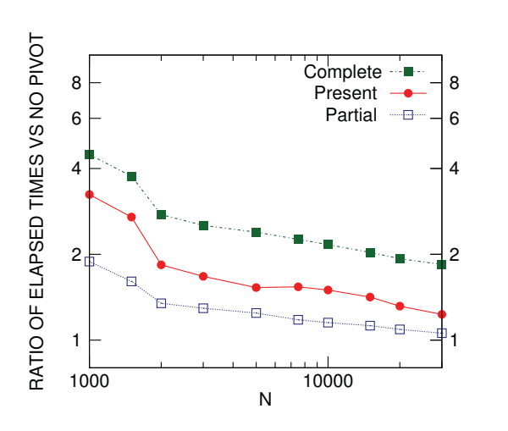
- (Vctor Engine) NEC-SX Aurola TSUBASA Type 20A: 10 cores / MEM 48GB.
  (Vctor Host) AMD EPIC 7402P (2.8GHz, 28Cores) Main Mem 128GB.
- Measured elapsed times (in sec) for LU decompositions.
- Compiler: NEC nfort (version 5.3.0);
  Compiler flags : `-O3 -fopenmp`
- FP64, OMP parallel (multi-core calculation)

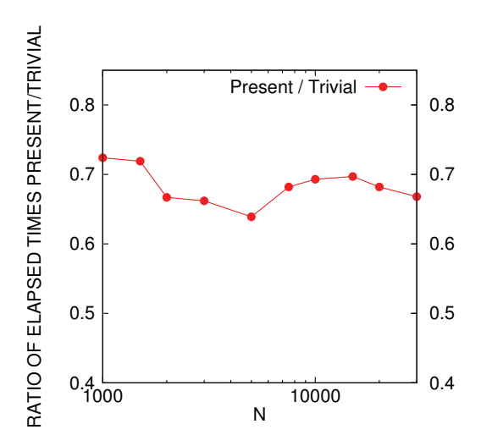

Ratios of elapsed times to the one without pivoting (VE, 4 threads)

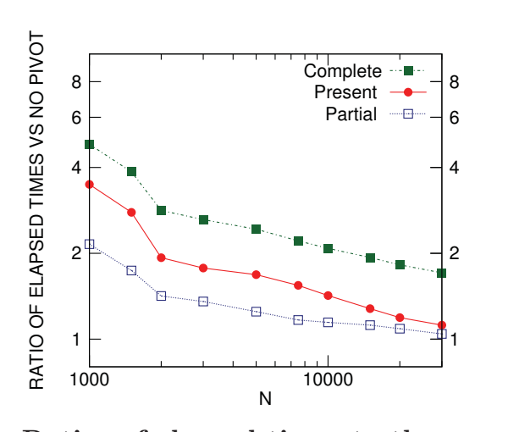

Ratio of elapsed times of complete pivotings (VE, 4 threads)

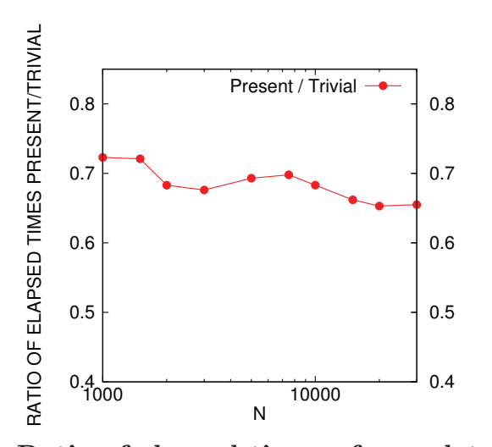

Ratios of elapsed times to the one without pivoting (VE, 6 threads)


Ratio of elapsed times of complete pivotings (VE, 6 threads)


Ratios of elapsed times to the one without pivoting (VE, 8 threads)

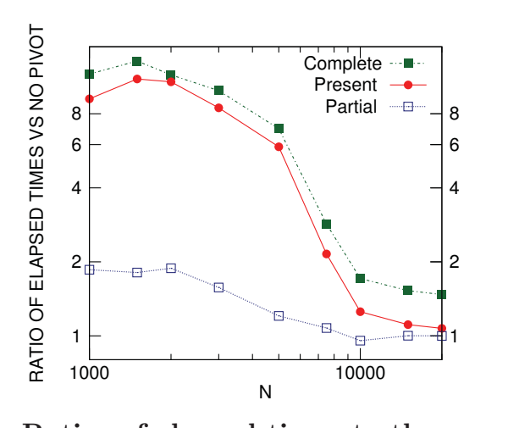

Ratio of elapsed times of complete pivotings (VE, 8 threads)

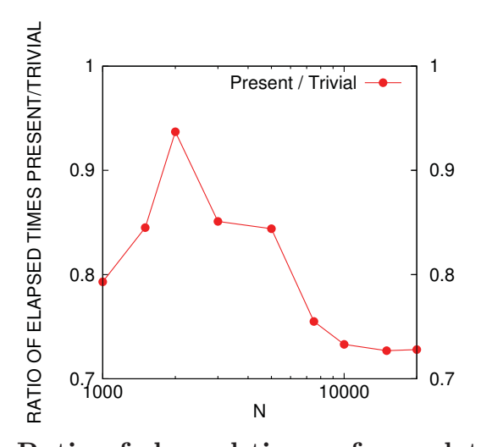

Ratios of elapsed times to the one without pivoting (VE, 10 threads)


Ratio of elapsed times of complete pivotings (VE, 10 threads)

Experiment 4 (GPU, OpenMP Parallel)

- GPU: NVIDIA A100 (MEM 80GB).
  CPU: intel Xeon Platinum 8368(2.4GHz, 38Cores) Dual; MEM 512GB.
- Measured elapsed times (in sec) for LU decompositions.
- Compiler: nvfort version 25.1-0
  Compiler flags : `-fast -O3 -stdpar=gpu -mp=gpu -gpu=mem:managed`
- FP32, OpenMP Parallel.


Ratios of elapsed times to the one without pivoting (GPU, 54 threads)


Ratio of elapsed times of complete pivotings (GPU, 54 threads)

Graphs of the experiments show that compared with the trivial approach, our present approach made the elapsed time to

- between 0.60 to 0.88 times for the PC (No Parallel),
- between 0.58 to 0.73 times for the Server (OpenMP 76 threads),
- between 0.65 to 0.72 times for the Vector Engine (OpenMP 10 threads),
- between 0.72 to 0.95 times for the GPU (OpenMP 54 threads).

When the matrix size is large and the number of threads is increased, the memory transfer tends to become the bottle-neck of the calculation. Under such situation, the elapsed time of the present LU decomposition method with complete pivoting is nearly equivalent to that of methods without pivoting or with partial pivoting.

Conclusions

In the k-th iteration of the LU decomposition, by a simple code modifications, we merged the process to eliminate the matrix elements `A(k+1:n,k+1:n)` and the process to make the complete pivoting which searches the largest matrix element in magnitude in the area `A(k+1:n,k+1:n)`, which reduced the amount of storage transfer of the array for the matrix A, and the elapsed time for calculation of the LU decomposition with complete pivoting was also reduced.

REFERENCES

[1] George Poole, Larry Neal: "The Rook's Pivoting Strategy", *J. Comput. Appl. Math.*, Vol.123, No.1-2 (2000), pp.353–369.

[2] Xiao-Wen Chang: "Some Features of Gaussian Elimination with Rook Pivoting", *BIT*, Vol.42, No.1, pp.066-083 (2002).

[3] Gene Howard Golub and Chales F. Van Loan: "Matrix Computations", 4th Ed., The John Hopkins Univ. Press (2013). in §3.4: 'Pivoting'.