Accelerated Mixed-Precision Algorithms for Hermitian Eigenvalue Problem

Yaohung (Mike) TSAI
Piotr LUSZCZEK
Jack DONGARRA
Innovative Computing Laboratory
University of Tennessee Knoxville

ABSTRACT

As the new hardware is being equipped with powerful low-precision capabilities driven primarily by the needs of the burgeoning field of Artificial Intelligence (AI), mixed-precision algorithms are now showing far greater potential and renewed interest in scientific computing community. The multi-precision methods commonly follow approximate-iterate scheme by first obtaining the approximate solution from a low-precision factorization and solve. Then, they iteratively refine the solution to the desired accuracy that is often as high as what is possible with traditional approaches. While targeting symmetric and Hermitian eigenvalue problems of the form $Ax=\lambda x$, we revisit the SICE algorithm by applying the Sherman-Morrison formula on the diagonally-shifted tridiagonal systems, we propose an updated SICE-SM algorithm. We exploited asynchronous scheduling techniques to take advantage of the new computational graph enabled by the use of mixed-precision in the eigensolver. This allowed us to hide some of the extra work required by mixed-precision algorithm behind the tasks required by the classic eigensolver and thus minimize the overheads resulting from introducing extra floating-point precision data and compute. This allowed us to maximize the efficiency on both multicore CPU as well as on GPU-accelerated platforms. By incorporating the latest two-stage algorithms from the PLASMA and MAGMA software libraries for numerical linear algebra, we achieved up to 3.6x speedup using the mixed-precision eigensolver with the blocked SICE-SM algorithm for iterative refinement when compared with full double complex precision solvers for the cases with a portion of eigenvalues and eigenvectors requested.

Algorithm 1 SICE algorithm

Input: Matrix $A \in \mathbb{R}^{n\times n}$. An approximate eigenvalue λ and the corresponding eigenvector x. $I_{\max}$ denotes the maximum number of iterations.
Output: Refined eigenvalue λ and its eigenvector x.
function $[\lambda, x] \leftarrow$ **SICE**$(A, \lambda, x, I_{\max})$
  $[Q, U] \leftarrow$ schur(A) ▸ obtain Schur decomposition $A = QUQ^T$, $QQ^T = I$.
  $[m, s] \leftarrow max(abs(x)); x \leftarrow x/m$ ▸ Normalizing x so that $\|x\|_\infty = s_y = 1$.
  for i in $1 : I_{\max}$ **do**
    $r \leftarrow \lambda x - Ax$
    $c \leftarrow -x - a_{\lambda s}$
    $d \leftarrow Q^T c$
    $f^T \leftarrow Q(s,:) = e_s^T Q$ ▸ s-th row of Q.
    $\hat{U}_1 \leftarrow Q_1(U - \lambda I)_d$ ▸ Givens rotations Q_1 from $Q_1 d = (P_2 P_3 \dots P_n)d = \gamma e_1$ where $\gamma = \|d\|_2$
    $\hat{U}_2 \leftarrow \hat{U}_1 * d(1)f^T$
    $\hat{U}_3 \leftarrow Q_2 \hat{U}_2$ ▸ Givens rotations Q_2 to introduce upper triangular form.
    Solve the triangular system $\hat{U}_3 z = Q_2 Q_1 Q^T r$
    $y \leftarrow Qy$
    $\lambda \leftarrow \lambda + y(s)$ ▸ Update eigenvalue.
    $y(s) \leftarrow 0$ ▸ Set $y(s)$ to 0.
    $x \leftarrow x + y$ ▸ Update eigenvector.
    if desired accuracy is reached **then**
      break
    end if
  end for
end function

Algorithm 2 SICE-SM algorithm: SICE algorithm with Sherman–Morrison formula

Input: Matrix $A = A^\dagger \in \mathbb{R}^{n\times n}$. An approximate eigenvalue λ and the corresponding eigenvector x. $I_{\max}$ denotes the maximum number of iterations.
Output: Refined eigenvalue λ and eigenvector x.
function $[\lambda, x] \leftarrow$ **SICE_SM**$(A, \lambda, x, I_{\max})$
  $[Q, T] \leftarrow$ tridiag(A) ▸ Tridiagonalization $A = QTQ^T$, $QQ^T = I$.
  $[m, s] \leftarrow max(abs(x)); x \leftarrow x/m$ ▸ Normalization of x so that $\|x\|_\infty = s_y = 1$.
  for i in $1 : I_{\max}$ **do**
    $r \leftarrow \lambda x - Ax$
    $c \leftarrow -x - a_{\lambda s}$
    $d \leftarrow Q^T c$
    $f^T \leftarrow Q(s,:) = e_s^T Q$ ▸ s-th row of Q.
    $rhs \leftarrow Q^T r$
    $u \leftarrow (T - \lambda I)^{-1} d$
    $v \leftarrow (T - \lambda I)^{-1} rhs$
    $y \leftarrow v - \frac{f^T v}{1+f^T u} u$ ▸ Sherman–Morrison formula
    $y \leftarrow Qy$
    $\lambda \leftarrow \lambda + y(s)$ ▸ Update eigenvalue.
    if $i \neq 1$ **then**
      $y(s) \leftarrow 0$ ▸ Set $y(s)$ to 0.
      $x \leftarrow x + y$ ▸ Update eigenvector.
    end if
    if desired accuracy reached **then**
      break
    end if
  end for
end function

Algorithm 3 Blocked SICE-SM algorithm

Input: $A = A^T \in \mathbb{R}^{n\times n}$, initial eigenvectors $X = [x_1|x_2|...|x_p] \in \mathbb{R}^{n\times r}$ and the corresponding initial eigenvalues $\Lambda = (\lambda_1, \lambda_2, \dots, \lambda_r)^T \in \mathbb{R}^l$. $I_{\max}$ denotes the maximum number of iterations.
Output: Refined eigenvectors X and refined eigenvalues Λ.
function $[X, \Lambda] \leftarrow$ **SICE_SM_BLK**$(A, X, \Lambda, I_{\max})$
  $[Q, T] \leftarrow$ tridiag(A) ▸ Tridiagonalization $A = QTQ^T$, $QQ^T = I$.
  for i in $1 : I_{\max}$ **do**
    $s \leftarrow i$
    $R \leftarrow X \times$ diag _matrix$(\Lambda) - A \times X$ ▸ Residual vectors need higher precision.
    for j in $1 : r$ **do**
      $c_j \leftarrow -x_j - A(:,s)$
    end for
    Compose matrix $C = [c_1|c_2|...|c_r]$ from column vectors c_j
    $C(s,:) \leftarrow C(s,:) + \Lambda^T$
    $D = [d_1|d_2|...|d_r] \leftarrow Q^T \times C$ ▸ Can be in lower precision.
    $RHS = [rhs_1|rhs_2|...|rhs_l] \leftarrow Q^T \times R$ ▸ Can be in lower precision.
    $f \leftarrow Q(s,:)$ ▸ s-th row of Q.
    for j in $1 : r$ **do**
      $u_i \leftarrow (T - \lambda I)^{-1} d_i$
      $v_i \leftarrow (T - \lambda I)^{-1} rhs_i$
      $y_i \leftarrow v_i - \frac{f^T v_i}{1+f^T u_i} u_i$ ▸ Sherman–Morrison
    end for
    Compose matrix $Y = [y_1|y_2|...|y_r]$ from correction vectors y_j
    $Y \leftarrow Q \times Y$
    $\Lambda \leftarrow \Lambda + Y(s,:)^T$ ▸ Update eigenvalues.
    if $i \neq 1$ **then**
      $Y(s,:) \leftarrow 0$ ▸ Set $y_j(s)$ to 0.
      $X \leftarrow X + Y$ ▸ Update eigenvectors.
      Normalize eigenvectors x_i in X.
    end if
    if desired accuracy reached **then**
      break
    end if
  end for
  $X \leftarrow X * \frac{1}{2}X(I - X^T X)$ ▸ Orthogonalization.
end function

INTRODUCTION

SICE (Subroutine for Improving Computed Eigevalues) algorithm [1]–[3] improves accuracy of Hermitian (or symmetric) eigenvalue problem. To improve its efficiency on modern platforms with hardware accelerators we introduced two new parallel agorithms and exploited the scheduling

ORIGINAL SICE ALGORITHM

SICE is the classic refinement algorithm we implemented to profile it on modern hardware with contemporary software stack including two-stage eigenvalue solver that was not available at the time of SICE's introduction. The main performance downside of SICE algorithm is the fact that it is based on element-wise and vector-wise operations. In other words, it relies on BLAS Level 1 and 2 which is highly inefficient on current systems with hardware accelerators. The problem is exacerbated by the path that these low-performance operations are on the critical path of the algorithm and their overhead increases with the increasing problem size.

NEW SICE ALGORITHM

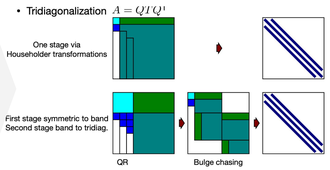
SICE-SM is our new parallel algorithm that overcomes the main bottlenecks of the original SICE algorithm by introducing Sherman-Morrison formula into the theoretical formulation. This allowed to lift some operations from lower-level BLAS and improve data reuse for better utilization of the memory hierarchy and accelerators' compute units.

NEW BLOCKED SICE-SM ALGORITHM

Blocked SICE-SM is our new parallel algorithm that further improves the cache-blocking opportunities that target the fastest levels of the memory hierarchy such as CPU caches, register files, and GPU shared memory. Better compute efficiency improves performance of the new mixed-precision tasks while the asynchronous scheduling continues to exploit independent operations for parallel execution and hiding less efficient tasks behind the compute-intensive ones for maximal overall utilization.

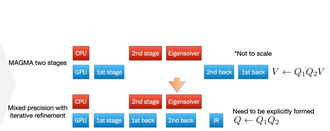
ONE- & TWO-STAGE EIGENVALUE SOLVER

The classic eigenvalue problem algorithm was one-stage (as shown at the top): reduction to tridiagonal form was performed by a single computational stage without intermediate steps. SBR (Successive Band Reduction) Toolbox allowed multiple stage with each one reducing bandwidth of the original matrix but those multiple stage increased back-transformation cost and computing eigenvectors was possible but not impractical. Hence, two stage algorithm (shown at the bottom) struck a balance by only admitting two computational stages: reduction from full to band form and reduction from band to tridiagonal. The computational overhead of transformation was twice as large but it still could outperform the classic single-stage algorithm due to its parallel and compute intensive implementation not possible for single-stage algorithm.



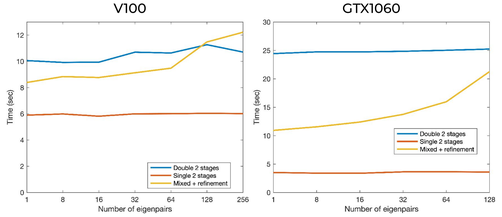
DAG OF COMPUTE TASKS

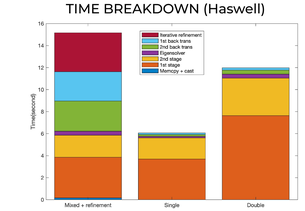
The compute DAG (Direct Acyclic Graph) shown in the figure is a simplified representation that underscores the main algorithmic components of the methods we proposed. The asynchrony of compute tasks is fully exploited and the hardware is presented with a mix of workloads that vary in compute intensity and reliance on the memory hierarchy and thus they use the hardware more efficiently.

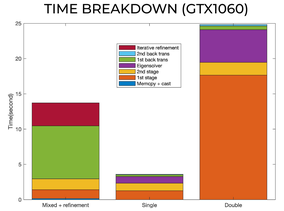


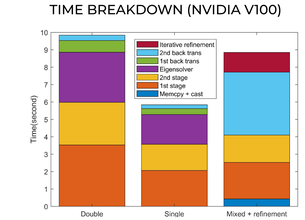
PERFORMANCE FOR VARYING COUNTS OF EIGENPAIRS

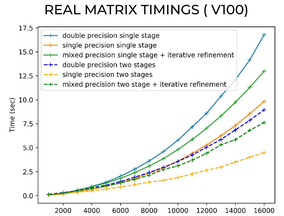
The performance figures below show the behavior of our HPC implementations of two accelerated platforms that differ in their mixed-precision compute balance: 2x and 32x, respectively.

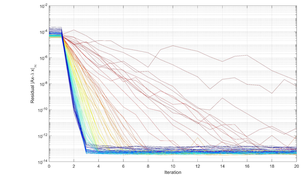


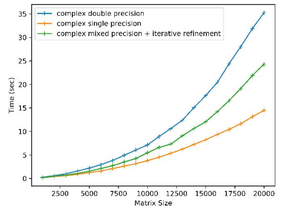






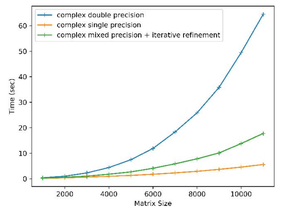






CONVERGENCE ACROSS THE EIGENSPECTRUM

Convergence graphs represent the convergence rates across the eigen spectrum of a small matrix with entries from a uniform random distribution. The largest eigenvalues experience very rapid convergence, that deteriorates for the smaller ones further down the spectrum. This is the main potential for further extension to leverage eigen spectrum shifting to level up the convergence in any portion of the eigenspectrum.



FOR MORE INFORMATION

https://icl.utk.edu/~luszczek/sc22/pstr_mxp_eig_sym/

ACKNOWLEDGEMENTS

This material is also based upon work supported by the National Science Foundation under OAC Grant No.-2004541 and the University of Tennessee grant MSE E01-1315-028 as Interdisciplinary Seed funding. This research used the computational resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No.-DE-AC05-00OR22725 provided by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The software library integration work was supported by the National Science Foundation under OAC Grant No.-2004541.