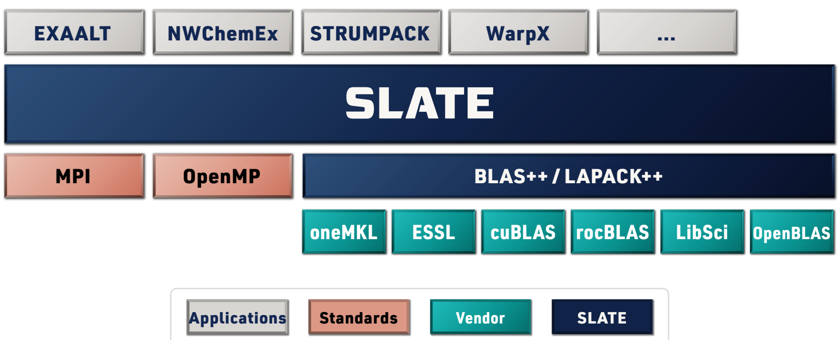
SLATE

FIND OUT MORE AT **https://icl.utk.edu/slate**

REPLACING ScaLAPACK

The objective of the Software for Linear Algebra Targeting Exascale (SLATE) project is to provide fundamental dense linear algebra capabilities to the US Department of Energy and to the high-performance computing (HPC) community at large. To this end, SLATE provides basic dense matrix operations (e.g., matrix multiplication, rank-k update, triangular solve), norms, linear systems solvers, least square solvers, and singular value and eigenvalue solvers.



The ultimate objective of SLATE is to replace the venerable Scalable Linear Algebra PACKage (ScaLAPACK) library, which has been the industry standard for dense linear algebra operations in distributed-memory environments. However, after two decades of operation, ScaLAPACK is past the end of its lifecycle and overdue for a replacement, as it can hardly be retrofitted to support hardware accelerators, which are an integral part of today's HPC hardware infrastructure.

Primarily, SLATE aims to extract the full performance potential and maximum scalability from modern, many-node HPC machines with large numbers of cores and multiple hardware accelerators per node. For typical dense linear algebra workloads, this means getting close to the theoretical peak performance and scaling to the full size of the machine (i.e., thousands to tens of thousands of nodes). This is accomplished in a portable manner by relying on MPI, OpenMP, and C++ standards.

SLATE *HIGHLIGHTS*

Targets Modern Hardware

such as emerging exascale systems, where the number of nodes is large, and each node contains a heavyweight multi-core processor and a number of heavyweight hardware accelerators.

Guarantees Portability

by relying on standard computational components (e.g., vendor implementations of BLAS and LAPACK) and standard parallel programming technologies (e.g., MPI, OpenMP) or portable runtime systems (e.g., PaRSEC).

Provides Scalability

by employing proven techniques of dense linear algebra, such as the 2-D block cyclic data distribution, as well as modern parallel programming approaches, like dynamic scheduling and communication overlapping.

Facilitates Productivity

by relying on the intuitive single program, multiple data (SPMD) programming model and a set of simple abstractions to represent dense matrices and dense matrix operations.

Ensures Maintainability

by employing useful facilities of the C++ language, such as templates and overloading of functions and operators, and focused on minimizing code bloat by relying on compact representations.

PAPERS AND WORKING NOTES

https://icl.utk.edu/slate/#papers

Gates, M., A. YarKhan, D. Sukkari, K. Akbudak, S. Cayrols, D. Bielich, A. Abdelfattah, M. Al Farhan, and J. Dongarra
Portable and Efficient Dense Linear Algebra in the Beginning of the Exascale Era
In *2022 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pp. 36-46. IEEE, 2022.

Gates, M., A. Abdelfattah, K. Akbudak, M. Al Farhan, R. Alomairy, D. Bielich, T. Burgess, S. Cayrols, N. Lindquist, D. Sukkari, and A. YarKhan
Evolution of the SLATE linear algebra library
The International Journal of High Performance Computing Applications, September 2024. DOI: 10.1177/10943420241286531

Sukkari, D., M. Gates, M. Al Farhan, H. Anzt, and J. Dongarra
Task-Based Polar Decomposition Using SLATE on Massively Parallel Systems with Hardware Accelerators
SC-W '23: Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, Denver, CO, ACM, November 2023. DOI: 10.1145/3624062.3624248

SPONSORED BY

U.S. DEPARTMENT OF ENERGY | Office of Science
ECP
NNSA National Nuclear Security Administration
NSF National Science Foundation

SLATE

FIND OUT MORE AT https://icl.utk.edu/slate

SLATE COVERAGE

BASIC LINEAR ALGEBRA ($C = AB$, ...)

	ScaLAPACK	SLATE
Level 1 PBLAS	✓	
Level 2 PBLAS	✓	✓ partial
Level 3 PBLAS	✓	✓
Auxiliary routines (add, set, scale, ...)	✓	✓
Matrix norms	✓	✓
Test matrix generation	✓	✓ refactored lib

LINEAR SYSTEMS ($Ax = b$)

	ScaLAPACK	SLATE
LU (partial pivoting)	✓	✓
CALU (tournament pivoting)		✓ panel on GPU
LU, band (pp)	✓	✓
LU (non-pivoting)		✓
LU Random Butterfly (RBT)		✓
LU BEAM solver		✓ dev branch
Cholesky	✓	✓ panel on GPU
Cholesky, band	✓	✓
Symmetric Indefinite (block Aasen)		✓ CPU only
Mixed precision (single-double)		✓ IR, GMRES-IR
Inverses (LU, Cholesky)	✓	✓
Condition estimate	✓	✓

LEAST SQUARES ($Ax \cong b$)

	ScaLAPACK	SLATE
QR	✓	✓ panel on GPU
Cholesky QR		✓
LQ	✓	✓
Least squares solver	✓	✓ QR, CholQR
PAQR (pivoting avoiding)		✓ dev branch

SVD, EIG, PD ($A = U\Sigma V^H$, $Ax = \lambda x$, $A = UH$)

	ScaLAPACK	SLATE
Singular value decomposition (SVD)	✓	✓ values & vectors
Hermitian eigenvalue	✓	✓ values & vectors
Generalized Hermitian eigenvalue	✓	✓ values & vectors
Polar decomposition (QDWH)		✓ dev branch
LOBPCG		✓ dev branch
Non-symmetric eigenvalue		proposed

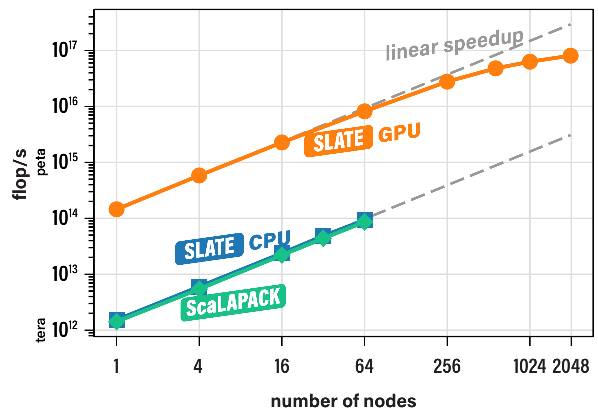
PERFORMANCE

The SLATE GPU-accelerated version attains up to 100× speedup over the CPU.

Weak scaling, from 1 to 2048 nodes, totaling 8 to 16384 AMD MI250X GPU GCDs. CPU-only runs were done from 1 to 64 nodes, totaling 56 to 3584 AMD EPYC CPU cores.

Weak scaling for matrix multiply (gemm)

on Frontier for a local problem size of 80,000 × 80,000 per node

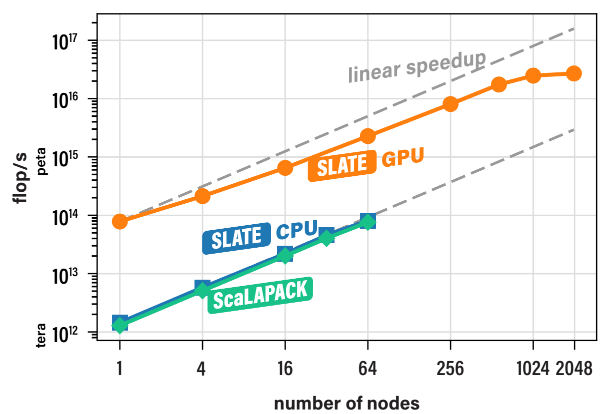


SLATE on GPUs achieves 94.7× to 107.2× speedup compared to the CPU. Performance is falling off the linear speedup line for a larger number of nodes.

SLATE on CPUs is 7.2% to 10.5% faster than ScaLAPACK; the lines appear almost coincident at this scale. This closeness is expected since both are a significant fraction of the CPU peak. On 64 nodes, SLATE CPU gemm achieves 80.7% of peak at 92.6 Tflop/s, and ScaLAPACK achieves 75.2% of peak at 86.2 Tflop/s.

Weak scaling of Cholesky (potrf)

on Frontier for a local problem size of 140,000 × 140,000 per node



The SLATE GPU-accelerated version attains 60.8× speedup over the CPU on 1 node (8 GPU GCDs), falling to 29.9× on 64 nodes (512 GPU GCDs).

SLATE CPU is 6.8% to 13.4% faster than ScaLAPACK. These are not as fast as gemm; on 64 nodes, SLATE CPU Cholesky achieves 81.6 Tflop/s, 71% of peak, and ScaLAPACK achieves 76.4 Tflop/s, 66.6% of peak.

SPONSORED BY

U.S. DEPARTMENT OF ENERGY Office of Science

ECP

NNSA National Nuclear Security Administration

NSF National Science Foundation