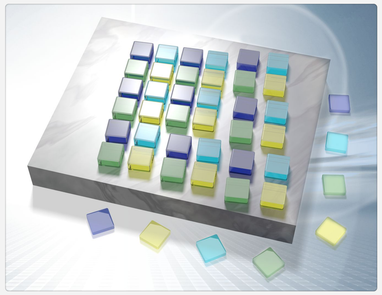
SLATE: Developing Sustainable Linear Algebra Software for Exascale

SOFTWARE DEVELOPMENT PRACTICES FOR THE NEXT GENERATION OF SUPERCOMPUTERS



SLATE will be a modern replacement for ScaLAPACK.

The SLATE software package will be a replacement for the LAPACK and ScaLAPACK numerical libraries, which over the last two decades set the bar for quality, sustainability, and community engagement. Improving upon the accomplishments of these legacy packages in those capacities is one of the major objectives of the SLATE project.

SLATE will accomplish these improvements by applying contemporary software engineering techniques in hosting, development, documentation, automated testing, and continuous integration, creating communication channels with academic and commercial application teams to actively engage the larger HPC community, and using modern tools that enable these practices and techniques.

SLATE OBJECTIVES

AGILITY	Weekly meetings, open doors, Slack channels
CONTINUOUS INTEGRATION	Jenkins builds all libraries for every check-in
CONTINUOUS DELIVERY	Code repository is public on Bitbucket
AUTOMATED TESTING	Jenkins tests run after every build
MONITORING	Build and test results delivered via email and Slack
CORRECTNESS	Test against ScaLAPACK, LAPACKE, CBLAS
COMMUNICATION	Public user group, Slack channel, Twitter

COMPATIBILITY APIS

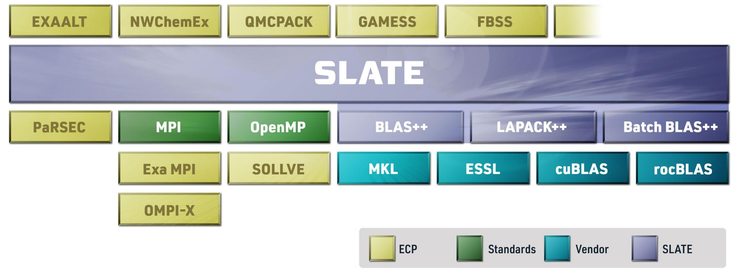
LAPACK COMPATIBILITY API

LAPACK function signatures, "slate_" prefix, e.g., slate_dgetrf(M, N, A, LDA, IPIV, INFO).
Additional settings through environment variables, e.g., "export LAPACK_NB=256".

SCALAPACK COMPATIBILITY API

ScaLAPACK function names and signatures, i.e., no changes to the source code required (link time replacement).

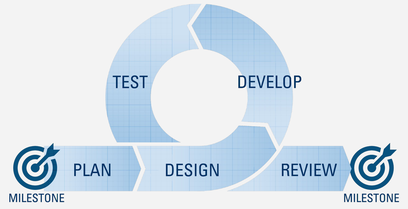
SLATE IN THE ECP SOFTWARE STACK



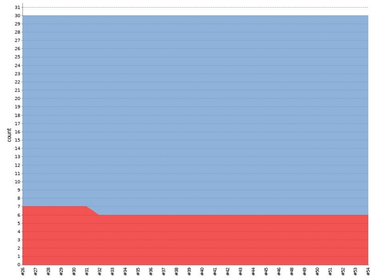
AGILE METHODOLOGY

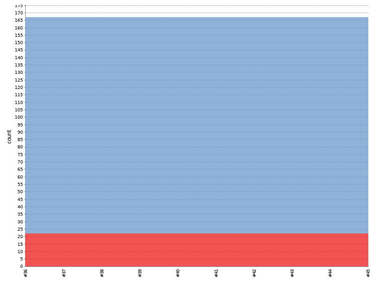
The SLATE team uses an extended scrum style methodology. Planning sessions are typically held over the first week of each milestone, as a team and in pairs. Features are planned as function groups, such as parallel BLAS operations, linear systems, or least squares.

These are broken into tasks which are tracked on a Trello board, where **To Do**, **Doing**, **Ongoing**, and **Done** items are held. Over the course of the milestones, smaller iterations are used to break down larger tasks, i.e., into individual functions. Weekly meetings are used in place of the daily scrum meeting to minimize distractions.



SLATE TESTING TRENDS





SLATE TOOLS

SLATE TEST REPORTING EXAMPLE

slate_suite.gbtrf

45 tests failed

Standard Output

```
input: ./test gbtrf --nb 10,100 --ref n --type s,d,c,z --lookahead 1 --dim 100:500:100 --kl 20,100 --ku 20,100
MPI size 1, OpenMP threads 4
```

								look	panel		SLATE			
target	type	m	n	kl	ku	nrhs	nb	ib	p	q	ahead	threads	error	time (s)	status
t	s	100	100	20	20	10	10	16	1	1	1	1	3.21e-10	0.01088	pass
t	s	200	200	20	20	10	100	16	1	1	1	1	1.81e-10	0.008226	pass
t	s	200	200	100	20	10	10	16	1	1	1	1	1.72e-05	0.01176	FAILED
t	s	200	200	100	20	10	100	16	1	1	1	1	2.38e-10	0.008649	pass
t	s	300	300	20	100	10	10	16	1	1	1	1	4.38e-05	0.007928	FAILED
t	s	300	300	20	100	10	100	16	1	1	1	1	7.27e-11	0.008942	pass
t	s	300	300	100	100	10	10	16	1	1	1	1	6.22e-05	0.04409	FAILED
t	s	300	300	100	100	10	100	16	1	1	1	1	2.03e-10	0.008178	pass
t	s	400	400	20	20	10	10	16	1	1	1	1	8.06e-11	0.008256	pass
t	s	400	400	20	20	10	100	16	1	1	1	1	9.06e-11	0.01394	pass
t	s	400	400	20	100	10	10	16	1	1	1	1	4.19e-05	0.01065	FAILED
...															
t	z	100	100	20	20	10	10	16	1	1	1	1	1.81e-04	0.007638	FAILED
t	z	100	100	20	20	10	100	16	1	1	1	1	8.63e-19	0.008050	pass
t	z	100	100	20	100	10	10	16	1	1	1	1	4.23e-19	0.001097	pass
t	z	100	100	20	100	10	100	16	1	1	1	1	6.84e-19	0.001431	pass
t	z	100	100	100	20	10	10	16	1	1	1	1	1.33e-18	0.005349	pass

BLAS++

Basic linear algebra subprograms (BLAS) have been around for many decades and serve as the de facto standard for a performance-portable and numerically robust implementation of essential linear algebra functionality. BLAS++ provides a convenient, performance-oriented API for development in the C++ language and preserves established conventions while taking advantage of modern C++ features.

HIGHLIGHTS

- Covers the entire BLAS (~120 routines)
- Multiple precisions using C++ templates
- Error handling with C++ exceptions
- Covered with a testing suite
- Documented with Doxygen

Mark Gates et al.
C++ API for BLAS and LAPACK
SLATE Working Note #2
http://www.icl.utk.edu/publications/swan-002

DOWNLOAD SOFTWARE AT
https://bitbucket.org/icl/blaspp

Batched BLAS++

Many scientific and engineering computing applications solve large numbers of small and independent linear algebra problems. Such workloads can be executed much more efficiently on modern hardware if they are issued in large batches rather than one by one. To standardize the API, the HPC community is developing an extension to the BLAS standard called Batched BLAS. The objective of Batched BLAS++ (BBLAS++) is to provide a convenient, performance-oriented API for development in the C++ language that preserves established conventions while taking advantage of modern C++ features.

Ahmad Abdelfattah et al.
C++ API for Batch BLAS
SLATE Working Note #4
http://www.icl.utk.edu/publications/swan-004

LAPACK++

The Linear Algebra PACKage (LAPACK) is a standard software library for numerical linear algebra that provides routines for solving systems of linear equations and linear least squares problems, eigenvalue problems, and singular-value decomposition problems. LAPACK++ provides a convenient, performance-oriented API for development in the C++ language and preserves established conventions while taking advantage of modern C++ features.

HIGHLIGHTS

- Covers majority of LAPACK (~1,200 routines)
- Multiple precisions using C++ templates
- Error handling with C++ exceptions
- Covered with a testing suite
- Documented with Doxygen

Mark Gates et al.
C++ API for BLAS and LAPACK
SLATE Working Note #2
http://www.icl.utk.edu/publications/swan-002

DOWNLOAD SOFTWARE AT
https://bitbucket.org/icl/lapackpp

SLATE: Developing Sustainable Linear Algebra Software for Exascale
SOFTWARE DEVELOPMENT PRACTICES FOR THE NEXT GENERATION OF SUPERCOMPUTERS

Jamie M Finney
Jakub Kurzak
Piotr Luszczek
Jack Dongarra
Gerald Ragghianti
Mark Gates
Asim Yarkhan
Innovative Computing Laboratory
University of Tennessee, Knoxville

@SLATE_ICL_UTK
slate-user@icl.utk.edu
https://goo.gl/forms/dsMt4zl10xAP4wsm1

CHECK OUT **SLATE** WORKING NOTES
http://www.icl.utk.edu/publications/series/swans

SPONSORED BY