ScaLAPACK AND ELPA:
HOW TO DIAGONALIZE REALLY LARGE DENSE MATRICES

Peter Karpov, Max Planck Computing and Data Facility, Garching

https://github.com/karpov-peter/elpa-tutorial

Meet MPCDF
01.02.2024

DIAGONALIZATION METHODS

for symmetric/hermitian eigenproblems $AX = \lambda X$

Direct methods	**Iterative methods**
$\longrightarrow$ all or substantial part (>10%) of eigenpairs	$\longrightarrow$ small part of eigenpairs (~several hundreds)
$\longrightarrow$ relatively small matrices (up to ~10^6)	$\longrightarrow$ much larger matrices (> 10^9)
$\longrightarrow$ dense matrices	$\longrightarrow$ (typically) sparse matrices
Software: LAPACK, ScaLAPACK, ELPA, …	Software: ARPACK, SLEPc, ChASE, …

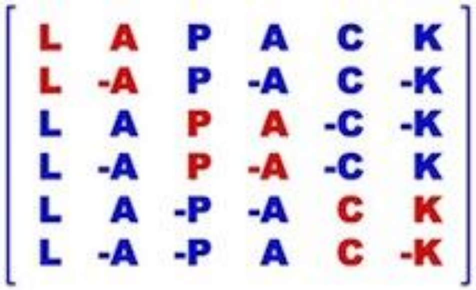
PARALLEL DENSE EIGENSOLVERS

Library	Distributed	GPU	Hybrid	Parallel model	Sparsity	Eigenproblem
LAPACK	×	×	×	OpenMP/pthreads	d/b	std/gen nsym/sym
MAGMA	×	yes (multi-GPU)	yes	OpenMP/pthreads/CUDA	d/s/b	std/gen nsym/sym
cuSolver	×	yes (multi-GPU)	×	CUDA	d/s	std/gen sym
EIGEN	×	×	×	OpenMP	d	std/gen nsym/sym
ScaLAPACK	yes	×	×	MPI/BLASC	d	
ELPA	yes	yes (GPU)	×	MPI/OpenMP/CUDA	d	std/gen sym
EigenEXA	yes	×	×	MPI/OpenMP	d	std sym
FEAST	yes	×	×	MPI	d/s/b	std/gen nsym/sym
Intel MKL	yes	yes (Intel GPU)	×	MPI/OpenMP/pthreads	d/b/s	std/gen nsym/sym
Elemental/Hydrogen	yes	yes (Hydrogen)	yes (Hydrogen)	MPI/OpenMP/(CUDA)	d	std sym
SLATE	yes	yes	yes	MPI/OpenMP/CUDA	d	std sym
P_ARPACK	yes	×	×	MPI/BLACS	s	std/gen nysm/sym
LIS	yes	×	×	MPI/OpenMP	d/s	

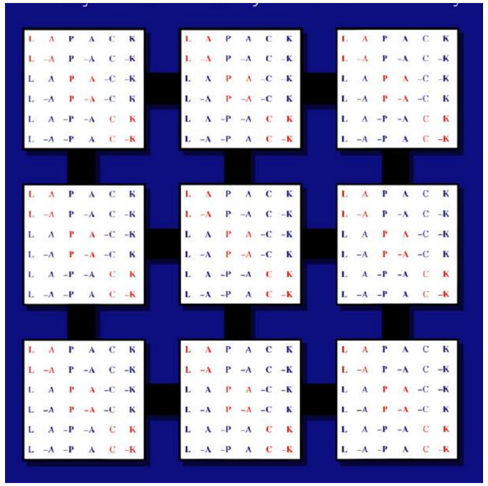
Davor Davidović, An overview of dense eigenvalue solvers for distributed memory systems, 44th International Convention on Information, Communication and Electronic Technology (2021)

OUTLINE

1. LAPACK
(warmup)



2. ScaLAPACK



3. ELPA

LAPACK SOFTWARE FAMILY

	matrix-matrix operations	"advanced" linear algebra
sequential	BLAS	LAPACK
parallel	PBLAS	ScaLAPACK

BLAS Level 1 Routines:
vector-vector operations

BLAS Level 2 Routines:
matrix-vector operations

BLAS Level 3 Routines:
matrix-matrix operations
(e.g **matrix-matrix multiplication**)

systems of linear equations
linear least squares
eigenvalue problems
singular value decomposition

LAPACK / BLAS

$$\begin{bmatrix} L & A & P & A & C & K \\ L & -A & P & -A & C & -K \\ L & A & P & A & -C & -K \\ L & -A & P & -A & -C & K \\ L & A & -P & -A & C & K \\ L & -A & -P & A & C & -K \end{bmatrix}$$

Linear **A**lgebra **PACK**age
Basic **L**inear **A**lgebra **S**ubprograms

+ fast

+ extensively tested, stable

– cumbersome interface

Many packages use it under the hood:

NumPy/SciPy (python), Armadillo (C++), MATLAB, …

LAPACK / BLAS

No distributed memory parallelism

“**driver**” routines – complete solution

“**computational**” routines – complete one solution step

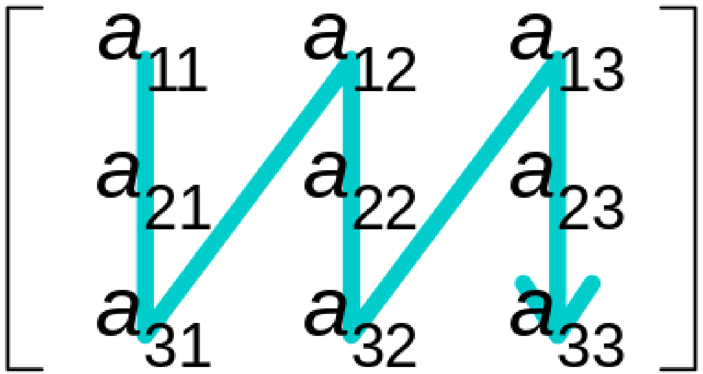
Naming convention: `pmmaa(a)`

`p` – precision:	`s,d`	–	real **s**ingle and **d**ouble precision
	`c,z`	–	complex single and double precision
`mm` – matrix type:	`sy`	–	**sy**mmetric
	`ge`	–	**ge**neral
`aa(a)` – algorithm:	`mm`	–	**m**atrix **m**ultiplication
	`evd`	–	**e**igen**v**alue problem with **d**ivide-and-conquer algorithm

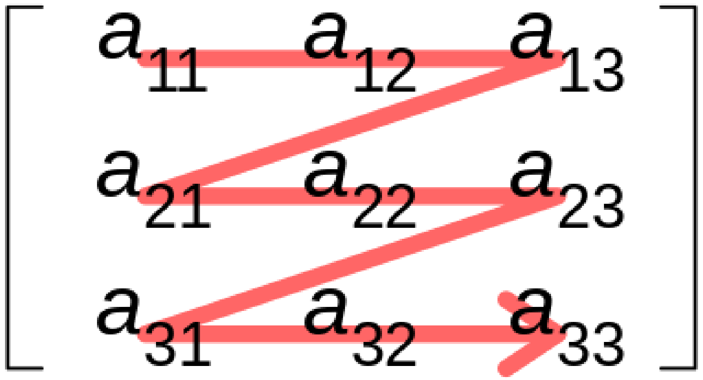
Examples: `sgemm`, `dsyevd`

LAPACK: MATRIX REPRESENTATION

Column-major order

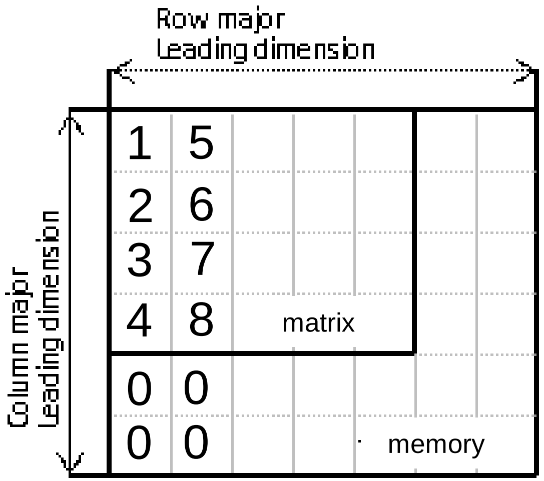


Row-major order



n=5

m=4



Column-major memory representation:
A = {1, 2, 3, 4, 0, 0, 5, 6, 7, 8, 0, 0, …}

(column-major) leading dimension of matrix A:
lda = 6

EXAMPLE: MATRIX-MATRIX MULTIPLICATION WITH LAPACK

$$C = \alpha AB + \beta C$$

```
call dgemm (transa, transb, m, n, k,
     alpha, a, lda, b, ldb, beta, c, ldc)
```

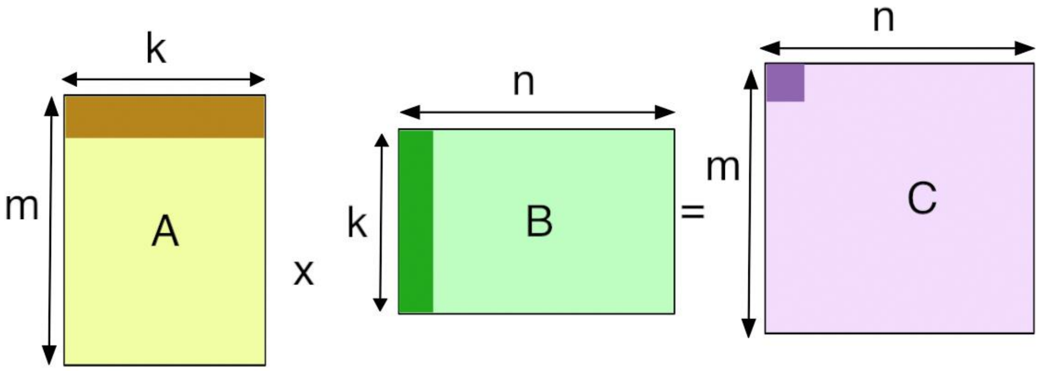


Figure: https://github.com/iVishalr/GEMM

```fortran
program dgemm_example
  implicit none
  double precision :: a(6), b(6), c(9)

  data a /1.d0, 1.d0, 2.d0, 2.d0, 3.d0, 3.d0/
  data b /1.d0, 2.d0, 3.d0, 1.d0, 2.d0, 3.d0/
  c = 0.d0

  call dgemm('N', 'N',
             3, 3, 2, 1.d0, a, 2, b, 3, 0.d0, c, 3)
end program
```

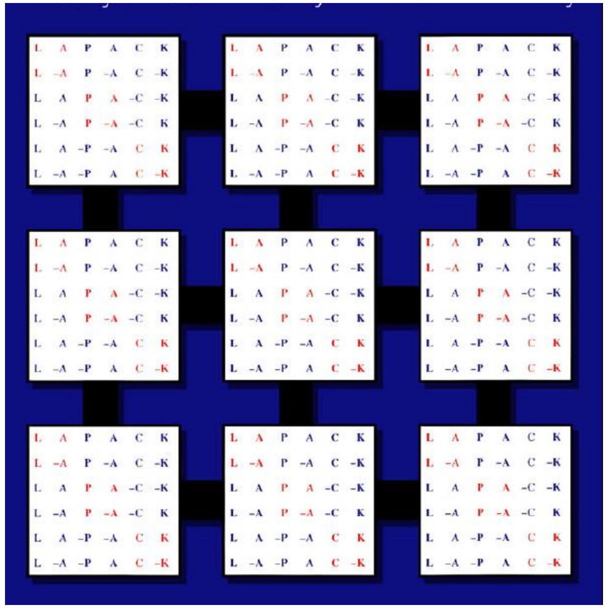
```c
#include "cblas.h"

int main()
{
double a[6] = {1, 1, 2, 2, 3, 3};
double b[6] = {1, 2, 3, 4, 5, 6};
double c[9] = {0};

cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,
                 3, 3, 2, 1.0, a, 2, b, 3, 0.0, c, 3);

return 0;
}
```

OUTLINE

1. LAPACK

2. ScaLAPACK

3. ELPA



ScaLAPACK

Scalable **LAPACK** – MPI parallel version of LAPACK

Scaling (e.g. GEMM, eigenproblem): memory ~ N^2, time ~ N^3

Less functions (e.g. no eigensolver for non-symmetric matrices)

Naming: p+LAPACK name → ScaLAPACK name:
Examples: sgemm → **p**sgemm
dsyevd → **p**dsyevd

Requires a special distribution of matrices among the processes:
“**block-cyclic matrix distribution**” → a major hurdle for the newcomers

ScaLAPACK: 2D PROCESS GRID

BLACS grid (Basic Linear Algebra Communication Subprograms)

my_pcol

my_prow	0	1	2	3
0	0	1	2	3
1	4	5	6	7

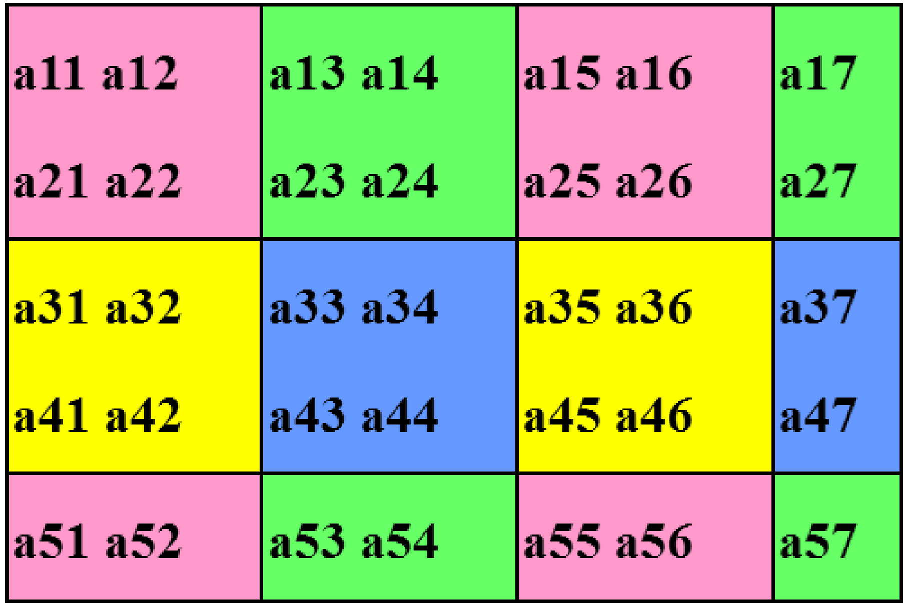
np_row = 2
np_col = 4

MPI rank	BLACS grid coordinates
world_rank=0	my_prow=0, my_pcol=0
world_rank=1	my_prow=0, my_pcol=1
world_rank=2	my_prow=0, my_pcol=2
world_rank=3	my_prow=0, my_pcol=3
world_rank=4	my_prow=1, my_pcol=0
…	…

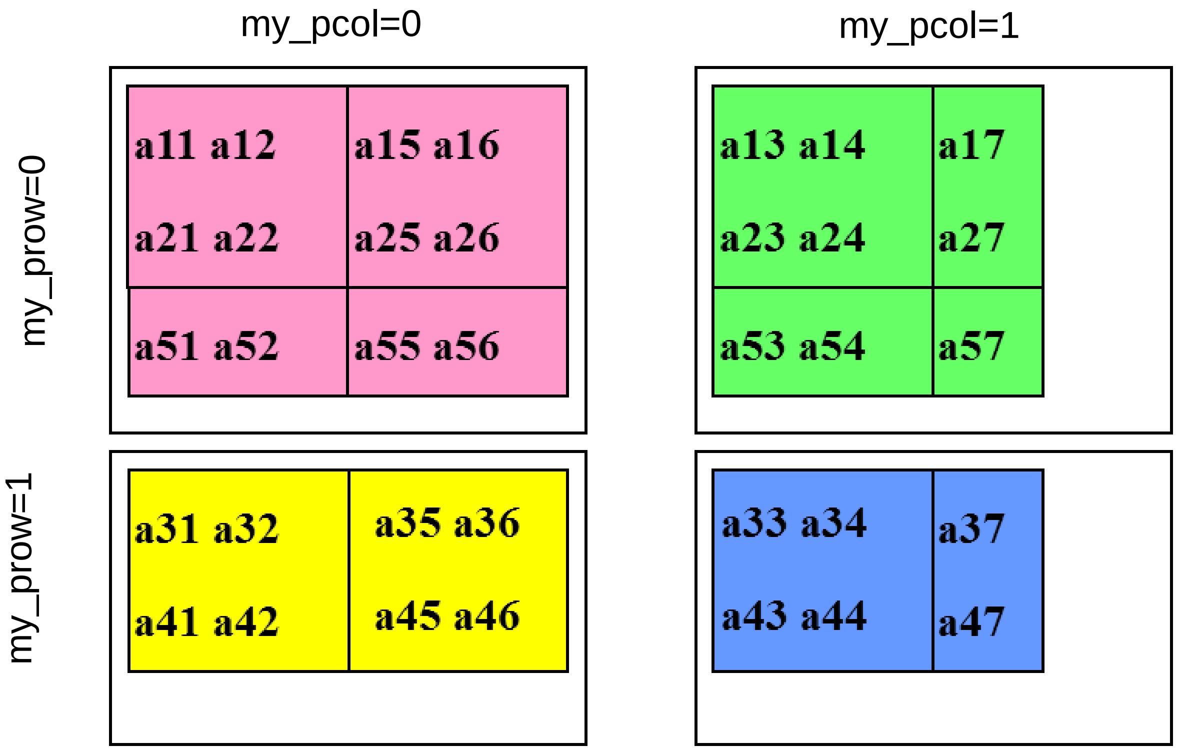
ScaLAPACK: BLOCK-CYCLIC DISTRIBUTION

```
m,n            number of rows and columns of the matrix
mb, nb         sizes of blocks in columns and in rows
np_row, np_col number of rows and columns of the two dimensional process grid
rsrc, csrc     row and column index of the process containing the first element of the matrix
```

Global matrix



Local matrices



m=5,n=7, mb=nb=2, np_row=np_col=2, rsrc=csrc=0

https://info.gwdg.de/wiki/doku.php?id=wiki:hpc:scalapack

MAPPING BETWEEN GLOBAL AND LOCAL INDICES

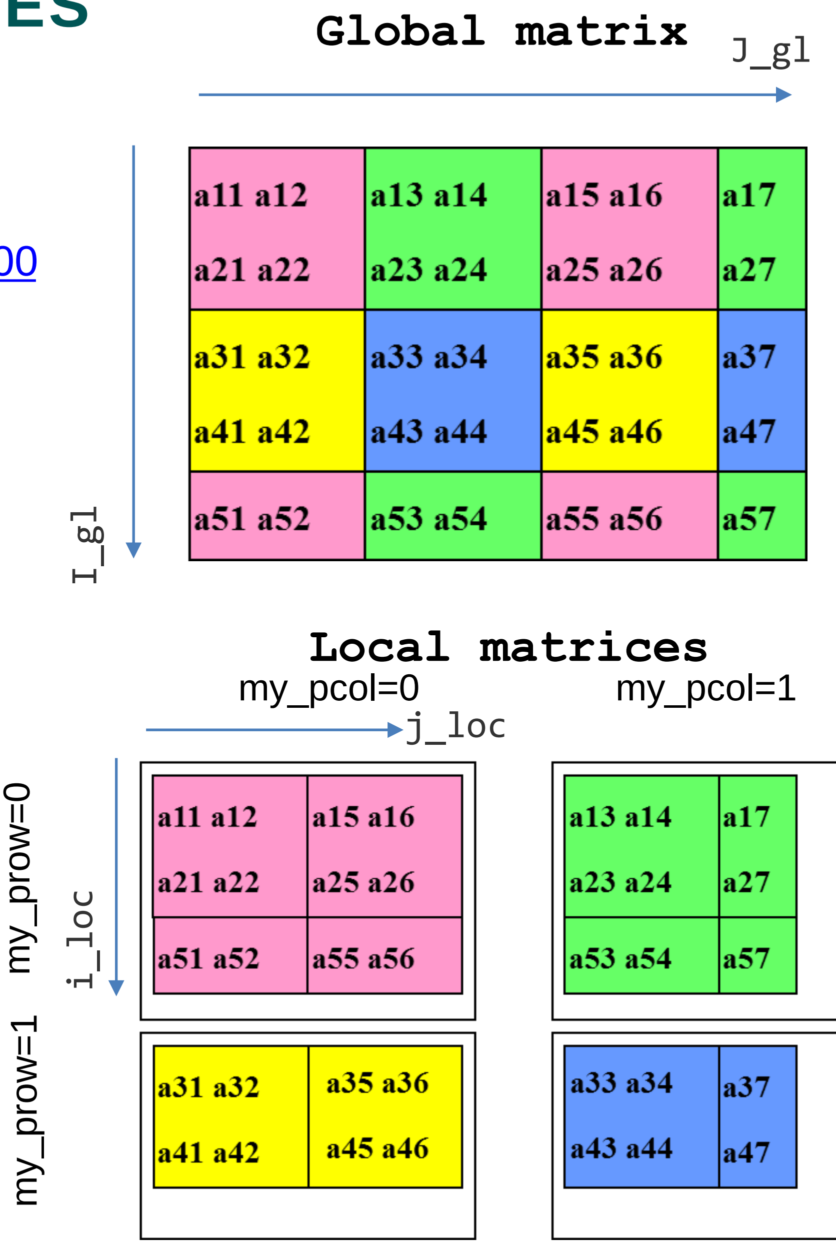
ScaLAPACK Users' Guide (1997), p.61-64
Sec. 4.3.2 Local Storage Scheme and Block-Cyclic Mapping
https://www.netlib.org/scalapack/slug/node76.html#SECTION04432000000000000000

GDWG HPC-wiki: ScaLAPACK (Göttingen)
https://info.gwdg.de/wiki/doku.php?id=wiki:hpc:scalapack

```
I_gl   <-->   (i_loc, my_prow)
J_gl   <-->   (j_loc, my_pcol)
```

```
I_gl    = INDXL2G(i_loc, nb, my_prow, rsrc  , np_row)

i_loc   = INDXG2L(I_gl , nb, dummy1 , dummy2, np_row)
my_prow = INDXG2P(I_gl , nb, dummy1 , rsrc  , np_row)
```

Fortran 1-based indexing! E.g. `I_gl=1,2,…`
`dummy1, dummy2` – dummy integer variables, to keep fixed the positions of arguments



EXAMPLE: FIND ALL EIGENVALUES AND EIGENVECTORS OF A SYMMETRIC MATRIX WITH SCALAPACK

```
! Initialize MPI
call MPI_Init(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, world_size, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, world_rank, ierr)

! BLACS grid initialization
call blacs_get(0, 0, ictxt)
call blacs_gridinit(ictxt, 'row', np_row, np_col)
call blacs_gridinfo(ictxt, np_row, np_col, my_prow, my_pcol)

! Compute local matrix sizes, m_loc, n_loc
call numroc(m_loc, N, my_prow, 0, np_row, NB)
call numroc(n_loc, N, my_pcol, 0, np_col, NB)

! Initialize array descriptors for distributed matrix A and Z
call descinit(descA, N, N, NB, NB, 0, 0, ictxt, m_loc, info)
call descinit(descZ, N, N, NB, NB, 0, 0, ictxt, m_loc, info)

! Allocate local storage for distributed matrix A (to be diagonalized) and Z (eigenvector matrix)
allocate(A_loc(m_loc, n_loc))
allocate(Z_loc(m_loc, n_loc))
! Allocate space for eigenvalues -- global array
allocate(W(N))

! Fill matrix A_loc
```

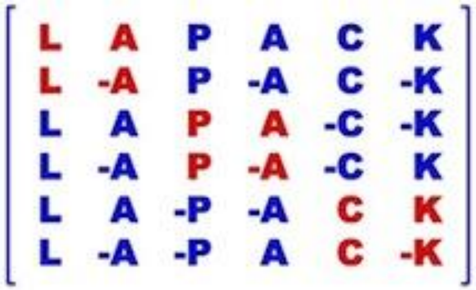
EXAMPLE: FIND ALL EIGENVALUES AND EIGENVECTORS OF A SYMMETRIC MATRIX WITH SCALAPACK

```
! work -- auxillary array of doubles for pdsyev, pdsyevd, pdsyevr, pdsyevx
allocate(work(1))
! lwork - integer parameter, size of the work array, to be determined later. We set it to -1 to
indicate that we first perform a dry run to determine the optimal size of the work array
lwork=-1

! dry run
call pdsyev('V', 'U', N, A_loc, 1, 1, descA, W, Z_loc, 1, 1, descZ, work, lwork, info)


deallocate(work)
allocate(work(lwork)) ! now array work has a proper size

! run with the actual calculation
! "V" - compute eigenvalues and eigenvectors ("N" - only eigenvalues)
! "U" - use upper ("L" - lower) triangular part of matrix A (it's symmetric anyhow)
call pdsyev('V', 'U', N, A_loc, 1, 1, descA, W, Z_loc, 1, 1, descZ, work, lwork, info)

! Cleanup
deallocate(A_loc)
deallocate(Z_loc)
deallocate(W)
deallocate(work)
call blacs_gridexit(ictxt)
call mpi_finalize(ierror)
```

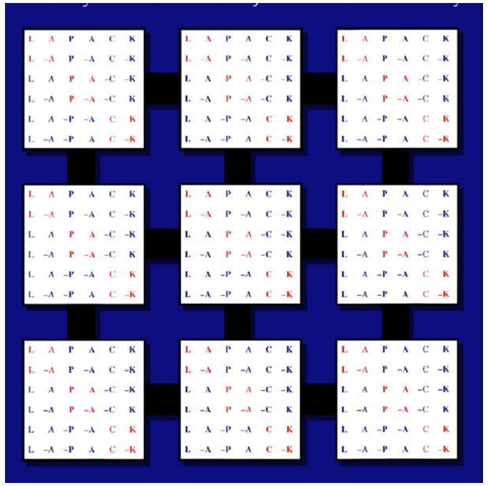
Eigenproblem: $AZ = wZ$

OUTLINE

1. LAPACK



2. ScaLAPACK



3. ELPA

ELPA

now we target **Exaflop** and beyond

Eigenvalue soLvers for Petaflop Applications (started in 2008 at MPCDF)
(Eigenwert-Löser für Petaflop-Anwendungen)

Direct eigensolver for **dense** large-scale symmetric/hermitian matrices

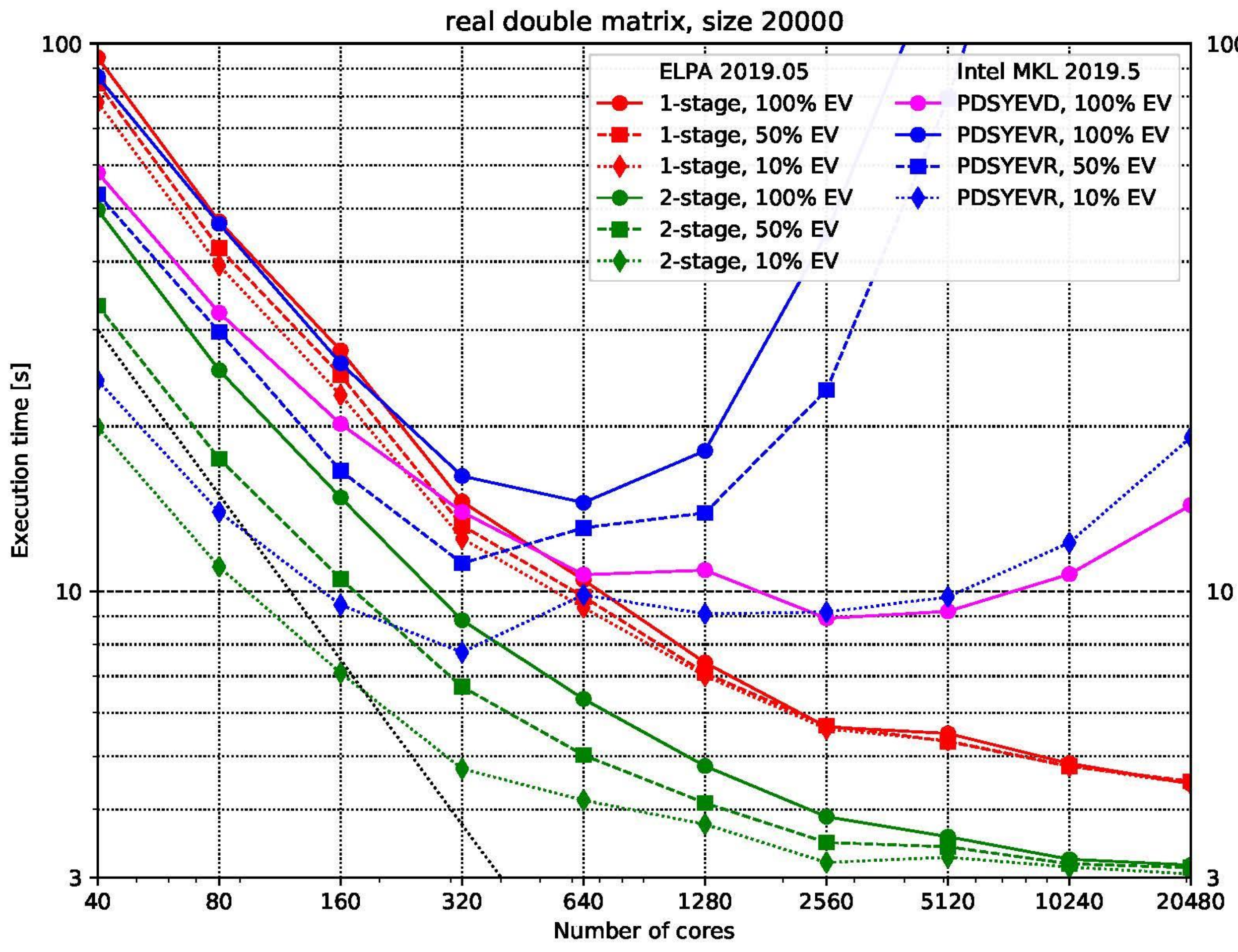
ELPA vs ScaLAPACK:

- ELPA is eigensolver, not general purpose linear algebra
- ELPA is up to ~x2 faster than ScaLAPACK
- **ELPA works on GPUs** (NVIDIA, AMD, Intel)
- ELPA uses same “block-cyclic“ matrix layout as ScaLAPACK

- ABINIT
- BerkeleyGW
- CP2K
- CPMD
- DFTB+
- EIGENKERNEL
- ELSI
- FHI-aims

- NWChem
- Octopus
- OpenMM
- OpenMX
- QuantumATK
- QuantumEspresso
- SIESTA
- VASP

ELPA VS ScaLAPACK

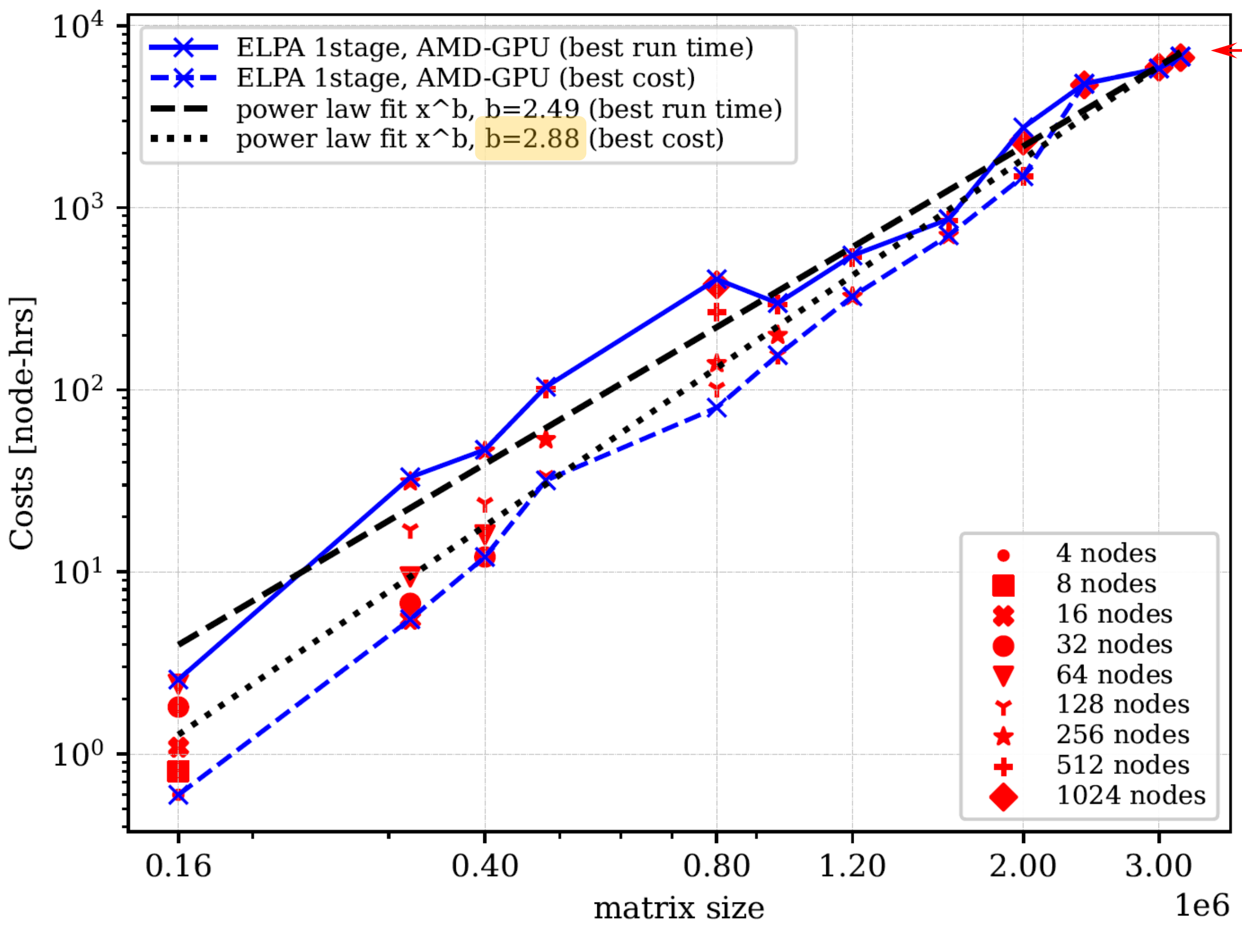


ELPA's additional perks:

- Generalized eigenproblem
- Antisymmetric eigenproblem
- PDGEMM-GPU (coming soon!)

Credit: P. Kus, A. Marek, H. Lederer (MPCDF)

FULL SUPPORT FOR AMD GPU'S: READY TO USE! (since 2022.11)



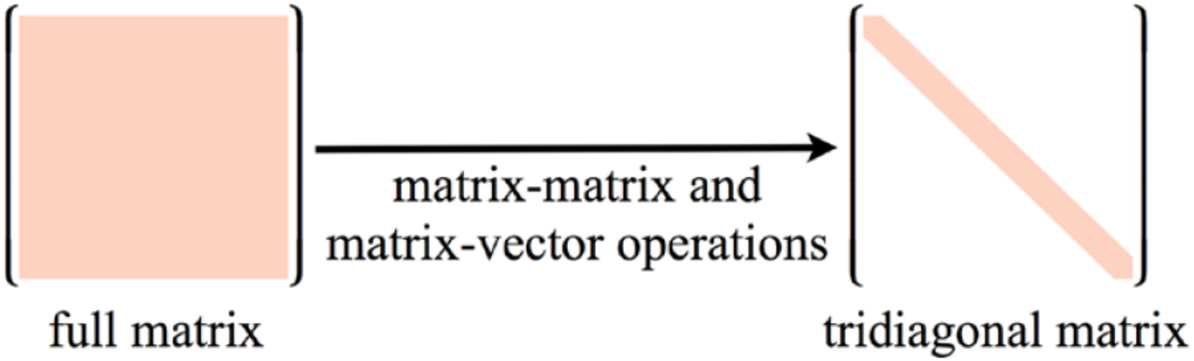
Nov. 2022 world record
3.2 M × 3.2 M matrix
(1000 nodes, 4000 GPUs, ELPA1)

Benchmarks on pre-exascale LUMI, Finland (2500 nodes, 4 AMD Mi250x GPUs)

Credit: Andreas Marek (MPCDF)

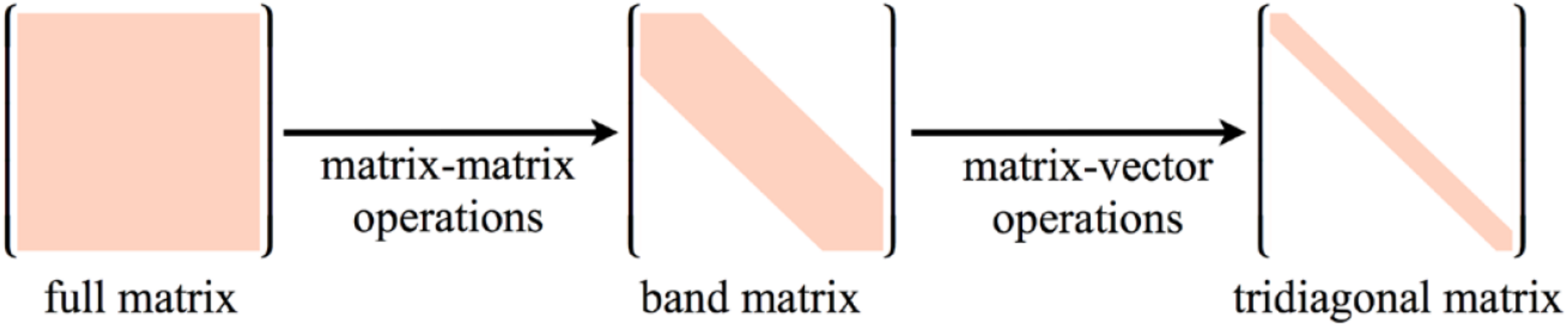
ELPA: 1- AND 2-STAGE SOLVERS

ELPA1 (one stage solver)
- for GPUs
- for the whole eigenspectrum



ELPA2 (two stage solver)
- for CPU
- for part of eigenspectrum

idea: Bruno Lang



A. Marek, V. Blum et al, J. Phys.: Condens. Matter 26 213201 (2014)

EXAMPLE: FIND ALL EIGENVALUES AND EIGENVECTORS OF A SYMMETRIC MATRIX WITH ELPA

```
use elpa

class(elpa_t), pointer :: elpaInstance

integer :: success
success = elpa_init(20170403) ! Initialize ELPA
! We recommend always check success code, e.g. with
if (success /= ELPA_OK) then
         print *, "ELPA API version not supported"
         ! Handle this error in your application
endif

! Allocate ELPA object
elpaInstance => elpa_allocate(success)
! Check success code ...

! Set mandatory parameters describing the matrix and its MPI distribution - can be done only once per ELPA object
call elpaInstance%set("na", na, success) ! global matrix size
call elpaInstance%set("nev", nev, success) ! number of eigenvectors to be computed
call elpaInstance%set("local_nrows", na_rows, success) ! number of rows in the local matrix
call elpaInstance%set("local_ncols", na_cols, success) ! number of columns in the local matrix
call elpaInstance%set("nblk", nblk, success) ! block size of the block-cyclic distribution
call elpaInstance%set("mpi_comm_parent", MPI_COMM_WORLD, success)
call elpaInstance%set("process_row", my_prow, success) ! process row in the BLACS grid
call elpaInstance%set("process_col", my_pcol, success) ! process column in the BLACS grid

! Finalize the setup of the elpa object for the given mandatory parameters
success = elpaInstance%setup()
```

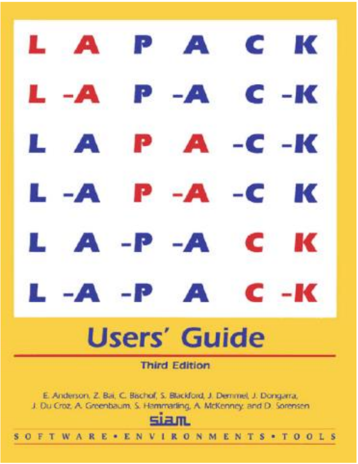
EXAMPLE: FIND ALL EIGENVALUES AND EIGENVECTORS OF A SYMMETRIC MATRIX WITH ELPA

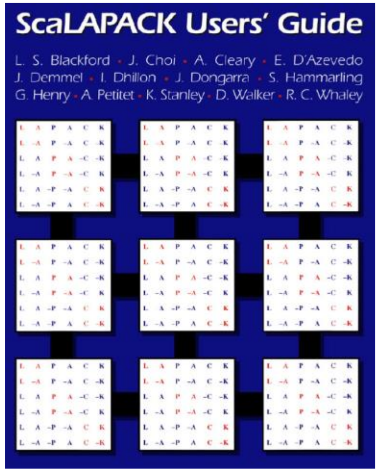
```fortran
! Set runtime options - they can be changed between the calls of ELPA computational routines
call elpaInstance%set("solver", ELPA_SOLVER_2STAGE)

! Fill matrix A_loc

! Solve the eigenvalue problem to obtain eigenvalues (W) and eigenvectors (Z_loc) pf the given matrix (A_loc)
call elpaInstance%eigenvectors(A_loc, W, Z_loc, success)

! ELPA cleanup
call elpa_deallocate(elpaInstance, success)
call elpa_uninit()
```

HOW COMPILE ON RAVEN YOUR PROGRAM THAT USES ELPA

```
module purge
module load intel/2023.1.0.x
module load impi/2021.9
module load mkl/2023.1
module load elpa/mpi/standard/2023.11.001

echo $ELPA_HOME
echo $ELPA_LIBS

ELPA_INCLUDE_DIR=$ELPA_HOME/include/elpa-2023.11.001
#ELPA_INCLUDE_DIR=$ELPA_HOME/include/elpa_openmp-2023.11.001

FileName="eigenproblem_elpa"

mpiicc -std=c11 -I$ELPA_INCLUDE_DIR $FileName.c -o $FileName $ELPA_LIBS
```

RESOURCES: SCALAPACK





LAPACK user guide https://www.netlib.org/lapack/lug/

ScaLAPACK user guide https://www.netlib.org/scalapack/slug/

Intel MKL Developer Reference for Intel® oneAPI Math Kernel Library for C
Developer Reference for Intel® oneAPI Math Kernel Library for Fortran

Short intro from GWDG https://info.gwdg.de/wiki/doku.php?id=wiki:hpc:scalapack

Slides on ScaLAPACK by Karim Hasnaoui (IDRIS)
https://events.prace-ri.eu/event/1286/attachments/1667/3912/ScaLAPACK_PTC.pdf

Slides on Scalable Linear Algebra by Nicola Spallanzani (SCAI)
https://materials.prace-ri.eu/441/3/scalableLinearAlgebraHNDSCi15.pdf

RESOURCES ELPA

Official website: https://elpa.mpcdf.mpg.de/

Official repository: https://gitlab.mpcdf.mpg.de/elpa/elpa

Mirror github repository: https://github.com/marekandreas/elpa

This presentation, ScaLAPACK/ELPA code examples (Fortran/C), **ELPA User Guide** [draft]:
https://github.com/karpov-peter/elpa-tutorial

Need help with ELPA? https://helpdesk.mpcdf.mpg.de/
elpa-library@mpcdf.mpg.de
petr.karpov@mpcdf.mpg.de