Divide & Conquer: a symmetric tridiagonal eigensolver

Grégoire Pichon

Supervisors: Mathieu Faverge,
Azzam Haidar, Jakub Kurzak

August 29, 2014

Eigenproblems in PLASMA

PLASMA symmetric eigensolvers are following three stages

- $A = QTQ^T$: reduction to tridiagonal form
- $T = V\Sigma V^T$: tridiagonal eigensolver
- $A = Z\Sigma Z^T$ with $Z = QV$: back-transformation

Objectives

- Implement a tridiagonal eigensolver in PLASMA
- Rely on sequential MKL in spite of multi-threaded MKL

Outline

Eigenproblems in LAPACK

Algorithm	Cost	Practical cost	Extra space	Accuracy
QR	$O(n^3)$		$O(n)$	$O(\sqrt{n}\epsilon)$
BI	$O(n^3)$		$O(n)$	$O(n\epsilon)$
DC	$O(n^2)$ to $O(n^3)$	$O(n^{2.5})$	$O(n^2)$	$O(\sqrt{n}\epsilon)$
MRRR	$O(n^2)$	$O(n^{2.3})$	$O(n)$	$O(n\epsilon)$

Figure: LAPACK solvers for computing all eigenpairs

Algorithm	Time	Accuracy
BI	$O(nk^2)$	$O(n\epsilon)$
MRRR	$O(nk)$	$O(n\epsilon)$

Figure: LAPACK solvers for computing k eigenpairs

Outline

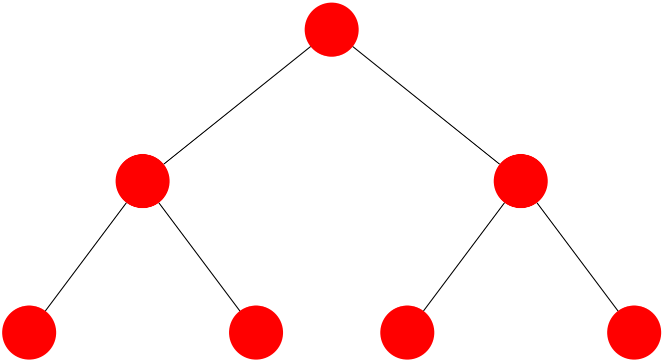
Divide & Conquer – Divide

- Divide the problem into two half subproblems and recursively solve each subproblem
- Divide with a rank-1 approximation:

$$\begin{pmatrix} * & * & 0 & 0 \\ * & * & \beta & 0 \\ 0 & \beta & * & * \\ 0 & 0 & * & * \end{pmatrix} = \begin{pmatrix} * & * & 0 & 0 \\ * & *' & 0 & 0 \\ 0 & 0 & *' & * \\ 0 & 0 & * & * \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \beta & \beta & 0 \\ 0 & \beta & \beta & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- Isolate independent subproblems: when an extra-diagonal element is small enough, no need for a rank-1 approximation

Divide & Conquer – Conquer on the tree

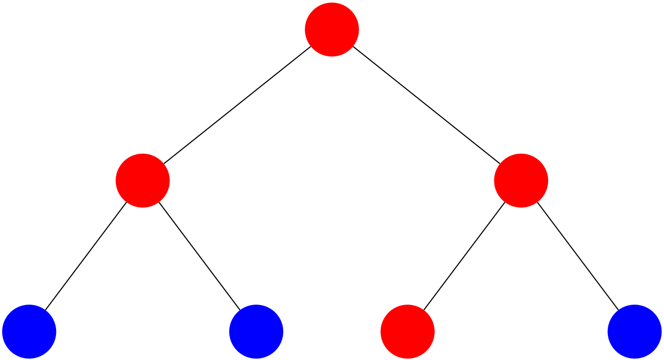


Merge 2

Merge 1.1 and 1.2

Solve

Divide & Conquer – Conquer on the tree

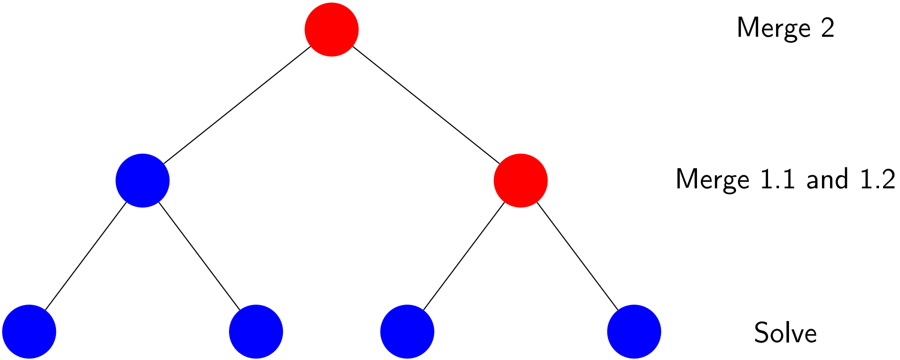


Merge 2

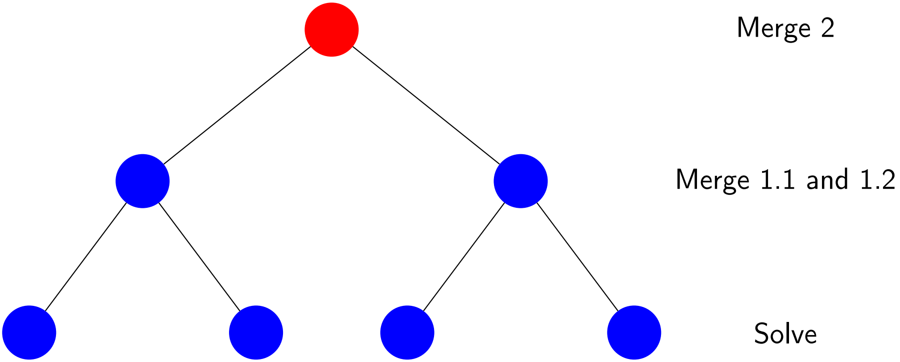
Merge 1.1 and 1.2

Solve

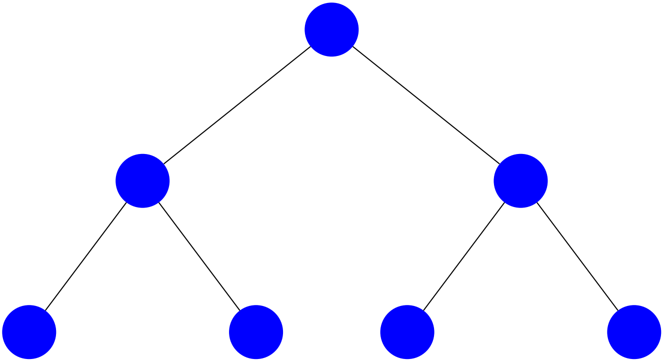
Divide & Conquer – Conquer on the tree



Divide & Conquer – Conquer on the tree



Divide & Conquer – Conquer on the tree

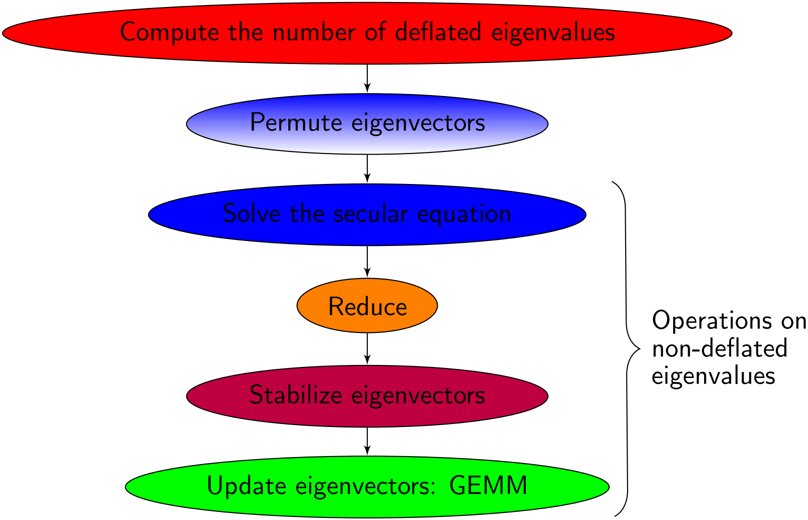


Merge 2

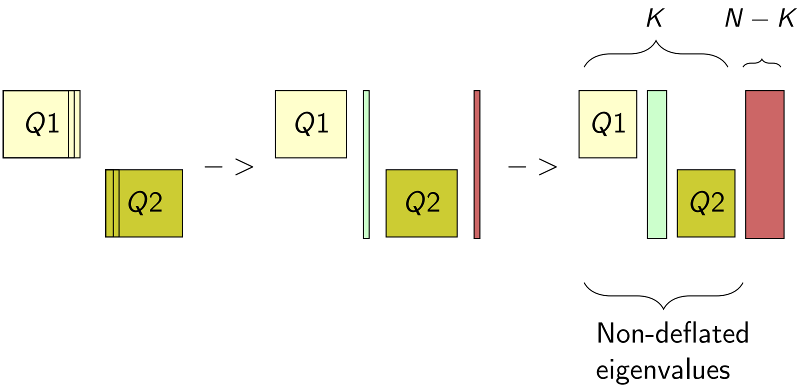
Merge 1.1 and 1.2

Solve

Merge step - ideas

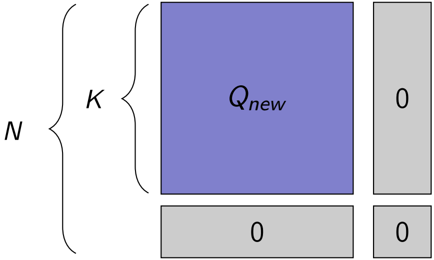


Merge step - deflate

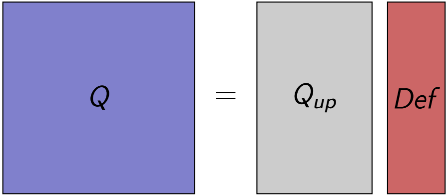


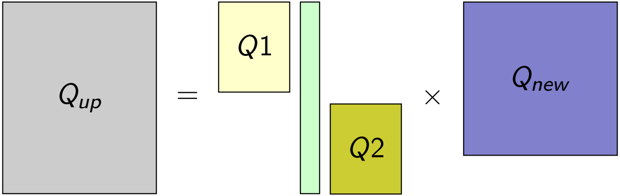
Merge step - compute eigenvectors

- Solve the secular equation (rely on LAPACK)
- Reduce
- Stabilize eigenvectors



Merge step - update eigenvectors





Outline

Implementations

MKL LAPACK

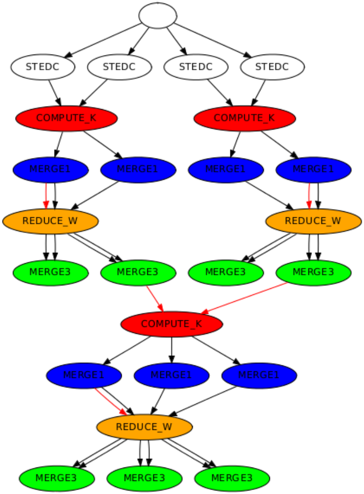
- MKL: LAPACK and link with optimised BLAS3
- Only GEMMs improved, a large part remains sequential
- Good speedup because in sequential, GEMMs represent 80% of the overall time

MKL ScaLAPACK

- Independent subproblems can be solved in parallel
- The merge process is parallel for the GEMMs and the secular equation
- Permutations are different from LAPACK version
- Limit: static scheduling

DAG

$$N = 1000,\ task\ size = 400,\ minimum\ solve\ size = 300$$



Four subproblems size 250

Two merges size 500

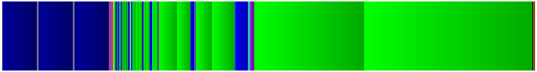
Two tasks: size 400 and size 100

One merge size 1000

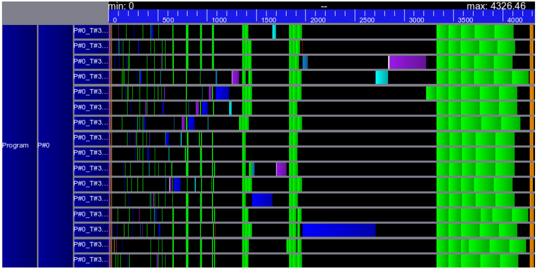
Three tasks: size 400 and size 200

Without deflation - GEMMs in parallel

Traces were obtained on sweetums, using 16 threads for a 10000×10000 matrix



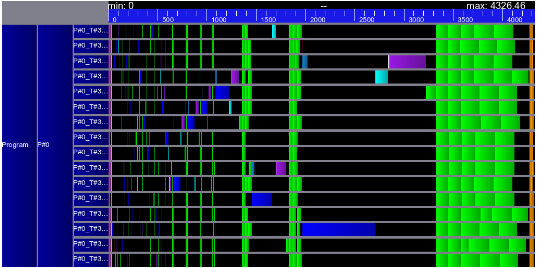
Sequential execution (55s)



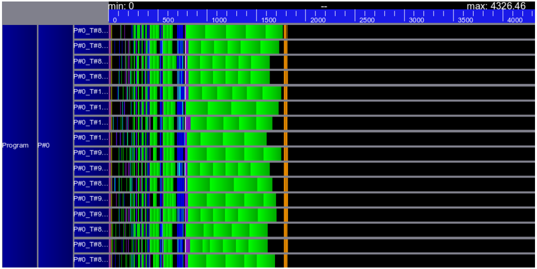
GEMMs in parallel (5s)

	GEMM		Stabilize
	Secular eq		Reduce

Without deflation - Merge step in parallel

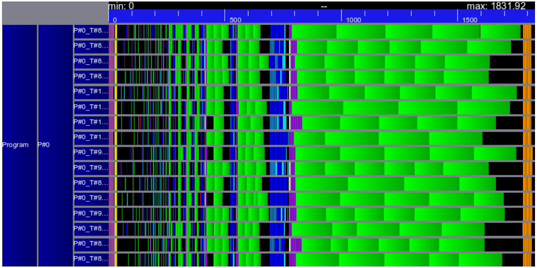


GEMMs in parallel

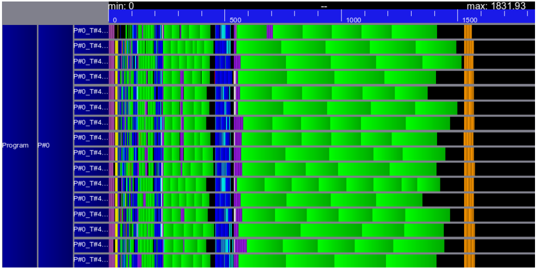


GEMMS and merge step in parallel

Without deflation - Independent subproblems in parallel



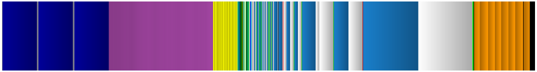
GEMMs and merge step in parallel



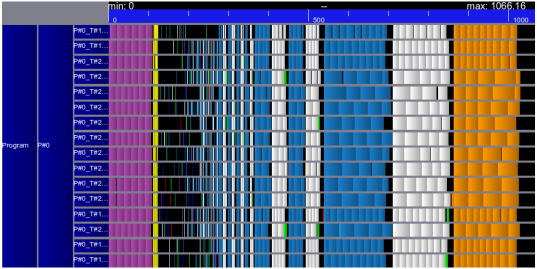
Subproblems in parallel

With deflation - Merge step in parallel

Traces were obtained on sweetums, using 16 threads
for a 10000×10000 matrix



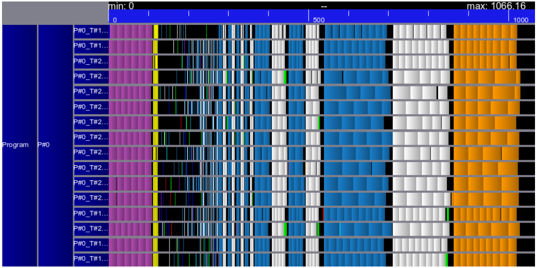
Sequential execution (3s)



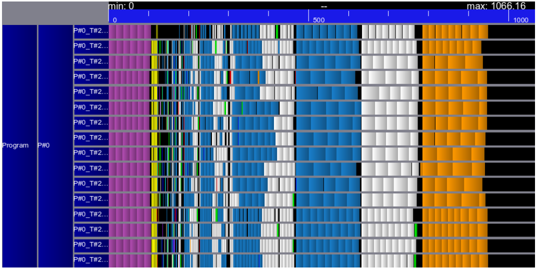
Merge step in parallel (1s)

	Dlaset		Swap
	Permute copy		Permute back

With deflation - Independent subproblems in parallel

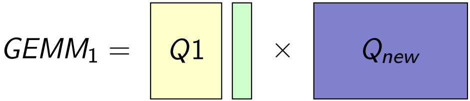


Merge step in parallel



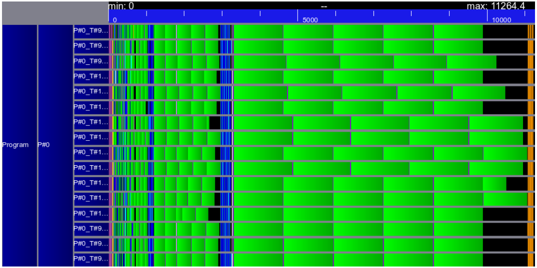
Subproblems in parallel

Extra workspace

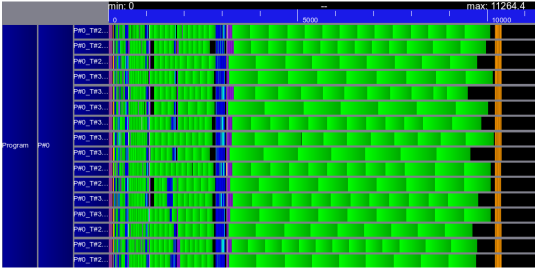


- GEMMs need a workspace because they use twice Q
- After the reduction some workspace is available
- GEMM: copy Q_{new} in S and then generate $Q = Q_{perm}S$
- With an extra workspace, it is possible to generate eigenvectors in S in spite of Q and then avoid the copy
- $GEMM_1$ and $GEMM_2$ in parallel

Extra workspace - on sweetums, 16 threads



LAPACK workspace



Extra workspace

Outline

Time On sweetums – 16 threads

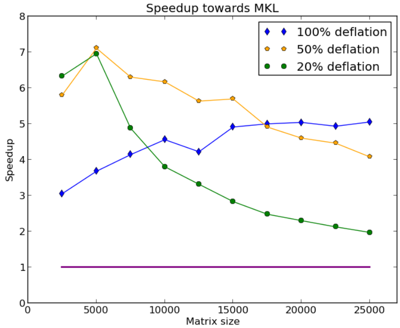


Figure: MKL LAPACK

- Subproblems can't be solved in parallel
- BLAS3 operations in parallel
- Compare with ScaLAPACK to be honest, even if designed for distributed architectures

100% deflation: $D(1:N-1) = 1$ and $D(N) = 1e^{-6}$
50% deflation: $D(i) = \frac{1}{1e6}^{-(i-1)/(N-1)}$
20% deflation: $D(i) = 1 - \frac{i-1}{N-1} \times (1 - 1e^{-6})$

Time On sweetums – 16 threads

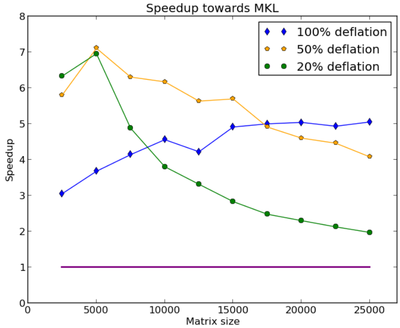


Figure: MKL LAPACK

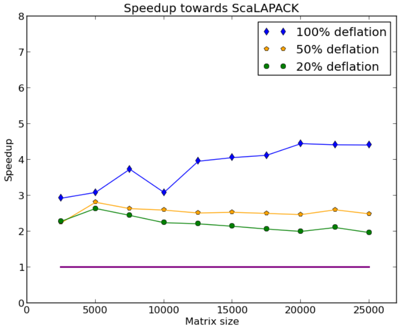


Figure: MKL ScaLAPACK

100% deflation: $D(1:N-1) = 1$ and $D(N) = 1e^{-6}$

50% deflation: $D(i) = \frac{1}{1e6}^{-(i-1)/(N-1)}$

20% deflation: $D(i) = 1 - \frac{i-1}{N-1} \times (1 - 1e^{-6})$

Time versus MR3 from University of Texas

Mode 1 $\rightarrow$ 6: dlatms
Mode 10: Laplacian
Mode 11: Wilkinson
Mode 13: Legendre
Mode 14: Laguerre
Mode 15: Hermite

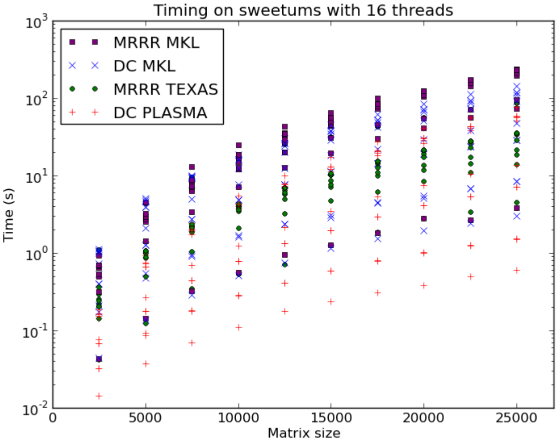


Figure: Timing on sweetums

Accuracy versus MR3 from University of Texas

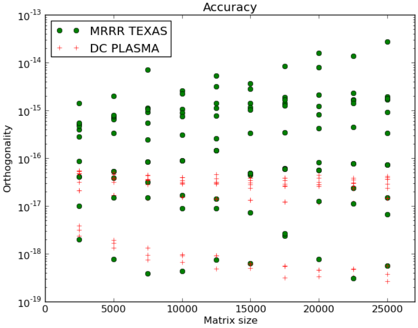


Figure: Orthogonality: $\frac{||Id - QQ^T||}{N}$

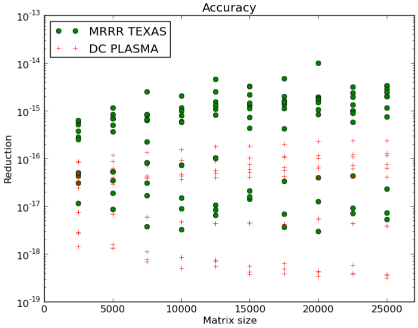


Figure: Reduction: $\frac{||A - Q\Sigma Q^T||}{||A|| \times N}$

Conclusion

In progress

- Timing with many different matrix properties and matrices from applications
- Compute subsets: optimisation in last GEMM for now

Future work

- Integrate the back-transformation in D&C
- Integration in DPLASMA, MAGMA
- Solving SVD using the same approach

Thanks for your attention.

Questions?

Timing versus MR3 from University of Texas

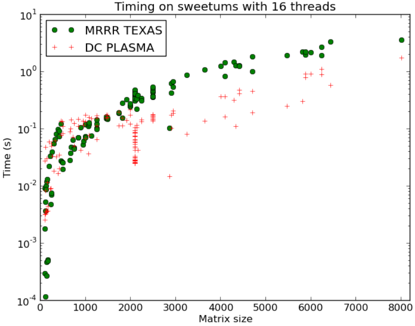


Figure: Application matrices

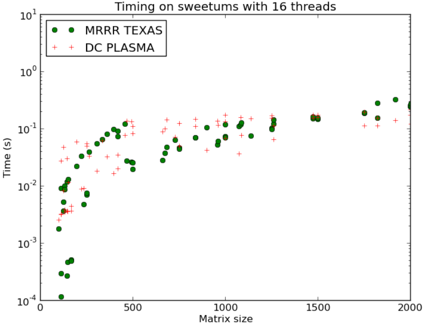


Figure: Zoom $N < 2000$

Matrices: bcsstkm, Alemdar, Godunov, matlab-ud, matlab-nd, nasa, nos, Wilkinson