Accelerating Sparse Iterative Solvers and Preconditioners Using RACE

Christie Alappat, Georg Hager, Gerhard Wellein

Erlangen National High Performance Computing Center (NHR@FAU)

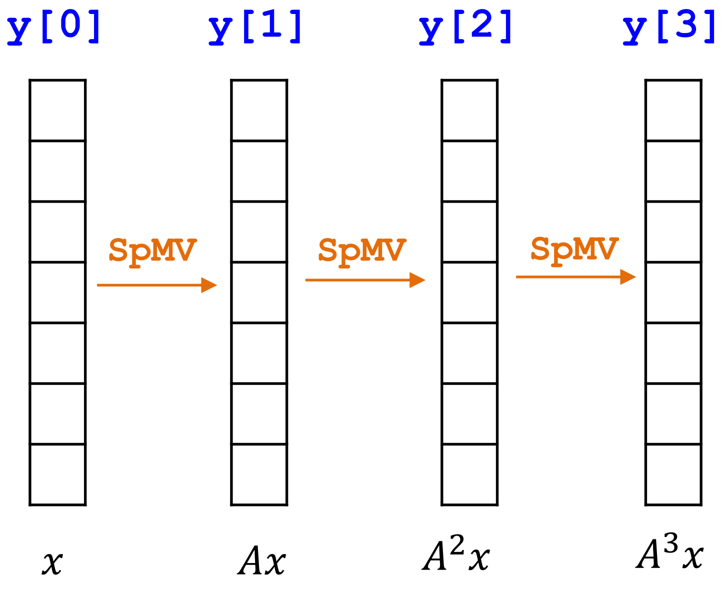
SIAM PP24

Matrix power kernel (MPK)

- Calculate: $y = A^p x$
- Repeatedly perform back to back SpMVs

```
for k=1:p; do
 y[k] = SpMV(A, y[k-1])
done
```

Matrix A loaded p times from main memory.



Can I cache matrix A?

Mohiyuddin et al., 2009. Minimizing communication in sparse matrix solvers. In Proceedings of the SC'09. [*https://doi.org/10.1145/1654059.1654096*](https://doi.org/10.1145/1654059.1654096)

But requires “ghosting". Indirect accesses or redundant copies of the matrix entries, which typically increases with number of threads.

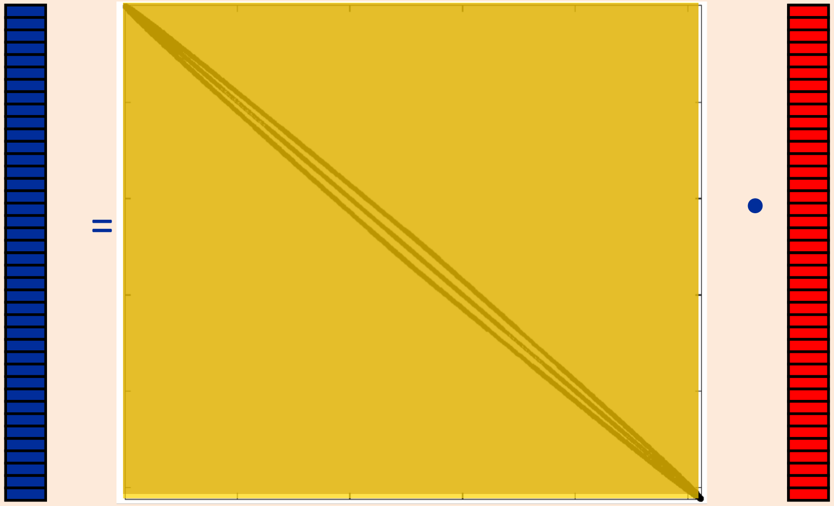
Can I avoid these overhead?

RACE
Hardware Friendly Coloring

Matrix power

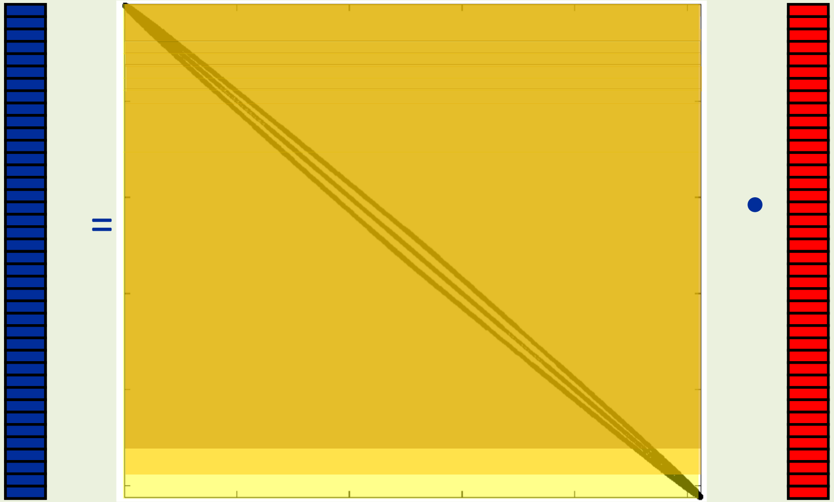
Calculate $y = A^3 x$

TRAD approach



Matrix accessed 3 times from memory

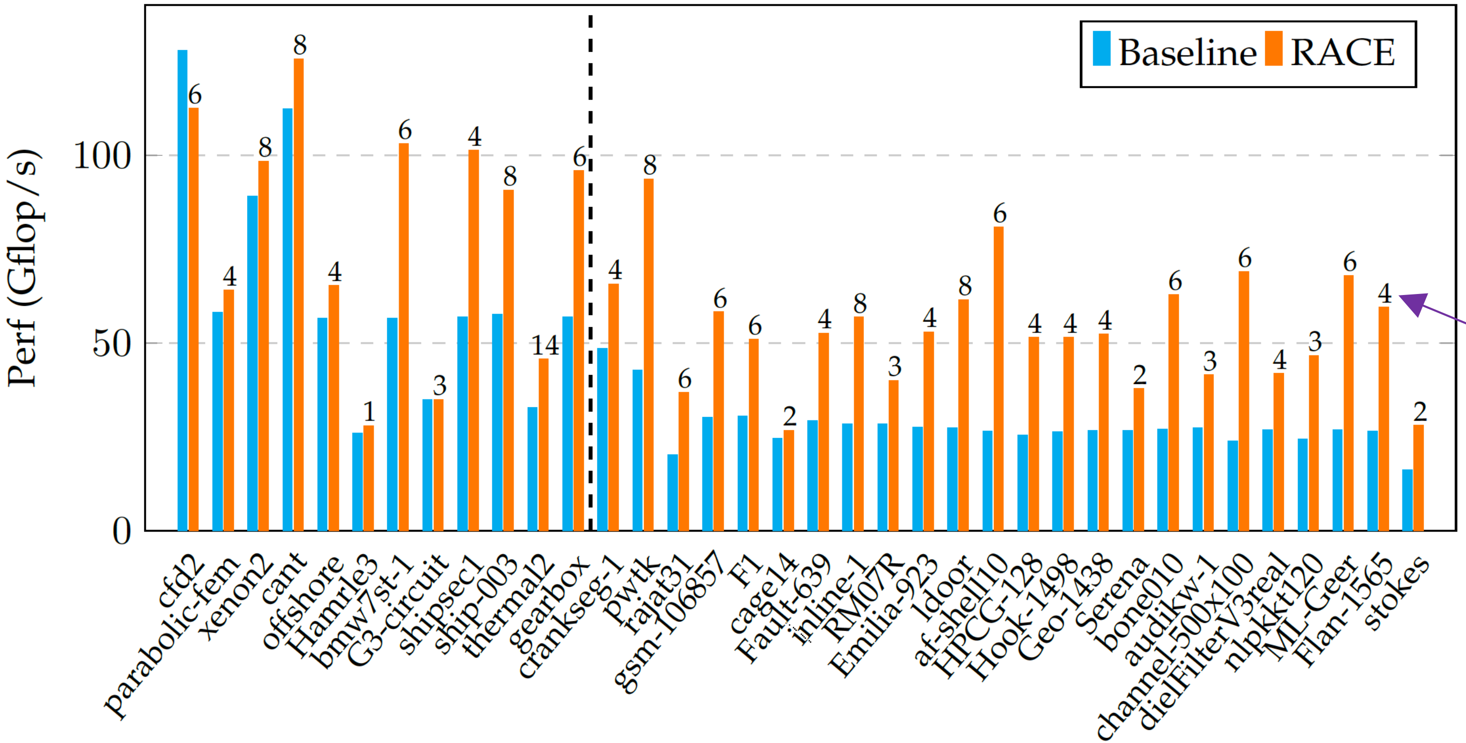
RACE approach



Matrix accessed 1 time from memory

Alappat et al, "Level-Based Blocking for Sparse Matrices: Sparse Matrix-Power-Vector Multiplication," in *IEEE Transactions on Parallel and Distributed Systems*, 2023, doi: 10.1109/TPDS.2022.3223512.

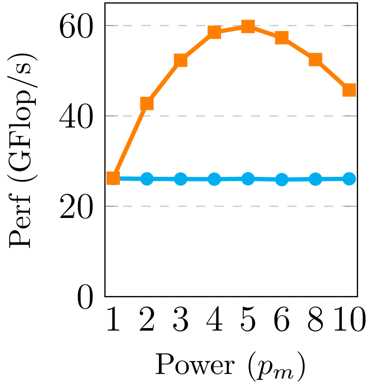
Matrix power kernel: Performance



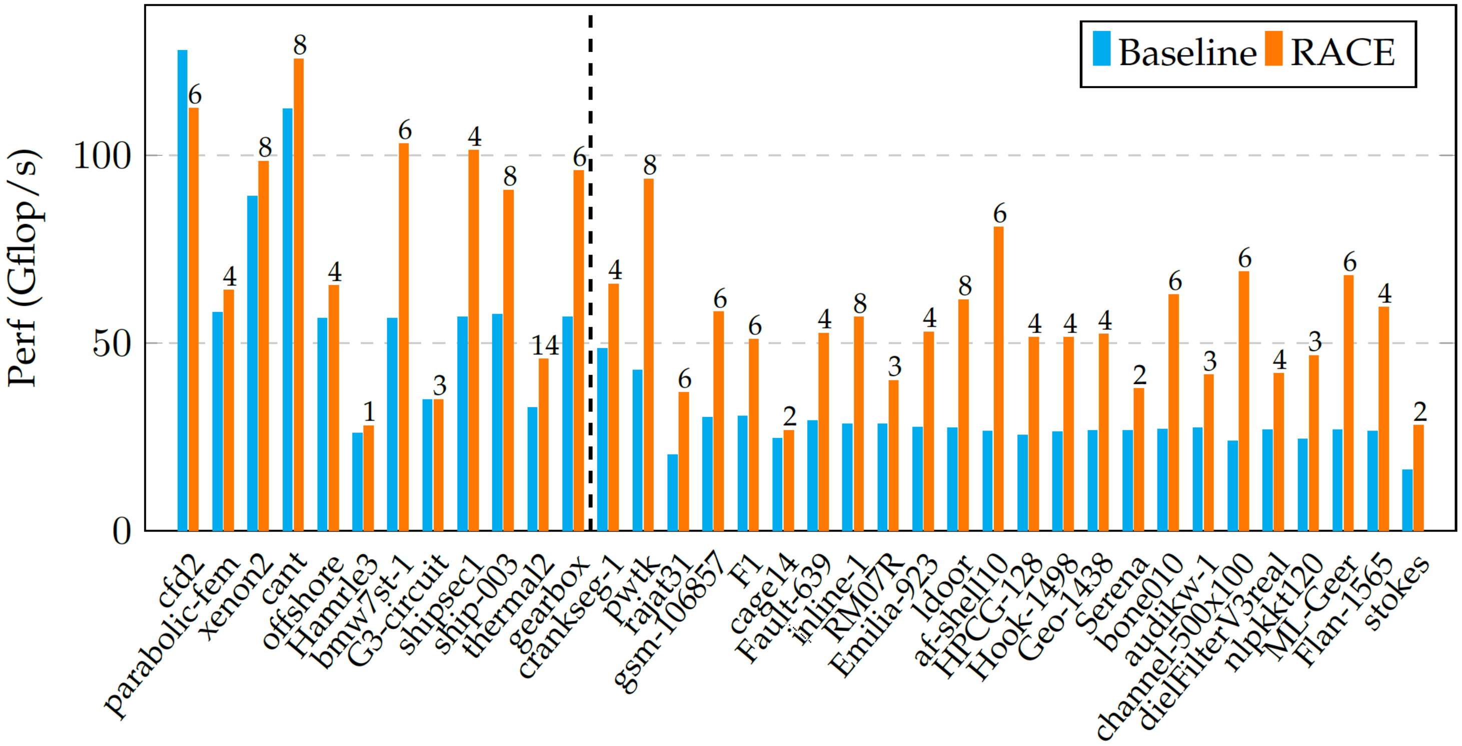
Intel Xeon Platinum 8368 (Ice Lake)

- 38 cores
- 104 MB cache (L2+L3)

Power value with maximum performance.

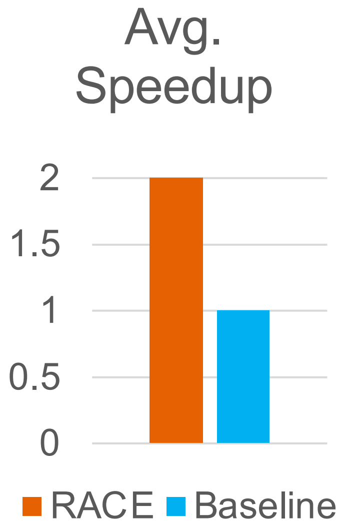


Matrix power kernel: Performance

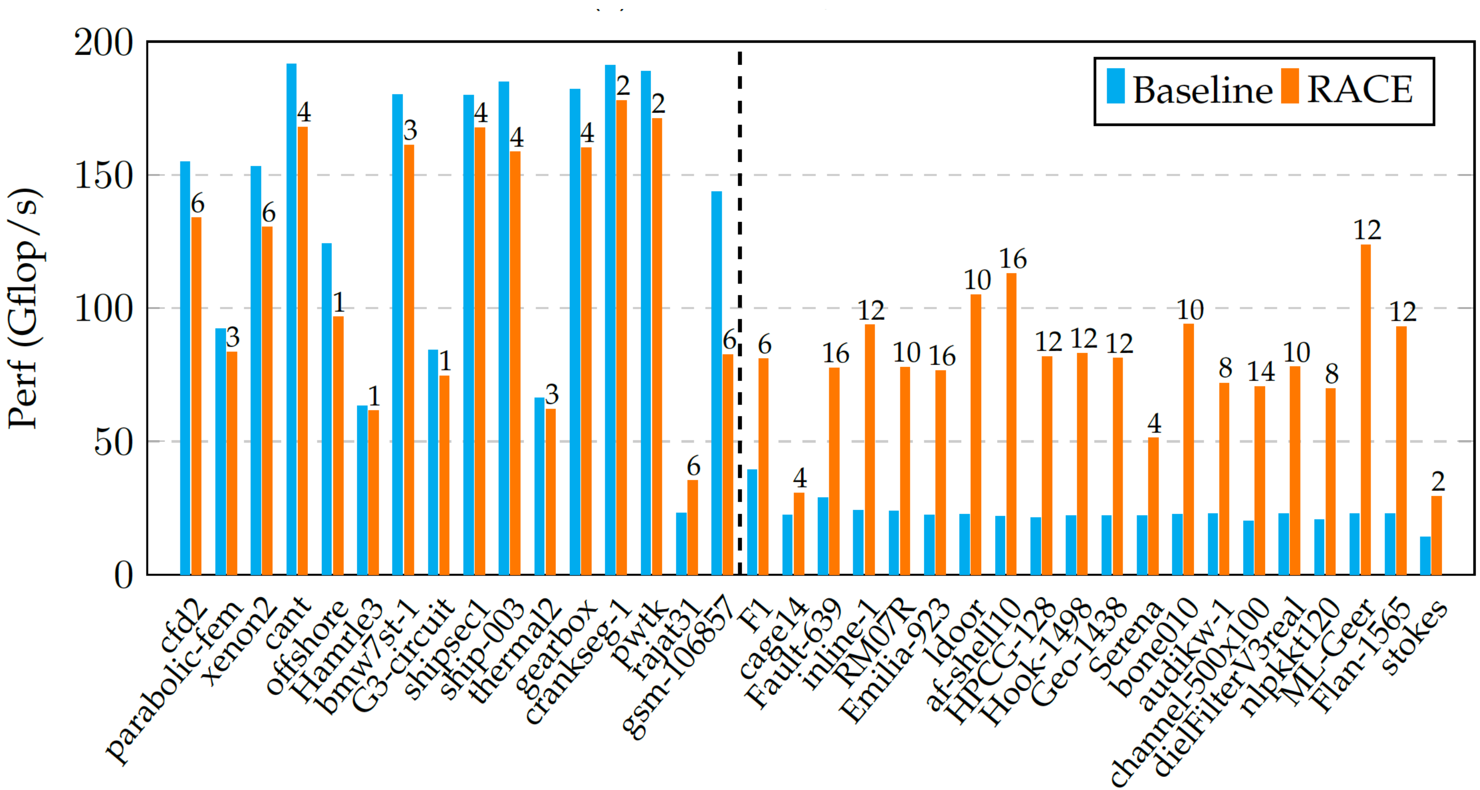


Intel Xeon Platinum 8368 (Ice Lake)

- 38 cores
- 104 MB cache (L2+L3)

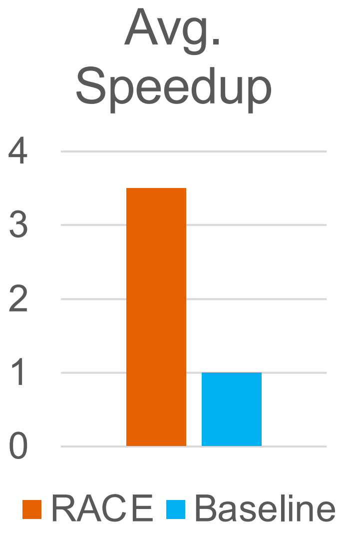


Matrix power kernel: Performance



AMD EPYC 7662 (ROME)

- 64 cores
- 288 MB cache (L2+L3)



Iterative solvers

Solve for x: $Ax = b$

Application: s-step Krylov schemes

Basic computation kernel

GMRES scheme: main kernel

```
for j=0:1:m; do
  v[j+1] = SpMV(A, v[j])
  Orthogonalize v[j+1] against v[0:j]
done
```

s-step GMRES scheme: main kernel

```
for j=0:s:m; do
  for p=0:1:s; do
    v[j+p+1] = SpMV(A, v[j+p])
  done
  Orthogonalize v[j+1:j+s+1] against v[0:j]
  Orthogonalize within v[j+1:j+s+1]
done
```

MPK

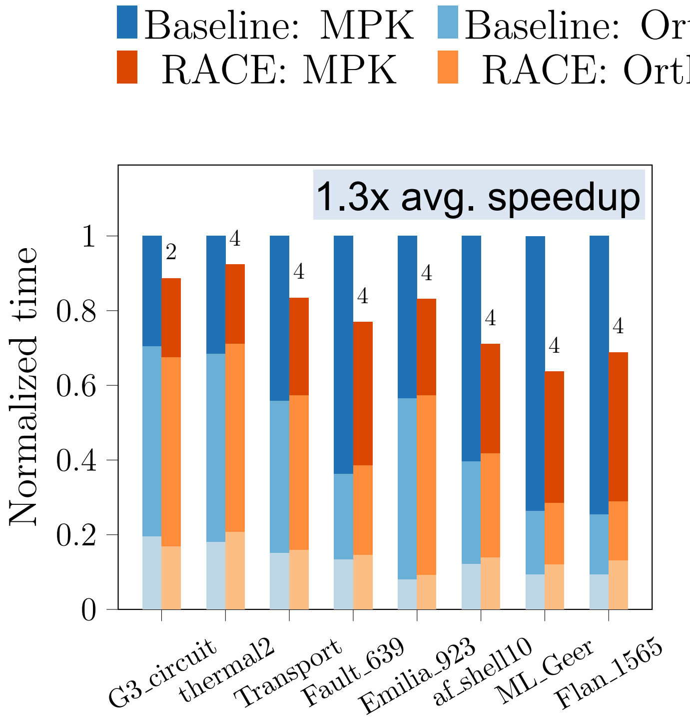
Also called CA-GMRES

Available with Trilinos framework.

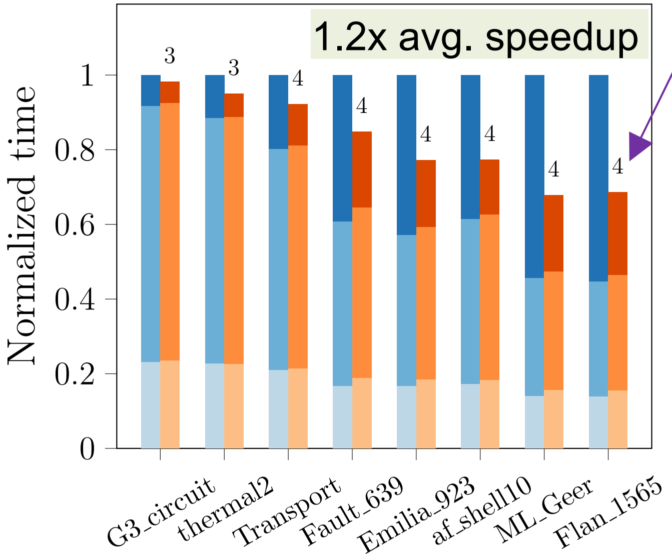
RACE integrated with Trilinos to accelerate MPK.

Mark Hoemmen, 2010, Communication-avoiding Krylov subspace methods, PhD Thesis,
https://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-37.pdf

s-step GMRES



Intel Icelake 8368



AMD EPYC 7662

Internal power value at which RACE operates.

Overall speedup limited by Ortho routines.

On AMD, BLAS kernels used for Ortho are not optimal.

s-step GMRES: Belos (TPETRA) library

settings: s=4, restart length (m)=50

MPK and preconditioners

Solve for x: $AM^{-1}u = b$, $M^{-1}u = x$

Preconditioning s-step GMRES

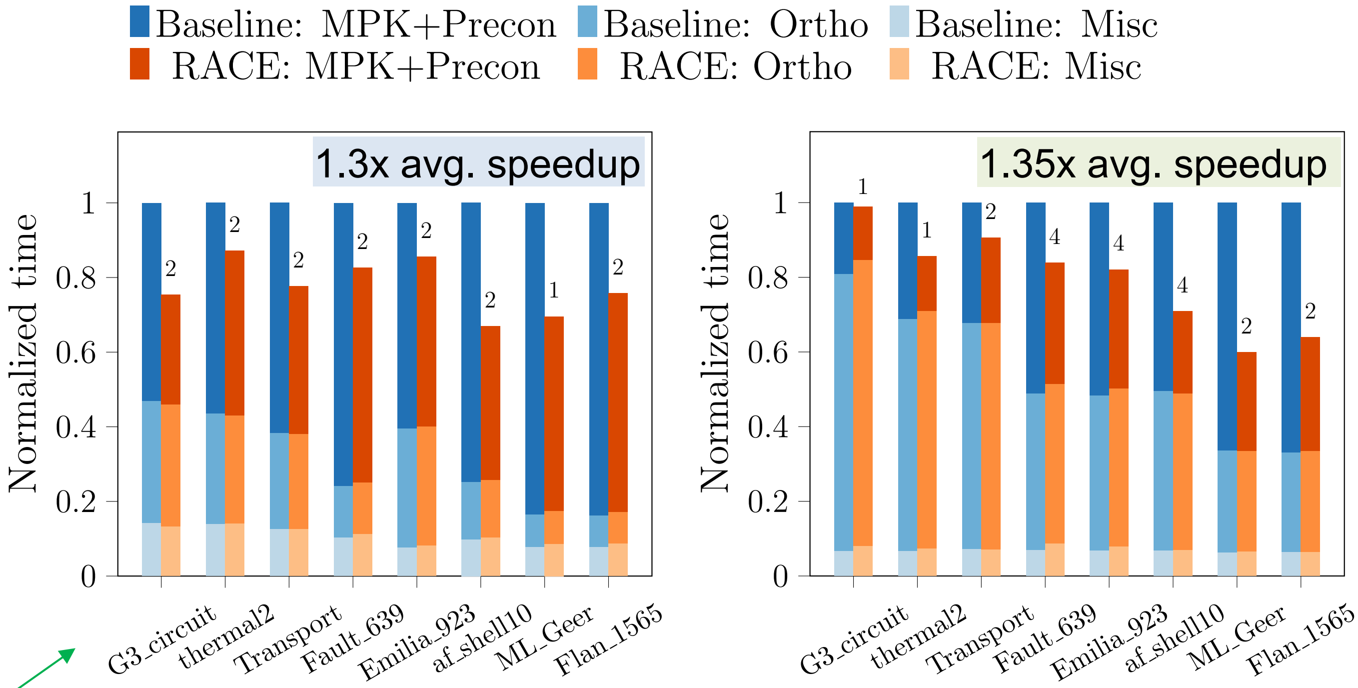
Combine Precon with MPK:

$A^s x \rightarrow (AM^{-1})^s x$

Pure MPK:
$A \rightarrow A^2 \rightarrow A^3 \rightarrow$ ...

Additional dependencies

MPK with Precon:
$M^{-1} \rightarrow AM^{-1}$
$\rightarrow M^{-1}AM^{-1} \rightarrow (AM^{-1})^2$
$\rightarrow$ ...



Intel Icelake 8368

AMD EPYC 7662

Ortho cost reduces

s-step GMRES: Belos, Precon: Ifpack2

settings: Two-stage GS (GS2), γ=2

Polynomial preconditioner

- Use matrix polynomials $\underbrace{(c_1 Ax + c_2 A^2 x + \ldots + c_d A^d x)}_{\text{MPK}}$ to approximate A^{-1}

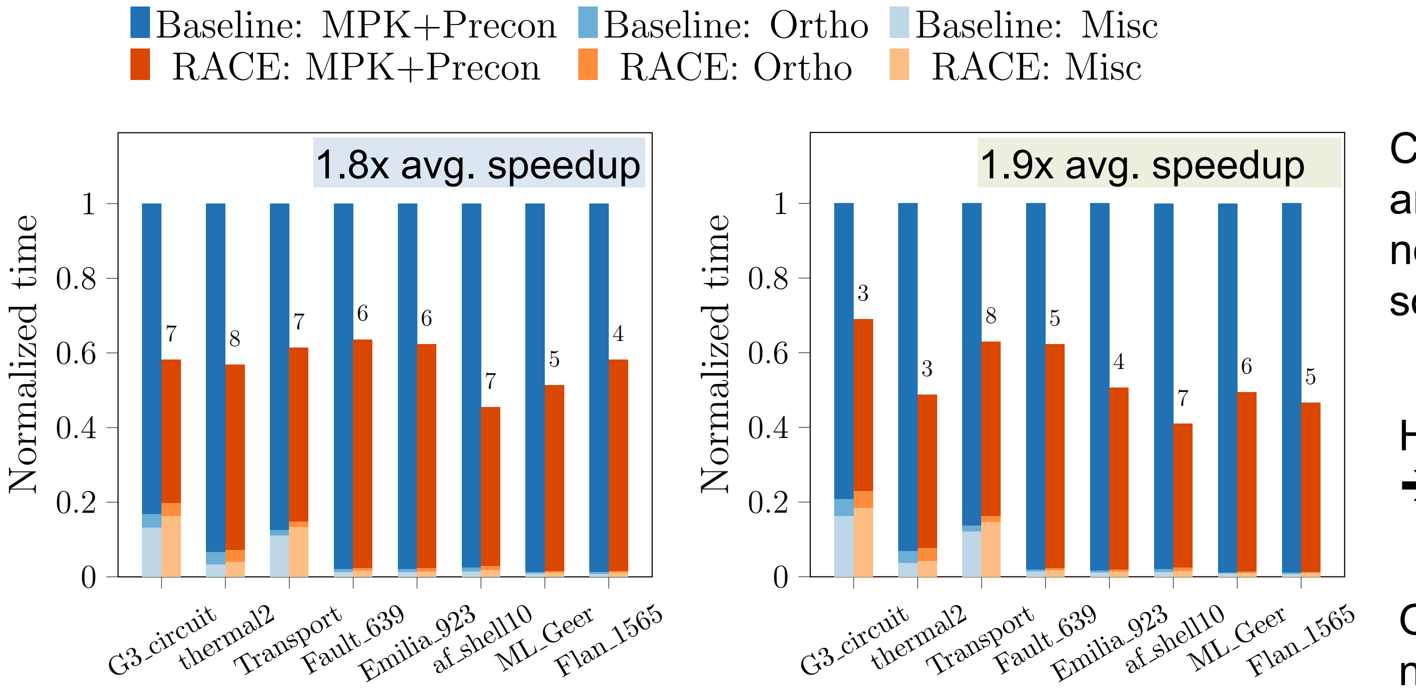
- Higher degree (d) of polynomial ➔ Better approximation

- Can use preconditioners like Jacobi, ILU on top of it

- Available in Trilinos*

*J. Loe, H. Thornquist, E. Boman, 2020, Polynomial Preconditioned GMRES in Trilinos: Practical Considerations for High-Performance Computing, https://doi.org/10.1137/1.9781611976137.4

Polynomial preconditioner with GMRES



Intel Icelake 8368

AMD EPYC 7662

Can be combined with any Krylov solvers. Does not strictly need s-step solvers.

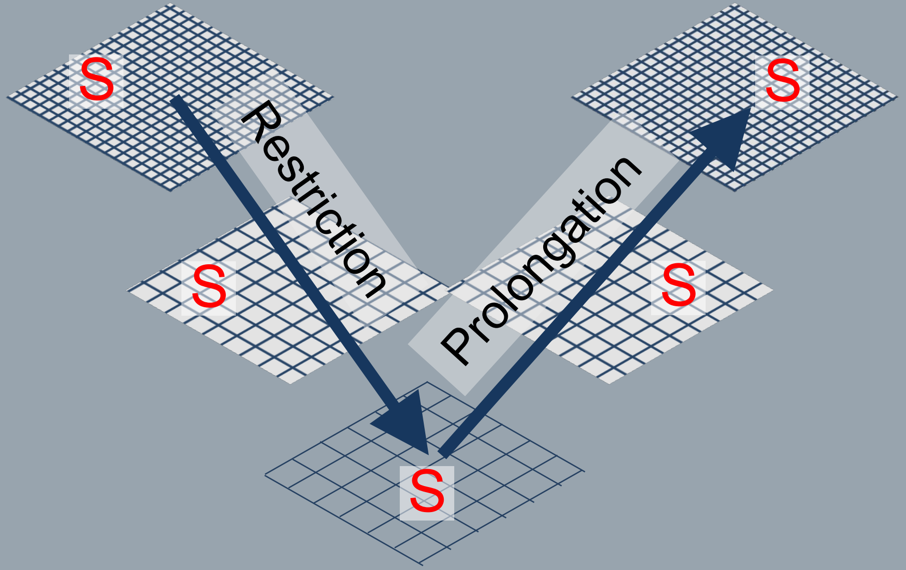
High powers in MPK
➔ RACE very effective.

Ortho cost becomes negligible.

GMRES polynomial precon: Belos

Poly+Jacobi precon, degree (d)=80

Algebraic multigrid (AMG)

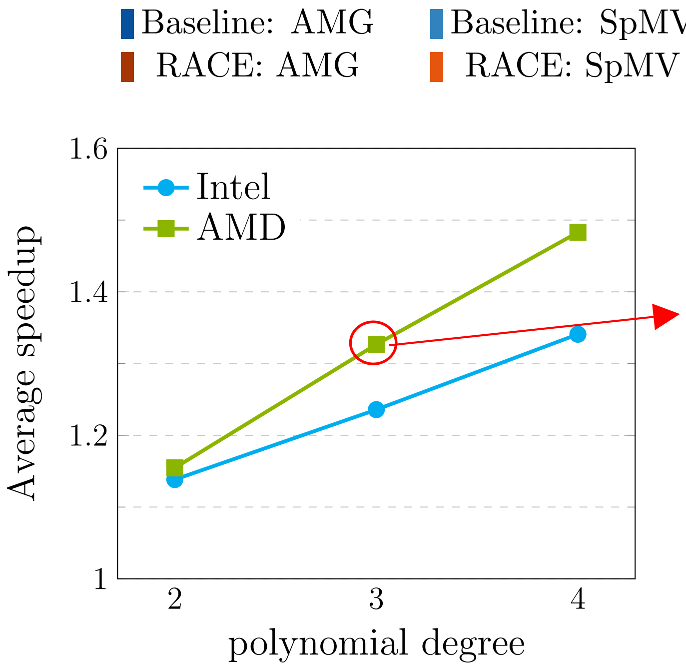


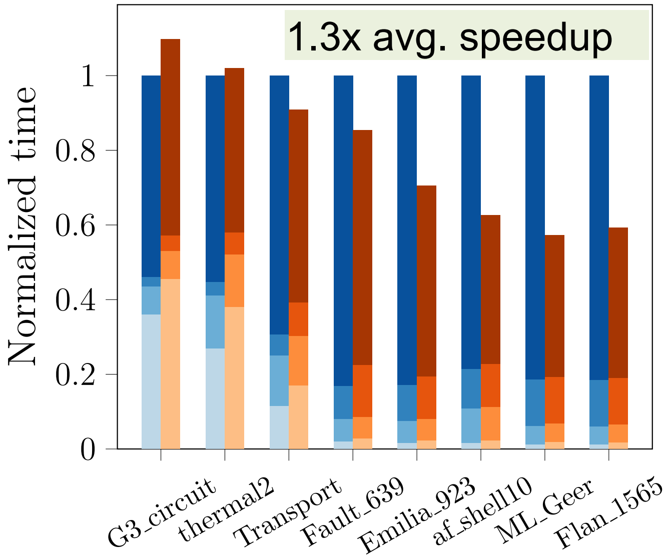
AMG

Smoothers involve back-to-back application of the same operation.

➔ Use RACE to cache block the smoother.

Currently RACE applied only to the finest level.





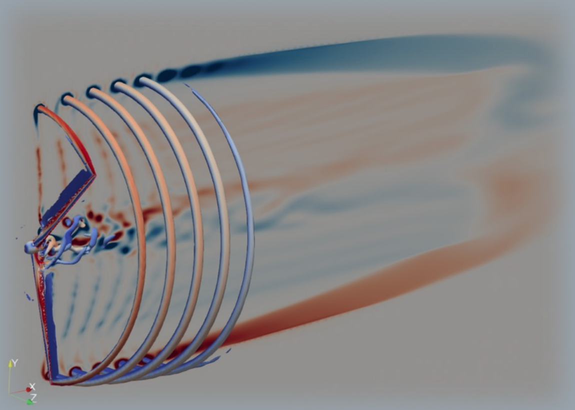
GMRES: Belos, SA-AMG: Muelu

settings: Chebyshev smoother

AMD EPYC 7662

degree-3

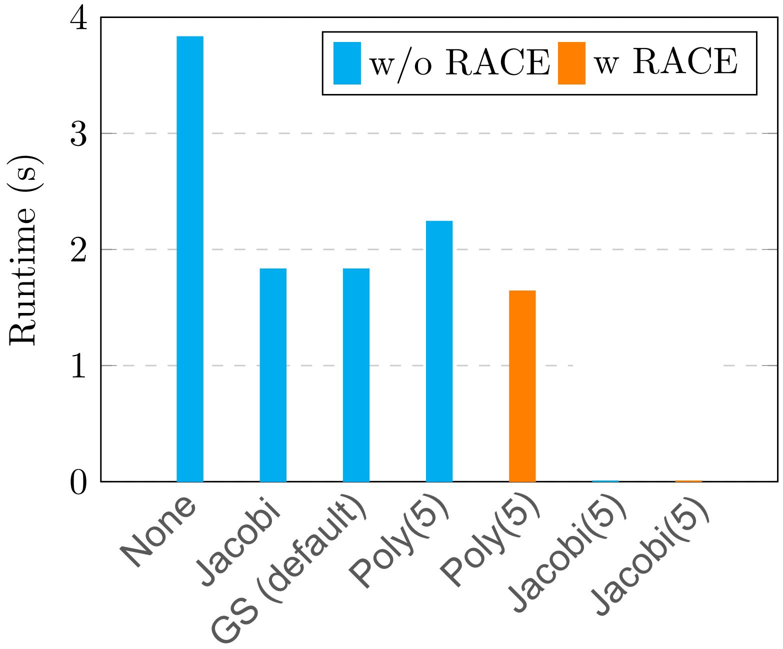
Case study: Nalu-Wind



Wind turbine simulation using Navier-Stokes equation on unstructured grid.

Picture taken from: Sprague et al., ``ExaWind: A multifidelity modeling and simulation environment for wind energy'', Journal of Physics, 2020, 10.1088/1742-6596/1452/1/012071

Case study: Nalu-Wind



In traditional setting polynomial preconditioner is not beneficial.

But RACE can change the picture.

➔ Adds another dimension to solver choice.

RACE can accelerate multiple sweeps of the same preconditioner ☺

GMRES solver runtime for solving momentum equation of dimension $N_r \approx 12\ M$, $N_{nz} = 300\ M$

Summary

- MPK kernel's performance can be improved by level-based cache blocking using RACE.
- Speedups up to 5x possible.
- Benefits iterative solvers and its components: s-step Krylov solvers, polynomial preconditioners, AMG, …
- RACE adds another dimension to solver selection/tuning.

Outlook

- Solvers like s-step GMRES and polynomial preconditioners are easy to parallelize and have lower communication overheads ➔ promising for large-scale solvers.
- MPI parallel version of RACE already in deployment.
- Cache blocking on GPU produces first successful results.

Thank you

Questions

RACE
Hardware Friendly Coloring

https://github.com/RRZE-HPC/RACE

C. Alappat, G. Hager, O. Schenk and G. Wellein, "Level-Based Blocking for Sparse Matrices: Sparse Matrix-Power-Vector Multiplication," in IEEE Transactions on Parallel and Distributed Systems, 2023, doi: 10.1109/TPDS.2022.3223512.

C. Alappat, A. Basermann, A.R. Bishop, H. Fehske, G. Hager, O. Schenk, J. Thies, and G. Wellein. 2020. A Recursive Algebraic Coloring Technique for Hardware-efficient Symmetric Sparse Matrix-vector Multiplication. ACM Trans. Parallel Comput., 2020. https://doi.org/10.1145/3399732