University of Minnesota

Supercomputing Institute

A Massively Parallel Linear Solver and Its Applications in Data Analysis

Vasileios Kalantzis and Yousef Saad, Computer Science and Engineering Department

Introduction

We consider the parallel solution of dense linear systems with multiple right-hand sides,

$$AX = Z,$$

where $A \in \mathbb{R}^{n\times n}$ is SPD and $Z \in \mathbb{R}^{n\times s}$ is the matrix of s right-hand sides.

The algorithm

The proposed numerical algorithm is based on the block Conjugate-Gradient (BCG) method combined with a sophisticated scheme of Galerkin oblique projections. A sketch of the method (abbreviated as PP-BCG) follows:

1. Choose $p < s$ systems and solve them by the BCG method. Store the Krylov subspace $\mathcal{P}$ formed in the course of the solution.
2. Deflate $\mathcal{P}$ from the remaining s-p systems through a series of Galerkin projections.
3. Solve the remaining s-p systems by the BCG method. Convergence should be much faster because of the Galerkin projections.
4. Estimate the diagonal of A^{-1}, $\mathcal{D}$.

The following table shows the order of the relative error of the approximation as well as the speedup obtained by using PP-BCG instead of BCG method. Matrix A was selected as a model covariance matrix and $p = s/10$.

s	$n = 2000$	$n = 4000$	PP-BCG/BCG
20	$O(10^{-4})$	$O(10^{-3})$	1.4
50	$O(10^{-4})$	$O(10^{-3})$	1.9
100	$O(10^{-5})$	$O(10^{-4})$	2.3

Applications

- **Uncertainty quantification:** Stochastic estimation of the diagonal of A^{-1}.
- **MLE:** Stochastic estimation of the trace of A^{-1}.
  Both applications share $AX = Z$ as their main computational kernel!

Parallel implementation

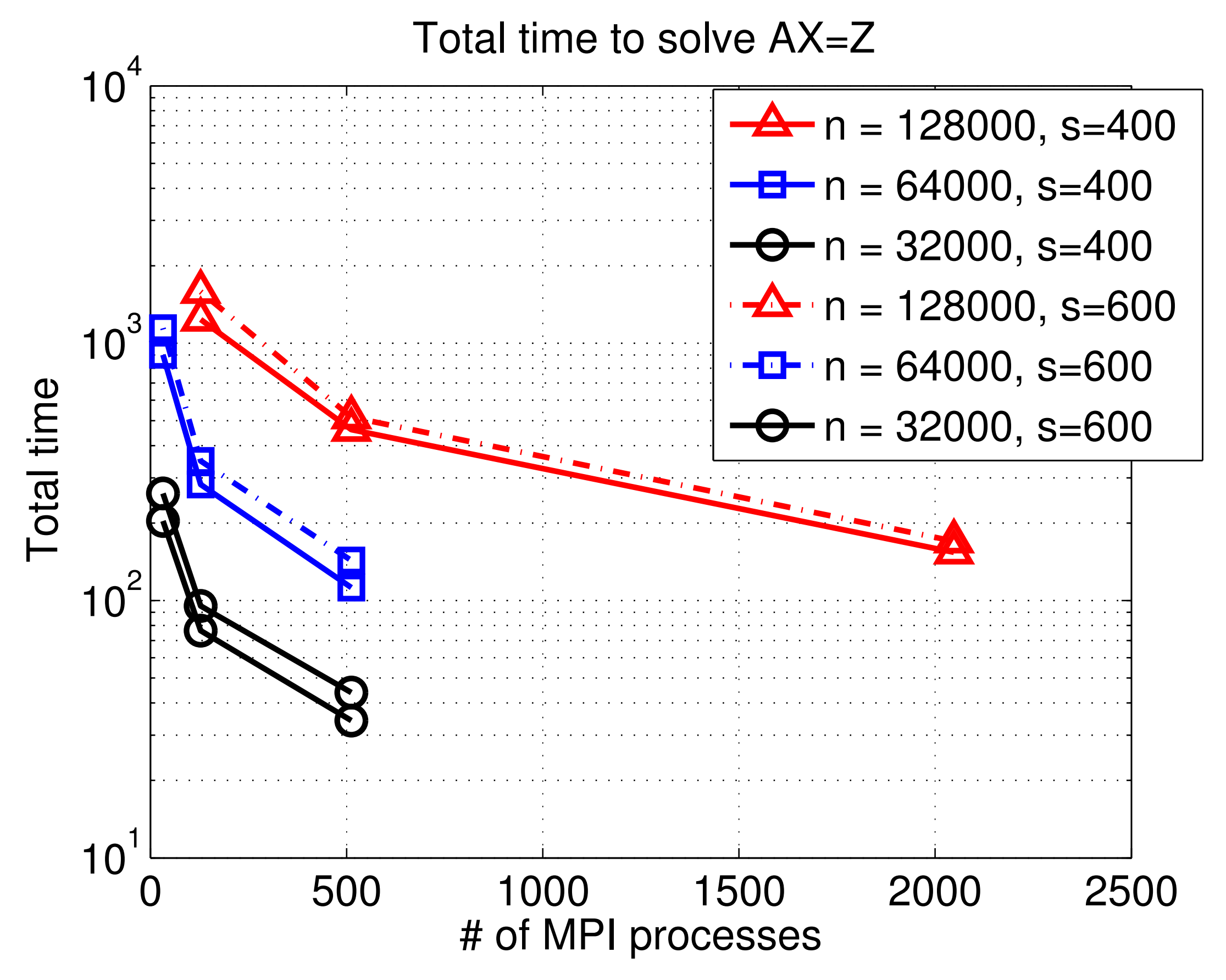
We implemented our algorithm in a 2-D grid of processors. All communication was performed by means of the Message Passing Interface (MPI) and we used one core per MPI process (distributed memory model only).

The main computational module of our scheme is the Matrix-MultiVector product at each iteration of BCG. Additionally, we have to perform block inner products and block AXPY operations. The AXPYs are perfectly parallel while the inner products are computed using the `mpi_allreduce` command. For the Matrix-MultiVector product we implemented the following scheme:

- Processors of the first row broadcast their local portion along their column (`mpi_bcast`).
- Each processor performs the local Matrix-MultiVector product (`dgemm`).
- Processors perform a row reduction to sum the local products (`mpi_reduce`).

Our implementation is written in Fortran 90 and is linked with the Intel MKL library to make use of the associated BLAS & LAPACK routines. All experiments performed at the Itasca Linux cluster hosted at MSI (`http://www.msi.umn.edu`).

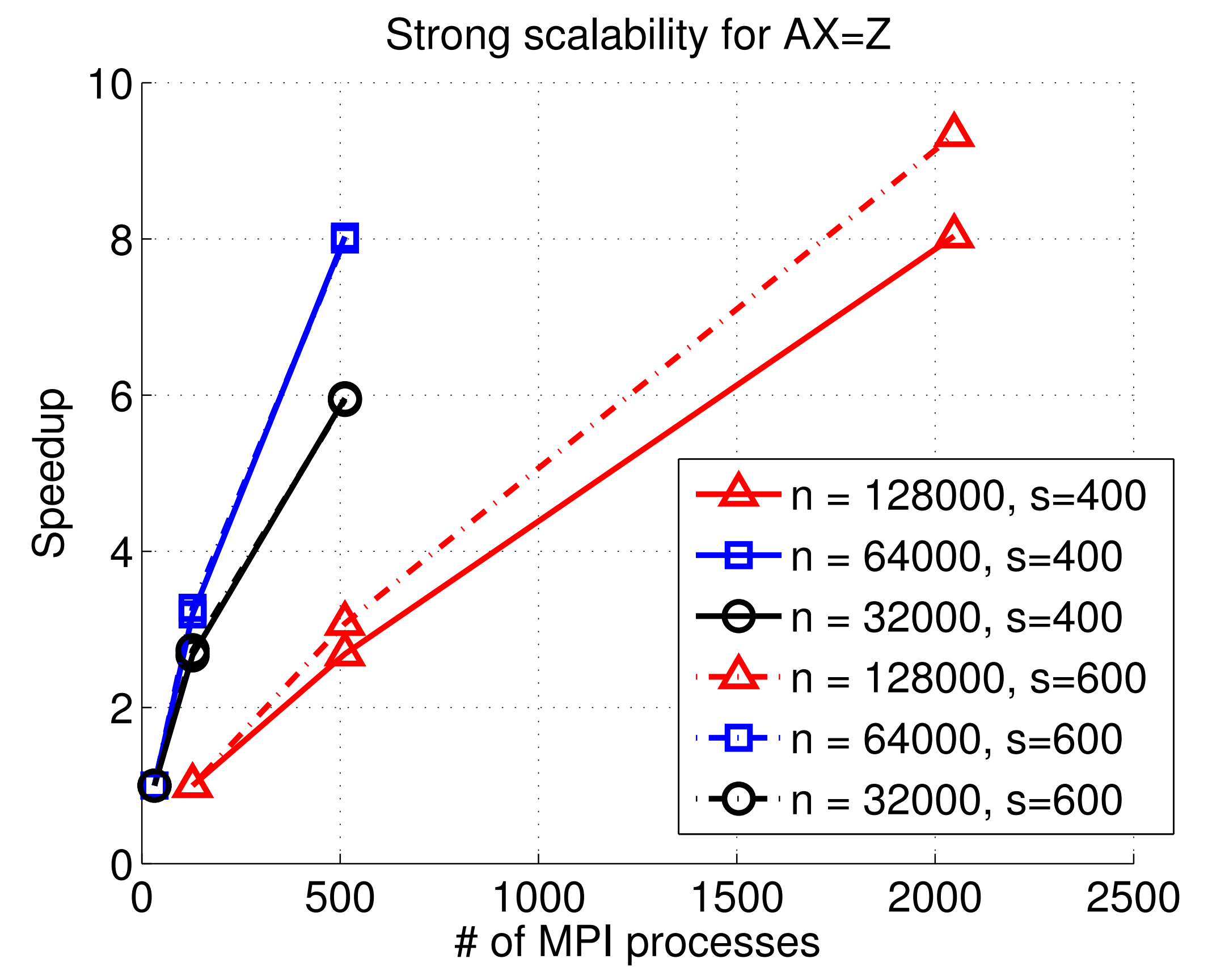
Strong scalability and performance

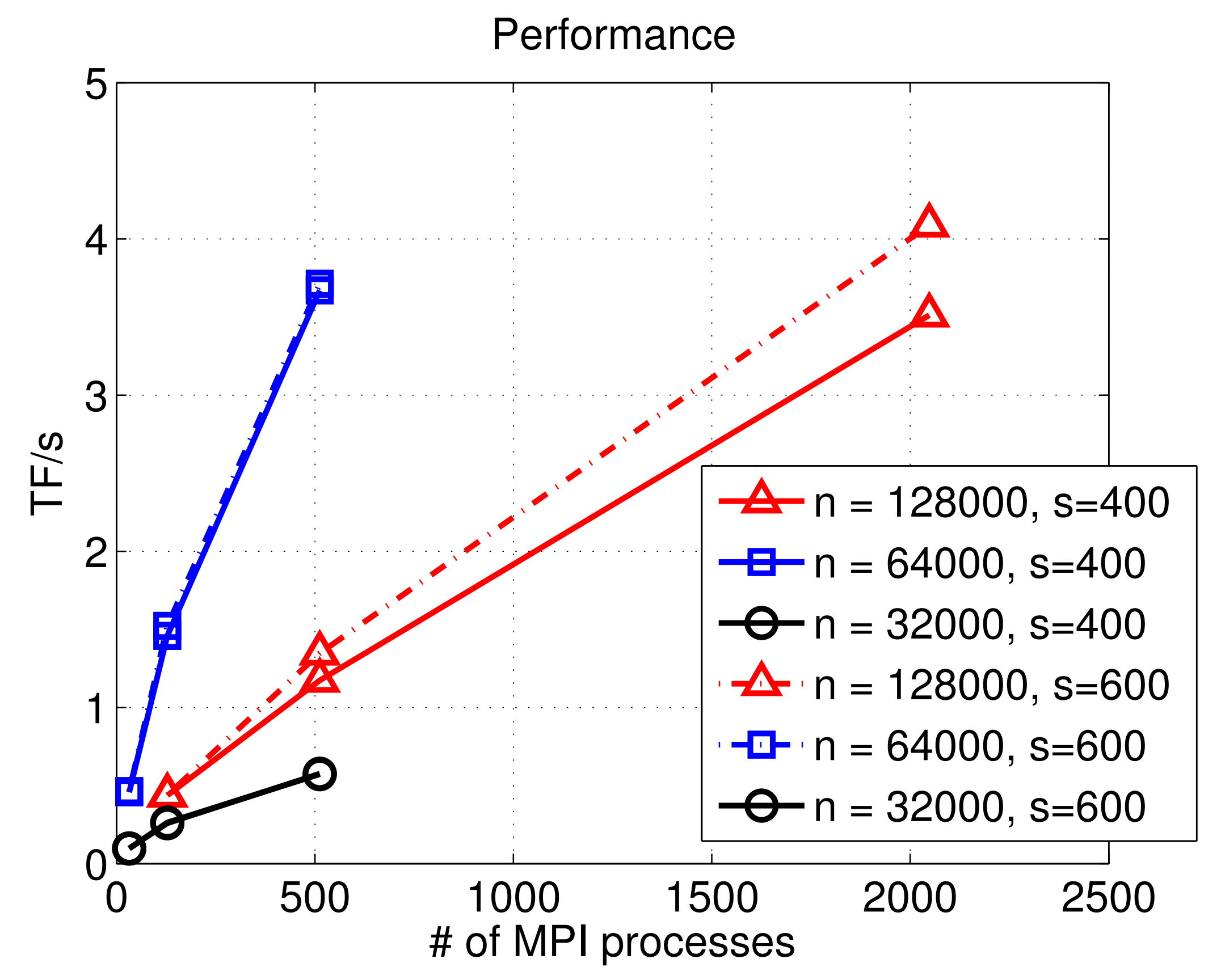


The model covariance matrices were generated as

$$A_{ii} = 1 + i^{\theta}, \quad A_{ij} = 1/|i-j|^{\kappa}, \quad i,j = 1,\ldots,n,$$

with $\theta = 0.8$ and $\kappa = 2$. Columns of Z selected as Rademacher vectors. We report results for $n = 32000,\ 64000,\ 128000$, $s = 400,\ 600$ and $p = s/10$. The top figure shows the actual wall-clock timings when solving $AX = Z$ while the bottom figures show the relative speedups and performance in TeraFlop/s when using 32, 128, 512 and 2048 MPI processes (cores). For the largest problem, we achieved a performance of more than 4 TF/s!





Comparisons against the state-of-the-art method

Finally, we present comparisons between our scheme, PP-BCG, and the current state-of-the-art method used to solve $AX = Z$, the BCG method. We selected $n = 128000$ with the same parameters as above. We solved $s = 200$ right-hand sides with PP-BCG and BCG methods for a different number of MPI processes (cores). Time is listed in seconds.

Our method is scalable and also faster than BCG.

# cores	BCG	PP-BCG
128	184.6	132.5
256	86.0	63.2
512	50.8	37.1

References

[1] V. Kalantzis et al. Accelerating data uncertainty quantification by solving linear systems with multiple right-hand sides. Numer Alg., 2013.

[2] V. Kalantzis et al. A massively parallel linear solver and its applications in data analysis. TR, 2015.

Conclusion

We presented a massively parallel linear solver for the efficient solution of multiple systems with the same coefficient matrix. Applications include Uncertainty Quantification and MLE.

The method implemented in distributed environments and proved superior to the current state-of-the-art methods. Experiments performed in up to $2K$ cores demonstrated the scalability of the scheme.

Acknowledgments

We are grateful to the University of Minnesota Supercomputing Institute for providing us with computational resources and assistance with the computations.