A Splitting Approach for the Parallel Solution of Large Linear Systems on GPU Cards

Ang Li ali28@wisc.edu

Radu Serban serban@wisc.edu

Dan Negrut negrut@wisc.edu

Simulation Based Engineering Lab (SBEL)
www.sbel.wisc.edu
Department of Mechanical Engineering, University of Wisconsin – Madison

Background and Motivation

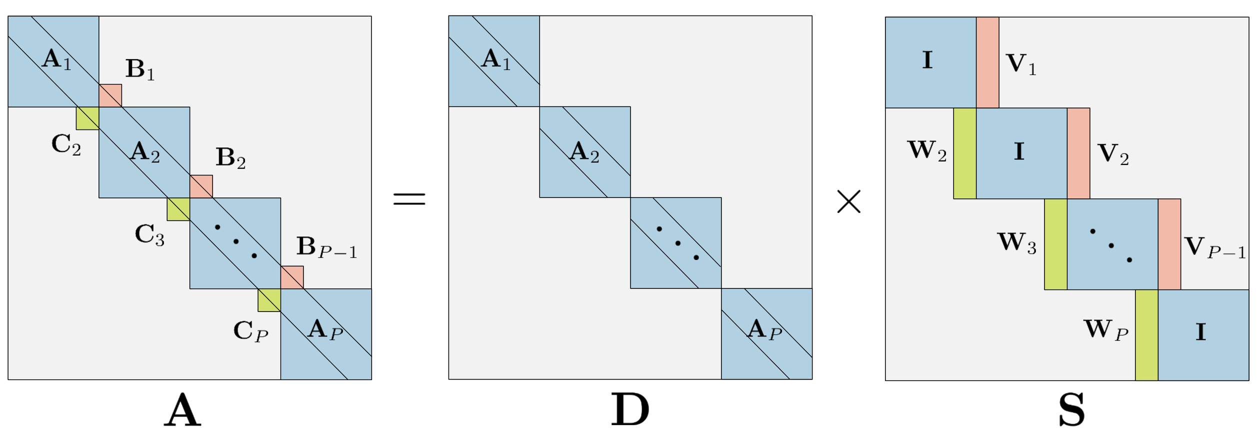
The task of solving a linear system is ubiquitous in numerous computational problems such as implicit integration of Ordinary Differential Equation (ODE) and Differential Algebraic Equation (DAE) problems, numerical solution of Partial Differential Equations (PDE), interior point optimization methods, least squares approximations, solving eigenvalue problems, and data analysis.

We discuss an approach for solving sparse or dense banded linear systems on a GPU card. The solver based on this approach, **SaP::GPU** (**SaP** for Splitting and Parallizing) developed in the Simulation Based Engineering Lab (SBEL), is compared in terms of efficiency with Intel's **MKL** solver (for dense banded matrices). We also compare **SaP::GPU** in terms of efficiency and robustness against three commonly used sparse direct solvers: **PARDISO**, **SuperLU**, and **MUMPS**. **SaP::GPU** is surprisingly robust and compares well in terms of efficiency with the aforementioned direct solvers.

SaP::GPU Overview of Methodologies

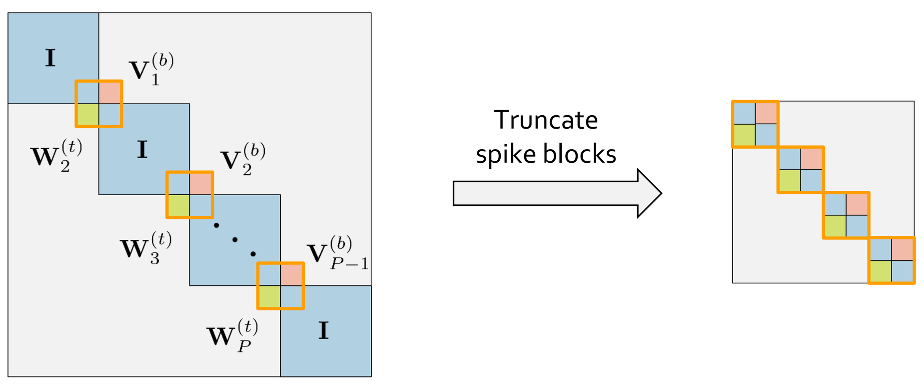
Banded Matrices. Assume that banded matrix $\mathbf{A}_{N\times N}$ has half-bandwidth $K \ll N$.

SaP::GPU partitions the matrix **A** into *P* blocks and decomposes it into a block-diagonal matrix **D** and a spike matrix **S**:



The goal it to produce a preconditioner and employ a Krylov-subspace iterative solver. Since **Ax = b** ⇔ **Dg = b** & **Sx = g**, we seek efficient ways to approximate **S**:

- Decoupled approach: **SaP::GPU-D**$_d$ (band-block-diagonal preconditioner)
- Coupled approach: **SaP::GPU-C**$_d$ (modified truncated SPIKE preconditioner)



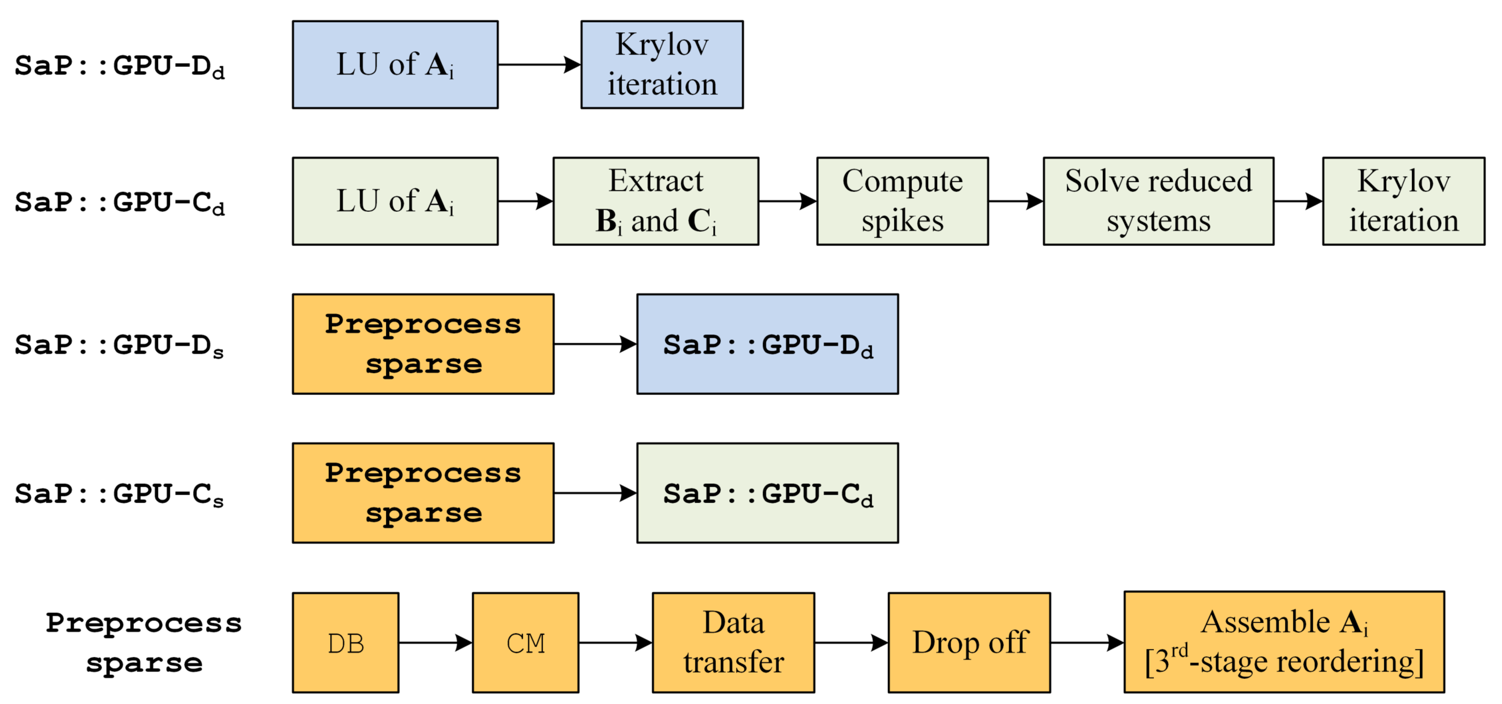
Use as a preconditioner for a Krylov iterative solver, e.g. BiCGStab2.

Sparse matrices. In a preprocessing, we perform row/column reorderings for diagonal boosting (DB*) and bandwidth reduction (CM**), and optional drop-off. A third stage reordering for bandwidth reduction is applied on each sub-matrix $\mathbf{A}_i$ to improve performance.

* DB: carried out to move heavy matrix entries onto the diagonal. We perform LU with no pivoting

** CM: a Cuthill-McKee, bandwidth reduction algorithm on a sparse matrix with a symmetric sparsity pattern. We implemented a hybrid GPU-CPU CM algorithm in a stage denoted as CM.

Computational flow.



Testing Environment and Methodology

GPU: Tesla K20X GPU card (Kepler architecture)

CPU: 2 x Xeon E5-2690v2

Method: Compare **SaP::GPU** against Intel's **MKL** banded solver on synthetic banded matrices, and against **PARDISO**, **SuperLU**, and **MUMPS** on application matrices obtained from the University of Florida Sparse Matrix Collection and Simulation Based Engineering Lab.

References

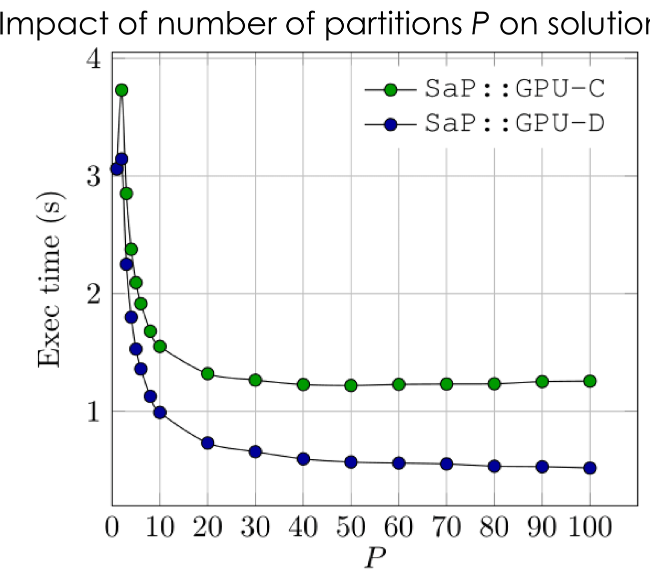
- A. Li, R. Serban, and D. Negrut, *Analysis of a splitting approach for the parallel solution of linear systems on GPU cards*, under review, SIAM Journal on Scientific Computing, 2015
- E. Polizzi and A. Sameh, *A parallel hybrid banded system solver: the SPIKE algorithm*, Parallel Computing, 32(2):177-194, 2006

Results for Banded Dense Matrices

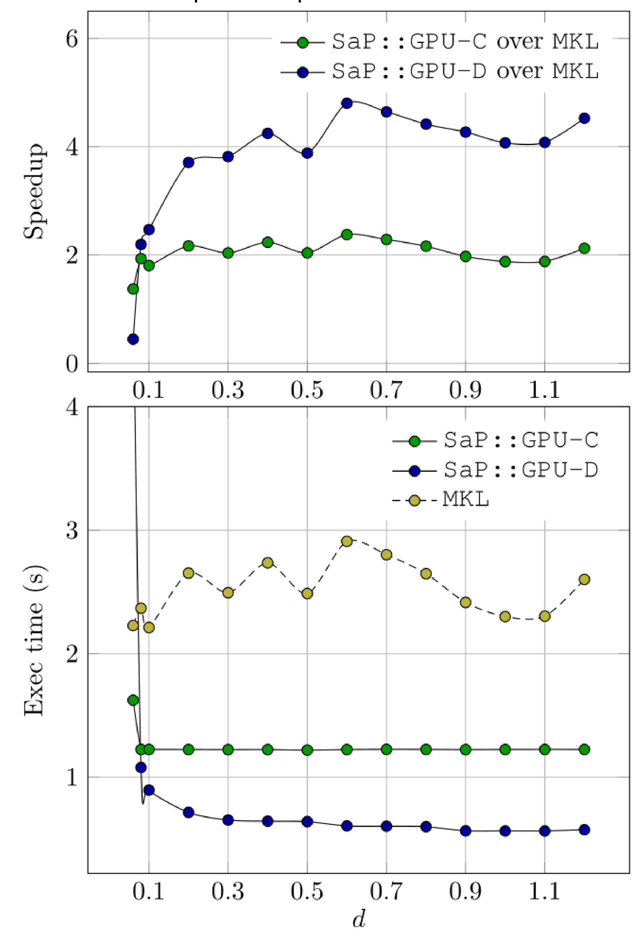
The results show the sensitivity of **SaP::GPU**'s execution time with respect to the number of partitions *P* (varied from 1 to 100) and the degree of diagonal dominance* (varied from 0.06 to 1.2). The matrix size is fixed to dimension N = 200 000 and half-bandwidth K = 200.

* Degree of diagonal dominance: the maximum of *d* which satisfies $\forall i\ |a_{ii}| \geq d \sum_{j\neq i} |a_{ij}|$

Impact of number of partitions *P* on solution time



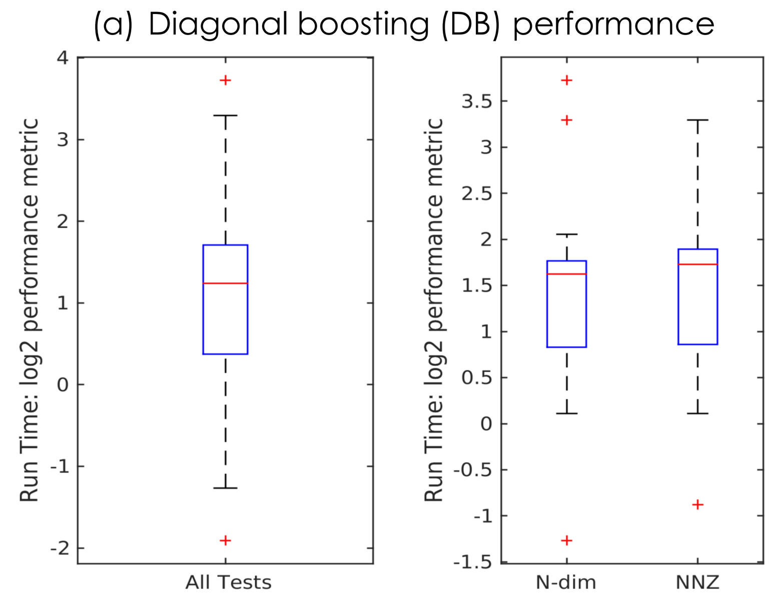
Impact of degree of diagonal dominance. Relative speedup over Intel's MKL



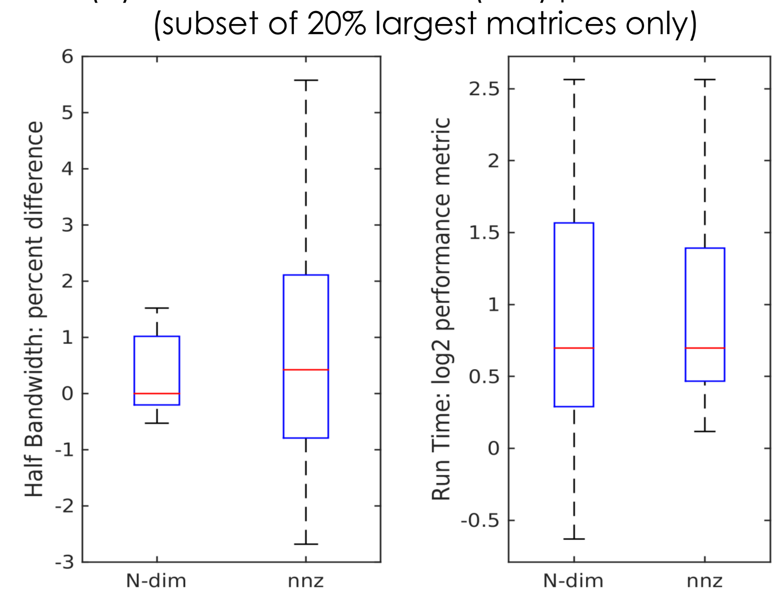
Matrix Reordering Issues

Diagonal boosting and bandwidth reduction are key stages in the solution process. Against this backdrop we compare **SaP::GPU** to Harwell Subroutine Library (HSL) on a set of 125 matrices. We also considered a second, smaller sample subset of the 20% largest matrices (separately on matrix dimension N-dim, and on number of nonzero elements NNZ).

(a) Diagonal boosting (DB) performance



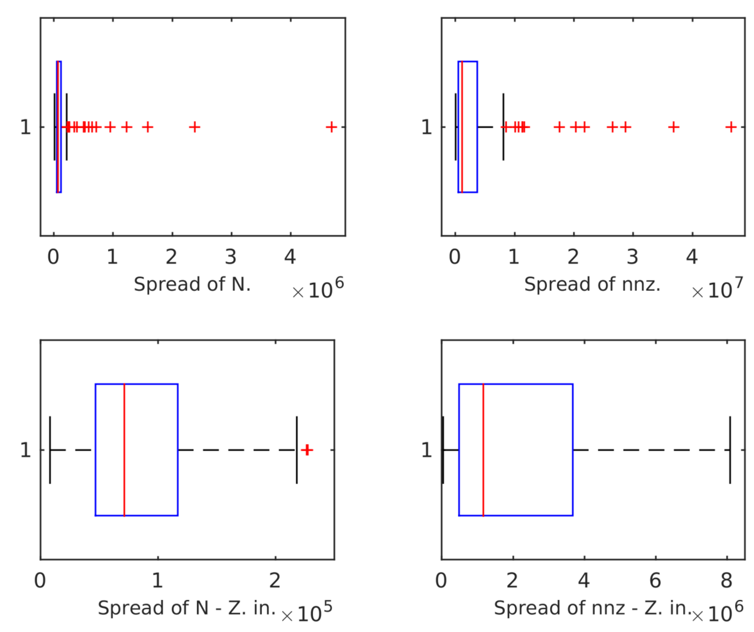
(b) Bandwidth reduction (CM) performance (subset of 20% largest matrices only)



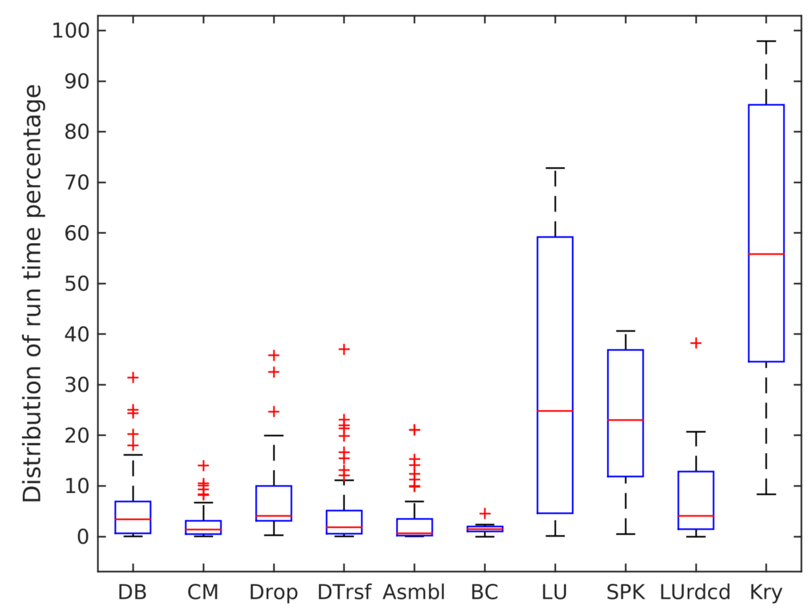
Results for Sparse Matrices, a Statistical Analysis

Figure (a) shows the problem sizes *N* encountered in this statistical analysis, which drew on 114 real application matrices. Fig. (b) shows time breakdown info on the solution of the sparse linear systems. Fig. (c) shows the performance comparison with **PARDISO**, **SuperLU**, and **MUMPS**. Table (d) shows the impact of the third-stage reordering.

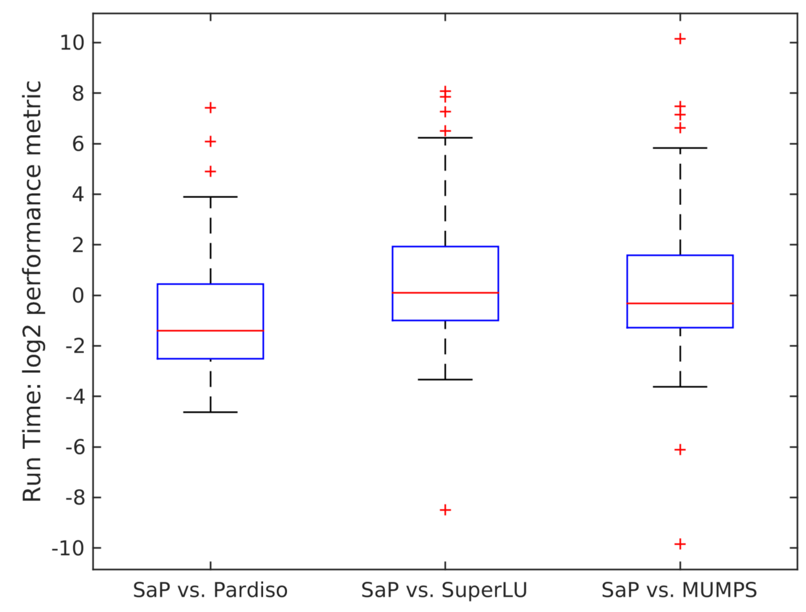
(a) Statistics on dimension and number of nonzeros (from 114 real application matrices)



(b) Statistics on **SaP::GPU** profiling (including the Krylov stage)



(c) Statistics on **SaP::GPU** comparison against **PARDISO**, **SuperLU**, and **MUMPS**



Out of 114 sparse linear systems considered, there are several observations from solver comparison:

- 57 were solved by both **SaP** and **PARDISO**. **SaP** was faster than **PARDISO** for 20 of these 57 linear systems.
- 60 were solved by both **SaP** and **MUMPS**. **SaP** was faster than **MUMPS** for 27 of these 60 linear systems.
- 71 were solved by both **SaP** and **SuperLU**. **SaP** was faster than **SuperLU** for 38 of these 71 linear systems.
- NOTE: Neither **MUMPS** nor **SuperLU** was ever faster than **PARDISO**.

Conclusions. Future Work

- **SaP::GPU** as a dense banded solver is fast. Over half of the tests run more than twice faster than Intel's **MKL**.
- As a sparse solver, **SaP::GPU** is slower than **PARDISO** and **MUMPS** but more robust. It's faster but less robust than **SuperLU**.
- Next step: multi-GPU solution. Emphasize hybrid CPU/GPU implementation.

http://sapgpu.sbel.org/

Simulation Based Engineering Lab
University Of Wisconsin - Madison

SuperComputing 2015