Background and Motivation

The task of solving a linear system is ubiquitous in numerous computational problems such as implicit integration of Ordinary Differential Equations (ODE) and Differential Algebraic Equations (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 Parallelizing) 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 $A_{i,j}$ has half-bandwidth $k \leq N$.

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

$$
A = D + S
$$

The goal is to produce a preconditioner and employ a Krylov-subspace iterative solver. Since $Ax = b \Leftrightarrow Dg + Sx = b$, we seek efficient ways to approximate $S$:

- Decoupled approach: SaP::GPU-$D_0$ (band-block-diagonal preconditioner)
- Coupled approach: SaP::GPU-$C_0$ (modified truncated SPIKE preconditioner)

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, simple subset of the 57 largest matrices (separately on matrix dimension N-dim, and on number of nonzero elements NNZ).

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 $N = 200000$ and half-bandwidth $K = 200$.

* Degree of diagonal dominance: the maximum of $\bar{a} = \frac{\sum |a_{ii}|}{\sum |a_{ii}|}$

Impact of number of partitions $P$ on solution time

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.

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.

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.

References