TR-2014-07

# A Comparison of the Performance of SaP::GPU and Intel's Math Kernel Library (MKL) for Solving Dense Banded Linear Systems

Ang Li, Omkar Deshmukh, Radu Serban, Dan Negrut

May 24, 2015

**Abstract**

`SaP::GPU` is a solver developed in the Simulation Based Engineering Lab (SBEL) [1] to solve large banded and sparse linear systems on the GPU. This report contributes the performance comparison of the banded solver of `SaP::GPU` and Intel's Math Kernel Library [2] on a large set of synthetic problems. The results of several numerical experiments indicate that when it is used in conjunction with large dense banded matrices, `SaP::GPU` is two to five times faster than the latest version of the `MKL` dense solver when the latter is run on the Haswell, Ivy Bridge, or Phi architectures.

# Contents

# 1 Introduction

`SaP::GPU` is a solver developed by Simulation Based Engineering Lab (SBEL) [1] to solve large banded and sparse linear systems on GPU. The solver uses preconditioned Krylov-subspace method and the truncated version of SPIKE algorithm [5–7] works as its preconditioner. By taking use of the floating-point processing ability and high bandwidth of GPU memory, and by leveraging the parallelism in truncated SPIKE, `SaP::GPU` is claimed to achieve high speedups compared to Intel's Math Kernel Library in a large set of synthetic banded matrices [3].

Intel's Math Kernel Library (MKL) [2] includes a wealth of routines to accelerate application performance and reduce development time. It includes highly vectorized and threaded Linear Algebra, Fast Fourier Transforms (FFT), Vector Math and Statistics functions. Through a single C or Fortran API call, these functions automatically scale across previous, current and future processor architectures by selecting the best code path for each.

We worked with factorization of banded matrices with routine `dgbtrf` and routine `dgbtrs` for the solution. The routine `dgbtrf` computes the LU factorization of a general banded $m$-by-$n$ matrix $\mathbf{A}$ with lower half-bandwidth $K_l$ and upper half-bandwidth $K_u$ (in our cases, however, we are working with test cases where $m = n$ and $K_l = K_u$. See §2.1 for detail). Following `dgbtrf`, by calling the routine `dgbtrs`, we will be able to achieve the solutions for the factorized banded system.

The remainder of this report is organized as follows. In §2 and §3, we introduce our test methods and test environment, respectively. In §4, we display plots and briefly summarize the observation from these plots.

# 2 Test Methods

This section briefly outlines the parameters of synthetic banded matrices, the methods to generate synthetic banded matrices, and how we make sure the comparison is apple-to-apple.

## 2.1 Parameters

As described in §1, we intend to compare `SaP::GPU` with `MKL` on matrices of various sizes. Three parameters were used to control a generated matrix $\{a_{ij}\}_{n\times n}$.

- **Dimension.** The dimension $n$ directly tells how large a matrix is. In §4, we use $N$ to denote the dimension.
- **Half-Bandwidth.** The half-bandwidth is defined as $\max_{i,j,a_{ij}\neq 0} |j-i|$. This parameter influences the number of floating point operations required in each iteration of both factorization and foward/backward sweeps. In §4, we use $K$ to denote the half-bandwidth
- **Degree of diagonal dominance.** The degree of diagonal dominance is defined as $d = \min_i \sup\{d_i : |a_{ii}| \geq d_i \cdot \sum_{j\neq i} |a_{ij}|\}$. This parameter intuitively shows compared

to all other entries, how large the diagonal entries are. As `SaP::GPU` uses truncated version of SPIKE algorithm as a preconditioner and [5,6] claims that truncated version is theoretically only applicable to matrices with $d \geq 1$, we are interested in observing how `SaP::GPU` behaves with matrices with $d < 1$. In this report, sometimes we will use the term "diagonal dominance" for simplicity.

In our tests, we generate sets of matrices, varying in these three parameters. The comparison is not an entire four dimensional comparisond, but one with two parameters fixed while the rest one is changing.

### 2.2 Matrix generation method

First of all, we let the user to specify the three parameters described in §2.1. Then for each non-zero matrix entry, a random value between $-10$ and 10 is assigned. At last, when all non-zero entries have been generated, for the entries except for the diagonal one in each row, we sum their absolute values up and replace the corresponding diagonal entry with value $d \cdot \sum_{j \neq i} |a_{ij}|$, where $d$ is the user-specified diagonal dominance.

### 2.3 Apple-to-apple guarantee

To make sure `SaP::GPU` and `MKL` are solving exactly the same set of problems, the matrices were all generated in `SaP::GPU` and were stored in a temporary file. After `SaP::GPU` has completed its job, `MKL` directly reads the file, forms its matrix and then solves. The right hand sides were assigned to all ones. For details of banded matrix storage for `SaP::GPU` and `MKL`, interested readers can refer to [3] and [2], respectively.

## 3 Test Environment

All numerical experiments for `SaP::GPU` were performed on a machine with an Intel Nehalem Xeon E5-2630 processor (in all figures, it is denoted as `Euler99`) and an NVIDIA Tesla K40 GPU accelerator [4]. We ran `MKL` on two machines for comparison: Intel Nahalem Xeon E-2630 (denoted as `Euler99`) and Intel Nahalem Xeon E5-2690v2 (denoted as `Lagrange`). The solution times reported are inclusive time that account for the amount of time to move data to/from the device. Also, where applicable, the solution times include times required to do the necessary matrix reorderings.

The GPU code was built with the `nvcc` compiler in CUDA version 6.0, while all CPU codes were built with Intel's compiler ICPC 13.0. In all cases, level 2 optimization was used.

## 4 Test Results

**Varied $N$, fixed $K$ and $d$.** In a first set of experiments, we investigate the performance and scalability of groups of solving matrices with the same half-bandwidth and diagonal

dominance (in this set of tests, we take $d = 1$ all the time) while differing in dimension. We can see that `SaP::GPU` outperforms `MKL` (on `Euler99`) in the whole range of space and outperforms outperforms `MKL` (on `Lagrange`) in the whole range of space except for one point with $(N, K) = (1000, 50)$. For scalability, `SaP::GPU` scales better than `MKL` (both on `Euler99` and `Lagrange`) for half-bandwidths $20, 50$ and $100$, but scales a bit worse for half-bandwidth 200.

**Varied $K$, fixed $N$ and $d$.** In the second set of experiments, groups of matrices with the same dimension and diagonal dominance while the half-bandwidth differs are tested. We can see that `SaP::GPU` outperforms `MKL` (on `Lagrange`) in the whole range of space and outperforms outperforms `MKL` (on `Euler99`) in the whole range of space except for one point with $(N, K) = (200000, 500)$. For scalability, `SaP::GPU` scales better than `MKL` (both on `Euler99` and `Lagrange`) for smaller half-bandwidths, but scales worse for larger half-bandwidths.

**Varied $d$, fixed $N$ and $K$.** Finally, we study how diagonal dominance impacts the efficacity of both `SaP::GPU` and `MKL`. As expected, the `MKL` banded solver is insensitive to changes in $d$ if we discard the variations introduced by the overload of the CPU. Somewhat surprising is the fact that the `SaP::GPU` solver demonstrated uniform performance over the whole range of degrees of diagonal dominance on the matrices in Figure 3; `SaP::GPU` required no more than one outer loop iteration for all $d \geq 0.1$. Our further studies show that, as the degree of diagonal dominance decreases further, the number of iterations and hence the time to solution increase significantly, since the assumptions supporting the truncated SPIKE approximation are strongly violated. What has not surprised us is that we can still see speedups from 2 to 4 almost in the whole range space.

# References


[1] Simulation based engineering lab. http://sbel.wisc.edu, 2006.

[2] Intel Math Kernel Library (Intel MKL) 11.0. http://software.intel.com/en-us/intel-mkl, 2012.

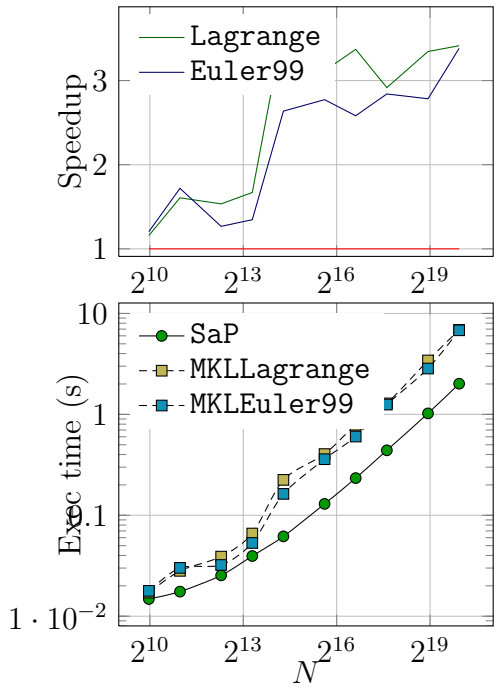
[3] SPIKE GPU: A SPIKE-based preconditioned GPU Solver for Sparse Linear Systems. http://sbel.wisc.edu/documents/TR-2014-01.pdf, 2014.

[4] Upgrade to the world's fastest GPU Accelerator - NVIDIA Tesla K40. http://www.nvidia.com/content/tesla/pdf/nvidia-tesla-k40-2014mar-lr.pdf, 2014.

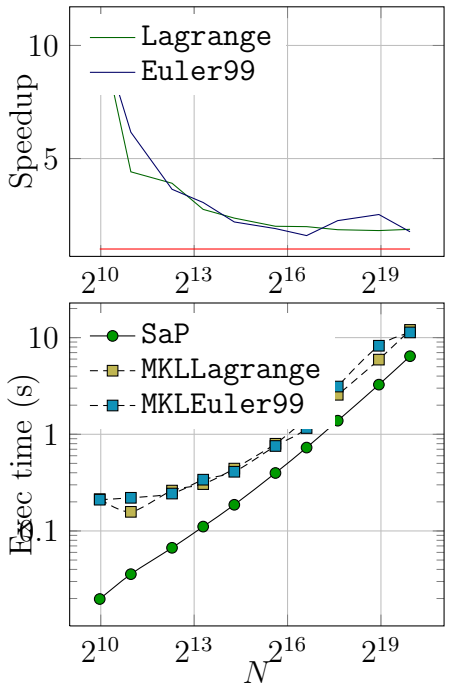
[5] E. Polizzi and A. Sameh. A parallel hybrid banded system solver: the SPIKE algorithm. *Parallel Computing*, 32(2):177–194, 2006.

[6] E. Polizzi and A. Sameh. SPIKE: A parallel environment for solving banded linear systems. *Computers & Fluids*, 36(1):113 – 120, 2007.

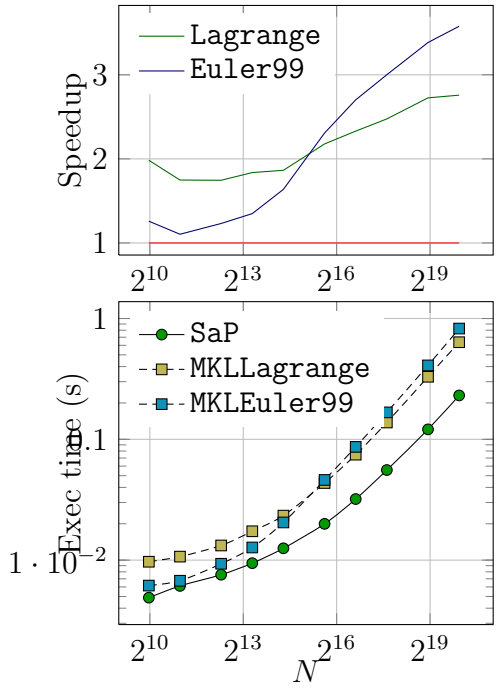
[7] A. Sameh and D. Kuck. On stable parallel linear system solvers. *JACM*, 25(1):81–91, 1978.



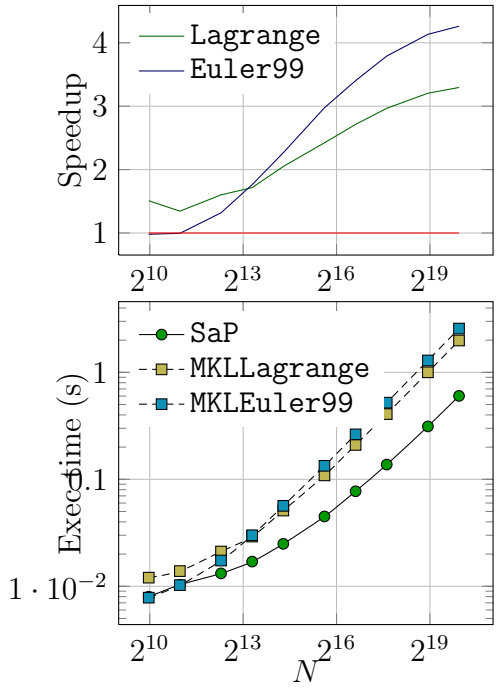

(a) Influence of the problem dimension $N$ for fixed values $K = 100$ and $d = 1$.




(b) Influence of the problem dimension $N$ for fixed values $K = 200$ and $d = 1$.

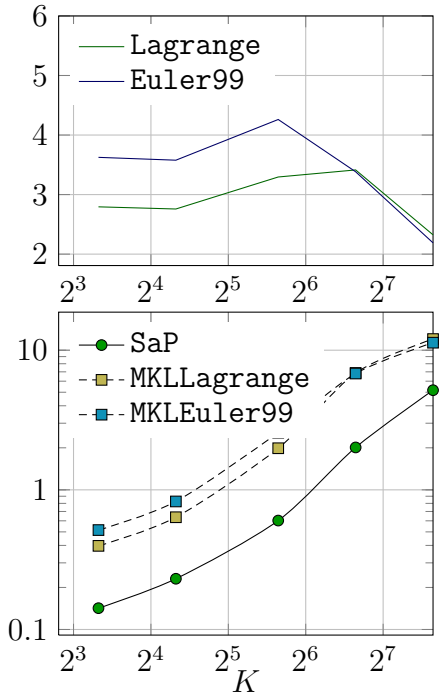



(c) Influence of the problem dimension $N$ for fixed values $K = 20$ and $d = 1$.

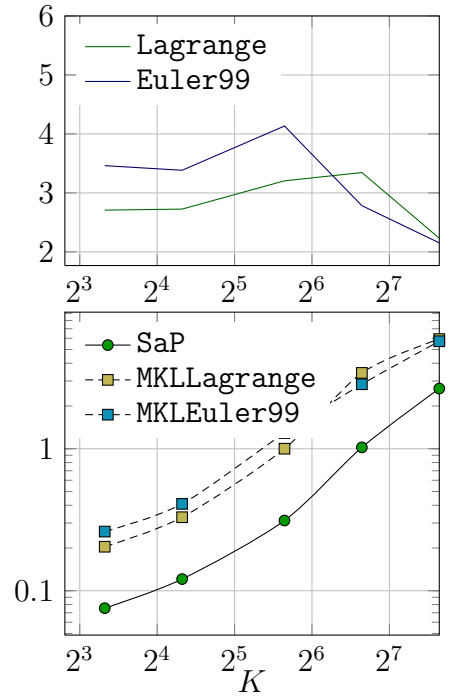



(d) Influence of the problem dimension $N$ for fixed values $K = 50$ and $d = 1$.

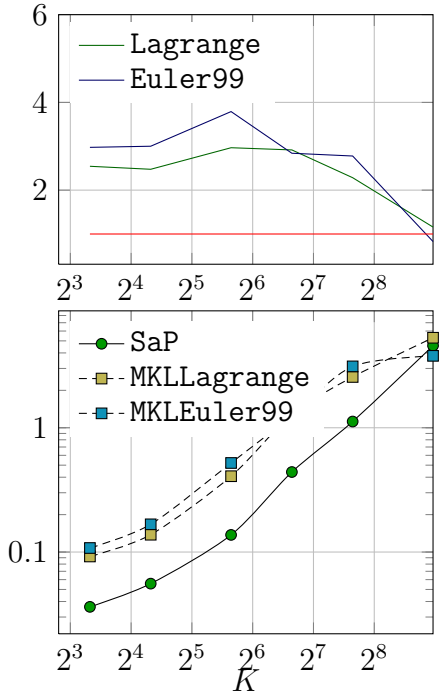
Figure 1: Parametric studies on synthetic banded matrices. Timing results for `SaP` and `MKL` are provided when $N$ (problem dimension) is varied, while $K$ (half-bandwidth) and $d$ (degree of diagonal dominance) are kept constant.



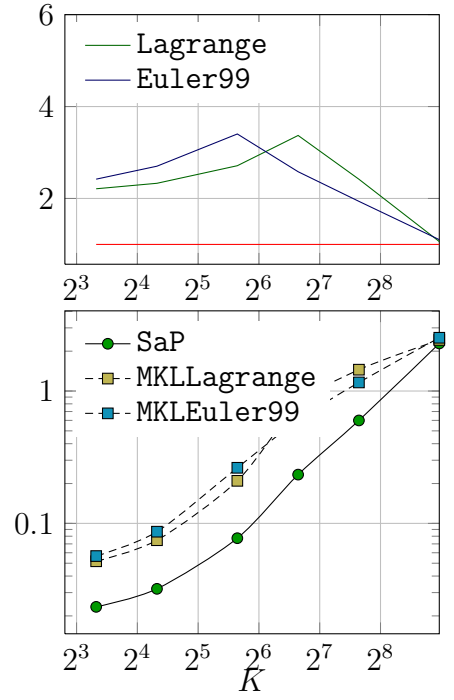

(a) Influence of the half-bandwidth $K$ for fixed values $N = 1,000,000$ and $d = 1$.




(b) Influence of the half-bandwidth $K$ for fixed values $N = 500,000$ and $d = 1$.

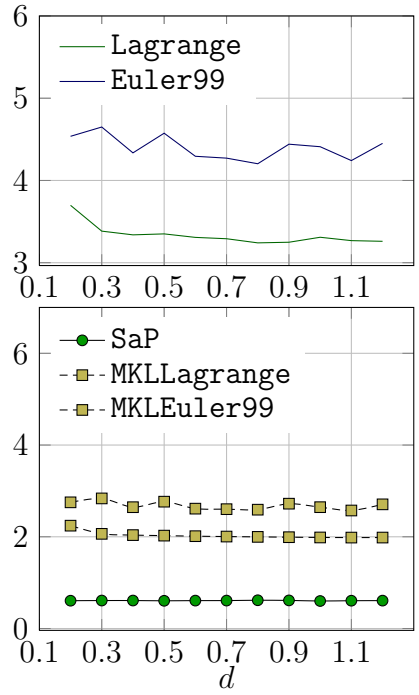



(c) Influence of the half-bandwidth $K$ for fixed values $N = 200,000$ and $d = 1$.

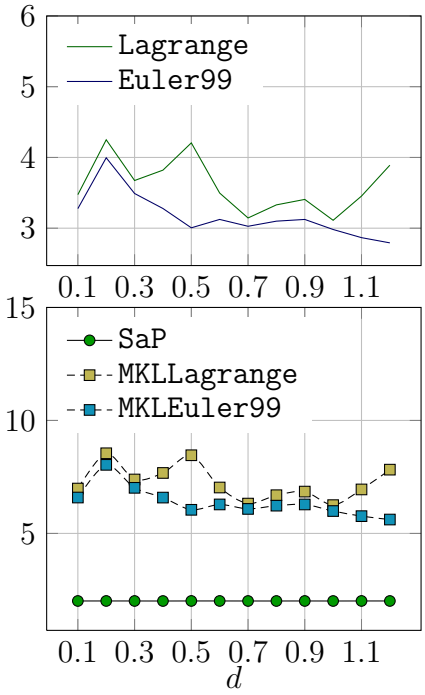



(d) Influence of the half-bandwidth $K$ for fixed values $N = 100,000$ and $d = 1$.

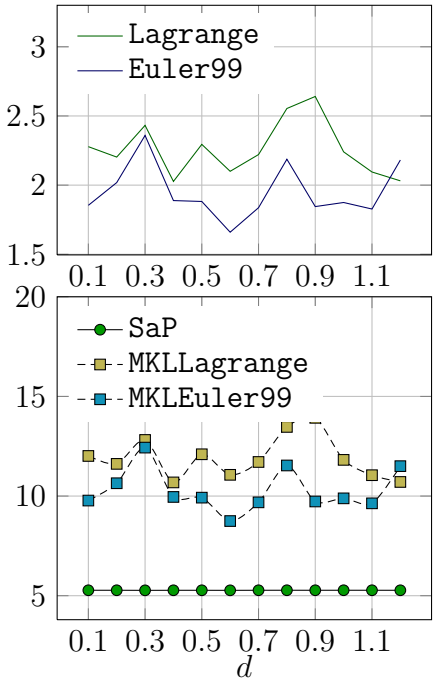
Figure 2: Parametric studies on synthetic banded matrices. Timing results for `SaP` and `MKL` are provided when $K$ (half-bandwidth) is varied, while $N$ (problem dimension) and $d$ (degree of diagonal dominance) are kept constant.




(a) Influence of the degree of diagonal dominance $d$ for fixed values $N = 1,000,000$ and $K = 50$.




(b) Influence of the degree of diagonal dominance $d$ for fixed values $N = 1,000,000$ and $K = 100$.




(c) Influence of the degree of diagonal dominance $d$ for fixed values $N = 1,000,000$ and $K = 200$.

Figure 3: Parametric studies on synthetic banded matrices. Timing results for `SaP` and `MKL` are provided when $d$ (degree of diagonal dominance) is varied, while $N$ (problem dimension) and $K$ (degree of diagonal dominance) are kept constant.