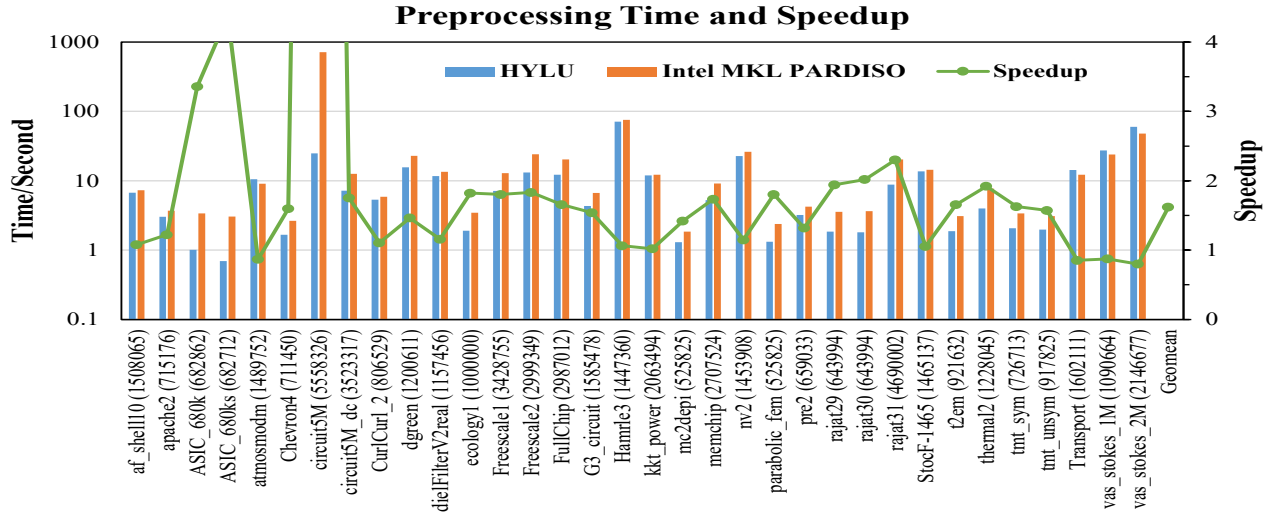
Test Results of HYLU and Performance Comparisons with Intel MKL PARDISO

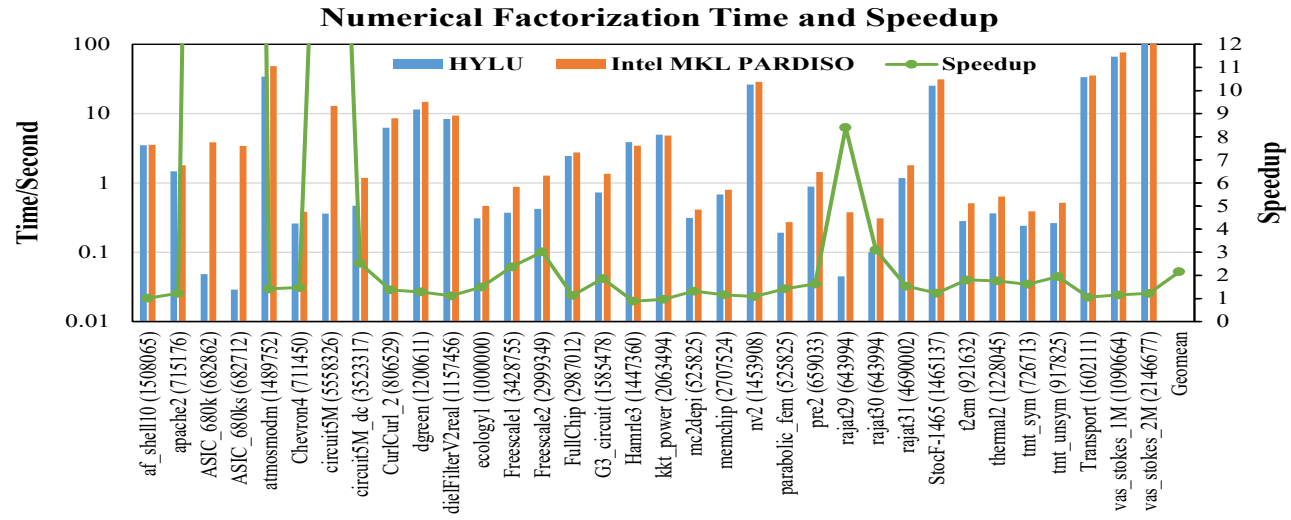
The experiments were carried out on a Linux server. The main hardware and software configurations are listed in the following table. All results presented in this document are wall-time measurements from **16-thread parallel execution**.

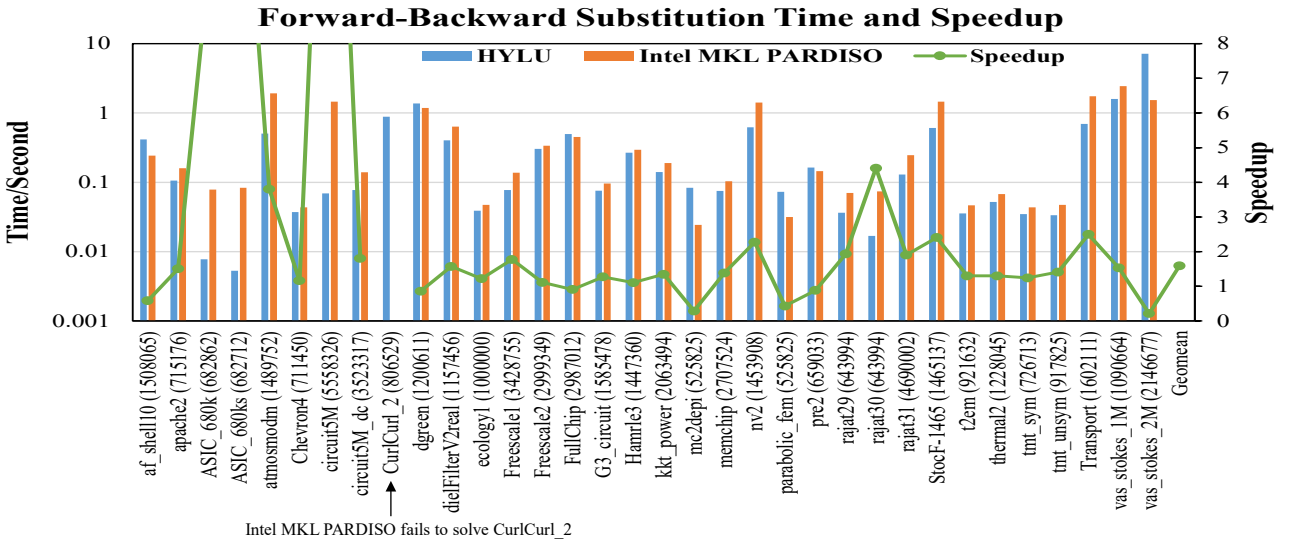
CPU	Intel Xeon Gold 6130 @ 2.1GHz
Memory	256GB
Operating system	Ubuntu 24.04 LTS for tests/Rocky Linux 8.10 for compiling HYLU library
Compiler	gcc 13.3.0 for compiling test code/gcc 8.5.0 for compiling HYLU library
MKL version	2025.2.0.629
HYLU version	20260116
Benchmarks	34 matrices from SuiteSparse Matrix Collection, dimensions from 525,825 to 5,558,326

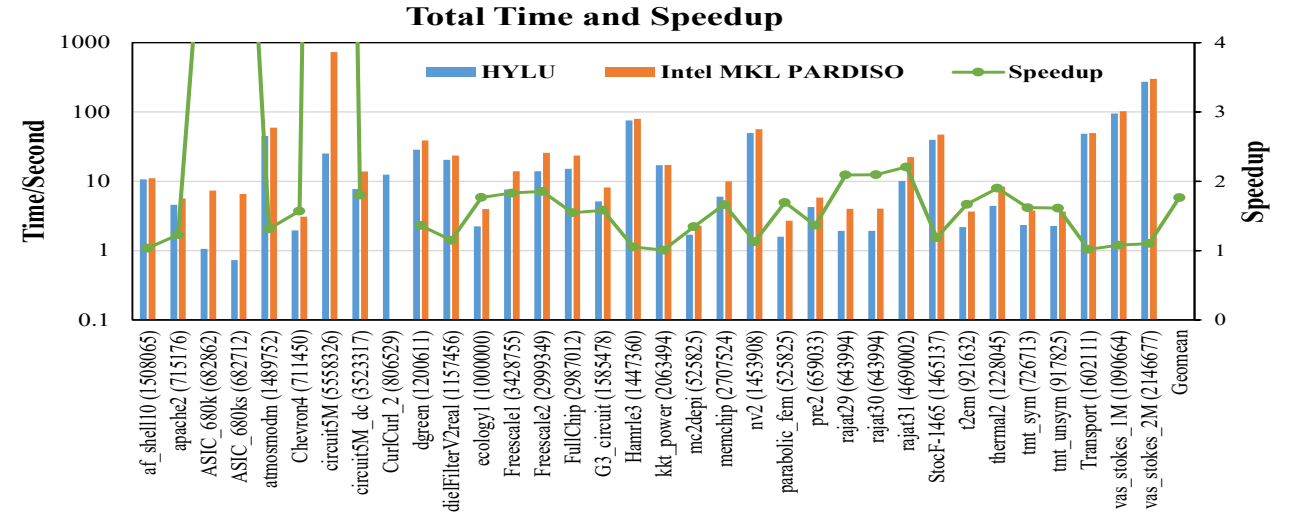
1. One-Time Solve

On geometric mean, HYLU achieves a **2.16X speedup in numerical factorization** compared with Intel MKL PARDISO, while the preprocessing and forward-backward substitution phases are also faster (**1.62X** and **1.59X** speedups, respectively). For the total one-time solve time (preprocessing + numerical factorization + forward-backward substitution), HYLU is faster than Intel MKL PARDISO for **ALL** of the tested benchmarks and the geometric mean of speedups is **1.77X**.









Another advantage of HYLU is its stable performance for various sparsities. Intel MKL PARDISO generates a huge quantity of fill-ins for some benchmarks (ASIC_680k, ASIC_680ks, and circuit5M), and the performance of these matrices is poor.

2. Repeated Solve

HYLU offers an optimization option for repeated solve of linear systems with an identical sparse pattern in the

coefficient matrix. In this case, HYLU achieves a **2.66X geometric mean speedup in numerical factorization** over Intel MKL PARDISO, while the forward-backward substitution phase is slightly faster (**1.37X** speedup). When comparing the total time of numerical factorization and forward-backward substitution, HYLU is faster than Intel MKL PARDISO for **ALL** of the tested benchmarks and the geometric mean of speedups is **2.53X**. In the repeated solve scenario, the preprocessing phase is on geometric mean 1.75X slower than that of the one-time solve scenario. However, in the repeated solve scenario, the pre-processing time is less important, as it is executed only once.