Synthesizing Numerical Linear Algebra using Julia

Sophie Xuan, Evelyne Ringoot, Rabab Alomairy, Felipe Tome, Julian Samaroo, and Alan Edelman

Objective

Improve efficiency of linear algebra routines:

- LAPACK / BLAS: most widely recognized linear algebra library
  - However, provides different routine for each data type and precision
- Each hardware provides own package

Provide agonistic implementations of LAPACK / BLAS routines:

- Recent efforts in C++ and Python
  - Ex: C++26 has all BLAS operations in language standard [3]

Use Julia Language for composability:

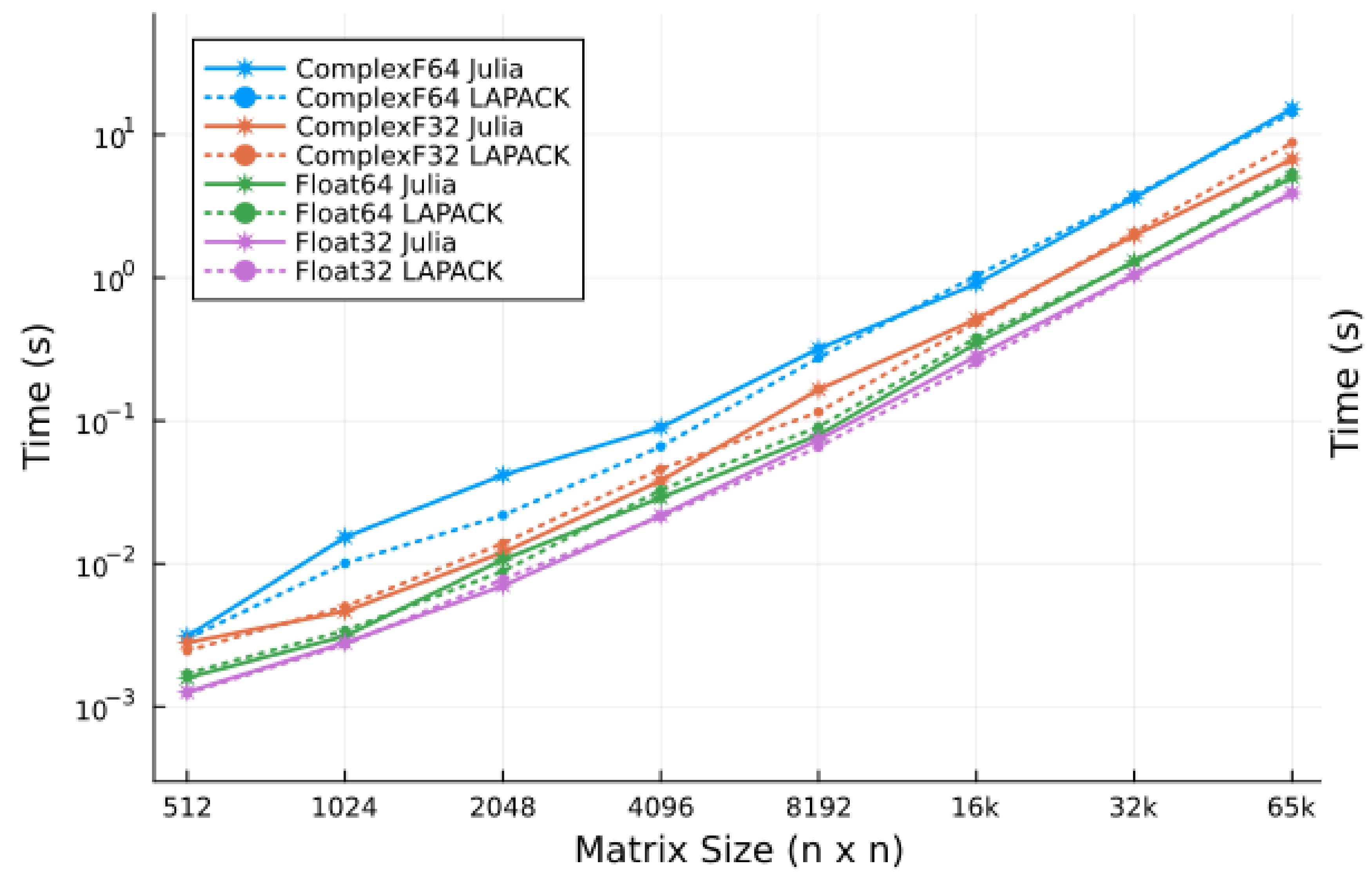
- Multiple-dispatch and type inference
  - LLVM dynamically generates optimized code for different data / hardware types [2]
- Provide single API
  - Ease of usage benefits
  - Shorter development time
- Extendibility
  - Update to new hardware
  - Extend to new data types

References

[1] Bezanson, Julia: Dynamism and Performance. *Proc. ACM Program. Lang.*, oct 2018.
[2] Bezanson, Array Operators Using Multiple Dispatch: In *Proceedings of ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming*
[3] cppreference. C++26: Basic linear algebra algorithm, 2024
[4] Danisch, GPUArrays

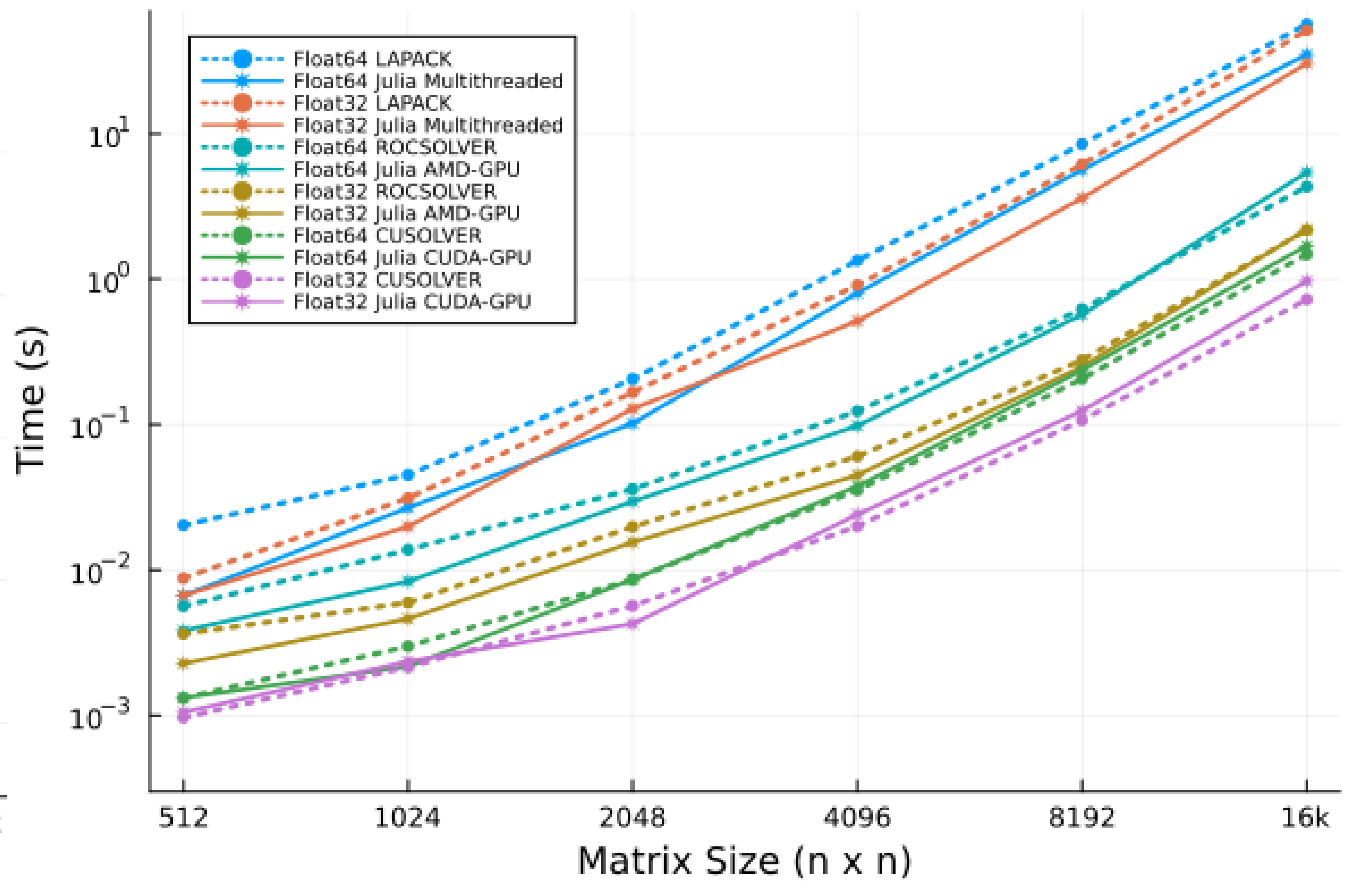
A generic API for linear algebra in Julia provides composability without sacrificing performance

Generic **larfb** function matches performance of data-type specific LAPACK implementation



Performance across data types
(Intel Ice Lake Processor)

Single-API **unmqr** closely matches performance of both LAPACK (CPU) and CUSOLVER/ROCSOLVER (GPU)



Performance across Hardware
(Intel CascadeLake CPU and V100/MI100 GPU)

What's next?

1. **Expand library to include more routines** -- LU Factorization , QR factorization, etc.
2. **Add support to mixed precisions computation**.
3. **Further optimizations for parallel computing** -- KernelAbstractions, Dagger, etc.

Methods

Implemented larfb and unmqr functions in Julia

- **unmqr**: applies orthogonal matrix Q of a QR factorization to generic rectangular matrix A:
  - $A + AYTY^T = AQ$
    &
    $A + YTY^TA = QA$
  - Y is block-householder factorization forming Q
  - T is scalar factors of elementary reflectors
- **larfb**: performs the individual projections; used in unmqr
- Publicly available in DLA.jl

Abstract Array interface allows for Unified API [4]

- Supports arbitrary data types
  - Precision (8/16/32/64 bit)
  - Data type (Integer/ Float/ Complex)
- Dispatch to CPU / GPU with minimal changes to API
  - Separate implementations historically

Vendor and Family	Intel Ice Lake	Intel CascadeLake	AMD Milan
Model	6330	6248	7713
Sockets(s)	2	2	2
Cores per Socket	28	20	64
Clock Speed	2 GHz	2.5 GHz	2.0GHz
DDR Memory Size	1 TB	384 GB	256GB
L3 Cache Size	84 MiB	27.5 MiB	512 MiB
GPU type	NVIDIA A100	NVIDIA V100	AMD MI100

Hardware Specifications