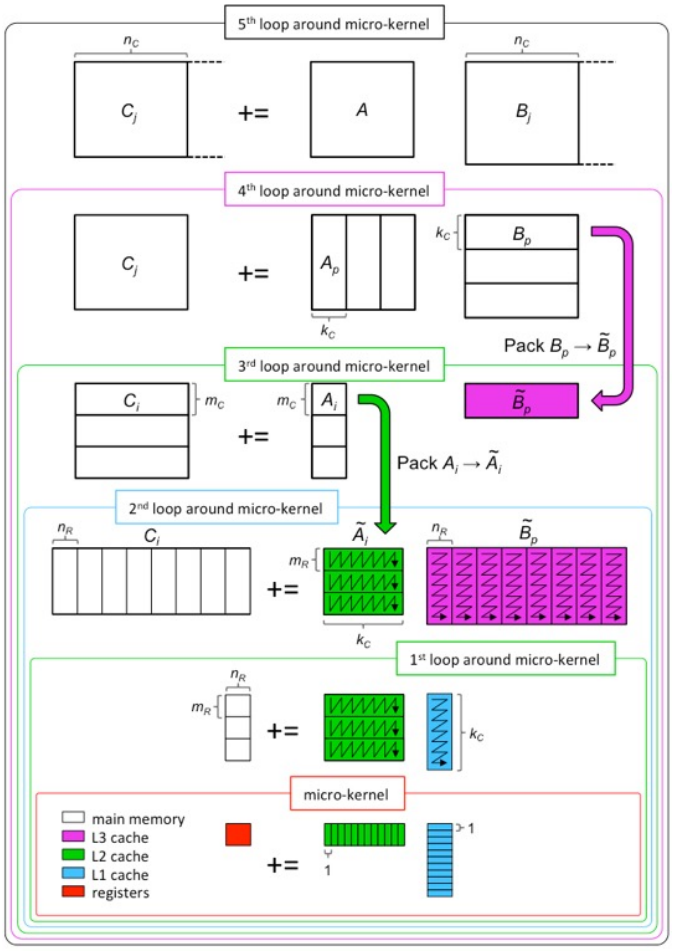
BLIS: 1.0, 2.0, and beyond

Devin Matthews

Southern Methodist University

What is BLIS?


5th loop around micro-kernel
nC
Cj
+=
A
Bj
4th loop around micro-kernel
Ap
kC
Bp
Pack Bp → B̃p
3rd loop around micro-kernel
Ci
mC
Ai
Pack Ai → Ãi
2nd loop around micro-kernel
nR
mR
1st loop around micro-kernel
micro-kernel
main memory
L3 cache
L2 cache
L1 cache
registers

Architectures

- Penryn (SSE3)
- Sandy/Ivy Bridge (AVX)
- Haswell/Broadwell (AVX2)
- KNC
- KNL
- Skylake and later (AVX512)
- Bulldozer
- Piledriver
- Excavator
- Zen 1,2,3
- BlueGene/Q
- POWER 7,9,10
- ARMv7a
- ARMv8a
- A64fx
- Altra/AltraMax
- Apple M1
- SiFive x280 (and other Risc-V)
- …

Features

- **Fully open-source BLAS/CBLAS implementation**
- Full BLAS Level 1,2,3 support
- GEMMT and other extensions
- Improved BLIS interface: general stride, object-based, …
- Hierarchical and tunable parallelism
- Rapid prototyping for new architectures
- Permissive licensing (BSD3)
- Clean, well-documented C99 code (no Fortran!)
- Contributions from >50 developers (academia, industry, individual)
- Small/skinny matrix optimizations

Ideas

- Extend Goto's algorithm down to a microkernel: the BLIS algorithm and portable performance
- Application of formal linear algebra methods to CPU cache hierarchy
- Analytical derivation of block sizes and threading parameters
- Hierarchical multithreading based on cache/memory sharing
- Analytical performance modeling
- Induced complex arithmetic: the 1m/3m/4m methods

BLIS 1.0

- Released May 6, 2024
- New version scheme: 0.9.0 -> 1.0 -> (2.0)…

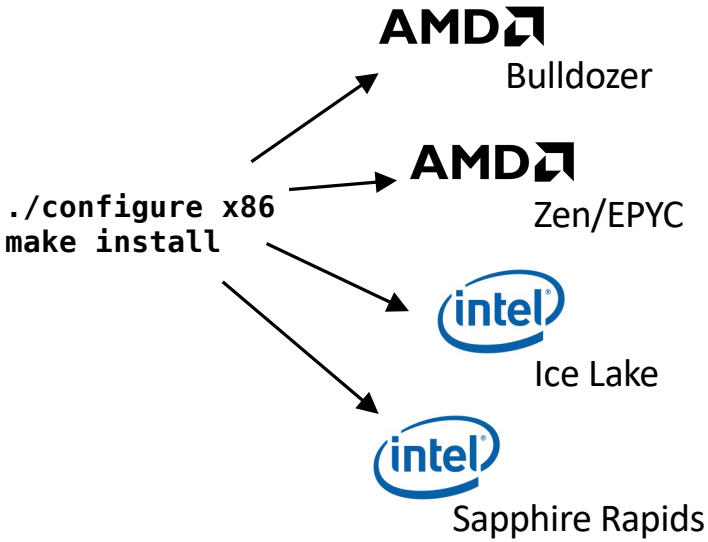
BLIS 1.0

- Released May 6, 2024
- New version scheme: 0.9.0 -> 1.0 -> (2.0)…
- Banner features:
  - Fat multithreading

Fat Multithreading

Fat binaries:

Configure once `./configure x86`, run anywhere



Slim multithreading:

Configuration depends on threading backend

```
./configure
make install
```
→ Single-threaded execution

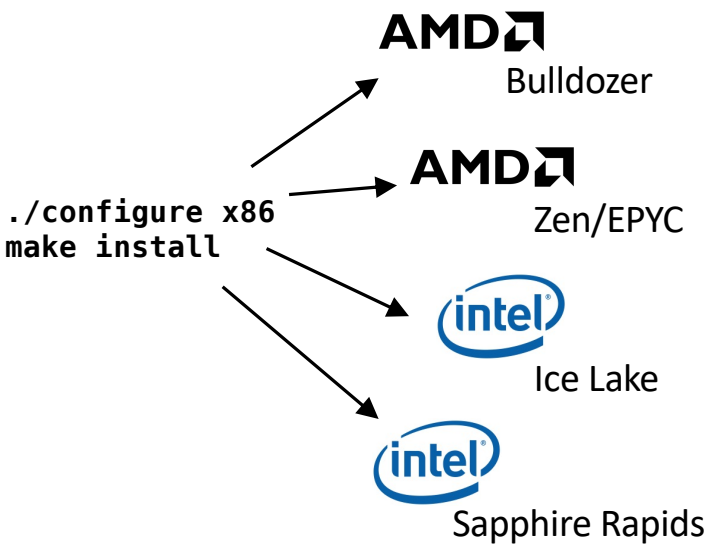
```
./configure –tomp
make install
```
→ OpenMP parallelization

```
./configure –tpthreads
make install
```
→ Pthreads parallelization

Fat Multithreading

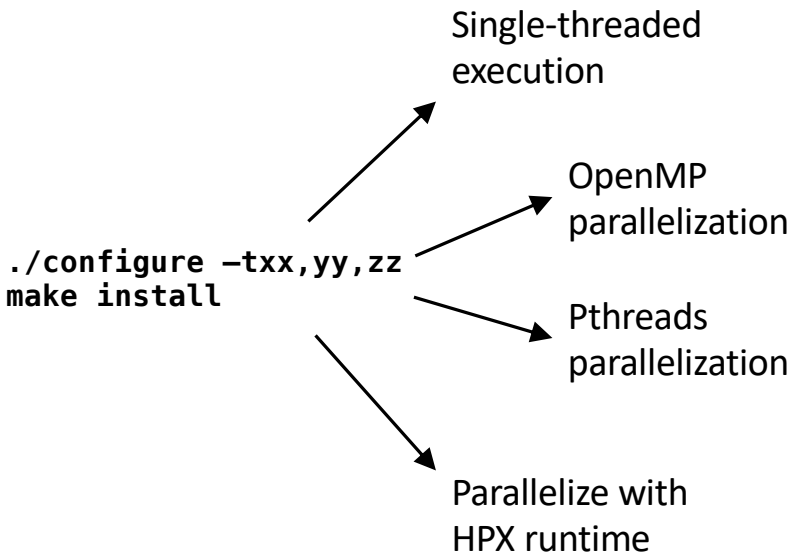
Fat binaries:

Configure once `./configure x86`, run anywhere



Fat multithreading:

Configure once, choose back-end at runtime

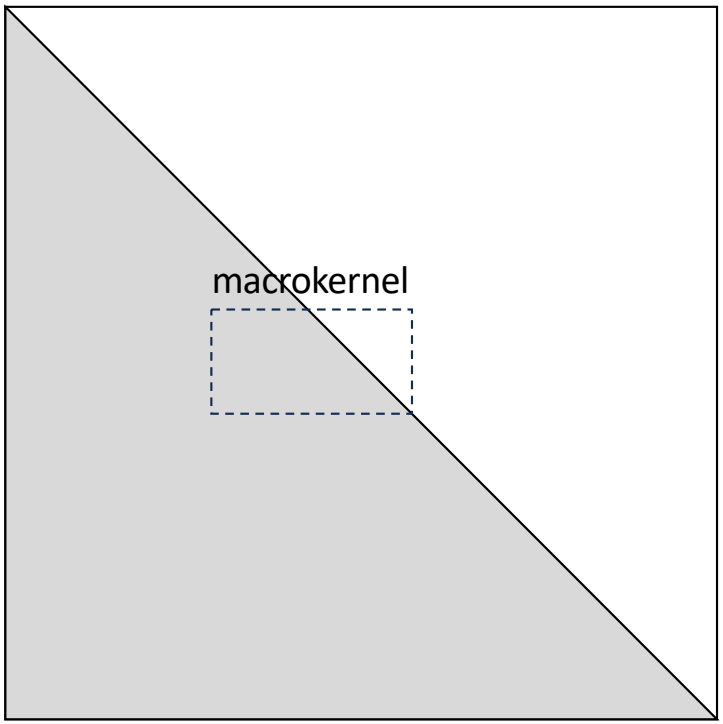


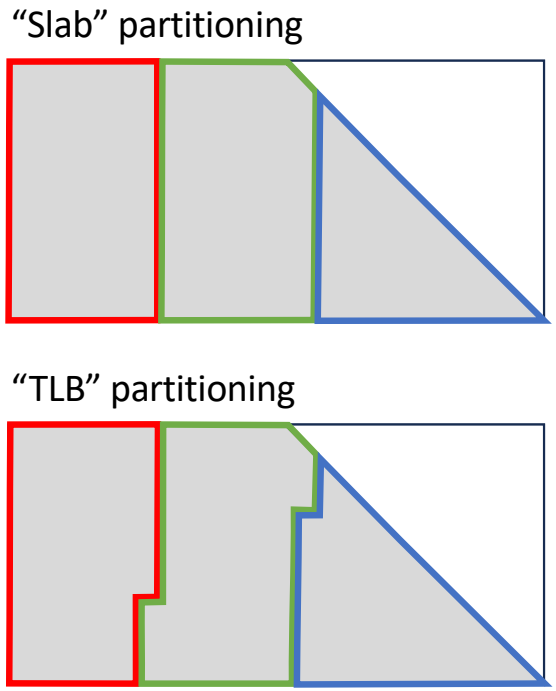
BLIS 1.0

- Released May 6, 2024
- New version scheme: 0.9.0 -> 1.0 -> (2.0)…
- Banner features:
  - Fat multithreading
  - Tile-level Load Balancing (TLB) and other threading improvements

Tile-Level Load Balancing

GEMMT: matrix multiplication, storing only upper/lower triangle





BLIS 1.0

- Released May 6, 2024
- New version scheme: 0.9.0 -> 1.0 -> (2.0)…
- Banner features:
  - Fat multithreading
  - Tile-level Load Balancing (TLB) and other threading improvements
  - Added support for Ampere Altra/AltraMax and RISC-V (generic and x280)
  - Fewer macro templates, generally easier debugging
  - Better compatibility with C++ compilers/code
  - Improved numerical consistency: **`invscalv`** and NaN/Inf handling in **`sumsqv`**
  - Many, many small improvements!

BLIS 2.0

- Release candidate out soon, full release later this year!
- Major functionality and under-the-hood changes:

BLIS 2.0

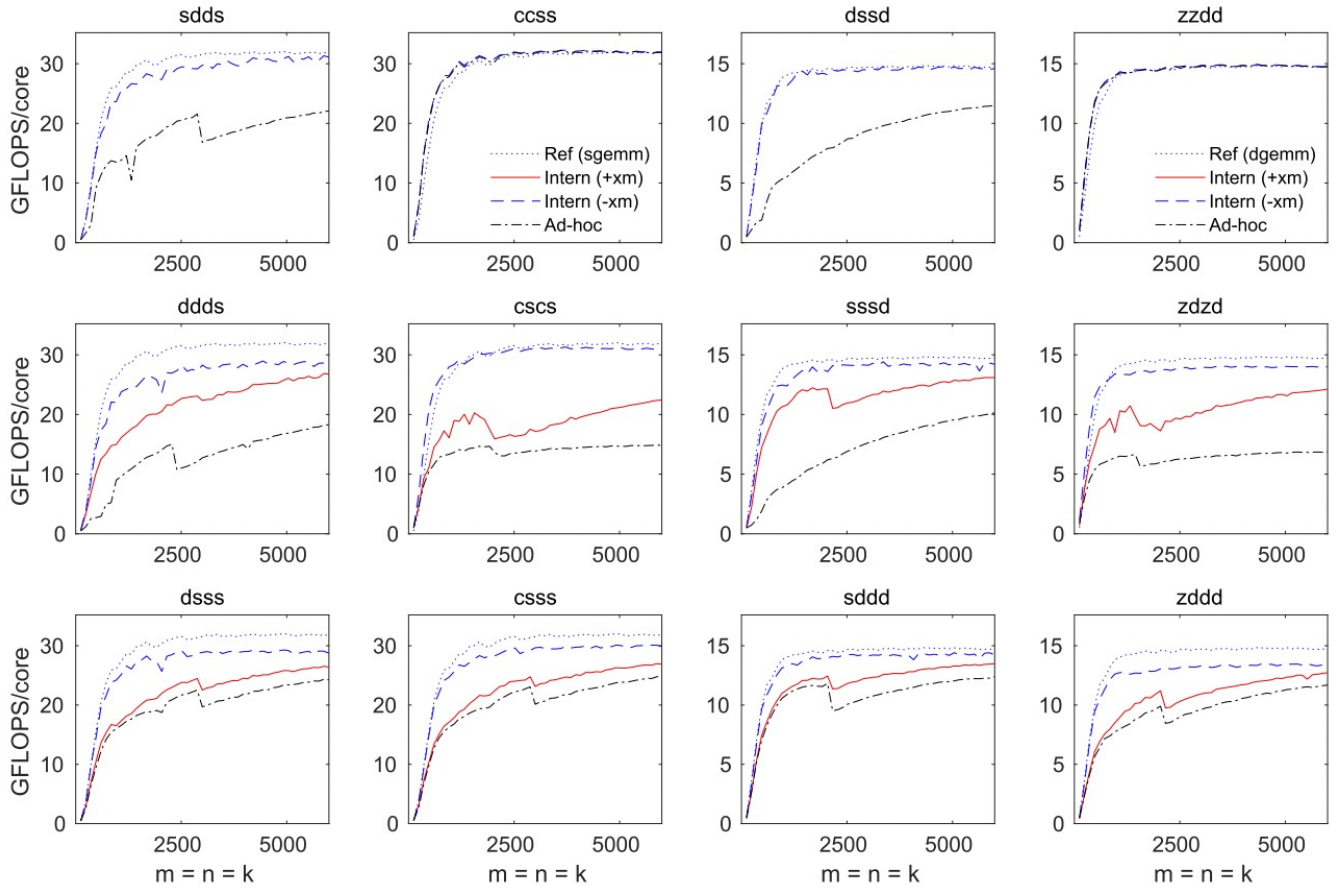
- Release candidate out soon, full release later this year!
- Major functionality and under-the-hood changes:
    - Full mixed-precision and mixed-domain level-3 BLAS (except TRSM)

Mixed-precision mixed domain BLAS

For GEMM:
FGVZ, DNP, RVDG,
ACM TOMS **47**, 12 (2021)

Now enabled for all
level-3 BLAS
(no TRSM)

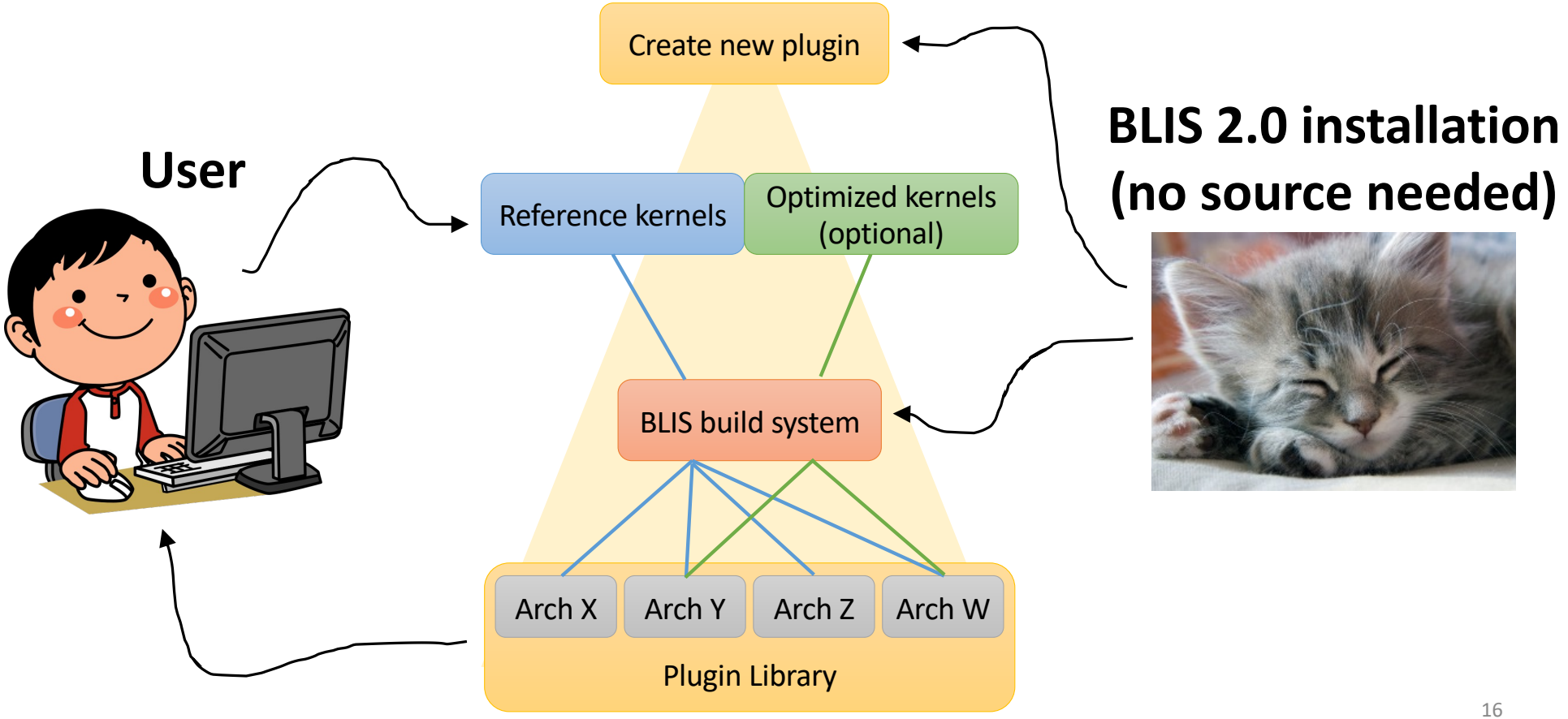
Can mixed S, D, C, Z
freely, no wasted FLOPS



BLIS 2.0

- Release candidate out soon, full release later this year!
- Major functionality and under-the-hood changes:
    - Full mixed-precision and mixed-domain level-3 BLAS (except TRSM)
    - A new “plugin” architecture

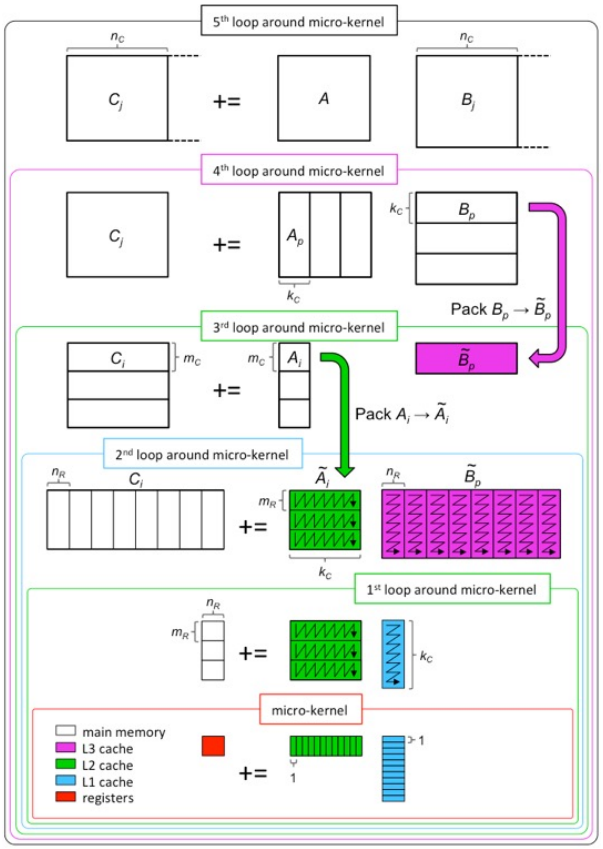
Plugins


Create new plugin
User
Reference kernels
Optimized kernels (optional)
BLIS 2.0 installation (no source needed)
BLIS build system
Arch X
Arch Y
Arch Z
Arch W
Plugin Library

BLIS 2.0

- Release candidate out soon, full release later this year!
- Major functionality and under-the-hood changes:
  - Full mixed-precision and mixed-domain level-3 BLAS (except TRSM)
  - A new “plugin” architecture
  - API for implementing user-define BLAS-like operations

User-defined operations



I want to do ~~SYRK~~ SYRK**D**!

$$C = \alpha A D A^T$$

$$(= \alpha A B^T, B = AD)$$

```
gemm_cntl_t cntl;
bli_gemm_cntl_init( …, BLIS_GEMMT, …, &cntl );

// address of D, incd, etc.
struct sykrd_params params = …;

bli_gemm_cntl_set_packb_ukr_simple(
       bli_cntl_get_ukrs( MY_PACKD_UKR, cntx ), &cntl );
bli_gemm_cntl_set_packb_params( &params, &cntl );

bli_l3_thread_decorator( …, &cntl, … );
```

The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!

The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!
- Some planned improvements:
  - Extended precisions, e.g. fp16, bf16, intX, double-double

The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!
- Some planned improvements:
  - Extended precisions, e.g. fp16, bf16, intX, double-double
  - Thread pool support for pthreads; internal support for thread locality

The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!
- Some planned improvements:
  - Extended precisions, e.g. fp16, bf16, intX, double-double
  - Thread pool support for pthreads; internal support for thread locality
  - Expanded BLAS-like functionality: SkewBLAS, “sandwich products”, face-splitting-product-times-matrix (gemm3diag), etc.

The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!
- Some planned improvements:
  - Extended precisions, e.g. fp16, bf16, intX, double-double
  - Thread pool support for pthreads; internal support for thread locality
  - Expanded BLAS-like functionality: SkewBLAS, “sandwich products”, face-splitting-product-times-matrix (gemm3diag), etc.
  - PC-loop parallelization (e.g. for k >> m,n)

The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!
- Some planned improvements:
  - Extended precisions, e.g. fp16, bf16, intX, double-double
  - Thread pool support for pthreads; internal support for thread locality
  - Expanded BLAS-like functionality: SkewBLAS, “sandwich products”, face-splitting-product-times-matrix (gemm3diag), etc.
  - PC-loop parallelization (e.g. for k >> m,n)
  - Non-destructive operations, e.g. D = A*B + C

The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!
- Some planned improvements:
  - Extended precisions, e.g. fp16, bf16, intX, double-double
  - Thread pool support for pthreads; internal support for thread locality
  - Expanded BLAS-like functionality: SkewBLAS, “sandwich products”, face-splitting-product-times-matrix (gemm3diag), etc.
  - PC-loop parallelization (e.g. for k >> m,n)
  - Non-destructive operations, e.g. D = A*B + C
  - Pre-packing of matrices

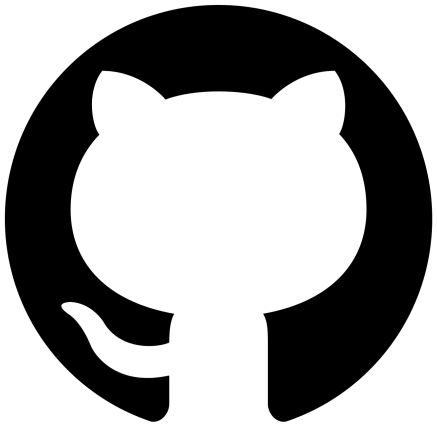
The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!
- Some planned improvements:
  - Extended precisions, e.g. fp16, bf16, intX, double-double
  - Thread pool support for pthreads; internal support for thread locality
  - Expanded BLAS-like functionality: SkewBLAS, “sandwich products”, face-splitting-product-times-matrix (gemm3diag), etc.
  - PC-loop parallelization (e.g. for k >> m,n)
  - Non-destructive operations, e.g. D = A*B + C
  - Pre-packing of matrices
  - Task-based parallelization (OpenMP, TBB, etc.); external thread communicators

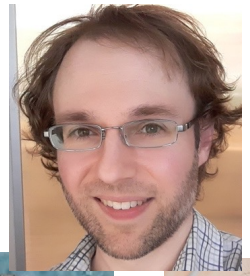
The Future

- BLIS is a “mature” linear algebra library, but it continues to improve!
- Some planned improvements:
  - Extended precisions, e.g. fp16, bf16, intX, double-double
  - Thread pool support for pthreads; internal support for thread locality
  - Expanded BLAS-like functionality: SkewBLAS, “sandwich products”, face-splitting-product-times-matrix (gemm3diag), etc.
  - PC-loop parallelization (e.g. for k >> m,n)
  - Non-destructive operations, e.g. D = A*B + C
  - Pre-packing of matrices
  - Task-based parallelization (OpenMP, TBB, etc.); external thread communicators
  - AMD GPU offloading

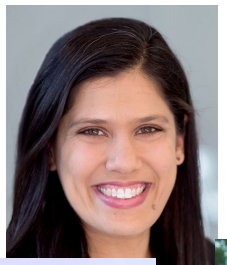
BLIS is a Community Project

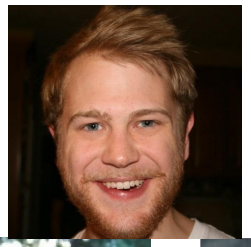


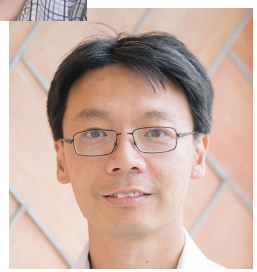
115+ contributors



Bryan Marker







AMD

ARM

ORACLE

NATIONAL INSTRUMENTS

hp

Microsoft

NEC

NASA

facebook

MathWorks

IBM

HUAWEI

intel

NSF

Qualcomm

NVIDIA

ti

SHPC

ODEN INSTITUTE FOR COMPUTATIONAL ENGINEERING & SCIENCES

WANTED

Kernels, configs, and general tuning:

Neoverse N1 (Graviton2), V1 (Graviton3), N2,
V2 (Graviton4, NVIDIA Grace)

Apple M2, M3, M4, Apple AMX

Intel Ice Lake(-SP) and later, Sapphire Rapids (Intel AMX)

REWARD: Fame, bragging rights, and Github cred!