
BLIS
THE FINAL FRONTIER

Some History…

Formal
Linear
Algebra
Methods
Environment

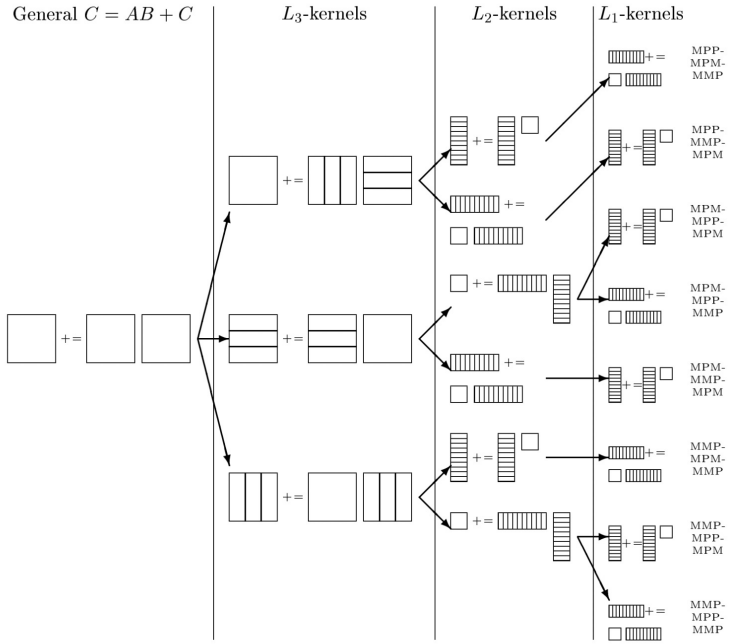


Figure 3: Possible algorithms for matrices in memory level L_3 given all L_2-kernels.

ITX-GEMM

John Gunnels (UT, now NVIDIA)

Greg Henry (Intel)

High-performance implementation of the level-3 BLAS

Authors: Kazushige Goto, Robert Van De Geijn Authors Info & Claims

ACM Transactions on Mathematical Software, Volume 35, Issue 1 • Article No.: 4, pp 1–14 • https://doi.org/10.1145/1377603.1377607

Published: 25 July 2008 Publication History

Check for updates

205 1,534

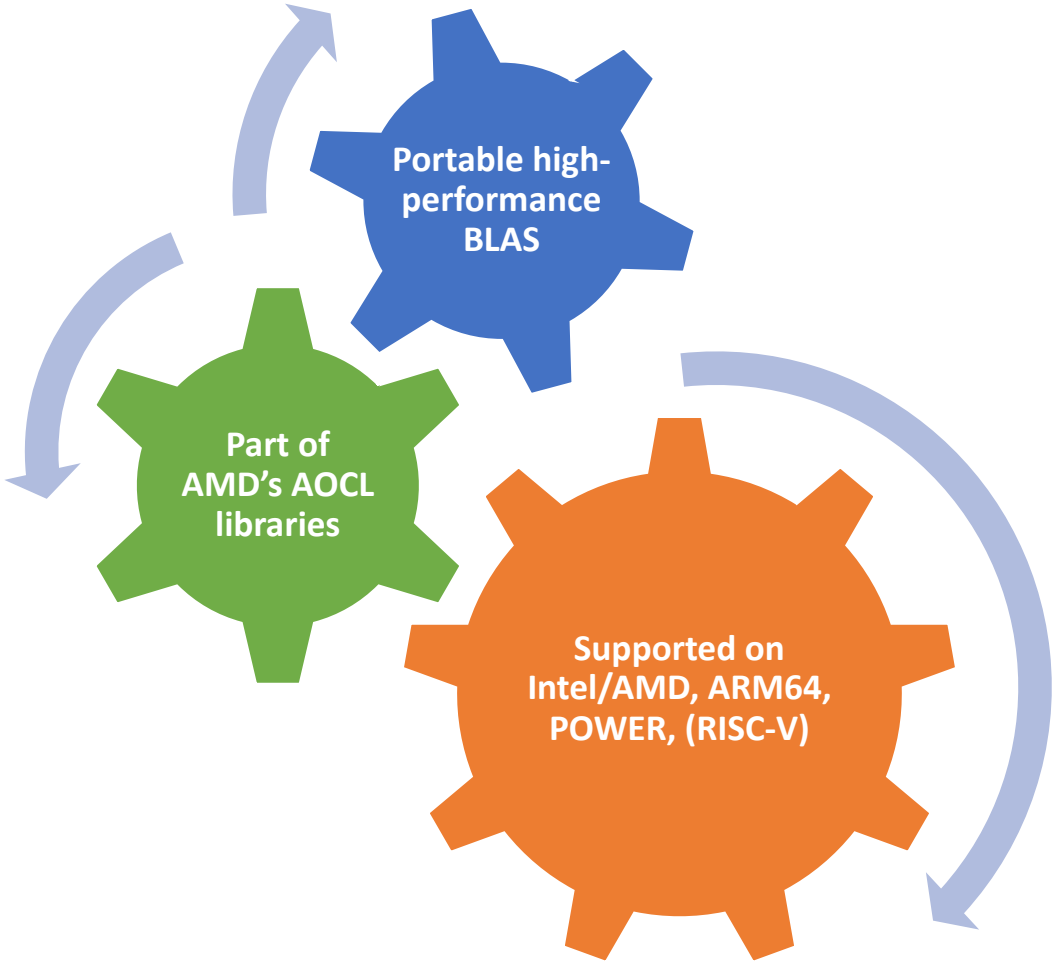
eReader PDF

Some History…

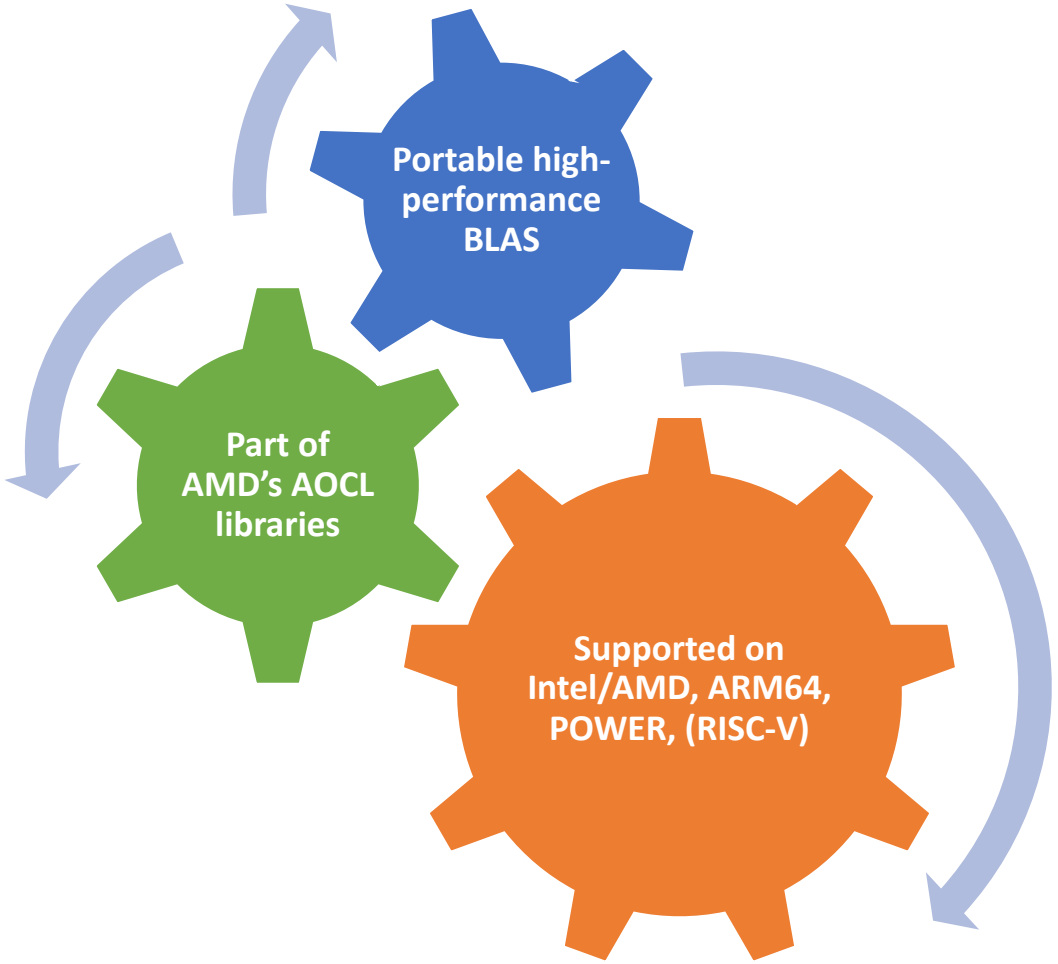
- **2012** • First NSF SI^2 award. Subsequent SI^2/CSSI awards in 2016 and 2020.
- **2012** • First BLIS commit in Github. BLIS existed before this within libflame.
- **2013** • First BLIS retreat.
- **2014** • Multi-threading support.
- **2016** • Runtime multi-architecture support.
- **2019** • “Skinny/unpacked” GEMM
- **2020** • SIAM Activity Group Best Paper Prize
- **2023** • James H. Wilkinson Prize for Numerical Software

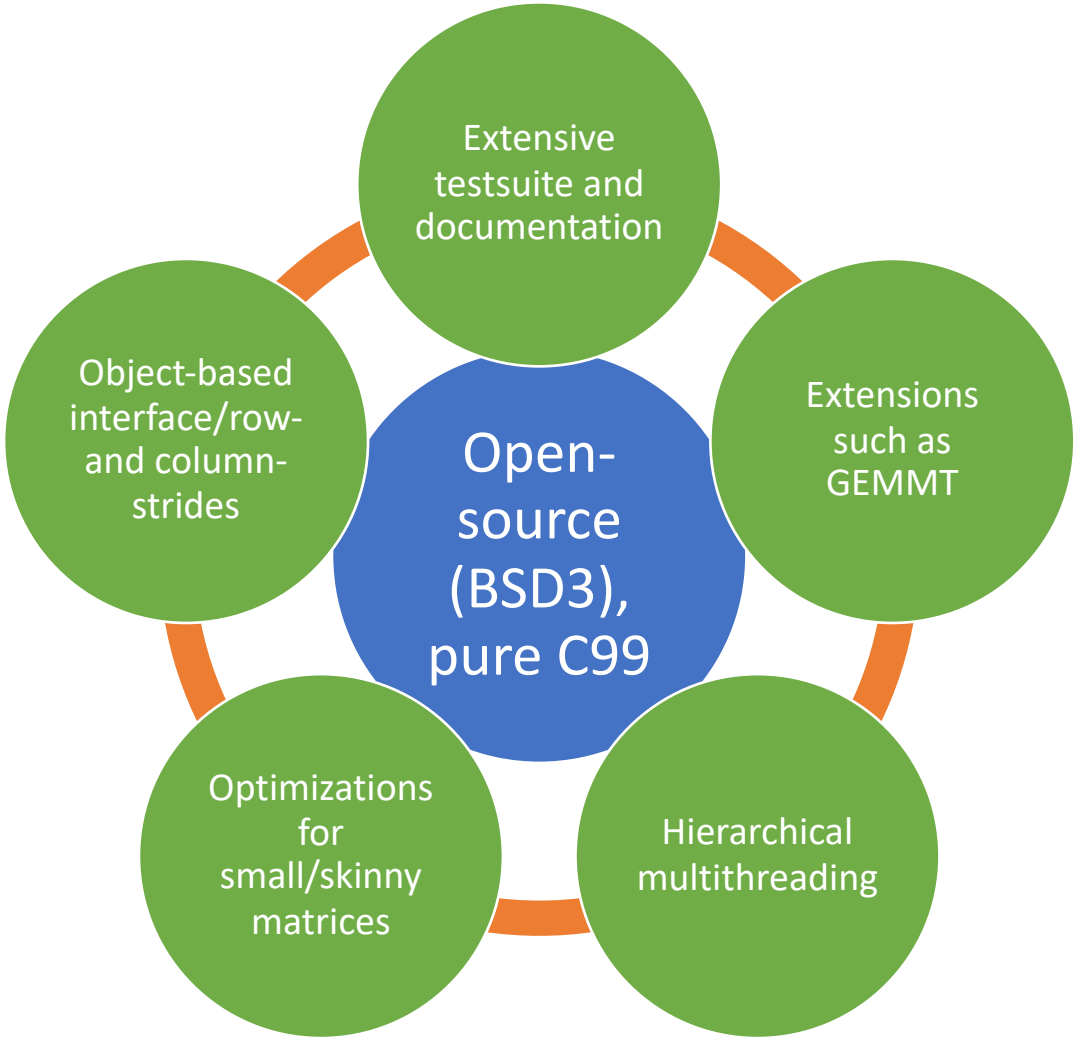
What *IS* BLIS?

What *IS* BLIS?

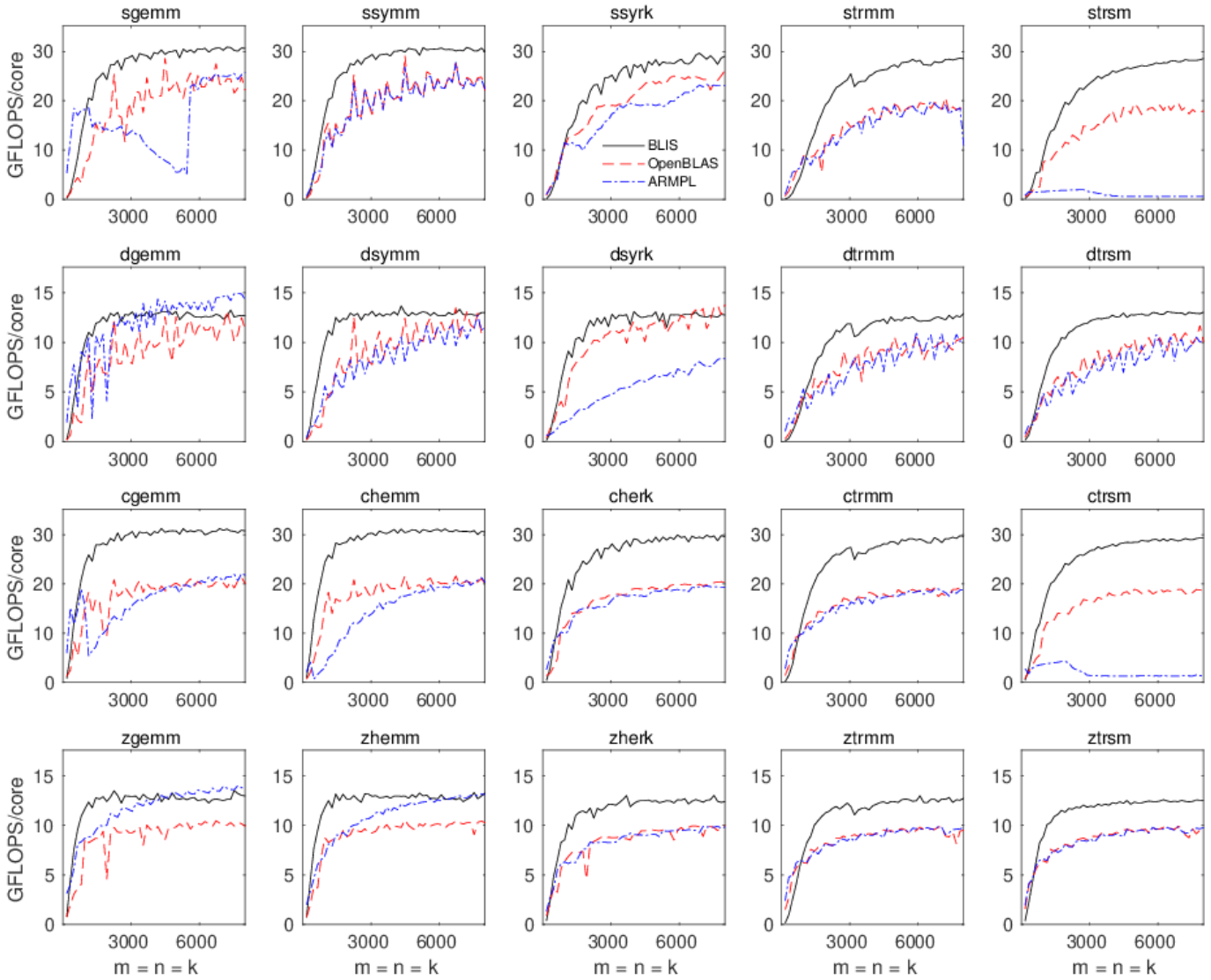

Portable high-performance BLAS
Part of AMD's AOCL libraries
Supported on Intel/AMD, ARM64, POWER, (RISC-V)

What *IS* BLIS?


Portable high-performance BLAS
Part of AMD's AOCL libraries
Supported on Intel/AMD, ARM64, POWER, (RISC-V)


Extensive testsuite and documentation
Object-based interface/row- and column-strides
Open-source (BSD3), pure C99
Extensions such as GEMMT
Optimizations for small/skinny matrices
Hierarchical multithreading

What *IS* BLIS?



ThunderX2

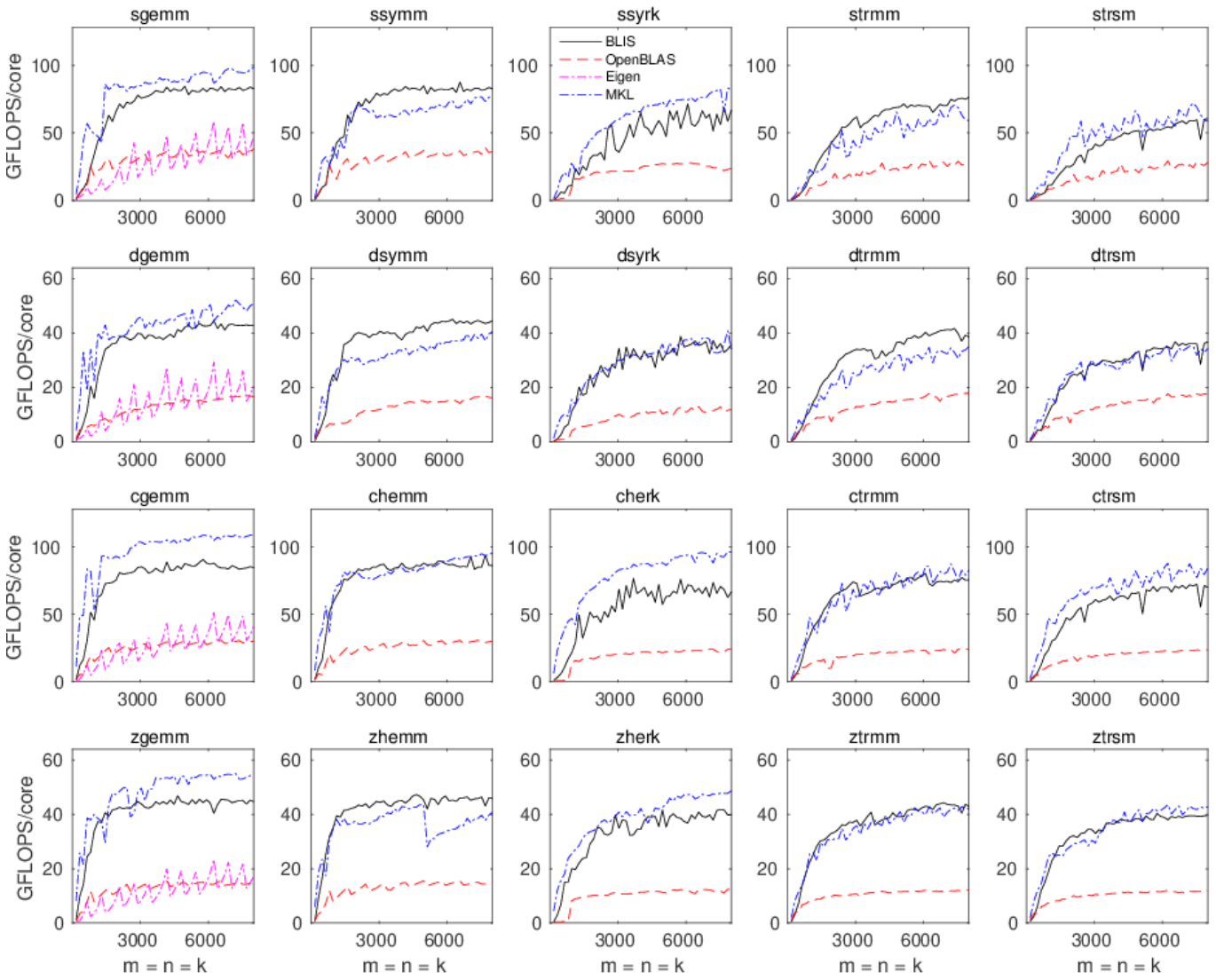
BLIS

The

Other

Guys

What *IS* BLIS?



Skylake X

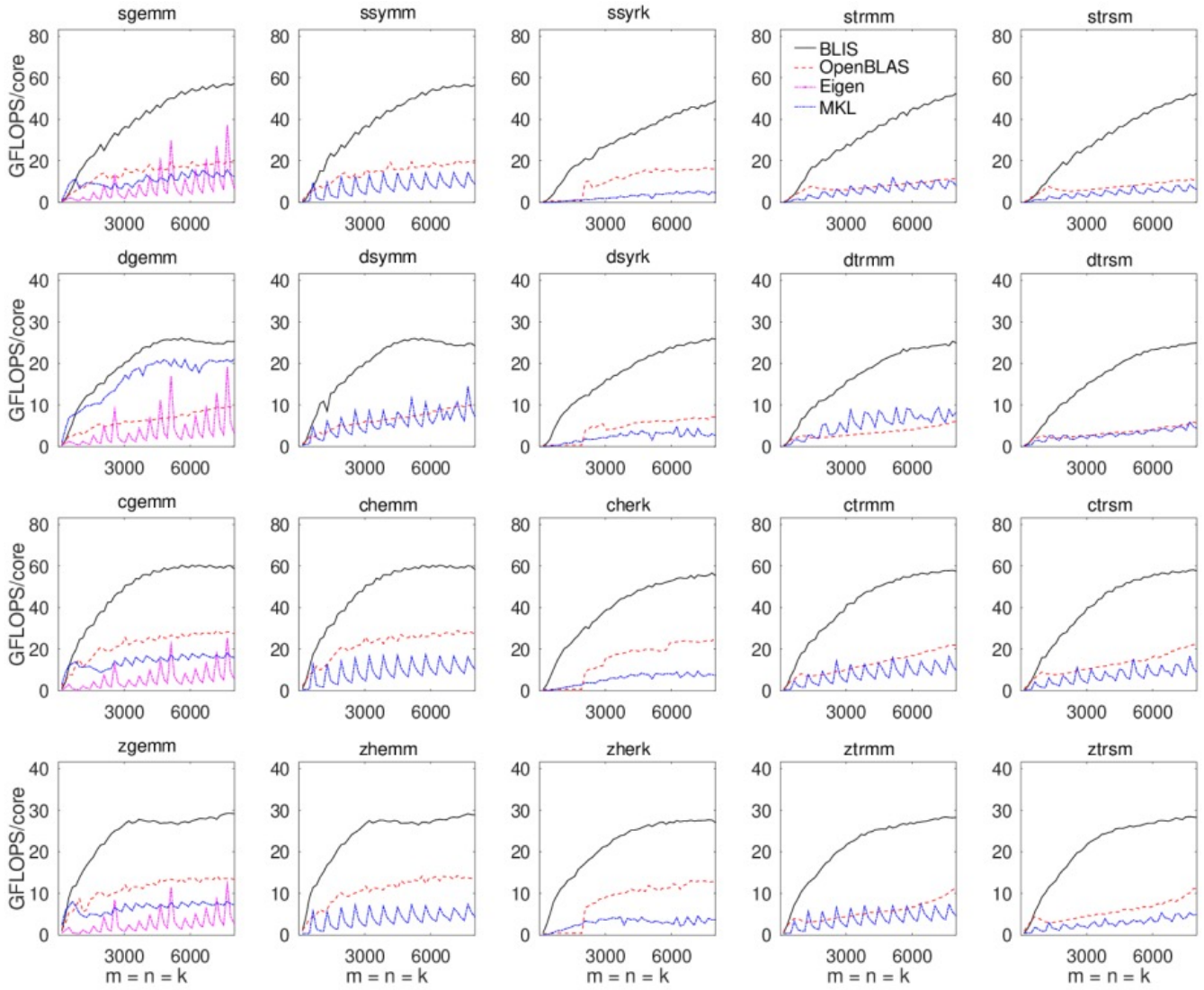
BLIS

The

Other

Guys

What *IS* BLIS?



Zen 2 (EPYC)

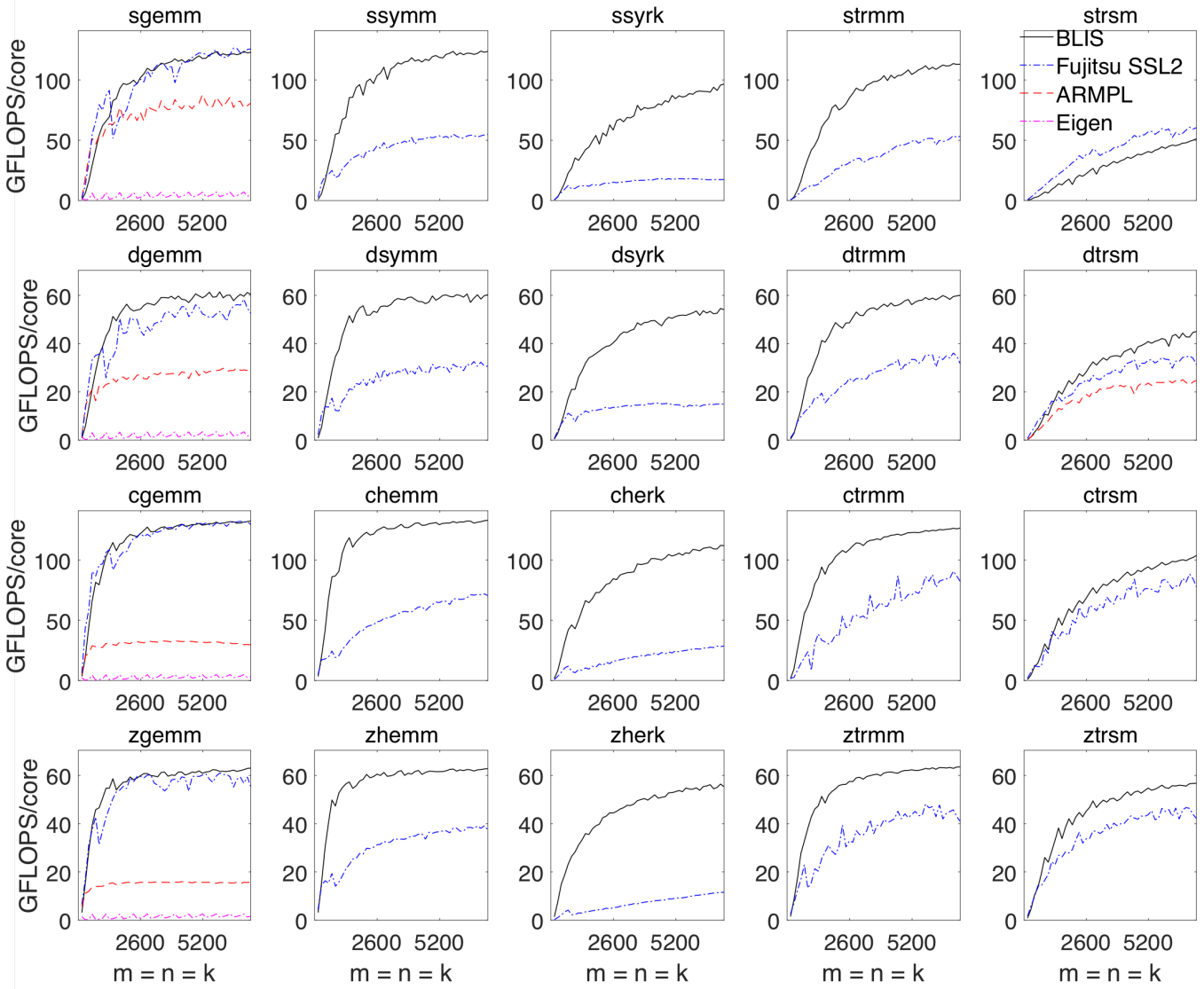
BLIS

The

Other

Guys

What *IS* BLIS?



A64fx

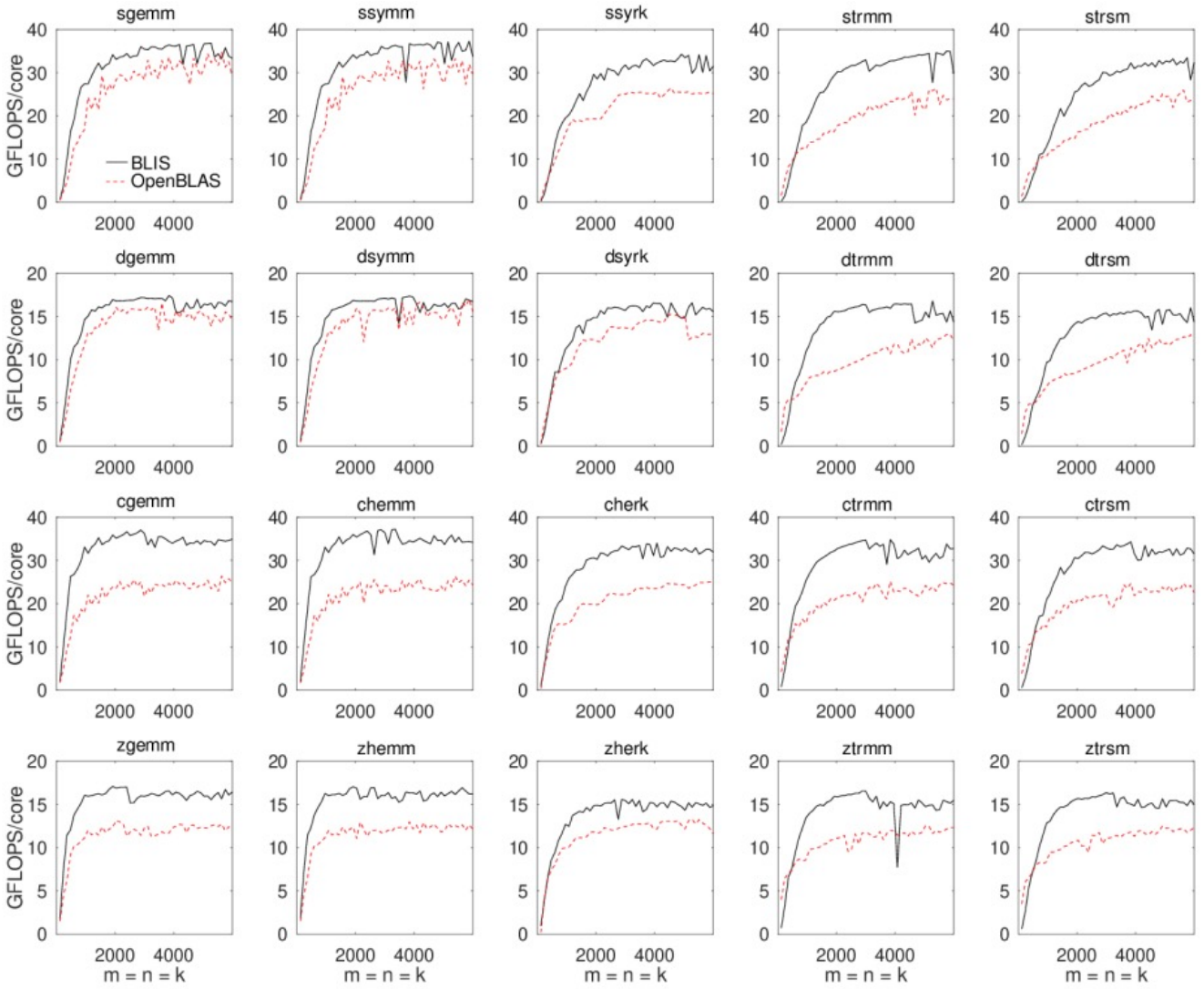
BLIS

The

Other

Guys

What *IS* BLIS?



Neoverse N1

BLIS

The

Other

Guys

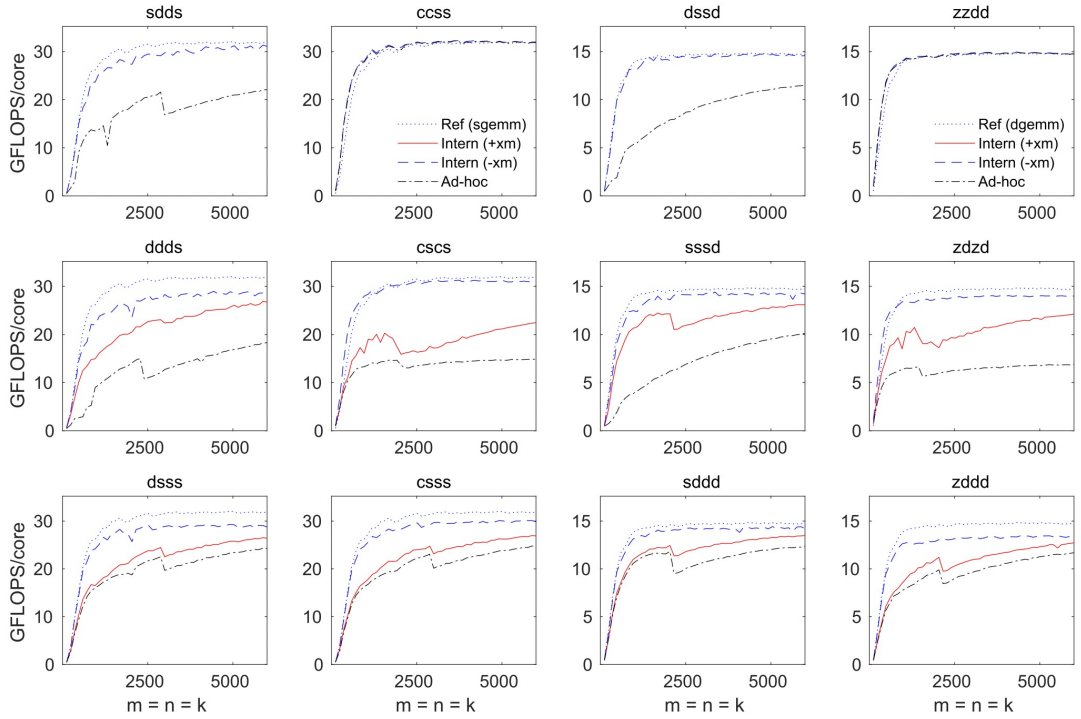
What *IS* BLIS?

1M and other “induced methods”

$$C^r := C^r + A^r B^r - A^i B^i,$$
$$C^i := C^i + A^i B^r + A^r B^i.$$

$$\begin{pmatrix} \gamma^r \\ \gamma^i \end{pmatrix} \mathrel{+}= \begin{pmatrix} \alpha^r & -\alpha^i \\ \alpha^i & \alpha^r \end{pmatrix} \begin{pmatrix} \beta^r \\ \beta^i \end{pmatrix}$$

Mixed-precision/mixed-domain **GEMM**

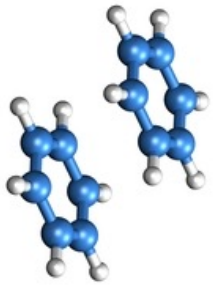


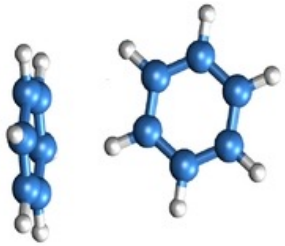
Some History…

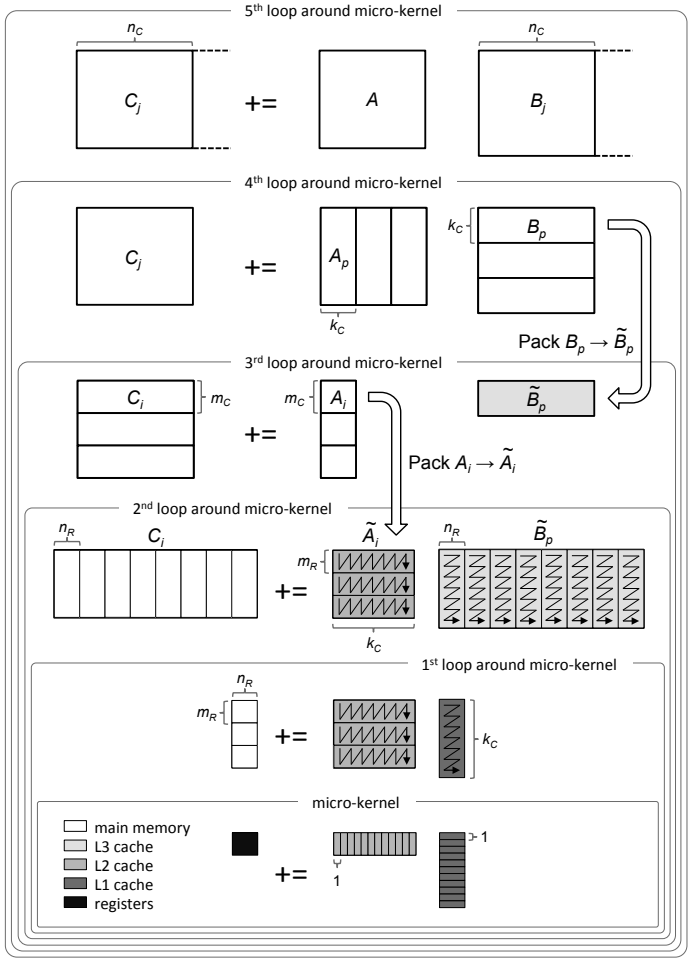
- **2012** • First NSF SI^2 award. Subsequent SI^2/CSSI awards in 2016 and 2020.
- **2012** • First BLIS commit in Github. BLIS existed before this within libflame.
- **2013** • First BLIS retreat.
- **2014** • Multi-threading support.
- **2016** • Runtime multi-architecture support.
- **2019** • “Skinny/unpacked” GEMM
- **2020** • SIAM Activity Group Best Paper Prize
- **2023** • James H. Wilkinson Prize for Numerical Software

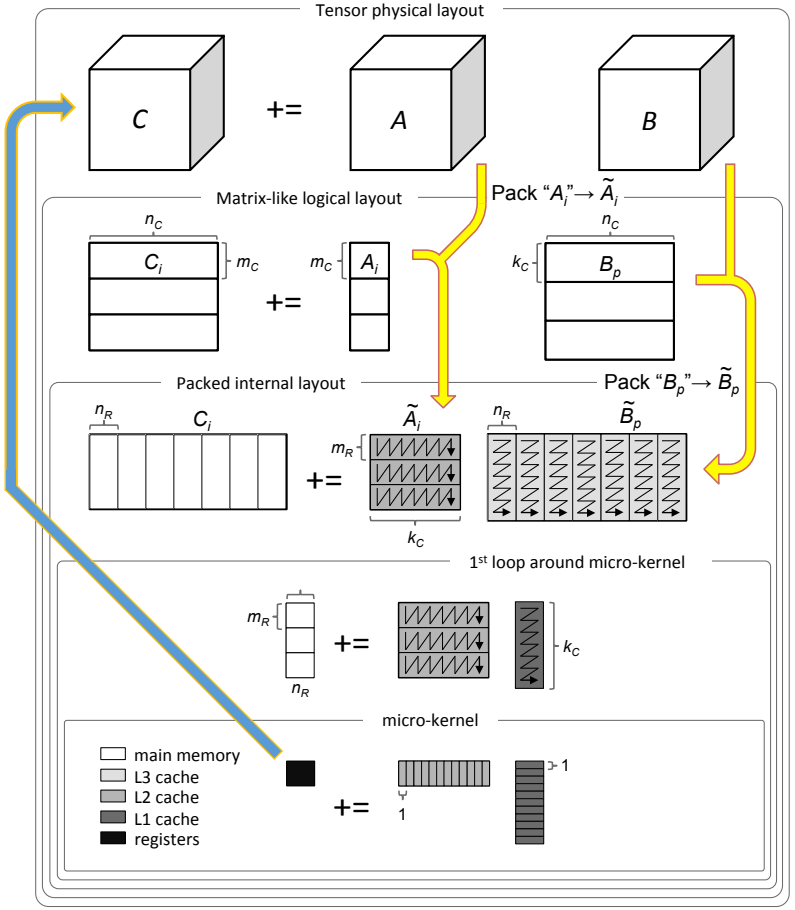
What *SHOULD* BLIS Be?

$$W_{ic}^{ak} = t_{im}^{ae} \cdot V_{ec}^{mk}$$

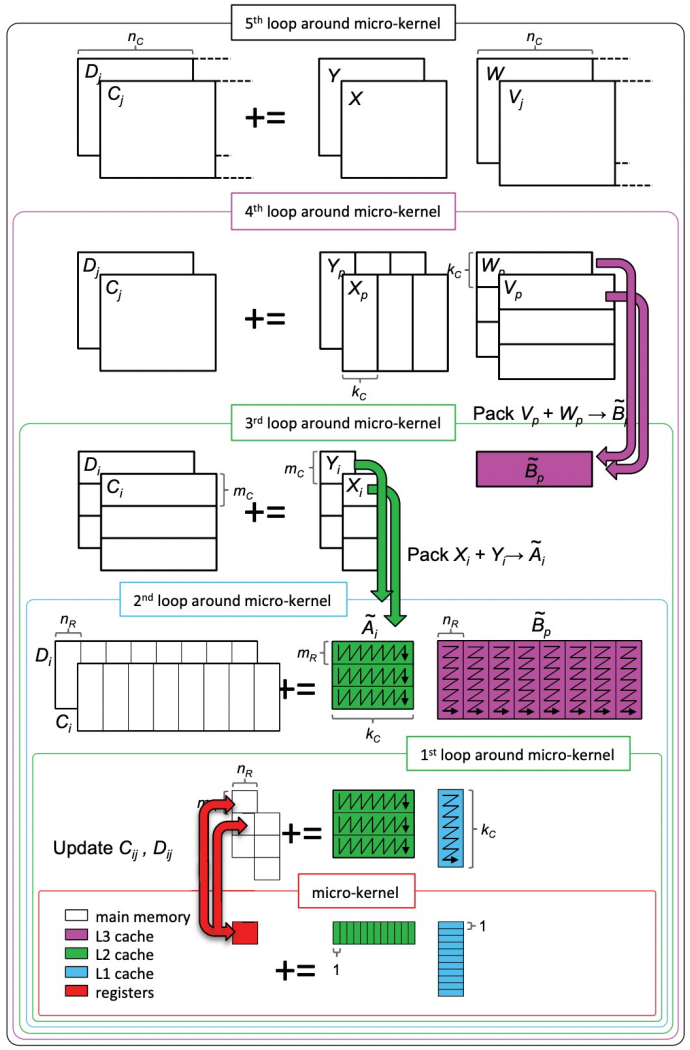








What *SHOULD* BLIS Be?

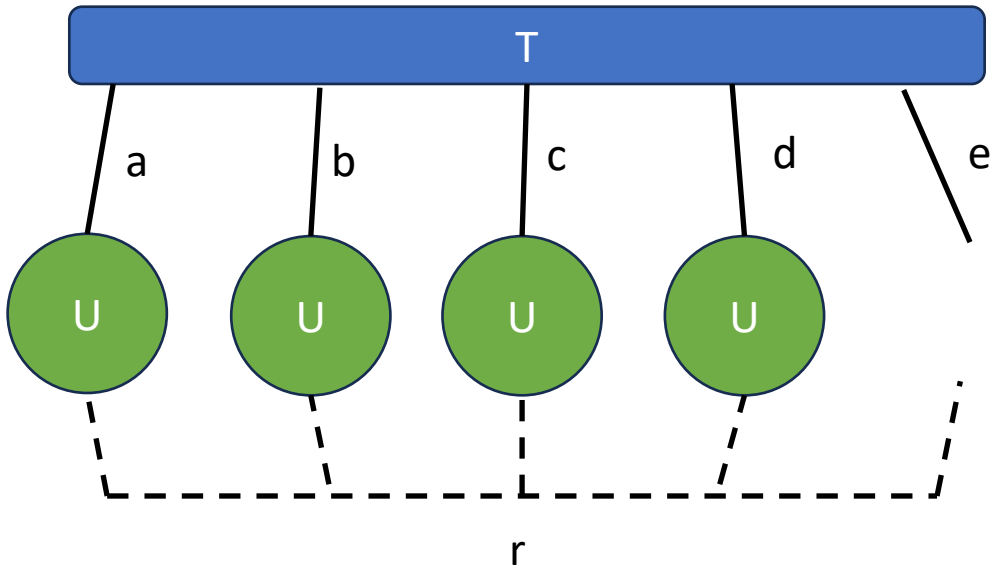


$$
\begin{aligned}
M_0 &= \alpha(A_{00} + A_{11})(B_{00} + B_{11}); & C_{00} &+= M_0; C_{11} += M_0;\\
M_1 &= \alpha(A_{10} + A_{11})B_{00}; & C_{10} &+= M_1; C_{11} -= M_1;\\
M_2 &= \alpha A_{00}(B_{01} - B_{11}); & C_{01} &+= M_2; C_{11} += M_2;\\
M_3 &= \alpha A_{11}(B_{10} - B_{00}); & C_{00} &+= M_3; C_{10} += M_3;\\
M_4 &= \alpha(A_{00} + A_{01})B_{11}; & C_{01} &+= M_4; C_{00} -= M_4;\\
M_5 &= \alpha(A_{10} - A_{00})(B_{00} + B_{01}); & C_{11} &+= M_5;\\
M_6 &= \alpha(A_{01} - A_{11})(B_{10} + B_{11}); & C_{00} &+= M_6;
\end{aligned}
$$

What *SHOULD* BLIS Be?

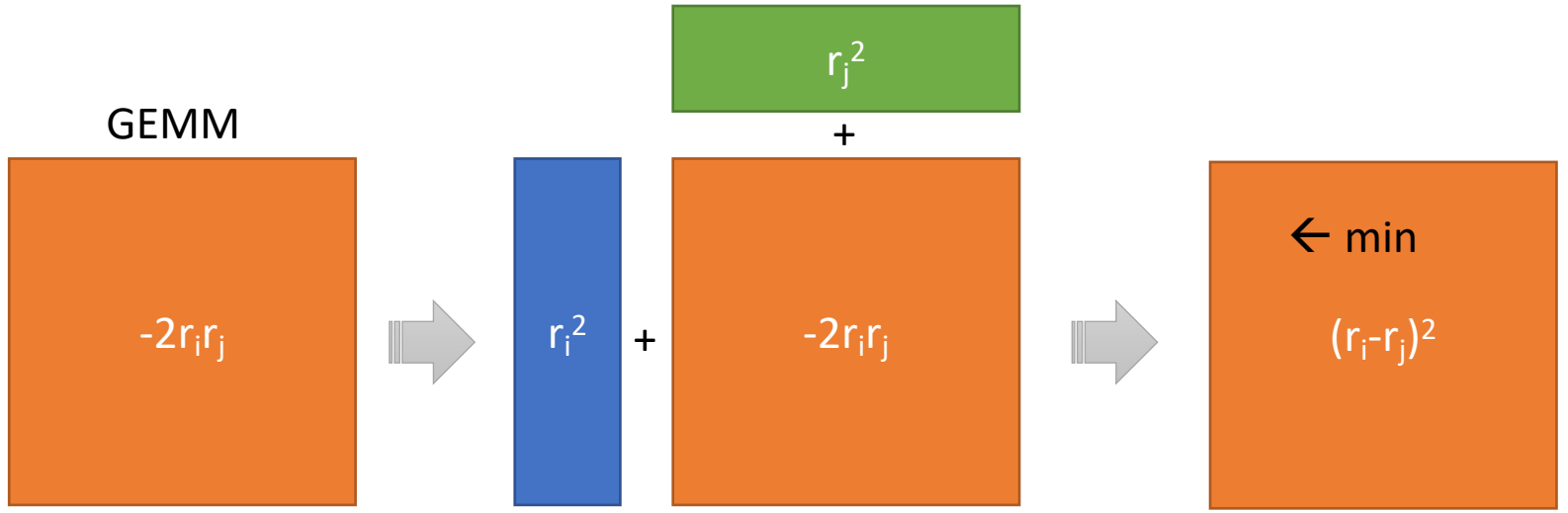
$$G_{re} = \sum_{bcd} \left[\sum_{a} (U_{r;b} U_{r;c} U_{r;d}) U_{ra} T_{ae;bcd} \right]$$

$$\mathbf{C} \mathrel{+}= \mathbf{DAB}$$

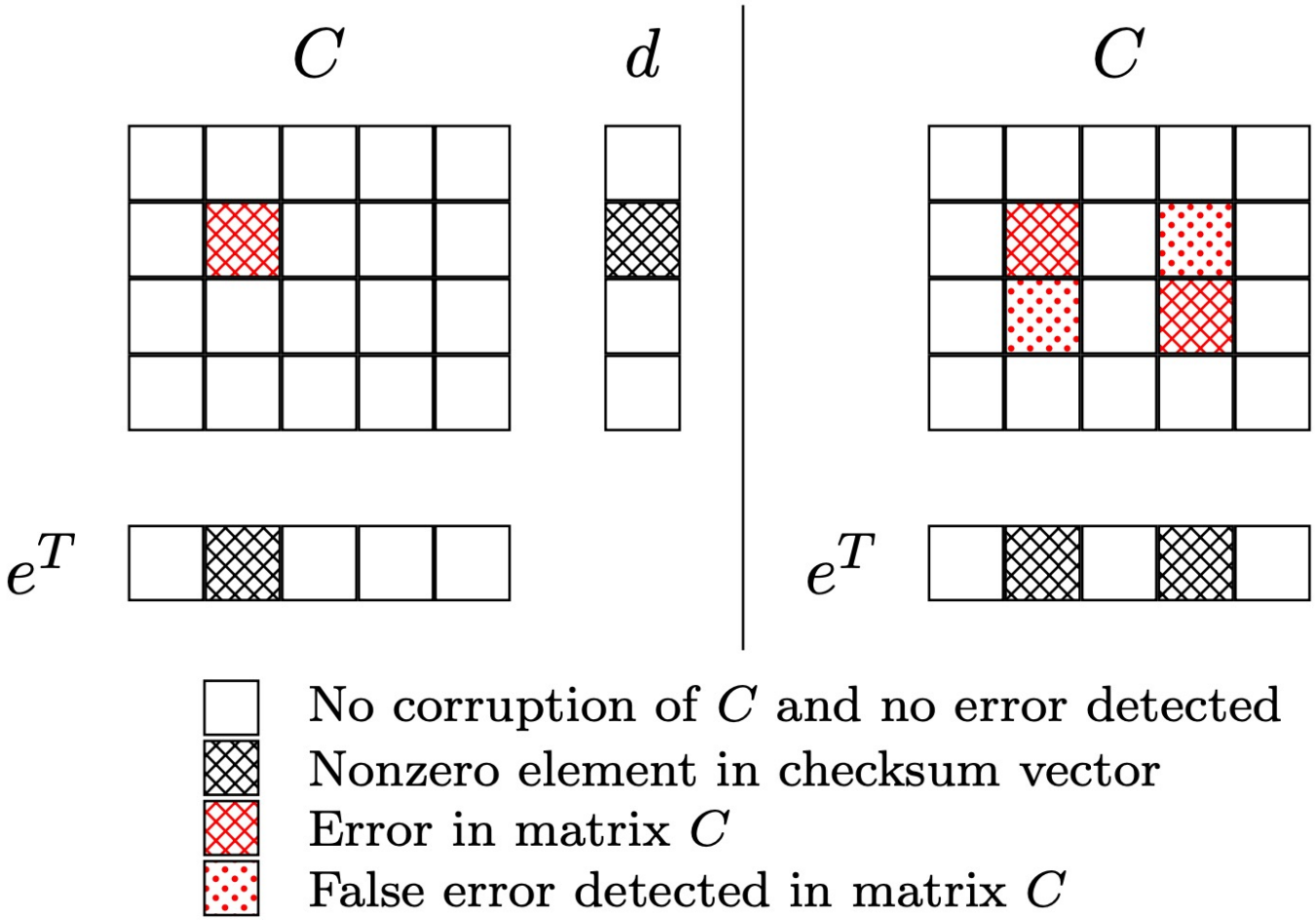


What *SHOULD* BLIS Be?

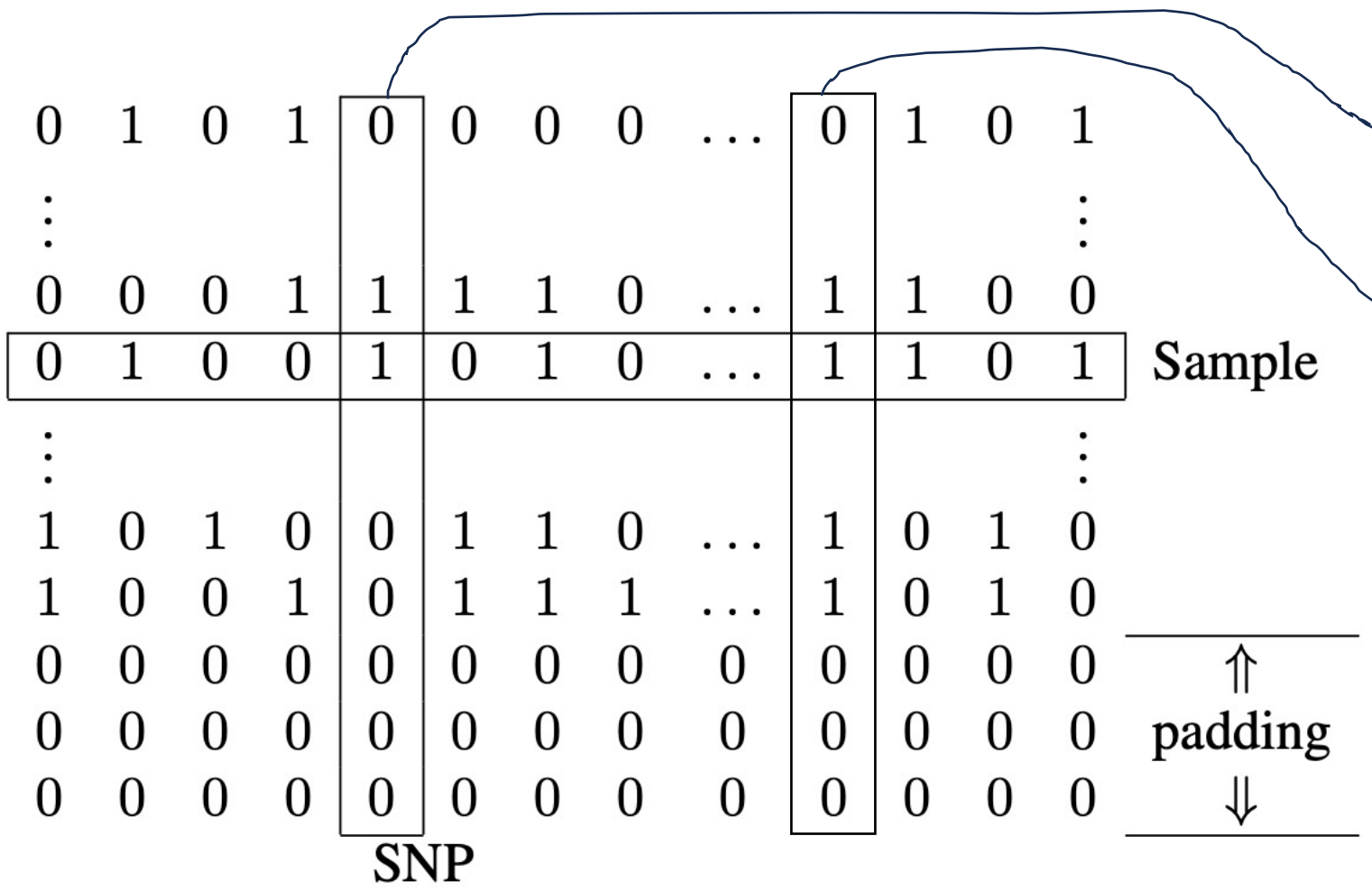
$$KNN_i = \left\{ \mathrm{k}\text{–}\min_j \left| r_i - r_j \right|^2 \right\}_{k=1}^{K}$$

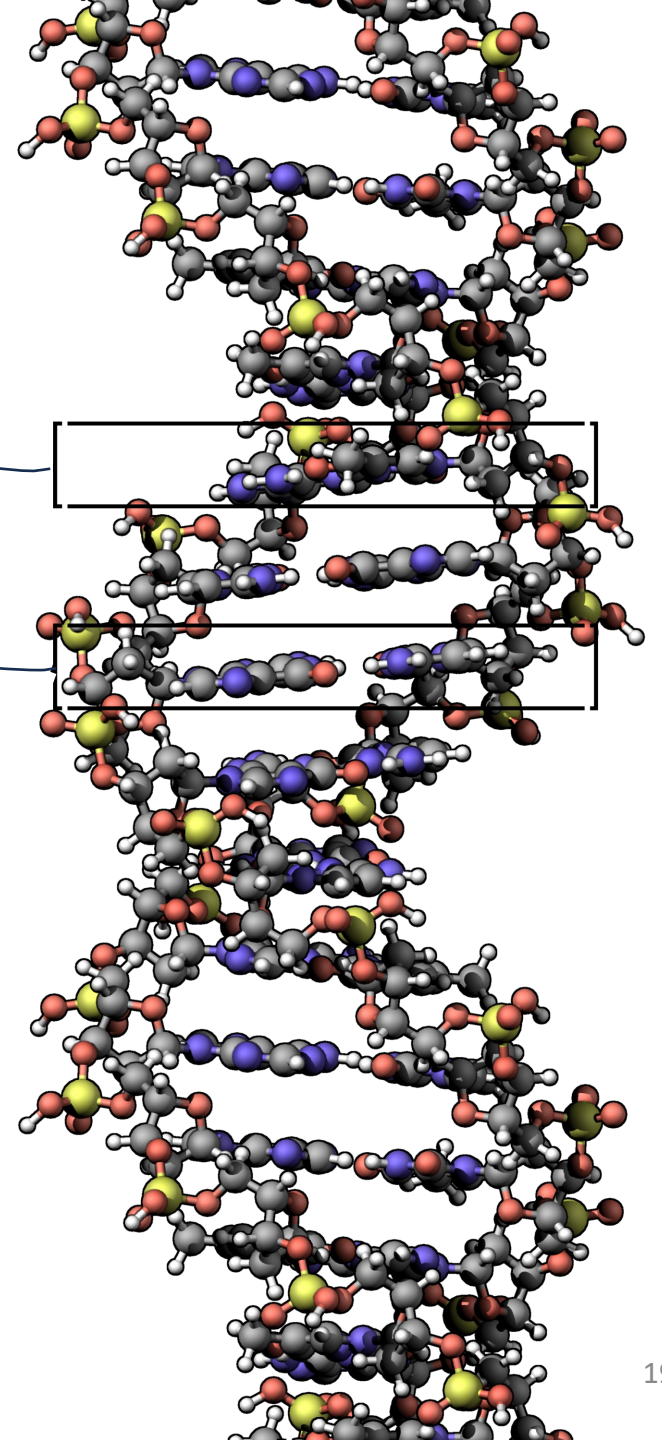


What *SHOULD* BLIS Be?



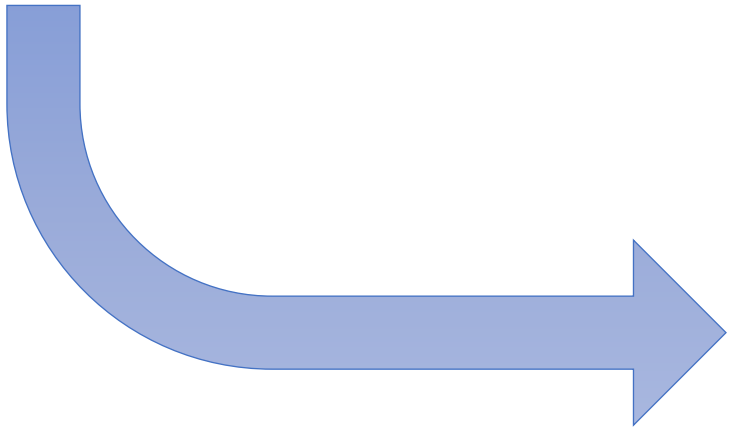
What *SHOULD* BLIS Be?



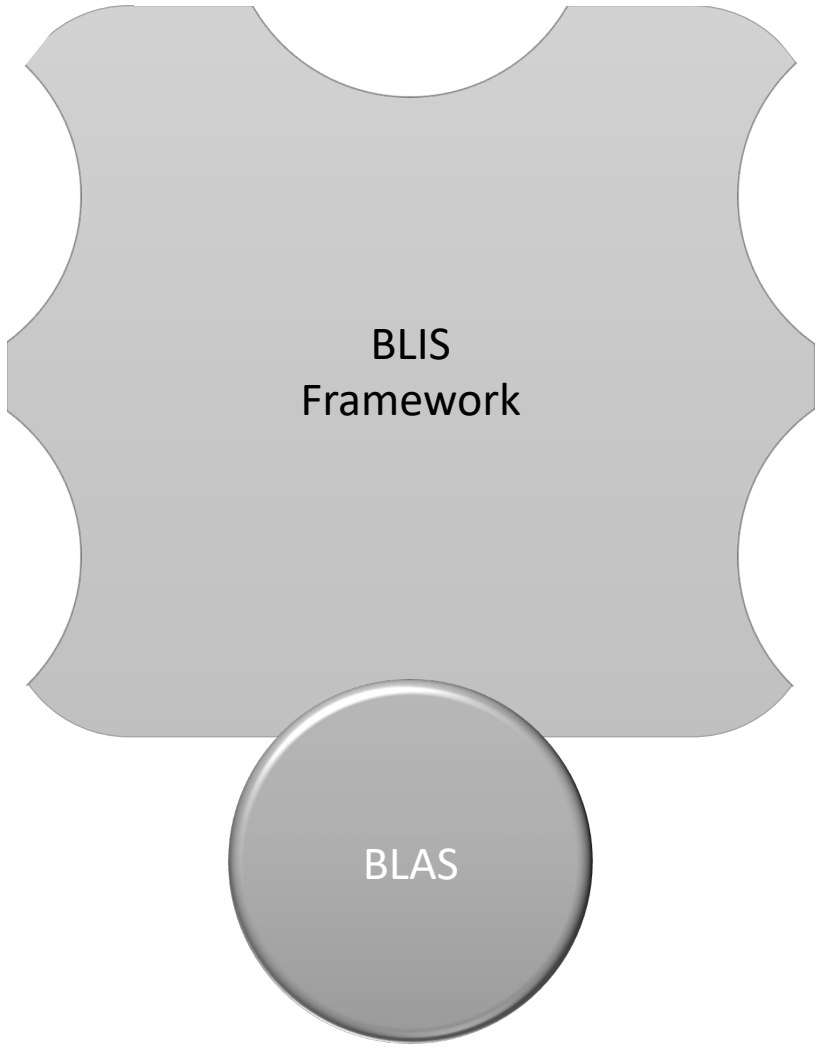


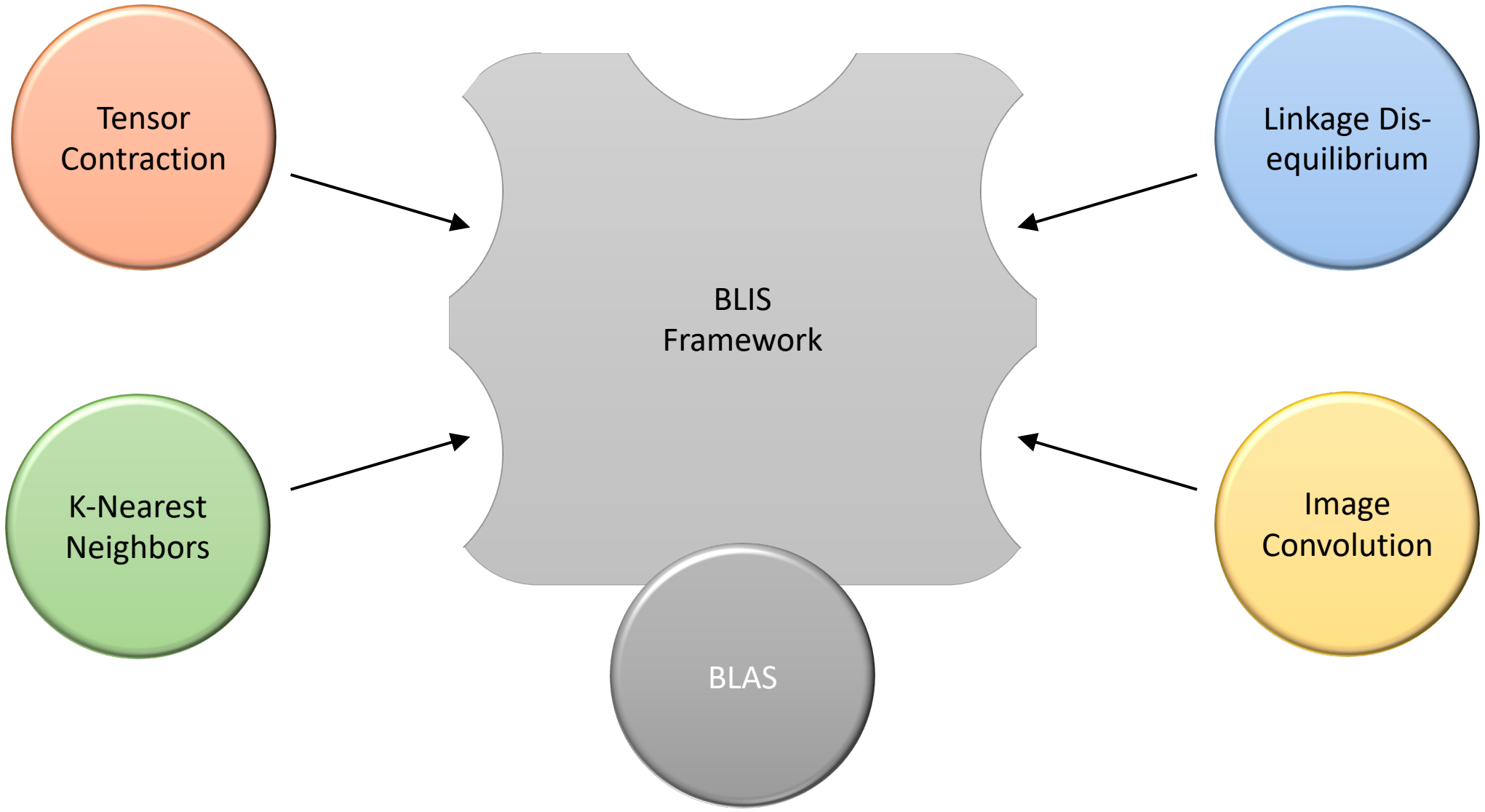
What *SHOULD* BLIS Be?

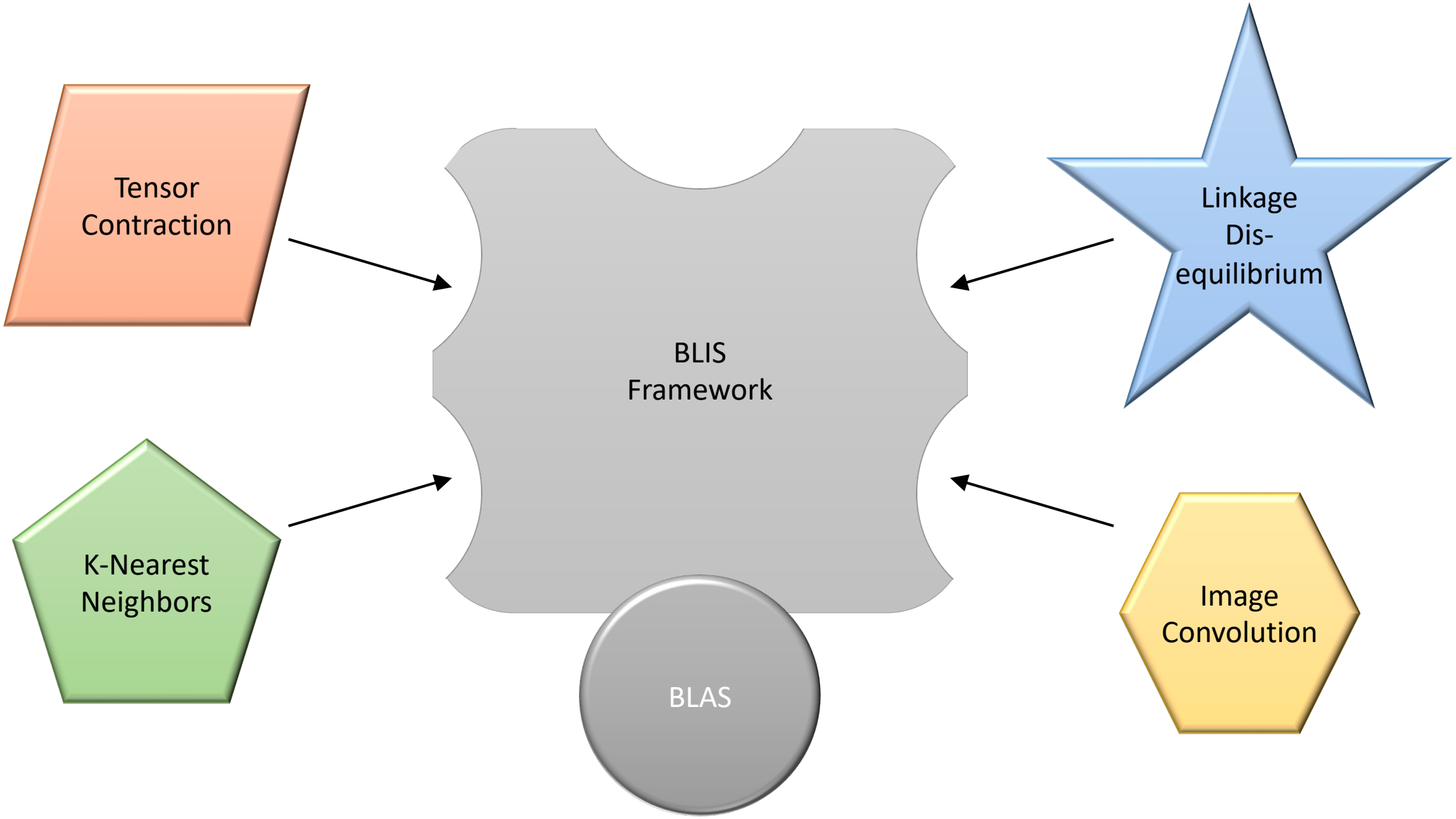
Is it “like GEMM but different”?

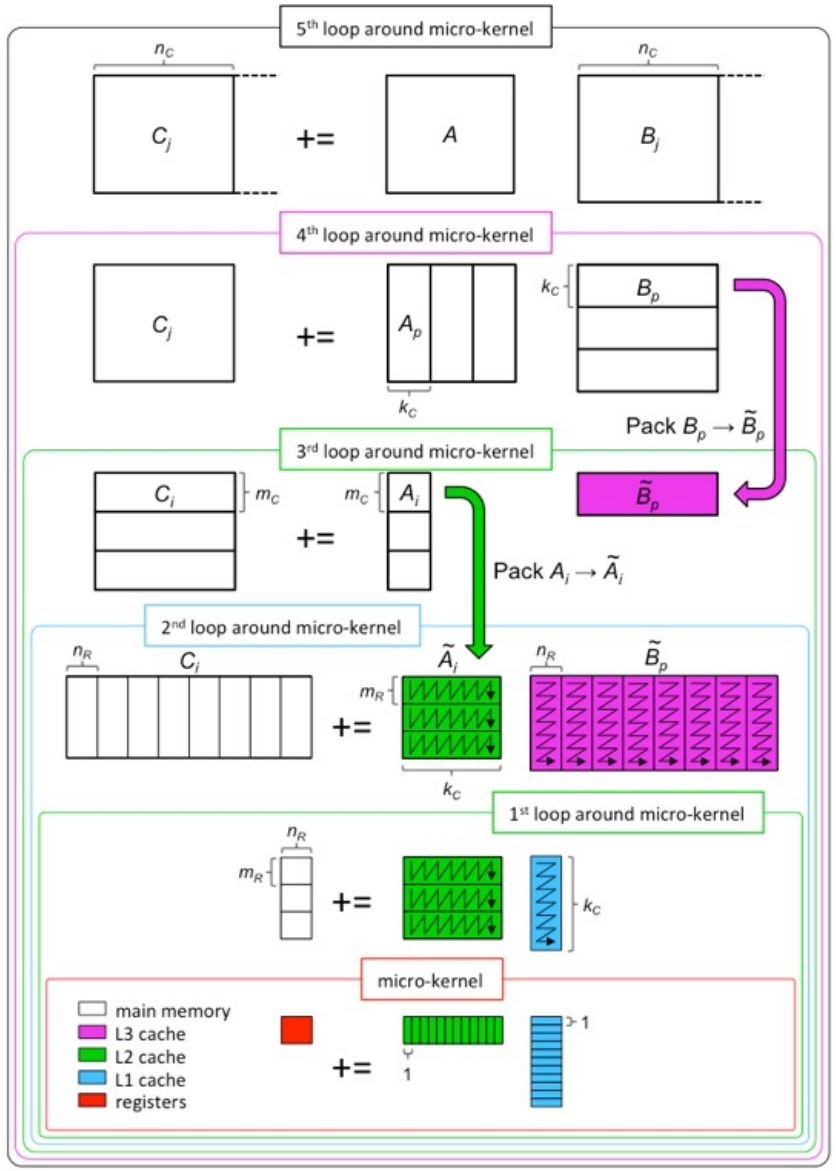


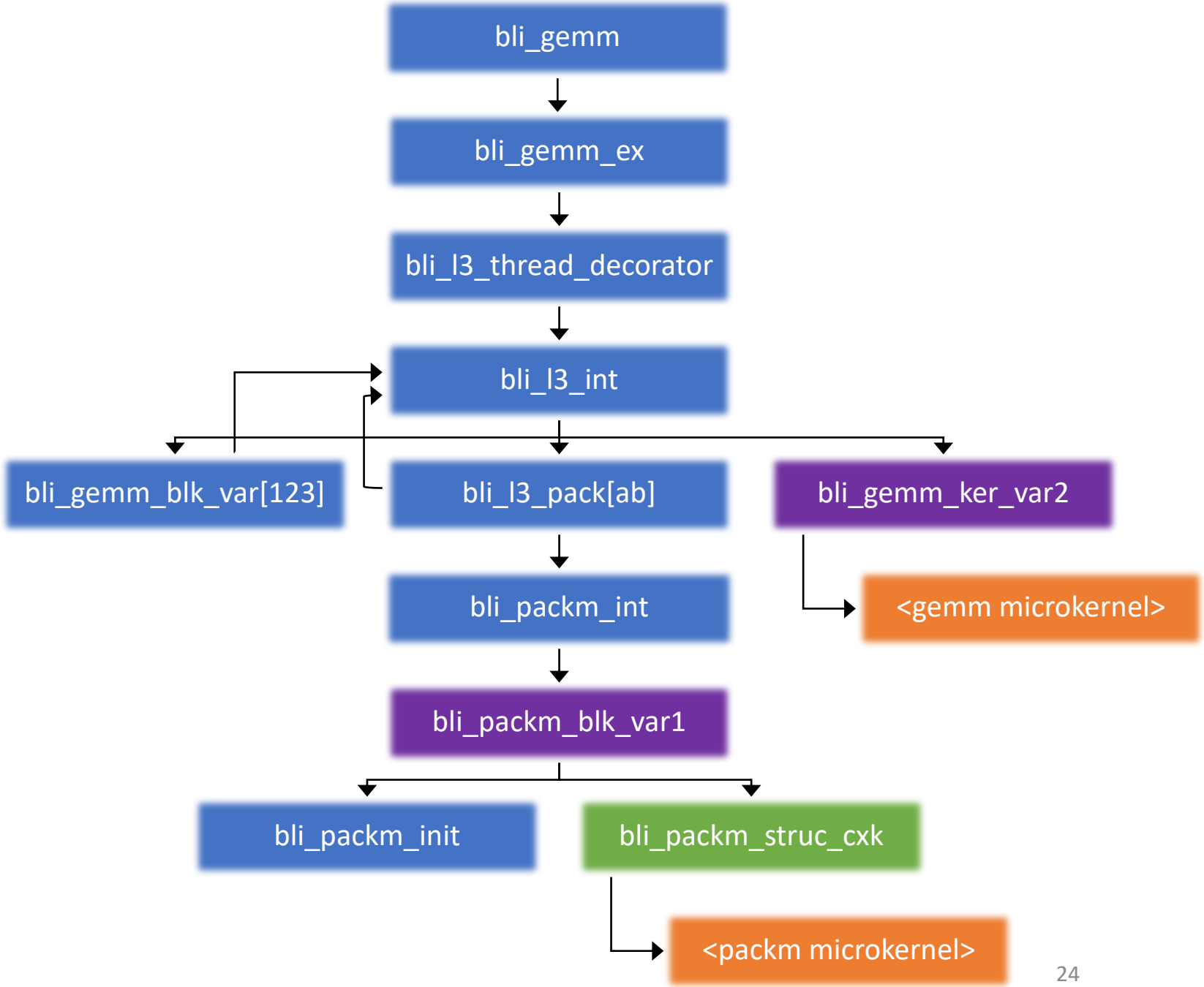
YES!


BLIS
Framework
BLAS


Tensor Contraction
Linkage Dis-equilibrium
BLIS Framework
K-Nearest Neighbors
Image Convolution
BLAS

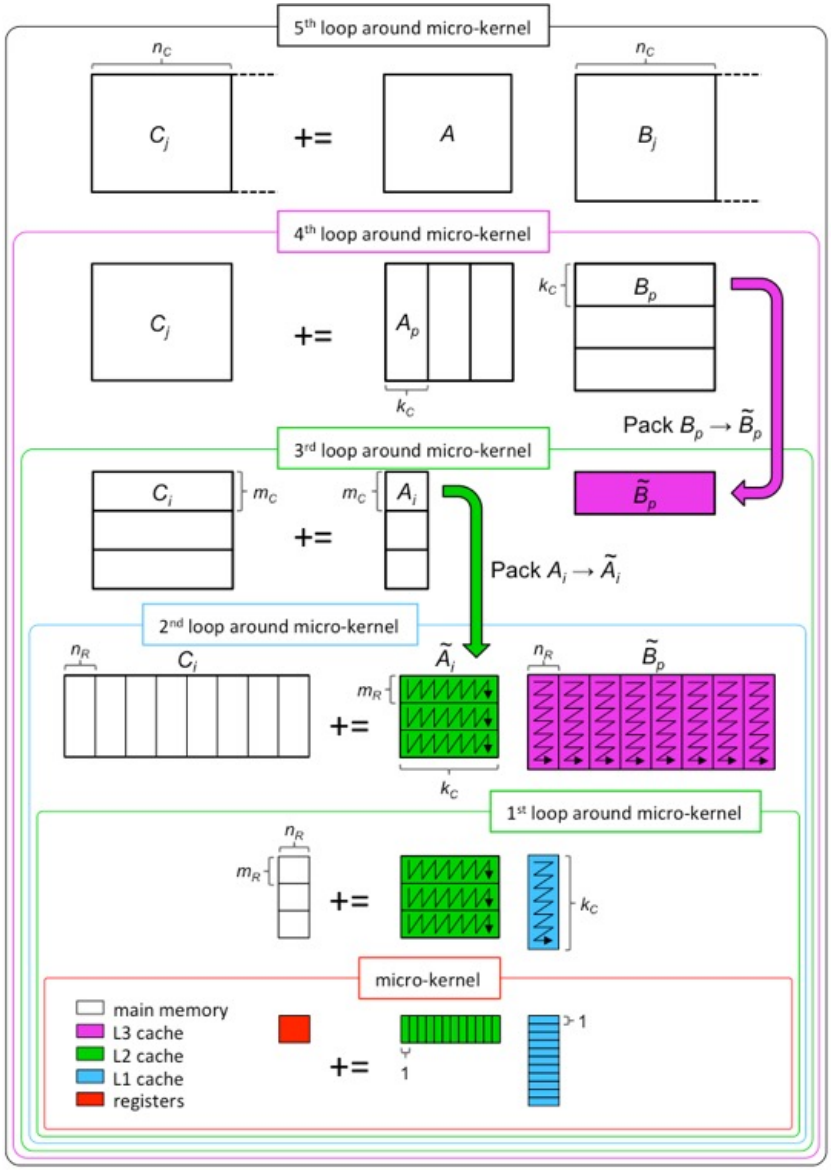

Tensor Contraction
BLIS Framework
Linkage Dis-equilibrium
K-Nearest Neighbors
Image Convolution
BLAS


5th loop around micro-kernel
4th loop around micro-kernel
3rd loop around micro-kernel
2nd loop around micro-kernel
1st loop around micro-kernel
micro-kernel
Pack $B_p \rightarrow \tilde{B}_p$
Pack $A_i \rightarrow \tilde{A}_i$
main memory
L3 cache
L2 cache
L1 cache
registers


bli_gemm
bli_gemm_ex
bli_l3_thread_decorator
bli_l3_int
bli_gemm_blk_var[123]
bli_l3_pack[ab]
bli_gemm_ker_var2
bli_packm_int
<gemm microkernel>
bli_packm_blk_var1
bli_packm_init
bli_packm_struc_cxk
<packm microkernel>

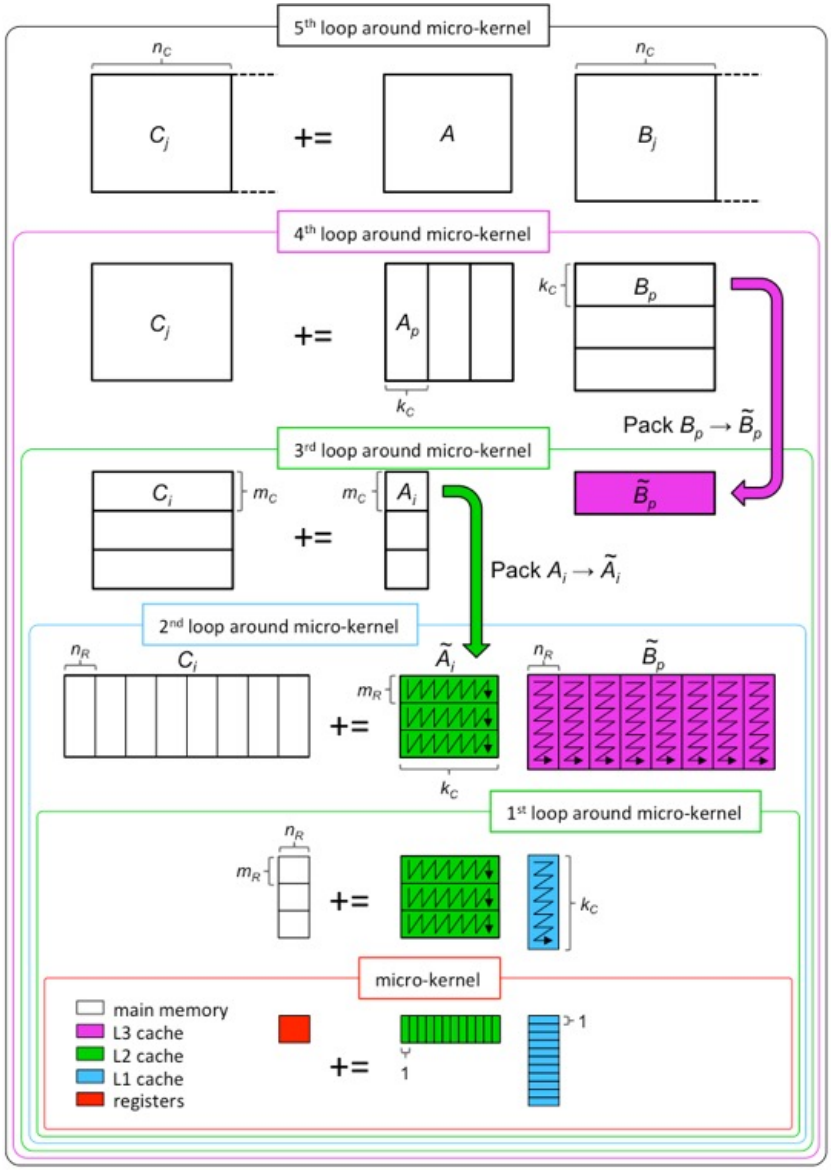
cntl_t* ***control_tree***

```
struct cntl_s
{
  // Basic fields
  // (usually required).
  opid_t          family;
  bszid_t         bszid;
  void_fp         var_func;
  struct cntl_s* sub_prenode;
  struct cntl_s* sub_node;

  // Optional fields (needed
  // only by some operations
  // such as packm).
  void*           params;

  // Internal fields that
  // track "cached" data.
  mem_t           pack_mem;
};
typedef struct cntl_s cntl_t;
```



cntl_t* ***control_tree***

```
struct cntl_s
{
  // Basic fields
  // (usually required).
  opid_t          family;
  bszid_t         bszid;
  void_fp         var_func;
  struct cntl_s* sub_prenode;
  struct cntl_s* sub_node;

  // Optional fields (needed
  // only by some operations
  // such as packm).
  void*           params;

  // Internal fields that
  // track "cached" data.
  mem_t           pack_mem;
};
typedef struct cntl_s cntl_t;
```

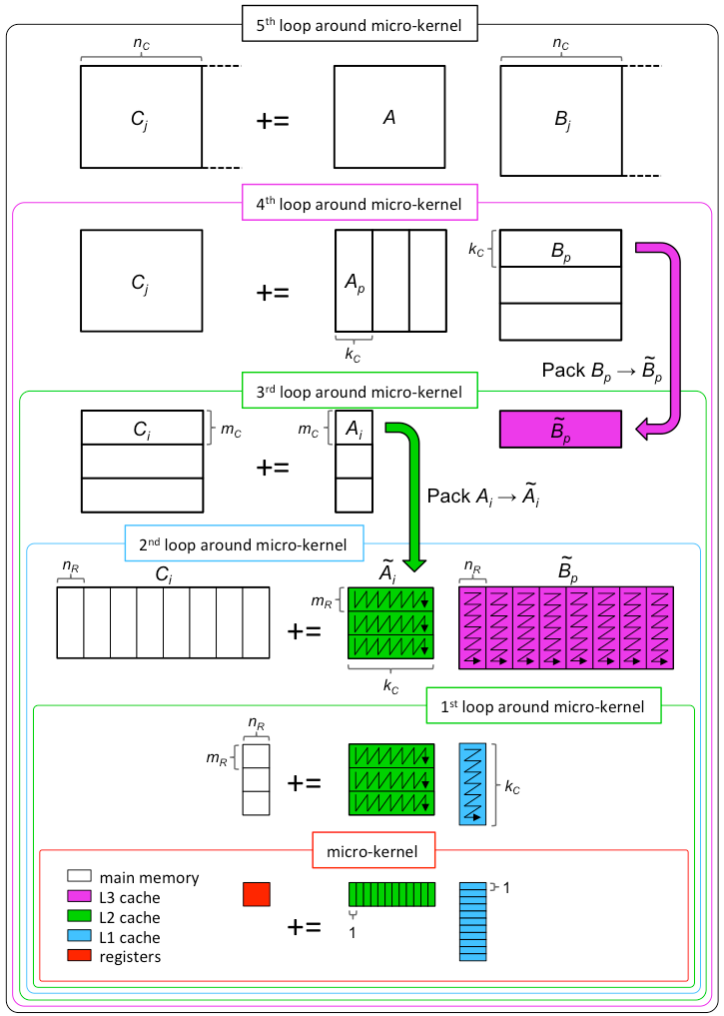


thrinfo_t* ***thread_control_tree***

```
struct thrinfo_s
{
  // The thread communicator for the
  // other threads sharing the same
  // work at this level.
  thrcomm_t*          ocomm;

  // Our thread id within the
  // ocomm thread communicator.
  dim_t               ocomm_id;

  // The number of distinct threads
  // used to parallelize the loop.
  dim_t               n_way;

  // What we're working on.
  dim_t               work_id;

  bool                free_comm;
  bszid_t             bszid;

  struct thrinfo_s*   sub_prenode;
  struct thrinfo_s*   sub_node;
};
typedef struct thrinfo_s thrinfo_t;
```

A Vision Five(+) Years in the Making

A Vision Five Years in the Making: 2018

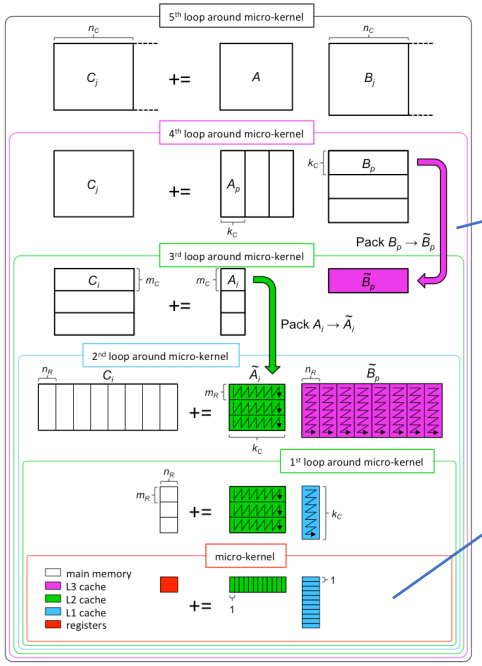


`X.partition(…)`

`X.pack(…)`

`X.accum_ukr(…)`

A Vision Five Years in the Making: 2019



`X.pack(…)`

- Scale row and/or columns?
- Cwise product?
- Generate on-the-fly: Toeplitz, Hankel, etc. (convolution)?
- Special ukr structure?
- Why not!

`X.gemm_ker(…) or X.gemm_ukr(…)`

- Accumulate cwise product (matrix dot)?
- Apply filter (e.g. Gaussian kernel)?
- $\alpha = (AB)_{ij}(B^T A^T)_{ij} = C_{ij}C_{ji}$?
  (no I'm not kidding, I need this one)

High-level algorithm (FLAMEy part): Composable pieces, loops, whatever

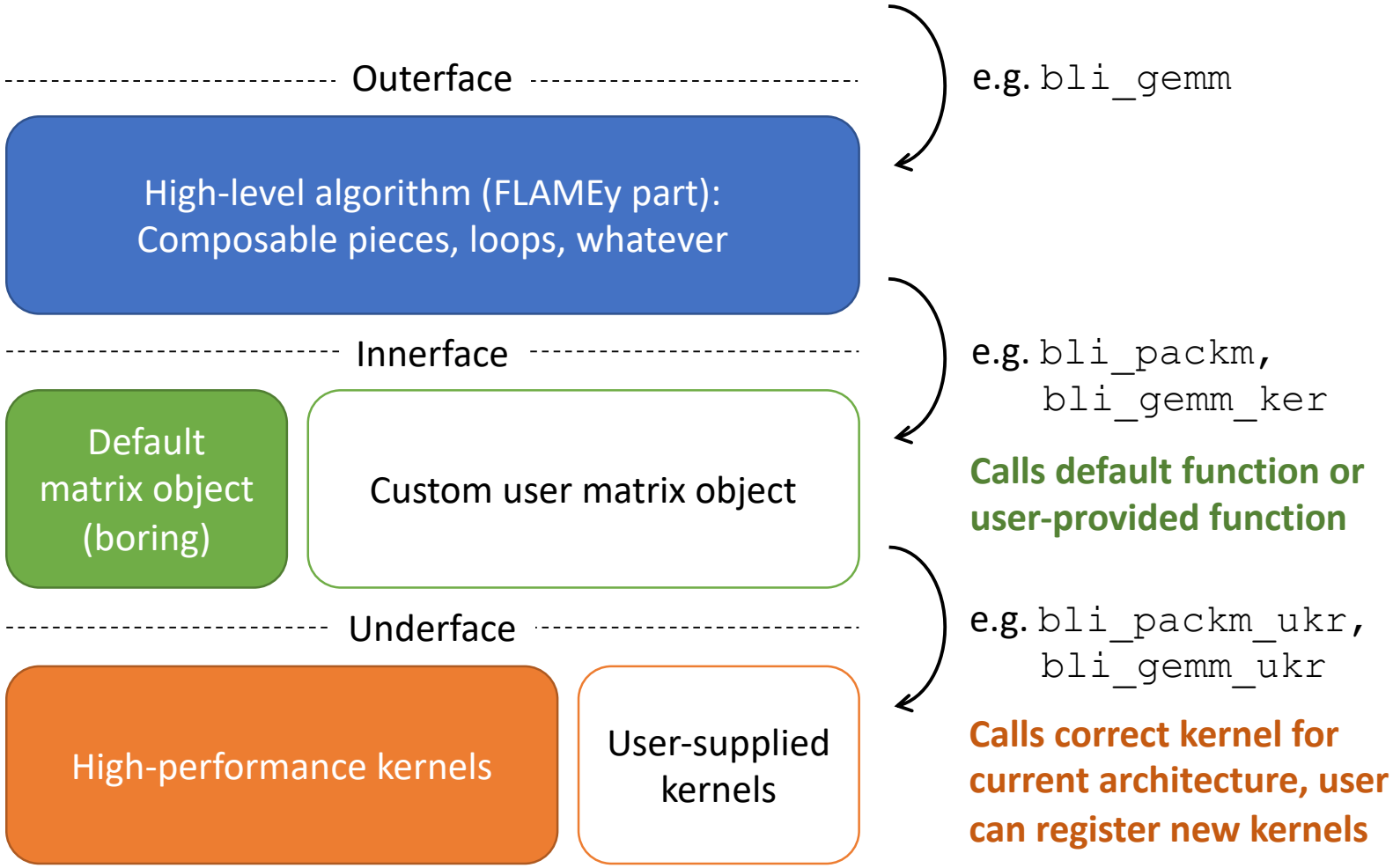
Default matrix object (boring)

Custom user matrix object

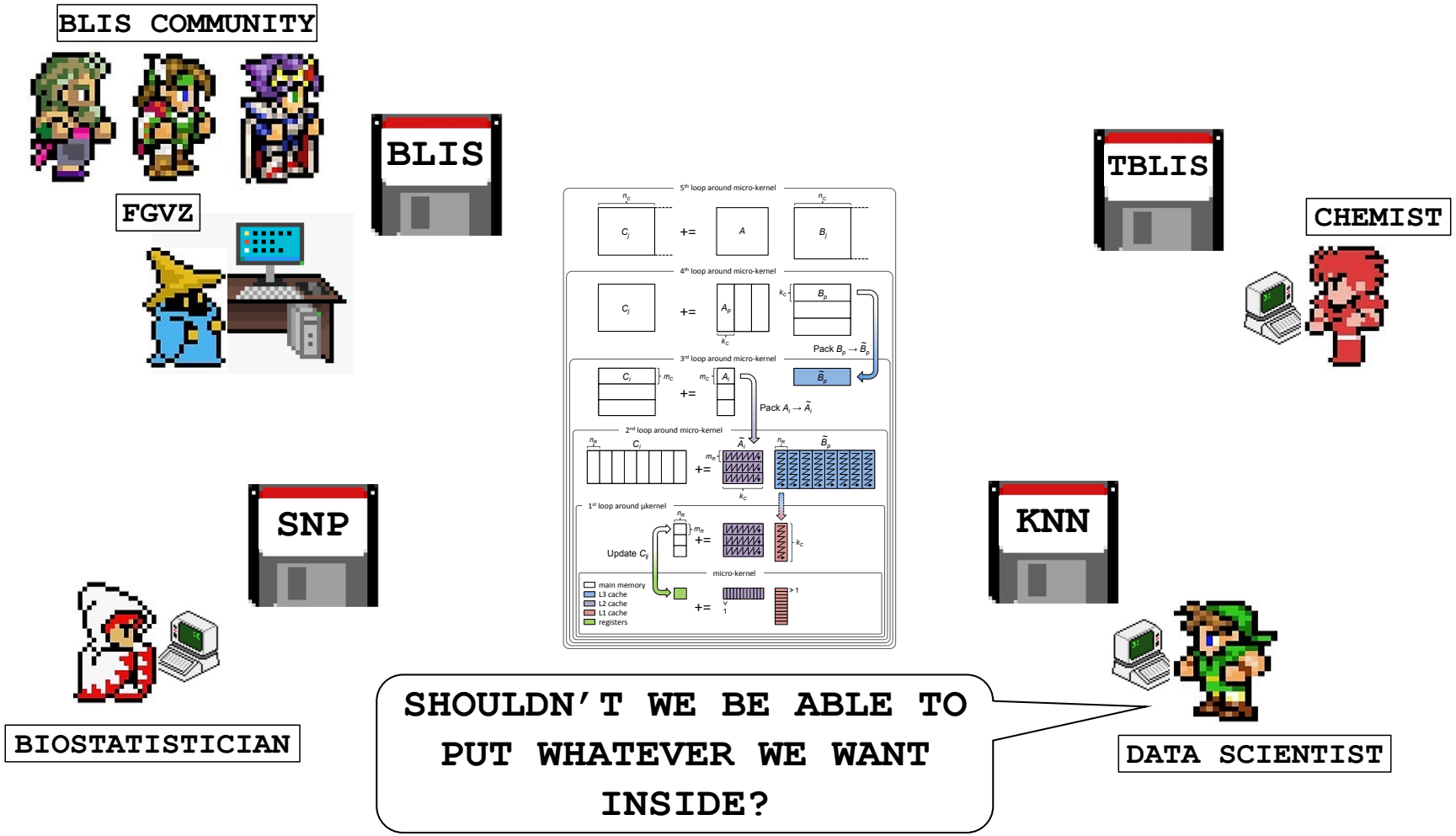
High-performance kernels

User-supplied kernels

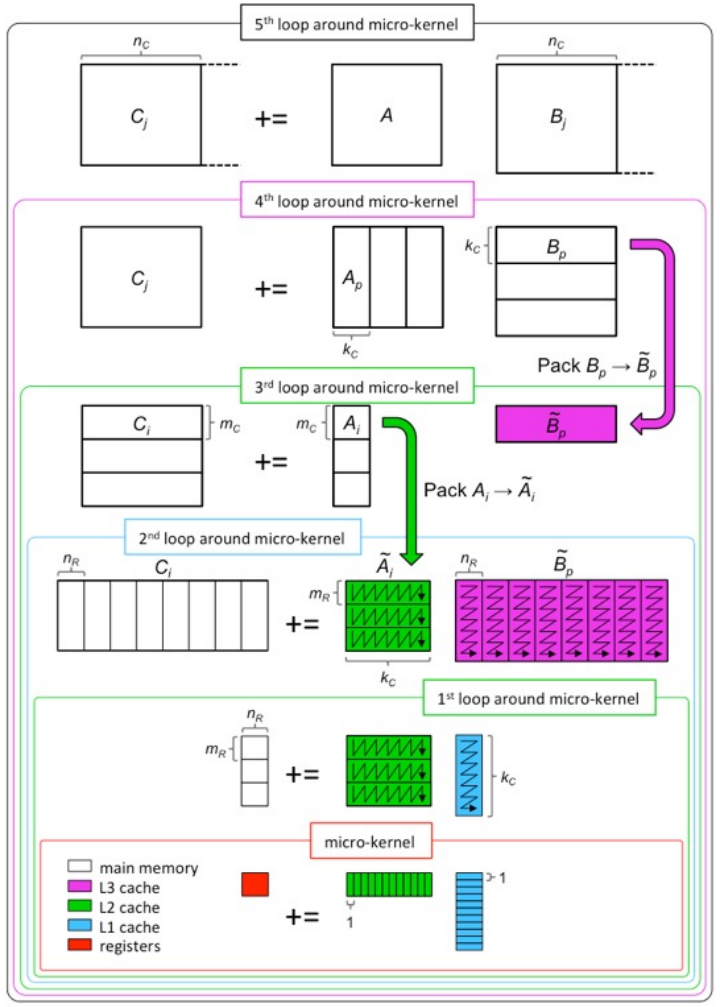
A Vision Five Years in the Making: 2020



A Vision Five Years in the Making: 2021



A Vision Five Years in the Making: 2022



bli_gemm_blk_var2
Partition by: **BLIS_NC**
Thread by: **BLIS_JC_NT**

bli_gemm_blk_var3
Partition by: **BLIS_KC**
Thread by: **none**

bli_packm_blk_var1
Partition by: **BLIS_NR**
Kernel: **bli_dpackm_struc_cxk**
Direction, Invert Diagonal, …

bli_gemm_blk_var1
Partition by: **BLIS_MC**
Thread by: **BLIS_IC_NT**

bli_packm_blk_var1
Partition by: **BLIS_MR**
Kernel: **bli_dpackm_struc_cxk**
Direction, Invert Diagonal, …

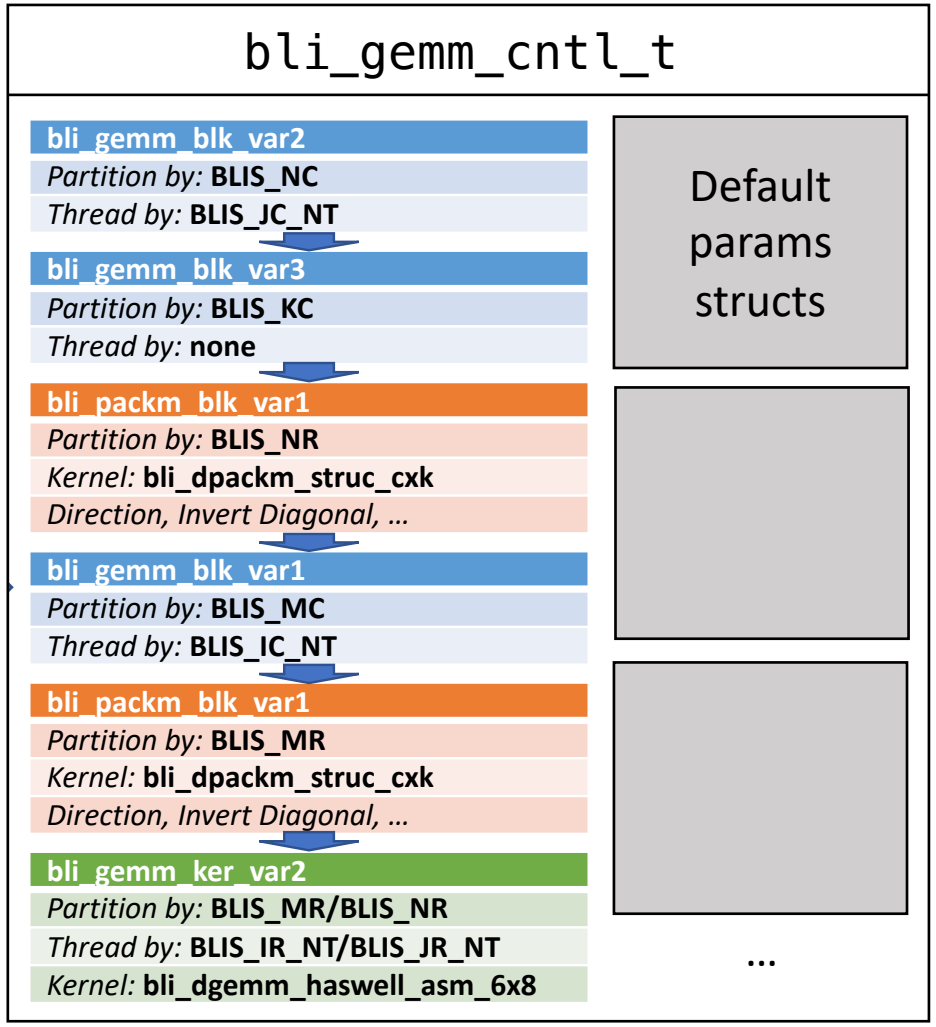
bli_gemm_ker_var2
Partition by: **BLIS_MR/BLIS_NR**
Thread by: **BLIS_IR_NT/BLIS_JR_NT**
Kernel: **bli_dgemm_haswell_asm_6x8**

Phase I: A Control Tree API

Complete revamp of the control tree structure:

- ***ALL*** parameters used by variants (partitioning, packing, macrokernel) encoded using the parameters struct. No more blocksize or microkernel IDs.
- Type information is stored and used to update the control tree with new information (user-defined kernels, blocksizes, etc.).
- 1m is built-in: most user-defined modifications continue to work with 1m.
- Users can supply blocksizes/kernels from the context, or arbitrary integers/pointers.
- For gemm-like operations, the control tree is stored entirely on the stack and shared by all threads.
- Users can supply additional operation-specific data via opaque structs.

Phase I: A Control Tree API




```
bli_gemm_cntl_set_pack[ab]_var(…)
bli_gemm_cntl_set_pack[ab]_ukr(…)
bli_gemm_cntl_set_pack[ab]_params(…)
bli_gemm_cntl_set_pack[ab]_schema(…)

bli_gemm_cntl_set_{mc,nc,kc}(…)
bli_gemm_cntl_set_{mr,nr}(…)

bli_gemm_cntl_set_ukr(…)
bli_gemm_cntl_set_params(…)
```

Phase II: External Plugin API

Users will often want to use custom kernels. With plugins…

- Users can code kernels outside of BLIS and register them to the context.
- Have reference kernels compiled for all configured architectures using the original BLIS compiler and flags.
- Provide optimized kernels for architectures of choice. These will override reference kernels when running on the matching architecture.
- Also provide blocksizes and kernel preferences.
- Register a (near-)unlimited number of new kernels, blocksizes, and preferences.
- Kernels are compiled into a static and/or shared object for inclusion in the user's codebase.

Phase II: External Plugin API

System or user-installed BLIS

$SHARE/blis/configure-plugin

Autogenerated plugin template

User-defined reference kernels

Optimized kernels

Plugin registration

Provide kernels IDs to host application/library

.so/.a

.so/.a

User library or application

Phase III: Mixed-Precision/Mixed-Domain

Most BLIS kernels are now explicitly mixed-precision

- Currently supported: all combinations of sdcz
- Packing (reference kernels are templated C99 code)
- Microkernel (reference mixed-precision microkernel is a type-casting wrapper kernel)
- Real <-> complex usually not needed (handled by induced methods)
- Soon to be extended to Level 1 and 2 BLAS

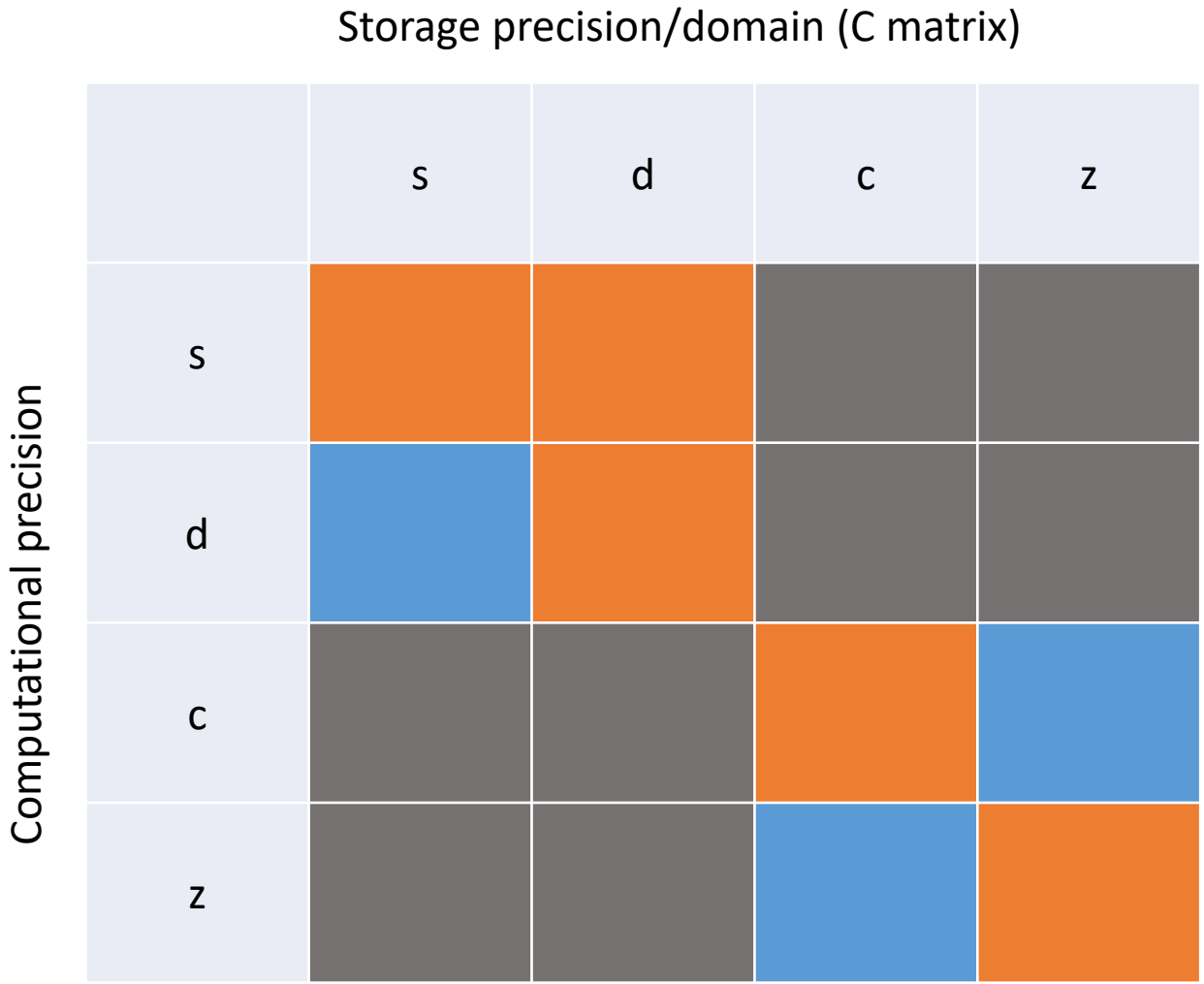
Phase III: Mixed-Precision/Mixed-Domain

Example: optimization for GEMM microkernel

Hand-optimized

Reference (wrapper ukr)

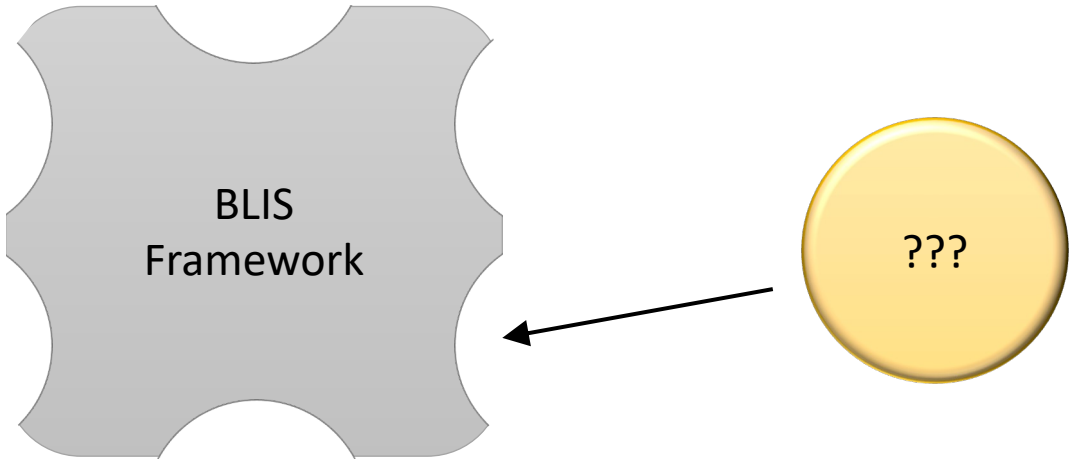
Not used



Phase IV: Expanded Precision (In Progress)

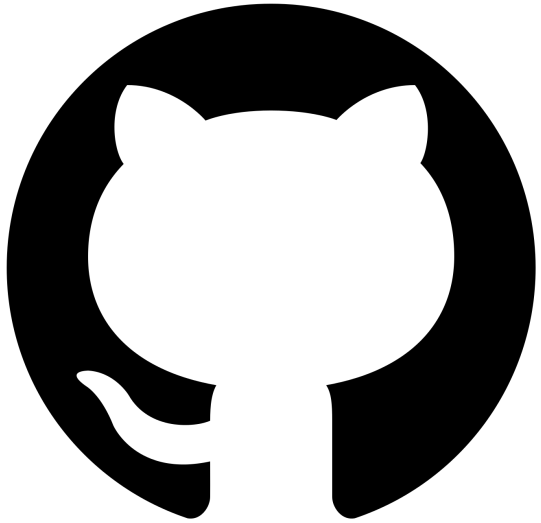
Framework support for reduced (e.g. float16, bfloat16, int8/16) and extended (e.g. double-double, float128) support.

- For “core” types: full reference kernel support, either via HW support or software implementation/upcasting when not available.
- Mixed-precision with standard and other expanded types.
- Rich macro library for Level 0 (scalar) operations.
- Potentially: run-time registration of new types!



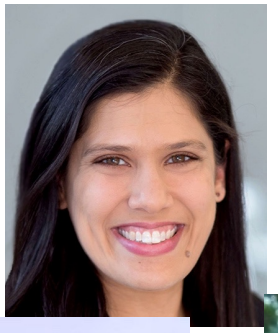
Demo of plugins and control tree API tomorrow after lunch!

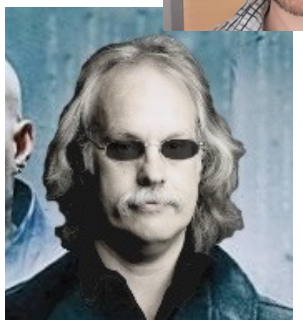
BLIS is a Community Project

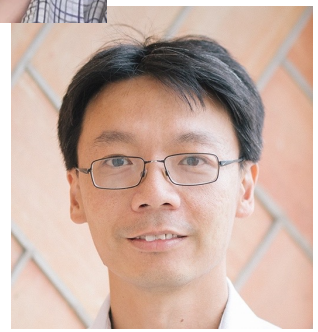


115+ contributors

Bryan Marker







AMD

ARM®

ORACLE

NATIONAL INSTRUMENTS

hp

Microsoft

NEC

NASA

MathWorks

IBM

HUAWEI

intel

NSF

Qualcomm

NVIDIA

facebook

SHPC

ODEN INSTITUTE
FOR COMPUTATIONAL ENGINEERING & SCIENCES