Managing the Dense Linear Algebra Software Stack

Robert van de Geijn
Department of Computer Science
Institute for Computational Engineering and Sciences

Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

UT Austin

Faculty/Staff

Don Batory
Victor Eijkhout
Andreas Gerstlauer
Maggie Myers
John Stanton
Robert van de Geijn
Field Van Zee

Graduate Students

Kyungjoo Kim (E.M.)
Tze Meng Low (CS)
Bryan Marker (CS)
Ardavan Pedram (ECE)
Scott Rabidoux
Martin Schatz (CS)
Tyler Smith (CS)

Univ. Jaume I, Spain

Faculty

Gregorio Quintana-Orti
Enrique Quintana-Orti

Graduate Students

Manuel Fogue

RWTH Aachen University, Germany

Faculty

Paolo Bientinesi

Graduate Students

Diego Fabregat
Matthias Petschow

Georgia Tech

Faculty Jack Poulson (soon)

Univ. of Chicago/Argonne Leadership Computing Facility

Faculty Jeff Hammond

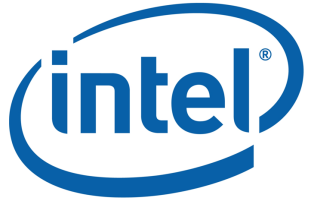
Univ. Complutense de Madrid

Fran Igual

Sponsors!

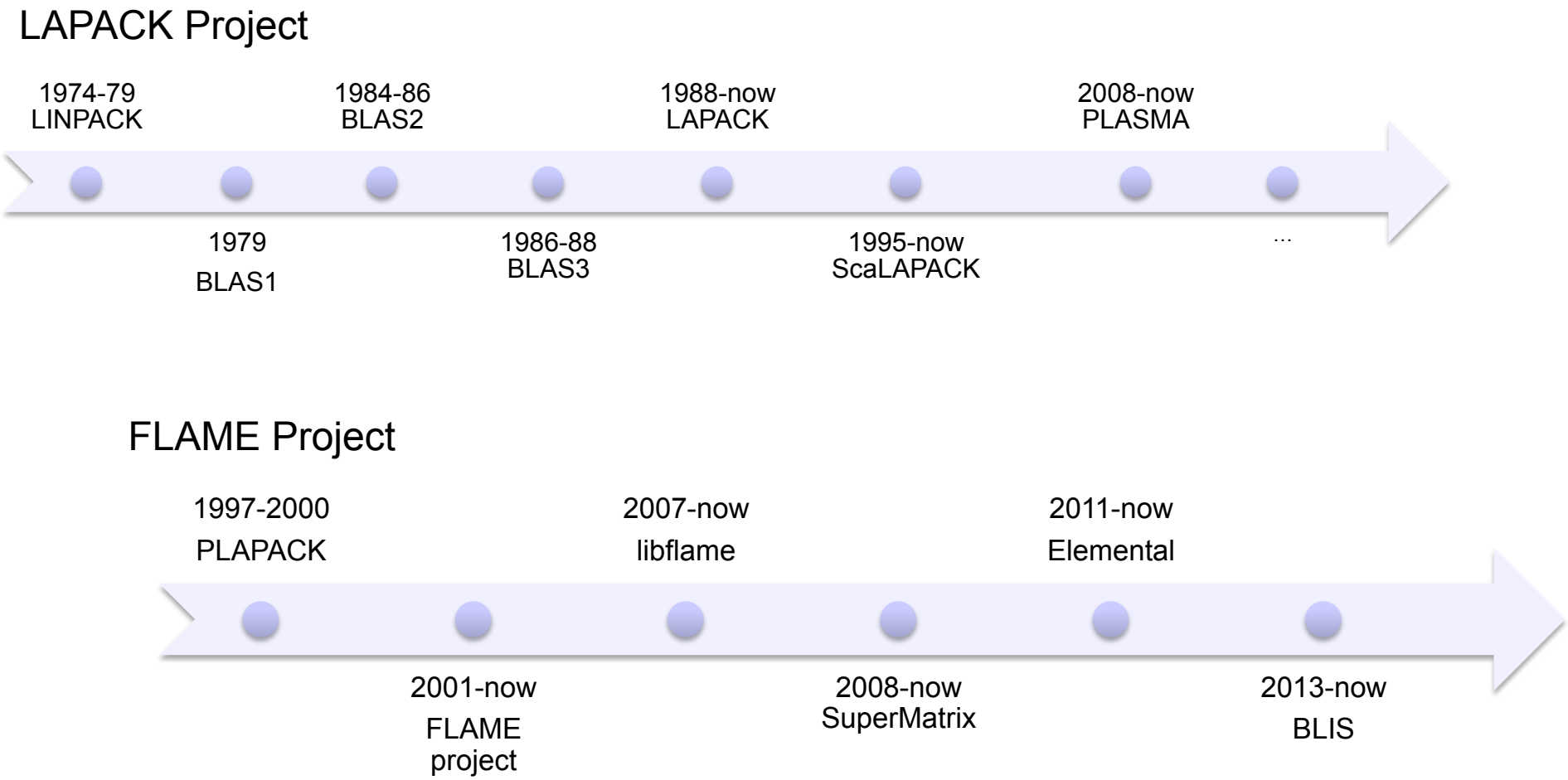
AMD

NEC



Your name here!!

Open Source
Dense Linear Algebra Libraries



Productizing the research

van de Geijn (PI-UTCS), Batory (CoPI-UTCS), Eijkhout (CoPI-TACC), Hammond (CoPI-ALCF/Uchicago), Myers (CoPI-UTCS), Stanton (CoPI-UTChem),

A Linear Algebra Software Infrastructure for Sustainable Innovation in Computational Chemistry and other Sciences

- Goal:
  - Take fundamental research (FLAME)
  - Build a new DLA software infrastructure
  - Use it for sustainable innovation in Chemistry/science

- Funded by NSF Softw. Infrastructure for Sustained Innovation (SI^2):

  "… identifies advancing new computational infrastructure as a priority for driving innovation in science and engineering."

The DLA Software Stack

ScaLAPACK	⟺	Elemental
PLASMA/Quark	⟺	libflame + SuperMatrix
LAPACK	⟺	libflame
BLAS	⟺	BLIS

FLAME

- Derive algorithms to be correct
  - Extraordinary confidence in correctness
  - Choose from Families of algorithms
- Avoid indexing
  - Stylistic representation of algorithms in code
- Layering
  - Layering in library mirrors layering in the math
- Mechanical transformation
  - Design-by-Transformation to mechanize the expert
- Exceptional performance
- Algorithm/architecture codesign
  - See talk by Ardavan Pedram

FLAME

- Derive algorithms to be correct
  - Extraordinary confidence in correctness
  - Choose from Families of algorithms
- **Avoid indexing**
  - **Stylistic representation of algorithms in code**
- Layering
  - Layering in library mirrors layering in the math
- Mechanical transformation
  - Design-by-Transformation to mechanize the expert
- Exceptional performance
- Algorithm/architecture codesign
  - See talk by Ardavan Pedram

Representing algorithms (Cholesky factorization)

Algorithm: $A := \text{CHOL\_BLK\_VAR3}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$$A_{11} = \Gamma(A_{11})$$
$$A_{21} = A_{21}\,\text{TRIL}\,(A_{11})^{-T}$$
$$A_{22} = A_{22} - \text{TRIL}\,(A_{21}A_{21}^T)$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

Gunnels et al. FLAME: Formal Linear Algebra Methods Environment. ACM *TOMS*. 2001.

```
#include "FLAME.h"

FLA_Error Chol_l_blk_var3( FLA_Obj A, int nb_alg )
{
  FLA_Obj ATL,   ATR,       A00, A01, A02,
          ABL,   ABR,       A10, A11, A12,
                            A20, A21, A22;
  int b;

  FLA_Part_2x2( A,     &ATL, &ATR,
                       &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_length( ATL ) < FLA_Obj_length( A ) ){

    b = min( FLA_Obj_length( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3(
          ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
       /* ************* */   /* ******************** */
                               &A10, /**/ &A11, &A12,
          ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                                b, b, FLA_BR );
    /*----------------------------------------------------*/
     Chol_l_unb_var3( A11 );

     FLA_Trsm( FLA_RIGHT, FLA_LOWER_TRIANGULAR,
               FLA_TRANSPOSE, FLA_NONUNIT_DIAG,
               FLA_ONE, A11, A21 );

     FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE,
             FLA_MINUS_ONE, A21, FLA_ONE, A22 );
    /*----------------------------------------------------*/

    FLA_Cont_with_3x3_to_2x2(
          &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                 A10, A11, /**/ A12,
       /* ************* */   /* ****************** */
          &ABL, /**/ &ABR,       A20, A21, /**/ A22, FLA_TL );
  }

  return FLA_SUCCESS;
}
```

Elemental (by Jack Poulson) Cholesky factorization

Algorithm: $A := \text{CHOL\_BLK\_VAR3}(A)$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

while $m(A_{TL}) < m(A)$ **do**

Determine block size b

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

where A_{11} is $b \times b$

$$A_{11} = \Gamma(A_{11})$$
$$A_{21} = A_{21}\,\text{TRIL}\,(A_{11})^{-T}$$
$$A_{22} = A_{22} - \text{TRIL}\,(A_{21}A_{21}^T)$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

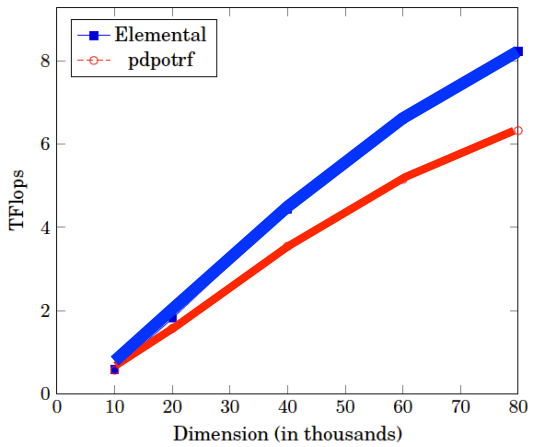
Jack Poulson et al. "Elemental: A New Framework for Distributed Memory Dense Matrix Computations." *TOMS*. 2013.

```
PartitionDownDiagonal
( A, ATL, ATR,
     ABL, ABR, 0 );
while( ABR.Height() > 0 )
{
    RepartitionDownDiagonal
    ( ATL, /**/ ATR,        A00,  /**/ A01, A02,
     /*************/ /****************/
           /**/              A10, /**/ A11, A12,
      ABL, /**/ ABR,         A20, /**/ A21, A22 );
    A21_VC_Star.AlignWith( A22 );
    A21_MC_Star.AlignWith( A22 );
    A21_MR_Star.AlignWith( A22 );
    //------------------------------------------------------//
    A11_Star_Star = A11;
    advanced::internal::LocalChol( Lower, A11_Star_Star );
    A11 = A11_Star_Star;

    A21_VC_Star = A21;
    basic::internal::LocalTrsm
    ( Right, Lower, ConjugateTranspose, NonUnit,
      (F)1, A11_Star_Star, A21_VC_Star );

    A21_MC_Star = A21_VC_Star;
    A21_MR_Star = A21_VC_Star;

    // (A21^T[* ,MC])^T A21^H[* ,MR]
    //      = A21[MC,* ] A21^H[* ,MR] = (A21 A21^H)[MC,MR]
    basic::internal::LocalTriangularRankK
    ( Lower, ConjugateTranspose,
      (F)-1, A21_MC_Star, A21_MR_Star, (F)1, A22 );

    A21 = A21_MC_Star;
    //------------------------------------------------------//
    A21_VC_Star.FreeAlignments();
    A21_MC_Star.FreeAlignments();
    A21_MR_Star.FreeAlignments();
    SlidePartitionDownDiagonal
    ( ATL, /**/ ATR,        A00, A01, /**/ A02,
                /**/                A10, A11, /**/ A12,
     /*************/ /*****************/
      ABL, /**/ ABR,        A20, A21, /**/ A22 );
}
```

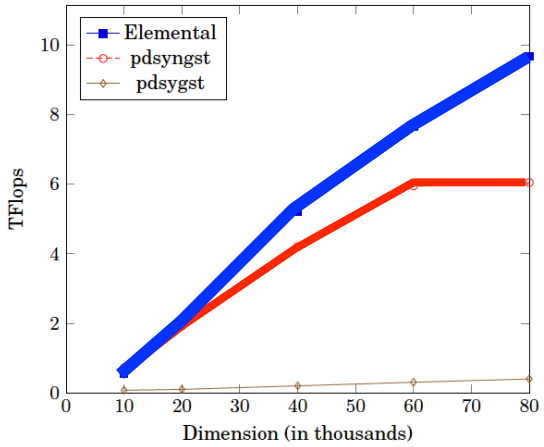
FLAME

- Derive algorithms to be correct
  - Extraordinary confidence in correctness
  - Choose from Families of algorithms
- Avoid indexing
  - Stylistic representation of algorithms in code
- Layering
  - Layering in library mirrors layering in the math
- Mechanical transformation
  - Design-by-Transformation to mechanize the expert
- **Exceptional performance**
- Algorithm/architecture codesign
  - See talk by Ardavan Pedram

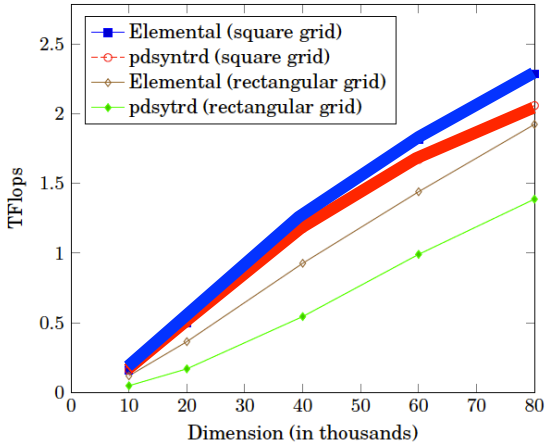
Performance of Elemental on 8192 cores (BlueGene/P)



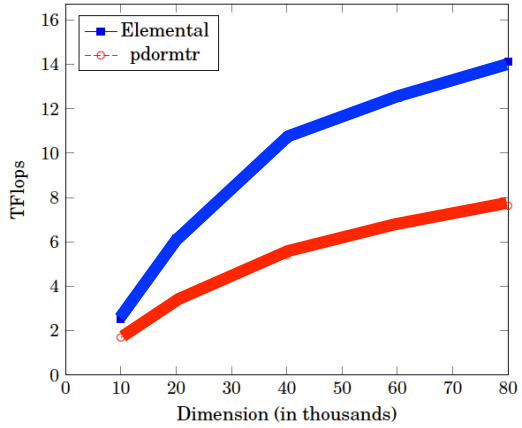
Cholesky



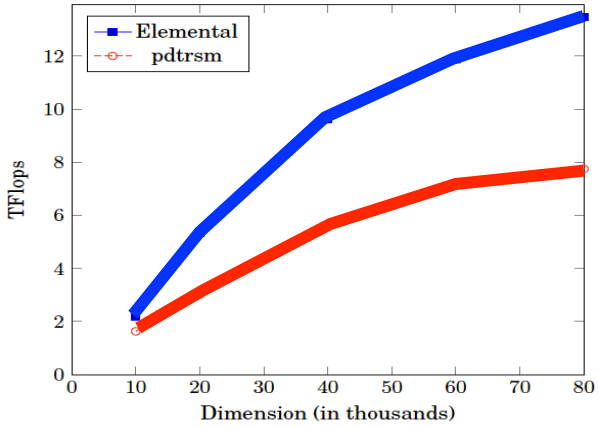
Two-sided triangular solve



Reduction to tridiag. form



Back transformation



TRSM

Elemental

ScaLAPACK

Poulson, Marker, Hammond, Romero, and van de Geijn. "Elemental: A New Framework for Distributed Memory Dense Matrix Computations." *TOMS*. 2013.

Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

Standing on the shoulders of giants

- Agarwal, Gustavson, Zubair. Exploiting functional parallelism on Power2 to design high-performance numerical algorithms. IBM J. of R. and D., 1994.
- Bilmes, Asanovic, Chin, Demmel. Optimizing Matrix Multiply using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology. ICS, 1997.
- Kagstrom, Ling, Van Loan. GEMM-based Level 3 BLAS High Performance Model Implementations and Performance Evaluation Benchmark. ACM TOMS, 1998.
- Whaley, Dongarra. Automatically Tuned Linear Algebra Software. SC98, 1998.
- Goto, van de Geijn. Anatomy of high-performance matrix multiplication. ACM TOMS, 2008.
- Many others.

What is BLIS?

- BLIS is a framework for
  - Quickly instantiating high-performance BLAS-like libraries
  - Building high-performance implementations for new operations
- User-level BLIS APIs are BLAS-like
  - BLAS compatibility layer (optional)

Van Zee, van de Geijn. BLIS: A Framework for Generating BLAS-like Libraries. FLAME Working Note #66. 2012.

Beyond the BLAS interface

- Generalized storage.
  - column-major storage,
  - row-major storage, and
  - general stride (tensors).
- ***Mixed*** storage formats.
  - Example: C = C + A B

    where A is column-stored, B is row-stored, and C has general stride.
- ***Mixed*** data types
  - Example: C = C + A B

    where A is single precision real and B is double precision complex.

Beyond the BLAS interface

- Complete support for complex valued operations.
- API for lower-level kernels
  - We want to be able to “break through” layers to optimize higher-level operations
- A framework for building libraries

  (rather than a library for end users)
- New operations not in the BLAS

Current BLIS functionality

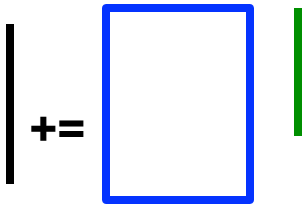
- Level 1v (roughly BLAS1 functionality)
- Level 1m (BLAS1 functionality with matrix operands)
- Level 1f (multiple vectors)
- Level 2 (roughly BLAS2 functionality)
- Level 3 (roughly BLAS3 functionality)

Overview

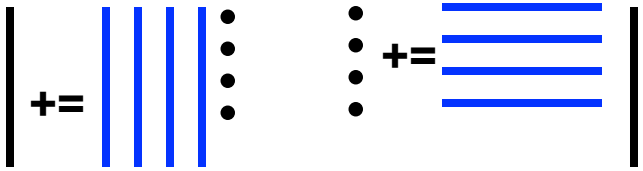
- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

Layering BLIS levels 1 and 2

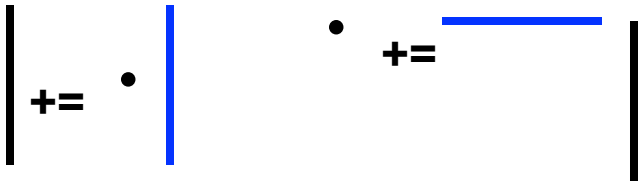
Level 2



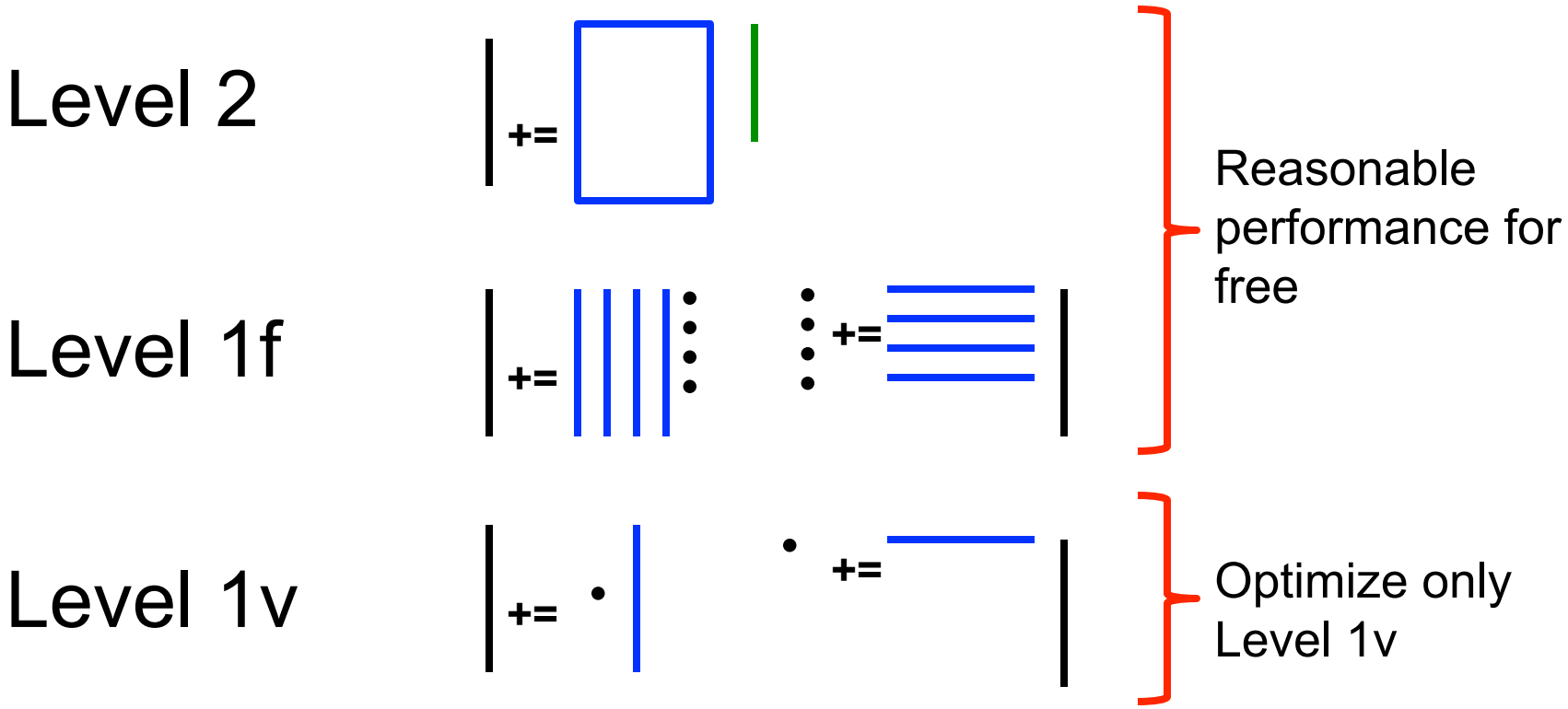
Level 1f



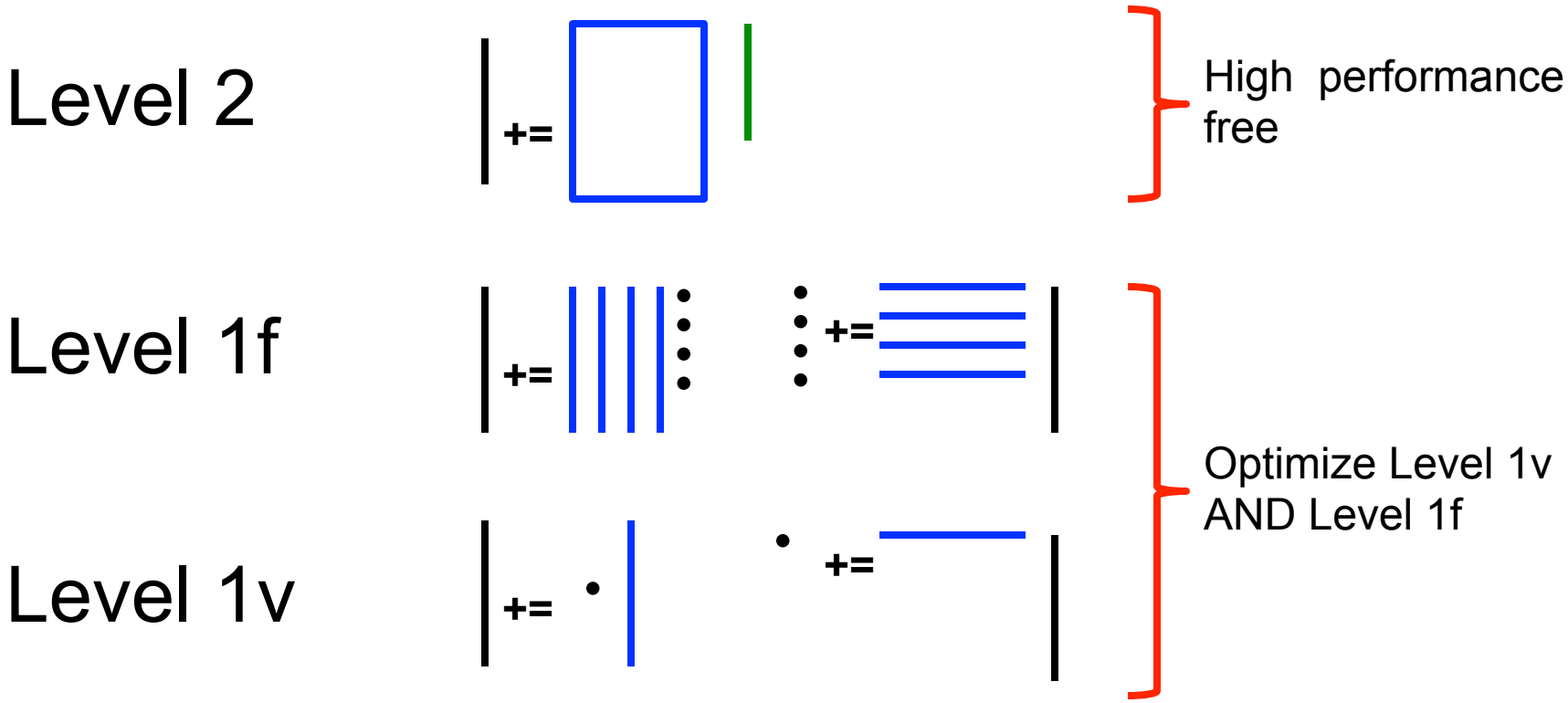
Level 1v



Layering BLIS levels 1 and 2



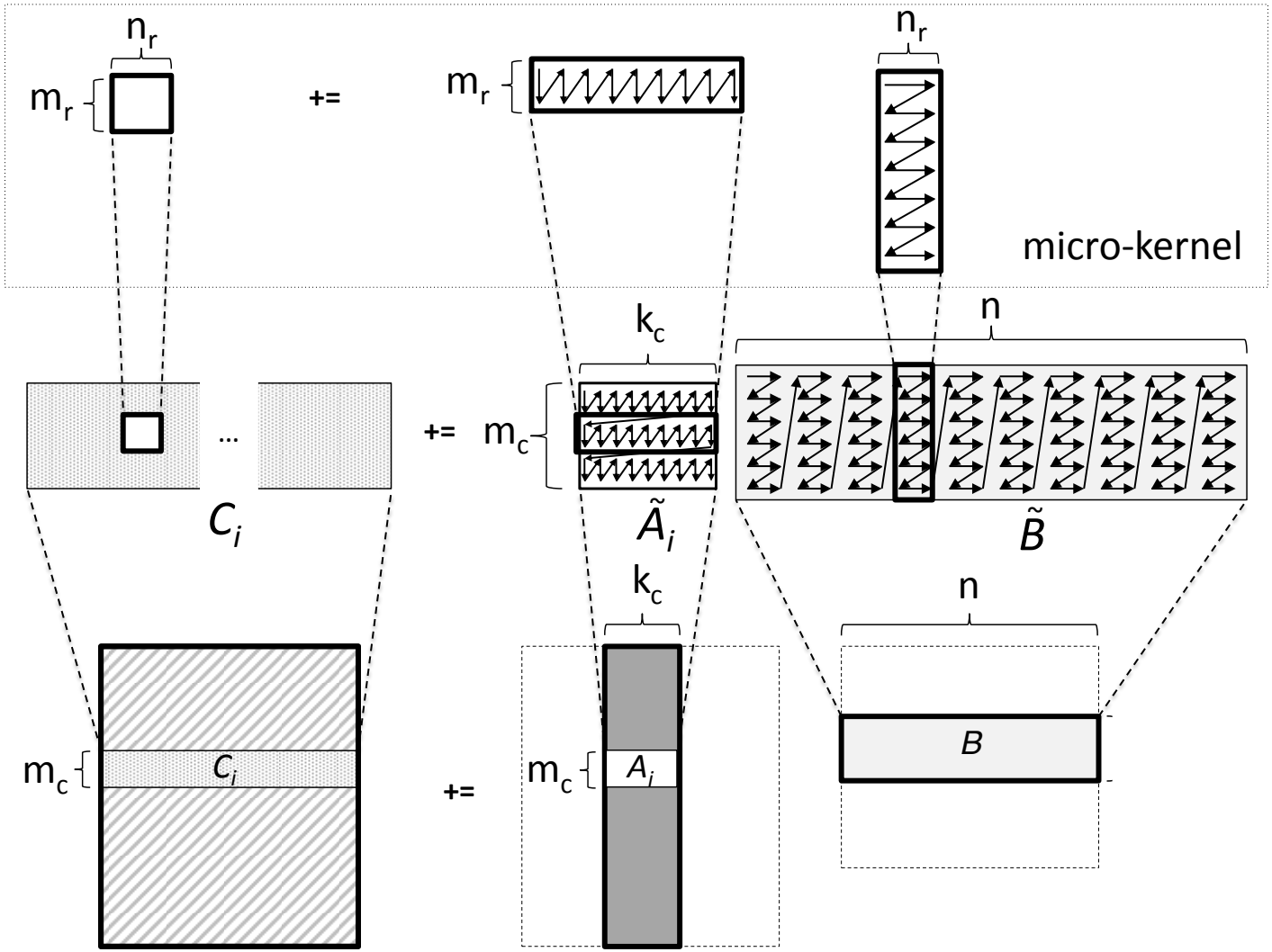
Layering BLIS levels 1 and 2



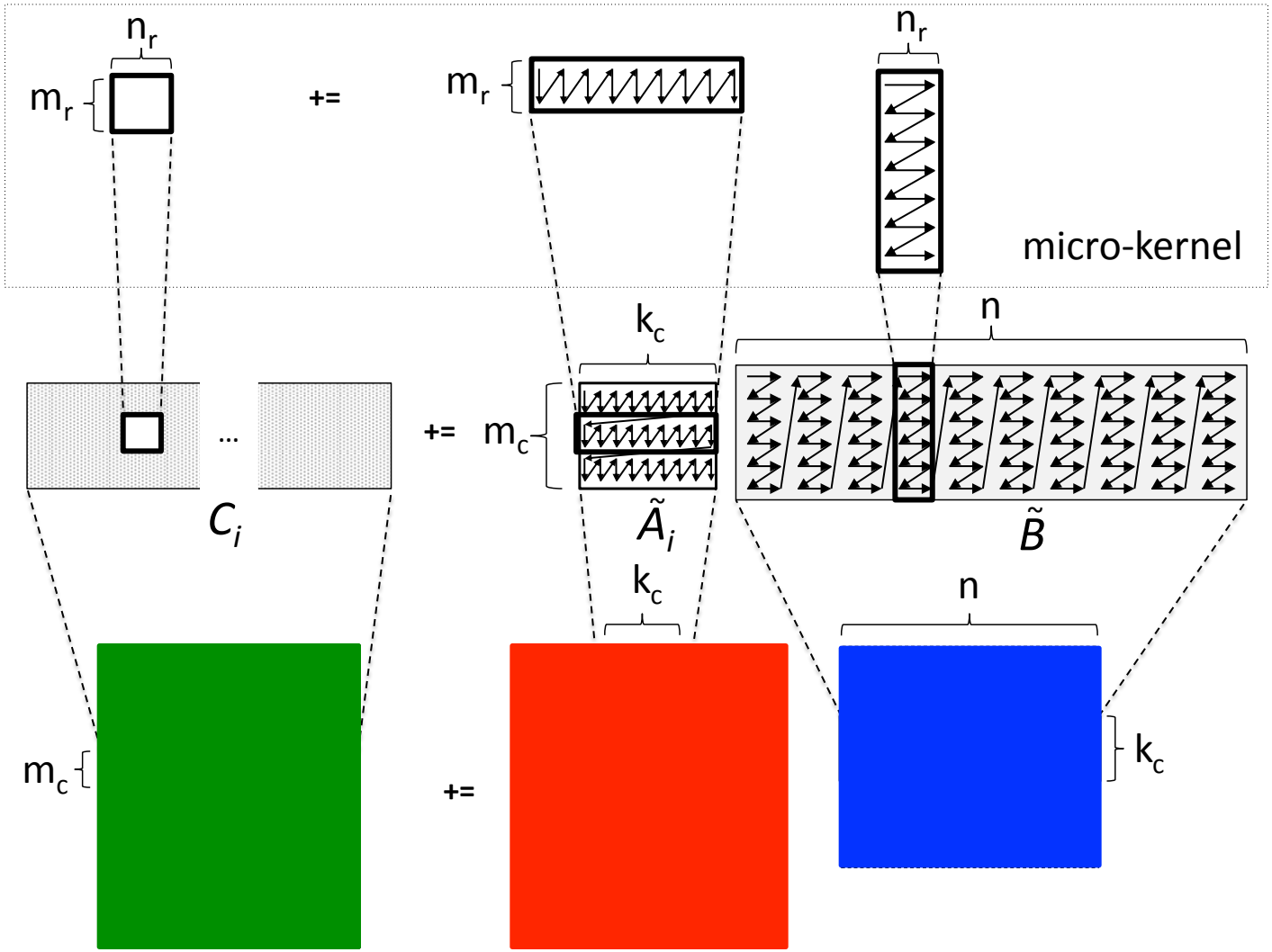
Layering BLIS level 3

- Even simpler!
- All level-3 operations are enabled with just
  - One gemm “micro-kernel”
    - where C is MR x NR ≈ 4 x 4 (and k is large)

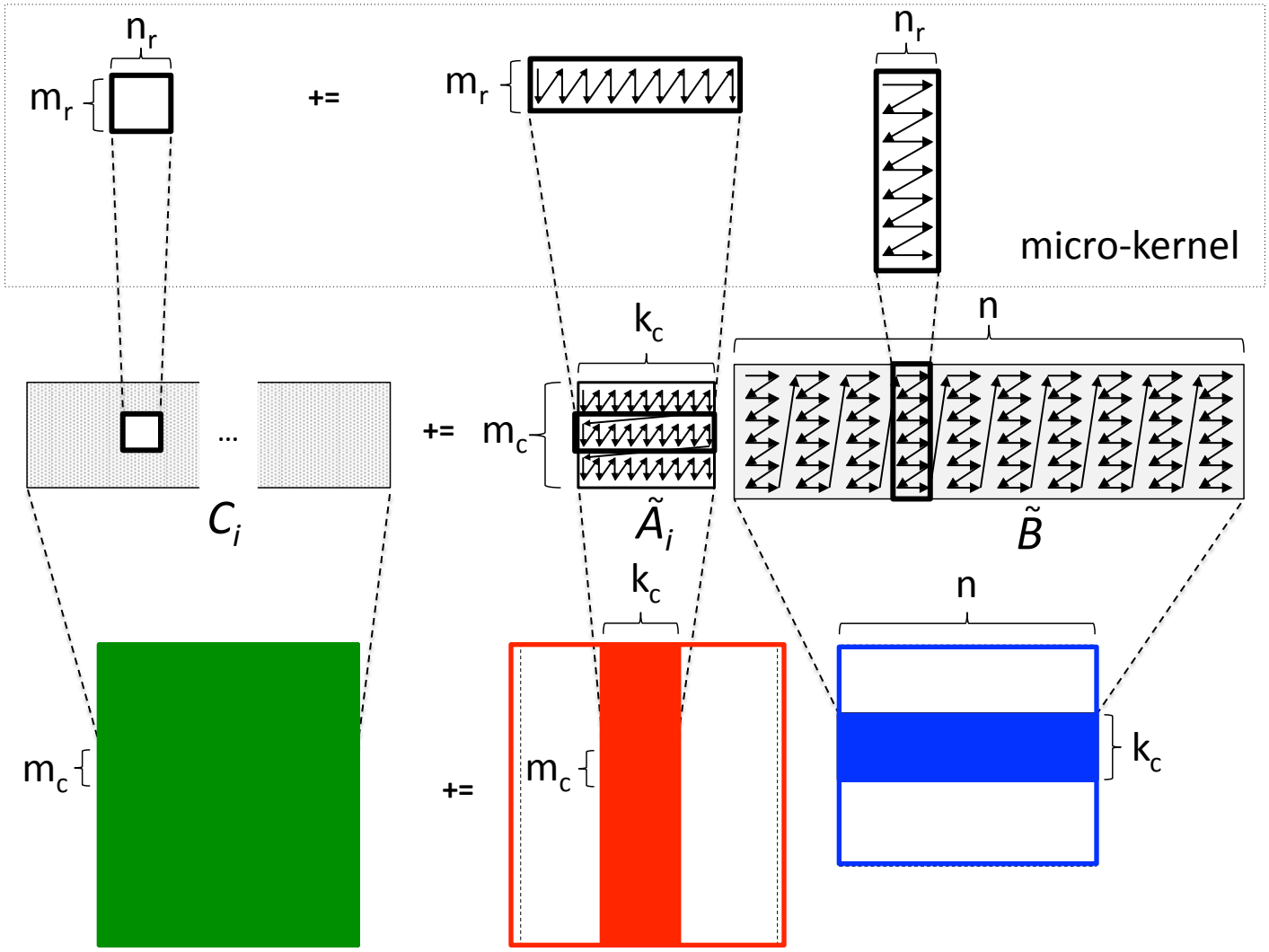
Layering matrix-matrix multiplication (gemm)



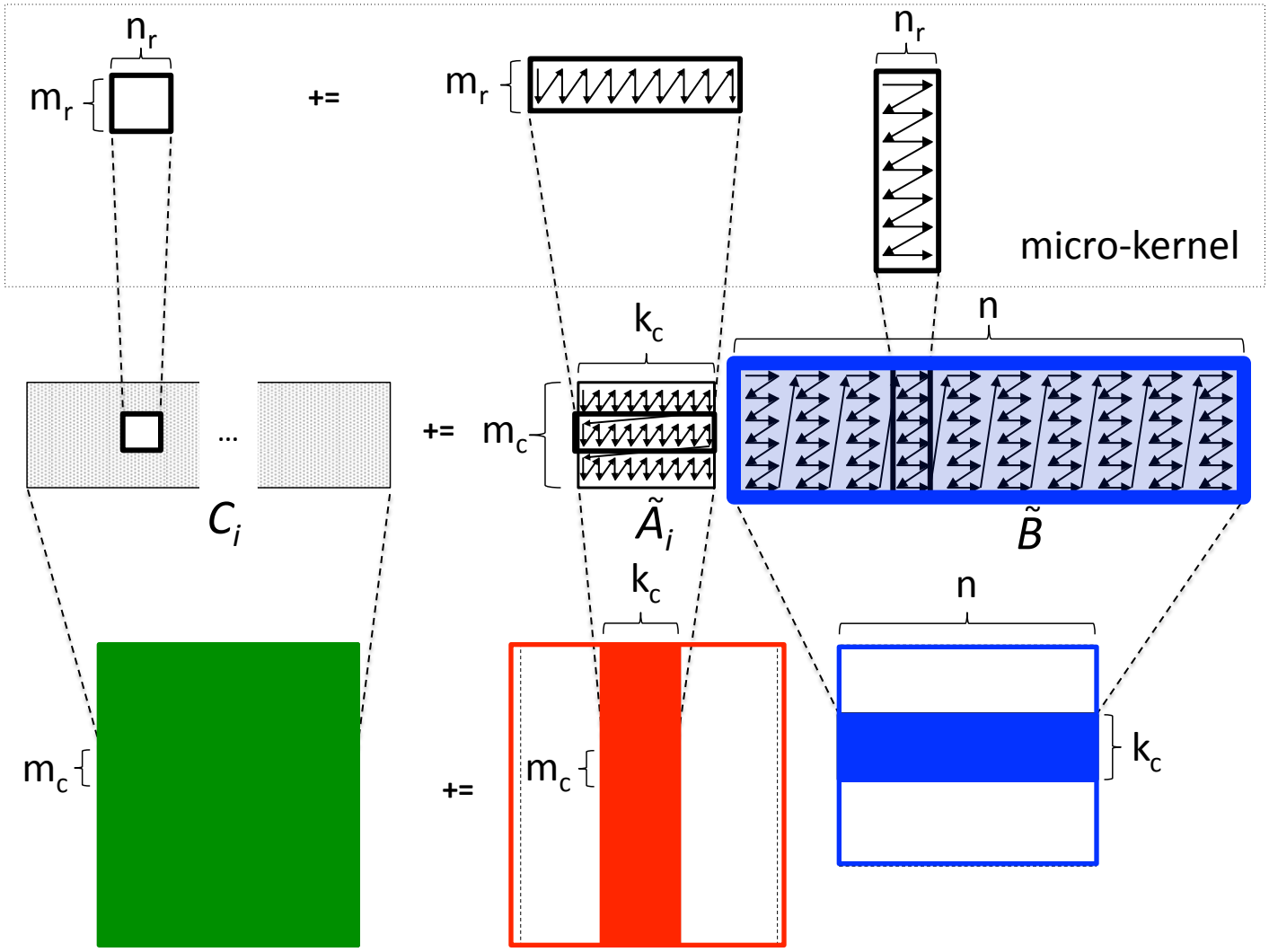
Layering matrix-matrix multiplication (gemm)



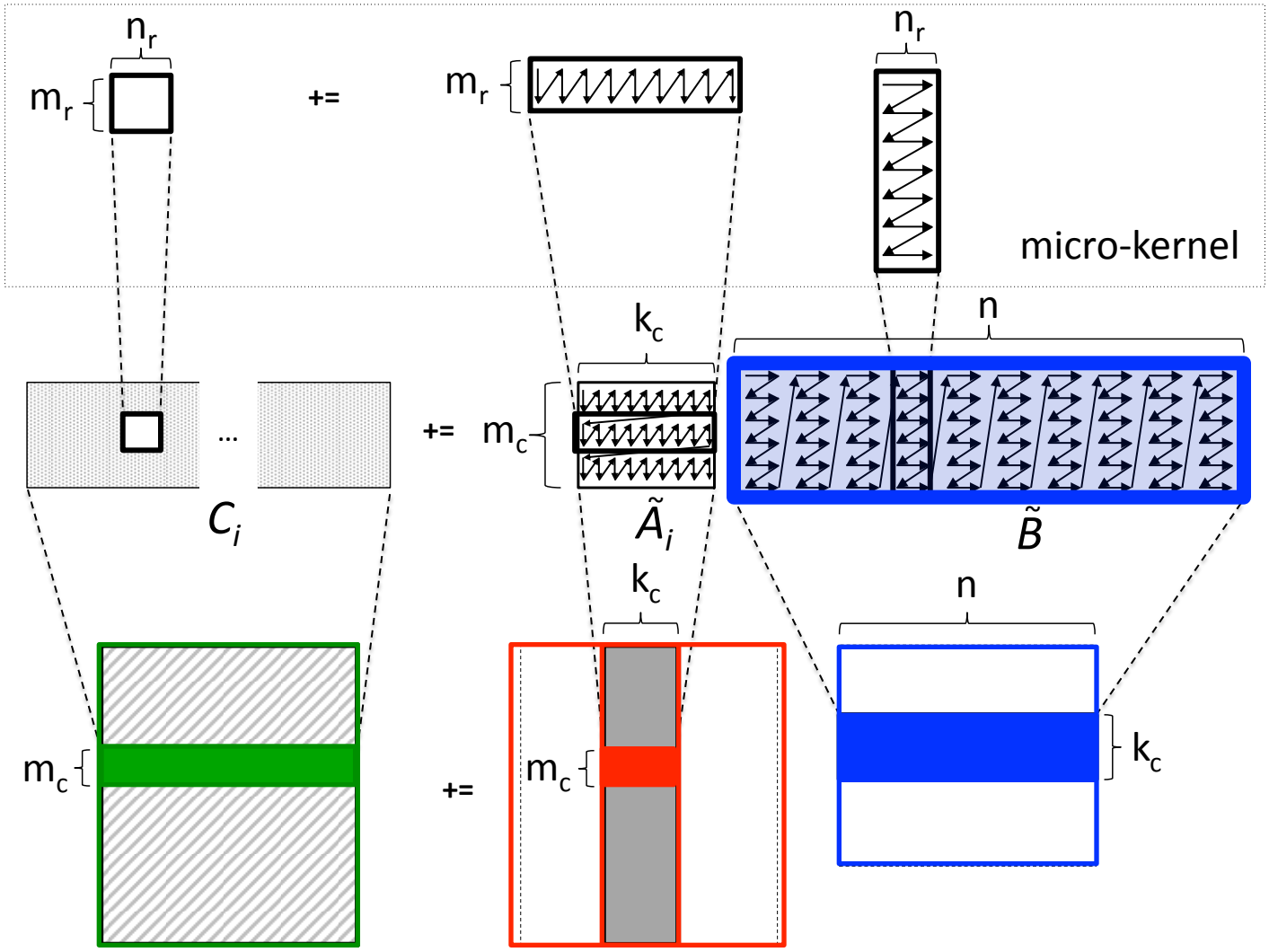
Layering matrix-matrix multiplication (gemm)



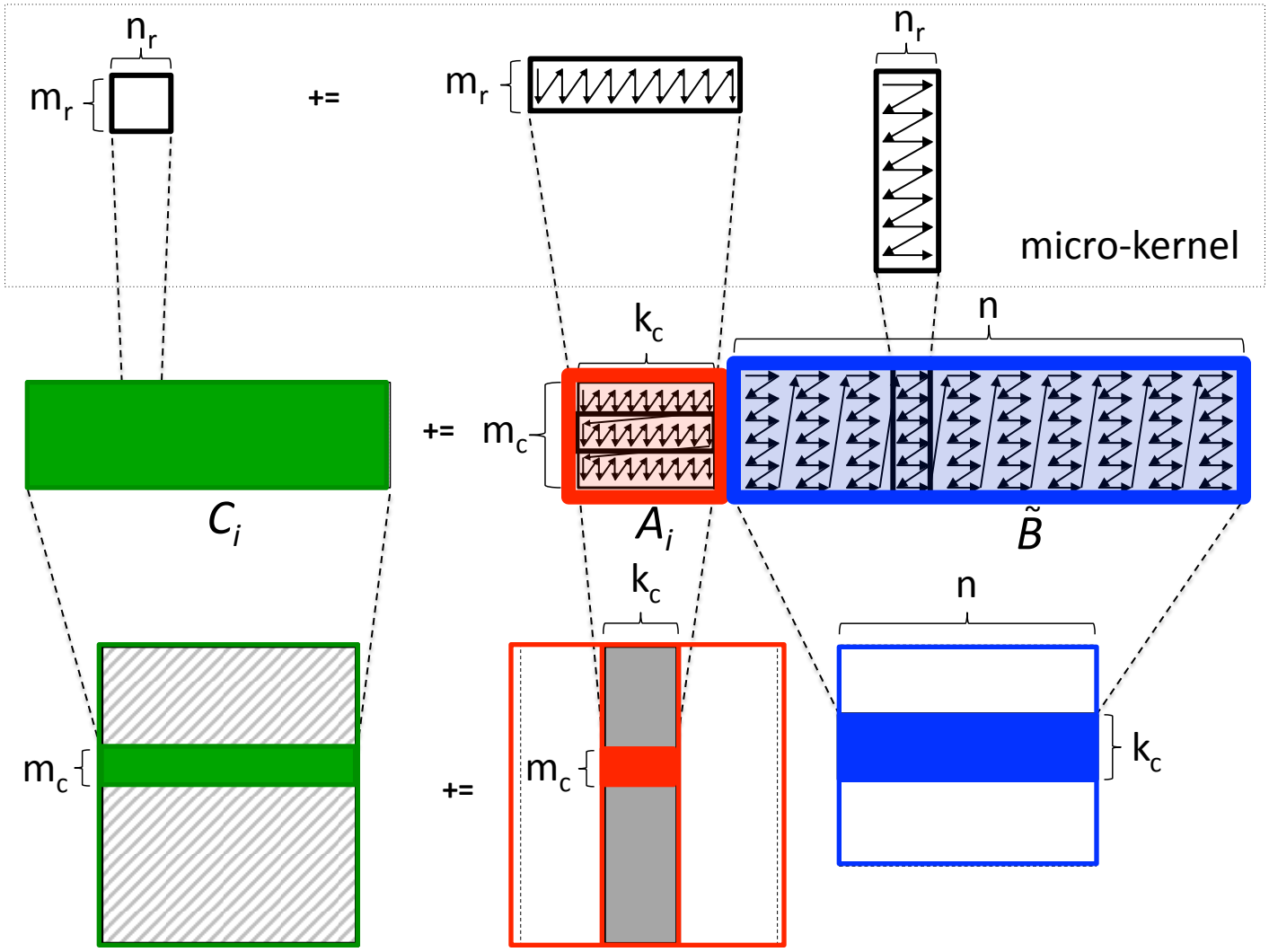
Layering matrix-matrix multiplication (gemm)



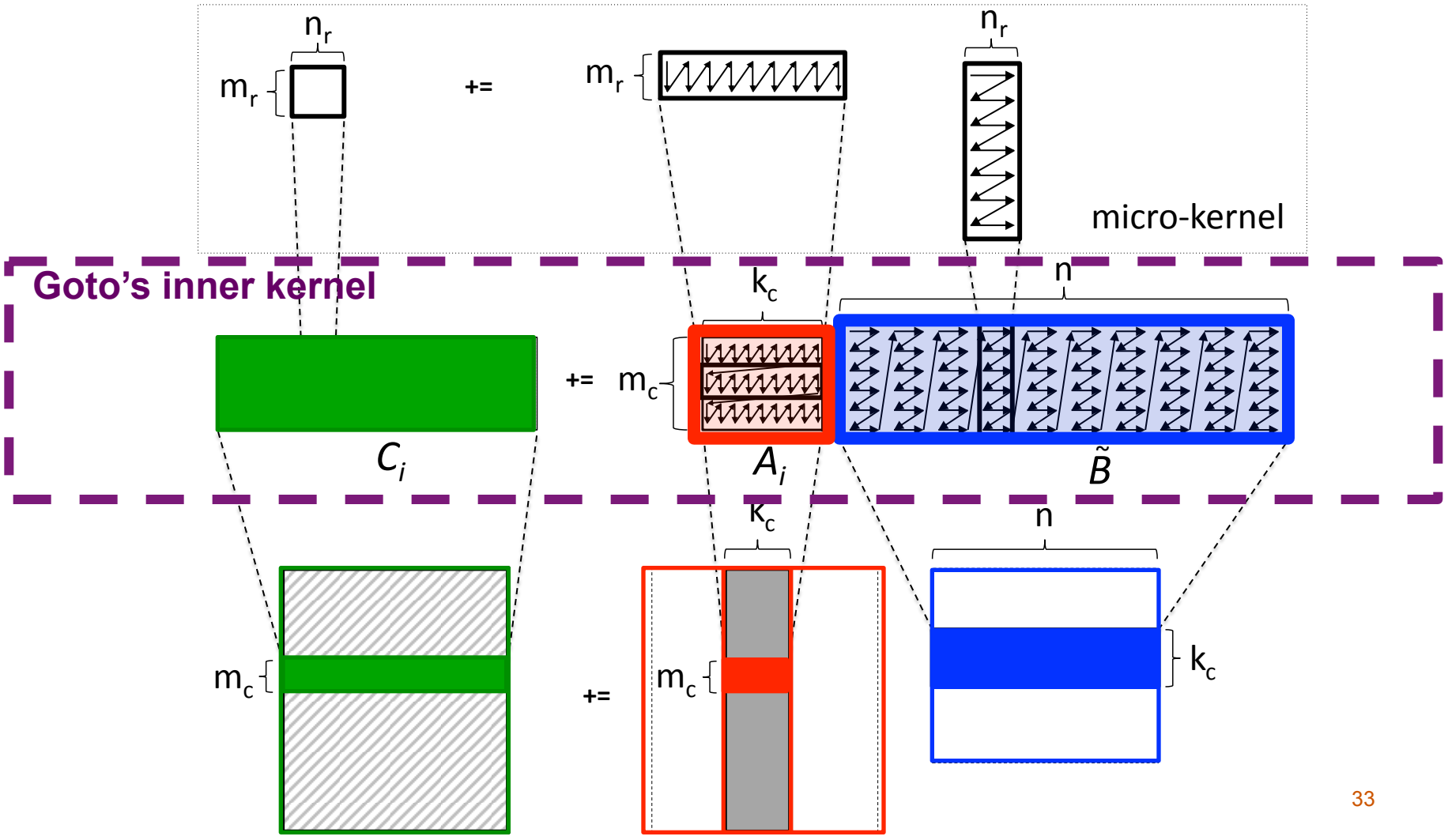
Layering matrix-matrix multiplication (gemm)



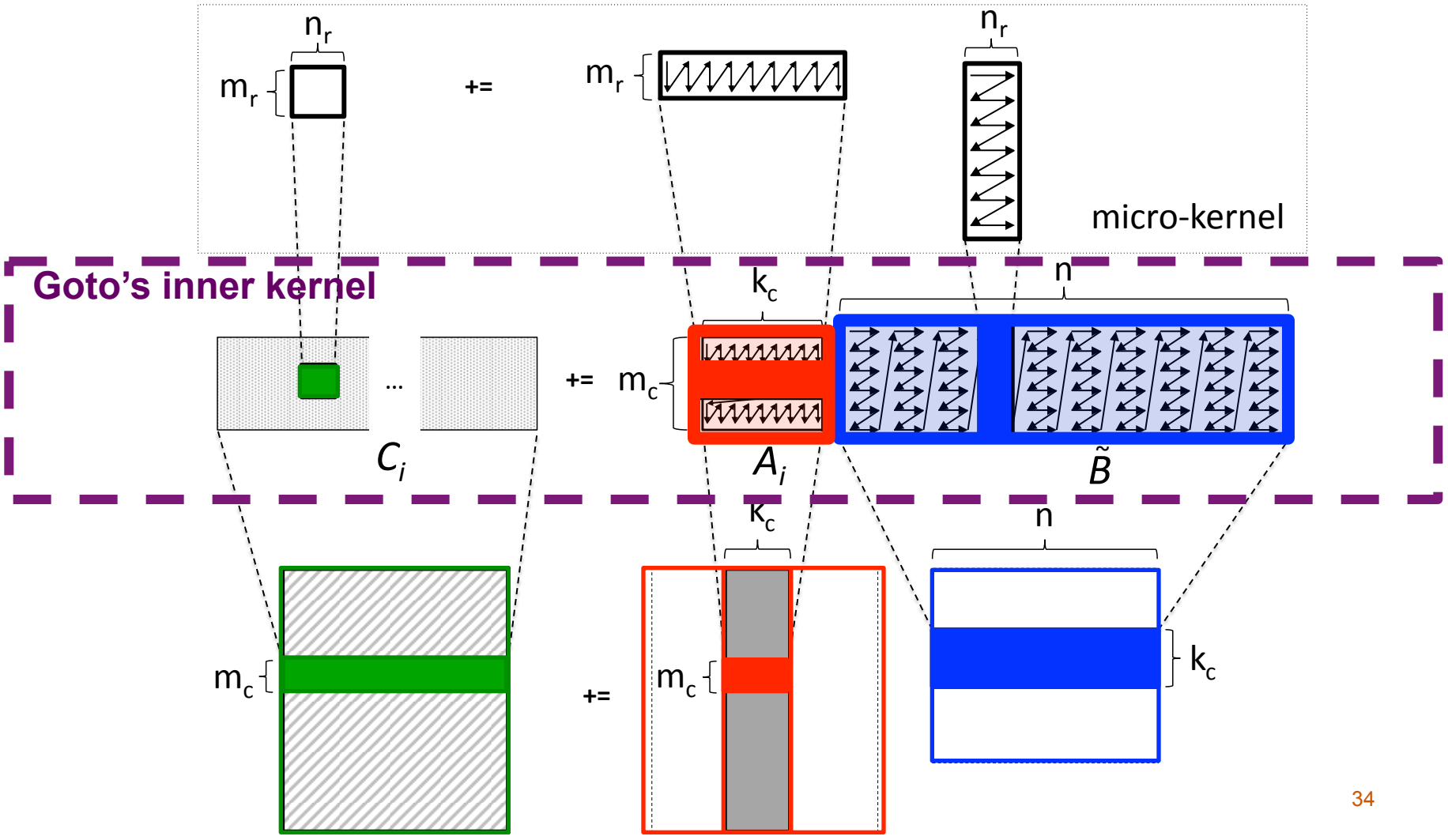
Layering matrix-matrix multiplication (gemm)



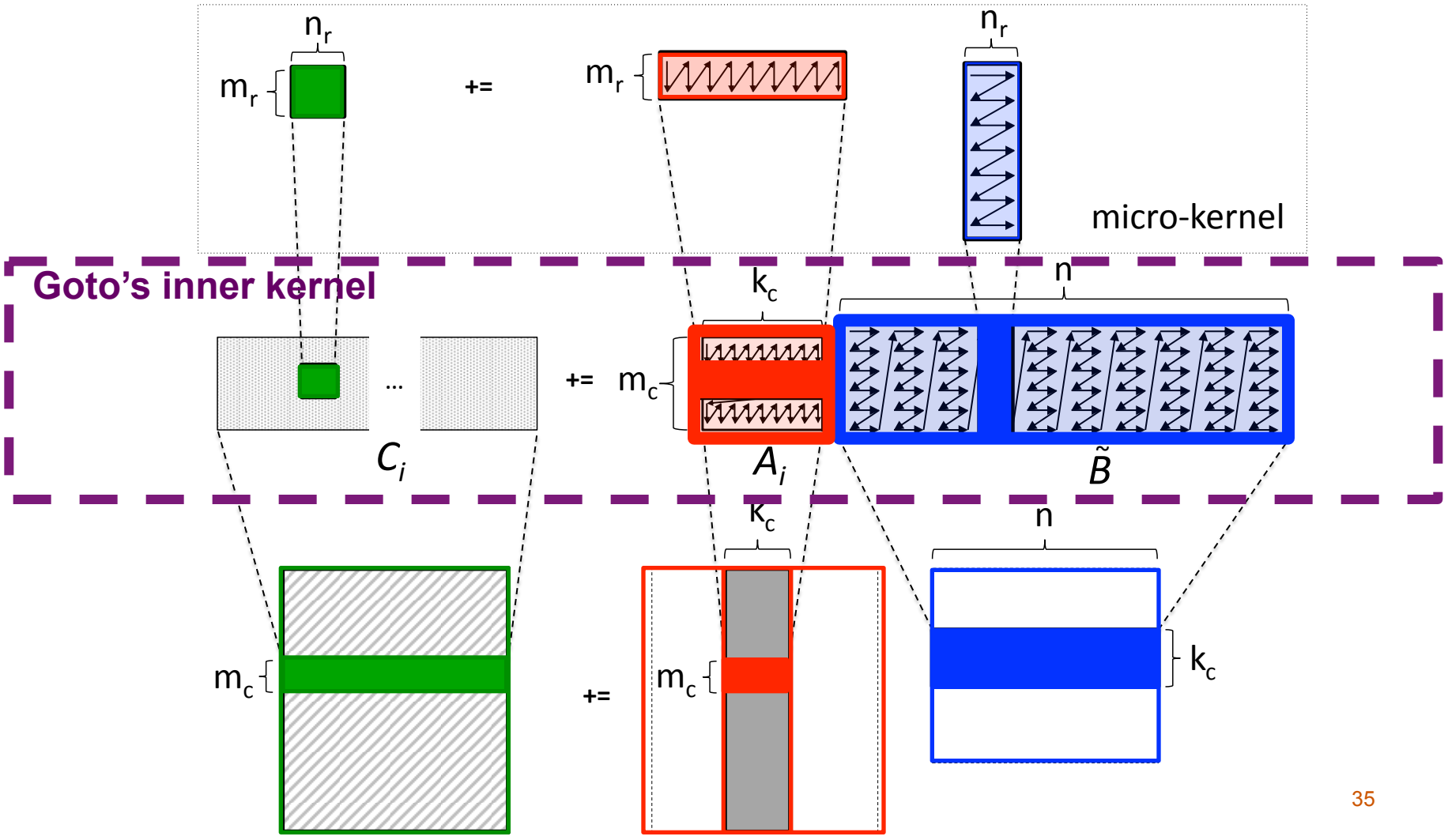
Layering matrix-matrix multiplication (gemm)



Layering matrix-matrix multiplication (gemm)



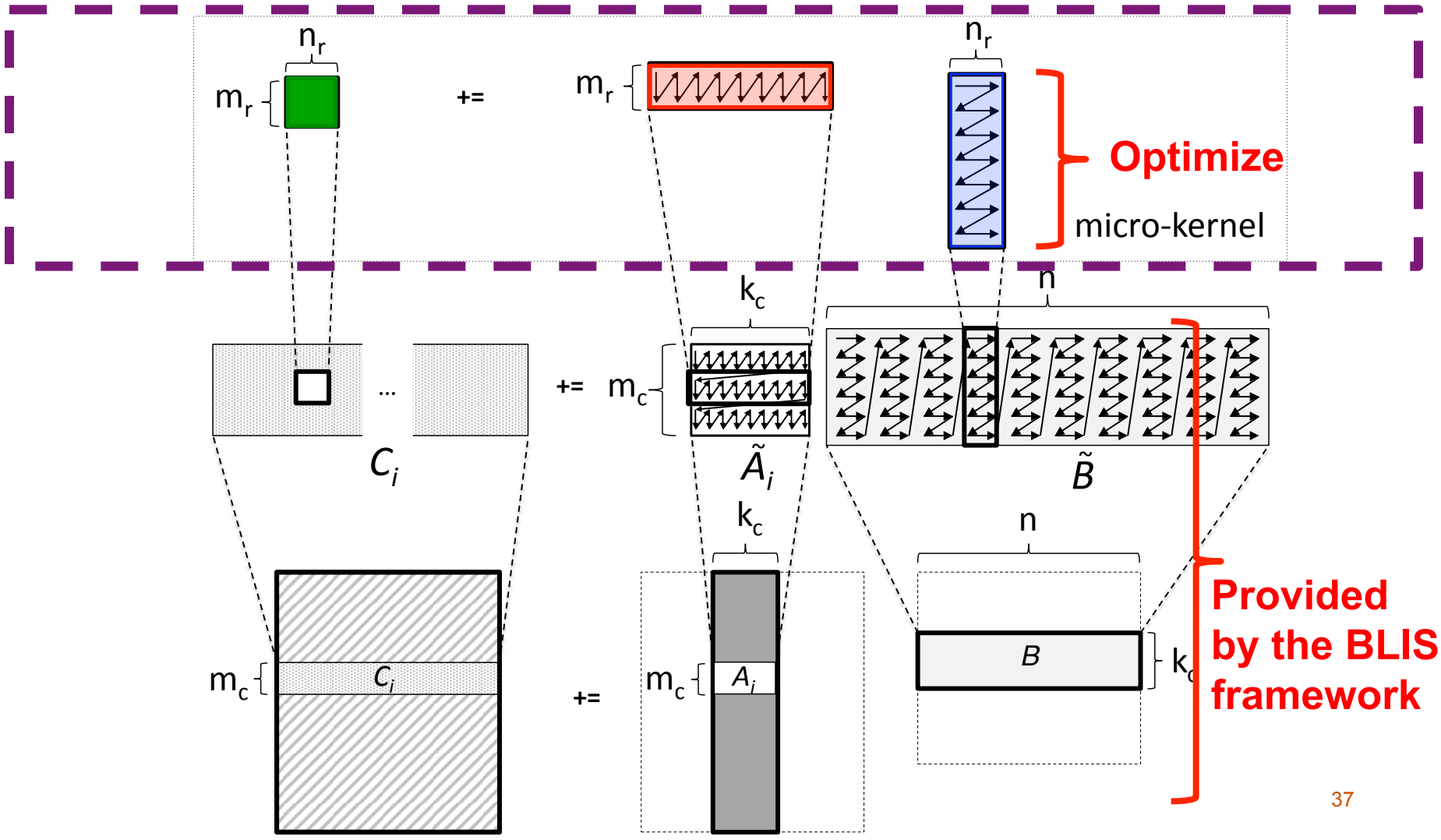
Layering matrix-matrix multiplication (gemm)



Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

Layering matrix-matrix multiplication (gemm)



Architectures targeted so far

- Intel
  - Dunnington
    - Field Van Zee (UT)
  - Sandy Bridge
    - Fran Igual (UCM-Spain)
  - Xeon Phi (MIC)
    - Tyler Smith (UT)
    - Misha Smelyanskiy (Intel)
- AMD
  - A10
    - Tyler Smith (UT)
- Texas Instruments
  - C6678 DSP
    - Fran Igual (UCM-Spain)
  - ARM Cortex A-9
    - Fran Igual (UCM-Spain)
- Loongson (MIPS)
  - 3A
    - Xianyi Zhang (Chinese Acad. Sc.)
- IBM
  - Power7
    - Mike Kistler (IBM)
  - BG/Q A2 PPC
    - John Gunnels/Vernon Austel (IBM)

Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

Libraries tested

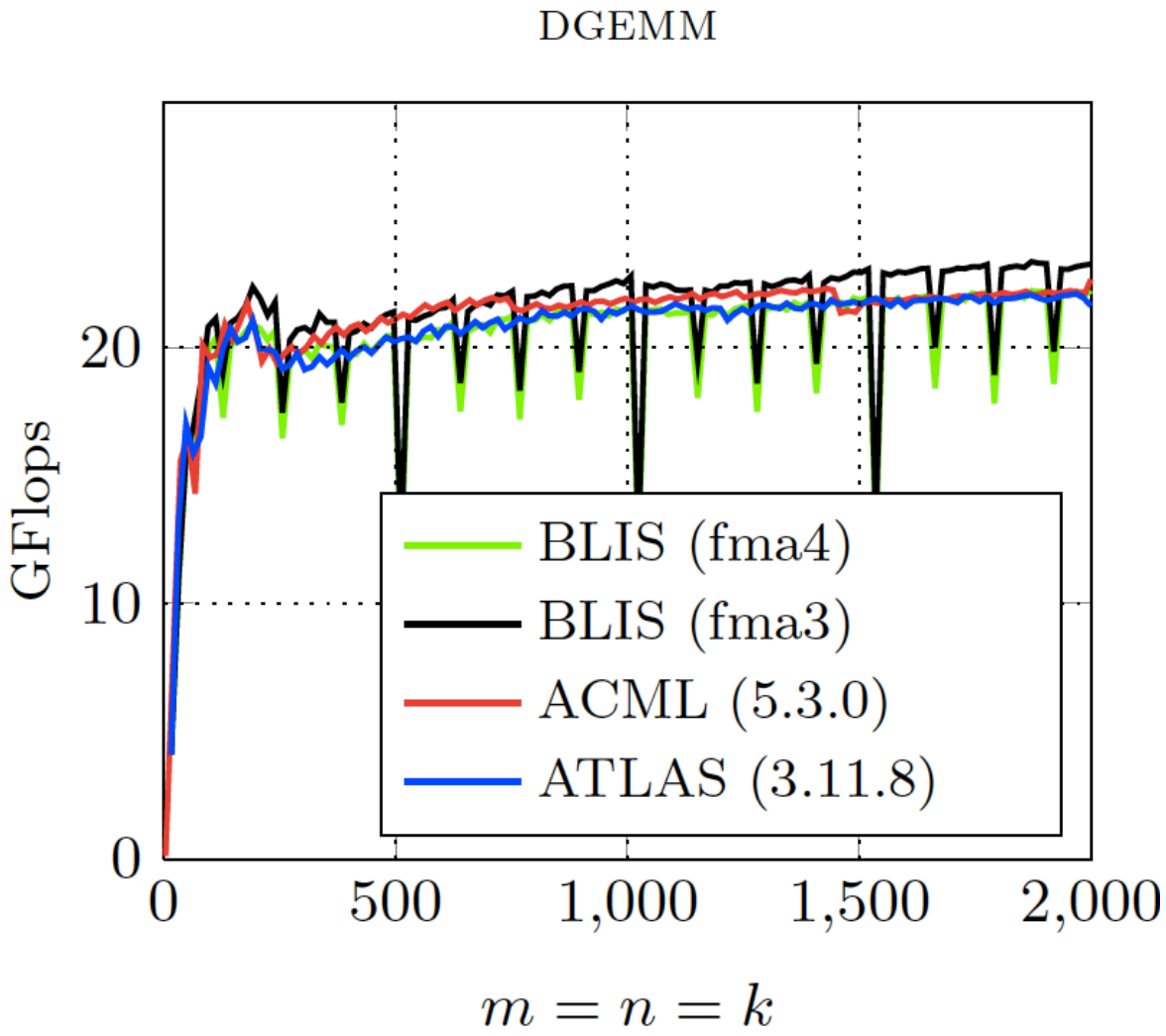
- AMD's ACML
- ATLAS

  [Whaley and Dongarra, 1998]
- BLIS
- Intel's MKL
- IBM ESSL
- OpenBLAS

  (GotoBLAS maintained by the Chinese Academy of Science)

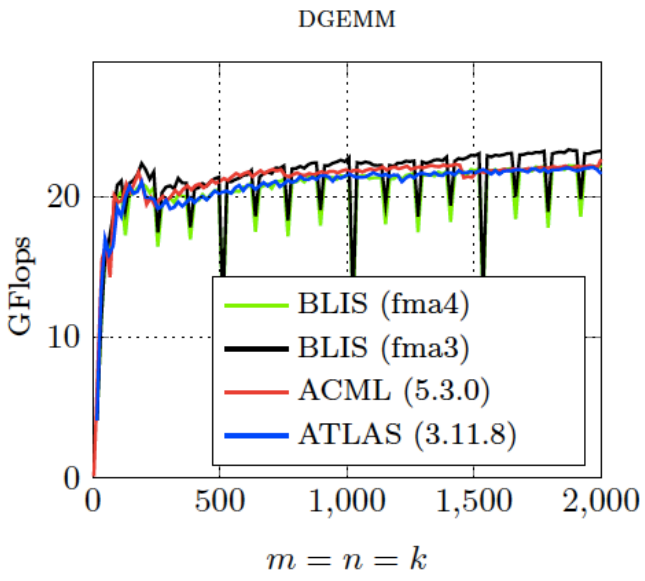
  [Zhang et al. 2012, Goto and van de Geijn, 2008]

AMD A10 processor (one core)



Van Zee, Igual, Smith, van de Geijn, Zhang, Kistler, Gunnels, Smelyanskiy. The BLIS Framework: Experiments in Portability. In preparation for SC13.

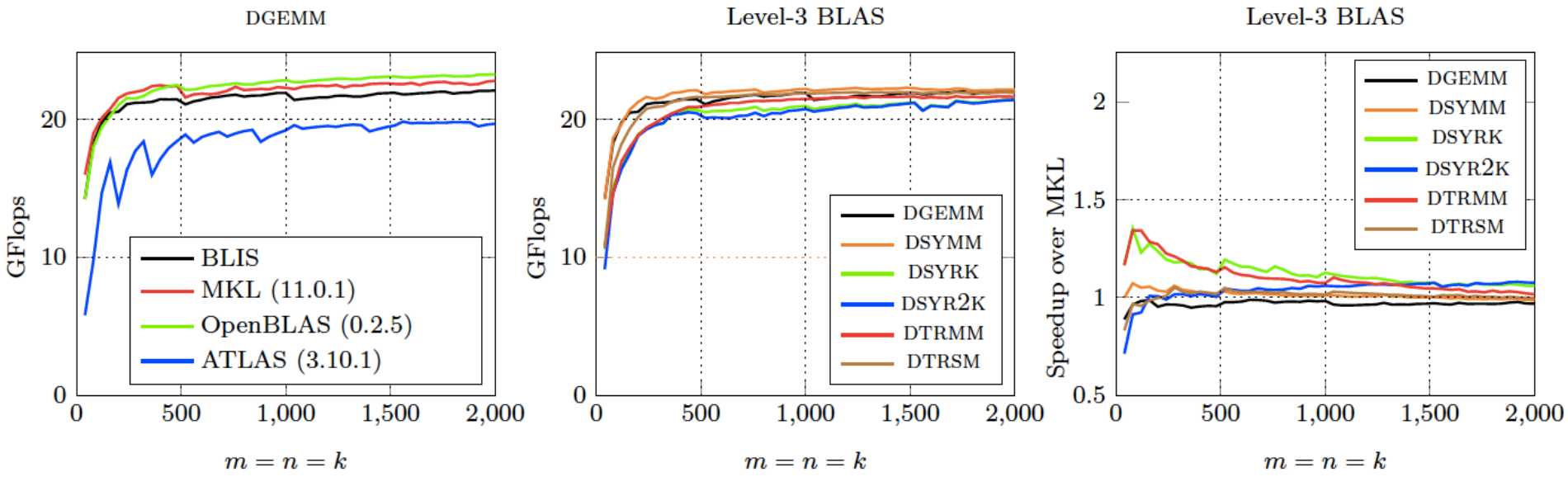
AMD A10 processor (one core)



For free!!!!

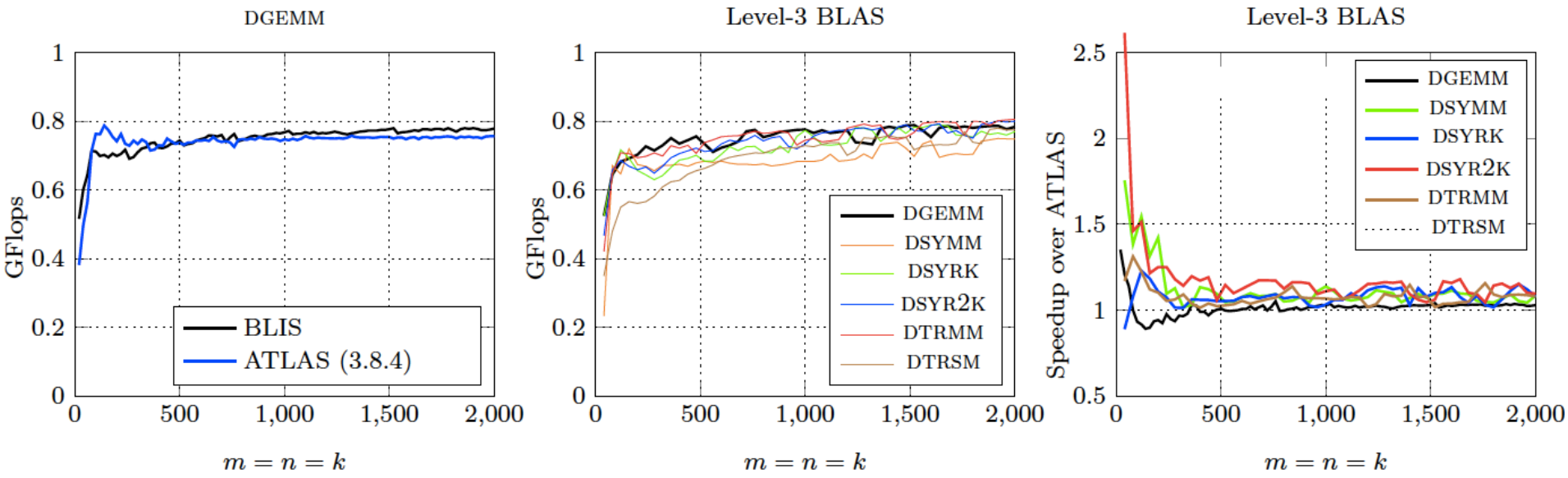
Van Zee, Igual, Smith, van de Geijn, Zhang, Kistler, Gunnels, Smelyanskiy. The BLIS Framework: Experiments in Portability. In preparation for SC13.

Intel Sandy Bridge (Xeon E3-1220) (one core)



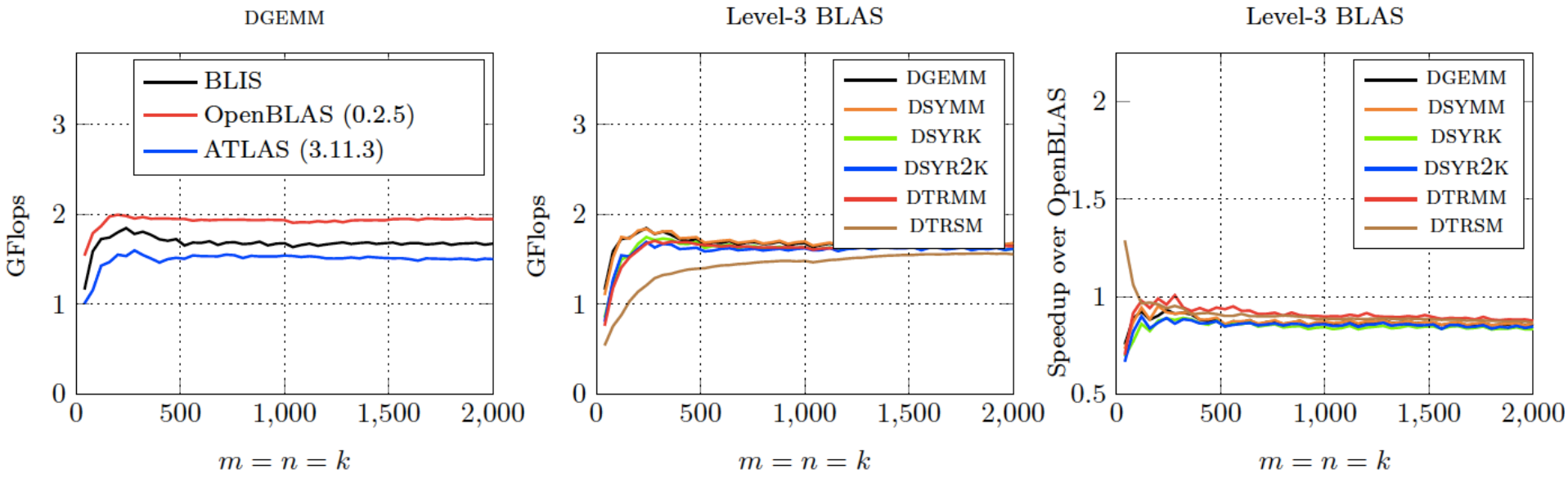
Van Zee, Igual, Smith, van de Geijn, Zhang, Kistler, Gunnels, Smelyanskiy. The BLIS Framework: Experiments in Portability. In preparation for SC13.

ARM Cortex A-9 (one core)



Van Zee, Igual, Smith, van de Geijn, Zhang, Kistler, Gunnels, Smelyanskiy. The BLIS Framework: Experiments in Portability. In preparation for SC13.

Loongson 3A - MIPS (one core)



Van Zee, Igual, Smith, van de Geijn, Zhang, Kistler, Gunnels, Smelyanskiy. The BLIS Framework: Experiments in Portability. In preparation for SC13.

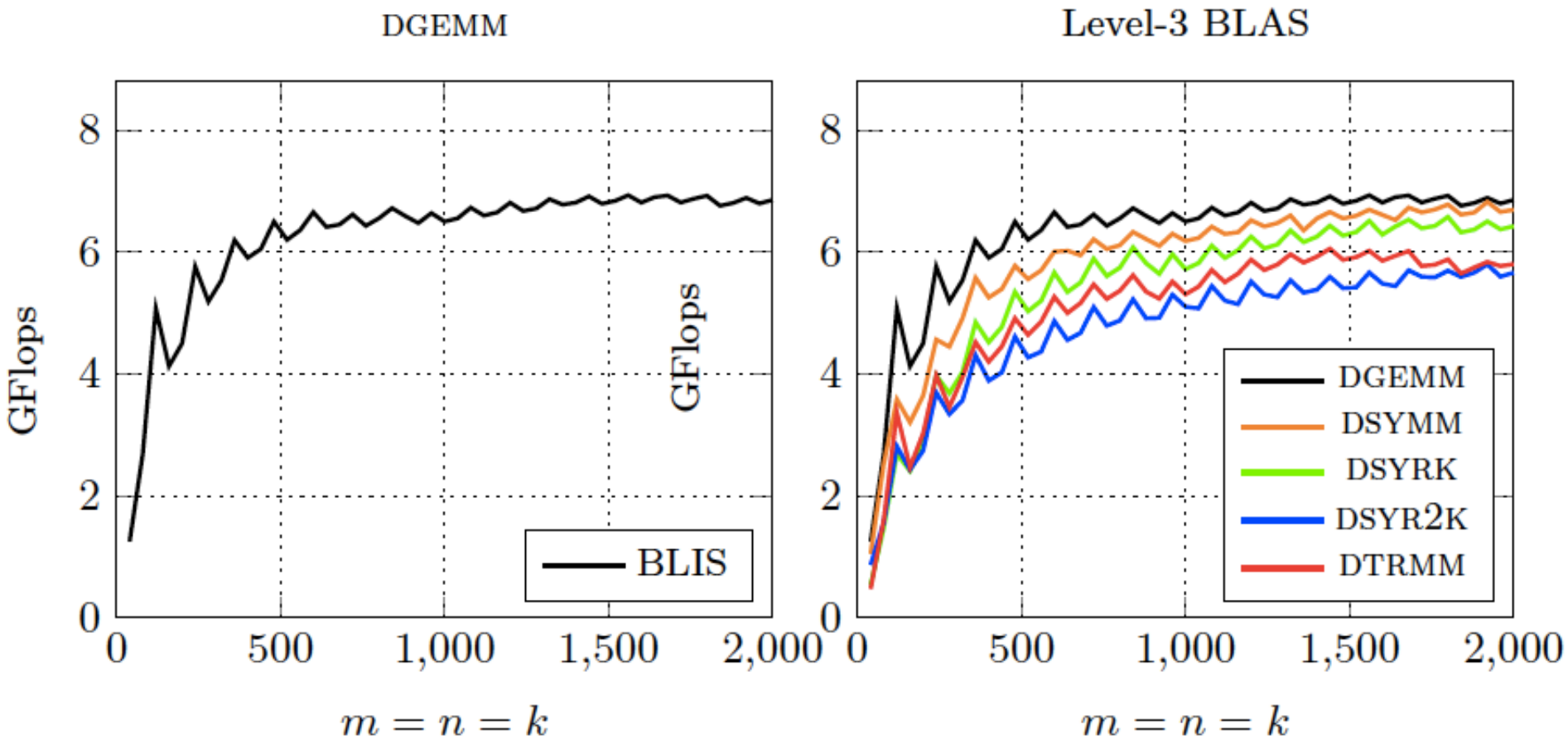
IBM Power7 (one core)

Performance graphs not yet approved for release

Van Zee, Igual, Smith, van de Geijn, Zhang, Kistler, Gunnels, Smelyanskiy.
The BLIS Framework: Experiments in Portability. In preparation for SC13.

Intel Xeon Phi (MIC) (one thread)

Important: to attain peak on one core, hyperthreading is required!!!!!!



Van Zee, Igual, Smith, van de Geijn, Zhang, Kistler, Gunnels, Smelyanskiy. The BLIS Framework: Experiments in Portability. In preparation for SC13.

Library footprint comparison

- BLIS libraries are relatively small

Driver with `dgemm()` linked to…	Library size**	Executable size
Nothing*	N/A	82K
OpenBLAS 0.1.1	3.00MB	107K
BLIS	1.06MB	180K
ATLAS 3.9.74	6.48MB	2.38M

* With `dgemm()` invocation removed.
** On an Intel Core2 system.

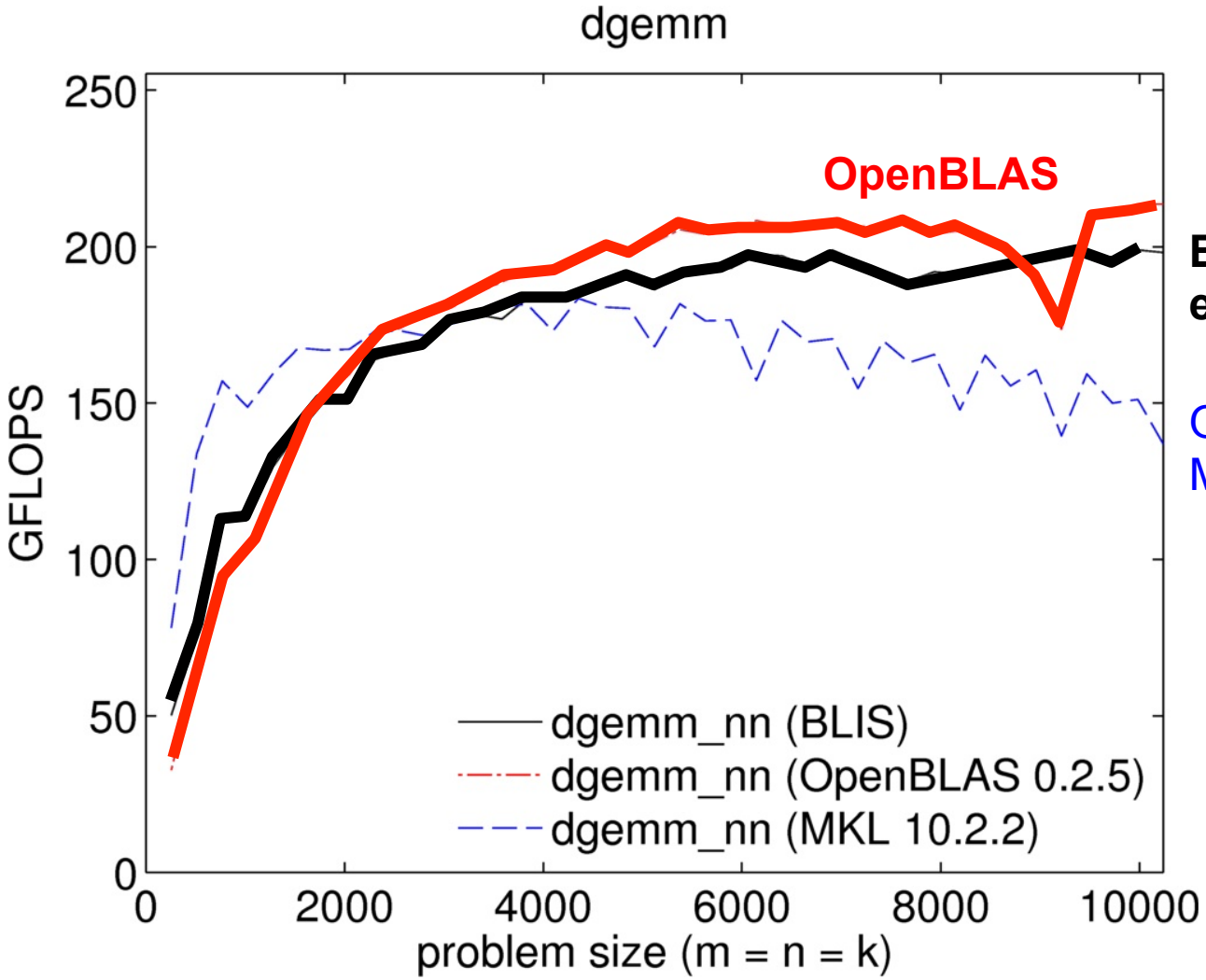
Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

What do we have planned?

- ~~Test suite~~ *Recently completed*
- ~~BLAS compatibility layer~~ *Recently completed*
- Complete set of kernels
- Multithreading
  - Early results are encouraging!

Intel Dunnington (24 cores)



BLIS early effort

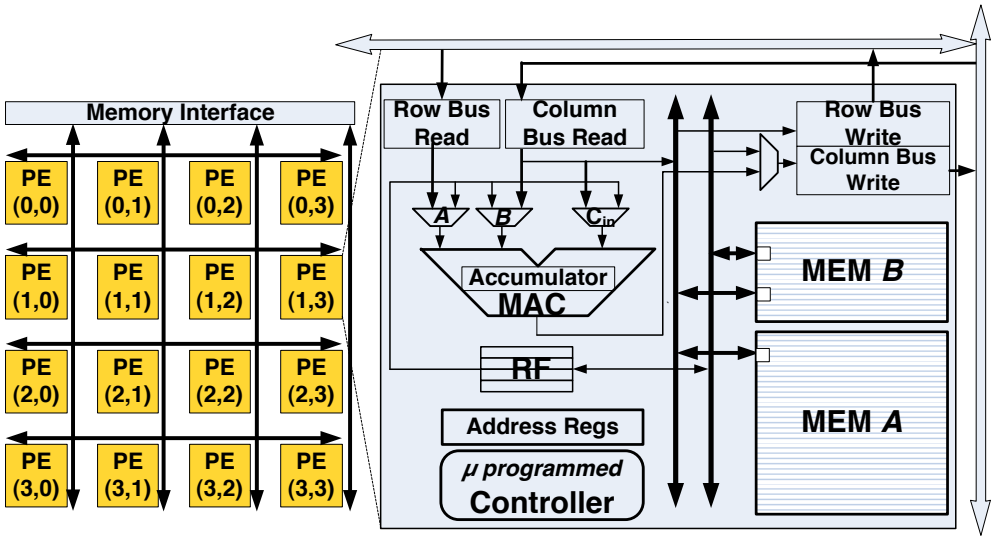
Old version of MKL

Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

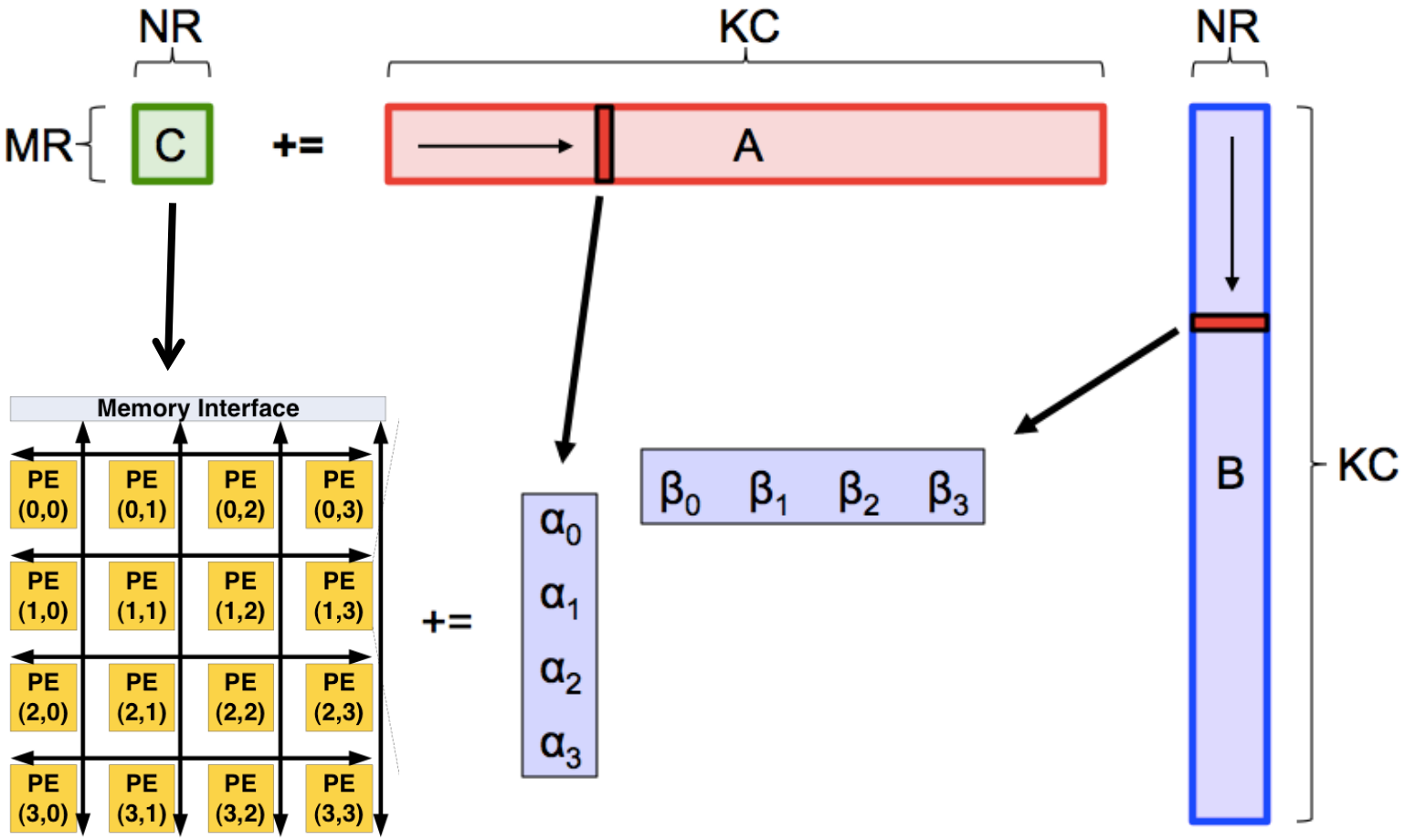
Why should you care? (as a ARITH 21 attendee)

- Linear Algebra Core algorithm/architecture codesign
  - Supports the microkernel in low power ASIC



Pedram, van de Geijn, Gerstlauer, Codesign Tradeoffs for High-Performance, Low Power Linear Algebra Architectures. IEEE Trans. on Computers. 2012.

Gemm micro-kernel on the LAC



Why should you care? (as a ARITH 21 attendee)

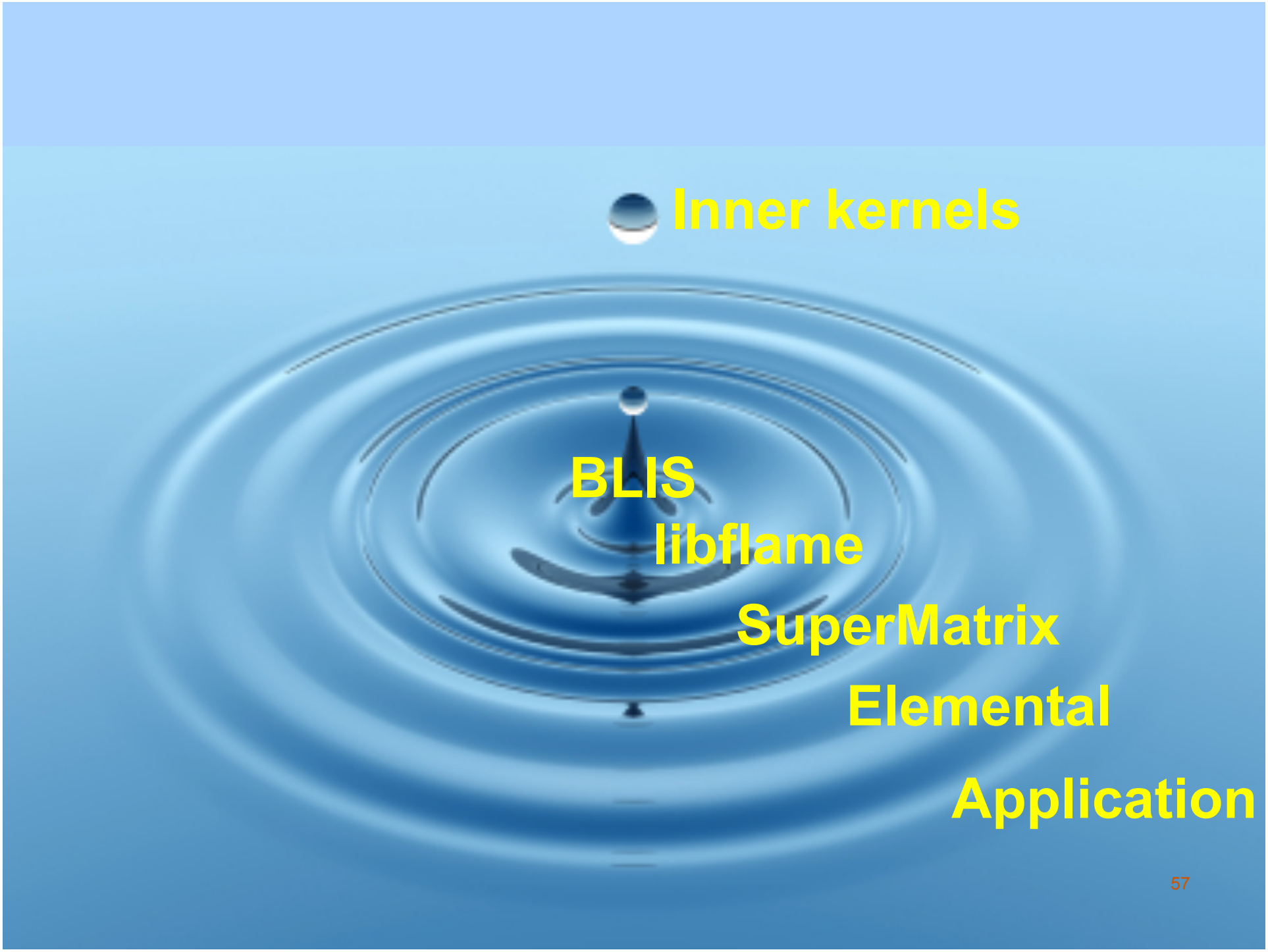
"… I can assess the performance of a candidate ukernel in BLIS in a matter of minutes, whereas in ATLAS a full retuning effort is required which can take hours. That alone is a huge boost in productivity."

– Mike Kistler, IBM-Austin

Similarly, autotuning is usually not an option when using a simulator. BLIS may be the solution.

Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion


Inner kernels
BLIS
libflame
SuperMatrix
Elemental
Application

Overview

- The FLAME Project
- A look at the bottom of the DLA software stack
  - What is BLIS
  - Layering in BLIS
  - The most basic building block: the micro-kernel
  - Performance
  - What do we have planned?
  - Why should you care?
- Managing the DLA software stack
- Conclusion

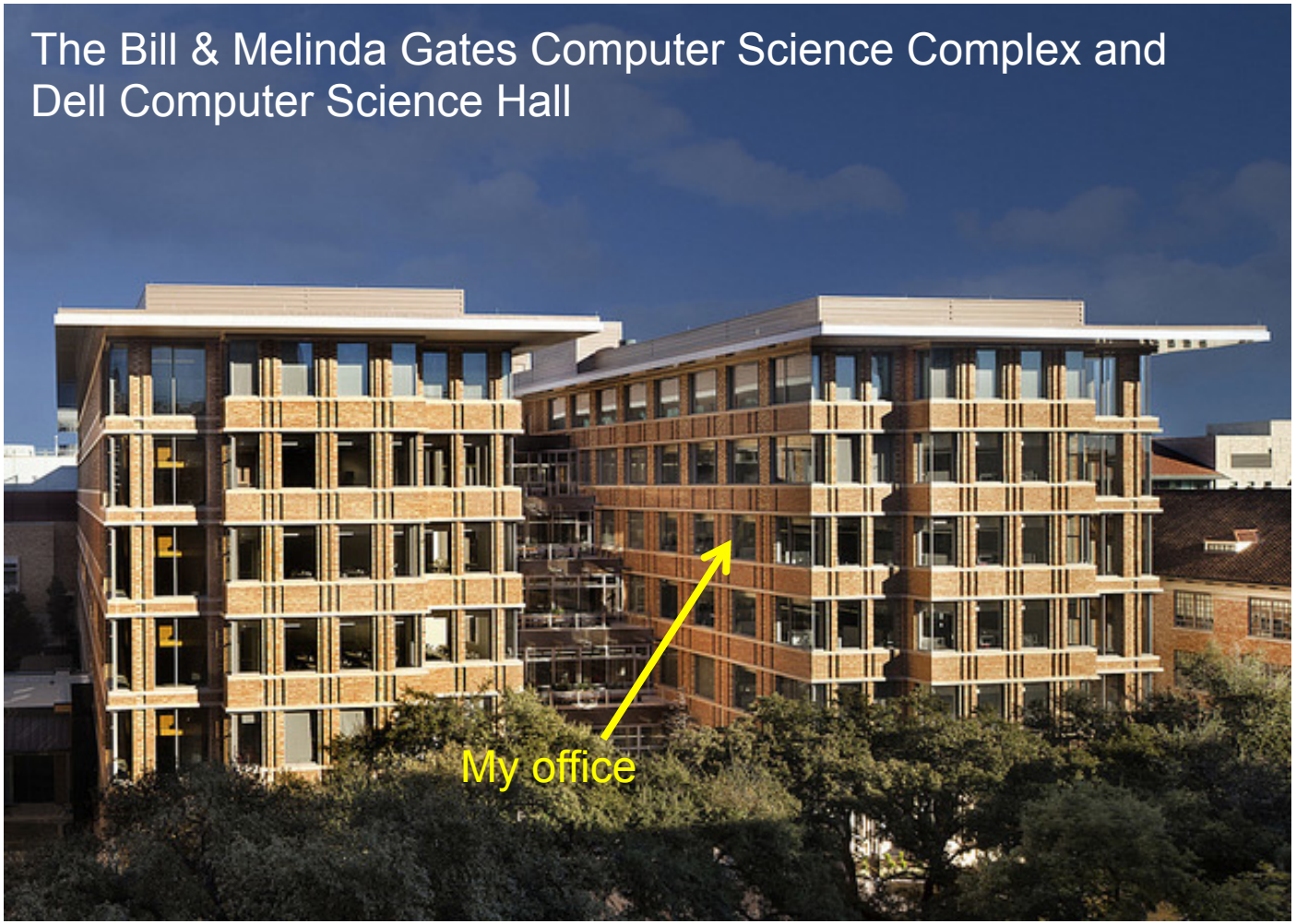
Conclusion

- The BLAS, LAPACK, and related libraries have served us well for the last 30 years.
- Time appears ripe for a new DLA software stack.
- BLIS appears to have exposed the smallest units upon which to build: the micro-kernels.
- The simplicity of BLIS enables algorithm/ architecture codesign.
- We believe we can now automate the DLA expert. (but that is another story…)

Further information

- Websites:
  - FLAME: http://www.cs.utexas.edu/~flame
  - BLIS: http://code.google.com/p/blis/
  - Elemental: http://code.google.com/p/elemental/
- Contact:
  - rvdg@cs.utexas.edu

Or visit me!


The Bill & Melinda Gates Computer Science Complex and
Dell Computer Science Hall
My office