

Packing - the next BLIS Frontier?

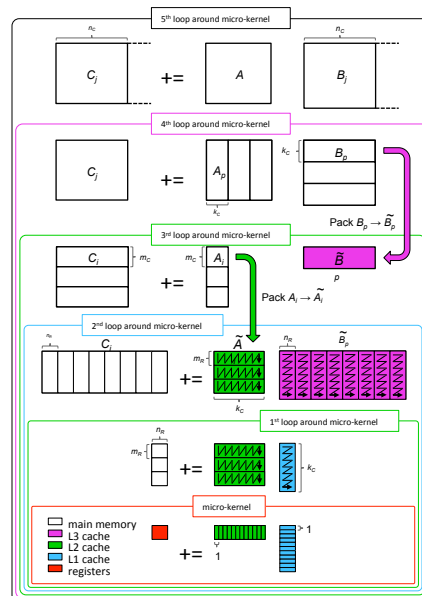
Tze Meng Low

BLIS Retreat 2017

Looking back

- Original Goals
 - The BLAS-like Library Instantiation Software (BLIS) framework is a new infrastructure for rapidly instantiating Basic Linear Algebra Subprograms (BLAS) functionality.
- Increase Productivity
- Extensible

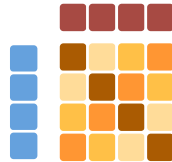
Etc...



F.G. Van Zee, R. van de Geijn, BLIS: A Framework for Rapidly Instantiating BLAS Functionality, ACM TOMS 2015

Ease of use

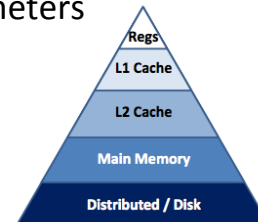
- What happens with a new architecture?
 - Write/Generate micro-kernel



- Use analytical modeling for parameters

$$C_{Ar} \leq \left\lfloor \frac{m_r}{m_r + n_r} (W_{L1} - 1) \right\rfloor$$

$$k_c = \frac{C_{Ar} N_{L1} C_{L1}}{m_r S_{data}}$$



T.M. Low, F. Igual, T. M. Smith, E. Quintana-Ortis, Analytical Modeling is Sufficient for High Performance BLIS, ACM TOMS, 2017

Ease of use

- Actual work
 - Create directory (recursive copy and rename)
 - Edit `bli_kernel.h`
 - Drop your kernel(s) into `kernels/3`
 - `./configure new_kernel & make install`

Carnegie Mellon Electrical & Computer ENGINEERING

Unexpected benefits

- More than just BLAS

Linkage Disequilibrium

SNP A 010101...
 SNP B 010011...
 Comm 010001... → 2

N. Alachiotis, Popovici, T.M. Low, Efficient Computation of Linkage Disequilibria as Dense Linear Algebra Operations, HICOMB 2016

kNN

APSP

DNA Fingerprinting

Carnegie Mellon Electrical & Computer ENGINEERING

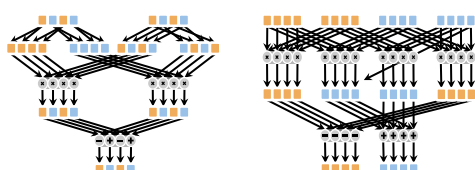
What everyone is saying...

PACKING

Carnegie Mellon Electrical & Computer ENGINEERING

Reasons for packing

- Work with different data layouts



- Hide additional operations

$$C = f \odot ((A \otimes B) \oplus C)$$

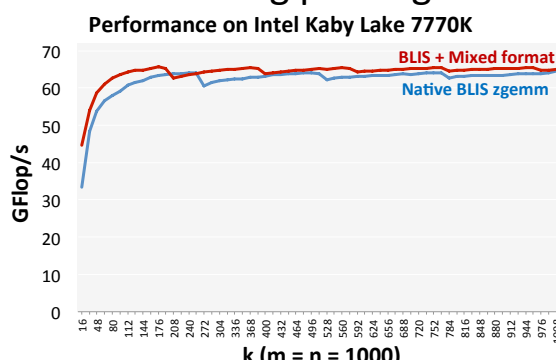
- Generalize the framework

Carnegie Mellon Electrical & Computer ENGINEERING

Work with Different Data Layout

- Limited SIMD support for complex multiplication
- Interleaved vs Non-interleaved
- Switch format during packing

Performance on Intel Kaby Lake 7770K



k (m = n = 1000)	Native BLIS zgemm (GFlop/s)	BLIS + Mixed format (GFlop/s)
16	35	45
48	55	60
80	60	65
112	62	66
144	63	66
176	63	66
208	63	66
240	63	66
272	63	66
304	63	66
336	63	66
368	63	66
400	63	66
432	63	66
464	63	66
496	63	66
528	63	66
560	63	66
592	63	66
624	63	66
656	63	66
688	63	66
720	63	66
752	63	66
784	63	66
816	63	66
848	63	66
880	63	66
912	63	66
944	63	66
976	63	66
1008	63	66

D. Popovici, F. Franchetti, T.M. Low, Mixed Data Format for Vectorized Complex Kernel, HPEC, 2017

Carnegie Mellon Electrical & Computer ENGINEERING

Hide additional operations

- Finite Field over large primes
 - Reuse BLAS & LAPACK for finite fields

```

graph TD
  Input[Input] --> TC1[Type Conversion]
  TC1 --> GEMM[GEMM]
  GEMM --> Modulo[Modulo p]
  Modulo --> TC2[Type Conversion]
  TC2 --> Output[Output]
  
```

J. Johnson, T.M. Low, M. Lambert, P. Oostema, B. D. Saunders, High-Performance Kernels for Exact Linear Algebra, ACA 2017

Carnegie Mellon Electrical & Computer ENGINEERING

Hide additional operations

- Finite Field over large primes
 - Reuse BLAS & LAPACK for finite fields

```

graph TD
  Input[Input] --> TC1[Type Conversion]
  TC1 --> GEMM[GEMM]
  GEMM --> Modulo[Modulo p]
  Modulo --> TC2[Type Conversion]
  TC2 --> Output[Output]
  
```

J. Johnson, T.M. Low, M. Lambert, P. Oostema, B. D. Saunders, High-Performance Kernels for Exact Linear Algebra, ACA 2017

Carnegie Mellon Electrical & Computer ENGINEERING

Generalize the framework

- Finite fields for small primes
 - 4 Russians Method

Table Creation

$$B = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad T_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

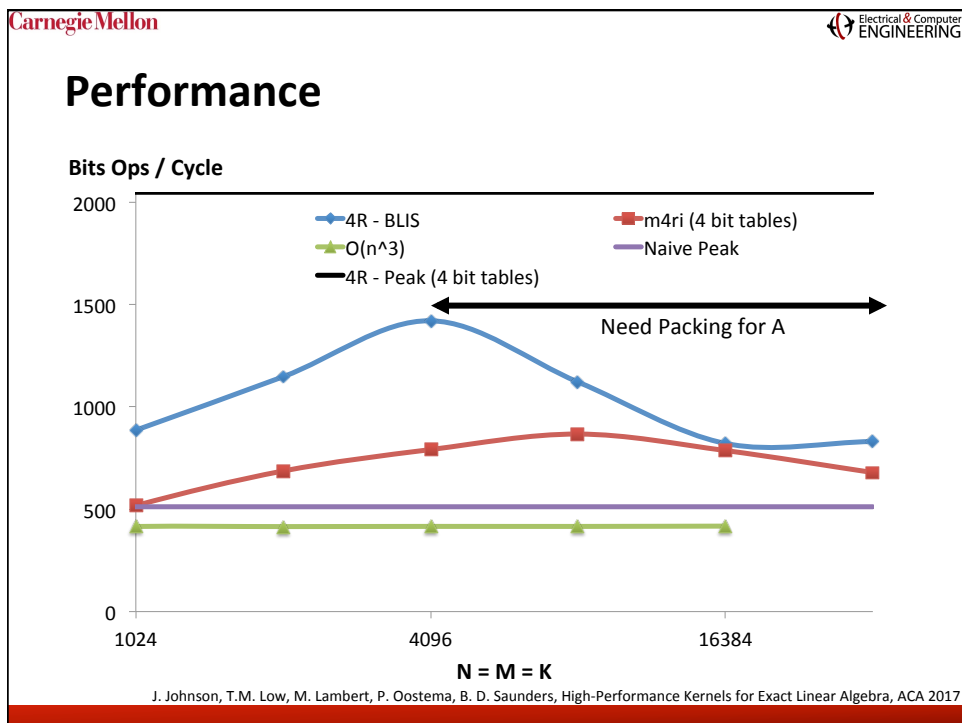
Compute with A as index


$$A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad T_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$


J. Johnson, T.M. Low, M. Lambert, P. Oostema, B. D. Saunders, High-Performance Kernels for Exact Linear Algebra, ACA 2017




Carnegie Mellon 

What everyone is NOT saying...

PACKING

Carnegie Mellon 

Remember how easy to add kernels?

Carnegie Mellon 

Ease of use

- Actual work
 - Create directory (recursive copy and rename)
 - Edit `bli_kernel.h`
 - Drop your kernel(s) into `kernels/3`
 - `./configure new_kernel` & make install

Introducing packing is not easy

- Files edited or added

```

frame/1m/bli_11m_ft.h
frame/1m/bli_11m_ker.h
frame/1m/packm/bli_packm.h
frame/1m/packm/bli_packm_cxk.c
frame/include/bli_kernel_macro_defs.h
frame/include/bli_kernel_pre_macro_defs.h
frame/include/bli_kernel_prototypes.h
frame/ind/include/bli_kernel_ind_macro_defs.h
frame/ind/include/bli_packm_ind_pre_macro_defs.h
frame/3/gemm/bli_gemm_cntl.c

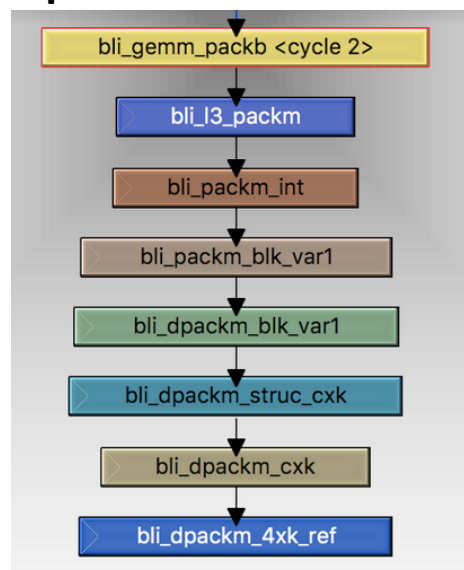
```

```

frame/1m/packm/bli_packm_blk_var2.c
frame/1m/packm/bli_packm_blk_var2.h
frame/1m/packm/bli_packm_cxk_cmu.c
frame/1m/packm/bli_packm_cxk_cmu.h
frame/1m/packm/bli_packm_struct_cxk_cmu.c
frame/1m/packm/bli_packm_struct_cxk_cmu.h
frame/1m/packm/ukernels/bli_packm_cxk_cmu_ref.c
frame/1m/packm/ukernels/bli_packm_cxk_cmu_ref.h
frame/ind/include/bli_packm_cmu_macro_defs.h

```

Calling sequence



My wish list

- As easy as adding kernels
- Files to edit or add

frame/1m/bli_1m_ft.h	frame/1m/packm/bli_packm_blk_var2.c
frame/1m/bli_1m_ker.h	frame/1m/packm/bli_packm_blk_var2.h
frame/1m/packm/bli_packm.h	frame/1m/packm/bli_packm_cxk_cmu.c
frame/1m/packm/bli_packm_cxk.c	frame/1m/packm/bli_packm_cxk_cmu.h
frame/include/bli_kernel_macro_defs.h	frame/1m/packm/bli_packm_struc_cxk_cmu.c
frame/include/bli_kernel_pre_macro_defs.h	frame/1m/packm/bli_packm_struc_cxk_cmu.h
frame/include/bli_kernel_prototypes.h	frame/1m/packm/ukernels/bli_packm_cxk_cmu_ref.c
frame/ind/include/bli_kernel_ind_macro_defs.h	frame/1m/packm/ukernels/bli_packm_cxk_cmu_ref.h
frame/ind/include/bli_packm_ind_pre_macro_defs.h	frame/ind/include/bli_packm_cmu_macro_defs.h
frame/3/gemm/bli_gemm_cntl.c	

- Move from `frame` to `kernel`

Summary

- Lighter and more nimble framework
- Refactoring of the packing is proposed
- Inputs from BLIS user and developers wanted

Discussion