Benchmarking the Linear Algebra Awareness of TensorFlow and PyTorch

Authors | **A. Sankaran[1], NA. Alashti[1], C. Psarras[1], P. Bientinesi[2]**

Speaker | **Aravind Sankaran**

[1] RWTH Aachen University, Germany
[2] Umeå University, Sweden

State of the art

- Linear algebra operations form major performance bottlenecks.

State of the art

- Linear algebra operations form major performance bottlenecks.

1973: Proposal for standard linear algebra subprograms:

"[..] a fairly small number of basic operations which are generally responsible for a significant percentage of the total execution time" - ***Hanson, Krogh, Lawson***

State of the art

- Linear algebra operations form major performance bottlenecks.

- Lots of mature numerical linear algebra libraries

1973: Proposal for standard linear algebra subprograms:

"[..] a fairly small number of basic operations which are generally responsible for a significant percentage of the total execution time" - ***Hanson, Krogh, Lawson***

2022: Linear algebra software landscape

PETSc, Trilinos, …

ScaLAPACK, PLAPACK, Elemental, …

LAPACK, Plasma, SuperMatrix, Magma, …

BLAS-1, BLAS-2, BLAS-3, ATLAS, BTO-BLAS, BLIS, …

State of the art

- Linear algebra operations form major performance bottlenecks.
- Lots of mature numerical linear algebra libraries
- Is that enough?

1973: Proposal for standard linear algebra subprograms:

“[..] a fairly small number of basic operations which are generally responsible for a significant percentage of the total execution time” - ***Hanson, Krogh, Lawson***

2022: Linear algebra software landscape

PETSc, Trilinos, …

ScaLAPACK, PLAPACK, Elemental, …

LAPACK, Plasma, SuperMatrix, Magma, …

BLAS-1, BLAS-2, BLAS-3, ATLAS, BTO-BLAS, BLIS, …

Kernel development: Typical Approach - ?

1. Identify the bottleneck
   e.g., linear system, eigen problem, least-squares problem, SVD

2. Encapsulate into a function
   e.g., dgesv, dsyevd, dgelsx, dgesvd

3. Optimize

2022: Linear algebra software landscape

PETSc, Trilinos, …

ScaLAPACK, PLAPACK, Elemental, …

LAPACK, Plasma, SuperMatrix, Magma, …

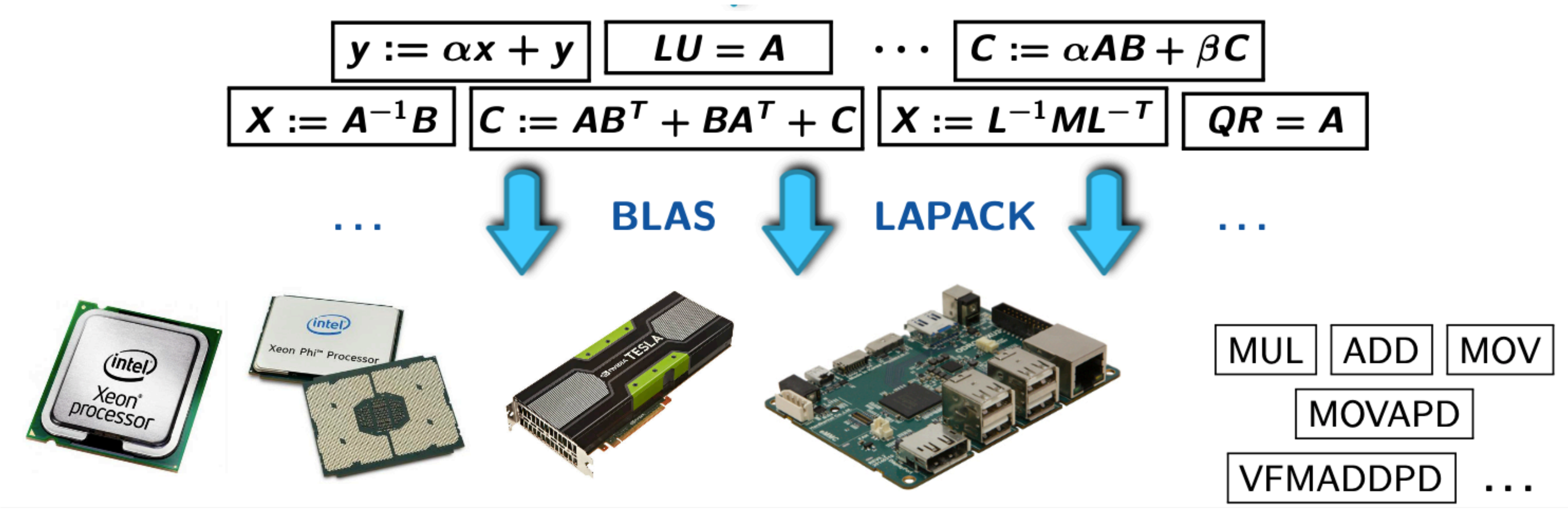
BLAS-1, BLAS-2, BLAS-3, ATLAS, BTO-BLAS, BLIS, …

Kernel development: Typical Approach - ?

1. Identify the bottleneck
   e.g., linear system, eigen problem, least-squares problem, SVD

2. Encapsulate into a function
   e.g., dgesv, dsyevd, dgelsx, dgesvd

3. Optimize



2022: Linear algebra software landscape

PETSc, Trilinos, …

ScaLAPACK, PLAPACK, Elemental, …

LAPACK, Plasma, SuperMatrix, Magma, …

BLAS-1, BLAS-2, BLAS-3, ATLAS, BTO-BLAS, BLIS, …

Where do TensorFlow and PyTorch fit in?

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k (z_k - H x_k^b); \quad P_k^a := (I - K_K H) P_k^b$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

?

$\boxed{y := \alpha x + y}$ $\boxed{LU = A}$ $\cdots$ $\boxed{C := \alpha AB + \beta C}$

$\boxed{X := A^{-1} B}$ $\boxed{C := AB^T + BA^T + C}$ $\boxed{X := L^{-1} M L^{-T}}$ $\boxed{QR = A}$

... BLAS LAPACK ...

Kernels are highly optimized.

MUL ADD MOV

MOVAPD

VFMADDPD ...

Where do TensorFlow and PyTorch fit in?

$$K_k := P^b_k H^T (H P^b_k H^T + R)^{-1}; \quad x^a_k := x^b_k + K_k(z_k - H x^b_k); \quad P^a_k := (I - K_K H)\, P^b_k$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A)\Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

TensorFlow PyTorch

$\boxed{y := \alpha x + y}$ $\boxed{LU = A}$ $\cdots$ $\boxed{C := \alpha AB + \beta C}$

$\boxed{X := A^{-1}B}$ $\boxed{C := AB^T + BA^T + C}$ $\boxed{X := L^{-1}ML^{-T}}$ $\boxed{QR = A}$

... BLAS LAPACK ...

Kernels are highly optimized.

MUL ADD MOV
MOVAPD
VFMADDPD ...

Does the frameworks make best use of the kernels?

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_k H) P_k^b$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

Does the mapping make best use of the available kernels?

TensorFlow PyTorch

$\boxed{y := \alpha x + y}$ $\boxed{LU = A}$ $\cdots$ $\boxed{C := \alpha AB + \beta C}$

$\boxed{X := A^{-1} B}$ $\boxed{C := AB^T + BA^T + C}$ $\boxed{X := L^{-1} M L^{-T}}$ $\boxed{QR = A}$

... BLAS LAPACK ...

Kernels are highly optimized.

MUL ADD MOV

MOVAPD

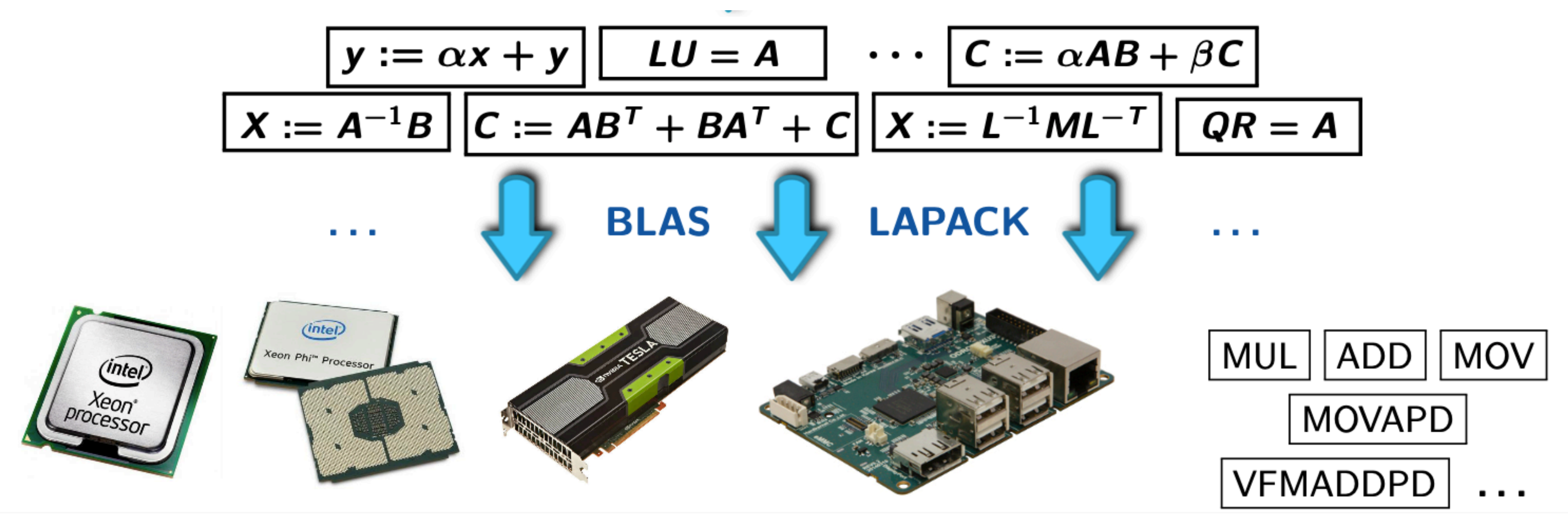
VFMADDPD ...

Does the frameworks make best use of the kernels?

Linear Algebra Expression

$$\mathbf{y} := H^T\mathbf{y} + (I_n - H^T H)\mathbf{x}$$

where $H \in \mathbb{R}^{n\times n}$ $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$



Does the frameworks make best use of the kernels?

Linear Algebra Expression

$$\mathbf{y} := H^T\mathbf{y} + (I_n - H^T H)\mathbf{x}$$

where $H \in \mathbb{R}^{n \times n}$ $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

Operation	Kernel	FLOPs
$\mathbf{t}_1 := H^T\mathbf{y}$	gemv	$2n^2$
$T_2 := H^T H$	gemm	$2n^3$
$T_3 := I_n - T_2$	axpy	n
$\mathbf{t}_4 := T_3\mathbf{x}$	gemv	$2n^2$
$\mathbf{y} := \mathbf{t}_1 + \mathbf{t}_4$	axpy	n

Total FLOPs $\approx 2n^3 + 4n^2 + 2n$

Exec time $\approx 0.43 sec$

n=3000

executed on a single core of Intel Xeon CPU

$\boxed{y := \alpha x + y}$ $\boxed{LU = A}$ $\cdots$ $\boxed{C := \alpha AB + \beta C}$

$\boxed{X := A^{-1}B}$ $\boxed{C := AB^T + BA^T + C}$ $\boxed{X := L^{-1}ML^{-T}}$ $\boxed{QR = A}$

... BLAS LAPACK ...

MUL ADD MOV

MOVAPD

VFMADDPD ...

Modern Inverse Problems

RWTH AACHEN UNIVERSITY

Does the frameworks make best use of the kernels?

Linear Algebra Expression

$$\mathbf{y} := H^T\mathbf{y} + (I_n - H^T H)\mathbf{x}$$

where $H \in \mathbb{R}^{n\times n}$ $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

Variant 1

$$\mathbf{y} := H^T\mathbf{y} + (I_n - H^T H)\mathbf{x}$$

Operation	Kernel	FLOPs
$\mathbf{t}_1 := H^T\mathbf{y}$	gemv	$2n^2$
$T_2 := H^T H$	gemm	$2n^3$
$T_3 := I_n - T_2$	axpy	n
$\mathbf{t}_4 := T_3\mathbf{x}$	gemv	$2n^2$
$\mathbf{y} := \mathbf{t}_1 + \mathbf{t}_4$	axpy	n

Total FLOPs $\approx 2n^3 + 4n^2 + 2n$

Exec time $\approx 0.43sec$

Variant 2

$$\mathbf{y} := H^T\mathbf{y} + x - H^T(H\mathbf{x})$$

Operation	Kernel	FLOPs
$\mathbf{t}_1 := H^T\mathbf{y}$	gemv	$2n^2$
$\mathbf{t}_2 := H\mathbf{x}$	gemv	$2n^2$
$\mathbf{t}_3 := H^T\mathbf{t}_2$	gemv	$2n^2$
$\mathbf{t}_4 := \mathbf{t}_1 - \mathbf{t}_3$	axpy	n
$\mathbf{y} := \mathbf{t}_4 + \mathbf{x}$	axpy	n

Total FLOPs $\approx 6n^2 + 2n$

Exec time $\approx 0.003sec$

n=3000
executed on a single core of Intel Xeon CPU

$y := \alpha x + y$ | $LU = A$ | ... | $C := \alpha AB + \beta C$

$X := A^{-1}B$ | $C := AB^T + BA^T + C$ | $X := L^{-1}ML^{-T}$ | $QR = A$

... BLAS LAPACK ...

MUL | ADD | MOV

MOVAPD

VFMADDPD ...

Do we make best use of the kernels?

Linear Algebra Expression

$$\mathbf{y} := H^T\mathbf{y} + (I_n - H^T H)\mathbf{x}$$

where $H \in \mathbb{R}^{n\times n}$ $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$

Variant 1

$$\mathbf{y} := H^T\mathbf{y} + (I_n - H^T H)\mathbf{x}$$

Operation	Kernel	FLOPs
$\mathbf{t}_1 := H^T\mathbf{y}$	gemv	$2n^2$
$T_2 := H^T H$	gemm	$2n^3$
$T_3 := I_n - T_2$	axpy	n
$\mathbf{t}_4 := T_3\mathbf{x}$	gemv	$2n^2$
$\mathbf{y} := \mathbf{t}_1 + \mathbf{t}_4$	axpy	n

Total FLOPs $\approx 2n^3 + 4n^2 + 2n$

Exec time $\approx 0.43sec$

Variant 2

$$\mathbf{y} := H^T\mathbf{y} + x - H^T(H\mathbf{x})$$

Operation	Kernel	FLOPs
$\mathbf{t}_1 := H^T\mathbf{y}$	gemv	$2n^2$
$\mathbf{t}_2 := H\mathbf{x}$	gemv	$2n^2$
$\mathbf{t}_3 := H^T\mathbf{t}_2$	gemv	$2n^2$
$\mathbf{t}_4 := \mathbf{t}_1 - \mathbf{t}_3$	axpy	n
$\mathbf{y} := \mathbf{t}_4 + \mathbf{x}$	axpy	n

Total FLOPs $\approx 6n^2 + 2n$

Exec time $\approx 0.003sec$

n=3000
executed on a single core of Intel Xeon CPU

$\boxed{y := \alpha x + y}$ $\boxed{LU = A}$ $\cdots$ $\boxed{C := \alpha AB + \beta C}$

$\boxed{X := A^{-1}B}$ $\boxed{C := AB^T + BA^T + C}$ $\boxed{X := L^{-1}ML^{-T}}$ $\boxed{QR = A}$

... BLAS LAPACK ...

MUL ADD MOV
MOVAPD
VFMADDPD ...

Variant 2 is orders of magnitude faster than Variant 1!

Linear Algebra Mapping Problem

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_k H) P_k^b$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A)\Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

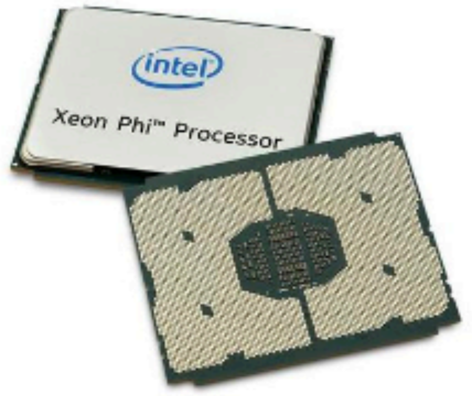
?

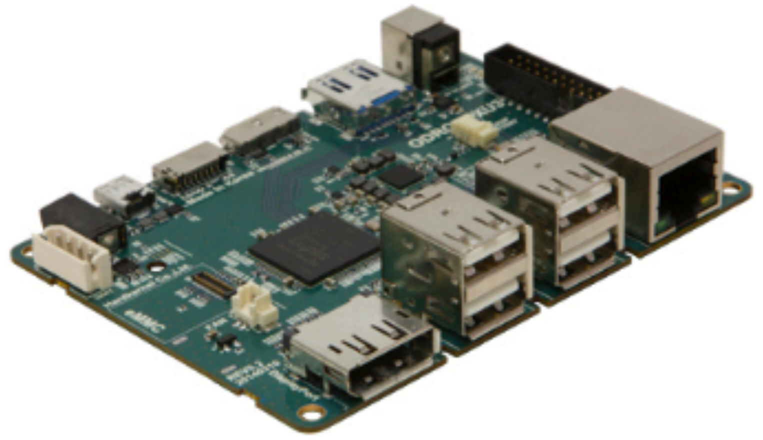
There can be hundreds of possible mappings!

$\boxed{y := \alpha x + y}$ $\boxed{LU = A}$ $\cdots$ $\boxed{C := \alpha AB + \beta C}$

$\boxed{X := A^{-1} B}$ $\boxed{C := AB^T + BA^T + C}$ $\boxed{X := L^{-1} M L^{-T}}$ $\boxed{QR = A}$

... BLAS LAPACK ...





MUL ADD MOV

MOVAPD

VFMADDPD ...

Linear Algebra Mapping Problem

$$K_k := P_k^b H^T (H P_k^b H^T + R)^{-1}; \quad x_k^a := x_k^b + K_k(z_k - H x_k^b); \quad P_k^a := (I - K_k H) P_k^b$$

$$\begin{cases} C_\dagger := PCP^T + Q \\ K := C_\dagger H^T (H C_\dagger H^T)^{-1} \end{cases}$$

$$\Lambda := S(S^T AWAS)^{-1} S^T; \; \Theta := \Lambda AW; \; M_k := X_k A - I$$
$$X_{k+1} := X_k - M_k \Theta - (M_k \Theta)^T + \Theta^T (A X_k A - A) \Theta$$

$$x := A(B^T B + A^T R^T \Lambda R A)^{-1} B^T B A^{-1} y \quad \ldots \quad E := Q^{-1} U (I + U^T Q^{-1} U)^{-1} U^T$$

Linear Algebra Mapping Problem (LAMP)

$\boxed{y := \alpha x + y}$ $\boxed{LU = A}$ $\cdots$ $\boxed{C := \alpha AB + \beta C}$

$\boxed{X := A^{-1} B}$ $\boxed{C := AB^T + BA^T + C}$ $\boxed{X := L^{-1} M L^{-T}}$ $\boxed{QR = A}$

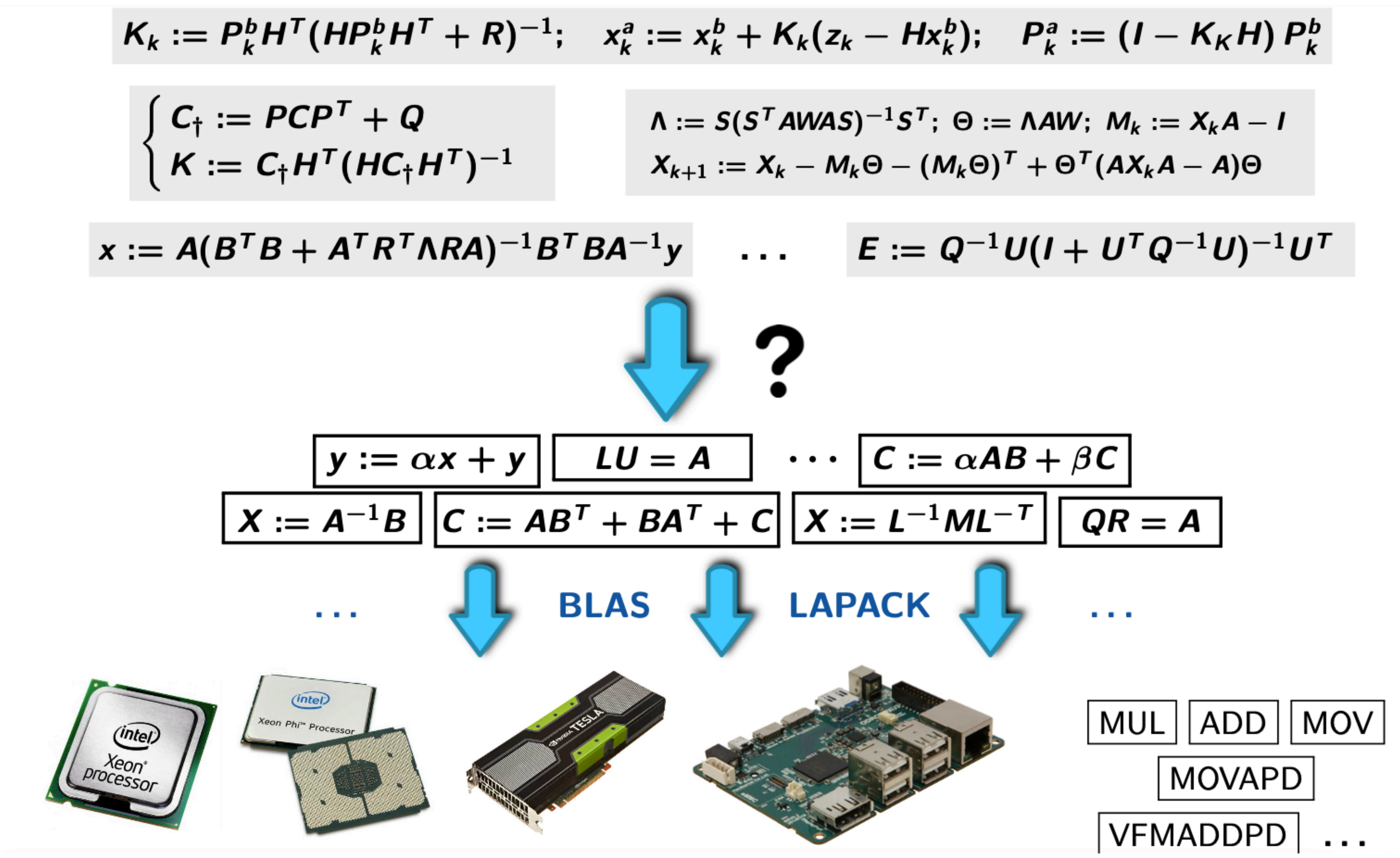
... BLAS LAPACK ...

MUL ADD MOV

MOVAPD

VFMADDPD ...

Linear Algebra Mapping Problem



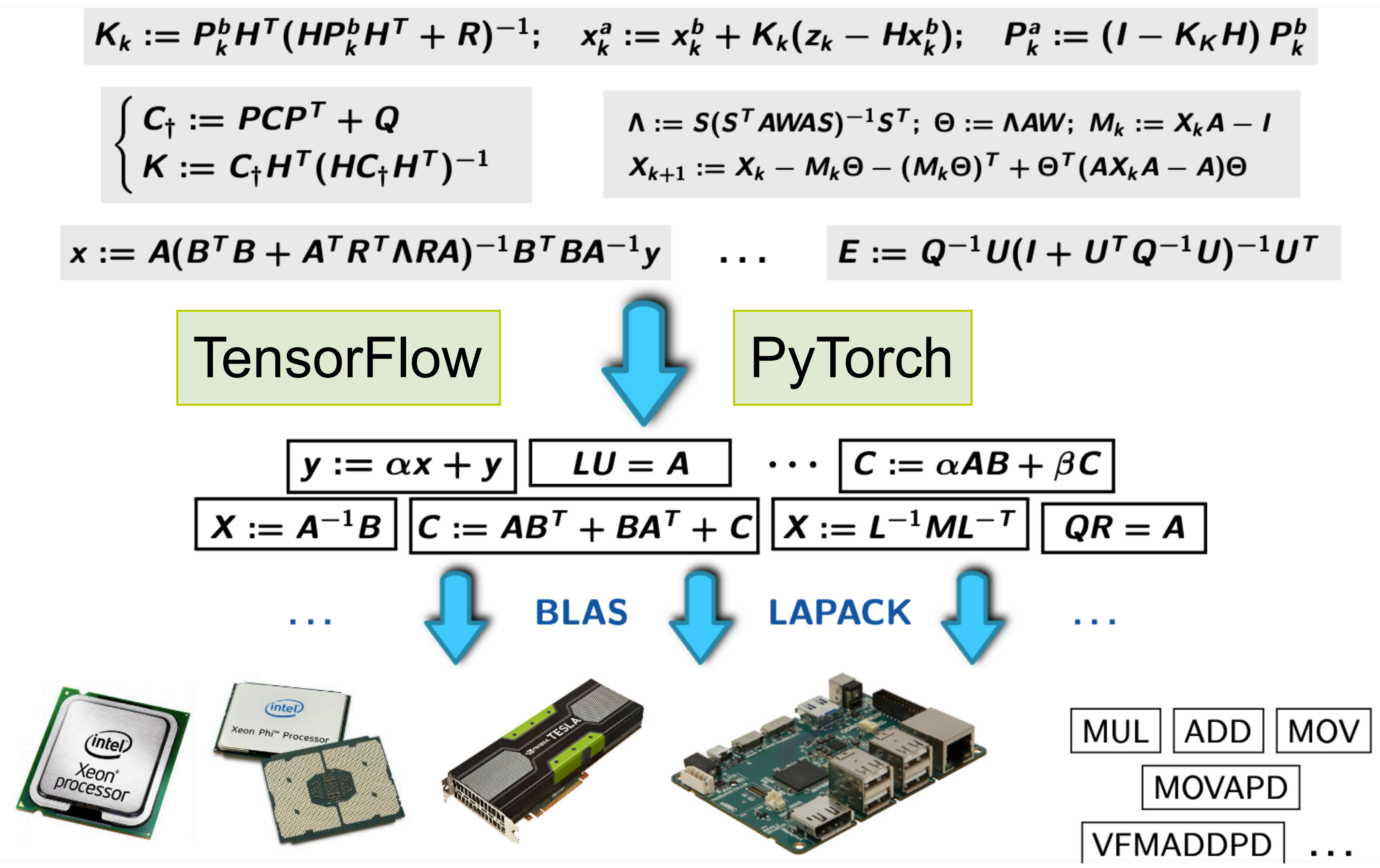
Linear Algebra Mapping Problem (LAMP)

- $\mathcal{E}$: a sequence of explicit assignments $\quad var_i := EXP_i$
- $\mathcal{K}$: a set of available computational building blocks $\quad$ e.g., BLAS, LAPACK, ...
- $\mathcal{M}$: a cost function defined over $\mathcal{K}^+$ $\quad$ #FLOPs, exec. time, #mem.ops, stability

LAMP:
Find a sequence of calls to building blocks in $\mathcal{K}$, optimal according to $\mathcal{M}$, that computes all the assignments in $\mathcal{E}$.

C. Psarras, H. Barthels, P. Bientinesi, [arXiv:1911.09421]
"The Linear Algebra Mapping Problem. Current state of linear algebra languages and libraries", ACM TOMS, 2022

Linear Algebra Mapping Problem



Linear Algebra Mapping Problem (LAMP)

- $\mathcal{E}$: a sequence of explicit assignments $\quad var_i := EXP_i$
- $\mathcal{K}$: a set of available computational building blocks $\quad$ e.g., BLAS, LAPACK, ...
- $\mathcal{M}$: a cost function defined over $\mathcal{K}^+$ $\quad$ #FLOPs, exec. time, #mem.ops, stability

LAMP:
Find a sequence of calls to building blocks in $\mathcal{K}$, optimal according to $\mathcal{M}$, that computes all the assignments in $\mathcal{E}$.

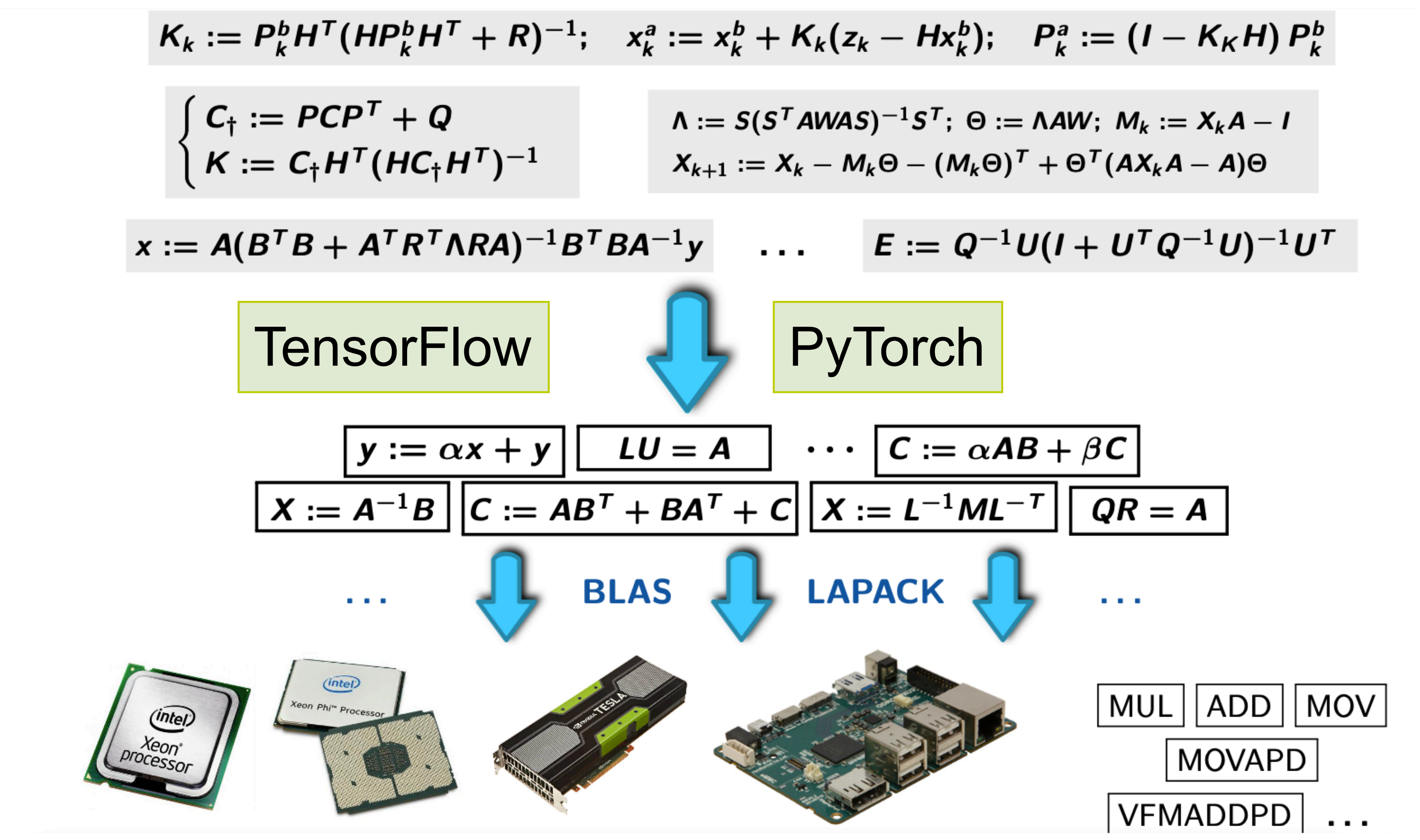
- Suboptimal solution $\rightarrow$ easy
- Optimality $\rightarrow$ NP complete

C. Psarras, H. Barthels, P. Bientinesi, [arXiv:1911.09421]
"The Linear Algebra Mapping Problem. Current state of linear algebra languages and libraries", ACM TOMS, 2022

Linear Algebra Awareness Benchmark

In this work:

- We pin point the sub-optimalities and expose optimization opportunities.
- Focus on ML frameworks - TensorFLow (TF) and PyTorch (PyT). Linear algebra operations are ubiquitous in machine learning.



LAMP

- Suboptimal solution $\longrightarrow$ easy
- Optimality $\longrightarrow$ NP complete

Modern Inverse Problems

Setting up TF and PyT

Given: $A, B \in R^{3000 \times 3000}$

Expression: $Y = A^T B$

Mapping:

```
Y = gemm(AᵀB)
```

Setting up TF and PyT

Given: $A, B \in R^{3000 \times 3000}$

Expression: $Y = A^T B$

Mapping:

```
Y = gemm(AᵀB)
```

Expression	MKL-C	Eager (TF / PyT)	Graph (TF / PyT)
$A^T B$	0.39	0.40 / 0.40	0.40 / 0.40
Invoke GEMM?			

Execution times in sec

Setting up TF and PyT

Given: $A, B \in R^{3000\times3000}$

Expression: $Y = A^T B$

Mapping:

```
Y = gemm(AᵀB)
```

Expression	MKL-C	Eager (TF / PyT)	Graph (TF / PyT)
$A^T B$	0.39	0.40 / 0.40	0.40 / 0.40
Invoke GEMM?		✓	✓

Execution times in sec

Setting up TF and PyT

Given: $A, B \in R^{3000\times3000}$

Expression: $Y = A^T B$

Mapping:

```
Y = gemm(AᵀB)
```

Expression	MKL-C	Eager (TF / PyT)	Graph (TF / PyT)
$A^T B$	0.39	0.40 / 0.40	0.40 / 0.40
Invoke GEMM?		✓	✓

Expression: $Y = (A^T B)^T (A^T B)$

Mappings:

```
S₁ = gemm(AᵀB);
S₂ = gemm(AᵀB);
Y = gemm(S₁ᵀS₂);
```

```
S = gemm(AᵀB);
Y = gemm(SᵀS);
```

Setting up TF and PyT

Given: $A, B \in R^{3000\times3000}$

Expression: $Y = A^T B$

Mapping:

```
Y = gemm(AᵀB)
```

Expression: $Y = (A^T B)^T (A^T B)$

Mappings:

```
S1 = gemm(AᵀB);
S2 = gemm(AᵀB);
Y = gemm(S1ᵀS2);
```

```
S = gemm(AᵀB);
Y = gemm(SᵀS);
```

Expression	MKL-C	Eager (TF / PyT)	Graph (TF / PyT)
$A^T B$	0.39	0.40 / 0.40	0.40 / 0.40
Invoke GEMM?		✓	✓
$(A^T B)^T (A^T B)$	–	1.25 / 1.27	0.78 / 0.80
Graph optimizations?			

Execution times in sec

Setting up TF and PyT

Given: $A, B \in R^{3000\times3000}$

Expression: $Y = A^TB$

Mapping:

$$\texttt{Y = gemm(A}^\texttt{T}\texttt{B)}$$

Expression: $Y = (A^TB)^T(A^TB)$

Mappings:

$$\begin{aligned}&\texttt{S}_1 \texttt{ = gemm(A}^\texttt{T}\texttt{B);}\\ &\texttt{S}_2 \texttt{ = gemm(A}^\texttt{T}\texttt{B);}\\ &\texttt{Y = gemm(S}_1{}^\texttt{T}\texttt{S}_2\texttt{);}\end{aligned}$$

$$\begin{aligned}&\texttt{S = gemm(A}^\texttt{T}\texttt{B);}\\ &\texttt{Y = gemm(S}^\texttt{T}\texttt{S);}\end{aligned}$$

Expression	MKL-C	Eager (TF / PyT)	Graph (TF / PyT)
A^TB	0.39	0.40 / 0.40	0.40 / 0.40
Invoke GEMM?		✓	✓
$(A^TB)^T(A^TB)$	–	1.25 / 1.27	0.78 / 0.80
Graph optimizations?		✗	✓

Execution times in sec

Graph mode in TF and PyT

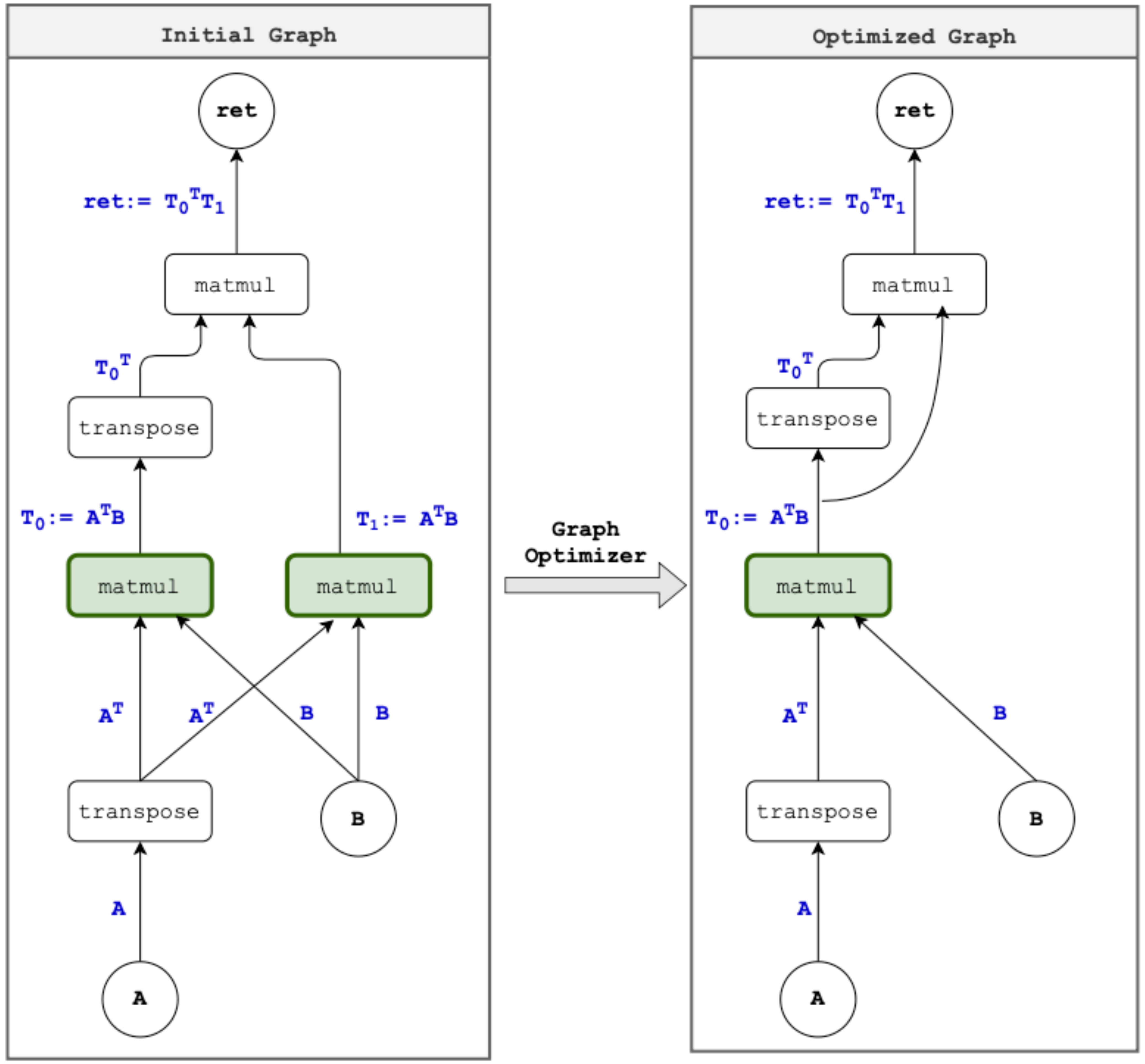
$\mathrm{ret} := (\mathbf{A}^T\mathbf{B})^T(\mathbf{A}^T\mathbf{B}) \qquad \mathbf{A},\mathbf{B} \in \mathbb{R}^{n\times n}$

```
import tensorflow as tf

@tf.function
def transformation(A,B){

    Aᵀ = tf.transpose(A)
    ret =tf.transpose(Aᵀ@B)@(Aᵀ@B)
    return ret
}
```

TensorFlow Code.

Expression	MKL-C	Eager (TF / PyT)	Graph (TF / PyT)
$A^T B$	0.39	0.40 / 0.40	0.40 / 0.40
$(A^T B)^T (A^T B)$	–	1.25 / 1.27	0.78 / 0.80

Execution time (in sec) for $n = 3000$



The Computational Graphs for $(A^T B)^T (A^T B)$.

Linear Algebra Awareness Benchmark

Conditions:

- Linked to optimized libraries (e.g., Intel MKL).
- Experiments are run in graph mode

Goal:

Evaluate the extent to which TF and PyT incorporate Linear algebra knowledge to speed up computations.

Experiments to evaluate Linear Algebra Awareness:

1. Common Sub-expression Elimination
2. Optimization of Matrix Chains
3. Exploiting Matrix Properties
4. Algebraic Manipulations
5. Code Motion

Experiment 1: Common Sub-expression Elimination

Parenthesized:

Expression	TF	PyT
$(A^T B)^T (A^T B)$	0.78 ✓	0.80 ✓

Experiment 1: Common Sub-expression Elimination

Expression: $Y = (A^T B)^T A^T B$

Mappings:

```
S1 = gemm(AᵀB);
S2 = gemm(AᵀB);
Y = gemm(S1ᵀS2);
```

```
S = gemm(AᵀB);
Y = gemm(SᵀS);
```

```
S1 = gemm(AᵀB);
S2 = gemm(S1ᵀAᵀ);
Y = gemm(S2B);
```

	Expression	TF	PyT
Parenthesized:	$(A^T B)^T (A^T B)$	0.78 ✓	0.80 ✓

Experiment 1: Common Sub-expression Elimination

Expression: $Y = (A^T B)^T A^T B$

Mappings:

```
S₁ = gemm(AᵀB);
S₂ = gemm(AᵀB);
Y  = gemm(S₁ᵀS₂);
```

```
S = gemm(AᵀB);
Y = gemm(SᵀS);
```

```
S₁ = gemm(AᵀB);
S₂ = gemm(S₁ᵀAᵀ);
Y  = gemm(S₂B);
```

	Expression	TF	PyT
Parenthesized:	$(A^T B)^T (A^T B)$	0.78 ✓	0.80 ✓
Not Parenthesized:	$(A^T B)^T A^T B$	1.17	1.15

Experiment 1: Common Sub-expression Elimination

Expression: $Y = (A^TB)^TA^TB$

Mappings:

```
S1 = gemm(AᵀB);
S2 = gemm(AᵀB);
Y  = gemm(S1ᵀS2);
```

```
S = gemm(AᵀB);
Y = gemm(SᵀS);
```

```
S1 = gemm(AᵀB);
S2 = gemm(S1ᵀAᵀ);
Y  = gemm(S2B);
```

	Expression	TF	PyT
Parenthesized:	$(A^TB)^T(A^TB)$	0.78 ✓	0.80 ✓
Not Parenthesized:	$(A^TB)^TA^TB$	1.17 ✗	1.15 ✗

Experiment 1: Common Sub-expression Elimination

Expression: $Y = (A^TB)^TA^TB$

Mappings:

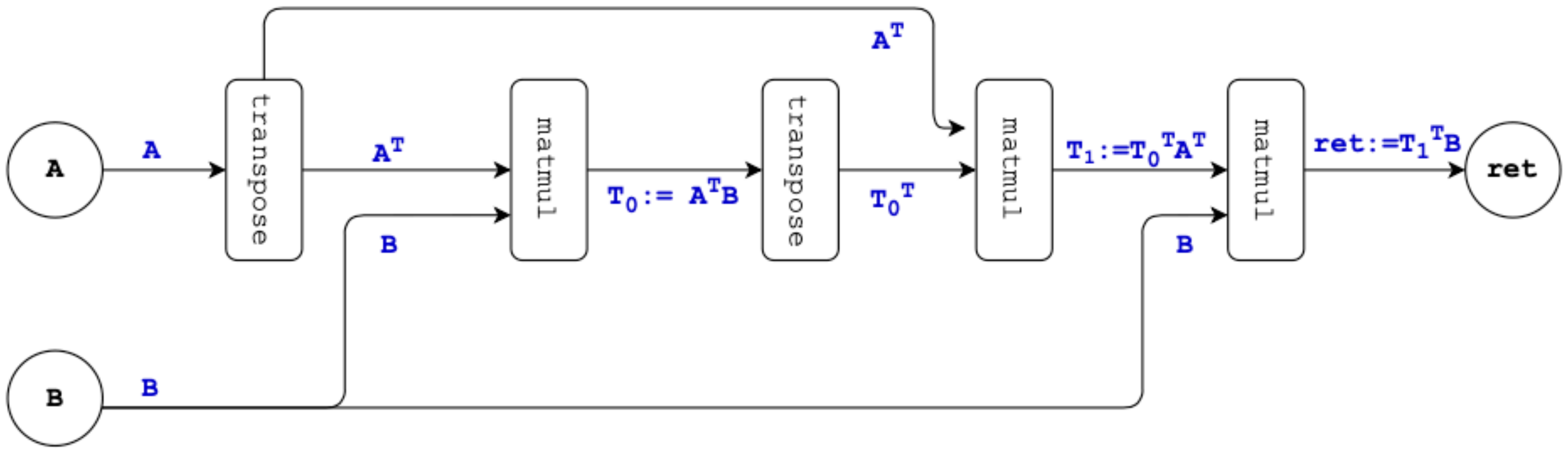
$$\begin{aligned} S_1 &= \texttt{gemm}(A^TB); \\ S_2 &= \texttt{gemm}(A^TB); \\ Y &= \texttt{gemm}(S_1^TS_2); \end{aligned}$$

$$\begin{aligned} S &= \texttt{gemm}(A^TB); \\ Y &= \texttt{gemm}(S^TS); \end{aligned}$$

$$\begin{aligned} S_1 &= \texttt{gemm}(A^TB); \\ S_2 &= \texttt{gemm}(S_1^TA^T); \\ Y &= \texttt{gemm}(S_2B); \end{aligned}$$

	Expression	TF	PyT
Parenthesized:	$(A^TB)^T(A^TB)$	0.78 ✓	0.80 ✓
Not Parenthesized:	$(A^TB)^TA^TB$	1.17 ✗	1.15 ✗

Execution times in sec



The Computational Graph for $(A^TB)^TA^TB$.

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H\mathbf{x}$	0.40	0.41	0.006
$H^T(H\mathbf{x})$	0.006	0.004	-

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H \mathbf{x}$	0.40	0.41	0.006
$H^T (H \mathbf{x})$	0.006	0.004	-

$$H^T(H\mathbf{x})$$

$$(H^T H)\mathbf{x}$$

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H\mathbf{x}$	0.40	0.41	0.006
$H^T(H\mathbf{x})$	0.006	0.004	-

$H^T(H\mathbf{x})$

$$\mathbf{t}_0 \leftarrow H\mathbf{x} \qquad 2n^2$$
$$\mathbf{t}_1 \leftarrow H^T\mathbf{t}_0 \qquad 2n^2$$

$(H^T H)\mathbf{x}$

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H\mathbf{x}$	0.40	0.41	0.006
$H^T(H\mathbf{x})$	0.006	0.004	-

$H^T(H\mathbf{x})$

$$\begin{aligned} \mathbf{t}_0 &\leftarrow H\mathbf{x} & 2n^2 \\ \mathbf{t}_1 &\leftarrow H^T\mathbf{t}_0 & 2n^2 \end{aligned}$$

$(H^T H)\mathbf{x}$

$$\begin{aligned} T_0 &\leftarrow H^T H & 2n^3 \\ \mathbf{t}_1 &\leftarrow T_0\mathbf{x} & 2n^2 \end{aligned}$$

Modern Inverse Problems

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H\mathbf{x}$	0.40	0.41	0.006
$H^T(H\mathbf{x})$	0.006	0.004	-

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H\mathbf{x}$	0.40	0.41	0.006
$H^T(H\mathbf{x})$	0.006	0.004	-
Right to left?	✗	✗	✓

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H\mathbf{x}$	0.40	0.41	0.006
$H^T(H\mathbf{x})$	0.006	0.004	-
Right to left?	✗	✗	✓
$\mathbf{y}^T H^T H$	0.006	0.005	0.005
$(\mathbf{y}^T H^T)H$	0.006	0.005	-
Left to right?	✓	✓	✓

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H \mathbf{x}$	0.40	0.41	0.006
$H^T (H \mathbf{x})$	0.006	0.004	-
Right to left?	✗	✗	✓
$\mathbf{y}^T H^T H$	0.006	0.005	0.005
$(\mathbf{y}^T H^T) H$	0.006	0.005	-
Left to right?	✓	✓	✓
$H^T \mathbf{y} \mathbf{x}^T H$	0.41	0.40	0.01
$(H^T \mathbf{y})(\mathbf{x}^T H)$	0.01	0.01	-
Mixed?	✗	✗	✓

Experiment 2: Optimization of Matrix Chains

Expression	TF	PyT	
	matmul	matmul	multi_dot
$H^T H \mathbf{x}$	0.40	0.41	0.006
$H^T (H \mathbf{x})$	0.006	0.004	-
Right to left?	✗	✗	✓
$\mathbf{y}^T H^T H$	0.006	0.005	0.005
$(\mathbf{y}^T H^T) H$	0.006	0.005	-
Left to right?	✓	✓	✓
$H^T \mathbf{y} \mathbf{x}^T H$	0.41	0.40	0.01
$(H^T \mathbf{y})(\mathbf{x}^T H)$	0.01	0.01	-
Mixed?	✗	✗	✓

- Evaluate all possible parenthesizations and choose the one with minimum FLOP?
- Number of possible parenthesizations for a matrix chain of length *m* is given by the *(m-1)*th Catalan number:

$$C_{m-1} = \frac{(2m)!}{(m+1)!m!}.$$

- Evaluate optimal order at least for *m* < *K*?

Aravind Sankaran (aravind.sankaran@rwth-aachen.de) | RWTH Aachen University

Experiment 3: Exploiting Matrix Properties

Given: $A, B \in R^{n \times n}$

Expression: $Y = A^T B$

Mapping:

```
Y = gemm(AᵀB)
```

$2n^3$

If *A* is triangular (L)

```
Y = trmm(LᵀB)
```

n^3

If $B = A$

```
Y = syrk(AᵀA)
```

n^3

Experiment 3: Exploiting Matrix Properties

Given: $A, B \in R^{n \times n}$

Expression: $Y = A^T B$

Mapping:

```
Y = gemm(AᵀB)
```

$2n^3$

If *A* is triangular (L)

```
Y = trmm(LᵀB)
```

n^3

If *B* = *A*

```
Y = syrk(AᵀA)
```

n^3

Expr	SciPy	TF		PyT		
	BLAS	matmul	optim	matmul	optim	
AB	0.40	0.40	-	0.41	-	
LB	0.24	0.40	n.a	0.40	n.a	❌
AA^T	0.24	0.41	n.a	0.39	n.a	❌

Execution times in sec

Experiment 3: Exploiting Matrix Properties

Given: $A, B \in R^{n\times n}$

Expression: $Y = A^T B$

Mapping:

```
Y = gemm(AᵀB)
```

$2n^3$

If A is triangular (L)

```
Y = trmm(LᵀB)
```

n^3

If $B = A$

```
Y = syrk(AᵀA)
```

n^3

Expr	SciPy	TF		PyT		
	BLAS	matmul	optim	matmul	optim	
AB	0.40	0.40	-	0.41	-	
LB	0.24	0.40	n.a	0.40	n.a	✗
AA^T	0.24	0.41	n.a	0.39	n.a	✗

- If matrix properties can be propogated through the computational graph, it would be possible to detect algebraic simplifications that can speed up computation.

  E.g. if it can be known that Q is orthogonal, then $Q^TQ = I$; matrix multiplication can be avoided.

Experiment 4: Algebraic Manipulations

Distributive Law:

	TF		PyT		
	LHS	RHS	LHS	RHS	
$AB + AC = A(B + C)$	0.78	040	0.81	0.41	❌
$A\mathbf{x} - H^T(H\mathbf{x}) = (A - H^T H)\mathbf{x}$	0.01	0.42	0.01	0.41	❌

Experiment 4: Algebraic Manipulations

Blocked Operands:

$$A_B := \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \in \mathbb{R}^{n \times n} \qquad B_B := \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

$$A_B B_B = \begin{bmatrix} (A_1 B_1) \\ (A_2 B_2) \end{bmatrix}$$

	TF		PyT		
	LHS	RHS	LHS	RHS	
Blocked matrices	0.40	0.20	0.40	0.20	✗

Loop-Invariant Code Motion:

Naive

```
V = [v1, v2, v3]
for i in range(3):
  Y[:,:,i] = A@B + V[i]@V[i]^T
```

Recommended

```
V = [v1, v2, v3]
tmp = A@B
for i in range(3):
  Y[:,:,i] = tmp + V[i]@V[i]^T
```

Property	TF		PyT	
	Naive	Reco	Naive	Reco
Loop-inv code motion	0.42	0.42	0.42	0.41

Experiment 5: Code Motion

Partial operand access:

Naive

```
#Sum
Y = (A+B)[2,2]

#Product
Y = (A@B)[2,2]
```

Recommended

```
#Sum
Y = A[2,2]+B[2,2]

#Product
Y = dot(A[2,:],B[:,2])
```

Property	TF		PyT		
	Naive	Reco	Naive	Reco	
Partial-op access (sum)	0.011	6e-4	0.018	2e-3	✗
Partial-op access (product)	0.39	2e-3	0.40	3e-3	✗

Summary

- Evaluated the extent to which TF and PyT use Linear Algebra knowledge to speed up computations.

- We exposed some of the sub-optimalities presented guidelines to help both the framework developers and end-users to achieve better performance.