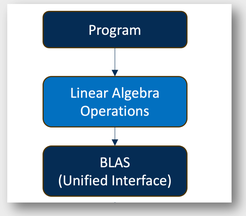
Machine-Learning-Driven Runtime Optimization of BLAS Level 3 on Modern Multi-Core Systems

Yufan Xia, Giuseppe Barca

E-mail: xiayufan12345@outlook.com, giuseppe.barca@unimelb.edu.au

Introduction

BLAS Illustration



BLAS III Operations:

1. Matrix Multiply
   1. GEMM
   2. SYMM
   3. SYRK
   4. SYR2K
   5. TRMM
2. Solve for Matrix
   1. TRSM

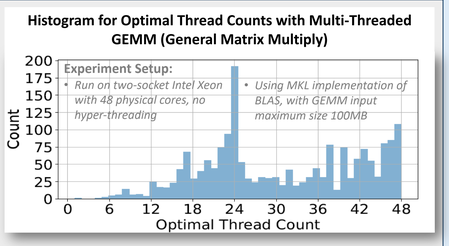
BLAS (Basic Linear Algebra Subsystem) **Level 3 operations** include various forms of matrix multiply and linear systems solving, they are essential for scientific computing. We **uses machine learning to optimize the runtime of multi-thread BLAS Level 3 operations**, as an extension to the **Architecture and Data-Structure Aware Linear Algebra (ADSALA)** library to provide **full support over BLAS III operations** [1]. Our method predicts the best number of threads for each operation based on the matrix dimensions and the system architecture and is synergistic to all single-thread optimizations.

We test our method on two HPC platforms with Intel and AMD processors using multi-threaded MKL and BLIS as baseline BLAS implementations. We **achieve average speedups from 1.07 to 2.9** across BLAS III operations, compared to using the maximum number of threads. Our work shows the effectiveness and generality of the ADSALA approach for optimizing BLAS routines on modern multi-core systems.

Motivation

Observations:

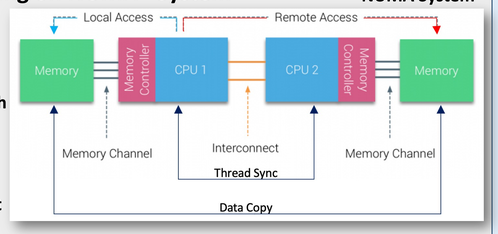
1. **Multi-threaded** BLAS III **do not** choose n_{threads} for **optimal performance.**



2. Simply applying a **fixed** n_{threads} is far from being optimal.
3. We need an algorithm to predict the best n_{threads} with **high accuracy** and **high speed.**

BLAS Modelling on NUMA* System

NUMA System



1. BLAS modelling is complex with **both computation/memory complexity** and **startup effect**
2. Modelling on **NUMA** systems is complex with **asymmetric memory access**
3. Fitting for **different machines** is **time-consuming**

Machine Learning Design

Predict the Best n_{threads}:

1. **Predict** the **running time's** for given input ($\boldsymbol{x}$) with $n_{\text{threads}} < n_{\max}$:
$$\hat{T}_{\text{BLAS}}(\boldsymbol{x}, n_{\text{threads}}) = \text{model}(\boldsymbol{x}, n_{\text{threads}})$$
2. **Select** the n_{threads} with the **least running time**:
$$\hat{n}_{\text{threads}} = \underset{n_{\text{threads}}}{\text{argmin}}\left(\hat{T}_{\text{BLAS}}(\boldsymbol{x}, n_{\text{threads}})\right)$$

Auto-Select the Best model:

We select the **largest** speedup (estimated) in **from model candidates**:

$$s = \frac{T_{\text{default}}}{T_{\hat{n}_{\text{threads}}} + T_{\text{model\_evaluation}}},$$

which helps choose the best balance between:

1. Low Error
2. Fast Model Inference

Data Gathering

1.Parameter Space:

1.Matrix dimensions: $m, (k, n)$
2.Number of threads to use: n_{threads}
3.Target: running duration

2. Quasi-random sampling enables **even-coverage** across the parameter space, with an **upper limit on memory size** for matrices -> see heatmap

Feature Engineering

1. Feature Extraction:

- Size of matrices and FLOPS needed
- All above with multi-thread speedup

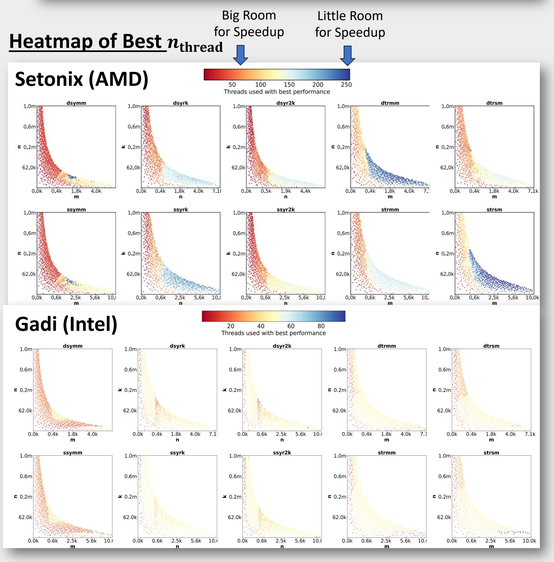
2. Feature Selection:

Highly correlated features are considered redundant and are removed (90%)

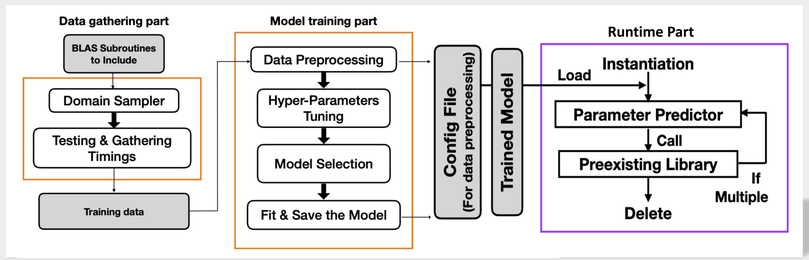
Model Candidates

Model Catagories	Models	Parametric	Good with Data Imbalance	Data Size Requirement
Linear Models	Linear Regression	Yes	No	Medium
	ElasticNet			Medium
	Bayesian Regression			Small
Tree Based Models	Decision Tree	No	Yes	Medium
	XGBoost			
	AdaBoost			
	Random Forest			
	LightGBM			
Other Models	SVM Regressor	No	No	Small
	KNN Regressor			Medium

Heatmap of Best n_{thread}



Software Design

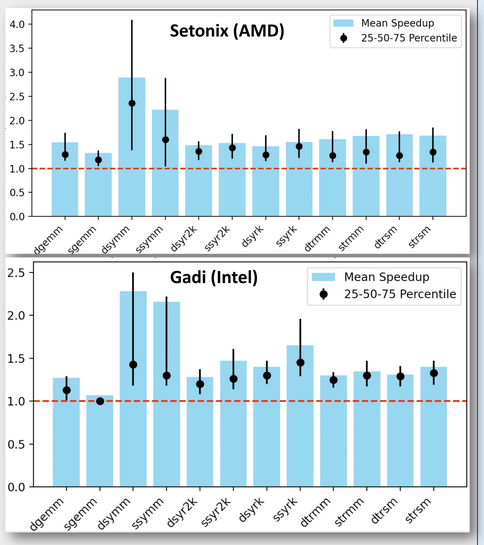


Some Runtime Features:

1. We do **runtime inference** of the ML model; **2.** The model is maintained in memory for **multiple inferences**

Results Summary

1. **Speedup** across **all BLAS III operations**
2. **Average speedup** from **1.07 to 2.89**
3. Better speedup is observed on **Setonix**
4. **dsymm** shows best speedup, **sgemm** shows least speedup



Training Dataset Size:

Around **1100** data points for each BLAS operation.

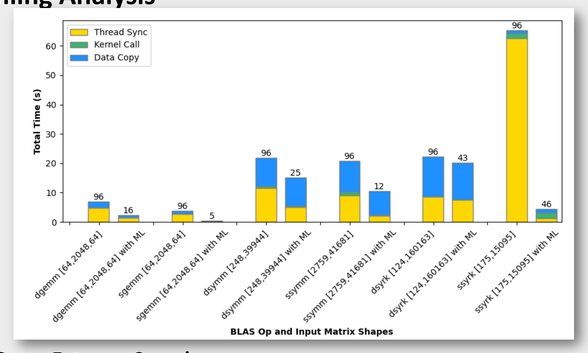
Data Gathering Cost:

Around **6 hours** with **15 nodes** on both platforms

Model Training and Selection Cost:

Around **4 hours on one node** on both platforms

Profiling Analysis



Some Extreme Speedups:

- Outstanding Speedups are obtained for some **slim input matrices**
- Time Breakdown shows time reduction in **all three components**
- **Data copy** yields the **largest** fraction of time reduction

[1] Xia, Y., De La Pierre, M., Barnard, A.S. and Barca, G.M.J., 2023, May. A Machine Learning Approach Towards Runtime Optimisation of Matrix Multiplication. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (pp. 524-534). IEEE.