Optimizing Built-To-Order BLAS while Considering Matrix Order

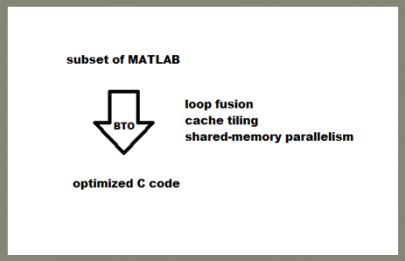
Robert Crimi[1], Thomas Nelson[1], Elizabeth Jessup[1], Jeremy Siek[2]
University of Colorado Boulder[1], Indiana University[2]

Abstract

Many applications of linear algebra use sequences of Basic Linear Algebra Subprograms (BLAS). Computer scientists apply tuning techniques to improve data locality to create efficient implementations of those routines enabling scientists to build high-performance software at reduced cost. However, because the BLAS are individually optimized, users may not see the performance they desire when using a sequence of BLAS. We are developing the Build To Order (BTO) compiler for tuning such sequences. One challenge of optimizing linear algebra is that each matrix order may require a different tuning technique. In this poster, we present an algorithm to efficiently choose code variants for a range of matrix orders.

BTO

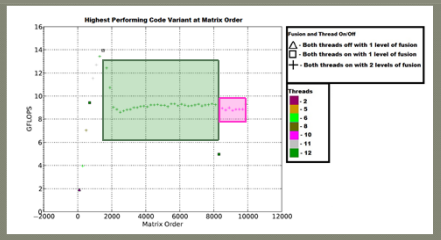
BTO performs a variety of optimization techniques for improving the performance of linear algebra computations. An input kernel and a matrix size must be specified for BTO to optimize.



- The effectiveness of these tuning techniques depends on matrix order.
- We would like to be able to specify a range of matrix orders for BTO to optimize over

Matrix Order is Critical

- Below shows the top performing code variant BTO finds for each matrix order. We observe ranges (highlighted in boxes) in which a specific code variant performs the best for $y = AA^Tx$. All results are for matrix orders 100 to 10,000.



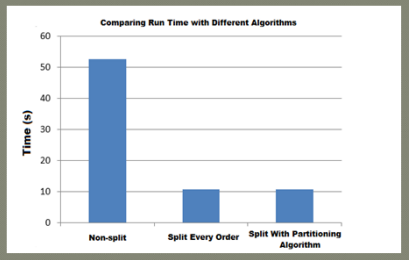
- To achieve the performance BTO can produce, an algorithm would need to create specialized code for each matrix order.
- These data imply that only a small number of code variants can provide nearly the same performance.

A Matrix Order Partitioning Algorithm

- Run BTO on desired matrix range; collect performance data
- When same code variant performs in the top three for consecutive matrix orders, use that variant over the entire range.
- Generate C code with partitions where these code variants change.

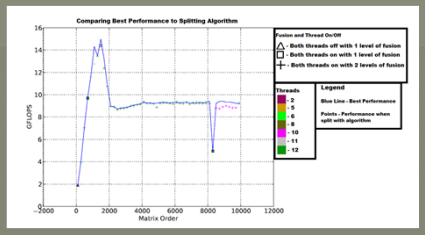
Advantages:

- Code less complex than for splitting at every order
- Lower runtime than not splitting



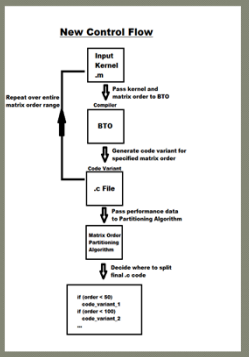
Performance with Algorithm

Matrix Order Partitioning Algorithm is no greater than (1%) less efficient than best performance for $y = AA^Tx$.



Project Summary

- This figure shows the updated control flow for optimizing linear algebra code.



Research made possible by NSF grant CCF-0830458