Introduction to My Current and Future Work

Krzysztof Drewniak

Visiting student at HPAC

4 Oct 2017

Outline

About Me

Why I'm Here

- Bachelor's student in Computer Science and (Pure) Mathematics at the University of Texas at Austin
- Degree almost complete — need to write and defend research thesis to graduate with honors
- Invited by Prof. Bientinesi to work on Linnea project
- Will be staying at RWTH from September to mid-January

Research Background

- Computational epidemiology with Dr. Armin R. Mikler at U. of North Texas while in early college high school
  - Worked on how to reduce epidemic spread
  - Vaccination strategies based on centrality (in the graph that served as a model)
- Joined Dr. Robert Van De Giejn's group at UT Austin last year
  - Was initially interested in FLAME
  - Created an algorithm for $D+ = ABC$
- Interests are in the intersection of program correctness and low-level systems programming

Industry Background

Last four summers were spent doing internships:

- Google Summer of Code — adding more thorough Unicode support to a Lisp implementation
- WhatsApp — Server-side media handling improvements and some encryption things
- TrueCar — Machine learning (that is, database scouring) for fake-order detection
- Microsoft, Bing Maps — Deep learning to improve maps suggestions

gemm3()

Problem

- Want to compute $D+ = ABC$
- If we only have `gemm()` (or `trmm()` ...), need to break into

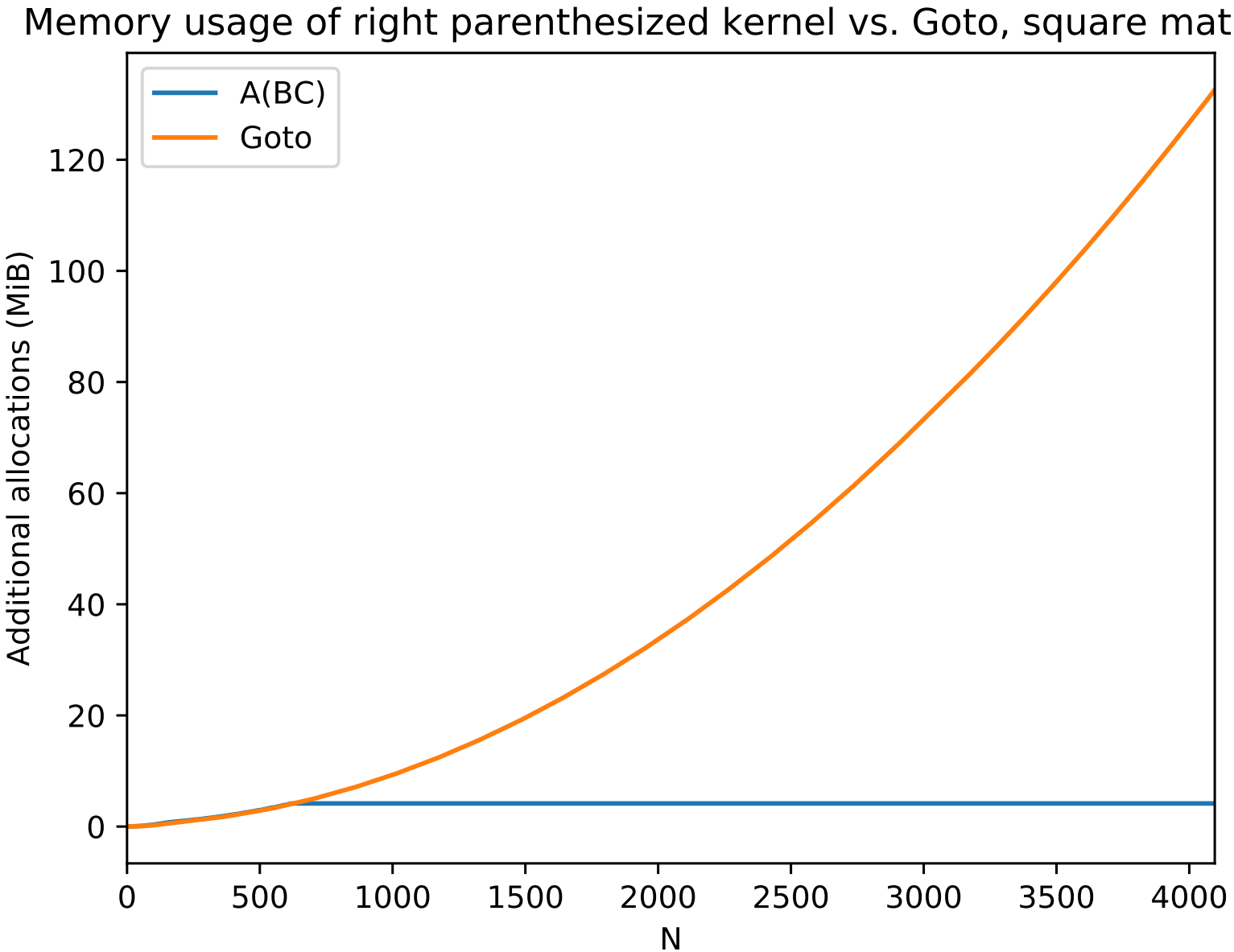
$$T = BC$$
$$D+ = AT$$

- T is often large, causing memory consumption issues
- Writing to/from memory can also be costly

Solution

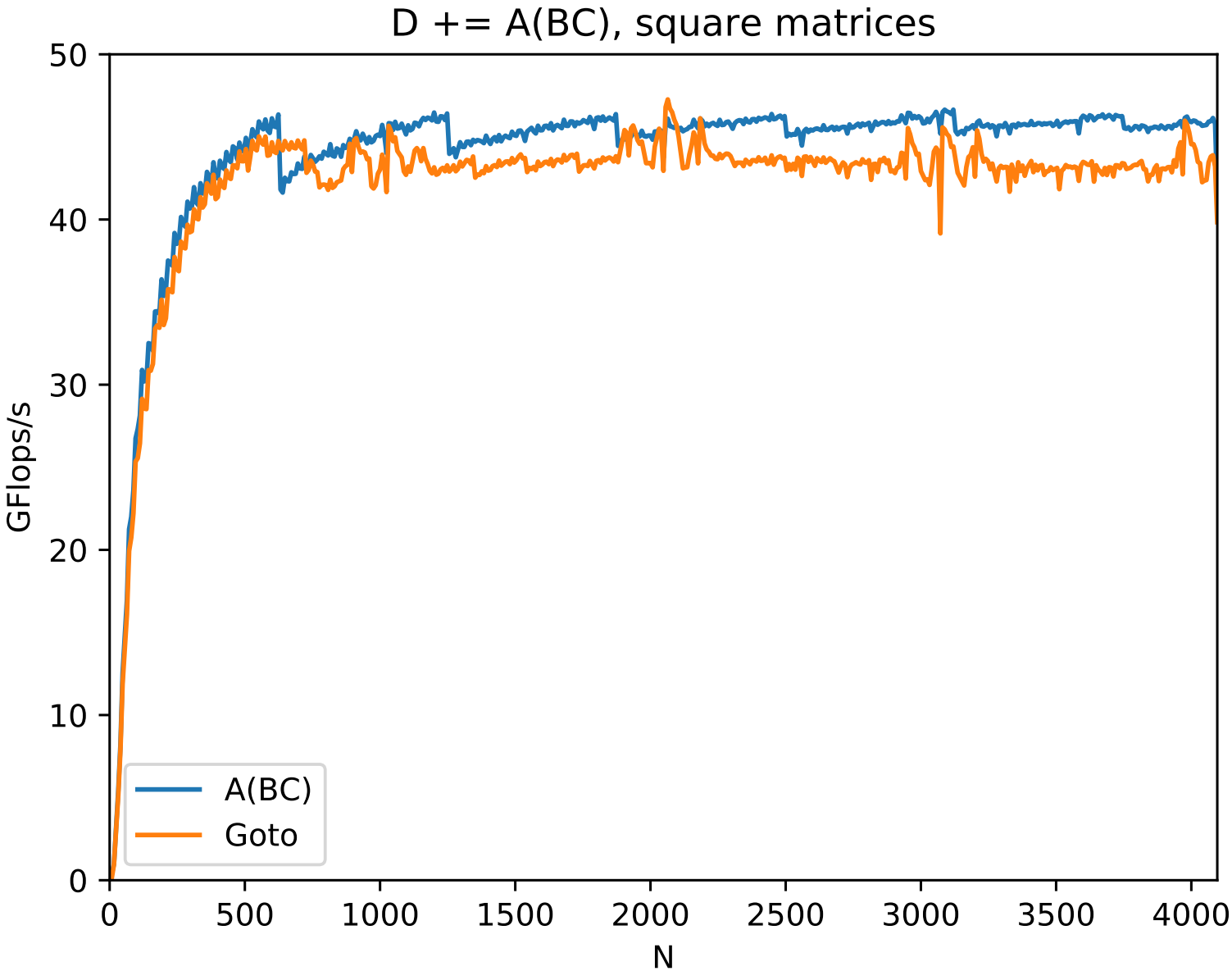
- New algorithm to compute $D+ = A(BC)$ in constant memory
- Uses two different matrix multiply algorithms
- Inner algorithm is called to compute at most cache-sized blocks of (BC)
- Reduces memory impact by never storing all of (BC)
- Also increases performance by keeping intermediates in cache

Memory results



Performance results

4.7% GFlops/s higher for square matrices (5.1% in left-parenthesized case)

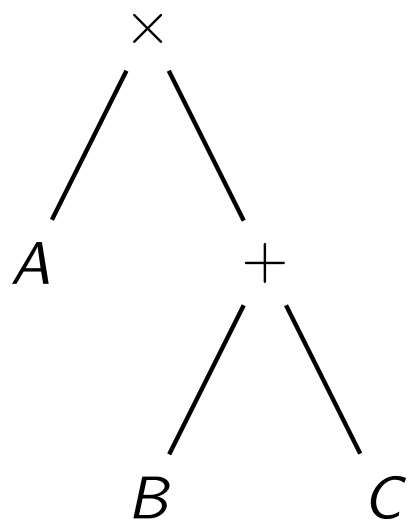


Simplifying Expressions in Linnea

Overview of Linnea

- Compiles linear algebra expressions into series of BLAS/LAPACK calls
- Generates code which significantly outperforms other systems
- Operates on symbolic expressions

$$A(B+C) \quad \equiv$$



- Simplifying expressions is central to Linnea
  Ex. $ABB^{-1} \rightarrow AI \rightarrow A$

Simplifying expressions in Linnea

- Linnea needs to simplify expressions in order to check their equivalence and correctly apply kernels
- The simplification rules are based on linear algebra identities
  - $\alpha A + \beta A \rightarrow (\alpha + \beta)A$
  - $A^T \rightarrow A$ (only when A is symmetric)
- Simplification code is currently handwritten, requiring much time and effort

Goal

- Find a way to autogenerate the simplification module for Linnea
- Benefits:
  - Requires only knowledge of the underlying identities
  - Less hand-written code, so fewer bugs
  - Generalizability — same code could be used for a tensor compiler or similar
  - Potentially improved performance

Generating term rewriting systems

- Term rewriting systems are a set of rewrite rules that operate on algebraic expressions
- Pattern matching identifies applicable rules
- Algorithms exist to turn sets of identities into rewrite systems that produce a normal form
  - Main example: Knuth-Bendix algorithm
- Do not always succeed or even terminate
- Not known to handle constraints (like $A = A^T$ if A is symmetric)

Plan

- Adapt Knuth-Bendix (or similar) algorithms for this purpose and implement them
  - If this fails, investigate other potential techniques for this problem
- Translate known identities into rewrite system
- Generate efficient simplification code based on rewrite rules