How to Write Fast Numerical Code

Spring 2012

Lecture 9

Instructor: Markus Püschel

TAs: Georg Ofenbeck & Daniele Spampinato

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Today

- Linear algebra software: history, LAPACK and BLAS
- Blocking (BLAS 3): key to performance
- How to make MMM fast: ATLAS, model-based ATLAS

Linear Algebra Algorithms: Examples

- Solving systems of linear equations
- Eigenvalue problems
- Singular value decomposition
- LU/Cholesky/QR/… decompositions
- … and many others

- Make up most of the numerical computation across disciplines (sciences, computer science, engineering)
- Efficient software is extremely relevant

The Path to LAPACK

- **EISPACK and LINPACK**
  - Libraries for linear algebra algorithms
  - Developed in the early 70s
  - Jack Dongarra, Jim Bunch, Cleve Moler, Pete Stewart, …
  - LINPACK still used as benchmark for the TOP500 (Wiki) list of most powerful supercomputers
- **Problem:**
  - Implementation “vector-based,” i.e., little locality in data access
  - Low performance on computers with deep memory hierarchy
  - Became apparent in the 80s
- **Solution: LAPACK**
  - Reimplement the algorithms “block-based,” i.e., with locality
  - Developed late 1980s, early 1990s
  - Jim Demmel, Jack Dongarra et al.

Matlab

- **Invented in the late 70s by Cleve Moler**
- **Commercialized (MathWorks) in 84**
- **Motivation: Make LINPACK, EISPACK easy to use**
- **Matlab uses LAPACK and other libraries but can only call it *if you operate with matrices and vectors and do not write your own loops***
  - A*B (calls MMM routine)
  - A\b (calls linear system solver)

LAPACK and BLAS

- **Basic Idea:**

- **Basic Linear Algebra Subroutines (BLAS, list)**
  - BLAS 1: vector-vector operations (e.g., vector sum)
  - BLAS 2: matrix-vector operations (e.g., matrix-vector product)
  - BLAS 3: matrix-matrix operations (e.g., MMM)
- **LAPACK implemented on top of BLAS**
  - Using BLAS 3 as much as possible

$$I(n) =$$
$$O(1)$$
$$O(1)$$
$$O(\sqrt{C})$$

cache size

Why is BLAS3 so important?

- **Using BLAS3 (instead of BLAS 1 or 2) in LAPACK**
  = blocking
  = high operational intensity I
  = high performance
- **Remember last time (blocking MMM):**

$$I(n) =$$

$$O(1)$$

$$O(\sqrt{C})$$

Today

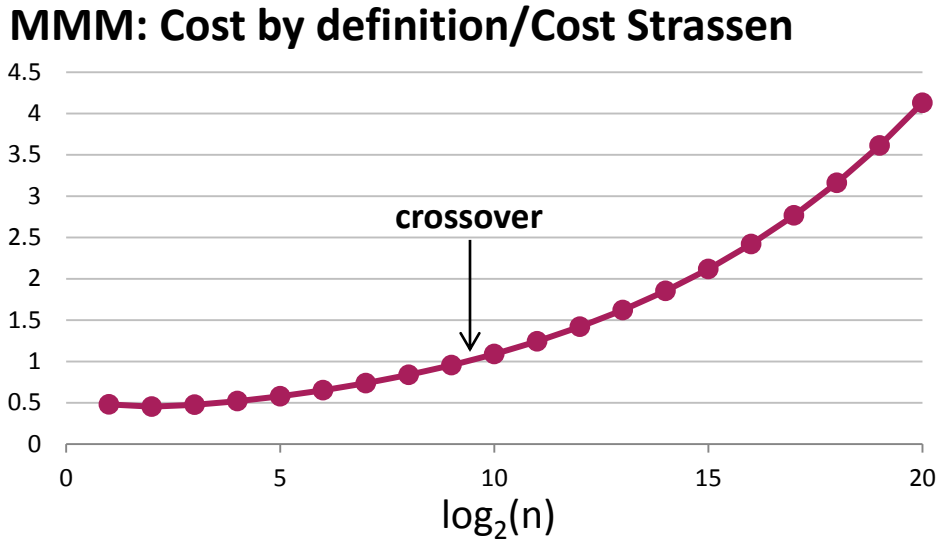
- Linear algebra software: history, LAPACK and BLAS
- Blocking (BLAS 3): key to performance
- **How to make MMM fast: ATLAS, model-based ATLAS**

MMM: Complexity?

- **Usually computed as C = AB + C**
- **Cost as computed before**
  - n^3 multiplications + n^3 additions = $2n^3$ floating point operations
  - = $O(n^3)$ runtime
- **Blocking**
  - Increases locality (see previous example)
  - Does not decrease cost
- **Can we reduce the op count?**

Strassen's Algorithm

- **Strassen, V. "Gaussian Elimination is Not Optimal," *Numerische Mathematik* 13, 354-356, 1969**
  Until then, MMM was thought to be $\Theta(n^3)$
- **Recurrence $T(n) = 7T(n/2) + O(n^2)$; hence $O(n^{\log_2(7)}) \approx O(n^{2.808})$**
- **Crossover point, in terms of cost: below n=1000, but …**
  - Structure more complex → performance crossover much later
  - Numerical stability inferior
- **Can we reduce more?**



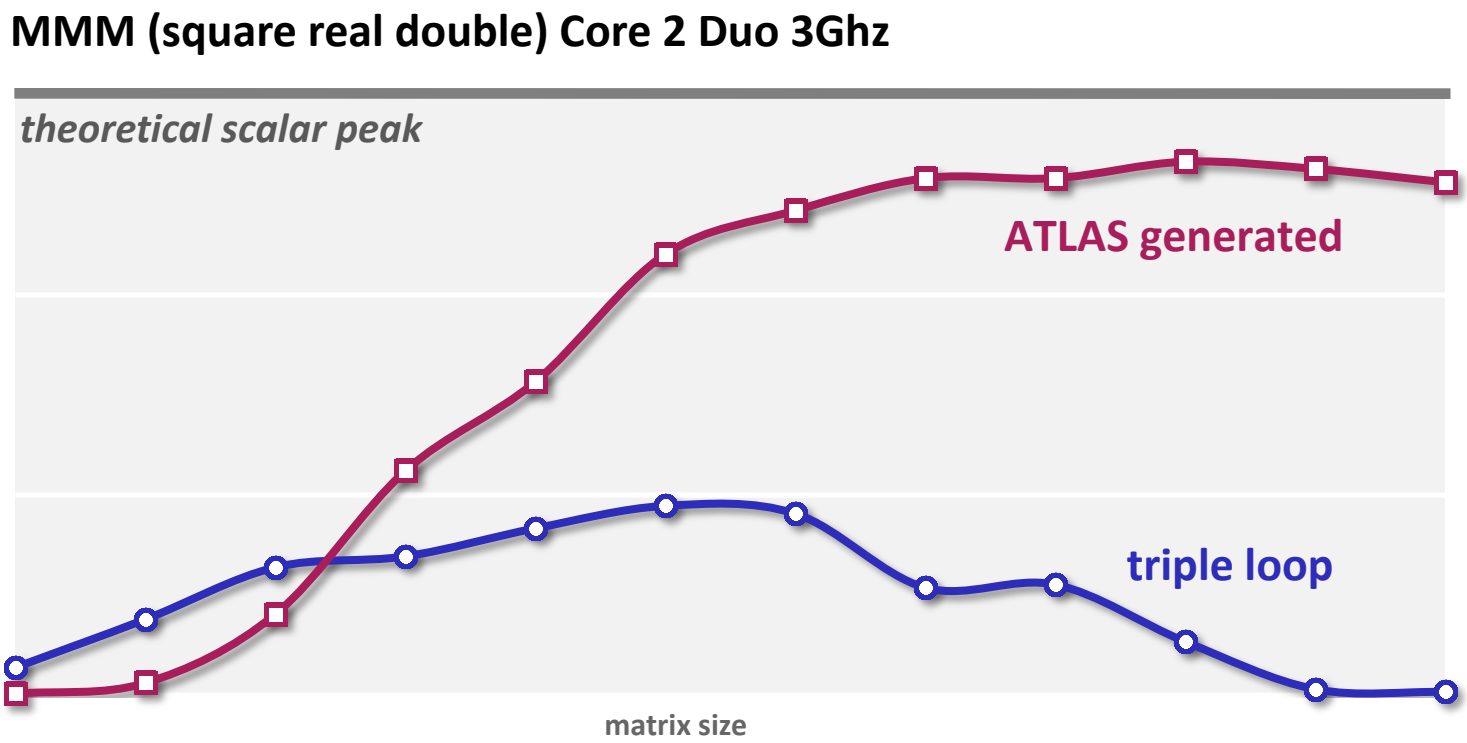
MMM Complexity: What is known

- **Coppersmith, D. and Winograd, S. "Matrix Multiplication via Arithmetic Programming," *J. Symb. Comput.* 9, 251-280, 1990**
- **MMM is $O(n^{2.376})$**

- **MMM is obviously $\Omega(n^2)$**
- **It could well be close to $\Theta(n^2)$**
- **Compare this to matrix-vector multiplication:**
  - Known to be $\Theta(n^2)$ (Winograd), i.e., boring

- **Practically all code out there uses $2n^3$ flops**

MMM: Memory Hierarchy Optimization



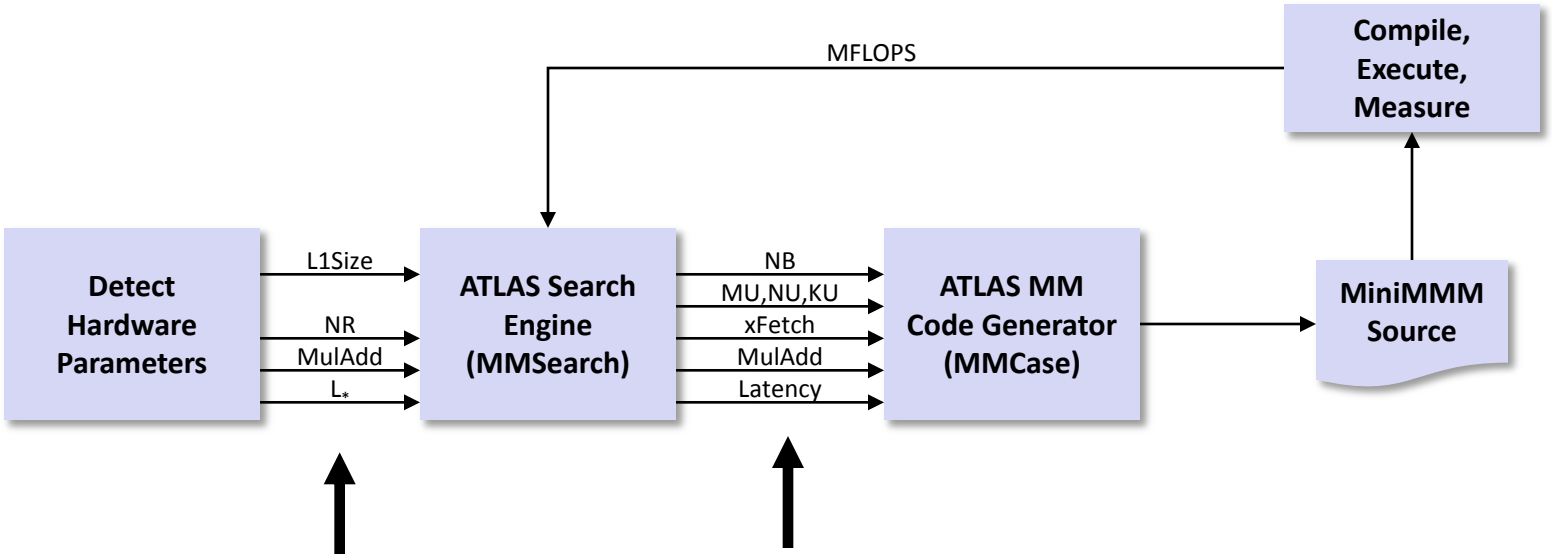
- Intel compiler icc –O2
- Huge performance difference for large sizes
- Great case study to learn memory hierarchy optimization

ATLAS

- BLAS program generator and library (web, successor of PhiPAC)
- Idea: automatic porting

- People can also contribute handwritten code
- The generator uses empirical search over implementation alternatives to find the fastest implementation
  no vectorization or parallelization: so not really used anymore
- We focus on BLAS 3 MMM
- Search only over cost $2n^3$ algorithms
  (cost equal to triple loop)

ATLAS Architecture



Search parameters:

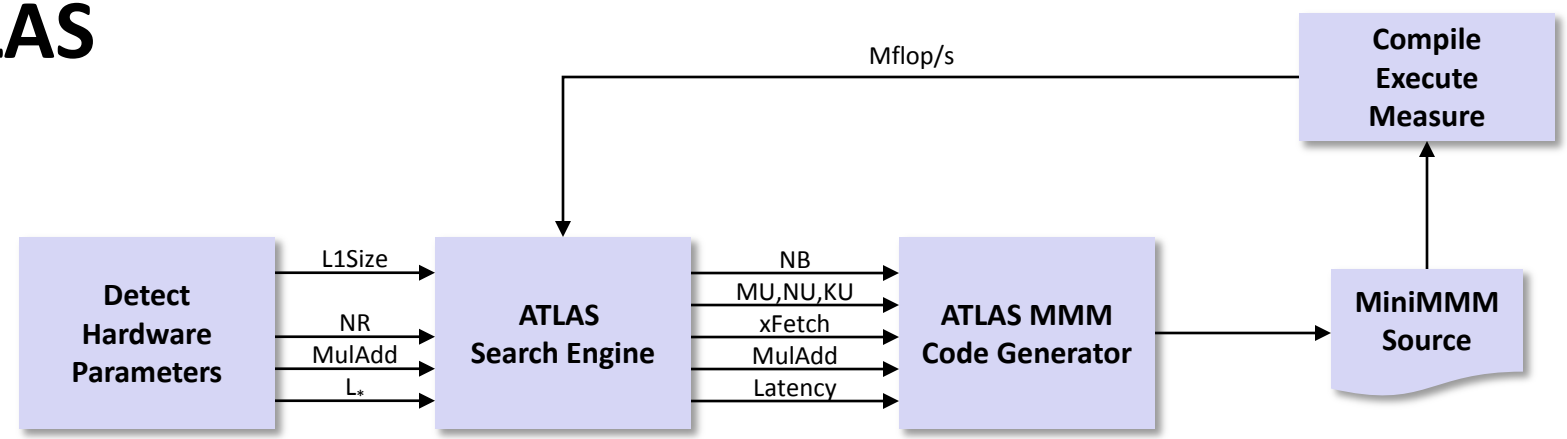
- **for example blocking size**
- **span search space**
- **specify code**
- **found by orthogonal line search**

Hardware parameters:

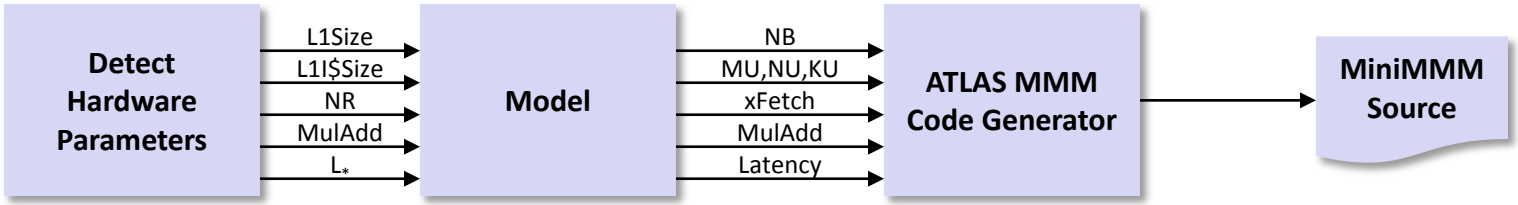
- **L1Size: size of L1 data cache**
- **NR: number of registers**
- **MulAdd: fused multiply-add available?**
- **L_* : latency of FP multiplication**

source: Pingali, Yotov, Cornell U.

ATLAS



Model-Based ATLAS



- Search for parameters replaced by model to compute them
- More hardware parameters needed

source: Pingali, Yotov, Cornell U.

Optimizing MMM

- **Blackboard**
- **References:**

  "Automated Empirical Optimization of Software and the ATLAS project" by R. Clint Whaley, Antoine Petitet and Jack Dongarra. *Parallel Computing*, 27(1-2):3-35, 2001

  K. Yotov, X. Li, G. Ren, M. Garzaran, D. Padua, K. Pingali, P. Stodghill, Is Search Really Necessary to Generate High-Performance BLAS?, Proceedings of the IEEE, 93(2), pp. 358–386, 2005.

  Our presentation is based on this paper