How to Write Fast Code

18-645, spring 2008

6th Lecture, Feb. 4th

Instructor: Markus Püschel

TAs: Srinivas Chellappa (Vas) and Frédéric de Mesmay (Fred)

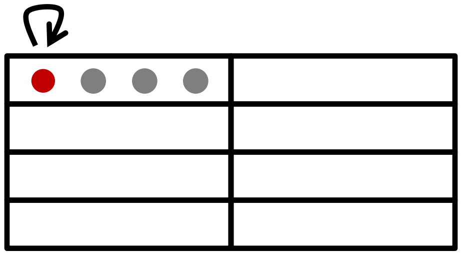
Technicalities

- Research project
- **First steps:**
  - Precise problem statement
  - Correct implementation (create verification environment for future use)
  - Analyze arithmetic cost
  - Measure runtime and create a performance plot
  - If algorithm consists of several steps: identify bottleneck(s) w.r.t. both cost and runtime

Temporal and Spatial Locality

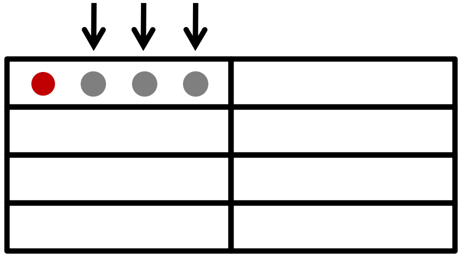
- Properties of a program
- **Temporal locality:** Data that is referenced is likely to be referenced again in the near future

  Promotes data reuse:



- **Spatial locality:** If data is referenced, data in proximity (address) is likely to be referenced in the near future

  Promotes neighbor use:



- Exists because: 1) this is how humans think; 2) structure of numerical algorithms
- History of locality

Today

- Linear algebra software: history, LAPACK and BLAS
- Blocking: key to performance
- MMM
- ATLAS: MMM program generator

Today

- **Linear algebra software: history, LAPACK and BLAS**
- Blocking: key to performance
- MMM
- ATLAS: MMM program generator

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., no 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.

LAPACK and BLAS

- **Basic Idea:**

LAPACK	static
BLAS	reimplemented for each platform

- **BLAS = Basic Linear Algebra Subroutines (list)**
  - BLAS1: vector-vector operations (e.g., vector sum)
  - BLAS2: matrix-vector operations (e.g., matrix-vector product)
  - BLAS3: matrix-matrix operations (mainly matrix-matrix product)

- **LAPACK implemented on top of BLAS (web)**
  - as much as possible using block matrix operations (locality) = BLAS 3
  - Implemented in F77 (to enable good compilation)
  - Open source

- **BLAS recreated for each platform to port performance**

Why is BLAS3 so important?

- Explain on blackboard
- Using BLAS3 = blocking
- Motivate blocking

- Blocking (for the memory hierarchy) is the single most important optimization for linear algebra algorithms

- The introduction of multicore processors requires a reimplementation of LAPACK
(just multithreading BLAS is not good enough)

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 (MMM)
  - A\b (solving linear system)

Today

- Linear algebra software: history, LAPACK and BLAS
- Blocking: key to performance
- **MMM**
- ATLAS: MMM program generator

MMM by Definition

- **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 do better?**

Strassen's Algorithm

- Strassen, V. "Gaussian Elimination is Not Optimal." *Numerische Mathematik* 13, 354-356, 1969
  Until then, MMM was thought to be $\Theta(n^3)$
- Check out algorithm at Mathworld
- Recurrence $T(n) = 7T(n/2) + O(n^2)$:
  Multiplies two n x n matrices in $O(n^{\log_2(7)}) \approx O(n^{2.808})$
- Similar to Karatsuba
- Crossover point, in terms of cost: n=654, but …
  - Structure more complex
  - Numerical stability inferior
- Can we do better?

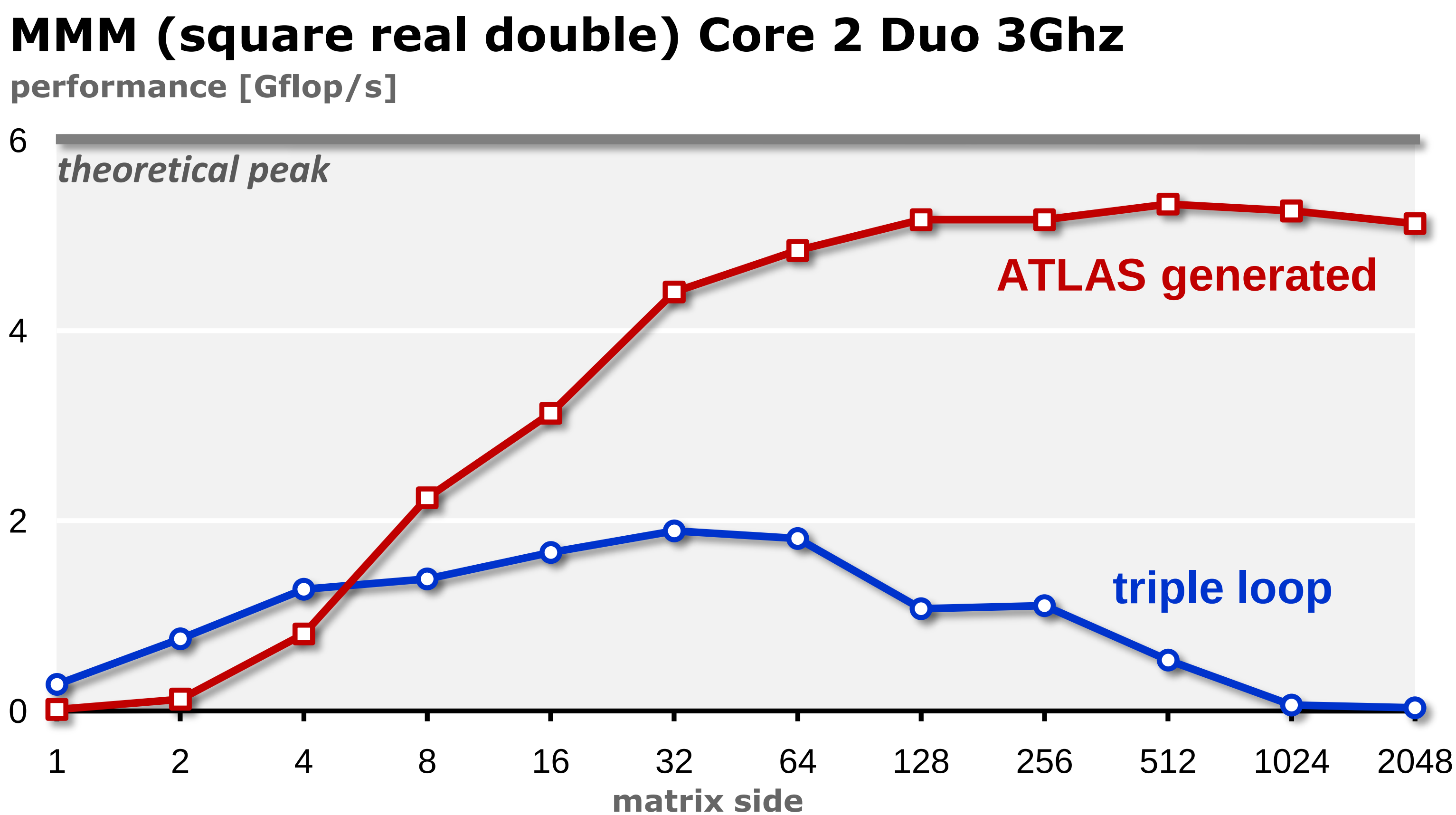
MMM Complexity: What is known

- Coppersmith, D. and Winograd, S. "Matrix Multiplication via Arithmetic Programming." *J. Symb. Comput.* 9, 251-280, 1990
- MMM is $O(n^{2.376})$ and (obviously) $\Omega(n^2)$
- It could well be $\Theta(n^2)$
- Compare this to matrix-vector multiplication, which is $\Theta(n^2)$ (Winograd), i.e., boring
- MMM is the single most important computational kernel in linear algebra (probably in whole numerical computing)

Today

- Linear algebra software: history, LAPACK and BLAS
- Blocking: key to performance
- MMM
- **ATLAS: MMM program generator**

MMM: Memory Hierarchy Optimization

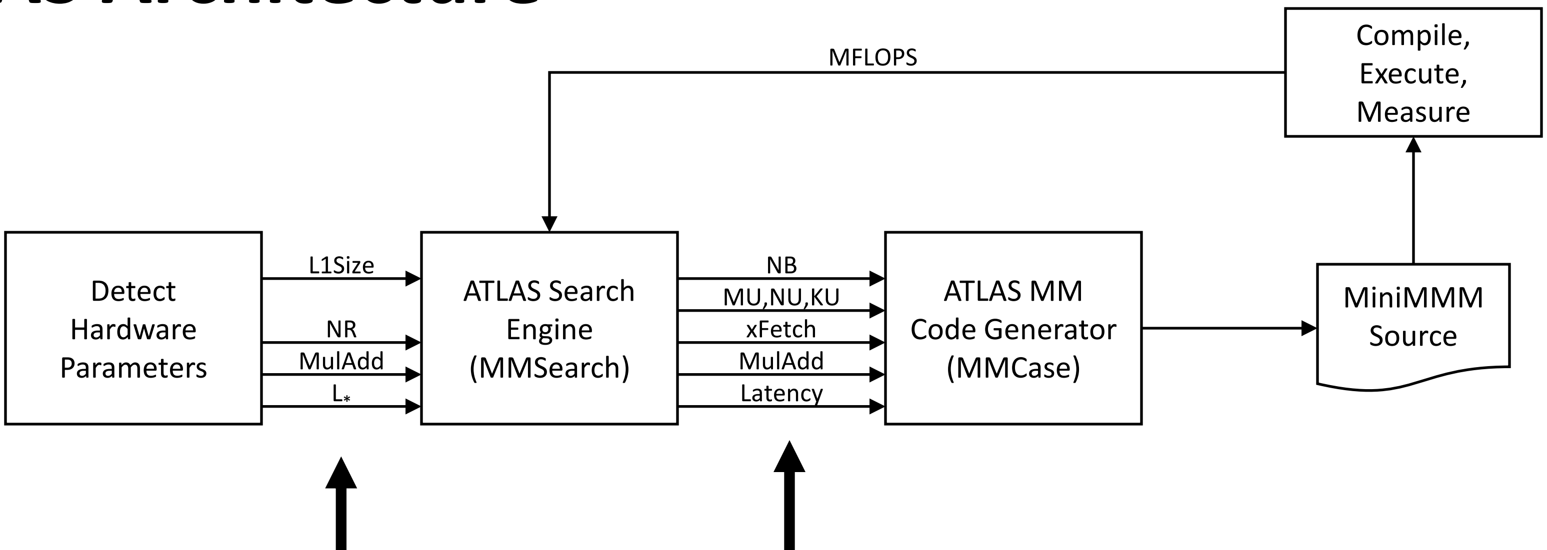


- Intel compiler icc –O2
- Huge performance difference for large sizes
- Great case study to learn memory hierarchy optimization

ATLAS

- Successor of PhiPAC, BLAS program generator (web)
- People can also contribute handwritten code
- The generator uses empirical search over implementation alternatives to find the fastest implementation
  no vectorization or parallelization
- We focus on BLAS3 MMM
- Search only over $2n^3$ algorithms
  (cost equal to triple loop)

ATLAS Architecture



Search parameters:
- **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.

How ATLAS Works

- **Blackboard**