How to Write Fast Numerical Code

Spring 2011

Lecture 8

Instructor: Markus Püschel

TA: Georg Ofenbeck

ETH

Eidgenössische Technische Hochschule Zürich

Swiss Federal Institute of Technology Zurich

Reuse (Inherent Temporal Locality)

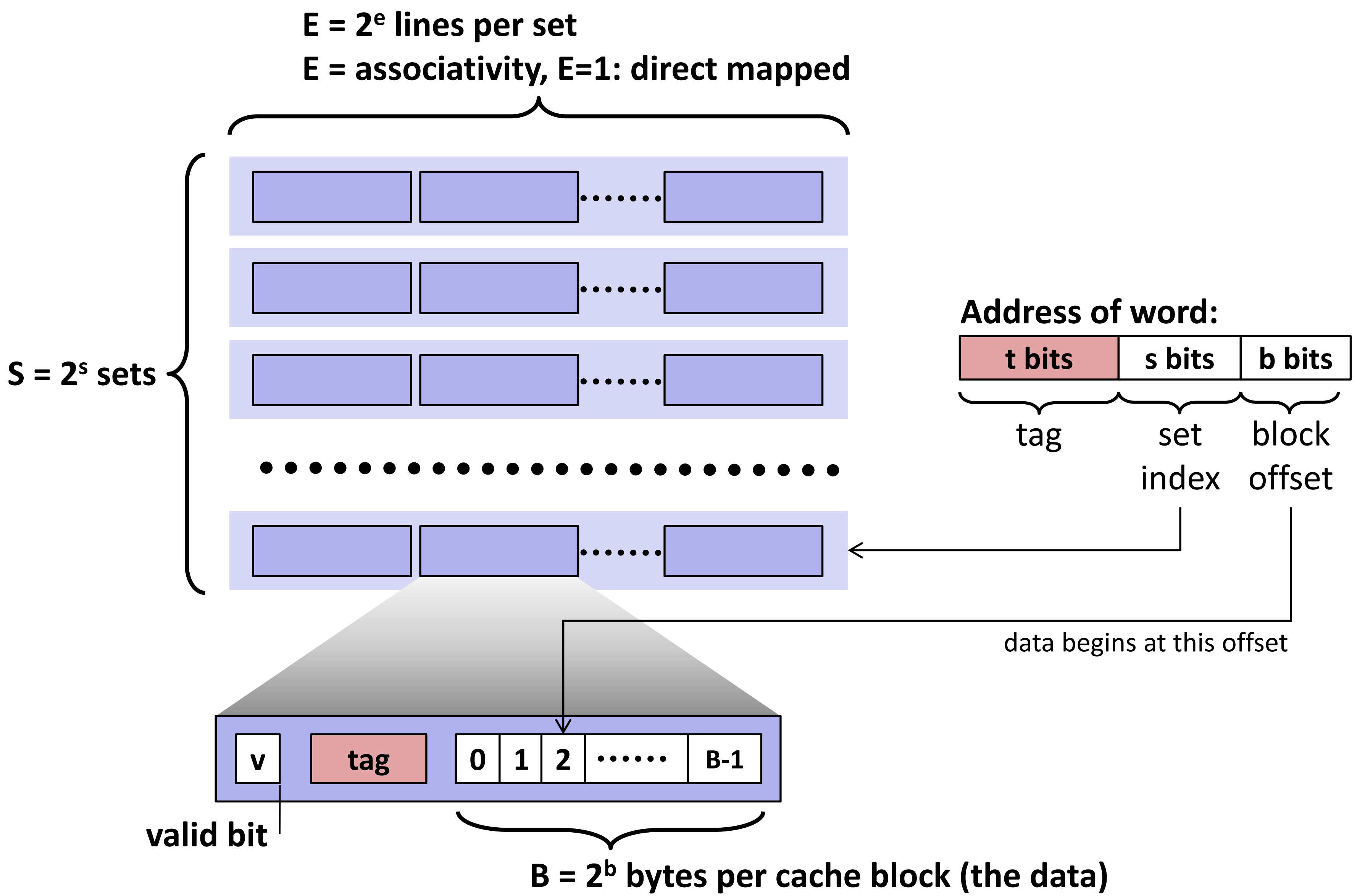
- ***Reuse of an algorithm:***

  $$\frac{\textit{Number of operations}}{\textit{Size of input + size of output data}}$$

  Minimal number of Memory accesses

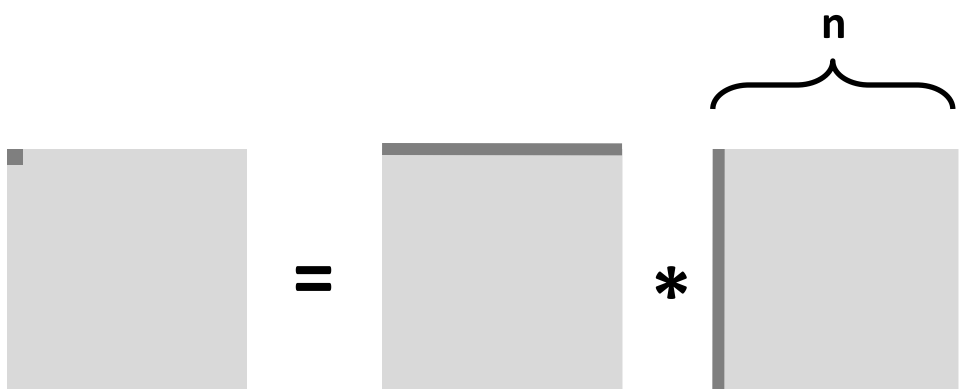
- **Examples:**
  - Matrix multiplication C = AB + C $\frac{2n^3}{3n^2} = \frac{2}{3}n = O(n)$
  - Discrete Fourier transform $\approx \frac{5n\log_2(n)}{2n} = \frac{5}{2}\log_2(n) = O(\log(n))$
  - Adding two vectors x = x+y $\frac{n}{2n} = \frac{1}{2} = O(1)$

Last Time: Caches

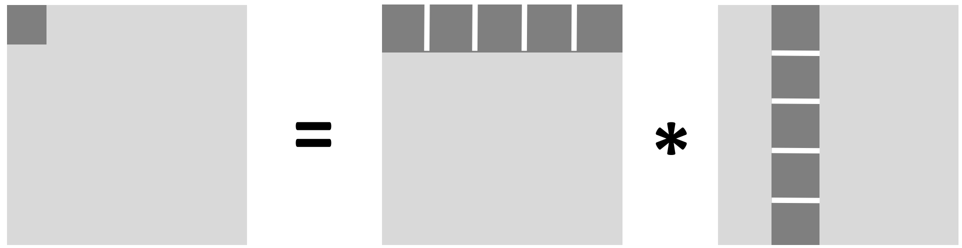


Last Time: Blocking

Cache misses



$(9/8)n^3$



$n^3/(4B)$

Today

- Linear algebra software: LAPACK and BLAS
- 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**

LAPACK and BLAS

- **Basic Idea:**

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

Why is BLAS3 so important?

- **Using BLAS3 = blocking**
- **Reuse O(1) → O(n)**
- **Cache analysis for blocking MMM (blackboard)**

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

- ***Unfortunately:* 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 (calls MMM routine)
  - A\b (calls linear system solver)

Today

- Linear algebra software: history, LAPACK and BLAS
- **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 $\Theta(n^3)$

- **Recurrence $T(n) = 7T(n/2) + O(n^2)$:**
  Multiplies two n x n matrices in $O(n^{\log_2(7)}) \approx O(n^{2.808})$

- **Crossover point, in terms of cost: n=654, but …**
  - Structure more complex → performance crossover much later
  - Numerical stability inferior

- **Can we do better?**

MMM Complexity: What is known

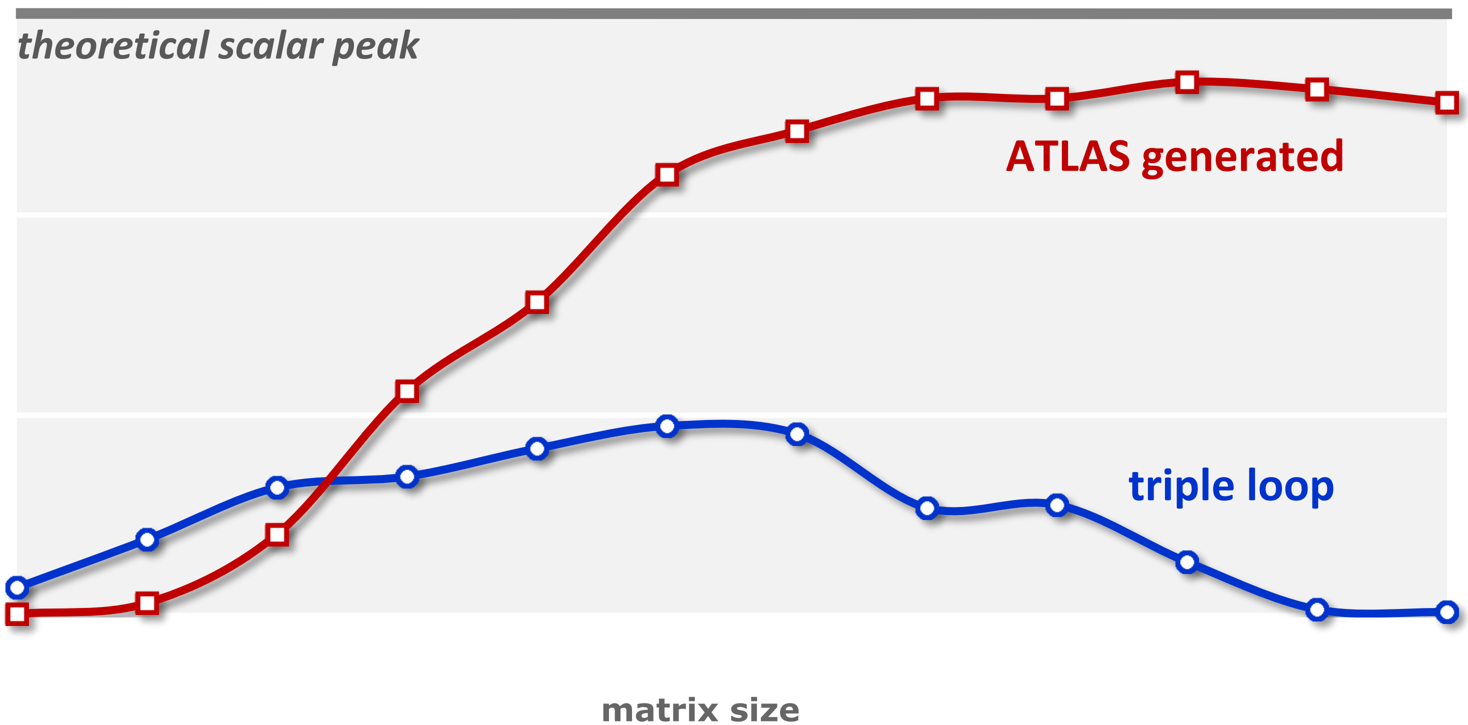
- **Coppersmith, D. and Winograd, S. "Matrix Multiplication via Arithmetic Programming,"** ***J. Symb. Comput.*** **9, 251-280, 1990**
- **MMM is $O(n^{2.376})$**
- **MMM is obviously $\Omega(n^2)$**
- **It could well be $\Theta(n^2)$**
- **Compare this to matrix-vector multiplication:**
  - Known to be $\Theta(n^2)$ (Winograd), i.e., boring

Today

- Linear algebra software: history, LAPACK and BLAS
- MMM
- **ATLAS: MMM program generator**

MMM: Memory Hierarchy Optimization

MMM (square real double) Core 2 Duo 3Ghz



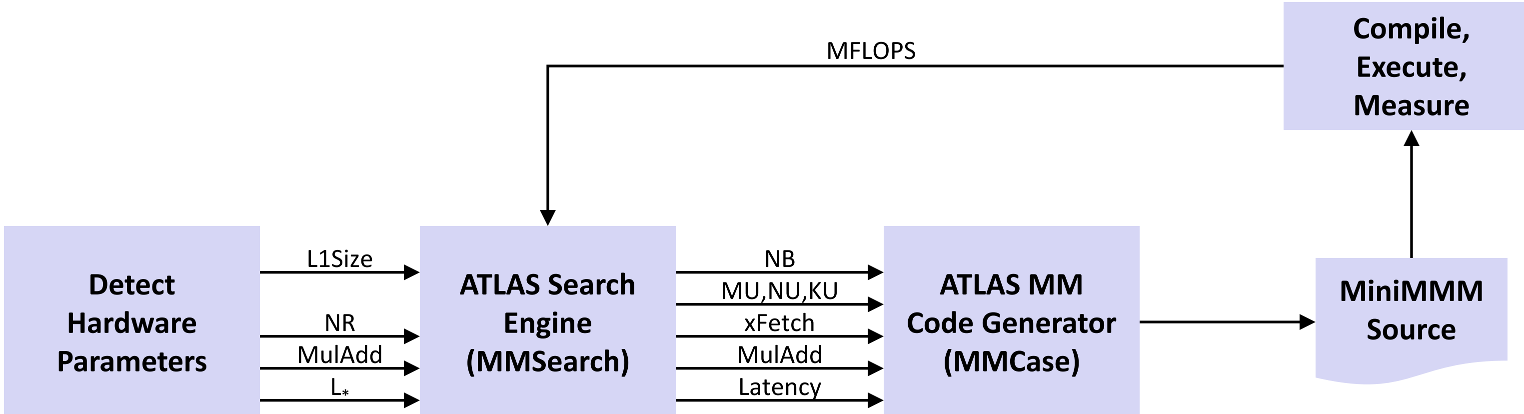
- Intel compiler icc –O2
- Huge performance difference for large sizes
- Great case study to learn memory hierarchy optimization

ATLAS

- Successor of PhiPAC, BLAS program generator (web)
- 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 BLAS3 MMM
- Search only over cost $2n^3$ 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**

- **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. Link.

  Our presentation is based on this paper