Numerical Linear Algebra for Computational Science and Information Engineering

Lecture 17
Introduction to Communication-Avoiding Algorithms

Nicolas Venkovic
nicolas.venkovic@tum.de

Group of Computational Mathematics
School of Computation, Information and Technology
Technical University of Munich

Summer 2025

Outline

1. Introduction
   CS294/Math270 – Demmel and Grigori (2016)
   MIT 6.172 – Shun (2018) 1
2. Matrix-matrix multiplication
   MIT 6.172 – Shun (2018) 7
3. Overview and principles of communication avoidance 18
4. Sparse matrix-vector product (SpMV) 20
5. Block Gram-Schmidt procedures 25
6. s-step iterative solvers 25
7. Matrix power kernels 25
8. Homework problems 25
9. Practice session 25

Introduction

CS294/Math270 – Demmel and Grigori (2016)
MIT 6.172 – Shun (2018)

Cost of algorithm deployment

- Deploying an algorithm has **two costs** measured in **time** (or **energy**):
  - **Arithmetic** (floating-point operations, FLOPs)

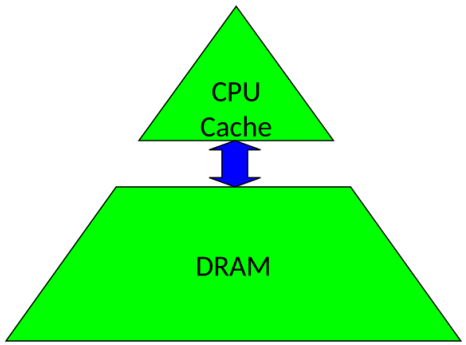
$$\#\text{ of }\textbf{FLOPs} \div (\#\text{ of }\textbf{FLOPs per cycle} \times \#\textbf{ cycles per unit of time})$$

  - **Communication** due to data movement between
    - **levels of a memory hierarchy**:

      sequential part of a program

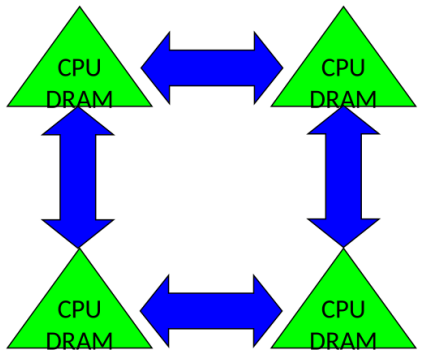
      DRAM, L3 cache, L2 cache, L1 cache, registers

      cache hit, cache miss



    - **processors over a network**:

      parallel part of a program



$$\#\text{ of }\textbf{messages} \times \textbf{latency} + \#\text{ of }\textbf{words} \div \textbf{bandwidth}$$

idle time

Roofline model

- **Algorithm** on given **hardware** characterized by **arithmetic intensity**:

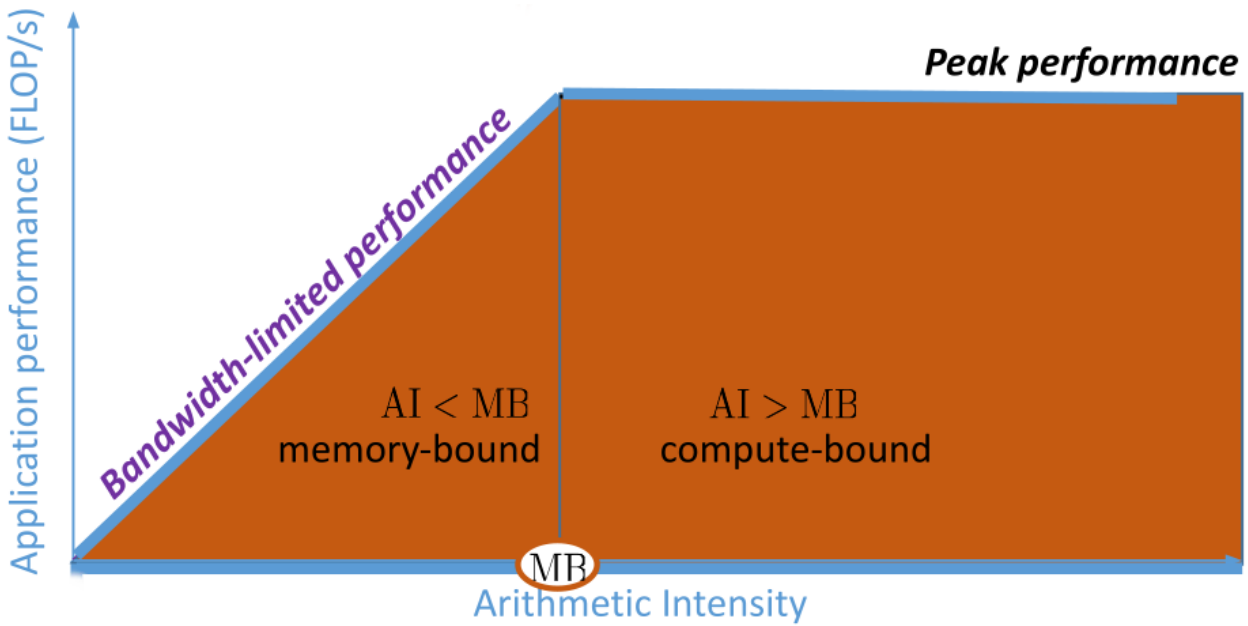
Arithmetic intensity (AI)

Ratio of number of FLOPs over amount of data moved.

- **Hardware** characterized by **machine balance**:

Machine balance (MB)

Ratio of peak floating-point performance over peak memory bandwidth.

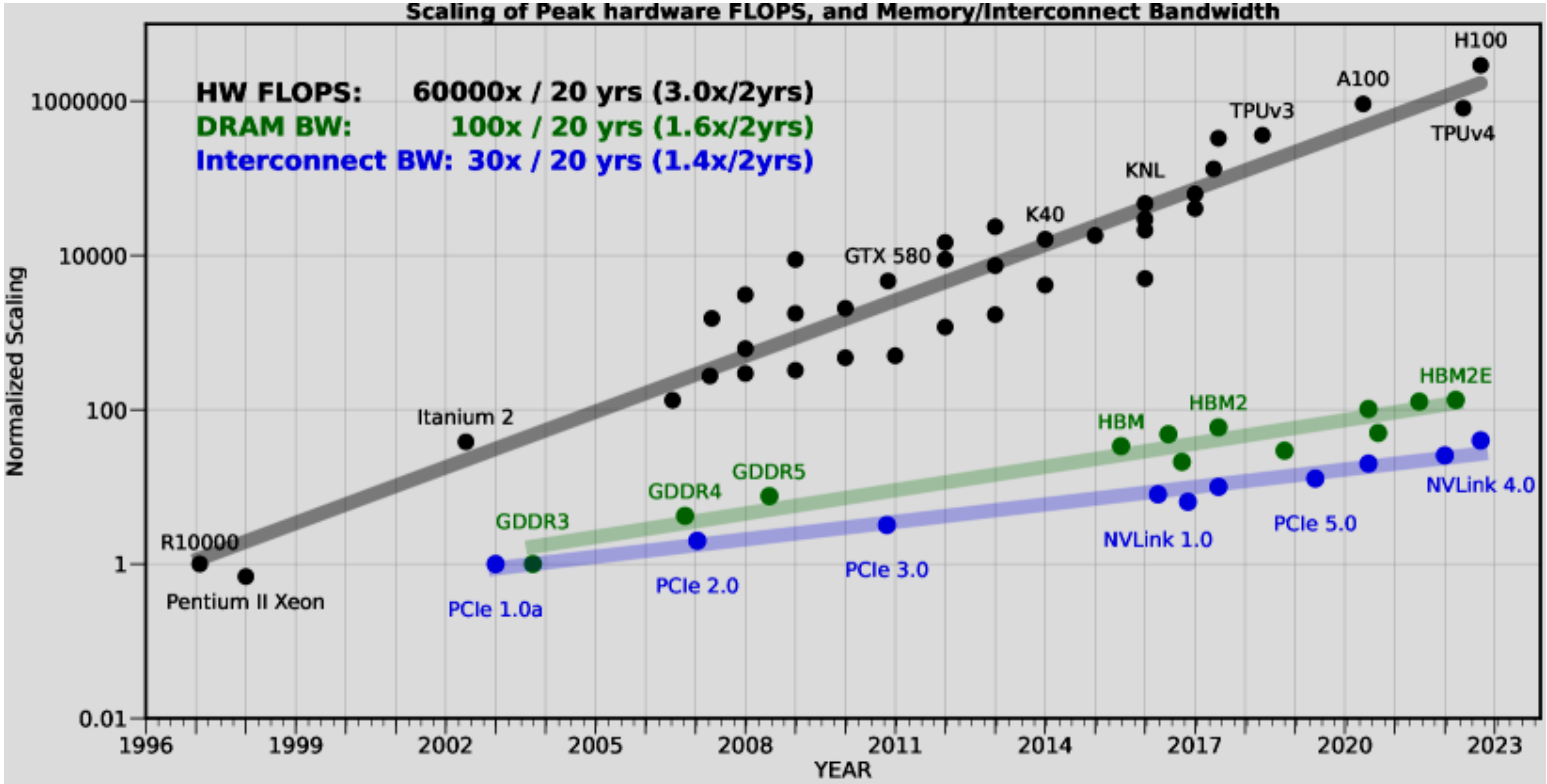


Memory wall

- Cost to move data much greater than cost of arithmetic:

$$\textbf{time per FLOP} \ll \frac{1}{\textbf{bandwidth}} \ll \textbf{latency}$$

- Peak floating-point performance evolves faster than memory bandwidth:

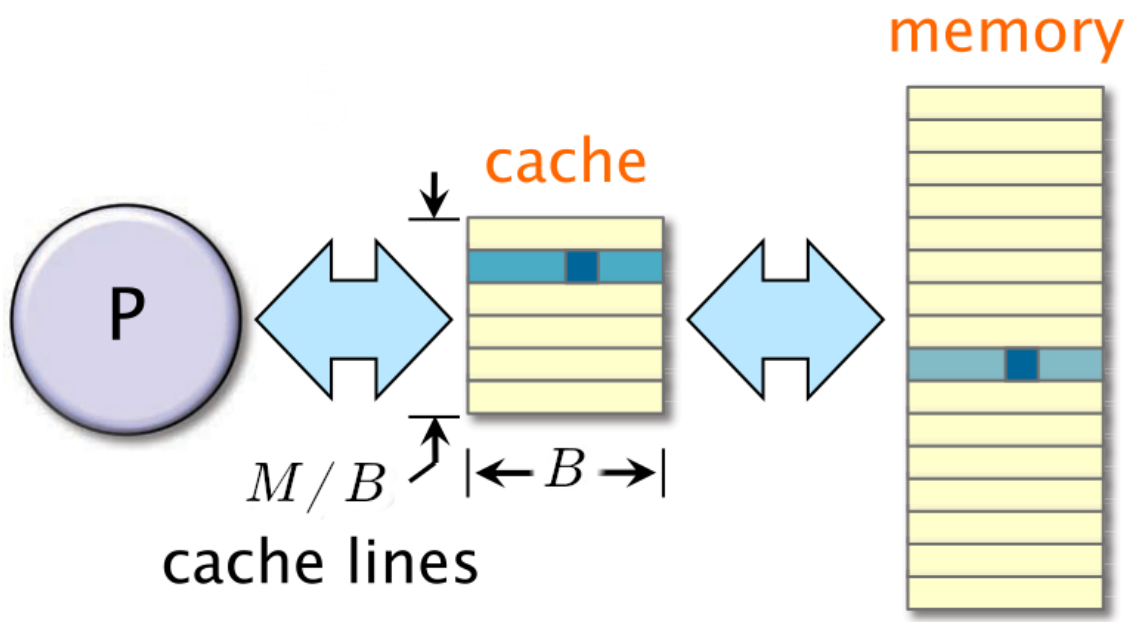


- Relative cost of algorithms due to communication larger every year

Gholami, A., Yao, Z., Kim, S., Hooper, C., Mahoney, M.W., Keutzer, K. (2024). AI and Memory Wall. arXiv:2403.14123v1.

Ideal cache model

- An **ideal cache model** is introduced for the analysis of algorithms:



- Model parameters:
  - Two-level hierarchy
  - Cache size of M bytes
  - Cache line length of B bytes
  - Fully associative cache (cache lines can be stored anywhere in cache)
  - Least-recently used (LRU) cache replacement policy
  - Arbitrary large main memory

Tall caches

- An

Communication lower bounds

- *

Matrix-matrix multiplication

MIT 6.172 – Shun (2018)

Cache misses analysis

- Assume a tall ideal cache

Cache misses analysis, cont'd$_1$

- Matrix-matrix multiplication with row major data:

$$
\begin{aligned}
&\text{for } i = 1, \ldots, n \\
&\quad \text{for } j = 1, \ldots, n \\
&\qquad C[i * n + j] := 0 \\
&\qquad \text{for } k = 1, \ldots, n \\
&\qquad\quad \texttt{C}[i * n + j] := \texttt{C}[i * n + j] + \texttt{A}[i * n + k] * \texttt{B}[k * n + j]
\end{aligned}
$$

- Assume a tall ideal cache
  - Case 1: $n > cM/B$
  - Case 2: $c'M^{1/2} < n < cM/B$
  - Case 3: $n < cM^{1/2}$

Cache misses analysis, cont'd$_2$

- Assume a tall ideal cache

Swapping inner loops

- *

Blocked (tiled) matrix multiplication

- *

Two-level cache

- *

Three-level cache

- *

Recursive matrix multiplication

- *

Recursive implementation

- *

Analysis of work

- *

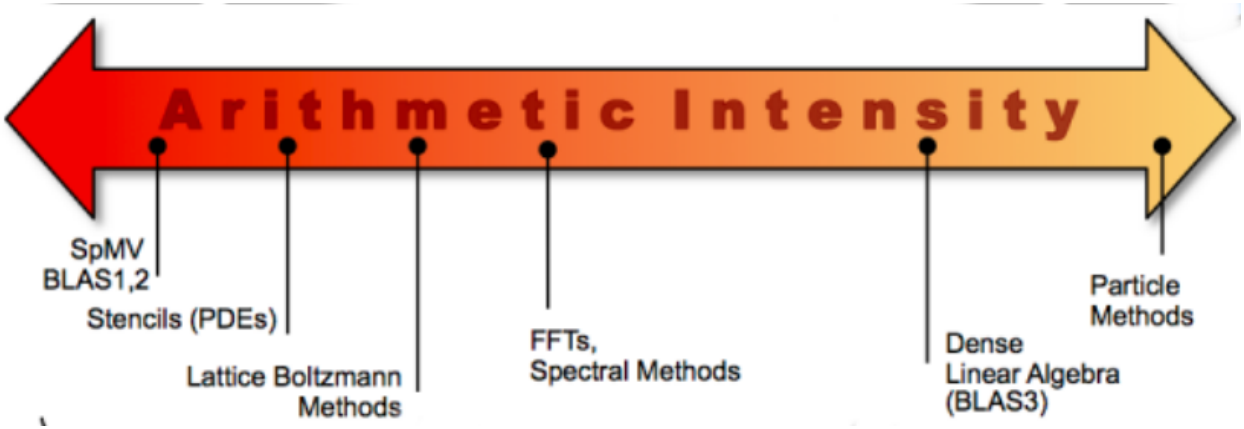
Analysis of cache misses

- *

Overview and principles of communication avoidance

Reducing data movement

- Reformulate algorithms to rely on **more arithmetically intense kernels**:



Sample speedups

- *

Sparse matrix-vector product (SpMV)

Low data locality of SpMV

- Recall the CSR data structure:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & a_{43} & 0 \end{bmatrix}$$

$$\texttt{val} = [a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{33}, a_{34}, a_{43}]$$

$$\texttt{col\_idx} = [1, 2, 3, 1, 2, 3, 4, 3]$$

$$\texttt{row\_start} = [1, 4, 6, 8, 9]$$

with the following SpMV kernel:

for $i = 1, \ldots, n$

  $\texttt{sum} := 0$

  for $\texttt{row\_start}[i], \ldots, \texttt{row\_start}[i+1] - 1$

    $\texttt{sum} := \texttt{sum} + \texttt{val}[j] * \texttt{x}[\texttt{col\_idx}[j]]$

  $\texttt{y}[i] := \texttt{sum}$

Low data locality of SpMV, cont'd

- SpMV kernel for CSR data structures:

$$
\begin{aligned}
&\texttt{sum} := 0 \\
&\text{for } i = 1, \ldots, n \\
&\quad \text{for } \texttt{row\_start}[i], \ldots, \texttt{row\_start}[i+1] - 1 \\
&\quad\quad \texttt{sum} := \texttt{sum} + \texttt{val}[j] * \texttt{x}[\texttt{col\_idx}[j]] \\
&\quad \texttt{y}[i] := \texttt{sum}
\end{aligned}
$$

- **Irregular access** to the components of `x`:
  - **No spatial locality**, **no time locality**.
  - **Every component** of `x` loaded for a single multiply-and-add is **trashed immediatly from register**, i.e., no loop unrolling, no SIMD, ...
  - **Sparsity induces lower arithmetic intensity** (AI) than GEMV.
  - **SpMV kernels** are **memory-bound**, even if `x` fits in cache.
  - **Performance** of SpMV kernels depends on **data structure**, **non-zero structure** and **hardware**.

Register blocking

Tuning of register blocks

MBR

Block Gram-Schmidt procedures

s-step iterative solvers

Matrix power kernels

Homework problems

Practice session