High-performance Matrix Computations

Prof. **Paolo Bientinesi**

pauldj@aices.rwth-aachen.de

Performance

Quality of an algorithm

Metrics?

- Execution time
- Memory usage
- Memory/network accesses
- Accuracy
- Size
- ...

Execution time

- It says how fast an algorithm is
- “Does not measure the memory usage”
- “It might depend on the input”
- “It depends on the processor”
- **It does not measure how well the algorithm takes advantage of a given architecture**

- The quality of an algorithm should be related to the potential of the architecture
- Potential of an architecture?

Theoretical Peak Performance

Theoretical Peak Performance =
#cores * frequency * #flops/cycle

- what is frequency? GHz
- what is a cycle?
  processor can initiate a new operation
- what is a flop?
  floating point operation

- The quality of an algorithm should be related to the potential of the architecture
- Potential of an architecture?

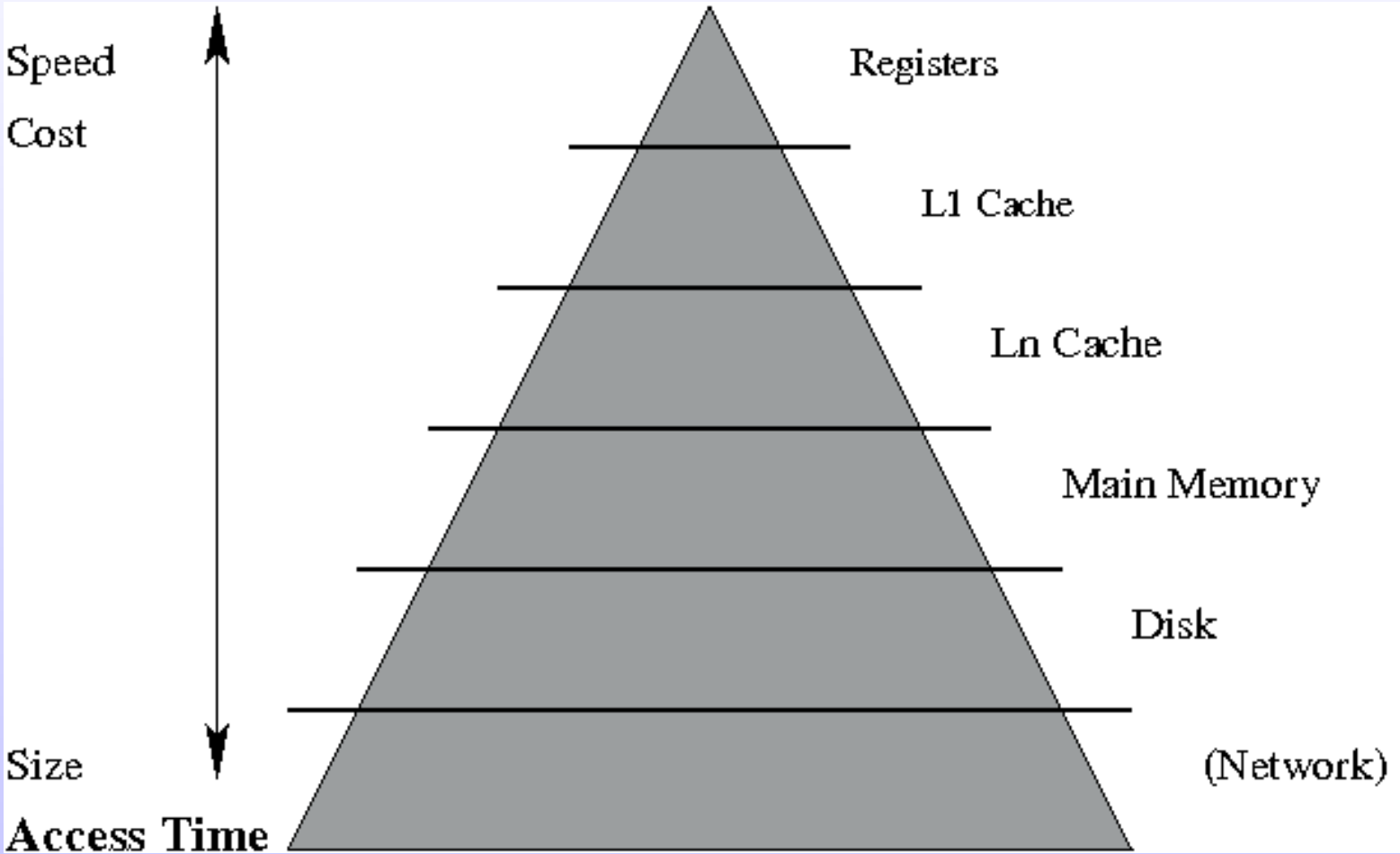
Theoretical Peak Performance

Theoretical Peak Performance =
#cores * frequency * #flops/cycle

- what is frequency? GHz
- what is a cycle?
  processor can initiate a new operation
- what is a flop?
  floating point operation

The theoretical peak performance is unattainable. Why?

Memory Hierarchy



CPU vs. Memory

- For the CPU to execute an operation, data needs to be in registers

CPU vs. Memory

- For the CPU to execute an operation, data needs to be in registers
- Cache misses
- L1 misses, L2, L3, TLB, instruction misses...
- Cache line

Back to performance

- Theroretical peak performance $\rightarrow$ unattainable
- Practical peak performance? (practical = attainable)
- DGEMM (BLAS)
  Double GEneral Matrix Matrix multiply
  Basic Linear Algebra Subroutines
- Peak performance $\equiv$ DGEMM

Back to performance

- Theroretical peak performance $\rightarrow$ unattainable
- Practical peak performance? (practical = attainable)
- DGEMM (BLAS)
  Double GEneral Matrix Matrix multiply
  Basic Linear Algebra Subroutines
- Peak performance $\equiv$ DGEMM
- How to compute DGEMM's performance?

Performance

$$\text{Perf} = \frac{\#\text{ops}}{\text{exec. time}}$$

Performance

$$\text{Perf} = \frac{\#\text{ops}}{\text{exec. time}}$$

- What is #ops?

Performance

$$\text{Perf} = \frac{\#\text{ops}}{\text{exec. time}}$$

- What is #ops?
- What is #ops in your algorithm?

Performance

$$\text{Perf} = \frac{\#\text{ops}}{\text{exec. time}}$$

- What is #ops?
- What is #ops in your algorithm?
- What if the algortihm is iterative?
  “iterative algorithm” != “loop-based”

Performance

$$\text{Perf} = \frac{\#\text{ops}}{\text{exec. time}}$$

- What is #ops?
- What is #ops in your algorithm?
- What if the algortihm is iterative?
  “iterative algorithm” != “loop-based”

$$\text{Efficiency} = \frac{\text{Perf}}{\text{Peak Perf}}$$

BLAS

Linear Algebra operations decomposed into simpler operations.

BLAS

Linear Algebra operations decomposed into simpler operations.

BLAS-1: $y := y + \alpha x$ $\quad dot := \alpha + x^T y$ $\quad x, y \in \mathbb{R}^n$

BLAS

Linear Algebra operations decomposed into simpler operations.

BLAS-1: $y := y + \alpha x$, $dot := \alpha + x^T y$ $\quad x, y \in \mathbb{R}^n$

BLAS-2: $y := y + Ax$, $y := L^{-1}x$ $\quad A, L \in R^{n \times n},\ x, y \in R^n$

BLAS

Linear Algebra operations decomposed into simpler operations.

BLAS-1:	$y := y + \alpha x$ $dot := \alpha + x^T y$	$x, y \in \mathbb{R}^n$
BLAS-2:	$y := y + Ax$ $y := L^{-1}x$	$A, L \in R^{n \times n},\ x, y \in R^n$
BLAS-3:	$C := C + AB$ $C := L^{-1}B$	$A, B, C, L \in R^{n \times n}$

BLAS

Linear Algebra operations decomposed into simpler operations.

BLAS-1:	$y := y + \alpha x$ $dot := \alpha + x^T y$	$x, y \in \mathbb{R}^n$
BLAS-2:	$y := y + Ax$ $y := L^{-1}x$	$A, L \in R^{n \times n},\ x, y \in R^n$
BLAS-3:	$C := C + AB$ $C := L^{-1}B$	$A, B, C, L \in R^{n \times n}$

BLAS	#FLOPS	Mem. refs.	Ratio
Level 1	$2n$	$3n$	$\mathbf{2/3}$

BLAS

Linear Algebra operations decomposed into simpler operations.

BLAS-1:	$y := y + \alpha x$ $dot := \alpha + x^T y$	$x, y \in \mathbb{R}^n$
BLAS-2:	$y := y + Ax$ $y := L^{-1}x$	$A, L \in R^{n \times n},\ x, y \in R^n$
BLAS-3:	$C := C + AB$ $C := L^{-1}B$	$A, B, C, L \in R^{n \times n}$

BLAS	#FLOPS	Mem. refs.	Ratio
Level 1	$2n$	$3n$	$\mathbf{2/3}$
Level 2	$2n^2$	n^2	$\mathbf{2}$

BLAS

Linear Algebra operations decomposed into simpler operations.

BLAS-1:	$y := y + \alpha x$ $dot := \alpha + x^T y$	$x, y \in \mathbb{R}^n$
BLAS-2:	$y := y + Ax$ $y := L^{-1}x$	$A, L \in R^{n \times n},\ x, y \in R^n$
BLAS-3:	$C := C + AB$ $C := L^{-1}B$	$A, B, C, L \in R^{n \times n}$

BLAS	#FLOPS	Mem. refs.	Ratio
Level 1	$2n$	$3n$	$\mathbf{2/3}$
Level 2	$2n^2$	n^2	$\mathbf{2}$
Level 3	$2n^3$	$4n^2$	$\boldsymbol{n/2}$

Time vs. Performance

- Can you cheat time?

Time vs. Performance

- Can you cheat time?
- Can you cheat performance?

Time vs. Performance

- Can you cheat time?
- Can you cheat performance?
- “fast” flops vs. “slow” flops

Time vs. Performance

- Can you cheat time?
- Can you cheat performance?
- “fast” flops vs. “slow” flops
- $5n \log n$ vs. $2n^2$

Time vs. Performance

- Can you cheat time?
- Can you cheat performance?
- “fast” flops vs. “slow” flops
- $5n \log n$ vs. $2n^2$

USE BOTH!

Parallel Performance

- p processors

Parallel Performance

- p processors
  Total performance? Execution time?

Parallel Performance

- p processors
  Total performance? Execution time?

- $T_p(n)$ = execution time for a problem of size n

Parallel Performance

- p processors
  Total performance? Execution time?
- $T_p(n)$ = execution time for a problem of size n
- Speedup = $\frac{T_1(n)}{T_p(n)}$

Parallel Performance

- p processors
  Total performance? Execution time?
- $T_p(n)$ = execution time for a problem of size n
- Speedup = $\frac{T_1(n)}{T_p(n)}$
  $T_1(n) = ?$

Parallel Performance

- p processors
  Total performance? Execution time?
- $T_p(n)$ = execution time for a problem of size n
- Speedup = $\frac{T_1(n)}{T_p(n)}$
  $T_1(n) = ?$
- Strong Scalability: $\frac{T_p(n)}{T_{2p}(n)}$, with p increasing
  fixed problem, increasing number of processors

Parallel Performance

- p processors
  Total performance? Execution time?
- $T_p(n)$ = execution time for a problem of size n
- Speedup = $\frac{T_1(n)}{T_p(n)}$
  $T_1(n) = ?$
- Strong Scalability: $\frac{T_p(n)}{T_{2p}(n)}$, with p increasing
  fixed problem, increasing number of processors
- Weak Scalability: $\frac{T_p(n)}{T_{4p}(2n)}$, with p increasing
  fixed memory use per processor, increasing number of processors