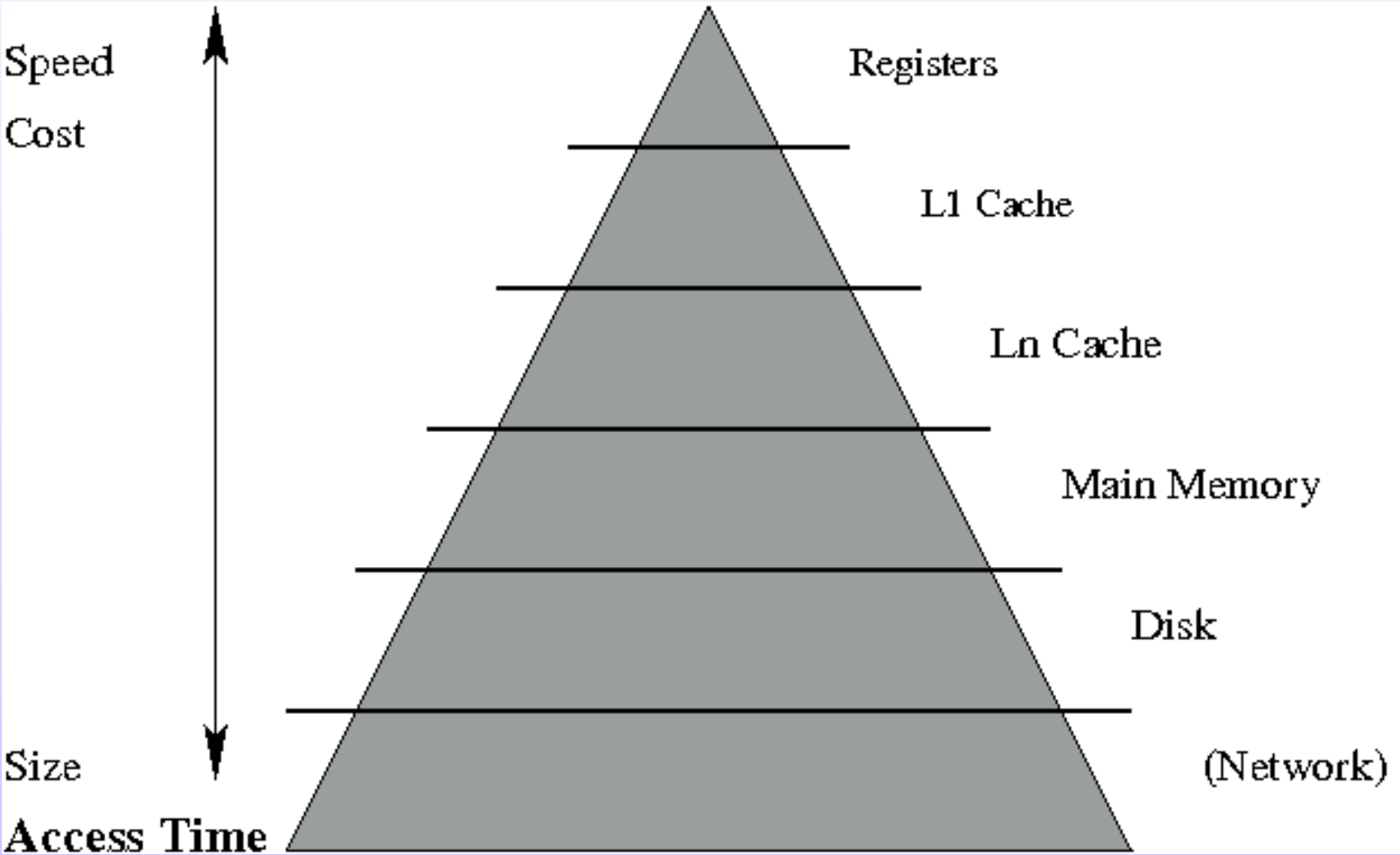
High-performance Matrix Computations

Prof. **Paolo Bientinesi**

pauldj@aices.rwth-aachen.de

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
- Cache line
- L1 misses, L2, L3, TLB, instruction misses. . .
- Prefetching (HW, SW)

Locality

"Locality of references", "Principle of locality"

- Temporal locality
- Spatial locality

Performance

Quality of an algorithm

Metrics?

Performance

Quality of an algorithm

Metrics?

- Execution time
- Complexity
- Stability
- Accuracy
- Memory usage
- Memory/network accesses
- ...

Execution time

Pros? Cons?

Execution time

Pros? Cons?

- 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**

- Quality of the algorithm IN RELATION to the potential of the architecture
- Potential of an architecture?

- Quality of the algorithm IN RELATION to the potential of the architecture
- Potential of an architecture?

Theoretical Peak Performance

TPP = #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

- Quality of the algorithm IN RELATION to the potential of the architecture
- Potential of an architecture?

Theoretical Peak Performance

TPP = #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

TPP is unattainable. Why?

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”

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

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!

To GEMM or not to GEMM

"GEMM makes it really hard to win with better algorithms that don't use it."

To GEMM or not to GEMM

"GEMM makes it really hard to win with better algorithms that don't use it."

"You can use GEMM stupidly and still win because on most processors GEMM is the speed-of-light."

To GEMM or not to GEMM

"GEMM makes it really hard to win with better algorithms that don't use it."

"You can use GEMM stupidly and still win because
on most processors GEMM is the speed-of-light."

"GEMM is holding back algorithmic innovation for tensor computations."
J.H.

BLAS

Linear Algebra operations decomposed into simpler operations.

BLAS

Linear Algebra operations decomposed into simpler operations.

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

$\gamma := \alpha + x^T y$

BLAS

Linear Algebra operations decomposed into simpler operations.

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

$\gamma := \alpha + x^T y$

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

$y := L^{-1} x$

BLAS

Linear Algebra operations decomposed into simpler operations.

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

BLAS

Linear Algebra operations decomposed into simpler operations.

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

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

BLAS-3: $C := AB + C$ $\quad C := L^{-1}B$ $\quad A, B, C, L \in R^{n \times n}$

BLAS	#FLOPS	Mem. refs.	Ratio

BLAS

Linear Algebra operations decomposed into simpler operations.

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

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

BLAS-3: $C := AB + C$ $\quad C := L^{-1}B$ $\quad A, B, C, L \in R^{n \times n}$

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

BLAS

Linear Algebra operations decomposed into simpler operations.

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

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

BLAS-3: $C := AB + C$ $\quad C := L^{-1}B$ $\quad A, B, C, L \in R^{n \times n}$

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: $y := \alpha x + y$ $\quad \gamma := \alpha + x^T y$ $\quad x, y \in \mathbb{R}^n$

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

BLAS-3: $C := AB + C$ $\quad C := L^{-1}B$ $\quad A, B, C, L \in R^{n \times n}$

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

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 = $\dfrac{T_1(n)}{T_p(n)}$ $\qquad T_1(n) = ?$

Parallel Performance

- p processors
  Total performance? Execution time?
- $T_p(n)$ = execution time for a problem of size n
- Speedup = $\frac{T_1(n)}{T_p(n)}$ $\quad T_1(n) = ?$ $\quad \frac{T_{p_0}(n)}{T_p(n)}$

Parallel Performance

- p processors
  Total performance? Execution time?
- $T_p(n)$ = execution time for a problem of size n
- Speedup = $\frac{T_1(n)}{T_p(n)}$ $\quad T_1(n) = ?$ $\quad \frac{T_{p_0}(n)}{T_p(n)}$
- Strong Scalability =
  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 = $\frac{T_1(n)}{T_p(n)}$ $\quad T_1(n) = ?$ $\quad \frac{T_{p_0}(n)}{T_p(n)}$
- Strong Scalability =
  fixed problem, increasing number of processors

  $T_p(n), \quad \frac{T_p(n)}{T_{2p}(n)}, \quad$ increasing p

Parallel Performance

- p processors
  Total performance? Execution time?
- $T_p(n)$ = execution time for a problem of size n
- Speedup = $\frac{T_1(n)}{T_p(n)}$ $\quad T_1(n) = ?$ $\quad \frac{T_{p_0}(n)}{T_p(n)}$
- Strong Scalability =
  fixed problem, increasing number of processors

  $T_p(n), \quad \frac{T_p(n)}{T_{2p}(n)}, \quad$ increasing p
- Weak Scalability =
  fixed memory use per processor, increasing number of processors

Parallel Performance

- p processors
  Total performance? Execution time?
- $T_p(n)$ = execution time for a problem of size n
- Speedup = $\frac{T_1(n)}{T_p(n)}$ $\quad T_1(n) = ?$ $\quad \frac{T_{p_0}(n)}{T_p(n)}$
- Strong Scalability =
  fixed problem, increasing number of processors

  $T_p(n), \quad \frac{T_p(n)}{T_{2p}(n)}, \quad$ increasing p
- Weak Scalability =
  fixed memory use per processor, increasing number of processors

  $T_p(n), \quad \frac{T_p(n)}{T_{4p}(2n)}, \quad$ increasing p