



# Chapter 10
# Parallel Algorithms

## Algorithm Theory
## WS 2015/16

Fabian Kuhn

# Brent's Theorem

**Brent's Theorem:** On $p$ processors, a parallel computation can be performed in time

$$T_p \leq \frac{T_1 - T_\infty}{p} + T_\infty.$$

**Corollary:** Greedy is a 2-approximation algorithm for scheduling.

**Corollary:** As long as the number of processors $p = O(T_1/T_\infty)$, it is possible to achieve a linear speed-up.

# Prefix Sums

- The following works for any associative binary operator $\oplus$:

  **associativity:** $(a \oplus b) \oplus c = a \oplus (b \oplus c)$

**All-Prefix-Sums:** Given a sequence of $n$ values $a_1, \dots, a_n$, the all-prefix-sums operation w.r.t. $\oplus$ returns the sequence of prefix sums:

$$s_1, s_2, \dots, s_n = a_1, a_1 \oplus a_2, a_1 \oplus a_2 \oplus a_3, \dots, a_1 \oplus \cdots \oplus a_n$$

- Can be computed efficiently in parallel and turns out to be an important building block for designing parallel algorithms

**Example:** Operator: $+$, input: $a_1, \dots, a_8 = 3, 1, 7, 0, 4, 1, 6, 3$

$s_1, \dots, s_8 =$

# Computing Prefix Sums

**Theorem:** Given a sequence $a_1, \dots, a_n$ of $n$ values, all prefix sums $s_i = a_1 \oplus \cdots \oplus a_i$ (for $1 \leq i \leq n$) can be computed in time $O(\log n)$ using $O(n/\log n)$ processors on an EREW PRAM.

**Proof:**

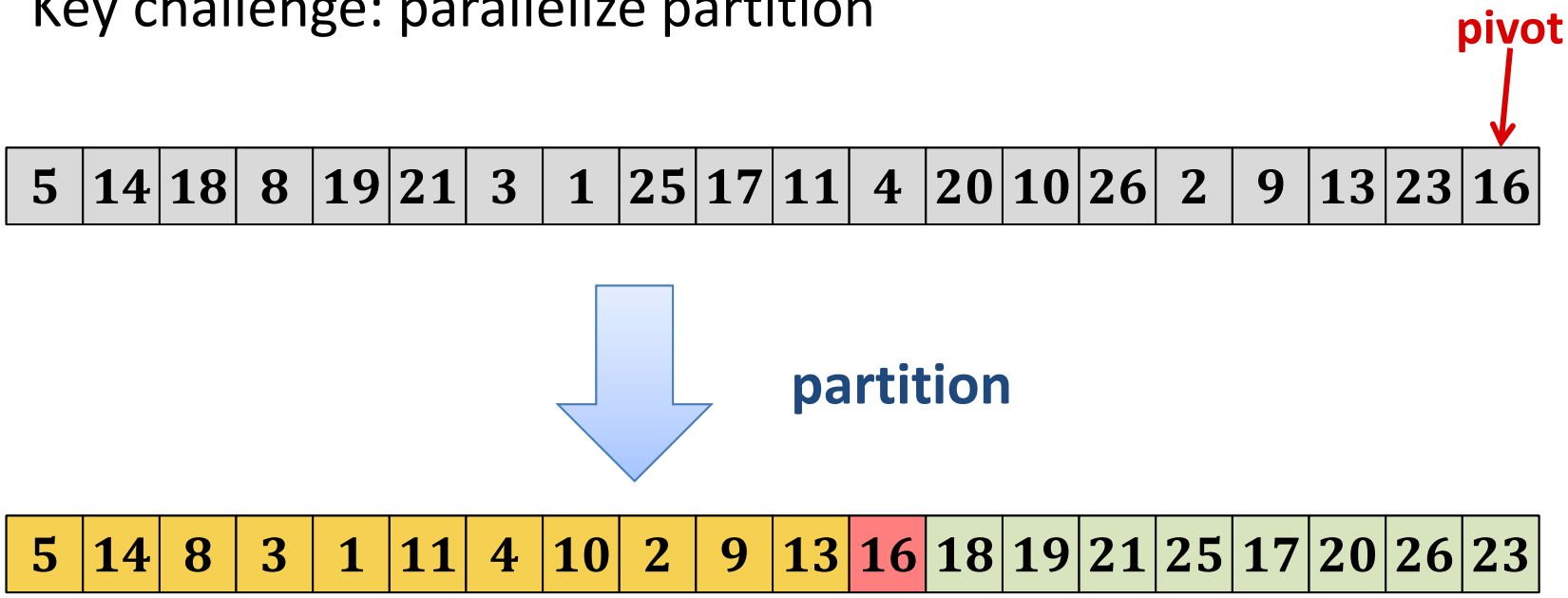
- Computing the sums of all sub-trees can be done in parallel in time $O(\log n)$ using $O(n)$ total operations.
- The same is true for the top-down step to compute the $r(v)$
- The theorem then follows from Brent's theorem:

$$T_1 = O(n), \qquad T_\infty = O(\log n) \qquad \Longrightarrow \qquad T_p < T_\infty + \frac{T_1}{p}$$

**Remark:** This can be adapted to other parallel models and to different ways of storing the value (e.g., array or list)

# Parallel Quicksort

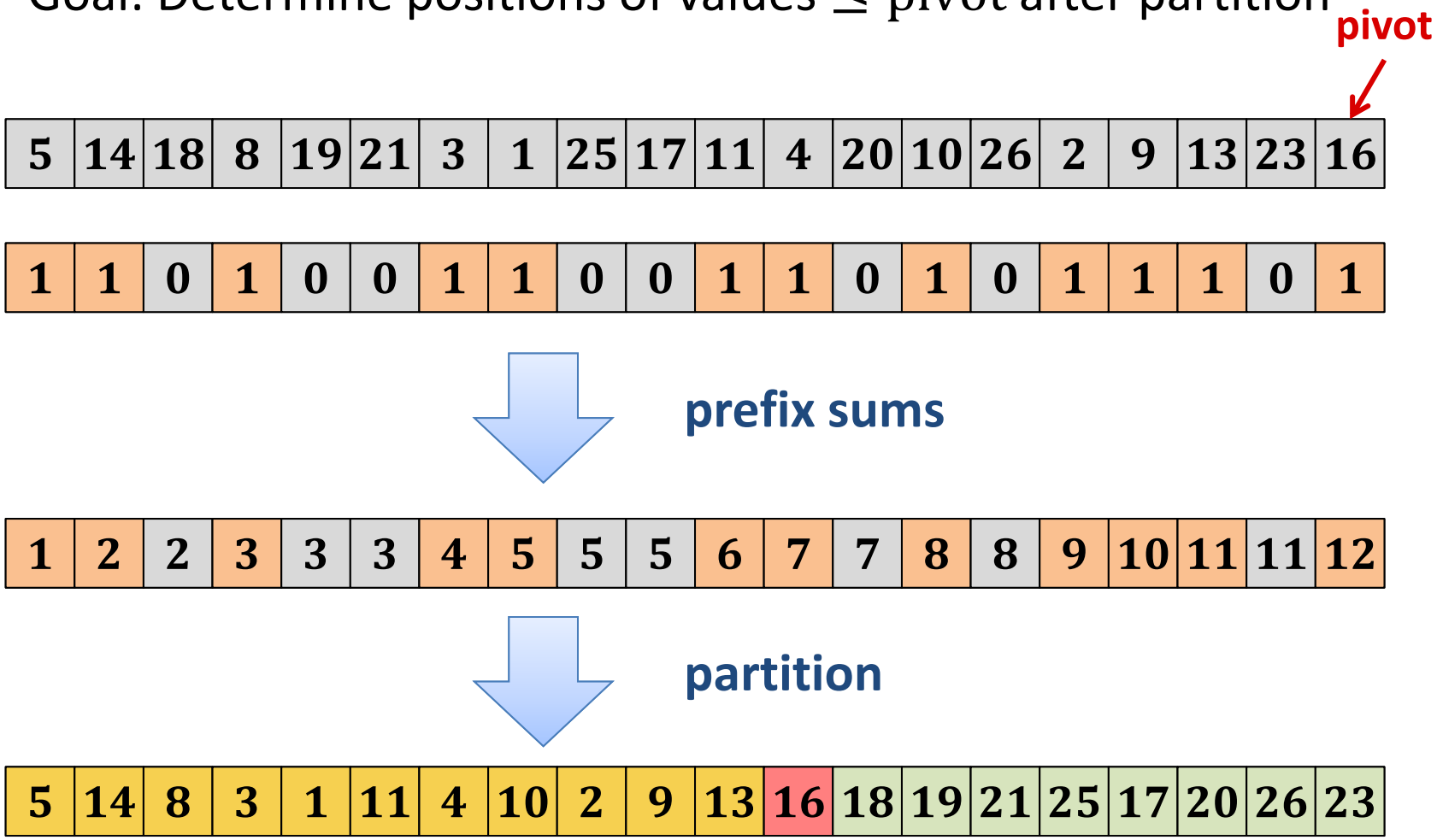
- Key challenge: parallelize partition




- How can we do this in parallel?
- For now, let's just care about the values $\leq$ pivot
- What are their new positions

# Using Prefix Sums

- Goal: Determine positions of values $\leq$ pivot after partition



# Partition Using Prefix Sums

- The positions of the entries $>$ pivot can be determined in the same way
- **Prefix sums:** $T_1 = O(n), \quad T_\infty = O(\log n)$
- **Remaining computations:** $T_1 = O(n), \quad T_\infty = O(1)$
- **Overall:** $T_1 = O(n), \quad T_\infty = O(\log n)$

**Lemma:** The partitioning of quicksort can be carried out in parallel in time $O(\log n)$ using $O\left(\frac{n}{\log n}\right)$ processors.

**Proof:**

- By Brent's theorem: $T_p \leq \frac{T_1}{p} + T_\infty$

# Applying to Quicksort

**Theorem:** On an EREW PRAM, using $p$ processors, randomized quicksort can be executed in time $T_p$ (in expectation and with high probability), where

$$T_p = O\left(\frac{n \log n}{p} + \log^2 n\right).$$

**Proof:**

**Remark:**

- We get optimal (linear) speed-up w.r.t. to the sequential algorithm for all $p = O(n/\log n)$.

# Other Applications of Prefix Sums

- Prefix sums are a very powerful primitive to design parallel algorithms.
  - Particularly also by using other operators than +

**Example Applications:**

- Lexical comparison of strings
- Add multi-precision numbers
- Evaluate polynomials
- Solve recurrences
- Radix sort / quick sort
- Search for regular expressions
- Implement some tree operations
- …