Karlsruher Institut für Technologie

Übung 9 – Algorithmen II

Michael Axtmann – *michael.axtmann@kit.edu*

http://algo2.iti.kit.edu/AlgorithmenII_WS16.php

Institut für Theoretische Informatik - Algorithmik II

Themenübersicht

Karlsruher Institut für Technologie

Organisatorisches

- Evaluation am Ende der Übung

Parallelverarbeitung

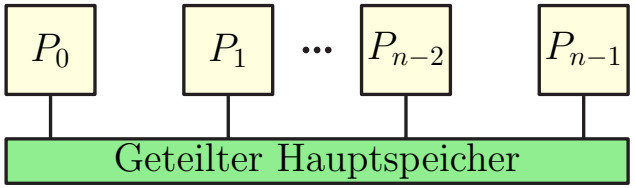
- Modelle
- Verbindungsstrukturen
- Anwendungen
  - Präfixsumme
  - Paralleles Sortieren
  - Effizienz

Parallelverarbeitung

Modelle

PRAM

- synchrone Prozessoren
- gemeinsamer Speicher
- Speicherkonflikte



Verteilter gemeinsamer Speicher

(symmetrisch) gemeinsamer Speicher

BulkSynchronousParallel

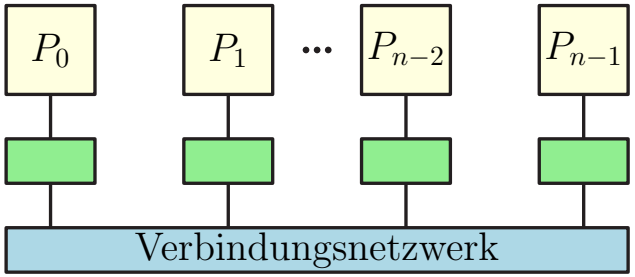
- kollektiver Nachrichtenaustausch aller Rechner
- BSP* berücksichtigt Nachrichtenlänge

Parallelverarbeitung

Modelle

PRAM

- synchrone Prozessoren
- gemeinsamer Speicher
- Speicherkonflikte



Verteilter gemeinsamer Speicher

(symmetrisch) gemeinsamer Speicher

BulkSynchronousParallel

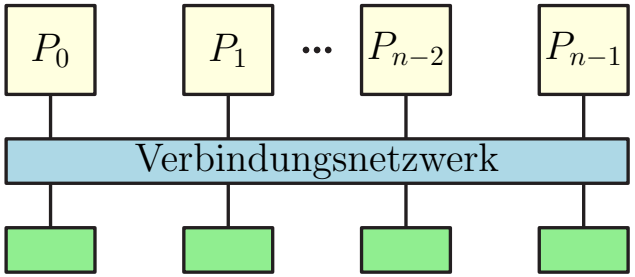
- kollektiver Nachrichtenaustausch aller Rechner
- BSP* berücksichtigt Nachrichtenlänge

Parallelverarbeitung

Modelle

PRAM

- synchrone Prozessoren
- gemeinsamer Speicher
- Speicherkonflikte



Verteilter gemeinsamer Speicher

(symmetrisch) gemeinsamer Speicher

BulkSynchronousParallel

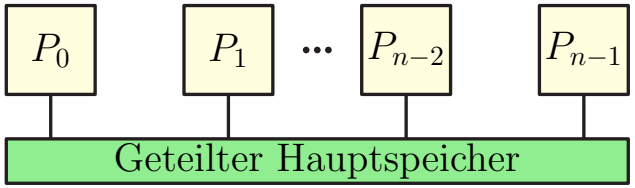
- kollektiver Nachrichtenaustausch aller Rechner
- BSP* berücksichtigt Nachrichtenlänge

PRAM

Speicherkonflikte

PRAM

- klassifiziert nach Lese- (*read*) und Schreibzugriff (*write*)
- betrachte gleichzeitigen (*concurrent*) oder exklusiven Zugriff (*exclusive*)
  - EREW
  - ERCW schwachsinn
  - CREW
  - CRCW
    - **common**: alle müssen den gleichen Wert schreiben
    - **arbitrary**: Wert eines zufälligen Prozessors
    - **priority**: Wert mit kleinster Prozessor-ID
    - **combine**: Aggregation der Werte (z.B. Summe)



Verbindungsnetzwerke

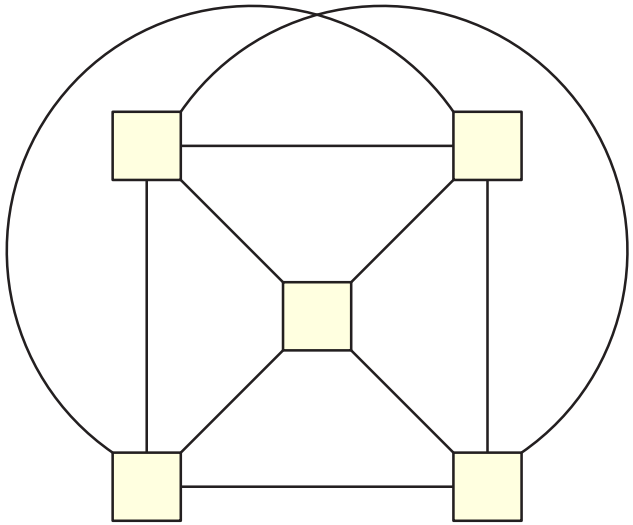
Struktur

Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p \cdot (p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex
  - Telefon
  - Duplex

Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $* * * 1 * * \leftrightarrow * * * 0 * *$



Kosten

- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

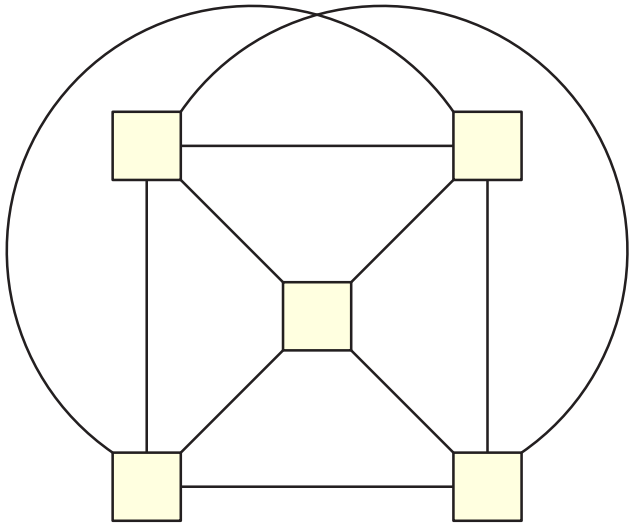
Struktur

Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p \cdot (p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \rightarrow j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \rightarrow j, k \rightarrow i$

Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$

Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$

$\bullet$
0

Kosten

- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p \cdot (p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$

Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$

Kosten

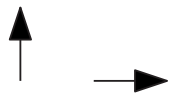
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

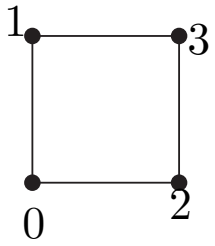
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

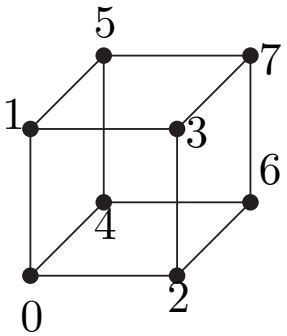
Struktur

Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \rightarrow j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \rightarrow j, k \rightarrow i$

Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

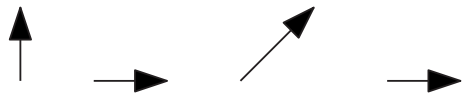
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

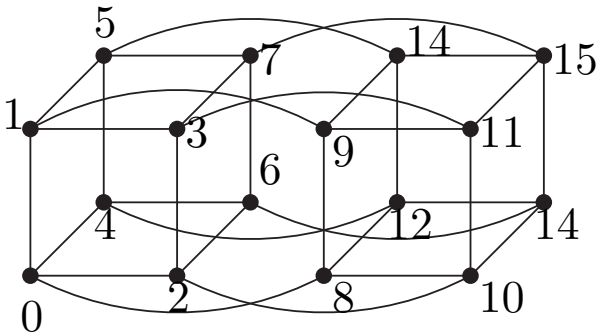
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

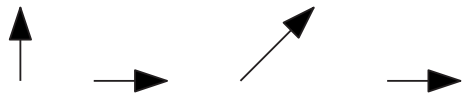
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

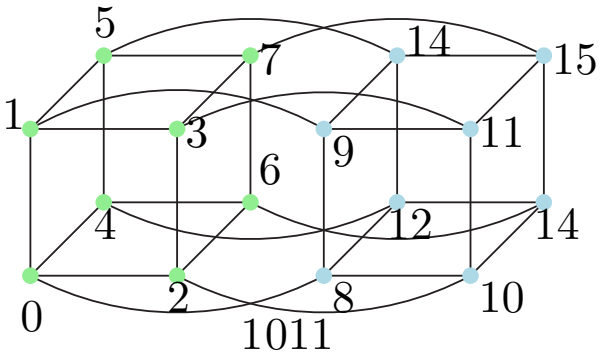
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \rightarrow j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \rightarrow j, k \rightarrow i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

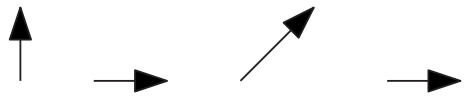
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

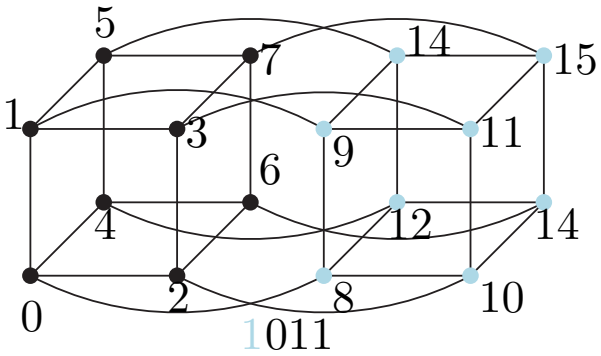
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

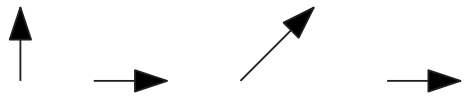
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

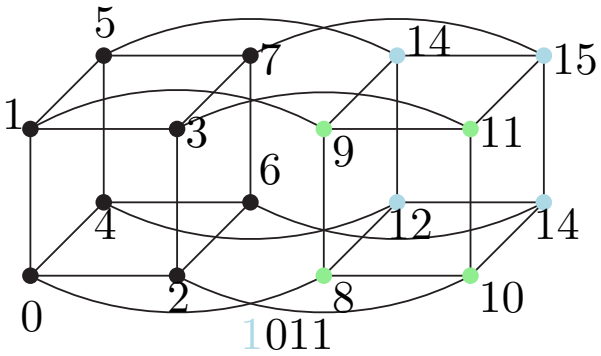
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

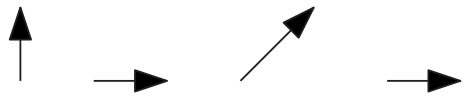
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

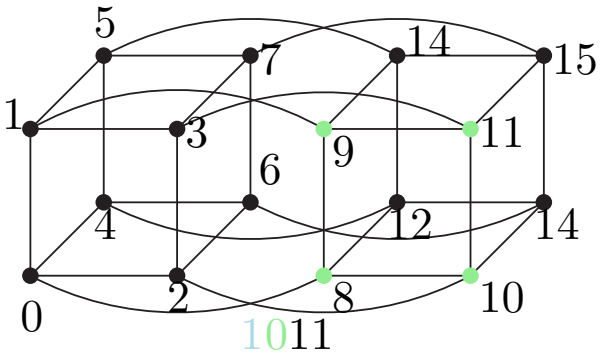
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

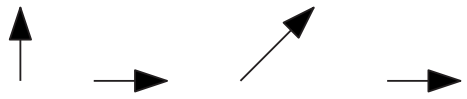
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

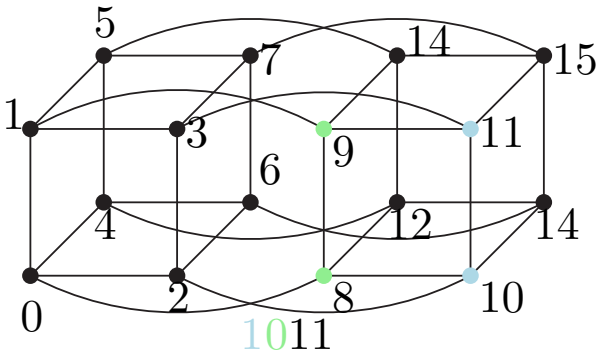
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

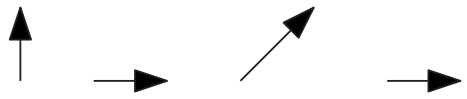
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

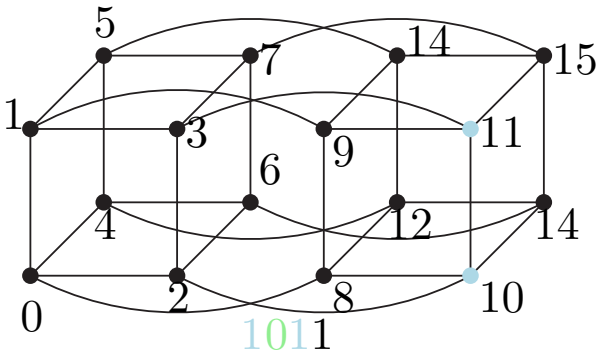
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

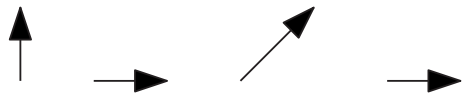
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

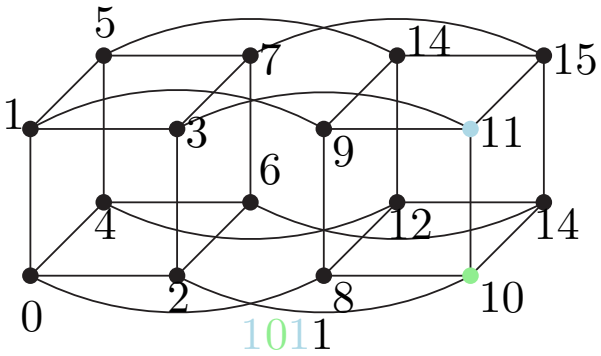
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

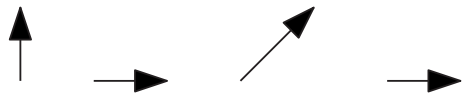
- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Verbindungsnetzwerke

Struktur

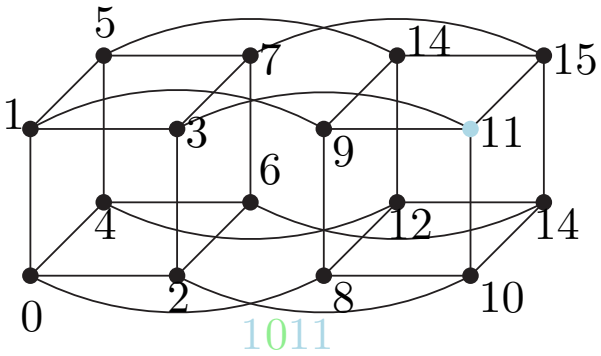
Vollverkabelt

- nur für geringe Anzahl an Rechnern
- $\frac{p\cdot(p-1)}{2}$ Verbindungen nötig
- Varianten
  - Simplex $i \to j$
  - Telefon $i \leftrightarrow j$
  - Duplex $i \to j, k \to i$



Hyperwürfel

- $p \log p$ Verbindungen
- klare Nummerierung von Nachbarn
  $***1** \leftrightarrow ***0**$



Kosten

- Kostenmaß Kommunikation $T_{comm} = T_{start} + l \cdot T_{byte}$

Anwendungen

PRAM Präfixsumme

- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

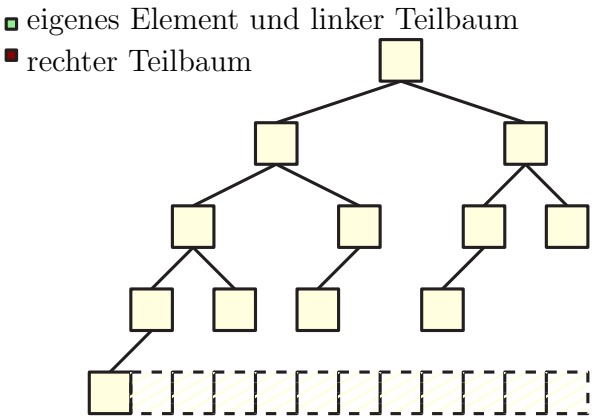
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

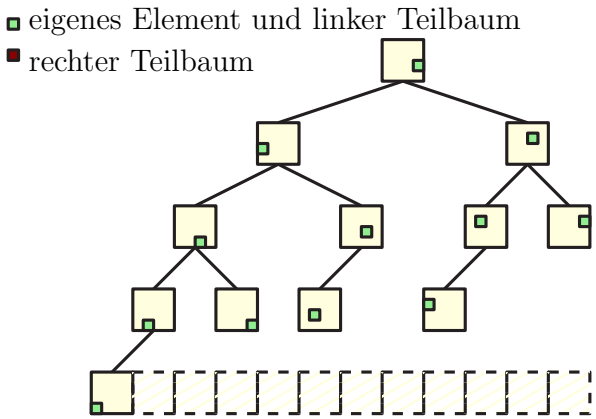
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

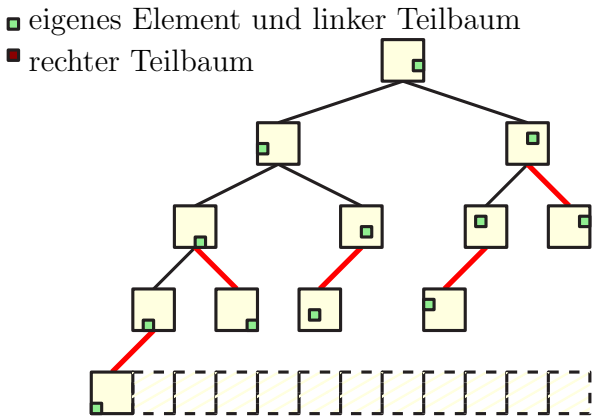
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

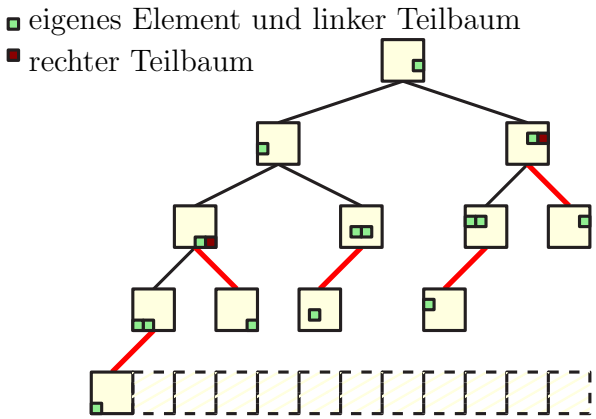
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

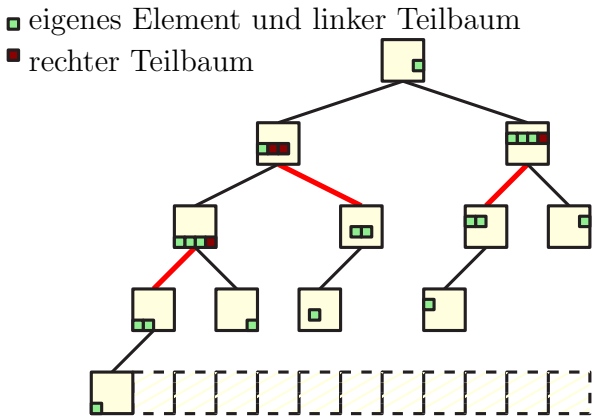
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

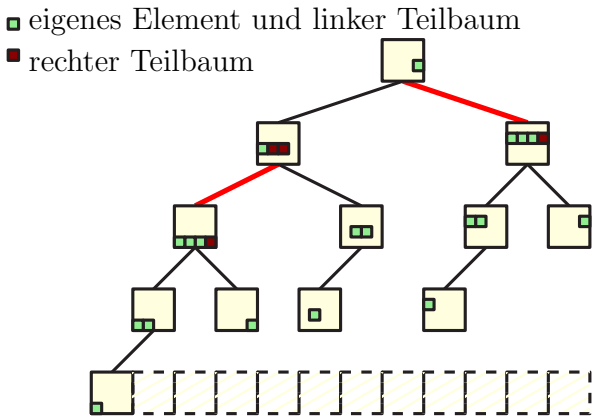
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

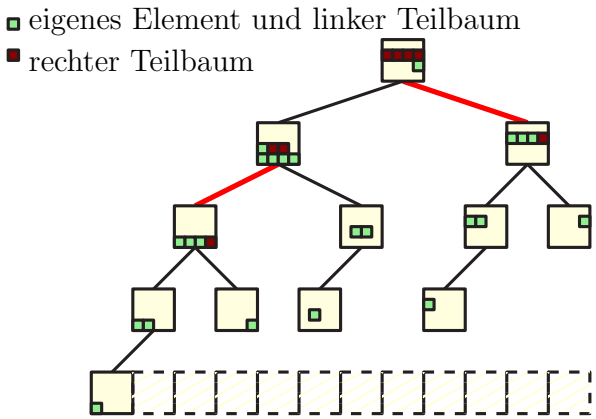
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

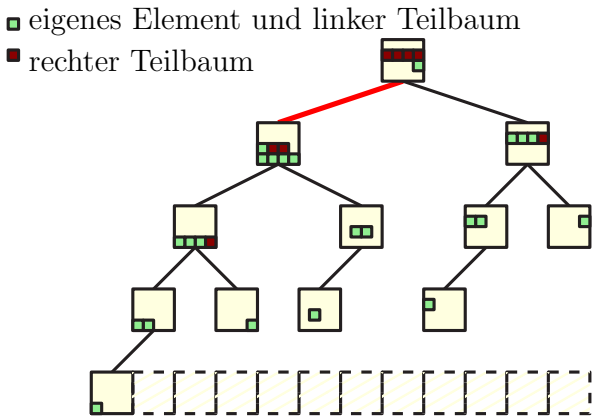
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

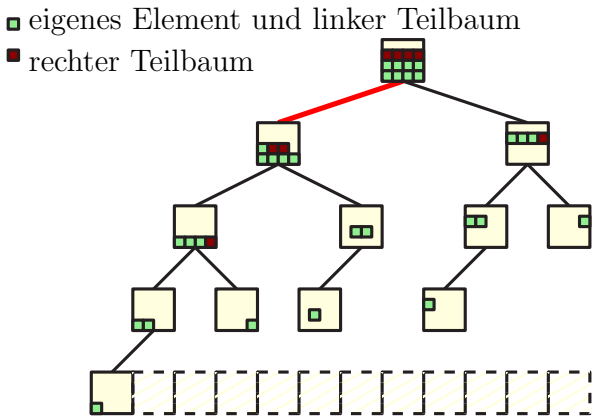
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

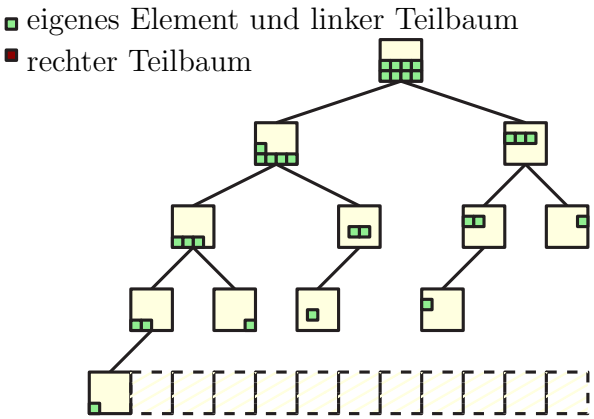
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

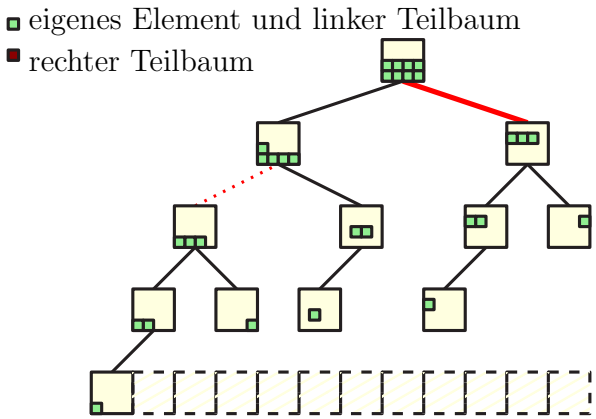
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

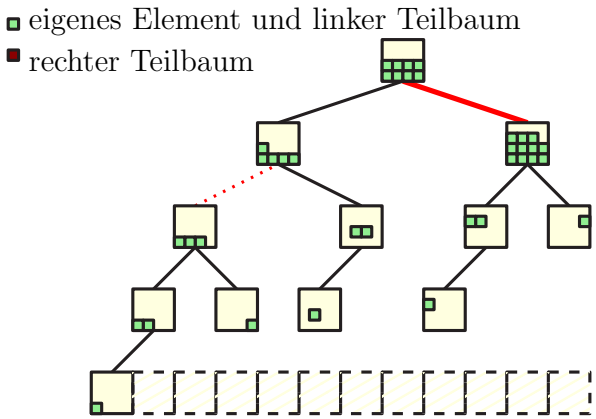
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

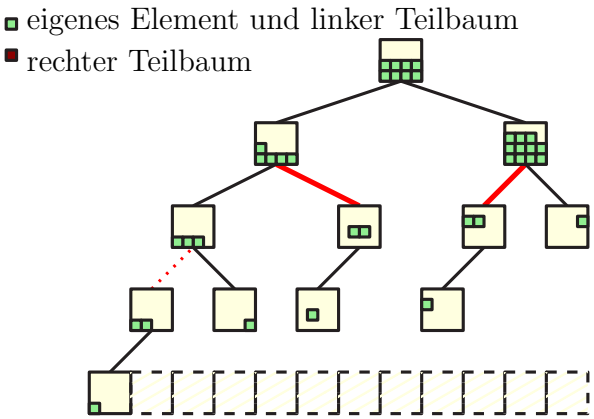
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

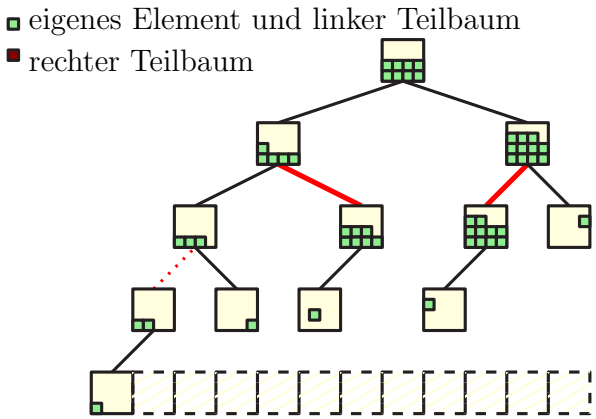
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

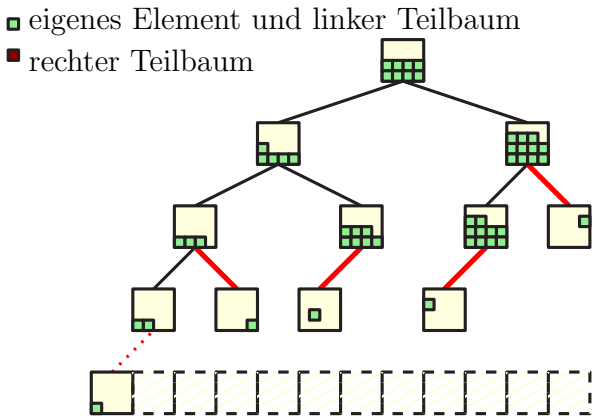
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

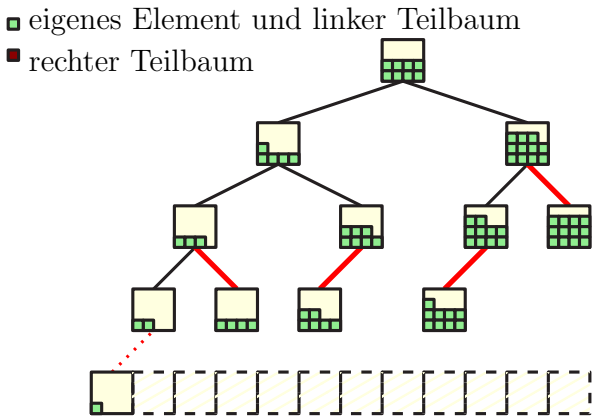
- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

PRAM Präfixsumme



- viele komplexere Algorithmen nutzen Reduktionsschemata in Baumform
- hier Beispiel Präfixsumme (Fibonacci-Baum)
- zweiphasig, aufwärts und abwärts

Aufwärtsphase

- Prozessoren aggregieren Daten aus Teilbäumen
- speichere Summe kleinerer (linker Teilbaum) und größerer (rechter Teilaum) Elemente (getrennt voneinander)
- leite Summe aller Elemente an Vorgängerknoten

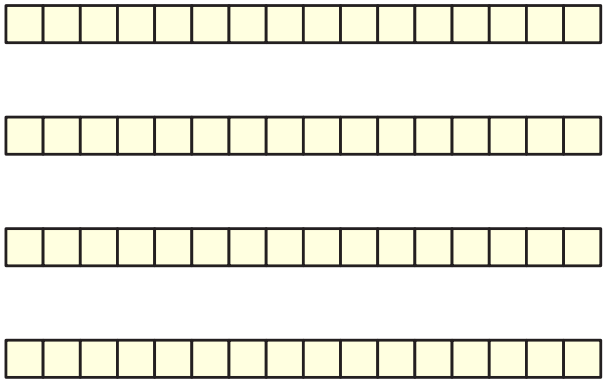
Abwärtsphase

- gesammelte Daten werden verteilt
- bisher in Abwärtsphase empfangene Daten; eigene Daten und Daten des linken Teilbaums nur nach rechts

Anwendungen

Paralleler Quicksort

Rekursives Verfahren



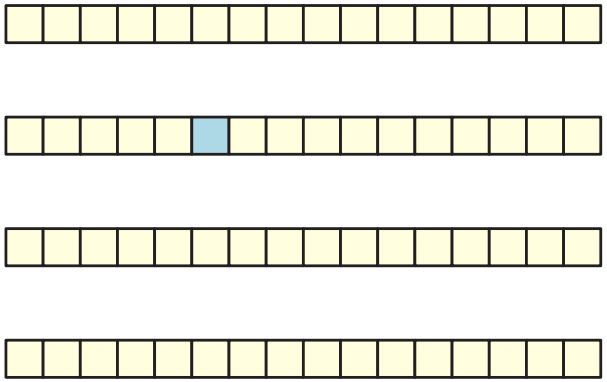
1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

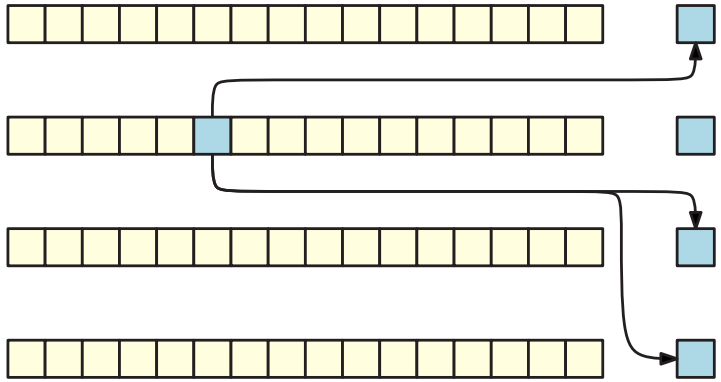


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

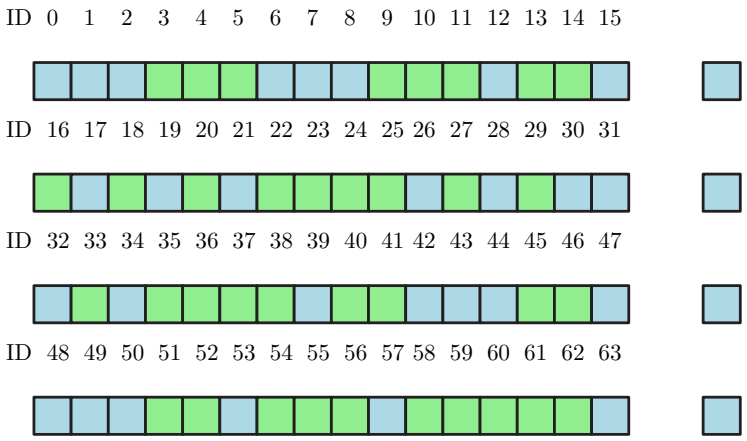


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

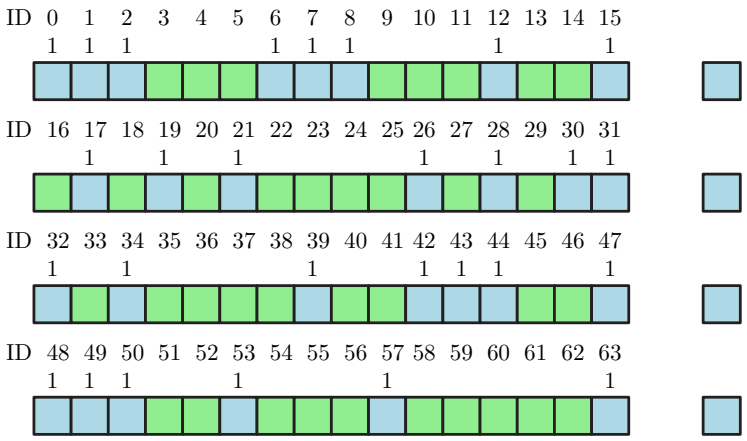


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

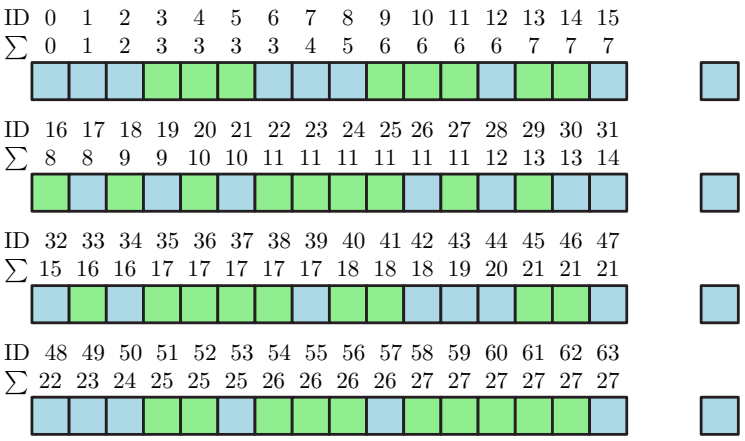


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

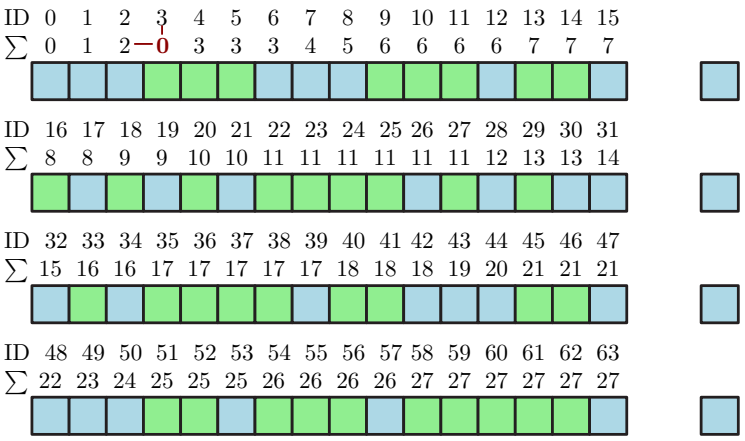


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

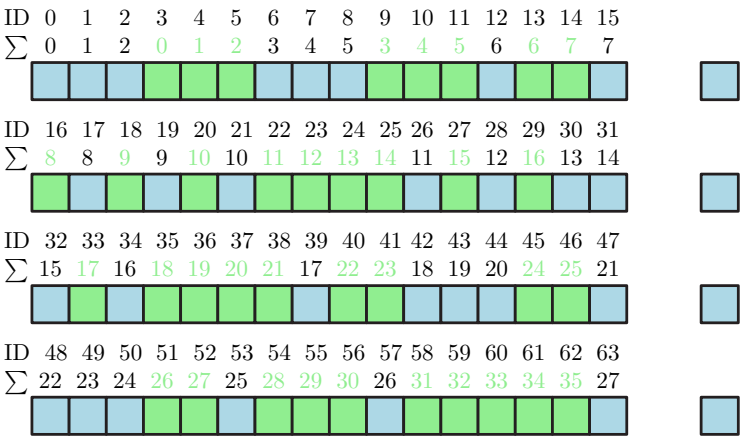


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

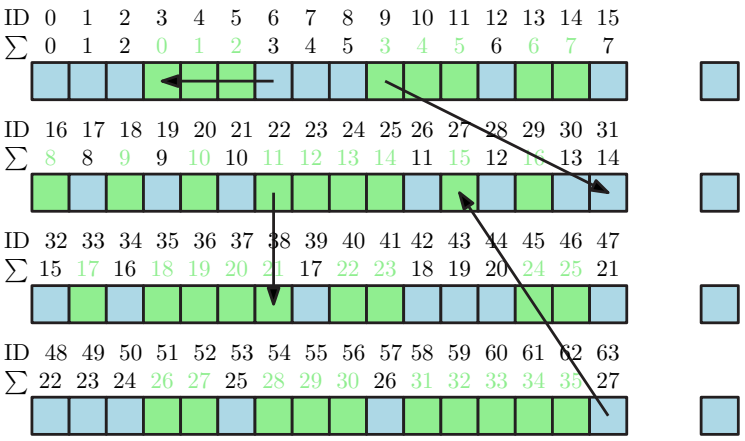


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

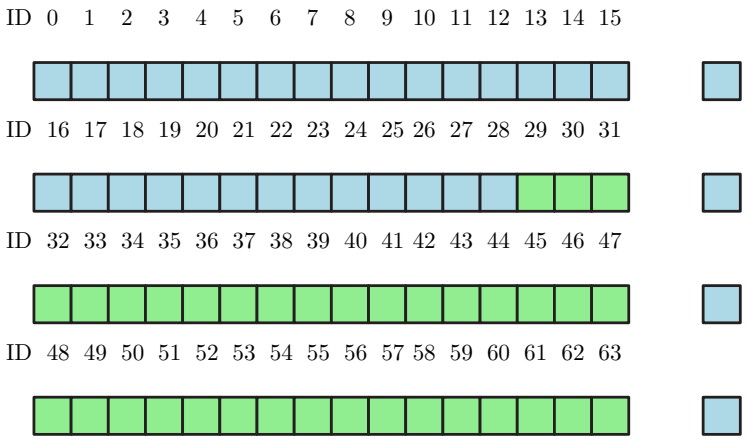


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

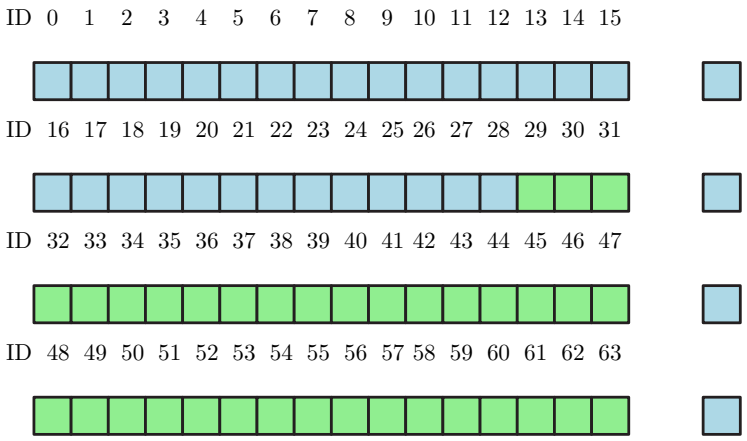


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

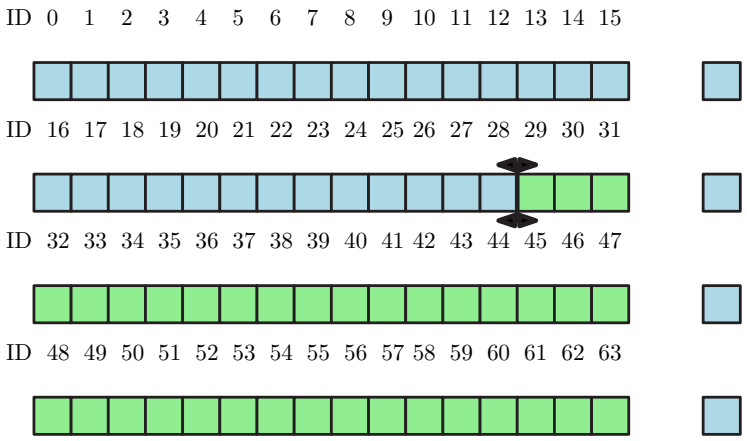


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion

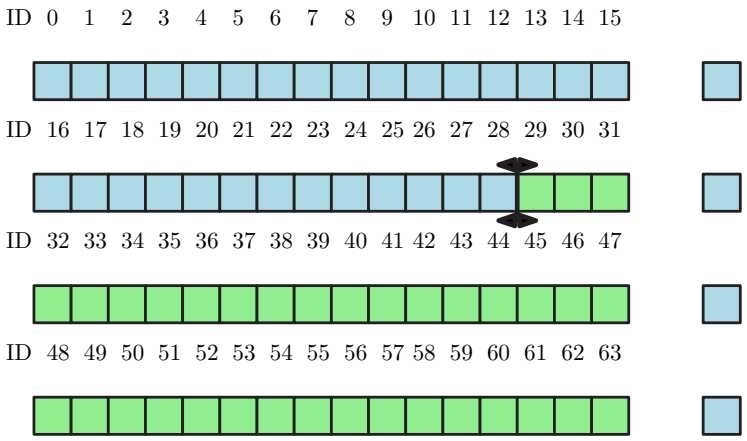


Anwendungen

Paralleler Quicksort

Rekursives Verfahren

1. ein PE stellt Pivot zufällig
2. Broadcast
3. lokaler Vergleich
4. kleine Elemente durchnummerieren
   $\rightarrow$ Präfixsumme
5. umverteilen
   - Präfixsumme für große Elemente folgt direkt aus ID und Präfixsumme kleiner Elemente
   - Position kleine Elemente ist Präfixsummenwert
   - Position großer Elemente ist Anzahl kleiner Elemente plus Wert der Präfixsumme für große Elemente
6. Prozessoren aufspalten Problematisch bei unbalancierter Verteilung
7. parallele Rekursion



Parallele Programmierung

Ein Einstieg

Karlsruher Institut für Technologie

Einstieg in parallele Programmierung?

- openMP
  - `www.openmp.org`
  - enthalten im GCC Compiler
  - Parallelität über Preprozessorflags #pragma omp parallel
- Boost Threads
  - `www.boost.org`
  - Bibliothek, eigenes Management
- Grafikkarten für die, die es anspruchsvoller mögen
  - `http://developer.nvidia.com/getting-started-parallel-computing`
  - cudafähige Grafikkarte nötig
  - Nebeneffekte durch ungewohnte Grafikkartenhardware

Ende!

Feierabend!