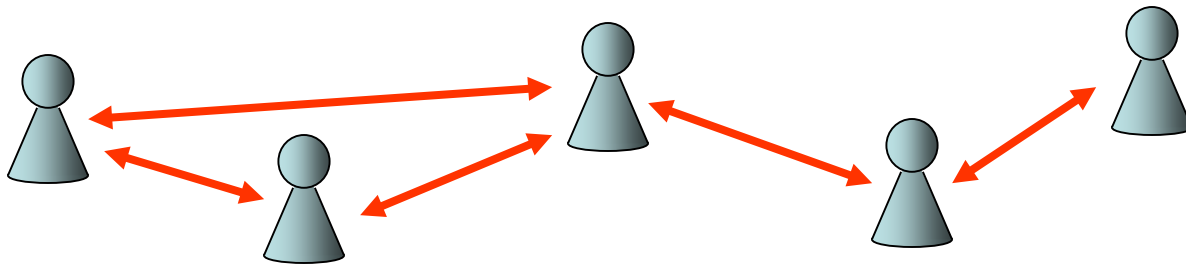


Verteilte Algorithmen und Datenstrukturen



Prof. Dr. Christian Scheideler
Institut für Informatik
Universität Paderborn

Verteilte Algorithmen und Datenstrukturen

Vorlesung: Mi 8:30 - 11:00 Uhr, über BBB

Übung: Mo 11:15 - 12:45 Uhr, über BBB

Vorlesungsaufzeichnungen und Lösungen: in PANDA

Webseite:

<https://cs.uni-paderborn.de/ti/lehre/veranstaltungen/4/4-1>

Prüfung:

- mündliche Prüfung (zwei Blöcke) 50%
- Programmierprojekt 50%
- Beide Teile müssen bestanden sein, um Kurs zu bestehen

Voraussetzungen: Grundkenntnisse in Algorithmen und
Datenstrukturen und Programmierung

Verteilte Algorithmen und Datenstrukturen

Übungsaufgaben:

- wöchentliche Ausgabe jeweils am Mittwoch und Abgabe bis zum darauffolgenden Mittwoch bis Mitternacht per Email an den Übungsgruppenleiter
- erstes Übungsblatt am Mittwoch, den 14. April
- teils theoretisch, teils praktisch

Folien und Übungsaufgaben: PANDA

Bücherempfehlungen: kein Buch (Vorlesung basiert auf neuesten Ergebnissen)

Verteilte Algorithmen und Datenstrukturen

Einordnung in Informatikstudium:

Bachelor I

Datenstrukturen
und Algorithmen

Bachelor II

Verteilte Algorithmen und
Datenstrukturen

Master

Fortgeschrittene verteilte
Algorithmen und Datenstrukturen

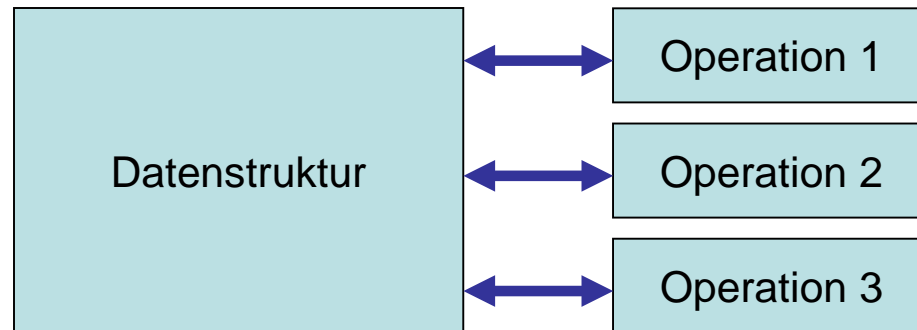
Verteilte Algorithmen und Datenstrukturen

Ziele:

1. Grundlegende verteilte Algorithmen und Datenstrukturen kennen zu lernen.
2. Wichtige Techniken und Entwurfsmethoden kennen und *anwenden* zu lernen.
3. Wichtige Analysemethoden kennen und *anwenden* zu lernen.

Einleitung

Abstrakte Sicht auf Datenstruktur:

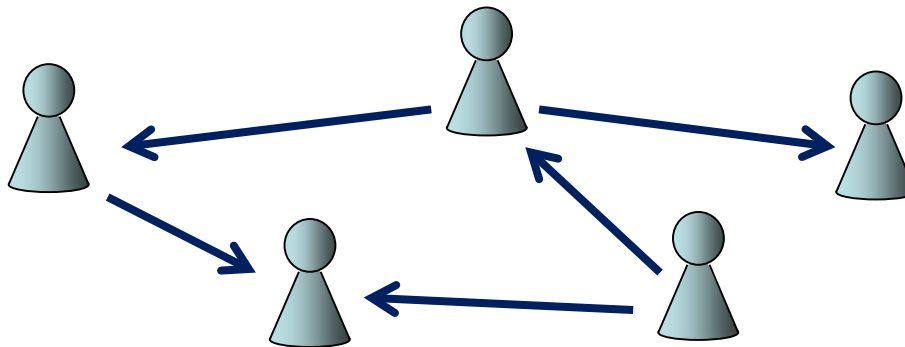


Zentrale Frage: mit welchen Methoden kann die Datenstruktur am **effizientesten** in verteilten Systemen umgesetzt werden?

Einleitung

Verteilte Systeme in dieser Vorlesung:

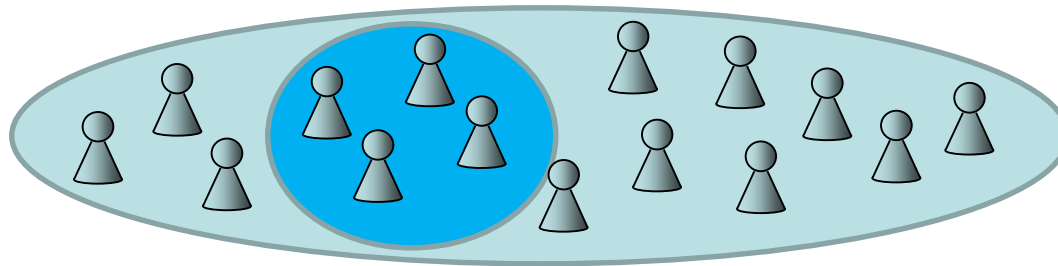
- Prozesse, die miteinander über Kommunikationsverbindungen interagieren.
- Es gibt keinen gemeinsamen Speicher sondern nur die lokalen Speicher der Prozesse.



Einleitung

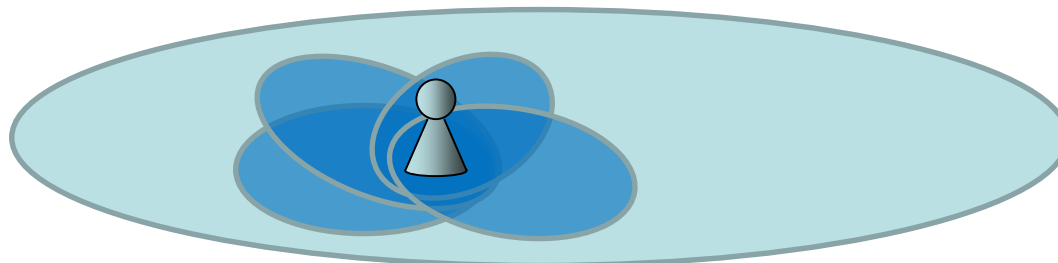
Ziele:

- Jede Operation involviert nur kleine Menge der Prozesse



geringer
Ressourcenbedarf

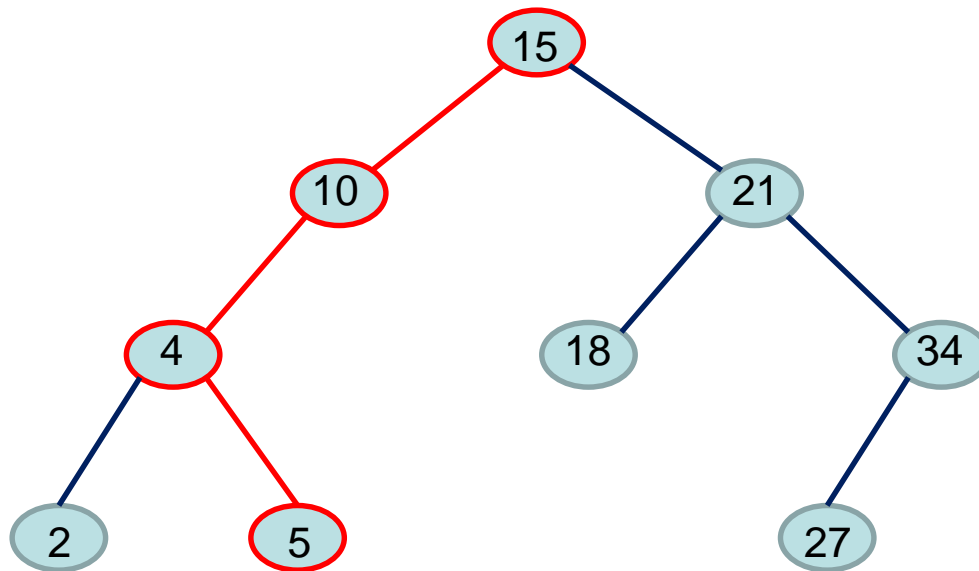
- Für gleichzeitig ausgeführte Operationen ist die maximale Anzahl an Operationen, in die ein Prozess involviert ist, möglichst gering.



hoher
Durchsatz

Einleitung

Sequentieller Fall: Suchbaum, Search(5)

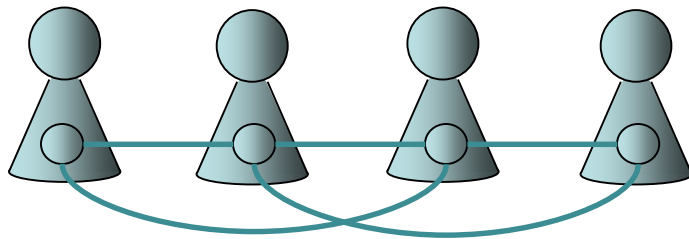


- Search involviert nur wenige Daten, falls Suchbaum balanciert
- Wegen Start bei Wurzel erzeugen **gleichzeitige** Search Operationen eine **hohe Belastung der Wurzel**

Einleitung

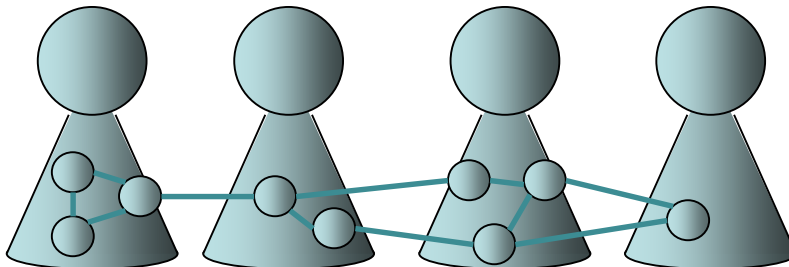
Szenarien für verteilte Datenstrukturen:

- verankerte Daten → **prozessorientierte Datenstruktur**
(nur Vernetzung ändert sich)



Beispiel:
Suchstruktur für
Prozesse (→DNS)

- bewegliche Daten → **informationsorientierte Datenstruktur**
(Datenpositionen und Vernetzungen ändern sich)



Beispiel:
Suchstruktur für
Daten (→Google)

Einleitung

Ziel der Vorlesung: Prozess/Informationsorientierte verteilte Datenstrukturen für elementare Datentypen

- (zyklische) Listen
- Queues und Stacks
- Hashing
- Suchstrukturen
- Priority Queues

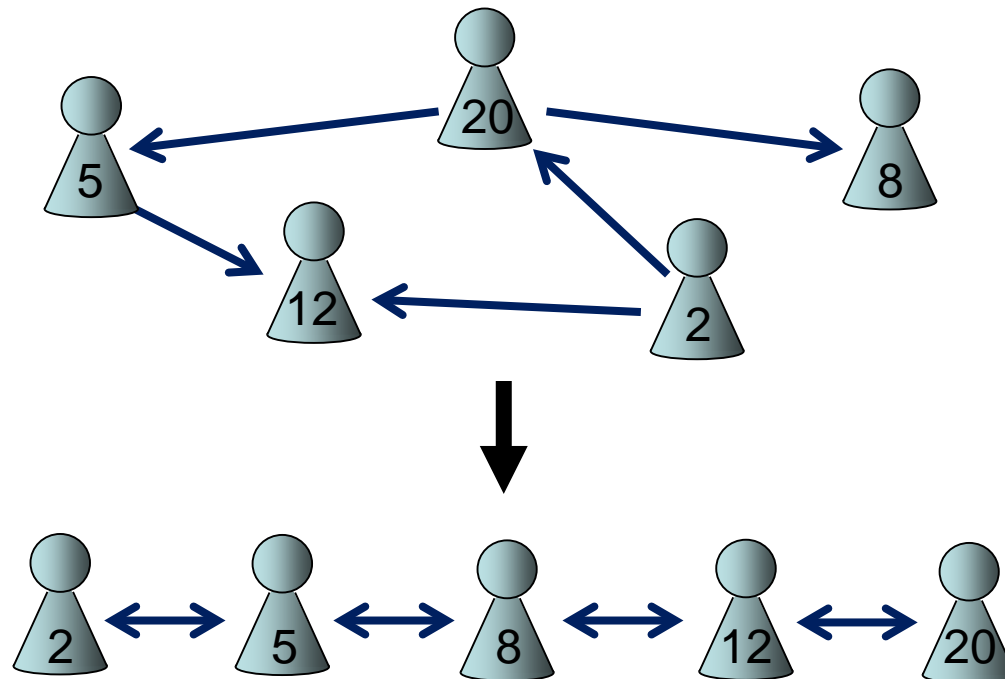
Beispiele in DuA:

- Linear probing
- AVL-Bäume
- binäre Heaps

Einleitung

Liste: Beispiel für prozessorientierte Datenstruktur

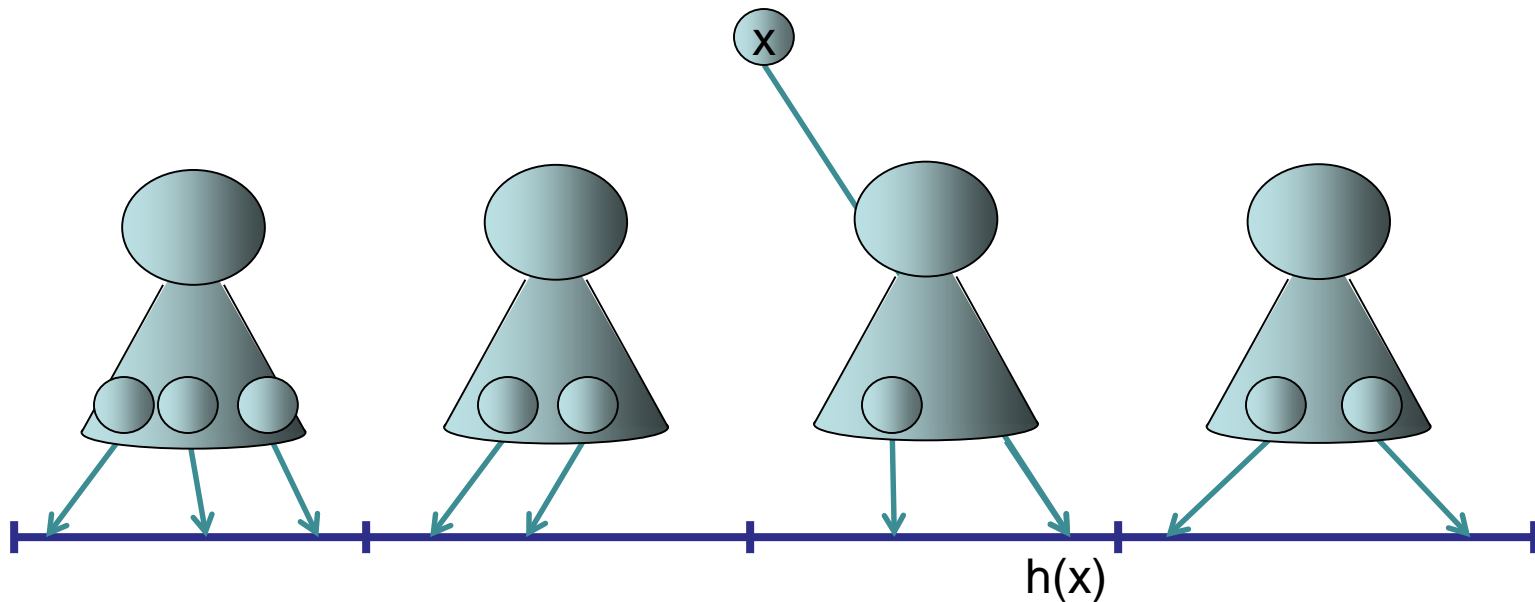
Konkrete Operation: **BuildList**: organisiere Prozesse in sortierter Liste gemäß ihrer Namen



Einleitung

Hashing: Beispiel für informationsorientierte Datenstruktur

Konkrete Operation: **Insert(x)**: füge Element x in Hash-tabelle ein

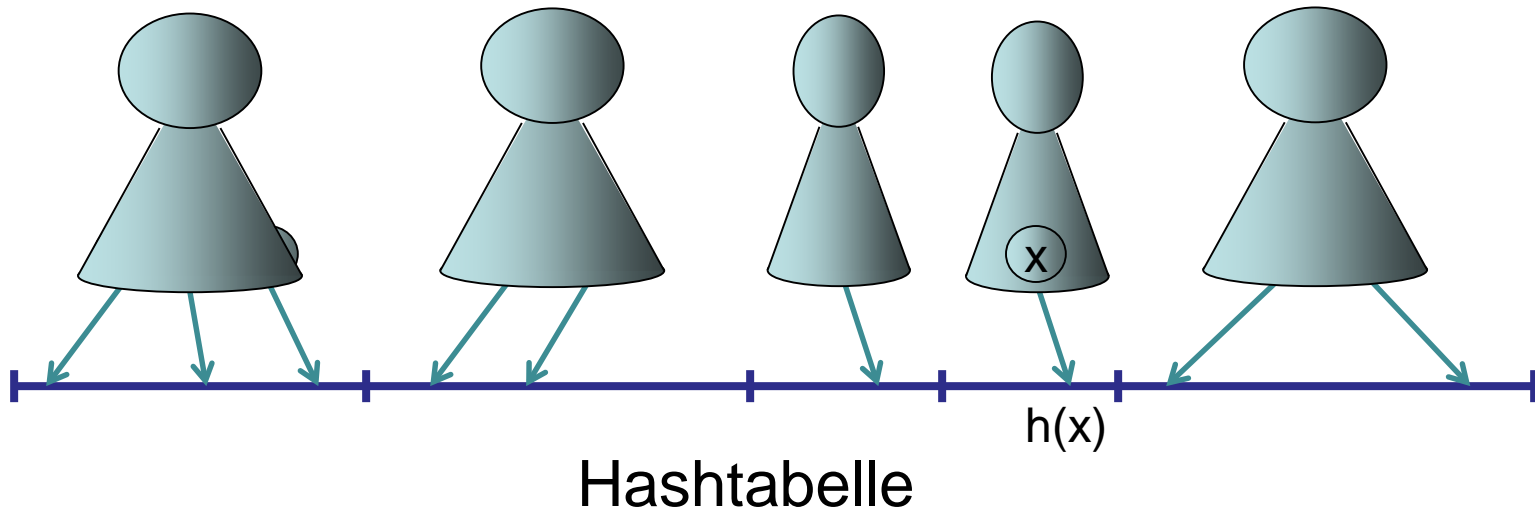


Hashtabelle

Einleitung

Hashing: Beispiel für informationsorientierte Datenstruktur

Konkrete Operation: **Insert(x)**: füge Element x in Hash-tabelle ein



Verteilte Algorithmen und Datenstrukturen

Inhalt:

1. Einführung
2. Netzwerktheorie
3. Designprinzipien für verteilte Algorithmen und Datenstrukturen
4. Einführung in die verteilte Programmierung
5. Prozessorientierte Datenstrukturen
 - a. (zyklische) Listen
 - b. De Bruijn Graphen (prozessorientiertes Hashing)
 - c. Skip Graphen (prozessorientierte Suche)
 - d. Delaunay Graphen (prozessorientierte 2-D Suche)
6. Informationsorientierte Datenstrukturen
 - a. Verteiltes Hashing
 - b. Verteilter Stack
 - c. Verteilte Queue
 - d. Verteilter Heap



Fragen?