technische universität
dortmund

# Präsentation Bachelor Arbeit:
# Simulation von Online Scheduling-Algorithmen für parallele Systeme

Hendrik Rassmann

November 02, 2020

TU Dortmund University - Department of Computer Science

# Übersicht

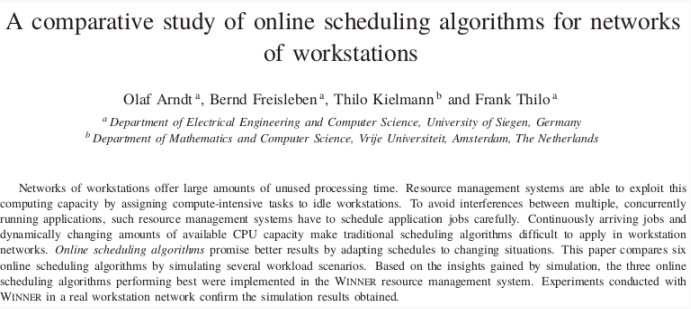
Problemstellung

Simulationsaufbau

Eigene Untersuchungen

Konklusion

# Startpunkt



A comparative study of online scheduling algorithms for networks of workstations

Olaf Arndt [a], Bernd Freisleben [a], Thilo Kielmann [b] and Frank Thilo [a]

[a] *Department of Electrical Engineering and Computer Science, University of Siegen, Germany*
[b] *Department of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, The Netherlands*

Networks of workstations offer large amounts of unused processing time. Resource management systems are able to exploit this computing capacity by assigning compute-intensive tasks to idle workstations. To avoid interferences between multiple, concurrently running applications, such resource management systems have to schedule application jobs carefully. Continuously arriving jobs and dynamically changing amounts of available CPU capacity make traditional scheduling algorithms difficult to apply in workstation networks. *Online scheduling algorithms* promise better results by adapting schedules to changing situations. This paper compares six online scheduling algorithms by simulating several workload scenarios. Based on the insights gained by simulation, the three online scheduling algorithms performing best were implemented in the WINNER resource management system. Experiments conducted with WINNER in a real workstation network confirm the simulation results obtained.

- Network
- Online
- Experiments ... confirm

# Problemstellung

## Problemstellung

Ausgehend von:

- Rechner/*Nodes* unterschiedlicher *Geschwindigkeiten* bilden zusammen ein (Rechen)*Cluster*

## Problemstellung

Ausgehend von:

- Rechner/*Nodes* unterschiedlicher *Geschwindigkeiten* bilden zusammen ein (Rechen)*Cluster*
- Aufträge/*Jobs* kommen im laufenden Betrieb rein (*online*)

## Problemstellung

Ausgehend von:

- Rechner/*Nodes* unterschiedlicher *Geschwindigkeiten* bilden zusammen ein (Rechen)*Cluster*
- Aufträge/*Jobs* kommen im laufenden Betrieb rein (*online*)
- Aufträge unterscheiden sich bez. *Bearbeitungszeit*, *Anzahl* an benötigten Knoten und *Ankunftszeitpunkt*

## Problemstellung

Ausgehend von:

- Rechner/*Nodes* unterschiedlicher *Geschwindigkeiten* bilden zusammen ein (Rechen)*Cluster*
- Aufträge/*Jobs* kommen im laufenden Betrieb rein (*online*)
- Aufträge unterscheiden sich bez. *Bearbeitungszeit*, *Anzahl* an benötigten Knoten und *Ankunftszeitpunkt*
- Aufträge sollen auf eine 'geeignete' Art und Weise bearbeitet werden

# Zielfunktionen [1]

- *makespan* "Zeit bis Feierabend"

[1] Kriterien von Arndt. et al.

## Zielfunktionen [1]

- *makespan* "Zeit bis Feierabend"
- *average flow time* "Supermarkt - Nur der Apfel? Gehen Sie doch gerne vor"

[1] Kriterien von Arndt. et al.

## Zielfunktionen [1]

- *makespan* "Zeit bis Feierabend"
- *average flow time* "Supermarkt - Nur der Apfel? Gehen Sie doch gerne vor"
- *average waiting time*

[1] Kriterien von Arndt. et al.

## Zielfunktionen [1]

- *makespan* "Zeit bis Feierabend"
- *average flow time* "Supermarkt - Nur der Apfel? Gehen Sie doch gerne vor"
- *average waiting time*
- *maximum waiting time* "Restaurant - Wenn ein Tisch nicht bedient wird, kommen Gäste nicht mehr wieder"

[1] Kriterien von Arndt. et al.

# Scheduling-Algorithmen (Referentiell Transparent)[2]

*Scheduling-Algorithmus* : *Warteliste* $\rightarrow$ *Auftrag*

[2] Auswahl von Arndt. et al.

# Scheduling-Algorithmen (Referentiell Transparent)[2]

*Scheduling-Algorithmus* : *Warteliste* $\rightarrow$ *Auftrag*

- First in First out (*FiFo*)

[2] Auswahl von Arndt. et al.

# Scheduling-Algorithmen (Referentiell Transparent)[2]

*Scheduling-Algorithmus* : *Warteliste* $\rightarrow$ *Auftrag*

- First in First out (*FiFo*)
- Shortest Processing Time first (*SPT*)

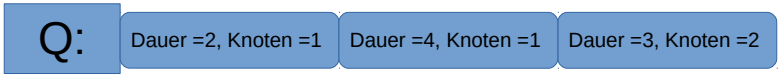
[2] Auswahl von Arndt. et al.

# Scheduling-Algorithmen (Referentiell Transparent)[2]

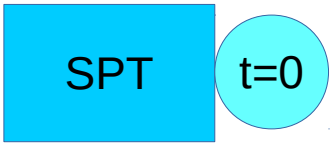
*Scheduling-Algorithmus* : *Warteliste* $\rightarrow$ *Auftrag*

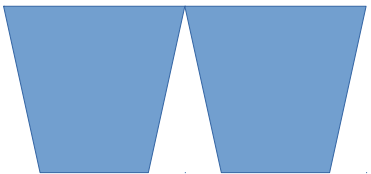
- First in First out (*FiFo*)
- Shortest Processing Time first (*SPT*)
- Greates Processing Time first (*GPT*)

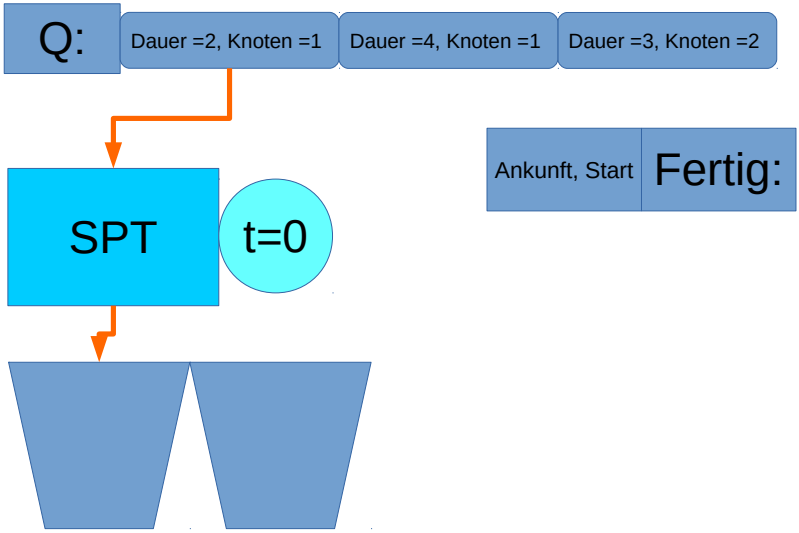
[2] Auswahl von Arndt. et al.

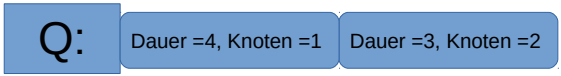

Q:
Dauer =2, Knoten =1
Dauer =4, Knoten =1
Dauer =3, Knoten =2


Ankunft, Start
Fertig:

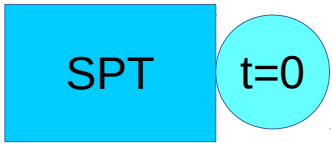

SPT
t=0



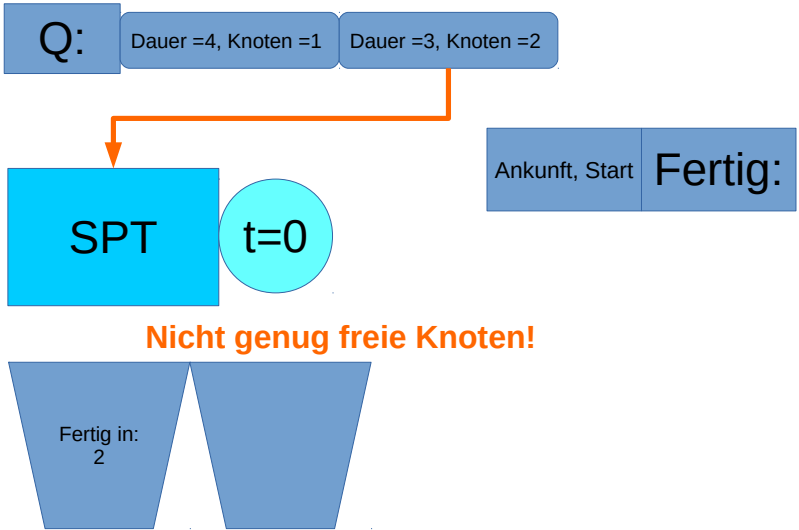

Q:
Dauer =2, Knoten =1
Dauer =4, Knoten =1
Dauer =3, Knoten =2
Ankunft, Start
Fertig:
SPT
t=0

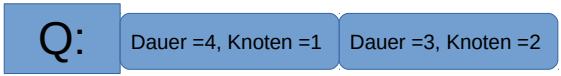

Q:
Dauer =4, Knoten =1
Dauer =3, Knoten =2


Ankunft, Start
Fertig:

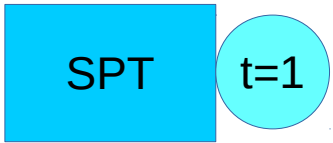

SPT
t=0


Fertig in:
2

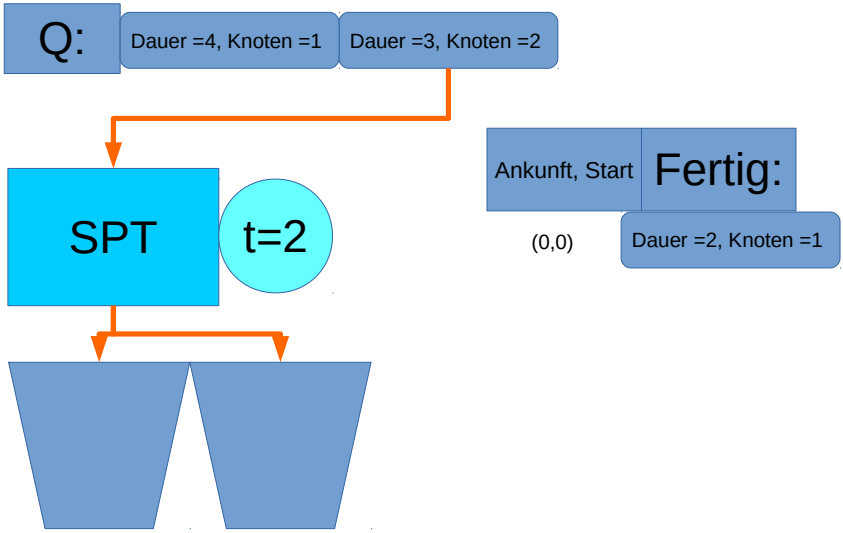

Q:
Dauer =4, Knoten =1
Dauer =3, Knoten =2
Ankunft, Start
Fertig:
SPT
t=0
Nicht genug freie Knoten!
Fertig in:
2

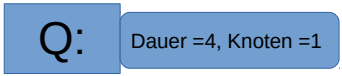

Q:
Dauer =4, Knoten =1
Dauer =3, Knoten =2

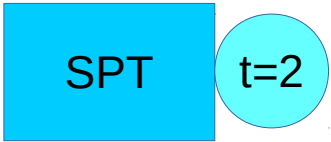

Ankunft, Start
Fertig:

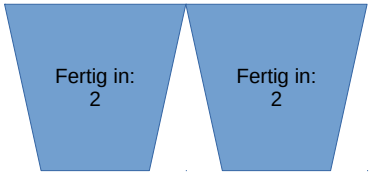

SPT
t=1


Fertig in:
1

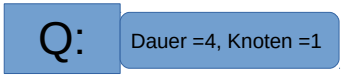

Q:
Dauer =4, Knoten =1
Dauer =3, Knoten =2
SPT
t=2
Ankunft, Start
Fertig:
(0,0)
Dauer =2, Knoten =1

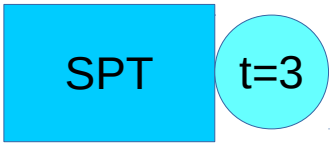

Q:
Dauer =4, Knoten =1

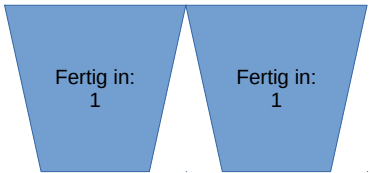

SPT
t=2


Fertig in:
2
Fertig in:
2

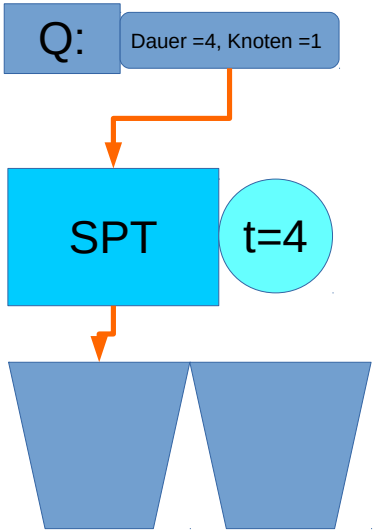

Ankunft, Start
Fertig:
(0,0)
Dauer =2, Knoten =1


Q:
Dauer =4, Knoten =1


SPT
t=3


Fertig in:
1
Fertig in:
1


Ankunft, Start
Fertig:
(0,0)
Dauer =2, Knoten =1




| Ankunft, Start | Fertig: |
|---|---|
| (0,0) | Dauer =2, Knoten =1 |
| (0,2) | Dauer =3, Knoten =2 |

Q:

SPT

t=4

Fertig in:
4

| Ankunft, Start | Fertig: |
| --- | --- |
| (0,0) | Dauer =2, Knoten =1 |
| (0,2) | Dauer =3, Knoten =2 |

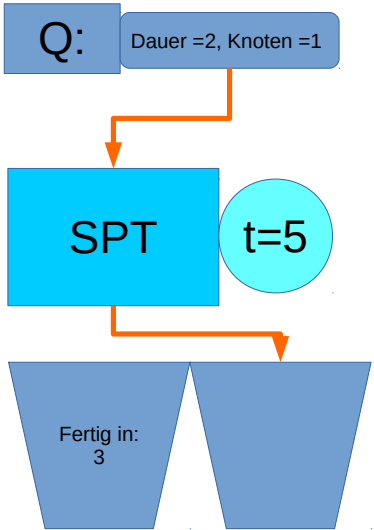
Q:

SPT

t=5

Fertig in:
3

| Ankunft, Start | Fertig: |
| --- | --- |
| (0,0) | Dauer =2, Knoten =1 |
| (0,2) | Dauer =3, Knoten =2 |




| Ankunft, Start | Fertig: |
|---|---|
| (0,0) | Dauer =2, Knoten =1 |
| (0,2) | Dauer =3, Knoten =2 |

Q:

SPT

t=5

| Ankunft, Start | Fertig: |
| --- | --- |
| (0,0) | Dauer =2, Knoten =1 |
| (0,2) | Dauer =3, Knoten =2 |

Fertig in:
3

Fertig in:
2

Q:

SPT

t=6

| Ankunft, Start | Fertig: |
| --- | --- |
| (0,0) | Dauer =2, Knoten =1 |
| (0,2) | Dauer =3, Knoten =2 |

Fertig in:
2

Fertig in:
1

Q:

SPT

t=7

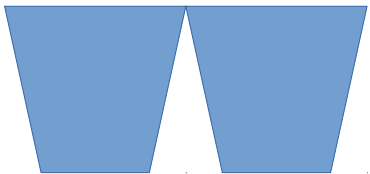
Fertig in:
1

| Ankunft, Start | Fertig: |
| --- | --- |
| (0,0) | Dauer =2, Knoten =1 |
| (0,2) | Dauer =3, Knoten =2 |
| (5,5) | Dauer =2, Knoten =1 |

Q:

SPT

t=8



| Ankunft, Start | Fertig: |
|---|---|
| (0,0) | Dauer =2, Knoten =1 |
| (0,2) | Dauer =3, Knoten =2 |
| (5,5) | Dauer =2, Knoten =1 |
| (0,4) | Dauer =4, Knoten =1 |

# Scheduling-Algorithmen (Referentiell Intransparent)

- Random

## Scheduling-Algorithmen (Referentiell Intransparent)

- Random
- First Fit (Wähle den am längsten wartende Auftrag, der sofort gestartet werden kann)

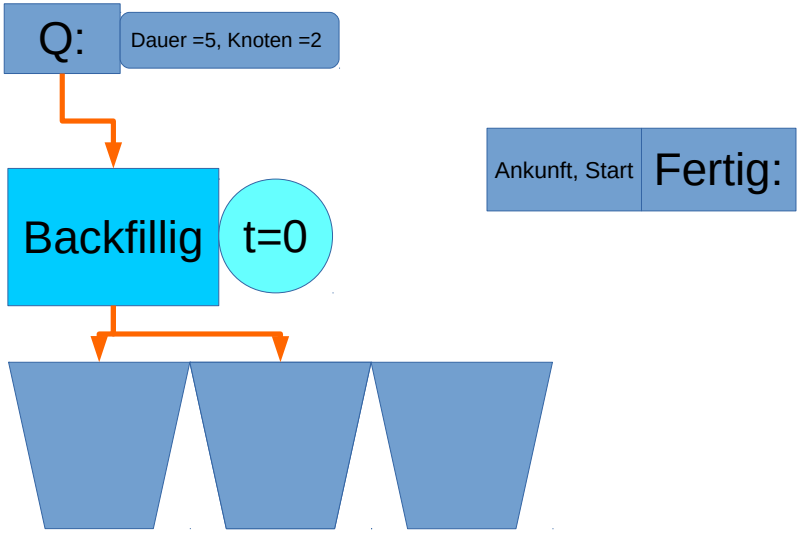
## Scheduling-Algorithmen (Referentiell Intransparent)

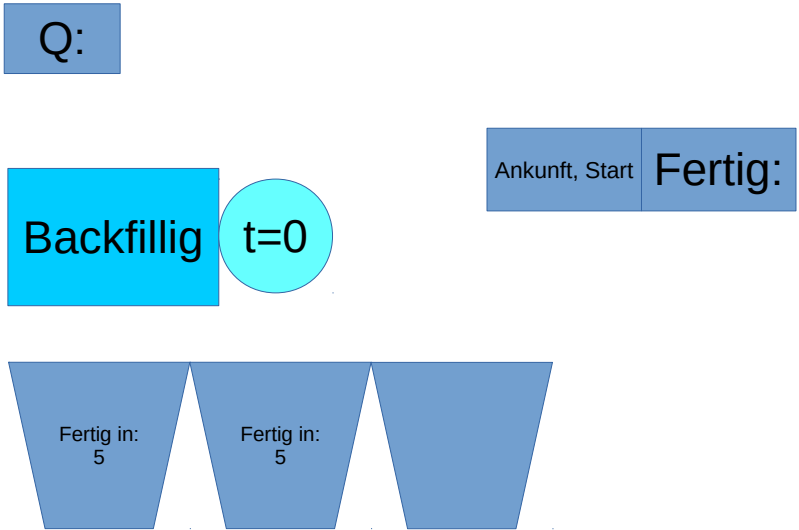
- Random
- First Fit (Wähle den am längsten wartende Auftrag, der sofort gestartet werden kann)
- Backfilling (Wähle einen startbaren Auftrag, der wenn er jetzt gestartet wird, terminiert bevor der am längsten wartende Auftrag starten wird)

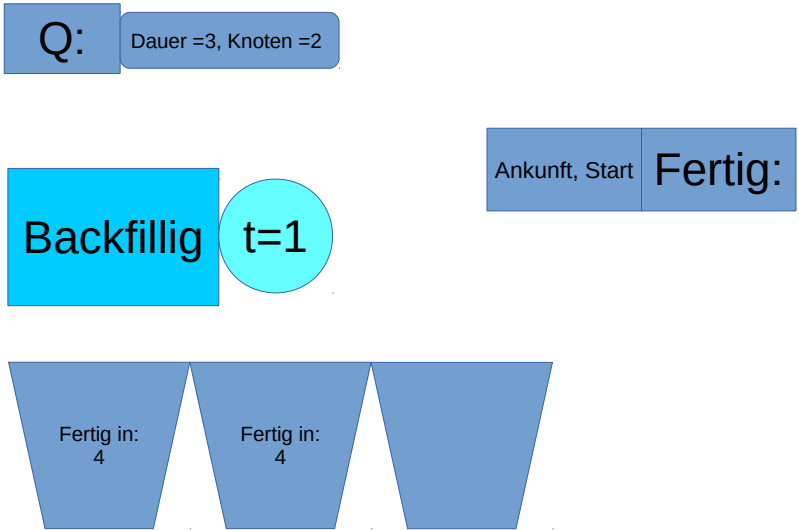
## Scheduling-Algorithmen (Referentiell Intransparent)

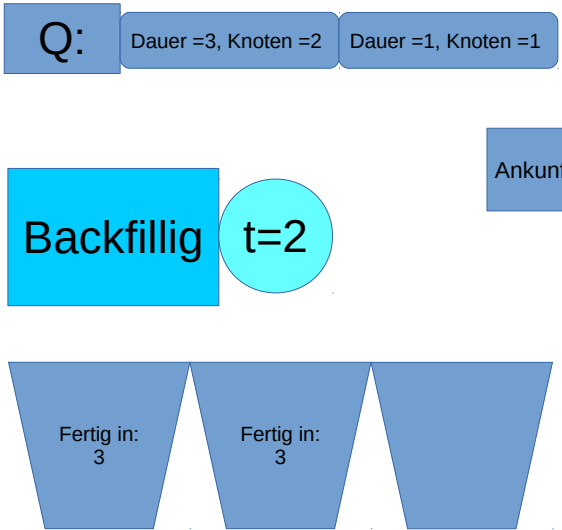
- Random
- First Fit (Wähle den am längsten wartende Auftrag, der sofort gestartet werden kann)
- Backfilling (Wähle einen startbaren Auftrag, der wenn er jetzt gestartet wird, terminiert bevor der am längsten wartende Auftrag starten wird)
- Sichtweise der Funktionalen Programmierung: FirstFit, Backfilling sind $FiFo(Filter(Warteschlange))$

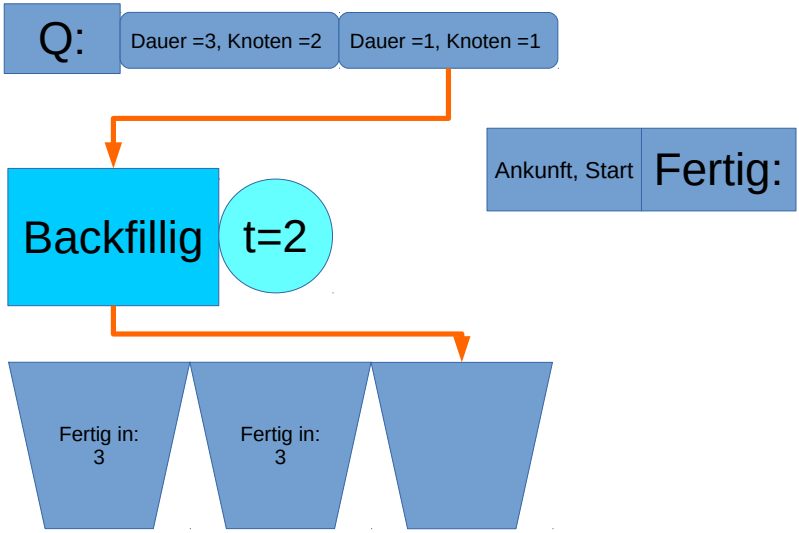
# Beispiel Backfilling

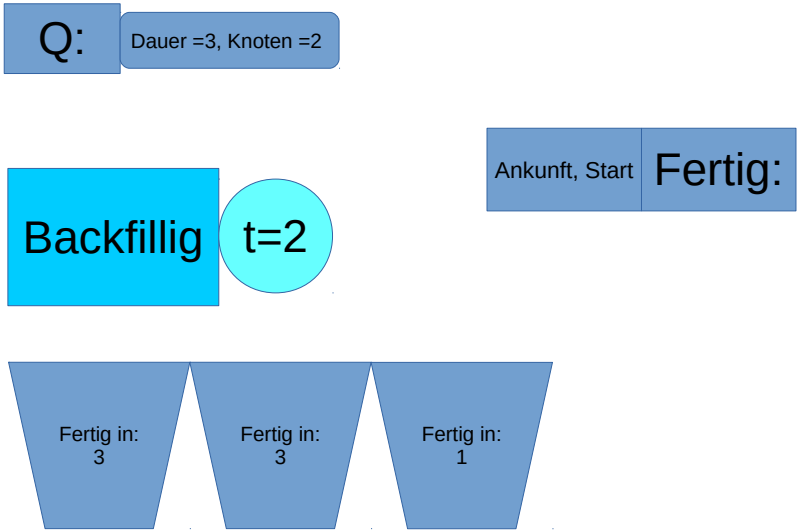

Q:
Dauer =5, Knoten =2
Backfillig
t=0
Ankunft, Start
Fertig:

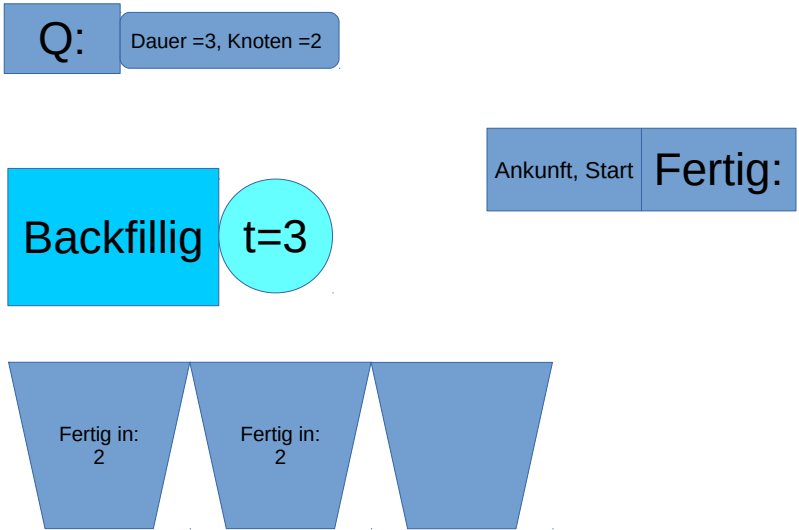

Q:
Ankunft, Start
Fertig:
Backfillig
t=0
Fertig in:
5
Fertig in:
5

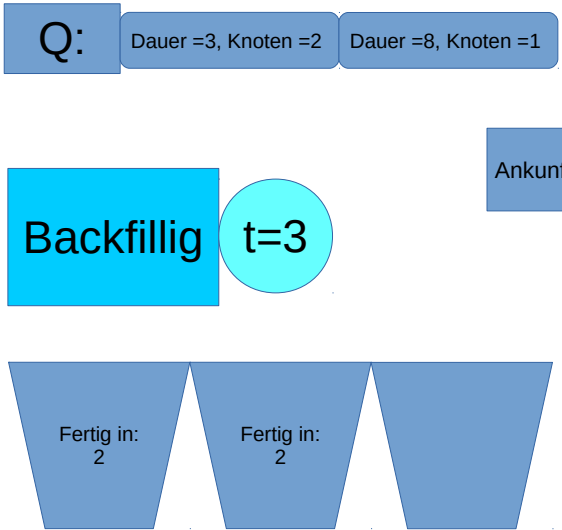

Q:
Dauer =3, Knoten =2
Ankunft, Start
Fertig:
Backfillig
t=1
Fertig in:
4
Fertig in:
4

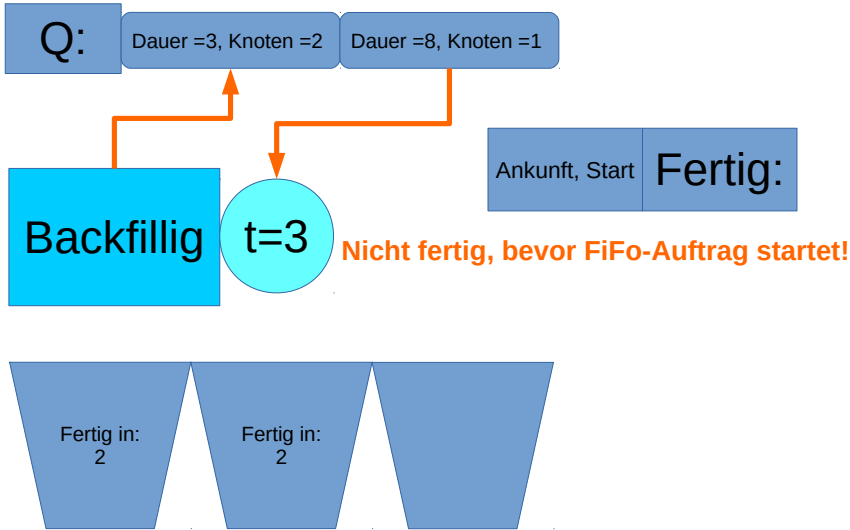

Q:
Dauer =3, Knoten =2
Dauer =1, Knoten =1
Ankunft, Start
Fertig:
Backfillig
t=2
Fertig in:
3
Fertig in:
3


Q:
Dauer =3, Knoten =2
Dauer =1, Knoten =1
Ankunft, Start
Fertig:
Backfillig
t=2
Fertig in:
3
Fertig in:
3


Q:
Dauer =3, Knoten =2
Ankunft, Start
Fertig:
Backfillig
t=2
Fertig in:
3
Fertig in:
3
Fertig in:
1


Q:
Dauer =3, Knoten =2
Ankunft, Start
Fertig:
Backfillig
t=3
Fertig in:
2
Fertig in:
2


Q:
Dauer =3, Knoten =2
Dauer =8, Knoten =1
Ankunft, Start
Fertig:
Backfillig
t=3
Fertig in:
2
Fertig in:
2


Q:
Dauer =3, Knoten =2
Dauer =8, Knoten =1
Ankunft, Start
Fertig:
Backfillig
t=3
Nicht fertig, bevor FiFo-Auftrag startet!
Fertig in:
2
Fertig in:
2

# Simulationsaufbau

## Simulations Paradigma[3]

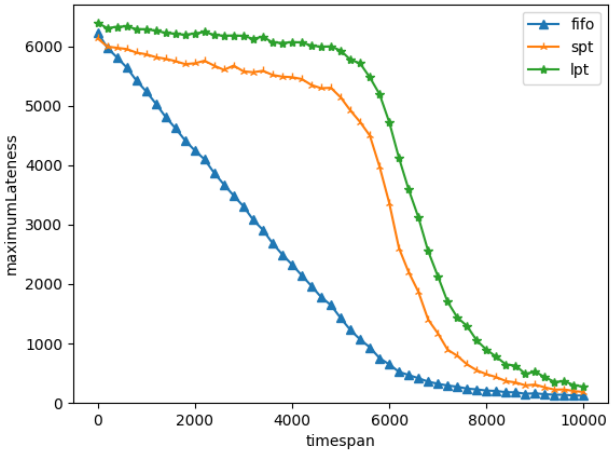
- Aktivitätsorientiert (Wettersystem, Physik Simulation in Videospielen)
- Ereignisorientiert (Fahrplan)
- Prozessorientiert (Parallele Programmierung)

[3] Matloff, Norm: Introduction to discrete-event simulati- on and the simpy language. In: Davis, CA. Dept of Com- puter Science. University of California at Davis. Retrieved on August 2 (2008), Nr. 2009, S. 133

## Simulation und Auswertung

- Arndt et al. designen 10 Experimente, um unterschiedliches Verhalten der Algorithmen zu demonstrieren
- Variieren einen Parameter und vergleichen gewählte Zielfunktionen
- "Bester Algorithmus" wird qualitativ bestimmt.

# Beispiel




- 250 Aufträge,1-50 k s Bearbeitungszeit
- 10 Knoten
- timespan - Spätester Ankunftszeitpunkt wird variiert
- Mittel von 100 Läufen

# Eigene Untersuchungen

## Optimistic Backfilling

Wähle den am längsten wartenden Auftrag, falls startbar. Sonst wähle einen Auftrag, der terminiert, bevor FiFo startet, oder einen Auftrag, der weniger Knoten benötigt, als übrig bleiben werden, wenn FiFo startet.

## Optimistic Backfilling Nachteil?

- Gibt es einen Fall, in dem sich dies negativ auf die maximum latenes auswirkt?

## Optimistic Backfilling Nachteil?

- Gibt es einen Fall, in dem sich dies negativ auf die maximum latenes auswirkt?
- Is fifo_optimistic allways better than fifo_backfill by maximumLateness?
  No! counterexample:
  queueintT, processingT, realProcessingT, degreeOfParall
  id: 0, qT: 1, pT: 3, rPT: 3, doP: 1
  id: 1, qT: 0, pT: 3, rPT: 3, doP: 2
  id: 2, qT: 0, pT: 1, rPT: 1, doP: 2
  id: 3, qT: 0, pT: 1, rPT: 1, doP: 3

# Optimistic Backfilling Nachteil?

```
queueintT, processingT, realProcessingT, degreeOfParallelism
id: 0, qT: 1, pT: 3, rPT: 3, doP: 1
id: 1, qT: 0, pT: 3, rPT: 3, doP: 2
id: 2, qT: 0, pT: 1, rPT: 1, doP: 2
id: 3, qT: 0, pT: 1, rPT: 1, doP: 3

maximumLateness of fifo\_optimistic: 4
[0]:112|-3
[1]:112|-3
[2]:-00|03

maximumLateness of fifo\_backfill: 3
[0]:112|3--|-
[1]:11-|300|0
[2]:--2|3--|-
```

# Konservativ vs Optimistisch

- Hier Paper Referenz einfgen

## Gegenbeispiele Automatisch Finden

- Anschauliche Beispiele, in denen verschd. Algorithmen verschd. Entscheidungen treffen, helfen Intuition aufzubauen
- Idealerweise "For Free" aus dem Simulationsmodell ableitbar, ohne das Modell zu übersetzten
- Lösung: Property Based Testing

# Property Based Testing [4]

- Eigenschaft einer Funktion, die immer gelten soll
- Zufällige Eingaben ausprobiert, bis Eigenschaft verletzt wird
- Beispiel verkleinern (shrinking)

[4] Hughes, John ; Claessen, Koen: Quickcheck: a light- weight tool for random testing of haskell programs. In: Proceedings of the Fifth ACM SIGPLAN International Confe- rence on Functional Programming, ICFP, 2000, S. 268279

## Beispiel

- Funktion: *reverse*
- Eigenschaft: $reverse(x) == x$
- Zufällige Eingaben: [] , [1], [2,2], [4,2]↯
- Verkleinern: 2 Möglichkeiten: Element löschen oder Element verkleinern.
- Verkleiern (löschen): [4,2] $\rightarrow$ [4] ↯
- Element Verkleinern: [4,2] $\rightarrow$ [1,2] $\rightarrow$ ... $\rightarrow$ [0,1]

## Neue Möglichkeiten

- Testen der Implementation: FiFo nie besser als Konservatives Backfilling
- Beliebig Eigenschaften Aufbauen: besser nach mehreren Zielfunktionen.
- Monotonität: System trifft bessere Entscheidungen, mit mehr Informationen: Frühere Ankuntszeiten $\Longrightarrow$ nicht schlechtere Performance
- Schneller Aufbau von Intuition bez. neuer Algorithmen

## Neue Algorithmen

- "-Fit": Smallest-Fit, Greatest-Fit
- Backfilling: Smallest-Backfill, Greatest-Backfill
- Optimistisches Backfilling nach Smallest,Greatest
- Backfilling nach zwei Funktionen; Wähle besten Kandidaten nach Funktion 1, falls nicht startbar, w¨hle nach Funktion 2, ohne besten Kandidaten zu benachteiligen
- Analog fr Optimistisches Backfilling

# Qualitative Analyse nicht mehr Möglich

content...

# PBT-Score

| J+k+o | fifo | spt | lpt | Fifo-fit | Fifo-back | Fifo-optim | Spt-fit | Spt-back | Lpt-fit | Lpt-back |
|---|---|---|---|---|---|---|---|---|---|---|
| fifo | + | 3+2+0 | 3+2+1 | 6+2+2 | X | 8+3+4 | 3+4+0 | 3+2+0 | 3+2+1 | 3+9+1 |
| spt | 3+2+1 | + | 3+2+1 | 4+2+1 | 4+2+1 | 4+2+1 | 6+8+0 | 6+10+5 | 3+3+1 | 4+5+3 |
| lpt | 3+2+0 | 3+2+0 | + | 4+4+2 | 3+2+0 | 3+2+2 | 3+2+0 | 3+2+0 | 4+2+2 | 7+5+2 |
| Fifo-fit | 3+2+2 | 3+2+2 | 3+2+0 | + | 5+4+2 | 3+2+1 | 3+2+0 | 3+2+0 | 3+2+1 | 4+3+0 |
| Fifo-back | 5+2+0 | 3+2+2 | 3+2+0 | 5+6+4 | + | 7+6+2 | 3+3+0 | 3+2+2 | 3+2+1 | 3+2+1 |
| Fifo-optim | 3+3+1 | 3+2+0 | 3+2+0 | 4+2+1 | 3+4+0 | + | 3+4+0 | 3+2+2 | 3+2+1 | 3+2+1 |
| Spt-fit | 3+2+2 | 3+2+2 | 3+2+0 | 3+2+1 | 3+2+2 | 3+2+0 | + | 3+3+0 | 3+2+1 | 5+5+4 |
| Spt-back | 3+2+0 | 5+2+2 | 3+2+1 | 3+2+1 | 3+2+1 | 3+2+1 | 5+4+2 | + | 3+2+1 | 3+2+1 |
| Lpt-fit | 3+2+2 | 3+2+1 | 3+2+0 | 3+2+0 | 3+2+1 | 3+2+0 | 3+2+0 | 3+2+0 | + | 4+4+3 |
| Lpt-back | 3+2+1 | 3+2+2 | 3+3+0 | 3+2+0 | 3+4+1 | 4+2+1 | 3+2+0 | 3+2+0 | 5+4+2 | + |

- 250 Aufträge,1-50 k s Bearbeitungszeit
- 10 Knoten
- timespan - Spätester Ankunftszeitpunkt wird variiert
- Mittel von 100 Läufen

# Konklusion