

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

Hendrik Rassmann

November 02, 2020

TU Dortmund University - Department of Computer Science

Problemstellung

Eigene Untersuchungen

Konklusion

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

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*

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*

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*

- *makespan*
- *average flow time*
- *average waiting time*
- *maximum waiting time*

¹Kriterien von Arndt. et al.

- *makespan*
- *average flow time*
- *average waiting time*
- *maximum waiting time*

¹Kriterien von Arndt. et al.

- *makespan*
- *average flow time*
- *average waiting time*
- *maximum waiting time*

¹Kriterien von Arndt. et al.

- *makespan*
- *average flow time*
- *average waiting time*
- *maximum waiting time*

¹Kriterien von Arndt. et al.

Scheduling-Algorithmus : Warteliste \rightarrow Auftrag

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

²Auswahl von Arndt. et al.

Scheduling-Algorithmus : Warteliste \rightarrow Auftrag

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

²Auswahl von Arndt. et al.

Scheduling-Algorithmus : Warteliste \rightarrow Auftrag

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

²Auswahl von Arndt. et al.

Scheduling-Algorithmus : Warteliste \rightarrow Auftrag

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

²Auswahl von Arndt. et al.

Q:

Dauer =2, Knoten =1

Dauer =4, Knoten =1

Dauer =3, Knoten =2

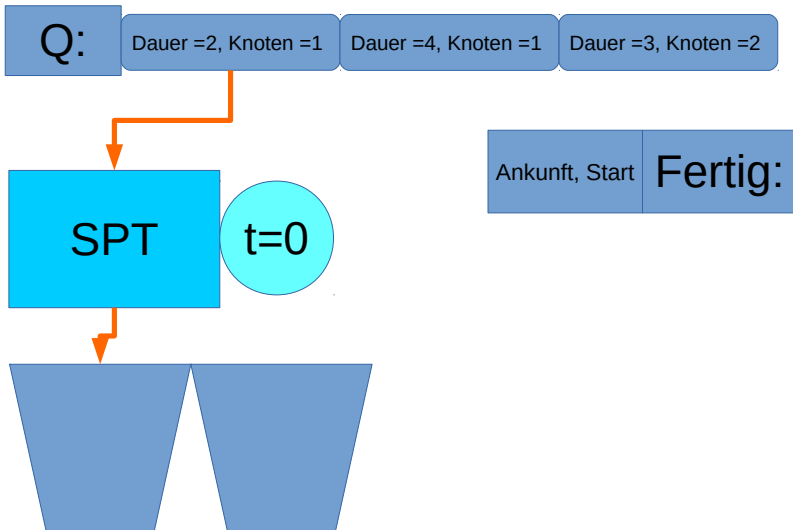
SPT

t=0

Ankunft, Start

Fertig:





Q:

Dauer =4, Knoten =1

Dauer =3, Knoten =2

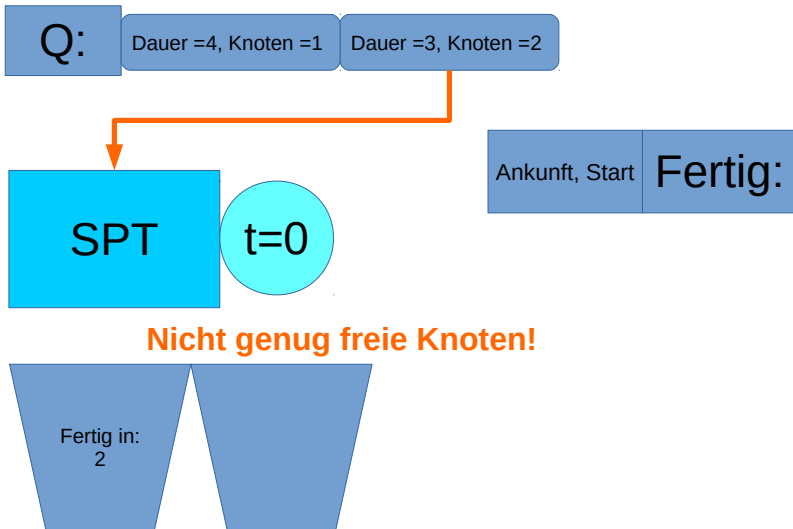
SPT

t=0

Ankunft, Start

Fertig:

Fertig in:
2



Q:

Dauer =4, Knoten =1

Dauer =3, Knoten =2

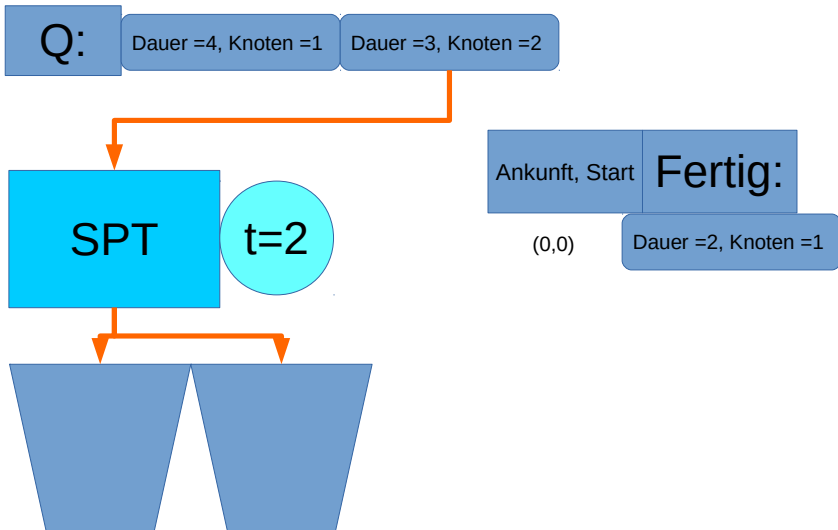
SPT

t=1

Ankunft, Start

Fertig:

Fertig in:
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

Q: Dauer =4, Knoten =1

SPT

t=4



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

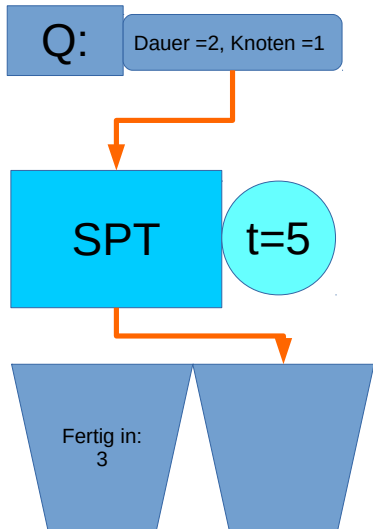
Fertig:

(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

Fertig in:
3

Fertig in:
2

Ankunft, Start

Fertig:

(0,0)

Dauer =2, Knoten =1

(0,2)

Dauer =3, Knoten =2

Q:

SPT

t=6

Fertig in:
2

Fertig in:
1

Ankunft, Start

Fertig:

(0,0)

Dauer =2, Knoten =1

(0,2)

Dauer =3, Knoten =2

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
- **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:**
 $FirstFit(Q) = FiFo(Filter(Q))$

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:**
 $FirstFit(Q) = FiFo(Filter(Q))$

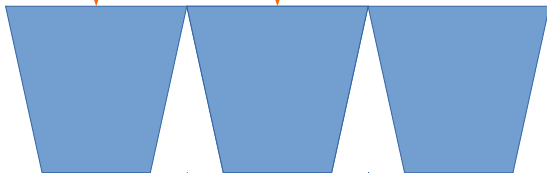
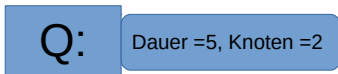
Scheduling-Algorithmen (Referentiell Intransparent)

- 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:
 $FirstFit(Q) = FiFo(Filter(Q))$

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:**
$$FirstFit(Q) = FiFo(Filter(Q))$$

Beispiel Backfilling



Q:

Backfilling

t=0

Fertig in:
5

Fertig in:
5

Q:

Dauer =3, Knoten =2

Backfilling

t=1

Fertig in:
4

Fertig in:
4

Q:

Dauer =3, Knoten =2

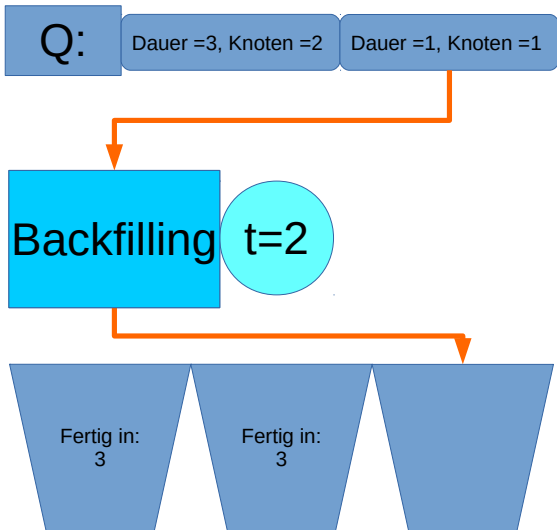
Dauer =1, Knoten =1

Backfilling

t=2

Fertig in:
3

Fertig in:
3



Q:

Dauer =3, Knoten =2

Backfilling

t=2

Fertig in:
3

Fertig in:
3

Fertig in:
1

Q:

Dauer =3, Knoten =2

Backfilling

t=3

Fertig in:
2

Fertig in:
2

Q:

Dauer =3, Knoten =2

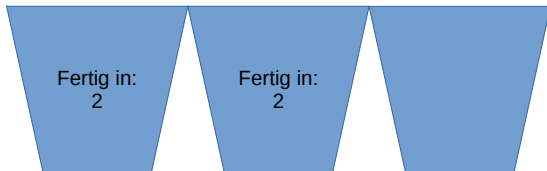
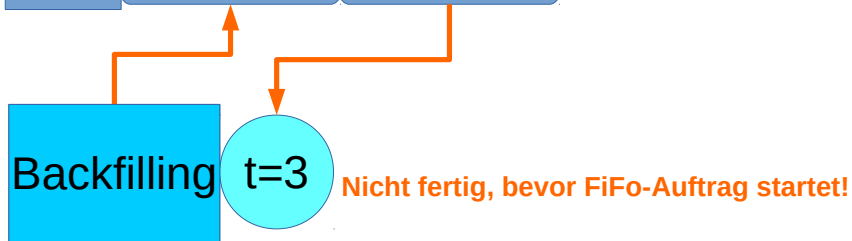
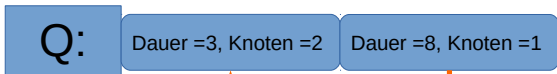
Dauer =8, Knoten =1

Backfilling

t=3

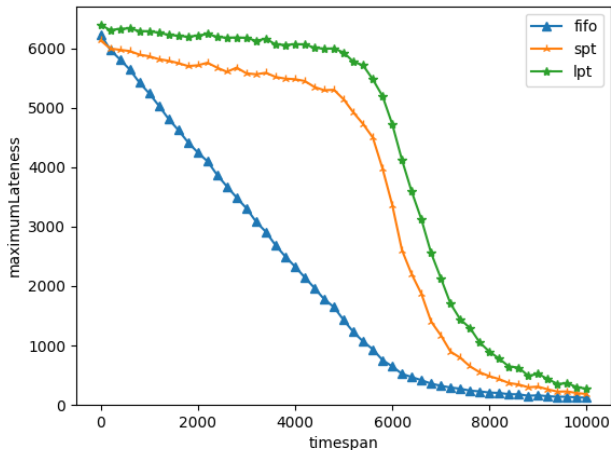
Fertig in:
2

Fertig in:
2



- 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. Make span am wichtigsten

Beispiel



- 250 Aufträge
- 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 immer besser?

```
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--|-
```

- "Results are that the performance (...) are practically identical."³
- "This implies that there is significant difference between normalized distributions of job runtime estimates, and actual running times."⁴

³Feitelson, Dror G. ; Weil, Ahuva M.: Utilization and predictability in scheduling the IBM SP2 with backfilling. In: Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing IEEE, 1998, S. 542546

⁴Keleher, Peter J. ; Zotkin, Dmitry ; Perkovic, Dejan: Attacking the bottlenecks of backfilling schedulers. In: Cluster Computing 3 (2000), Nr. 4, S. 245254

- "Results are that the performance (...) are practically identical."³
- "This implies that there is significant difference between normalized distributions of job runtime estimates, and actual running times."⁴

³Feitelson, Dror G. ; Weil, Ahuva M.: Utilization and predictability in scheduling the IBM SP2 with backfilling. In: Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing IEEE, 1998, S. 542546

⁴Keleher, Peter J. ; Zotkin, Dmitry ; Perkovic, Dejan: Attacking the bottlenecks of backfilling schedulers. In: Cluster Computing 3 (2000), Nr. 4, S. 245254

- Anschauliche Beispiele bauen Intuition auf
- Am besten "For Free"
- Lösung: Property Based Testing

- Anschauliche Beispiele bauen Intuition auf
- Am besten "For Free"
- Lösung: Property Based Testing

- Anschauliche Beispiele bauen Intuition auf
- Am besten "For Free"
- Lösung: Property Based Testing

- Funktion: *reverse*
- Eigenschaft: $\text{reverse}(x) == x$
- Zufällige Eingaben: $[]$, $[1]$, $[2,2]$, $[4,2]$ ⚡
- Verkleinern (löschen): $[4,2] \rightarrow [4]$ ⚡
- Element Verkleinern: $[4,2] \rightarrow [1,2] \rightarrow \dots \rightarrow [0,1]$

⁵Hughes, John ; Claessen, Koen: Quickcheck: a light- weight tool for random testing of haskell programs. In: Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP, 2000, S. 268279

- Funktion: *reverse*
- Eigenschaft: $\text{reverse}(x) == x$
- Zufällige Eingaben: $[]$, $[1]$, $[2,2]$, $[4,2]$ ⚡
- Verkleinern (löschen): $[4,2] \rightarrow [4]$ ⚡
- Element Verkleinern: $[4,2] \rightarrow [1,2] \rightarrow \dots \rightarrow [0,1]$

⁵Hughes, John ; Claessen, Koen: Quickcheck: a light- weight tool for random testing of haskell programs. In: Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP, 2000, S. 268279

- Funktion: *reverse*
- Eigenschaft: $\text{reverse}(x) == x$
- Zufällige Eingaben: $[]$, $[1]$, $[2,2]$, $[4,2]$ ⚡
- Verkleinern (löschen): $[4,2] \rightarrow [4]$ ⚡
- Element Verkleinern: $[4,2] \rightarrow [1,2] \rightarrow \dots \rightarrow [0,1]$

⁵Hughes, John ; Claessen, Koen: Quickcheck: a light- weight tool for random testing of haskell programs. In: Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP, 2000, S. 268279

- Funktion: *reverse*
- Eigenschaft: $\text{reverse}(x) == x$
- Zufällige Eingaben: $[]$, $[1]$, $[2,2]$, $[4,2]$ ⚡
- Verkleinern (löschen): $[4,2] \rightarrow [4]$ ⚡
- Element Verkleinern: $[4,2] \rightarrow [1,2] \rightarrow \dots \rightarrow [0,1]$

⁵Hughes, John ; Claessen, Koen: Quickcheck: a light- weight tool for random testing of haskell programs. In: Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP, 2000, S. 268279

- Funktion: *reverse*
- Eigenschaft: $\text{reverse}(x) == x$
- Zufällige Eingaben: $[]$, $[1]$, $[2,2]$, $[4,2]$ ⚡
- Verkleinern (löschen): $[4,2] \rightarrow [4]$ ⚡
- Element Verkleinern: $[4,2] \rightarrow [1,2] \rightarrow \dots \rightarrow [0,1]$

⁵Hughes, John ; Claessen, Koen: Quickcheck: a light- weight tool for random testing of haskell programs. In: Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming, ICFP, 2000, S. 268279

- Testen der Implementation: FiFo nie besser als Konservatives Backfilling
- Monotonität: System trifft schlechtere Entscheidungen, mit mehr Informationen?
- Schneller Aufbau von Intuition bez. neuer Algorithmen

- Testen der Implementation: FiFo nie besser als Konservatives Backfilling
- Monotonität: System trifft schlechtere Entscheidungen, mit mehr Informationen?
- Schneller Aufbau von Intuition bez. neuer Algorithmen

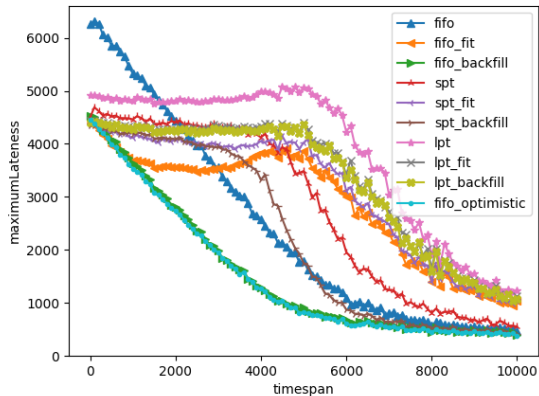
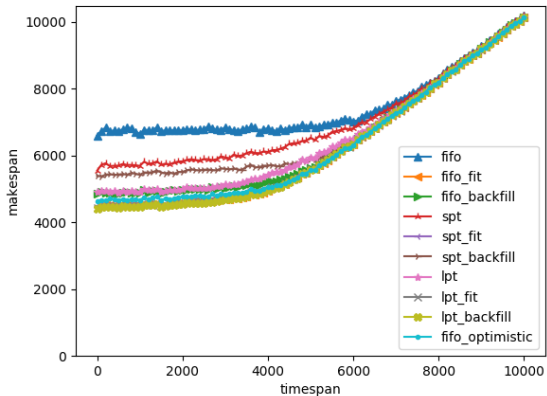
- Testen der Implementation: FiFo nie besser als Konservatives Backfilling
- Monotonität: System trifft schlechtere Entscheidungen, mit mehr Informationen?
- Schneller Aufbau von Intuition bez. neuer Algorithmen

- "-Fit": Smallest-Fit, Greatest-Fit
- (Optimistisches)-Backfilling: Smallest-Backfill, Greatest-Backfill
- (Optimistisches)-Backfilling nach zwei Funktionen; Wähle besten Kandidaten nach Funktion 1, falls nicht startbar, wähle nach Funktion 2, ohne besten Kandidaten zu benachteiligen

- "-Fit": Smallest-Fit, Greatest-Fit
- (Optimistisches)-Backfilling: Smallest-Backfill, Greatest-Backfill
- (Optimistisches)-Backfilling nach zwei Funktionen; Wähle besten Kandidaten nach Funktion 1, falls nicht startbar, wähle nach Funktion 2, ohne besten Kandidaten zu benachteiligen

- "-Fit": Smallest-Fit, Greatest-Fit
- (Optimistisches)-Backfilling: Smallest-Backfill, Greatest-Backfill
- (Optimistisches)-Backfilling nach zwei Funktionen; Wähle besten Kandidaten nach Funktion 1, falls nicht startbar, wähle nach Funktion 2, ohne besten Kandidaten zu benachteiligen

Vergleich mit Simulation



Qualitative Analyse nicht mehr Möglich

- 80 Scheduling Algorithmen kombinierbar
- Zu jedem Ergebnis lässt sich ein passendes Experiment finden
- Algorithmen ohne konkretes Experiment vergleichen?

Qualitative Analyse nicht mehr Möglich

- 80 Scheduling Algorithmen kombinierbar
- Zu jedem Ergebnis lässt sich ein passendes Experiment finden
- Algorithmen ohne konkretes Experiment vergleichen?

Qualitative Analyse nicht mehr Möglich

- 80 Scheduling Algorithmen kombinierbar
- Zu jedem Ergebnis lässt sich ein passendes Experiment finden
- Algorithmen ohne konkretes Experiment vergleichen?

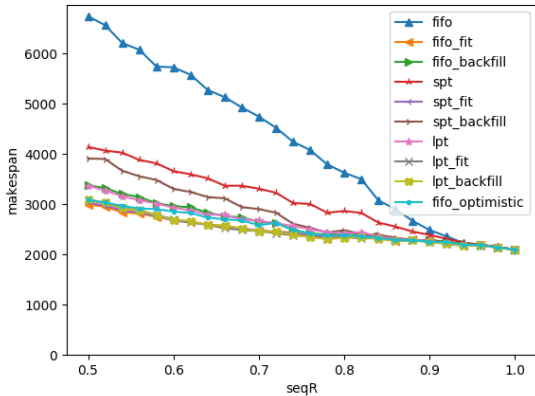
PBT-Score

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	+

PBT-Score

	V/G	fifo	spt	lpt	Fifo-fit	Fifo-back	Fifo-optim	Spt-fit	Spt-back	Lpt-fit	Lpt-back	
0.756	fifo	0	5	6	10	nn	15	7	5	6	13	8.375
0.667	spt	6	0	6	7	7	7	14	21	7	12	9.667
0.766	lpt	5	5	0	10	5	7	5	5	8	14	7.111
1.203	Fifo-fit	7	7	5	0	11	6	5	5	6	7	6.556
0.867	Fifo-back	7	7	5	15	0	15	6	7	6	6	8.222
1.281	Fifo-optim	7	5	5	7	7	0	7	7	6	6	6.333
1.032	Spt-fit	7	7	5	6	7	5	0	6	6	14	7
1.082	Spt-back	5	9	6	6	6	6	11	0	6	6	6.778
1.127	Lpt-fit	7	6	5	5	6	5	5	5	0	11	6.111
1.483	Lpt-back	6	7	6	5	8	7	5	5	11	0	6.667
		6.333	6.444	5.444	7.889	7.125	8.111	7.222	7.333	6.889	9.889	

Vergleich PBT-score und Simulation⁶



algorithm	area under curve	pbt score	a.u.c. Rank	pbt Rank
fifo	25862.6322330097	0.756218905472637	1	9
spt	23798.8361165049	0.666666666666667	2	10
lpt	23054.2840776699	0.765625	4	8
Fifo-fit	21039.4811650485	1.20338983050847	8	3
Fifo-back	21378.4883018868	0.866554054054054	5	8
Fifo-optim	21008.5106796117	1.28070175438597	9	2
Spt-fit	21211.5052427184	1.03174603174603	6	6
Spt-back	23685.2388349515	1.08196721311475	7	5
Lpt-fit	21073.0885436893	1.12727272727273	3	4
Lpt-back	20213.0893203883	1.48333333333333	10	1

⁶Rangkorrelations nach Spearman: -0.92 wenn mit Experiment 6 von Artndt et al. verglichen

Konklusion

- Funktionale Sichtweise nützlich
- PBT nützlich, (fast) umsonst
- Zu jedem Ergebnis ist ein Experiment generierbar
- Algorithmen in beliebigen Situationen vergleichen problematisch
- Experimente aus der Realität ableiten
- Algorithmen können ohne konkreten Fall verglichen werden

- Funktionale Sichtweise nützlich
- PBT nützlich, (fast) umsonst
- Zu jedem Ergebnis ist ein Experiment generierbar
- Algorithmen in beliebigen Situationen vergleichen problematisch
- Experimente aus der Realität ableiten
- Algorithmen können ohne konkreten Fall verglichen werden

- Funktionale Sichtweise nützlich
- PBT nützlich, (fast) umsonst
- Zu jedem Ergebnis ist ein Experiment generierbar
- Algorithmen in beliebigen Situationen vergleichen problematisch
- Experimente aus der Realität ableiten
- Algorithmen können ohne konkreten Fall verglichen werden

- Funktionale Sichtweise nützlich
- PBT nützlich, (fast) umsonst
- Zu jedem Ergebnis ist ein Experiment generierbar
- Algorithmen in beliebigen Situationen vergleichen problematisch
- Experimente aus der Realität ableiten
- Algorithmen können ohne konkreten Fall verglichen werden

- Funktionale Sichtweise nützlich
- PBT nützlich, (fast) umsonst
- Zu jedem Ergebnis ist ein Experiment generierbar
- Algorithmen in beliebigen Situationen vergleichen problematisch
- Experimente aus der Realität ableiten
- Algorithmen können ohne konkreten Fall verglichen werden

- Funktionale Sichtweise nützlich
- PBT nützlich, (fast) umsonst
- Zu jedem Ergebnis ist ein Experiment generierbar
- Algorithmen in beliebigen Situationen vergleichen problematisch
- Experimente aus der Realität ableiten
- Algorithmen können ohne konkreten Fall verglichen werden