Gruppe

- Linus Keiser
- Simon Keiser

1

1a

Prozesse	P_1	P_2	P_3	P_4	P_5	P_6
Ankunftszeit	0	4000	5000	39000	42000	43000
Rechenzeit	15000	20000	5000	50000	25000	10000

Variablen:

`t` ist die Zeit, P der momentan ausgewählte Prozess, W die Menge der bereiten Prozesse, und ggf. deren verbleibender Rechenzeit, und P_{complete} die Menge der abgeschlossenen Prozesse.

Nicht-unterbrechendes Scheduling mit SJF-Verfahren

t	P	W	P_{complete}
0	P_1	$\{P_2, P_3\}$	$\emptyset$
15000	P_3	$\{P_2\}$	$\{P_1\}$
20000	P_2	$\{P_4\}$	$\{P_1, P_3\}$
40000	P_4	$\emptyset$	$\{P_1, P_2, P_3\}$
90000	P_6	$\{P_5\}$	$\{P_1, P_2, P_3, P_4\}$
100000	P_5	$\emptyset$	$\{P_1, P_2, P_3, P_4, P_6\}$
125000		$\emptyset$	$\{P_1, P_2, P_3, P_4, P_5, P_6\}$

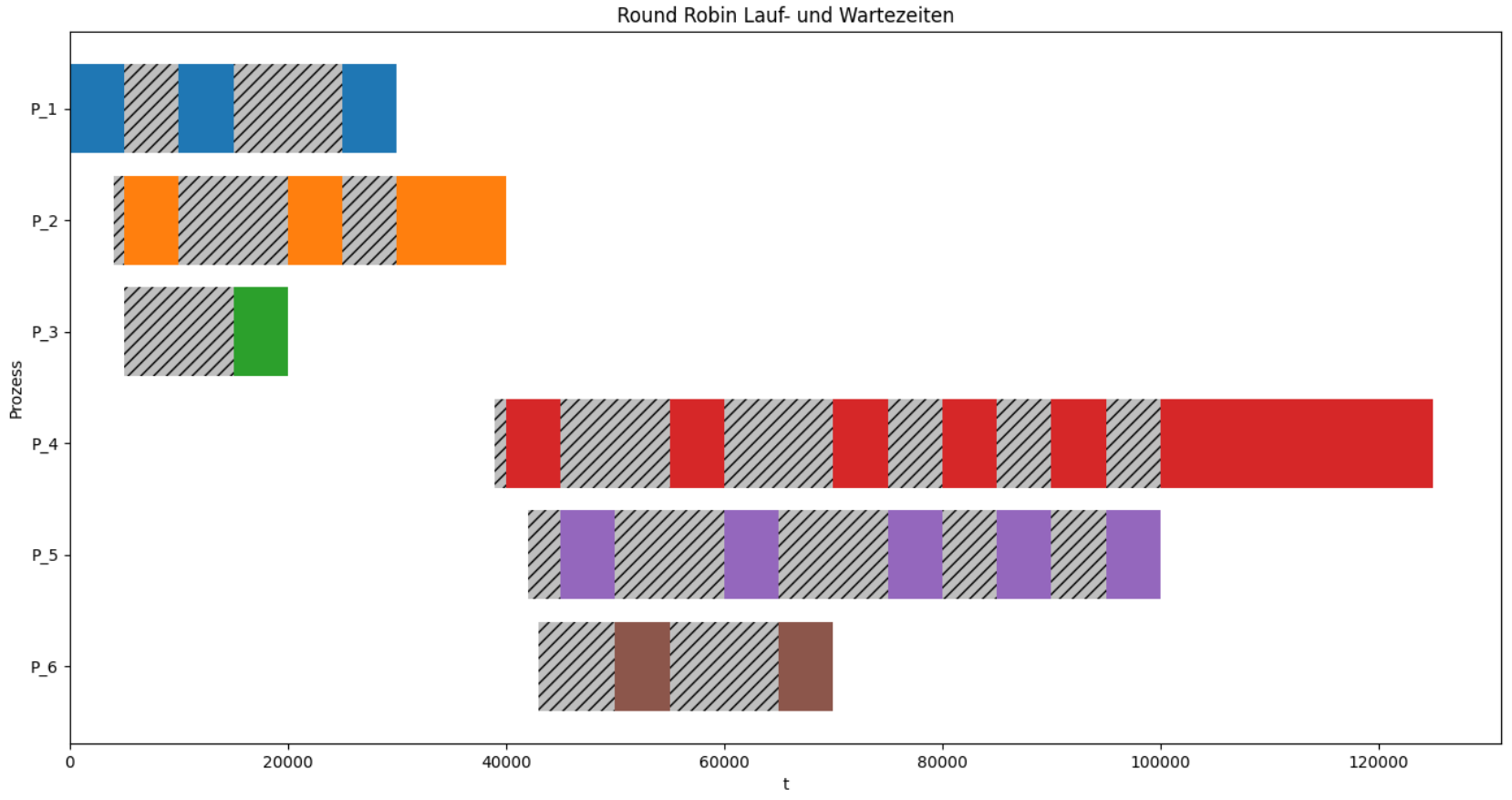
Resultat:

id	t_{arrival}	t_{start}	t_{end}	w
P_1	0	0	15000	0
P_2	4000	20000	40000	16000
P_3	5000	15000	20000	10000
P_4	39000	40000	90000	1000
P_5	42000	100000	115000	58000
P_6	43000	90000	100000	47000

$$\overline{w} = \frac{0 + 16000 + 10000 + 1000 + 58000 + 47000}{6} = 22000$$

Unterbrechendes Scheduling mit Round-Robin-Verfahren

Die Länge eines Zeitschlitzes beträgt in diesem Beispiel arbiträr 5000 Zeiteinheiten.



t	P	W	P_{complete}
0	P_1	$\emptyset$	$\emptyset$
5000	P_2	$\{(P_1, 10000)\}$	$\emptyset$
10000	P_1	$\{(P_3, 5000), (P_2, 15000)\}$	$\emptyset$
15000	P_3	$\{(P_2, 15000), (P_1, 5000)\}$	$\emptyset$
20000	P_2	$\{(P_1, 5000)\}$	$\{P_3\}$
25000	P_1	$\{(P_2, 10000)\}$	$\{P_3\}$
30000	P_2	$\emptyset$	$\{P_3, P_1\}$
35000	P_2	$\emptyset$	$\{P_3, P_1\}$
40000	P_4	$\emptyset$	$\{P_3, P_1, P_2\}$
45000	P_5	$\{(P_6, 10000), (P_4, 45000)\}$	$\{P_3, P_1, P_2\}$
50000	P_6	$\{(P_4, 45000), (P_5, 20000)\}$	$\{P_3, P_1, P_2\}$
55000	P_4	$\{(P_5, 20000), (P_6, 5000)\}$	$\{P_3, P_1, P_2\}$
60000	P_5	$\{(P_6, 5000), (P_4, 40000)\}$	$\{P_3, P_1, P_2\}$
65000	P_6	$\{(P_4, 40000), (P_5, 15000)\}$	$\{P_3, P_1, P_2\}$
70000	P_4	$\{(P_5, 15000)\}$	$\{P_3, P_1, P_2, P_6\}$
75000	P_5	$\{(P_4, 35000)\}$	$\{P_3, P_1, P_2, P_6\}$
80000	P_4	$\{(P_5, 10000)\}$	$\{P_3, P_1, P_2, P_6\}$
85000	P_5	$\{(P_4, 30000)\}$	$\{P_3, P_1, P_2, P_6\}$
90000	P_4	$\{(P_5, 5000)\}$	$\{P_3, P_1, P_2, P_6\}$
95000	P_5	$\{(P_4, 25000)\}$	$\{P_3, P_1, P_2, P_6\}$
100000	P_4	$\emptyset$	$\{P_3, P_1, P_2, P_6, P_5\}$
125000		$\emptyset$	$\{P_3, P_1, P_2, P_6, P_5, P_4\}$

Resultat:

A ist die Menge der Zeitspannen, in denen ein Prozess aktiv war.

id	t_{arrival}	t_{end}	w	A
P_3	5000	20000	10000	$\{(15000, 20000)\}$

id	t_{arrival}	t_{end}	w	A
P_1	0	30000	15000	$\{(0, 5000), (10000, 15000), (25000, 30000)\}$
P_2	4000	40000	16000	$\{(5000, 10000), (20000, 25000), (30000, 35000), (35000, 40000)\}$
P_6	43000	70000	17000	$\{(50000, 55000), (65000, 70000)\}$
P_5	42000	100000	33000	$\{(45000, 50000), (60000, 65000), (75000, 80000), (85000, 90000), (95000, 100000)\}$
P_4	39000	125000	36000	$\{(40000, 45000), (55000, 60000), (70000, 75000), (80000, 85000), (90000, 95000), (100000, 125000$

$$\overline{w} = \frac{15000 + 16000 + 10000 + 36000 + 33000 + 17000}{6} = 21166.666$$

1b

Das allgemeine Kriterium für die Auswahl des nächsten zu aktivierenden Prozesses, sowohl im nicht-unterbrechenden als auch im unterbrechenden Scheduling, ist die **kürzeste verbleibende Rechenzeit**. Dieses Kriterium lässt sich z.B. durch das Schedulingverfahren SJF für nicht-unterbrechendes Scheduling und SRT für unterbrechendes Scheduling erfüllen.

1. Nicht-unterbrechendes Scheduling: SJF
   - Wenn ein Prozess beendet wird und ein neuer zu aktivierenden Prozess ausgewählt wird, wählt der Scheduler den Prozess mit der kürzesten Gesamtausführungszeit aus der Menge der bereiten Prozesse
   - Wenn es mehrere Prozesse mit der gleichen Rechenzeit gibt, kann der Scheduler den Prozess auswählen, der zuerst eingetroffen ist (First Come First Serve - FCFS), oder andere Mechanismen zur Aufhebung von Gleichständen verwenden.
   - Indem der Scheduler den Prozess mit der kürzesten verbleibenden Rechenzeit als nächstes auswählt, wird sichergestellt, dass kürzere Prozesse schnell beendet werden, wodurch sich die Wartezeit für alle Prozesse in der Warteschlange verkürzt.
2. Unterbrechendes Scheduling: SRT
   - In diesem Fall sollte der Scheduler die verbleibende Rechenzeit der Prozesse berücksichtigen, nicht nur ihre Gesamtausführungszeit. Wenn ein neuer Prozess eintrifft oder der aktuelle Prozess unterbrochen wird, wählt der Scheduler den Prozess mit der kürzesten verbleibenden Rechenzeit aus der Menge der bereiten Prozesse aus.
   - Wenn es mehrere Prozesse mit der gleichen verbleibenden Rechenzeit gibt, kann der Scheduler dieselben Mechanismen zur Aufhebung des Gleichstands wie oben erwähnt verwenden.
   - Indem der Scheduler den Prozess mit der kürzesten verbleibenden Zeit zuerst auswählt, wird sichergestellt, dass Prozesse mit weniger verbleibender Zeit schnell beendet werden, wodurch die durchschnittliche Wartezeit für alle Prozesse in der Warteschlange verringert wird.

Durch die konsequente Auswahl des Prozesses mit der kürzesten verbleibenden Rechenzeit ermöglicht der Scheduler, dass kürzere Prozesse schneller abgeschlossen werden können, wodurch sich die Gesamtwartezeit für alle Prozesse in der Warteschlange verringert. Die Auswahl nach diesem Kriterium trägt dazu bei, die durchschnittliche Wartezeit für das System zu minimieren und gleichzeitig die konkurrierenden Anforderungen der verschiedenen Prozesse auszugleichen.

2

2a

Durch das mehrfache Aufführen eines Prozesses in der Liste wird er gewissermaßen bevorzugt. Sollte der Prozess nach Ablauf der Zeitscheibe noch nicht abgeschlossen sein, wird er am Ende der Prozessliste eingeordnet. Da er jedoch durch die mehrfache Auflistung bereits an einer früheren Stelle in der Liste steht wird er erneut Rechenzeit bekommen bevor es das round-robin-Verfahren eigentlich vorsähe. Dadurch können sich die Rechenzeiten verschiedener Prozesse in einem gegebenen Zeitintervall trotz ähnlicher Ankunfts- und Gesamtrechenzeit sehr stark unterscheiden: Ein Prozess der zu beginn dreimal in der Liste steht wird auf lange Sicht auch dreimal soviel Rechenzeit bekommen wie nur einmal aufgeführter Prozess. Das mehrfache Auflisten erhöht also die Frequenz in der ein Prozess Rechenzeit zugeteilt bekommt und bricht somit die Fairness des round-robin-Verfahrens. Dieser Effekt wird durch im Verhältnis zu den Zeitscheiben lange Rechenzeiten verstärkt.

2b

Genau aus den zuvor beschriebenen Gründen kann eine Mehrfachauflistung “besonders wichtiger” Prozesse oder solcher mit einer sehr hohen Rechenzeit sinnvoll sein, um eine Priorisierung bei der Zuteilung der Prozessorzeit zu erwirken.

2c

Wird ein Prozess beim round-robin-Verfahren blockiert wird sofort der nächste Prozess ausgewählt - d.h. ein Prozesswechsel durchgeführt. Durch eine mehrfache Listung eines blockierten Prozesses werden somit mehrere “überflüssige” Prozesswechsel durchgeführt. Dies wirkt sich negativ auf die Performance aus, da es zu hohen Prozesswechselkosten führt.