

Universität Karlsruhe (TH)

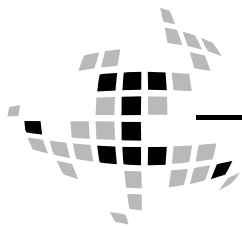
Forschungsuniversität · gegründet 1825

Ablaufplanung (Scheduling)

Prof. Dr. Walter F. Tichy

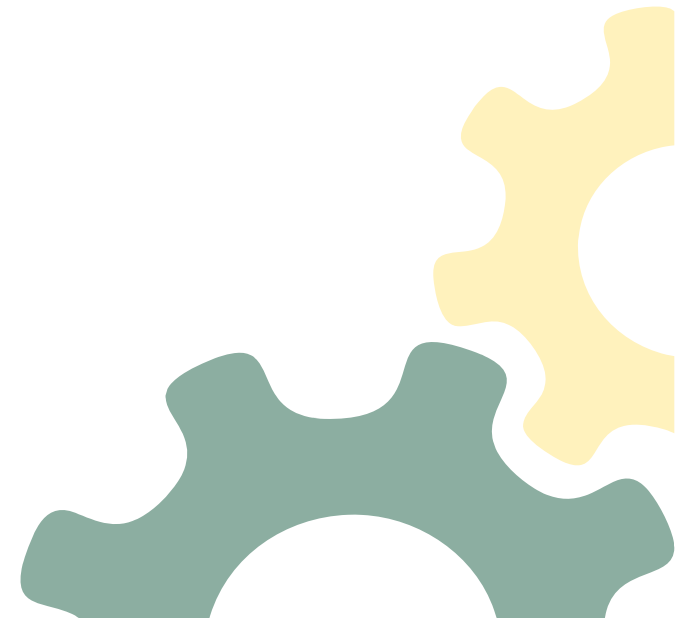
Dr. Victor Pankratius

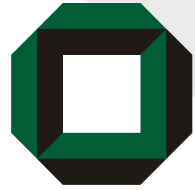
Ali Jannesari



Fakultät für **Informatik**

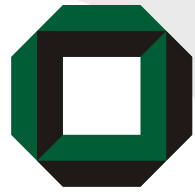
Lehrstuhl für Programmiersysteme





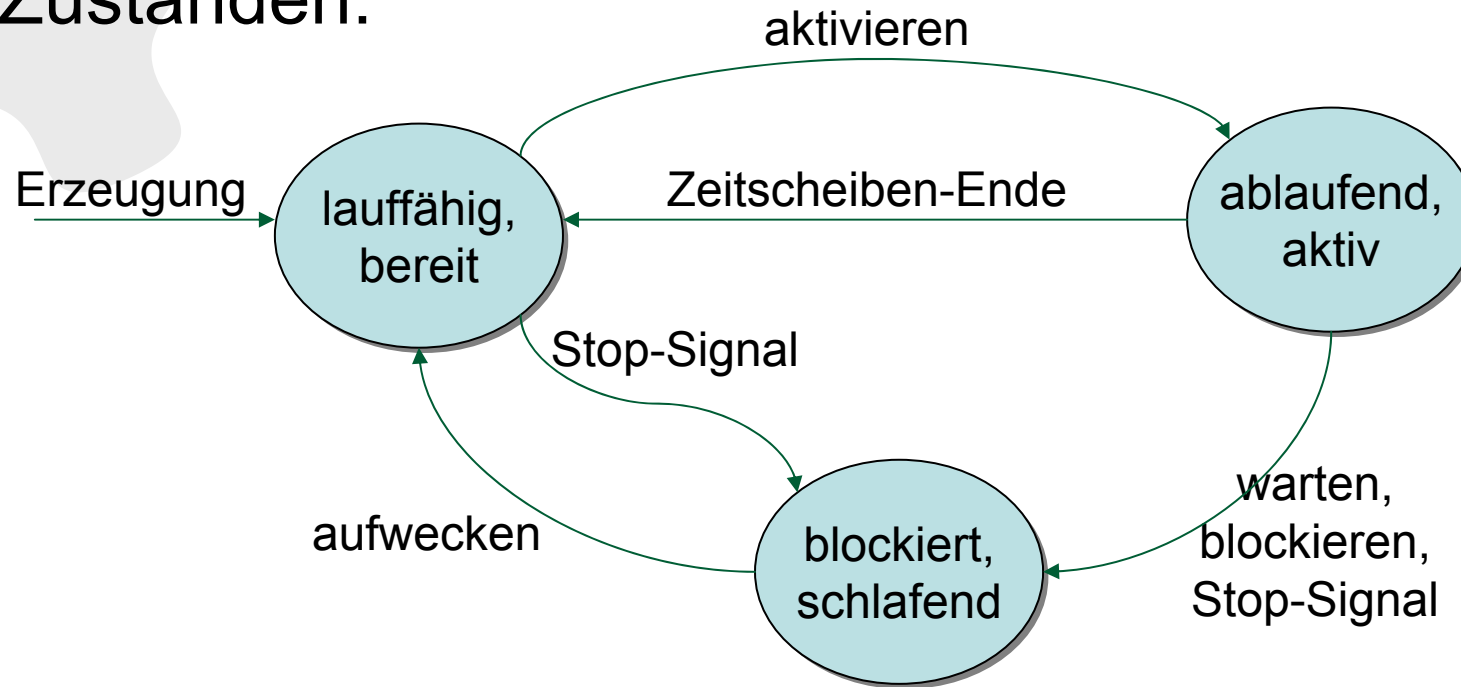
Vorlesung „Cluster Computing“

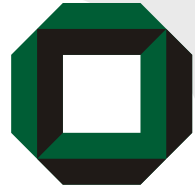
- Architektur von Rechnerbündeln
- Betrieb von Rechnerbündeln
 - Single System Image
 - Ablaufplanung (Scheduling)
- Nutzung und Programmierung von Rechnerbündeln



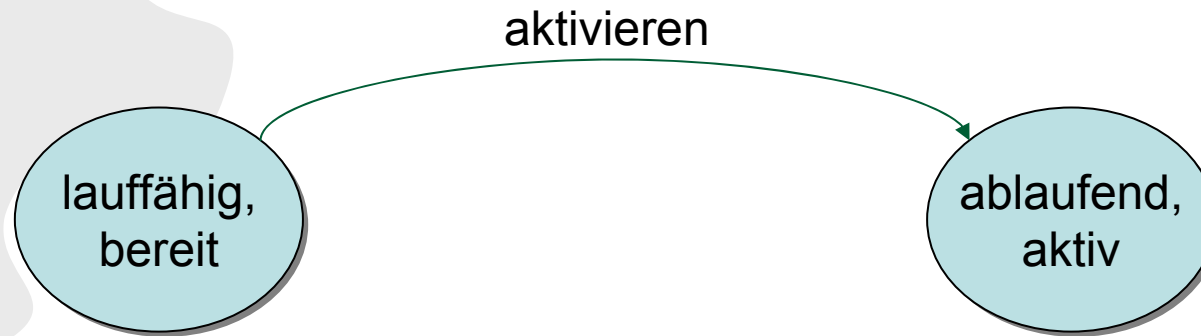
Erinnerung

- Ein Prozess ist stets in einem von drei Zuständen:



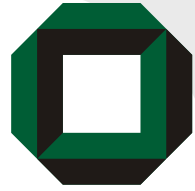


Traditionelle Ablaufplanung (1)

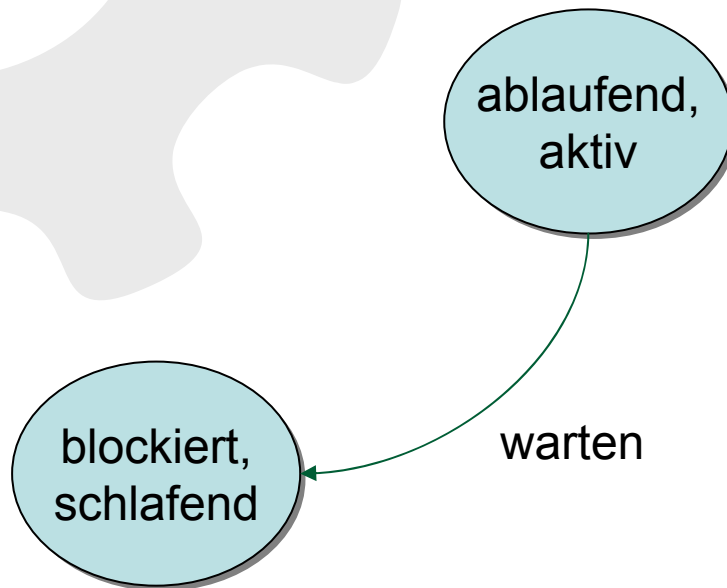


Traditionelle Ablaufplaner aktivieren einen beliebigen Prozess unter folgenden Annahmen:

- Prozesse sind unabhängig
 - Kommunikation ist selten
- } gilt auf Rechnerbündeln nicht mehr



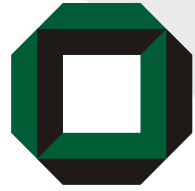
Traditionelle Ablaufplanung (2)



1+2 führen in der Regel

- zum Blockieren des wartenden Prozesses und
- und zu einem Kontextwechsel.

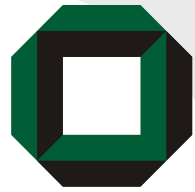
- Warten ist bei kommunizierenden Prozessen unvermeidbar.
- Drei typische Wartesituationen:
 1. Empfänger wartet auf das Eintreffen von Daten vom Sender.
 2. Sender wartet auf Abnahme der Daten (bei ungepuffertem Senden oder wenn der Sendepuffer voll ist).
 3. Die Daten warten in einem Puffer auf die Abnahme durch den Empfänger. Betriebssysteme optimieren diesen Standardfall wegen Platten-E/A: das langsamere Gerät muss niemals auf das schnellere warten.



Prozessgruppe

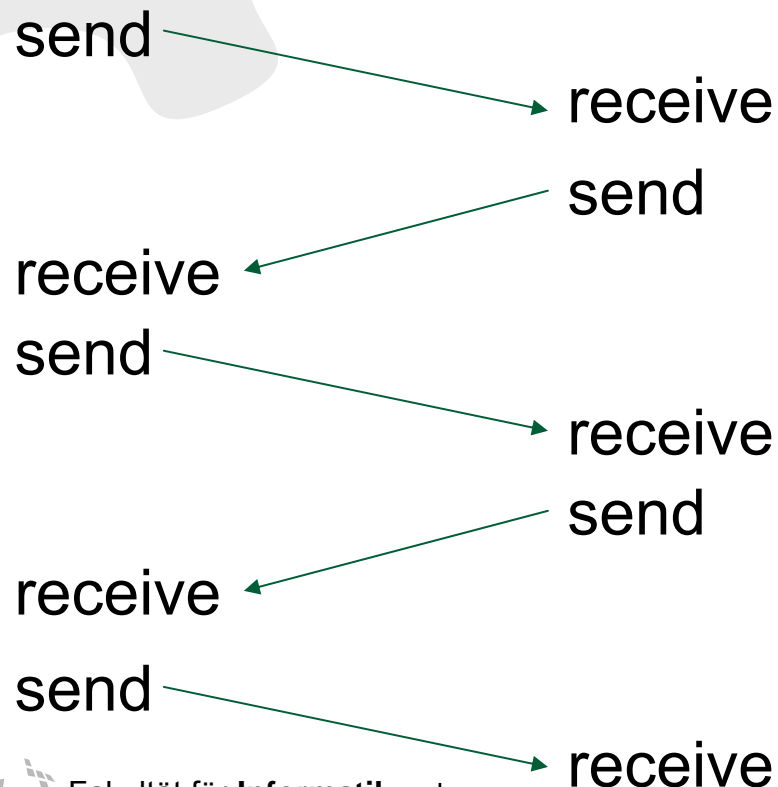
Auf Rechnerbündeln ist die Situation anders:

- Prozesse sind nicht unabhängig und kommunizieren öfter.
 - **Prozessgruppe** = alle zu **einer** Anwendung gehörenden Prozesse
- Mehrere Prozessgruppen laufen auf Bündel quasi-simultan.
- Auftreten und Größe von Prozessgruppen nur dynamisch bekannt.
- Es gibt mehrere Prozessoren.



Prozessflattern

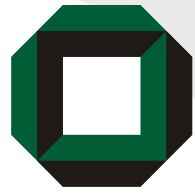
Betrachte zwei Prozesse, die sich abwechselnd per asynchronem "send" Botschaften zusenden und diese per "receive" empfangen.



Beide Prozesse bilden eine Prozessgruppe.

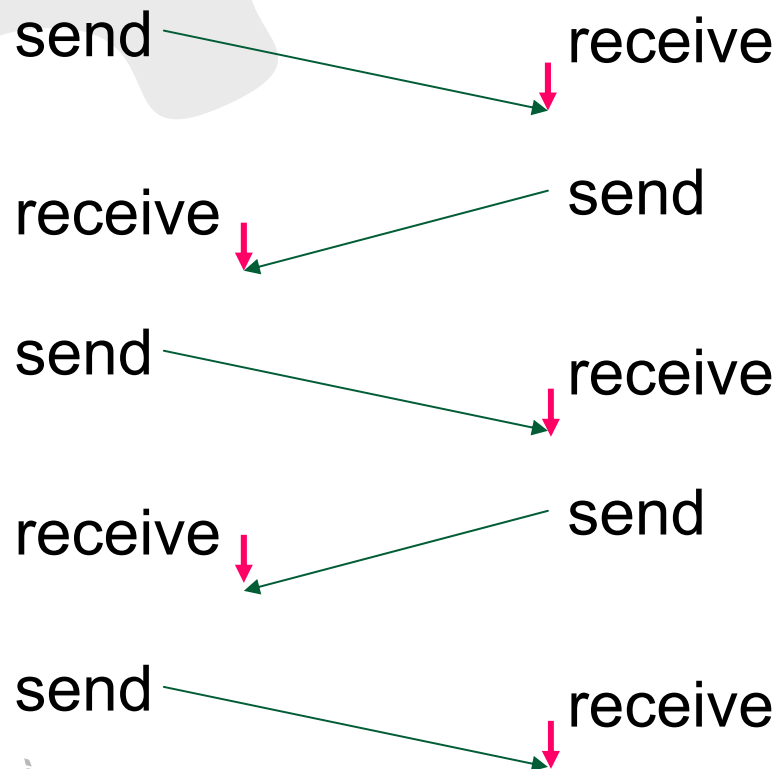
Bei idealer Ablaufplanung wird nie ein Prozess blockiert; es gibt keine Kontextwechsel.

Gesamtlaufzeit: fünf Botschaften

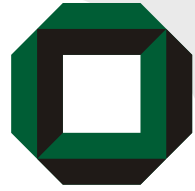


Prozessflattern

Betrachte zwei Prozesse, die sich abwechselnd per asynchronem "send" Botschaften zusenden und diese per "receive" empfangen.



- Wenn receive Aufruf immer zu früh, erfolgt jedes Mal Blockierung, d.h. Wechsel zu drittem Prozess. Anschließend Rückwechsel.
Kurz: **Prozessflattern**.
- Ein Prozess der Prozessgruppe läuft, der andere blockiert. Gruppe ist **fragmentiert**
- Kommunikation nur bei Kontextwechsel.
- **Längere Gesamtlaufzeit**: 10 Kontextwechsel (raus/rein) plus mind. 5 Zeitscheiben (in anderen Prozessen)

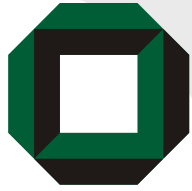


Minimalverbesserung

Falls Prozess warten muss: nicht sofort blockieren, sondern bis zu zwei Kontextwechselzeiten dynamisch abwarten.

Motivation:

- Blockieren und späteres Fortsetzen kostet mindestens zwei Kontextwechselzeiten. Aktives Warten günstiger, wenn viel und feingranular kommuniziert wird.
- Wenn nach aktiver Wartezeit doch noch blockiert wird, hat man die aktive Wartezeit verloren. Während der Pause hätte ggf. ein anderer Prozess sinnvoll weiterarbeiten können.

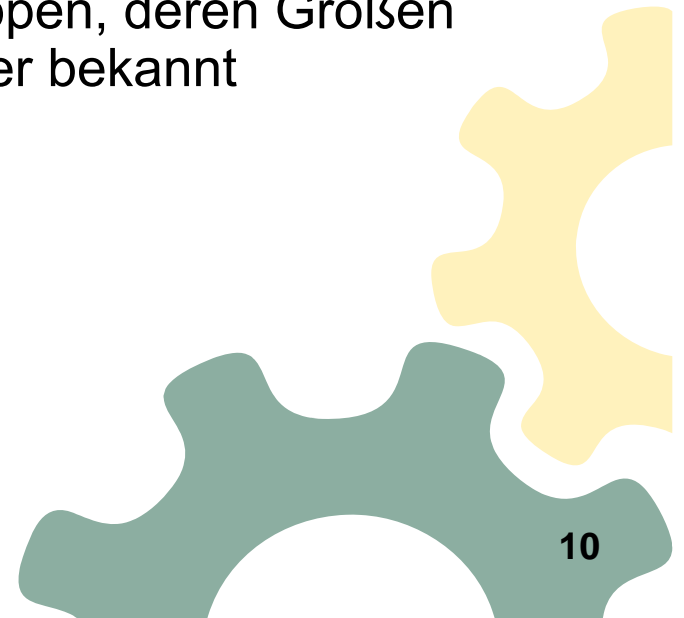


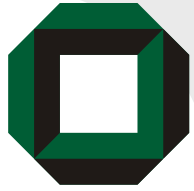
Ablaufplanung auf Rechnerbündeln

- Ziel: Bei gleichzeitigem Vorhandensein mehrerer Prozessgruppen ist eine (zeitliche) Zuordnung der Prozesse zu den Prozessoren gesucht, die minimale Ausführungszeit benötigt.

Ansätze:

- Statische Ablaufplanung
 - Zeitpunkt des Auftretens aller Prozessgruppen, deren Größen und deren Kommunikationsverhalten vorher bekannt
- **Dynamische Ablaufplanung**
 - Zeitpunkt nicht vorher bekannt
 - Gesucht: koordinierte Ablaufplanung
 - Explizite Koordination
 - Implizite Koordination





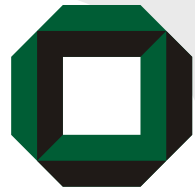
Koordinierte Ablaufplanung Gang-Scheduling, Co-Scheduling

Idee:

Wartezeiten beim Austausch von Botschaften innerhalb einer Anwendung können reduziert werden, indem eine Prozessgruppe möglichst vollständig simultan (auf allen zugeteilten Prozessoren) abläuft.

Koordinierte Ablaufplanung =

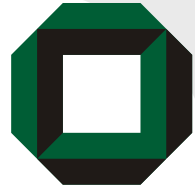
Alle lauffähigen Prozesse einer Anwendung oder Prozessgruppe laufen gleichzeitig ab.



Definitionen

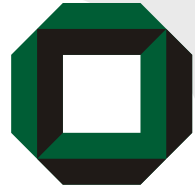
Explizit koordinierte Ablaufplanung =
eine globale Instanz koordiniert die Ablaufplanung

Implizit koordinierte Ablaufplanung =
Es gibt keine globale Instanz zur Ablaufplanung.
Die Ablaufplaner der Einzelknoten bestimmen aus dem
Botschaftenverkehr, welche Prozesse jeweils aktiv
gesetzt werden sollten, um (heuristisch) eine koordinierte
Ablaufplanung zu erreichen.
(unvollständige Information, dafür aber kein globaler
Informationsaustausch)



Explizit koordinierte Ablaufplanung

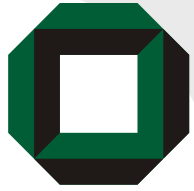
- Lösungsansätze:
 - Ousterhout: „Scheduling Techniques for Concurrent Systems“, Proc. 3rd Int. Conf. on Distrib. Computing Systems, Oct. 1982
 1. Matrixkoordination
 2. Kontinuierliche Koordination
 3. Unzerteilte Koordination
- Nebenbedingungen:
 - Jede Prozessgruppe hat nicht mehr Prozesse als Prozessoren.
 - Jede Prozessgruppe wird beim Start bestimmt und ändert sich während des Laufes nicht (statisch).
 - Prozesse können nicht migrieren. Sie werden beim Start einer Anwendung einem Prozessor fest zugewiesen.
 - Prozessoren sind identisch (→ Lastbalance einfach).



Matrixkoordination

- Es gibt P Prozessoren; jeder Prozessor verwaltet bis zu Q Prozesse.
- Sicht der globalen Ablaufplanung:
 $Q \times P$ Matrix mit $P \cdot Q$ Prozesskacheln
 - Spalten $p_1 \dots p_P$: Prozesse eines Prozessors
 - Zeilen $q_1 \dots q_Q$: enthalten Prozessgruppe(n)

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	...	p_P
q_1									
q_2									
q_3									
q_4									
q_5									
⋮									
q_Q									

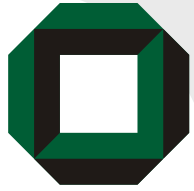


Matrixkoordination (1)

Prozessgruppenallokation

- Allokation sehr einfach: Finde Zeile in Matrix mit genügend freien Kacheln, um die Prozesse der Gruppe unterzubringen.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
q_1	gray	gray	gray	gray	gray	yellow	yellow	yellow
q_2	orange	orange	orange	orange	orange	orange	dark green	dark green
q_3	purple	purple	red	red	red	red	white	brown
q_4	light blue	light blue	light blue	light blue	light blue	light blue	light blue	white
q_5	white	white	white	white	white	white	white	white
q_6	white	white	white	white	white	white	white	white

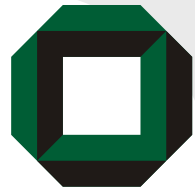


Matrixkoordination (2) globale Ablaufplanung

- In der Zeitscheibe $i \bmod Q$
 - haben Prozesse in Zeile i die höchste Priorität;
 - werden alle Prozessgruppen in Zeile i ausgeführt.
- Zyklisches Fortschalten von i .
- Globale Ablaufplanung ist einfach: Zeilennummer wählen. Das kann mit einer globalen Uhr sogar verteilt gemacht werden.

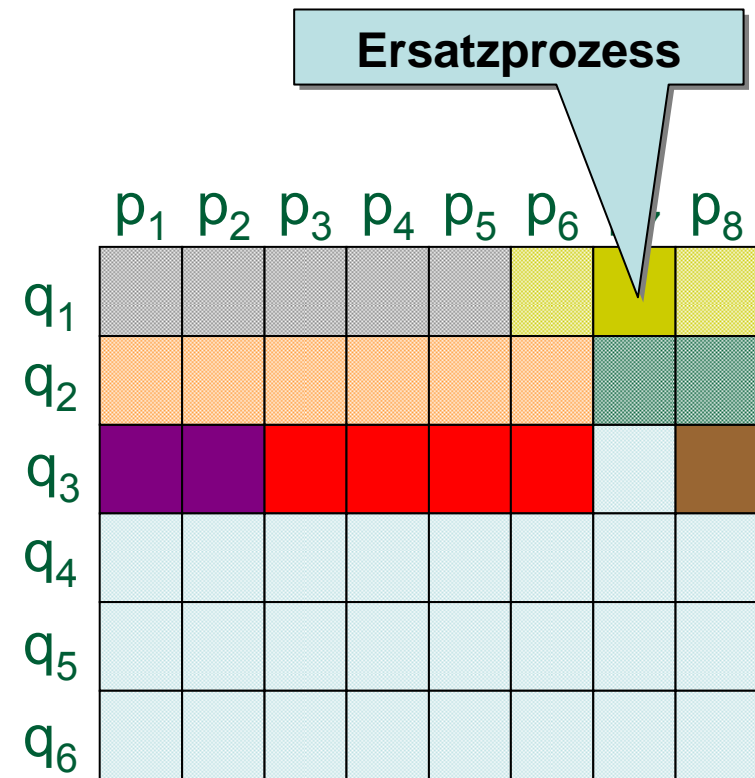
	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
q_1								
q_2								
q_3								
q_4								
q_5								
q_6								

Noch ist Zeile 2 aktiv.
Zeile 3 kommt als nächste.

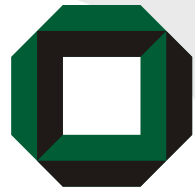


Matrixkoordination (3) lokale Ablaufplanung

- Lokale Ablaufplanung wird aktiv
 - wenn einer der Prozesse in der aktiven Zeile blockiert ist (z.B. auf E/A wartet)
 - oder wenn Prozessor "arbeitslos" wäre weil kein Prozess eingetragen ist.und wählt Ersatzprozess aus Spalte ohne Abstimmung mit anderen Knoten.
- Lokale Ablaufplanung ist einfach.
- In der Regel bilden Ersatzprozesse keine komplette Prozessgruppe sondern nur Fragmente.
- Blockieren alle Prozesse einer Zeile, wird vor Ende der Zeitscheibe keine neue Prozessgruppenwechselentscheidung gefällt. (d.h. Ersatzprozesse laufen bis Ende der Zeitscheibe.)

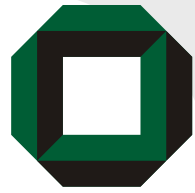


Jetzt ist Zeile 3 aktiv.



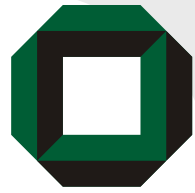
Matrixkoordination (4) Implementierung

- Der globale Ablaufplaner speichert die ganze Matrix.
- Jeder lokale Ablaufplaner speichert nur seine jeweilige Spalte.
- Bei Allokation und Deallokation von Prozessgruppen werden die betroffenen Prozessoren mit Änderungsinformationen für ihre Spalten versorgt.
- Der Prozessgruppenwechsel muss koordiniert werden:
 - entweder über eine Kommunikation vom globalen Ablaufplaner an alle lokalen Ablaufplaner und einer Barrierensynchronisation
 - oder durch synchronisierte Uhren; dann sind alle Entscheidungen lokal, abgesehen von gelegentlicher Uhren-Synchronisation (einmal pro Stunde?)



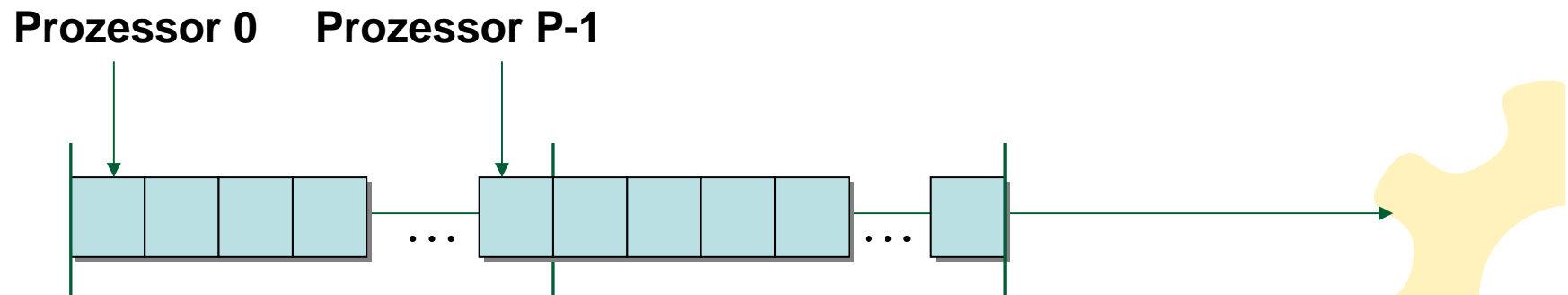
Matrixkoordination (5) Bewertung

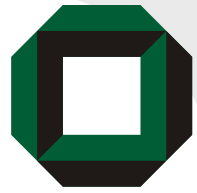
- + einfache globale und lokale Ablaufplanung
- + einfache Implementierung
- + wenig globale Kommunikation zur Koordination
- schlechte Nutzung der Kacheln (interne Fragmentierung); bei Prozessgruppenallokation bleiben "Löcher"
- keine gute Strategie für Ersatzprozesse; Möglichkeiten für Koordination werden nicht genutzt
- alle Prozesswechsel erfolgen gleichzeitig; daher ggf. hohe Anforderungen an gemeinsame Ressourcen (z.B. Cache-Invalidierung auf allen Knoten gleichzeitig)



Kontinuierliche Koordination

Matrix wird als zeilenweise ausgerollte Folge von Prozess-Kacheln betrachtet:



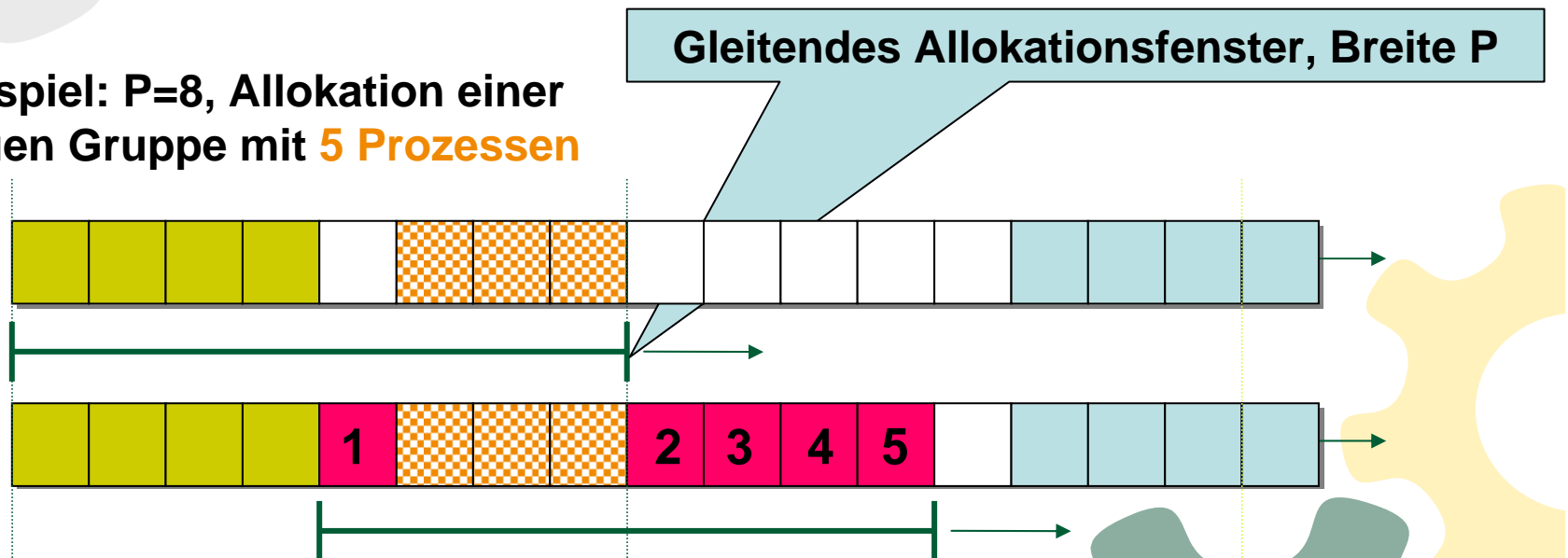


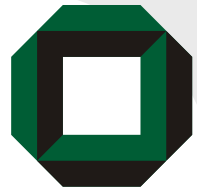
Kontinuierliche Koordination (1)

Progressgruppenallokation

- Suche nach der ersten Folge von P Kacheln, bei denen genügend viele frei sind, um die Gruppe aufzunehmen.

Beispiel: $P=8$, Allokation einer neuen Gruppe mit **5 Prozessen**

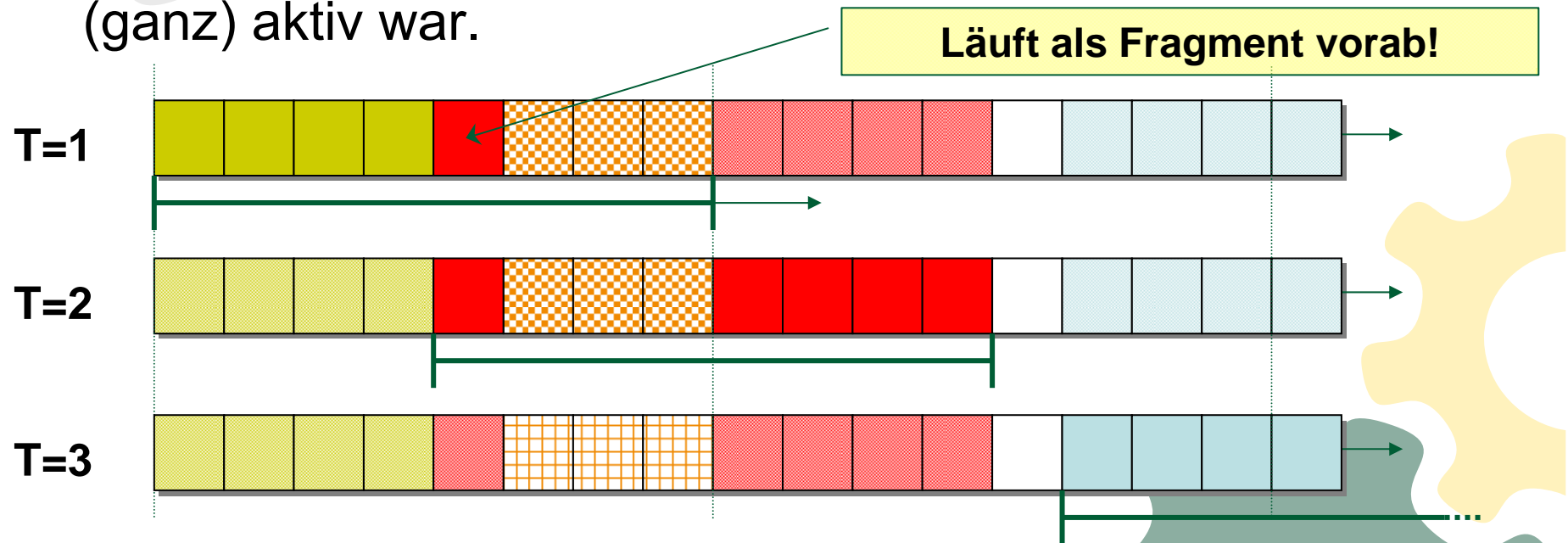


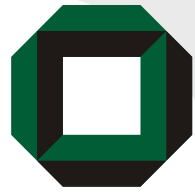


Kontinuierliche Koordination (2a)

Globale Ablaufplanung

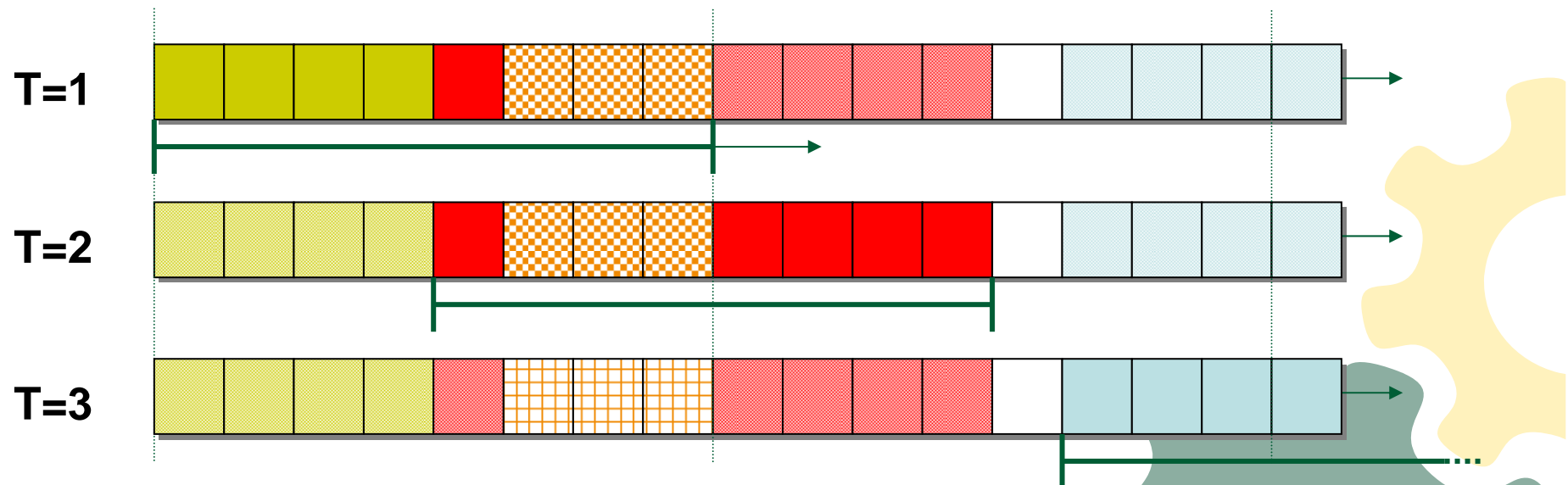
- Aktiv sind Prozesse, die im Ablaufplanungsfenster (Größe P) sind.
- Gruppenwechsel: Schiebe das Ablaufplanungsfenster so weit nach rechts, dass die linke Kachel im Fenster zur ersten Gruppe gehört, die im letzten Zyklus noch nicht (ganz) aktiv war.

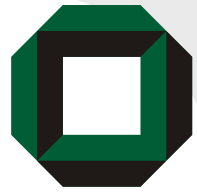




Kontinuierliche Koordination (2b) Globale Ablaufplanung

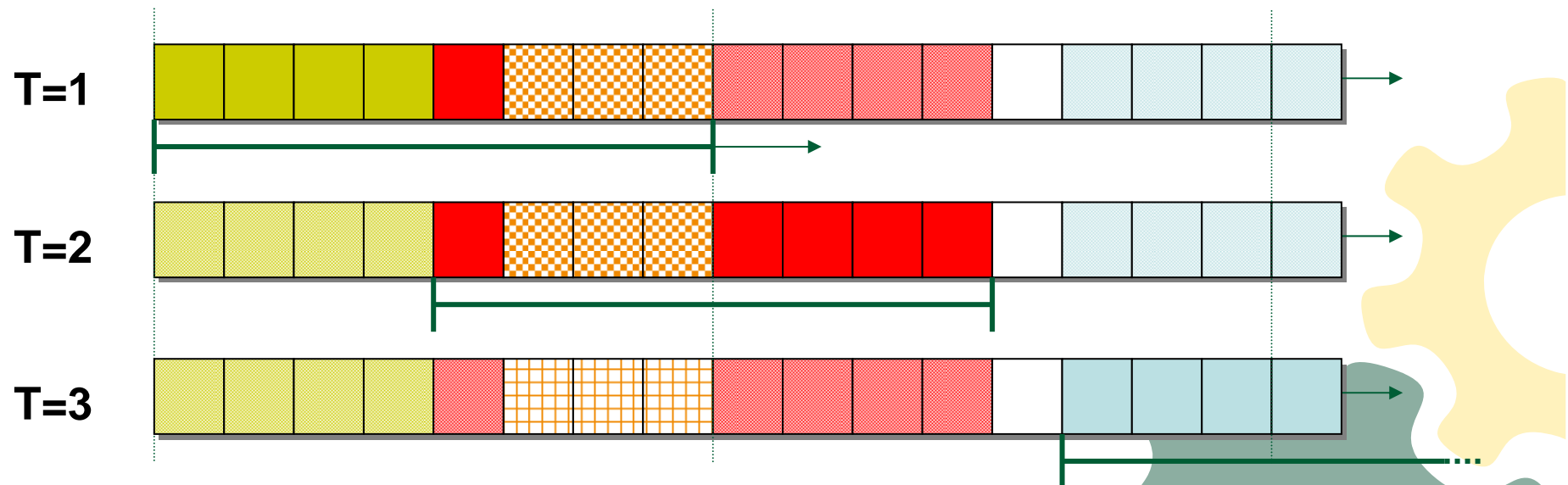
- Um Ungleichbehandlung zu vermeiden, muss immer zum ersten Prozess einer noch nicht komplett aktiven Gruppe verschoben werden.
- Nach $T=2$ wird das Ablaufplanungsfenster nicht um 1 verschoben, weil sonst die karierten Prozesse nochmals an der Reihe wären.

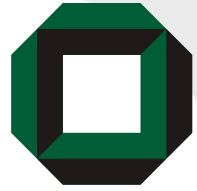




Kontinuierliche Koordination (2c) Globale Ablaufplanung

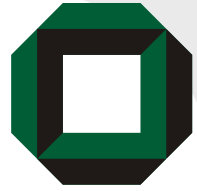
- Wegen Fragmentierung kann es aber (bei fragmentiert allozierten größeren Prozessgruppen) zu einer Bevorzugung kleiner Prozessgruppen kommen.
- Die karierten Prozesse sind in T=1 und T=2 an der Reihe.





Kontinuierliche Koordination (3) Lokale Ablaufplanung

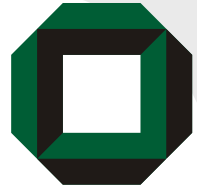
- Wie bei Matrixkoordination:
- Lokale Ablaufplanung wird aktiv
 - wenn einer der Prozesse in der aktiven Zeile blockiert ist (z.B. auf E/A wartet)
 - oder wenn Prozessor „arbeitslos“ wäre weil kein Prozess eingetragen ist
- und wählt Ersatzprozess aus Spalte (zurück zur Matrix-Ansicht mit Hilfe der vertikalen Linie in obigen Darstellungen) ohne Abstimmung mit anderen Knoten.



Kontinuierliche Koordination (4) Implementierung

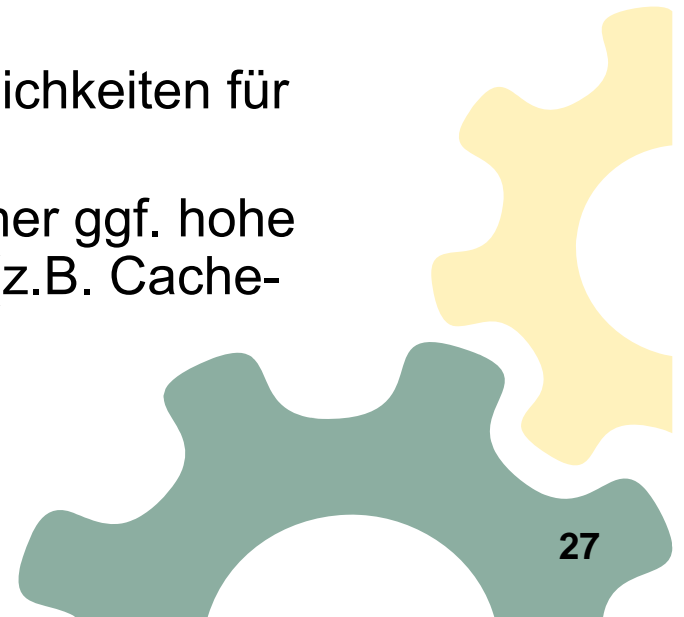
Wie bei Matrixkoordination:

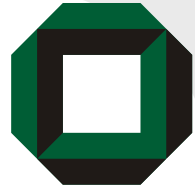
- Der globale Ablaufplaner speichert die ganze (ausgerollte) Matrix.
- Jeder lokale Ablaufplaner speichert nur seine jeweilige Spalte (zurück zur Matrix-Sicht).
- Bei Allokation und Deallokation von Prozessgruppen werden die betroffenen Prozessoren mit Änderungsinformationen für ihre Spalten versorgt.
- Der Prozessgruppenwechsel muss über eine Kommunikation vom globalen Ablaufplaner an alle lokalen Ablaufplaner und eine Barriersynchronisation koordiniert werden.



Kontinuierliche Koordination (5) Bewertung

- + einfache globale und lokale Ablaufplanung
- + einfache Implementierung
- + wenig globale Kommunikation zur Koordination
- + bessere Nutzung der Kacheln (weniger interne Fragmentierung); bei Prozessgruppenallokation bleiben keine „Löcher“ mehr
- unfaire Bevorzugung kleiner Prozessgruppen bei fragmentiert allokierten größeren Prozessgruppen
- keine gute Strategie für Ersatzprozesse; Möglichkeiten für Koordination werden nicht genutzt
- alle Prozesswechsel erfolgen gleichzeitig; daher ggf. hohe Anforderungen an gemeinsame Ressourcen (z.B. Cache-Invalidierung auf allen Knoten gleichzeitig)

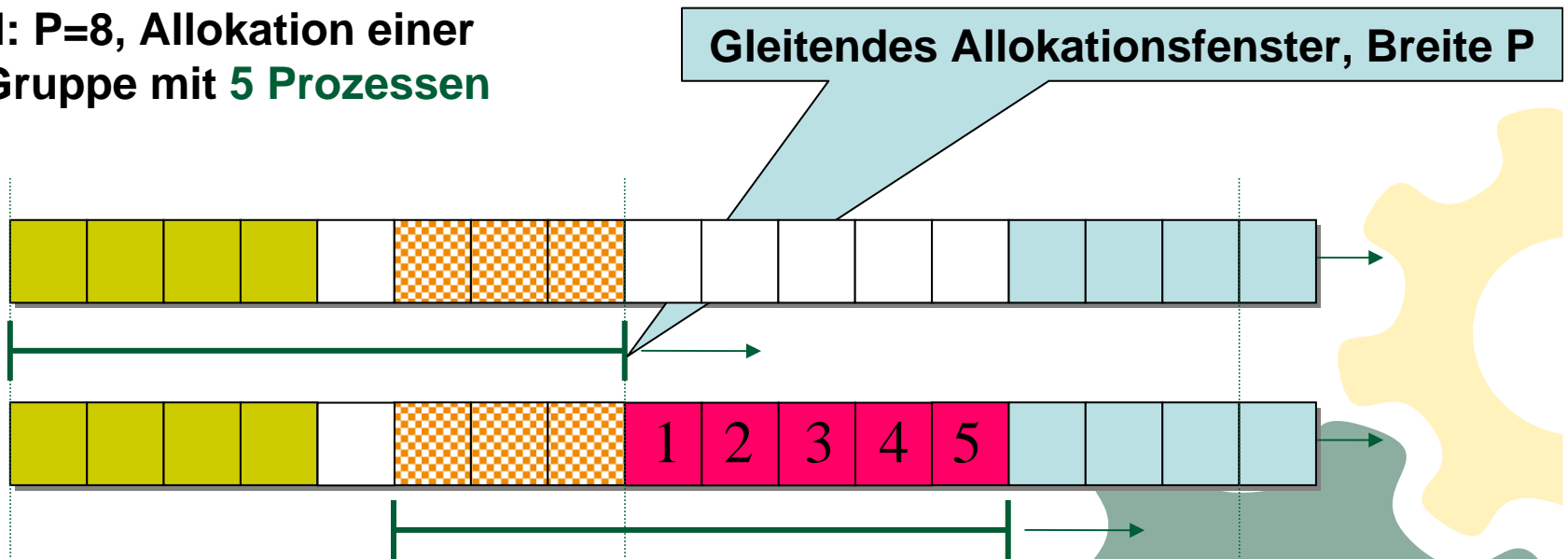


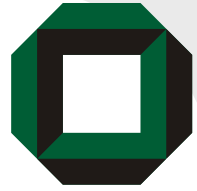


Unzerteilte Koordination

Analog der kontinuierlichen Koordination, außer: Prozessgruppen werden immer in einer ununterbrochenen Folge von Kacheln allokiert.

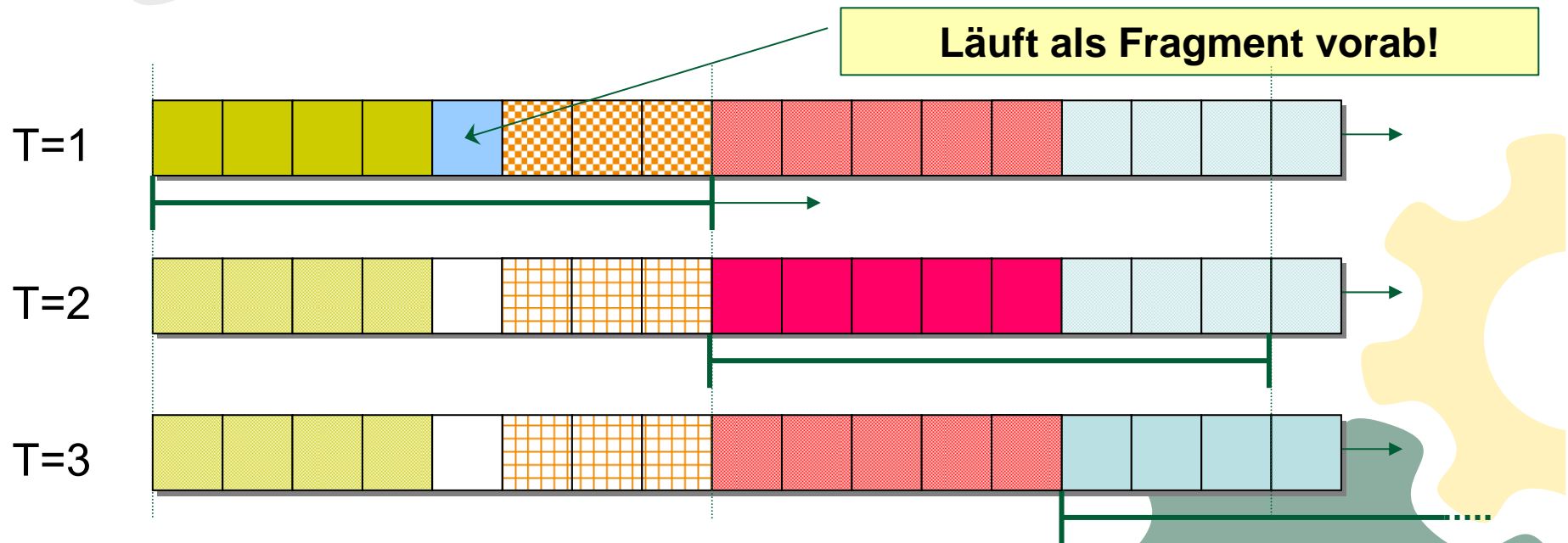
Beispiel: $P=8$, Allokation einer neuen Gruppe mit 5 Prozessen

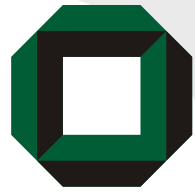




Unzerteilte Koordination

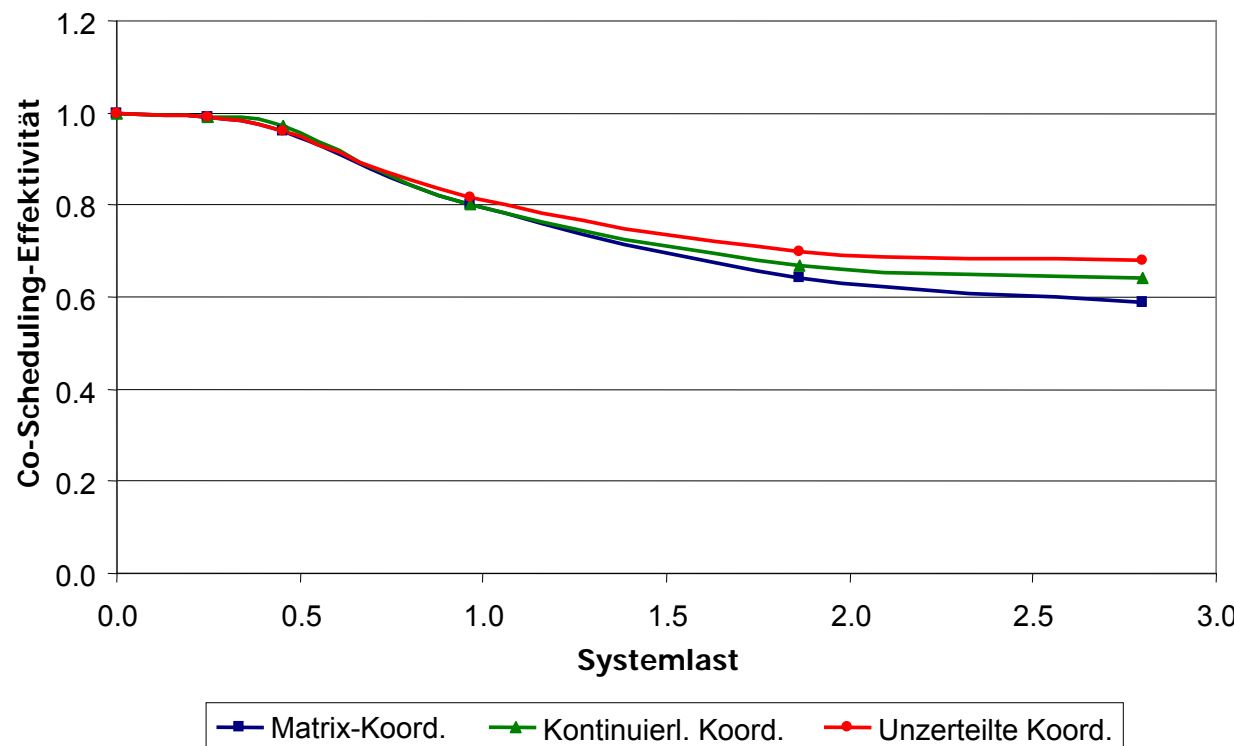
Kleinere Prozessgruppen werden nicht mehr unfair bevorzugt.





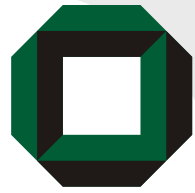
Vergleich

Effektivität



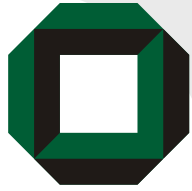
- Effektivität =
(Zahl der Prozessoren, die koord. Prozesse ausführen) / (Zahl Prozessoren mit lauffähigen Prozessen)
- Systemlast =
(Zahl lauffähiger Prozesse) /
Prozessorzahl

- 50-Prozessor-System
- Mittlere Gruppengröße:
13,5



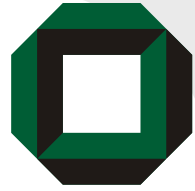
Implementierung auf Rechnerbündeln

- Lokale Zeitscheiben auf den einzelnen Knoten müssen global synchronisiert werden, z.B. durch:
 - Kopplung an globalen Zeitgeber
 - Kopplung an lokal abgegliche Zeitgeber (NTP)
 - Rundruf von Befehlen zum Zeitscheibenwechsel
- Synchronisierung gelingt ohne Echtzeitkern nur bedingt wegen möglicher Unterbrechungen.
- Performanzverlust durch Prozessflattern steht zeitl. Aufwand zur Synchronisierung gegenüber.



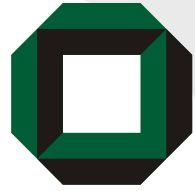
Ablaufplanung auf Rechnerbündeln

- Ziel: Bei gleichzeitigem Vorhandensein mehrerer Prozessgruppen ist eine (zeitliche) Zuordnung der Prozesse zu den Prozessoren gesucht, die minimale Ausführungszeit benötigt.
- Ansätze:
 - Statische Ablaufplanung
 - Zeitpunkt des Auftretens aller Prozessgruppen, deren Größen und deren Kommunikationsverhalten vorher bekannt
 - Dynamische Ablaufplanung
 - Zeitpunkt nicht vorher bekannt
 - Gesucht: koordinierte Ablaufplanung
 - Explizite Koordination
 - Implizite Koordination



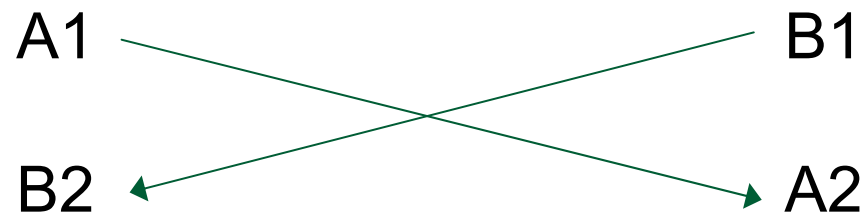
Implizit koordinierte Ablaufplanung

- Keine **globale** Steuerung
- Jeder Prozessor beobachtet die Kommunikation lokal und leitet daraus ab, welche Prozesse gemeinsam ablaufen sollten.
- **Naive Idee:** erhält ein Prozessor eine Nachricht, dann gehört er offensichtlich zu einer Prozessgruppe, die gerade von anderen Prozessoren zum Ablaufen gebracht ist. Daher Priorität erhöhen.

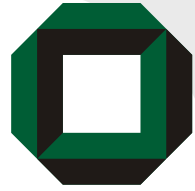


Nachteile des naiven Ansatzes

- Benutzer könnten durch Botschaften erreichen, dass die eigenen Prozesse bevorzugt ablaufen.
- Mögliche Konflikte bei **mehreren Prozessgruppen**:

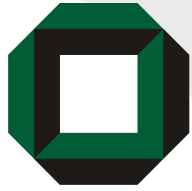


Wegen Prozesswechsel auf beiden Knoten laufen Prozessgruppen doch nicht koordiniert ab.

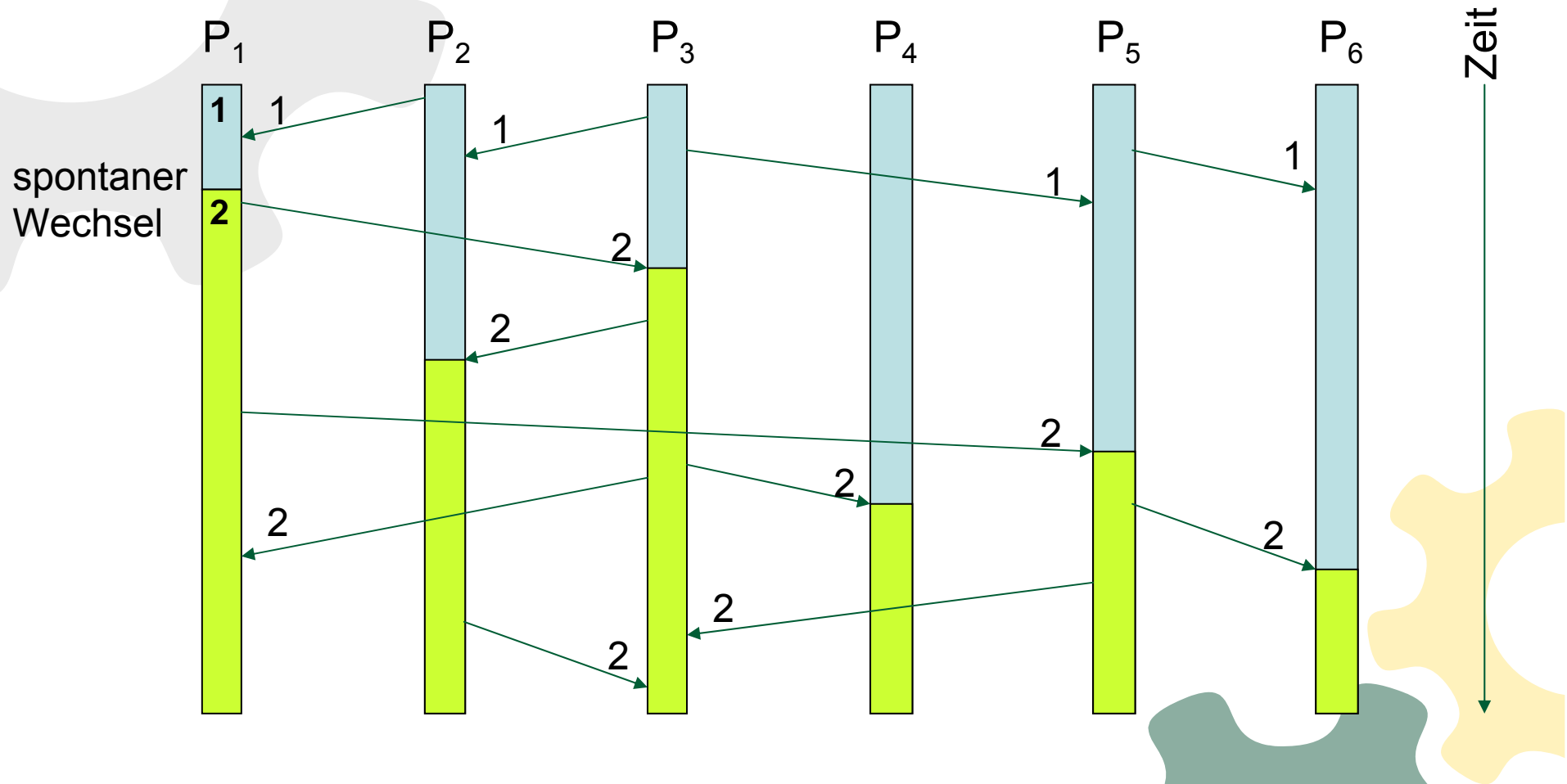


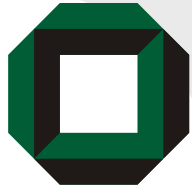
Epochen-Nummerierung

- Jeder Knoten führt eine Epochen-Nummer, die ...
 - ... bei einem spontanen Prozesswechsel erhöht wird (z.B. am Ende der Zeitscheibe), jedoch nicht bei Prozesswechsel wegen kommender Botschaft;
 - ... mit Botschaft verschickt wird.
- Bei Botschaftsempfang: Vergleich der lokalen Epoche mit Botschaftsepoche.
 - Wechsel zum Zielprozess nur, wenn die Botschaft aus einer späteren Epoche stammt. Dann wird lokale Epoche auf Botschaftsepoche gesetzt.
 - Sonst kein Prozesswechsel. Kein Rückschalten in frühere Epoche.
- Neu ausgewählte Prozessgruppen verbreiten ihren Einfluss auf alle Knoten ... bis zum Ende der Zeitscheibe.
- Umschalten erfolgt im Rechnerbündel nach und nach.



Epochen-Nummerierung





Epochen-Nummerierung

