3. Techniken der Leistungsanalyse

Modelle zur Leistungsanalyse sind Warteschlangennetze, diese bestehen zum einen aus Ressourcen

- Ressourcen umfassen einen oder mehrere identische Bediener und einen Warteraum
- Ressourcen werden auch als Stationen bezeichnet
- unterschiedliche Typen von Stationen existieren

und im System zirkulierenden Aufträge,

- die Bedienung an Stationen verlangen (und dabei evtl. auf die Verfügbarkeit der des/eines Bedieners warten müssen),
- deren Bedienwünsche i.d.R. als Zufallsvariablen spezifiziert werden, von denen meist nur der Erwartungswert zur Analyse verwendet wird,
- deren Besuchsanzahl an einer Station ebenfalls durch eine Zufallsvariable beschrieben wird, von der i.d.R. nur der Erwartungswert betrachtet wird,
- die auch als Kunden, Jobs etc. bezeichnet werden,
- die von unterschiedlichem Typ sein können.

Bisherige Betrachtung von Warteschlangennetzen unter operationalen Regeln

- Annahmen über das "typisches" Systemverhalten in endlichen Intervallen
- Herleitung von Leistungsgrößen auf Basis einfacher Gesetze
- Resultate gelten für endliche Intervalle und können z.T. auf unendliche Intervalle übertragen werden

Vorteil der operationalen Sichtweise:

Einfache Herleitung und Beweise für viele Resultate

Andere Sichtweise: stochastische Analyse

- Ankunfts- und Bedienzeiten unterliegen stochastischen Einflüssen und sind beschrieben durch Zufallsvariablen (ZVs) mit gegebener Verteilungsfunktion
- Wege durch ein Netz sind durch Verzweigungswahrscheinlichkeiten definiert
- Stochastische Analyse untersucht System über unendlichen Zeithorizont (Flussgleichgewicht wird zu stationärem Gleichgewicht)

Unterschiedliche Betrachtung:

- Operationale Analyse macht Aussagen über einen typischen Ablauf
- Stochastische Analyse macht Aussagen über alle möglichen Abläufe

Trotzdem existieren Analogien und Ergebnisse der operationalen Analyse gelten teilweise auch für stochastische Systeme!

Aber Vorsicht: Eine Übertragung der Ergebnisse für beliebige Systeme führt zu falschen Ergebnissen!!

Regeln der operationalen Analyse gelten oft für stochastische Systeme mit exponentiellen Ankunfts- und Bedienzeiten und liefern einfache Herleitungen für Ergebnisse über diese Systeme

Im Rahmen dieses Kapitels betrachten wir

3.1 Einfache Wartesysteme

3.2 Offene Warteschlangennetze

3.3 Geschlossene Warteschlangennetze

3.4 Stochastische Interpretation

3.5 Anwendungsbeispiele

3.6 Modellierung mit Markov-Prozessen

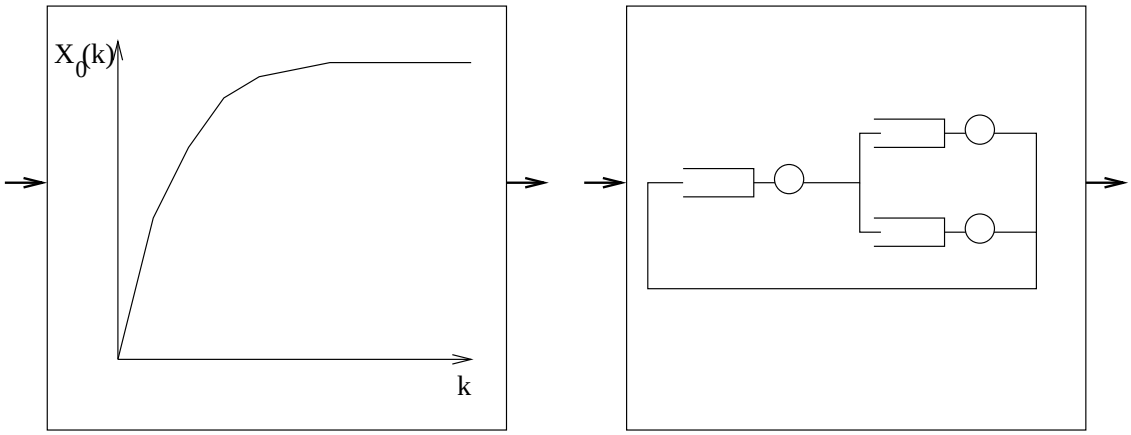
3.7 Zuverlässigkeit und Verfügbarkeit

3.8 Erweiterte Analysetechniken

3.1 Einfache Wartesysteme

Systeme können als sogenannte "black box" oder
"white box" Modelle analysiert werden

Die "black box" Sichtweise betrachtet nur
das Ein-/Ausgangsverhalten und nicht die interne Struktur



Charakterisierung des Ein-/Ausgangsverhalten durch:

- Auftragszahl k im System
- Durchsatz $X_0(k)$ bei k Aufträgen im System

Unsere "black box" Modelle korrespondieren zu
Zustands-/Transitionssystemen,
die einfache Wartesysteme beschreiben

Bis Kap. 3.1.5 Annahme exponentieller Zeiten d.h.
Gesetze der operationalen Analyse gelten
auch im stochastischen Fall

Einschub:

Exponentialverteilung und Poisson-Prozess

Kontinuierlichen ZV X unterliegt Exponentialverteilung mit Parameter λ:

Dichtefunktion: $f(x) = \lambda \cdot e^{-\lambda x}$

Verteilungsfunktion: $F(x) = P(X < x) = 1 - e^{-\lambda x}$

Erwartungswert: $1/\lambda$ Varianz: $1/\lambda^2$

Gedächtnislosigkeitseigenschaft

$$\begin{aligned} P(X < x + y | X > y) &= \frac{P(y<X<y+x)}{P(X>y)} \\ &= \frac{P(X<x+y)-P(X<y)}{P(X>y)} \\ &= \frac{(1-e^{-\lambda(x+y)})-(1-e^{-\lambda x})}{e^{-\lambda x}} \\ &= 1 - e^{-\lambda x} \end{aligned}$$

einzige kontinuierliche Verteilung mit dieser Eigenschaft

Poisson Prozess ist ein diskreter stochastischer Prozess $N(t)$ mit Parameter a:

$$P(N(t) = k) = \frac{e^{-at} \cdot (at)^k}{k!}$$

Es gilt:

$$P(N(s+t) - N(s) = k) = P(N(t) = k)$$

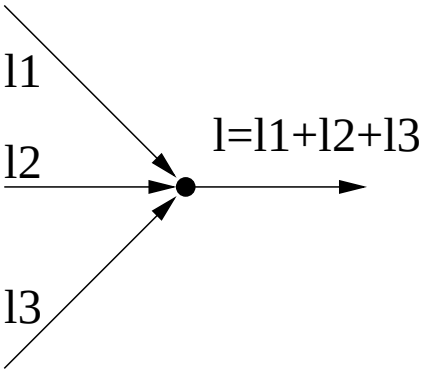
für alle $k \geq 0$ und $s, t \geq 0$

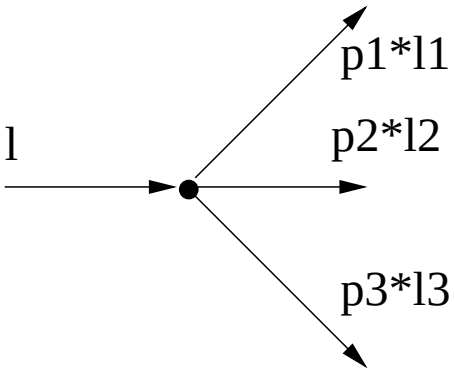
$$E(N(t)) = a \cdot t$$

- Falls $N(t)$ein Poisson-Prozess mit Parameter a ist, so sind die Zeiten zwischen zwei Ereignissen exponentiell verteilt mit Parameter $\lambda = a^{-1}$

- Falls X eine exponentiell verteilte ZV mit Parameter λ ist, so beschreibt die Anzahl der Ereignisse im Intervall $(0, T]$ einen Poisson Prozess $N(t)$ mit Parameter $a = \lambda^{-1}$

Einige weitere Eigenschaften von Exponentialverteilungen/Poisson-Prozessen:





Es gelte $p1 + p2 + p3 = 1$

Zusammenfassung oder zufällige Ausdünnung von Exponentialverteilungen/Poisson-Prozessen liefert Exponentialverteilungen/Poisson-Prozesse mit entsprechend geänderten Parametern

Sei $N(t)$ ein Poisson-Prozess und $X(t)$ ein beliebiger aber stationärer stochastischer Prozess

$N(t)$ und $X(t)$ seien unabhängig

Es werde folgendes Experiment im Intervall $(0, T]$ durchgeführt:

Jedesmal, wenn ein Ereignis des Poisson-Prozesses eintritt, wird der Zustand des Prozesses $X(t)$ protokolliert

Sei n die Anzahl der Ereignisse und x_i der $i$te protokollierte Zustand

Es gilt dann für alle y:

Falls $X(t)$ein zustandsdiskreter Prozess ist:

$$\lim_{T\to\infty} \frac{\sum_{i=1}^{n} \delta(x_i = y)}{n} = \frac{1}{T}\int_0^T P(X(t) = y)dt$$

Falls $X(t)$ein zustandskontinuierlicher Prozess ist:

$$\lim_{T\to\infty} \frac{\sum_{i=1}^{n} \delta(x_i \in (y, y+\Delta])}{n} = \frac{1}{T}\int_0^T P(X(t) \in (y, y+\Delta])dt$$

für beliebiges $\Delta > 0$

$\Rightarrow$ Poisson-Prozess beschreibt einen zufälligen Beobachter (Random-Observer)

3.1.1 Das verkleidete M/M/1 System

DB-Server, auf den von vielen Benutzer zugegriffen wird

Annahme: Ankunftsrate der Anfragen ist unabhängig von der Zahl wartender Anfragen (*infinite population case*)

Anfragen

- treffen mit Rate λ ein
- sind von identischer Struktur
- können in beliebiger Zahl am Server warten

Durchsatz sei lastunabhängig, es gelte $X_0(k) = \mu \ (k > 0)$

(unter stochastischen Annahmen beschreibt dies das M/M/1 System)

Ziel der Berechnung

- Wahrscheinlichkeit k Aufträge am Server p_k
- mittlere Anzahl Aufträge N
- mittlere Antwortzeit R
- mittlere Auslastung U
- mittlerer Durchsatz X

Erste Entscheidung: Zustandsbeschreibung festlegen

Hier: Ein Parameter, Anzahl Aufträge

Zustandsbeschreibung impliziert Annahmen:

Zukünftige Entwicklung hängt vom aktuellen Zustand ab!

Also hier Gedächtnislosigkeit des Prozesses bzgl.

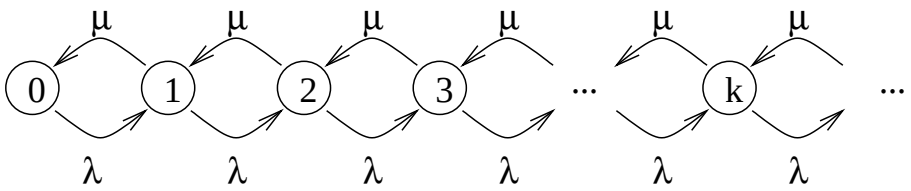
- Verweilzeit im Zustand
- Weg zum Zustand

Mögliche Zustandsübergänge

▷ von i nach $i+1$ mit Rate λ

▷ von i nach $i-1$ mit Rate μ falls $i > 0$

<u>Zustandsübergangsdiagramm</u>



Erster Schritt: Berechnung der p_k

daraus können andere Leistungsgrößen gewonnen werden

Zentrale Annahme: Flussgleichgewicht

Fluss herein = Fluss heraus (für jeden Zustand)

Dies gilt für beliebige Zustandsmengen!

Fluss aus einem Zustand =

Wahrscheinlichkeit im Zustand zu sein $\star$

Summe der abgehende Raten

Für Zustandmengen,
die jeweils die ersten k $(k = 0, 1, 2, \ldots)$
Zustände umfassen,
ergeben sich folgende Flussgleichgewichte

$\mu \cdot p_1 = \lambda \cdot p_0$

$\mu \cdot p_2 = \lambda \cdot p_1$

$\ldots$

$\mu \cdot p_k = \lambda \cdot p_{k-1}$

$\ldots$

Dies liefert

$$p_k = \tfrac{\lambda}{\mu} \cdot p_{k-1} = \tfrac{\lambda}{\mu} \cdot (\tfrac{\lambda}{\mu} \cdot p_{k-2}) = \left(\tfrac{\lambda}{\mu}\right)^k \cdot p_0$$

Berechnung von p_k als Funktion von p_0

Berechnung p_0:

p_k sind Wahrscheinlichkeiten $\Rightarrow$
$\sum_{k=0}^{\infty} p_k = 1$ und damit

$$\sum_{k=0}^{\infty} p_k = p_0 \cdot \sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k = 1$$

mit der Lösung

$$p_0 = \left[\sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k\right]^{-1} = 1 - \frac{\lambda}{\mu}$$

Beachte: die letzte Gleichheit gilt nur wenn $\lambda < \mu$!!

Für die Auslastung gilt

$U = 1 - p_0 = \lambda/\mu$ und damit auch

$$p_k = (1 - U) \cdot U^k$$

Für die mittlere Population gilt

$$N = \sum_{k=0}^{\infty} k \cdot p_k = (1 - U) \cdot \sum_{k=0}^{\infty} k \cdot U^k$$

Für $U < 1$ gilt

$$\sum_{k=0}^{\infty} k \cdot U^k = U \frac{d}{dU} \sum_{k=0}^{\infty} U^k = U \frac{d}{dU} \frac{1}{1-U} = \frac{U}{(1-U)^2}$$

und damit $N = U/(1 - U)$

Für den Durchsatz gilt (wie erwartet)

$X = U \cdot \mu = (\lambda/\mu) \cdot \mu = \lambda$

(auch hier mit der Annahme $U < 1$)

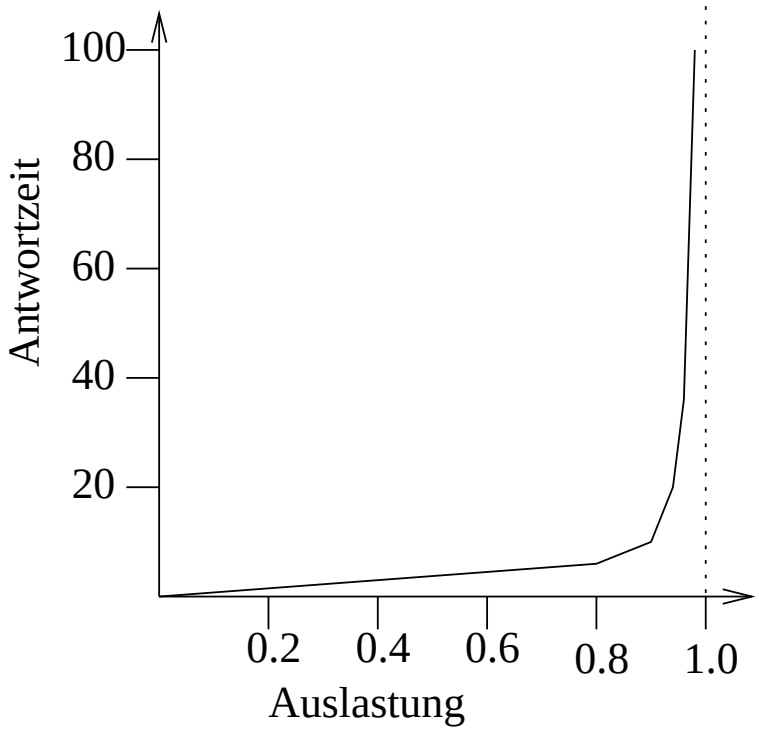
Antwortzeit lässt sich über den Satz von Little ermitteln

$R = N/X = (U/\lambda)/(1-U) = S/(1-U) = \frac{1}{\mu-\lambda}$

mit $S = 1/\mu$

Interpretation des Verhaltens:

- Falls die Auslastung klein ist, ist die Antwortzeit gleich der Bedienzeit
- Wenn die Auslastung groß ist (nahe bei 1) geht die Antwortzeit gegen unendlich



Schritte zur Problemanalyse

1. Zustandsdarstellung bestimmen

2. Zustandsraum aufbauen

3. Transitionen aus Ereignissen im System ableiten

4. Transitionsraten bestimmen

5. Flussgleichgewicht aufstellen

6. Wahrscheinlichkeiten p_k berechnen

Diese Schritte werden auch für
allgemeinere Systeme angewendet!

$\Rightarrow$ stationäre Analyse von Markov Prozessen

Hier Spezialfall Geburts-/Todesprozess!!

Allgemeines Vorgehen:

Lösung eines linearen Gleichungssystems
(etwas mehr dazu in 3.6)

3.1.2 Ein Modell mit endlicher Pufferkapazität

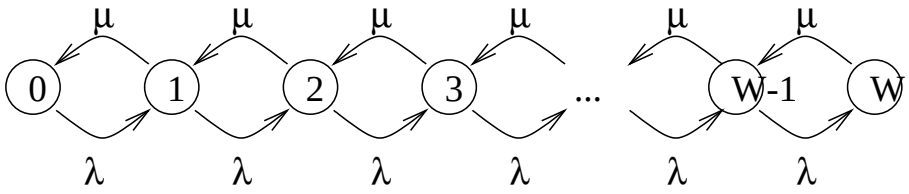
Im Unterschied zum vorherigen Fall nun
beschränkte Kapazität des Servers

Server speichert maximal W Aufträge,
weitere Aufträge werden abgewiesen

Zustandbeschreibung wie vorher

Aber endliche Zustandszahl $W + 1$

Zustandsübergangsdiagramm



Es gilt (Herleitung wie vorher)

$p_k = \left(\frac{\lambda}{\mu}\right)^k \cdot p_0$ für $k = 1, \ldots, W$ und

$$p_0 \cdot \sum_{k=0}^{W} \left(\frac{\lambda}{\mu}\right)^k = p_0 \cdot \left(\sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k - \left(\frac{\lambda}{\mu}\right)^{W+1} \sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k\right) =$$

$$p_0 \cdot \left[\frac{1-(\lambda/\mu)^{W+1}}{1-\lambda/\mu}\right] = 1$$

mit der Lösung $p_0 = \left[\frac{1-\lambda/\mu}{1-(\lambda/\mu)^{W+1}}\right]$

Berechnung der Leistungsgrößen

$$U = 1 - p_0 = \frac{(\lambda/\mu)\cdot[1-(\lambda/\mu)^W]}{1-(\lambda/\mu)^{W+1}}$$

Interessant ist auch die Anzahl der abgewiesenen Aufträge

Diese entspricht (unter exp. Ankünften!)

$$p_{loss} = p_W$$

Für die mittlere Population gilt

$$N = \sum_{k=0}^{W} k \cdot p_k = p_0 \cdot \sum_{k=0}^{W} k \cdot (\lambda/\mu)^k$$

unter Benutzung der Beziehung

$$\sum_{k=0}^{W} k \cdot a^k = \frac{W\cdot a^{W+2}-(W+1)\cdot a^{W+1}+a}{(1-a)^2}$$

gilt dann

$$N = \frac{(\lambda/\mu)[W(\lambda/\mu)^{W+1}-(W+1)(\lambda/\mu)^W+1]}{[1-(\lambda/\mu)^{W+1}](1-\lambda/\mu)} \text{ für } \lambda < \mu$$

und für den Durchsatz

$$X = U \cdot \mu = \frac{\lambda[1-(\lambda/\mu)^W]}{1-(\lambda/\mu)^{W+1}}$$

mittels des Satzes von Little $R = N/X$

3.1.3 Lastabhängigkeit in Ankunft und Bedienung

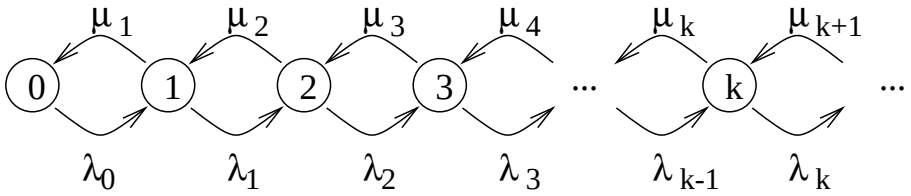
Bisherige Voraussetzung:

konstante Ankunftsrate und Bedienrate

Hier nun verallgemeinerte Sicht

- Ankunftsrate bei k Aufträgen im System λ_k
- Bedienrate bei k Aufträgen im System μ_k

Zustandsübergangsdiagramm



und in ähnlicher Form für endliche Kapazität W

Auf Basis des Flussgleichgewichts gilt

$\lambda_{k-1} \cdot p_{k-1} = \mu_k \cdot p_k$

und damit auch

$p_k = p_0 \cdot \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}}$

Berechnung von p_0

$p_0 = \left[\sum_{k=0}^{\infty} \prod_{i=0}^{k-1} \frac{\lambda_i}{\mu_{i+1}}\right]^{-1}$

Berechnung der Leistungsgrößen

Auslastung

$$U = 1 - p_0$$

Durchsatz

$$X = \sum_{k=1}^{\infty} \mu_k \cdot p_k$$

Population

$$N = \sum_{k=1}^{\infty} k \cdot p_k$$

Antwortzeit

$$R = \frac{N}{X} = \frac{\sum_{k=1}^{\infty} k \cdot p_k}{\sum_{k=1}^{\infty} \mu_k \cdot p_k}$$

Analyse funktioniert falls p_0 berechenbar

(damit muss nicht unbedingt $\lambda_k < \mu_k$ für alle k gelten!)

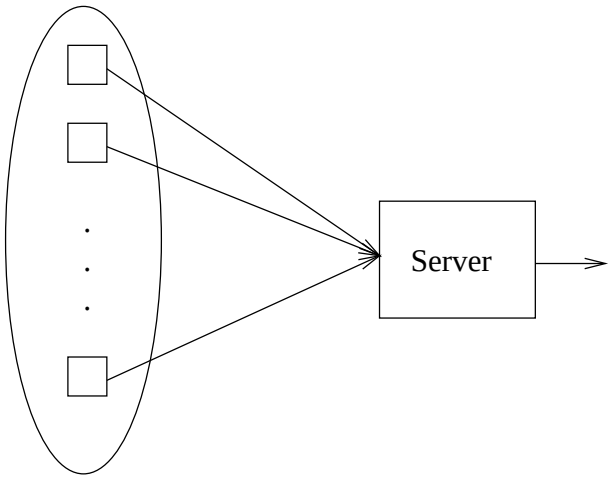
Analyse dieses allgemeinen Systems ist Basis für Analyse vieler spezieller Modelle

3.1.4 Weitere M/M/.. Systeme

Modelle des vorherigen Abschnitts als
spezielle Modelle interpretierbar

- Population im System endlich oder unendlich
- Bedienrate konstant oder lastabhängig
- Kapazität unbeschränkt oder beschränkt

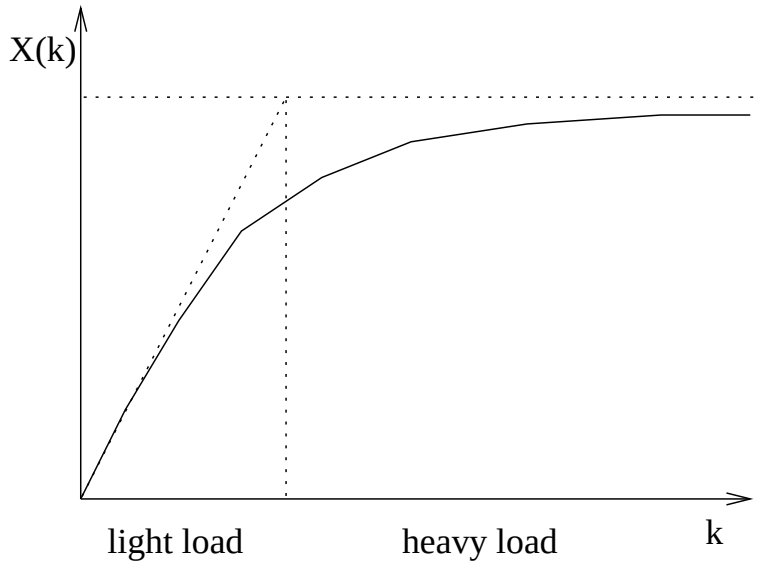
Im WWW Umfeld im allgemeinen Systeme mit
unbeschränkter Population



Clients
(unbeschränkte Zahl)

d.h. Ankunftsrate ist konstant und
unabhängig von der Population
damit ist $\lambda_k = \lambda$ und $\mu_k = X(k)$

Typischer Verlauf von $X(k)$



Die legt folgende Beschreibung nahe

$$\mu_k = \begin{cases} X(k) & k \leq J \\ \\ X(J) & k \geq J \end{cases}$$

Notationen

$\beta(k) = \prod_{i=1}^{k} X(k)$ und $\rho = \lambda / X(J) < 1$

Dann gilt

$$p_0 = \left[1 + \sum_{k=1}^{J} \frac{\lambda^k}{\beta(k)} + \frac{\lambda^J}{\beta(J)} \frac{\rho}{1-\rho}\right]^{-1}$$

$$p_k = \begin{cases} p_0 \lambda^k / \beta(k) & k \leq J \\ \\ p_0 X(J)^J \rho^k / \beta(J) & k > J \end{cases}$$

Herleitung der Population:

$$
\begin{aligned}
N &= p_0 \cdot \sum_{k=1}^{J} \frac{k \cdot \lambda^k}{\beta(k)} + \sum_{k=J+1}^{\infty} \frac{\lambda^J}{\beta(J)} \cdot k \cdot \rho^{k-J} \\
&= p_0 \cdot \sum_{k=1}^{J} \frac{k \cdot \lambda^k}{\beta(k)} + \frac{\lambda^J}{\beta(J)} \cdot \sum_{k=J+1}^{\infty} k \cdot \rho^{k-J} \\
&= p_0 \cdot \sum_{k=1}^{J} \frac{k \cdot \lambda^k}{\beta(k)} + \frac{\lambda^J}{\beta(J)} \cdot \sum_{l=1}^{\infty} (J+l) \cdot \rho^{l} \\
&= p_0 \cdot \sum_{k=1}^{J} \frac{k \cdot \lambda^k}{\beta(k)} + \frac{\lambda^J}{\beta(J)} \cdot \left(J \cdot \sum_{l=1}^{\infty} \rho^{l} + \sum_{l=1}^{\infty} l \cdot \rho^{l} \right) \\
&= p_0 \cdot \sum_{k=1}^{J} \frac{k \cdot \lambda^k}{\beta(k)} + \frac{\lambda^J}{\beta(J)} \cdot \left(\frac{J \cdot \rho}{1-\rho} + \frac{\rho}{(1-\rho)^2} \right) \\
&= p_0 \cdot \sum_{k=1}^{J} \frac{k \cdot \lambda^k}{\beta(k)} + \frac{\lambda^J}{\beta(J)} \cdot \left(\frac{J \cdot \rho \cdot (1-\rho) + \rho}{(1-\rho)^2} \right) \\
&= p_0 \cdot \sum_{k=1}^{J} \frac{k \cdot \lambda^k}{\beta(k)} + \frac{\lambda^J \cdot \rho \cdot (J \cdot (1-\rho) + 1)}{\beta(J) \cdot (1-\rho)^2}
\end{aligned}
$$

Da für den Durchsatz $X = \lambda$ gilt

(unbeschränkte Kapazität!)

ist auch die Verweilzeit/Antwortzeit (über Little)

in geschlossener Form gegeben

$$R = p_0 \cdot \sum_{k=1}^{J} \frac{k \cdot \lambda^{k-1}}{\beta(k)} + \frac{\lambda^{J-1} \cdot \rho \cdot (J \cdot (1-\rho) + 1)}{\beta(J) \cdot (1-\rho)^2}$$

Diese Modell umfasst das M/M/m System!

System mit beschränktem Warteraum

Es gilt

$$\mu_k = \begin{cases} X(k) & k \leq J \\ \\ X(J) & k = J+1, \cdots, W \end{cases}$$

und damit auch

$$p_0 = \left[1 + \sum_{k=1}^{J} \frac{\lambda^k}{\beta(k)} + \frac{\rho \lambda^J (1-\rho^{W-J})}{\beta(J)(1-\rho)}\right]^{-1}$$

$$p_k = \begin{cases} p_0 \lambda^k / \beta(k) & k \leq J \\ \\ p_0 X(J)^J \rho^k / \beta(J) & k = J+1, \ldots, W \end{cases}$$

System mit endlicher Population

Bisher untersucht offene Modelle

(potentiell unendliche Population)

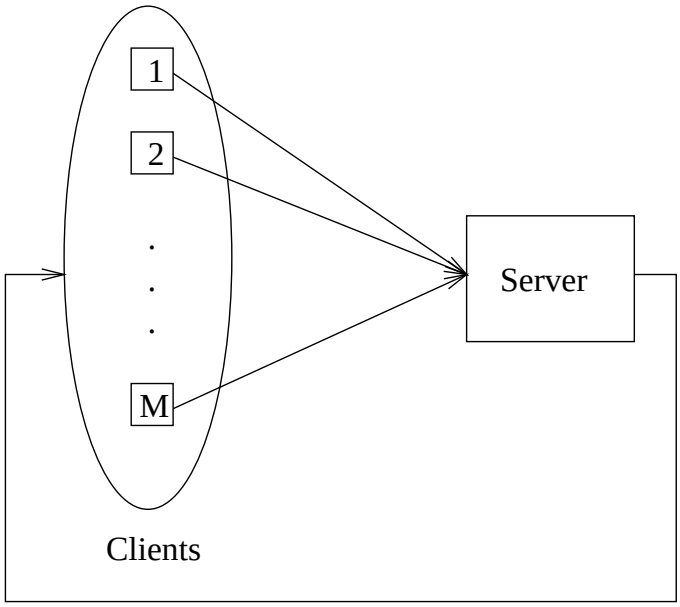
Natürliche Sichtweise für WWW Anwendungen

In Intranets ist die Population beschränkt,

d.h. Ankunftsrate fällt mit der Population am Server

(jeder Benutzer arbeitet lokal oder wartet auf Antwort)

$\Rightarrow$ geschlossenes Modell



Feste Anzahl von Aufträgen:

Jeder Auftrag

- befindet sich am Client (lokales arbeiten) oder
- befindet sich am Server (Anfrage wird bearbeitet)

Annahmen:

- Population M im Modell
- Benutzer hat durchschnittliche Denkzeit Z,
  die bis zum Abschicken der nächsten Anfrage vergeht
  (d.h. mit Rate $1/Z$ sendet jeder (!) Client
  seine Anfrage zum Server)

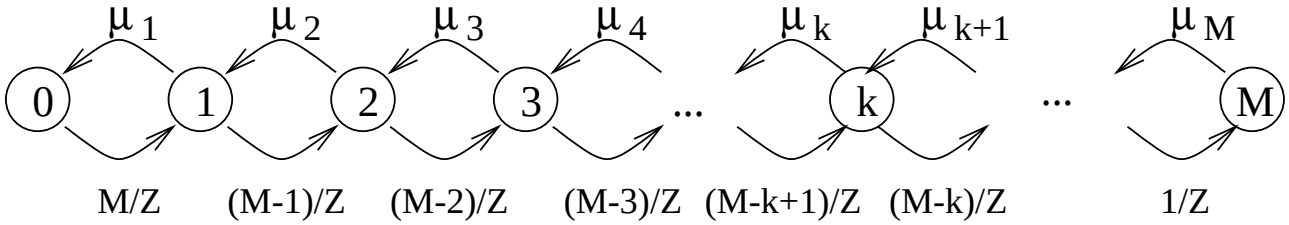
Zustandbeschreibung:

Anzahl Anfragen am Server $(0, \ldots, M)$

Im Zustand k sind k Anfragen beim Server in Bearbeitung und $M - k$ Clients "denken" über eine neue Anfrage nach

$\Rightarrow \lambda_k = (M - k)/Z$

Zustandsübergangsdiagramm



Lastunabhängige Bedienraten

Nach den allgemeinen Formeln gilt dann

$p_k = p_0 \cdot \prod_{i=0}^{k-1} \frac{M-i}{\mu \cdot Z} = p_o \cdot \frac{M!}{(M-k)! \cdot (\mu \cdot Z)^k}$

mit

$p_0 = \left[\sum_{k=0}^{M} \frac{M!}{(M-k)! \cdot (\mu \cdot Z)^k} \right]^{-1}$

Bemerkung zur Berechnung:

Für große M ist $M!$ nicht direkt berechenbar,

deshalb sollte die Form

$p_k = p_0 \cdot \frac{M}{\mu Z} \cdot \frac{M-1}{\mu Z} \cdot \ldots \cdot \frac{M-(k-1)}{\mu Z}$

zur Berechnung verwendet werden

(entsprechend zur Berechnung von p_0)

$$p_0 = \left(1 + \sum_{k=1}^{M} \left(\prod_{l=0}^{k-1} \frac{M-l}{\mu Z} \right) \right)^{-1}$$

Beispiel: Lastabhängige Bedienraten

Bedienrate

$$\mu_k = \begin{cases} X(k) & k = 1, \ldots, J \\ \\ X(J) & k > J \end{cases}$$

Berechnung von p_k:

$$p_k = \begin{cases} p_0 \cdot \frac{M!}{(M-k)! Z^k \beta(k)} & k = 1, \ldots, J \\ \\ p_0 \cdot \frac{M! X(J)^J}{(M-k)! [Z \cdot X(J)]^k \beta(J)} & k > J \end{cases}$$

mit

$p_0 = [1 + \sum_{k=1}^{J} \frac{M!}{(M-k)! Z^k \beta(k)} + \frac{X(J)^J}{\beta(J)} \cdot \sum_{k=J+1}^{M} \frac{M!}{(M-k)! [Z \cdot X(J)]^k}]^{-1}$

$X = \sum_{k=1}^{M} X(k) \cdot p_k$ und $N = \sum_{k=1}^{M} k \cdot p_k$

3.1.5 Nicht-Markovsche Bedienzeiten

Inwieweit gelten die bisherigen Annahmen

für allgemeine Systeme?

Implizite Annahmen bei allen Analysen:

1. Ankommender Auftrag, der $k-1$ $(k>1)$ Aufträge antrifft, muss durchschnittlich $1/\mu_k$ Zeiteinheiten bis zur Beendigung der aktuellen Bedienung warten!

2. Ankommender Auftrag trifft mit Wahrscheinlichkeit p_k k Aufträge an!

Annahmen gelten für

exponentiell verteilte Ankunfts-/Bedienzeiten

Aber leider nicht für andere Verteilungen

Beispiel D/D/1:

- Deterministische Ankunftszeitabstände mit Zwischenankunftszeit a

- Deterministische Bedienzeiten mit Bedienzeit b

Sei $a > b$ (damit Flussgleichgewicht gilt!)

dann wird ein ankommender Auftrag immer(!)

ein leeres System antreffen

Gleichzeitig ist aber

$p_0 = 1 - b/a$, $p_1 = b/a$ und $p_k = 0$ für $k > 1$

Also: Vorsicht bei der Übertragung operationaler
Zusammenhänge auf stochastische Systeme!

Allgemein gilt Littles Gesetz
(falls Flussgleichgewicht gegeben)

Bedingung 2. gilt, falls Zwischenankunftszeiten
negativ-exponentiell verteilt sind (PASTA-Eigenschaft)

(PASTA = Poisson Arrivals See Time Averages
durch "Random-Observer"-Eigenschaft des Poisson-Prozesses)

Wir betrachten im folgenden diesen Fall

- Negativ exponentiell verteilte Zwischenankunftszeiten mit Rate λ
- Allgemein verteilte Bedienzeiten mit Mittelwert S und Standardabweichung σ

Annahme $1/\lambda > S$ (Flussgleichgewicht)

Für die Auslastung gilt $U = \lambda \cdot S$

PASTA-Eigenschaft sagt aus, dass ankommender Auftrag mit Wahrscheinlichkeit p_k k-Aufträge vor sich sieht

p_k ist unbekannt: Kann man trotzdem etwas berechnen?

Wie groß ist die mittlere Wartezeit eines Auftrags,
der k (> 0) Aufträge antrifft

- Restbedienzeit des Auftrags in Bedienung
- $(k-1) \cdot S$ für die Aufträge in der Warteschlange

Also Ermittlung der Restbedienzeit

Im exponentiellen Fall:
Restbedienzeit S_{res} gleich S (Gedächtnislosigkeit)!

Allgemein kann es sein, dass die Restbedienzeit
kleiner oder größer als die mittlere Bedienzeit ist

Wir führen dazu Faktor κ ein, um den die Bedienzeit
verlängert/verkürzt wird

$$S_{res} = (1 - \kappa) \cdot S$$

Damit gilt für die Verweilzeit

$$R = S + S \cdot N - U \cdot \kappa \cdot S$$

es gilt nach Little $N = R \cdot X$ und damit

$$R = S + S \cdot R \cdot X - U \cdot \kappa \cdot S$$

da $U = S \cdot X$ gilt auch

$$R \cdot (1 - U) = S \cdot (1 - U \cdot \kappa) \Rightarrow R = S \cdot \frac{1 - U \cdot \kappa}{1 - U}$$

Bleibt die Ermittlung von κ

PASTA Eigenschaft sagt aus, dass ankommender Auftrag zu einem zufälligen Zeit während der Bedienung ankommt

Einige Spezialfälle:

- Exponentielle Bedienzeiten $\kappa = 0$ und damit $S_{res} = S$
- Konstante Bedienzeiten $\kappa = 0.5$ und damit $S_{res} = 0.5 \cdot S$

Allgemein kann man κ mit Hilfe von S und σ ausdrücken

Quadrierter Variationskoeffizient: $VK^2 = \left(\frac{\sigma}{S}\right)^2$

Es gilt dann $\kappa = 0.5 \cdot (1 - VK^2)$

oder direkt eingesetzt

$R = S + \frac{U \cdot S \cdot (1 + VK^2)}{2 \cdot (1 - U)}$

die berühmte Pollaczek-Khinchine Formel für die Verweilzeit in M/G/1 Systemen

Da $X = \lambda$ können wir N direkt über Little berechnen

Analyse allgemeiner G/M/1 oder G/G/1 System ist deutlich komplexer!

(und betrachten wir in eingeschränkter Form in 3.6)

3.2 Offene Warteschlangennetze

Bisherige Sichtweise "black box"

System charakterisiert als Durchsatzfunktion $X(k)$ und

dargestellt als einzelne Station

in unterschiedlichen Ausprägungen

Bei detaillierter Betrachtung der Systeme besteht ein System aus unterschiedlichen Komponenten, die jeweils als einzelne Stationen beschrieben werden

$\Rightarrow$Warteschlangennetze

Grundidee: Aufträge durchlaufen Bedienstationen

- hier vorgestellt spezielle Klasse von Netzen (separable- oder Produktformnetze) mit effizienten Analysemöglichkeiten
- erst einmal offene Netze, d.h. Umgebung produziert Aufträge

Eingabegrößen

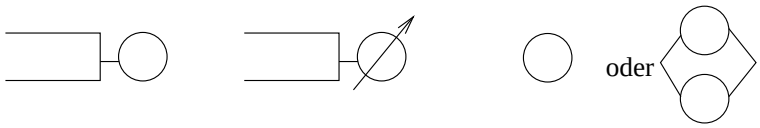
- Maschinenmodell: Stationen
- Lastmodell: Ankunftsrate(n), Bedienanforderungen

Ausgabegrößen:

- Antwortzeiten, Durchsätze als Mittelwerte
- Anzahl Aufträge an einer Station oder im Netz als Mittelwerte (evtl. auch als Verteilungen)

Nutzen der Modelle

- offene Netze geben die Sicht eines Web-Servers im Internet wieder
- Anfragen nach Webseiten, Transaktionen, ... sind Bedienanforderungen von Aufträge
- Web-Server, Anwendungs-Server, ... sind Stationen
- bei vielen Anfragen entstehen Wartezeiten an einem Server, Antwortzeiten werden länger
- mit offenen Netzen lassen sich Systeme modellieren, bei denen der Flaschenhals nicht vorab feststeht (je nach Last "wandert" der Flaschenhals)



lastunabhängige Station | lastabhängige Station | Verzögerungs–station

Man unterscheidet folgende Stationstypen

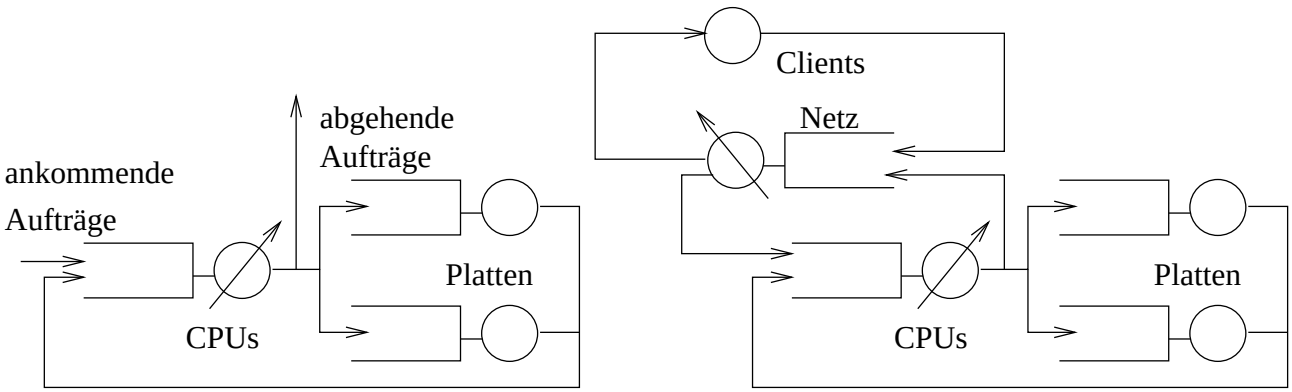
- lastunabhängige Stationen mit konstanter Bedienzeit $S(n) = S$ und einem Bediener
- lastabhängige Stationen mit lastabhängiger Bedienzeit $S(n)$ (umfasst Stationen mit mehr als einem Bediener)
- Verzögerungsstationen mit lastunabhängiger Bedienzeit $S(n) = S$ und einem Bediener pro Auftrag (keine Wartezeiten)

Typen von Warteschlangennetzen

Unterscheidung in

- Einklassennetze
  (alle Aufträge sind (stochastisch) identisch)
- Mehrklassennetze
  (Aufträge gehören zu Klassen, die Klassenzugehörigkeit beschreibt das (stochastische) Verhalten)
- Offene Netze
  (Aufträge betreten und verlassen das Netz)
- Geschlossene Netze
  (im Netz zirkuliert eine feste Auftragspopulation)
- Gemischte Netze
  (einzelne Klassen sind geschlossen, andere offen)

Beispiele:



3.2.1 Einklassennetze

Wir betrachten offene Netze mit einer Auftragsklasse und Stationen vom Typ

- lastunabhängige Station oder
- Verzögerungsstation

Notationen

Eingabeparameter:

- λ Ankunftsrate
- K Anzahl Stationen
- X_0 Netzdurchsatz, da Flussgleichgewicht gilt $X_0 = \lambda$
- V_i mittlere Anzahl von Besuchen an Station i
- S_i mittlere Bedienzeit an Station i

Resultatwerte:

- W_i mittlere Wartezeit pro Besuch an Station i
- U_i Auslastung Station i
- $R_i = S_i + W_i$ mittlere Verweilzeit pro Besuch an i
- $R'_i = V_i \cdot R_i$ Gesamtverweilzeit eines Auftrags an i
- $R_0 = \sum_{i=1}^{K} R'_i$ Gesamtverweilzeit im Netz
- n_i mittlere Population an Station i
- $N = \sum_{i=1}^{R} n_i$ mittlere Population im Netz

Zentrales Resultat für offene Netze

Ankunftstheorem:

Ein Auftrag, der an einer Station ankommt,
trifft dort im Mittel n_i Aufträge an

Weiterhin gilt $X_i = V_i \cdot \lambda$ und $U_i = X_i \cdot S_i = \lambda \cdot D_i$

Damit gilt für lastunabhängige Stationen

$R_i = S_i + n_i \cdot S_i$ oder mittels $n_i = X_i \cdot R_i$ (Little)

$R_i = S_i + X_i \cdot R_i \cdot S_i \Rightarrow R_i = S_i/(1 - X_i \cdot S_i) = S_i/(1 - U_i)$

mit $D_i = V_i \cdot S_i$ gilt $R'_i = V_i \cdot R_i = \frac{D_i}{1-U_i}$

Mittlere Population folgt nach Little

$n_i = R_i \cdot X_i = \frac{S_i}{1-U_i} \cdot \frac{U_i}{S_i} = \frac{U_i}{1-U_i}$

Für Verzögerungsstationen gilt

$R_i = S_i$ und $R'_i = S_i \cdot V_i$ sowie $n_i = S_i \cdot X_i = D_i \cdot \lambda$

$\Rightarrow$ alle Leistungsgrößen sind berechenbar

Maximale Ankunftsrate, die ein Netz verkraften kann

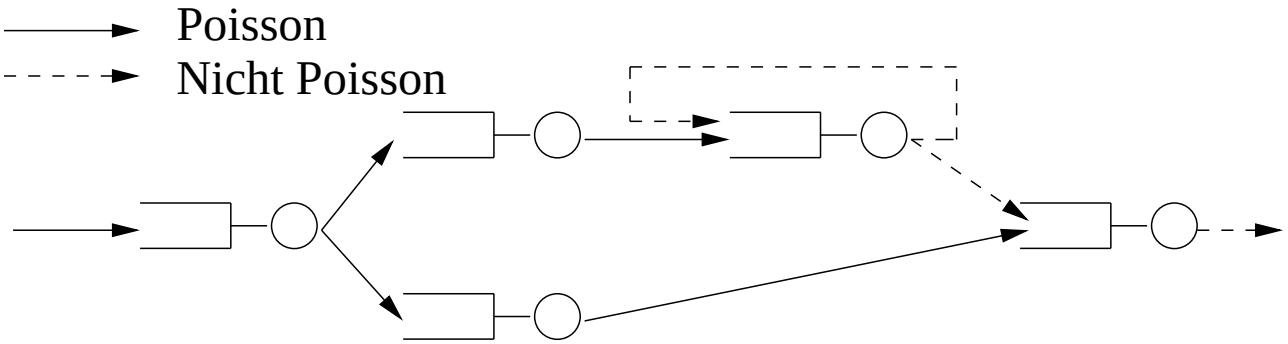
$\lambda < \frac{1}{\max_{i=1}^{K} D_i}$

Bedeutung des Ankunftstheorems:

Ankunftstheorem sagt aus, dass sich eine Station in einem offenen Netz wie eine isolierte Station mit identischen Parametern und Poisson-Ankünften mit Rate λ verhält (siehe PASTA-Eigenschaft auf Folie 26)

Sind die Flüsse im Netz tatsächlich Poisson-Prozesse?

Ein Beispiel:



- Poisson-Flüsse nur von außen und in Vorwärtsrichtung
- Zyklen führen zu komplexen Flüssen, die Abhängigkeiten induzieren
- Trotzdem wird das Netz so analysiert, als ob alle Flüsse Poisson-Prozesse wären!
- Zur Genauigkeit der Analyse und Gültigkeit des Ankunftstheorems später etwas mehr!

3.2.2 Mehrklassennetze

Netz mit R Klassen, jede Klasse beschreibt ein individuelles Verhaltensmuster

Annahme: Keine Klassenwechsel

Erweiterung der Notation durch zusätzlichen Klassenindex

$R_{i,r}$ Verweilzeit Klasse r an Station i

Vektor der Ankunftsraten $\underline{\lambda} = (\lambda_1, \ldots, \lambda_R)$

Stationszustand nun beschrieben durch Vektor

$$\underline{n_i} = (n_{i,1}, \ldots, n_{i,R})$$

Für jede Klasse gilt dann $D_{i,r} = V_{i,r} \cdot S_{i,r}$

Damit kann man die Auslastung jeder lastunabhängigen Station berechnen

pro Klasse $U_{i,r} = \lambda_r \cdot V_{i,r} \cdot S_{i,r} = \lambda_r \cdot D_{i,r}$

insgesamt $U_i = \sum_{r=1}^{R} U_{i,r} < 1$

Ein Ankunftsratenvektor $\underline{\lambda}$ garantiert

das Flussgleichgewicht, falls

$\max_{i=1}^{K}(\sum_{r=1}^{R} \lambda_r \cdot V_{i,r} \cdot S_{i,r}) < 1$

Angenommen, die Station würde sich wie eine isolierte M/M/1 Station verhalten (analog zum Einklassenfall), dann wäre $n_i = \frac{U_i}{1-U_i}$ die mittlere Population (über alle Klassen)

Unter der Voraussetzung, dass der prozentuale Anteil der Klasse r Aufträge proportional zu ihrer Auslastung ist, gilt $n_{i,r} = \frac{U_{i,r}}{U_i} \cdot n_i = \frac{U_{i,r}}{1-U_i}$

Zwei mögliche Überlegungen zur Berechnung der Verweilzeit

1. Vor dem Auftrag befinden sich n_i Aufträge, die erst bedient werden müssen. Davon sind $n_{i,r}$ Aufträge aus Klasse r, deren Bedienzeit beträgt $S_{i,r}$

2. Mit dem Auftrag befinden sich n_i weitere Aufträge an der Station. Die Bedienkapazität wird gleichmäßig aufgeteilt, so dass der Auftrag $1/(n_i + 1)$ von der Gesamtkapazität bekommt

Wir benutzen erst einmal Sichtweise 2) und erhalten $R_{i,r} = (n_i + 1) \cdot S_{i,r} = \frac{S_{i,r}}{1-U_i}$

Für Verzögerungsstationen gilt natürlich $R_{i,r} = S_{i,r}$

3.3 Geschlossene Warteschlangennetze

Geschlossene Warteschlangennetze werden zur Modellierung von Systemen mit beschränkter Population eingesetzt

(z.B. C/S-System mit fester Population,
Computer mit maximaler Jobzahl)

Basis der Analysen:

Ankunftstheorem für geschlossene Netze

Ein Auftrag, der an einer Station ankommt, trifft die mittlere Population im Netz ohne sich selbst an

(Reiser/Lavenberg 1980)

Berechnung damit rekursiv über die Population im Netz

Basisalgorithmus Mittelwertanalyse (MVA)

Dies ist nur einer von mehreren Analysealgorithmen
(andere Convolution, LBANC)

MVA ist intuitiv und einfach zu realisieren

Im folgenden

- Einklassennetze
- Mehrklassennetze
- Netze mit lastabhängigen Stationen

3.3.1 Einklassennetze

Notationen:

Größen in Abhängigkeit der Netzpopulation n

Also

- $n_i(n)$ mittlere Population an i bei Population n im Netz
- $R_i(n)$ mittlere Verweilzeit an i bei Population n im Netz

  . . .

Sei $n_i^a(n)$ mittlere Population, die ein Auftrag antrifft,
wenn er Station i betritt und n Aufträge im Netz sind

Nach dem Ankunftstheorem gilt

$$n_i^a(n) = n_i(n-1)$$

Wir betrachten nur Verzögerungsstationen und
lastunabhängige Stationen

Verweilzeit in Station i: Wartezeit + Bedienzeit

$$R_i(n) = \begin{cases} S_i & \text{falls Verzögerungsstation} \\ \\ S_i \cdot (1 + n_i(n-1)) & \text{sonst} \end{cases}$$

Zu Berechnung der Gesamtverweilzeit im Netz werden die mittleren Besuchszahlen an den Stationen benötigt

In offenen Netzen ist die Interpretation klar:

$V_i :=$ Anzahl Besuche eines Auftrags an Station i
zwischen Ankunft und Abgang

Wie definiert man V_i im geschlossenen Netz?

Aufträge verlassen das Netz nicht,
deshalb $V_i = \infty$ nach obiger Interpretation

Alternative: Definition eines Arbeitszyklus pro Auftrag
(z.B. vollständige Abarbeitung einer DB-Anfrage)

$\Rightarrow$ Verhältnis der V_i ist vom Modell festgelegt,
nicht aber die konkreten Werte!

$\Rightarrow$ Berechnung der Resultate bzgl.
eines frei wählbaren Faktors (Zeitskala)

Unter diesen Voraussetzungen:

Gesamtverweilzeit an Station i

$$R'_i(n) = V_i \cdot R_i(n)$$

Gesamtverweilzeit im Netz

$$R_0(n) = \sum_{i=1}^{K} R'_i(n)$$

Damit gilt für den Durchsatz

$X_0(n) = \frac{n}{R_0(n)} = \frac{n}{\sum_{i=1}^{K} V_i \cdot R_i(n)}$

und

$X_i(n) = V_i \cdot X_0(n)$

Nach Little folgt

$n_i(n) = X_i(n) \cdot R_i(n) = X_0(n) \cdot R_i'(n)$

Berechnung der Größen damit rekursiv

beginnend mit $n_i(0) = 0$

Algorithmus

for $(n = 1$ *to* $N)$ *do begin*

for $(i = 1$ *to* $K)$ *do*

if $(i =$ *Verzögerungsstation)* *then*

$R_i(n) = S_i(n)$;

else

$R_i(n) = S_i \cdot (1 + n_i(n-1))$;

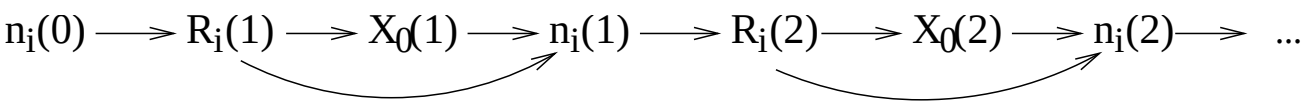
$X_0(n) = n/(\sum_{k=1}^{K} V_k \cdot R_k(n))$;

for $(i = 1$ *to* $K)$ *do*

$n_i(n) = X_0(n) \cdot V_i \cdot R_i(n)$;

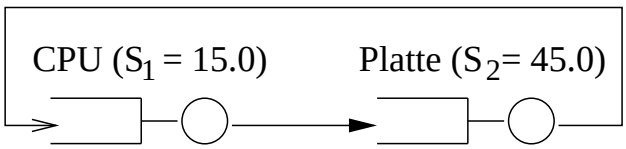
end

Ablauf des Algorithmus:



Aufwand $O(K)$ und Speicherplatz $O(K)$

Beispiel:



n	R_{CPU}	R_{Pl}	R_0	X_0	n_{CPU}	n_{Pl}
0	0.000	0.000	0.000	0.000	0.000	0.000
1	15.00	45.00	60.00	0.0167	0.250	0.750
2	18.75	78.75	97.50	0.0205	0.385	1.615
3	20.77	117.69	138.46	0.0217	0.450	2.550
4	21.75	159.75	181.50	0.0220	0.479	3.521

Wie zu erwarten ist die Platte der Flaschenhals und der Durchsatz konvergiert schnell gegen das Maximum $1.0/45 \approx 0.0222$

Da keine Ankünfte von außen erfolgen, ist ein geschlossenes Netz immer stabil (d.h. es erreicht immer den stationären Zustand)

3.3.2 Mehrklassennetze

Vorgehen im Mehrklassenfall unterscheidet sich
nicht grundsätzlich vom Einklassenfall

Population ist nun ein Vektor

$\underline{N} = (N_1, \ldots, N_R)$

Population in einer Station bei Population $\underline{N}$ im Netz

$n_i(\underline{N}) = \sum_{r=1}^{R} n_{i,r}(\underline{N})$

wobei $n_{i,r}(\underline{N})$:= mittlere Population von Klasse r an
Station i bei Population $\underline{N}$ im Netz

Notationen

$\underline{N} - \underline{1}_r = (N_1, \ldots, N_{r-1}, N_r - 1, N_{r+1}, \ldots, N_R)$

(d.h. ein Auftrag der Klasse r weniger im Netz)

$\underline{0}$ ist der Nullvektor

Es gilt offensichtlich

$n_{i,r}(\underline{0}) = 0$

Analog zum Einklassenfall gelten rekursive Beziehungen

Verweilzeit

$$R_{i,r}(\underline{n}) = \begin{cases} S_{i,r} & \text{falls Verzögerungsstation} \\ S_{i,r} \cdot (1 + n_i(\underline{n} - \underline{1}_r)) & \text{sonst} \end{cases}$$

Durchsatz Klasse r

$$X_{0,r}(\underline{n}) = \frac{n_r}{\sum_{i=1}^{K} V_{i,r} \cdot R_{i,r}(\underline{n})}$$

Gesamtdurchsatz

$$X_0(\underline{n}) = \sum_{r=1}^{R} X_{0,r}(\underline{n})$$

Mittlere Population pro Klasse (nach Little)

$$n_{i,r}(\underline{n}) = X_{0,r}(\underline{n}) \cdot V_{i,r} \cdot R_{i,r}(\underline{n})$$

Mittlere Population über alle Klassen

$$n_i(\underline{n}) = \sum_{r=1}^{R} n_{i,r}(\underline{n})$$

Aus diesen Formeln lässt sich ein Algorithmus formulieren

(Mehrklassen MVA)

Problem:

Zur Berechnung der Werte für $\underline{n}$ werden

alle Werte für $\underline{n} - \underline{1}_r$ benötigt

$\Rightarrow$

Aufwand wächst schnell mit Anzahl

Klassen und Stationen

Zeitbedarf $O(K \cdot R \cdot \prod_{r=1}^{R}(N_r + 1))$ Speicherplatz $O(K \cdot R)$

Algorithmus für mehrere Klassen

for $(j_1 = 0$ *to* $N_1)$ *do begin*

for $(j_2 = 0$ *to* $N_2)$ *do* $\{$

$\ldots$

for $(j_R = 0$ *to* $N_R)$ *do* $\{$

$\underline{N} = (j_1, \ldots, j_R)$;

for $(r = 1$ *to* $R)$ *do* $\{$

$X_{0,r}(\underline{N}) = 0$;

if $(j_r > 0)$ $\{$

for $(i = 1$ *to* $K)$ *do* $\{$

if $(i =$ *Delay*$)$

$R_{i,r}(\underline{N}) = S_{i,r}$;

else

$R_{i,r}(\underline{N}) = S_{i,r} * (1 + n_i(\underline{N} - \underline{1}_r))$;

$X_{0,r}(\underline{N}) += R_{i,r}(\underline{N})$;$\}$ $\}$

$X_{0,r}(\underline{N}) = j_r / X_{0,r}(\underline{N})$; $\}$

for $(i = 1$ *to* $K)$ *do* $\{$

for $(r = 1$ *to* $R)$ *do*

$n_i(\underline{N}) += X_{0,r}(\underline{N}) + R_{i,r}(\underline{N})$;

$\}\ldots\}$

Oft verwendete Approximation(nach Schweitzer)

Annahme:

Anzahl der Aufträge an einer Station steigt

proportional zur gesamten Auftragszahl

$\frac{n_{i,r}(\underline{N}-\underline{1}_r)}{n_{i,r}(\underline{N})} = \frac{N_r-1}{N_r}$

Damit gilt

$n_{i,r}(\underline{N}-\underline{1}_r) = \frac{N_r-1}{N_r} \cdot n_{i,r}(\underline{N})$

Dies Darstellung definiert einen **Fixpunktansatz**

1. Initialisiere $n_{i,r}(\underline{N})$ z.B. mit $N_r \cdot V_{i,r} / \sum_{j=1}^{K} V_{j,r}$
2. Berechne $n_{i,r}(\underline{N}-\underline{1}_r) = n_{i,r}(\underline{N}) \cdot \frac{N_r-1}{N_r}$
3. Bestimme $R_{i,r}(\underline{N})$ und $X_{0,r}(\underline{N})$ mit MVA Formeln
4. Berechne $n^e_{i,r}(\underline{N}) = X_{0,r}(\underline{N}) \cdot V_{i,r} \cdot R_{i,r}(\underline{N})$
5. Falls $\max_{i,r} \left| n^e_{i,r}(\underline{N}) - n_{i,r}(\underline{N}) \right| < \varepsilon$
   beende Berechnung
6. $n_{i,r}(\underline{N}) = n^e_{i,r}(\underline{N})$ und fahre bei 2. fort

Algorithmus benötigt in der Regel wenige Iterationen und ist für Netze mit vielen Klassen und großen Populationen deutlich effizienter als exakter MVA

Fehler ist in diesen Fällen oft vernachlässigbar

Gemischte Modelle

In gemischten Modellen sind einige Klassen offen, andere geschlossen

Seien

- die Klassen $1, \ldots, R$ geschlossen mit Population N_r
- die Klassen $R+1, \ldots, R+S$ offen mit Ankunftsrate λ_r

Idee des Lösungalgorithmus:

Analysiere ein offenes und ein geschlossenes Modell, wobei jeweils der Einfluss der anderen Klassen durch ein Vergrößerung der Bedienzeiten berücksichtigt wird!

Für $r \in \{R, \ldots, R+S\}$ sei $U_{i,r} = \lambda_r \cdot S_{i,r}$

die Auslastung durch die offene Klasse!

Sei $U_{i,open} = \sum_{r=R+1}^{R+S} U_{i,r}$,

die Auslastung von Station i durch offene Klassen

Für die geschlossenen Klassen steht damit an Station i nur noch $1 - U_{i,open}$ der Bedienkapazität zur Verfügung

$\Rightarrow$

ihre Bedienzeit verlängert sich um den Faktor $(1 - U_{i,open})^{-1}$

Sei $S^e_{i,r} = \frac{S_{i,r}}{1-U_{i,open}}$ für $r \in \{1, \ldots, R\}$

Algorithmus für gemischte Netze

1. Bestimme $U_{i,open}$ und $S^e_{i,r}$;

2. Löse das geschlossene Netz für die Klassen $1, \ldots, R$ mit Bedienzeiten $S^e_{i,r}$
   $\Rightarrow$
   $R_{i,r}(\underline{N}), n_{i,r}(\underline{N})$ und $X_{0,r}(\underline{N})$
   für $r = 1, \ldots, R$ und $i = 1, \ldots, K$

3. Sei $n_{i,closed} = \sum_{r=1}^{R} n_{i,r}(\underline{N})$;

4. Berechne für $r = R+1, \ldots, R+S$:
   $R'_{i,r} = \frac{D_{i,r} \cdot (1+n_{i,closed})}{1-U_{i,open}}$ und $n_{i,r} = \lambda_r \cdot R'_{i,r}$

Bemerkungen:

- Der Aufwand wird durch den Aufwand der Lösung des geschlossenen Subnetzes bestimmt.
- Das Netz ist stabil, falls das offene Netz stabil ist, unabhängig von der Population und den Bedienzeiten der geschlossenen Klassen

3.3.3 Lastabhängige Bediener

Bisher nur behandelbar:

- Verzögerungsstationen
- Stationen mit einem Bediener

aber keine Stationen mit lastabhängigen Bedienraten

Solche Stationen sind aber wichtig, da sie

- Stationen mit mehreren Bedienern beschreiben
- in vielen realen Systemen auftreten

▷ wachsende Bedienzeiten im Ethernet bei steigender Last (durch Kollisionen)

▷ fallende Bedienzeiten bei Platten mit steigender Last (durch Scheduling)

Lastabhängige Stationen sind für Analyse mittels

- Dekomposition und Aggregierung (siehe 3.8.1) notwendig

Problem bei der Behandlung von lastabhängigen Raten in MVA:

- mittlere Populationen an Stationen reichen nicht aus
- Verteilung der Auftragspopulation muss berechnet werden

Dies erhöht den Aufwand und

kann zu numerischen Problemen führen

Definieren wir zur Behandlung lastabhängiger Stationen

$P_i(\underline{n}|\underline{N})$:= Wahrscheinlichkeit, dass sich

bei Population $\underline{N}$ im Netz

$\underline{n}$ Aufträge an Station i befinden,

Es gilt (hier ohne Beweis)

$$P_i(\underline{n}|\underline{N}) = X_{i,r}(\underline{N}) \cdot S_{i,r}(\underline{N}) \cdot P_i(\underline{n} - \underline{1}_r|\underline{N} - \underline{1}_r)$$

mit

$$P_i(\underline{0}|\underline{0}) = 1.0 \text{ und } P_i(\underline{0}|\underline{N}) = 1 - \sum_{\underline{0}<\underline{n}\leq\underline{N}} P_i(\underline{n}|\underline{N})$$

(Berechnung numerisch problematisch)

und es gilt

$$R_{i,r}(\underline{N}) = \sum_{\underline{n}\leq\underline{N}} S_{i,r}(\underline{n}) \cdot (\textstyle\sum_{s=1}^{R} n_s) \cdot P_i(\underline{n} - \underline{1}_r|\underline{N} - \underline{1}_r)$$

sehr komplex, auch wenn Vereinfachungen möglich sind, falls Raten bestimmten Bedingungen gehorchen

(was notwendig ist, wie sich später zeigen wird)

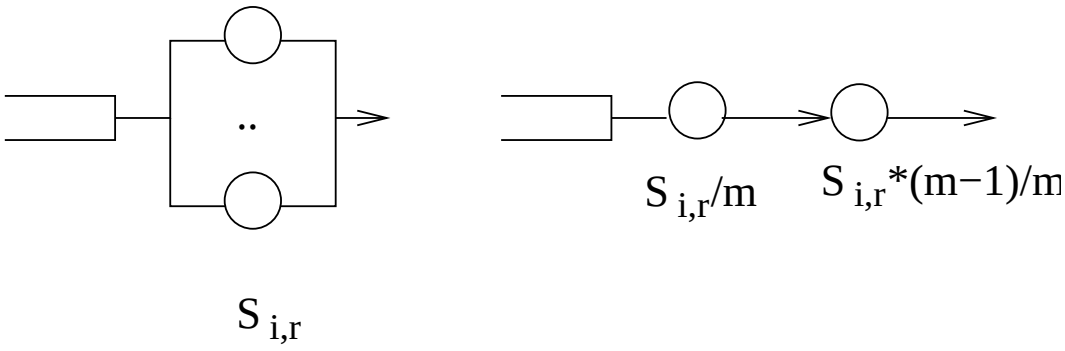
Es gibt noch andere Analysealgorithmen, die (etwas) besser für lastabhängige Stationen geeignet sind, aber Aufwand bleibt beträchtlich!

Approximatives Vorgehen nach Menasce

für Systeme mit m Bedienerstationen

Ersetze m Bediener mit Bedienzeiten $S_{i,r}$ durch

- Einzelbediener mit Bedienzeit $S_{i,r}/m$ und
- Verzögerungsstation mit Bedienzeit $S_{i,r} \cdot (m-1)/m$



Verhalten in Extremsituationen identisch zur Station mit m Bedienern

- Bei geringer Last durchläuft Auftrag Station in $S_{i,r}$ Zeiteinheiten
- Bei hoher Last ist der einzelne Bediener immer aktiv und der Durchsatz beträgt $m/S_{i,r}$

Größte Fehler im Bereich mittlerer Last,
aber auch in diesem Bereich oft noch tolerable Fehler
($< 5\%$)

3.4 Stochastische Interpretation

Ansatz der operationalen Analyse einfach und intuitiv
aber auch korrekt?

Unter stochastischer Betrachtung:

1. sind Routingwahrscheinlichkeiten im Netz definiert
2. wird aus der Untersuchung des Systems in einem endlichen Intervall die stationäre Analyse des stochastischen Prozesses
3. gehorchen Ankunfts- und Bedienzeiten bestimmten Verteilungen

zu 1.

Annahmen:

Mehrklassennetze ohne Klassenwechsel
$\Rightarrow$ jede Klasse kann separat untersucht werden

Aufträge der Klasse besuchen alle Stationen im Netz
(Stationen, die nicht besucht werden,
können bei Berechnung weggelassen werden)

Routingwahrscheinlichkeiten für offene Netze

$p_{i,j}$ Wahrscheinlichkeit, dass Auftrag,
der Station i verlässt, zu Station j geht

$p_{i,0}$ Wahrscheinlichkeit, dass Auftrag
nach Verlassen von Station i das System verlässt

$p_{0,i}$ Wahrscheinlichkeit, dass Auftrag
System in Station i betritt

Es gilt $\sum_{j=0}^{K} p_{i,j} = 1$ für alle i

Die Besuchshäufigkeiten V_i sind die eindeutige Lösung
des linearen Gleichungssystems

$V_i = p_{0,i} + \sum_{j=1}^{K} p_{j,i} \cdot V_j$

Routingwahrscheinlichkeiten für geschlossene Netze

Routingwahrscheinlichkeiten $p_{i,0}$ und $p_{0,i}$ fallen weg

Annahme: jede Station ist
von jeder anderen im Netz erreichbar

d.h. für Stationen i und j existieren
Stationen $k_1 \ldots k_n$, so dass

$p_{i,k_1} > 0$, $p_{k_i,k_{i+1}} > 0$ und $p_{k_n,j} > 0$

Wahrscheinlichkeiten $p_{i,j}$ in
Routingmatrix P zusammenfassen

Routingmatrix P ist eine

- irreduzibel
- stochastische
- $K \times K$ Matrix
- mit Rang $K - 1$

für den Vektor der Besuchshäufigkeiten

$\underline{V} = (V_1, \ldots, V_K)$ gilt $\underline{V} = \underline{V}P$

$\Rightarrow$ Lösung existiert eindeutig bis auf eine Konstante

Lösung eines linearen Gleichungssystems der Dimension K zur Bestimmung der "*relativen*" Besuchshäufigkeiten

zu 2.

stationäre Analyse betrachtet System über

unendlichen Zeithorizont

Flussgleichgewicht entspricht Stationarität

Voraussetzung in beiden Fällen

mittlere Bedienzeit kleiner als

mittlere Zwischenankunftszeit

dies gilt immer in geschlossenen Netzen!

zu 3.

Streng genommen gelten Regeln

der operationalen Analyse nur

bei exponentiellen Ankunfts- und Bedienzeiten

(auch dort nur mit einer Einschränkung siehe unten)

Aber wie in *BCMP 1975* (und anderen Arbeiten) nachgewiesen, gibt es zahlreiche Stationstypen, die sich so verhalten, als ob sie exponentielle Bedienzeitverteilungen hätten

$\Rightarrow$ Klasse der **Produktformnetze**

d.h. Analyse des Netzes mit exponentiellen Bedienzeitverteilungen (oder nach Gesetzen der operationalen Analyse) liefert korrekte Resultate

Netzklasse ist wie folgt charakterisiert:

- Im Netze können sich Aufträge verschiedener Klassen befinden
  - ▷ Klassenwechsel sind erlaubt,
    Klassen zwischen denen Wechselmöglichkeiten
    bestehen fasst man zu Ketten zusammen,
  - ▷ Netze können offen, geschlossen oder für einige
    Ketten offen für andere geschlossen sein,
    für geschlossene Ketten muss die Routingmatrix
    irreduzibel sein

- Die Zwischenankunftszeiten offener Ketten sind negativ exponentiell verteilt (die Ankunftsrate darf von der Anzahl der Aufträge im Netz bzgl. der ankommenden Kette oder aller Ketten abhängen)
- Stationstypen sind vom einem der folgenden Typen
  - ▷ Verzögerungsstation mit beliebiger kettenabhängiger Bedienzeitverteilung
  - ▷ *Last Come First Served Preemptive Resume*, mit beliebiger kettenabhängiger Bedienzeitverteilung
  - ▷ *Processor Sharing* mit beliebiger kettenabhängiger Bedienzeitverteilung
  - ▷ *First Come First Served* oder *Random* mit negativ exponentieller Bedienzeitverteilung, der Mittelwert für alle Klassen ist identisch
- Bediengeschwindigkeiten an einer Station können von der Population an der Station abhängen, müssen aber die folgende Bedingung erfüllen

$$\frac{S_{i,r}(\underline{n})}{S_{i,s}(\underline{n})} = \frac{S_{i,r}(\underline{n}-\underline{1}_s)}{S_{i,s}(\underline{n}-\underline{1}_r)}$$

(Geschwindigkeit erlaubt einen freien Parameter pro Population $\underline{n}$ und nicht R freie Parameter wie im allgemeinen Fall!!)

Mögliche Form der lastabhängigen Bediengeschwindigkeiten

Lastabhängigkeit in Abhängigkeit von Gesamtpopulation an der Station

also

$$S_{i,r}(\underline{n}) = (\mu_{i,r} \cdot n_r \cdot C_i(\sum_{s=1}^{R} n_s))^{-1}$$

$C_i(j) :=$ Geschwindigkeit des Bedieners mit j Aufträgen

Geschwindigkeiten erfüllen Bedingungen für Produktform

Mittelwertanalyse ist dann etwas einfacher:

Es gilt

$$P_i(j|\underline{N}) = \frac{1}{C_i(j)} \cdot \sum_{r=1}^{R} \frac{X_{i,r}(\underline{N})}{\mu_{i,r}} \cdot P_i(k-1|\underline{N} - \underline{1}_r)$$

mit $P_i(j|\underline{N})$ W. für Population j an Station i bei Population $\underline{N}$ im Netz

damit gilt auch

$$R_{i,r}(\underline{N}) = \sum_{j=1}^{n} j \cdot P_i(j-1|\underline{N} - \underline{1}_r) \cdot (\mu_{i,r} \cdot C_i(j))^{-1}$$

3.5 Anwendungsbeispiele

Produktformwarteschlangennetze werden breit eingesetzt für die Modellierung von

- Rechensystemen
- Rechnernetzen
- Kommunikationssystemen
- Fehlertoleranten Systemen
- Fertigungssystemen
- Verkehrssystemen

Wir betrachten Anwendungen aus ersten zwei Gebieten

Analytische Lösbarkeit des Modells erfordert Abstraktion

$\Rightarrow$ Resultate mit gewissen Modellierungsfehlern

Allgemeine Beobachtung

- Durchsätze relativ robust gegenüber Abstraktion
- Verweilzeiten deutlich sensitiver

aber praktischer Einsatz belegt für viele Systeme sind Leistungsgrößen mit Genauigkeit von 10%-30% ermittelbar

Oft ausreichend insbesondere

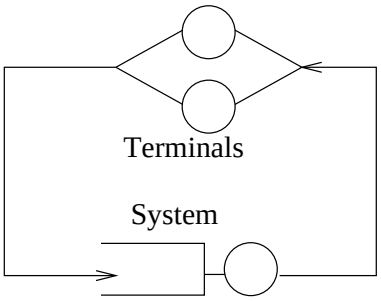
- zur Bestimmung von Flaschenhälsen
- zum Vergleich von Konfigurationen

detailliertere Ergebnisse u.U. durch Simulation

3.5.1 Computersysteme

Primäres Anwendungsgebiet für Produktformnetze

Erstes Modell einer IBM 7094 (1965)



Ziel: Antwortzeitberechnung

Systemeigenschaften:

- ein aktiver Benutzer

  deshalb keine Unterscheidung von CPU und Platten

- Wechsel der Benutzer nach Zeitscheibenverfahren

Systemparameter legen

Zeitscheibenlänge + Overhead zum Prozesswechsel fest

Messungen zur Ermittlung

- der mittleren Denkzeit
- der mittleren Anzahl Zeitscheiben

Im Modell Vereinfachung durch exponentielle Zeiten

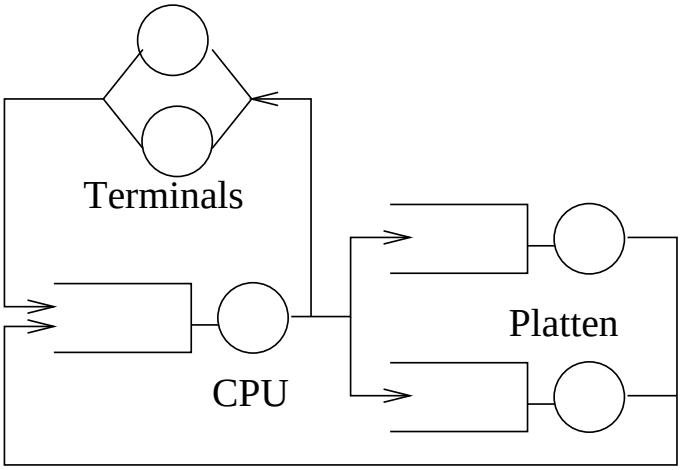
Ergebnisse sehr gut

  Fehler von 5% bei Ermittlung der Antwortzeit

Moderne Systeme erlauben Mehrprogrammbetrieb

$\Rightarrow$ System nicht mehr als einzelne Station darstellbar

Typisches Modell *Central Server*



Modell existiert in verschiedenen Varianten und wurde erfolgreich eingesetzt für Analyse von Großrechner insbesondere IBM 360 Systemen

Erweiterungen:

- zusätzliche Kanäle für Platten
  Kanäle als zusätzliche Stationen
  (Problem: Aufträge können Platte und Kanal nicht simultan belegen)
- obere Schranke für Anzahl Jobs im System
  Zahl der aktiven Jobs an CPU + Platten beschränken
  (Problem: warten auf Zugang kann nicht modelliert werden)

Mehrklassennetze für komplexere Systeme

i.a. immer noch *central server* Struktur aber

- unterschiedliche Lastklassen
- Batch Jobs oft als offene Klasse
- Dialog Jobs als geschlossene Klasse
  mehrere Klassen falls unterschiedliche Anforderungen,
  Klassenwechsel falls Anforderungen sich ändern

Modellierung der Ressourcen

- CPU als Station mit PS Disziplin
- Terminals als Verzögerungsstationen
- Platten, Speicher, Bus als FCFS Station

Zur Festlegung der Lastparameter:

- Messung und Einteilung der Messwerte in Klassen
- nur Mittelwerte sind relevant (Produktformnetze sind
  unempfindlich gegenüber Verteilungen)

Probleme:

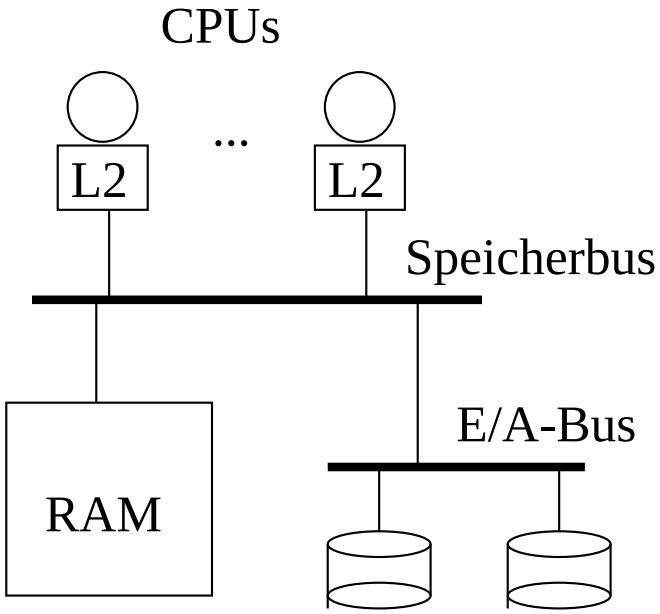
- FCFS Stationen setzen exponentielle Bedienzeiten
  voraus (oft nicht realistisch bei Platten)
- verschiedene Klassen müssen gleiche Bedienan-
  forderungen an FCFS Stationen haben
  (schränkt die Modellierung mit mehreren Klassen ein)

3.5.2 Multiprozessorsysteme

Wir betrachten SMPs (*symmetric multiprocessors*)
unter grobkörniger Last
(d.h. Last besteht aus unabhängigen Prozessen)

Typische Struktur

- mehrere (i.a. 4-8) Standardprozessoren
  mit lokalen Caches
- gemeinsamer globaler Speicher
- gemeinsame Plattenperipherie
- Betriebssystem verdeckt parallele Prozessoren
  Beispiele Sun SS1000, HP 9000, SNI RM 600



Speicherbus kann

- verbindungsorientiert
  (Bus bleibt belegt bis Anfrage abgearbeitet)
- paketorientiert
  (Anfragen werden asynchron geschickt)

arbeiten

Weiteres Problem Konsistenz der Caches

Strategien

- *write through*
  (jedes Schreiben direkt in den Speicher)
- *write back*
  (nur in den Speicher schreiben, wenn Daten aus dem Cache entfernt werden)

Konsistenz zwischen Caches durch Abhören des Speicherbusses (*snooping protocol*)

Leistungsfähigkeit von SMPs hängt stark vom Programmierstil ab

- Datenlokalität auf einzelnen Prozessoren ist wichtig
- bisher kaum Unterstützung in verfügbaren Systemen

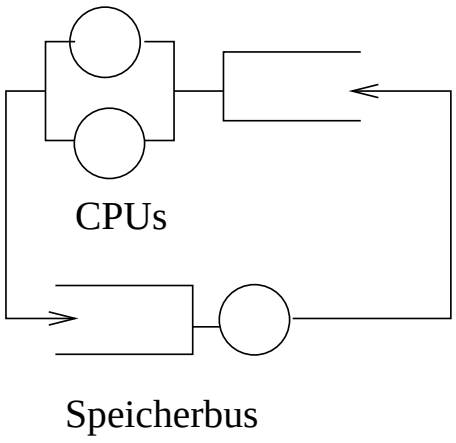
Einfachstes Modell M/M/m System

(Bediener sind Prozessoren, Aufträge in Warteschlange warten auf CPU Zuteilung)

Dieses grobe Modell

- vernachlässigt Mehrprogrammfähigkeit der Prozessoren
- hat mit Auslastung wachsende Zahl von wartenden Prozessen
- beinhaltet weder Ein-/Ausgabe noch Speicherzugriffe
- kann aber zur ersten Abschätzung des maximal erreichbaren Durchsatzes dienen

Nächstes Modell mit Speicherbus



beide Stationen FCFS Bediener mit exponentieller Bedienzeitverteilungen

Einfache Erweiterung für mehrere Speicherbusse

Modell ist pessimistisch da

- Speicherzugriffszeiten exponentiell und nicht deterministisch
- Caches nicht beachtet werden

Modell ist optimistisch da

- Plattenzugriffe vernachlässigt werden
- Prozesse von beliebiger CPU bedient werden
- Speicherlocks unbeachtet bleiben

Trotzdem kann Modell erste Ergebnisse liefern

Übliche Leistungsmaße bei Multiprozessoren

Power

= durchschnittliche Anzahl aktiver Prozessoren

und

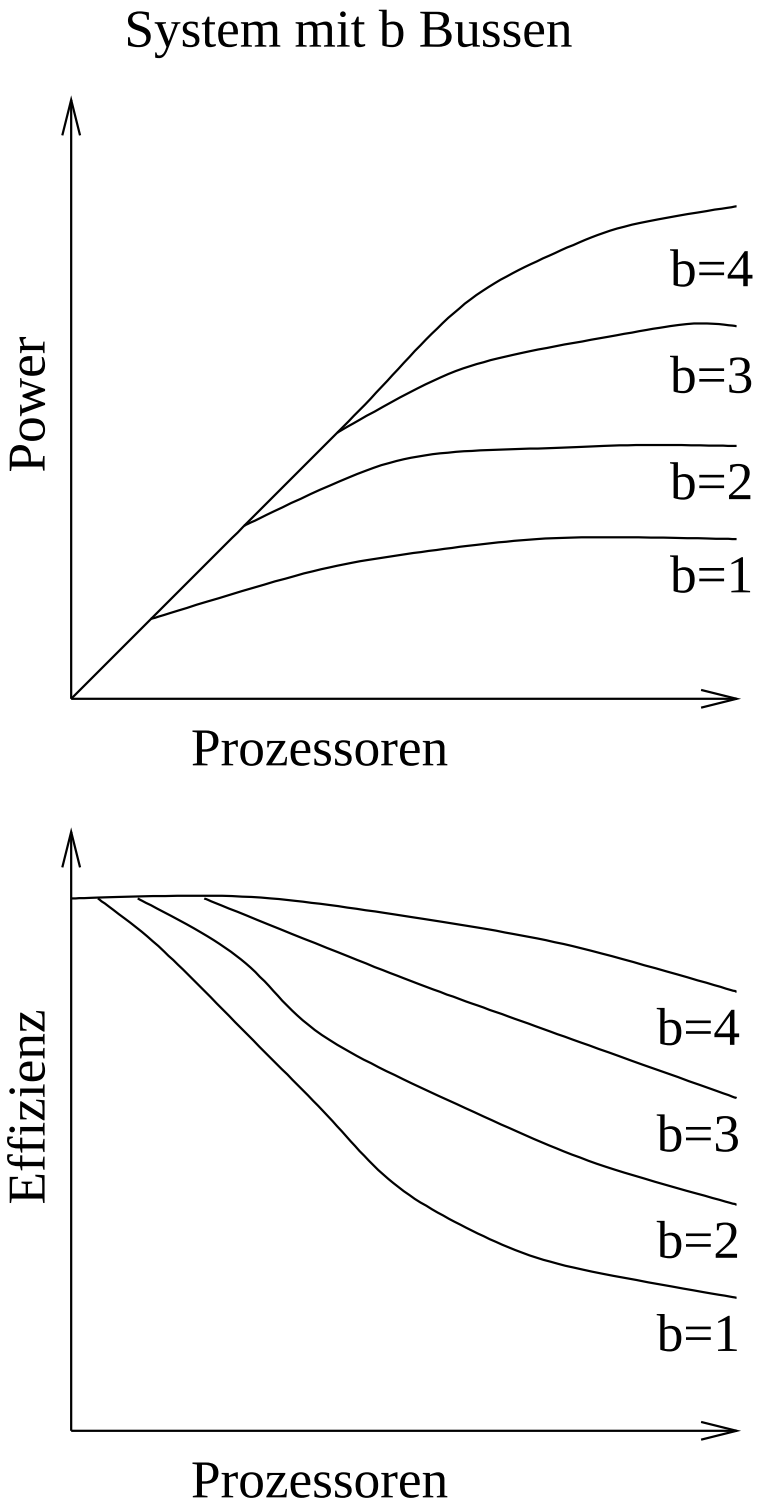
Effizienz =

Anzahl aktiver Prozessoren / Anzahl Prozessoren

insbesondere interessant bei einem Prozess pro Prozessor

(Effizienz parallelisierter Programme im Vergleich zur sequentiellen Implementierung)

Typischer Verlauf der Maße bei steigender Prozessorzahl

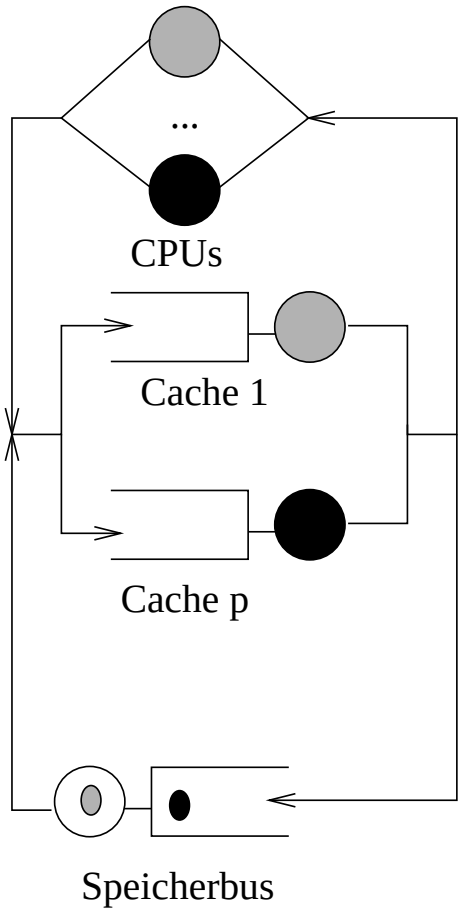


Nächster Erweiterungsschritt:

- Mehrklassennetz zur Beachtung
  - ▷ unterschiedlicher Prozessorzeiten
  - ▷ unterschiedlicher Speicherzugriffszeiten
- Beachtung der Effekte durch Caches

Produktformwarteschlangennetze erlauben nur abstrakte Modellierung des Cache-Protokolls detaillierte Modelle erfordern Simulation

Basismodell



Aufträge bei der Strategie *write-through*

- Leseaufträge, die aus dem Cache befriedigt werden können
- Leseaufträge, die einen Speicherzugriff erfordern
- Schreibaufträge, die immer auch in den Speicher schreiben

Aufträge bei der Strategie *write-back*

- Leseaufträge, die aus dem Cache befriedigt werden können
- Leseaufträge, die einen Speicherzugriff erfordern
  - ▷ ohne Verdrängung einer geschriebenen Seite
  - ▷ mit Verdrängung einer zu geschriebenen Seite
- Schreibaufträge, die in den Cache schreiben
- Schreibaufträge, die in den Speicher schreiben

Analyse des Modells liefert

- Antwortzeit: mittlere Zeit zwischen zwei Besuchen an der CPU (bzw. 1/CPU Durchsatz - Denkzeit)
- Power: mittlere Anzahl belegter CPUs
- Effizienz:
  mittlere Anzahl belegter CPUs / Anzahl CPUs
- Auslastung Speicher

Modell als Mehrklassennetz

- eine Klasse pro Prozessor

Verzweigungswahrscheinlichkeiten auf Basis
relativer Besuchshäufigkeiten und Systemparameter

Sei

- p_r Anteil Leseoperationen,
- $p_w = 1 - p_r$ Anteil Schreiboperationen,
- p_{miss} W. eines $Cache\text{-}misses$

Basis 1 Besuch an der CPU

bei $\underline{write\text{-}through}$

Besuche Cache $p_r + p_w = 1$

Besuche Speicher $p_r \cdot p_{miss} + p_w$

bei $\underline{write\text{-}back}$

Besuche Cache $p_r \cdot (1 + p_{miss} \cdot p_w) + p_w$

Besuche Speicher $p_r \cdot p_{miss} + p_w \cdot p_{miss} + p_r \cdot p_w \cdot p_{miss}$

Modell ist Basis für Vergleich von
$write\text{-}through$ und $write\text{-}back$

Mögliche Erweiterungen:

- Plattenperipherie
- Paketorientierter Speicherbus

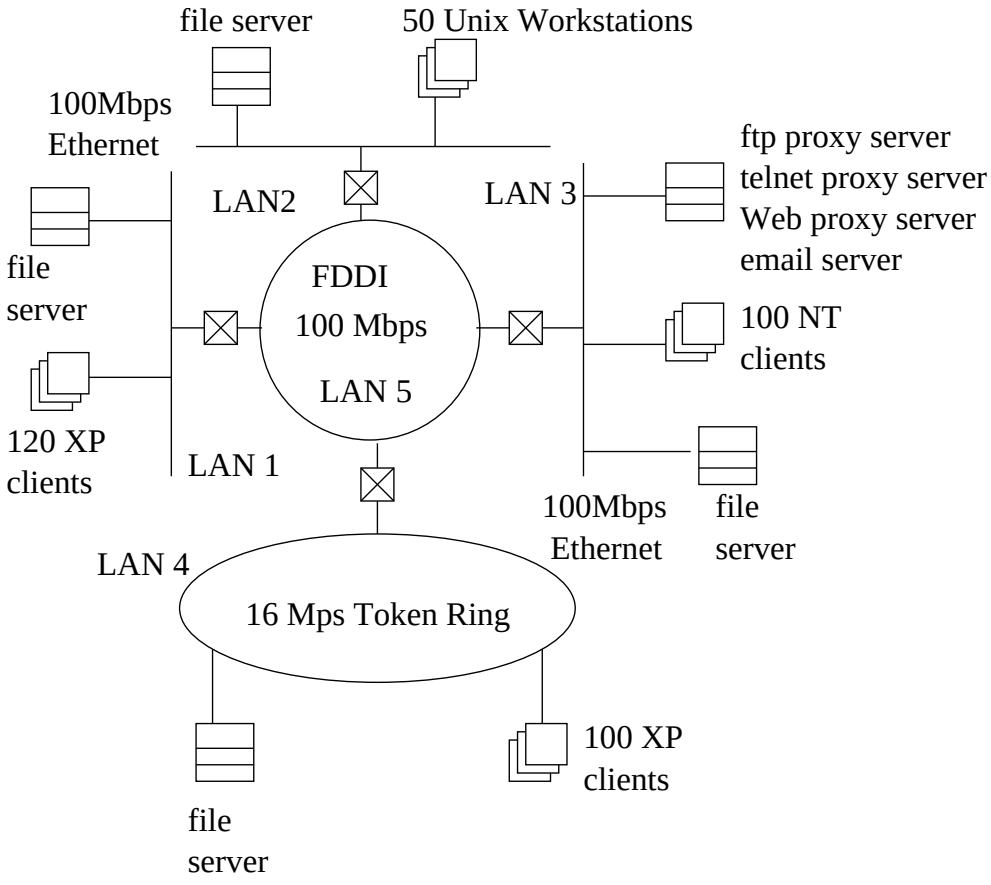
3.5.3 Client-Server Systeme

Komponenten aus vorherigen Kapiteln bilden Basis für Modellierung von Client-Server Systemen

Schritte bei Modellerstellung:

- Festlegung auf offenes, geschlossenes oder gemischtes Modell
- Festlegung der Komponenten und ihrer Darstellung
  z.B. Menge von Clients als Verzögerungsstation oder
  jeder Client als einzelne Station oder
  als Teilmodell mit CPU, Speicher und Platte
  dito für Server
  Darstellung der Netzkomponenten als (lastabhängige) Stationen
- Beschreibung der einzelnen Komponenten
- Beschreibung der Last
  Aufteilung in Klassen (mehr dazu in Kapitel 4)
- Berechnung des Bedienbedarfs an den Komponenten (mit den bereits eingeführten Formeln)
- Kalibrierung und Validierung des Modells
- Ermittlung von Resultaten

Typisches Beispiel:



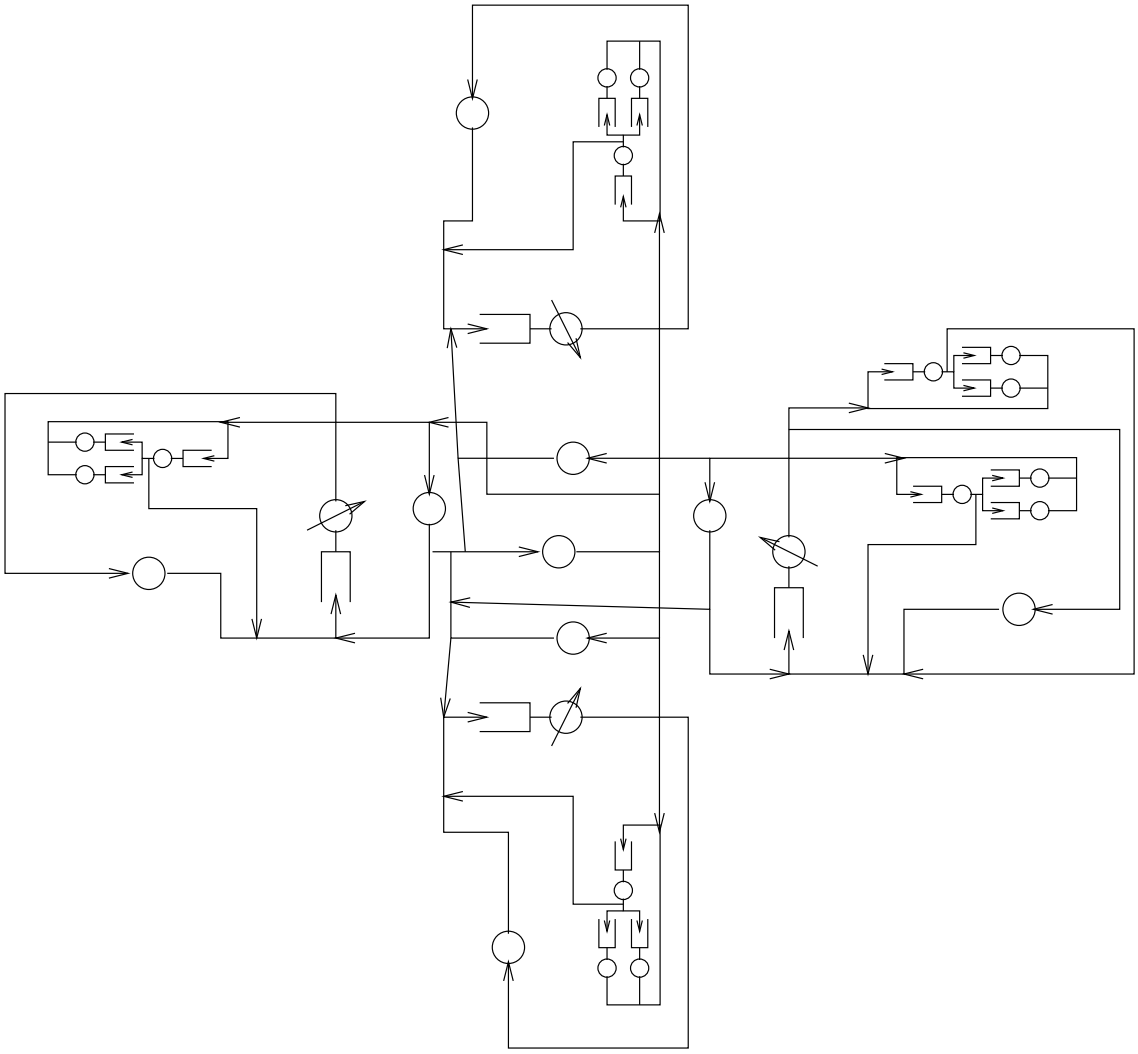
Entscheidungen die zu treffen sind:

- Ziele der Analyse und notwendige Resultatgrößen festlegen
- Entscheidung für offenes, geschlossenes oder gemischtes Modell
- Lastklassen festlegen und beschreiben
- Repräsentation der einzelnen Klassen
- Anzahl Aufträge oder Ankunftsrate pro Klasse

Entscheidung hier:

- Ermittlung von Durchsätzen und Antwortzeiten für die einzelnen Klassen
- Geschlossenes Mehrklassenmodell
  - ▷ jede Klasse ist charakterisiert durch Tripel (Client Gruppe, Anwendung, Server)
  - ▷ 4 Klassen von Clients CL1,. . . ,CL4
  - ▷ 2 Anwendungen: Zugriff auf File-Server (FS) und Web-basiertes Training (TR)
  - ▷ 4 File Server (FS1,. . . ,FS4) plus Web-Server (WS)
  - ▷ Service FS greift auf den lokalen File-Server zu, also (CLx, FS, FSx) mit x=1,. . . ,4
  - ▷ Service TR greift auf den Web-Server zu, also (CLx, TR, WS)
- Beschreibung der Ressourcen
  - ▷ Clients, FDDI Ring und Router werden als Verzögerungsstationen modelliert
  - ▷ FS und TR werden als Central Server Systeme mit jeweils zwei Platten und einer CPU modelliert
  - ▷ LAN 1,. . . ,4 als lastabhängige Stationen modelliert

Damit entsteht folgendes Warteschlangennetz



Graphische Darstellung ohne weitere Erklärung
- unübersichtlich

Funktionalität nur durch detailliertere Spezifikation
- verständlich

insbesondere Darstellung der unterschiedlichen Klassen
- notwendig

3.5.4 Web Server

Web ist auch ein Client Server System

Besonderheiten

- jeder Client greift auf unterschiedliche Server zu
- Größe des Webs mehrere Millionen Clients mehrere Tausend Server (Verhältnis ca. 10000 zu 1)
- pro Server viele Clients, von denen jeder einzeln selten zugreift
- hohe Varianz in den Dokumentengrößen
- Zugriffsmuster gekennzeichnet durch *burstiness* und *heavy tails*

Spezifikation der Last nur durch Mittelwerte führt zu starken Verfälschungen bei den Ergebnissen

$\Rightarrow$

burstiness und *heavy tails* sind explizit zu berücksichtigen

Modellierung von *bursts*

Bursts sorgen dafür, dass hintereinander
Anforderungen in kurzen Zeitabständen ankommen
und dann längere Pausen entstehen

Dies ist bei der Analyse zu berücksichtigen

Auf Basis der operationalen Analyse kann dies nur
durch Justierung des Mittelwerts geschehen

Definition eines Faktors zur Charakterisierung

- L Anzahl Anfragen
- T Zeitraum in dem die Anfragen eintreffen
- $\lambda = L/T$ Ankunftsrate
- Unterteile T in n Teilintervalle der Länge T/n
  $Arr(k)$ Anzahl Anfragen im Intervall k
  $\lambda_k = \frac{Arr(k)}{T/n} = \frac{n \cdot Arr(k)}{T}$ Ankunftsrate Intervall k
- Arr^+ Anzahl Anfragen, die in Intervallen ankommen,
  in denen Ankunftsrate höher als der Durchschnitt ist
  $Arr^+ = \sum_{\lambda_k > \lambda} Arr(k)$
- Arr^- Anzahl Anfragen, die in Intervallen ankommen,
  in denen Ankunftsrate kleiner gleich Durchschnitt ist
  $Arr^- = \sum_{\lambda_k \leq \lambda} Arr(k)$

Es gilt $L = Arr^+ + Arr^-$

Unter Annahmen der operationalen Analyse gilt für

Strom, der keine *bursts* enthält

$Arr(k) = L/n$ und damit $\lambda_k = \lambda$ für alle k

Definition *burstiness* Parameter

$b =$ (Anzahl Intervalle mit $\lambda_k > \lambda$)/ n

Für einen Strom ohne *bursts* gilt $b = 0$

Ankunftsrate während eines *bursts*

$\lambda^+ = \frac{Arr^+}{b \cdot T}$

Relation der Ankunftsraten

$a = \frac{\lambda^+}{\lambda} = \frac{Arr^+}{b \cdot L}$

Berücksichtigung von *bursts*

in den mittleren Bedienanforderungen

Beobachtung:

Durchsatz fällt mit wachsendem *burst*-Faktor

Maximaler Durchsatz entspricht der inversen Bedienzeit

am Flaschenhals

Neujustierung der Bedienzeit am Flaschenhals als

$$D = D_f + a \cdot b$$

wobei D_f Bedienzeit ohne Einfluss der *bursts*

Bestimmung von Faktor a aus Messungen
in folgenden Schritten

- Messung Web Server im Intervall $[0, T)$
  liefert L Anfragen

▷ Unterteile Intervall in zwei gleichgroße Teilintervalle
mit L_1 und L_2 Anfragen

▷ Bestimme für jedes Intervall die Auslastung aller
Ressourcen, den Durchsatz und den *burst* Faktor b_i

▷ Sei U_i die Auslastung am Flaschenhals im Intervall i
und X_i der Durchsatz, dann gilt

$$U_i/X_i = D_f + a \cdot b_i$$

und damit auch

$$a = \frac{U_1/X_1 - U_2/X_2}{b_1 - b_2}$$

Berücksichtigung von *heavy tails*

durch Verwendung von mehreren Klassen

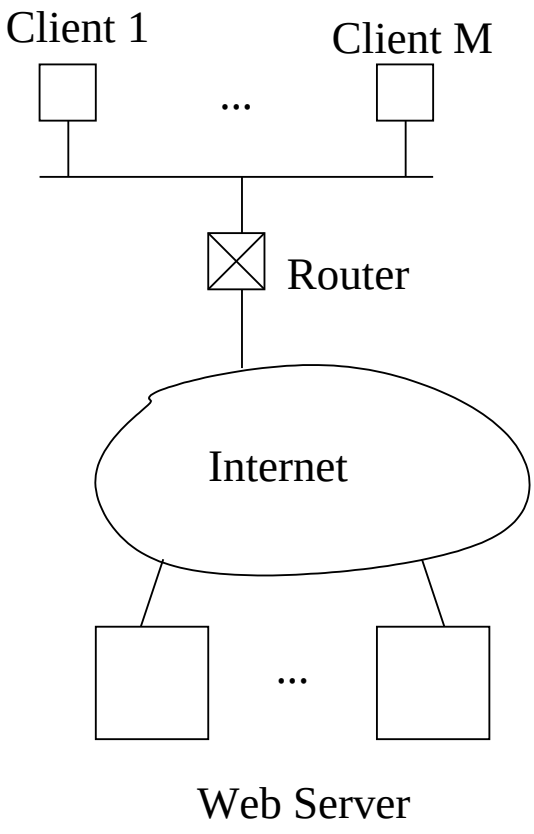
Modelle auf der Client Seite

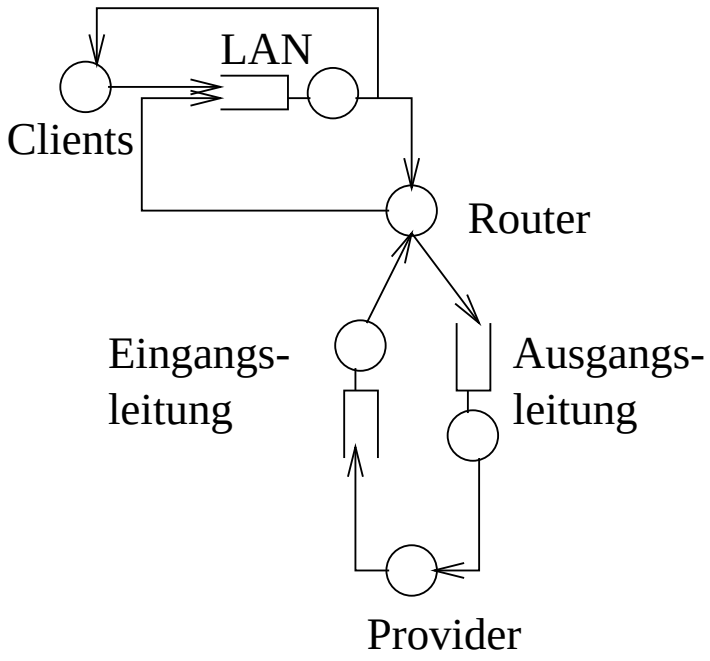
Kapazitätsplanungsfragen auf Client-Seite

1. Bandbreite der Verbindung zum Service Provider
2. Bandbreite des internen LANs
3. Einsatz von Proxy Cache Servern

Erstes Modell ohne Proxy Cache

- M Client Workstations
- mittels LAN verbunden
- über Router mit einem Service Provider verbunden





Systemparameter

- *LANBandw* Bandbreite des LANs
- *MaxPDU* auf Netzwerkebene
- *FrameOv* Overhead durch LAN link-layer
- *RouterLat* Latenzzeit des Routers
- *LinkBandw* Bandbreite der Verbindung zum Provider
- *InternetDel* mittlere Verzögerungszeit einer Internet-Anfrage
- *InternetRate* Datenraten des Internetanschlusses
- *BrowserRate* Rate mit der vom Browser neue Dokumente angefordert werden
- *NumberCl* Anzahl Clients
- *PercAct* Prozentsatz aktiver Clients (d.h. Clients, die auf das Internet zugreifen)
- *AvgReqSize* durchschnittliche Größe einer HTTP Anfrage
- *DocSize* durchschnittliche Größe der angeforderten Dokumente

  i.a. Zerlegung in Klassen 1,. . . ,R:

  DocSize=$\sum_{r=1}^{R} DocSize_r \cdot DocPerc_r$

Bestimmung des Bedienbedarfs

$D_{cl} = 1/BrowserRate$ (Clients)

$D_{LAN} = NetTime(AvgReqSize + DocSize)$

wobei

$NetTime(m) = \frac{8 \cdot (m + Overh(m))}{LANBandw}$

$$Overh(m) = TcpOv + NDatag(m) \cdot (IpOv + FrameOv) = 20 + NDatag(m) \cdot (20 + FrameOv)$$

$NDatag(m) = \left\lceil \frac{m + TcpOv}{MaxPDU - IpOv} \right\rceil$

$D_{Router} = (NDatag \cdot DocSize + 6) \cdot RouterLat$

6 ensteht durch zus. TCP Nachrichten

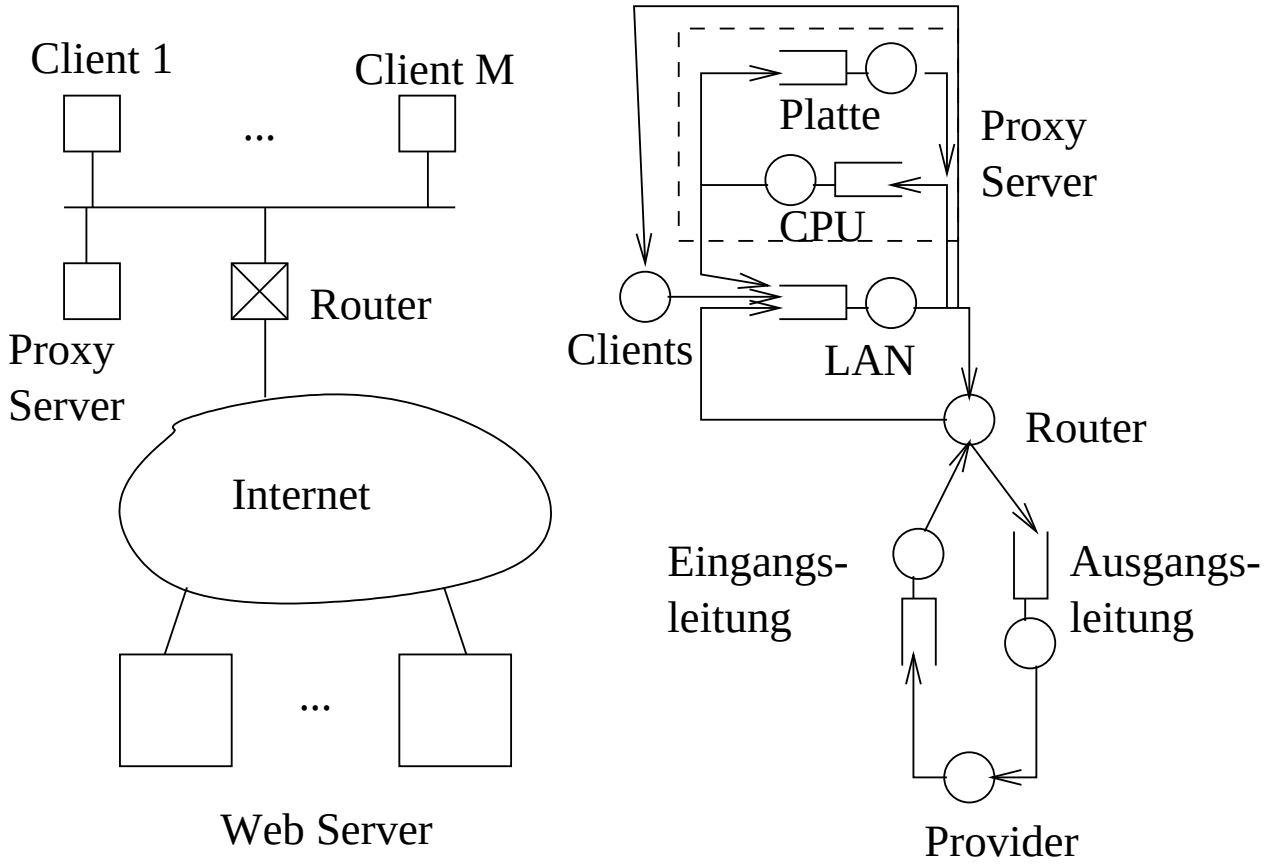
$D_{AusL} = \frac{AvgReqSize + 5 \cdot (TcpOv + IpOv)}{LinkBandw}$

5 entsteht durch zus. TCP Nachrichten

$D_{Prov} = 1.5 \cdot InternetDel + DocSize/InternetRate$

$D_{EinL} = \frac{DocSize + LinkOv}{LinkBandw}$

Modell mit Proxy Cache



Systemverhalten

- für jede Anfrage wird zuerst im Cache des Proxy Servers gesucht
- bei Erfolg Übertragung von dort zum Client
- bei Misserfolg agiert Proxy Server als Client
  - ▷ holt Dokument
  - ▷ speichert Dokument
  - ▷ überträgt Dokument zum Client

Zusätzliche Parameter

- p_{hit} Trefferwahrscheinlichkeit im Proxy Cache
- $HitCPUTi$ CPU Zeitbedarf bei einem Treffer
- $MissCPUTi$ CPU Zeit bei Misserfolg
- $DiskTi$ Zeit für Plattenzugriffe
  (identisch bei Treffer oder Misserfolg,
  im ersten Fall lesen, im zweiten schreiben!)

Berechnung der Bedienzeiten

$$D^P_{LAN} = p_{hit} \cdot D_{LAN} + (1 - p_{hit}) \cdot 2 \cdot D_{LAN}$$

$$D^P_{Router} = (1 - p_{hit}) \cdot D_{Router}$$

$$D^P_{EinL} = (1 - p_{hit}) \cdot D_{EinL}$$

$$D^P_{AusL} = (1 - p_{hit}) \cdot D_{AusL}$$

$$D^P_{Prov} = (1 - p_{hit}) \cdot D_{Prov}$$

sowie

$$D^P_{CPU} = p_{hit} \cdot HitCPUTi + (1 - p_{hit}) \cdot MissCPUTi$$

$$D^P_{Platte} = DiskTi \cdot DocSize$$

Modelle auf der Server Seite

Kapazitätsplanungsfragen auf Server Seite

1. Bandbreite der Verbindung zum Internet
2. Einrichtung von Spiegelservern
3. Server auf einem Rechner oder auf Rechnernetz verteilt
4. Dokumentenverteilung über mehrere Platten
5. Maximal erreichbarer Durchsatz
6. Komprimierte Speicherung großer Dokumente
7. CGI Skripten oder Java Applets
8. Auswahl Servertyp

Modellierung von Web Servern

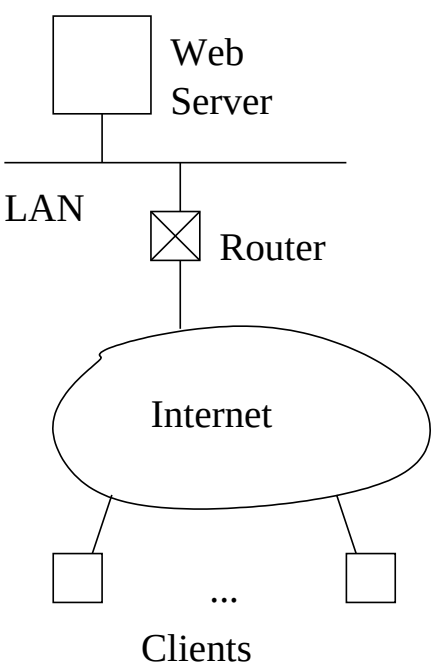
Web Server werden von zahlreichen Clients angesprochen

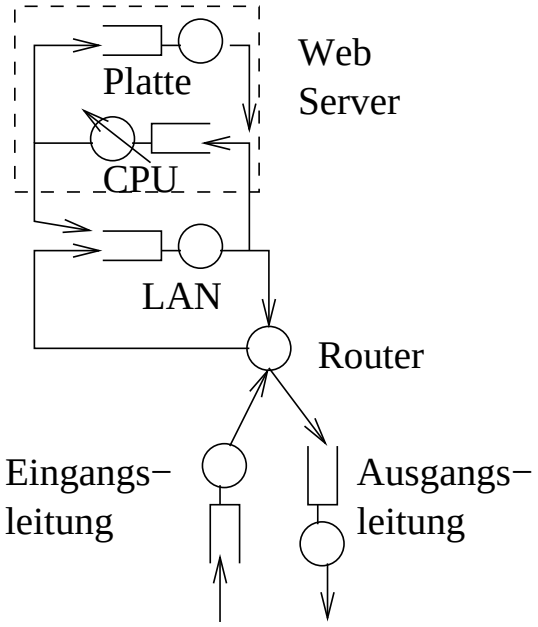
$\Rightarrow$ Modellierung als offene Mehrklassennetze

Wir unterscheiden

- Web Server auf einem Rechner realisiert
- Web Server als verteiltes System

Web Server auf einem Rechner





Belastung durch R Klassen mit

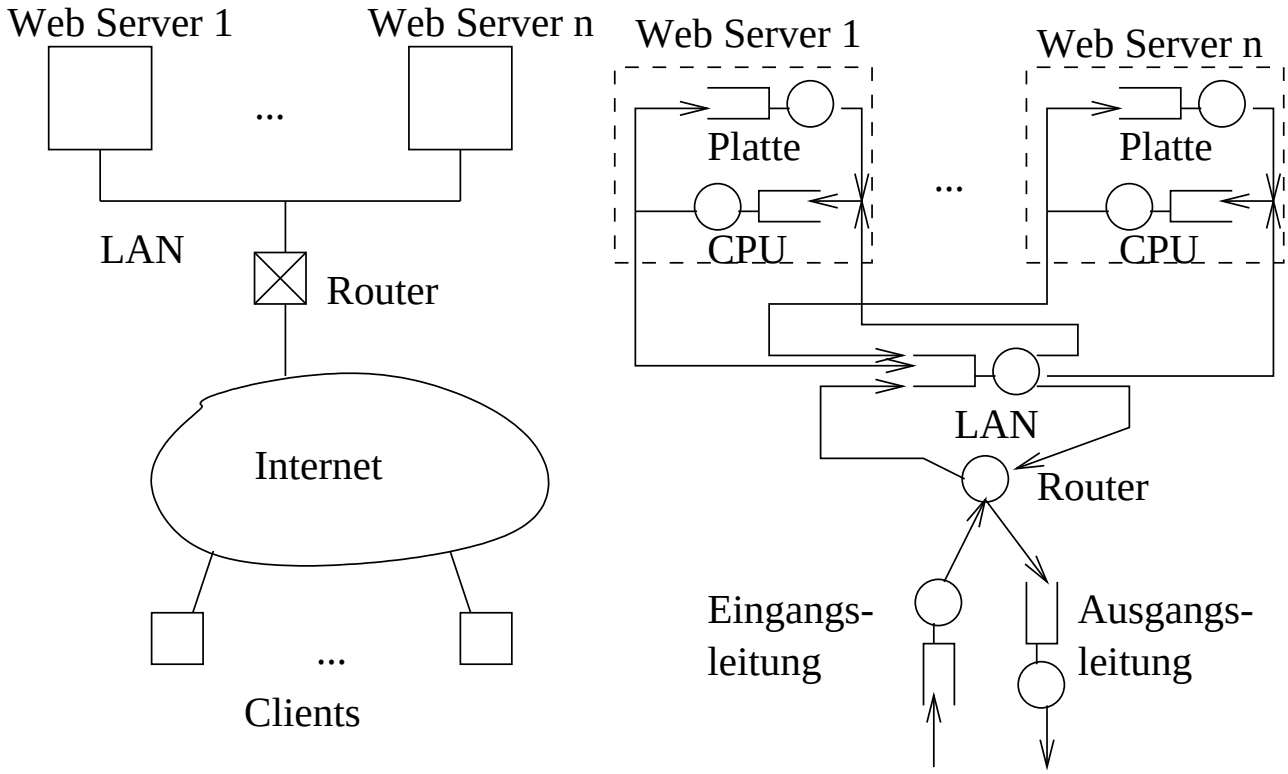
- klassenspezifischen Ankunftsraten λ_r
- klassenspezifischen Bedienbedürfnissen $D_{i,r}$
  (Bestimmung wie bei Client)

aber Bedienzeit CPU i.a. lastabhängig durch
zusätzlichen Overhead

Berücksichtigung durch

- lastabhängige Bedienzeiten oder
- $D_{CPU,r} = CPUTiReq + f(\overline{n}) \cdot CPUOv$
  mit $\overline{n}$ mittlere Population am Web Server
  $\Rightarrow$ Fixpunktansatz

Web Server als verteiltes System



Last jedes einzelnen Servers $1/n$ der ankommenden Last

Dies kann erreicht werden indem

$D^v_{CPU,r} = D_{CPU,r}/n$

und

$D^v_{Platte,r} = D_{Platte,r}/n$

gesetzt werden

(u.U. zusätzliche Berücksichtigung der Lastabhängigkeit)

Restliche Zeiten bleiben gleich

3.6 Modellierung mit Markov-Prozessen

Bisherige Modellierung basierte im wesentlichen auf der Betrachtung und Untersuchung von Mittelwerten

- einfache Modellparametrisierung
- effiziente Analyse auch großer Systeme

aber

- unzureichende Berücksichtigung von Verteilungen (Ausnahme M/G/1)
- keine Berücksichtigung von Korrelationen

Allgemeineres und analysierbares Modelle: Markov-Prozesse

$X(t)$ stochastischer Prozess mit

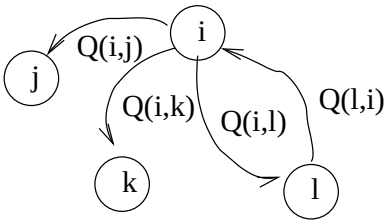
- Zustandsraum $S = \{1, \ldots, n\}$ oder $S = \mathbb{N}$
- Übergangsratenmatrix (oder auch Generatormatrix) $Q \in \mathbb{R}^{n,n}$mit
  - $Q(i,j) \geq 0$ falls $i \neq j$ und
  - $Q(i,i) = -\sum_{j \neq i} Q(i,j)$
  - Q hat Zeilensumme 0 und nicht-negative Nichtdiagonalelemente

Verhalten des Markov-Prozesses (Annahme $|Q(i,i)| > 0$):

- Prozess verweilt eine exponentiell verteilte Zeit mit Rate $|Q(i,i)|$ im Zustand i
- wechselt anschließend mit Wahrscheinlichkeit $P(i,j) = Q(i,j)/\,|Q(i,i)|$ in Zustand j
  (P ist die stochastische Transitionsmatrix der eingebetteten Markov-Kette)

$\Rightarrow$ Zukünftiges Verhalten hängt nur vom aktuellen Zustand ab

Matrix Q beschreibt einen gerichteten Graphen



Graph stark zusammenhängend $\Longleftrightarrow$ Markov-Prozess irreduzibel

Ziel der Analyse von Markov-Prozessen:

- Berechnung transienten Verteilung, d.h. Verteilung der Zustandswahrscheinlichkeiten zum Zeitpunkt t (> 0) unter Kenntnis der Zustandwahrscheinlichkeiten zum Zeitpunkt 0
- Berechnung der stationären Verteilung, d.h. Verteilung der Zustandswahrscheinlichkeiten für $t \to \infty$

3.6.1 Analyse von Markov-Prozessen

Stationäre Analyse

Voraussetzungen: Markov Prozess ist ergodisch, d.h.

- Q ist irreduzibel
- falls S unendlich viele Zustände beinhaltet, so muss der Prozess rekurrent sein
  - Prozess ist rekurrent, falls alle Zustände rekurrent sind
  - Zustand i ist rekurrent, falls der Prozess mit Wahrscheinlichkeit 1 nach dem Verlassen von i wieder nach i zurückkehrt
    Sei $f_{ii}^{(k)}$die Wahrscheinlichkeit nach k Schritten wieder zu Zustand i zurückzukehren, es gilt $f_{ii}^{(k)} = (P^k)(i,i)$
    Zustand i ist rekurrent, falls $f_{ii} = \sum_{k=1}^{\infty} f_{ii}^{(k)} = 1$
    (Bsp. M/M/1-System mit $\lambda < \mu$ ist rekurrent, falls $\lambda \geq \mu$, so ist kein Zustand rekurrent)

Ergodizität wird im Folgenden bei der stationären Analyse vorausgesetzt

Bei ergodischen Markov-Prozessen hängt die stationäre Verteilung nicht von der Verteilung zum Zeitpunkt 0 ab

Berechnung der stationären Verteilung:

Stationäre Verteilung wird dargestellt als Vektor $p \in \mathbb{R}_+^{1,n}$, wobei $p(i)$ die stationäre Wahrscheinlichkeit von Zustand i ist

Im stationären Zustand gilt:

Fluss in einen Zustand = Fluss aus einem Zustand

wobei Fluss = Zustandswahrscheinlichkeit · Rate, also

$$\sum_{j=1,j\neq i}^{n} p(j)Q(j,i) = \sum_{i=1,j\neq i}^{n} p(i)\cdot Q(i,j)$$

$$\Leftrightarrow$$

$$\sum_{j=1,j\neq i}^{n} p(j)Q(j,i) + p(i)Q(i,i) = 0$$

oder in Matrixschreibweise $pQ = 0$

Falls Q irreduzibel, so ist der Rang von Q gleich $n-1$

$\Rightarrow$ Normierungsbedingung $\sum_{i=1}^{n} p(i) = 1$ als n-te Gleichung

Falls n endlich, Lösung eines linearen Gleichungssystems

- oft ist n endlich aber sehr groß
- oft ist Q sehr spärlich besetzt

$\Rightarrow$ Anwendung spezieller numerischer Techniken

(nicht Inhalt dieser Vorlesung)

Berechnung von p für $n = \infty$

effektive Berechnung nur bei regulären Strukturen

Beispiel M/M/1 oder M/M/m, weitere Beispiele später

Im allgemeinen Fall Lösung durch

- Simulation (Aussagen auf Basis einer endlichen Untermenge besuchter Zustände)
- Approximation durch die Lösung eines Markov-Prozesses mit endlichem Zustandsraum (Aussagen auf Basis einer vorab festgelegten endlichen Untermenge von Zuständen)

Ableitung von Leistungsgrößen aus den Zustandswahrscheinlichkeiten:

- Interpretation der Zustände
- Ableitung der benötigten Resultatwerte

Beispiel M/M/m:

Zustandsindex - 1 = Population im System

Mittlere Population: $\sum_{i=2}^{\infty}(i-1) \cdot p(i)$

(in diesem Fall geschlossene Form vorhanden)

Wahrscheinlichkeit mindestens 10 Aufträge im System $\sum_{i=11}^{\infty} p(i) = 1 - \sum_{i=1}^{10} p(i)$

...

Transiente Analyse:

Sei p_t Vektor der Zustandswahrscheinlichkeiten zum Zeitpunkt t

Die Verteilung p_t ist durch das Differentialgleichungssystem $\frac{dp_t}{dt} = p_t \cdot Q$ charakterisiert

Die allgemeine Lösung dieses Differentialgleichungssystems lautet $p_t = p_0 \cdot e^{Qt}$

Die einfachste Methode der Lösung ist die Reihenentwicklung der Exponentialfunktion:

$p_t = p_0 \cdot \sum_{k=0}^{\infty} \frac{t^k}{k!} Q^k$ (Abbruch bei endlichem K)

Nachteile des Vorgehens:

- da Matrix Q positive und negative Elemente enthält, ist diese Berechnung numerisch instabil, insbesondere wenn t größer wird
- wenn die Summation abgebrochen wird, ist keine Fehlerschranke verfügbar

Möglicher Ausweg definiere $0 = t_0 < t_1 < t_2 < \ldots < t_n = t$ und berechne $p_{t_m} = p_{t_{m-1}} \cdot \sum_{k=0}^{\infty} \frac{(t_m - t_{m-1})^k}{k!} Q^k$

Für kleine Abstände $t_m - t_{m-1}$müssen nur wenige Summanden berechnet werden, dadurch numerisch stabiler.

Besser ist die Anwendung der Randomisierungs- oder Uniformisierungsmethode!

Sei $\alpha \geq \max_{i \in S}(|Q(i,i)|)$ und $\hat{P} = Q/\alpha + I$, dann gilt:

$e^{Qt} = e^{\alpha t(\hat{P}-I)} = e^{(\alpha t)\hat{P}-(\alpha t)I} = e^{-\alpha t}e^{(\alpha t)\hat{P}}$

Also können wir die Reihenentwicklung von e^{Qt} durch die Reihenentwicklung von $e^{(\alpha t)\hat{P}}$ ersetzen und erhalten

$p_t = p_0 \cdot \sum_{k=0}^{\infty} \hat{P}^k e^{-\alpha t}\frac{(\alpha t)^k}{k!}$

Unterschiede zur ursprünglichen Summe:

- $\hat{P}$ enthält keine negativen Elemente, damit können die Werte numerisch stabil berechnet werden
- $\sum_{k=0}^{\infty} e^{-\alpha t}\frac{(\alpha t)^k}{k!} = 1.0$ Wahrscheinlichkeiten eines Poisson-Prozesses mit Parameter αt
- Sei $\beta(k, \alpha t) = e^{-\alpha t}\frac{(\alpha t)^k}{k!}$ und $p^{(k)} = p_0\hat{P}^k$, dann gilt bei Abbruch der Summation nach K Schritten:

  $$\sum_{k=0}^{K} \beta(k,\alpha t)p^{(k)} \leq p_t$$
  $$\leq \left(\sum_{k=0}^{K} \beta(k,\alpha t)p^{(k)}\right) + \left(1 - \sum_{k=0}^{K} \beta(k,\alpha t)\right)e$$

  wobei e der Einheitsvektor ist

Zur Implementierung:

Berechnung der Vektoren $p^{(k)}$ erfordert nur Vektor-Matrix-Multiplikationen mit nicht-negativen Elementen

Schwieriger ist die Berechnung der $\beta(k, \alpha t)$:

Eine direkte Implementierung führt zu numerischen Problemen

Numerisch stabiler Algorithmus:

$K = 1; s = 1.0; b = 1.0; a = (1 - \epsilon)/e^{-\alpha t};$

while $(b < a)$ do $\{$

$K = K + 1;$

$s = s \cdot (\alpha t)/K;$

$b = b + s;\}$

K enthält die Anzahl der Iterationen, um einen Fehler kleiner als ϵ zu bekommen!

Berechnung der Lösung:

$\pi = p_0; y = p_0$

for $(k = 1$ to $K)$ do $\{$

$y = (\alpha t/k) \cdot (y \cdot \hat{P});$

$\pi = \pi + y; \}$

$p_t = e^{-\alpha t} \cdot \pi;$

Methode anwendbar für endliche n

(falls Vektoren und Matrizen gespeichert werden können)

Für unendliche Zustandsräume theoretisch exakte Berechnung,

- falls p_0 nur endlich viele Zustandswahrscheinlichkeiten größer 0 enthält
- und $\max_{i \in S}(|Q(i.i)|) < \infty$ bekannt,
- so werden für endliches t und Fehlerschranke $\epsilon > 0$ auch nur endlich viele Zustände erreicht

$\Rightarrow$ in endlich vielen Schritten

mit endlichem Speicherplatz analysierbar

Dazu müssen aber Zustandsübergangsraten während der Laufzeit des Algorithmus generiert werden

$\Rightarrow$ keine Standardimplementierung möglich!

3.6.2 Absorbierende Markov-Prozesse

Absorbierender Markov-Prozess enthält einen ausgezeichneten Zustand n, aus dem keine Transitionen abgehen

$$Q = \begin{pmatrix} D_0 & d_1 \\ 0, \ldots, 0 & 0 \end{pmatrix}$$

- $D_0 \in \mathbb{R}^{n-1,n-1}$ mit nicht-negativen Elementen außerhalb der Diagonalen und $D_0 e^T \leq 0$
  (wobei e ein Vektor mit nur 1 ist)

- $d_1 \in \mathbb{R}_+^{n-1,1}$, so dass $D_0 e^T + d_1 = 0$

Wir nehmen an, dass Zustand n von allen anderen Zuständen aus erreichbar ist

Sei $p_0 \in \mathbb{R}^{1,n}$ die initiale Verteilung des absorbierenden Markov-Prozesses

Sei $p_0(n) = 0$ und π_0 der Subvektor der ersten $n-1$ Elemente von p_0

Verhalten des absorbierenden Markov-Prozesses:

- Prozess startet mit Wahrscheinlichkeit $\pi_0(i)$ im Zustand i

- Für $t \to \infty$ endet der Prozess mit absorbierenden Zustand

Analyse der Absorptionszeiten, d.h. der Zeit, bis Zustand n ausgehend von der Startverteilung erreicht ist

Sei $m^k \in \mathbb{R}^{n-1,1}$, der Vektor der bedingten k-ten Momente der Absorptionszeit, d.h. $m^k(i)$ ist das k-te Moment der Absorptionszeit ausgehen von Zustand i und $E[T^k]$ das k-te Moment der Absorptionszeit

Dann gilt

- $E[T^k] = p_0 \cdot m^k$und
- $m^k = k\,(-D_0)^{-1}\,m^{k-1}$ mit $m^0 = e^T$

Also Berechnung des k-ten Moments erfordert die Lösung von k Gleichungssystemen $-D_0 m^k = k \cdot m^{k-1}$ der Ordnung $n-1$

Numerische Berechnung der Verteilungsfunktion der Absorptionszeit $F(t)$

- Bestimme p_t ausgehend von p_0 mittels der Randomisierung
- $F(t) = p_t(n)$

Anwendung absorbierender Markov-Prozesse:

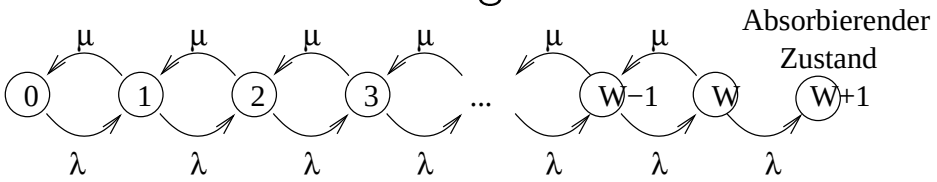
- Analyse von Phasenverteilungen
- Ermittlung detaillierten Zeiten in Modellen (Ausfallzeiten, Verweilzeiten, ...)

Beispiel: Berechnung der Zeitverteilung, bis zum ersten Pufferüberlauf in einem M/M/1/W-System

Zum besseren Verständnis Beibehaltung der Zustandnummerierung nach Auftragszahl, also Zustand $i \in \{0, \ldots, W\}$ beschreibt das System mit i Aufträgen

Ablauf zur Berechnung der Verteilung bis zum ersten Überlauf

- Einführung eines absorbierenden Zustands $n = W + 1$ zur Anzeige eines Überlaufs
- Zustandstransitionsdiagramm des neuen Prozesses:



- Eintritt in den absorbierenden Zustand zeigt ersten Überlauf an.

Mögliche Startverteilungen:

- Leeres System: $p_0(0) = 1$ und $p_0(i) = 0$ für $i > 0$
- Stationäre Verteilung: Sei p_{MM1W} stationäre Verteilung eines M/M/1/W-Modells mit identischen Parametern, wähle $p_0(i) = p_{MM1W}(i)$ für $0 \leq i \leq W$ und $p_0(W + 1) = 0$

Berechnung der Momente und Verteilungsfunktionswerte wie angegeben.

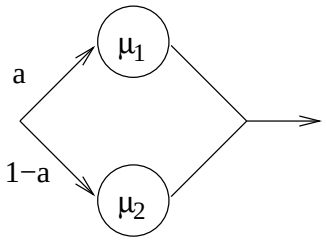
3.6.3 Phasenverteilungen

Markov-Prozesse basieren auf exponentiell verteilten Verweilzeiten in den Zuständen

Kann man trotzdem andere Verteilungen darstellen?

Ja, durch die Verwendung von Phasenverteilungen lassen sich (theoretisch fast alle) Verteilungen exakt oder beliebig genau repräsentieren

Hyperexponentialverteilung:



hier mit 2 Phasen ($\mu_i > 0$ und $0 \leq a \leq 1$)

Es gilt:

- $E(T) = \frac{a}{\mu_1} + \frac{(1-a)}{\mu_2}$
- $E(T^2) = \frac{2a}{(\mu_1)^2} + \frac{2(1-a)}{(\mu_2)^2}$
- $$\begin{aligned} \sigma^2(T) &= E(T^2) - (E(T))^2 \\ &= \frac{a(2-a)}{(\mu_1)^2} + \frac{1-a^2}{(\mu_2)^2} - \frac{2a(1-a)}{\mu_1\mu_2} \end{aligned}$$
  (Varianz)
- $$\begin{aligned} VK^2(T) &= \frac{\sigma^2(T)}{(E(T))^2} \\ &= \frac{a(2-a)(\mu_2)^2+(1-a^2)(\mu_1)^2-2a(1-a)\mu_1\mu_2}{a^2(\mu_2)^2+(1-a)^2(\mu_1)^2+2a(1-a)\mu_1\mu_2} \geq 1 \end{aligned}$$
  (quadrierter Variationskoeffizient)

Mit 2 Phasen ist jeder Variationskoeffizient ≥ 1 erreichbar!

Zur Wahl der Parameter:

Wähle μ_1, μ_2 und a so, dass vorgegebene Werte für $E(T)$ (> 0) und $VK^2(T)$ (≥ 1) erreicht werden.

Drei Parameter und 2 Werte $\Rightarrow$

ein Parameter kann in Grenzen frei gewählt werden

Annahme (üblicherweise) beide Phasen gleich ausgelastet, dann gilt

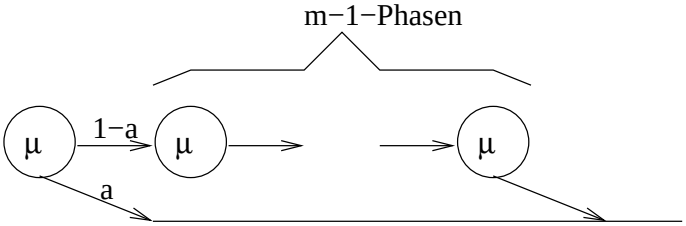
$\frac{a}{\mu_1} = \frac{1-a}{\mu_2} \Leftrightarrow \mu_1 = \frac{a\mu_2}{1-a}$

(andere Wahl z.B. Anpassung 3. Moment ist möglich)

Wahl der Parameter

- $a = VK^2(T)\left(1 - \sqrt{1 - \frac{2}{1+VK^2(T)}}\right)$
- $\mu_1 = \frac{\left(1+\sqrt{1-\frac{2}{1+VK^2(T)}}\right)}{E(T)}$
- $\mu_2 = \frac{\left(1-\sqrt{1-\frac{2}{1+VK^2(T)}}\right)}{E(T)}$

Verallgemeinerte Erlang-Verteilung:



mit den Parametern $\mu > 0$ und $0 \leq a \leq 1$

Es gilt

- $E(T) = \frac{1+(1-a)(m-1)}{\mu}$
- $E(T^2) = \frac{m(1-a+m(1-a))+2a}{\mu^2}$
- $\sigma^2(T) = \frac{m-3ma+m^2a+2a-m^2a^2+2,a^2-a^2}{\mu^2}$
- $VK^2(T) = \frac{m-3ma+m^2a+2a-m^2a^2+2ma^2-a^2}{(ma-m-a)^2}$

  es gilt $1/m \leq VK^2(T) \leq 1$

Bei gegebenen $E(T)$ und $VK^2(T)$ (< 1) wähle

- $m = \lceil 1/VK^2(T) \rceil$
- $a = \frac{m+2VK^2(T)m-2-\sqrt{m^2-4VK^2(T)m-4}}{2(VK^2+1)(m-1)}$
- $\mu = \frac{1+a(m-1)}{E(T)}$

Damit eindeutige Festlegung einer Phasenverteilung bei gegebenem ersten und zweiten Moment

Wachsende Phasenzahl bei Verkleinerung des Variationskoeffizienten

Anpassung weiterer Momente oder der Dfkt./Vfkt. mit allgemeinen Phasenverteilungen

Interpretation der Verteilung als absorbierender Markov-Prozess

- Transitionsraten definieren Matrix Q mit n Zuständen bei $n-1$ Phasen
  - implizit dadurch definiert D_0, d_1 und Startvektor $\pi = p_0$
- Absorptionszeit ist äquivalent zur dargestellten Verteilung

Parameteranpassung

- durch Anpassung weiterer Momente
  - Lösung nichtlinearer Gleichungssysteme
  - Verfahren für dritte Momente bekannt, höhere Momente kaum berücksichtigt
- durch Anpassung der Verteilungsfunktion an beobachtete Stichprobe mit Werten $(t_1, \ldots, t_K)$
  - finde für vorgegebene Phasenzahl $n-1$ die Parameter so, dass $\prod_{k=1}^{K} \pi e^{D_0 t_k} d_1$ maximiert wird
  - komplexes nichtlineares Optimierungsproblem, im allgemeinen Fall kaum effizient lösbar
  - für eingeschränkte Klassen (z.B. D_0 obere Dreiecksmatrix) existieren iterative Optimierungsalgorithmen, die akzeptable Ergebnisse liefern

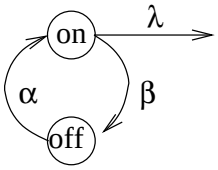
Ankunftsprozesse:

Bisherige Modelle erzeugen unabhängig, identisch verteilte Realisierungen

Viele reale Prozesse (z.B. Ankunftsprozesse an Web-Servern) weisen Korrelationen auf

Anpassung durch Phasenverteilungen, bei denen die Anfangsverteilung vom Abgangszustand abhängt

Spezialfall: Interrupted Poisson Prozess (IPP)



- Markov-Prozess wechselt zwischen Zuständen on und off
- Im Zustand off keine Ankünfte
- Im Zustand on Poisson Ankunftsprozess mit Rate λ

$\Rightarrow$ bei entsprechender Parametrisierung burst-Ankünfte

Darstellung Matrix D_0und D_1(Zustandsübergang nach Ankunft)

$$D_0 = \begin{pmatrix} -\beta-\lambda & \beta \\ \alpha & -\alpha \end{pmatrix} \text{ und } D_1 = \begin{pmatrix} \lambda & 0 \\ 0 & 0 \end{pmatrix}$$

Zustandsverteilung:

$p(1) = \alpha/(\alpha+\beta)$ und $p(2) = \beta/(\alpha+\beta)$

Mittlere Zeit zwischen zwei Ankünften: $(\alpha+\beta)/(\alpha \cdot \lambda)$

Markov Modulierter Poisson Prozess (MMPP):

Verallgemeinerung des IPP mit $n-1$ Zuständen:

Matrix D_0 ist eine $n-1 \times n-1$ Matrix mit

- nicht negativen Elementen außerhalb der Diagonalen
- negativen Diagonalelementen, so dass $D_0 e^T \leq 0$

D_1 ist eine $n-1 \times n-1$ Matrix mit

- nicht negativen Elementen
- $(D_0 + D_1)$ ist irreduzibel und hat Zeilensumme 0

Analyse von MMPPs über absorbierende Markov-Prozesse

Bestimmung der initialen Verteilung:

Initiale Verteilung direkt nach einer beliebigen Ankunft:

$\pi_a = pD_1/(pD_1e^T)$ mit $p(D_0 + D_1) = 0$ und $pe^T = 1.0$

(stationäre Lösung eindeutig, da $D_0 + D_1$ Matrix eines ergodischen Markov-Prozesses)

- Berechnung der Momente: $E(T^k) = \pi_a \cdot m^k$
  mit $m^k = k! \cdot \left((-D_0)^{-1} \right)^k e^T$
- Berechnung der Verteilungsfkt. $F(t) = 1 - \pi_a e^{D_0 t} e^T$
  (Berechnung mittels Randomisierung)

Da Zeiten nicht unabhängig sind, liefert Beobachtung einer Sequenz von Ankünften Informationen über zukünftiges Verhalten

Annahme: Es wurden Ankünfte zu den Zeitpunkten $t_1, \ldots, t_K$ beobachtet

Startverteilung zur Berechnung der Zeit bis zur Ankunft $K+1$:

$$p_0 = \left(\pi_a \prod_{k=1}^{K} e^{D_0 t_k} D_1 \right) / \left(\pi_a \prod_{k=1}^{K} e^{D_0 t_k} D_1 e^T \right)$$

Berechnung von $\pi e^{D_0 t}$ mittels Randomisierung siehe Folie 91

Markovsche Ankunftsprozesse (MAPs):

- Verallgemeinerung von MMPPs
- Matrix D_1 ist eine beliebige nicht negative Matrix
- $D_0 + D_1$ ist Generator-Matrix eines ergodischen Markov-Prozesses

MAPs sind sehr allgemeine Klasse von Modellen, die komplexe Ankunftsprozesse modellieren können

Parametrisierung von MAPs bzgl. gemessener Daten ist komplexes nichtlineares Optimierungsproblem, welches nur für kleinere Zustandszahlen oder spezielle Strukturen beherrscht wird.

Jeder IPP oder MMPP ist auch ein MAP

3.6.4 Analyse von MAP/PH/1-Modellen

Phasenverteilungen und MAPs können zur Modellierung komplexer Verteilungen und Prozesse eingesetzt werden

$\Rightarrow$ es entsteht ein Markov-Prozess

Bei endlichem Zustandsraum numerische Analyse eines linearen Gleichungssystems (stationäre Analyse) oder Randomisierung (transiente Analyse)

Praktische Probleme durch Zustandsraumexplosion:

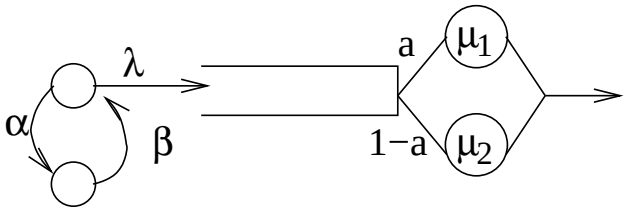
- Ersetzung einer Exponentialverteilung durch eine Phasenverteilung mit n Phasen vergrößert den Zustandsraum um den Faktor n
- Bei Ersetzung mehrerer Verteilungen multiplizieren sich die Faktoren
- Mit heutigen Methoden können Gleichungssysteme mit bis zu 10^7 Zuständen auf einem üblichen PC analysiert werden

Analyse von Systemen mit unendlichem Zustandsraum nur bei Vorliegen einer regelmäßigen Struktur (siehe M/M/..)

Allgemeines Modell: MAP/PH/1

- MAP als Ankunftsprozess, PH Bedienzeitverteilung
- 1 Bediener und unendliche Warteraumkapazität

Beispiel $IPP/H_2/1$-System



Zustandsbeschreibung (i, j, k)mit

- $i \in \{1, 2\}$ Phase Ankunftsprozess
- $k \in \mathbb{N}$ Population im System
- $j = 0$ falls $k = 0$, ansonsten $j \in \{1, 2\}$ Bedienphase

Zustände können in Blöcke bzgl. der Population unterteilt werden

Blöcke werden von 0 ausgehend aufsteigend nummeriert

Block 0 enthält 2 Zustände, alle weiteren Blöcke 4 Zustände

Struktur der Generator-Matrix:

$$Q = \begin{pmatrix} A_1 & A_0 & & & & \\ A_2 & B_1 & B_0 & & & \\ & B_2 & B_1 & B_0 & & \\ & & B_2 & B_1 & B_0 & \\ & & & \ddots & \ddots & \ddots \\ & & & & \ddots & \ddots \end{pmatrix}$$

(gilt allgemein für MAP/PH/1-Modelle)

Aufbau der Matrizen im Beispiel:

$$A_1 = \begin{pmatrix} -\alpha-\lambda & \alpha \\ \beta & -\beta \end{pmatrix}, A_0 = \begin{pmatrix} a\lambda & (1-a)\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} \mu_1 & 0 \\ \mu_2 & 0 \\ 0 & \mu_1 \\ 0 & \mu_2 \end{pmatrix}, B_0 = \begin{pmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$B_1 = \begin{pmatrix} -\mu_1-\alpha-\lambda & 0 & \alpha & 0 \\ 0 & -\mu_2-\alpha-\lambda & 0 & \alpha \\ \beta & 0 & -\mu_1-\beta & 0 \\ 0 & \beta & 0 & -\mu_2-\beta \end{pmatrix}$$

$$B_2 = \begin{pmatrix} a\mu_1 & (1-a)\mu_1 & 0 & 0 \\ a\mu_2 & (1-a)\mu_2 & 0 & 0 \\ 0 & 0 & a\mu_1 & (1-a)\mu_1 \\ 0 & 0 & a\mu_2 & (1-a)\mu_2 \end{pmatrix}$$

Systematische Generierung der Matrizen aus den Matrizen der Ankunfts- und Bedienzeitverteilung

Analyse des Systems:

- Annahme: mittlere Ankunftsrate < mittlere Bedienrate

Sei $\pi = (\pi_0, \pi_1, \ldots)$ der stationäre Verteilung
(π_k ist die Verteilung mit Population k im System)
Sei n die Dimension der quadratischen Matrix A_1 und $n \cdot m$ die Dimension der quadratischen Matrix B_1
(dann ergeben sich automatisch die Dimensionen der restlichen Matrizen)

Weitere Annahme:

- Es existiere eine $nm \times nm$ Matrix R, so dass $\pi_{k+1} = \pi_k R$ für $k \geq 1$

Bei Einsetzen dieser Beziehung für einen Block $k(> 1)$:

$$\pi_k B_1 + \pi_{k-1} B_0 + \pi_{k+1} B_2 = 0 \quad \Rightarrow$$
$$\pi_{k-1} \left(B_0 + R B_1 + R^2 B_2 \right) = 0$$

Diese Gleichung ist offensichtlich erfüllt, wenn

$$B_0 + R B_1 + R^2 B_2 = 0$$

Man kann nun zeigen (ohne dass wir es im Detail tun werden), dass eine minimale positive Matrix R mit Spektralradius < 1 existiert, welche die Gleichung erfüllt

Damit gilt für R: $\lim_{k\to\infty} R^k = 0$ und damit $\lim_{k\to\infty} \pi_k = 0$ (wie zu erwarten)

Angenommen wir kennen R, dann gilt $\pi_k = \pi_1 R^{k-1}$

Bleibt noch die Berechnung von π_0 und π_1

Aus den Flussgleichungen folgt:

$\pi_0 A_1 + \pi_1 A_2 = 0 \Rightarrow \pi_0 = \pi_1 A_2 (-A_1)^{-1}$

π_1 kann aus folgendem Gleichungssystem berechnet werden:

$$\begin{aligned} \pi_0 A_0 + \pi_1 B_1 + \pi_2 B_2 &= 0 \quad \Rightarrow \\ \pi_0 A_0 + \pi_1 (B_1 + R B_2) &= 0 \quad \Rightarrow \\ \pi_1 (A_2 (-A_1)^{-1} A_0 + B_1 + R B_2) &= 0 \end{aligned}$$

Das Gleichungssystem hat Rang $nm - 1$,

d.h. π_1 kann bis auf einen Faktor berechnet werden

Dieser Faktor wir dadurch festgelegt, dass die Summe der Wahrscheinlichkeiten 1 sein muss:

$$\begin{aligned} \sum_{k=0}^{\infty} \pi_k e^T &= 1 \quad \Rightarrow \\ \pi_1 \left(A_2 (A_1)^{-1} + \sum_{k=0}^{\infty} R^k \right) e^T &= 1 \quad \Rightarrow \\ \pi_1 \left(A_2 (A_1)^{-1} + (I - R)^{-1} \right) e^T &= 1 \end{aligned}$$

Liefert die fehlende Gleichung, so dass π_1 und damit alle Vektoren π_k exakt berechnet werden können!

Bleibt noch die Berechnung der Matrix R:

Zentraler Schritt der Analyse

Hier Vorstellung eines Basisverfahrens,

bessere Methoden existieren

Umstellen der Gleichung für R liefert:

$R = (B_0 + R^2 B_2)(-B_1)^{-1}$

Die resultierenden Fixpunktgleichung

$$R_{l+1} = \left(B_0 + (R_l)^2 B_2\right)(-B_1)^{-1}$$

mit $R_0 = 0$ konvergiert gegen Matrix R

Verfahren kann auf weitere ähnliche Modelle erweitert werden:

- $MAP/PH/m$-Systeme
- $GI/PH/1$-Systeme
- $MAP/GI/1$-Systeme

Analyse ähnlich, aber i.d.R. aufwändiger

3.7 Zuverlässigkeit und Verfügbarkeit

Bisher wurde die Leistung von Systemen analysiert,
ein weiterer Aspekt ist die Verfügbarkeit der angebotenen Dienste
Unverfügbarkeit über einen längeren Zeitraum kann
desaströse Konsequenzen haben

(Verlust an Aufträgen, Umsatz, Gewinn, ...)

Gründe für Unverfügbarkeit von Diensten

- Hardware-Fehler
- Netzwerkprobleme
- Software-Fehler
- Denial of Service Angriffe

$\Rightarrow$ viele Aspekte werden berührt

(Hardware, Software, Sicherheit ...)

$\Rightarrow$ Zusammenfassung unter dem Term *dependability*

Hier eine kurze Einführung

3.7.1 Grundlagen der Systemzuverlässigkeit

3.7.2 Verfügbarkeit von Komponenten und Systemen

3.7.3 Detaillierte Modellierung des Ausfallverhaltens

3.7.4 Verfügbarkeit und Leistung

3.7.1 Grundlagen der Systemzuverlässigkeit

Taxonomie zur Klassifizierung von Fehlern

- Dauer
  - Permanente Fehler
  - Behebbare Fehler
  - Transiente Fehler
- Effekt
  - Funktionaler Fehler
  - Quantitativer Fehler (Leistungsreduktion)
- Umfang
  - Partieller Fehler
  - Totaler Fehler

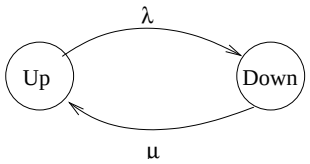
Verfügbarkeit = Prozentualer Anteil der Zeit,
in der das System verfügbar ist

Zugehörige Zeiten

- MTBF Mean Time Between Failure
- MTTF Mean Time To Failure
- MTTR Mean Time To Repair
- MTBF = MTTF + MTTR

Klasse	Verfüg-barkeit	Unverfügbark. (Min./Jahr)	System-typ
1	90.0%	52560	Unmanaged
2	99.0%	5256	Managed
3	99.9%	526	Well-Managed
4	99.99%	52.6	Fault-Tolerant
5	99.999%	5.3	Highly Available
6	99.9999%	0.53	Very Highly Available
7	99.99999%	0.053	Ultra Available

Einfaches Modell einer Komponente mit Fehlern und Reparatur



$\lambda = \frac{1}{MTTF}$ und $\mu = \frac{1}{MTTR}$

Flussgleichgewicht liefert: $\lambda \cdot p_{up} = \mu \cdot p_{down}$

p_{up} und p_{down} sind die Wahrscheinlichkeiten,

dass das System im Zustand up bzw. down ist

Da $p_{up} + p_{down} = 1$ gilt

Verfügbarkeit: $A = p_{up} = \frac{\lambda}{\lambda+\mu}$

Unverfügbarkeit: $U = p_{down} = \frac{\mu}{\lambda+\mu}$

i.a. MTTR leichter zur verkleinern, als MTTF zu vergrößern

3.7.2 Verfügbarkeit von Komponenten und Systemen

Systeme bestehen aus Komponenten $\Rightarrow$

Systemzuverlässigkeit ergibt sich aus

Komponentenzuverlässigkeit

Systemzuverlässigkeit = Wahrscheinlichkeit, dass ein System

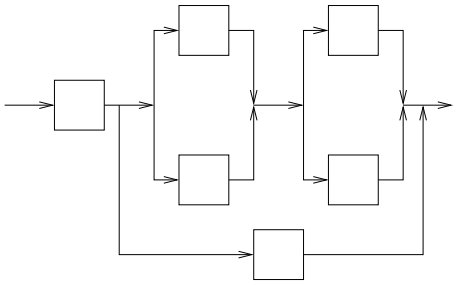
während eines vorgegebene Intervalls durchgängig

verfügbar ist

Typische Abhängigkeiten:

- serielle Abhängigkeit (wenn eine Komponente ausfällt, arbeitet das gesamte System nicht mehr)
- parallele Abhängigkeit (wenn alle Komponenten ausgefallen sind arbeitet das System nicht mehr)
- Kombinationen aus serieller und paralleler Abhängigkeit

Darstellung als Zuverlässigkeitsblockdiagramm:



System ist verfügbar,

wenn ein Pfad durch das Diagramm existiert!

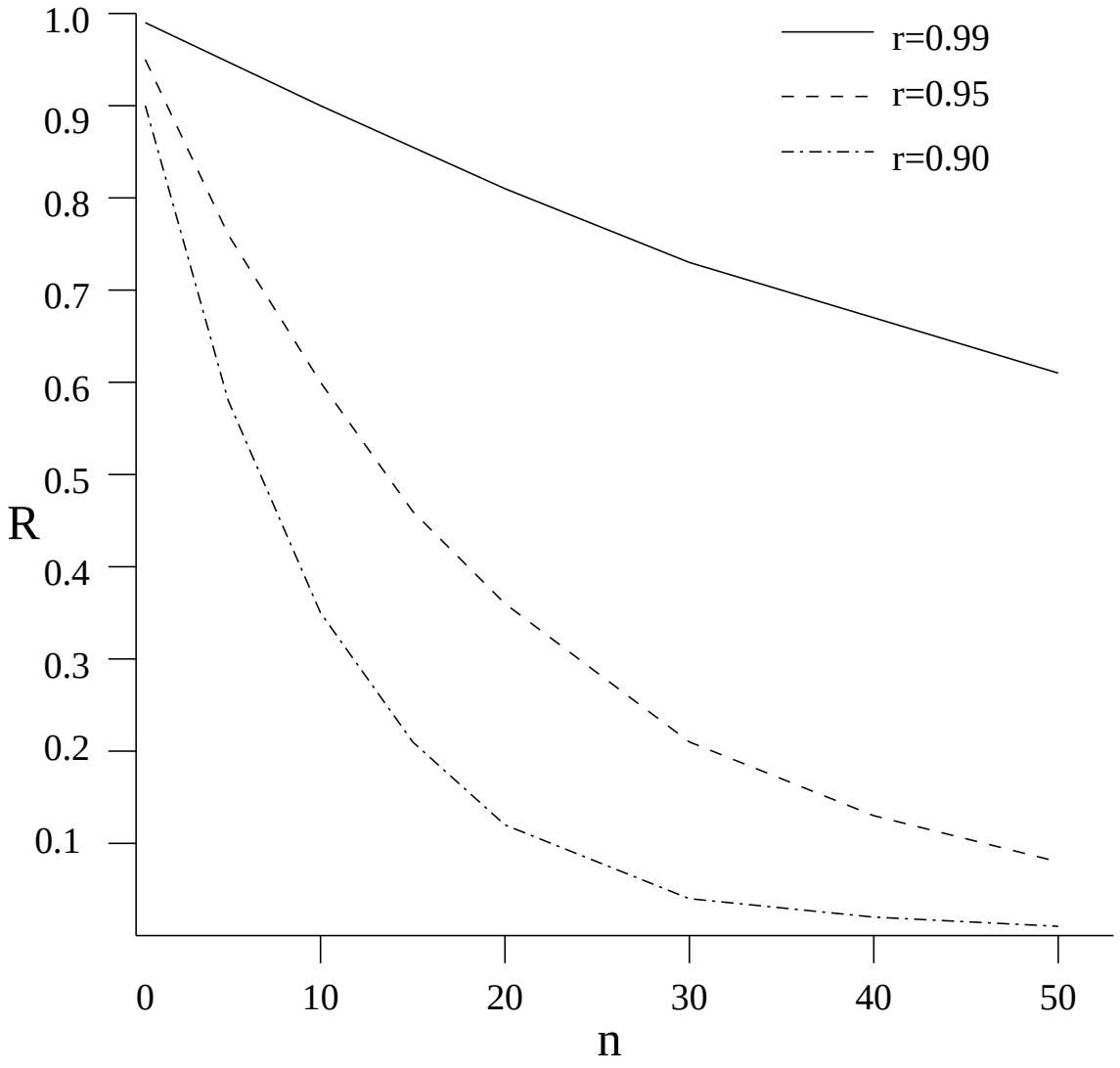
Serielle Abhängigkeit:

R_s = Systemverfügbarkeit

r_i = Verfügbarkeit Komponente i

Es gilt $R_s = \prod_{i=1}^{n} r_i$ für ein System mit n Komponenten

Verlauf der Systemverfügbarkeit für $r_i = r$

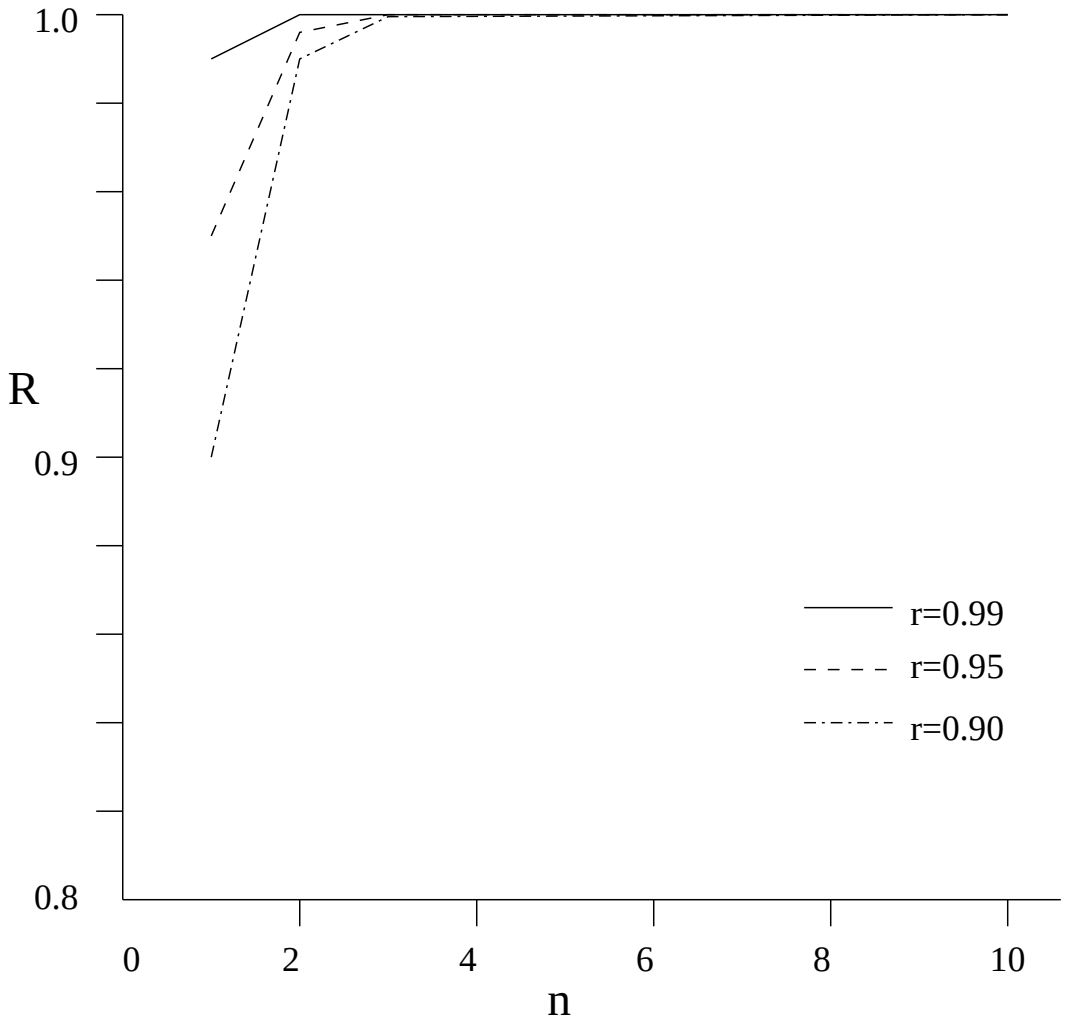


Parallele Abhängigkeit:

Es gilt $R_s = 1 - \prod_{i=1}^{n}(1 - r_i)$

für ein System mit n Komponenten

Verlauf der Systemverfügbarkeit für $r_i = r$



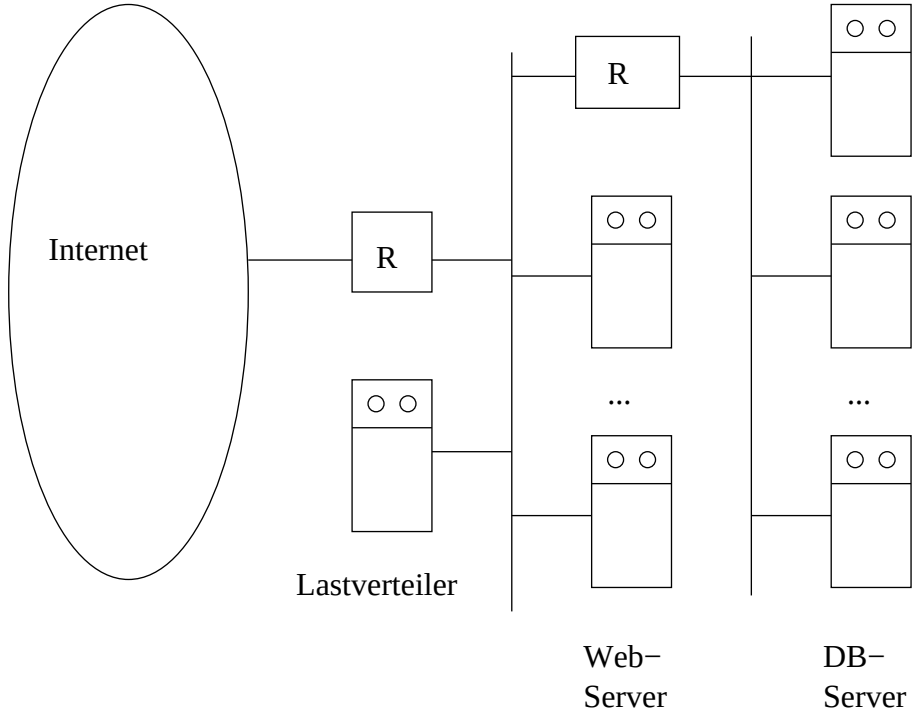
Minimale Anzahl von parallelen Komponenten $n_{\min}$, um Systemverfügbarkeitkeit R_p bei Komponentenverfügbarkeit r zu erreichen: $n_{\min} = \left\lceil \frac{\ln(1-R_p)}{\ln(1-r)} \right\rceil$

Ein Beispiel:

Online-Broker, der die üblichen Transaktionen (Kauf und Verkauf von Aktionen, Optionen, .., Depotverwaltung etc.) über einen Web-Server abwickelt

Anforderung Verfügbarkeit 99.99%

Server-Architektur des Brokers

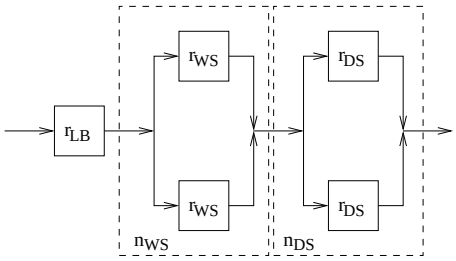


n_{WS} Web-Server und n_{DS} DB-Server

Für beide Server sind hoch-zuverlässige und teurere Systeme (Verfügbarkeit 99.9%) und billigere Standardsysteme (Verfügbarkeit 85%) einsetzbar

Zentrale Frage: Wie kann die geforderte Verfügbarkeit möglichst kostengünstig erreicht werden?

Zuverlässigkeitsblockdiagramm für das System
(vereinfacht, da Netzelemente nicht modelliert)



Für die Verfügbarkeit des Gesamtsystems gilt:

$$\begin{aligned} R_s &= r_{LB} \cdot R_{WS} \cdot R_{DB} \\ &= r_{LB} \cdot [1-(1-r_{WS})^{n_{WS}}] \cdot [1-(1-r_{DS})^{n_{DS}}] \end{aligned}$$

Lastverteiler muss sehr zuverlässig sein,

um $R_s = 0.9999$ überhaupt erreichen zu können!

$\Rightarrow$ sei $r_{LB} = 0.99999$

Damit gilt $0.99991 = [1-(1-r_{WS})^{n_{WS}}] \cdot [1-(1-r_{DS})^{n_{DS}}]$

Einige mögliche Konfigurationen:

Konfiguration	r_{WS}	r_{DS}	n_{WS}	n_{DS}	R_s
le WS/le DS	0.850	0.850	6	5	99.990%
le WS/he DS	0.850	0.999	5	2	99.991%
he WS/he DS	0.999	0.999	2	2	99.999%

Auswahl hängt ab, von

- den Kosten der einzelnen Konfigurationen
- weiterer Zuverlässigkeits- und Sicherheitskriterien
- der jeweils erreichbaren Leistung

3.7.3 Detaillierte Modellierung des Ausfallverhaltens

Bisherige Modellierung ging von unabhängigen Ausfällen aus, war dann aber unabhängig von der Verteilung der Ausfall- und Reparaturzeiten anwendbar

Hier nun eine die Beschreibung komplexerer Modelle

Annahmen:

- Ausfall- und Reparaturzeiten lassen sich durch Phasenverteilungen modellieren
- Ausfälle können korreliert sein
- Reparaturen werden von Reparatureinheiten durchgeführt, die nur in beschränkte Anzahl zur Verfügung stehen

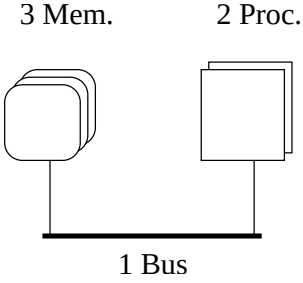
$\Rightarrow$ System lässt sich auf einen Markov-Prozess abbilden

$\Rightarrow$ resultierender Markov-Prozess lässt sich bzgl. stationärer oder transienter Maße analysieren
(siehe Kap. 3.6.1 und 3.6.2)

Vorstellung des Vorgehens an Hand einiger Beispiele

Beispiel:

Berechnung der Zeit bis zum ersten Ausfall eines Rechensystems



System besteht aus

- 1 Bus mit Ausfallrate $1.0e-6$ und Reparaturrate 1.0
- 2 Prozessoren mit Ausfallrate $2.0e-4$ und Reparaturrate 0.5
- 3 Speicherbausteinen mit Ausfallrate $5.0e-4$ und Reparaturrate 0.3

Weitere Festlegungen

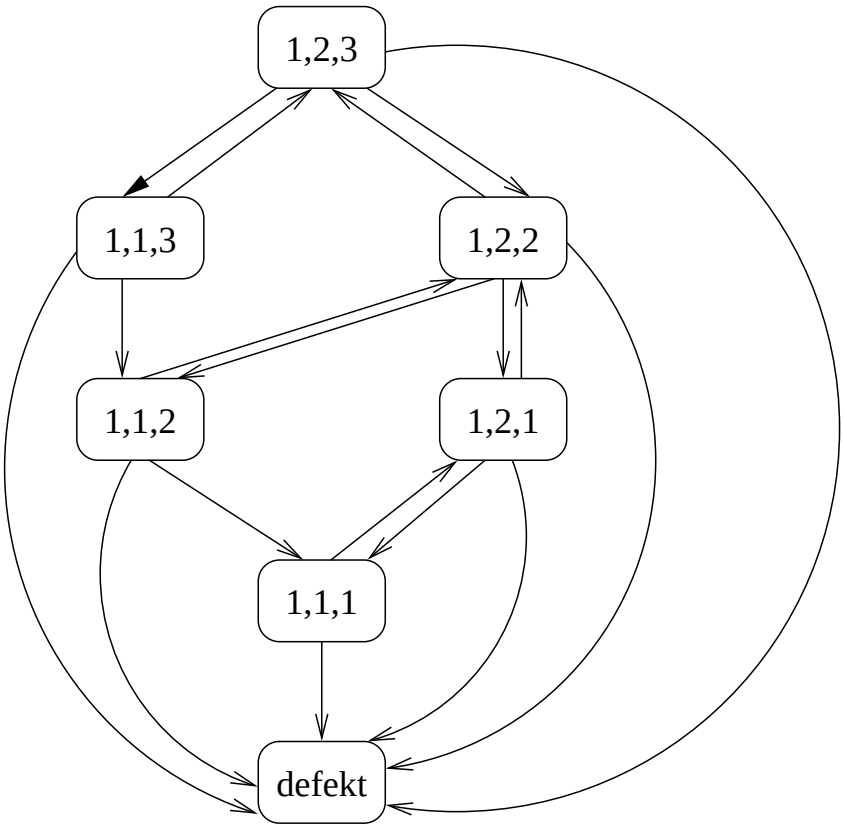
- das System ist funktionsfähig, solange ein Prozessor, ein Speicherbaustein und der Bus verfügbar sind
- wenn das System nicht funktionsfähig ist, so wird es abgeschaltet und es können keine weiteren Komponenten ausfallen
- Ausfall- und Reparaturzeiten seien exponentiell verteilt

Reparaturen werden von einer Reparatureinheit ausgeführt, die mit unterbrechenden Prioritäten arbeitet, zuerst wird der Bus, dann werden die Prozessoren und zuletzt die Speicherbausteine repariert

Ziel: Bestimmung der Verteilung/Momente der Zeit bis zum ersten Systemausfall ausgehend vom System, in dem alle Komponenten funktionsfähig sind

Zustandsübergangsgraph

ohne Angabe der Transitionsraten!



Zustandsbeschreibung (B,P,M) =

Anzahl verfügbarer Busse, Prozessoren und Speicherbausteine

Analyse der Zeit bis zum Systemausfall wie beschrieben durch

- Berechnung der Momente der Absorptionszeit
- Berechnung der Verteilung der Absorptionszeit

Berechnung der mittleren Verfügbarkeit eines Systems:

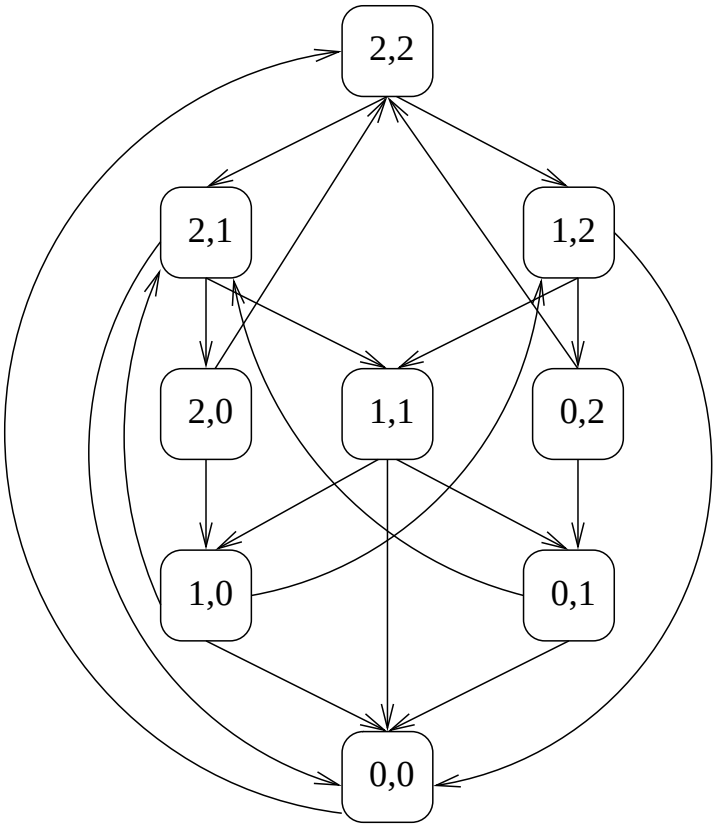
- System besteht aus 2 Komponenten, deren Ausfallzeiten Erlang 2 verteilt sind mit Mittelwert $(2/\mu_i)$ $(i = 1, 2)$
- Fällt eine Komponente aus, so wird mit Wahrscheinlichkeit c die zweite Komponente im Mitleidenschaft gezogen und fällt ebenfalls aus
- Komponenten werden von einer Reparatureinheit repariert, die Reparaturzeiten sind exponentiell verteilt
- Falls nur eine Komponente ausgefallen ist, dauert die Reparatur im Mittel λ^{-1} Zeiteinheiten
- Falls beide Komponenten ausgefallen sind, so werden beide gleichzeitig repariert, die Reparatur dauert dann im Mittel $2/\lambda$ Zeiteinheiten
- Das System ist funktionsfähig, solange mindestens eine Komponente arbeitet, das System ist damit verfügbar, wenn eine Komponente verfügbar ist

Zustandsbeschreibung (s_1, s_2) mit $s_i \in \{0, 1, 2\}$ wobei

$s_i = 1, 2$: Komponente i befindet sich in der $i$ten Ausfallphase

$s_i = 0$: Komponente i ist ausgefallen

Zustandsübergangsgraph



$\Rightarrow$ Aufbau der Generatormatrix Q

Lösung von $pQ = 0$ und $pe^T = 1.0$

1.0 - Wahrscheinlichkeit Zustand (0,0) entspricht der mittleren Verfügbarkeit

3.7.4 Verfügbarkeit und Leistung

Leistung ist abhängig von der Zuverlässigkeit eines Systems und auch von der verfügbaren Konfiguration

Analyse muss beides berücksichtigen,

- Verfügbarkeit des Dienstes und
- Leistung mit der ein Dienst erbracht wird (evtl. ohne SLAs einzuhalten)

Beispiel:

Web-Server bestehend aus n Servern,

die jeweils verfügbar oder defekt sein können

Ankunftsrate von Anfragen λ Anf./Sek.

$\Rightarrow$ bei k verfügbaren Servern Ankunftsrate pro Server λ/k

$\Rightarrow$ Leistungsmaße wie Antwortzeit hängt von der Zahl der verfügbaren Server ab

Kombinierte Analyse von Leistung und Zuverlässigkeit

$\Rightarrow$ $Performability$ Analyse

Analyse in drei Schritten

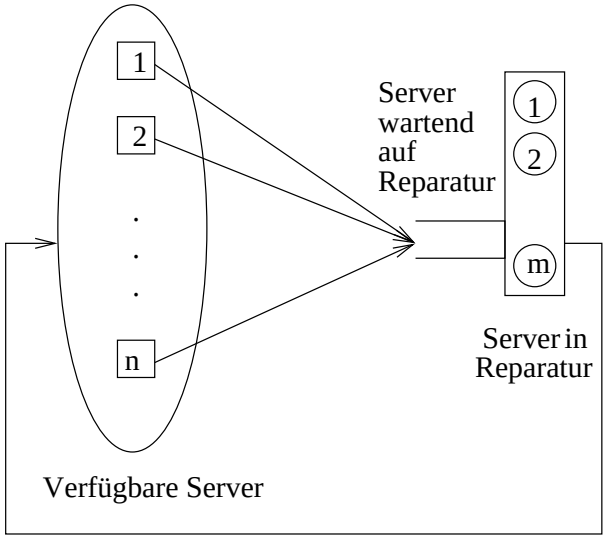
1. Analyse der Zuverlässigkeit
2. Analyse der Leistung für alle möglichen Konfigurationen
3. Kombination der Ergebnisse

Zuverlässigkeitsanalyse

Basismodell der Zuverlässigkeitsanalyse

- n Server
- jeder Server fällt mit Rate $1/MTTF$ aus
- bis zu m Server können gleichzeitig repariert werden
- die Reparaturrate eines Servers beträgt $1/MTTR$

Darstellung des Zuverlässigkeitsmodells



Hier vorausgesetzt:

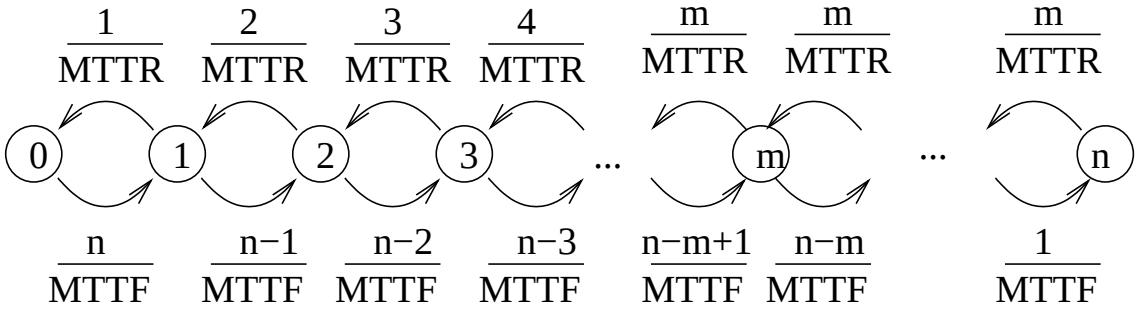
Annahmen der operationalen Analyse gelten bzw. Ausfall- und Reparaturzeiten sind exponentiell verteilt

$\Rightarrow$ Modell ist eine spezielle Form des M/M/m Systems

mit endlicher Population und

lastabhängiger Bedienrate (siehe Folie 24)

Zustandsübergangsdiagramm



Sei $\rho = MTTR/MTTF$, dann gilt

$$p_k = \begin{cases} p_0 \rho^k \begin{pmatrix} n \\ k \end{pmatrix} & k = 1, \ldots, m \\ \frac{p_0 \rho^k n!}{m!(n-k)! m^{k-m}} & k = m+1, \ldots n \end{cases}$$

wobei p_k die Wahrscheinlichkeit für k defekte Server ist

Auf Grund der Normierungsbedingung gilt ferner

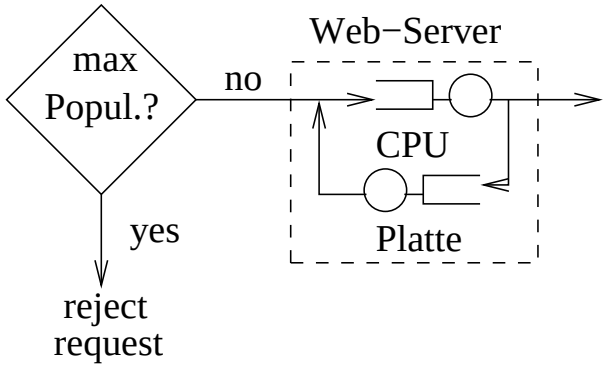
$$p_0 = \left[\sum_{k=0}^{m} \rho^k \begin{pmatrix} n \\ k \end{pmatrix} + \frac{n!}{m!} \sum_{k=m+1}^{n} \frac{\rho^k}{(n-k)! m^{k-m}} \right]^{-1}$$

Aus den Wahrscheinlichkeiten lassen sich die Zuverlässigkeitsmaße berechnen:

- Verfügbarkeit $A = 1 - p_n$
- Durchschnittliche Zahl verfügbarer Server $\bar{N}_{up} = \sum_{k=0}^{n-1} (n-k) \cdot p_k$

Leistungsanalyse

Modell eines einzelnen Web-Servers



Jeder Web-Server kann bis zu

r Anfragen gleichzeitig bearbeiten

weitere Anfragen werden abgewiesen

Vorgehen bei der Leistungsanalyse:

- Analysiere den Web-Server als geschlossenes Modell für feste Populationen $l = 1, \ldots r$

  (d.h. Aufträge, die den Server verlassen betreten ihn auch sofort wieder, siehe auch Kap. 3.7.2)

  $\Rightarrow$ Durchsatz $X(l)$ mit l Aufträgen

- Analysiere für $k = 1, \ldots, n$ ein M/M/.. Modell mit Kapazität r, Ankunftsrate λ/k und lastabhängiger Bedienrate $\mu_l = X(l)$ (siehe 3.1.3)

  $\Rightarrow$ Durchsatz X_0^k, Antwortzeit R_0^k, Population N_0^k und Verlustwahrscheinlichkeit p_{loss}^k

Performability Analyse

Kombination der Resultate

der Zuverlässigkeits- und der Leistungsanalyse

- Gesamtdurchsatz: $X_0 = \sum_{k=1}^{n} X_0^k \cdot p_{n-k}$
- Verlustwahrscheinlichkeit $p_{loss} = 1 - \frac{X_0}{\lambda}$
- Mittlere Antwortzeit $R_0 = \sum_{k=1}^{n} R_0^k \cdot p_{n-k}$
- Mittlere Population $N_0 == \sum_{k=1}^{n} N_0^k \cdot p_{n-k}$

$\Rightarrow$ da die Zeitkonstanten des Leistungsmodells und des Zuverlässigkeitsmodells sich deutlich unterscheiden, liefert das Vorgehen genaue Resultate (sofern die Resultate der beiden Analyseschritte genau sind)

3.8 Erweiterte Analysetechniken

Es existieren zahlreiche Erweiterungen der exakten Analyse von Produktformnetzen

Primäre Zielrichtungen:

1. Approximative Analyse von Nichtproduktformnetzen
   (Erweiterung der Modellklasse)
2. Approximative Analyse sehr großer Netze
   (Effizienzsteigerung)

zu 1. wurden bereits erste Ansätze eingeführt
(Mehrklassennetze für Verteilungen mit *heavy tails*,
Berücksichtigung von *bursts* durch Justierung der
Ankunftsrate)

Oft auftretende Charakteristika,
die bisher nicht behandelbar sind:

- FCFS Bediener mit klassenspezifischer Bedienzeit
- Prioritäten bei der Bedienung
- Simultane Belegung von Ressourcen
- Zugangsbeschränkungen zu Teilsystemen

Ansätze zur Behandlung dieser Charakteristika

- Erweiterungen der Mittelwertanalyse
- Dekomposition und Aggregierung

zu 2.

Im wesentlichen Schrankenberechnung

relevant für geschlossene Netze mit

- großer Population
- vielen Stationen

also keine "exakten Resultate"

sondern obere und untere Schranken für Leistungsmaße

Struktur des Kapitels

3.8.1 Flussäquivalente Aggregierung

3.8.2 Anwendungsbeispiele Aggregierung

3.8.3 Approximative Analyse von Nichtproduktformnetzen

3.8.4 Simultane Ressourcenbelegung

3.8.5 Schrankenberechnung

3.8.6 Grenzen der Modellierung mit Warteschlangennetzen

3.8.1 Flussäquivalente Aggregierung

Basis:

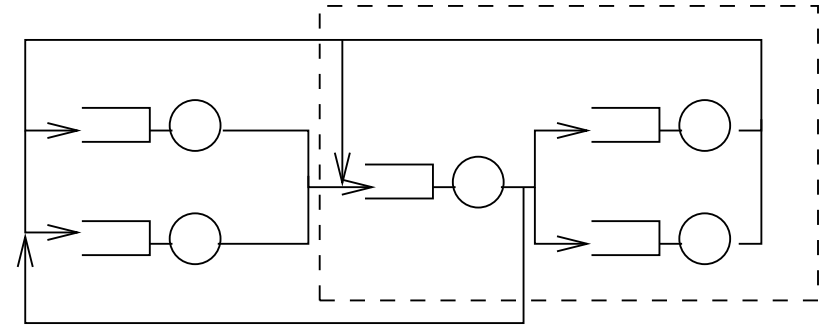
Ersetzung eines Teilnetzes durch
eine lastabhängige Station

- exakte Resultate für Produktformnetze
- oftmals gute Approximation für allgemeine Netze
- Basis für hybride Analysen
  (d.h. Mischung unterschiedlicher Techniken)

Vorgehen bei der flussäquivalenten Aggregierung

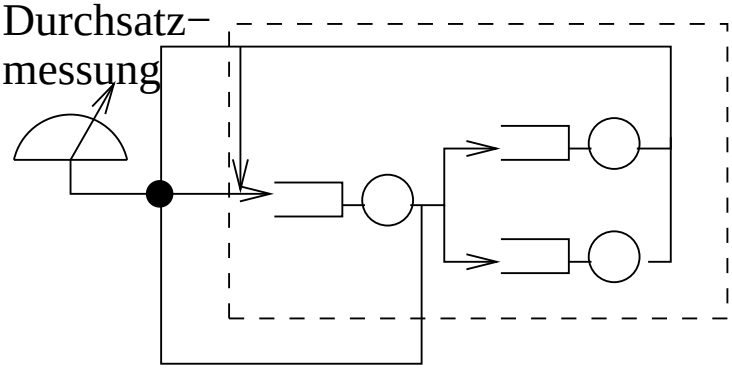
1. Dekomposition des Netzes in Teilnetz und Rest
2. Isolierte Analyse des Teilnetzes für alle möglichen Populationen und Bestimmung der lastabhängigen Durchsätze
3. Definition Aggregat als lastabhängige Station mit
   Bedienrate = lastabhängiger Teilnetzdurchsatz
4. Einbindung Aggregat in Umgebung
5. Analyse aggregiertes Gesamtnetz

Einklassenfall



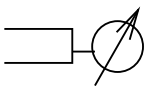
Dekomposition

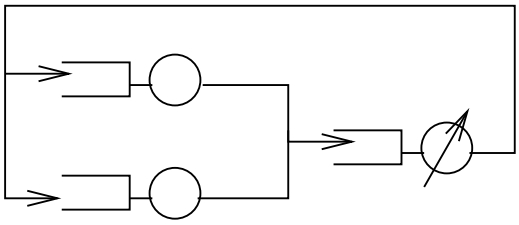
Durchsatz−
messung



Voranalyse
für 1..N

Aggregierung





Aggregiertes Modell

Algorithmisches Vorgehen

Kontext der Mittelwertanalyse $\Rightarrow$

Topologie des Teilnetzes spielt keine Rolle

Analyse auf Basis der relativen Besuchshäufigkeiten!

Also

- Netz mit Stationen $1, \ldots, K$ und Population N
- Aggregierung der Stationen $L+1, \ldots, K$
- Aggregiertes Netz mit Stationen $1, \ldots, L$ plus Aggregat $L+1$

Voranalyse

Stationen $L+1, \ldots, K$ definieren geschlossenes Netz

mit Bedienbedarf $D_i = V_i \cdot S_i$

an Station $i \in \{L+1, \ldots, K\}$

$\Rightarrow$ Bestimme für dieses Netz

für alle Populationen $n = 1, \ldots, N$

Durchsätze $X_0(n)$

Ein (!!) Durchlauf durch Mittelwertanalysealgorithmus (Folie 40) ist notwendig zur Berechnung aller $X_0(n)$

Aggregatbildung

Definiere lastabhängige Station mit

Bedienzeit $S_{L+1}(n) = (X_0(n))^{-1}$

Aggregiertes Modell

bestehend aus Stationen $1, \ldots, L$ mit

unveränderten Parametern

und lastabhängiger Station $L+1$ mit $V_{L+1} = 1$

Falls Ausgangsnetz Produktformnetz gilt:

Resultate für Stationen $1, \ldots, L$ im aggregierten Netz und im Ausgangsnetz sind identisch

$\Rightarrow$ exakte Aggregierung

Besonders effizient für parametrische Studien

Experimente über Modell bei dem Stationen $1, \ldots, L$

ihre Parameter ändern

- einmal aggregieren
- Aggregat mehrfach verwenden

Anwendung bei offenen Netzen

Problem: Population potenziell unendlich

Lösung: wähle großes N

(d.h. N so dass $|X_0(N+1) - X_0(N)| < \varepsilon$)

und setze $S_{L+1}(k) = S_{L+1}(N)$ falls $k > N$

Mehrklassennetze

Übertragung vom Einklassenfall

1. Analysiere Teilnetz der Stationen $L+1,\ldots,M$ für alle Population $\underline{0},\ldots,\underline{N}$ und bestimme last- und klassenabhängige Durchsätze $X_{0,r}(\underline{n})$

2. Bilde Ersatzstation mit Bedienzeiten

   $$S_{L+1,r}(\underline{n}) = (X_{0,r}(\underline{n}))^{-1}$$

3. Analysiere aggregiertes Modell

Bemerkungen:

Voranalyse erfordert einen Durchlauf
der Mittelwertanalyse

Resultierendes Aggregat erfüllt
Produktformeigenschaften d.h.

$$\frac{S_{L+1,r}(\underline{n})}{S_{L+1,s}(\underline{n})} = \frac{S_{L+1,r}(\underline{n}-\underline{1}_s)}{S_{L+1,s}(\underline{n}-\underline{1}_r)}$$

(falls Stationen $L+1,\ldots,M$ Produktformnetz bilden)

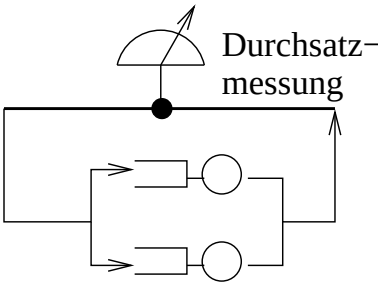
Anwendung auf allgemeine Modelle

Betrachtung allgemeiner Warteschlangennetze mit stochastischem Routing und allgemeinen Stationstypen

Anforderungen an Teilnetze:

1 Eingang und 1 Ausgang pro Klasse

Voranalyse wird zur "Kurzschlussanalyse"



Hybride Analyse

1. Voranalyse mittels Simulation oder numerischer Analyse, Analyse aggregiertes Modell mittels analytischer Techniken
2. Voranalyse mittels analytischer Techniken, Analyse aggregiertes Modell mittels Simulation oder numerischer Techniken

in beiden Fällen Aufwandsreduktion

aber auch keine exakten Resultate

Vorsicht bei 1. und mehreren Klassen: Es entstehen i.d.R. keine Produktformaggregate!

3.8.2 Anwendungsbeispiele Aggregierung

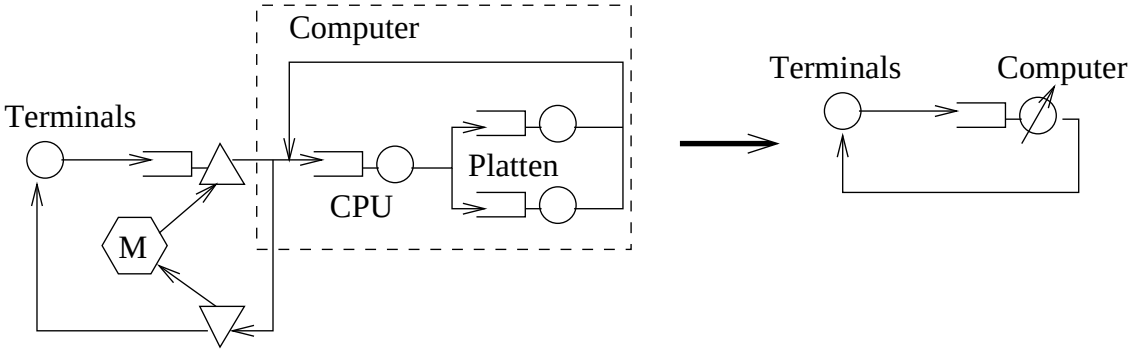
Typisches Anwendungsproblem:

Simultane Belegung von Ressourcen

Tritt in verschiedenen Ausprägungen

in Rechen- und Kommunikationssystemen auf

Rechnermodell mit Speicherbeschränkung



maximal M von N Prozessen können aktiv sein

Analyse mittels Aggregierung

Analysiere Computermodell für Popul. $1, \ldots, M$

Ersetze Computermodell und Speichermodule

durch lastabhängige Station mit Bedienzeit

$$S_{Comp}(n) = \begin{cases} (X_{Comp}(n))^{-1} & falls\, n \leq M \\ (X_{Comp}(M))^{-1} & sonst \end{cases}$$

Berechnung der Resultate:

Direkt aus dem aggregierten Modell ablesebar:

- Durchsatz
- Population und Verweilzeit

  Computer + Speicherzuteilung

Aus dem aggregierten Modell ablesbar,

  falls Populationsverteilung berechnet wird

- Population Speicherzuteilung und Population Computer

  separat

$n_{mem} = \sum_{n=M+1}^{N} P_{Comp}(n|N) \cdot (n - M)$

$n_{comp} = n_{Computer} - n_{mem}$

über Little

- Verweilzeiten Computer und Speicherzuteilung separat
- Resultate für Stationen im Teilnetz über Kombination

  Ergebnisse Voranalyse und Analyse aggregiertes System

<u>Mehrklassennetze</u>

Individuelle Speicheranforderungen je Klasse

Im Prinzip genauso behandelbar

aber: Geschwindigkeiten erfüllen

Produktformbedingungen nicht mehr!!

3.8.3 Approximative Analyse von Nichtproduktformnetzen

Zentrale Anwendung:
Analyse komplexer Schedulingalgorithmen an Stationen mit lastunabhängiger Bedienzeit
(es gibt weitere Anwendungen, die wir hier aber nicht betrachten wollen!)

Basisannahme: Ankunftstheorem gilt
(ankommender Auftrag sieht stationäre Verteilung)
dies kann zu starken Verfälschungen führen!
$\Rightarrow$ Validierung der Ergebnisse ist notwendig

Aber die Annahmen die Gültigkeit des Ankunftstheorems sorgt dafür, dass der Algorithmus der Mittelwertanalyse
(siehe Folie 40 oder 44) anwendbar bleibt, wenn eine neue Formel zur Berechnung der mittleren Verweilzeiten verwendet wird

Damit erfordert die Entwicklung eines MVA-Algorithmus "nur" die Herleitung einer Formel für die Verweilzeit eines Auftrags der n bzw. N Aufträge bei seiner Ankunft antrifft

Allgemein verteilte Bedienzeiten an FCFS-Stationen:

Einklassenfall

Übertragung der Ergebnisse für M/G/1 Systeme

Für offene Netze:

$$\begin{aligned} R_i &= S_i + S_i \cdot n_i + U_i \cdot (S_i^{res} - S_i) \\ &= S_i + S_i \cdot R_i \cdot X_i + U_i \cdot (S_i^{res} - S_i) \\ &= \frac{S_i + U_i \cdot S_i^{res} - U_i \cdot S_i}{1 - S_i \cdot X_i} \\ &= S_i + \frac{U_i \cdot S_i^{res}}{1 - U_i} \end{aligned}$$

Für geschlossene Netze:

$$R_i(n) = S_i \cdot (1 + n_i(n-1)) + U_i(n-1) \cdot (S_i^{res} - S_i)$$

mit $S_i^{res} = S_i \cdot \frac{VK_i^2 + 1}{2}$ und

VK_i^2 quadr. Variationskoeffizient der Bedienzeit an i

Mehrklassenfall

Einfache Übertragung für geschlossene Netze:

$$\begin{aligned} R_{i,r}(\underline{n}) = \quad & S_{i,r} + \sum_{s=1}^{R} S_{i,s} \cdot n_{i,s}(\underline{n} - \underline{1}_r) + \\ & \sum_{s=1}^{R} U_{i,s}(\underline{n} - \underline{1}_r) \cdot (S_{i,s}^{res} - S_{i,s}) \end{aligned}$$

mit $S_{i,r}^{res} = S_{i,r} \cdot \frac{VK_{i,r}^2 + 1}{2}$

Für offene Netze:

$R_{i,r} = S_{i,r} + W_{i,r}$

Für die Wartezeit gilt:

$$\begin{aligned} W_{i,r} &= \sum_{s=1}^{R} (n_{i,s} - U_{i,s}) \cdot S_{i,s} + \sum_{s=1}^{R} U_{i,s} \cdot S_{i,s}^{res} \\ &= \sum_{s=1}^{R} X_{i,s} \cdot W_{i,s} \cdot S_{i,s} + \sum_{s=1}^{R} U_{i,s} \cdot S_{i,s}^{res} \\ &= \sum_{s=1}^{R} U_{i,s} \cdot W_{i,s} + \sum_{s=1}^{R} U_{i,s} \cdot S_{i,s}^{res} \end{aligned}$$

Damit ist $W_{i,s} = W_{i,t}$ für alle Klassen $t, s \in \{1, \ldots, R\}$

$\Rightarrow$ die Wartezeit ist für alle Klassen identisch!

Wenn wir $W_{i,s}$ durch W_i ersetzen, erhalten wir

$$\begin{aligned} W_i &= W_i \cdot \sum_{s=1}^{R} U_{i,s} + \sum_{s=1}^{R} U_{i,s} \cdot S_{i,s}^{res} \\ &= \frac{\sum_{s=1}^{R} U_{i,s} \cdot S_{i,s}^{res}}{1 - U_i} \end{aligned}$$

Im Gegensatz ergibt sich die Wartezeit einer Produktformstation zu (z.B. processor sharing)

$W_{i,r} = \frac{U_i}{1-U_i} S_{i,r}$

Prioritäten

Üblicher Mechanismus in Rechensystemen

Einteilung von Aufträgen in Klassen

Bevorzugte Bedienung bestimmter Klassen

Wir nehmen an: r hat Priorität über s, falls $r > s$

Man unterscheidet

- unterbrechende Prioritäten
  - höher prioriger Auftrag unterbricht niedriger priorigen bei Ankunft,
  - unterbrochener Auftrag beginnt später mit neuer Bedienung (*repeat*) oder
  - unterbrochener Auftrag fährt mit der Bedienung fort, wo er unterbrochen wurde (*resume*)
- nicht unterbrechende Prioritäten
  - nach Beginn der Bedienung wird diese erst beendet,
  - bei Auswahl des nächsten Auftrags wird der mit höchster Priorität aller wartenden ausgewählt

Innerhalb einer Klasse FCFS Scheduling

Unterbrechende Prioritäten mit resume:

Betrachten wir einen Auftrag der Klasse r

Verweilzeitweilzeit an Station i setzt sich zusammen aus:

1. seiner eigenen Bedienzeit
2. der Restbedienzeit des Auftrags in Bedienung, falls dieser keine niedrigere Priorität als r hat
3. der Bedienzeit der Aufträge in der Warteschlange mit Priorität $s \geq r$
4. der Bedienzeit bis zum Ende der Bedienung ankommender Aufträge mit Priorität $s > r$

zur Berechnung der einzelnen Terme

1. Eigene Bedienzeit: $S_{i,r}$
2. Restbedienzeit:

W. [Auftrag mit mind. gleich hoher Priorität in Bedienung] multipliziert mit dessen Restbedienzeit

$\sum_{s=r}^{R} U_{i,s}(\underline{N} - \underline{1}_r) \cdot S_{i,s}^{res}$

3. Bedienzeit der Aufträge in der Warteschlange vor dem Auftrag mit mind. gleich hoher Priorität

$\sum_{s=r}^{R}(n_{i,s}(\underline{N} - \underline{1}_r) - U_{i,s}(\underline{N} - \underline{1}_r)) \cdot S_{i,s})$

4. Während der Wartezeit oder Bedienung des Auftrags
   eintreffende Aufträge mit höherer Priorität

$R_{i,r}(\underline{N}) \cdot \sum_{s=r+1}^{R} \lambda_{i,s}(\underline{N} - \underline{1}_r) \cdot S_{i,s}$

mit $\lambda_{i,s}(\underline{N} - \underline{1}_r)$ Ankunftsrate von Klasse s Aufträgen
   an Station bei $\underline{N} - \underline{1}_{:r}$ Aufträgen im Netz

$R_{i,r}(\underline{N}) \cdot \lambda_{i,s}(\underline{N} - \underline{1}_r)$

   entspricht der Anzahl der während der Verweilzeit
      des Klasse r Aufträge ankommenden Aufträge
         der Klasse s

Weiterhin gilt $U_{i,s}(\underline{N} - \underline{1}_r) = \lambda_{i,s}(\underline{N} - \underline{1}_r) \cdot S_{i,s}$

Dies ergibt folgende Darstellung der Verweilzeit

$$R_{i,r}(\underline{N}) =$$

$$\frac{S_{i,r} + \sum_{s=r}^{R} (S_{i,s}^{res} \cdot U_{i,s}(\underline{N} - \underline{1}_r) + S_{i,s} \cdot (n_{i,s}(\underline{N} - \underline{1}_r) - U_{i,s}(\underline{N} - \underline{1}_r))}{1 - \sum_{s=r+1}^{R} U_{i,r}(\underline{N} - \underline{1}_r)}$$

Verweilzeit ist unabhängig von Aufträgen der Klassen $s < r$
(diese werden einfach verdrängt)

Auf Basis der Verweilzeitdarstellung kann
   Mittelwertanalyse einfach realisiert werden

Nicht unterbrechende Prioritäten

Vorgehen ähnlich zum unterbrechenden Fall

Unterschiede:

In Bedienung befindlicher Auftrag kann nicht
unterbrochen werden

Damit wird 2. Berechnung der Restbedienzeit zu

$$\sum_{s=1}^{R} U_{i,s}(\underline{N} - \underline{1}_r) \cdot S_{i,s}^{res}$$

Auftrag kann selbst nicht während seiner Bedienung unterbrochen werden

Damit werden höher priorige Aufträge nur berücksichtigt,
wenn sie während der Wartezeit eintreffen

$$W_{i,r}(\underline{N}) \cdot \sum_{s=r+1}^{R} \lambda_{i,s}(\underline{N} - \underline{1}_r) \cdot S_{i,s}$$

wobei $W_{i,r}(\underline{N})$ = Wartezeit von Klasse r an Station i
bei $\underline{N}$ Aufträgen im Netz

Berechnung der Wartezeit

$$W_{i,r}(\underline{N}) = \frac{\sum_{s=1}^{R} S_{i,s}^{res} \cdot U_{i,s}(\underline{N}-\underline{1}_r) + \sum_{s=r}^{R} S_{i,s} \cdot (n_{i,s}(\underline{N}-\underline{1}_r) - U_{i,s}(\underline{N}-\underline{1}_r)}{1 - \sum_{s=r+1}^{R} U_{i,r}(\underline{N}-\underline{1}_r)}$$

Verweilzeit: $R_{i,r}(\underline{N}) = W_{i,r}(\underline{N}) + S_{i,r}$

Vorgehen intuitiv und leicht verständliche

Aber:

oft große Approximationsfehler!

Grund:

Gerade für Aufträge mit niedriger Priorität treffen Annahmen der Mittelwertanalyse nicht zu

Ankunftstheorem ist in der Regel stark verfälscht

Beispiel:

Bedienung eines Auftrags mit niedriger Priorität an einer Station mit unterbrechender Priorität

Während der Bedienung kann kein Auftrag mit höherer Priorität an der Station sein!

D.h. während der Bedienzeit des Auftrags werden typischerweise weniger Aufträge mit höher Priorität eintreffen als üblich

⇒Verweilzeiten von Aufträgen mit niedriger Priorität werden überschätzt

⇒ Fehler wächst mit steigender Auslastung durch Aufträge mit hoher Priorität

3.8.4 Simultane Ressourcenbelegung

Typisches Szenario in C/S-Systemen:

Prozesse benötigen unterschiedliche Ressourcen gleichzeitig

Insbesondere gleichzeitige Belegung von Hardware und Software-Ressourcen

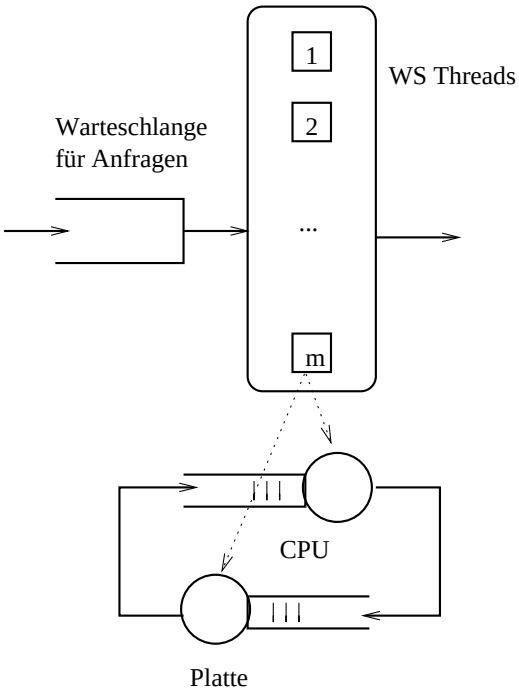
- Software-Ressourcen sind
  - Threads, die auf verschiedenen Ebenen Abfragen bearbeiten
    - Web-Server Thread
    - Anwendungs-Server Threads
    - DB-Server Thread und DB-Locks
- Hardware Ressourcen sind
  - Computer und Kommunikationssysteme, auf denen die Threads ablaufen
    - CPU
    - Platten
    - Hauptspeicher
    - LAN

Szenario in den bisherigen Modellen

nur sehr eingeschränkt darstellbar

$\Rightarrow$ Methoden zur Beschreibung und effizienten Analyse sind notwendig

Typische Beispiele



Anfragen treffen am Web-Server ein und
werden von den Threads des Web Servers bearbeitet

m gleichzeitig aktive Threads im Web-Server

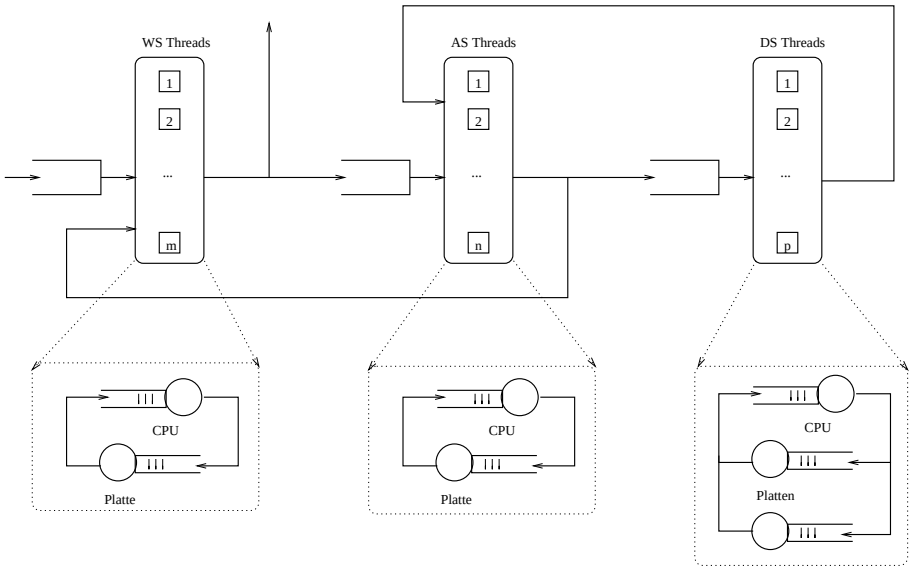
$\Rightarrow$ bis zu m Anfragen können simultan bearbeitet werden,
weitere Anfragen müssen warten

alle Server Threads werden auf einer Maschine abgearbeitet,
d.h. konkurrieren um CPU und Platten

$\Rightarrow$ Anfrage belegt simultan Web-Server Thread und
CPU oder Platte

Analyse dieses einfachen Modells z.B.
mit flussäquivalenter Aggregierung (siehe Folie 136)

Komplexeres Modell eines Web-Servers



Ablauf von Anfragen:

- Anfragen treffen am Web-Server (WS) ein und versuchen einen von m Threads zu belegen
  - falls kein Thread frei, müssen Anfragen auf Zuteilung warten
  - Threads des WS werden auf einem Rechner mit CPU und einer Platte abgearbeitet
  - WS entscheidet, welche Funktionen die jeweilige Anfrage ausführen muss
  - einige Funktionen erfordern Aufruf von Funktionen des Anwendungs-Servers (AS)
    - AS besteht aus n Threads, die auf einem Rechner mit CPU und einer Platte ablaufen
    - falls ein Thread frei ist, wird Anfrage sofort abgearbeitet

(der Thread im WS bleibt dazu ebenfalls belegt!)

* falls kein Thread frei ist, muss Anfrage warten (auch dabei bleibt der Thread im WS belegt)
* einige Anfragen an AS benötigen Zugriffe auf die Datenbank, die durch den Datenbank-Server (DS) bearbeitet werden
  - DS besteht aus p Threads, die auf einem Rechner mit zwei Platten ablaufen
  - AS und DS laufen auf dem selben Rechner, d.h. sie konkurrieren um die CPU, aber greifen auf jeweils eigene Platten zu (nicht sichtbar in der graphischen Darstellung)
  - wie zuvor werden Anfragen von BS abgearbeitet, wenn ein freier Thread vorhanden ist (dabei belegt die Anfrage simultan jeweils einen Thread von WS, AS und DS)
* nach Abarbeitung wird der DS Thread freigegeben und der AS Thread fährt mit seiner Bearbeitung fort
* der AS Thread generiert weitere DB Abfragen oder beendet die Bearbeitung

– nach Beendigung der Bearbeitung durch den AS Thread geht Kontrolle an WS Thread der Bearbeitung der Anfrage beendet oder neue AS Anfrage startet

Berechnung der Antwortzeit einer Anfrage:

$$Antwortzeit :=$$

$$SoftwareWartezeit + Ausfuehrungszeit$$

mit

$$SoftwareWartezeit :=$$

$$Wartezeit_{WS} + Wartezeit_{AS} + Wartezeit_{DS}$$

$$Ausfuehrungszeit :=$$

$$HardwareWartezeit + Bedienzeit$$

$$HardwareWartezeit :=$$

$$HwWartezeit_{WS} + HwWartezeit_{AS} + HwWartezeit_{DS}$$

$$Bedienzeit :=$$

$$Bedienzeit_{WS} + Bedienzeit_{AS} + Bedienzeit_{DS}$$

Zeiten mit bisherigen Modellen nicht ermittelbar,
da Abhängigkeiten über verschiedene Ebenen existieren

Analyseansätze

- *Layered Queueing Networks*

- Simulation

Layered Queueing Networks (LQNs)

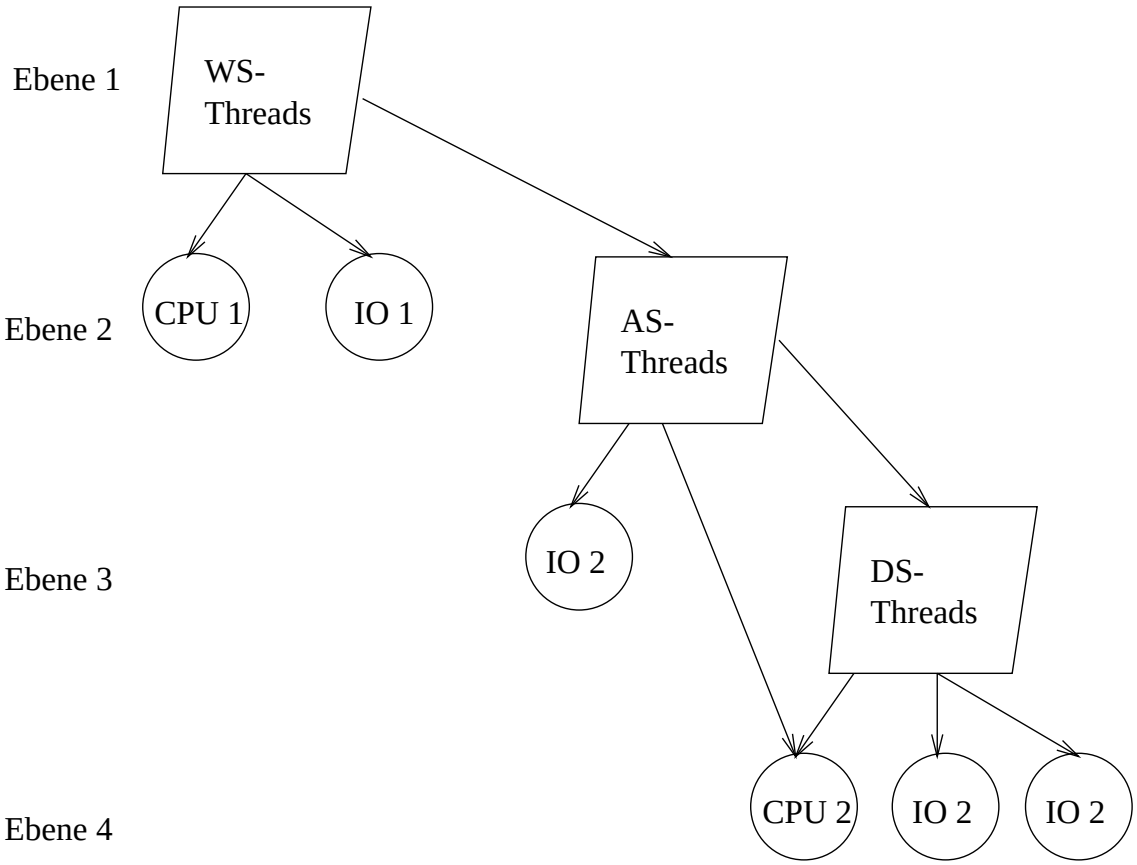
neue Entwicklung basierend auf

- Beschreibung eines Systems in Ebenen
- jede Ebene lässt sich als erweitertes Warteschlangennetz beschreiben
- Ebenen bestehen aus
  - Ressourcen (z.B. CPU, Platte) durch Warteschlangen modelliert
  - Software-Threads auch durch Warteschlangen modelliert
    - der darunterliegenden Ebenen
    - der darüber liegenden Ebenen
- bei der Analyse ist zu beachten, dass
  - Warteschlangen für Hardware-Ressourcen vollständig beschrieben sind
  - Warteschlangen für Software-Ressourcen (Threads) Bedienzeiten haben, die aus den Modellen der unteren/oberen Ebenen resultieren

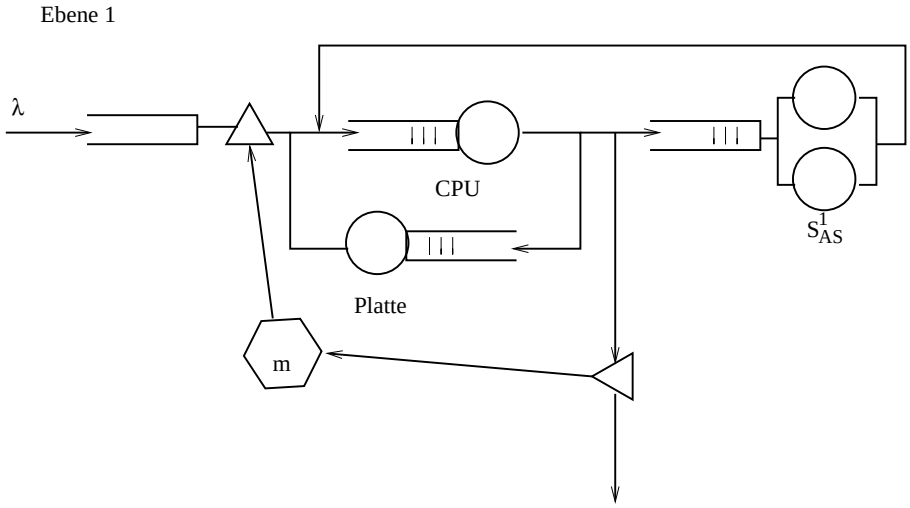
$\Rightarrow$ iterativer Fixpunktansatz zur Analyse

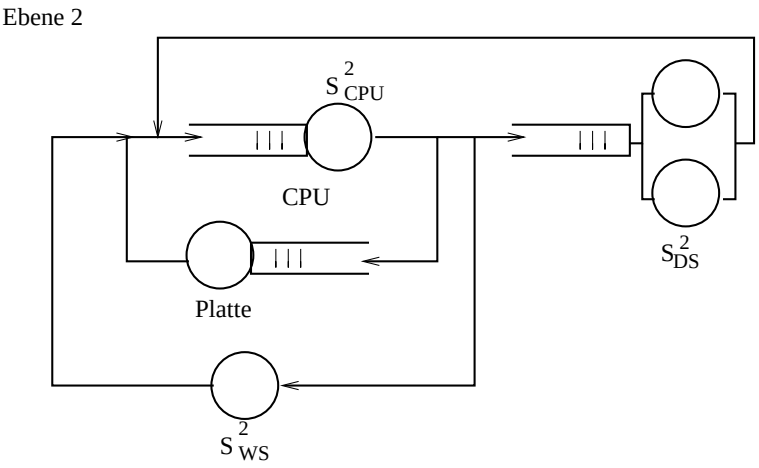
Modelle dieser Art existieren in unterschiedlichen Ausprägungen, hier nur kurze Vorstellung des Ansatzes am Beispiel (Details in der Originalliteratur)

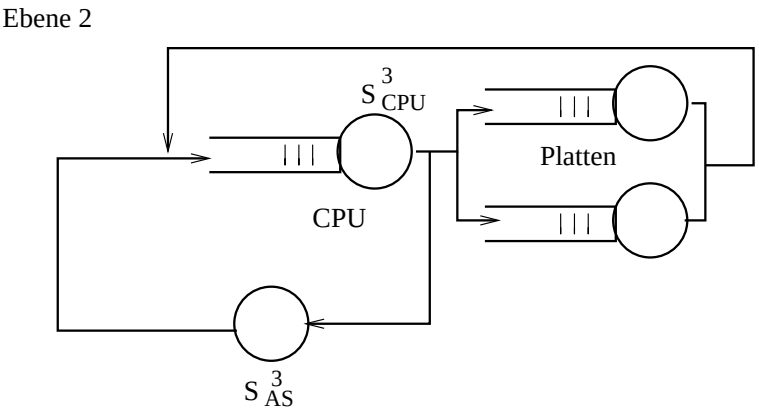
Darstellung des Beispiele als LQN



Resultierende Warteschlangennetze







Beziehungen zwischen den einzelnen Parametern

(n Threads in Ebene 2 und p Threads in Ebene 3)

Sei X_j^i = Durchsatz durch j in Ebene i

U_j^i = Auslastung von j in Ebene i

Berechnung der Bedienzeiten (Annahme $m \geq n \geq p$)

$$S_{AS}^1 = n/X_{WS}^2 - S_{WS}^2 \text{ und } S_{WS}^2 = m/X_{AS}^1 - S_{AS}^1$$

$$S_{AS}^3 = n/X_{DS}^2 - S_{DS}^2 \text{ und } S_{DS}^2 = p/X_{AS}^3 - S_{AS}^3$$

Bestimmung der Bedienzeit an der geteilten CPU

$$S^2_{CPU} = S_{CPU_2}/(1 - U^3_{CPU})$$

(S_{CPU_2} Bedienbedarf eines einzelnen Auftrags der Ebene 2 an der CPU)

$$S^3_{CPU} = S_{CPU_3}/(1 - U^2_{CPU})$$

(S_{CPU_3} Bedienbedarf eines einzelnen Auftrags der Ebene3 an der CPU)

Berechnung iterativ, wobei initial Werte geschätzt werden

- in der Regel schnelle Konvergenz der Iteration
  (dies ist aber nicht gesichert)
- in der Regel akzeptable Genauigkeit
  (auch dies ist nicht gesichert)

andere Möglichkeiten der iterativen Parameterbestimmung existieren

Modelle können deutlich komplexer sein

- mehrere Threads in einer Ebene
- Interaktion über mehrere Ebene

Lösungsansatz trotzdem anwendbar

Simulation

- Nachbildung des Systemverhalten durch ein Programm
- statistische Auswertung der Resultate

$\Rightarrow$ im Prinzip sind beliebig komplexe Systeme modellierbar

Vorteile der Simulation

- komplexe Abhängigkeiten sind darstellbar
- Verteilungen, nicht nur Mittelwerte, sind in die Analyse einbeziehbar
- auch zeitabhängiges Verhalten analysierbar

Nachteile der Simulation

- höherer Aufwand bei der Modellerstellung
  (Modelle des beschriebenen Typs mit den meisten Simulationswerkzeugen nicht direkt spezifizierbar)
- hoher Aufwand bei der Datenerhebung
  (detaillierte Modelle erfordern detaillierte Daten)
- hoher Analyseaufwand

$\Rightarrow$ für die Kapazitätsplanung sind in fast allen Fällen analytische Modelle vorzuziehen

3.8.5 Schrankenberechnung

Oft sind

- Systeme sehr groß
- Parameter eines Systems noch nicht vollständig bekannt
- viele Parameter vorhanden, die in Analysen variiert werden müssen

Dann ist eine vollständige Analyse unmöglich,
es können aber Schranken berechnet werden, um

- erste Aussagen über die Leistungsfähigkeit zu bekommen
- Grenzen für mögliche Parameter festzulegen
- interessante von uninteressanten Konfigurationen zu unterscheiden

Zahlreiche Methoden zur Schrankenberechnung existieren

- im wesentlichen für Einklassennetze
- ohne lastabhängige Stationen
- im wesentlichen zur Durchsatzberechnung

Einklassennetze

- N Aufträge im Netz
- Netz mit $K+1$ Stationen
- Stationen $1 \ldots K$ lastunabhängige Stationen
- Station $K+1$ Verzögerungsstation mit Bedienzeit Z
  (falls mehrere Verzögerungsstationen im Netz, diese erst zu einer zusammenfassen durch Addition der einzelnen Bedienzeiten)

Es gelte $D_1 \leq D_2 \leq \ldots \leq D_K$ (mit $D_k = V_k \cdot S_k$)

Station K ist der Flaschenhals des Netzes

auf Basis der Berechnungen gilt auch

$n_K(N) \geq n_k(N)$ für alle $k < K$ und alle N

- durchschnittliche Bedienzeit $D_{av} = \sum_{k=1}^{K} D_k/K$
- Summe der Bedienzeiten $D_\Sigma = \sum_{k=1}^{K} D_k$
- Zykluszeit für einen Auftrag $R(N) = N/X_0(N)$

Asymptotische Schranken

einfachste Form durch Betrachtung von Extremsituationen

1. An keiner Station wird gewartet
   - der Flaschenhals des Netzes voll ausgelastet
     obere Schranke für den Durchsatz

     $$X_0(N) \leq \min(N/(D_\Sigma + Z), 1/D_K)$$

     Herleitung:
     erster Teil: Kehrwert der minimalen Durchlaufzeit
     ist maximaler Durchsatz pro Auftrag
     zweiter Teil: maximaler Durchsatz am Flaschenhals
     liefert Durchsatzschranke
2. Auftrag wartete an jeder Station auf $N - 1$ Aufträge
   - untere Schranke für den Durchsatz

     $$X_0(N) \geq N/(N \cdot D_\Sigma + Z)$$

     Herleitung:
     Kehrwert der maximalen Zykluszeit liefert
     minimalen Durchsatz pro Auftrag

Schranken für Zykluszeit über Little

Schranken in der Regel sehr weit auseinander

Balanced job bounds

Schranken durch Anwendung der Mittelwertanalyse

Basis: System, in dem alle Stationen gleich ausgelastet sind (*balanced*) liefert bessere Leistung als System mit ungleicher Auslastung (*unbalanced*)

(bessere Leistung bedeutet höheren Durchsatz geringere Zykluszeit)

untere Schranke für $R(N)$/ obere Schranke für $X(N)$

Untersuchung des *balanced* Systems liefert

- mittlere Kundenzahl an der Verzögerungsstation n_Z

$$\frac{Z}{Z+(N-1)D_\Sigma} \leq n_Z \leq \frac{Z}{Z+D_\Sigma}$$

- mittlere Kundenzahl an den restlichen Stationen n_Σ

$$\frac{D_\Sigma}{Z+D_\Sigma} \leq n_\Sigma \leq \frac{(N-1)D_\Sigma}{Z+(N-1)D_\Sigma}$$

Damit gilt (wg. besserer Leistung *balanced* System)

$$R(N) \geq Z + D_\Sigma + (N-1) \cdot D_{av} \cdot \frac{D_\Sigma}{D_\Sigma+Z}$$

und

$$X_0(N) \leq N/(Z + D_\Sigma + (N-1) \cdot D_{av} \cdot \frac{(N-1)\cdot D_\Sigma}{(N-1)\cdot D_\Sigma+Z})$$

obere Schranke für $R(N)$/untere Schranke für $X(N)$

Konstruktion eines $balanced$ Systems

mit schlechterer Leistung als Originalsystem hat

Schrittweises Vorgehen:

1. Setze $l^+ = \max_k(D_k < D_K)$, $l^- = \min_k(D_k > 0)$
2. Falls $l^+ < l^-$ STOP
3. Reduziere Bedienzeit an Station l^- um Δ und
   erhöhe Bedienzeit an Station l^+ um Δ
   wobei $\Delta = \min(D_{l^-}, D_K - D_{l^+})$

Schritt 3 reduziert die Systemleistung $\Rightarrow$

resultierendes Netz hat schlechtere Leistung

als Ausgangsnetz

Stationen mit Bedienzeit 0 können gestrichen werden

Es entsteht ein *balanced* Netz mit

$K' = \lceil D_\Sigma / D_K \rceil$ Stationen mit Bedienzeit D_K

Zykluszeit dieses Netzes obere Schranke für Ausgangsnetz

$$R(N) \leq Z + D_\Sigma + (N-1) \cdot D_K \cdot \frac{(N-1)D_\Sigma}{(N-1)D_\Sigma + Z}$$

und

$$X_0(N) \geq N/(Z + D_\Sigma + (N-1) \cdot D_K \cdot \frac{(N-1)D_\Sigma}{(N-1)D_\Sigma + Z})$$

3.8.6 Grenzen der Modellierung mit Warteschlangennetzen

Warteschlangennetze werden breit eingesetzt

aber es gibt klare Restriktionen bzgl.

- der Spezifizierbarkeit bestimmter Verhaltensmuster
- der analytischen Lösbarkeit

Probleme bei der Spezifikation bereitet

- simultane Ressourcenbelegung
- *fork/join* von Prozessen
- manche Blockierstrategien
- komplexe lastabhängige Routingstrategien
- Synchronisation von Prozessen
- exklusiver Zugriff auf mehrere Ressourcen
- manche Schedulingstrategien wie *shortest job first*

Probleme bei der analytischen Analyse

- nicht exponentielle Bedien- oder Ankunftszeiten
- *batch* oder *bulk* Ankünfte oder Bedienung
- zeitanhängige Analysen